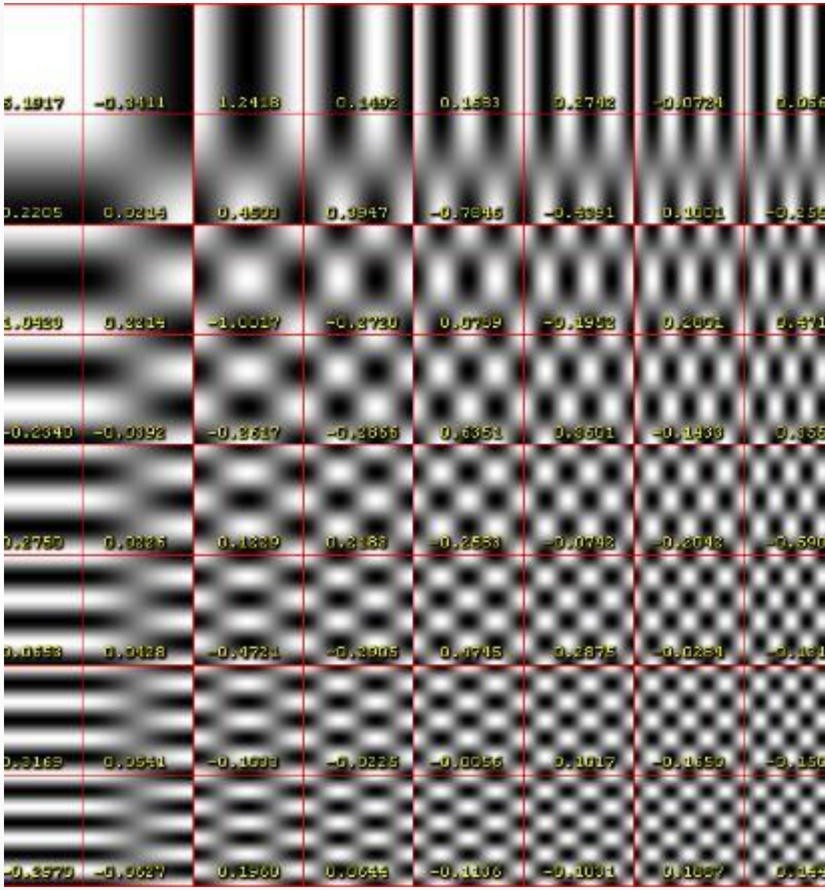
FEATURES AND EVALUATION



許志仲 (Chih-Chung Hsu)

Assistant Professor
Institute of Data Science
National Cheng Kung University

Features

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

Where do they come from?

Provided features

- Predicting the age of abalone from physical measurements
  - Name / Data Type / Measurement Unit / Description
  - -----------------------------
  - Sex / nominal / -- / M, F, and I (infant)
  - Length / continuous / mm / Longest shell measurement
  - Diameter / continuous / mm / perpendicular to length
  - Height / continuous / mm / with meat in shell
  - Whole weight / continuous / grams / whole abalone
  - Shucked weight / continuous / grams / weight of meat
  - Viscera weight / continuous / grams / gut weight (after bleeding)
  - Shell weight / continuous / grams / after being dried
  - Rings / integer / -- / +1.5 gives the age in years

Provided features

- Predicting breast cancer recurrence
  - 1. Class: no-recurrence-events, recurrence-events
  - 2. age: 10-19, 20-29, 30-39, 40-49, 50-59, 60-69, 70-79, 80-89, 90-99.
  - 3. menopause: lt40, ge40, premeno.
  - 4. tumor-size: 0-4, 5-9, 10-14, 15-19, 20-24, 25-29, 30-34, 35-39, 40-44, 45-49, 50-54, 55-59.
  - 5. inv-nodes: 0-2, 3-5, 6-8, 9-11, 12-14, 15-17, 18-20, 21-23, 24-26, 27-29, 30-32, 33-35, 36-39.
  - 6. node-caps: yes, no.
  - 7. deg-malig: 1, 2, 3.
  - 8. breast: left, right.
  - 9. breast-quad: left-up, left-low, right-up, right-low, central.
  - 10. irradiated: yes, no.

Provided features

- In many physical domains (e.g. biology, medicine, chemistry, engineering, etc.)
  - the data has been collected and the relevant features identified
  - we cannot collect more features from the examples (at least “core” features)

- In these domains, we can often just use the provided features

Raw data vs. features

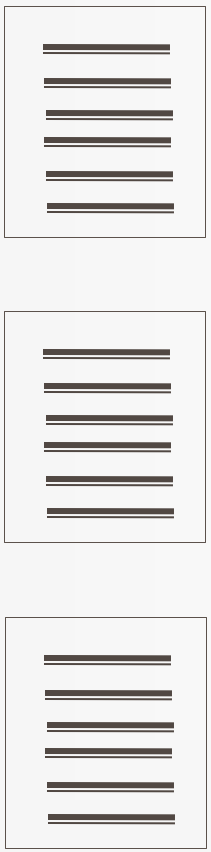
- In many other domains, we are provided with the raw data, but must extract/identify features

- For example
  - image data
  - text data
  - audio data
  - log data
  - …

Text: raw data

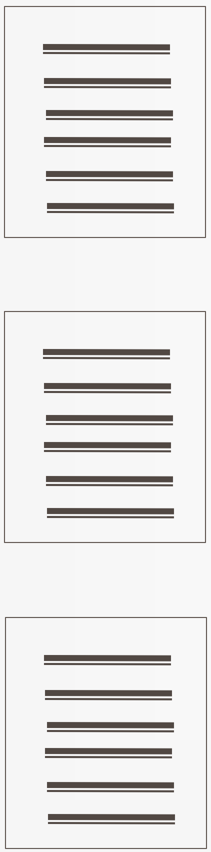
Raw data

Features?



Feature examples

Raw data



Features

Trump said banana repeatedly last week on tv, “banana, banana, banana”

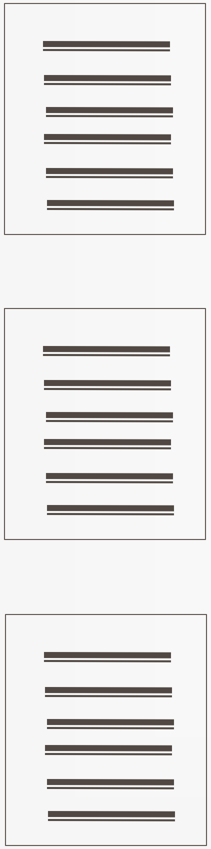
(1, 1, 1, 0, 0, 1, 0, 0, ...)

banana Trump said California across tv wrong capital

Occurrence of words

Feature examples

Raw data



Features

Trump said banana repeatedly last week on tv, “banana, banana, banana”

(4, 1, 1, 0, 0, 1, 0, 0, ...)

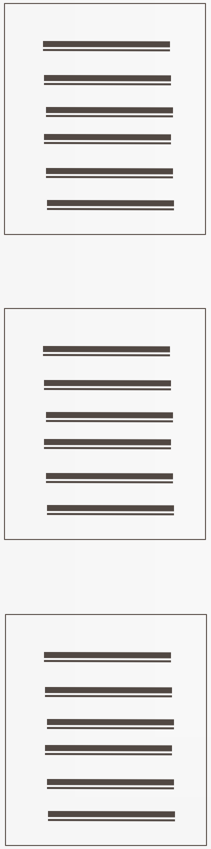
banana Trump said California across tv wrong capital

Frequency of word occurrence

Do we retain all the information in the original docume

Feature examples

Raw data



Features

Trump said banana repeatedly last week on tv, “banana, banana, banana”

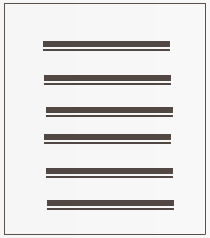
(1, 1, 1, 0, 0, 1, 0, 0, ...)

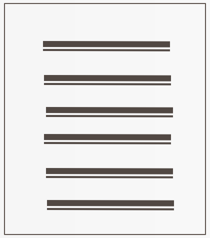
banana repeatedly
Trump said
said banana
California schools
across the
tv banana
wrong way
capital city

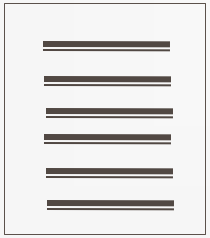
Occurrence of bigrams

Feature examples

Raw data







Features

Trump said banana repeatedly last week on tv, “banana, banana, banana”

(1, 1, 1, 0, 0, 1, 0, 0, ...)

banana repeatedly
Trump said
said banana
California schools
across the
tv banana
wrong way
capital city

Other features?

Lots of other features

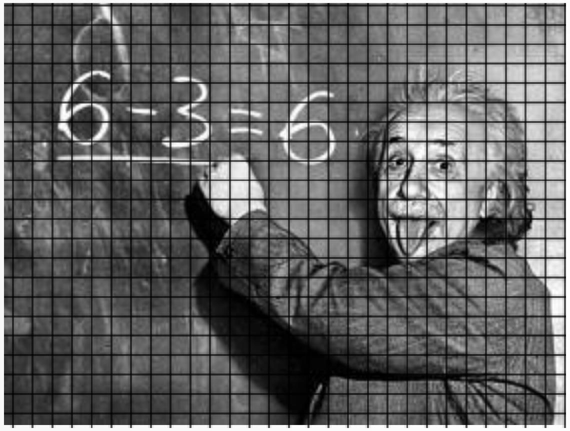
- POS: occurrence, counts, sequence
- Constituents
- Whether ‘V1agra’ occurred 15 times
- Whether ‘banana’ occurred more times than ‘apple’
- If the document has a number in it
- …
- Features are very important, but we’re going to focus on the models today

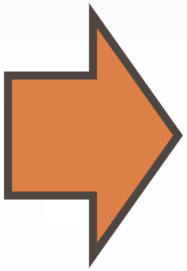
How is an image represented?

How is an image represented?

- images are made up of pixels
- for a color image, each pixel corresponds to an RGB value (i.e. three numbers)

Image features





for each pixel:

R[0-255]

G[0-255]

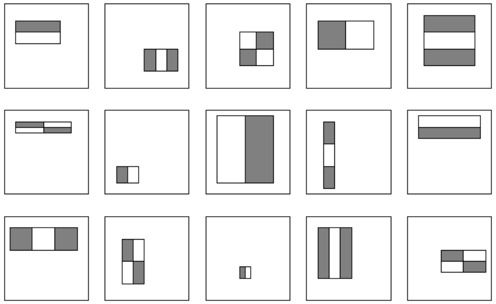
B[0-255]

Do we retain all the information in the original document?

Other features for images?

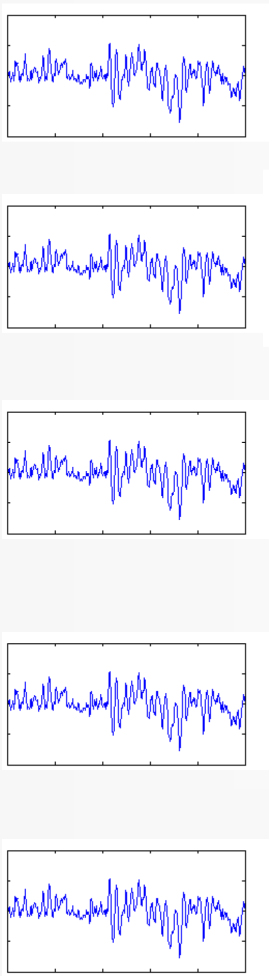
Lots of image features

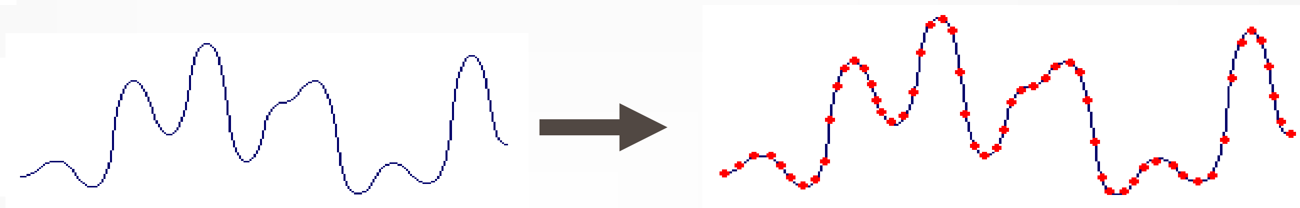
- Use “patches” rather than pixels (sort of like “bigrams” for text)
- Different color representations (i.e. L*A*B*)
- Texture features, i.e. responses to filters



- Shape features
- …

Audio: raw data





How is audio data stored?

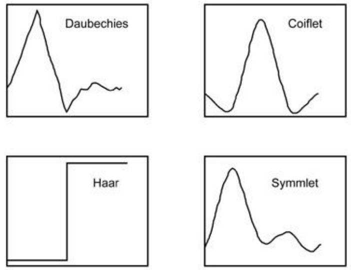
Many different file formats, but some notion of the frequency over time

Audio features?

Audio features

- frequencies represented in the data (FFT)
- frequencies over time (STFT)/responses to wave patterns (wavelets)

- beat
- timber
- energy
- zero crossings
- …



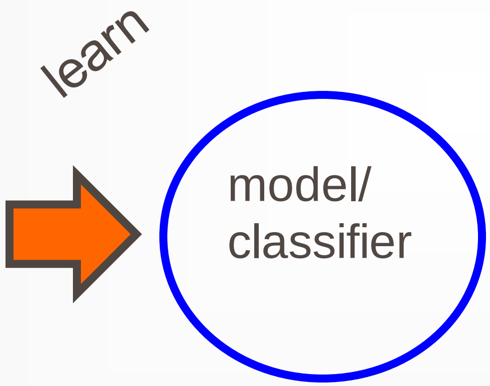
Obtaining features

- Very often requires some domain knowledge
- As ML algorithm developers, we often have to trust the “experts” to identify and extract reasonable features
- That said, it can be helpful to understand where the features are coming from

Current learning model

training data
(labeled examples)

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES



Pre-process training data

training data
(labeled examples)

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

pre-process data

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

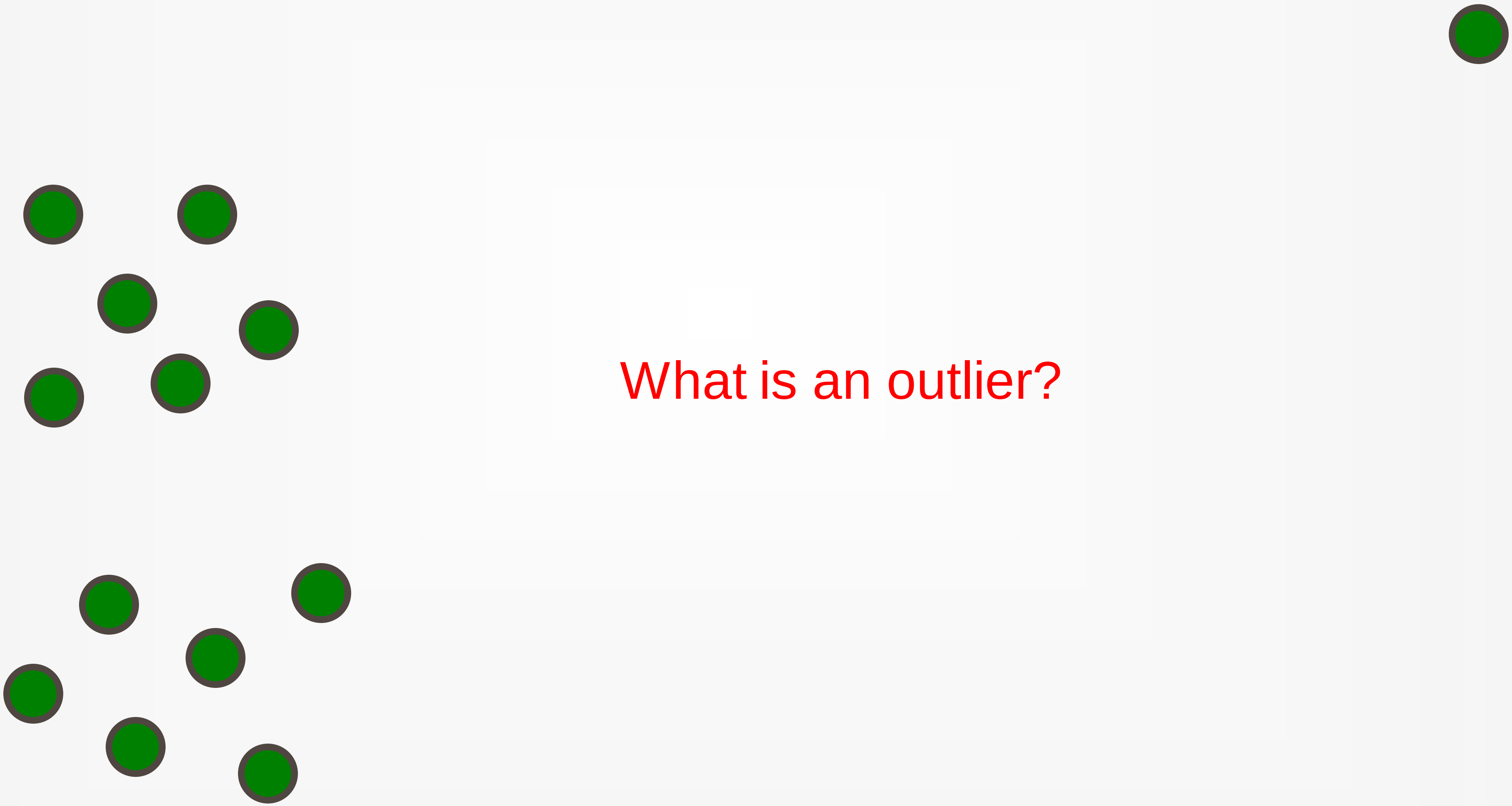
"better" training data

learn

model/
classifier

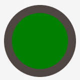
What types of preprocessing might we want to do?

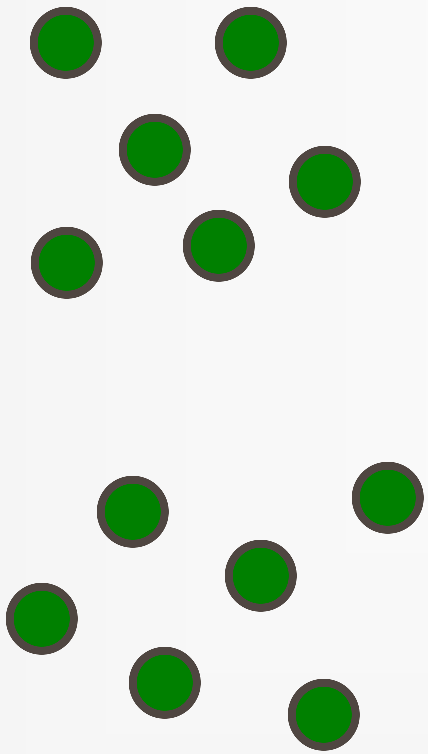
Outlier detection



What is an outlier?

Outlier detection

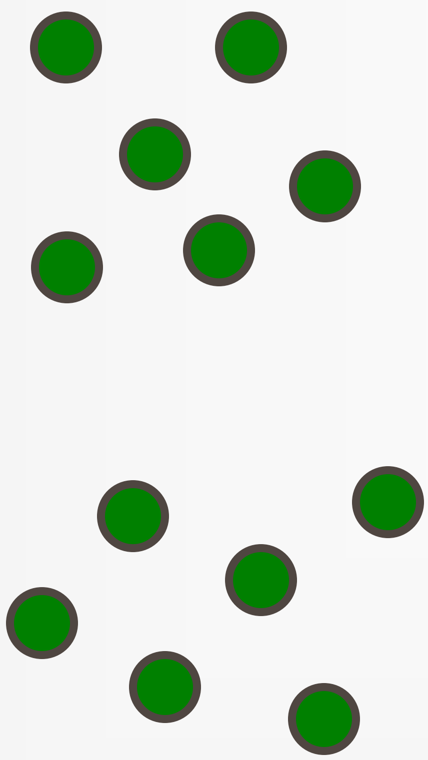


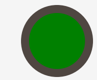


An example that is inconsistent with the other examples

What types of inconsistencies?

Outlier detection

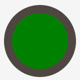


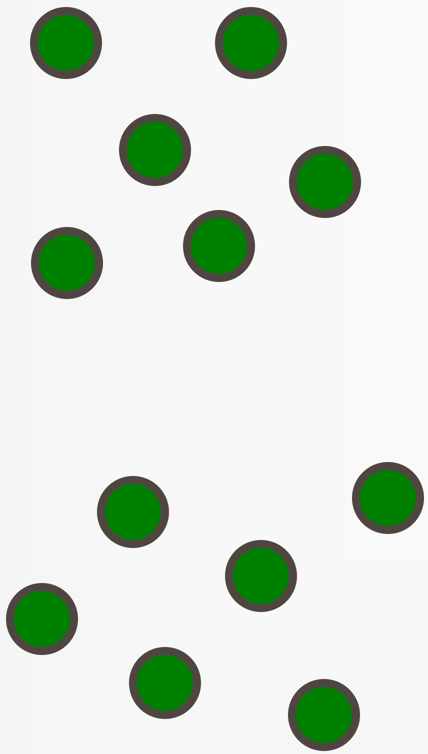


An example that is inconsistent with the other examples

- extreme feature values in one or more dimensions
- examples with the same feature values but different labels

Outlier detection





An example that is inconsistent with the other examples

- extreme feature values in one or more dimensions
- examples with the same feature values but different labels

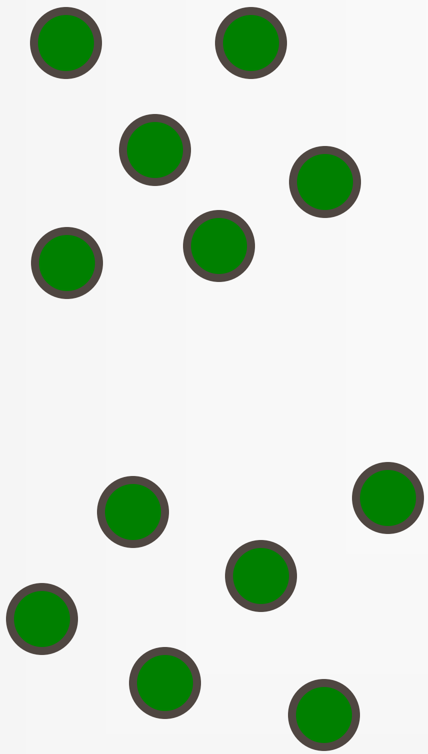
Fix?

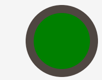
Removing conflicting examples

- Identify examples that have the same features, but differing values
  - For some learning algorithms, this can cause issues (for example, not converging)
  - In general, unsatisfying from a learning perspective

- Can be a bit expensive computationally (examining all pairs), though faster approaches are available

Outlier detection





An example that is inconsistent with the other examples

- extreme feature values in one or more dimensions
- examples with the same feature values but different labels

How do we identify these?

Removing extreme outliers

- Throw out examples that have extreme values in one dimension
- Throw out examples that are very far away from any other example
- Train a probabilistic model on the data and throw out “very unlikely” examples
- This is an entire field of study by itself! Often called outlier or anomaly detection.

Quick statistics recap

- What are the mean, standard deviation, and variance of data?

Quick statistics recap

mean: average value, often written as μ

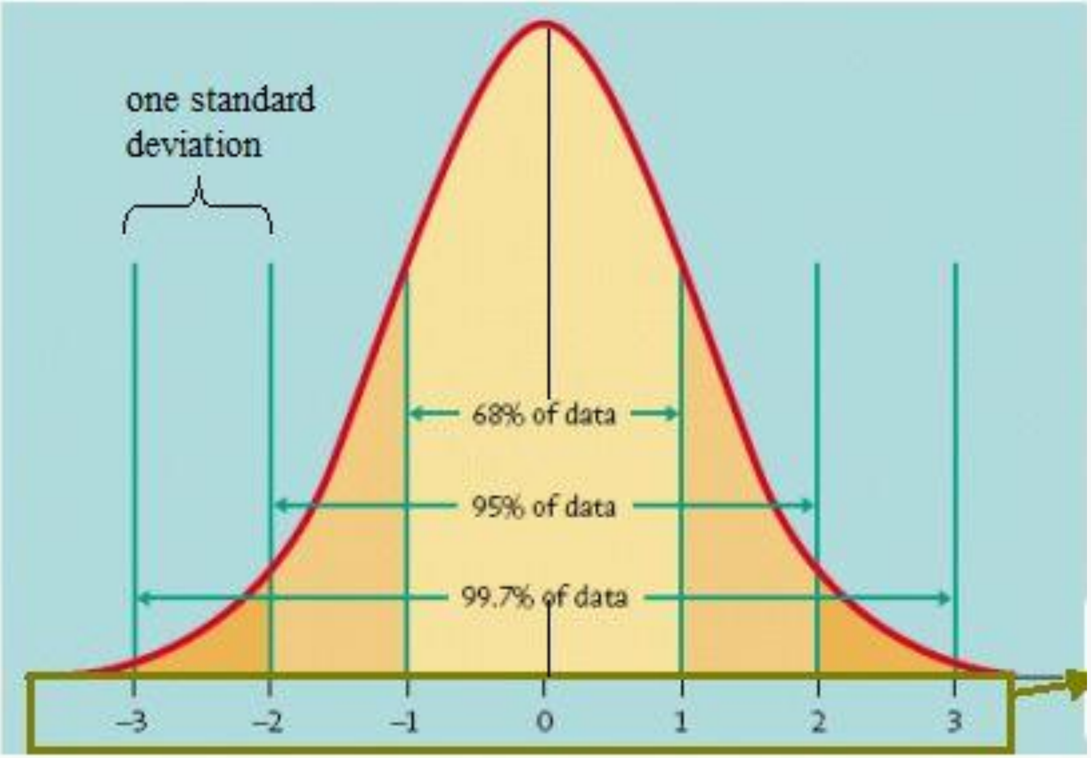
variance: a measure of how much variation there is in the data. Calculated as:

$$\sigma^2 = \frac{\sum_{i=1}^{n} (x_i - \mu)^2}{n}$$

standard deviation: square root of the variance (written as σ)

How can these help us with outliers?

Outlier detection

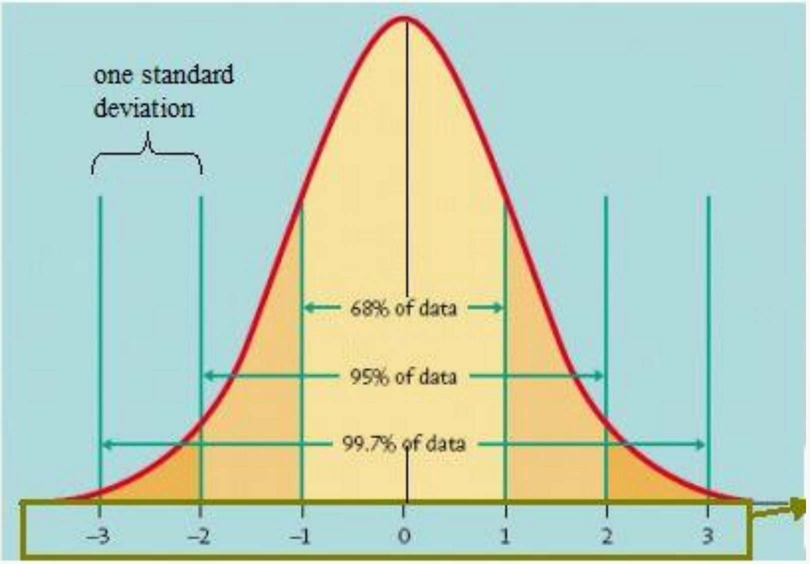


If we know the data is distributed normally (i.e. via a normal/Gaussian distribution)

Outliers in a single dimension

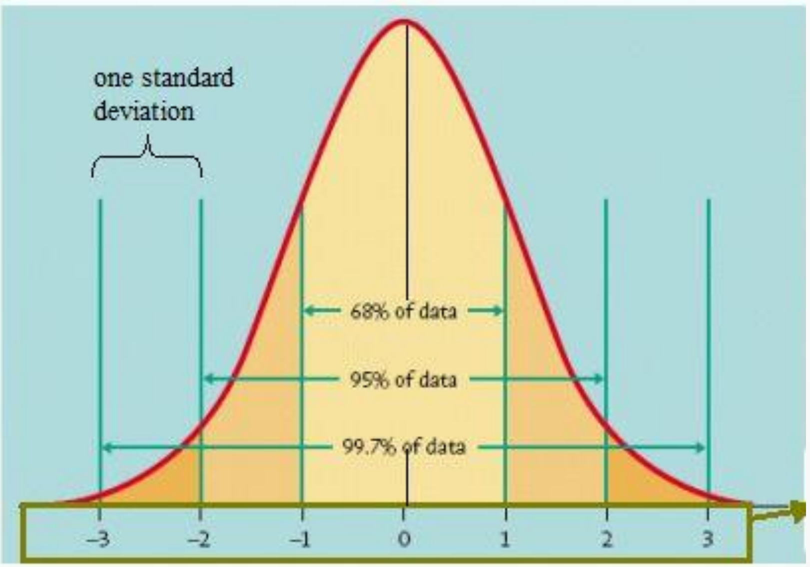
- Examples in a single dimension that have values greater than $|k\sigma|$ can be discarded (for k >>3)

- Even if the data isn't actually distributed normally, this is still often reasonable



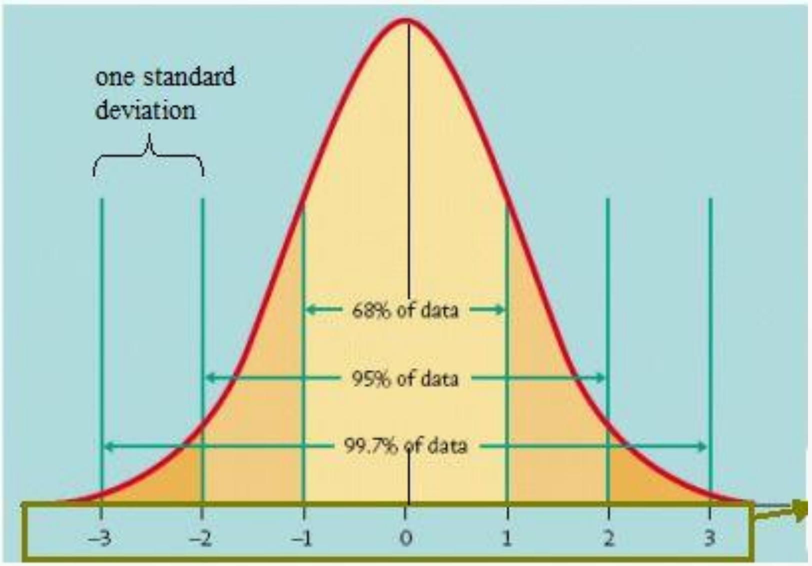
Outliers in general

- Calculate the centroid/center of the data
- Calculate the average distance from center for all data
- Calculate standard deviation and discard points too far away
  - Again, many, many other techniques for doing this



Outliers for machine learning

- Some good practices:
  - Throw out conflicting examples
  - Throw out any examples with obviously extreme feature values (i.e. many, many standard deviations away)
  - Check for erroneous feature values (e.g. negative values for a feature that can only be positive)
  - Let the learning algorithm/other pre-processing handle the rest



Feature pruning

- Good features provide us information that helps us distinguish between labels
- However, not all features are good
- What makes a bad feature and why would we have them in our data?

Bad features

Each of you are going to generate a feature for our data set: pick 5 random binary numbers

$f_1 \; f_2 \; \ldots$

label

I’ve already labeled these examples and I have two features

Bad features

Each of you are going to generate some a feature for our data set: pick 5 random binary numbers

f_1 f_2 …

label

1

0

1

1

0

Is there any problem with using your feature in addition to my two real features?

Feature pruning/selection

- Good features provide us information that helps us distinguish between labels.
  - However, not ALL features are good

- **Feature pruning** is the process of removing “bad” features

- **Feature selection** is the process of selecting “good” features

- What makes a bad feature and why would we have them in our data?

Bad features

label

1
0
1
1
0

If we have a “random” feature, i.e. a feature with random binary values, what is the probability that our feature perfectly predicts the label?

Bad features

label	f_i	probability
1	1	0.5
0	0	0.5
1	1	0.5
1	1	0.5
0	0	0.5

Is that the only way to get perfect prediction?

$0.5^5 = 0.03125 = 1/32$

Bad features

label	f_i	probability
1	0	0.5
0	1	0.5
1	0	0.5
1	0	0.5
0	1	0.5

$0.5^5 = 0.03125 = 1/32$

Total = 1/32+1/32 = 1/16

Why is this a problem?

Although these features perfectly correlate/predict the training data, they will generally NOT have any predictive power on the test set!

Bad features

label	f_i	probability
1	0	0.5
0	1	0.5
1	0	0.5
1	0	0.5
0	1	0.5

$0.5^5 = 0.03125 = 1/32$

Total = 1/32+1/32 = 1/16

Is perfect correlation the only thing we need to worry about for random features?

Bad features

label	f_i
1	1
0	0
1	1
1	0
0	0

Any correlation (particularly any strong correlation) can affect performance!

Noisy features

- Adding features can give us more information, but not always
- Determining if a feature is useful can be challenging

Terrain	Unicycle-type	Weather	Jacket	ML grade	Go-For-Ride?
Trail	Mountain	Rainy	Heavy	D	YES
Trail	Mountain	Sunny	Light	C-	YES
Road	Mountain	Snowy	Light	B	YES
Road	Mountain	Sunny	Heavy	A	YES
Trail	Normal	Snowy	Light	D+	NO
Trail	Normal	Rainy	Heavy	B-	NO
Road	Normal	Snowy	Heavy	C+	YES
Road	Normal	Sunny	Light	A-	NO
Trail	Normal	Sunny	Heavy	B+	NO
Trail	Normal	Snowy	Light	F	NO
Trail	Normal	Rainy	Light	C	YES

Noisy features

These can be particularly problematic in problem areas where we automatically generate features

Features

Trump said banana repeatedly last week on tv, “banana, banana, banana”

(1, 1, 1, 0, 0, 1, 0, 0, ...)

Trump said
said banana
across the
california schools
tv banana
wrong way
capital city

Noisy features

Ideas for removing noisy/random features?

Terrain	Unicycle-type	Weather	Jacket	ML grade	Go-For-Ride?
Trail	Mountain	Rainy	Heavy	D	YES
Trail	Mountain	Sunny	Light	C-	YES
Road	Mountain	Snowy	Light	B	YES
Road	Mountain	Sunny	Heavy	A	YES
Trail	Normal	Snowy	Light	D+	NO
Trail	Normal	Rainy	Heavy	B-	NO
Road	Normal	Snowy	Heavy	C+	YES
Road	Normal	Sunny	Light	A-	NO
Trail	Normal	Sunny	Heavy	B+	NO
Trail	Normal	Snowy	Light	F	NO
Trail	Normal	Rainy	Light	C	YES

Removing noisy features

- The expensive way:
  - Split training data into train/dev
  - Train a model on all features
  - for each feature ***f***:
    - Train a model on all features – ***f (A subset of original Features)***
    - Compare performance of all vs. ***all-f*** on dev set

  - Remove all features where decrease in performance between all and ***all-f*** is less than some constant

Feature ablation study

Issues/concerns?

Removing noisy features

- Binary features:
  - remove "rare" features, i.e. features that only occur (or don't occur) a very small number of times
- Real-valued features:
  - remove features that have low variance
- In both cases, can either use thresholds, throw away lowest x%, use development data, etc.

Why?

Some rules of thumb for the number of features

- Be very careful in domains where:
  - the number of features > number of examples
  - the number of features ≈ number of examples
  - the features are generated automatically
  - there is a chance of “random” features

- In most of these cases, features should be removed based on some domain knowledge (i.e. problem-specific knowledge)

So far…

- Throw out outlier examples
- Remove noisy features
- Pick “good” features

Feature selection

- Let’s look at the problem from the other direction, that is, selecting good features.

- What are good features?
  - How can we pick/select them?

Good features

A good feature correlates well with the label

label				
1	1	0	1	
0	0	1	1	
1	1	0	1	…
1	1	0	1	
0	0	1	0	

How can we identify this?

- training error (like for DT)
- correlation model
- statistical test
- probabilistic test
- …

Training error feature selection

- for each feature f:
  - calculate the training error if only feature f were used to pick the label

- rank each feature by this value
- pick top k, top x%, etc.
  - can use a development set to help pick k or x

So far…

- Throw out outlier examples
- Remove noisy features
- Pick “good” features

Feature normalization

Length	Weight	Color	Label
4	4	0	Apple
5	5	1	Apple
7	6	1	Banana
4	3	0	Apple
6	7	1	Banana
5	8	1	Banana
5	6	1	Apple

Length	Weight	Color	Label
40	4	0	Apple
50	5	1	Apple
70	6	1	Banana
40	3	0	Apple
60	7	1	Banana
50	8	1	Banana
50	6	1	Apple

Would our three classifiers (DT, k-NN and perceptron) learn the same models on these two data sets?

Feature normalization

Length	Weight	Color	Label
4	4	0	Apple
5	5	1	Apple
7	6	1	Banana
4	3	0	Apple
6	7	1	Banana
5	8	1	Banana
5	6	1	Apple

Length	Weight	Color	Label
40	4	0	Apple
50	5	1	Apple
70	6	1	Banana
40	3	0	Apple
60	7	1	Banana
50	8	1	Banana
50	6	1	Apple

Decision trees don’t care about scale, so they’d learn the same tree

Feature normalization

Length	Weight	Color	Label
4	4	0	Apple
5	5	1	Apple
7	6	1	Banana
4	3	0	Apple
6	7	1	Banana
5	8	1	Banana
5	6	1	Apple

Length	Weight	Color	Label
40	4	0	Apple
50	5	1	Apple
70	6	1	Banana
40	3	0	Apple
60	7	1	Banana
50	8	1	Banana
50	6	1	Apple

k-NN: NO! The distances are biased based on feature magnitude.

$$D(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2}$$

Feature normalization

Length	Weight	Label
4	4	Apple
7	5	Apple
5	8	Banana

Which of the two examples are closest to the first?

Length	Weight	Label
40	4	Apple
70	5	Apple
50	8	Banana

$$D(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2}$$

Feature normalization

Length	Weight	Label
4	4	Apple
7	5	Apple
5	8	Banana

$$D = \sqrt{(7-4)^2 + (5-4)^2} = \sqrt{10}$$

$$D = \sqrt{(5-4)^2 + (8-4)^2} = \sqrt{17}$$

Length	Weight	Label
40	4	Apple
70	5	Apple
50	8	Banana

$$D = \sqrt{(70-40)^2 + (5-4)^2} = \sqrt{901}$$

$$D = \sqrt{(70-50)^2 + (8-4)^2} = \sqrt{416}$$

$$D(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \ldots + (a_n - b_n)^2}$$

Feature normalization

Length	Weight	Color	Label
4	4	0	Apple
5	5	1	Apple
7	6	1	Banana
4	3	0	Apple
6	7	1	Banana
5	8	1	Banana
5	6	1	Apple

Length	Weight	Color	Label
40	4	0	Apple
50	5	1	Apple
70	6	1	Banana
40	3	0	Apple
60	7	1	Banana
50	8	1	Banana
50	6	1	Apple

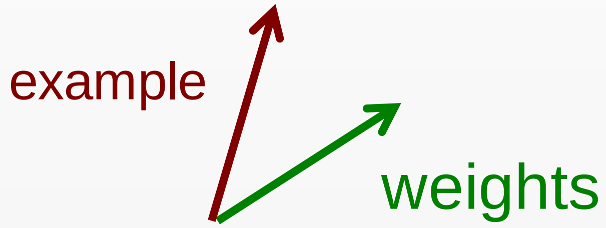
perceptron: NO!
The classification and weight update are based on the magnitude of the feature value

Geometric view of perceptron update

for each w_i:

$$w_i = w_i + f_i * \text{label}$$

Geometrically, the perceptron update rule is equivalent to “adding” the weight vector and the feature vector

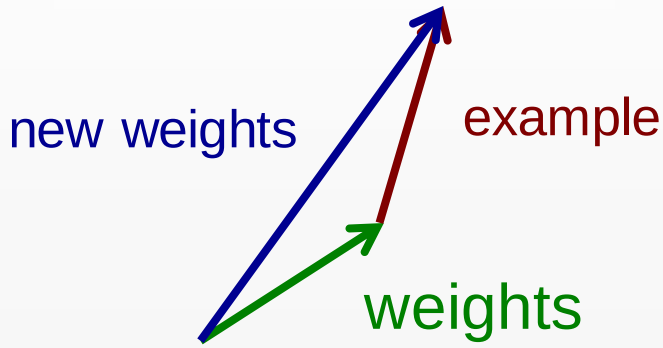


Geometric view of perceptron update

for each w_i:

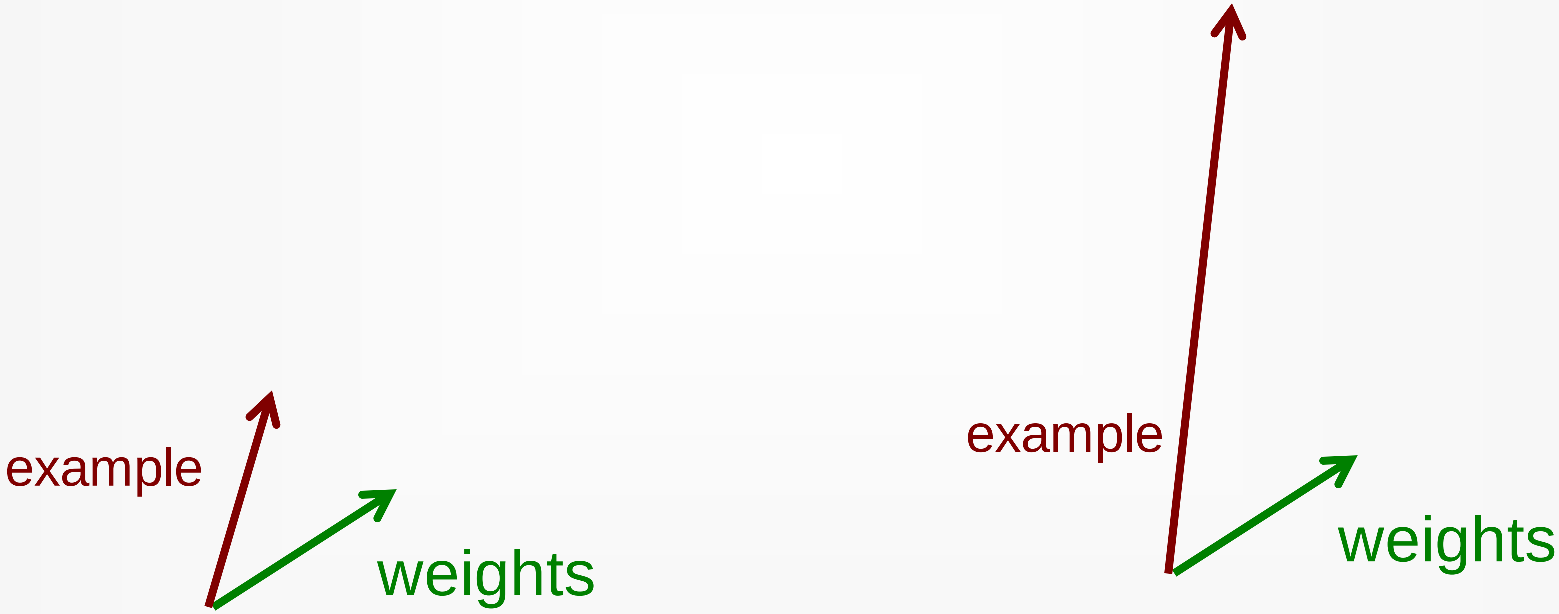
$$w_i = w_i + f_i * \text{label}$$

Geometrically, the perceptron update rule is equivalent to "adding" the weight vector and the feature vector



Geometric view of perceptron update

If the features dimensions differ in scale, it can bias the update



same f1 value, but larger f2

Geometric view of perceptron update

If the features dimensions differ in scale, it can bias the update

- different separating hyperplanes
- the larger dimension becomes much more important

Feature normalization

Length	Weight	Color	Label
4	4	0	Apple
5	5	1	Apple
7	6	1	Banana
4	3	0	Apple
6	7	1	Banana
5	8	1	Banana
5	6	1	Apple

Length	Weight	Color	Label
40	4	0	Apple
50	5	1	Apple
70	6	1	Banana
40	3	0	Apple
60	7	1	Banana
50	8	1	Banana
50	6	1	Apple

How do we fix this?

Feature normalization

Length	Weight	Color	Label
40	4	0	Apple
50	5	1	Apple
70	6	1	Banana
40	3	0	Apple
60	7	1	Banana
50	8	1	Banana
50	6	1	Apple

Modify all values for a given feature

Normalize each feature

- For each feature (over all examples):
- Center: adjust the values so that the mean of that feature is 0.
- How do we do this?

Normalize each feature

- For each feature (over all examples):

- Center: adjust the values so that the mean of that feature is 0: subtract the mean from all values

- Rescale/adjust feature values to avoid magnitude bias. Ideas?

Normalize each feature

- For each feature (over all examples):

- Center: adjust the values so that the mean of that feature is 0: subtract the mean from all values

- Rescale/adjust feature values to avoid magnitude bias:
  - Variance scaling: divide each value by the std dev
  - Absolute scaling: divide each value by the largest value

Pros/cons of either scaling technique?

So far…

- Throw out outlier examples
- Remove noisy features
- Pick “good” features
- Normalize feature values
  - center data
  - scale data (either variance or absolute)

Example normalization

Length	Weight	Color	Label
4	4	0	Apple
5	5	1	Apple
7	6	1	Banana
4	3	0	Apple
6	7	1	Banana
5	8	1	Banana
5	6	1	Apple

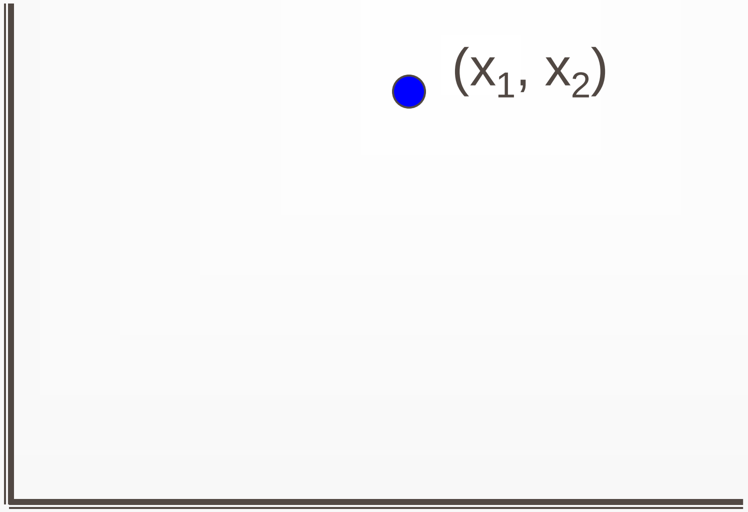
Length	Weight	Color	Label
4	4	0	Apple
5	5	1	Apple
70	60	1	Banana
4	3	0	Apple
6	7	1	Banana
5	8	1	Banana
5	6	1	Apple

Any problem with this?
Solutions?

Example length normalization

Make all examples roughly the same scale, e.g. make all have length = 1

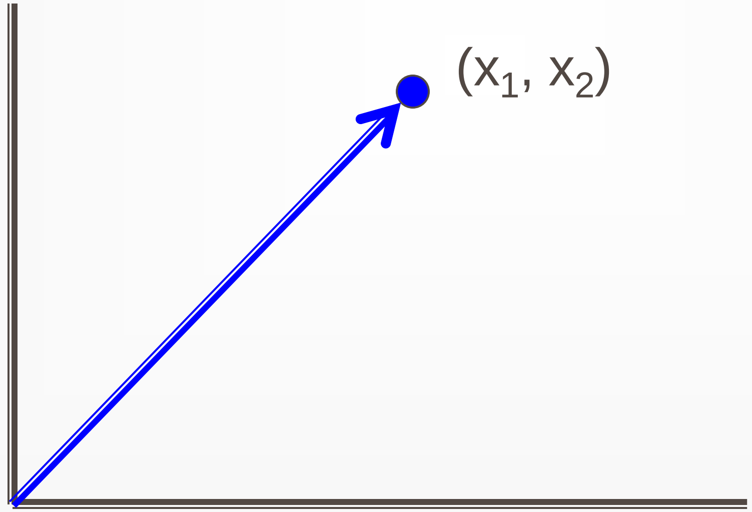
What is the length of this example/vector?



Example length normalization

Make all examples roughly the same scale, e.g. make all have length = 1

What is the length of this example/vector?

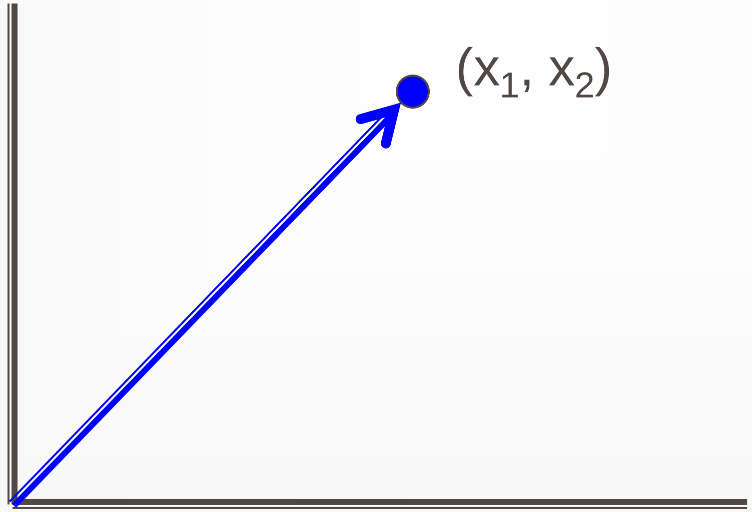


$$length(x) = \|x\| = \sqrt{x_1^2 + x_2^2}$$

Example length normalization

Make all examples roughly the same scale, e.g. make all have length = 1

What is the length of this example/vector?



$$length(x) = \|x\| = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$$

Example length normalization

Make all examples have length = 1

Divide each feature value by ||x||

- Prevents a single example from being too impactful
- Equivalent to projecting each example onto a unit sphere

$$length(x) = \|x\| = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}$$

So far…

- Throw out outlier examples
- Remove noisy features
- Pick “good” features
- Normalize feature values
  - center data
  - scale data (either variance or absolute)
- Normalize example length
- Finally, train your model!

What about testing?

training data
(labeled examples)

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

pre-process data

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

learn

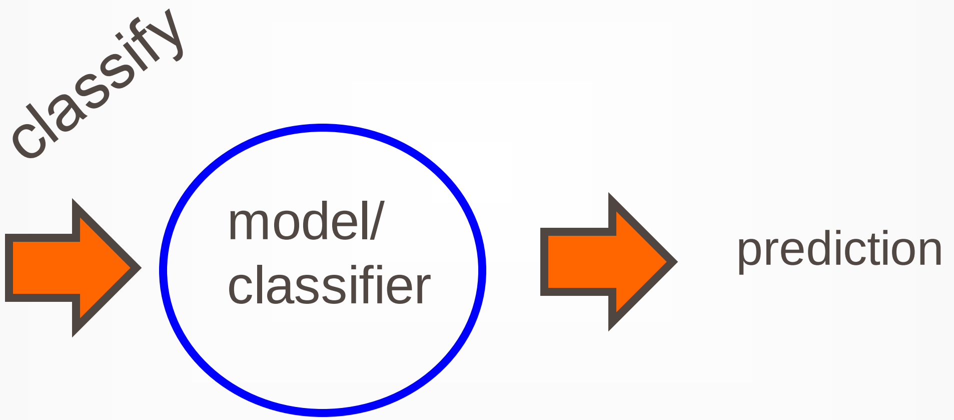
model/
classifier

“better” training data

What about testing?

test data

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES



What about testing?

test data

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

pre-process data

Terrain	Unicycle-type	Weather	Go-For-Ride?
Trail	Normal	Rainy	NO
Road	Normal	Sunny	YES
Trail	Mountain	Sunny	YES
Road	Mountain	Rainy	YES
Trail	Normal	Snowy	NO
Road	Normal	Rainy	YES
Road	Mountain	Snowy	YES
Trail	Normal	Sunny	NO
Road	Normal	Snowy	NO
Trail	Mountain	Snowy	YES

classify

model/
classifier

prediction

How do we preprocess the test data?

Test data preprocessing

- Throw out outlier examples
- Remove noisy features
- Pick “good” features
- Normalize feature values
  - center data
  - scale data (either variance or absolute)
- Normalize example length

Which of these do we need to do on test data? Any issues?

Test data preprocessing

- Throw out outlier examples
- Remove irrelevant/noisy features
- Pick "good" features
- Normalize feature values
  - center data
  - scale data (either variance or absolute)
- Normalize example length

Remove/pick same features

Do these

Do this

Whatever you do on training, you have to do the EXACT same on testing!

Normalizing test data

- For each feature (over all examples):

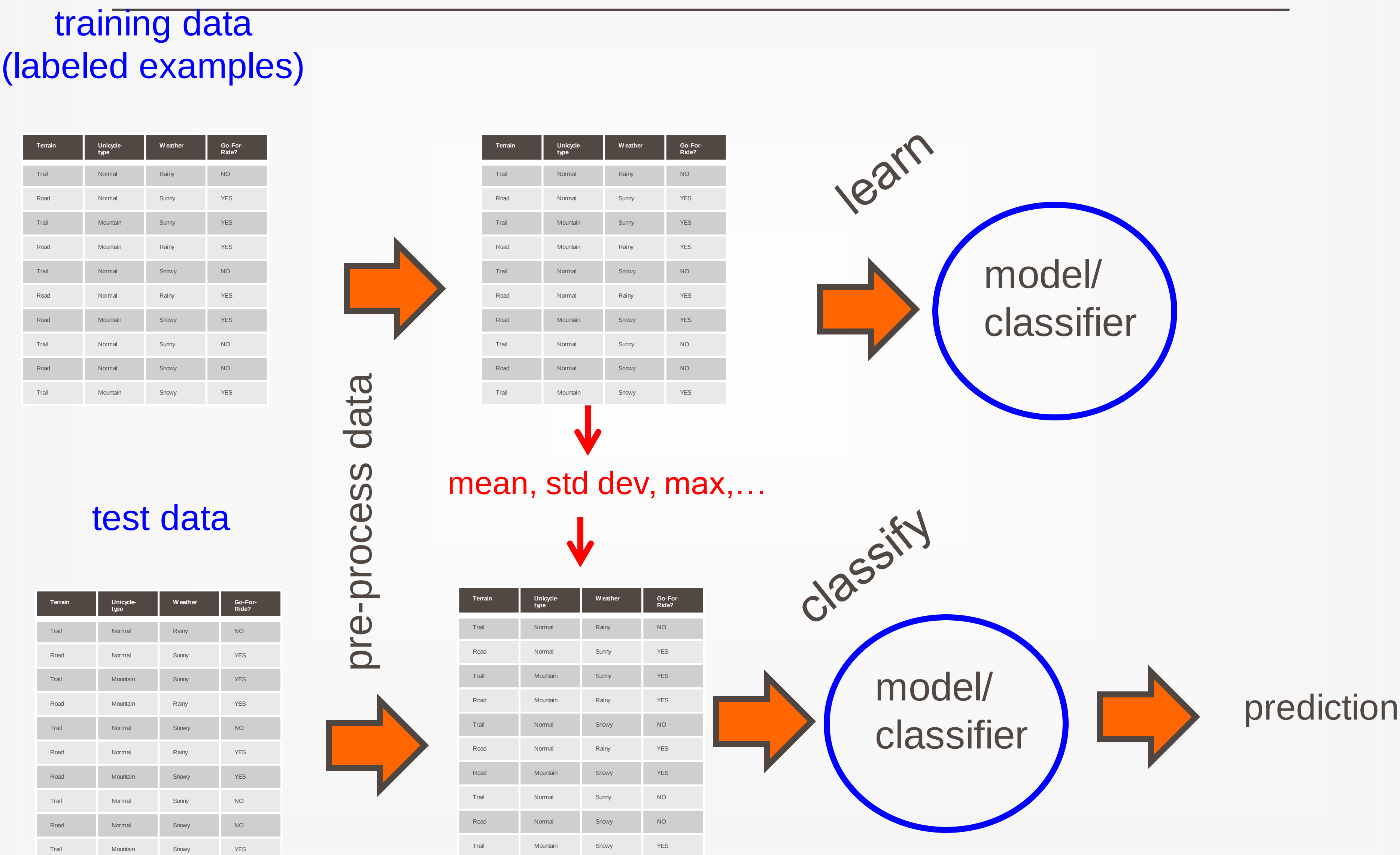
- Center: adjust the values so that the mean of that feature is 0: subtract the mean from all values

- Rescale/adjust feature values to avoid magnitude bias:
  - Variance scaling: divide each value by the std dev
  - Absolute scaling: divide each value by the largest value

What values do we use when normalizing testing data?

Save these from training normalization!

Normalizing test data



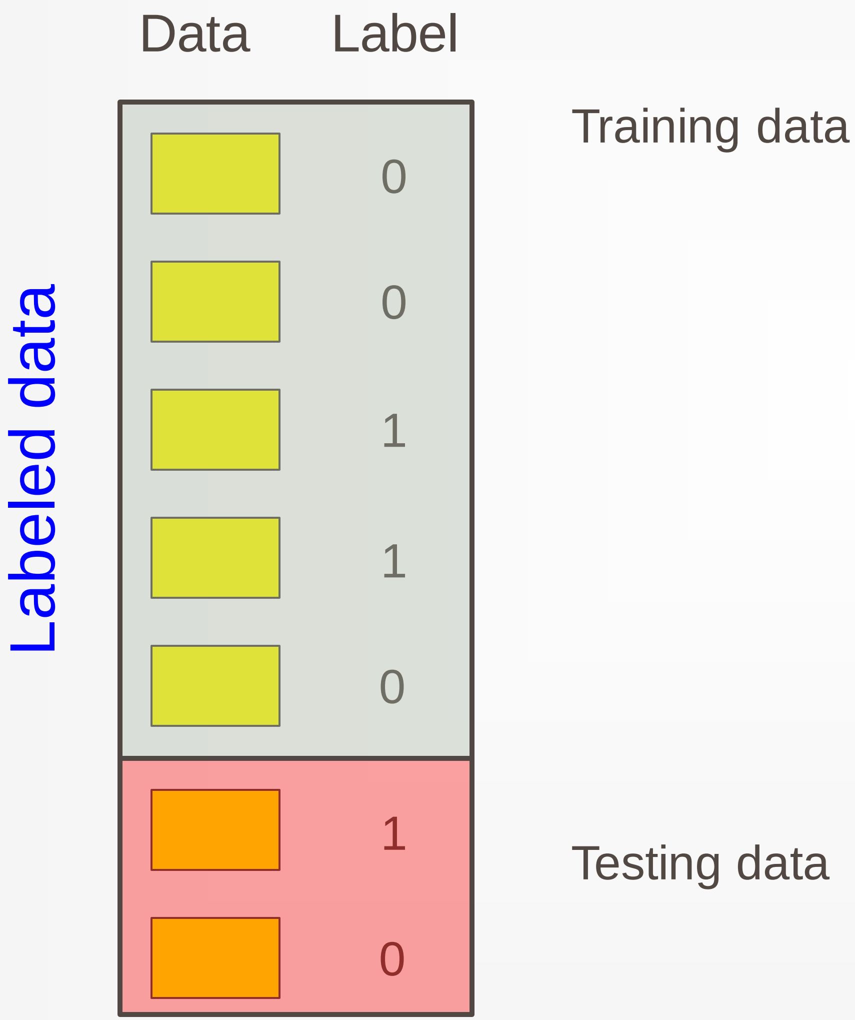
Features pre-processing summary

- Many techniques for preprocessing data
- Which will work well will depend on the data and the classifier
- Try them out and evaluate how they affect performance on dev data
- Make sure to do **exact same** pre-processing on train and test

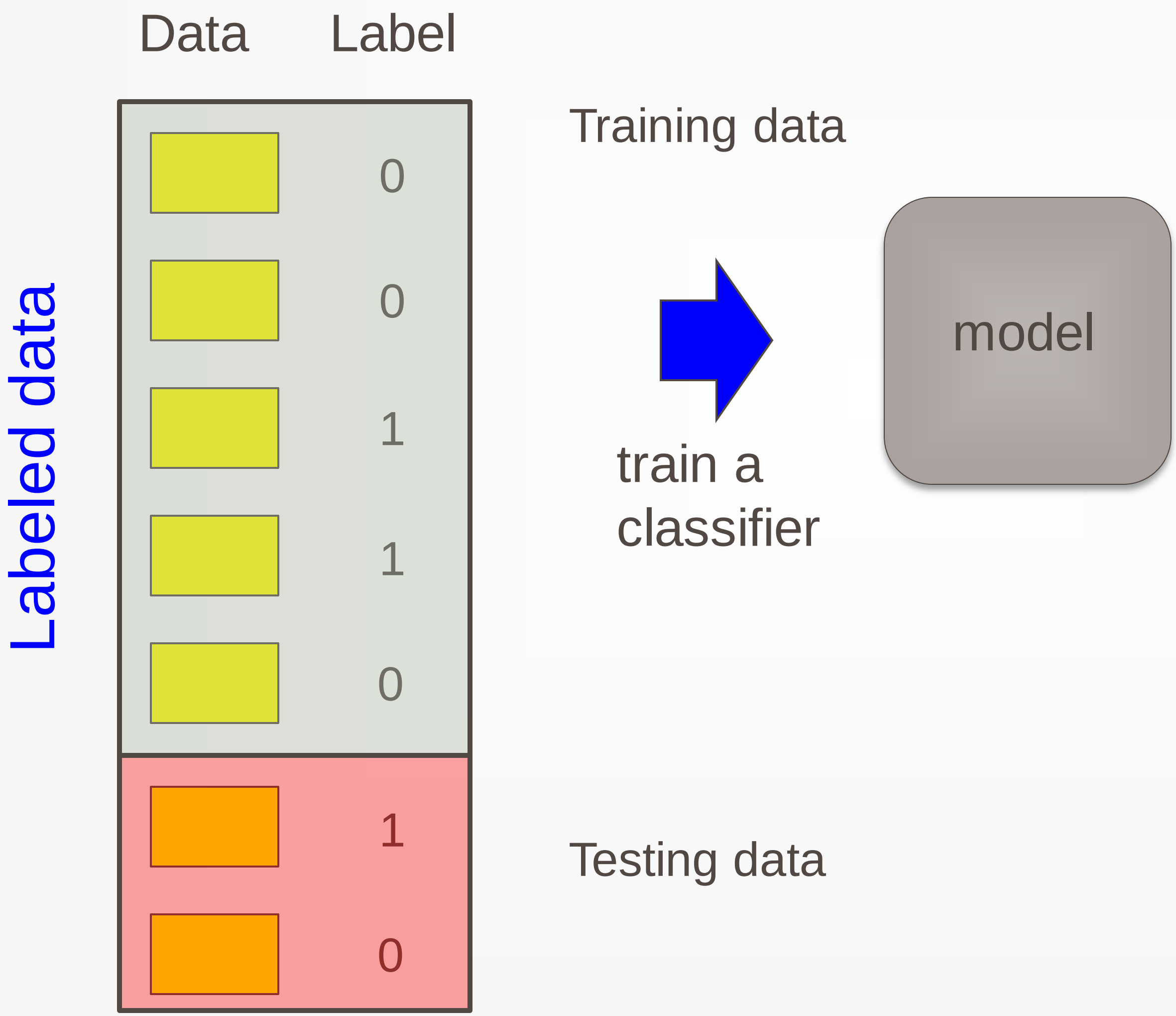
- Throw out outlier examples
- Remove noisy features
- Pick “good” features
- Normalize feature values
  - center data
  - scale data (either variance or absolute)
- Normalize example length

EVALUATION

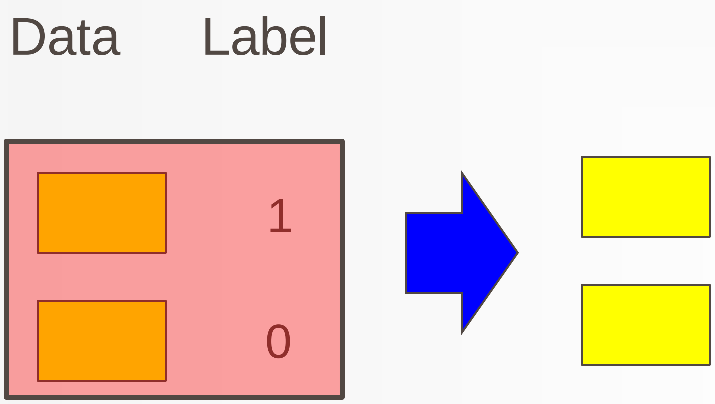
Supervised evaluation



Supervised evaluation

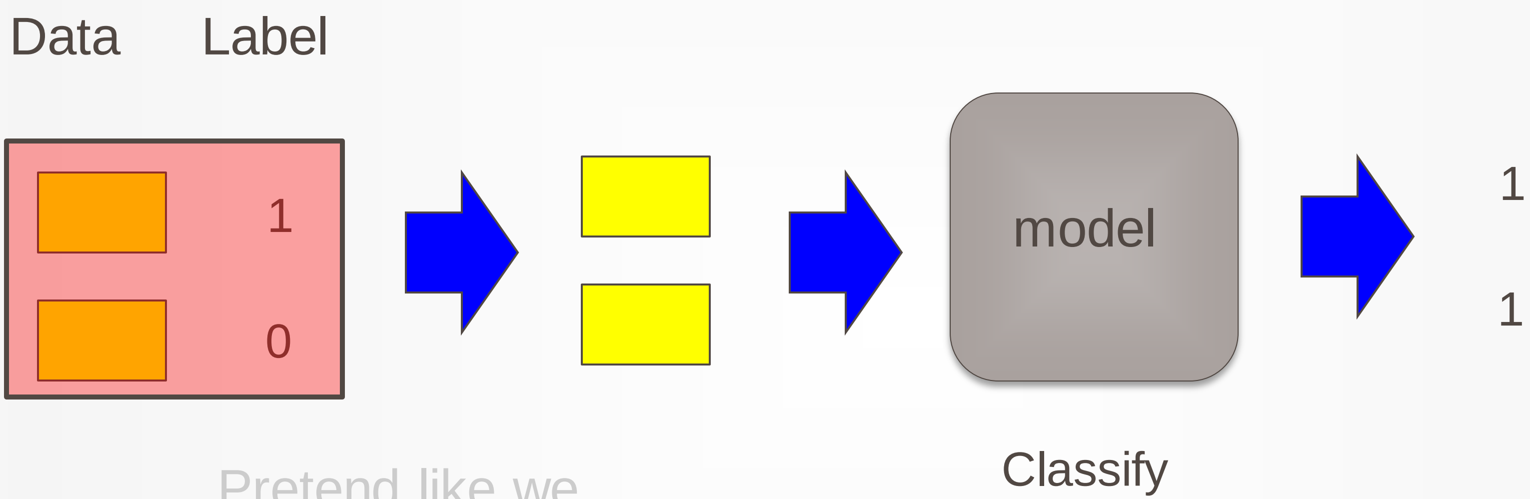


Supervised evaluation



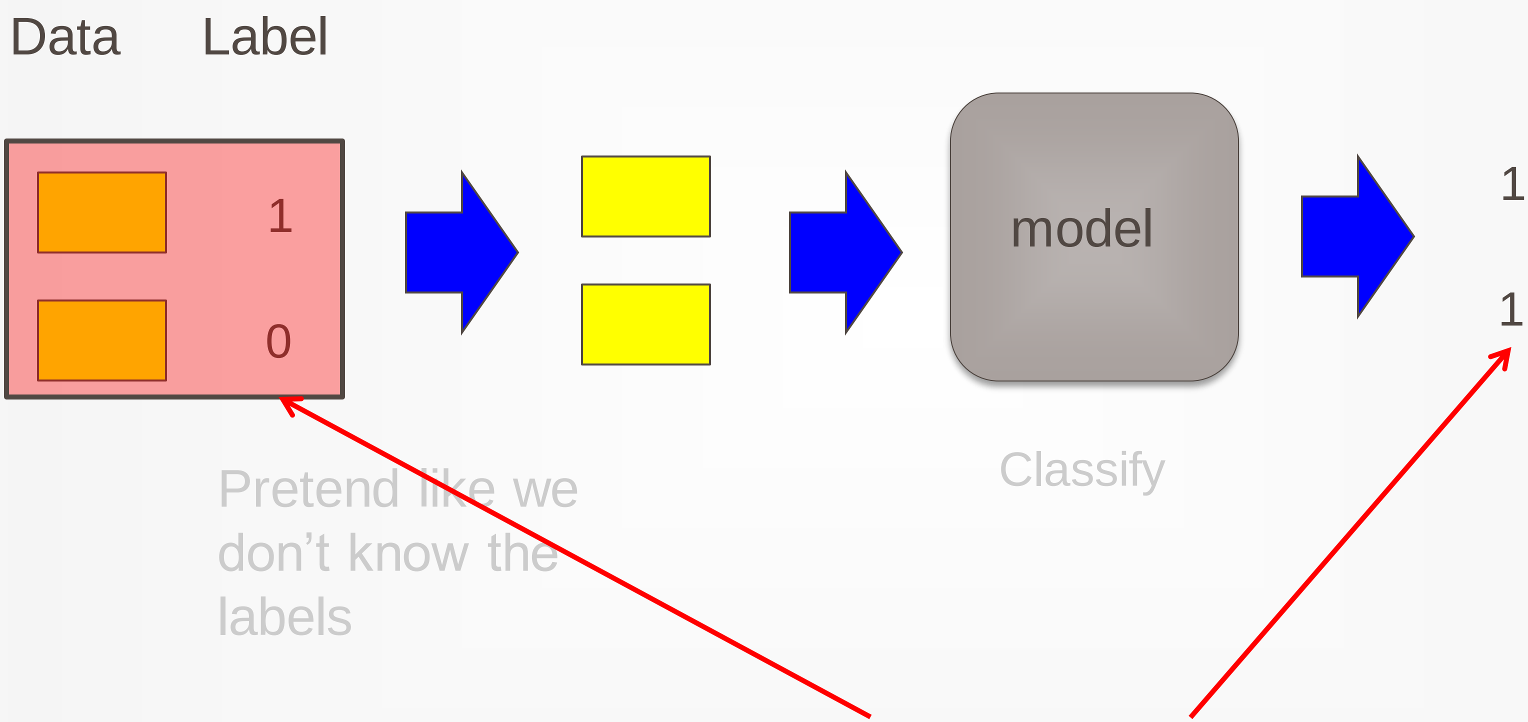
Pretend like we don’t know the labels

Supervised evaluation



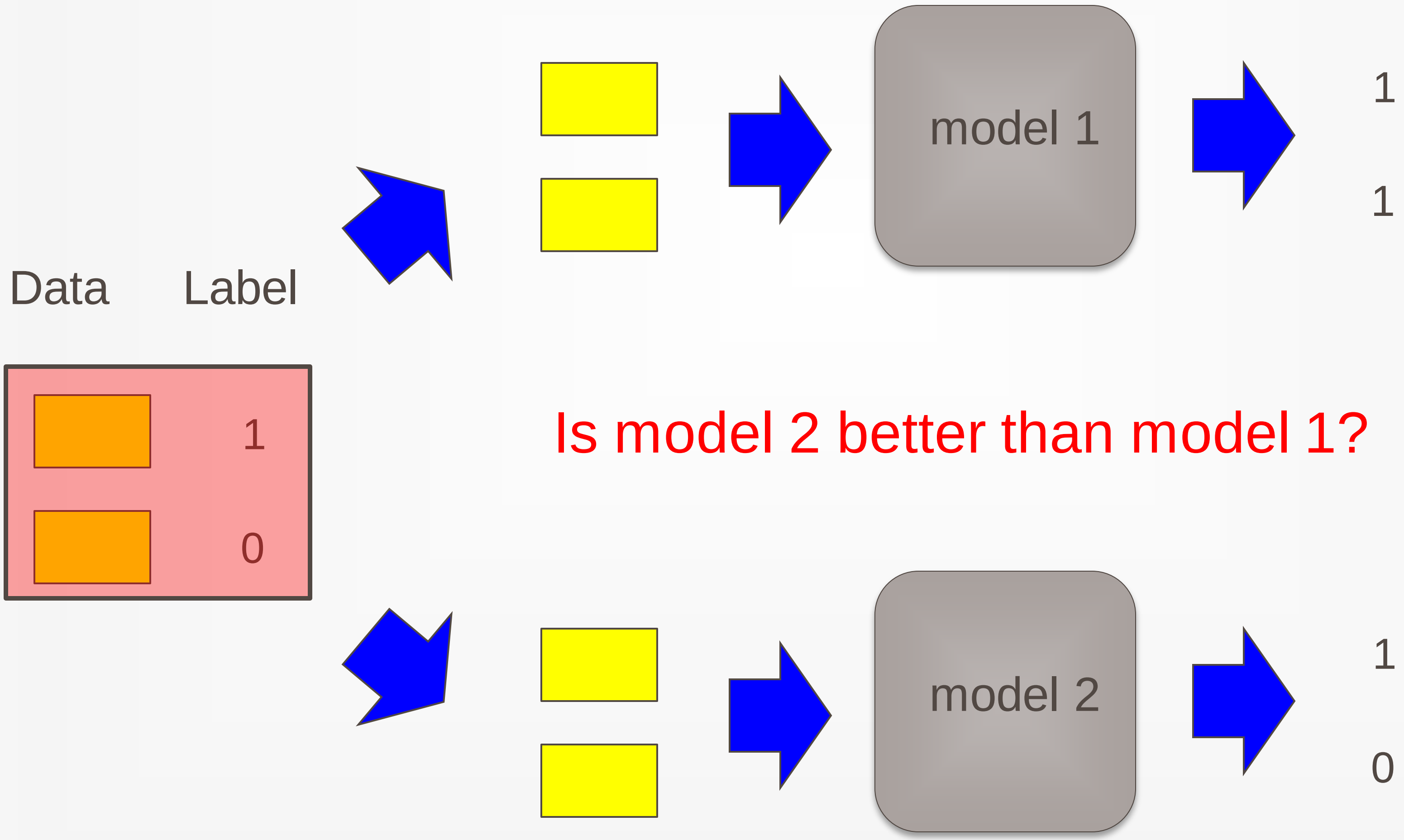
Pretend like we don't know the labels

Supervised evaluation

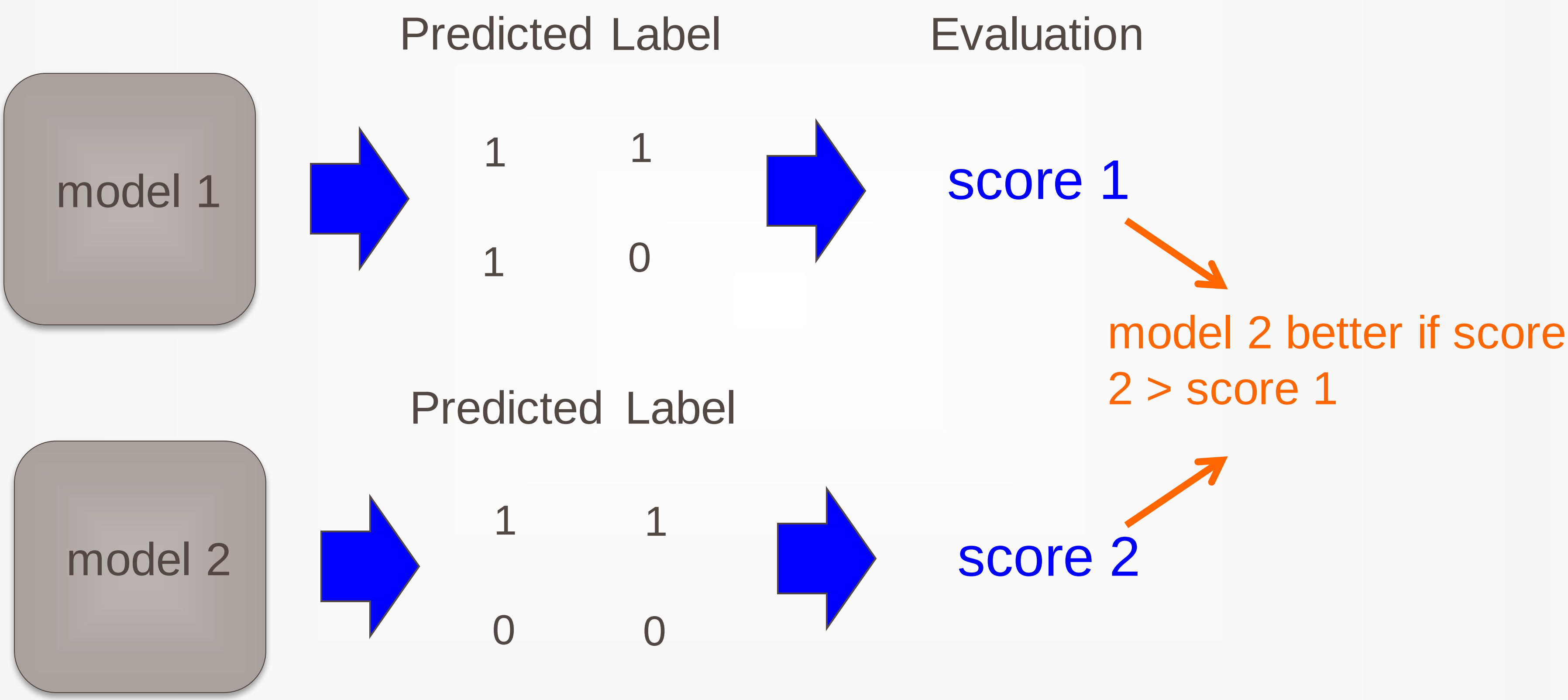


Compare predicted labels to actual labels

Comparing algorithms

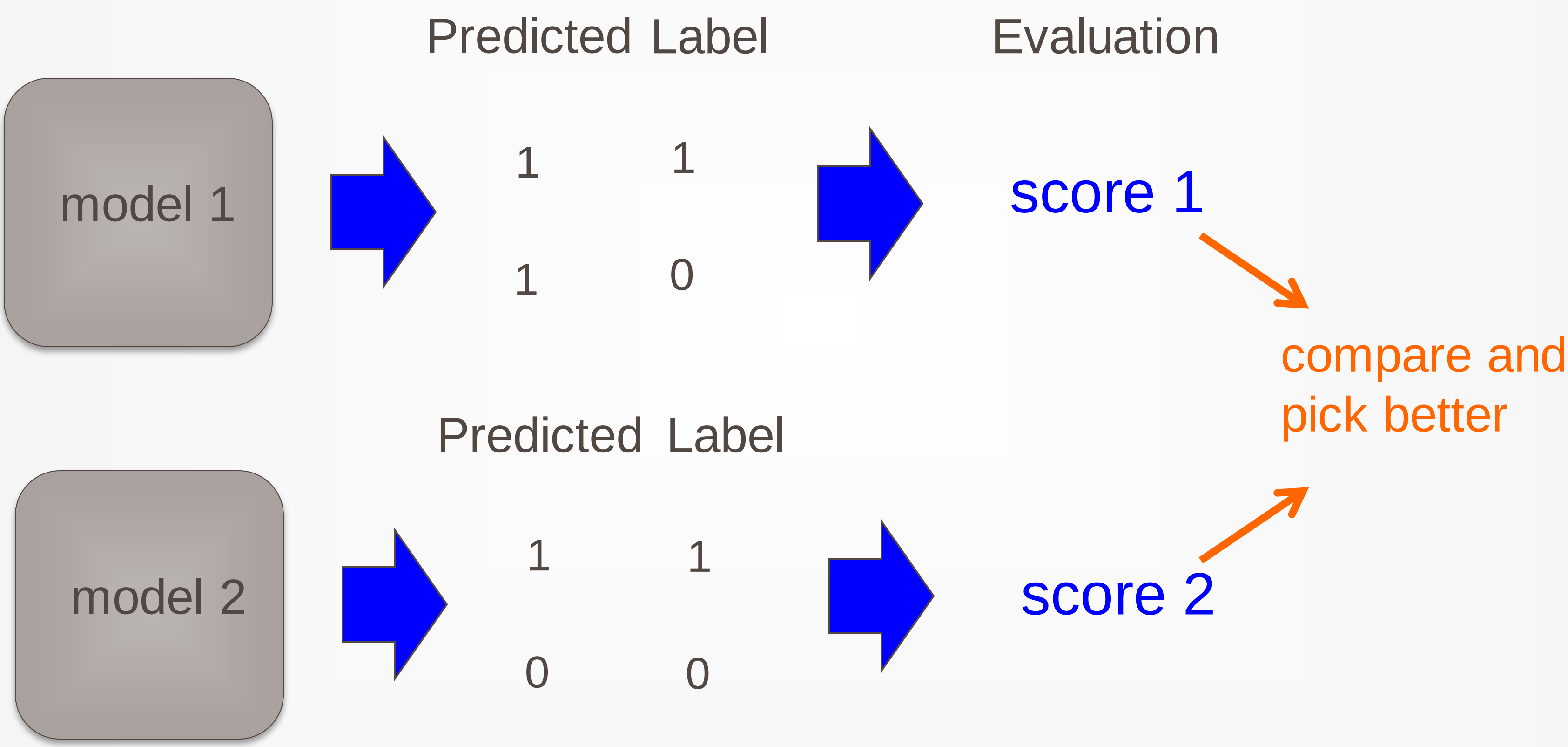


Idea 1



When would we want to do this type of comparison?

Idea 1



Any concerns?

Is model 2 better?

Model 1: 85% accuracy
Model 2: 80% accuracy

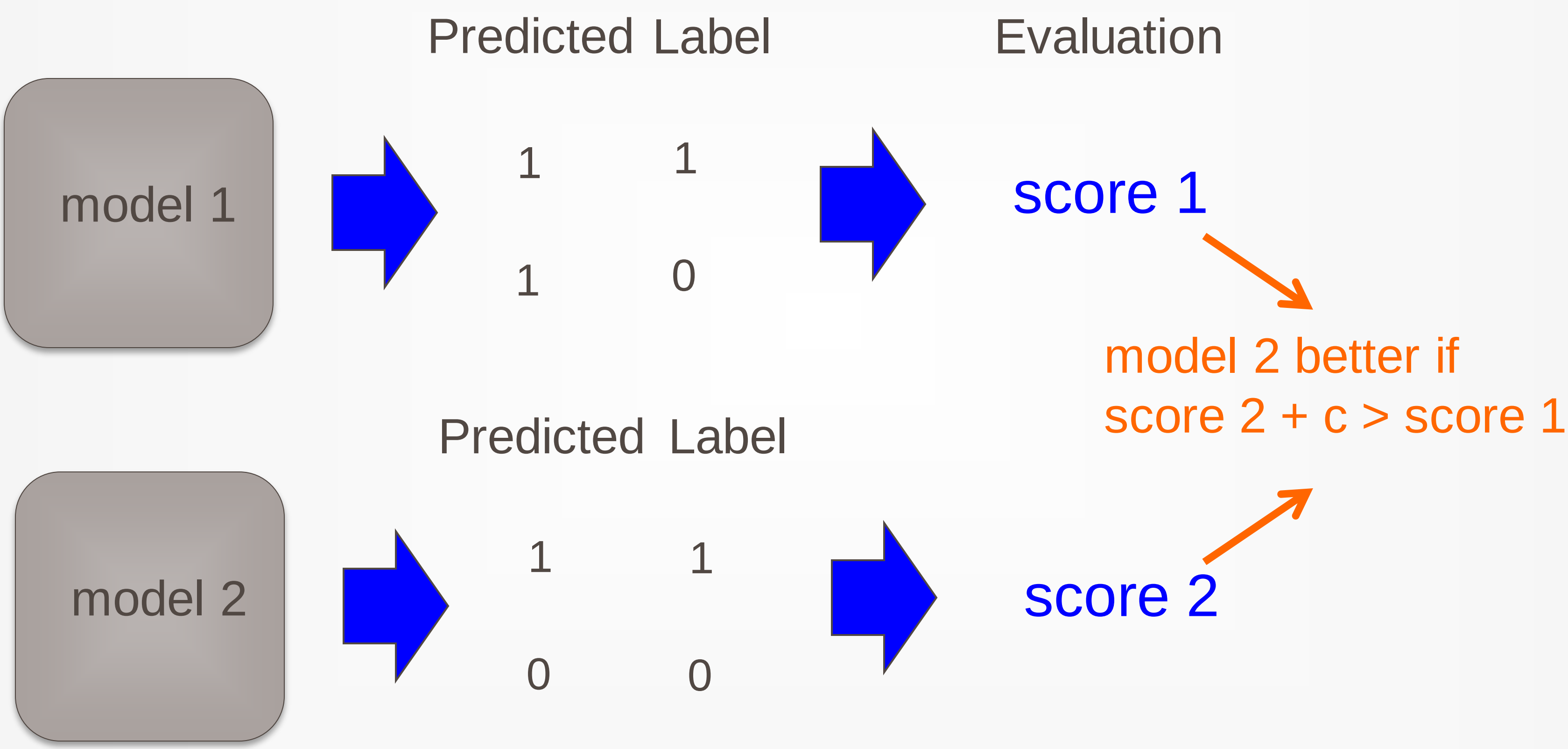
Model 1: 85.5% accuracy
Model 2: 85.0% accuracy

Model 1: 0% accuracy
Model 2: 100% accuracy

Comparing scores: significance

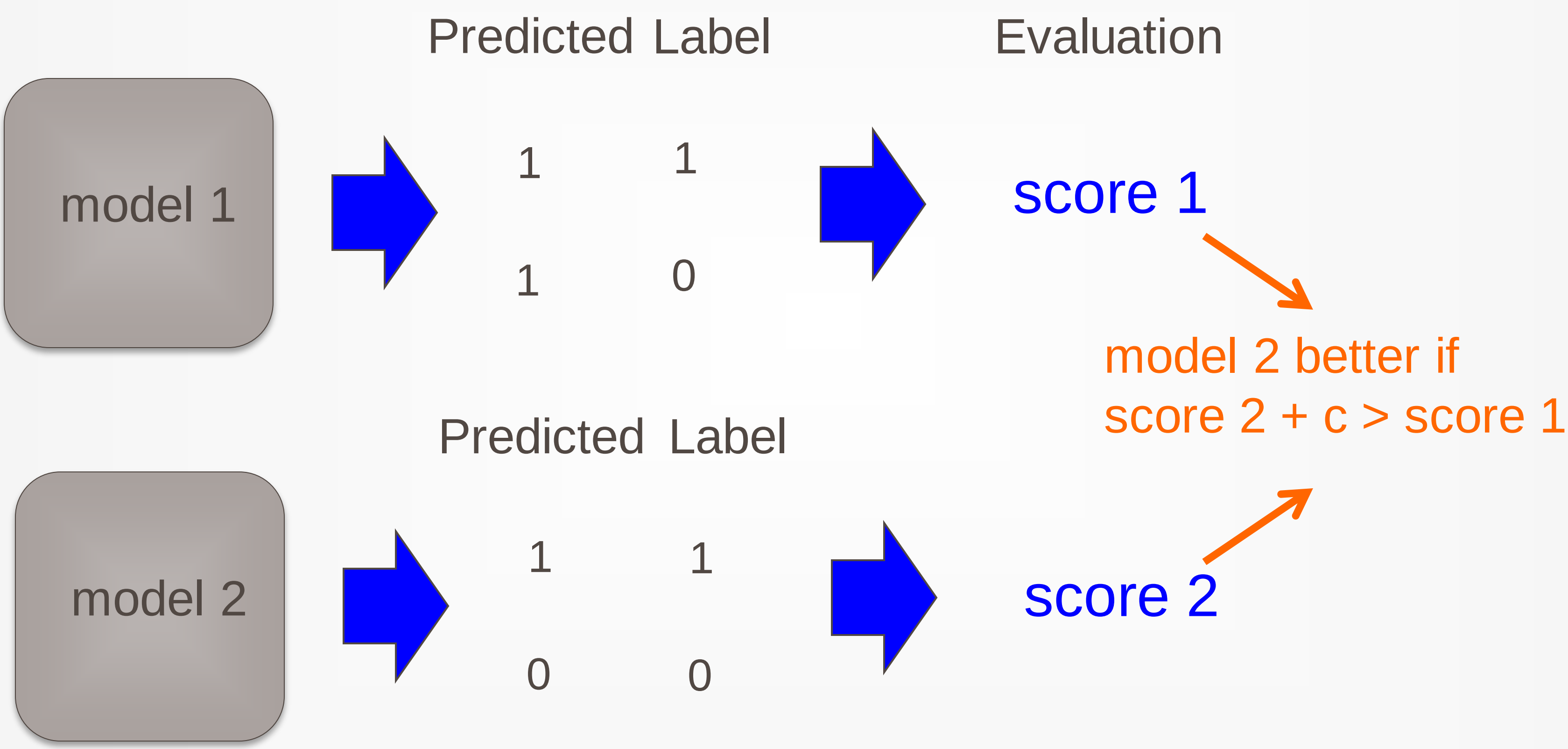
- Just comparing scores on one data set isn't enough!
- We don't just want to know which system is better on this particular data, we want to know if model 1 is better than model 2 in general
- Put another way, we want to be confident that the difference is real and not just do to random chance

Idea 2



Is this any better?

Idea 2

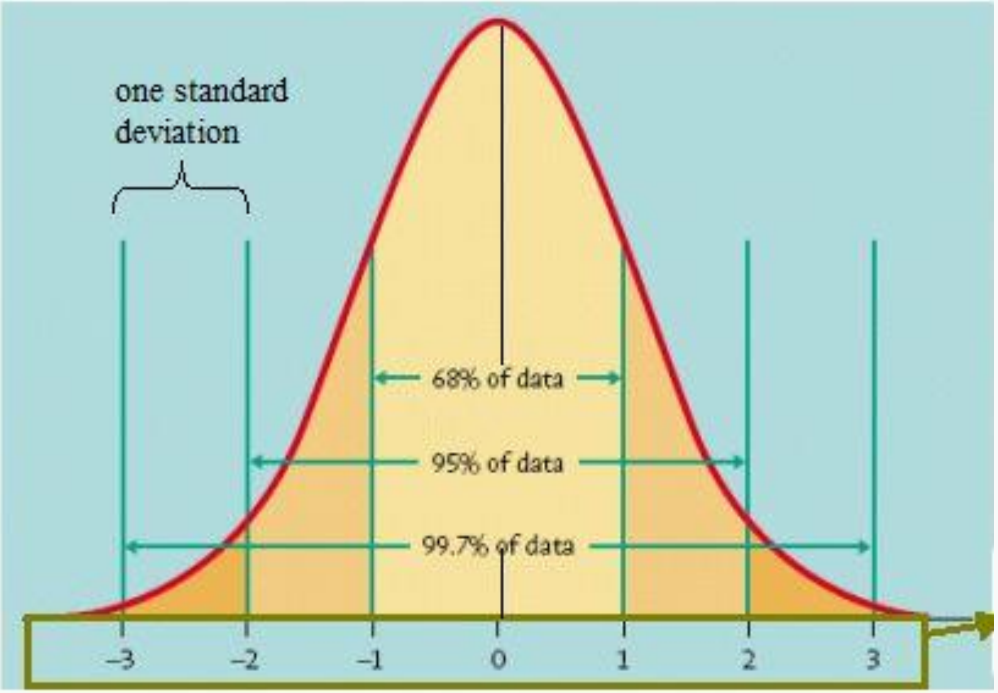


NO!

Key: we don't know the variance of the output

Variance

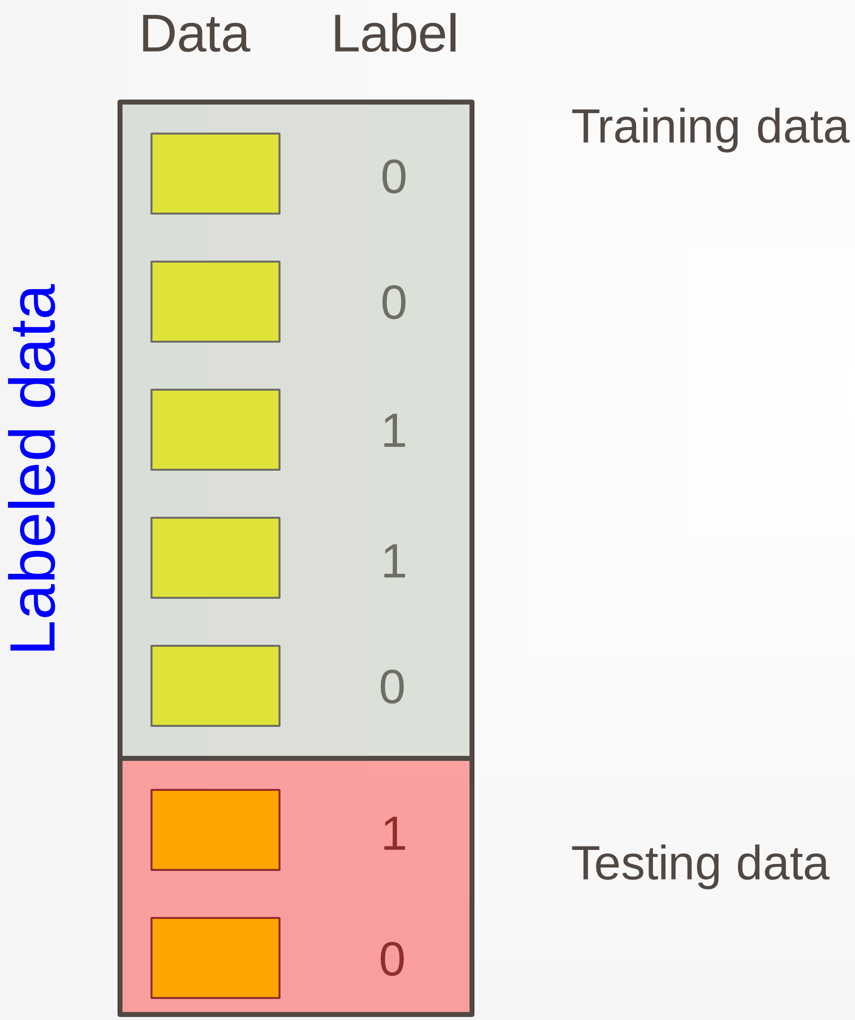
- Recall that variance (or standard deviation) helped us predict how likely certain events are:



How do we know how variable a model's accuracy is?

We need multiple accuracy scores! Ideas?

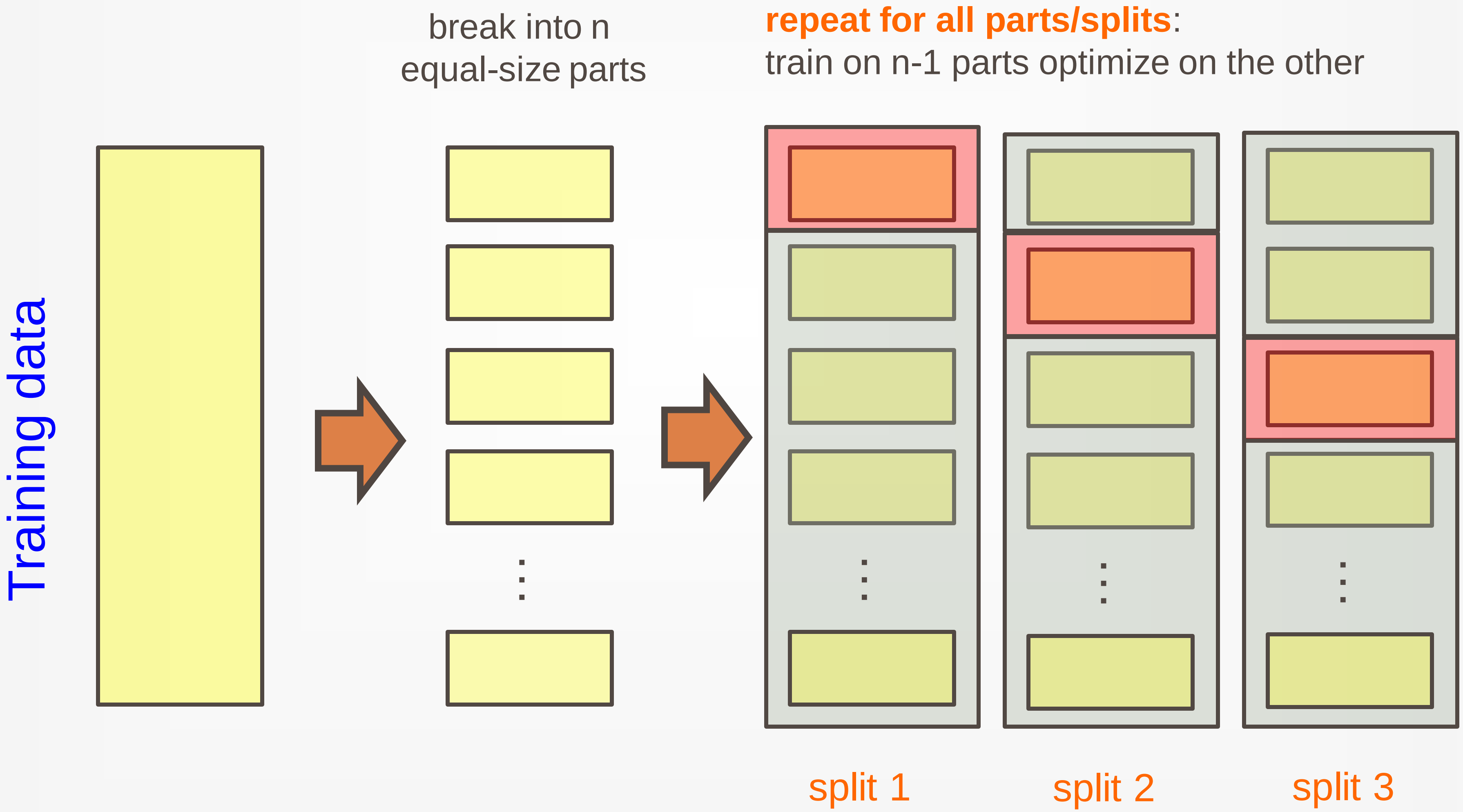
Repeated experimentation



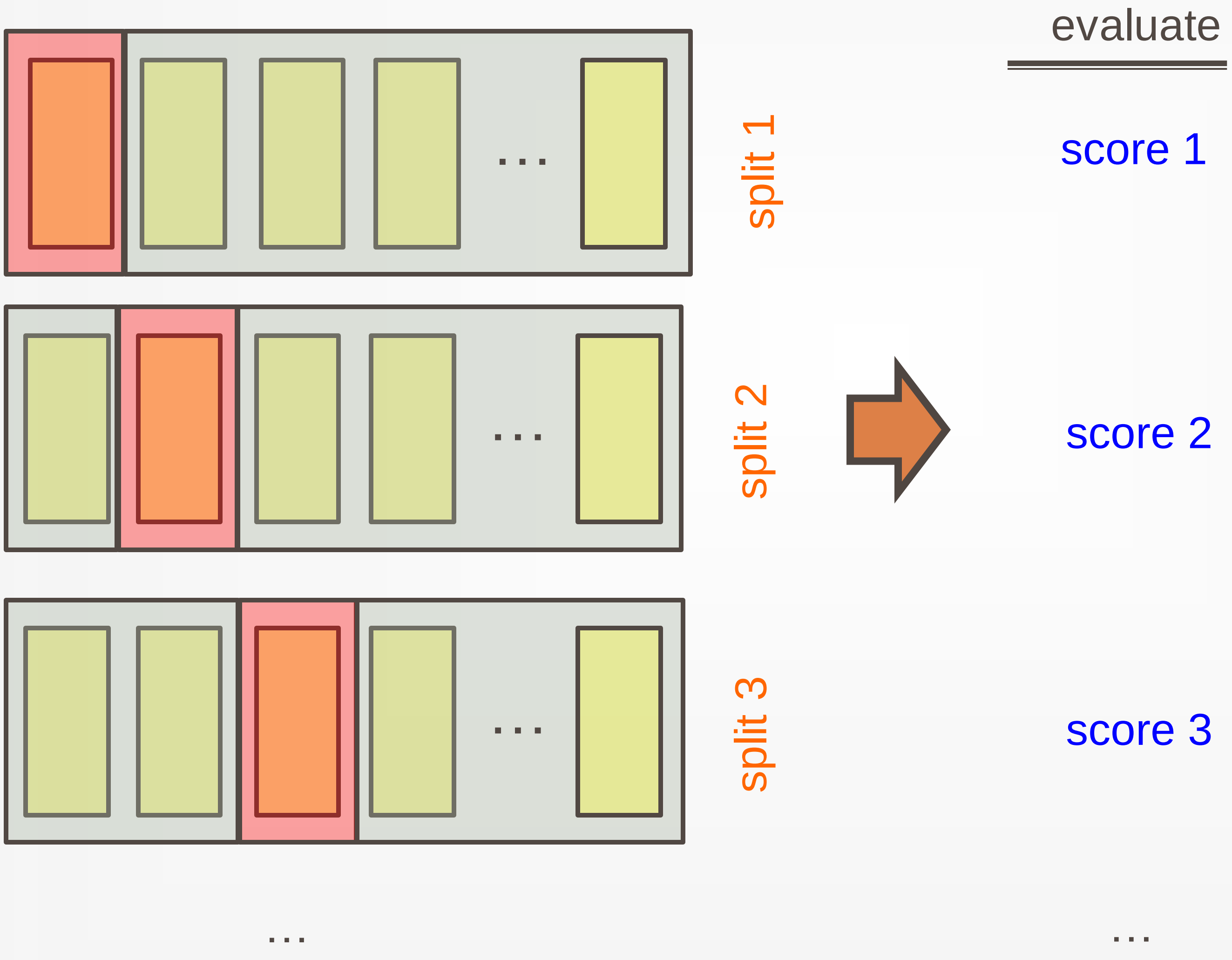
Rather than just splitting once, split multiple times

Repeated experimentation

n-fold cross validation



n-fold cross validation



n-fold cross validation

- better utilization of labeled data
- more robust: don’t just rely on one test/development set to evaluate the approach (or for optimizing parameters)
- multiplies the computational overhead by n (have to train n models instead of just one)
- 10 is the most common choice of n

Leave-one-out cross validation

- n-fold cross validation where n = number of examples
- aka “jackknifing”
- pros/cons?
- when would we use this?

Leave-one-out cross validation

- Can be very expensive if training is slow and/or if there are a large number of examples

- Useful in domains with limited training data:
  - maximizes the data we can use for training

- Some classifiers are very amenable to this approach (e.g.?)

Comparing systems: sample 1

split	model 1	model 2
1	87	88
2	85	84
3	83	84
4	80	79
5	88	89
6	85	85
7	83	81
8	87	86
9	88	89
10	84	85
average:	85	85

Is model 2 better than model 1?

Comparing systems: sample 2

split	model 1	model 2
1	87	87
2	92	88
3	74	79
4	75	86
5	82	84
6	79	87
7	83	81
8	83	92
9	88	81
10	77	85
average:	82	85

Is model 2 better than model 1?

Comparing systems: sample 3

split	model 1	model 2
1	84	87
2	83	86
3	78	82
4	80	86
5	82	84
6	79	87
7	83	84
8	83	86
9	85	83
10	83	85
average:	82	85

Is model 2 better than model 1?

Comparing systems

split	model 1	model 2
1	84	87
2	83	86
3	78	82
4	80	86
5	82	84
6	79	87
7	83	84
8	83	86
9	85	83
10	83	85
average:	**82**	**85**

split	model 1	model 2
1	87	87
2	92	88
3	74	79
4	75	86
5	82	84
6	79	87
7	83	81
8	83	92
9	88	81
10	77	85
average:	**82**	**85**

What's the difference?

Even though the averages are same, the variance is different!

Comparing systems: sample 4

split	model 1	model 2
1	80	82
2	84	87
3	89	90
4	78	82
5	90	91
6	81	83
7	80	80
8	88	89
9	76	77
10	86	88
average:	**83**	**85**
std dev	**4.9**	**4.7**

Is model 2 better than model 1?

Comparing systems: sample 4

split	model 1	model 2	model 2 – model 1
1	80	82	2
2	84	87	3
3	89	90	1
4	78	82	4
5	90	91	1
6	81	83	2
7	80	80	0
8	88	89	1
9	76	77	1
10	86	88	2
average:	**83**	**85**	
std dev	**4.9**	**4.7**	

Is model 2 better than model 1?

Comparing systems: sample 4

split	model 1	model 2	model 2 – model 1
1	80	82	2
2	84	87	3
3	89	90	1
4	78	82	4
5	90	91	1
6	81	83	2
7	80	80	0
8	88	89	1
9	76	77	1
10	86	88	2
average:	**83**	**85**	
std dev	**4.9**	**4.7**	

Model 2 is ALWAYS better

Comparing systems: sample 4

split	model 1	model 2	model 2 – model 1
1	80	82	2
2	84	87	3
3	89	90	1
4	78	82	4
5	90	91	1
6	81	83	2
7	80	80	0
8	88	89	1
9	76	77	1
10	86	88	2
average:	**83**	**85**	
std dev	**4.9**	**4.7**	

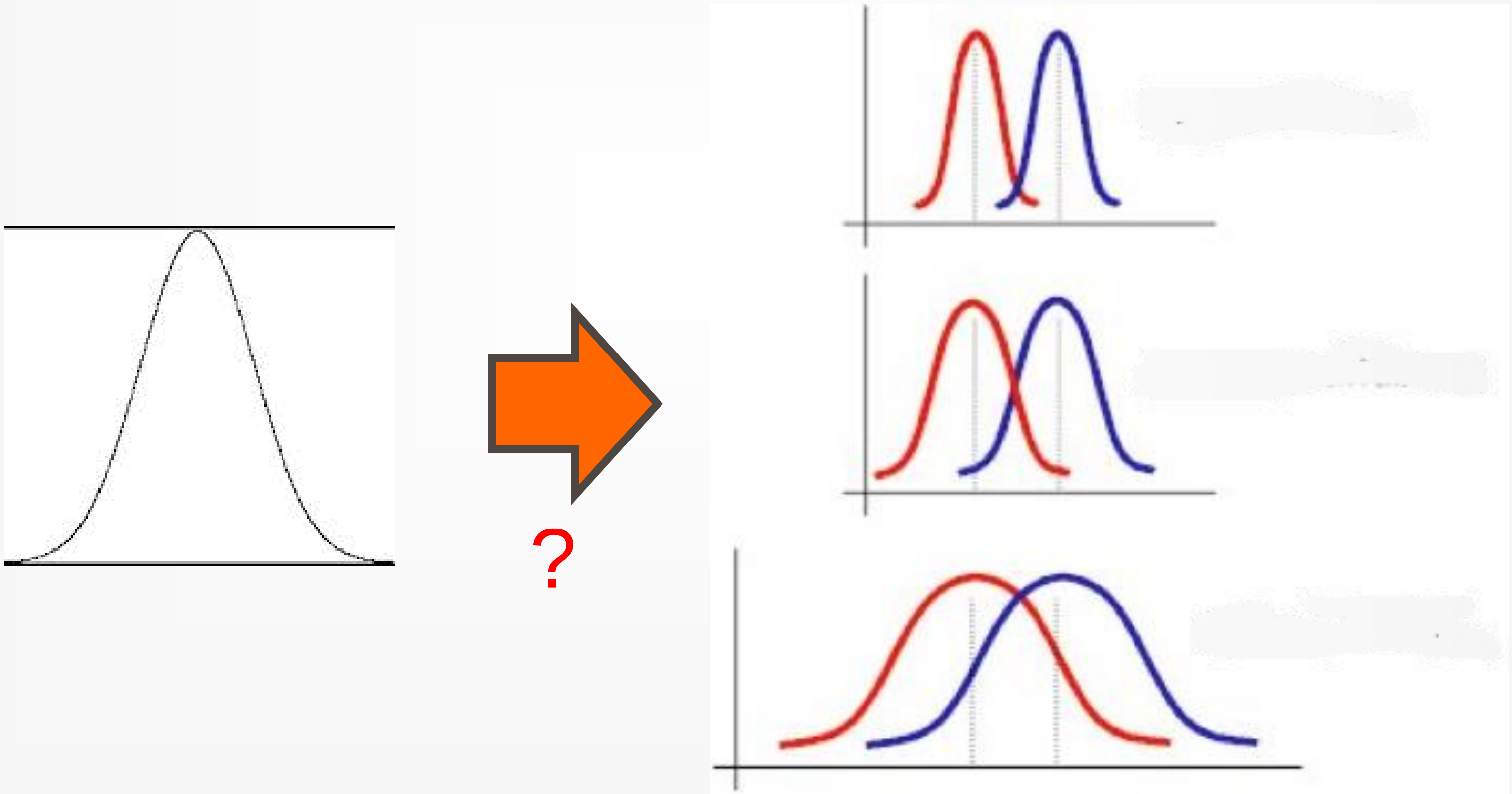
How do we decide if model 2 is better than model 1?

Statistical tests

- Setup:
  - Assume some default hypothesis about the data that you'd like to disprove, called the null hypothesis
  - e.g. model 1 and model 2 are not statistically different in performance
- Test:
  - Calculate a test statistic from the data (often assuming something about the data)
  - Based on this statistic, with some probability we can reject the null hypothesis, that is, show that it does not hold

t-test

- Determines whether two samples come from the same underlying distribution or not



t-test

- Null hypothesis: model 1 and model 2 accuracies are no different, i.e. come from the same distribution

- Assumptions: there are a number that often aren’t completely true, but we’re often not too far off

- Result: probability that the difference in accuracies is due to random chance (low values are better)

Calculating t-test

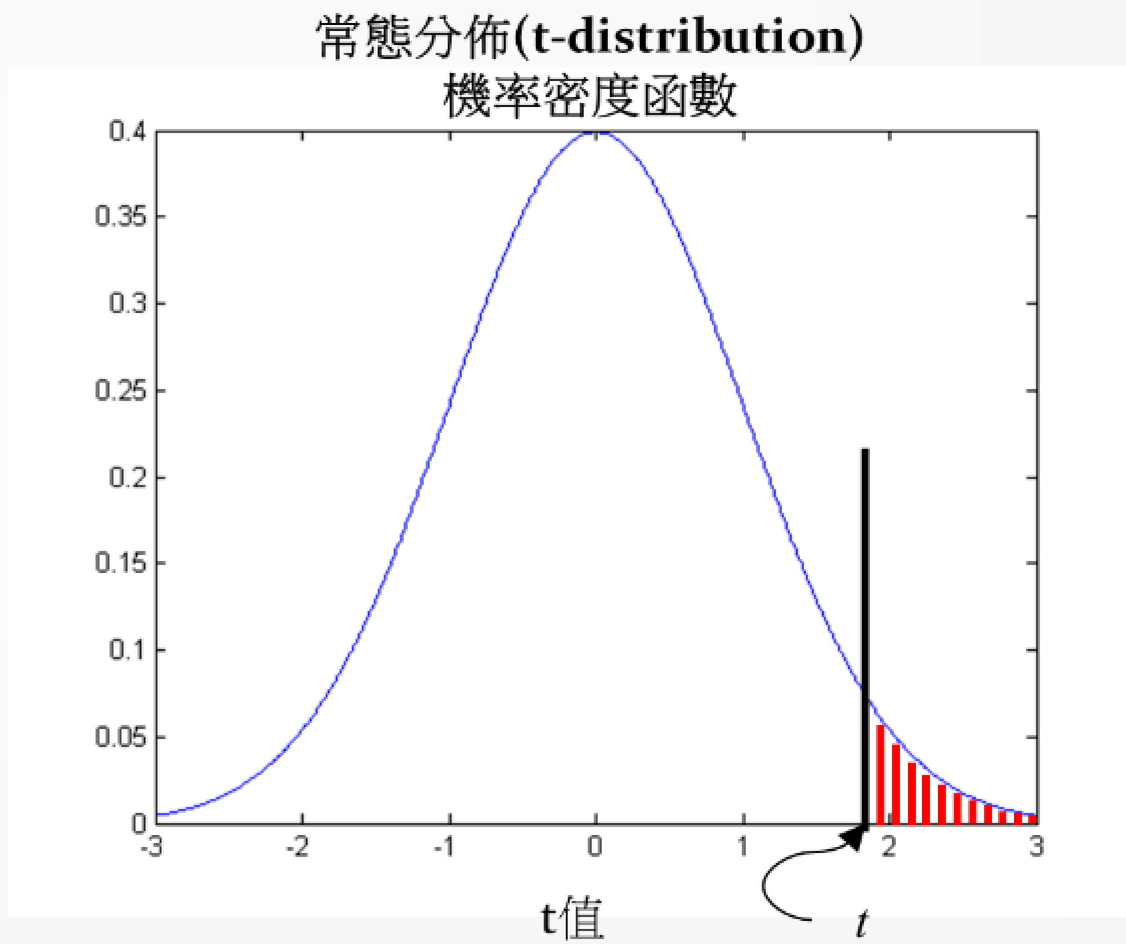
- For our setup, we'll do what's called a "pair t-test"
  - The values can be thought of as pairs, where they were calculated under the same conditions
  - In our case, the same train/test split
  - Gives more power than the unpaired t-test (we have more information)
- For almost all experiments, we'll do a "two-tailed" version of the t-test
- Can calculate by hand or in code, but why reinvent the wheel: use excel or a statistical package
- http://en.wikipedia.org/wiki/Student's_t-test

p-value

- The result of a statistical test is often a p-value
- p-value: the probability that the null hypothesis holds. Specifically, if we re-ran this experiment multiple times (say on different data) what is the probability that we would reject the null hypothesis incorrectly (i.e. the probability we'd be wrong)
- Common values to consider "significant": 0.05 (95% confident), 0.01 (99% confident) and 0.001 (99.9% confident)

t-test and t-value

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{\sigma_1}{N_1} - \frac{\sigma_2}{N_2}}}$$



t-value to p-value

- As we may know t-value is a specific cutoff point
  - But p-value is an area outsides t-value
    - Calculus, distribution
- What’s distribution?
  - Normal distribution but not normal typically
- The t distribution is used to represent the "normal distribution under sampling“
  - Define “degree of freedom”
    - Simply say, N1+N2-2
  - Now the t distribution can be defined as

$$f(t) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\,\Gamma\left(\frac{\nu}{2}\right)}\left(1+\frac{t^2}{\nu}\right)^{-\frac{\nu+1}{2}}$$

$$\Gamma(x) = \int_0^\infty \frac{t^{x-1}}{e^t}\,dt$$

Proof (optional)

$$
\begin{aligned}
&\mathrm{E}\left[X^2\right] \\
&= \int_{-\infty}^{\infty} x^2 f_X(x)dx \\
&= \int_{-\infty}^{0} x^2 f_X(x)dx + \int_{0}^{\infty} x^2 f_X(x)dx \\
&= -\int_{\infty}^{0} t^2 f_X(-t)dt + \int_{0}^{\infty} x^2 f_X(x)dx \quad \text{(change of variable in the first integral: } t = -x\text{)} \\
&= \int_{0}^{\infty} t^2 f_X(-t)dt + \int_{0}^{\infty} x^2 f_X(x)dx \quad \text{(exchanging the bounds of integration)} \\
&= \int_{0}^{\infty} t^2 f_X(t)dt + \int_{0}^{\infty} x^2 f_X(x)dx \quad \text{(since } f_X(-t) = f_X(t)\text{)} \\
&= 2\int_{0}^{\infty} x^2 f_X(x)dx \\
&= 2c\int_{0}^{\infty} x^2 \left(1 + \frac{x^2}{n}\right)^{-\frac{1}{2}(n+1)} dx \\
&= 2c\int_{0}^{\infty} nt(1+t)^{-n/2-1/2} \frac{\sqrt{n}}{2}\frac{1}{\sqrt{t}}dt \quad \text{(by a change of variable: } t = \frac{x^2}{n}\text{)} \\
&= cn^{3/2}\int_{0}^{\infty} t^{3/2-1}(1+t)^{-3/2-(n/2-1)}dt \\
&= cn^{3/2}B\left(\frac{3}{2}, \frac{n}{2} - 1\right) \quad \text{(integral representation of the Beta function)} \\
&= \frac{1}{\sqrt{n}}\frac{1}{B\left(\frac{n}{2}, \frac{1}{2}\right)} n^{3/2} B\left(\frac{1}{2} + 1, \frac{n}{2} - 1\right) \quad \text{(by the definition of } c\text{)} \\
&= n\frac{\Gamma\left(\frac{n}{2} + \frac{1}{2}\right)}{\Gamma\left(\frac{n}{2}\right)\Gamma\left(\frac{1}{2}\right)} \frac{\Gamma\left(\frac{1}{2} + 1\right)\Gamma\left(\frac{n}{2} - 1\right)}{\Gamma\left(\frac{n}{2} + \frac{1}{2}\right)} \quad \text{(by the definition of Beta function)} \\
&= n\frac{\Gamma\left(\frac{1}{2} + 1\right)\Gamma\left(\frac{n}{2} - 1\right)}{\Gamma\left(\frac{n}{2}\right)\Gamma\left(\frac{1}{2}\right)} \\
&= n\frac{\Gamma\left(\frac{1}{2}\right)\frac{1}{2}\Gamma\left(\frac{n}{2}\right)\frac{2}{n-2}}{\Gamma\left(\frac{n}{2}\right)\Gamma\left(\frac{1}{2}\right)} \quad \text{(because } \Gamma(z) = \Gamma(z-1)(z-1)\text{)} \\
&= \frac{n}{n-2} \\
&\mathrm{E}[X]^2 = 0 \\
&\mathrm{Var}[X] = \mathrm{E}\left[X^2\right] - \mathrm{E}[X]^2 = \frac{n}{n-2}
\end{aligned}
$$

Comparing systems: sample 1

split	model 1	model 2
1	87	88
2	85	84
3	83	84
4	80	79
5	88	89
6	85	85
7	83	81
8	87	86
9	88	89
10	84	85
average:	**85**	**85**

Is model 2 better than model 1?

They are the same with: $p = 1$

Comparing systems: sample 2

split	model 1	model 2
1	87	87
2	92	88
3	74	79
4	75	86
5	82	84
6	79	87
7	83	81
8	83	92
9	88	81
10	77	85
average:	**82**	**85**

Is model 2 better than model 1?

They are the same with: $p \sim= 0.14$

Calculating p-value

$$f(t) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\,\Gamma\left(\frac{\nu}{2}\right)}\left(1+\frac{t^2}{\nu}\right)^{-\frac{\nu+1}{2}}$$

- Based on t-value $t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{{\sigma_1}^2}{N_1} - \frac{{\sigma_2}^2}{N_2}}}$ = -.5517, degree of freedom = 18

$$f_{18}(t) = \frac{\Gamma(\frac{18+1}{2})}{\sqrt{18\pi}\,\Gamma(\frac{18}{2})}\left(1+\frac{t^2}{18}\right)^{-\frac{18+1}{2}} = \frac{\Gamma(9.5)}{\sqrt{18\pi}\,\Gamma(9)}\left(1+\frac{t^2}{18}\right)^{-9.5}$$

we also know

$$p_{18}(t \geq \text{t−value}) = \int_t^{\infty} f_{18}(t)dt = 0.5 - \int_0^t f_{18}(t)dt$$

Calculating p-value

$$f(t) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\,\Gamma\left(\frac{\nu}{2}\right)}\left(1+\frac{t^2}{\nu}\right)^{-\frac{\nu+1}{2}}$$

- So we have

$$p_{18}(t \geq 1.5517) = 0.5 - \int_0^{1.5517} f_{18}(t)dt$$

$$= 0.5 - \int_0^{1.5517} \frac{\Gamma(9.5)}{\sqrt{18\pi}\Gamma(9)}\left(1+\frac{t^2}{18}\right)^{-9.5}$$

$$= 0.5 - 0.393 * \int_0^{1.5517} \left(1+\frac{t^2}{18}\right)^{-9.5}$$

$$= 0.5 - 1.09528976764558 4 * 0.393442517730356 6 \approx 0.07$$

Comparing systems: sample 3

split	model 1	model 2
1	84	87
2	83	86
3	78	82
4	80	86
5	82	84
6	79	87
7	83	84
8	83	86
9	85	83
10	83	85
average:	**82**	**85**

Is model 2 better than model 1?

They are the same with: $p = 0.007$

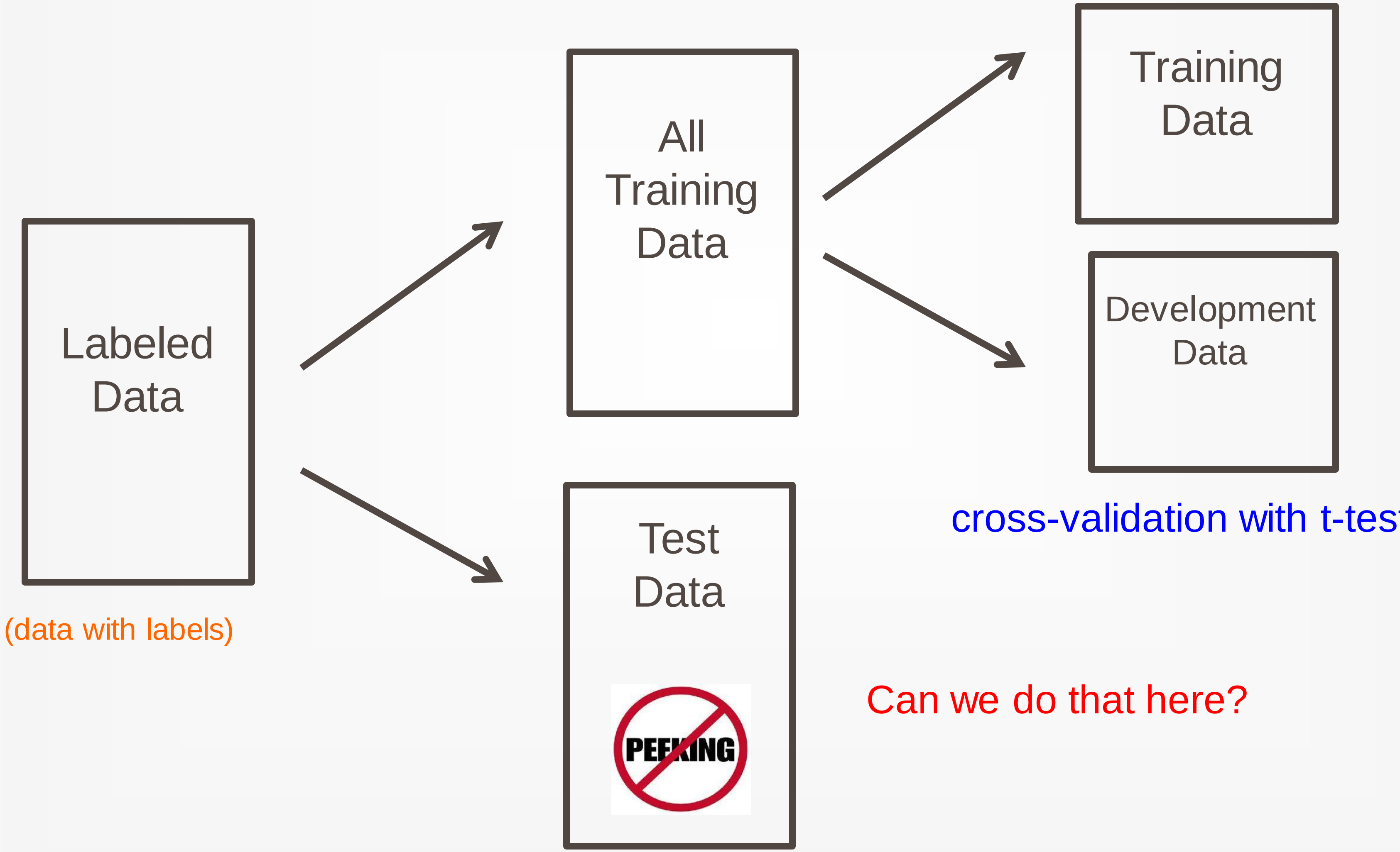
Comparing systems: sample 4

split	model 1	model 2
1	80	82
2	84	87
3	89	90
4	78	82
5	90	91
6	81	83
7	80	80
8	88	89
9	76	77
10	86	88
average:	**83**	**85**

Is model 2 better than model 1?

They are the same with: $p = 0.001$

Statistical tests on test data



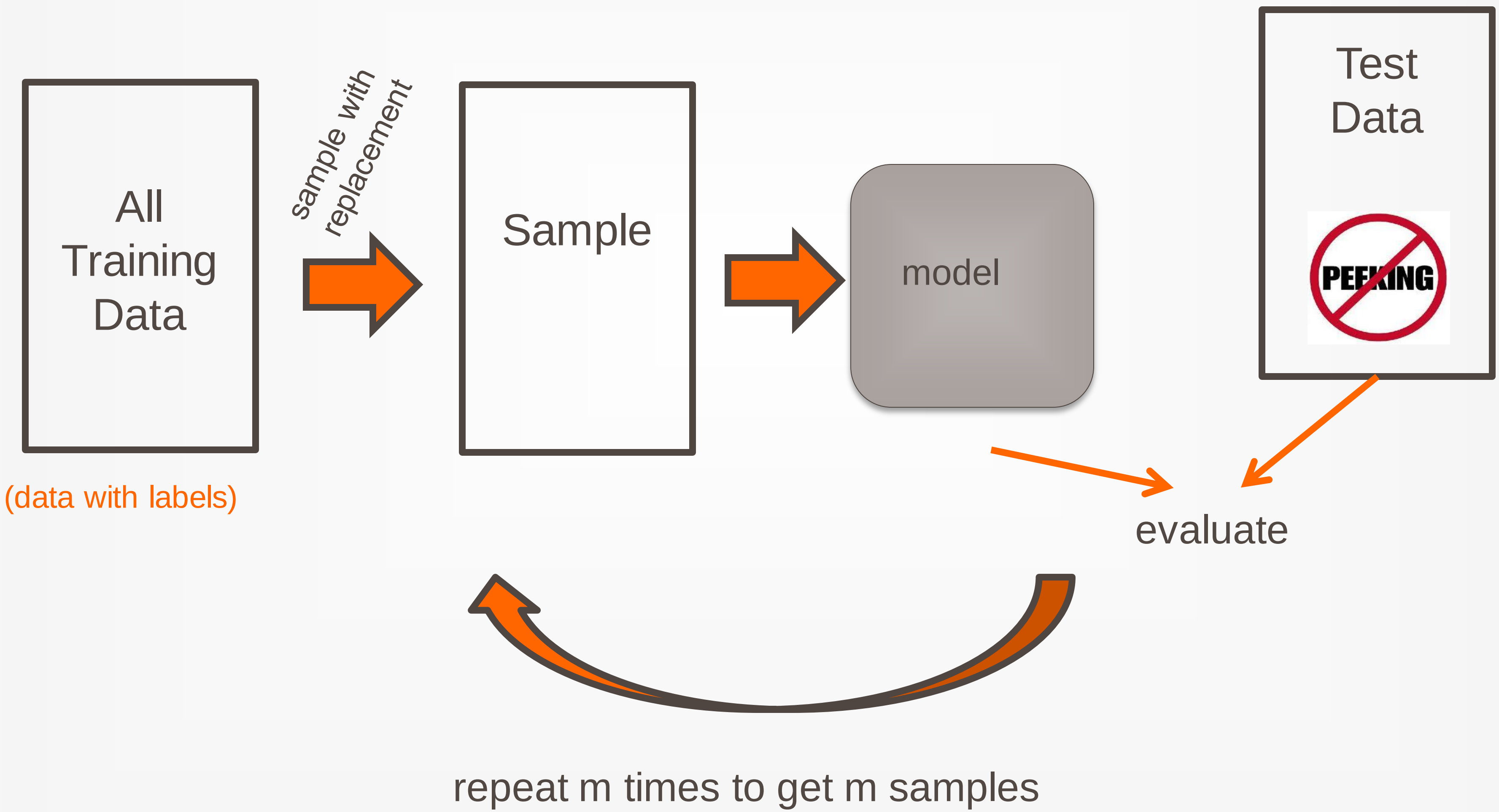
Bootstrap resampling

- training set t with n samples

- do m times:
- sample n examples with replacement from the training set to create a new training set *t'*
- train model(s) on *t'*
- calculate performance on test set

- calculate t-test (or other statistical test) on the collection of m results

Bootstrap resampling



Experimentation good practices

- Never look at your test data!

- During development
  - Compare different models/hyperparameters on development data
  - use cross-validation to get more consistent results
  - If you want to be confident with results, use a t-test and look for $p = 0.05$

- For final evaluation, use bootstrap resampling combined with a t-test to compare final approaches