GBUS 738 Data Mining

Decision Trees and Random Forests

David Svancer – George Mason University School of Business

GEORGE MASON UNIVERSITY
School of Business

The Basics of Decision Trees

Tree-Based Methods for Regression and Classification

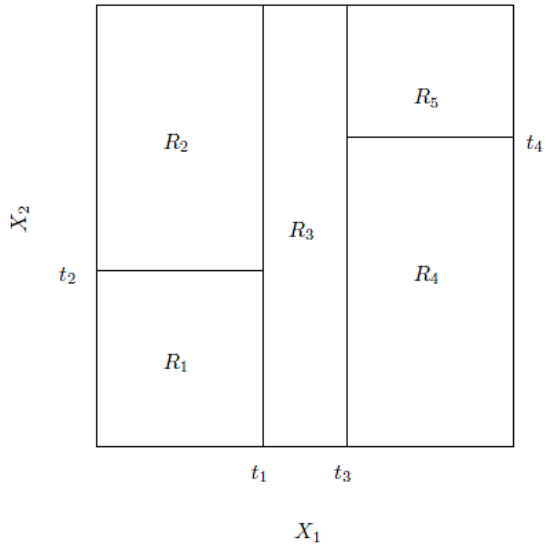
Tree-based methods can be used for both regression and classification problems

Trees are built using ***recursive binary splitting***

- *Stratifies* the predictor space into simple rectangular regions
- Within each region a single prediction is made:
  - Regression – the ***average*** outcome value of the training observations within the region
  - Classification – the ***mode*** of the outcome categories for the training observations within the region

In the regression setting, tree-based methods produce a ***piecewise step function*** *for generating predictions*

Tree-based methods can be represented by a tree diagram and are *easy to interpret*, however, they may lack the predictive power of other supervised learning methods due to their simplicity



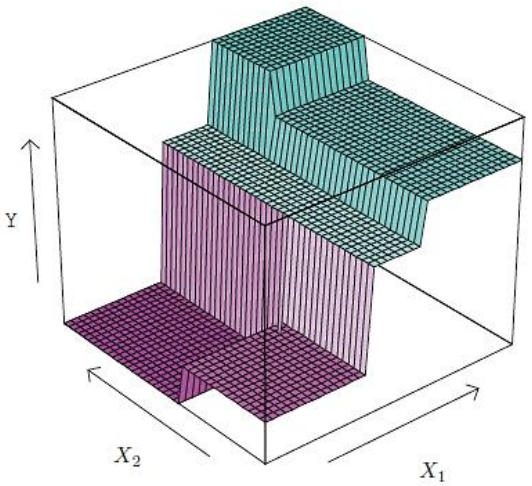


Image source: An Introduction to Statistical Learning with Applications in R James, Witten, Hastie, Tibshirani

The Basics of Decision Trees

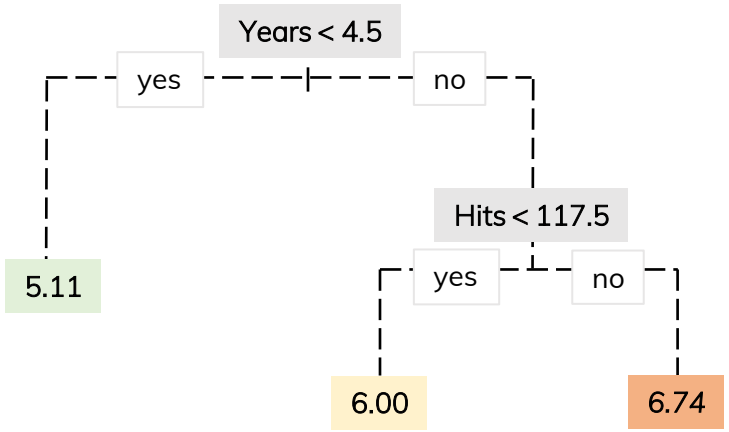
Visualizing Decision Trees - Tree Diagrams

Goal: Predict the log salary of baseball players using the number of years of experience in the major leagues and the number of hits made the previous year

```
# A tibble: 263 x 3
   log_salary  hits years
        <dbl> <int> <int>
 1       6.16    81    14
 2       6.17   130     3
 3       6.21   141    11
 4       4.52    87     2
 5       6.62   169    11
 6       4.25    37     2
 7       4.61    73     3
 8       4.32    81     2
 9       7.00    92    13
10       6.25   159    10
# ... with 253 more rows
```

Tree diagram

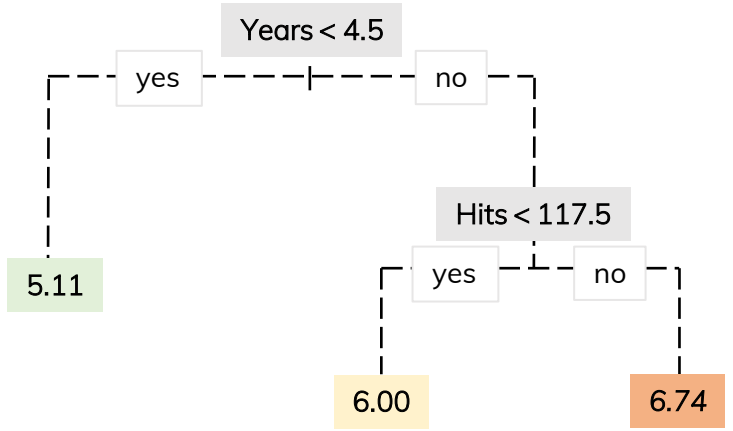
- Each split from a decision tree model creates a pair of branches
- Regions of the predictor space that are split are called **interior nodes**
  - **Years < 4.5** and **Hits < 117.5** are interior nodes
- Regions which are not split further are called **terminal nodes** or **leaves**
  - We have three terminal nodes corresponding to the three regions in the predictor space where we predict a log salary of 5.11, 6.00, and 6.74 respectively



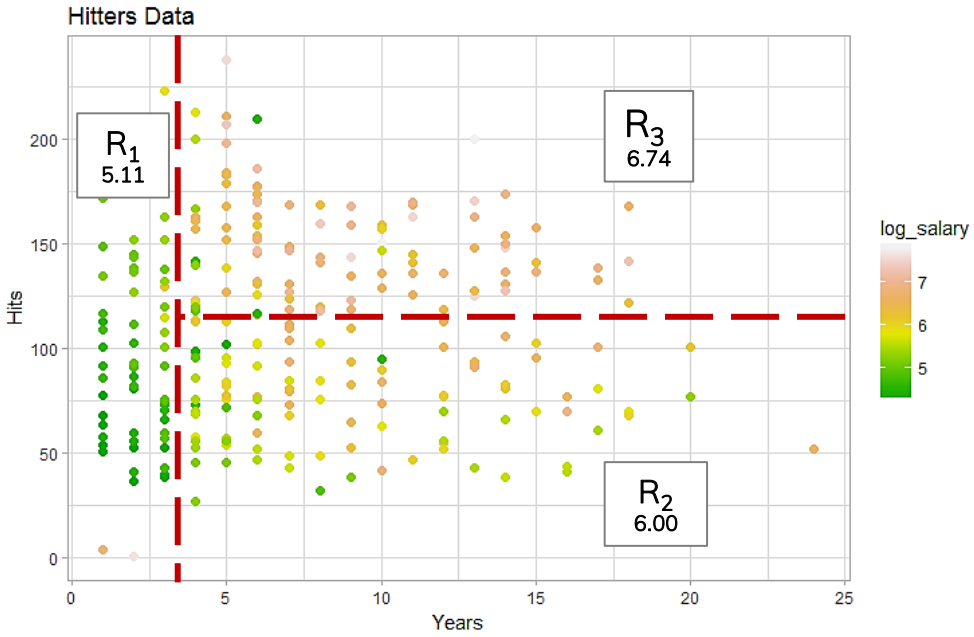
The Basics of Decision Trees

Visualizing Decision Trees

On the right are two visualizations of the same decision tree model which predicts the log salary of baseball players using the number of years of experience in the major leagues and the number of hits made the previous year




```
# A tibble: 263 x 3
   log_salary  hits years
        <dbl> <int> <int>
 1       6.16    81    14
 2       6.17   130     3
 3       6.21   141    11
 4       4.52    87     2
 5       6.62   169    11
 6       4.25    37     2
 7       4.61    73     3
 8       4.32    81     2
 9       7.00    92    13
10       6.25   159    10
# ... with 253 more rows
```



The Basics of Decision Trees

Recursive Binary Splitting for Regression Trees

Find first split of the form $X_j < t_1$, where X_j is one of the predictor variables and t_1 is a cut point along the values of X_j

How is this done?

For each predictor variable:

- **Unique values** in a *quantitative* predictor variable are arranged sequentially
  - Example: 1, 3, 6, 10
- Candidate cut points are chosen as the mid-points
  - Example: 2, 4.5, 8
- Each cut point produces an RSS value, and the minimum over all predictors and cut points is chosen as the first split
- **For categorical predictors**, instead of midpoints of unique numerical values, all non-empty subsets of the unique levels of the predictor variable are used as candidate cut points

The process is repeated ***within each of the two regions*** resulting from the first split. This continues until a stopping criteria is met, such as a terminal node with fewer than 4 training observations

Measure For Choosing a Split in the Predictor Space

Regression – Minimize the Residual Sum of Squares (RSS)

$$\sum_{i:\, x_{ij} < t_j} (y_i - \widehat{y_L})^2 \quad + \quad \sum_{i:\, x_{ij} > t_j} (y_i - \widehat{y_R})^2$$

Here $\widehat{y_L}$ is the sample average of the outcome values associated with X values to the left of the cut point and $\widehat{y_R}$ is the sample average of the outcome values associated X values to the right

The Basics of Decision Trees

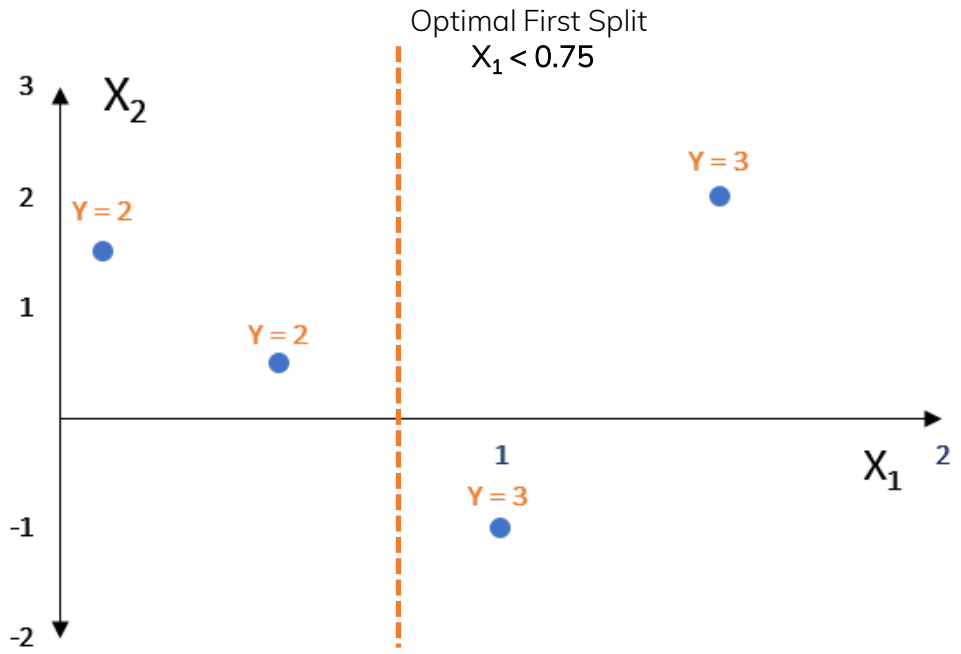
Recursive Binary Splitting for Regression Trees: An Example of the First Split

The example on the right demonstrates how recursive binary splitting would make the first split in the (X_1, X_2) predictor space of a simple data set

The optimal first split is $X_1 < 0.75$

Data

Y	X_1	X_2
2	0.1	1.5
2	0.5	0.5
3	1	-1
3	1.5	2



Possible Splits

Predictor	Cut	Y_L	Y_R	RSS
X_1	0.3	2	2.67	0.67
X_1	0.75	2	3	0
X_1	1.25	2.33	3	0.67
X_2	-0.25	3	2.33	0.67
X_2	1	2.5	2.5	1
X_2	1.75	2.33	3	0.67

The Basics of Decision Trees

Recursive Binary Splitting for Classification Trees

The steps of growing a classification tree are the same as for the regression setting, however, we are now minimizing the *Gini Index* when looking at candidate cut points in the predictor space

To get a feel for this index, suppose a particular node has the following:

Measures For Choosing a Split in the Predictor Space

Classification – Minimize the Gini Index

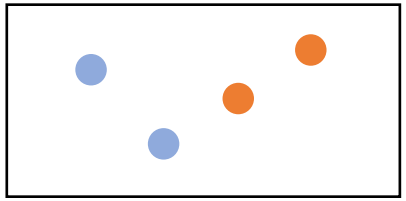
$$\sum_{i=1}^{class\ K} \hat{p}_i\ (1\ -\hat{p}_i)$$

Terminal Node Scenario 1

4 training observations with 2 “Yes” and 2 “No”

Gini Index = $\left(\frac{2}{4}\right)\left(1-\frac{2}{4}\right)+\left(\frac{2}{4}\right)\left(1-\frac{2}{4}\right)=\frac{1}{2}$

Blue – **Yes** Orange - **No**



$\widehat{p_{Yes}} = \frac{2}{4}$

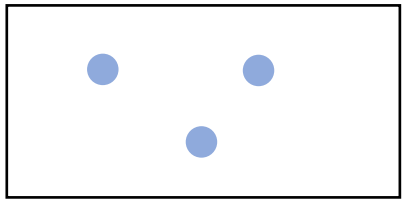
$\widehat{p_{No}} = \frac{2}{4}$

Gini $\rightarrow \frac{1}{2}$

Terminal Node Scenario 2

3 training observations with 3 “Yes” outcome values

Gini Index = = $\left(\frac{3}{3}\right)\left(1\ -\ \frac{3}{3}\right)+(0)(1-0)=0$



$\widehat{p_{Yes}} = \frac{3}{3}$

$\widehat{p_{No}} = 0$

Gini $\rightarrow 0$

School of Business

Decision Tree Hyperparameters

Tree Pruning

Hyperparameters in Decision Tree Models

- *Cost complexity parameter*
  - Referred to as C_p or λ
  - Adds penalty to RSS for large trees
  - $RSS + \lambda|T|$
    - $|T|$ is the number of terminal nodes
- *Tree depth*
  - Controls how long the path from the root to any terminal node can be
- *Minimum n*
  - Minimum data points required in a node for further splitting

tidymodels

```
tree_model <- decision_tree(cost_complexity = tune(),
                            tree_depth = tune(),
                            min_n = tune()) %>%
              set_engine('rpart') %>%
              set_mode('classification')
```

Improving Prediction Accuracy

Bootstrap Aggregation (Bagging)

Bootstrap aggregation (***Bagging***) is a method that can be used to construct more powerful decision tree models in terms of prediction accuracy

The mechanism that powers the technique is *repeated sampling with replacement of the training data* to produce a sequence of decision tree models which are ultimately averaged to obtain a single prediction for a given value in the predictor space

With B bootstrap training data sets:

1. Build B decision tree models

$$\hat{f}^1(x), \hat{f}^2(x), \ldots, \hat{f}^B(x)$$

2. Average the models for the final predictions

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

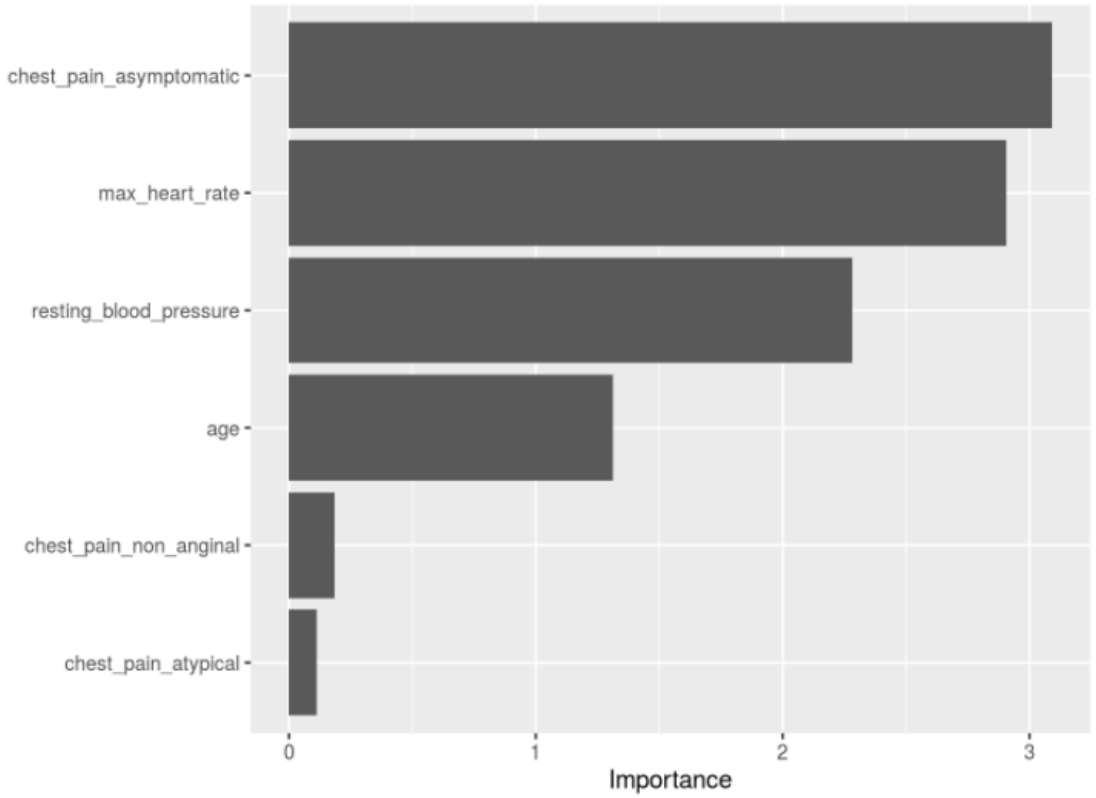
With bagging, ***all predictor variables are considered for splitting the predictor space*** when building decision trees

Improving Prediction Accuracy

Random Forests

A random forest is a special case of bagging, where a random subset of predictor variables are considered for splitting the predictor space in the tree building process

- If there are **p** predictor variables, then the default behavior of a random forest is to randomly select $m \approx \sqrt{p}$ predictor variables for each iteration
- This has the effect of *decorrelating* the sequence of final trees because it removes the influence of strong predictor variables between each iteration of tree building
- The benefit of random forests is that we obtain **variable importance** measures for all predictors in our data. This is another method that can be used for variable selection



Random Forest

Hyperparameters

Hyperparameters in Random Forest Models

- *Number of Predictors to Select at Random*
  - Referred to as ***mtry*** *in tidymodels*
- *Number of Trees*
  - The number of decision trees models to fit and average together into a single ensemble model
- *Minimum n*
  - Minimum data points required in a node for further splitting for each decision tree model

tidymodels

```
rf_model <- rand_forest(mtry = tune(),
                        trees = tune(),
                        min_n = tune()) %>%
            set_engine('ranger', importance = "impurity") %>%
            set_mode('classification')
```