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")
```

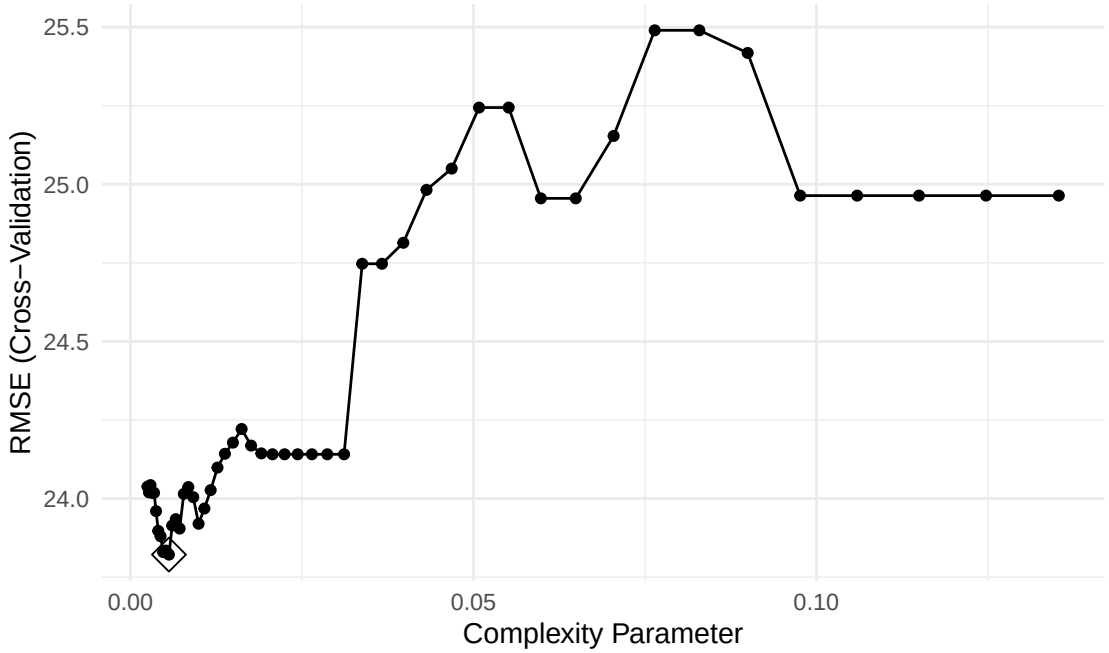
(a) Regression tree

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

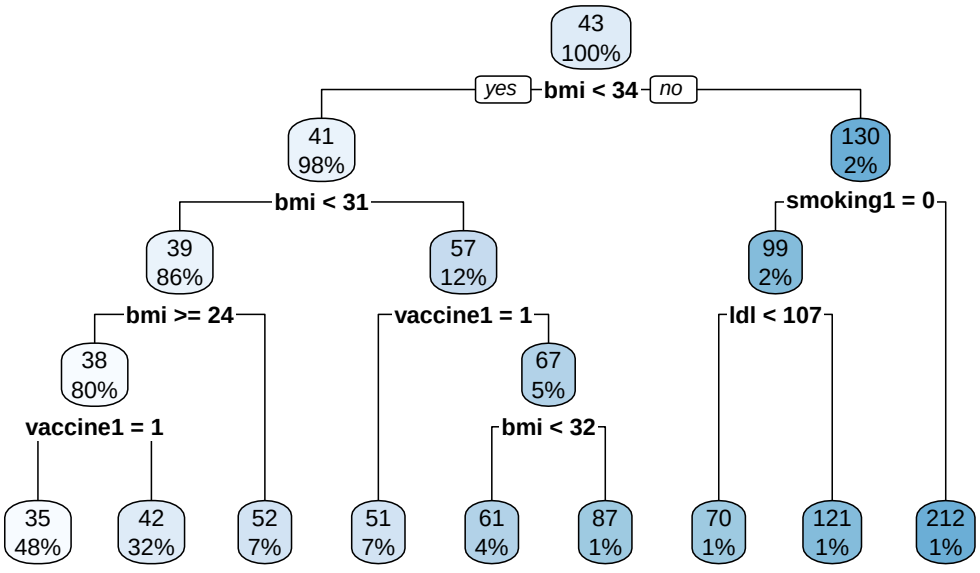
```
set.seed(2266)

# 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)
```




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



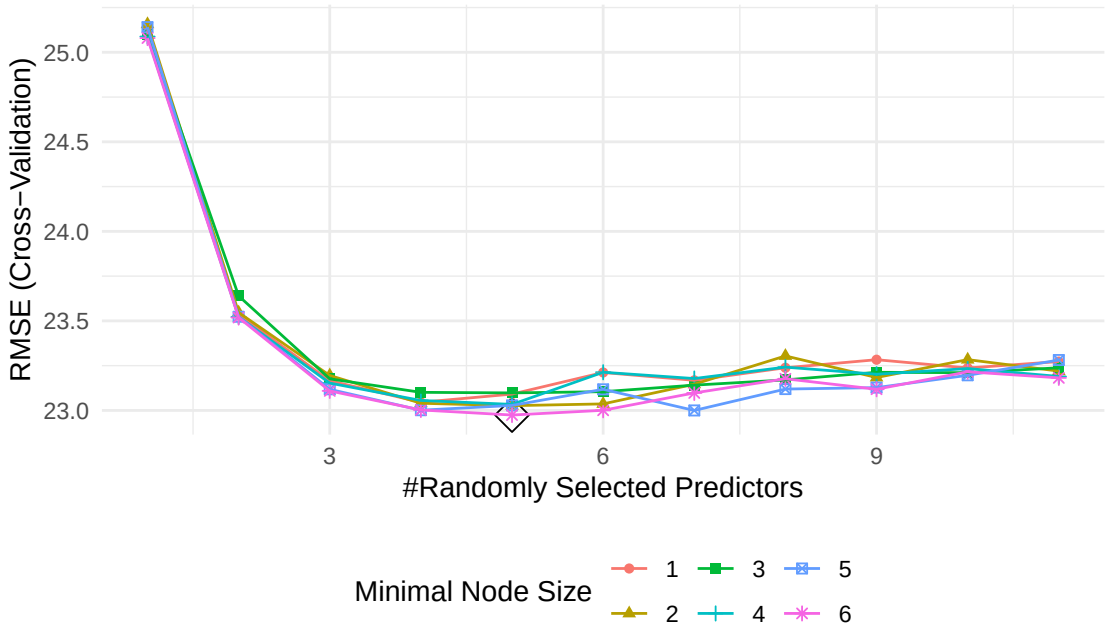
- The root node is `bmi` over or under 34.
- The optimal cp is 0.00560737.

- The pruned tree based on the optimal cp value is plotted as above. It's quite complicated with 9 terminal nodes and 8 splits.

(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)
```



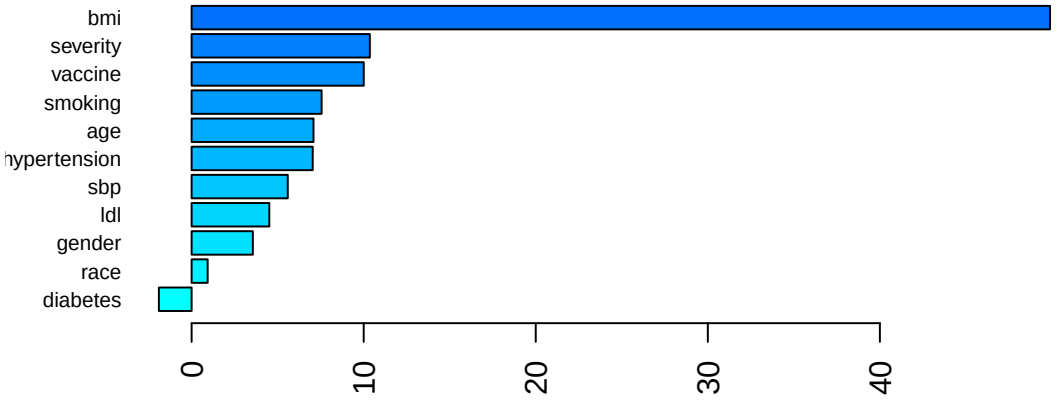

```
rf.fit$bestTune
```

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

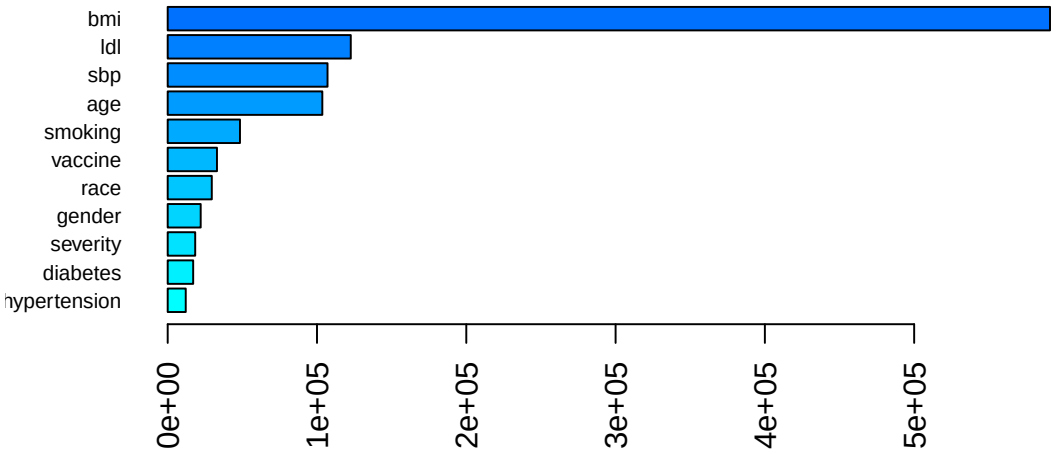
- Using ranger method, we perform Random Forest algorithm with minimum node size 6 and 5 selected predictors.

- 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 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, smoking or not and the vaccinated status also contributes to the outcome.

```
# 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 25.71229