Ensembles 1: Trees to Forests

Lecture 10

Connor Dowd

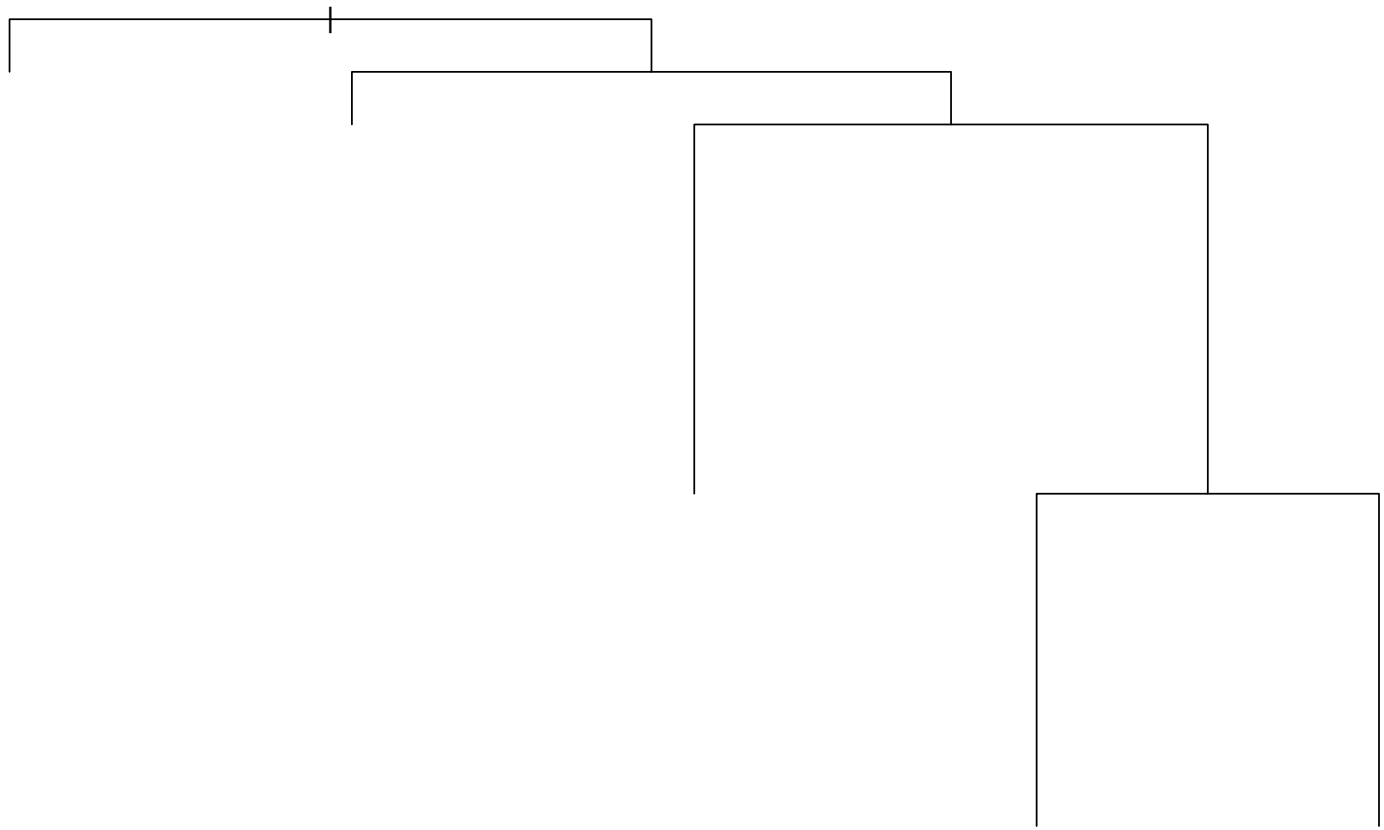
April 29th, 2021

Today's Class

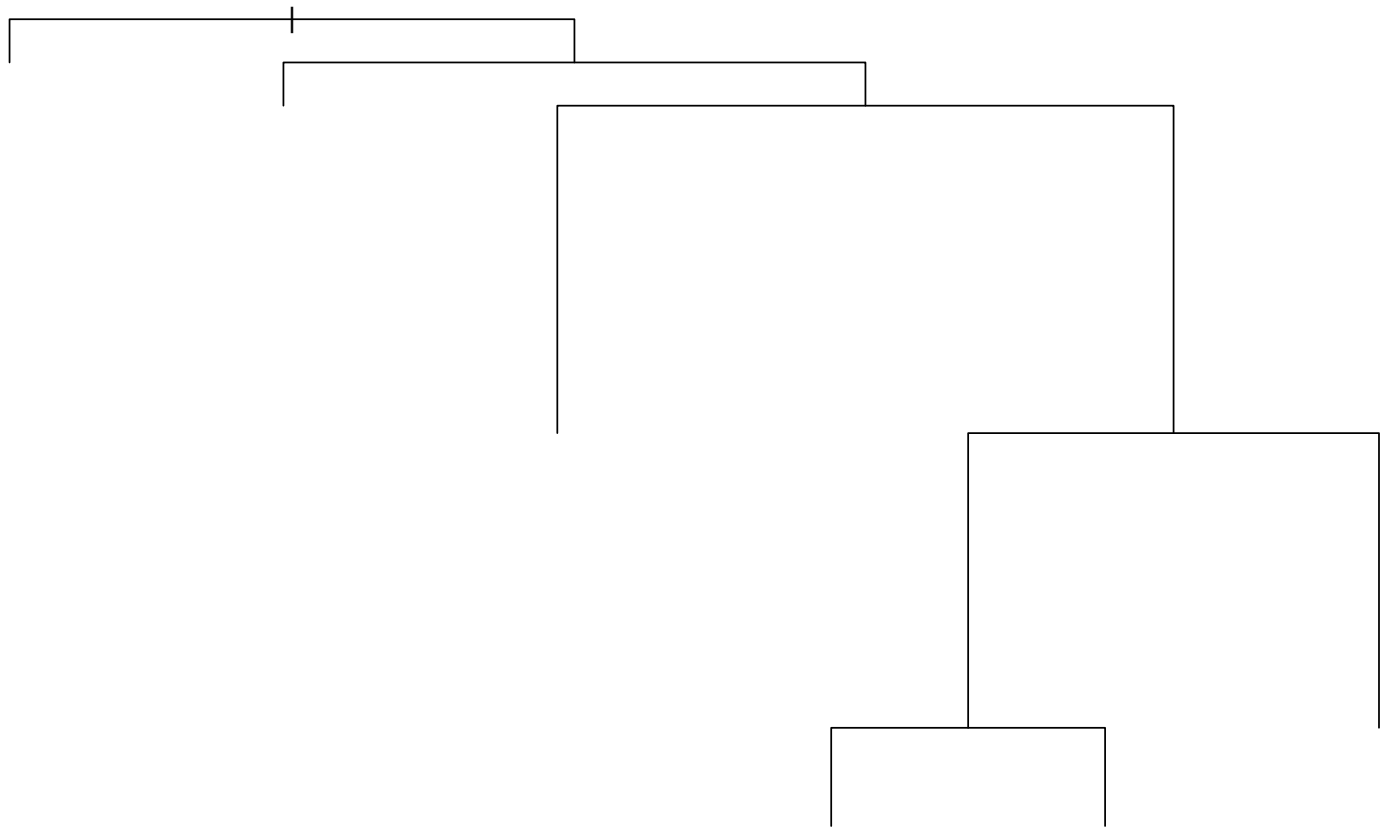
1. Trees: a few more details
2. Bagging: Many Trees
3. Random Forests
4. Boosting??

Trees Continued

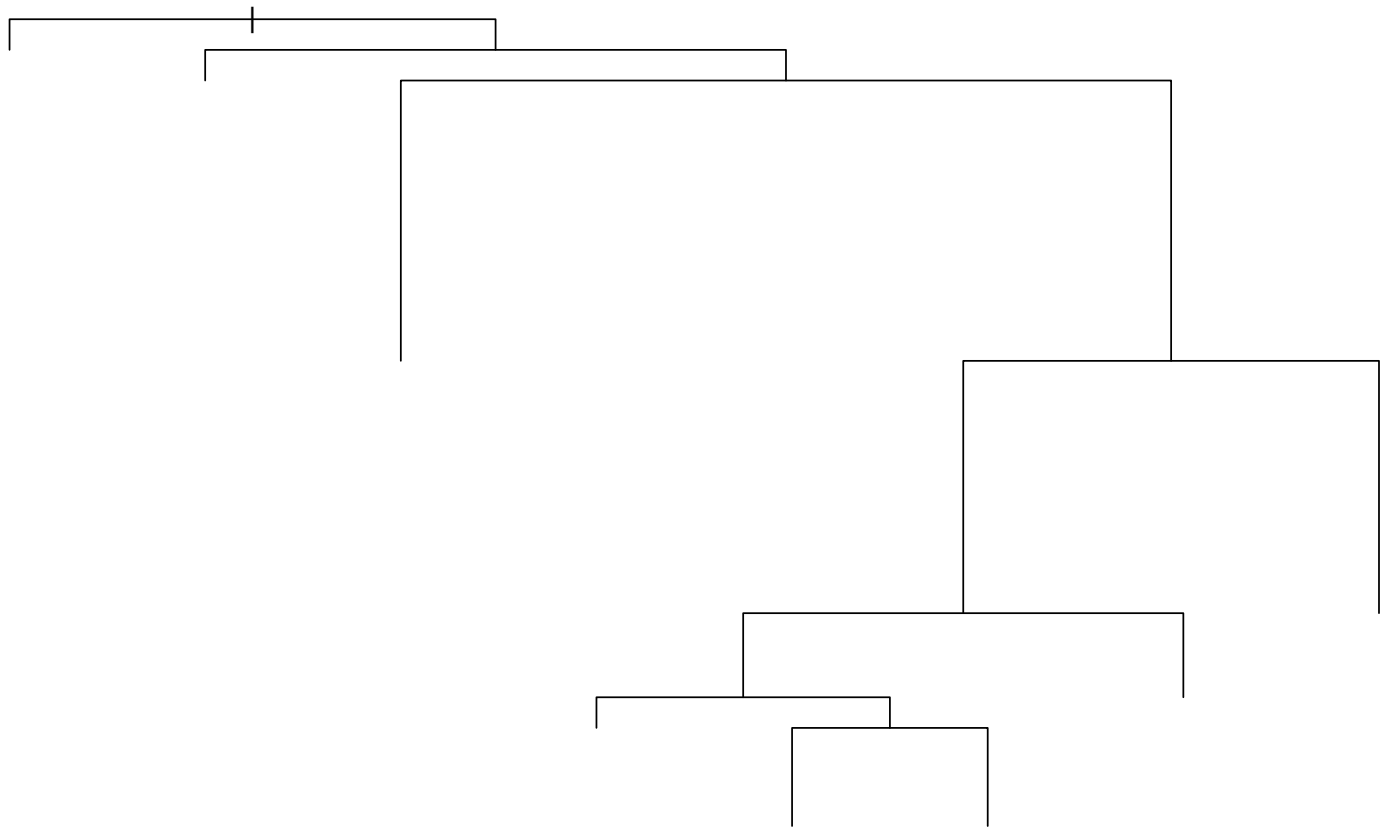
Trees



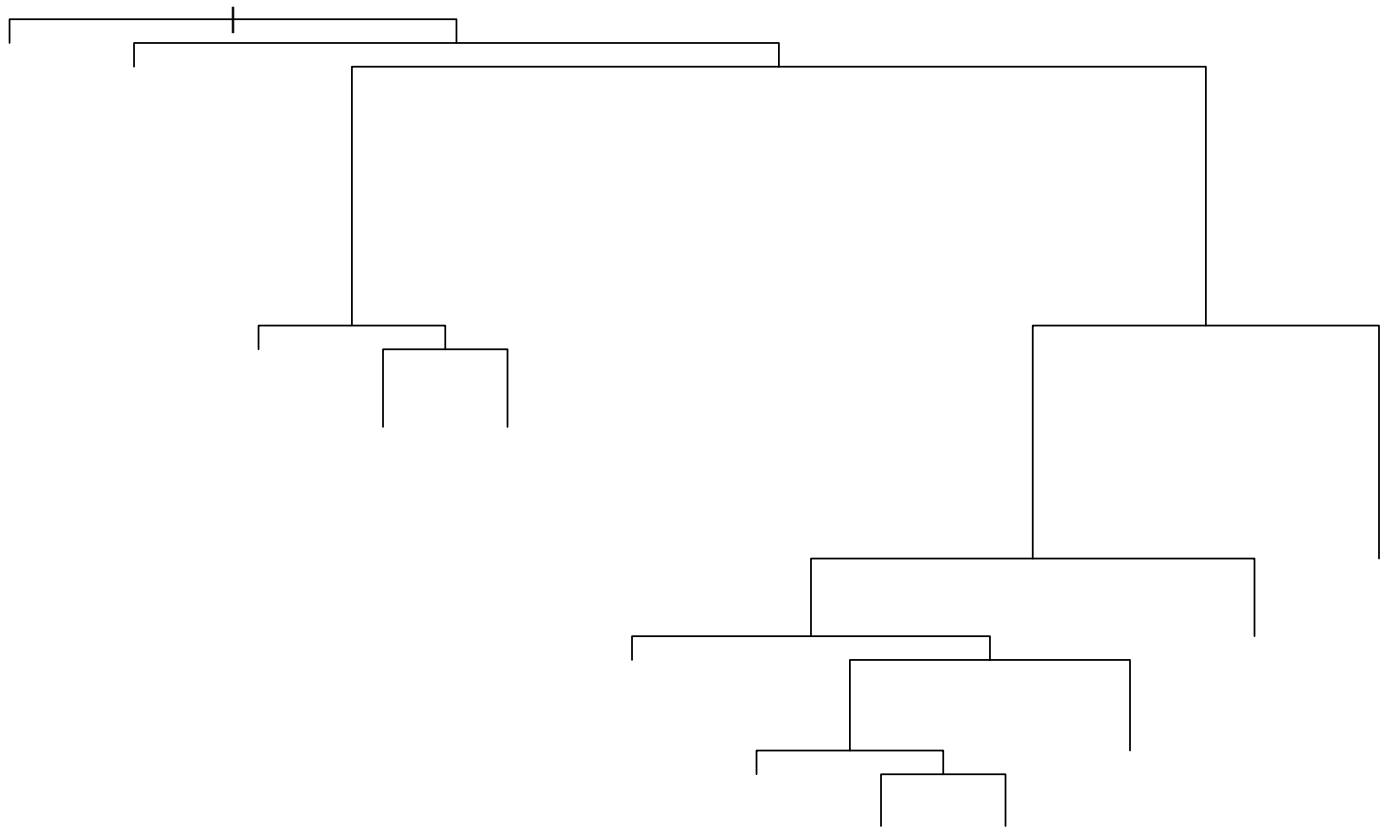
Trees



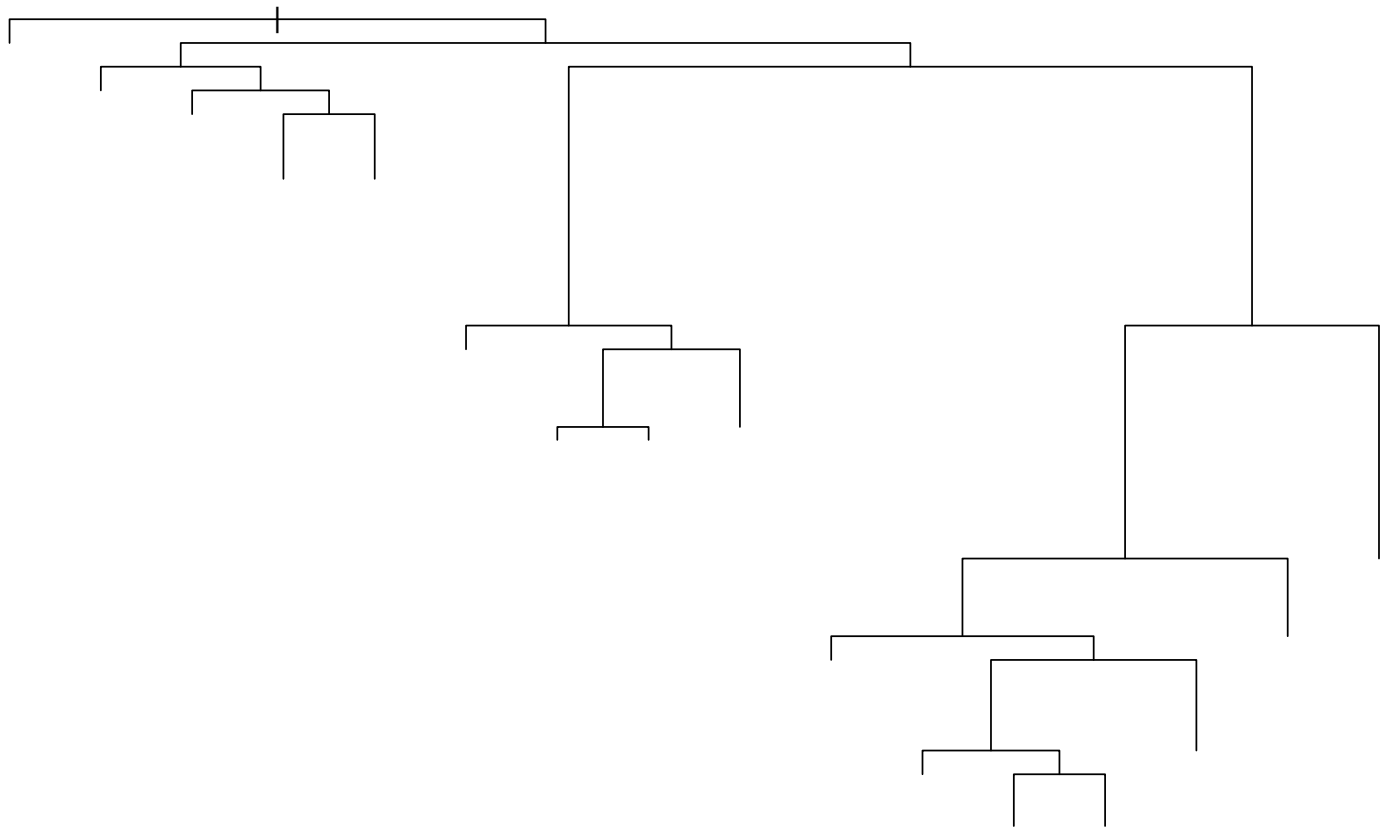
Trees



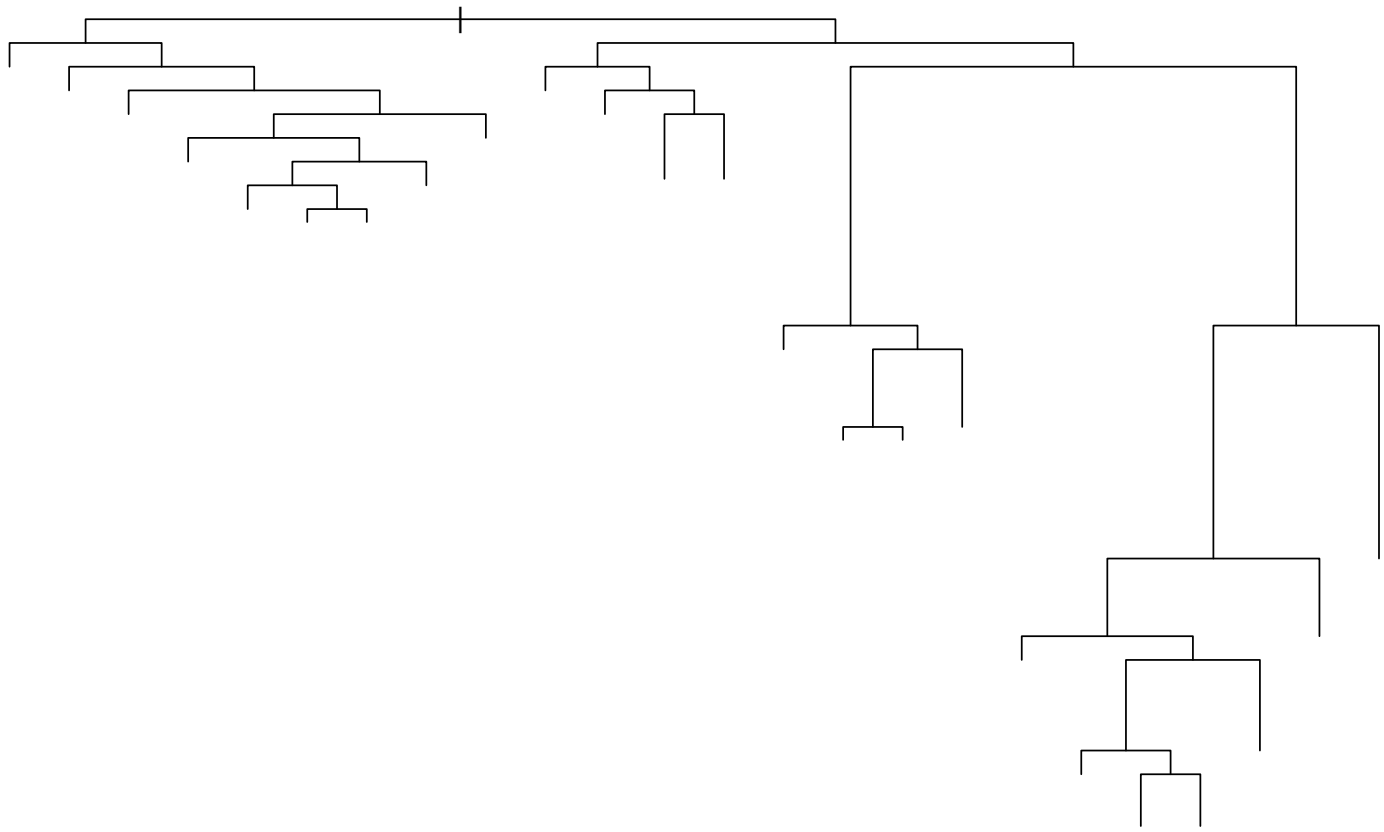
Trees



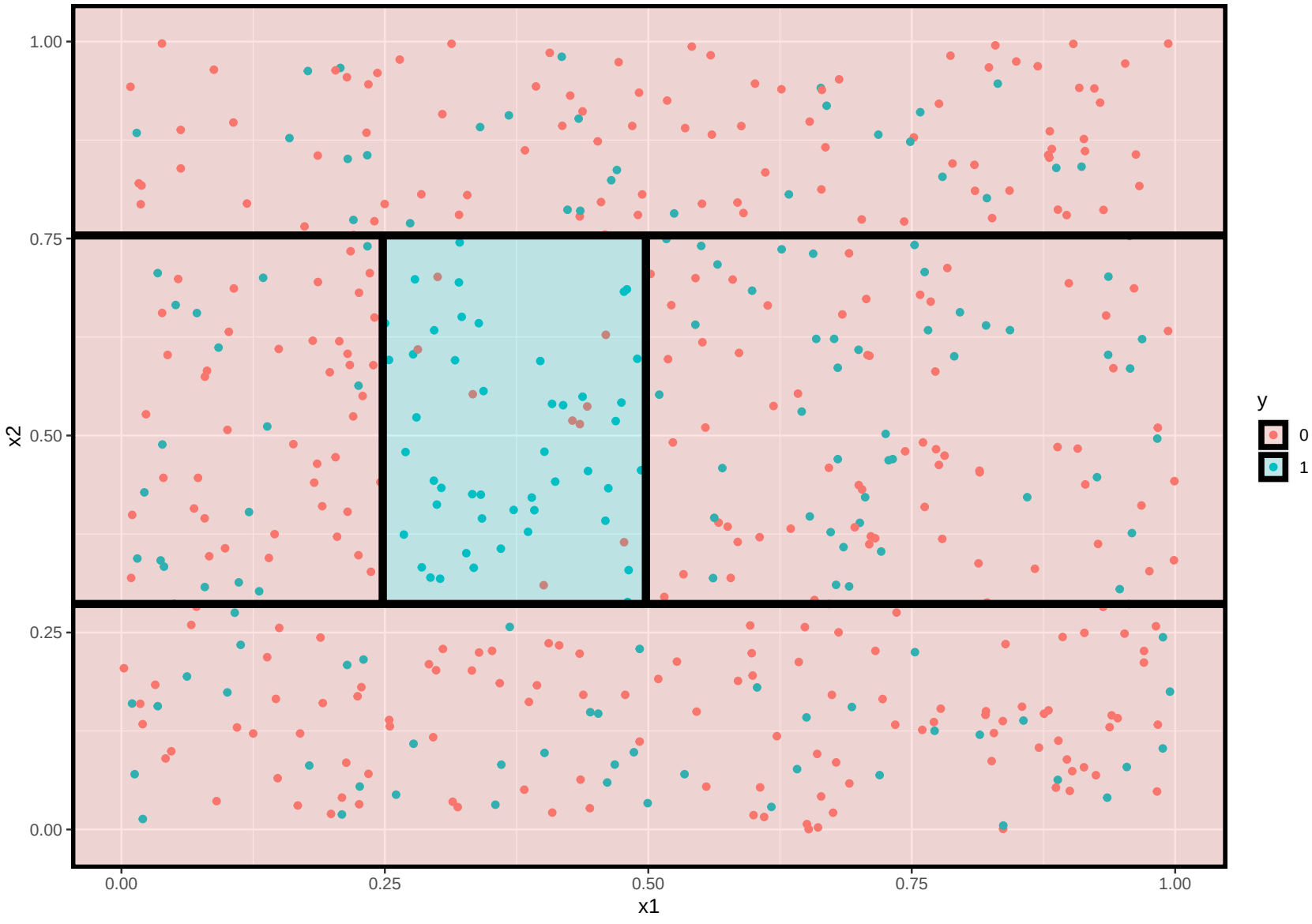
Trees



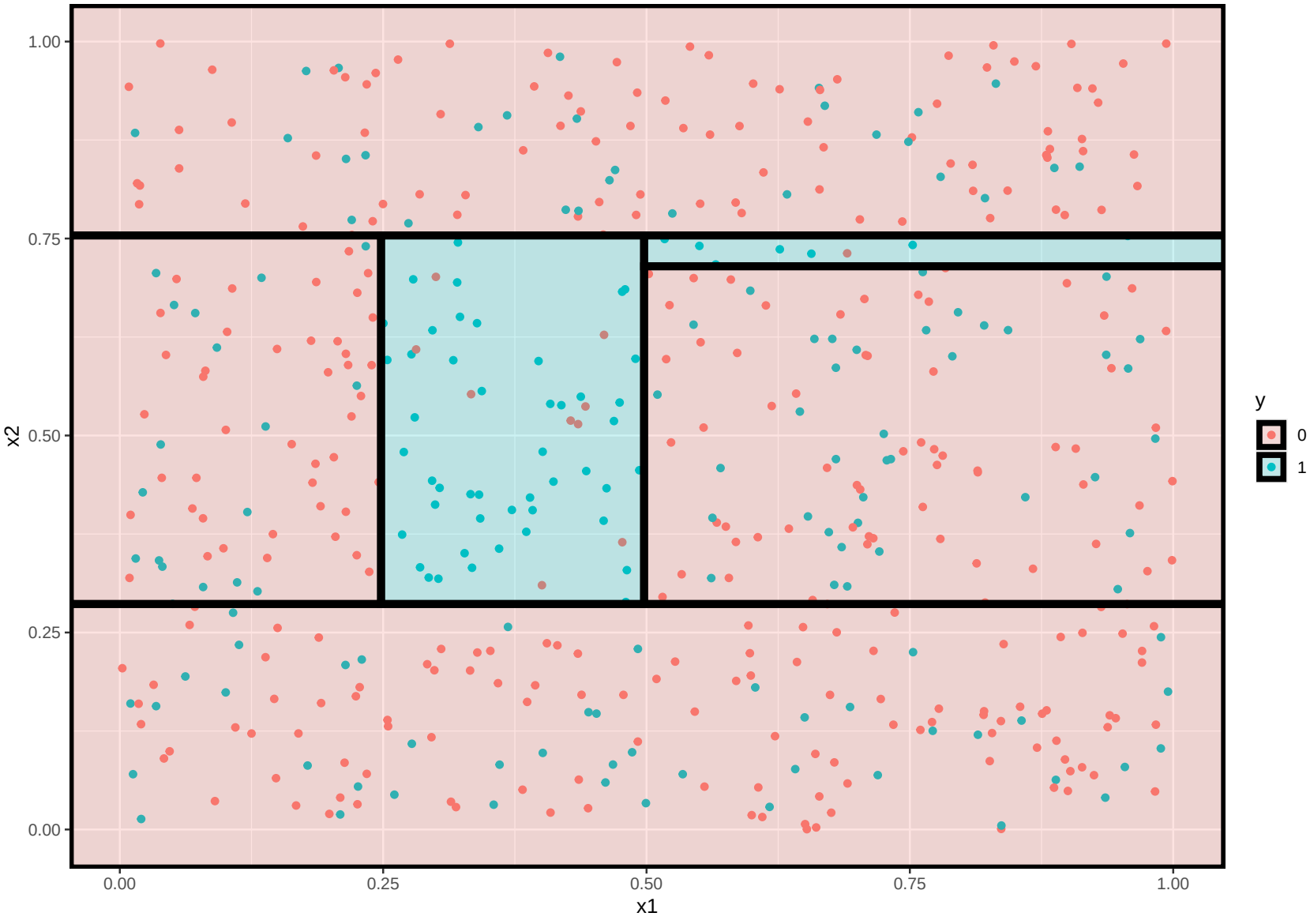
Trees



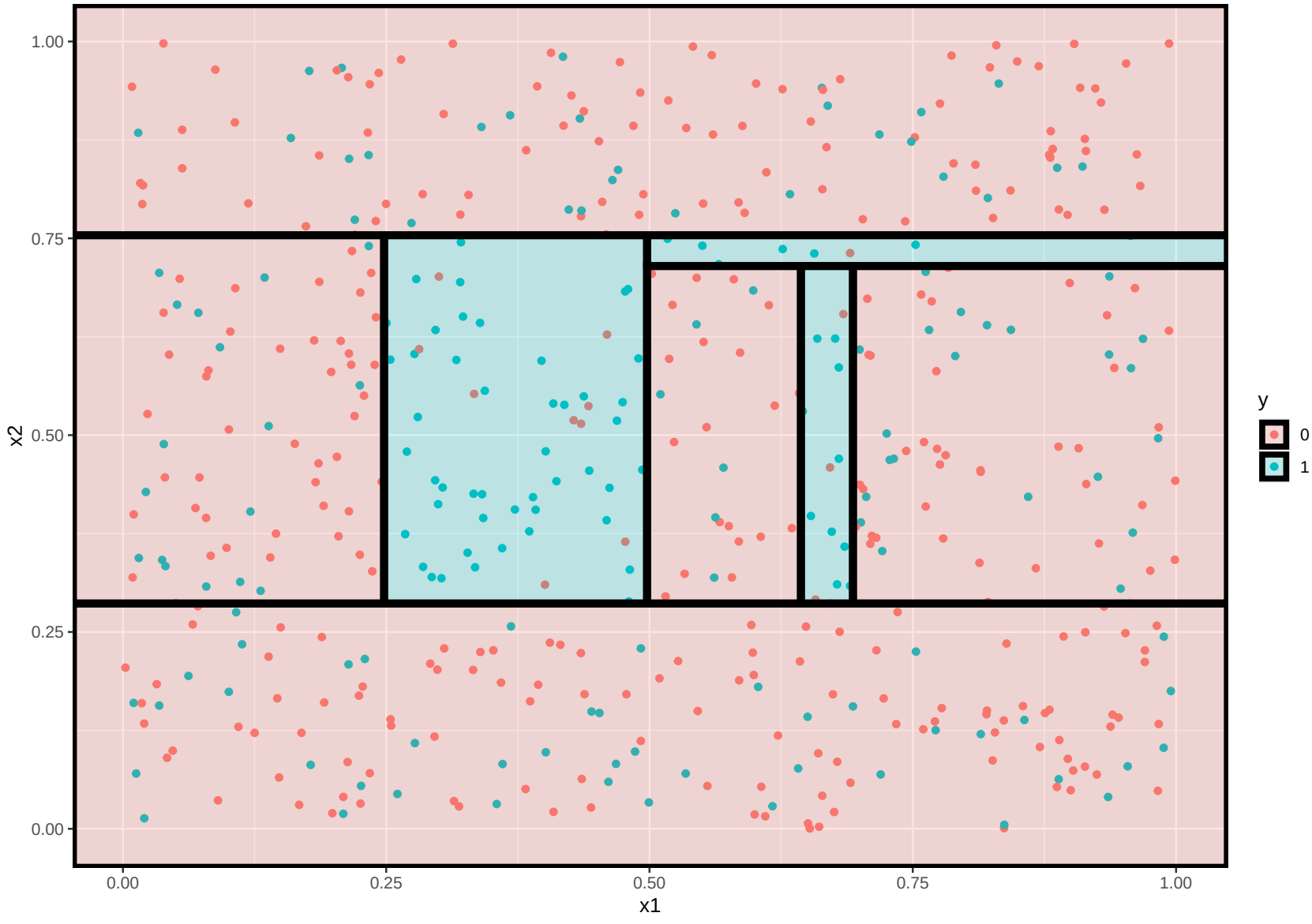
Trees



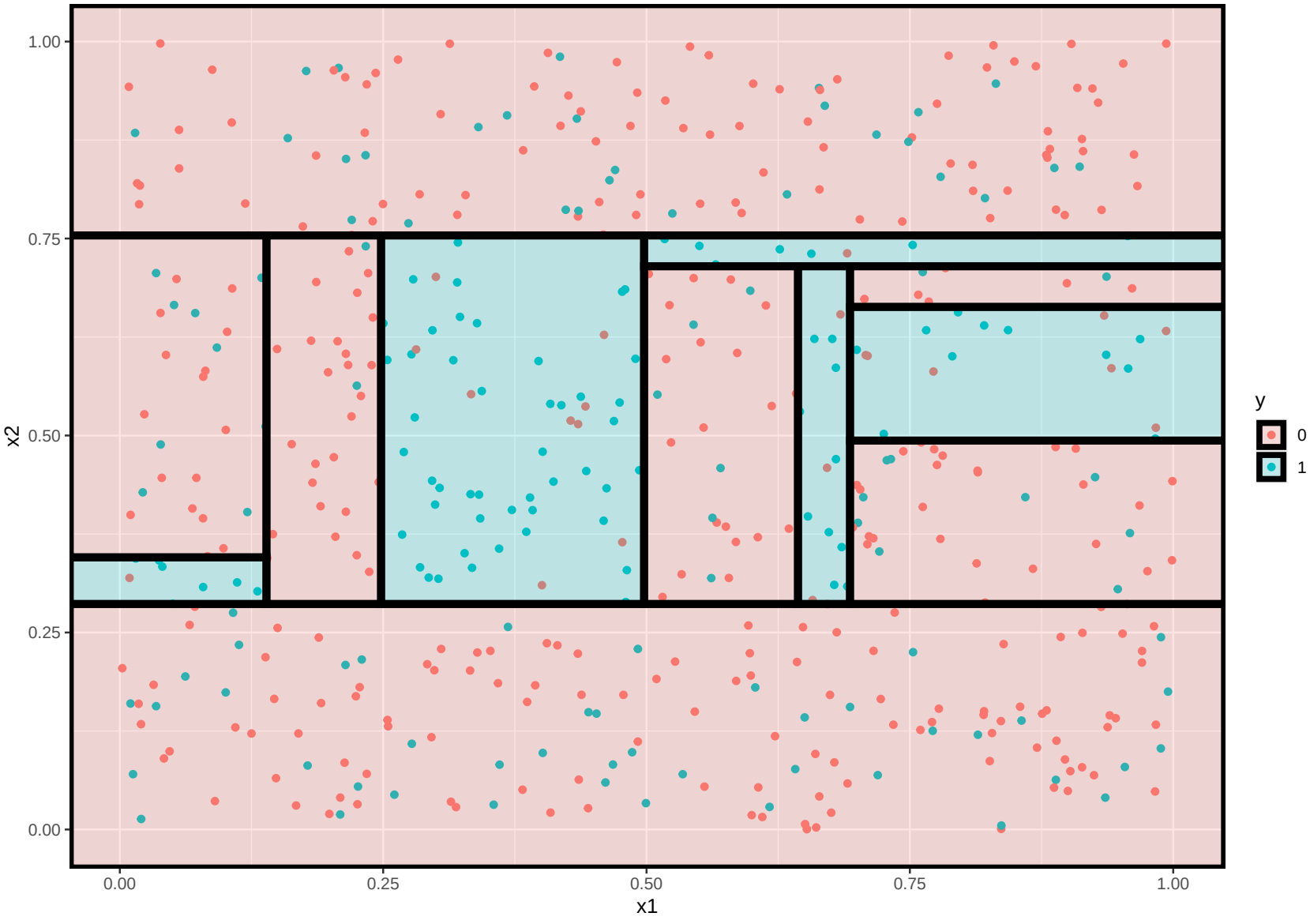
Trees



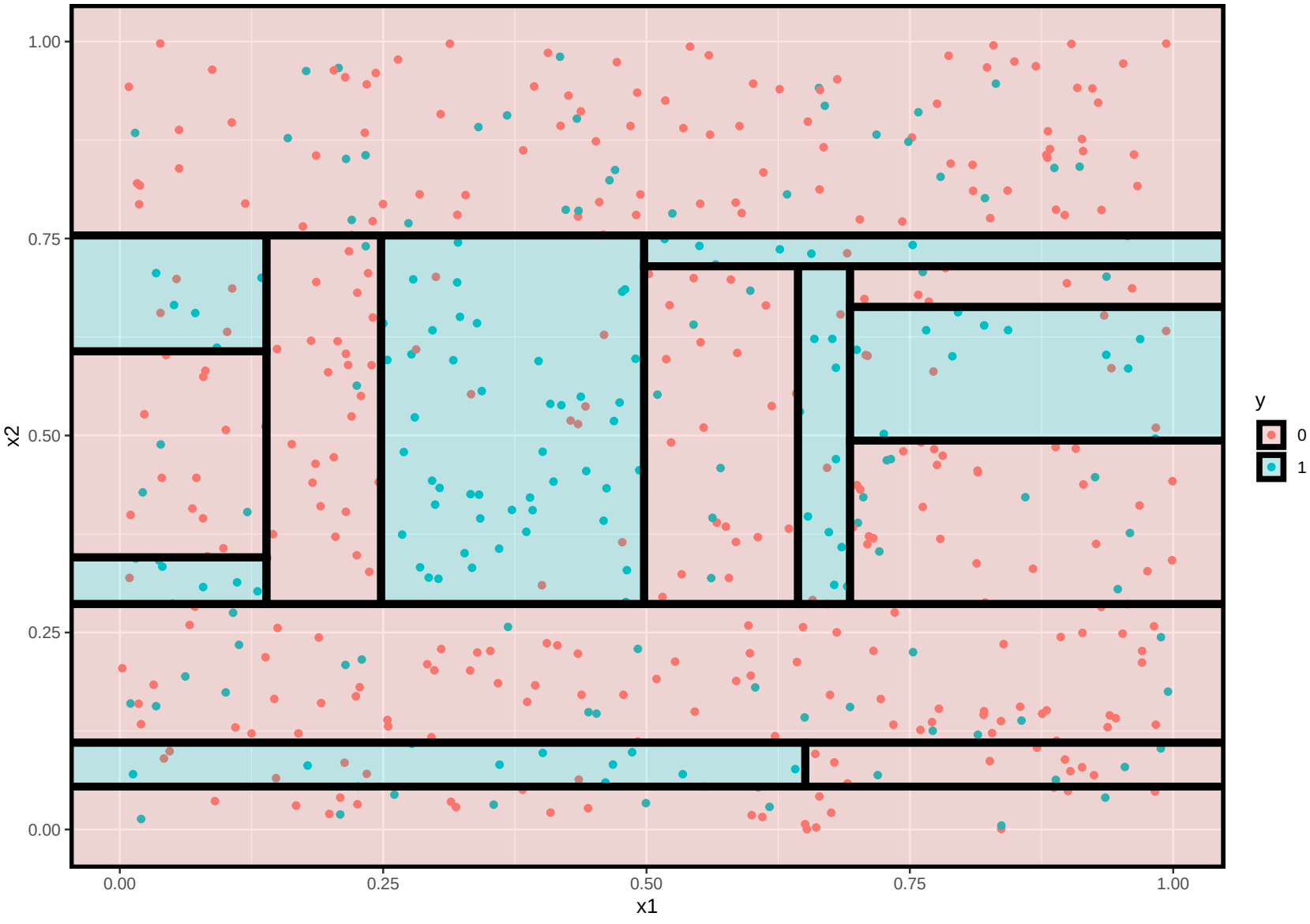
Trees



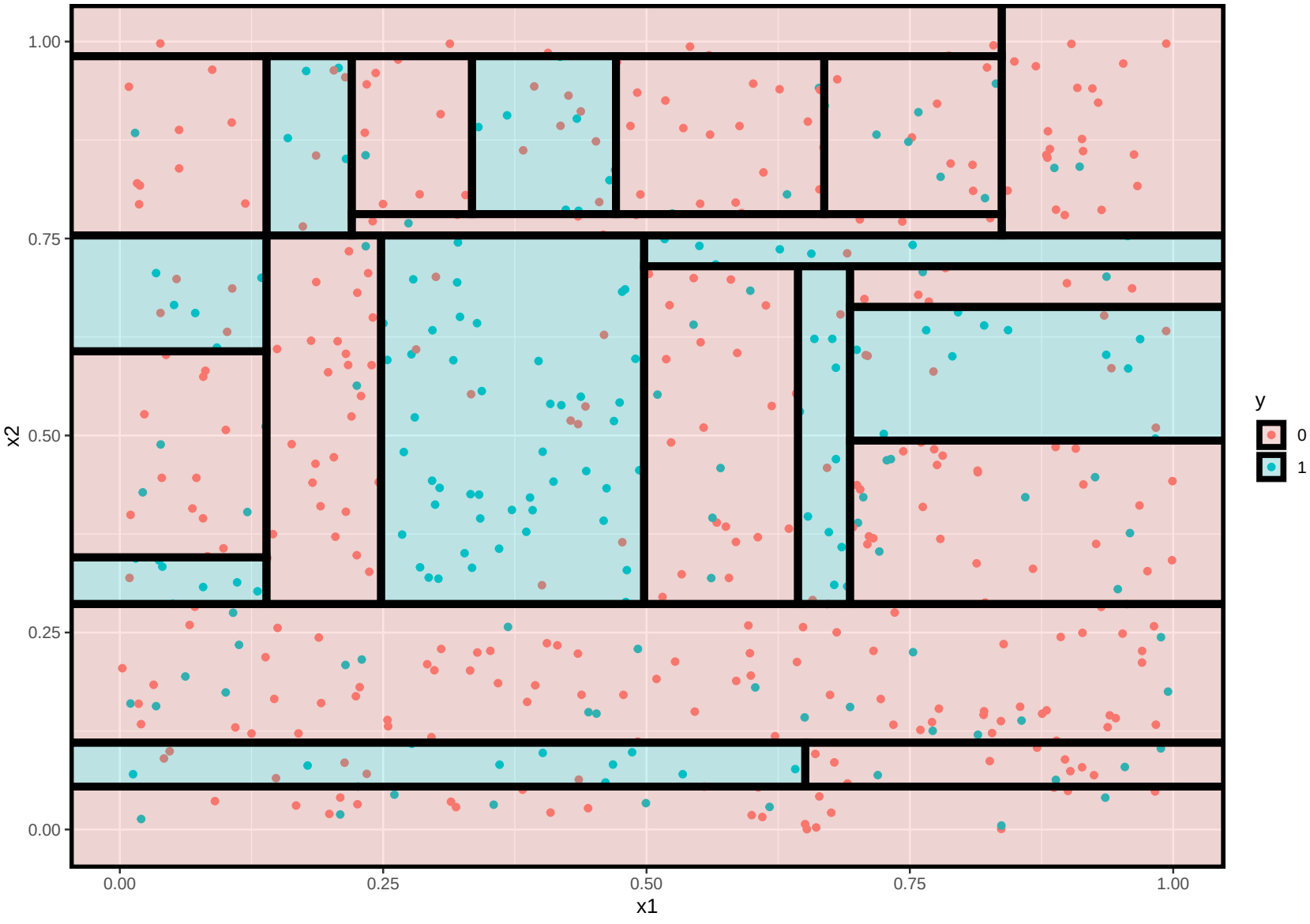
Trees



Trees



Trees



Gini Impurity

Split rule is usually based on “gini-impurity”, but other measures are possible and widespread.

The gini impurity is a measure of how likely we are to make a wrong prediction for some subset if we randomly choose a prediction based on the probabilities in a subset.

Gini Impurity

The gini-impurity within a subset with J classes:

$$I_G(p) = \sum_{i=1}^{J} p_i(1 - p_i) = 1 - \sum_{i=1}^{J} p_i^2$$

When there are two classes this is:

$$I_G(\mathbf{p}) = \sum_{i=1}^{2} p_i(1 - p_i) = p_1(1 - p_1) + p_2(1 - p_2) = 1 - p_1^2 - p_2^2$$

Which is minimized when $p_1 \in \{0, 1\}$.

Gini Impurity Choices

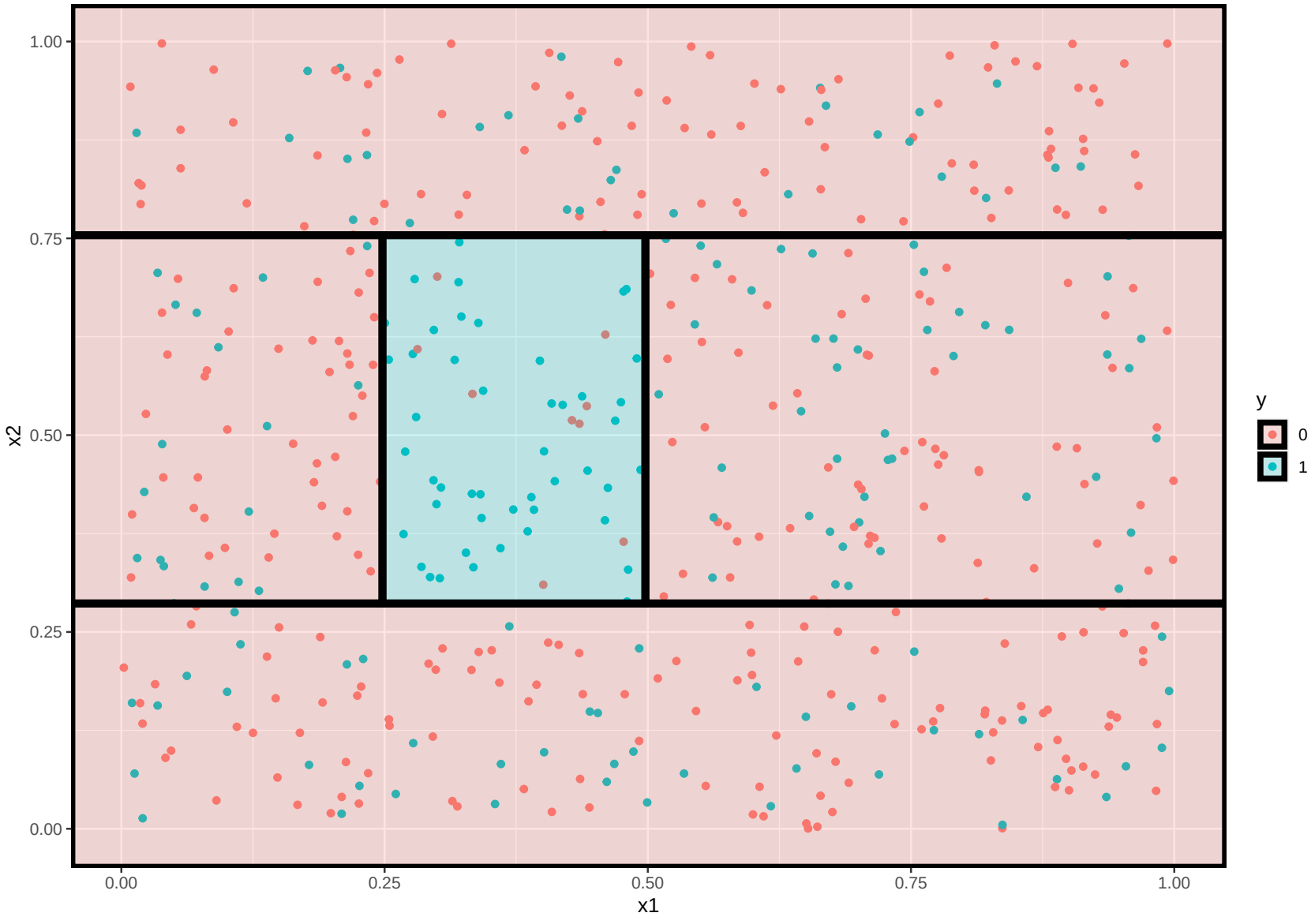
But we care about this error given a random selection of points.

$$\sum_{i \in L_j} I_G(\mathbf{p}_i)$$

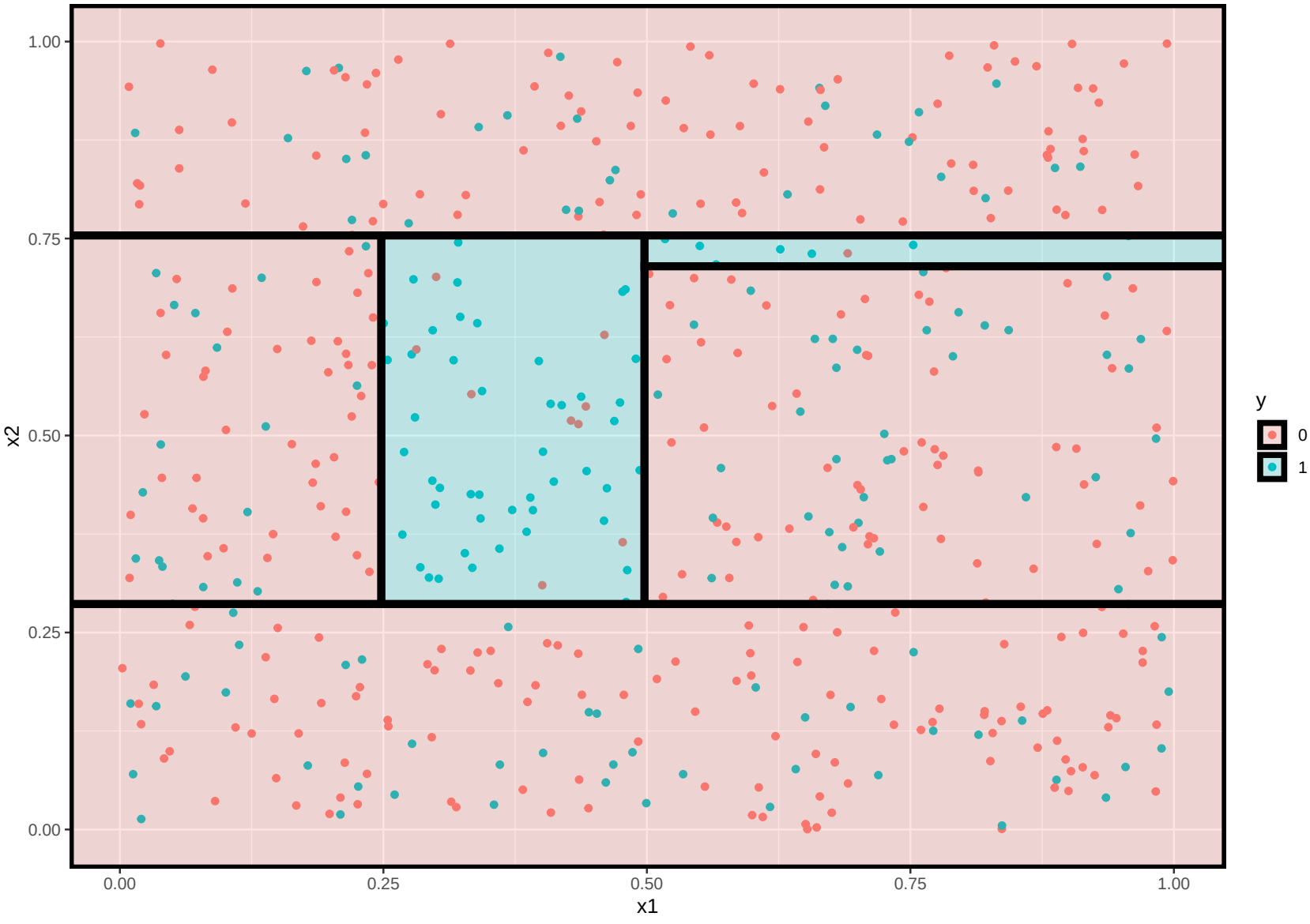
If some partition (and the predictions it implies) improves this measure the most, that is the partition we choose.

L_j is the set of members of leaf j.

Example



Example



Trees – An aside: Factors

Trees do not like "unordered factor variables".

This is because it does not know how to split them up. Despite usually having a small number of possible values (e.g. 50 diff states). The lack of ordering means there are a large number of possible splits. (2^50).

For this reason, to use trees with factor variables, you need to either:

1. Impose some ordering. (E.g. just rank the states by population, etc)
2. Create a dummy variable for each state (50 possible splits, not 2^50)

Trees – Aside 2: Continous Outcomes

We could also make trees for situations where our outcome is a continuous variable.

The prediction will be easier. Instead of picking the "plurality" class in a leaf, we predict the mean in the leaf.

The splits will be similar. Instead of minimizing the gini-impurity, we minimize the *in-node* MSE.

Trees – Aside 2: Continous Outcomes Continued

Where will this not work well?

Linear models. They have a constant slope in some dimension, which trees will have difficulty with. The trees will have to constantly split off new leaves in that dimension, with new means. And even then, the predictions won't be great.

- We will see a solution shortly.

Trees Problems

- Overfitting
  - How many leaves? We need to choose.
- Very unstable.
  - **Slight Peturbations** of the data can change the entire tree structure

One solution? Bootstrap-aggregating ("BAgging")

Bagging: In Brief

We will boostrap the data:

- Recall: this means drawing another similar size sample with replacement and building the model with that sample – repeatedly.

This in essence creates numerous small "peturbations" of the data.

And it will let us create 1000 very similar models. Each of which is potentially very overfit.

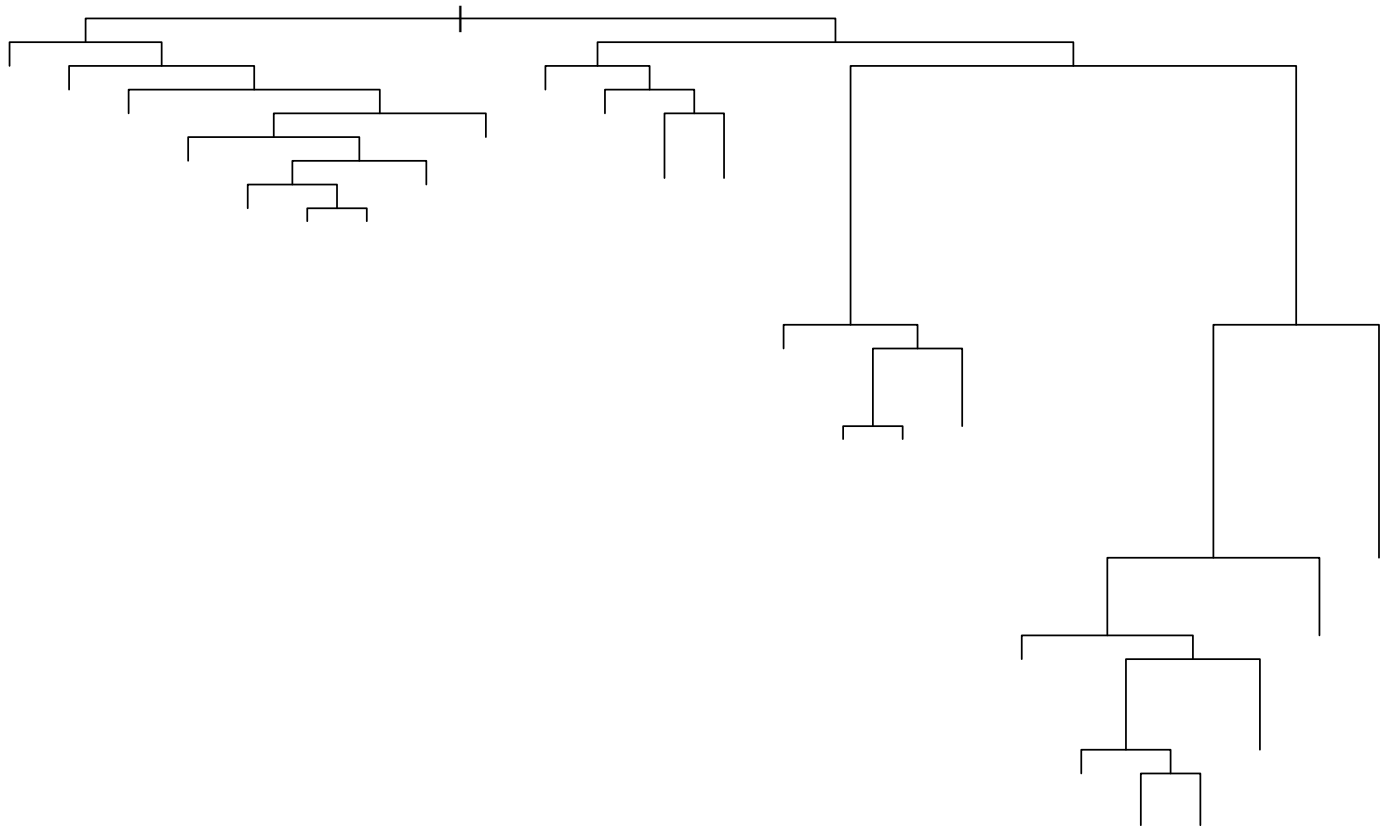
Example

```
resampled_mod = function(x) {
  ind = sample(nrow(df),,replace=T)
  rpart(y~x1+x2,data=df[ind,],cp=0)
}

mod =  rpart(y~x1+x2,data=df,cp=0)
modb1 = resampled_mod(1)
modb2 = resampled_mod(1)
modb3 = resampled_mod(1)
modb4 = resampled_mod(1)
modb5 = resampled_mod(1)
modb6 = resampled_mod(1)
modb7 = resampled_mod(1)
```

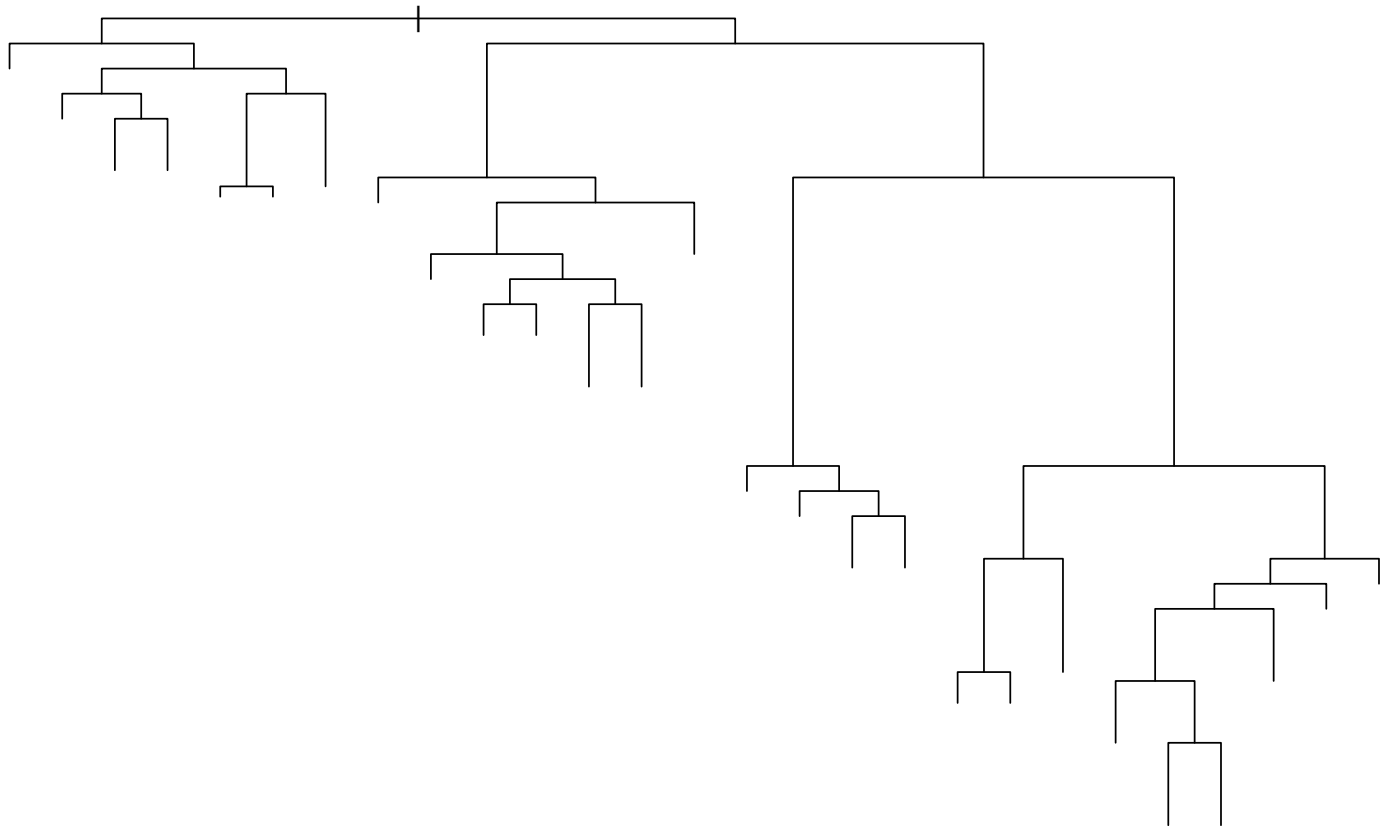
Example

```
plot(mod)
```



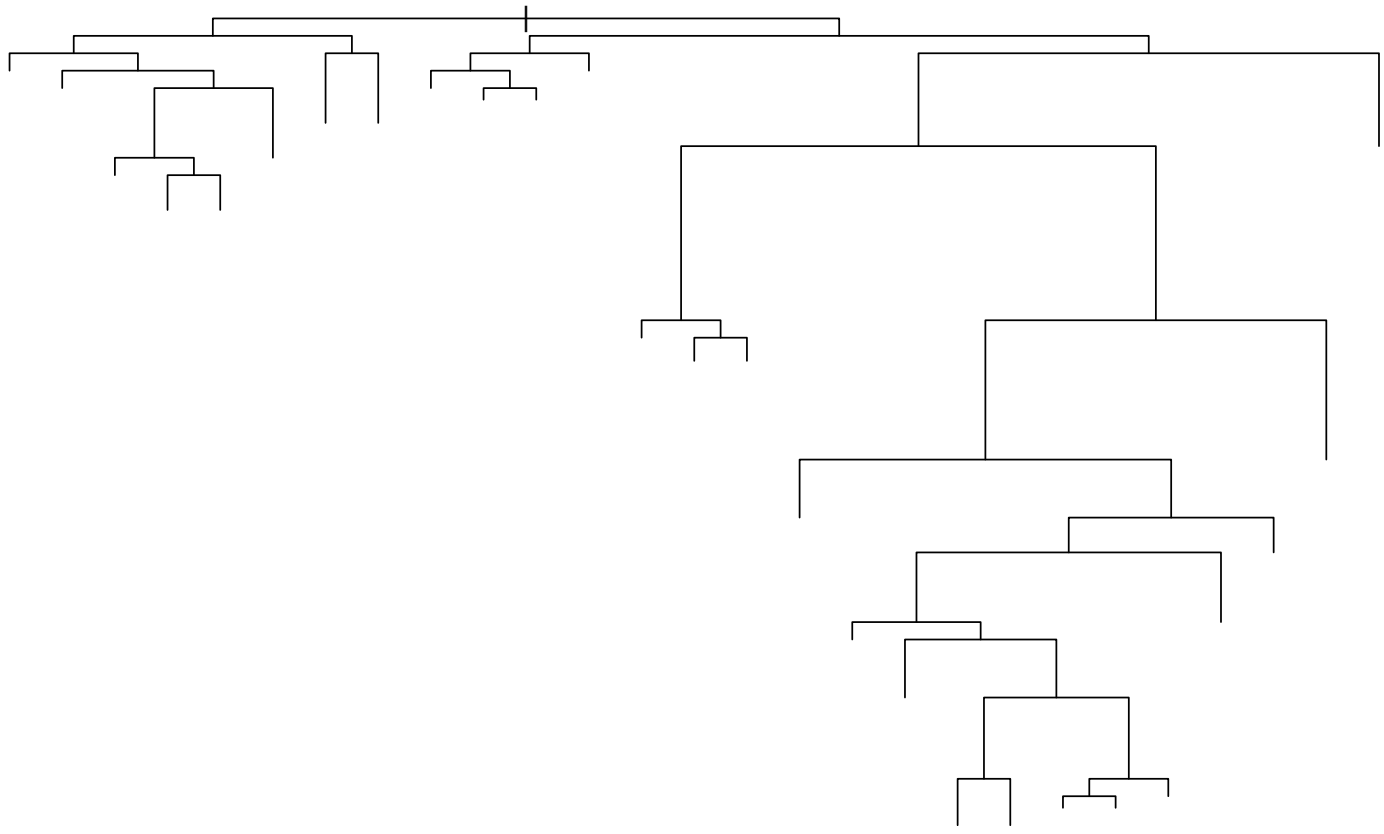
Example

```
plot(modb1)
```



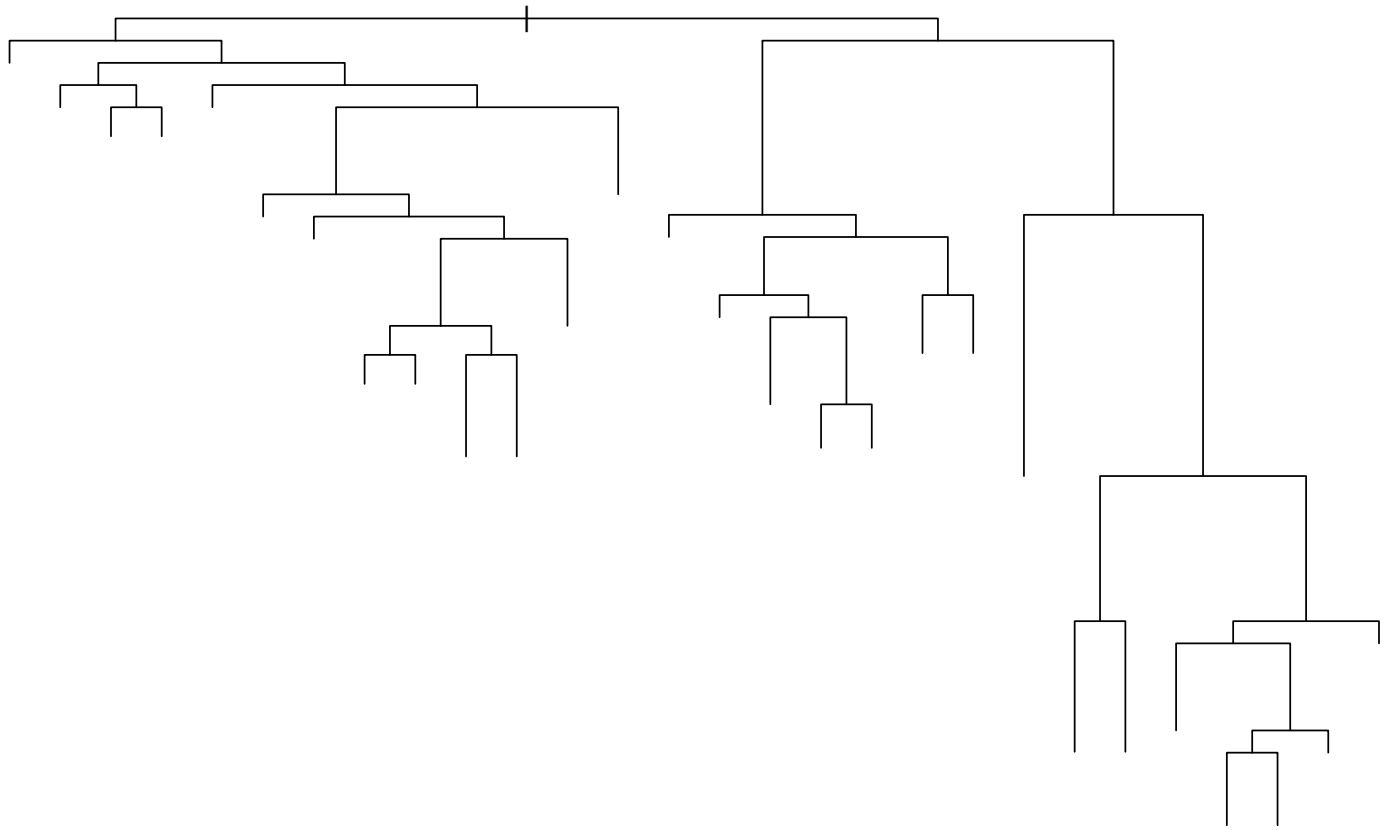
Example

```
plot(modb2)
```



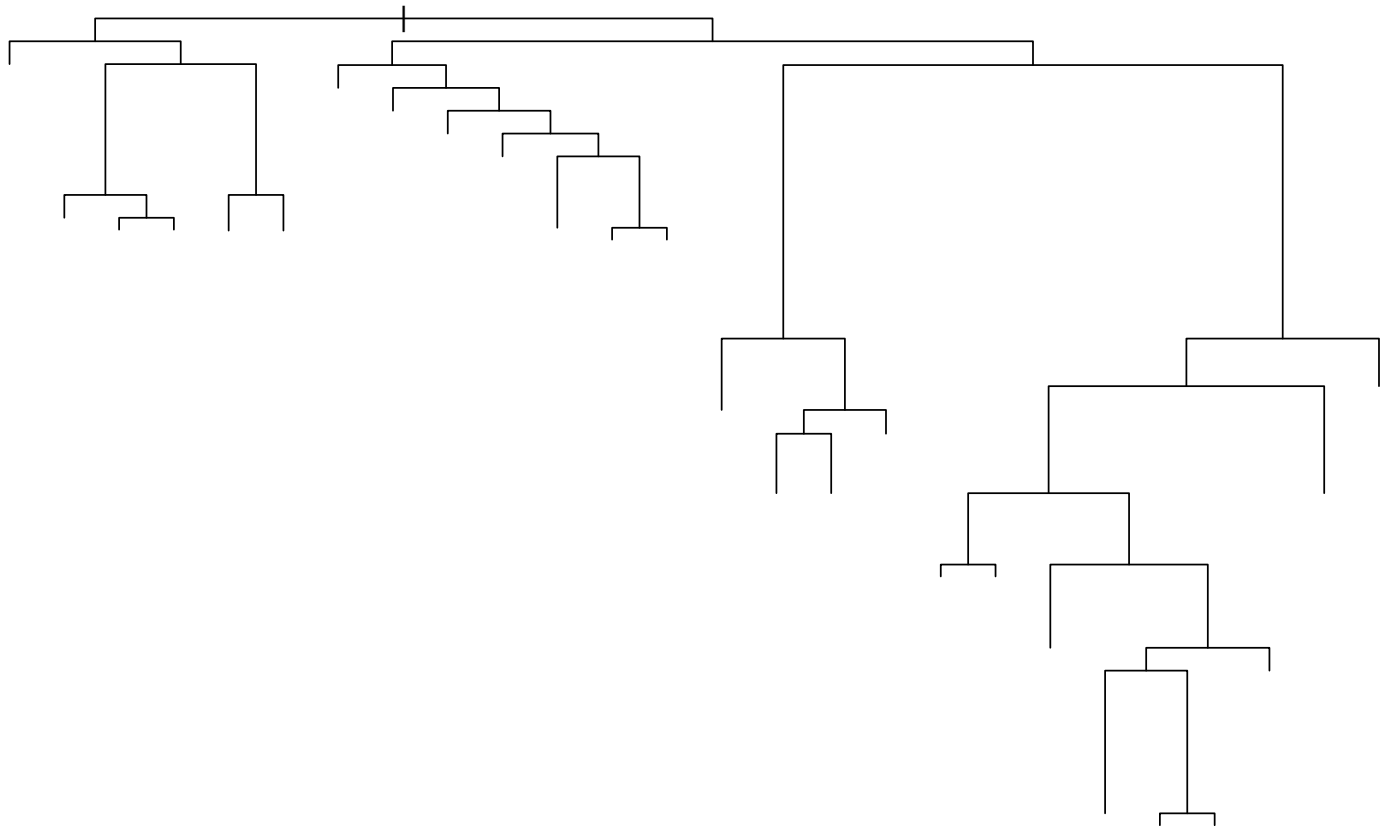
Example

```
plot(modb3)
```



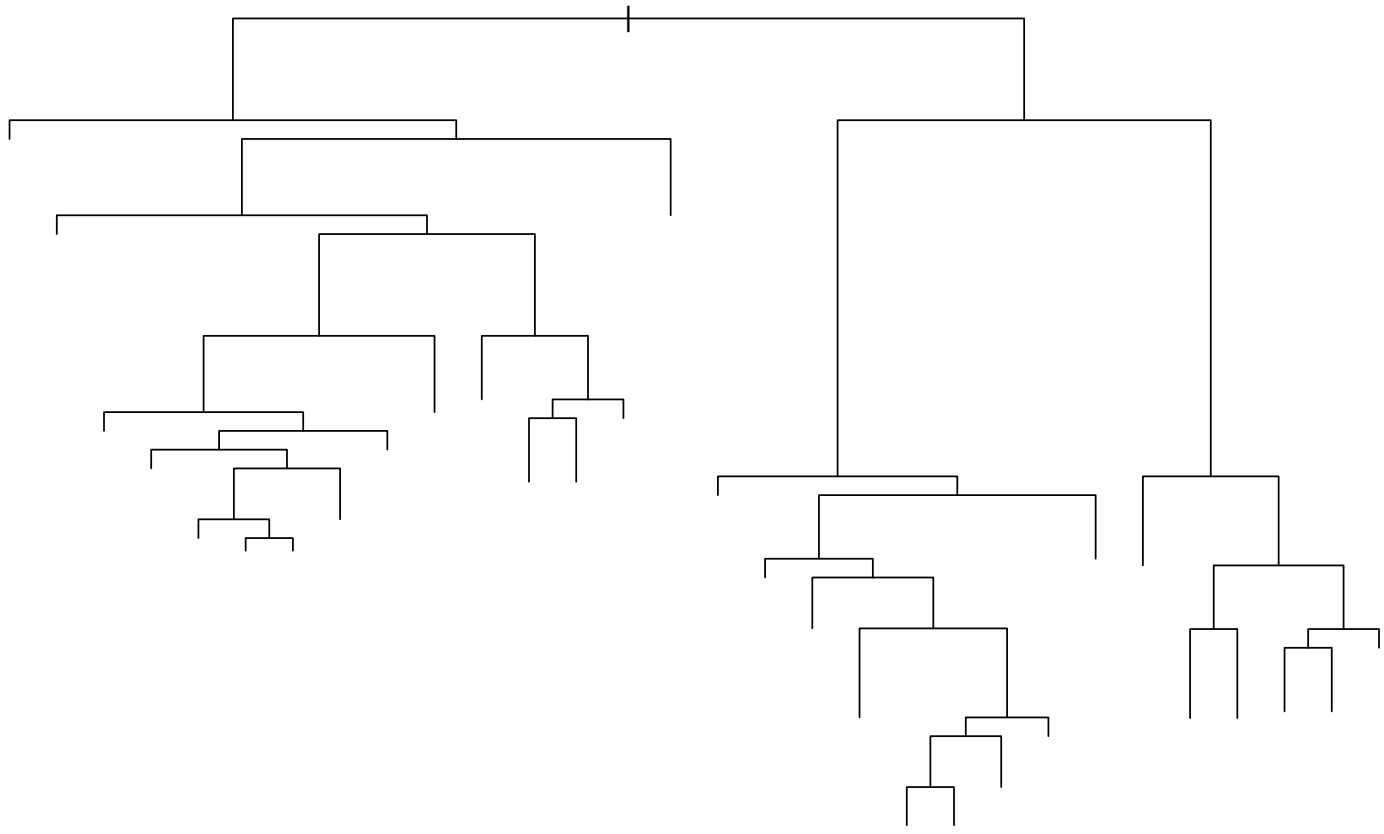
Example

```
plot(modb4)
```



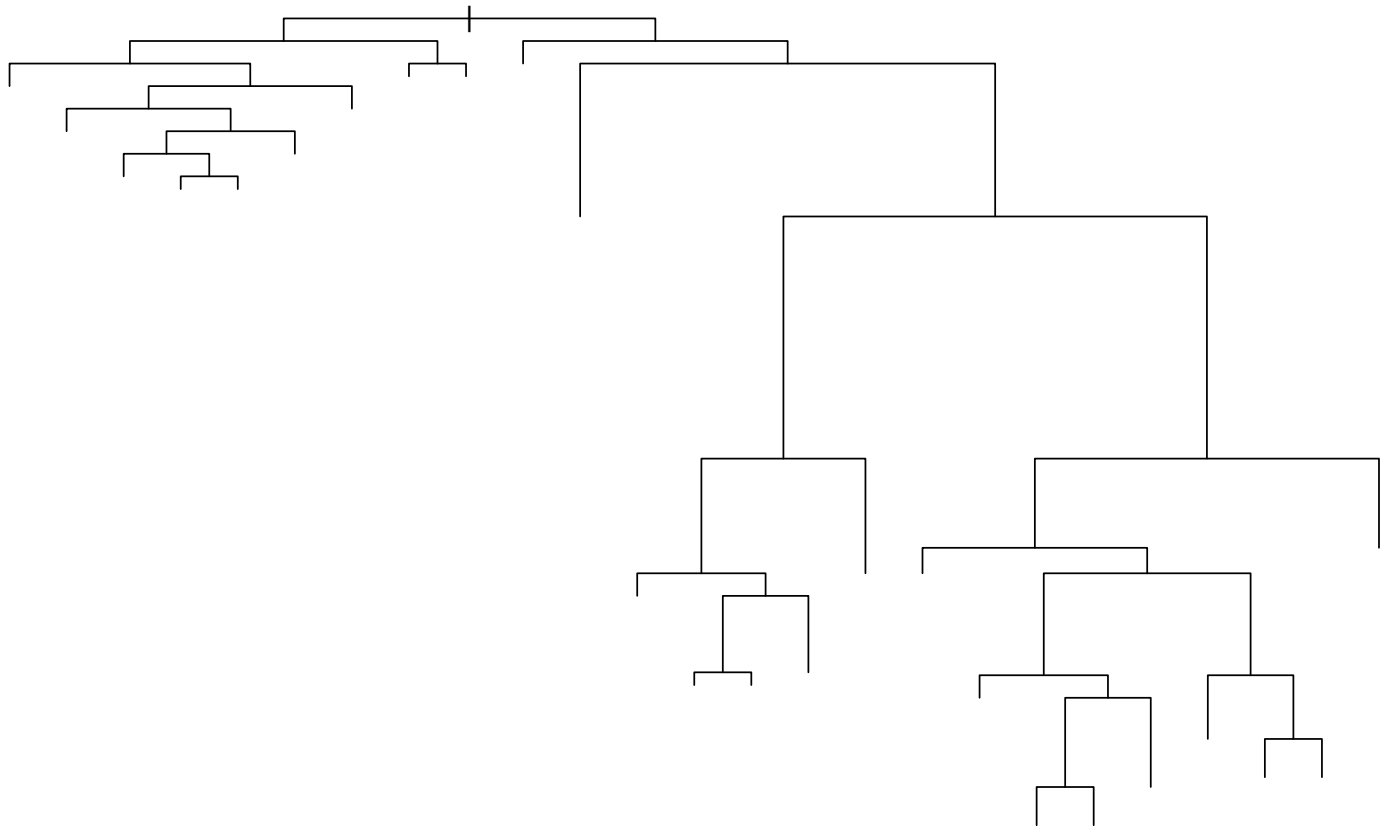
Example

```
plot(modb5)
```



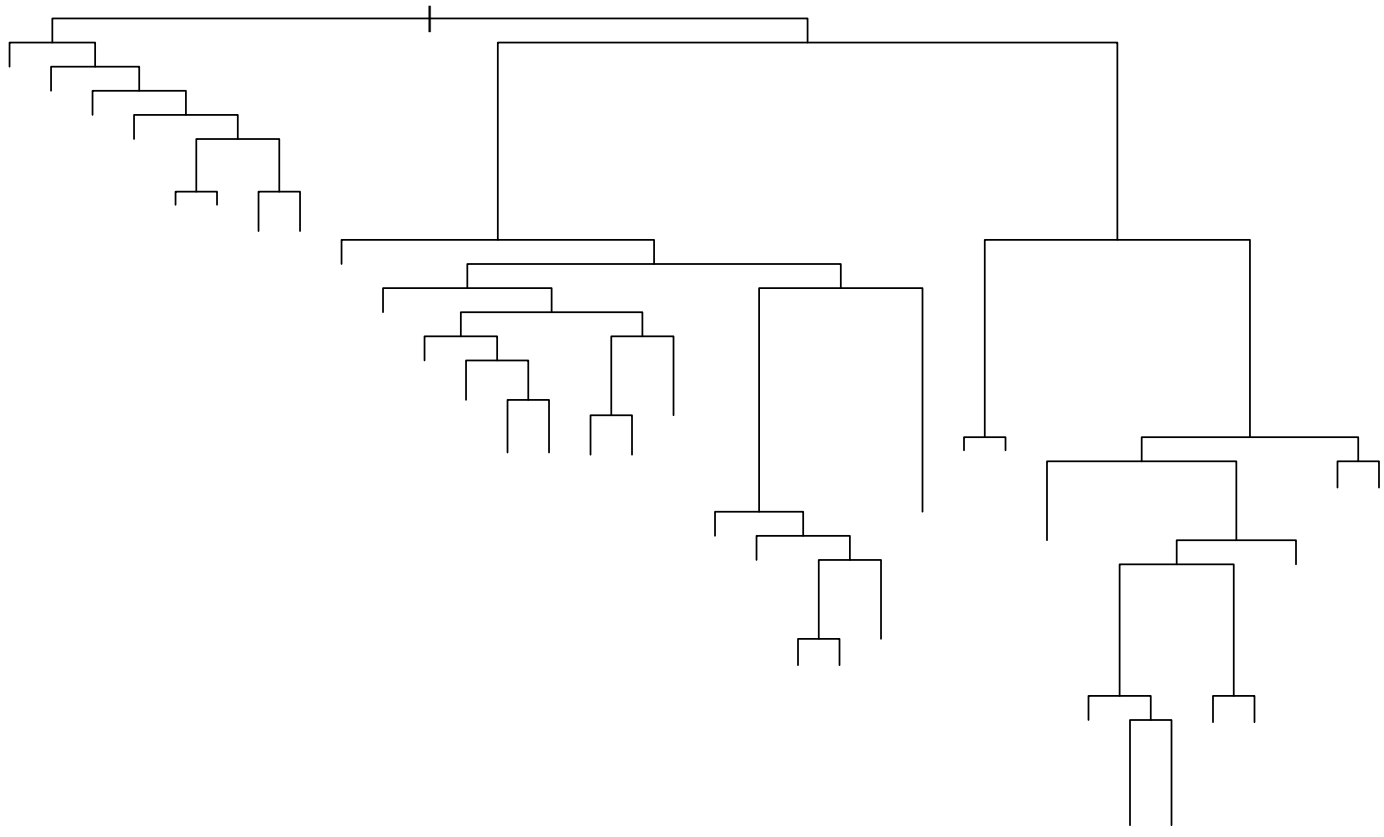
Example

```
plot(modb6)
```

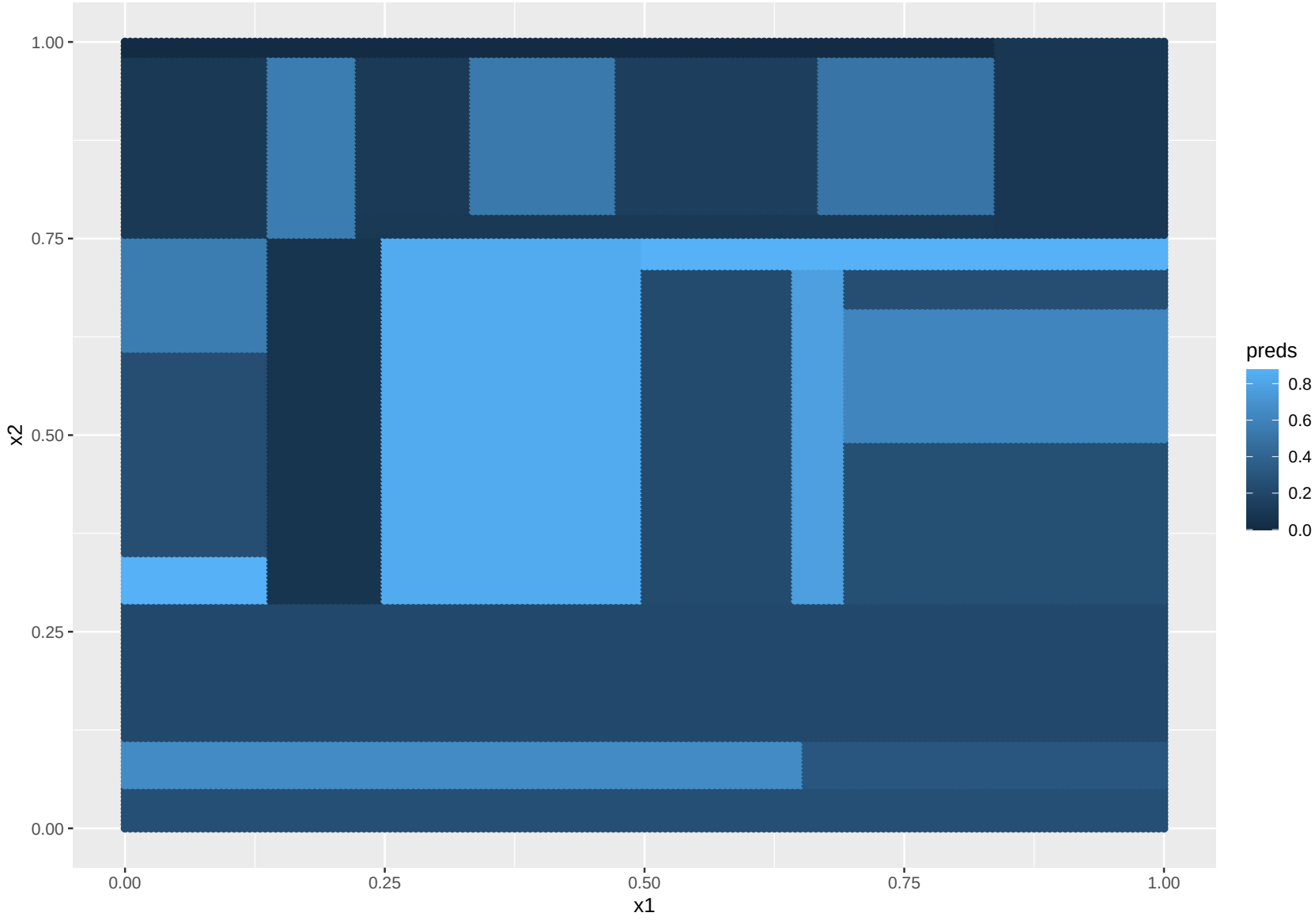


Example

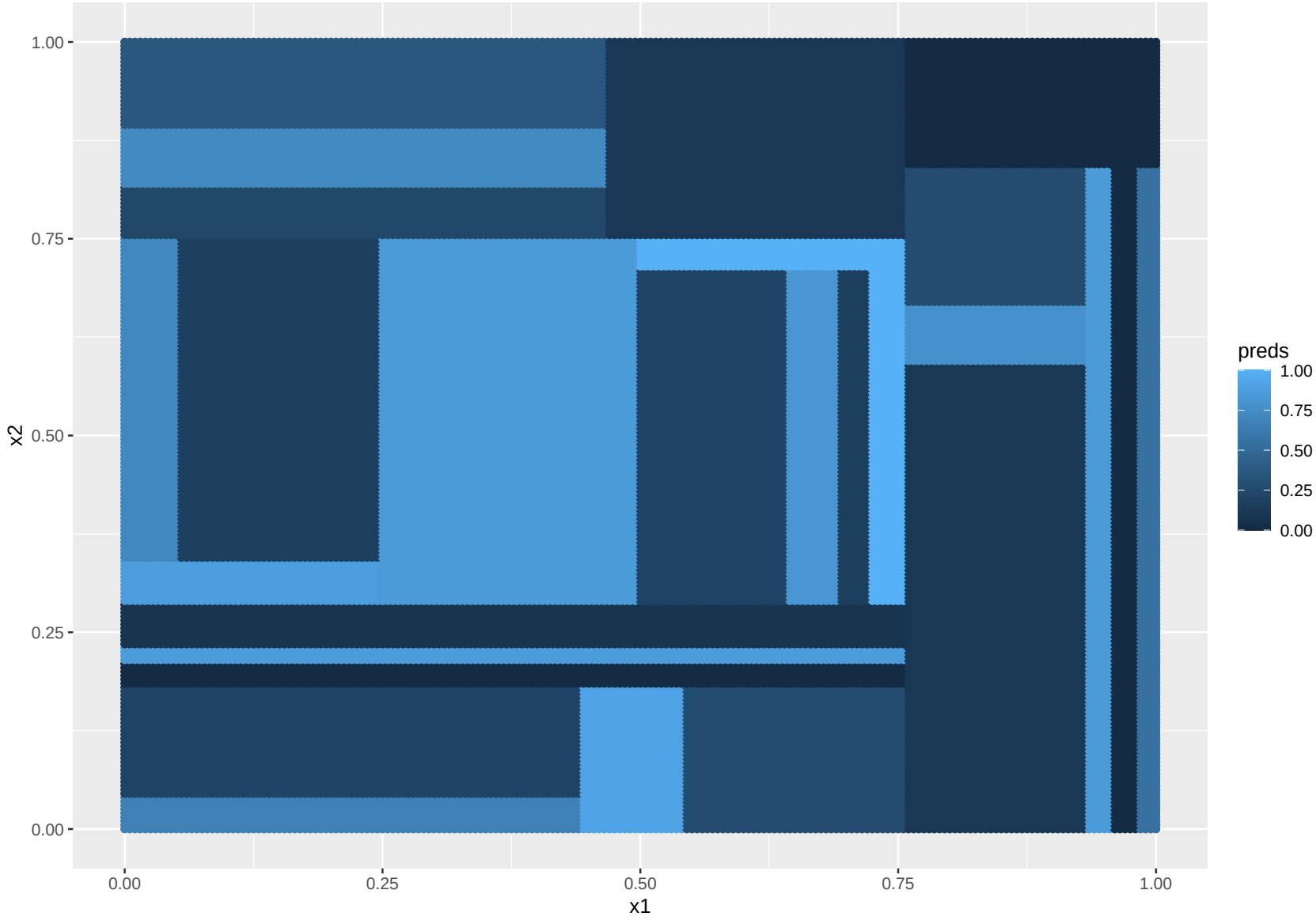
```
plot(modb7)
```



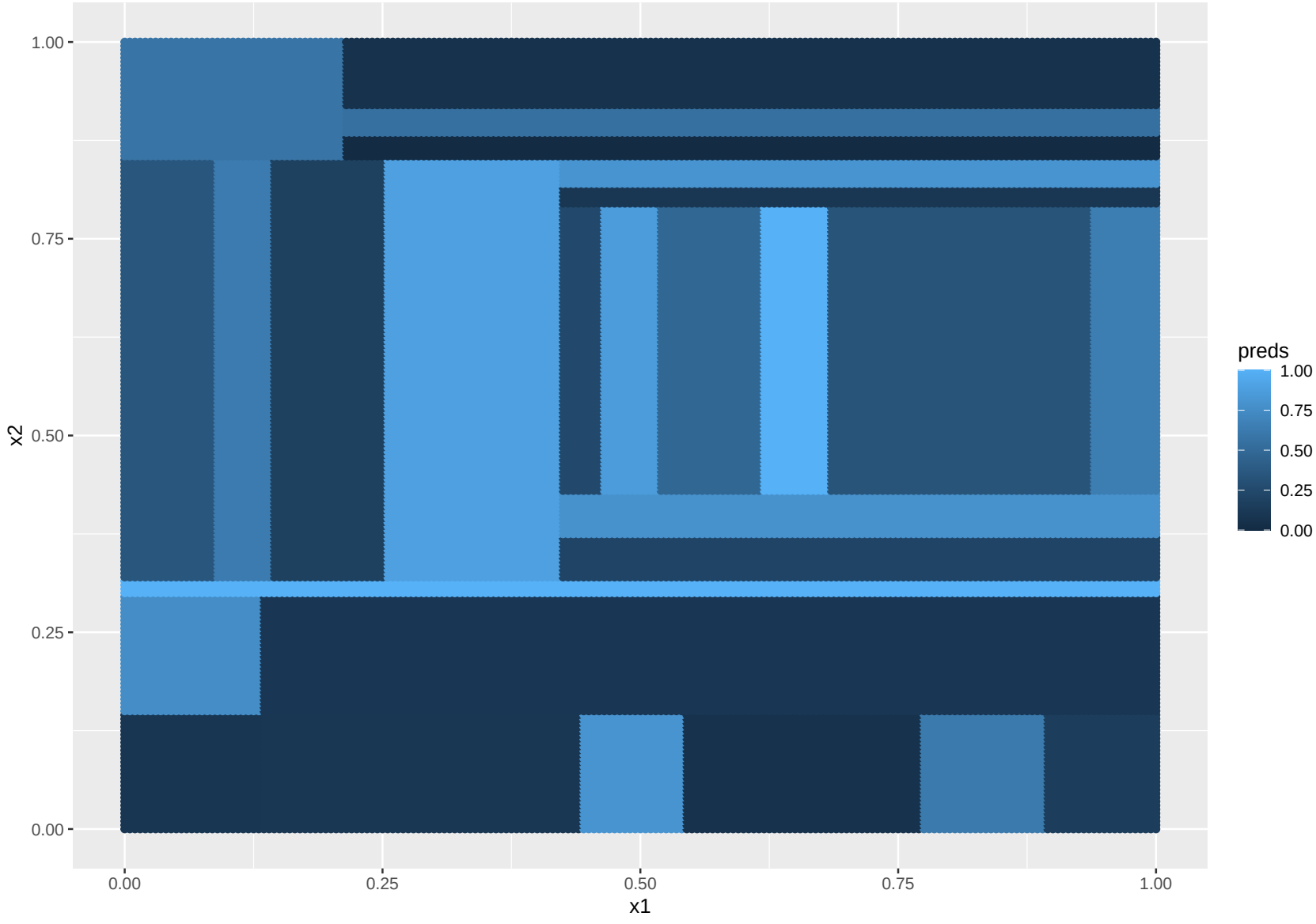
Example – Main



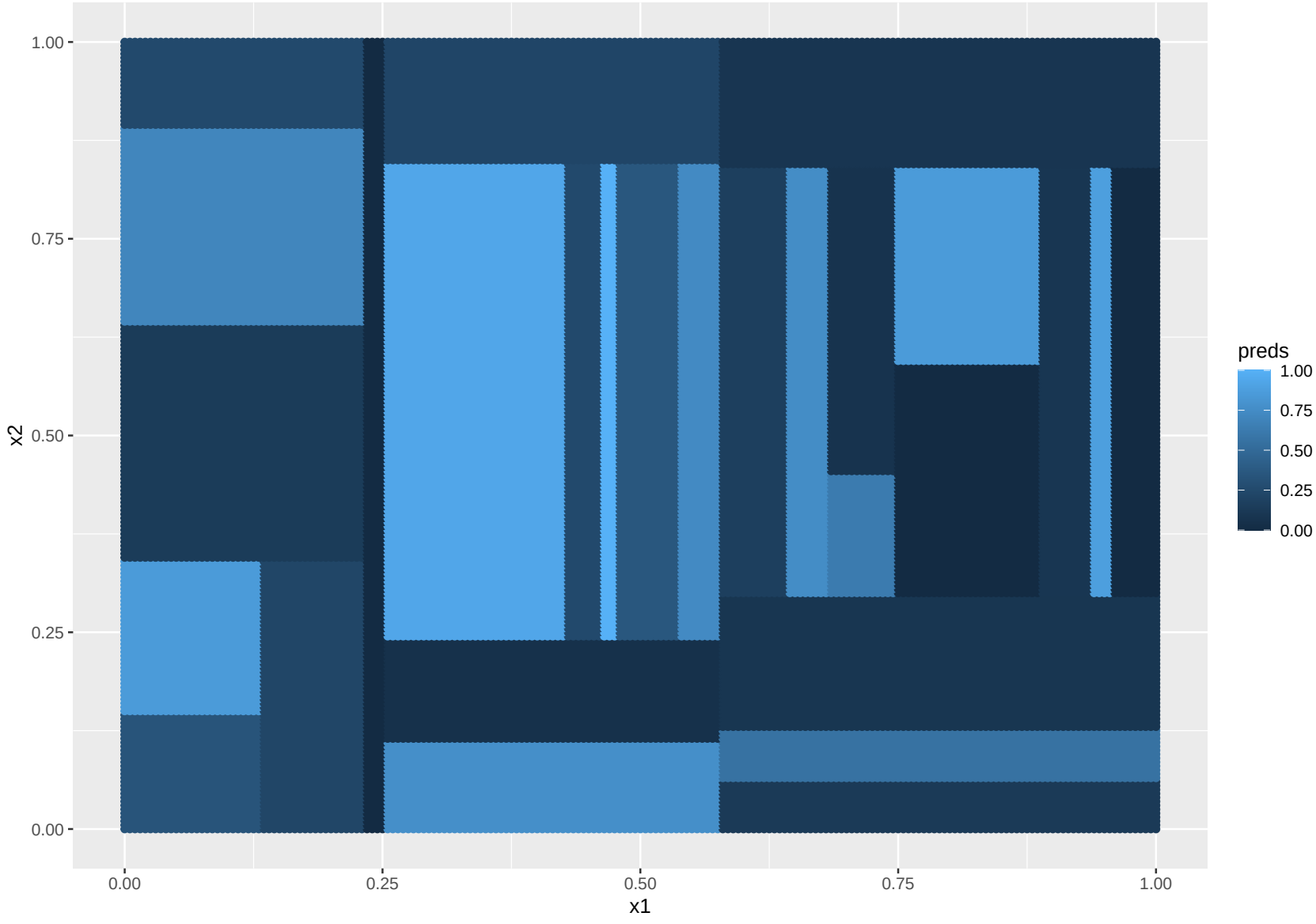
Example – B1


1.00
0.75
0.50
0.25
0.00
x2
0.00
0.25
0.50
0.75
1.00
x1
preds
1.00
0.75
0.50
0.25
0.00

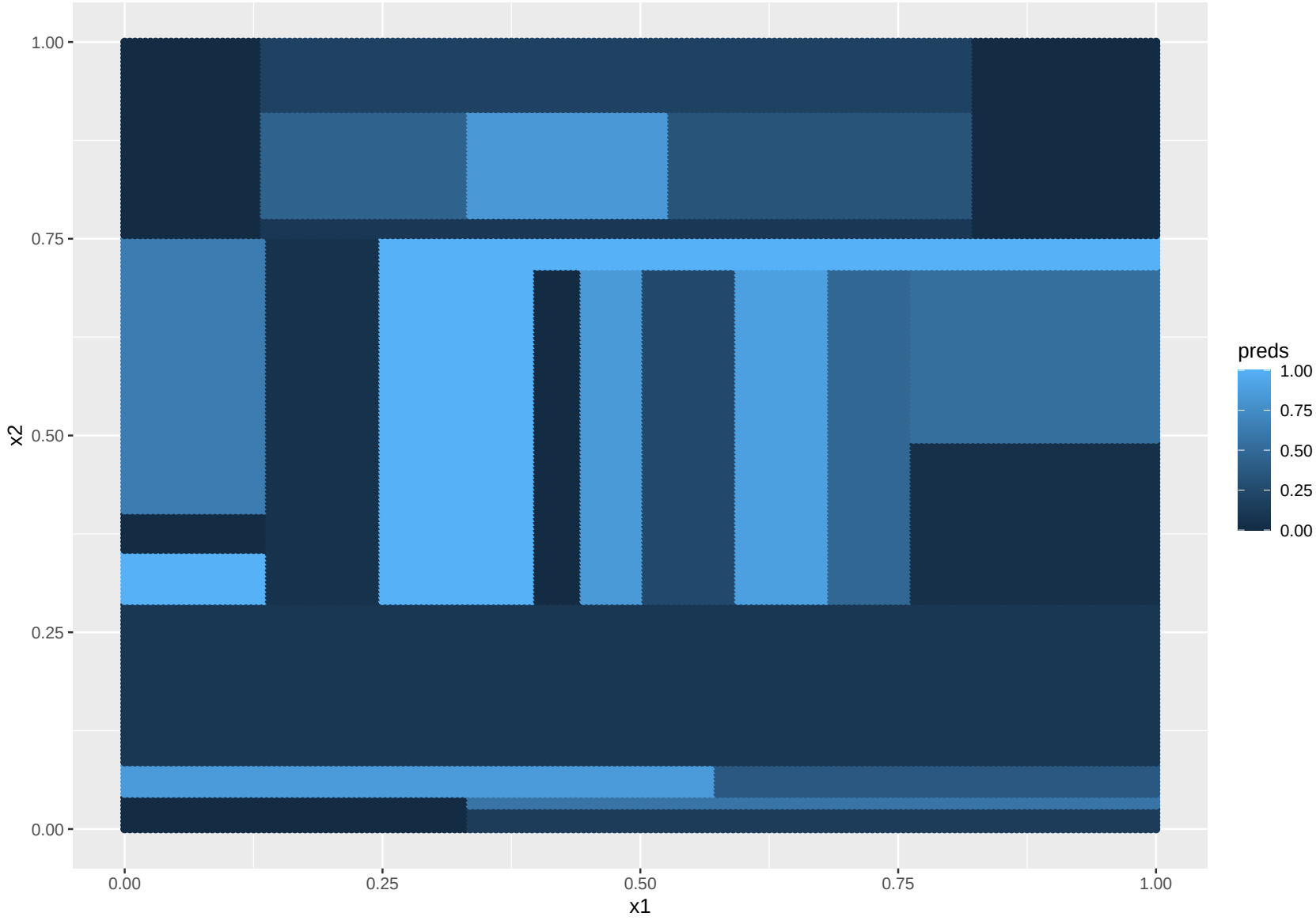
Example – B2


1.00
0.75
0.50
0.25
0.00
x2
0.00
0.25
0.50
0.75
1.00
x1
preds
1.00
0.75
0.50
0.25
0.00

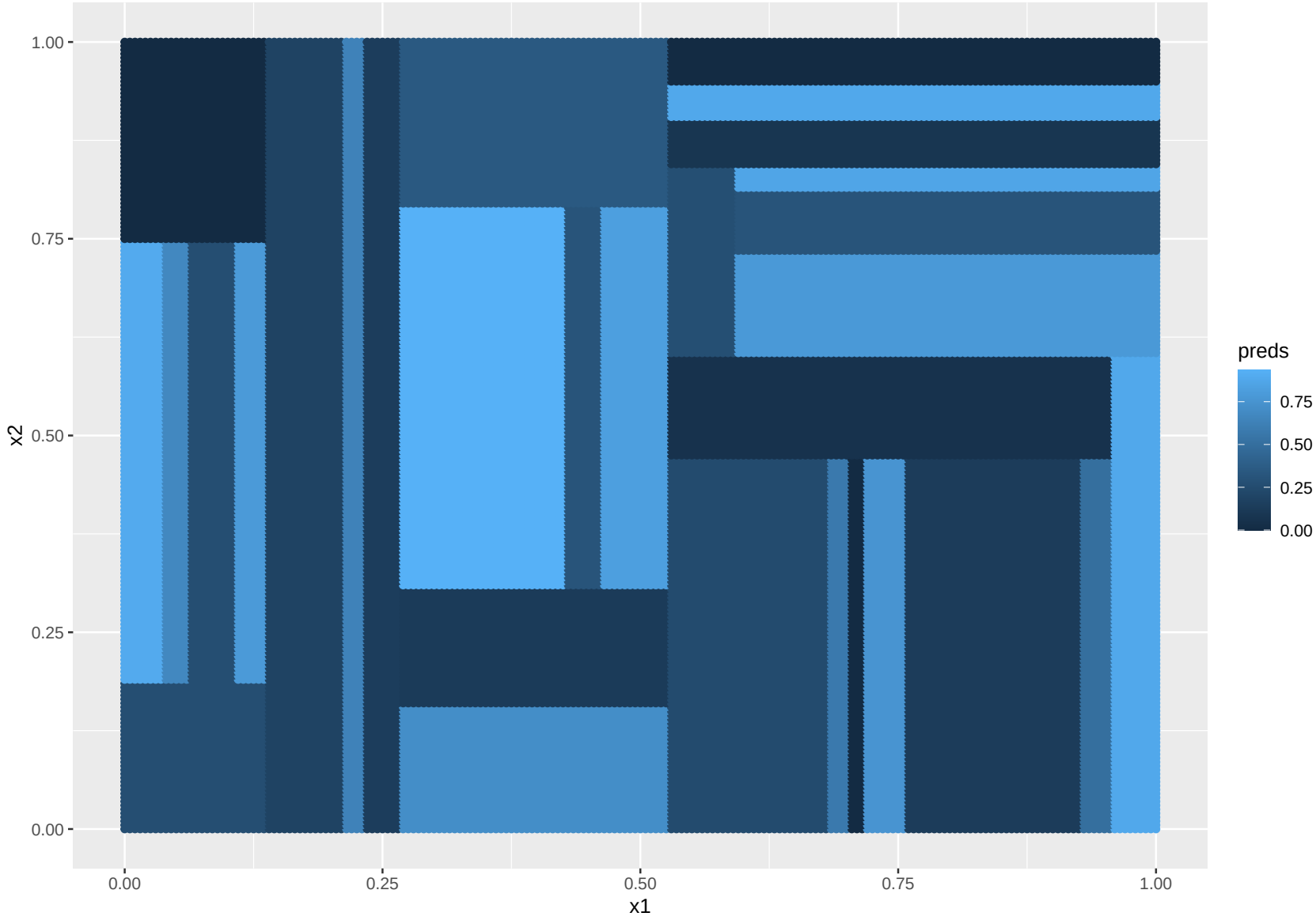
Example – B3


1.00
0.75
0.50
0.25
0.00
x2
0.00
0.25
0.50
0.75
1.00
x1
preds
1.00
0.75
0.50
0.25
0.00

Example – B4


1.00
0.75
0.50
0.25
0.00
x2
0.00
0.25
0.50
0.75
1.00
x1
preds
1.00
0.75
0.50
0.25
0.00

Example – B5


1.00
0.75
0.50
0.25
0.00
x2
0.00
0.25
0.50
0.75
1.00
x1
preds
0.75
0.50
0.25
0.00

Now what?

Now we have 1000 models. Each of which is making wildly overfit predictions. How do we use this?

This is a more general problem that it might look.

- We have a linear and logistic model and we like aspects of each, can we use them together?
- We have several different theories for making predictions about elections (e.g. poll averages, YoY gdp growth & incumbency, betting markets), can we combine them into one forecast?
- We have three different ideas about the optimal stock portfolio for next week, what do we do?

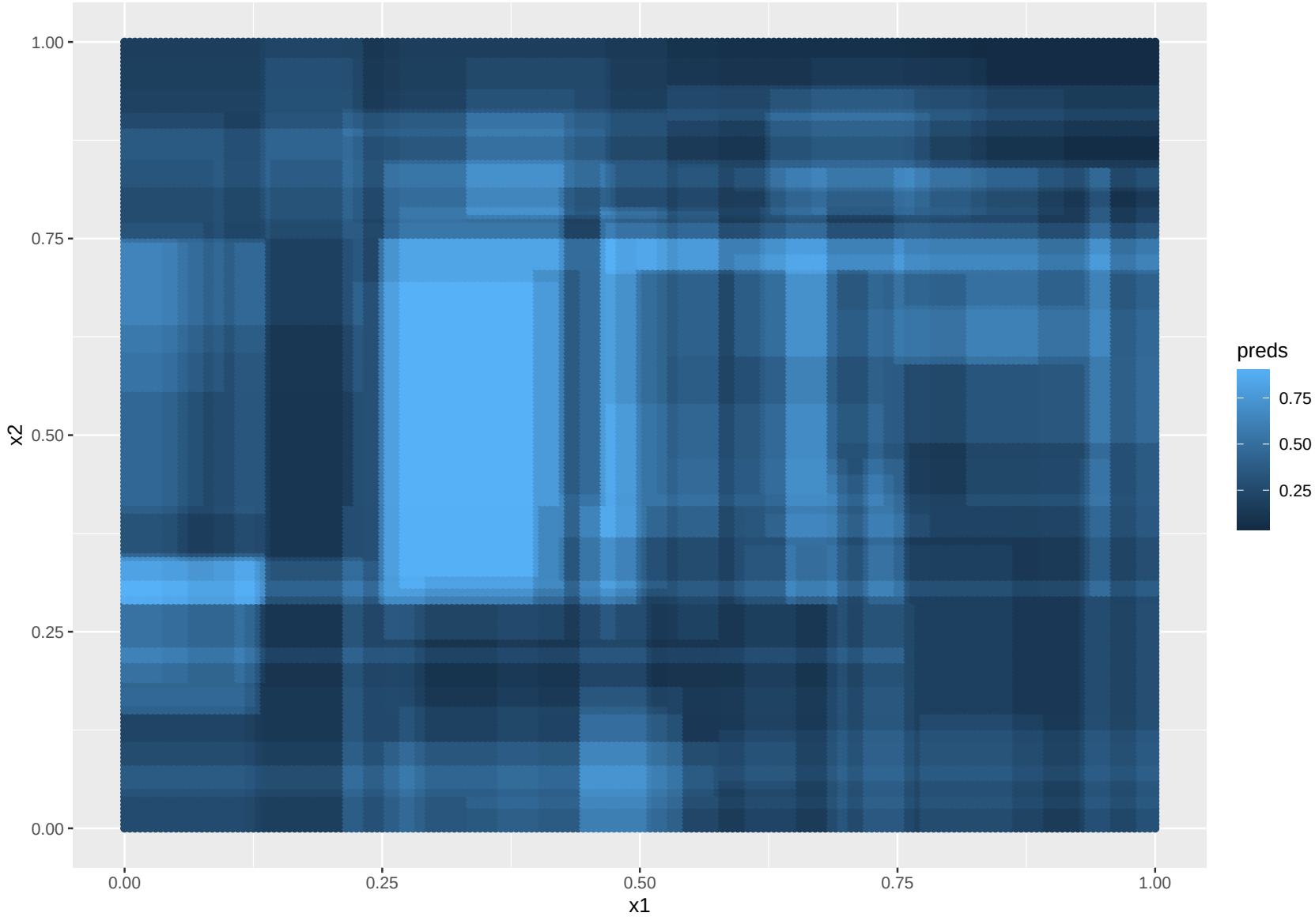
This is the question of building ensemble models.

Ensembles – The bare bones ensemble

We can take an average across the models.

Any time we want a prediction, we average our predicted probabilities across each model type.

Averaged across 7 models



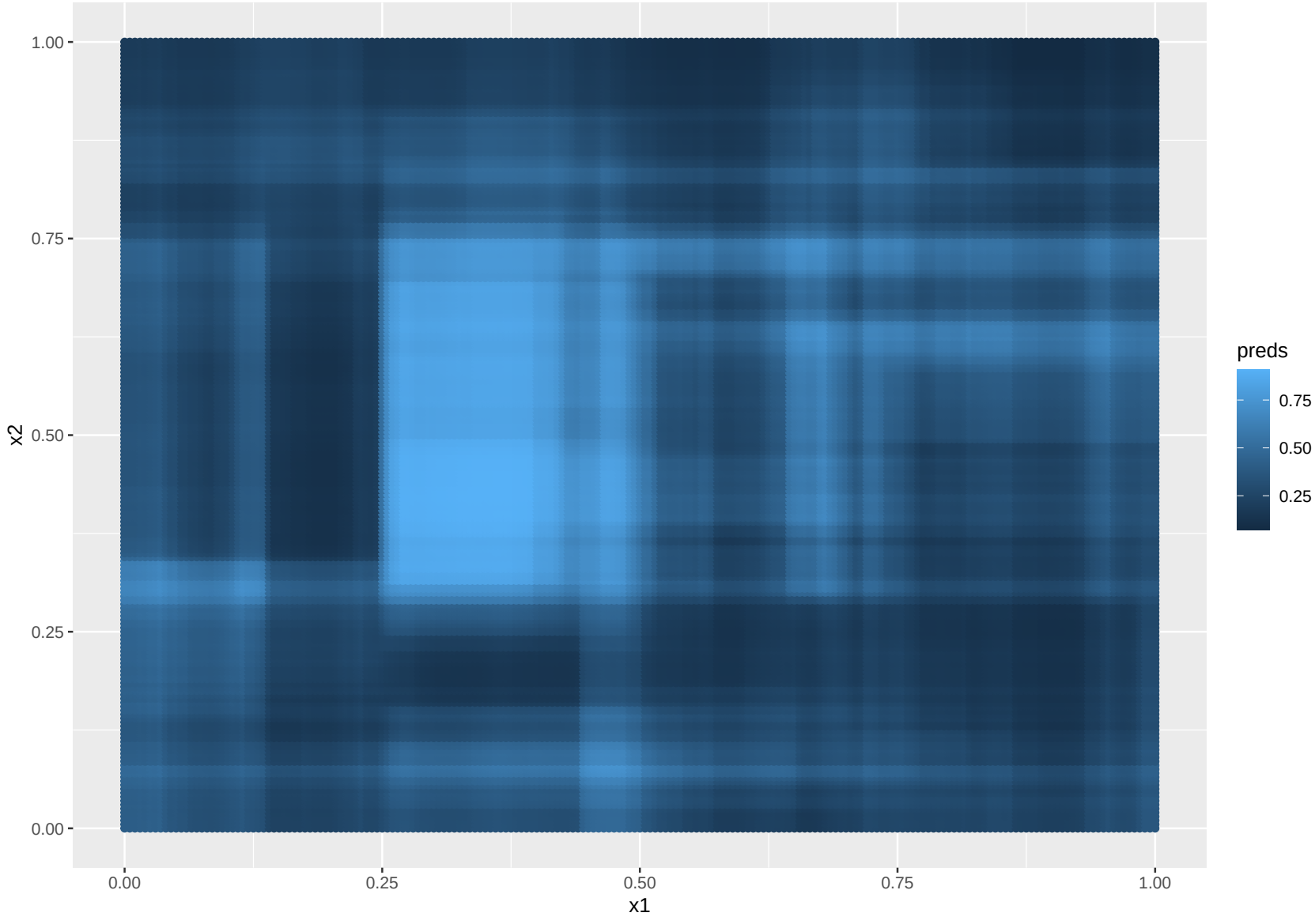
Averaging?

```
mods = lapply(1:1000,resampled_mod)
# Gives a list with 1000 models

pred_helper = function(x,xdata=grid) predict(x,newdata=xdat
preds = sapply(mods,pred_helper)

preds = apply(preds,1,mean)
#preds = rowMeans(preds) #does the same thing
grid$preds = preds
```

Averaged across 1000 models



Averaging

Advantages:

- Helps us deal with model instability
- Also *helps*, but doesn't *fix* overfitting.
- Helps overcome issues between smooth underlying functions and our threshold-y model

Disadvantages:

- Trees are already somewhat slow. Fitting 1000 trees is even slower.
- Naive average may not be optimal? Other things?
  - Boosting, etc – later.

OOB Bonus

Each tree is fit to a subset of the data. (Resampling with replacement means we miss some observations).

Thus each tree has observations that are 'in-sample' and 'out-of-sample'. We can calculate the OOB error for each tree pretty easily.

- We could use this to weight observations and improve our "naive" average

Speed Fixes

There are some simple things we can do to improve speed.

1. We don't need super overfit models. We are doing a lot of averaging, and relying on that.
   - Why not just say "stop building trees after 20 nodes".
2. We don't need to look at all possible variable choices at every single node.
   - Looking at only 25% of variables to find 'optimal split' for each node is 4x faster.
   - We can randomly choose variables each node looks at. Important variables will come up at some point, so the trees won't miss them.

These two innovations bring us to the "random forest".

Random Forests

Forests take the notion of Bagging, and make some minor improvements – mostly in the name of speed.

By introducing variation into the variables under consideration, they create *even more instability* between trees.

But it turns out, because we are averaging across our trees, this leads to improvements in predictive power.

They search across a wider range of models.

Forests Bonus

Forests only consider a subset of variables at each node.

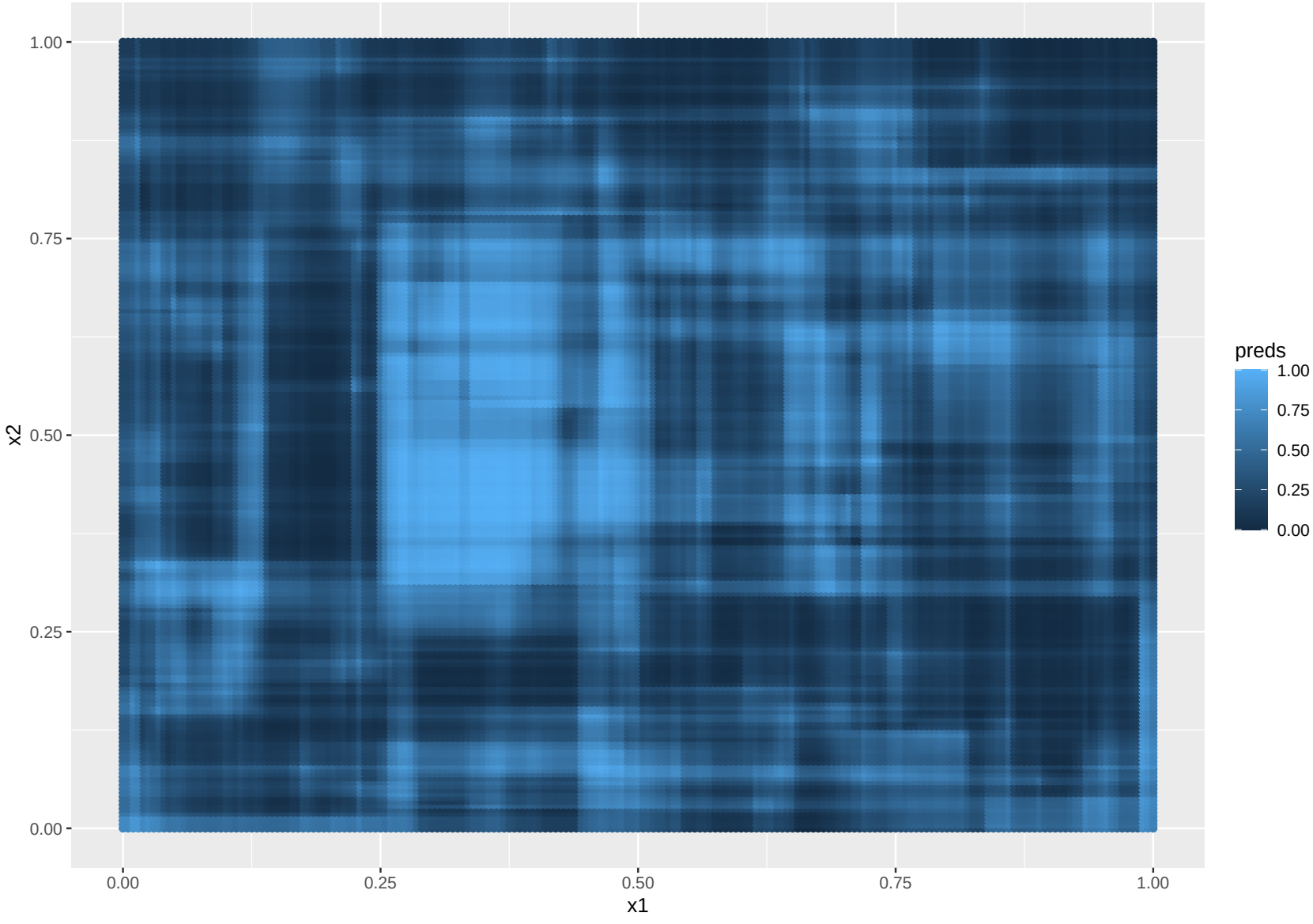
- Thus, we often see direct head-to-head matchups in predictive power between 5 different varaibles.
- And we see which variable was best.
- So we can start to rank the variables "importance".

Forests

```
library(ranger)
forest = ranger(y~x1+x2,data=df,importance="impurity_corre
forest
```

```
## Ranger result
##
## Call:
##  ranger(y ~ x1 + x2, data = df, importance = "impurity_
##
## Type:                             Classification
## Number of trees:                  500
## Sample size:                      500
## Number of independent variables:  2
## Mtry:                             1
## Target node size:                 1
## Variable importance mode:         impurity_corrected
## Splitrule:                        gini
## OOB prediction error:             32.60 %
```

Forests


1.00
0.75
0.50
0.25
0.00
x2
0.00
0.25
0.50
0.75
1.00
x1
preds
1.00
0.75
0.50
0.25
0.00

Forests

We will see more of them (in more depth) next week.

But first, let me ask you this: suppose you had two models that made predictable mistakes?

Could we take the model predictions from our series of models, and use them as inputs for a different, better model?

- Yes. More to think about.

Wrap up

Things to do

No Homework *immediately*.

A quiz will go out on canvas tomorrow. Due before class Tuesday. Graded for participation. Useful for me understanding where the class is. Open book/note/internet. Please don't communicate about it.

There will be another prediction contest starting Monday. You will have until Thursday to make a prediction (i.e. this is for your planning)

See you Tuesday.

Bye!