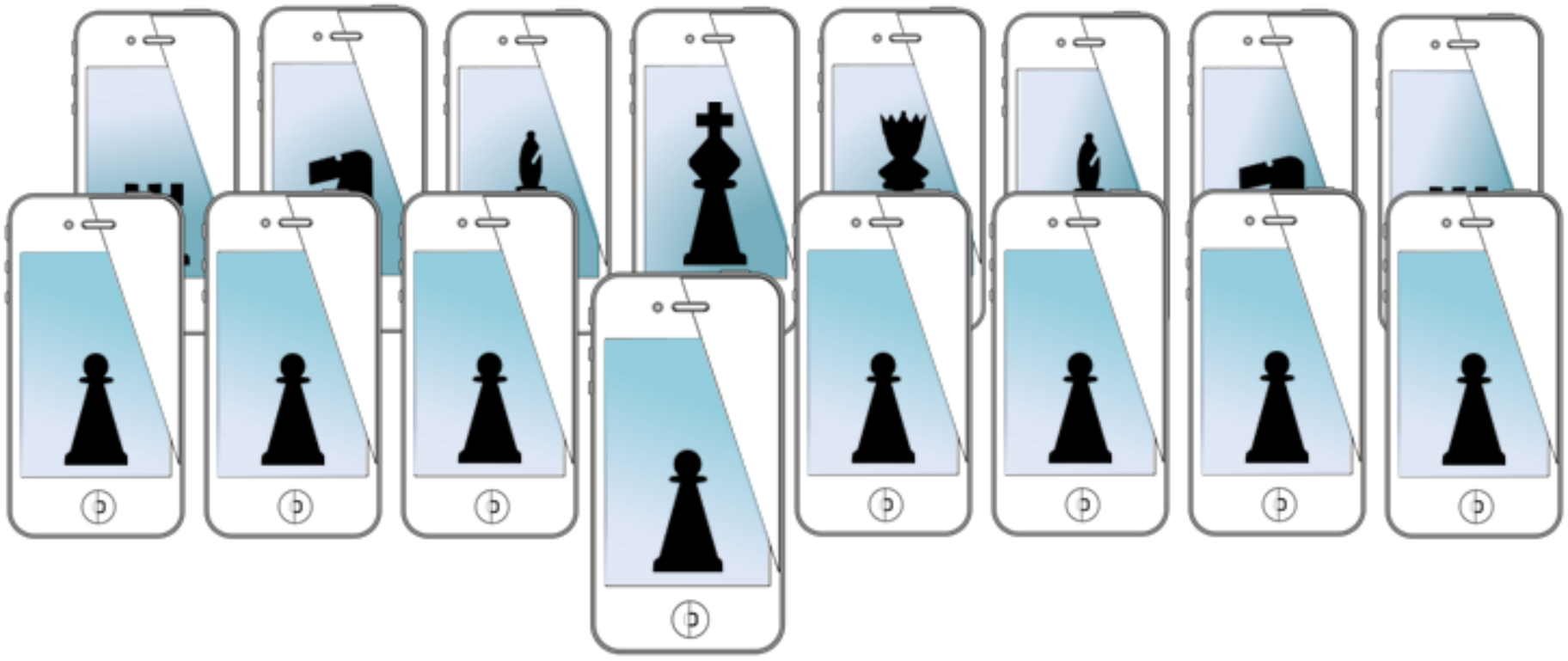
MOBILE SENSING LEARNING



CSE5323 & 7323

Mobile Sensing and Learning

week 10: tornado, pymongo, and http requests

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

course logistics

- A5 is due Friday
- start to think about the final project proposal

agenda

- finish tornado
- mongodb
- http requests in iOS

what are we doing?

- **preparing for A6**, need HTTP server that can:
  - accept (any) data
  - save it into a database
  - learn a (ML) model from that database
  - mediate queries and training of the model
- tornado is the event-driven architecture for interpreting the commands, routing the data, etc.
- our focus is building a deployment server, not an advanced ML algorithm (take DM or ML courses for that)

sub-classing application

```
# tornado imports
import tornado.web
from tornado.web import HTTPError
from tornado.httpserver import HTTPServer
from tornado.ioloop import IOLoop
from tornado.options import define, options

# Setup information for tornado class
define("port", default=8000,
       help="run on the given port", ty

CUSTOM CLASSES AND DEFINITIONS

def main():
    '''Create server, begin IOLoop
    '''
    tornado.options.parse_command_line()
    http_server = HTTPServer(Application(), xheaders=True)
    http_server.listen(options.port)
    IOLoop.instance().start()

if __name__ == "__main__":
    main()
```

custom class for handling requests

sub-classing application

```
# Utility to be used when creating the Tornado server
# Contains the handlers and the database connection
class Application(tornado.web.Application):
    def __init__(self):
        '''Store necessary handlers,
           connect to database
        '''

        handlers = [(r"/[/]?",
                     BaseHandler),
                    (r"/Test[/]?",
                        examplehandlers.TestHandler),
                    (r"/DoPost[/]?",
                    MORE HANDLERS AND URL PATHS
                    ]

        settings = {'debug':True}
        tornado.web.Application.__init__(self, handlers, **settings)

        SETUP DATABASE

    def __exit__(self):
        self.client.close()
```

I wrote the base handler for you

handlers should subclass it

more to come in a moment

post versus get

Compare GET vs. POST

The following table compares the two HTTP methods: GET and POST.

	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure compared to POST because data sent is part of the URL Never use GET when sending passwords or other sensitive information!	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

credit: w3schools.com

BaseHandler Demo

- check out what it does
  - built for analyzing and writing back json
  - implements both get and post requests
  - put this in the main python file to access these:

```
# custom imports
from basehandler import BaseHandler
import examplehandlers
```

post versus get

- if we are sending data to a server for processing
  - of unknown length
  - of many different formats
  - possibly in a multi-part file
- should we use post or get requests?
- why?

post requests

- identical handling code in python
- in our implementation, return json

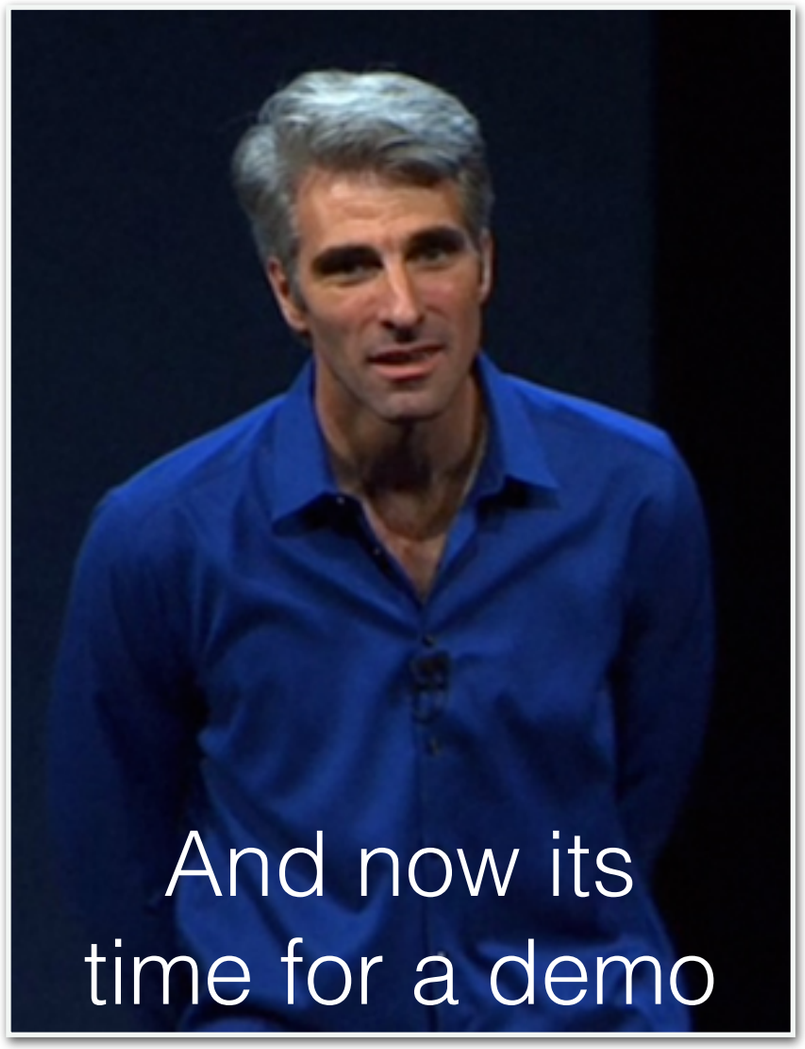
convenience function I wrote for you!

```
class PostHandler(BaseHandler):
  def get(self):
   '''respond with arg1*2
   '''
   arg1 = self.get_float_arg("arg1",default="none");
   self.write("Get from Post Handler? " + str(arg1*2));

  def post(self):
   '''Respond with arg1 and arg1*4
   '''
   arg1 = self.get_float_arg("arg1",default=1.0);
   self.write_json({"arg1":arg1,"arg2":4*arg1});
```

convenience function I wrote for you!

tornado examples

- with everything up except the database
- note that quick database queries are “okay” to block on



mongodb

- hu**mongo**us data
- NoSQL database (vs relational database)
  - >document database
- everything stored as a document
  - more or less json
  - key: value/array
- schema is dynamic
  - the key advantage of NoSQL

mongodb install

- install it
- http://www.mongodb.org/downloads
- to run single server database:
  - make a directory for the db
    - like data/db
  - run mongodb
    - ./mongod --dbpath “<path to db>”
  - its running! localhost

make sure this exists!

mongodb

- a document, as stated by mongodb

Document Database

A record in MongoDB is a document, which is a data structure composed of field and value pairs. MongoDB documents are similar to JSON objects. The values of fields may include other documents, arrays, and arrays of documents.

A MongoDB document.

The advantages of using documents are:

- Documents (i.e. objects) correspond to native data types in many programming languages.
- Embedded documents and arrays reduce need for expensive joins.
- Dynamic schema supports fluent polymorphism.

docs and collections

Database: MSLC_creations

limit on document size: ***16MB***

apps_collection

```
{
  app: "mongoApp",
  users: 100005,
}

{
  app: “StepCount",
  users: 45,
  rating: 2.6,
}
....

{
  app: "shattner",
  users: 4050000,
  rating: 5,
}
```

teams_collection

```
{
  team: "mongo",
  members: [ "Eric", “Ringo”, “Paul” ],
  numApps: 21,
  website: “teammongo.org",
}
{
  team: “ran off",
  members: [ "John", “Yoko” ],
  website: “flewthecoop.org",
}
....
{
  team: "shattner",
  members: [ "Bill", “Will”, “Tom” ],
  numApps: 1,
  website: “shattner.com”,
}
```

pymongo

- python wrapper for using mongo db

```
client  = MongoClient() # localhost, default port
db = client.some_database # access database
```

create this database, if it does not exist

```
collect = client.some_database.some_collection # access a collection
```

relational equivalent of a table

create this database, if it does not exist

nothing is **created** until the first insert!!!

```
db.collection_names()
[u'system.indexes', u'some_collection']
```

get collections

pymongo

- insertion

```
dbid = db.some_collect.insert(
    {“key1”:values,"key2":more_values,
     “coolkey”:with_cool_values}
);
```

unique key, _id

- update

```
db.some_collect.update({"thiskey":keyValue},
    {  "$set": {"keyToSet":valueToSet}  },
    upsert=True)
```

where ever this key is…

equal to this

set

this key to this value

insert if it does not exist

pymongo

- find one datum in database

```
a = db.some_collect.find_one(sort=[("sortOnThisKey", -1)])
newData = float( a['sortOnThisKey'] );
```

its a list!

access the result

sort with this key

last element

- loop through all results

```
f=[];
for a in db.some_collect.find({"keyIWant":valueOfKeyIWant}):
    f.append( str(a['keyToGrabDataWith']) )
```

- lots of advanced queries are possible

https://api.mongodb.org/python/current/

teams example

```
>>> from pymongo import MongoClient
>>> client = MongoClient()

>>> db = client.some_database
>>> collect1 = db.some_collection
>>> collect1.insert({"team":"TeamFit","members":["Matt","Mark","Rita","Gavin"]})
ObjectId('53396a80291ebb9a796a8af1')

>>> db.collection_names()
[u'system.indexes', u'some_collection']

>>> db.some_collection.find_one()
{u'_id': ObjectId('53396a80291ebb9a796a8af1'), u'members': [u'Matt', u'Mark', u'Rita', u'Gavin'],
u'team': u'TeamFit'}

>>> collect1.insert({"team":"Underscore","members":["Carly","Lauryn","Cameron"]})
ObjectId('53396c80291ebb9a796a8af2')

>>> db.some_collection.find_one()
{u'_id': ObjectId('53396a80291ebb9a796a8af1'), u'members': [u'Matt', u'Mark', u'Rita', u'Gavin'],
u'team': u'TeamFit'}

>>> db.some_collection.find_one({"team":"Underscore"})
{u'_id': ObjectId('53396c80291ebb9a796a8af2'), u'members': [u'Carly', u'Lauryn', u'Cameron'],
u'team': u'Underscore'}
```

bulk operations

```
from pymongo import MongoClient

client = MongoClient()
db=client.some_database
collect1 = db.some_collection

insert_list = [{"team":"MCVW","members":["Matt","Rowdy","Jason"]},
               {“team":"CHC", "members":["Hunter","Chelsea","Conner"]}]
obj_ids=collect1.insert(insert_list)
```

```
for document in collect1.find({"members":"Matt"}):
  print(document)
```

```
{u'_id': ObjectId('53396a80291ebb9a796a8af1'), u'members': [u'Matt', u'Mark', u'Rita', u'Gavin'], u'team': u'TeamFit'}
{u'_id': ObjectId('53397331291ebb9afdd3cd2f'), u'members': [u'Matt', u'Rowdy', u'Jason'], u'team': u'MCVW'}
```

```
document = collect1.find_one({"members":"Matt","team":"MCVW"})
print (document)
```

```
{u'_id': ObjectId('53397331291ebb9afdd3cd2f'), u'members': [u'Matt', u'Rowdy', u'Jason'], u'team': u'MCVW'}
```

mongodb and binary data

- want to store binary data more than 16MB?
- use `gridfs`, its real simple
- use `put()` and `get()` instead of `insert()` and `find()`
  - `get()` returns a “file-like” object, so you can read in chunks

```
> from pymongo import MongoClient
> import gridfs
> db = MongoClient().gridfs_ex
> fs = gridfs.GridFS(db)

> a = fs.put("hello world”)
> fs.get(a).read()
'hello world'
```

```
> b = fs.put(fs.get(a),
       filename="foo", bar="baz")
> out = fs.get(b)
> out.read()          'hello world'
> out.filename          u'foo'
> out.bar               u'baz'
> out.upload_date
datetime.datetime(...)
```

http://api.mongodb.com/python/current/examples/gridfs.html

mongodb and binary data

- want to store binary data more than 16MB?
- use `gridfs`, its real simple
- use `put()` and `get()` instead of `insert()` and `find()`
  - `get()` returns a “file-like” object, so you can read in chunks

```
for grid_out in fs.find({"filename": "foo.txt"},
                        no_cursor_timeout=True):
    data = grid_out.read()


most_recent_three = fs.find().sort(
                        "uploadDate", -1).limit(3)
```

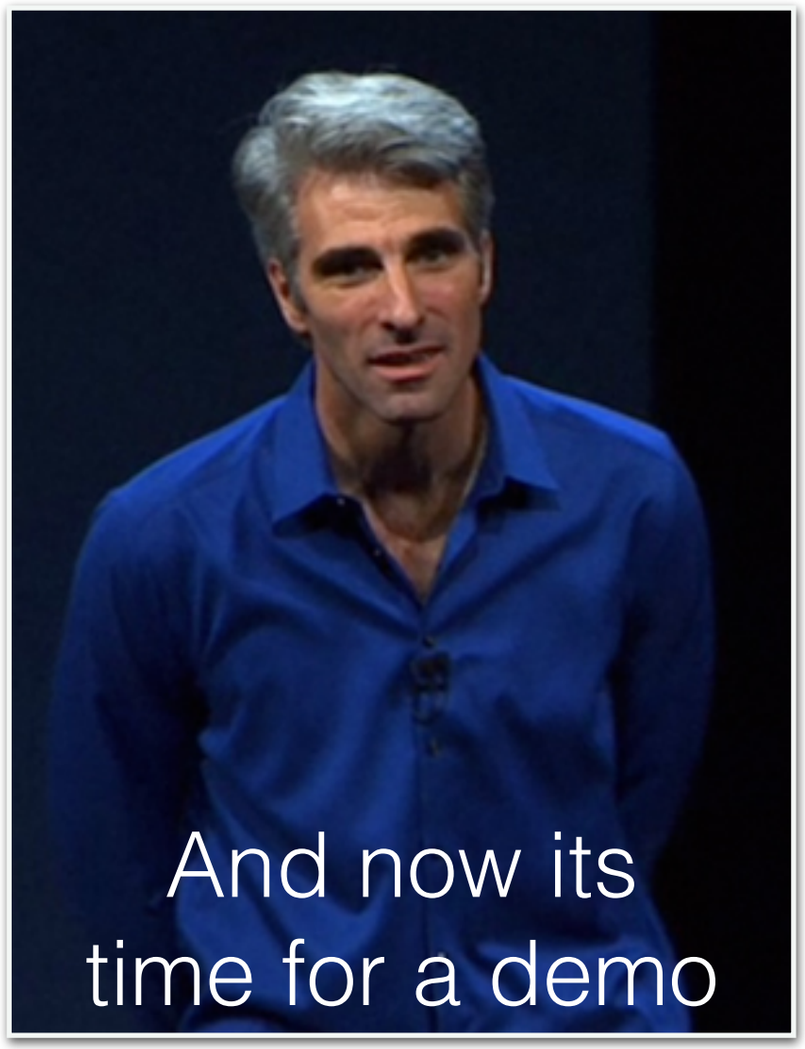
http://api.mongodb.com/python/current/examples/gridfs.html

mongodb + tornado

- we will use pymongo and tornado
  - mongodb runs localhost, tornado mediates access
  - good for learning
    - but real product would need load balancing (nginx)
- a non-blocking mongodb + tornado (motor)
  - in alpha: https://github.com/mongodb/motor
  - have not used yet, but looks good

mongodb + tornado

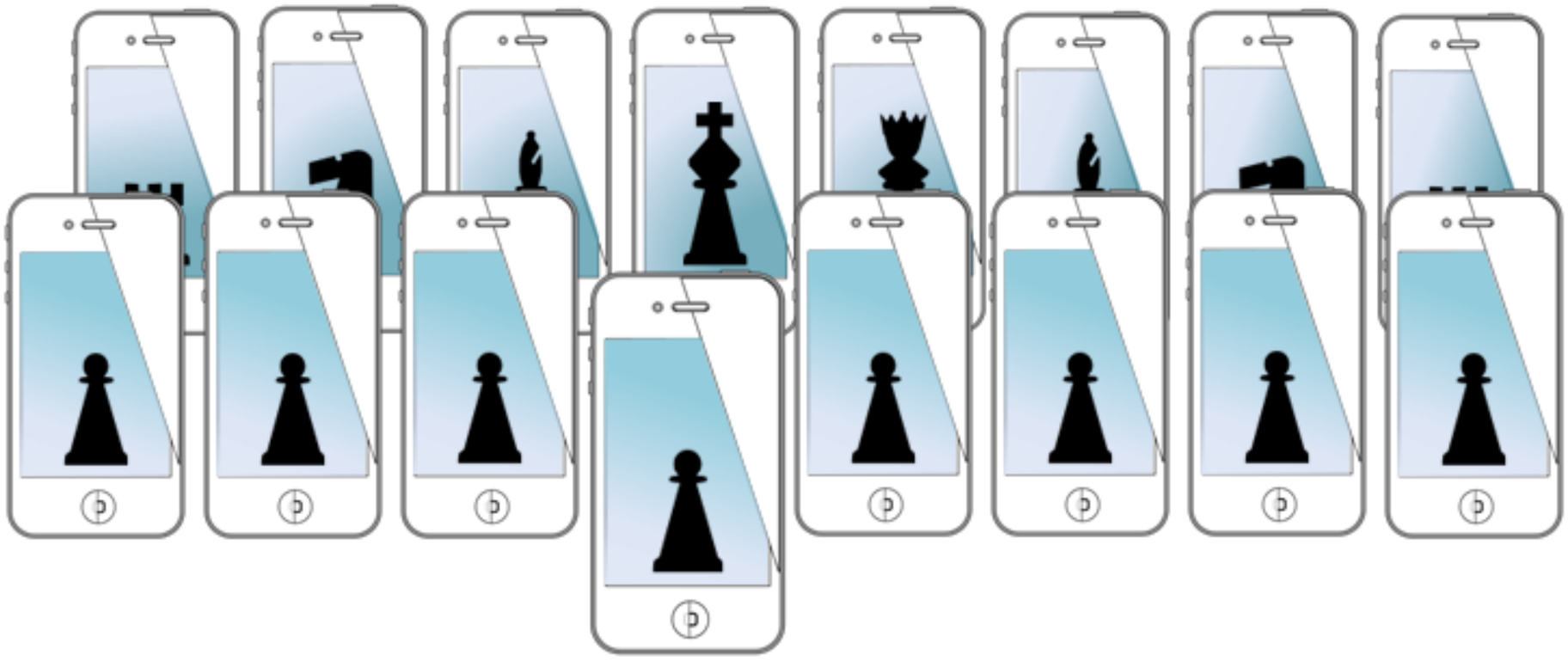
- demo:
  - store data inside mongodb with each http request



and add **something** to it

- ifconfig | grep "inet "

MOBILE SENSING LEARNING



CSE5323 & 7323

Mobile Sensing and Learning

week 10: tornado, pymongo, and http requests

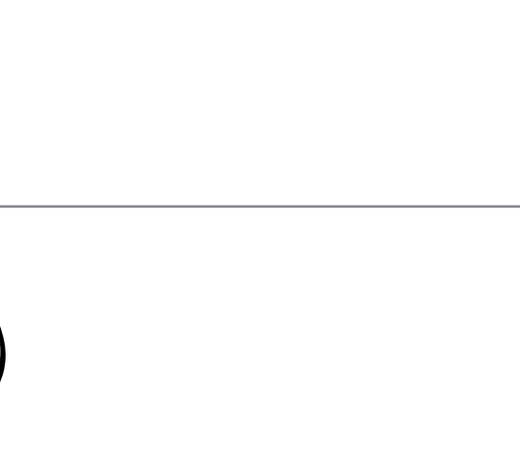
Eric C. Larson, Lyle School of Engineering,
Computer Science and Engineering, Southern Methodist University

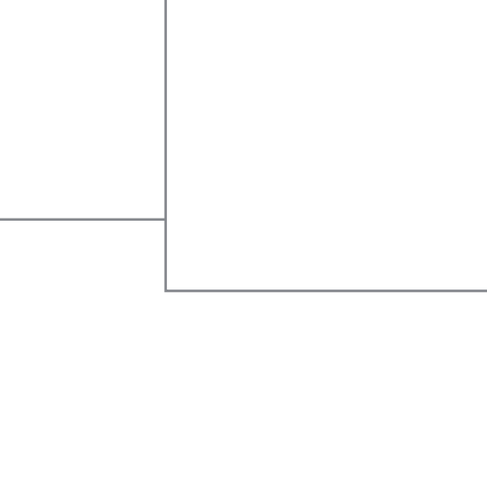
course logistics

- A5 is due Friday
- start to think about the final project proposal

agenda

- *finish tornado (done!)*
- *mongodb (done!)*
- http requests in iOS
- project proposals





working with your web server

- we want to send data to our hosted server!
  - or any server for that matter
- need to form POST and GET requests from iOS
- we will use NSURLSession

NSURLSession

- proper way to configure a session with a server
- new format starting in iOS7
  - old way was to use `NSURLConnection`
  - before that was to use `sendAsynchronousRequest`
- you may see code for `initWithContentsOfURL`:
  - **never, never, never** use that for networking
- sessions are a huge improvement in iOS
  - and extremely powerful
  - the Stanford course talks about these (check it out)!
  - as promised, we will cover different topics than Stanford course

NSURLSession

- delegate model
- does authentication if you need it!
  - we won't use that though — who would hack our server?
- implements pause / resume
- can setup download tasks as locally saved files
  - nice for separating the data and metadata

configure a session

```
@interface SMUViewController () <NSURLSessionTaskDelegate>

@property (strong,nonatomic) NSURLSession *session;

@end
```

delegation

will reuse session

```
//setup NSURLSession (ephemeral)
NSURLSessionConfiguration *sessionConfig =
    [NSURLSessionConfiguration ephemeralSessionConfiguration];
```

e·phem·er·al
/əˈfem(ə)rəl/

adjective

1. lasting for a very short time.
   "fashions are ephemeral"
   synonyms: transitory, transient, fleeting, passing, short-lived, momentary, brief, short; More

noun

this is private!!

- do not cache
- no cookies
- do not store credentials

background queue

configure a session

- other options:

`ephemeralSessionConfiguration`

`defaultSessionConfiguration`

use global cache, cookies, and credential storage objects

`backgroundSessionConfiguration`

make my session respond to push notifications, launch my app, if needed, handle download completion

configure a task

- tasks are common types of requests
- tied to a session
- give a way to specify URL and type of request
- will format data in specific ways to handle response from server

- we will use a “completion handler”
  - more involved downloads allow use of delegates
    - progress indicators
    - completion indicators

NSURLSessionDataTask

- common to use for GET requests
- uses blocks for completion — or could use… ?

```
dataTaskWithURL:completionHandler:

 NSURL *getUrl = [NSURL URLWithString: [baseURL stringByAppendingString:query]];

 NSURLSessionDataTask *dataTask = [self.session dataTaskWithURL:getUrl
                   completionHandler:^(NSData *data,
                                       NSURLResponse *response,
                                       NSError *error) {
                       NSLog(@"%@",response);
                       NSLog(@"%@",[[NSString alloc] initWithData: data
                                                         encoding:NSUTF8StringEncoding]);

                   }];
      [dataTask resume]; // start the task
```

NSString NSString

convert bytes to string

must call, or stays suspended

NSURLDownloadTask

- sub-class of NSURLSessionDataTask
- many delegate methods for getting progress

```
    NSURLSessionDownloadTask *downloadTask = [self.session downloadTaskWithURL:[NSURL
URLWithString:@"someurlfordownloadingimages.com/coolimage"]
          completionHandler:^(NSURL *location, NSURLResponse *response, NSError *error) {
                if(!error)
                {
                    UIImage *img = [UIImage imageWithData:[NSData dataWithContentsOfURL:location]];
                }
            }];
    [downloadTask resume];
```

could use delegate instead of completion handler

```
-(void)URLSession:(NSURLSession *)session
     downloadTask:(NSURLSessionDownloadTask *)downloadTask
didFinishDownloadingToURL:(NSURL *)location

-(void)URLSession:(NSURLSession *)session
     downloadTask:(NSURLSessionDownloadTask *)downloadTask
     didWriteData:(int64_t)bytesWritten
totalBytesWritten:(int64_t)totalBytesWritten
totalBytesExpectedToWrite:(int64_t)totalBytesExpectedToWrite
```

NSURLUploadTask

- common to use for POST requests
- need to setup your own HTTP request (not just URL)

```
uploadTaskWithRequest:fromData:completionHandler



// create a custom HTTP POST request
NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:postUrl];
[request setHTTPMethod:@“POST"];

NSData *imageData = UIImageJPEGRepresentation(image, 0.25);

NSURLSessionUploadTask *uploadTask =
[self.session uploadTaskWithRequest:request
                           fromData:imageData
          completionHandler:^(NSData *data, NSURLResponse *response, NSError *error) {

          }];
```

could be any data

could use delegate methods

NSURLDataTask

- can also use this for PUT or POST requests
- highly similar to uploadTask
  - but you don't get the delegate methods for progress

```
// create a custom HTTP POST request
NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:postUrl];
[request setHTTPMethod:@"POST"];

NSData *imageData = UIImageJPEGRepresentation(image, 0.25);
[request setHTTPBody:imageData];

NSURLSessionDataTask *dataTask =
[self.session dataTaskWithRequest:request
        completionHandler:^(NSData *data, NSURLResponse *response, NSError *error) {

         }];
```

JSON serialization

- parse in tornado

```
import json

class JSONPostHandler(BaseHandler):
   def post(self):
    '''Parse some posted data
    '''
     data = json.loads(self.request.body)
```

- parse in iOS

```
NSDictionary *responseData = [NSJSONSerialization JSONObjectWithData:data
                              options: NSJSONReadingMutableContainers
                                error: &error];
```

the output in both scenarios is a dictionary

NSDictionary

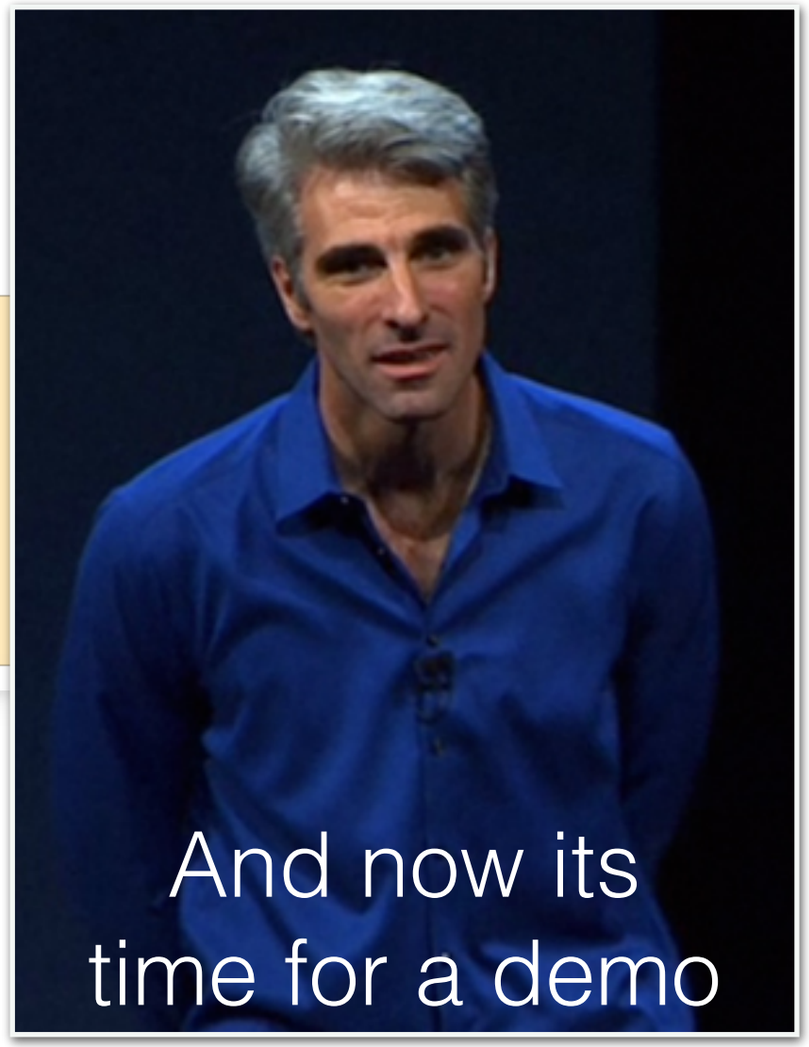
- serialize in iOS

```
NSData *requestBody=[NSJSONSerialization dataWithJSONObject:jsonUpload
                                 options:NSJSONWritingPrettyPrinted
                                   error:&error];
```

tornado + iOS demo

- send a GET request, handle query in tornado
- do POST with GET-like query
- do POST with JSON in, JSON out

we just learned objective c syntax, but..
I rewrote the examples in Swift!



before talking about proposals for next time…

- next time: basics of machine learning
  - machine learning as a service
    - install scikit-learn (already there with anaconda)
- next next time: **in-class-assignment** with our own restful API
  - using ML and networking to do cool things…
  - some code has changed since filming, but mostly the same

project proposal

Final Project Proposal

All final projects should be approved by the instructor via the final project proposal. This is a description explaining the overall idea, which labs the project builds from, and a list of four or more design constraints that the design will meet. Turn in the project proposal via canvas or talk it over with the instructor.

If your group is opting into the "mother of all demos" (see explanation below) your proposal must specify this (you can opt out later, but you cannot opt in). Note that MOD projects should have more difficult constraints. The instructor may tighten constraints for those that wish to opt into the MOD.

exceptional final project example

JukeboxHero

We will build an iOS application that works as a guitar pedal. We are aiming for users to be musicians who want play along and practice songs from their music library. The phone will allow a “Jam Session.” Users connect their guitar and listen via headphones or external speakers. It will have five different guitar effects including distortion and chorus. Some effects will have varying levels of tuning (*i.e.*, the amount of distortion).

The app will play songs from the user’s music library and will consult our web server for information about the songs (like the tuning of the guitar, so the user can play along). It will allow the users to tune their guitar to the song being played. We will allow control over the effects through CoreMotion. For example, moving the phone with your foot controls a predefined progression of different effects.

An example of the closest app on the app store is <here>.

The constraints to be met in the project are:

1. Recognition of a “foot” gesture. We will validate our recognition by teaching the instructor to “select” through the foot gesture, then allowing the instructor to use the “foot select” to change effects. We will have at least 90% recognition.
2. Five different effects will be implemented. At least two will be tunable. We will evaluate by showing a progression of effects through external speakers and connecting a guitar during the final demonstration.
3. We will have a tuner in the application for tuning the strings of a guitar. We will demonstrate this with a guitar during the final demonstration. We will also show the output of a real tuner, for reference. The phone tuner and the purchased tuner will be no more than 1 Hz apart.
4. last constraint — the web server. **What might this look like?**

past final projects 2014

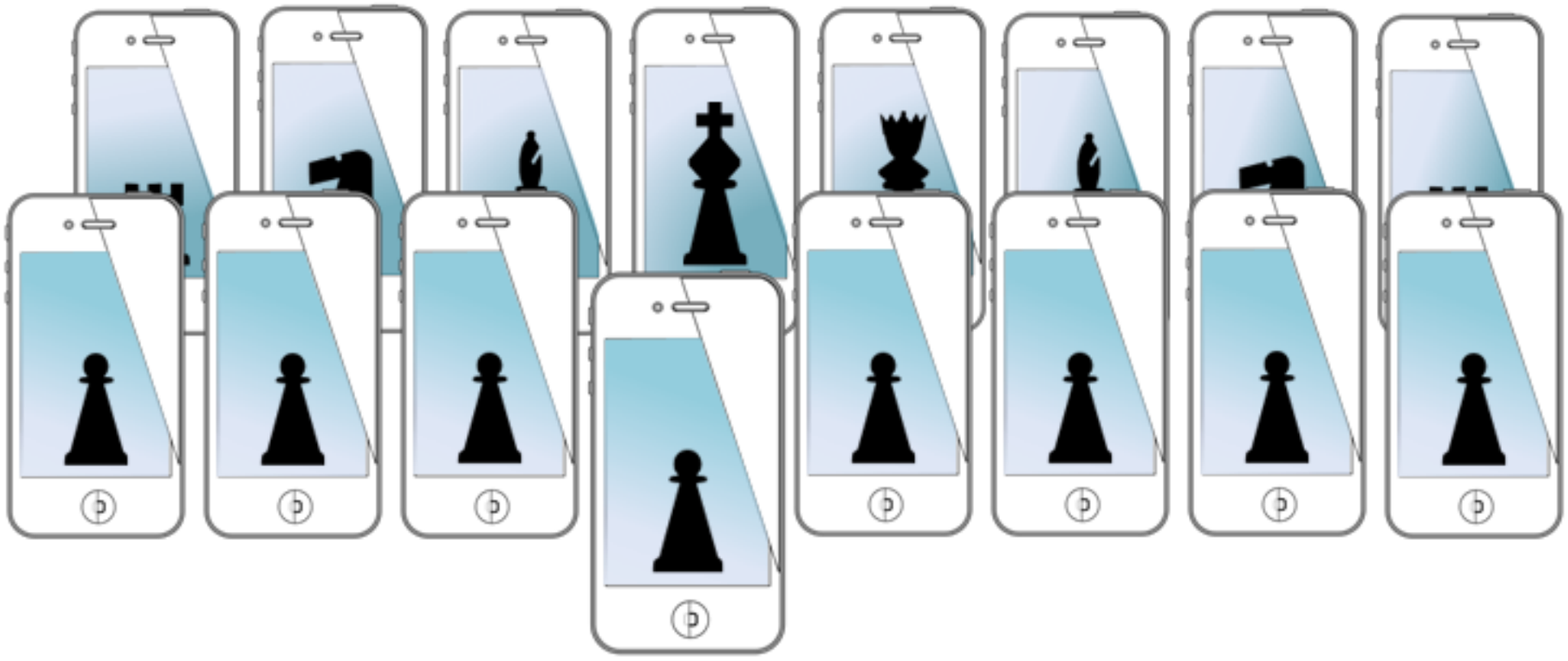
past final projects 2014

past final projects 2015

past final projects 2015

past final projects 2016

MOBILE SENSING LEARNING



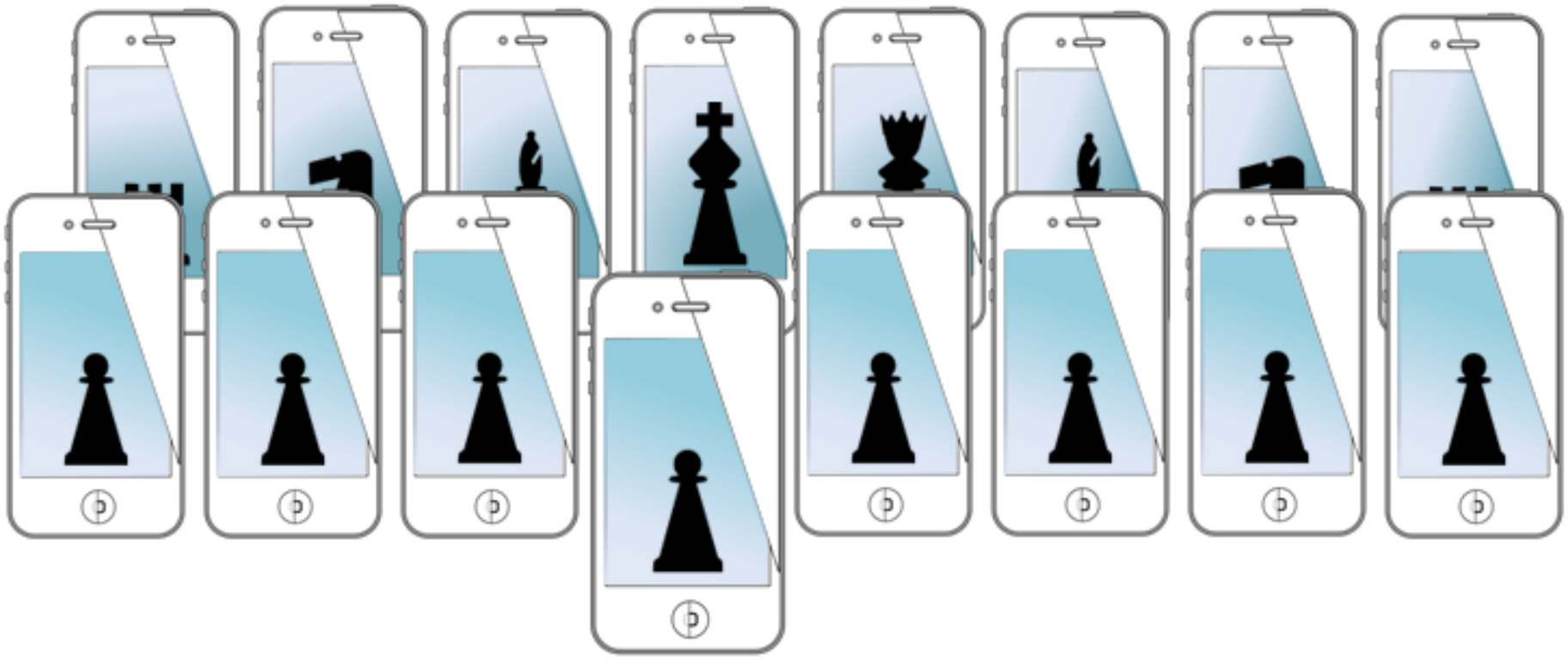
CSE5323 & 7323

Mobile Sensing and Learning

week 10: tornado, pymongo, and http requests

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

MOBILE SENSING LEARNING



CSE5323 & 7323

Mobile Sensing and Learning

week 11a: machine learning crash course

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

course logistics

- **A6** is due next week
  - 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
  - either draft or talk to me
- there are some changes to the schedule!!
  - canvas…

assignment 6

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

Assignment Six - Machine Learning as a Service

Module A

Create an iOS application using the HTTPExample that:

- Collects some form of low throughput (sampling rate > 1s) **feature data** for processing, audio, video, motion, or from the arduino
- 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 or AWS or whatever
- Trains a model from the labeled data (***e.g.***, KNN)
- **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 form of model. Verify the functionality of the application to the instructor during lab time or office hours (or scheduled via email).

assignment 6

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

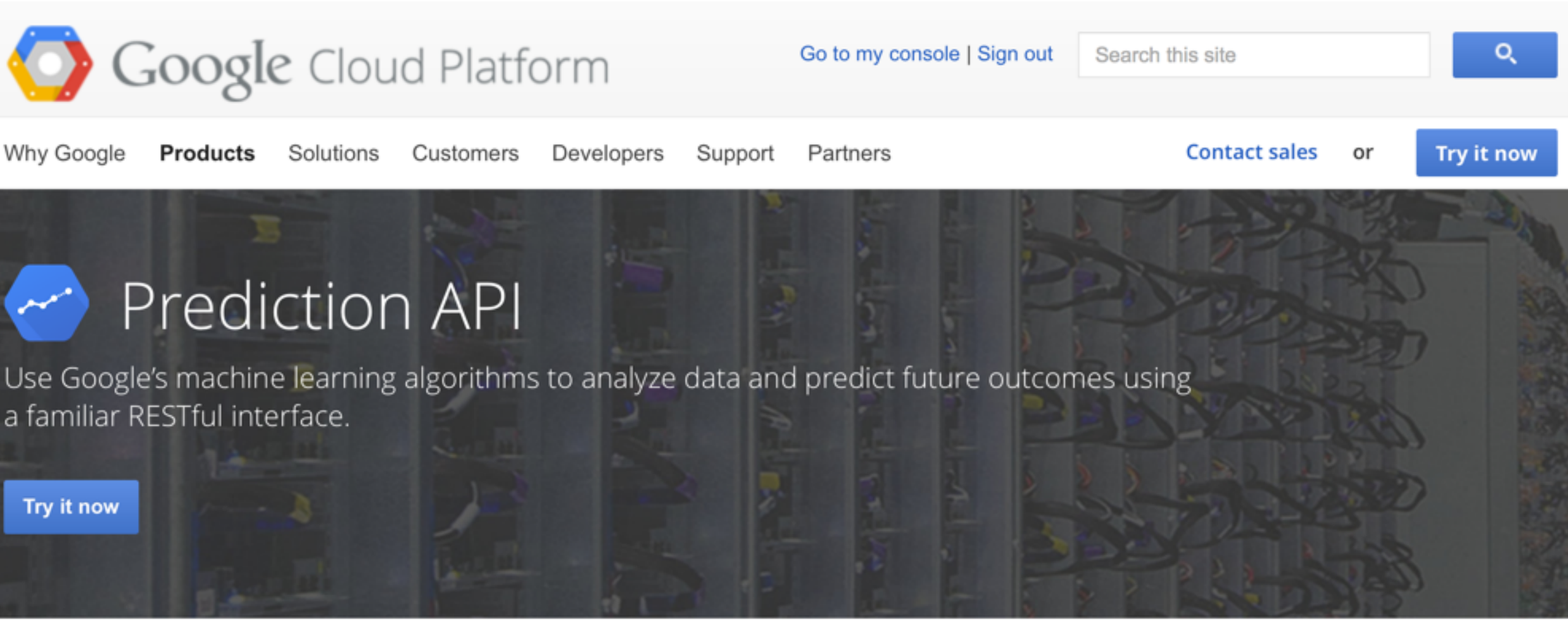
Assignment Six - Machine Learning as a Service

Module B

Update the HTTPExample and the tornado web server to:

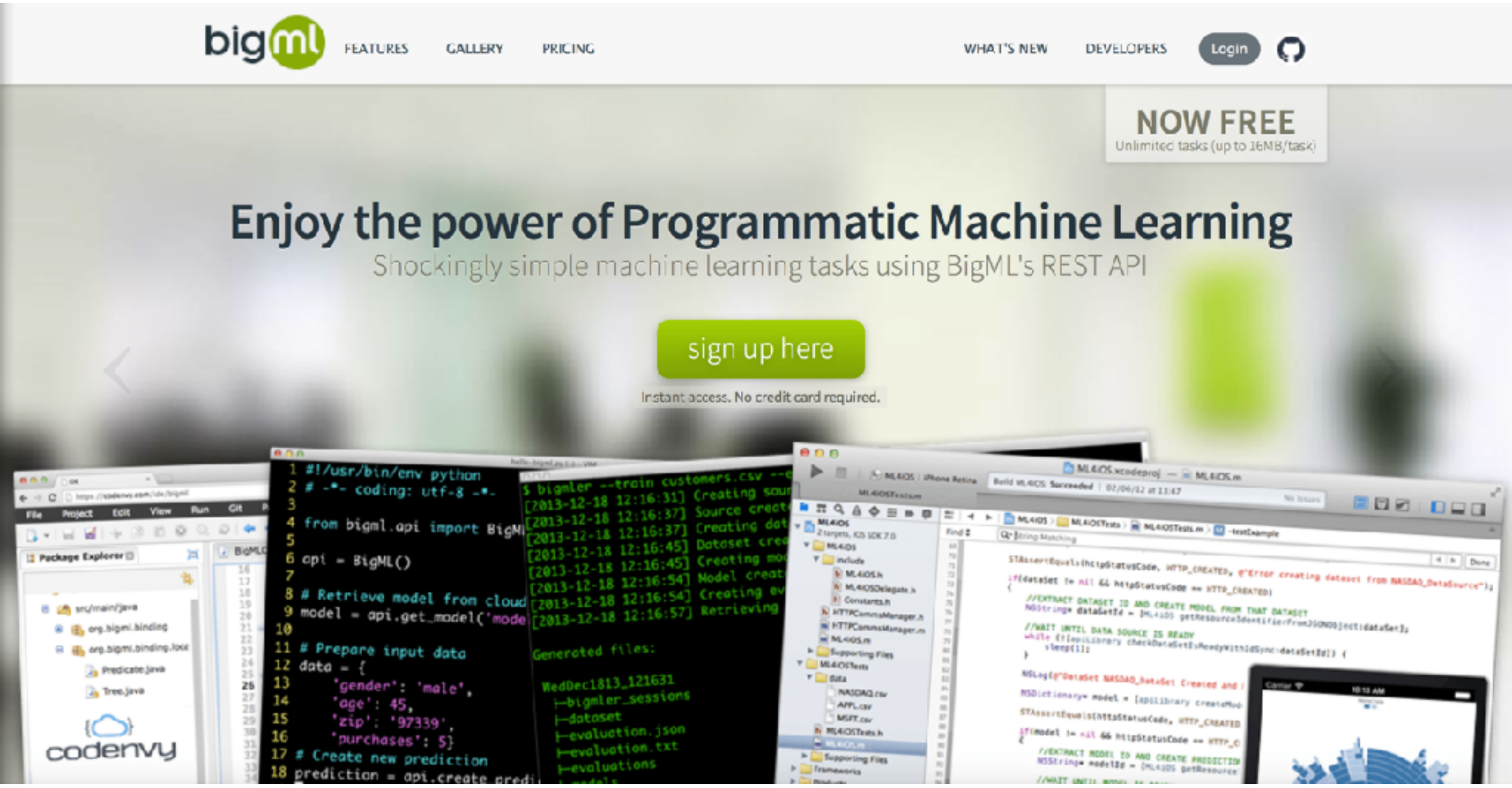
- Specify the type of model to use in the Machine Learning
  - at least **two different types** of machine learning models (***e.g.***, SVM and KNN)
- Compare the efficacy of two or more different models
  - **send parameters** to use in the machine learning models from the phone (***e.g.***, number of neighbors to use in KNN)
- An option for exceptional work:
  - 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

machine learning



machine learning

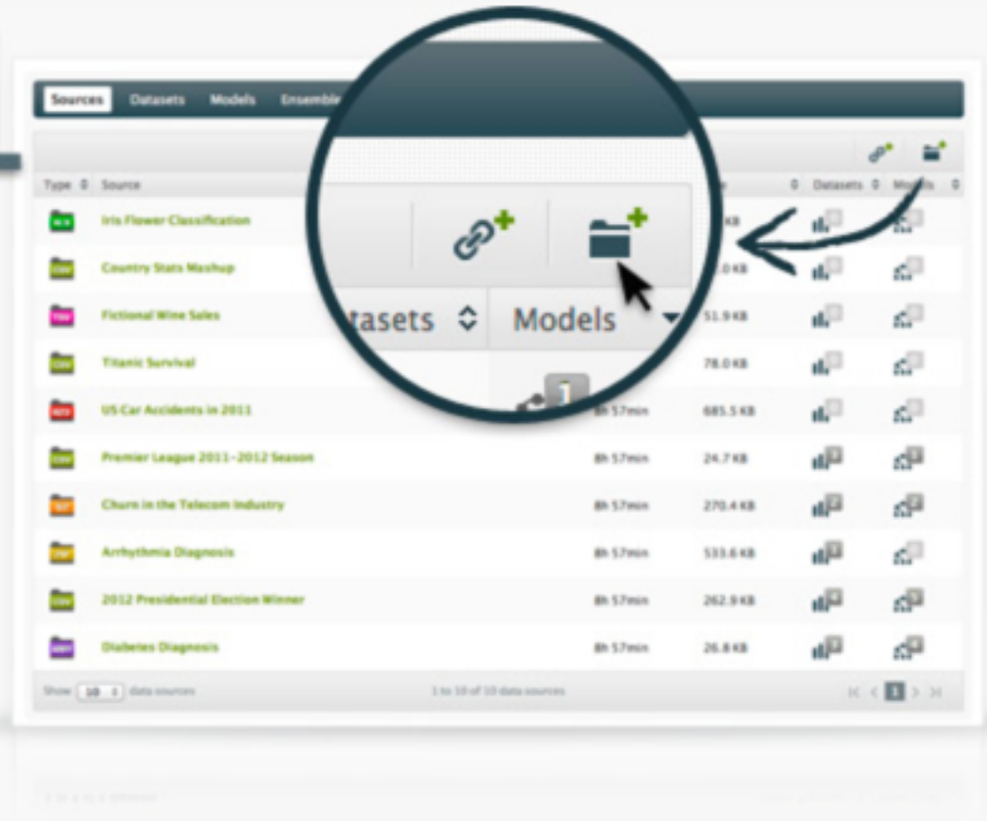
machine learning



machine learning

BigML

1



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

BigML

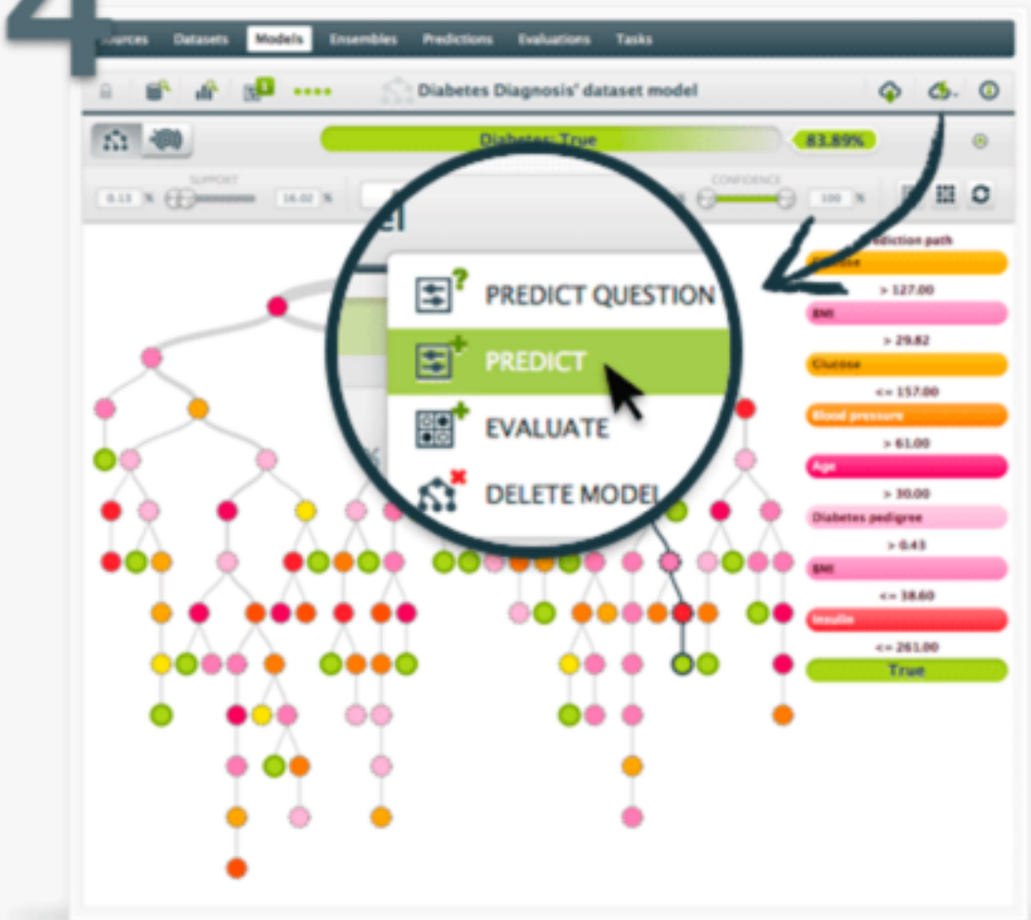
Create a Dataset

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

BigML

4



Generate Predictions

To create a prediction, you can either use an automatically generated web form or a question by question interface. Just click the **predict** or **predict question** buttons and you'll be presented with the corresponding interfaces. When the number of input fields is big or you want to automate the generation of predictions you can use our API. Stay tuned for our upcoming High Performance Prediction Servers that will allow you to generate thousands of predictions per second.

How to make predictions

BigML

Create a Model

To create a model, just use the **1-click model** from a dataset view if you want to use all active fields and instances in your dataset to generate a model that predicts (i.e., has as **objective field**) the last column in the input data. You can also use a **configure model** panel to select a different **objective field**, specific fields, or sample your dataset using multitude of options. Your dataset will be processed to build a predictive model that will not only show you the most relevant patterns in your data but also will allow you to generate predictions for new data instances.

How to create a model

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 use 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

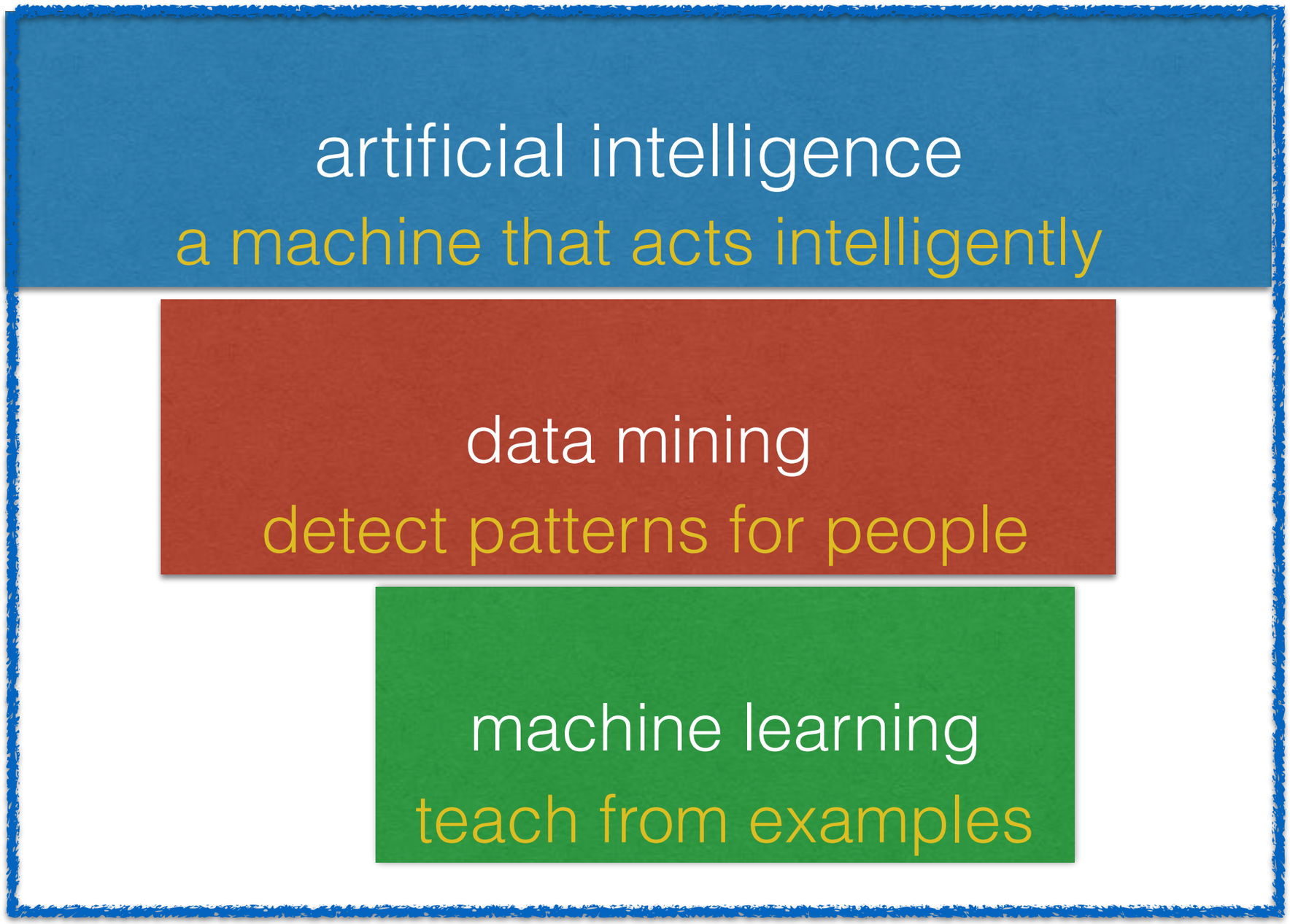
/ˈlərniNG/

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



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

- create a model
- cluster data
- help me visualize

- create a model
- predict something

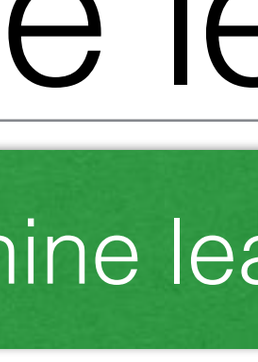
machine learning

machine learning

- what examples can be used to teach a machine?
  - features
  - labels
  - predictions

what we know

what we want to know



labels: like like like dislike

like

some features: genre
actors

adolescent male?

features

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

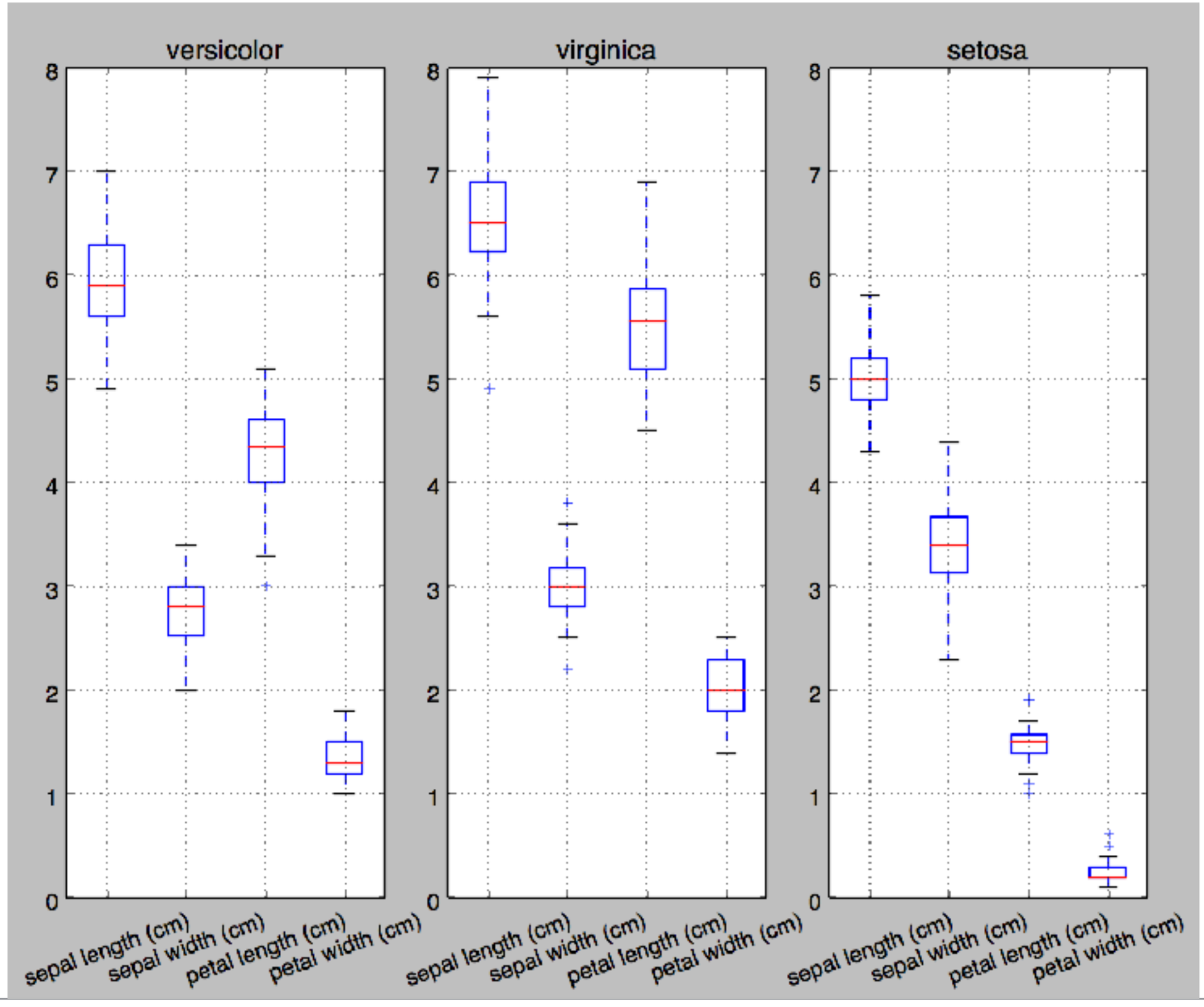
setosa versicolor virginica

- 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
```

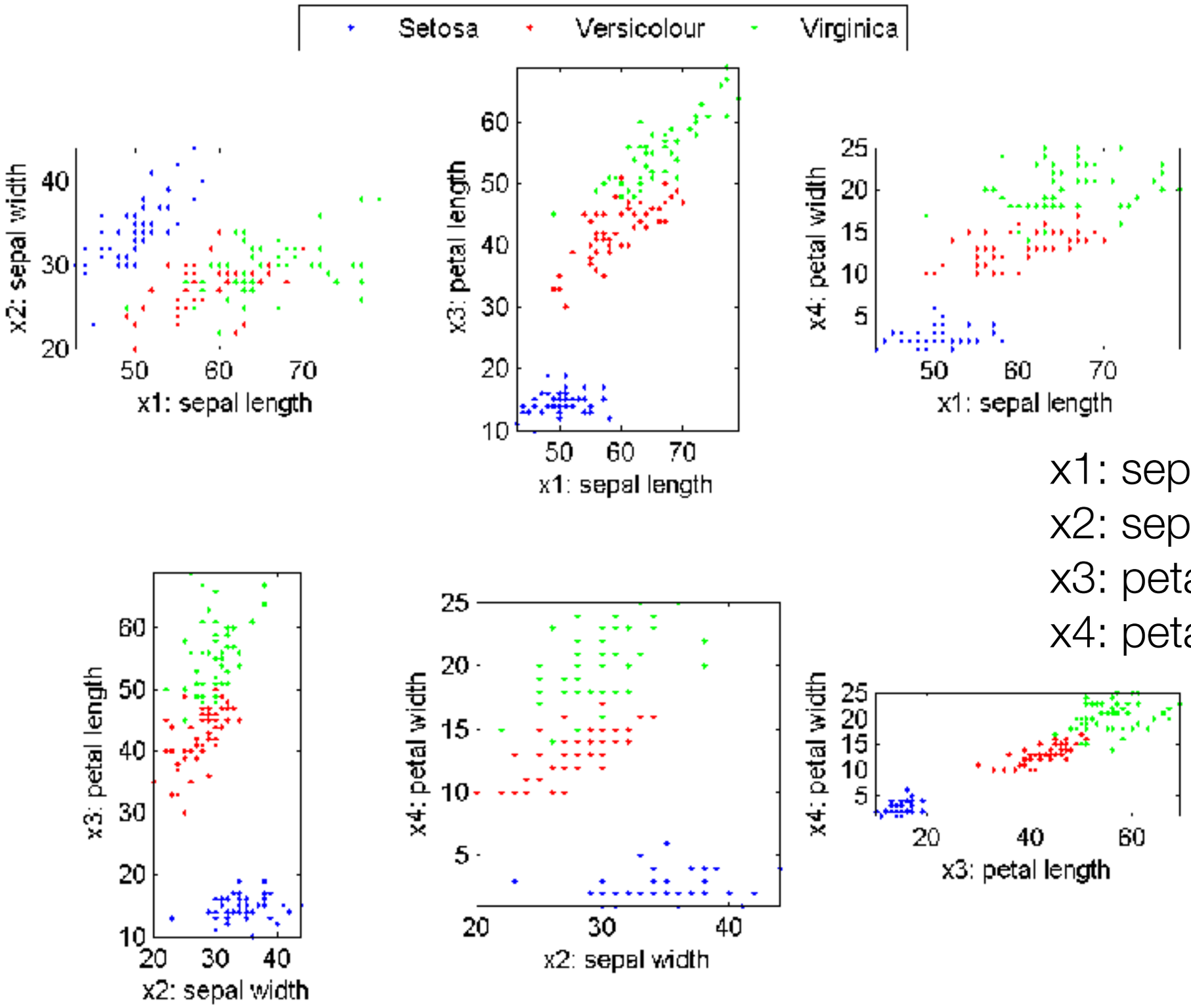
50 examples each

visualizing features



separability

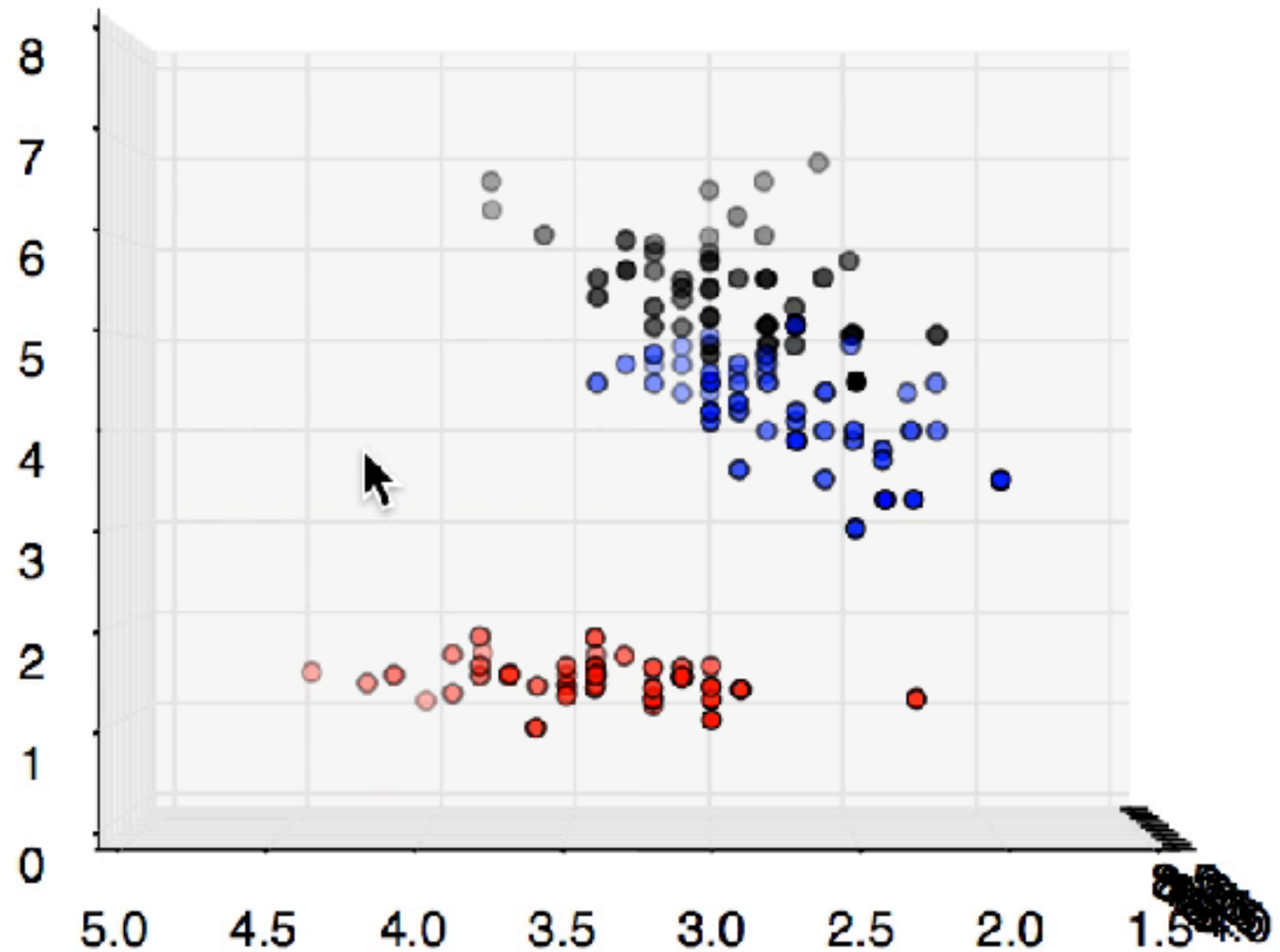
visualize features



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

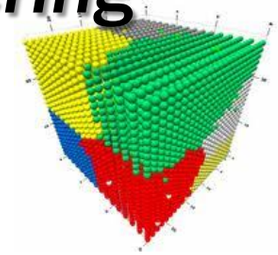
- most common is numeric
- vector quantization
- bag of words
  - term frequency: percentage of a times a word appears in a type of document
  - inverse document frequency
- graphs
  - used to quantize

take data mining!
or python machine learning!

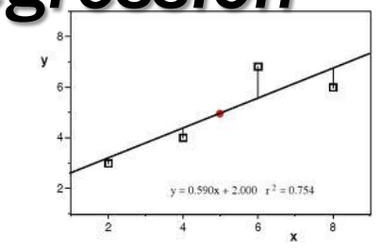
types of machine learning

- 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
- logistic regression
- neural networks
- gaussian mixtures

- support vector machines
- and many many more…

decision trees

- remember the iris data set?

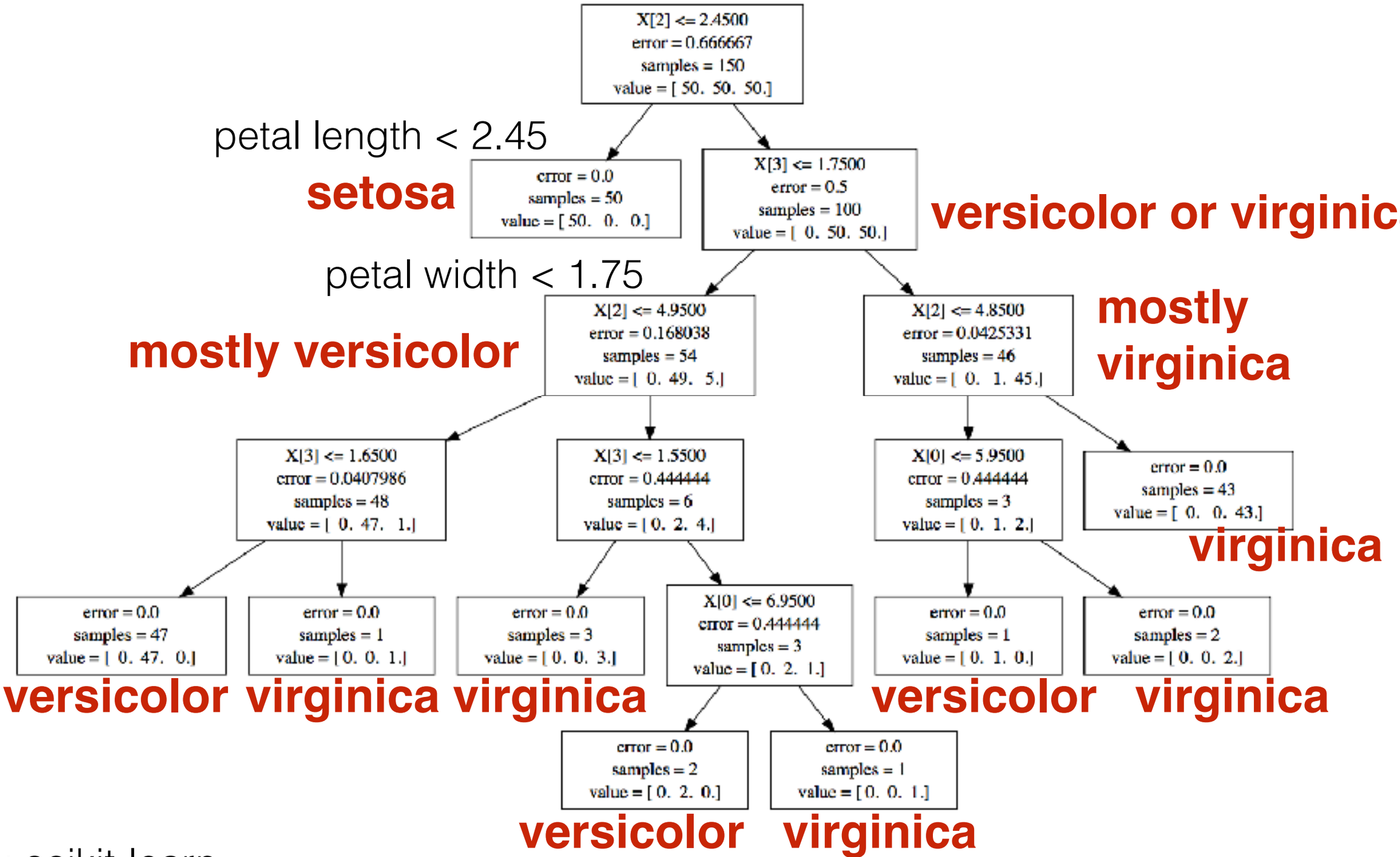
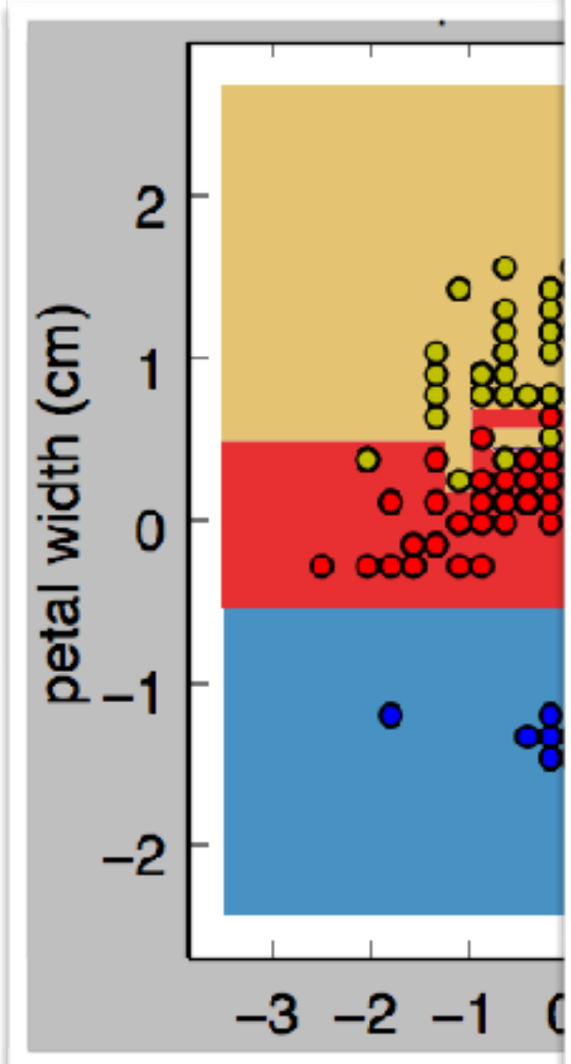
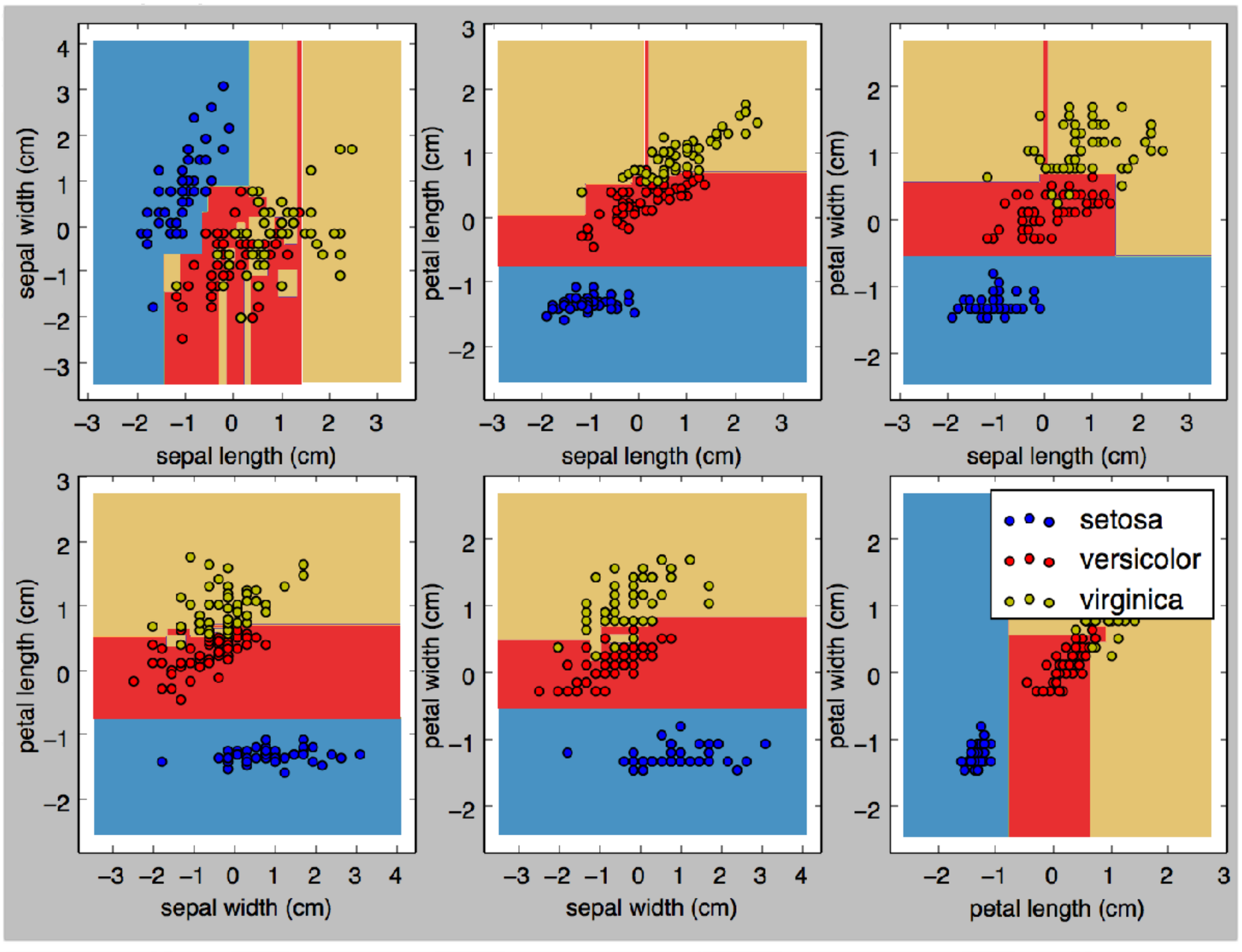


image: scikit-learn

decision tree

- decision

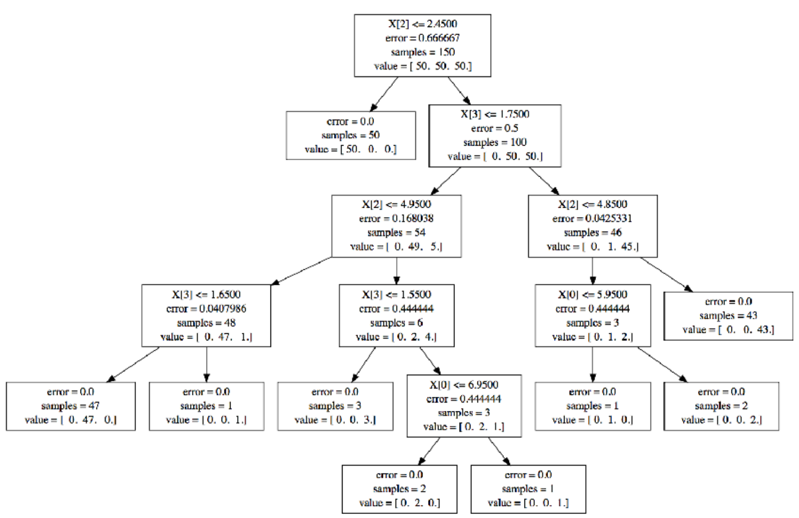




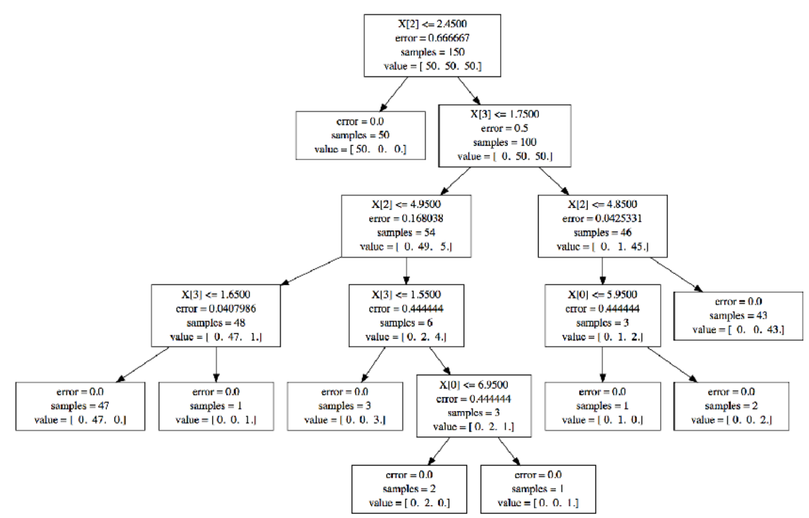
random forests

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

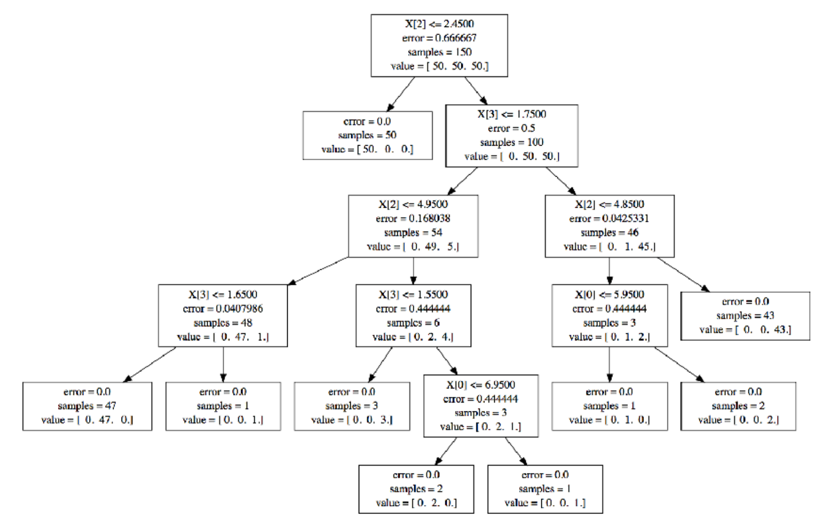
120 examples
3 features



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

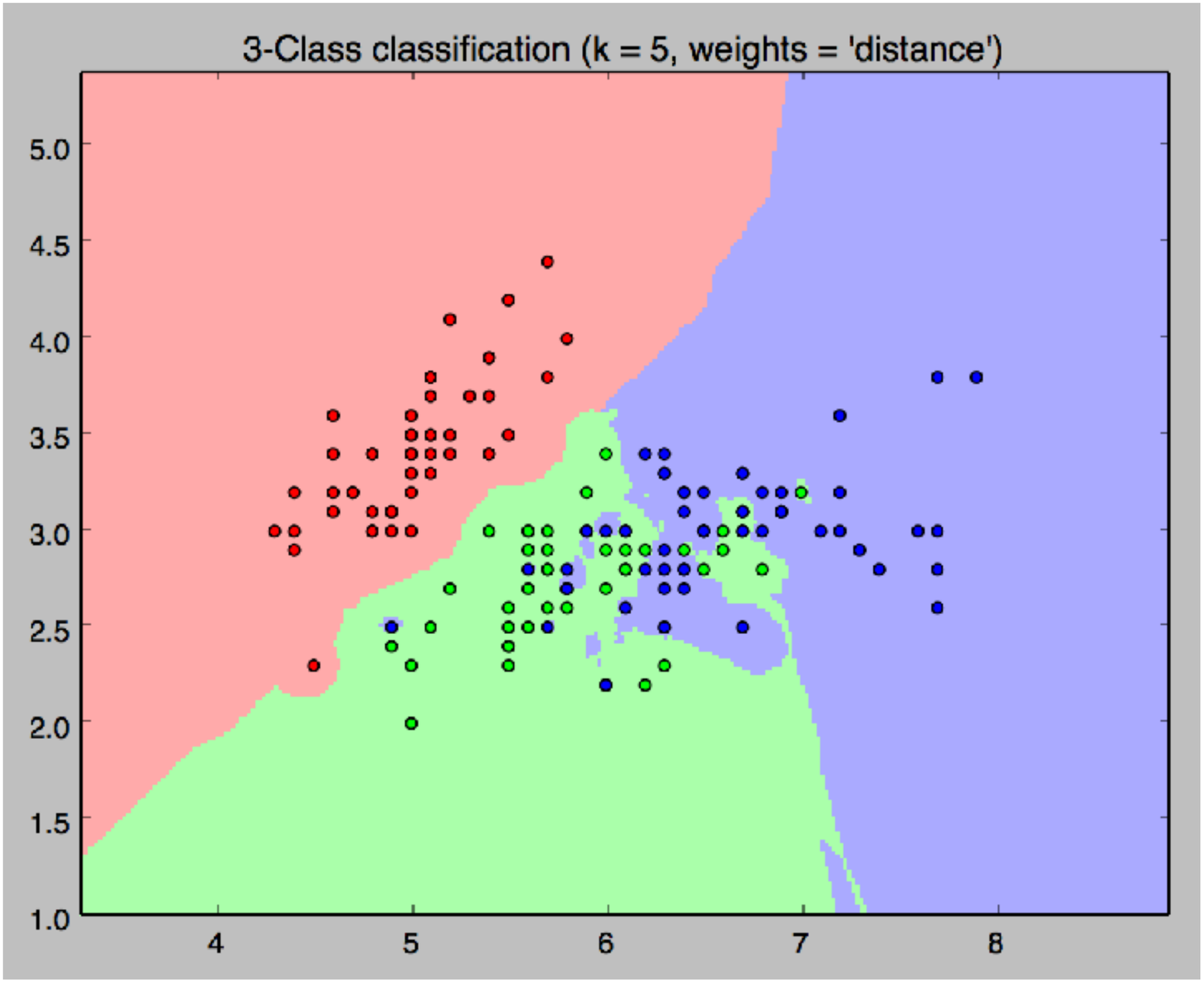
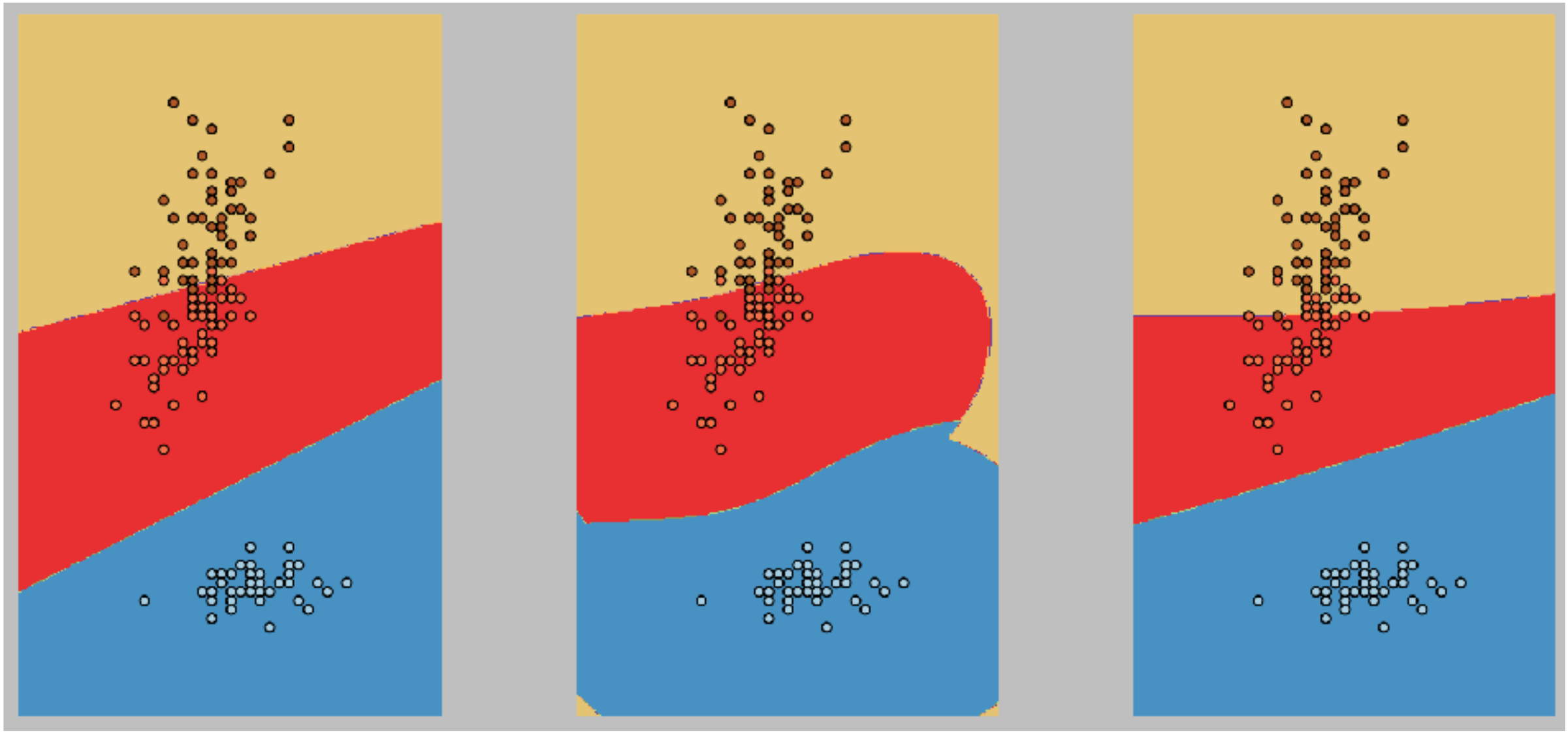


image:
datasciencerules.org

support vector machines

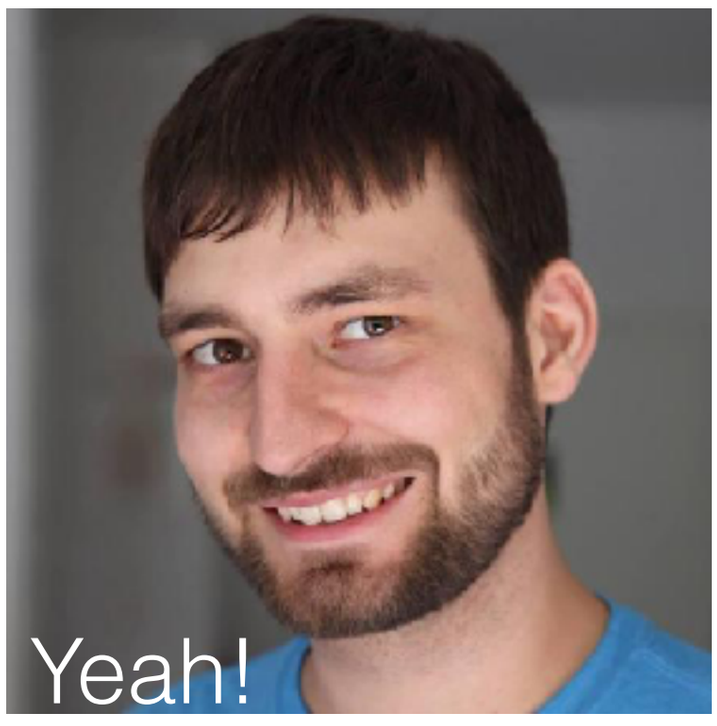
- find a vector that separates the training data

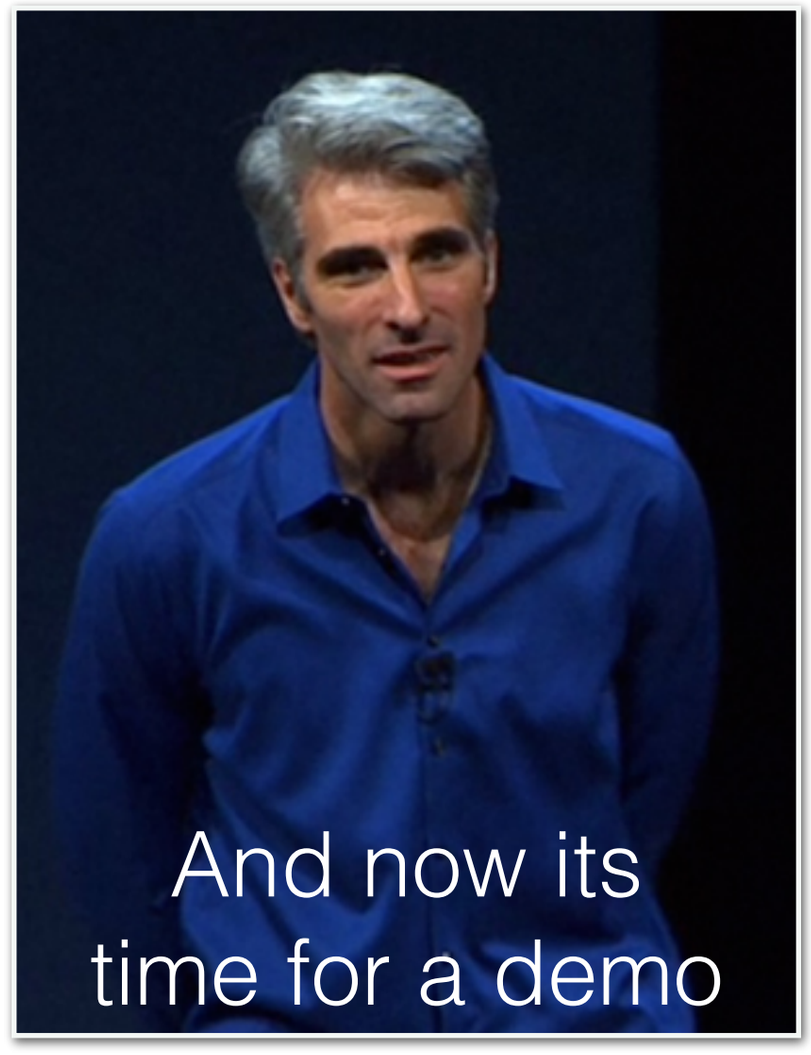


linear vector gaussians sigmoid

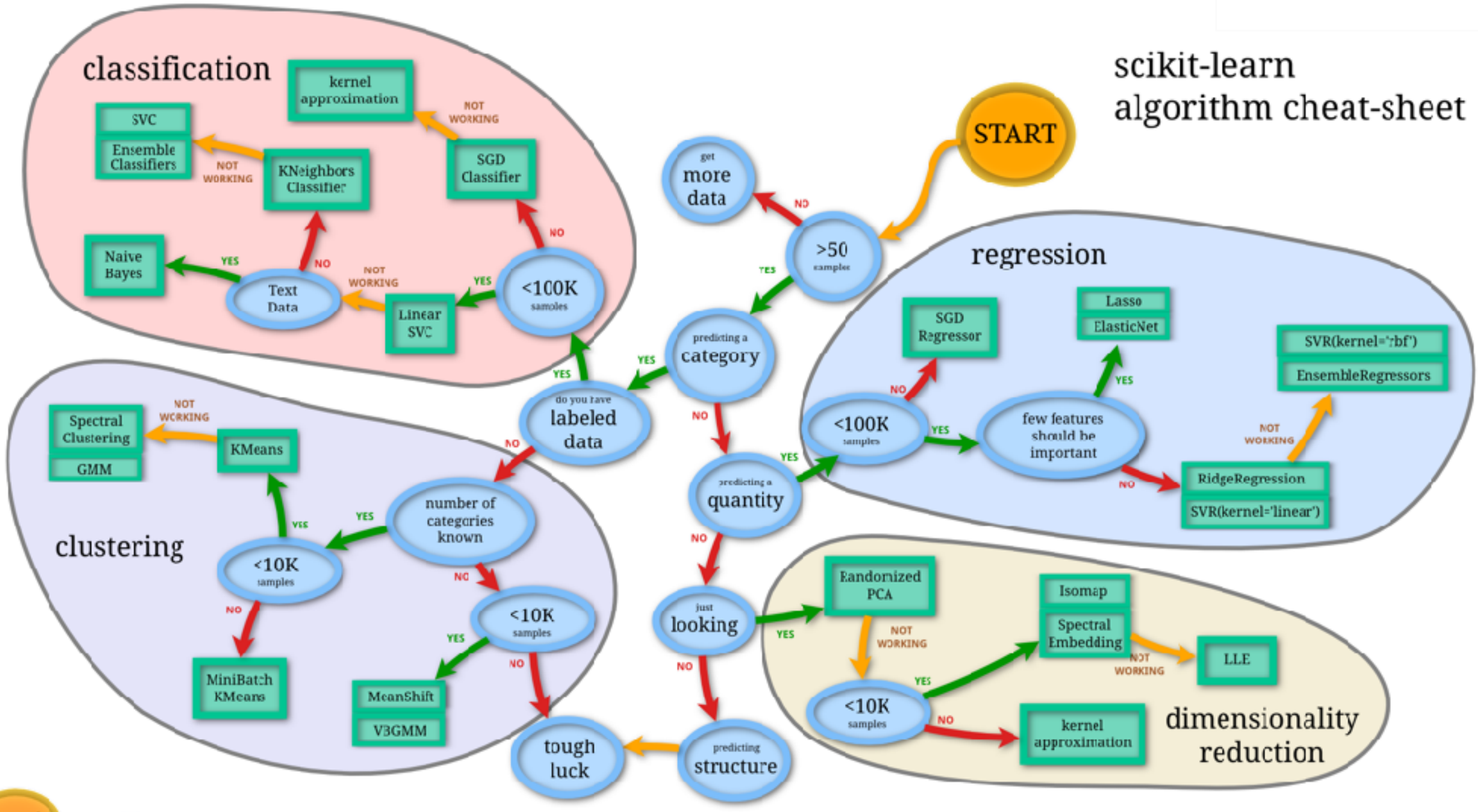
SVM

- demo of support vectors
- scikit-learn GUI





my data?



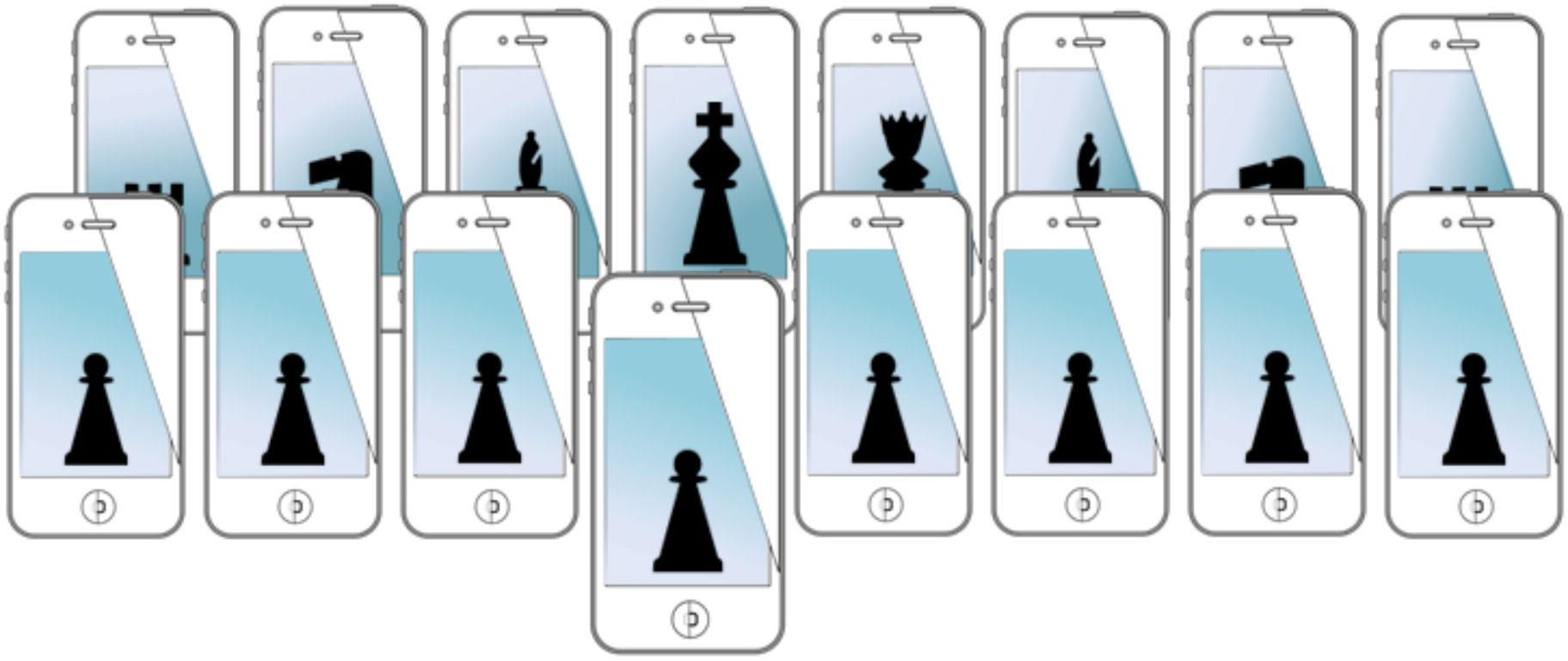
Back

image: Andreas Muller

for next time…

- flipped module for using scikit-learn 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



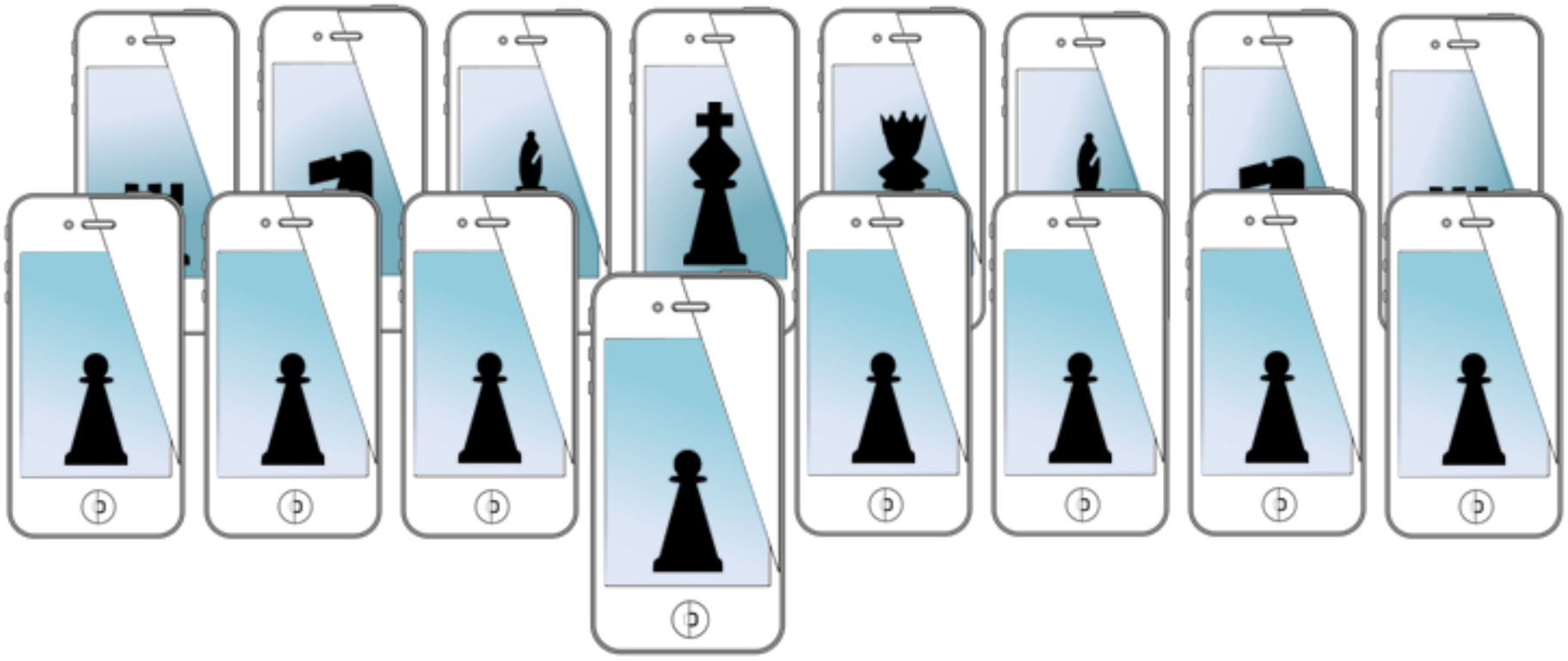
CSE5323 & 7323

Mobile Sensing and Learning

week 11a: machine learning

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

MOBILE SENSING LEARNING



CSE5323 & 7323

Mobile Sensing and Learning

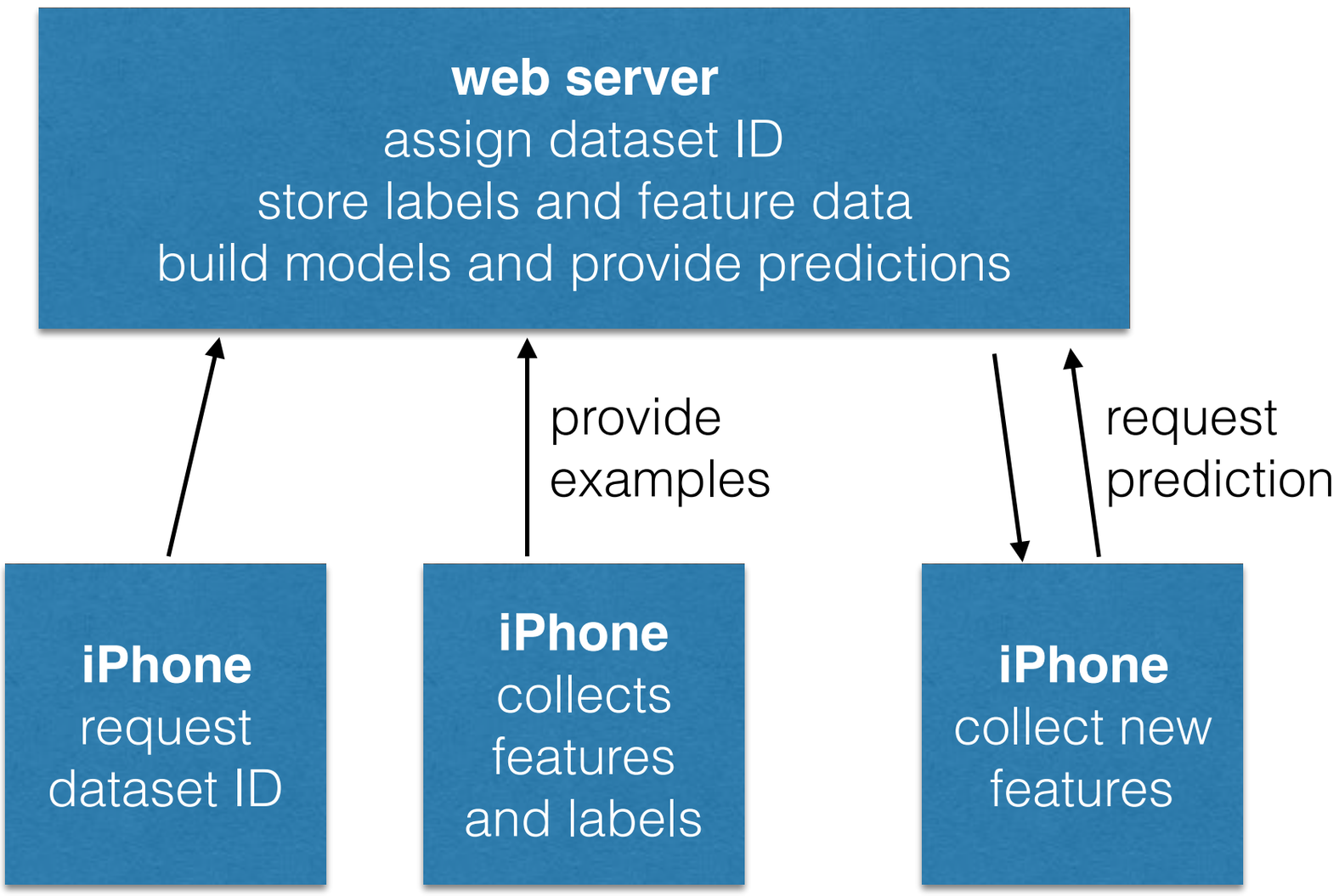
Video Lecture: machine learning and scikit-learn

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

Video Agenda

- pitfalls of machine learning
- numpy, scipy, and scikit-learn
- 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

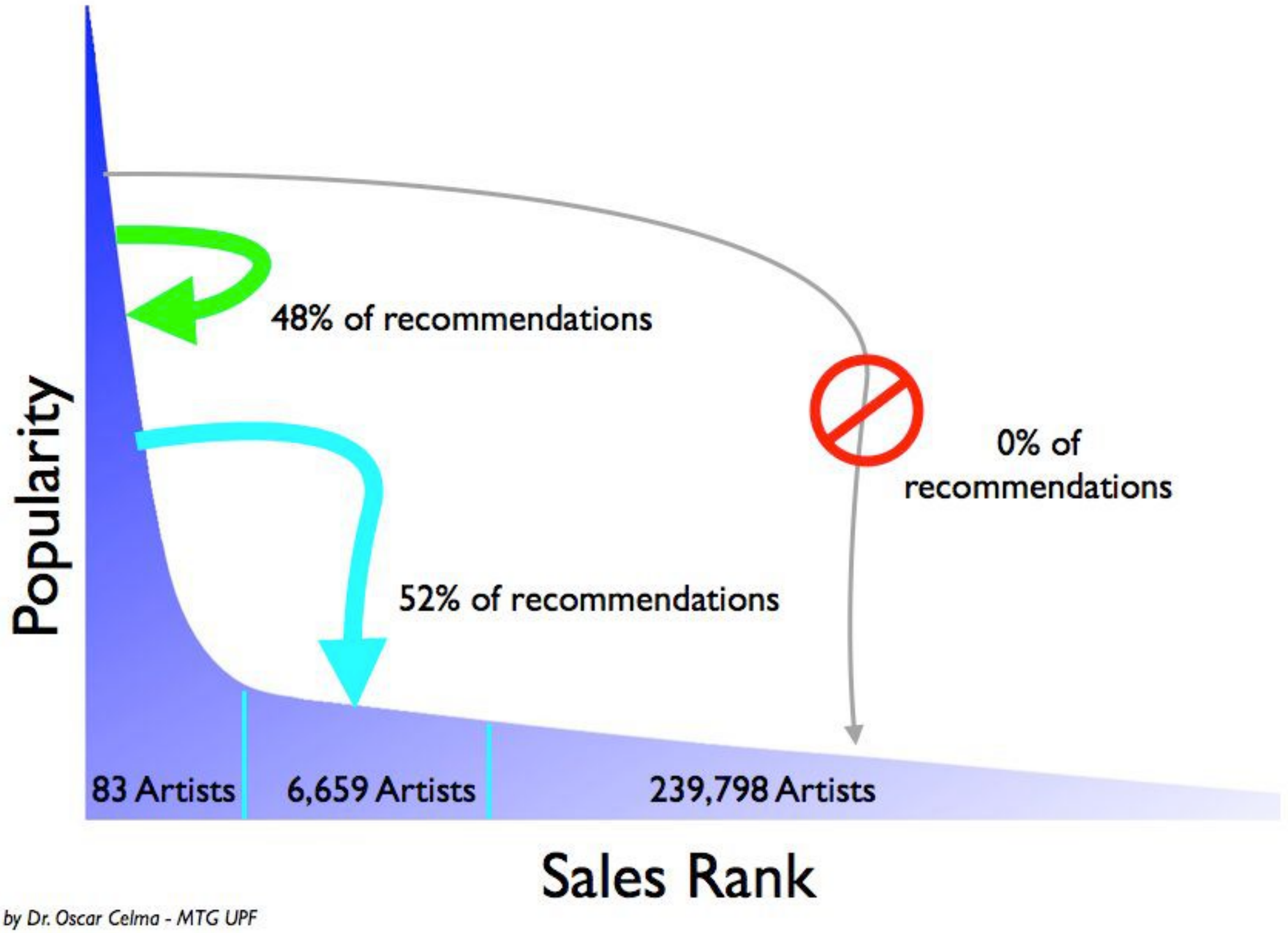
A recommendation that no human would make

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

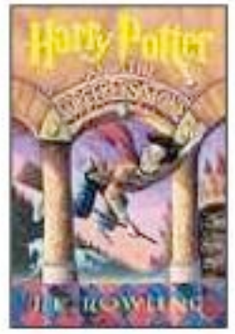
If you enjoyed Java RMI by *William Grosso*, you might also enjoy the following titles:

$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

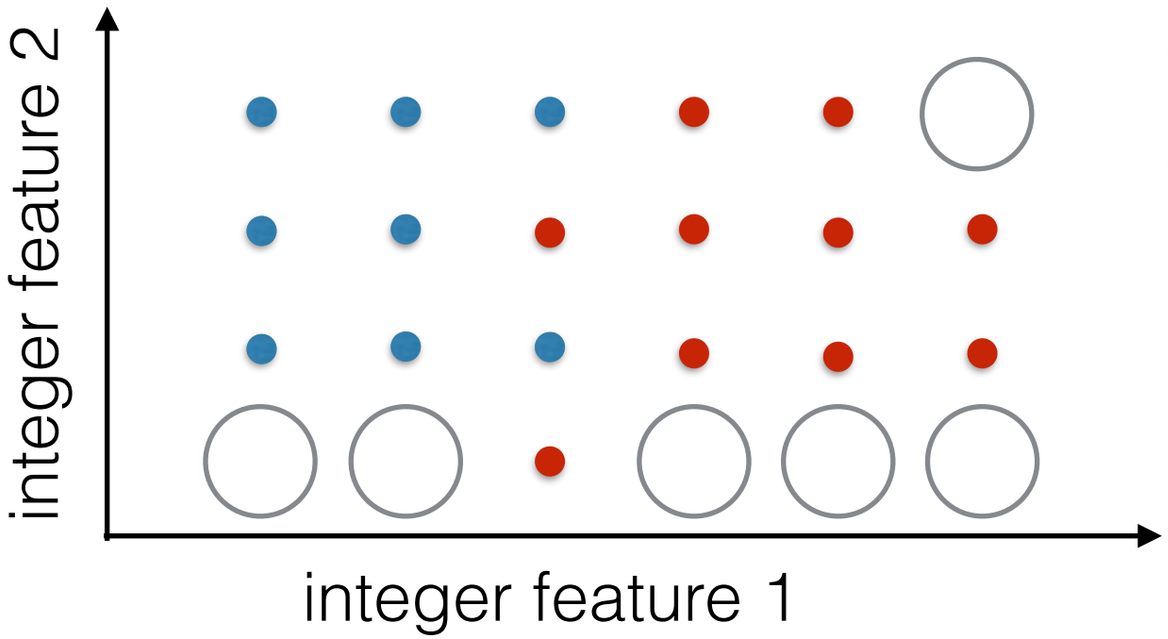
- there are lots of ways to be wrong
  - not enough diversity in classes
  - http://musicmachinery.com/tag/netflix/
  - outliers and novelty
    - ignore the deviant information
    - …or use it to detect novelty
- more data is better than cleverness
- but data alone is not enough
- performance on training 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

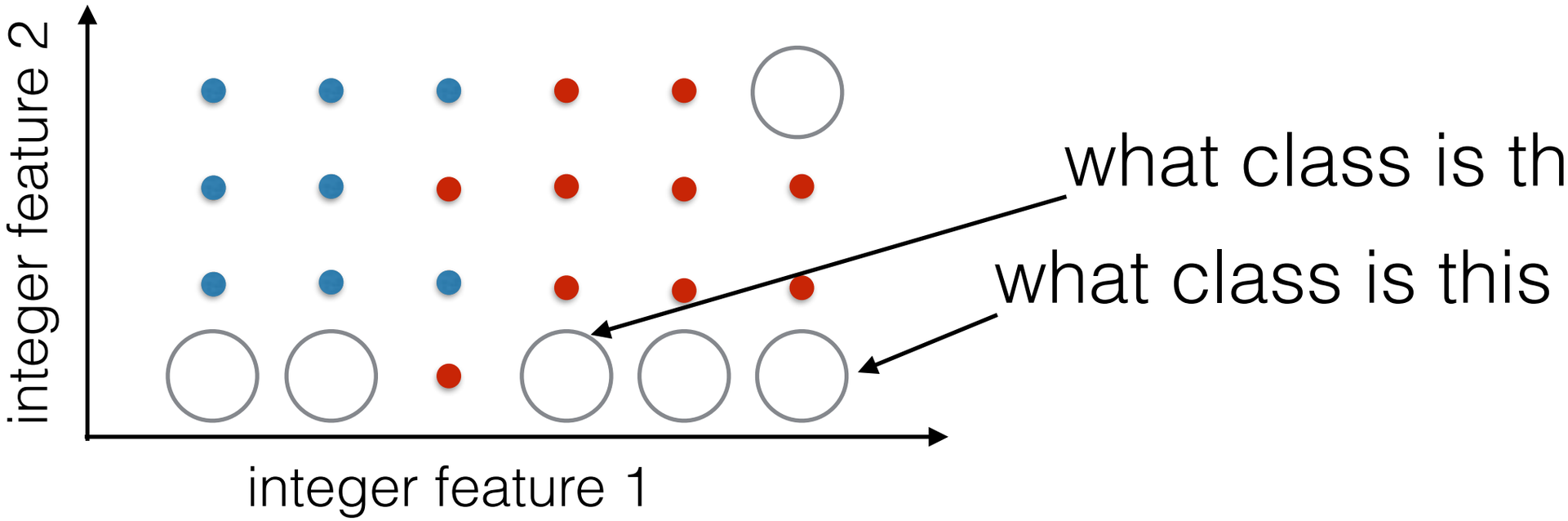
more realistic: 50 binary variables
2^{50} = 1.1 x 10^{15} possibilites

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



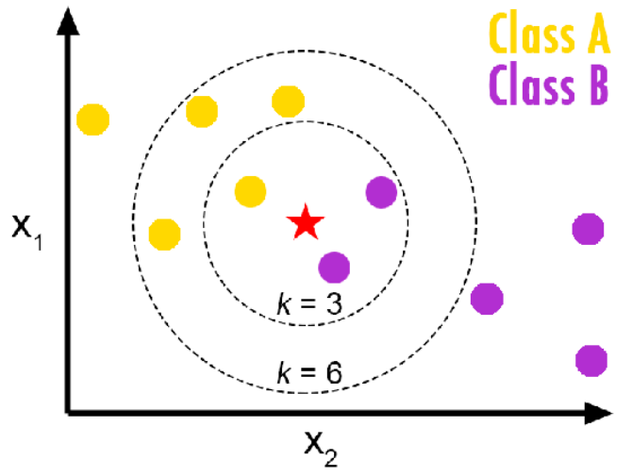
data does not span the entire space of features

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

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],
       [ 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

scikit-learn

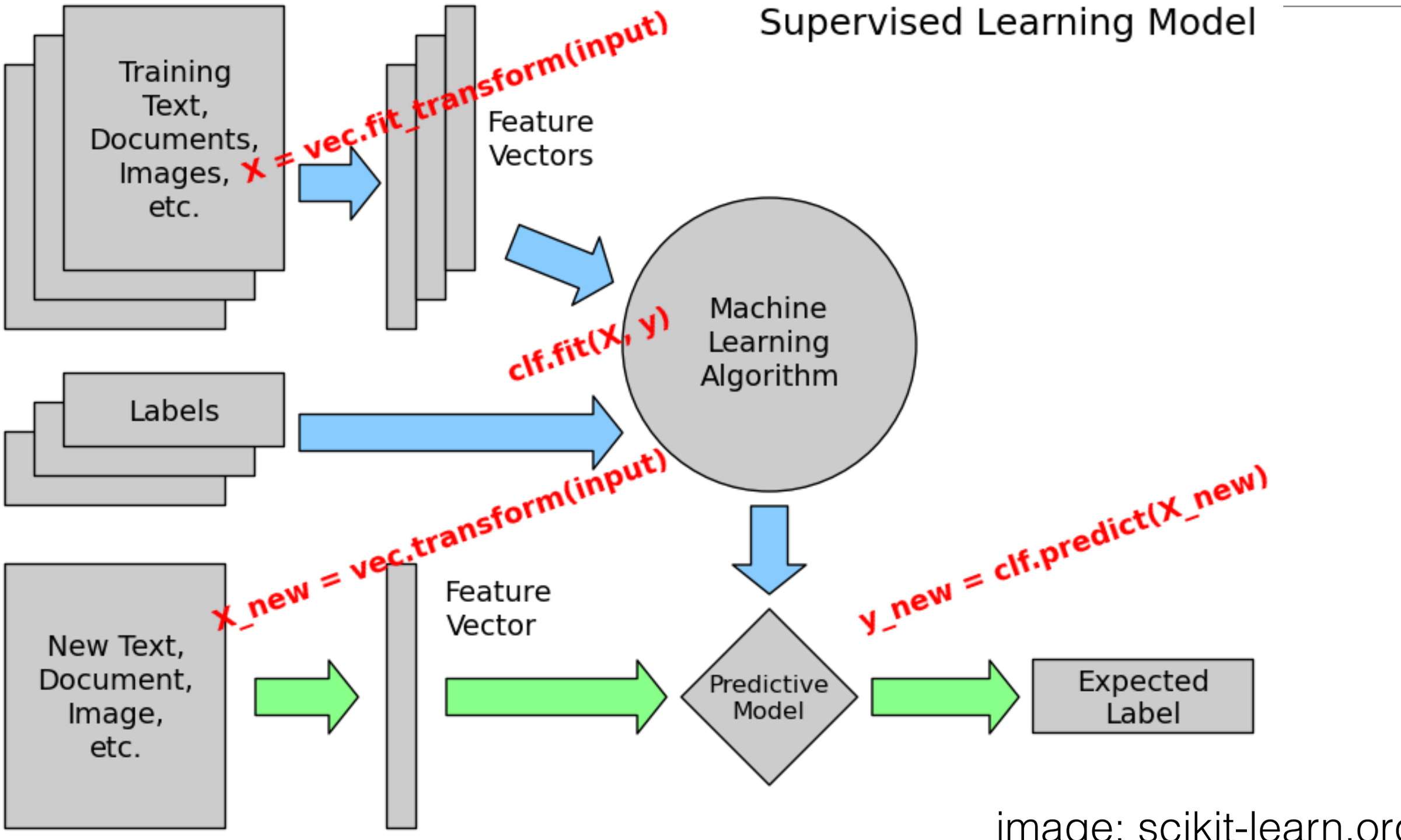


image: scikit-learn.org

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

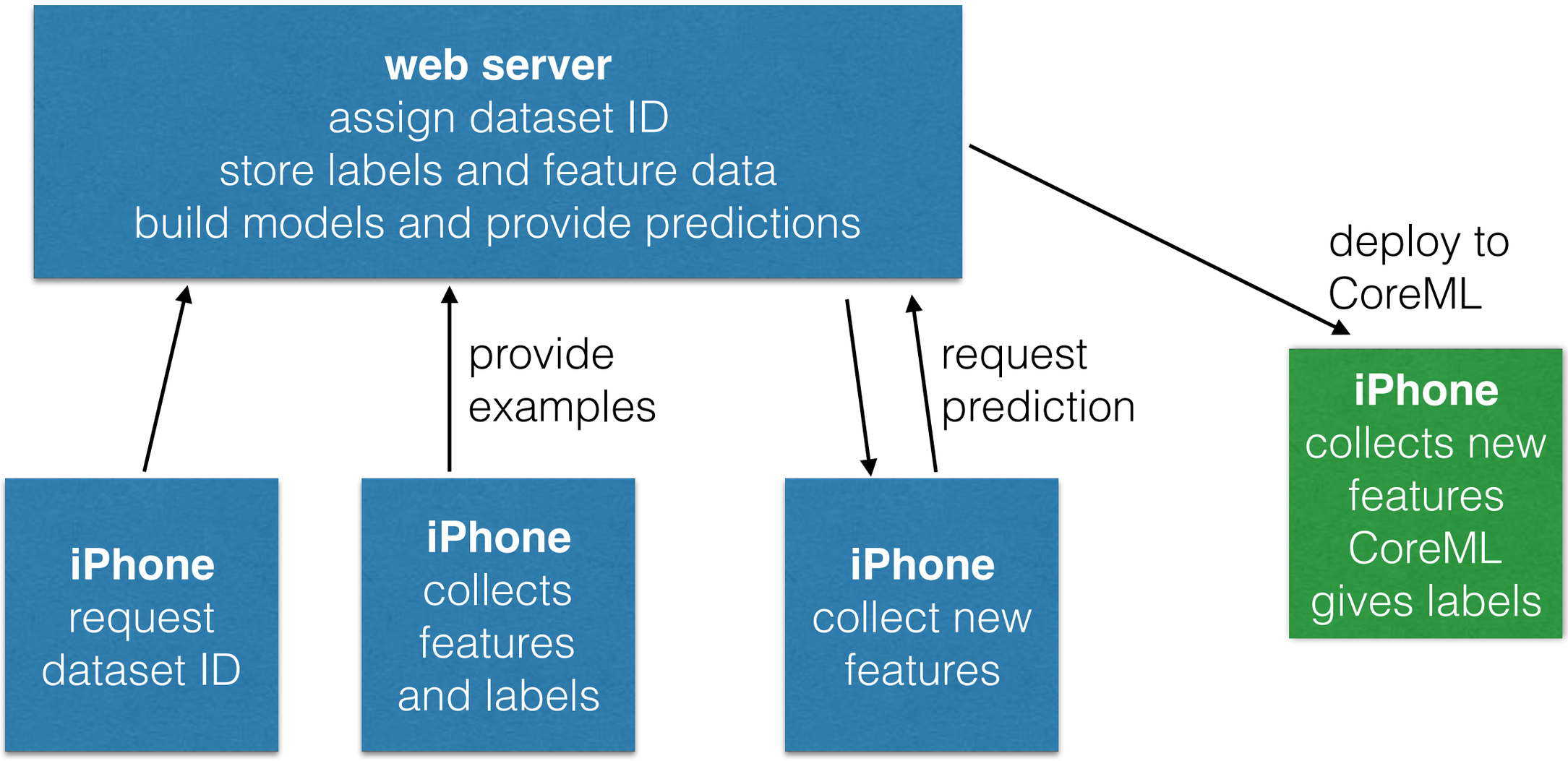
- get a classifier and train it

```
>>> 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

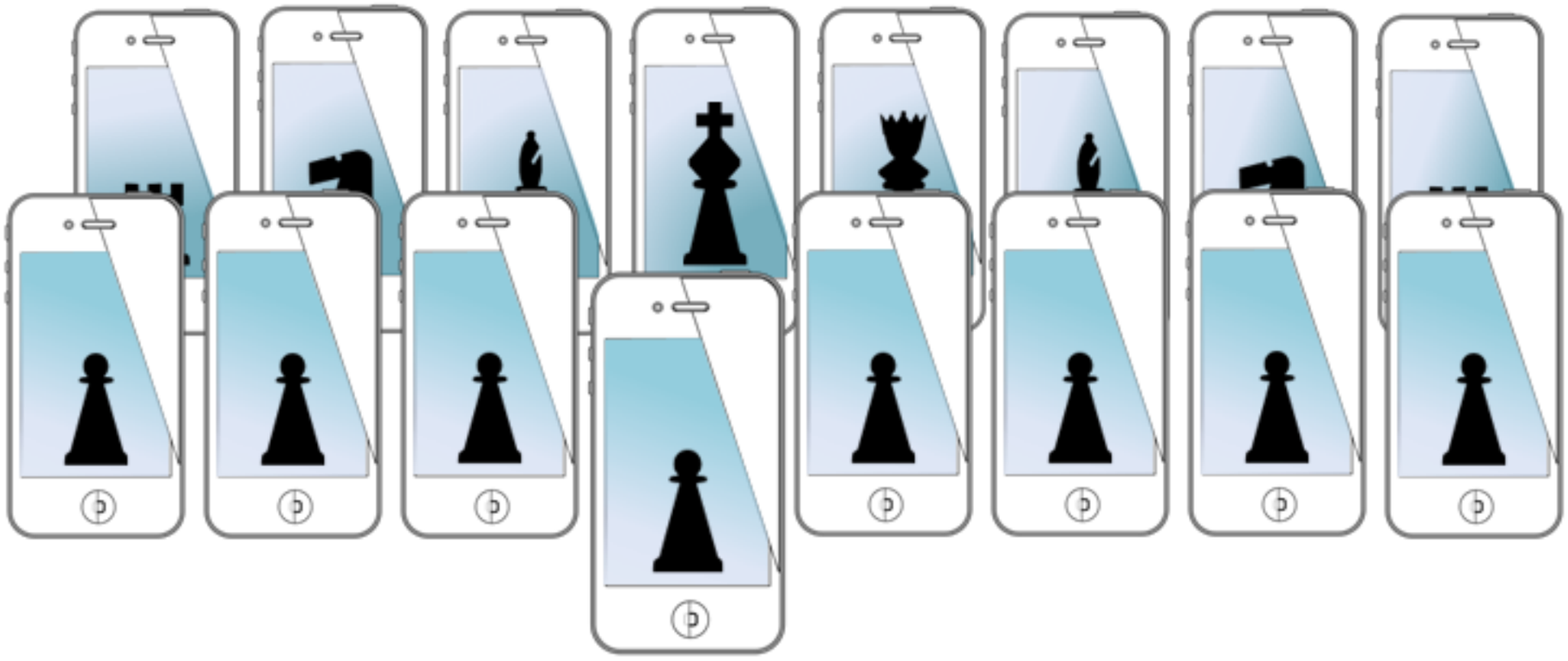
assignment 6 demo



all uploads handled via JSON formatted HTTP requests

blackboard: most of code is already posted

MOBILE SENSING LEARNING



CSE5323 & 7323

Mobile Sensing and Learning

Video Lecture: machine learning and scikit-learn

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