Upcoming Deadlines

- HW 2 due on Wednesday
- Quiz 3 due on Thursday

Optional Extra Readings: Logistic Regression

- Hastie and Tibshirani Ch 4.1-4
- Hardt and Recht Ch 3: Supervised Learning
  - Linear and logistic regression introduced as instances of a “perceptron”: https://mlstory.org/supervised.html
- d2l.ai interactive textbook chapter on logistic regression, taught as a simple instance of a neural network: https://d2l.ai/chapter_linear-classification/index.html (recommended to use in pytorch mode)

Lecture 7: Logistic Regression

CIS 4190/5190

Spring 2023

Classification Metrics

- While we minimize the NLL, we often evaluate using **accuracy**

- However, even accuracy isn't necessarily the "right" metric
  - If 99% of labels are negative (i.e., $y_i = 0$), accuracy of $f_\beta(x) = 0$ is 99%!
  - For instance, very few patients test positive for most diseases
  - "Imbalanced data"

- What are alternative metrics for these settings?

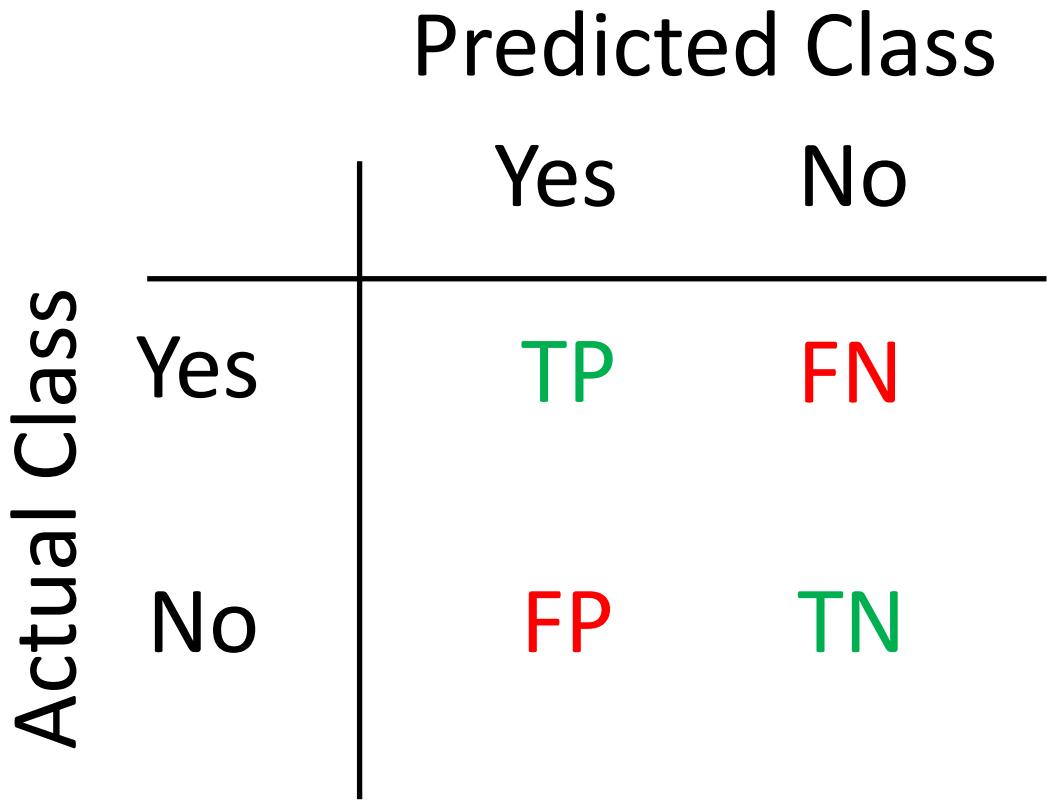
Classification Metrics

- **Classify test examples as follows:**
  - **True positive (TP):** Actually positive, predictive positive
  - **False negative (FN):** Actually positive, predicted negative
  - **True negative (TN):** Actually negative, predicted negative
  - **False positive (FP):** Actually negative, predicted positive

- Many metrics expressed in terms of these; for example:

$$\text{accuracy} = \frac{TP + TN}{n} \qquad \text{error} = 1 - \text{accuracy} = \frac{FP + FN}{n}$$

Confusion Matrix



Confusion Matrix

		Predicted Class: Yes	Predicted Class: No
Actual Class	Yes	3 TP	4 FN
	No	6 FP	37 TN

Accuracy = 0.8

Classification Metrics

- For imbalanced metrics, we roughly want to disentangle:
  - Accuracy on “positive examples”
  - Accuracy on “negative examples”

- Different definitions are possible (and lead to different meanings)!

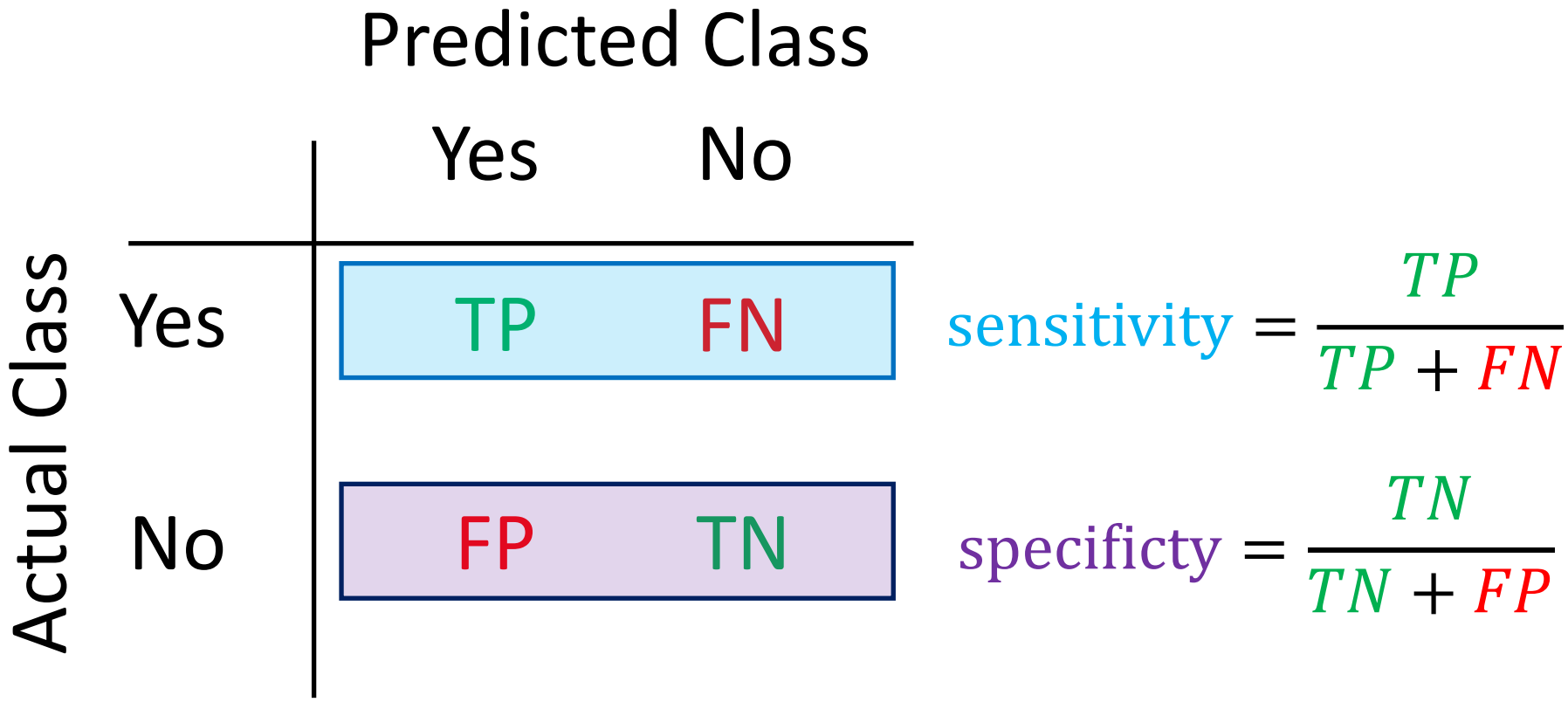
Sensitivity & Specificity

- **Sensitivity:** What fraction of **actual positives** are **predicted positive**?
  - **Good sensitivity:** If you have the disease, the test correctly detects it
  - Also called **true positive rate**

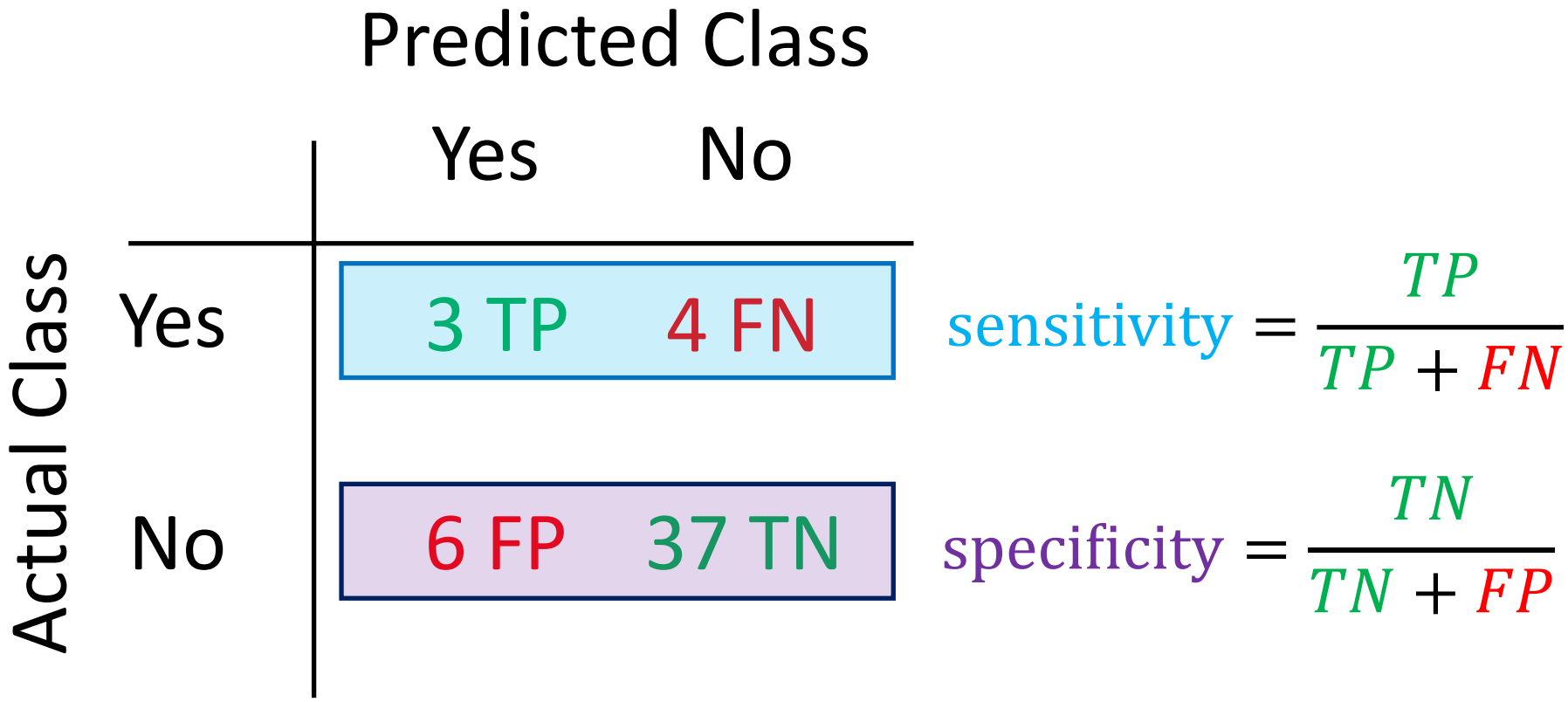
- **Specificity:** What fraction of **actual negatives** are **predicted negative**?
  - **Good specificity:** If you do not have the disease, the test says so
  - Also called **true negative rate**

- Commonly used in medicine

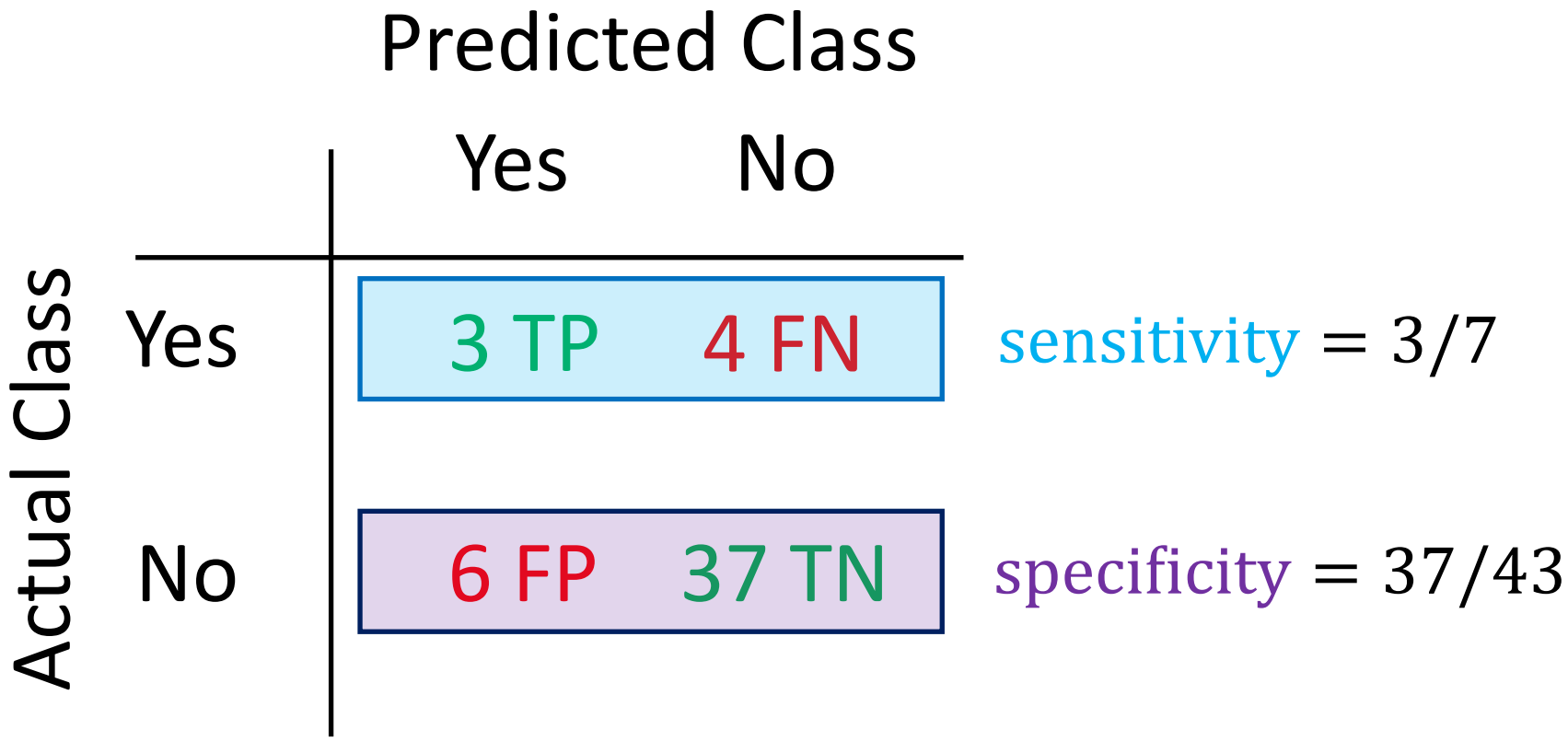
Sensitivity & Specificity



Sensitivity & Specificity



Sensitivity & Specificity



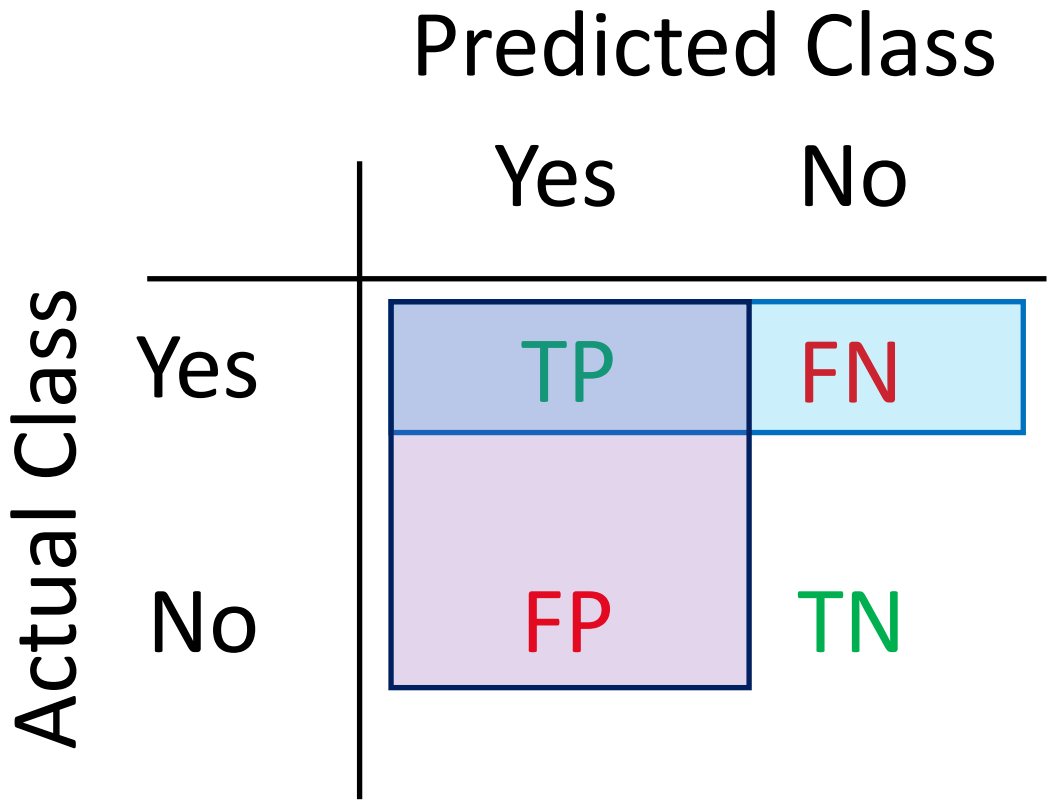
Precision & Recall

- **Recall:** What fraction of **actual positives** are **predicted positive**?
  - **Good recall:** If you have the disease, the test correctly detects it
  - Also called the **true positive rate** (and sensitivity)

- **Precision:** What fraction of **predicted positives** are **actual positives**?
  - **Good precision:** If the test says you have the disease, then you have it
  - Also called **positive predictive value**

- Used in information retrieval, NLP

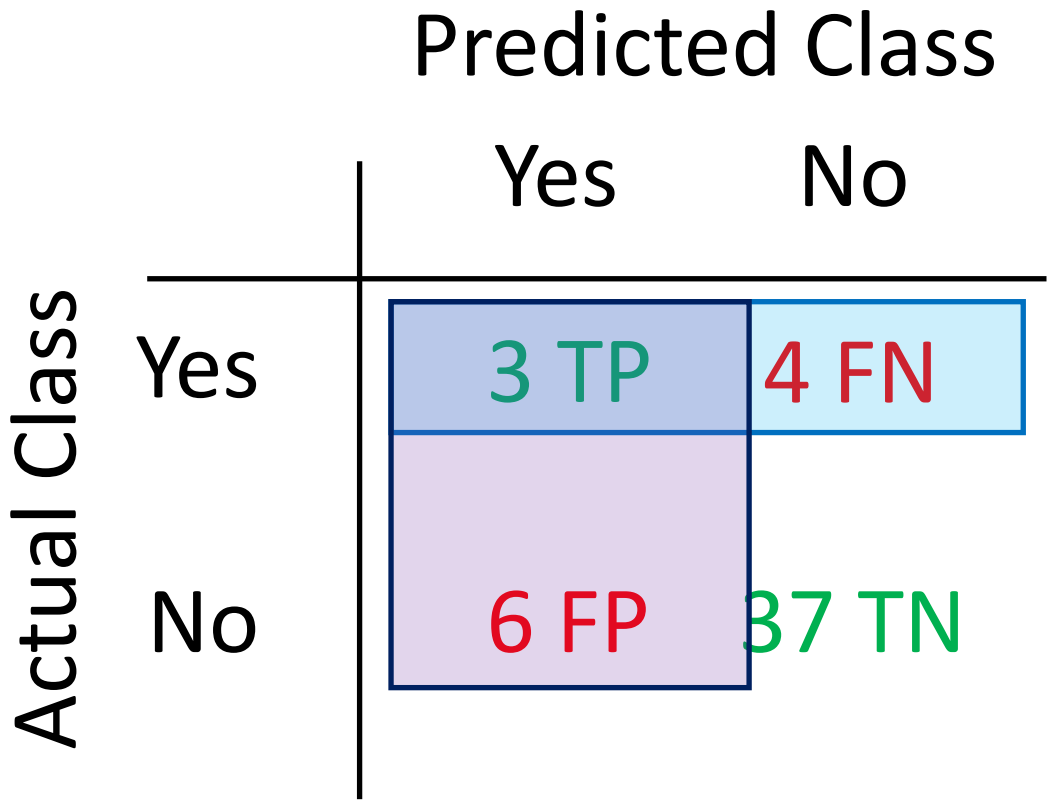
Precision & Recall



$$\text{recall} = \frac{TP}{TP + FN}$$

$$\text{precision} = \frac{TP}{TP + FP}$$

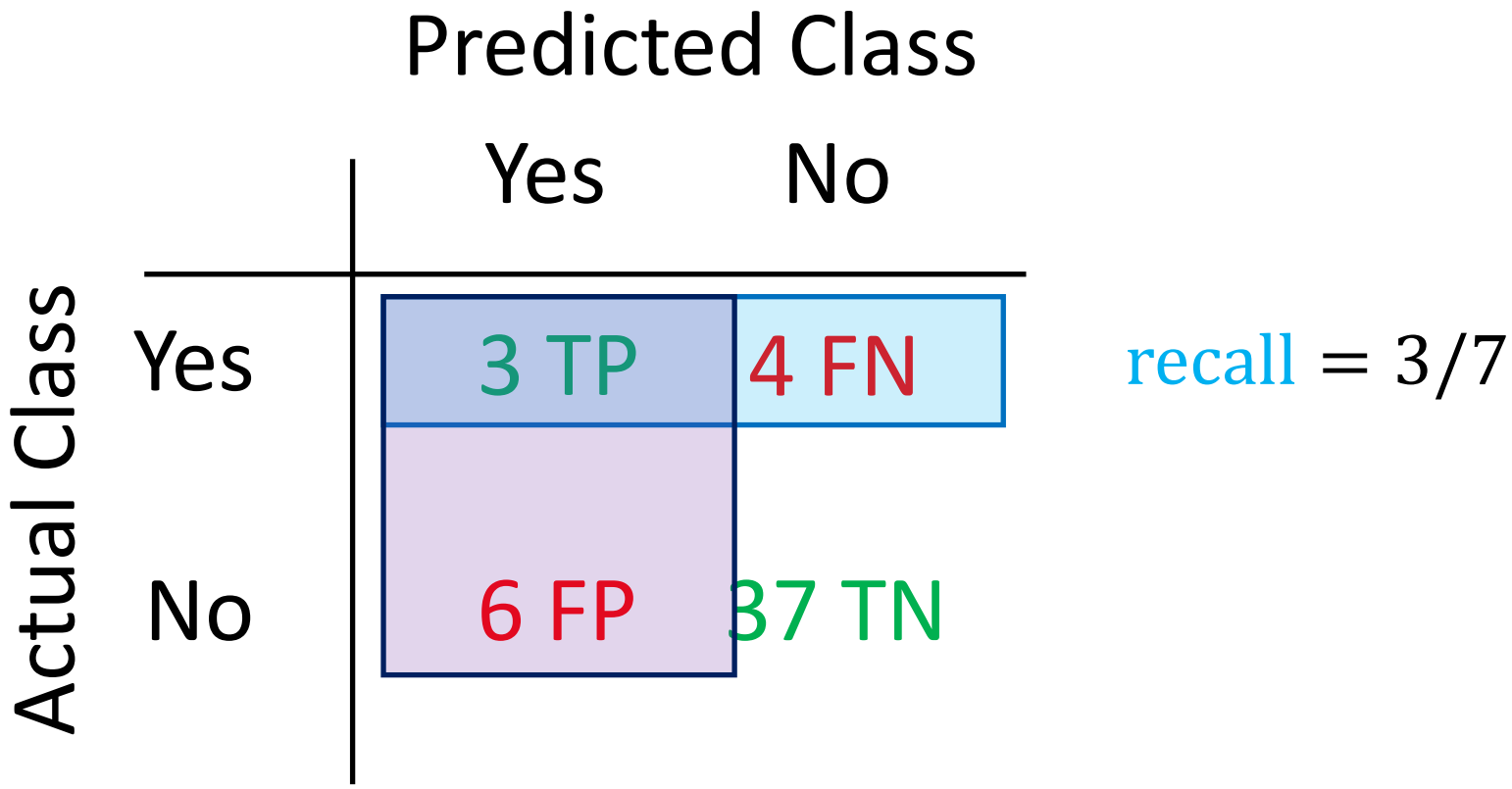
Precision & Recall



$$\text{recall} = \frac{TP}{TP + FN}$$

$$\text{precision} = \frac{TP}{TP + FP}$$

Precision & Recall



Classification Metrics

- **How to obtain a single metric?**
  - Combination, e.g., $F_1\ \text{score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$ is the harmonic mean
  - More on this later

- **How to choose the "right" metric?**
  - No generally correct answer
  - Depends on the goals for the specific problem/domain

Optimizing a Classification Metric

- We are training a model to minimize NLL, but we have a different "true" metric that we actually want to optimize

- Two strategies (can be used together):
  - **Strategy 1:** Optimize prediction threshold threshold
  - **Strategy 2:** Upweight positive (or negative) examples

Optimizing Prediction Threshold

- Consider hyperparameter τ for the threshold:

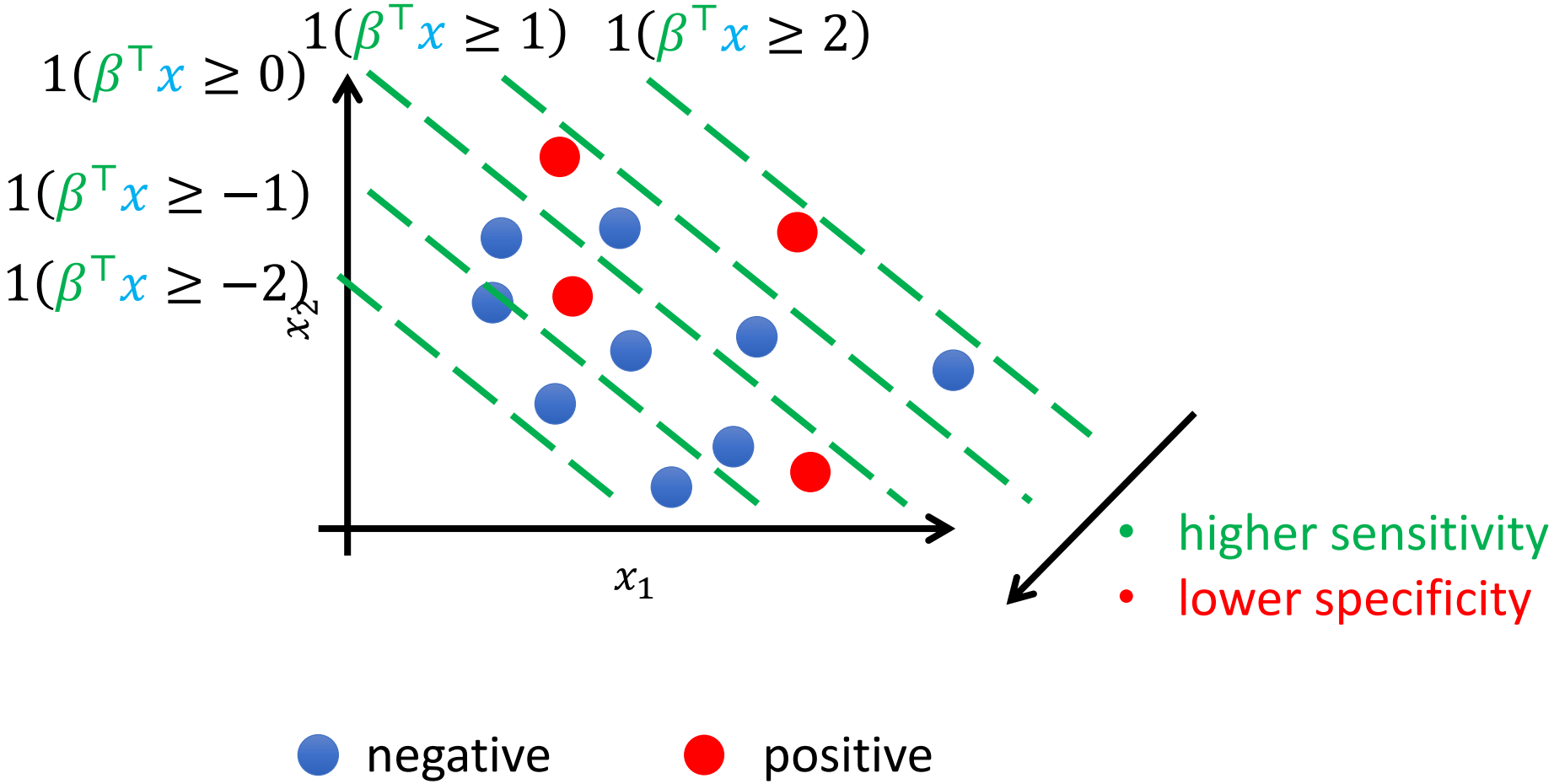
$$f_\beta(x) = 1(\beta^\top x \geq 0)$$

Optimizing Prediction Threshold

- Consider hyperparameter τ for the threshold:

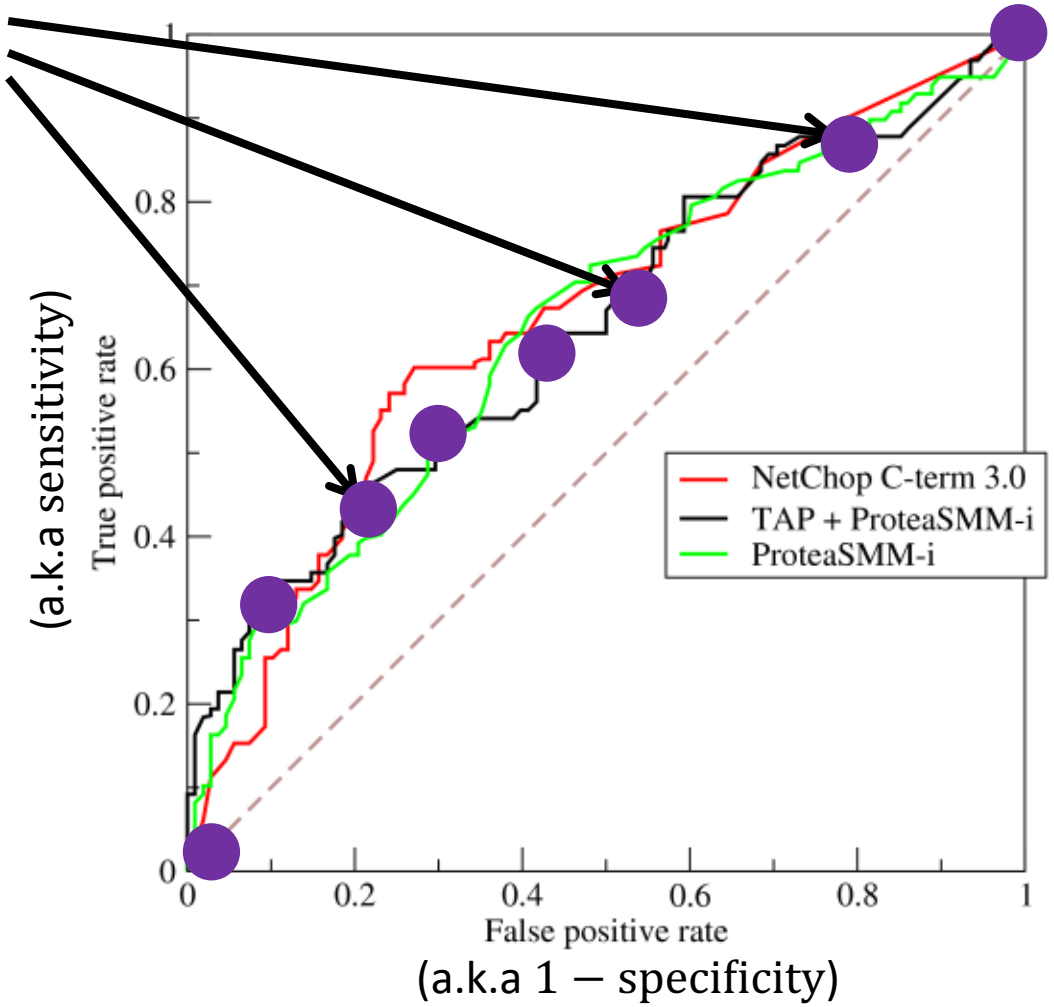
$$f_\beta(x) = 1(\beta^\top x \geq \tau)$$

Optimizing Prediction Threshold



Visualization: ROC Curve

Each point on this curve corresponds to a choice of τ



Aside: Area under ROC curve is another metric people consider when evaluating $\hat{\beta}(Z)$

Optimizing Prediction Threshold

- Consider hyperparameter τ for the threshold:

$$f_\beta(x) = 1(\beta^\top x \geq \tau)$$

- Unlike most hyperparameters, we choose this one **after** we have already fit the model on the training data
  - Then, choose the value of τ that optimizes the desired metric
  - Fit using validation data (training data is OK if needed)

Optimizing Prediction Threshold

- **Step 1:** Compute the optimal parameters $\hat{\beta}(Z_{\text{train}})$
  - Using gradient descent on NLL loss over the training dataset
  - **Resulting model:** $f_{\hat{\beta}(Z_{\text{train}})}(x) = 1\left(\hat{\beta}(Z_{\text{train}})^{\top} x \geq 0\right)$

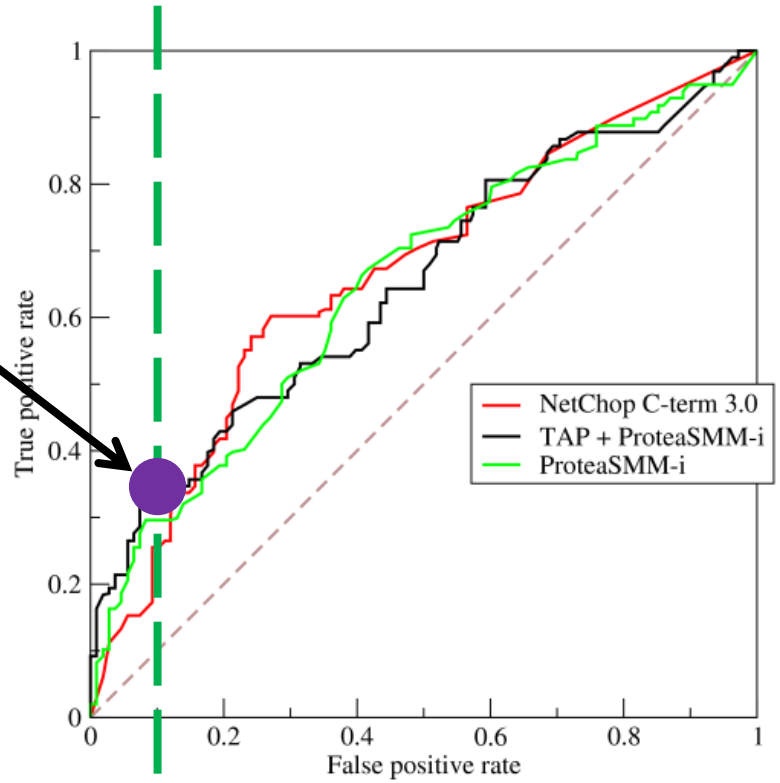
- **Step 2:** Modify threshold τ in model to optimize desired metric
  - Search over a fixed set of τ on the validation dataset
  - **Resulting model:** $f_{\hat{\beta}(Z_{\text{train}}),\hat{\tau}(Z_{\text{val}})}(x) = 1\left(\hat{\beta}(Z_{\text{train}})^{\top} x \geq \hat{\tau}(Z_{\text{val}})\right)$

- **Step 3:** Evaluate desired metric on test set

Choice of Metric Revisited

- **Common strategy:** Optimize one metric at fixed value of another

Choose τ corresponding to model at this point



$\text{specificity} = 0.9$

Optimizing a Classification Metric

- We are training a model to minimize NLL, but we have a different "true" metric that we actually want to optimize

- Two strategies (can be used together):
  - **Strategy 1:** Optimize prediction threshold threshold
  - **Strategy 2:** Upweight positive (or negative) examples

Class Re-Weighting

- **Weighted NLL:** Include a class-dependent weight w_y:

$$\ell(\beta; Z) = -\sum_{i=1}^{n} w_{y_i} \cdot \log p_\beta(y_i \mid x_i)$$

- **Intuition:** Tradeoff between accuracy on negative/positive examples
  - To improve sensitivity (true positive rate), upweight positive examples
  - To improve specificity (true negative rate), upweight negative examples

- Can use this strategy to learn β, and the first strategy to choose τ

Classification Metrics

- NLL isn't usually the "true" metric
  - Instead, frequently used due to good computational properties
- Many choices with different meanings
- Typical strategy:
  - Learn β by minimizing the NLL loss
  - Choose class weights w_y and threshold τ to optimize desired metric

“Non-Parametric” Machine Learning Approaches

K-Nearest Neighbors and Decision Trees

So far, we have seen:

- Machine learning methods are defined by:
  - A model family / hypothesis space
  - An objective function
  - An optimization approach

So far, we have seen:

- Machine learning methods are defined by:
  - A model family / hypothesis space
    - Defined in terms of some fixed-length parameter vector $\beta \in \mathbb{R}^D$
      - Linear regression: $\hat{y} = \beta^T x$
      - Logistic regression: $p(\hat{y} = 1) = \sigma(\beta^T x)$
  - An objective function
  - An optimization approach

So far, we have seen:

- Machine learning methods are defined by:
  - A model family / hypothesis space
    - Defined in terms of some fixed-length parameter vector $\beta \in \mathbb{R}^D$
      - Linear regression: $\hat{y} = \beta^T x$
      - Logistic regression: $p(\hat{y} = 1) = \sigma(\beta^T x)$
  - An objective function
    - $L(\beta; Z)$ defines what it means for parameters β to be good given training set Z,
      - e.g. MSE for linear regression, or maximum-likelihood logistic regression objective
  - An optimization approach

So far, we have seen:

- Machine learning methods are defined by:
  - A model family / hypothesis space
    - Defined in terms of some fixed-length parameter vector $\beta \in \mathbb{R}^D$
      - Linear regression: $\hat{y} = \beta^T x$
      - Logistic regression: $p(\hat{y} = 1) = \sigma(\beta^T x)$
  - An objective function
    - $L(\beta; Z)$ defines what it means for parameters β to be good given training set Z,
      - e.g. MSE for linear regression, or maximum-likelihood logistic regression objective
  - An optimization approach
    - Some process of searching for optimal parameter vector β

So far, we have seen:

- Machine learning methods are defined by:
  - A model family / hypothesis space
    - Defined in terms of some fixed-length parameter vector $\beta \in \mathbb{R}^D$
      - Linear regression: $\hat{y} = \beta^T x$
      - Logistic regression: $p(\hat{y} = 1) = \sigma(\beta^T x)$

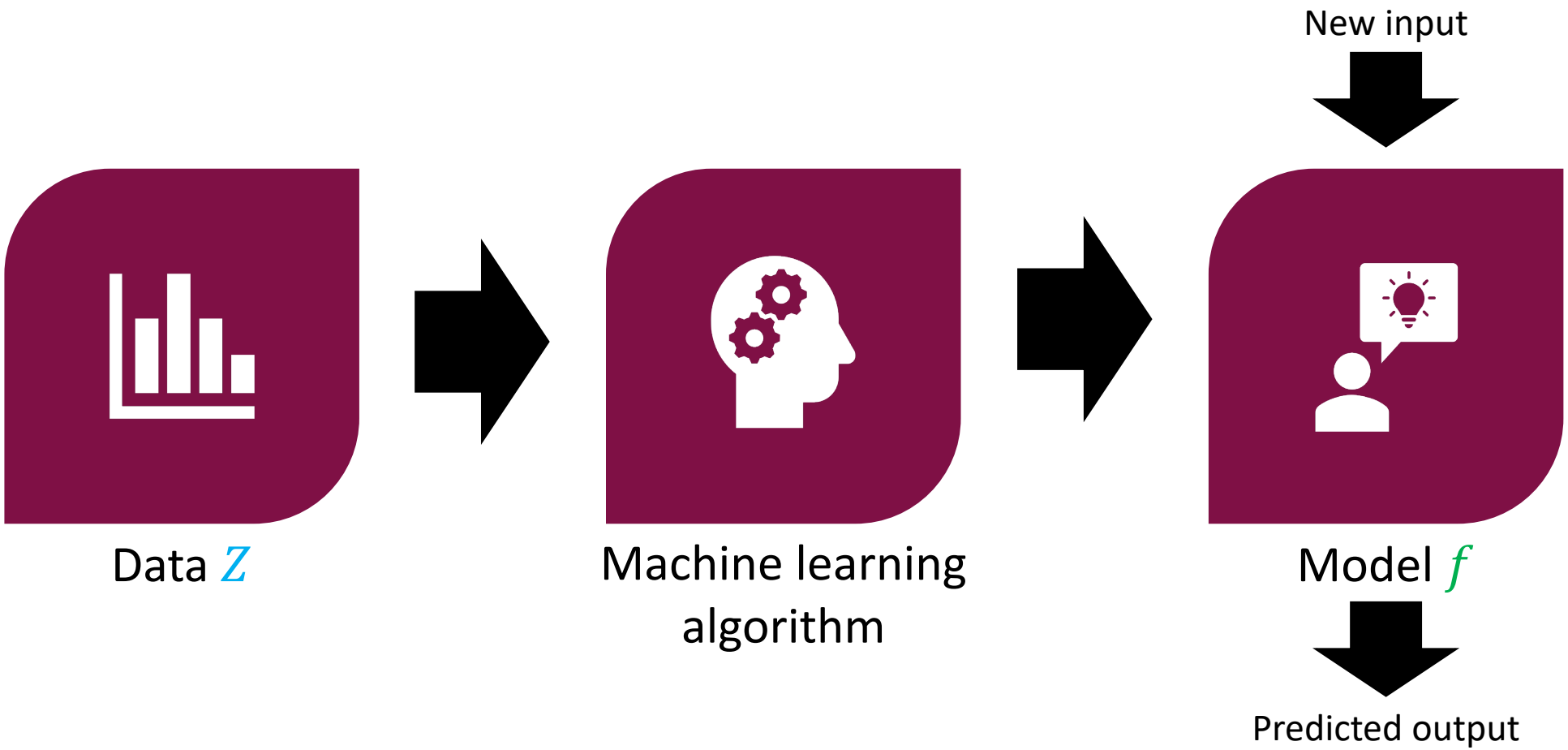
But not all machine learning approaches fit into this framework!

$L(\beta, Z)$ defines what it means for parameters β to be good given training set Z,

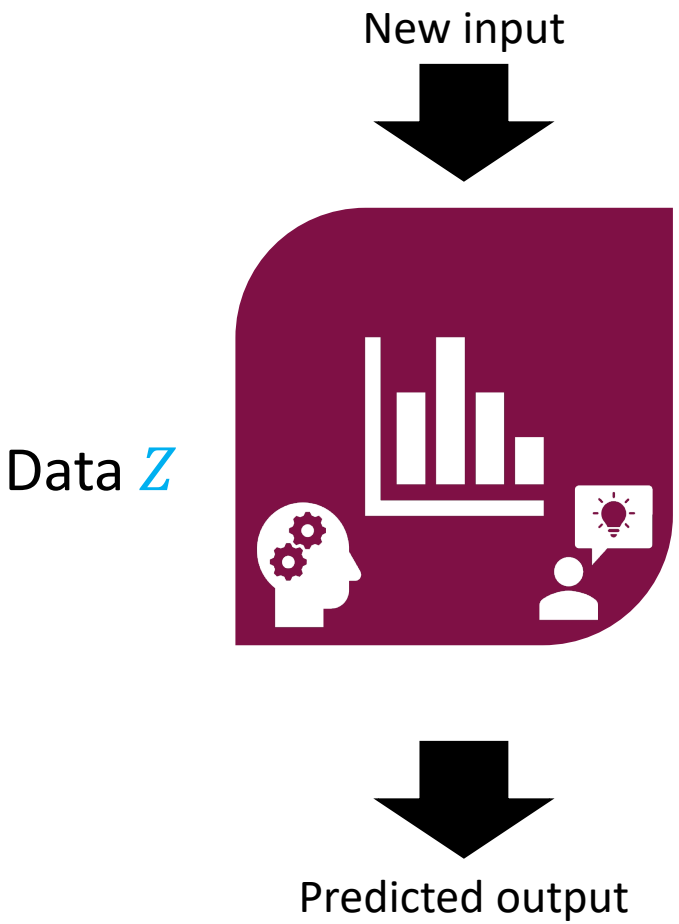
- e.g. MSE for linear regression, or maximum-likelihood logistic regression objective

  - An optimization approach
    - Some process of searching for optimal parameter vector β

Recall: The Typical Machine Learning Pipeline



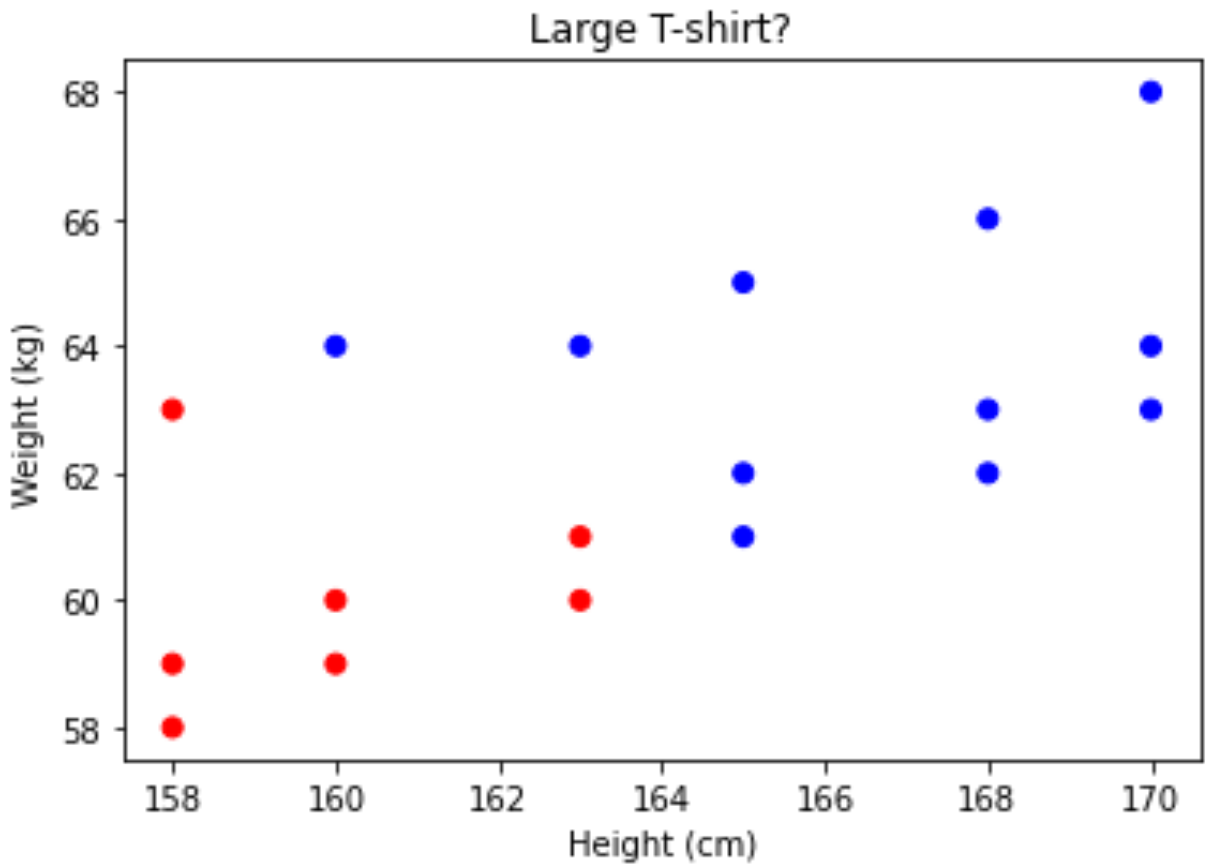
Next Up: k-Nearest Neighbors.
A Simple Approach, Connected Directly to The Data



Note: this schematic seems to skip any explicit “model training” on data.

In a sense, the data *is* the model.
How might this work?

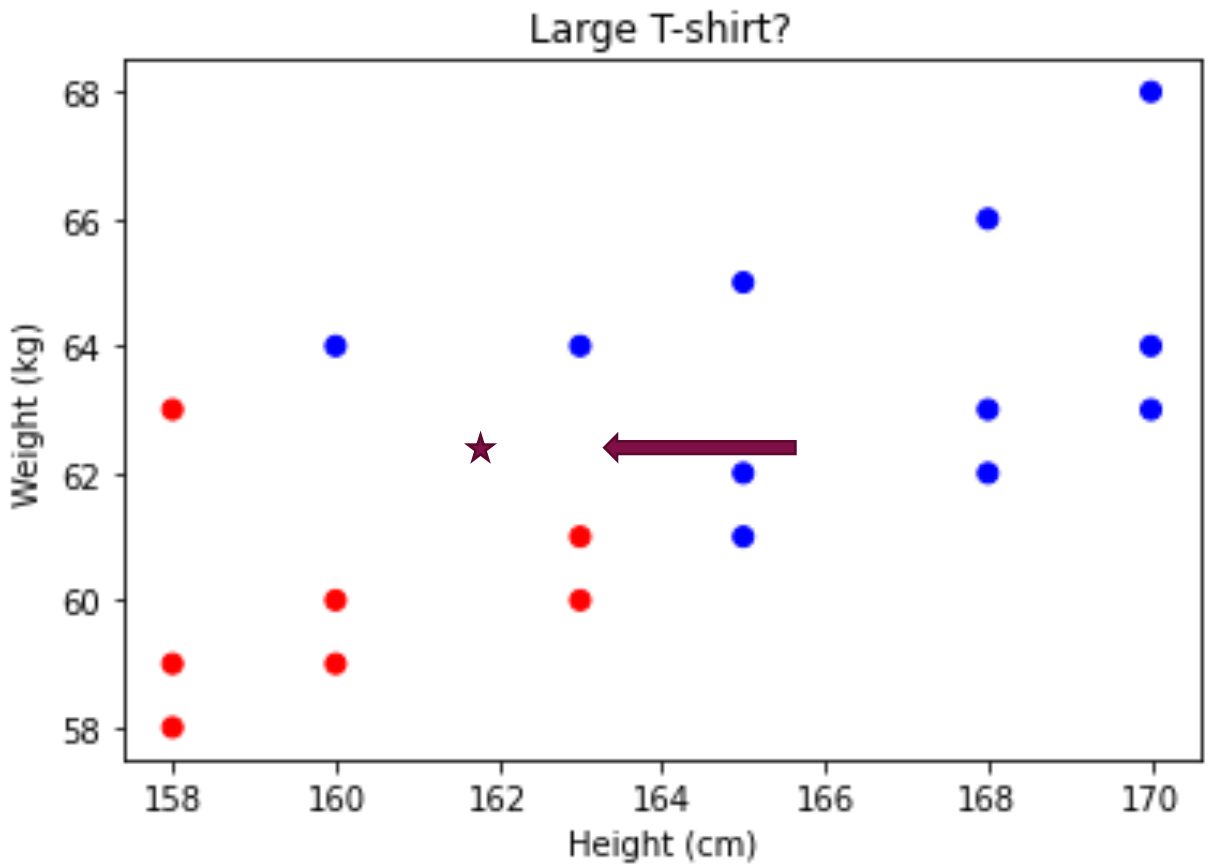
Setup: Binary Classification (Training)



Height (cm)	Weight (kg)	Large (vs Medium) t-shirt?
158	58	F
158	59	F
158	63	F
160	59	F
160	60	F
163	60	F
163	61	F
160	64	T
163	64	T
165	61	T
165	62	T
165	65	T
168	62	T
168	63	T
168	66	T
170	63	T
170	64	T
170	68	T

Based on data from https://www.listendata.com/2017/12/k-nearest-neighbor-step-by-step-tutorial.html

Test Time! Guess the Label For A New Sample?



Height (cm)	Weight (kg)	Large (vs Medium) t-shirt?
158	58	F
158	59	F
158	63	F
160	59	F
160	60	F
163	60	F
163	61	F
160	64	T
163	64	T
165	61	T
165	62	T
165	65	T
168	62	T
168	63	T
168	66	T
170	63	T
170	64	T
170	68	T

Based on data from https://www.listendata.com/2017/12/k-nearest-neighbor-step-by-step-tutorial.html

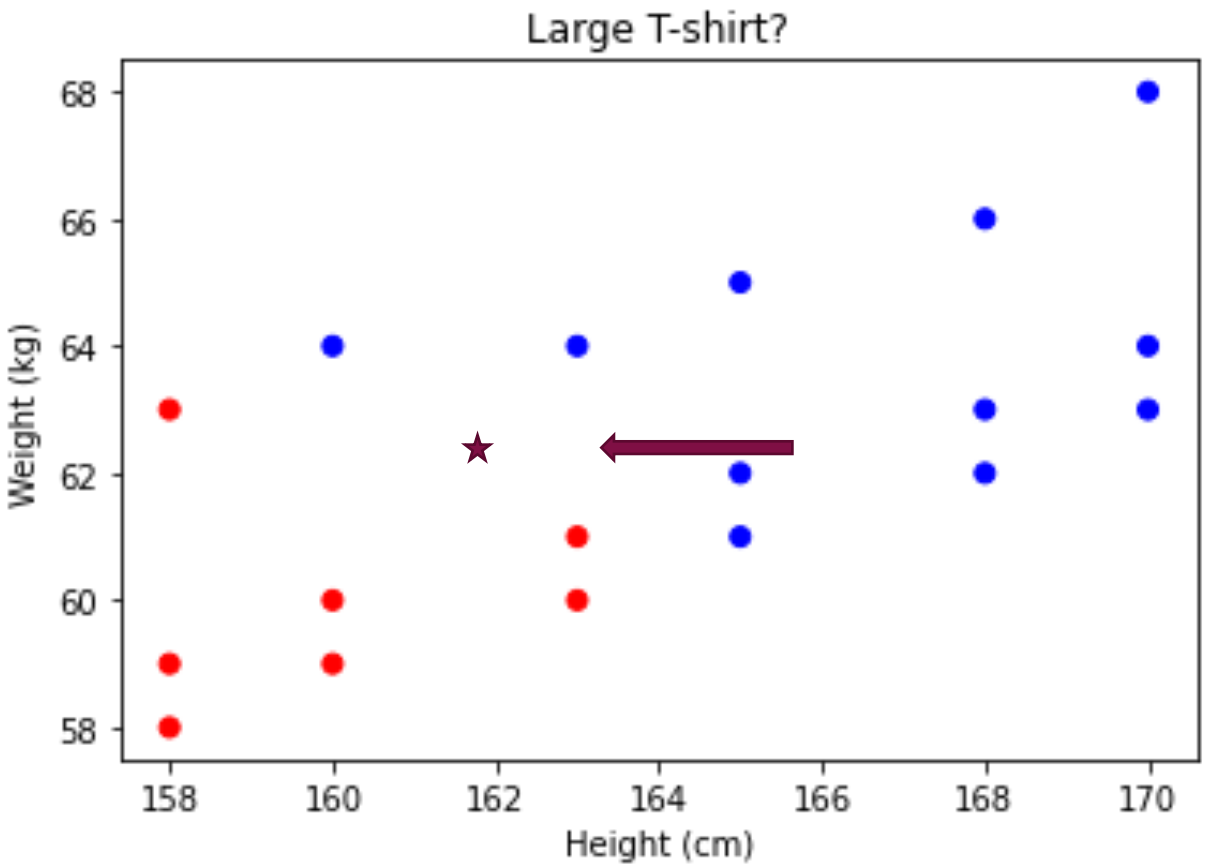
k-Nearest Neighbors (kNN)

- **kNN Classification:** To predict category label y of a new point $\boldsymbol{x}$:
  - Find k nearest neighbors
  - Assign the majority label

- **kNN regression:** To predict numeric value y of a new point $\boldsymbol{x}$:
  - Find k nearest neighbors
  - Average the values associated with the neighbors

In each case, varying k could change the predictions

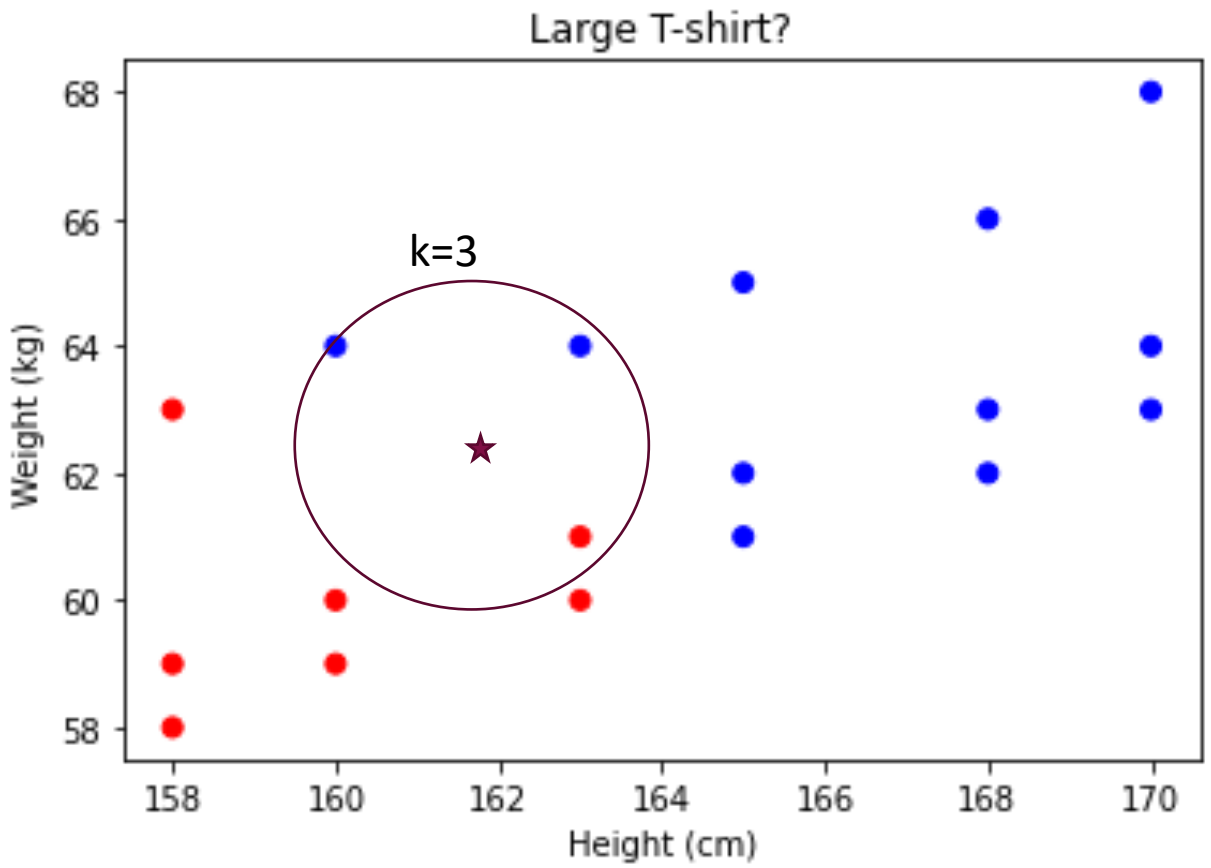
kNN Prediction: What Label?



Height (cm)	Weight (kg)	Large (vs Medium) t-shirt?
158	58	F
158	59	F
158	63	F
160	59	F
160	60	F
163	60	F
163	61	F
160	64	T
163	64	T
165	61	T
165	62	T
165	65	T
168	62	T
168	63	T
168	66	T
170	63	T
170	64	T
170	68	T

Based on data from https://www.listendata.com/2017/12/k-nearest-neighbor-step-by-step-tutorial.html

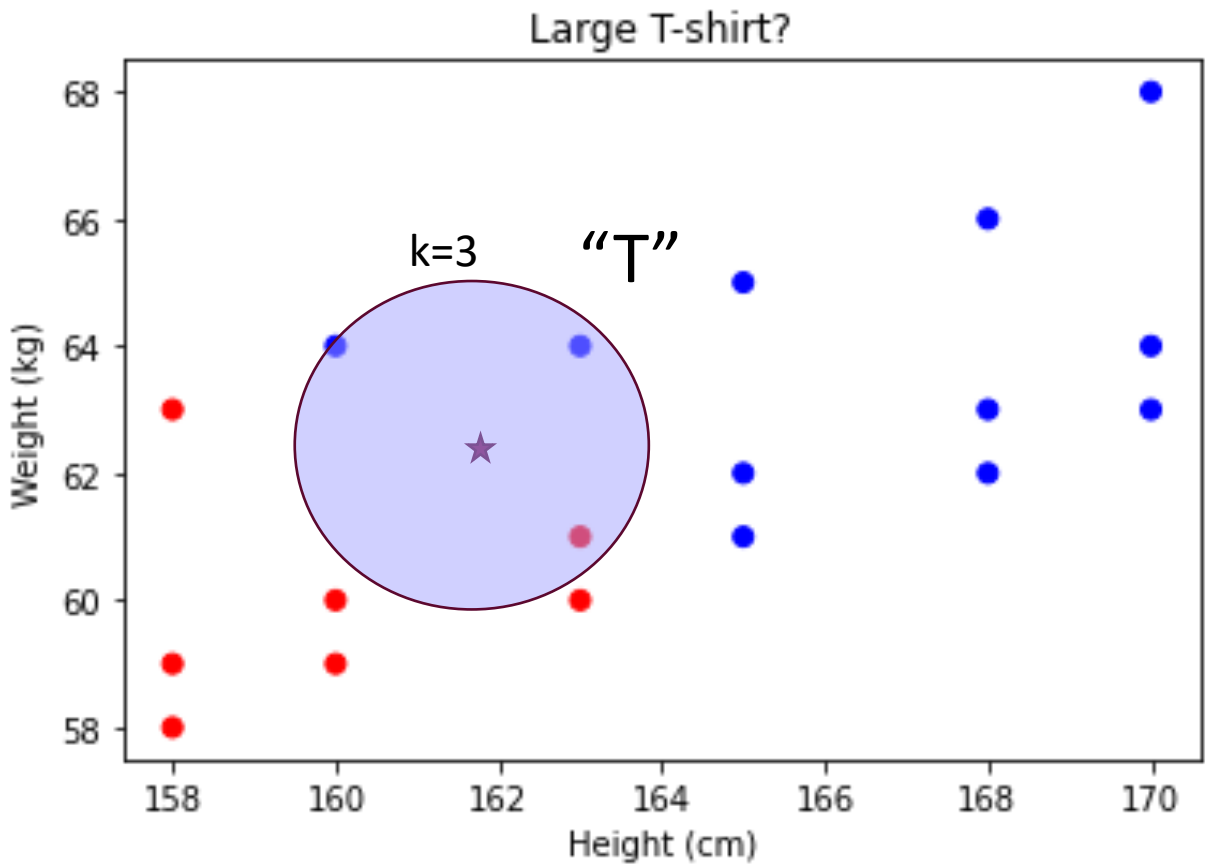
kNN Prediction: What Label?



Height (cm)	Weight (kg)	Large (vs Medium) t-shirt?
158	58	F
158	59	F
158	63	F
160	59	F
160	60	F
163	60	F
163	61	F
160	64	T
163	64	T
165	61	T
165	62	T
165	65	T
168	62	T
168	63	T
168	66	T
170	63	T
170	64	T
170	68	T

Based on data from https://www.listendata.com/2017/12/k-nearest-neighbor-step-by-step-tutorial.html

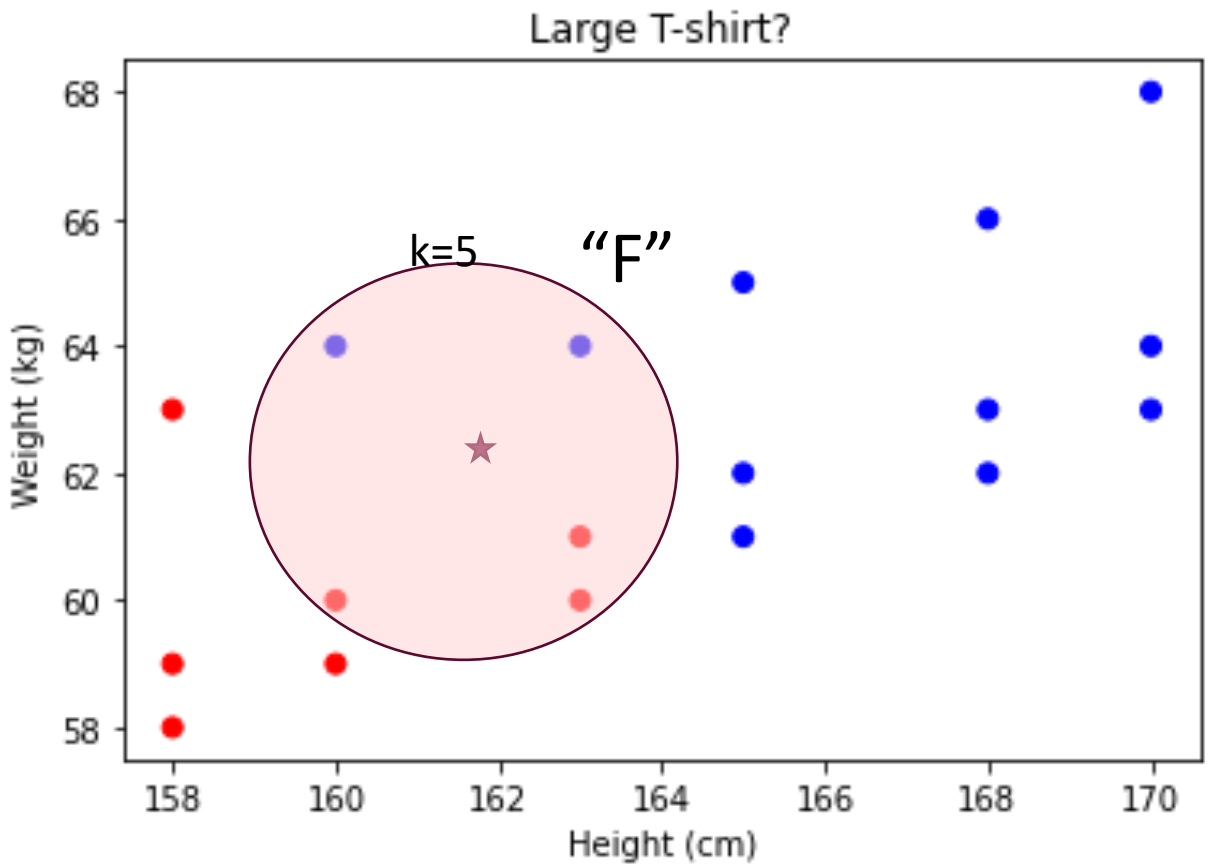
kNN Prediction: What Label?



Height (cm)	Weight (kg)	Large (vs Medium) t-shirt?
158	58	F
158	59	F
158	63	F
160	59	F
160	60	F
163	60	F
163	61	F
160	64	T
163	64	T
165	61	T
165	62	T
165	65	T
168	62	T
168	63	T
168	66	T
170	63	T
170	64	T
170	68	T

Based on data from https://www.listendata.com/2017/12/k-nearest-neighbor-step-by-step-tutorial.html

kNN Prediction: What Label?



Height (cm)	Weight (kg)	Large (vs Medium) t-shirt?
158	58	F
158	59	F
158	63	F
160	59	F
160	60	F
163	60	F
163	61	F
160	64	T
163	64	T
165	61	T
165	62	T
165	65	T
168	62	T
168	63	T
168	66	T
170	63	T
170	64	T
170	68	T

Based on data from https://www.listendata.com/2017/12/k-nearest-neighbor-step-by-step-tutorial.html

What Does “Nearest” Mean?

“Nearest neighbors” = training instances with the least “distance”.

The choice of “distance function” is critical!

Some commonly used distances $d(\boldsymbol{x_1}, \boldsymbol{x_2})$ are:

$$\left(\sum_j \left(|x_{1j} - x_{2j}|\right)^1\right)^{\frac{1}{1}}$$

ℓ_1 distance

$$\sum_j |x_{1j} - x_{2j}|$$

$$\left(\sum_j \left(|x_{1j} - x_{2j}|\right)^2\right)^{\frac{1}{2}}$$

ℓ_2 distance

Also, “Euclidean” distance

$$\left(\sum_j \left(|x_{1j} - x_{2j}|\right)^{\to\infty}\right)^{\to 0}$$

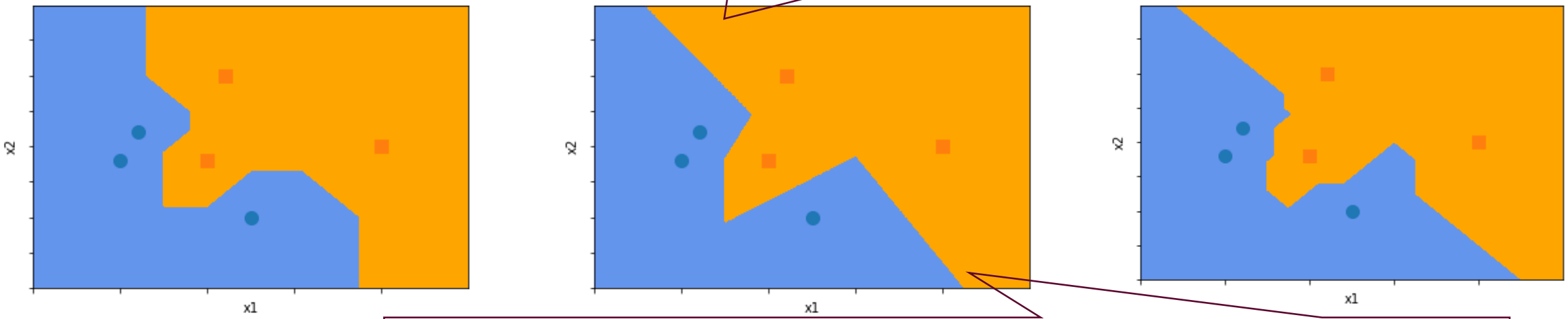
ℓ_∞ distance

$$\max_j \left(|x_{1j} - x_{2j}|\right)$$

Different distances produce different outcomes

Fix k =1 neighbors

Classifier "decision boundary" plots show what class would be assigned at *every point* $\boldsymbol{x}$



Predictions are usually less reliable when the nearest neighbors are far away …

ℓ_1 distance

$$\sum_j |x_{1j} - x_{2j}|$$

ℓ_2 distance

Also, "Euclidean" distance

ℓ_∞ distance

$$\max_j \left(|x_{1j} - x_{2j}|\right)$$

Where Are The Learned "Parameters" in K-NN?

- Think broadly of the "parameters" as everything required at test time to produce the output, for a given model class. i.e.
  - Model class + parameters + new input $\boldsymbol{x} \rightarrow$ predicted y

"kNN classifier" ??

A: The full training dataset!

Funnily, methods like these where the parameters are either the training data itself, or instead grow in size "automatically" with the training data, are called "**non-parametric**" machine learning approaches.

When Is The Training Phase in kNN?

- There is no explicit “training” phase!*
  - The moment we have the dataset, we are ready to produce predictions for new input data!

* caveat: some “approximate nearest neighbors” involve a dataset preprocessing phase that may be thought of as training.

Where Are The Hyperparameters in KNN?

- Distance function
  - Often Euclidean distance by default
    - (unless you want to encode some special information you have about what features are more/less important for this problem. This is relatively uncommon.)

- Choice of k
  - Usually need to find through cross-validation
    - Small values easily affected by noisy data.
    - Large values make it difficult to model sharp changes in the true function.
  - For binary classification, usually an odd number to avoid ties.

Tricky Q: What Is The Hypothesis Space in K-NN?

Bonus Exercise: What is the hypothesis space of a nearest neighbor binary classifier, with some known k (say, 1) and some fixed distance function (say, ℓ_2)?

- Hint:
  - First answer this with a training dataset containing fixed 2D input features $\{x_i\}_{i=1}^N$.
  - Different label assignments $\{y_i\}_{i=1}^N$ to these N points induce different nearest neighbor classification functions.
  - Can you characterize the space of all such functions in terms of $\{x_i\}_{i=1}^N$?

Come to my office hours to discuss!

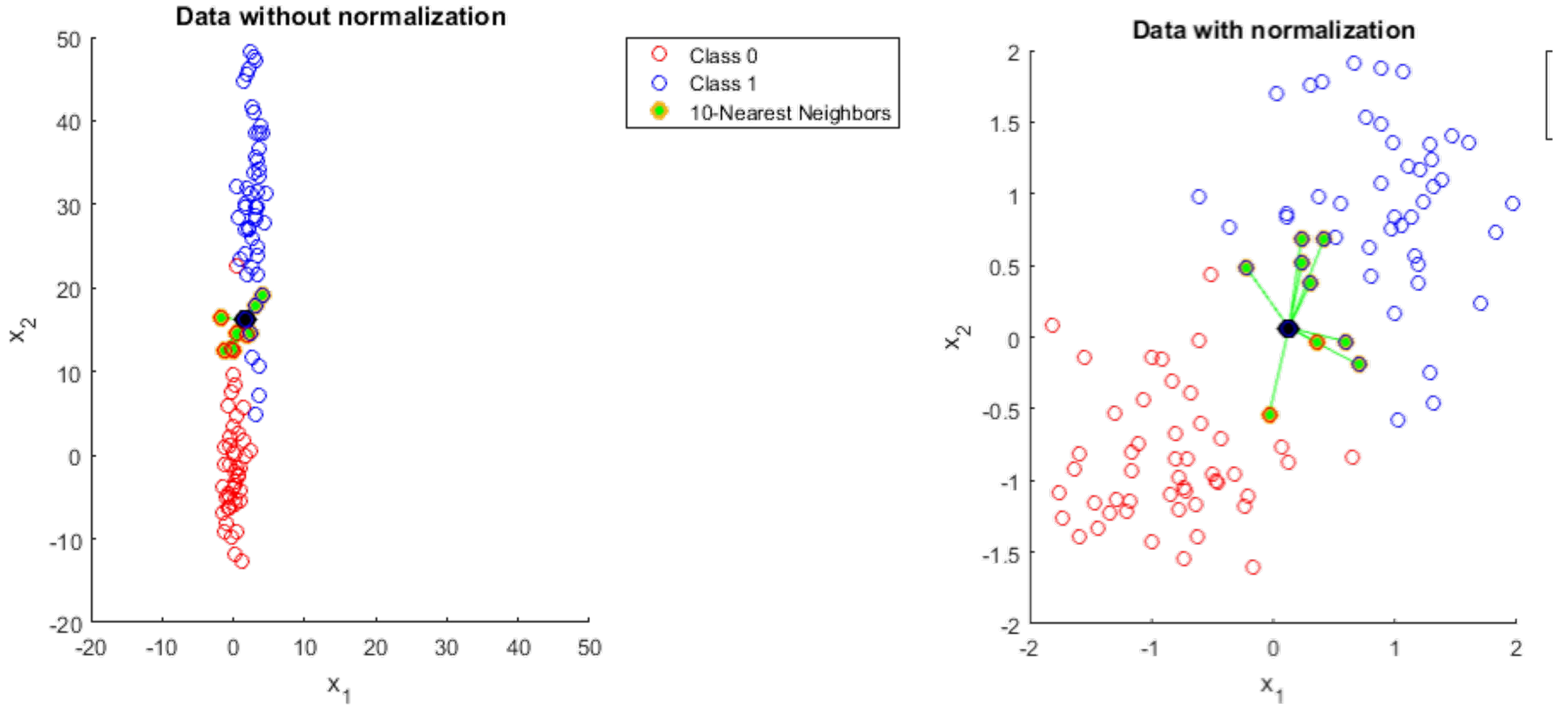
An Excellent First Algorithm To Try

- For many applied machine learning problems, it is useful to think of k-Nearest Neighbors as the algorithm choice for your first attempt.
  - Very easy to write in code.
  - Versatile, does not impose specific restrictions on the learned function, such as “linearity”.
  - Very easy to interpret the outcomes because of the direct connection to training data.

- Often works surprisingly well!
- kNN is not without its problems, of course. But more on that later.

Aside: Scale Invariance in kNN

- kNN approaches are not inherently invariant to feature scaling.
  - E.g. if distance measure is L_2, and one feature in the data is scaled 100x, it suddenly plays a much bigger role than before in determining what neighbors are "nearest".
- Same solution works as before: feature standardization / "normalization".



https://stats.stackexchange.com/questions/287425/why-do-you-need-to-scale-data-in-knn

Aside: kNN Distance Functions for String Data Types

Hamming distance (number of characters that are different)

ABCDE vs AGDDF → 3

Edit distance (number of character inserts/replacements/deletes to go from one to the other)

ROBOT vs BOT → 2

Jaccard distance between sets $\frac{|A \cap B|}{|A \cup B|}$

between **n-grams** (n-character substrings of the strings, with (n-1) character padding)

$$ROBOT$$ vs $$BOT$$ → $\frac{3}{9}$ |{BOT,OT$,T$$}| / |{$$R,$RO,ROB,OBO,$$B,BO,BOT,OT,T$$}|

Aside: Probabilistic Predictions From kNN Classifiers

- Easy to extend to produce probabilistic predictions too.
- One example: for a multi-class classification problem:
  - Find k nearest neighbors
  - Set P(class i) = 1/k * number of instances of class i among the neighbors.

- More sophisticated approaches are possible, e.g., by sorting the k neighbors by distance, and assigning most importance to the closest neighbors.