APMA E4990.02: Introduction to Data Science In Industry

Instructor: Dorian Goldman

Who am I?

- Dorian Goldman, Ph.D 2013
- Ph.D at Courant Institute NYU and Paris VI UPMC) (Calculus of Variations and Partial Differential Equations)
- Instructor of mathematics at University of Cambridge (2013-2014)
- Data Scientist - The New York Times 2014-2016
- Data Scientist/Researcher Engineer - Conde Nast (2016 - 2018)
- Senior Research Scientist - Lyft (2018-Current)

What are the goals of this class?

- Learn the rigorous mathematical foundation of machine learning as it relates to problems faced in realistic industry scenarios.

- Learn the tools used in industry, and how these algorithms and methods are used in practice (Python, Scikit-learn, SQL/Map Reduce, Github, Web scraping)

- Build your own web app which uses machine learning to solve a problem using Machine Learning. Examples could include:
    - Recommendation engine for Yelp/Netflix/Amazon (or any other service).
    - Taxi route time estimator (taxi data exists, and uber just released theirs).
    - MTA time estimator
    - Music recommendation system (Soundcloud, Spotify).
    - Stock/Investment tools (https://www.interactivebrokers.com/).

What do you need to be ready?

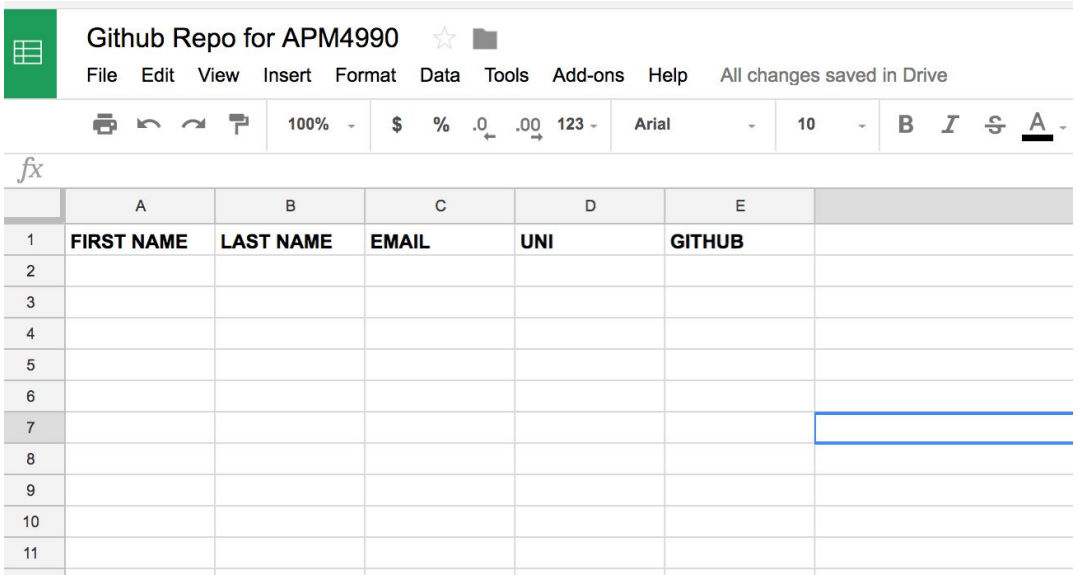
- A laptop (although not explicitly necessary), preferably running MacOS or Linux, but also not essential (you will be judged though). We will be working through algorithms in the class.

- Anaconda - Scientific Python package with IPython Notebook.
  https://www.continuum.io/downloads

- Github - If you don't have an account, please go to
  https://github.com/, and register an account.
- Github repo:
  https://github.com/Columbia-Intro-Data-Science/APMAE4990-

Add your name and Github name to this sheet

https://docs.google.com/spreadsheets/d/1KEBDe8H0x_drnqx4ZeMyMUN-6-LrDVy2tBBJ97RZtMY/edit?usp=sharing



Additional Resources

You can also read my blog for supplemental material:

https://doriang102.github.io/

What is Data Science?

Introduction and Examples

Outline

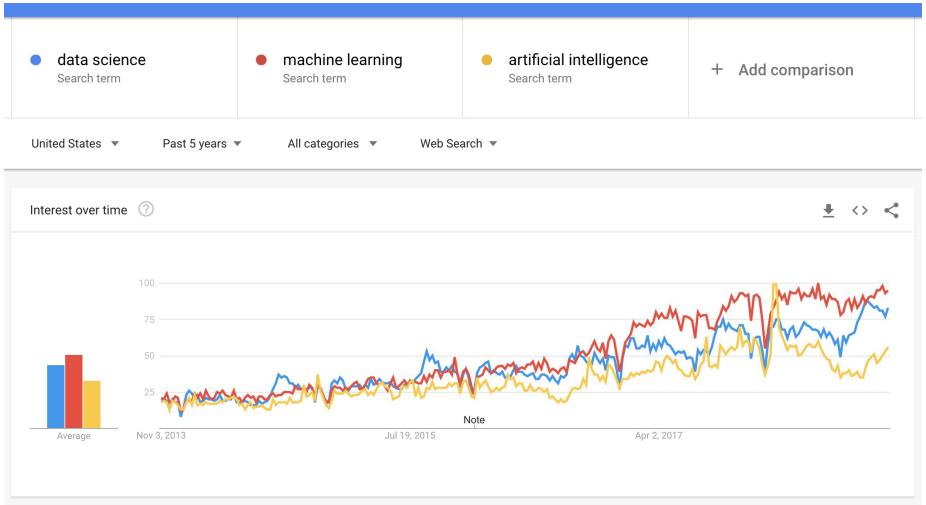
- What is Data Science? What are the skills needed?
- Examples from Industry. Lyft, Amazon, Netflix, Booking.com, New York Times.
  - Supervised Learning
  - Unsupervised Learning
  - Reinforcement Learning
- Why is Data Science Important?
- Overview of methods of Machine Learning.
- What will you learn in this course?
- How do we learn from data?

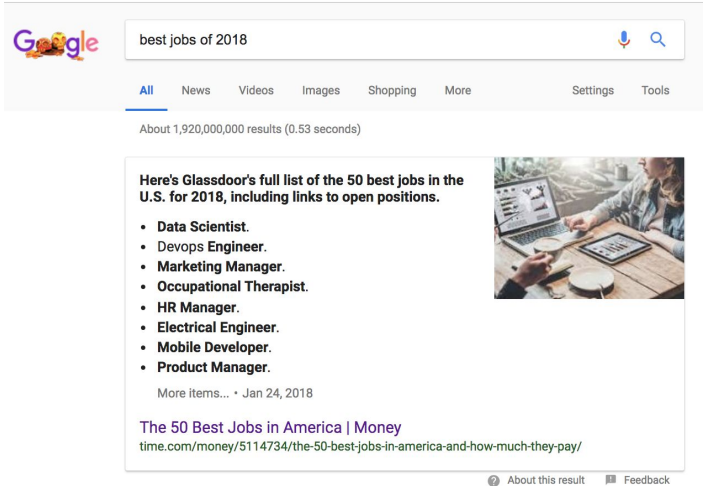
What is Data Science?

- **Supervised Learning:** The science of using data to predict an outcome (clicking, subscription, cancerous cells, price of a stock)
- **Unsupervised Learning:** Using data to group items/users into categories (ie. extract topics/categories from articles)
- **Reinforcement Learning:** Optimizing action based on response variable (ie. who should receive a marketing email, based on sign ups from an experiment, what is a bad Lyft/Uber pickup spot?)
- **Exploratory:** Can we describe characteristics of items/users with particular attributes we are interested in? (ie. Are new users who sign up for the new york times mostly Democrats?)
- **Experimental:** Conduct experiments and interpret their outcome.

- **Goal of this course:** Master the basics from a theoretical and practical viewpoint.

Interest in Data Science Still Exploding





- Intellectually rich landscape of problems in a relatively new field.
- Can save a company millions of dollars by implementing the right algorithm effectively, allowing us to have significant impact.
- Might be relabeled eventually, but will never not be relevant.

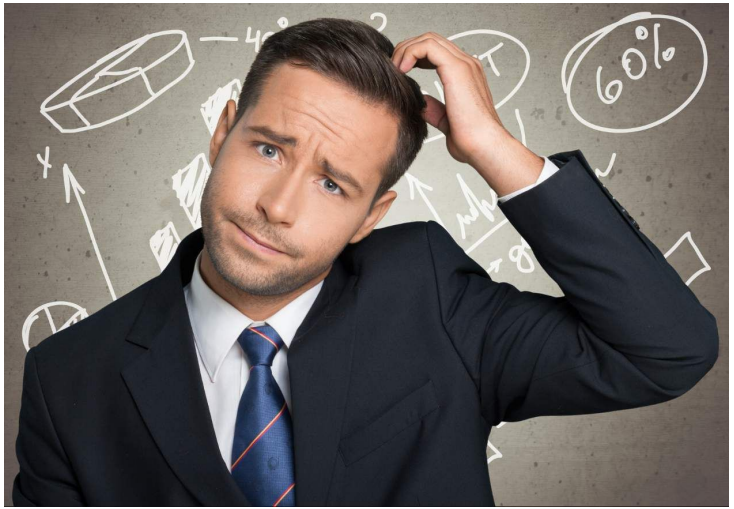
“"Anderson left Harvard before getting his PhD because he came to view the field much as Boykin does—as an intellectual pursuit of diminishing returns. But that’s not the case on the internet. “Implicit in ‘the internet’ is the scope, the coverage of it,” Anderson says. “It makes opportunities are much greater, but it also enriches the challenge space, the problem space. There is intellectual upside.”" - ***WIRED***

https://www.wired.com/2017/01/move-coders-physicists-will-soon-rule-silicon-valley/

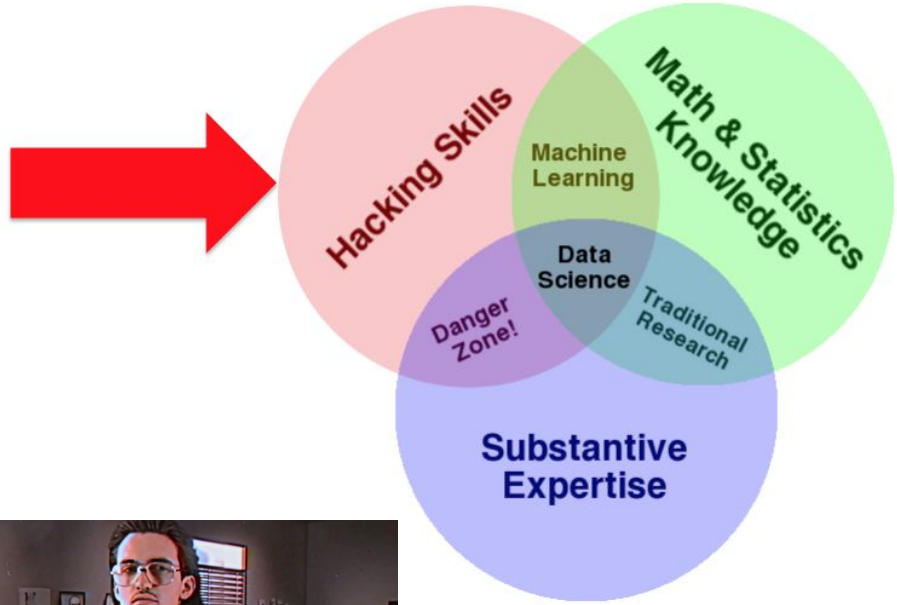
But Data Science is losing its meaning

- Because of the popularity of data science, there are far too many “fake” data scientists.
- More and more candidates are graduating from data science masters programs without being able to answer simple questions about which model to use where.
- Data scientist, as of 2018 is meaning analyst more frequently.
- Don’t be a fake data scientist!

- This course has been slightly adjusted this year to account for these issues.



What is Data Science?

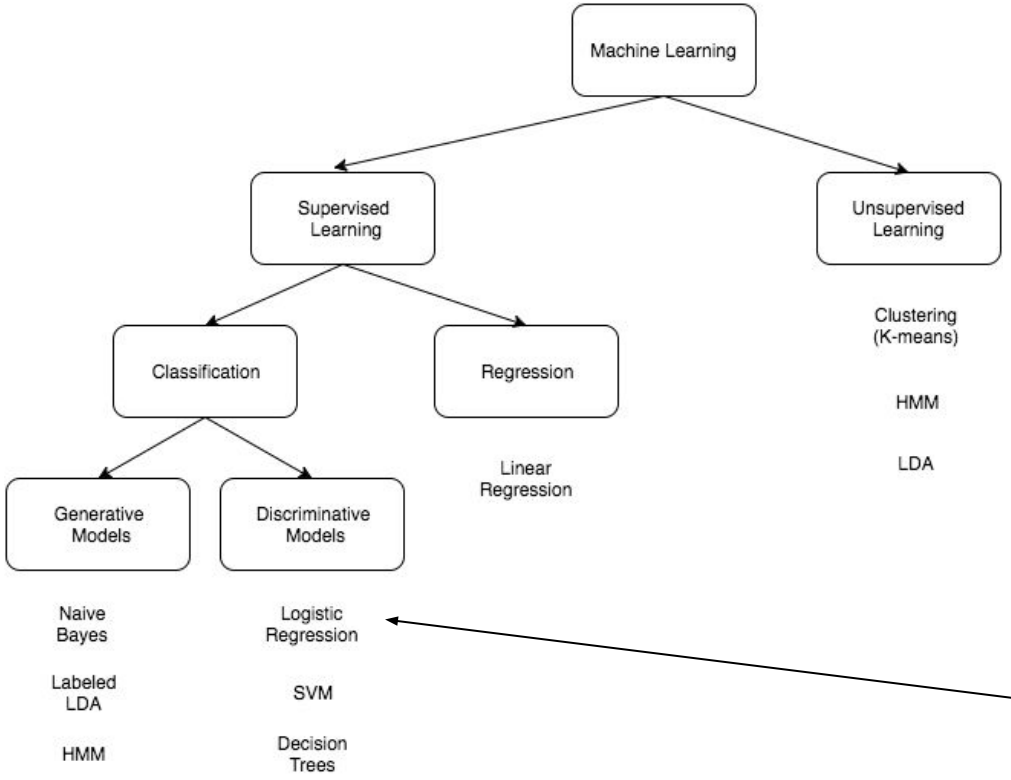


Math/Statistical Knowledge: Need understanding probability, statistics, optimization methods to create and use models. What model is best given the data? How to use it properly?

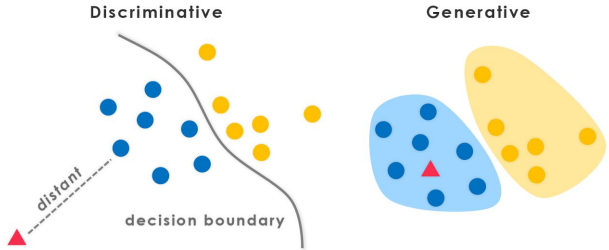
Hacking Skills: Comfort with Linux/Unix, networks, databases, working from the command line, debugging code, Github. Quick solutions.

Substantive Experience: Need experience working with real data and business problems along with the problems that come along with them. Also need ability to communicate technical ideas to stakeholders.

How do we break down machine learning algos?



- Generative models try to understand the probability distribution of the data (x,y)
- Discriminative models try only to understand how classes are separated based on the attributes (more on this later).
- Generally unsupervised algorithms are best used to solve supervised problems.



Supervised Learning

Where do problems arise?

Supervised Learning Topics

- Supervised learning attempts to learn a model from data **X** which predicts a variable **y** (ie. type of movie, number of views would be **X**, **y** is your rating). Supervised means you know what the “right” answers **y**, given context **X.**

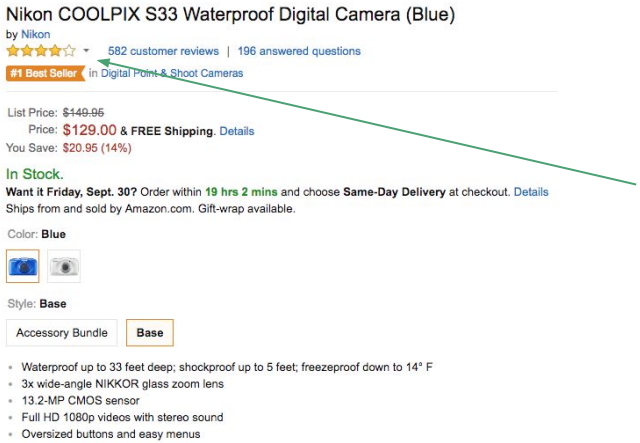
- Learns from data which has **‘correct’** answers given data inputs - this is why it’s **“supervised”**.

- **Topics (you will learn):**

    - Linear Regression (Regression)/Logistic Regression(Classification).
    - Random Forest/Decision Trees/Gradient Boosting.
    - SVM (Support Vector Machines) and nonlinear kernels.
    - Recommendation Engines: Graph Diffusion, Matrix Factorization.
    - Maximum Likelihood
    - Time Series Modeling
    - Reinforcement Learning - Why I think it’s underused currently!
    - Neural Nets

Amazon.com purchases

Can we predict how a user will rate an item? Why do we care?

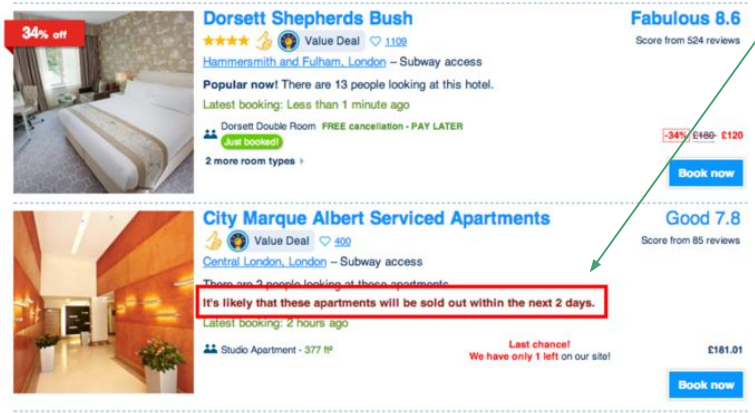


- Can we predict how you would rate this item based on what we know about you?
- **Why do we care? Answer:** Will increase purchase rate and this can be measured in an experiment.
- **Good recommendations = $$.**

1. Generate the model, evaluate.
2. Run A/B test to measure performance or utility.
3. Learn from the model and improve.

Booking.com hotel bookings

Can we predict that a hotel is likely to sell out soon? If so when? Why do we care?



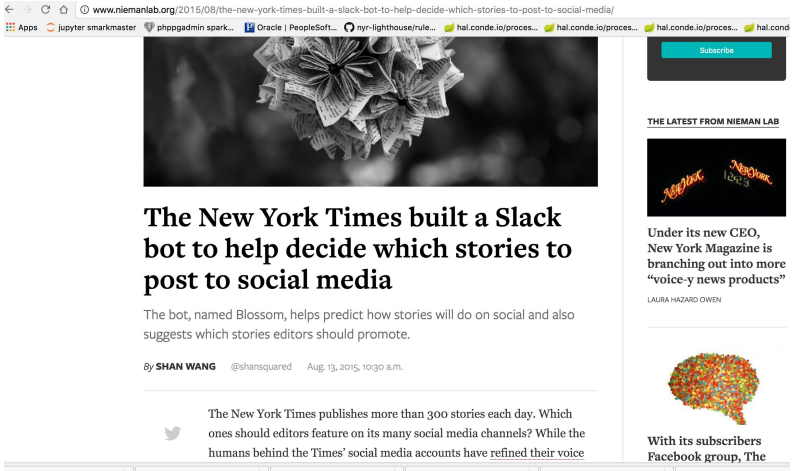
- **Conversion:** Users may be **more inclined to purchase** if they are aware the room may sell out soon (improve purchase rate).
- **Retention:** Users may be **only interested in certain hotel options** and not be aware that they don't have the luxury of waiting - **this could upset customers** if the hotel sells out without warning (customer service).

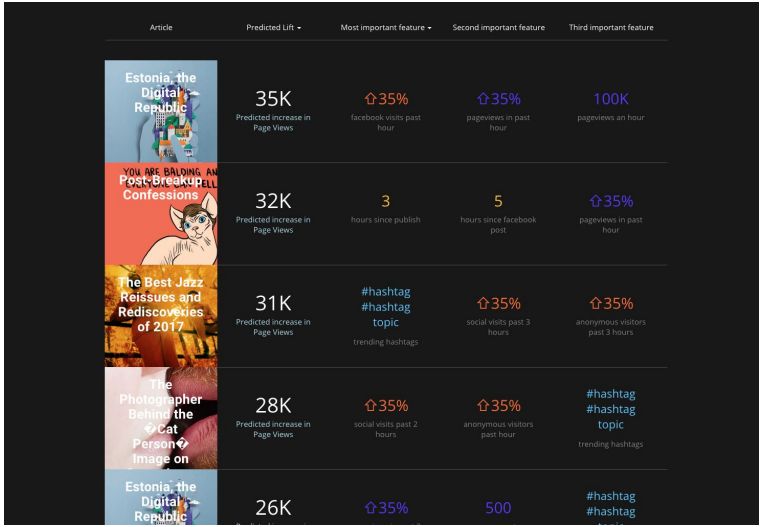
Netflix.com movie ratings

Can we predict how you would rate a movie?

- **Engagement:** Users will be more engaged if movies they are likely to rate highly are shown to them first.
- **Retention:** Engaged customers are loyal customers, which means $$.

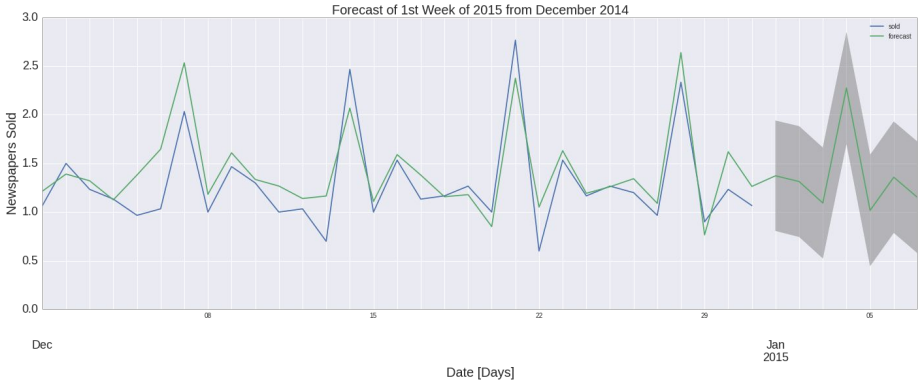
Predicting viral content





- Which content will go viral? (predictive)
- Where is the optimal place to post it? Twitter, Facebook? (prescriptive)

Optimizing paper distribution

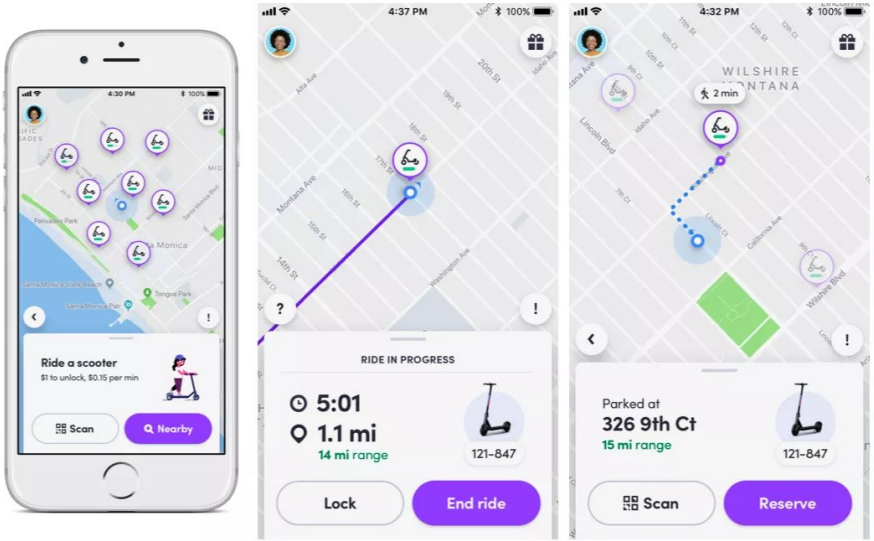


- Can we optimize profits by knowing how many papers to deliver to each Starbucks across America?
- Answer: Yes!
- Problem involves profit optimization, time series regression, maximum likelihood methods and running live experiments to evaluate performance.

TM

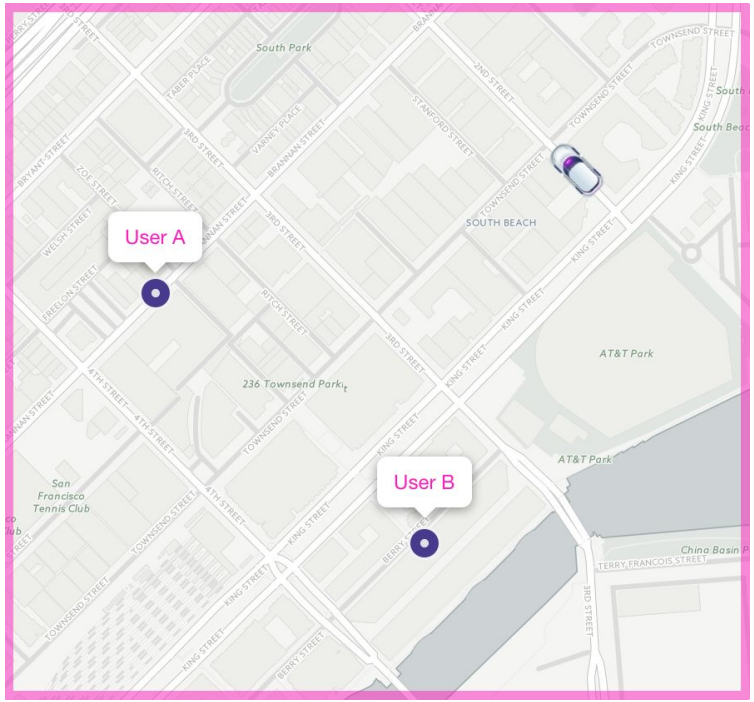
The New York Times

Lyft Pickup Locations

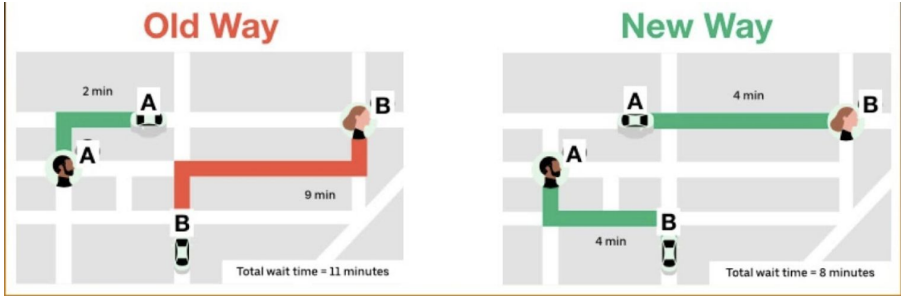


- What defines a “good” pickup spot? For classic rides and shared rides.
- How do we match drivers and passengers?
- How do we compute ETAs?
- If a passenger or driver is likely to cancel, can we predict that before hand?

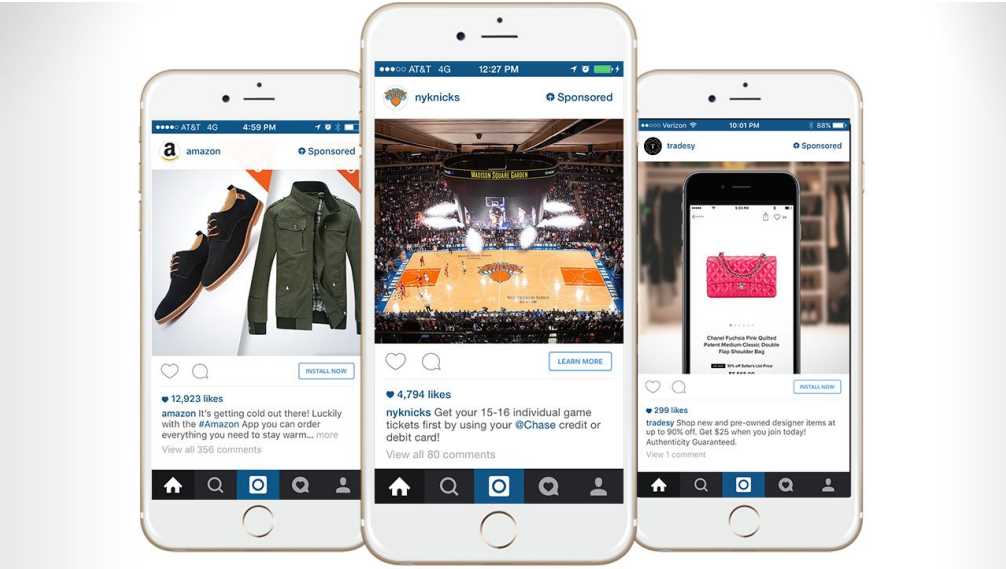
Marketplace Optimization



- Should we match the driver to user A or user B?
- This turns out to be a very complex and interesting optimization problem.



Instagram Ads



- Ever noticed how you will start receiving ads for shoes after going into a shoe store?
- We will learn how this works.

Modern Artificial Intelligence

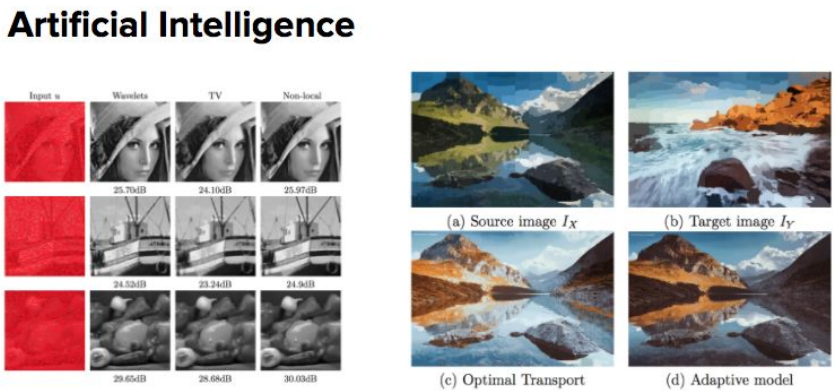
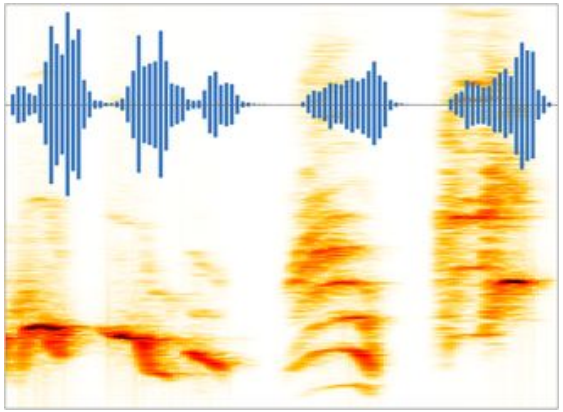


Image classification, segmentation, denoising, and many more applications! (Probably not at the level of this course but we will see!



How do we test our model? A/B Testing

- Our model suggestions →
- Top items (BAU) →

How do we test our model? A/B Testing

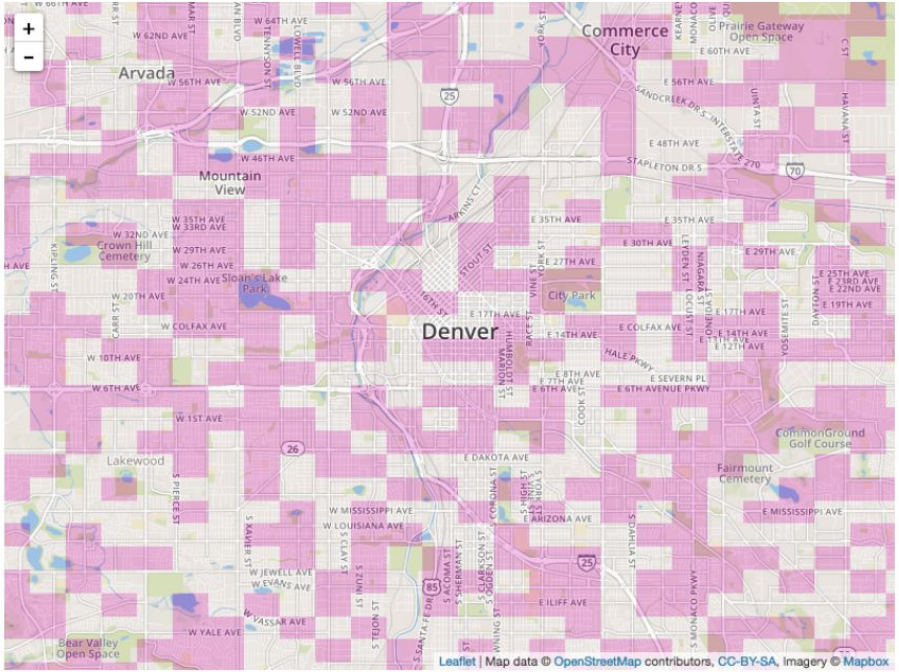


Figure 1. Spatial randomization at the geohash-5 (top) and geohash-6 (bottom) levels, for a randomly chosen Lyft market (Denver, CO). In this realization of the designs, pink cells might be given the hypothetical treatment while remaining cells would get the control.

- A/B testing isn't always so straightforward though!
- If Uber/Lyft wanted to test out a prime time model, could they just randomly assign users into a test/control group?
- **Answer:** No. There would be interference!

More information:
https://eng.lyft.com/experimentation-in-a-ridesharing-marketplace-f75a9c4fcf01

Supervised Learning

Basic methodology

Supervised Learning - Problem Statement

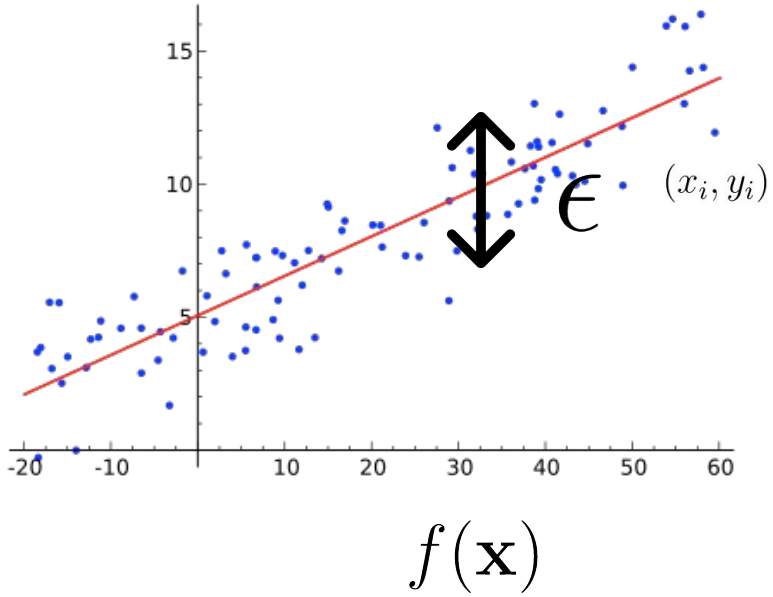
We assume that y, the response variable, is a function of covariates **X** (ie. variables, features).

$$y = f(\mathbf{x}) + \epsilon$$

Systematic information that x provides about y

Irreducible error

- The irreducible error ϵ <u>is natural and cannot be avoided</u>. For example, for any given attributes of an individual, height will have a natural variance. Or predicting income given someone's education, GPA, SAT scores, etc (assuming we chose the right distribution)



Supervised Learning - Problem Statement

We assume that y, the response variable, is a function of covariates **X** (ie. variables, features).

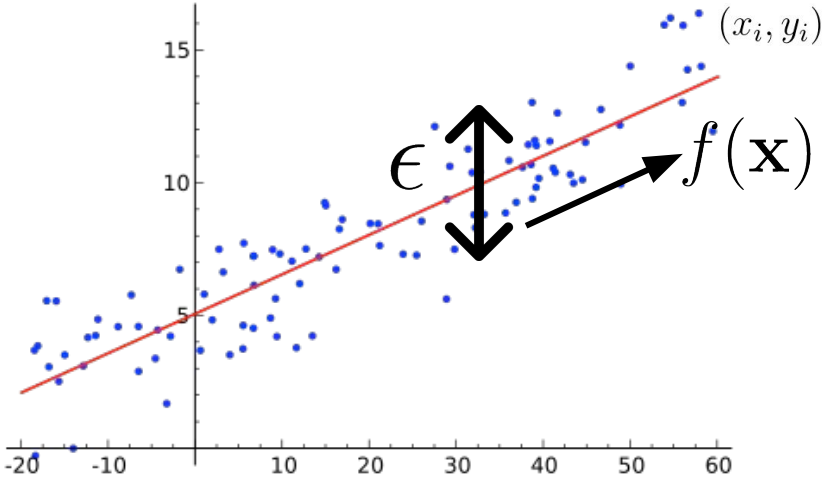
$$y = f(\mathbf{x}) + \epsilon$$

$\hat{Y}$ is our *estimator*. Our best guess at f given data.

$$\mathbb{E}(Y - \hat{Y})^2 = \mathbb{E}(f(\mathbf{X}) + \epsilon - \hat{Y})^2$$

$$= \mathbb{E}(f(\mathbf{X}) - \hat{Y})^2 + 2\mathbb{E}(\epsilon(f(\mathbf{X}) - \hat{Y})) + \mathbb{E}(\epsilon^2)$$

$$= \underbrace{\mathbb{E}(f(\mathbf{X}) - \hat{Y})^2}_{\text{reducible}} + \underbrace{\text{Var}(\epsilon)}_{\text{irreducible}}$$



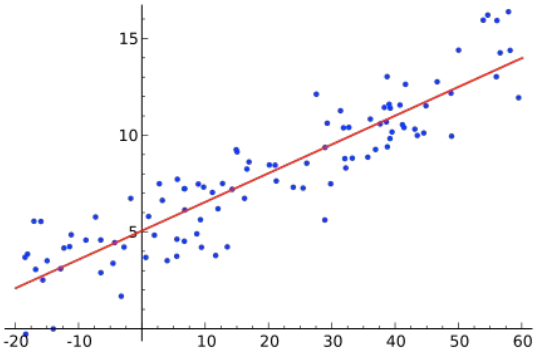
We cannot do anything to reduce ϵ, so we need to find the best $\hat{Y}$

Supervised Learning - Regression

Given a collection of points to learn from: (x_i, y_i)

Can we find a function $f : X \to Y$ which minimizes the distance to the data, ie:

$$\mathbb{E}(f(\mathbf{X}) - \hat{Y})^2 \sim \frac{1}{N}\sum_{i=1}^{N}(f(\mathbf{x_i}) - y_i)^2$$

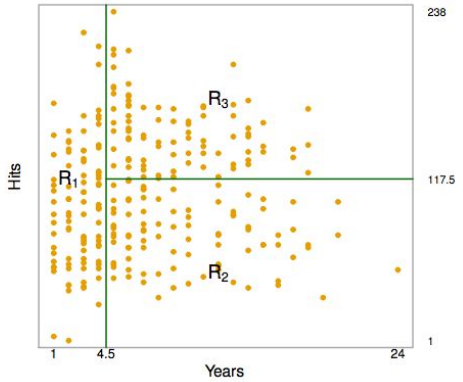


All of predictive machine learning is based on discovering ways to find f .

Regression Examples for f

Linear: $f(\mathbf{x}_i) = \beta \cdot \mathbf{x}_i$

Trees (Nonlinear): $f(x_i) = c_t \mathbf{1}_{R_t(i)}(x_i)$ for $x_i \in R_{t(i)}$

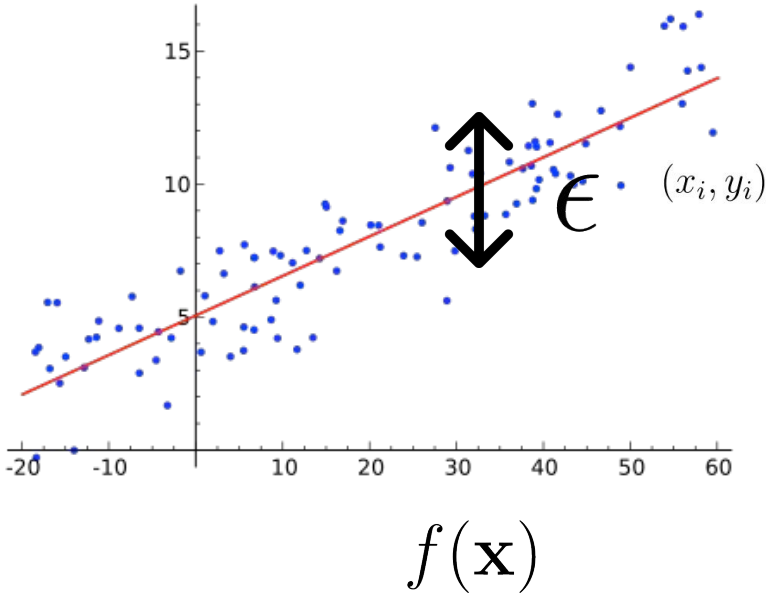


Supervised Learning - Problem Statement

We assume that y, the response variable, is a function of covariates **X** (ie. variables, features).

$$y = f(\mathbf{x}) + \epsilon$$

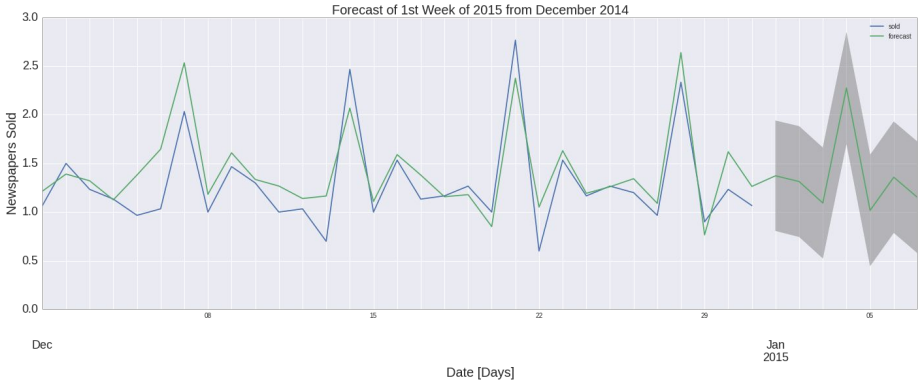
Our goal is to find an *estimator* $\hat{Y} = \hat{f}(\mathbf{X})$



- Our estimate of f will generally not be perfect, so we will have errors:
  - $f - \hat{f}$ which is known as **reducible error.**
  - ϵ which is known as **irreducible error.**

Goal: Our goal is to find the best f and to understand ϵ

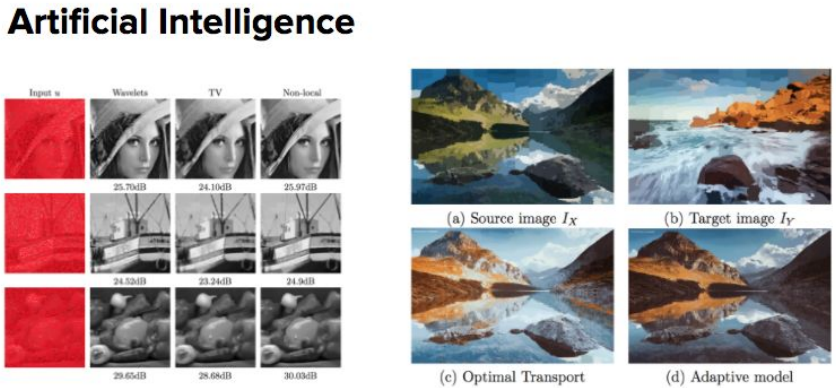
Supervised Learning - Time Series



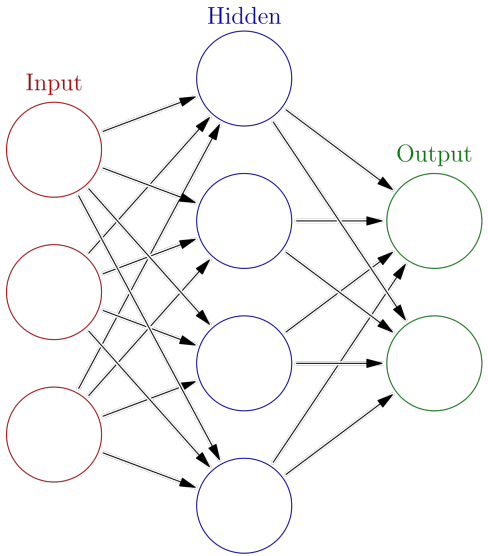
- The values at time t+1 are predicted using the values up to and including time t.

$$\mathbf{X}^t = \sum_{k=1}^{T} \alpha_k \mathbf{X}^{t-k} + \epsilon$$

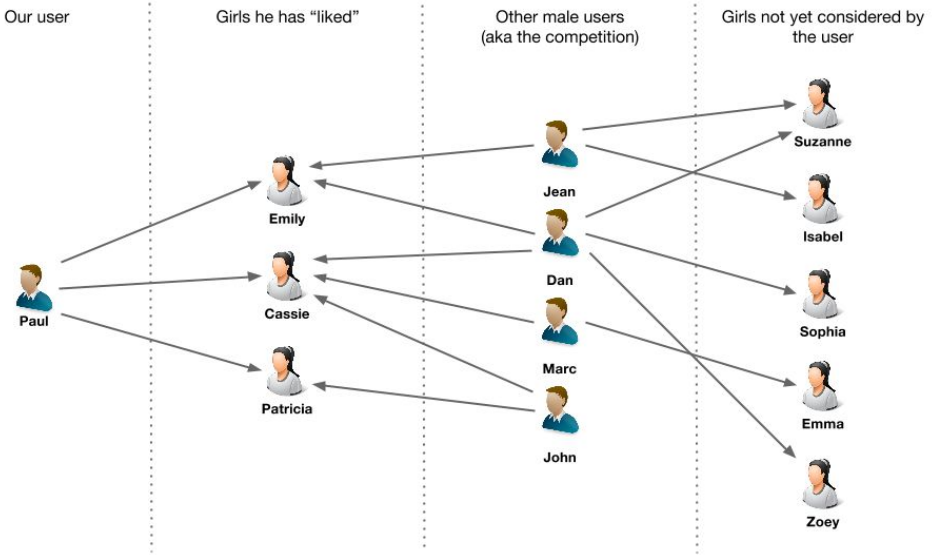
Supervised Learning - Deep Learning



Solving problems that humans can’t currently !

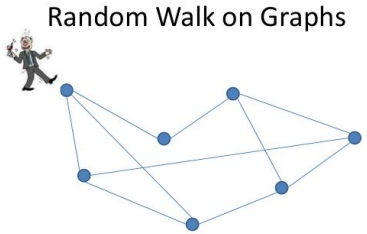


Supervised Learning - Recommendation Engines



$$\pi(j,n) := \sum_{j',n'} p(n|j')p(j|n')q_{n'} = M^T G q_j,$$

- Person j for Person n has a propensity measured by a bipartite graph model.

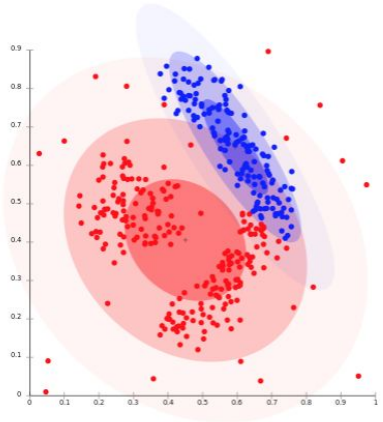


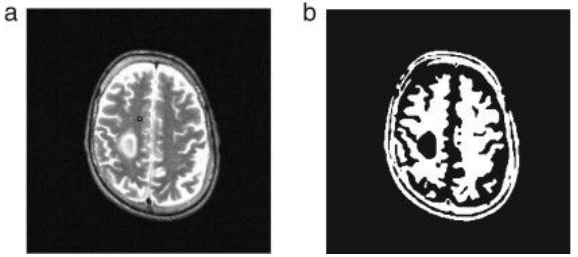
Unsupervised Learning

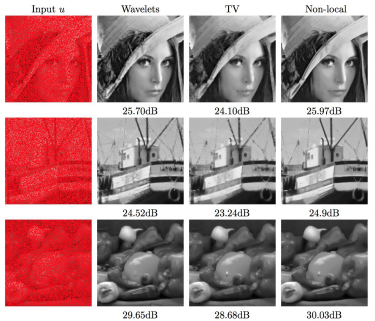
Where do problems arise?

Unsupervised Learning - Summary

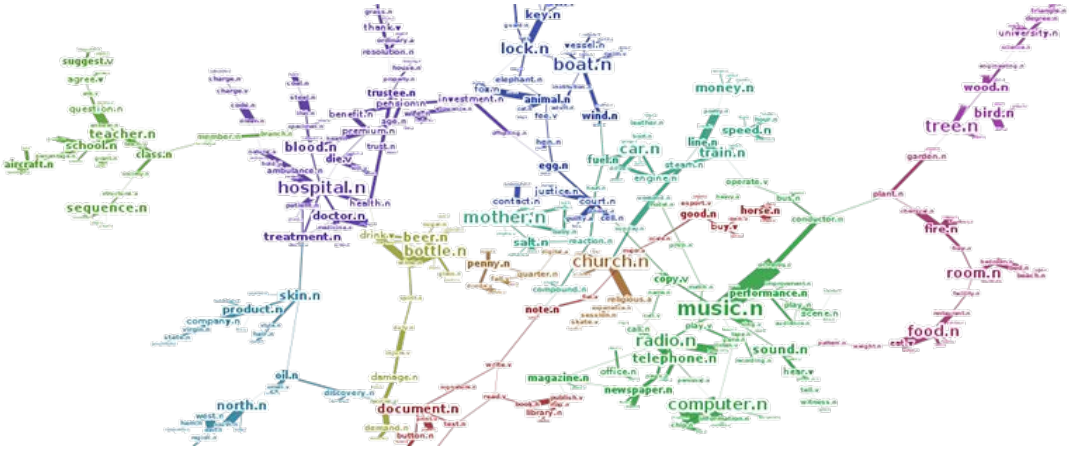
- Try to infer a hidden structure in the data without proper training examples (no teacher, hence ‘unsupervised’).
- This course will focus less on unsupervised learning.
- **Algorithms:**
  - K-means clustering.
  - Decision Tree Clustering.
  - Support Vector Machines
  - Topic Models (LDA, etc)
  - Gaussian Mixture Models and Expectation Maximization







Topic Models

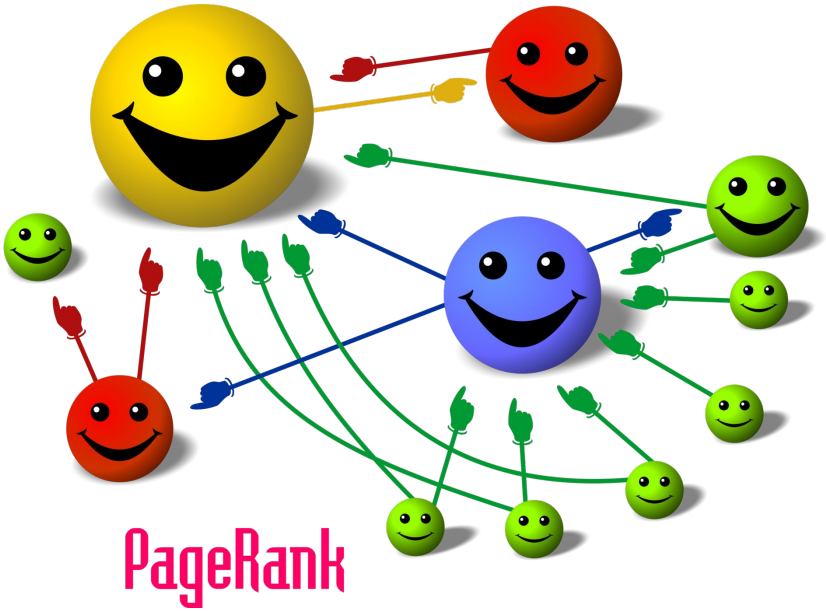


the dog is on the table

0	0	1	1	0	1	1	1
are	cat	dog	is	now	on	table	the

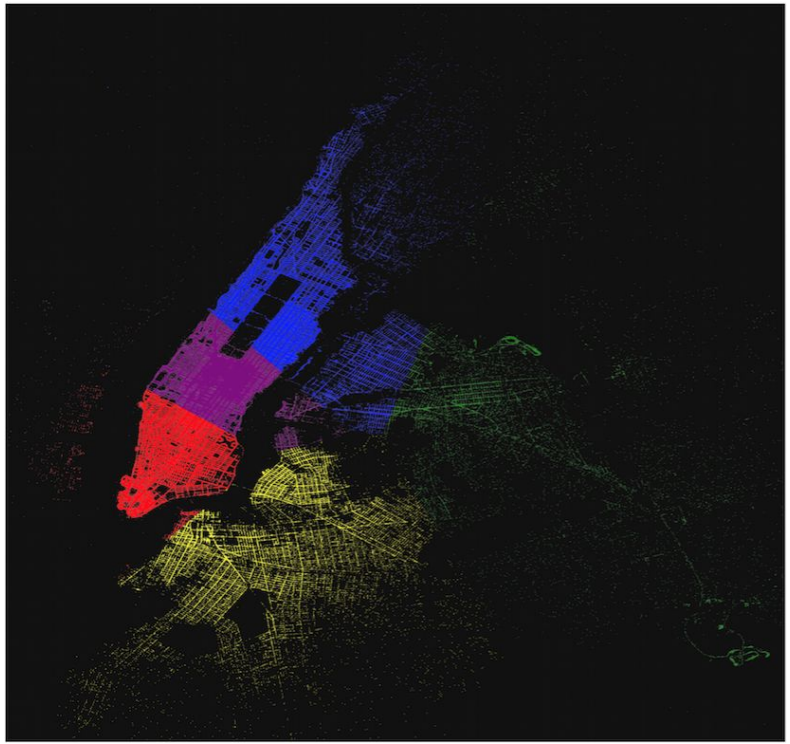
- Topic models attempt to cluster content into a finite collection of ‘topics’.
- The model creates topic categories by clustering words commonly occurring together into groups (roughly speaking).
- Reading/Writing behavior, while unsupervised, has tremendous predictive power for many algorithms in practice.

Google’s PageRank



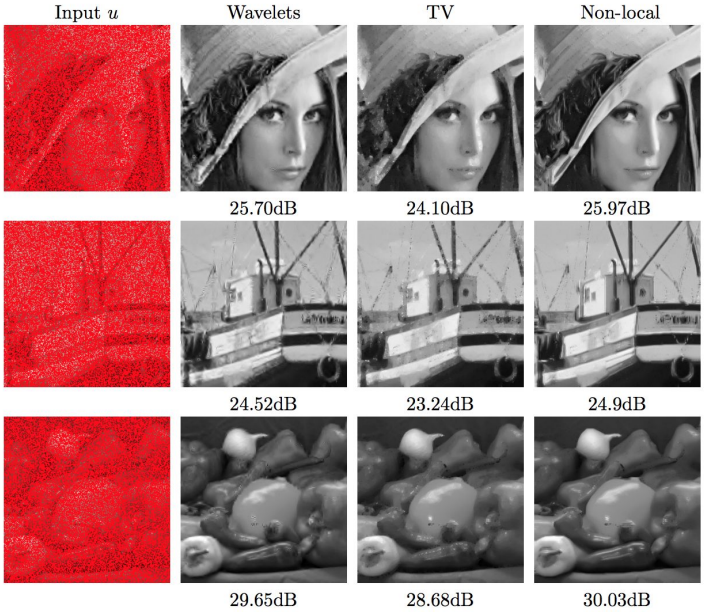
- Cartoon illustrating the basic principle of PageRank.
- The size of each face is proportional to the total size of the other faces which are pointing to it. (Source: Wiki)

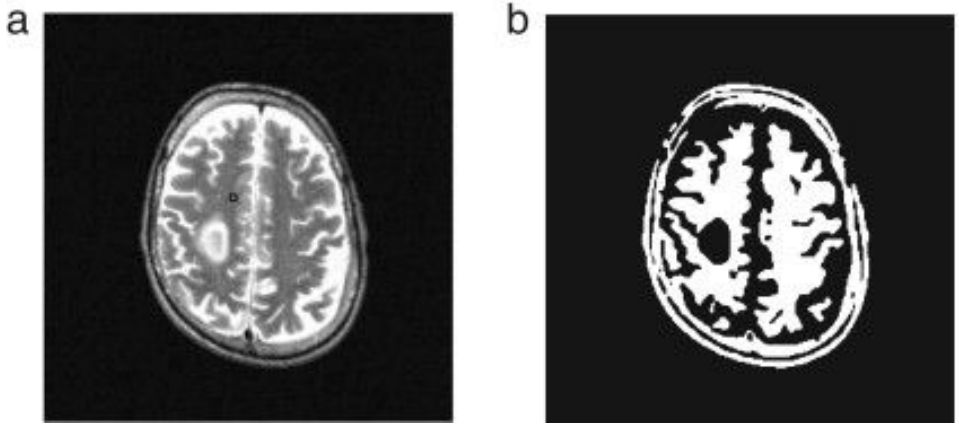
NYC Taxi Pickup Clusters



- K means clustering on NYC taxi pickup locations
- Travel statistics between clusters can be used in various supervised models.

Denoising and MRI lesion detection





Clustering is applied to identifying lesions in the brain and to denoising images.

Reinforcement Learning

Learning through experience

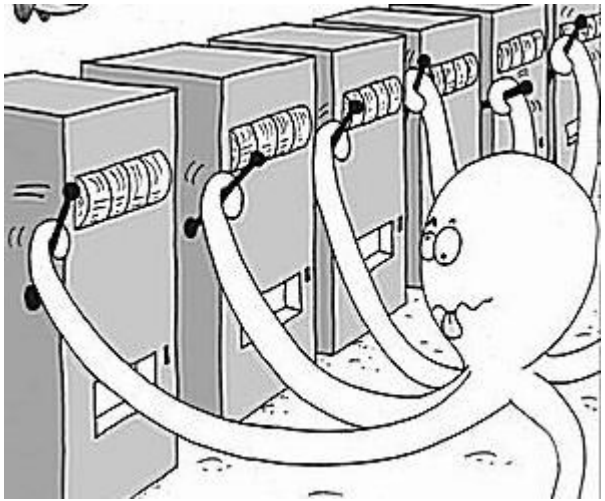
Reinforcement Learning - Summary

- Prescriptive learning attempts to find an **optimal action** to **maximize the expected reward/outcome** (ie. who should we show this kind of ad to?, or who should we send this marketing email to?, what's the next chess move?).

- **Advantages**
  - Relies entirely on maximizing expectation of reward conditioned on contextual attributes and the action chosen.
  - Incredibly useful since it's actionable and maximizes your objective directly.
  - Can be "live" (multiarmed bandit) or from logged data (uplift modeling).
  - Combines both experimental observation with supervised learning, to maximize a reward.

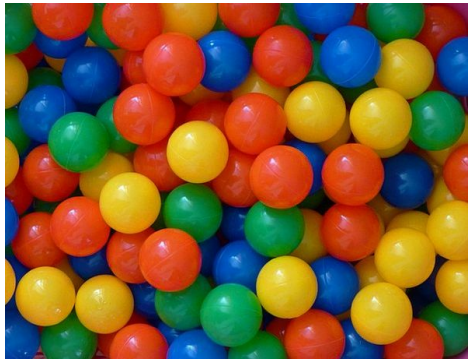
Example: Multi-armed Bandits

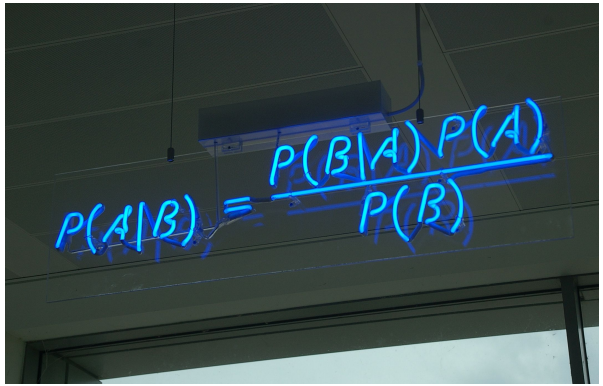
For a new user with no data, how do we predict what you should see?



- If you were to play 5 slot machines, one of which was the best (but you don't know). How would you balance **exploration** with **exploitation**?
- Multiarmed Bandits do this in a rigorous mathematical fashion.

Example: Belief updating

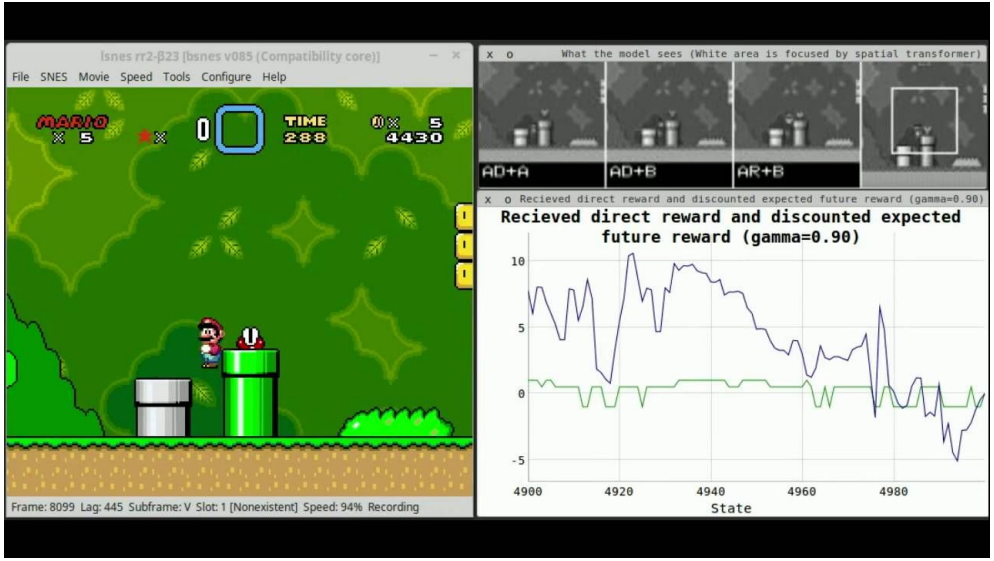




Your friend tells you that he will give you $100 if you can guess the right number of red balls after 10 tries, in a pit of 20 balls.

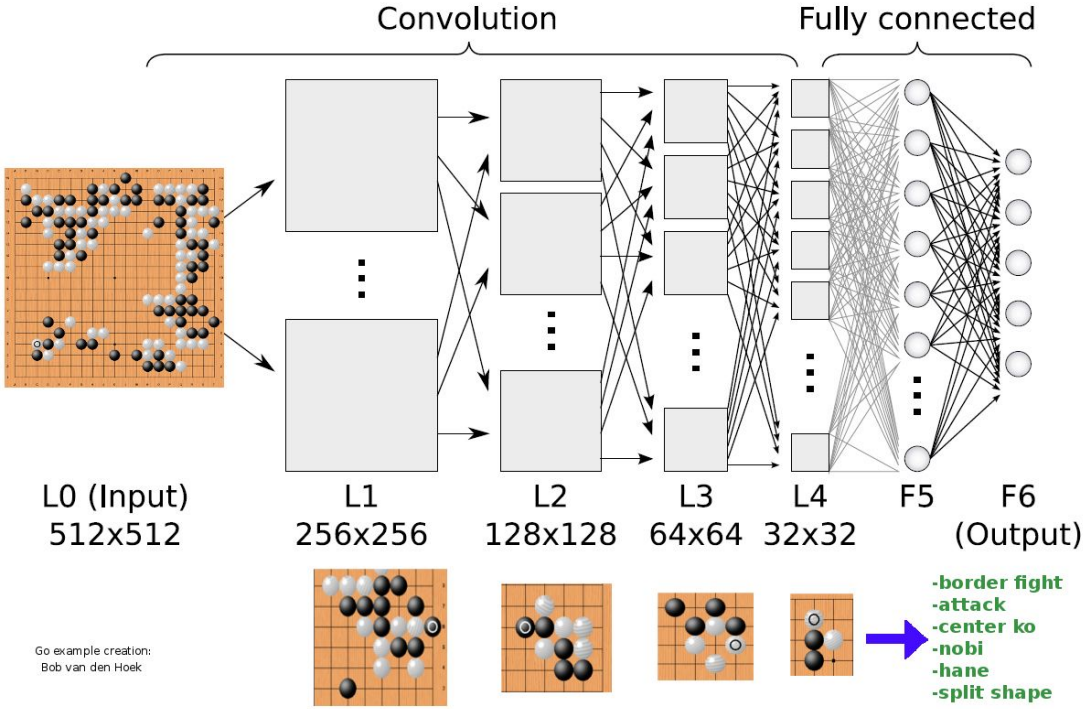
- You start off with an initial guess.
- Each time you take a ball out of the pit, you observe it, and put it back in.
- How does your belief change after observation?
- How confident are you?
- **This is how humans learn!**

Example: Learning to play video games



Example: Alpha Go

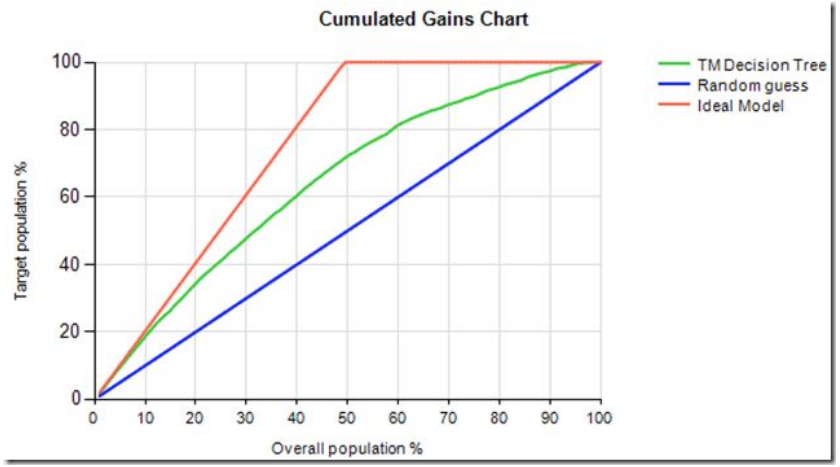
A deep reinforcement learning approach recently beat the world champion at Go.



Example: Uplift Modeling

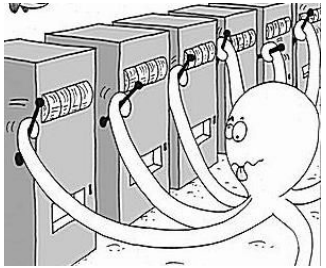
- Not everyone should receive the same action.
- Will users leave buy if they receive an offer?

How do we determine the right action to maximize our desired outcome?



Percentage of people who bought

Mathematical Formulation



- Given a collection of outcomes, features and actions, how do we find the optimal action that a given user should receive to maximize the expected outcome for the next iteration?

- In other words, given a distribution of rewards, how do we maximize:

1. Given current reward distribution and context, choose action to maximize expected reward
   $$a_t = \text{argmax}_a \mathbb{E}_{R_{t-1}}(R_t | X_{t-1}, a)$$
2. Take action a_t
3. Observe reward R_t and update distribution of rewards. Then repeat from step 1.

What will you learn in this course?

What will you learn in this course? (hopefully!)

- Most common methods used in **Machine Learning** and how they are used in industry, more precisely, **how to build models from data**. Some examples we will cover are:
  - Recommendation engines (how do we deliver content meant for you?)
  - Predicting virality churn, acquisition, etc.
  - ETA forecasting
  - Time Series analysis - how do we predict what will happen at a future time based on the past? (Related to stock forecasting, paper distribution, etc).
  - Experimental Design: Understanding how our models work (or don't!).
- **Methods of Machine Learning:**
  - Regression/Classification: Linear and Nonlinear modeling approaches
  - Model complexity and regularization (variance/bias tradeoff). Cross validation.
  - Graph Diffusion, collaborative filtering, random walks. Graphical models.
  - Bayesian inference. Reinforcement learning.
  - Unsupervised learning. Clustering. Expectation maximization.
  - Time Series Analysis. Autoregression. Poisson Regression, etc.
  - Neural Nets
- **Map Reduce/SQL and Data Engineering:** Will learn why and how we use distributed computing for processing data.
- **Build your own web app:** By the end of the class, the final project will be to build your own web app using the tools covered in this class.

What you will also learn

Everything I present here has an intended purpose. Each homework exercise or topic is related to something I need on a daily basis for my work, or related to a job interview question I received in the past. My goal is that if you can do well in this class, you will do well on the job market.

Most common interview topics:

- Model Selection/Regularization
- Central Limit Theorem
- Stochastic Gradient Descent
- Coding Efficiency
- Differences between linear and nonlinear models. When does one have an advantage?
- Sampling Methods

How to gain the most from this class

“The only way to learn a quantitative subject is to struggle through difficult problems yourself. One has to discover all of the wrong ways of approaching a problem and how to learn all of the right questions to ask in order to find the right one. In this sense, you should see me as a tour guide who will guide you through a landscape of concepts of varying relevance/importance. But it is up to you to explore and understand.

Dror Bar-Natan, University of Toronto

“If there is a problem you cannot solve, then find an easier problem that you can solve. Once you’ve done that, try the harder problem again.

Charles Pugh, University of California, Berkeley

What should you know?

- Undergraduate math - linear algebra, probability, calculus III, statistics.
- Background in programming (ideally Python).
- Basic unix commands are good to know.
- However many people from last year had no CS background and still managed to do well!
- Homework 0 will ensure people are up to the level of the class. Covers basic linear algebra, multivariable calculus and probability.

Warning!

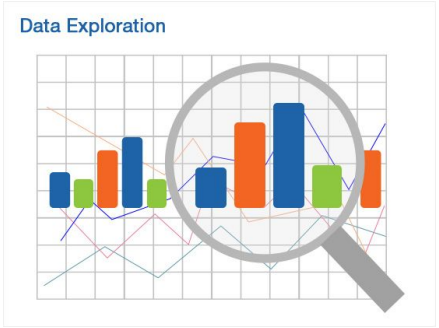
- This year the course will be much more rigorous!
- In previous years, I found that many students that did well in the class did not perform well on interviews. I see that as a failure on my part.
- Final grades will be similar, but you will be pushed harder in this year's version of the course. **Please be prepared to review advanced calculus, linear algebra and statistics.**

How do we ‘learn’ from data?

Is it even possible?

General Model Construction

Your intuition and data exploration leads you to believe certain variables should be related to what you're trying to predict



The data has to then be gather, processed, cleaned and aggregated into a final dataset.

amazon web services

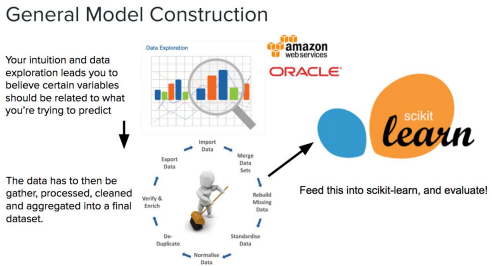
ORACLE

scikit learn

Feed this into scikit-learn, and evaluate!

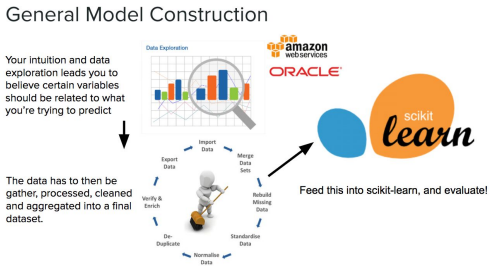
Example 1: MINST dataset

- Can you recognize these digits? Why?
- You most likely have a notion of a ‘6’ or ‘7’ in your mind - what allows to you classify these? Mostly experience. How do we mimic ‘experience’ with data?
- How do we convert this into a collection of features?
- How do we model the data?

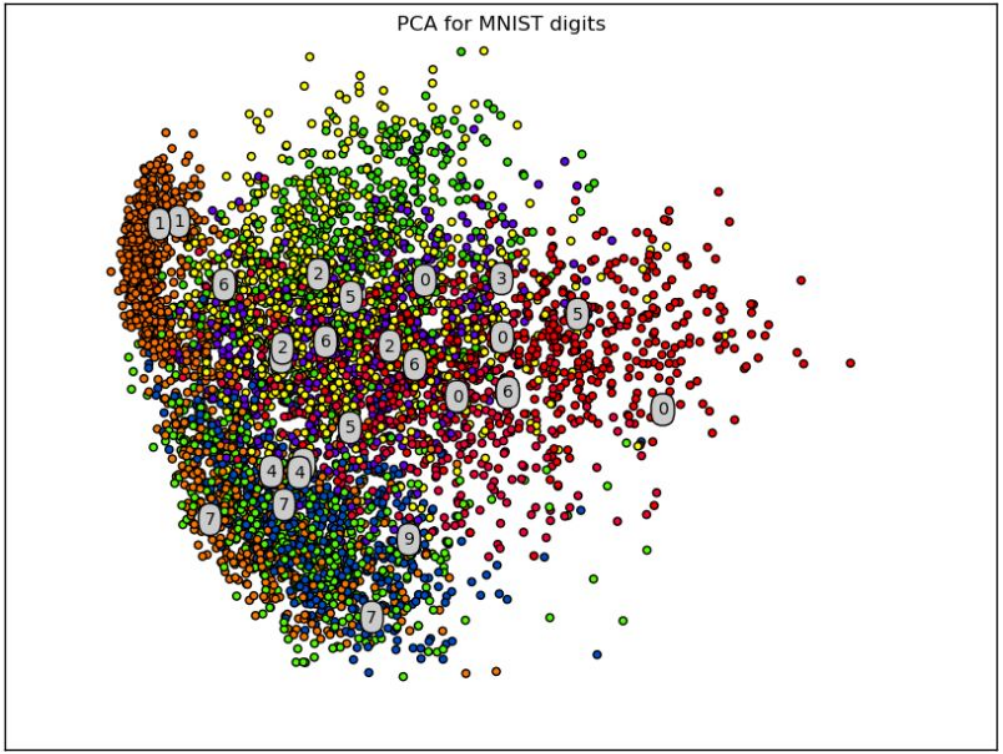


Example 1: MINST dataset

- Can you recognize these digits? Why?
- You most likely have a notion of a ‘6’ or ‘7’ in your mind - what allows to you classify these? Mostly experience. How do we mimic ‘experience’ with data?
- How do we convert this into a collection of features?
- How do we model the data?
    - **Data: Features**- have a unique feature for each pixel in an 8x8 image (64 features total).
    - **Model: K Nearest Neighbors**- based on the above features, which images am I “closest” to?
    - **Improvement: Dimensionality Reduction**- reduce complexity to optimize for test set performance.

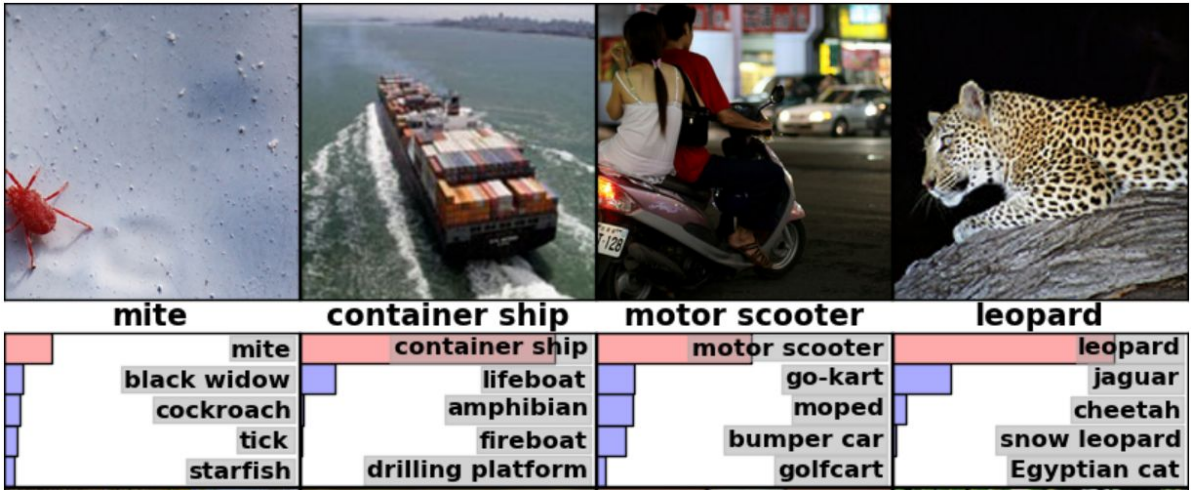


Visualization of the top two components



- Here we see a clustering of the most ‘significant’ components extracted from the 64 pixels/features (*will be explained later via PCA*).
- Take a fixed circle around each point, look at the K nearest neighbors, and take the majority vote.

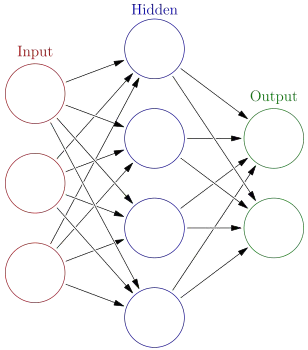
Can we recognize more advanced images?



- Taken from Tensor Flow
- https://www.tensorflow.org/tutorials/image_recognition/

- Modern neural nets are able to classify objects better than humans can in some instances!

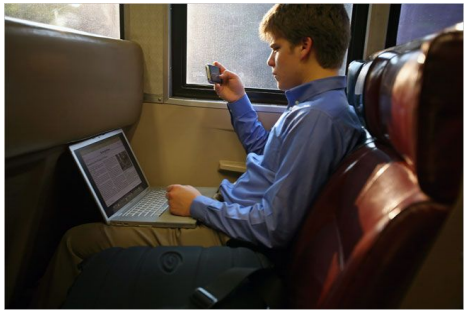
- Current model in Tensor Flow uses what is called a ‘convolutional neural network’ (to be explained later)



Example 2: User churn at New York Times

Question: What would you guess are the features/variables most predictive of user churn?

Build hypotheses for features



Behavioral

Examples:

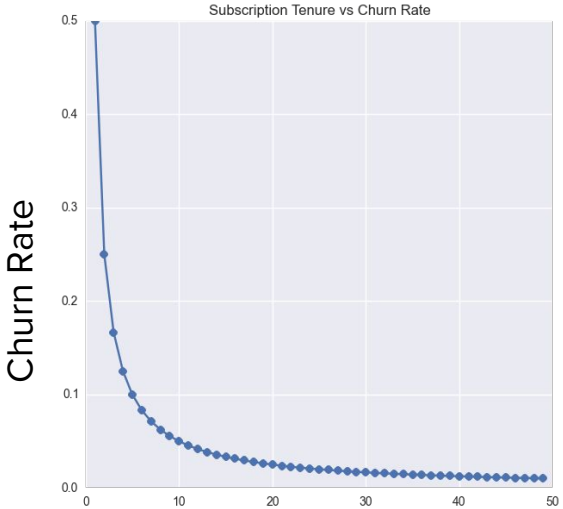
- Online activity. Which sections?
- Subscription length.
- Frequency of reading.

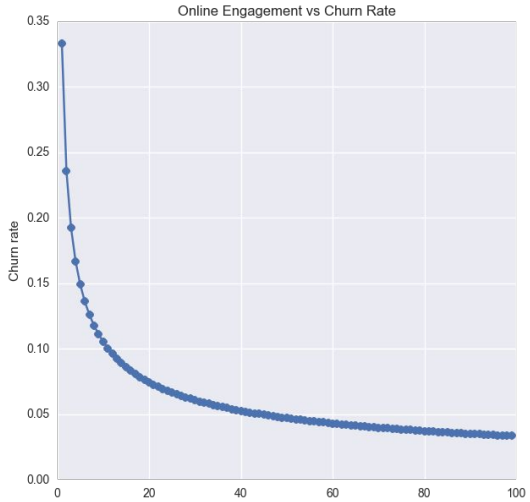
Geographical

Examples:

- Urban city or suburbs?
- Apartment or house?
- Liberal or republican state?

Example 2: User churn at New York Times





- **Subscription tenure** and **Online Engagement** seem to be related to **user churn.**
- Let's create features:
  - days_subscriber,
  - articles viewed

After our hypothesis, we check to see if there is a relationship between our variable and the churn rate.

amazon web services

ORACLE

The New York Times

```
In [78]: df = pd.DataFrame({'days_subscriber':days_subscriber,'articles_viewed':articles_viewed, 'outcome':outcome})
In [79]: df
```

Out[79]:

	articles_viewed	days_subscriber	outcome
0	21	62	1
1	20	210	0
2	15	49	0
3	22	283	0
4	22	8	0
5	5	37	1
6	25	268	0
7	27	152	0
8	21	229	0

ORACLE

amazon web services

- **Step 1:**
  - Explore Data.
  - Find relationships, develop features
  - Investigate relationships between
  - data sources.

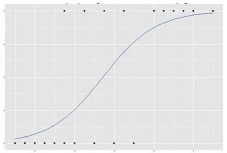
- **Step 2:**
  - Merge data sources, create dataframe.
  - Remove outliers, clean corrupt data, null entries, causal features.

- **Step 3:**
  - Train/fit models
  - Evaluate performance
  - Optimize, investigate.

scikit learn

$$p_{\text{churn}} = \frac{1}{1 + \exp(-\beta_0 \cdot \text{days-active} - \beta_1 \cdot \text{pages-viewed})}$$

Make predictions



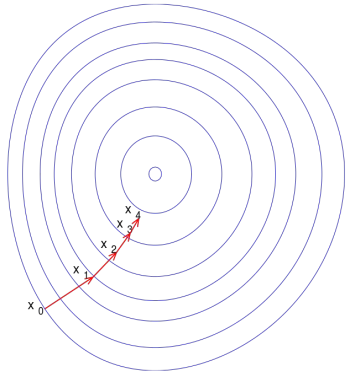
How do we learn the parameter ?

The logistic regression model:

$$p_\beta(\mathbf{x}) = \frac{1}{1+\exp(-\beta^T \cdot \mathbf{x})}$$

$$\text{Cost}_2(p_\beta(\mathbf{x}), y) = \begin{cases} -\log p_\beta(\mathbf{x}) & \text{if } y \text{ is } 1 \\ -\log(1-p_\beta(\mathbf{x})) & \text{if } y \text{ is } 0 \end{cases}$$

$$\min_\beta \frac{1}{N}\sum_{i=1}^{N} \text{Cost}_2(p_\beta(\mathbf{x}_i), y_i)$$



We minimize this convex cost function via gradient descent.

- When y is 1, $p_\beta(\mathbf{x})$ tries to be close to 1
- When y is 0, $p_\beta(\mathbf{x})$ tries to be close to 0.
- Will be explained in later lectures.

$$\mathbf{x} = [\text{days\_active}, \text{pages\_viewed}]$$
$$y = 1 \text{ if churned}$$
$$y = 0 \text{ if retained}$$

Summary

- We always explore the data, find relationships, then develop features.
- We then merge data sources, create a dataframe and try out various models.
- We showed the example of K nearest neighbors for MINST and a linear model for churn, but there are many others which we will cover!
- **But how do we evaluate?**

How do we measure performance?

How complex of a model is ideal? What does ideal mean?

How do we measure performance?

Regression:
(examples)

$$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2}$$

RSME

$$1 - \frac{\sum_{i=1}^{N}(f(x_i) - y_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

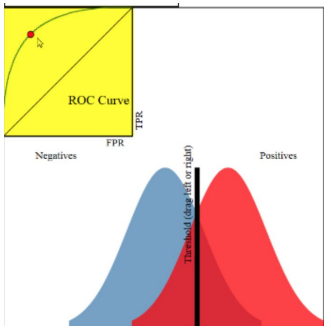
R^2

Classification:
(examples)

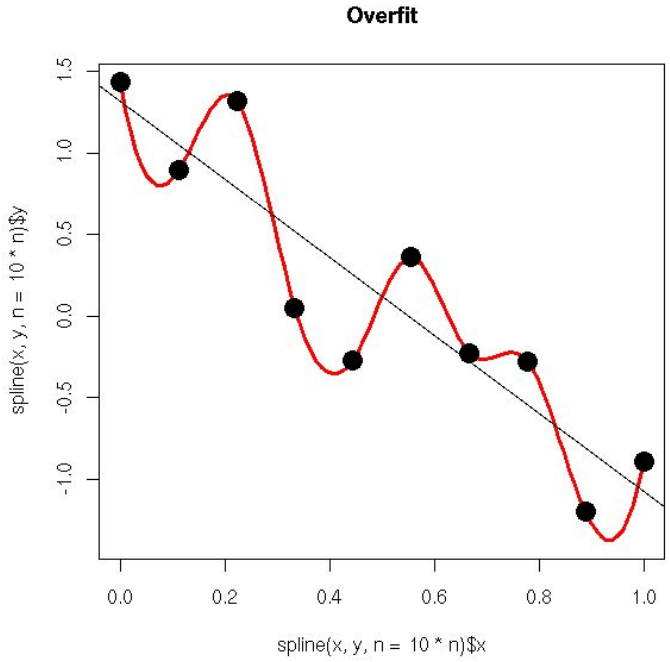
$$\frac{1}{N}\sum_{i=1}^{N}\mathbf{1}(y_i = f(x_i))$$

accuracy

		Actual Value (as confirmed by experiment)	
		positives	negatives
Predicted Value (predicted by the test)	positives	**TP** True Positive	**FP** False Positive
	negatives	**FN** False Negative	**TN** True Negative

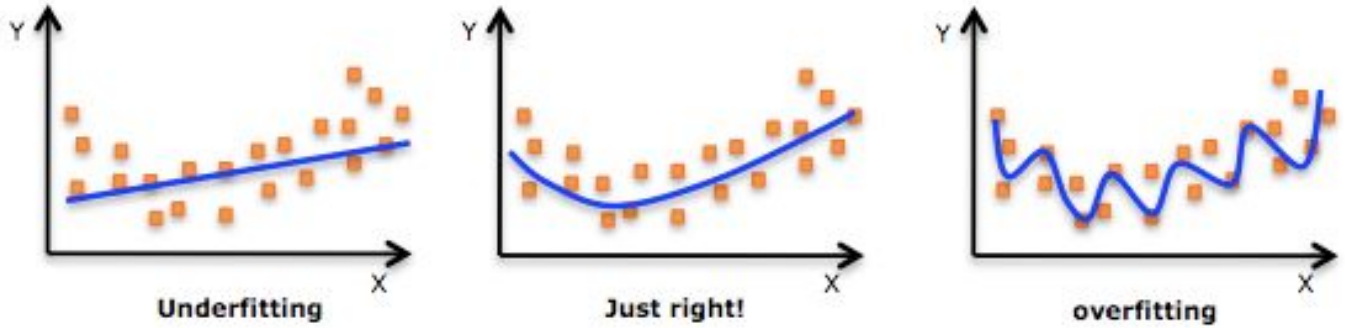


What's wrong with this picture?

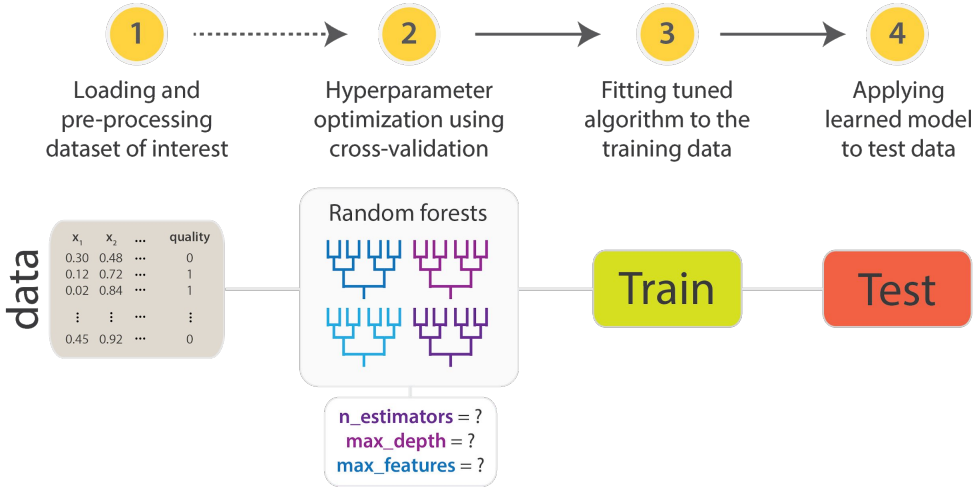


- The model seems to fit the data perfectly! Looks like we found a great model.
- What is wrong, and can you explain how we fix it?

Choosing the right model complexity



How do we evaluate performance properly?

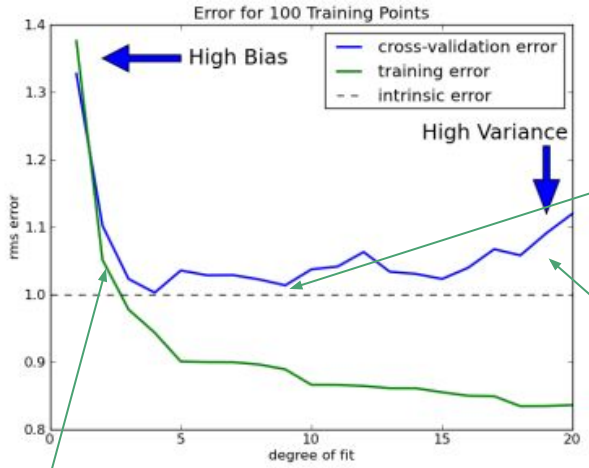


- We we train the model on **different data than we evaluate it.**
- The training data and testing data are sampled from similar distributions.
- We optimize the complexity of the model to prevent overfitting.

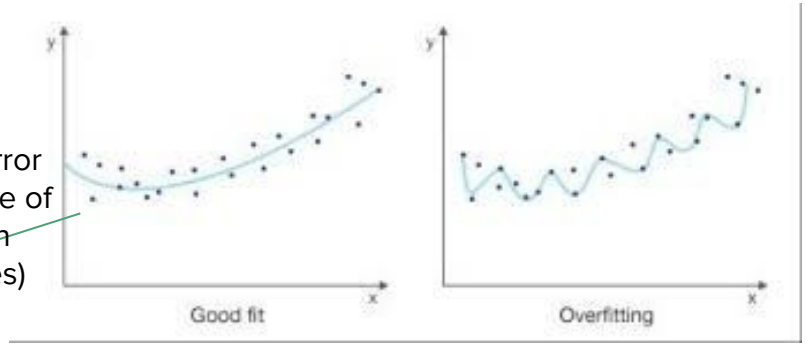
Cross Validation: The idea of holding out a **test set** to measure the model trained on your **training set.**

Parameter Optimization: Optimizing the performance of your model on the test data.

What complexity is ideal?



The lowest point here on testing error is our ideal degree of fit (depth of tree in previous examples)



We've gone too far here, since our testing terror has started to increase. This means there is too much variance.

Our model here is too simple, and has too high of a bias.

The secret truth about data science

- The hardest part of being a good data scientist, and where most of your effort is put, is dealing with difficult, unreliable, messy data sources.
- One needs the scientific rigor of a physicist, along with the hacking skills of a computer scientist to truly be able to develop effective algorithms.
- 95% of your time as a data scientist is dealing with data collection, integrity analysis and cleansing.

Python For Data Science *Cheat Sheet*

Scikit-Learn

Learn Python for data science Interactively at www.DataCamp.com

Scikit-learn

Scikit-learn is an open source Python library that implements a range of machine learning, preprocessing, cross-validation and visualization algorithms using a unified interface.

A Basic Example

```
>>> from sklearn import neighbors, datasets, preprocessing
>>> from sklearn.cross_validation import train_test_split
>>> from sklearn.metrics import accuracy_score
>>> iris = datasets.load_iris()
>>> X, y = iris.data[:, :2], iris.target
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=33)
>>> scaler = preprocessing.StandardScaler().fit(X_train)
>>> X_train = scaler.transform(X_train)
>>> X_test = scaler.transform(X_test)
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
>>> knn.fit(X_train, y_train)
>>> y_pred = knn.predict(X_test)
>>> accuracy_score(y_test, y_pred)
```

Loading The Data

Also see NumPy & Pandas

Your data needs to be numeric and stored as NumPy arrays or SciPy sparse matrices. Other types that are convertible to numeric arrays, such as Pandas DataFrame, are also acceptable.

```
>>> import numpy as np
>>> X = np.random.random((10,5))
>>> y = np.array(['M','M','F','F','M','F','M','M','F','F','F'])
>>> X[X < 0.7] = 0
```

Training And Test Data

```
>>> from sklearn.cross_validation import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(X,
                                                        y,
                                                        random_state=0)
```

Create Your Model

Supervised Learning Estimators

Linear Regression

```
>>> from sklearn.linear_model import LinearRegression
>>> lr = LinearRegression(normalize=True)
```

Support Vector Machines (SVM)

```
>>> from sklearn.svm import SVC
>>> svc = SVC(kernel='linear')
```

Naive Bayes

```
>>> from sklearn.naive_bayes import GaussianNB
>>> gnb = GaussianNB()
```

KNN

```
>>> from sklearn import neighbors
>>> knn = neighbors.KNeighborsClassifier(n_neighbors=5)
```

Unsupervised Learning Estimators

Principal Component Analysis (PCA)

```
>>> from sklearn.decomposition import PCA
>>> pca = PCA(n_components=0.95)
```

K Means

```
>>> from sklearn.cluster import KMeans
>>> k_means = KMeans(n_clusters=3, random_state=0)
```

Model Fitting

Supervised learning	
`>>> lr.fit(X, y)`	Fit the model to the data
`>>> knn.fit(X_train, y_train)`	
`>>> svc.fit(X_train, y_train)`	
Unsupervised Learning	
`>>> k_means.fit(X_train)`	Fit the model to the data
`>>> pca_model = pca.fit_transform(X_train)`	Fit to data, then transform it

Prediction

Supervised Estimators	
`>>> y_pred = svc.predict(np.random.random((2,5)))`	Predict labels
`>>> y_pred = lr.predict(X_test)`	Predict labels
`>>> y_pred = knn.predict_proba(X_test)`	Estimate probability of a label
Unsupervised Estimators	
`>>> y_pred = k_means.predict(X_test)`	Predict labels in clustering algos

Evaluate Your Model's Performance

Classification Metrics

Accuracy Score — Estimator score method

```
>>> knn.score(X_test, y_test)
```

Metric scoring functions

```
>>> from sklearn.metrics import accuracy_score
>>> accuracy_score(y_test, y_pred)
```

Classification Report — Precision, recall, f1-score and support

```
>>> from sklearn.metrics import classification_report
>>> print(classification_report(y_test, y_pred))
```

Confusion Matrix

```
>>> from sklearn.metrics import confusion_matrix
>>> print(confusion_matrix(y_test, y_pred))
```

Regression Metrics

Mean Absolute Error

```
>>> from sklearn.metrics import mean_absolute_error
>>> y_true = [3, -0.5, 2]
>>> mean_absolute_error(y_true, y_pred)
```

Mean Squared Error

```
>>> from sklearn.metrics import mean_squared_error
>>> mean_squared_error(y_test, y_pred)
```

R^2 Score

```
>>> from sklearn.metrics import r2_score
>>> r2_score(y_true, y_pred)
```

Clustering Metrics

Adjusted Rand Index

```
>>> from sklearn.metrics import adjusted_rand_score
>>> adjusted_rand_score(y_true, y_pred)
```

Homogeneity

```
>>> from sklearn.metrics import homogeneity_score
>>> homogeneity_score(y_true, y_pred)
```

V-measure

```
>>> from sklearn.metrics import v_measure_score
>>> metrics.v_measure_score(y_true, y_pred)
```

Cross-Validation

```
>>> from sklearn.cross_validation import cross_val_score
>>> print(cross_val_score(knn, X_train, y_train, cv=4))
>>> print(cross_val_score(lr, X, y, cv=2))
```

Tune Your Model

Grid Search

```
>>> from sklearn.grid_search import GridSearchCV
>>> params = {"n_neighbors": np.arange(1,3),
              "metric": ["euclidean", "cityblock"]}
>>> grid = GridSearchCV(estimator=knn,
                        param_grid=params)
>>> grid.fit(X_train, y_train)
>>> print(grid.best_score_)
>>> print(grid.best_estimator_.n_neighbors)
```

Randomized Parameter Optimization

```
>>> from sklearn.grid_search import RandomizedSearchCV
>>> params = {"n_neighbors": range(1,5),
              "weights": ["uniform", "distance"]}
>>> rsearch = RandomizedSearchCV(estimator=knn,
                                 param_distributions=params,
                                 cv=4,
                                 n_iter=8,
                                 random_state=5)
>>> rsearch.fit(X_train, y_train)
>>> print(rsearch.best_score_)
```

Preprocessing The Data

Standardization

```
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler().fit(X_train)
>>> standardized_X = scaler.transform(X_train)
>>> standardized_X_test = scaler.transform(X_test)
```

Normalization

```
>>> from sklearn.preprocessing import Normalizer
>>> scaler = Normalizer().fit(X_train)
>>> normalized_X = scaler.transform(X_train)
>>> normalized_X_test = scaler.transform(X_test)
```

Binarization

```
>>> from sklearn.preprocessing import Binarizer
>>> binarizer = Binarizer(threshold=0.0).fit(X)
>>> binary_X = binarizer.transform(X)
```

Encoding Categorical Features

```
>>> from sklearn.preprocessing import LabelEncoder
>>> enc = LabelEncoder()
>>> y = enc.fit_transform(y)
```

Imputing Missing Values

```
>>> from sklearn.preprocessing import Imputer
>>> imp = Imputer(missing_values=0, strategy='mean', axis=0)
>>> imp.fit_transform(X_train)
```

Generating Polynomial Features

```
>>> from sklearn.preprocessing import PolynomialFeatures
>>> poly = PolynomialFeatures(5)
>>> poly.fit_transform(X)
```

References

Main References: These are references to deepen your understanding of material presented in lecture. The list is by no means exhaustive.

- Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, *An Introduction to Statistical Learning*, Springer 2013
- Trevor Hastie, Robert Tibshirani, Jerome Friedman, *Elements of Statistical Learning*, Springer 2013
- Christopher M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006.
- Cameron Davidson-Pilon, *Bayesian Methods for Hackers*, https://github.com/CamDavidsonPilon/Probabilistic-Programming-and-Bayesian-Methods-for-Hackers

Next Class

- Lecture 2: Introduction to Supervised Learning
  - Homework discussion, set up Github. Review of Linear Algebra and Probability.
  - Linear Regression and Derivation of Analytical Solution when p=2.
  - Model Training and Testing.
  - Gradient Descent, and Introduction to Convex Optimization.
  - Concrete Examples of Linear Regression and Gradient Descent in Python.