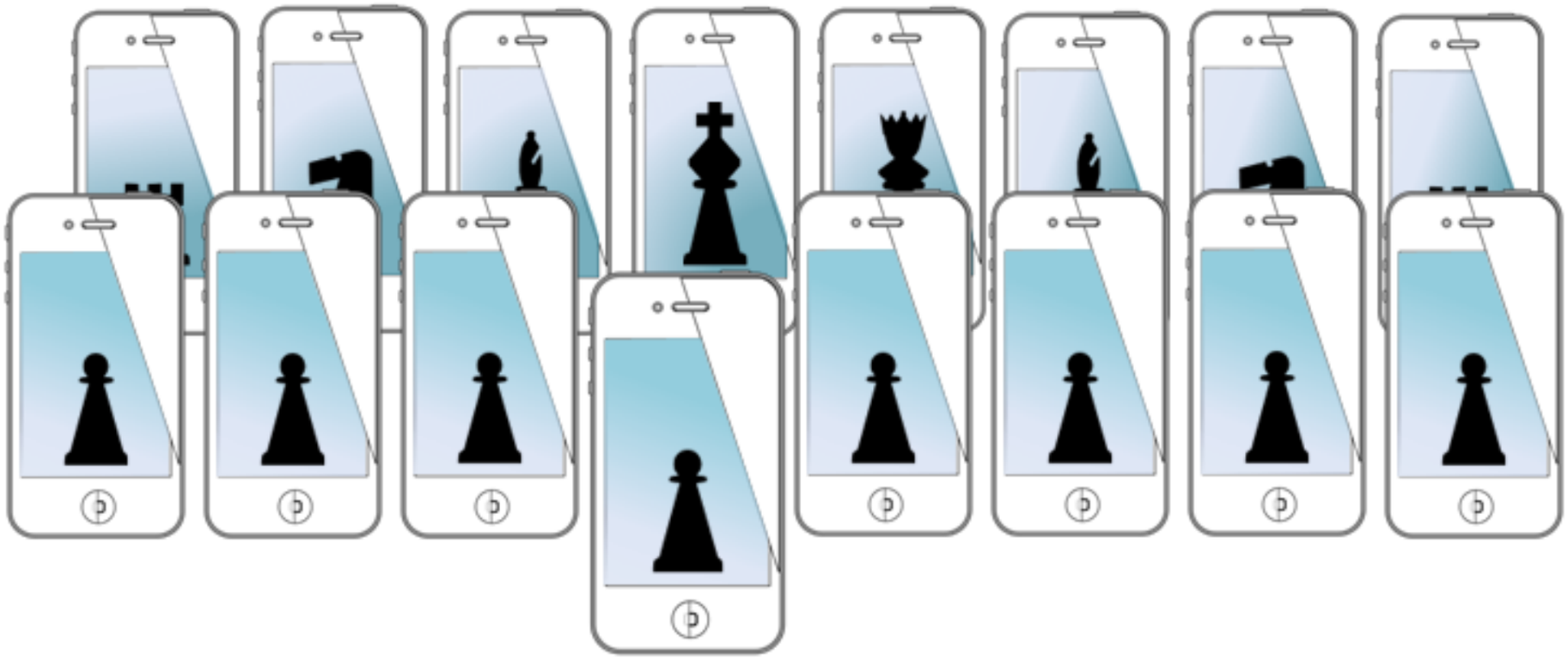
MOBILE SENSING LEARNING



CS5323 & 7323

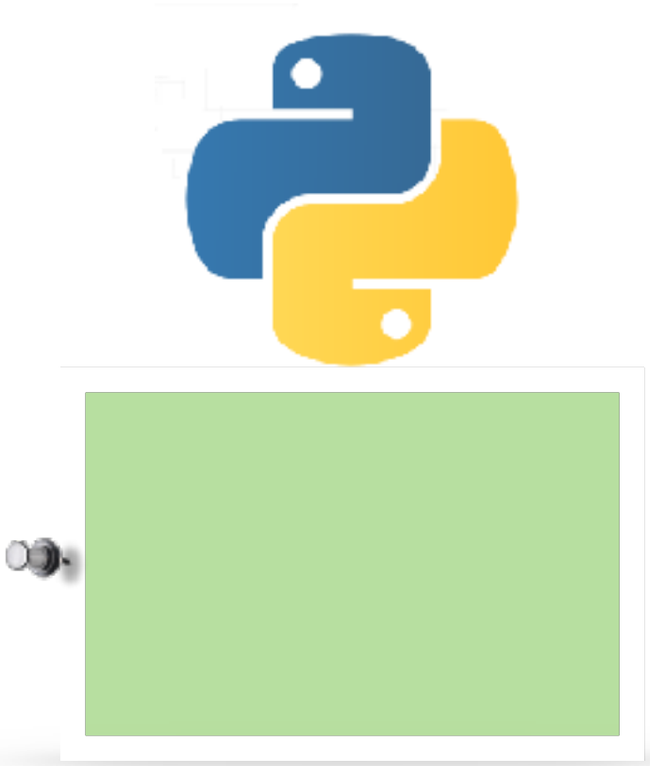
Mobile Sensing and Learning

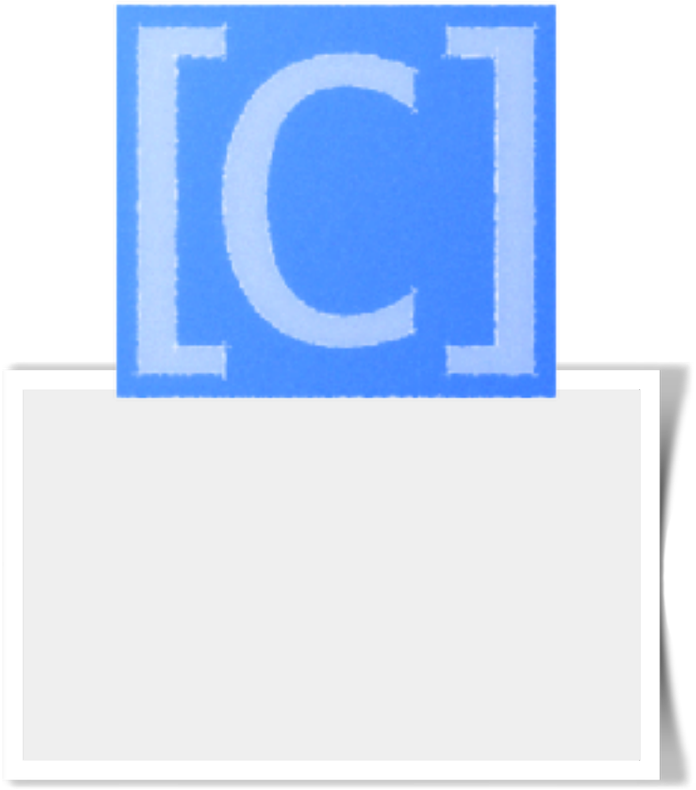
tornado, pymongo, and http requests

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

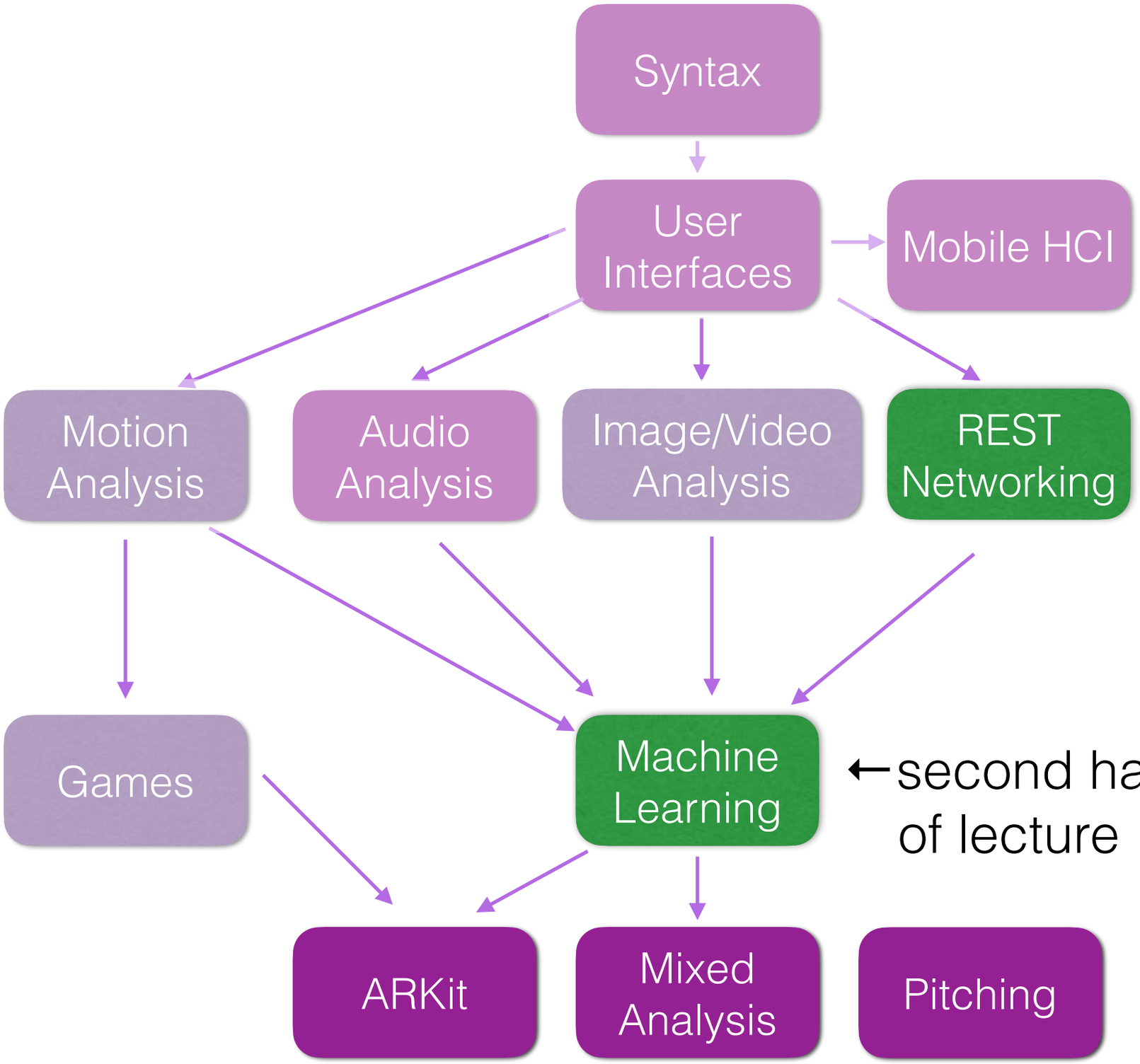
agenda and logistics

- logistics
  - next time flipped assignment!
- agenda
  - http requests in iOS
  - Turi ML





class overview



←first half of lecture

←second half of lecture

←remainder of course

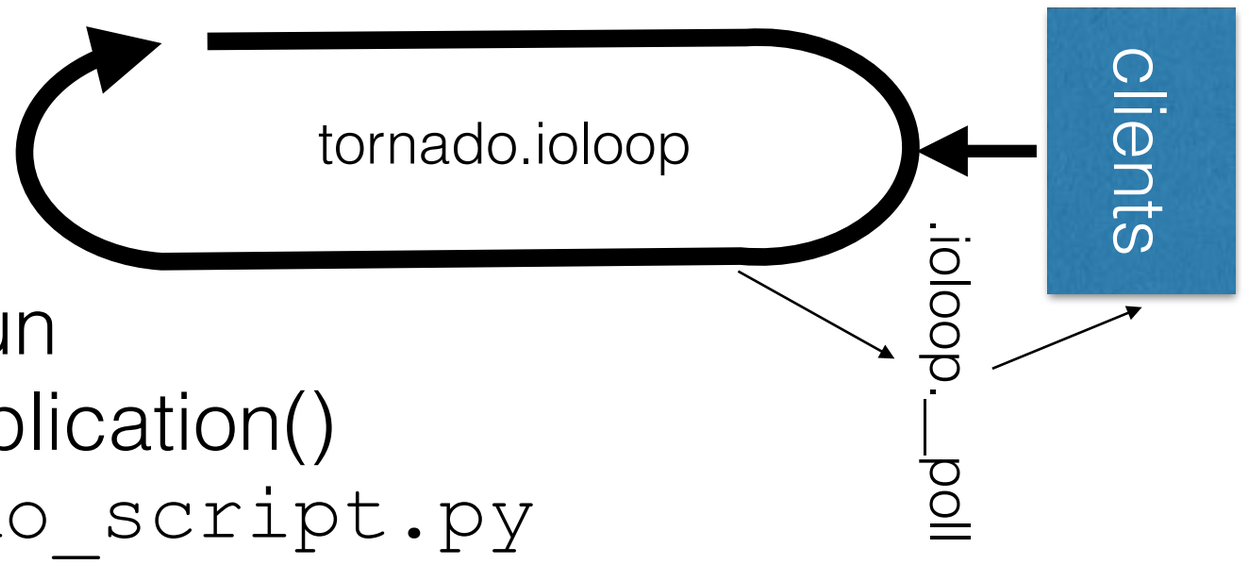
working with your web server

1. `brew services start`

Mongo Client

Tied to Tornado Instance Property

- 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 URLSession



2. run tornado.Application()

`python tornado_script.py`

3. GET/POST from iOS client

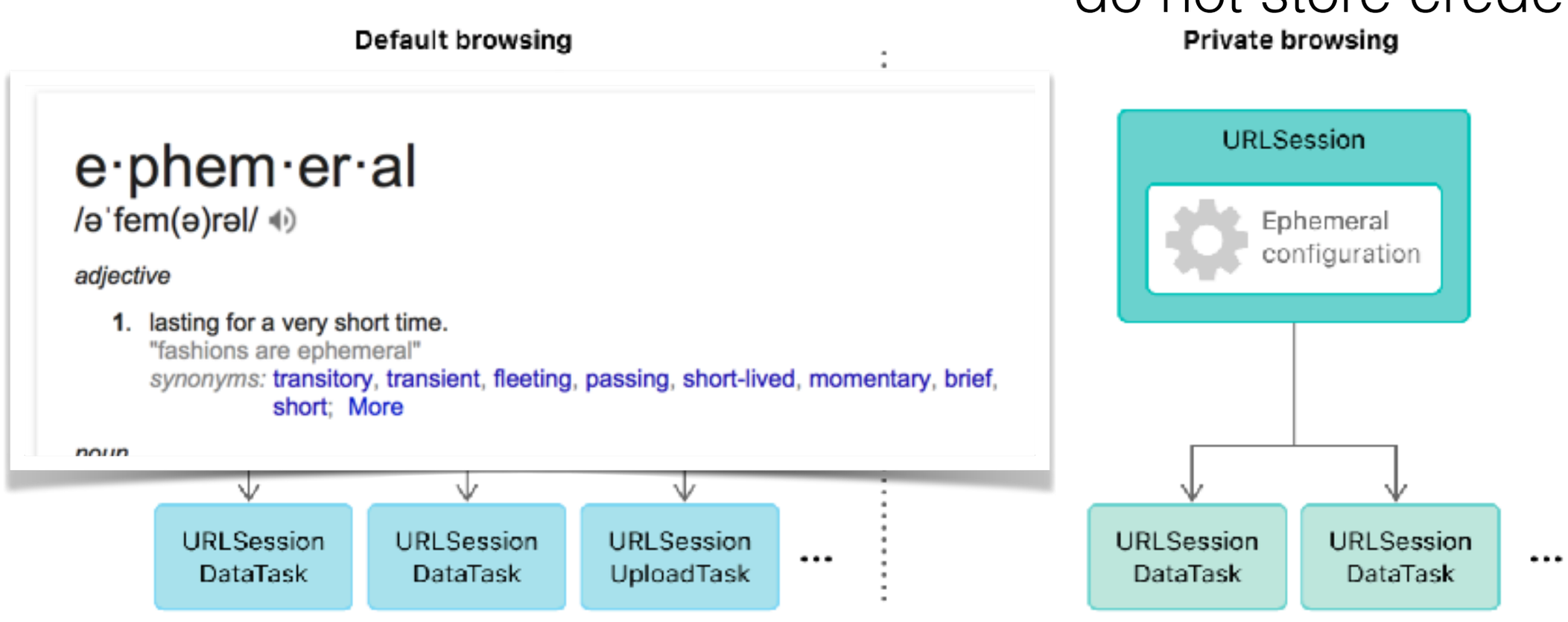
URLSession

- proper way to configure a session with a server
  - previous ways: `NSURLSession, NSURLConnection, 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

URLSession -> DataTask

- delegate model
- does authentication if you need it!
- implements pause / resume, tasks

do not cache
no cookies
do not store credentials



configure a session

```
class ViewController: UIViewController, URLSessionDelegate {

    // MARK: Class Properties
    let operationQueue = OperationQueue()
```

delegation

custom queue

```
 //setup NSURLSession (ephemeral)
    let sessionConfig = URLSessionConfiguration.ephemeral

    sessionConfig.timeoutIntervalForRequest = 5.0
    sessionConfig.timeoutIntervalForResource = 8.0
    sessionConfig.httpMaximumConnectionsPerHost = 1

    self.session = URLSession(configuration: sessionConfig,
            delegate: self,
            delegateQueue:self.operationQueue)
```

configure a DataTask

- DataTask objects are common requests tied to a session
- configure task to **specify URL** and type of **request**
- we will use a **completion handler** to interpret response from Server
- **larger** downloads allow use of delegates
    - progress indicators, completion indicators

URLSessionDataTask: GET

`dataTaskWithURL:completionHandler:(Data?,URLResponse?,Error?)`

URL as String

GET arguments as String

```
let baseURL = “\(SERVER_URL)/GetRequestURL" + query

let getUrl = URL(string: baseURL)
let request: URLRequest = URLRequest(url: getUrl!)

let dataTask : URLSessionDataTask = self.session.dataTask(with: request,
    completionHandler:{(data, response, error) in
        print("Response:\n%@",response!)
})

dataTask.resume() // start the task
```

must call, or stays suspended

URLSessionDataTask: POST

```
dataTaskWithURL:completionHandler:(Data?,URLResponse?,Error?)
```

```
// create a custom HTTP POST request
let baseURL = "\(SERVER_URL)/PostUrl"
let postUrl = URL(string: "\(baseURL)")
var request = URLRequest(url: postUrl!)

let requestBody:Data? = UIImageJPEGRepresentation(image, 0.25);

request.httpMethod = “POST"
request.httpBody = requestBody

let postTask : URLSessionDataTask = self.session.dataTask(with: request,
        completionHandler:{(data, response, error) in

 })

 postTask.resume() // start the task
```

could be any data

…we want this to be JSON…

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

```
let jsonDictionary: Dictionary =
      try JSONSerialization.jsonObject(with: data!,
       options: JSONSerialization.ReadingOptions.mutableContainers) as! Dictionary
```

the output in both scenarios is a dictionary

Dictionary

- serialize in iOS

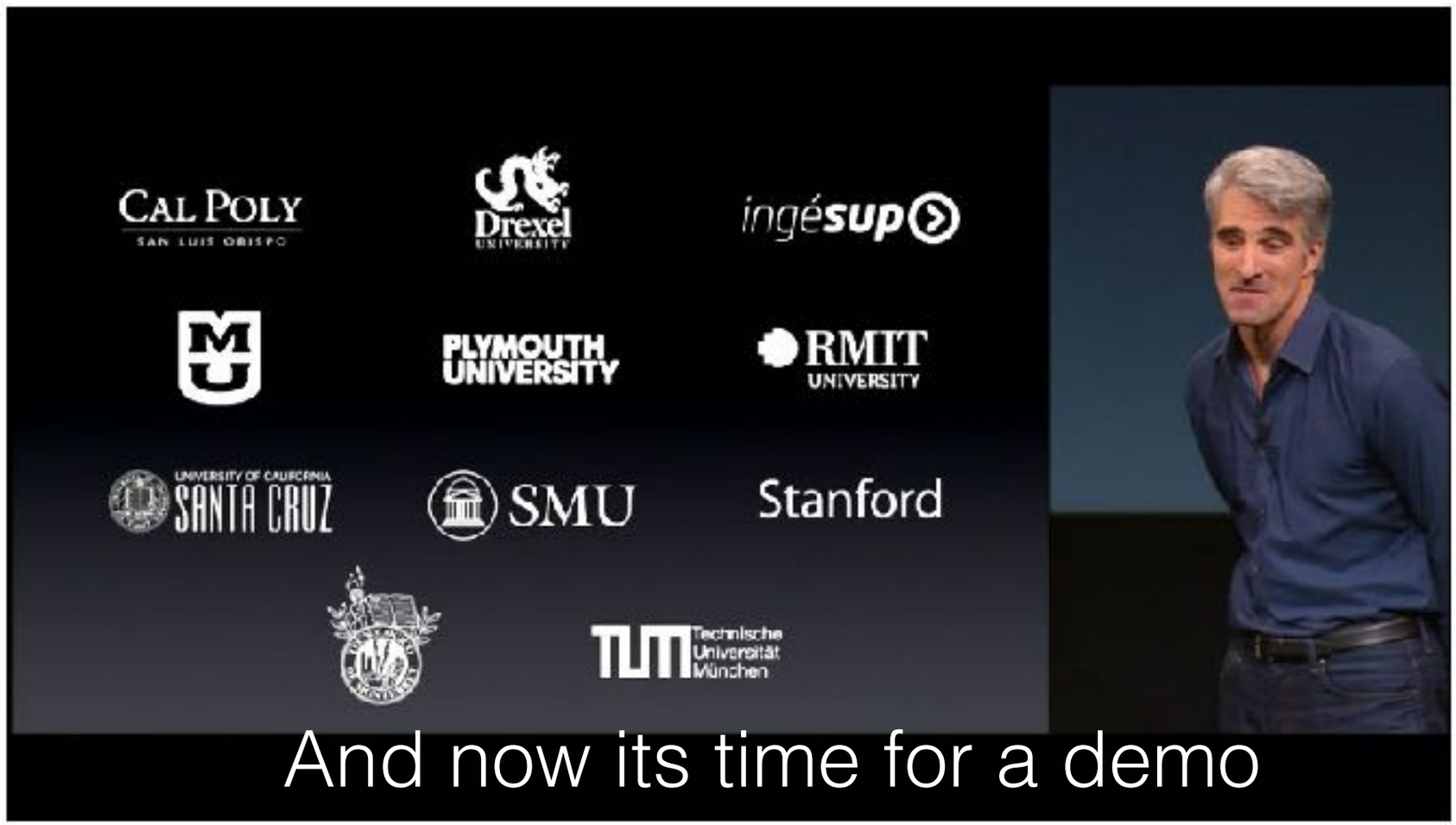
```
let requestBody = try JSONSerialization.data(withJSONObject: jsonUpload,
            options:JSONSerialization.WritingOptions.prettyPrinted)
```

tornado + iOS demo

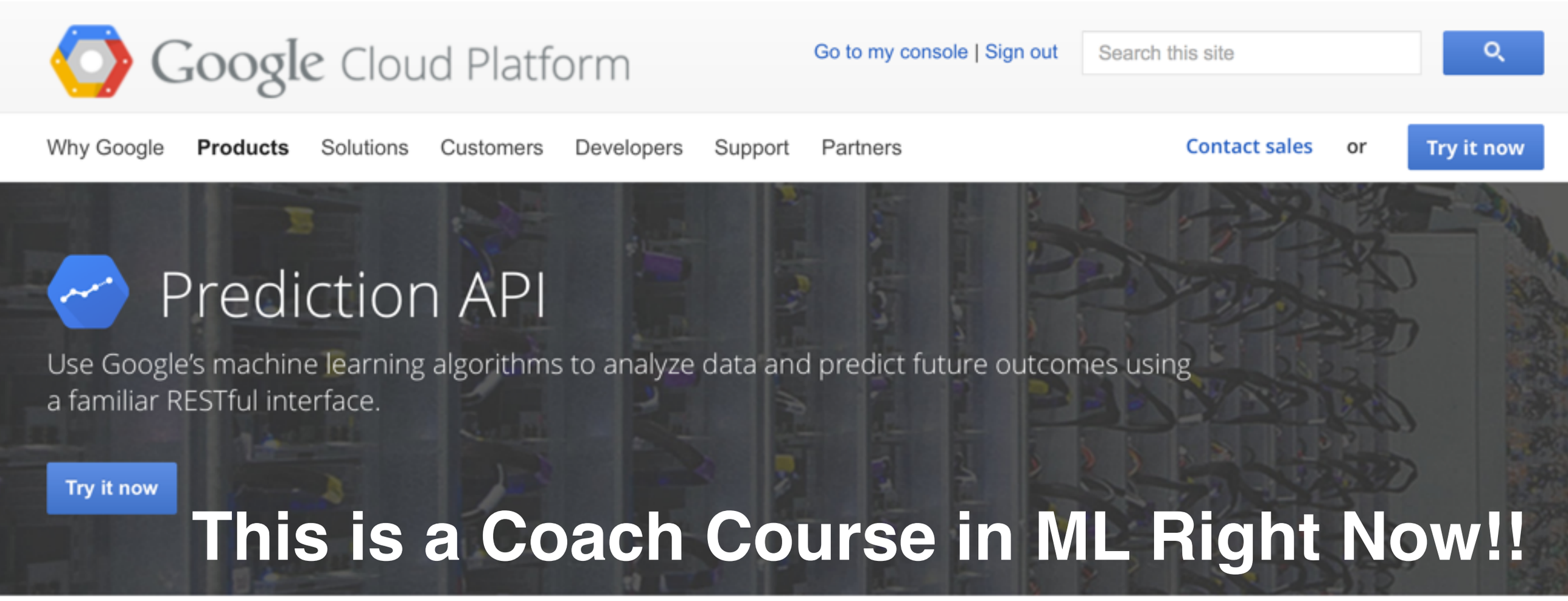
`HTTPSwiftExample.xcodeproject` (branch

`tornado_example.py`

1. send a GET request, handle query in tornado
2. do POST with GET-like arguments 😐
3. do POST with JSON in, JSON out

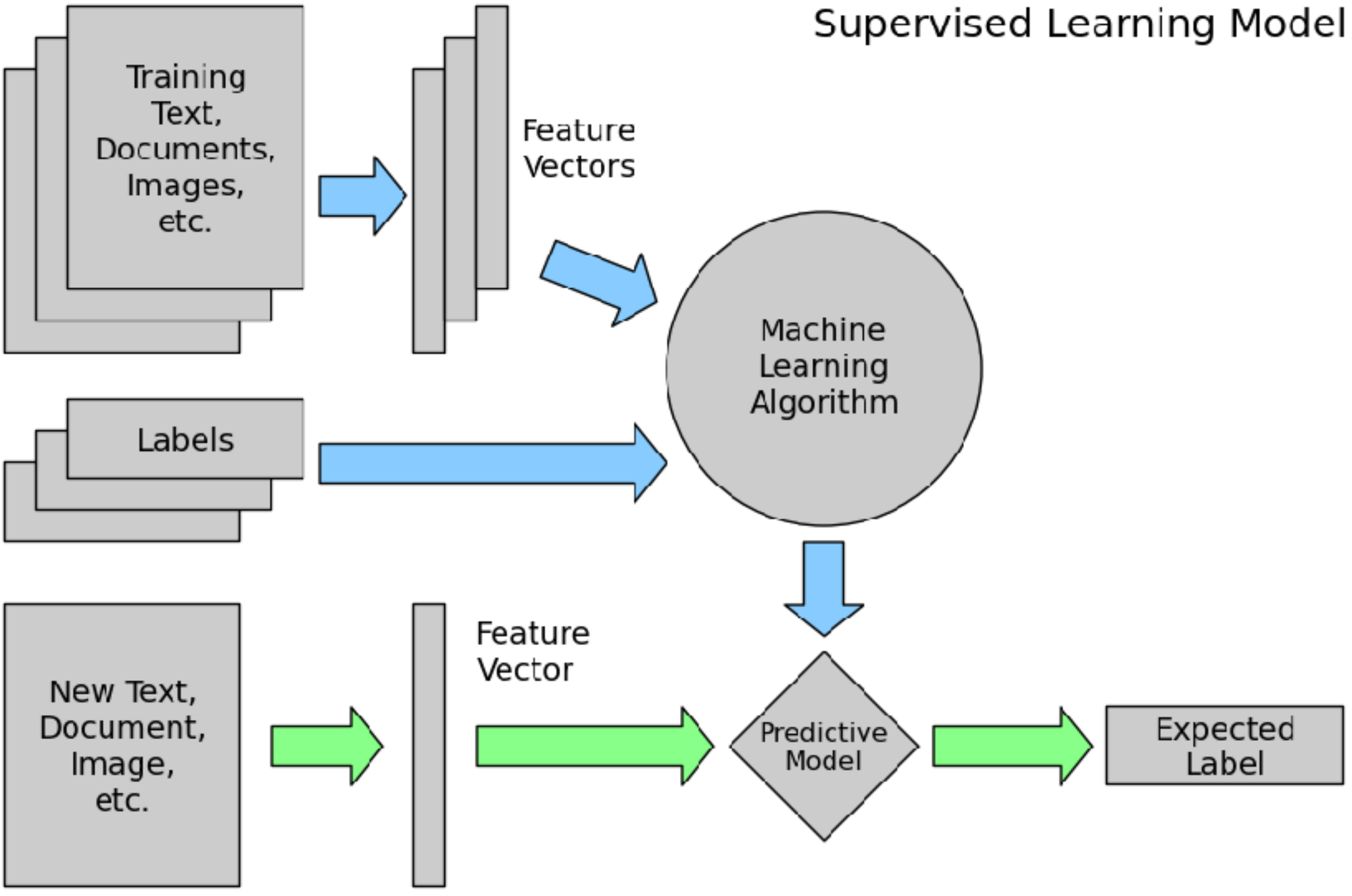


machine learning



This is a Coach Course in ML Right Now!!

machine learning models



types of data

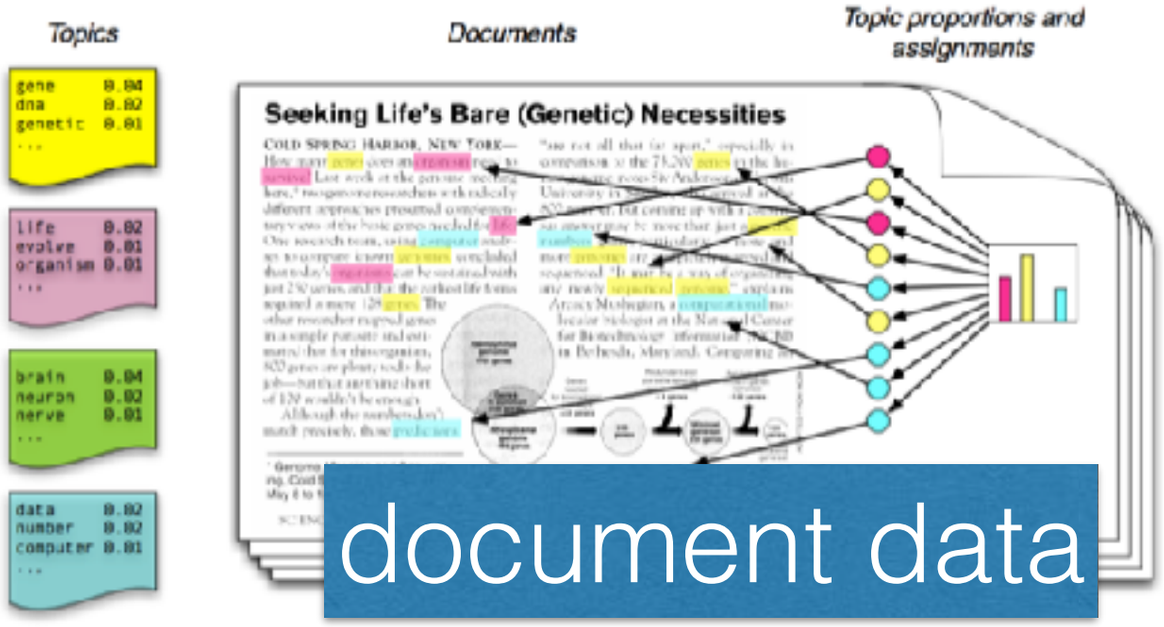


Figure source: Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, 55(4), 77-84.

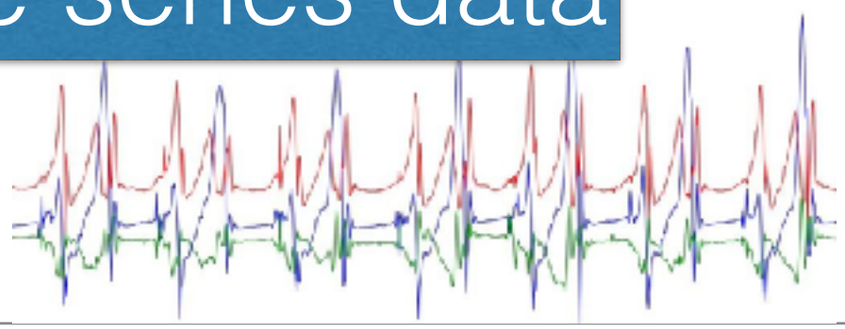
table data

Attributes, columns, variables, fields, characteristics, **Features**

Objects, records, rows, points, **samples**, cases, entities, **instances**

TID	Pregnant	BMI	Age	Diabetes
1	Y	33.6	41-50	positive
2	N	26.6	31-40	negative
3	Y	23.3	31-40	positive
4	N	28.1	21-30	negative
5	N	43.1	31-40	positive
6	Y	25.6	21-30	negative
7	Y	31.0	21-30	positive
8	Y	35.3	21-30	negative
9	N	30.5	51-60	positive
10	Y	37.6	51-60	positive

time series data



features and labels

- actually, feature vectors
- **classic example**: the iris dataset—table data

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

features

- most common is numeric and categorical
- vector quantization (numeric to categories)
- text:
  - bag of words
  - term frequency inverse document frequency
  - text embeddings from neural nets
- graphs
  - used to quantize

take data mining!
or python machine learning!

common ML algorithms

nonparametric

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

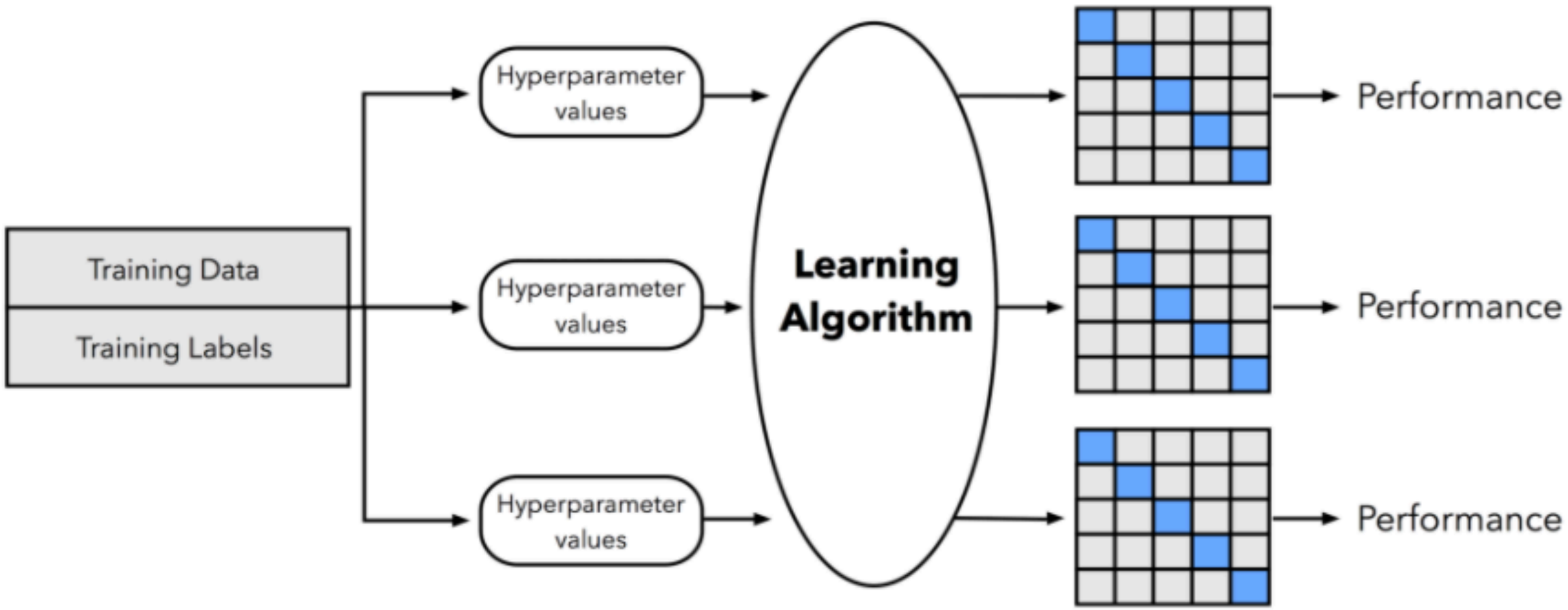
parametric

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

- support vector machines
- and many many more…

finding the best ML model

- try a bunch of stuff until it works well enough

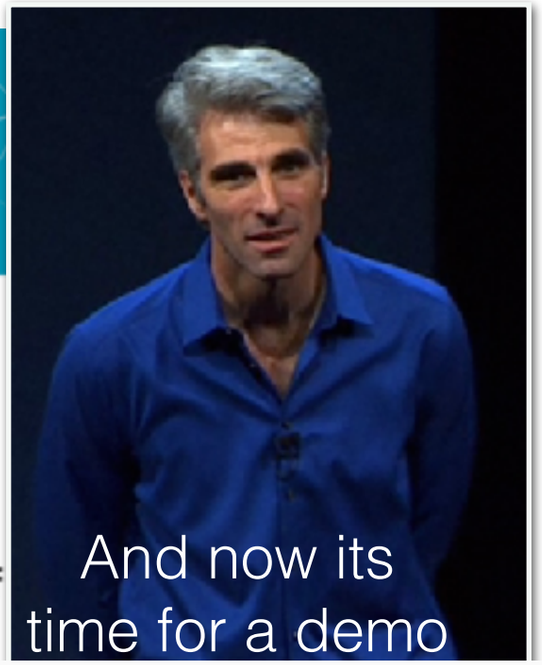


http://ethen8181.github.io/machine-learning/model_selection/model_selection.html

turi create demo, with SFrame

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`

turicreate.SFrame

class `turicreate.` `SFrame` ***(data=None, format='auto', _proxy=None)***

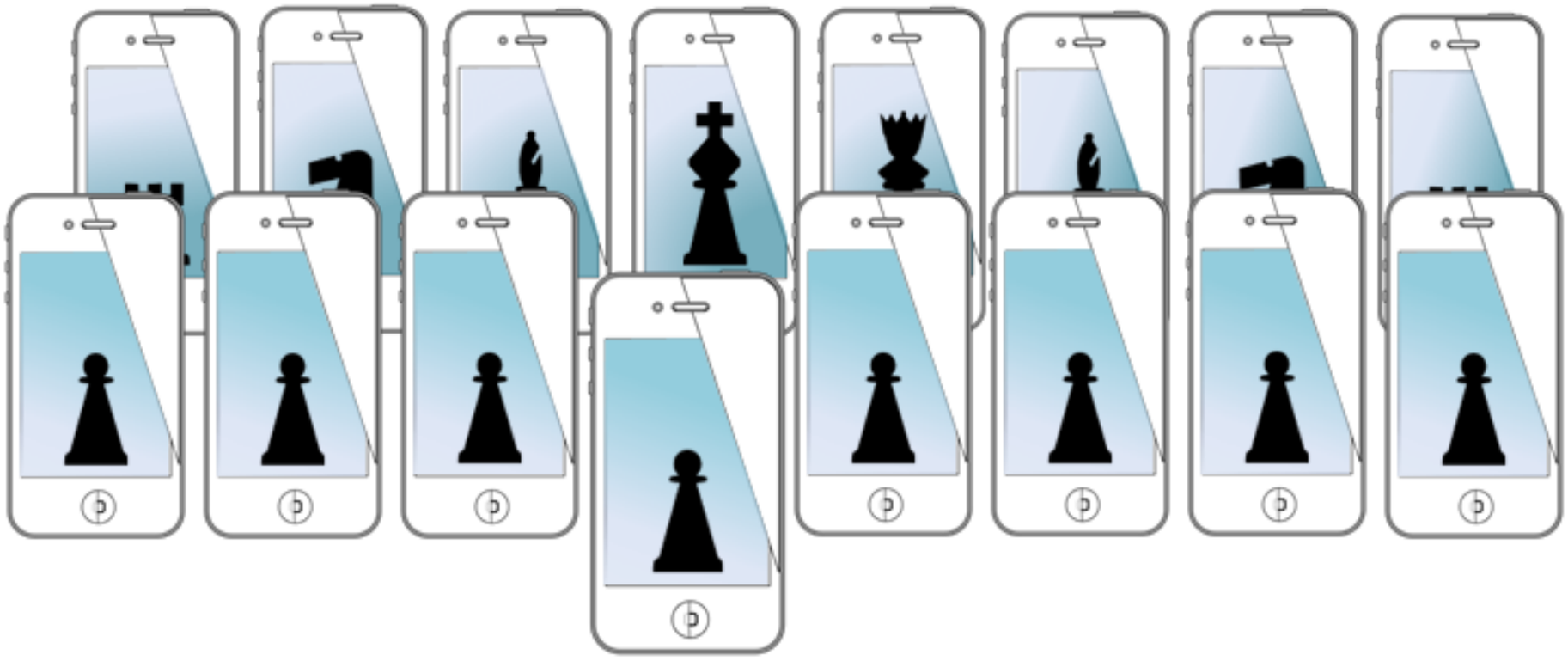
SFrame means scalable data frame. A tabular, column-mutable dataframe object that can scale to big data. The data in SFrame is stored column-wise, and is stored on persistent storage (e.g. disk) to avoid being constrained by memory size. Each column in an SFrame is a size-immutable `SArray`, but SFrames are mutable in that columns can be added and subtracted with ease. An SFrame essentially acts as an ordered dict of SArrays.

Currently, we support constructing an SFrame from the following data formats:

- csv file (comma separated value)
- sframe directory archive (A directory where an sframe was saved previously)
- general text file (with csv parsing options, See `read_csv()`)
- a Python dictionary
- pandas.DataFrame
- JSON

Additional Lab 4 discussion

MOBILE SENSING LEARNING



CS5323 & 7323

Mobile Sensing and Learning

tornado, pymongo, and http requests

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