YData: Introduction to Data Science

Lecture 24: Classification continued

Overview

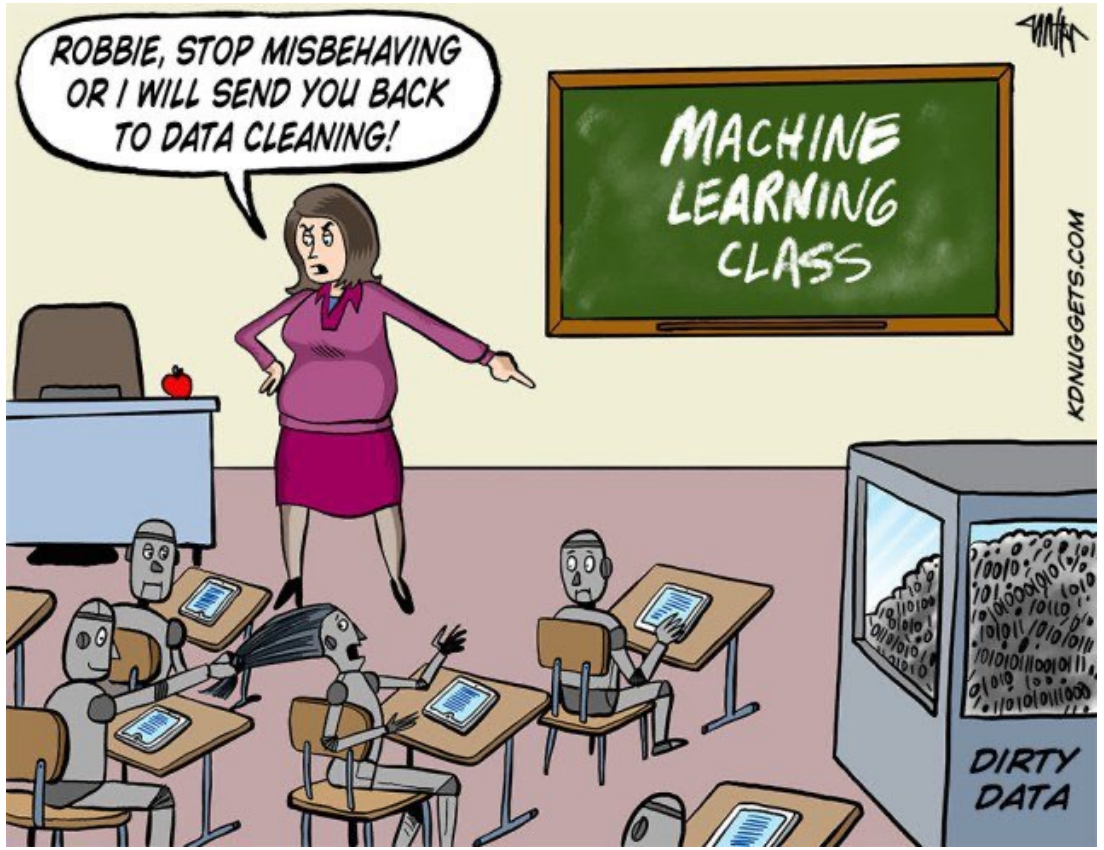
Quick review of creating confidence intervals (using hypothesis tests)

Classification

- KNN classifier
- Overfitting

If there is time

- Building our own KNN classifier



Project timeline

~~Tuesday, April 11th~~
- ~~Projects are due on Gradescope at 11pm on~~
- ~~Also, email a pdf of your project to your peer reviewers~~
  - ~~A list of whose paper you will review has been posted to Canvas~~

Wednesday, April 19th
- Jupyter notebook files with your reviews need to be sent to the authors and a pdf needs to be submitted to Gradescope
- A template for doing your review is available on Canvas

Sunday, April 30th
- Project is due on Gradescope
  - Add peer reviews to an Appendix of your project

Homework 9 has been posted
- It is due April 23rd

Very quick review of using hypothesis tests to construct confidence intervals

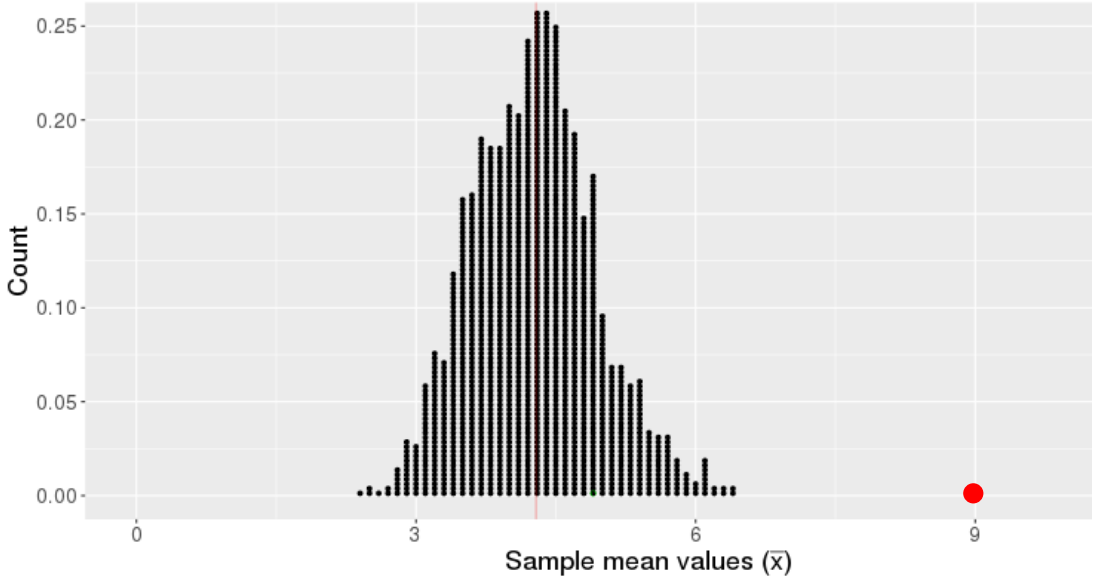
Very quick review of using hypothesis tests to construct confidence intervals

Hypothesis tests tell use whether a particular parameter value is **not** plausible

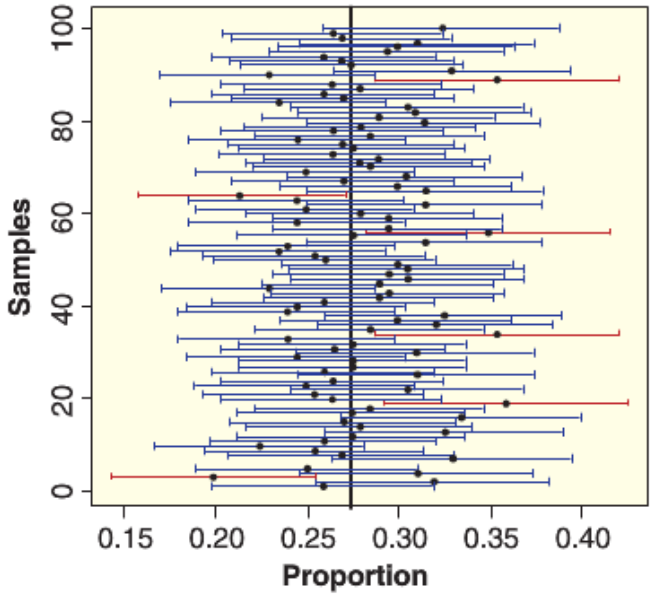
- E.g., μ = [illegible]

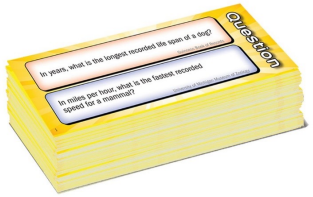
Confidence interval give us a range of plausible parameter values

- Most of the time, the parameter will be within this range



π





Very quick review of using hypothesis tests to construct confidence intervals

Last class we used 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

Questions?

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

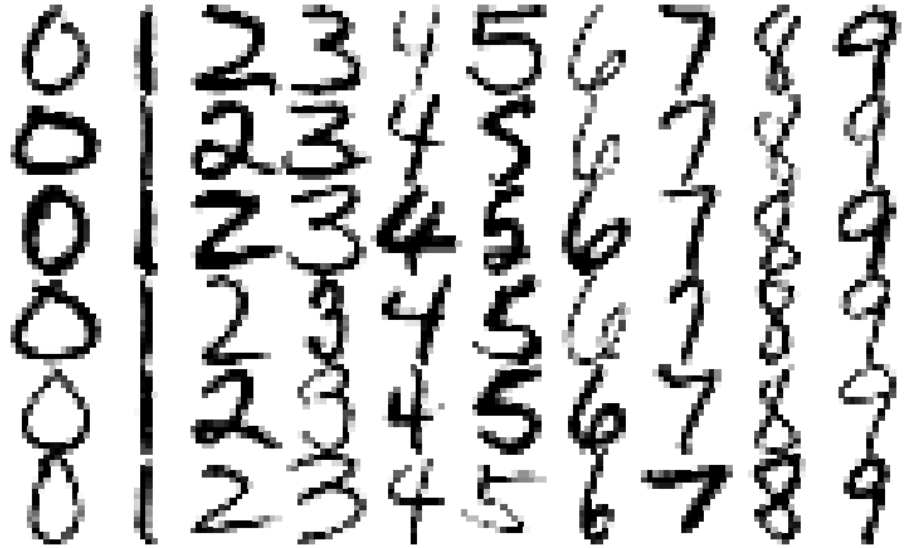
Example: salmon or sea bass?

What are the features and labels 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

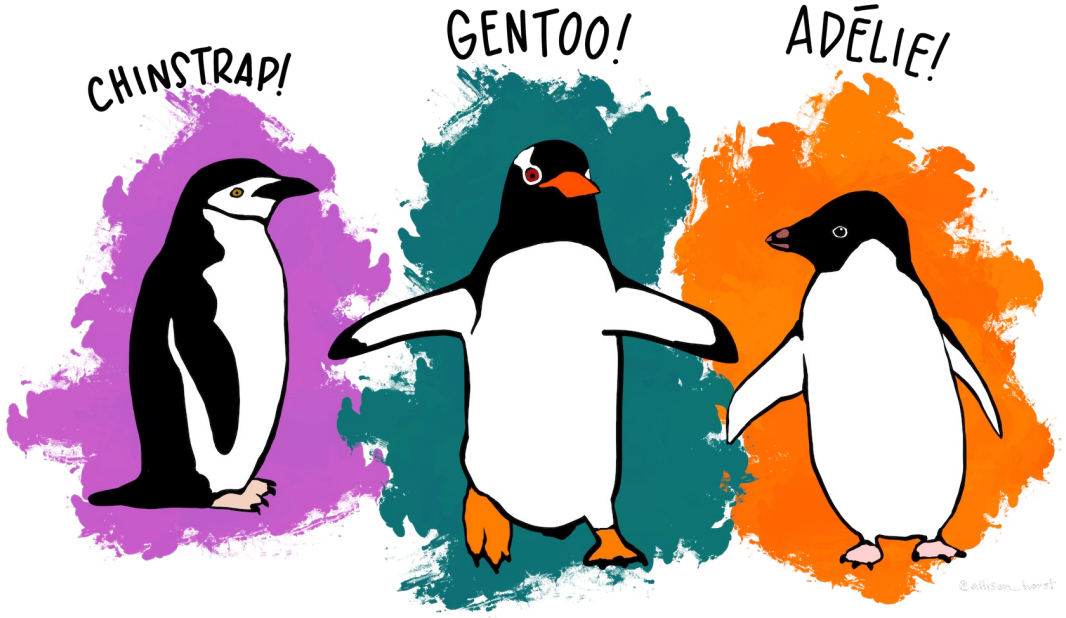
Example: what is in this image?



What are the features and labels in this task?

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

Example: Penguin species



What are the features and labels 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"

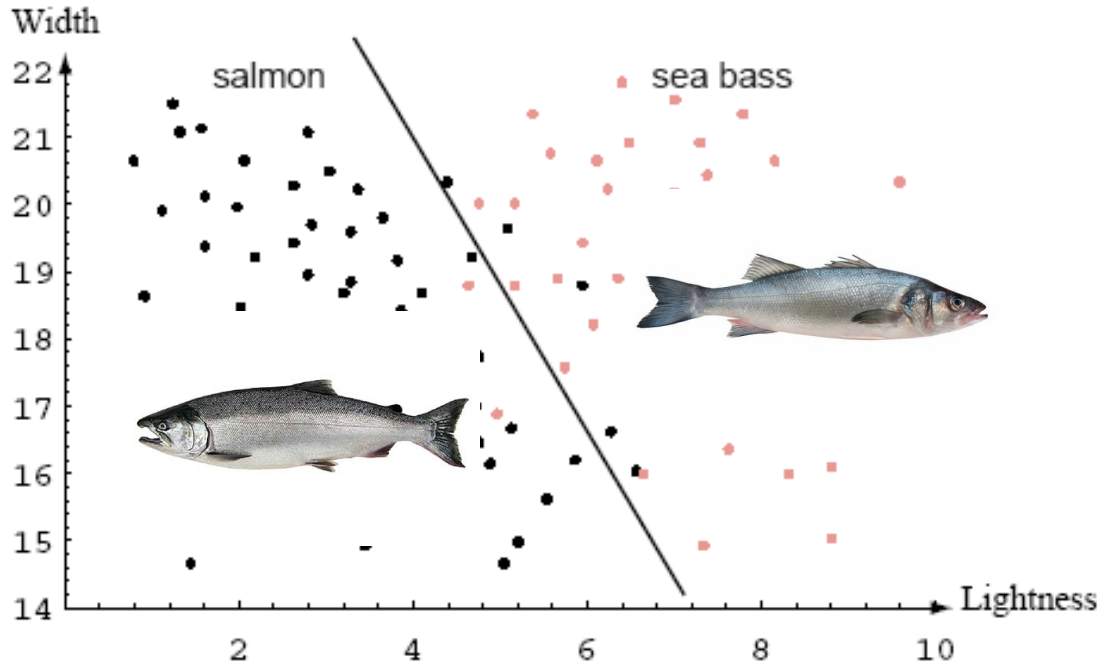
What are the features and labels in this task?

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

Example of classifier on a feature set

A classifier is a decision rule that can be used to separate members of different classes

- E.g. a line to separate salmon from sea bass



Let's explore features and labels in Jupyter!

Training a classifier

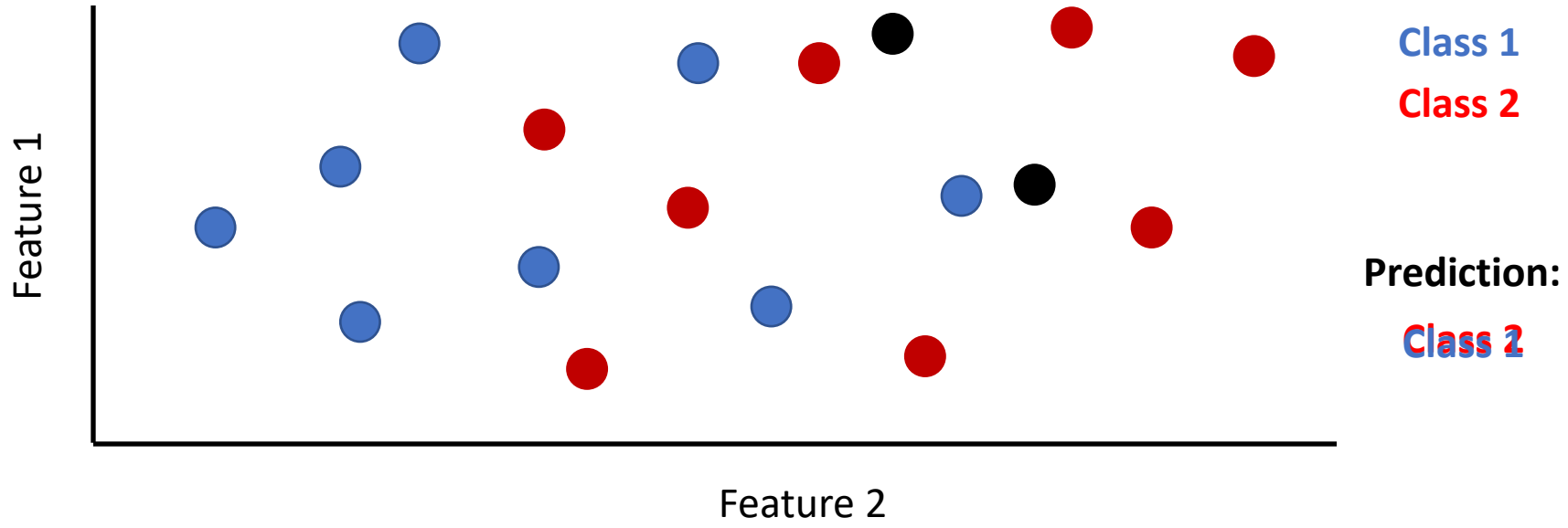
k-Nearest Neighbor classifier

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…

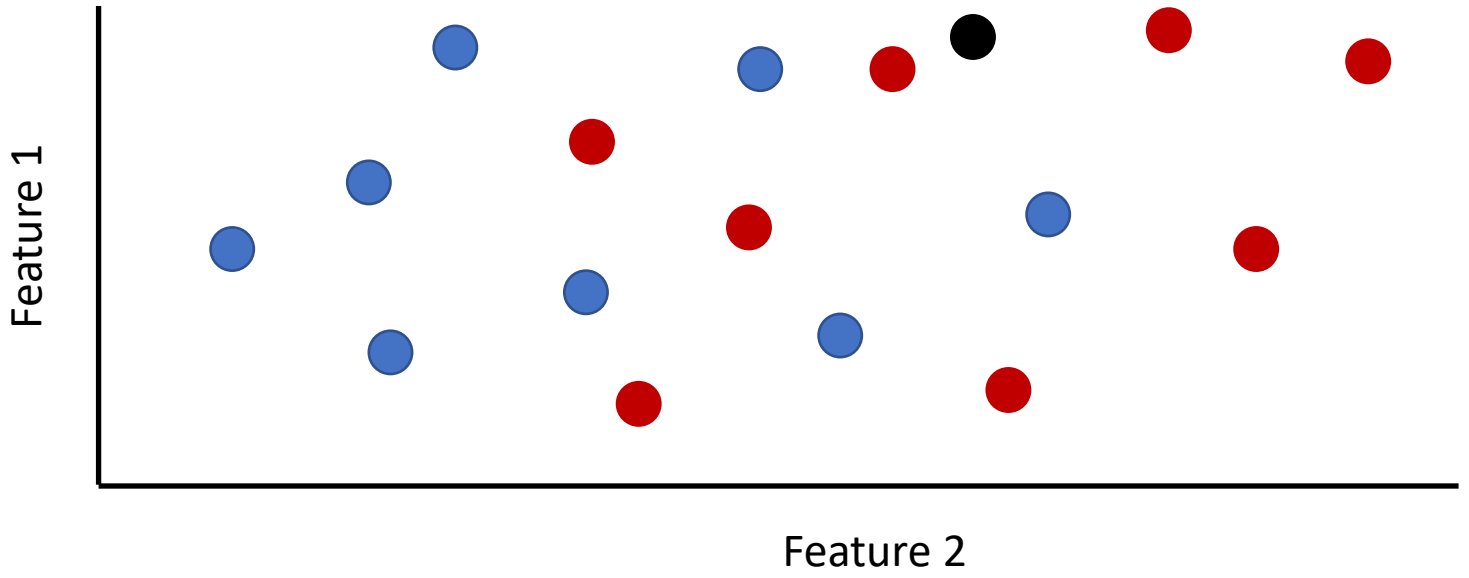
It's important the features are standardized

- If not, features that typically have larger values will dominate the distance measurement
  - You can explore this on the bonus homework problem

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

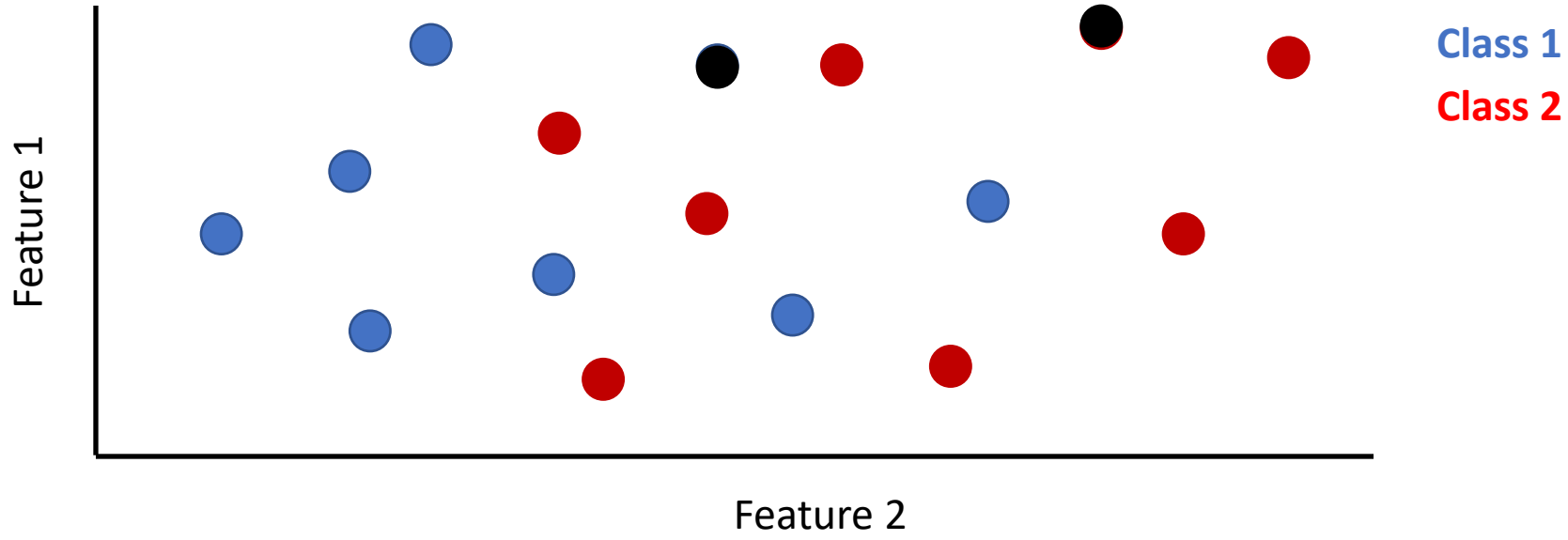
Training and test accuracy

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

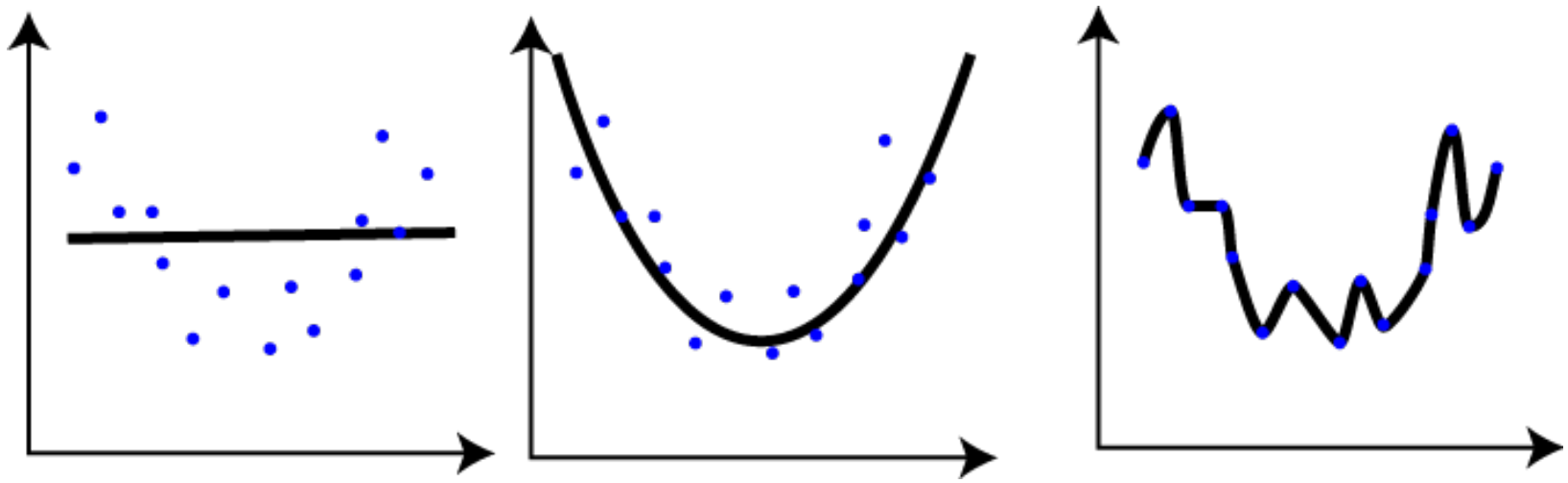
A: We would have 100% accuracy

Q: Would this indicate that the classifier is good?

- A: No!



Review: overfitting



Review: 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…

Review 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

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

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

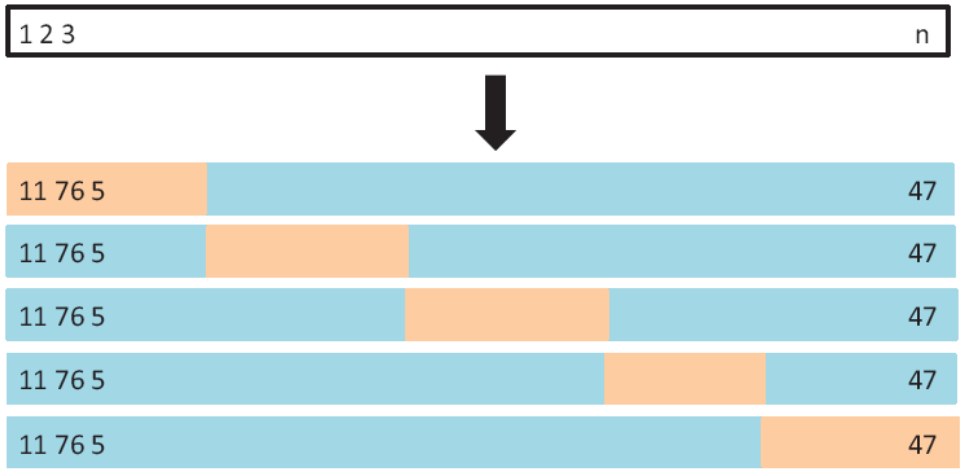
Problem: Then our estimate of the generalization error would be worse

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

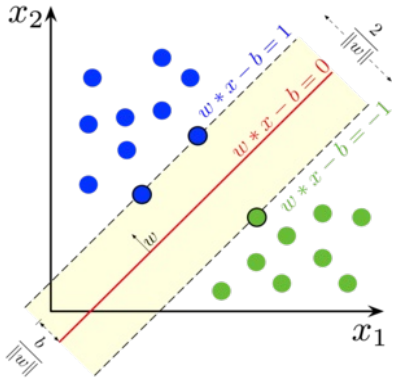


Let's explore this in Jupyter!

Other classifiers

There are many other classification algorithms such as:

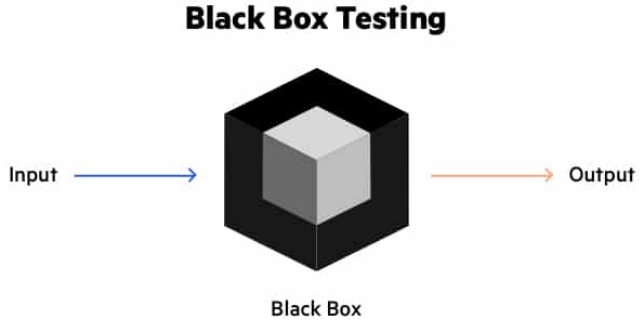
- Support Vector Machines (SVM)
- Decision Trees/Random Forests
- Deep Neural Networks,
- etc.



Scikit-learn makes it easy to try out different classifiers to find one that works best for a particular problem

On homework 9 you will try out SVMs and Random Forests

- (and you can take a machine learning class to understand how they work)



Let's explore this in Jupyter!

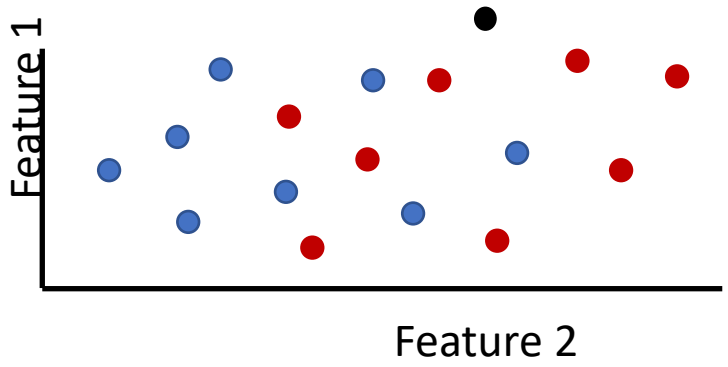
Building the KNN classifier

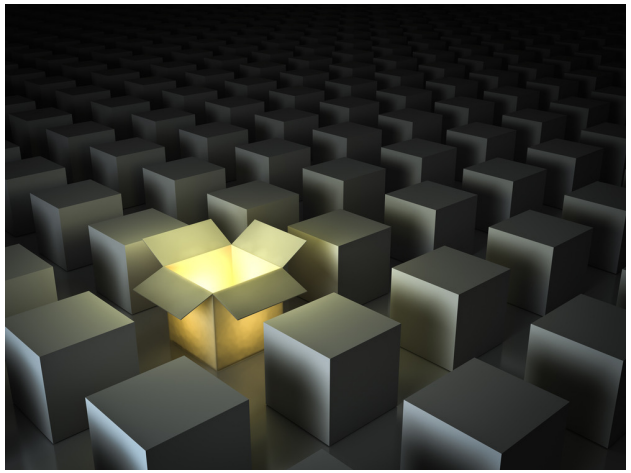
Building the KNN classifier

So far we have used a KNN classifier

- and we have some idea of how it works

Let's now see if we can write to to implement the classifier ourselves…





Steps to build a KNN classifier

We build our KNN classifier by creating a series of functions…

$$D = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2 + (z_0 - z_1)^2}$$

1. euclid_dist(x1, x2)
   - Calculates the Euclidean distance between two points x1 and x2

2. get_labels_and_distances(test_point, X_train_features, y_train_labels)
   - Finds the distance between a test point and all the training points

3. classify_point(test_point, k, X_train_features, y_train_labels)
   - Classifies one test point by returning the majority label of the k closest points

4. classify_all_test_data(X_test_data, k, X_train_features, y_train_labels)
   - Classifiers all test points

Let's explore this in Jupyter!