Covid Recovery Analysis - Figures and outputs

Cary Ni

```r
# load the external dataset
load("recovery.rdata")
# change the variable type based on the reference
index_factor = c(3, 4, 5,  9, 10, 13, 14, 15)
index_numer = c(2, 6, 7, 8, 11, 12)
dat[, index_factor]= lapply(dat[, index_factor], as.factor)
# extract 2000+2000 samples for analysis
set.seed(2604)
dat_1 <- dat[sample(1:10000, 2000),]
set.seed(3508)
dat_2 <- dat[sample(1:10000, 2000),]
# merge the dataset for unique values
# create a new variable length_ind with 30 days as threshold
dat = rbind(dat_1, dat_2) %>%
  unique() %>%
  mutate(
    length_ind = ifelse(recovery_time>30, 1, 0),
    length_ind = as.factor(length_ind)
  )

# seperate training set and test set
set.seed(2023)
train_row = createDataPartition(y = dat$recovery_time, p = 0.8, list = FALSE)
```

```r
# create covariates matrix for training and test
predictors_train = model.matrix(recovery_time ~ ., data = dat[train_row, -c(1,17)])[, -1]
predictors_test = model.matrix(recovery_time ~ ., data = dat[-train_row, -c(1,17)])[, -1]
# create response vector for training and test
response_train = dat[train_row, -c(1,17)]$recovery_time
response_test = dat[-train_row, -c(1,17)]$recovery_time
```

```r
# create covariates matrix for training and test
predictors_train_new = model.matrix(length_ind ~ ., data = dat[train_row, -c(1,16)])[, -1]
predictors_test_new = model.matrix(length_ind ~ ., data = dat[-train_row, -c(1,16)])[, -1]
# create response vector for training and test
response_train_new = dat[train_row, -c(1,16)]$length_ind
response_test_new = dat[-train_row, -c(1,16)]$length_ind
```

Exploratory analysis and data visualization

```r
# summary statistics
dat %>%
  skimr::skim() %>%
  knitr::knit_print()
```

Table 1: Data summary

Name	Piped data
Number of rows	3603
Number of columns	17
Column type frequency:	
factor	9
numeric	8
Group variables	None

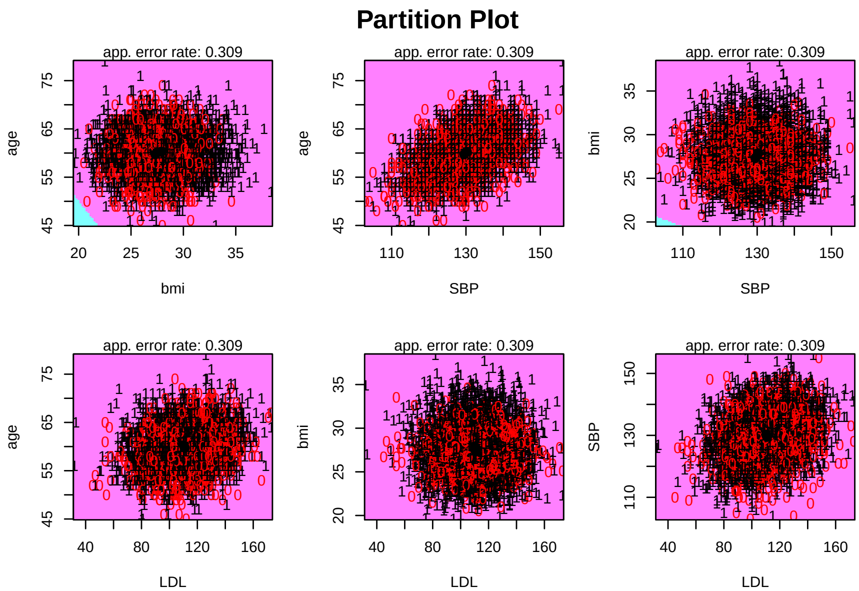
Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
gender	0	1	FALSE	2	0: 1867, 1: 1736
race	0	1	FALSE	4	1: 2350, 3: 720, 4: 358, 2: 175
smoking	0	1	FALSE	3	0: 2176, 1: 1072, 2: 355
hypertension	0	1	FALSE	2	0: 1886, 1: 1717
diabetes	0	1	FALSE	2	0: 3065, 1: 538
vaccine	0	1	FALSE	2	1: 2119, 0: 1484
severity	0	1	FALSE	2	0: 3252, 1: 351
study	0	1	FALSE	3	B: 2201, A: 718, C: 684
length_ind	0	1	FALSE	2	1: 2491, 0: 1112

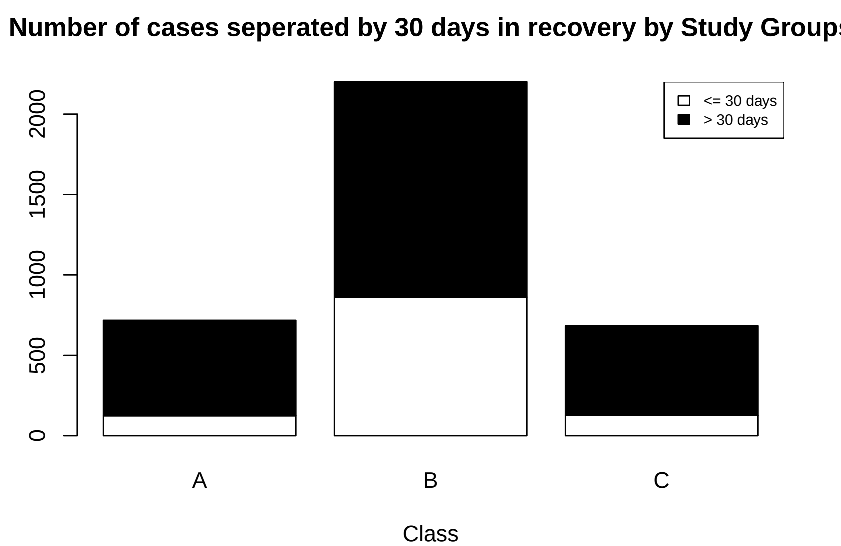
Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
id	0	1	4979.49	2845.35	2.0	2528.0	4987.0	7438.0	9999.0	
age	0	1	60.10	4.46	45.0	57.0	60.0	63.0	79.0	
height	0	1	170.02	5.93	148.1	166.1	170.0	173.9	189.1	
weight	0	1	79.91	7.08	57.1	75.0	79.9	84.7	104.2	
bmi	0	1	27.71	2.77	19.6	25.8	27.6	29.4	38.4	
SBP	0	1	130.01	7.88	103.0	125.0	130.0	135.0	156.0	
LDL	0	1	110.51	19.83	32.0	97.0	111.0	125.0	173.0	
recovery_time	0	1	42.98	29.46	2.0	28.0	39.0	50.0	365.0	

```r
klaR::partimat(length_ind ~ age+bmi+SBP+LDL,
         data = dat, method = "lda")
```

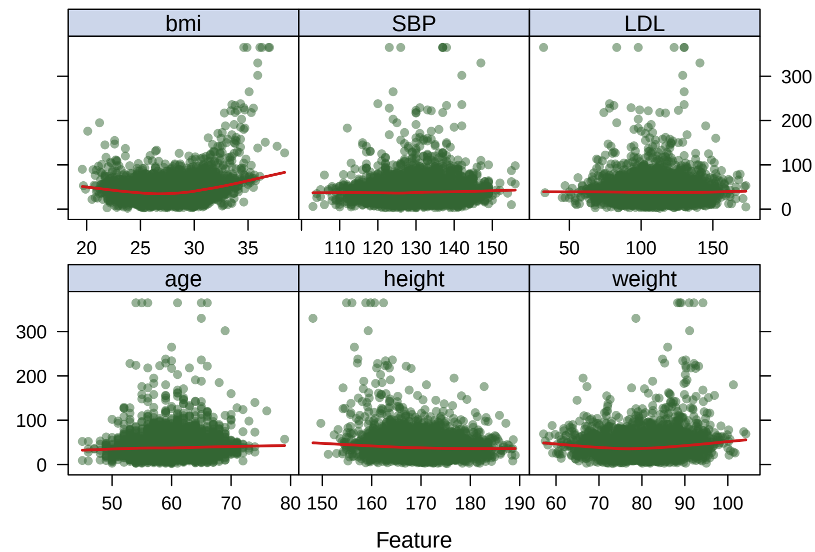



```
table(dat$length_ind, dat$study) %>% barplot(main = "Number of cases seperated by 30 days in recovery by
                                             xlab = "Class",
                                             col = c("White","Black"))
legend("topright",
       c("<= 30 days","> 30 days"),
       fill = c("White","Black"),
       cex = 0.7)
```



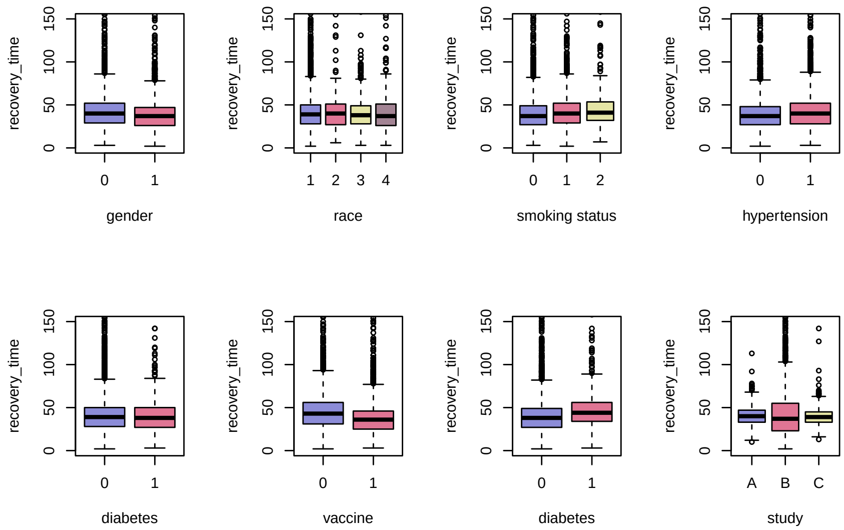
Visualize potential relationship between reponse variable and numeric predictors

```r
# simple visualization of the numeric data
theme1 <- trellis.par.get()
theme1$plot.symbol$col <- rgb(.2, .4, .2, .5)
theme1$plot.symbol$pch <- 16
theme1$plot.line$col <- rgb(.8, .1, .1, 1)
theme1$plot.line$lwd <- 2
theme1$strip.background$col <- rgb(.0, .2, .6, .2)
trellis.par.set(theme1)
featurePlot(x = dat[, index_numer],
            y = dat$recovery_time,
            plot = "scatter",
            type = c("p", "smooth"),
            layout = c(3, 2))
```



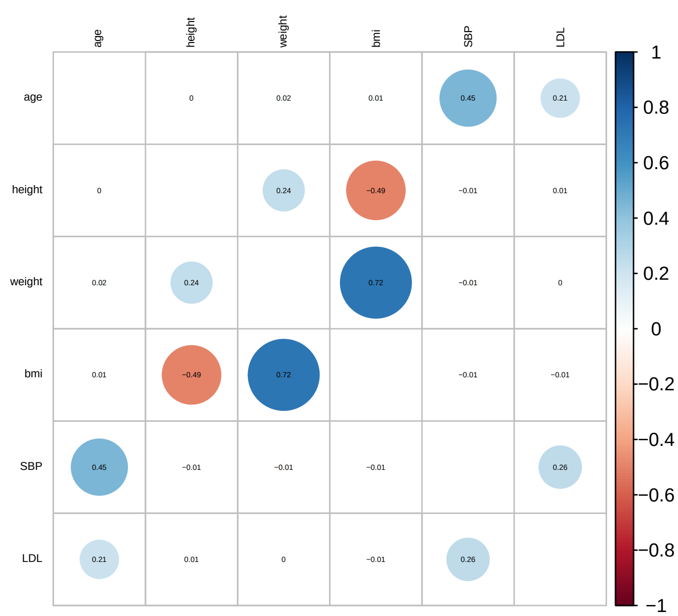
Visualize potential relationship between reponse variable and categorical predictors

```r
# simple visualization of the categorical data
par(mfrow=c(2,4))
myColors = c(rgb(0.1,0.1,0.7,0.5) , rgb(0.8,0.1,0.3,0.6), rgb(0.8,0.8,0.3,0.5),
             rgb(0.4,0.2,0.3,0.6))

boxplot(recovery_time ~ gender, data = dat, xlab = "gender", col = myColors, ylim=c(0, 150))
boxplot(recovery_time ~ race, data = dat, xlab = "race", col = myColors, ylim=c(0, 150))
boxplot(recovery_time ~ smoking, data = dat, xlab = "smoking status", col = myColors, ylim=c(0, 150))
boxplot(recovery_time ~ hypertension, data = dat, xlab = "hypertension", col = myColors, ylim=c(0, 150)
boxplot(recovery_time ~ diabetes, data = dat, xlab = "diabetes", col = myColors, ylim=c(0, 150))
boxplot(recovery_time ~ vaccine, data = dat, xlab = "vaccine", col = myColors, ylim=c(0, 150))
boxplot(recovery_time ~ severity, data = dat, xlab = "diabetes", col = myColors, ylim=c(0, 150))
boxplot(recovery_time ~ study, data = dat, xlab = "study", col = myColors, ylim=c(0, 150))
```



Correlation plot to check collinearity between covariates (based on training data)

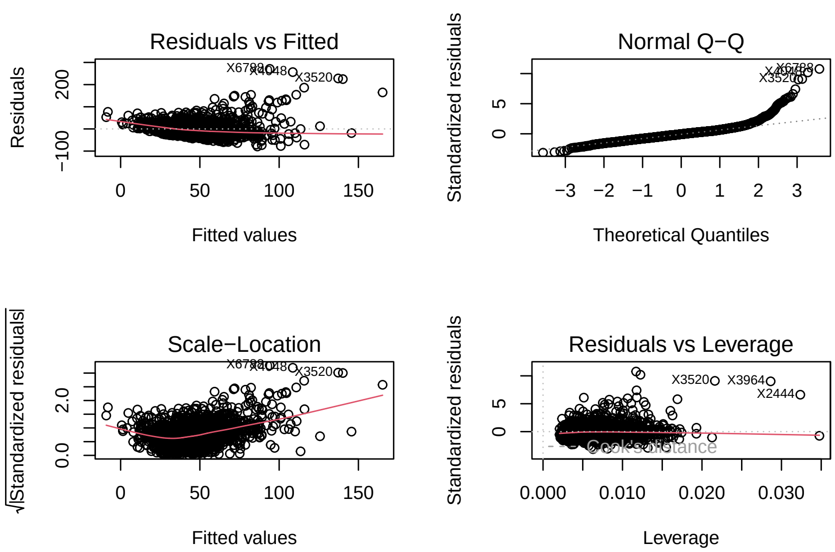
```
cor(predictors_train[, c(1, 8, 9, 10, 13, 14)]) %>% corrplot(
  method = "circle", type = "full",
  addCoef.col = 1, number.font =0.5,
  tl.col="black", tl.srt=90, tl.cex = 0.5,
  insig = "blank", diag=FALSE, number.cex = .3)
```



Model Training

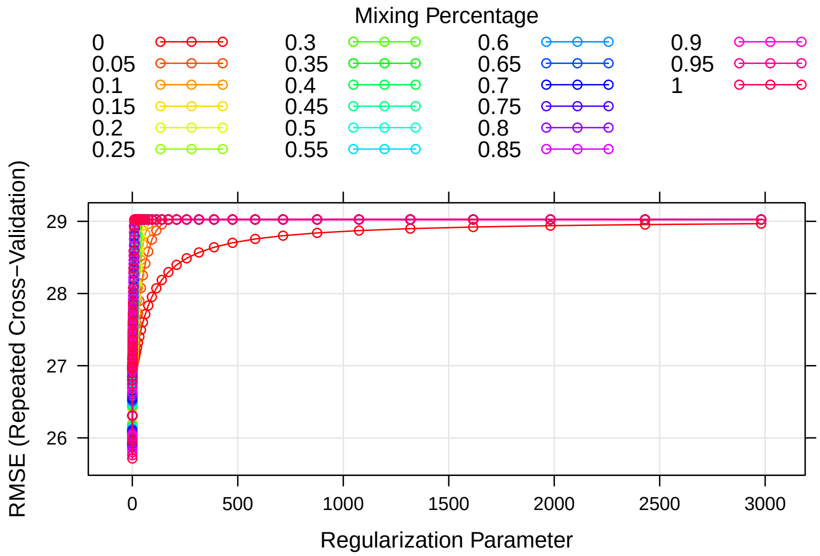
Ordinary Least square

```
# set train method
ctrl_1 = trainControl(method = "repeatedcv", number = 10, repeats = 5)
# build the linear least squared model with caret
set.seed(1)
lm_model = train(predictors_train, response_train, method = "lm", trControl = ctrl_1)
par(mfrow = c(2, 2))
plot(lm_model$finalModel)
```



Elastic net regression

```
set.seed(1)
# build elastic net model with caret
elnet_model = train(predictors_train, response_train,
                    method = "glmnet",
                    tuneGrid = expand.grid(alpha = seq(0, 1, length=21),
                                           lambda = exp(seq(-2, 8, length=50))),
                    trControl = ctrl_1)
myCol<- rainbow(21)
myPar <- list(superpose.symbol = list(col = myCol),
superpose.line = list(col = myCol))
plot(elnet_model, par.settings = myPar)
```

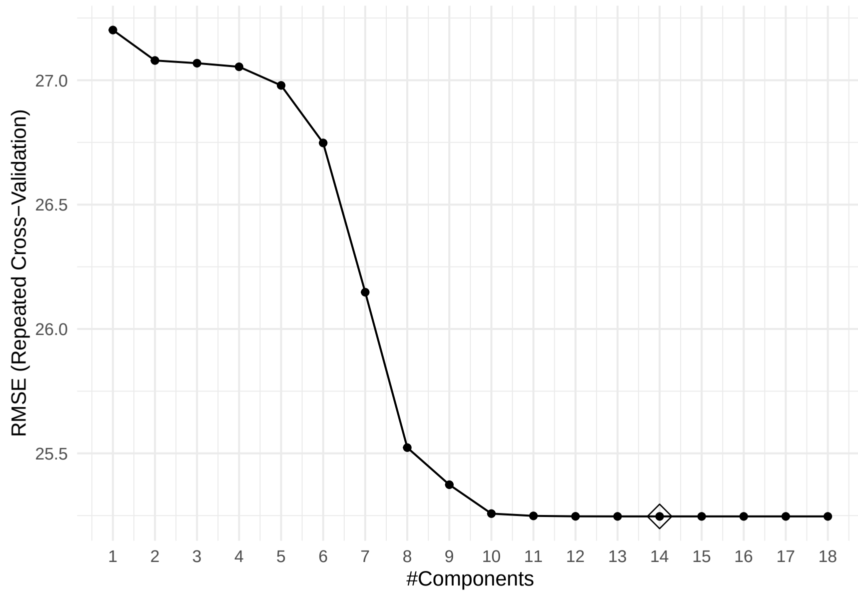



```r
# show the best lambda and alpha combination with lowest cv rmse
elnet_model$bestTune
```

```
##      alpha    lambda
## 1001     1 0.1353353
```

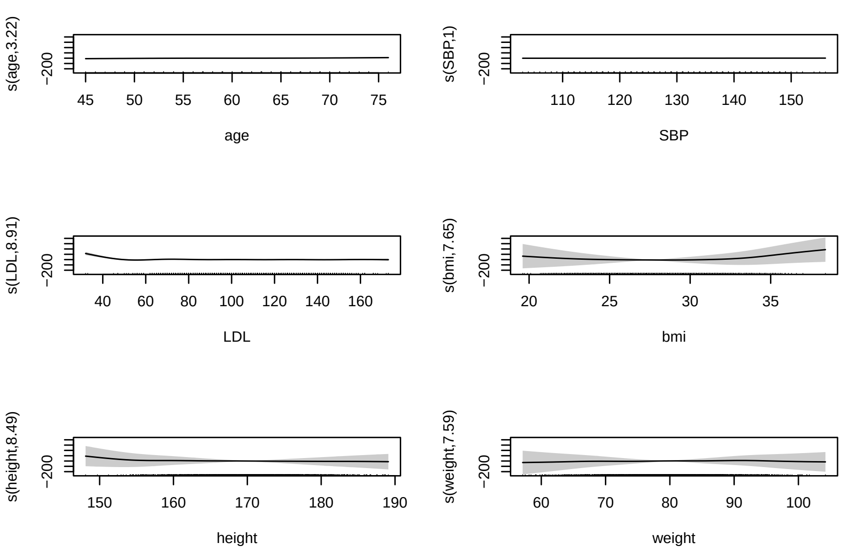
Partial least squares (PLS)

```r
set.seed(1)
pls_model = train(predictors_train, response_train,
                method = "pls",
                # 18 variables in total
                tuneGrid = data.frame(ncomp = 1:18),
                trControl = ctrl_1,
                preProcess = c("center", "scale"))
ggplot(pls_model, highlight = TRUE) +
  scale_x_continuous(breaks = seq(0,20,by=1))
```



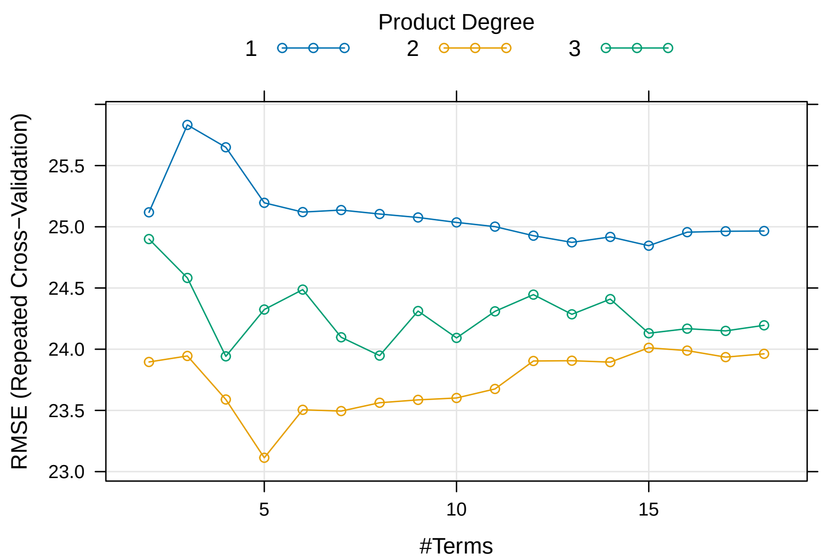
Generalized Additive Models (GAM)

```
set.seed(1)
tune_Grid = data.frame(method = "GCV.Cp", select = c(TRUE, FALSE))
gam_model = train(predictors_train, response_train,
                   method = "gam",
                  tuneGrid = tune_Grid,
                  trControl = ctrl_1)
par(mfrow = c(3, 2))
plot(gam_model$finalModel, shade = TRUE)
```



Multivariate adaptive regression spline model (MARS)

```
set.seed(1)
# Set tuning parameters (18 as maximum number of terms taken since only 18 predictors are used)
mars_grid = expand.grid(degree = 1:3, nprune = 2:18)
# Fit MARS model
mars_model = train(predictors_train, response_train,
                   method = "earth",
                   tuneGrid = mars_grid,
                   trControl = ctrl_1)
# Plot the model
plot(mars_model)
```




```
summary(mars_model)
```

```
## Call: earth(x=matrix[2884,18], y=c(17,33,92,20,4...), keepxy=TRUE, degree=2,
##             nprune=5)
##
##                        coefficients
## (Intercept)               -2.941972
## vaccine1                  -9.506054
## h(bmi-24.5)                7.647737
## h(30.6-bmi)                7.320881
## h(bmi-30.6) * studyB      20.811559
##
## Selected 5 of 24 terms, and 3 of 18 predictors (nprune=5)
## Termination condition: Reached nk 37
## Importance: bmi, studyB, vaccine1, age-unused, gender1-unused, ...
## Number of terms at each degree of interaction: 1 3 1
## GCV 516.0731    RSS 1477023    GRSq 0.4041199    RSq 0.4082465
```

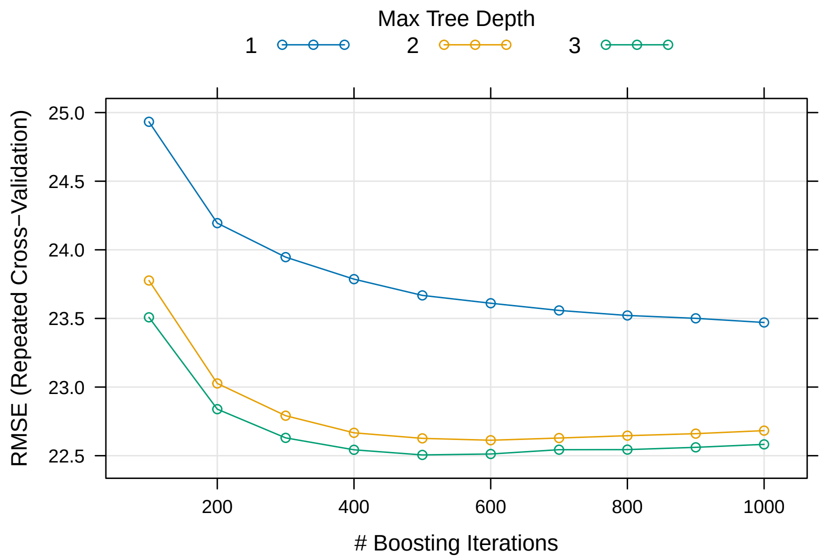
K-Nearest Neighbors (KNN)

```
set.seed(1)
knn_grid = expand.grid(k = seq(5, 15, by = 1))
knn_model = train(predictors_train, response_train,
                  method = "knn",
                  preProcess = c("center", "scale"),
                  trControl = ctrl_1,
                  tuneGrid = knn_grid)
knn_model$bestTune
```

```
##    k
## 9 13
```

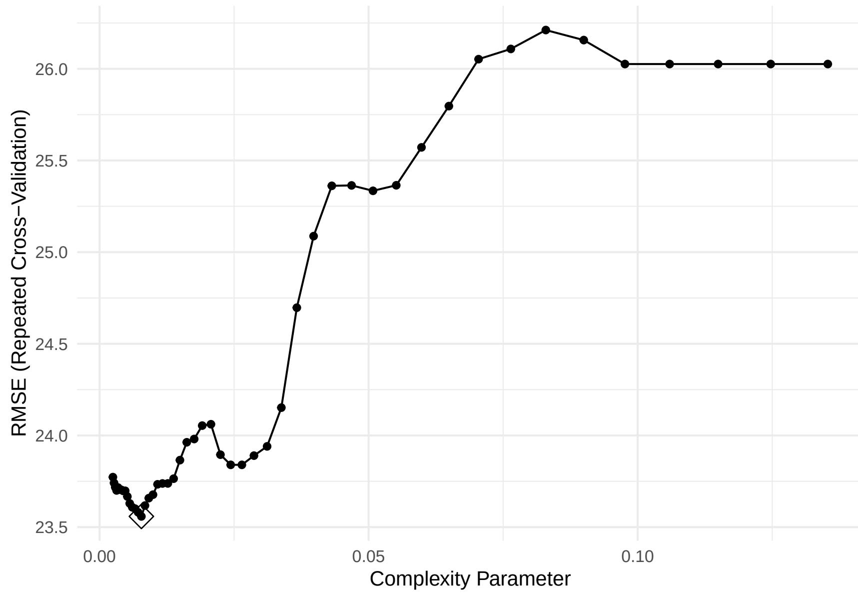
Generalized Boosted Regression

```
set.seed(1)
# setting number of trees, depth, learning rate, and default leaves number
# learning rate = max(0.01, 0.1*(min(1, nl/10000)))
gbm_grid = expand.grid(
  n.trees = c(seq(100, 1000, by = 100)),
  interaction.depth = c(1, 2, 3),
  shrinkage = 0.02,
  n.minobsinnode = 5
)
gbm_model = train(predictors_train, response_train,
                  method = "gbm",
                 preProcess = c("center", "scale"),
                 trControl = ctrl_1,
                 tuneGrid = gbm_grid,
                 verbose = FALSE)
plot(gbm_model)
```

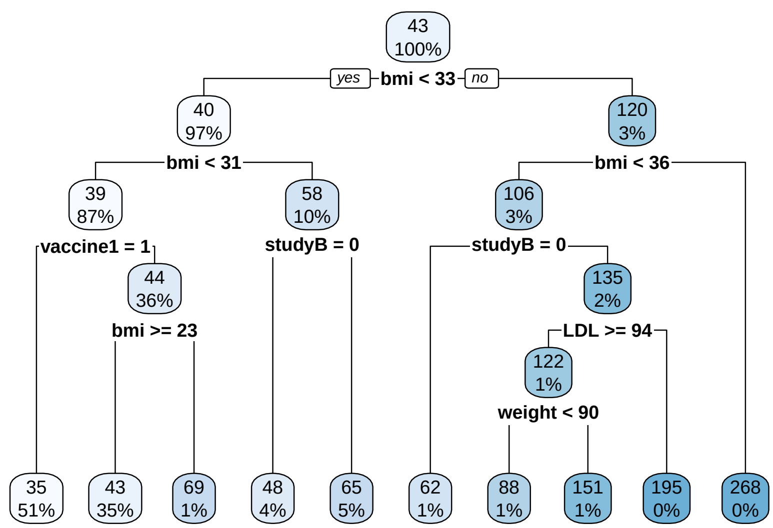


Regression Tree (CART)

```
set.seed(1)

# build a regression tree on the training data using the caret package
rpart_model <- train(recovery_time ~ . ,
                     data = dat[train_row, -c(1,17)], # training data
                     method = "rpart", # regression tree model
                     tuneGrid = data.frame(cp = exp(seq(-6,-2, length = 50))), # candidate values for the
                     trControl = ctrl_1)

# create a plot of the complexity parameter selection
ggplot(rpart_model, highlight = TRUE) # highlight the optimal cp value
```



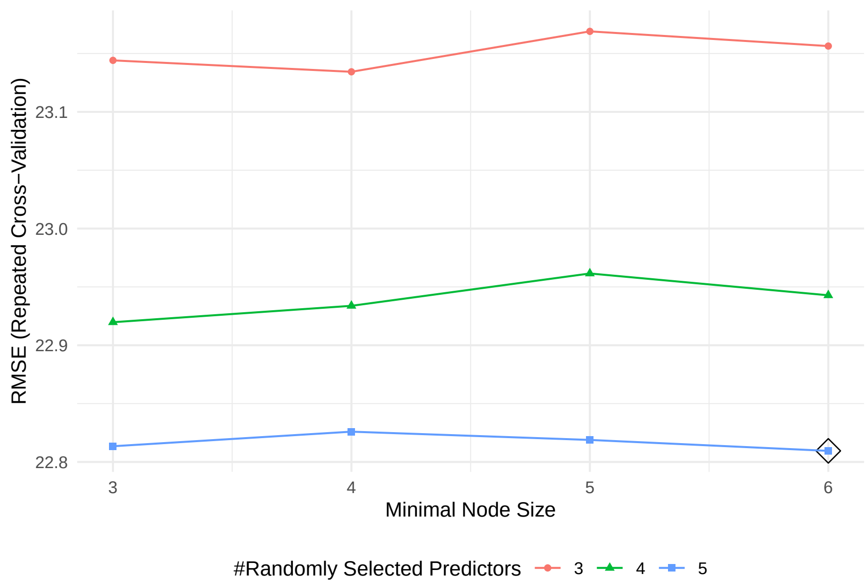

```
# create a plot of the tree using the rpart.plot() function
rpart.plot(rpart_model$finalModel)
```



Random Forest

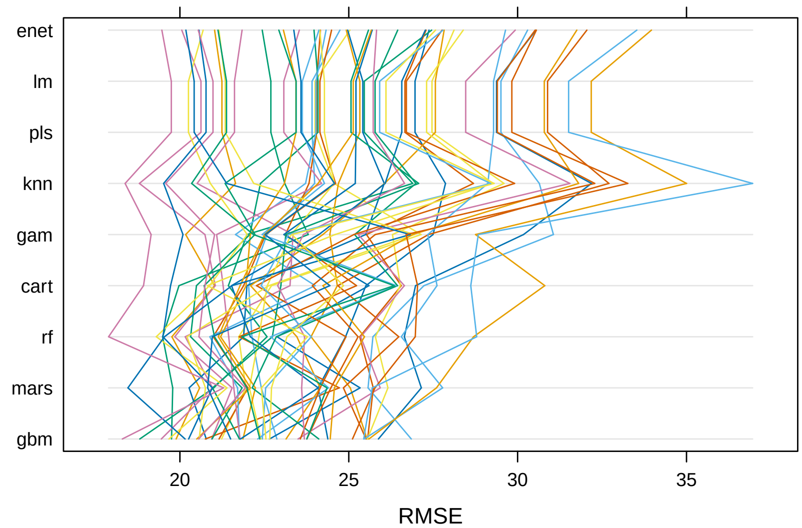
```
set.seed(1)
# build a random forest model
# mtry: Since mtry should be set to the square root of the total number of predictors for classificatio

rf_grid <- expand.grid(mtry = 3:5,
                       splitrule = "variance",
```

```
                    min.node.size = 3:6)

rf_model <- train(recovery_time ~ .,
                  dat[train_row, -c(1,17)],
                  method = "ranger",
                  tuneGrid = rf_grid,
                  trControl = ctrl_1)

# create a plot
ggplot(rf_model, highlight = TRUE)
```

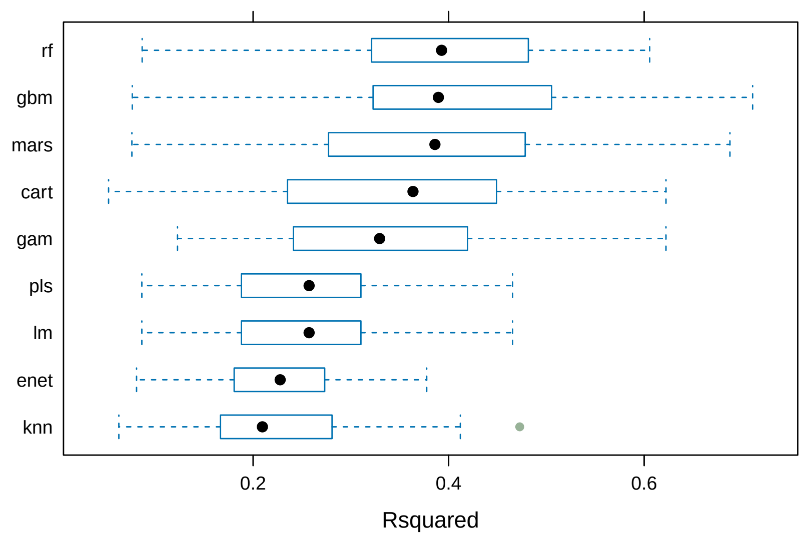


Models comparsion based on cross validation error

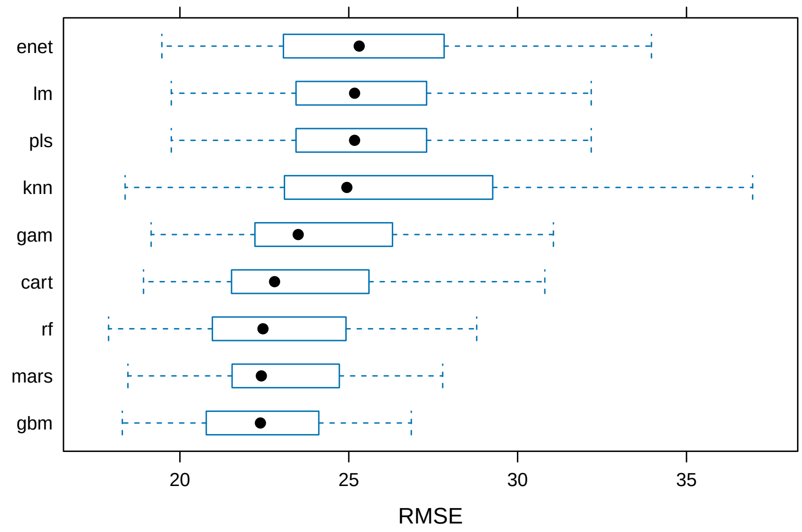
```
# compare model performance through sampling method
resamp = resamples(list(
  lm = lm_model,
  enet = elnet_model,
  pls = pls_model,
  gam = gam_model,
  mars = mars_model,
  knn = knn_model,
  gbm = gbm_model,
  cart = rpart_model,
  rf = rf_model
))
# plot resampling rmse
parallelplot(resamp, metric = "RMSE")
```




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




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




```r
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: lm, enet, pls, gam, mars, knn, gbm, cart, rf
## Number of resamples: 50
##
## MAE
##          Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lm   15.16547 16.10302 16.78177 16.83446 17.42209 19.04602    0
## enet 14.56452 15.92803 16.52216 16.46752 17.01105 18.64227    0
## pls  15.16548 16.10302 16.78177 16.83446 17.42208 19.04604    0
## gam  14.35229 15.33438 15.83680 15.95120 16.56241 17.79521    0
## mars 13.93931 14.88663 15.26803 15.37548 15.95721 17.55273    0
## knn  13.77991 14.91692 15.66110 15.73718 16.40355 18.31290    0
## gbm  13.21976 14.36055 14.86937 14.97374 15.48263 16.78132    0
## cart 14.24679 14.87635 15.29857 15.47210 16.11928 17.75836    0
## rf   13.54828 14.39094 14.84221 15.05416 15.75639 17.39155    0
##
## RMSE
##          Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lm   19.74488 23.43892 25.17216 25.24668 27.21931 32.18024    0
## enet 19.46546 23.14204 25.30888 25.71390 27.81579 33.96273    0
## pls  19.74490 23.43892 25.17216 25.24668 27.21931 32.18025    0
## gam  19.14744 22.27447 23.50243 24.21183 26.16733 31.05932    0
## mars 18.46040 21.55759 22.41118 23.11308 24.68399 27.78102    0
## knn  18.37738 23.09791 24.94296 26.01258 29.24863 36.95880    0
## gbm  18.29410 20.81975 22.38344 22.50542 24.02405 26.85158    0
## cart 18.92266 21.53195 22.80303 23.55843 25.57517 30.80530    0
## rf   17.88756 20.96764 22.45933 22.80951 24.91435 28.78803    0
##
```

```
## Rsquared
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lm   0.08620897 0.1900059 0.2572871 0.2531044 0.3099187 0.4654627    0
## enet 0.08078819 0.1811701 0.2277041 0.2290413 0.2731629 0.3775725    0
## pls  0.08620900 0.1900061 0.2572879 0.2531045 0.3099185 0.4654625    0
## gam  0.12269099 0.2416508 0.3294108 0.3350657 0.4183874 0.6223217    0
## mars 0.07601256 0.2807337 0.3859944 0.3750641 0.4765129 0.6877643    0
## knn  0.06268675 0.1685928 0.2095025 0.2280360 0.2800124 0.4727332    0
## gbm  0.07642602 0.3251101 0.3895383 0.4023679 0.5004710 0.7109789    0
## cart 0.05216633 0.2359045 0.3635301 0.3517752 0.4471816 0.6223333    0
## rf   0.08652982 0.3224828 0.3928160 0.3915504 0.4790679 0.6056875    0
```

Results

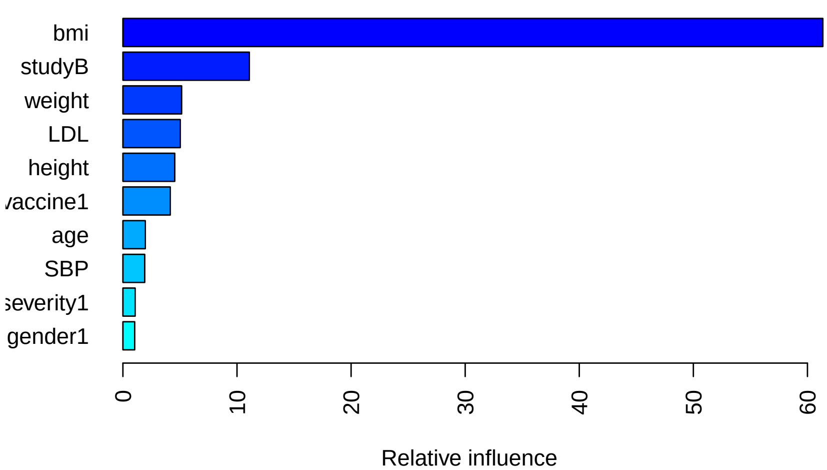
Test Mean Squared Error

```
# get test mse
predict_value = predict(gbm_model, newdata = predictors_test)
test_mse = mean((predict_value - response_test)^2)
test_mse %>% knitr::knit_print()
```

```
## [1] 479.9529
```

Variable importance plots

```
par(mfrow = c(1, 1))
var_df = summary(gbm_model,
        cBars = 10,
        las = 2)
```

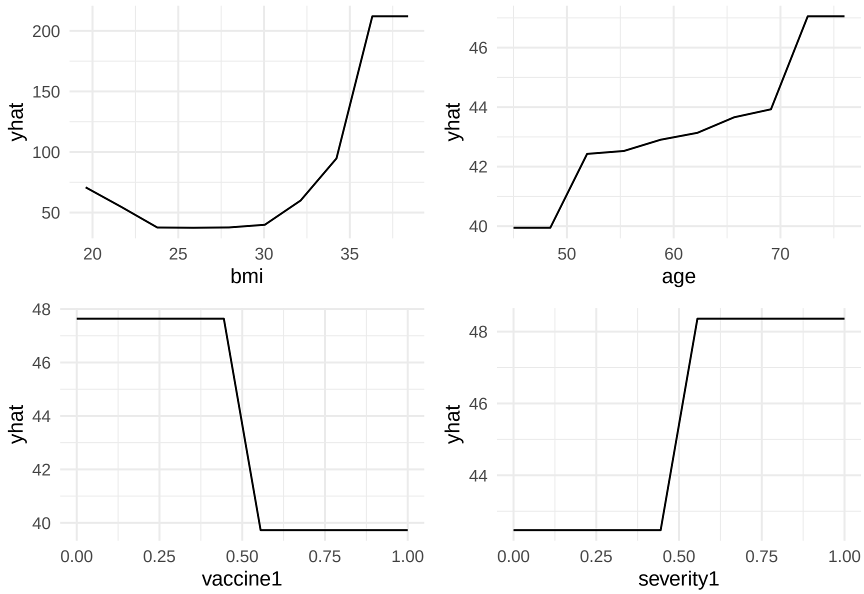


```r
var_df %>%
  as.data.frame() %>%
  dplyr::select(-var) %>%
  knitr::kable()
```

	rel.inf
bmi	61.3499576
studyB	11.0818448
weight	5.1468465
LDL	5.0259253
height	4.5435387
vaccine1	4.1495410
age	1.9610485
SBP	1.9116559
severity1	1.0737375
gender1	1.0317295
smoking2	0.7984746
race2	0.5512056
smoking1	0.4402374
race4	0.4217472
hypertension1	0.3594399
diabetes1	0.1530700
race3	0.0000000
studyC	0.0000000

Partial dependance plots

```r
p1 = pdp::partial(gbm_model, pred.var = c("bmi"),
                  grid.resolution = 10) %>% autoplot()
p2 = pdp::partial(gbm_model, pred.var = c("vaccine1"),
                  grid.resolution = 10) %>% autoplot()
p3 = pdp::partial(gbm_model, pred.var = c("severity1"),
                  grid.resolution = 10) %>% autoplot()
p4 = pdp::partial(gbm_model, pred.var = c("age"),
                  grid.resolution = 10) %>% autoplot()

grid.arrange(arrangeGrob(p1, p4, ncol = 2), arrangeGrob(p2, p3, ncol = 2))
```



200
150
100
50
yhat
20
25
30
35
bmi
46
44
42
40
yhat
50
60
70
age
48
46
44
42
40
yhat
0.00
0.25
0.50
0.75
1.00
vaccine1
48
46
44
yhat
0.00
0.25
0.50
0.75
1.00
severity1