More ROC & Trees

Lecture 9

Connor Dowd

April 27th, 2021

Today's Class

1. ROC in depth
2. Classification Trees

Quick Review

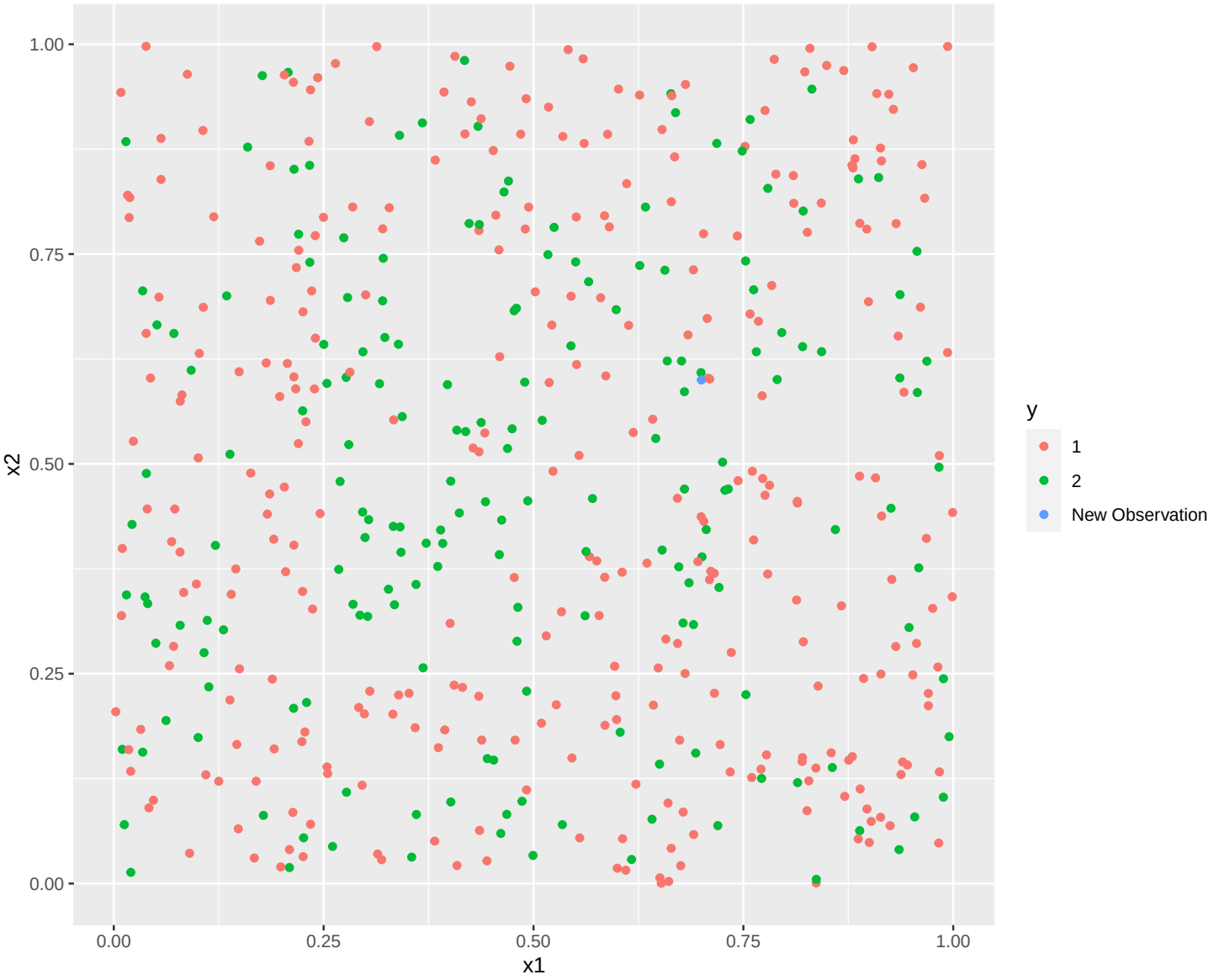
KNN Basics

Basic Idea: Estimate $P[y|x]$ locally using the labels of similar observations in the training data.

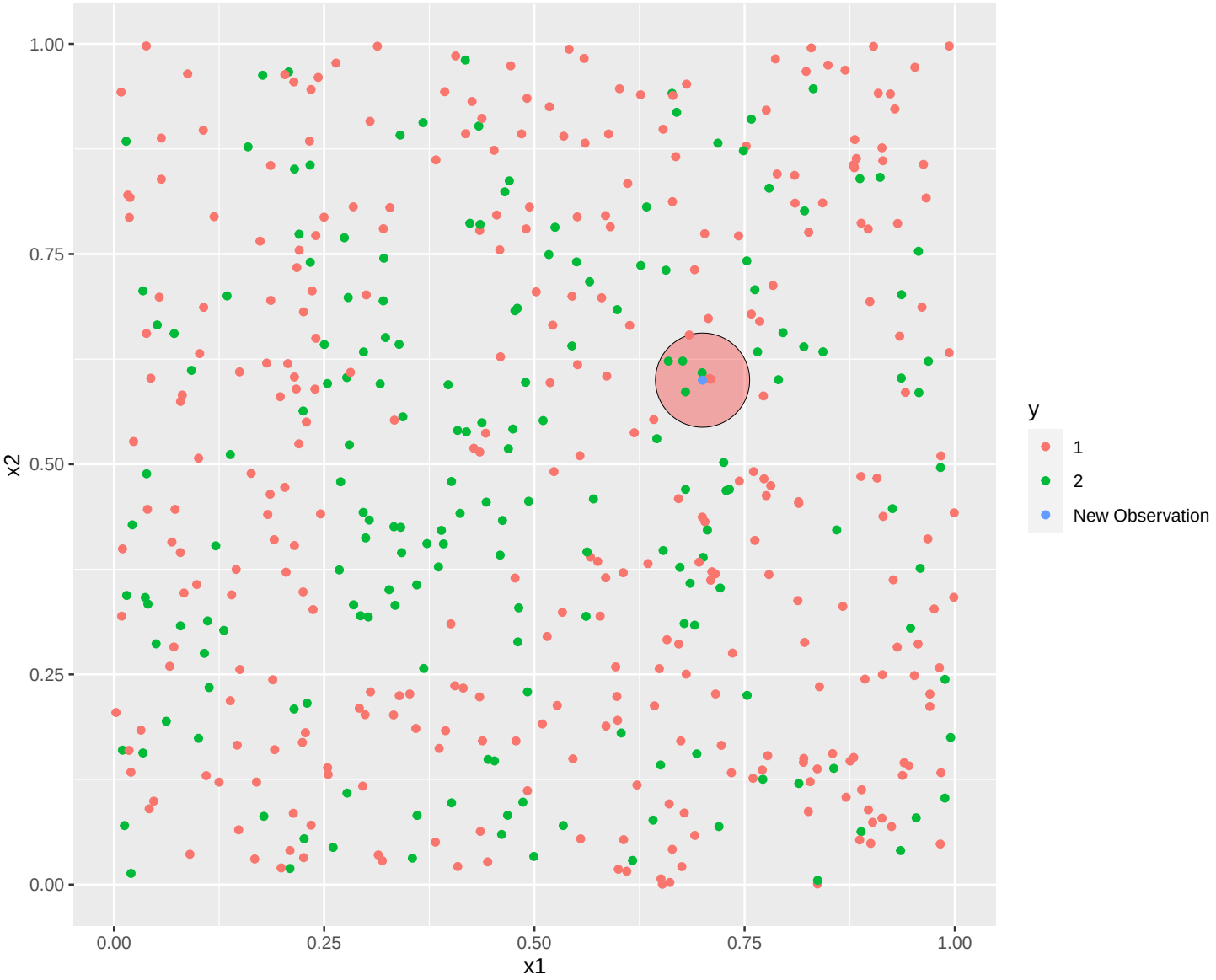
KNN: What is the most common class near x^{new}?

1. Take the K nearest neighbors $x_{i,1}, ..., x_{i,K}$ of x^{new} in the training data
   - Nearness is (usually) Euclidean distance: $\sqrt{\sum_{j=1}^{p}(x_j^{new} - x_{i,k,j})^2}$
2. Estimate $P[y = j|x] = \frac{1}{n}\sum_{i=1}^{K} 1(y_i = j)$
3. Select the class j with the highest probability.

KNN Example Data



KNN Example $K = 7$



The relative 'vote counts' are a very crude estimate of probability.

KNN Example

Function that finds the most common outcome among nearest neighbors

```
#data in "df", "x1", "x2", "y"
knn_prob = function(x,k=5) {
  dists = sqrt((x1-x[1])^2+(x2-x[2])^2) #Find all distance
  bound = sort(dists)[k]                #Find kth smallest
  indices = which(dists <= bound)       #Find which obs ha
  outcomes = as.integer(y[indices])     #Find correspondin
  mean(outcomes)-1                      #Taking advantage
}
```

Grid out

Build a grid of points.

```
grid.fineness = 81
sequence = seq(0,1,length.out=grid.fineness)
grid = expand.grid(sequence,sequence)
colnames(grid) = c("x1","x2")
```

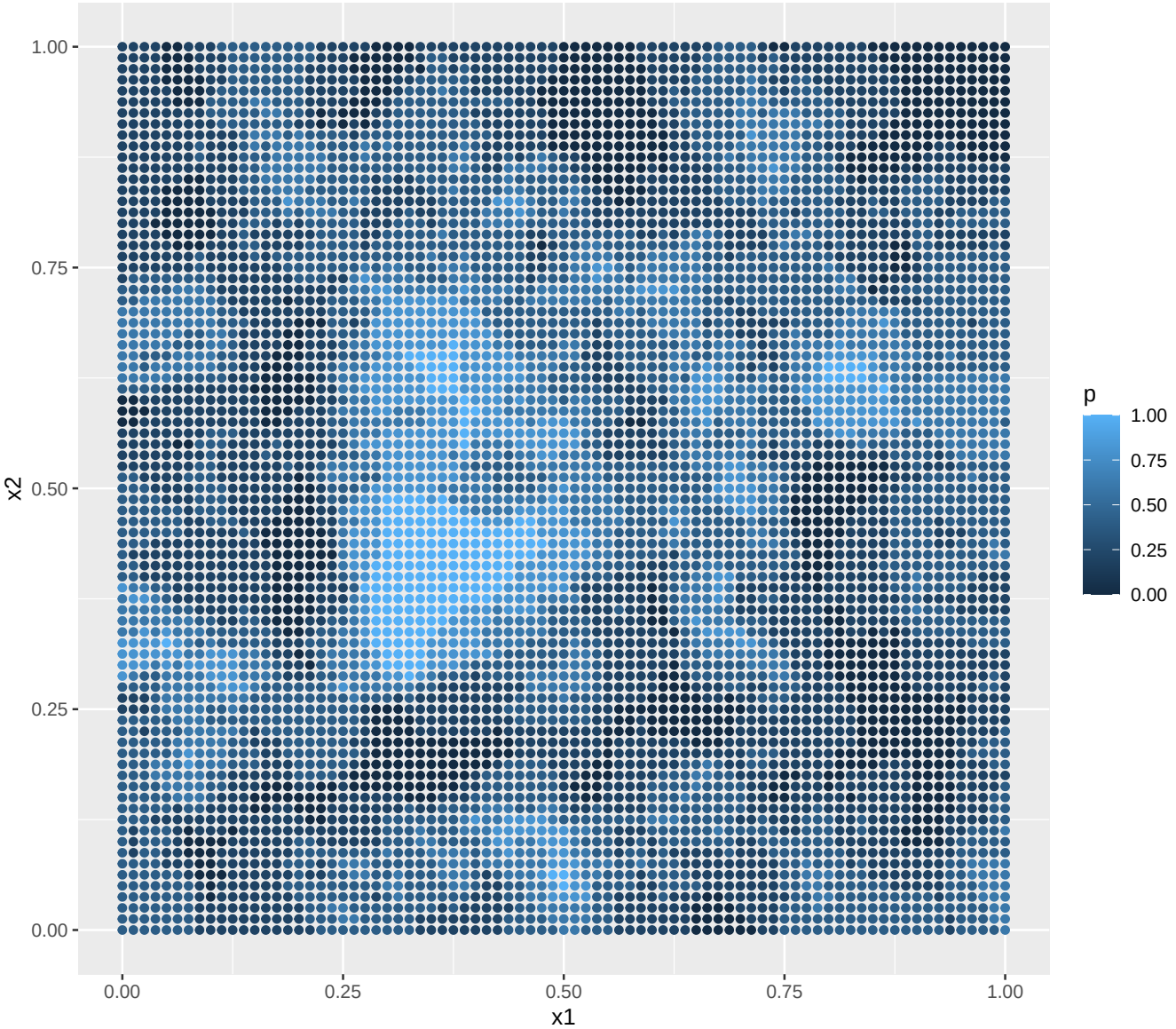
Make Predictions

Make a prediction at each point *in the grid*.

```
phat = apply(grid,1,knn_prob)
yhat = as.factor((phat >= 0.5)+1)
df_grid = data.frame(x1=grid$x1,
                     x2=grid$x2,
                     y=yhat,
                     p=phat)
```

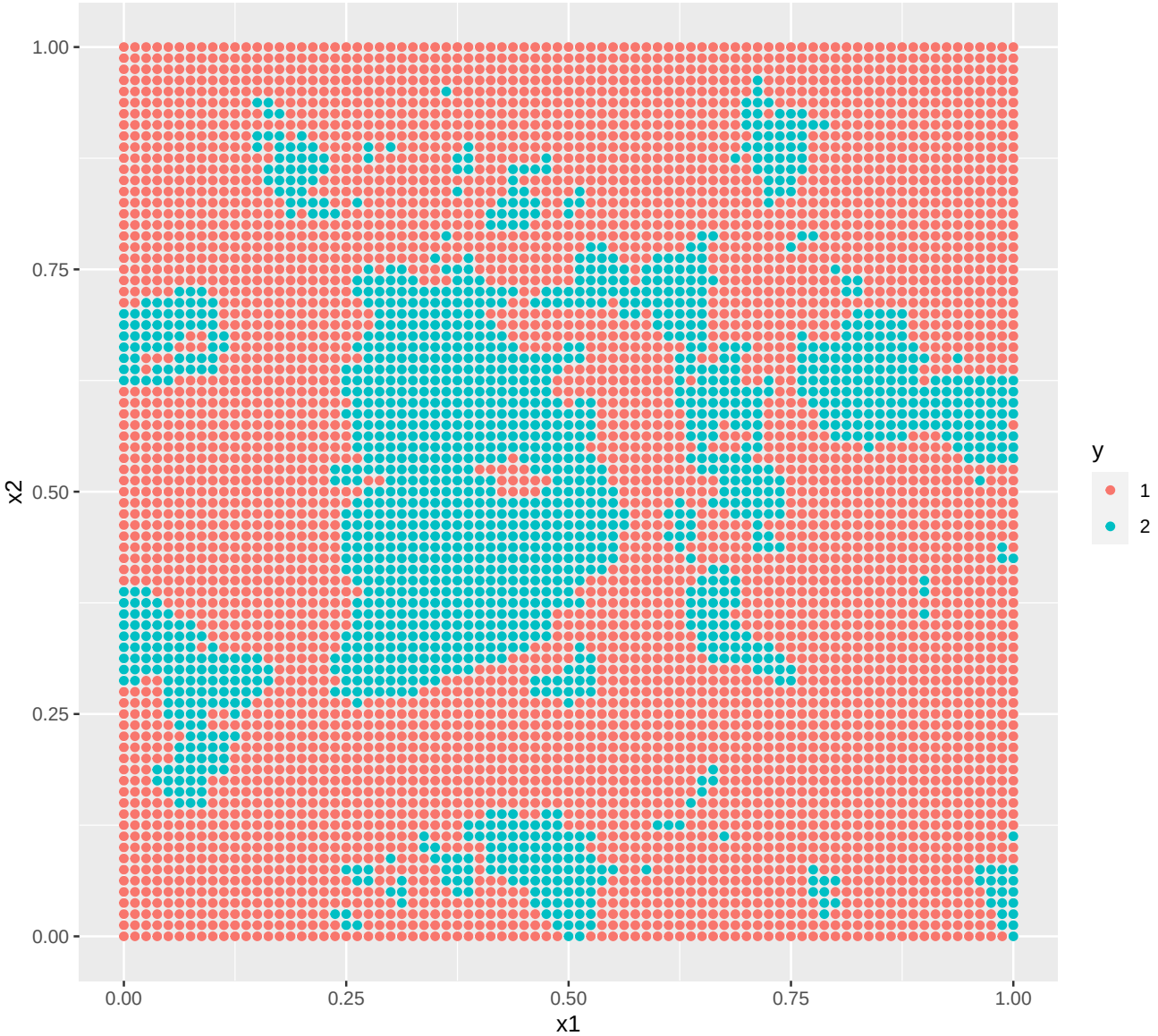
Plots – Probabilities

```
ggplot(df_grid,aes(x=x1,x2,col=p))+geom_point()+theme(aspec
```



Plots – Predictions

```
ggplot(df_grid,aes(x=x1,y=x2,col=y))+geom_point()+theme(asp
```



Probalities $\rightarrow$ Predictions

How did we jump from probabilities to predictions?

```
yhat = as.factor((phat >= 0.5)+1)
```

This is a threshold of 50%. When $P[Y == 2] >= 0.5$, we predict $Y = 2$, otherwise we predict $Y = 1$.

Different Thresholds may be preferable

Differential costs mean we may want to choose a threshold that isn't 0.5. The 50% threshold minimizes our misclassification risk, but not necessarily the costs we face.

This is where the ROC curve comes in.

Thresholds – 0

```
##     yhat
## y      1   2
##   1    0 318
##   2    0 182
```

Therefore: All 2s correctly predicted. All 1s incorrectly predicted. (Sensitivity & Specificity respectively)

Thresholds – 0.2

```
##     yhat
## y      1   2
##   1   63 255
##   2   13 169
```

Most 2s correct, most 1s incorrect.

Thresholds – 0.6

```
##    yhat
## y     1   2
##   1 311   7
##   2 149  33
```

Most 2s incorrect, most 1s correct.

Thresholds – 1

```
##     yhat
## y      1   2
##   1  318   0
##   2  182   0
```

All 2s incorrect. All 1s correct.

Sensitivity and Specificity

- Sensitivity: proportion of true $Y = 1$ classified as such.
- Specificity: proportion of true $Y = 0$ classified as such.

A rule is sensitive if it mostly gets the 1s right. A rule is specific if it mostly gets the 0s right.

(We will redefine Y=y-1 – so that sensitivity is "proportion of true 2s classified as such" and specificity is "proporition of true 1s classified as such"). Ultimately, just about false positives vs false negatives again.

ROC Curve

The ROC curve says,

"We face a tradeoff between getting one type of error and the other, let us look at this tradeoff for different threshold values – within a single model".

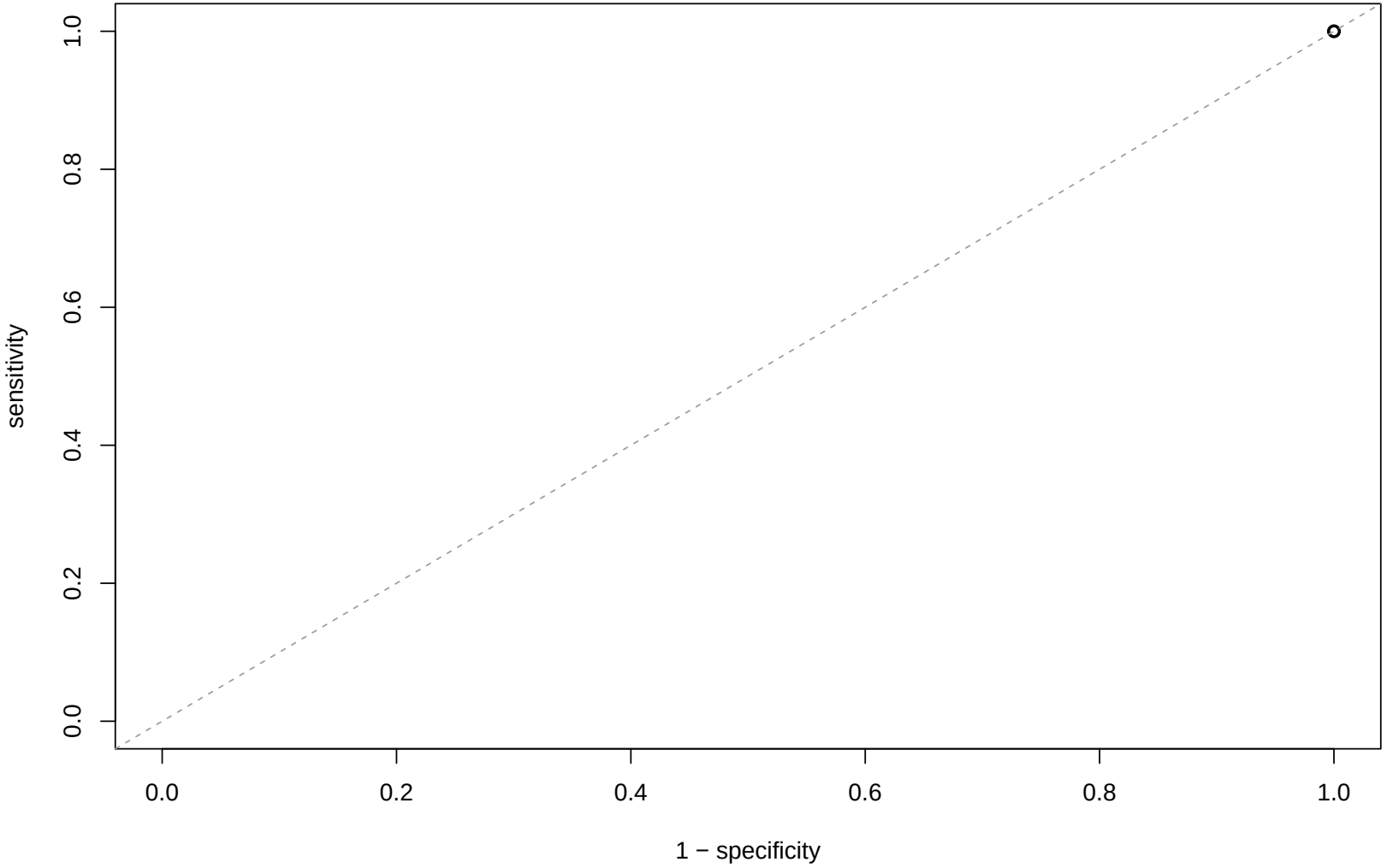
ROC Curve

Comparing Sensitivity and Specificity for Different Classification Thresholds.

```
roc = function(p,y, ...){
  y = factor(y)
  n = length(p)
  p = as.vector(p)
  probs = seq(0,1,length=1001)
  mat = matrix(rep(probs,n),ncol=length(probs),byrow=TRUE)
  Q = p > mat
  specificity = colMeans(!Q[y==levels(y)[1],])
  sensitivity = colMeans(Q[y==levels(y)[2],])
  plot(1-specificity, sensitivity, type="l", ...)
  abline(a=0,b=1,lty=2,col=8)
}
```

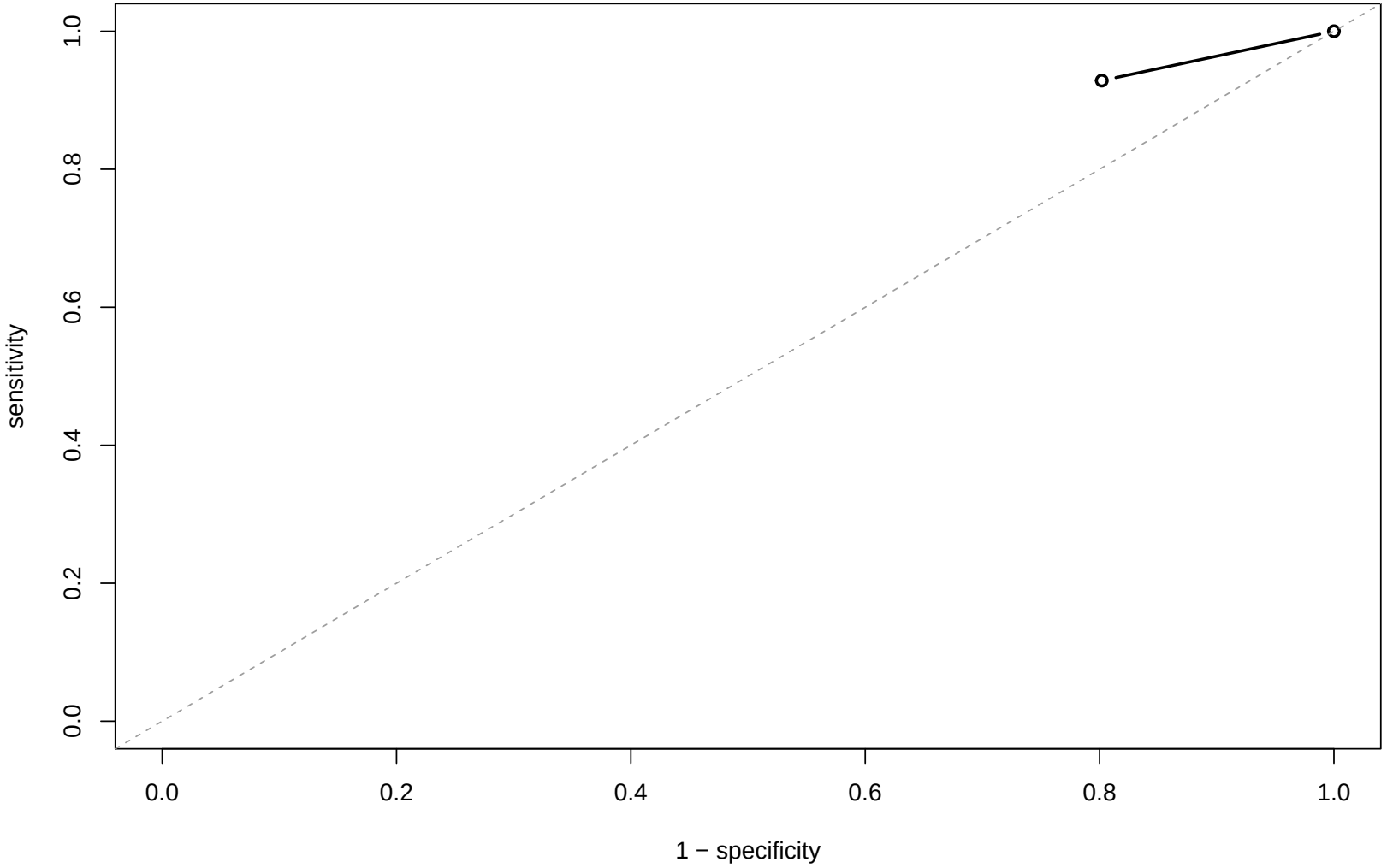
ROC Curve Broken Down – Threshold 0

Pick a threshold. Calculate Sensitivity and 1-Specificity at that threshold. Plot that point. Rinse and Repeat.



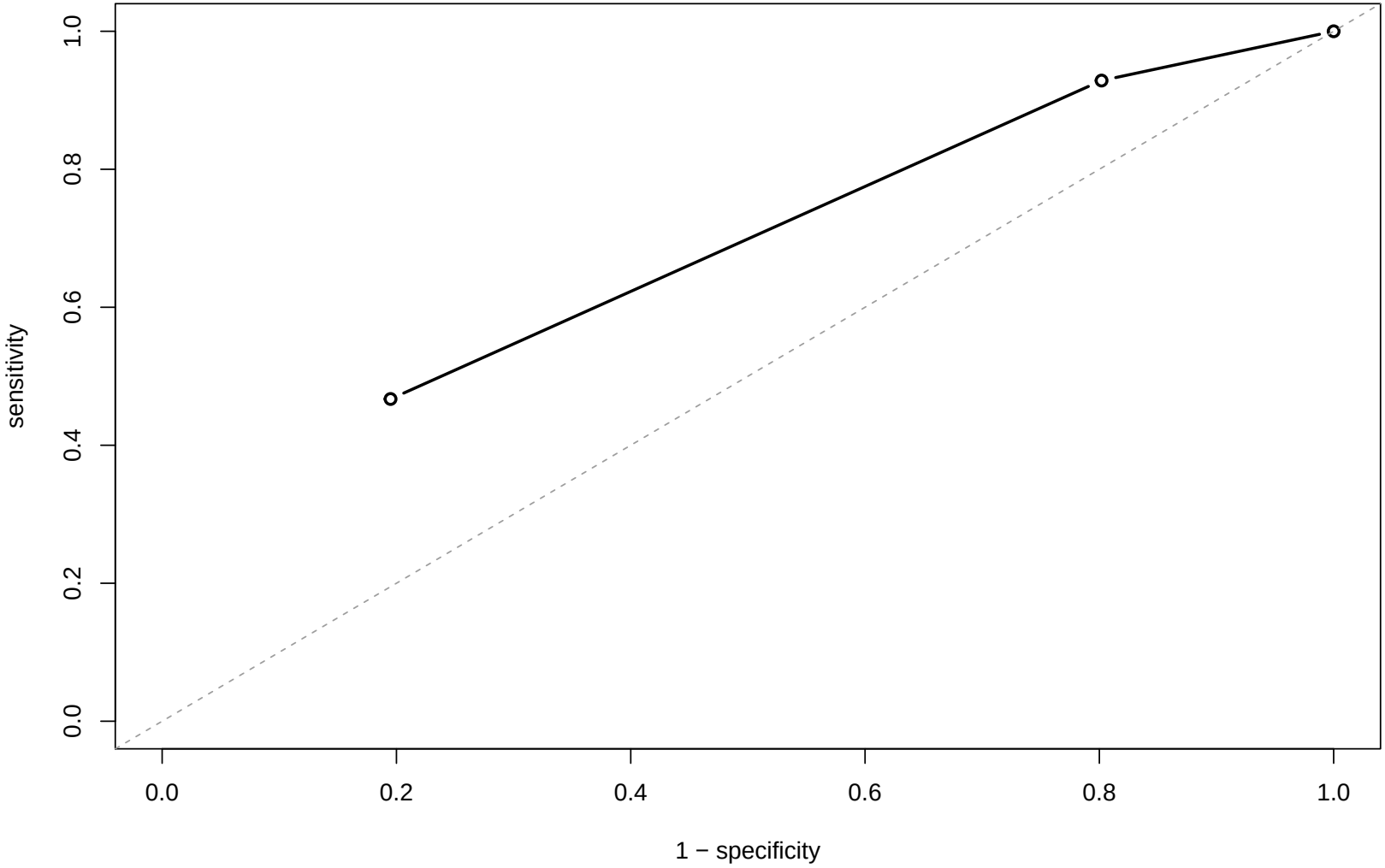
ROC Curve Broken Down – Threshold 0.2

Pick a threshold. Calculate Sensitivity and 1-Specificity at that threshold. Plot that point. Rinse and Repeat.



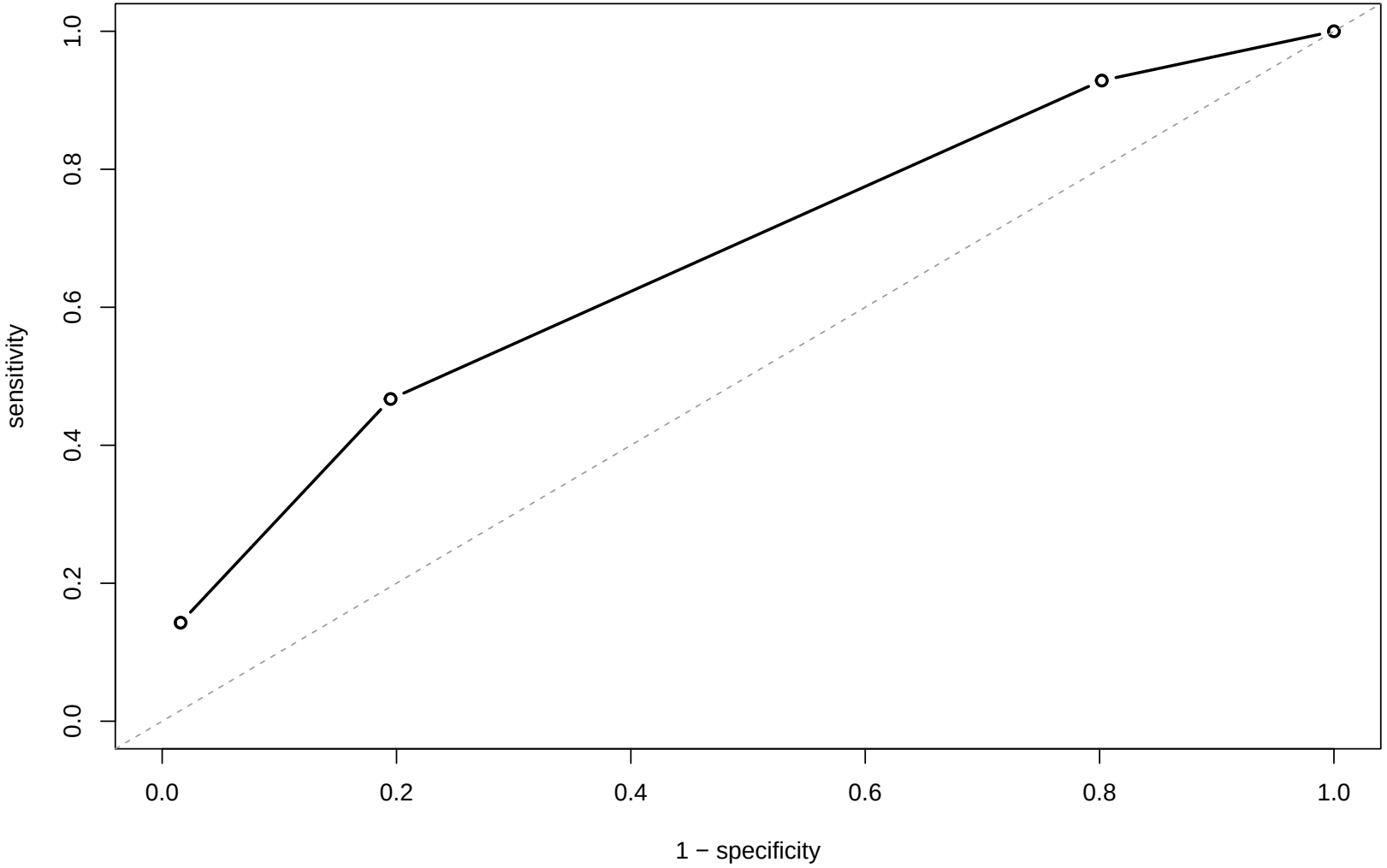
ROC Curve Broken Down – Threshold 0.4

Pick a threshold. Calculate Sensitivity and 1-Specificity at that threshold. Plot that point. Rinse and Repeat.



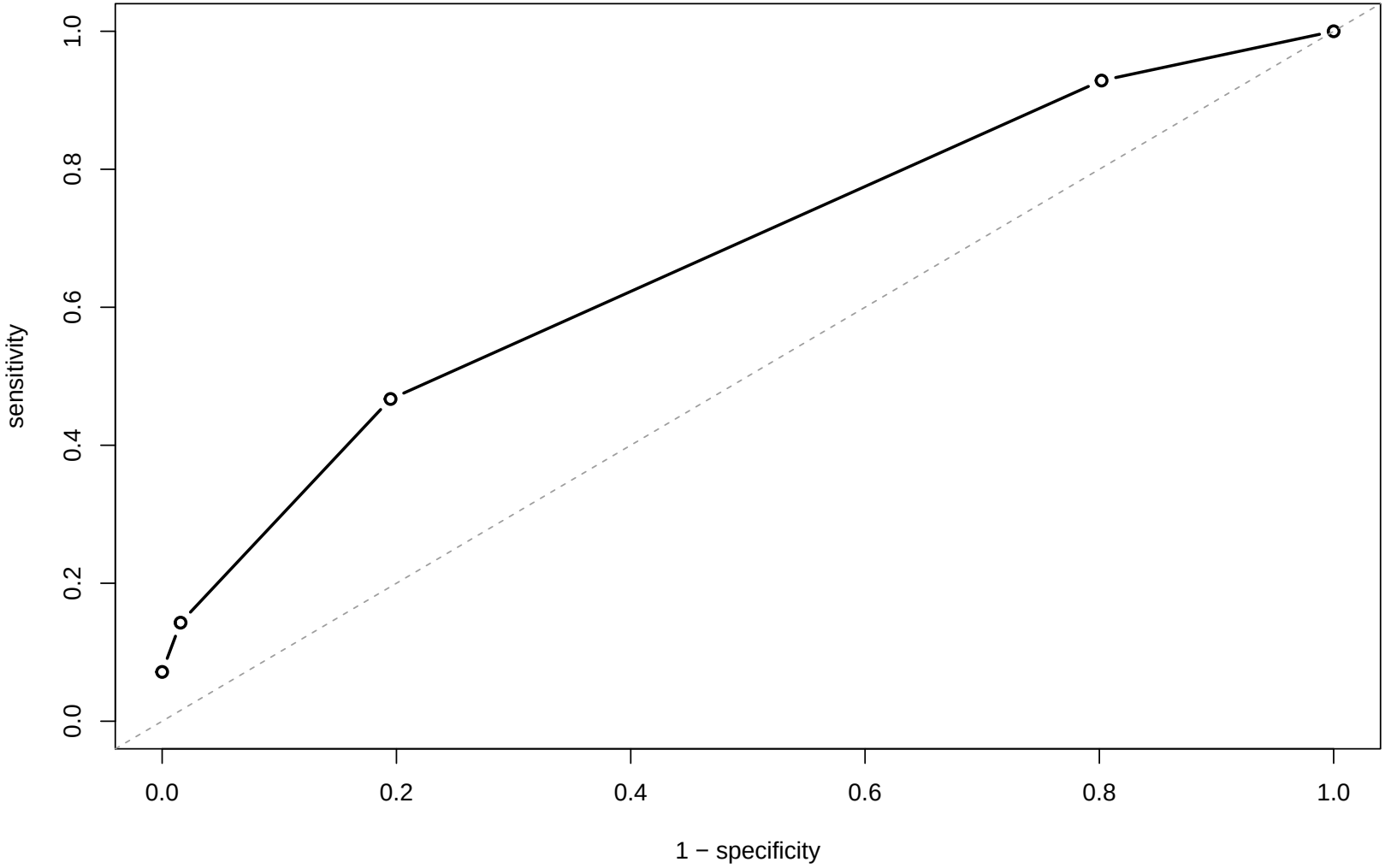
ROC Curve Broken Down – Threshold 0.6

Pick a threshold. Calculate Sensitivity and 1-Specificity at that threshold. Plot that point. Rinse and Repeat.



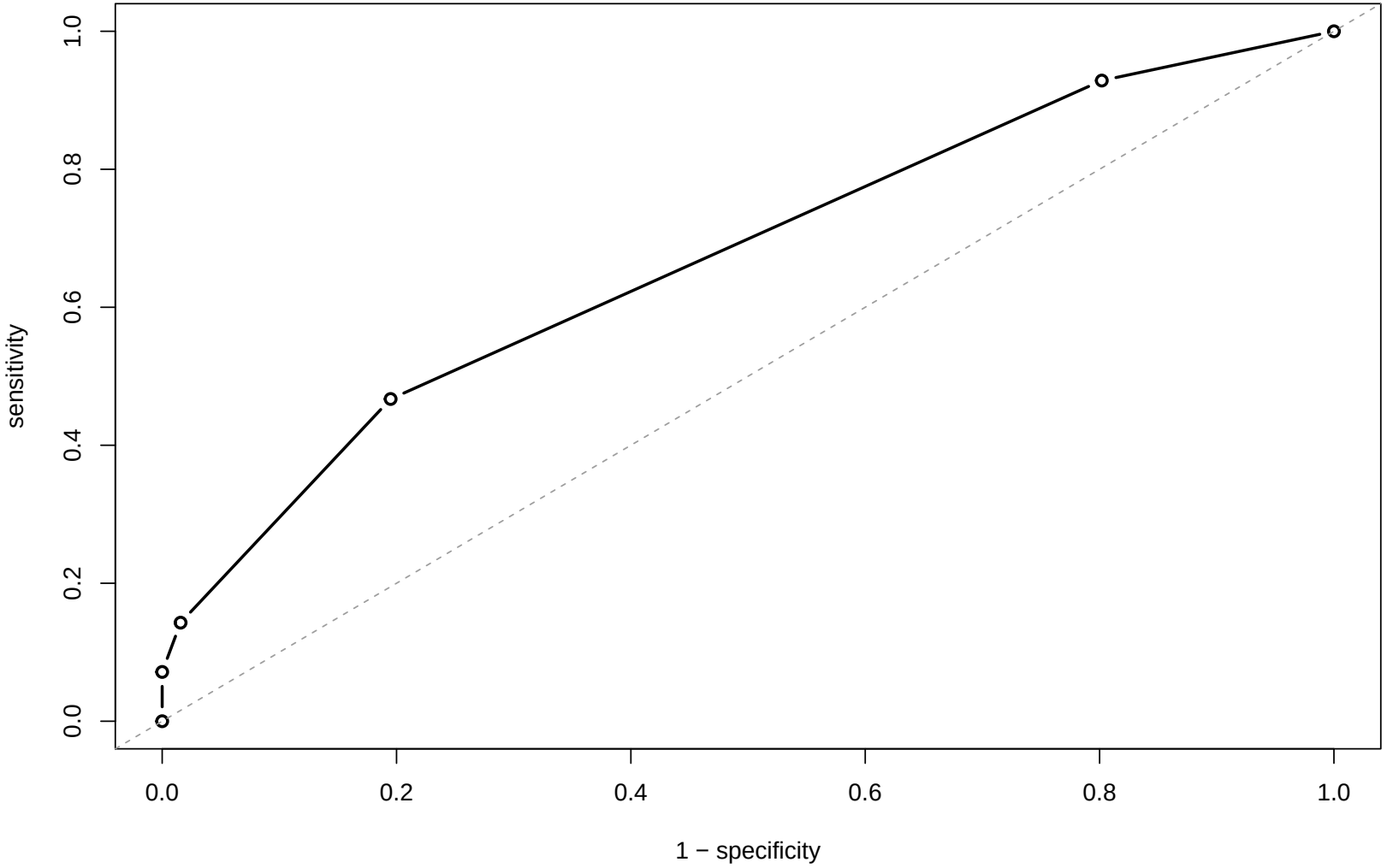
ROC Curve Broken Down – Threshold 0.8

Pick a threshold. Calculate Sensitivity and 1-Specificity at that threshold. Plot that point. Rinse and Repeat.



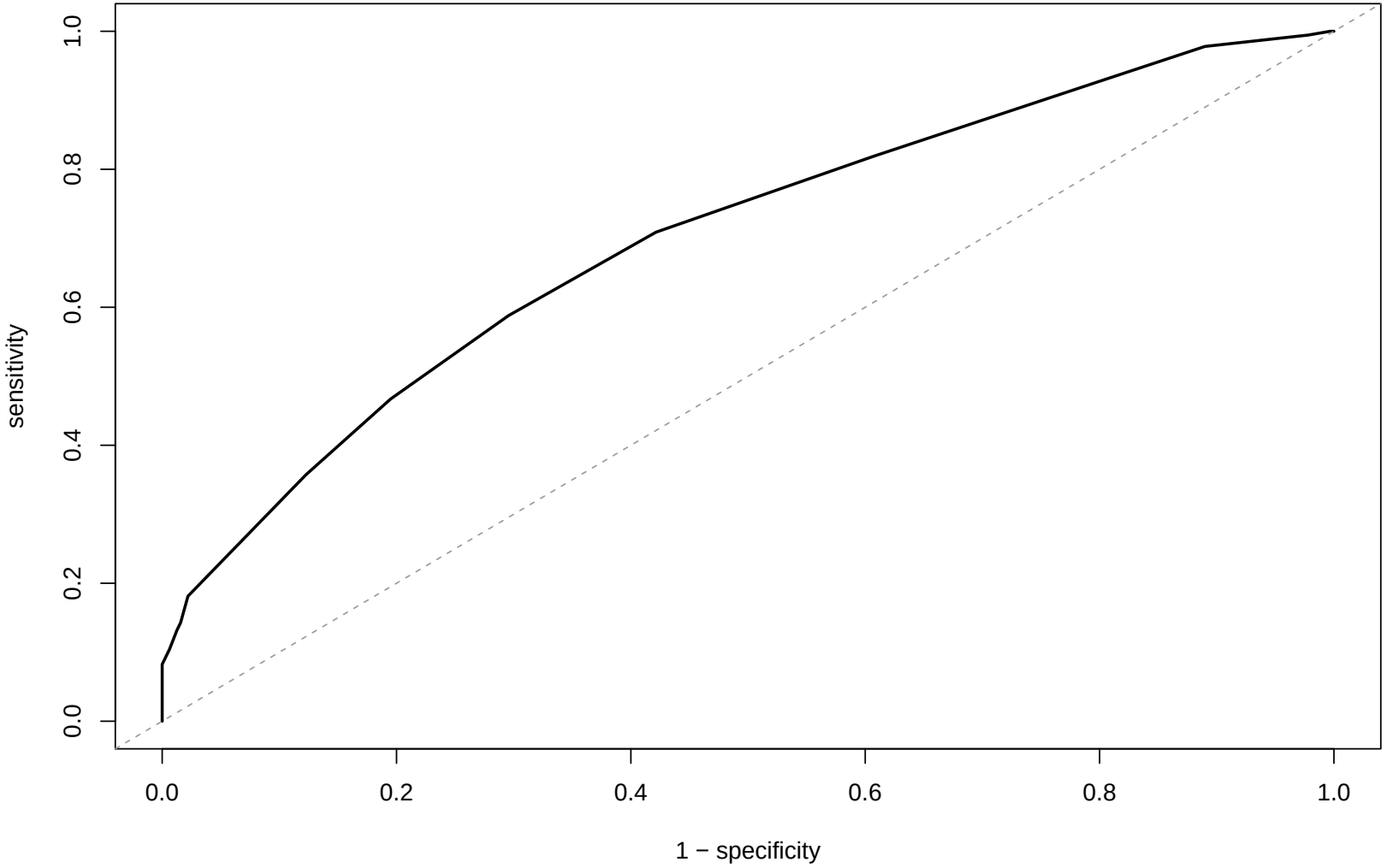
ROC Curve Broken Down – Threshold 1

Pick a threshold. Calculate Sensitivity and 1-Specificity at that threshold. Plot that point. Rinse and Repeat.



ROC Curve Broken Down – Finer Sequence

Pick a threshold. Calculate Sensitivity and 1-Specificity at that threshold. Plot that point. Rinse and Repeat.



ROC Curves First Recap

- Useful for understanding predictive power of different models.
- We've only looked at *in-sample* ROC curve
- Area under the curve is a simplifying metric
- Slope of ROC curve is also meaningful

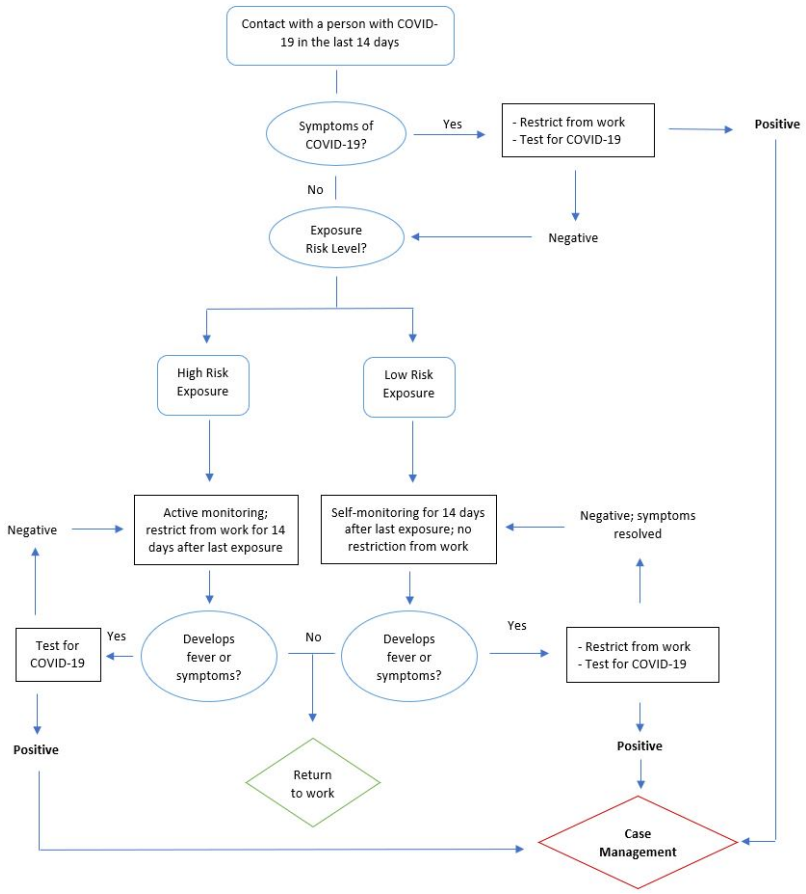
Some questions remain:

- Models with Comparable AUC, but different curves: How to choose?

Trees

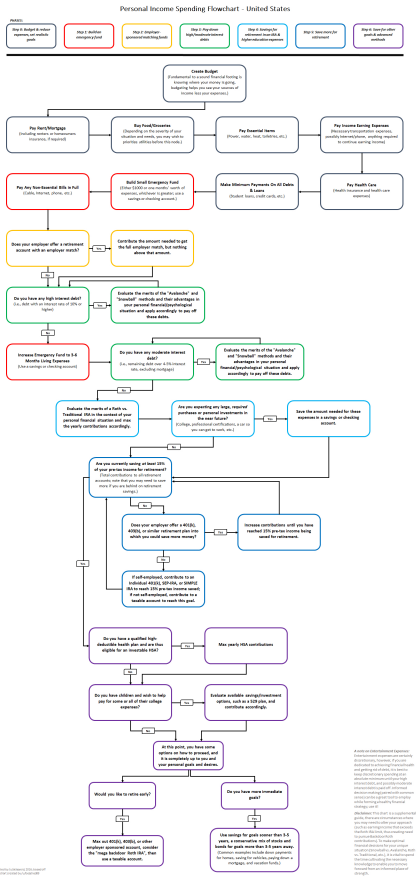
Decision Trees

You may be familiar with decision trees.



Decision Trees

There are worse ones:



Decision Trees

But they are very useful for guiding others on making decisions.

Do thing 1. If yes, go left, if no, go right. Do next thing.

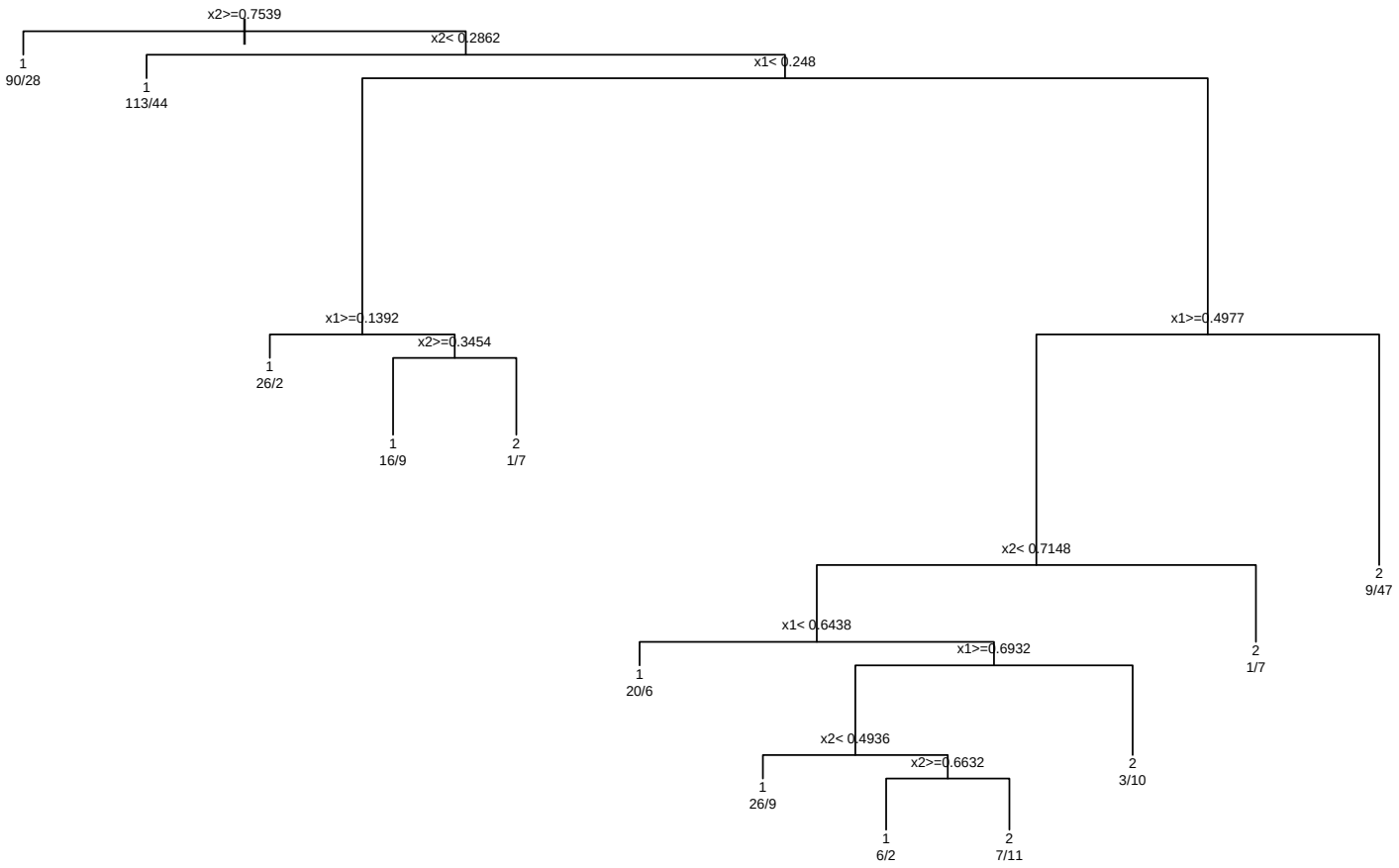
Classification Trees

Statistics decided to steal this structure for modelling.

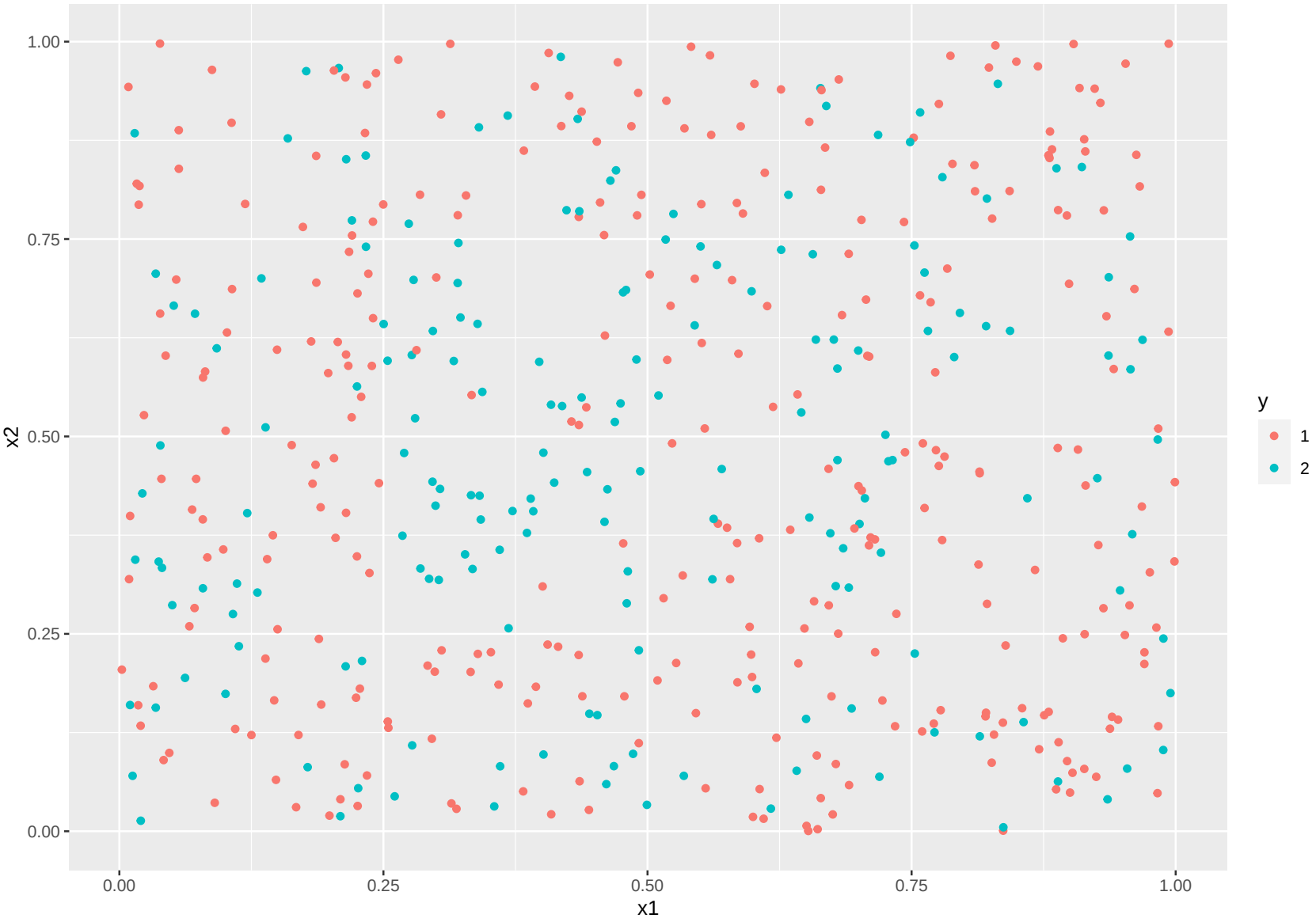
"CART" – Classification and Regression Trees. (We'll come back to regression side)

CART

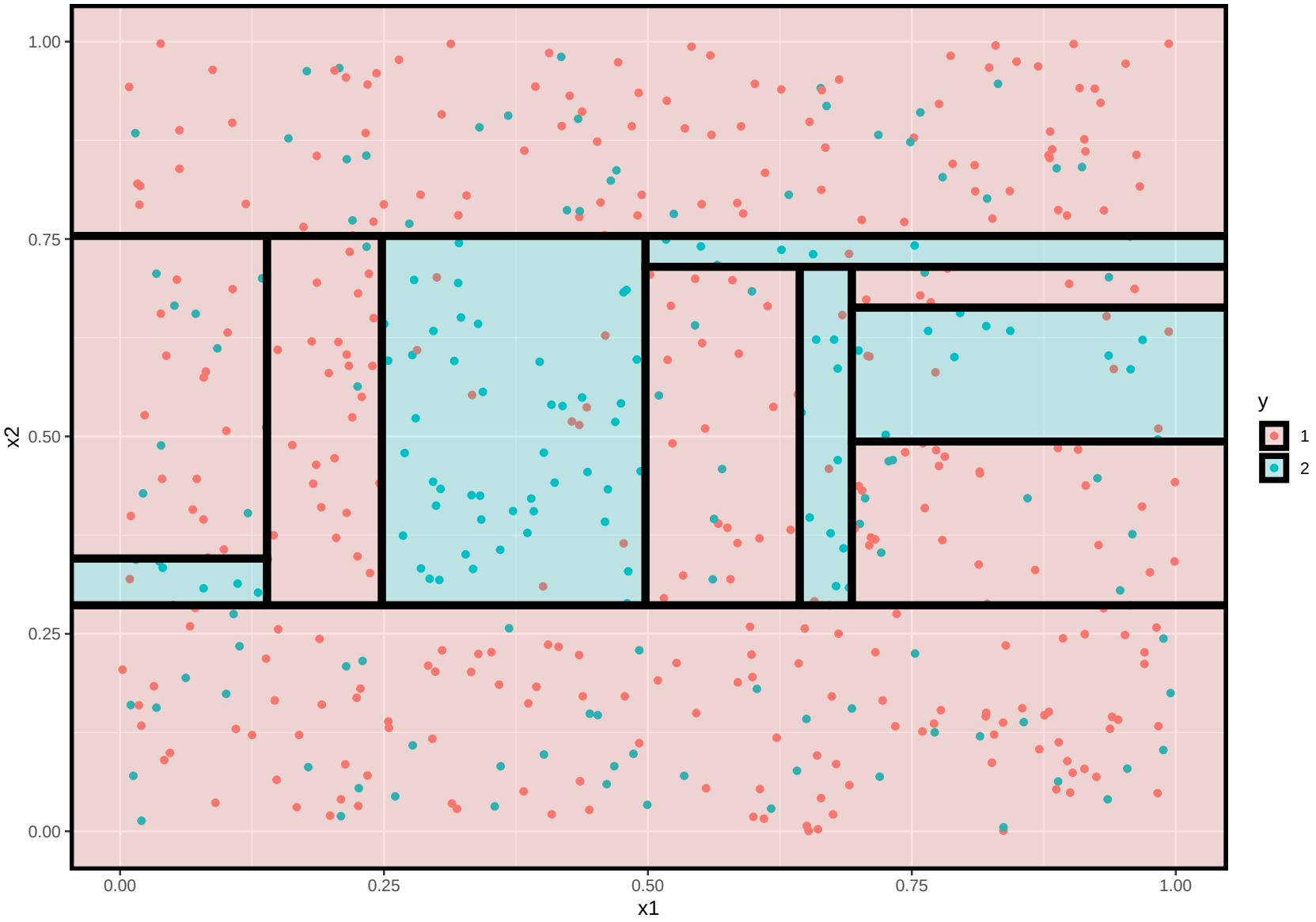
Basic Idea is that we want to split up the model space in a way that makes good predictions.



CART



CART



CART

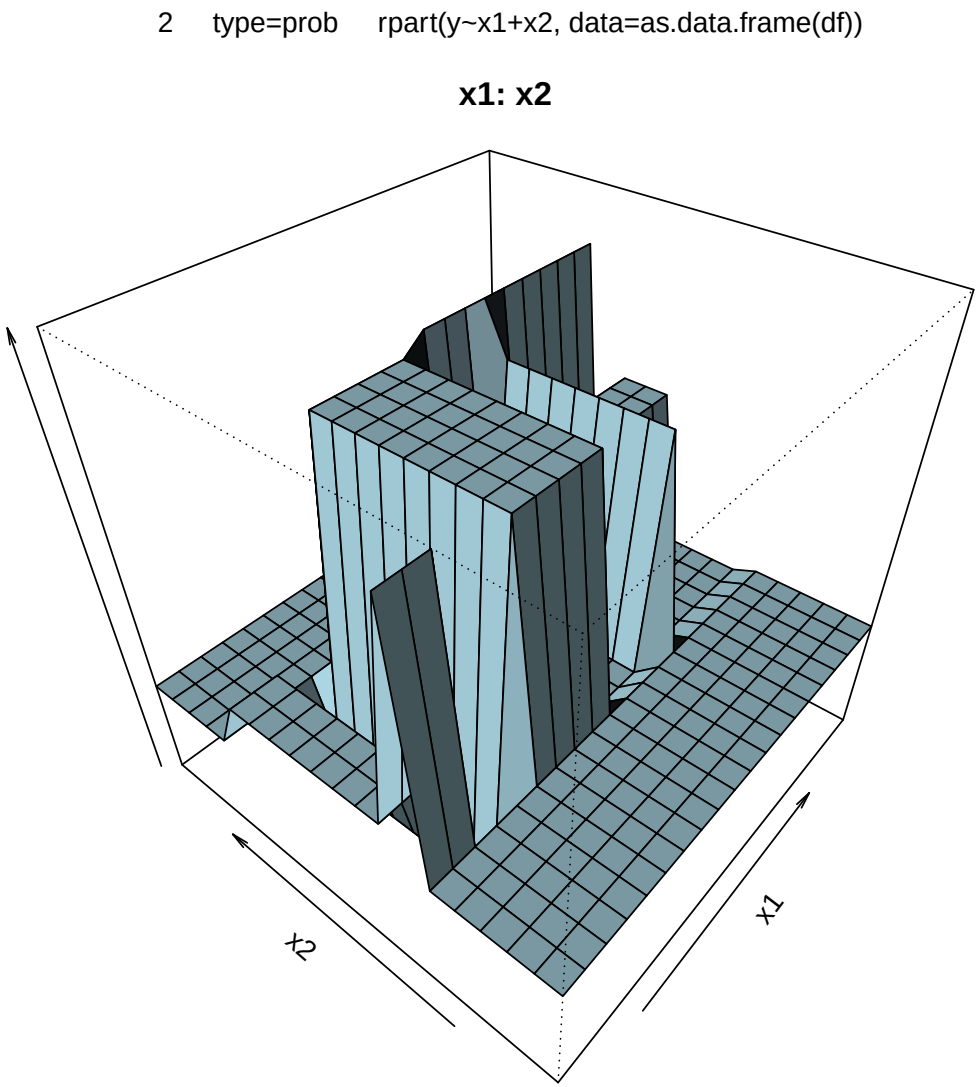
Number of possible "subsets" is even more *huge*. (See the number of splits with just 2 variables)

So we will again turn to a 'greedy' algorithm.

1. Find best place to split current data.
2. Split data there.
3. Find the best subset of current splits to split the data on.
4. Split there.
5. Repeat steps 3&4 until predictive performance improvement is marginal.

Recursive Partitioning

CART



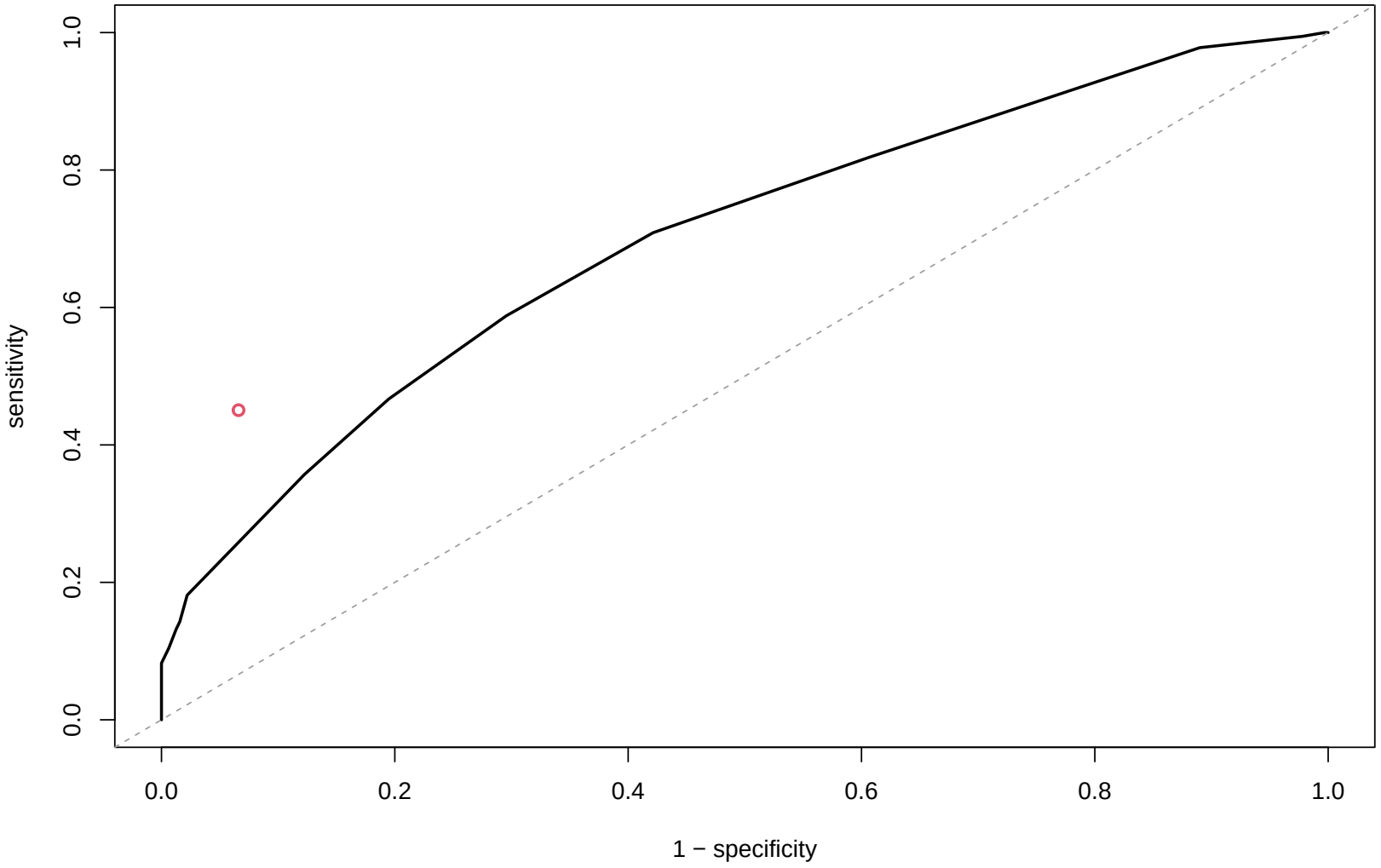
Prediction Errors:

For every observation in some subset, cost is about the misclassification risk.

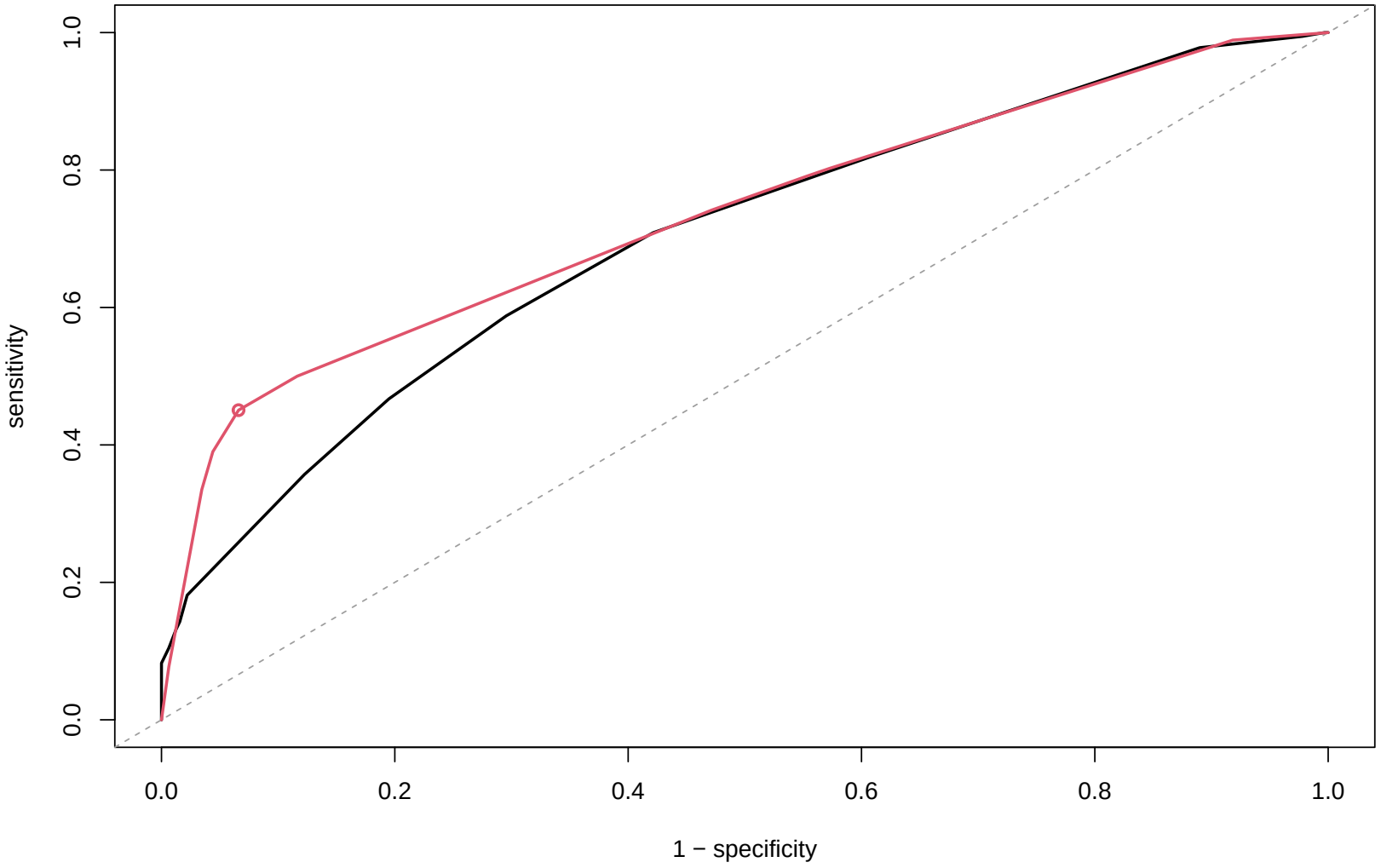
```
table(predicted=predict(mod,type="class"),actual=y)
```

```
##           actual
## predicted   1   2
##         1 297 100
##         2  21  82
```

Prediction Errors



Prediction Errors



CART: Details

How do we make the tree?

At each step

1. Loop through each node in the tree and
2. Loop through each variable and
3. Loop through every possible breakpoint and
4. Find the prediction error if we break here.

Now we know the new prediction error for every possible split.

CART: Details

Now we can:

1. Find the best prediction improvement for each variable in each node.
2. Find the best improvement for each node.
3. Find the best improvement across all nodes.
4. Make a new split there.

Repeat all of this.

CART: WHY?

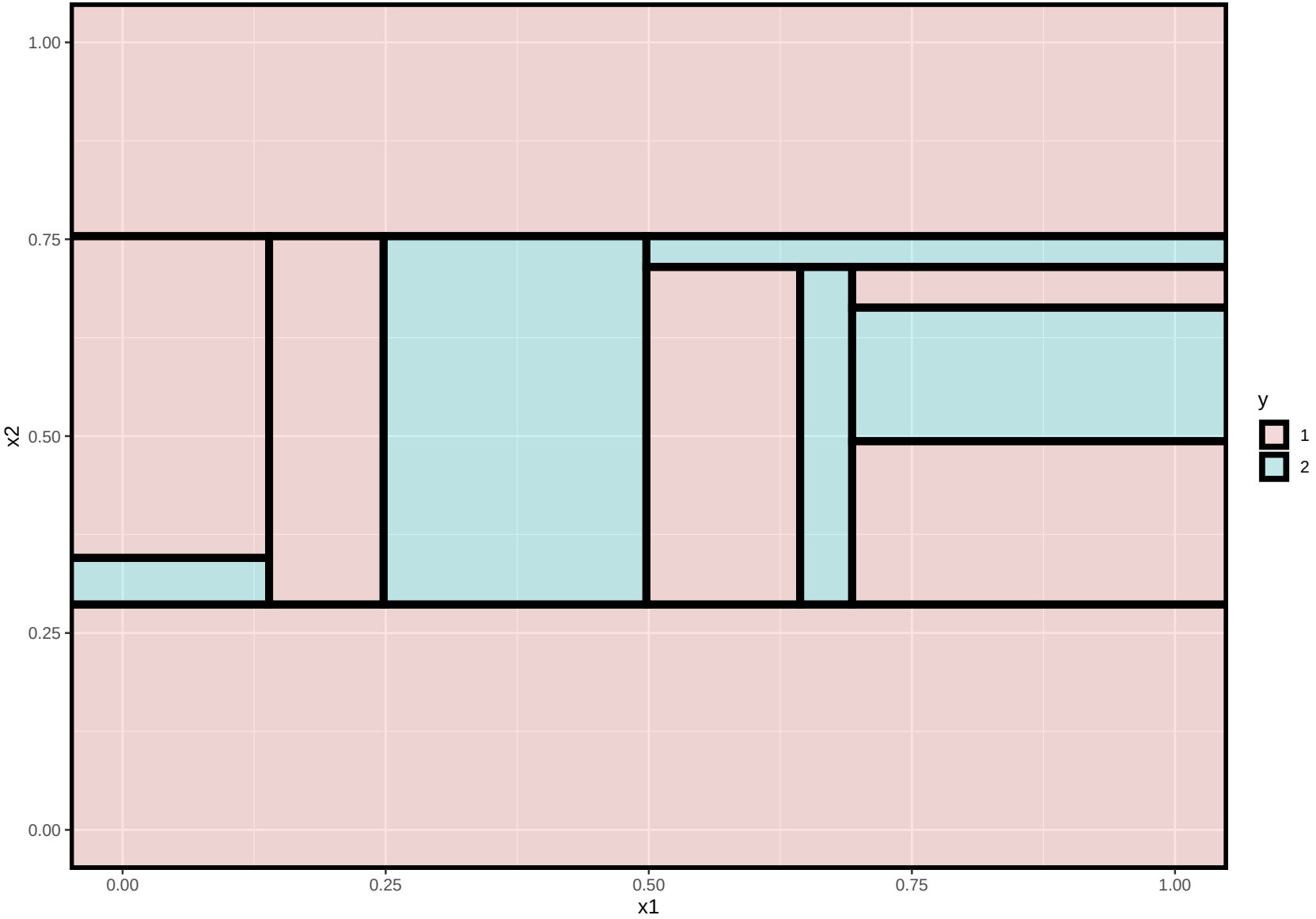
Automatic nonlinearities and interactions.

We can capture "regions" where weirdness happens without having to find those regions ourselves.

$\implies$ this is also why we have so many parameters, and so many models available.

CART: Problems?

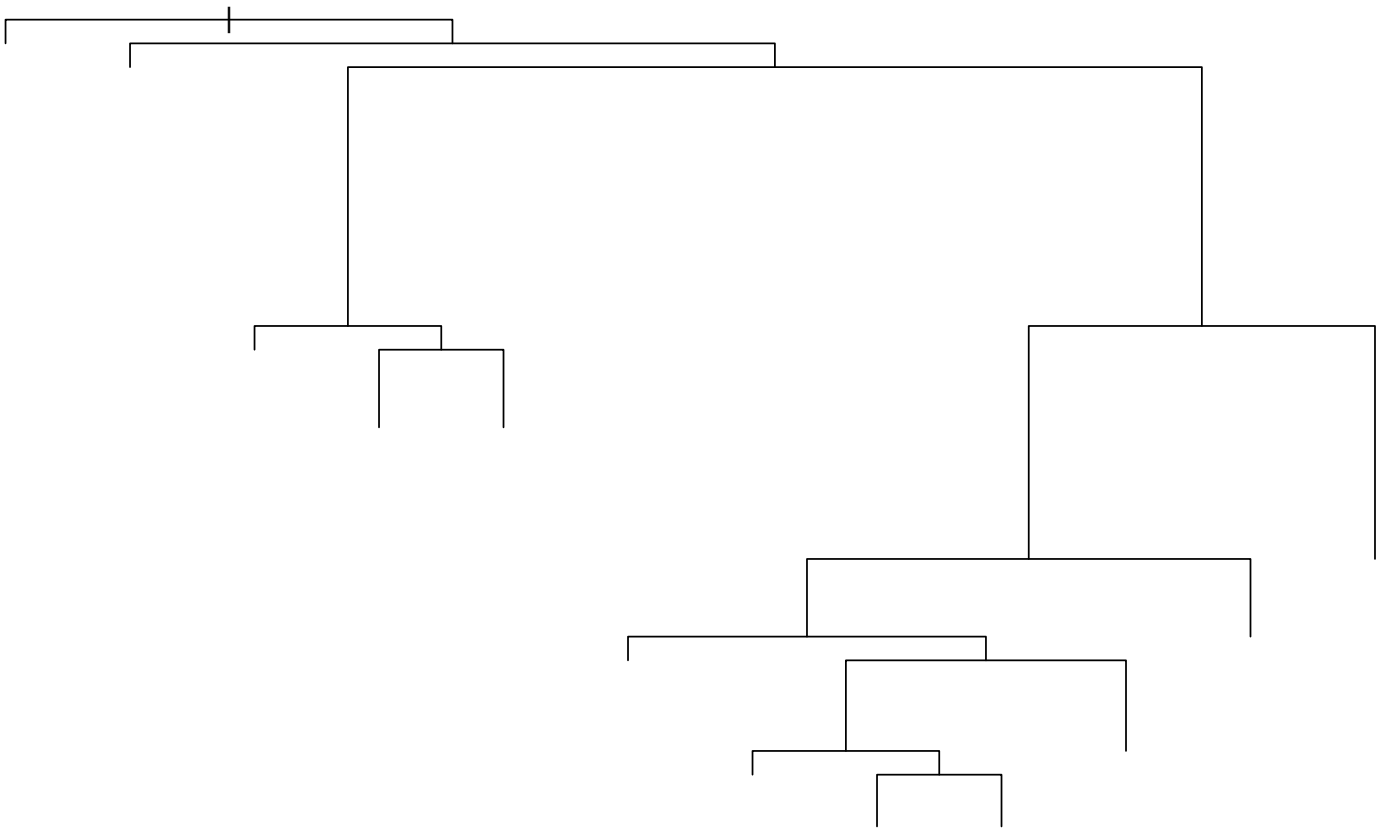
- Overfitting. Very fast overfitting.



CART: Overfitting

The overfitting comes from the "tree depth"

```
plot(mod)
```



Pruning

So we often fit an incredibly deep tree (this is quickish) and then prune back the really small nodes until we have something reasonable.

How much to prune?

Cross-validation will help again!

A basic overview of what we've seen in this class so far

- Model types:
  - Parametric: linear, logistic, LASSO
  - Non-parametric: KNN, trees
- Model Evaluation Tools:
  - Loss functions: MSE, l_p, Deviance
  - Misclassification Risk: sensitivity, specificity, ROC, AUC
- Model Selection Tools:
  - R^2
  - AIC, BIC
  - Cross-Validation

Questions for next class

1. How would you use trees to make predictions for real-valued outcomes (i.e. standard regression outcomes – not classificiation problems)
2. What if we have two different models performing well, but differently? Can we take advantage of both?

Wrap up

Things to do

HW 4 is due tomorrow night.

See you Thursday

Rehash

- ROC curves show us the predictive power across many possible threshold choices inside one model
- Trees can do an incredible job of fitting data with non-linearities and interactions. We will see more of this.

Bye!