

# P8106 Final Project: Predicting COVID-19 Recovery Time and Identifying Significant Risk Factors

Runze Cui (rc3521), Yuchen Hua (yh3555), Hongpu Min (hm2946)

2023-05-01

## Contents

<b>Background</b>	<b>2</b>
<b>Data:</b>	<b>2</b>
<b>Exploratory Analysis and Data Visualization</b>	<b>3</b>
Exploratory Analysis . . . . .	3
Data Visualization . . . . .	5
<b>Primary Analysis</b>	<b>12</b>
Linear methods: . . . . .	12
Nonlinear Methods: . . . . .	20
<b>Secondary Analysis</b>	<b>28</b>

# Background

[Check the report]

## Data:

[Description check the report]

```
# For primary analysis:
# Dataset Loading:
load("data/recovery.Rdata")

set.seed(3521) # Runze Cui's uni(2183):
# Create a first random sample of 2000 participants:
dat1 <- dat[sample(1:10000, 2000),]

set.seed(3555) # Yuchen Hua's uni(3555)
# Create a second random sample of 2000 participants:
dat2 <- dat[sample(1:10000, 2000),]

# Merged the two datasets and remove repeated observations:
dat <- unique(rbind(dat1, dat2))

# Get rid of the id variable from the merged dataset and do the data cleaning:
dat <- dat %>%
  select(-id) %>%
  mutate(gender = as.factor(gender)) %>%
  mutate(race = as.factor(race)) %>%
  mutate(smoking = as.factor(smoking)) %>%
  mutate(hypertension = as.factor(hypertension)) %>%
  mutate(diabetes = as.factor(diabetes)) %>%
  mutate(vaccine = as.factor(vaccine)) %>%
  mutate(severity = as.factor(severity)) %>%
  mutate(study = as.factor(study)) %>%
  na.omit() %>%
  relocate(recovery_time)

head(dat)
```

	recovery_time	age	gender	race	smoking	height	weight	bmi	hypertension
## 8158	52	61	0	1	1	169.9	87.6	30.4	0
## 3387	24	60	1	1	2	173.4	70.6	23.5	0
## 1709	36	60	1	1	1	178.2	79.9	25.1	0
## 4051	23	70	1	4	0	167.4	77.7	27.7	1
## 954	24	63	1	4	0	175.4	88.7	28.8	1
## 531	36	65	0	1	0	160.4	74.4	28.9	1
	diabetes	SBP	LDL	vaccine	severity	study			
## 8158	0	118	103	0	0	C			
## 3387	0	129	101	1	0	B			
## 1709	0	130	107	1	0	A			
## 4051	0	145	128	1	0	B			
## 954	0	131	100	0	0	A			
## 531	0	137	153	1	0	A			

```

# Separate the data as training and test data:
set.seed(3521)
# Specify rows of training data:
trRows <- createDataPartition(dat$recovery_time, p = 0.7, list = FALSE)

# Training data:
training <- dat[trRows, ]
## Covariates' matrix:
x <- model.matrix(recovery_time ~ ., dat)[trRows, -1]
## Response's vector:
y <- dat$recovery_time[trRows]

# Test data:
test <- dat[-trRows, ]
## Covariates' matrix:
x2 <- model.matrix(recovery_time ~ ., dat)[-trRows, -1]
## Response's vector:
y2 <- dat$recovery_time[-trRows]

# For secondary analysis:
dat_2 <- dat %>%
  mutate(recovery_time = ifelse(recovery_time > 30, "great", "less")) %>%
  mutate(recovery_time = as.factor(recovery_time))

# Training data:
training_sec <- dat_2[trRows, ]
## Covariates' matrix:
x_sec <- model.matrix(recovery_time ~ ., dat_2)[trRows, -1]
## Response's vector:
y_sec <- dat_2$recovery_time[trRows]

# Test data:
test_sec <- dat_2[-trRows, ]
## Covariates' matrix:
x2_sec <- model.matrix(recovery_time ~ ., dat_2)[-trRows, -1]
## Response's vector:
y2_sec <- dat_2$recovery_time[-trRows]

```

## Exploratory Analysis and Data Visualization

[Description check the report]

### Exploratory Analysis

```

# Summary tables separated by continuous/categorical variables:
skimr::skim(dat)

```

Table 1: Data summary

Name	dat
Number of rows	3587
Number of columns	15
Column type frequency:	
factor	8
numeric	7
Group variables	None

**Variable type: factor**

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
gender	0	1	FALSE	2	0: 1847, 1: 1740
race	0	1	FALSE	4	1: 2332, 3: 731, 4: 350, 2: 174
smoking	0	1	FALSE	3	0: 2191, 1: 1044, 2: 352
hypertension	0	1	FALSE	2	0: 1817, 1: 1770
diabetes	0	1	FALSE	2	0: 3045, 1: 542
vaccine	0	1	FALSE	2	1: 2174, 0: 1413
severity	0	1	FALSE	2	0: 3236, 1: 351
study	0	1	FALSE	3	B: 2129, A: 737, C: 721

**Variable type: numeric**

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
recovery_time	0	1	43.27	29.57	2.0	28.0	39.0	50.0	365.0	
age	0	1	60.09	4.48	45.0	57.0	60.0	63.0	75.0	
height	0	1	169.94	6.00	149.7	165.9	169.9	173.9	189.6	
weight	0	1	79.93	7.02	57.2	75.2	80.0	84.7	105.7	
bmi	0	1	27.74	2.77	18.8	25.8	27.7	29.5	38.1	
SBP	0	1	130.28	7.96	102.0	125.0	130.0	136.0	158.0	
LDL	0	1	110.16	19.75	47.0	97.0	110.0	124.0	178.0	

```
skimr::skim(dat2)
```

Table 4: Data summary

Name	dat2
Number of rows	2000
Number of columns	16
Column type frequency:	
character	1
factor	2
numeric	13

Table 4: Data summary

Group variables	None
-----------------	------

**Variable type: character**

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
study	0	1	1	1	0	3	0

**Variable type: factor**

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
race	0	1	FALSE	4	1: 1281, 3: 411, 4: 207, 2: 101
smoking	0	1	FALSE	3	0: 1242, 1: 562, 2: 196

**Variable type: numeric**

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
id	0	1	4902.78	2915.29	1.0	2344.75	4849.5	7410.0	9995.0	
age	0	1	60.02	4.50	45.0	57.00	60.0	63.0	75.0	
gender	0	1	0.49	0.50	0.0	0.00	0.0	1.0	1.0	
height	0	1	170.03	6.03	151.9	166.00	169.9	174.1	189.6	
weight	0	1	79.85	7.00	57.2	75.20	79.9	84.5	100.6	
bmi	0	1	27.69	2.78	20.5	25.80	27.5	29.5	38.1	
hypertension	0	1	0.50	0.50	0.0	0.00	0.0	1.0	1.0	
diabetes	0	1	0.14	0.35	0.0	0.00	0.0	0.0	1.0	
SBP	0	1	130.36	8.04	102.0	125.00	130.0	136.0	155.0	
LDL	0	1	110.43	19.30	47.0	97.00	111.0	124.0	171.0	
vaccine	0	1	0.61	0.49	0.0	0.00	1.0	1.0	1.0	
severity	0	1	0.10	0.29	0.0	0.00	0.0	0.0	1.0	
recovery_time	0	1	42.41	28.90	2.0	28.00	38.0	49.0	365.0	

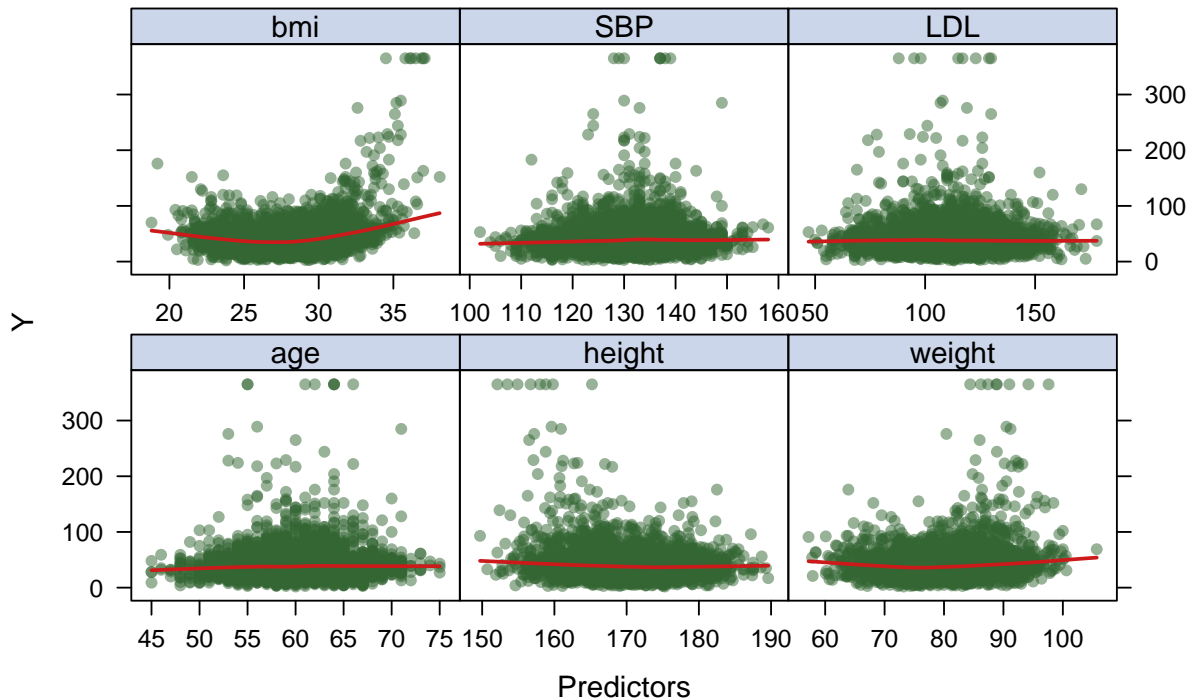
**Data Visualization**

```
# For primary analysis:
## For continuous variables:
theme = trellis.par.get()
theme$plot.symbol$col = rgb(.2, .4, .2, .5)
theme$plot.symbol$pch = 16
theme$plot.line$col = rgb(.8, .1, .1, 1)
theme$plot.line$lwd = 2
theme$strip.background$col = rgb(.0, .2, .6, .2)
trellis.par.set(theme)

featurePlot(x = dat %>% dplyr::select(age, height, weight, bmi, SBP, LDL),
            y = dat$recovery_time,
            plot = "scatter",
```

```
span = .5,
labels = c("Predictors", "Y"),
main = "Figure 1.1. Lattice Plots for Continuous Variables in Primary Analysis",
type = c("p", "smooth"))
```

**Figure 1.1. Lattice Plots for Continuous Variables in Primary Analysis**



```
## For categorical variables:
gender_plot = dat %>%
  ggplot(aes(x = gender, y = recovery_time, fill = gender)) +
  geom_violin(color = "black", alpha = .5) +
  scale_x_discrete(labels = c('Female', 'Male')) +
  ylab("Recovery") +
  theme(legend.position = "none")

race_plot = dat %>%
  ggplot(aes(x = race, y = recovery_time, fill = race)) +
  geom_violin(color = "black", alpha = .5) +
  scale_x_discrete(labels = c('White', 'Asian', 'Black', 'Hispanic')) +
  ylab("Recovery") +
  theme(legend.position = "none")

smoking_plot = dat %>%
  ggplot(aes(x = smoking, y = recovery_time, fill = smoking)) +
  geom_violin(color = "black", alpha = .5) +
  scale_x_discrete(labels = c('Never smoked', 'Former smoker', 'Current smoker')) +
  ylab("Recovery") +
  theme(legend.position = "none")
```

```

hyper_plot = dat %>%
  ggplot(aes(x = hypertension, y = recovery_time, fill = hypertension)) +
  geom_violin(color = "black", alpha = .5) +
  ylab("Recovery") +
  scale_x_discrete(labels = c('No', 'Yes')) +
  theme(legend.position = "none")

diabetes_plot = dat %>%
  ggplot(aes(x = diabetes, y = recovery_time, fill = diabetes)) +
  geom_violin(color = "black", alpha = .5) +
  scale_x_discrete(labels = c('No', 'Yes')) +
  ylab("Recovery") +
  theme(legend.position = "none")

vac_plot = dat %>%
  ggplot(aes(x = vaccine, y = recovery_time, fill = vaccine)) +
  geom_violin(color = "black", alpha = .5) +
  scale_x_discrete(labels = c('Not vaccinated', 'Vaccinated')) +
  ylab("Recovery") +
  theme(legend.position = "none")

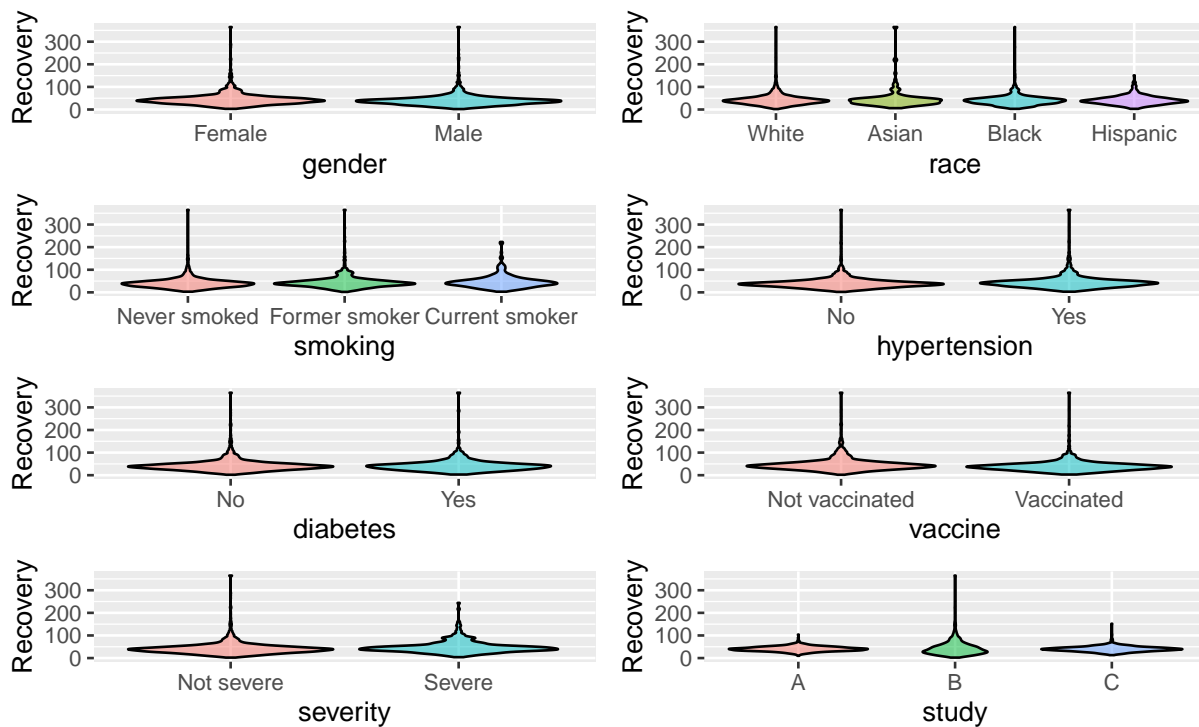
severity_plot = dat %>%
  ggplot(aes(x = severity, y = recovery_time, fill = severity)) +
  geom_violin(color = "black", alpha = .5) +
  scale_x_discrete(labels = c('Not severe', 'Severe')) +
  ylab("Recovery") +
  theme(legend.position = "none")

study_plot = dat %>%
  ggplot(aes(x = study, y = recovery_time, fill = study)) +
  geom_violin(color = "black", alpha = .5) +
  ylab("Recovery") +
  theme(legend.position = "none")

(gender_plot + race_plot + smoking_plot + hyper_plot) / (diabetes_plot + vac_plot + severity_plot + stu
  plot_layout(guides = "collect") +
  plot_annotation(title = "Figure 1.2. Violin Plots for Categorical Variables in Primary Analysis")

```

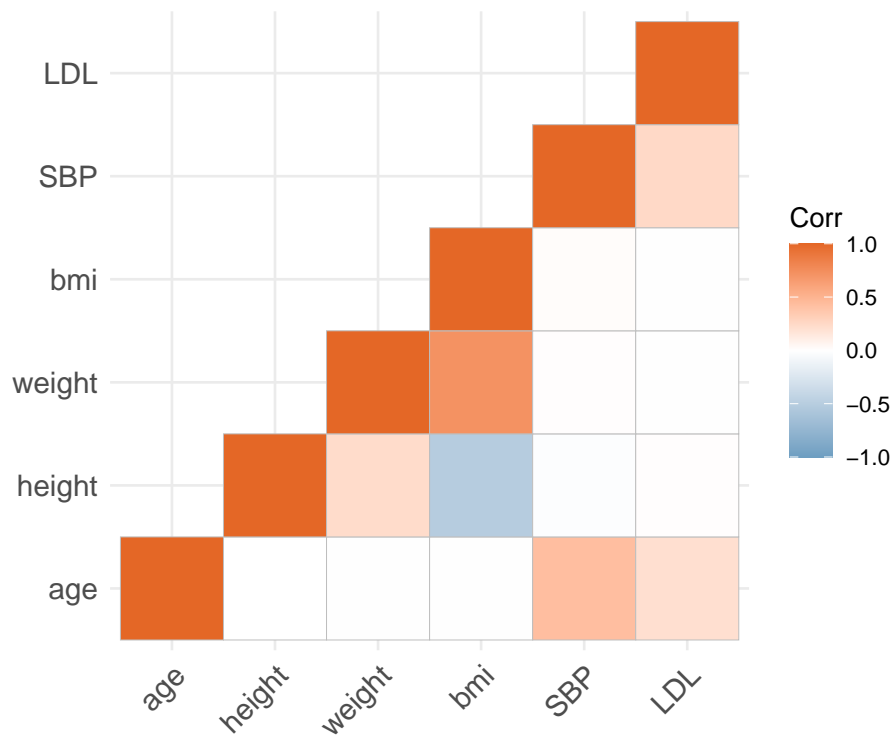
Figure 1.2. Violin Plots for Categorical Variables in Primary Analysis



```
## Correlation matrix for continuous variables ONLY:
model.matrix( ~ 0+., data = dat %>% dplyr::select(age, height, weight, bmi, SBP, LDL)) %>%
  cor(use = "pairwise.complete.obs") %>%
  ggcorrplot::ggcorrplot(show.diag = T,
                          type = "lower",
                          lab = F,
                          colors = c("#6D9EC1", "white", "#E46726")) +
  ggtitle("Figure 1.3. Correlation matrix for continuous datasat")
```



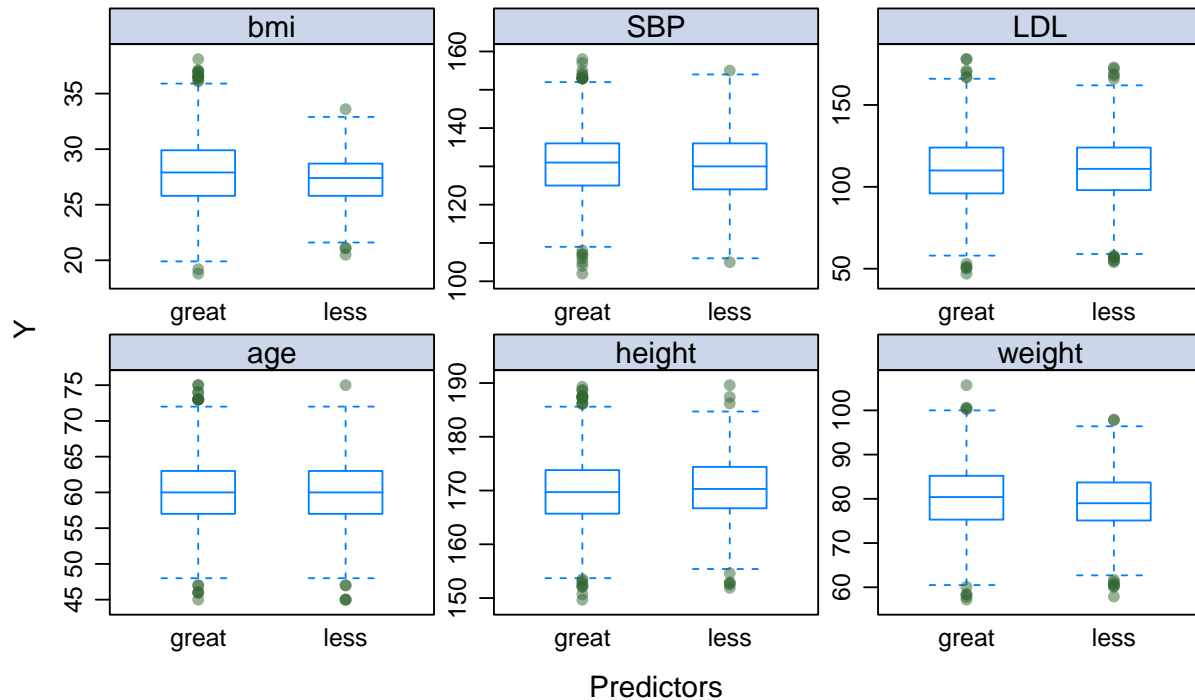
Figure 1.3. Correlation matrix for continuous dataset



```
# For secondary analysis:
## For continuous variables:
theme = trellis.par.get()
theme$plot.symbol$col = rgb(.2, .4, .2, .5)
theme$plot.symbol$pch = 16
theme$plot.line$col = rgb(.8, .1, .1, 1)
theme$plot.line$lwd = 2
theme$strip.background$col = rgb(.0, .2, .6, .2)
trellis.par.set(theme)

featurePlot(x = dat_2 %>% dplyr::select(age, height, weight, bmi, SBP, LDL),
            y = dat_2$recovery_time,
            plot = "box", pch = "|",
            scales = list(x = list(relation = "free"),
                          y = list(relation = "free")),
            labels = c("Predictors", "Y"),
            main = "Figure 1.4. Lattice Plots for Continuous Variables in Secondary Analysis",
            auto.key = list(columns = 2))
```

**Figure 1.4. Lattice Plots for Continuous Variables in Secondary Analysis**



```
## For categorical variables:
gender_plot_sec = dat_2 %>%
  ggplot(aes(x = gender, fill = recovery_time)) +
  geom_bar(color = "black", alpha = .5) +
  scale_x_discrete(labels = c('Female', 'Male')) +
  ylab("Recovery") +
  theme(legend.position = "none")

race_plot_sec = dat_2 %>%
  ggplot(aes(x = race, fill = recovery_time)) +
  geom_bar(color = "black", alpha = .5) +
  scale_x_discrete(labels = c('White', 'Asian', 'Black', 'Hispanic')) +
  ylab("Recovery") +
  theme(legend.position = "none")

smoking_plot_sec = dat_2 %>%
  ggplot(aes(x = smoking, fill = recovery_time)) +
  geom_bar(color = "black", alpha = .5) +
  scale_x_discrete(labels = c('Never smoked', 'Former smoker', 'Current smoker')) +
  ylab("Recovery") +
  theme(legend.position = "none")

hyper_plot_sec = dat_2 %>%
  ggplot(aes(x = hypertension, fill = recovery_time)) +
  geom_bar(color = "black", alpha = .5) +
  ylab("Recovery") +
  scale_x_discrete(labels = c('No', 'Yes')) +
```

```

theme(legend.position = "none")

diabetes_plot_sec = dat_2 %>%
  ggplot(aes(x = diabetes, fill = recovery_time)) +
  geom_bar(color = "black", alpha = .5) +
  scale_x_discrete(labels = c('No', 'Yes')) +
  ylab("Recovery") +
  theme(legend.position = "none")

vac_plot_sec = dat_2 %>%
  ggplot(aes(x = vaccine, fill = recovery_time)) +
  geom_bar(color = "black", alpha = .5) +
  scale_x_discrete(labels = c('Not vaccinated', 'Vaccinated')) +
  ylab("Recovery") +
  theme(legend.position = "none")

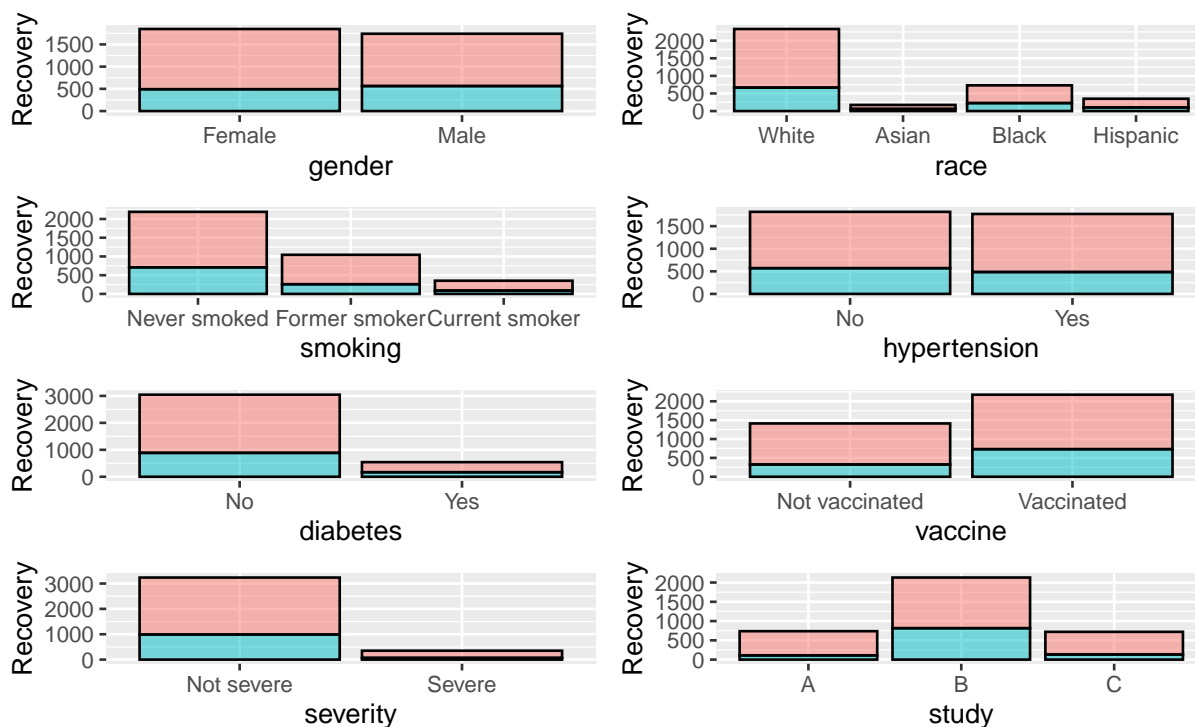
severity_plot_sec = dat_2 %>%
  ggplot(aes(x = severity, fill = recovery_time)) +
  geom_bar(color = "black", alpha = .5) +
  scale_x_discrete(labels = c('Not severe', 'Severe')) +
  ylab("Recovery") +
  theme(legend.position = "none")

study_plot_sec = dat_2 %>%
  ggplot(aes(x = study, fill = recovery_time)) +
  geom_bar(color = "black", alpha = .5) +
  ylab("Recovery") +
  theme(legend.position = "none")

(gender_plot_sec + race_plot_sec + smoking_plot_sec + hyper_plot_sec) / (diabetes_plot_sec + vac_plot_sec)
plot_layout(guides = "collect") +
plot_annotation(title = "Figure 1.5. Bar Plots for Categorical Variables in Secondary Analysis")

```

Figure 1.5. Bar Plots for Categorical Variables in Secondary Analysis



Note: Red is recovery time less than and equal to 30. Blue is greater than 30.

ctrl and parallel computing setup:

```
ctrl = trainControl(method = "repeatedcv", number = 10, repeats = 5)
ctrl1 = trainControl(method = "repeatedcv",
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE)
no_cores = detectCores() - 1
```

## Primary Analysis

Recovery time as continuous variable.

### Linear methods:

Linear model:

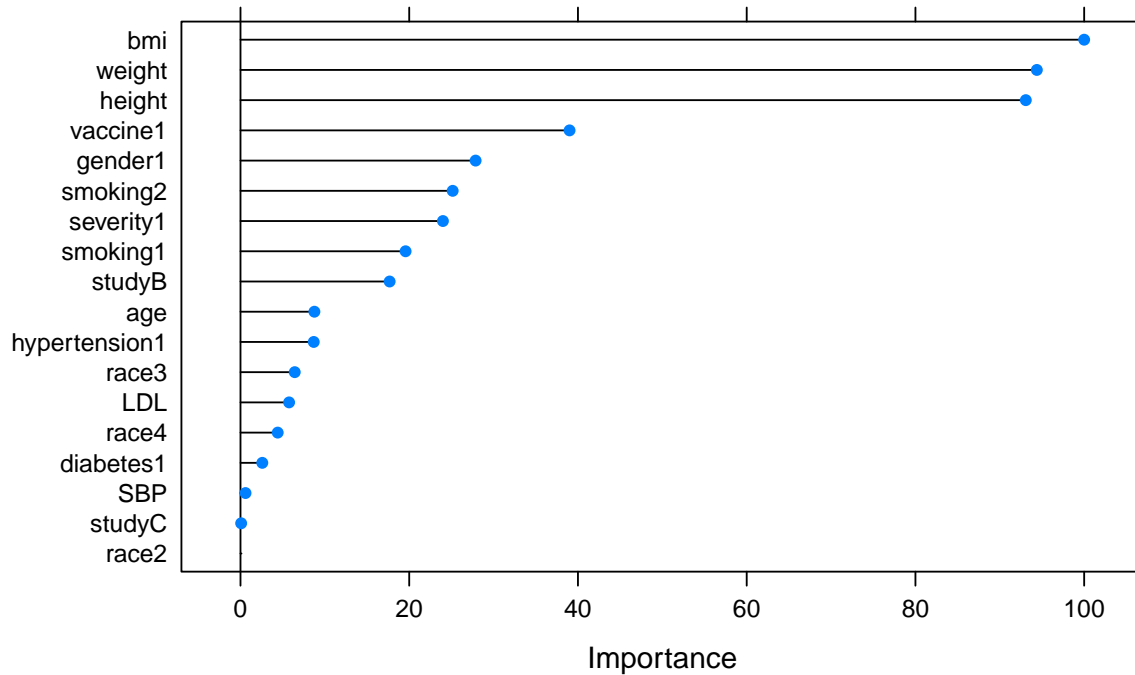
```
set.seed(3521)
# Fit a linear regression model:
lm = train(recovery_time ~ age + gender + race + smoking + height +
           weight + bmi + hypertension + diabetes + SBP +
           LDL + vaccine + severity + study,
           data = training, ## use training dataset
           method = "lm",
```

```

trControl = ctrl)
summary(lm)
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -62.043 -13.881  -1.317   10.006  236.072
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.460e+03  1.398e+02 -17.598 < 2e-16 ***
## age          2.353e-01  1.205e-01   1.952 0.051045 .
## gender1     -5.201e+00  9.521e-01  -5.462 5.17e-08 ***
## race2        7.627e-01  2.226e+00   0.343 0.731920
## race3     -1.850e+00  1.214e+00  -1.524 0.127587
## race4     -1.890e+00  1.639e+00  -1.153 0.248913
## smoking1     4.200e+00  1.067e+00   3.938 8.46e-05 ***
## smoking2     8.195e+00  1.651e+00   4.963 7.43e-07 ***
## height      1.433e+01  8.213e-01  17.442 < 2e-16 ***
## weight     -1.541e+01  8.717e-01 -17.682 < 2e-16 ***
## bmi         4.676e+01  2.499e+00  18.713 < 2e-16 ***
## hypertension1 3.050e+00  1.574e+00   1.938 0.052794 .
## diabetes1    -1.085e+00  1.328e+00  -0.817 0.413725
## SBP         -4.700e-02  1.039e-01  -0.452 0.651026
## LDL        -3.455e-02  2.466e-02  -1.401 0.161337
## vaccine1    -7.333e+00  9.768e-01  -7.508 8.33e-14 ***
## severity1     7.807e+00  1.644e+00   4.750 2.15e-06 ***
## studyB       4.365e+00  1.216e+00   3.590 0.000337 ***
## studyC     -5.355e-01  1.499e+00  -0.357 0.720955
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 23.81 on 2494 degrees of freedom
## Multiple R-squared:  0.2355, Adjusted R-squared:  0.23
## F-statistic: 42.68 on 18 and 2494 DF,  p-value: < 2.2e-16
# Importances:
plot(varImp(lm, scale = TRUE), main = "Figure 2. Linear Model Variable's Importance Plot")

```

**Figure 2. Linear Model Variable's Importance Plot**



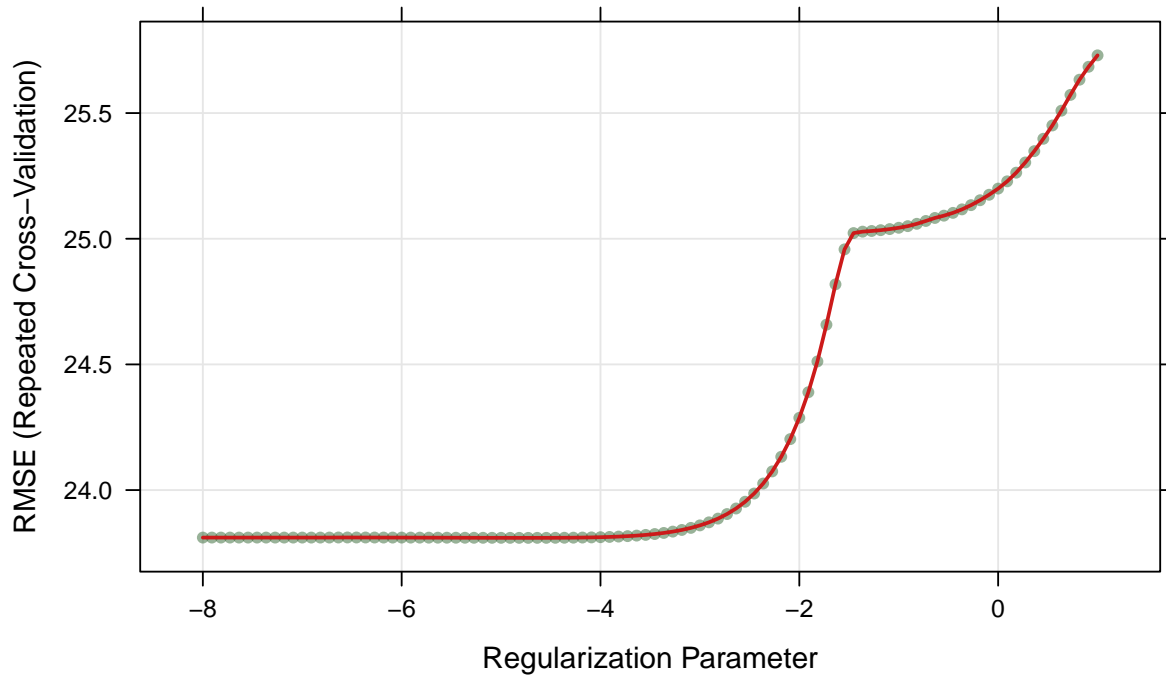
```
# Report the CV error:
lm$results$RMSE
## [1] 23.81421
# Report the test error:
lm_pred = predict(lm, newdata = test) ## Test dataset
lm_mse = sqrt(mean((lm_pred - y2)^2))
lm_mse
## [1] 29.06371
```

[Describe the outputs above]

**LASSO model:**

```
set.seed(3521)
# Fit a LASSO model:
lasso = train(x = x,
              y = y, ## training dataset
              method = "glmnet",
              tuneGrid = expand.grid(alpha = 1,
                                    lambda = exp(seq(-8, 1, length = 100))),
              trControl = ctrl)
# Plot for tuning parameter selection:
plot(lasso, xTrans = log, main = "Figure 3. LASSO Model CV RMSE Plot", highlight = T)
```

**Figure 3. LASSO Model CV RMSE Plot**



```
# Choose best tuning parameter value:
lasso$bestTune
##      alpha      lambda
## 37      1 0.008850576
# Report the coefficients after applying the best tuning parameter:
coef(lasso$finalModel, lasso$bestTune$lambda)
## 19 x 1 sparse Matrix of class "dgCMatrix"
##              s1
## (Intercept) -2.326105e+03
## age         2.302849e-01
## gender1     -5.174310e+00
## race2       6.917049e-01
## race3      -1.881002e+00
## race4      -1.852093e+00
## smoking1    4.170664e+00
## smoking2    8.130377e+00
## height     1.353195e+01
## weight     -1.457152e+01
## bmi        4.434163e+01
## hypertension1 2.956707e+00
## diabetes1   -1.028034e+00
## SBP        -3.873666e-02
## LDL        -3.421155e-02
## vaccine1   -7.301952e+00
## severity1   7.790966e+00
## studyB     4.368825e+00
## studyC     -4.916229e-01
```

```

# Report the CV error:
lasso$results[37, ]$RMSE
## [1] 23.80994
# Report the test error:
lasso_pred = predict(lasso, newdata = x2) ## Test dataset
lasso_mse = sqrt(mean((lasso_pred - y2)^2))
lasso_mse
## [1] 29.16892

```

[Describe the outputs above]

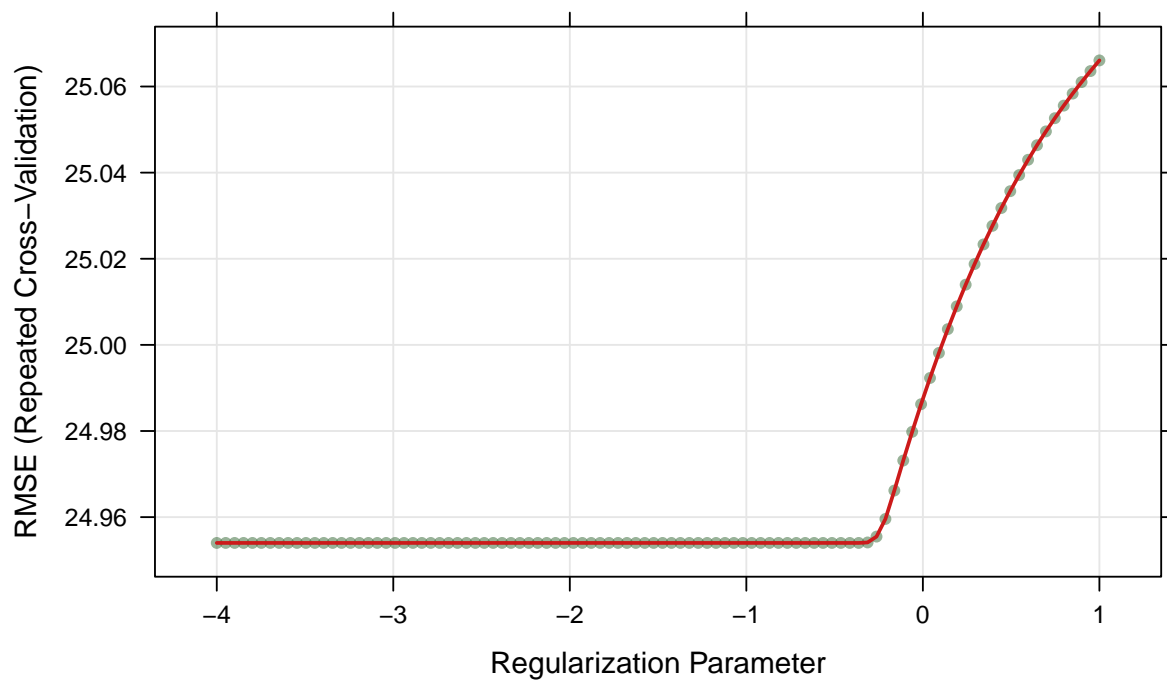
**Ridge model:**

```

set.seed(3521)
# Fit a ridge model:
ridge = train(x = x,
              y = y, ## training dataset
              method = "glmnet",
              tuneGrid = expand.grid(alpha = 0,
                                     lambda = exp(seq(-4, 1, length = 100))),
              trControl = ctrl)
# Plot for tuning parameter selection:
plot(ridge, xTrans = log, main = "Figure 4. Ridge Model CV RMSE Plot", highlight = T)

```

**Figure 4. Ridge Model CV RMSE Plot**





```

# Choose best tuning parameter value:
ridge$bestTune
##      alpha      lambda
## 73      0 0.6951439
# Report the coefficients after applying the best tuning parameter:
coef(ridge$finalModel, ridge$bestTune$lambda)
## 19 x 1 sparse Matrix of class "dgCMatrix"
##              s1
## (Intercept) -1.149931e+02
## age         2.064306e-01
## gender1     -4.932110e+00
## race2       1.218372e-01
## race3      -2.740022e+00
## race4      -1.799371e+00
## smoking1    4.007457e+00
## smoking2    7.504769e+00
## height     4.750406e-01
## weight     -6.857228e-01
## bmi        4.479836e+00
## hypertension1 2.800025e+00
## diabetes1   -4.992704e-01
## SBP        -2.006797e-04
## LDL        -3.530830e-02
## vaccine1   -6.887245e+00
## severity1   7.819336e+00
## studyB     4.395135e+00
## studyC    -2.104760e-01

# Report the CV error:
ridge$results[73, ]$RMSE
## [1] 24.95401
# Report the test error:
ridge_pred = predict(ridge, newdata = x2) ## Test dataset
ridge_mse = sqrt(mean((ridge_pred - y2)^2))
ridge_mse
## [1] 32.00583

```

[Describe the outputs above]

**Elastic net model:**

```

set.seed(3521)
# Fit a Elastic net model:
enet = train(x, y, ## training dataset
            method = "glmnet",
            tuneGrid = expand.grid(alpha = seq(0, 1, length = 21),
                                   lambda = exp(seq(-8, -2, length = 50))),
            trControl = ctrl)
# Plot for tuning parameters selection:
myCol = rainbow(25)
myPar = list(supерpose.symbol = list(col = myCol),

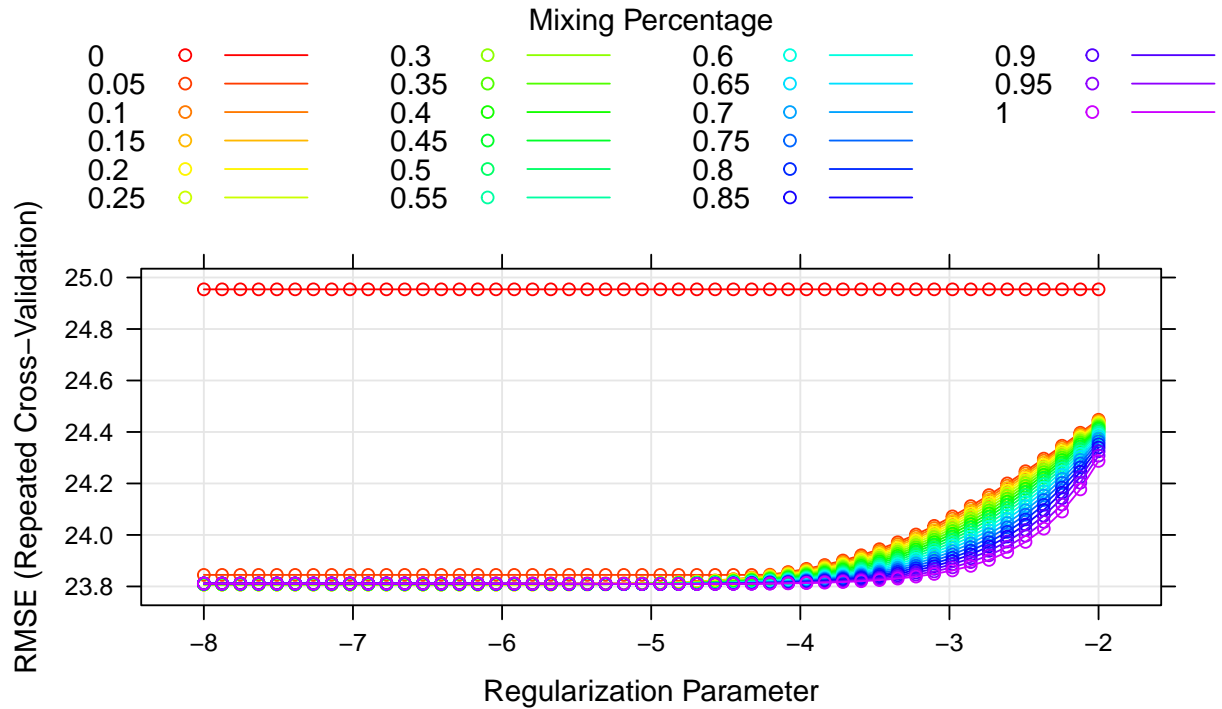
```

```

superpose.line = list(col = myCol))
plot(enet, par.settings = myPar, main = "Figure 5. Elastic Net Model CV RMSE Plot", xTrans = log)

```

**Figure 5. Elastic Net Model CV RMSE Plot**



```

# Choose best tuning parameter value:
enet$bestTune
##      alpha      lambda
## 220  0.2 0.003435924
# Report the coefficients after applying the best tuning parameter:
coef(enet$finalModel, enet$bestTune$lambda)
## 19 x 1 sparse Matrix of class "dgCMatrix"
##              s1
## (Intercept) -2.282729e+03
## age         2.333649e-01
## gender1     -5.189117e+00
## race2       7.082009e-01
## race3      -1.922687e+00
## race4      -1.886490e+00
## smoking1    4.193610e+00
## smoking2    8.158687e+00
## height     1.327902e+01
## weight     -1.430227e+01
## bmi        4.356889e+01
## hypertension1 3.037924e+00
## diabetes1   -1.039260e+00
## SBP        -4.372863e-02
## LDL        -3.464473e-02
## vaccine1   -7.312501e+00

```

```
## severity1      7.822090e+00
## studyB        4.381763e+00
## studyC        -4.985479e-01

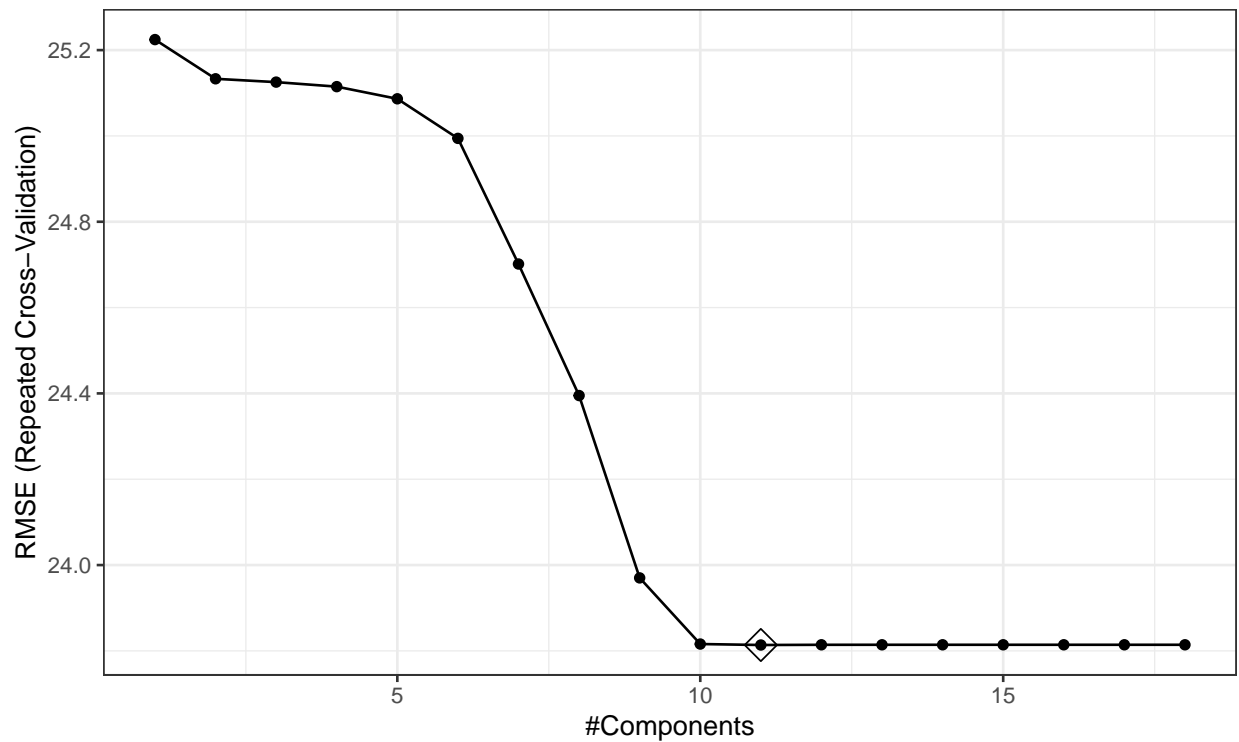
# Report the CV error:
enet$results[220, ]$RMSE
## [1] 23.80708
# Report the test error:
enet_pred = predict(enet, newdata = x2) ## Test dataset
enet_mse = sqrt(mean((enet_pred - y2)^2))
enet_mse
## [1] 29.20678
```

[Describe the outputs above]

Partial least squares model (PLS):

```
set.seed(3521)
# Fit a PLS model:
pls = train(x, y, ## training dataset
            method = "pls",
            tuneGrid = data.frame(ncomp = 1:18),
            trControl = ctrl,
            preprocess = c("center", "scale"))
summary(pls)
## Data:      X dimension: 2513 18
## Y dimension: 2513 1
## Fit method: oscorespls
## Number of components considered: 11
## TRAINING: % variance explained
##           1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X           9.44   16.31   26.35   31.00   35.86   41.51   45.81
## .outcome    12.45   13.46   13.57   13.71   13.99   14.69   16.91
##           8 comps  9 comps 10 comps 11 comps
## X           49.91   52.62   56.45   61.63
## .outcome    19.43   22.34   23.49   23.55
# Choose best tuning parameter value:
pls$bestTune
##      ncomp
## 11      11
# Plot for the number of components:
ggplot(pls, highlight = T) +
  theme_bw() +
  ggtitle("Figure 6. PLS Model CV RMSE Plot")
```

Figure 6. PLS Model CV RMSE Plot



```
# Report the CV error:
pls$results[11, ]$RMSE
## [1] 23.81368
# Report the test error:
pls_pred = predict(pls, newdata = x2) ## Test dataset
pls_mse = sqrt(mean((pls_pred - y2)^2))
pls_mse
## [1] 29.06252
```

[Describe the outputs above]

## Nonlinear Methods:

### Generalized additive model (GAM):

```
set.seed(3521)
# Fit a GAM model for all predictors:
gam = train(x, y, ## training dataset
            method = "gam",
            trControl = ctrl)
summary(gam)
##
## Family: gaussian
## Link function: identity
##
```

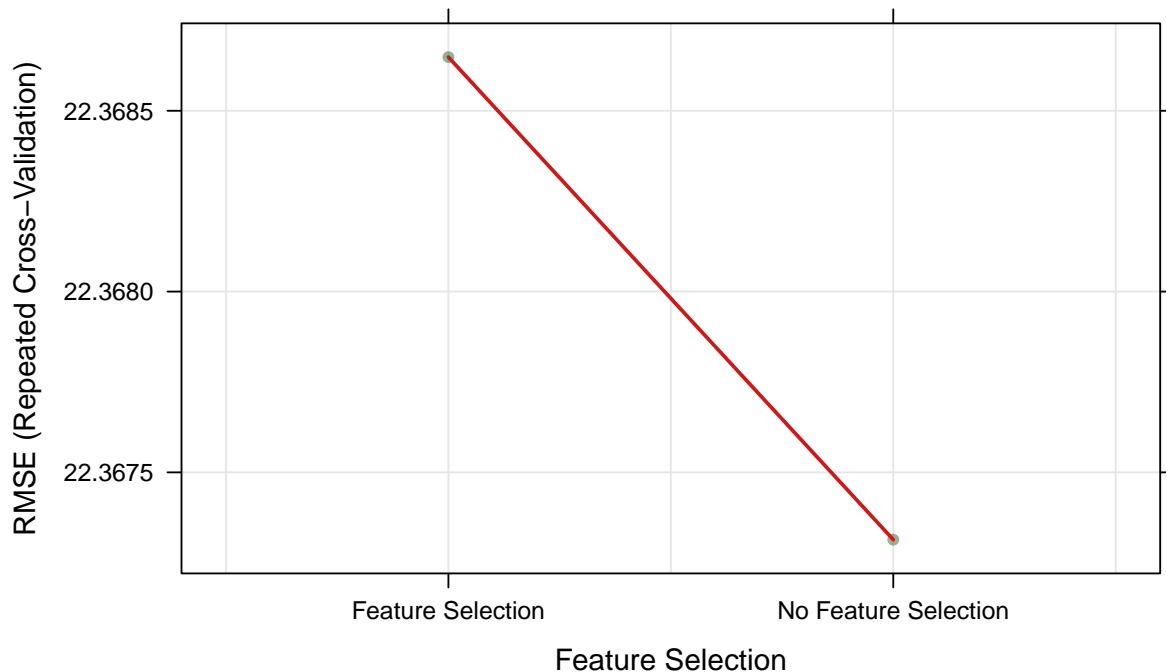
```

## Formula:
## .outcome ~ gender1 + race3 + race4 + smoking1 + smoking2 + hypertension1 +
##      diabetes1 + vaccine1 + severity1 + studyB + studyC + s(age) +
##      s(SBP) + s(LDL) + s(bmi) + s(height) + s(weight)
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  43.2614      1.5011  28.820 < 2e-16 ***
## gender1      -4.8382      0.8761  -5.523 3.68e-08 ***
## race3        -1.9715      1.1066  -1.782 0.074945 .
## race4        -2.2377      1.5006  -1.491 0.136053
## smoking1      4.7304      0.9809   4.823 1.50e-06 ***
## smoking2      8.4125      1.5209   5.531 3.51e-08 ***
## hypertension1 4.0622      1.5700   2.587 0.009728 **
## diabetes1     -0.1520      1.2212  -0.124 0.900971
## vaccine1      -7.4441      0.8996  -8.275 < 2e-16 ***
## severity1      8.0127      1.5105   5.305 1.23e-07 ***
## studyB         3.9614      1.1211   3.533 0.000418 ***
## studyC        -1.1218      1.3803  -0.813 0.416468
## ---
## 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)        1.000  1.000  3.717 0.053976 .
## s(SBP)        2.033  2.564  1.853 0.161840
## s(LDL)        1.000  1.000  0.929 0.335229
## s(bmi)        8.079  8.737 84.927 < 2e-16 ***
## s(height)     1.000  1.000  0.591 0.442071
## s(weight)     6.720  7.840  3.688 0.000309 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.351   Deviance explained = 35.9%
## GCV = 483.87   Scale est. = 477.74      n = 2513
# Report the model:
gam$bestTune
##      select method
## 1 FALSE GCV.Cp
gam$finalModel
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ gender1 + race3 + race4 + smoking1 + smoking2 + hypertension1 +
##      diabetes1 + vaccine1 + severity1 + studyB + studyC + s(age) +
##      s(SBP) + s(LDL) + s(bmi) + s(height) + s(weight)
##
## Estimated degrees of freedom:
## 1.00 2.03 1.00 8.08 1.00 6.72 total = 31.83
##
## GCV score: 483.8652

```

```
# Plot for tuning parameter selection:
plot(gam, main = "Figure 7. GAM Model CV RMSE Plot")
```

**Figure 7. GAM Model CV RMSE Plot**



```
# Report the CV error:
gam$results$RMSE
## [1] 22.36731 22.36865
# Report the test error:
gam_pred = predict(gam, newdata = x2) ## Test dataset
gam_mse = sqrt(mean((gam_pred - y2)^2))
gam_mse
## [1] 25.85336
```

[Describe the outputs above]

Multivariate adaptive regression spline (MARS) model:

```
set.seed(3521)
# Create grid for two tuning parameters in MARS
mars_grid = expand.grid(degree = 1:3, ## number of possible product hinge functions
                        nprune = 1:20) ## the number of basis functions to be retained after the pruning
# Fit the MARS model:
cl = makePSOCKcluster(no_cores)
registerDoParallel(cl)
mars = train(x, y, # training dataset
```

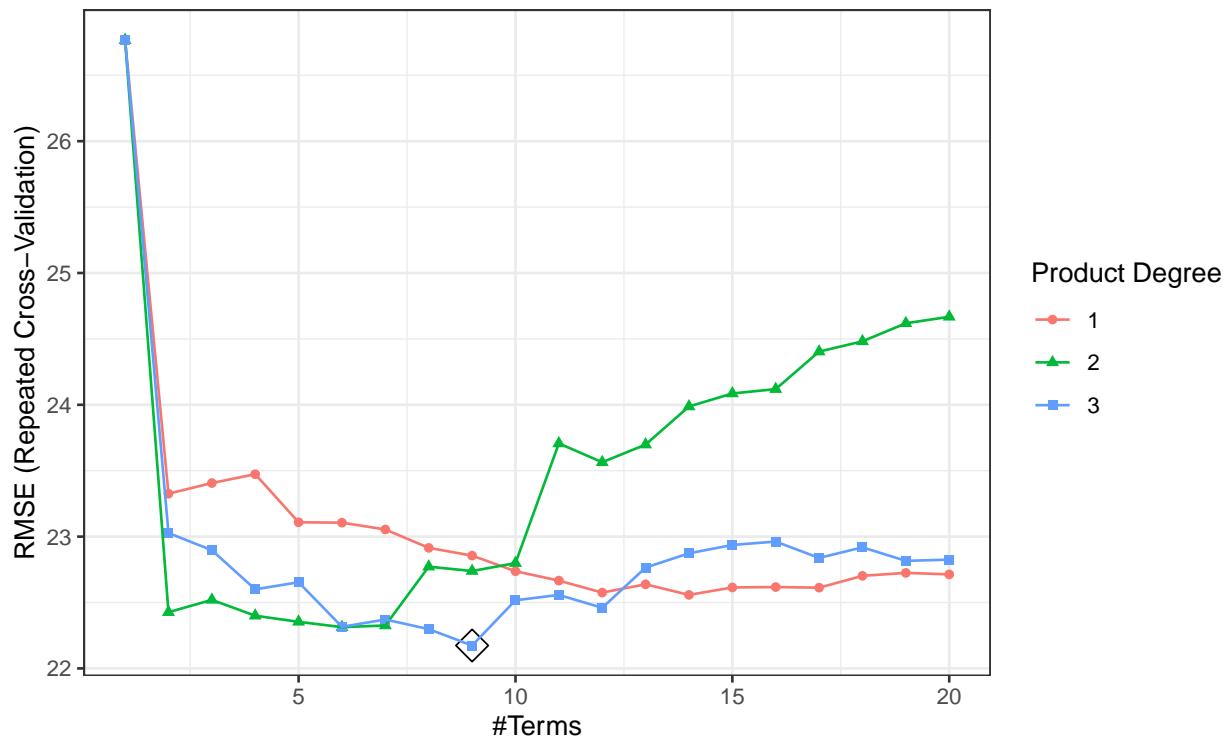
```

        method = "earth",
        tuneGrid = mars_grid,
        trControl = ctrl)
stopCluster(cl)
registerDoSEQ()

# Report the coefficients:
coef(mars$finalModel)
##              (Intercept)              h(30.5-bmi)
##              12.631786              5.164229
##              h(bmi-30.5) * studyB      race2 * h(bmi-30.5) * studyB
##              22.177779              22.832642
##              vaccine1              h(bmi-25.4)
##              -7.057776              6.227136
## h(bmi-30.5) * h(SBP-128) * studyB h(bmi-30.5) * h(SBP-130) * studyB
##              24.889685              -18.901047
## h(bmi-30.5) * h(SBP-124) * studyB
##              -7.493282
# Plot for tuning parameter selection:
ggplot(mars, highlight = T) +
  theme_bw() +
  ggtitle("Figure 8. MARS Model CV RMSE Plot")

```

Figure 8. MARS Model CV RMSE Plot



```

# Choose best tuning parameter value:
mars$bestTune
##      nprune degree
## 49      9      3

```

```

# Report the CV error:
mars$results[27, ]$RMSE
## [1] 22.17363
# Report test error:
mars_pred = predict(mars, newdata = x2) ## Test dataset
mars_mse = sqrt(mean((mars_pred - y2)^2))
mars_mse
## [1] 27.02613

```

[Describe the outputs above]

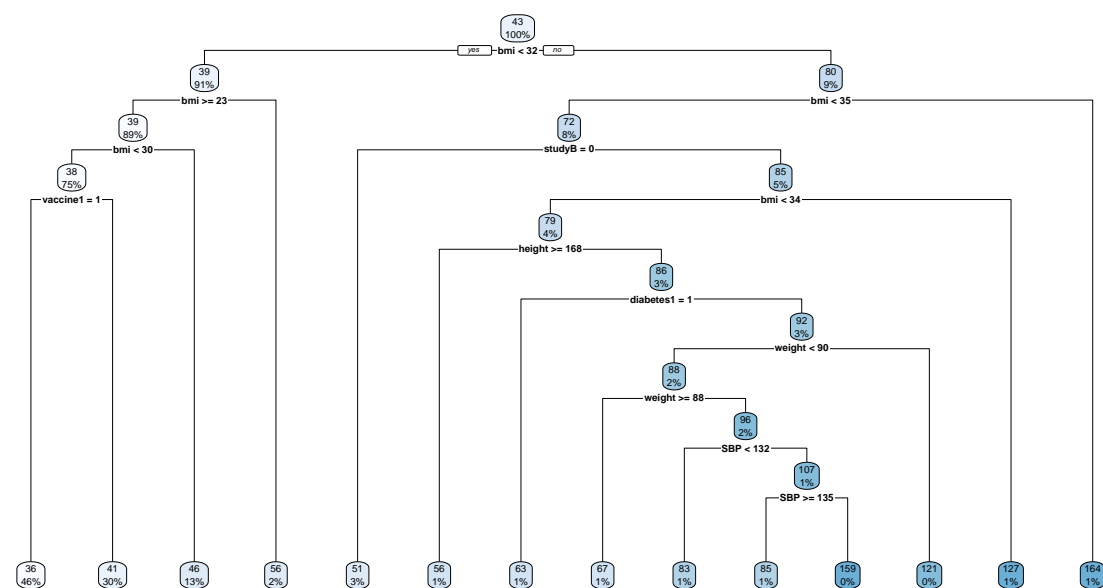
## Regression Tree:

```

set.seed(3521)
# Fit the regression tree model:
cl = makePSOCKcluster(no_cores)
registerDoParallel(cl)
reg_tree = train(x, y, ## training dataset
                 method = "rpart",
                 tuneGrid = data.frame(cp = exp(seq(-6, -2, length = 50))),
                 trControl = ctrl)
stopCluster(cl)
registerDoSEQ()
# Plot the regression tree:
rpart.plot(reg_tree$finalModel, main = "Figure 9.1. Regression Tree")

```

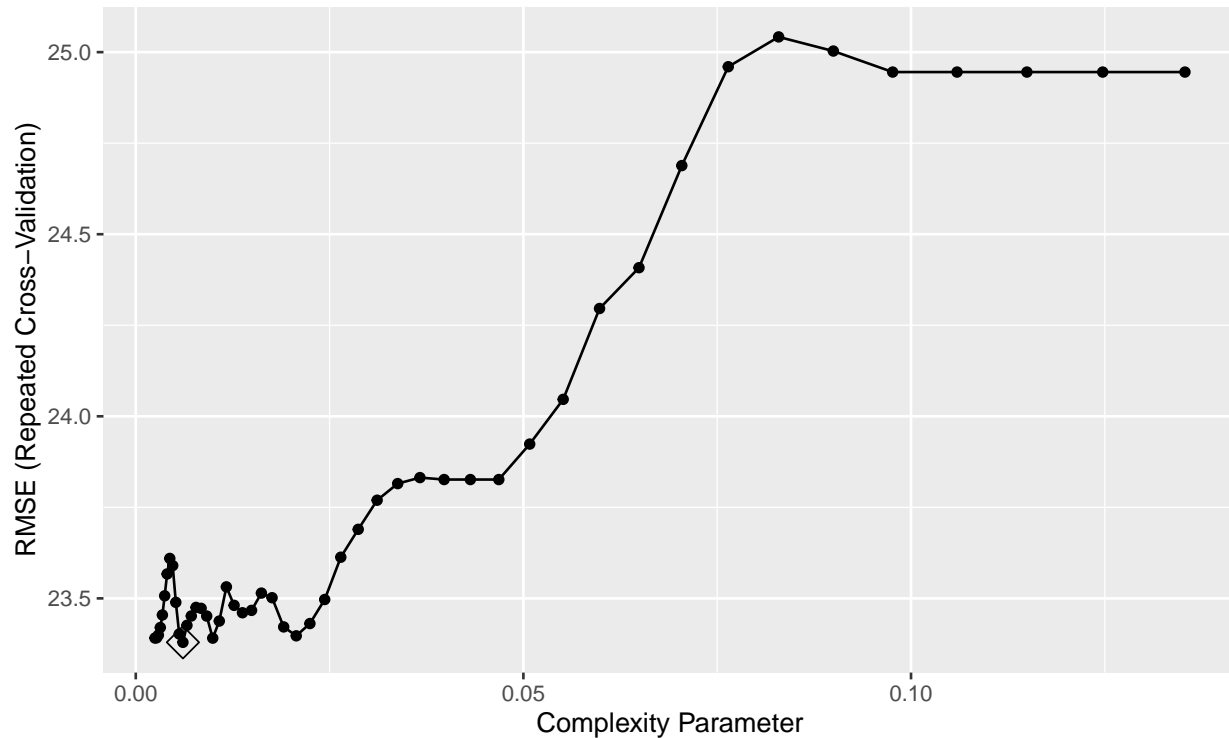
Figure 9.1. Regression Tree





```
# Plot for tuning parameter selection:
ggplot(reg_tree, highlight = TRUE) + ggtitle("Figure 9.2 Regression Tree Model CV RMSE Plot")
```

Figure 9.2 Regression Tree Model CV RMSE Plot



```
# Choose best tuning parameter value:
reg_tree$bestTune
##          cp
## 12 0.006084317

# Report the CV error:
reg_tree$results[12, ]$RMSE
## [1] 23.37935

# Report test error:
reg_tree_pred = predict(reg_tree, newdata = x2) ## Test dataset
reg_tree_mse = sqrt(mean((reg_tree_pred - y2)^2))
reg_tree_mse
## [1] 26.16509
```

[Describe the outputs above]

## Random Forests:

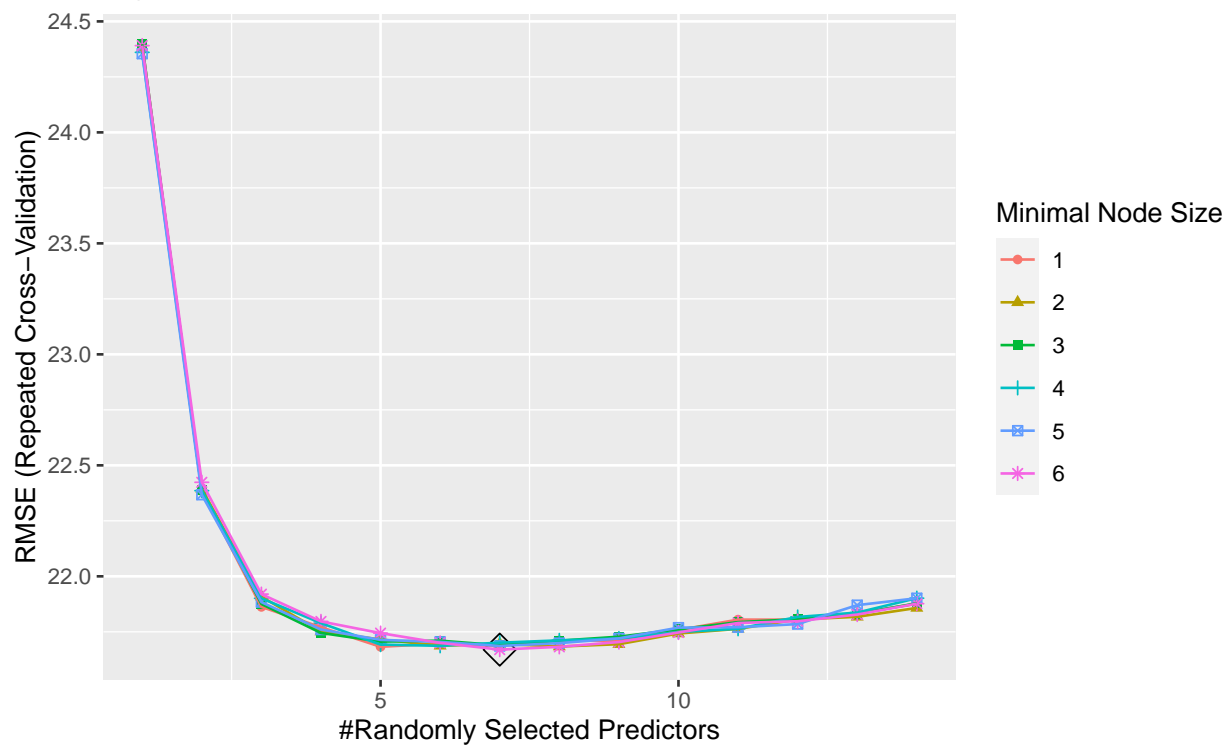
```
set.seed(3521)
# Fit the random forest model:
rf.grid = expand.grid(mtry = 1:14, ## We have 14 predictors in training data in total
                     splitrule = "variance",
```

```

min.node.size = 1:6) ## large/small model flexibility
set.seed(3521)
cl = makePSOCKcluster(no_cores)
registerDoParallel(cl)
rf = train(x, y,
          method = "ranger",
          tuneGrid = rf.grid,
          trControl = ctrl)
stopCluster(cl)
registerDoSEQ()
# Plot for tuning parameter selection:
ggplot(rf, highlight = TRUE) + ggtitle("Figure 10. Random Forests CV RMSE Plot")

```

Figure 10. Random Forests CV RMSE Plot



```

# Choose best tuning parameter value:
rf$bestTune
##      mtry splitrule min.node.size
## 42      7  variance              6

# Report the CV error:
rf$results[42, ]$RMSE
## [1] 21.6698

# Report test error:
rf_pred = predict(rf, newdata = x2) ## Test dataset
rf_mse = sqrt(mean((rf_pred - y2)^2))
rf_mse
## [1] 25.76281

```

[Describe the outputs above]

### Boosting:

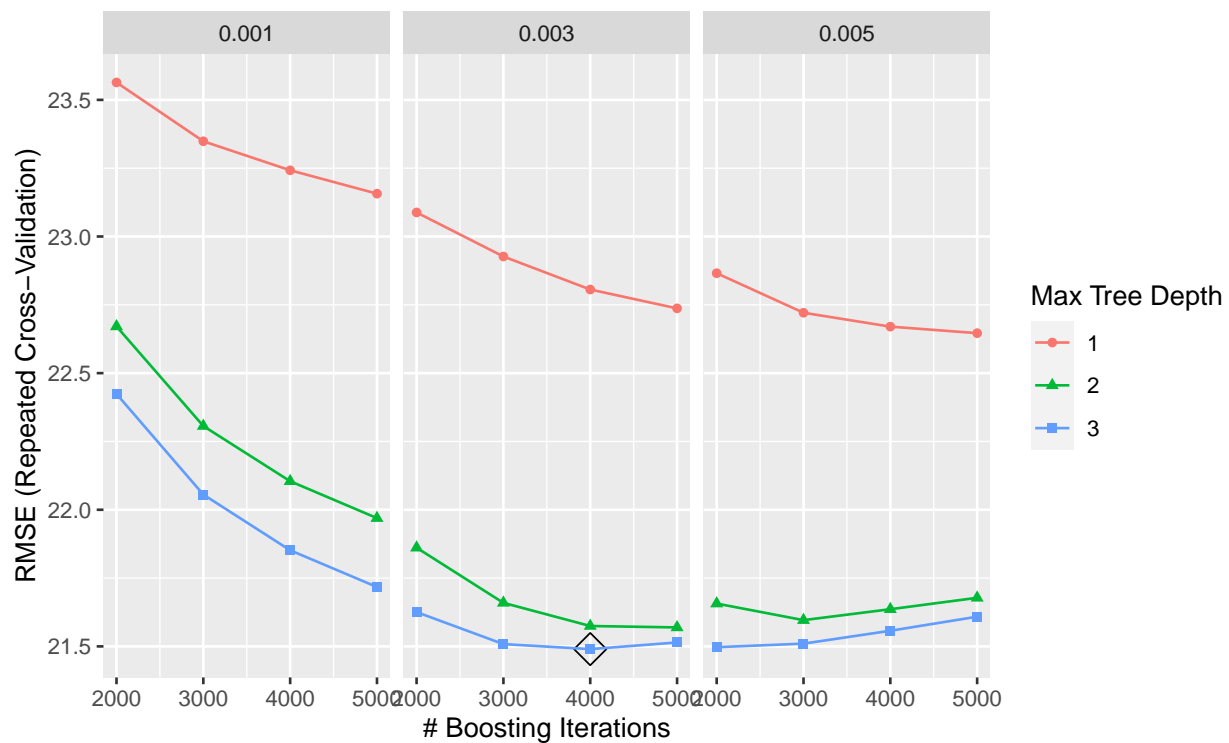
```
set.seed(3521)
boosting.grid = expand.grid(n.trees = c(2000, 3000, 4000, 5000),
                           interaction.depth = 1:3,
                           shrinkage = c(0.001, 0.003, 0.005),
                           n.minobsinnode = 1)

set.seed(3521)
cl = makePSOCKcluster(no_cores)
registerDoParallel(cl)
boosting = train(x, y,
                method = "gbm",
                tuneGrid = boosting.grid,
                trControl = ctrl,
                verbose = FALSE)

stopCluster(cl)
registerDoSEQ()

# Plot for tuning parameter selection:
ggplot(boosting, highlight = TRUE) + ggtitle("Figure 11. Boosting CV RMSE Plot")
```

Figure 11. Boosting CV RMSE Plot



```
# Choose best tuning parameter value:
boosting$bestTune
##      n.trees interaction.depth shrinkage n.minobsinnode
```

```
## 23      4000              3      0.003              1

# Report the CV error:
boosting$results[26, ]$RMSE
## [1] 21.49009
# Report test error:
boosting_pred = predict(boosting, newdata = x2) ## Test dataset
boosting_mse = sqrt(mean((boosting_pred - y2)^2))
boosting_mse
## [1] 25.29429
```

[Describe the outputs above]

## Secondary Analysis

### Logistic Model:

```
set.seed(3521)
glm = train(recovery_time ~ .,
            data = training_sec,
            method = "glm",
            metric = "ROC",
            trControl = ctrl1)
glm$finalModel
##
## Call:  NULL
##
## Coefficients:
##      (Intercept)          age      gender1      race2      race3
##      7.736e+01     -3.527e-02     2.843e-01     3.878e-01     1.411e-01
##      race4      smoking1      smoking2      height      weight
##      7.673e-03     -4.077e-01     -4.431e-01     -4.425e-01     4.718e-01
##      bmi hypertension1      diabetes1      SBP      LDL
##      -1.453e+00     -1.643e-01     4.339e-02     -6.311e-05     4.684e-03
##      vaccine1      severity1      studyB      studyC
##      5.229e-01     -6.944e-01     1.335e+00     1.320e-01
##
## Degrees of Freedom: 2512 Total (i.e. Null);  2494 Residual
## Null Deviance:      3039
## Residual Deviance: 2725  AIC: 2763

# Report the CV ROC:
glm$results$ROC
## [1] 0.7095743
```

[Describe the outputs above]

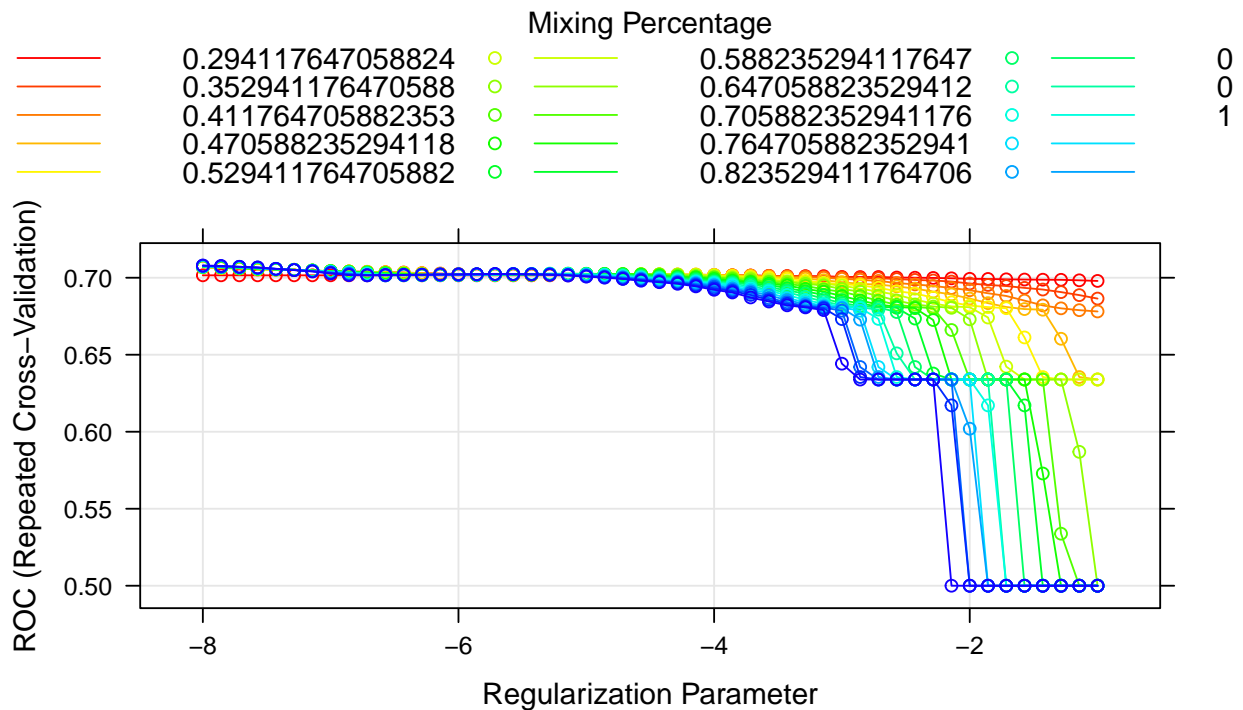
## Penalized Logistic Model:

```
set.seed(3521)
glmGrid = expand.grid(alpha = seq(0, 1, length = 18),
                      lambda = exp(seq(-8, -1, length = 50)))

glm = train(recovery_time ~ .,
            data = training_sec,
            method = "glmnet",
            tuneGrid = glmGrid,
            metric = "ROC",
            trControl = ctrl1)

# Plot for tuning parameter selection:
myCol = rainbow(25)
myPar = list(superpose.symbol = list(col = myCol),
             superpose.line = list(col = myCol))
plot(glm, par.settings = myPar, xTran = function(x)log(x), highlight = T, main = "Figure I. Penalized Logistic Model CV ROC Plot")
```

**Figure I. Penalized Logistic Model CV ROC Plot**



```
# Choose best tuning parameter value:
glm$bestTune
##      alpha      lambda
## 851      1 0.0003354626

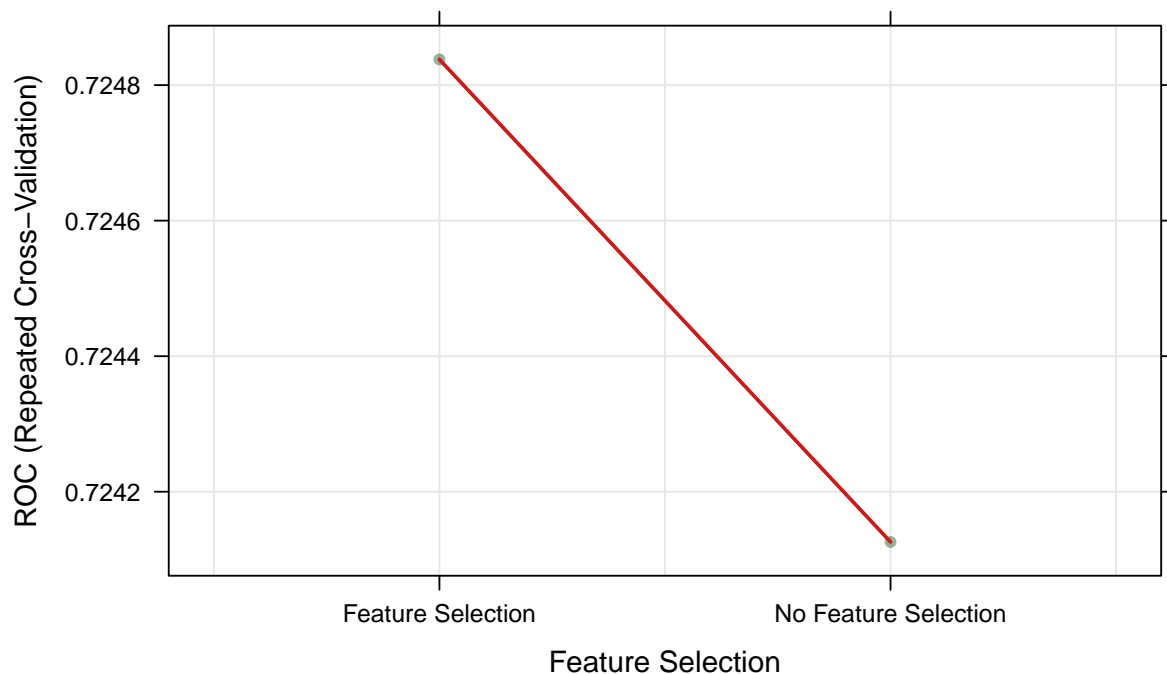
# Report the CV ROC:
glm$results[851,]$ROC
## [1] 0.7079283
```

[Describe the outputs above]

GAM model for binary response:

```
set.seed(3521)
cl = makePSOCKcluster(no_cores)
registerDoParallel(cl)
gam_2 = train(recovery_time ~ .,
              data = training_sec,
              method = "gam",
              metric = "ROC",
              trControl = ctrl1)
stopCluster(cl)
registerDoSEQ()
# Plot for tuning parameter selection:
plot(gam_2, main = "Figure II. GAM Model CV ROC Plot")
```

**Figure II. GAM Model CV ROC Plot**



```
# Report the final model:
gam_2$bestTune
## select method
## 2 TRUE GCV.Cp
coef(gam_2$finalModel)
## (Intercept) gender1 race3 race4 smoking1
## -2.094415e+00 2.880745e-01 9.310134e-02 6.559474e-03 -4.371700e-01
## smoking2 hypertension1 diabetes1 vaccine1 severity1
## -4.784380e-01 -1.857253e-01 1.970598e-02 5.356926e-01 -7.120308e-01
```

```
##      studyB      studyC      s(age).1      s(age).2      s(age).3
## 1.373336e+00 1.675730e-01 -3.547967e-04 1.690790e-03 -1.838724e-03
##      s(age).4      s(age).5      s(age).6      s(age).7      s(age).8
## -5.165523e-03 1.730323e-03 -4.076744e-03 2.181238e-03 -2.341176e-02
##      s(age).9      s(SBP).1      s(SBP).2      s(SBP).3      s(SBP).4
## -1.313838e-01 5.677327e-06 3.678094e-07 -1.373775e-07 -4.587642e-07
##      s(SBP).5      s(SBP).6      s(SBP).7      s(SBP).8      s(SBP).9
## -1.936734e-07 3.521817e-07 3.268165e-07 4.034795e-07 1.063455e-06
##      s(LDL).1      s(LDL).2      s(LDL).3      s(LDL).4      s(LDL).5
## 9.888402e-06 9.235410e-07 2.118225e-06 -5.184814e-07 -6.001575e-07
##      s(LDL).6      s(LDL).7      s(LDL).8      s(LDL).9      s(bmi).1
## -1.067971e-07 3.793180e-07 -2.515578e-06 6.736418e-02 2.008212e-01
##      s(bmi).2      s(bmi).3      s(bmi).4      s(bmi).5      s(bmi).6
## 6.053645e-02 4.628156e-02 -3.554393e-01 9.872649e-02 2.925849e-01
##      s(bmi).7      s(bmi).8      s(bmi).9      s(height).1      s(height).2
## 6.796104e-02 1.848588e+00 -2.913658e-01 -3.745810e-06 -1.490001e-08
##      s(height).3      s(height).4      s(height).5      s(height).6      s(height).7
## 7.316762e-07 -6.473545e-07 4.594748e-07 5.722461e-07 3.972392e-07
##      s(height).8      s(height).9      s(weight).1      s(weight).2      s(weight).3
## -2.537800e-06 -1.011582e-06 1.537831e-06 1.359224e-07 3.751313e-07
##      s(weight).4      s(weight).5      s(weight).6      s(weight).7      s(weight).8
## 3.344343e-07 -2.836629e-07 -2.968417e-07 2.050625e-07 1.225004e-06
##      s(weight).9
## -4.006080e-06

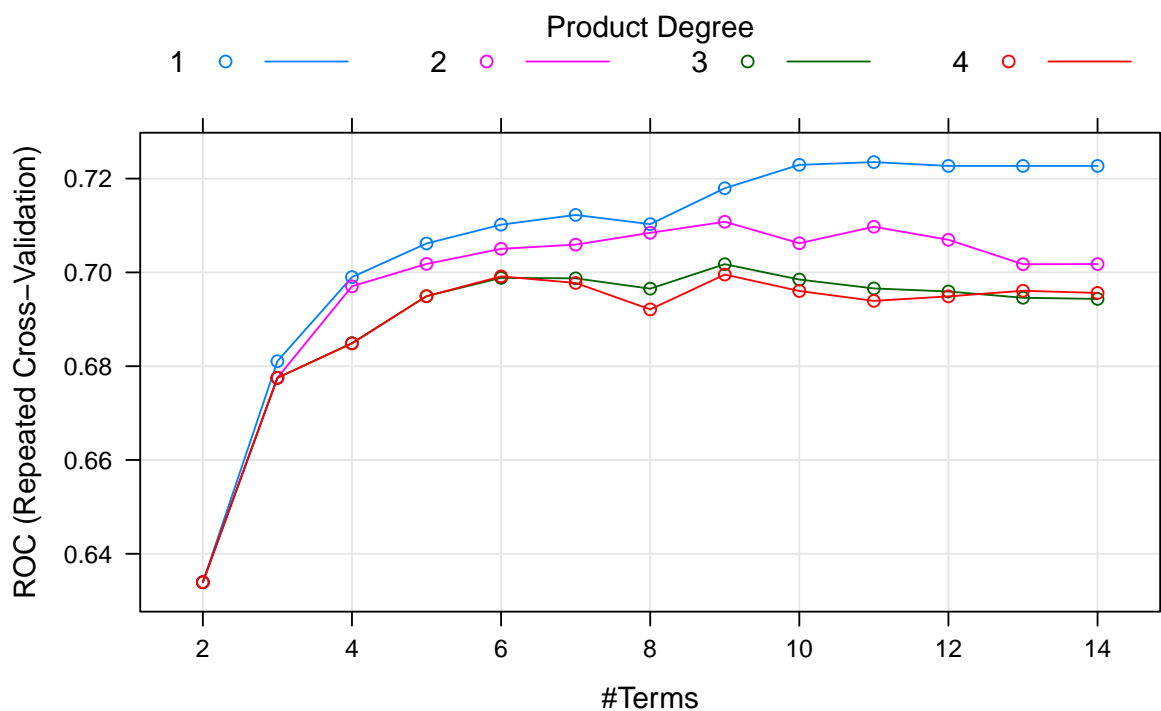
# Report the CV ROC:
gam_2$results[2, ]$ROC
## [1] 0.7248379
```

[Describe the outputs above]

MARS model for binary response:

```
set.seed(3521)
cl = makePSOCKcluster(no_cores)
registerDoParallel(cl)
mars_2 = train(recovery_time ~ .,
               data = training_sec,
               method = "earth",
               tuneGrid = expand.grid(degree = 1:4, nprune = 2:14),
               metric = "ROC",
               trControl = ctrl1)
stopCluster(cl)
registerDoSEQ()
# Plot for tuning parameter selection:
plot(mars_2, main = "Figure III. MARS Model CV ROC Plot", highlight = T)
```

**Figure III. MARS Model CV ROC Plot**



```
# Report the final model:
mars_2$bestTune
##      nprune degree
## 10      11      1
coef(mars_2$finalModel)
## (Intercept)      studyB h(bmi-27.8) h(27.8-bmi)      vaccine1      severity1
## -1.5347506    1.2977373 -0.4282987 -0.1841055    0.5316052   -0.7196799
## h(61-age)      smoking1      gender1      smoking2
##  0.0590261   -0.4376436    0.2849957   -0.4923424

# Report the CV ROC:
mars_2$results[37, ]$ROC
## [1] 0.7235168
```

[Describe the outputs above]

**Linear discriminant analysis (LDA):**

```
set.seed(3521)
lda = train(recovery_time ~ .,
            data = training_sec,
            method = "lda",
            metric = "ROC",
            trControl = ctrl1)
# Report the final model:
coef(lda$finalModel)
```



```
##                               LD1
## age                -4.414766e-02
## gender1             3.483168e-01
## race2               5.127539e-01
## race3               2.087032e-01
## race4               1.652280e-02
## smoking1            -5.093387e-01
## smoking2            -5.565945e-01
## height              -4.842308e-01
## weight              5.127571e-01
## bmi                 -1.587689e+00
## hypertension1      -2.035363e-01
## diabetes1           5.532383e-02
## SBP                 -7.703822e-06
## LDL                 6.184843e-03
## vaccine1            6.694769e-01
## severity1           -7.851869e-01
## studyB              1.615000e+00
## studyC              1.104844e-01

# Report the CV ROC:
lda$results$ROC
## [1] 0.7100636
```

[Describe the outputs above]

### Quadratic discriminant analysis (QDA):

```
set.seed(3521)
qda = train(recovery_time ~ .,
            data = training_sec,
            method = "qda",
            metric = "ROC",
            trControl = ctrl1)

# Report the final model:
qda$finalModel
## Call:
## qda(x, grouping = y)
##
## Prior probabilities of groups:
##      great      less
## 0.707123 0.292877
##
## Group means:
##      age gender1      race2      race3      race4 smoking1 smoking2
## great 60.29994 0.4687676 0.04614519 0.1963984 0.09848059 0.3168261 0.10298255
## less  59.69429 0.5339674 0.05706522 0.2146739 0.09375000 0.2500000 0.08016304
##      height weight      bmi hypertension1 diabetes1      SBP      LDL
## great 169.7715 80.20096 27.89994      0.5098481 0.1508160 130.5093 109.5937
## less  170.3648 79.04755 27.26454      0.4538043 0.1535326 129.6087 110.7527
##      vaccine1 severity1      studyB      studyC
## great 0.5751266 0.10635903 0.5160383 0.2341024
```

```
## less 0.6942935 0.05842391 0.7839674 0.1127717
```

```
# Report the CV ROC:
```

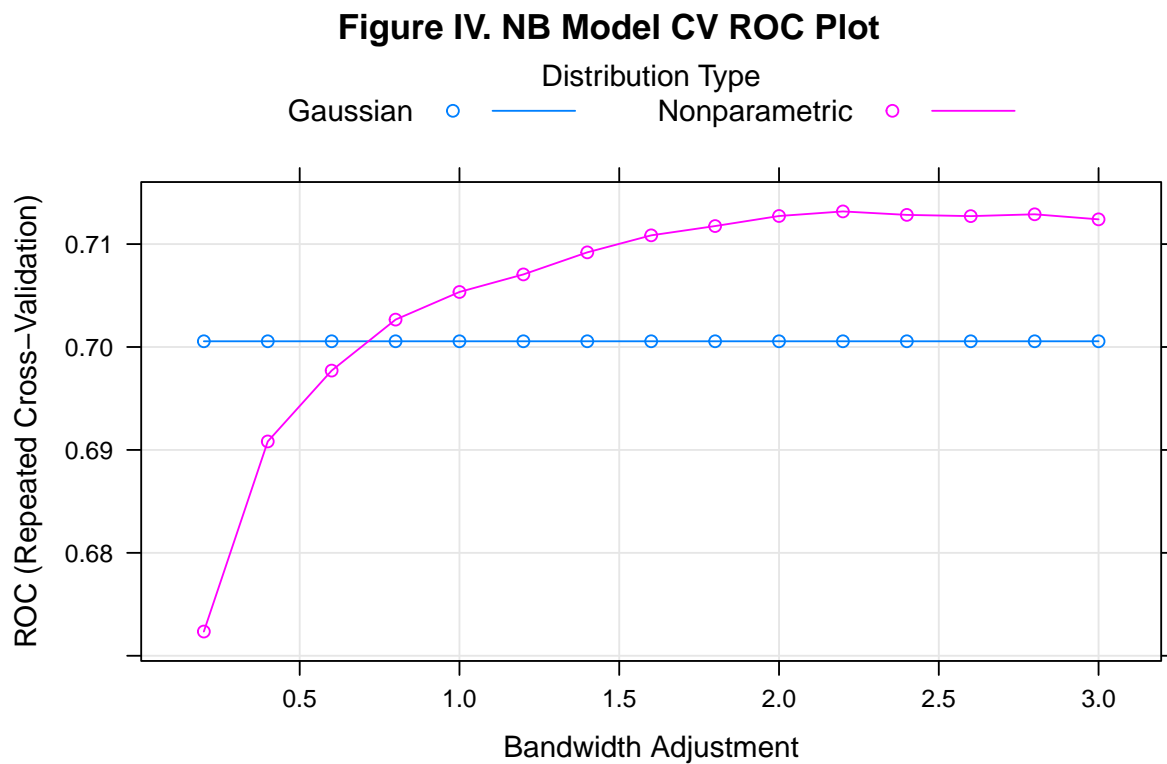
```
qda$results$ROC
```

```
## [1] 0.6965235
```

[Describe the outputs above]

Naïve Bayes:

```
set.seed(3521)
nbGrid = expand.grid(usekernel = c(FALSE, TRUE),
                    fL = 1,
                    adjust = seq(.2, 3, by = .2))
nb = train(recovery_time ~ .,
          data = training_sec,
          method = "nb",
          tuneGrid = nbGrid,
          metric = "ROC",
          trControl = ctrl1)
# Plot for tuning parameter selection:
plot(nb, main = "Figure IV. NB Model CV ROC Plot", highlight = T)
```



```
# Report the final model:
nb$bestTune
##      fL usekernel adjust
## 26  1      TRUE    2.2

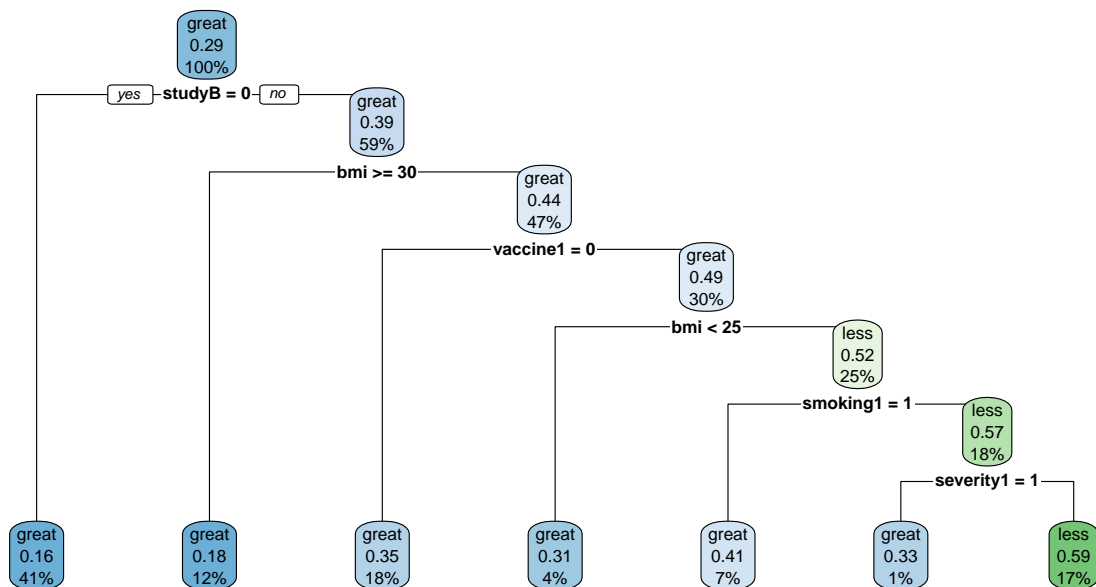
# Report the CV ROC:
nb$results[26,]$ROC
## [1] 0.7131731
```

[Describe the outputs above]

Classification tree:

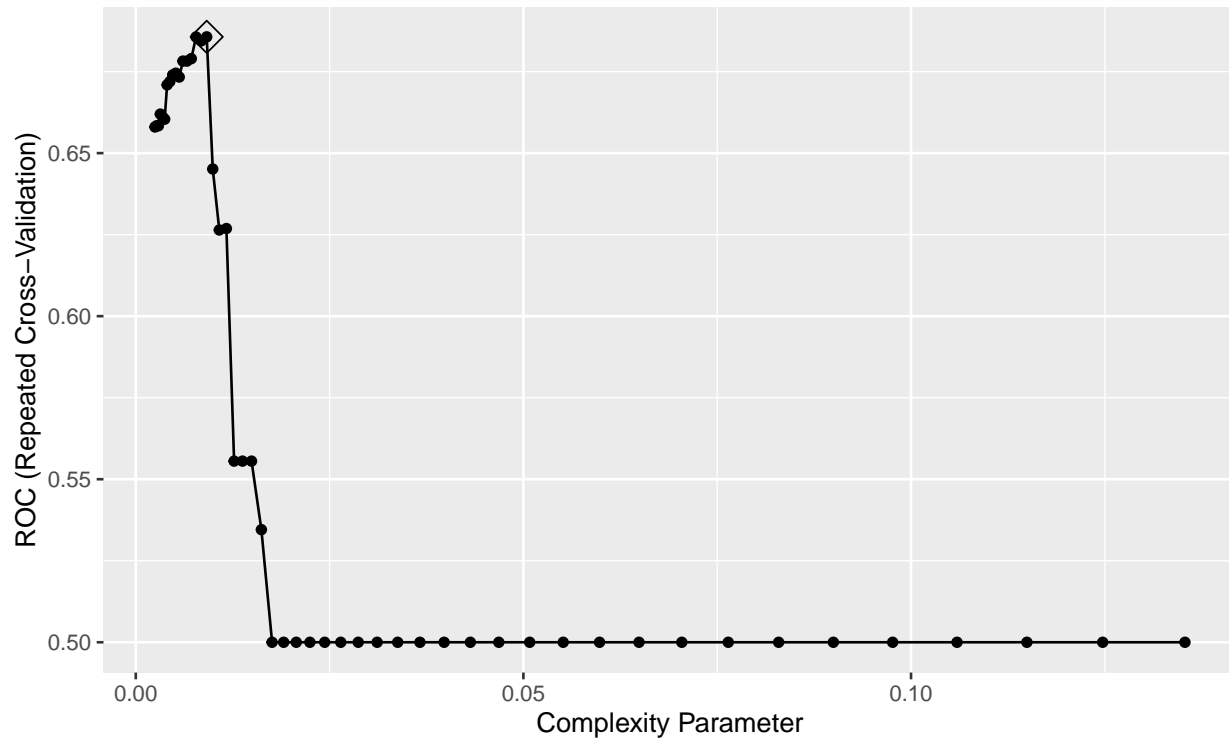
```
set.seed(3521)
rpart = train(recovery_time ~ . ,
              data = training_sec,
              method = "rpart",
              tuneGrid = data.frame(cp = exp(seq(-6, -2, length = 50))),
              trControl = ctrl1,
              metric = "ROC")
# Plot for classification tree:
rpart.plot(rpart$finalModel, main = "Figure V(I). Classification Tree")
```

Figure V(I). Classification Tree



```
# Plot for tuning parameter selection:
ggplot(rpart, highlight = TRUE) + ggtitle("Figure V(II). Classification Tree Model CV ROC Plot")
```

Figure V(II). Classification Tree Model CV ROC Plot



```
# Choose best tuning parameter value:
rpart$bestTune
##          cp
## 17 0.009151133

# Report the CV ROC
rpart$results[17, ]$ROC
## [1] 0.6856528
```

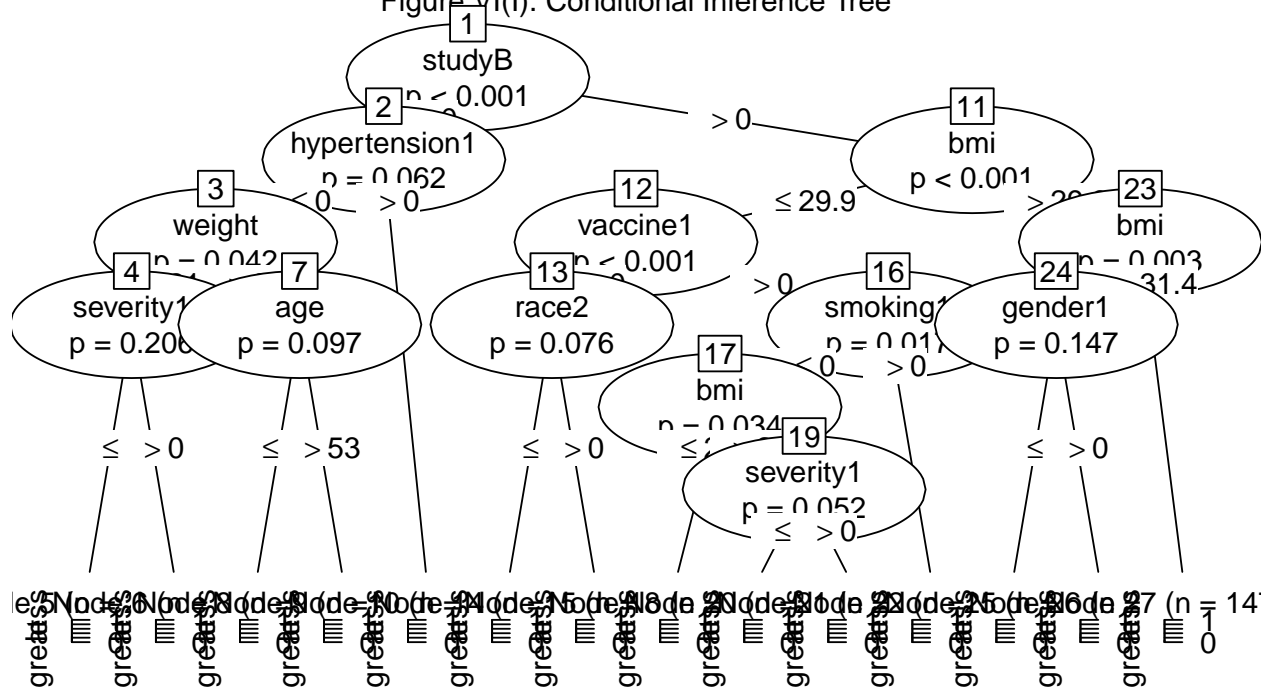
[Describe the outputs above]

### Conditional Inference Tree:

```
set.seed(3521)
ctree = train(recovery_time ~ . ,
              data = training_sec,
              method = "ctree",
              tuneGrid = data.frame(mincriterion = 1 - exp(seq(-2, -1, length = 50))),
              trControl = ctrl1,
              metric = "ROC")

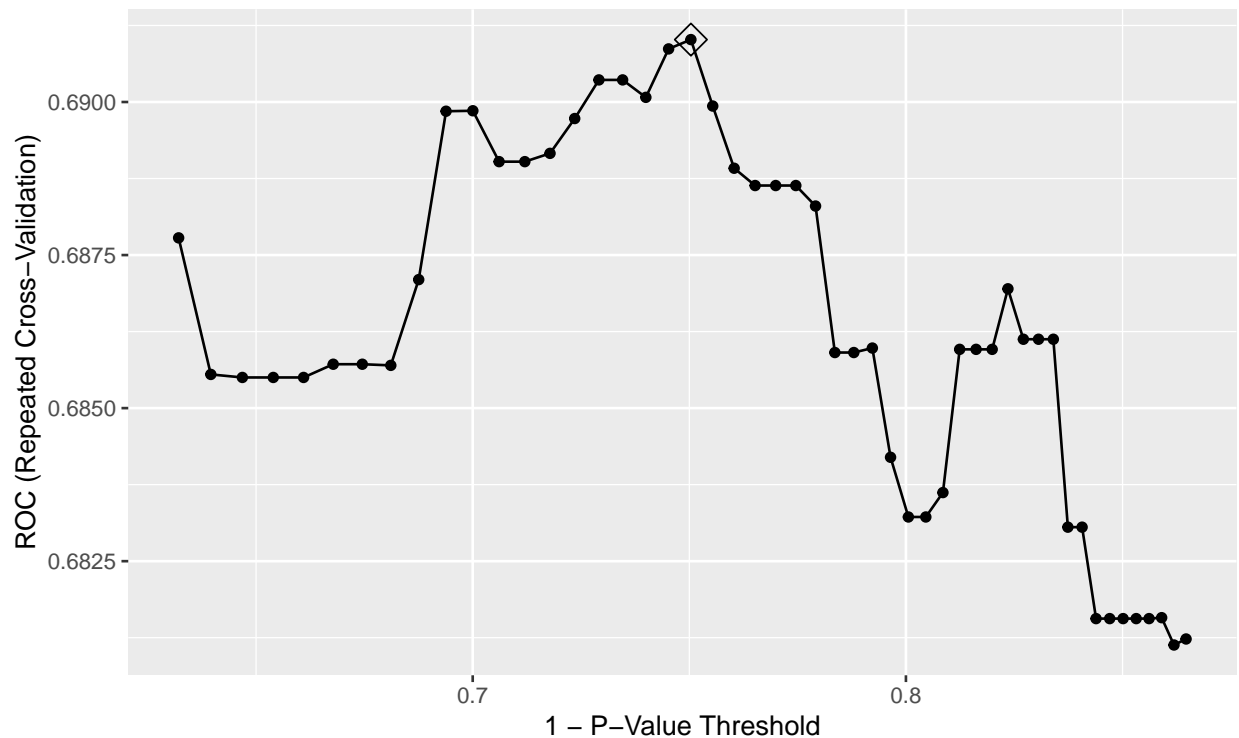
# Plot for CIT:
plot(ctree$finalModel, main = ("Figure VI(I). Conditional Inference Tree"))
```

1 (1). Conventional interference free



```
# Plot for tuning parameter selection:
ggplot(ctree, highlight = T) + ggtitle("Figure VI(II). CIT Model CV ROC Plot")
```

Figure VI(II). CIT Model CV ROC Plot



```
# Choose best tuning parameter value:
ctree$bestTune
##      mincriterion
## 20      0.7503649

# Report the CV ROC
ctree$results[20, ]$ROC
## [1] 0.6910193
```

[Describe the outputs above]

Random Forests for binary response:

```
rf.grid = expand.grid(mtry = 0:6,
                     splitrule = "gini",
                     min.node.size = seq(from = 2, to = 10, by = 2))

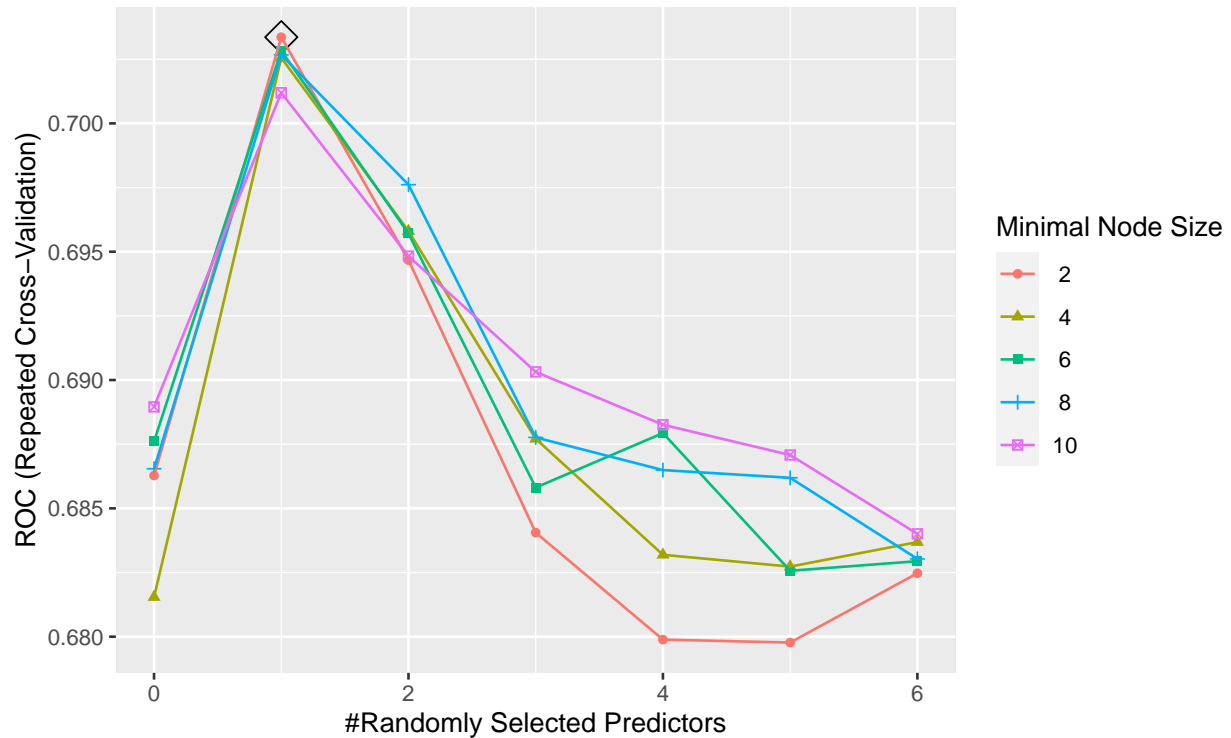
set.seed(3521)
cl = makePSOCKcluster(no_cores)
registerDoParallel(cl)
rf_2 = train(recovery_time ~ . ,
             data = training_sec,
             method = "ranger",
             tuneGrid = rf.grid,
             trControl = ctrl1,
             metric = "ROC")
stopCluster(cl)
```

```
registerDoSEQ()
```

```
# Plot for tuning parameter selection:
```

```
ggplot(rf_2, highlight = TRUE) + ggtitle("Figure VII. Random Forests Model CV ROC Plot")
```

Figure VII. Random Forests Model CV ROC Plot



```
# Choose best tuning parameter value:
```

```
rf_2$bestTune
```

```
## mtry splitrule min.node.size  
## 6 1 gini 2
```

```
# Report the CV ROC
```

```
rf_2$results[6,]$ROC
```

```
## [1] 0.703361
```

[Describe the outputs above]

**Adaboost Model:**

```
gbmA.grid = expand.grid(n.trees = c(1000, 2000, 3000, 4000),  
                        interaction.depth = 1:6,  
                        shrinkage = c(0.0005, 0.001, 0.0015),  
                        n.minobsinnode = 1)
```

```
set.seed(3521)
```

```
c1 = makePSOCKcluster(no_cores)
```

```
registerDoParallel(c1)
```

```

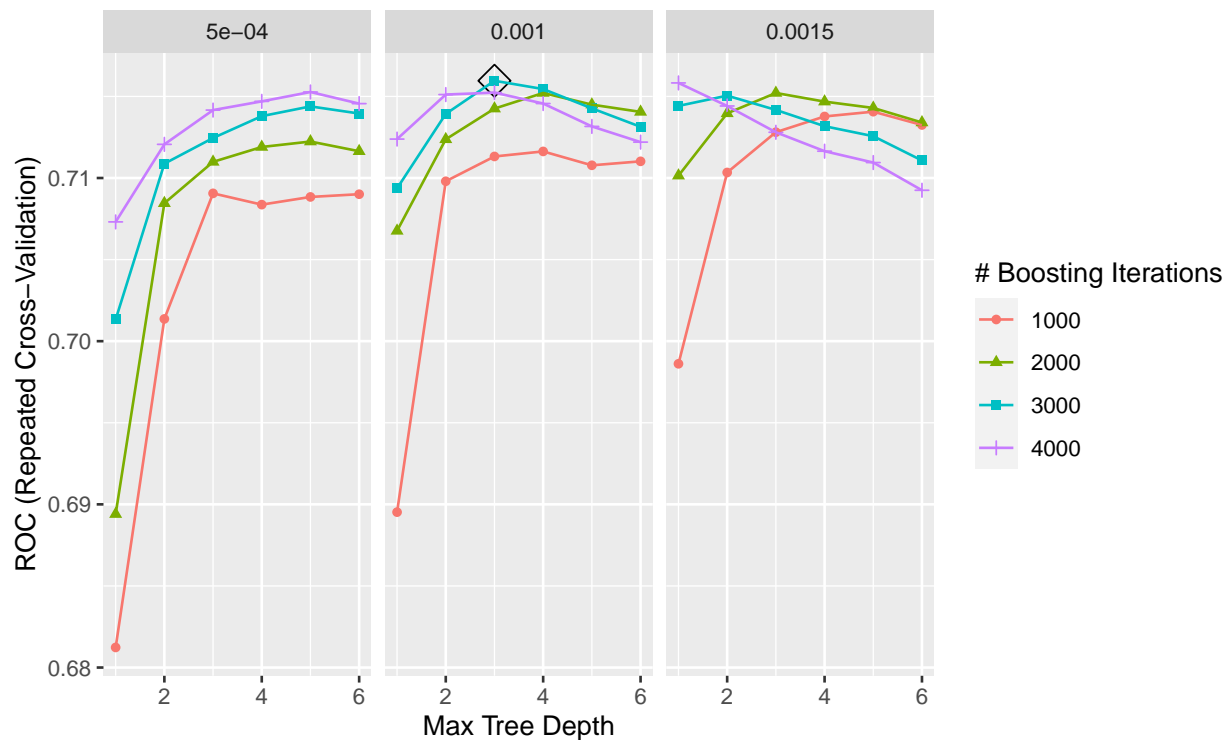
Adaboost = train(recovery_time ~ . ,
                 data = training_sec,
                 tuneGrid = gbmA.grid,
                 trControl = ctrl1,
                 method = "gbm",
                 distribution = "adaboost",
                 metric = "ROC",
                 verbose = FALSE)

stopCluster(cl)
registerDoSEQ()

# Plot for tuning parameter selection:
ggplot(Adaboost, highlight = TRUE) + ggtitle("Figure VIII. Adaboost Model CV ROC Plot")

```

Figure VIII. Adaboost Model CV ROC Plot



```

# Choose best tuning parameter value:
Adaboost$bestTune
##      n.trees interaction.depth shrinkage n.minobsinnode
## 35      3000                3      0.001                1

# Report the CV ROC:
Adaboost$results[44,]$ROC
## [1] 0.7159581

```

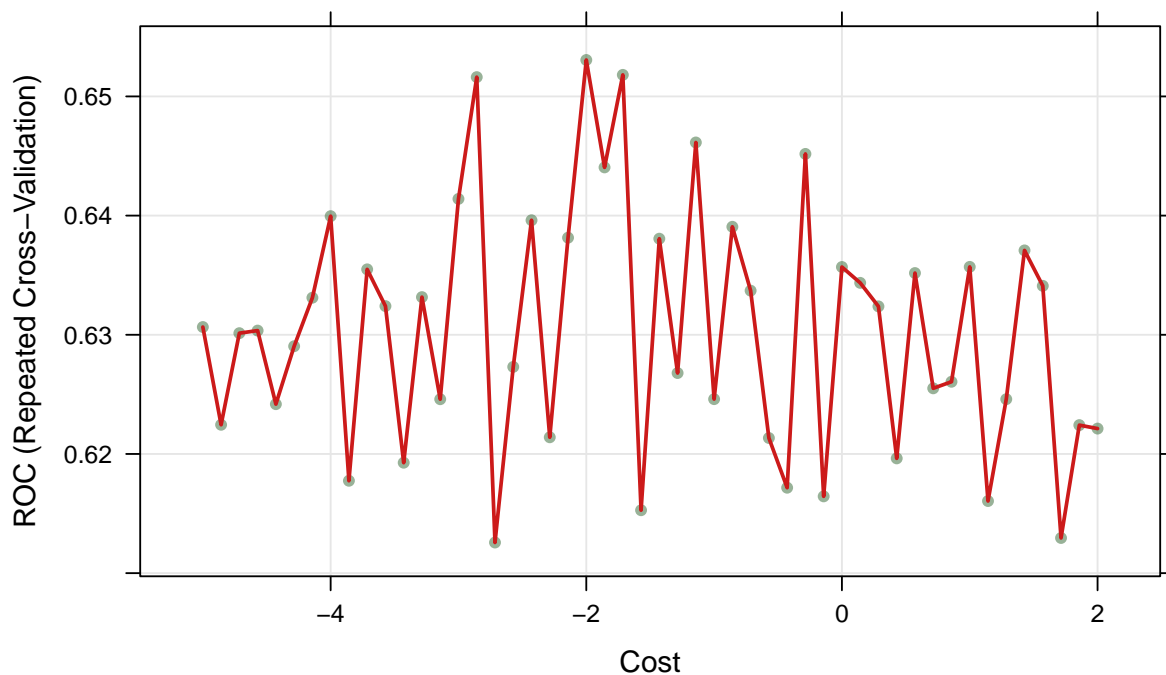
[Describe the outputs above]



## Support Vector Machine (linear kernel):

```
set.seed(3521)
# kernlab
cl = makePSOCKcluster(no_cores)
registerDoParallel(cl)
svml = train(recovery_time ~ . ,
             training_sec,
             method = "svmLinear",
             tuneGrid = data.frame(C = exp(seq(-5, 2, len = 50))),
             trControl = ctrl1)
## maximum number of iterations reached 0.007078654 0.006600204
stopCluster(cl)
registerDoSEQ()
# Plot for tuning parameter selection:
plot(svml, highlight = TRUE, xTrans = log, main = ("Figure IX. SVM(linear) Model CV ROC Plot"))
```

Figure IX. SVM(linear) Model CV ROC Plot



```
# Choose best tuning parameter value:
svml$bestTune
##          C
## 22 0.1353353

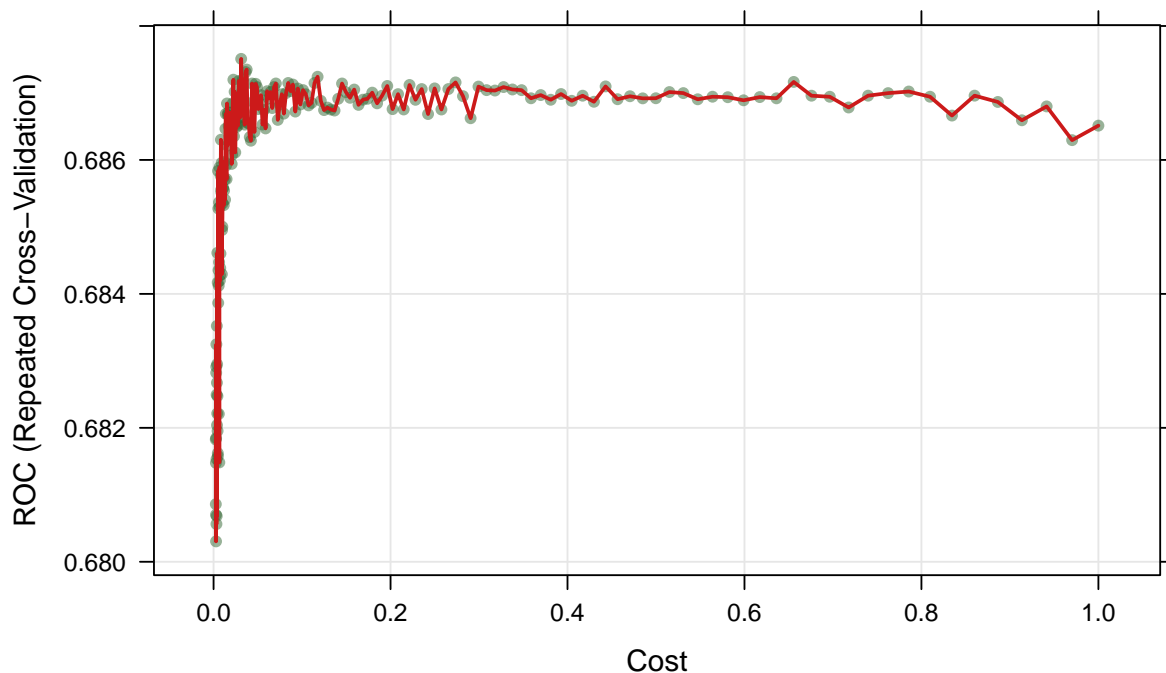
# Report the CV ROC:
svml$results[22, ]$ROC
## [1] 0.6530562
```

[Describe the outputs above]

Support Vector Machine (radical kernel):

```
set.seed(3521)
cl = makePSOCKcluster(no_cores)
registerDoParallel(cl)
svmr = train(recovery_time ~ . ,
             training_sec,
             method = "svmRadialCost",
             tuneGrid = data.frame(C = exp(seq(-6, 0, len = 200))),
             trControl = ctrl1)
stopCluster(cl)
registerDoSEQ()
# Plot for tuning parameter selection:
myCol = rainbow(25)
myPar = list(superpose.symbol = list(col = myCol),
             superpose.line = list(col = myCol))
plot(svmr, highlight = TRUE, par.settings = myPar, main = "Figure X. SVM(radical) Model CV ROC Plot")
```

Figure X. SVM(radical) Model CV ROC Plot



```
# Choose best tuning parameter value:
svmr$bestTune
##           C
## 85 0.03120002
# Report the ROC
svmr$results[85,]$ROC
## [1] 0.6875091
```

[Describe the outputs above]