Topic # 06: Data Analytics

Decision Trees

Instructor: Prof. Arnab Bisi, Ph.D.

Johns Hopkins Carey Business School

Tree-based Methods

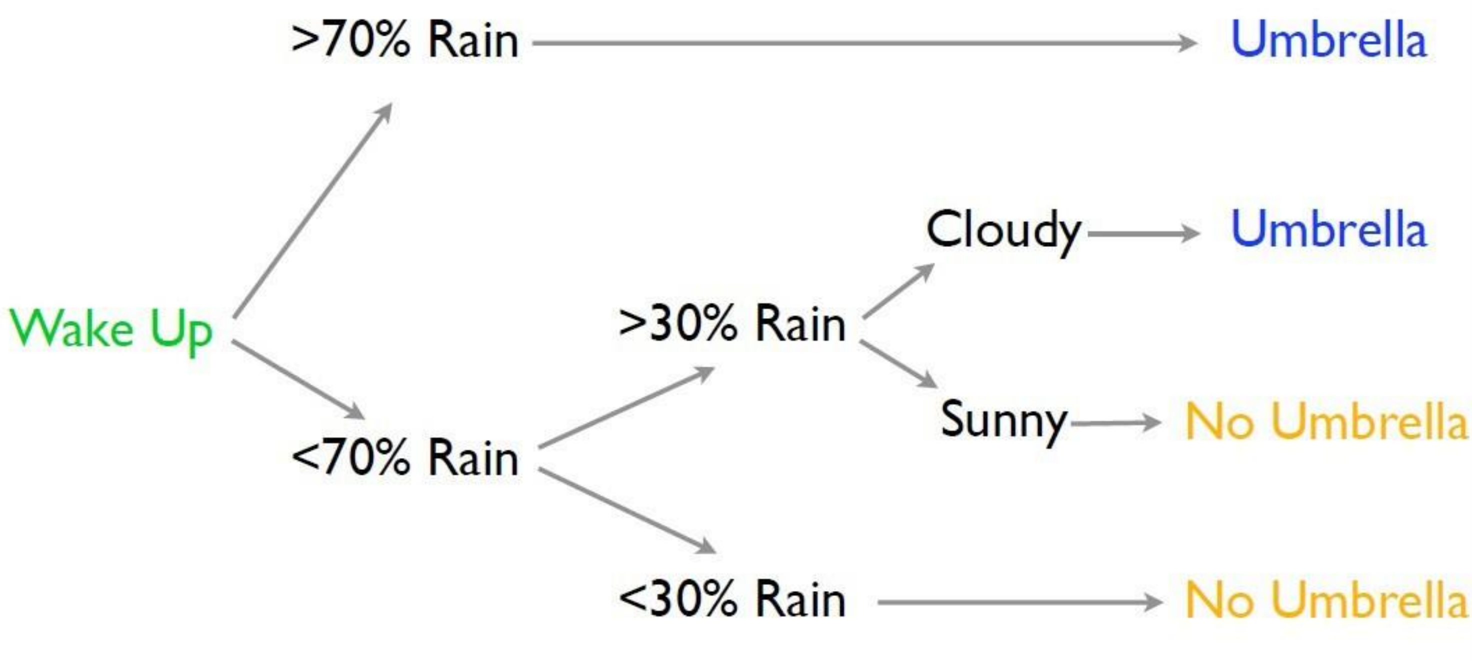
- Tree-based Methods for regression and classification
- Use mean or mode of the training observations to make a prediction
- Key ideas
  - Iteratively split variables into groups
  - Split where maximally predictive
  - Evaluate "homogeneity" within each branch
  - Fitting multiple trees often works better (forests)

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

Tree-based Methods

- Pros
  - Trees are very easy to explain to people.
    - In fact, they are even easier to explain than linear regression!
  - Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches seen in previous sessions.
  - Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).
  - Trees can easily handle qualitative predictors without the need to create dummy variables.
- Cons
  - Without pruning/cross-validation, trees can lead to overfitting
  - Harder to estimate uncertainty
  - Results may be variable

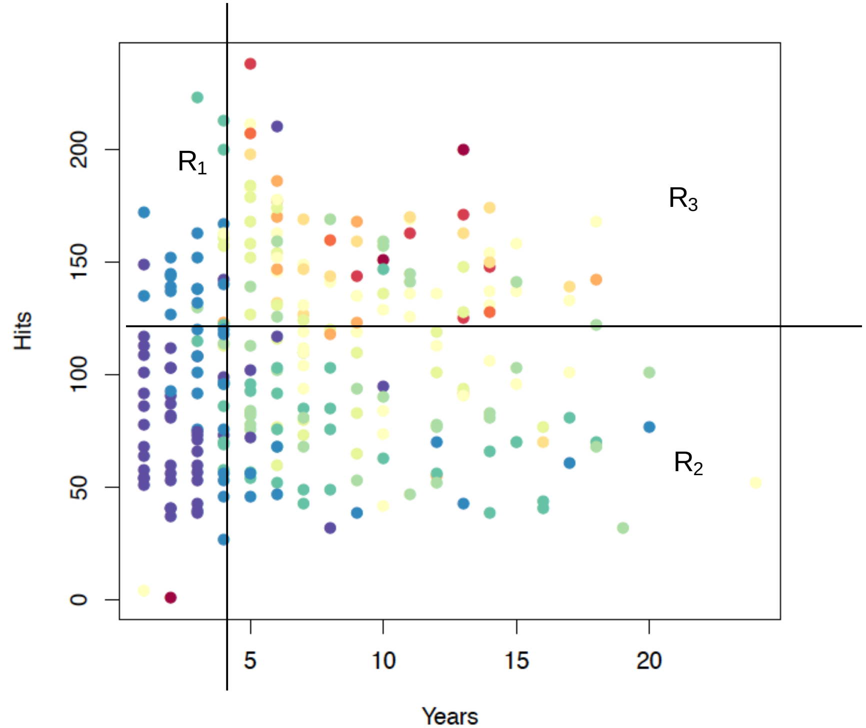
What do we mean by "Decision Tree"?



- Tree-logic uses a series of steps to come to a conclusion. The trick is to have mini-decisions combine for good choices. Each decision is a node, and the final prediction is a leaf node.

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

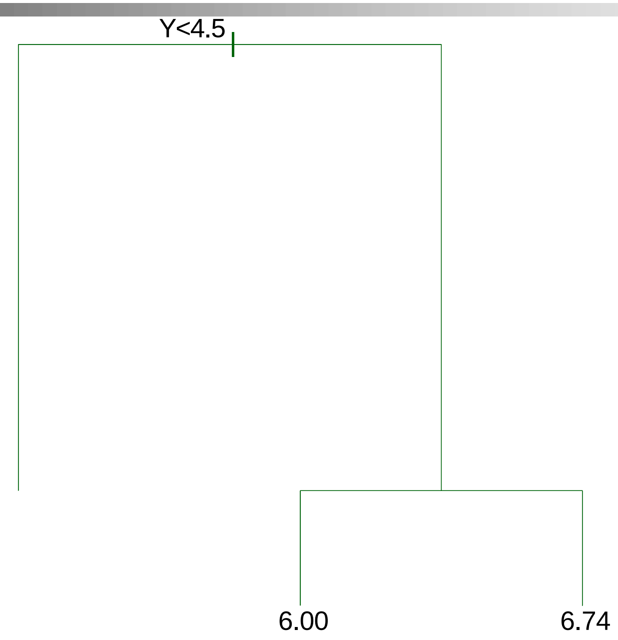
What is Decision Tree? Another Example



- These three regions can be written as:
  - $R_1 = \{X|$ Years $< 4.5\}$
  - $R_2 = \{X|$ Years $>= 4.5$, Hits $< 117.5\}$
  - $R_3 = \{X|$ Years $>= 4.5$, Hits $>= 117.5\}$

JOHNS HOPKINS CAREY BUSINESS SCHOOL

Decision Tree Example



- For the hitters data, a regression tree for predicting the log salary based on the number of years that he has played and the number of hits he got in the previous year
- At a given internal node, $X_j < t_k$ indicates the left hand branch, and the right hand branch corresponds to $X_j >= t_k$

Decision Tree

- Decision Trees are a Regression Model
- You have inputs (forecast, current conditions) and an output of interest (need for an umbrella).
- Based on previous data, the goal is to specify branches of choices that lead to good predictions in new scenarios.
- In other words, you want to estimate a Tree Model.
- Estimation for regression models is not new material, but instead of β's to fit we now have decision nodes

Estimation of Decision Trees

- As usual, we'll maximize data likelihood (minimize deviance). But what are the observation probabilities in a tree model?

- Two types of likelihood: classification and regression trees.
  - A given covariate x dictates your path through tree nodes, leading to a leaf node at the end.

- Classification trees have class probabilities at the leaves.
  - Probability I'll be in heavy rain is 0.9 (so take an umbrella).

- Regression trees have a mean response at the leaves.
  - The expected amount of rain is 2in (so take an umbrella).

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

Estimation of Decision Trees

- Tree deviance is the same as in linear models
  - Classification Deviance: $-\sum_{i=1}^{n} \log(\widehat{p}_{y_i})$
  - Regression Deviance: $\sum_{i=1}^{n}(y_i - \widehat{y}_i)^2$ (sum of squared residuals)
- Instead of being based on $\mathbf{x}'\beta$, predicted $\widehat{p}$ and $\widehat{y}$ are functions of $\mathbf{x}$ passed through the decision nodes.
- We need a way to estimate the sequence of decisions.
  - How many are they? What is the order?
- There is a huge set of possible tree configurations.

Estimation of Decision Trees

- Process of building a regression tree
  - We divide the predictor space, i.e., the set of possible values for $X_1, X_2, \ldots, X_p$ into J distinct and non-overlapping regions, $R_1, R_2, \ldots, R_J$.
  - For every observation that falls into the region R_j, we make the same prediction, which is simply the mean of the response values for the training observations in R_j.
- The goal is to find boxes $R_1, R_2, \ldots, R_J$ that minimize the RSS, given by

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \widehat{y}_{R_j})^2,$$

  $\widehat{y}_{R_j}$ is the mean response for the training observations within the j-th box.
- Unfortunately, it is computationally infeasible to consider every possible partition of the feature space into J boxes.
- For this reason, we take a top-down, greedy approach that is known as recursive binary splitting.

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

CAREY BUSINESS SCHOOL

Estimation of Decision Trees: CART algorithm

- We solve the problem by thinking recursively:
  - Split the data into two different decisions about y.
  - Take each new partition and split again.
- Growing your tree with the CART algorithm:
- Find the split location in $\mathbf{x}$ that minimizes deviance.
  - $\widehat{y}_i$ or $\widehat{p}_i$ change depending on whether $x_i < x_{split}$.
- You then grow the tree at this point
  - Each new child node contains a subset of the data.
  - Each subset has its own $\widehat{y}_i$ or $\widehat{p}_i$ for prediction.
- View each child as a new dataset, and try to grow again.
  - Stop splitting/growing when there are some fixed minimum number of observations in each leaf node.
  - Sometimes there are also minimum deviance improvements to be met before splitting.

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

Use the tree library for CART in R

- The syntax is essentially the same as for lm:
  - mytree=tree(y~x1 + x2 + x3 + …+ xp, data=mydata)
- There are only a few other possible arguments
  - mincut: the minimum size for a new child (default = 5)
  - mindev: the smallest improvement in deviance that accompanies a new split (default = 0.01)
- As usual you can use print, summary, and plot to view a tree

Examples with R: Carseats data set

Library tree()

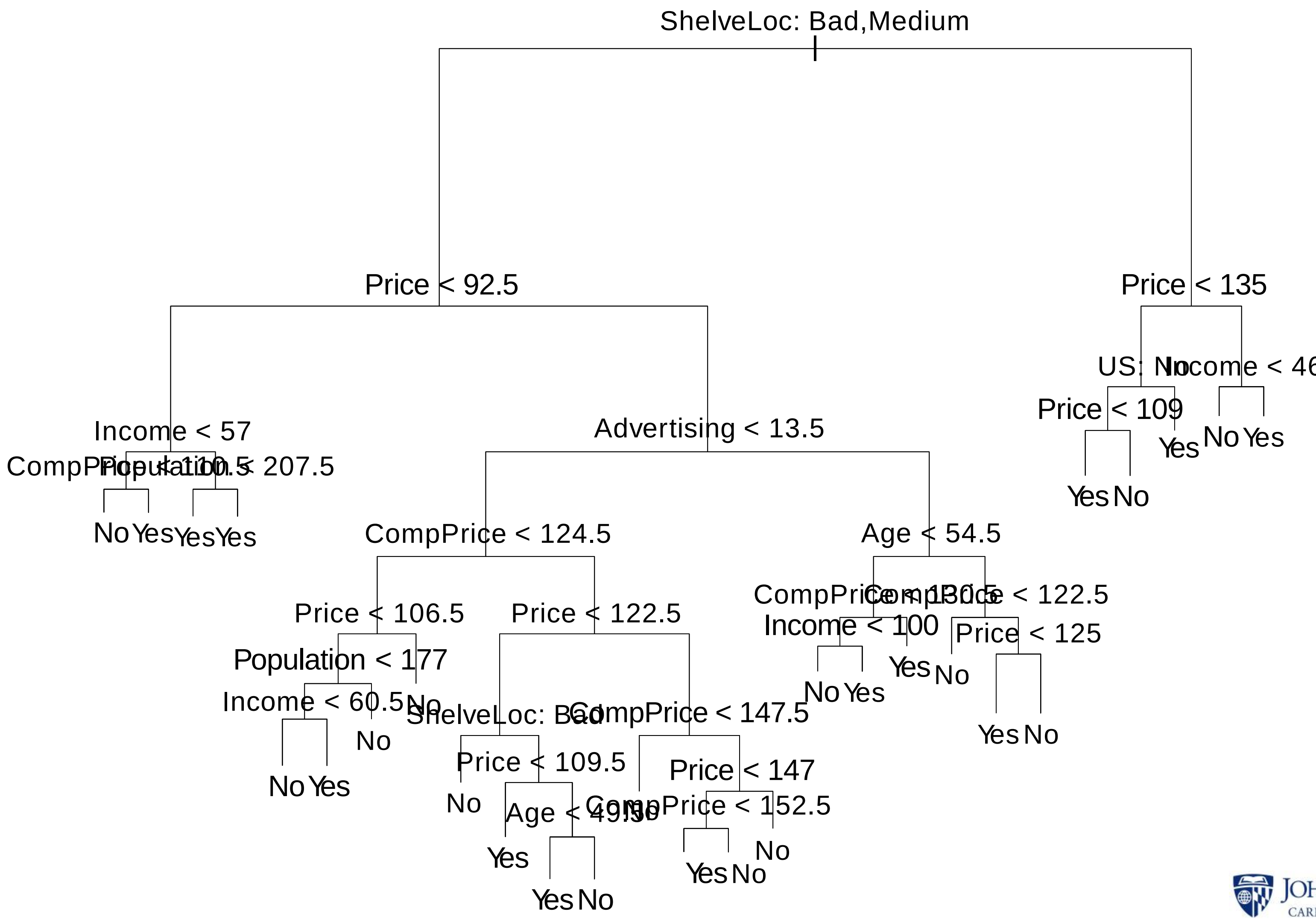
Example with simulated data

```
library(tree)
library(ISLR)
attach(Carseats)
High=ifelse(Sales <= 8, "No", "Yes")

Carseats = data.frame(Carseats, High)
tree.carseats = tree(High˜.-Sales, Carseats)
summary(tree.carseats)

plot(tree.carseats)
text(tree.carseats,pretty=0)
tree.carseats
```

Decision Tree



Tree Pruning

- Biggest challenge with such flexible models is avoiding over-fitting
- For trees the usual solution is to rely on cross validation
- The basic constraints (mincut, mindev) lead to a full tree fit
- Prune tree by removing split rules from the bottom up:
  - At each step, remove the split that contributes least to the reduction in deviance
  - Pruning yields candidate trees and we use CV to choose
  - Each step produces a candidate tree model and we compare prediction performance on test sample

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

Example with R: Carseats data set

Function cv.tree() performs cross-validation in order to determine optimal level of tree

Argument FUN=prune.misclass indicate classification error rate, guide cross-validation and pruning process (default is deviance)

Example with simulated data

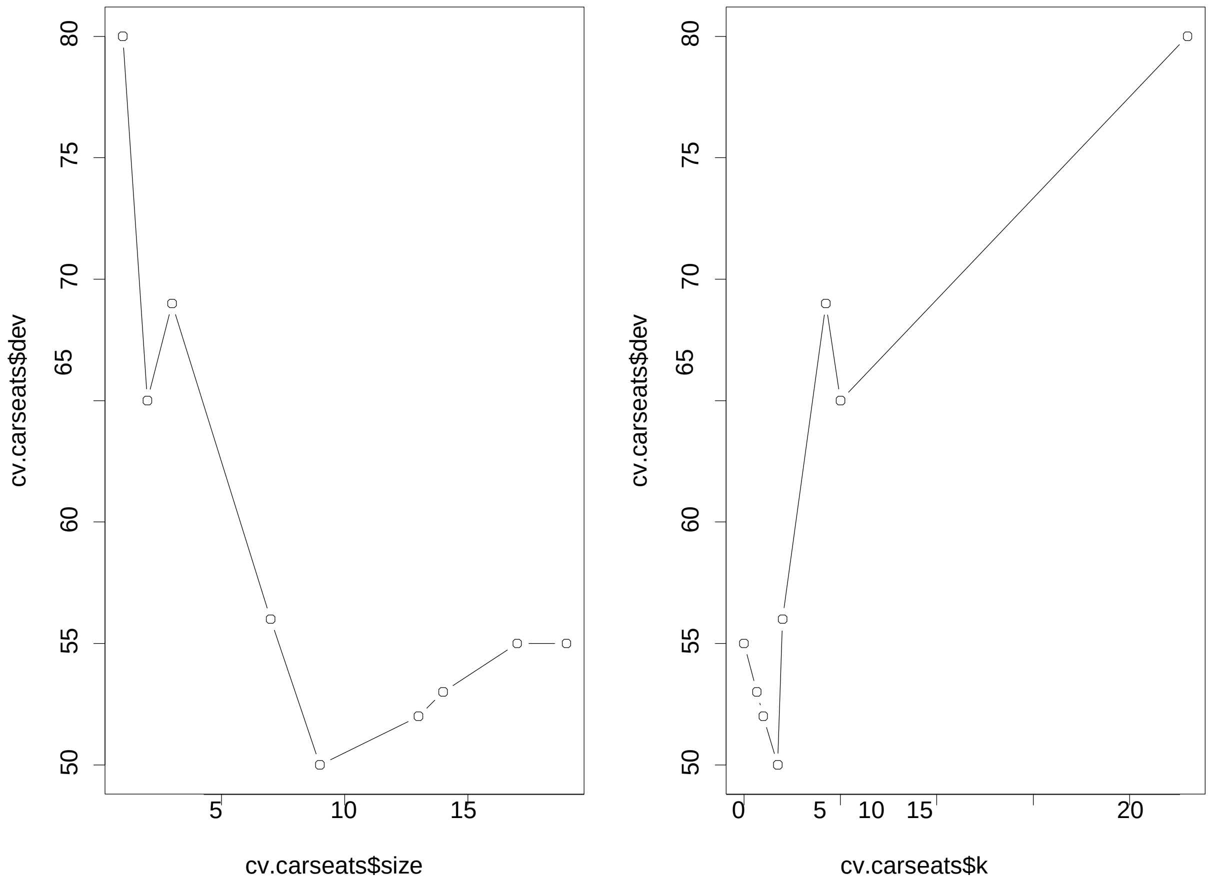
```
set.seed(3)
cv.carseats = cv.tree(tree.carseats,
FUN=prune.misclass)
names(cv.carseats)
cv.carseats

par(mfrow=c(1,2))
plot(cv.carseats$size,cv.carseats$dev,type="b")
plot(cv.carseats$k,cv.carseats$dev,type="b")
prune.carseats =
prune.misclass(tree.carseats,best=9)
plot(prune.carseats)
text(prune.carseats,pretty=0)
```

JOHNS HOPKINS
CAREY BUSINESS SCHOOL

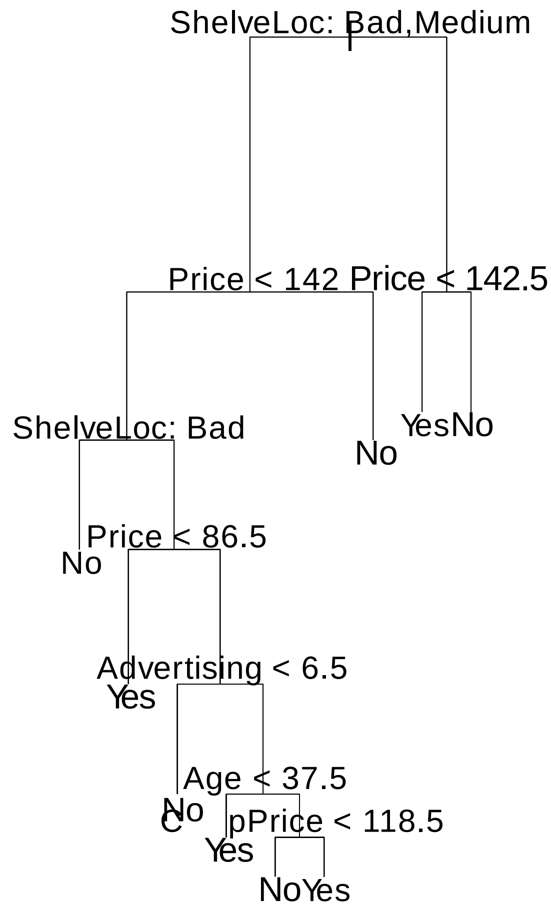
Prune Tree

The tree with 9 terminal nodes results in the lowest error rate.



Prune Tree, cont.

The tree with 9 terminal nodes results in the lowest error rate.



Fitting Regression Trees

Fit a regression tree to Boston data set: first create a training set, and fit the tree to the training data.

```
library(MASS)
set.seed(1)
train = sample(1:nrow(Boston), nrow(Boston)/2)
tree.boston = tree(medv~., data=Boston,
subset=train)
summary(tree.boston)

plot(tree.boston)
text(tree.boston,pretty=0)
```

summary() indicates three variables are used in constructing the tree; the deviance is the sum of squared errors for the tree.

Fitting Regression Trees, cont.

Prune the tree

```
prune.boston  = prune.tree(tree.boston, best=5)
plot(prune.boston)
text(prune.boston,  pretty=0)
```

Make prediction

```
yhat  = predict(tree.boston,  newdata=Boston[-train,])
#attach(Boston)
boston.test=  Boston[-
train,"medv"]
plot(yhat,  boston.test)
abline(0,1)
mean((yhat -  boston.test)^2)
```

The MSE is around 25.05, indicating that this model leads to test predictions within $5,005 of the true median home value.

Exercise

- Use Boston data set
- Perform linear regression w.r.t. the same three predictors on the training data
- Make predictions on the test data
  - Calculate the MSE of the predictions
  - Compare to the tree method

- 15 mins

Questions, Comments?

Let's move to the Code.