MACHINE LEARNING TOOLBOX

Logistic regression on Sonar

Classification models

- Categorical (i.e. qualitative) target variable
- Example: will a loan default?
- Still a form of supervised learning
- Use a train/test split to evaluate performance
- Use the `Sonar` dataset
- Goal: distinguish rocks from mines

Example: Sonar data

```
> # Load the Sonar dataset
> library(mlbench)
> data(Sonar)

> # Look at the data
> Sonar[1:6, c(1:5, 61)]
      V1     V2     V3     V4     V5 Class
1 0.0200 0.0371 0.0428 0.0207 0.0954     R
2 0.0453 0.0523 0.0843 0.0689 0.1183     R
3 0.0262 0.0582 0.1099 0.1083 0.0974     R
4 0.0100 0.0171 0.0623 0.0205 0.0205     R
5 0.0762 0.0666 0.0481 0.0394 0.0590     R
6 0.0286 0.0453 0.0277 0.0174 0.0384     R
```

Splitting the data

- Randomly split data into training and test sets
- Use a 60/40 split, instead of 80/20
- `Sonar` dataset is small, so 60/40 gives a larger, more reliable test set

Splitting the data

```
# Randomly order the dataset
> rows <- sample(nrow(Sonar))
> Sonar <- Sonar[rows, ]

# Find row to split on
> split <- round(nrow(Sonar) * .60)
> train <- Sonar[1:split, ]
> test <- Sonar[(split + 1):nrow(Sonar), ]

# Confirm test set size
> nrow(train) / nrow(Sonar)
[1] 0.6009615
```

Let's practice!

Confusion matrix

Confusion matrix

Prediction \ Reference	Yes	No
Yes	True positive	False positive
No	False negative	True negative

Confusion matrix

```
# Fit a model
> model <- glm(Class ~ ., family = binomial(link = "logit"),
train)
> p <- predict(model, test, type = "response")
> summary(p)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.0000  0.9885  0.5296  1.0000  1.0000

# Turn probabilities into classes and look at their frequencies
> p_class <- ifelse(p > .50, "M", "R")
> table(p_class)
p_class
 M  R
44 39
```

Confusion matrix

- Make a 2-way frequency table
- Compare predicted vs. actual classes

```
# Make simple 2-way frequency table
> table(p_class, test[["Class"]])
p_class  M  R
      M 13 31
      R 30  9
```

Confusion matrix

```
# Use caret's helper function to calculate additional statistics
> confusionMatrix(p_class, test[["Class"]])
         Reference
Prediction  M  R
         M 13 31
         R 30  9

               Accuracy : 0.2651
                 95% CI : (0.1742, 0.3734)
    No Information Rate : 0.5181
    P-Value [Acc > NIR] : 1

                  Kappa : -0.4731
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.3023
            Specificity : 0.2250
         Pos Pred Value : 0.2955
         Neg Pred Value : 0.2308
```

Let's practice!

Class probabilities and class predictions

Different thresholds

- Not limited to 50% threshold
  - 10% would catch more mines with less certainty
  - 90% would catch fewer mines with more certainty
- Balance true positive and false positive rates
- Cost-benefit analysis

Confusion matrix

```
# Use a larger cutoff
> p_class <- ifelse(p > .99, "M", "R")
> table(p_class)
p_class
 M  R
41 42

# Make simple 2-way frequency table
> table(p_class, test[["Class"]])
p_class  M  R
      M 13 28
      R 30 12
```

Confusion matrix with caret

```
# Use caret to produce confusion matrix
> confusionMatrix(p_class, test[["Class"]])
          Reference
Prediction  M  R
         M 13 28
         R 30 12

               Accuracy : 0.3012
                 95% CI : (0.2053, 0.4118)
    No Information Rate : 0.5181
    P-Value [Acc > NIR] : 1.0000

                  Kappa : -0.397
 Mcnemar's Test P-Value : 0.8955

            Sensitivity : 0.3023
            Specificity : 0.3000
         Pos Pred Value : 0.3171
         Neg Pred Value : 0.2857
```

Let's practice!

Introducing the ROC curve

The challenge

- Many possible classification thresholds
- Requires manual work to choose
- Easy to overlook a particular threshold
- Need a more systematic approach

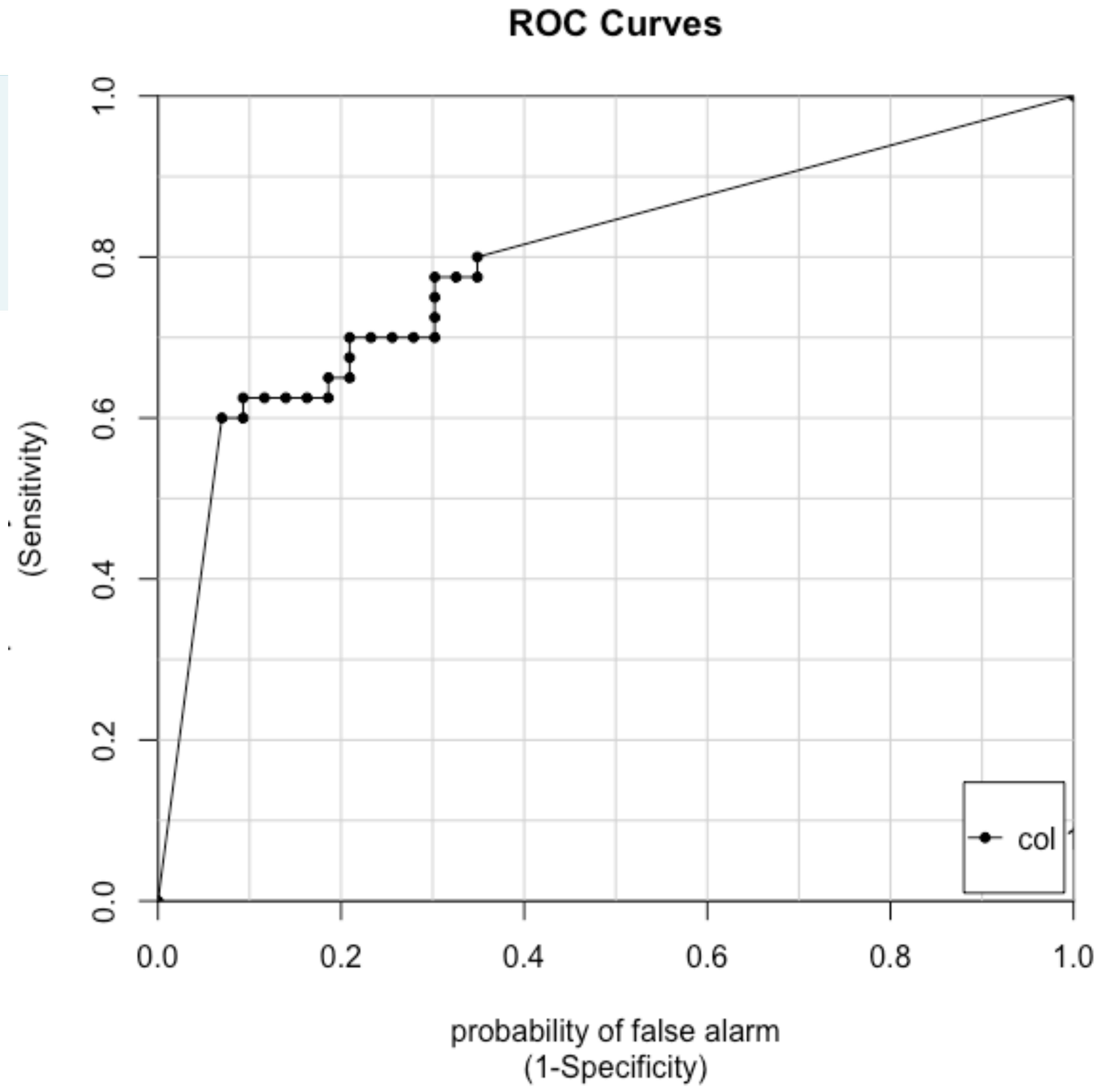
ROC curves

- Plot true/false positive rate at every possible threshold
- Visualize tradeoffs between two extremes **100% true positive rate vs. 0% false positive rate**
- Result is an ROC curve
- Developed as a method for analyzing radar signals

An example ROC curve

```
# Create ROC curve
> library(caTools)
> colAUC(p, test[["Class"]], plotROC = TRUE)
```

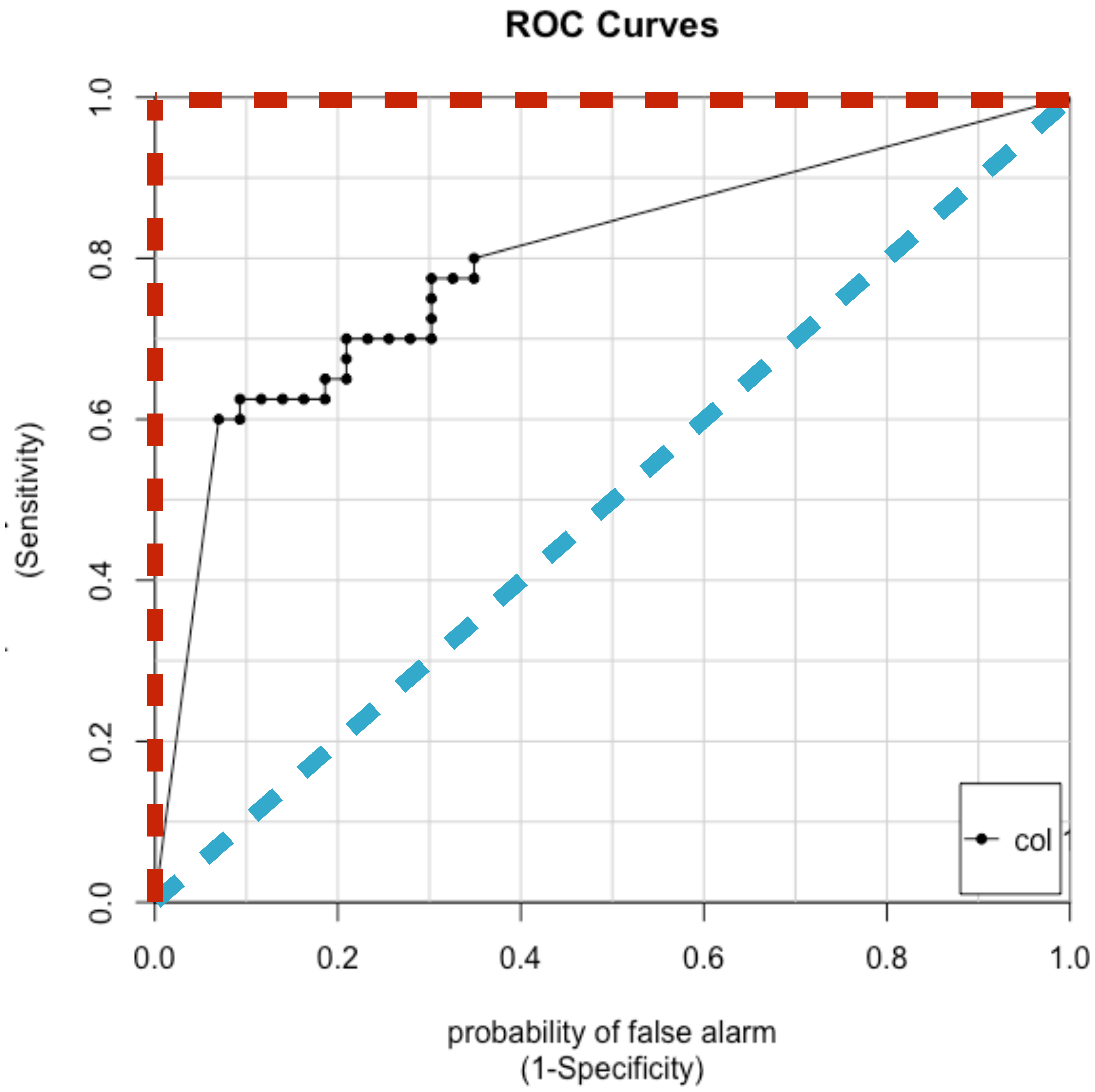
- X-axis: false positive rate
- Y-axis: true positive rate
- Each point along the curve represents a different threshold



Let's practice!

Area under the curve (AUC)

From ROC to AUC



Defining AUC

- Single-number summary of model accuracy
- Summarizes performance across all thresholds
- Rank different models within the same dataset

Defining AUC

- Ranges from 0 to 1
  - 0.5 = random guessing
  - 1 = model always right
  - 0 = model always wrong
- Rule of thumb: AUC as a letter grade
  - 0.9 = "A"
  - 0.8 = "B"
  - ...

Let's practice!