# Machine Learning for Web Developers

Al Barrentine
albarrentine@gmail.com

# OMG no setup today

- The stuff I'm presenting is on my GitHub
- git clone https://github.com/thatdatabaseguy/ml4webdevs

# Why you should care

- Recommendations systems
- Spam/fraud detection
- Finding similar documents/objects/people
- Categorizing text
- Serving advertisements

# The main libraries

- The Natural Language Toolkit (nltk)
  - easy to set
  - includes natural language constructs
- scikits.learn
  - FAST!!!! Production ready
  - Need to learn scipy and linear algebra

# Some bubble-bursting

- The following does not exist:

```
import machine_learning
learn(my_data)
predict(my_new_data)
```

- And it NEVER will, no matter what any library or company advertises
- This stuff cannot be done through an API

# Models for today's talk

- Wrote most of it myself (why not?)
- Supervised learning
  - Naive Bayes classifier
- Unsupervised learning
  - Jaccard similarity
- Tried not to require scipy

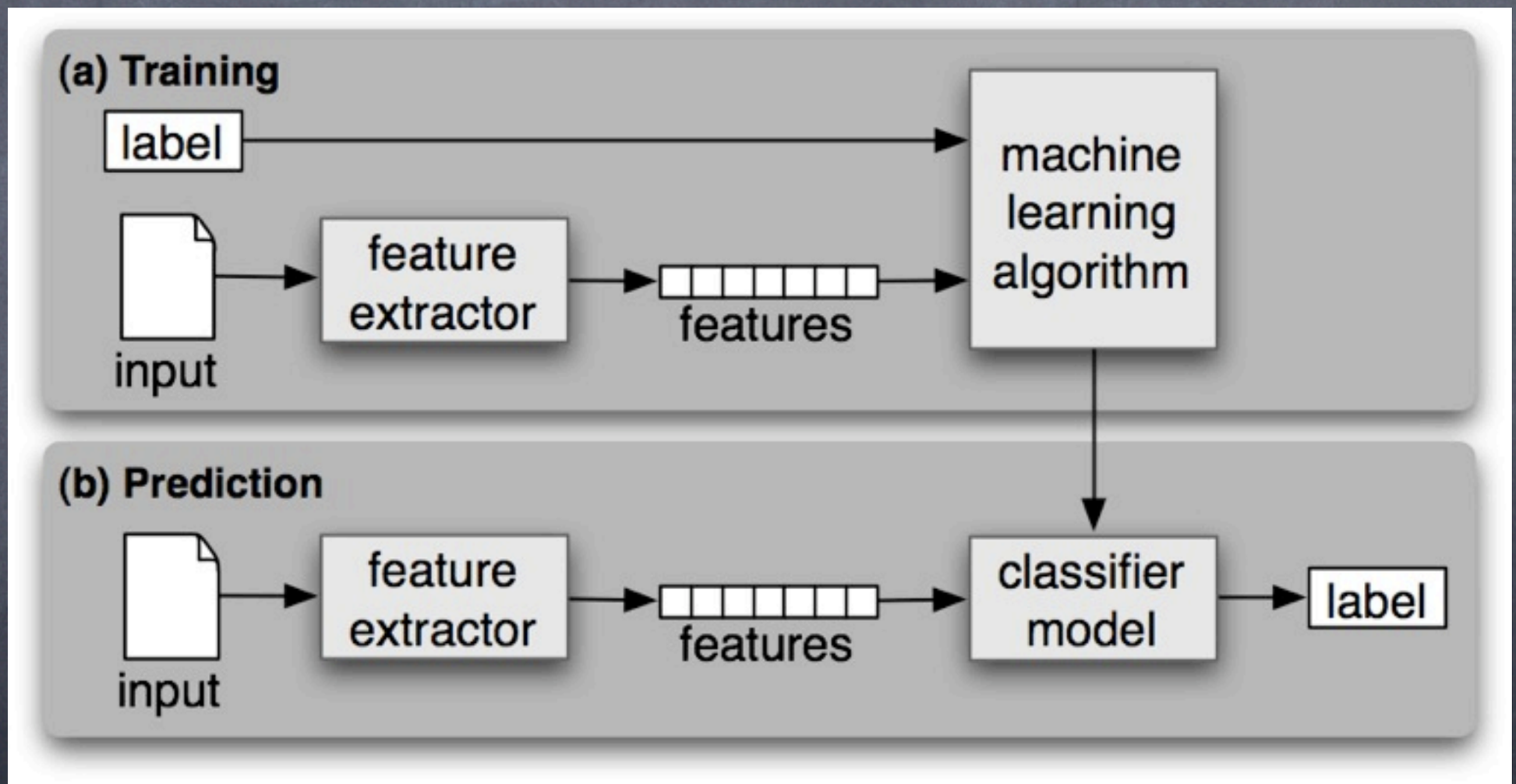
# Two "flavors" of ML

- Supervised learning
  - Classfication
  - Regression
- Unsupervised learning
  - Clustering

# Supervised Learning

- Have some known, correctly labeled data
- Extract some **features** from the data, forming a set of **training tuples**
- Use an algorithm (there are several) to learn from the data and produce a **model**
- The model can be used in the **prediction** of labels from unseen instances

# Or a picture


(a) Training
label
machine learning algorithm
input
feature extractor
features
(b) Prediction
input
feature extractor
features
classifier model
label

# Example #1: Political tweet classification

- Top two political hashtags on Twitter
- #p2 - Progressives 2.0
- #tcot - Top conservatives on Twitter
- Boolean searches (#p2 -#tcot, #tcot -#p2)
- See if we can predict which party based on content of tweet (mentions, words, etc.)

# Methods

- Naive Bayes classifier
- Fetch 7 pages of 200 tweets each
- All these tweets are "correctly labeled" as either hashtag


Stand back!
MILK
100
80
60
40
20
Iz goin to do science!
ICANHASCHEEZBURGER.COM

# Old Man Bayes' Law

- Promised to stay light on the notation, but:

$$P(C \mid A) = \frac{P(A \mid C)P(C)}{P(A)}$$

- P(C | A) is read "probability of C given A"

# A Naive Bayes Classifier

- Hello World of supervised learning
- Single classification (maximum likelihood)

$$\text{classify}(f_1, \ldots, f_n) = \operatorname*{argmax}_c \, p(C = c) \prod_{i=1}^{n} p(F_i = f_i | C = c).$$

- "Naive" - assumes features are independent
- Very efficient to train (multiply probabilities)

# Demo time

"Hold onnn to your butts..."

# Some results

- 80-87% accuracy on test set
- Some salient words from past runs:
  - "read" - 6:1 DEM
  - "hate" - 8:1 GOP
  - "America" - GOP, "Americans" - DEM
  - "Obama" - GOP, "@BarackObama" - DEM

# Unsupervised Learning (warning: BFT=Big Topic)

- Used for clustering data
- Don't know any right answers beforehand
- Good for similarity (Latent Semantic Indexing or Latent Dirichlet Allocation)
- Difficult to evaluate accuracy

# Example #2 - how similar ru2me?

- Grab your interests
- Grab your friends' interests
- Take the **Jaccard coefficient** of all your friends with respect to you
- Can also do friends with respect to each other (code included)

# Methods

- Facebook interests are semantically tagged, so they might be ('Field of study', 'computer science') and ('Interest', 'computers')
- Jaccard coefficient - intersection / union

# Jaccard Coefficient

- (A ∩ B) / (A ∪ B)
- Takes into account both how similar AND how different two sets are

# Demo time

"Hold onnn to your butts..."

# Results

- No fancy performance stats (difficult to evaluate accuracy of unsupervised methods)
- For me: results are not that great
- I don't put much on Facebook
- Other factors: photos tagged in, past events attended - who do you DO stuff with?

# Some stuff for web developers

- Need to save/load the model
  - Cache (small models)
  - joblib + mmap (larger models)
- Think about datamine machines as services
  - Decouple from web servers
  - Serve JSON, use IDs

# Further Study

- scipy.sparse - great for linear algebra and matrix math
- nltk
- scikits.learn

# Advanced Topics (if there's time)

# Unsupervised Learning Models

- Usually involves choosing a number of topics or clusters
- Don't need to compare every instance to every other instance, just every instance to every cluster

# To check out

- K-means clustering
- Latent Semantic Indexing
- Latent Dirichlet Allocation

# scipy

- nltk is great, but not that scalable
- Really need scipy for large-scale problems
- Instead of a dict, we could have built a **term-document matrix** where the rows are documents and the columns are features with numerical ids. Need a mapping.
- Blindingly fast, everything's in C or Fortran

# scipy (cont'd)

- sparse matrices (mostly 0's)
- data = numpy.array([10,20,30]); row=numpy.array([0,1,2]); col = numpy.array([5, 8, 9])
- Only store the non-zeroes and their positions
  - matrix = csr_matrix((data, (row, col)), shape=(3,10), dtype=numpy.int32)

# scipy (cont'd)

- matrix.todense()

```
matrix([[ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
        [ 0,  0,  0,  0,  0, 10,  0,  0,  0,  0],
        [ 0,  0,  0,  0,  0,  0,  0,  0, 20,  0],
        [ 0,  0,  0,  0,  0,  0,  0,  0,  0, 30]],
dtype=int32)
```

# scipy (cont'd)

- Very efficient for storing things like text documents as vectors
- Each column is a feature (word, bigram, etc.)
- Need a dict of {feature: col_id}
- scikits.learn can use these sparse matrices instead of dictionaries, tends to scale better
- joblib releases the GIL, good for multicore