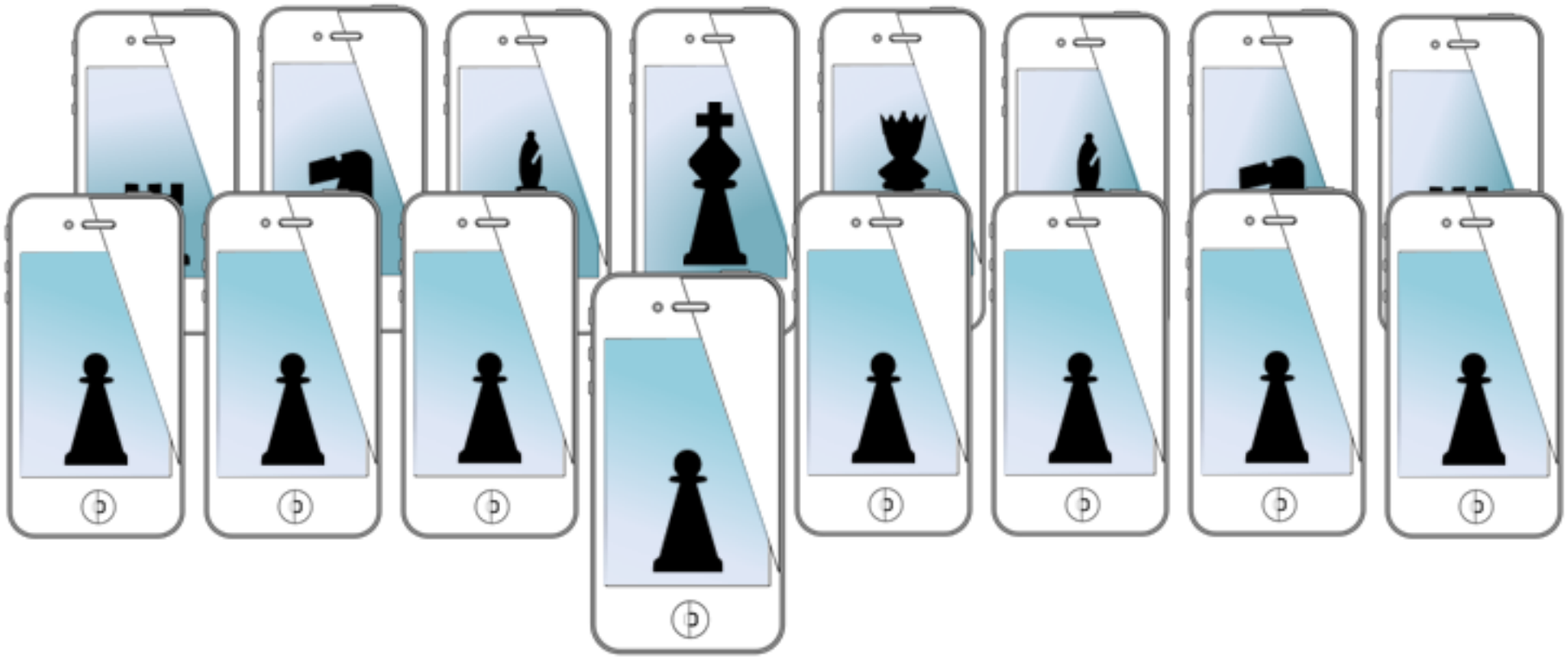
MOBILE SENSING LEARNING



CS5323 & 7323

Mobile Sensing and Learning

machine learning crash course

Eric C. Larson, Lyle School of Engineering,
Computer Science, Southern Methodist University

course logistics

- **A4** is due this week
- **A5** is due in two weeks
  - you can run server from your laptop (or cloud)
  - make it a first draft of your final project
    - seriously—make things easier on yourself
- **final project proposal** is also due at same time
  - feel free talk to me about it

assignment 5

- try to make this the first iteration of the final project!

Assignment Five - Machine Learning as a Service

Module A

Create an iOS application using the HTTPSwiftExample that:

- Collects some form of **low throughput (sampling rate > 1s)** feature data for processing: audio, video, motion, or from the micro-controller
- Uploads **labeled feature data** to a server via HTTP POST requests
  - you can run the server from your laptop or mac mini
  - Alternatively you can use a virtual machine, AWS, or other cloud service
- Trains a model from the labeled data (e.g., KNN, SVM, Random Forest, etc.)
- **Requests predictions** from the server by uploading unknown feature vectors
  - can be periodically or initiated by user
- Note that the server code given to you will automatically save any feature data you upload and train a machine learning model, given the correct POST/GET request commands

You should not need to update the server for any of the given functionality. However, the predictions from the server may not be sufficient without updating the training parameters or the type of model used. Verify the functionality of the application to the instructor during lab time or office hours (or scheduled via email).

assignment 5

- try to make this the first iteration of the final project!

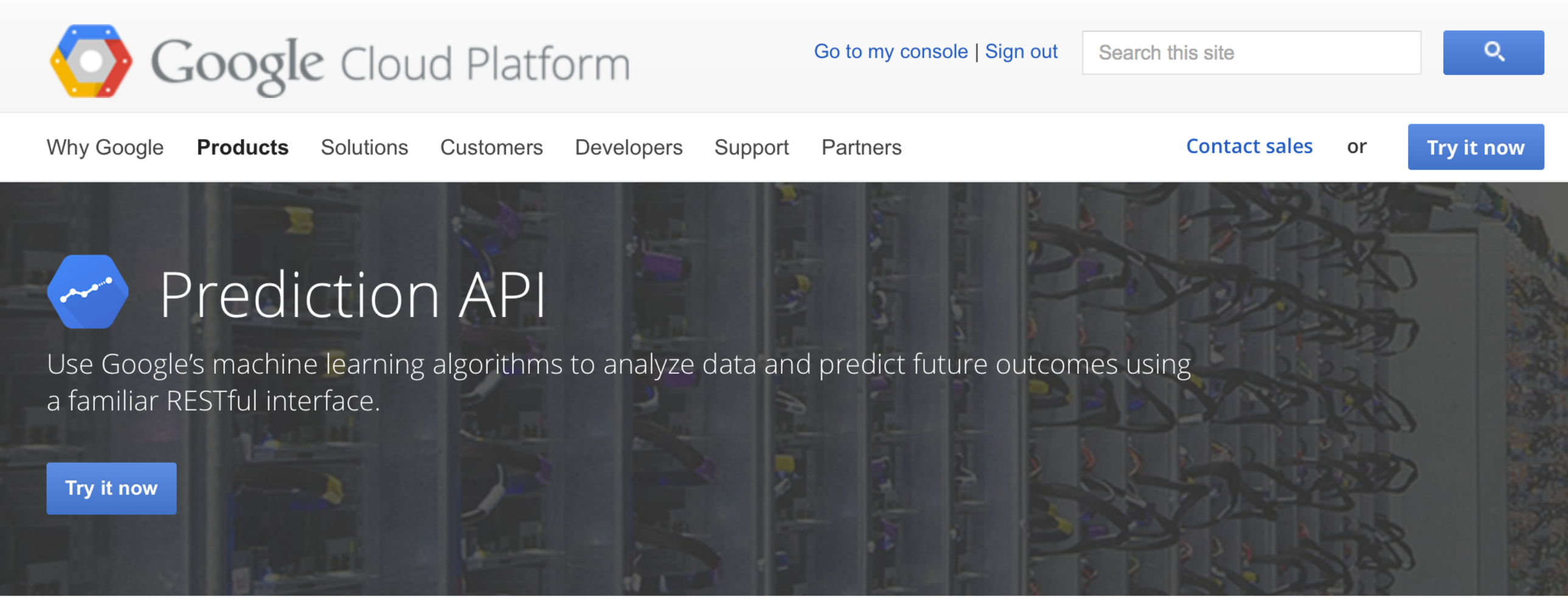
Assignment Five - Machine Learning as a Service

Module B

Update the HTTPExample and the tornado web server to:

- Specify the type of model to use in the Machine Learning
  - at least **two different types** of machine learning models (***e.g.***, SVM and KNN)
- Compare the efficacy of two or more different models
  - **send parameters** to use in the machine learning models from the phone (***e.g.***, number of neighbors to use in KNN)
- **Exceptional Work:** 7000 Level Students Choose ONE of the following:
  - make the training of the model non-blocking to the tornado IOLoop
    - (requires co-routines and manipulation of sklearn fit function)
  - implement **authentication** in tornado and in your iOS application
    - cookie secret is fine
    - or third party
  - Use CoreML to export your custom trained machine learning model and run the machine learning prediction locally on the iOS app (NOTE: the **CoreML model must be exported from the data you create** on your HTTPServer)

machine learning



machine learning

BigML

Set up a Source

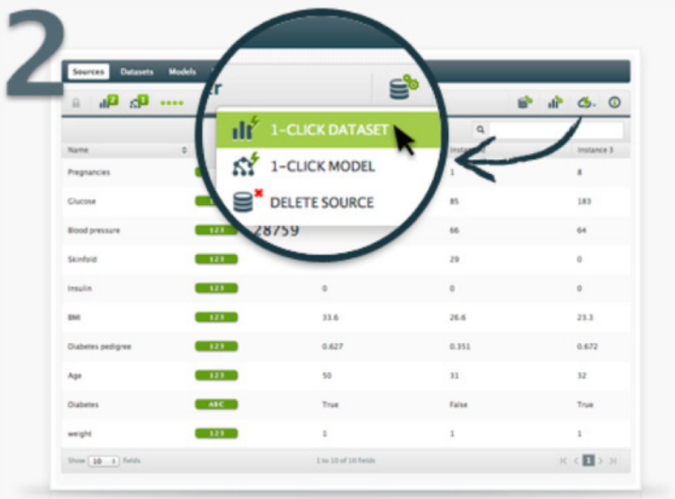
To create a new source, just **drag and drop** your data file onto BigML's interface, or select the file you want to use with the **upload** icon. Sources can be created from almost any tabular data (**csv** or **arff** files). You can gzip (**.gz**) or compress (**.bz2**) them to save bandwidth. You can also create sources from remote locations using protocols such as **HTTP(S)**, **s3**, **azure**, or **odata**. You can upload files of up to **64GB** or up to **5TB** if you use remote S3 buckets. Once your source has been created, you can use a configuration panel to update **types**, **names**, **labels**, **descriptions**, and other parsing preferences.

How to create a source

Create a Model

To create a model, just use the **1-cl**
if you want to use all active fields an
generate a model that predicts (i.e.,
column in the input data. You can al
panel to select a different **objective**
your dataset using multitude of opt
processed to build a predictive mod
the most relevant patterns in your d
generate predictions for new data in

How to create a model



Create a Dataset

To create a new dataset, just use the **1-click dataset** button from a source view if you want to include all the fields and the complete source, or use the **configure dataset** panel to select a few specific fields or limit the total size of data to analyze. BigML will start computing the distribution of values for each of the fields in your dataset. This process can take from a few seconds to a few hours depending on the size of your data. As your dataset is being created, BigML shows you a visualization that gives you immediate feedback about your data.

How to create a dataset

Generate Predictions

To create a prediction, you can eith
generated web form or a question b
click the **predict** or **predict questio**
presented with the corresponding i
of input fields is big or you want to a
predictions you can use our API. Sta
High Performance Prediction Server
generate thousands of predictions p

How to make predictions

agenda

- intro to machine learning
- numpy, scipy, and scikit-learn
    - using iOS with python:
        - applepy?
- by the end of this lesson I want you to know:
    - what is machine learning
    - know the keywords
    - know enough about ML toolkit to approach it
    - lose whatever barrier you had to ML

machine learning

ma·chine

/məˈSHēn/

noun
noun: **machine**; plural noun: **machines**

1. an apparatus using or applying mechanical power and having several parts, each with a definite function and together performing a particular task.
   "a fax machine"
   synonyms: apparatus, appliance, device, contraption, contrivance, mechanism, engine, gadget, tool More
   - a coin-operated dispenser.
     "a candy machine"
   - *technical*
     any device that transmits a force or directs its application.

learn·ing

/ˈlərniNG/

noun
noun: **learning**

1. the acquisition of knowledge or skills through experience, study, or by being taught.
   "these children experienced difficulties in learning"
   synonyms: study, studying, education, schooling, tuition, teaching, academic work; More
   - knowledge acquired through experience, study, or being taught.
     "I liked to parade my learning in front of my sisters"
     synonyms: scholarship, knowledge, education, erudition, intellect, enlightenment, illumination, edification, book learning, information, understanding, wisdom More
     antonyms: ignorance

teach a computer to learn something
through experiences

the hierarchy

artificial intelligence
a machine that acts intelligently

- terminator
- follows a set of rules
- play chess
- play a game

data mining
detect patterns for people

- create a model
- cluster data
- help me visualize

machine learning
teach from examples

- create a model
- predict something

features

- actually, feature vectors
- classic example: the iris dataset

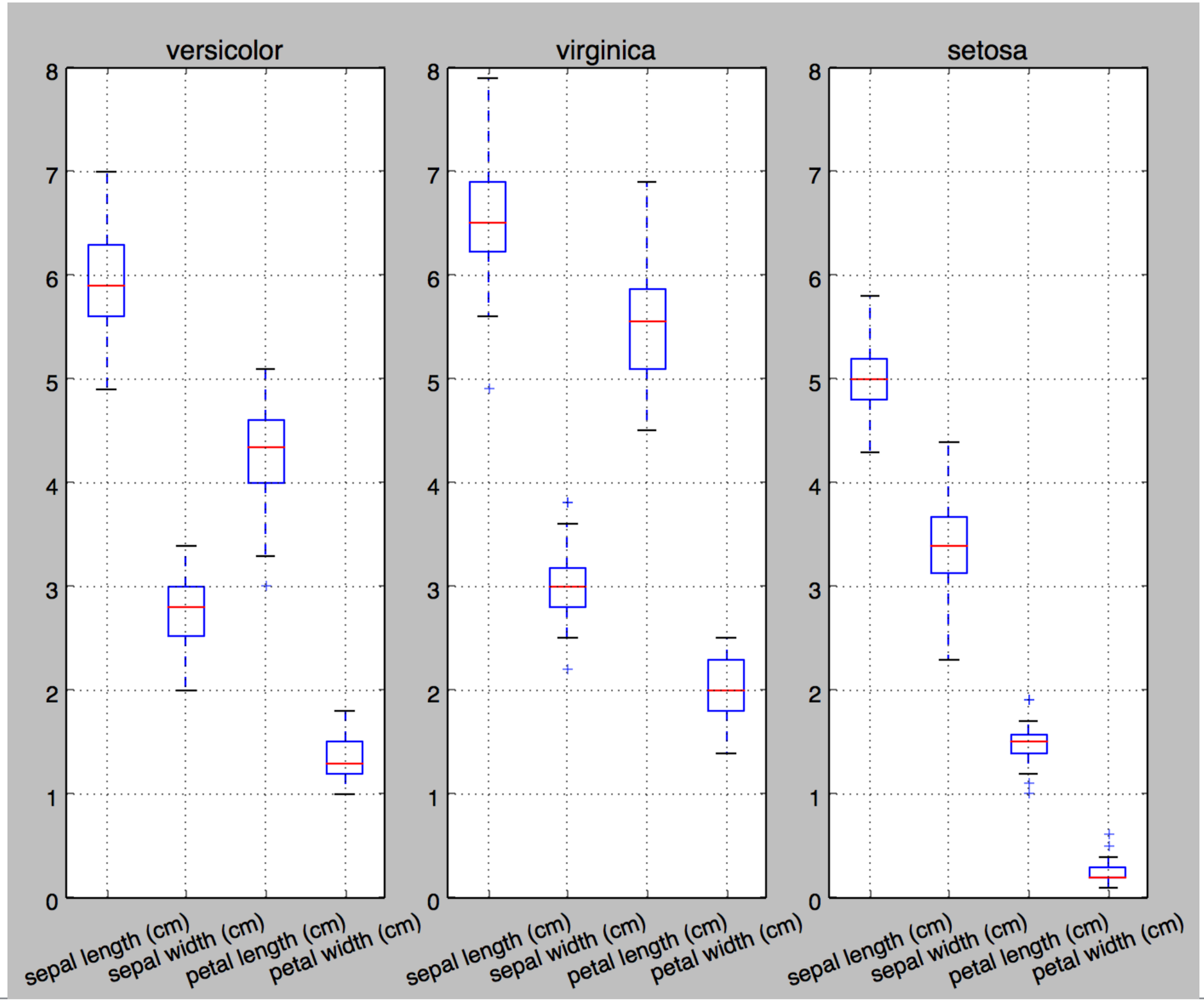
setosa versicolor virginica

- 4 features
  - sepal length in cm
  - sepal width in cm
  - petal length in cm
  - petal width in cm

```
[5.1,  3.5,  1.4,  0.2] setosa
[5.7,  2.8,  4.5,  1.3] versicolor
[7.6,  3. ,  6.6,  2.1] virginica
```

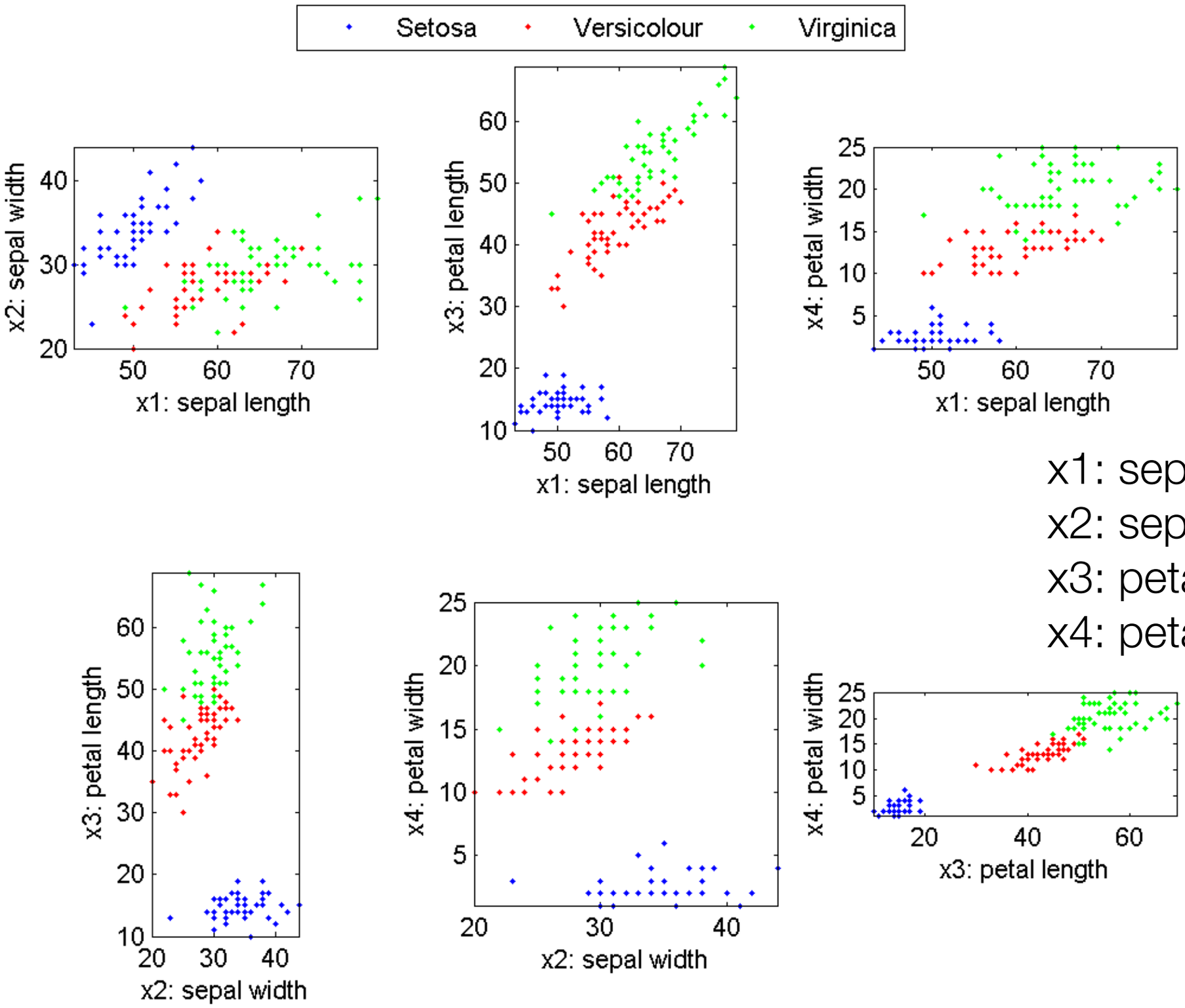
50 examples each

visualizing features



separability

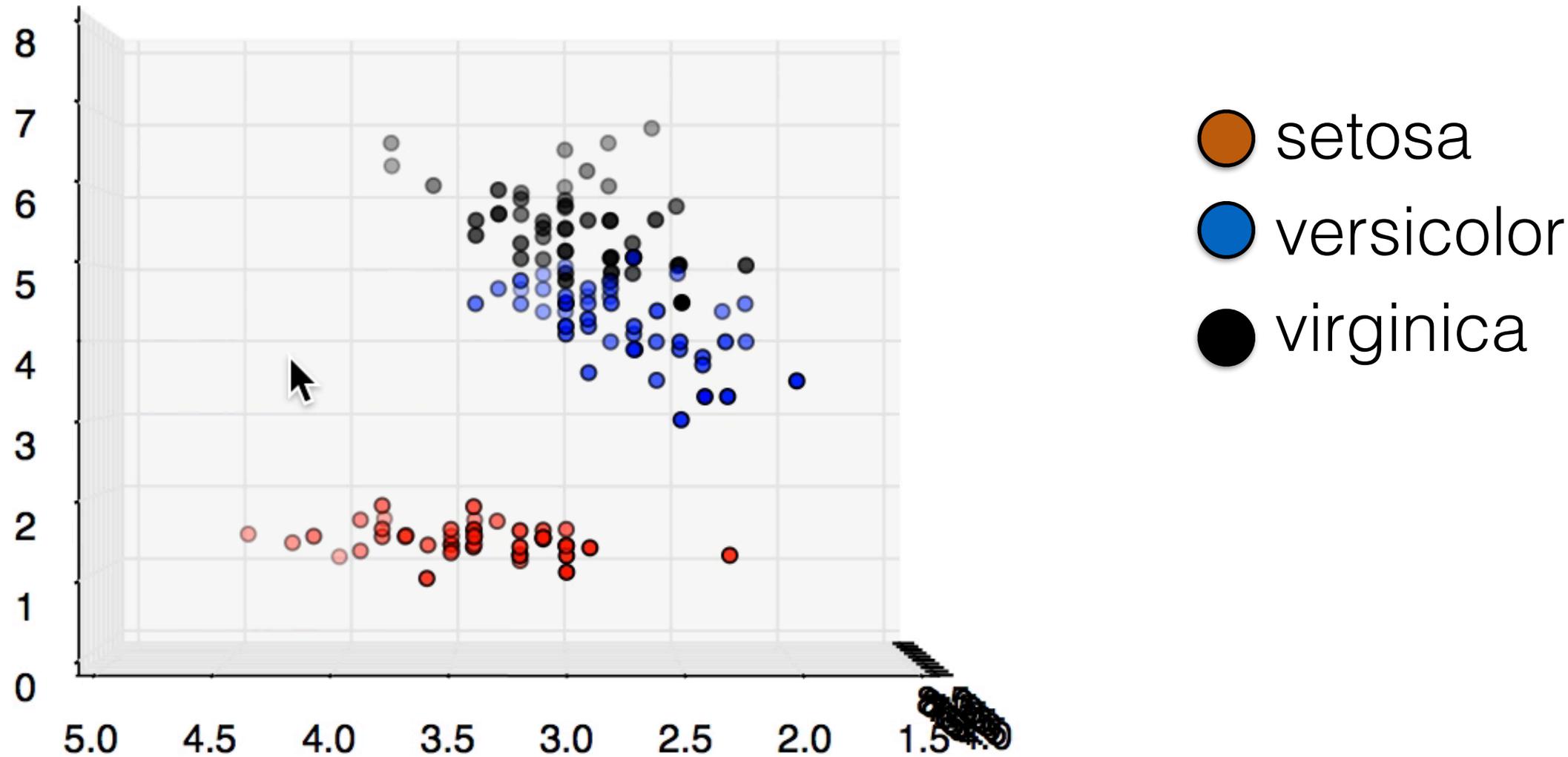
visualize features



x1: sepal length in cm
x2: sepal width in cm
x3: petal length in cm
x4: petal width in cm

image source:
mirlab.org

visualizing features



what about **four** dimensions?

features

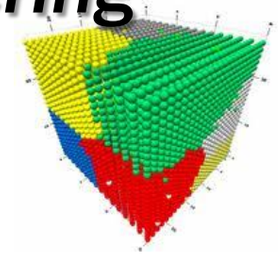
- most common is numeric
- vector quantization
- bag of words
  - term frequency: percentage of a times a word appears in a type of document
  - inverse document frequency
- graphs
  - used to quantize

take data mining!
or python machine learning!

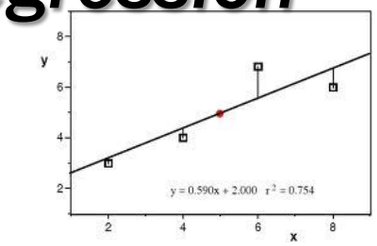
types of machine learning

- many ways to categorize machine learning

Clustering



Regression



Prediction

Classification

Ordinal Reg.

unlabeled

labeled

unsupervised

supervised

types of machine learning

```
[5.1,  3.5,  1.4,  0.2]
[5.7,  2.8,  4.5,  1.3]
[7.6,  3. ,  6.6,  2.1]
```

setosa
versicolor
virginica

unstructured
predict one

Classification

The quick brown fox jumped over the lazy dog.

ia adj adj noun verb prep. ia adj noun

structured **predict a structured object**

types of machine learning

Classification

unknown

`[7.5, 3.7 , 5.0, 2.7]`

`*[1,0,1,1] =15.2`

setosa	`[5.1, 3.5, 1.4, 0.2]*[1,0,1,1] =5.7`	**<7**	
versicolor	`[5.7, 2.8, 4.5, 1.3]*[1,0,1,1] =11.7`	**else**	
virginica	`[7.6, 3. , 6.6, 2.1]*[1,0,1,1] =14.3`	**>13**	

store all labeled examples **nonparametric**

method 1: closest example in training set

multiply input by [1, 0, 1, 1] **parametric**

method 2: sum greater than value for each class

ML algorithms

nonparametric

- nearest neighbor
- k-nearest neighbor (KNN)
- kernel density estimator

parametric

- decision tree
- random forest/boosted trees
- logistic regression
- neural networks
- gaussian mixtures

- support vector machines
- and many many more…

decision trees

- remember the iris data set?

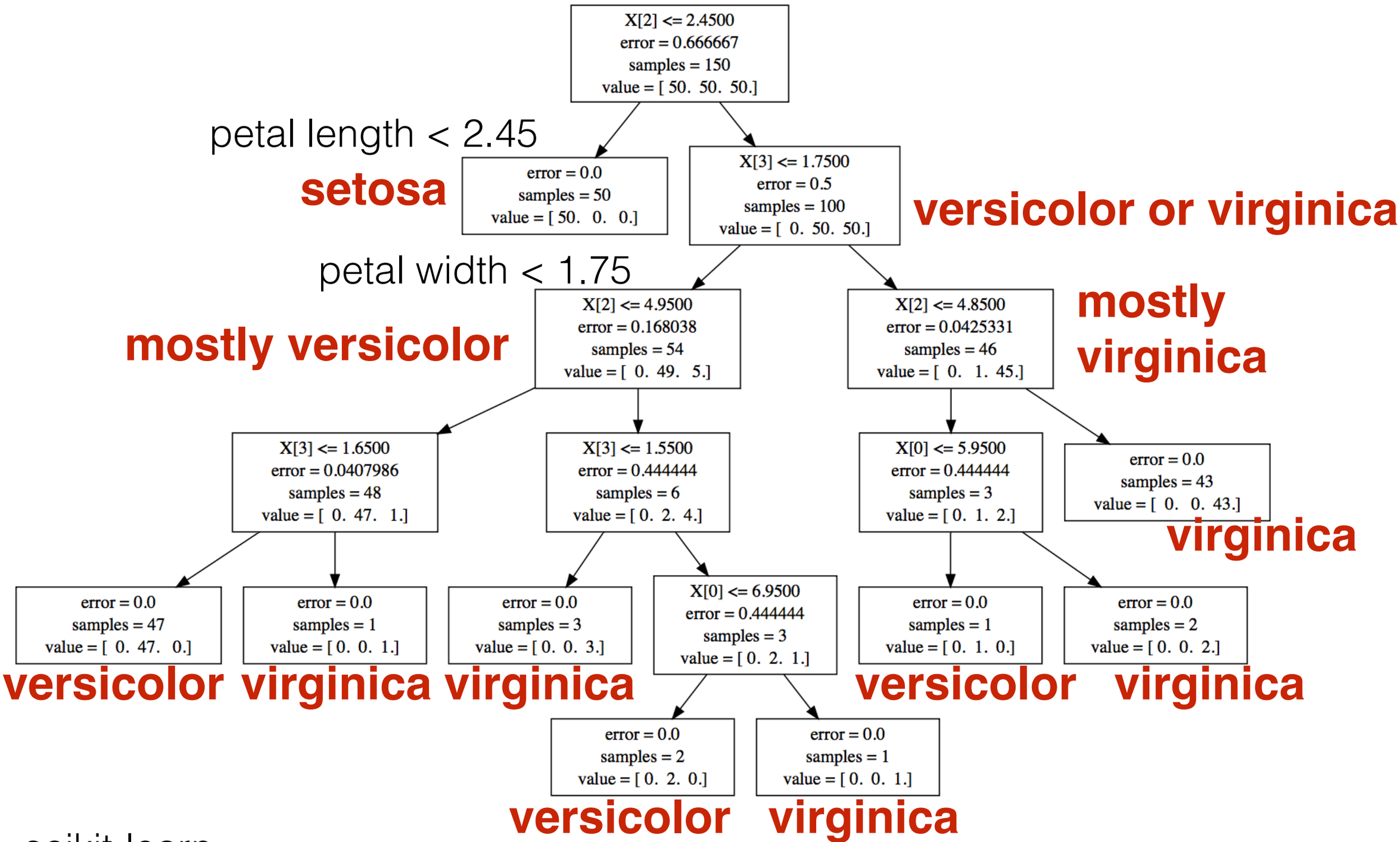
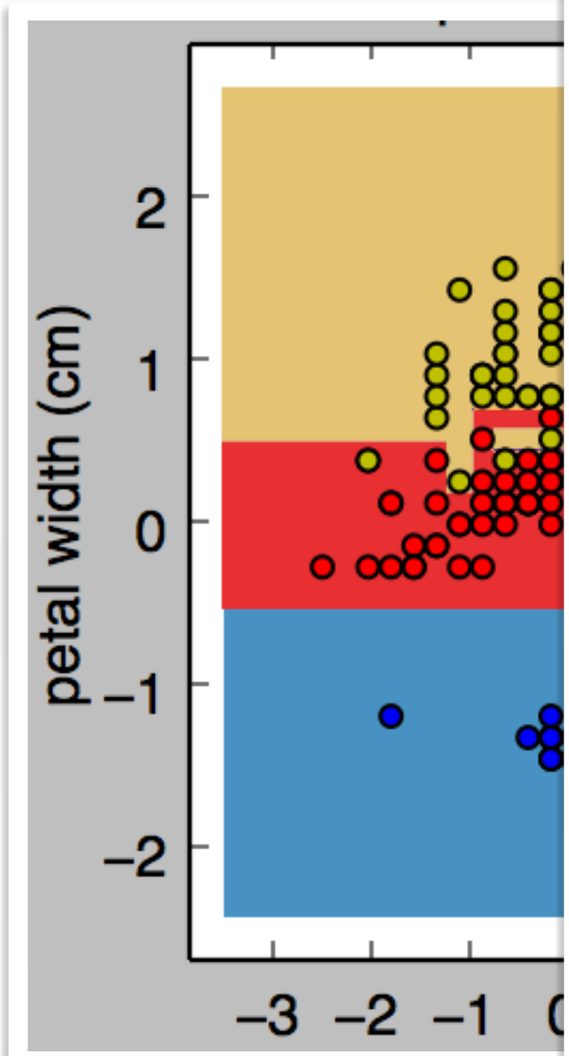
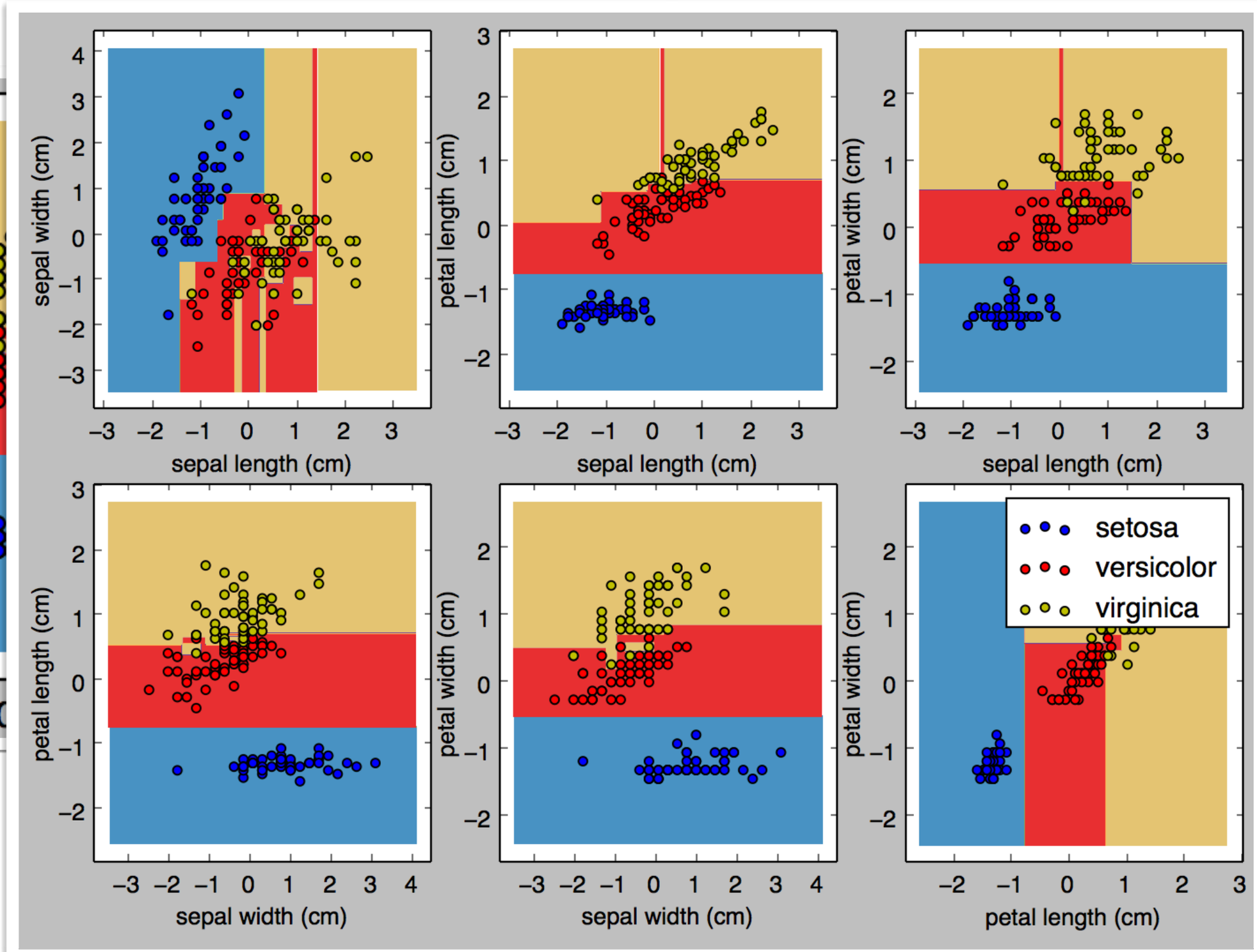


image: scikit-learn

decision tree

- decision

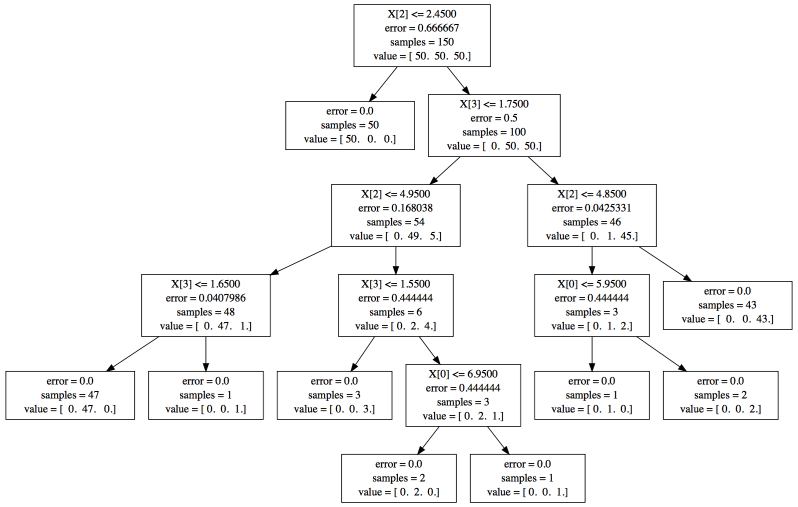


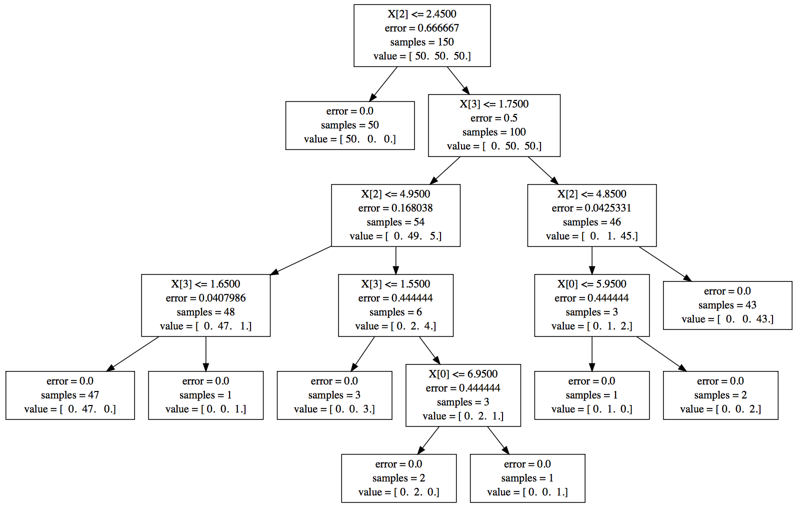


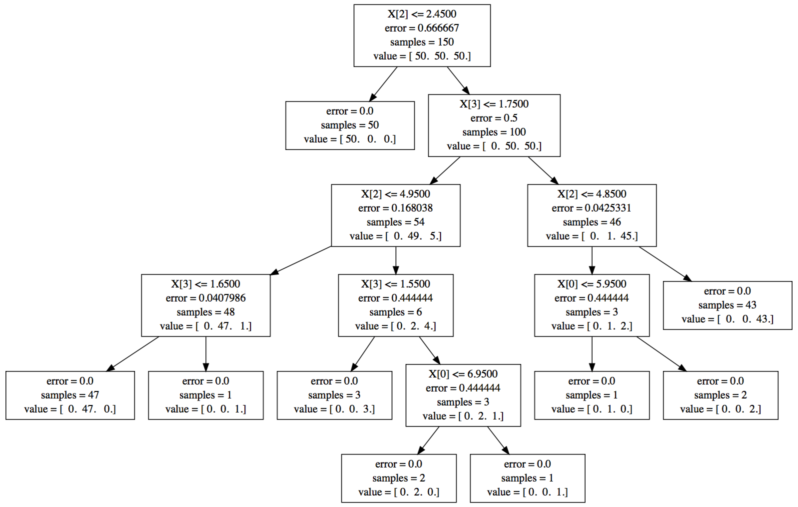
random forests

- make a bunch of trees
  - each tree made with a random subset all training data
  - and a random subset of the features

120 examples 3 features	120 examples 3 features	120 examples 3 features







x 200 trees

unknown vector?
query each tree

95% say setosa
4% say versicolor
1% say virginica

k-nearest neighbor

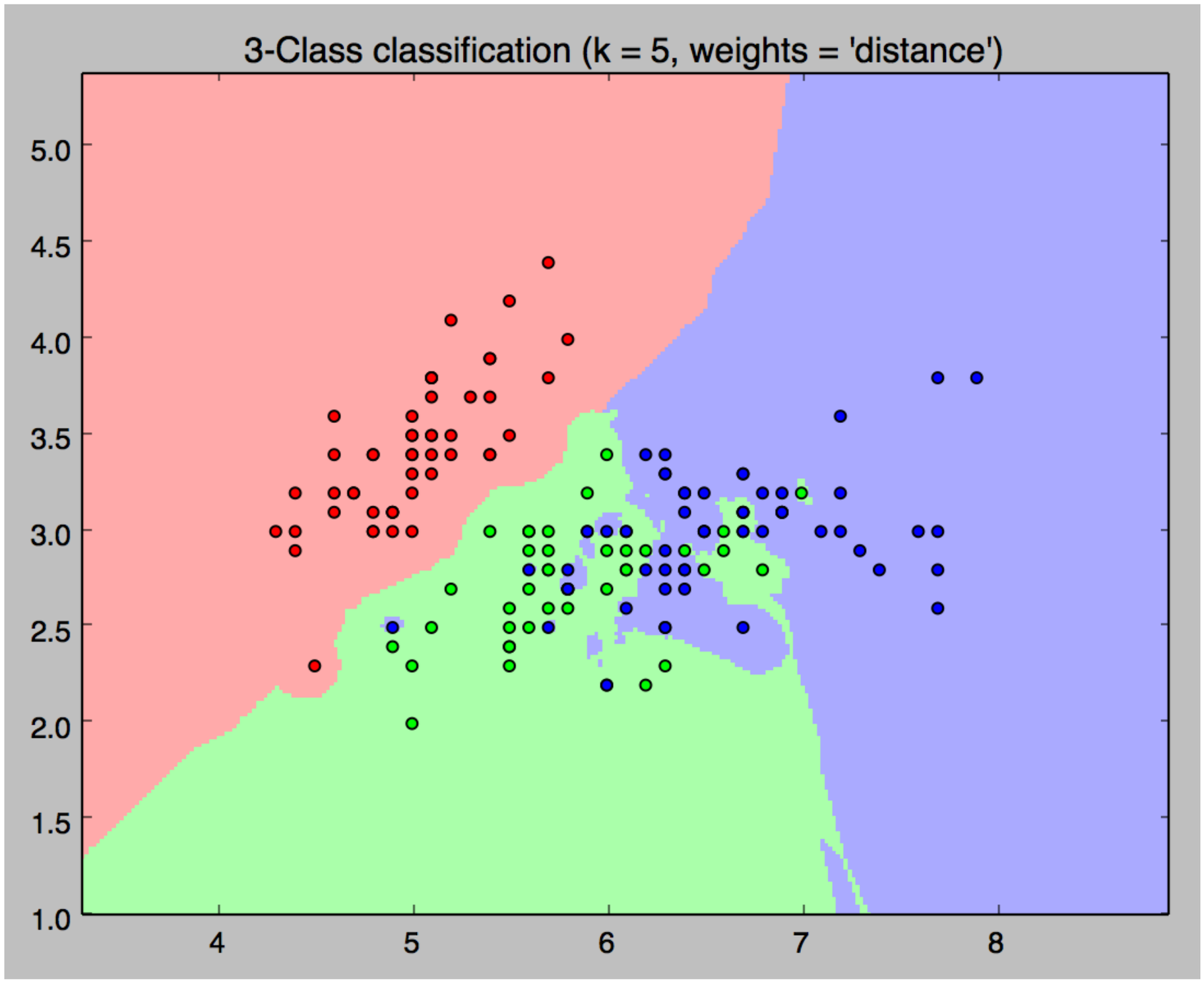
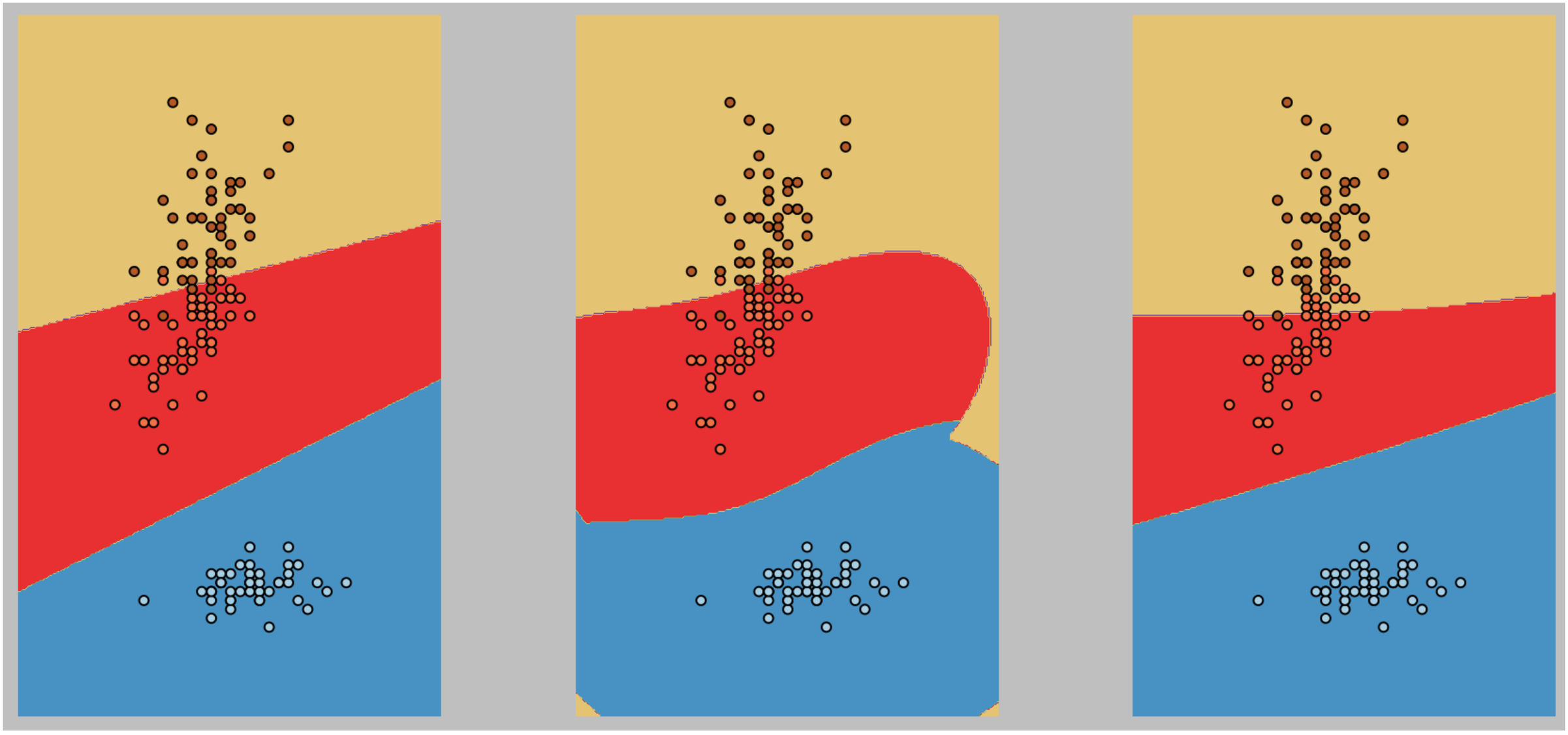


image:
datasciencerules.org

support vector machines

- find a vector that separates the training data



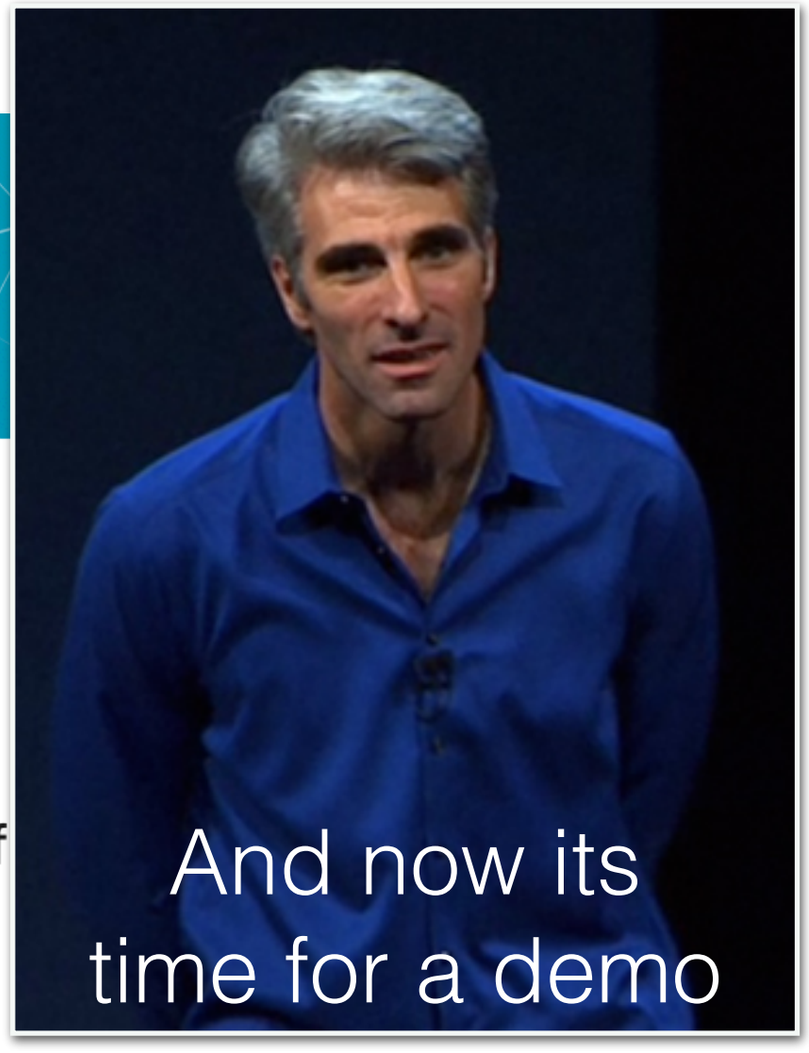
linear vector | gaussians | sigmoid

turi create

- demo of turi create

Carlos Guestrin · 2nd

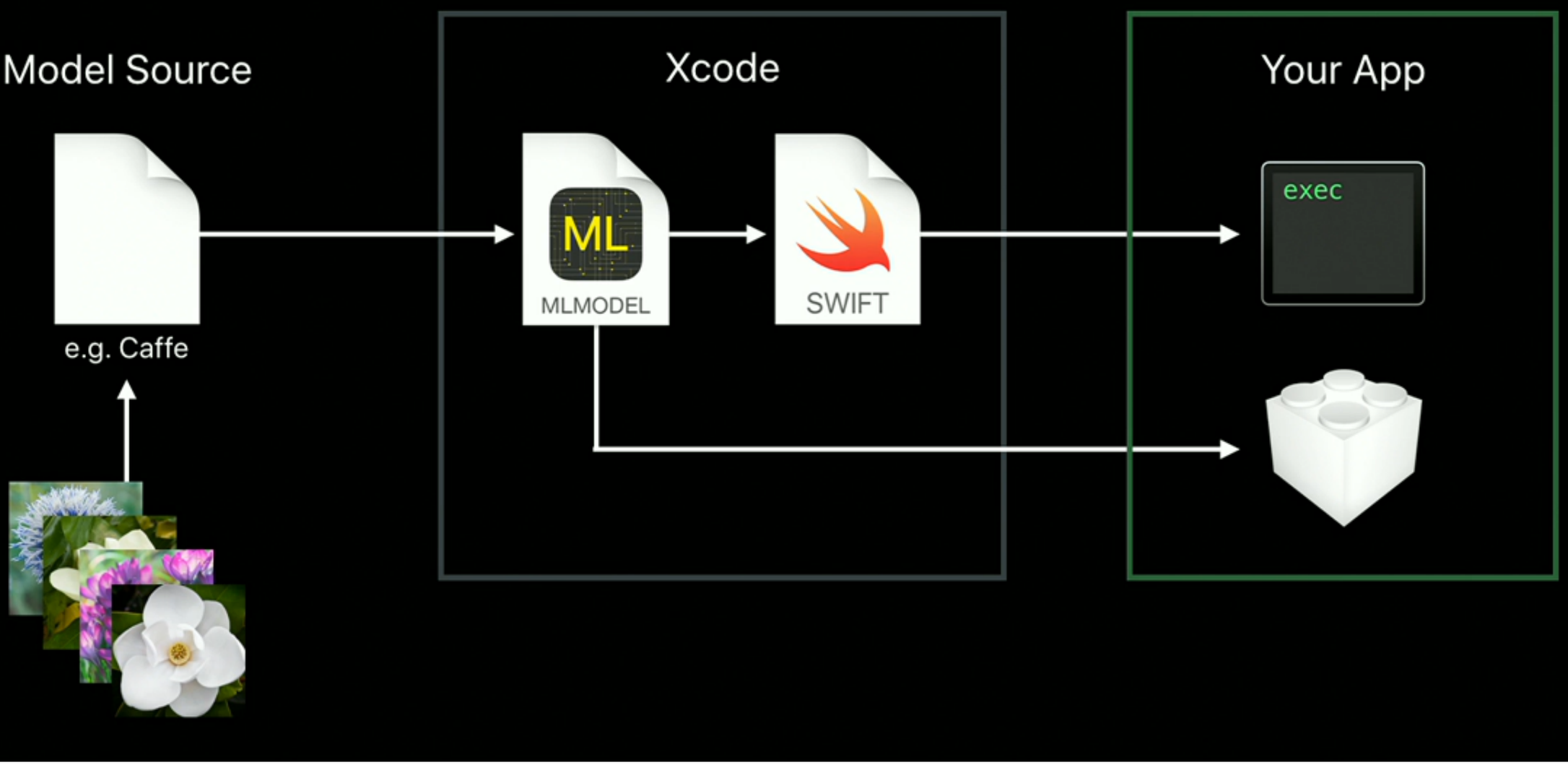
Senior Director of AI and Machine Learning at Apple & Amazon Professor of Machine Learning at University of Washington



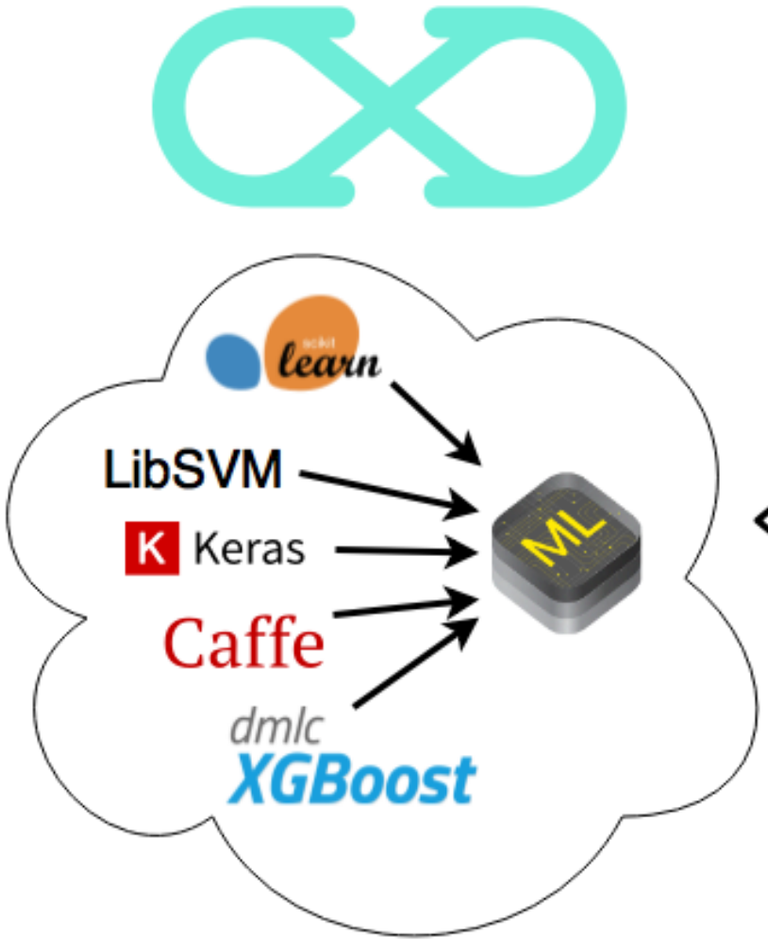
python_short_examples>TuriExample.ipynb

CoreML

Conversion Workflow

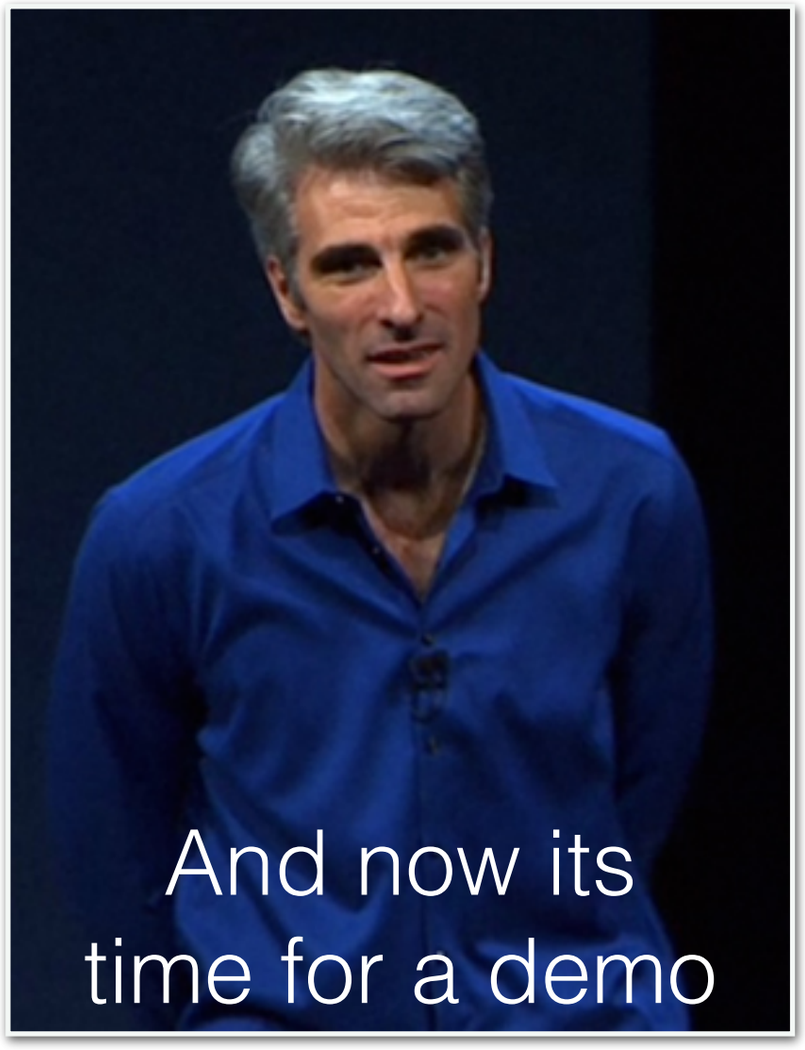


CoreML



CoreML, a taste

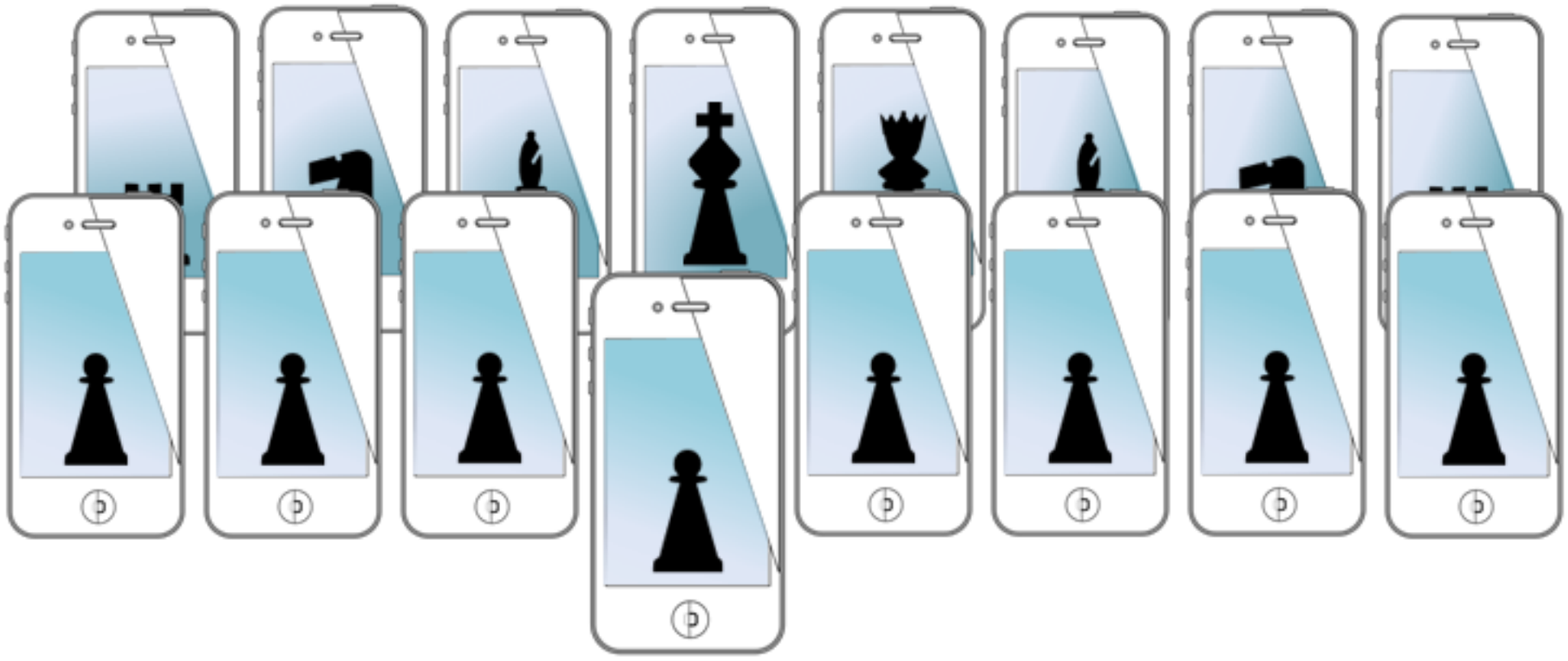
- demo using vision API



for next time…

- flipped module for using turi and iOS
- even if you are in Data Mining or Python ML, take a look
  - maybe skip over parts you know
  - we will have a near production level ML as a service platform by the end of the video lecture

MOBILE SENSING LEARNING



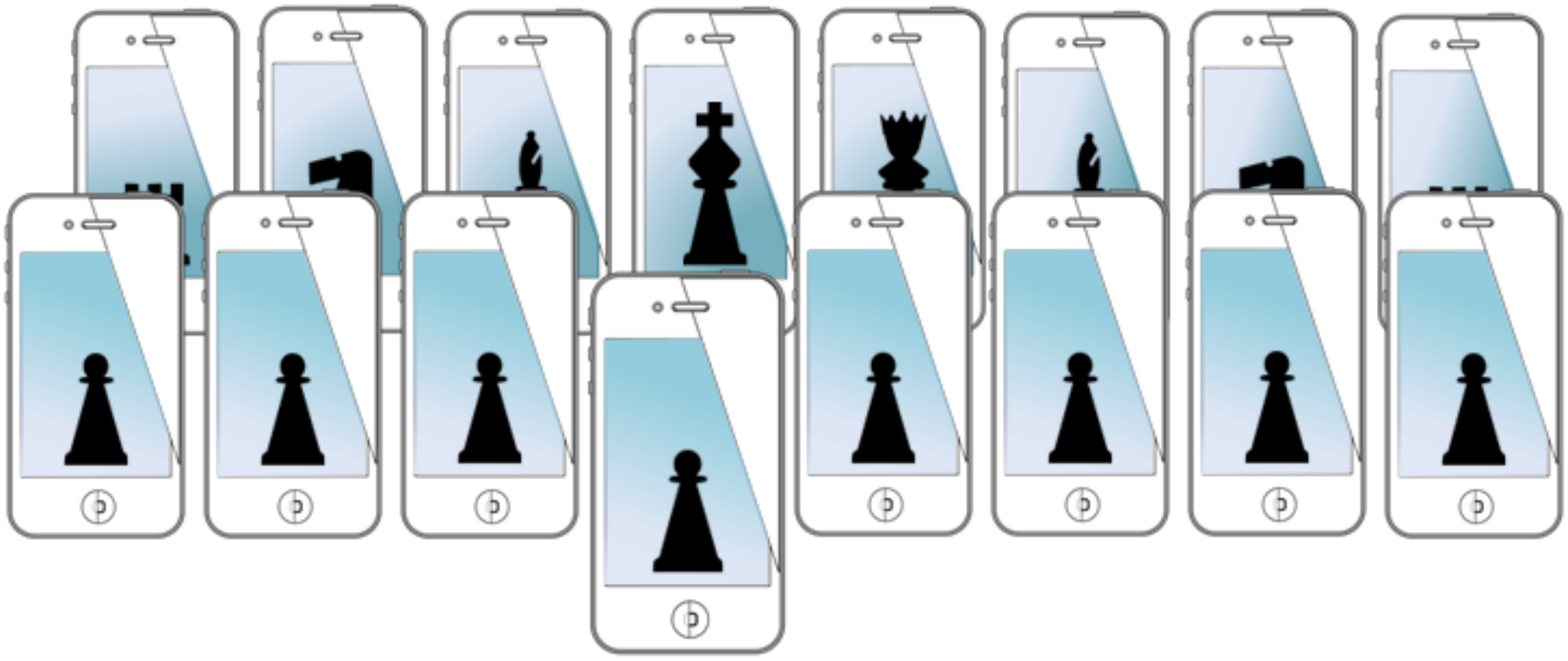
CS5323 & 7323

Mobile Sensing and Learning

machine learning crash course

Eric C. Larson, Lyle School of Engineering,
Computer Science, Southern Methodist University

MOBILE SENSING LEARNING



CS5323 & 7323

Mobile Sensing and Learning

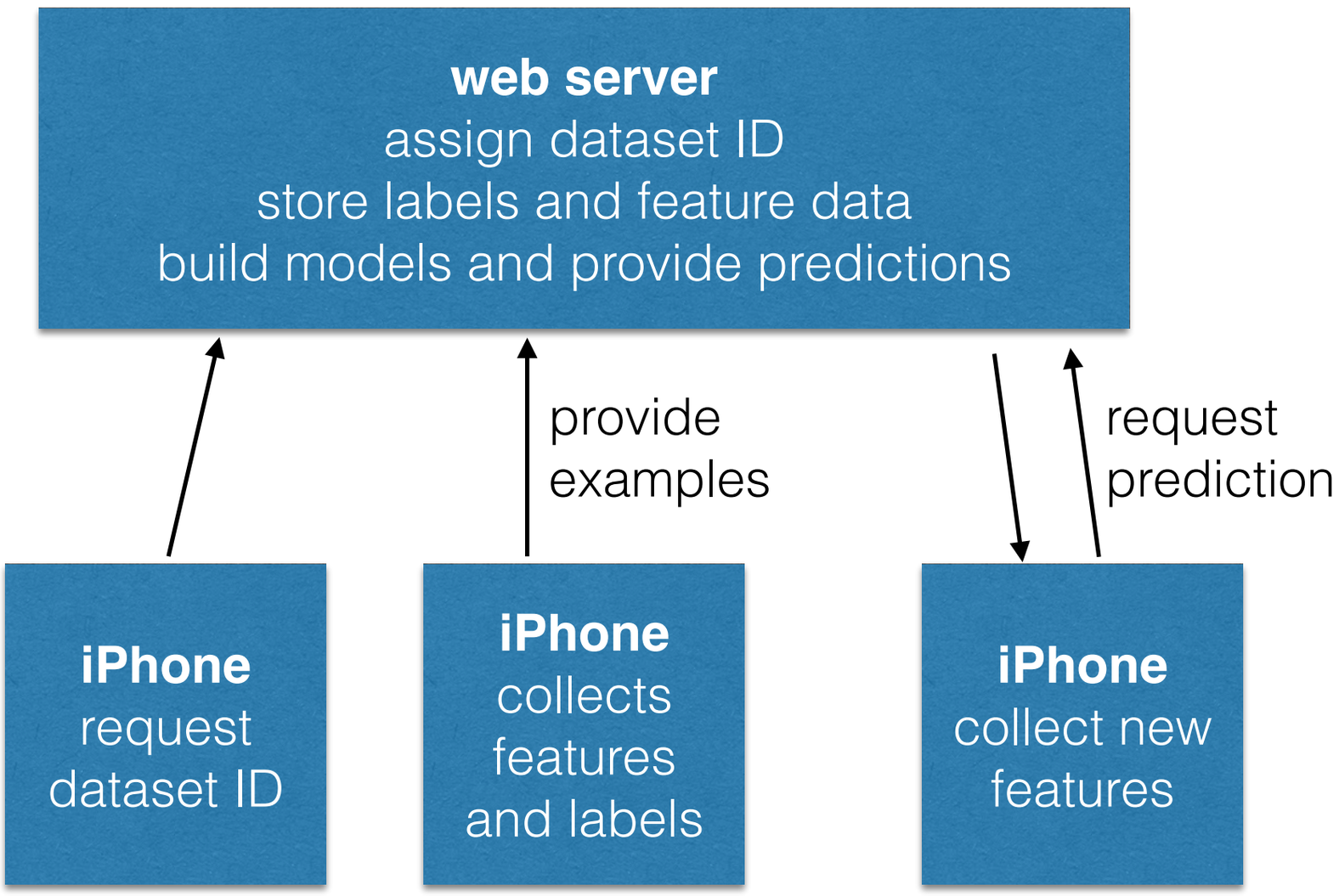
Video Lecture: machine learning and turi-create

Eric C. Larson, Lyle School of Engineering,
Computer Science, Southern Methodist University

Video Agenda

- pitfalls of machine learning
- turi and SFrames
- assignment 6 code base

assignment 6



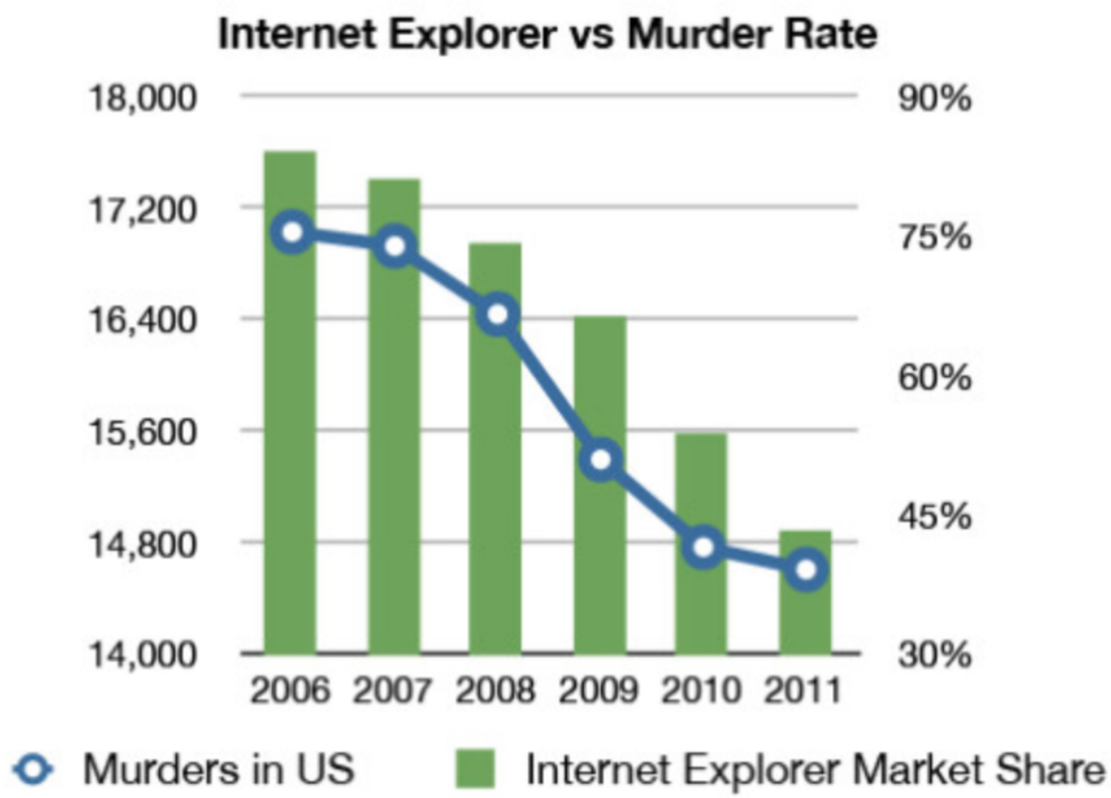
all uploads handled via JSON formatted HTTP requests

big data pitfalls

- last time
  - machine learning was going to save the world
  - we talked about decision trees, forests, KNN, and SVMs
- there is no free lunch

big data pitfalls

- machine learning can detect correlations
- correlation is not necessarily causal
  - murder rate and internet explorer
  - organic food and autism
- machine learning does not
  - humans interpret the resul
  - you need to know someth
  - example: animal camoufla



big data pitfalls

- machine learning can still be easy to game
  - google hits
  - MS word “writing score”
- results are less robust than you think
  - stock market
  - google’s flu tracker
  - incoming data changes - especially on the web

big data pitfalls

- the sky net problem
  - big data from big data
  - reinforcement

WIKIPEDIA
The Free Encyclopedia

Main page
Contents

Google Translate

From Wikipedia, the free encyclopedia

Google Translate is a free, multilingual statistical machine-translation service provided by Google Inc. to translate written text from one language into another.

- cultural contribution

big data pitfalls

- there is never enough data to replace human thought
  - we can put a sentence together that has never been uttered in history
- big data is really hyped
  - its cool, but won't answer everything "they" say it will
  - antibiotics are more important than finding malaria or the flu
  - if it seems like magic, there is likely something wrong

http://www.marketsforgood.org/eight-no-nine-problems-with-big-data/

avoiding pitfalls

- there are lots of ways to be wrong
  - not enough diversity in classes
  - http://musicmachinery.com/tag/netflix/

Music recommendation is broken

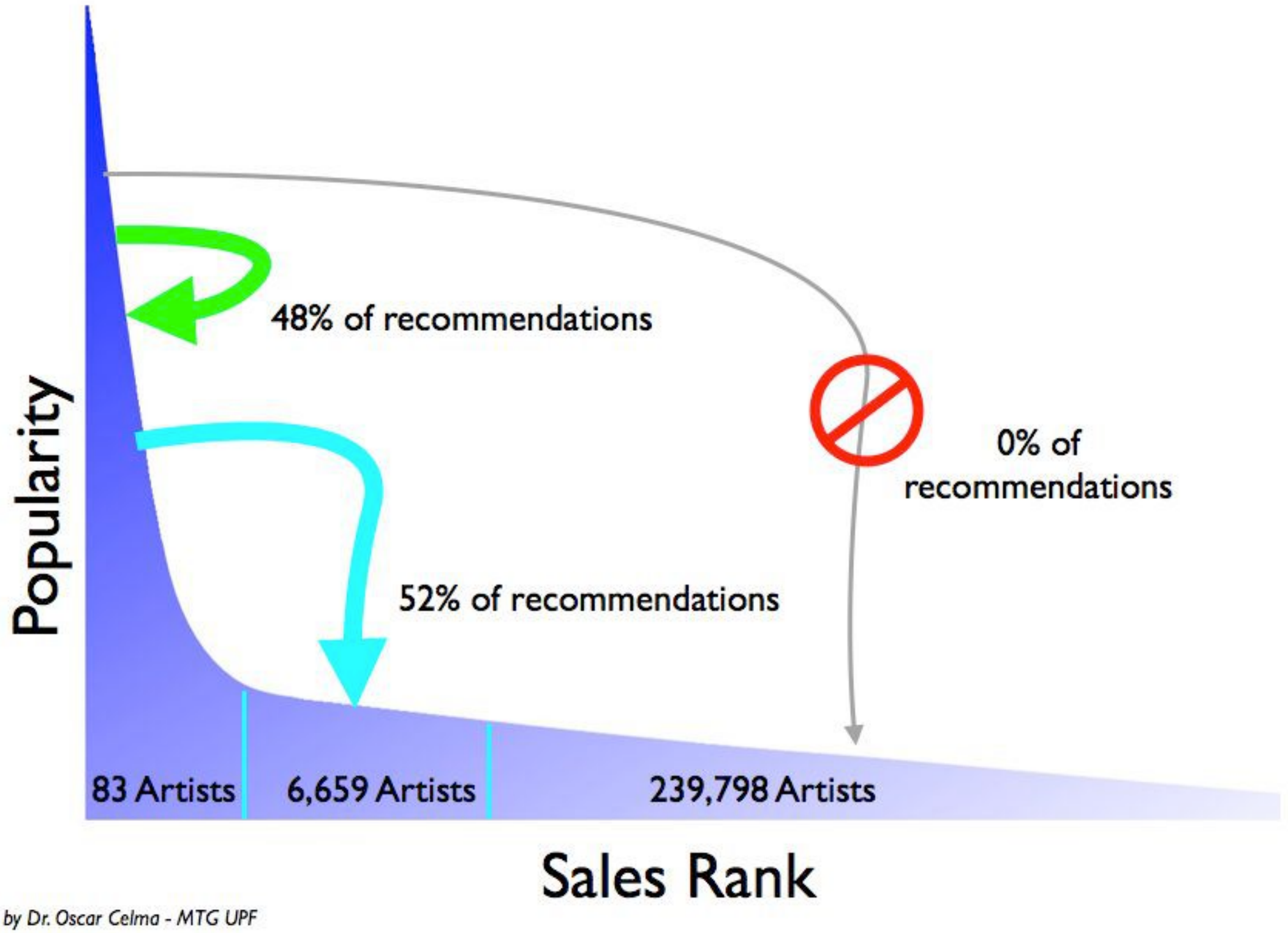
A recommendation that no human would make

If you like Britney Spears ...

You might like the Report on Pre-War Intelligence

Help! I'm stuck in the head

The limited reach of music recommendation



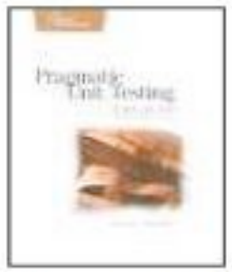
Study by Dr. Oscar Celma - MTG UPF

The Harry Potter Problem

If you like X you might like Harry Potter

Powell's Recommendations

If you enjoyed Java RMI by *William Grosso*, you might also enjoy the following titles:

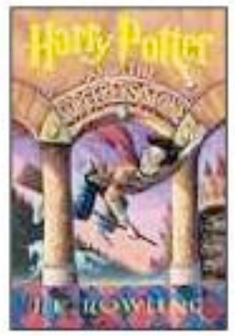


$29.95
New Trade Paper
ADD TO CART
add to wishlist

Pragmatic Unit Testing in Java with JUnit (Pragmatic Programmers)
Andrew Hunt

$41.00
Used Hardcover
ADD TO CART
add to wishlist

Design Patterns: Elements of Reusable Object-Oriented Software (Addison-Wesley Professional Computing)
Erich Gamma



$14.95
Used Hardcover
ADD TO CART
add to wishlist

Harry Potter #01: Harry Potter and the Sorcerer's Stone
J K Rowling

avoiding pitfalls

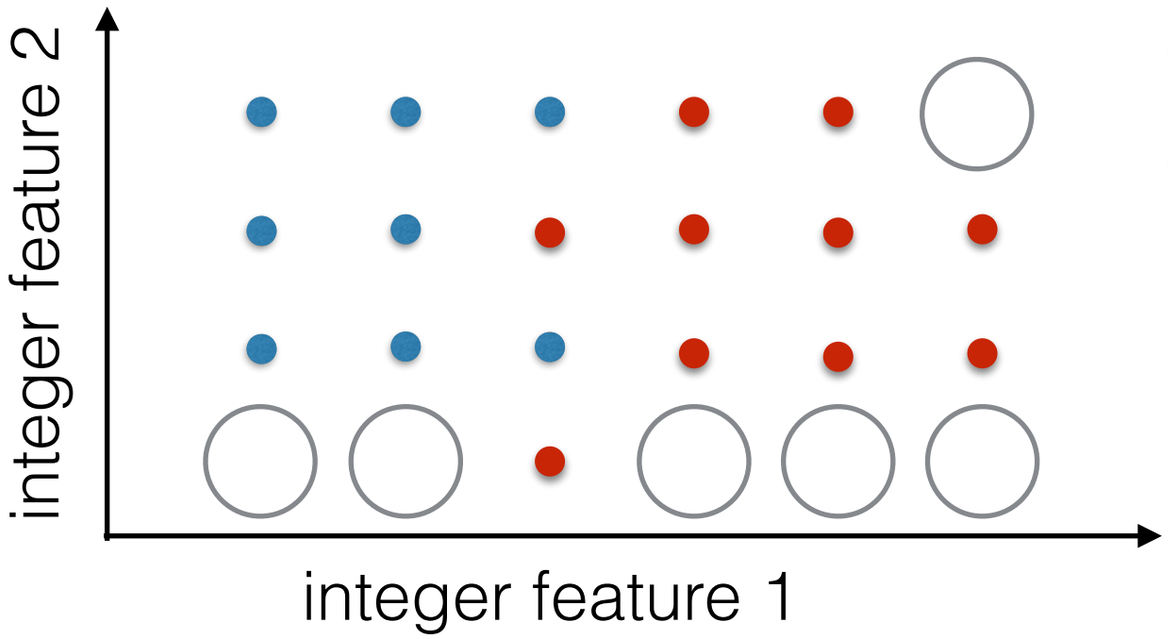
- there are lots of ways to be wrong
  - not enough diversity in classes
  - http://musicmachinery.com/tag/netflix/
  - outliers and novelty
    - ignore the deviant information
    - …or use it to detect novelty
  - more data is better than cleverness
  - but data alone is not enough
  - performance on training data means almost nothing
  - dimensionality can be a curse

collecting the **right** data is **difficult**

http://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf

more data!

- a paradox
  - example characterizing a 2D space of integer features



feature 2 can have 4 values
feature 1 can have 6 values
4 x 6 = 24 possible values
we have 18 examples
we need to predict
6 unknown values

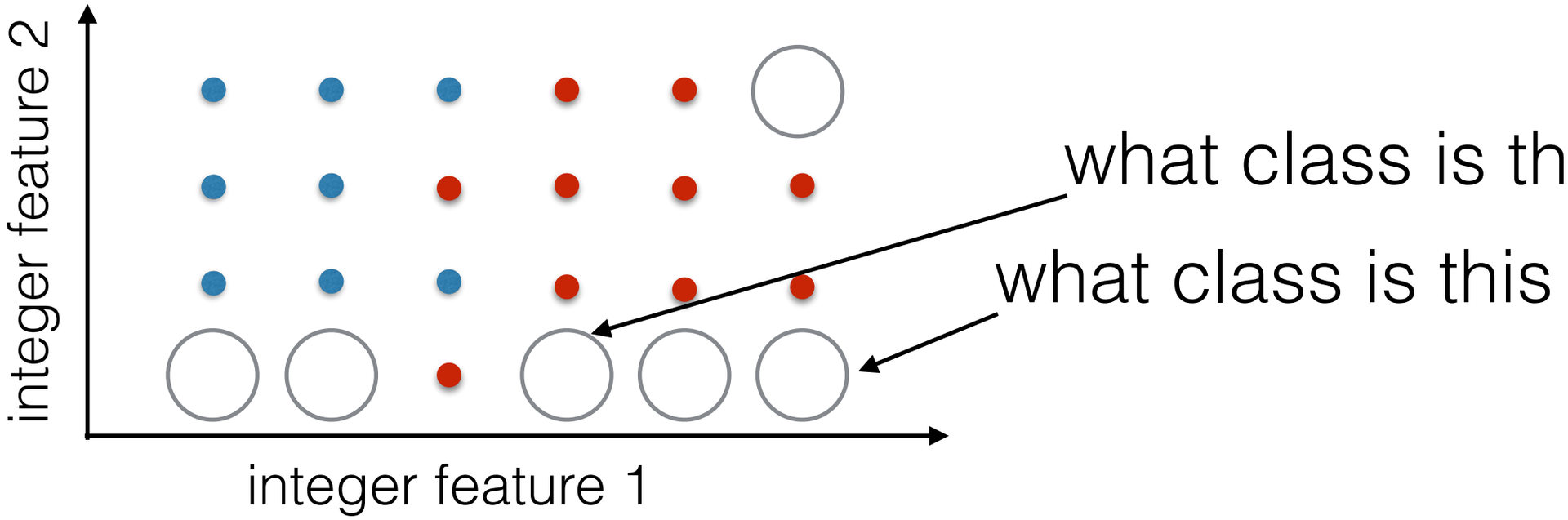
more realistic: 50 binary variables
2^{50} = 1.1 x 10^{15} possibilites

with one million examples
there are still 1.1 x 10^{15}
examples we have not seen

you can never collect enough data, ever…

more data!

- a paradox
  - example characterizing a 2D space of integer features



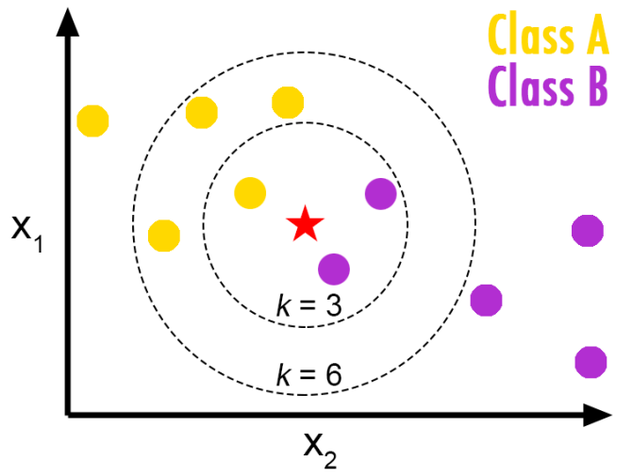
data does not span the entire space of features

and usually there is structure to exploit

so more data helps when examples show structure

dimensionality

- intuition breaks in high dimensions
- an example using KNN
  - 100 features
  - 3 features are important and the rest are noise
  - the 97 other features mask the other three
    - nothing gets learned
    - even if all 100 features are relevant, the distances look similar
- another example: in higher dimensions most volume is in the shell
  - *i.e.*, an orange in higher dimensions



dimensionality

- reduce dimensionality
  - PCA, ICA, SVD, NMF, CMF
  - sparse coding (with a dictionary)
  - feature selection
- pre-process training data
  - scale/normalize each dimension
  - demean each dimension
  - de-correlate the features (whitening)
  - make features binary / categories

find a way to ease the learning process

take statistics class

now you know keywords!

- that's 60% of the battle
- having a vague idea about these concepts will:
  - help with intuition in machine learning
  - impress your interviewers
- allow you to use toolkits without completely knowing the math
  - and no one knows the math in a complete way
  - unless they implemented your entire toolkit

turi-create

Carlos Guestrin · 2nd

Senior Director of AI and Machine Learning at Apple & Amazon Professor of Machine Learning at University of Washington

- builds upon
  - graphlab (graph structures in python)
  - SArray (scalable matrix arrays in python)
- you classify a label based upon the features
- many algorithms to choose from (auto selection)
- **automatic** data transformation and scaling
- text feature extraction for documents
- graphlab base lets you build on:
  - out of core memory
  - subsumed parallelism

turi
create intelligence™

SArrays

```
>>> sa = SArray([1,1,1,1,1])
>>> sb = SArray([2,2,2,2,2])
>>> sc = sa + sb
>>> sc
dtype: int
Rows: 5
[3, 3, 3, 3, 3]
>>> sc + 2
dtype: int
Rows: 5
[5, 5, 5, 5, 5]
```

- numeric operations in python (similar to numpy)
- we will use it for creating vectors of data
- but its much more than that
  - can handle billions of data examples
  - even if they cannot be stored in memory

```
>>> sa[1000:] # Returns an SArray containing rows 1000 to the end
>>> sa[:1000] # Returns an SArray containing rows 0 to row 999 inclusive
>>> sa[0:1000:2] # Returns an SArray containing rows 0 to row 1000 in steps of 2
>>> sa[-100:] # Returns an SArray containing last 100 rows
>>> sa[-100:len(sa):2] # Returns an SArray containing last 100 rows in steps of 2
```

SFrame feature vectors

- make vector of feature vectors

```
array([[ 0.03305292,  0.79966245,  0.324738  ,  0.16299925,  0.67718303],
       [ 0.4452092 ,  0.41934637,  0.05948603,  0.39552644,  0.3674299 ],
       [ 0.95650065,  0.31275392,  0.68912272,  0.7687402 ,  0.59179067],
       [ 0.79916251,  0.56120797,  0.38133393,  0.87441416,  0.10916493],
       [ 0.91580924,  0.24394441,  0.084804  ,  0.38619409,  0.543082  ],
       [ 0.18087033,  0.82412418,  0.83376823,  0.83815408,  0.76500212],
       [ 0.62735831,  0.30187534,  0.04393703,  0.91338623,  0.20756493],
       [ 0.46796337,  0.31295322,  0.70044522,  0.85973994,  0.08801671],
       [ 0.71538186,  0.02943904,  0.55127129,  0.06445052,  0.91766324],
       [ 0.72201574,  0.9279875 ,  0.7838015 ,  0.30510338,  0.6055239 ]])
```

0th feature vector

1st feature vector

9th feature vector

five features per vector

matrix notation

each row is an observed feature vector

each column is a separate feature

classifiers

```
# now let's repeat us
import turicreate as
from sklearn import d
import numpy as np

bunch = ds.load_iris(

# make dictionary fro
tmp_dict = {'target':
for i in range(0,len(
    key = bunch.featu
    tmp_dict[key] = b

iris_sframe = tc.SFra

# clean up memory
del tmp_dict, bunch

model = tc.classifier
yhat_sframe = model.c
yhat_sframe
```

```
# Load the data
data =  tc.SFrame('ratings-data.csv')

# Restaurants with rating >=3 are good
data['is_good'] = data['stars'] >= 3

# Make a train-test split
train_data, test_data = data.random_split(0.8)

# Automatically picks the right model based on your data.
model = tc.classifier.create(train_data, target='is_good',
                             features = ['user_avg_stars',
                                         'business_avg_stars',
                                         'user_review_count',
                                         'business_review_count'])

# Generate predictions (class/probabilities etc.), contained in an SFrame.
predictions = model.classify(test_data)

# Evaluate the model, with the results stored in a dictionary
results = model.evaluate(test_data)
```

Turi Create implementations are built to work with up to billions of examples and up to millions of features.

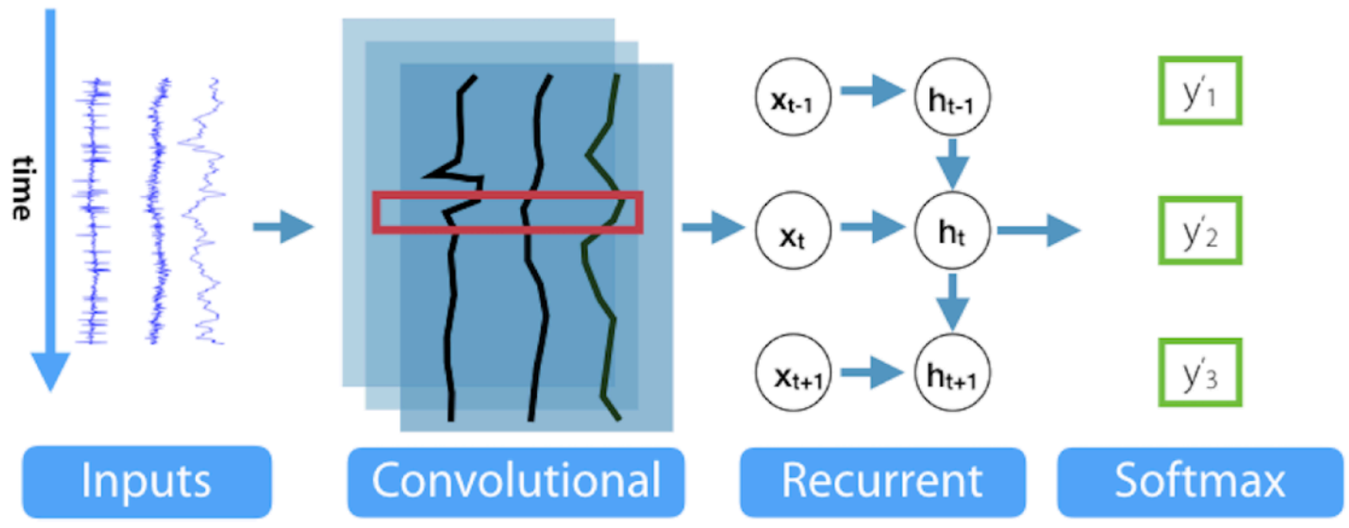
many built in classifiers for iOS data

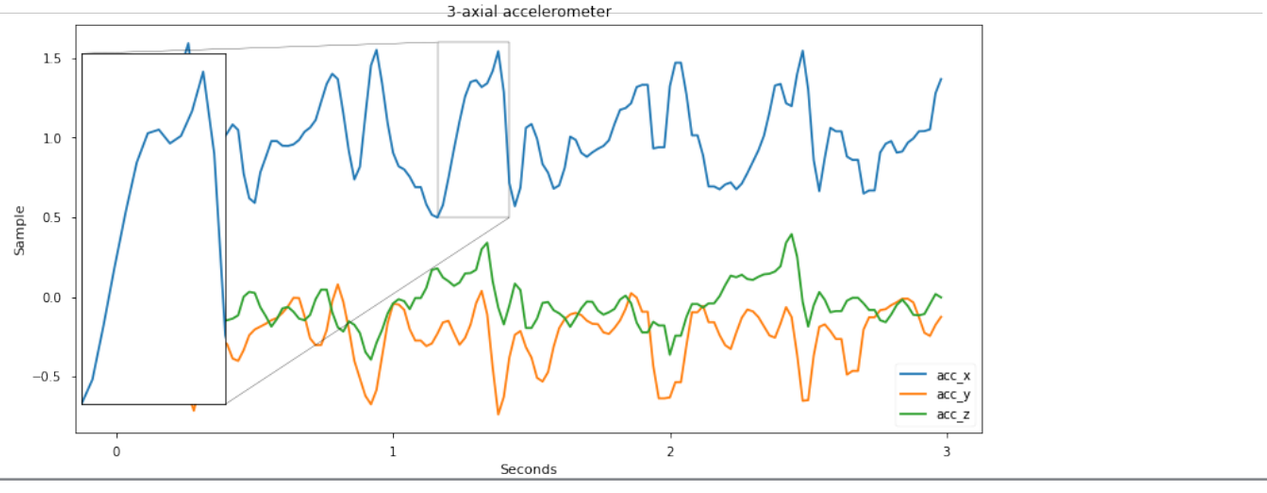
Task focused toolkits

- Recommender Systems
- Image Classification
- Drawing Classification
- Sound Classification
- Image Similarity
- Object Detection
- One-Shot Object Detection
- Style Transfer
- Activity Classification
- Text Classifier

How does this work?

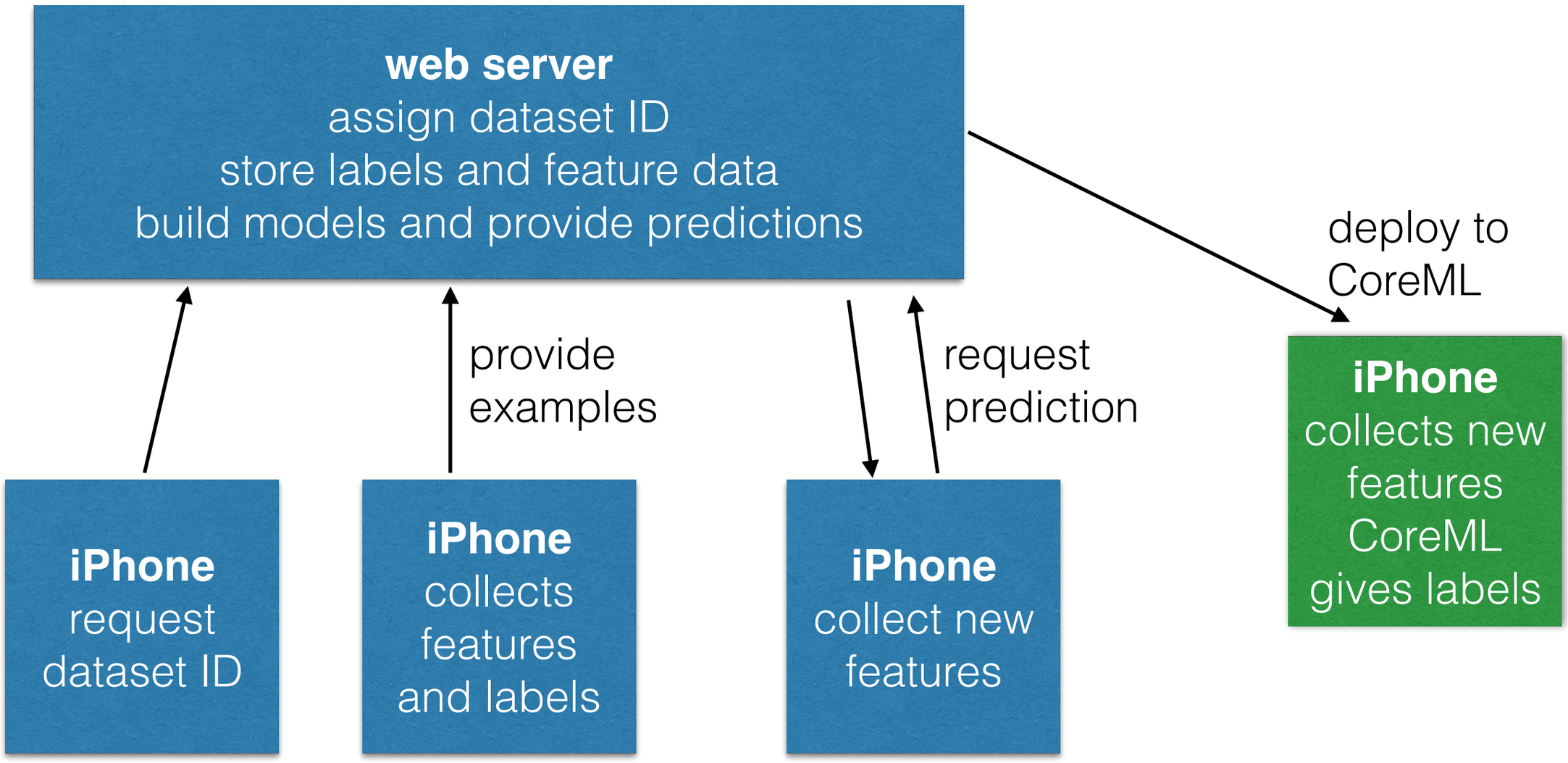
The underlying model of the activity classifier in Turi Create is a deep learning model capable of detecting temporal patterns in sensor data, lending itself well to the task of activity classification. The deep learning model relies on convolutional layers to extract temporal features from a single prediction window, for example an arching movement could possibly be a strong indicator of swimming. Furthermore, it relies on recurrent layers to extract temporal features over time, for example if a subject was swimming in the previous timestamp, then it is most likely not sky diving in the next. Below is a sketch of the neural network used for the activity classifier in Turi Create.





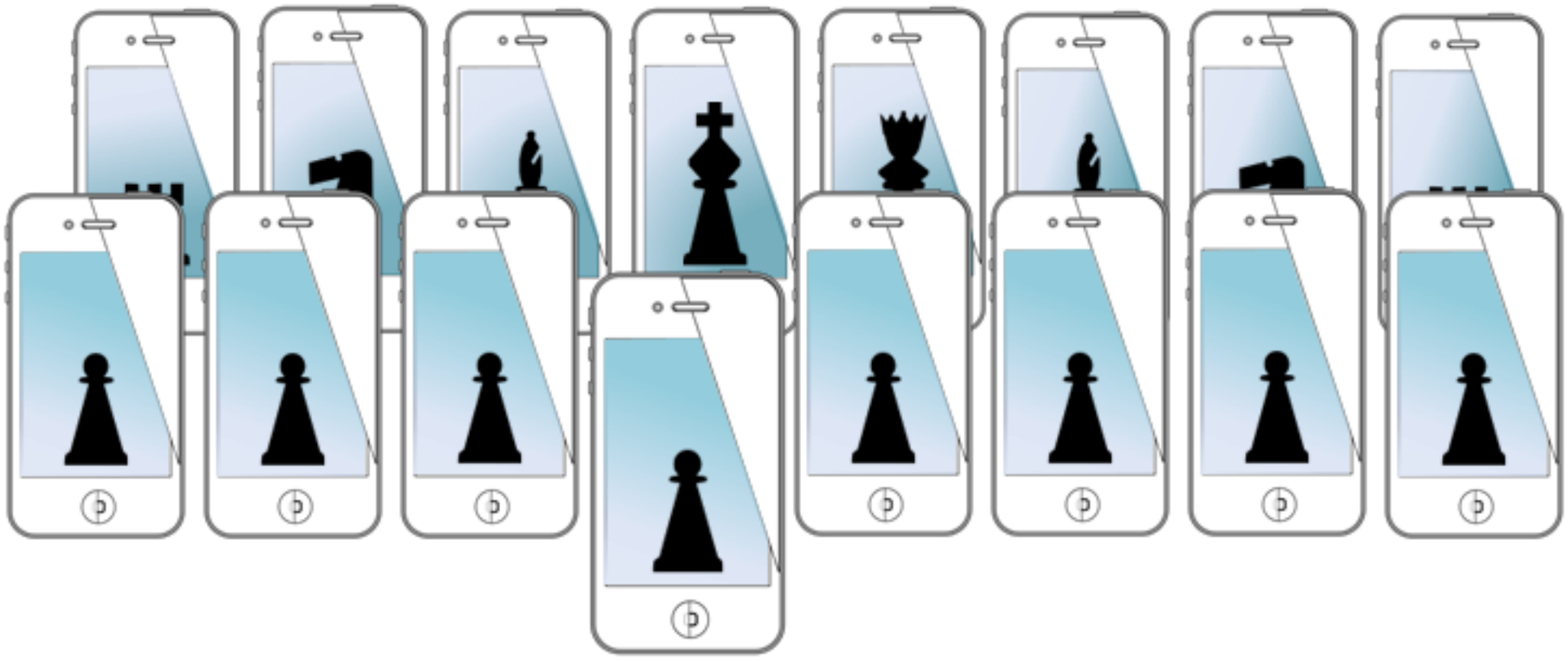
https://apple.github.io/turicreate/docs/userguide/applications/

assignment 6 demo



all uploads handled via JSON formatted HTTP requests posted on canvas!

MOBILE SENSING LEARNING



CS5323 & 7323

Mobile Sensing and Learning

Video Lecture: machine learning and scikit-learn

Eric C. Larson, Lyle School of Engineering,
Computer Science, Southern Methodist University

Back up slides
& archived slides

scikit-learn

- builds upon
  - numpy (matrix math in python)
  - scipy (math in for scientific applications)
- you classify a label based upon the features
- many algorithms to choose from (unstructured)
- automatic data transformation and scaling
- text feature extraction for documents
- numpy and scipy base lets you build on:
  - memory mapped training data
  - parallelism through sklearn.joblib and IPython

numpy and scipy

- numeric operations in python
- we will use it for creating vectors of data
- but its much more than that

```
import numpy as np
>>> a = np.array([3,4,5,5])
>>> a.shape
(4,)
>>> a = np.array([3,4,5,5]).reshape(1,4)
>>> a
array([[3, 4, 5, 5]])
>>> a.shape
(1, 4)
>>> b = np.array([6,6,7,8]).reshape(1,4)
>>> b
array([[6, 6, 7, 8]])
>>> c = np.row_stack((a,b))
>>> c
array([[3, 4, 5, 5],
       [6, 6, 7, 8]])
>>> d = np.column_stack((a,b))
>>> d
array([[3, 4, 5, 5, 6, 6, 7, 8]])
```

```
>>> c[0,3]
5
>>> c[:,3]
array([5, 8])
>>> c[1,:]
array([6, 6, 7, 8])
>>> i = [0,3]
>>> c[0,i]
array([3, 5])
>>> mx = c.max()
>>> mx
8
>>> mx = c.max(axis=0)
>>> mx
array([6, 6, 7, 8])
```

numpy and scipy

```
>>> t = np.linspace(0,3,30)
array([ 0.        ,  0.10344828,  0.20689655,  0.31034483,  0.4137931 ,
        0.51724138,  0.62068966,  0.72413793,  0.82758621,  0.93103448,
        1.03448276,  1.13793103,  1.24137931,  1.34482759,  1.44827586,
        1.55172414,  1.65517241,  1.75862069,  1.86206897,  1.96551724,
        2.06896552,  2.17241379,  2.27586207,  2.37931034,  2.48275862,
        2.5862069 ,  2.68965517,  2.79310345,  2.89655172,  3.        ])
>>> sin_of_t = np.sin(t)
array([ 0.        ,  0.10326387,  0.20542363,  0.30538701,  0.40208519,
        0.49448427,  0.58159632,  0.66248994,  0.73630021,  0.80223796,
        0.85959818,  0.90776756,  0.94623109,  0.97457752,  0.99250375,
        0.99981813,  0.99644245,  0.9824128 ,  0.95787919,  0.92310392,
        0.87845883,  0.82442125,  0.76156895,  0.69057395,  0.61219533,
        0.52727112,  0.43670932,  0.34147822,  0.24259603,  0.14112001])
```

```
>>> fft_of_sint = np.fft.fft(sin_of_t)
array([ 19.28999597 +0.00000000e+00j,  -6.27278308 +2.40171713e-01j,
        -1.29176102 +9.56185021e-02j,  -0.57513293 +6.03739172e-02j,
        -0.33617629 +4.35297803e-02j,  -0.22775143 +3.33829204e-02j,
        -0.16962290 +2.64488806e-02j,  -0.13503582 +2.13038604e-02j,
        -0.11297946 +1.72520130e-02j,  -0.09824882 +1.39102053e-02j,
        -0.08813125 +1.10480386e-02j,  -0.08111064 +8.51670442e-03j,
        -0.07629879 +6.21379015e-03j,  -0.07316375 +4.06429058e-03j,
        -0.07139229 +2.00951252e-03j,  -0.07081905 -1.39645240e-15j,
        -0.07139229 -2.00951252e-03j,  -0.07316375 -4.06429058e-03j,
        -0.07629879 -6.21379015e-03j,  -0.08111064 -8.51670442e-03j,
        -0.08813125 -1.10480386e-02j,  -0.09824882 -1.39102053e-02j,
        -0.11297946 -1.72520130e-02j,  -0.13503582 -2.13038604e-02j,
        -0.16962290 -2.64488806e-02j,  -0.22775143 -3.33829204e-02j,
        -0.33617629 -4.35297803e-02j,  -0.57513293 -6.03739172e-02j,
        -1.29176102 -9.56185021e-02j,  -6.27278308 -2.40171713e-01j])
```

feature vectors

- make vector of feature vectors

```
>>> features = np.random.random((10,5))

array([[ 0.03305292,  0.79966245,  0.324738  ,  0.16299925,  0.67718303],   0th feature vector
       [ 0.4452092 ,  0.41934637,  0.05948603,  0.39552644,  0.3674299 ],   1st feature vector
       [ 0.95650065,  0.31275392,  0.68912272,  0.7687402 ,  0.59179067],
       [ 0.79916251,  0.56120797,  0.38133393,  0.87441416,  0.10916493],
       [ 0.91580924,  0.24394441,  0.084804  ,  0.38619409,  0.543082  ],
       [ 0.18087033,  0.82412418,  0.83376823,  0.83815408,  0.76500212],
       [ 0.62735831,  0.30187534,  0.04393703,  0.91338623,  0.20756493],
       [ 0.46796337,  0.31295322,  0.70044522,  0.85973994,  0.08801671],
       [ 0.71538186,  0.02943904,  0.55127129,  0.06445052,  0.91766324],
       [ 0.72201574,  0.9279875 ,  0.7838015 ,  0.30510338,  0.6055239 ]]   9th feature vector
```

five features per vector

matrix notation

each row is an observed feature vector
each column is a separate feature

scikit-learn

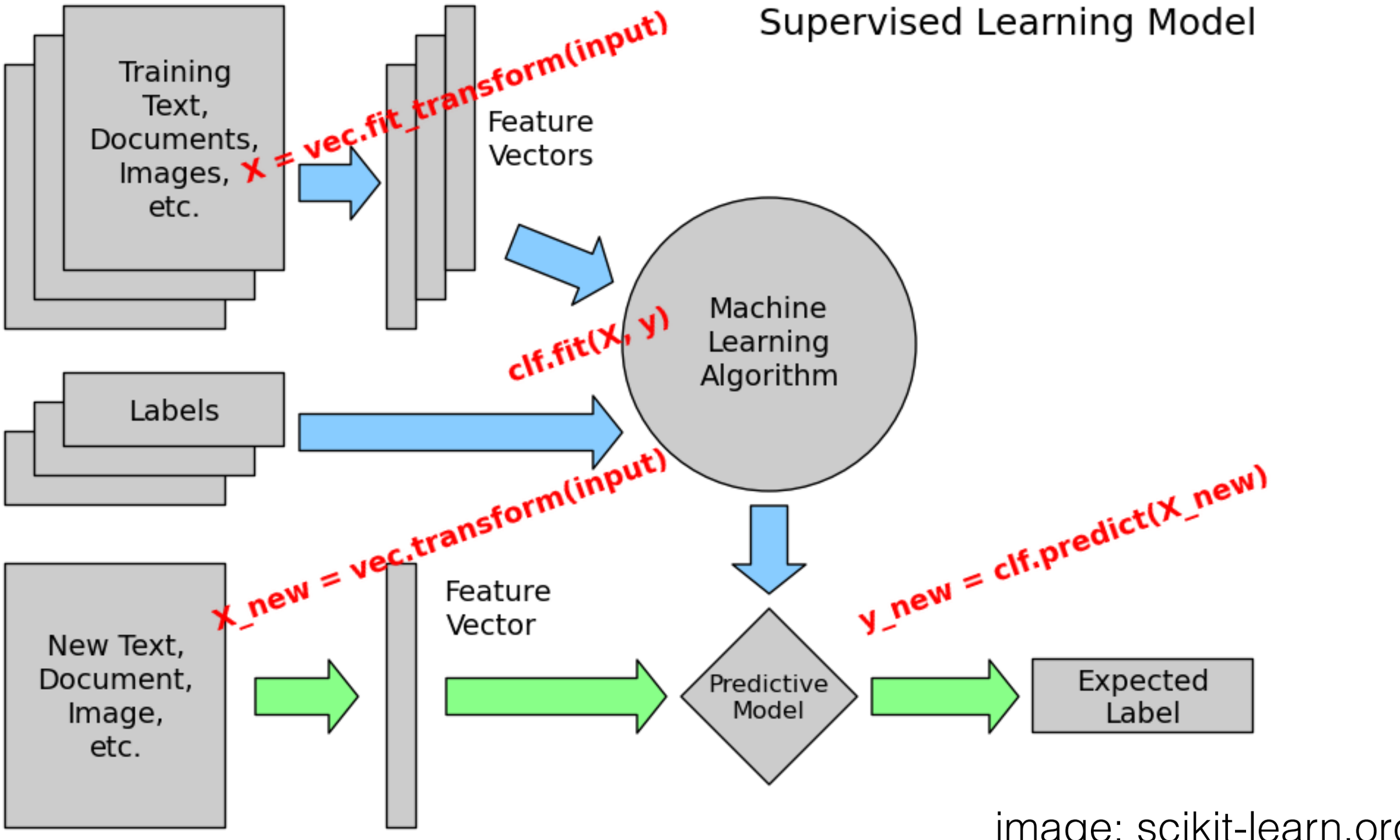


image: scikit-learn.org

scikit-learn

- many datasets are included

```
from sklearn import datasets

# load the data
iris_ds = datasets.load_iris()
featuredata = iris_ds.data;
target      = iris_ds.target;
```

```
>>> iris_ds.data
array([[ 5.1,  3.5,  1.4,  0.2],
       [ 4.9,  3. ,  1.4,  0.2],
       [ 4.7,  3.2,  1.3,  0.2],
       [ 4.6,  3.1,  1.5,  0.2],
       [ 5. ,  3.6,  1.4,  0.2],
       [ 5.4,  3.9,  1.7,  0.4],
       [ 4.6,  3.4,  1.4,  0.3],
                  ...
```

```
from sklearn import datasets
from sklearn import preprocessing

# load the data
iris_ds = datasets.load_iris()
featuredata = iris_ds.data;
target      = iris_ds.target;

featuredata = preprocessing.scale(featuredata)
```

```
>>> iris_ds.target
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
              ...
```

classifiers

- get a classifier and train it

```
>>> from sklearn import tree
>>> clf = tree.DecisionTreeClassifier()

>>> clf.fit(iris_ds.data,iris_ds.target)
DecisionTreeClassifier(compute_importances=None, criterion='gini',
                       max_depth=None, max_features=None, min_density=None,
                       min_samples_leaf=1, min_samples_split=2, random_state=None,
                       splitter='best')

>>> predicted = clf.predict(iris_ds.data)

>>> from sklearn.metrics import accuracy_score
>>> accuracy_score(iris_ds.target,predicted)
1.0
```

this is called re-substitution

but we should be using a validation set

scikit-learn demo

- lets look at some interesting scikit learn examples
  - loading and testing datasets
  - data transformation
  - visualizing
  - training and prediction
- more examples and an intro demo
  - http://scikit-learn.org/stable/tutorial/basic/tutorial.html

for next time…

- predicting on the phone