

qz2266_primary_regression tree_random forest

Qing Zhou

2023-05-04

Data preparation

Data partition

Next, we split the dataset into two parts: training data (80%) and test data (20%).

```
set.seed(2266)
trainRows <- createDataPartition(y = covid$recovery_time, p = 0.8, list = FALSE)

# Training data
covid_train = covid[trainRows, ]
x_train = model.matrix(recovery_time~.,covid)[trainRows, -1]
y_train = covid$recovery_time[trainRows]

# Test data
covid_test = covid[-trainRows, ]
x_test = model.matrix(recovery_time~.,covid)[-trainRows, -1]
y_test = covid$recovery_time[-trainRows]

# create cross-validation objects
ctrl1 <- trainControl(method = "cv")
```

(a) Regression tree

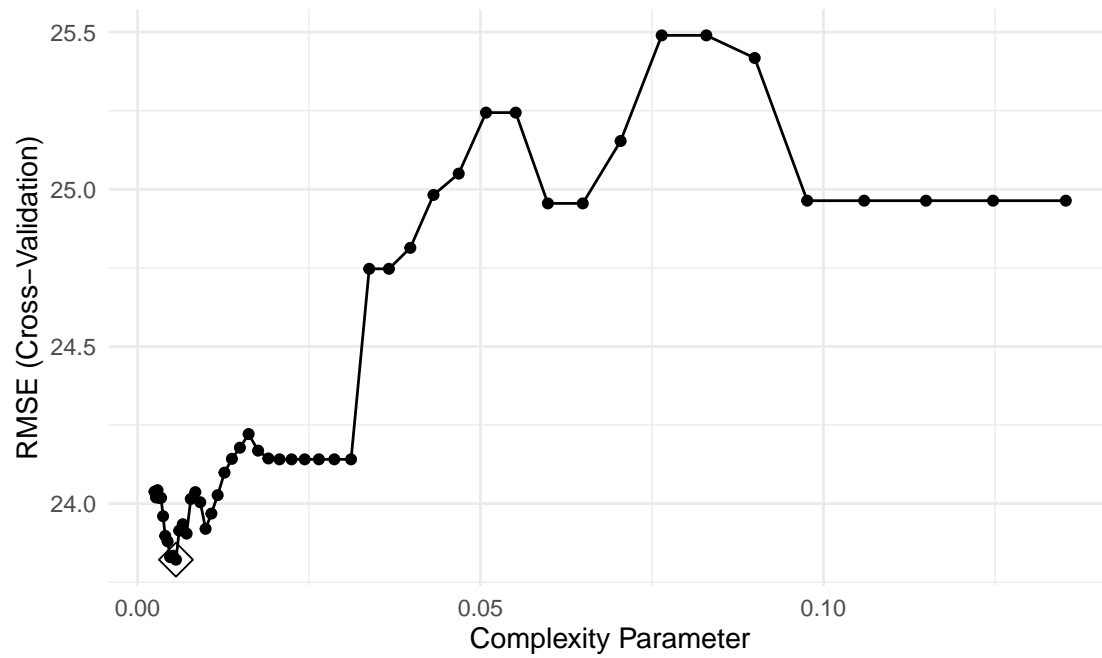
Build a regression tree on the training data to predict the response. Create a plot of the tree.

```
set.seed(2266)

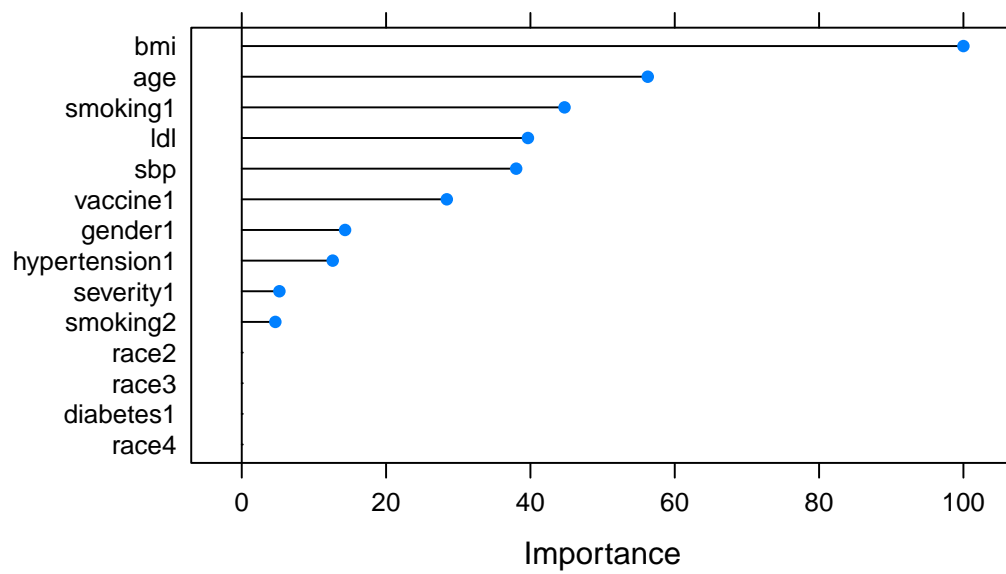
# build a regression tree on the training data
rpart.fit <- train(recovery_time ~ . ,
                  data = covid_train,
                  method = "rpart",
                  tuneGrid = data.frame(cp = exp(seq(-6,-2, length = 50))),
                  trControl = ctrl1)
rpart.fit$bestTune

##           cp
## 11 0.00560737
```

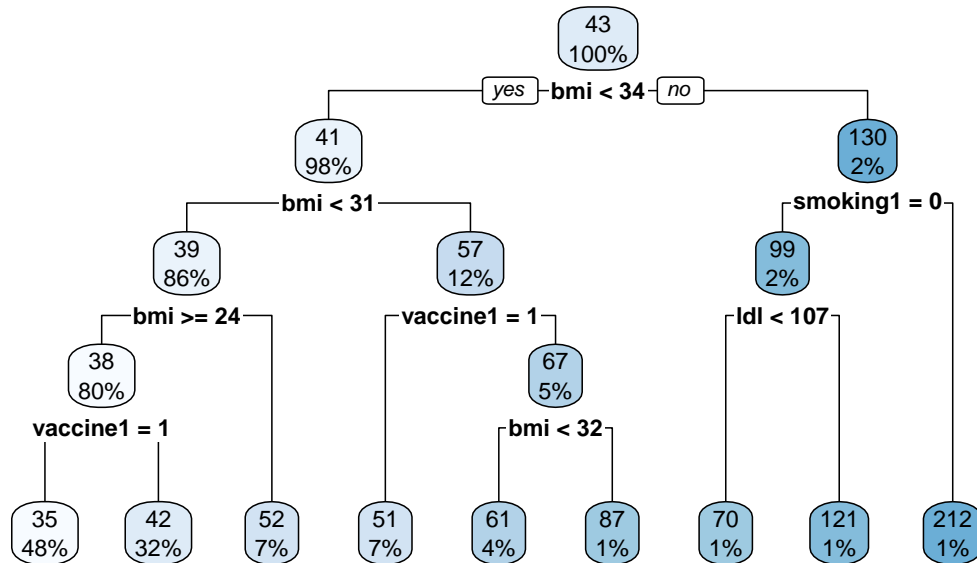
```
# plot of the complexity parameter
ggplot(rpart.fit, highlight = TRUE)
```



```
# importance
plot(varImp(rpart.fit, scale = TRUE))
```



```
# create a plot of the tree
rpart.plot(rpart.fit$finalModel)
```



- The root node is `bmi` over or under 34.
- The optimal `cp` is 0.002918356.
- The pruned tree based on the optimal `cp` value is plotted as above. It's quite complicated with 15 terminal nodes and 14 splits.
- The model indicated that the variables `bmi` has the highest predictive power, followed by `vaccine1`, `age`, and `sbp`.

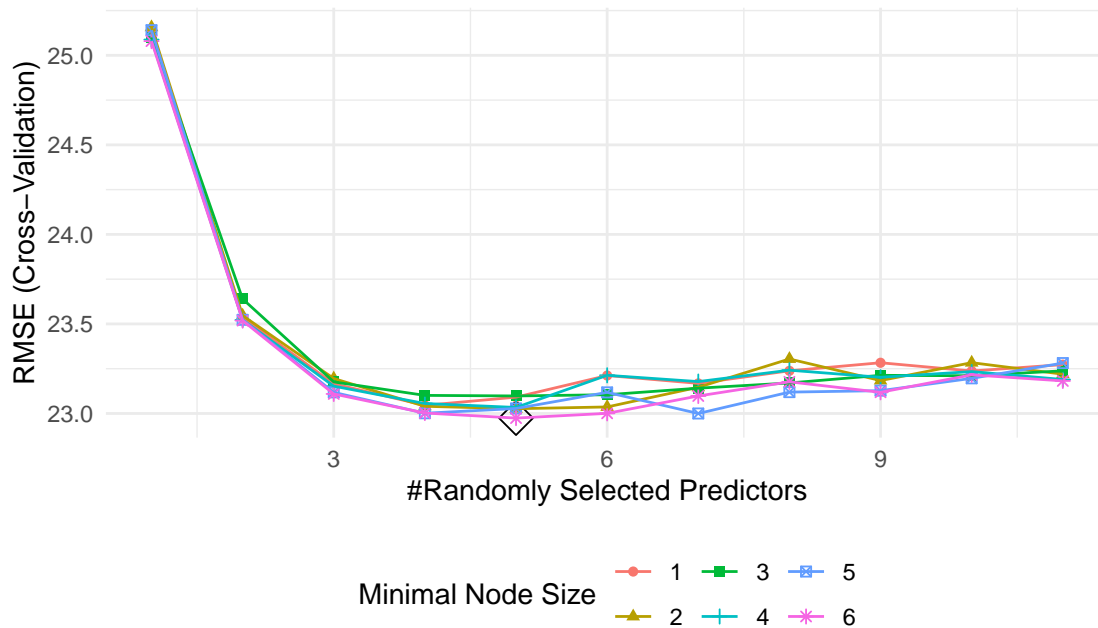
(b) Random forest

Perform random forest on the training data. Report the variable importance and the test error.

```
rf.grid <- expand.grid(mtry = 1:11,
                      splitrule = "variance",
                      min.node.size = 1:6)

set.seed(2266)
# train a random forest model on the training data
rf.fit <- train(recovery_time ~ .,
               data = covid_train,
               method = "ranger",
               tuneGrid = rf.grid,
               trControl = ctrl1)

ggplot(rf.fit, highlight = TRUE)
```

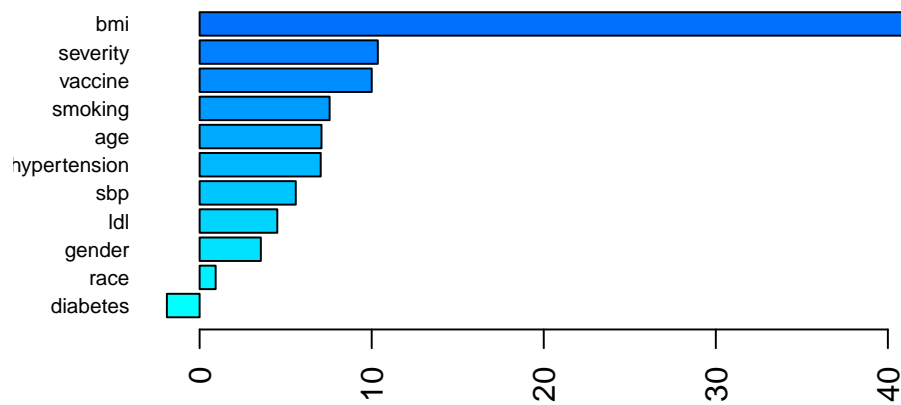


```
rf.fit$bestTune
```

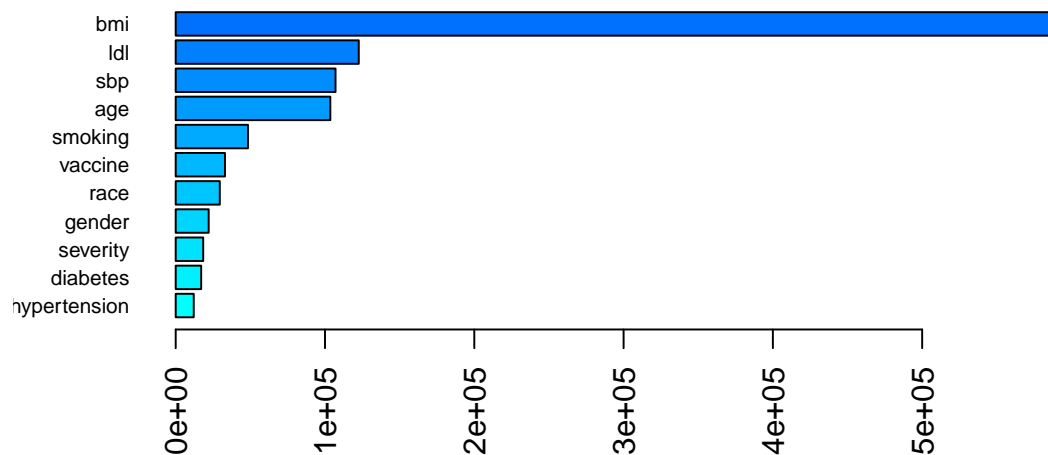
```
##      mtry splitrule min.node.size
## 30      5  variance              6
```

- Using ranger method, we perform Random Forest algorithm with minimum node size 6 and 4 selected predictors.
- It is a common practice to investigate the variables with the greatest predictive power once a random forest model has been trained. These important variables play a crucial role in determining the outcome, and their values can have a substantial impact on the results. Conversely, variables with low importance may be omitted from the model, leading to a streamlined model that is more efficient in terms of fitting and prediction accuracy.

```
# variable importance using permutation methods
set.seed(2266)
rf.perm = ranger(recovery_time ~ .,
                  data = covid_train,
                  mtry = rf.fit$bestTune[[1]],
                  splitrule = "variance",
                  min.node.size = rf.fit$bestTune[[3]],
                  importance = "permutation",
                  scale.permutation.importance = TRUE)
# report variable importance
barplot(sort(ranger::importance(rf.perm), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("cyan", "blue"))(19))
```



```
# variable importance using impurity methods
set.seed(2266)
rf.impu <- ranger(recovery_time ~ .,
                  data = covid_train,
                  mtry = rf.fit$bestTune[[1]],
                  splitrule = "variance",
                  min.node.size = rf.fit$bestTune[[3]],
                  importance = "impurity")
# report variable importance
barplot(sort(ranger::importance(rf.impu), decreasing = FALSE),
        las = 2, horiz = TRUE, cex.names = 0.7,
        col = colorRampPalette(colors = c("cyan", "blue"))(19))
```



- Calculate and graph variable importance using permutation and impurity metrics.
- The model using impurity method indicated that the variables **bmi** has the highest predictive power, followed by **ldl**, **sbp**, and **age**. Their values were the most significant in determining **recovery_time**. This suggests that these variables play a crucial role in influencing the Covid recovery time. Moreover, vaccinated status, race and smoking or not also contributes to the outcome.
- Similarly, The model using permutation method indicated that the variables **bmi** has the highest predictive power, followed by **vaccine**, **gender**, **severity**, **hypertension** and **sbp**.

```
# test error
pred.rf <- predict(rf.fit, newdata = covid_test)
RMSE(pred.rf, covid_test$recovery_time)
```

```
## [1] 25.71229
```

- The test error of the model is 25.54926

(c) Boosting

Perform boosting on the training data. Report the variable importance and the test error.

```
# train model using gbm with grid of tuning parameters
bst.grid = expand.grid(n.trees = c(2000,3000,4000,5000),
                      interaction.depth = 1:3,
                      shrinkage = c(0.005,0.01),
                      n.minobsinnode = c(1))

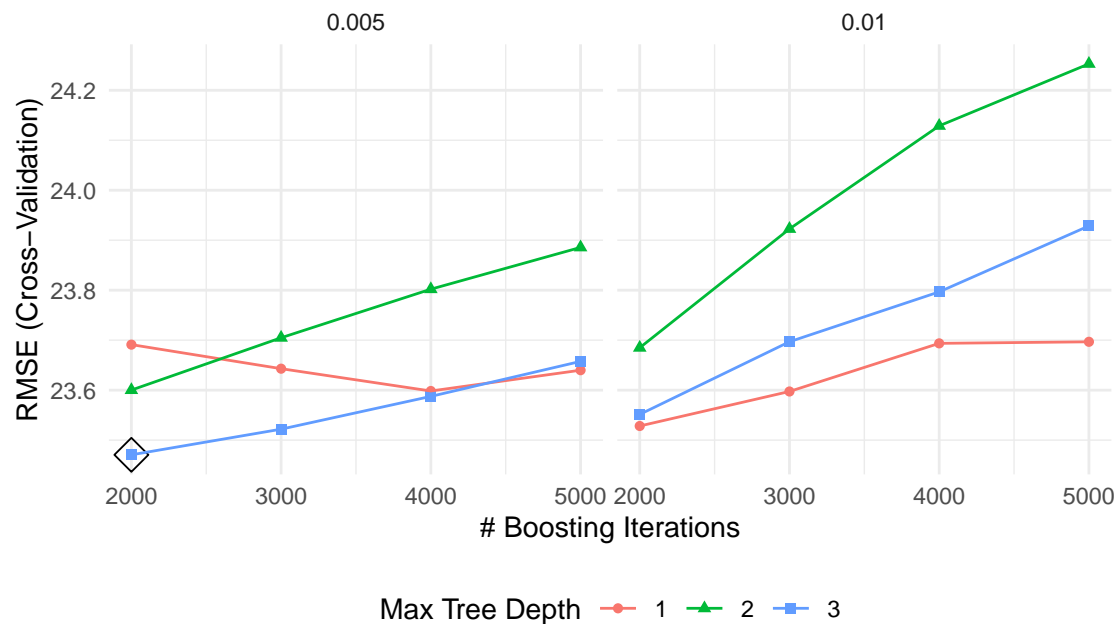
set.seed(2266)
bst.fit <- train(recovery_time ~.,
```

```

data = covid_train,
method = "gbm",
tuneGrid = bst.grid,
trControl = ctrl1,
verbose = FALSE)

ggplot(bst.fit, highlight = TRUE)

```



```
bst.fit$bestTune
```

```

##   n.trees interaction.depth shrinkage n.minobsinnode
## 9    2000                3    0.005                1

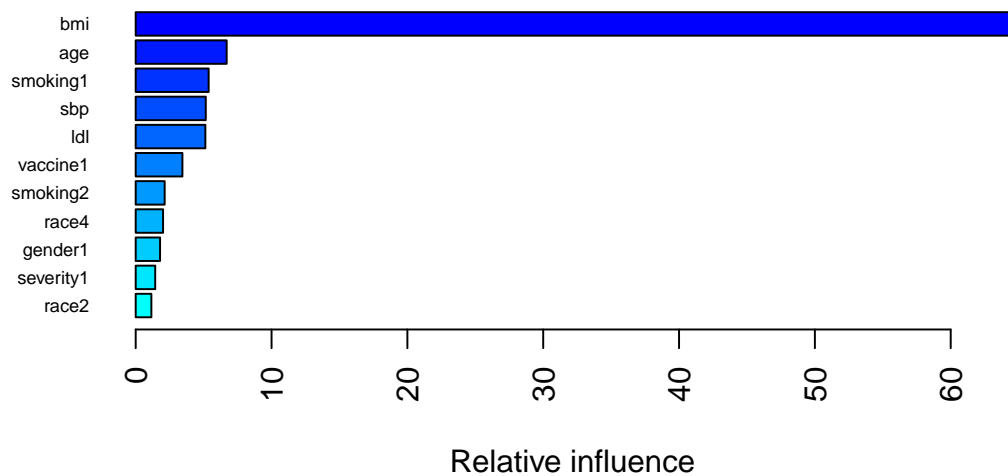
```

We use the gradient boosting method implemented with gbm in caret package.

```

# variable importance
summary(bst.fit$finalModel, las = 2, cBars = 11, cex.names = 0.6)

```



```
##           var      rel.inf
## bmi          bmi 64.95300060
## age          age  6.69718208
## smoking1     smoking1 5.37107139
## sbp          sbp  5.15898089
## ldl          ldl  5.12714761
## vaccine1     vaccine1 3.43987525
## smoking2     smoking2 2.13179324
## race4        race4  2.01577972
## gender1      gender1 1.79422224
## severity1    severity1 1.43776415
## race2        race2  1.15442420
## race3        race3  0.35802293
## hypertension1 hypertension1 0.31363805
## diabetes1    diabetes1 0.04709765
```

```
# test error
pred.bst <- predict(bst.fit, newdata = covid_test)
RMSE(pred.bst, covid_test$recovery_time)
```

```
## [1] 26.51462
```

- The most important variables for gradient boosting are still bmi. Other important variables include ldl, sbp and age.
- The test error for boosting is 25.13826, smaller than the test error for random forest.
-