Data Analysis 2021 Spring





**Lecture 11:**

**Tree-Based Methods**

May 19 & May 24, 2021

**Taesoo Kwon**

[tskwon@seoultech.ac.kr](mailto:tskwon@seoultech.ac.kr)

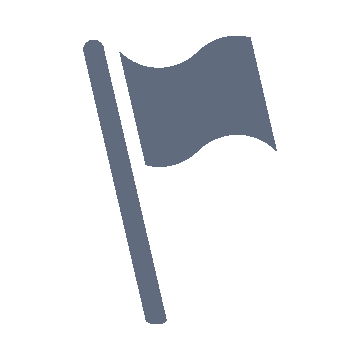
©copyright CIEL 2015 https://ciel.seoultech.ac.kr/

# Course Schedule (Tentative)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Week** | **Topics** | **Note** | **Date (W)** | **Date (M)** |
| 1 | Orientation, Statistical Learning (Ch2) | Online | 03/03 | 03/08 |
| 2 | Statistical Learning (Ch2), Python Programming | Online | 03/10 | 03/15 |
| 3 | Probability & Statistics | Online | 03/17 | 03/22 |
| 4 | Probability & Statistics | Online | 03/24 | 03/29 |
| 5 | Linear Regression (Ch3) | Online | 03/31 | 04/05 |
| 6 | Linear Regression (Ch3) | Online | 04/07 | 04/12 |
| 7 | Classification (Ch4) | Online | 04/14 | 04/19 |
| 8 | **Midterm exam** | **Class hours (W1-W7)** | **04/21** | **04/26** |
| 9 | Resampling Methods (Ch5) | Online | 04/28 | 05/03 |
| 10 | Linear Model Selection and Regularization (Ch6) | Online | 05/05 | 05/10 |
| 11 | Moving Beyond Linearity (Ch7) | Online | 05/12 | 05/17 |
| **12** | Tree-Based Methods (Ch8) | Online | 05/19 | 05/24 |
| 13 | Support Vector Machines (Ch9) | Online | 05/26 | 05/31 |
| 14 | Unsupervised Learning (Ch10) | Online | 06/02 | 06/07 |
| 15 | **Final exam** | **7pm or Class hours (W9-W14)** | **06/??** | **06/??** |

©copyright CIEL 2015 **2**

* Tree-based methods



**OUTLINES**

* + Basics of decision trees
  + Bagging, random forest, boosting
* Python lab
* Summary & Next class

©copyright CIEL 2015



**Tree-Based Methods**



**: Ch 8**

## Tree-based methods

* Python lab
* Summary & Next class

©copyright CIEL 2015 https://ciel.seoultech.ac.kr/

# Tree-based Methods

* Here we describe tree-based methods for regression and classification
* These involve stratifying or segmenting the predictor space into a number of simple regions
* Since the set of splitting rules used to segment the predictor space can be summarized in a tree, these types of approaches are known as decision-tree methods

©copyright CIEL 2015 **5**

# Pros and Cons

* Tree-based methods are simple and useful for interpretation
* However, they typically are not competitive with the best supervised learning approaches in terms of prediction accuracy
* Hence we also discuss bagging, random forests, and boosting.
  + These methods grow multiple trees which are then combined to yield a single consensus prediction
* Combining a large number of trees can often result in dramatic improvements in prediction accuracy, at the expense of some loss interpretation

©copyright CIEL 2015 **6**

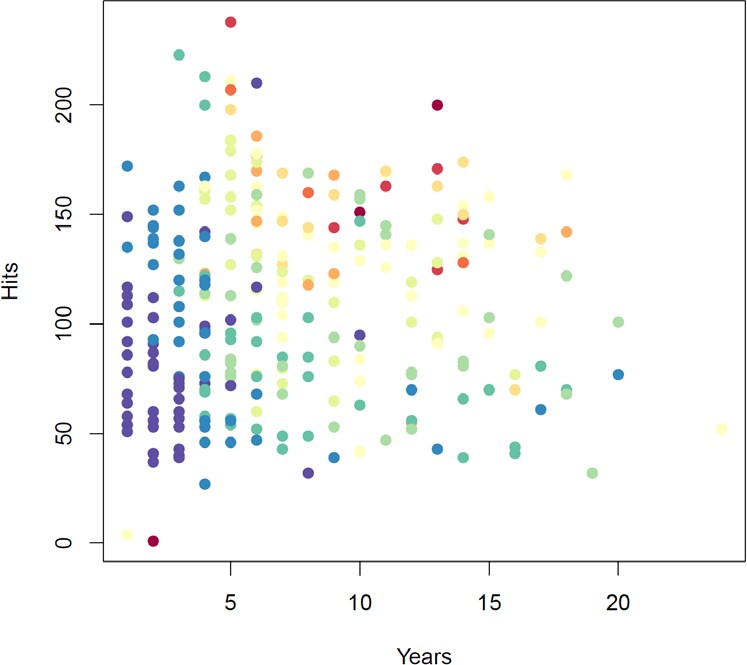
# Basics of Decision Trees

* Decision trees can be applied to both regression and classification problems
* We first consider regression problems, and then move on to classification

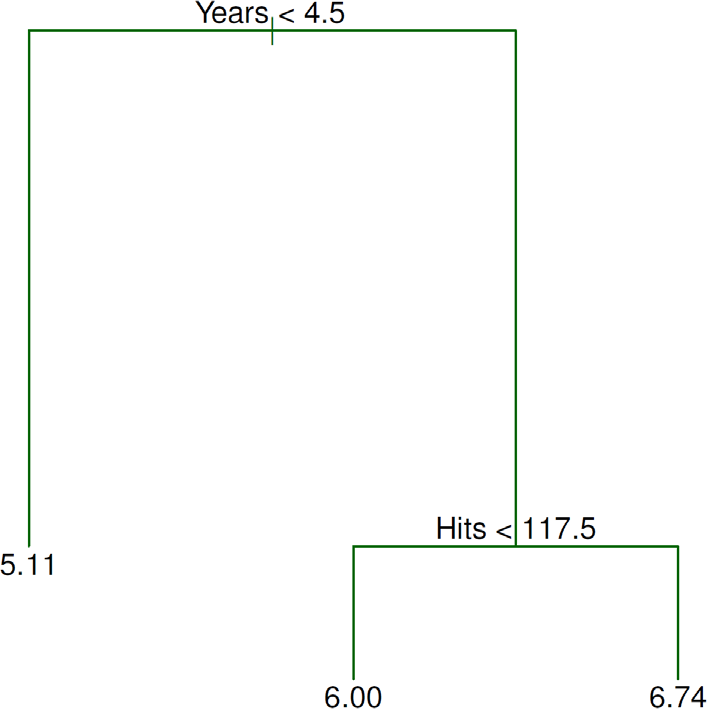
©copyright CIEL 2015 **7**

# Baseball Salary Data: How Would You Stratify It?

* Salary is color-coded from low (blue, green) to high (yellow, red)



* Decision tree for these data



©copyright CIEL 2015 **8**

# Baseball Salary Dataset: Decision Tree

* For Hitters data, a regression tree for predicting the log salary of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year
* At a given internal node, the label (of the form 𝑋𝑋𝑗𝑗 < 𝑡𝑡𝑘𝑘) indicates the left-hand branch emanating from that split, and the right-hand branch corresponds to 𝑋𝑋𝑗𝑗 ≥ 𝑡𝑡𝑘𝑘
  + For instance, the split at the top of the tree results in two large branches
  + The left-hand branch corresponds to Years<4.5, and the right-hand branch corresponds to Years ≥ 4.5
* The tree has two internal nodes and three terminal nodes, or leaves
  + The number in each leaf is the mean of the response for the observations that fall there

©copyright CIEL 2015 **9**

# Baseball Salary Dataset: Results

* Overall, the tree stratifies or segments the players into three regions of predictor space:

𝑋𝑋|Years < 4.5

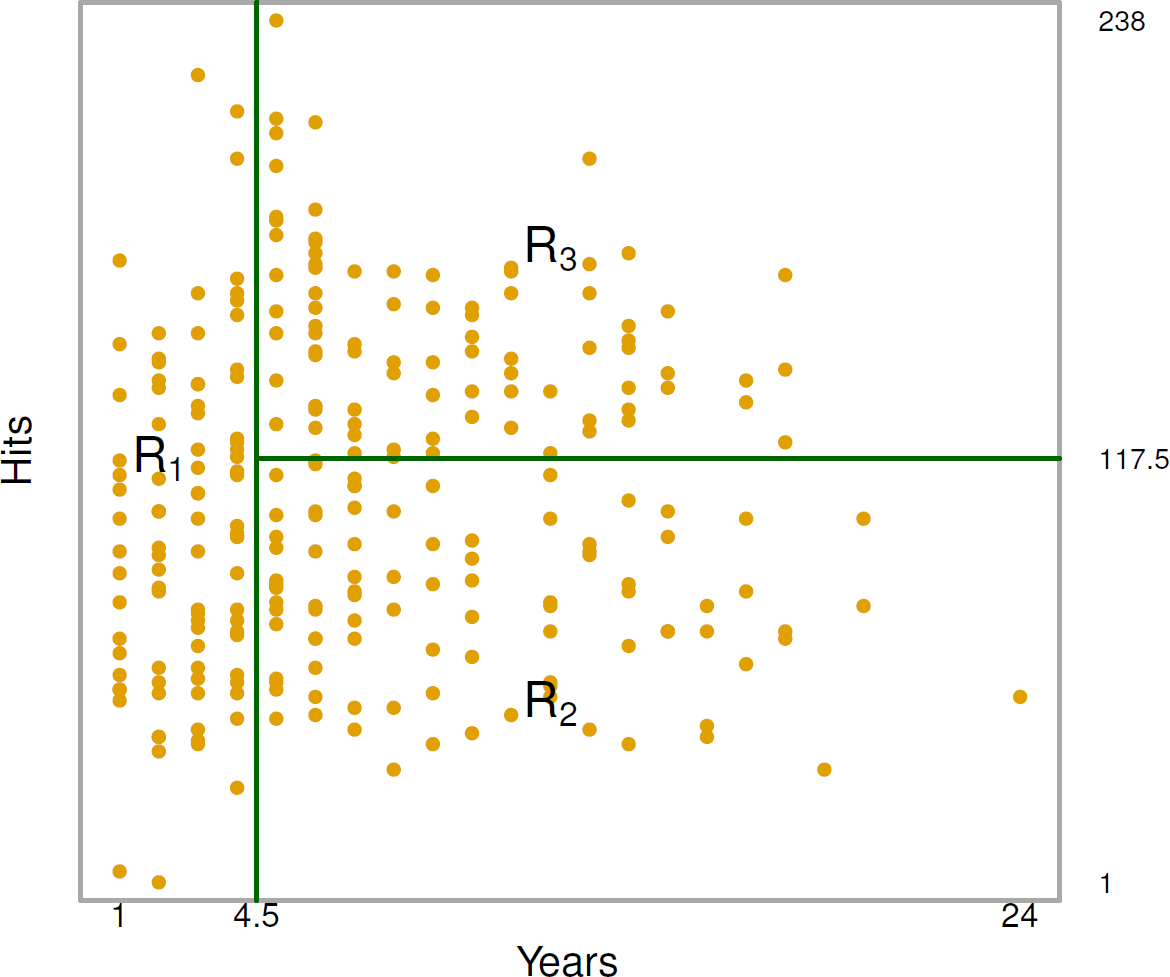
𝑋𝑋|Years ≥ 4.5, Hits < 117.5

𝑋𝑋|Years ≥ 4.5, Hits ≥ 117.5

* + 𝑅𝑅1 =

, 𝑅𝑅2 =

, and 𝑅𝑅3 =



©copyright CIEL 2015 **10**

# Terminology for Trees

* In keeping with the tree analogy, the regions 𝑅𝑅1, 𝑅𝑅2, and 𝑅𝑅3 are known as terminal nodes
* Decision trees are typically drawn upside down, in the sense that the leaves are at the bottom of the tree.
* The points along the tree where the predictor space is split are referred to as internal nodes
* In the hitters tree, the two internal nodes are indicated by the text Years <4.5 and Years <117.5

©copyright CIEL 2015 **11**

# Baseball Salary Dataset: Interpretation of Results

* Years is the most important factor in determining Salary, and players with less experience earn lower salaries than more experienced players
* Given that a player is less experienced, the number of Hits that he made in the previous year seems to play little role in his Salary
* But among players who have been in the major leagues for five or more years, the number of Hits made in the previous year does affect Salary, and players who made more Hits last year tend to have higher salaries
* Surely an over-simplification, but compared to a regression model, it is easy to display, interpret and explain

©copyright CIEL 2015 **12**

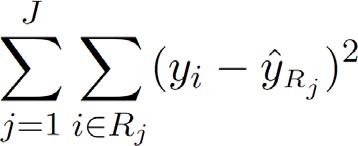
# Details of Tree-building Process

1. We divide the predictor space, i.e., the set of possible values for 𝑋𝑋1, 𝑋𝑋2, ⋯ , 𝑋𝑋𝑝𝑝, into 𝐽𝐽 distinct and non-overlapping regions, 𝑅𝑅1, 𝑅𝑅2, ⋯ , 𝑅𝑅𝐽𝐽
2. For every observation that falls into the region 𝑅𝑅𝑗𝑗 , we make the same prediction, which is simply the mean of the response values for the training observations in 𝑅𝑅𝑗𝑗

©copyright CIEL 2015 **13**

# More Details of Tree-building Process

* In theory, the regions could have any shape
* However, we choose to divide the predictor space into high-dimensional rectangles, or boxes, for simplicity and for ease of interpretation of the resulting predictive model
* The goal is to find boxes 𝑅𝑅1, 𝑅𝑅2, ⋯ , 𝑅𝑅𝐽𝐽 that minimize the RSS, given by



* + 𝑦𝑦�𝑅𝑅 is the mean response for the training observations within the 𝑗𝑗th box

𝑗𝑗

©copyright CIEL 2015 **14**

# More Details of Tree-building Process [cont.]

* Unfortunately, it is computationally infeasible to consider every possible partition of the feature space into 𝐽𝐽 boxes
* For this reason, we take a top-down, greedy approach that is known as recursive binary splitting
* The approach is top-down because it begins at the top of the tree and then successively splits the predictor space
  + Each split is indicated via two new branches further down on the tree.
* It is greedy because at each step of the tree-building process, the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step

©copyright CIEL 2015 **15**

# More Details of Tree-building Process [cont.]

* We first select the predictor 𝑋𝑋𝑗𝑗 and the cutpoint 𝑠𝑠 such that splitting the predictor space into the

regions

𝑋𝑋|𝑋𝑋𝑗𝑗 < 𝑠𝑠

𝑋𝑋|𝑋𝑋𝑗𝑗 ≥ 𝑠𝑠

and

leads to the greatest possible reduction in RSS

* Next, we repeat the process, looking for the best predictor and best cutpoint in order to split the data further so as to minimize the RSS within each of the resulting regions
* However, this time, instead of splitting the entire predictor space, we split one of the two previously identified regions
  + We now have three regions
* Again, we look to split one of these three regions further, so as to minimize the RSS. The process continues until a stopping criterion is reached; for instance, we may continue until no region contains more than five observations

©copyright CIEL 2015 **16**

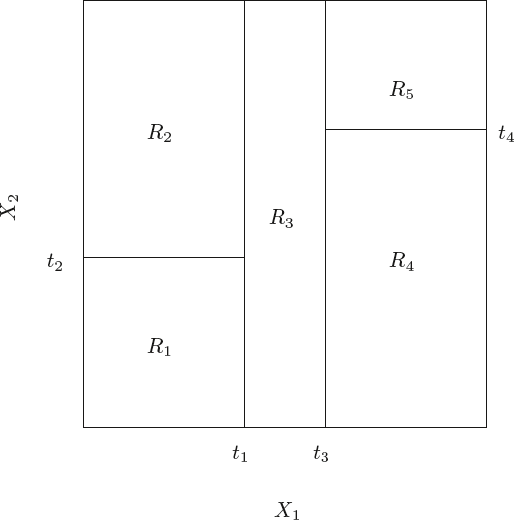
# Predictions

* We predict the response for a given test observation using the mean of the training observations in the region to which that test observation belongs
* A five-region example of this approach is shown in the next slide

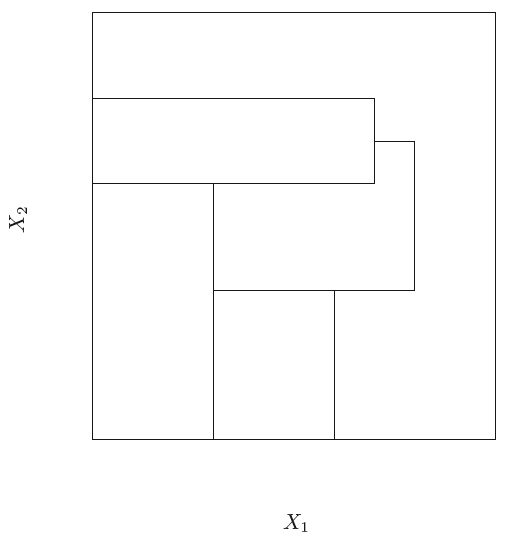
©copyright CIEL 2015 **17**

# Predictions [cont.]

* A partition of two-dimensional feature space that could not result from recursive binary splitting
* The output of recursive binary splitting on a two-dimensional example

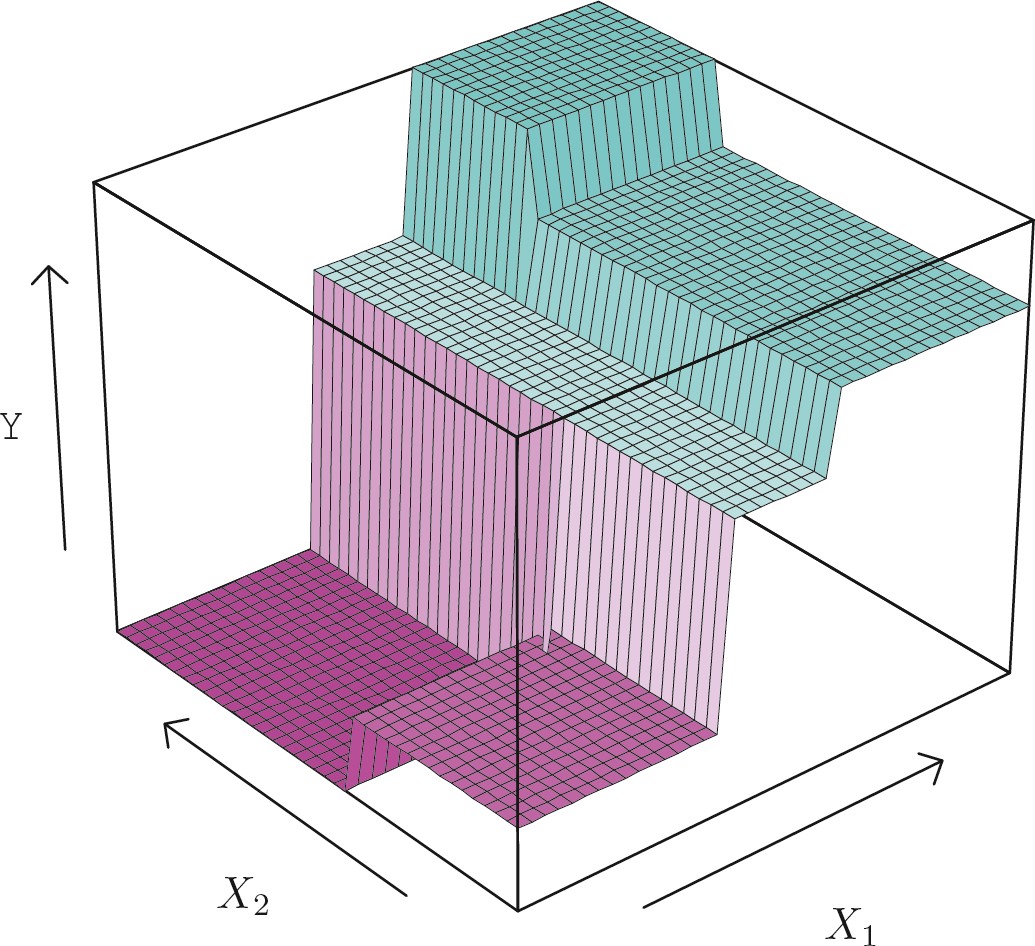


©copyright CIEL 2015 **18**

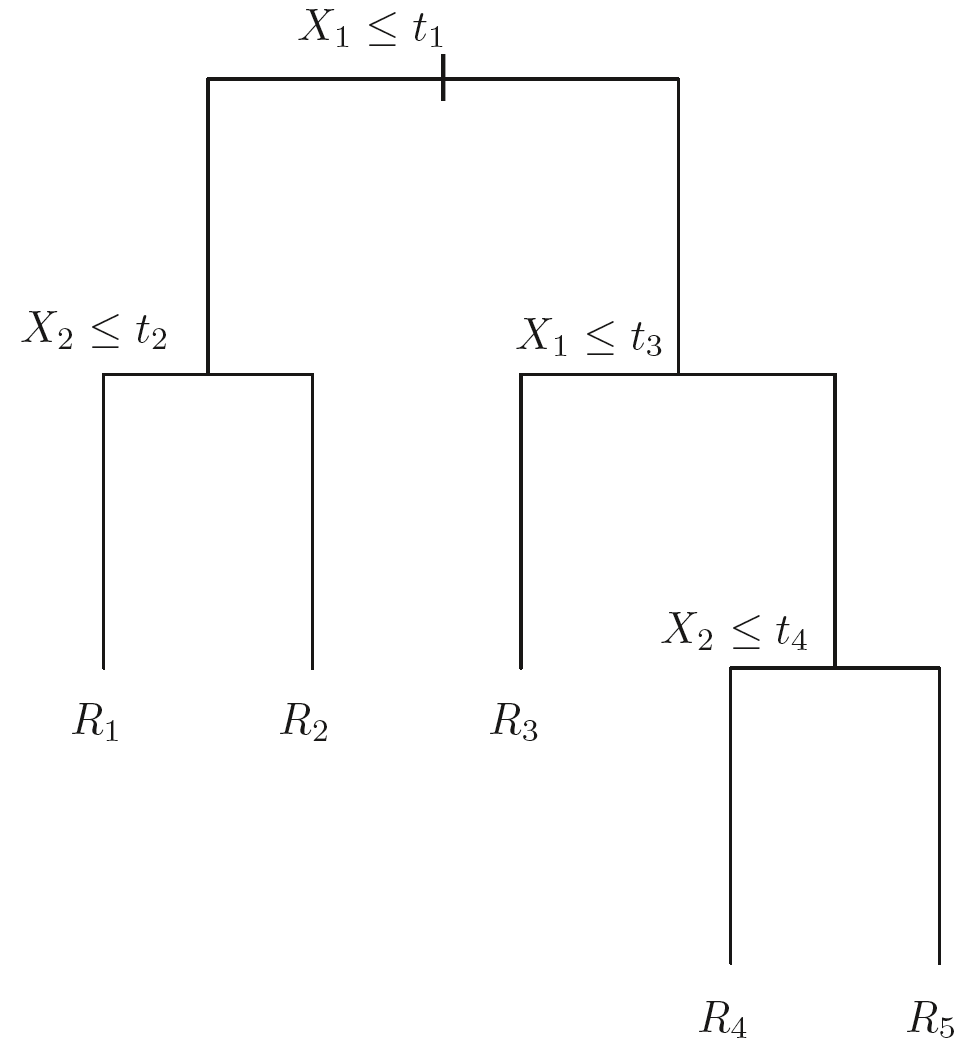


# Predictions [cont.]

* A tree corresponding to the partition in the right panel in the previous slide
* A perspective plot of the prediction surface corresponding to that tree



©copyright CIEL 2015 **19**



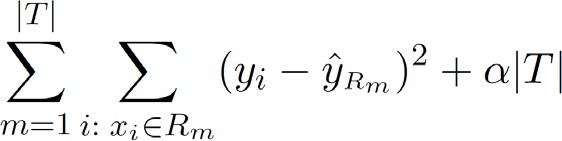
# Pruning A Tree

* The process described previously may produce good predictions on the training set, but is likely to overfit the data, leading to poor test set performance
* A smaller tree with fewer splits (that is, fewer regions 𝑅𝑅1, 𝑅𝑅2, ⋯ , 𝑅𝑅𝐽𝐽) might lead to lower variance and better interpretation at the cost of a little bias
* One possible alternative to the process described above is to grow the tree only so long as the decrease in the RSS due to each split exceeds some (high) threshold
* This strategy will result in smaller trees, but is too short-sighted
  + A seemingly worthless split early on in the tree might be followed by a very good split, that is, a split that leads to a large reduction in RSS later on

©copyright CIEL 2015 **20**

# Pruning A Tree [cont.]

* A better strategy is to grow a very large tree 𝑇𝑇0, and then prune it back in order to obtain a subtree
* Cost complexity pruning, also known as weakest link pruning, is used to do this
* We consider a sequence of trees indexed by a nonnegative tuning parameter 𝛼𝛼
* For each value of 𝛼𝛼 there corresponds a subtree 𝑇𝑇 ⊂ 𝑇𝑇0 such that

 is as small as possible

* + |𝑇𝑇| indicates the number of terminal nodes of the tree 𝑇𝑇, 𝑅𝑅𝑚𝑚 is the rectangle (i.e. the subset of predictor

space) corresponding to the 𝑚𝑚th terminal node, and 𝑦𝑦�𝑅𝑅 is the mean of the training observations in 𝑅𝑅𝑚𝑚

𝑚𝑚

©copyright CIEL 2015 **21**

# Choosing The Best Subtree

* The tuning parameter 𝛼𝛼 controls a trade-off between the subtree's complexity and its fit to the training data
* We select an optimal value

𝛼𝛼�

using cross-validation

* We then return to the full data set and obtain the subtree corresponding to

𝛼𝛼�

©copyright CIEL 2015 **22**

# Tree Algorithm

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations
2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of 𝛼𝛼
3. Use 𝐾𝐾-fold cross-validation to choose 𝛼𝛼. For each 𝑘𝑘 = 1, ⋯ , 𝐾𝐾:
   1. Repeat Steps 1 and 2 on the 𝐾𝐾−1th fraction of the training data, excluding the 𝑘𝑘th fold

𝐾𝐾

* 1. Evaluate the mean squared prediction error on the data in the left-out 𝑘𝑘 th fold, as a function of 𝛼𝛼

Average the results, and pick 𝛼𝛼 to minimize the average error

1. Return the subtree from Step 2 that corresponds to the chosen value of 𝛼𝛼

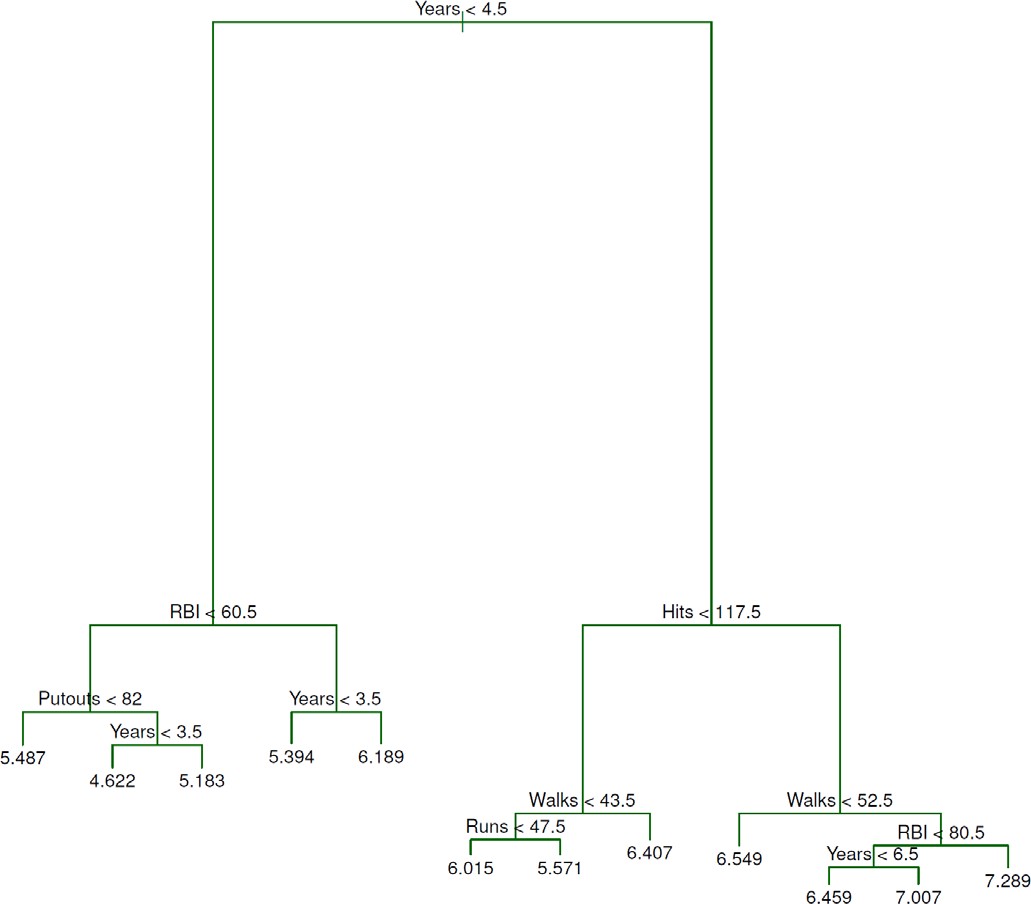
©copyright CIEL 2015 **23**

# Baseball Example Continued

* First, we randomly divided the data set in half, yielding 132 observations in the training set and 131 observations in the test set
* We then built a large regression tree on the training data and varied in 𝛼𝛼 in order to create subtrees with different numbers of terminal nodes
* Finally, we performed six-fold cross-validation in order to estimate the cross-validated MSE of the trees as a function of 𝛼𝛼

©copyright CIEL 2015 **24**

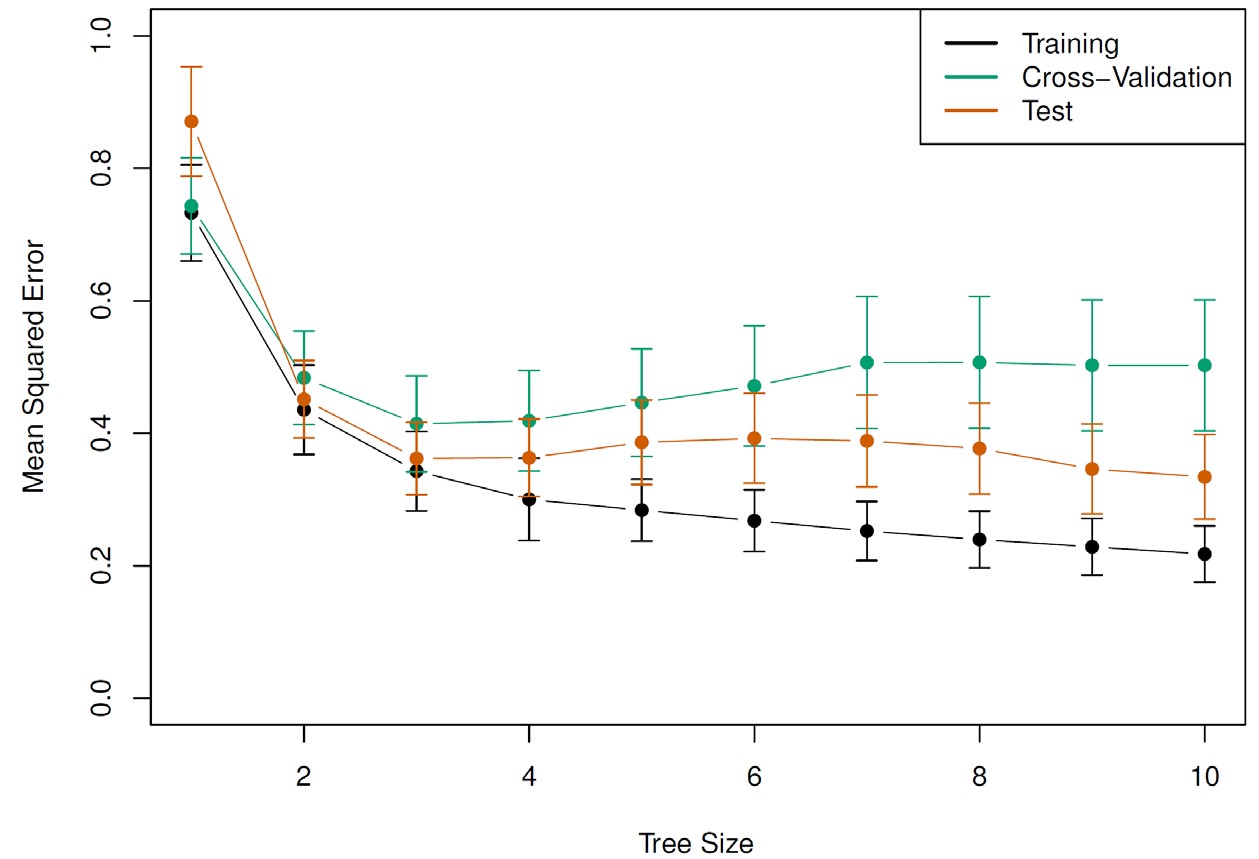
# Baseball Example Continued [cont.]



©copyright CIEL 2015 **25**

# Baseball Example Continued [cont.]

©copyright CIEL 2015 **26**



# Classification Trees

* Very similar to a regression tree, except that it is used to predict a qualitative response rather than a quantitative one
* For a classification tree, we predict that each observation belongs to the most commonly occurring class of training observations in the region to which it belongs

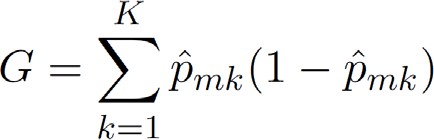
©copyright CIEL 2015 **27**

# Details of Classification Trees

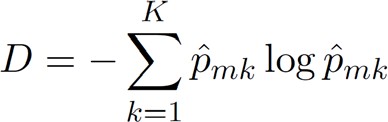
* Just as in the regression setting, we use recursive binary splitting to grow a classification tree
* In the classification setting, RSS cannot be used as a criterion for making the binary splits
* A natural alternative to RSS is the classification error rate
  + This is simply the fraction of the training observations in that region that do not belong to the most common class:
  + 𝑝𝑝̂𝑚𝑚𝑘𝑘 represents the proportion of training observations in the 𝑚𝑚th region that are from the 𝑘𝑘th class
* However classification error is not sufficiently sensitive for tree-growing, and in practice two other measures are preferable

©copyright CIEL 2015 **28**

# Gini Index and Deviance

* The Gini index is defined by
  + A measure of total variance across the 𝐾𝐾 classes
  + The Gini index takes on a small value if all of the

𝑝𝑝̂𝑚𝑚𝑘𝑘's are close to zero or one

* For this reason the Gini index is referred to as a measure of node purity
  + A small value indicates that a node contains predominantly observations from a single class
* An alternative to the Gini index is cross-entropy, given by
* It turns out that the Gini index and the cross-entropy are very similar numerically

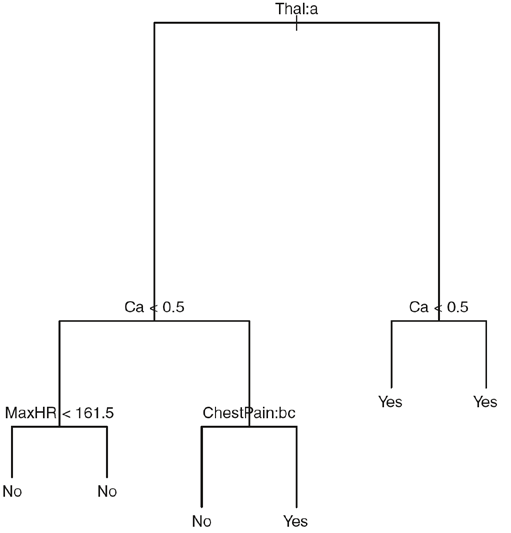
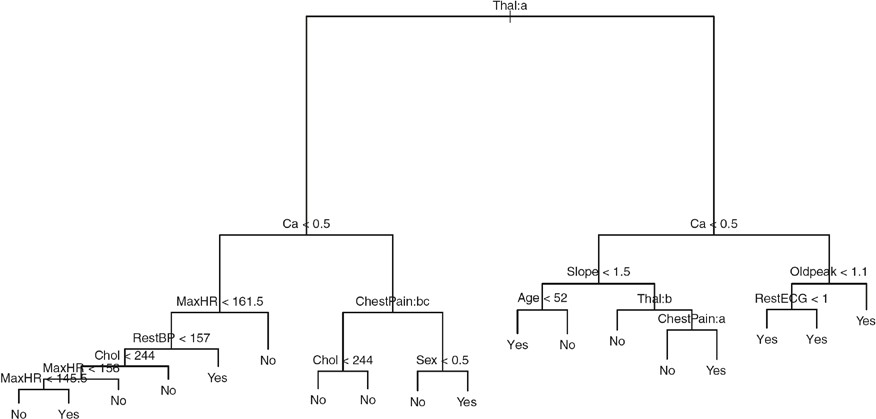
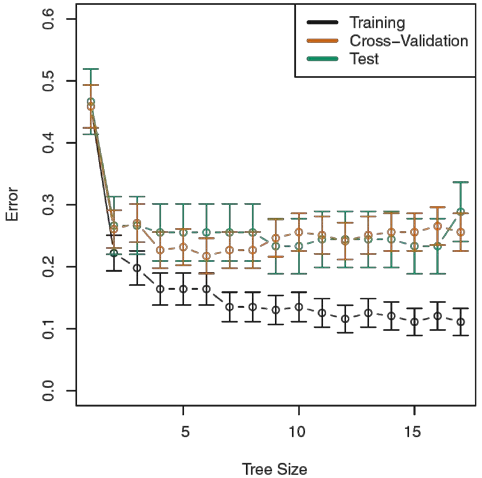
©copyright CIEL 2015 **29**

# Example: Heart Data

* These data contain a binary outcome HD for 303 patients who presented with chest pain
* An outcome value of Yes indicates the presence of heart disease based on an angiographic test, while No means no heart disease
* There are 13 predictors including Age, Sex, Chol (a cholesterol measurement), and other heart and lung function measurements
* Cross-validation yields a tree with six terminal nodes. See next figure

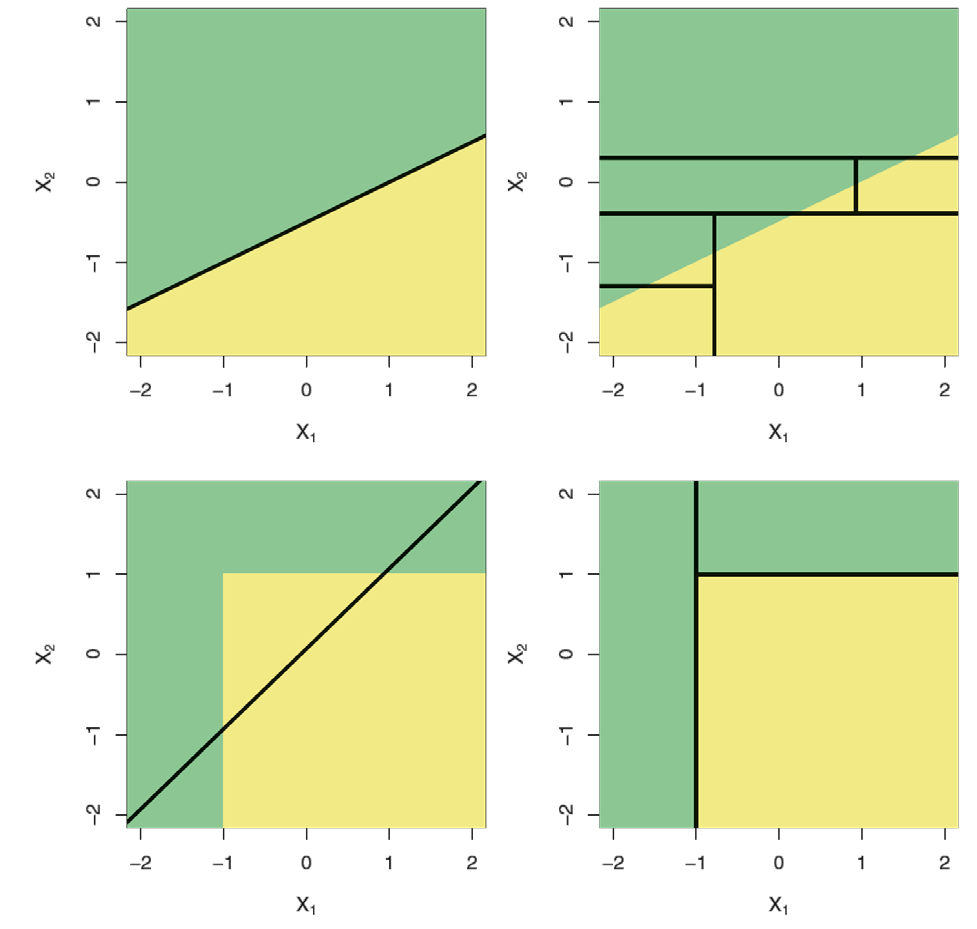
©copyright CIEL 2015 **30**

# Example: Heart Data [cont.]



©copyright CIEL 2015 **31**

# Trees Versus Linear Models

* Top Row: True linear boundary
* Bottom row: true non-linear boundary
* Left column: linear model
* Right column: tree-based model

©copyright CIEL 2015 **32**

# Advantages and Disadvantages of Trees

▲ 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 chapters

▲ 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

▲ Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches seen in this book

* However, by aggregating many decision trees, the predictive performance of trees can be substantially improved. We introduce these concepts next.

©copyright CIEL 2015 **33**

# Bagging

* Bootstrap aggregation, or bagging, is a general-purpose procedure for reducing the variance of a statistical learning method
  + We introduce it here because it is particularly useful and frequently used in the context of decision trees
* Recall that given a set of 𝑛𝑛 independent observations 𝑍𝑍1, ⋯ , 𝑍𝑍𝑛𝑛, each with variance 𝜎𝜎2, the

variance of the mean 𝑍𝑍 of the observations is given by 𝜎𝜎2/𝑛𝑛

* In other words, averaging a set of observations reduces variance
  + Of course, this is not practical because we generally do not have access to multiple training sets

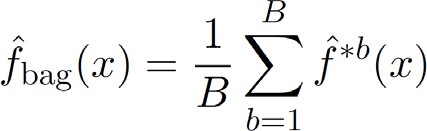
©copyright CIEL 2015 **34**

# Bagging [cont.]

* Instead, we can bootstrap, by taking repeated samples from the (single) training data set
* In this approach we generate 𝐵𝐵 different bootstrapped training data sets. We then train our

method on the 𝑏𝑏th bootstrapped training set in order to get 𝑓𝑓̂∗𝑏𝑏(𝑥𝑥), the prediction at a point 𝑥𝑥

* We then average all the predictions to obtain



* This is called bagging

©copyright CIEL 2015 **35**

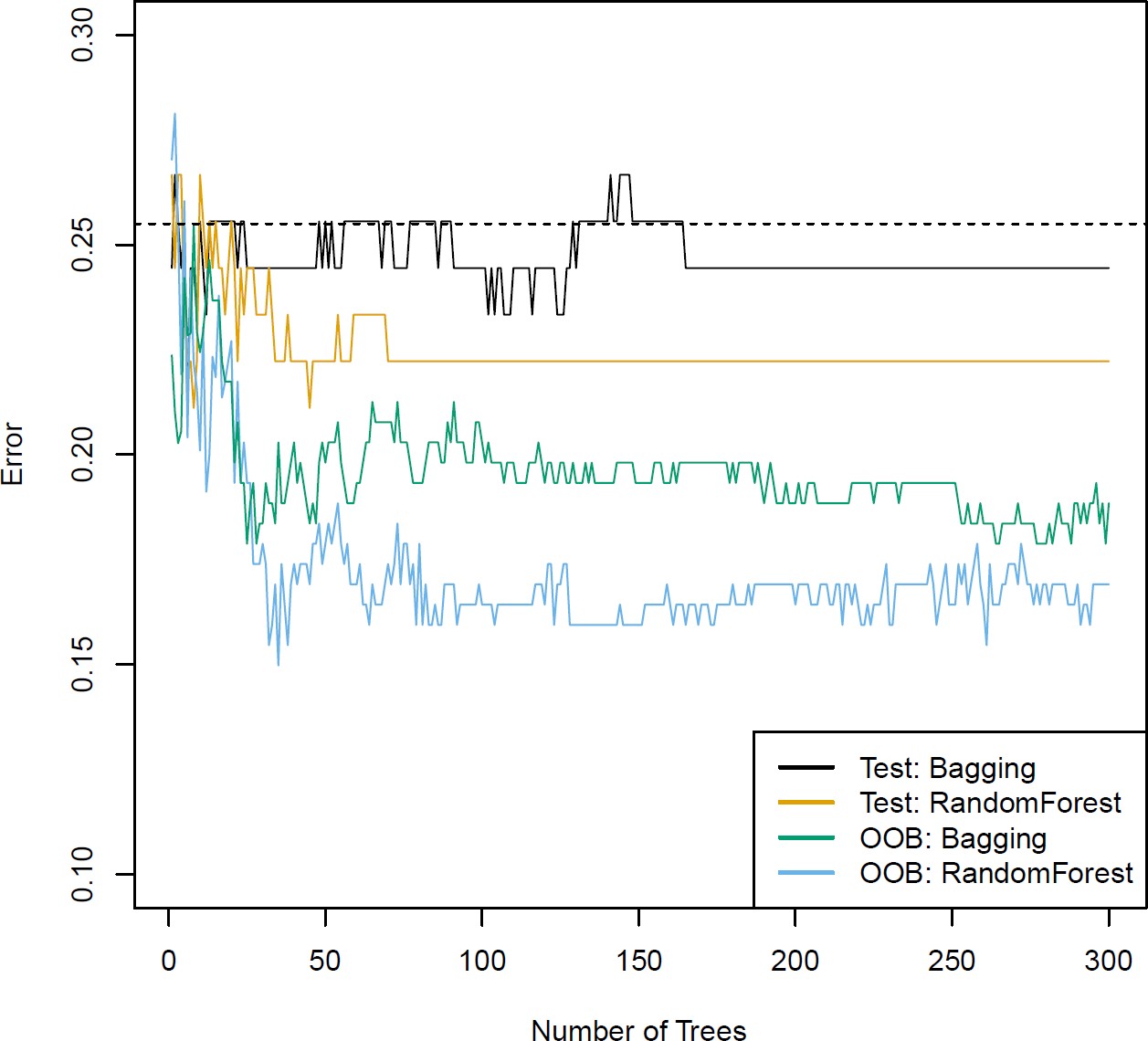
# Bagging Classification Trees

* The previous prescription applied to regression trees
* For classification trees
  + For each test observation, we record the class predicted by each of the 𝐵𝐵 trees, and take a majority vote

o Overall prediction is the most commonly occurring class among the 𝐵𝐵 predictions

©copyright CIEL 2015 **36**

# Bagging Heart Data



©copyright CIEL 2015 **37**

# Bagging Heart Data [cont.]

* Bagging and random forest results for the Heart data
  + The test error (black and orange) is shown as a function of 𝐵𝐵, the number of bootstrapped training sets used
  + Random forests were applied with 𝑚𝑚 =

𝑝𝑝

* + The dashed line indicates the test error resulting from a single classification tree
  + The green and blue traces show the OOB (out-of-bag) error, which in this case is considerably lower

©copyright CIEL 2015 **38**

# Out-of-Bag Error Estimation

* It turns out that there is a very straightforward way to estimate the test error of a bagged model
* Recall that the key to bagging is that trees are repeatedly fit to bootstrapped subsets of the observations
  + One can show that on average, each bagged tree makes use of around two-thirds of the observations
* The remaining one-third of the observations not used to fit a given bagged tree are referred to as the out-of-bag (OOB) observations
* We can predict the response for the 𝑖𝑖th observation using each of the trees in which that observation was OOB
  + This will yield around 𝐵𝐵/3 predictions for the 𝑖𝑖th observation, which we average
* This estimate is essentially the LOO cross-validation error for bagging, if *B* is large.

©copyright CIEL 2015 **39**

# Random Forests

* Random forests provide an improvement over bagged trees by way of a small tweak that decorrelates the trees
  + This reduces the variance when we average the trees
* As in bagging, we build a number of decision trees on bootstrapped training samples
* But when building these decision trees, each time a split in a tree is considered, a random selection of 𝑚𝑚 predictors is chosen as split candidates from the full set of 𝑝𝑝 predictors
  + The split is allowed to use only one of those 𝑚𝑚 predictors
* A fresh selection of 𝑚𝑚 predictors is taken at each split, and typically we choose 𝑚𝑚 ≈ 𝑝𝑝, i.e., the number of predictors considered at each split is approximately equal to the square root of the total number of predictors (4 out of the 13 for the Heart data)

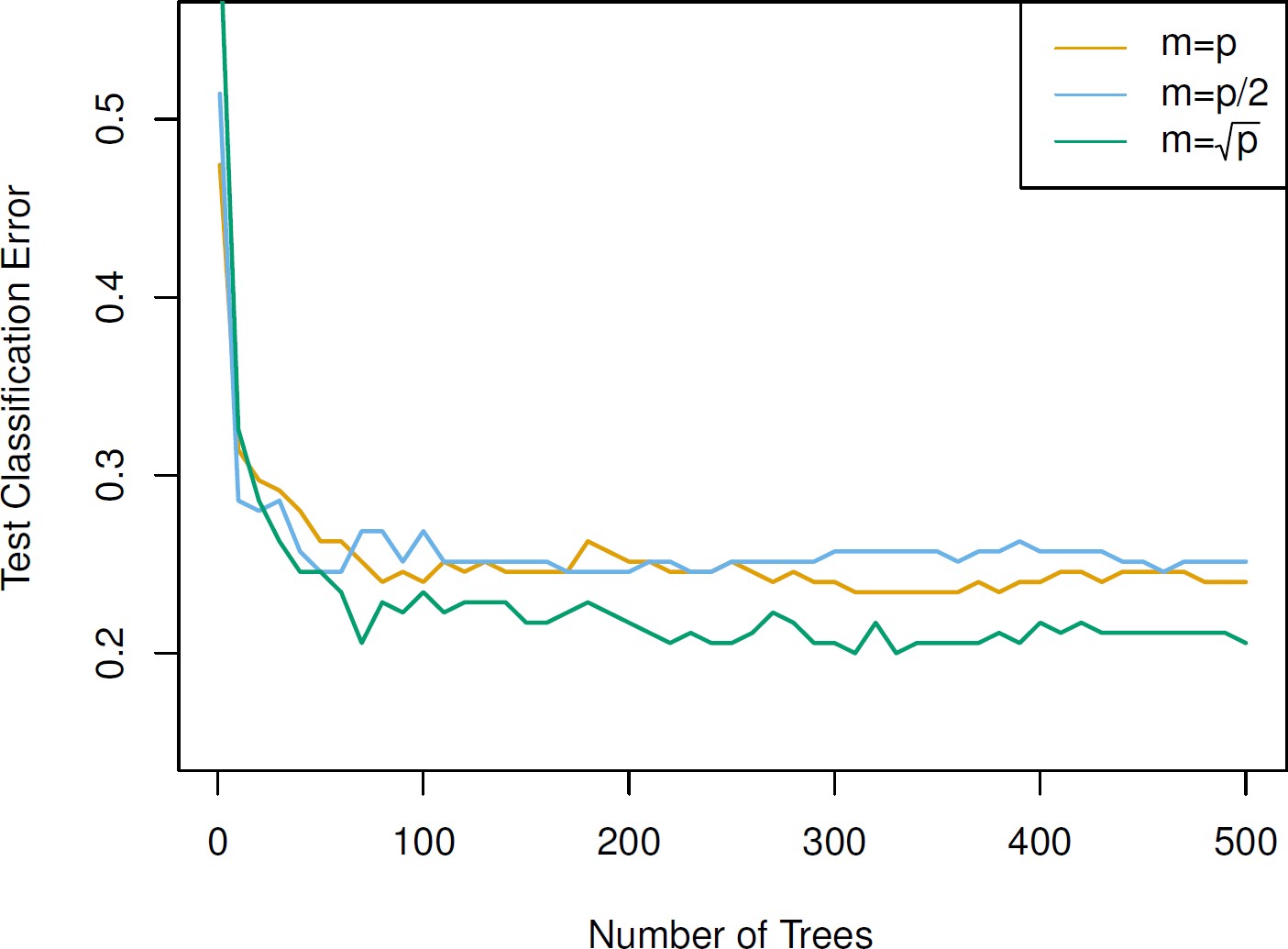
©copyright CIEL 2015 **40**

# Example: Gene Expression Data

* We applied random forests to a high-dimensional biological data set consisting of expression measurements of 4,718 genes measured on tissue samples from 349 patients
* There are around 20,000 genes in humans, and individual genes have different levels of activity, or expression, in particular cells, tissues, and biological conditions
* Each of the patient samples has a qualitative label with 15 different levels: either normal or one of 14 different types of cancer
* We use random forests to predict cancer type based on the 500 genes that have the largest variance in the training set
* We randomly divided the observations into a training and a test set, and applied random forests to the training set for three different values of the number of splitting variables m.

©copyright CIEL 2015 **41**

# Results: Gene Expression Data



©copyright CIEL 2015 **42**

# Results: Gene Expression Data [cont.]

* Results from random forests for the fifteen-class gene expression data set with 𝑝𝑝 = 500

predictors

* The test error is displayed as a function of the number of trees. Each colored line corresponds to a different value of 𝑚𝑚, the number of predictors available for splitting at each interior tree node
* Random forests (𝑚𝑚 < 𝑝𝑝) lead to a slight improvement over bagging (𝑚𝑚 = 𝑝𝑝)
  + A single classification tree has an error rate of 45.7%.

©copyright CIEL 2015 **43**

# Boosting

* Like bagging, boosting is a general approach that can be applied to many statistical learning methods for regression or classification
  + We only discuss boosting for decision trees
* Recall that bagging involves creating multiple copies of the original training data set using the bootstrap, fitting a separate decision tree to each copy, and then combining all of the trees in order to create a single predictive model
* Notably, each tree is built on a bootstrap data set, independent of the other trees.
* Boosting works in a similar way, except that the trees are grown sequentially
  + Each tree is grown using information from previously grown trees

©copyright CIEL 2015 **44**

# Boosting Algorithm for Regression Trees

1. Set

𝑥𝑥

𝑓𝑓̂

= 0 and 𝑟𝑟𝑖𝑖 = 𝑦𝑦𝑖𝑖 for all 𝑖𝑖 in the training set

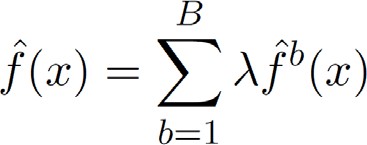
2. For 𝑏𝑏 = 1,2, ⋯ , 𝐵𝐵, repeat:

* 1. Fit a tree 𝑓𝑓̂𝑏𝑏 with 𝑑𝑑 splits (𝑑𝑑 + 1 terminal nodes) to the training data (𝑋𝑋, 𝑟𝑟)
  2. Update

𝑓𝑓̂

by adding in a shrunken version of the new tree:

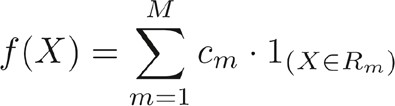
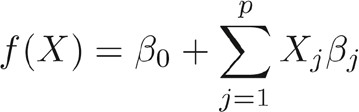
* 1. Update the residuals,

3. Output the boosted model



©copyright CIEL 2015 **45**

* Linear regression
* Regression tree



# What is the Idea behind This Procedure?

* Unlike fitting a single large decision tree to the data, which amounts to fitting the data hard and potentially overfitting, the boosting approach instead learns slowly
* Given the current model, we fit a decision tree to the residuals from the model. We then add this new decision tree into the fitted function in order to update the residuals
* Each of these trees can be rather small, with just a few terminal nodes, determined by the parameter 𝑑𝑑 in the algorithm
* By fitting small trees to the residuals, we slowly improve well

𝑓𝑓̂

in areas where it does not perform

* + The shrinkage parameter 𝜆𝜆 slows the process down even further, allowing more and different shaped trees to attack the residuals

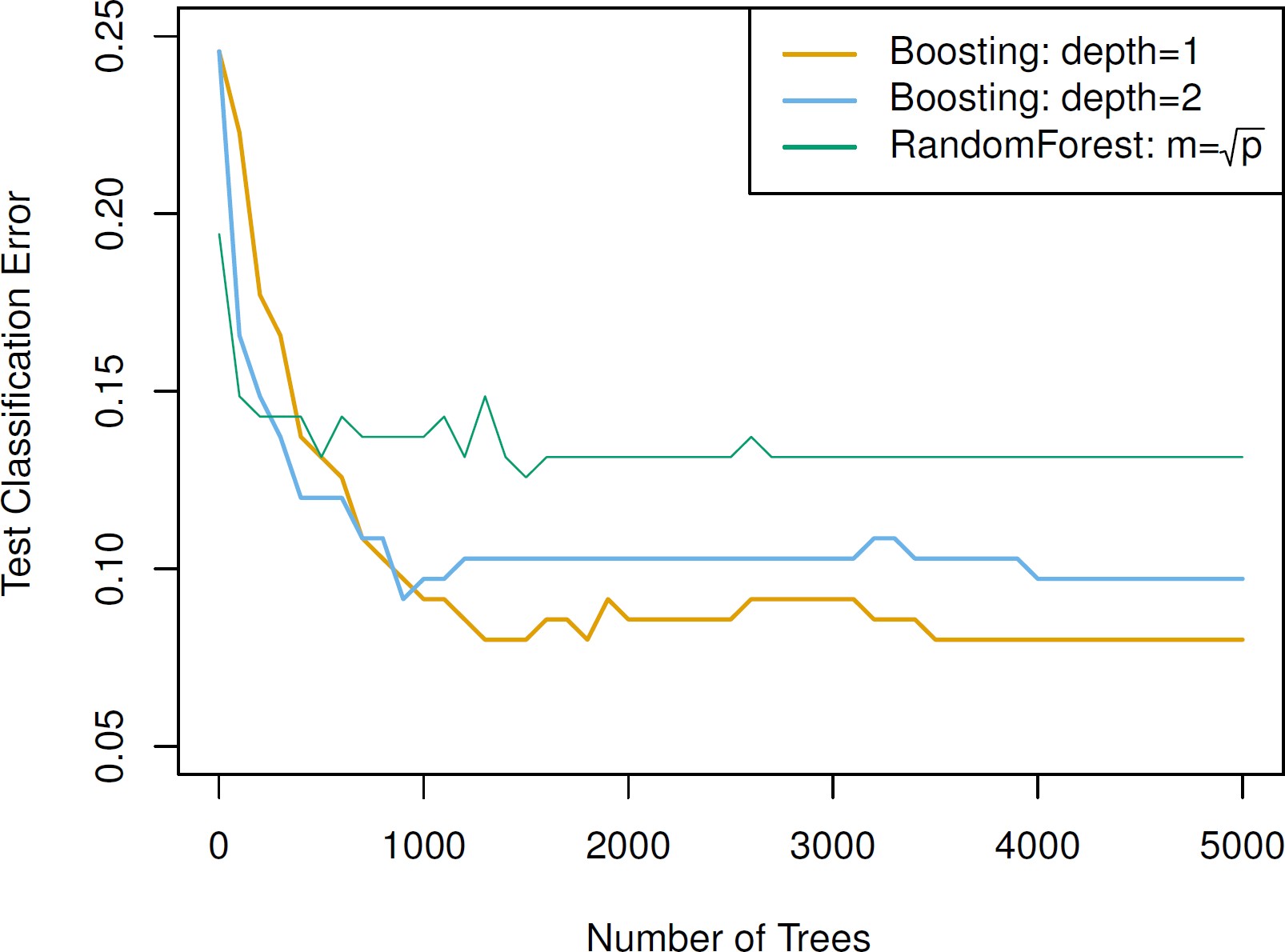
©copyright CIEL 2015 **46**

# Boosting for Classification

* Boosting for classification is similar in spirit to boosting for regression, but is a bit more complex
* We will not go into detail here, nor do we in the textbook
  + But, we will practice it through Python

©copyright CIEL 2015 **47**

# Gene Expression Data Continued



©copyright CIEL 2015 **48**

# Gene Expression Data Continued [cont.]

* Results from performing boosting and random forests on the fifteen-class gene expression data set in order to predict cancer versus normal
* The test error is displayed as a function of the number of trees
  + For the two boosted models, 𝜆𝜆 = 0.01.
  + Depth-1 trees slightly outperform depth-2 trees
  + Both outperform the random forest, although the standard errors are around 0.02, making none of these differences significant
* The test error rate for a single tree is 24%.

©copyright CIEL 2015 **49**

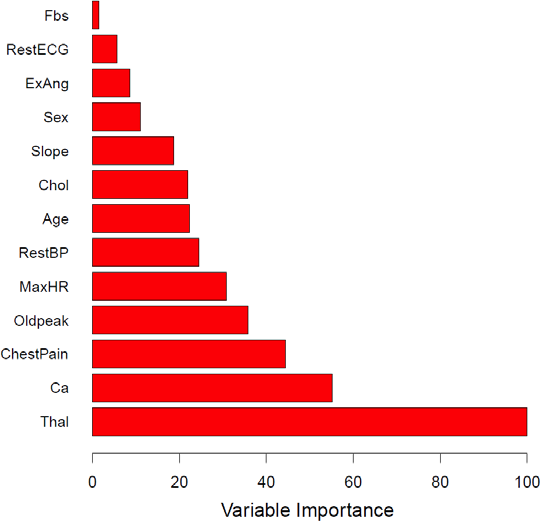
# Tuning Parameters for Boosting

* The number of trees 𝐵𝐵
  + Unlike bagging and random forests, boosting can overfit if 𝐵𝐵 is too large, although this overfitting tends to occur slowly if at all
  + We use cross-validation to select 𝐵𝐵
* The shrinkage parameter 𝜆𝜆, a small positive number
  + This controls the rate at which boosting learns
  + Typical values are 0.01 or 0.001, and the right choice can depend on the problem
  + Very small 𝜆𝜆 can require using a very large value of 𝐵𝐵 in order to achieve good performance
* The number of splits 𝑑𝑑 in each tree
  + 𝑑𝑑 controls the complexity of the boosted ensemble
  + Often 𝑑𝑑 = 1 works well, in which case each tree is a stump, consisting of a single split and resulting in an additive model
  + More generally 𝑑𝑑 is the interaction depth, and controls the interaction order of the boosted model, since 𝑑𝑑

splits can involve at most 𝑑𝑑 variables

©copyright CIEL 2015 **50**

# Variable Importance Measure

* For bagged/RF regression trees, we record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all 𝐵𝐵 trees.
  + A large value indicates an important predictor
* Similarly, for bagged/RF classification trees, we add up the total amount that the Gini index is decreased by splits over a given predictor, averaged over all 𝐵𝐵 trees
* Variable importance plot for Heart data

©copyright CIEL 2015 **51**



**Python Lab**

* Tree-based methods

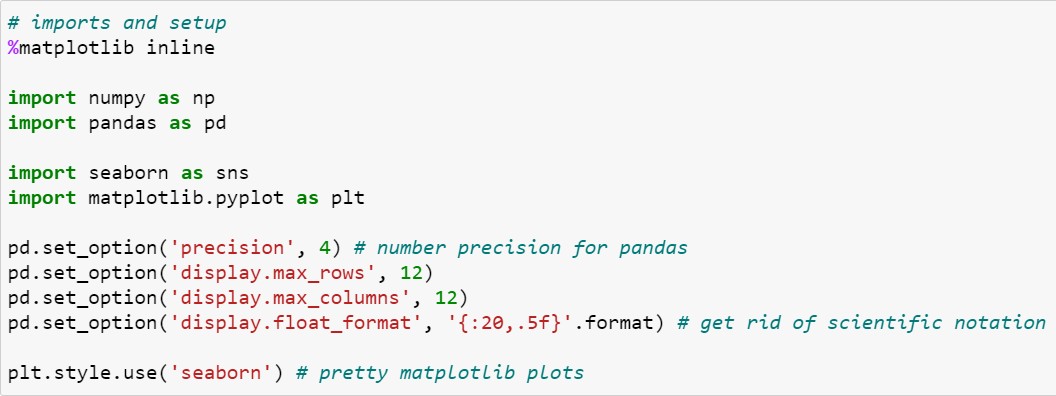
## Python lab

* Summary & Next class

©copyright CIEL 2015 https://ciel.seoultech.ac.kr/

# 8.3 Lab: Tree-Based Methods

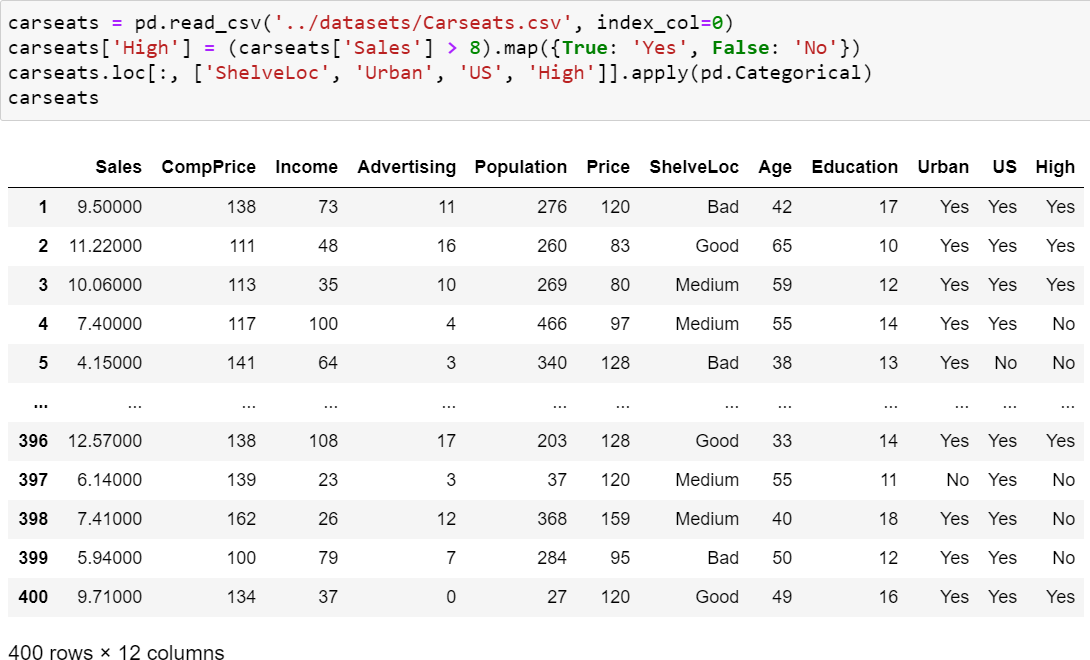
* + Using Python Libraries
    - Import the libraries that are often used for data analysis



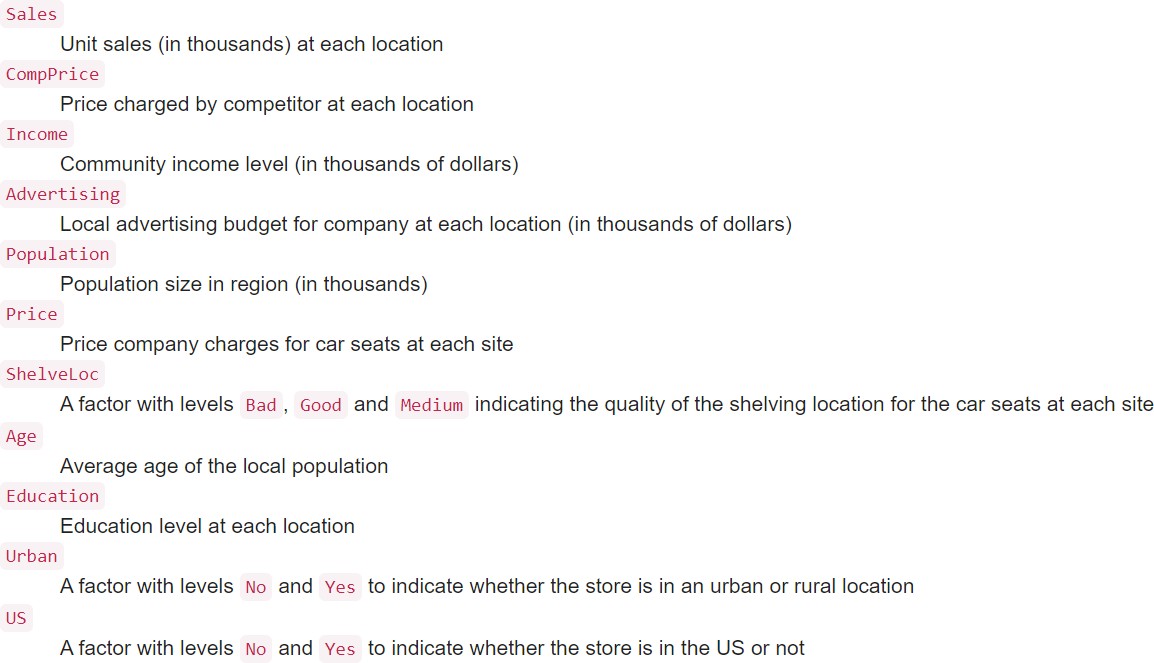
©copyright CIEL 2015 **53**

# 8.3.1 Fitting Classification Trees

* + Load data: Carseats data set
    - For analyzing Carsets data



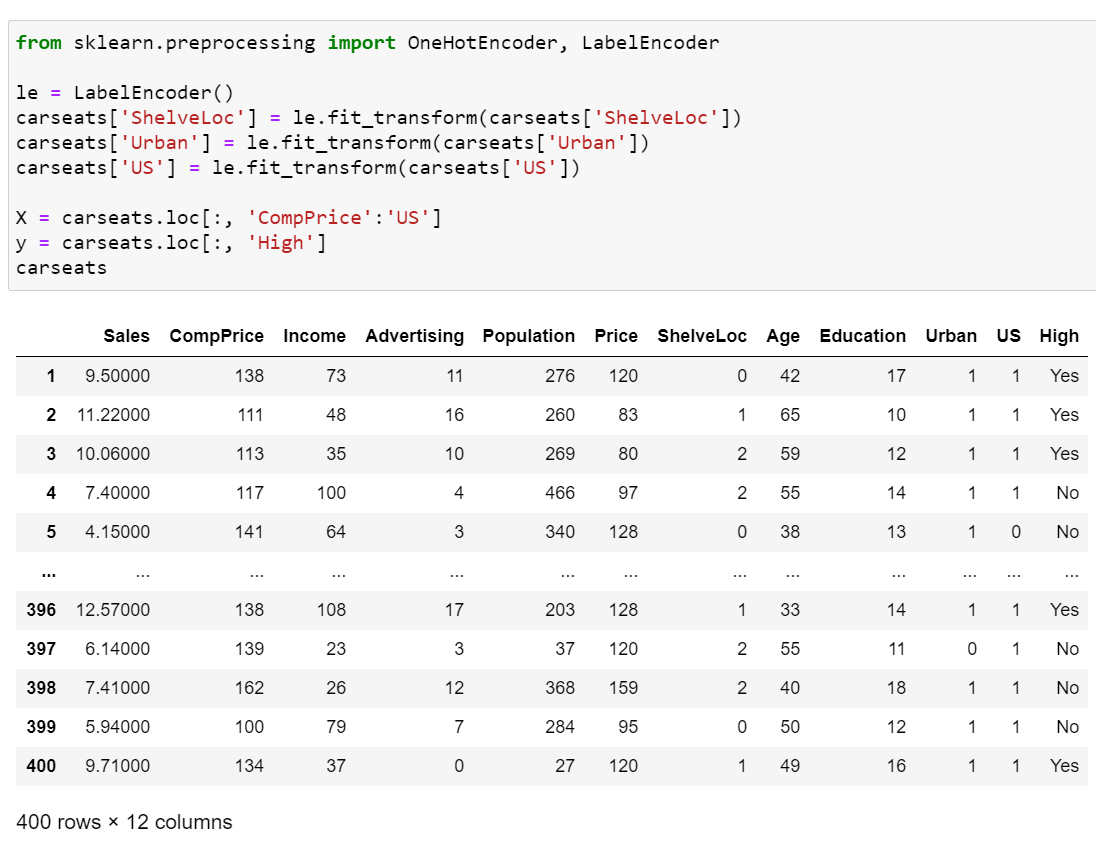
Adding new column



©copyright CIEL 2015 **54**

# 8.3.1 Fitting Classification Trees

* + Encoding for converting each text category to numbers
    - Label encoding

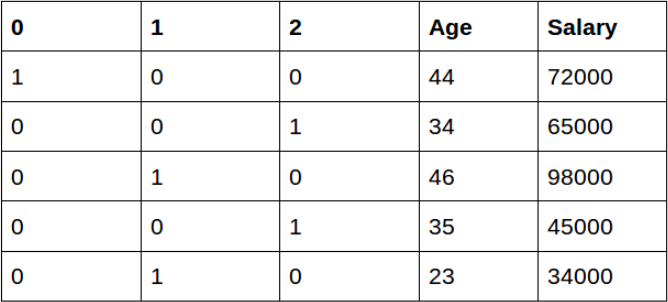


Fit label encoder and return encoded labels

* + - * Each label is assigned a unique integer based on alphabetical ordering



* + - One-hot encoding
      * Creating additional features based on the number of unique values in the categorical feature

**55**

©copyright CIEL 2015

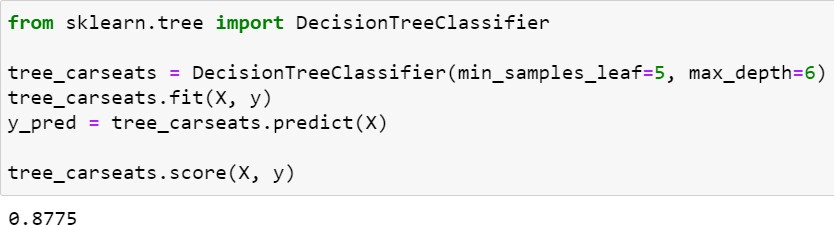
# 8.3.1 Fitting Classification Trees

* + Tree classifier
    - Sklearn.tree.DecisionTreeClassifier

Return the mean accuracy on the given test data and labels

Decision tree classifier

* min\_samples\_leaf: minimum number of samples required to be at a leaf node



* max\_depth: maximum depth of the tree If

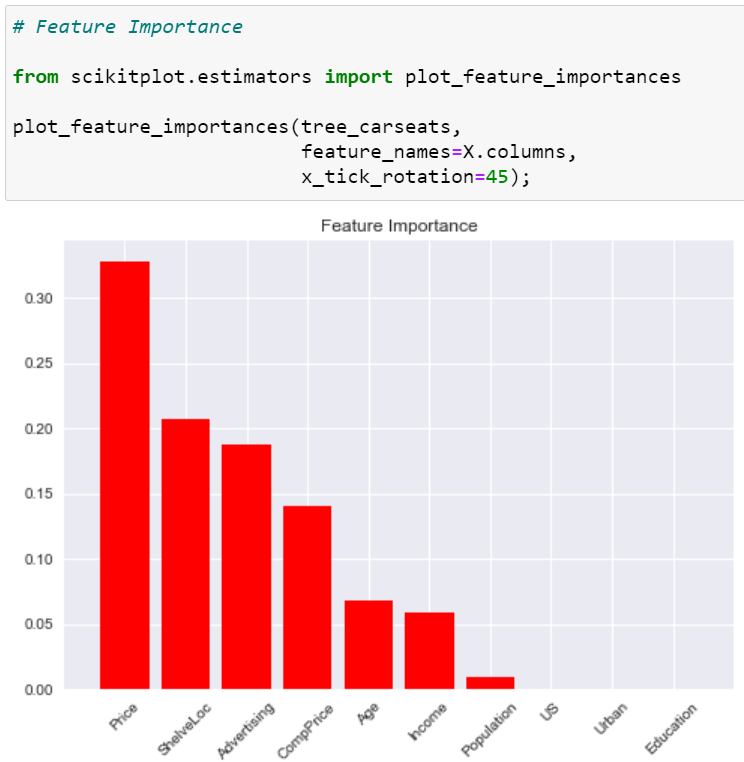
None, then nodes are expanded until all leaves are pure or until all leaves contain less than min\_samples\_split samples

©copyright CIEL 2015 **56**

# 8.3.1 Fitting Classification Trees

* + Feature importance for variable importance measures
    - Use of Scikit-plot: visualizations that frequently pop up in machine learning

scikitplot.estimators.plot\_feature\_importances



: generates a plot of a classifier’s feature importances

* + - * clf: classifier instance that has a feature\_importances\_attribute
      * Feature\_names: determines the feature names used to plot the feature importances
      * x\_tick\_rotation: rotates x-axis tick labels by the specified angle

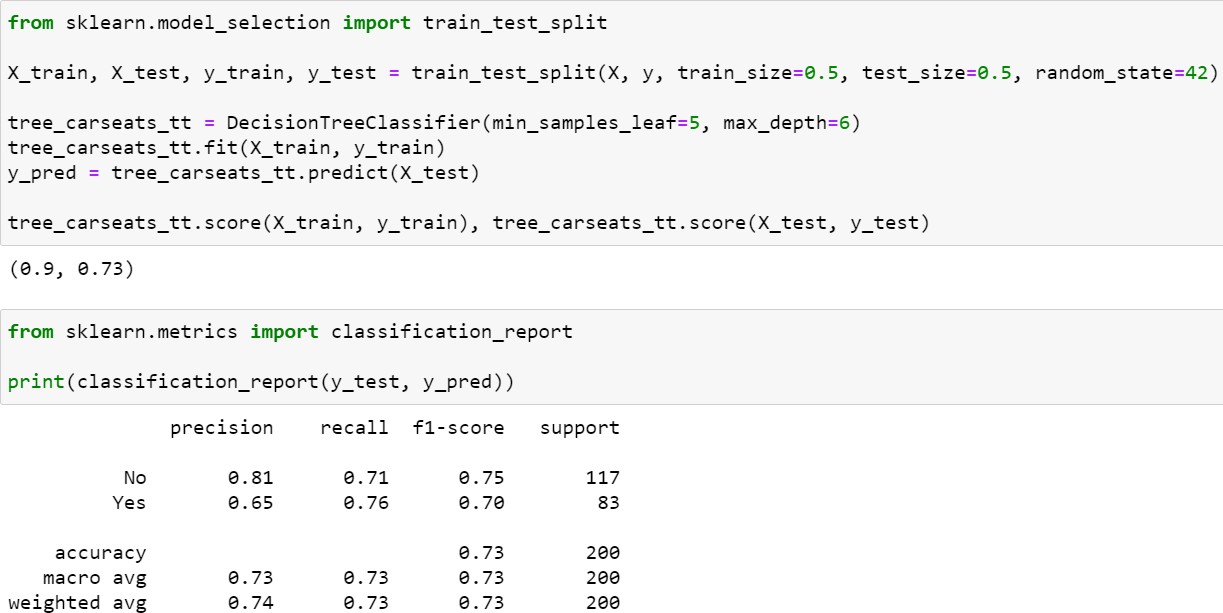
**57**

©copyright CIEL 2015

# 8.3.1 Fitting Classification Trees

* + Performances of tree classifier

Build a text report showing the main classification metrics

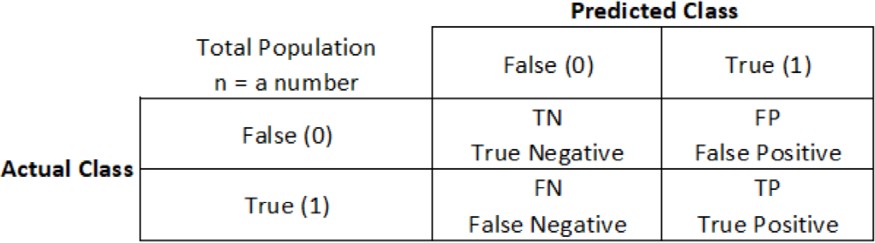


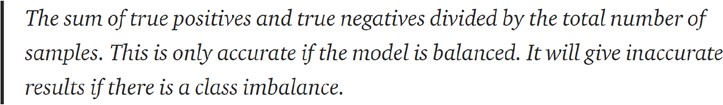
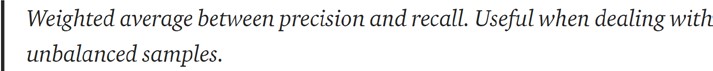
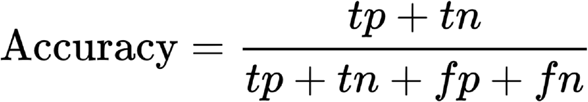
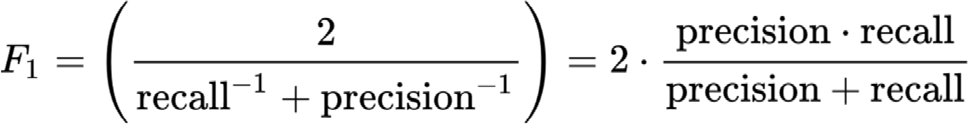
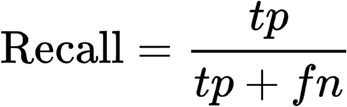
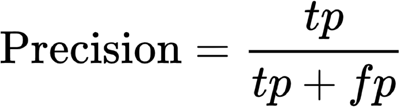
* + - macro avg: unweighted mean
    - weighted avg: average weighted by support

**58**

©copyright CIEL 2015

# Fitting Classification Trees

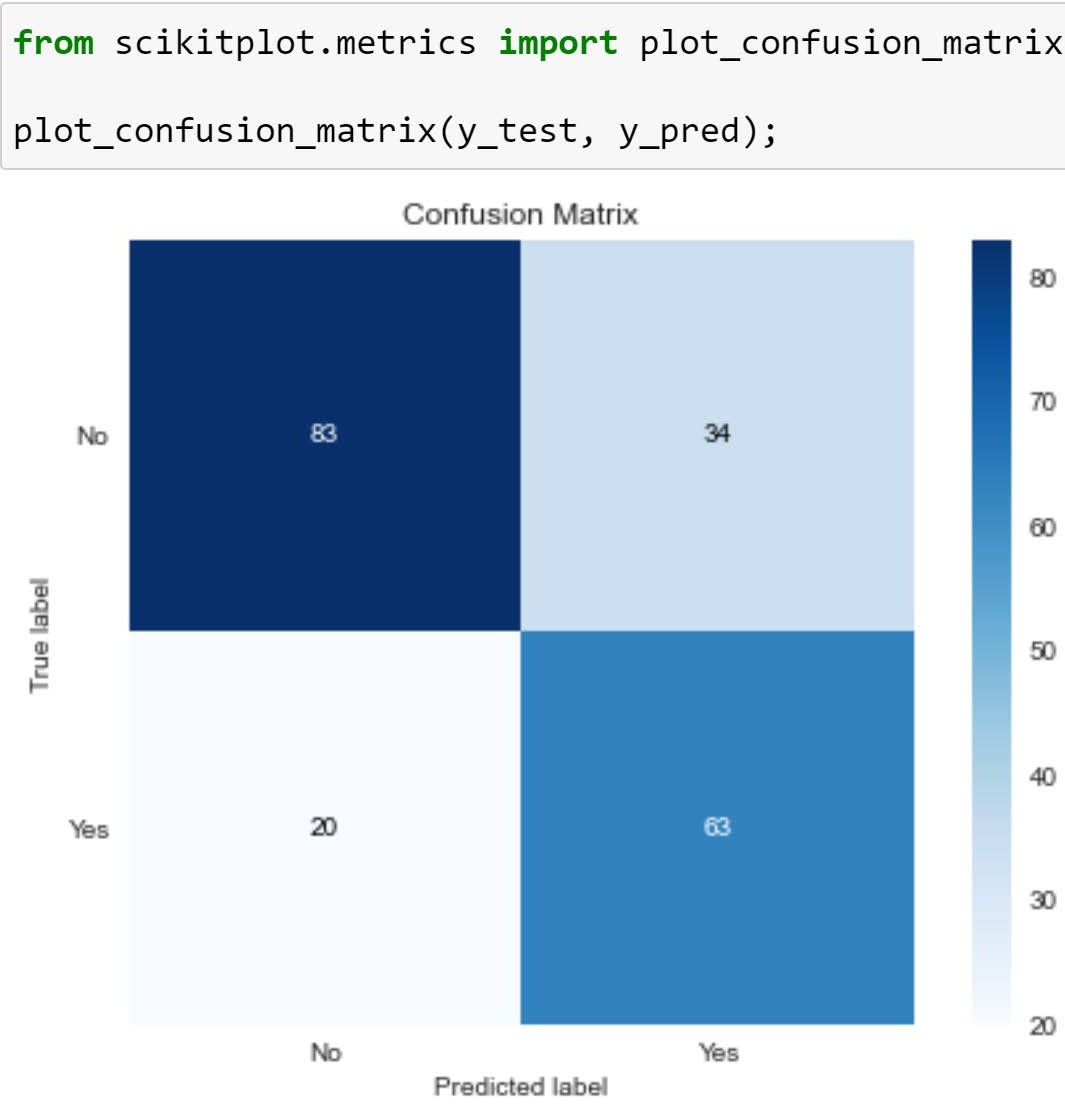
* + - * Confusion matrix
      * Main classification metrics



©copyright CIEL 2015 **59**

# 8.3.1 Fitting Classification Trees

* + Plotting confusion matrix



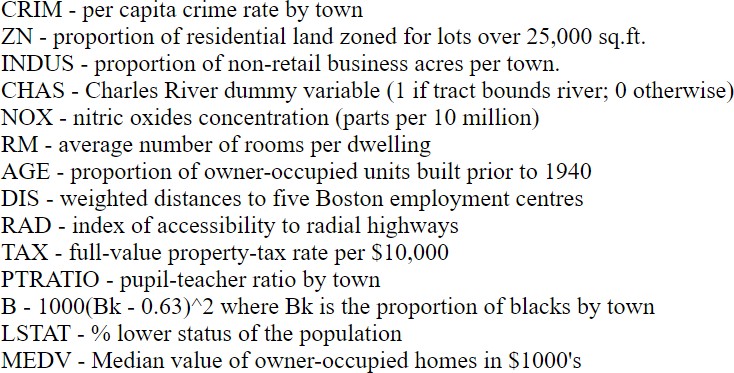
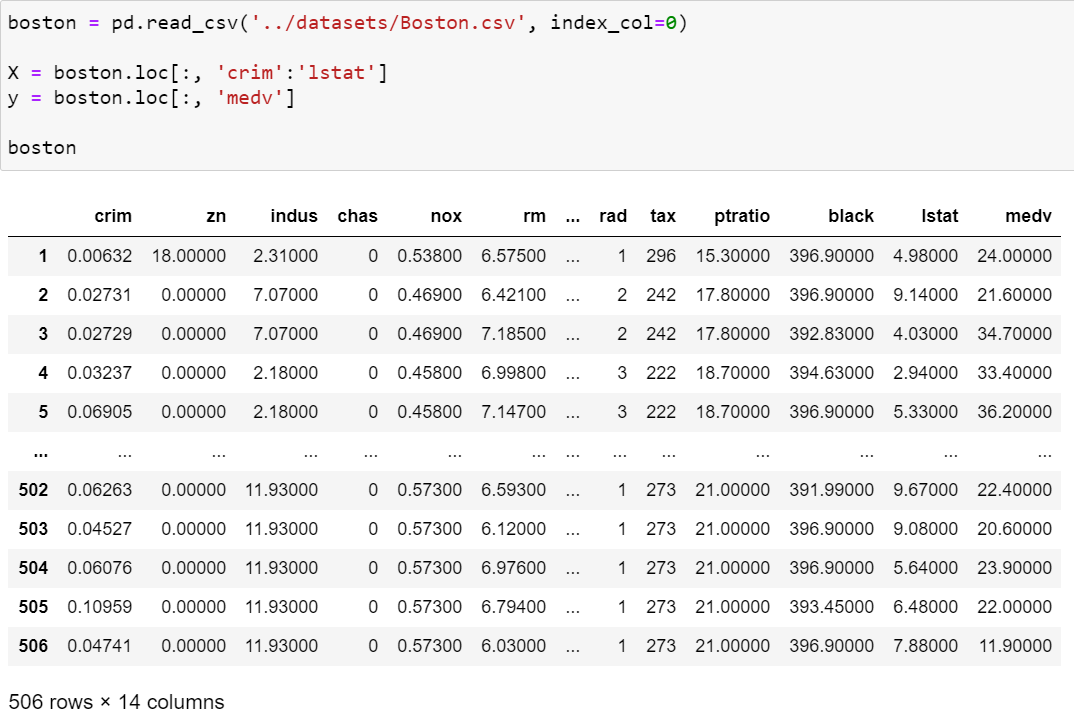
Plot Confusion Matrix

**60**

©copyright CIEL 2015

# 8.3.2 Fitting Regression Trees

* + Load data: Boston data set
    - For analyzing Boston data

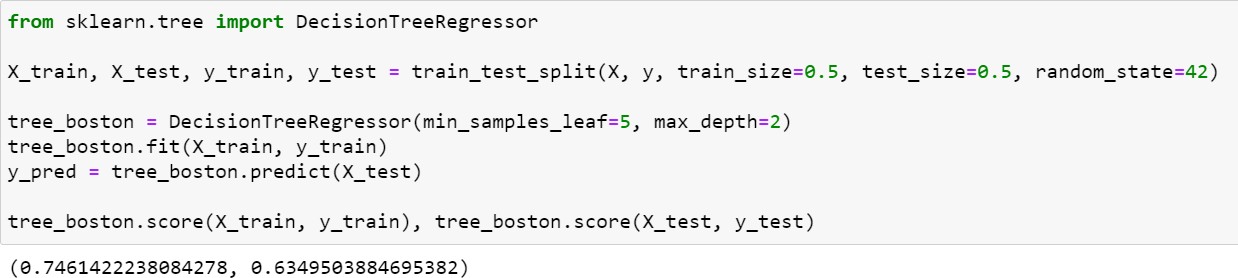


**61**

©copyright CIEL 2015

# 8.3.2 Fitting Regression Trees

* + Accuracy of tree regressor



Decision tree regressor

* min\_samples\_leaf: minimum number of samples required to be at a leaf node
* max\_depth: maximum depth of the tree If

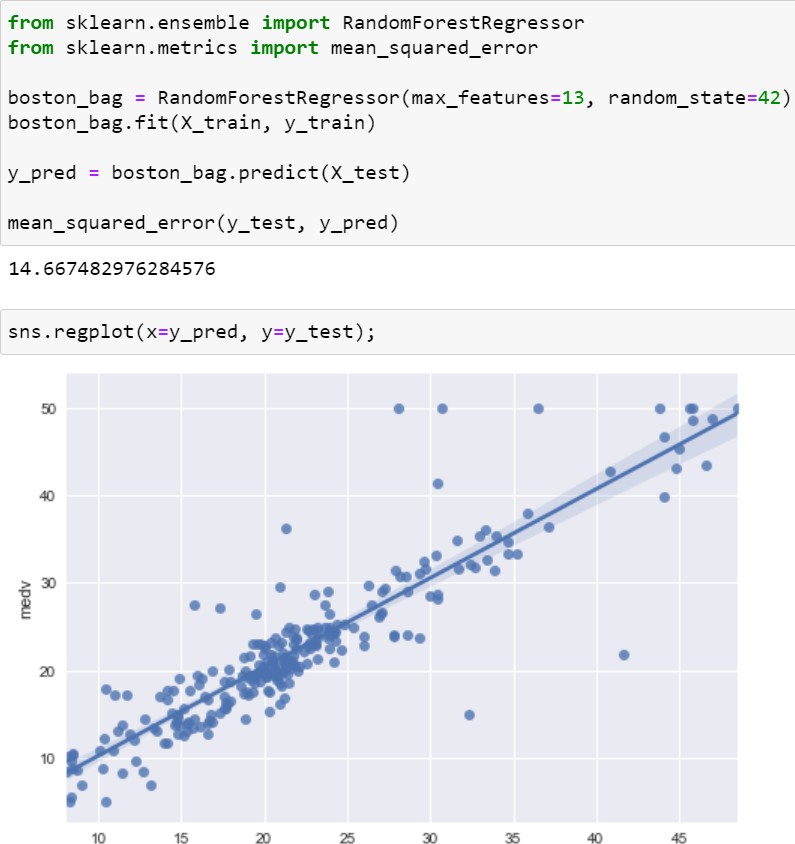
None, then nodes are expanded until all leaves are pure or until all leaves contain less than min\_samples\_split samples

©copyright CIEL 2015 **62**

# 8.3.3 Bagging and Random Forests

* + Bagging

All of predictors



Decision tree regressor

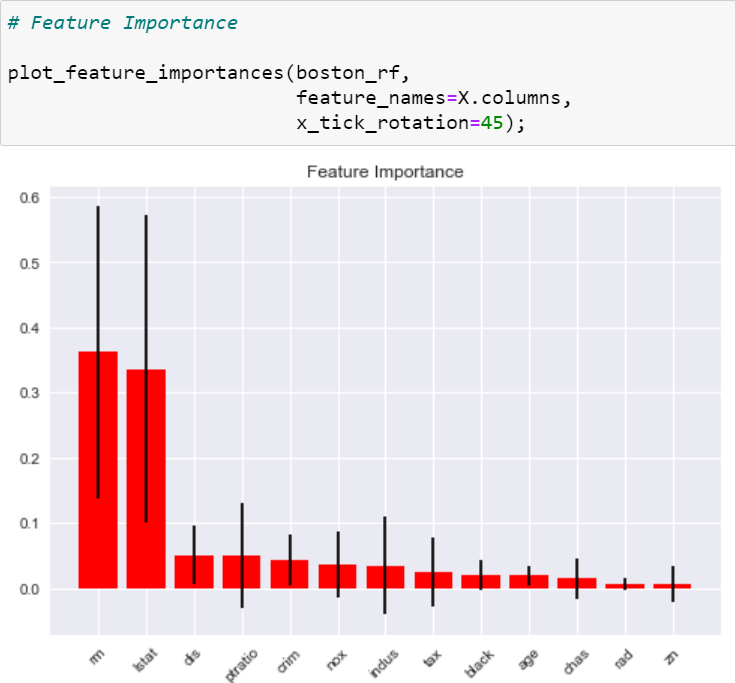
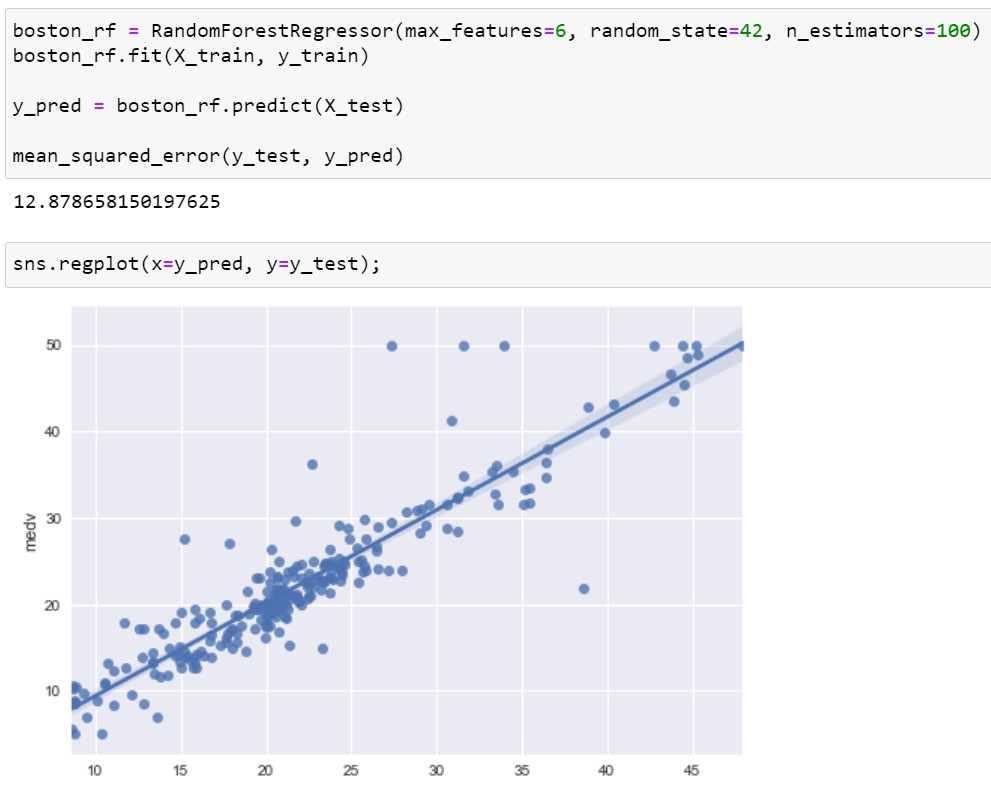
* + - min\_samples\_leaf: minimum number of samples required to be at a leaf node
    - max\_depth: maximum depth of the tree If

None, then nodes are expanded until all leaves are pure or until all leaves contain less than min\_samples\_split samples

©copyright CIEL 2015 **63**

# 8.3.3 Bagging and Random Forests

* + Random forests



Some of predictors

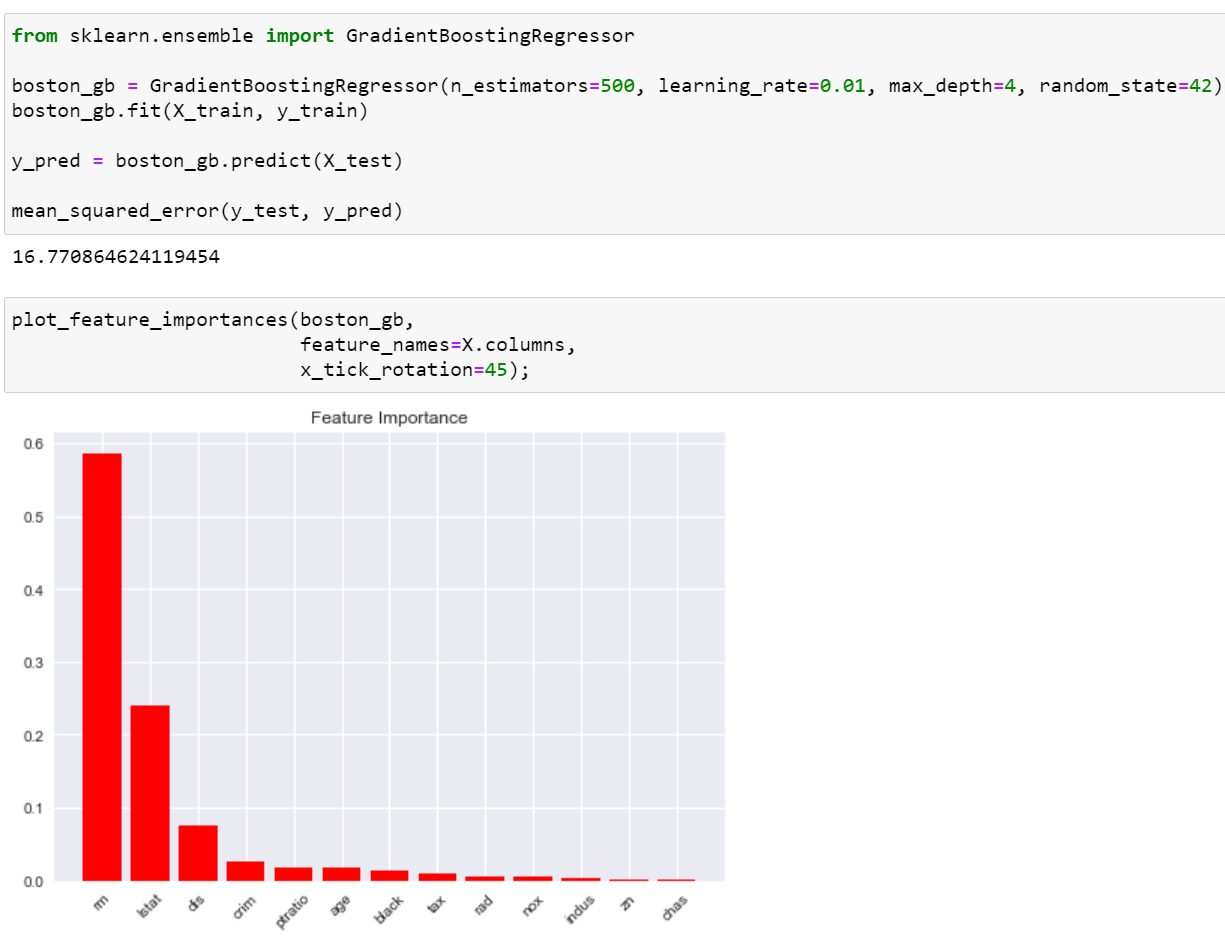
* + - n\_estimators: number of trees in the forest

©copyright CIEL 2015 **64**

# 8.3.4 Boosting

* + Boosting with learning rate (i.e., 𝜆𝜆) = 0.01

Gradient Boosting for regression



* + - n\_estimators: number of boosting stages to perform
    - learning\_rate: learning rate shrinks the

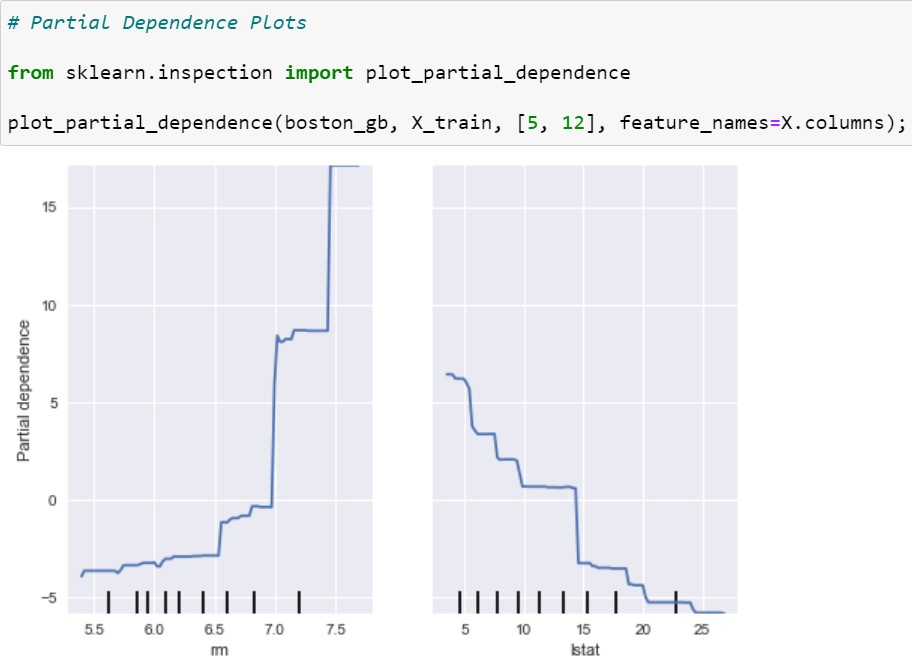
contribution of each tree by learning\_rate

* + - max\_depth: Maximum depth of the individual regression estimators. The maximum depth limits the number of nodes in the tree

©copyright CIEL 2015 **65**

# 8.3.4 Boosting

* + Boosting with learning rate (i.e., 𝜆𝜆) = 0.01 [cont.]
    - Partial dependence plots



The 5th and 12th columns

Partial dependence of features

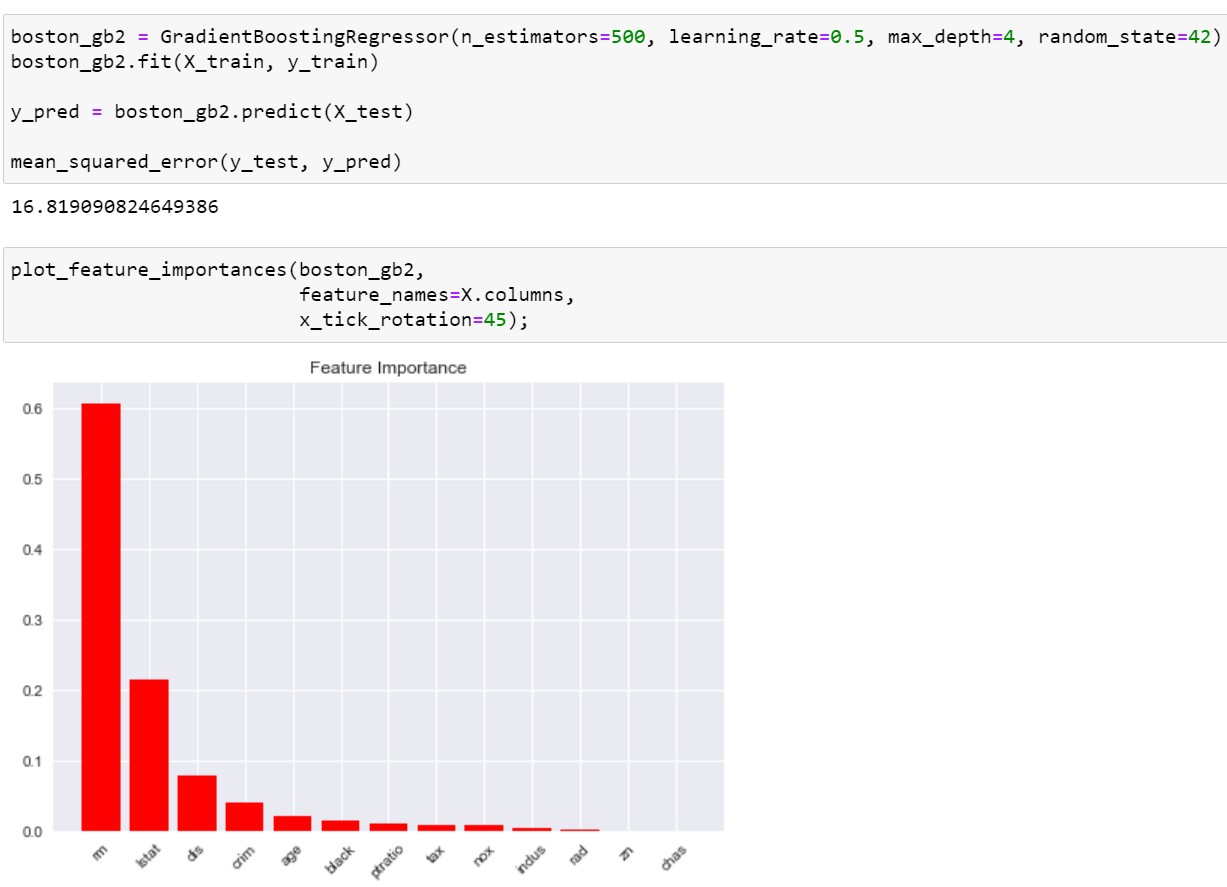
: Partial dependence of a feature (or a set of features) corresponds to the average response of an estimator for each possible value of the feature

©copyright CIEL 2015 **66**

# 8.3.4 Boosting

* + Boosting with learning rate (i.e., 𝜆𝜆) = 0.5

**67**



©copyright CIEL 2015



**Summary & Next Class**

* + - Tree-based methods
    - Python lab

## Summary & Next class

©copyright CIEL 2015 https://ciel.seoultech.ac.kr/

# Summary

* + Tree-based methods
* Decision trees are simple and interpretable models for regression and classification
* However they are often not competitive with other methods in terms of prediction accuracy
* Bagging, random forests, and boosting are good methods for improving the prediction accuracy of trees
  + They work by growing many trees on the training data and then combining the predictions of the resulting ensemble of trees
* The latter two methods, i.e., random forests and boosting, are among the state-of-the-art methods for supervised learning
  + However their results can be difficult to interpret
  + Python lab
* Using scikit-learn (scikit-plot for plotting feature importance)
  + Fitting classification trees and regression trees
  + Bagging and random forests
  + Boosting

©copyright CIEL 2015 **69**

# Assignments

* + eClass > Assignments
* Upload files (do not compress them)
  + Python practices in today’s lecture
* Upload a single ipynb file
* Referring to the lecture slides marked with [P]
* File name: “StudentID” + “\_AssignmentNo w/ 2 digits” + “\_1.ipynb”, e.g., **20211234\_02\_1.ipynb**
  + Textbook exercise problems for today’s lecture
* Conceptual
  + Solving the given problems, then upload your own solution (only docx/hwp formats, not pdf/jpg/bmp etc.)
  + Only include your answers (not need to describe problems)
  + File name: “StudentID” + “\_AssignmentNo w/ 2 digits” + “\_2.ipynb”, e.g., **20211234\_02\_2.docx**
* Applied
  + Implement your Python code for the given problems, then upload another single ipynb file
  + File name: “StudentID” + “\_AssignmentNo w/ 2 digits” + “\_1.ipynb”, e.g., **20211234\_02\_3.ipynb**
  + If not complying with the above policies, some penalty on assignment scores may be given.

©copyright CIEL 2015 **70**

# Course Schedule (Tentative)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Week** | **Topics** | **Note** | **Date (W)** | **Date (M)** |
| 1 | Orientation, Statistical Learning (Ch2) | Online | 03/03 | 03/08 |
| 2 | Statistical Learning (Ch2), Python Programming | Online | 03/10 | 03/15 |
| 3 | Probability & Statistics | Online | 03/17 | 03/22 |
| 4 | Probability & Statistics | Online | 03/24 | 03/29 |
| 5 | Linear Regression (Ch3) | Online | 03/31 | 04/05 |
| 6 | Linear Regression (Ch3) | Online | 04/07 | 04/12 |
| 7 | Classification (Ch4) | Online | 04/14 | 04/19 |
| 8 | **Midterm exam** | **Class hours (W1-W7)** | **04/21** | **04/26** |
| 9 | Resampling Methods (Ch5) | Online | 04/28 | 05/03 |
| 10 | Linear Model Selection and Regularization (Ch6) | Online | 05/05 | 05/10 |
| 11 | Moving Beyond Linearity (Ch7) | Online | 05/12 | 05/17 |
| 12 | Tree-Based Methods (Ch8) | Online | 05/19 | 05/24 |
| **13** | Support Vector Machines (Ch9) | Online | 05/26 | 05/31 |
| 14 | Unsupervised Learning (Ch10) | Online | 06/02 | 06/07 |
| 15 | **Final exam** | **7pm or Class hours (W9-W14)** | **06/??** | **06/??** |

©copyright CIEL 2015 **71**