YData: Introduction to Data Science

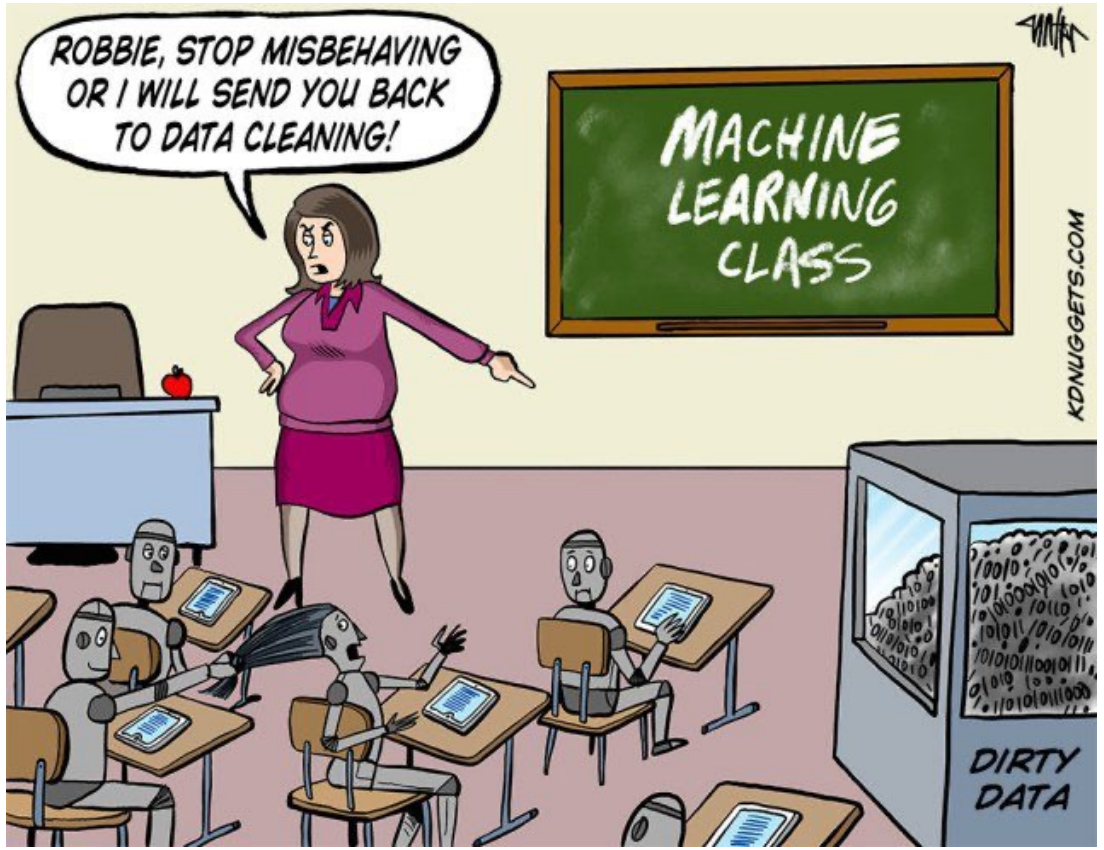
Lecture 23: Classification

Overview

Quick review and continuation of creating confidence intervals using hypothesis tests

Classification

- KNN classifier
- If there is time: ovefitting



Project timeline

~~Sunday, April 7th~~

- ~~Projects are due on Gradescope at 11pm~~
- Email a pdf of your project to your peer reviewers
  - A list of whose paper you will review is on Canvas
  - Fill out the draft reflection on Canvas

Wednesday, April 17th

- Jupyter notebook files with your reviews need to be sent to the authors
- A template for doing your review is available

Sunday, April 28th

- Project is due on Gradescope
  - Add peer reviews to the Appendix of your project

Homework 9 has been posted

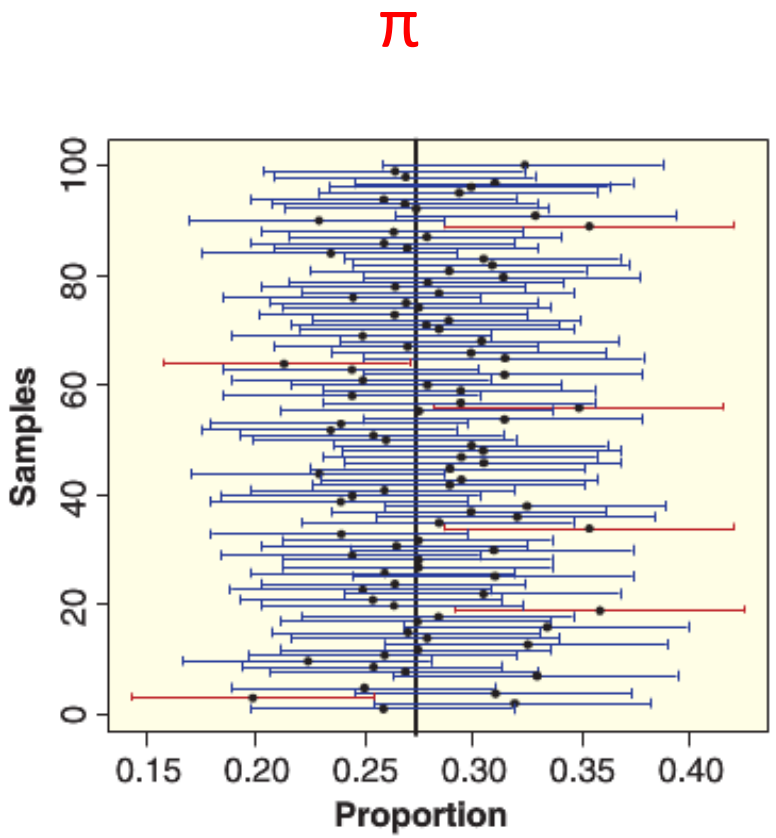
- It is due April 21st

Quick review: confidence Intervals

A **confidence interval** is an interval computed by a method that will contain the *parameter* a specified percent of times

- i.e., if the estimation were repeated many times, the interval will have the parameter x% of the time

The **confidence level** is the percent of all intervals that contain the parameter

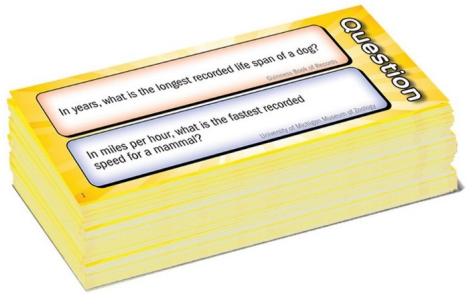


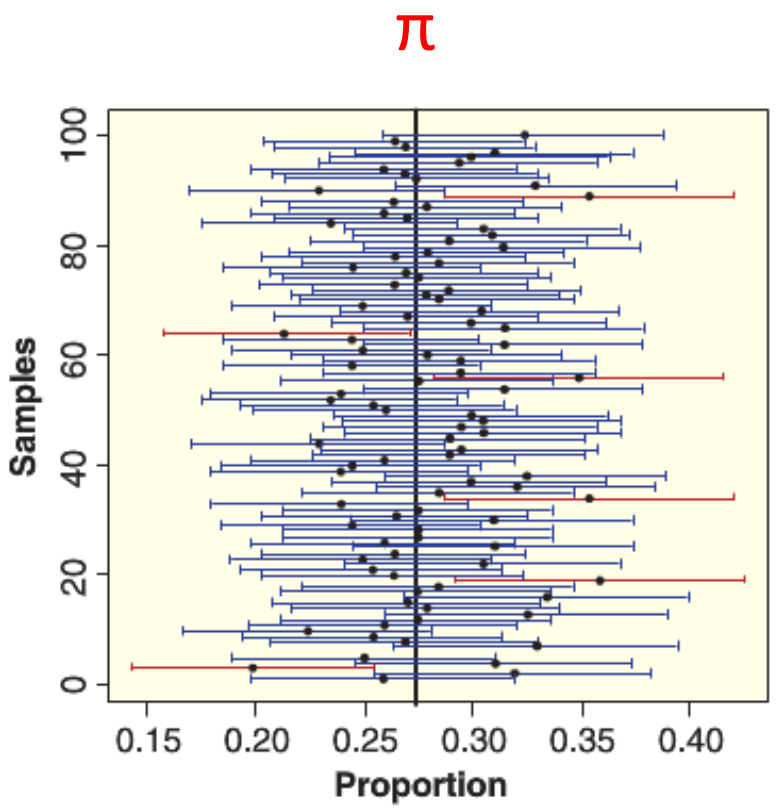
90% Confidence Intervals

For any given confidence interval, we don't know of it has the parameter in it

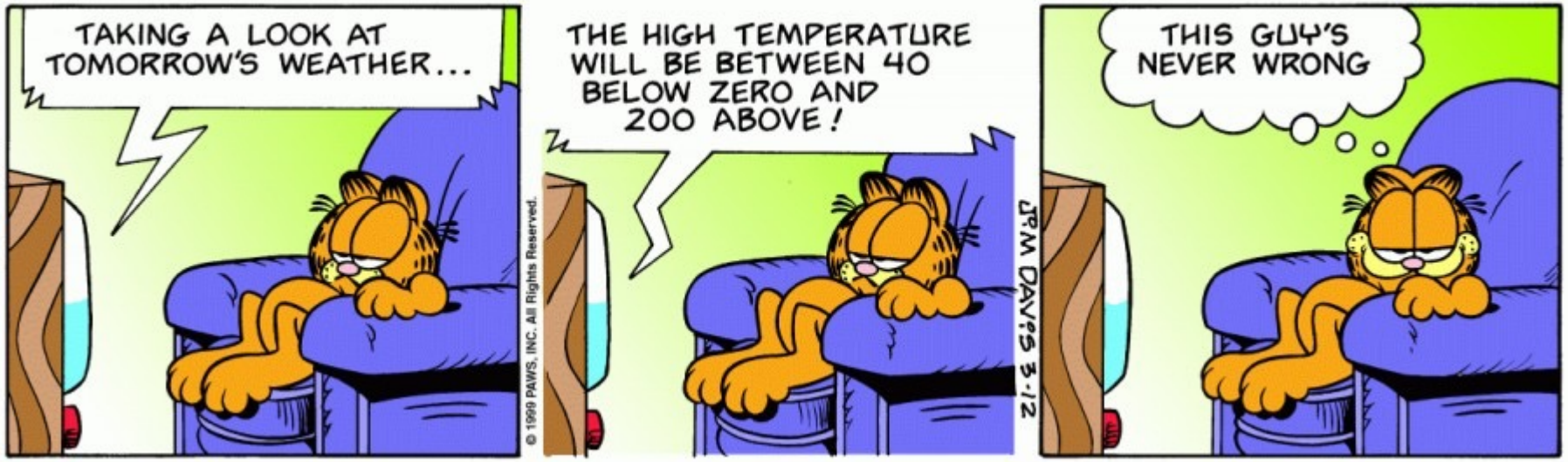
We just know that it will be in the interval most of the time

- E.g., 9 out of 10 times for a 90% confidence level





Tradeoff between interval size and confidence level



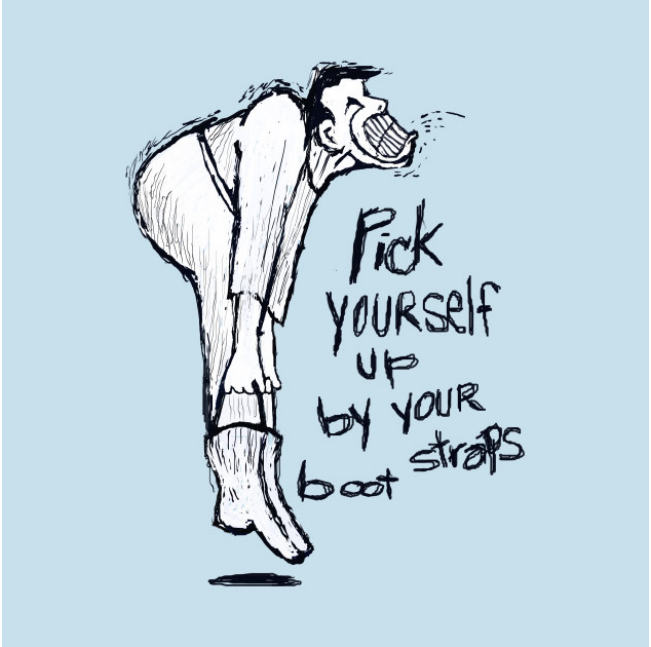
There is a tradeoff between the **confidence level** (percent of times we capture the parameter) and the **confidence interval size**

Review: using hypothesis tests to construct confidence intervals

Constructing confidence intervals

There are several methods that can be used to construct confidence intervals including

- "Parametric methods" that use probability functions
  - E.g., confidence intervals based on the normal distribution
- A "bootstrap method" where data is resampled from our original sample to approximate a sampling distribution



To learn more about these methods, take Introductory Statistics!

Constructing confidence intervals

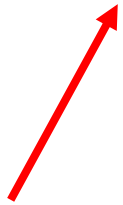
We are going to use a less conventional method to get confidence intervals based on the relationship between confidence intervals and hypothesis tests

- The method we will discuss is valid, but can be more computationally expensive than other methods

What we will do is to run a series of hypothesis test with different null hypothesis parameter values

Our confidence interval will be all parameters values where we **fail to reject** the null hypothesis

$H_0: \pi = \pi_0$



Failure to reject $\pi = \pi_0$ means π_0 is plausible

Motivation: Bechdel Confidence Interval

From running a hypothesis test on the Bechdel data, we saw that H_0: $\pi = .5$ is unlikely

- i.e., it was not plausible that 50% of movies pass the Bechdel test

But what is a reasonable range of values for the population proportion of movies that pass the Bechdel test?

We can create a confidence interval for $\pi_{Bechdel}$ to find out!

Very quick review of using hypothesis tests to construct confidence intervals

All parameter values where we fail to reject the null hypothesis make up the confidence interval

- Using a threshold of p-value < 0.05 yields a 95% CI
- Using a threshold of p-value < 0.01 yields a 99% CI

Let's quickly explore this in Jupyter!

π	p-values
0.41	0
0.41	0.0013
0.42	0.0179
0.43	0.1361
0.44	0.5269
0.45	0.85
0.46	0.296
0.47	0.0614
0.48	0.0067
0.49	0.0004

Classification

Prediction: regression and clasification

We “learn” a function f

- f(**x**) ⟶ y

Input: **x** is a data vector of "features"

Output:

- Regression: output is a real number (y ϵ R)
- Classification: output is a categorical variable y_k

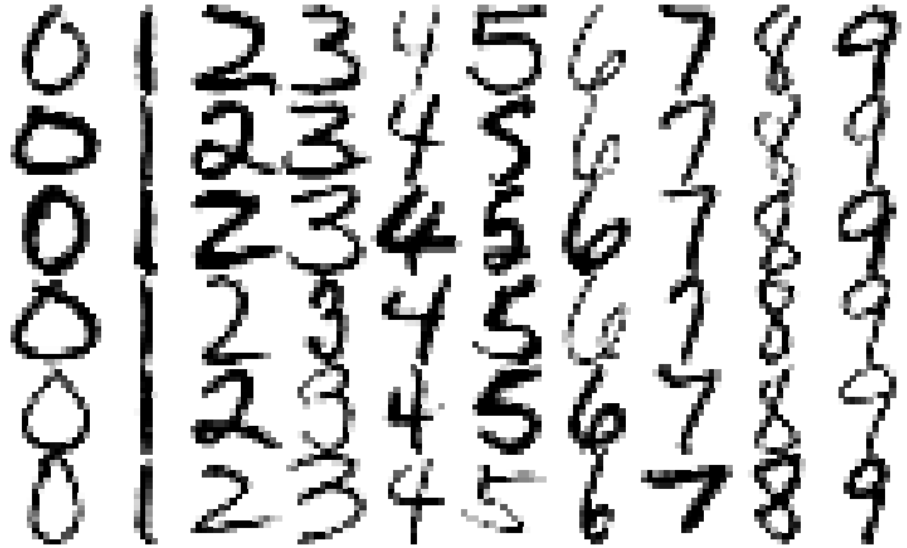
Example: salmon or sea bass?

What are the labels and features in this task?

- Labels (y): Salmon or Sea bass
- Features (X): Length, width, lightness, number and shape of fins, etc.

From Duda, Hart and Stork, 2001

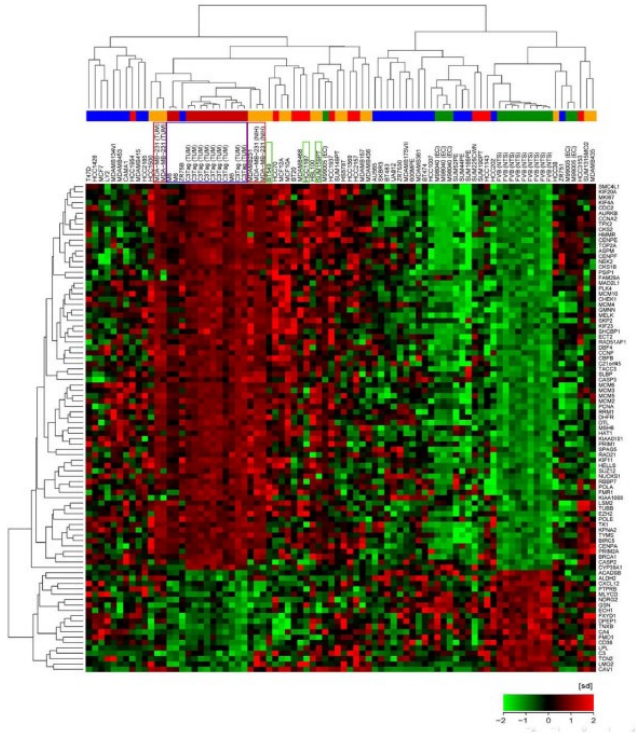
Example: what is in this image?



What are the labels and features in this task?

- Labels (y): cat, dog, etc.,   or numbers 0 to 9
- Features (X): Pixel values

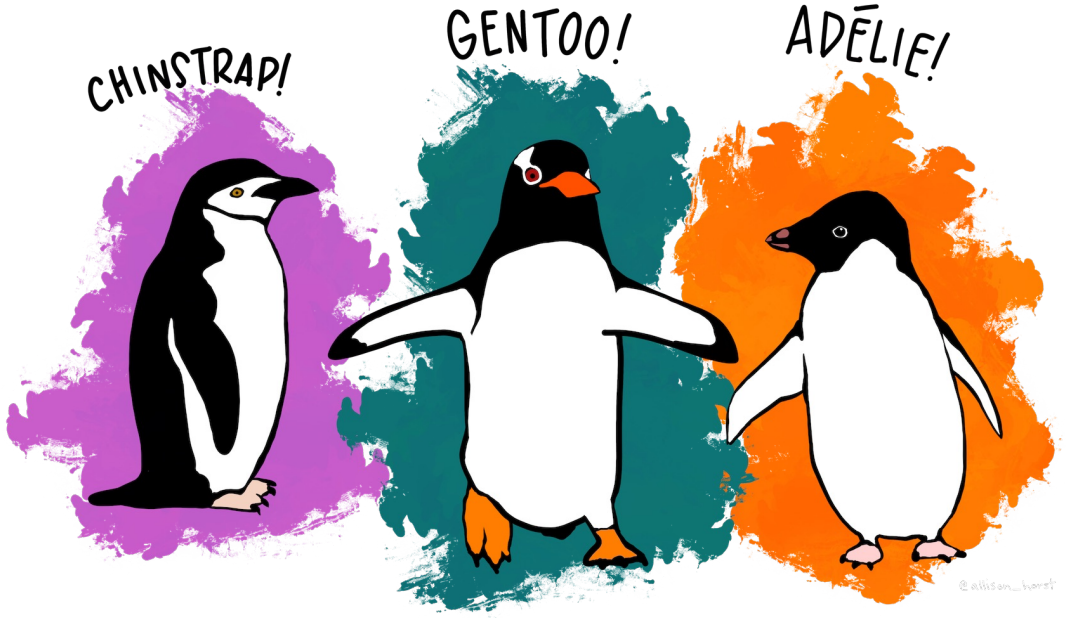
Example: predicting cancer



What are the labels and features in this task?

- Labels (y): Cancer/no caner
- Features (X): The expression level of different genes

Example: Penguin species



What are the labels and features in this task?

- Labels (y): Chinstrap, Gentoo, Adelie
- Features (X): Flipper length, bill length, body mass, …

Example: GPT-3 predicting/generating text

Question answering:

Are we living in a simulation?

Image generation

"Draw an astronaut riding a horse"

What are the labels and features in this task?

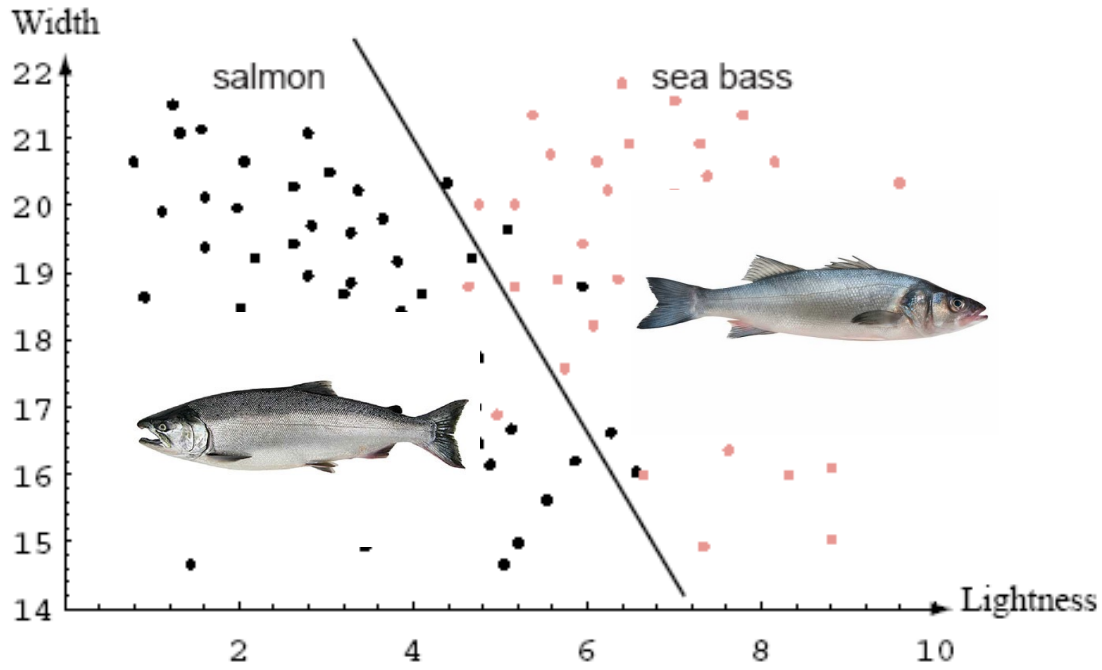
- Labels (y): Next word in a sentence (or an image)
- Features (X): Previous words/prompt words

Example of classifier on a feature set

A binary classifier is a function from the set of input features to {0, 1}

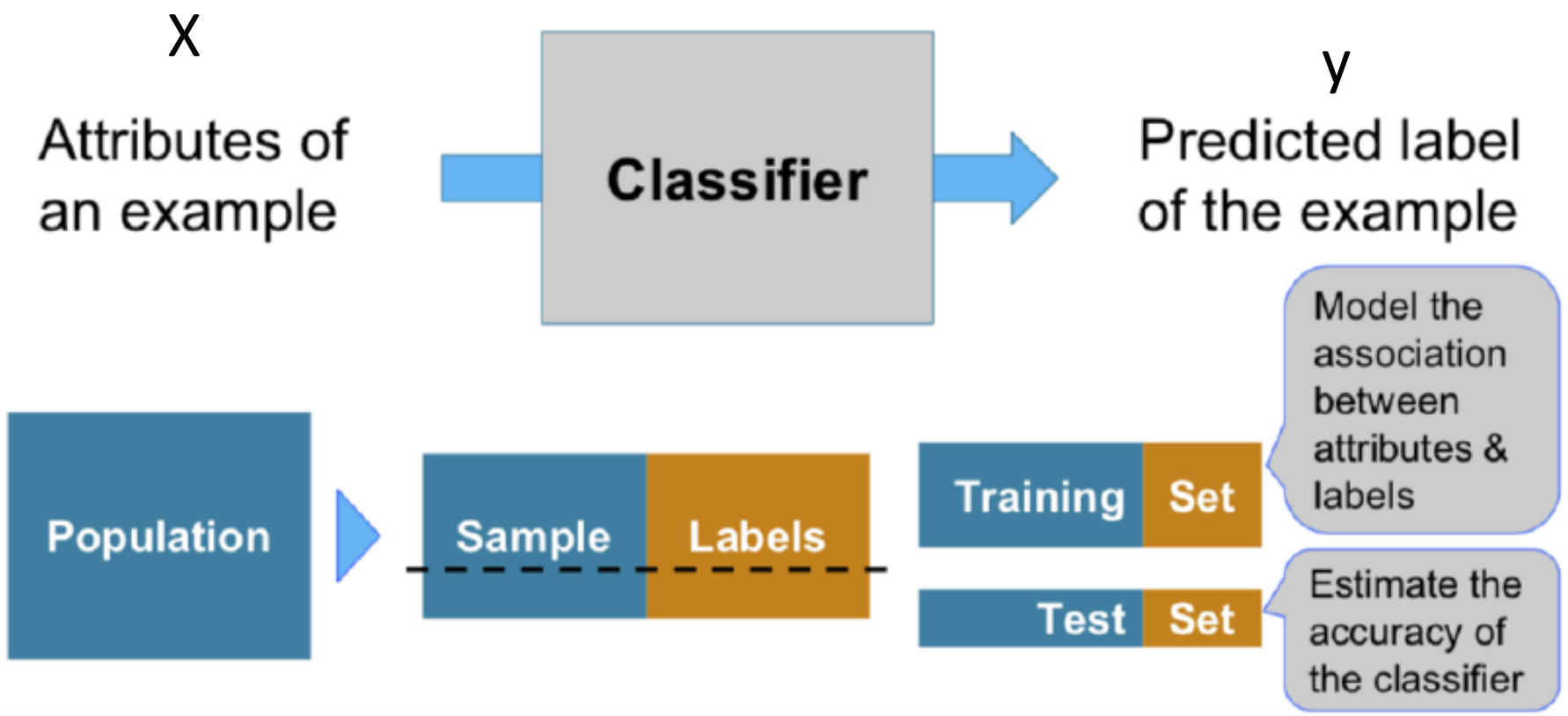
- E.g., f(pixel values) ⟶ salmon or bass

A linear classifier draws a straight line (or a multi-dimensional plane) between the two predicted values



Let's explore features and labels in Jupyter!

Training a classifier



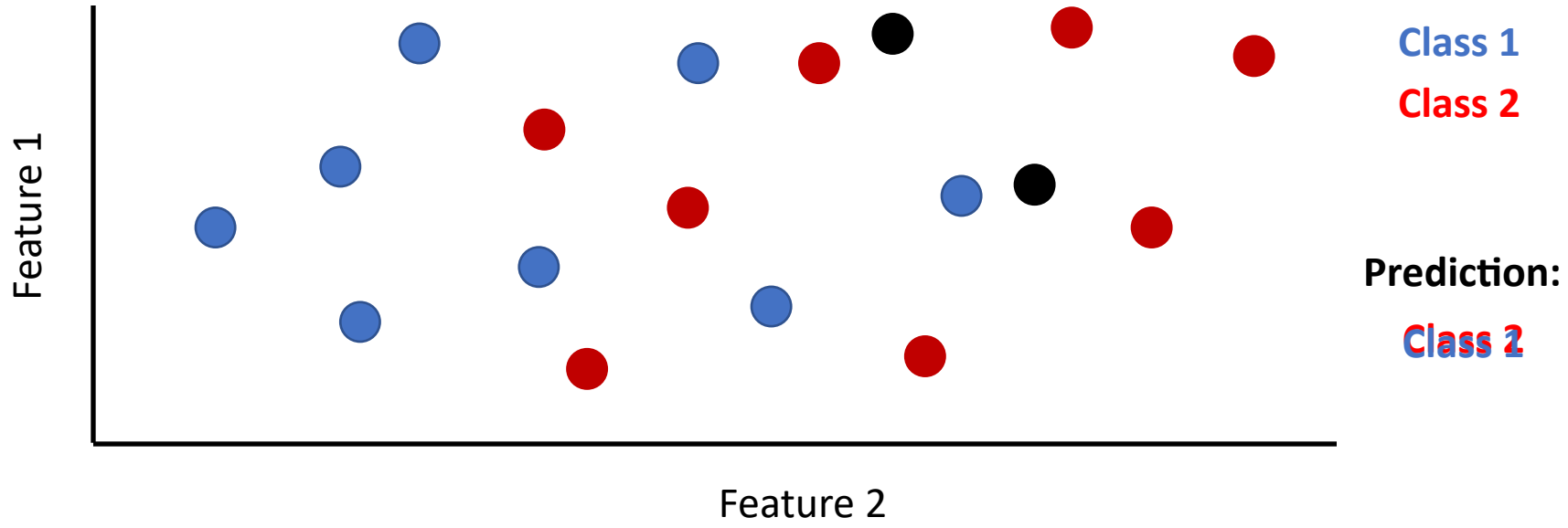
k-Nearest Neighbor classifier

Nearest Neighbor Classifier (k = 1)

Training the classifier: Store all the features with their labels

Making predictions: The label of closest training point is returned

- i.e., we find the distance between a test point and all training points, and the closest point is the prediction



Distance between two points

Two features x and y: $D = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}$

Three features x, y, and z: $D = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2 + (z_0 - z_1)^2}$

- And so on for more features…

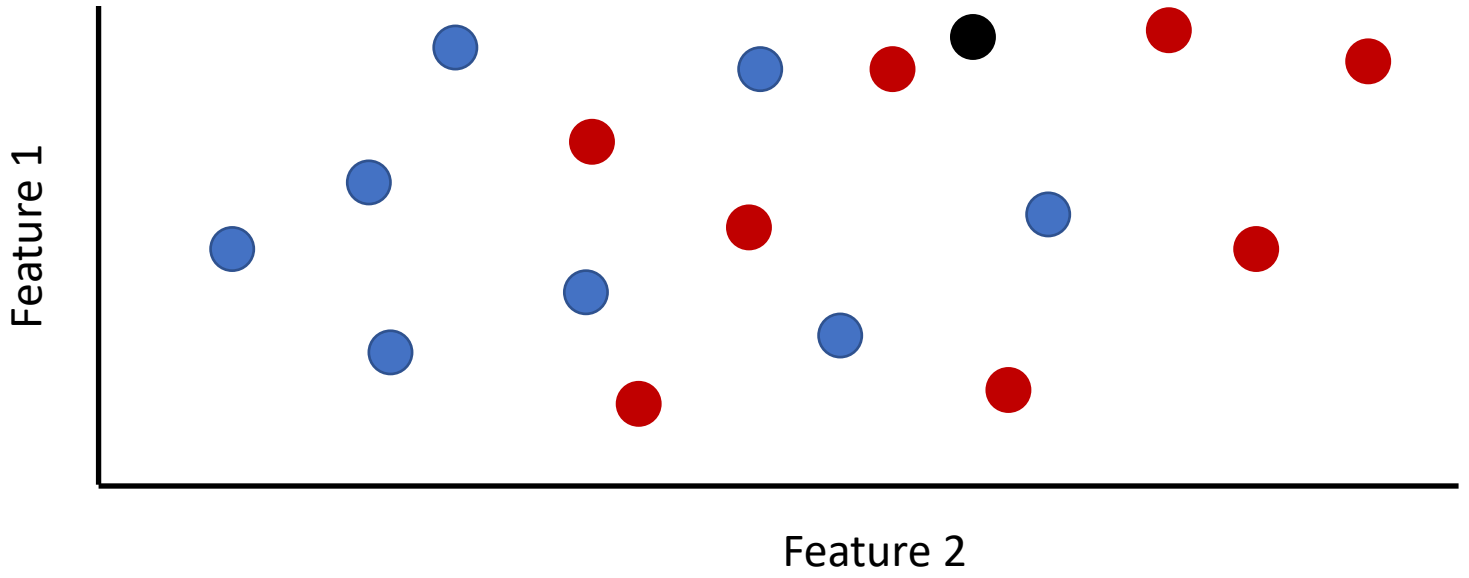
Side note: It's important the features are standardized

- If not, features that typically have larger values will dominate the distance measurement

Finding the k Nearest Neighbors (k ≥ 1)

To classify a point:

- Find its k nearest neighbors
- Take a majority vote of the k nearest neighbors to see which of the two classes appears more often
- Assign the point the class that wins the majority vote



Let's explore this in Jupyter!

Evaluation

Accuracy of a classifier

The accuracy of a classifier on a labeled data set is the proportion of examples that are labeled correctly on the ***test set***

If the labeled data set is sampled at random from a population, then we can infer accuracy on that population

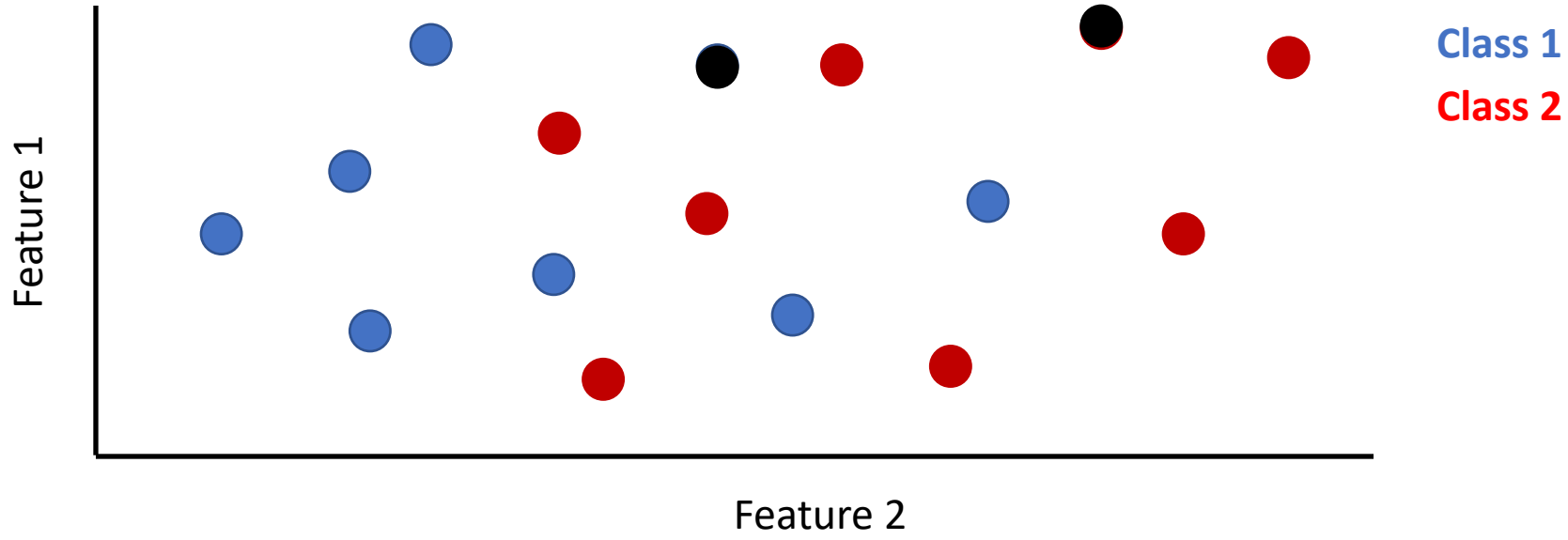
Training and test accuracy

Q: What would happen if we tested the classifier using the training data with k = 1?

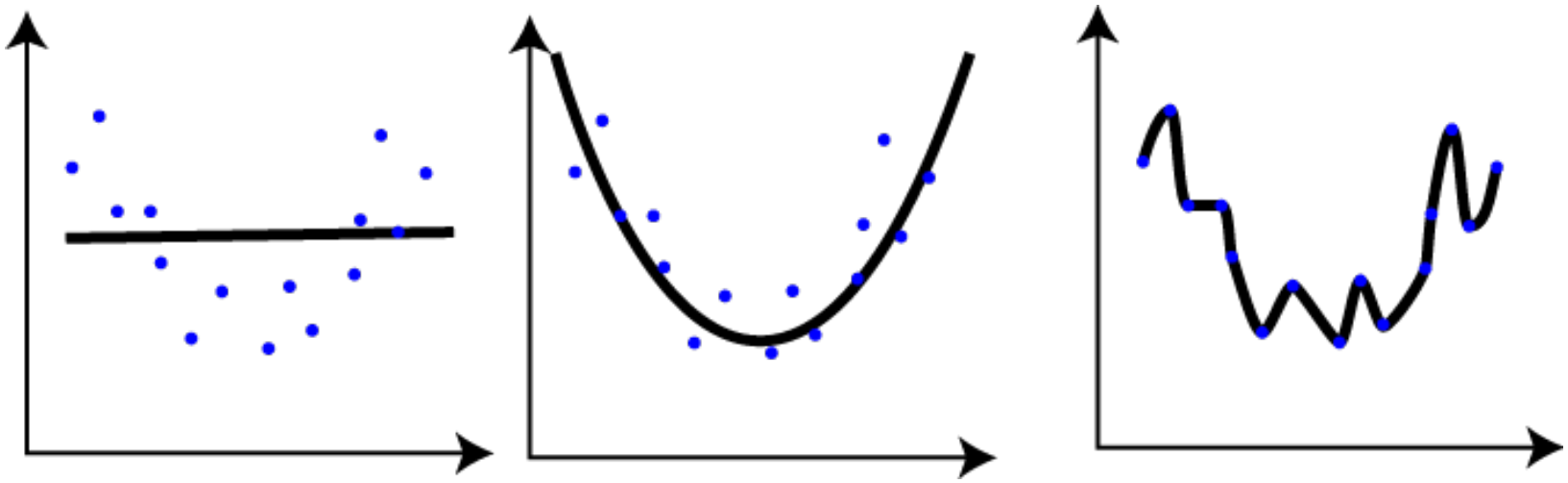
A: We would have 100% accuracy

Q: Would this indicate that the classifier is good?

- A: No!



Overfitting



Overfitting

If our classifier has over-fit to the training data then:

a) We might not have a realistic estimate of how accurate its predictions will be on new data

b) There might be a better classifier that would not over-fit to the data and thus can make better predictions

What we really want to estimate is how well the classifier will make predictions on new data, which is called the **generalization (or test) error**

Overfitting song…

Cross-validation

Training error rate (training accuracy): model predictions are made on using the same data that the model was fit with

Test error rate (test accuracy): model predictions are made on a separate set of data

k-fold cross-validation

Are there any downsides to using half the data for training and half for testing?

Since we are only using half the data for training, potentially can build a better model if we used more of our data

- Say 90% of our data for training and 10% of testing

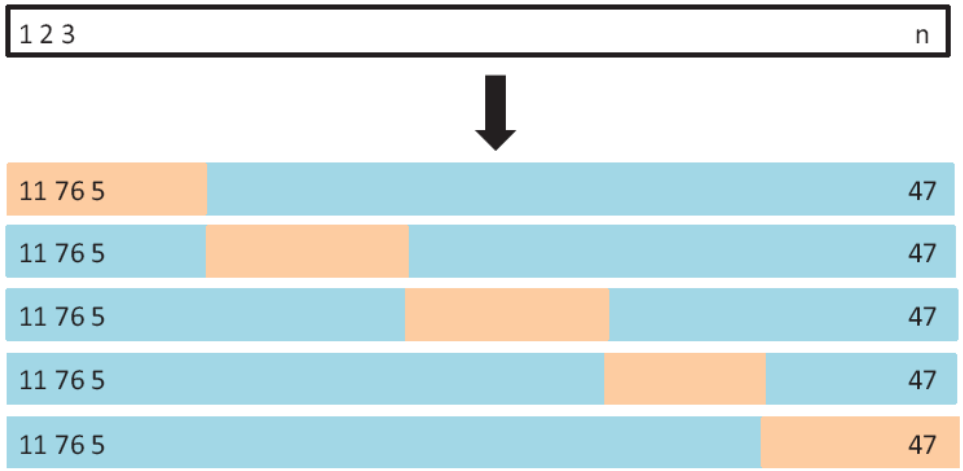
Problem: Then our estimate of the generalization error would be worse

Solutions?

k-fold cross-validation

k-fold cross-validation

- Split the data into k parts
- Train on k-1 of these parts and test on the left out part
- Repeat this process for all k parts
- Average the prediction accuracies to get a final estimate of the generalization error



Let's explore this in Jupyter!

Next class, building the KNN classifier ourselves