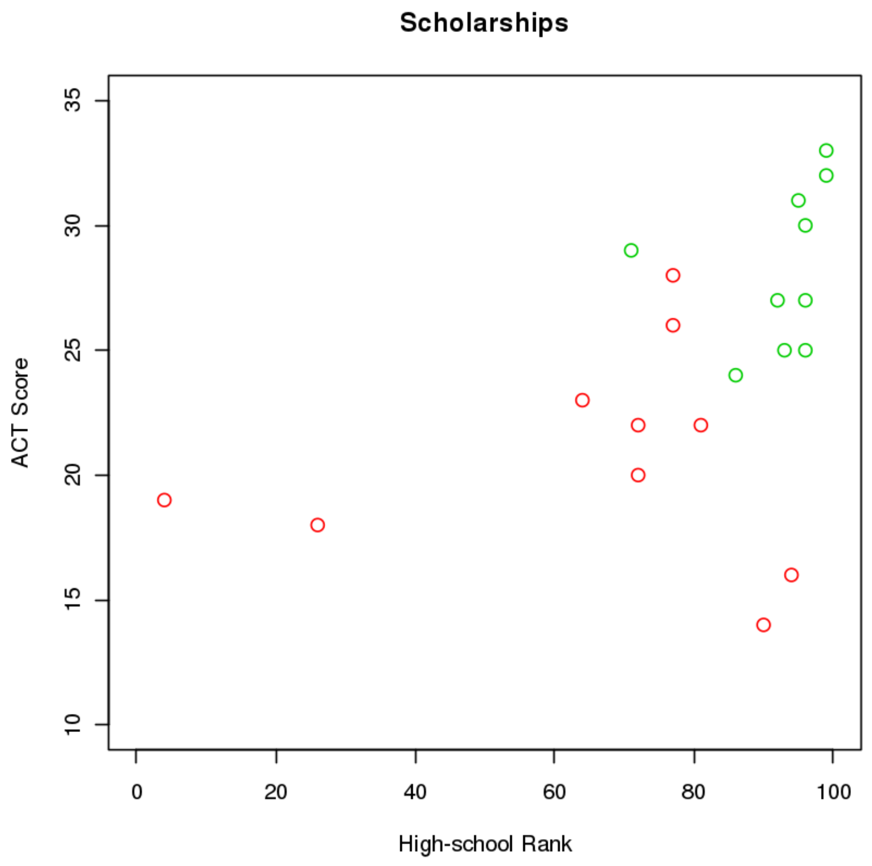
Value Estimation Methods

S. Hagiwara, P. Boily

What is a Classification Tree?

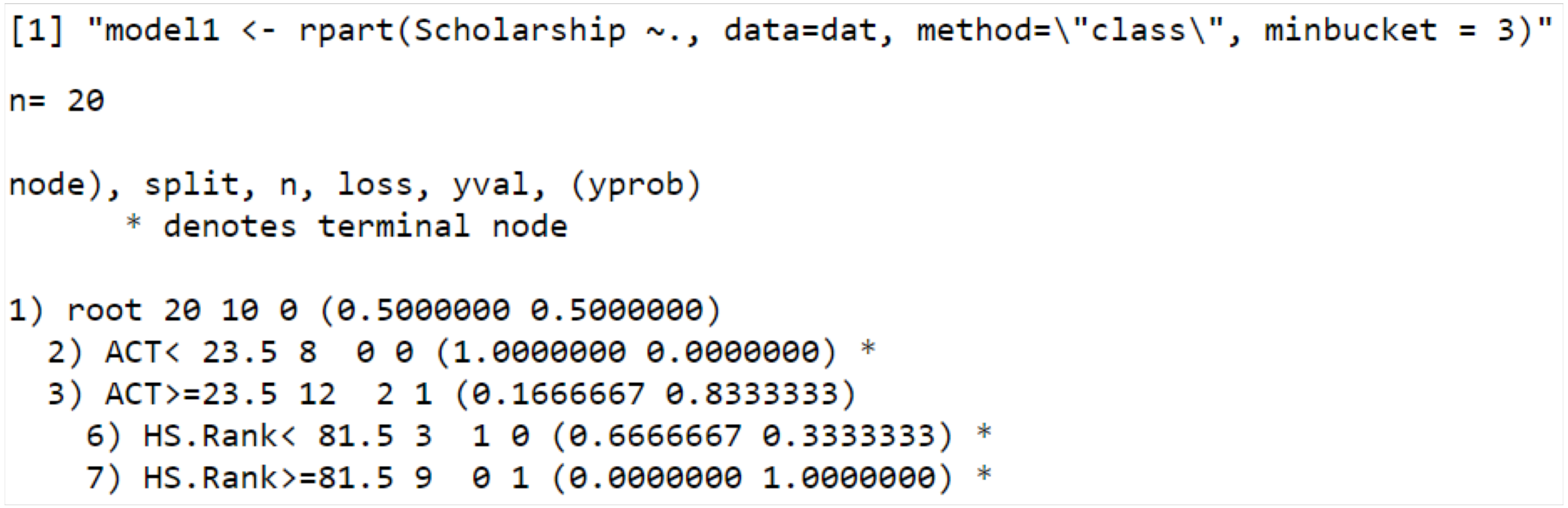
The **concept**:

Can you partition the region using only the horizontal and vertical lines to separate **green circles** (did not receive a scholarship) from **red ones** (did receive a scholarship)?



[Based on Kutner, M. *et al.*, *Applied Linear Statistical Models*, Dataset C.4]

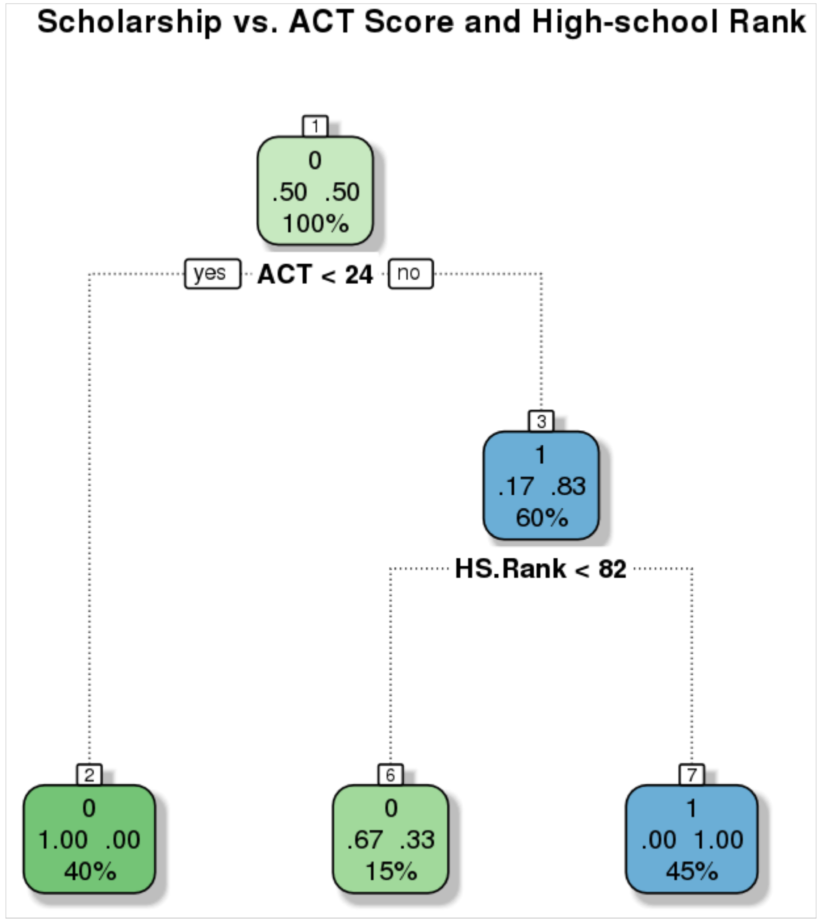
Illustrative Example – R Output

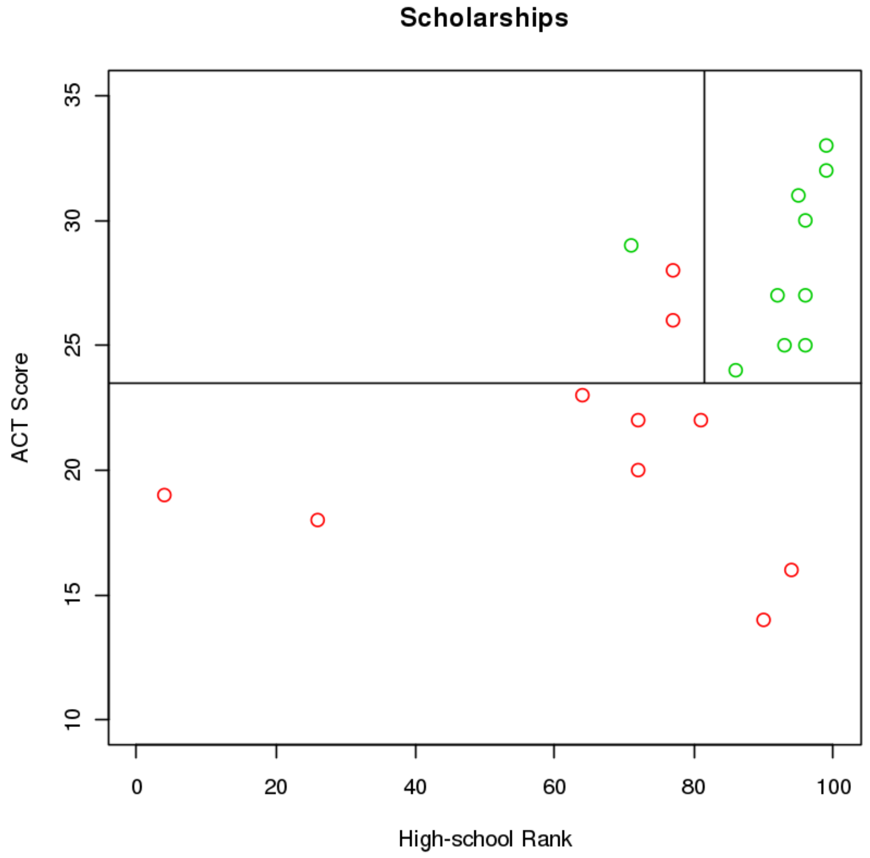


```
[1] "model1 <- rpart(Scholarship ~., data=dat, method=\"class\", minbucket = 3)"

n= 20

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 20 10 0 (0.5000000 0.5000000)
  2) ACT< 23.5 8  0 0 (1.0000000 0.0000000) *
  3) ACT>=23.5 12  2 1 (0.1666667 0.8333333)
    6) HS.Rank< 81.5 3  1 0 (0.6666667 0.3333333) *
    7) HS.Rank>=81.5 9  0 1 (0.0000000 1.0000000) *
```

[Based on Kutner, M. *et al.*, *Applied Linear Statistical Models*, Dataset C.4]

Illustrative Example





[Based on Kutner, M. *et al.*, *Applied Linear Statistical Models*, Dataset C.4]

Growing Classification Tree

1. Sort observations in ascending order (for each variable)
2. Perform a binary split of the region by taking a mid-point between observations
3. For each sub-region R_m, compute the impurity measure Q_m, and add them up
4. Repeat this for all possible splits, and choose the one with smallest $Q = \sum Q_m$
5. Repeat the above process until prespecified minimum node size is reached (which is often set to 5)

Impurity Measures (for Classification)

1. Cross-entropy

$$Q_m = -\sum_{k=1}^{K} \hat{p}_{mk} \ln(\hat{p}_{mk})$$

2. Gini index

$$Q_m = 1 - \sum_{k=1}^{K} \hat{p}_{mk}$$

3. Generalized Gini index

$$Q_m = n_m \left(1 - \sum_{k=1}^{K} \hat{p}_{mk}\right)$$

4. Misclassification error

$$Q_m = 1 - \arg\max_k (\hat{p}_{mk})$$

where $\hat{p}_{mk} = \frac{n_{mk}}{n_m} = \frac{\text{number of observations in } k^{\text{th}} \text{class in node } m}{\text{number of observations in node } m}$

Pruning Classification Tree

Technically, we can split the regions into so many sub-regions such that each rectangle contains only one observation (unless two observations are identical), and such partition provides minimal impurity.

This results in over-fitting (i.e., an overly complex model with less predictive power).

Thus, we use a **cost-complexity criterion** $C_\alpha(T)$ to prune the tree.

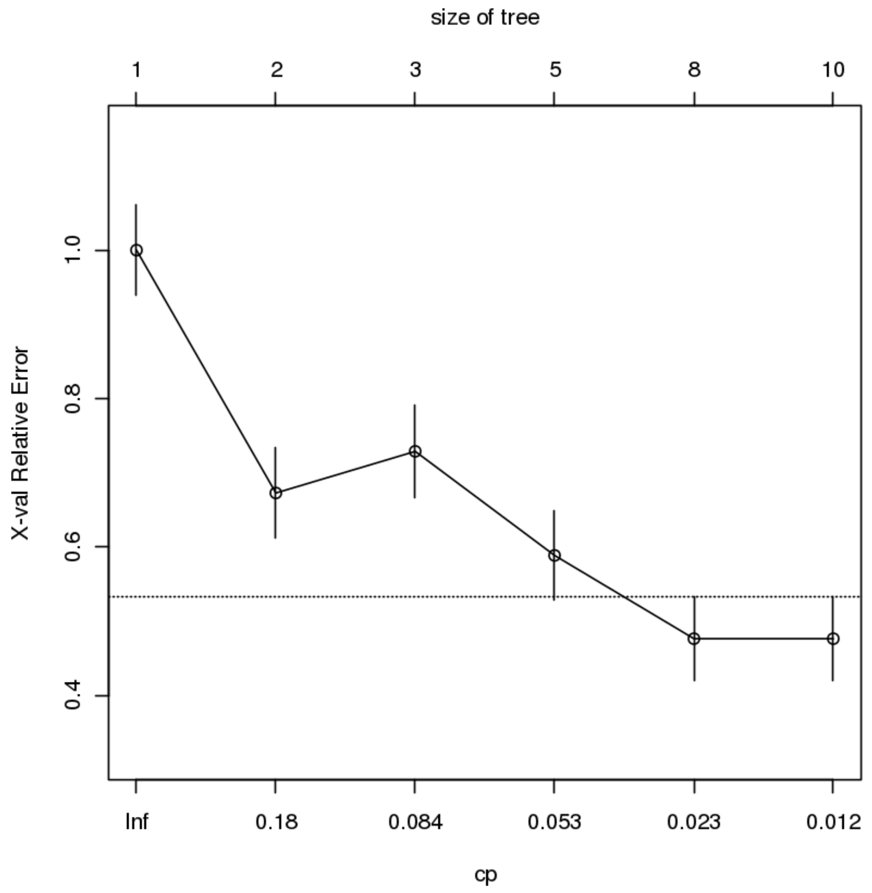
Pruning Classification Tree

Our goal is to minimize

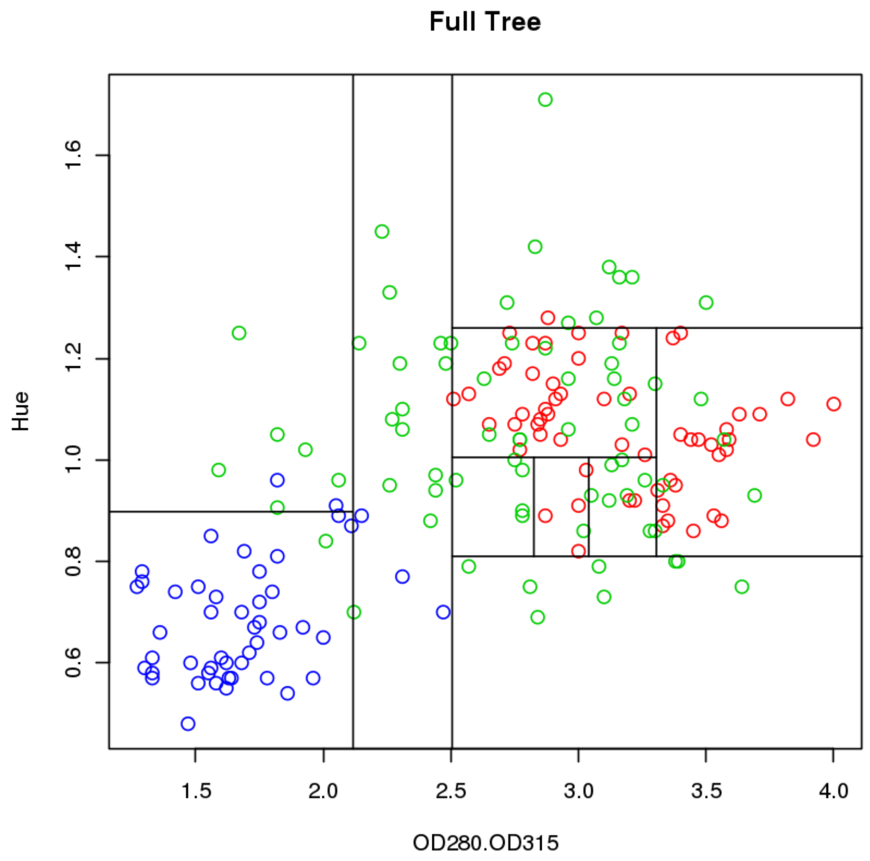
$$C_\alpha(T) = \sum_{m=1}^{|T|} n_m Q_m(T) + \alpha|T|$$

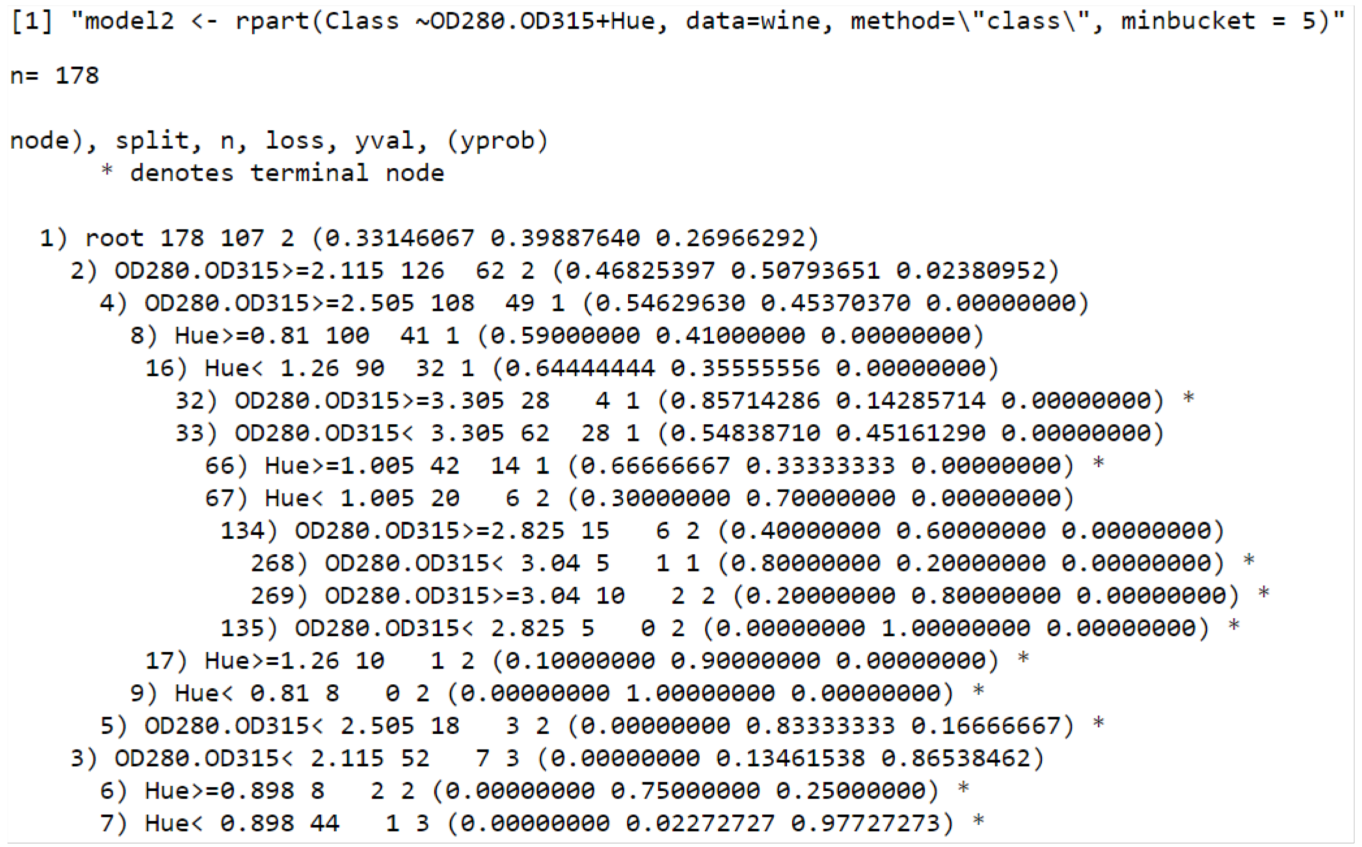
where T is a subtree with $|T|$ terminal nodes, and $\alpha \geq 0$ is a user-defined tuning parameter

- large α penalizes complexity
- $\alpha = 0$: no pruning



Pruning in Action – Wine Dataset

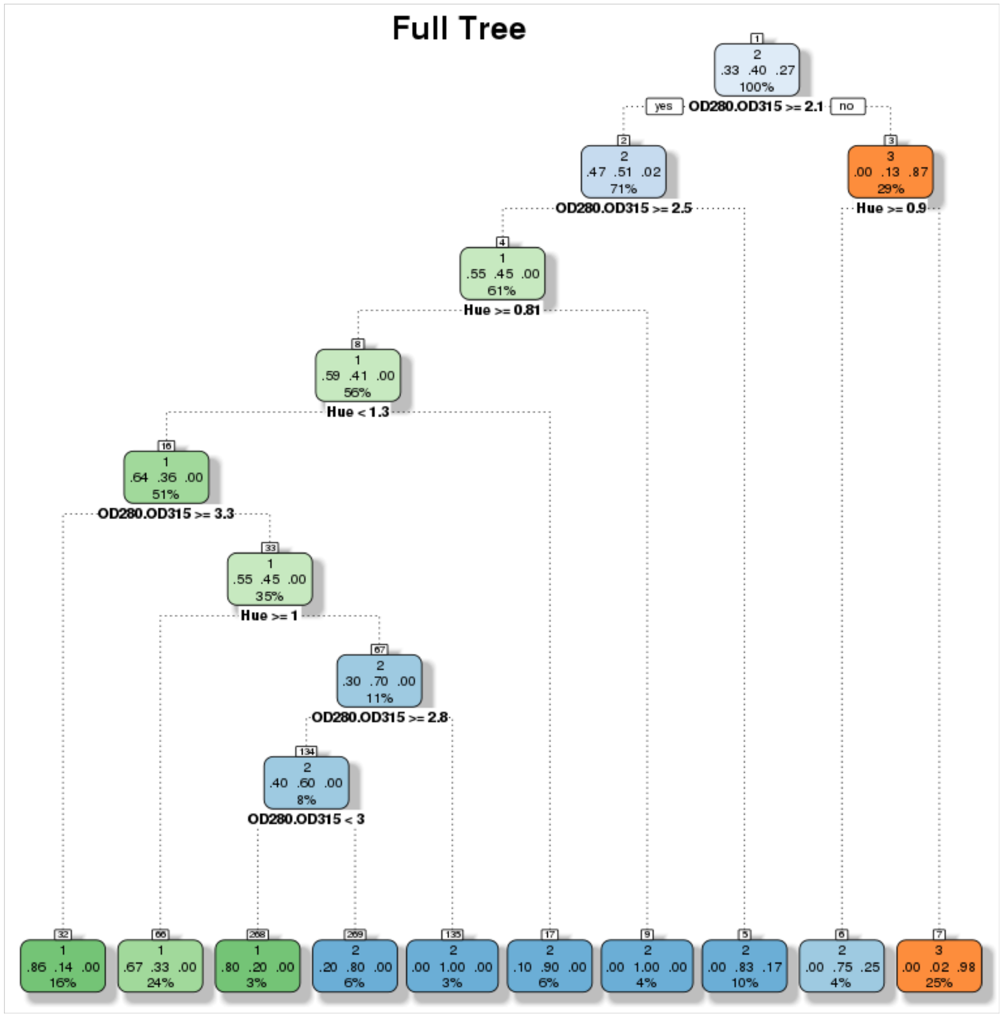




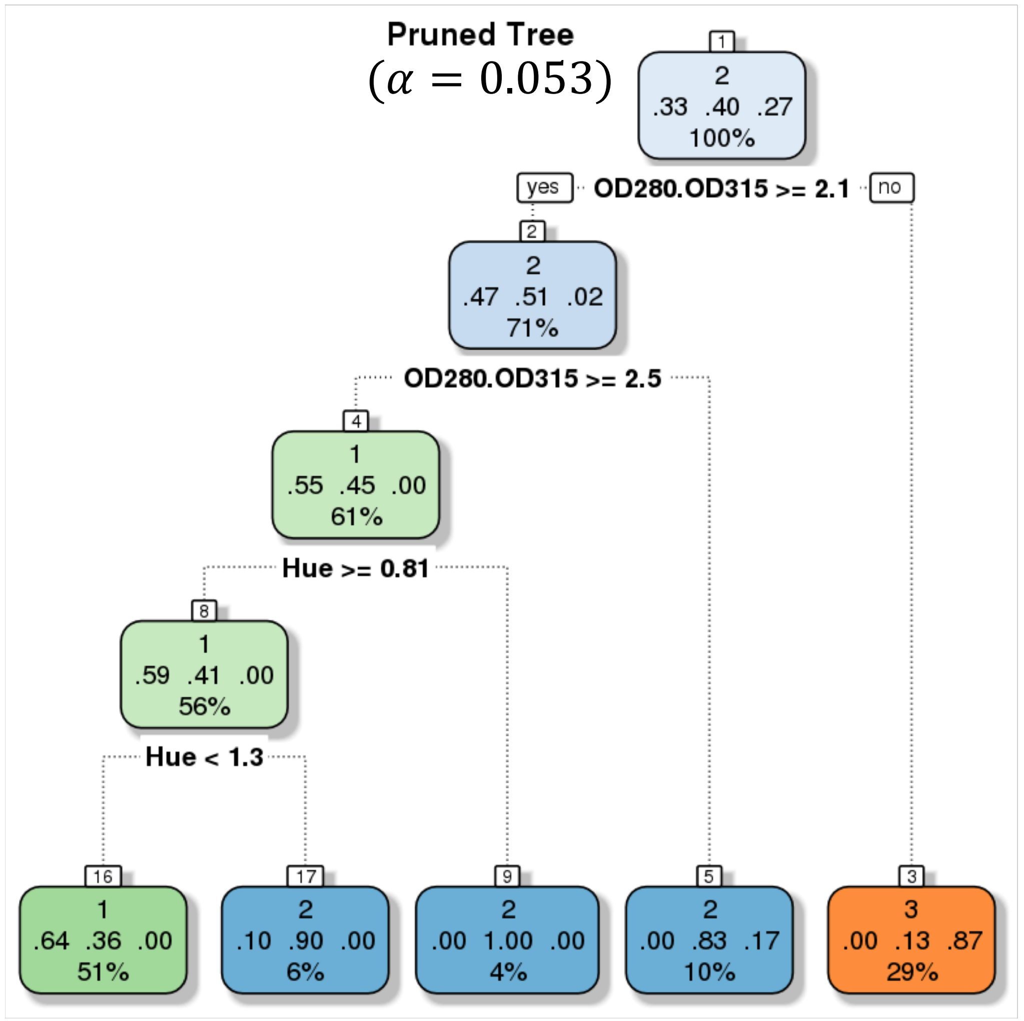
```
[1] "model2 <- rpart(Class ~OD280.OD315+Hue, data=wine, method=\"class\", minbucket = 5)"

n= 178

node), split, n, loss, yval, (yprob)
      * denotes terminal node

  1) root 178 107 2 (0.33146067 0.39887640 0.26966292)
    2) OD280.OD315>=2.115 126  62 2 (0.46825397 0.50793651 0.02380952)
      4) OD280.OD315>=2.505 108  49 1 (0.54629630 0.45370370 0.00000000)
        8) Hue>=0.81 100  41 1 (0.59000000 0.41000000 0.00000000)
         16) Hue< 1.26 90  32 1 (0.64444444 0.35555556 0.00000000)
           32) OD280.OD315>=3.305 28   4 1 (0.85714286 0.14285714 0.00000000) *
           33) OD280.OD315< 3.305 62  28 1 (0.54838710 0.45161290 0.00000000)
             66) Hue>=1.005 42  14 1 (0.66666667 0.33333333 0.00000000) *
             67) Hue< 1.005 20   6 2 (0.30000000 0.70000000 0.00000000)
              134) OD280.OD315>=2.825 15   6 2 (0.40000000 0.60000000 0.00000000)
                268) OD280.OD315< 3.04 5   1 1 (0.80000000 0.20000000 0.00000000) *
                269) OD280.OD315>=3.04 10   2 2 (0.20000000 0.80000000 0.00000000) *
              135) OD280.OD315< 2.825 5   0 2 (0.00000000 1.00000000 0.00000000) *
         17) Hue>=1.26 10   1 2 (0.10000000 0.90000000 0.00000000) *
        9) Hue< 0.81 8   0 2 (0.00000000 1.00000000 0.00000000) *
      5) OD280.OD315< 2.505 18   3 2 (0.00000000 0.83333333 0.16666667) *
    3) OD280.OD315< 2.115 52   7 3 (0.00000000 0.13461538 0.86538462)
      6) Hue>=0.898 8   2 2 (0.00000000 0.75000000 0.25000000) *
      7) Hue< 0.898 44   1 3 (0.00000000 0.02272727 0.97727273) *
```

[Wine Dataset, UCI Machine Learning Repository]

Pruning in Action – Dendrograms



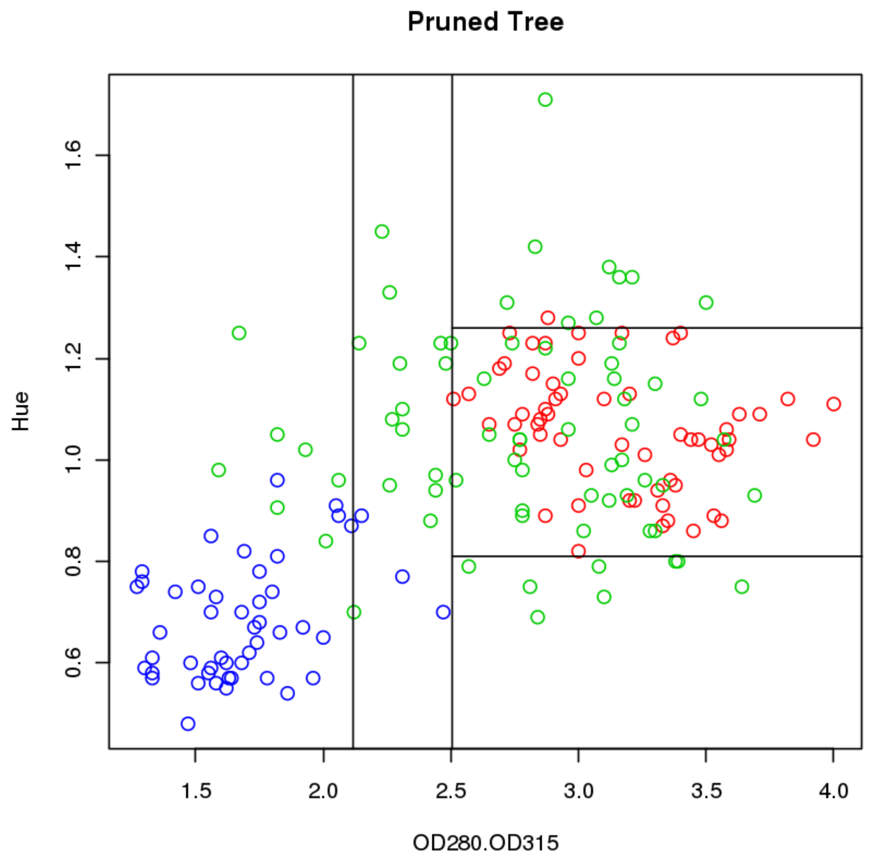
(err. rate = 15.7%)

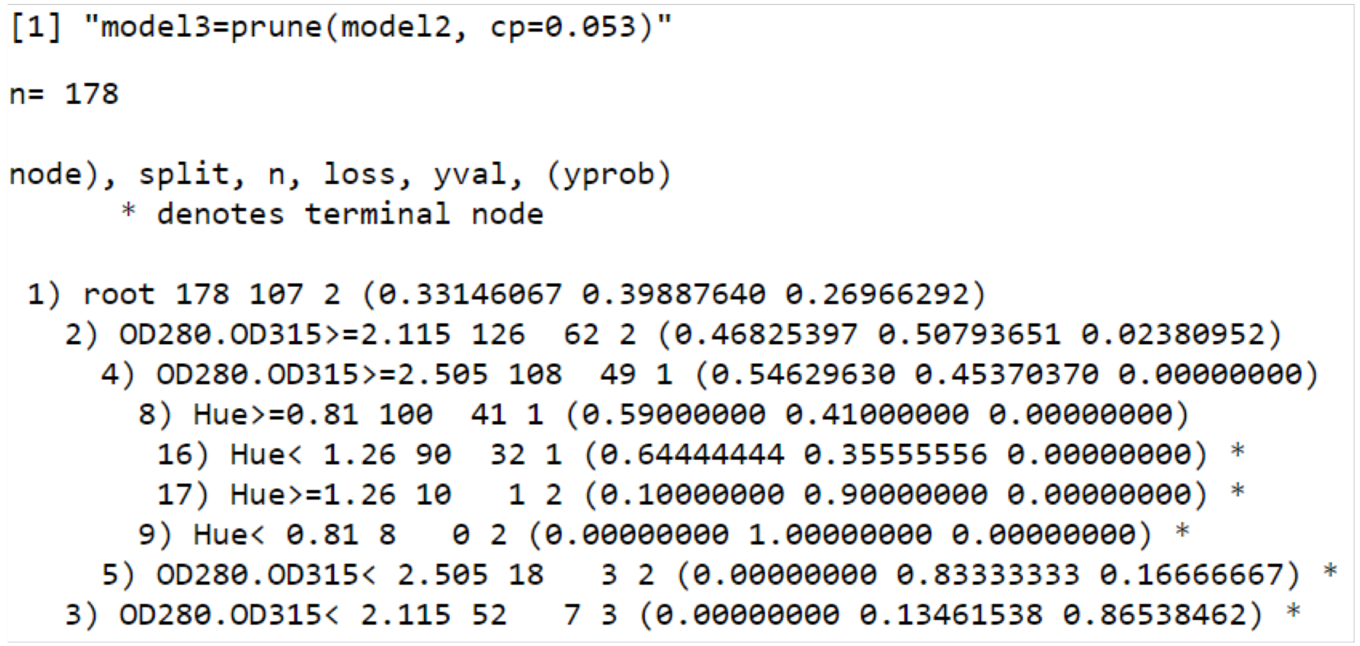


(err. rate = 24.2%)

[Wine Dataset, UCI ML Rep.]

Pruning in Action – Wine Dataset





```
[1] "model3=prune(model2, cp=0.053)"

n= 178

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 178 107 2 (0.33146067 0.39887640 0.26966292)
   2) OD280.OD315>=2.115 126  62 2 (0.46825397 0.50793651 0.02380952)
     4) OD280.OD315>=2.505 108  49 1 (0.54629630 0.45370370 0.00000000)
       8) Hue>=0.81 100  41 1 (0.59000000 0.41000000 0.00000000)
        16) Hue< 1.26 90  32 1 (0.64444444 0.35555556 0.00000000) *
        17) Hue>=1.26 10   1 2 (0.10000000 0.90000000 0.00000000) *
       9) Hue< 0.81 8   0 2 (0.00000000 1.00000000 0.00000000) *
     5) OD280.OD315< 2.505 18   3 2 (0.00000000 0.83333333 0.16666667) *
   3) OD280.OD315< 2.115 52   7 3 (0.00000000 0.13461538 0.86538462) *
```

[Wine Dataset, UCI Machine Learning Repository]

Classification Tree vs. Regression Tree

Conceptually, they are the same as both algorithms:

- For each variable, perform a binary split and find the “best” split that minimizes the impurity measure
- Repeat the process until all terminal nodes have reached minimum node size
- Then prune the tree to reduce complexity

However, they differ in:

- Types of outcomes we wish to predict (categories/probabilities vs. scores/values)
- Impurity measure to minimize (e.g., cross-entropy vs. SSE)

Impurity Measures (for Regression)

Regression trees usually predict observations using **regional means,** i.e., for the m^{th} node R_m, the prediction/estimation is a constant

$$\hat{y}_m = \frac{1}{n_m} \sum_{x_i \in R_m} y_i$$

SSE is used as the impurity measure

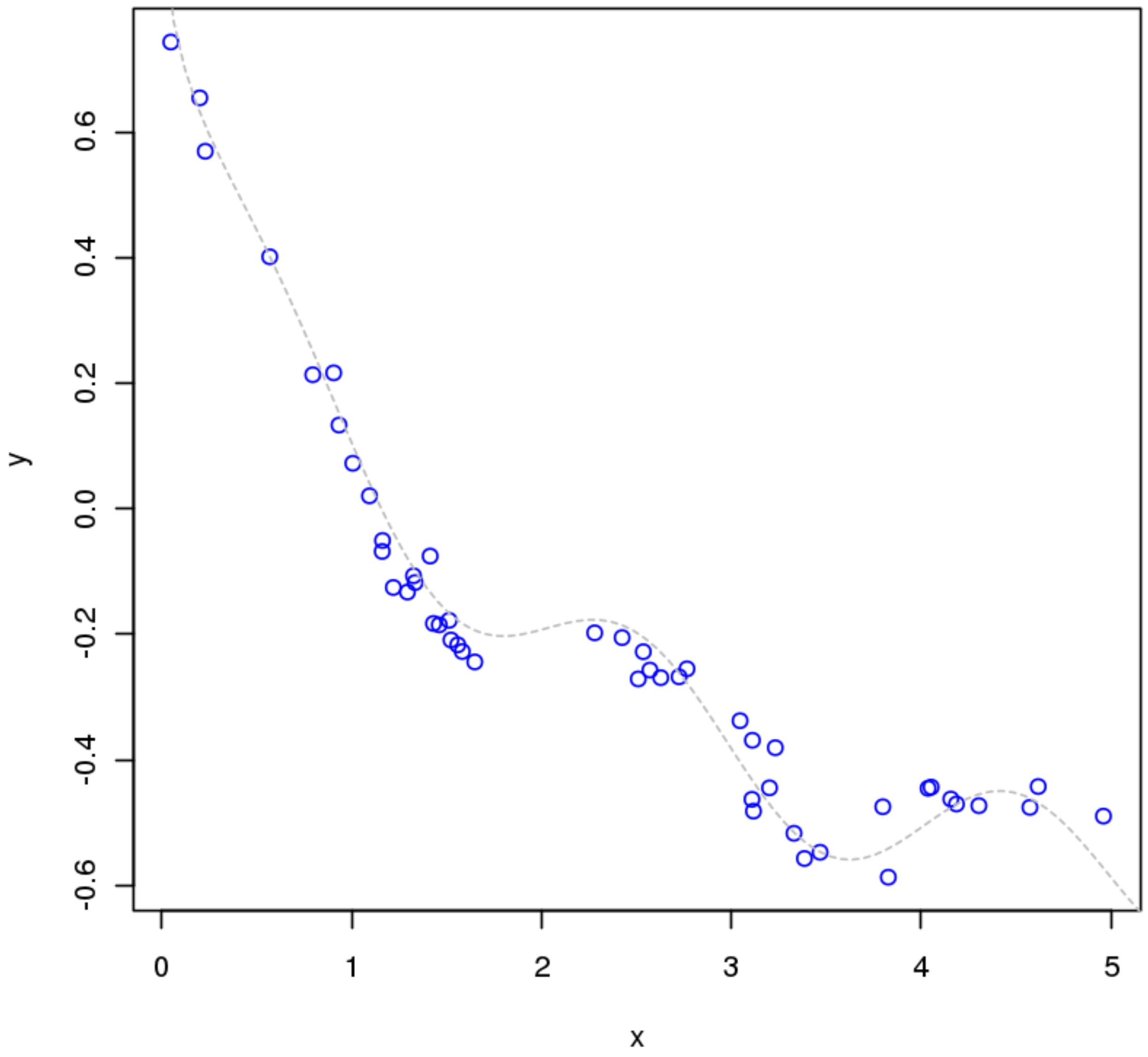
$$Q_m = \sum_{x_i \in R_m} (y_i - \hat{y}_m)^2$$

Illustrative Example

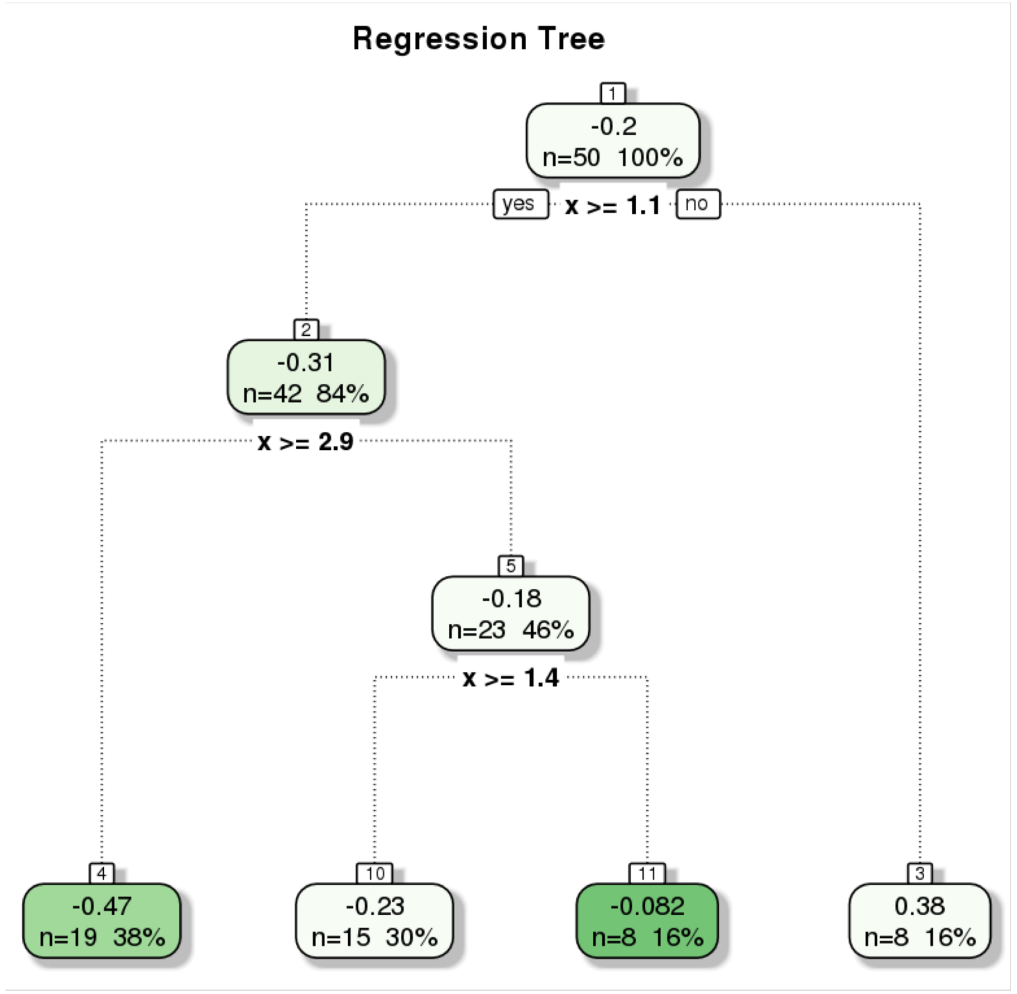
We have a non-linear model

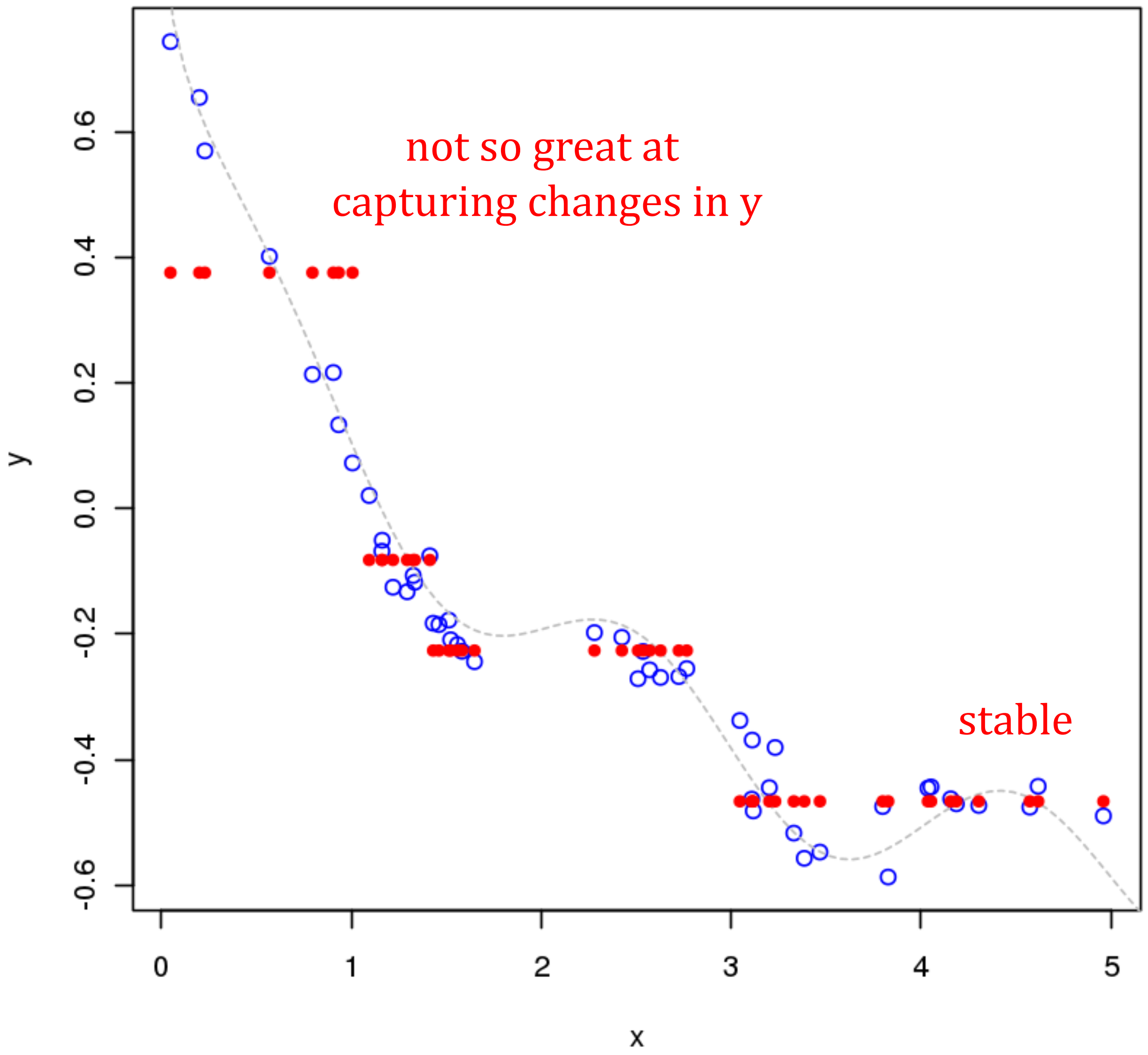
$$y = \frac{\sin(\pi x)}{10} - \sqrt{x} + e^{x/10} + \varepsilon$$

Can a regression tree approximate such a function?



Illustrative Example





not so great at
capturing changes in y

stable

Comments

CARTs are easy to interpret; however, caution is warranted: due to hierarchical nature of splits, a small change in dataset may change the structure of the tree completely.

Averaging via bootstrapping (called *bagging*) may increase the stability of the tree, but interpretability will be lost.

CART does not perform as well when the underlying structure is based on linear combinations of variables.

Comments

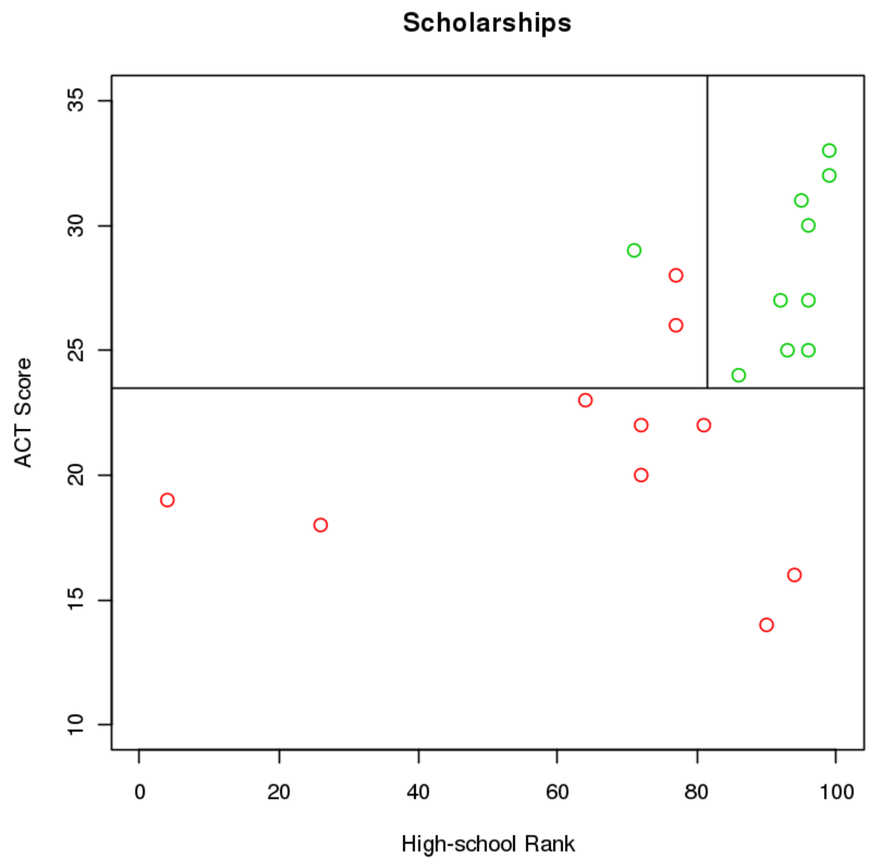
Categorical predictors with large number of categories can be computationally heavy and may lead to severe overfitting.

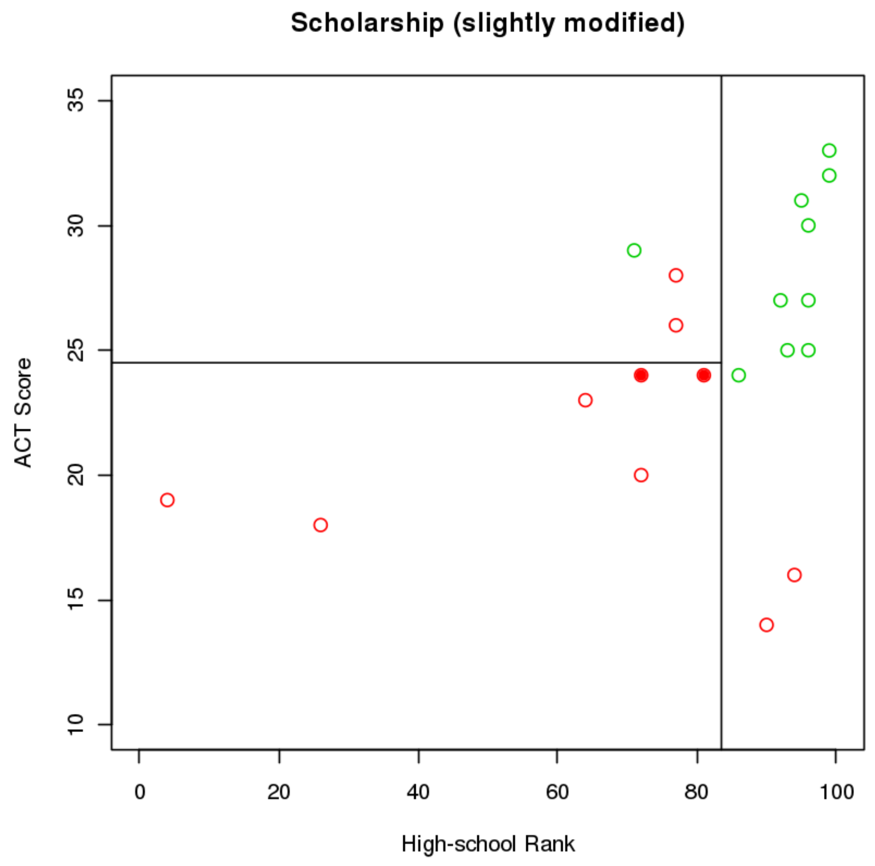
Regression trees lack in **smoothness of prediction surfaces**, which may be problematic as underlying structures are assumed to be smooth in many instances.

`rpart()` can deal with missing observations by using surrogate variables, while `tree()` cannot handle missing cases.

How robust are CARTs to small data modifications?

Comments


Scholarships
ACT Score
High-school Rank
35
30
25
20
15
10
0
20
40
60
80
100


Scholarship (slightly modified)
ACT Score
High-school Rank
35
30
25
20
15
10
0
20
40
60
80
100

Prediction of Housing Prices – Dataset

Sale prices for $n = 1{,}460$ homes in Ames, Iowa, along with 79 attributes on each houses. These attributes cover almost every aspect of a home potential buyers might like to know about before purchasing:

- Type of utilities available
- Type of dwelling
- Overall condition of the house
- Dates of original construction and remodel
- Total square feet of each floor and lot size
- Conditions of driveway and garage
- etc.

[https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data]

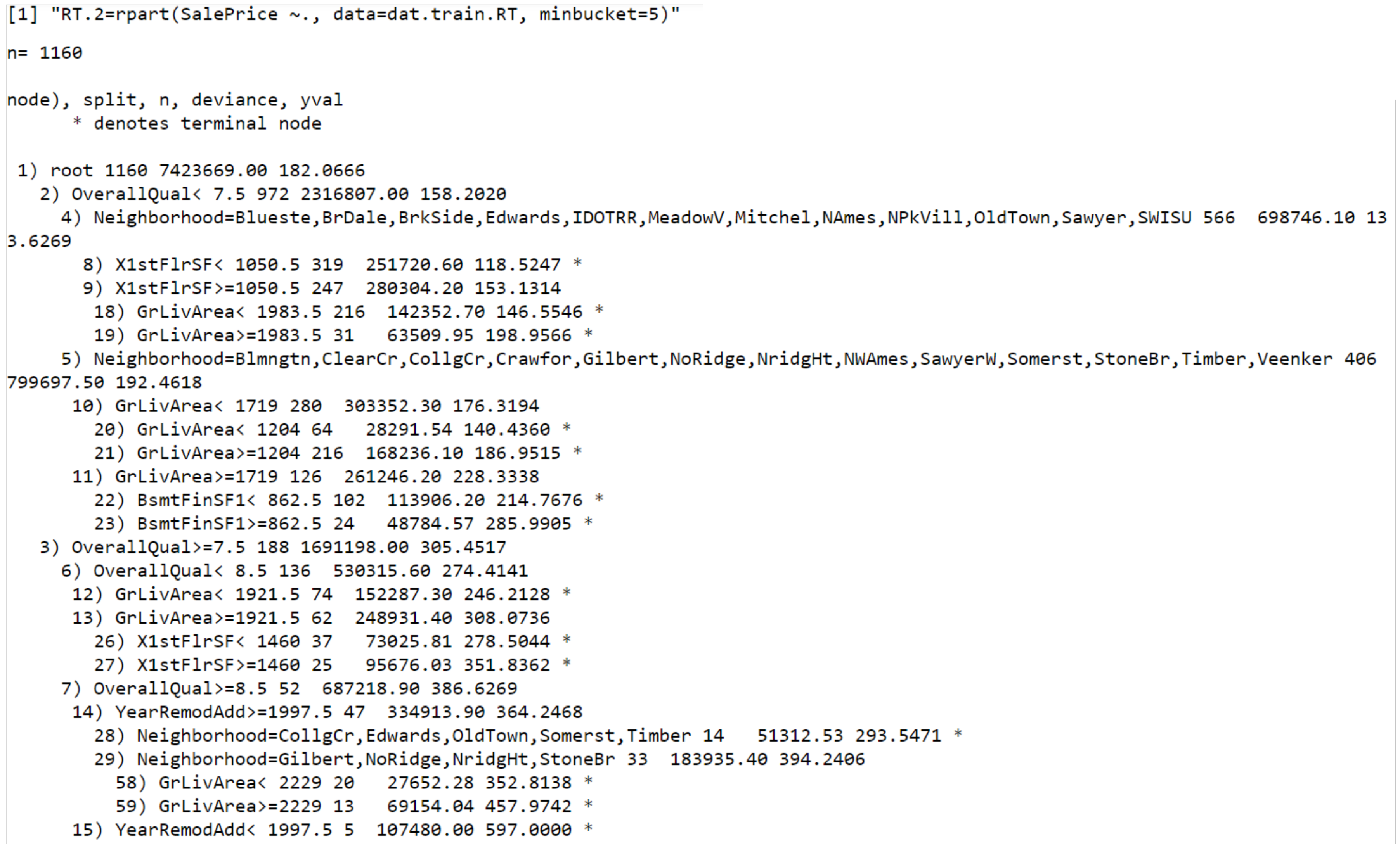
Housing Prices – Objectives

Goal: train a **regression tree** on 1,160 observations and assess how well it predicts the housing prices on the remaining 300 homes.

Secondary Goal: understand and interpret the housing prices based on the available attributes.

For this analysis, we train a tree using all 79 attributes.

Housing Prices – Regression Tree (Full)



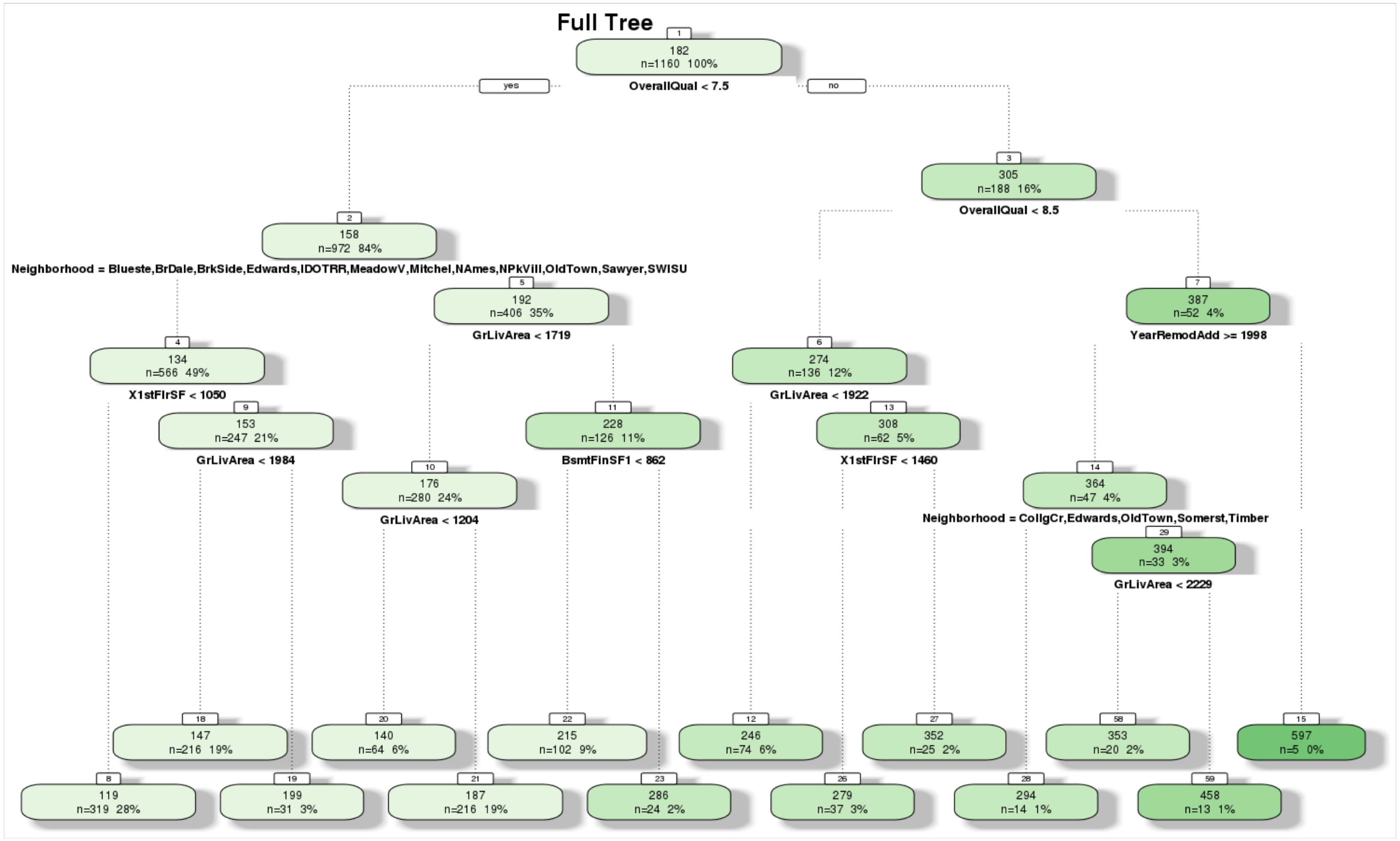
```
[1] "RT.2=rpart(SalePrice ~., data=dat.train.RT, minbucket=5)"

n= 1160

node), split, n, deviance, yval
      * denotes terminal node

 1) root 1160 7423669.00 182.0666
   2) OverallQual< 7.5 972 2316807.00 158.2020
     4) Neighborhood=Blueste,BrDale,BrkSide,Edwards,IDOTRR,MeadowV,Mitchel,NAmes,NPkVill,OldTown,Sawyer,SWISU 566  698746.10 13
3.6269
       8) X1stFlrSF< 1050.5 319  251720.60 118.5247 *
       9) X1stFlrSF>=1050.5 247  280304.20 153.1314
        18) GrLivArea< 1983.5 216  142352.70 146.5546 *
        19) GrLivArea>=1983.5 31   63509.95 198.9566 *
     5) Neighborhood=Blmngtn,ClearCr,CollgCr,Crawfor,Gilbert,NoRidge,NridgHt,NWAmes,SawyerW,Somerst,StoneBr,Timber,Veenker 406
799697.50 192.4618
      10) GrLivArea< 1719 280  303352.30 176.3194
        20) GrLivArea< 1204 64   28291.54 140.4360 *
        21) GrLivArea>=1204 216  168236.10 186.9515 *
      11) GrLivArea>=1719 126  261246.20 228.3338
        22) BsmtFinSF1< 862.5 102  113906.20 214.7676 *
        23) BsmtFinSF1>=862.5 24   48784.57 285.9905 *
   3) OverallQual>=7.5 188 1691198.00 305.4517
     6) OverallQual< 8.5 136  530315.60 274.4141
      12) GrLivArea< 1921.5 74  152287.30 246.2128 *
      13) GrLivArea>=1921.5 62  248931.40 308.0736
        26) X1stFlrSF< 1460 37   73025.81 278.5044 *
        27) X1stFlrSF>=1460 25   95676.03 351.8362 *
     7) OverallQual>=8.5 52  687218.90 386.6269
      14) YearRemodAdd>=1997.5 47  334913.90 364.2468
        28) Neighborhood=CollgCr,Edwards,OldTown,Somerst,Timber 14   51312.53 293.5471 *
        29) Neighborhood=Gilbert,NoRidge,NridgHt,StoneBr 33  183935.40 394.2406
          58) GrLivArea< 2229 20   27652.28 352.8138 *
          59) GrLivArea>=2229 13   69154.04 457.9742 *
      15) YearRemodAdd< 1997.5 5  107480.00 597.0000 *
```

NOTE: Housing prices are given in ($1,000)

Here, we used 6 variables (out of 79) to build the regression tree

- *OverallQual*: Overall Quality
- *Neighbourhood*: Physical locations within Ames
- *X1stFlrSF*: First Floor square feet
- *GrLivArea*: Above Ground living area square feet
- *BsmtFinSF1*: Basement type 1 finished square feet
- *YearRemodAdd*: Remodel date

Housing Prices – Regression Tree (Full)



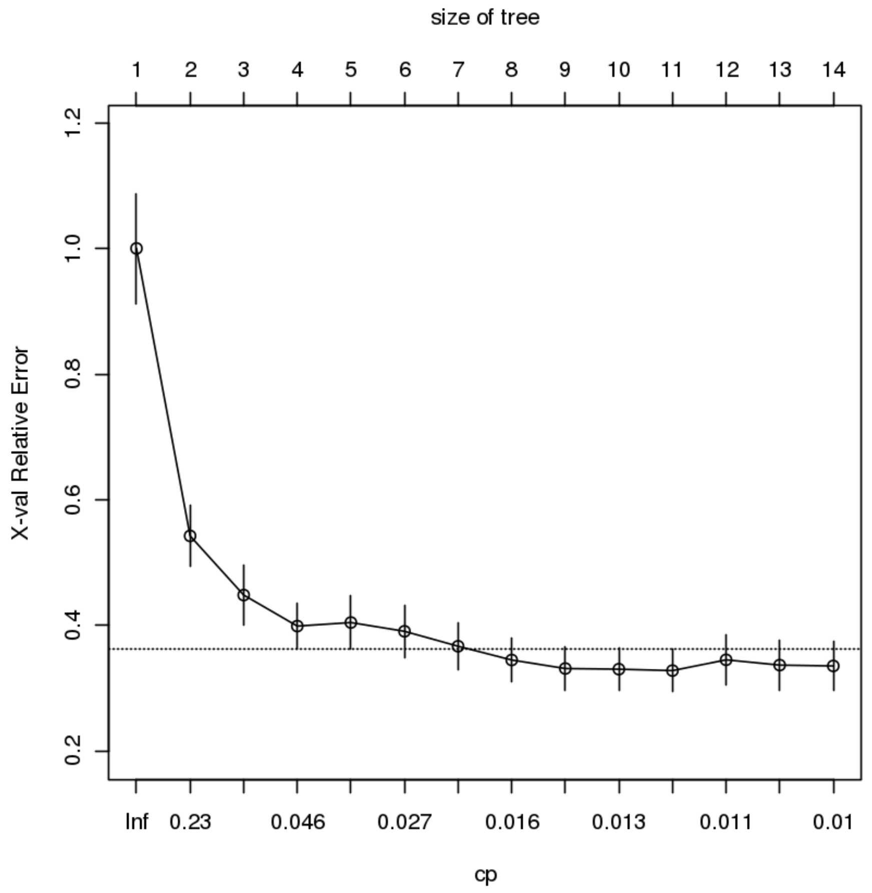
Here, we can really appreciate the interpretability of regression tree:

- **at 2nd level:** according to this model, overall quality of the house above and below 7.5 is the first indicator of the price (the average price is roughly double for houses with quality 7.5+)
- **At 3rd level:** for the left half, we further separate the houses by neighbourhood (again, this is reasonable since certain locations are more preferable than others)
- **at 3rd level:** for the right half, we further separate the houses by overall quality=8 vs. overall quality=9+
- **at 4th level:** for the right most terminal node, there are 5 houses with Remodeling year prior to 1998. These 5 houses are the most expensive houses (on average); perhaps remodeling in "ancient" times means these are historical/heritage houses (hence their high prices)
- **For all others**, the last separation (to each terminal node) is based on the square footage of finished basement or above ground area (this also makes sense as the bigger the house, the more expensive it becomes in general, all else being equal)

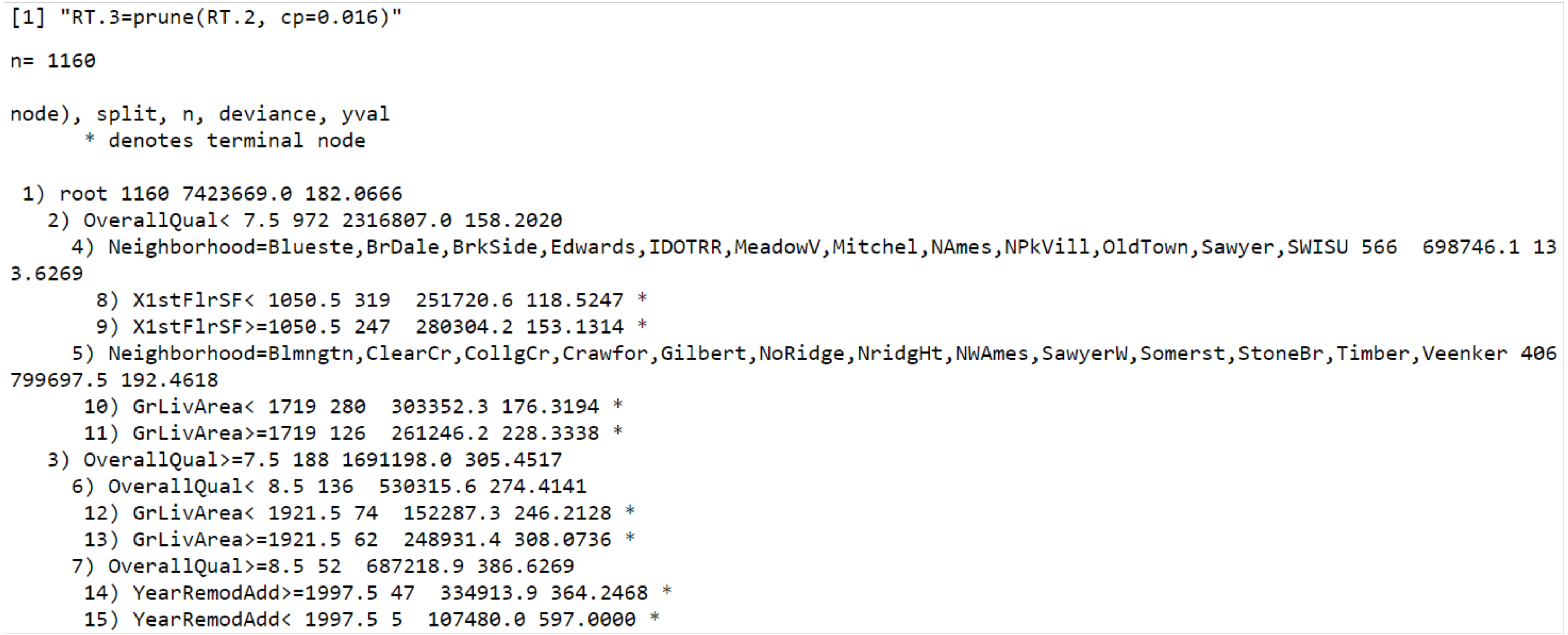
Housing Prices – Pruning the Tree

As in classification tree, we look at optimal cost-complexity parameter α using `plotcp()`.

Running this function multiple times seems to indicate that we should use $\alpha = 0.016$

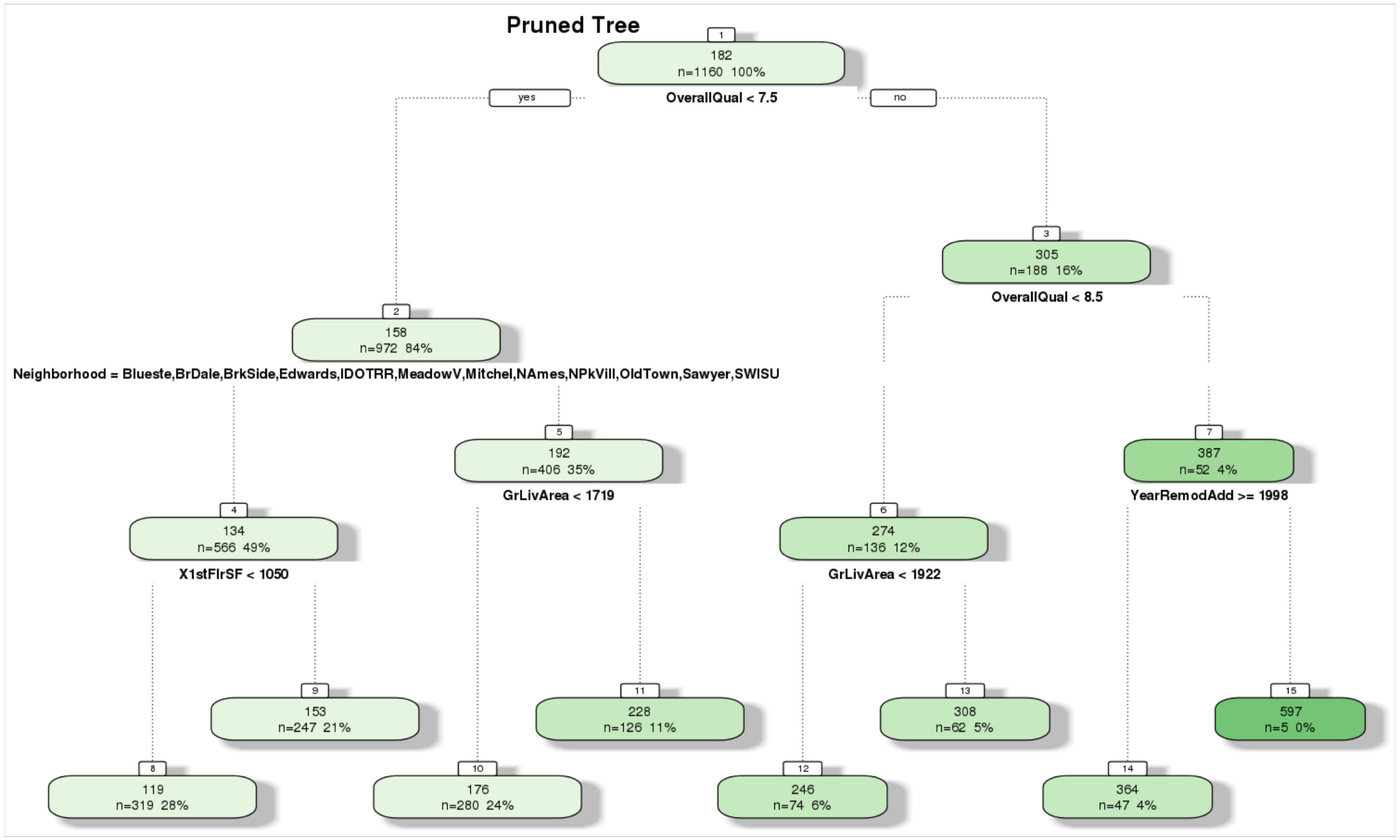


Housing Prices – RT (Pruned)



```
[1] "RT.3=prune(RT.2, cp=0.016)"

n= 1160

node), split, n, deviance, yval
      * denotes terminal node

 1) root 1160 7423669.0 182.0666
   2) OverallQual< 7.5 972 2316807.0 158.2020
     4) Neighborhood=Blueste,BrDale,BrkSide,Edwards,IDOTRR,MeadowV,Mitchel,NAmes,NPkVill,OldTown,Sawyer,SWISU 566  698746.1 13
3.6269
       8) X1stFlrSF< 1050.5 319  251720.6 118.5247 *
       9) X1stFlrSF>=1050.5 247  280304.2 153.1314 *
     5) Neighborhood=Blmngtn,ClearCr,CollgCr,Crawfor,Gilbert,NoRidge,NridgHt,NWAmes,SawyerW,Somerst,StoneBr,Timber,Veenker 406
799697.5 192.4618
      10) GrLivArea< 1719 280  303352.3 176.3194 *
      11) GrLivArea>=1719 126  261246.2 228.3338 *
   3) OverallQual>=7.5 188 1691198.0 305.4517
     6) OverallQual< 8.5 136  530315.6 274.4141
      12) GrLivArea< 1921.5 74  152287.3 246.2128 *
      13) GrLivArea>=1921.5 62  248931.4 308.0736 *
     7) OverallQual>=8.5 52  687218.9 386.6269
      14) YearRemodAdd>=1997.5 47  334913.9 364.2468 *
      15) YearRemodAdd< 1997.5 5  107480.0 597.0000 *
```

Housing Prices – RT (Pruned)



We now have 11 terminal nodes rather than the 14 in the Full Tree:

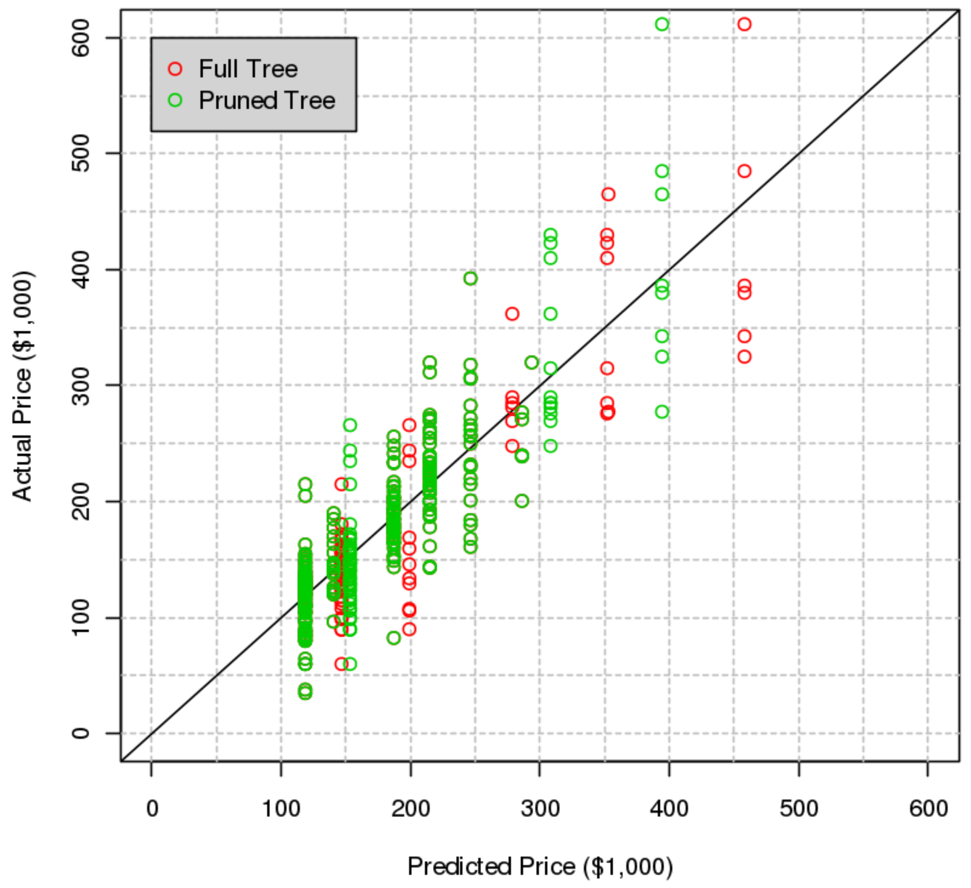
- again, it seems that the **overall quality of the house** is the main variable affecting the house price (should the trunk be affected by pruning?)
- the **size of the house** also seems influential (as most terminal nodes are divided by some house size threshold)
- with the exception of instances where quality of the house is greater than 8.5, the history and the neighbourhood play a role in determining the price of the house

Housing Prices – Full vs. Pruned (Preds)

- Both trees achieve (practically) same level of reduction of SSE
- Both models are fairly symmetrical in their prediction
- So, pruned tree is the preferred model, being simpler.

Model	SSE	Reduction in SSE
None	1,776,836	0%
Full	465,340	73.8%
Pruned	470,899	73.6%

- Can you explain why the stratified structure of the predictions?



MARS Concept

CART produces **flat prediction surfaces**, which may be problematic since:

- these surfaces are **discontinuous** at the joints, which could be an unreasonable assumption in many scenarios
- as a result, a small change in x may introduce a large change in $\hat{y}$

Multivariate Adaptive Regression Splines (MARS) was introduced to overcome these CART shortcomings.

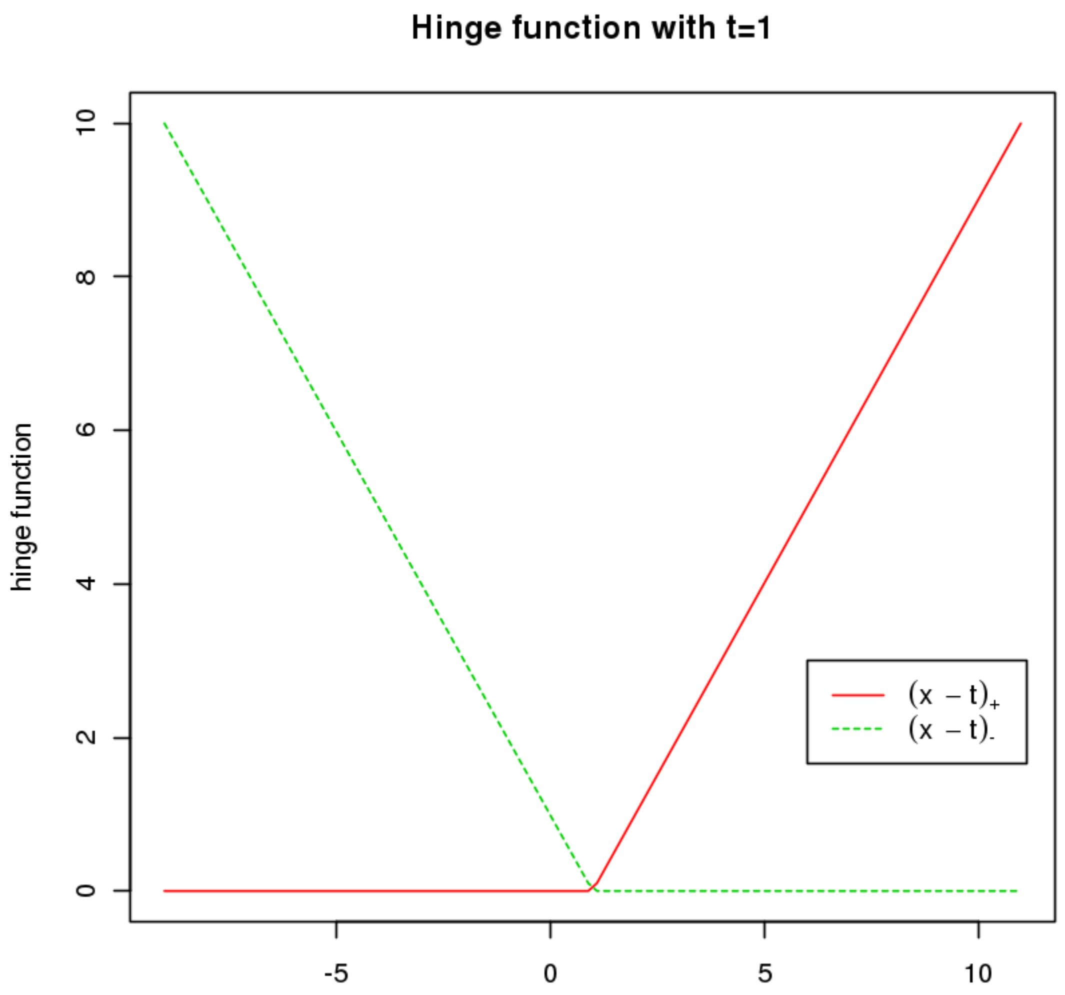
Hinge Function/Basis Function

MARS is built around the concept of a **hinge**/basis function, which is defined as

$$(x-t)_+ = \begin{cases} x - t \text{ if } x > t \\ 0 \ \text{ otherwise} \end{cases}$$

$$(x-t)_- = \begin{cases} t - x \text{ if } t > x \\ 0 \ \text{ otherwise} \end{cases}$$

The location t is called a **knot**.



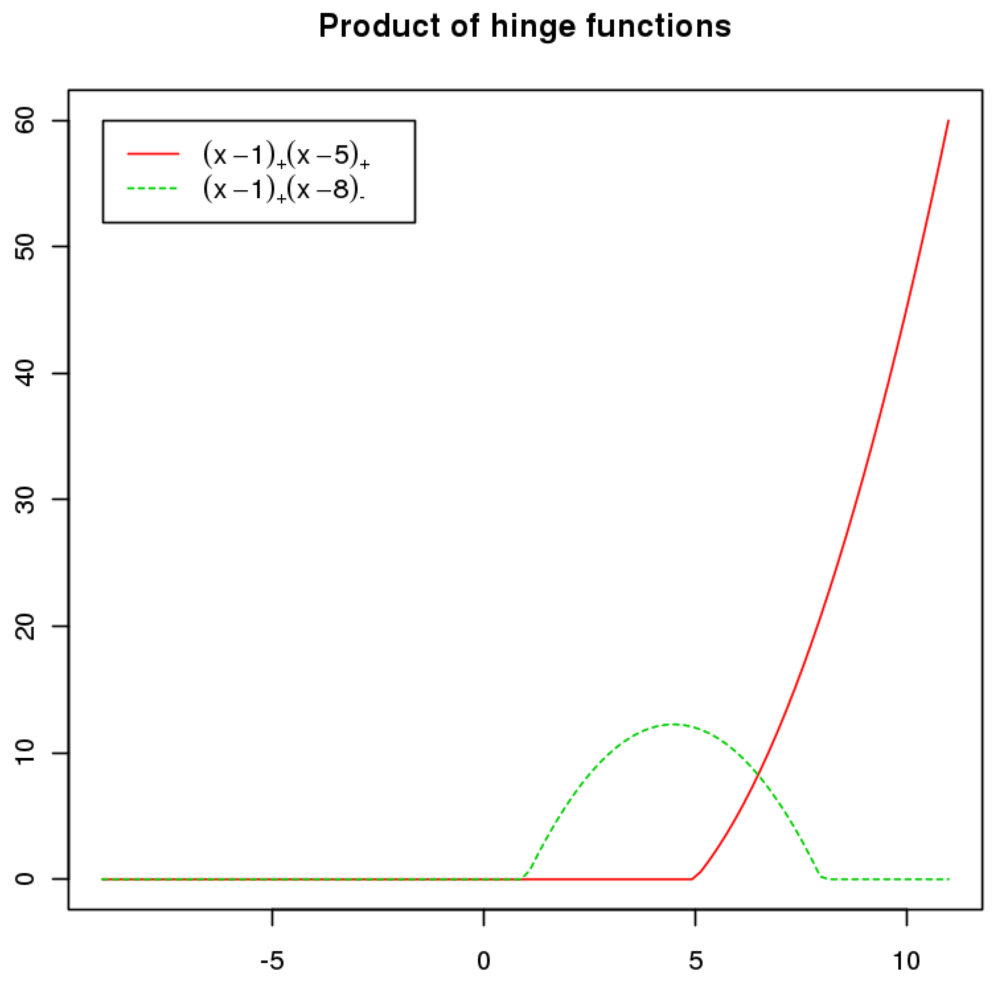
Non-linearity of MARS

Using the product of two (or more) hinge functions, MARS can model non-linear functions

$$(x-1)_+(x-5)_+ = \begin{cases} x^2 - 6x + 5 \text{ if } x > 5 \\ 0 \ \text{ otherwise} \end{cases}$$

$$(x-1)_+(x-8)_- = \begin{cases} -x^2 + 9x - 8 \text{ if } 1 < x < 8 \\ 0 \ \text{ otherwise} \end{cases}$$

In general, MARS only gets complicated over **small** regions.



MARS in a Nutshell (Model Structure)

A MARS model can be expressed as

$$\hat{y} = \hat{f}(x) = \sum_{k=1}^{K} \hat{\beta}_k h_k(x)$$

where $h_k(x)$ is either a

- **constant** (i.e., $h_k(x) = 1$)
- **hinge function** (e.g., $h_k(x) = (x - t)_+$)
- **product of hinge functions** (e.g., $h_k(x) = (x - t)_+(x - t')_+$)

MARS in a Nutshell (Model Building)

As is the case to CART, MARS iteratively add terms to its model. Once a stopping criterion is met, unwanted terms are removed.

Step 1: Let $j = 1$ (feature), $i = 1$ (observation), and $t = x_{ij}$ (knot).

Step 2: Fit $\hat{y} = \hat{f}(x) = \hat{\beta}_0 + \hat{\beta}_1 \left(x_{ij} - t\right)_+ + \hat{\beta}_2 \left(x_{ij} - t\right)_-$ and calculate SSE

Step 3: Repeat Step 2 for all $i = 1, \cdots, n$, and keep the model with smallest SSE

Take-Away: for a fixed j, find the knot t among the observations that produces the smallest SSE (i.e., for the j^{th} predictor variable, find the best split t at one of the observations x_{ij}).

MARS in a Nutshell (Model Building)

Step 4: Repeat Steps 2 and 3 for all $\mathrm{j} = 1, \cdots, p$. The model with the smallest SSE becomes our tentative model

Step 5: Repeat Steps 1 to 4 and add a pair of new hinge functions or product of hinge functions to the existing model

Step 6: Stop the model building phase when either the

a. maximum number of iteration has reached, or
b. reduction in SSE is not significant

Take-Away: repeat the process over all j, and find the best knot t among the observations that produces the smallest SSE overall. Repeat this process until “meaningful” terms cannot be added to the model anymore.

MARS in a Nutshell (Backward Phase)

As it is likely that the model built up to step 6 overfits the data, a backward elimination may be applied (WARNING!).

This step is based on criterion called **generalized cross-validation** (GCV), which is defined as:

$$GCV(\theta) = \frac{\sum_{i=1}^{n} \left(y_i - \hat{f}_\theta(x_i) \right)^2}{(1 - M(\theta)/n)^2}$$

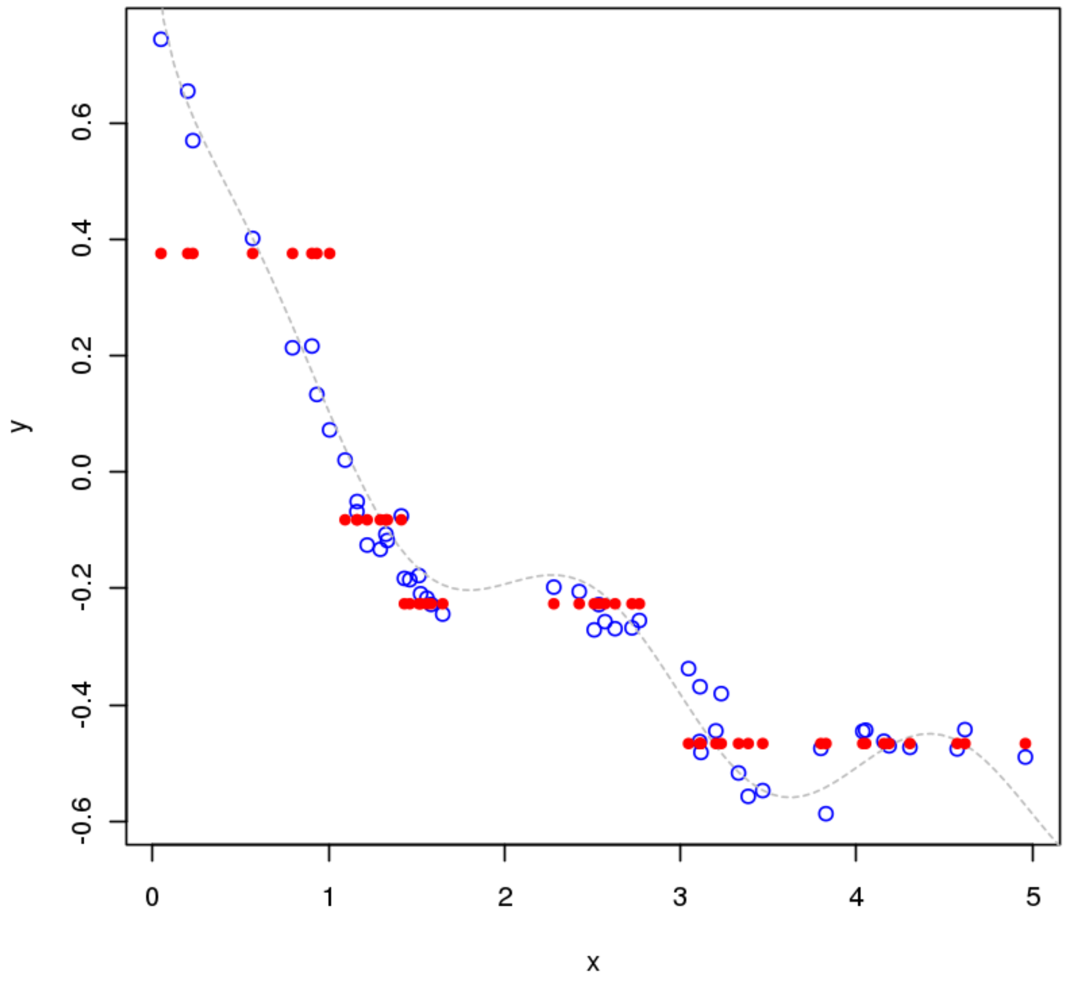
where θ denotes the number of terms used in reduced model, $M(\theta) = r + (2 \text{ or } 3)K$; r is the number of linearly independent hinge functions, and K is the number of knots

Illustrative Example

Recall that CART was not successful in describing the underlying model

$$y = \frac{\sin(\pi x)}{10} - \sqrt{x} + e^{x/10} + \varepsilon$$

Let's take a look at how MARS works with this dataset.



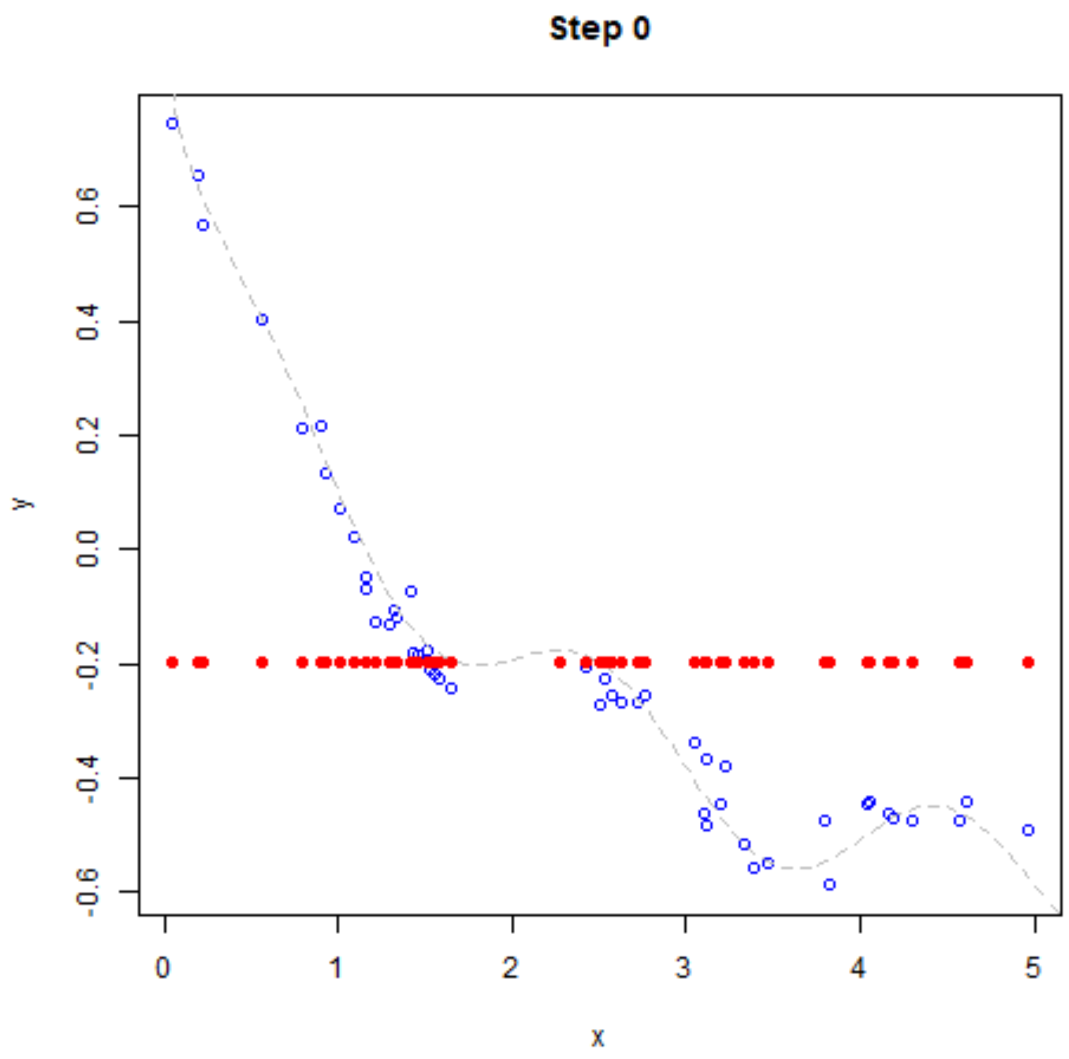
Illustrative Example (Step 0)

We start our model by fitting a constant term. We obtain

$$\hat{y}^{(0)} = -0.1983$$

with $SSE^{(0)} = 4.680$.

Clearly, this is not a great fit…



Illustrative Example (Step 1)

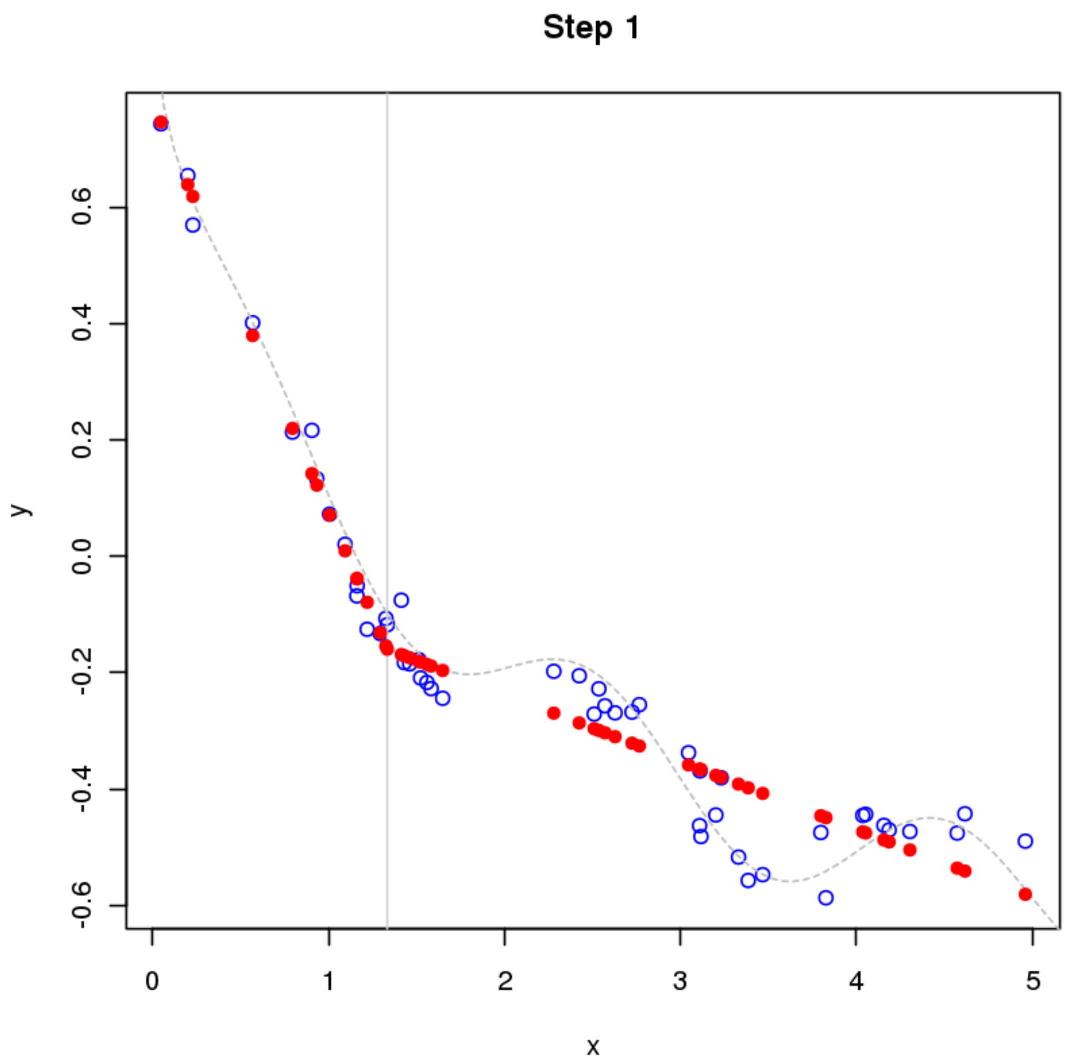
After the first pass, we find that

$$t = 1.334$$

provides the best fit with

$$SSE^{(1)} = 0.1937.$$

Term	Coefficient
Intercept	- 0.1597
$(x - 1.334)_+$	- 0.1162
$(x - 1.334)_-$	0.7045



After the first pass means that we proceed from $t = x_1$ to $t = x_n$ (i.e. from the smallest to largest observation), creating two hinge functions and fitting a linear regression for each knot $t = x_i$.

In this example, $t = 1.334$ provides the linear regression with smallest SSE among all knots.

Illustrative Example (Step 2)

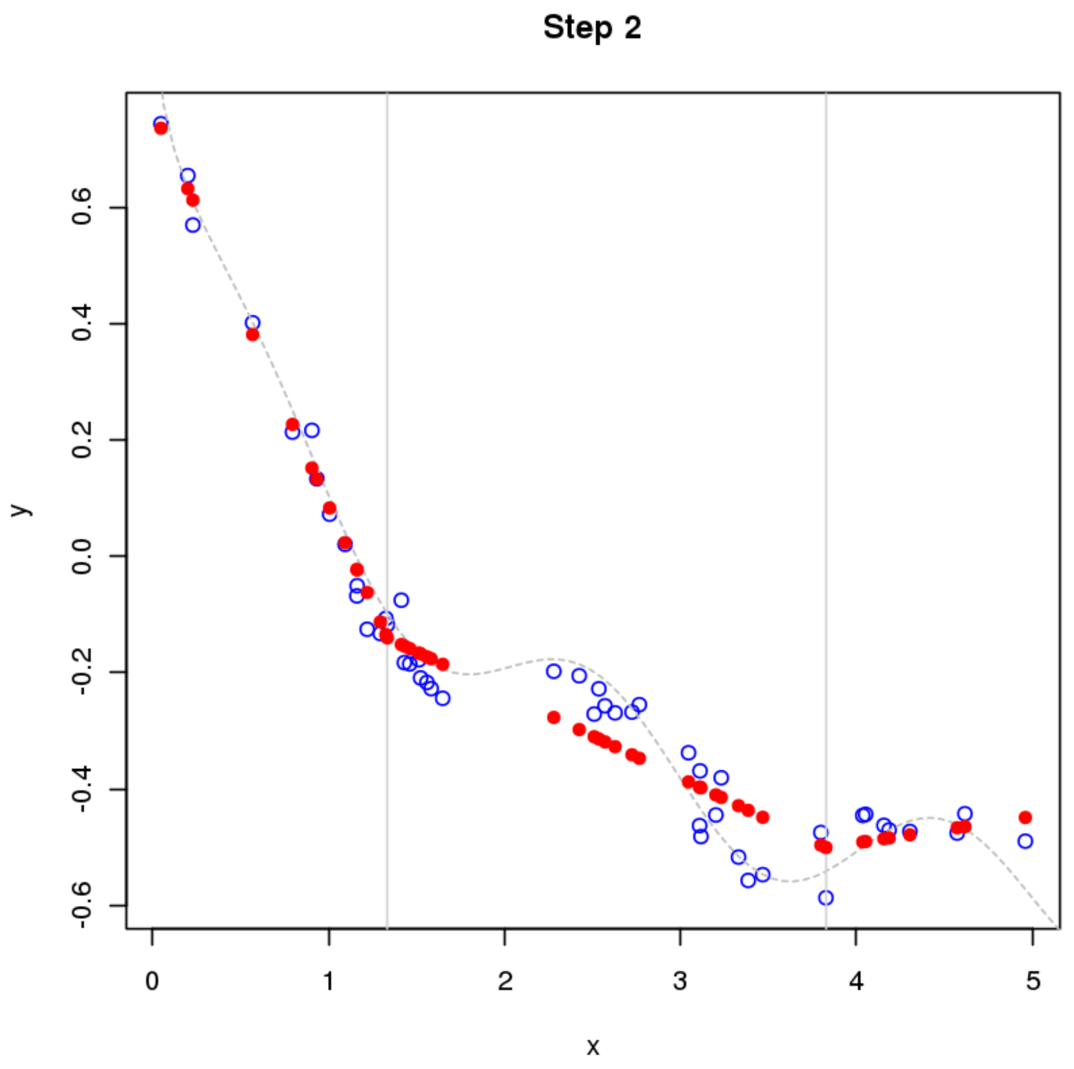
After the second pass, we find that

$$t = 3.827$$

provides the best fit with

$$SSE^{(2)} = 0.1423.$$

Term	Coefficient
Intercept	-0.1402
$(x - 1.334)_+$	0.1446
$(x - 1.334)_-$	0.6811
$(x - 3.827)_+$	0.1901



Given that $t = 1.334$ ha already been implemented in the model, what additional t (with associated hinge function) will boost the reduction in SSE?

In this example, it is $t = 3.827$.

We are only adding either a "+" or "-" hinge function (not a pair). This applies only because x is one-dimensional.

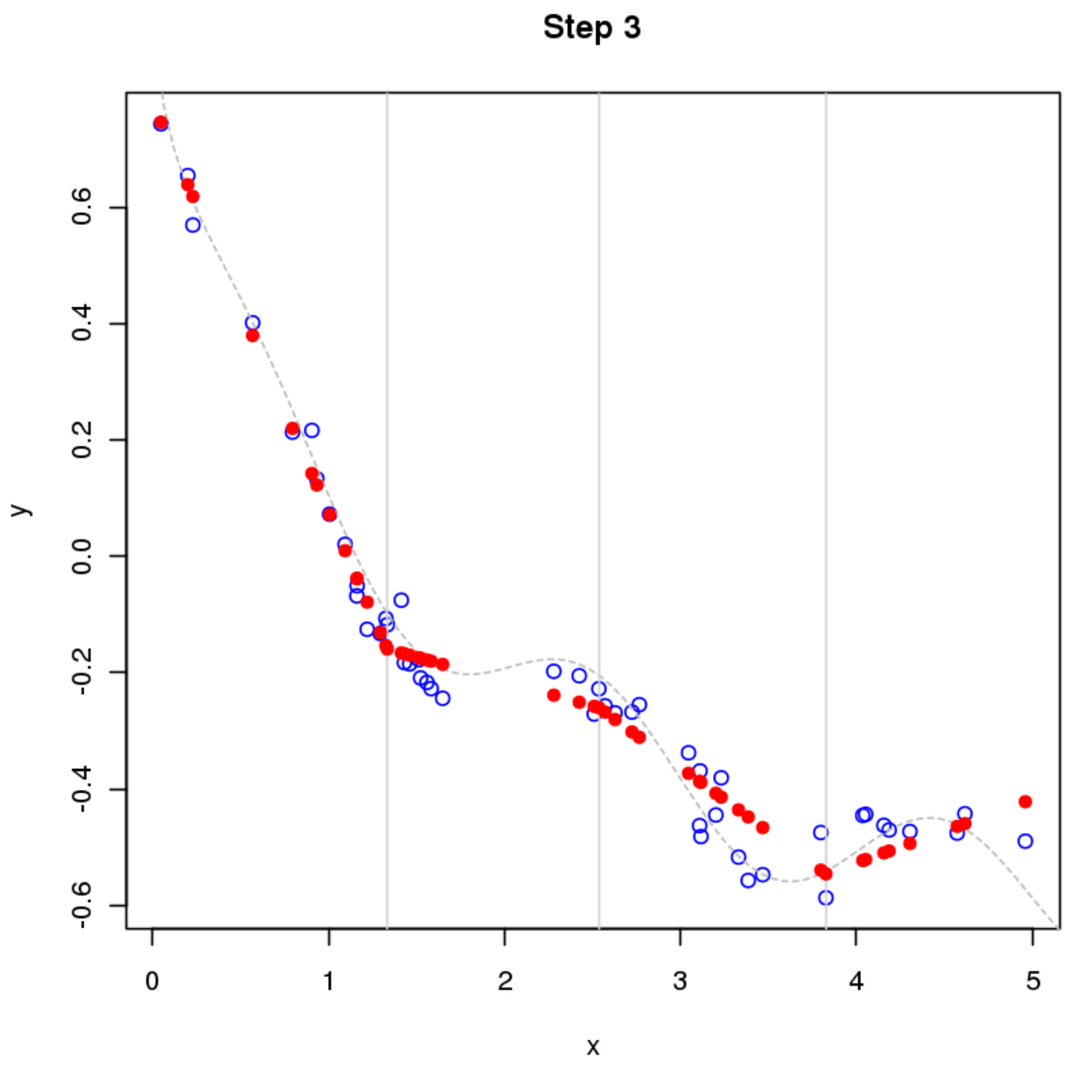
Illustrative Example (Step 3)

After the third pass, we find that

$$t = 2.537$$

provides the best fit with

$$SSE^{(3)} = 0.1132.$$



Illustrative Example (Step 4)

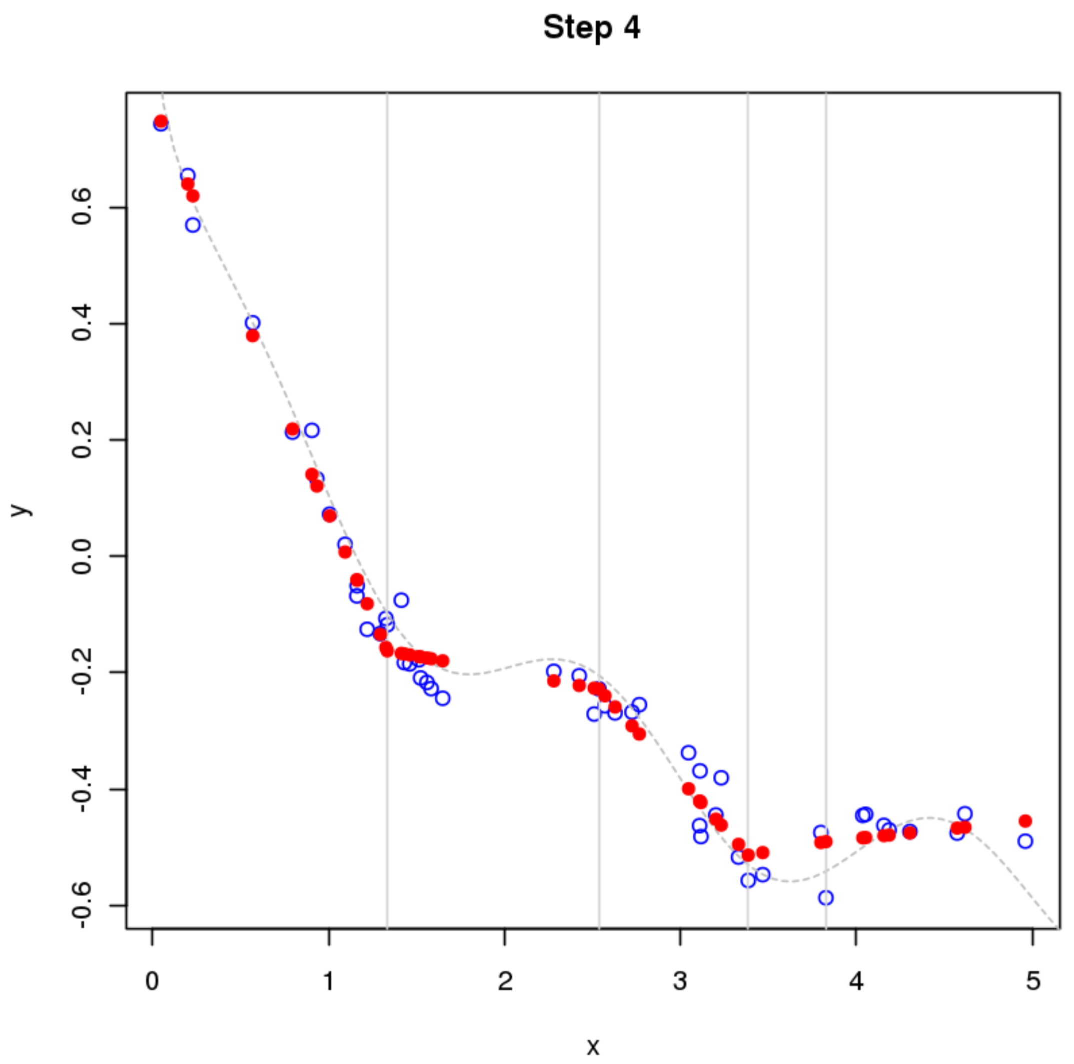
After the forth pass, we find that

$$t = 3.385$$

provides the best fit with

$$SSE^{(4)} = 0.0786.$$

At this point, we stop building our model as there is no t that significantly reduces SSE.

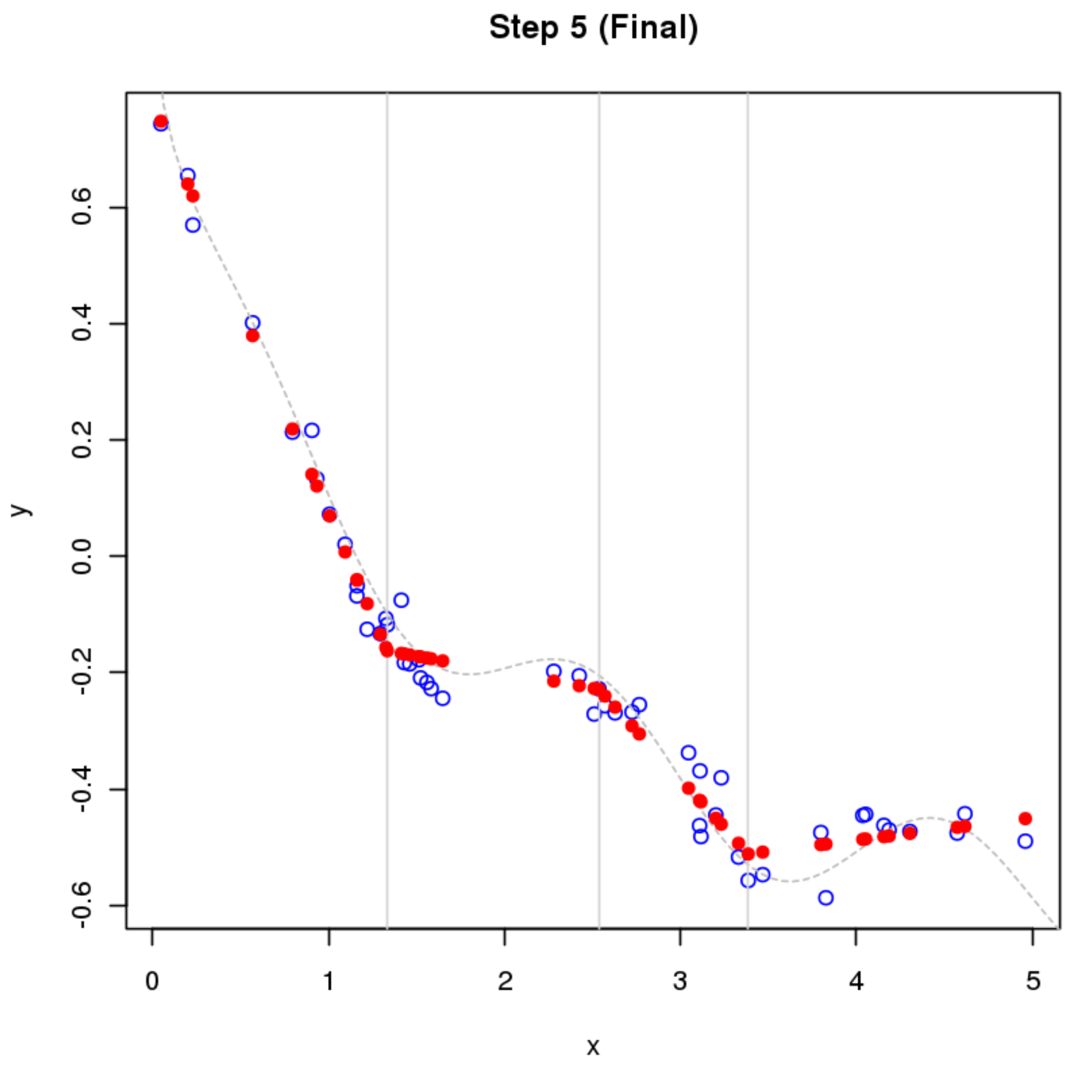


Illustrative Example (Step 5)

Using CGV, we remove one of the hinge functions $(x - 3.827)_+$

The final model has $SSE = 0.787$ with coefficients

Term	Coefficient
Intercept	-0.1625
$(x - 1.334)_+$	-0.0550
$(x - 1.334)_-$	0.7079
$(x - 2.537)_+$	-0.2785
$(x - 3.385)_+$	0.3721



So, while we cannot interpret the MARS model as combination of sine function, square-root, and exponential function… the model still fits the data fairly well. Not bad for linear combinations!

Prediction of Housing Prices – Dataset

Sale prices for $n = 1{,}460$ homes in Ames, Iowa, along with 79 attributes on each houses. These attributes cover almost every aspect of a home potential buyers might like to know about before purchasing:

- Type of utilities available
- Type of dwelling
- Overall condition of the house
- Dates of original construction and remodel
- Total square feet of each floor and lot size
- Conditions of driveway and garage
- etc.

[https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data]

Dealing with Missing Observations

Unlike the function `rpart()` from package `rpart`, neither `mda`'s `mars()` or `earth`'s `earth()` can handle missing values.

There are 2 major types of missing value patterns:

- **missing in majority of cases:** some variables are missing in between 18% to 96% of the cases, with no suitable logical surrogates.
- **features unavailable or N/A:** some houses do not have a garage or a basement and so variables such as *Quality of Garage* and *Basement Condition* are missing.

Dealing with Missing Observations

For variables with a majority of the items missing, we simply perform variable deletion.

For variables with unavailable features, we consider two approaches:

a. removing columns with missing observations (i.e., complete column-wise deletion)
b. removing rows with missing items (i.e., list-wise deletion)

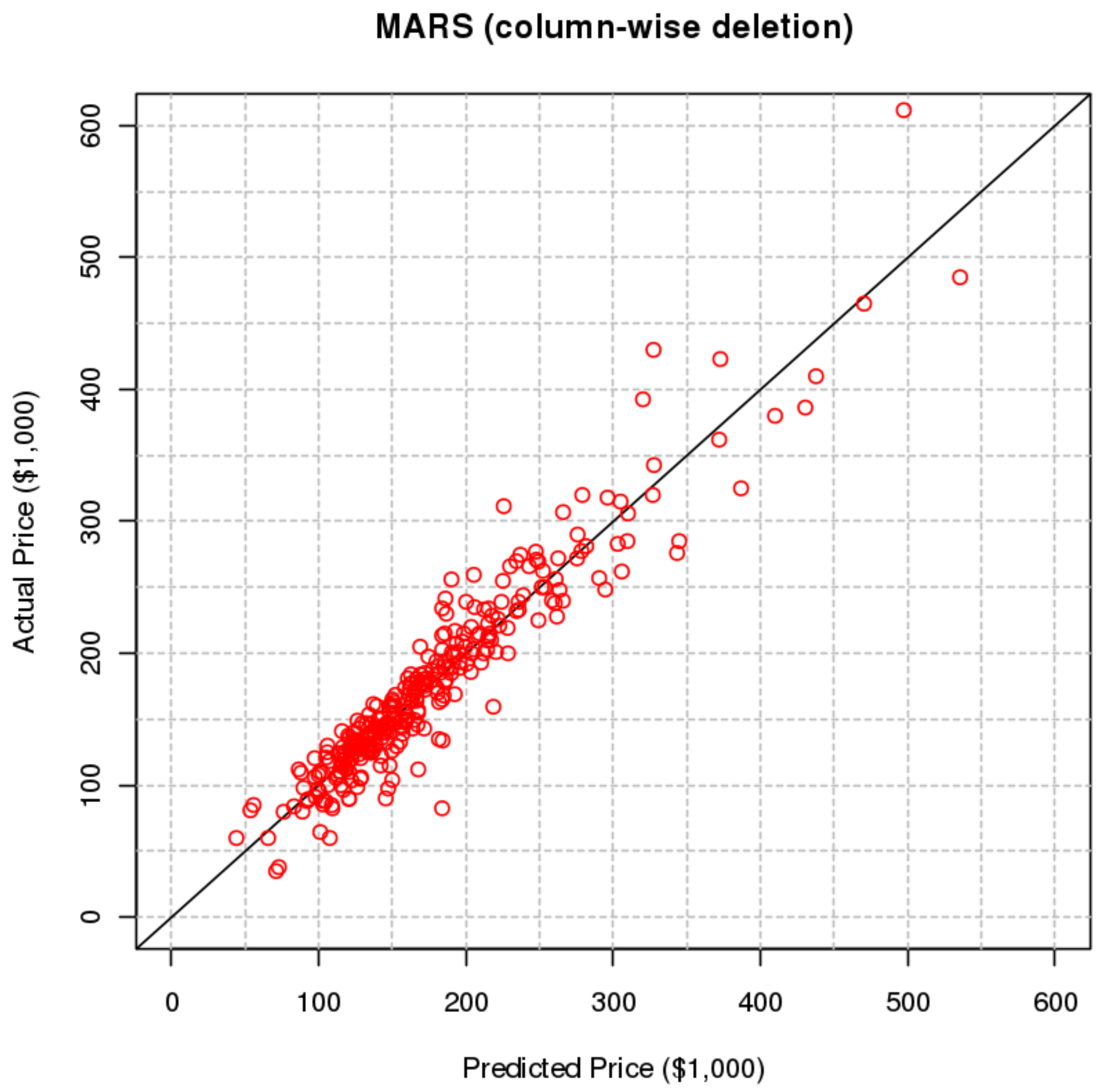
MARS with Complete Columns

The first approach is to remove all variables with at least one missing observations.

This leaves us with 61 explanatory variables (rather than 79). However, we still have all $n = 1{,}460$ observations to train and test the model (with $n_{tr} = 1{,}160$ cases in training, and the remaining $n_{test} = 300$ cases for testing).

MARS with Complete Columns

```
> fit.mars2 <- earth(SalePrice~., data=dat.train, degree=2)
> summary(fit.mars2)
Call: earth(formula=SalePrice~., data=dat.train, degree=2)

                                         coefficients
(Intercept)                                305.237326
h(14892-LotArea)                            -0.002288
h(LotArea-14892)                             0.000482
h(7-OverallQual)                           -24.721032
h(OverallQual-7)                            36.814607
h(2004-YearBuilt)                           -0.141857
h(YearBuilt-2004)                            8.596582
h(1518-BsmtFinSF1)                          -0.020161
h(TotalBsmtSF-483)                           0.024087
h(X1stFlrSF-2113)                           -0.191760
h(2945-GrLivArea)                           -0.061113
h(GrLivArea-2945)                           -0.172599
h(2-GarageCars)                             -5.901028
h(GarageCars-2)                             31.002837
NeighborhoodCrawfor * h(2945-GrLivArea)      0.013660
NeighborhoodStoneBr * h(YearBuilt-2004)     18.098359
Condition1Norm * h(GrLivArea-2945)           0.301483
Condition2PosN * h(TotalBsmtSF-483)         -0.054718
Exterior1stBrkFace * h(TotalBsmtSF-483)      0.019778
ExterQualTA * h(TotalBsmtSF-483)            -0.015652
h(OverallQual-7) * h(X1stFlrSF-1746)         0.038738
h(OverallQual-7) * h(1746-X1stFlrSF)        -0.026056
h(7-OverallQual) * h(2945-GrLivArea)         0.010799
h(OverallQual-7) * h(2945-GrLivArea)        -0.024288
h(OverallQual-7) * h(TotRmsAbvGrd-9)        19.422241
h(OverallQual-7) * h(9-TotRmsAbvGrd)         8.504301
h(5-OverallCond) * h(YearBuilt-2004)        -4.215653
h(OverallCond-5) * h(YearBuilt-2004)        17.551782
h(7-OverallCond) * h(2004-YearBuilt)        -0.147613
h(2004-YearBuilt) * h(BsmtUnfSF-1198)       -0.001882
h(2004-YearBuilt) * h(1198-BsmtUnfSF)       -0.000159
h(YearBuilt-2004) * h(1-BsmtFullBath)       -5.918784
h(GarageCars-2) * h(98-OpenPorchSF)         -0.300027

Selected 33 of 39 terms, and 19 of 188 predictors
Termination condition: RSq changed by less than 0.001 at 39 terms
Importance: OverallQual, GrLivArea, OverallCond, YearBuilt, Condition1Norm, ...
Number of terms at each degree of interaction: 1 13 19
GCV 422.0449    RSS 423587.9    GRSq 0.9341662    RSq 0.9429409
```



This model has:

- 1 intercept term
- 13 individual hinge functions
- 19 interaction terms

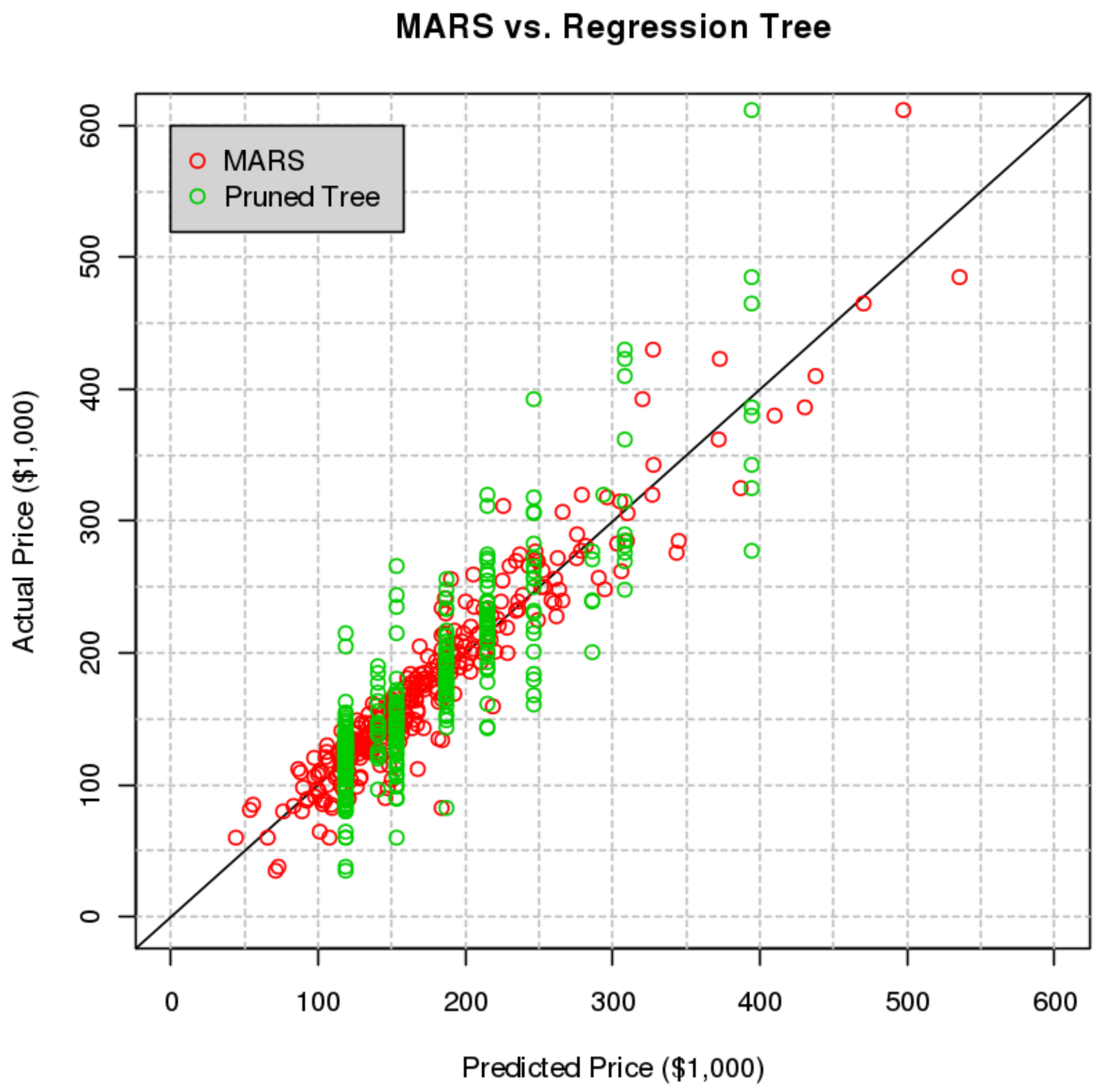
for a total of 33 terms

It seems that the variability is slightly larger when the Actual Price is 350k+ (but again, sample size for such a region is relatively small)!

MARS vs. CART

As seen on the plot, the MARS predictions are much tighter than CART's, meaning that MARS outperforms CART in terms of prediction due to the continuous nature of the prediction surface.

But CART is easier to interpret.


MARS vs. Regression Tree
MARS
Pruned Tree
Actual Price ($1,000)
Predicted Price ($1,000)