Ensembles 2: Forests and Boosting

Lecture 11

Connor Dowd

May 4th, 2021

Today's Class

1. Admin
   - Quiz Outcomes
   - Quiz Qs
   - Meetings
2. Prediction Competition
   - Round 1 outcomes
   - Round 2 start
3. Review:
   - Trees
   - Bagging
4. Random Forests
5. Boosting

Quiz

This was a hard quiz – way harder than I would have given if I was grading accuracy not trying to find areas of concern.

The final is going to be much more closely related to the homeworks/a project, than this quiz.

Average grade at last check was a 59%, and I'm quite happy with that. (Looking back at it, feels like even the wrong answers were mostly bad question writing, and not bad answers.)

Quiz Questions

Q1 and Q2 - In vs out of sample losses

As I add variables to a regression my:

- In-sample MSE goes down and R^2 goes up
  - This is the direction for each that is "improving"
- Out of sample we don't know
  - Definitely could each could improve
  - Definitely could each get worse

Q3 - CV 1/many

Can we still have problems with overfitting after cross-validation?

- Yes: because we may have overfit to the holdout sample

But in retrospect, this is not fundamentally different enough from:

- Yes - because our model optimizes the in-sample deviance

$\approx 1/3$ of you put the second one, and got "marked off" for it.

Q4 - basic model types

What can model log(price)?

- Trees, Linear regressions, Linear LASSO, Forests, KNN.

Could you do it with logit? Not if prices $>$ \$3 are in your data.

Could you do it with multinomial logit? . . . . Please don't.

Q5 - basic model types

What can model car make?

- KNN, Trees, Forests, Multinomial Logit

Could you do it with a logit? . . . . again, please don't? But maybe.

Q6 - Complications

LASSOs and Forests represent complications of basic underlying models (logit/linear regression and trees respectively). Because of that, any outcome a logit/linear regression can predict, the logit LASSO and linear LASSO can predict. Likewise any outcome a tree can predict, a forest can predict.

- True.

Notably,"quality" of model may change drastically with this complication (usually for the better).

Q7 - LASSO

Comparing OLS to LASSO coefficients, the non-zero lasso coefficients are:

- Closer to 0.

This is “shrinkage”.

Q8 - CV 2/many

Cross validation:

- True:
  - relies on independence
  - needs modification in time series
    - these are very similar.
- False:
  - is faster than AIC
  - does not help us capture model uncertainty
  - cannot help determine optimal tree depth

Q9 - AIC/BIC

Some model outputs:
Model A: AIC = 2100.2, BIC = 2428.3, Nparams = 232
Model B: AIC = 2093.9, BIC = 2456.9

- True:
  - we can find sample size for model A.
    - AIC = Deviance+2Nparams, BIC = Deviance +log(n)Nparams.
  - If same data, can find params in model b.
  - If not same data, shouldn't compare AIC/BIC.
- False:
  - A has better AIC
  - B has better BIC
  - Possible to determine nparams in model b.
  - We know the data is the same for each
  - We know the data is different for each
    - I don't think this is true, you could maybe persuade me

Q10 - ROC Curves

- True:
  - ROC curves illustrate a tradeoff between two types of errors
  - AUC can be used for selection
  - Slope of the ROC can help us choose thresholds
  - Adding variables makes AUC higher
    - Not guaranteed – extremely likely. Close relationship to deviance.
- False:
  - ROC curve higher everywhere with more variables
  - Adding variables makes out of sample AUC better
  - In sample ROC curve always above 45 degree line
    - We saw the opposite happening in the last homework
  - OOS ROC always below 45 degree line

Q11 - KNN (w/ K=2)

- True
  - Makes predictions with two nearest points
  - May need a tiebreaker
- False
  - Uses points within distance of 2
  - Makes predictions using all points
    - Debatable. Like, every point could be involved in some prediction.

Q12 - Trees

- True
  - Greedy Algorithm
  - Predict mean or plurality at each node
- False
  - Stable
  - Work well with factor predictors
    - Debatable.
  - fit a linear regression at each node
    - Wouldn't it be great if they did? (So slow though)

“Midterm” Quiz Wrapup

- Overall very happy
- Looks like most people are understanding most of the content pretty well.
- If you have questions, had issues, have concerns, please talk to me.

Meetings

I've (finally) emailed people who wanted a one-on-one meeting.

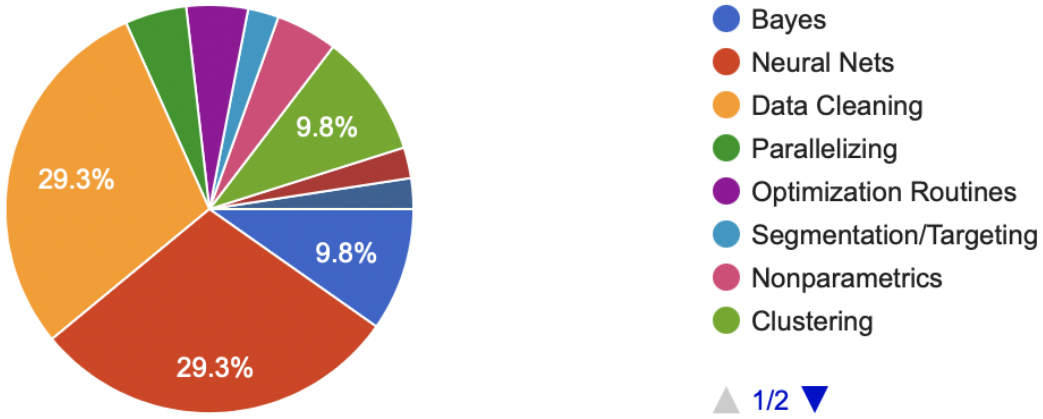
If you didn't sign up for that, but would like to now, email me.

I *may* be able to do outdoors/in-person meetings in chicago starting next week – if you're interested in *that* email me. Likely to be small groups?

Content for Final weeks

If you had to pick only one of the above topics, which would you pick?

41 responses



Data cleaning, Neural nets, Bayes, Clustering all got top picks from folks.

- Data cleaning I'm honestly not sure how to teach, looking into it – it will involve a *lot* of coding. May do an optional assignment or something.
- Neural Nets I'm happy to do, fair warning, I'm no expert in them. Learning experience for all of us.
- Bayes we will do. Probably Thursday

Prediction Competition

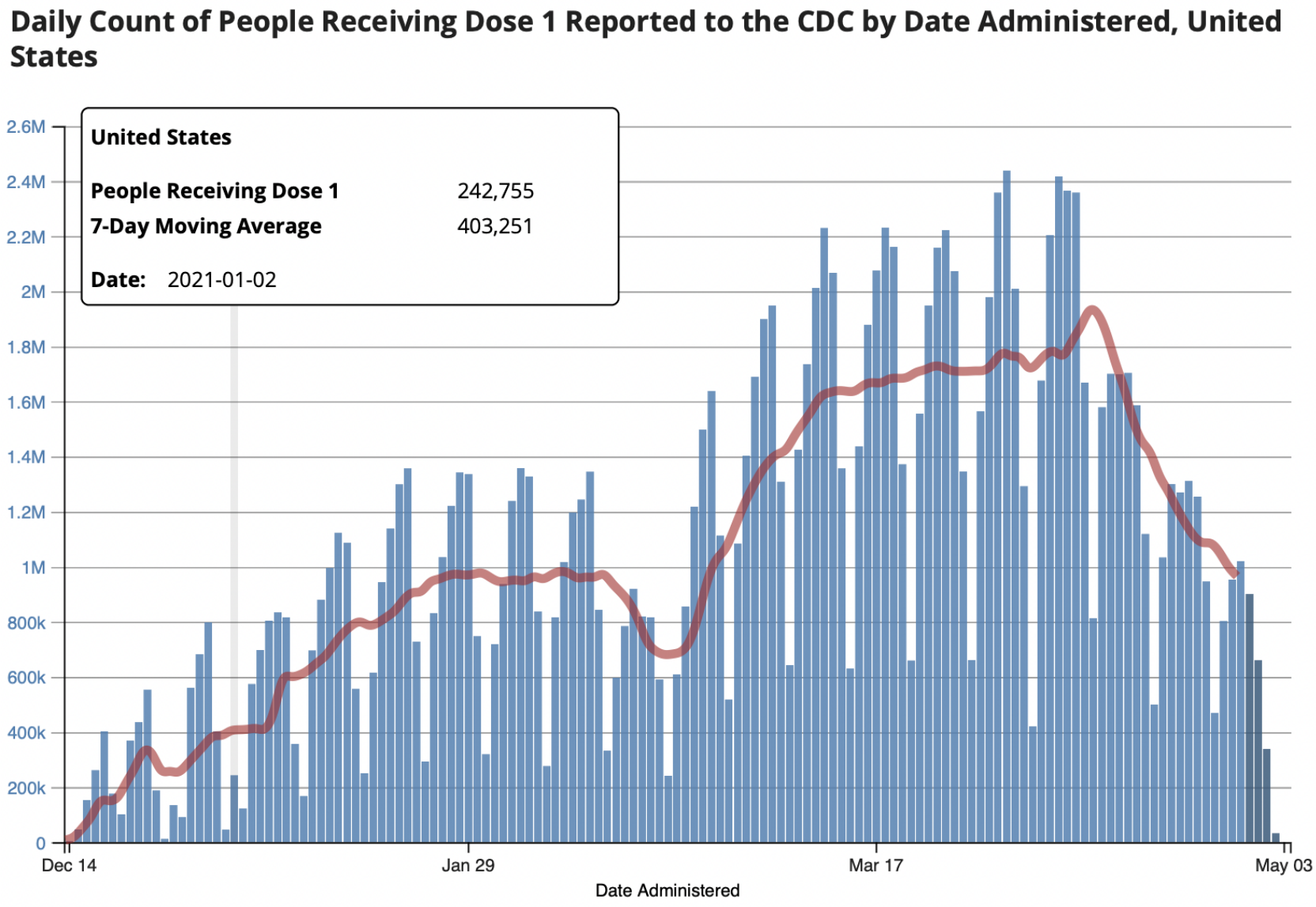
Round 1

IS OVER!

Final result: 144.89 million people. Link

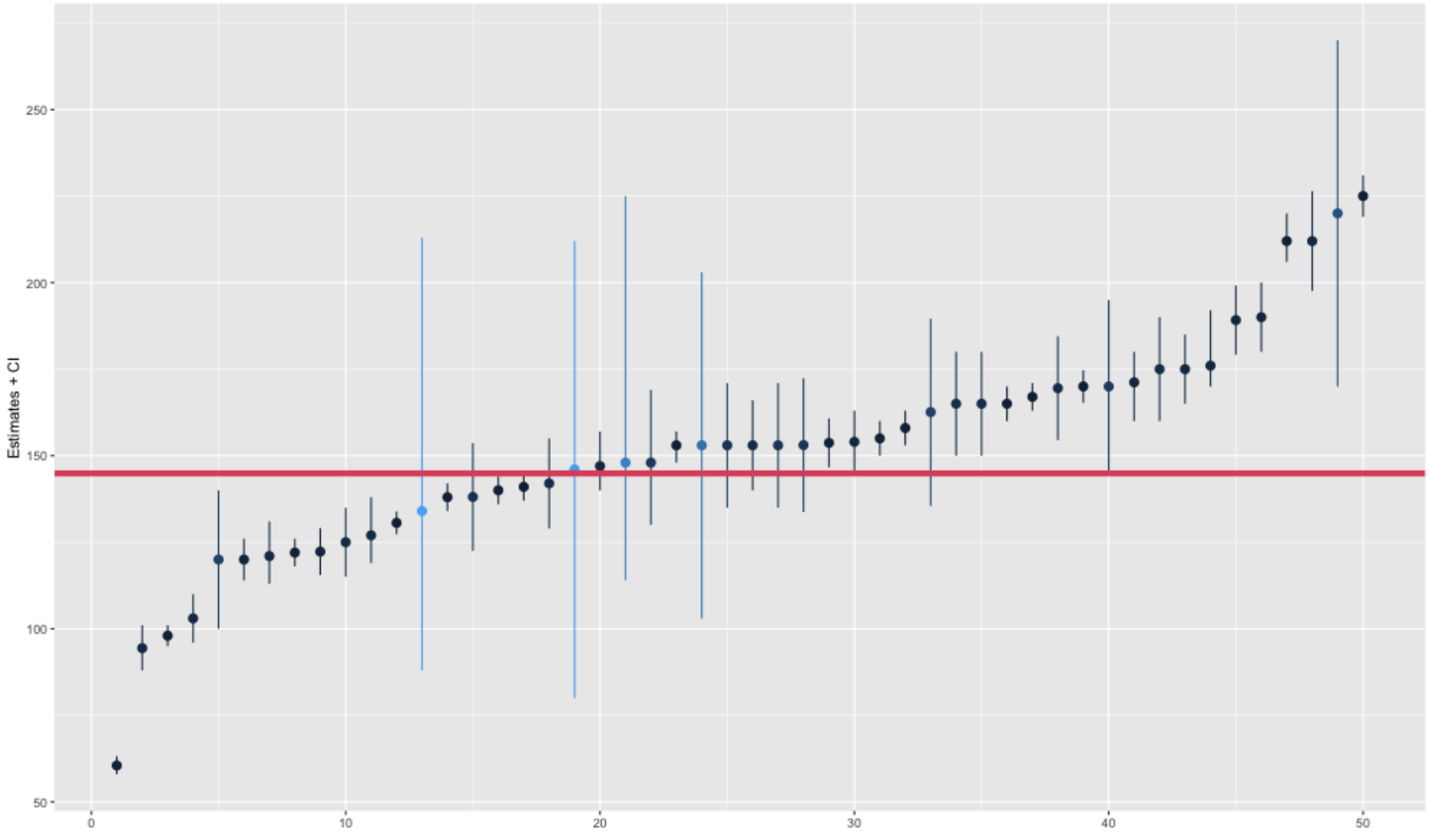
My prediction: 148 million. [130,169].

Recap


Daily Count of People Receiving Dose 1 Reported to the CDC by Date Administered, United States
United States
People Receiving Dose 1 242,755
7-Day Moving Average 403,251
Date: 2021-01-02
2.6M
2.4M
2.2M
2M
1.8M
1.6M
1.4M
1.2M
1M
800k
600k
400k
200k
0
Dec 14
Jan 29
Mar 17
May 03
Date Administered

Winners

We have two winners:



Winners

- Caden Kalinowski: 147 million
- Jessica Zhan: 142 million

Honorable Mention:

- Keda Song: CI of [137,145] is the smallest to contain the truth.

Unfortunate:

- One submission exactly the same as mine after the deadline
- The closest entry was by someone who dropped the class.

Methods:

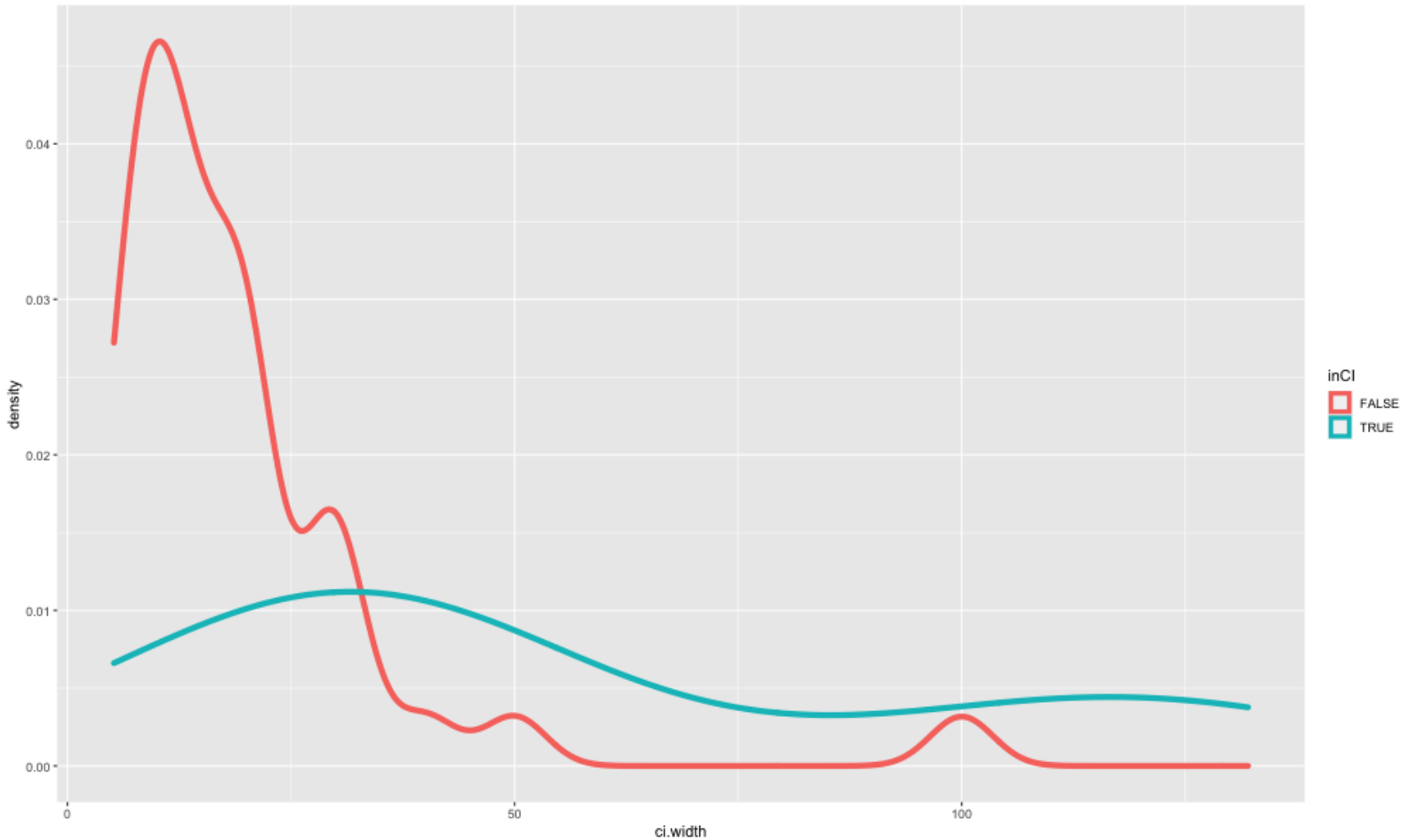
- No idea, nothing was disclosed – prompting a rule change going forwards.

Prizes

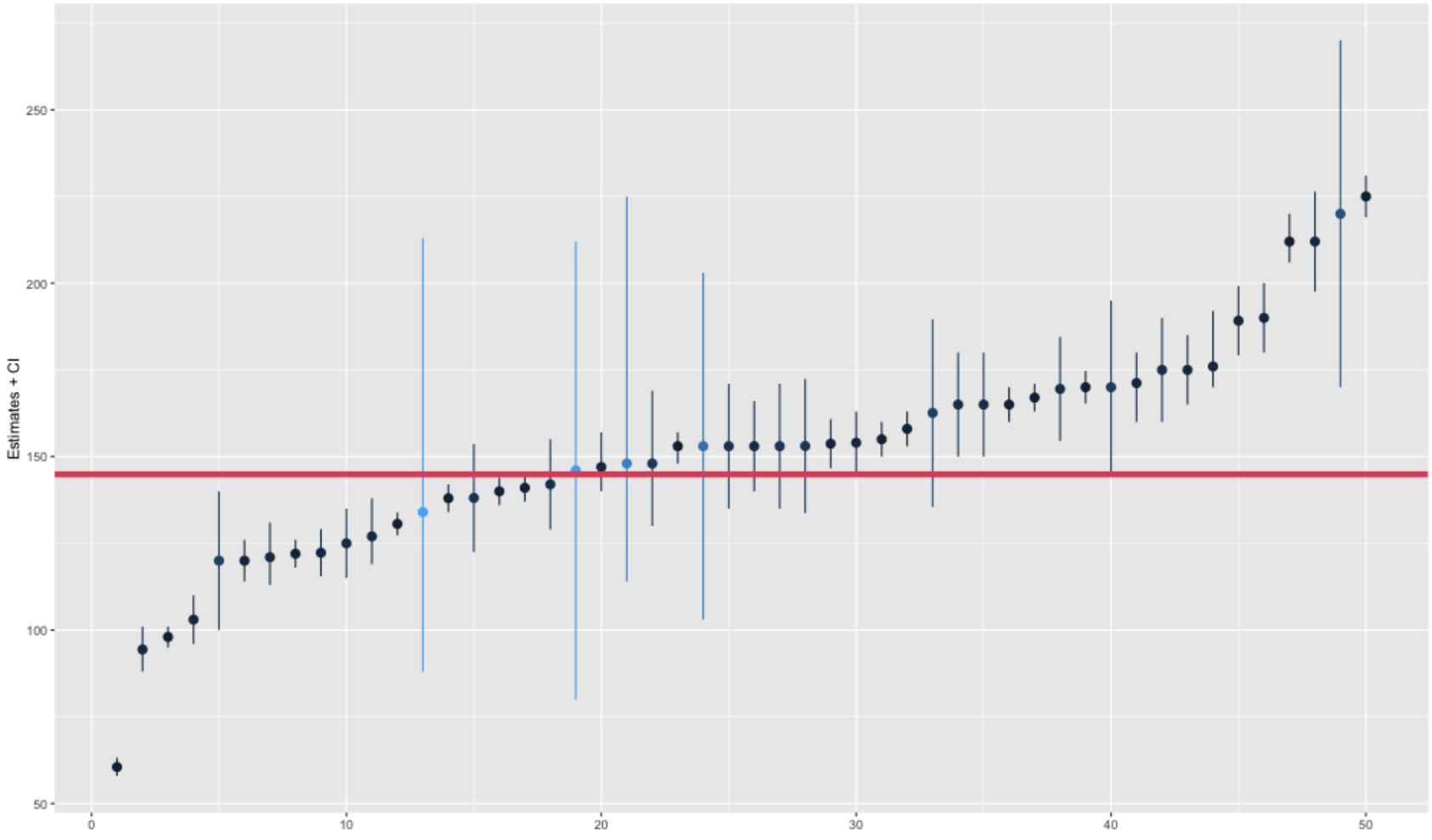
- I've been caught up with administrative nonsense. Working through 'what is allowed' – will keep you three apprised.

Two quick asides: CI widths

- The CIs were really questionable. (Arguably optimized for winning?)
- Less than 1-in-3 CIs contained the true value – despite stated goal of 90%. (Some correlation here)



Alternate view, CI.widths



Prediction Averaging

The average class prediction did very well.

- Mean(Estimate) = 151.7
  - Median = 153
- Mean(Lower Bound) = 137.7
  - Median = 136.5
- Mean(Upper Bound) = 167.5
  - Median = 169.5

The mean, as a prediction, would have beaten 85% of the individual predictions.

This is the essence behind using 1000s of trees. And building ensemble models more generally. And the stock market/ efficient market hypothesis. And...

Round two: US Case Counts on May 9th.

Predictions Competition 2

I want to know how many Covid cases will be detected in the US on Sunday.

Predictions due Friday midnight.

This is less like an average across a month, more like a single number. Much harder to forecast. It is much more likely that a large fraction of you beat my prediction in this setting (more noise), so this is now a proper competition. Top 3 in two different categories will get *something*.

- The mean and median predictions of the group will also be entered as competitors.
  - Unlikely to do well in this case.

Details

Three predictions, only two really matter.

1. Prediction – judged on MSE.
2. Prediction – judged on price-is-right rules.
   - "Don't cause panic" by going over
3. P[x $<$ 25,000].

Only 1 and 2 are scored, because scoring one-off binaries is *really* tough. (hard to not create incentives for)

Details on webpage.

Other details like source, etc are on the website. You have more time, so if you have questions, there should be fewer issues.

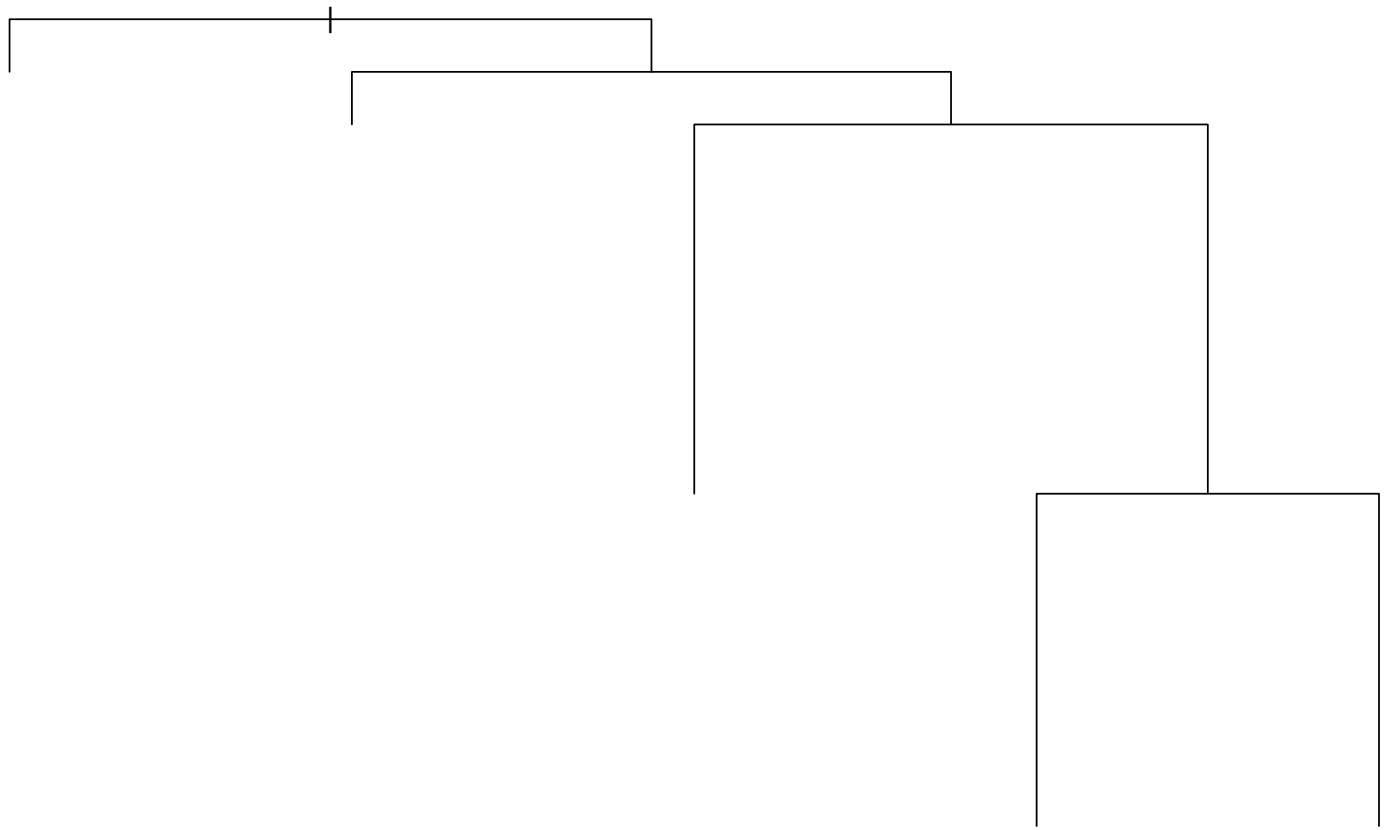
Canvas submission isn't live yet.

Review

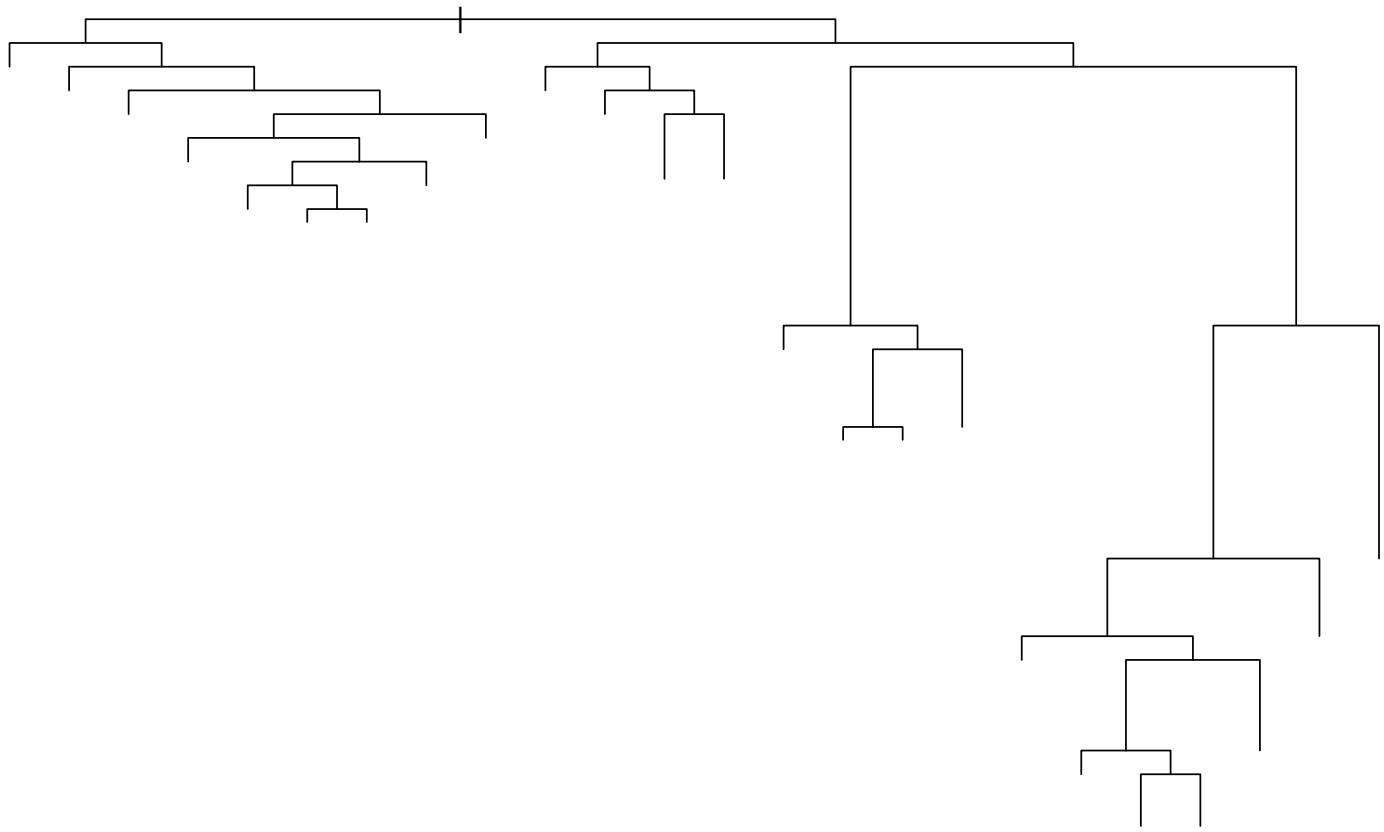
Reminder

- Trees
- Bagging
- Forests

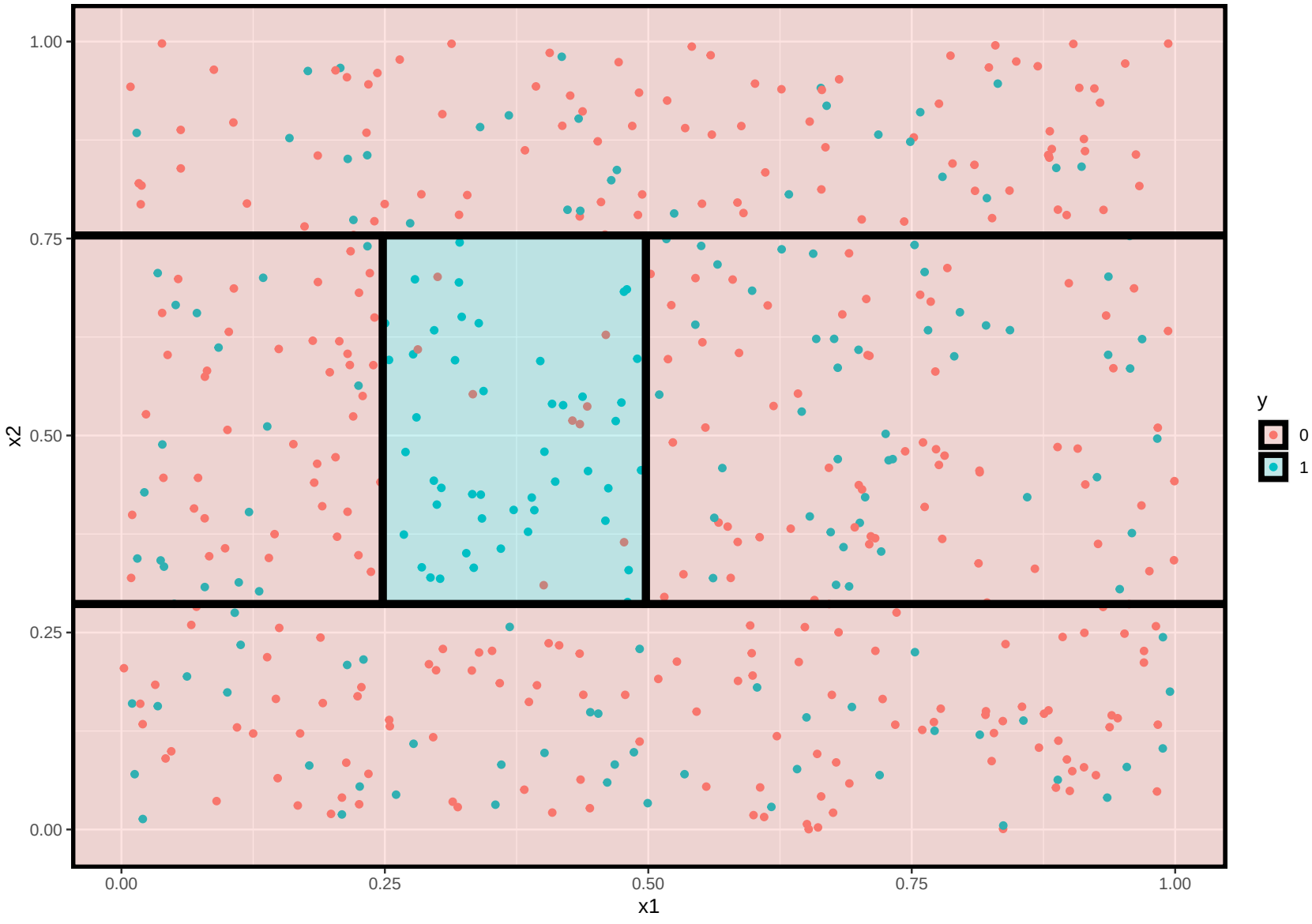
Trees



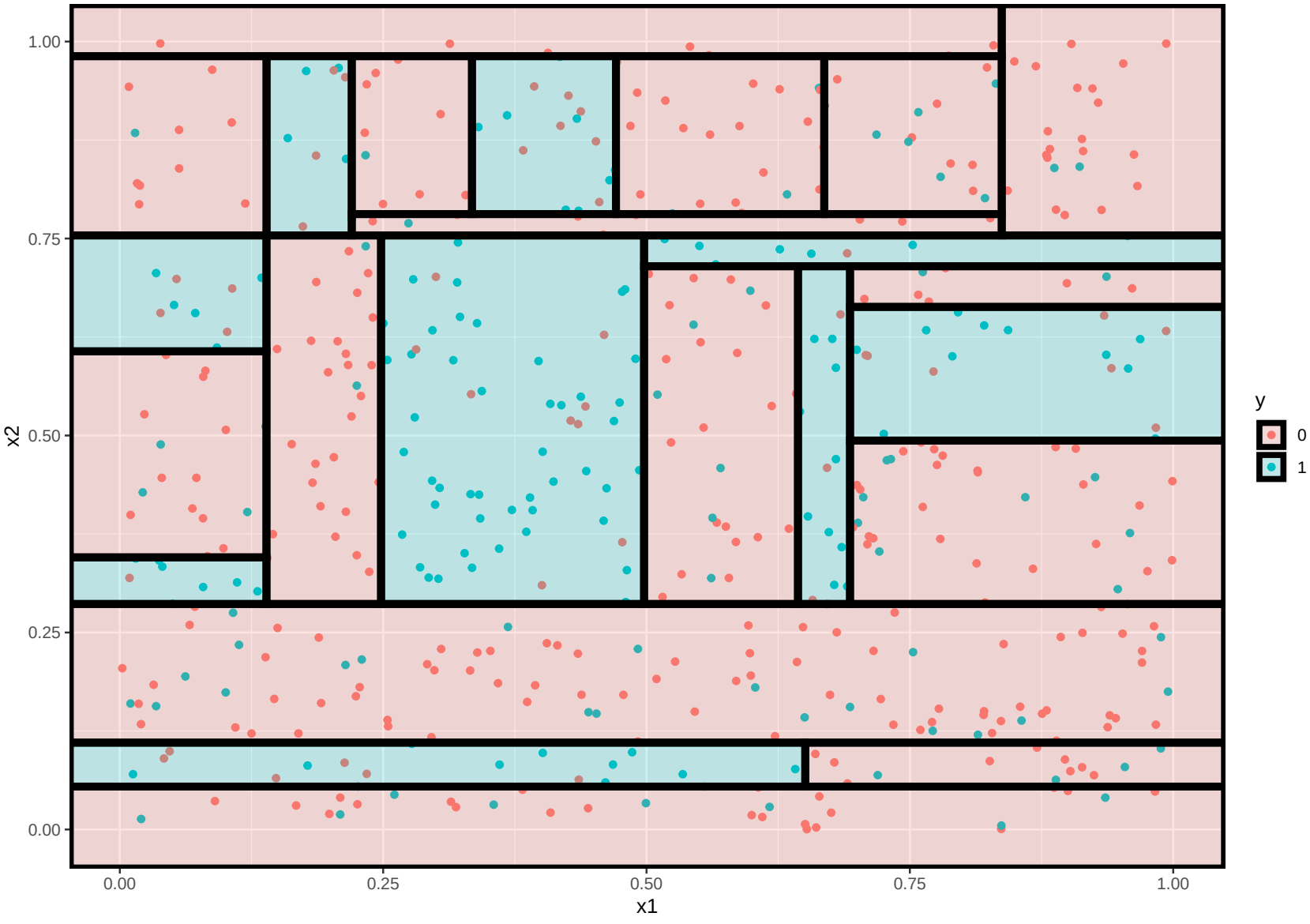
Trees



Trees



Trees



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.

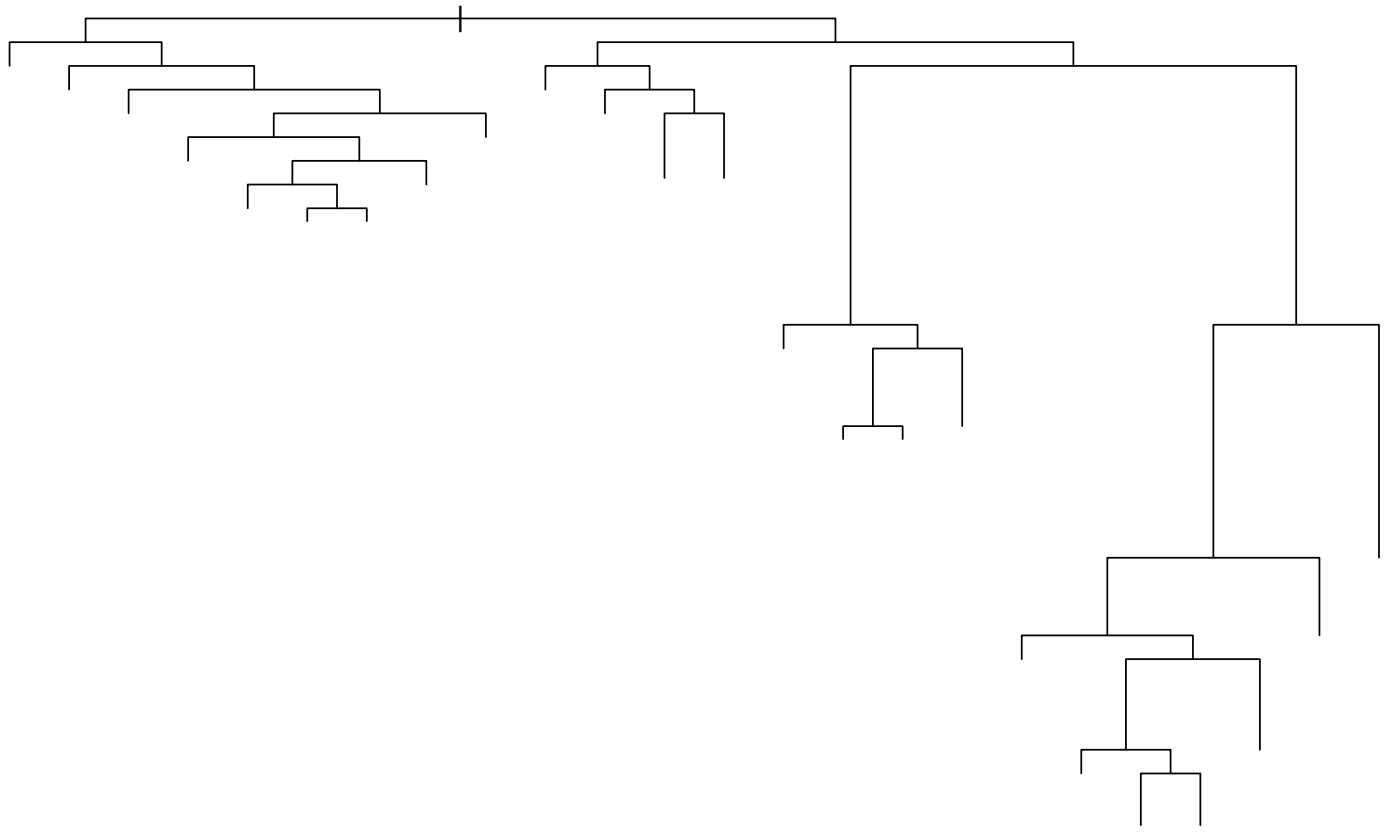
Example

```
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)
```

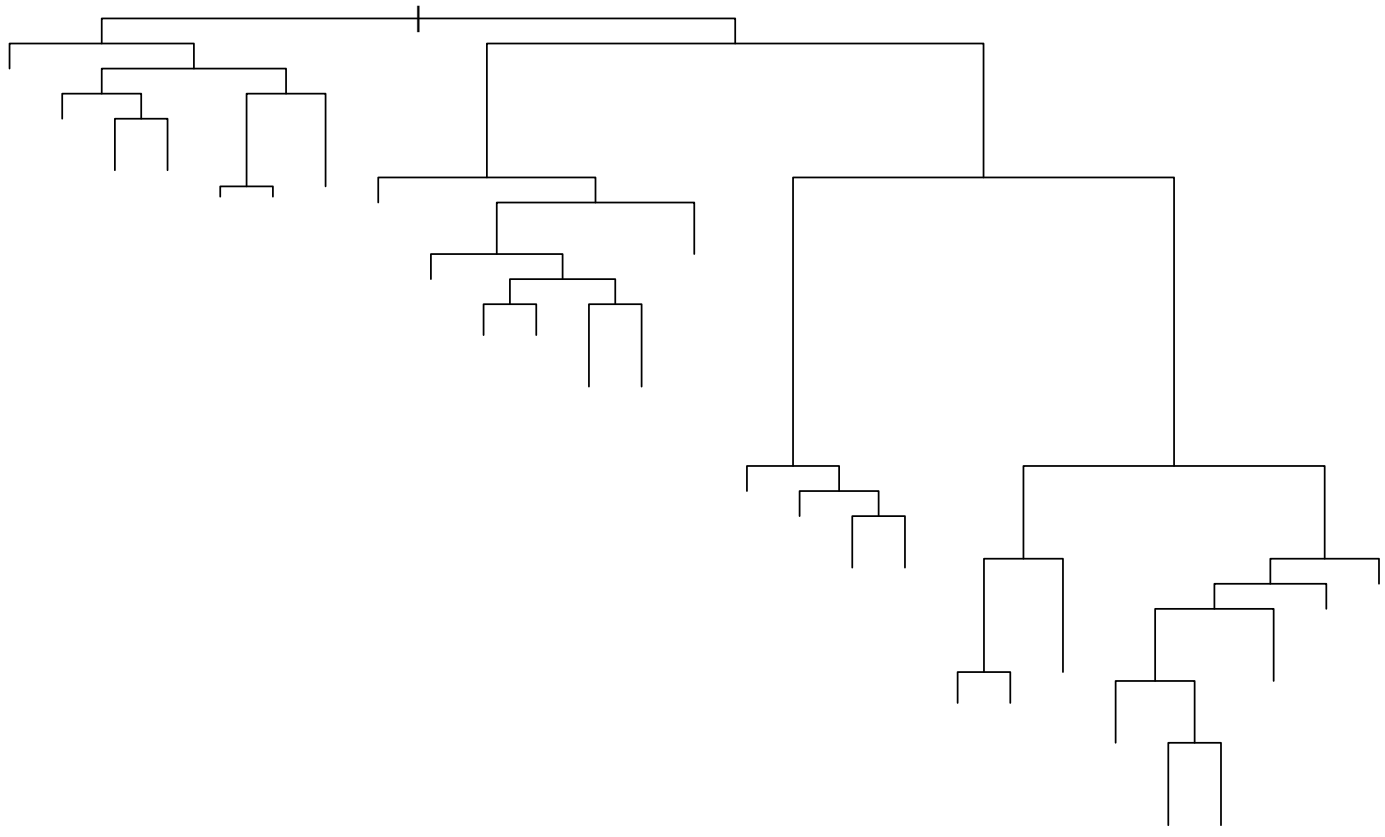
Example

```
plot(mod)
```



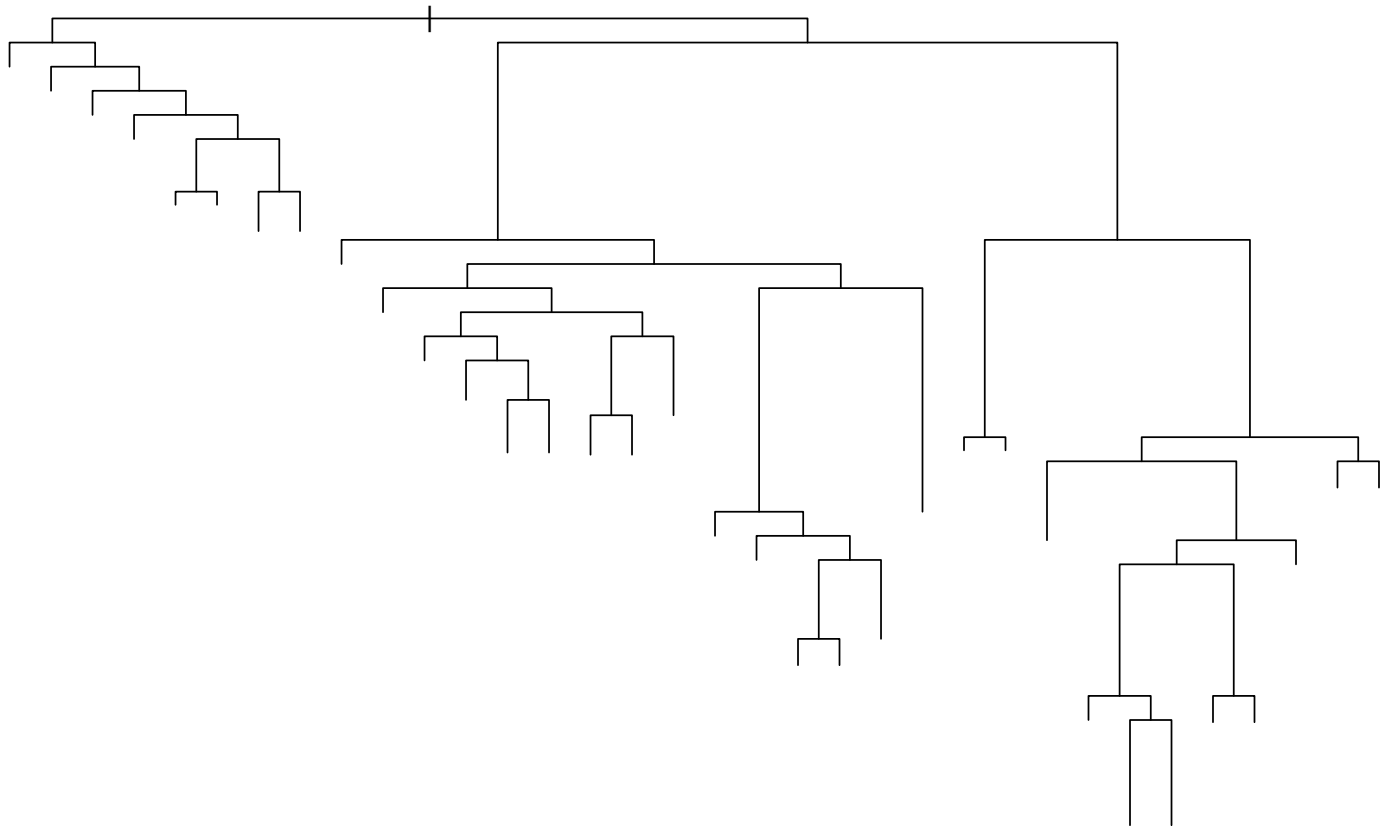
Example

```
plot(modb1)
```

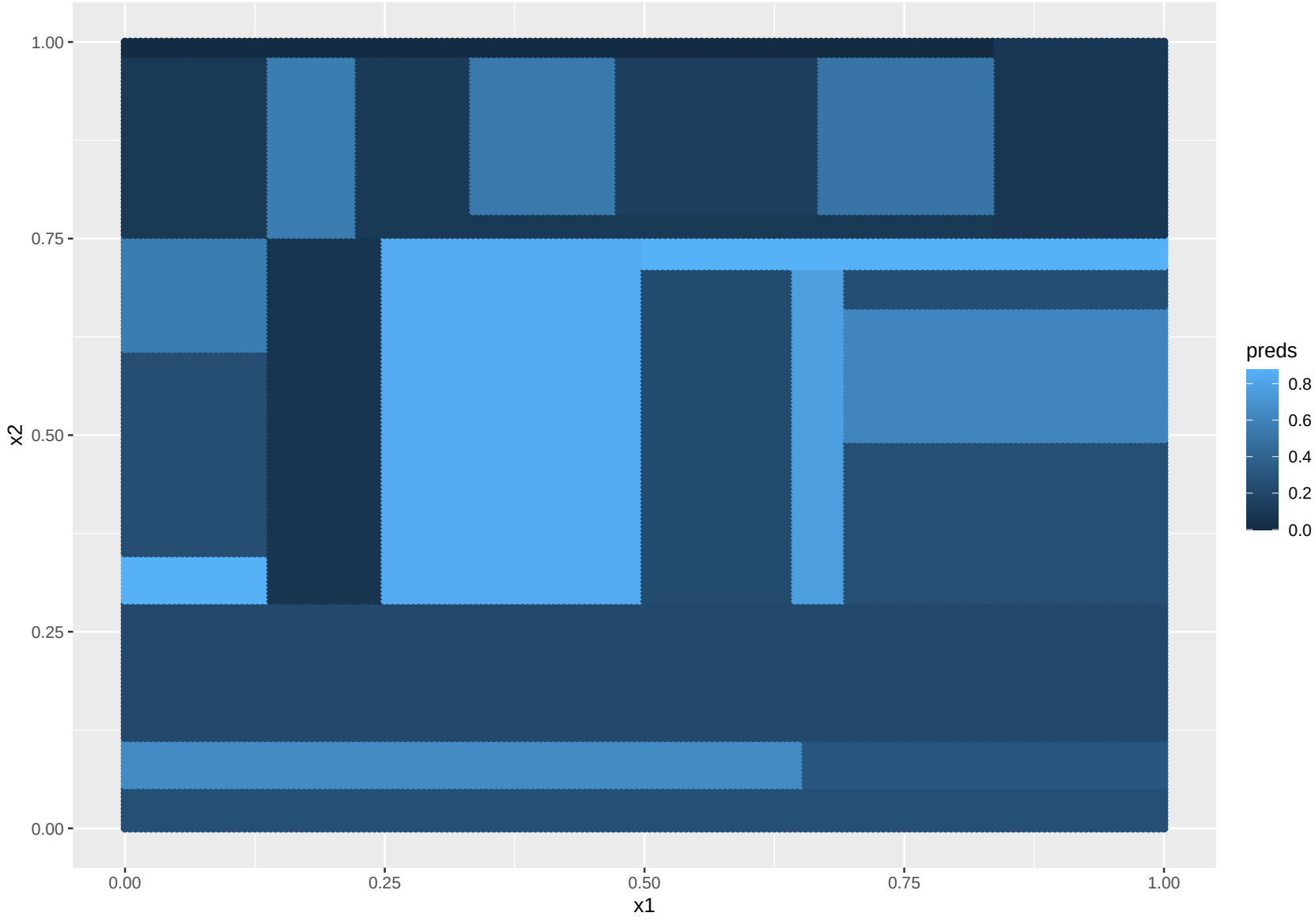


Example

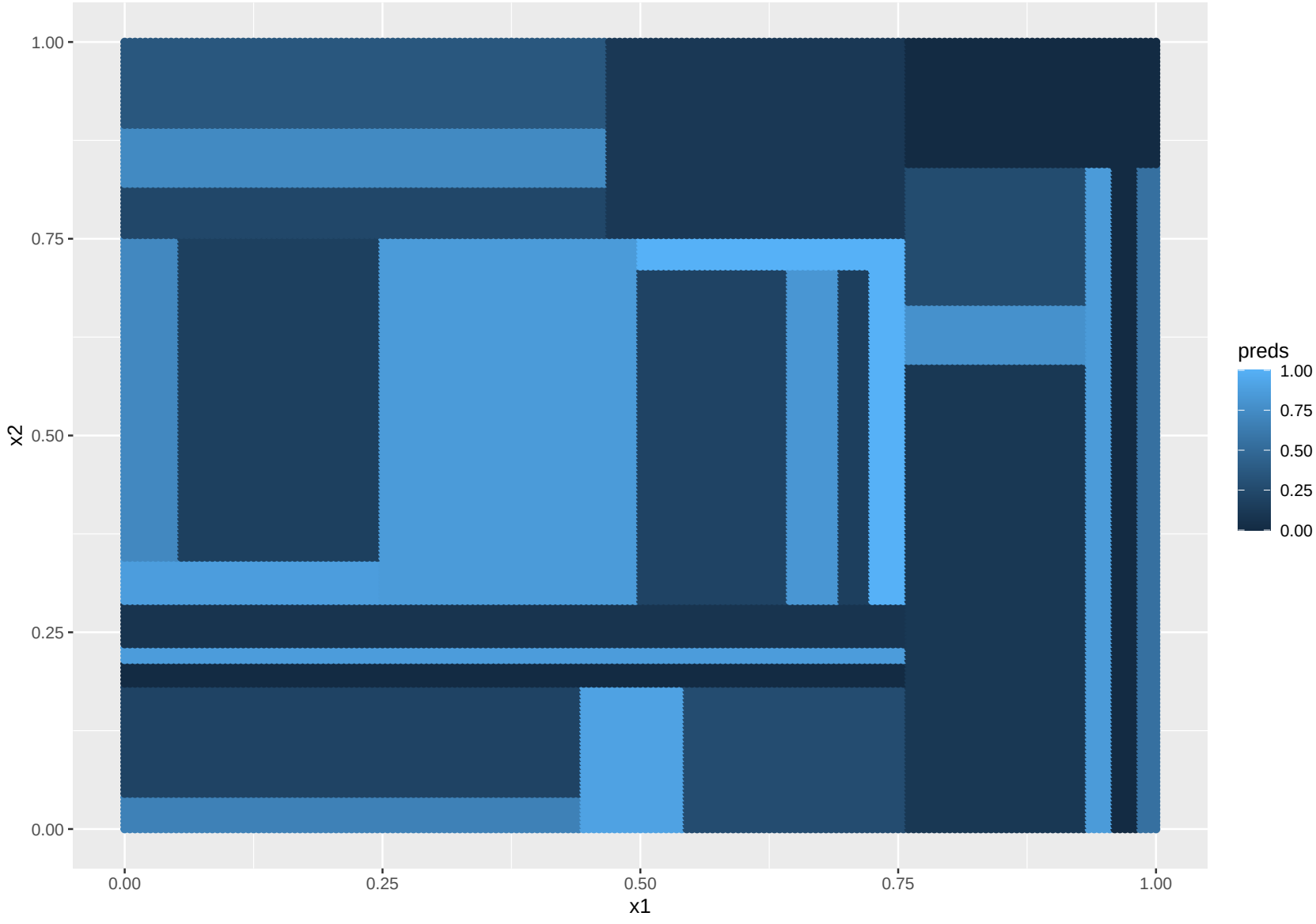
```
plot(modb7)
```



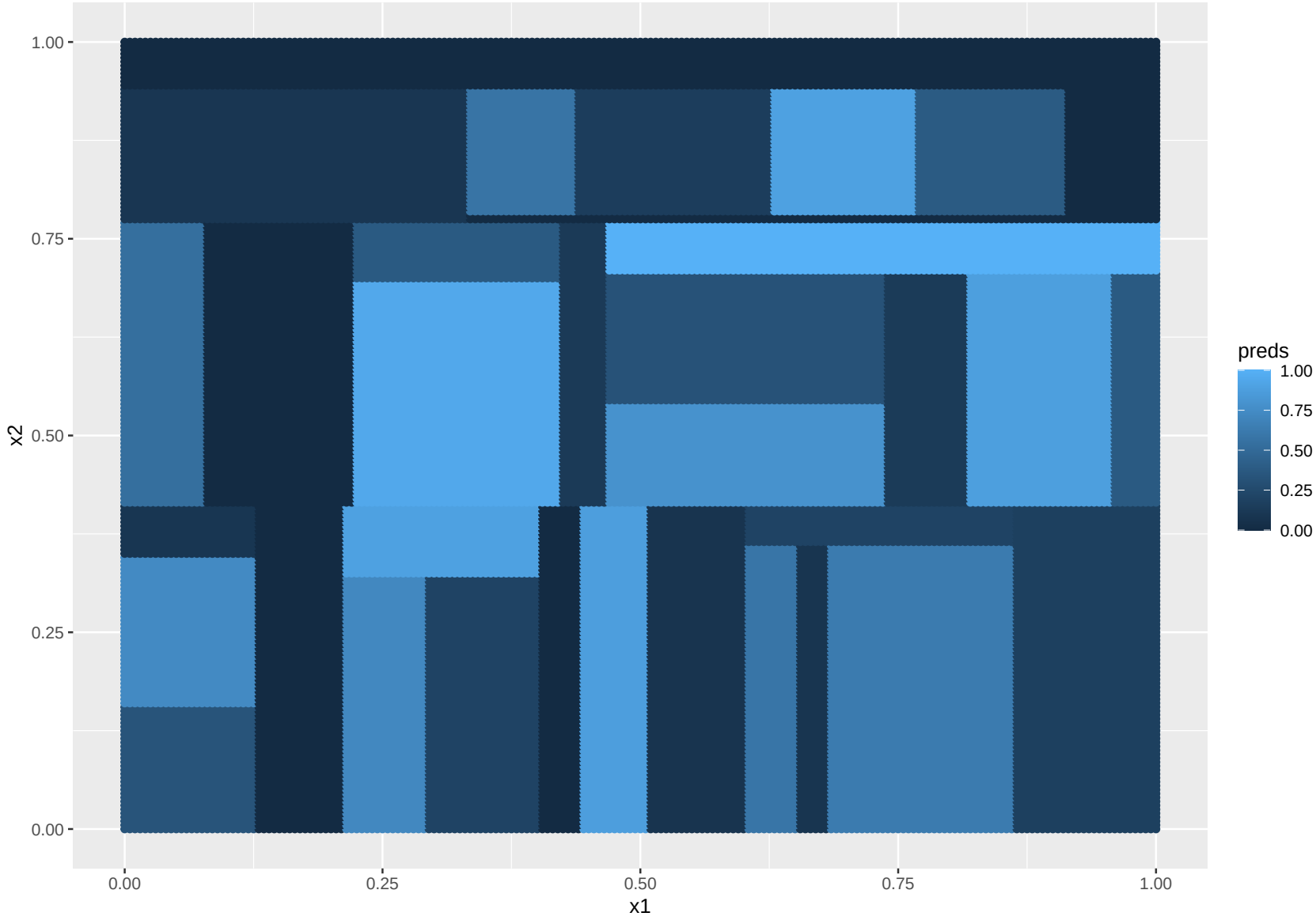
Example – Main



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 – B7

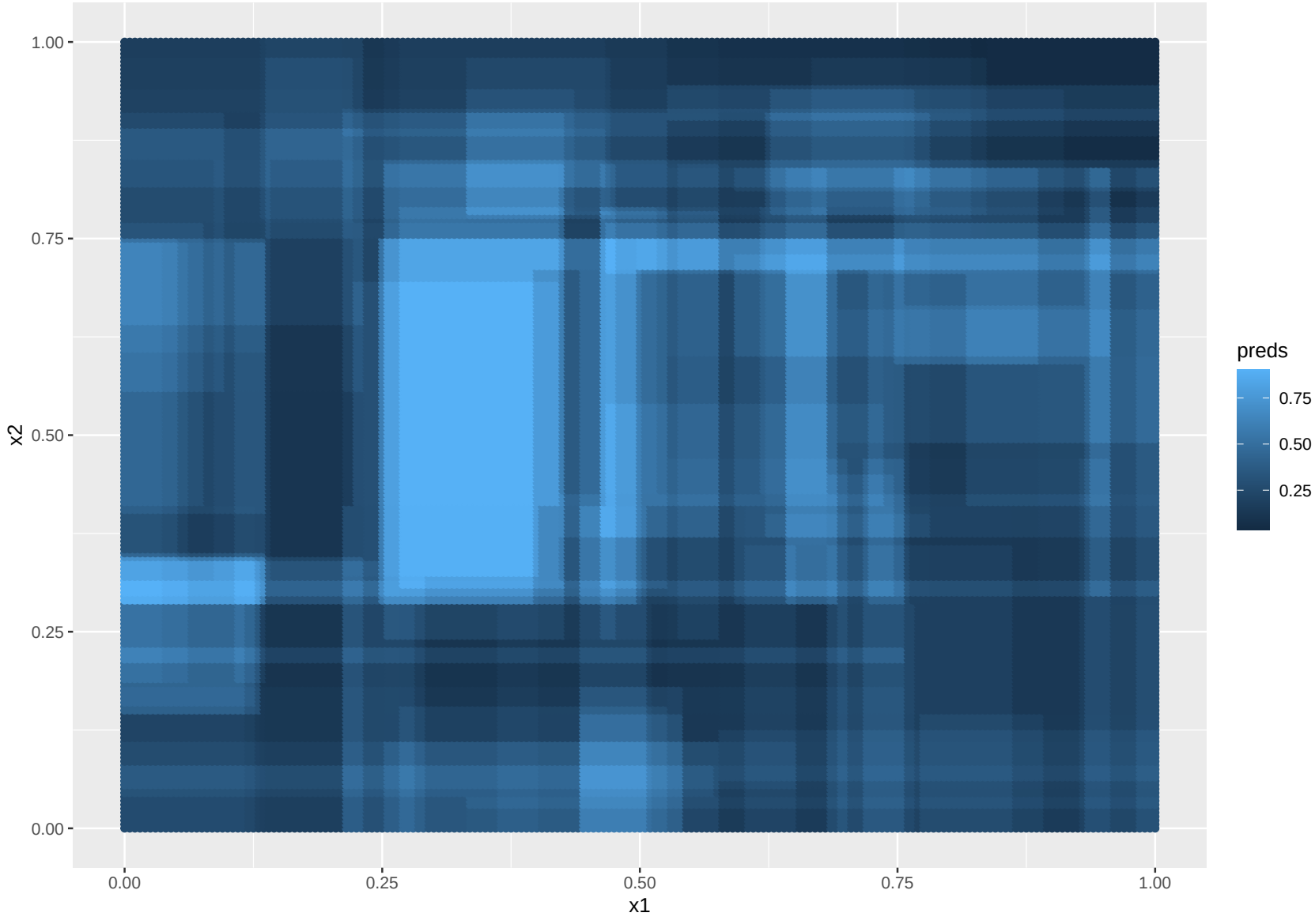

1.00
0.75
0.50
0.25
0.00
x2
0.00
0.25
0.50
0.75
1.00
x1
preds
1.00
0.75
0.50
0.25
0.00

Ensembles – The bare bones ensemble

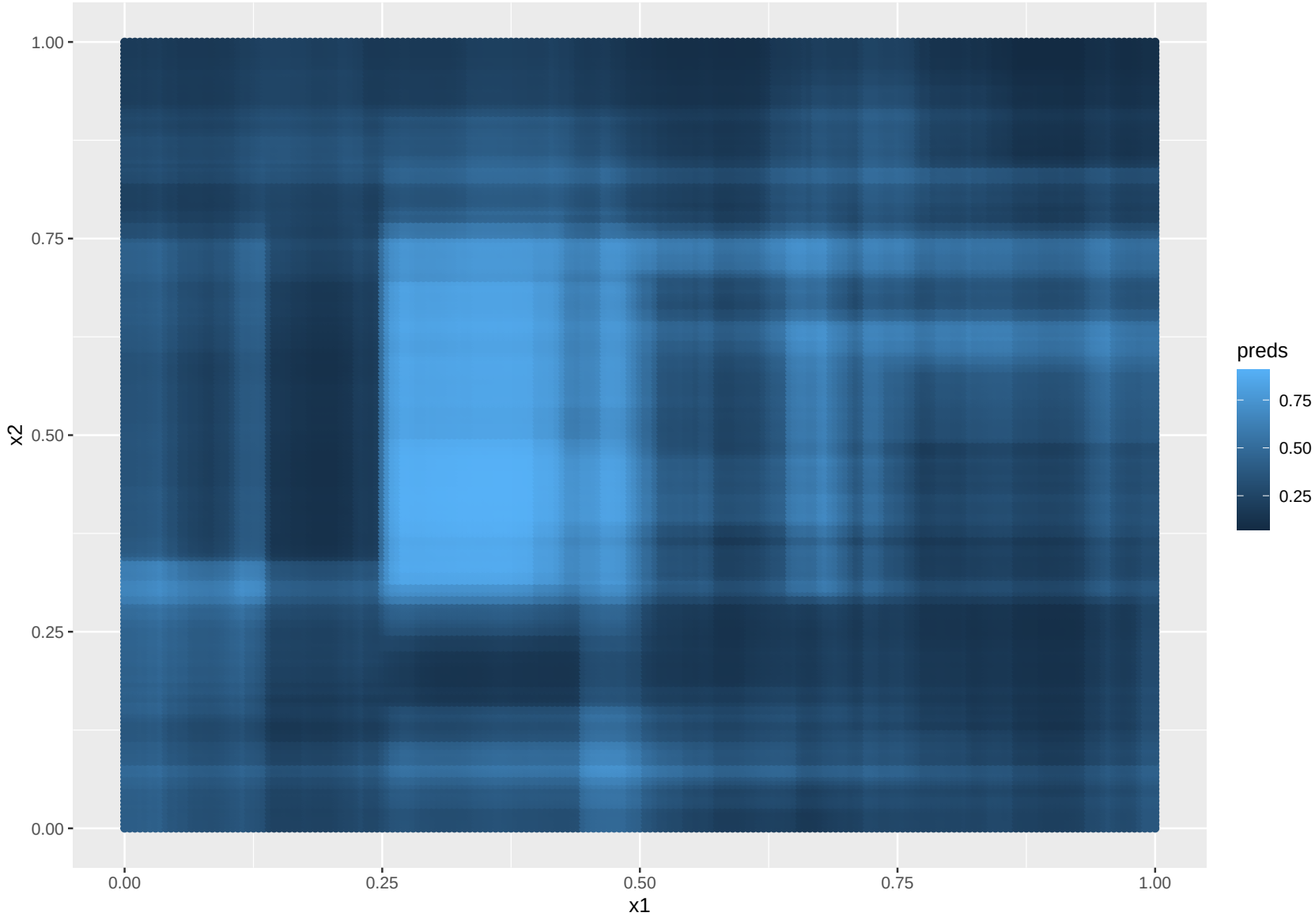
We can take an average across the models.

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

Averaged across 7 models



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.

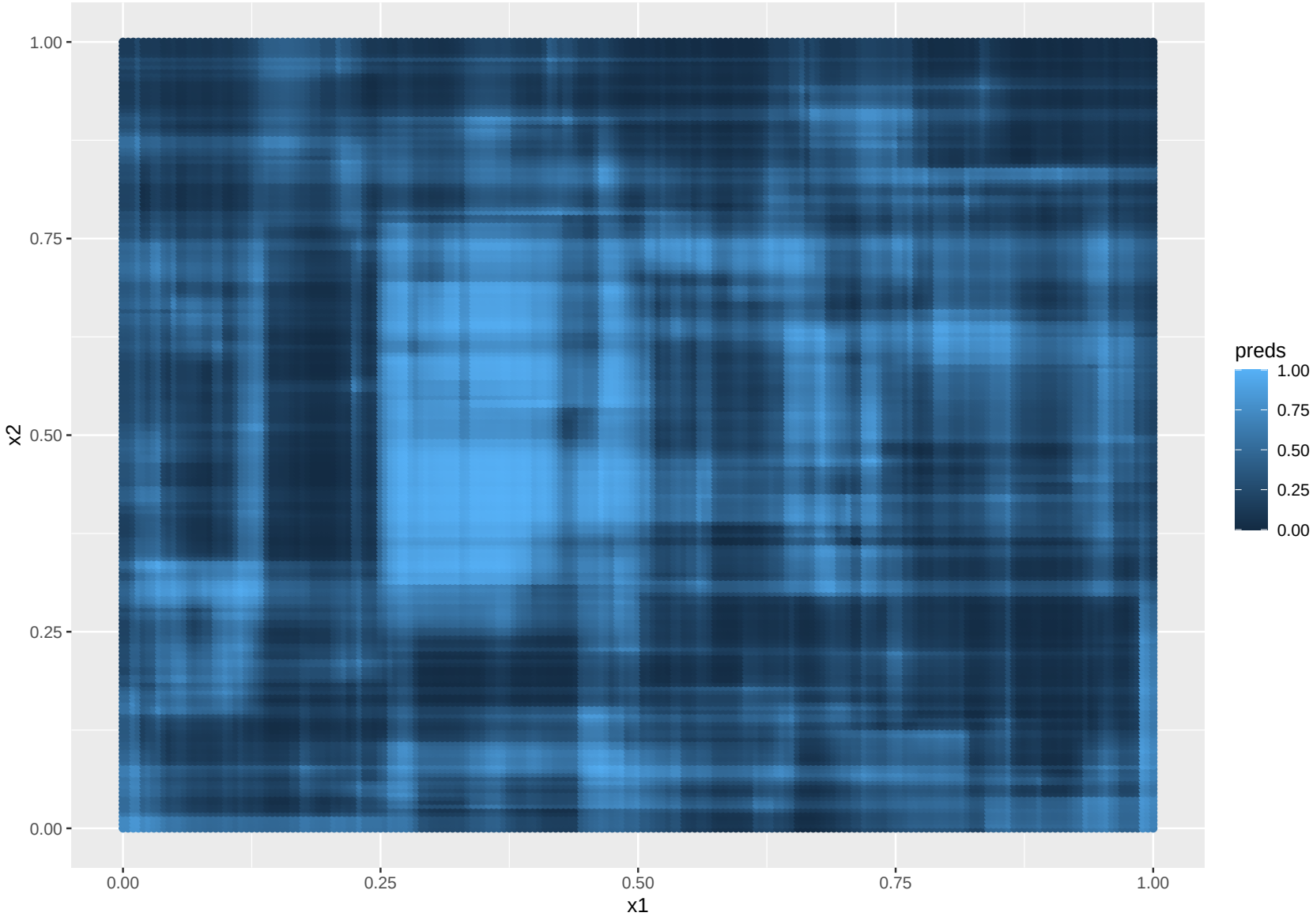
They search across a wider range of models.

Forests

```
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

Example Data: Zillow Zestimates

Zillow is a site where you can shop for homes. It shows what houses are on the market, it has pictures, it has a large number of facts about those houses.

Prominently, it makes a 'zestimate' – which is Zillow's prediction about the current price at which the home would sell.

In 2017-18 zillow had a competition seeking to improve its 'zestimates'. They published a lot of data on a number of properties (location, fireplaces, saunas, bathrooms, zip codes, etc), as well as a list of properties, sales (on different dates),

and the zestimates log error.

Log-error

Specifically:

$$logerror = log(zestimate) - log(price) = log(zestimate/price)$$

This is the log of the ratio of the prediction error from their model.

If you could forecast this well (out of sample), you could substantially improve their forecasts.

- We are building on top of their model. They have errors, we are now trying to model those errors.

This is useful

This was such an important thing to zillow, there was a $1million prize for the winners.

Random Forest

We can start.

```
holdout = sample(nrow(prop2na),0.2*nrow(prop2na),replace=F)
zilmod = ranger(logerror~.,data=prop2na[-holdout,])
preds = predict(zilmod,data=prop2na[holdout,])
oob.errs = preds$predictions-prop2na$logerror[holdout]
mean(oob.errs^2)          # 0.02204
mean(prop2na$logerror^2) # 0.02608
```

This is an out-of-sample R^2 of ~15% – ON THEIR ERRORS. In not many lines of code.

(NB I don't include the data or code here – we're going to have fun in a week cleaning it – There is a reason `NA` is in the df name.)

Predicting Errors

Predicting Errors is a tough problem. We know they should be mean 0 (when predicting means). We also know that most of the variation has already been soaked up. What we are looking at – is the residual.

If the base model is any good, this is mostly noise. Getting an out of sample R^2 above 0 is *hard*. If the base model is near perfect, it becomes impossible.

Predicting Errors

But... it is so valuable. If we can predict the errors, we can just improve our baseline predictions by making them, predicting the errors, and then adjusting our predictions appropriately.

If we are making "predictable" mistakes with our predictions, this is super useful.

Boosting

This is the essence of boosting. Instead of building up a model by averaging across a bunch of model predictions for a bunch of trees, what if we build a simple tree, then we try to predict its errors with another tree, and so forth.

Boosting

I suspect I'll run out of time here. So I'll say this:

There are many options for boosting. And it can be *extremely* effective.

The basic notion though "we want to predict our errors" is not complicated.

Wrap up

Things to do

Predictions. No homework until Thursday.

See you Thursday.

Bye!