University of Colorado **Boulder**

Department of Computer Science

CSCI 5622: Machine Learning

Chris Ketelsen

Lecture 6:
Validation Techniques and Evaluation Metrics

The Story So Far

We've seen a few classification methods (KNN, NB, LogReg)

You've done your own experiments with KNN and answered some questions like:

- How does the amount of training data affect **accuracy**?
- How does the choice of K affect **accuracy**?

We've talked about the importance of evaluating a learning model on a set of **held-out data**

You've been introduced to the **Confusion Matrix** and what it can tell you about how your learning algorithm makes mistakes

The Story So Far

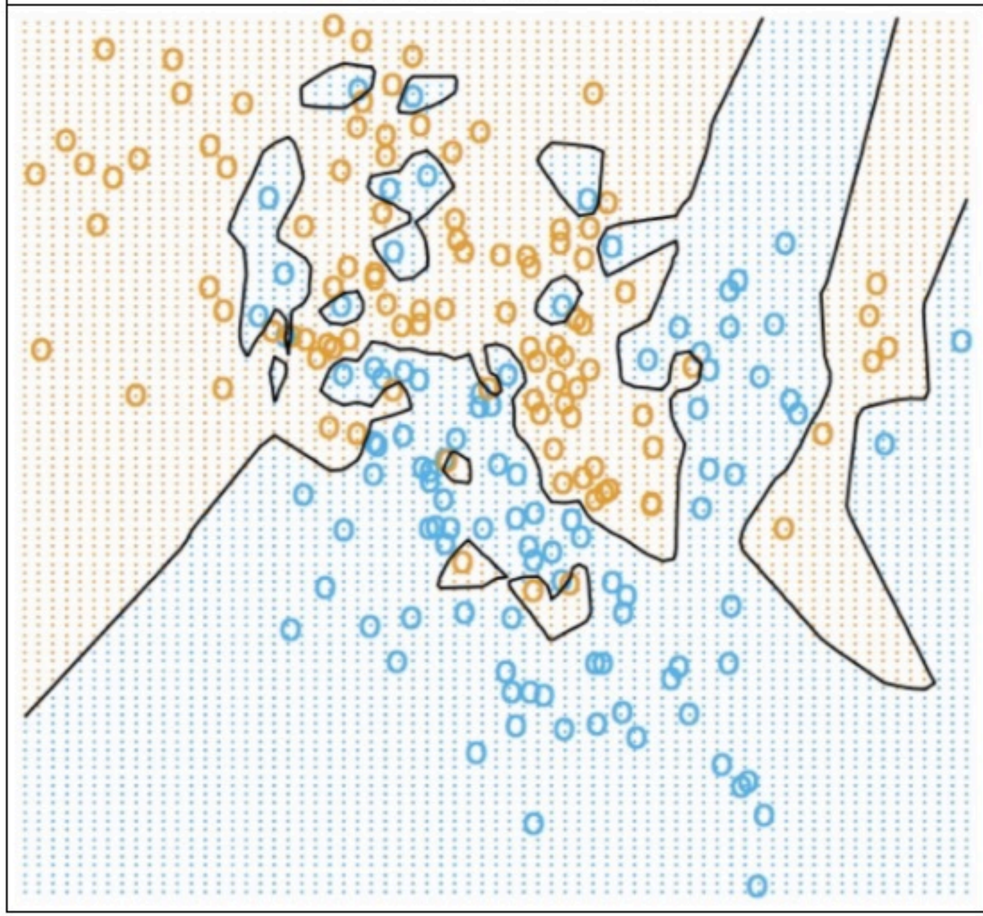
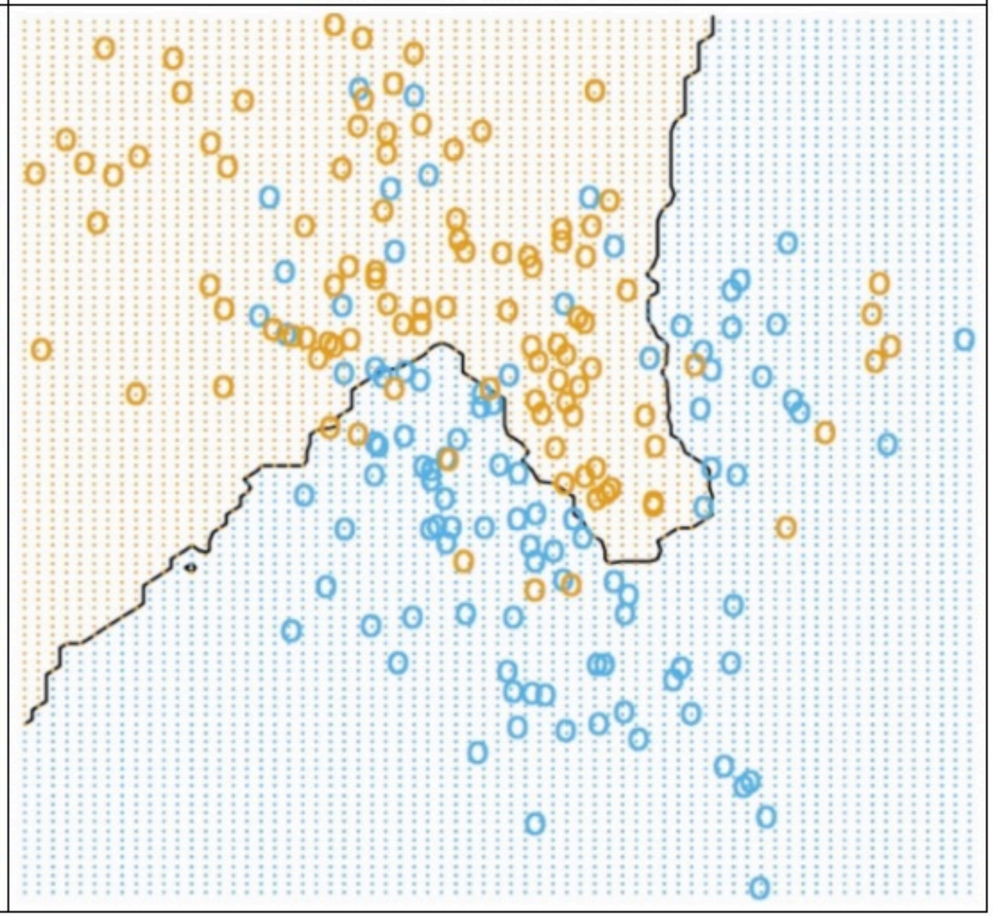
Today we'll:

- look closer at how we validate our learning models
- ask questions about if we can believe our validation results
- ask questions about whether our simple accuracy measurements really are good measurements

Train / Validation / Test - Split

Overfitting occurs when our model works too hard to fit training data

Really care about how model does on unseen test data

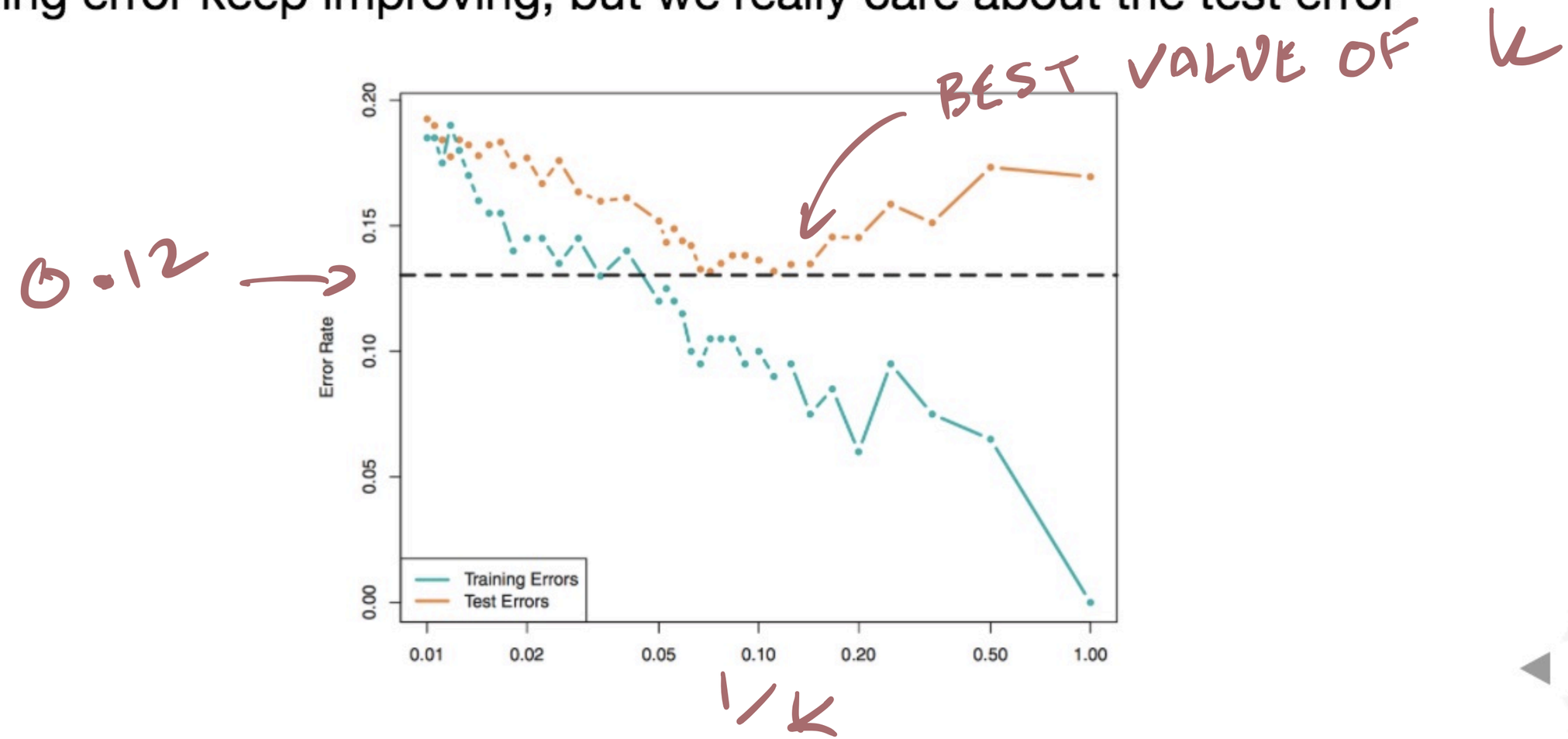
KNN: K = 1	KNN: K = 15
	

Train / Validation / Test - Split

Need to determine good tuning parameters for model (K in KNN)

Helpful to plot training and test error vs. model complexity

Training error keep improving, but we really care about the test error



Train / Validation / Test - Split

The test error serves multiple purposes:

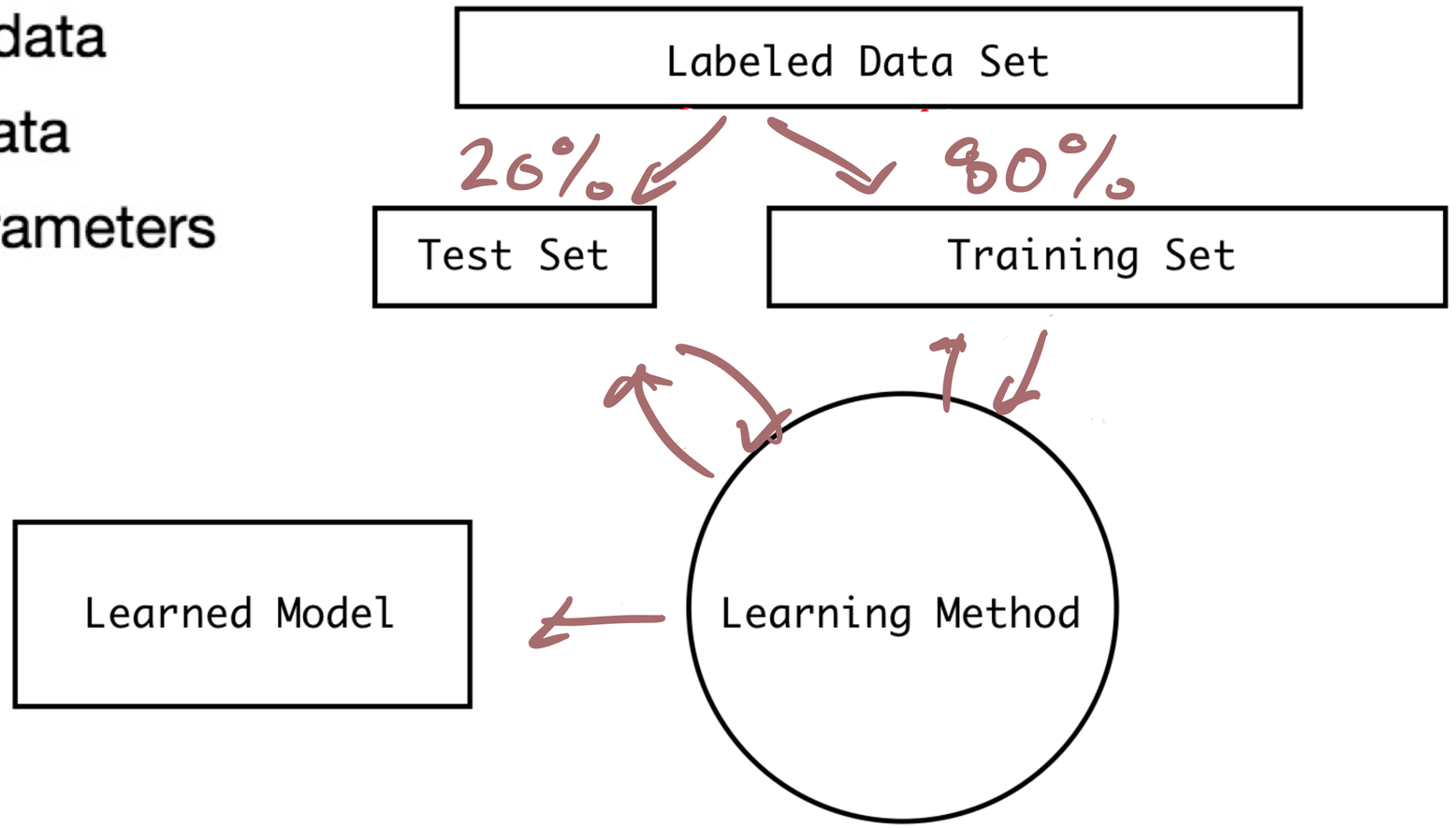
- Helps us find optimal tuning parameters
- Gives estimate of how model will perform out in real world

But, need to think hard about how much we can trust the test error

To do that we need to think about the life-cycle of a learning model

Train / Validation / Test - Split

- Train on training data
- Test on testing data
- Tweak tuning parameters
- Repeat

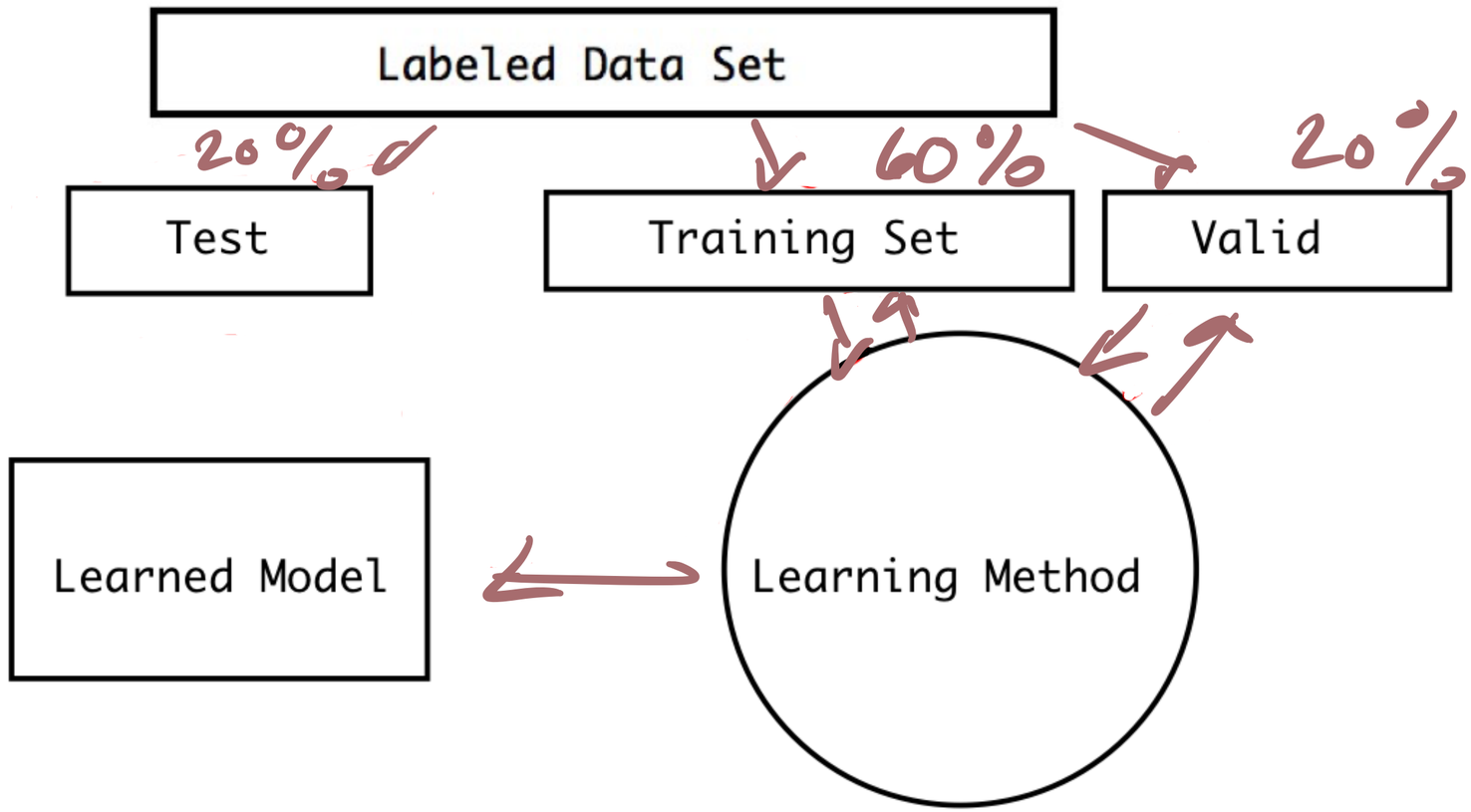


Seems reasonable. Probably can get an accurate model this way

But what are the **potential pitfalls**?

Train / Validation / Test - Split

- Can actually do **human overfitting** of test data
- Test error no longer accurate estimator of true error
- Need another partition - validation set



- 60% / 20% / 20% - split is popular but not a firm rule

Train / Validation / Test - Split

OK, but that kinda sucks. We just lost 20% of our training data

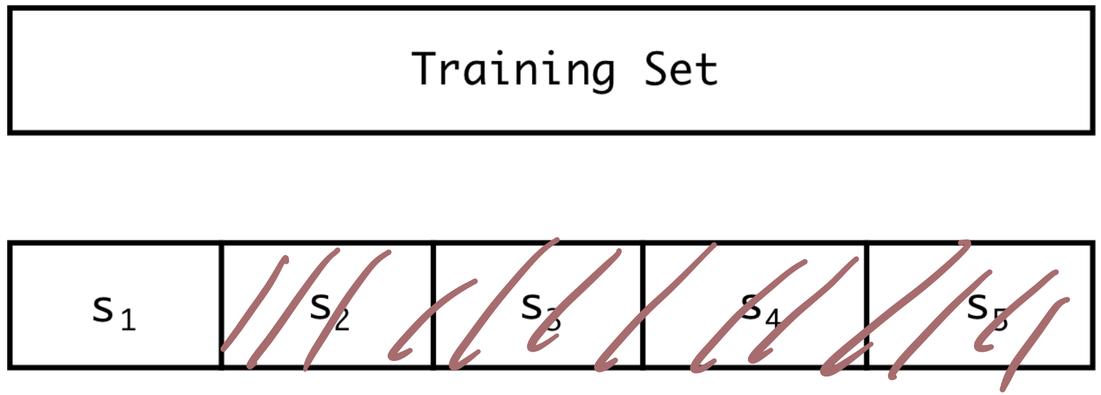
What if you just can't spare it?

Step 1: Try to get more data

Step 2: Do **Cross-Validation**

k-Fold Cross-Validation

Randomly(ish) partition training data into k folds (partitions)



5 DIFFERENT FOLDS.

Iteratively leave one fold out for testing, train model on the rest

iteration	train on	test on	accuracy
1	s_1, s_2, s_3, s_4	s_5	11/20
2	s_1, s_2, s_3, s_5	s_4	17/20
3	s_1, s_2, s_4, s_5	s_3	16/20
4	s_1, s_3, s_4, s_5	s_2	13/20
5	s_2, s_3, s_4, s_5	s_1	16/20

k-Fold Cross-Validation

Iteratively leave one fold out for testing, train model on the rest

iteration	train on	test on	accuracy
1	s_1, s_2, s_3, s_4	s_5	11/20
2	s_1, s_2, s_3, s_5	s_4	17/20
3	s_1, s_2, s_4, s_5	s_3	16/20
4	s_1, s_3, s_4, s_5	s_2	13/20
5	s_2, s_3, s_4, s_5	s_1	16/20

Use mean of CV error rate as estimate of prediction error rate

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} CV_i = \frac{1}{k}\sum_{i=1}^{k}\frac{1}{|F_i|}\sum_{j\in F_i} \mathcal{I}(\hat{y}_j \neq y_j)$$

wrong / total

k-Fold Cross-Validation

Iteratively leave one fold out for testing, train model on the rest

iteration	train on	test on	accuracy
1	s_1, s_2, s_3, s_4	s_5	11/20
2	s_1, s_2, s_3, s_5	s_4	17/20
3	s_1, s_2, s_4, s_5	s_3	16/20
4	s_1, s_3, s_4, s_5	s_2	13/20
5	s_2, s_3, s_4, s_5	s_1	16/20

Use mean of CV error rate as estimate of prediction error rate

Example: $CV_{(5)} = \frac{1}{5}\left(\frac{9}{20} + \frac{3}{20} + \frac{4}{20} + \frac{7}{20} + \frac{4}{20}\right) = 0.27$

Corresponds to an accuracy of 73%

But can we even **believe** this?!

k-Fold Cross-Validation

But can we even **believe** this?!

Sorta, but don't take it as gospel. Use statistics

Compute standard deviation of fold-errors

$$SD = \sqrt{\text{var}(CV_1, CV_2, \ldots, CV_k)}$$

Compute the standard error $SE = \dfrac{SD}{\sqrt{k}}$

Then the 95% confidence interval is roughly $CV_{(k)} \pm 2\ SE$

Example: 95% confidence interval 0.27 ± 0.10

$[.17, .37]$

k-Fold Cross-Validation

So how many folds should I use?

The state-of-the-art strategies for CV are very much rules of thumb and not justified by particularly rigorous theory

Most people use $k = 5$ or $k = 10$

Some people will do bootstrapping or multiple 10-fold runs of CV

The larger k is the less biased the estimate is, but the more variable and more expensive it becomes

Validation

Take-Aways:

- Test Set is sacred and not to be touched for validation
- Either have a validation set or do Cross-Validation

Next Up:

- Is accuracy / misclassification error always the best metric for evaluating our model?
- Can we do better?

Evaluation Metrics

Thus far we have only been concerned with the misclassification error rate and the standard definition of accuracy that comes with it

INDICATOR FUNCTION

$$\text{Err} = \frac{1}{m}\sum_{i=1}^{m} \mathcal{I}\left(\hat{y}_i \neq y_i\right)$$

$\frac{1}{m}$ · # WRONG

And for many binary classification tasks this is perfectly fine

In fact, many of the methods we'll look at are designed specifically to minimizes this error rate

But there are many common situations when the misclassification error is misleading

RATE

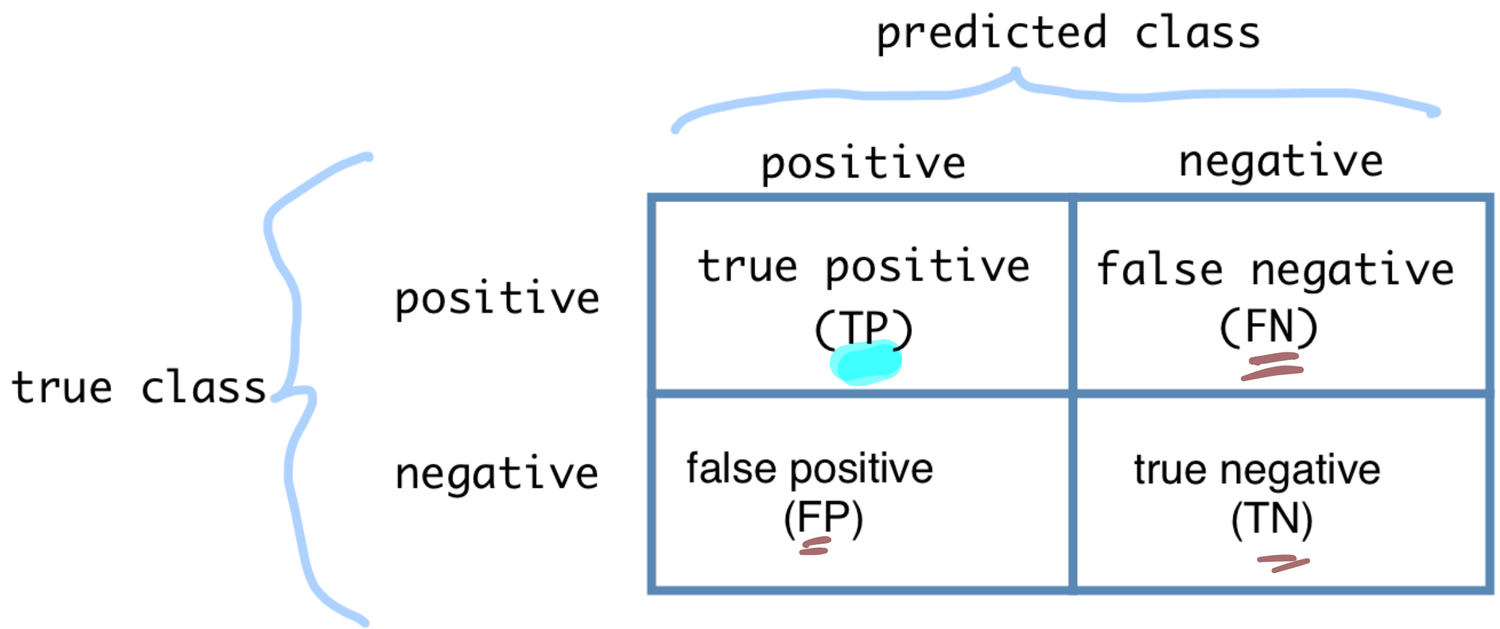
Evaluation Metrics

- Consider the case when your training set is heavily skewed towards a particular class
    - If 98% of training data is from negative class, should you feel good about a model with 98.5% accuracy?

- What about when there are different consequences for types of misclassification?
    - If you're trying to predict if a patient has cancer, you'd much rather commit a false positive misclassification than a false negative

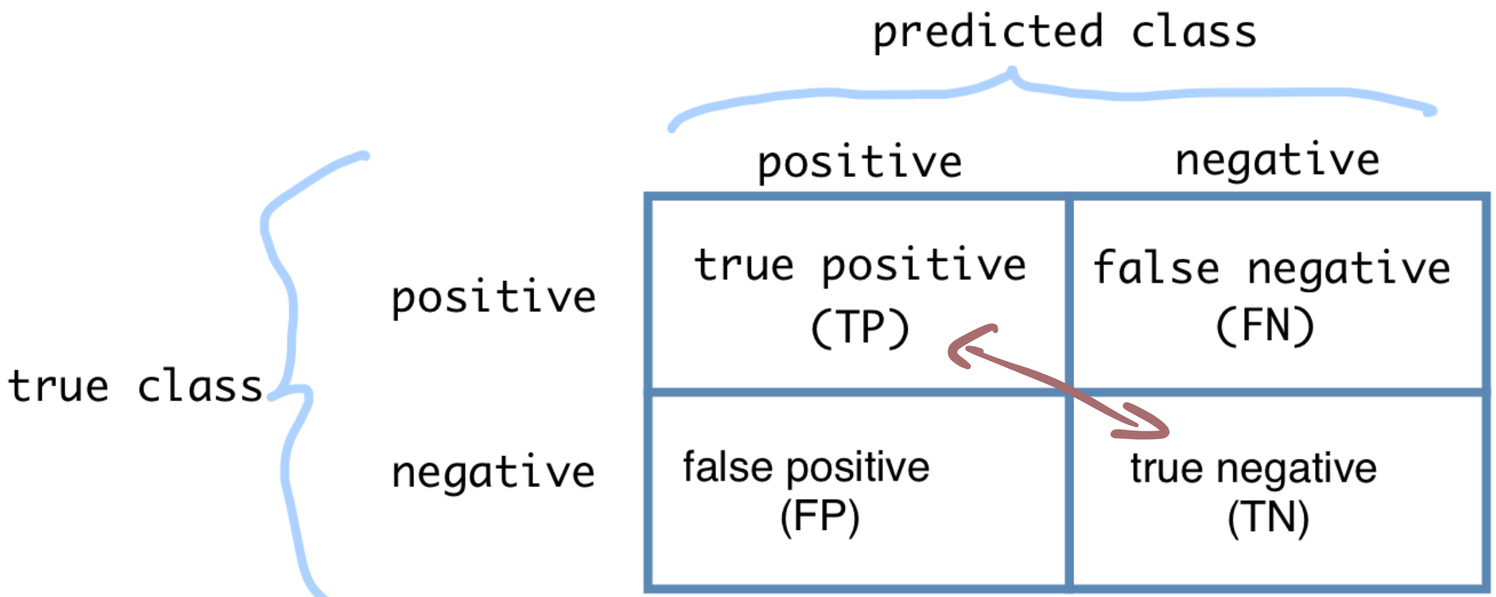
Evaluation Metrics

- Consider the case when your training set is heavily skewed towards a particular class



The Confusion Matrix

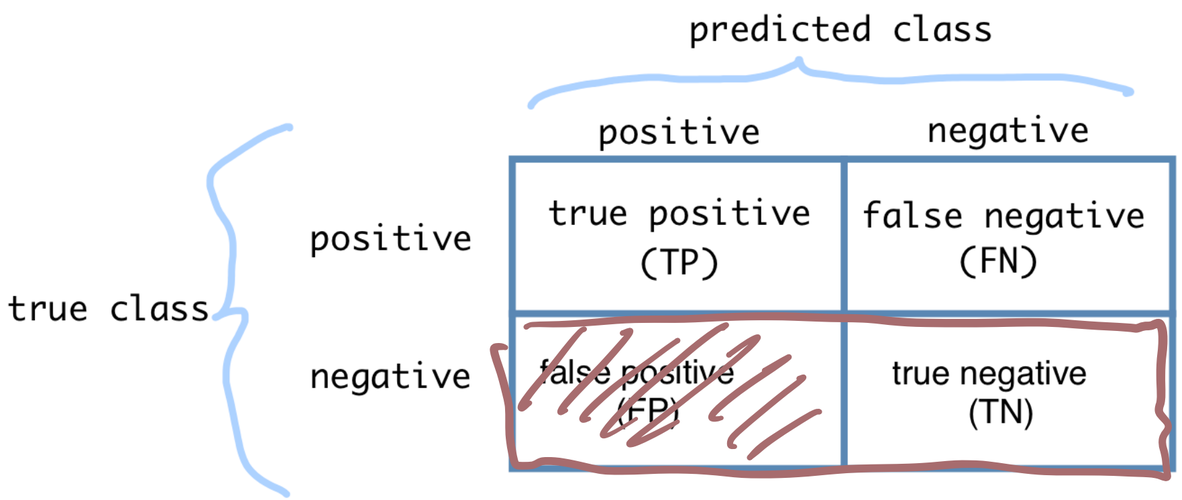
Everything starts from the Confusion matrix



$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

The Confusion Matrix

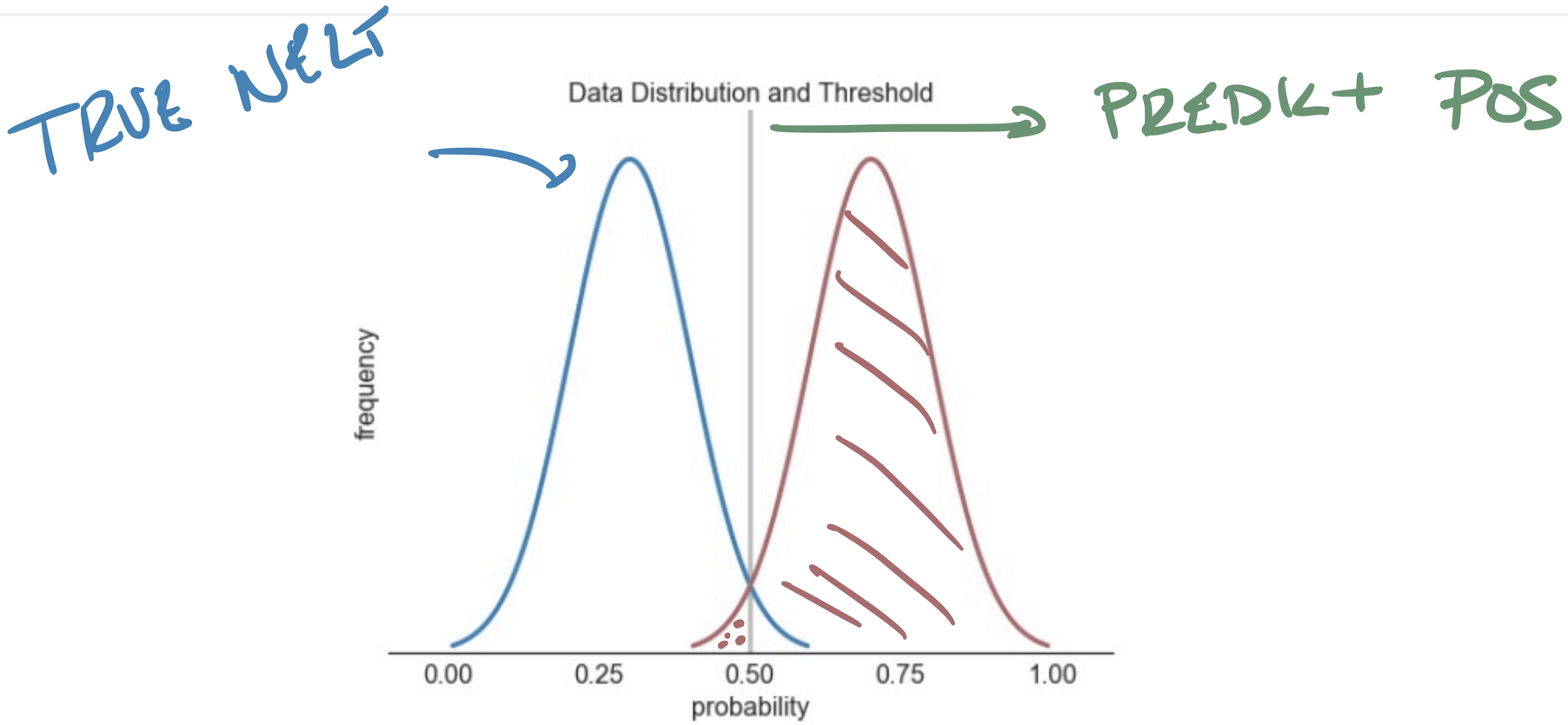
Everything starts from the Confusion matrix



$$\text{True Positive Rate} = P(\text{predict Pos} \mid \text{is Pos}) = \frac{TP}{TP + FN}$$

$$\text{False Positive Rate} = P(\text{predict Pos} \mid \text{is Neg}) = \frac{FP}{FP + TN}$$

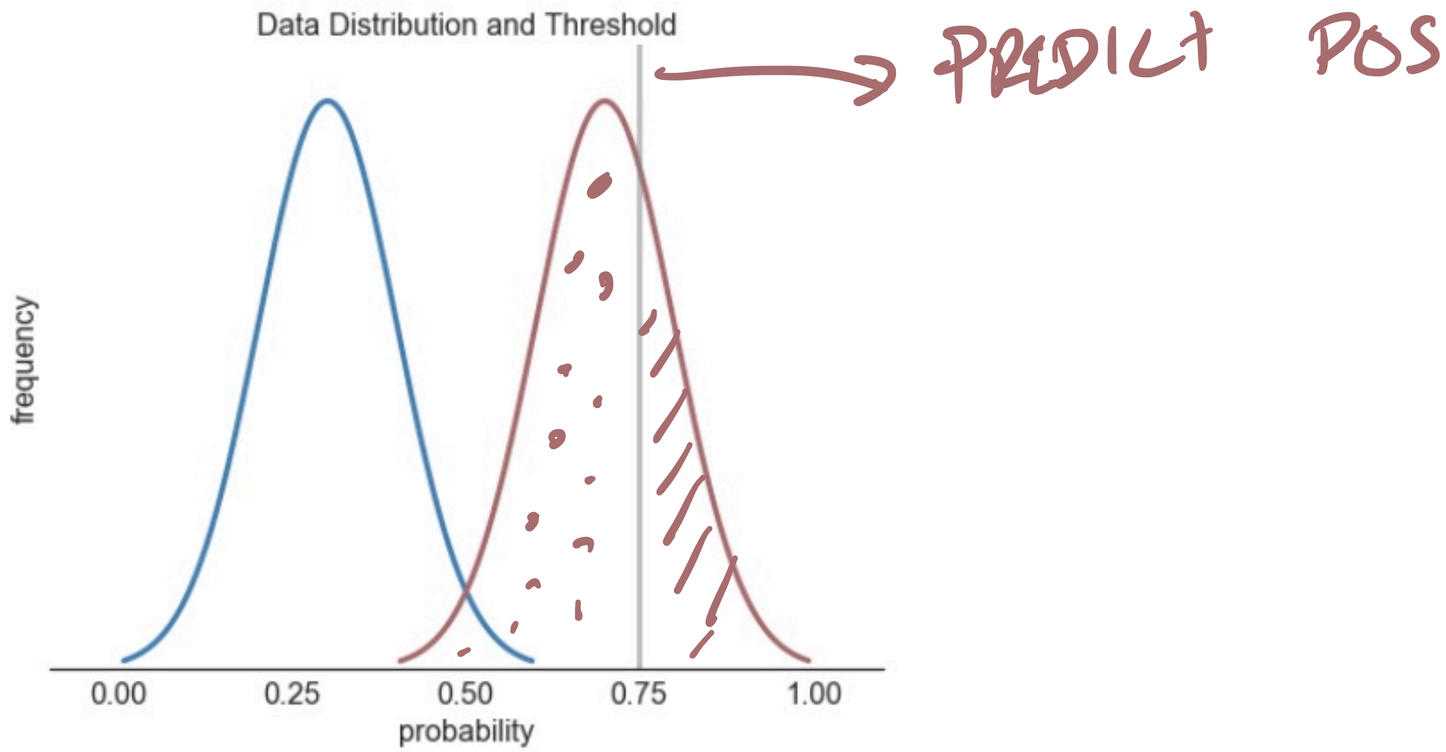
TPR vs. FPR



$$\text{True Positive Rate} = P(\text{predict Pos} \mid \text{is Pos}) = \frac{TP}{TP + FN}$$

$$\text{False Positive Rate} = P(\text{predict Pos} \mid \text{is Neg}) = \frac{FP}{FP + TN}$$

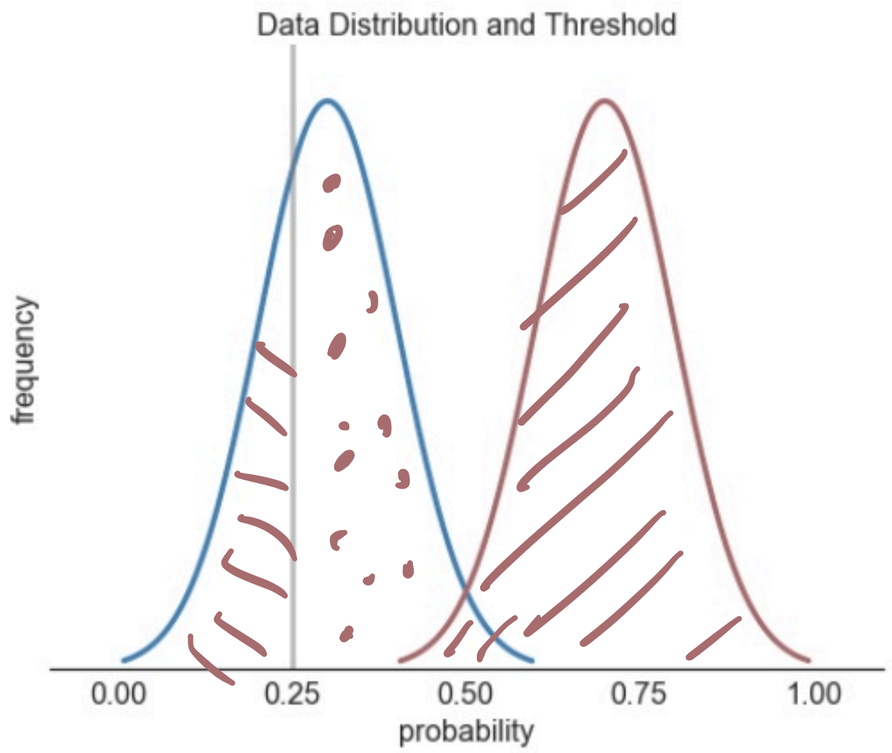
TPR vs. FPR



$$\text{True Positive Rate} = P(\text{predict Pos} \mid \text{is Pos}) = \frac{TP}{TP + FN}$$

$$\text{False Positive Rate} = P(\text{predict Pos} \mid \text{is Neg}) = \frac{FP}{FP + TN}$$

TPR vs. FPR

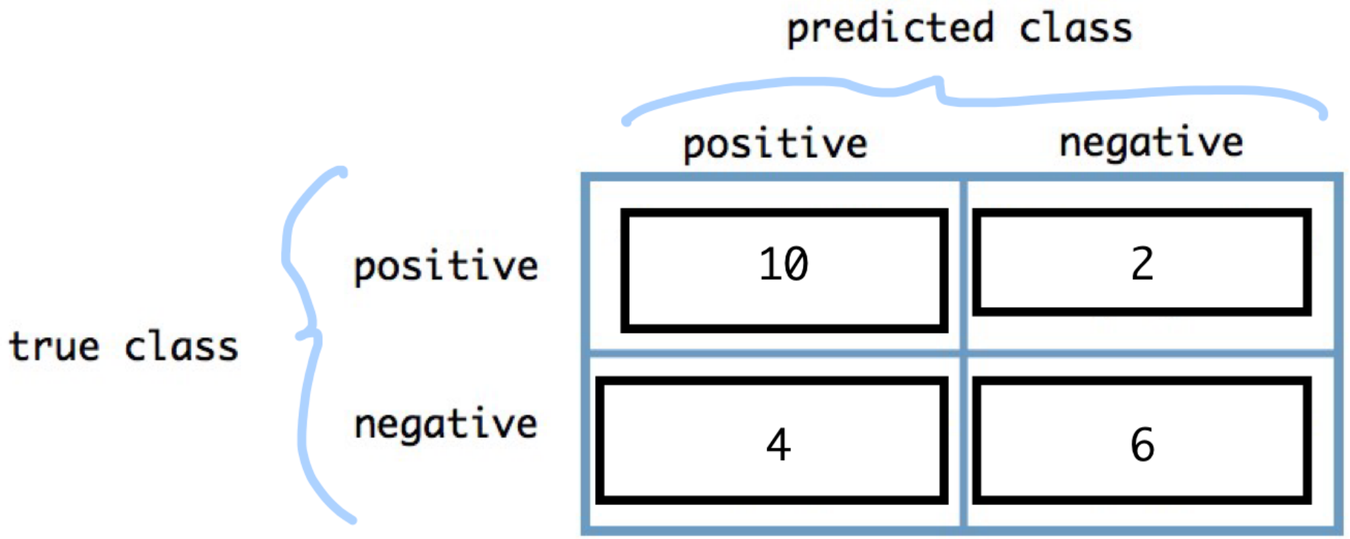


$$\text{True Positive Rate} = P(\text{predict Pos} \mid \text{is Pos}) = \frac{TP}{TP + FN} = 1$$

$$\text{False Positive Rate} = P(\text{predict Pos} \mid \text{is Neg}) = \frac{FP}{FP + TN} \ll \frac{1}{2}$$

The Confusion Matrix

Example: Compute TPR, FPR from the following Confusion Matrix

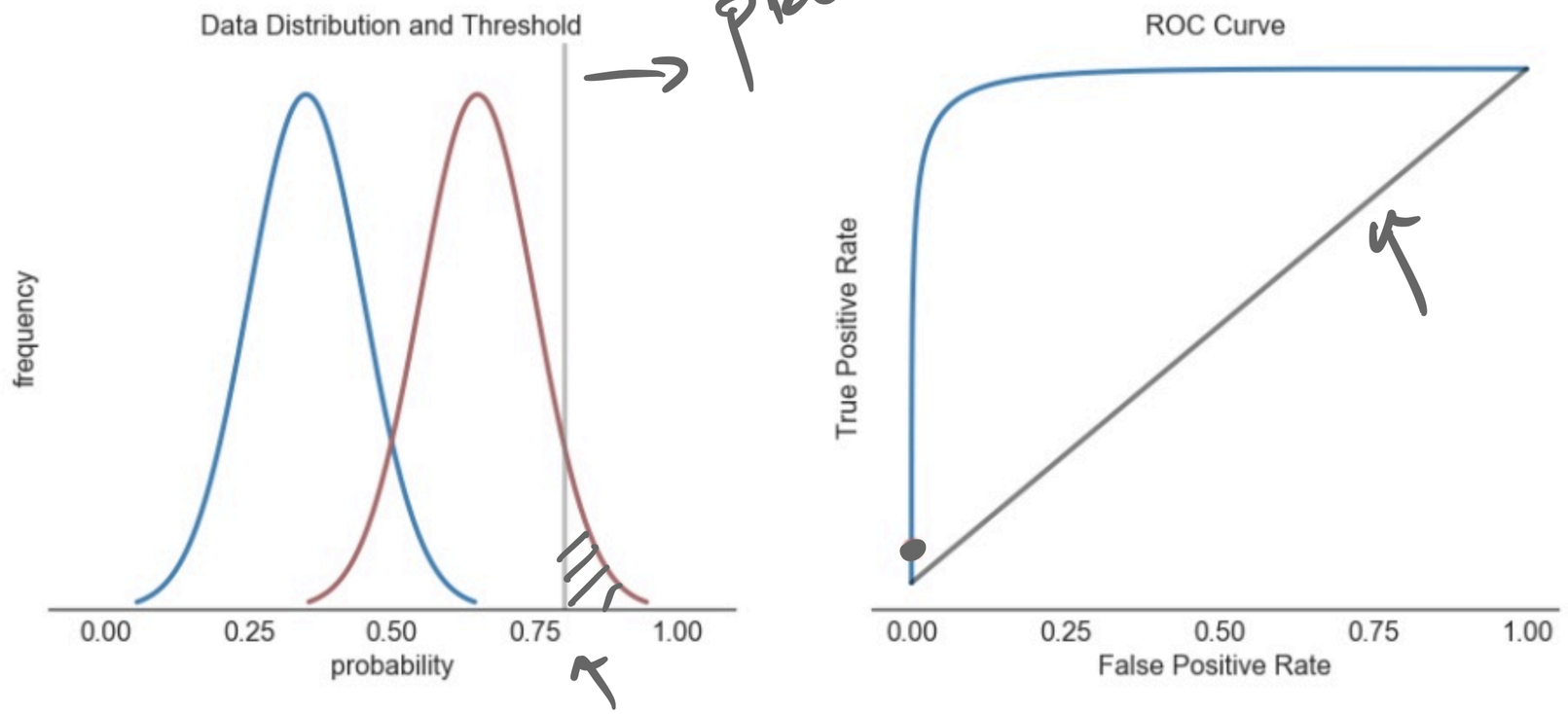


$$\text{TPR} = P(\text{predict Pos} \mid \text{is Pos}) = \frac{TP}{TP + FN} = \frac{10}{12} = \frac{5}{6}$$

$$\text{FPR} = P(\text{predict Pos} \mid \text{is Neg}) = \frac{FP}{FP + TN} = \frac{4}{10} = \frac{2}{5}$$

The ROC Curve

FPR ≃ 0
TPR ≃ 0

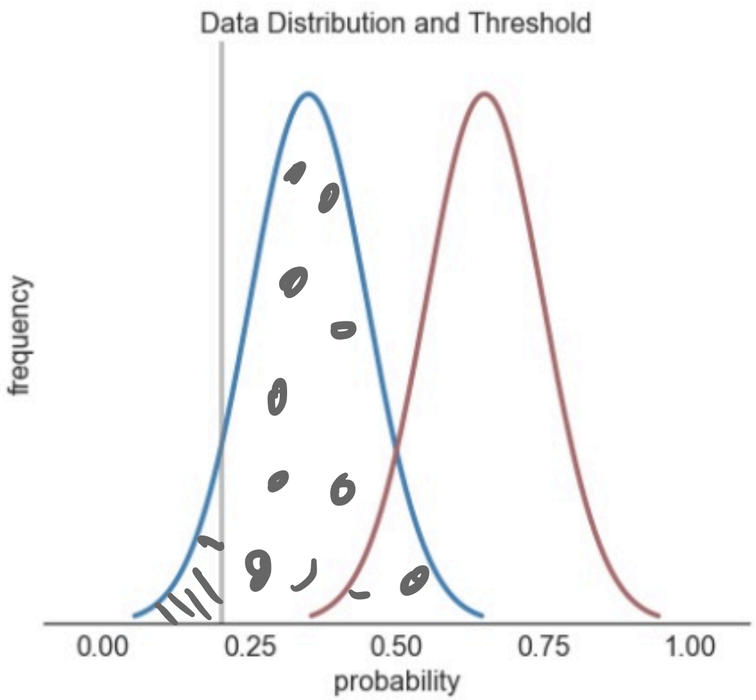


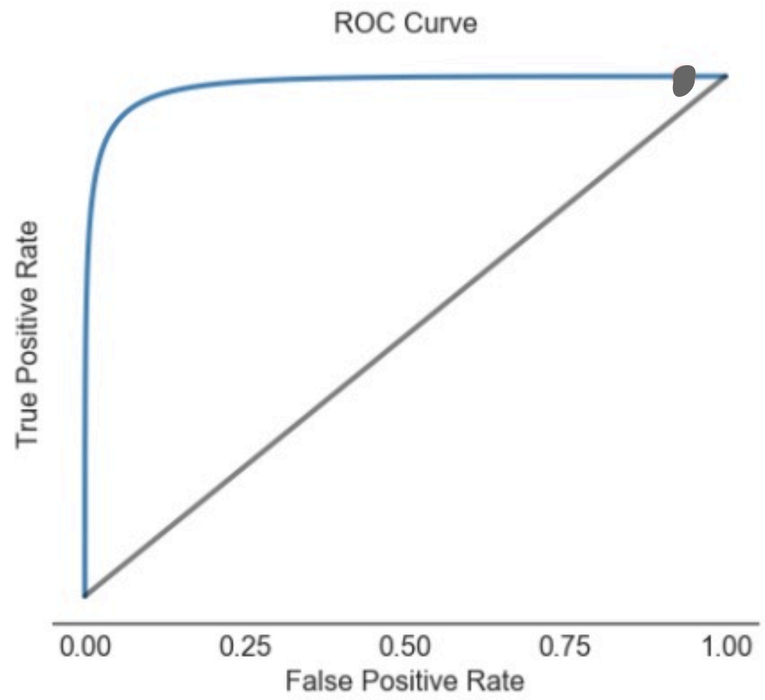
A ROC Curve is the plot of TPR (vertical) vs. FPR (horizontal) for all possible values of the threshold parameter

Extremely convenient to see how model would perform at all thresholds simultaneously, rather than looking at misclassification rate for thresholds individually

The ROC Curve

TPR = 1 FPR ≈ 1



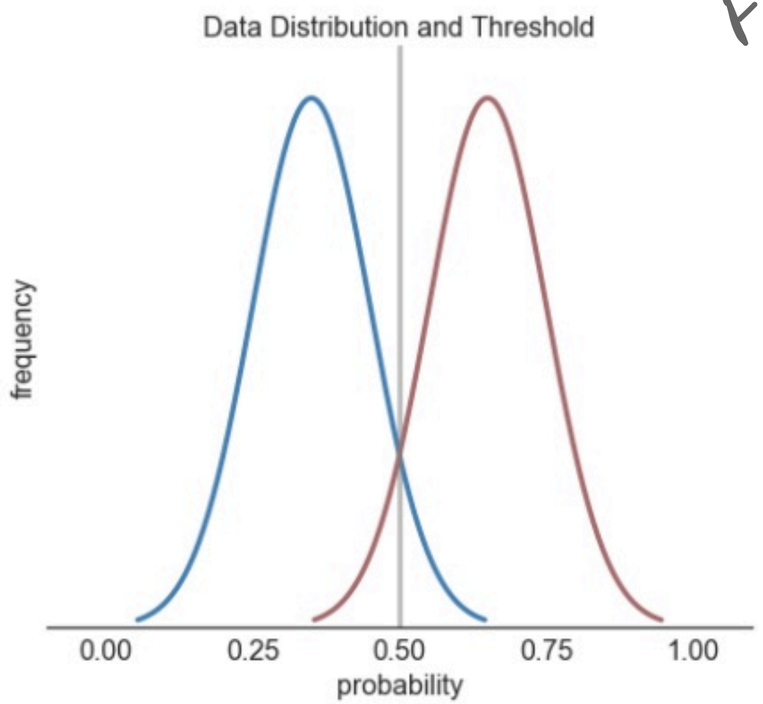


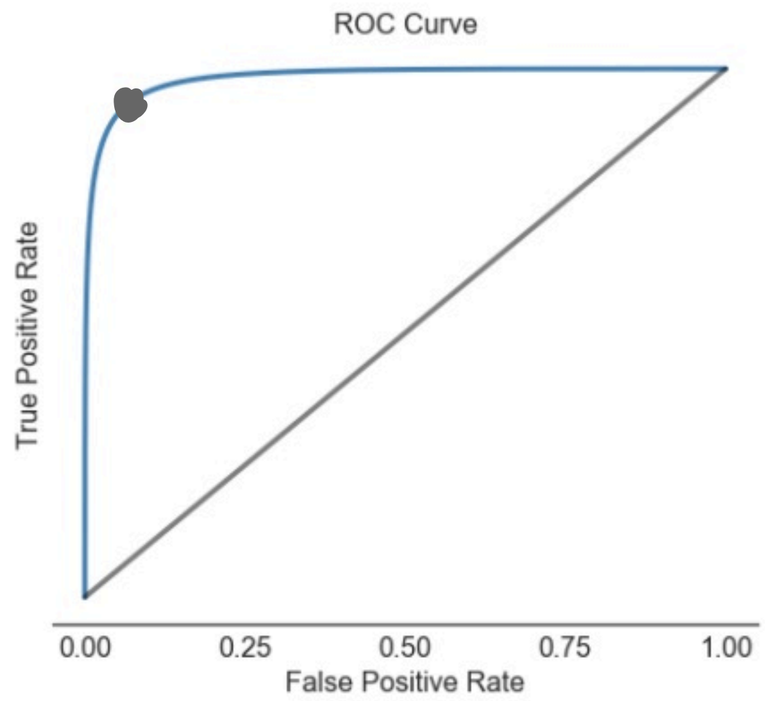
A ROC Curve is the plot of TPR (vertical) vs. FPR (horizontal) for all possible values of the threshold parameter

Extremely convenient to see how model would perform at all thresholds simultaneously, rather than looking at misclassification rate for thresholds individually

The ROC Curve

TPR = High
FPR = Low

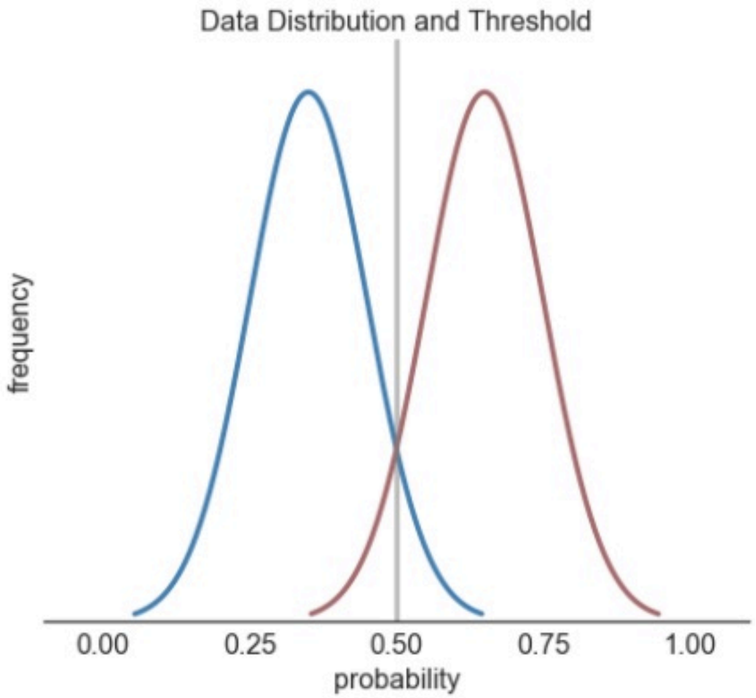


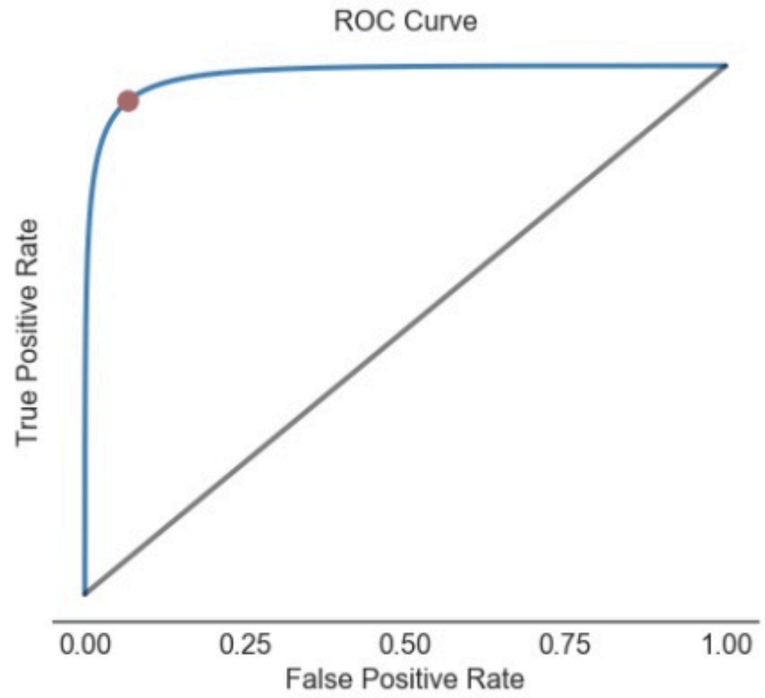


A ROC Curve is the plot of TPR (vertical) vs. FPR (horizontal) for all possible values of the threshold parameter

Extremely convenient to see how model would perform at all thresholds simultaneously, rather than looking at misclassification rate for thresholds individually

The ROC Curve

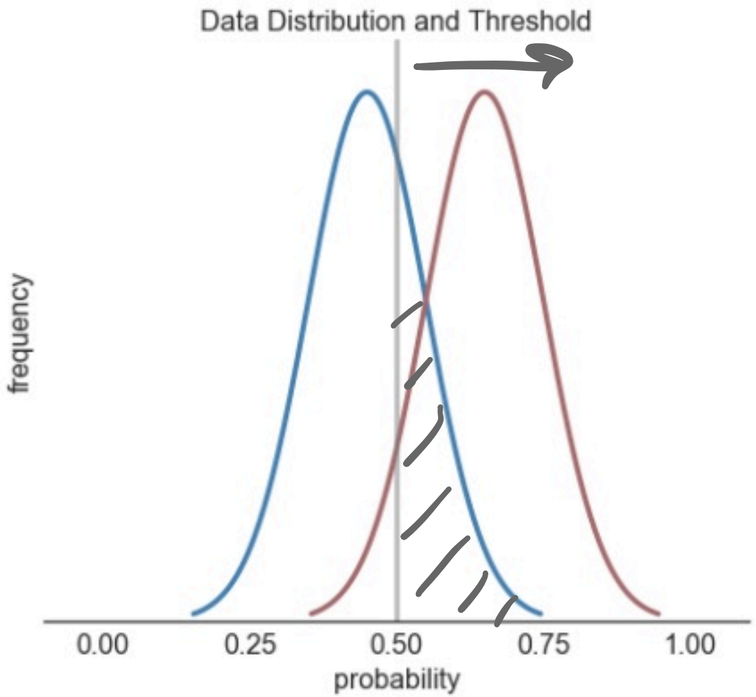


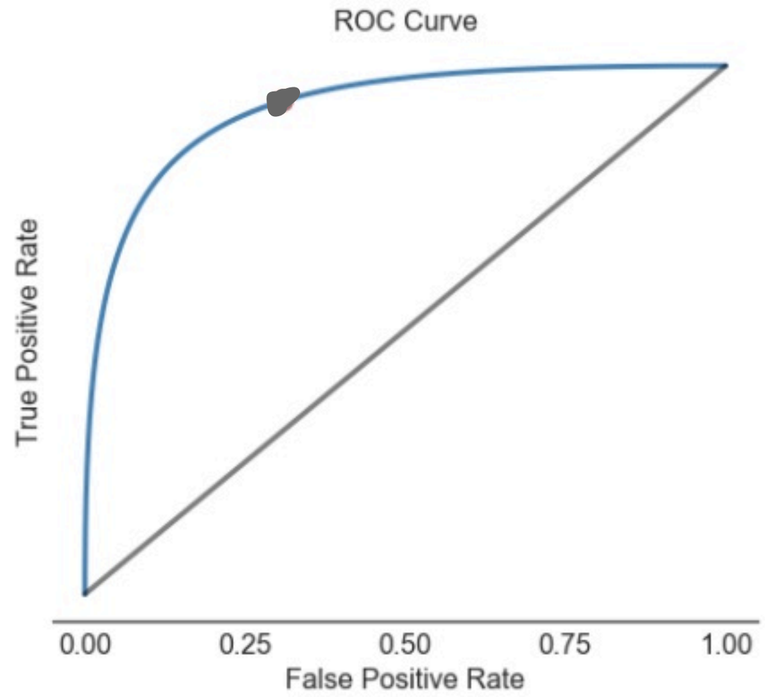


Notice that here, where we've separated the true classes very well, the ROC curve is very far away from the random chance line

What happens if we do slightly worse?

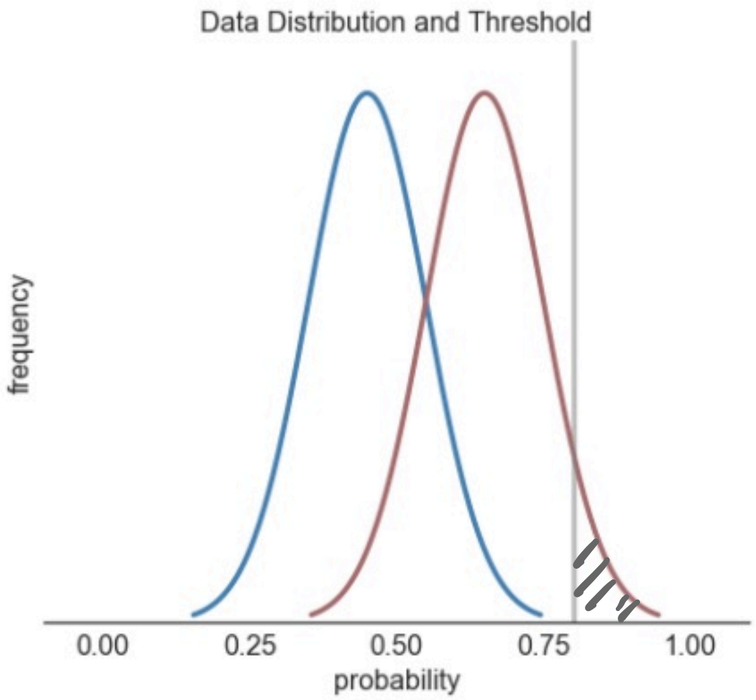
The ROC Curve

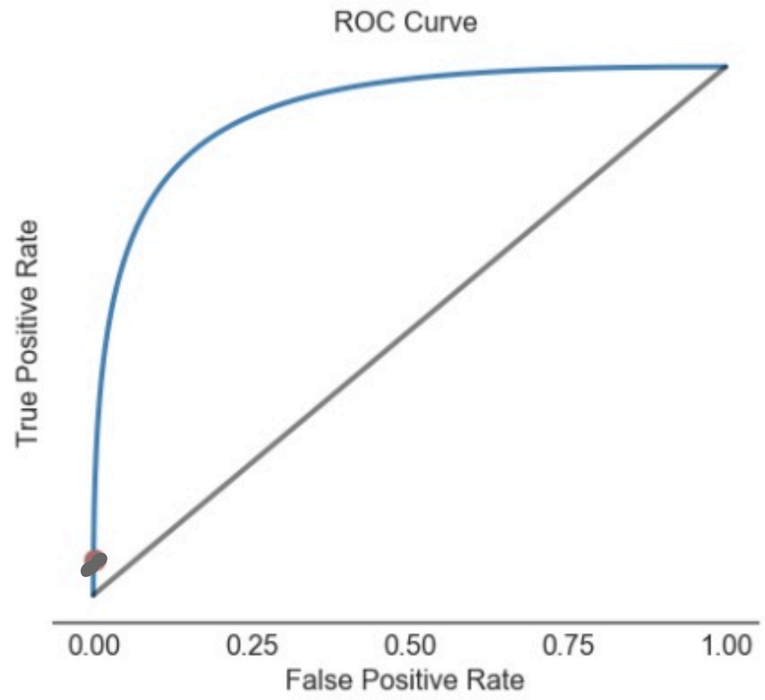




Now we're not doing as well. The ROC curve starts to bend towards the center.

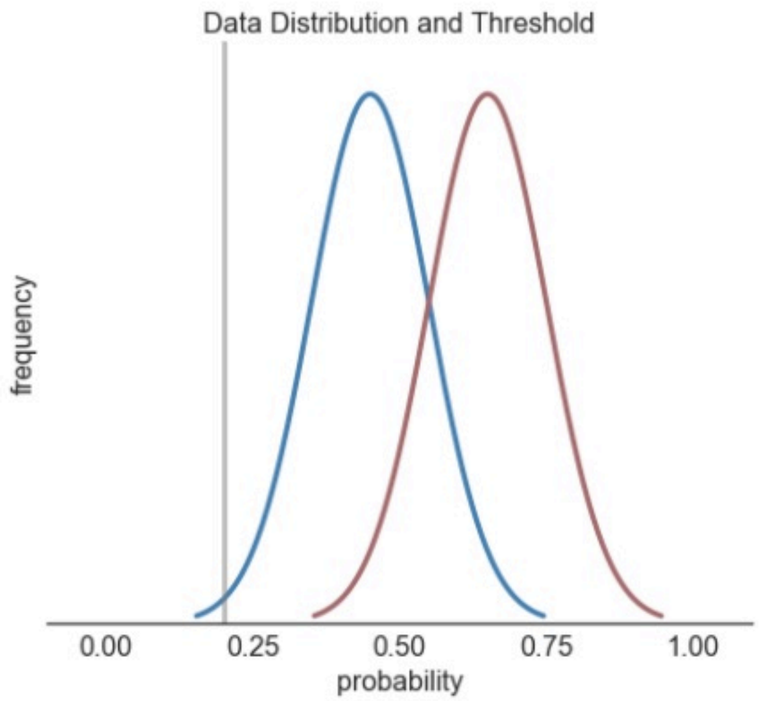
The ROC Curve

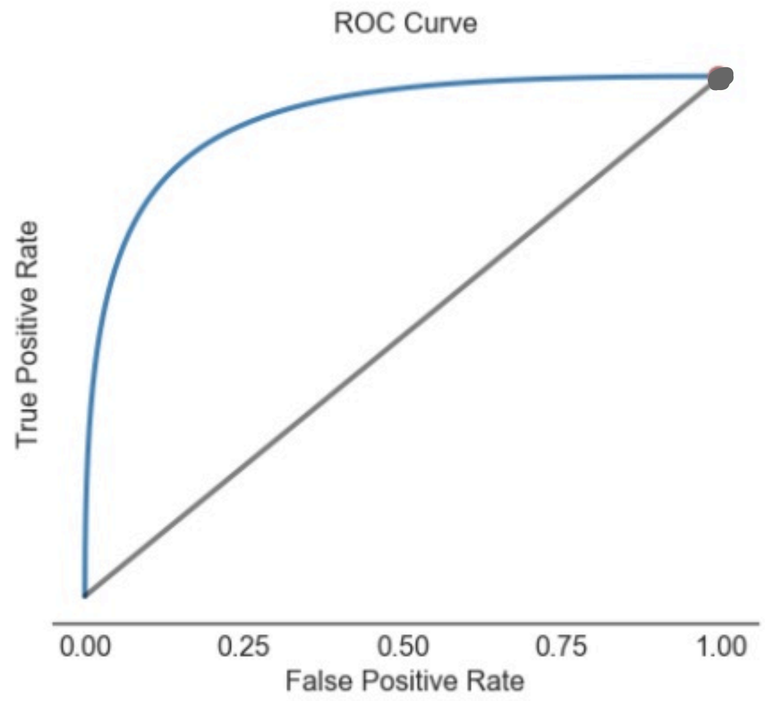




Now we're not doing as well. The ROC curve starts to bend towards the center.

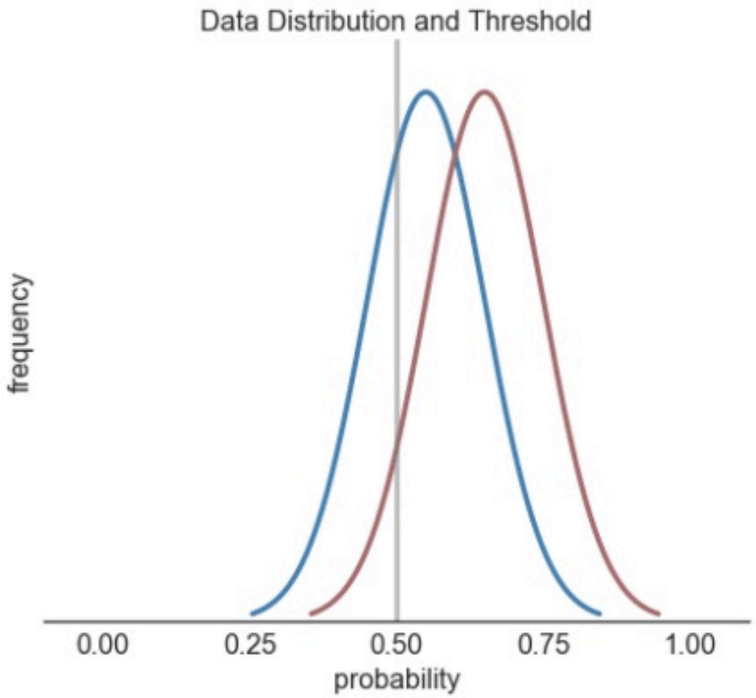
The ROC Curve

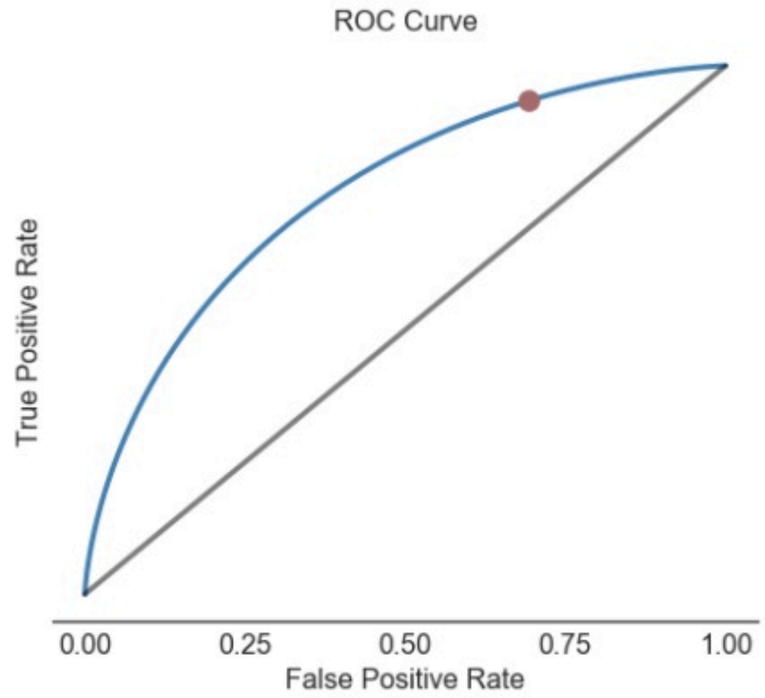




Now we're not doing as well. The ROC curve starts to bend towards the center.

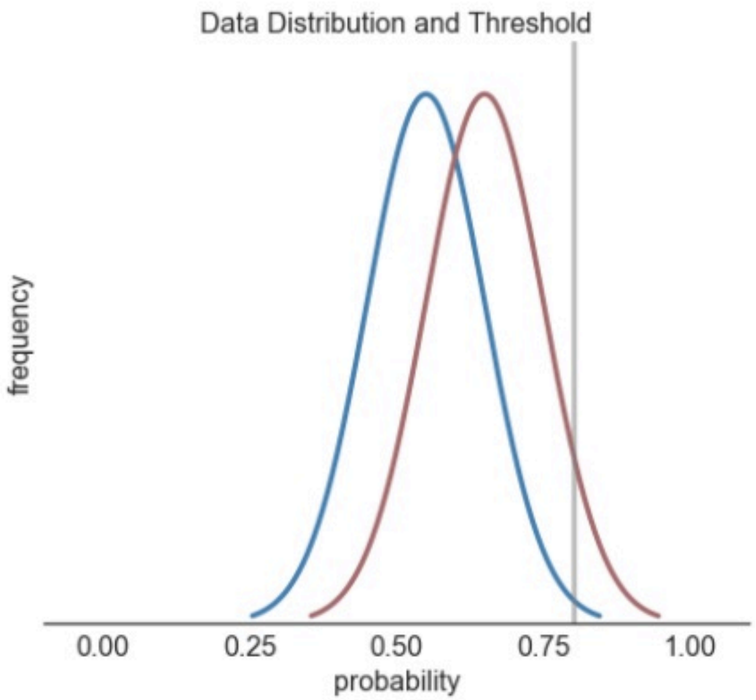
The ROC Curve

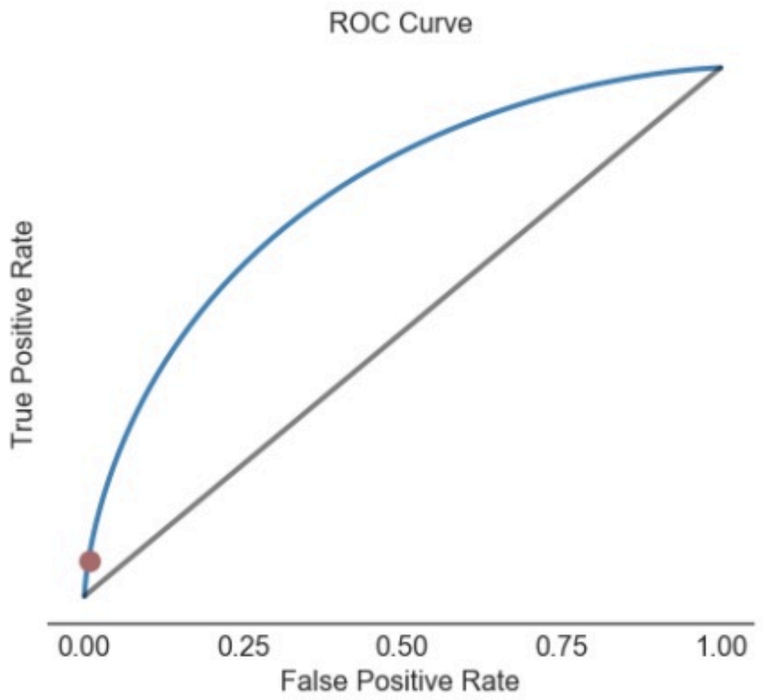




And as we do a poorer job of separating the classes ...

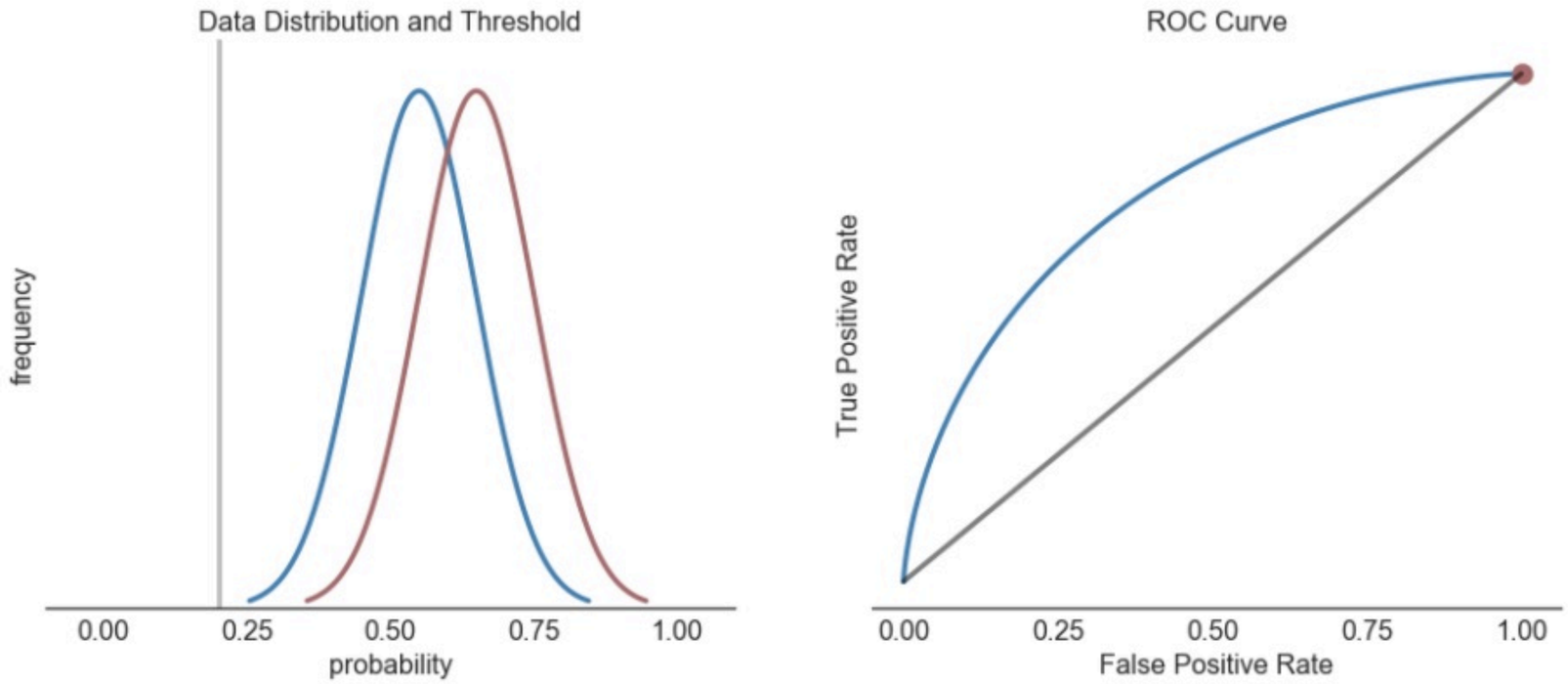
The ROC Curve





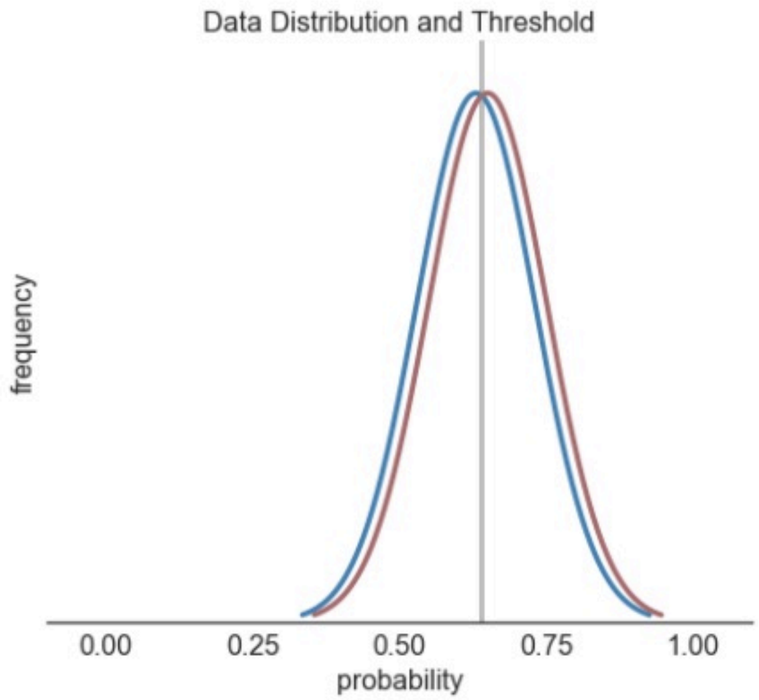
And as we do a poorer job of separating the classes ...

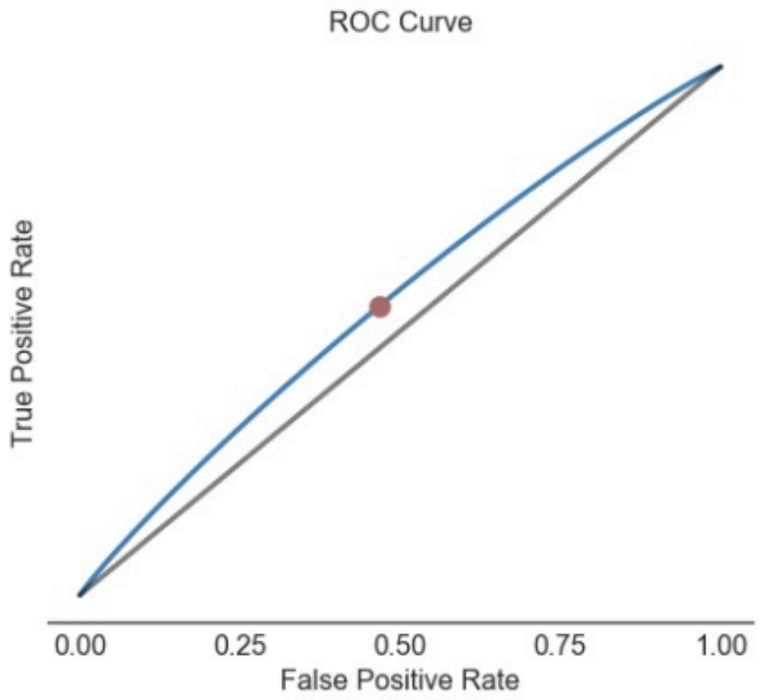
The ROC Curve



And as we do a poorer job of separating the classes ...

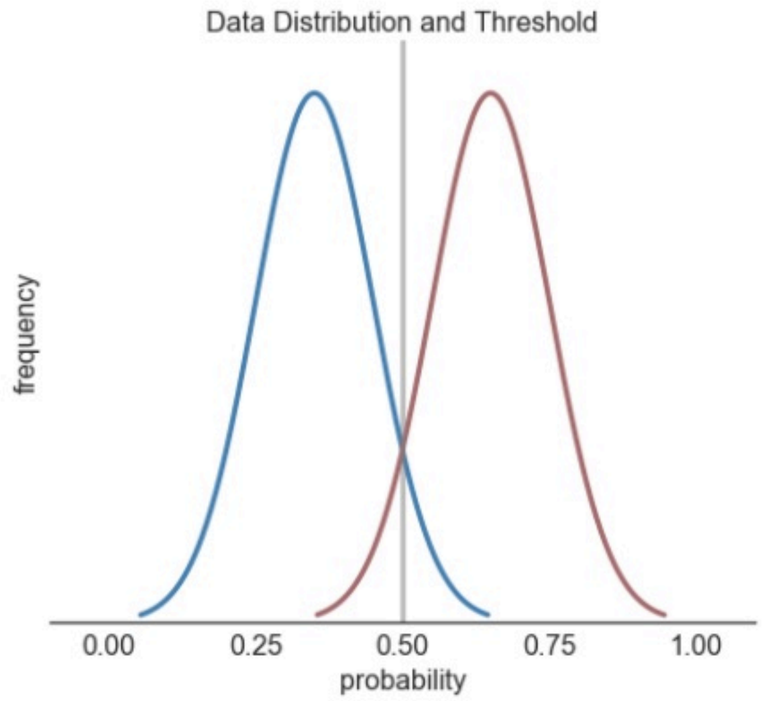
The ROC Curve

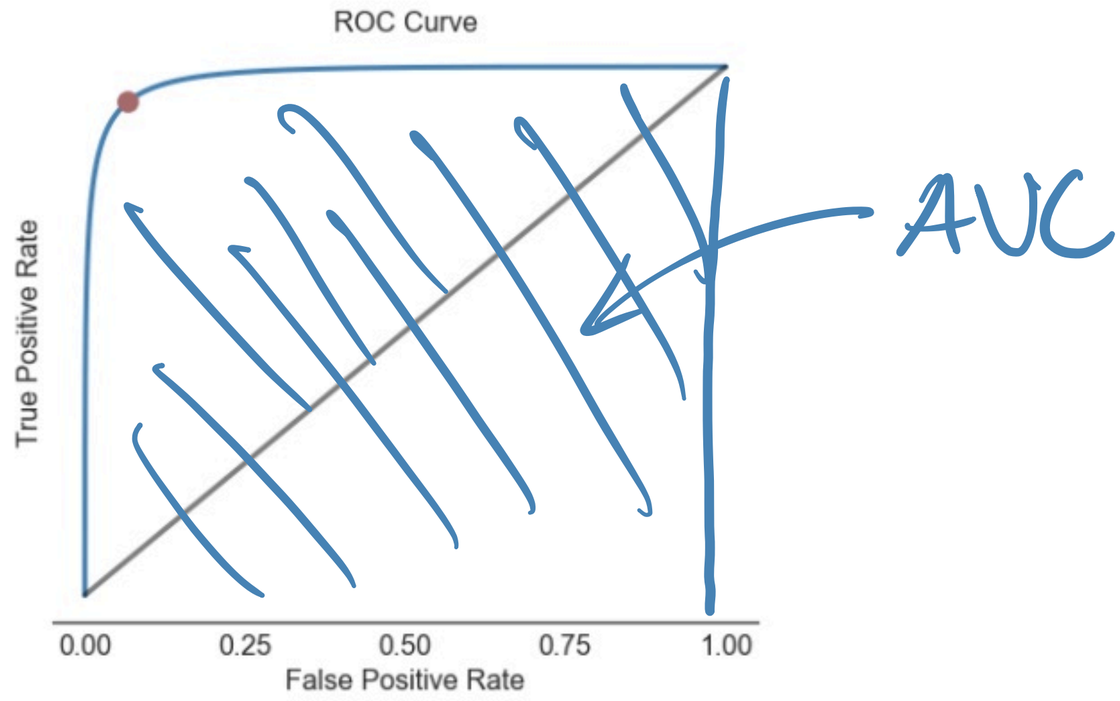




And when we do a terrible job, the ROC curve sits almost on top of the random chance line

The ROC Curve

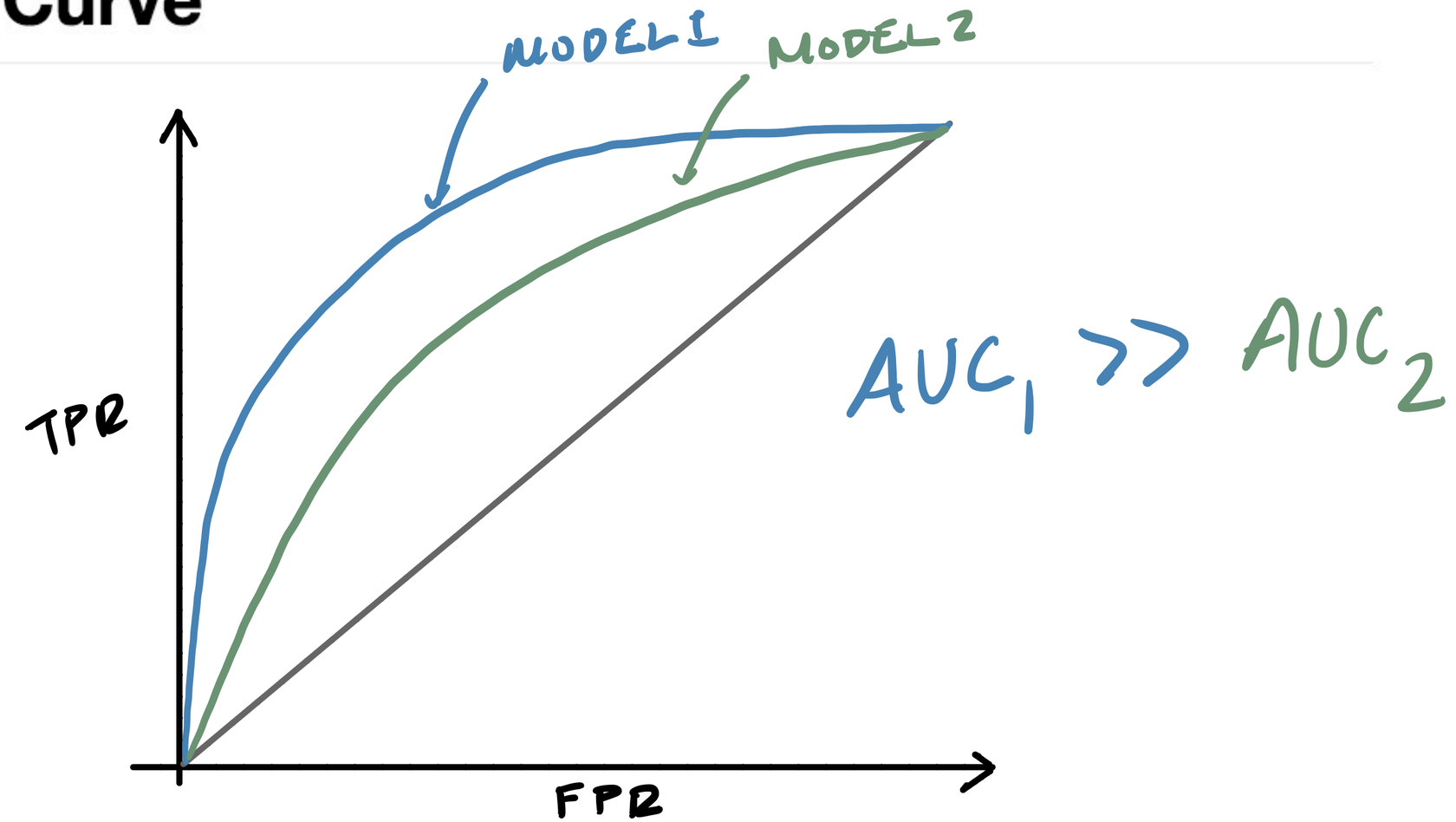




The ROC Curve allows us to analyze how well your classifier will do in settings when you're worried about False Positives and settings when you're worried about False Negatives simultaneously

If you prefer just one number, you can computed the AUC (area under the ROC curve)

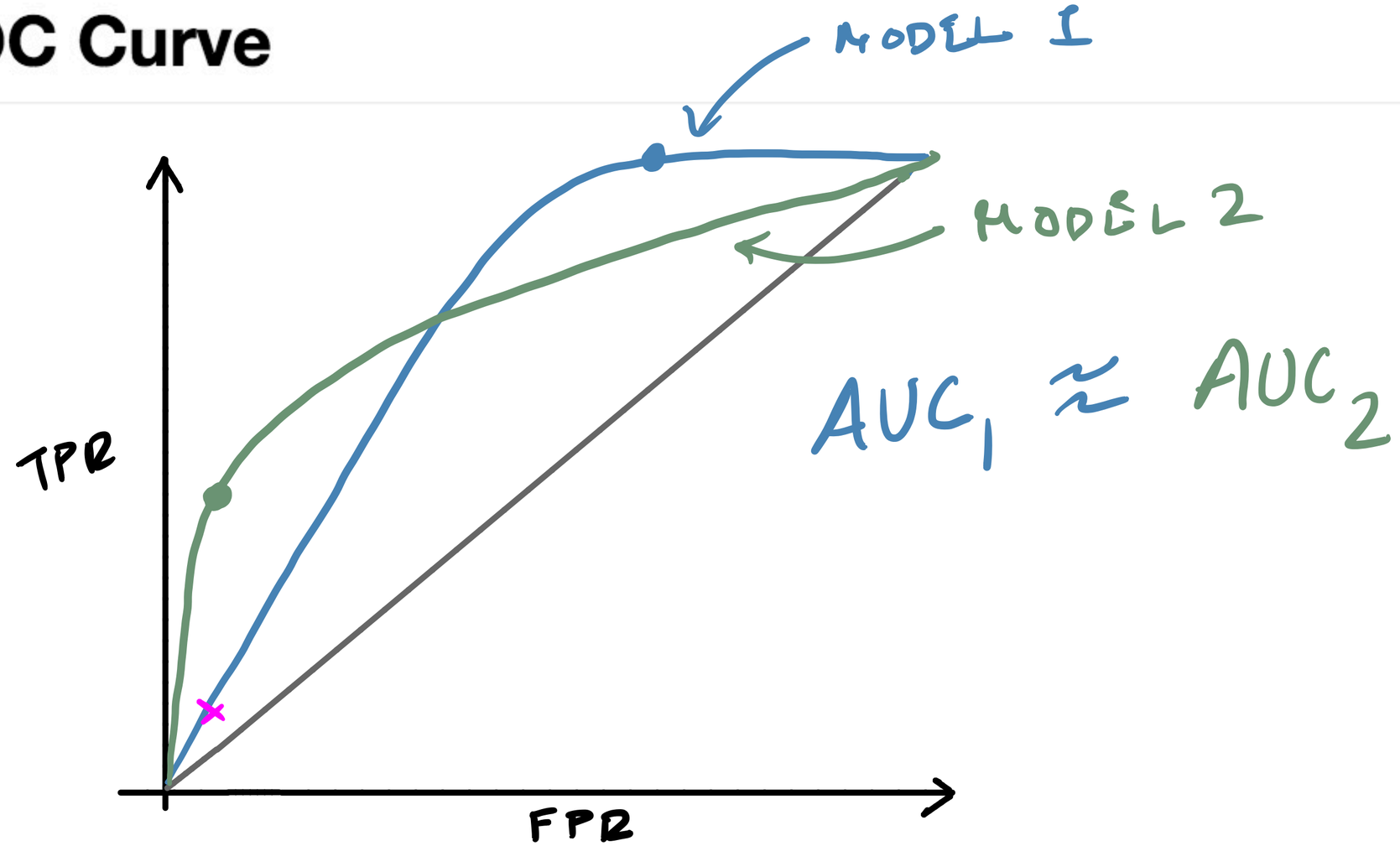
The ROC Curve



To compare two models, plot their ROCs them on the same axes

If one encloses the other, then it's better at both ends of the spectrum, and will also have a larger AUC

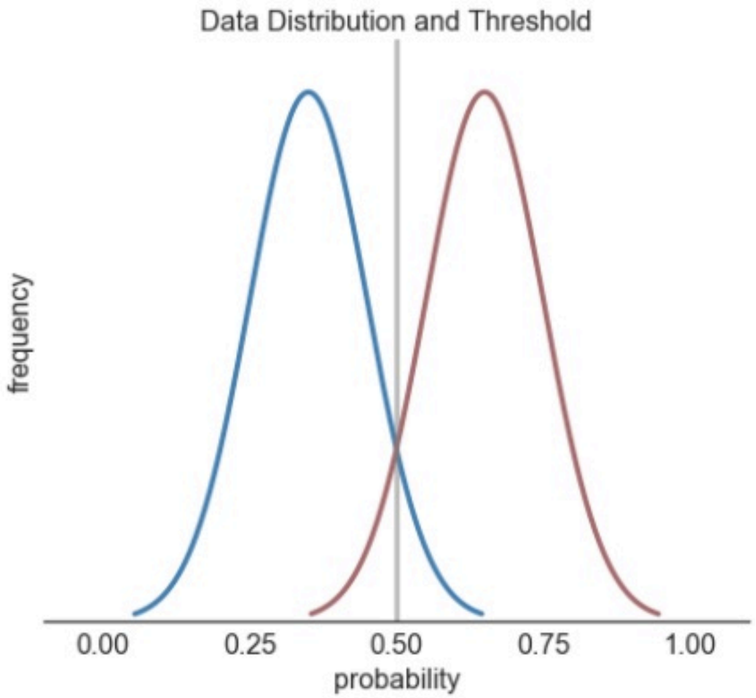
The ROC Curve

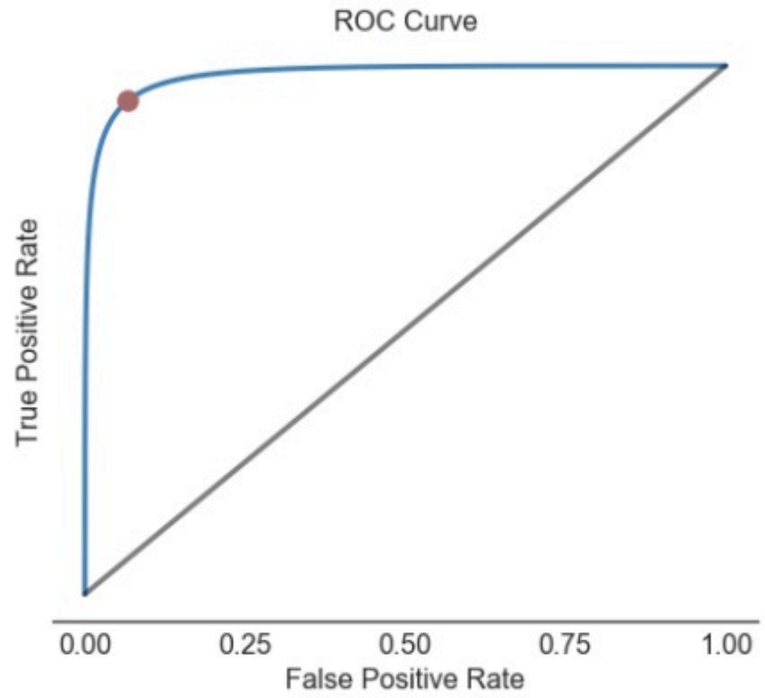


To compare two models, plot their ROCs them on the same axes

But if they have similar AUCs, they plot may show that one will do better on one end of the spectrum, and the other on other end

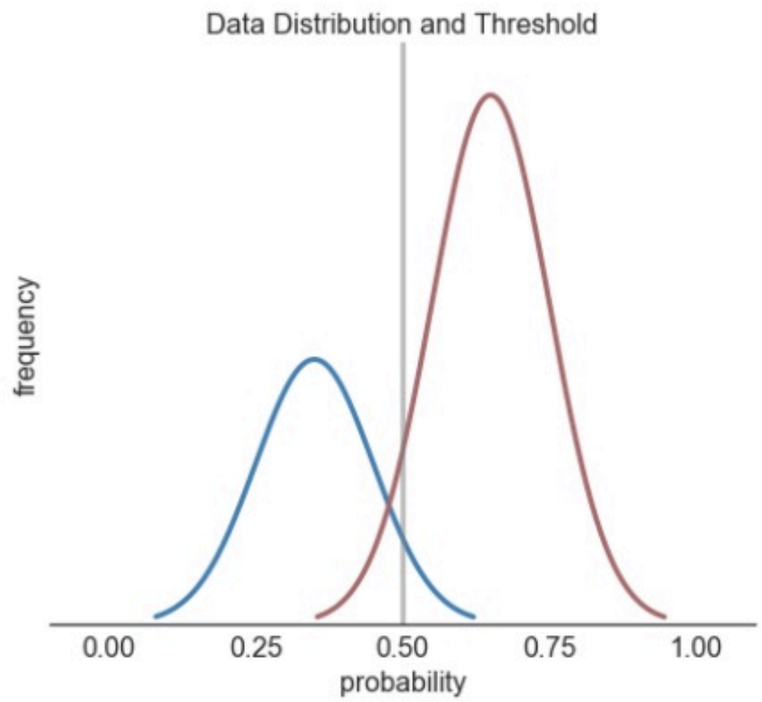
The ROC Curve

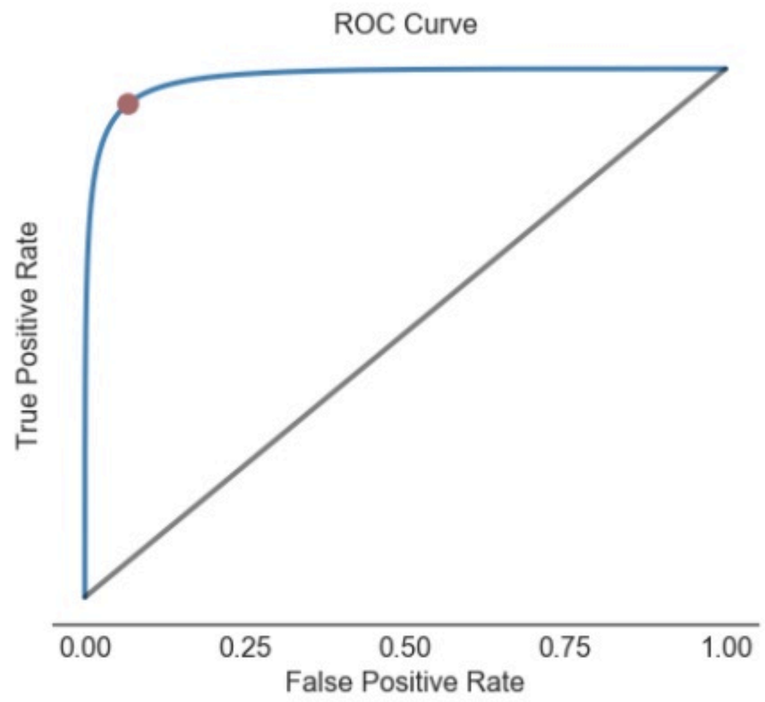




And here's my favorite part ...

The ROC Curve

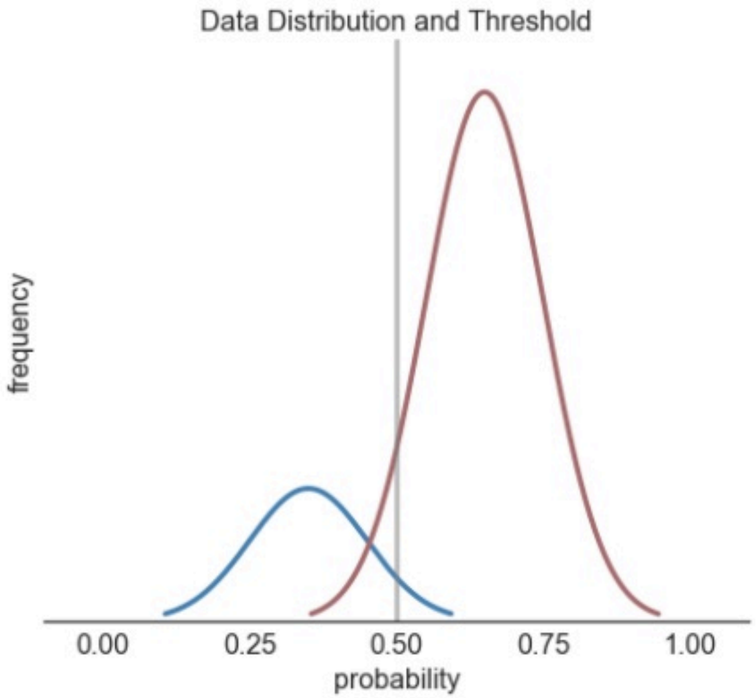


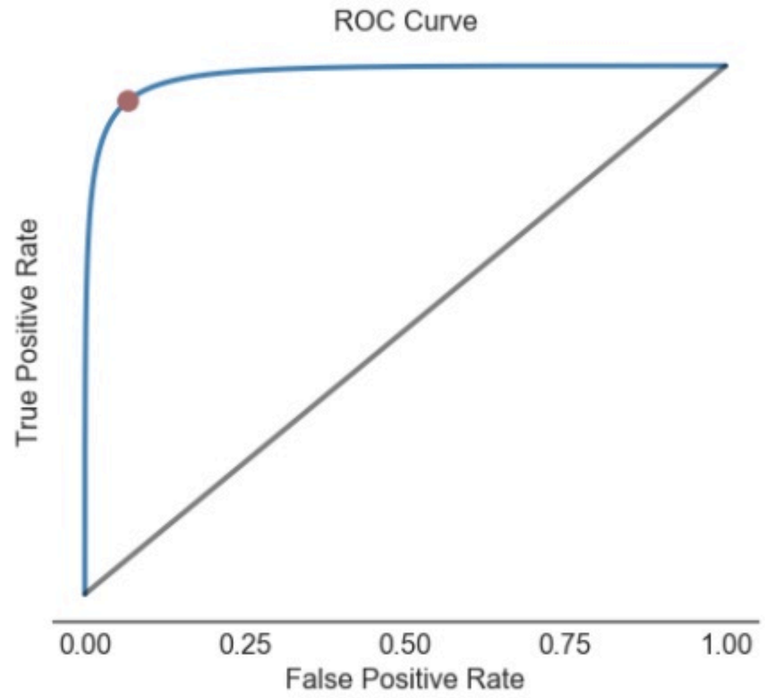


And here's my favorite part ...

Since it's based on proportions of pos-to-pos and neg-to-neg predictions, ROC is insensitive to skewed class imbalances

The ROC Curve





And here's my favorite part ...

Since it's based on proportions of pos-to-pos and neg-to-neg predictions, ROC is insensitive to skewed class imbalances

Evaluation Metrics

Take-Aways:

- Misclassification error / accuracy is unsatisfactory if you have imbalanced classes or care more about false positives or negatives
- The ROC curve and AUC don't suffer from these limitations
- ROC-AUC requires a binary classifier (eh, sorta)
- ROC-AUC requires a binary classifier that ranks predictions

- There are loads of other evaluation metrics out there (Precision vs. Recall, F-Scores)

In Class

- **Get in groups and get out laptops!**
- Work through some simple examples
- Cross-Validation and Accuracy metrics in Sklearn

Acknowledgements

The discussion on ROC was adopted from Kevin Markhom

Some of the figures in this presentation are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani

In-Class

In-Class

In-Class

In-Class

In-Class

In-Class

In-Class