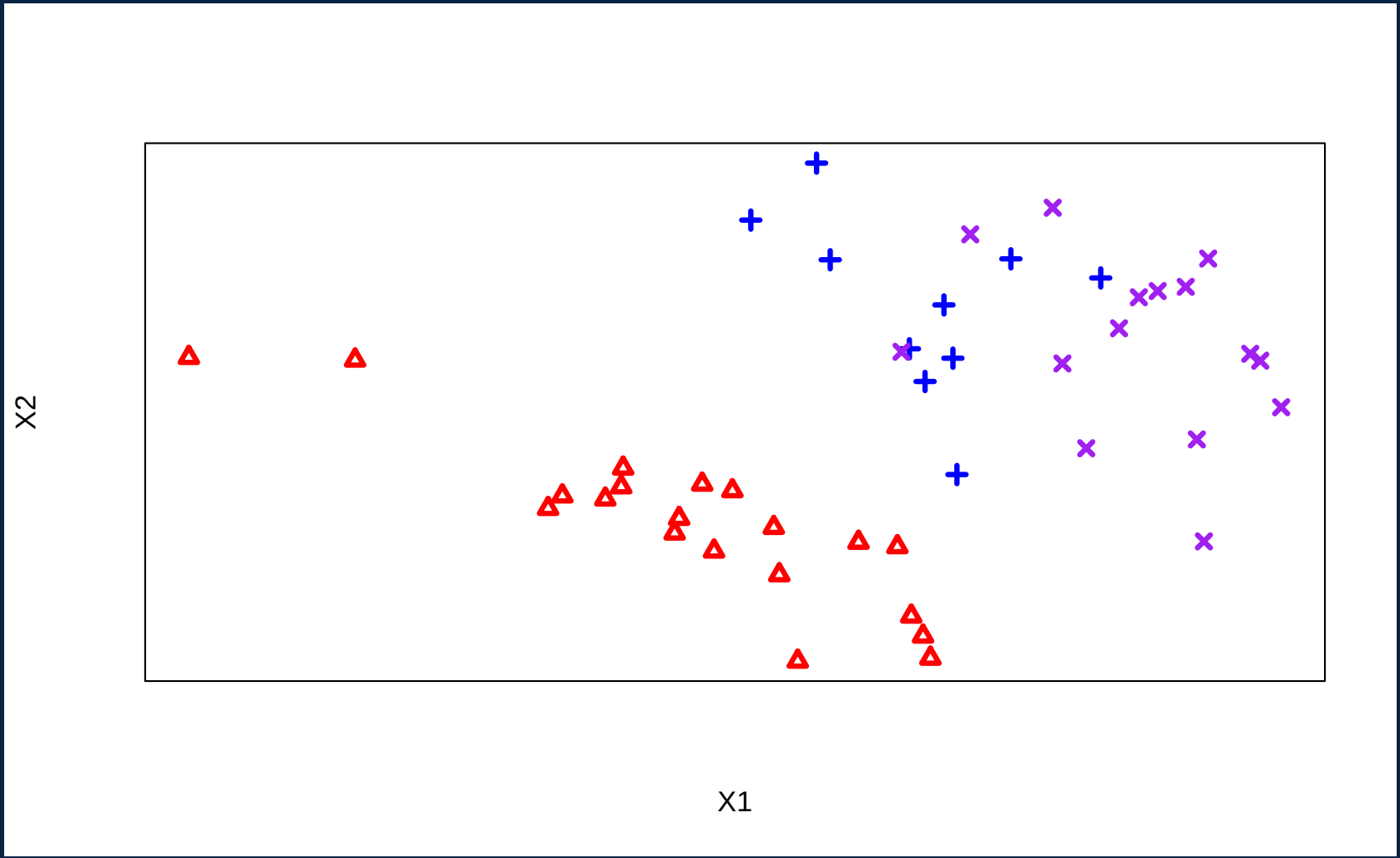
Classification and Regression Trees

UBCO MDS — DATA 571

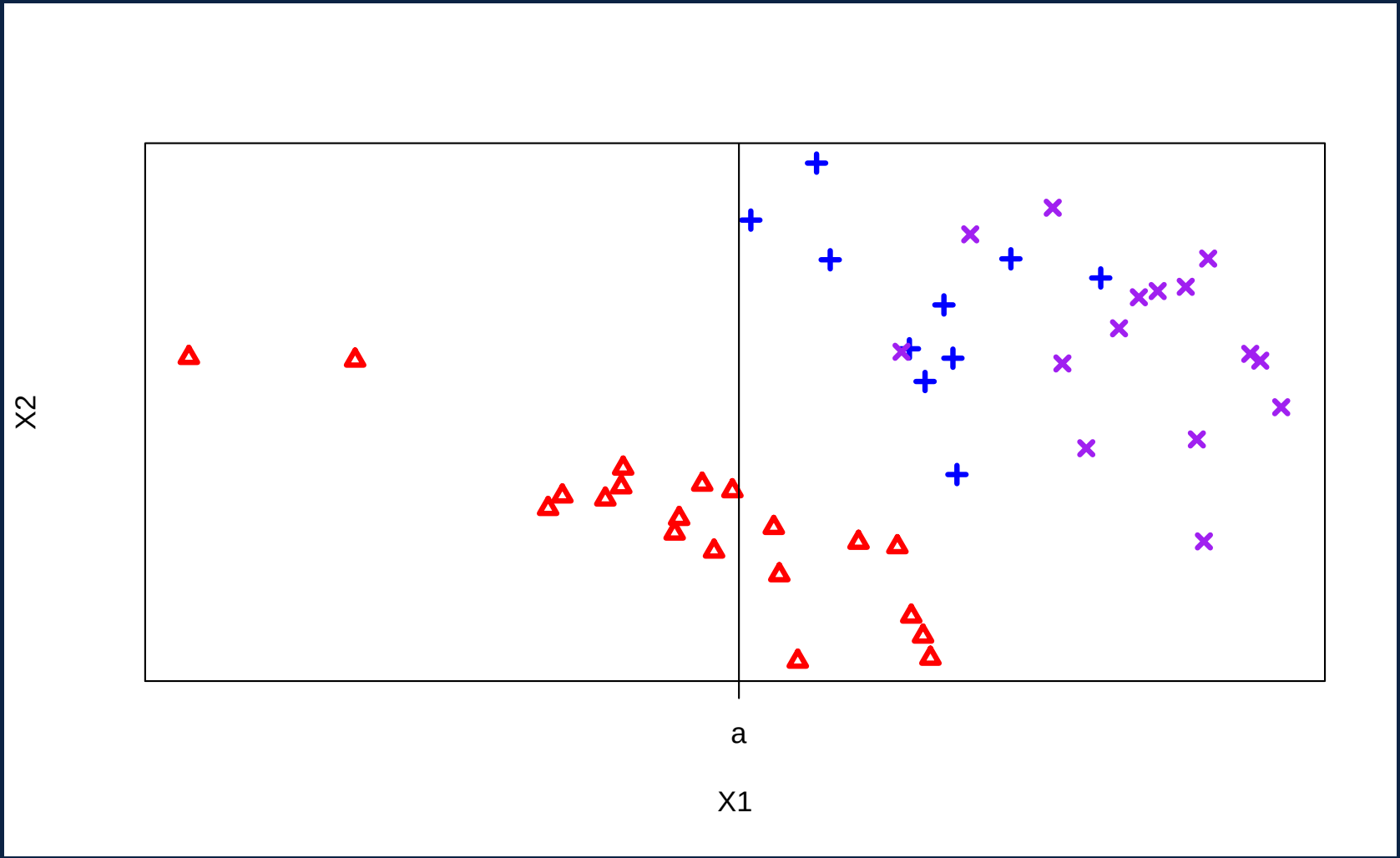
Motivation

- Decision trees (or CARTs) partition the predictor space by simple binary splitting rules. Think 'flow chart'.
- They have both advantages and disadvantages over other methods...but first, let's motivate their idea.

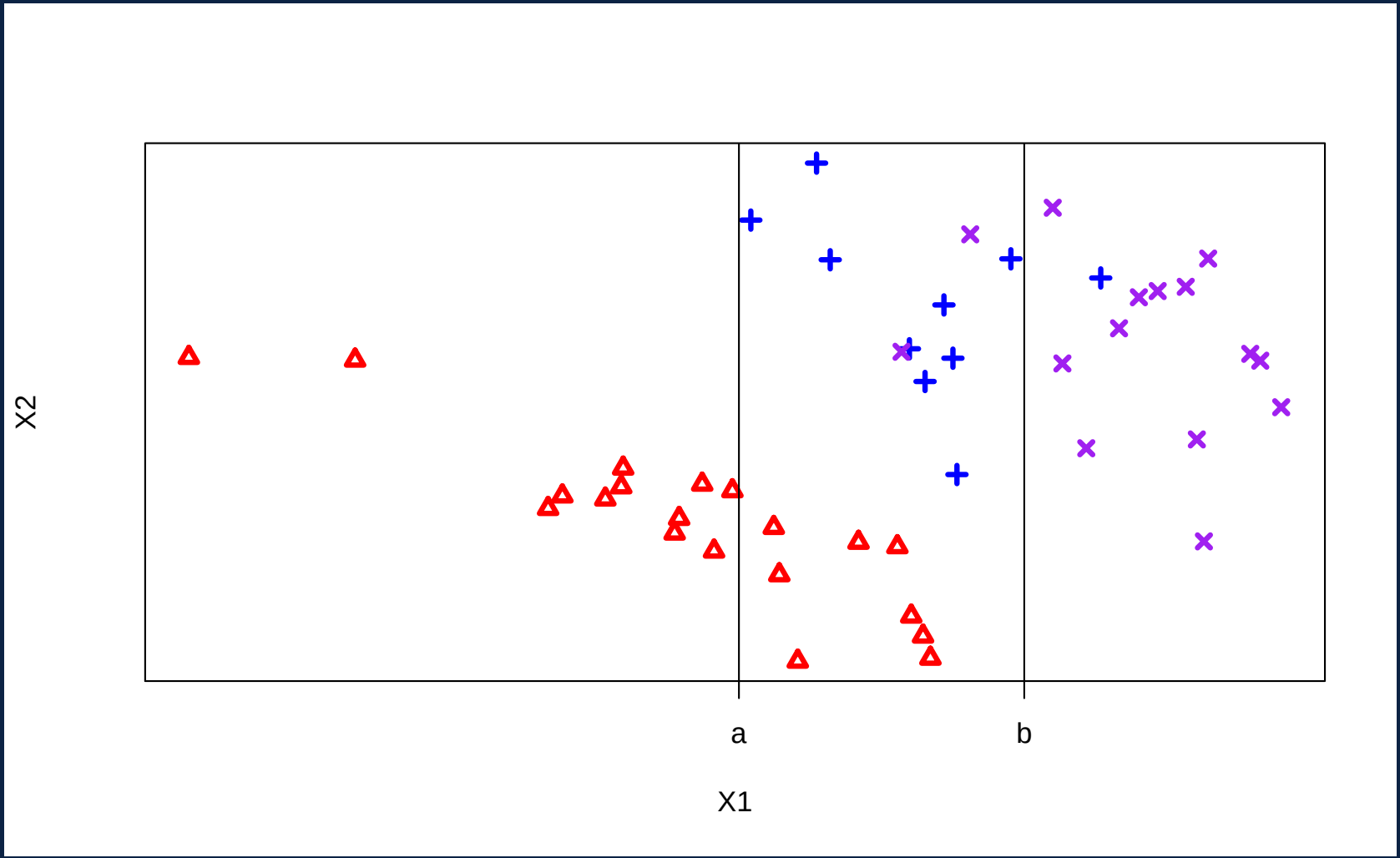
Motivating Example: Classification



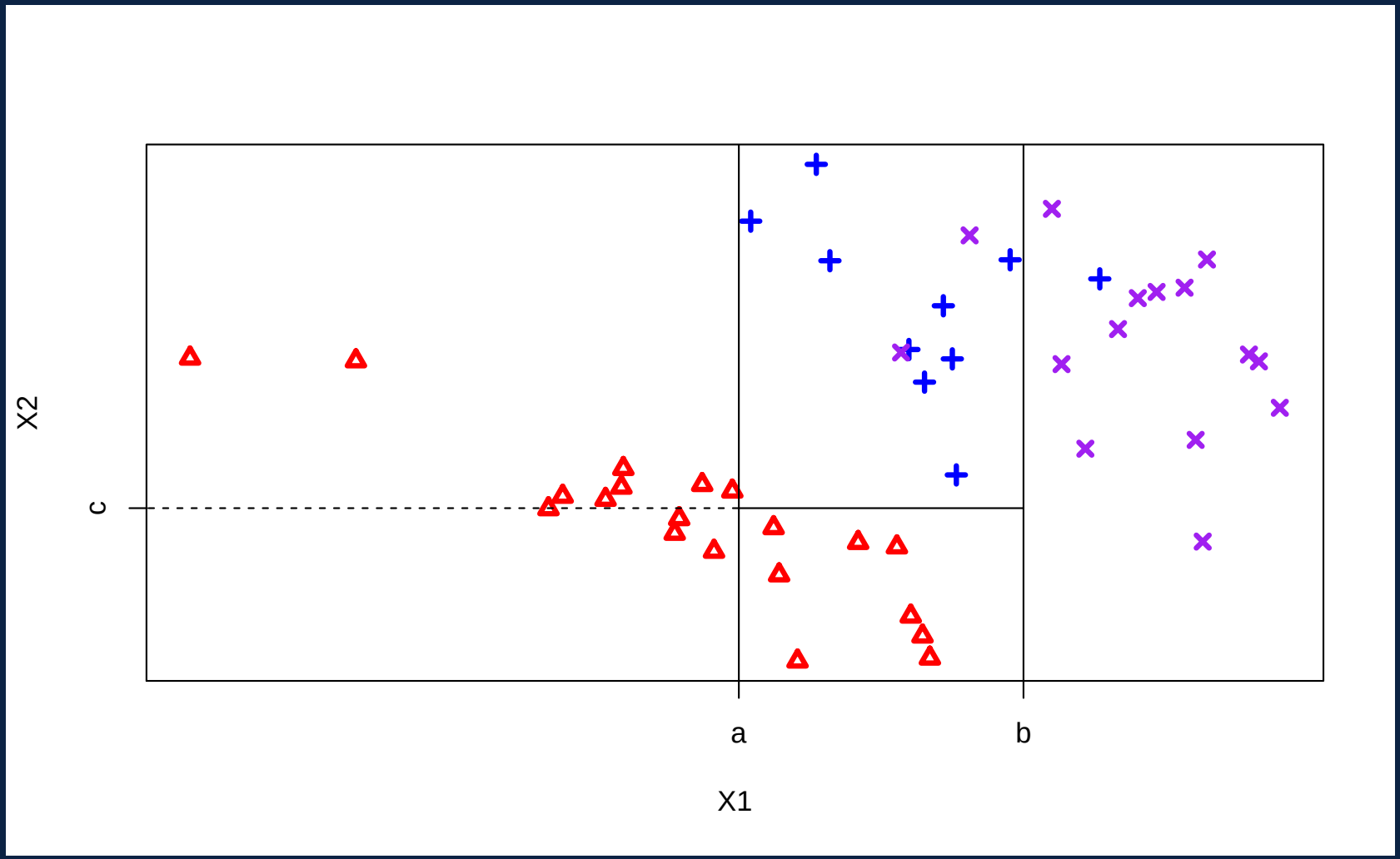
Motivating Example: Classification



Motivating Example: Classification



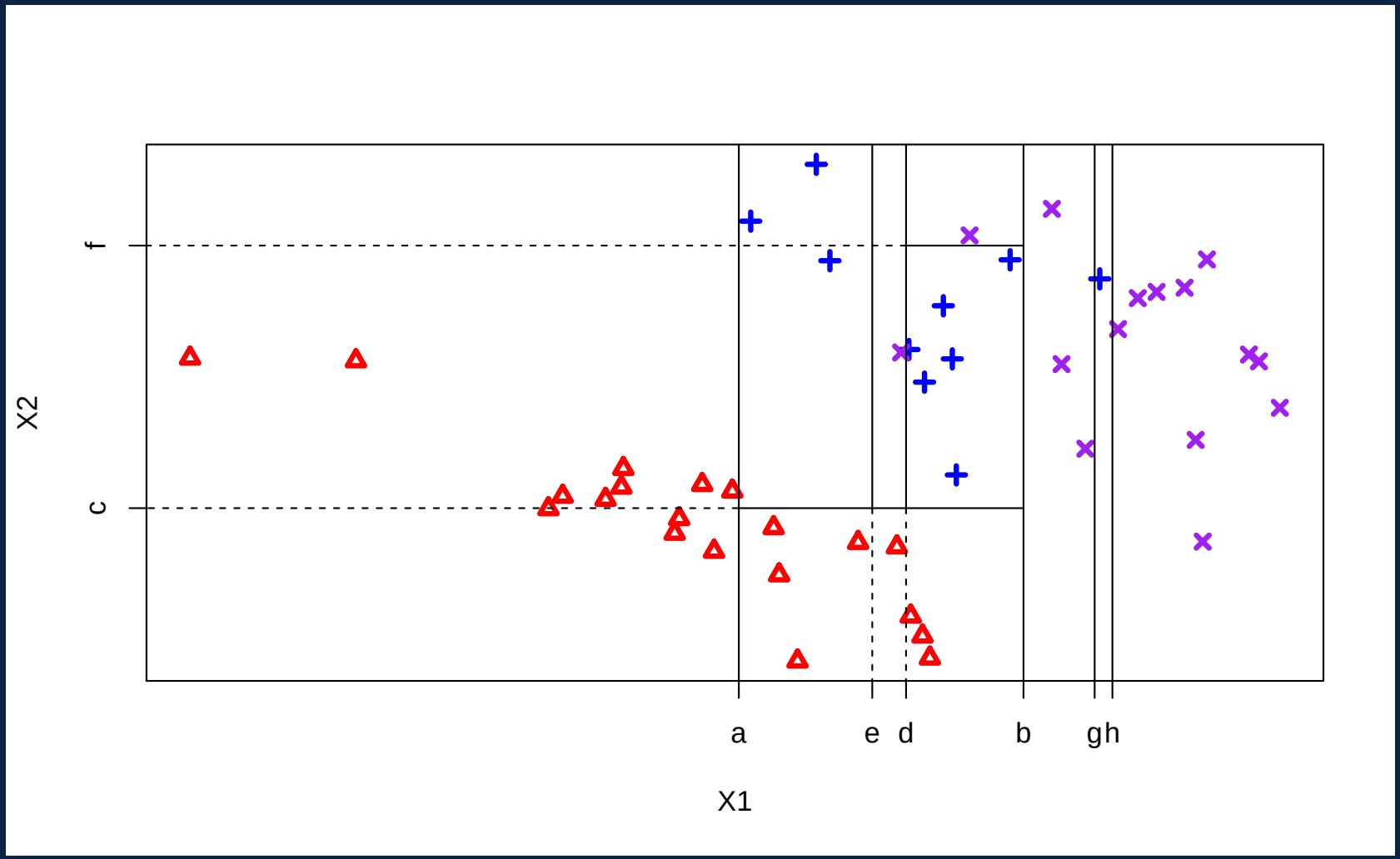
Motivating Example: Classification



Motivating Example: Classification

- Are we done with 3 splits?
- If no then maybe...

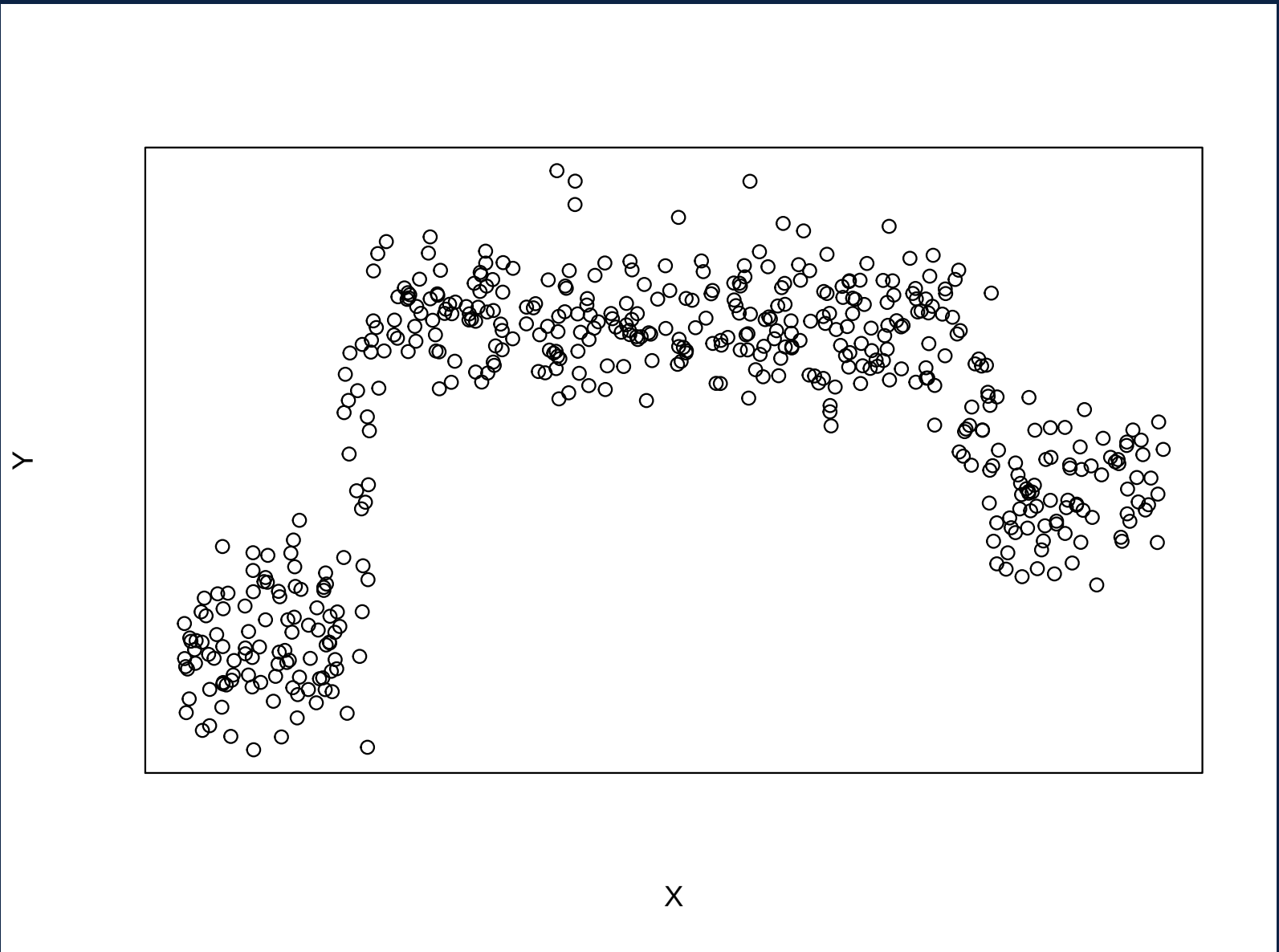
Motivating Example: Classification



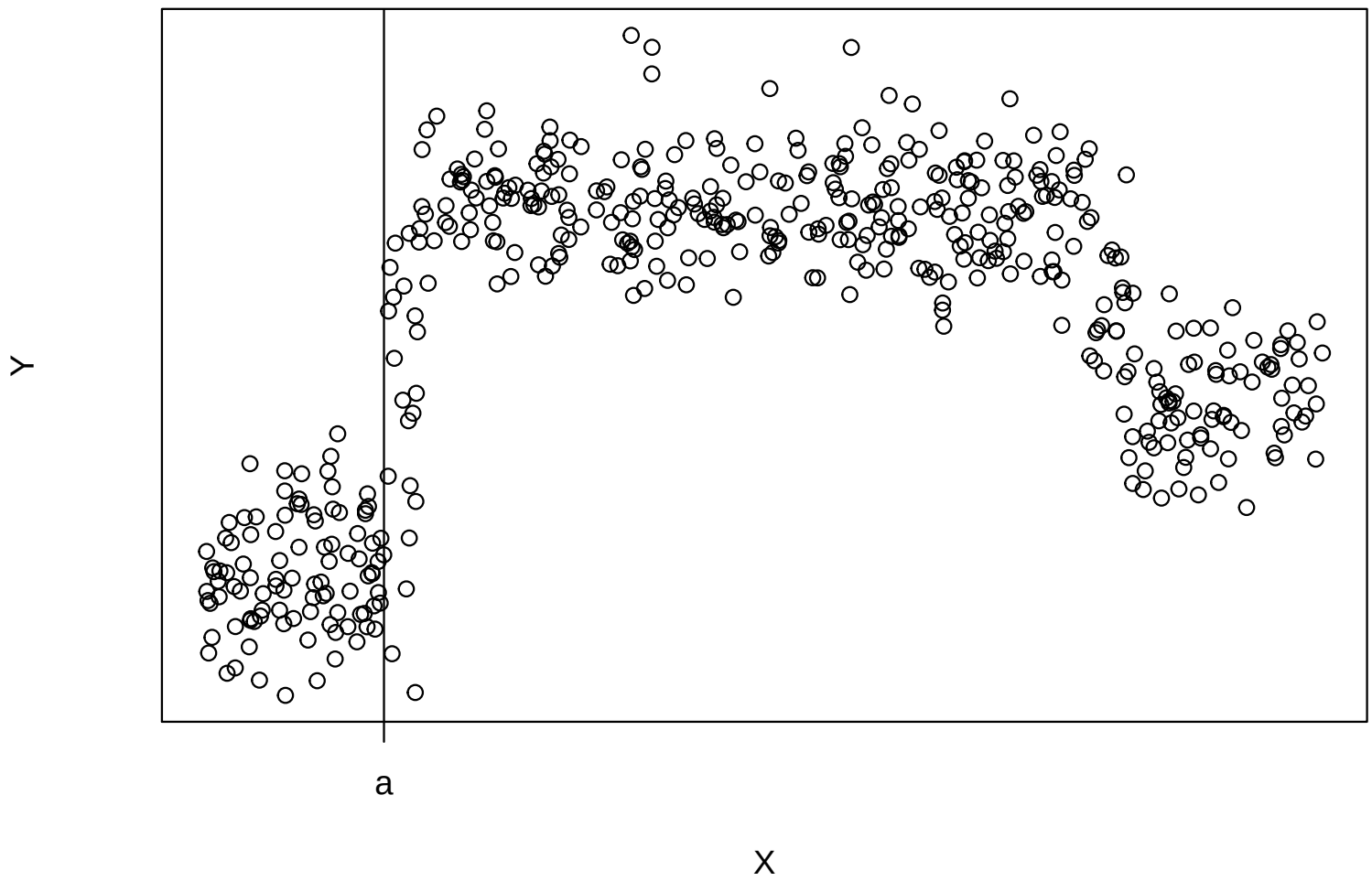
Motivation

- So, it appears there are some things we may need to consider going forward in the classification realm.
- Before diving further into specifics, let's motivate trees for regression through an example.

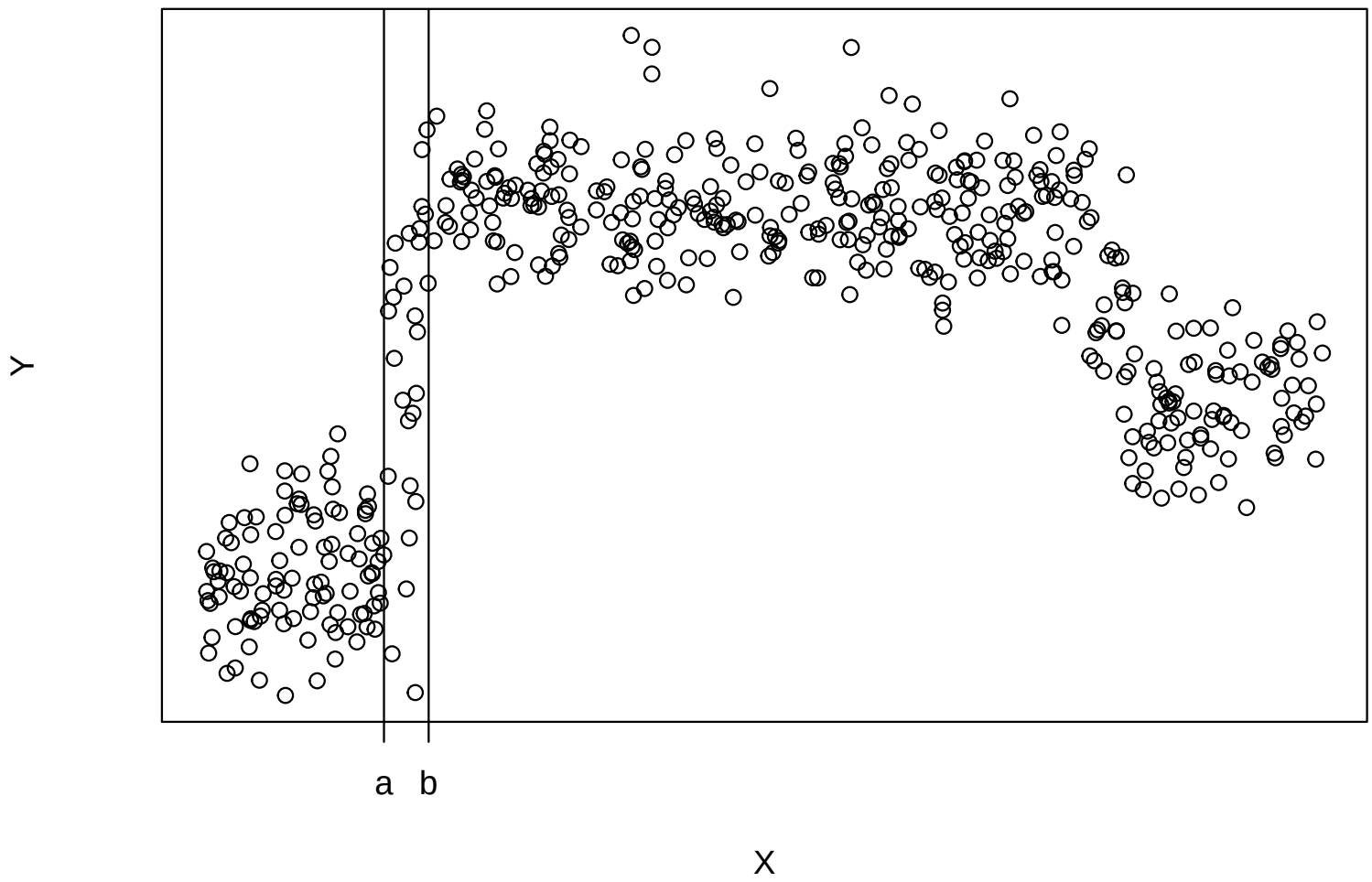
Motivating Example: Regression



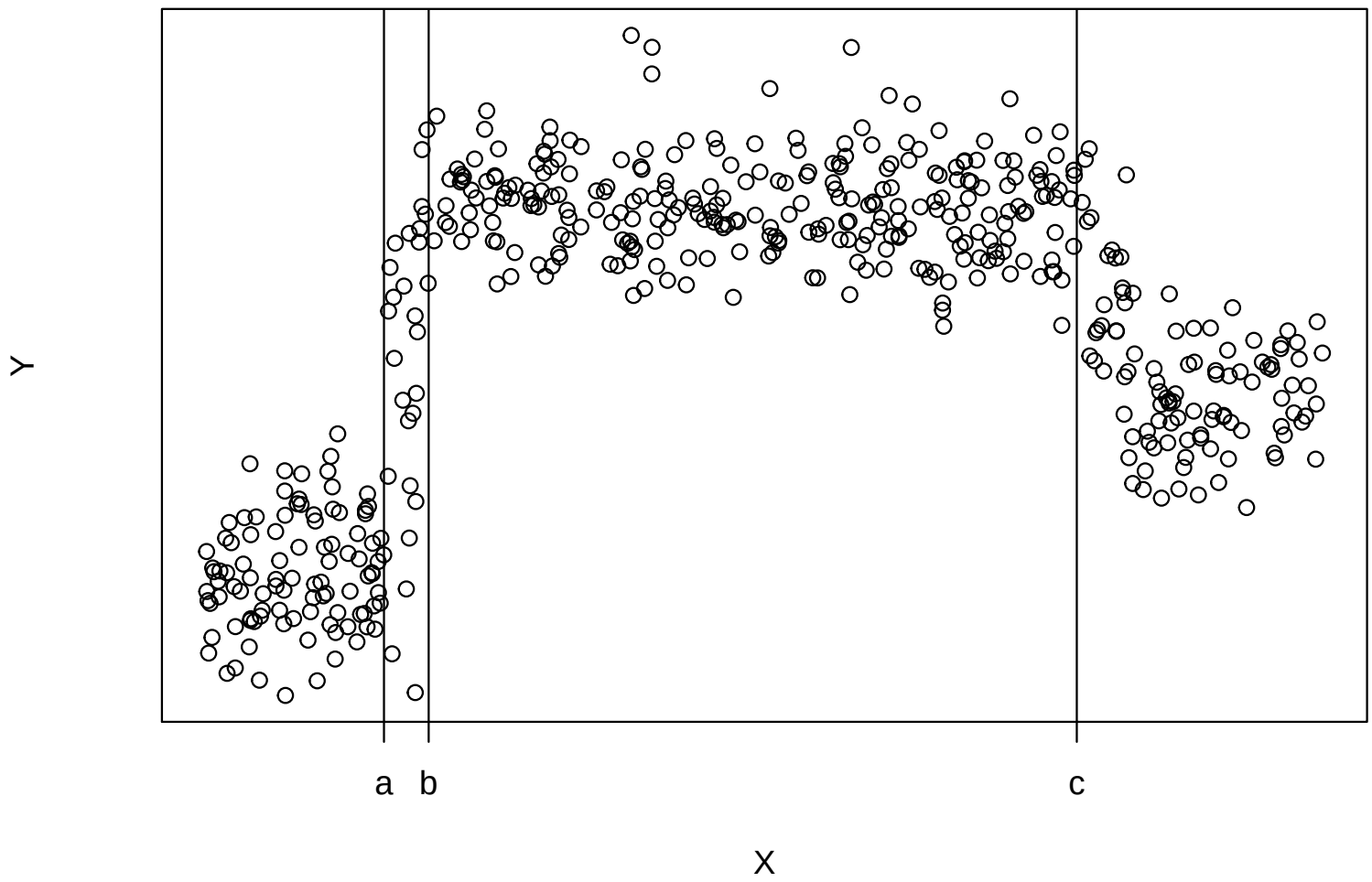
Motivating Example: Regression



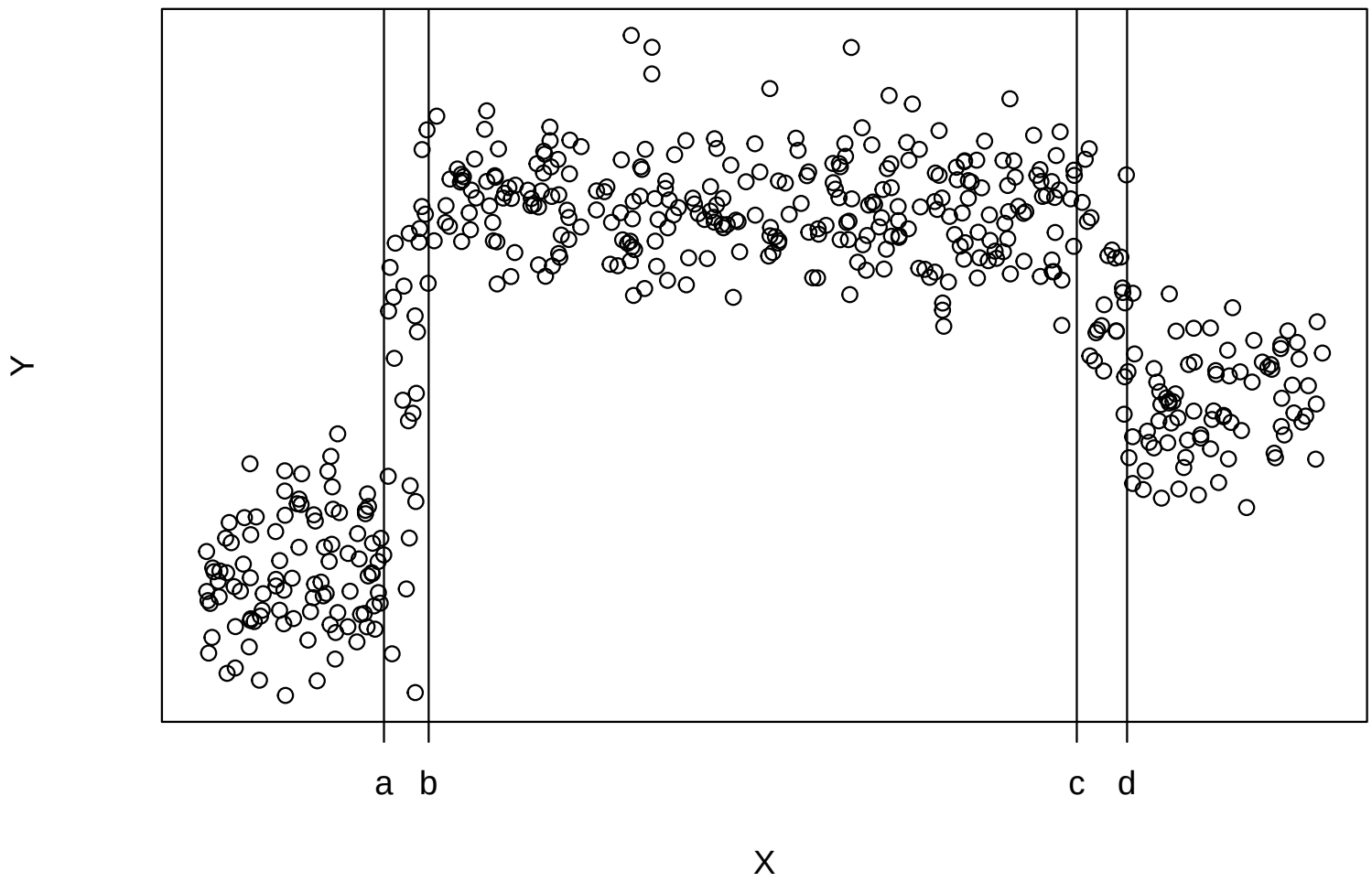
Motivating Example: Regression



Motivating Example: Regression



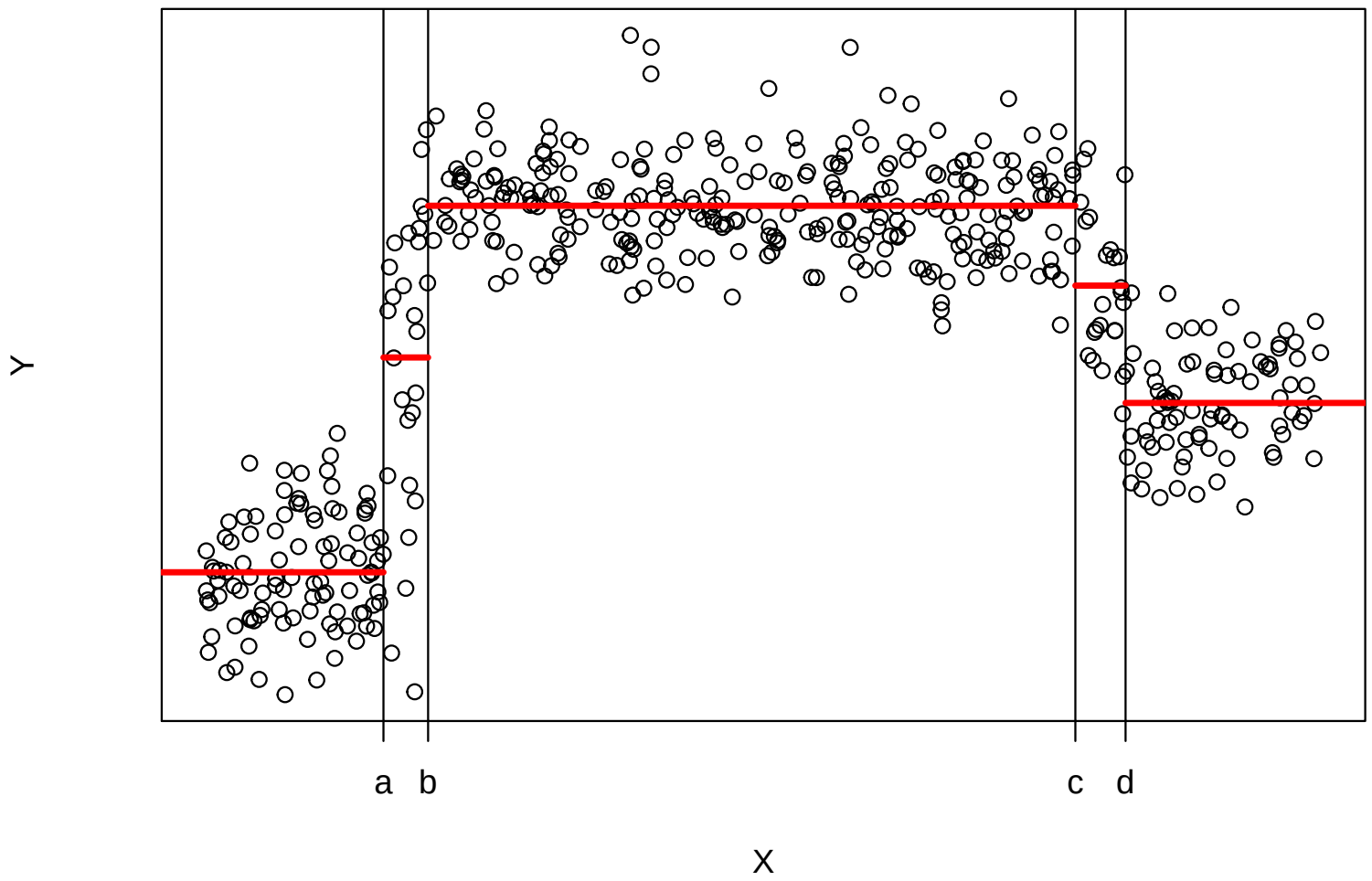
Motivating Example: Regression



Motivation

- Now that we have regions of relative homogeneity in Y...what would the simplest plan be?
- Average the observations in the region to provide the predicted value...

Motivating Example: Regression



Growing a Tree

- Goals:
  - Divide the predictor space into J non-overlapping regions $R_1, R_2, \ldots, R_J$. Using binary splits, these regions are hyper-rectangles.
  - Regression: Predict the mean response for the observations that fall into each region R_j from the training set.

    Classification: Predict the categorical response as the most commonly occurring in each region R_j from the training set.

Growing a Tree

- How do we 'best' partition the predictor space?
- For regression: find binary splits which minimize the RSS

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

- For classification: minimizing the misclassification rate is not usually done at the growth stage. Usually the Gini index, deviance, or cross-entropy is minimized. Most of the time, it won't make a large difference which is chosen. Here's Gini for each node m, where K is the number of response classes...

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

Growing a Tree - Regression

- We run into a common computational hurdle here! We cannot consider all possible hyper-rectangles in our data-set to find the optimal tree...computationally impossible as dimensionality increases.

- Instead, we can start seek variable j and cutpoint s such that

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

- Then, via the same process, we subdivide region R_1 into R_3 and R_4. Then we subdivide R_2 into R_5 and R_6, etc...until some stopping criteria is met (for example, each region contains a minimum of 3 observations).

Problem

- Building a tree through the description just given will tend to heavily overfit the training data.
- A common strategy is to grow a large tree, and then prune it back using cross-validation and a cost-complexity tuning parameter.

Pruning

- We introduce a non-negative tuning parameter α.
- For every value of α there is a subset (T) of our large tree (T_0) such that

$$\sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

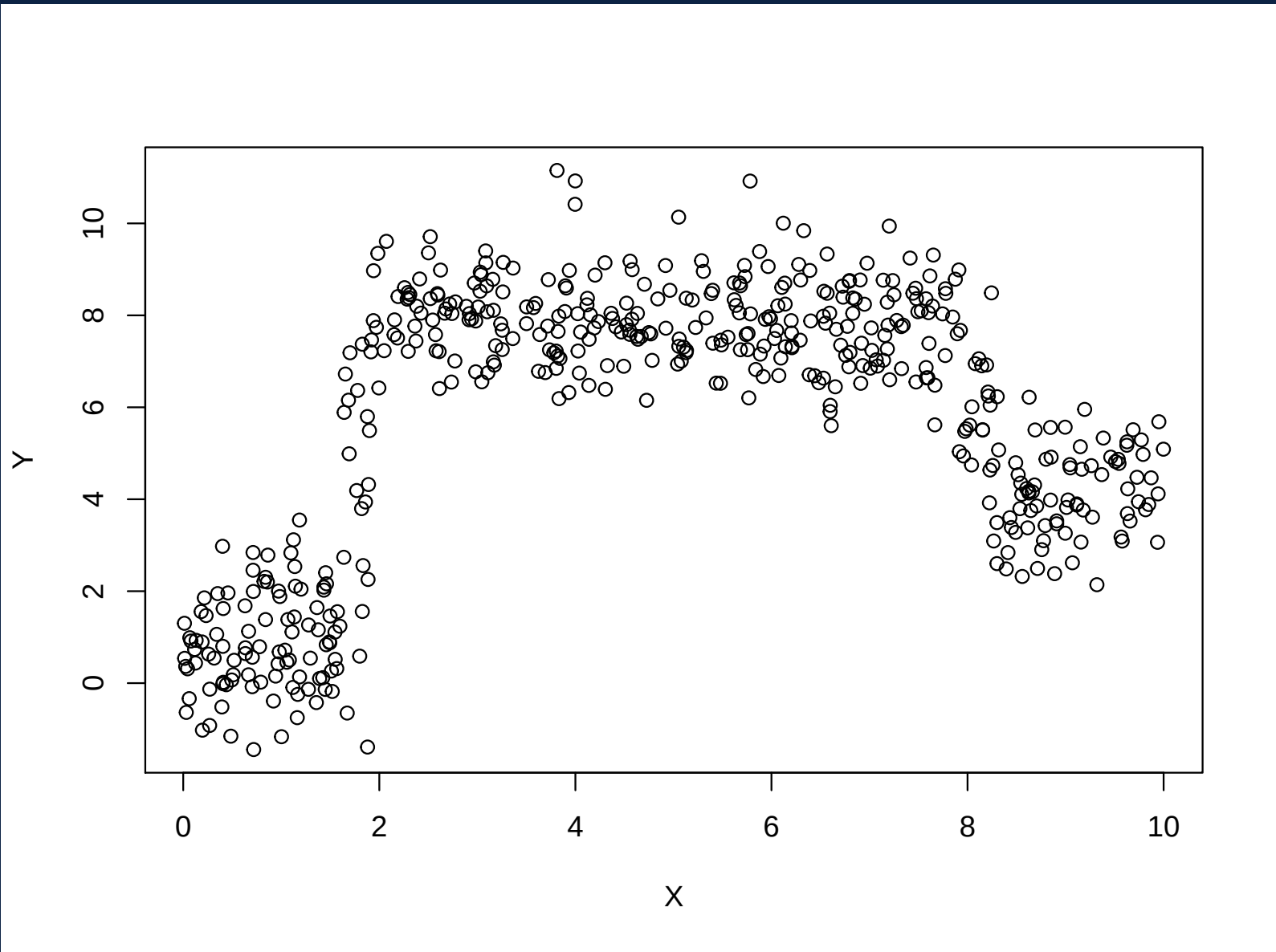
  is minimized. Where $|T|$ is the number of terminal nodes in the tree.
- Using k-fold cross validation, we both build trees, and prune them back for possible values of α. Each tree and subtree built in this fashion will produce an estimate of the MSE.
- Pick the α that minimizes the predicted MSE using the above cross-validation. Then prune the original tree (built on the full data) with the chosen α.

Classification Trees

- For classification trees, the process is very much the same.
- We use the previously mentioned Gini index for choosing splits (growing the tree), and often use the misclassification rate during the pruning stage (again with cost-complexity tuning parameter).

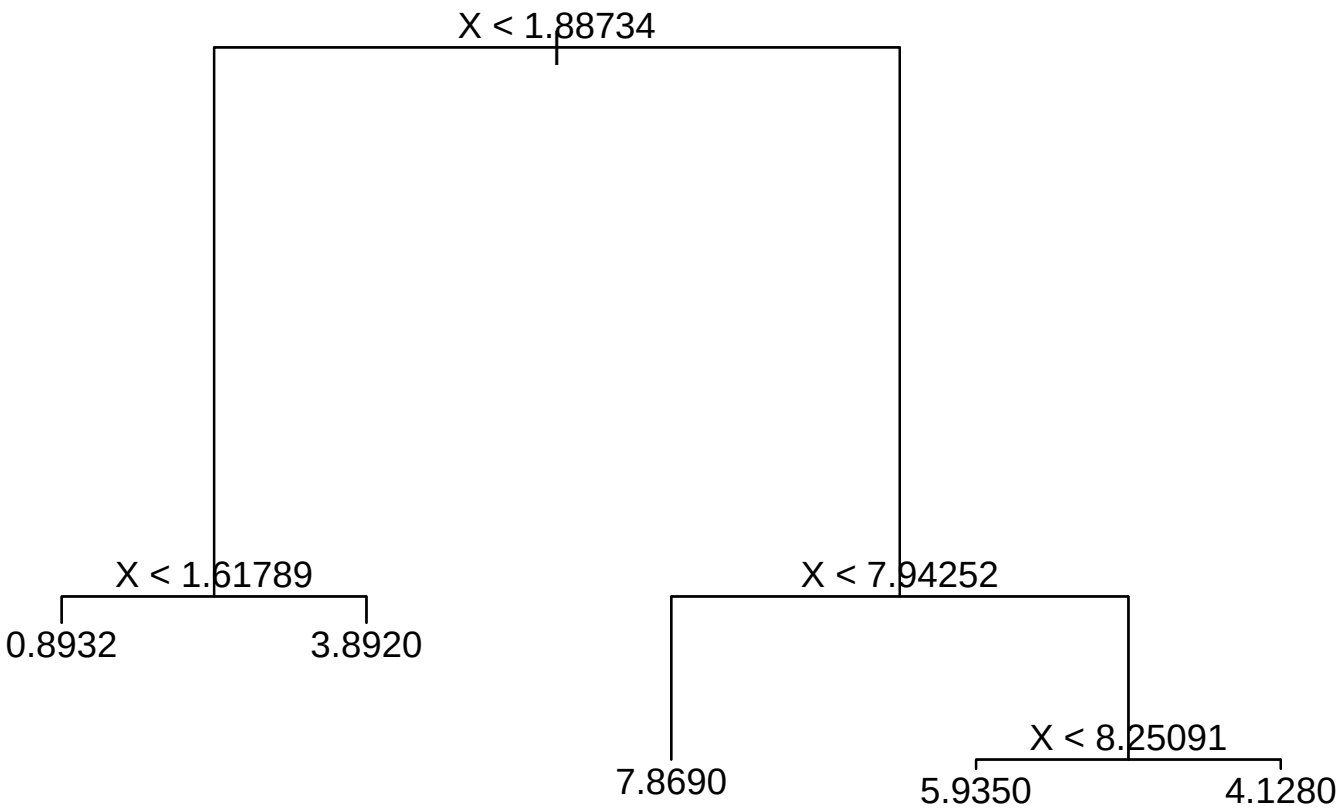
Motivating Example: Regression

- Back to the motivating example for regression.



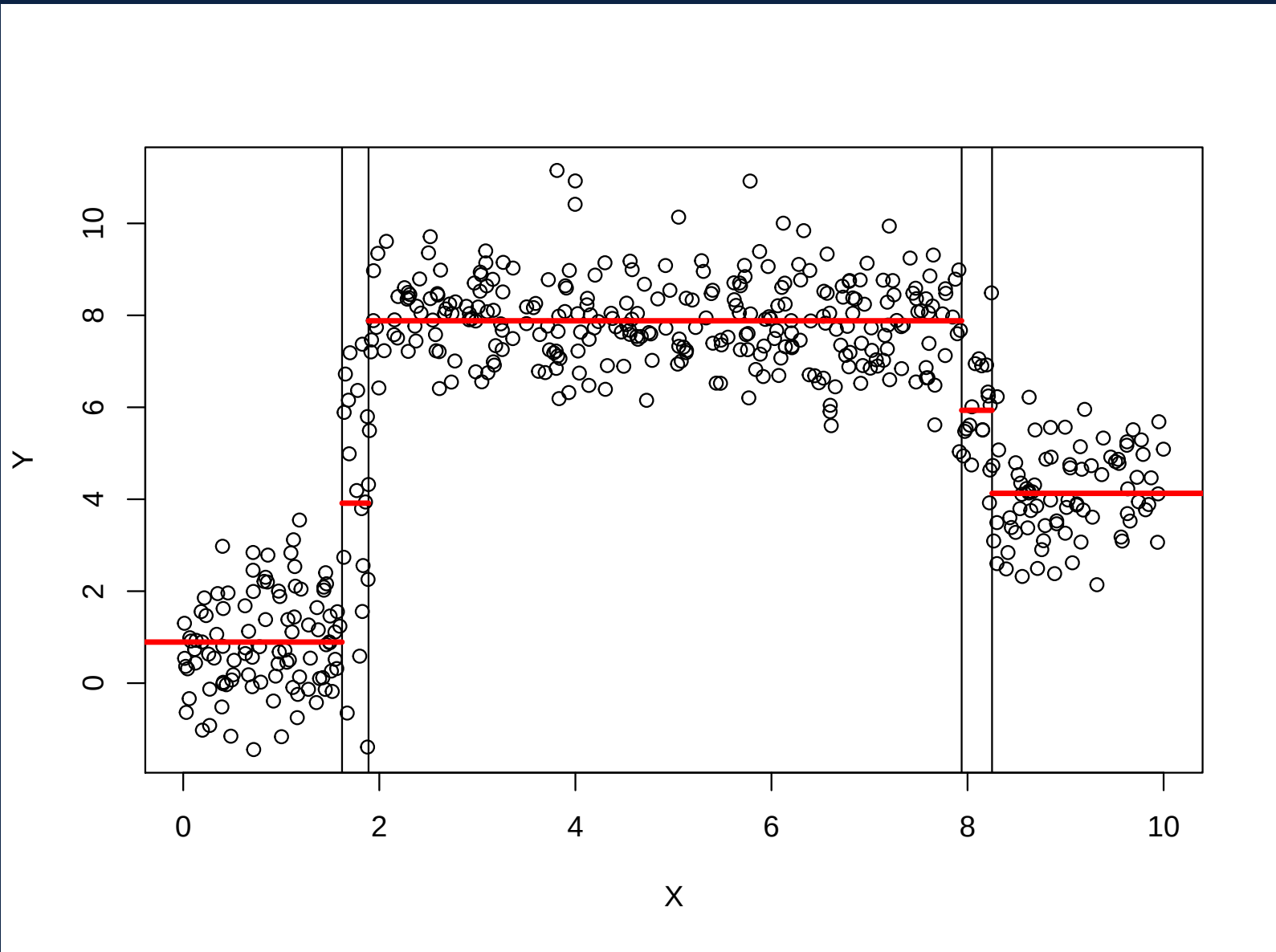
Motivating Example: Regression

- Actual fitting in R gives



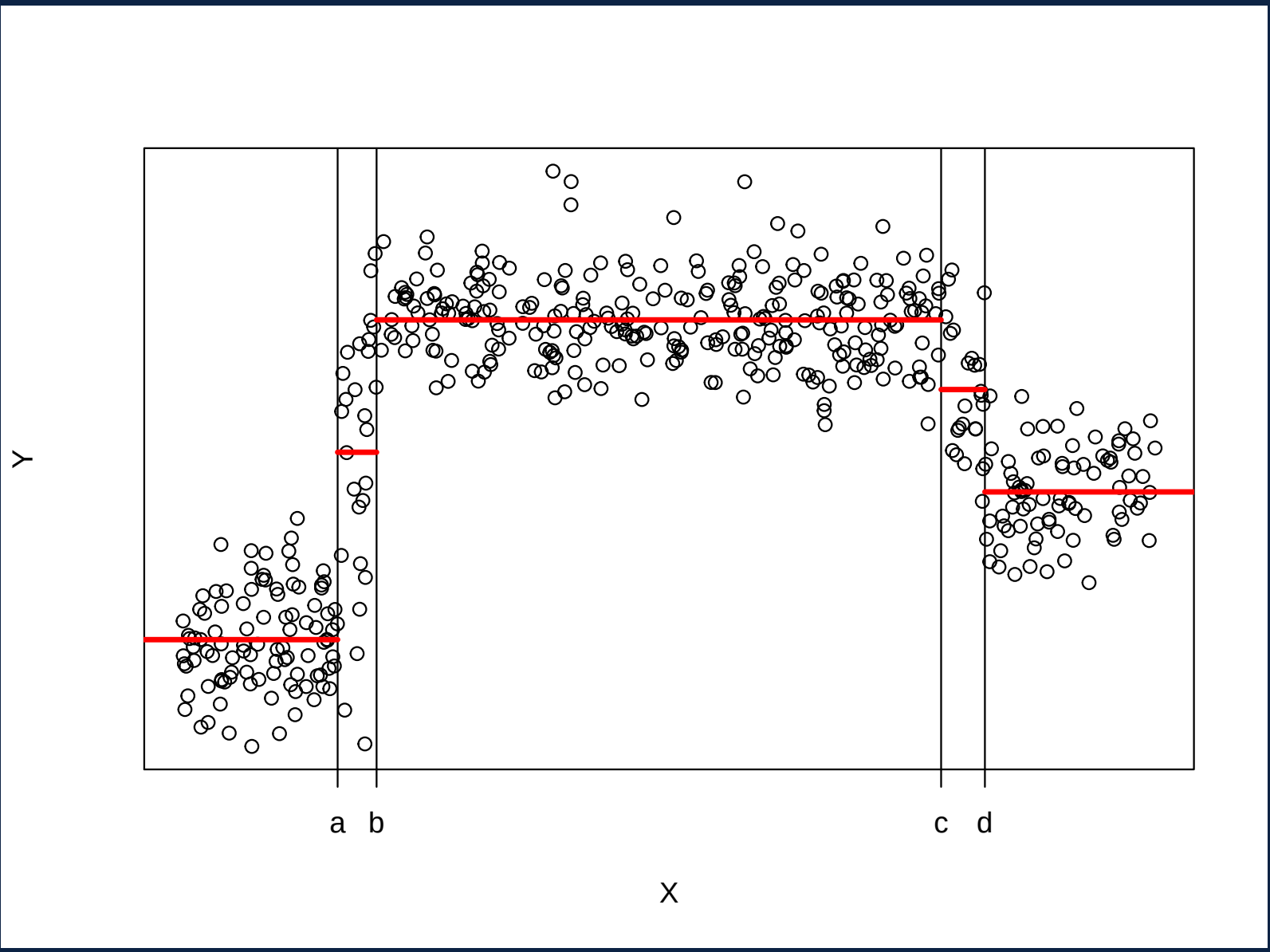
Motivating Example: Regression

- Which back to the plot gives



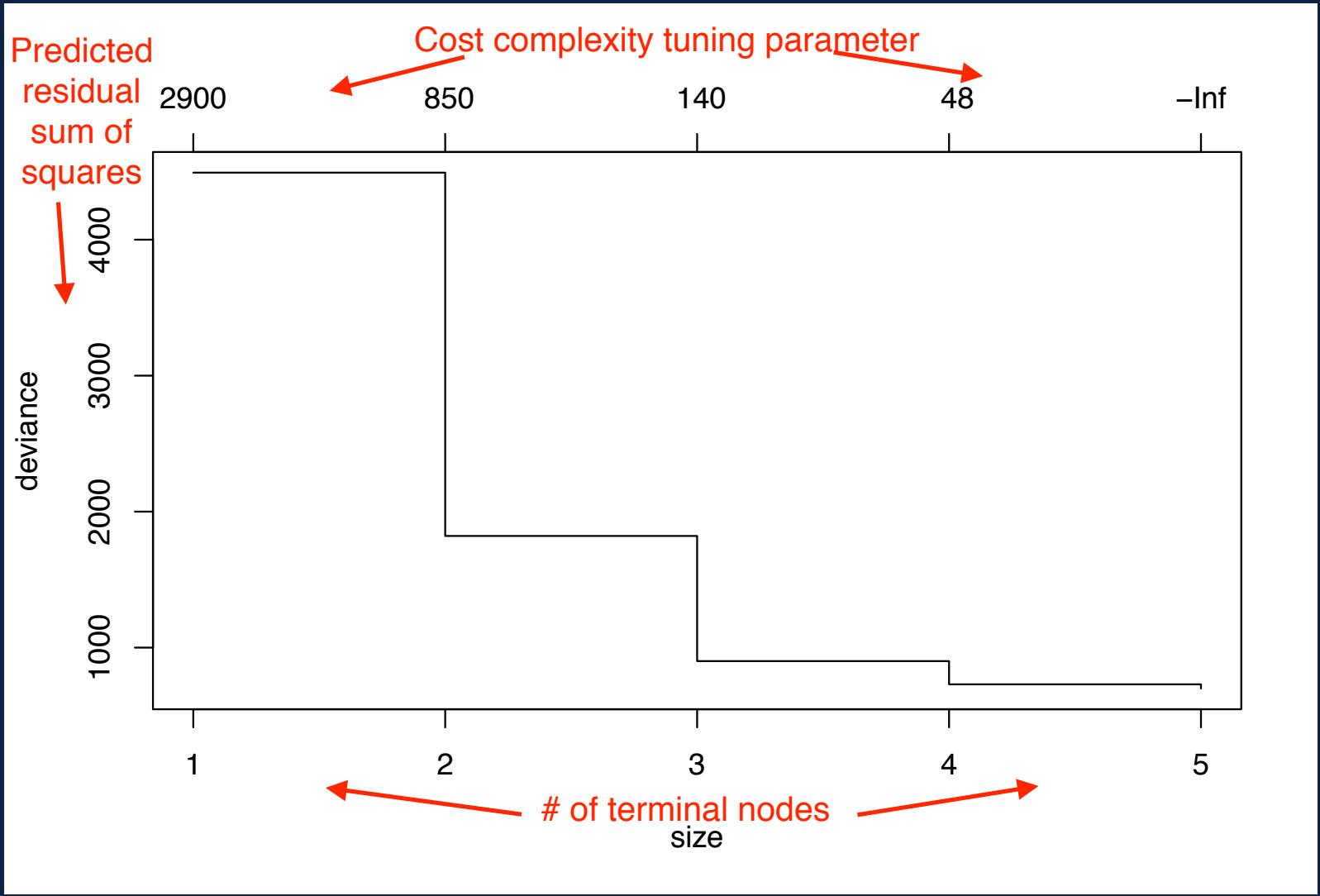
Motivating Example: Regression

- (Which is very close to what we eyeballed earlier)



Motivating Example: Regression

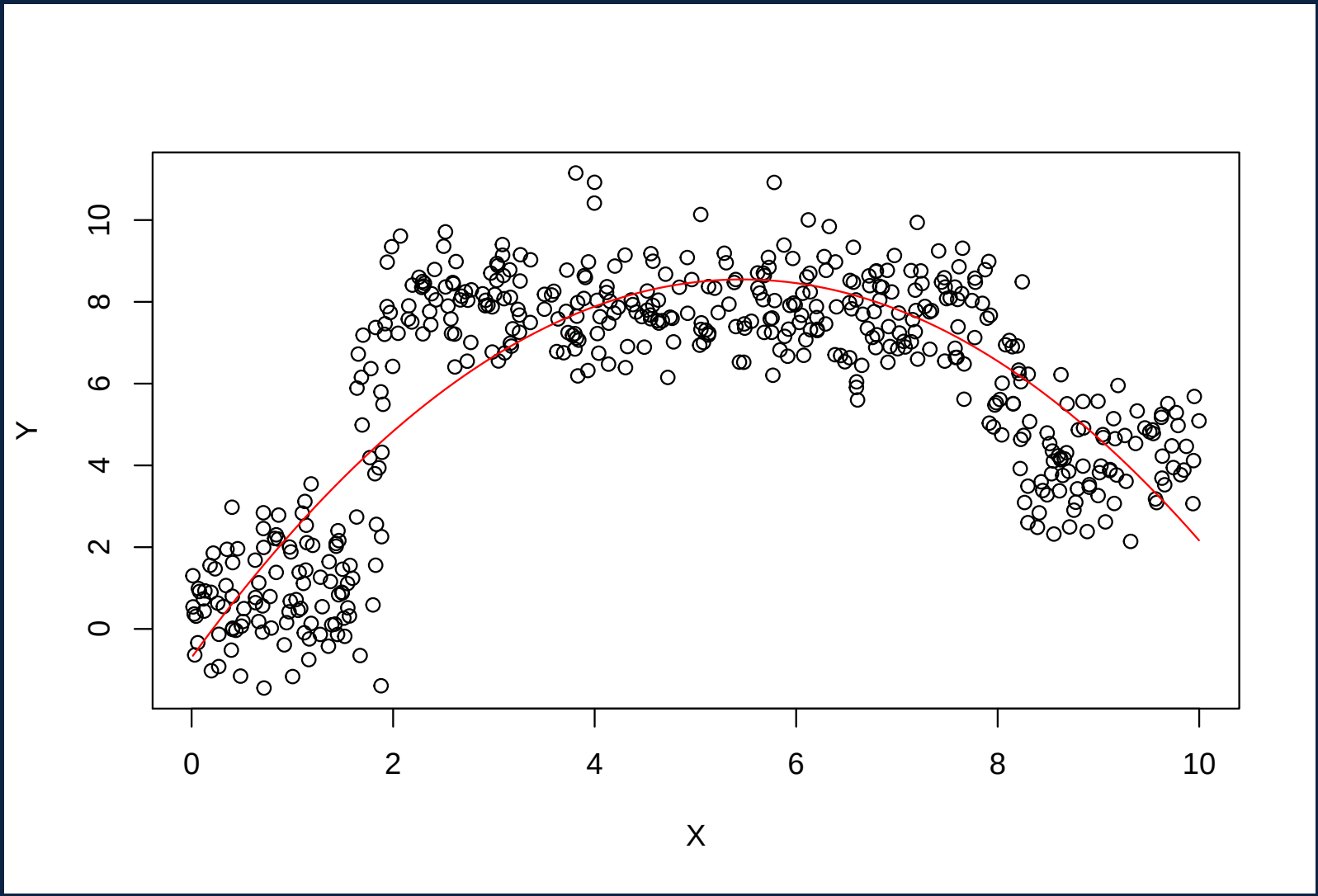
- BUT! We haven't pruned yet



Motivating Example: Regression

- So it turns out, no pruning needed, according to CV
- Now, how does it compare to (quadratic) linear regression ?

Motivating Example: Regression



Motivating Example: Regression

- Training sum squared errors:
  - Tree: 590
  - Quad: 1321
- CV sum squared errors:
  - Tree: 699
  - Quad: 1345

Example: Regression

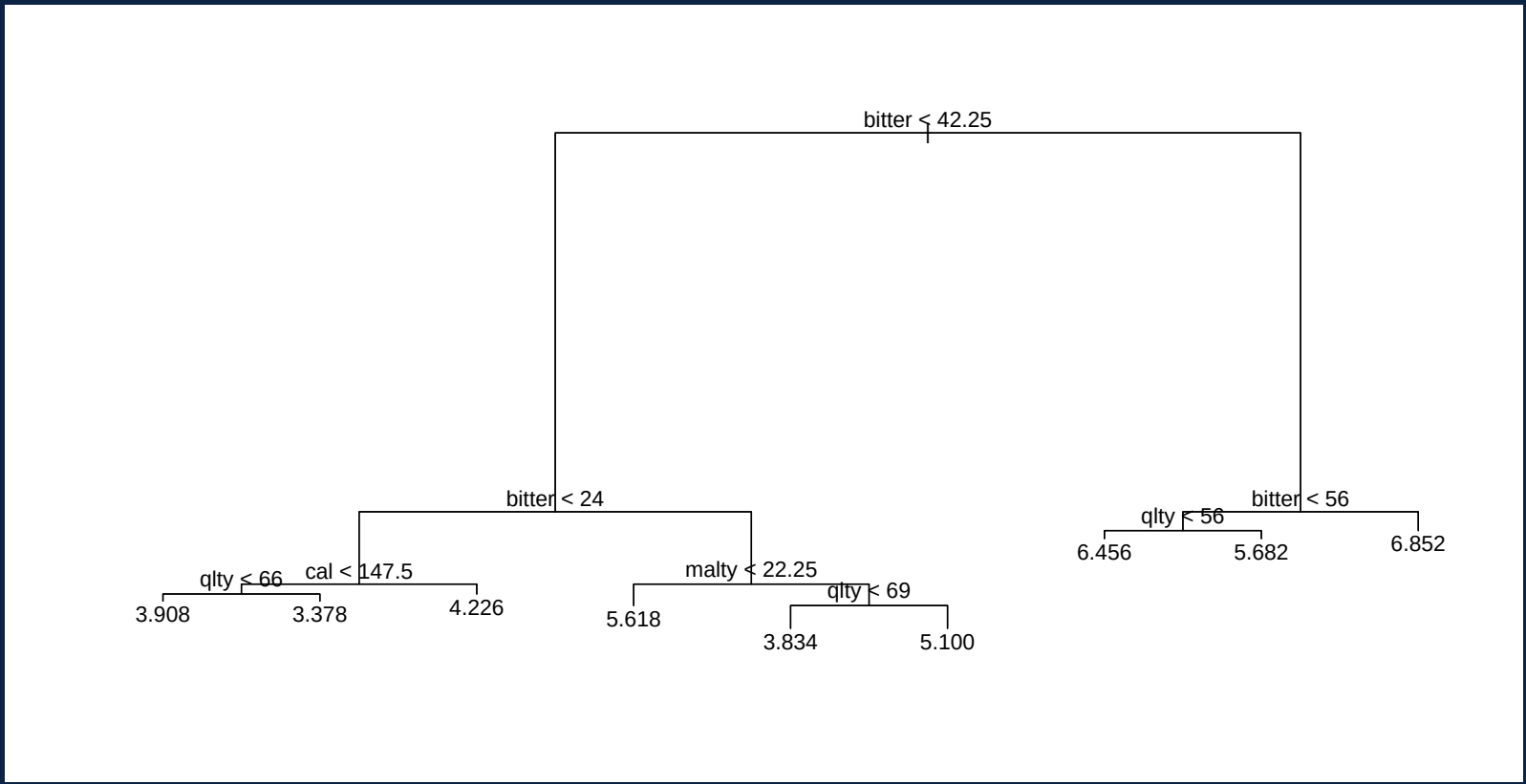
- Let's work with that beer data set again...

```
> head(beer[,-1])
  price qlty cal alc bitter malty
1  6.24   97 159 5.2   52.5  50.5
2  4.79   92 160 5.0   41.5  48.5
3  5.96   90 160 4.9   54.5  48.5
4  4.70   79 162 4.9   37.0  48.0
5  4.11   72 157 5.5   35.5  41.0
6  3.85   66 151 4.9   35.0  24.0
```

- Using a standard call, we fit a large tree...

Example: Regression

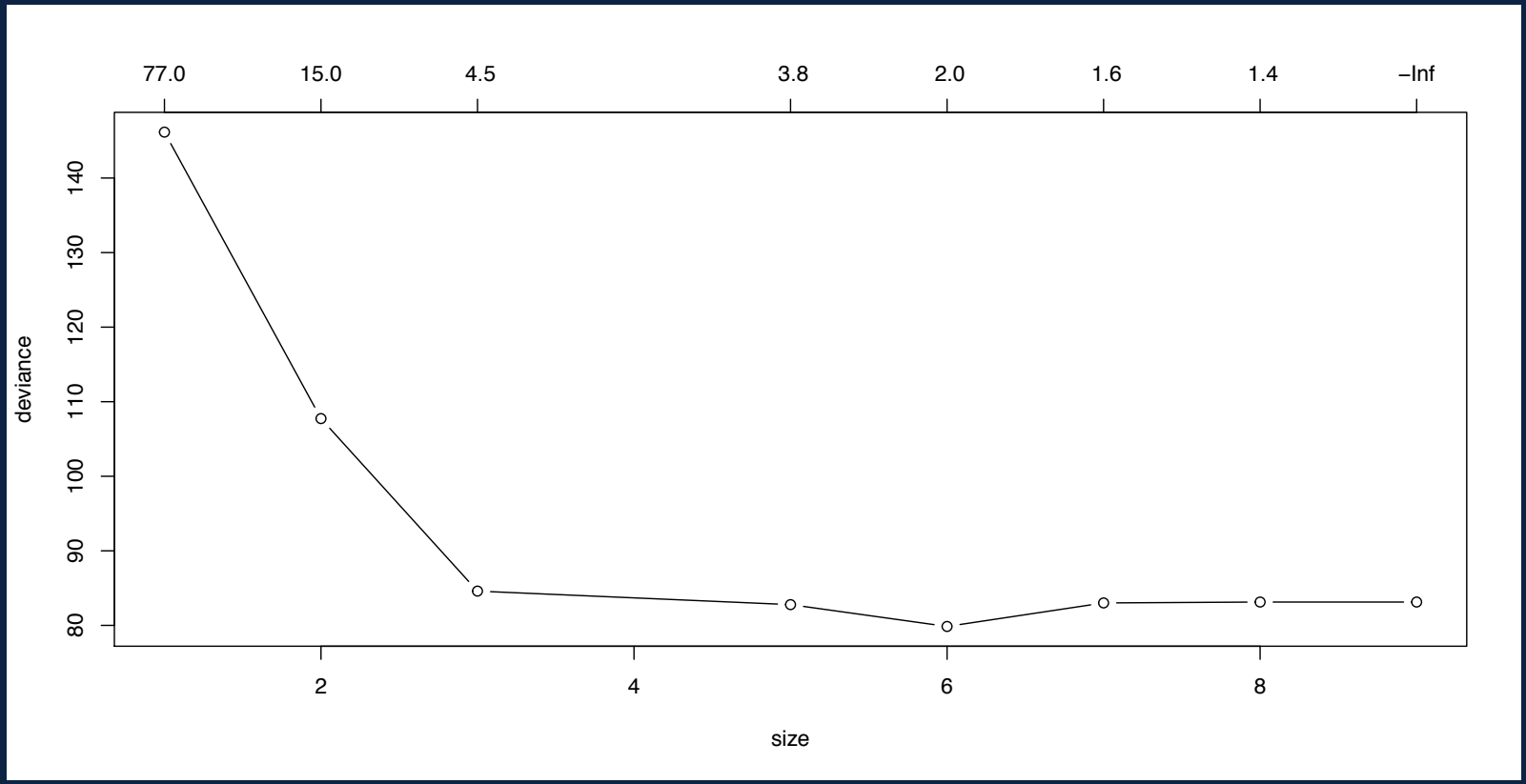
```
> library(tree)
> beert <- tree(price~., data=beerdat)
> plot(beert)
> text(beert)
```



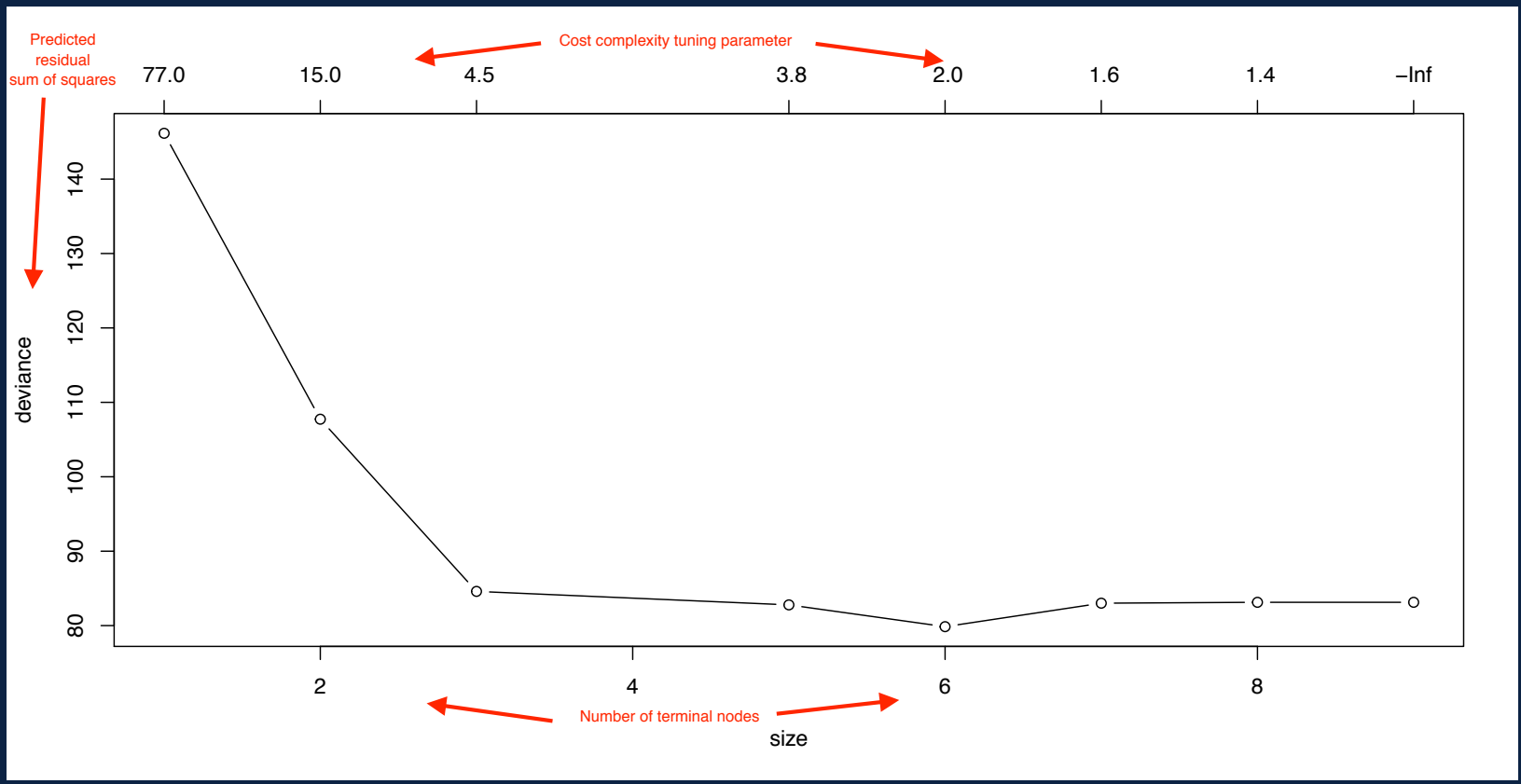
Example: Regression

- As previously noted, this is likely heavily overfitting, so...

```
> cv.beert <- cv.tree(beert)
> plot(cv.beert, type="b")
```



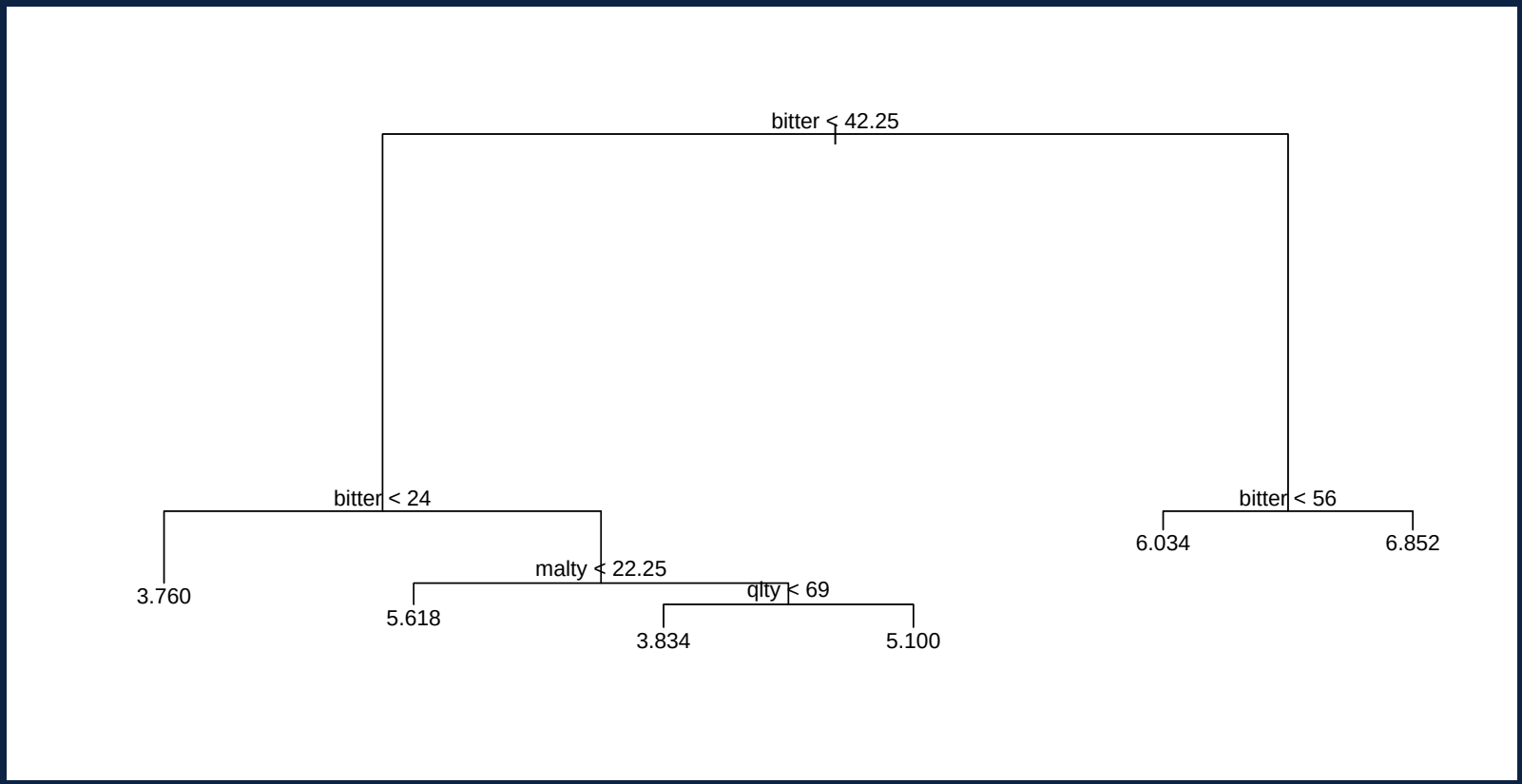
Example: Regression



Example: Regression

- Time to prune...

```
> p.beert <- prune.tree(beert, best=6)
> plot(p.beert)
> text(p.beert)
```



Example: Regression

- We can check the RSS, or equivalently (in this case) the deviance, of the model, and compare to the linear model

```
> sum((yhat-beerdat[,1])^2)
[1] 37.22908
> deviance(p.beert)
[1] 37.22908
> beerlm <- lm(price~., data=beerdat)
> sum(residuals(beerlm)^2)
[1] 47.26182
```

- So is the regression tree better?

Example: Regression

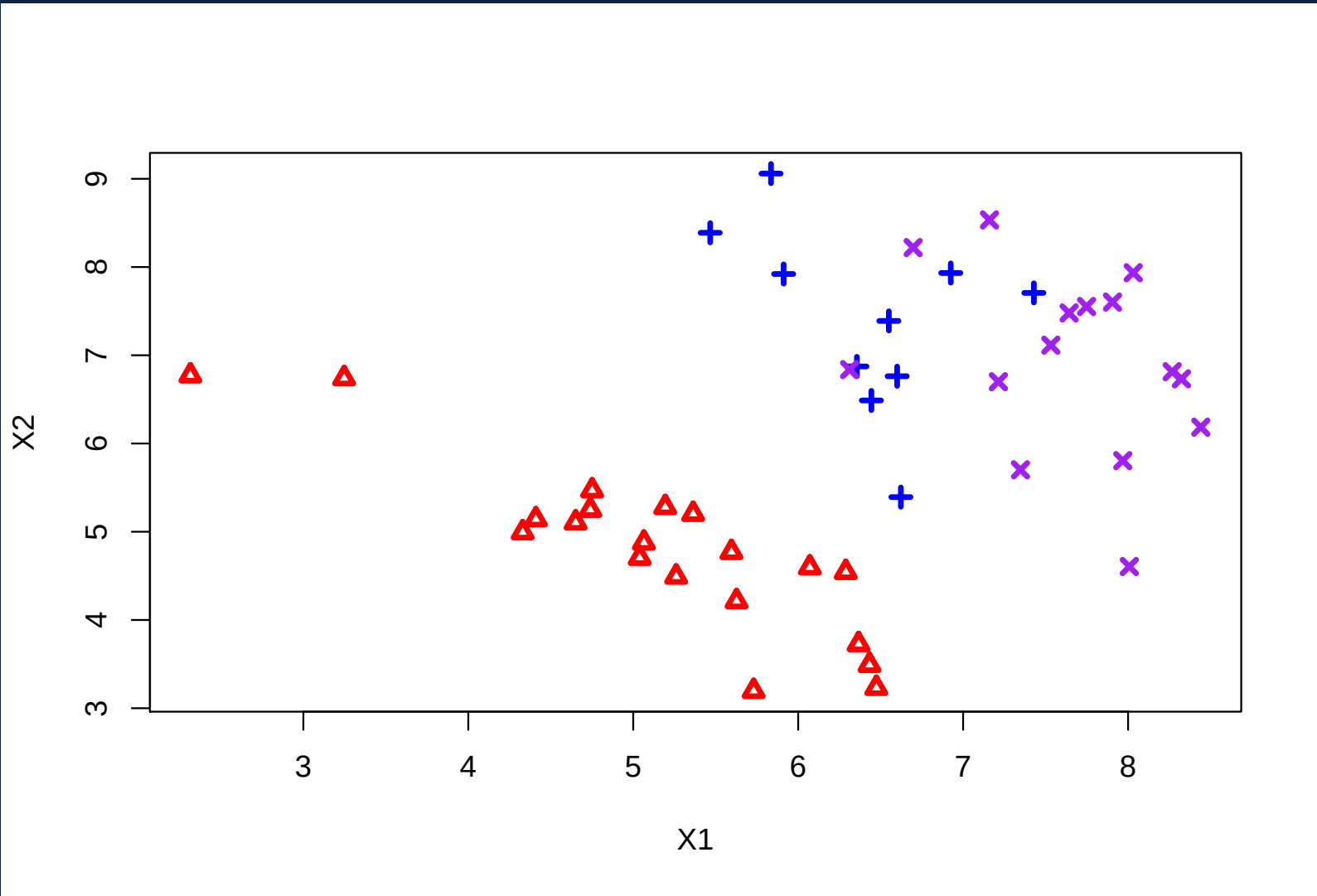
- Those were training RSS!

```
> library(DAAG)
> beercv <- cv.lm(data=beerdat, beerlm, m=10)
> sum((beercv$price-beercv$cvpred)^2) #10CV RSS for lm
[1] 54.6
> min(cv.beert$dev) #10CV RSS for 6 terminal tree
[1] 74.3
```

- So even with relatively simple models we need to be careful!

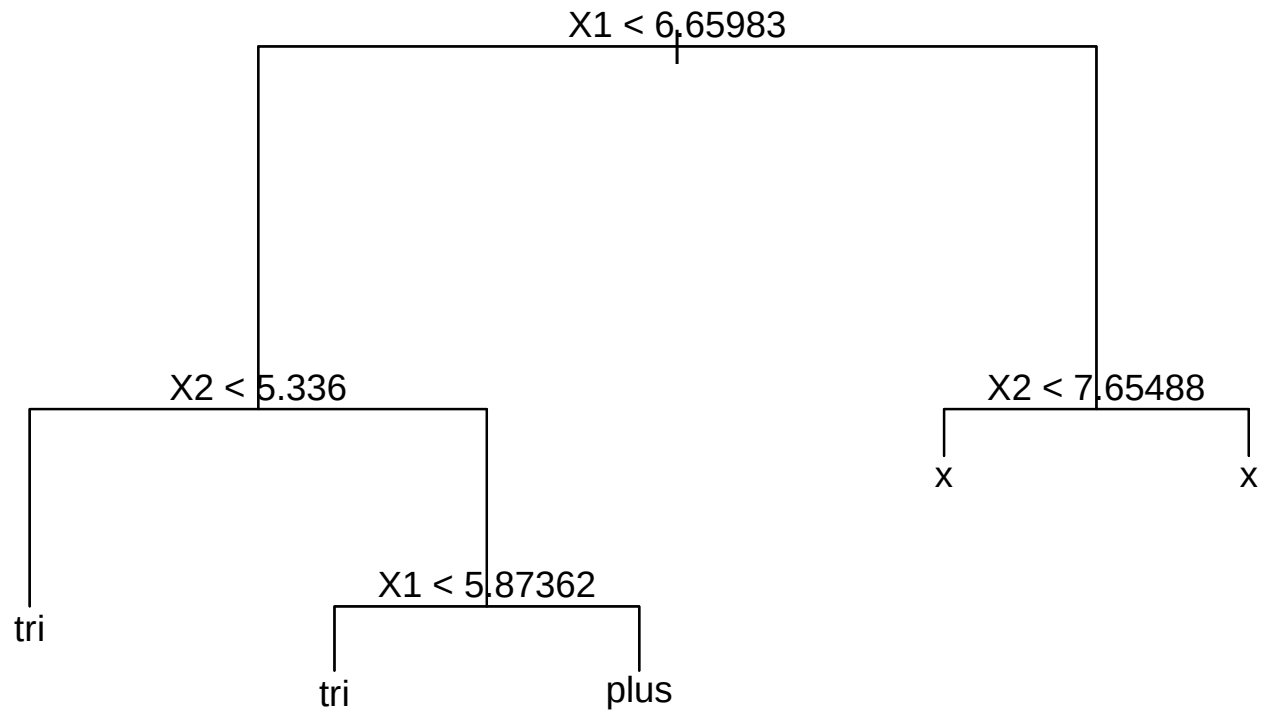
Motivating Example: Classification

- Back to the motivating example for classification.



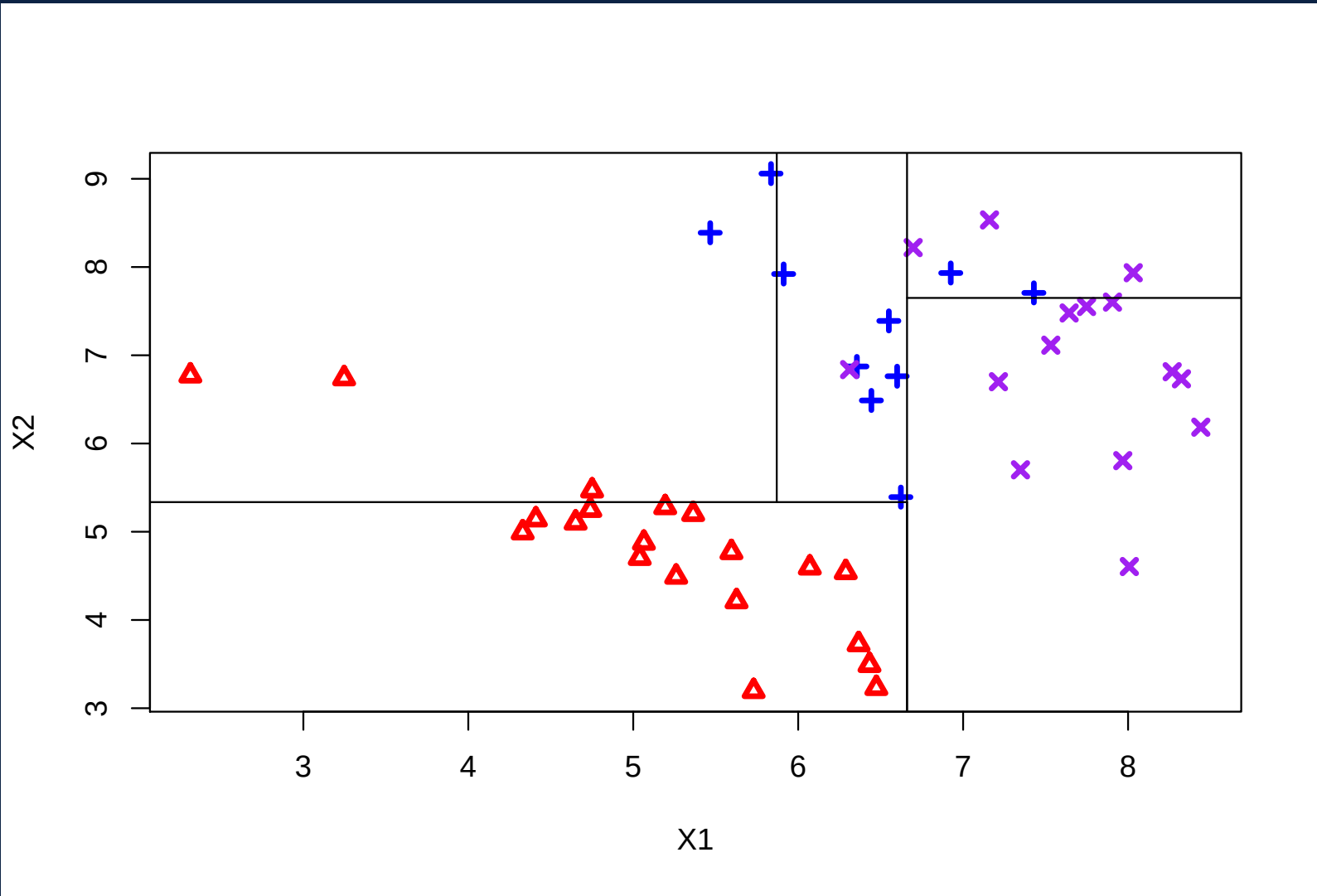
Motivating Example: Classification

- Under default settings, fitting gives



Motivating Example: Classification

- Results in the following plot. Concerns?

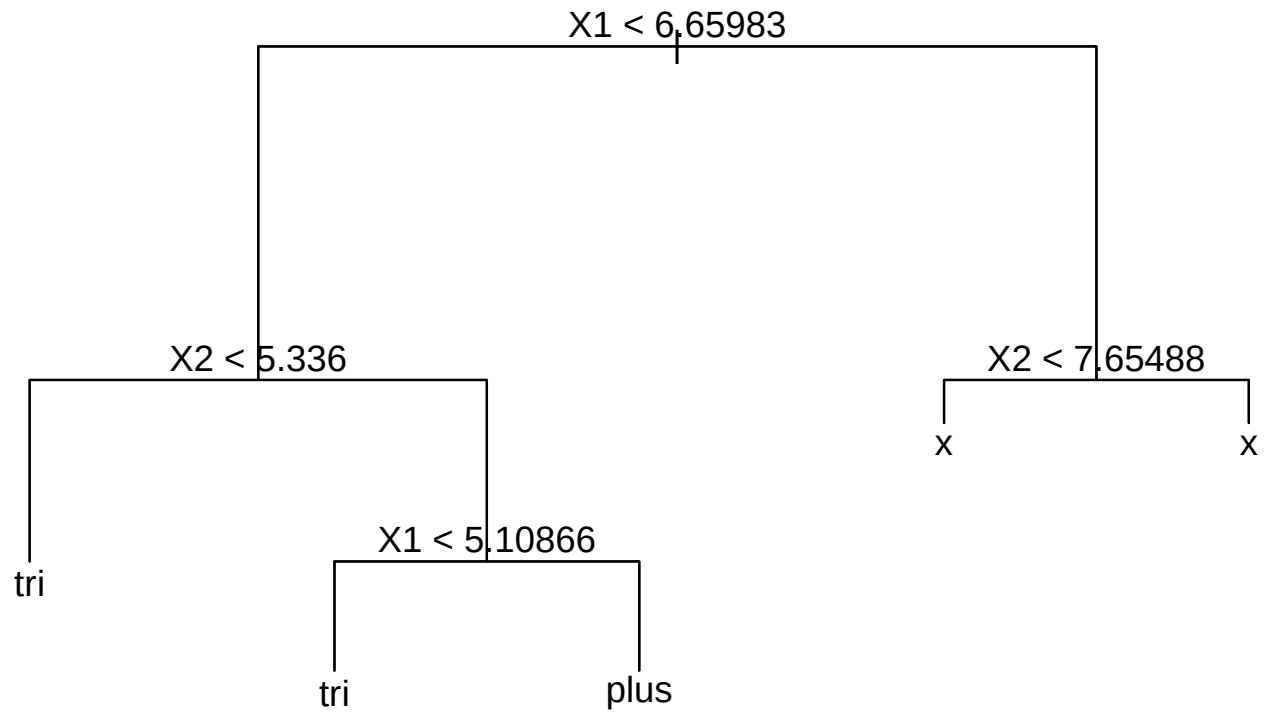


Motivating Example: Classification

- Default model fitting settings have a minimum of 5 observations in any terminal node.
- Change the minimum to 3....and we get

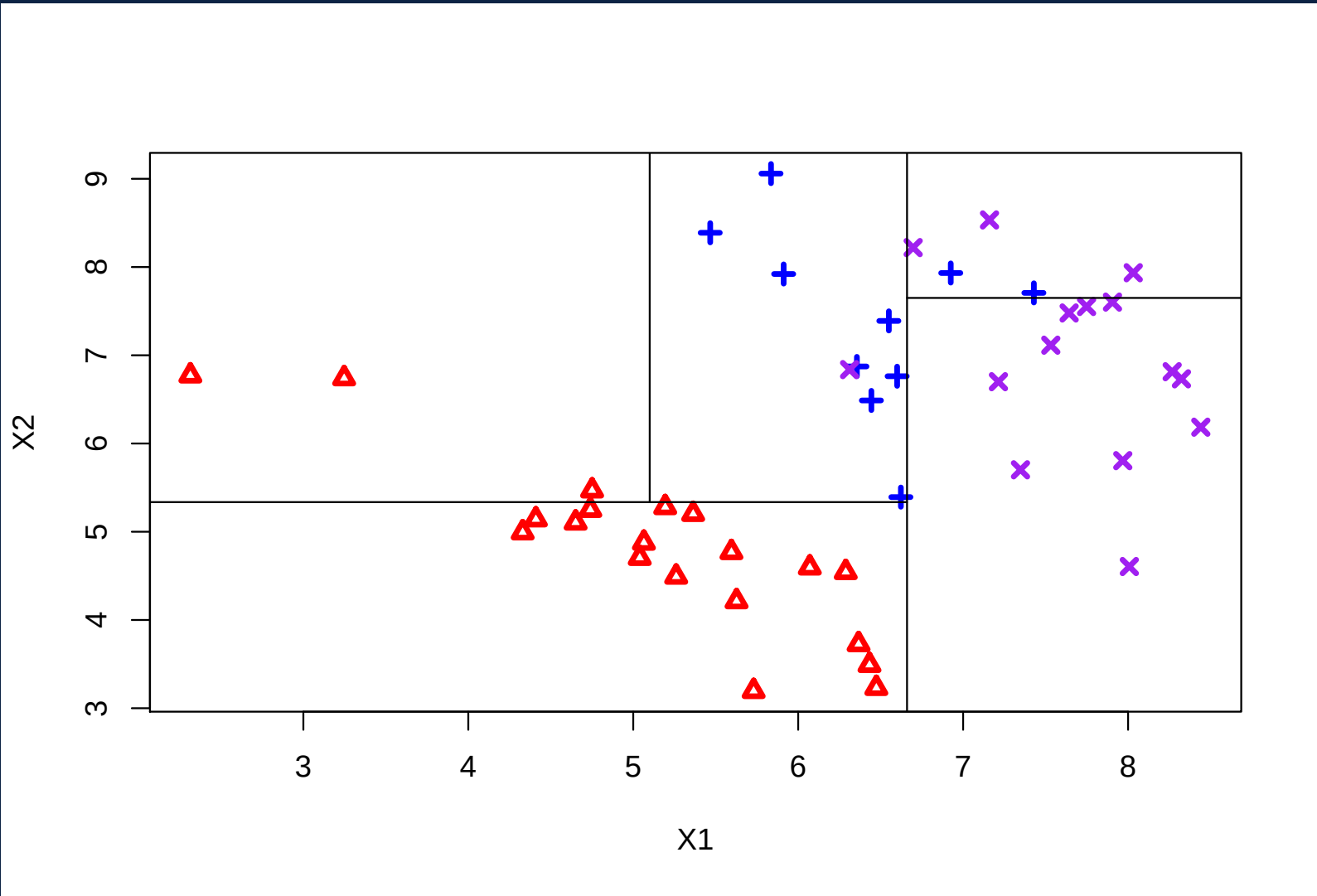
Motivating Example: Classification

- Under custom settings, fitting gives



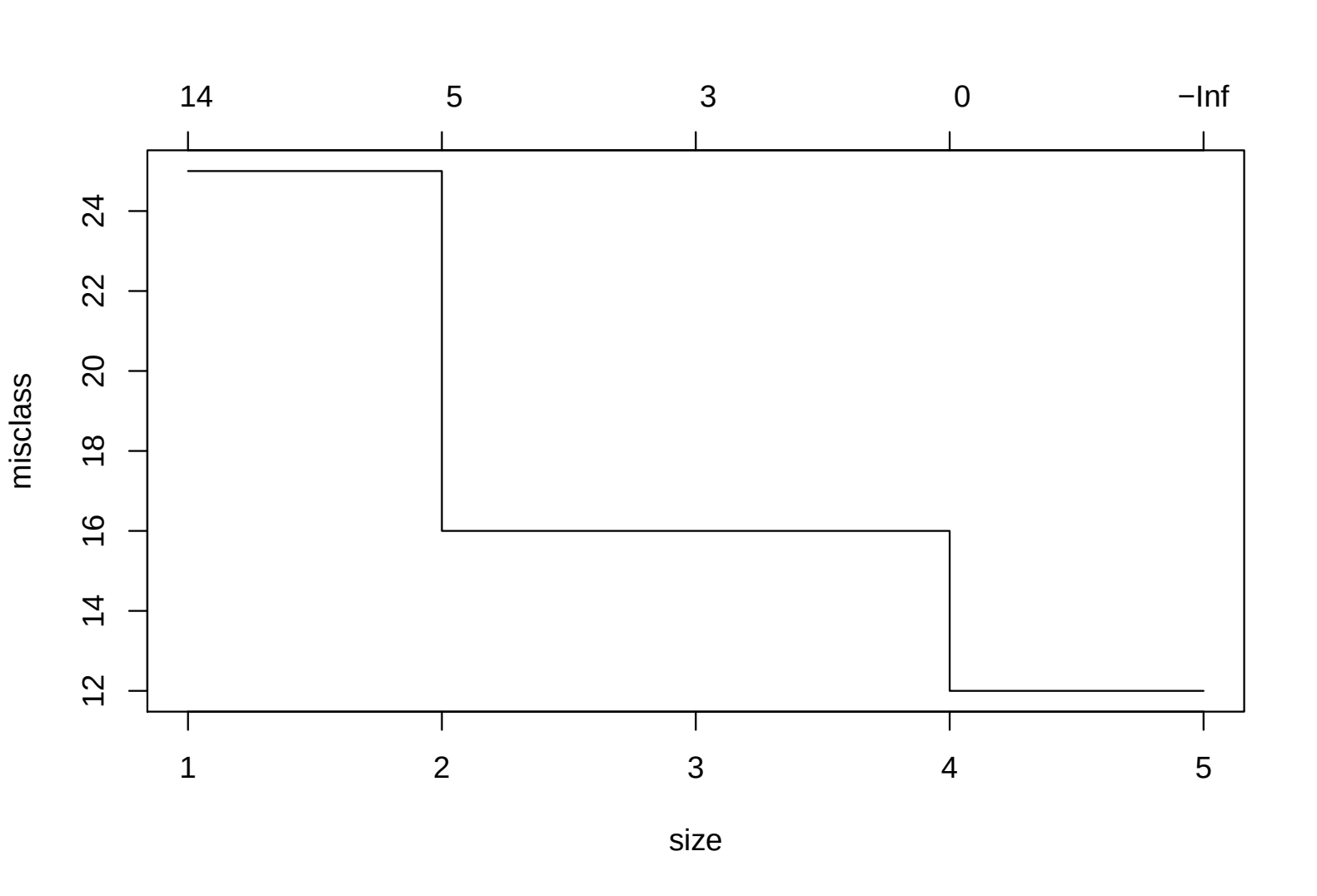
Motivating Example: Classification

- Results in the following plot.



Motivating Example: Classification

- CV suggests to prune to 4 terminal nodes (for parsimony)



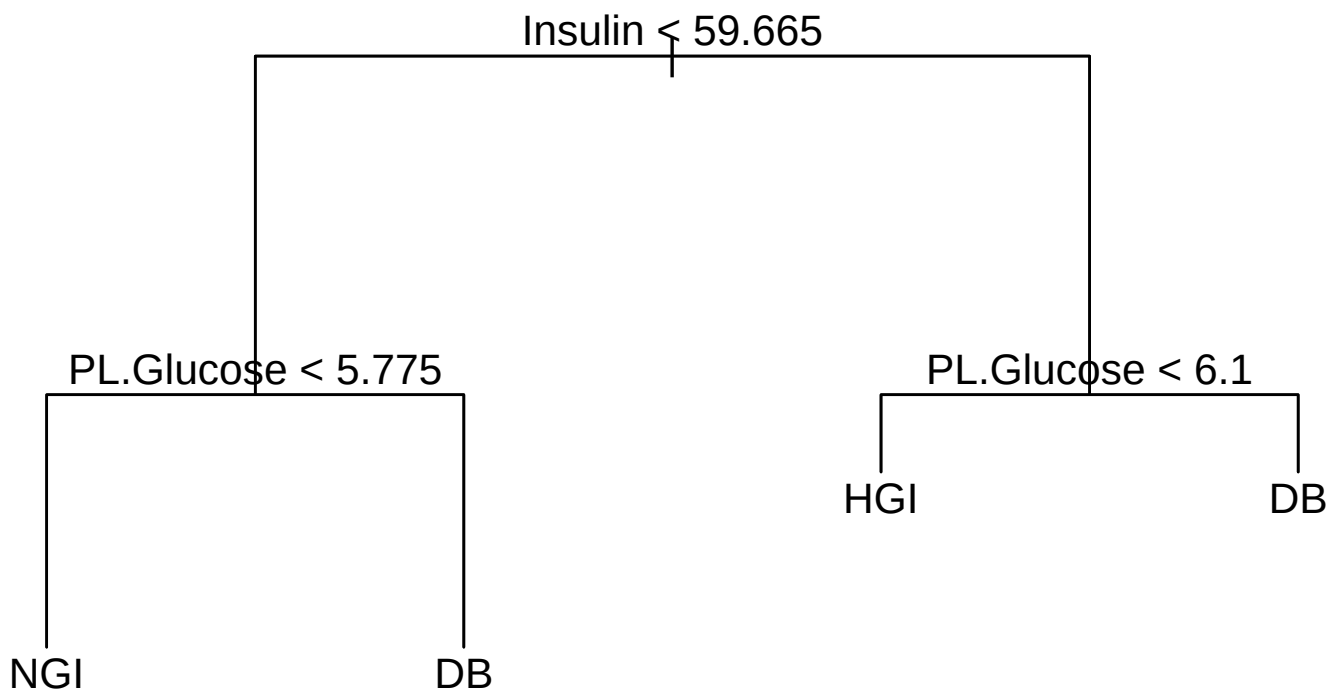
Motivating Example: Classification

- Misclassification rate: $3/45 = .067$
- CV misclassification rate: $12/45 = .267$

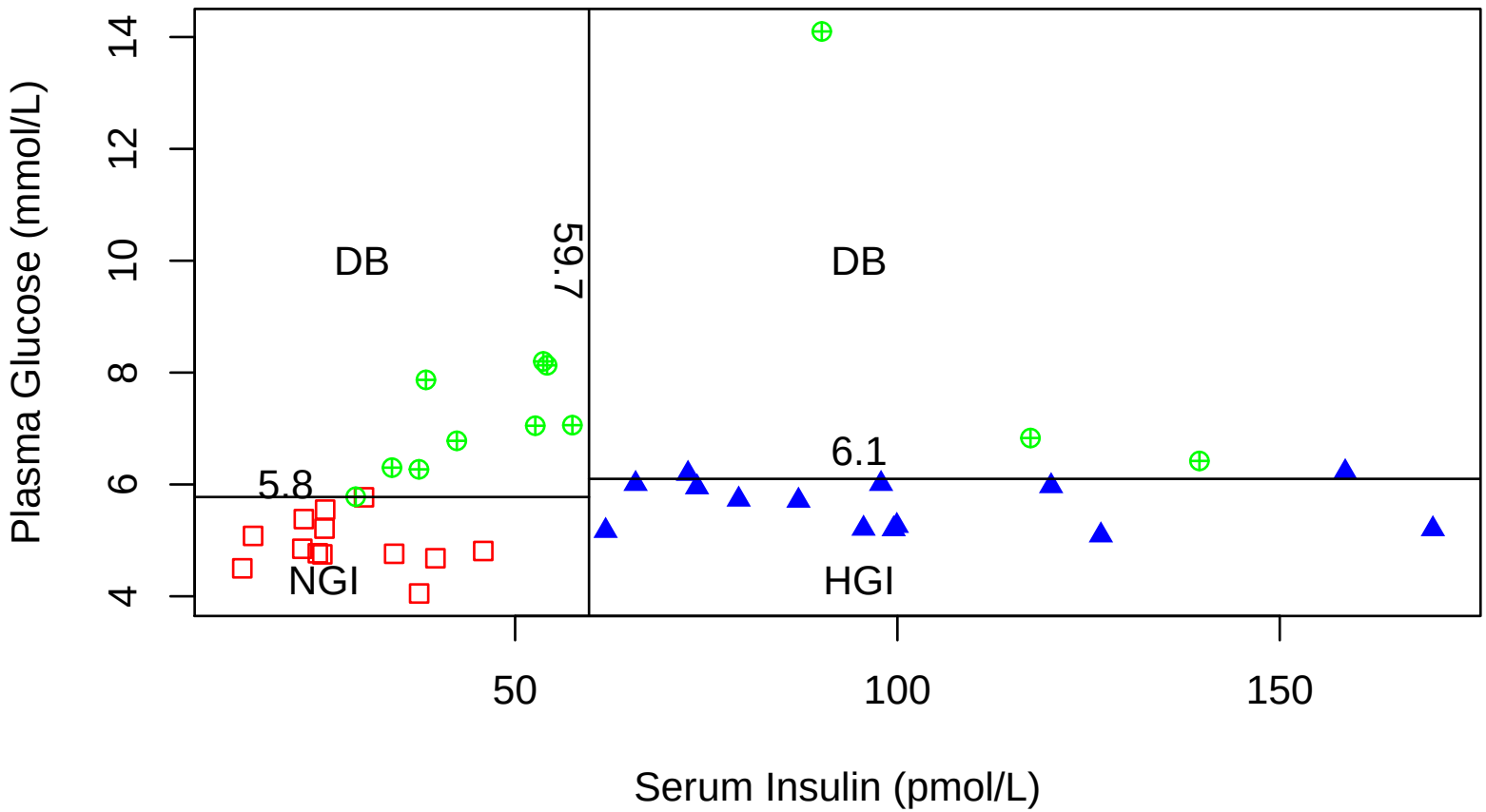
Example: Classification

- Actual prior consultation I worked on.
- Lots of measurements collected on patients with normal glucose levels (NGI), high levels (HGI), and diabetes (DB)
- I fit a tree to predict the above classifications, and...

Example: Classification



Example: Classification



Example: Classification

- Only 2 misclassifications! Rate $2/39 = 0.0513$
- And, I was told, results match fairly closely to current diagnostic procedures.
- Cross validation suggests the full size tree proposed in the previous pictures, though it does predict a higher misclassification rate of $10/39 = 0.256$

Discussion

- Pros/cons/discussion on board


THE UNIVERSITY OF BRITISH COLUMBIA