Data Science Methods for Clean Energy Research

Week 9 L1: Decision Trees

~~Feb 26, 2017~~ → March 1 2017

UNIVERSITY *of* WASHINGTON

Outline

- Quick review from last time
- Intro tree methods
  - Definition and properties
  - Regression trees
  - Classifier trees
- Python project
- Methods to improve decision trees
  - Bagging, boosting, random forests
- Python example
- Wrap up

Topics last time

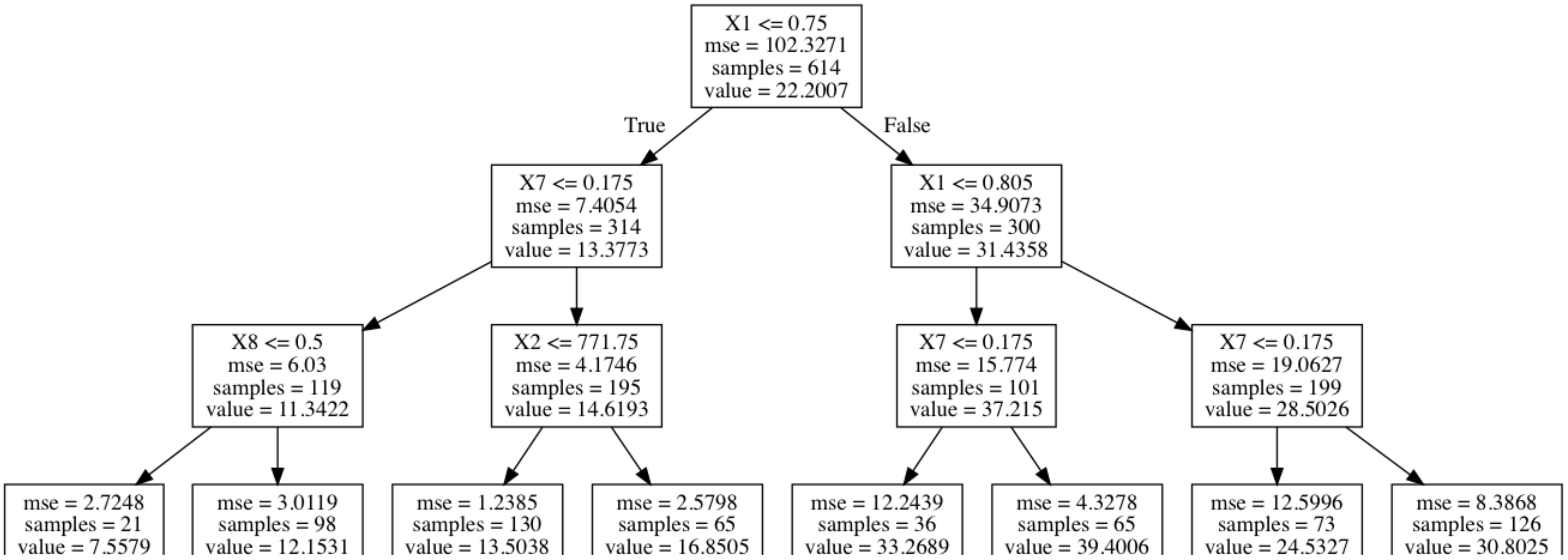
> K-fold cross validation
> Subset selection
> Shrinkage/regularization methods
  – Ridge regression
  – LASSO regression

Big picture concepts:

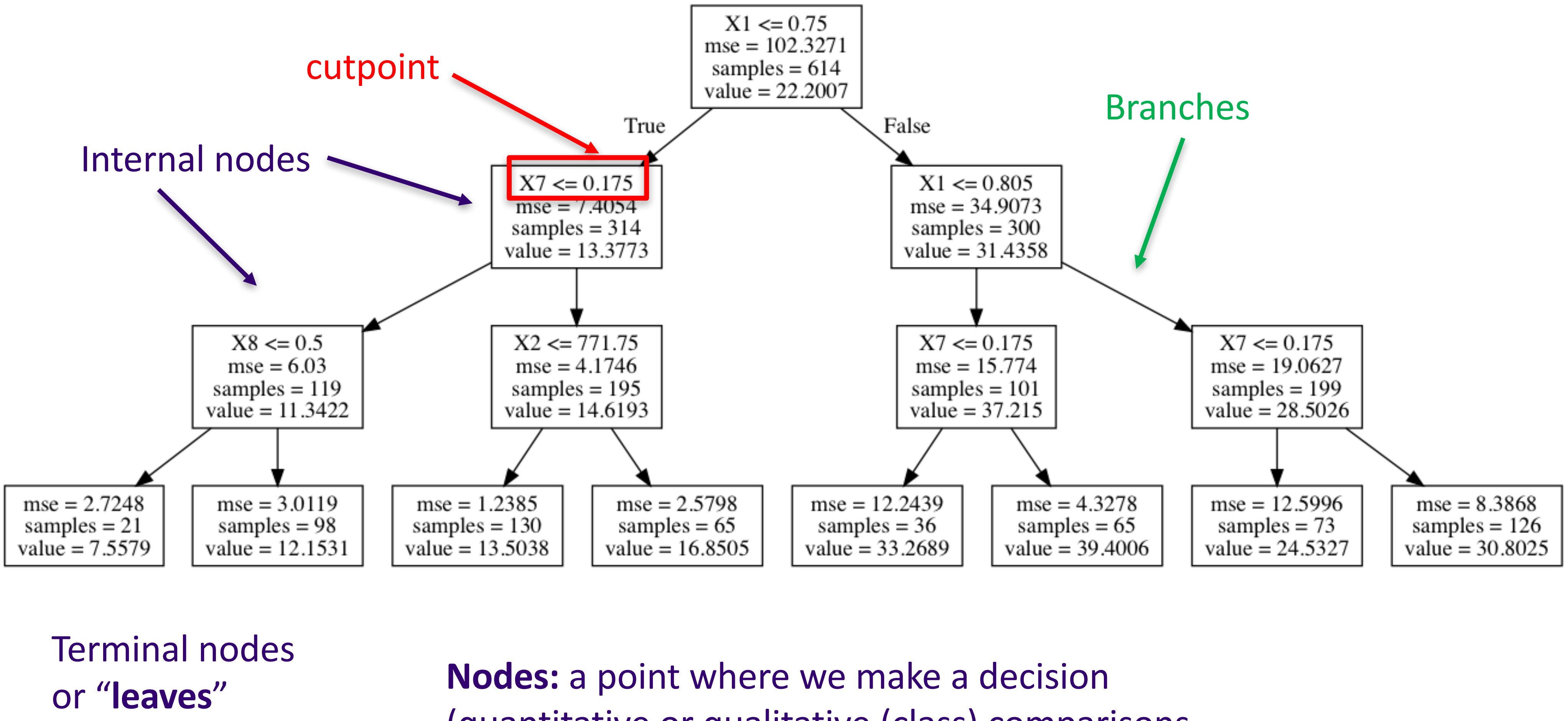
- Test error can be estimated using bootstrap or approximated with known methods
- Subset selection algorithms help determine a smaller set of X's that explain more of the variance in Y
- Regularization and shrinkage methods address the bias/variance tradeoff by adding a penalty that *shrinks* all coefficients toward zero

Decision trees: Example

> **Example of a decision tree that predicts the Y1 predictor from the UCI data set using a simple** 3 level regression decision tree



Decision trees: Nomenclature



Nodes: a point where we make a decision (quantitative or qualitative (class) comparisons
Branch: connections between nodes

Building a regression tree

- **The data in ISL Fig 8.2 shows baseball player salary vs "Hits" vs "Years playing". The regression tree is divided into three regions (R1-R3) based on sub-divisions of the feature space**
- **The goal is to find the** cutpoints (s) **that minimize the RSS**
  - **This is done iteratively by first finding the biggest decreases in RSS by appropriately selecting which predictor (j) and cutpoint give the biggest decrease in RSS…**

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2, \qquad (8.1)$$

^ RSS for a single region for predictor j

$$R_1(j,s) = \{X | X_j < s\} \text{ and } R_2(j,s) = \{X | X_j \geq s\}, \qquad (8.2)$$

^ Cutpoint breaks it into two regions

$$\sum_{i:\, x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\, x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2, \qquad (8.3)$$

^ The total RSS is minimized (8.3)

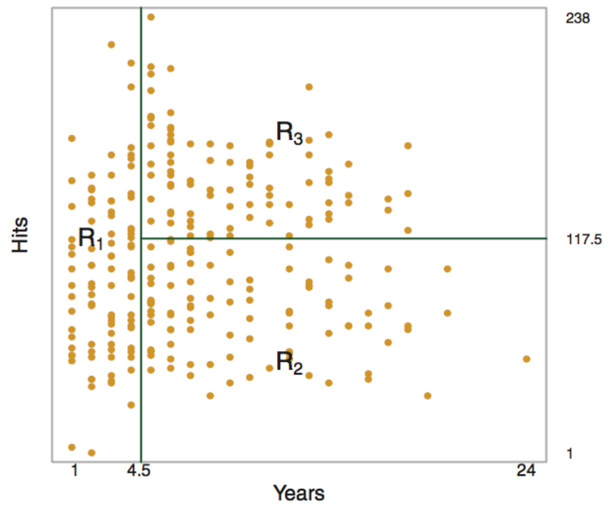


FIGURE 8.2. *The three-region partition for the* Hitters *data set from the regression tree illustrated in Figure 8.1.*

Options for tree building

- **Approach on the last page is known as** "top down" **or** "greedy" **→ goes after the biggest reductions in RSS first (**recursive binary splitting**)**
  - **Does not simulate all possible trees, so possible you are not finding the minimum RSS**
  - **Also possible that trees designed in this approach can lead to overfitting (**too much bias**)**
    - **The procedure of "tree pruning" can address this (not natively supported in sklearn)**
- **Other options/choices in building your DT will be explored in the Python segment…**

Regression trees vs Classification trees

> Concepts are identical
> Error metric is based on the classification error rate not RSS , just as in KNN and related methods
> The DT leaves have your qualitative class assignment, not a quantitative prediction
> Please see section 8.1.2 of ISL for further information, there are some qualitative differences between the different DT's and assessment of their quality...

Big picture concepts in building a tree

> **Decision trees are constructed as an alternative to regression models we have seen**
>
> **Fig 8.7 shows some extreme examples of** ”regression vs. classification” **and comparisons**
>
> **Main advantage is ability to handle nonlinear relationships**
>
> **Main disadvantage is high sensitivity to changes in test set**

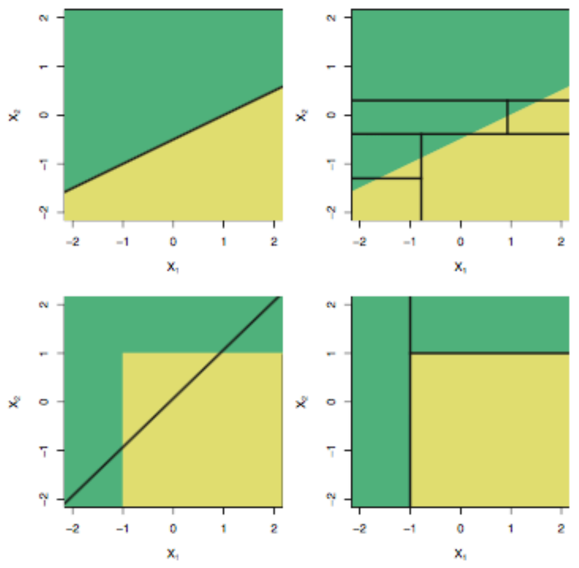


FIGURE 8.7. Top Row: *A two-dimensional classification example in which the true decision boundary is linear, and is indicated by the shaded regions. A classical approach that assumes a linear boundary (left) will outperform a decision tree that performs splits parallel to the axes (right).* Bottom Row: *Here the true decision boundary is non-linear. Here a linear model is unable to capture the true decision boundary (left), whereas a decision tree is successful (right).*

In class Python project: Teams of 2

- **You should use the UCI data set and look to predict the Y1 (heating load) response vs all 8 predictors (X1-X8)**
  - **Break the data into 80% (training), 20% (testing)**
- **Use a Python package that can perform decision tree regression**
- **Make a plot of testing/training error vs. max tree depth (**discuss what you expect the graph to look like first! Really. Draw a sketch!**)**
  - **Don't worry about differentiating validation vs test set data**
  - **You can hijack a lot of the code from last Wed**
- **We will do 2 min** standups **every 10 mins and go for ~50 mins (<u>I will pick teams or ask for volunteer</u>)**

- **If you finish early:**
  - **Study the properties of your tree at the depth you decided on**
    - **Visualize it:** ask me for a code snippet
  - **Study the sensitivity to the training test set size, random seed, etc.**
    - **Test set sensitivity is very important in DT creation!**
  - **Consider a bootstrap to enable true test MSE estimation**
  - **Compare error in DT regression to the best fits obtained from MLR, Ridge and LASSO**
  - **Discuss how you would pick a method to use in the future?**

Building better trees with ensembles

> **Big picture ensemble concepts**
> **Three common ensemble methods**

- **Bagging**
- **Random forests**
- **Boosting [**not covered, but similar in qualitative approach**]**

Ensemble methods (general)

- **A major takeaway from ISL CH8 is that the error of a DT is highly dependent on the training set used –** sometimes much more than in regression
- **One way to avoid this is to use so-called** ensemble methods
  - **Conceptually very similar to resampling (bootstrap and cross-validation)**
    - **Recall main purpose of resampling:** error estimation
  - **In contrast, we go beyond error estimation to use ensemble methods for** improved model training
- **Important: these methods get introduced in ISL in the context of DTs but can also be applied to other ML techniques**

Application of ensemble methods to decision trees: Bagging

- **The bagging concept builds on bootstrap methods**
- **Bagging algorithm**
  1. **Build a model based on B individual bootstrap data sets**
  2. **Make predictions $f(x)$ for each model and average the predicted response**

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x).$$

- **When applying bagging to the creation of DTs, you should grow trees with** high bias **(deep trees) as the bagging will reduce** variance
- **Bagging classifier trees use "**majority vote**" (most common occurring response)**

Application of ensemble methods to decision trees: Random Forest

- **Bagging methods work well, but the bootstrap sets can still be** highly correlated, **leading to increased training set error**
- **To reduce correlation, the random forest method uses a** random sample of predictors **at each split (node) in the tree**
  - **Heuristic is that** $m \sim= p^{0.5}$ **predictors are randomly chosen at each split**
  - **The random DT is combined with bagging to create** a random forest

Next week plan

- > **Unsupervised learning methods, including Python**
- > **Neural networks** OR **ML methods for time series data**
  - – **Which do you prefer?**
- > **Course evaluation and DIRECT update**
  - – **Data Science Option , course numbers , Capstone projects**