CLASSIFICATION

UNIVERSITY OF CALGARY

FOR LATER

DOWNLOAD THE

"Exercise - Classification.zip"

NOTEBOOK & DATASET

FROM THE COURSE SITE

CLASSIFICATION

Classification and **Types of Classifiers**

Some simple classification algorithms

- k-nearest neighbors (kNN)
- Naïve Bayes

Data Story Presentations

Test/Train & Cross-Validation Procedures

Evaluation

A Few More Clustering Techniques

MACHINE LEARNING APPROACHES

	continuous	categorical
supervised	regression	classification
unsupervised	dimension reduction	clustering


TRAINING DATA +
LABELS
CLASSIFIER


UNLABELED DATA
CLASSIFIER

CLUSTERING VS. CLASSIFICATION

Unsupervised

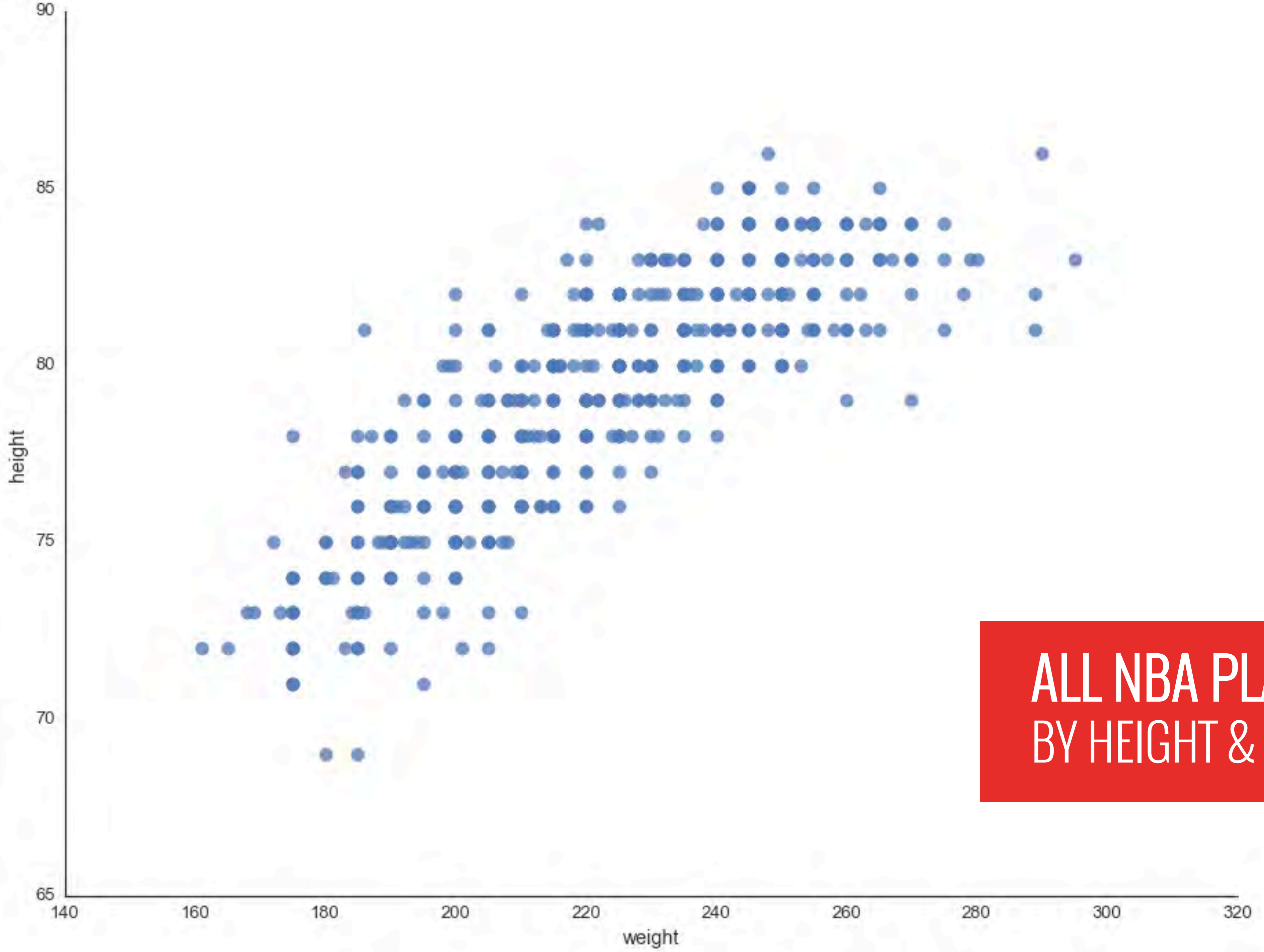
We think there are categories or groupings, but we **don't know exactly what they are.**

For example – Different types of people, countries, animals, etc.

Often **supervised** (or semi-supervised)

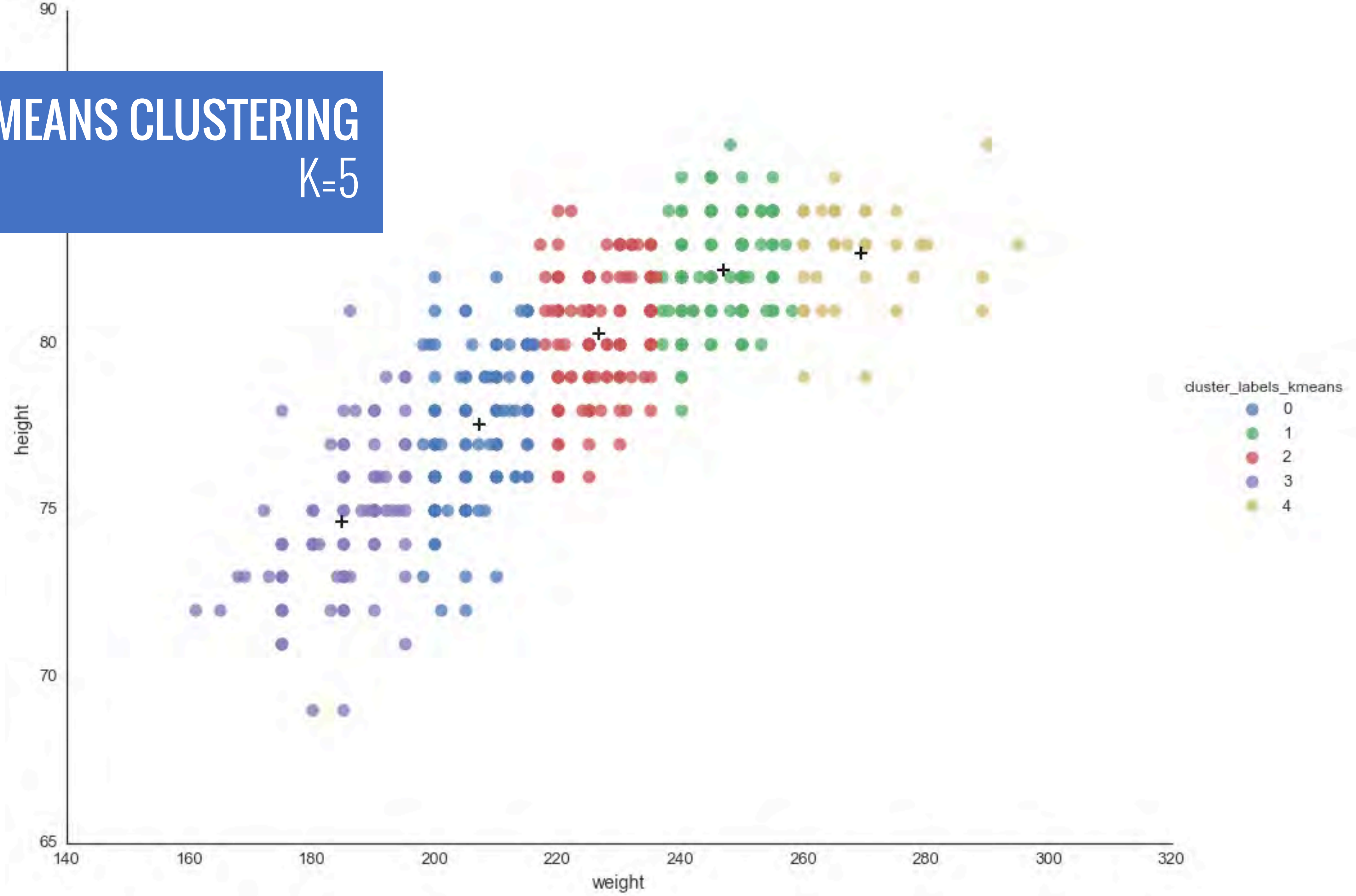
We know specific categories and want to apply them:

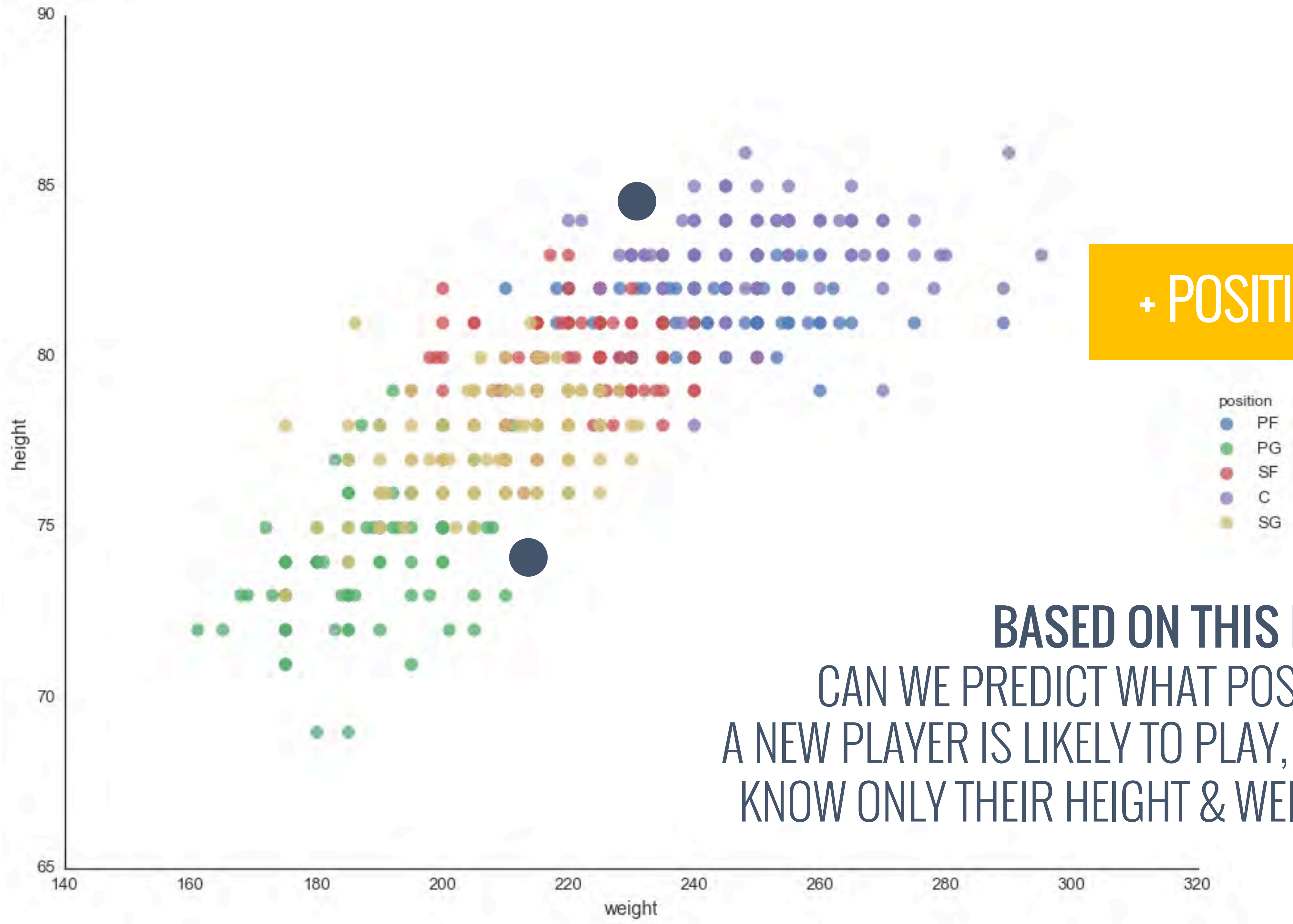
Spam versus **Not-spam**
English versus **Spanish** versus **French**
Pop versus **Classical**



ALL NBA PLAYERS
BY HEIGHT & WEIGHT

# K-MEANS CLUSTERING
K=5





+ POSITION

BASED ON THIS DATA
CAN WE PREDICT WHAT POSITION
A NEW PLAYER IS LIKELY TO PLAY, IF WE
KNOW ONLY THEIR HEIGHT & WEIGHT?

CLUSTERING VS. CLASSIFICATION

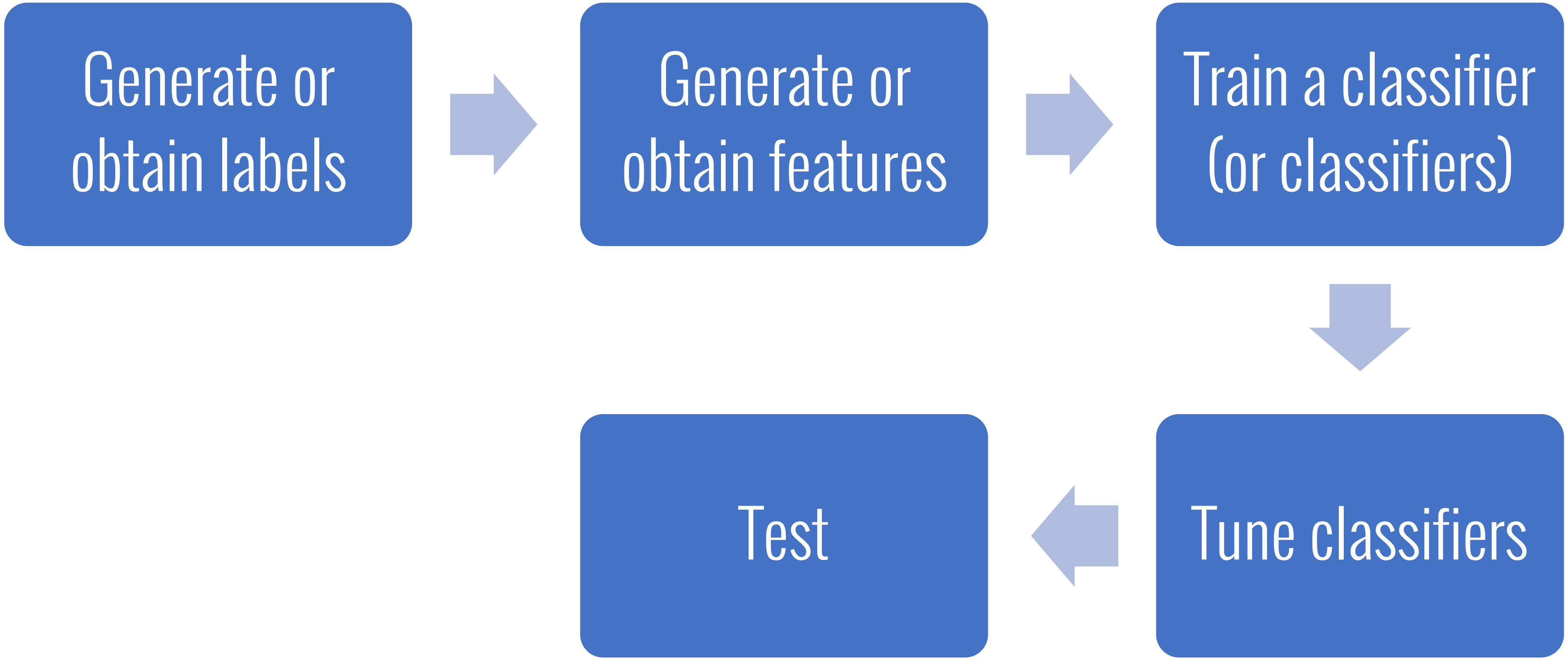
Clustering

Tries to separate groups by using (dis)simillarity
Is player 1 like player 2 but unlike player 3?

Classification

Tries to find important features and weights
What features distinguish a point guard from a center?

CLASSIFICATION PIPELINE



GETTING LABELS

USUALLY THIS IS DATA YOU DON'T HAVE
IF YOU DID, YOU WOULDN'T NEED A CLASSIFIER

The painful part

- Often human labor
  - Hire a bunch of out of work musicians to label your music for you
  - Use Mechanical Turk

Can distribute the pain

- Every time an individual hits the "spam" button in gmail

Can infer labels

- Predict gender by writing style
- Find articles with a byline → guess gender by name (database of names) → tag article by guessed gender
- Noisy, but might be mostly right

Can generate

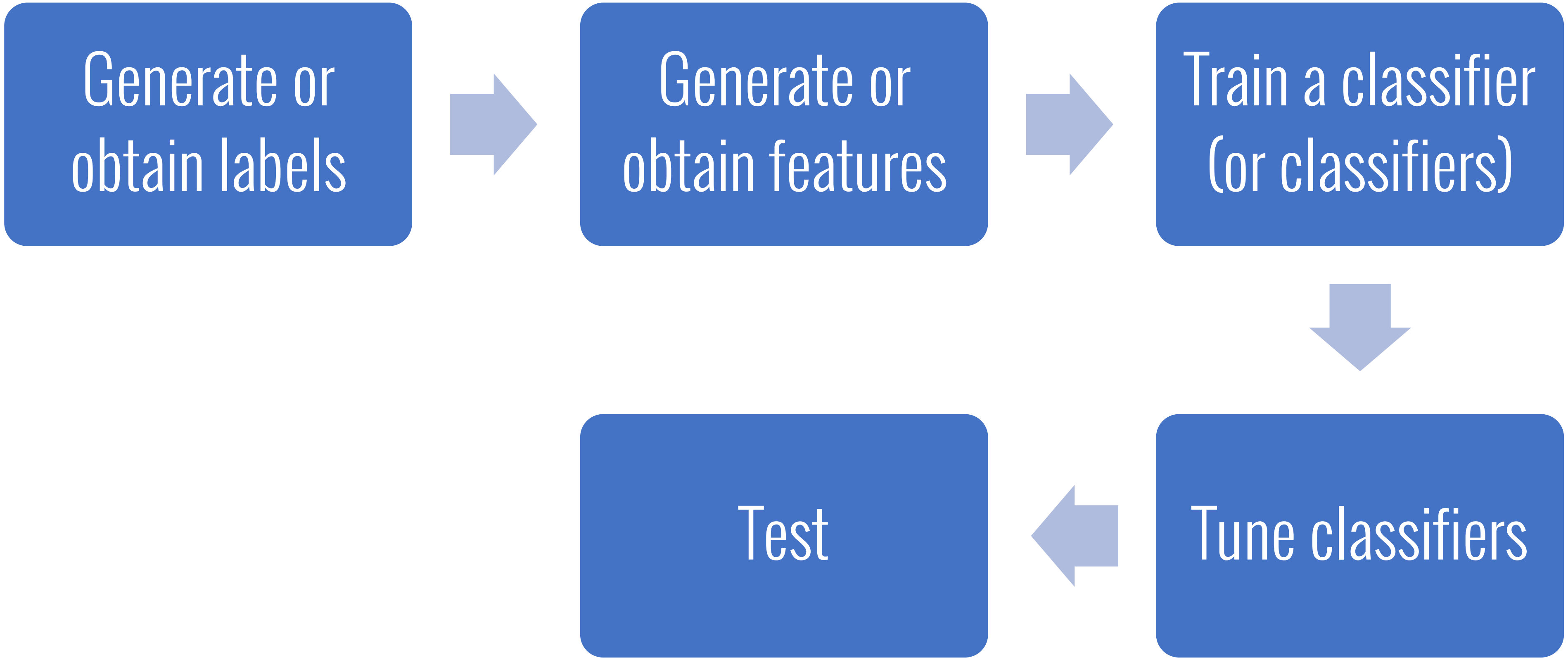
- Can sometimes generate artificial datasets
  - Make images of cats and dogs in different poses
- Since we generated, we know the labels

TRADEOFFS

How **much** data do I need?
How **hard/expensive** is it to label?
How **accurate** do I need the labels to be?
How much **will the “experts” agree?**

...

CLASSIFICATION PIPELINE



FEATURES

Same idea as in clustering

Some set of “descriptions” for an object

Explicit: Petal length, player height, miles per gallon, number of times word V1@gra is seen in text, shares sold last period

Inferred/calculated: average rate of change in shares sold in last month

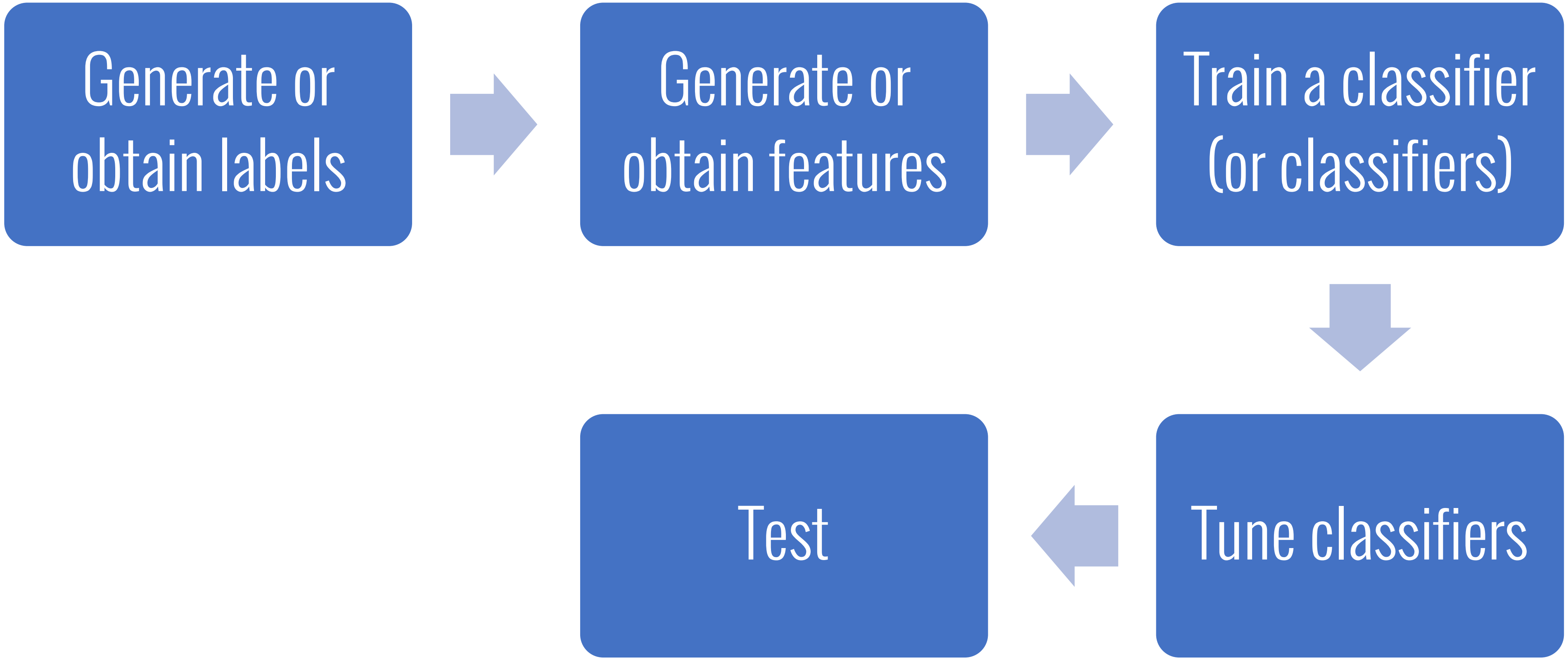
FEATURE ENGINEERING

Could fill an entire course by itself…

Some rules/suggestions:

- If you have lots of features you need **lots of examples**
- Use EDA techniques!
  - Look at **distributions**
    (is MPG for sports cars obviously different than MPG for station wagons?)
- Start with the **most discriminative features**

CLASSIFICATION PIPELINE



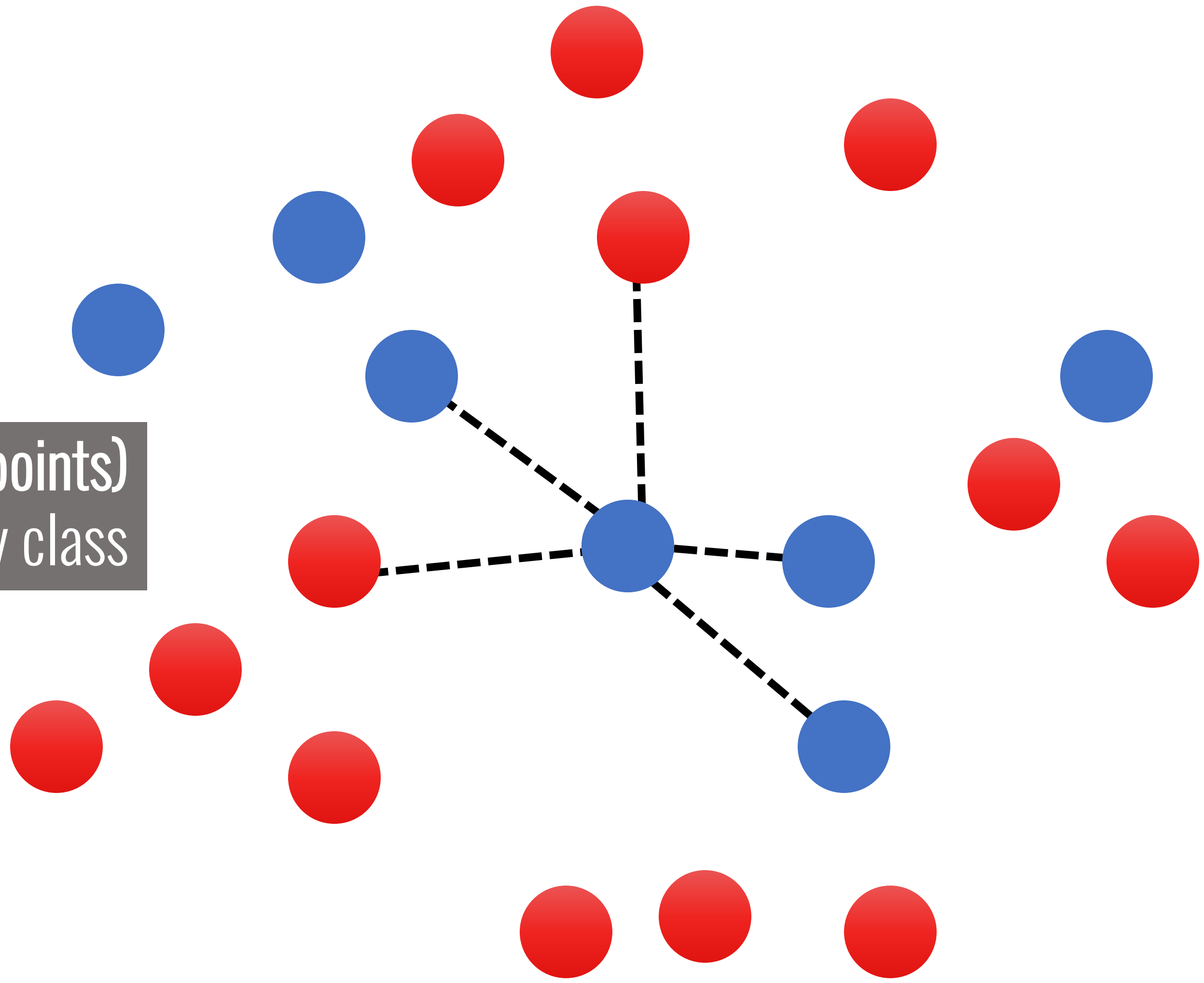
SOME BASIC/POPULAR CLASSIFIERS

kNN (k-Nearest-Neighbor)
Naïve Bayes
Logistic Regression
Decision Trees
Random Forests
SVM (Support-Vector Machines)
Neural Nets ("Deep Learning")

K-NEAREST NEIGHBOR

(Classification via similarity.)

PICK THE CLOSEST *k* POINTS



K=5 (find closest 5 points)
Pick the majority class

K-NEAREST NEIGHBOR CLASSIFIER

Advantages

No training needed
Can be applied to **any distance measure** and feature representation
Empirically **effective**

Disadvantages

Finding nearest neighbors has **high time complexity**
Need to keep the **whole training set**
Imprecise with small numbers of examples
- often true in high-dimensional spaces (neighbors aren't similar enough to be trusted)

NAIVE BAYES CLASSIFICATION

(Classification via probabilistic reasoning.)

BAYES THEOREM

$$Prob(A\ given\ B) = \frac{Prob\ (A\ and\ B)}{Prob(B)}$$

GIVES THE PROBABILITY OF AN EVENT OCCURRING GIVEN THAT ANOTHER EVENT HAS ALREADY OCCURRED

BAYES THEOREM

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

GIVES THE PROBABILITY OF AN EVENT OCCURRING GIVEN THAT ANOTHER EVENT HAS ALREADY OCCURRED

PREDICTING LABELS

LIKELIHOOD OF FEATURES GIVEN LABEL

POSTERIOR PROBABILITY

(LIKELIHOOD OF LABEL GIVEN FEATURES)

LABEL PRIOR PROBABILITY

$$P(L|F) = \frac{P(F|L) \times P(L)}{P(F)}$$

FEATURE PRIOR PROBABILITY

THE PROBABILITY OF AN ITEM GETTING A PARTICULAR LABEL (L) GIVEN THAT IT CONTAINS A PARTICULAR FEATURE (F)

MULTIPLE FEATURES

POSTERIOR PROBABILITY
(LIKELIHOOD OF LABEL GIVEN FEATURES)

LIKELIHOOD OF
MULTIPLE FEATURES
GIVEN LABEL

LABEL PRIOR PROBABILITY

$$P(L|F) = \frac{P(f_1|L) \times P(f_2|L) \times \cdots \times P(f_n|L) \times P(L)}{P(F)}$$

FEATURE PRIOR PROBABILITY

ASSUMES (**NAIVELY**) THAT ALL FEATURES
ARE COMPLETELY INDEPENDENT
BUT CAN STILL BE REALLY EFFECTIVE!

FOR EXAMPLE

LIKELIHOOD OF FEATURES GIVEN LABEL

LABEL PRIOR PROBABILITY

POSTERIOR PROBABILITY

(LIKELIHOOD OF LABEL GIVEN FEATURES)

$$P(spam|words) = \frac{P(viagra, rich, \ldots friend|spam) \times P(spam)}{P(viagra, rich, \ldots friend)}$$

FEATURE PRIOR PROBABILITY

http://gerardnico.com/wiki/data_mining/naive_bayes

AN EVEN EASIER EXAMPLE - THE GOLF DATASET

Outlook	Temperature	Humidity	Windy	Play Golf
overcast	hot	high	FALSE	yes
overcast	cool	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
[illegible]	[illegible]	[illegible]	TRUE	no
[illegible]	[illegible]	[illegible]	FALSE	yes
[illegible]	[illegible]	[illegible]	TRUE	no
[illegible]	[illegible]	[illegible]	FALSE	no
[illegible]	[illegible]	[illegible]	TRUE	no
[illegible]	[illegible]	[illegible]	FALSE	no
[illegible]	[illegible]	[illegible]	FALSE	yes
sunny	mild	normal	TRUE	yes

HOW CAN WE USE THIS TO PREDICT WHETHER OR NOT WE'LL PLAY GOLF ON A NEW DAY? (EVEN IF WE HAVEN'T SEEN A PARTICULAR COMBINATION BEFORE)

COMPUTE PROBABILITY TABLES

Outlook					Temperature					Humidity					Wind				
	Yes	No	P(Yes)	P(No)		Yes	No	P(Yes)	P(No)		Yes	No	P(Yes)	P(No)		Yes	No	P(Yes)	P(No)
Sunny	2	3	2/9	3/5	Hot	2	2	2/9	2/5	High	3	4	3/9	4/5	False	6	2	6/9	2/5
Overcast	4	0	4/9	0/5	Mild	4	2	4/9	2/5	Normal	6	1	6/9	1/5	True	3	3	3/9	3/5
Rainy	3	2	3/9	2/5	Cool	3	1	3/9	1/5										
Total	**9**	**5**	**100%**	**100%**	**Total**	**9**	**5**	**100%**	**100%**	**Total**	**9**	**5**	**100%**	**100%**	**Total**	**9**	**5**	**100%**	**100%**

Play		P / Total
Yes	9	9/14
No	5	5/14
Total	14	100%

THEN USE THEM TO COMPUTE PROBABILITIES

For a new day that's **rainy**, **cool**, **high humidity**, and **windy:**

$$P(Yes|NewDay)=\frac{P(RainyOutlook|Yes)\times P(CoolTemperature|Yes)\times P(HighHumidity|Yes)\times P(WithWind|Yes)\times P(Yes)}{\cancel{P(NewDay)}}$$

$$P(No|NewDay)=\frac{P(RainyOutlook|No)\times P(CoolTemperature|No)\times P(HighHumidity|No)\times P(WithWind|No)\times P(No)}{\cancel{P(NewDay)}}$$

$$P(Yes|NewDay)+P(No|NewDay)=1$$

http://gerardnico.com/wiki/data_mining/naive_bayes

COMPUTE

$$P(Yes|NewDay) \propto \frac{3}{9} \cdot \frac{3}{9} \cdot \frac{3}{9} \cdot \frac{3}{9} \cdot \frac{9}{14} \approx 0.0079$$

$$P(No|NewDay) \propto \frac{2}{5} \cdot \frac{1}{5} \cdot \frac{4}{5} \cdot \frac{3}{5} \cdot \frac{5}{14} \approx 0.0137$$

NORMALIZE

$$P(Yes|NewDay) = \frac{0.0079}{0.0079+0.0137} = 0.36$$

$$P(No|NewDay) = \frac{0.0137}{0.0079+0.0137} = 0.63$$

CHOOSE A LABEL

NO GOLF!

http://gerardnico.com/wiki/data_mining/naive_bayes

NAÏVE BAYESIAN CLASSIFIER

Advantages:

- Easy to implement
- Very efficient
- Good results for many applications

Disadvantages

- **Assumes feature independence**, and can break when that assumption is seriously violated (highly correlated data sets).

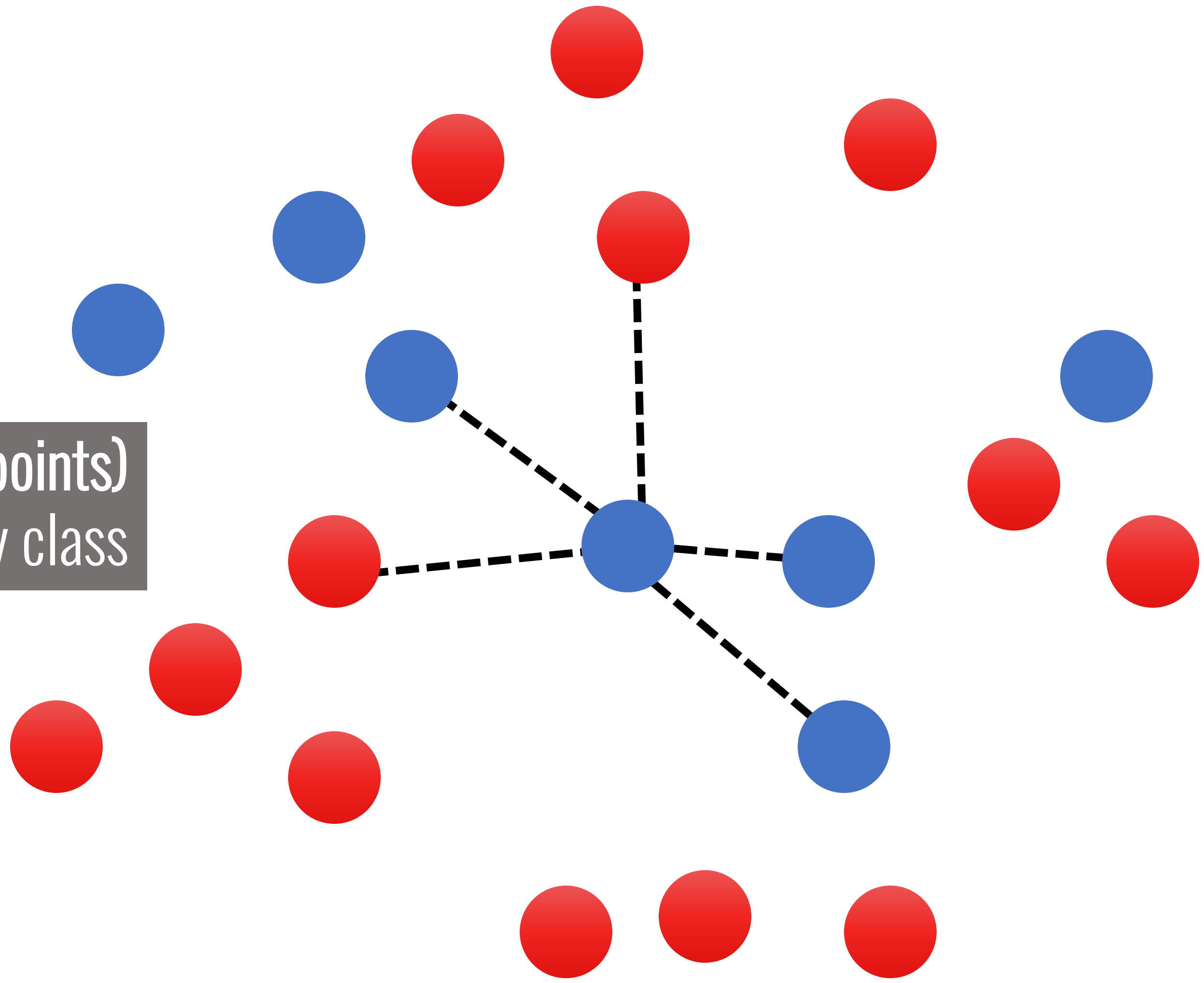
LET'S TRY SOME CLASSIFYING

TO FOLLOW ALONG

DOWNLOAD THE CLASSIFICATION NOTEBOOK & DATASET FROM THE COURSE PAGE

kNN (k-NEAREST NEIGHBORS)

K=5 (find closest 5 points)
Pick the majority class



NAÏVE BAYES CLASSIFICATION

$$P(Yes|NewDay) = \frac{P(RainyOutlook|Yes) \times P(CoolTemperature|Yes) \times P(HighHumidity|Yes) \times P(WithWind|Yes) \times P(Yes)}{\cancel{P(NewDay)}}$$

$$P(No|NewDay) = \frac{P(RainyOutlook|No) \times P(CoolTemperature|No) \times P(HighHumidity|No) \times P(WithWind|No) \times P(No)}{\cancel{P(NewDay)}}$$

Outlook						
	Yes	No	P(Yes)	P(No)		Yes
Sunny	2	3	2/9	3/5	Hot	2
Overcast	4	0	4/9	0/5	Mild	4
Rainy	3	2	3/9	2/5	Cool	3
Total	**9**	**5**	**100%**	**100%**	**Total**	**9**

COMPUTE

$$P(Yes|NewDay) \propto \frac{3}{9} \cdot \frac{3}{9} \cdot \frac{3}{9} \cdot \frac{3}{9} \cdot \frac{9}{14} \approx 0.0079$$

$$P(No|NewDay) \propto \frac{2}{5} \cdot \frac{1}{5} \cdot \frac{4}{5} \cdot \frac{3}{5} \cdot \frac{5}{14} \approx 0.0137$$

NORMALIZE

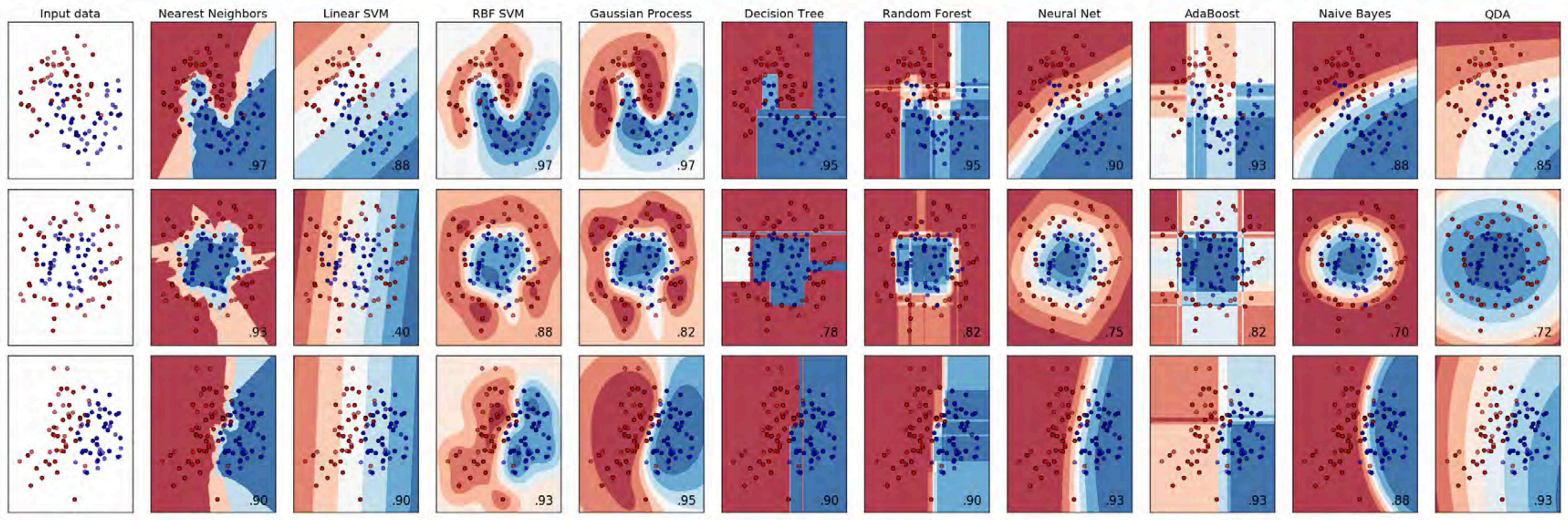
$$P(Yes|NewDay) = \frac{0.0079}{0.0079+0.0137} = 0.36$$

$$P(No|NewDay) = \frac{0.0137}{0.0079+0.0137} = 0.63$$

THERE ARE MANY OTHER OPTIONS AT YOUR DISPOSAL

Just a few of the classifiers available in scikit-learn:

```
from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
```



Different classifiers can give dramatically different results.

http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

BUT FIRST...

TRAINING

CLASSIFICATION TRAINING

All our algorithms need to be “trained”
Our algorithm also needs to be “tested”

What happens if we use all our data for training?

“Overfitting!”

CLASSIFICATION TRAINING

First impulse (usually a reasonable one)
- Split the data into **train** and **test** (aka “holdout”)
  - e.g., 70% to train, 30% retained to test

Still need to be careful
- You might need to train again!
- Some algorithms work best with **“balanced” training**
- Too many positive or negative examples can cause problems
- Sometimes we don’t have enough data to do this (expensive labels)
- Can still overfit

THREE-WAY SPLIT

Training

Validation

- Use to tune parameters

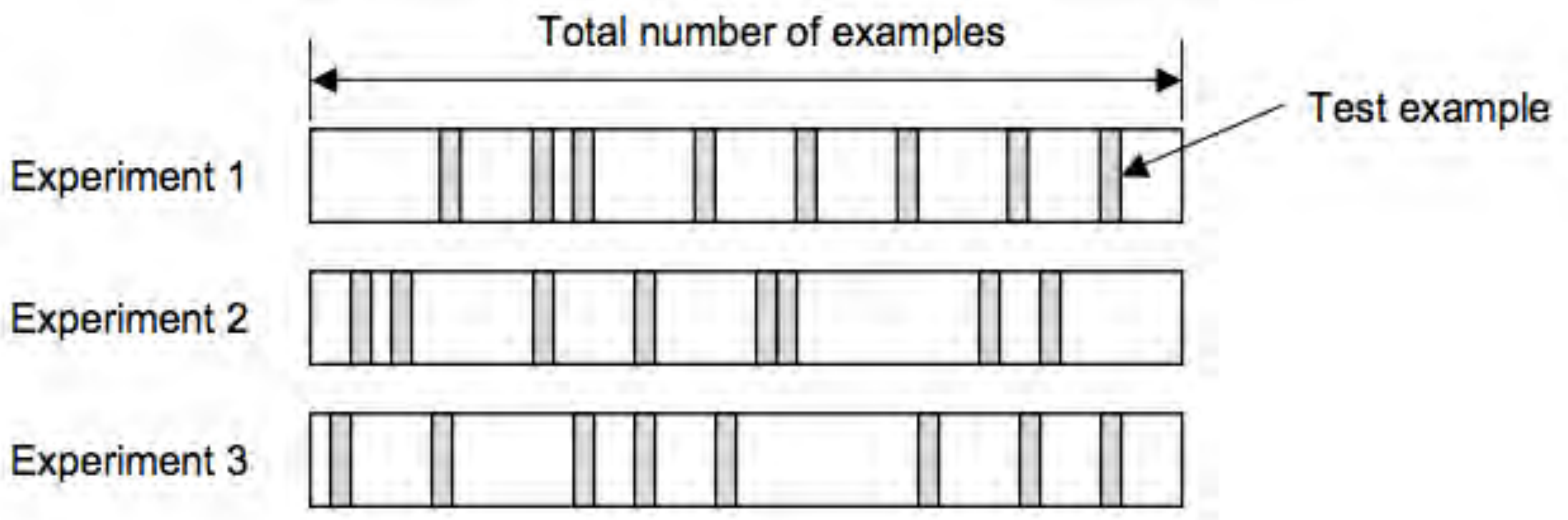
Test

- After running this one, you're done

CROSS-VALIDATION

An even better option – Try many holdouts

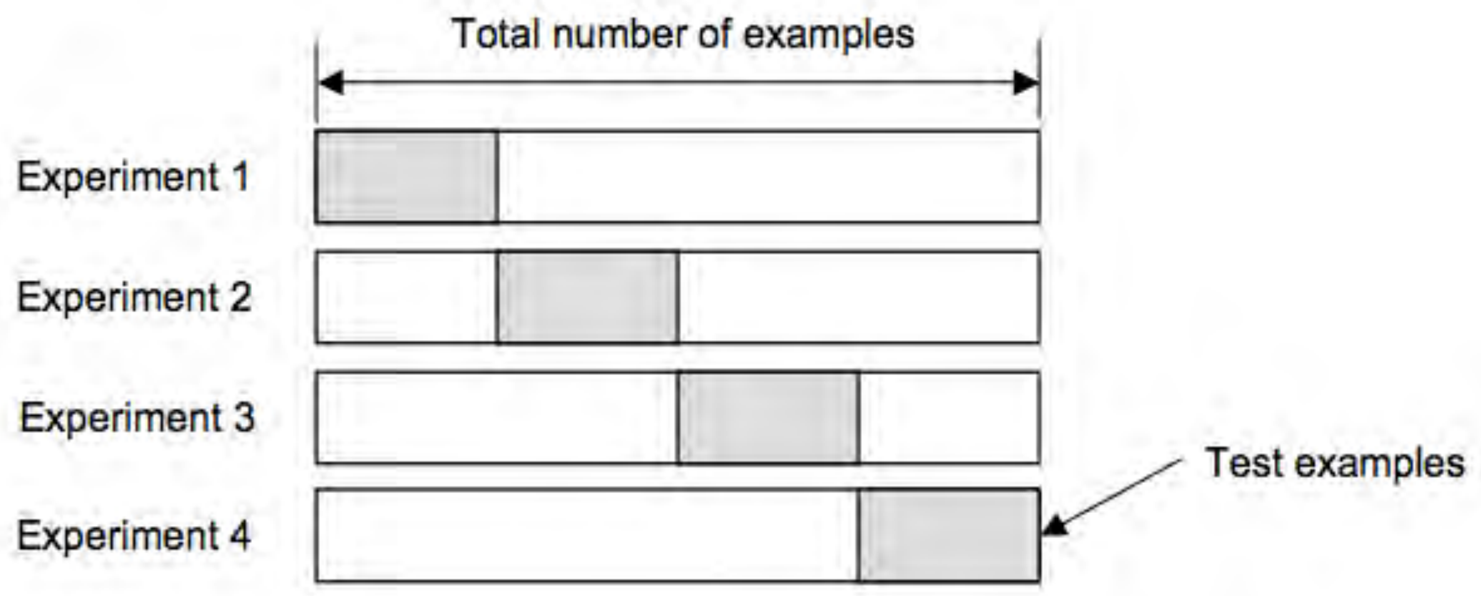
- Randomly select test/train data
  - Pick a different 70% each time
- Average scores at the end



EXTENSION: K-FOLD CROSS-VALIDATION

Move a sliding window

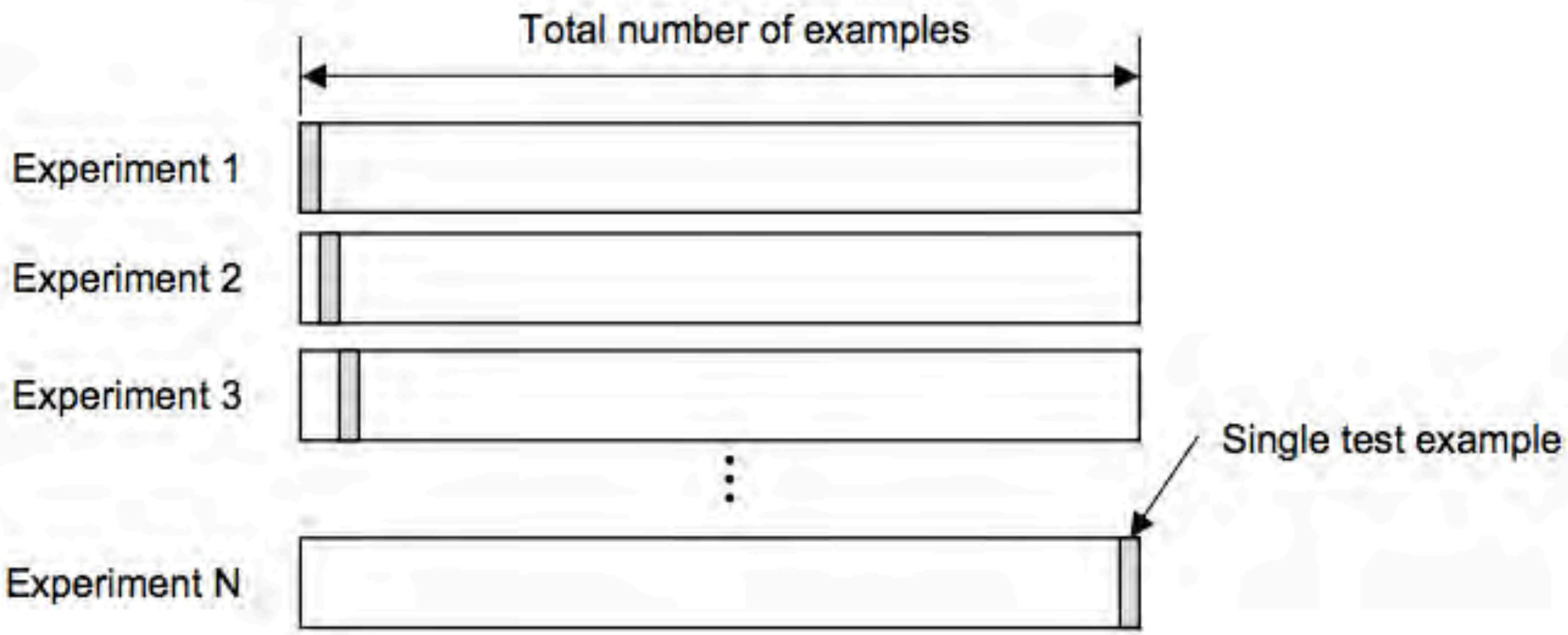
  Makes sure that all data is used



EXTENSION: LEAVE-ONE-OUT

Extreme version of k-fold

Remove one at a time, train on all the others



GENERAL GUIDANCE ON FOLDS

Many folds

- Accurate but expensive (variance might be high)

More data = get away with less folds

Sparse = use more folds

Common choice = 10

OK...
SO DOES IT WORK?

EVALUATION

Accuracy
Precision
Recall
F1 score
Confusion Matrix
Precision-Recall curve
ROC curve

ACCURACY

Simplest of evaluations

Count the percentage of times we classified correctly.

Algorithm:

```
For each item:
    Compare the label we generated to the ground truth
    Add 1 if correct, otherwise add 0
Divide by number of items
```

PROBLEMS WITH ACCURACY

Think of a spam filter, why might accuracy be the wrong measure?

Hint: What's the distribution of messages? How many spam and how many good?

To win at spam detection (or any unequal task)

– **just guess the majority type.**

If 90% of messages are good: guess that **all are good**

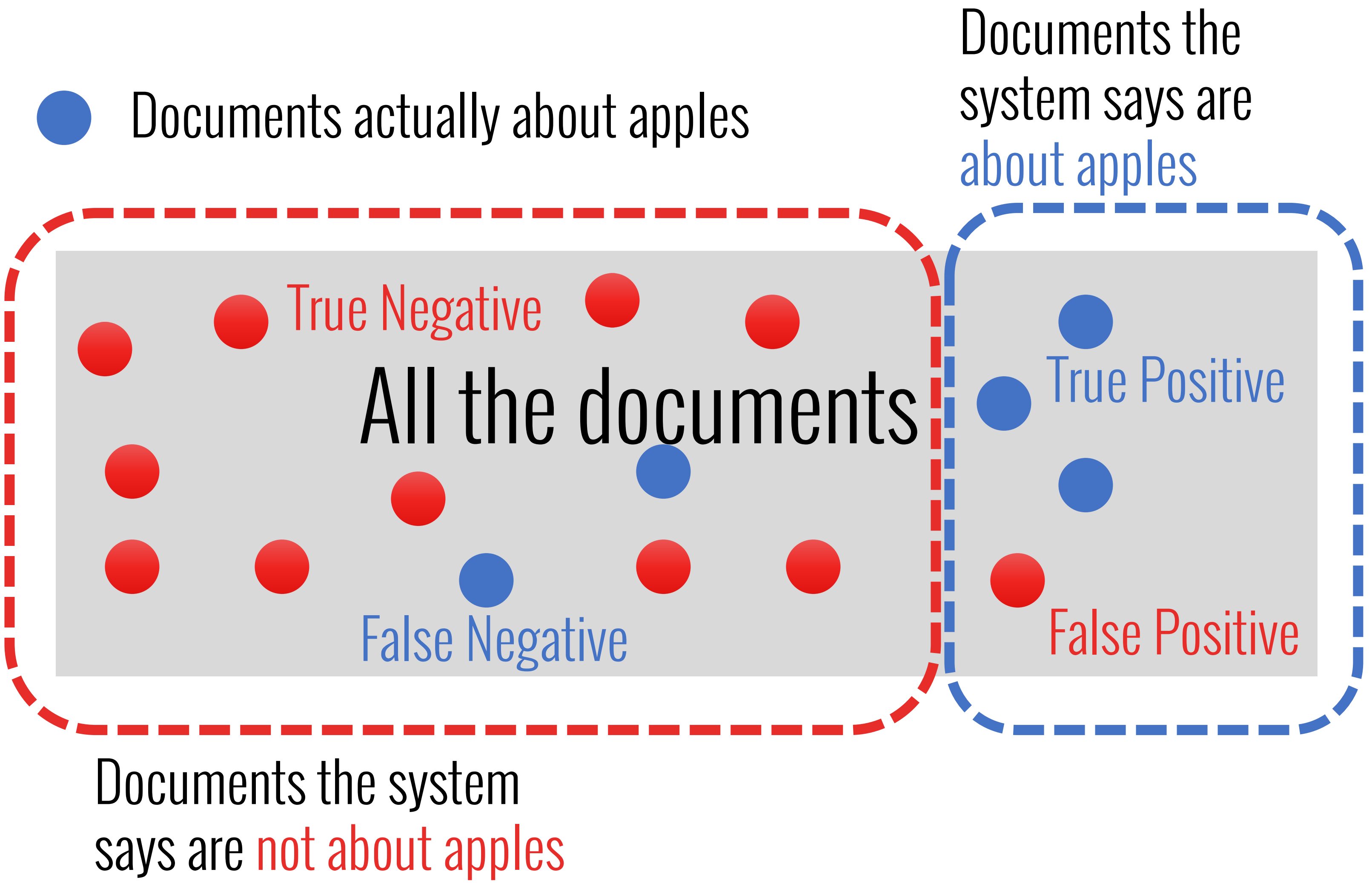
Accuracy is **guaranteed to be 90%**

PRECISION AND RECALL

Heavily used in Information Retrieval

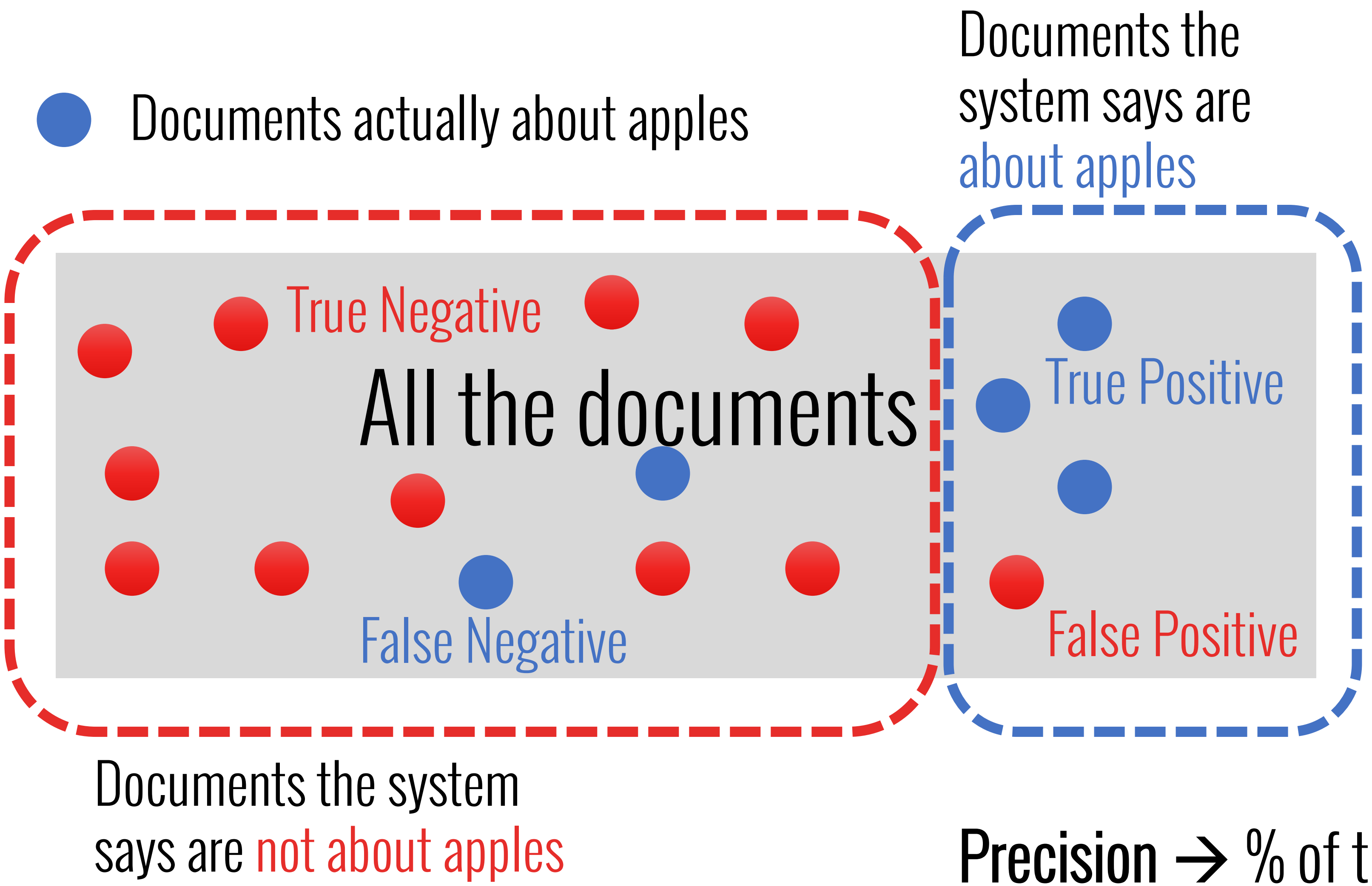
Assumption: we have a query "**apple**"

- Our corpus contains documents about apples (positive) and documents about other topics (negative)


Documents actually about apples
Documents the system says are about apples
True Negative
All the documents
True Positive
False Negative
False Positive
Documents the system says are not about apples

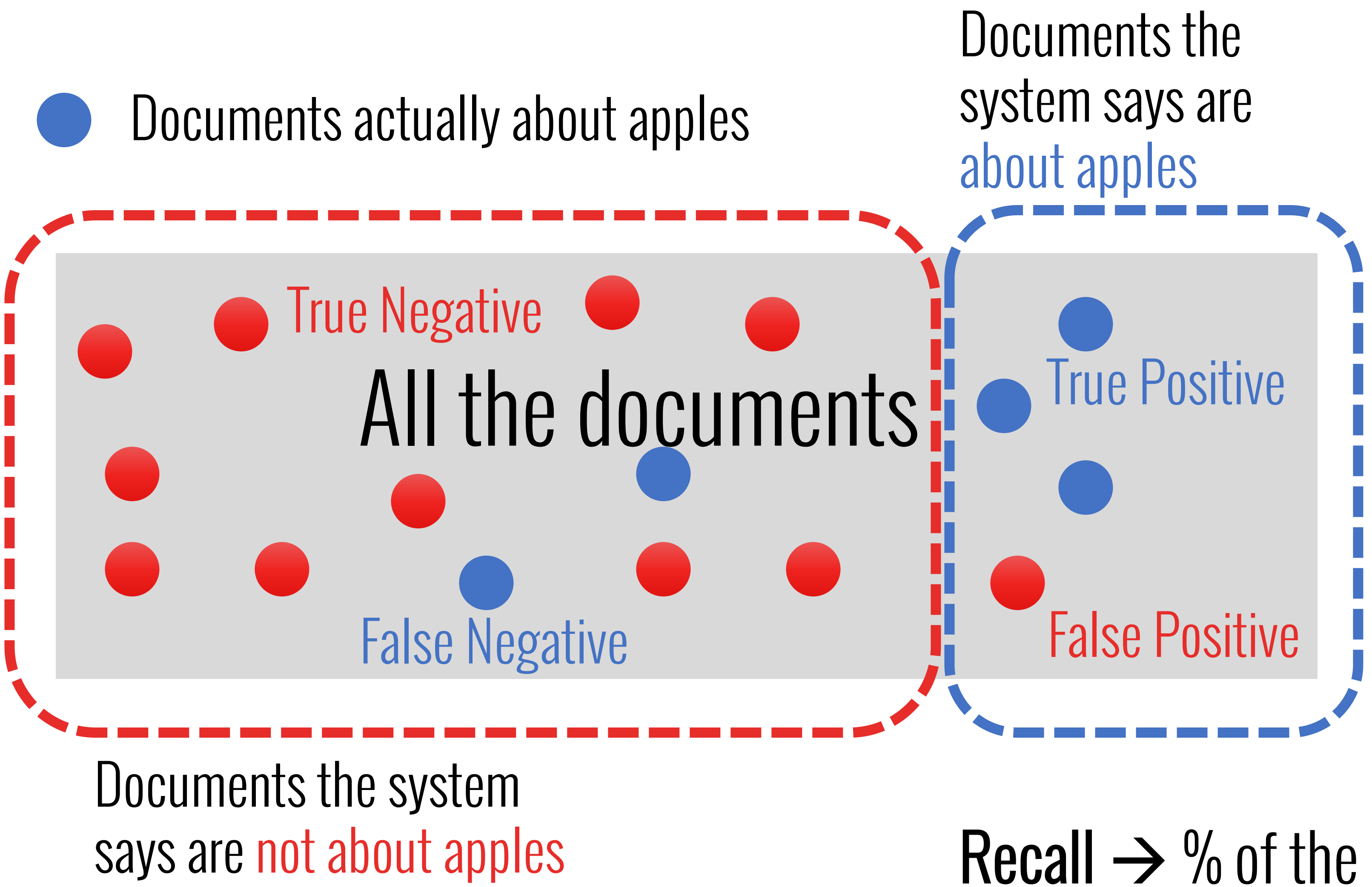
CONFUSION MATRIX (SIMPLE)

	Actually Positive	Actually Negative
Predicted Positive	True Positive	False Positive
Predicted Negative	False Negative	True Negative



Precision → % of the items returned that are true positive

TP / (TP + FP)



Recall → % of the positive items returned

TP / (TP + FN)

Precision: TP / **Pr**edicted positive

Recall: TP / **Re**al positive

HOW MANY OF THE POSITIVE LABELS WERE CORRECT?

HOW MANY OF THE REAL POSITIVES DID WE LABEL?

F1 SCORE

Annoying to deal with two measures

Why not just combine them into one?

F1 Measure: weighted average of precision and recall:

$$F1 = 2 * (\text{Recall} * \text{Precision})/(\text{Recall} + \text{Precision})$$

Good if you think false positives and false negatives are **relatively the same "badness"**

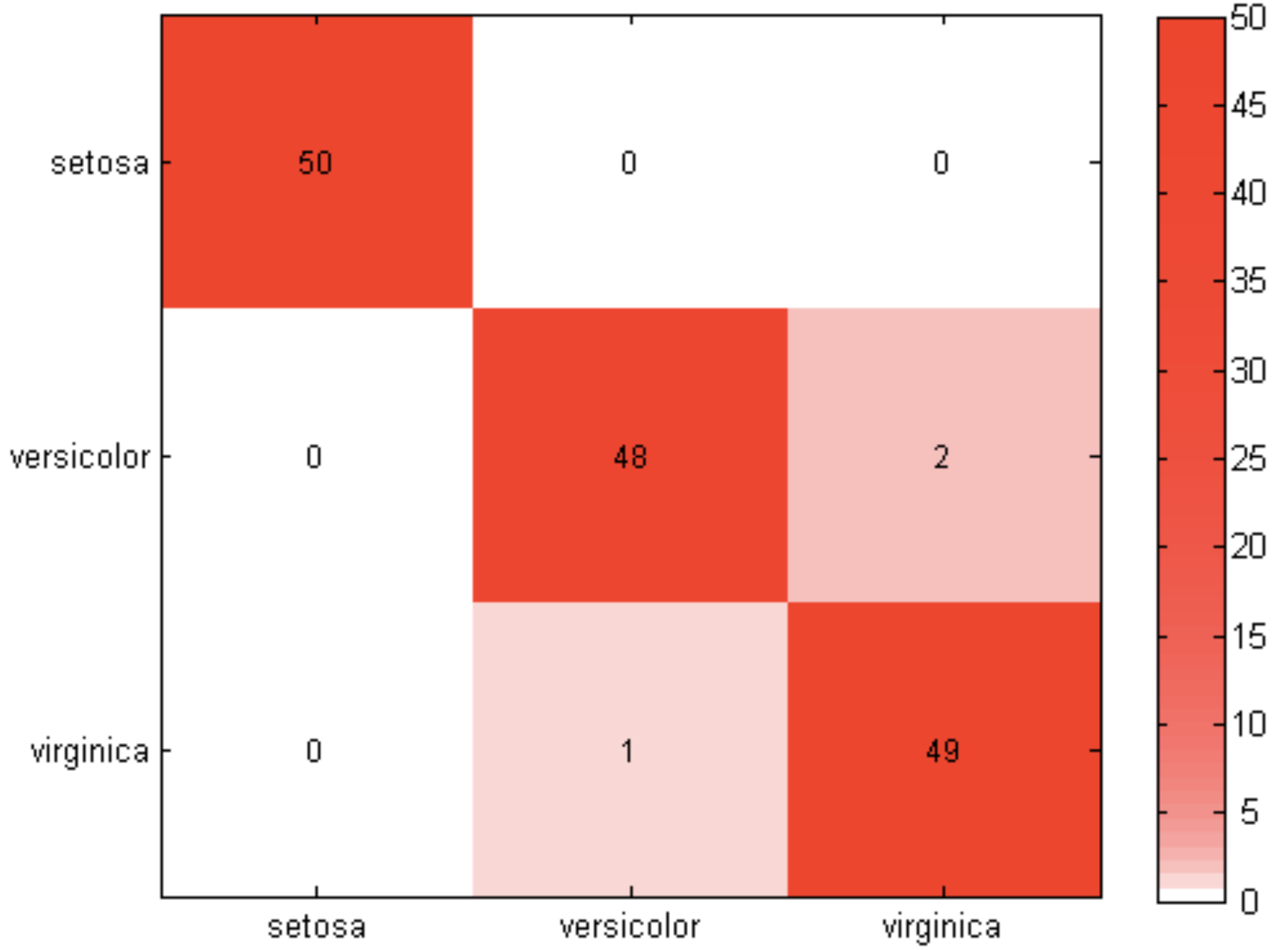
FOR MULTI-CLASSIFIERS

If instead of two classes **(positive / negative)** we have multiple classes **(“spam”/ “high priority” / “normal”)**, we can compute precision and recall in several ways.

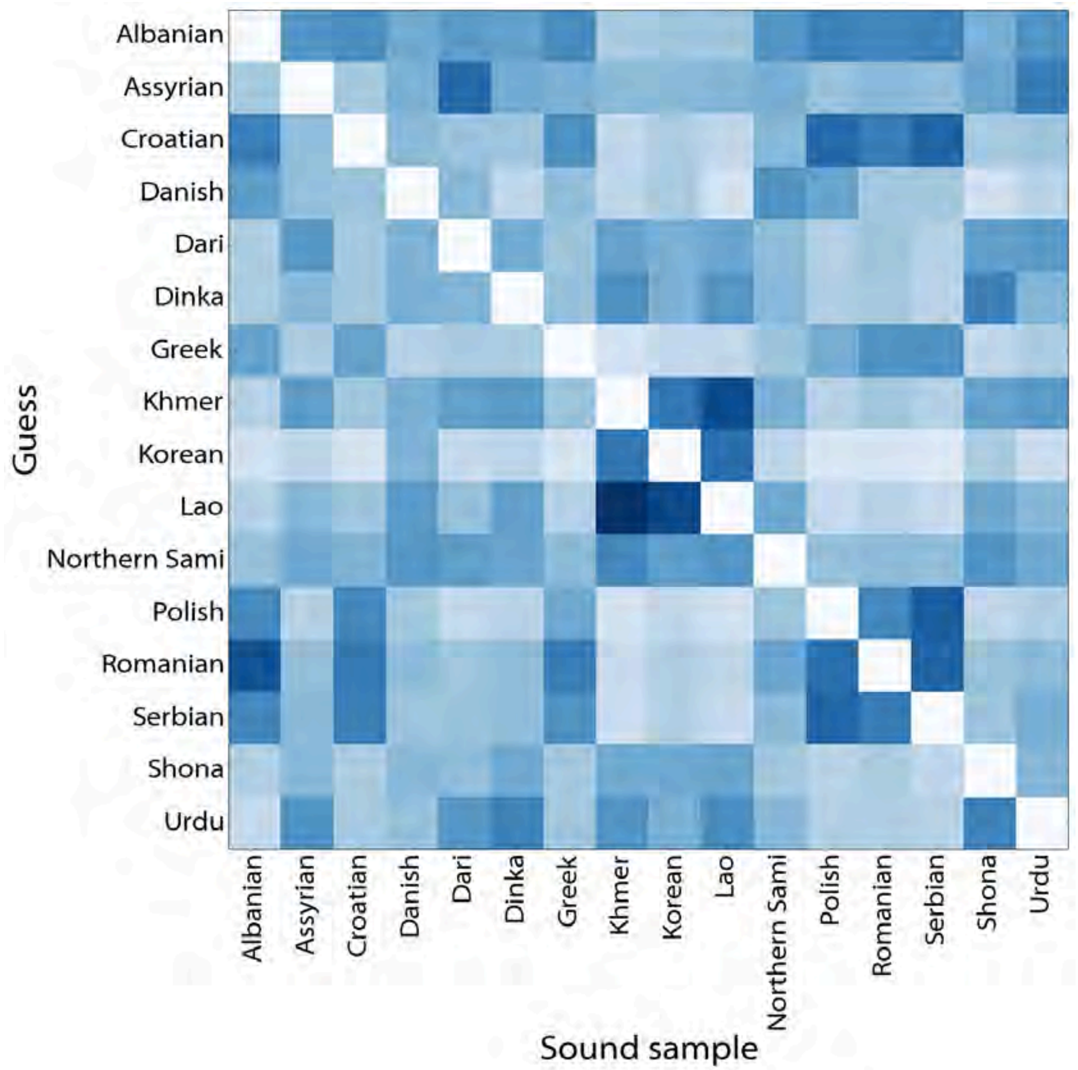
One option – take class one at a time:

- **“spam”** = positive, **all other classes** = negative
- **“high priority”**= positive, **all other classes** = negative
- **“normal”**= positive, **all other classes** = negative
- Calculate *mean* Precision/Recall

CONFUSION MATRIX (MULTI-CLASS)



CONFUSION MATRIX (MULTI-CLASS)



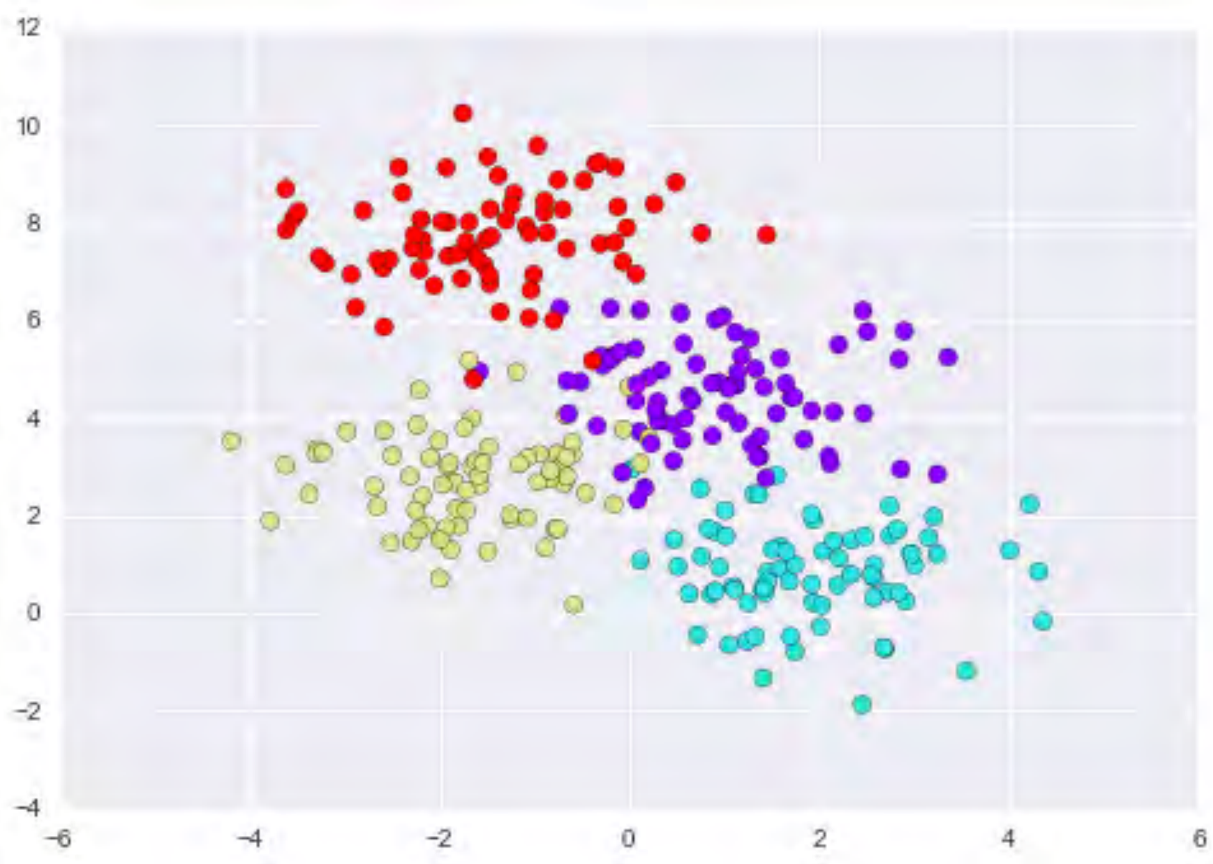
http://quietlyamused.org/blog/2014/03/12/language-confusion/

BACK TO THE NOTEBOOK!

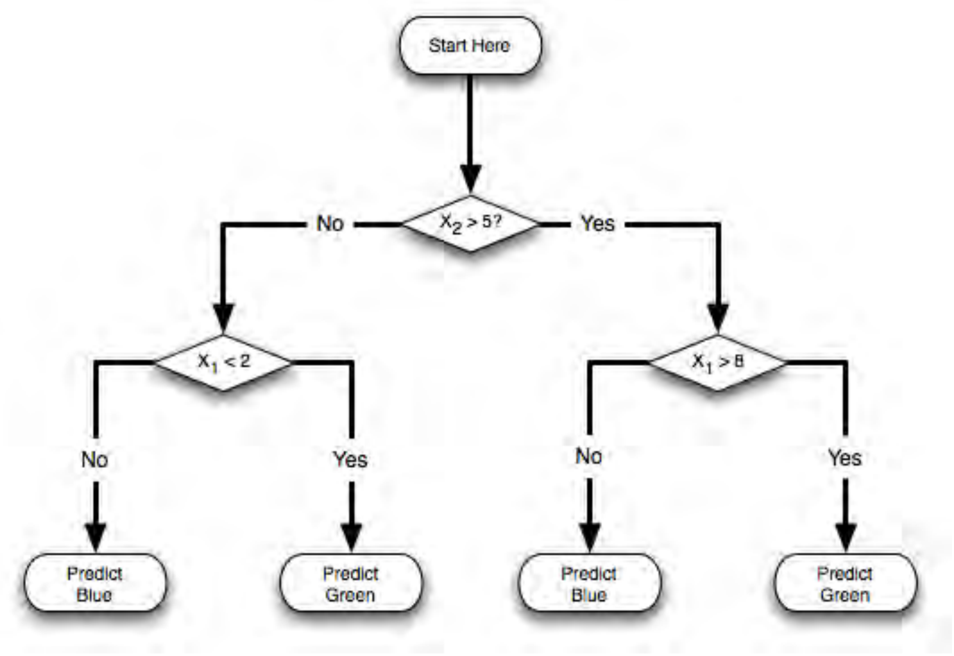
DECISION TREES

Classification as sets of (usually) binary decisions based on data attributes.

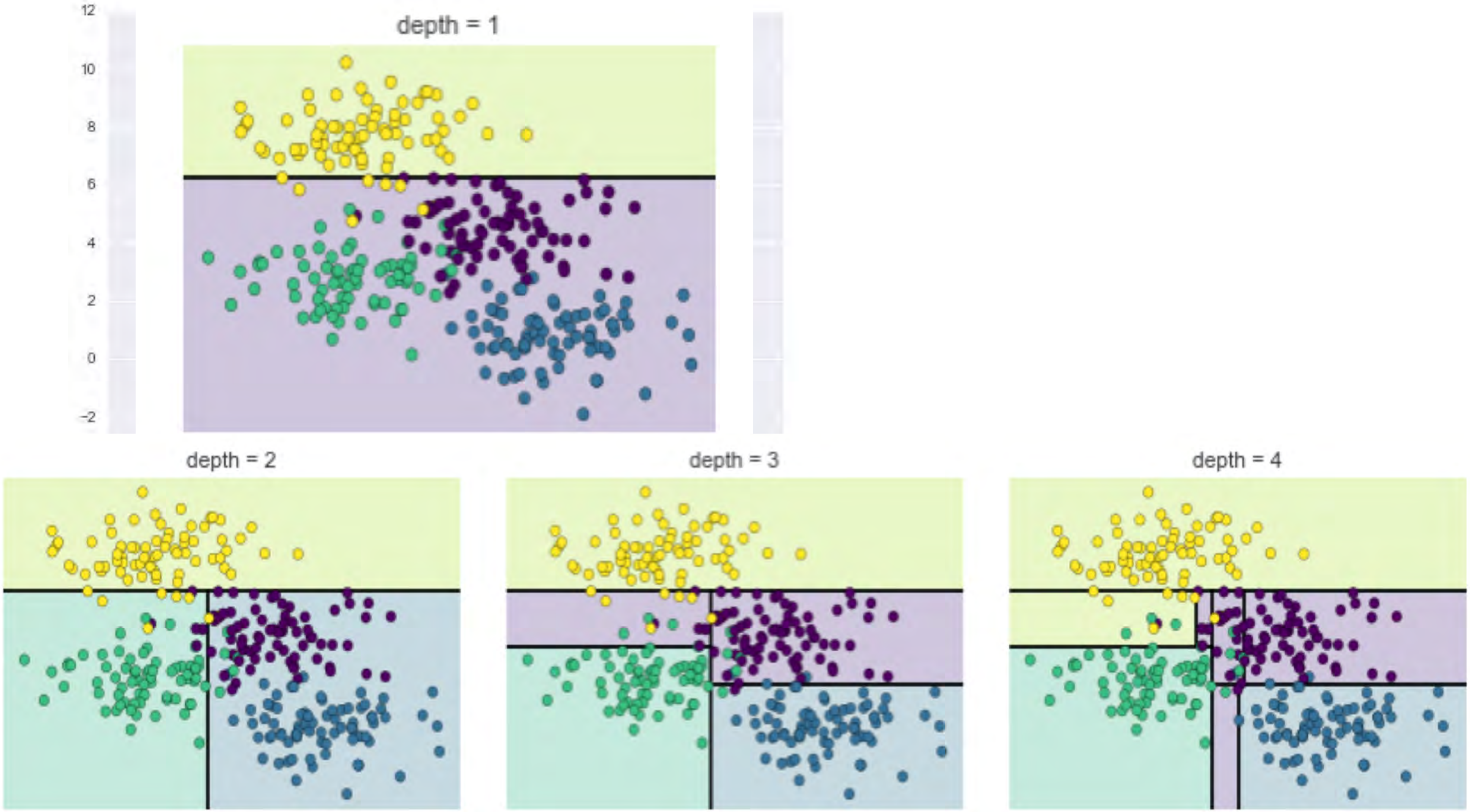
SHOULD I DO LAUNDRY?



REALLY USEFUL IF YOU CAN CARVE UP THE ATTRIBUTE SPACE USING A SET OF **IF-THEN** RULES



[Jake VanderPlas](#)


depth = 1
12
10
8
6
4
2
0
−2
depth = 2
depth = 3
depth = 4

DECISION TREES

Advantages:

Easy to interpret (not all classifiers can be explained)

Prediction process obvious

Can handle mixed data types

INTUITION FOR CONSTRUCTING DTs

Ask the question with the most valuable answer

If I knew the answer to this, **how much closer to the solution would I be?**

Solutions that divide the **space 50/50** are
better than solutions that divide the **space 2/98.**

MEASURING "INFORMATION GAIN"

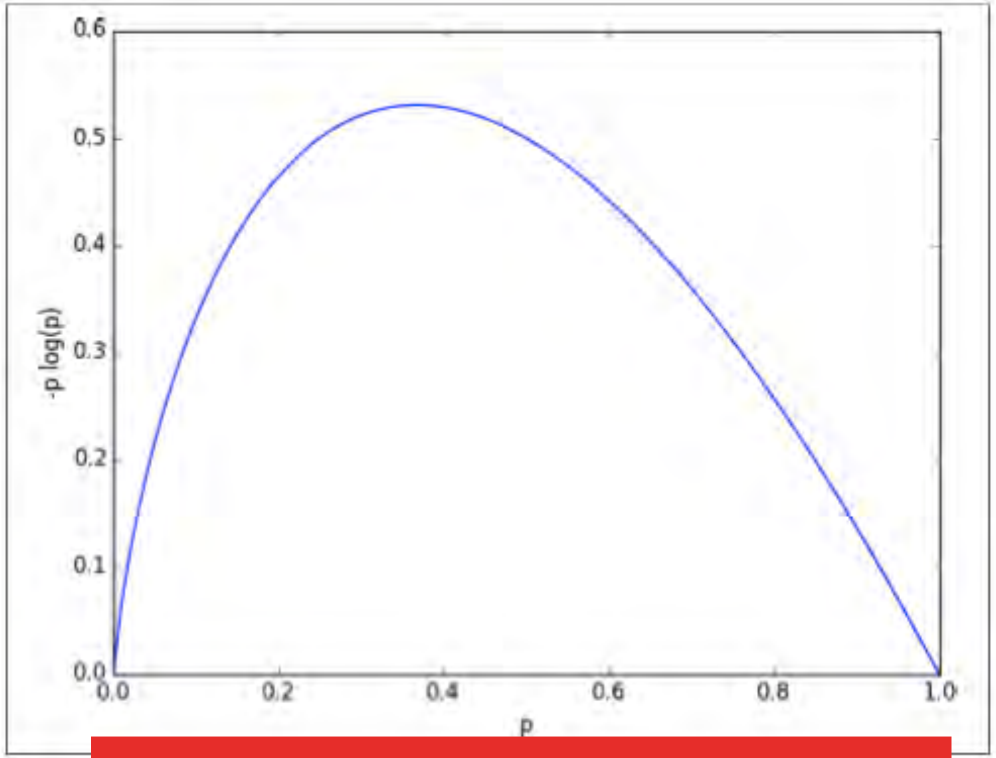
Use **entropy** or **purity** metrics
to assess information gain.

For a set of items S with J classes

Entropy:

$$H(S) = -\sum_{i=1}^{J} p_i log_2 p_i$$

p_i = fraction of items in the set with that class



Entropy =
0 for a homogeneous sample
1 for an equally divided one

TO ASSESS THE INFORMATION GAIN OF A SPLIT

$$Gain(T,a) = \underbrace{H(T)}^{\text{Entropy of Tree}} - \underbrace{H(T,a)}^{\text{Sum of Entropy of Children after Splitting on Attribute } \mathbf{a}}$$

Gain will be **high** if a split produces **pure subtrees.**

Can **compute information gain** for all possible splits and then choose the one with the **greatest gain.**

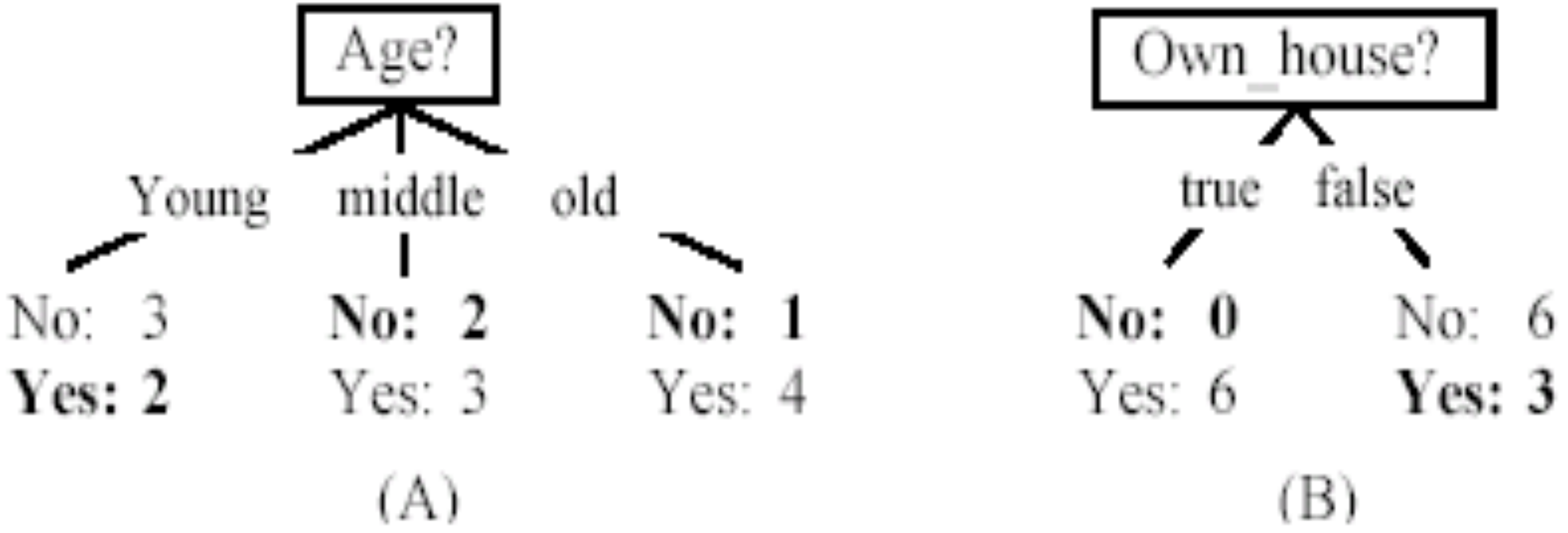
FOR EXAMPLE… HOUSING LOAN DATA

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

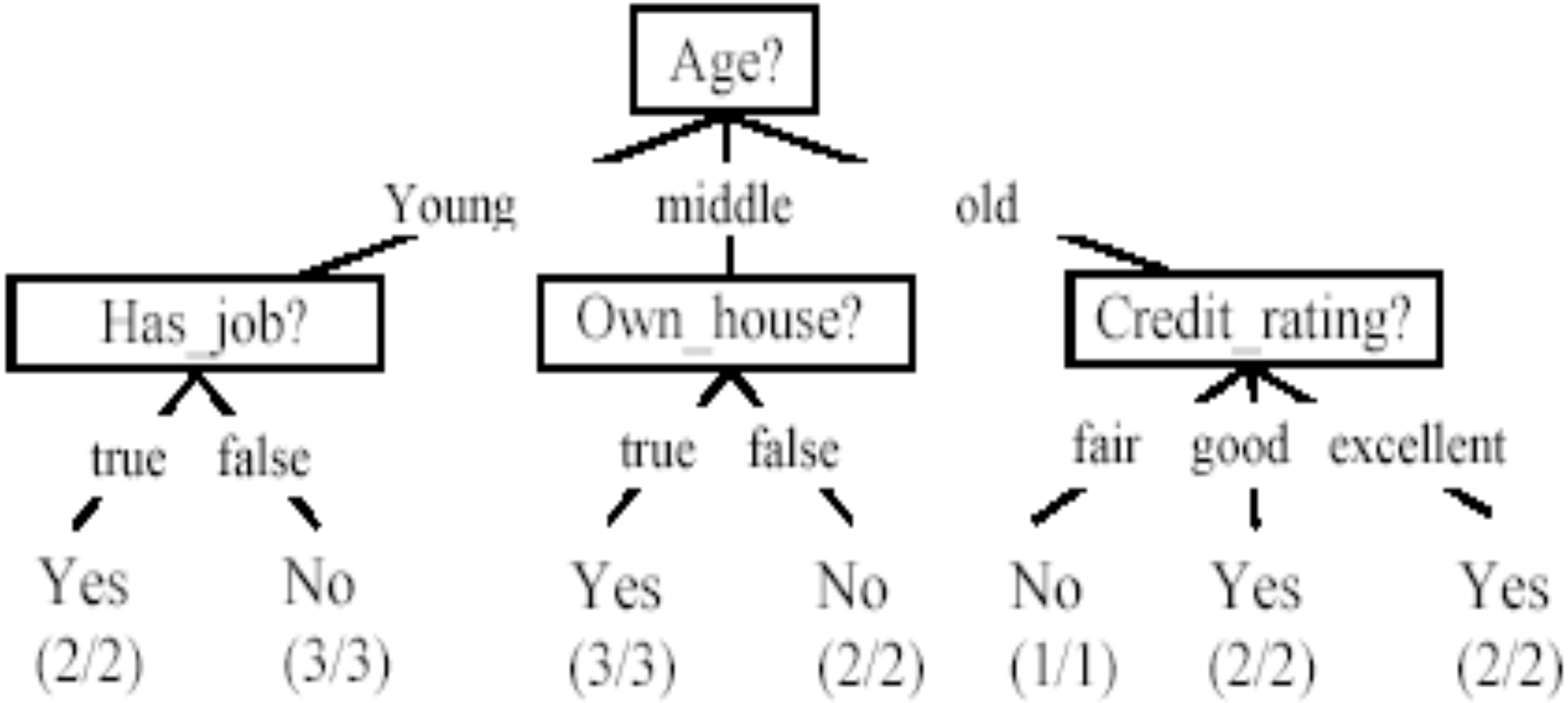
A RECURSIVE ALGORITHM

1. **Split** on the best feature (lowest partition entropy)
2. **Add** a decision node
3. **Repeat** (recursively) with each group of children
4. **Stop** if we hit an entropy threshold or partitions get too small

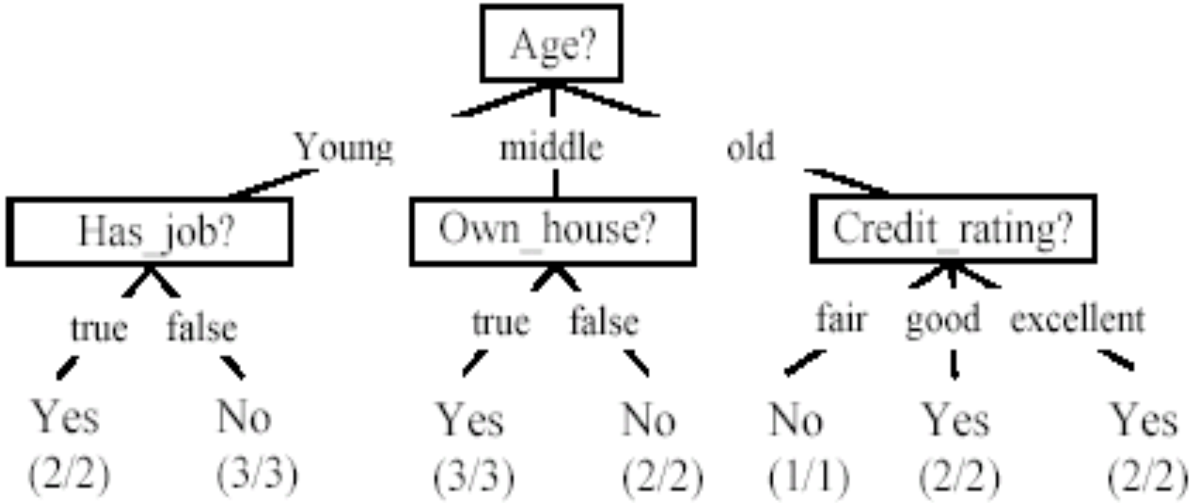


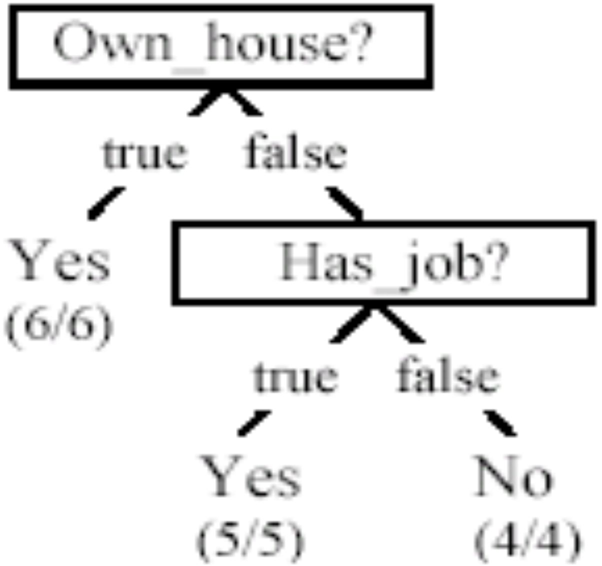
A DECISION TREE FOR THIS DATA

Decision nodes and leaf nodes (classes)



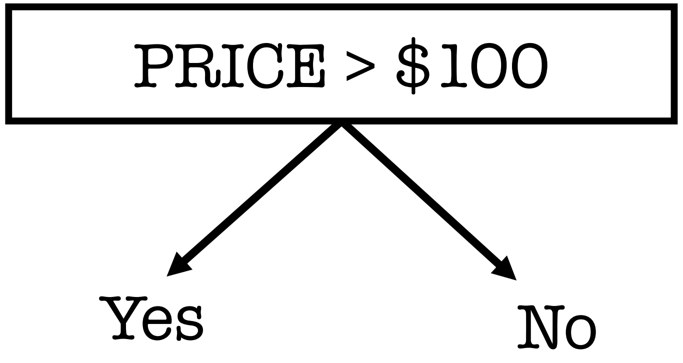
MULTIPLE VALID TREES ARE POSSIBLE





HANDLING CONTINUOUS ATTRIBUTES

Split into two (or more) intervals.

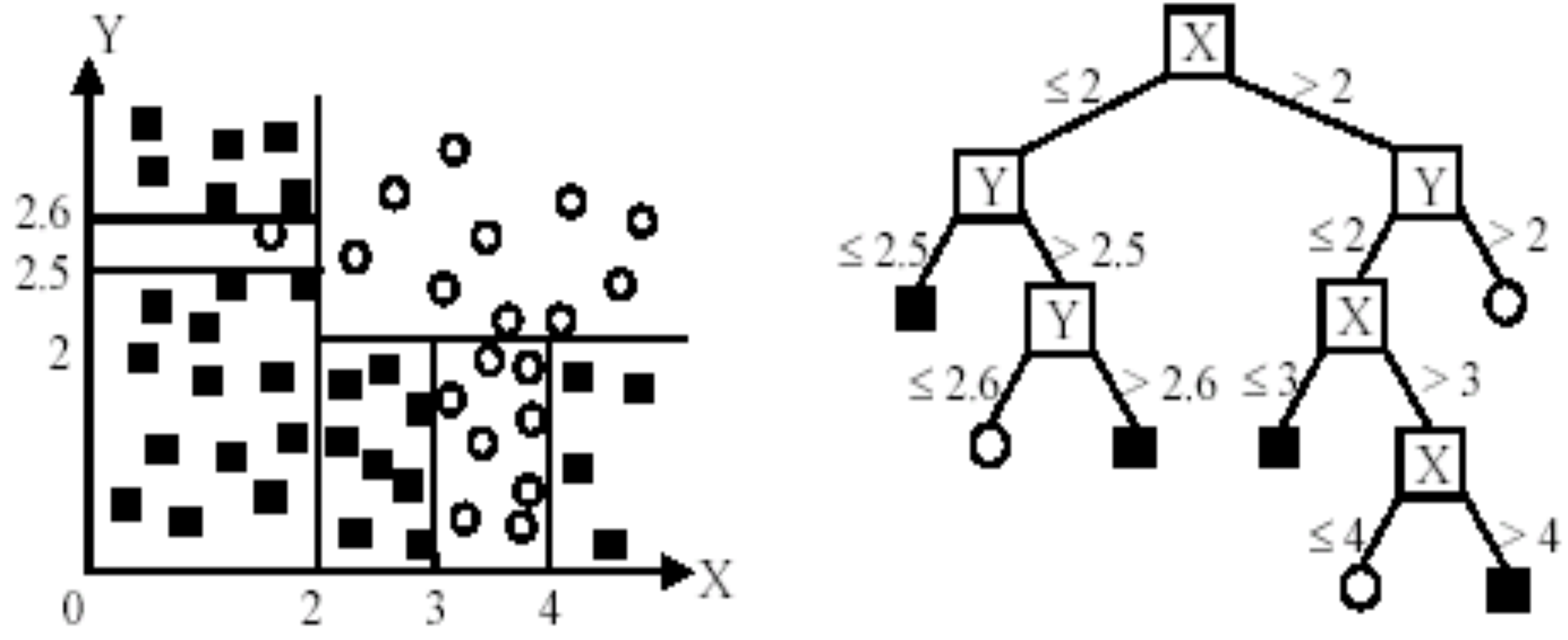


How to find the best threshold to divide?

- Use **information gain** or **gain ratio** again
- **Sort the values** in increasing order $\{v_1, v_2, \ldots v_r\}$,
- Consider **possible thresholds** between adjacent values v_i and v_{i+1}.
- Test possible thresholds and choose one that **maximizes the gain.**

From: CS583, Bing Liu, UIC

AN EXAMPLE IN A CONTINUOUS SPACE



(A) A partition of the data space

(B). The decision tree

From: CS583, Bing Liu, UIC

MANY FORMS OF DECISION TREES

By **Ross Quinlan**
- Most similar to what we just saw

ID3 – Basic entropy-based decision trees (discrete only)

C4.5 – Handles continuous and discrete attributes

C5.0 – Enhanced version of C4.5

sk-learn supports

CART (Classification And Regression Trees)
– Similar to C4.5 but uses "Gini Index" for purity instead of entropy

```
from sklearn.tree import DecisionTreeClassifier
```

And others … CHAID/MARS/etc.

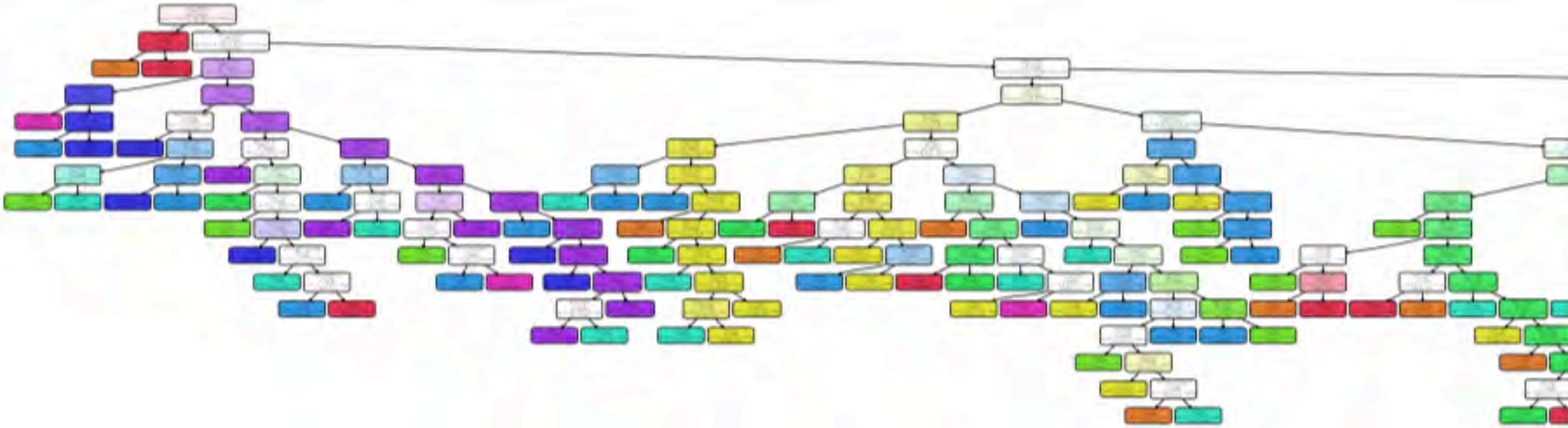
DECISION TREE

Advantages:

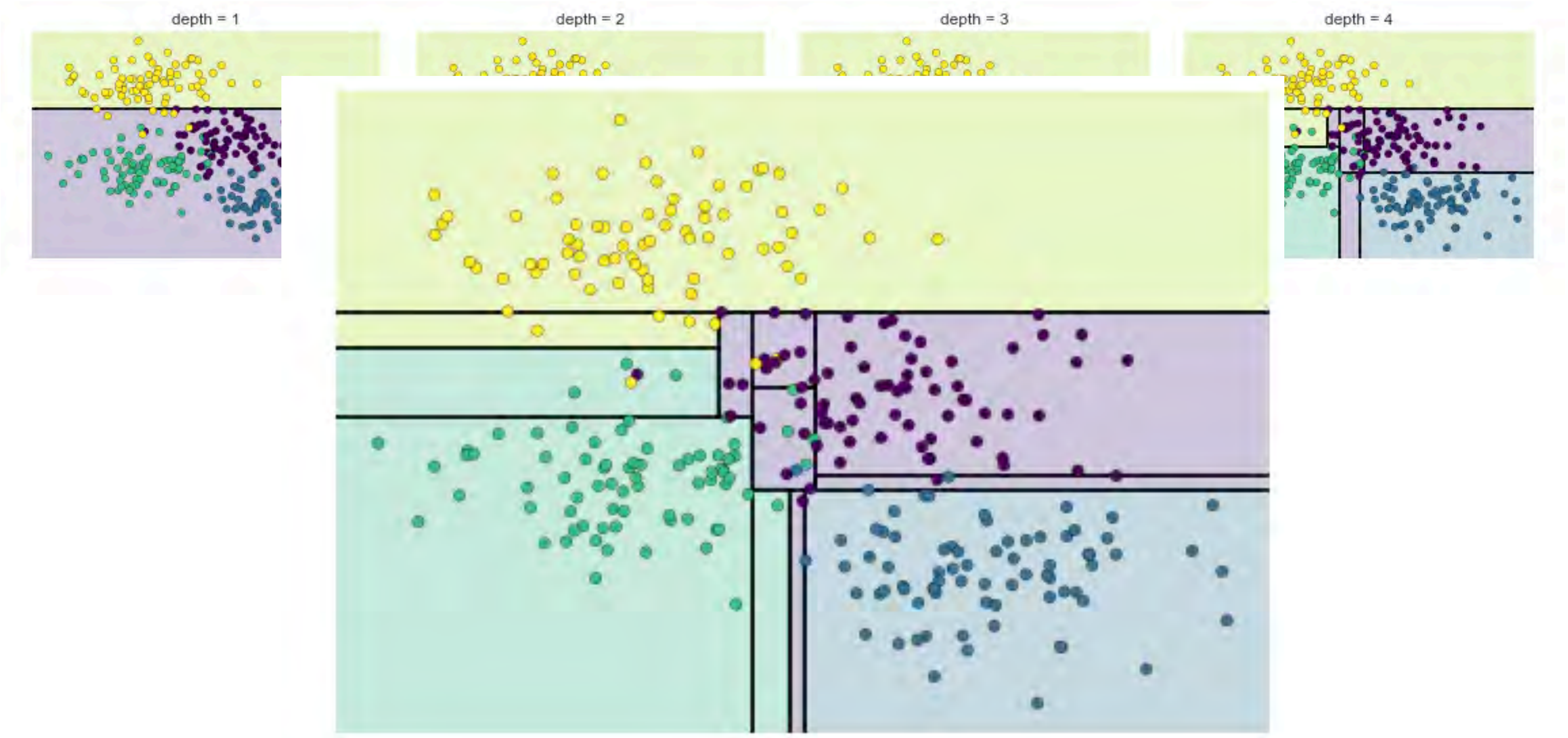
Easy to interpret – not all classifiers can be "explained"
Prediction process obvious
Handle mixed data types

Disadvantages:

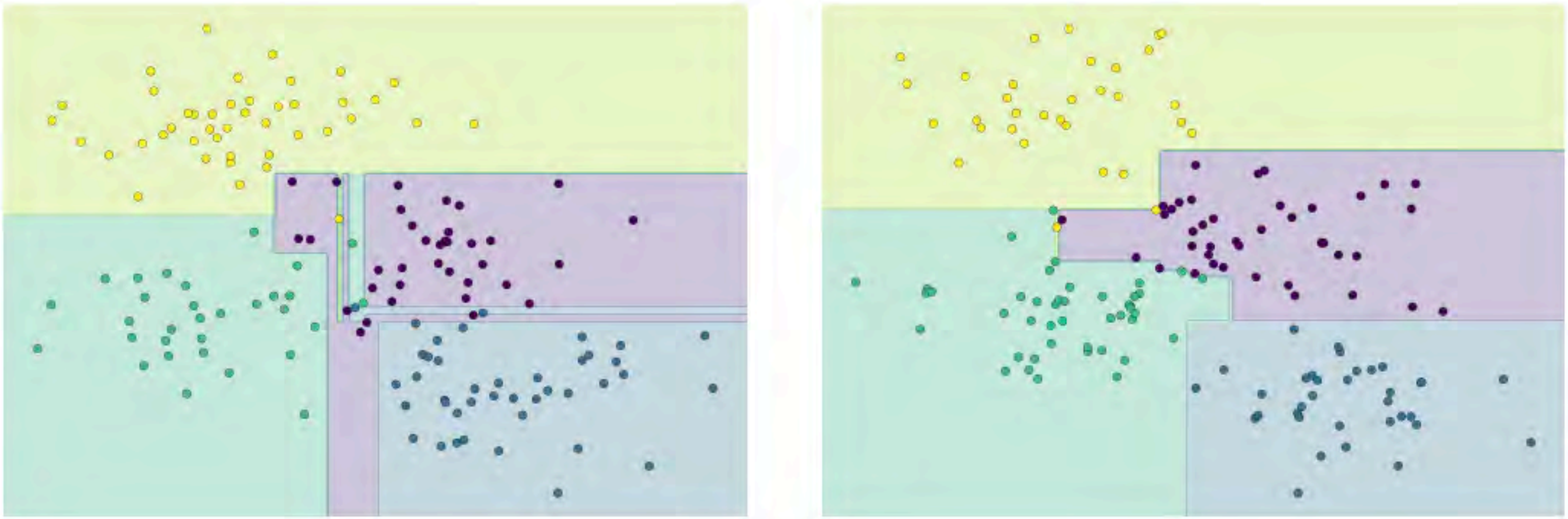
Expensive to calculate
Tendency to overfit
Can get large



RISK OF OVERFITTING



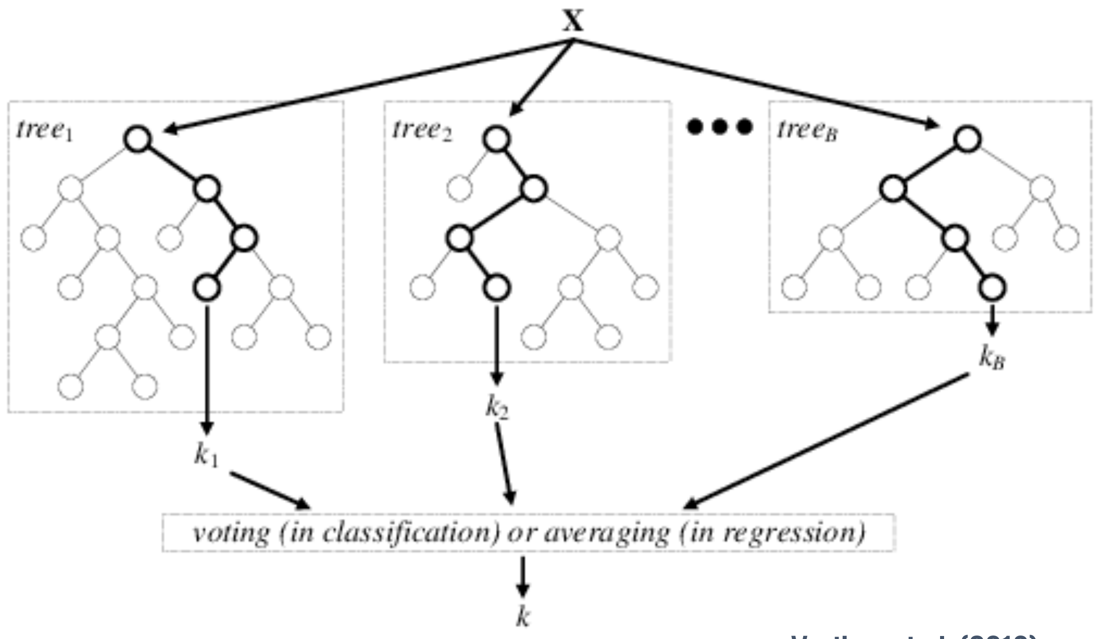
Different **training samples** can give really different results.



RANDOM FOREST

“Ensemble” classifiers

Create many trees and **have them vote.**



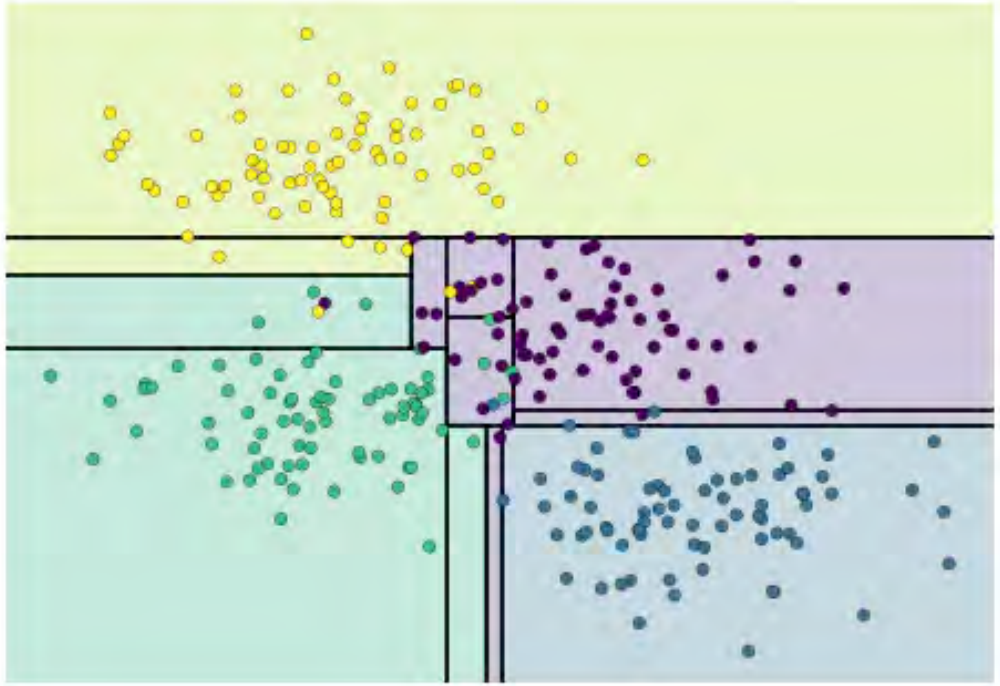
Verikas et al. (2016)

Problem: how to generate many trees from one dataset?

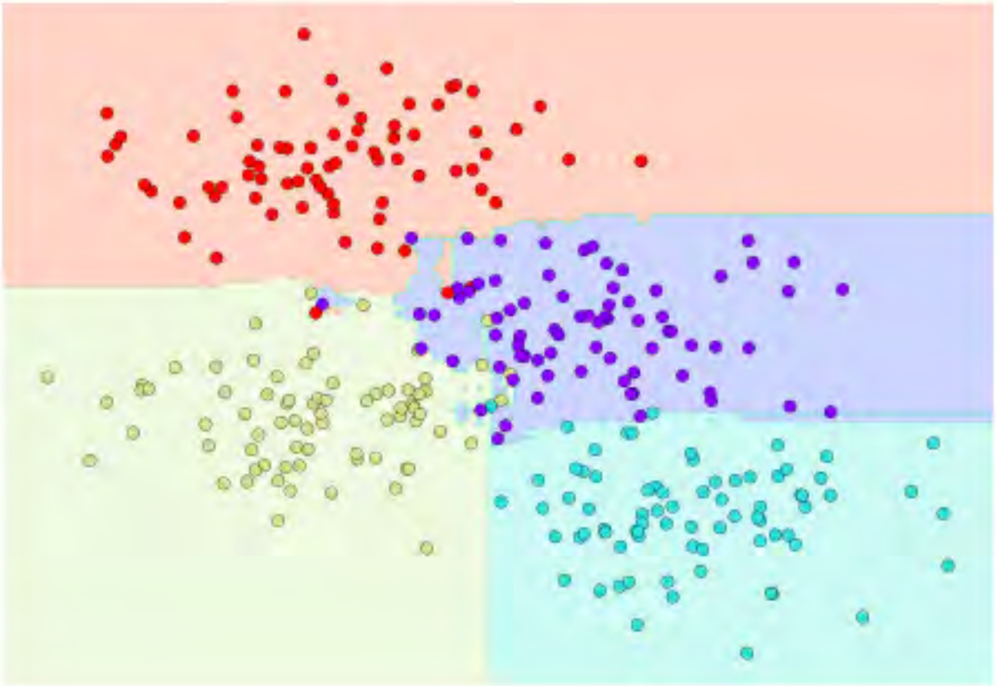
- Various way of randomizing
- Pick different data subsets (often using bootstrapping “with replacement”)
- Pick different features

```
from sklearn.ensemble import RandomForestClassifier
```

```python
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=100, random_state=0)
```



Single Decision Tree



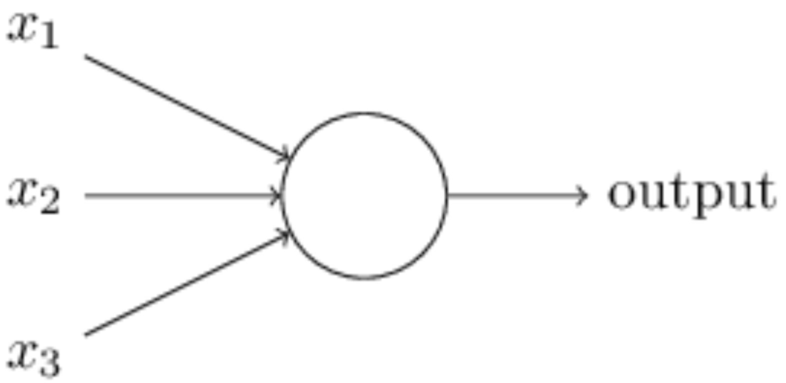
Random Forest with 100 Trees

NEURAL NETS

(and “Deep Learning”)

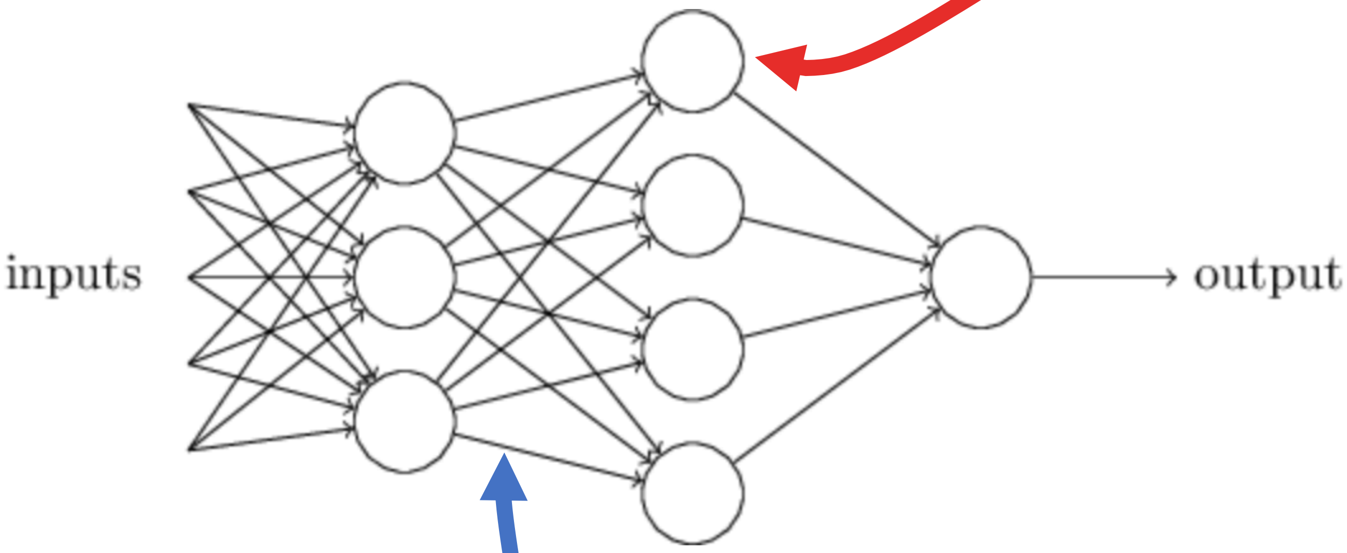
MIMICING LEARNING IN THE BRAIN

"Neurons"



EACH NEURON OUTPUTS A **WEIGHTED SUM** OF ALL OUTPUTS FROM THE PREVIOUS LAYER

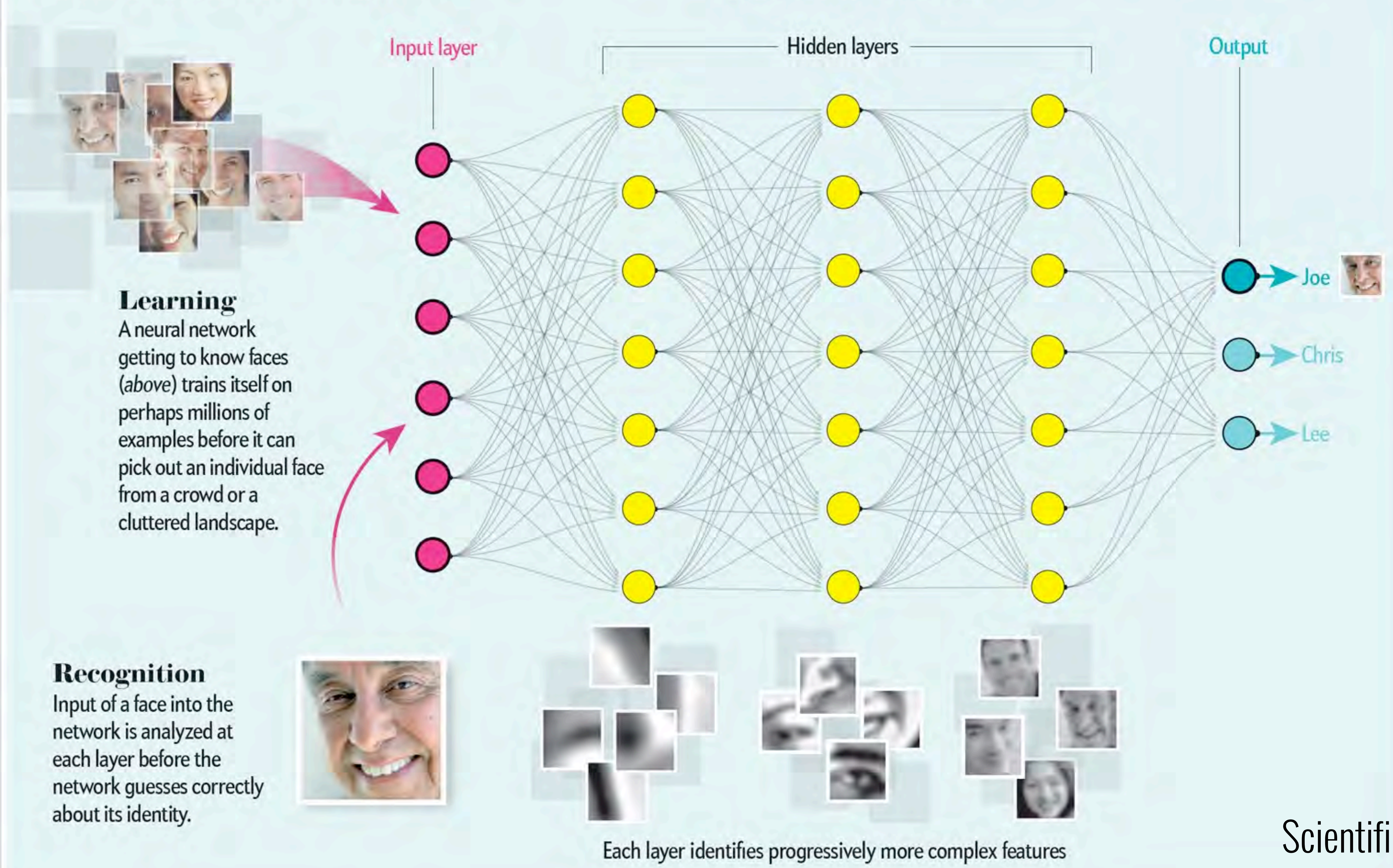
Multiple layers with weighted "**feed-forward**" connections



Learning via **back-propagation**

EVERY EDGE HAS A UNIQUE **WEIGHT** THAT CAN CHANGE OVER TIME

[Michael Nielsen](#)



Scientific American

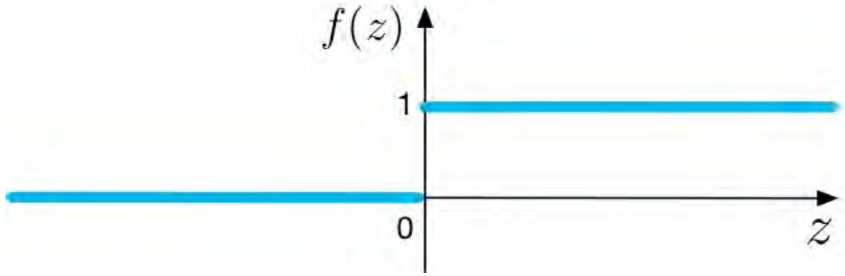
LEARNING PREDICTABLY

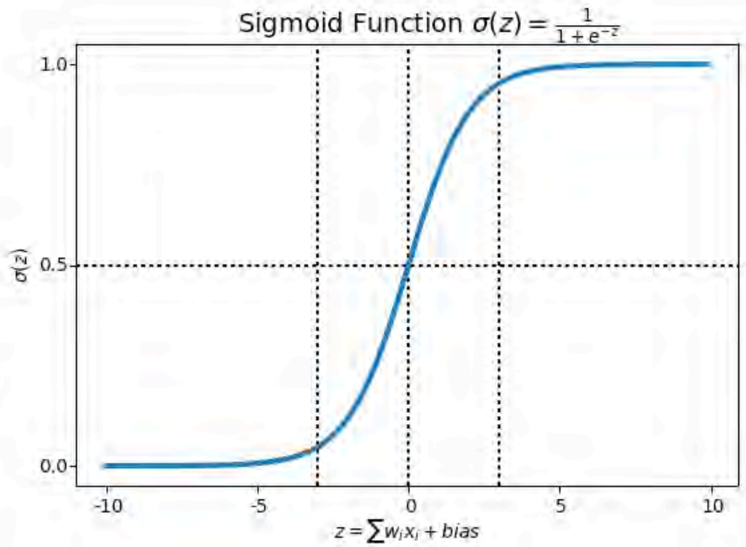
Smooth activation functions

Sigmoid, Tanh, etc. (not stepwise)

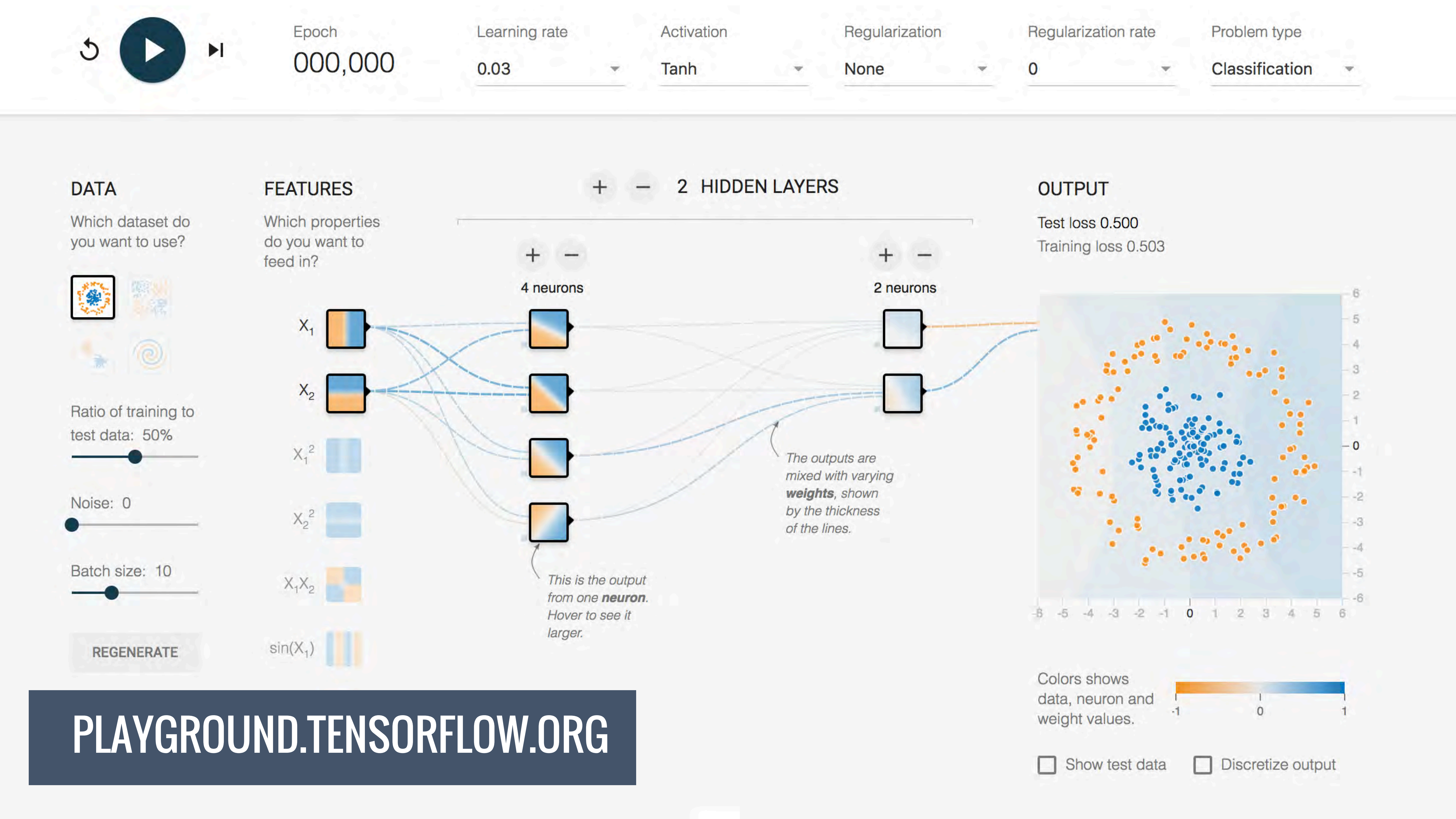
Randomize weights initially

Adjust weights slowly





Nahua Kang



PLAYGROUND.TENSORFLOW.ORG

NEURAL NETS IN PYTHON

Supported in SciKitLearn (but not very scalable)

```
from sklearn.neural_network import MLPClassifier
```

A bunch of other libraries…

- **tensorflow**
- pylearn2
- sklearn_theano
- nolearn
- keras
- lasagne
- …

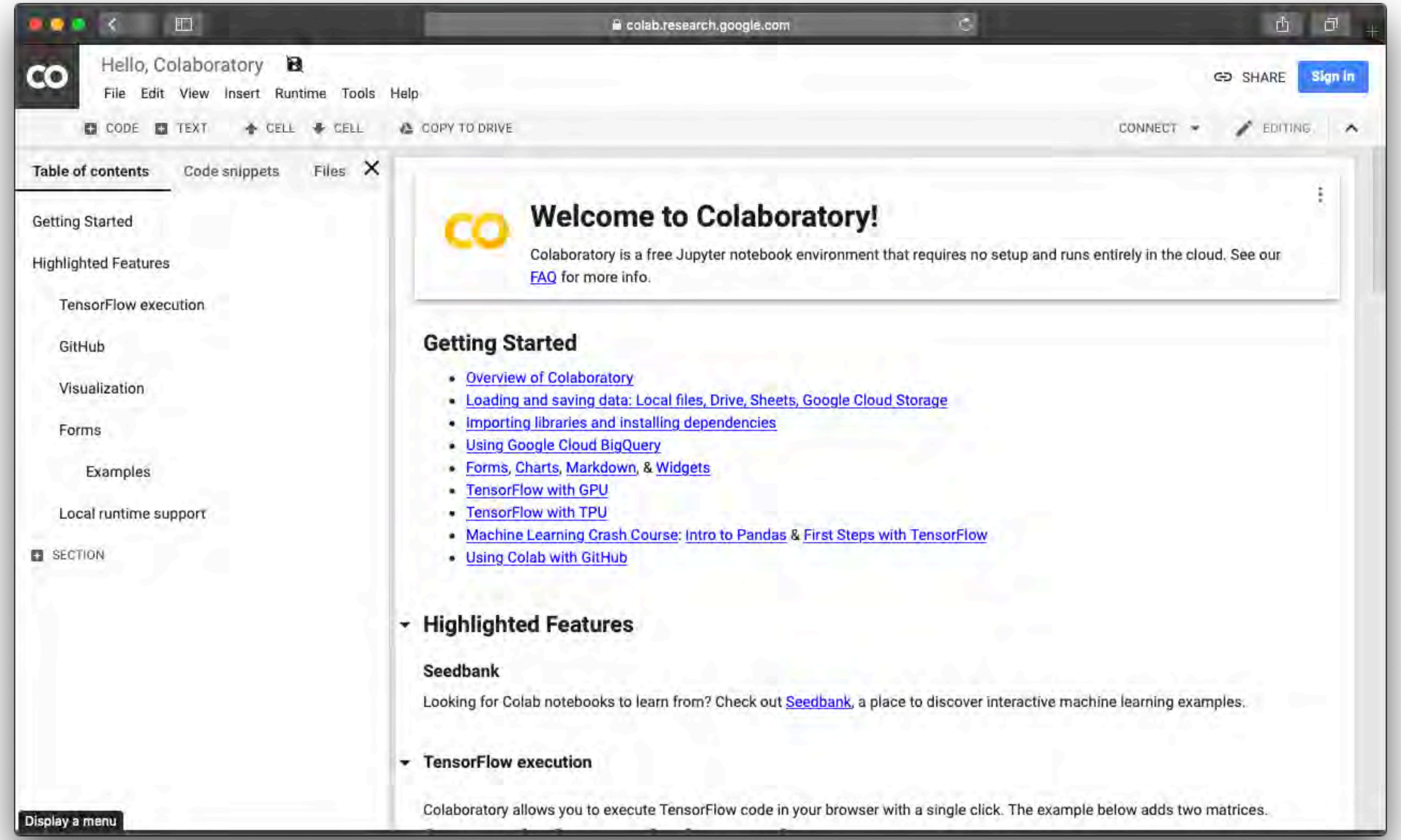
TensorFlow

TensorFlow is a fast and flexible open-source ML library for research and production.

colab

TensorFlow + **Colaboratory**

If you're going to start really diving into ML – this is a good place to start.



BUT… NEURAL NETS ARE BASICALLY BLACK BOXES

To perform well, NNs usually require
large numbers of nodes, multiple hidden layers, and **lots of edges!**

Internal behavior can be **really difficult to understand.**

Especially true with "**deep learning**"
(5, 10, or even more hidden layers).

OTHER APPROACHES

Logistic Regression

- **Multiple linear regression** extended to support **categorical outputs** instead of just quantitative ones.

```
from sklearn.linear_model import LogisticRegression
```

Support Vector Machines (SVM)

- Find a **hyperplane in multidimensional space** that best splits items with a label from items without it.

```
from sklearn.linear_model import LogisticRegression
```

ROLL-YOUR-OWN ENSEMBLE METHOD

Use a **VotingClassifier** to combine multiple classifiers.

```
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier

clf1 = LogisticRegression()
clf2 = RandomForestClassifier()
clf3 = GaussianNB()
eclf = VotingClassifier(estimators=[('lr', clf1), ('rf', clf2), ('gnb', clf3)])
```

Can also weight models, tweak voting schemes, etc.

SUMMARY

Classification

Useful when we know something about the structure

Use a few labeled examples to classify many more

If you're interested – **start playing**
or **take a machine learning course (like DATA 607)!**

A NICE STARTING POINT

A nice, recent (Sept 2019) Machine Learning reference built around Python examples.

Available online via the UofC.

https://learning.oreilly.com/library/view/hands-on-machine-learning/9781492032632/