Lab Week 9 : Tree and ensemble methods

STAT5003

Dr. Justin Wishart

Contents

1 Load Data **1**

2 Tree Implementation **1**
2.1 Build decision tree model 2
2.2 Assess decision tree model 2

3 Random forest implementation **3**
3.1 Fit Random Forest 3
3.2 Assess random forest 3
3.3 Calibration 3

4 Optional Boosting **4**
4.1 `gbm` fitting 4
4.2 `xgboost` fitting 5

Preparation and assumed knowledge

- Viewed the tree and ensemble method content in Module 9.
- Installed the following packages on your system.
  - `tree`, `randomForest` (or `ranger`), `gbm`, and `xgboost`
- Downloaded the `wine-quality.csv` dataset from Canvas.

Aims

- Implement trees and ensemble methods, specifically randomforest and boosting.

1 Load Data

Load the `winequality-data.csv` data from the course Canvas website. Note that the last column is the ID of each sample and should be removed from further analysis.

```r
data <- read.csv("winequality-data.csv", header = TRUE)
data <- data[, -ncol(data)]
```

2 Tree Implementation

Split the dataset into 50% training and 50% testing.

```r
library(caret)
```

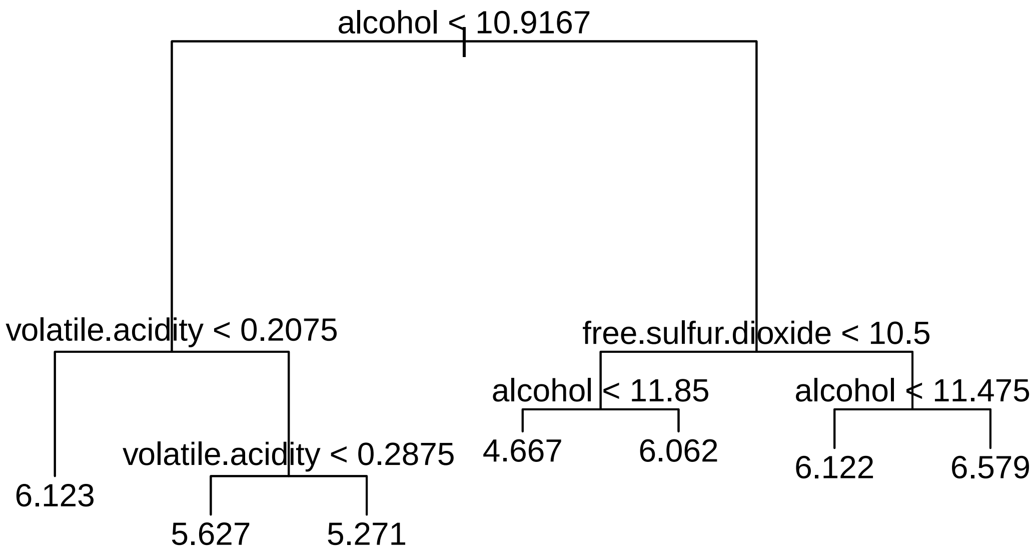
```
## Loading required package: ggplot2
## Loading required package: lattice
```

```
inTrain <- createDataPartition(data[["quality"]], p = 0.5)[[1]]
wine.train <- data[inTrain,-13]
wine.test <- data[-inTrain,-13]
```

2.1 Build decision tree model

Build a decision tree model to predict wine *quality* using the 11 other features using the `tree()` function in the `tree` package and visualize the results

```
library(tree)
tree.model <- tree(quality ~ ., data = wine.train)
plot(tree.model)
text(tree.model)
```



```
## cat("\n")
```

2.2 Assess decision tree model

Use the single decision tree model to predict wine quality in the test set and calculate the residual sum of squares error.

```
# Use the single decision tree to predict test data
tree.test <- predict(tree.model, newdata = wine.test)

# RSS - residual sum of squares
RSS <- sum((tree.test - wine.test$quality)^2)
# MSE - mean squared error
MSE <- mean((tree.test - wine.test$quality)^2)

print(RSS)
```

```
## [1] 1151.806
```

```
print(MSE)
```

```
## [1] 0.5879559
```

3 Random forest implementation

Use the `randomForest` or `ranger` package for this question.

3.1 Fit Random Forest

Implement a random forest classifier trained on the training data from question 2 and use it to predict the test data.

```
library(ranger)

rf.model <- ranger(quality ~. , data = wine.train)
rf.test <- predict(rf.model, data = wine.test)
```

3.2 Assess random forest

Calculate the RSS on the test set prediction and compare it to the result of the single decision tree from question 2.

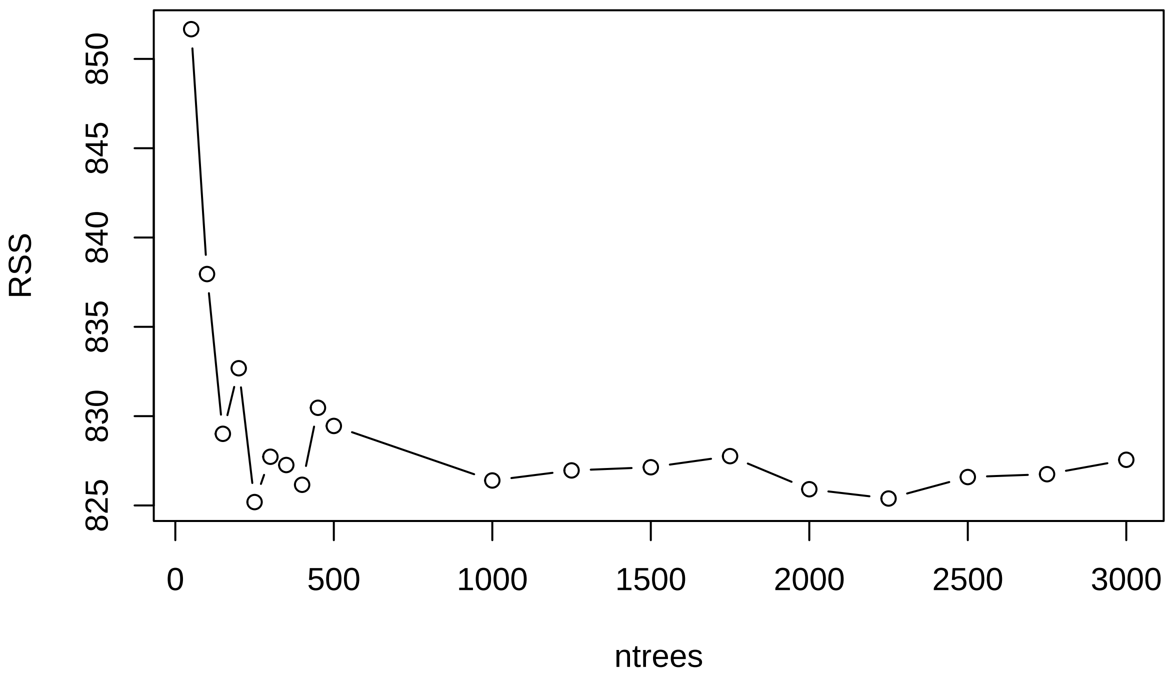
```
RSS.rf <- sum((rf.test$predictions - wine.test$quality)^2)

# Test set RSS from random forest
print(RSS.rf)
```

```
## [1] 830.8845
```

3.3 Calibration

There are two parameters to tune in the Random Forest model — number of trees to use (`ntree`in `randomForest` and `num.trees` in `ranger`) and the number of variables to consider at each split (`mtry` in both packages). Find the `ntree` (or `num.trees`) value at which adding more trees does not improve performance.

```
set.seed(5003)
# Trying different values of ntree
RSS.test <- c()
ntree.seq <- c(seq(from = 50, to = 500, by = 50),
               seq(from = 1000, to = 3000, by = 250))
for(i in ntree.seq) {
  rf.model <- ranger(quality ~ ., data = wine.train, num.trees = i)
  rf.test <- predict(rf.model, data = wine.test)
  RSS.rf <- sum((rf.test$predictions - wine.test$quality)^2)
  RSS.test <- c(RSS.test, RSS.rf)
}
plot(ntree.seq, RSS.test, type = 'b', xlab = "ntrees", ylab = "RSS")
```



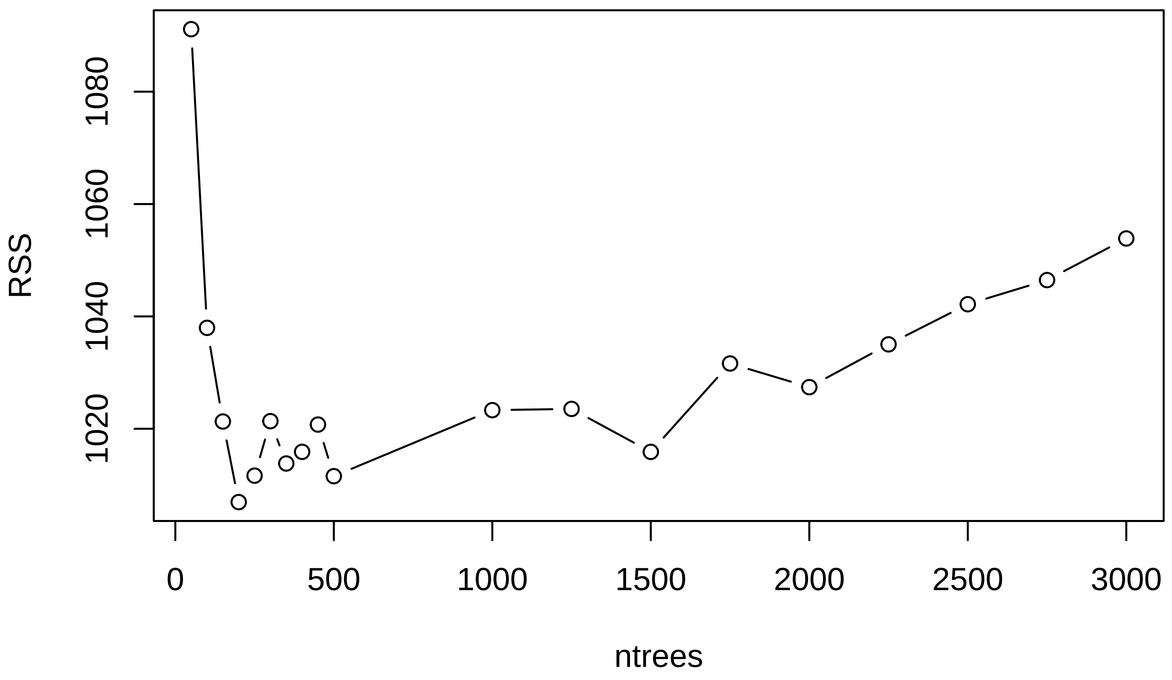
After a few hundred trees the performance doesnt seem to improve and it stabilizes.

4 Optional Boosting

4.1 gbm fitting

Use the `gbm` package for this question. Implement a boosted tree model trained on the training data from question 2 and use it to predict the test data. What is the RSS on the test set prediction and how does it compare to the random forest model? Again, try different number of trees.

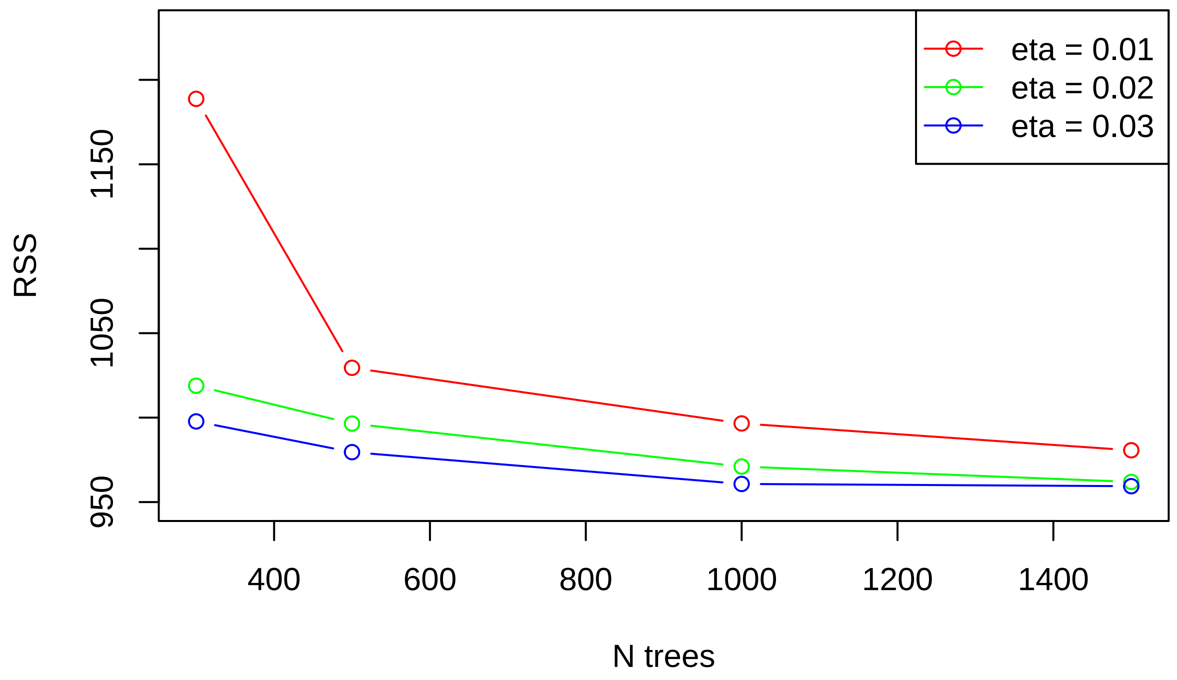
```
library(gbm)

RSS.test.gbm <- c()
ntree.seq <- c(seq(from = 50, to = 500, by = 50), seq(from = 1000, to = 3000, by = 250))
for(i in ntree.seq) {
  gbm.model <- gbm(quality ~ ., data = wine.train,
   distribution = "gaussian", n.trees = i)
  gbm.test <- predict(gbm.model, newdata = wine.test, n.trees = i)
  RSS.gbm <- sum((gbm.test - wine.test$quality)^2)
  RSS.test.gbm <- c(RSS.test.gbm, RSS.gbm)
}

# Should notice the RSS is higher for the gbm model compared to random forest

plot(ntree.seq, RSS.test.gbm, type = 'b', xlab = "ntrees", ylab = "RSS")
```



4.2 xgboost fitting

Use the `xgboost` package for this question. Implement a boosted tree model trained on the training data from question 2 and use it to predict the test data. Use `max_depth = 2` and plot the RSS for `nrounds = 300, 500, 1000, 1500` and `eta = 0.01, 0.02, 0.03`. Check that a small value for eta is required for larger value of nrounds.

```r
library(xgboost)

RSS.test.xgb <- matrix(0, nrow = 4, ncol = 3)
j = 1
for(eta in c(0.01, 0.02, 0.03)) {
    i = 1
    for(nrounds in c(300,500,1000,1500)) {
    xgb.model <- xgboost(data = as.matrix(wine.train[,-12]),
                         label =wine.train[,12], nrounds = nrounds,
                         max_depth = 2, eta = eta, verbose = FALSE)
    xgb.test <- predict(xgb.model, newdata = as.matrix(wine.test[,-12]))
    RSS.xgb <- sum((xgb.test - wine.test$quality)^2)

    RSS.test.xgb[i, j] <- RSS.xgb
    i = i + 1
    }
    j <- j + 1
}

plot(x = c(300,500,1000,1500), y =RSS.test.xgb[,1], col = "red", type = "b", ylim = c(950,1230),
     xlab = "N trees", ylab = "RSS")
points(x = c(300, 500,1000,1500), y =RSS.test.xgb[,2], col = "green", type = "b")
points(x = c(300, 500,1000,1500), y =RSS.test.xgb[,3], col = "blue", type = "b")
legend("topright", legend = paste0("eta = ", c(0.01, 0.02, 0.03)), col = c("red", "green", "blue"), lty
```


eta = 0.01
eta = 0.02
eta = 0.03
RSS
950
1050
1150
400
600
800
1000
1200
1400
N trees