Real Time Series analysis and modelling

Aid machine learning with dynamical insight

Hailiang Du
Department of Mathematical Sciences, Durham University
hailiang.du@durham.ac.uk
Data Science Institute, London School of Economics and Political Science
h.l.du@lse.ac.uk

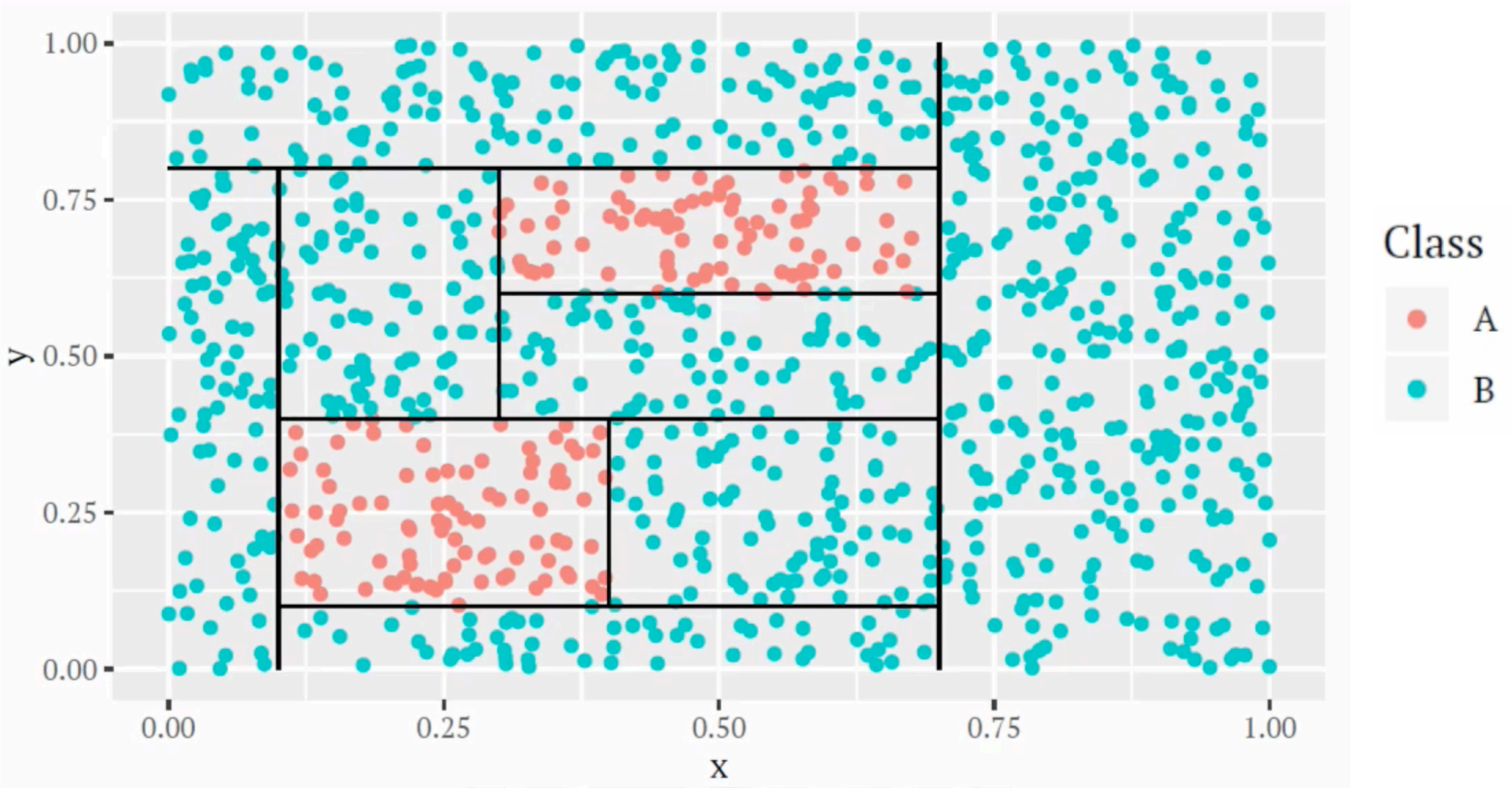
All theorems are true, All models are wrong. All data are inaccurate. What are we to do?

The aim of this course is to teach you how to deal with real data, to increase your scepticism regarding reliable modelling in practice, and to expand the tool box you carry to include nonlinear techniques, both deterministic and stochastic with the aid of dynamical insight.
In short: to get you to think before you compute (and perhaps afterwards too.)

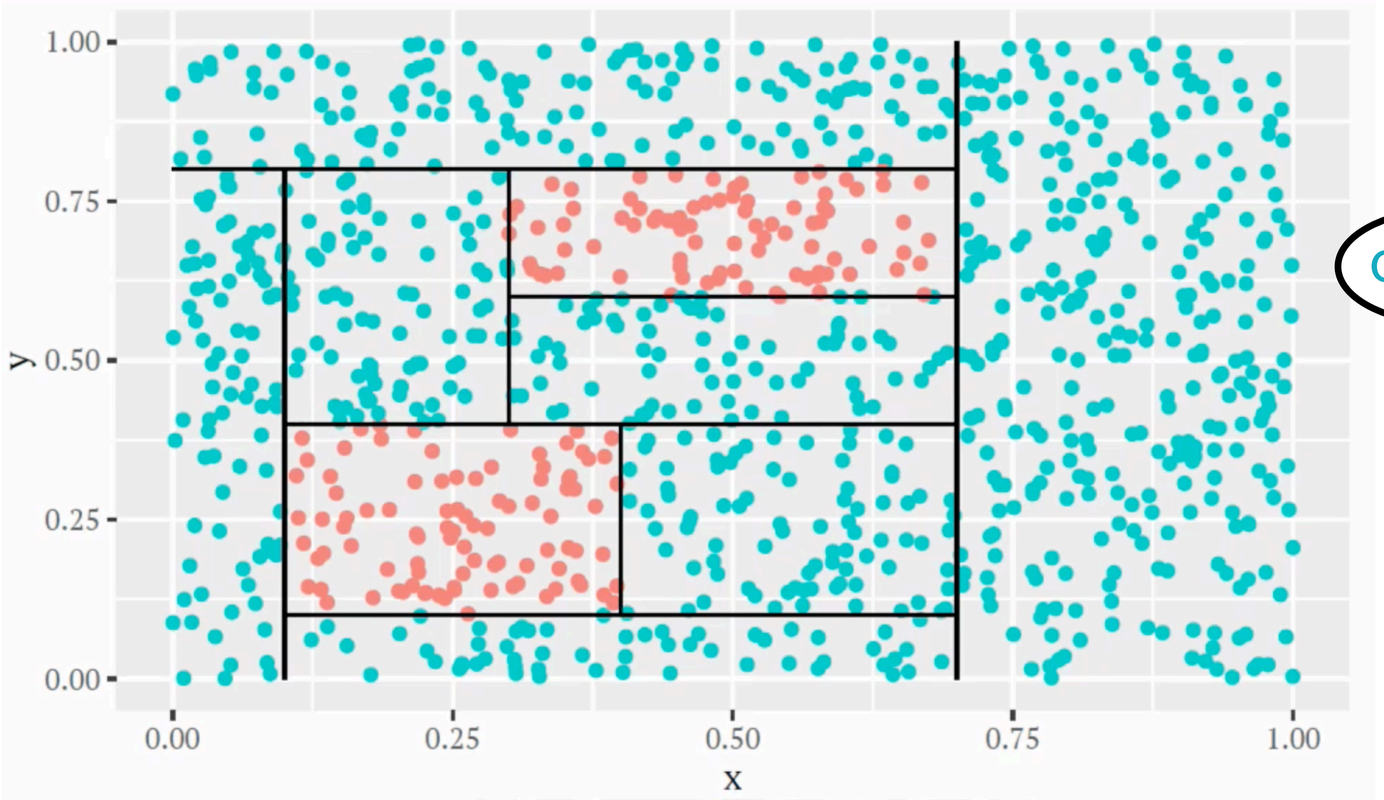
Lecture7

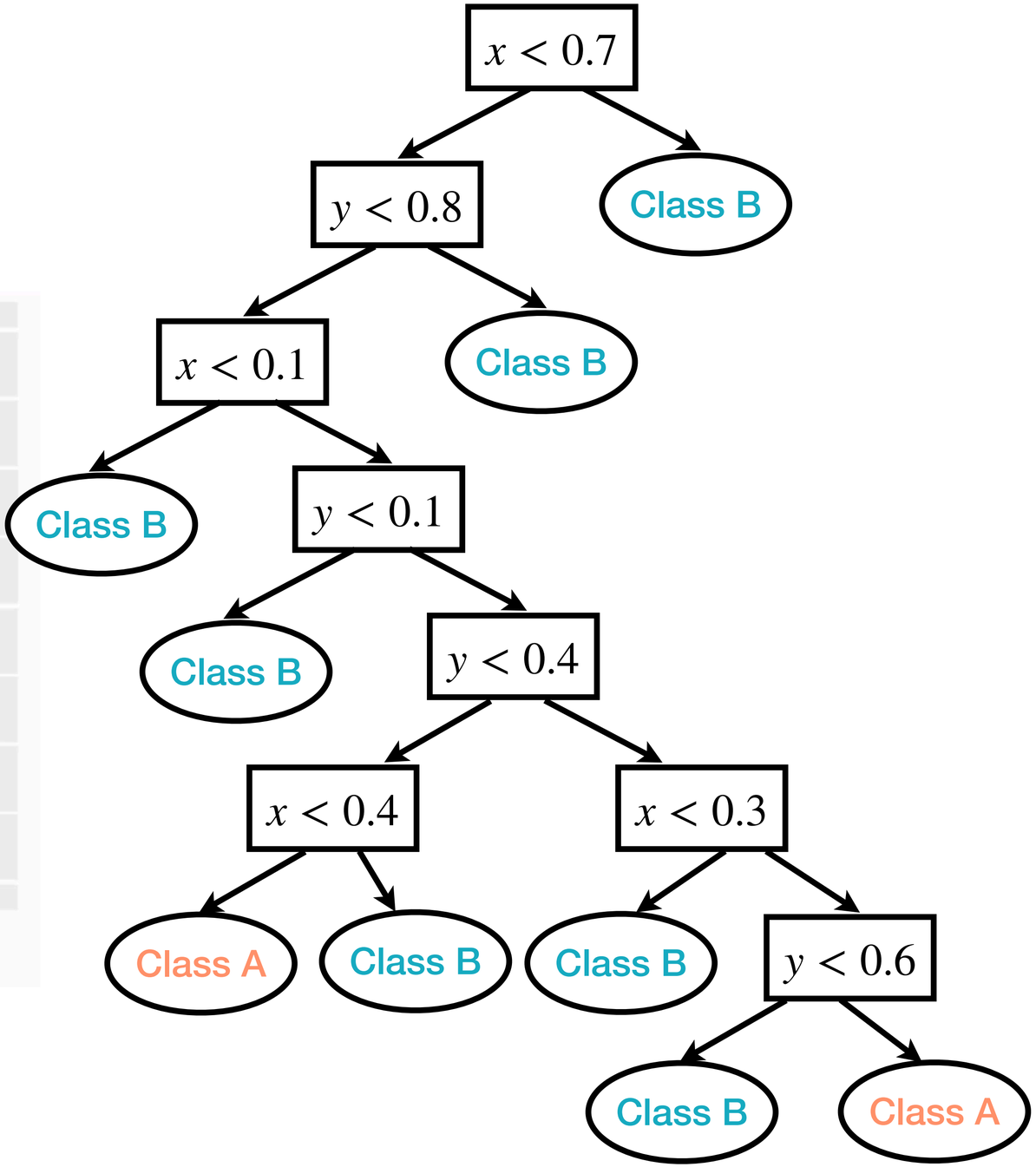
Classic nonlinear statistical (machine) learning models

General idea about Trees

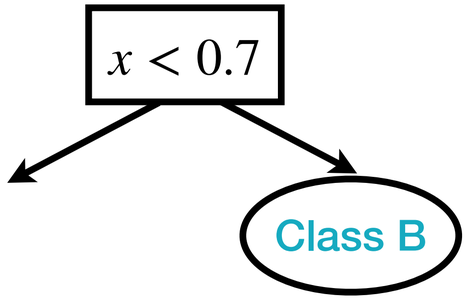


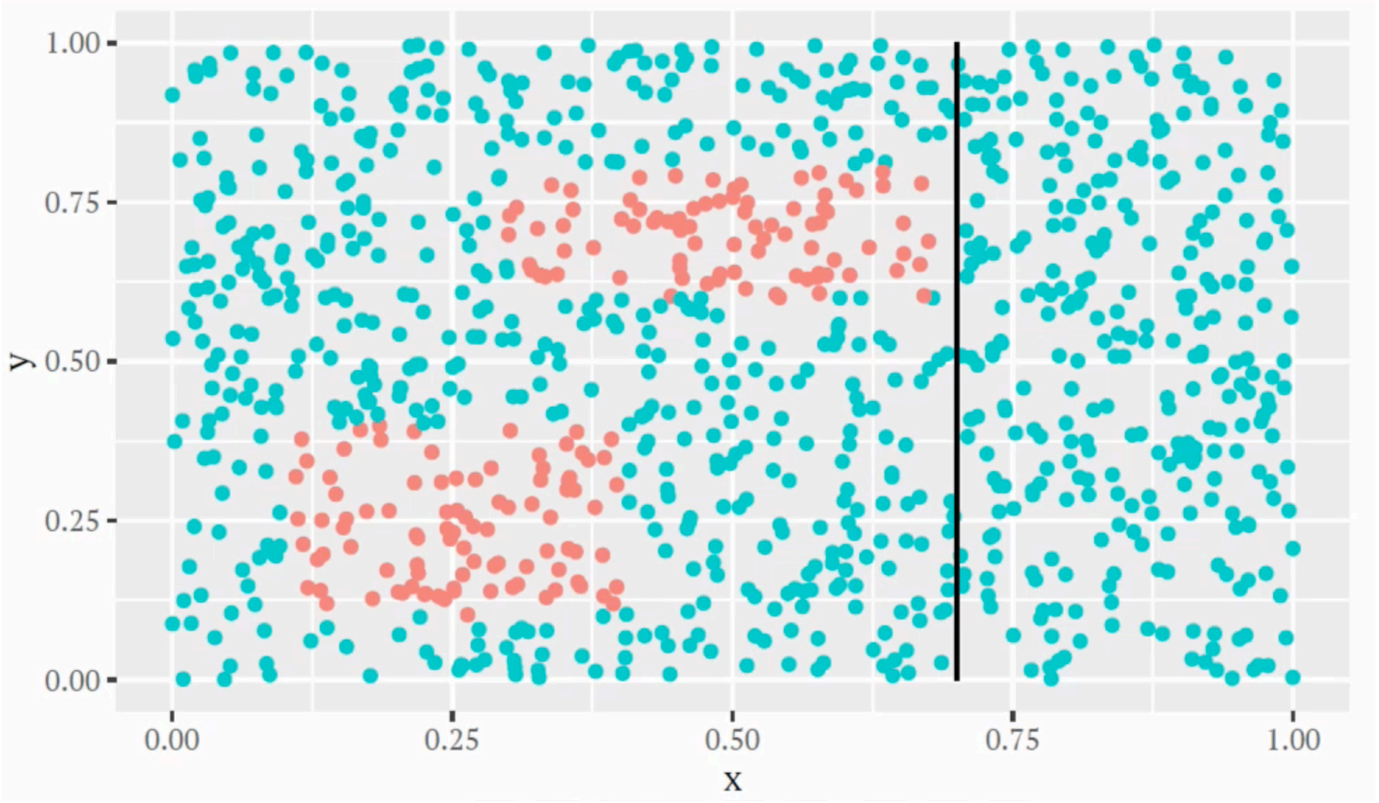
General idea about Trees



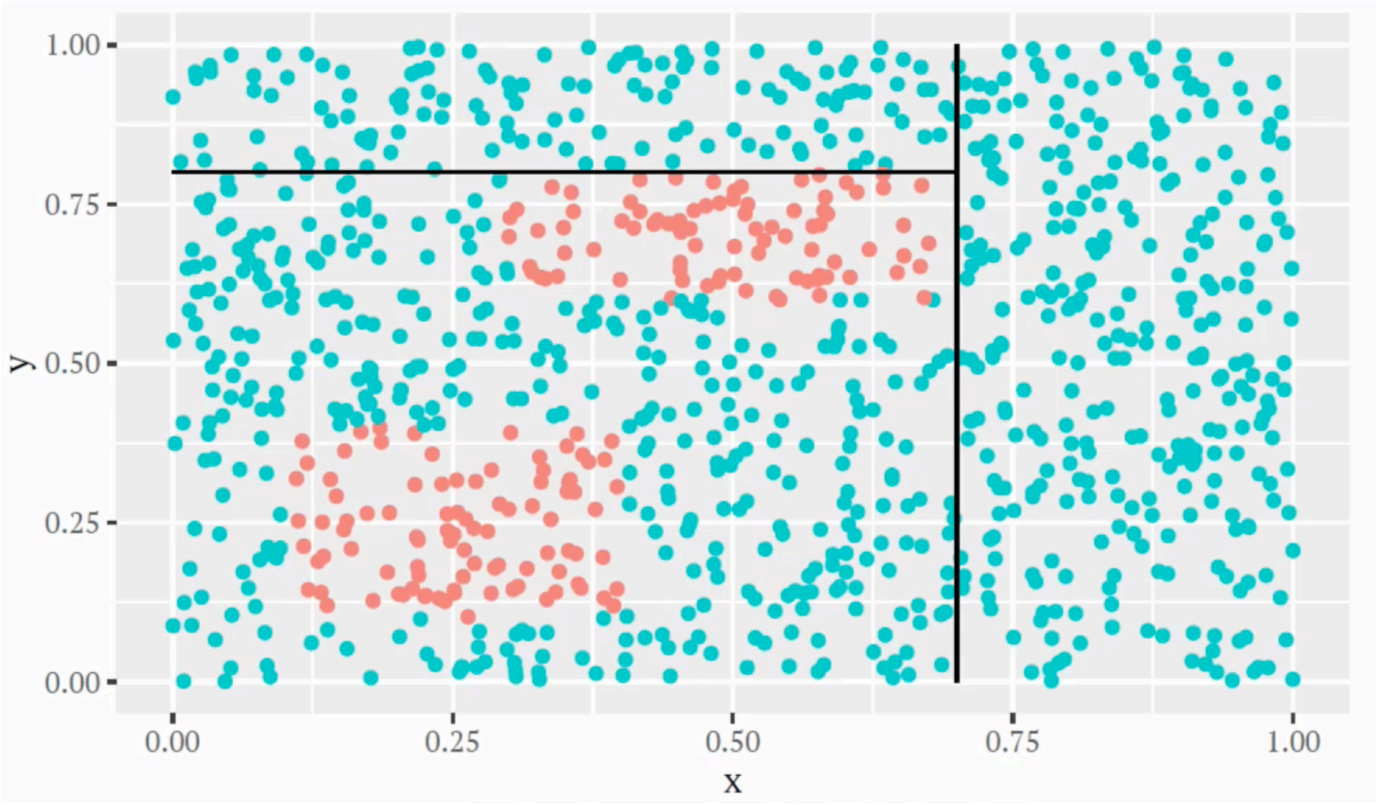


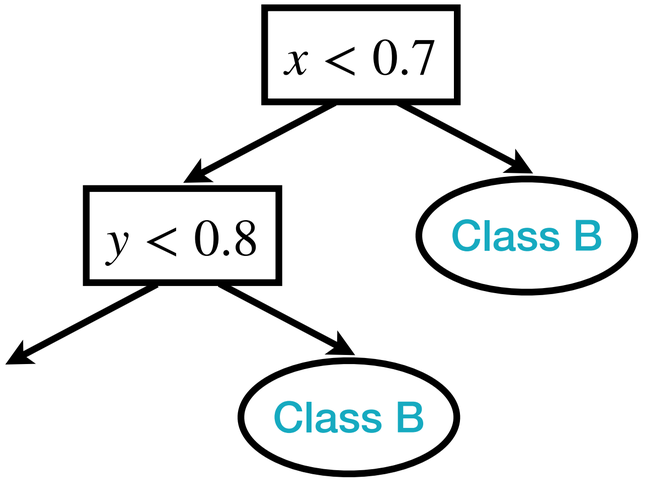
General idea about Trees



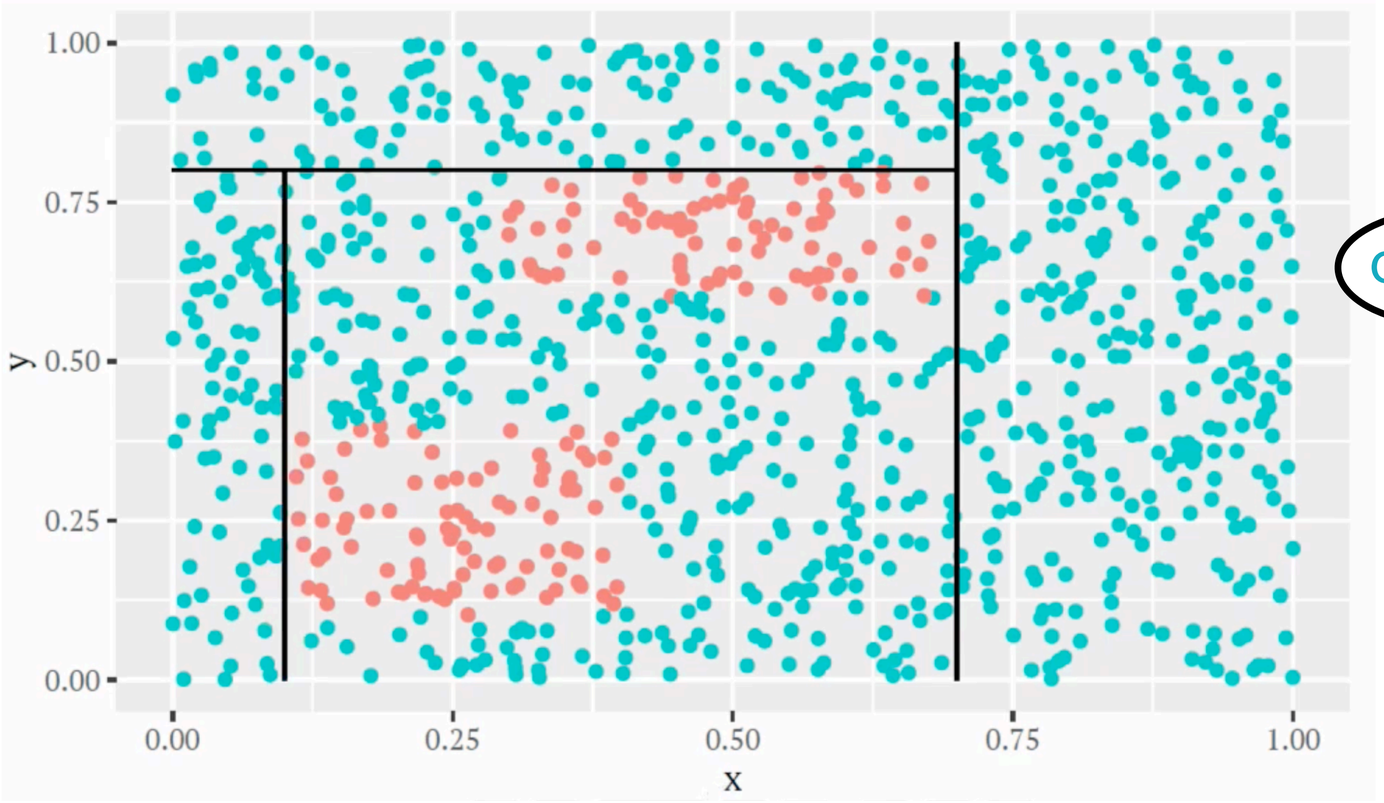


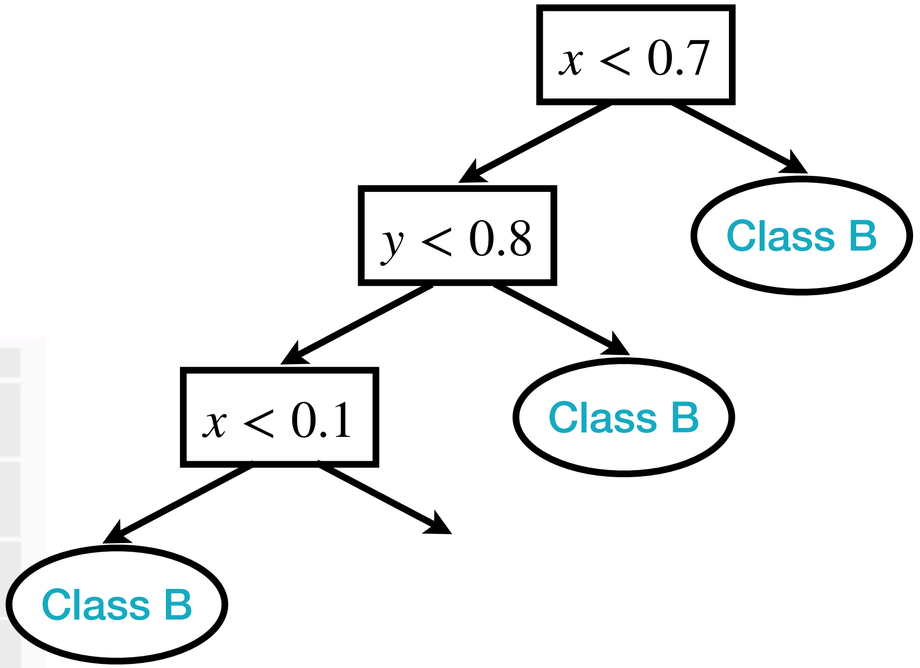
General idea about Trees



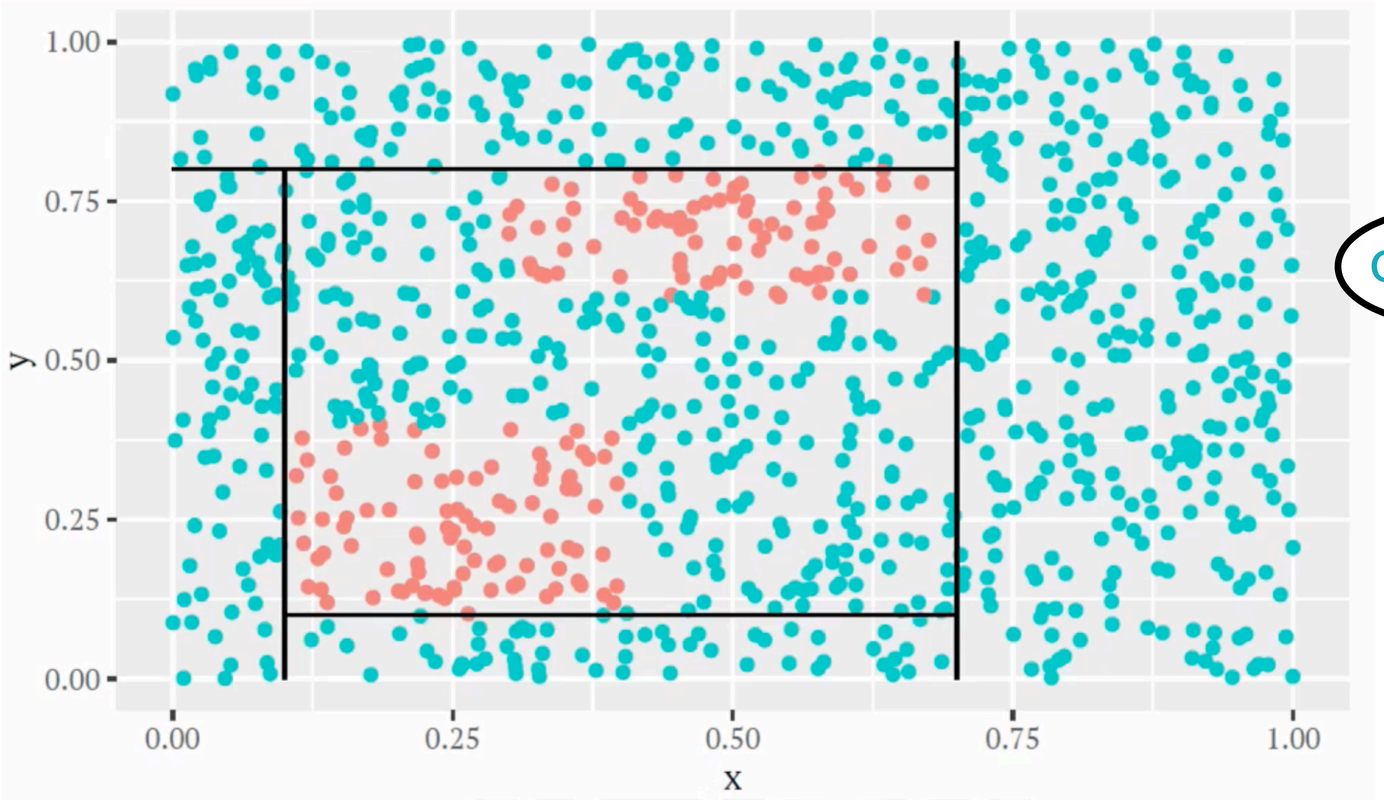


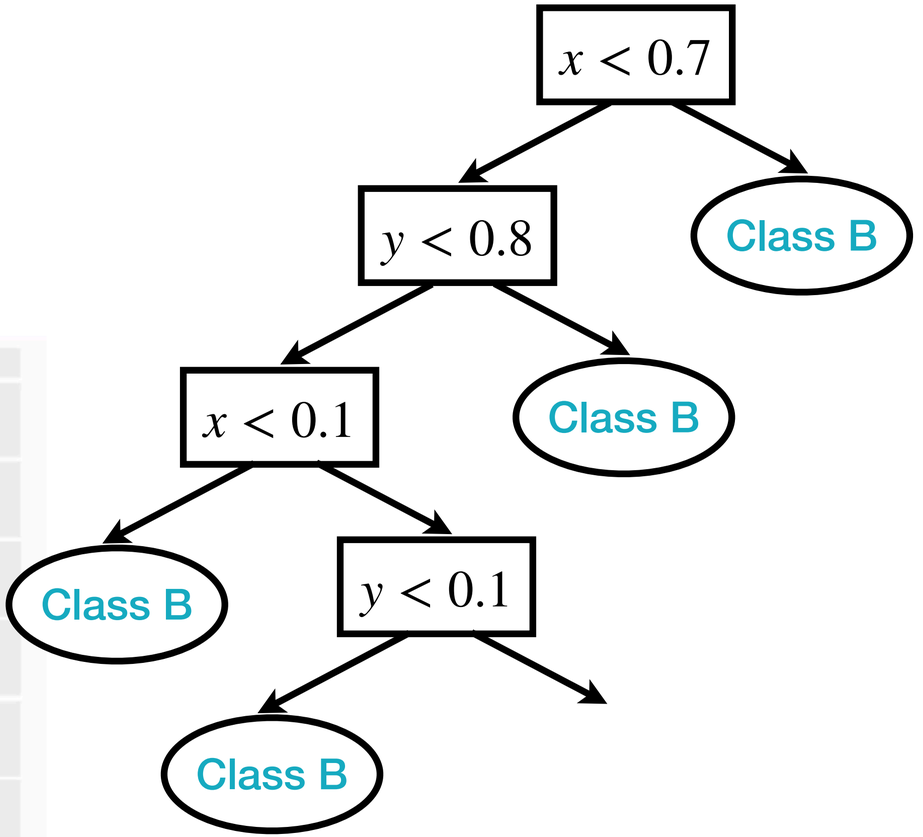
General idea about Trees



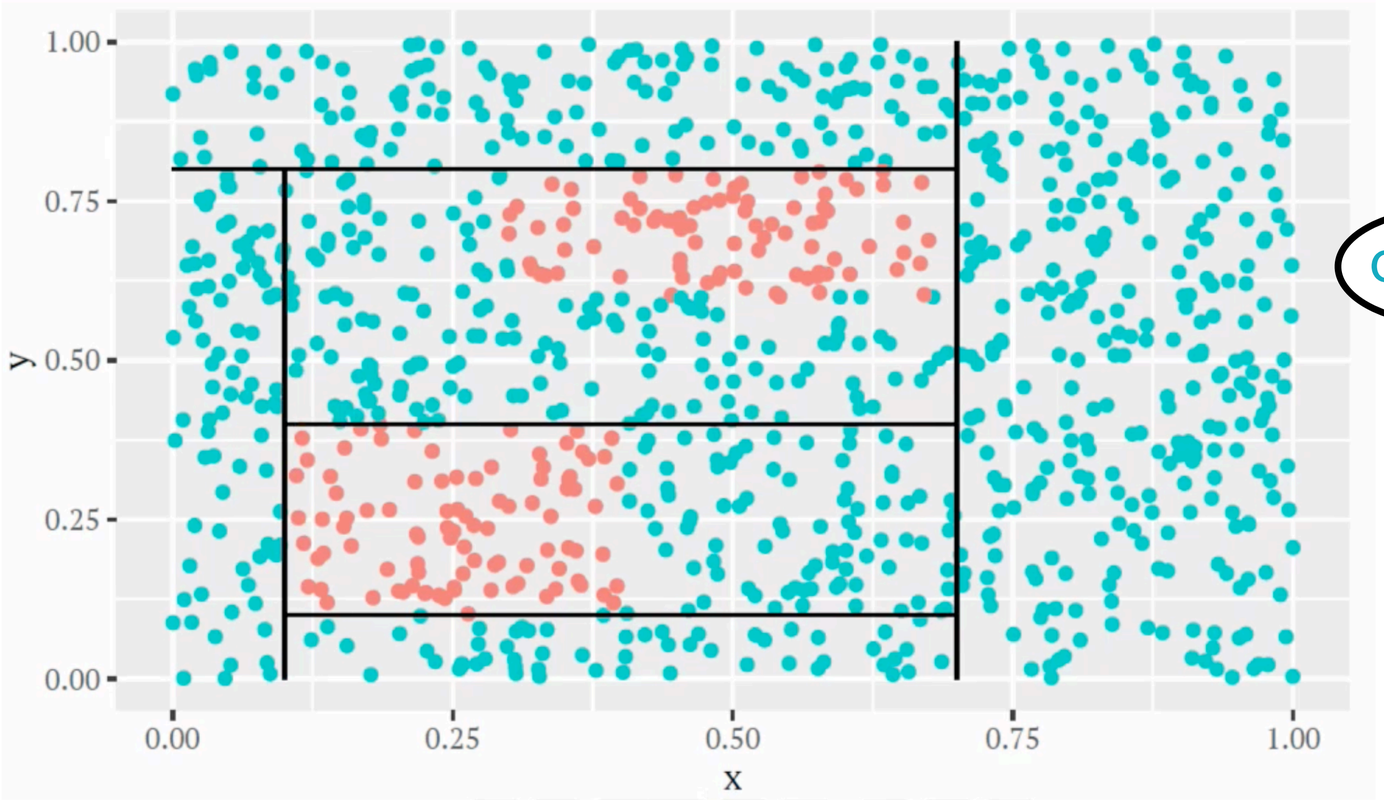


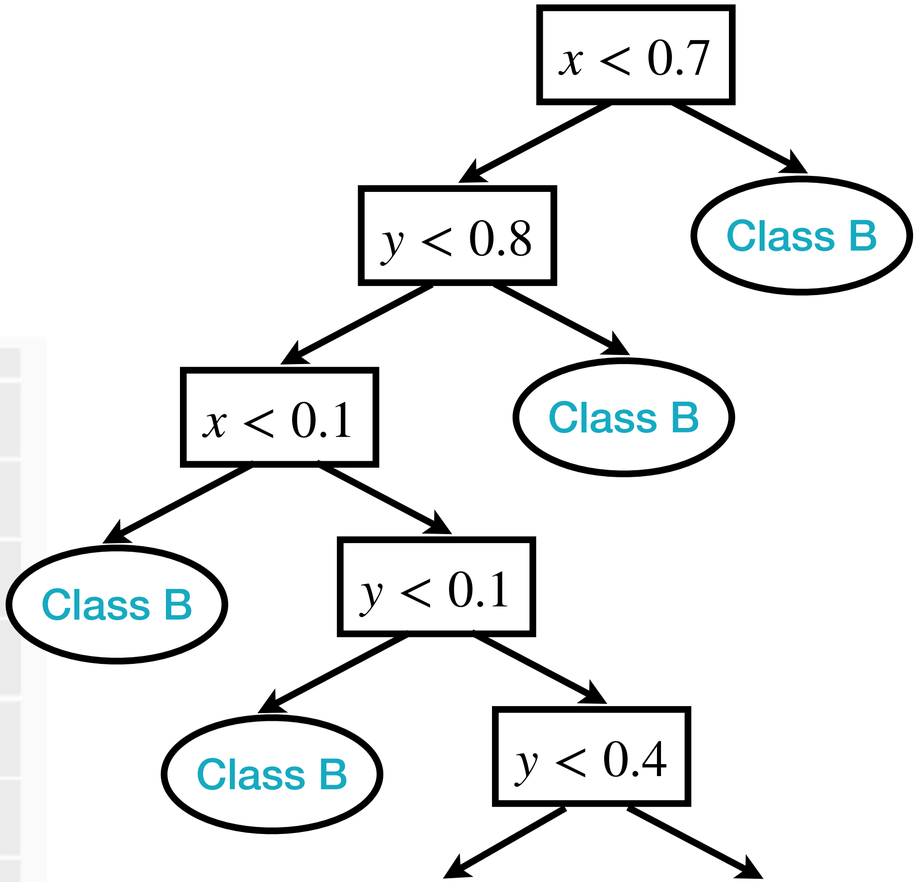
General idea about Trees



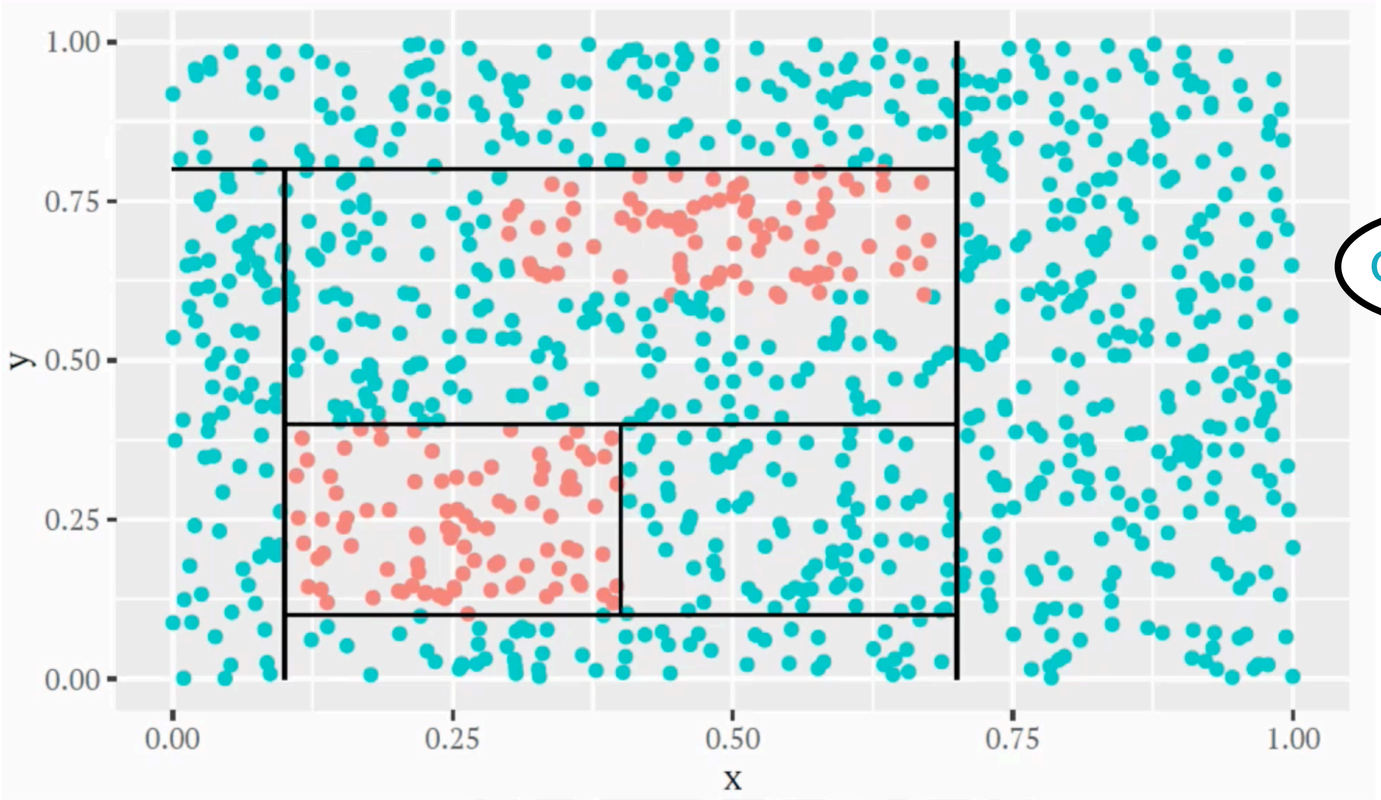


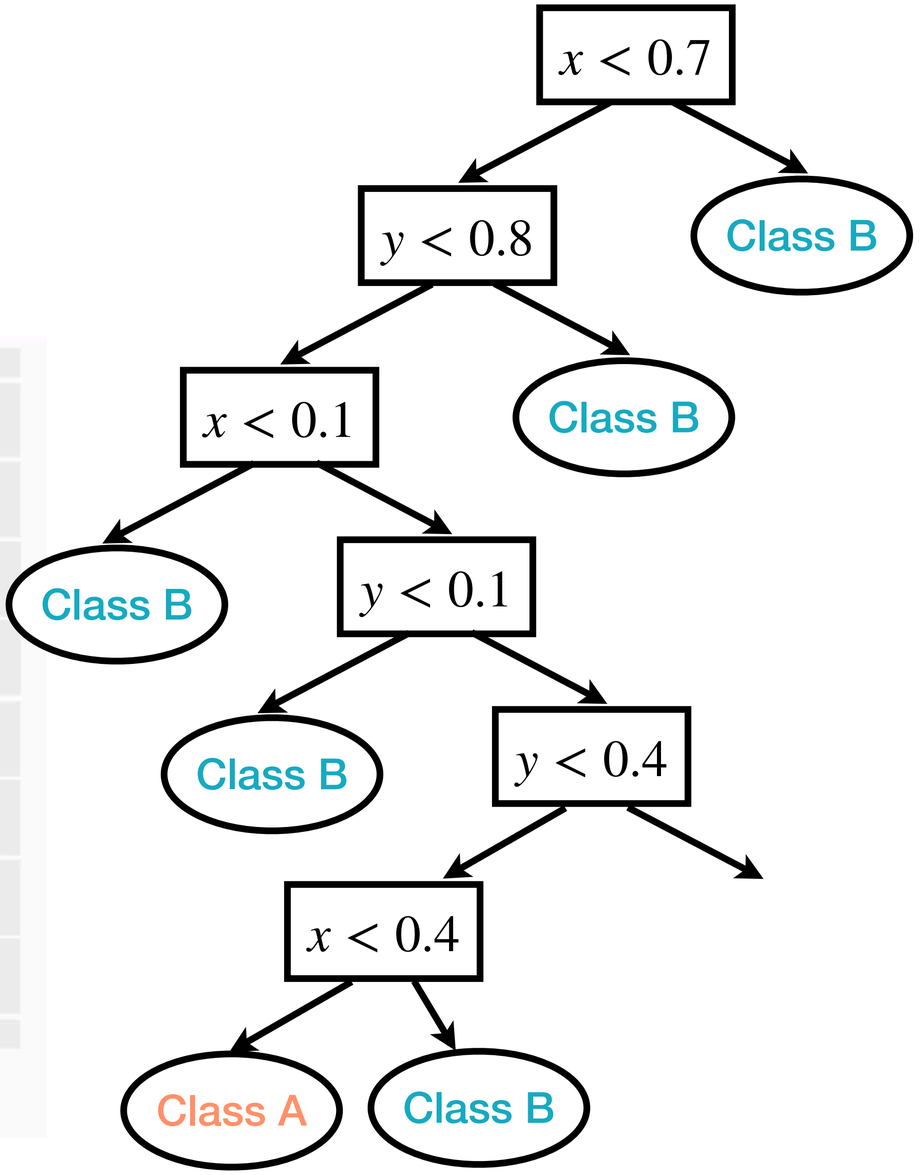
General idea about Trees



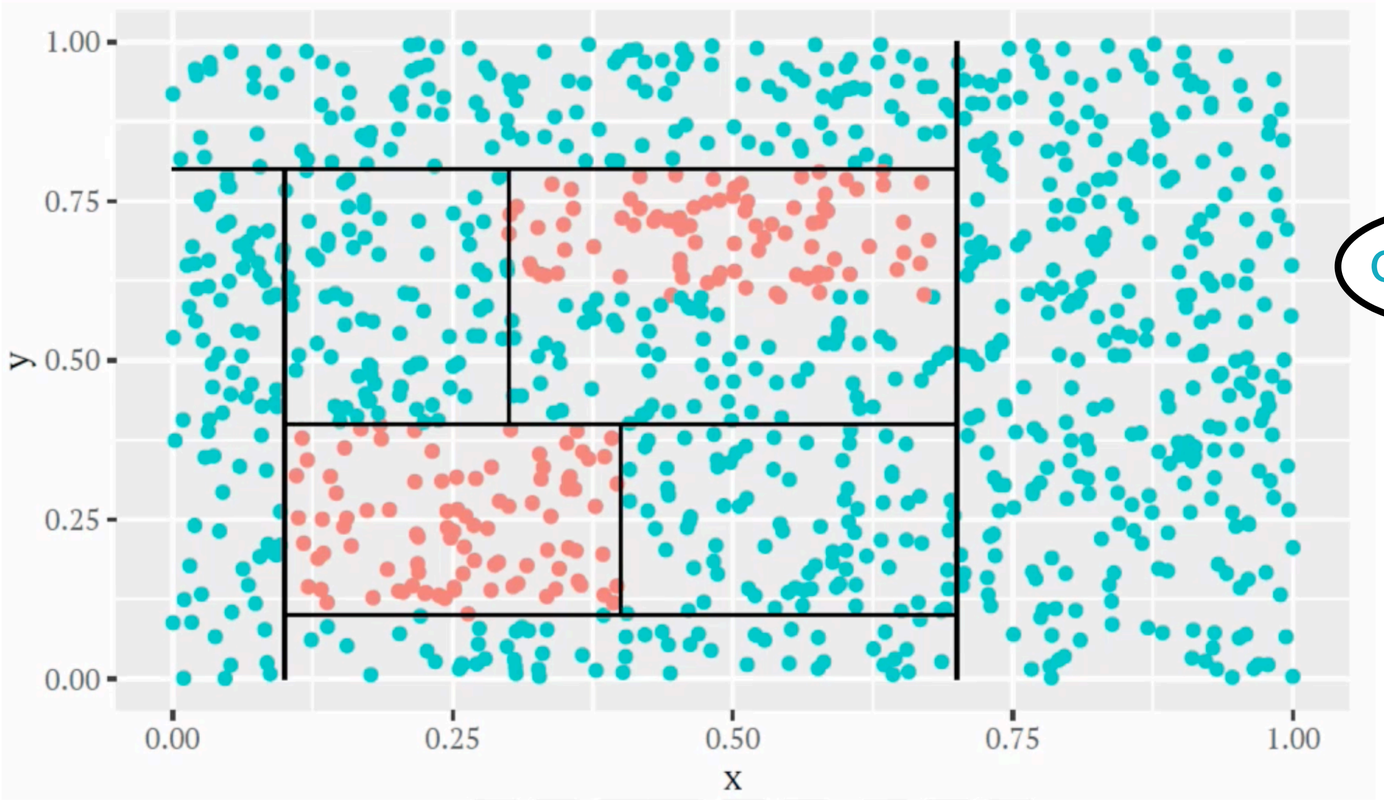


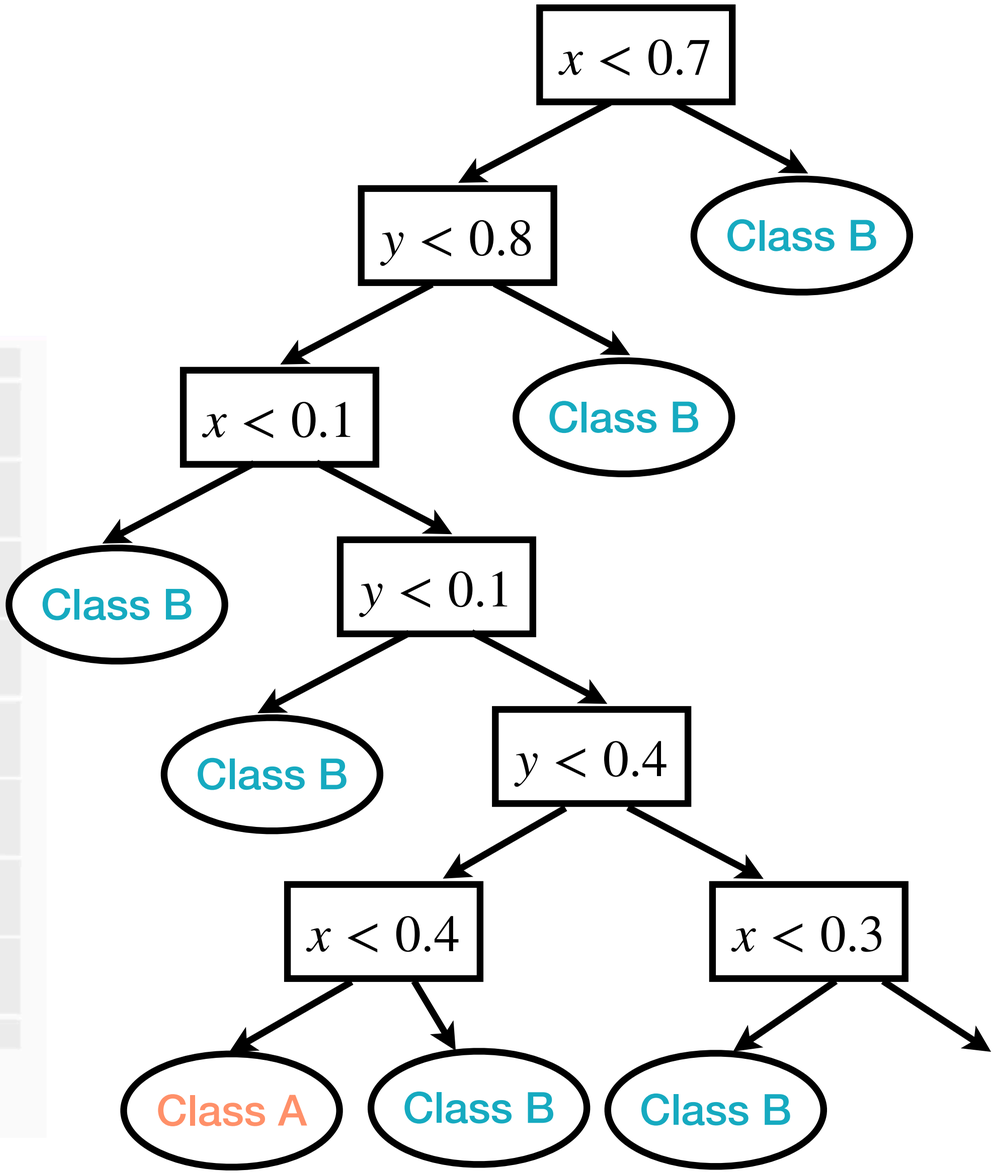
General idea about Trees



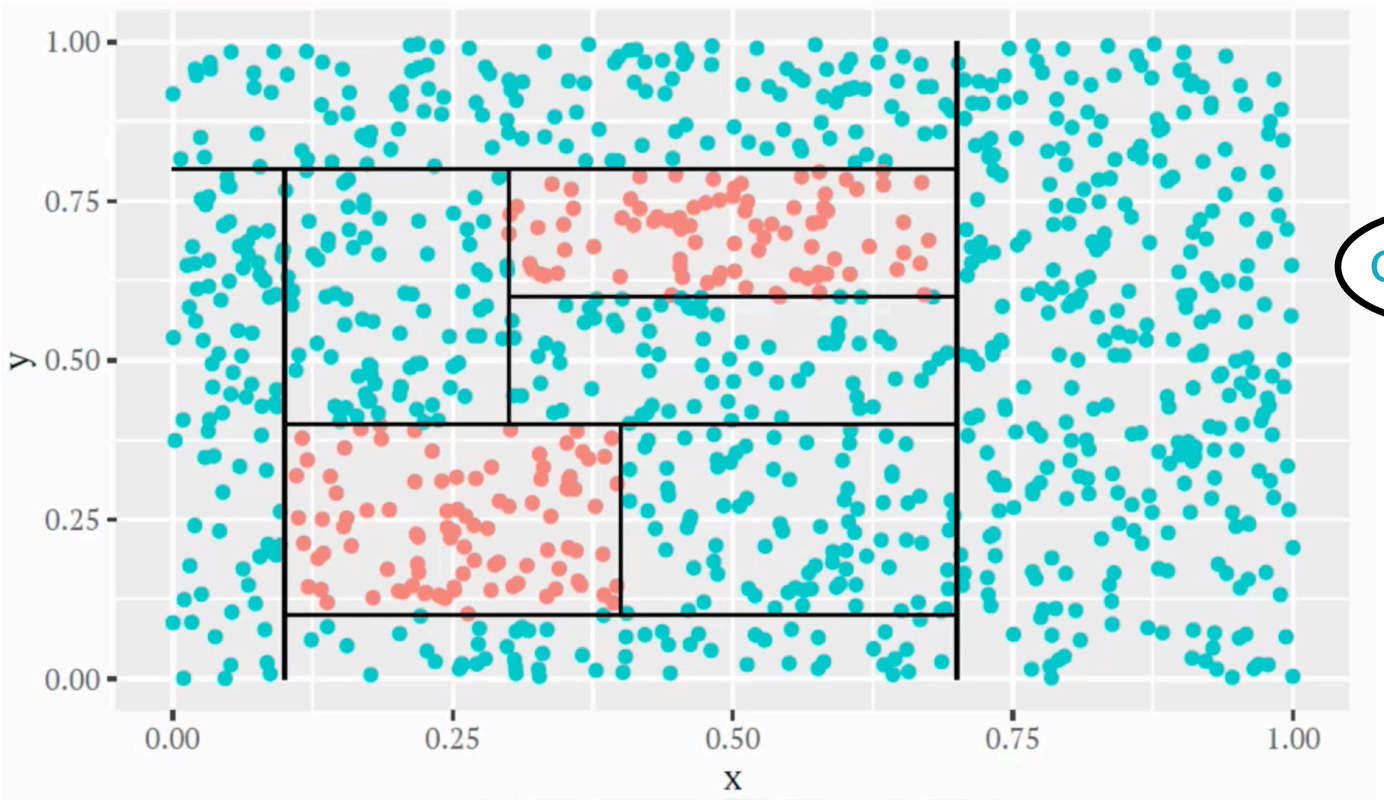


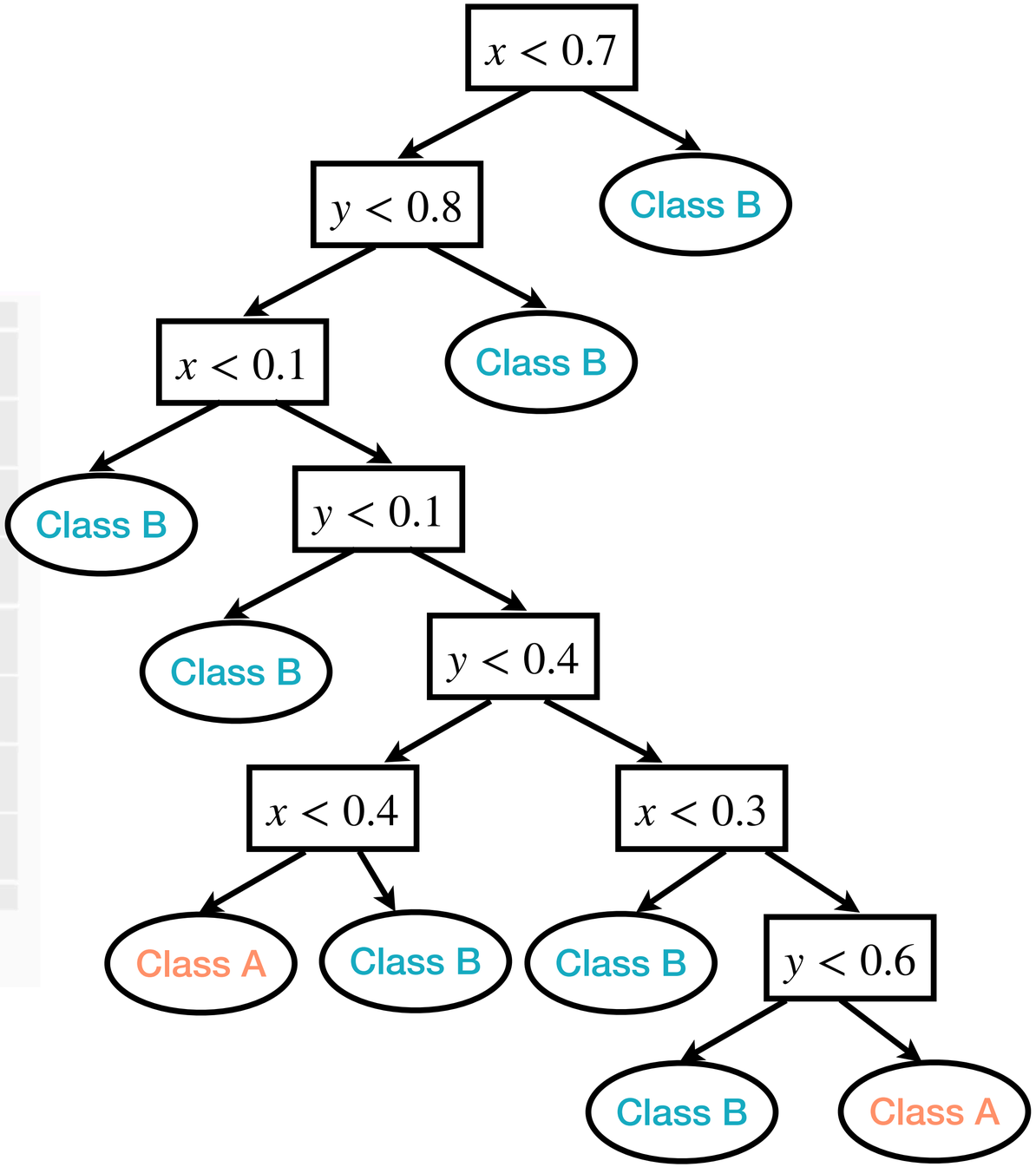
General idea about Trees





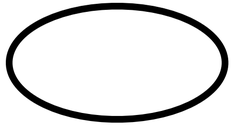
General idea about Trees





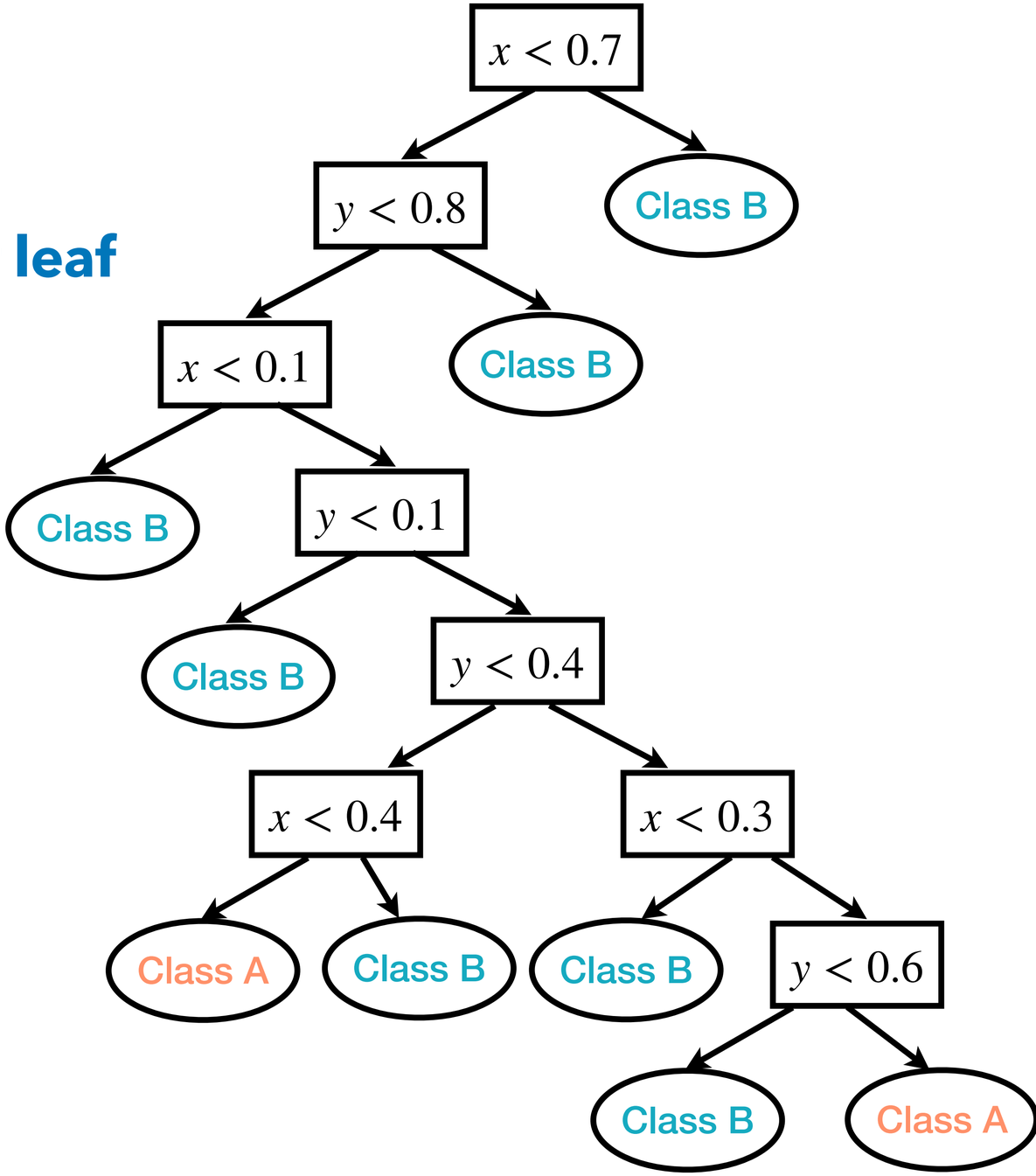
Definition

internal node



terminal node or leaf

8 internal nodes and
9 terminal nodes (leaves)
in this tree



Prediction using classification tree

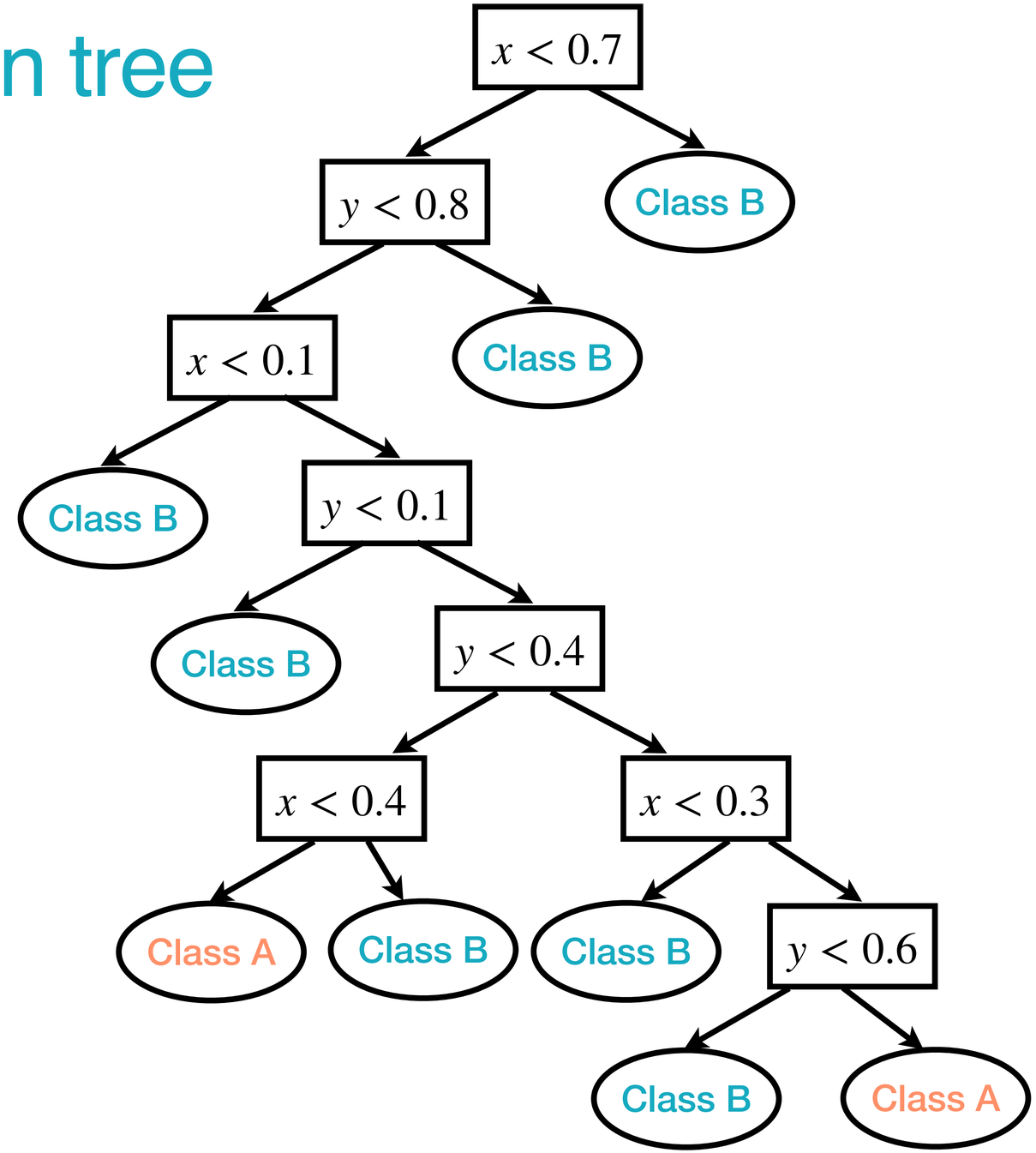
Given a new input $\{x^*, y^*\}$ belongs to a terminal node region

i) report the most commonly occurring class in the terminal node region;

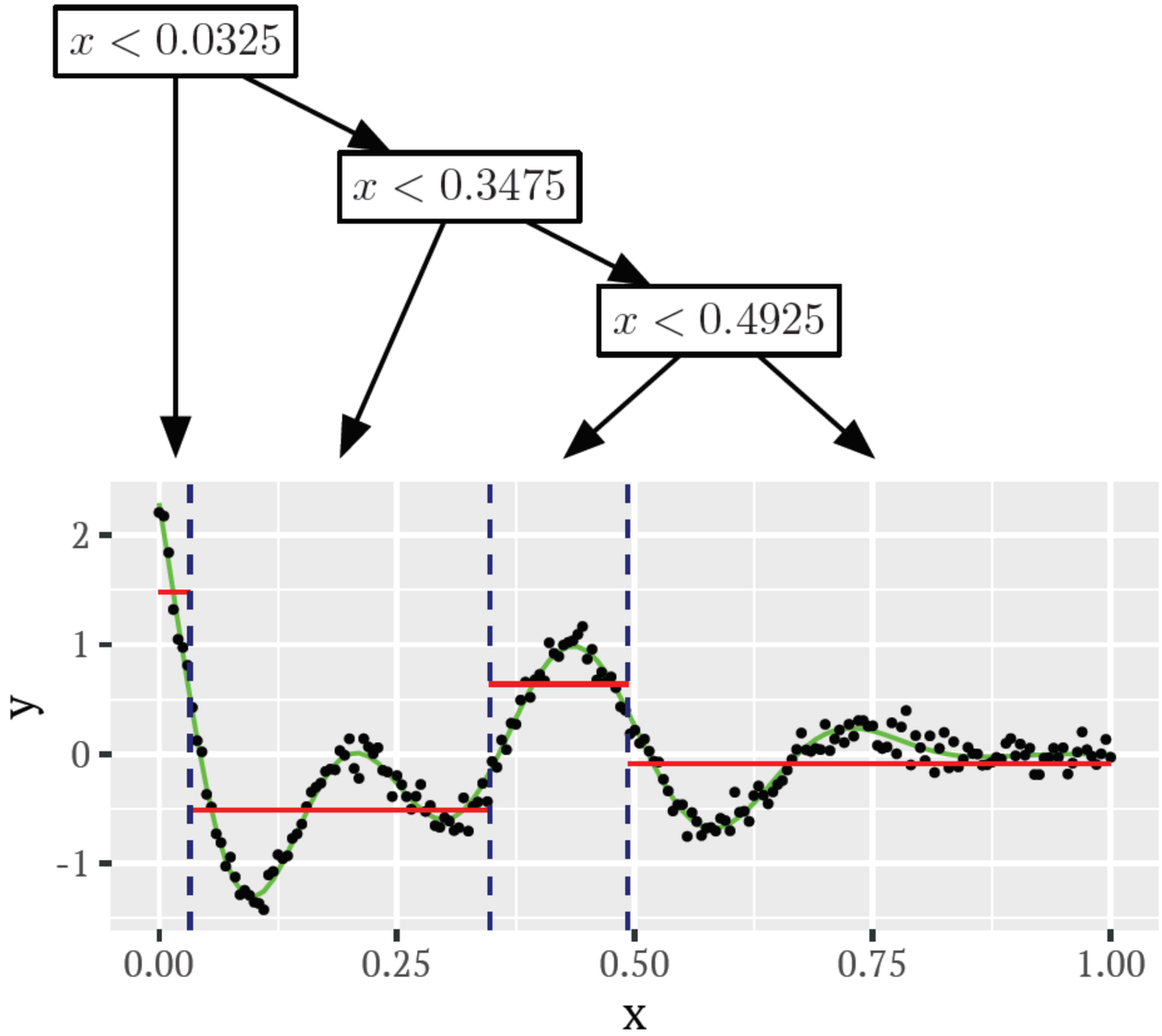
Class A or Class B or Class C….

ii) report the class proportions that fall into the terminal node region;

$\hat{p}_{ClassA}$ and $\hat{p}_{ClassB}$ and $\hat{p}_{ClassC}$….



Similar idea for regression tree

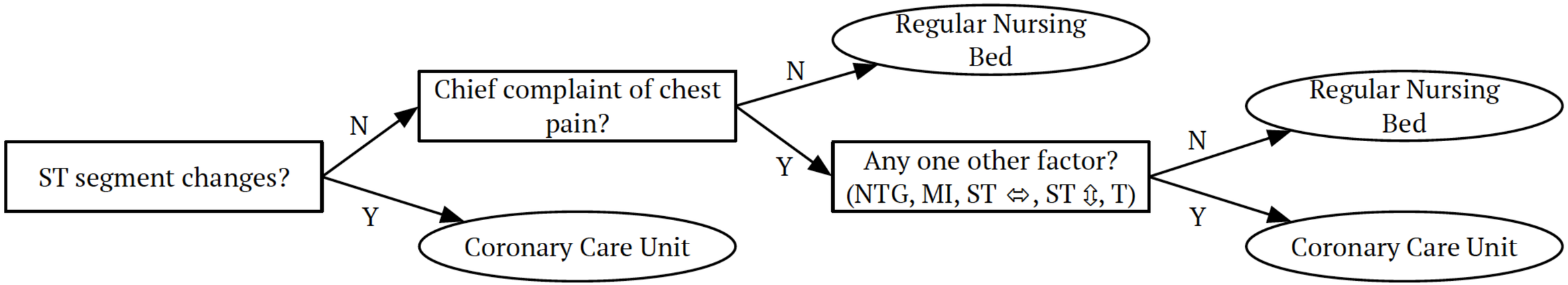


Prediction:

report the mean of the response for the observations that fall in the *leaf* node

Interpret as decision trees

Classification trees mimic some decision making processes. For example, the following decision tree is used by A&E doctors to decide what bed type to assign:



Decision trees can be constructed even in the presence of qualitative predictor variables.

This perhaps makes them popular for the feeling of interpretability and of being like a data-learned expert system.

Definition

Classification and Regression Trees (CART)

A tree ends in a *leaf* node which corresponds to some hyper-rectangle in **feature (predictor) space**, R_j, $j = \{1,...,J\}$, i.e. the feature space defined by $X_1, X_2, \ldots, X_p$ is divided into J distinct and non-overlapping regions.

Classification assigns a class label to each R_j

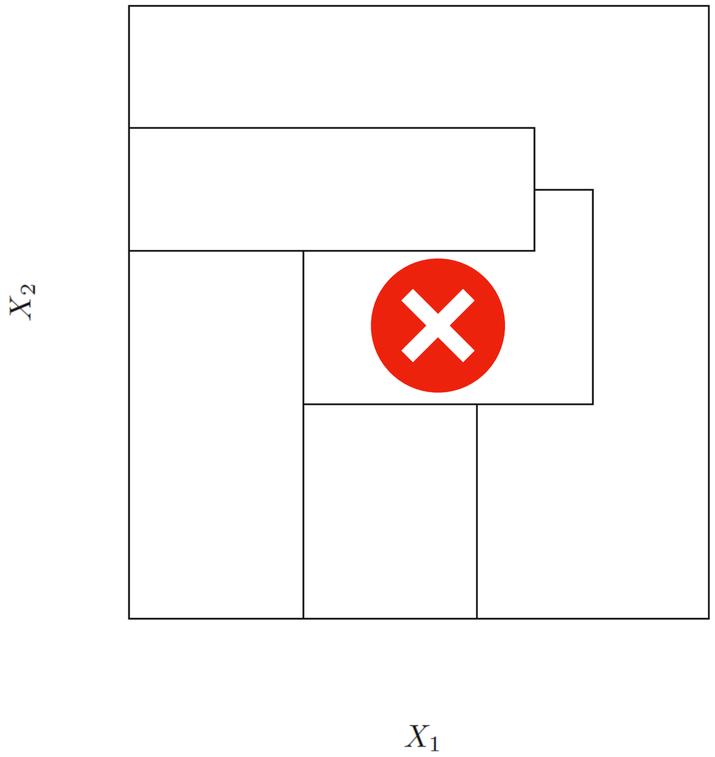
Regression assigns a value to each R_j

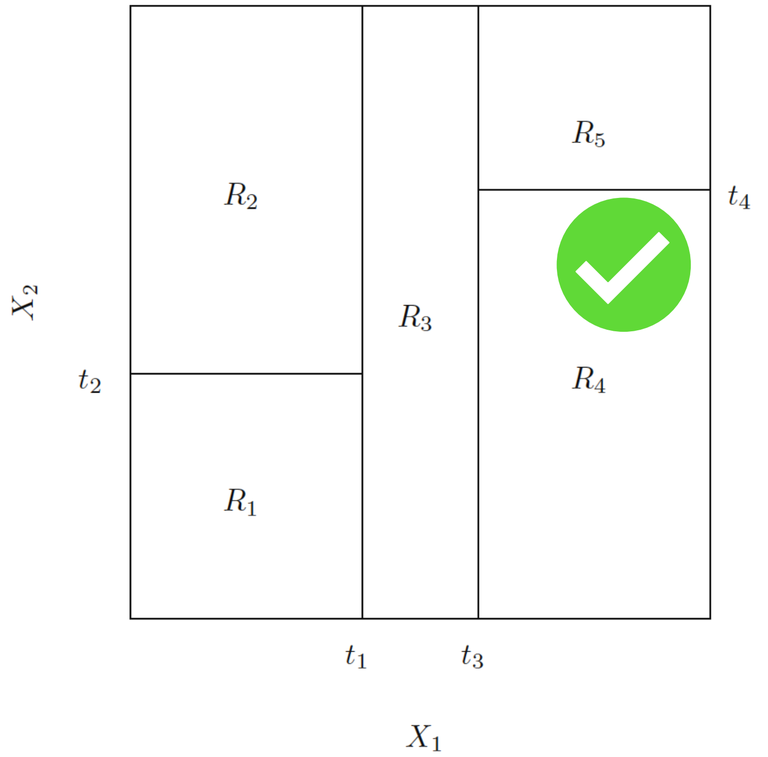
every observation **x** falling in R_j is given the same predicted value.

Methodology

Building trees

- In theory, the regions R_j could have any shape. But we choose hyper (high-dimensional)-rectangles (boxes) for simplicity.





Methodology

Building trees

- Goal: for regression trees, find R_j, $j = \{1,...,J\}$ that minimize the RSS:

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

where $\hat{y}_{R_j}$ is the mean response for the training observations within the j^{th} rectangle.

Methodology

Building trees

- Goal: For classification trees, we may adopt *classification error rate:*

$$E = 1 - \max_k(\hat{p}_{mk}),$$

where $\hat{p}_{mk}$ represents the proportion of training observations in the m^{th} region that are from the k^{th} class.

Methodology

Building trees

- Goal: For classification trees, we may adopt *classification error rate:*

$$E = 1 - \max_k(\hat{p}_{mk}),$$

Other objective functions:

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

G is a measure of total variance across the K classes.

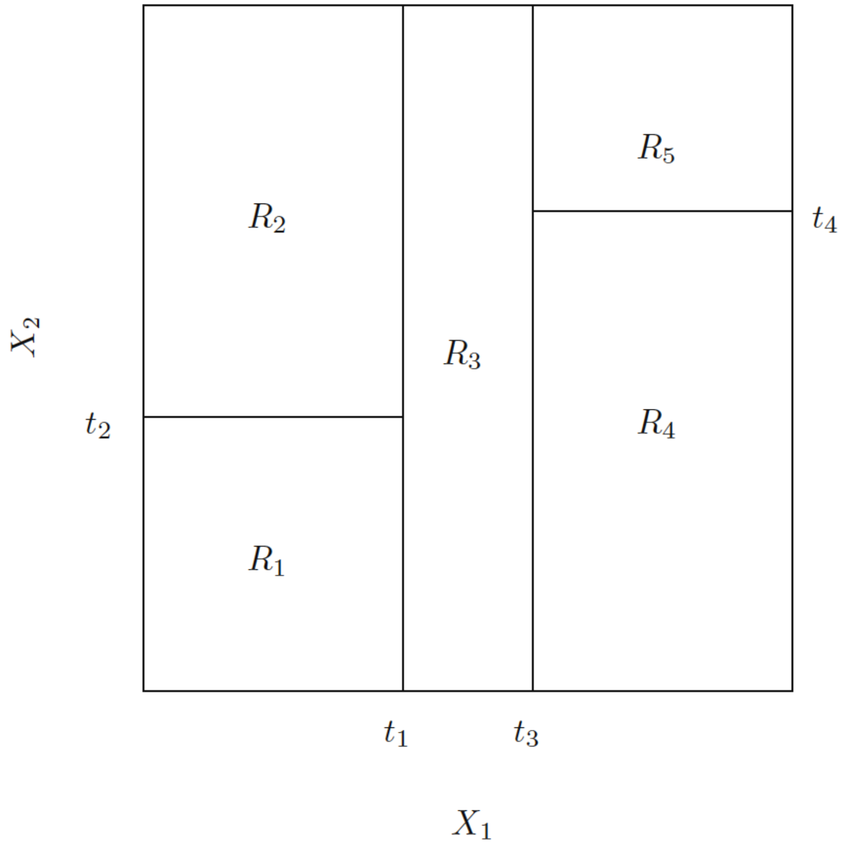
measures of ***purity***

ii) cross-entropy: $$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

Methodology

Building trees

- It is computationally infeasible to consider every possible partition of the feature space into J boxes.



Think about how many ways one could divide this 2-D space into 5 boxes. Not to mention high dimensional feature space.

Methodology

Building trees

- ***greedy (top-down)* fitting algorithm**

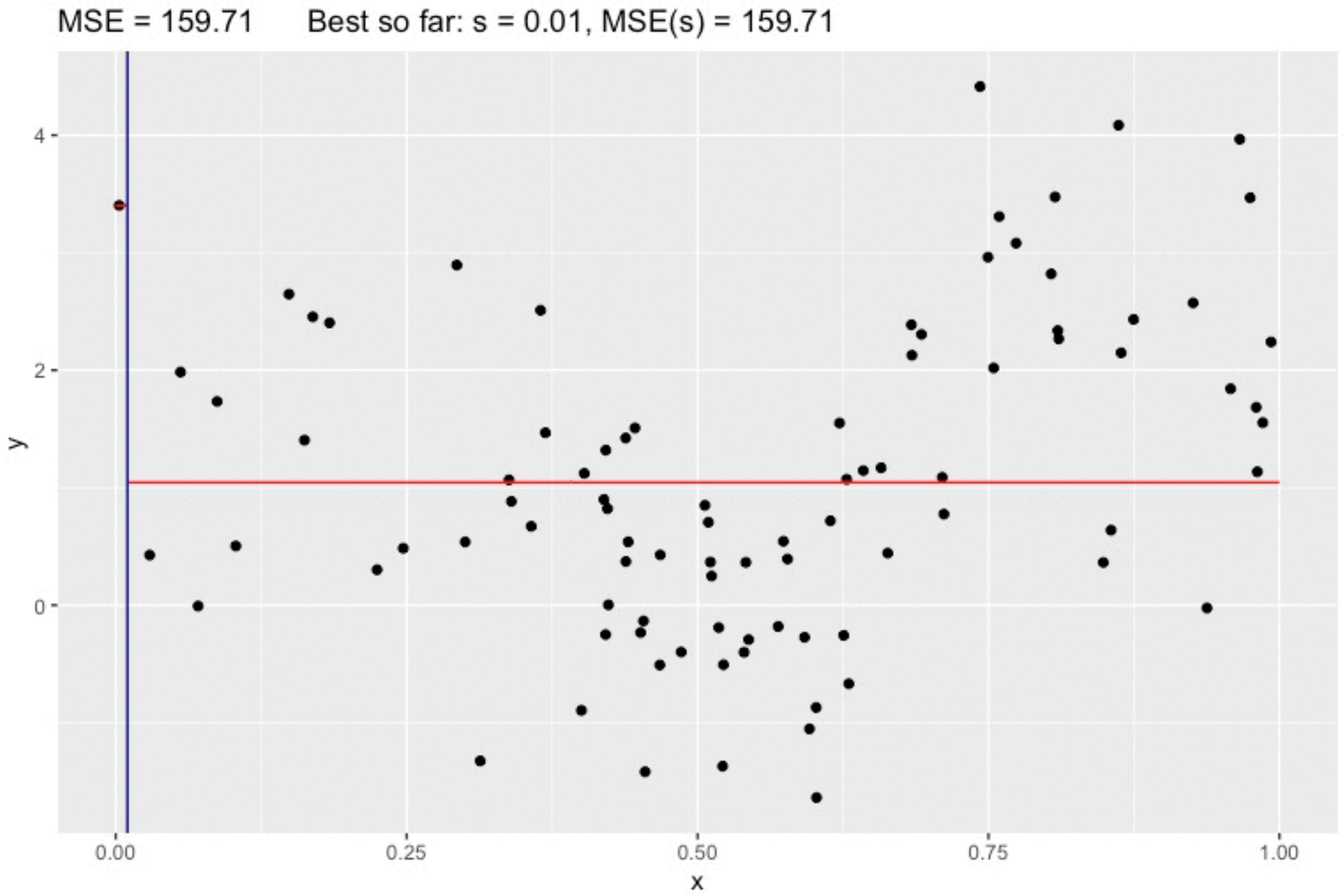
based on some objective function

1. Initialise hyper-rectangle, $R = \mathbb{R}^d$.
2. Find the *best* split on variable j at location s, gives: $R_1(j,s) = \{x \in R : x_j < s\}$ and $R_2(j,s) = \{x \in R : x_j \geq s\}$
3. Repeat step 2. twice, once with $R = R_1(j,s)$ and once with $R = R_2(j,s)$

There will be some stopping rule, for example require a minimum number (for example, 5) of training observations in each hyper-rectangle (box).

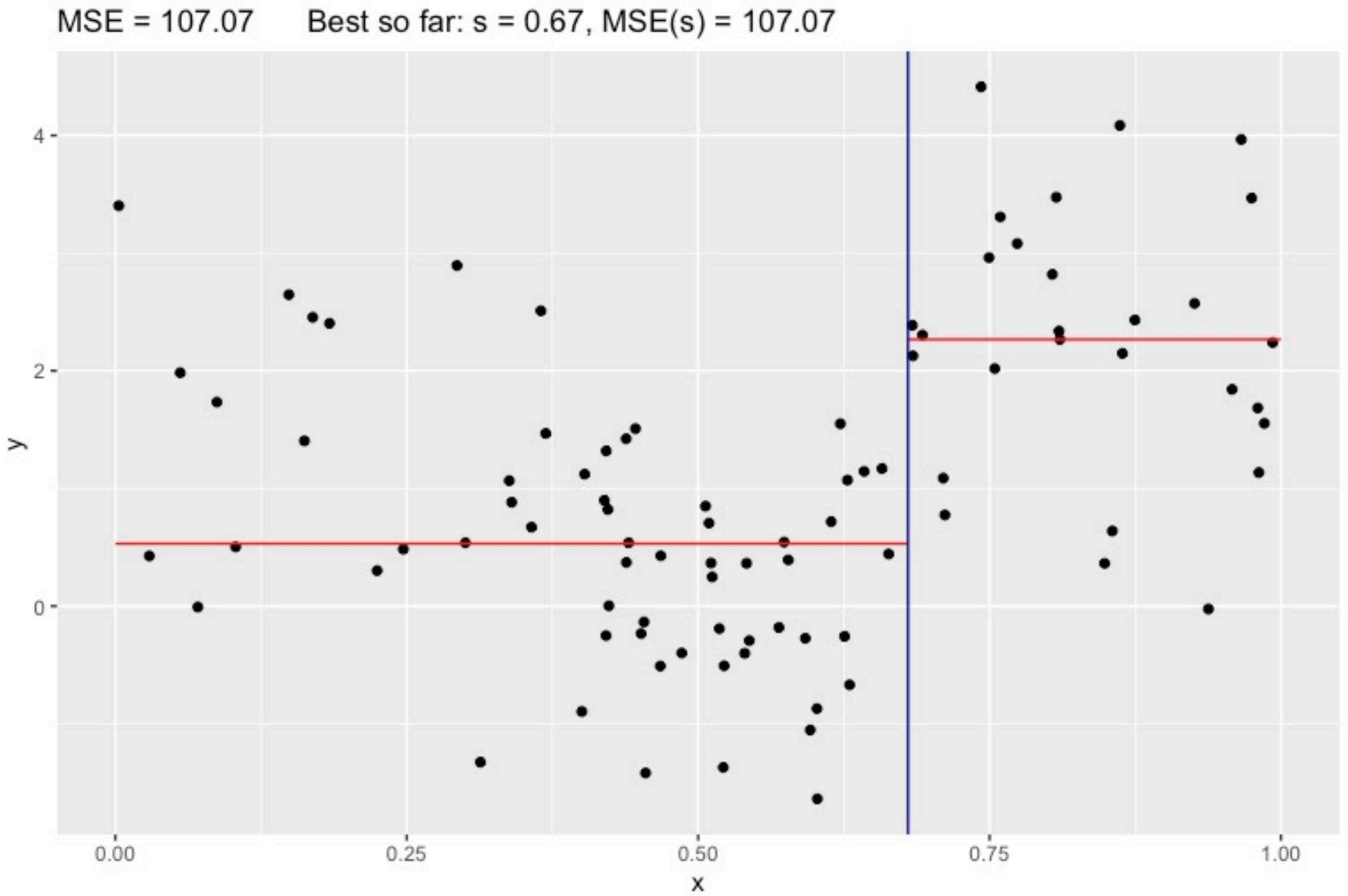
Methodology

Building trees



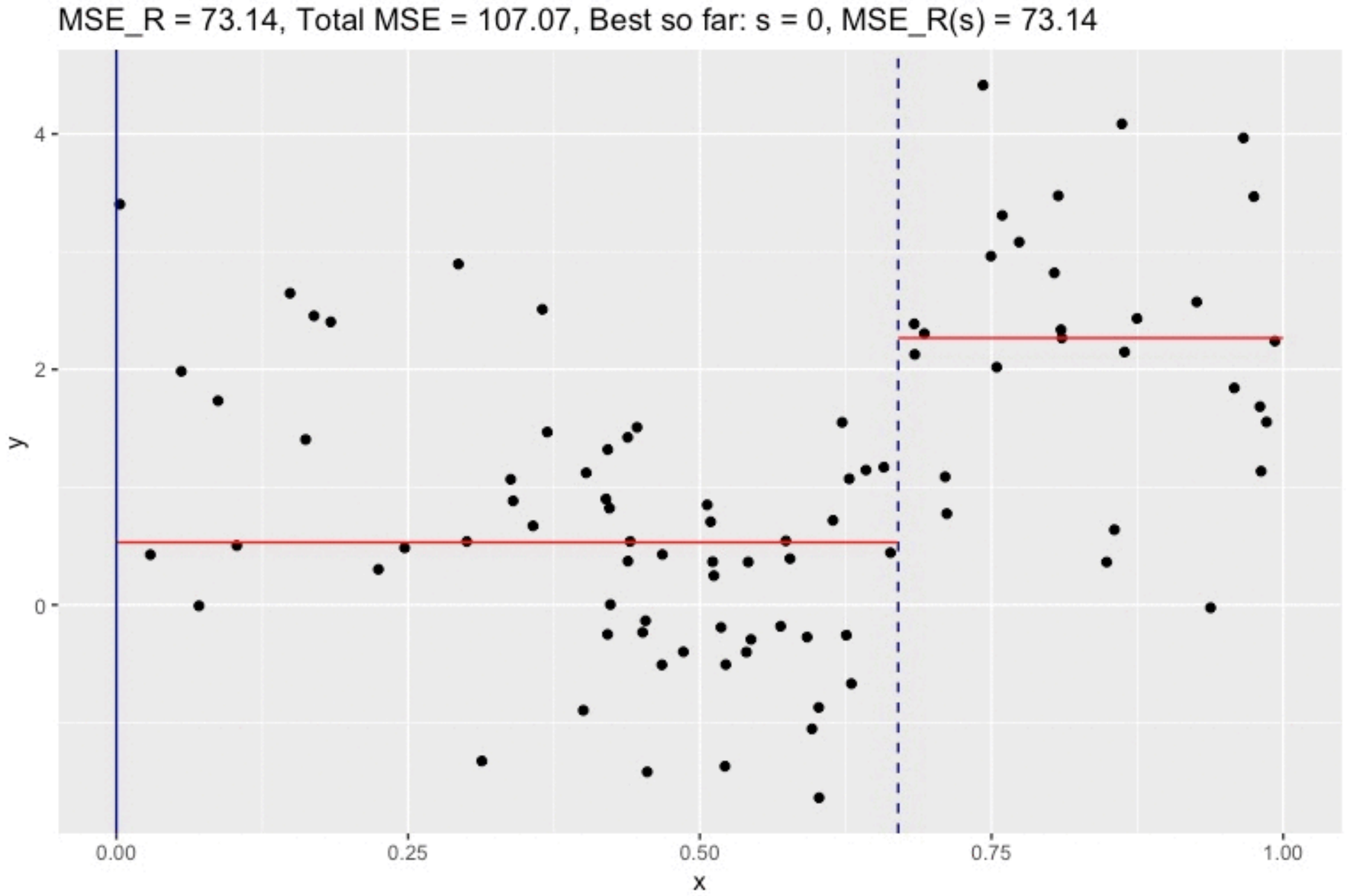
Methodology

Building trees



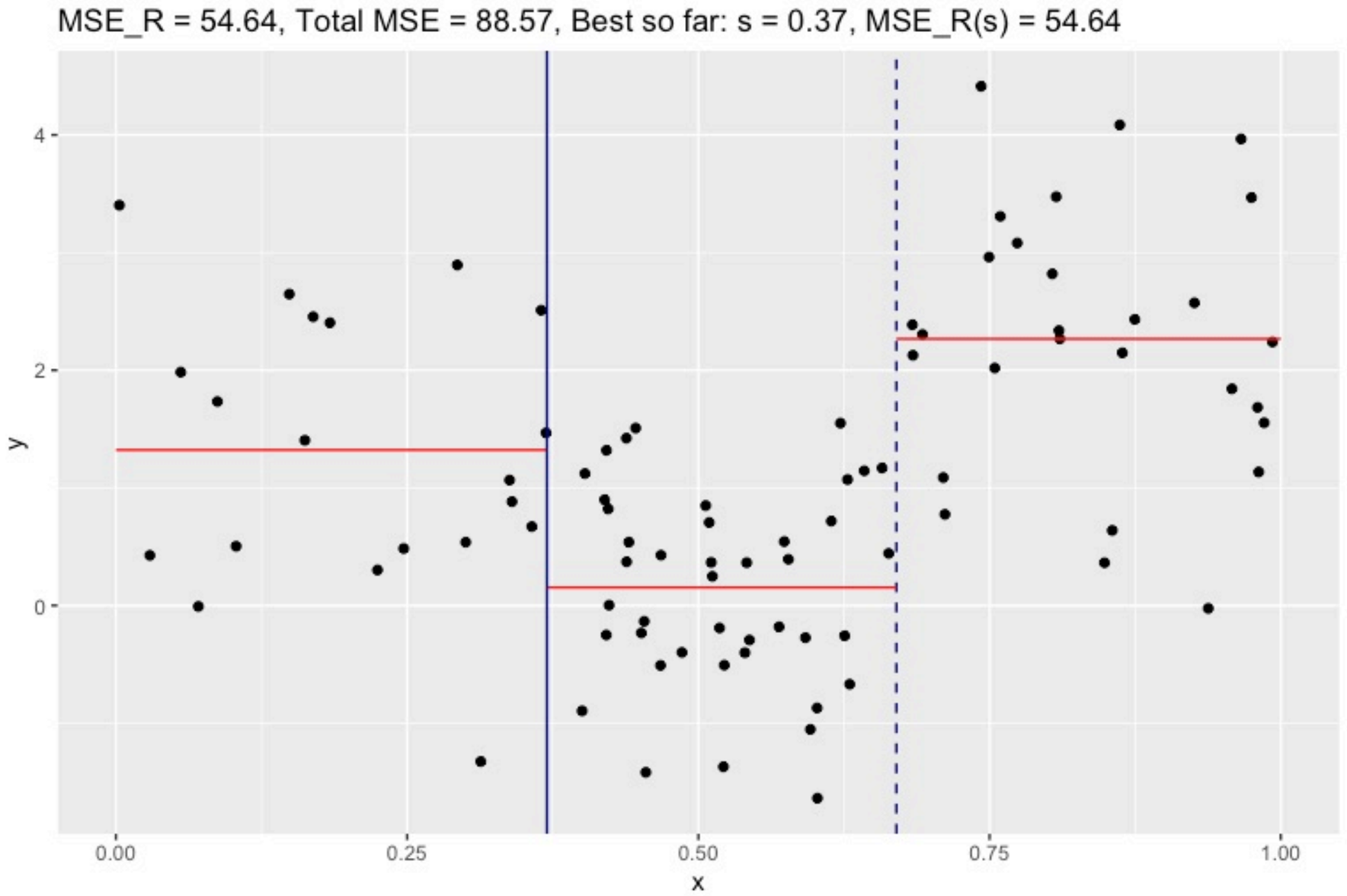
Methodology

Building trees



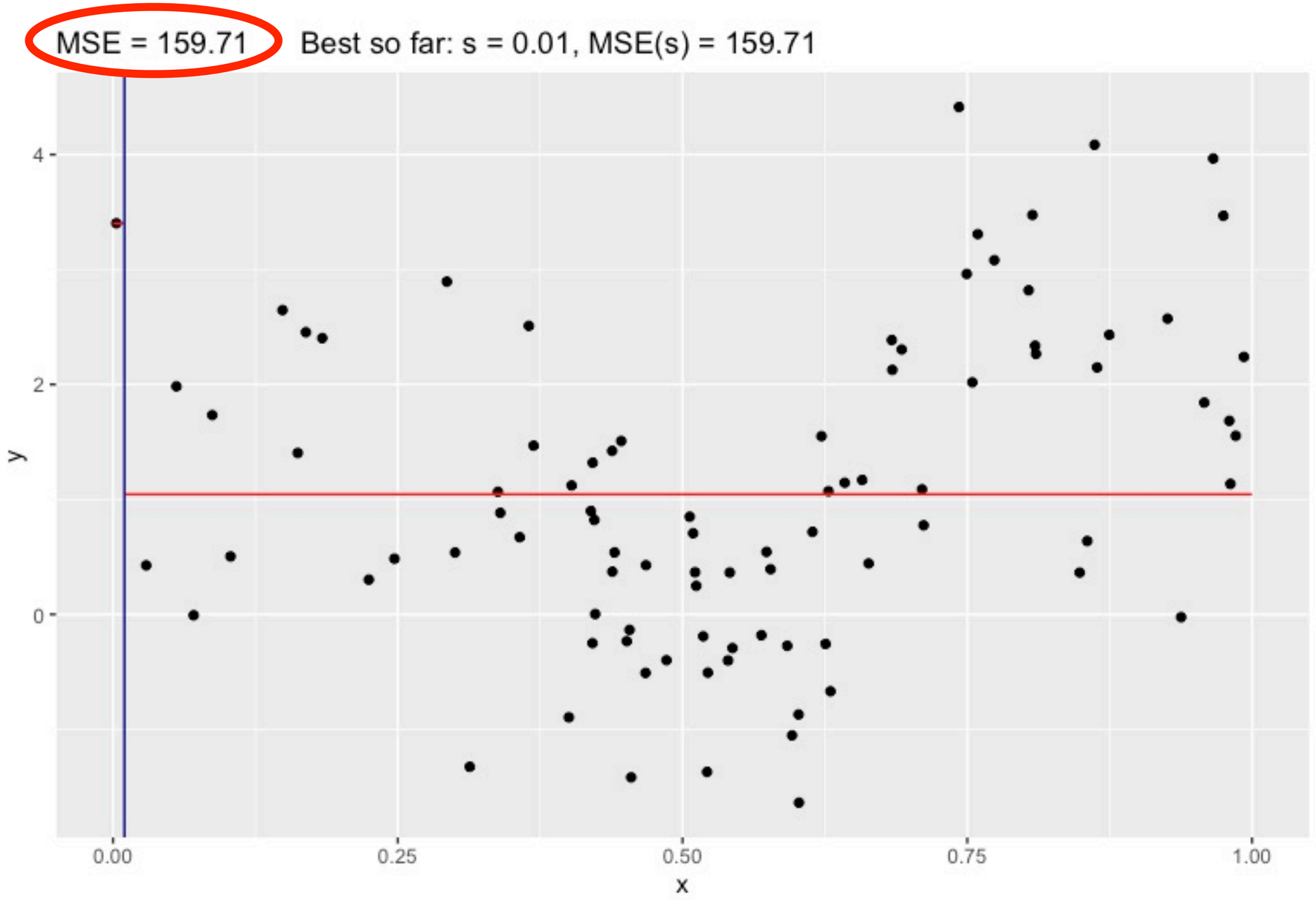
Methodology

Building trees



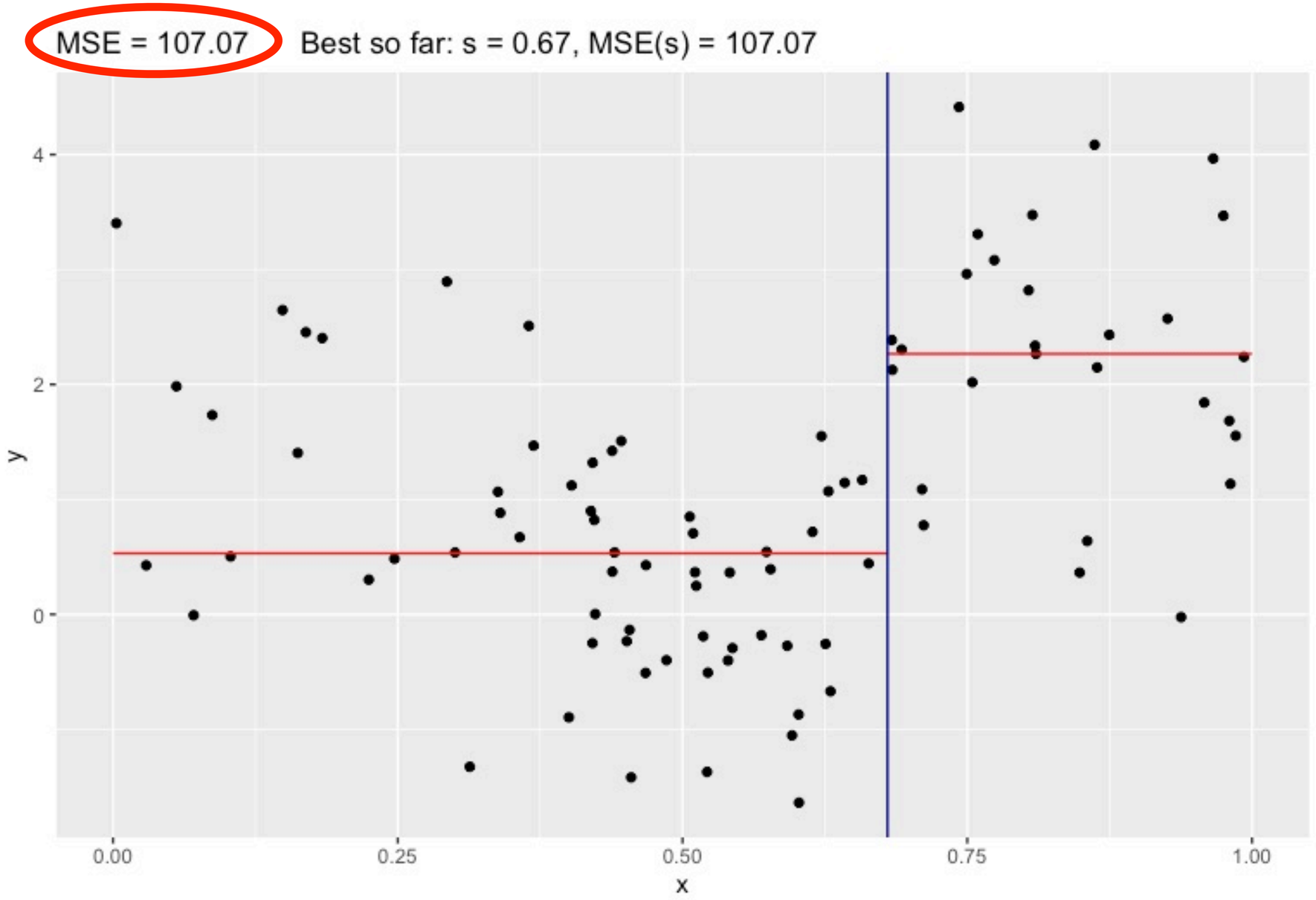
Methodology

Building trees



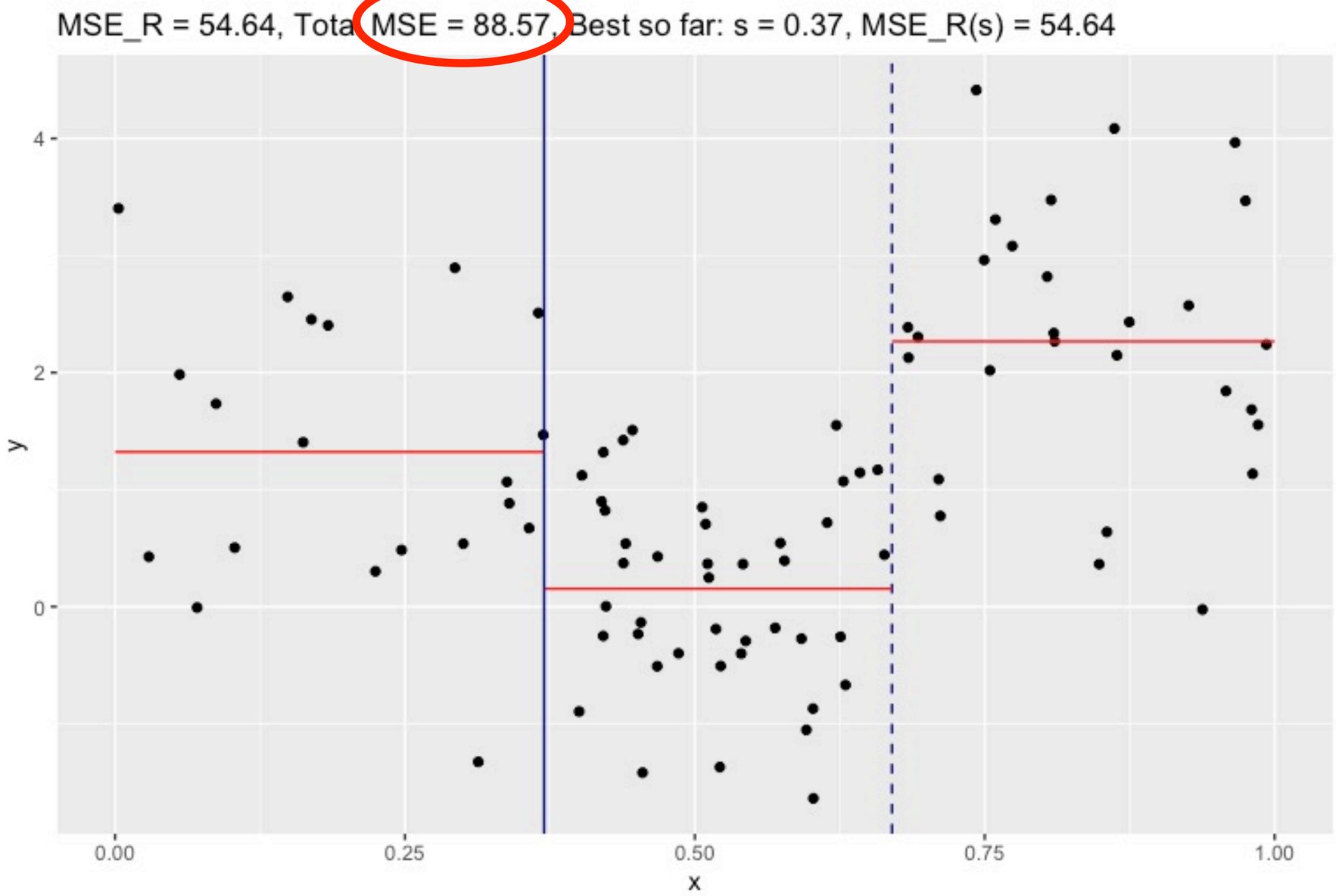
Methodology

Building trees



Methodology

Building trees



Methodology

Building trees

- ***greedy (top-down)* fitting algorithm**

based on some objective function

1. Initialise hyper-rectangle, $R = \mathbb{R}^d$.
2. Find the *best* split on variable j at location s, gives: $R_1(j,s) = \{x \in R : x_j < s\}$ and $R_2(j,s) = \{x \in R : x_j \geq s\}$
3. Repeat step 2. twice, once with $R = R_1(j,s)$ and once with $R = R_2(j,s)$

There will be some stopping rule, for example require a minimum number (for example, 5) of training observations in each hyper-rectangle (box).

Methodology

Building trees

- ***pruning*** a tree to avoid overfitting

The fitting procedure described above may produce good predictions on the training set, but is likely to overfit the data, leading to poor test set performance.

A trade-off between variance and bias!

To balance this trade-off and avoid overfitting, prune trees (weakest link pruning), removing weakest nodes.

Methodology

Building trees

- *weakest link pruning (cost complexity pruning)*

  For a nonnegative tuning parameter α, minimise

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

  where $|T|$ is the number of terminal nodes of the tree T.

  Minimisation is achieved by attempting to remove terminal nodes from the base of the tree upward.

  α is a penalty on the size (complexity) of the tree

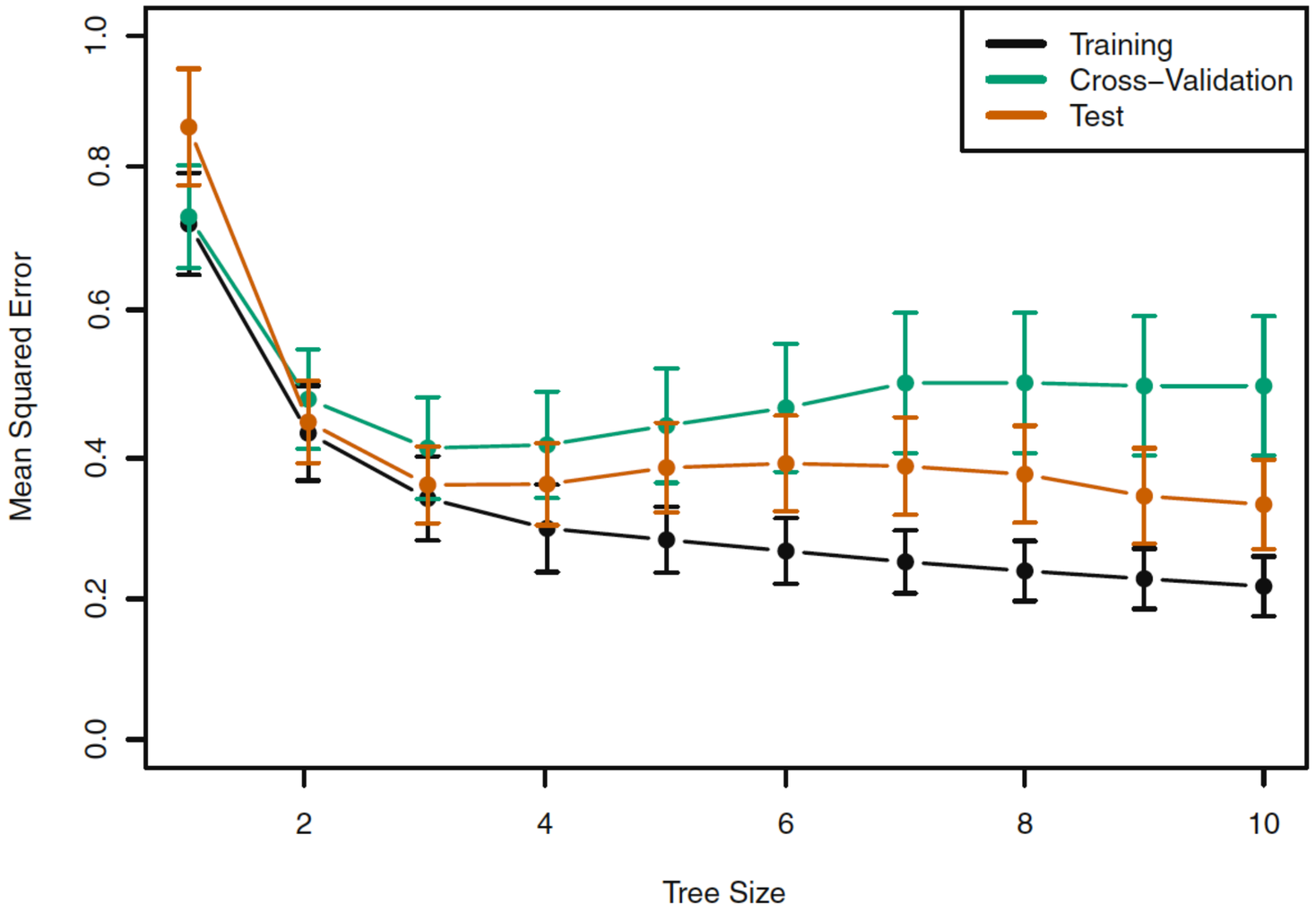
  - $\alpha = 0 \implies$ keep whole tree;
  - $\alpha = \infty \implies$ prune everything back to null tree.

Select α using cross-validation

Methodology

Building trees

- *weakest link pruning (cost complexity pruning)*



CART discussion

Pros

- Simple and easy to interpret
- Akin to common decision making processes
- Good visualisations
- Easy to handle qualitative predictors (avoid dummy variables)

Cons

- Trees tend to have high variance…

  …small changes in the sample lead to

  dramatic cascading changes in the fit!
- poor prediction accuracy.

Bagging (Bootstrap aggregating)

Have seen CART has attractive features, but biggest problem is high variance.

We now explore strategies for stabilising trees and lowering the variance.

- + will improve prediction accuracy
- - will sacrifice interpretability

Tools

Bootstrap

A bootstrap sample of size m is:

$$(y_i^\star, \mathbf{x}_i^\star)_{i=1}^m$$

where each $(y_i^\star, \mathbf{x}_i^\star)$ is a uniformly random draw from the training data $(y_1, \mathbf{x}_1), \ldots, (y_n, \mathbf{x}_n)$

- Resample with replacement!
- Samples in the original data set may appear more than once in the bootstrap sample
- Still sample $(y, \mathbf{x})$ in pairs, relationship between y and $\mathbf{x}$ is reserved.

Tools

Bootstrap

$(y_i^\star, \mathbf{x}_i^\star)_{i=1}^m$ are an approximation to drawing iid samples from $\mathbb{P}(\mathbf{X}, Y)$.

If we set $m = n$, then notionally we sampled whole new training set. However, we will only have on average $\approx 63.2\,\%$ of the original training data in the bootstrap resample.

Why 63.2%????

Maths

If we set $m = n$, then we will only have on average $\approx 63.2\,\%$ of the original training data in the bootstrap resample. why $\approx 63.2\,\%$??

$$\begin{aligned}
&\mathbb{P}(\text{sample } i \text{ is in the bootstrap resample}) \\
&\quad = 1 - \mathbb{P}(\text{sample } i \text{ is not in the bootstrap resample}) \\
&\quad = 1 - \mathbb{P}\left((\mathbf{x}_1^\star, y_1^\star) \neq (\mathbf{x}_i, y_i) \cap \cdots \cap (\mathbf{x}_n^\star, y_n^\star) \neq (\mathbf{x}_i, y_i)\right) \\
&\quad = 1 - \prod_{j=1}^{n} \mathbb{P}\left((\mathbf{x}_j^\star, y_j^\star) \neq (\mathbf{x}_i, y_i)\right) \\
&\quad = 1 - \left(1 - \frac{1}{n}\right)^n \\
&\quad \approx 1 - e^{-1} \\
&\quad \approx 0.632
\end{aligned}$$

Tools

Bootstrap

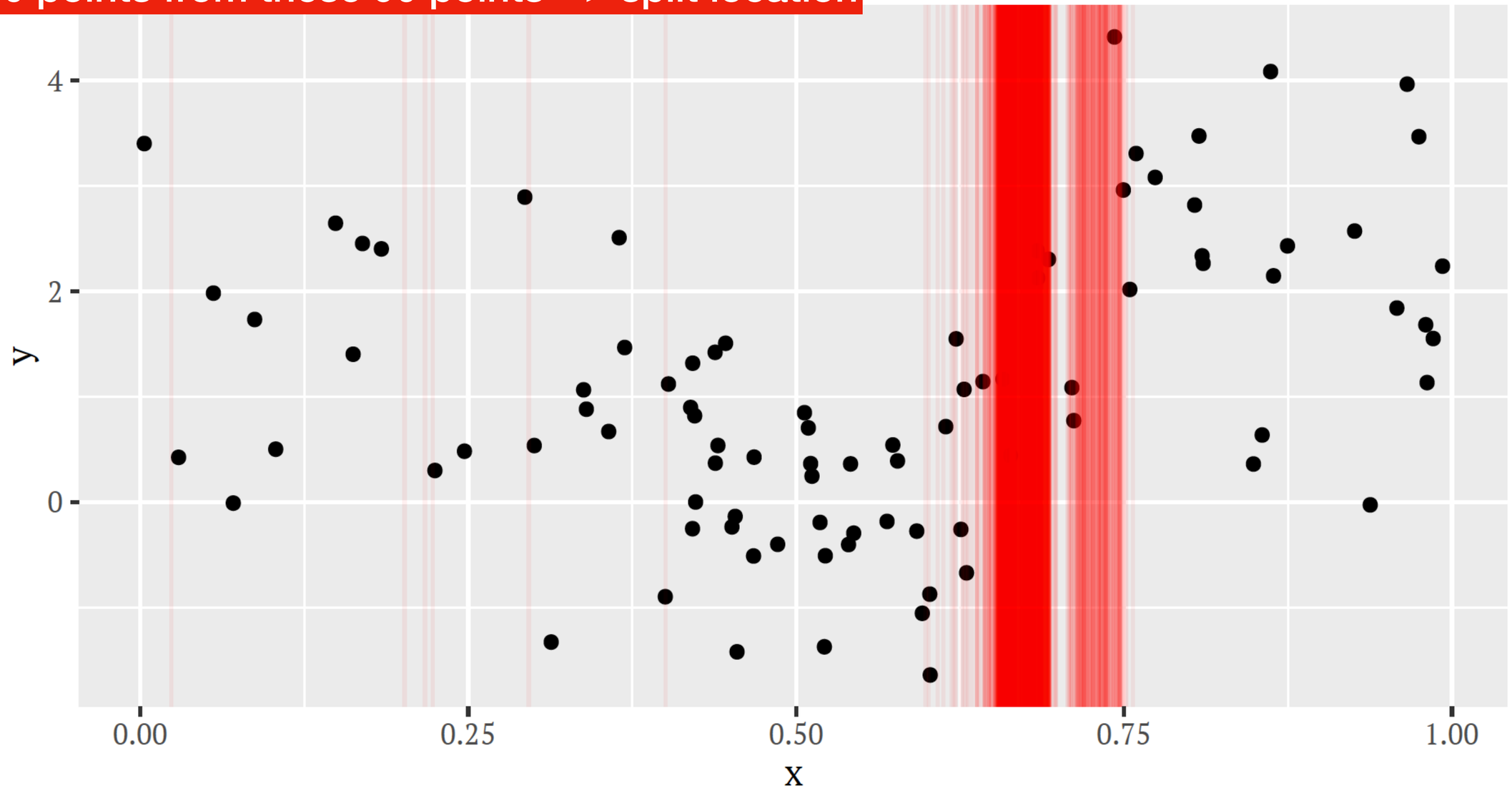
$(y_i^\star, \mathbf{x}_i^\star)_{i=1}^m$ are an approximation to drawing iid samples from $\mathbb{P}(\mathbf{X}, Y)$.

Note: we don't really imagine we have a new iid training sample! But bootstrap methods allow us to produce model-free estimates of sample distributions. If we have a lot of data then the estimate will be quite good.

- intuitively we can roughly approximate the sampling distribution of tress for training sets of size n.
- can achieve variance reduction by averaging many sampled trees.

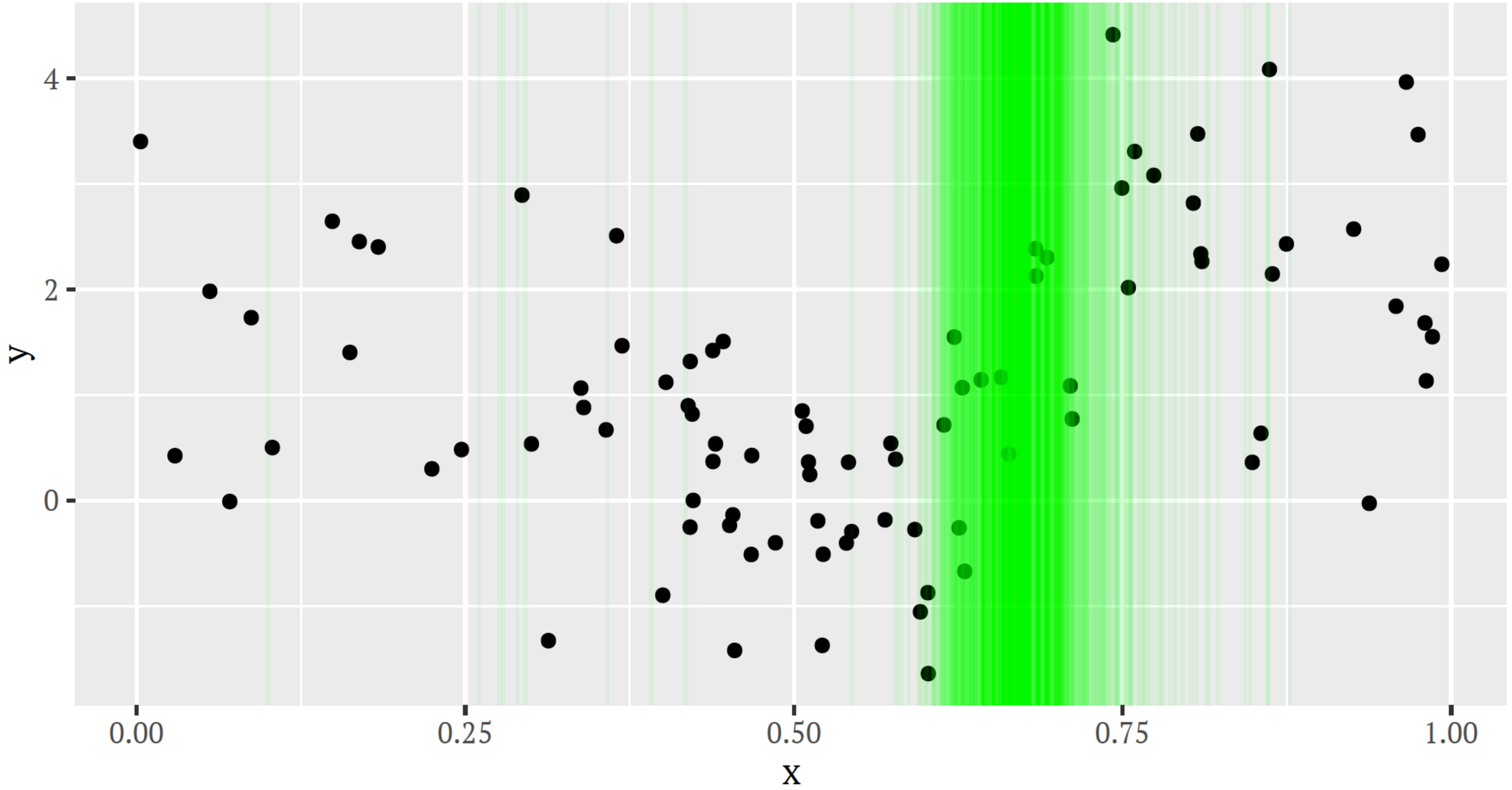
Bootstrap split locations

draw 90 points from these 90 points—> split location

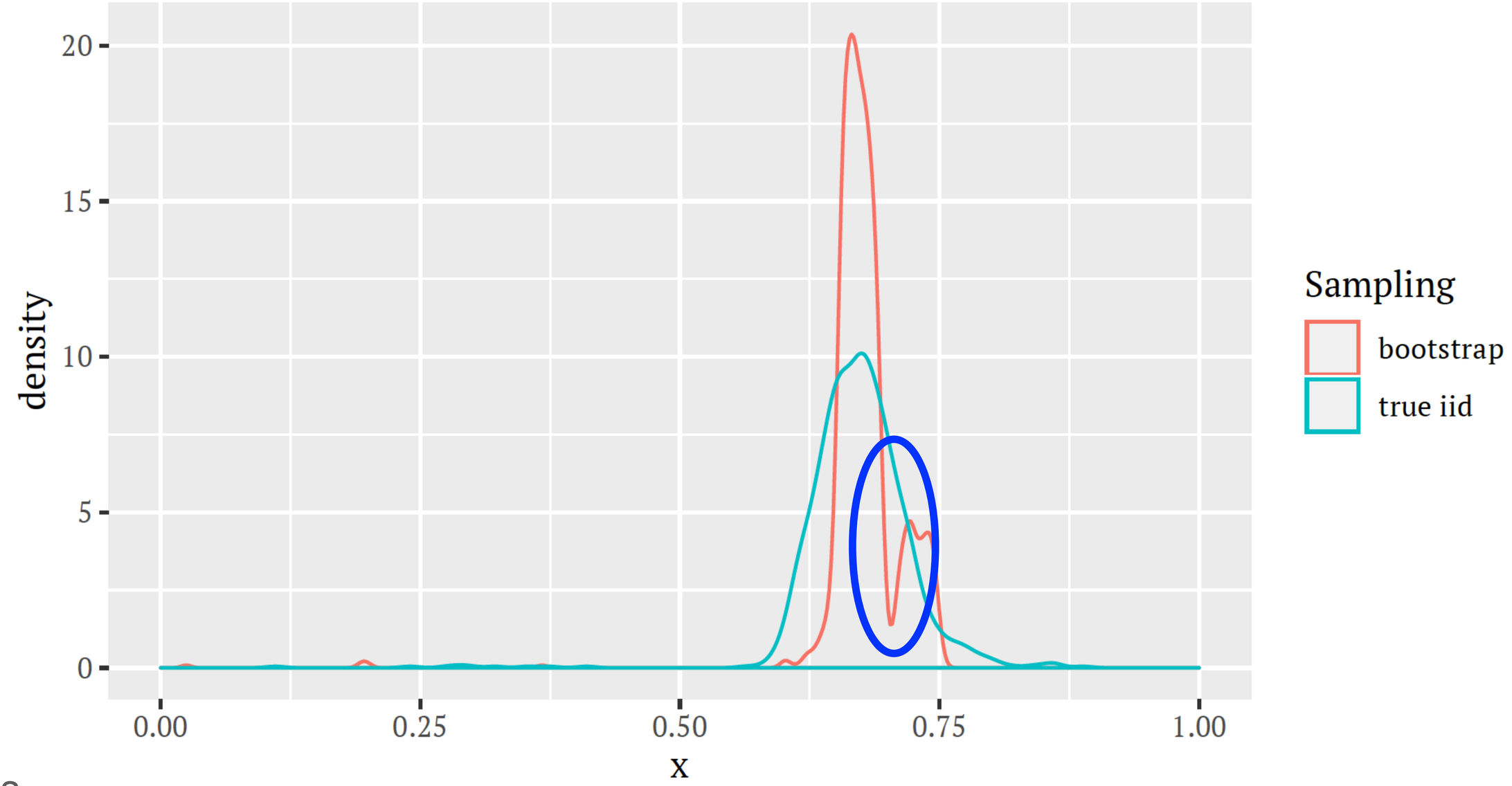


Sample from population

draw independent samples from the population—> true iid split location

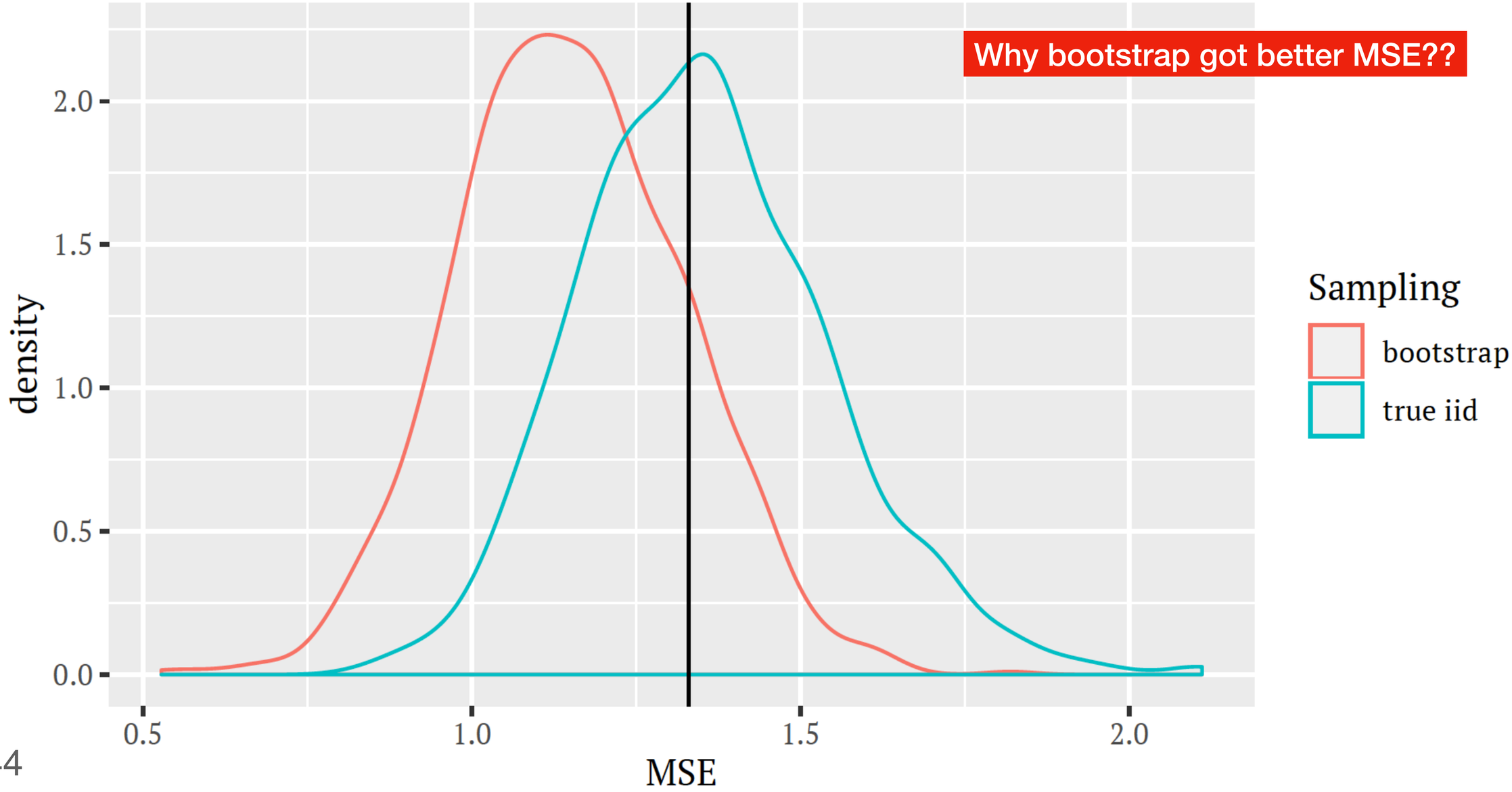


Bootstrap and Trees — split location densities



Some discretization, can't filling this when no data are there.

Bootstrap and Trees — MSE densities



Bagging (Bootstrap aggregating) for CART

For $b = 1,\ldots,B$, draw n bootstrap samples $(y_i^\star, \mathbf{x}_i^\star)_{i=1}^n$ and each time fit a tree and get $\hat{f}^{*b}(\mathbf{x})$, the prediction at a point $\mathbf{x}$.

- For regression trees, average all the predictions to obtain:

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

- For classification trees, we choose the class label for which the most bootstrapped trees "voted".

Probabilities from Bagging classification trees

Care is required when constructing a probabilistic prediction. Is this valid?

$$\hat{p}_j^{\text{bag}}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} \mathbb{1}\left(\arg\max_k \left(\hat{f}_k^b(\mathbf{x})\right) = j\right) \qquad \text{for } j^th \text{ class}$$

No! This is proportion of trees voting for given class.

What happened if $p_j(\mathbf{x}) = 0.8$?

$\hat{p}_j^{bag}(\mathbf{x})$ will likely be 1!

Instead, we should directly average tree output probabilities

$$\hat{p}_j^{\text{bag}}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_j^b(\mathbf{x})$$

Bagging (Bootstrap aggregating) for CART

Address overfitting:

- Grow large trees with minimal (or no) pruning. Relies on bagging procedure directly to avoid overfit.
- Prune each tree as was described before.

Out of sample evaluation:

In fact, we don't need to do cross-validation for bagging, because we know on average $36.8\,\%$ of original data will NOT be in bootstrap sample so can be used as a test set.

–> "Out Of Bag" (OOB) error estimation ($\sim$ 3-fold cross validation)

Bagging discussion

Pros

- If you have a lot of data, bagging makes sense because the empirical distribution will be close to the true population distribution.
- Bagging is not restricted to trees!
- Most useful when it is desirable to reduce the variance of a predictor.

  *under the (inaccurate) independence assumption, variance will reduce by a factor of $\sim 1/B$ for B bootstrap resamples.

  CLT

- Out of sample evaluation without using cross-validation, since any given observation will not be used in around 36.8% of the models.

Bagging discussion

Cons

- We lose interpretability as the final estimate is not a tree.
- Computational cost is multiplied by a factor of at least B.
- Bagging trees are correlated, the more correlated random variables are, the less the variance reduction of their average.

Random Forests

Bagging improved on a single CART model by reducing the variance through resampling. But, in fact the bagged models are still quite correlated. And the more correlated random variables are, the less the variance reduction of their average.

Q: Can we make them more independent in order to produce a better variance reduction?

A: Random forecasts achieve this by not only resampling observations, but by restricting the model to random subspaces, $\mathcal{X}' \subset \mathcal{X}$.

Random forests = Bagging + Random Subspaces

Methodology

Random Forests algorithm

i) Take a bootstrap resample $(y_i^\star, \mathbf{x}_i^\star)_{i=1}^n$ of the training data as for bagging.

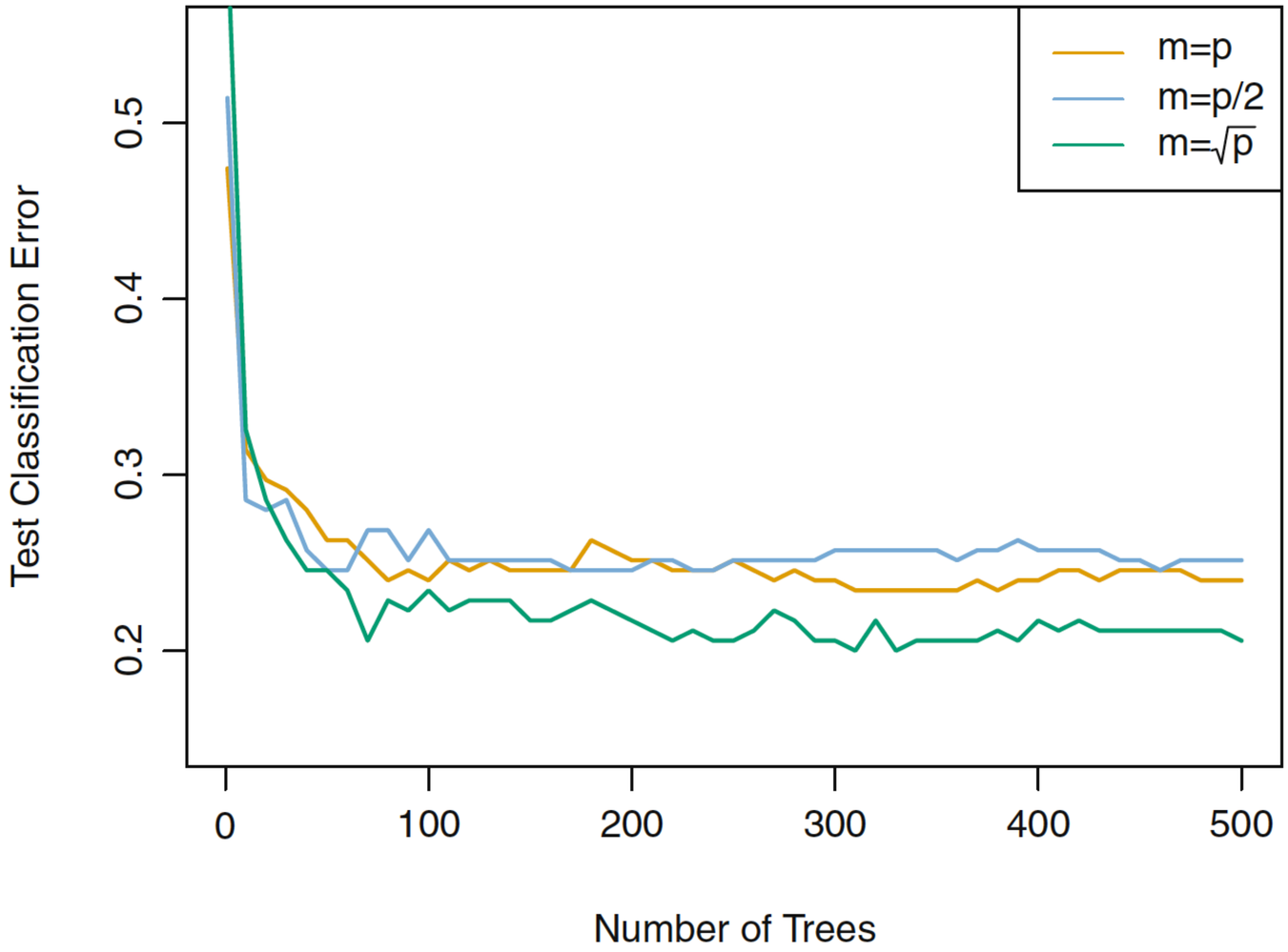
ii) Building a tree, each time a split in a tree is considered, random select m predictors out of the full set of p predictors as split candidates, and find the "optimal" split within those m predictors. (typically choose $m \approx \sqrt{p}$)

iii) Repeat i) and ii), average the prediction of all trees

The random subspaces causes less correlation in the trees. And combined with bagging this means the trees are a lot less correlated and variance is reduced substantially.

Methodology

Random Forests algorithm

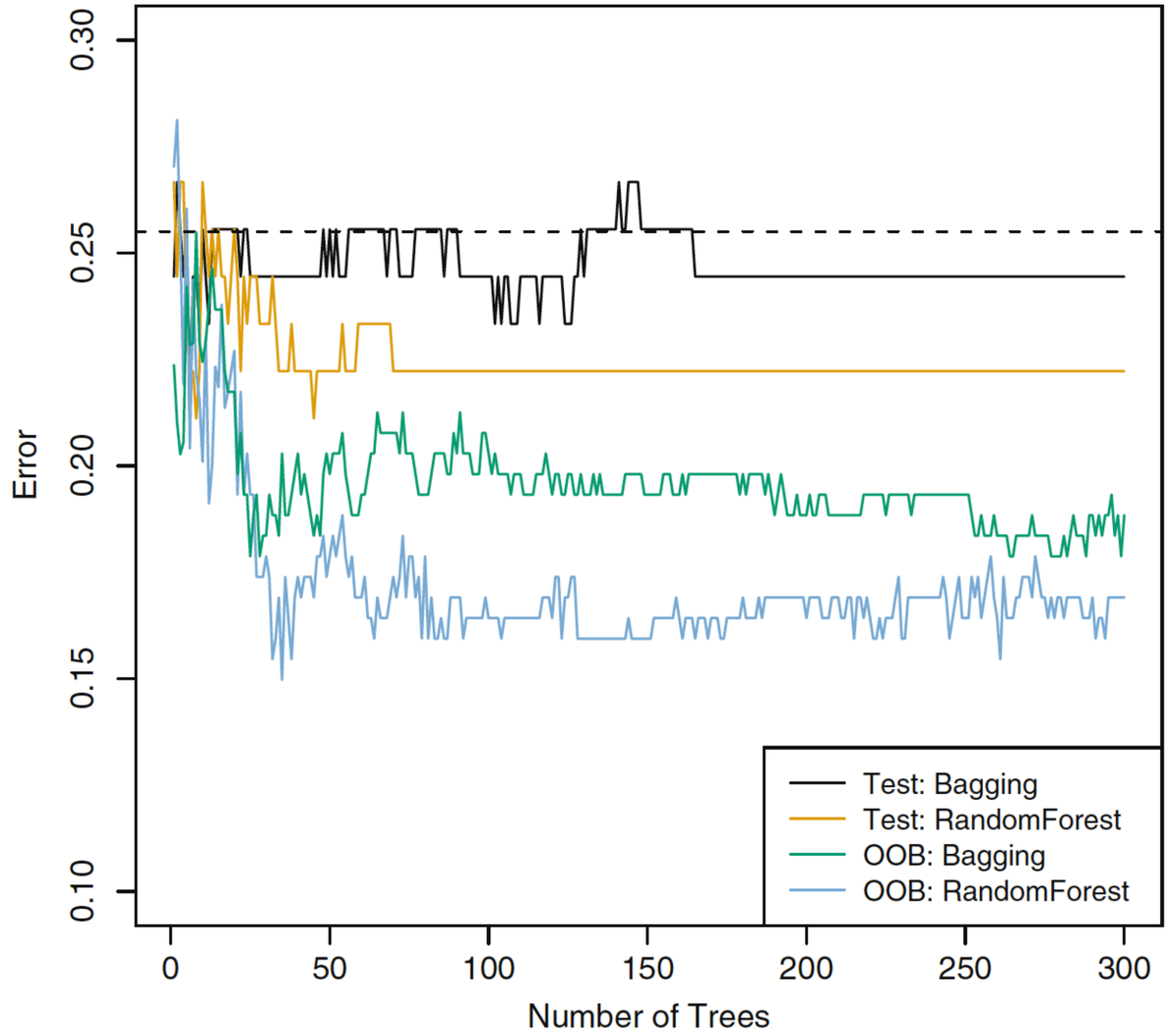


choosing $m \approx \sqrt{p}$ also reduces the computational cost in fitting the trees

Methodology

Random Forests algorithm

Reduce correlation –> reduced variance



Variable importance from random forests

Random Forecasts may seem great, but we've sacrificed interpretability with bagging and random subspace.

- Trees are easy to interpret and follow common human decision making mechanisms.
- Linear models provides statistical significance tests (e.g. t-test, z-test of parameter significance)

Q: Can we somehow quantify the importance of each variable?

Variable importance from random forests

Option 1:

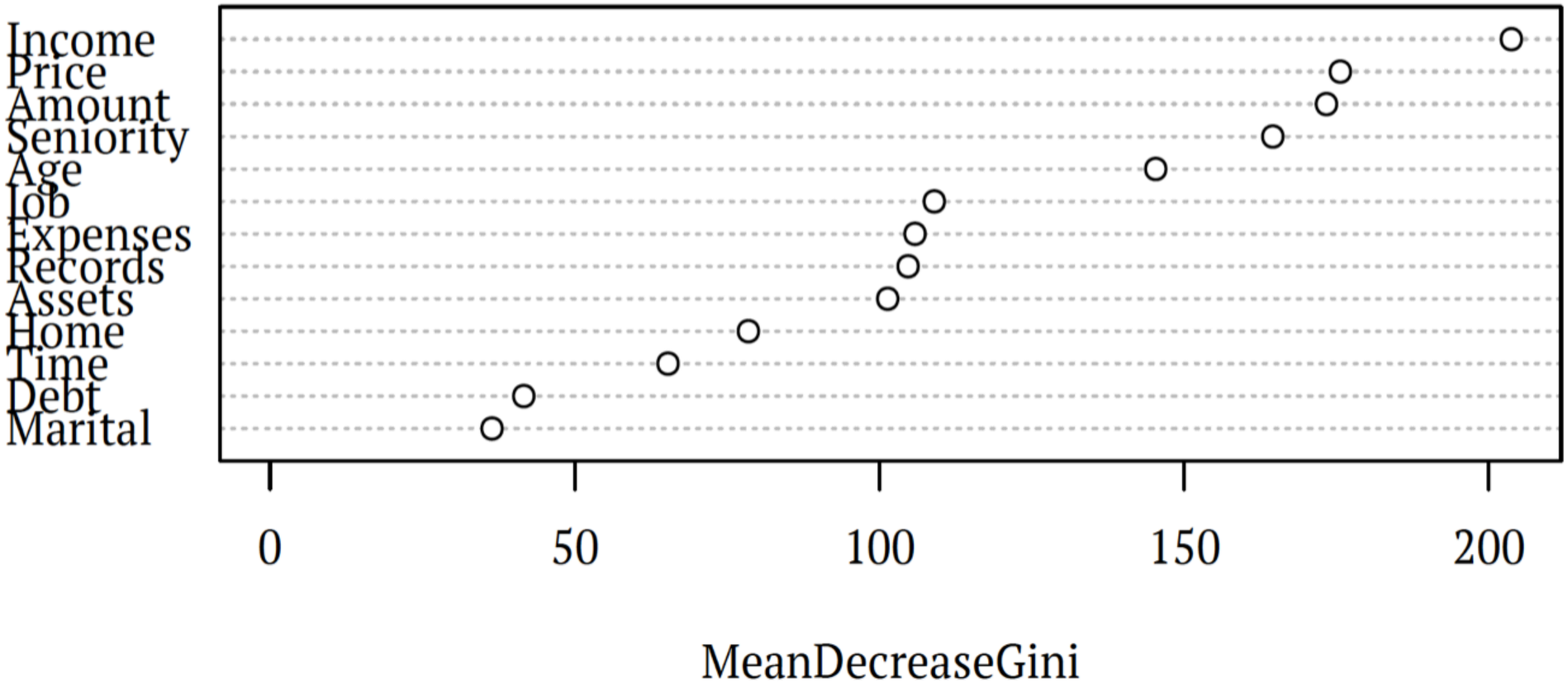
For each feature variable x_j, $j = 1,\ldots,p$, loop over each tree in the forest

- find all nodes that make a split on x_j
- compute the improvement in loss criterion the split causes (e.g. accuracy/Gini/etc)
- sum improvements across nodes in the tree

Finally, sum improvement across all tress.

Variable importance from random forests

Option 1:



Variable importance from random forests

Option 2:

Already discussed that bagging enables out of bag error estimate.

To compute out of bag variable importance, for each feature $x_j, j = 1,\ldots,p$

- for each tree, take the out of bag samples data matrix, $\mathbf{x}^{oob}$, and compute the predictive accuracy for that tree;
- now, take $\mathbf{x}^{oob}$ and randomly permute all the entries j^{th} column (break the ties between x_j and the rest of variables);
- pass the modified $\mathbf{x}^{oob*}$ through the tree and compute the change in predictive accuracy for that tree.

Finally average the decrease in accuracy over all trees.

Random Forests discussion

Pros

- Inherits the advantages of bagging and trees
- Very easy to parallelise
- Works well with high dimensional data
- Tuning is rarely needed (easy to get good quality forecast)

Cons

- Often quite suboptimal for regression.
- Same extrapolation issue as trees.
- Harder to implement and memory hungry model.

Methodology

Boosting

Boosting is similar to bagging in the sense that we combine results from multiple classifiers, but it is fundamentally different in approach…

1) Set $\hat{f}(\mathbf{x}) = 0$ and $\epsilon_i = y_i$ for $i = 1, \ldots, n$.

2) For $b = 1, \ldots, B$, iterate:

i) Fit a model (eg tree) $\hat{f}^b(\mathbf{x})$ to the response $\epsilon_1, \ldots, \epsilon_n$

ii) Update the predictive model to

$$\hat{f}(\mathbf{x}) \leftarrow \hat{f}(\mathbf{x}) + \lambda \hat{f}^b(\mathbf{x})$$

iii) Update the residuals

$$\epsilon_i \leftarrow \epsilon_i - \lambda \hat{f}^b(\mathbf{x}) \quad \forall i$$

3) Output as final boosted model

$$\hat{f}(\mathbf{x}) = \sum_{b=1}^{B} \lambda \hat{f}^b(\mathbf{x})$$

Intuition

Slow learning: Aim to learn usually simple model (→low variance, but high bias) in each round. Then bring bias down by repeating over many rounds.

Sequential learning: Each round of boosting aims to correct the error of the previous rounds.

Generic methodology: Any model at all can be boosted! Completely general method.

Methodology

Boosting

Tuning parameters for boosting:

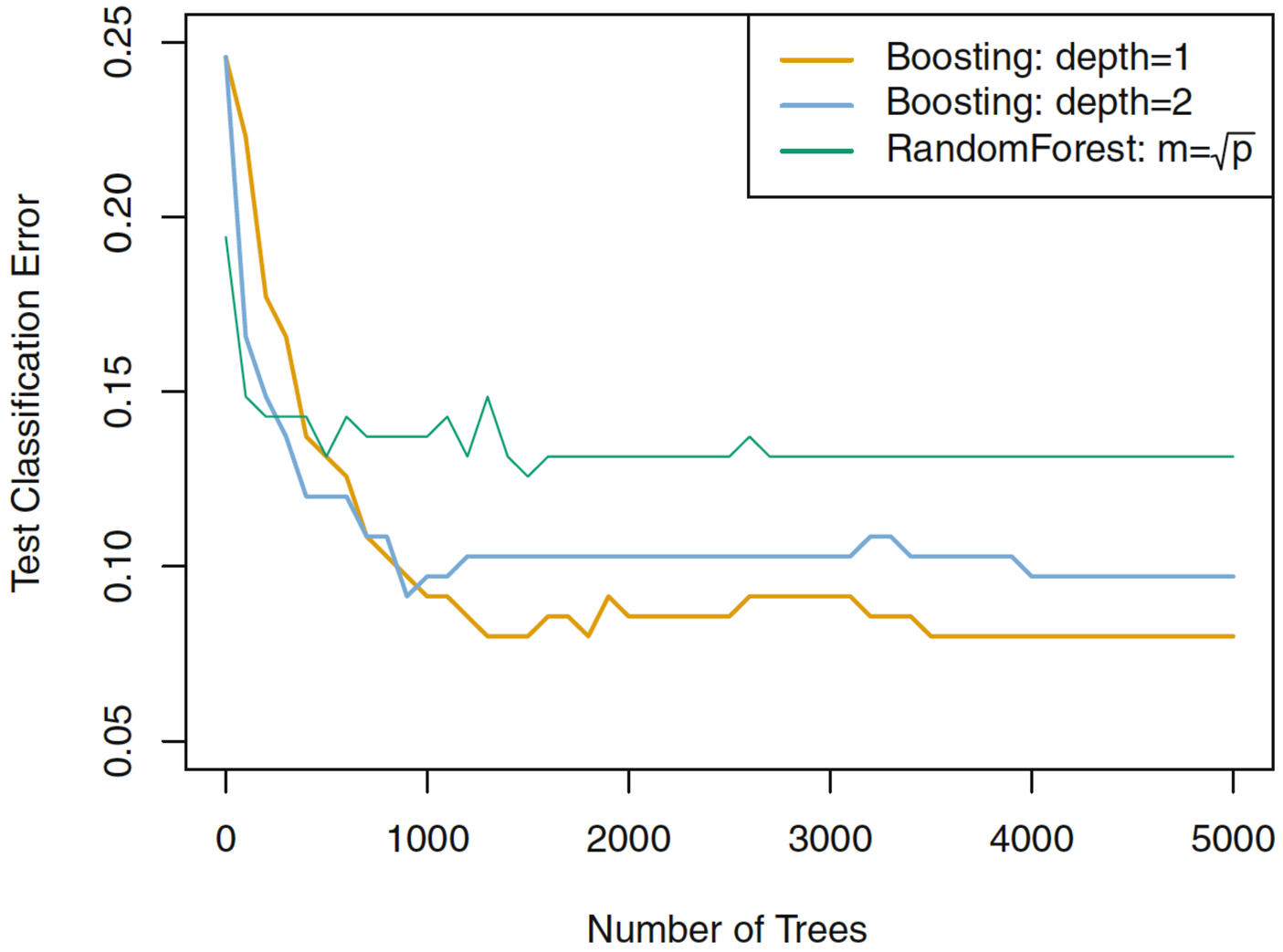
1) The number of trees B. Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all. We use cross-validation to select B.

2) The shrinkage parameter λ, 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 λ can require using a very large value of B in order to achieve good performance.

3) The number of splits d in each tree, which controls the complexity of the boosted ensemble. Often $d = 1$ works well, in which case each tree consists of a single split and resulting in an additive model. More generally d is the interaction depth, and controls the interaction order of the boosted model, since d splits can involve at most d variables.

Methodology

Boosting



Our old friend cross-validation helps choose λ and B.

Boosting discussion

Pros

- Boosting is an incredibly powerful technique and regularly is instrumental in winning machine learning competitions.
- Any model at all can be boosted.

Cons

- Tuning is needed (not trivial)
- Difficulties in interpretation and extrapolation.