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 - Conde Nast 2016 - 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. Examples 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/).
    - Anything you can think of which can use data to make predictions/suggestions.

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. Send me an email at d2991@columbia.edu with your Github account and I'll add you to the repo: https://github.com/Columbia-Intro-Data-Science/APMAE4990-

What is Data Science?

Introduction and Examples

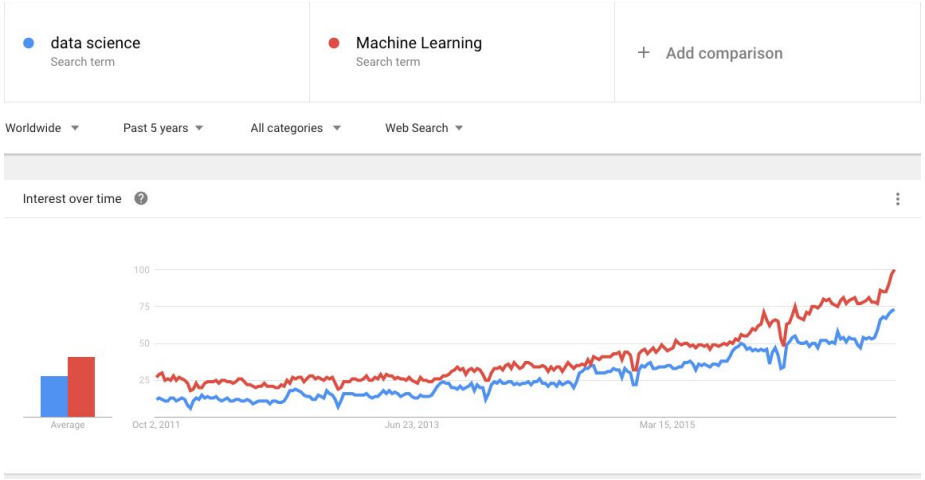
Outline

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

What is Data Science?

- **Predictive (Supervised Learning):** The science of using data to predict an outcome (clicking, subscription, cancerous cells, price of a stock)
- **Descriptive (Unsupervised Learning):** Using data to group items/users into categories (ie. extract topics/categories from articles)
- **Prescriptive (Reinforcement Learning):** Optimizing action based on response variable (ie. who should receive a marketing email, based on sign ups from an experiment)
- **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 is Blowing Up

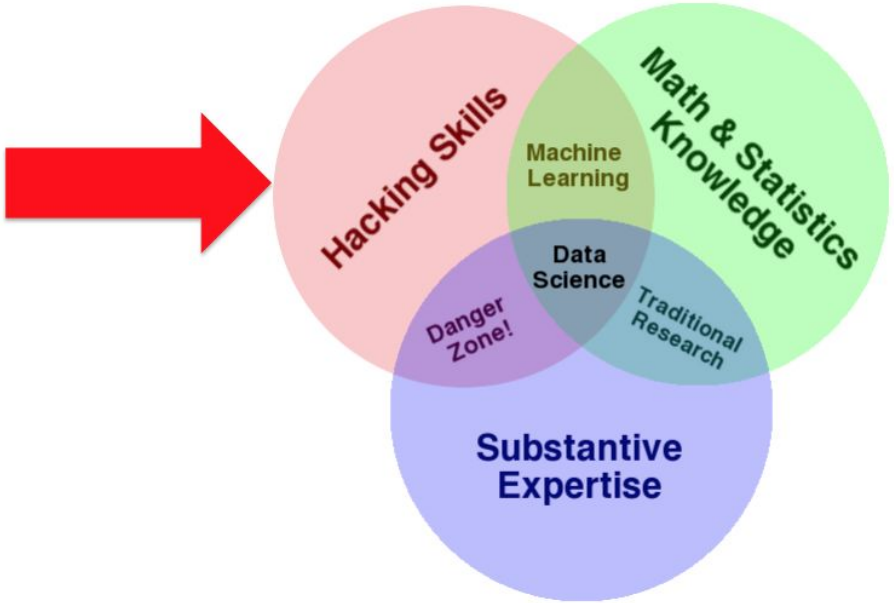


- Intellectually rich landscape of problems in a relatively new field.
- Can save a company millions of dollars by implementing the right algorithm effectively, allowing us to have significant impact.

“"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/

What is Data Science?

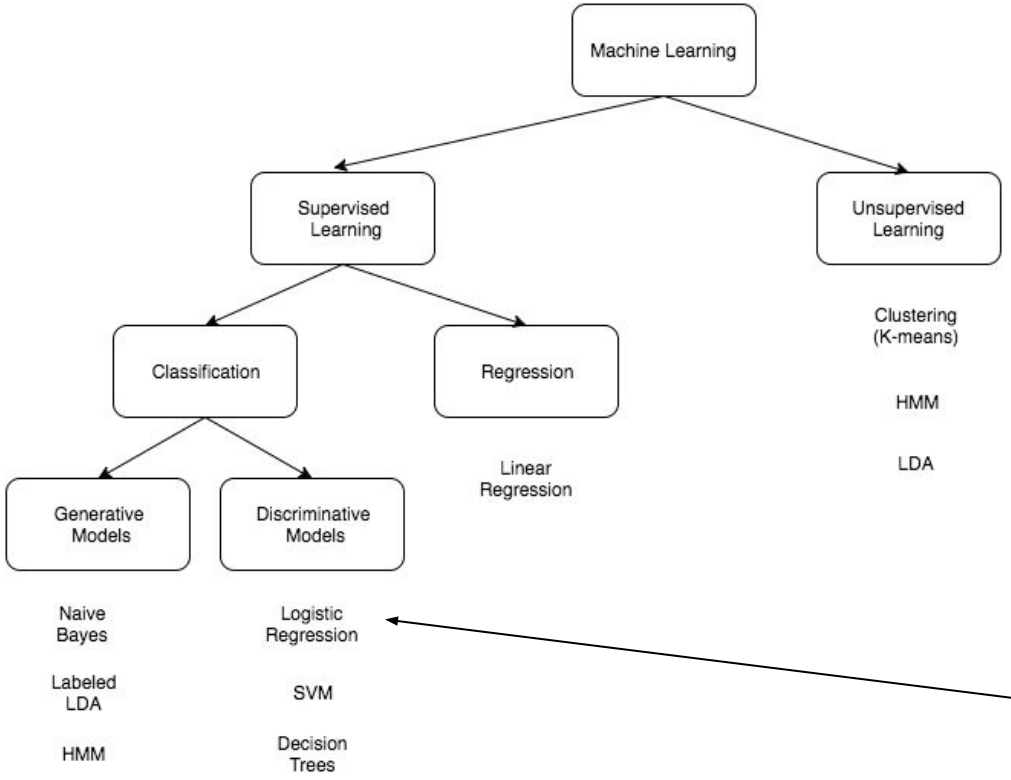


Math/Statistical Knowledge: Need understanding probability, statistics, optimization methods to create and use models.

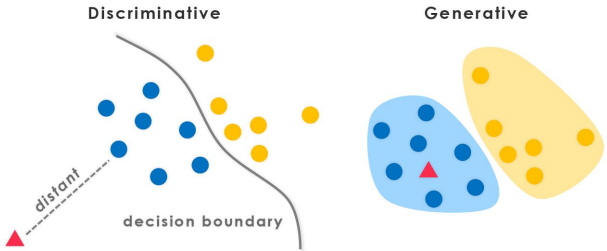
Hacking Skills: Comfort with Linux/Unix, networks, databases, working from the command line, debugging code.

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).



Predictive Learning

(Supervised)

Predictive Learning - Summary

- Predictive 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).

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

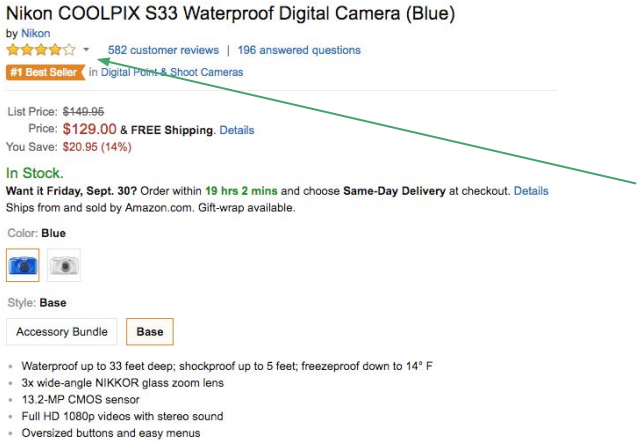
- **Algorithms (you will learn):**
  - Linear Regression (Regression)/Logistic Regression(Classification).
  - Random Forest/Decision Tree Regression/Classification.
  - SVM (Support Vector Machines).
  - Maximum Likelihood and Time Series Modeling (more advanced - fit data to a prior probability distribution).
  - Neural Nets (if time permits)

Predictive Learning - Examples

(Supervised)

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 = $$.**

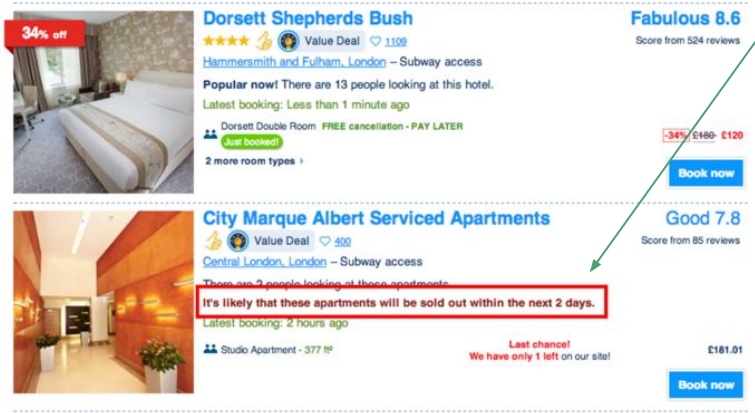
1. Generate the model, evaluate.
2. Run A/B test to measure performance or utility.
3. Learn from the model and improve.

How do we use our model? A/B Testing

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

Booking.com hotel bookings

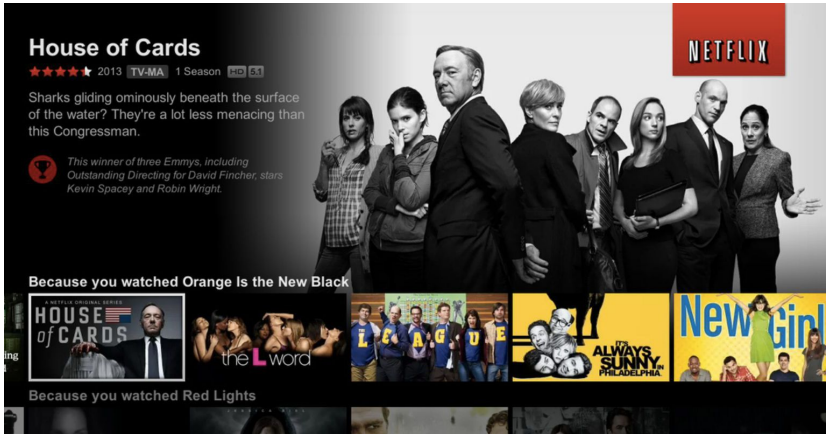
Can we predict that a hotel is likely to sell out soon? If so when? Why do we care?



- **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).

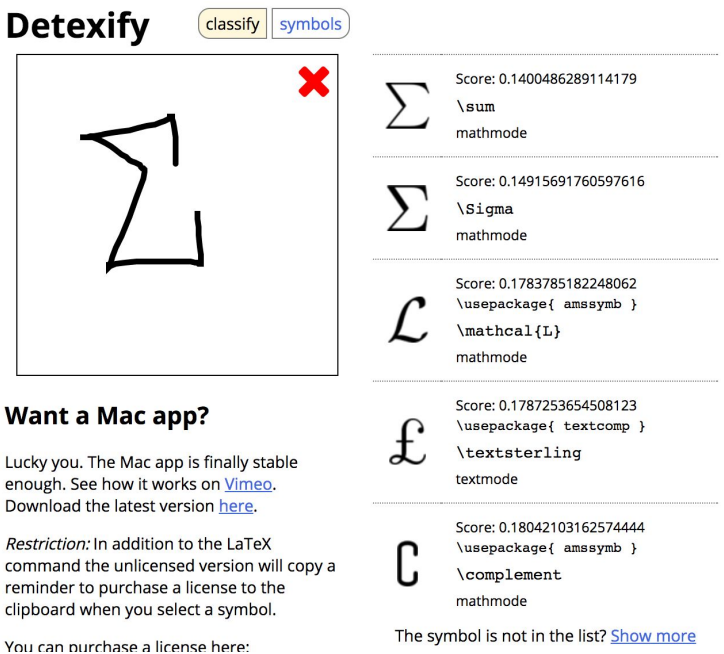
Netflix.com movie ratings

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 $$.

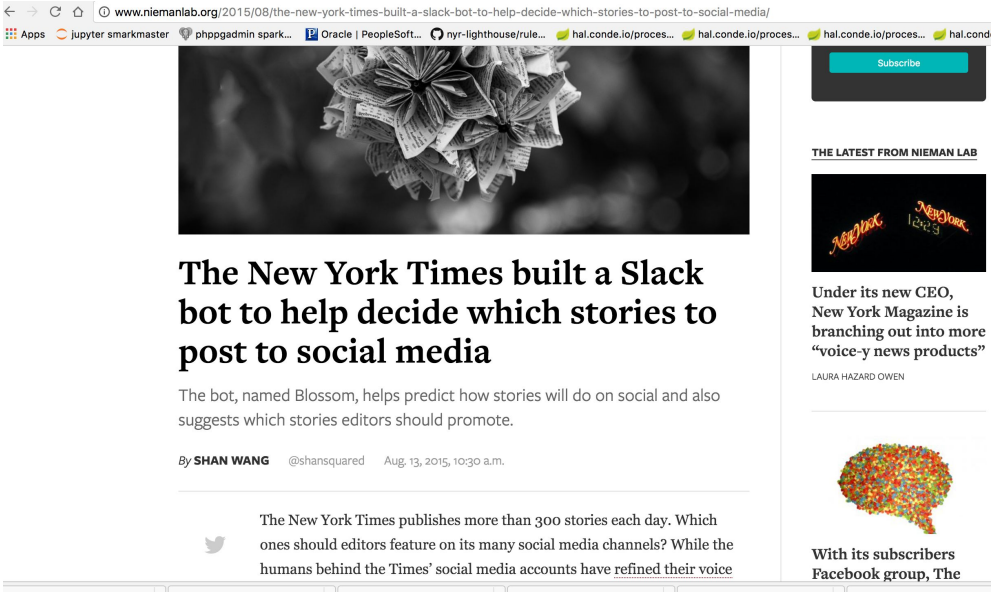
Handwritten latex symbol recognition



- This interactive handwritten latex reader was trained by people actively ‘teaching’ it.
- It started off with some basis training set, then update it’s scores based on what users click.
- Continues to actively learn by example.

Source: http://detexify.kirelabs.org/classify.html

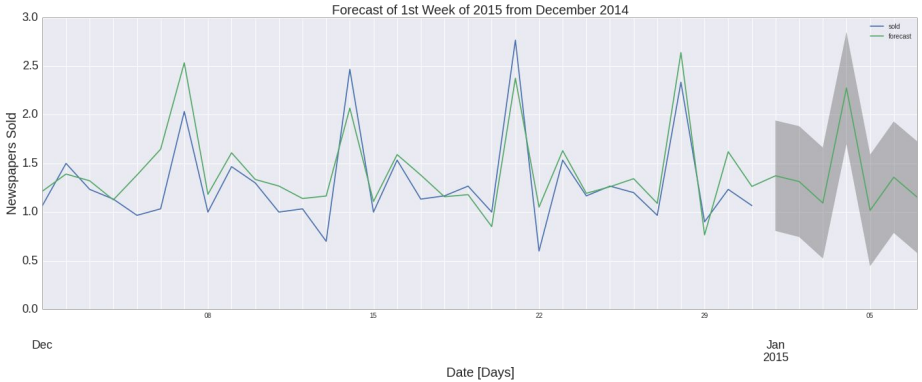
New York Times predicting viral content



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

My colleague **Colin Russel**, who created Blossom (which received press exposure for his unique algorithm) will give a guest lecture and explain how he did it

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

Predictive Learning - Methods

(Supervised)

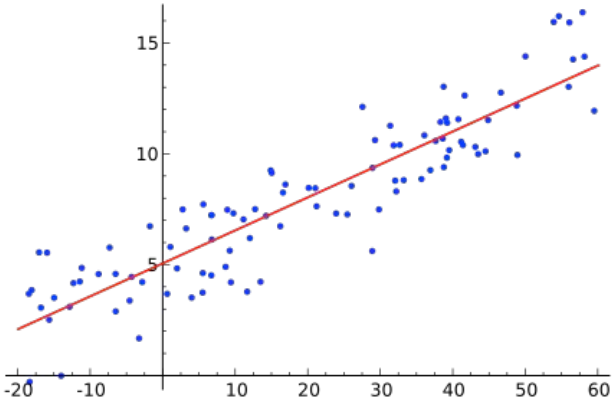
Predictive Learning - Linear Regression

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

Can we find a function $f : X \to Y$ minimizing the distance to the data.

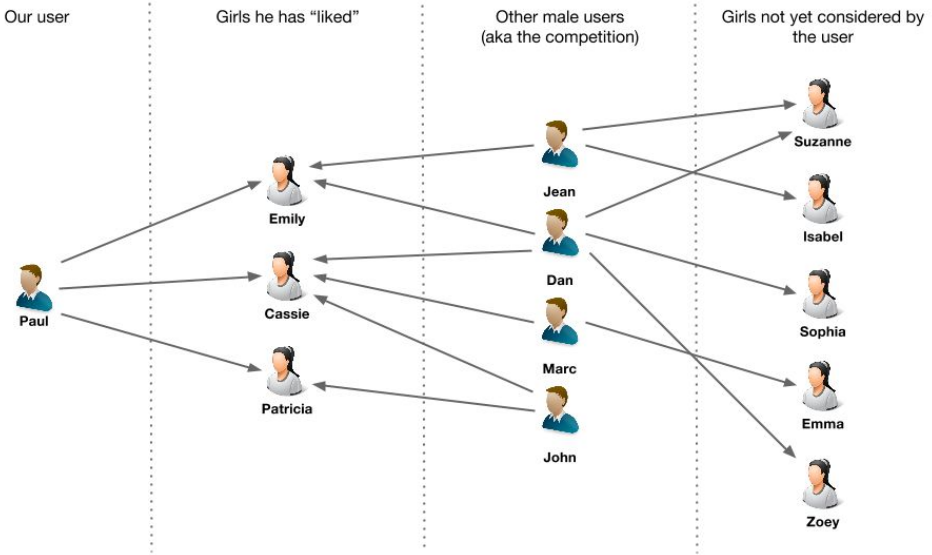
$$\frac{1}{N}\sum_{i=1}^{N} |y_i - f(x_i)|^p$$

Linear: $f(x_i) = \beta \cdot x_i$



All of predictive machine learning is based on discovering ways to find f (although the norms we use will depend on the problem at hand, it won't always be this one).

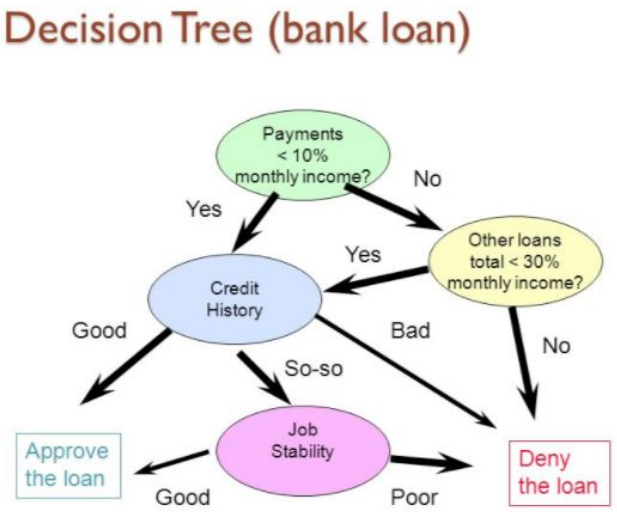
Predictive Learning - Recommendation Engines



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

- Guy j for Girl n has a propensity measured by a bipartite graph diffusion model.

Predictive Learning - Decision Trees



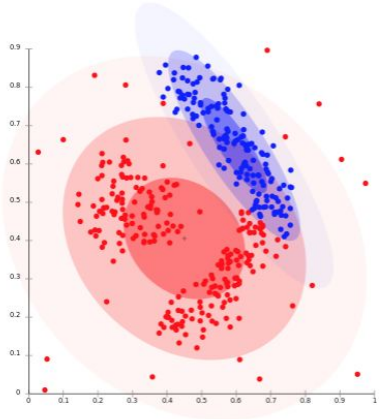
- Should this person receive a loan?
- A decision tree is another way of finding a “rule” which assigns user attributes to an outcome.

Descriptive Learning

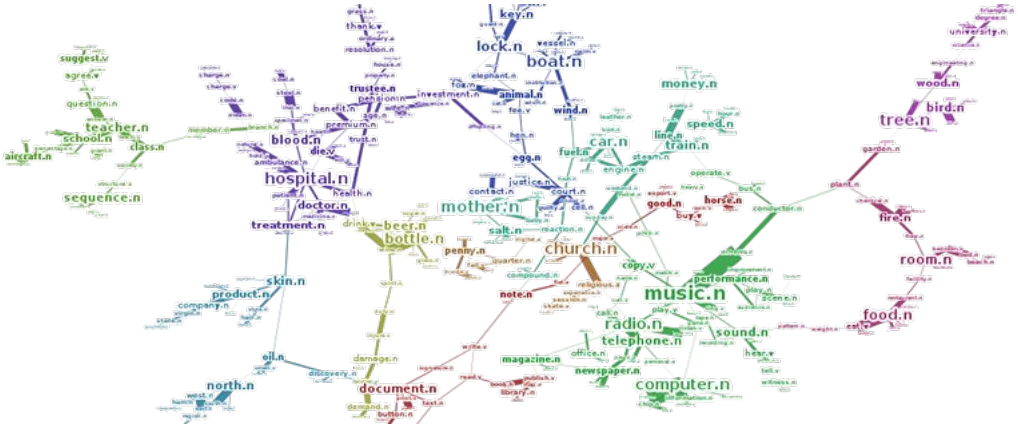
(Unsupervised)

Descriptive 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.
  - SVM
  - Topic Models (LDA, etc)



Topic Models

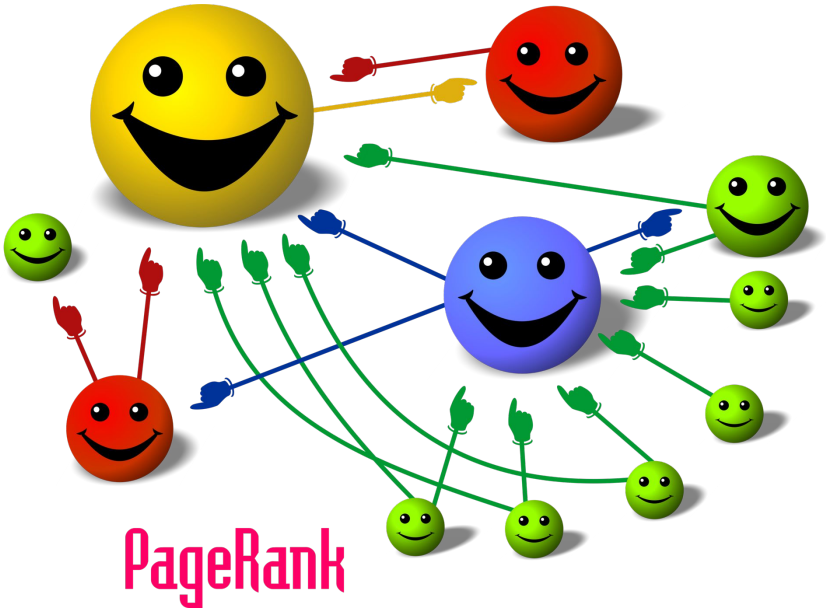


- Topic models attempt to cluster content into a finite collection of ‘topics’.
- The model creates topic categories by clustering words commonly occurring together into groups (roughly speaking).
- Reading/Writing behavior, while unsupervised, has tremendous predictive power for many algorithms in practice.

Readerscope at The New York Times

- Can we identify topics in various countries and regions around the world? (uses topic model)
- Can use this to improve engagement worldwide.
- Can we use this cluster users and direct advertisers to these specific subscribers?

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)

Prescriptive Learning

Reinforcement Learning

Prescriptive 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?).

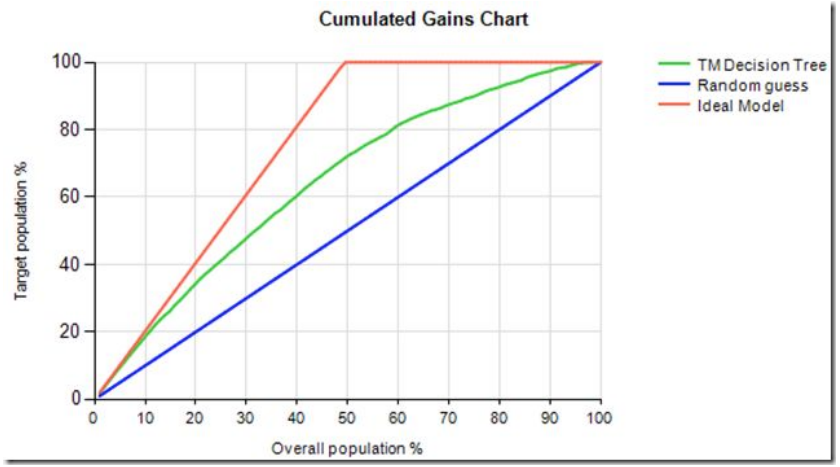
- A well defined metric is used to determine the performance of such a model (will be seen later).

- **Algorithms:**
  - Relies entirely on maximizing expectation of reward conditioned on user attributes and action.
  - Incredibly useful since it's actionable.
  - Can be "live" (multiarmed bandit) or from logged data (uplift modeling).
  - Easiest when we have an A/B (randomized controlled trial) where we can measure causal inference of an outcome conditioned on an action.

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

Uplift Modeling - 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?

- Basic foundation of causal inference.

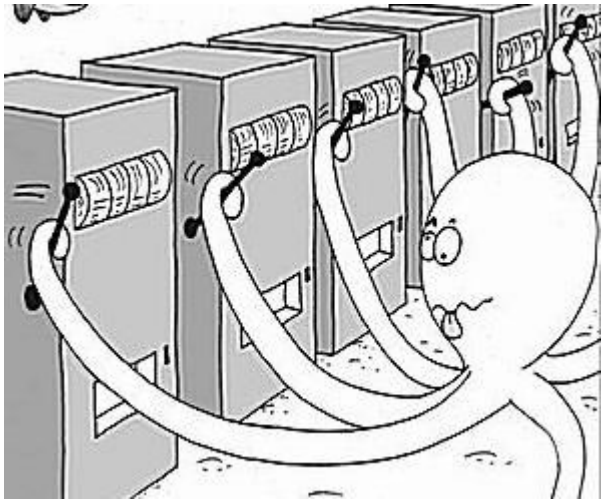
$$\max_{a \in \Omega} \mathbb{E}(Y|X, a)$$

(this will be explained in later lectures)

$y -$ outcome
$x -$ item/user attribute
$a -$ action taken
$\Omega -$ distribution of rewards

Example: Multiarmed 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.

What will you learn in this course?

What will you learn in this course?

- 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.
  - 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).
- **Methods of Machine Learning:**
  - Regression, classification, decision trees, RF and clustering.
  - Model complexity and regularization (variance/bias tradeoff). Cross validation.
  - Graph Diffusion, collaborative filtering, random walks. Graphical models.
  - Bayesian statistics. 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 any of the tools (or others!) covered in this class.

What should you know?

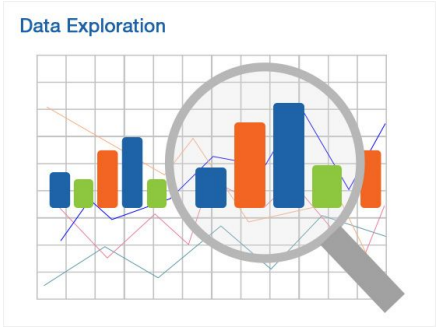
- Undergraduate math - linear algebra, basic probability, calculus.
- Basic unix commands are good to know.
- Background in programming (ideally Python).

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

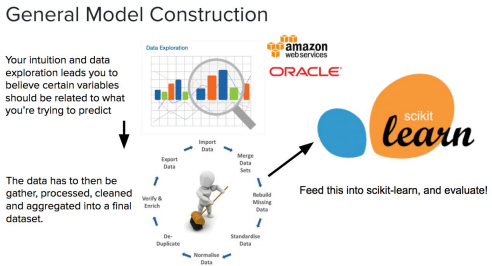
ORACLE

scikit learn

Feed this into scikit-learn, and evaluate!

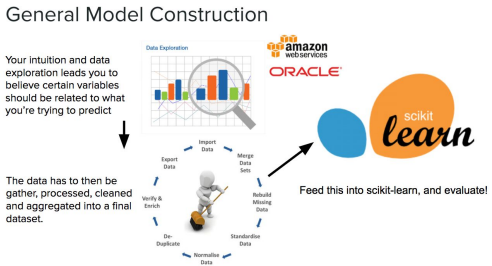
Example 1: MINST dataset

- 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?

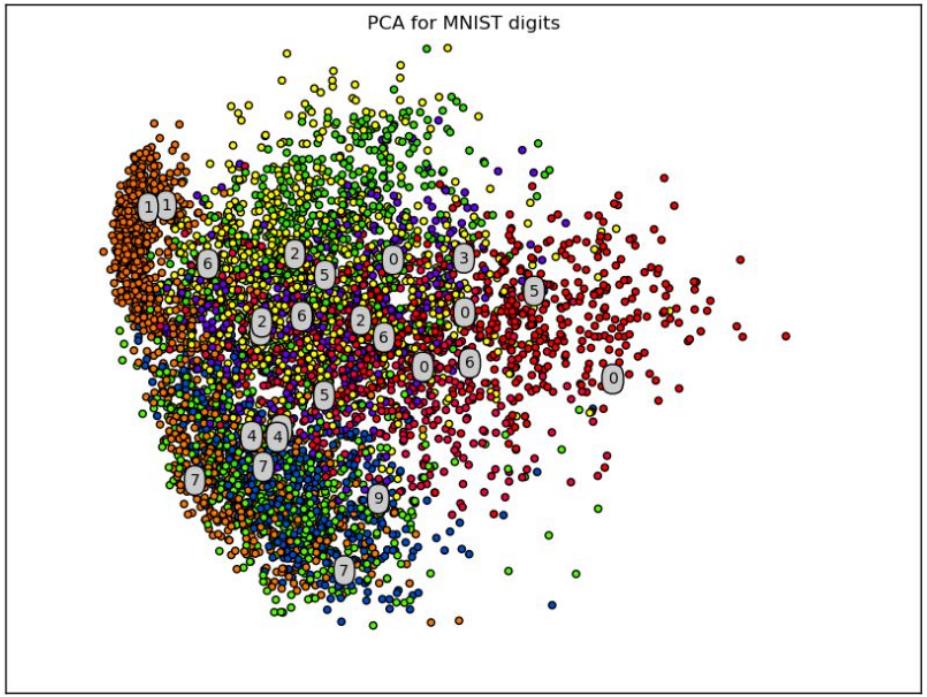


Example 1: MINST dataset

- 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.

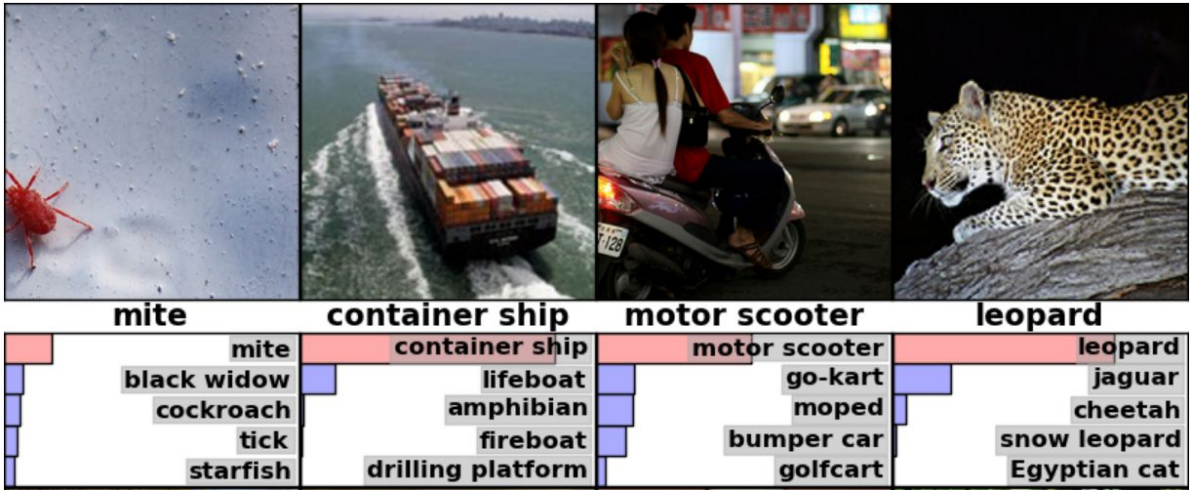


Visualization of the top two components



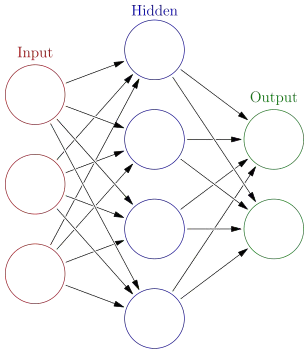
- Here we see a clustering of the most ‘significant’ components extracted from the 64 pixels/features (*will be explained later via PCA*).
- Take a fixed circle around each point, look at the K nearest neighbors, and take the majority vote.

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)

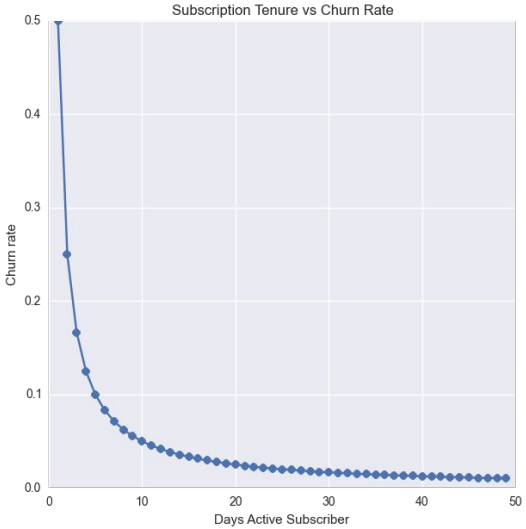


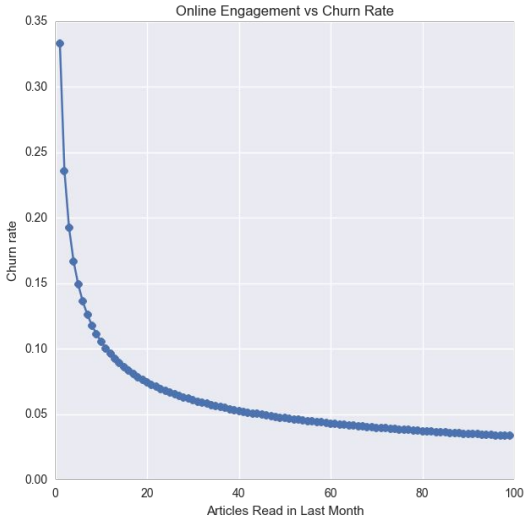
The New York Times

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

The New York Times

Example 2: Predicting user churn





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

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

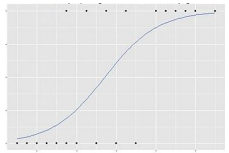
- **Step 1:**
  - Explore Data.
  - Find relationships, develop features
  - Investigate relationships between
  - data sources.

- **Step 2:**
  - Merge datasources, 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

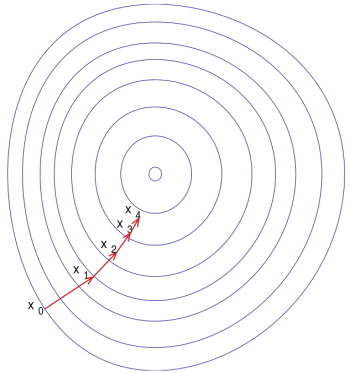
How do we learn the parameter ?

The logistic regression model:

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

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

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

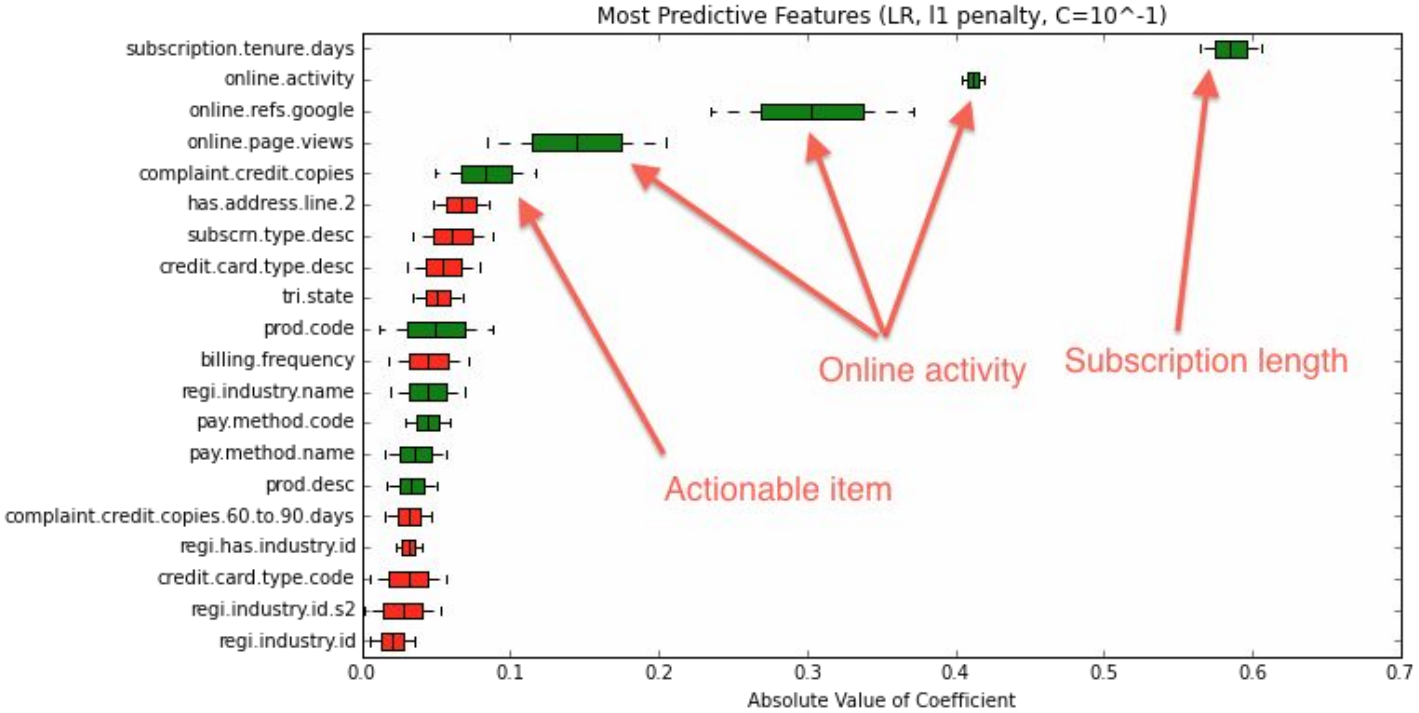


We minimize this convex cost function via gradient descent.

- 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.

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

The New York Times



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

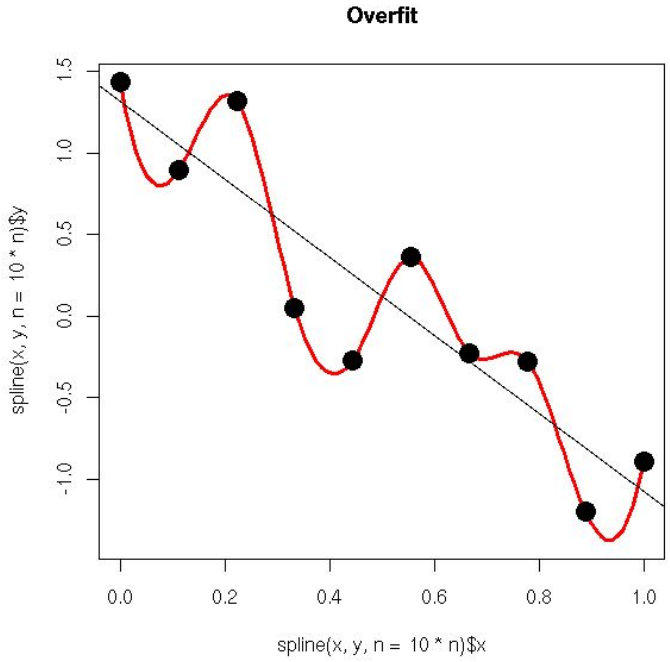
Classification:
(examples)

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

accuracy

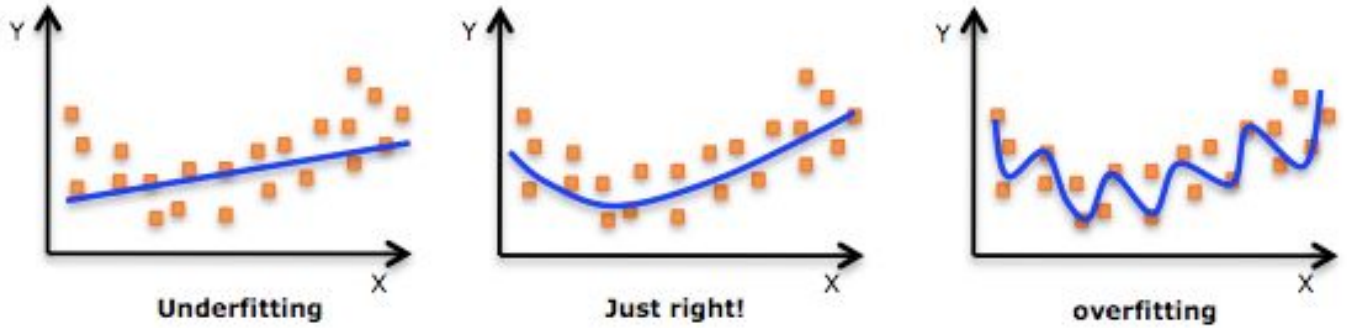
Predicted Value (predicted by the test)		Actual Value (as confirmed by experiment): positives	Actual Value (as confirmed by experiment): negatives
	positives	**TP** True Positive	**FP** False Positive
	negatives	**FN** False Negative	**TN** True Negative

- What’s wrong with this picture?

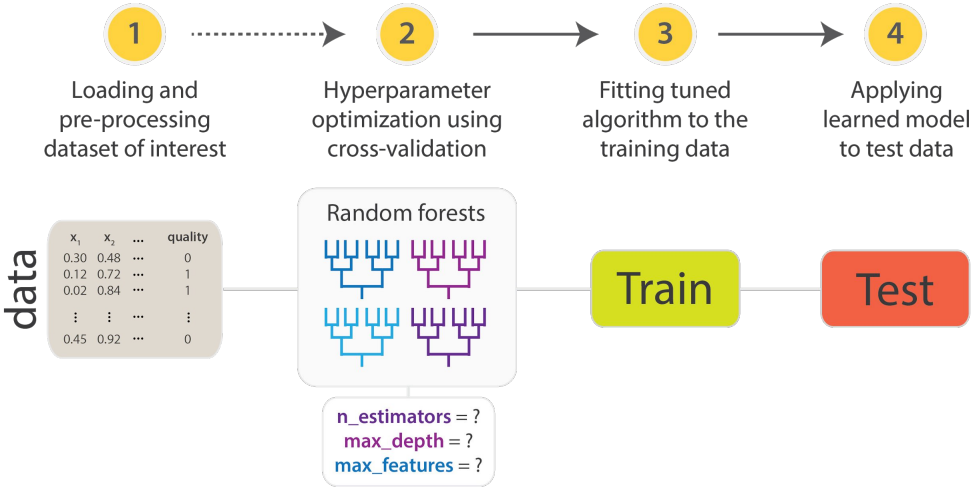


- The model seems to fit the data perfectly! Looks like we found a great model.
- What is wrong, and can you explain how we fix it?

Choosing the right model complexity



How do we evaluate performance properly?

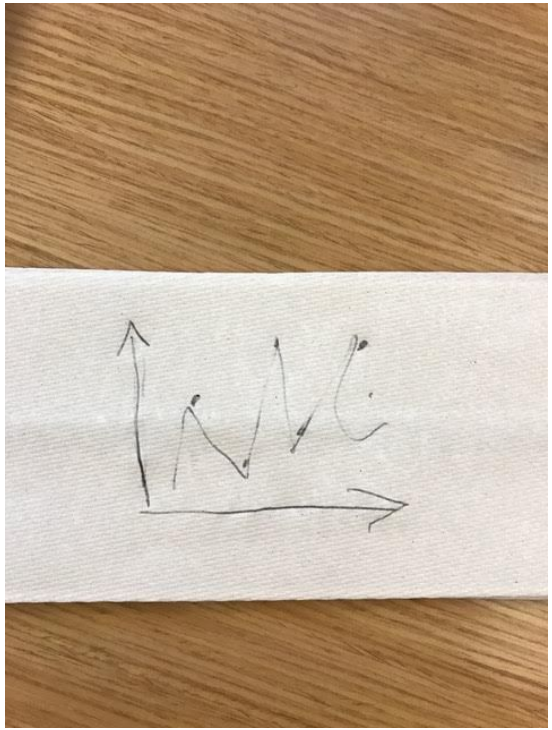


- We we train the model on **different data than we evaluate it.**
- The training data and testing data are sampled from similar distributions.
- We optimize the complexity of the model to prevent overfitting.

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

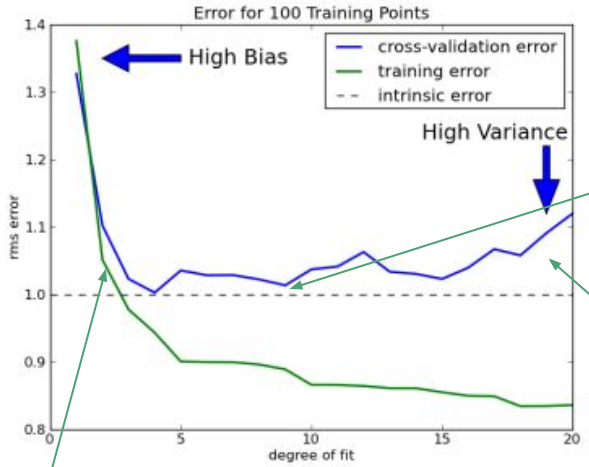
Parameter Optimization: Optimizing the performance of your model on the test data.

A napkin I kept from Chris Wiggins

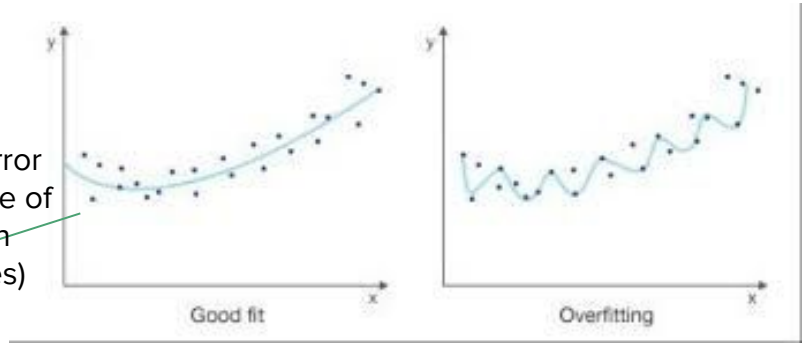


- When I argued regularization wasn't necessary since I had used cross validation to Chris Wiggins, he made me this note.

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 Linear Regression
  - Homework discussion, set up Github. Review of Linear Algebra and Probability.
  - Definition, Derivation and comparison of Lp norms. How does it affect the model?
  - 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 in an iPython Notebook.

Appendix

Examples [edit]

Gender classification [edit]

Problem: classify whether a given person is a male or a female based on the measured features. The features include height, weight, and foot size.

Training [edit]

Example training set below.

Gender	height (feet)	weight (lbs)	foot size(inches)
male	6	180	12
male	5.92 (5'11")	190	11
male	5.58 (5'7")	170	12
male	5.92 (5'11")	165	10
female	5	100	6
female	5.5 (5'6")	150	8
female	5.42 (5'5")	130	7
female	5.75 (5'9")	150	9

The classifier created from the training set using a Gaussian distribution assumption would be (given variances are *unbiased* sample variances):

Gender	mean (height)	variance (height)	mean (weight)	variance (weight)	mean (foot size)	variance (foot size)
male	5.855	3.5033e-02	176.25	1.2292e+02	11.25	9.1667e-01
female	5.4175	9.7225e-02	132.5	5.5833e+02	7.5	1.6667e+00

Let's say we have equiprobable classes so P(male)= P(female) = 0.5. This prior probability distribution might be based on our knowledge of frequencies in the larger population, or on frequency in the training set.

Testing [edit]

Below is a sample to be classified as a male or female.

Gender	height (feet)	weight (lbs)	foot size(inches)
sample	6	130	8

We wish to determine which posterior is greater, male or female. For the classification as male the posterior is given by

$$\text{posterior (male)} = \frac{P(\text{male})\, p(\text{height} \mid \text{male})\, p(\text{weight} \mid \text{male})\, p(\text{foot size} \mid \text{male})}{evidence}$$

For the classification as female the posterior is given by

$$\text{posterior (female)} = \frac{P(\text{female})\, p(\text{height} \mid \text{female})\, p(\text{weight} \mid \text{female})\, p(\text{foot size} \mid \text{female})}{evidence}$$

The evidence (also termed normalizing constant) may be calculated:

Example Projects