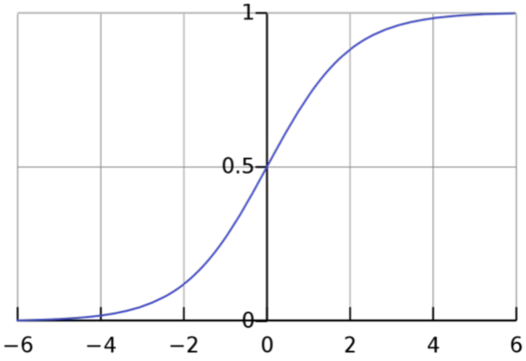
Tufts COMP 135: Introduction to Machine Learning

https://www.cs.tufts.edu/comp/135/2019s/

Binary Classification



Many slides attributable to:
Erik Sudderth (UCI)
Finale Doshi-Velez (Harvard)
James, Witten, Hastie, Tibshirani (ISL/ESL books)

Prof. Mike Hughes

Logistics

- Waitlist: We have some room, contact me

- HW2 due TONIGHT (Wed 2/6 at 11:59pm)
  - What you submit: PDF and zip
  - Please annotate pages in Gradescope!

- HW3 out later tonight, due a week from today
  - What you submit: PDF and zip
  - Please annotate pages in Gradescope!

- Next recitation is Mon 2/11
  - Practical binary classifiers in Python with sklearn
  - Numerical issues and how to address them

Objectives: Classifier Overview

- 3 steps of a classification task
  - Prediction
    - Making hard binary decisions
    - Predicting class probabilities
  - Training
  - Evaluation
    - Performance Metrics

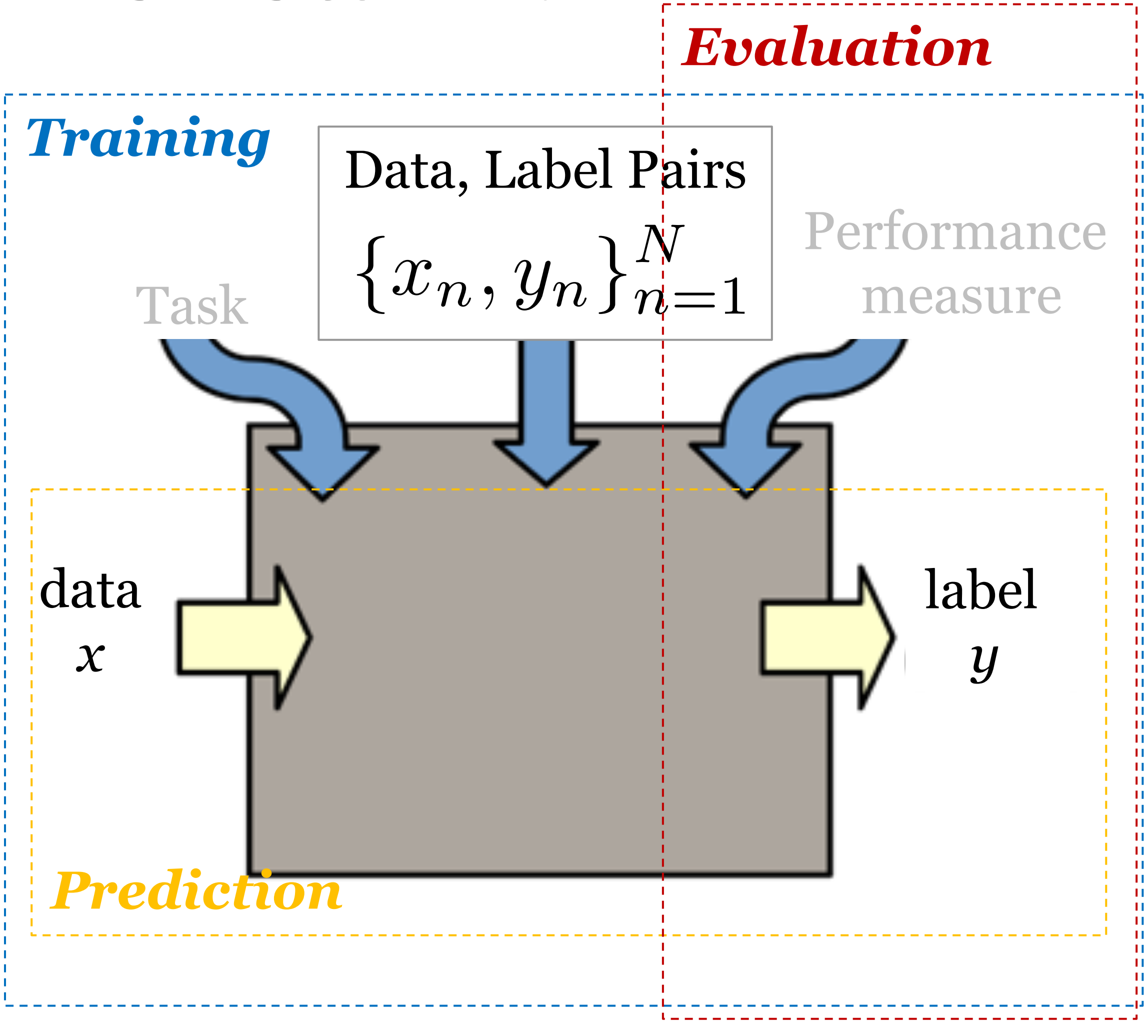
- A “taste” of 3 Methods
  - Logistic Regression
  - K-Nearest Neighbors
  - Decision Tree Regression

What will we learn?

Supervised Learning

Unsupervised Learning

Reinforcement Learning



Before: Regression

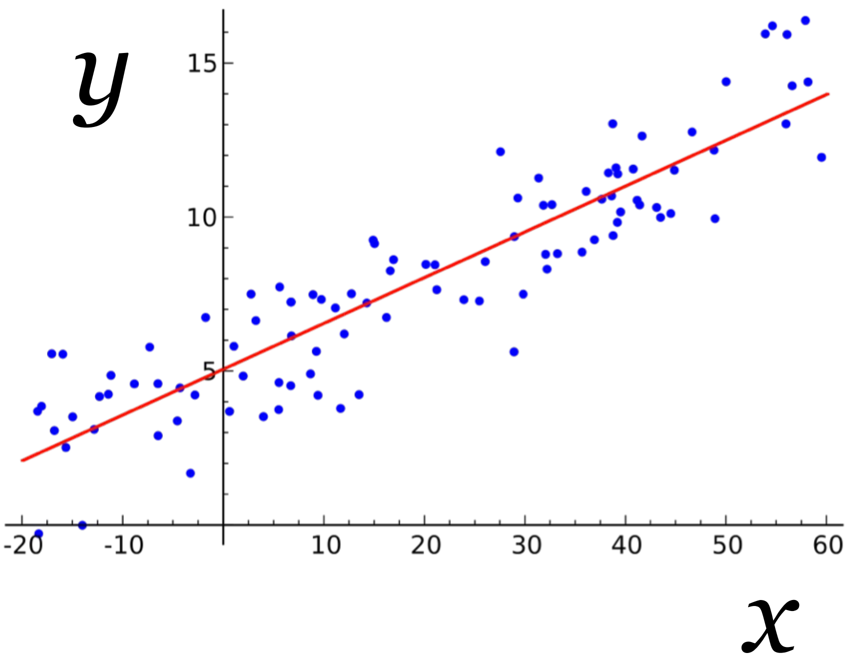
Supervised Learning

regression

Unsupervised Learning

Reinforcement Learning

y is a numeric variable e.g. sales in $$



Task: Binary Classification

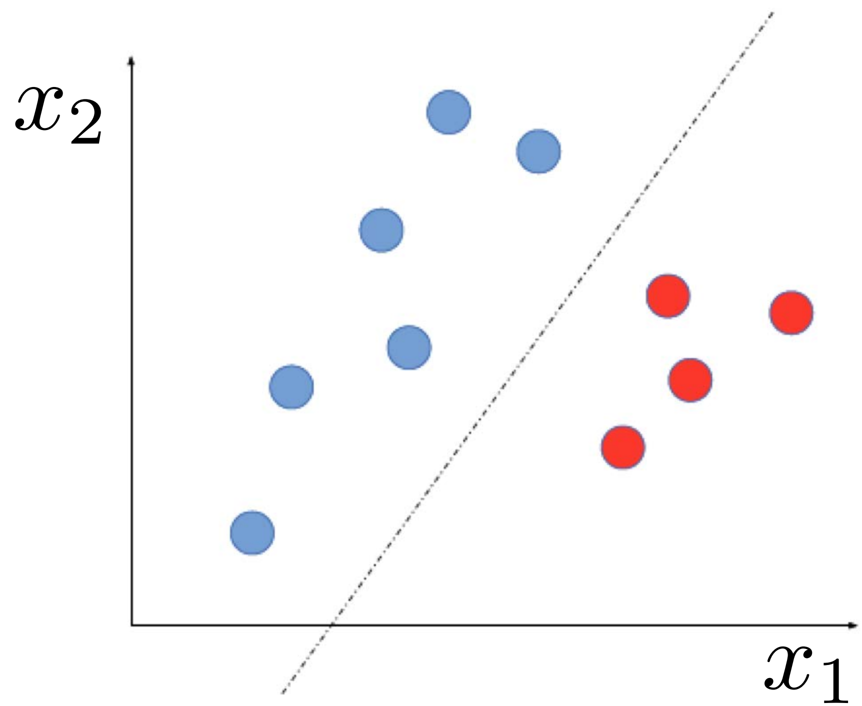
Supervised Learning

binary classification

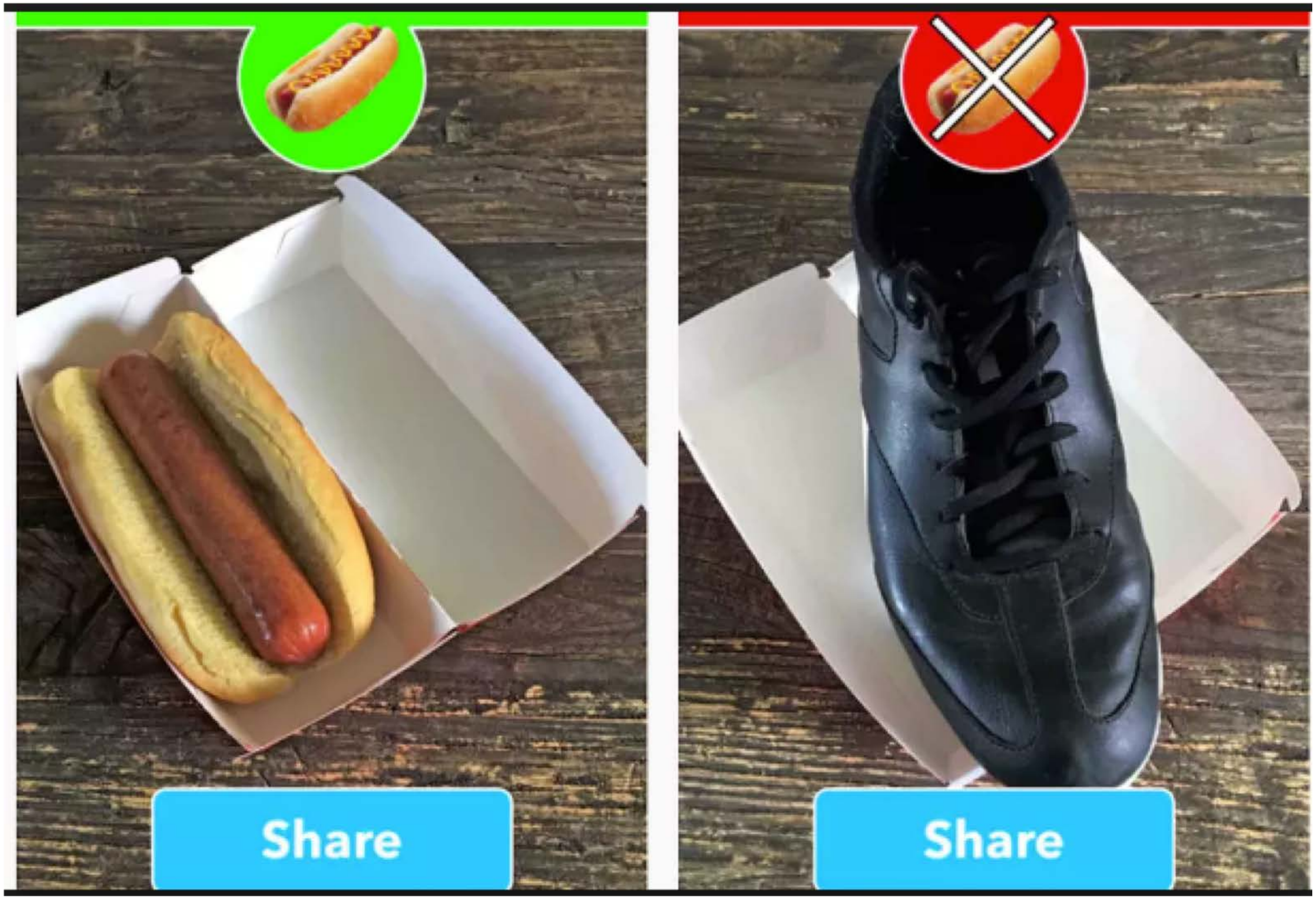
Unsupervised Learning

Reinforcement Learning

y is a binary variable (red or blue)



Example: Hotdog or Not



Task: Multi-class Classification

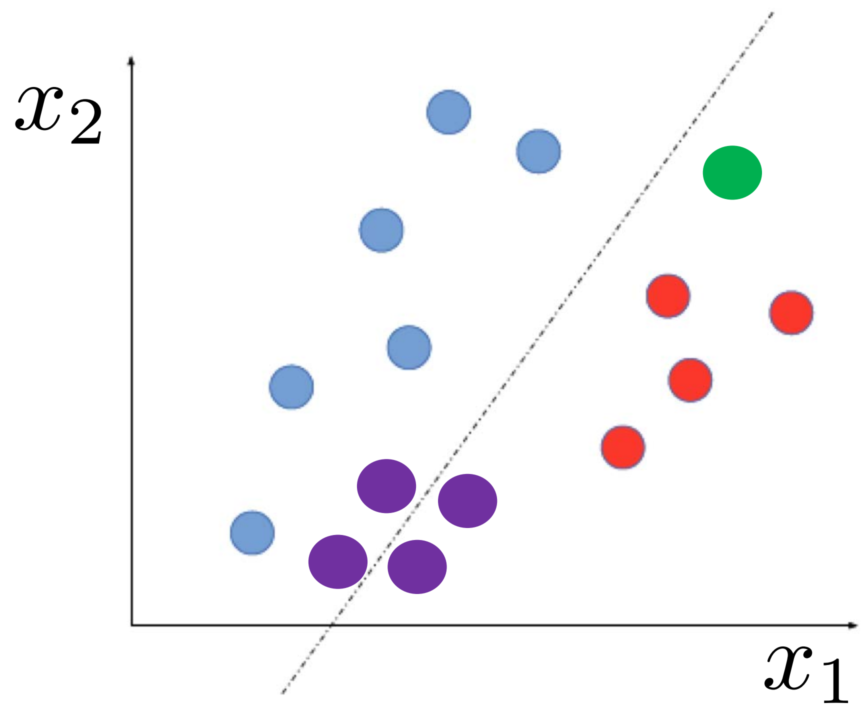
Supervised Learning

multi-class classification

Unsupervised Learning

Reinforcement Learning

y is a discrete variable
(red or blue or green or purple)



Classification Example: Swype

Predict words from keyboard trajectories

Novel Product:
An easier way
to input text on
mobile devices

Many possible letters: *Multi-class* classification

Binary Prediction Step

Goal: Predict label (0 or 1) given features x

- Input: $x_i \triangleq [x_{i1}, x_{i2}, \ldots x_{if} \ldots x_{iF}]$

  "features"
  "covariates"
  "predictors"
  "attributes"

  Entries can be real-valued, or other numeric types (e.g. integer, binary)

- Output: $y_i \in \{0, 1\}$

  "responses"
  "labels"

  Binary label (0 or 1)

Binary Prediction Step

```
>>> # Given: pretrained regression object model
>>> # Given: 2D array of features x

>>> x_NF.shape
(N, F)

>>> yhat_N = model.predict(x_NF)
>>> yhat_N[:5] # peek at predictions
[0, 0, 1, 0, 1]

>>> yhat_N.shape
(N,)
```

Types of binary predictions

TN : true negative
FN : false negative

FP : false positive
TP : true positive

		classifier calls	
		"negative" C=0	"positive" C=1
true outcome	Y=0	TN	FP
	Y=1	FN	TP

Example:

Which outcome is this?

		classifier calls	
		 "negative" C=0	"positive" C=1 
true outcome	Y=0	TN	FP
	Y=1	FN	TP

TN : true negative

FN : false negative

FP : false positive

TP : true positive

Example:

Which outcome is this?

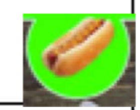
Answer:
True Positive

		classifier calls	
		"negative" C=0	"positive" C=1
true outcome	Y=0	TN	FP
	Y=1	FN	TP

TN : true negative
FN : false negative
FP : false positive
TP : true positive

Example:

Which outcome is this?

		classifier calls	
		 "negative" C=0	"positive" C=1 
true outcome	Y=0	TN	FP
	Y=1	FN	TP

TN : true negative

FN : false negative

FP : false positive

TP : true positive

Example:

Which outcome is this?

Answer:
True Negative (TN)

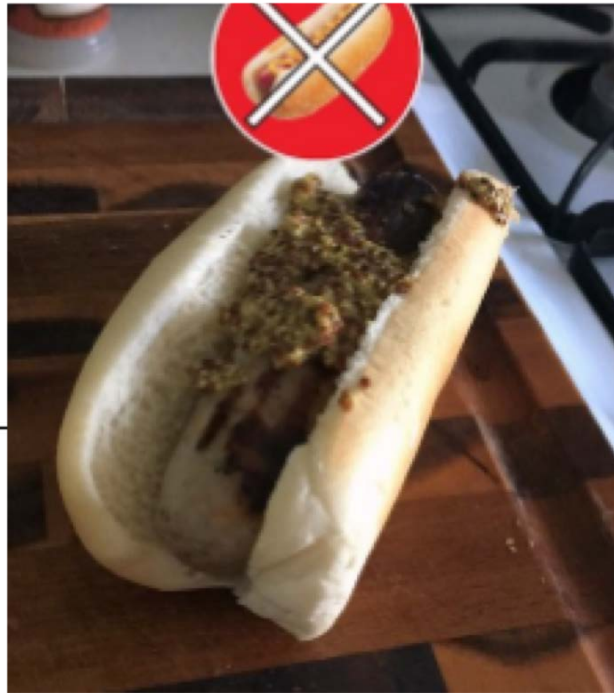
		classifier calls	
		 “negative” C=0	“positive” C=1 
true outcome	Y=0	TN	FP
	Y=1	FN	TP

TN : true negative
FN : false negative

FP : false positive
TP : true positive

Example:

Which outcome is this?



		classifier calls	
		 "negative" C=0	"positive" C=1 
true outcome	Y=0	TN	FP
	Y=1	FN	TP

TN : true negative

FN : false negative

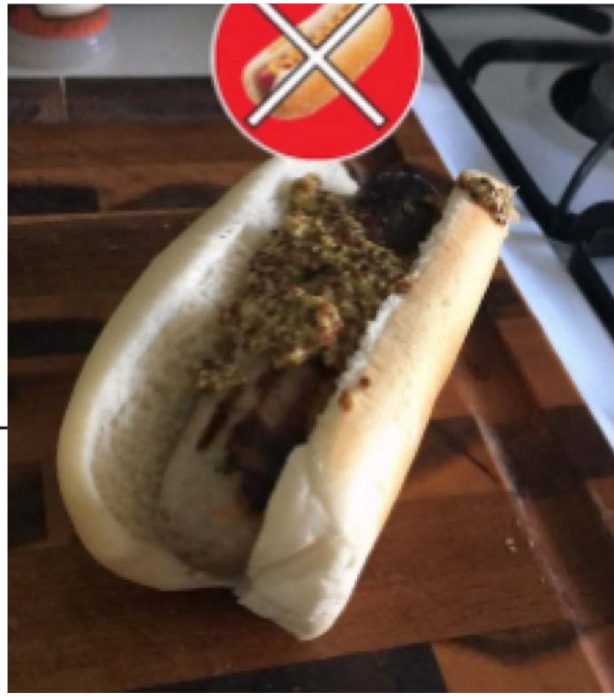
FP : false positive

TP : true positive

Example:

Which outcome is this?

Answer:
False Negative (FN)



		classifier calls	
		 “negative” C=0	“positive” C=1 
true outcome	Y=0	TN	FP
	Y=1	FN	TP

TN : true negative
FN : false negative
FP : false positive
TP : true positive

Example:

Which outcome is this?

		classifier calls	
		 "negative" C=0	"positive" C=1 
true outcome	Y=0	TN	FP
	Y=1	FN	TP

TN : true negative
FN : false negative
FP : false positive
TP : true positive

Example:

Which outcome is this?

Answer:
False Positive (FP)

		classifier calls	
		 "negative" C=0	"positive" C=1 
true outcome	Y=0	TN	FP
	Y=1	FN	TP

TN : true negative
FN : false negative

FP : false positive
TP : true positive

Probability Prediction Step

Goal: Predict probability p(Y=1) given features x

- Input: $x_i \triangleq [x_{i1}, x_{i2}, \ldots x_{if} \ldots x_{iF}]$

  "features"
  "covariates"
  "predictors"
  "attributes"

  Entries can be real-valued, or other numeric types (e.g. integer, binary)

- Output: $\hat{p}_i$

  "probabilities"

  Probability between 0 and 1
  e.g. 0.001, 0.513, 0.987

Probability Prediction Step

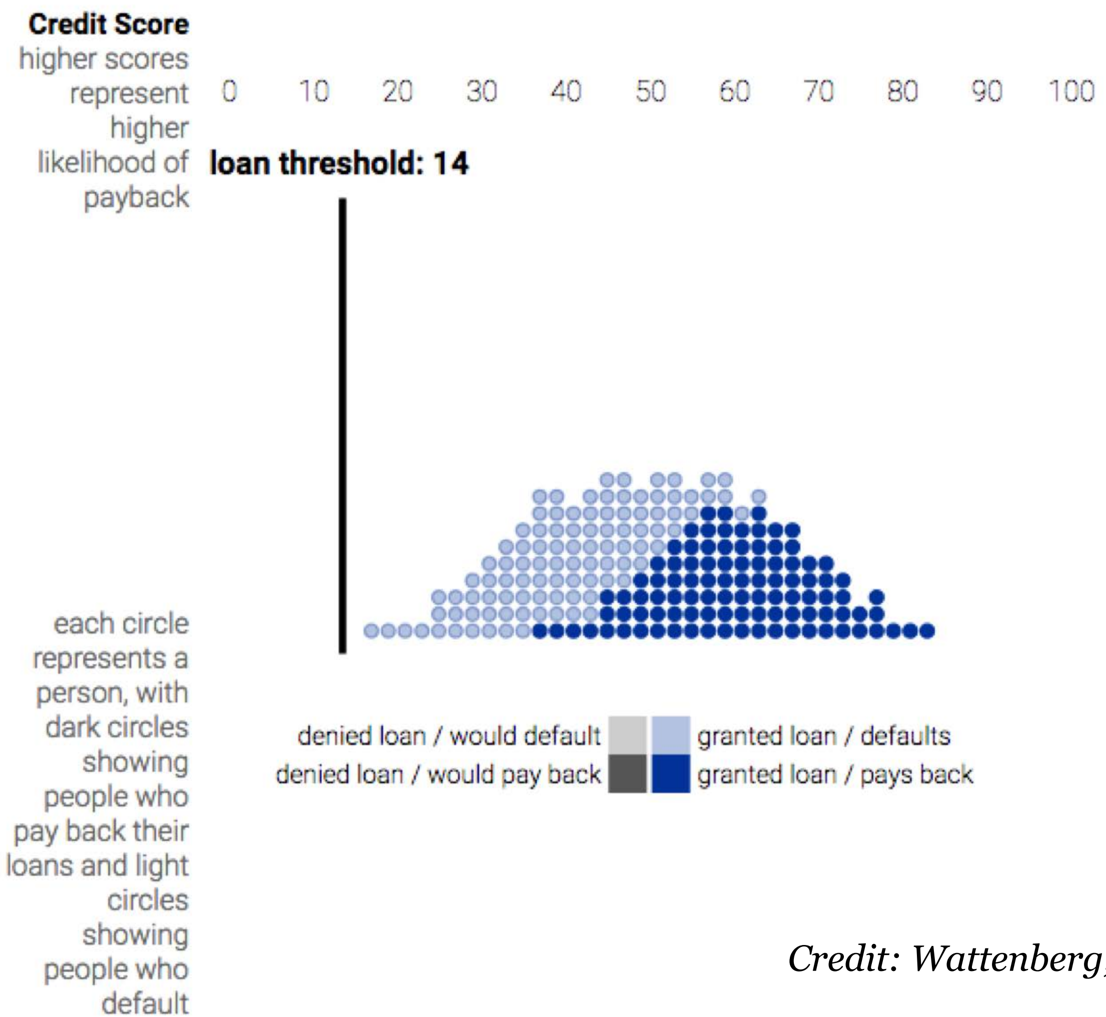
```
>>> # Given: pretrained regression object model
>>> # Given: 2D array of features x

>>> x_NF.shape
(N, F)
```

```
>>> yproba_N2 = model.predict_proba(x_NF)
>>> yproba_N2.shape
(N, 2)
```

Column index 1 gives probability of positive label $p(Y = 1)$

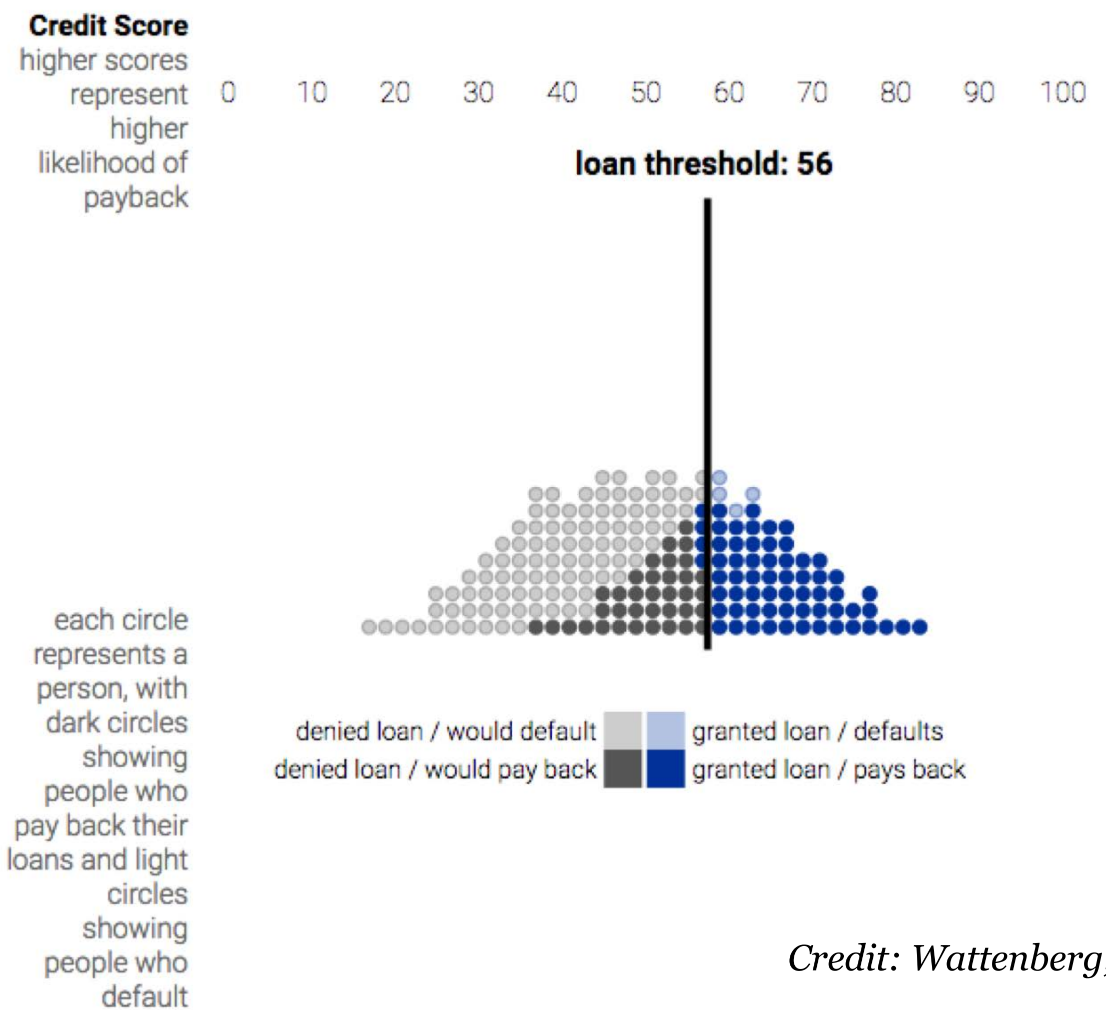
```
>>> yproba_N2[:, 1]
[0.003, 0.358, 0.987, 0.111, 0.656]
```

Thresholding to get Binary Decisions



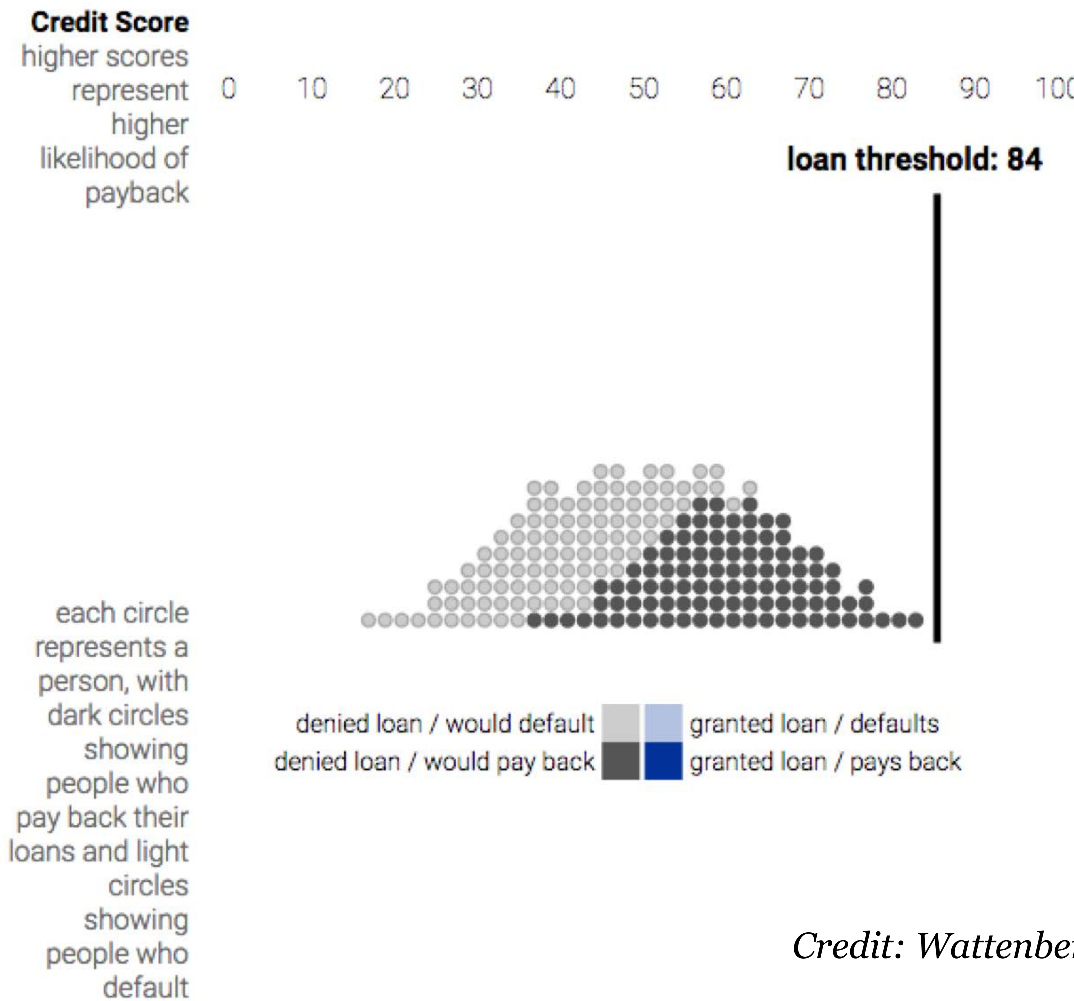
Credit: Wattenberg, Viégas, Hardt

Thresholding to get Binary Decisions



Credit: Wattenberg, Viégas, Hardt

Thresholding to get Binary Decisions



Credit: Wattenberg, Viégas, Hardt

Pair Exercise

[Interactive Demo:](https://research.google.com/bigpicture/attacking-discrimination-in-ml/)

https://research.google.com/bigpicture/attacking-discrimination-in-ml/

Loan and pay back: +$300

Loan and not pay back: -$700

Goals:

- What threshold maximizes accuracy?
- What threshold maximizes profit?
- What needs to be true of costs so threshold is the same for profit and accuracy?

Classifier: Training Step

Goal: Given a labeled dataset, learn a **function** that can perform prediction well

- Input: Pairs of features and labels/responses

$$\{x_n, y_n\}_{n=1}^N$$

- Output: $\hat{y}(\cdot) : \mathbb{R}^F \rightarrow \{0, 1\}$

Useful to break into two steps:
1) *Produce probabilities in [0, 1] OR real-valued scores*
2) *Threshold to make binary decisions*

Classifier: Training Step

```
>>> # Given: 2D array of features x
>>> # Given: 1D array of binary labels y

>>> y_N.shape
(N,)
>>> x_NF.shape
(N, F)

>>> model = BinaryClassifier()
>>> model.fit(x_NF, y_N)
>>> # Now can call predict or predict_proba
```

Classifier: **Evaluation Step**

Goal: Assess quality of predictions

Many ways in practice:

1) Evaluate probabilities / scores directly

    logistic loss, hinge loss, …

2) Evaluate binary decisions at specific threshold

    accuracy, TPR, TNR, PPV, NPV, etc.

3) Evaluate across range of thresholds

    ROC curve, Precision-Recall curve

Metric: Confusion Matrix
Counting mistakes in binary predictions

#TN : num. true negative
#FN : num. false negative

#TP : num. true positive
#FP : num. false positive

		classifier calls	
		"negative" C=0	"positive" C=1
true outcome	Y=0	#TP	#FP
	Y=1	#FN	#TP

Metric: Accuracy

accuracy = fraction of correct predictions

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

Potential problem:

Suppose your dataset has 1 positive example and 99 negative examples

What is the accuracy of the classifier that always predicts ”negative”?

Metric: Accuracy

accuracy = fraction of correct predictions

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

Potential problem:

Suppose your dataset has 1 positive example and 99 negative examples

What is the accuracy of the classifier that always predicts "negative"?

99%!

Metrics for Binary Decisions

METRIC	FORMULA	IN WORDS **"Probability that …"** **Or** **"How often the …"**	EXPRESSION
True Positive Rate (TPR) *"sensitivity", "recall"*	$\frac{TP}{TP + FN}$	subject who is positive will be called positive	Pr(C = 1 | Y = 1)
True Negative Rate (TNR) *"specificity", 1 - FPR*	$\frac{TN}{FP + TN}$	subject who is negative will be called negative	Pr(C = 0 | Y = 0)
Positive Predictive Value (PPV) *"precision"*	$\frac{TP}{TP + FP}$	subject called positive will actually be positive	Pr(Y = 1 | C = 1)
Negative Predictive Value (NPV)	$\frac{TN}{TN + FN}$	subject called negative will actually be negative	Pr(Y = 0 | C = 0)

Emphasize the metrics appropriate for your application.

Goal: App to classify cats vs. dogs from images

Which metric might be most important? Could we just use accuracy?

METRIC	FORMULA	IN WORDS **"Probability that …"** **Or** **"How often the …"**	EXPRESSION
True Positive Rate (TPR)	$\frac{TP}{TP + FN}$	subject who is positive will be called positive	Pr(C = 1 | Y = 1)
True Negative Rate (TNR)	$\frac{TN}{FP + TN}$	subject who is negative will be called negative	Pr(C = 0 | Y = 0)
Positive Predictive Value (PPV)	$\frac{TP}{TP + FP}$	subject called positive will actually be positive	Pr(Y = 1 | C = 1)
Negative Predictive Value (NPV)	$\frac{TN}{TN + FN}$	subject called negative will actually be negative	Pr(Y = 0 | C = 0)

Goal: Classifier to find relevant tweets to list on website

Which metric might be most important? Could we just use accuracy?

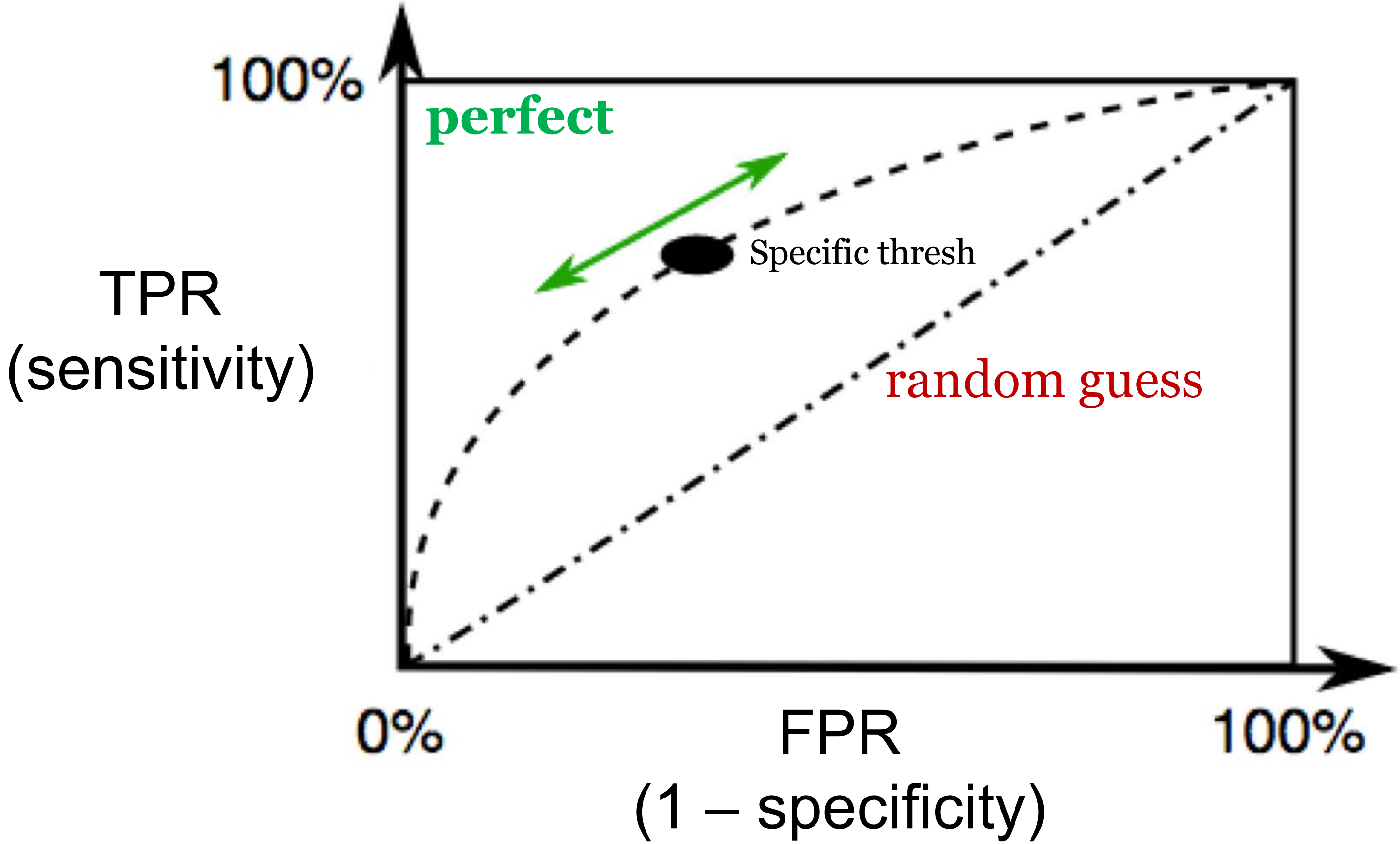
METRIC	FORMULA	IN WORDS **"Probability that …"** **Or** **"How often the …"**	EXPRESSION
True Positive Rate (TPR)	$\frac{TP}{TP + FN}$	subject who is positive will be called positive	Pr(C = 1 | Y = 1)
True Negative Rate (TNR)	$\frac{TN}{FP + TN}$	subject who is negative will be called negative	Pr(C = 0 | Y = 0)
Positive Predictive Value (PPV)	$\frac{TP}{TP + FP}$	subject called positive will actually be positive	Pr(Y = 1 | C = 1)
Negative Predictive Value (NPV)	$\frac{TN}{TN + FN}$	subject called negative will actually be negative	Pr(Y = 0 | C = 0)

Goal: Detector for tumors based on medical image

Which metric might be most important? Could we just use accuracy?

METRIC	FORMULA	IN WORDS **"Probability that …"** **Or** **"How often the …"**	EXPRESSION
True Positive Rate (TPR)	TP / (TP + FN)	subject who is positive will be called positive	Pr(C = 1 | Y = 1)
True Negative Rate (TNR)	TN / (FP + TN)	subject who is negative will be called negative	Pr(C = 0 | Y = 0)
Positive Predictive Value (PPV)	TP / (TP + FP)	subject called positive will actually be positive	Pr(Y = 1 | C = 1)
Negative Predictive Value (NPV)	TN / (TN + FN)	subject called negative will actually be negative	Pr(Y = 0 | C = 0)

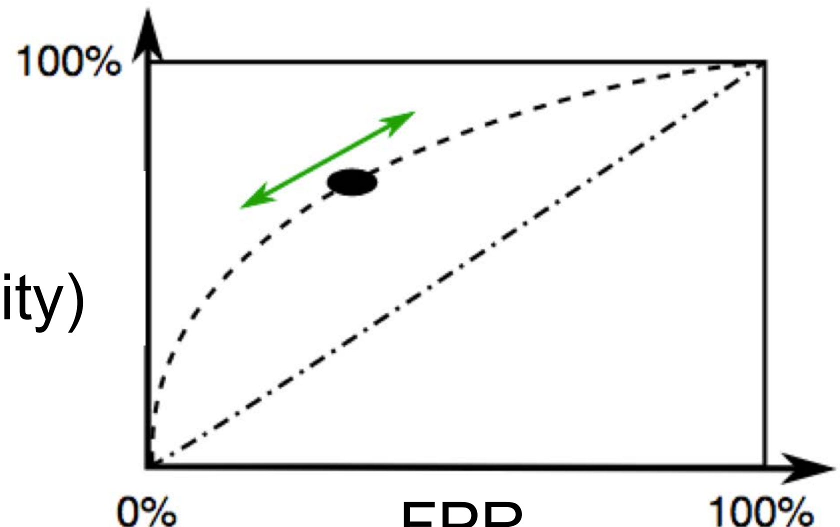
ROC Curve (across thresholds)



Area under ROC curve
(aka AUROC or AUC or “C statistic”)

Area varies from 0.0 – 1.0.
0.5 is random guess.
1.0 is perfect.

Graphical:



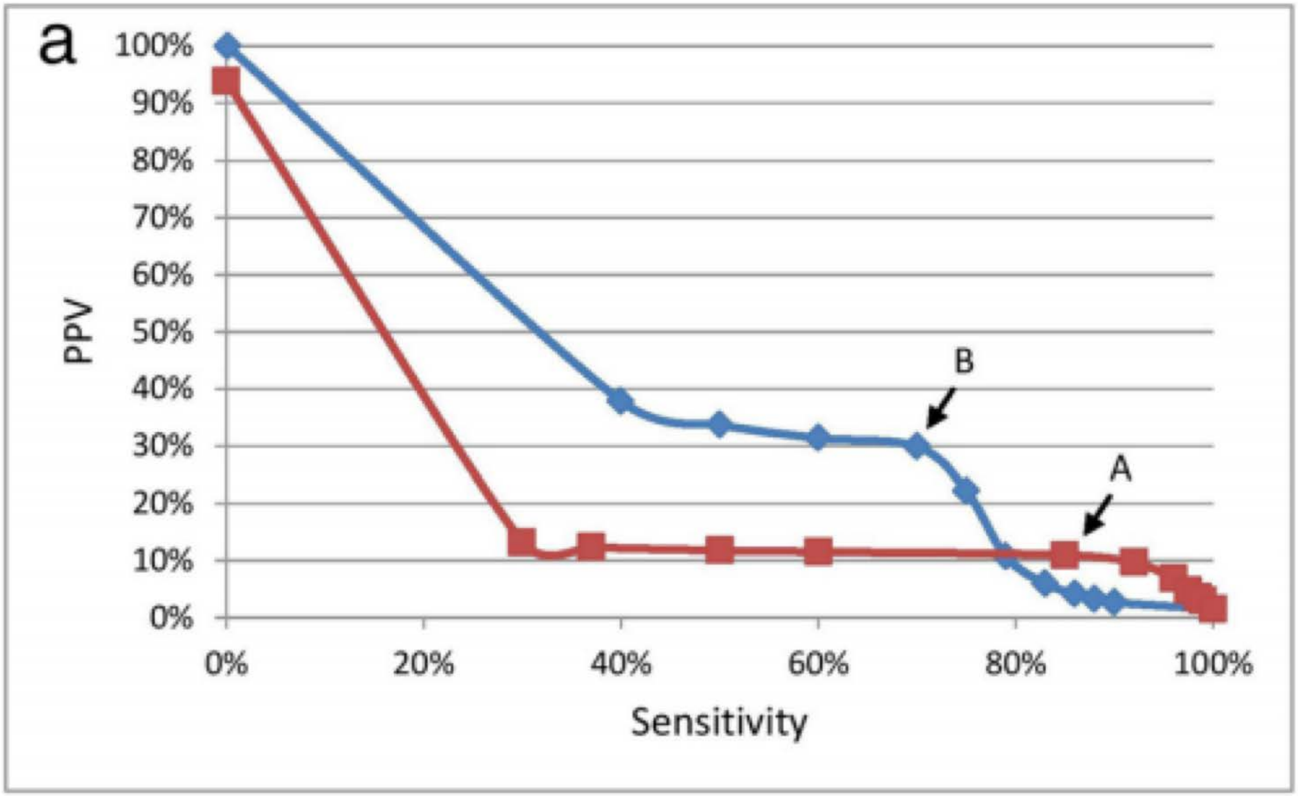
Probabilistic:

$$\text{AUROC} \triangleq \Pr(\hat{y}(x_i) > \hat{y}(x_j) | y_i = 1, y_j = 0)$$

For random pair of examples, one positive and one negative,
What is probability classifier will rank positive one higher?

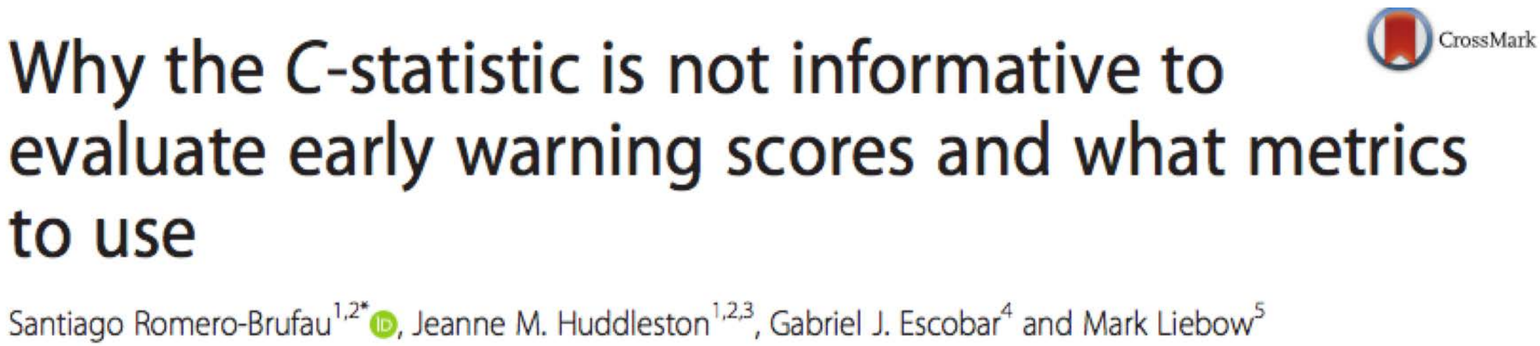
Precision-Recall Curve

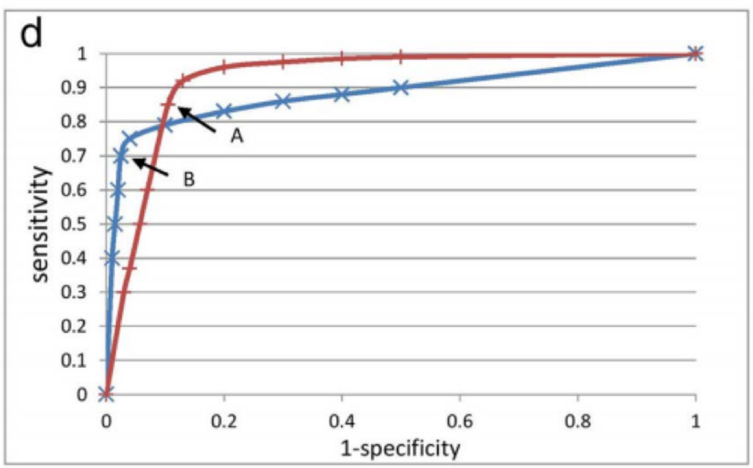
precision



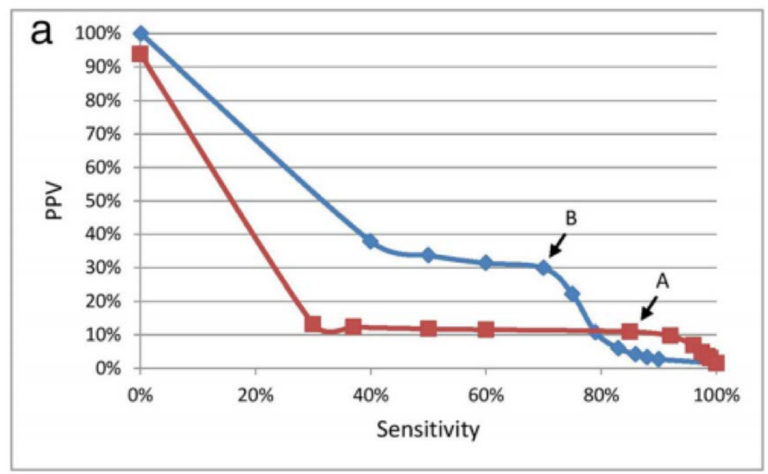
recall (= TPR)

AUROC not always best choice





AUROC: red is better



Blue much better for alarm fatigue

Classifier: Evaluation Metrics

https://scikit-learn.org/stable/modules/model_evaluation.html

1) To evaluate predicted scores / probabilities

log_loss (y_true, y_pred[, eps, normalize, …])	Log loss, aka logistic loss or cross-entropy loss.
hinge_loss (y_true, pred_decision[, labels, …])	Average hinge loss (non-regularized)

2) To evaluate specific binary decisions

'precision' etc.	metrics.precision_score
'recall' etc.	metrics.recall_score

3) To make ROC or PR curves and compute areas

'average_precision'	metrics.average_precision_score
'roc_auc'	metrics.roc_auc_score

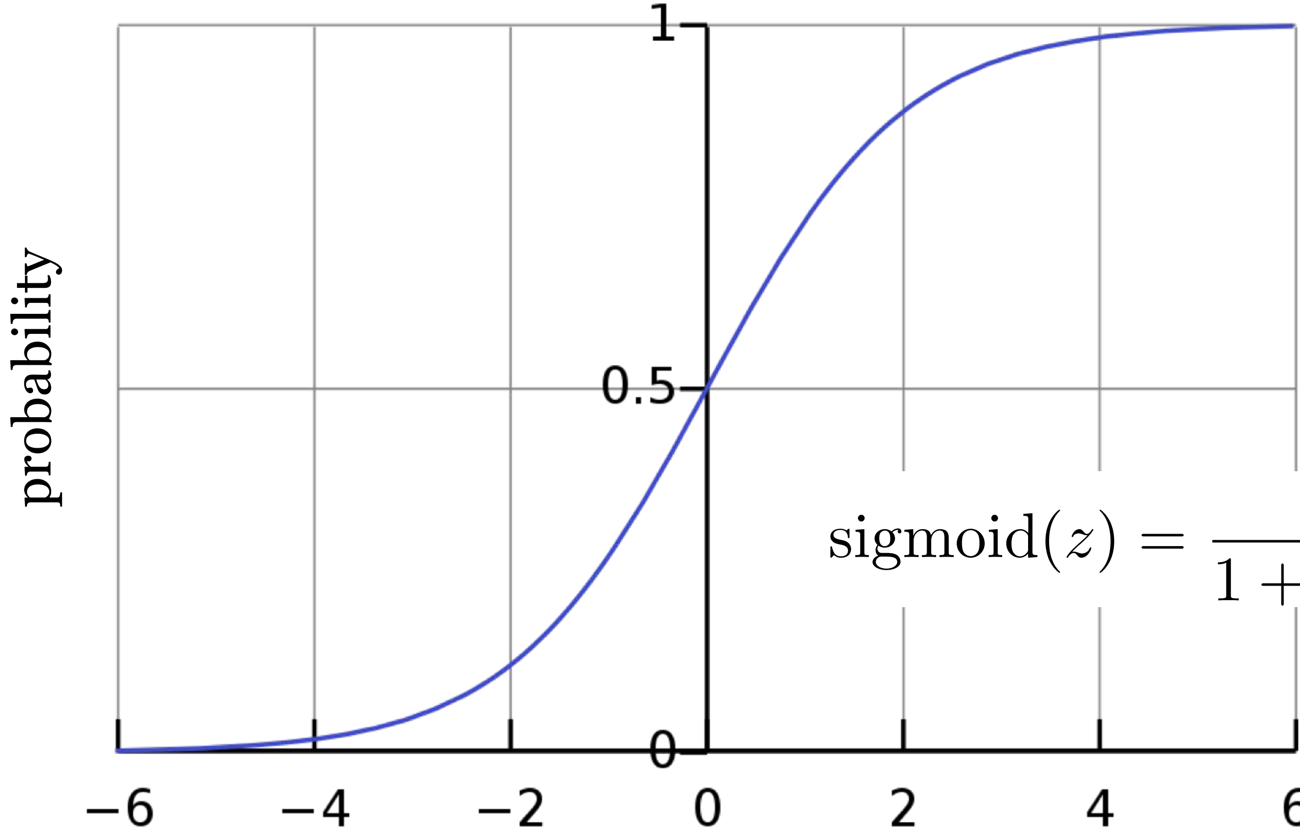
Objectives: Classifier Overview

- 3 steps of a classification task
  - Prediction
    - Making hard binary decisions
    - Predicting class probabilities
  - Training
  - Evaluation
    - Performance Metrics

- A “taste” of 3 Methods
  - Logistic Regression
  - K-Nearest Neighbors
  - Decision Tree Regression

Logistic Sigmoid Function

Goal: Transform real values into probabilities



$$\mathrm{sigmoid}(z) = \frac{1}{1 + e^{-z}}$$

Logistic Regression

Parameters:

weight vector $w = [w_1, w_2, \ldots w_f \ldots w_F]$

bias scalar b

Prediction:

$$\hat{p}(x_i, w, b) = p(y_i = 1 | x_i) \triangleq \text{sigmoid}\left(\sum_{f=1}^{F} w_f x_{if} + b \right)$$

Training:

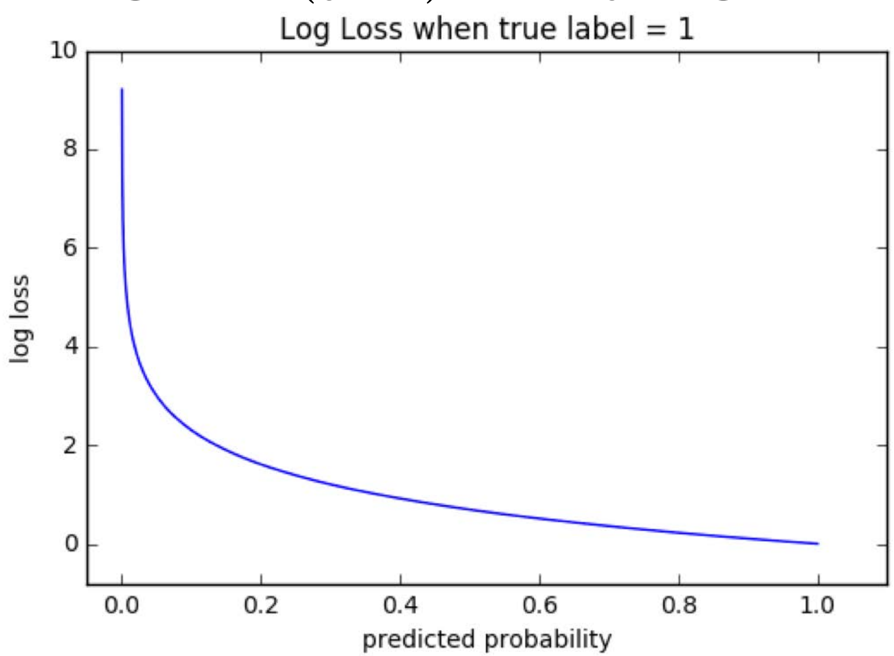
find weights and bias that minimize error

Measuring prediction quality for a probabilistic classifier

Use the log loss (aka "binary cross entropy", related to"logistic loss")

```
from sklearn.metrics import log_loss
```

$$\text{log\_loss}(y, \hat{p}) = -y \log \hat{p} - (1 - y) \log(1 - \hat{p})$$



Advantages:

- smooth
- easy to take derivatives!

Logistic Regression: Training

Optimization: Minimize total log loss on train set

$$\min_{w,b} \sum_{n=1}^{N} \text{log\_loss}(y_n, \hat{p}(x_n, w, b))$$

Algorithm: Gradient descent

More in next class!

Avoid overfitting: Use L2 or L1 penalty on weights

Nearest Neighbor Classifier

Parameters:

none

Prediction:

- find “nearest” training vector to given input x
- predict y value of this neighbor

Training:

none needed (use training data as lookup table)

K nearest neighbor classifier

Parameters:

K : number of neighbors

Prediction:

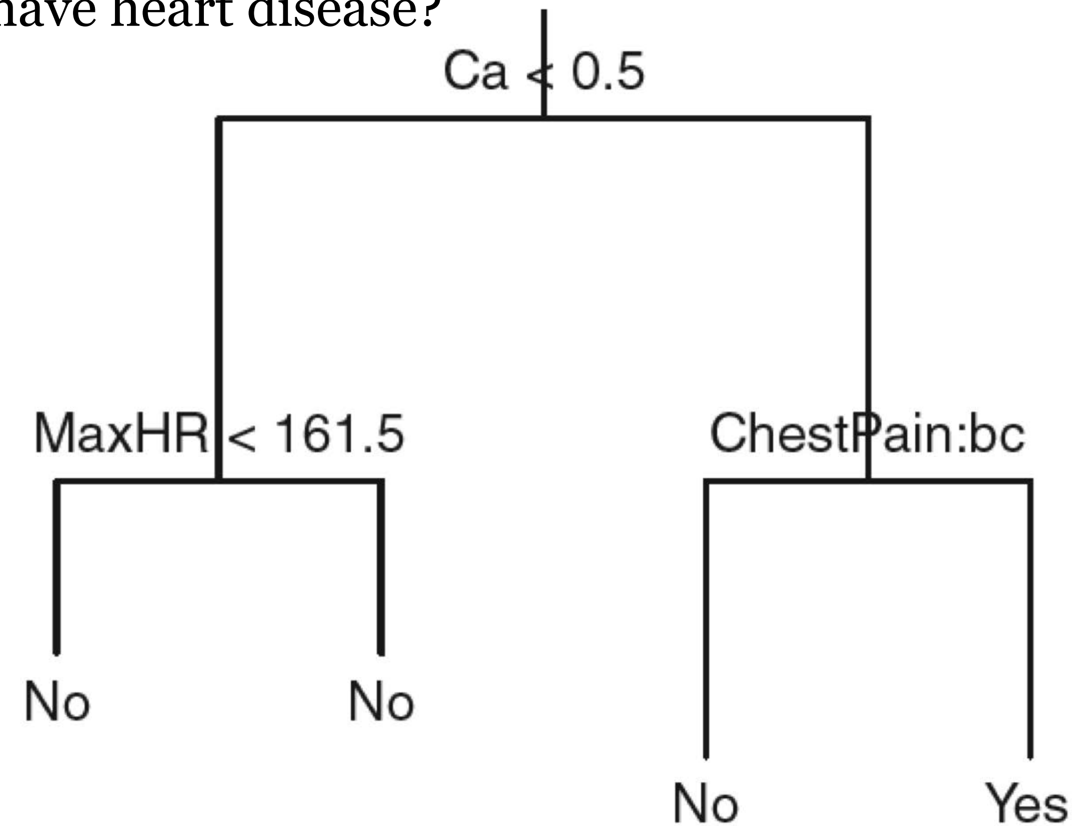
- find K “nearest” training vectors to input x
- predict: vote most common y in neighborhood
- predict_proba: report fraction of labels

Training:

none needed (use training data as lookup table)

Decision Tree Classifier

Goal: Does patient have heart disease?



Leaves make binary predictions! *(but can be made probabilistic)*

Decision Tree Classifier

Parameters:

- *at each internal node: x* variable id and threshold
- *at each leaf:* probability of positive *y* label

Prediction:

- identify rectangular region for input x
- predict: most common y value in region
- predict_proba: report fraction of each label in regtion

Training:

- minimize error on training set
- often, use greedy heuristics

Summary of Methods

	Function class flexibility	Knobs to tune	Interpret?
Logistic Regression	Linear	L2/L1 penalty on weights	Inspect weights
Decision Tree Classifier	Axis-aligned Piecewise constant	Max. depth Min. leaf size Goal criteria	Inspect tree
K Nearest Neighbors Classifier	Piecewise constant	Number of Neighbors Distance metric How neighbors vote	Inspect neighbors