EDA_and_models_candice

Candice Yu

2024-03-23

Contents

```r
library(caret)
library(earth)
library(tidyverse)
library(gridExtra)
```

Load Data

```r
load("./Data/recovery.RData")
dat <- dat %>%
  mutate(gender = as_factor(gender),
         diabetes = as_factor(diabetes),
         hypertension = as_factor(hypertension),
         vaccine = as_factor(vaccine),
         severity = as_factor(severity)) %>%
  select(-id)
```

EDA

Overview of the Data

```r
# brief summary of the data
skimr::skim(dat)
```

Table 1: Data summary

Name	dat
Number of rows	3000
Number of columns	15
Column type frequency:	
character	1
factor	7
numeric	7
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
study	0	1	1	1	0	2	0

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
gender	0	1	FALSE	2	0: 1544, 1: 1456
race	0	1	FALSE	4	1: 1967, 3: 604, 4: 271, 2: 158
smoking	0	1	FALSE	3	0: 1822, 1: 859, 2: 319
hypertension	0	1	FALSE	2	0: 1508, 1: 1492

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
diabetes	0	1	FALSE	2	0: 2537, 1: 463
vaccine	0	1	FALSE	2	1: 1788, 0: 1212
severity	0	1	FALSE	2	0: 2679, 1: 321

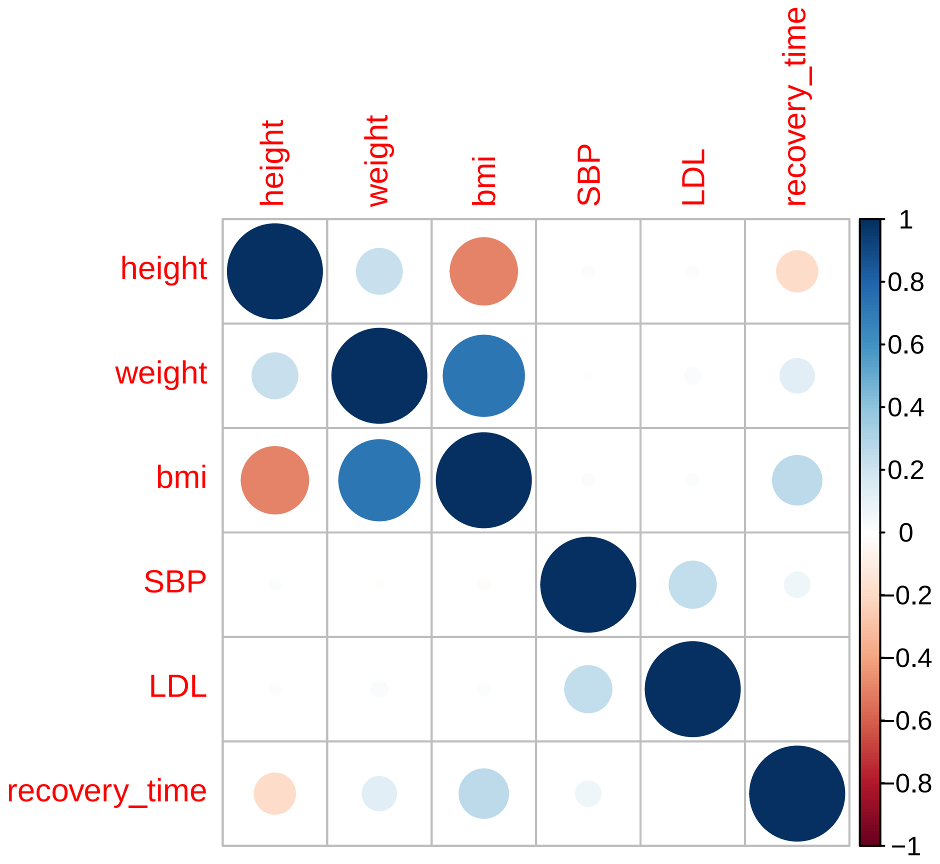
Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
age	0	1	60.20	4.48	42.0	57.0	60.00	63.0	79.0	
height	0	1	169.90	5.97	147.8	166.0	169.90	173.9	188.6	
weight	0	1	79.96	7.14	55.9	75.2	79.80	84.8	103.7	
bmi	0	1	27.76	2.79	18.8	25.8	27.65	29.5	38.9	
SBP	0	1	130.47	7.97	105.0	125.0	130.00	136.0	156.0	
LDL	0	1	110.45	19.76	28.0	97.0	110.00	124.0	178.0	
recovery_time	0	1	42.17	23.15	2.0	31.0	39.00	49.0	365.0	

EDA for Continuous Variables

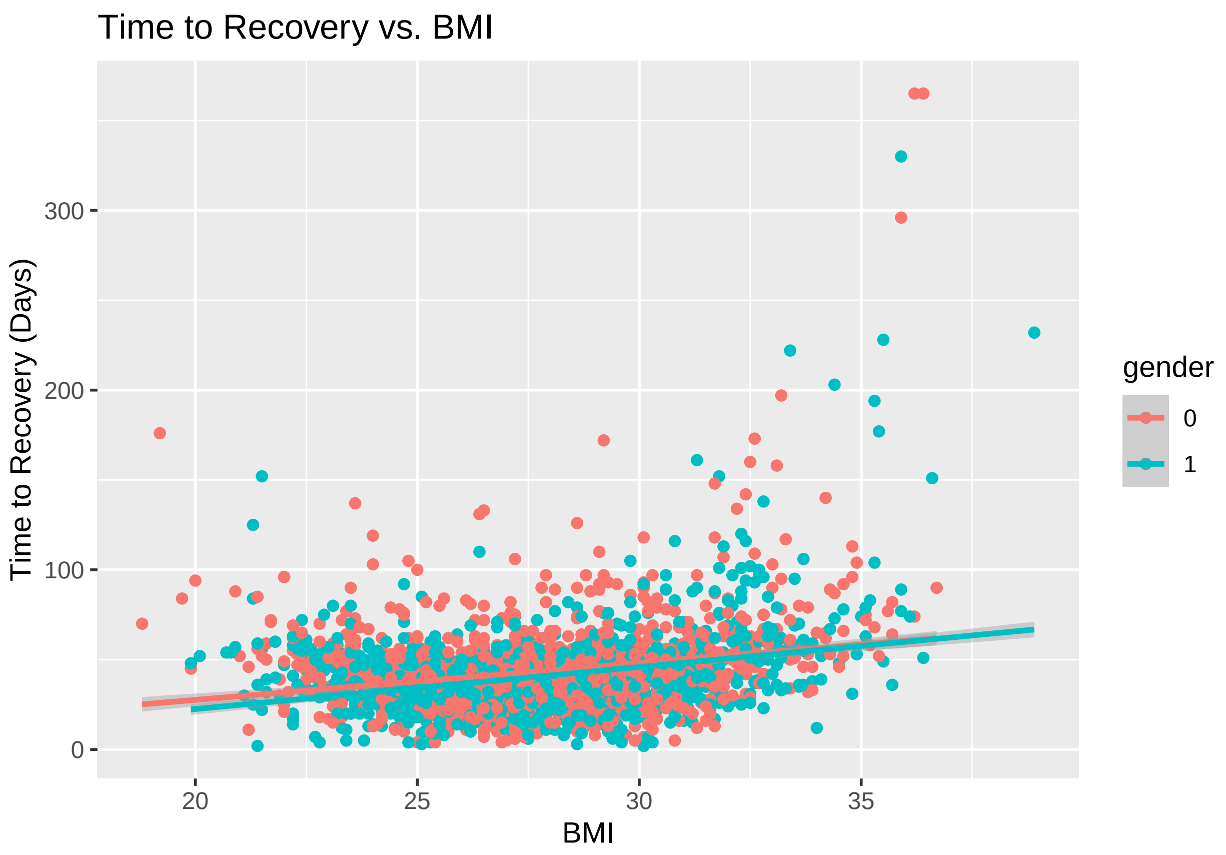
Correlation plot for continuous variables

```
# correlation plot for continuous variables
continuous_vars <- dat %>%
  select(height, weight, bmi, SBP, LDL, recovery_time)
correlations <- cor(continuous_vars)
corrplot::corrplot(correlations, method = "circle")
```

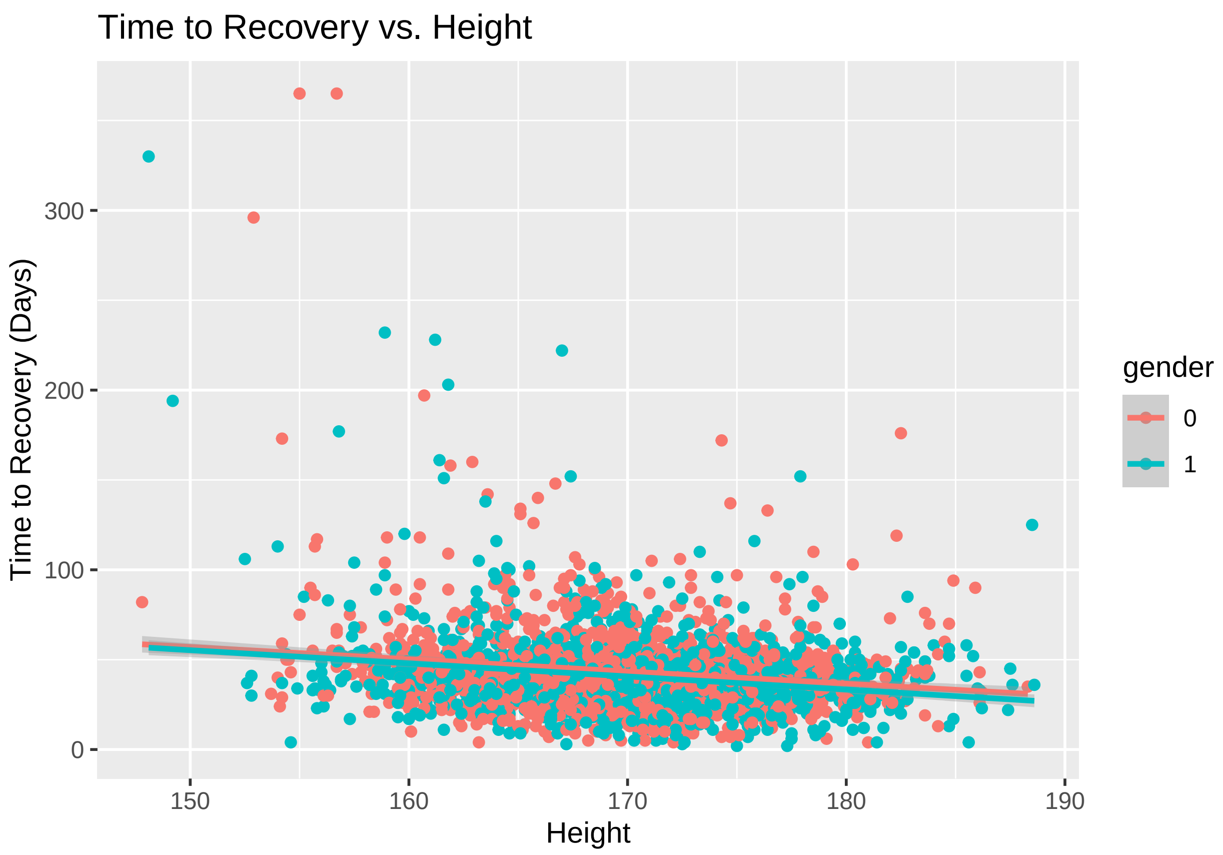


Scatter plots to explore potential relationships

```r
# between time to recovery and bmi
ggplot(dat, aes(x = bmi, y = recovery_time, color = gender)) +
  geom_point() + geom_smooth(method = "lm") +
  labs(title = "Time to Recovery vs. BMI",
       x = "BMI",
       y = "Time to Recovery (Days)")
```



```
# between time to recovery and height
ggplot(dat, aes(x = height, y = recovery_time, color = gender)) +
  geom_point() + geom_smooth(method = "lm") +
  labs(title = "Time to Recovery vs. Height",
       x = "Height",
       y = "Time to Recovery (Days)")
```

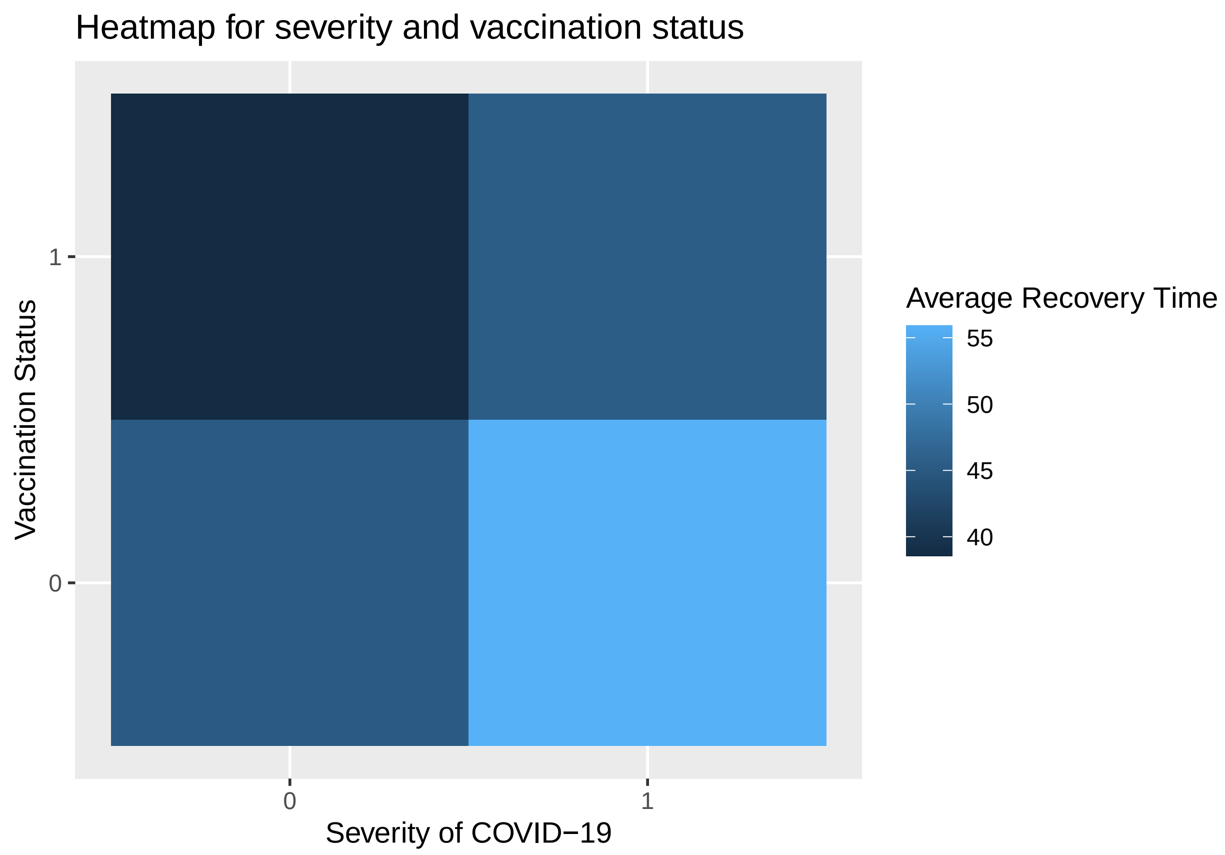


The correlation plot and scatter plots show some relationships between continuous variables, but none of them appear to be strongly correlated with `recovery_time`. This may suggest that linear relationships are not strong, and hence a non-linear model could be more appropriate.

EDA for Discrete Variables

Heatmap for severity and vaccination status

```
# Heatmap for systolic blood pressure across severity and vaccination status
dat %>%
  group_by(severity, vaccine) %>%
  summarise(avg_recovery_time = mean(recovery_time)) %>%
  ggplot(aes(x = factor(severity), y = factor(vaccine), fill = avg_recovery_time)) +
  geom_tile() +
  labs(title = "Heatmap for severity and vaccination status",
       x = "Severity of COVID-19",
       y = "Vaccination Status",
       fill = "Average Recovery Time")
```



The heatmap helps in understanding the bivariate relationship between severity, vaccination status, and recovery time.

Observations from the Heatmap:

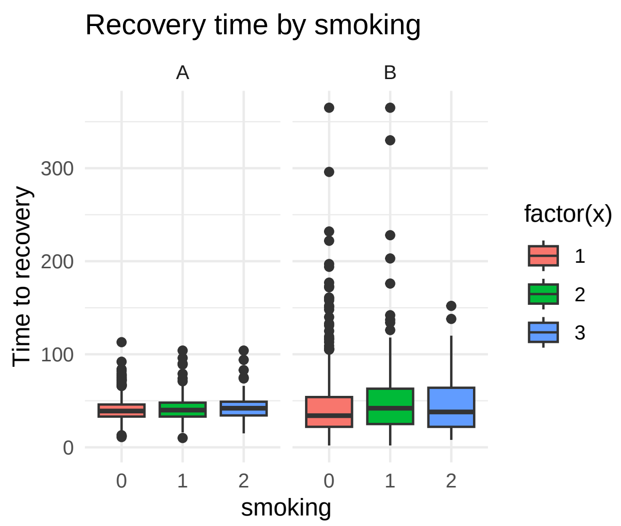
- Individuals with severe COVID-19 infection (1 on the x-axis) have longer average recovery times than those with non-severe infections, regardless of vaccination status.
- Vaccination status seems to have an influence on the recovery time. Those who are vaccinated (1 on the y-axis) tend to have shorter recovery times even when the infection is severe.

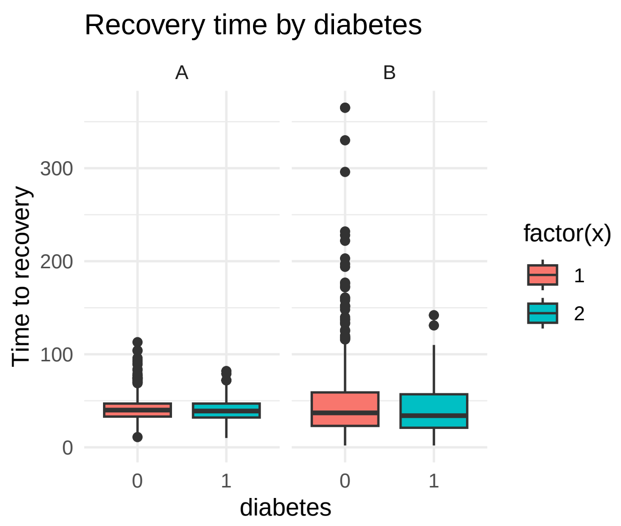
Implications for Modeling:

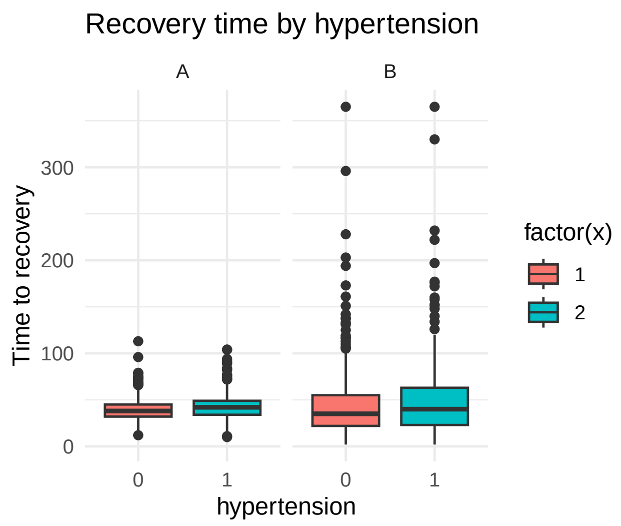
- The heatmap suggests there might be an interaction effect between severity and vaccination status on the recovery time. Therefore, when modeling, consider including an interaction term between these two variables.
- Given the apparent differences in recovery time across the groups, both severity and vaccination status should be included as important predictors in the model.
- If developing separate models for different subgroups is a consideration, you might want to stratify the analysis by severity or vaccination status.

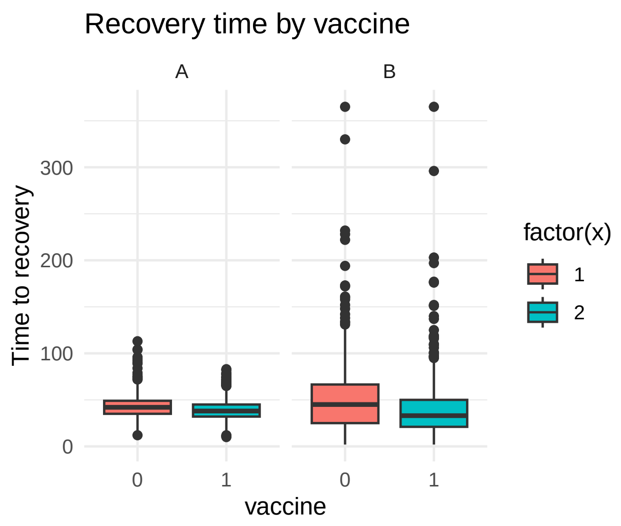
Faceted grid plot for categorical variables

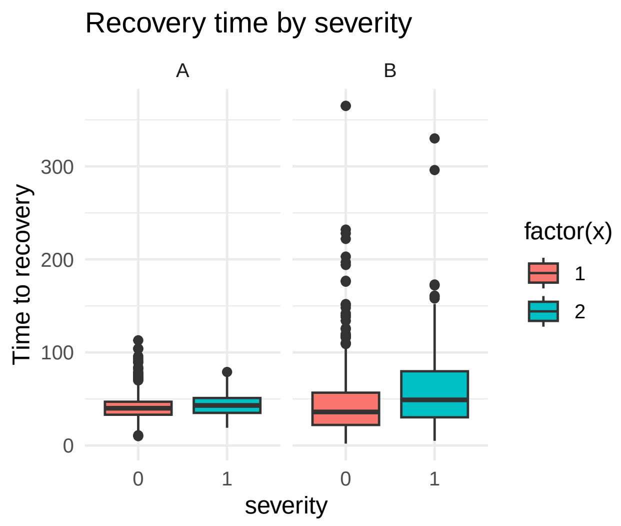
```r
# faceted grid plot for categorical variables
categorical_vars <- c("smoking", "diabetes", "hypertension", "vaccine", "severity")
faceted_plots <- lapply(categorical_vars, function(var) {
  ggplot(dat, aes_string(x = var, y = "recovery_time")) +
    geom_boxplot(aes(fill = factor(..x..))) +
    facet_wrap(~study) +
    labs(title = paste("Recovery time by", var), y = "Time to recovery") +
    theme_minimal()
})

# combine the plots into one grid
grid.arrange(scatter_bmi, scatter_sbp, scatter_ldl, grobs = faceted_plots, ncol = 2)
```











The boxplots indicate a significant difference in recovery times between study groups A and B across several categorical factors, which suggests that `study` is an important variable to include in the model.

Preprocess of the Data

```
data <- dat %>%
  select(-weight, -height)

# normalize/standardize numerical variables
#num_vars <- names(data)[sapply(data, is.numeric)][-7]
#preprocess_params <- preProcess(data[, num_vars], method = c("center", "scale"))
#data[num_vars] <- predict(preprocess_params, data[, num_vars])

# log transform 'recovery_time' since it's highly skewed
#data$recovery_time <- log(data$recovery_time)
```

Split data & Define the control method

```
# split data into training and test sets
set.seed(2716)
indexes <- createDataPartition(data$recovery_time, p = 0.8, list = FALSE)
train_data <- data[indexes, ]
test_data <- data[-indexes, ]

# matrix of predictors
x <- train_data %>% select(-recovery_time)
y <- train_data$recovery_time

# define the control method for training
ctrl1 <- trainControl(method = "cv", number = 10) # 10-fold cross-validation
```

Model Training Procedure and Final Model:

1. Data Splitting: The dataset was split into training (80%) and test (20%) sets using a stratified random sampling approach based on `recovery_time`.
2. The `train` function from the `caret` package was used to train the MARS model using 10-fold cross-validation. This approach helps to prevent overfitting and gives an estimate of the model performance on new data.
3. The model with the lowest cross-validated Root Mean Squared Error (RMSE) was selected as the final model.

Export the the training/test set & control method

```
# save the training and test sets to CSV files
write.csv(train_data, "./Data/train_data.csv", row.names = FALSE)
write.csv(test_data, "./Data/test_data.csv", row.names = FALSE)

# save the control method using saveRDS
saveRDS(ctrl1, "./Data/train_control.rds")
```

Load the training/test set & control method

```
# Load the training and test sets
train_data <- read.csv("./Data/train_data.csv")
test_data <- read.csv("./Data/test_data.csv")
```

```r
# Load the control method
ctrl1 <- readRDS("./Data/train_control.rds")

# change variables to be factors again
train_data <- train_data %>%
  mutate(gender = as_factor(gender),
         diabetes = as_factor(diabetes),
         hypertension = as_factor(hypertension),
         vaccine = as_factor(vaccine),
         race = as_factor(race),
         smoking = as_factor(smoking),
         severity = as_factor(severity))

test_data <- test_data %>%
  mutate(gender = as_factor(gender),
         diabetes = as_factor(diabetes),
         hypertension = as_factor(hypertension),
         race = as_factor(race),
         smoking = as_factor(smoking),
         vaccine = as_factor(vaccine),
         severity = as_factor(severity))
```

Load the training/test set & control method

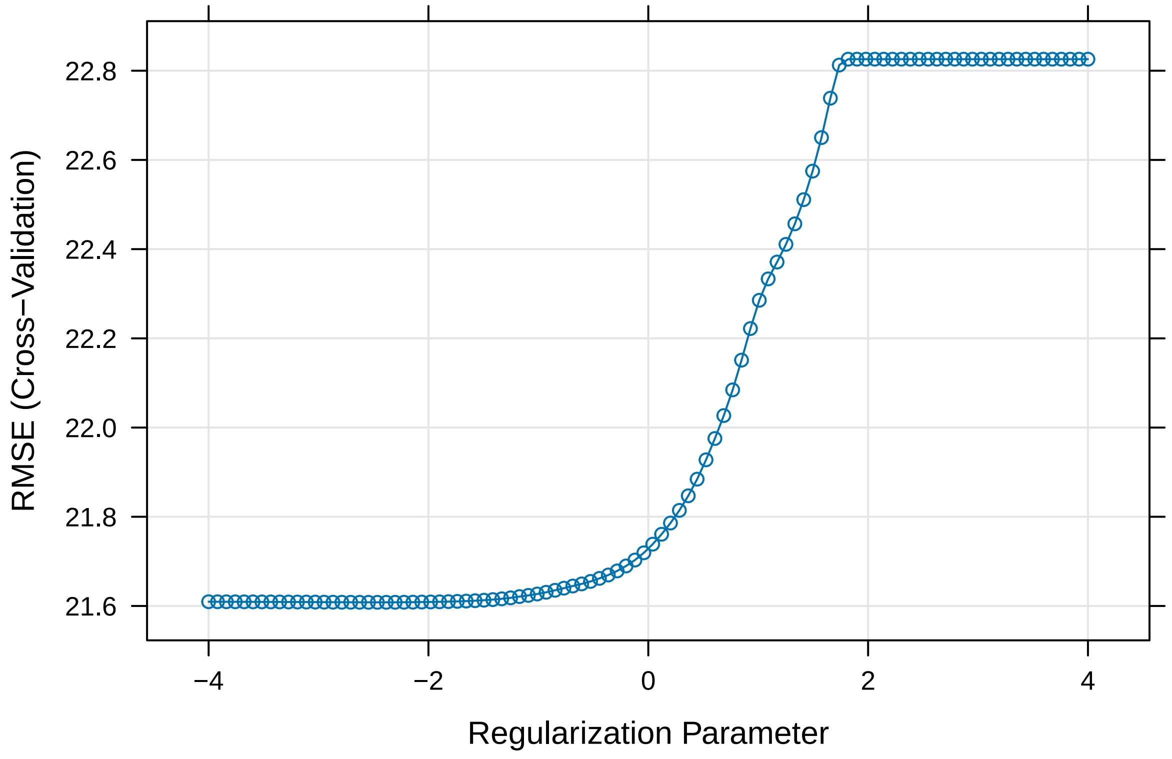
```r
# Load the training and test sets
train_data <- read.csv("./Data/train_data.csv")
test_data <- read.csv("./Data/test_data.csv")

set.seed(2716)
# Load the control method
ctrl1 <- trainControl(method = "cv", number = 10) # 10-fold cross-validation

# change variables to be factors again
train_data <- train_data %>%
  mutate(gender = as_factor(gender),
         diabetes = as_factor(diabetes),
         hypertension = as_factor(hypertension),
         vaccine = as_factor(vaccine),
         race = as_factor(race),
         smoking = as_factor(smoking),
         severity = as_factor(severity))

test_data <- test_data %>%
  mutate(gender = as_factor(gender),
         diabetes = as_factor(diabetes),
         hypertension = as_factor(hypertension),
         race = as_factor(race),
         smoking = as_factor(smoking),
         vaccine = as_factor(vaccine),
         severity = as_factor(severity))
# matrix of predictors
x <- train_data %>% select(-recovery_time)
```

```r
y <- train_data$recovery_time

x_test <- test_data %>% select(-recovery_time)
y_test <- test_data$recovery_time
```

Model Training: Linear models

Lasso Regression Model

```r
lasso_grid <- expand.grid(
  alpha = 1,
  lambda = exp(seq(-4, 4, length.out = 100))
)

set.seed(2716)

lasso_fit <- train(recovery_time ~ .,
                   data = train_data,
                   method = "glmnet",
                   tuneGrid = lasso_grid,
                   trControl = ctrl1,
                   preProcess = c("center", "scale")
)

plot(lasso_fit, xTrans = log)
```



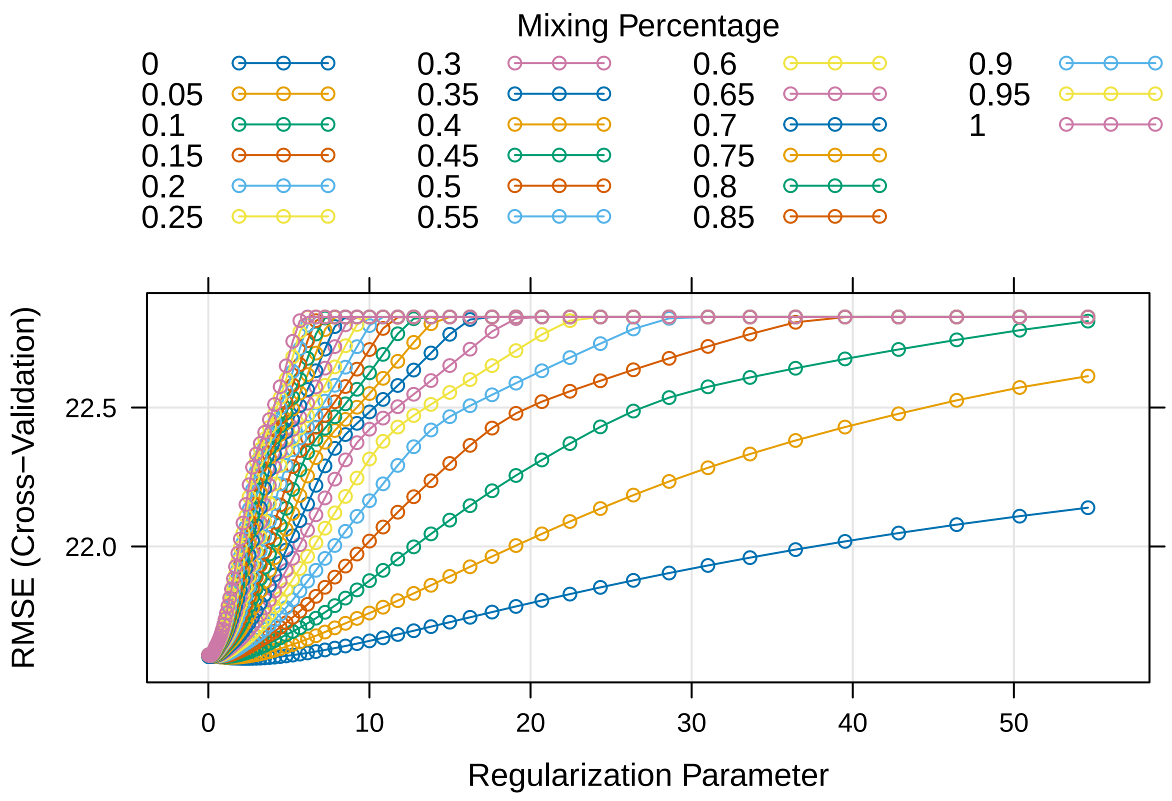
```r
# Get the index of the model with the lowest RMSE
best_model_index <- which.min(lasso_fit$results$RMSE) # Get the coefficients of the optimal model
```

```r
optimal_model_coeffs <- coef(lasso_fit$finalModel,
                                s = lasso_fit$results$lambda[best_model_index])
# Print the coefficients
print(optimal_model_coeffs)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept)   42.2564530
## age            0.5535977
## gender1       -1.2647392
## race2          0.2757148
## race3         -0.1749597
## race4         -0.5370051
## smoking1       0.8024902
## smoking2       0.7445695
## bmi            5.4520392
## hypertension1  1.1692823
## diabetes1     -0.6813186
## SBP            0.8393015
## LDL           -0.7012207
## vaccine1      -2.7709182
## severity1      2.5406012
## studyB         2.5403617
```

Elastic Net Model

```r
set.seed(2716)
enet_grid <- expand.grid(
  alpha = seq(0, 1, length.out = 21),
  lambda = exp(seq(-4, 4, length.out = 100))
)

enet_fit <- train(recovery_time ~ .,
                  data = train_data,
                  method = "glmnet",
                  tuneGrid = enet_grid,
                  trControl = ctrl1,
                  preProcess = c("center", "scale")
)
plot(enet_fit)
```

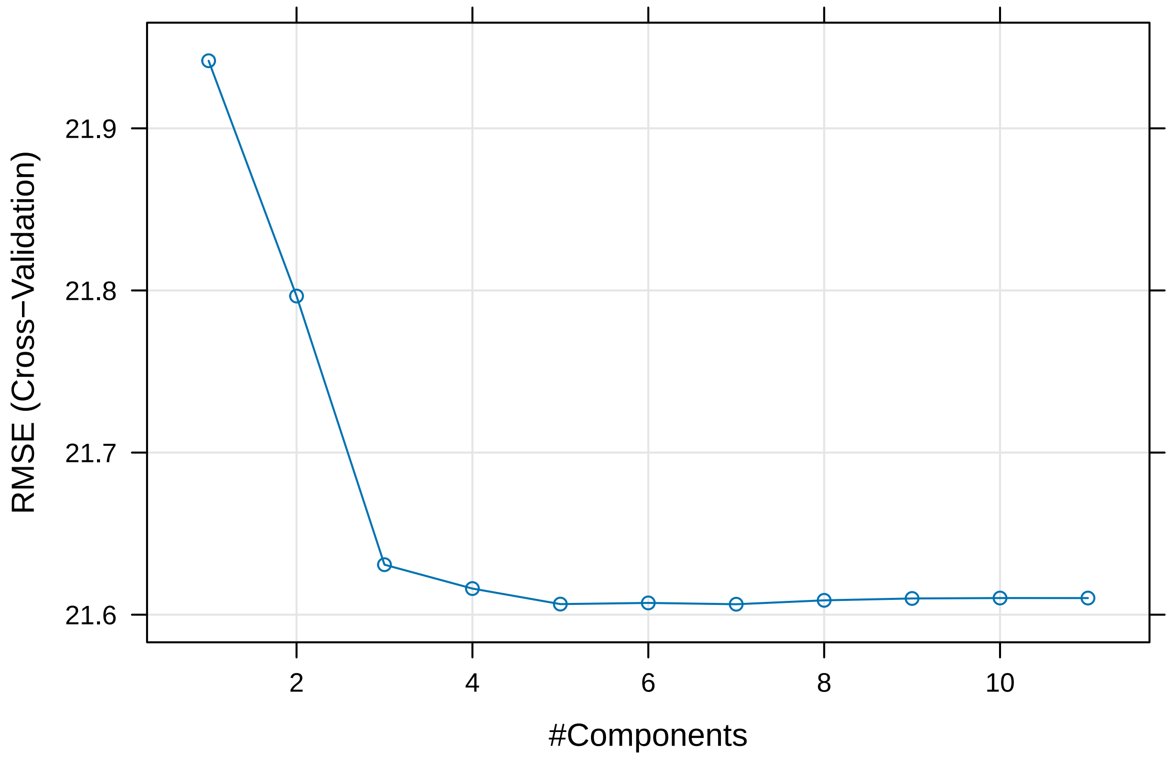


```r
# Get the index of the model with the lowest RMSE
best_model_index <- which.min(enet_fit$results$RMSE) # Get the coefficients of the optimal model
optimal_model_coeffs <- coef(enet_fit$finalModel,
                             s = enet_fit$results$lambda[best_model_index])
# Print the coefficients
print(optimal_model_coeffs)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                         s1
## (Intercept)   42.2564530
## age            0.6045153
## gender1       -1.2367352
## race2          0.3276704
## race3         -0.2377269
## race4         -0.5657359
## smoking1       0.8149331
## smoking2       0.7479438
## bmi            5.0341688
## hypertension1  1.0958328
## diabetes1     -0.7061193
## SBP            0.8578848
## LDL           -0.7089821
## vaccine1      -2.6026293
## severity1      2.3966295
## studyB         2.3902154
```

Partial Least Squares

```r
set.seed(2716)

pls_fit <- train(x, y,
                 method = "pls",
                 tuneLength = 20,
                 trControl = ctrl1,
                 preProcess = c("center", "scale")
                 )

plot(pls_fit)
```



Evaluate the performance of linear models

```r
lasso_pred <- predict(lasso_fit, newdata = x_test)
enet_pred <- predict(enet_fit, newdata = x_test)
pls_pred <- predict(pls_fit, newdata = x_test)

lasso_performance <- postResample(pred = lasso_pred, obs = test_data$recovery_time)
lasso_performance
```

```
##       RMSE   Rsquared        MAE
## 21.6024677  0.1637076 12.8215166
```

```r
enet_performance <- postResample(pred = enet_pred, obs = test_data$recovery_time)
enet_performance
```

```
##       RMSE   Rsquared        MAE
## 21.6530528  0.1628637 12.7871753
```

```
pls_performance <- postResample(pred = pls_pred, obs = test_data$recovery_time)
pls_performance
```

```
##       RMSE   Rsquared        MAE 
## 21.5903221  0.1631082 12.8407993
```

Model Training: Nonlinear Methods

The EDA plots show that the relationship between predictors and recovery time is likely non-linear, and there may be interactions between variables, especially considering the difference between study groups A and B.

Given the results from the EDA plots and the nature of the data, both generalized additive models (GAM) and multivariate adaptive regression splines (MARS) could be suitable choices for modeling. They both are capable of modeling complex, non-linear relationships in the data.

Multivariate Adaptive Regression Spline (MARS)

Build the MARS model

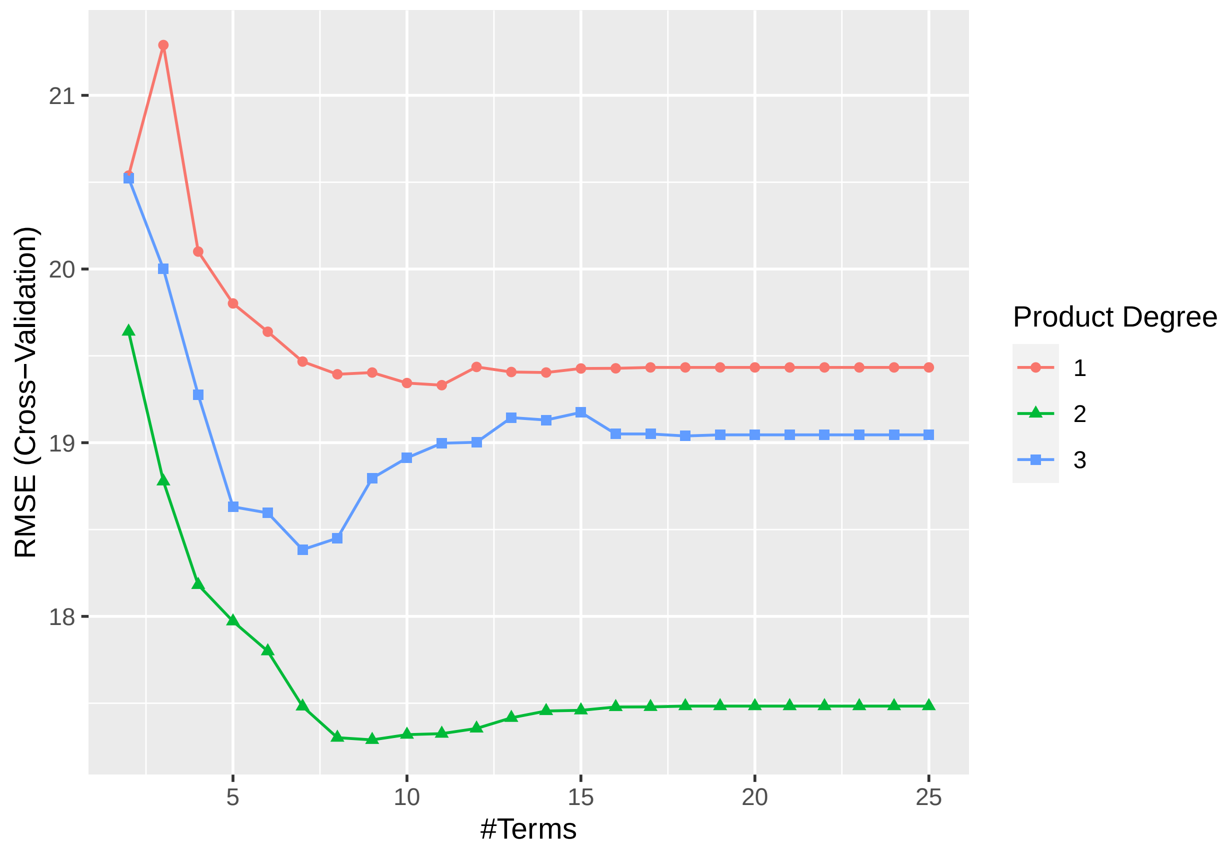
```
# train the MARS model
mars_grid <- expand.grid(degree = 1:3, nprune = 2:25)

set.seed(2716) # set the same seed
mars_fit <- train(x, y,
                    method = "earth",
                    tuneGrid = mars_grid,
                    trControl = ctrl1)
```

MARS Model Summary

```
# Model summary
summary(mars_fit)
```

```
## Call: earth(x=data.frame[2402,12], y=c(44,29,40,31,5...), keepxy=TRUE,
##             degree=2, nprune=9)
## 
##                         coefficients
## (Intercept)                 9.358604
## gender1                    -3.292896
## hypertension1               3.841507
## vaccine1                   -5.668675
## h(bmi-25.1)                 6.146988
## h(30.5-bmi)                 5.345431
## severity1 * studyB         16.973744
## h(bmi-30.5) * studyB       13.128169
## h(bmi-35.2) * studyB      198.572004
## 
## Selected 9 of 21 terms, and 6 of 15 predictors (nprune=9)
## Termination condition: Reached nk 31
## Importance: bmi, studyB, severity1, vaccine1, hypertension1, gender1, ...
## Number of terms at each degree of interaction: 1 5 3
## GCV 290.93    RSS 686648.1    GRSq 0.4529171    RSq 0.4619934
```

```r
ggplot(mars_fit)
```



```r
mars_fit$bestTune
```

```
##    nprune degree
## 32      9      2
```

```r
coef(mars_fit$finalModel)
```

```
##          (Intercept)          h(30.5-bmi) h(bmi-30.5) * studyB
##             9.358604             5.345431            13.128169
##          h(bmi-25.1) h(bmi-35.2) * studyB             vaccine1
##             6.146988           198.572004            -5.668675
##        hypertension1              gender1   severity1 * studyB
##             3.841507            -3.292896            16.973744
```

MARS Model Description:

The MARS model is a flexible regression method capable of uncovering complex nonlinear relationships between the dependent variable (recovery_time) and a set of independent variables. It does this by fitting piecewise linear regressions, which can adapt to various data shapes. This is particularly useful for modeling the recovery time from COVID-19 since the relationship between predictors and recovery time could be highly nonlinear and interaction-heavy.

Assumptions:

- The relationships between predictors and the response can be captured using piecewise linear functions.
- Interactions between variables can be important and are modeled by products of basis functions.
- There is no assumption of a parametric form of the relationship between predictors and the response.

Final Model Selection:

- The optimal hyperparameters were degree (degree of interaction) = 3 and nprune (number of terms) = 16.
- The selected model terms involve interactions between patient characteristics, their biometrics, the specific study group they belong to, and some non-linear transformations of these variables.

Evaluate performance on the test set

```r
# Evaluate its performance on the test set:
predictions <- predict(mars_fit, newdata = test_data)
postResample(pred = predictions, obs = test_data$recovery_time)
```

```
##       RMSE   Rsquared        MAE
## 34.8585974  0.3524154 12.9107465
```

The results from evaluating the MARS model on the test set provide three key metrics:

1. **Root Mean Squared Error (RMSE):** RMSE measures the average magnitude of the prediction error. It represents the square root of the average squared differences between the predicted and actual values. An RMSE of 19.629 suggests that, on average, the model's predictions of the recovery time are about 19.629 days off from the actual recovery times.
2. **R-squared (R^2):** R^2 is a statistical measure that represents the proportion of the variance for the dependent variable that's explained by the independent variables in the model. In your case, the R^2 value is 0.2177, which means approximately 21.77% of the variance in the recovery time is explained by the model. This is a relatively low value, indicating that there is a lot of variability in the recovery time that is not captured by the model.
3. **Mean Absolute Error (MAE):** MAE measures the average absolute difference between the predicted values and the actual values, providing a linear score that reflects the average error magnitude without considering its direction. An MAE of 12.409 suggests that the model's predictions are, on average, 12.409 days different from the actual recovery time.

Interpretation

- The **RMSE** of 19.629 days is relatively high, depending on the context of the recovery times' range. If the typical recovery time is on the order of a few days, this is a substantial error. However, if recovery times are generally several weeks, the error may be more acceptable.
- The **R-squared** value of 0.2177 is not very high, suggesting that there might be other factors not included in the model that affect the recovery time. It also indicates that the relationship between the predictors and the recovery time has a significant amount of unexplained variability.
- The **MAE** gives us an indication that, despite the direction of the errors, the model's predictions are off by about two weeks on average. MAE is less sensitive to outliers than RMSE, so this value suggests that the model has a consistent average error across the test dataset.

GAM model

Build the GAM model

```r
set.seed(2716) # set the same seed
gam_fit <- train(x = x, y = y,
                 method = "gam",
                 trControl = ctrl1)
```

Display the summary of the final model

```r
gam_model_final <- gam_fit$finalModel
summary(gam_model_final)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ gender + hypertension + diabetes + vaccine + severity +
##     study + smoking + race + s(age) + s(SBP) + s(LDL) + s(bmi)
##
## Parametric coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   42.45538    0.96899  43.814  < 2e-16 ***
## gender1       -3.32709    0.77911  -4.270 2.03e-05 ***
## hypertension1  3.73230    0.78134   4.777 1.89e-06 ***
## diabetes1     -2.12159    1.09065  -1.945   0.0519 .
## vaccine1      -6.15159    0.79351  -7.752 1.33e-14 ***
## severity1      8.47963    1.24622   6.804 1.28e-11 ***
## studyB         4.90825    0.82831   5.926 3.56e-09 ***
## smoking1       2.23808    0.88018   2.543   0.0111 *
## smoking2       3.02769    1.30043   2.328   0.0200 *
## race2          1.11665    1.74207   0.641   0.5216
## race3          0.06618    0.99249   0.067   0.9468
## race4         -1.12042    1.42608  -0.786   0.4321
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##              edf Ref.df       F p-value
## s(age) 8.829e-07      9   0.000   0.736
## s(SBP) 4.201e-07      9   0.000   0.420
## s(LDL) 1.296e-01      9   0.016   0.285
## s(bmi) 8.924e+00      9 104.072  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.321   Deviance explained = 32.7%
## GCV = 364.15  Scale est. = 360.96    n = 2402
```

Evaluate the GAM model's performance

```r
test_predictions <- predict(gam_model_final, x_test)
postResample(pred = test_predictions, obs = test_data$recovery_time)
```

```
##       RMSE   Rsquared        MAE
## 21.5949142  0.3554629 13.4168111
```

Random Forest

Build the rf model

```
set.seed(2716)
# Parameters for Random Forest training
tunegrid <- expand.grid(mtry = 1:5)

# build the rf model
rf_fit <- train(
   x = x, y = y,
   method = "rf",
   trControl = ctrl1,
   tuneGrid = tunegrid
 )
rf_model_final <- rf_fit$finalModel
```

Evaluate the rf model's performance

```
# Calculate and print the RMSE for training and test datasets
rf_predictions <- predict(rf_model_final, x_test)
postResample(pred = rf_predictions, obs = test_data$recovery_time)
```

```
##       RMSE   Rsquared        MAE
## 18.1800886  0.4208812 12.1826118
```

Model Comparison

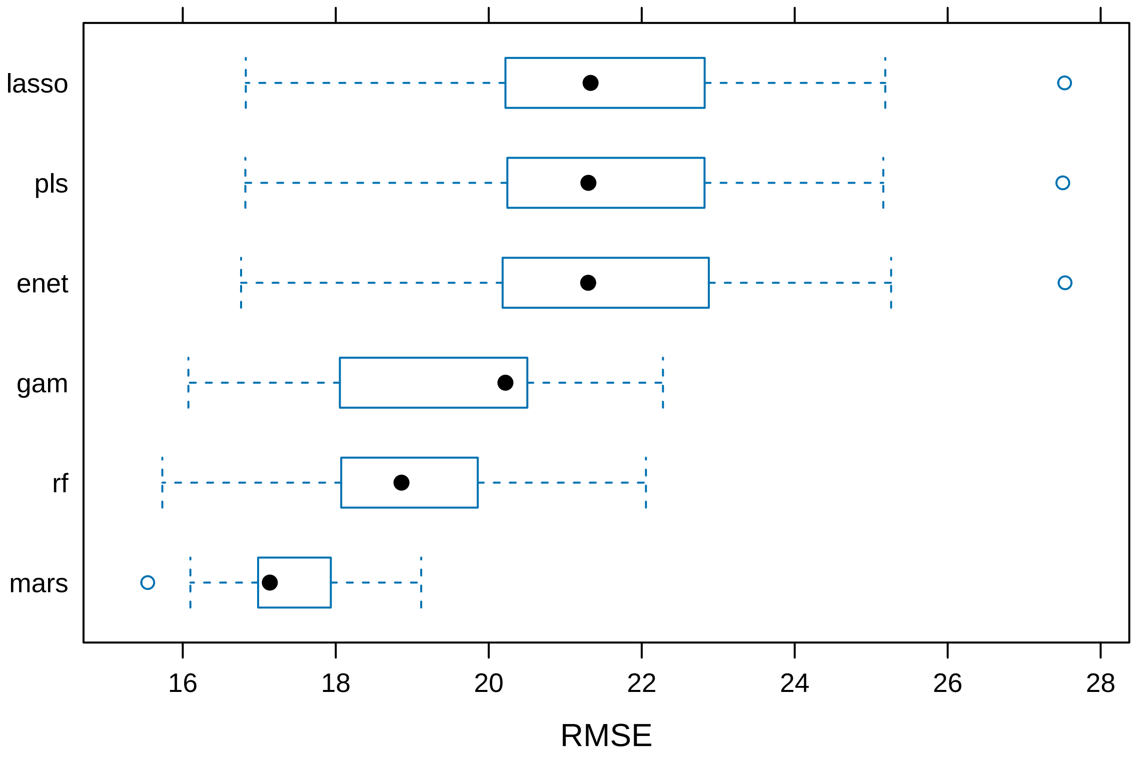
```
set.seed(2716)
resamp =
  resamples(list(lasso = lasso_fit,
                 gam = gam_fit,
                 enet = enet_fit,
                 pls = pls_fit,
                 mars = mars_fit,
                 rf = rf_fit))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: lasso, gam, enet, pls, mars, rf
## Number of resamples: 10
##
## MAE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lasso 11.77360 13.12825 13.46284 13.44561 13.85827 14.75997    0
## gam   11.42544 12.50332 12.78923 12.69601 12.95004 13.66259    0
## enet  11.69228 13.05191 13.39720 13.37765 13.78440 14.68798    0
## pls   11.78212 13.14454 13.49967 13.47720 13.88119 14.76913    0
## mars  10.99323 11.68725 11.85075 11.79140 12.04602 12.23483    0
## rf    11.06627 11.66679 12.22921 12.08414 12.37878 12.80023    0
##
```

```
## RMSE
##           Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lasso 16.82415 20.25832 21.33137 21.60820 22.53236 27.52847    0
## gam   16.07328 18.19815 20.21817 19.45737 20.49510 22.27783    0
## enet  16.76218 20.22903 21.30010 21.59700 22.57970 27.53475    0
## pls   16.81815 20.27157 21.30276 21.60646 22.53514 27.50506    0
## mars  15.54181 17.01221 17.13794 17.28887 17.81367 19.11708    0
## rf    15.73258 18.20358 18.85918 18.79321 19.63882 22.05531    0
##
## Rsquared
##             Min.    1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lasso 0.04652295 0.07524240 0.1253847 0.1120313 0.1355565 0.1804910    0
## gam   0.16842839 0.21972879 0.2804230 0.3024507 0.3985251 0.4775290    0
## enet  0.04603024 0.07509423 0.1244039 0.1125004 0.1374560 0.1803765    0
## pls   0.04558123 0.07561878 0.1255099 0.1125534 0.1369174 0.1791608    0
## mars  0.18856074 0.28716270 0.3861960 0.4071805 0.5113954 0.6510990    0
## rf    0.16870684 0.25535301 0.3215659 0.3405483 0.4240297 0.5263780    0
```

Using bw-plot to compare their RMSE

```
bwplot(resamp, metric = "RMSE")
```



```
lasso_pred <- predict(lasso_fit, newdata = x_test)
enet_pred <- predict(enet_fit, newdata = x_test)
pls_pred <- predict(pls_fit, newdata = x_test)
mars_pred <- predict(mars_fit, newdata = x_test)
gam_pred <- predict(gam_fit, newdata = x_test)
rf_pred <- predict(rf_model_final, x_test)
```

```r
lasso_performance <- postResample(pred = lasso_pred, obs = test_data$recovery_time)
lasso_performance
```

```
##       RMSE   Rsquared        MAE 
## 21.6024677  0.1637076 12.8215166
```

```r
enet_performance <- postResample(pred = enet_pred, obs = test_data$recovery_time)
enet_performance
```

```
##       RMSE   Rsquared        MAE 
## 21.6530528  0.1628637 12.7871753
```

```r
pls_performance <- postResample(pred = pls_pred, obs = test_data$recovery_time)
pls_performance
```

```
##       RMSE   Rsquared        MAE 
## 21.5903221  0.1631082 12.8407993
```

```r
mars_performance <- postResample(pred = mars_pred, obs = test_data$recovery_time)
mars_performance
```

```
##       RMSE   Rsquared        MAE 
## 34.8585974  0.3524154 12.9107465
```

```r
gam_performance <- postResample(pred = gam_pred, obs = test_data$recovery_time)
gam_performance
```

```
##       RMSE   Rsquared        MAE 
## 21.5949142  0.3554629 13.4168111
```

```r
rf_performance <- postResample(pred = rf_pred, obs = test_data$recovery_time)
rf_performance
```

```
##       RMSE   Rsquared        MAE 
## 18.1800886  0.4208812 12.1826118
```