NBA 4920/6921 Lecture 18

Ensemble Methods: Bagging Application

Murat Unal

11/02/2021

```r
rm(list=ls())
options(digits = 3, scipen = 999)
library(tidyverse)
library(ISLR)
library(sandwich)
library(jtools)
library(caret)
library(glmnet)
library(rpart)
library(rpart.plot)
library(ROCR)
library(ipred)
library(vip)
set.seed(2)
```

```r
Hitters <- ISLR::Hitters
Hitters <- na.omit(Hitters)
```

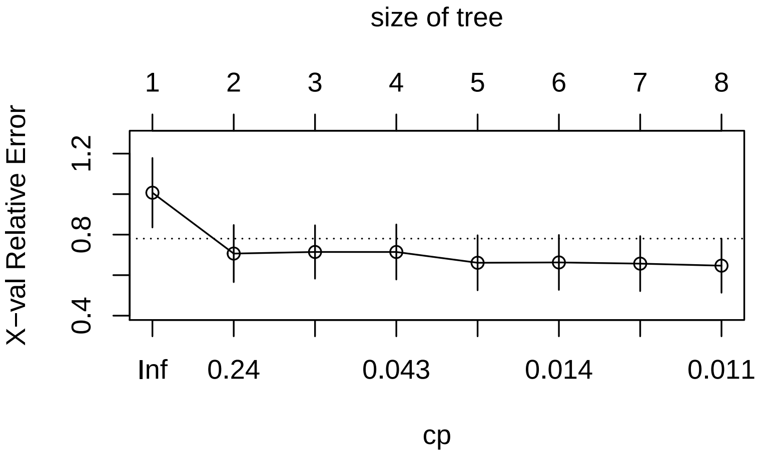
```r
train = sample(1:nrow(Hitters), 0.7*nrow(Hitters))
```

Regression Tree

```r
hit_tree = rpart(Salary ~ .,data = Hitters[train,],
                 method="anova")

hit_tree$cptable
```

```
      CP nsplit rel error xerror  xstd
1 0.3639      0     1.000  1.007 0.172
2 0.1576      1     0.636  0.707 0.141
3 0.0524      2     0.479  0.714 0.132
4 0.0348      3     0.426  0.714 0.136
5 0.0145      4     0.391  0.661 0.136
6 0.0126      5     0.377  0.663 0.136
7 0.0122      6     0.364  0.657 0.136
8 0.0100      7     0.352  0.647 0.134
```

```
plotcp(hit_tree)
```



```r
pruned_hit_tree <- prune(hit_tree, cp=0.24)
pruned_hit_tree
```

```
n= 184

node), split, n, deviance, yval
      * denotes terminal node

1) root 184 40400000 537
  2) CRuns< 218 87  5140000 238 *
  3) CRuns>=218 97 20500000 804 *
```

- Make predictions on the test data

```
pred.tree <- predict(pruned_hit_tree, Hitters[-train,])
rmse.tree <- sqrt(mean((Hitters[-train,"Salary"] -
                        pred.tree)^2))
rmse.tree
```

```
[1] 323
```

Bagging

- The `bagging()` function comes from the `ipred` package and we use `nbagg` to control how many iterations to include in the bagged model and `coob = TRUE` indicates to use the `OOB` error rate

```r
hit_bag1 <- bagging(Salary ~ .,data = Hitters[train,],
                    nbagg=100,coob=TRUE,
                    control=rpart.control(cp=0))
hit_bag1
```

```
Bagging regression trees with 100 bootstrap replications

Call: bagging.data.frame(formula = Salary ~ ., data = Hitte
    ], nbagg = 100, coob = TRUE, control = rpart.control(cp

Out-of-bag estimate of root mean squared error:  316
```

- Make predictions on the test data

```
pred.bag1 <- predict(hit_bag1, Hitters[-train,])
rmse.bag1 <- sqrt(mean((Hitters[-train,"Salary"] -
                         pred.bag1)^2))
rmse.bag1
```

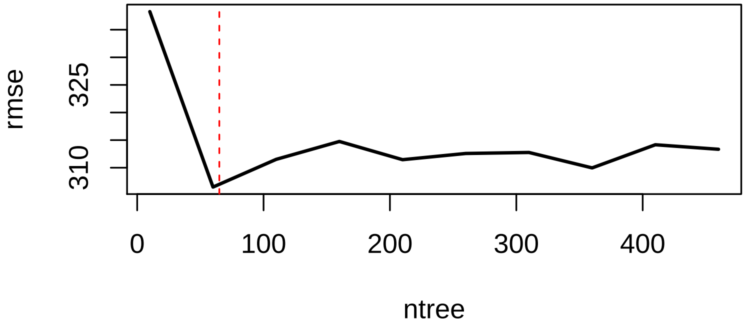
```
[1] 269
```

- We can assess the error versus number of trees as below.

```r
# assess 10-500 bagged trees
ntree <- seq(10,500,by=50)
# create empty vector to store OOB RMSE values
rmse <- vector(mode = "numeric", length = length(ntree))

for (i in seq_along(ntree)) {
  # perform bagged model
  model <- bagging( formula = Salary ~ .,
  data    = Hitters[train,],coob    = TRUE,
  nbagg   = ntree[i])
  # get OOB error
  rmse[i] <- model$err}
```

- We see that the error is stabilizing at about 65 trees so we will likely not gain much improvement by simply bagging more trees.

```
plot(ntree, rmse, type = 'l', lwd = 2)
abline(v = 65, col = "red", lty = "dashed")
```



- We can also apply bagging within caret and use 10-fold CV to see how well our ensemble will generalize

```
hit_bag2 <- train(
  Salary ~ .,
  data = Hitters[train,],
  method = "treebag",
  trControl = trainControl(method = "cv", number = 10),
  nbagg = 100,
  control = rpart.control(cp = 0))
hit_bag2
```

```
Bagged CART

184 samples
 19 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 165, 166, 166, 167, 166, 165, ...
Resampling results:
```

- Make predictions on the test data

```
pred.bag2 <- predict(hit_bag2, Hitters[-train,])
rmse.bag2 <- sqrt(mean((Hitters[-train,"Salary"] -
                          pred.bag2)^2))
rmse.bag2
```

```
[1] 269
```

- Compare prediction performance

```
rmse.tree
```

```
[1] 323
```

```
rmse.bag1
```

```
[1] 269
```

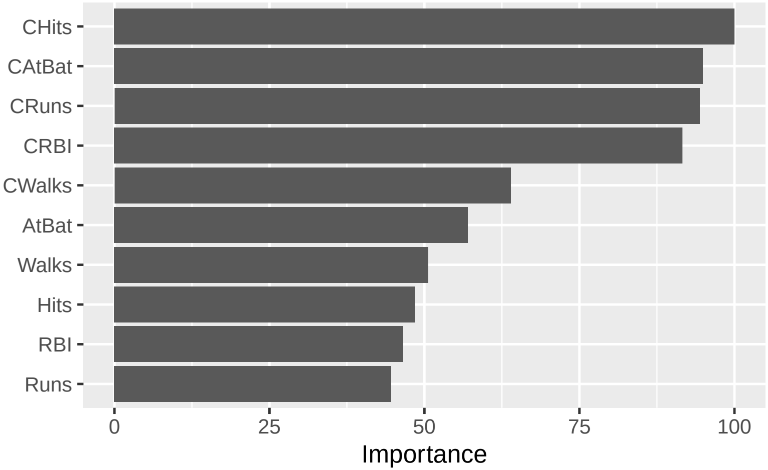
```
rmse.bag2
```

```
[1] 269
```

Variable Importance

-We measure feature importance based on the sum of the reduction in the loss function (e.g., SSE) attributed to each variable at each split in a given tree.

```
vip(hit_bag2)
```



Exercise

```
Carseats <- ISLR::Carseats
Carseats = na.omit(Carseats)
Carseats$Sales = as.factor(ifelse(Carseats$Sales <= 8,
                                  "Low", "High"))
train = sample(1:nrow(Carseats), 0.7*nrow(Carseats))
```

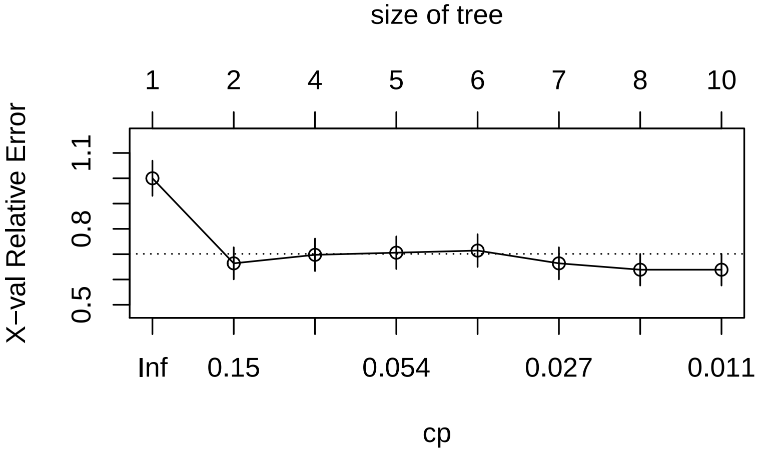
```r
sales_tree = rpart(Sales ~ .,data = Carseats[train,],
                   method="class")

sales_tree
```

```
n= 280

node), split, n, loss, yval, (yprob)
      * denotes terminal node

  1) root 280 119 Low (0.4250 0.5750)
    2) ShelveLoc=Good 64  12 High (0.8125 0.1875)
      4) Price< 142 54   4 High (0.9259 0.0741) *
      5) Price>=142 10   2 Low (0.2000 0.8000) *
    3) ShelveLoc=Bad,Medium 216  67 Low (0.3102 0.6898)
      6) Price< 110 92  46 High (0.5000 0.5000)
       12) CompPrice>=130 21   2 High (0.9048 0.0952) *
       13) CompPrice< 130 71  27 Low (0.3803 0.6197)
         26) Price< 92.5 29  11 High (0.6207 0.3793)
           52) ShelveLoc=Medium 20   4 High (0.8000 0.2000)
```

```
sales_tree$cptable
```

```
      CP nsplit rel error xerror   xstd
1 0.3361      0     1.000  1.000 0.0695
2 0.0714      1     0.664  0.664 0.0633
3 0.0588      3     0.521  0.697 0.0642
4 0.0504      4     0.462  0.706 0.0644
5 0.0420      5     0.412  0.714 0.0647
6 0.0168      6     0.370  0.664 0.0633
7 0.0126      7     0.353  0.639 0.0625
8 0.0100      9     0.328  0.639 0.0625
```

```
plotcp(sales_tree)
```



- Make predictions on the test data

```
sales_pred <- data.frame("p_hat"=predict(sales_tree,
                Carseats[-train,],type = "prob")[,"High"],
                        "predicted"=predict(sales_tree,
                Carseats[-train,], type = "class"),
                        "actual"=Carseats[-train,"Sales"])
```

- Call the `confusion matrix`

```
cm <- confusionMatrix(data=sales_pred$predicted,
                      reference=sales_pred$actual,
                      positive="High")
cm$table
```

```
          Reference
Prediction High Low
      High   29  18
      Low    16  57
```

- Performance metrics

```
c(cm$overall[1],cm$byClass[c(1,2,7)])
```

```
   Accuracy Sensitivity Specificity          F1
      0.717       0.644       0.760       0.630
```

```r
pruned_sales_tree <- prune(sales_tree, cp=0.15)
pruned_sales_tree
```

```
n= 280

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 280 119 Low (0.425 0.575)
  2) ShelveLoc=Good 64  12 High (0.812 0.187) *
  3) ShelveLoc=Bad,Medium 216  67 Low (0.310 0.690) *
```

- Make predictions on the test data

```
pruned_sales_pred <- data.frame("p_hat"=predict(pruned_sale
                Carseats[-train,],type = "prob")[,"High"],
                        "predicted"=predict(pruned_sales_tr
                Carseats[-train,], type = "class"),
                        "actual"=Carseats[-train,"Sales"])
```

- Call the confusion matrix

```
cm_pruned <- confusionMatrix(data=pruned_sales_pred$predict
                 reference=pruned_sales_pred$actual,
                 positive="High")
cm_pruned$table
```

```
          Reference
Prediction High Low
      High   14   7
      Low    31  68
```

- Performance metrics

```
c(cm_pruned$overall[1],cm_pruned$byClass[c(1,2,7)])
```

```
   Accuracy Sensitivity Specificity          F1
      0.683       0.311       0.907       0.424
```

```r
sales_bag <- bagging(Sales ~ .,data = Carseats[train,],
                     nbagg=100,coob=TRUE,
                     control=rpart.control(cp=0))
sales_bag
```

```
Bagging classification trees with 100 bootstrap replicatio

Call: bagging.data.frame(formula = Sales ~ ., data = Carsea
    ], nbagg = 100, coob = TRUE, control = rpart.control(cp

Out-of-bag estimate of misclassification error:  0.171
```

- Make predictions on the test data

```
sales_pred_bag <- data.frame("p_hat"=predict(sales_bag,
                Carseats[-train,],type = "prob")[,"High"],
                        "predicted"=predict(sales_bag,
                Carseats[-train,], type = "class"),
                        "actual"=Carseats[-train,"Sales"])
```

- Call the confusion matrix

```r
cm_bag <- confusionMatrix(data=sales_pred_bag$predicted,
                reference=sales_pred_bag$actual,
                positive="High")
cm_bag$table
```

```
          Reference
Prediction High Low
      High   27   9
      Low    18  66
```

- Performance metrics

```
c(cm$overall[1],cm$byClass[c(1,2,7)])
```

```
   Accuracy Sensitivity Specificity          F1
      0.717       0.644       0.760       0.630
```

```
c(cm_pruned$overall[1],cm_pruned$byClass[c(1,2,7)])
```

```
   Accuracy Sensitivity Specificity          F1
      0.683       0.311       0.907       0.424
```

```
c(cm_bag$overall[1],cm_bag$byClass[c(1,2,7)])
```

```
   Accuracy Sensitivity Specificity          F1
      0.775       0.600       0.880       0.667
```

- Variable Importance

```
varImp(sales_bag)%>%arrange(-Overall)
```

	Overall
Price	53.05
ShelveLoc	45.29
Advertising	40.12
Income	29.85
Age	28.72
CompPrice	23.55
Population	12.99
US	7.54
Education	6.15
Urban	2.43