# Tutorial 7: Tree-based Methods

## ECO3080: Machine Learning in Business

Instructor: Prof. Qihui Chen
Teaching Assistant: Long Ma

November 15, 2022

# Today's Roadmap

# A Simple Classification Tree

- The dataset we use in this example is "Carseat".
- We want to predict "Sales" using other features.
- Transform "Sales" into categories:
  1. "Sales <= 8", "High == no"
  2. "Sales > 8", "High == yes"
- The preprocessing is:

```
####  Get the data we want to use
####  we want to predict the sales of cars by other features
library(ISLR)
Data001 <- Carseats
High <- as.factor(ifelse(Data001$Sales <= 8, "No", "Yes"))
Data002 <- data.frame(Data001, High)
```

# A Simple Classification Tree

- Separate the dataset into training set and the test set.
- The code is as follows:

```
set.seed(911)
train <- sample(1:nrow(Data002), 200)
Testset <- Data002[-train, ]
High.test <- High[-train]
```

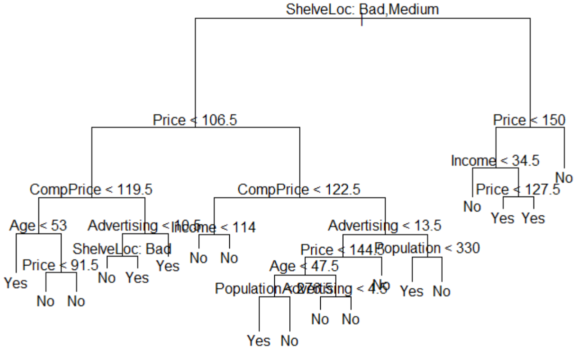
- Then, build up a simple tree on the training set.
- The code is as follows:

```
TrainTree <- tree(High ~ . -Sales, Data002, subset = train)
plot(TrainTree)
text(TrainTree, pretty = 0)
```

# A Simple Classification Tree



- The plot is like:



- Use this tree to make prediction on test set:

```
Pred001 <- predict(TrainTree, Testset, type = "class")
table(Pred001, High.test)
```

- The confusion matrix is:

```
          High.test
Pred001 No Yes
    No  87  36
    Yes 28  49
```

- Then, we can calculate a lot of things (sensitivity, lift...). The most straightforward indicator is: $(87+49)/200 = 0.68$

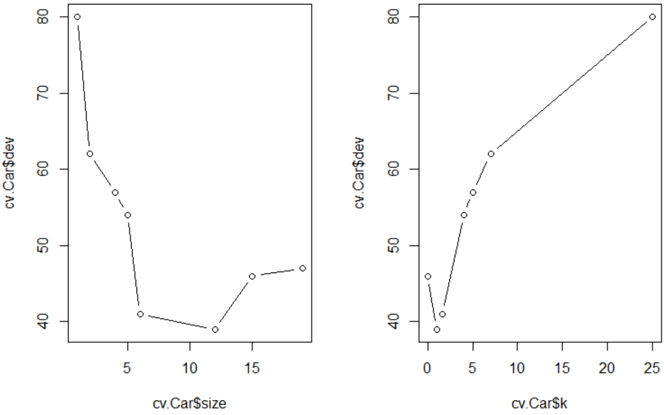
- Then, consider how to prune this tree. CV is feasible. The code is like following:

```
set.seed(1997)
cv.Car <- cv.tree(TrainTree, FUN = prune.misclass)
cv.Car

par(mfrow = c(1, 2))
plot(cv.Car$size, cv.Car$dev, type = "b")
plot(cv.Car$k, cv.Car$dev, type = "b")
```

- Thus, we can choose the best number of leaves (From the figures below, 6 or 12 may be good choices. From my perspective, I prefer to choose a simpler tree where size $= 6$) .

# A Simple Classification Tree

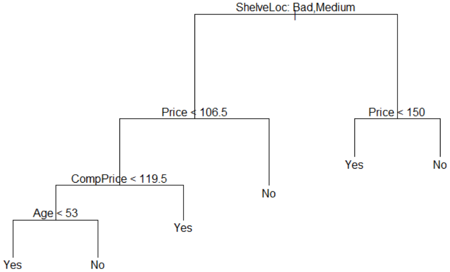




# A Simple Classification Tree

- Plug the best "6" into the function "prune.misclass", then we can get a pruned tree.

```
prune.Car <- prune.misclass(TrainTree, best = 6)

par(mfrow = c(1, 1))
plot(prune.Car)
text(prune.Car, pretty = 0)
```



- Use this pruned tree to make prediction on test set.

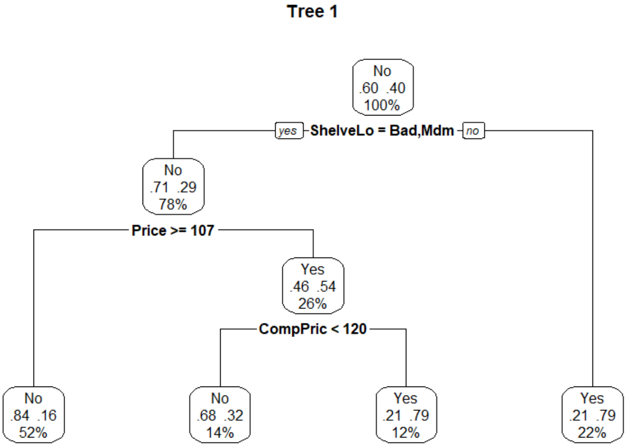
```
Pred002 <- predict(prune.Car, Testset, type = "class")
table(Pred002, High.test)
```

- The confusion matrix is:

```
          High.test
Pred002 No  Yes
    No  92   38
    Yes 23   47
```

- $(92+47)/200 = 0.695$

- There is another way to build up a tree. "rpart"

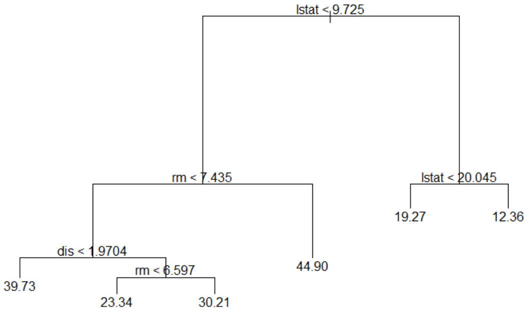


# A Simple Regression Tree

- Same logic:

```
####  Revisit the data set "Boston"
library(MASS)
set.seed(911)
train <- sample(1:nrow(Boston), nrow(Boston)/2)

####  Build up a regression tree
library(tree)
tree.boston <- tree(medv ~ ., data = Boston, subset = train)
summary(tree.boston)

####  Plot this tree
plot(tree.boston)
text(tree.boston, pretty = 0)

####  Use cross validation to get the optimal number of terminal nodes
cv.boston <- cv.tree(tree.boston)
plot(cv.boston$size, cv.boston$dev, type = "b")

prune.boston <- prune.tree(tree.boston, best = 6)
plot(prune.boston)
text(prune.boston, pretty = 0)
```

- A pruned tree:



# A Simple Regression Tree

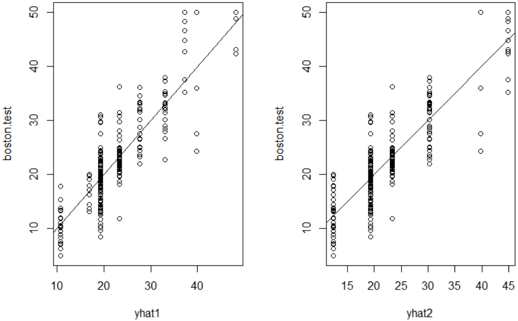
- Prediction:

```r
####  Make predictions on test set
yhat1 <- predict(tree.boston, newdata = Boston[-train, ])
yhat2 <- predict(prune.boston, newdata = Boston[-train, ])

boston.test <- Boston[-train, "medv"]

par(mfrow = c(1, 2))
plot(yhat1, boston.test)
abline(0, 1)

plot(yhat2, boston.test)
abline(0, 1)

mean((yhat1 - boston.test)^2)
mean((yhat2 - boston.test)^2)
```

# A Simple Regression Tree



- Prediction:




- $mean((yhat1 - boston.test)^2) = 20.14976$
- $mean((yhat2 - boston.test)^2) = 19.75798$

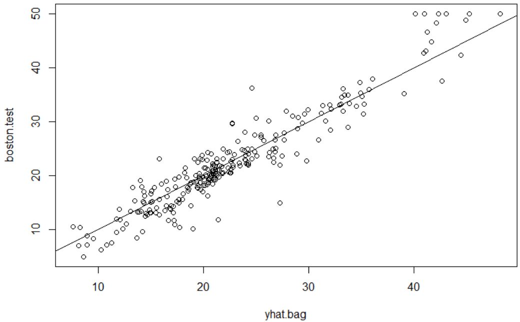
# A Simple Regression Tree



- You can also use rpart() to generate a regression tree;
- Please choose the "method = "anova"";
- Also, by using the package "fancyRpartPlot", you are able to make your plots prettier and fancier.

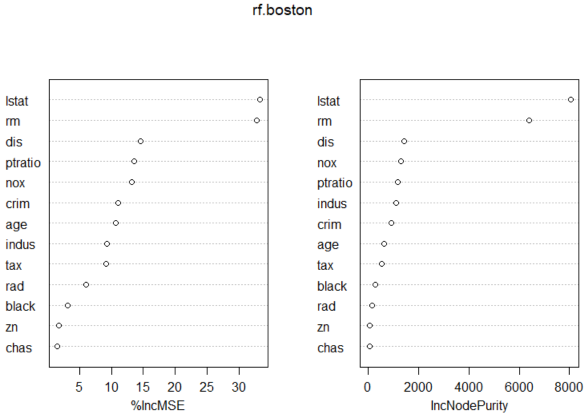
- The code for bagging (from the text book) is:

```
###################### 3 Bagging and Random Forest  ############################
################################################################################
install.packages("randomForest")
library(randomForest)
set.seed(911)
bag.boston <- randomForest(medv ~ ., data = Boston, subset = train, mtry = 13,
                           importance = TRUE) ## why mtry = 13?
####  when mtry = the number of variables, then randomforest == bagging
bag.boston

####  make predictions on test set
yhat.bag <- predict(bag.boston, newdata = Boston[-train, ])
par(mfrow = c(1, 1))
plot(yhat.bag, boston.test)
abline(0, 1)
mean((yhat.bag - boston.test)^2)
```


boston.test
yhat.bag
10
20
30
40
50

- The code for RF is:

```
####  Change the number of trees
bag.boston <- randomForest(medv ~., data = Boston, subset = train,
                           mtry = 13, ntree = 25)
yhat.bag <- predict(bag.boston, newdata = Boston[-train, ])
mean((yhat.bag - boston.test)^2)

####  randomForest
set.seed(911)
rf.boston <- randomForest(medv ~., data = Boston, subset = train,
                          mtry = 6, importance = TRUE)
yhat.rf <- predict(rf.boston, newdata = Boston[-train, ])
mean((yhat.rf - boston.test)^2)

importance(rf.boston)
varImpPlot(rf.boston)

#### ipred/adabag can also be used in bagging
```

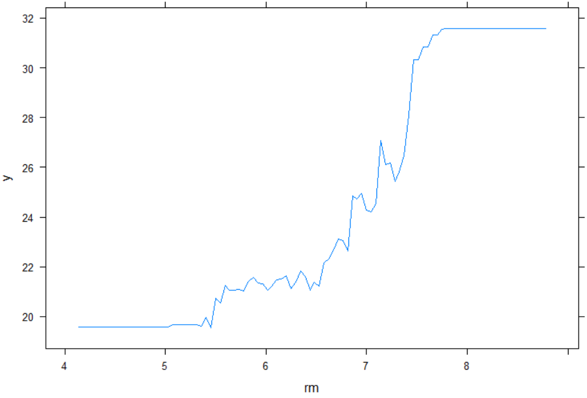


rf.boston
lstat
rm
dis
ptratio
nox
crim
age
indus
tax
rad
black
zn
chas
5
10
15
20
25
30
%IncMSE
lstat
rm
dis
nox
ptratio
indus
crim
age
tax
black
rad
zn
chas
0
2000
4000
6000
8000
IncNodePurity

```r
############################### 4 Boosting  ####################################
################################################################################
install.packages("gbm")
library(gbm)

set.seed(911)
boost.boston <- gbm(medv ~ ., data = Boston[train, ], distribution = "gaussian",
                    n.trees = 5000, interaction.depth = 4)
summary(boost.boston)

par(mfrow = c(1, 2))
plot(boost.boston, i = "rm")
plot(boost.boston, i = "lstat")

yhat.boost <- predict(boost.boston, newdata = Boston[-train, ], n.trees = 5000)
mean((yhat.boost - boston.test)^2)

boost.boston <- gbm(medv ~., data = Boston[train, ], distribution = "gaussian",
                    n.trees = 5000, interaction.depth = 4, shrinkage = 0.2,
                    verbose = F)
yhat.boost <- predict(boost.boston, newdata = Boston[-train, ], n.trees = 5000)
mean((yhat.boost - boston.test)^2)

####  highly recommand you: adaboost, xgboost ......
####  there are lots of materials on the internet, you can check by yourself
```

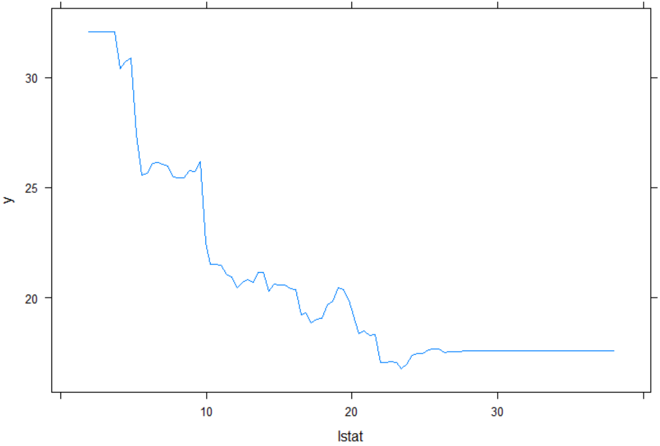
# Boosting





# Boosting





# Boosting

- Adaboost, Xgboost ......