

# analysis\_II

2023-05-07

```
library(janitor)
library(tidyverse)
library(AppliedPredictiveModeling)
library(lattice)
library(caret)
library(corrplot)
library(GGally)
library(miscset)
library(ggpubr)
library(knitr)
library(rpart)
library(rpart.plot)
library(ranger)
library(kernlab)

library(rpart)
library(rpart.plot)
library(party)
library(partykit)
library(pROC)

library(randomForest)
library(ranger)
library(gbm)
library(pdp)

library(doParallel)
library(gtsummary)
```

## 1. Load Data

```
load("final_data.RData")
```

## 2. Train/test split

```
set.seed(2)
final_data <- final_data %>%
  mutate(binary_recovery_time = factor(binary_recovery_time))
levels(final_data$binary_recovery_time) = c("low", "high")
training_rows <- createDataPartition(final_data$binary_recovery_time,
```

```
p = 0.8,
list = F)
```

```
train_x <- model.matrix(binary_recovery_time~., final_data %>% select(-id, -recovery_time))[training_rows]
train_y <- final_data$binary_recovery_time[training_rows]
test_x <- model.matrix(binary_recovery_time~., final_data %>% select(-id, -recovery_time))[-training_rows]
test_y <- final_data$binary_recovery_time[-training_rows]
```

```
training_set <- final_data[training_rows,]
```

### 3.EDA

basic summary statistics

```
summary(training_set)
```

```
##      id      age      gender      race      smoking      height
## Min.   :    5   Min.   :42.00  0:1475   1:1840   0:1784   Min.   :151.2
## 1st Qu.: 2510   1st Qu.:57.00  1:1413   2: 151   1: 832   1st Qu.:166.1
## Median : 4909   Median :60.00           3: 599   2: 272   Median :170.1
## Mean   : 4962   Mean   :60.07           4: 298   Mean   :169.9
## 3rd Qu.: 7469   3rd Qu.:63.00           3rd Qu.:173.9
## Max.   :10000   Max.   :79.00           Max.   :195.9
##      weight      bmi      hypertension diabetes      sbp
## Min.   : 57.90   Min.   :19.90  0:1525      0:2445   Min.   :105
## 1st Qu.: 75.40   1st Qu.:25.98  1:1363      1: 443   1st Qu.:125
## Median : 80.10   Median :27.70           Median :130
## Mean   : 80.08   Mean   :27.80           Mean   :130
## 3rd Qu.: 84.90   3rd Qu.:29.50           3rd Qu.:135
## Max.   :105.00   Max.   :38.90           Max.   :158
##      ldl      vaccine severity study      recovery_time
## Min.   : 32.0  0:1177  0:2617  A: 582   Min.   : 2.00
## 1st Qu.: 97.0  1:1711  1: 271  B:1734   1st Qu.: 28.00
## Median :110.0           C: 572   Median : 39.00
## Mean   :110.2           Mean   : 42.96
## 3rd Qu.:123.0           3rd Qu.: 50.00
## Max.   :175.0           Max.   :365.00
##      binary_recovery_time
## low : 896
## high:1992
##
##
##
##
```

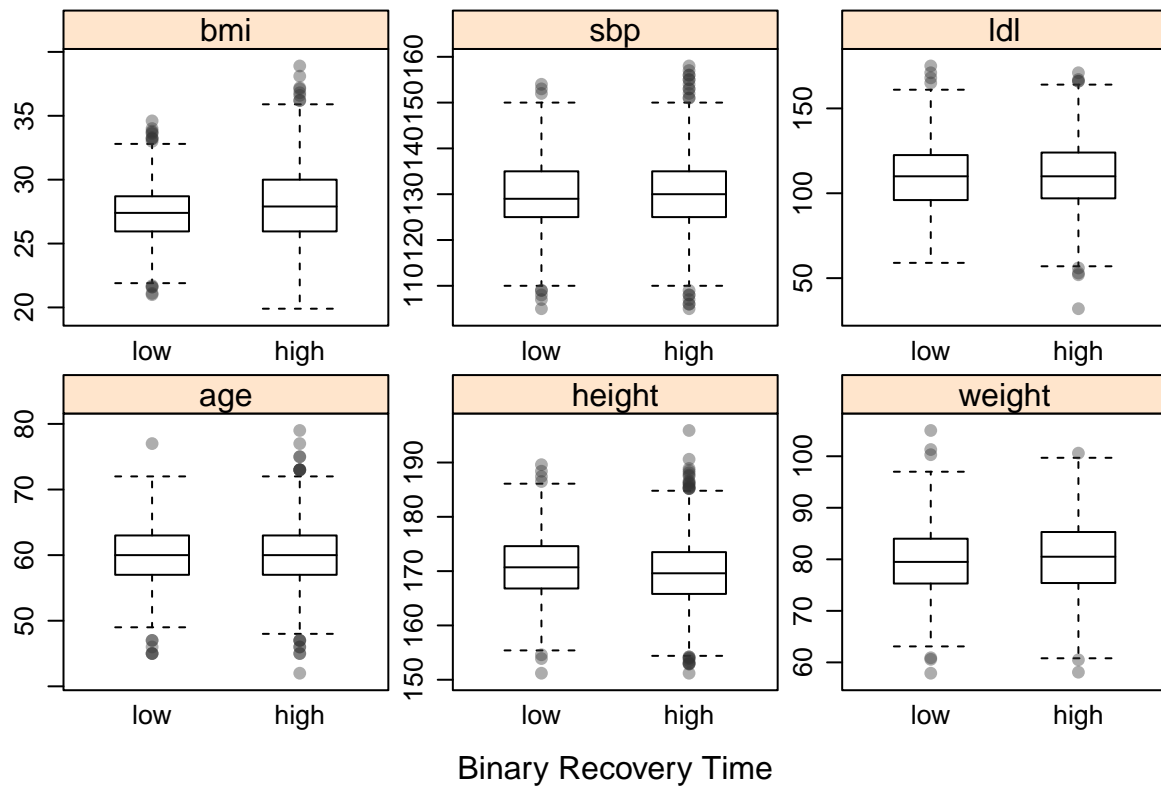
quantatative variables

```
theme <- transparentTheme(trans = 0.4)
theme$plot.symbol$col = rgb(.2, .2, .2, .4)
theme$plot.symbol$pch = 16
```

```

theme$plot.line$col = rgb(1, 0, 0, .7)
theme$plot.line$lwd <- 2
trellis.par.set(theme)
featurePlot(x = training_set %>% dplyr::select(-recovery_time, -binary_recovery_time, -id, -gender, -ra
            y = training_set$binary_recovery_time,
            plot = "box",
            pch = "|",
            scales = list(x = list(relation = "free"),
                          y = list(relation = "free")),
            auto.key = list(columns = 2),
            labels = c("Binary Recovery Time", ""))

```



```

ggplotGrid(ncol = 2,
  lapply(c("bmi", "sbp", "ldl", "age", "height", "weight"),
    function(col) {
      ggplot(training_set, aes_string(col)) + geom_density(aes(y = ..density..))
    })

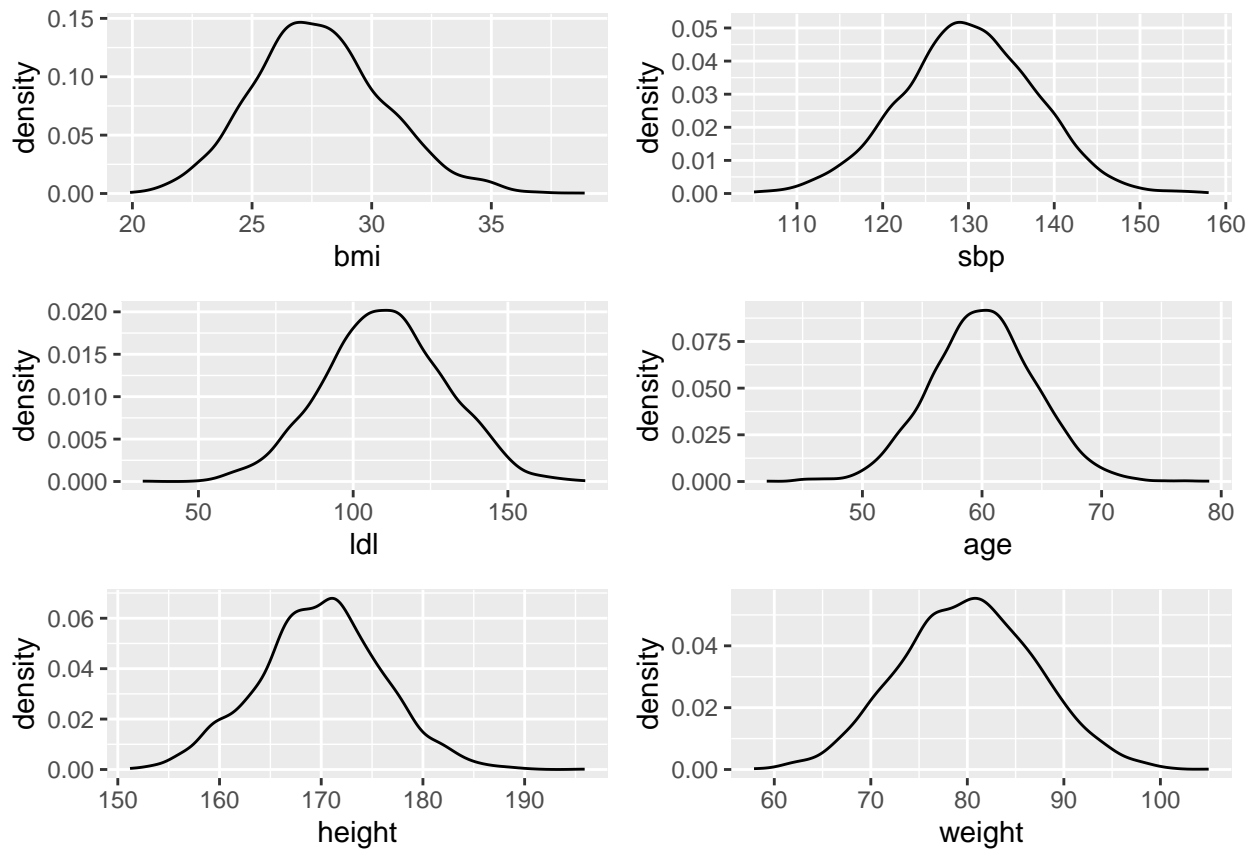
```

```

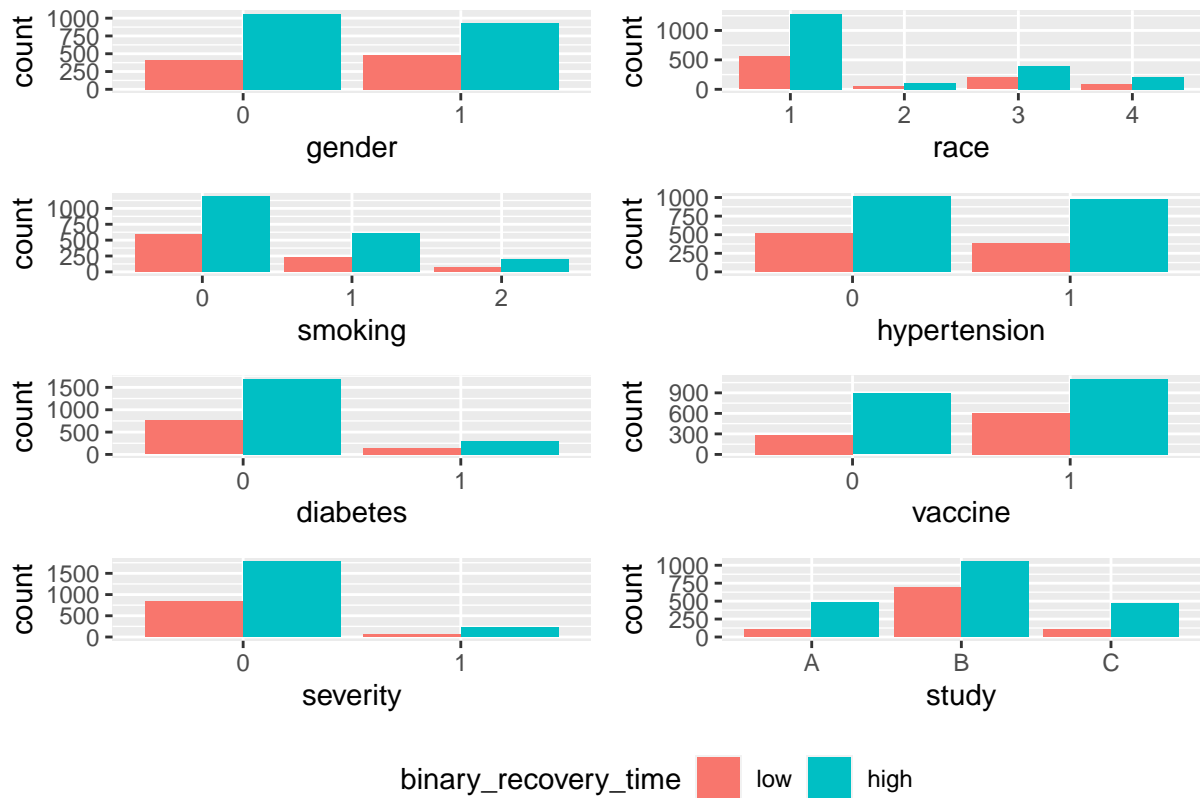
## Warning: 'aes_string()' was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with 'aes()'.
## i See also 'vignette("ggplot2-in-packages")' for more information.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.

```

```
## Warning: The dot-dot notation ('..density..') was deprecated in ggplot2 3.4.0.
## i Please use 'after_stat(density)' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```



```
l <-
  lapply(c("gender", "race", "smoking", "hypertension", "diabetes", "vaccine", "severity", "study"),
    function(col) {
      ggplot(training_set, aes_string(col)) + geom_bar(aes(fill = binary_recovery_time), stat="count",
    })
  patchwork::wrap_plots(l, ncol = 2, guides = "collect") & theme(legend.position = "bottom")
```



```
sum(train_y == "low")
```

```
## [1] 896
```

```
sum(train_y == "high")
```

```
## [1] 1992
```

```
sum(test_y == "low")
```

```
## [1] 223
```

```
sum(test_y == "high")
```

```
## [1] 498
```

```
sum(final_data$binary_recovery_time == "low")
```

```
## [1] 1119
```

```
sum(final_data$binary_recovery_time == "high")
```

```
## [1] 2490
```

#### 4. set up control

```
ctrl2 = trainControl(method = "cv", number = 10)
```

#### 5. Model training

##### a. Logistic regression

```
set.seed(2)
#fit a logistic regression using caret
model.glm = train(train_x, train_y, method = "glm", metric = "Accuracy",
                  trControl = ctrl2)
summary(model.glm)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.4670  -1.1407   0.6120   0.8681   1.6411
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -7.460e+01  1.388e+01  -5.374 7.70e-08 ***
## age          1.720e-02  1.092e-02   1.575 0.115307
## gender1     -3.090e-01  8.555e-02  -3.611 0.000304 ***
## race2       -3.677e-02  1.961e-01  -0.188 0.851260
## race3       -1.764e-01  1.070e-01  -1.648 0.099291 .
## race4        1.095e-03  1.439e-01   0.008 0.993928
## smoking1     2.959e-01  9.780e-02   3.026 0.002480 **
## smoking2     4.111e-01  1.549e-01   2.654 0.007949 **
## height       4.321e-01  8.133e-02   5.313 1.08e-07 ***
## weight      -4.735e-01  8.678e-02  -5.457 4.84e-08 ***
## bmi          1.437e+00  2.506e-01   5.737 9.66e-09 ***
## hypertension1 2.714e-01  1.415e-01   1.918 0.055148 .
## diabetes1    -3.350e-02  1.193e-01  -0.281 0.778852
## sbp          -1.345e-04  9.589e-03  -0.014 0.988810
## ld1          -5.396e-04  2.260e-03  -0.239 0.811341
## vaccine1     -5.989e-01  8.919e-02  -6.715 1.88e-11 ***
## severity1     6.580e-01  1.656e-01   3.974 7.07e-05 ***
## studyB       -1.103e+00  1.211e-01  -9.111 < 2e-16 ***
## studyC        3.074e-02  1.557e-01   0.197 0.843475
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 3577.1  on 2887  degrees of freedom
## Residual deviance: 3253.9  on 2869  degrees of freedom
## AIC: 3291.9
##
## Number of Fisher Scoring iterations: 4
```

```
coef(model.glm$finalModel) %>%
  as.matrix() %>%
  as.data.frame() %>%
  rename(value = V1) %>%
  kable(caption = "Logistic Regression Parameter Coefficients")
```

Table 1: Logistic Regression Parameter Coefficients

	value
(Intercept)	-74.5985809
age	0.0171971
gender1	-0.3089547
race2	-0.0367652
race3	-0.1764199
race4	0.0010954
smoking1	0.2959354
smoking2	0.4110928
height	0.4320885
weight	-0.4735363
bmi	1.4373521
hypertension1	0.2714113
diabetes1	-0.0334999
sbp	-0.0001345
ldl	-0.0005396
vaccine1	-0.5989130
severity1	0.6579947
studyB	-1.1031862
studyC	0.0307435

```
contrasts(final_data$binary_recovery_time)
```

```
##      high
## low      0
## high     1
```

*#We first consider the simple classifier with a cut-off of 0.5 and evaluate its performance on the test*

```
test.pred.prob = predict(model.glm$finalModel, newdata = as.data.frame(test_x), type = "response")
test.pred = rep("low", length(test.pred.prob))
test.pred[test.pred.prob > 0.5] = "high"
```

```
confusionMatrix = confusionMatrix(data = as.factor(test.pred),
                                   reference = test_y,
                                   positive = "high")
```

```
## Warning in confusionMatrix.default(data = as.factor(test.pred), reference =
## test_y, : Levels are not in the same order for reference and data. Refactoring
## data to match.
```

```
confusionMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction low high
##      low   54   42
##      high 169  456
##
##              Accuracy : 0.7074
##              95% CI : (0.6726, 0.7403)
##      No Information Rate : 0.6907
##      P-Value [Acc > NIR] : 0.1772
##
##              Kappa : 0.1873
##
##  Mcnemar's Test P-Value : <2e-16
##
##              Sensitivity : 0.9157
##              Specificity : 0.2422
##              Pos Pred Value : 0.7296
##              Neg Pred Value : 0.5625
##              Prevalence : 0.6907
##              Detection Rate : 0.6325
##      Detection Prevalence : 0.8669
##              Balanced Accuracy : 0.5789
##
##      'Positive' Class : high
##
```

```
#Testing error rate is 0.2926491
1 - confusionMatrix$overall["Accuracy"]
```

```
## Accuracy
## 0.2926491
```

```
#Training error rate is 0.2939751
train.pred.prob = predict(model.glm$finalModel, newdata = as.data.frame(train_x), type = "response")
train.pred.prob[train.pred.prob > 0.5] = "high"
train.pred.prob[train.pred.prob < 0.5] = "low"
table(train_y, train.pred.prob)
```

```
##           train.pred.prob
```



```
## train_y high low
##      low   682 214
##      high 1825 167
```

```
mean(train.pred.prob != train_y)
```

```
## [1] 0.2939751
```

## b. Random Forest

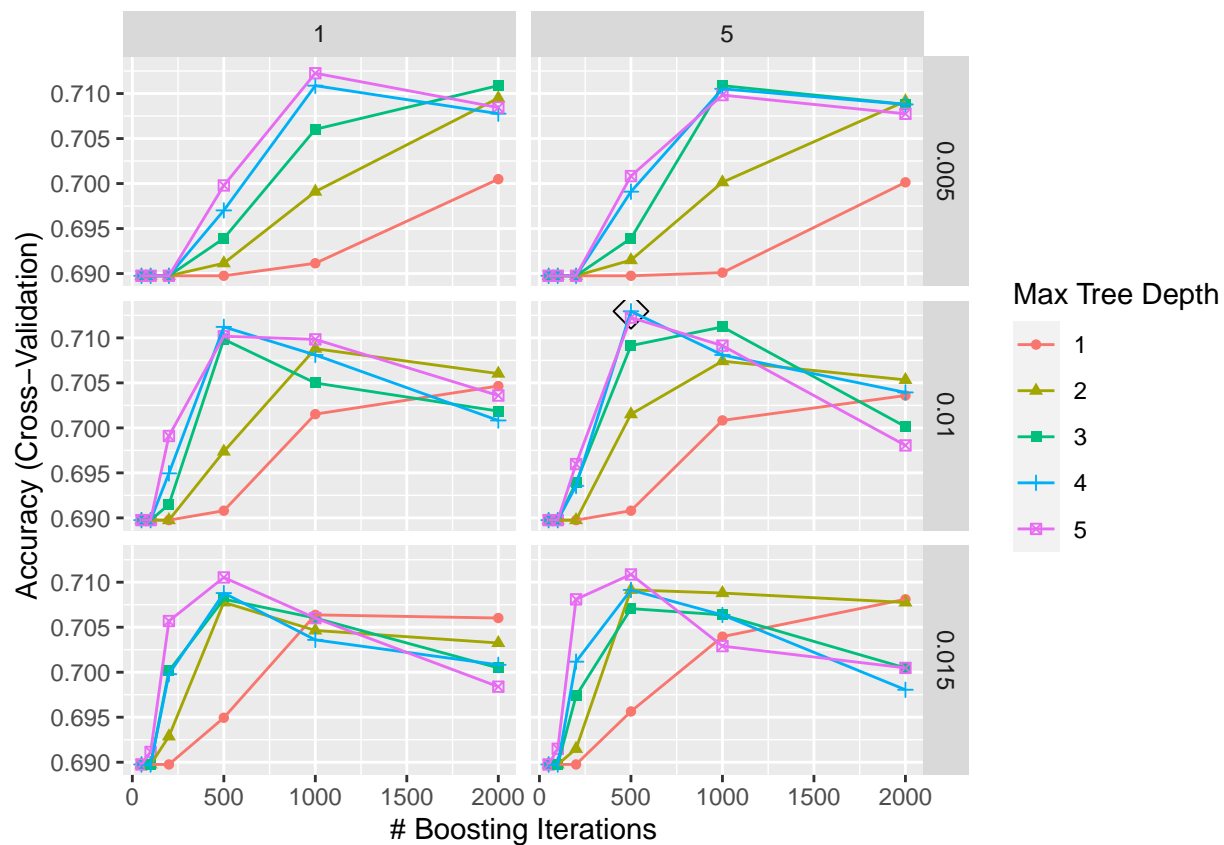
```
set.seed(2)
gbm_grid <- expand.grid(
  # upper bound for number of trees
  n.trees = c(50, 100, 200, 500, 1000, 2000),
  # similar to number of splits in the tree
  # number of layers in the tree
  interaction.depth = 1:5,
  shrinkage = c(0.005, 0.01, 0.015),
  # min obs allowed in a node
  n.minobsinnode = c(1, 5))
```

```
Mycluster = makeCluster(detectCores() - 2)
registerDoParallel(Mycluster)

boost_fit <- train(train_x,
  train_y,
  method = "gbm",
  tuneGrid = gbm_grid,
  trControl = ctrl2,
  distribution = "adaboost",
  verbose = FALSE)
```

```
stopCluster(Mycluster)
registerDoSEQ()
```

```
ggplot(boost_fit, highlight = TRUE)
```

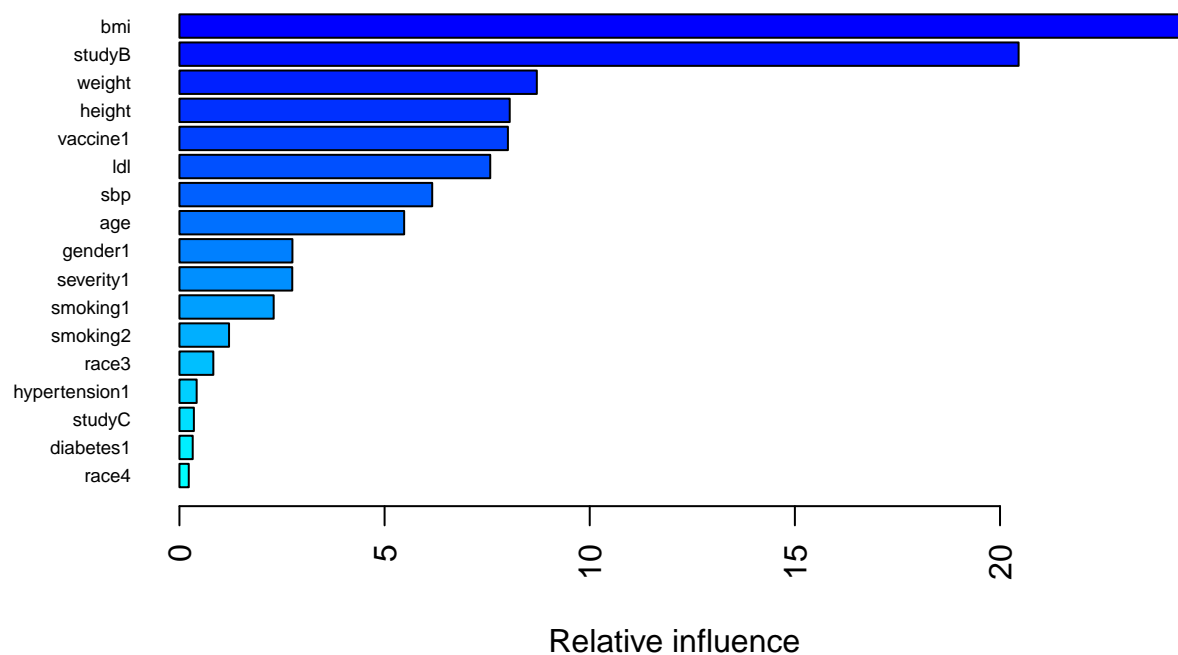


```
boost_fit$bestTune
```

```
##      n.trees interaction.depth shrinkage n.minobsinnode
## 106      500                4      0.01                5
```

```
variable importance
```

```
summary(boost_fit$finalModel, las = 2, cBars = 17, cex.names = 0.6)
```



```
##           var      rel.inf
## bmi          bmi 24.37052777
## studyB       studyB 20.45243388
## weight       weight 8.71029731
## height       height 8.04982577
## vaccine1     vaccine1 8.00572107
## ldl          ldl 7.57484863
## sbp          sbp 6.16092664
## age          age 5.47703636
## gender1      gender1 2.75357894
## severity1    severity1 2.74995110
## smoking1     smoking1 2.29700139
## smoking2     smoking2 1.20859525
## race3        race3 0.82622068
## hypertension1 hypertension1 0.41987580
## studyC       studyC 0.35222659
## diabetes1    diabetes1 0.32341450
## race4        race4 0.22773196
## race2        race2 0.03978636
```

```
trboost_prediction <- predict(boost_fit$finalModel, newdata = train_x, type = "response")
```

```
## Using 500 trees...
```

```

# As predictions are made for the positive class, we set a threshold 0.5 and assign anything above to b
trboost_prediction[trboost_prediction > 0.50] = "low"
trboost_prediction[trboost_prediction < 0.50] = "high"

boost_tr_mse <- mean(trboost_prediction != train_y)
boost_tr_mse

```

```
## [1] 0.2690443
```

```
boost_prediction <- predict(boost_fit$finalModel, newdata = test_x, type = "response")
```

```
## Using 500 trees...
```

```

# As predictions are made for the positive class, we set a threshold 0.5 and assign anything above to b
boost_prediction[boost_prediction > 0.50] = "low"
boost_prediction[boost_prediction < 0.50] = "high"

boost_ts_mse <- mean(boost_prediction != test_y)
boost_ts_mse

```

```
## [1] 0.2884882
```

### c. SVM

```

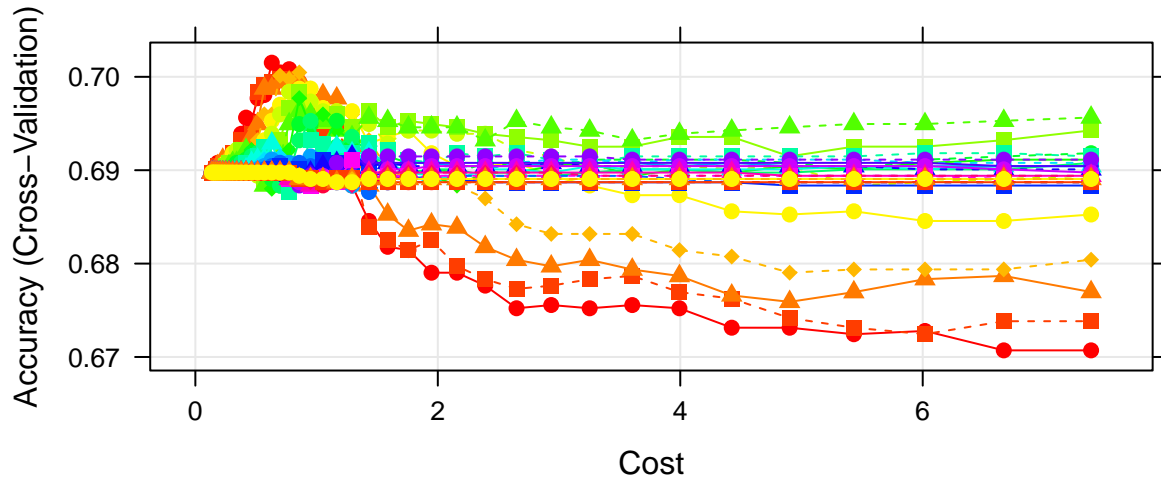
svmr_grid <- expand.grid(C = exp(seq(-2,2,len=40)),
                        sigma = exp(seq(-1,2,len=30)))
set.seed(2)

Mycluster = makeCluster(detectCores()-2)
registerDoParallel(Mycluster)
svmr_fit <- train(binary_recovery_time ~.,
                  data = training_set %>% select(-id, -recovery_time),
                  method = "svmRadialSigma",
                  tuneGrid = svmr_grid,
                  trControl = ctrl2)
stopCluster(Mycluster)
registerDoSEQ()

myCol<- rainbow(25)
myPar <- list(superpose.symbol = list(col = myCol),
              superpose.line = list(col = myCol))
plot(svmr_fit, highlight = TRUE, par.settings = myPar)

```

Sigma					
2	0.841630840067284	◆	1.92547446711558	■	4.
2	0.933358864311723	●	2.135328907083	▲	4.
2	1.03508418194327	●	2.36805505307725	◆	5.
7	1.14789638227654	■	2.62614565643902	●	6.
3	1.27300380725534	▲	2.91236514956502	●	6.
1	1.41174649411535	◆	3.22977925599986	■	7.
3	1.56561052864724	●	3.58178782768535	▲	
9	1.73624396279946	●	3.97216126109006	◆	



```
svmr_fit$bestTune
```

```
##          sigma          C
## 451 0.3678794 0.6303132
```

```
svm_train_prediction <- predict(svmr_fit$finalModel, newdata = train_x)
mean(svm_train_prediction!=train_y)
```

```
## [1] 0.2171053
```

```
svm_test_prediction <- predict(svmr_fit$finalModel, newdata = test_x)
```

```
mean(svm_test_prediction!=test_y)
```

```
## [1] 0.2884882
```

## 6. Comparison

### Resample

```

set.seed(2)
resamp <- resamples(list(
  Logistic = model.glm,
  Random_Forest = boost_fit,
  SVM = svmr_fit))
summary(resamp)

```

```

##
## Call:
## summary.resamples(object = resamp)
##
## Models: Logistic, Random_Forest, SVM
## Number of resamples: 10
##
## Accuracy
##           Min.   1st Qu.   Median     Mean   3rd Qu.     Max. NA's
## Logistic    0.6608997 0.6825260 0.7019115 0.6991054 0.7124282 0.7370242    0
## Random_Forest 0.6747405 0.7004367 0.7214533 0.7129559 0.7253053 0.7361111    0
## SVM         0.6805556 0.6963668 0.7019115 0.7015132 0.7077206 0.7206897    0
##
## Kappa
##           Min.   1st Qu.   Median     Mean   3rd Qu.     Max. NA's
## Logistic    0.07408134 0.1262342 0.1821818 0.1656036 0.1868010 0.2744450    0
## Random_Forest 0.12480860 0.1715600 0.2130027 0.1966640 0.2263556 0.2424081    0
## SVM         0.02563818 0.0783677 0.1054410 0.1001623 0.1309254 0.1535135    0

```

```

bwplot(resamp)

```

