

# Getting Set Up



# Installation Checklist

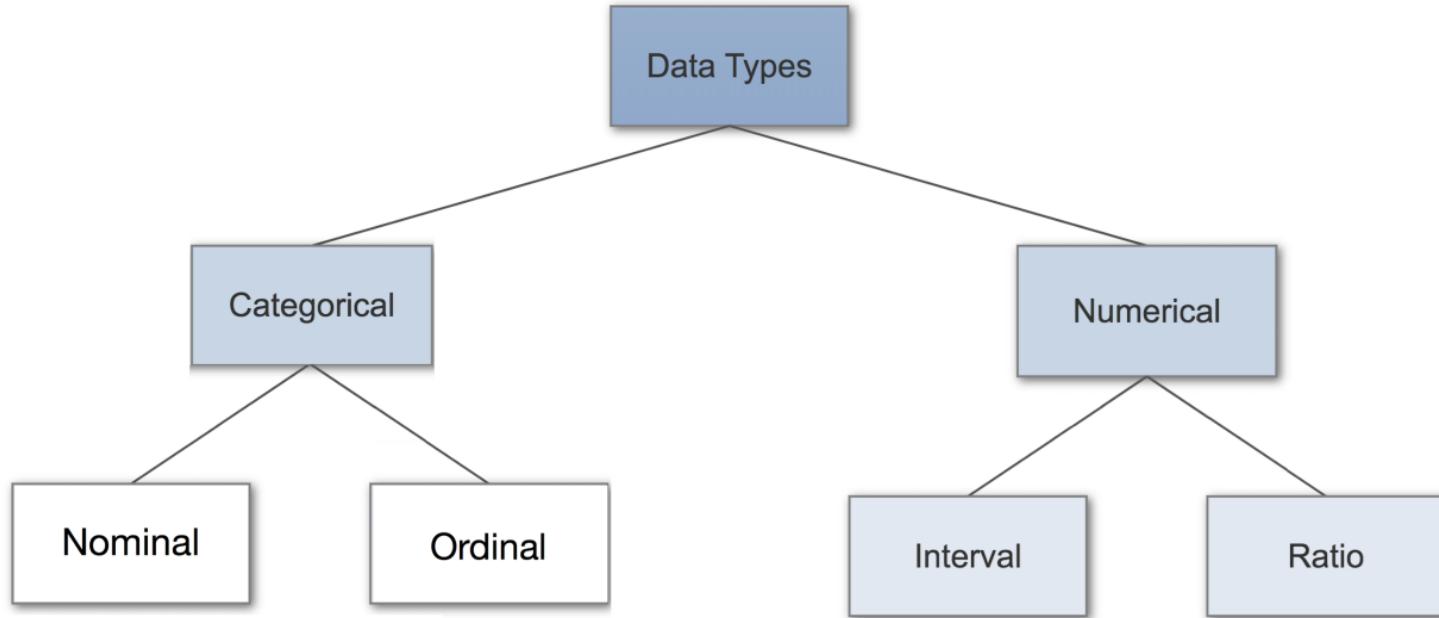
- Install Anaconda
- Open up an Anaconda command prompt
  1. conda install pydotplus
  2. conda install tensorflow-gpu (or tensorflow if not Nvidia)
- Go to the course environment: [ernesto.co](http://ernesto.co)

# Python Basics

Let's just jump right into some code.



# Many Flavors of Data



# Types of Data

## Major Types of Data

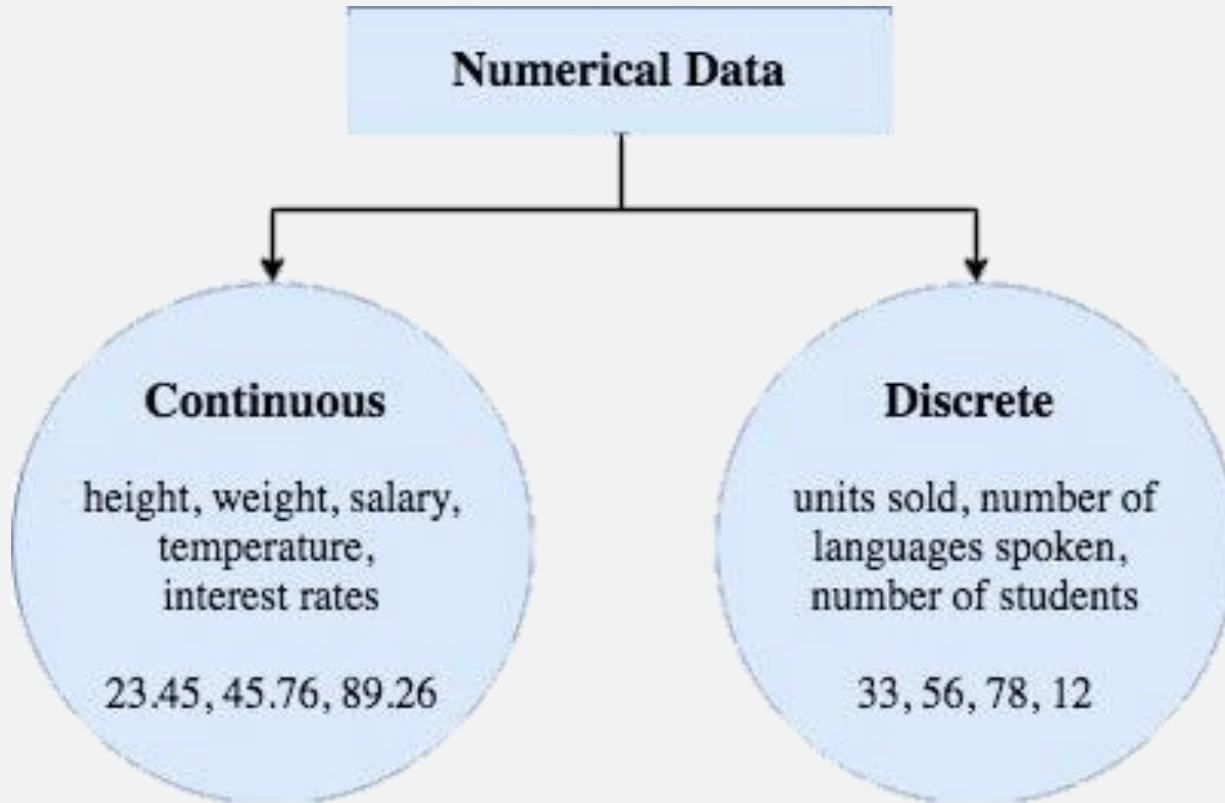
- Numerical
- Categorical
- Ordinal

# Numerical

- Represents some sort of quantitative measurement
  - Heights of people, page load times, stock prices, etc.
- Discrete Data
  - Integer based; often counts of some event.
    - How many purchases did a customer make in a year?
    - How many times did I flip “heads”?
- Continuous Data
  - Has an infinite number of possible values
    - How much time did it take for a user to check out?
    - How much rain fell on a given day?

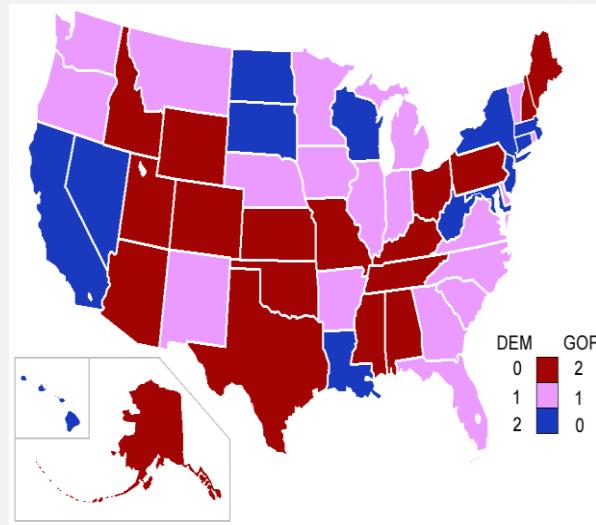


# Numerical



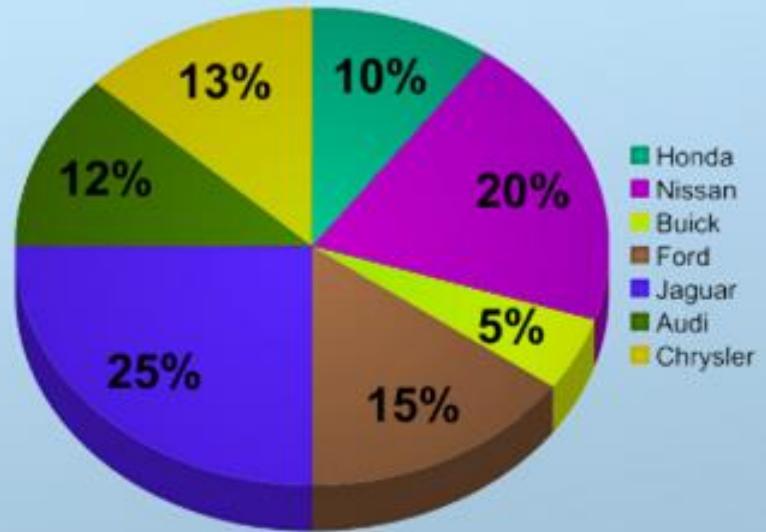
# Categorical

- Qualitative data that has no inherent mathematical meaning
  - Gender, Yes/no (binary data), Race, State of Residence, Product Category, Political Party, etc.
- You can assign numbers to categories in order to represent them more compactly, but the numbers don't have mathematical meaning



# Categorical

## Categorical Data



***when numbers are collected in groups or categories***

# Ordinal

- A mixture of numerical and categorical
- Categorical data that has mathematical meaning
- Example: movie ratings on a 1-5 scale.
  1. Ratings must be 1, 2, 3, 4, or 5
  2. But these values have mathematical meaning; 1 means it's a worse movie than a 2.



# Quiz time!

- Are the following types of data numerical, categorical, or ordinal?
  - How much gas is in your gas tank
  - A rating of your overall health where the choices are 1, 2, 3, or 4, corresponding to “poor”, “moderate”, “good”, and “excellent”
  - The races of your classmates
  - Ages in years
  - Money spent in a store



# Mean, Median, and Mode



# Mean

- AKA Average
- Sum / number of samples
- Example:
  - Number of children in each house on my street:

0, 2, 3, 2, 1, 0, 0, 2, 0

The MEAN is  $(0+2+3+2+1+0+0+2+0) / 9 = 1.11$

# Median

- Sort the values, and take the value at the midpoint.
- Example:

0, 2, 3, 2, 1, 0, 0, 2, 0

Sort it:

0, 0, 0, 0, 2, 2, 2, 3

# Median

- If you have an even number of samples, take the average of the two in the middle.
- Median is less susceptible to outliers than the mean
  - Example: mean household income in the US is \$72,641, but the median is only \$51,939 – because the mean is skewed by a handful of billionaires.
  - Median better represents the “typical” American in this example.

# Mode

- The most common value in a data set
  - Not relevant to continuous numerical data
- Back to our number of kids in each house example:

0, 2, 3, 2, 1, 0, 0, 2, 0

How many of each value are there?

0: 4, 1: 1, 2: 3, 3: 1

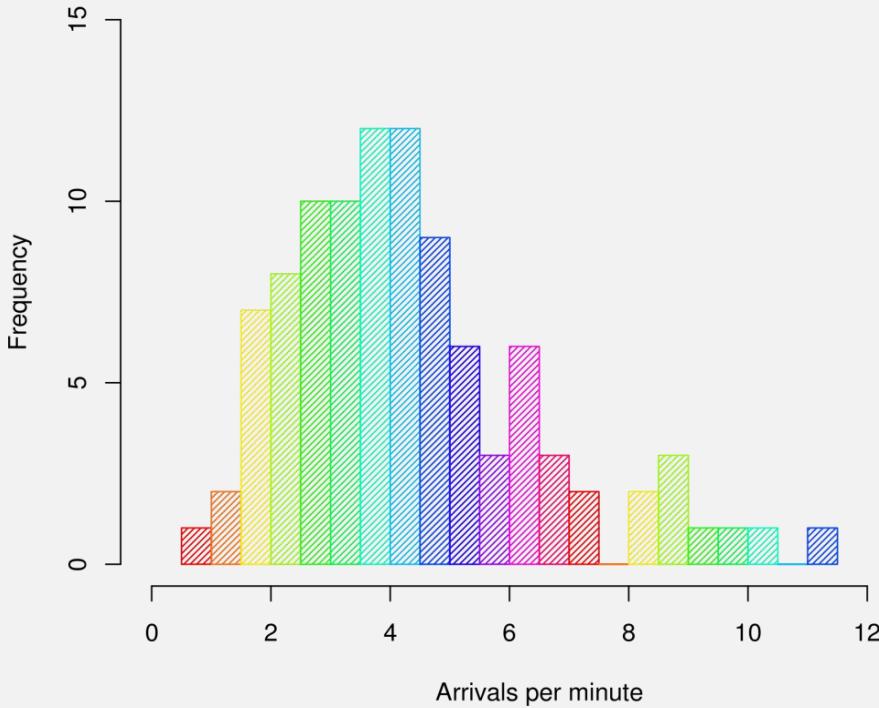
The MODE is 0

# Standard Deviation and Variance



# An example of a histogram

Histogram of arrivals



# Variance measures how “spread-out” the data is.

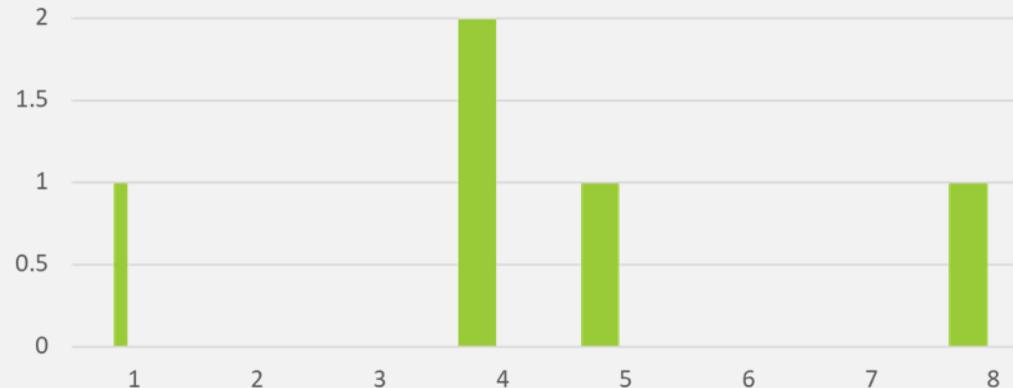
- Variance ( $\sigma^2$ ) is simply the **average of the squared differences from the mean**
- Example: What is the variance of the data set (1, 4, 5, 4, 8)?
  - First find the mean:  $(1+4+5+4+8)/5 = 4.4$
  - Now find the differences from the mean: (-3.4, -0.4, 0.6, -0.4, 3.6)
  - Find the squared differences: (11.56, 0.16, 0.36, 0.16, 12.96)
  - Find the average of the squared differences:
    - $\sigma^2 = (11.56 + 0.16 + 0.36 + 0.16 + 12.96) / 5 = 5.04$

# Standard Deviation $\sigma$ is just the square root of the variance.

$$\sigma^2 = 5.04$$

$$\sigma = \sqrt{5.04} = 2.24$$

So the standard deviation of (1, 4, 5, 4, 8) is 2.24.



*This is usually used as a way to identify outliers. Data points that lie more than one standard deviation from the mean can be considered unusual.*

*You can talk about how extreme a data point is by talking about “how many sigmas” away from the mean it is.*

# Population vs. Sample

- If you're working with a sample of data instead of an entire data set (the *entire population*)...
  - Then you want to use the “sample variance” instead of the “population variance”
  - For N samples, you just divide the squared variances by N-1 instead of N.
  - So, in our example, we computed the population variance like this:
    - $\sigma^2 = (11.56 + 0.16 + 0.36 + 0.16 + 12.96) / 5 = 5.04$
  - But the sample variance would be:
    - $S^2 = (11.56 + 0.16 + 0.36 + 0.16 + 12.96) / 4 = 6.3$

# Fancy Math

- Population variance:

$$\sigma^2 = \frac{\sum (X - \mu)^2}{N}$$

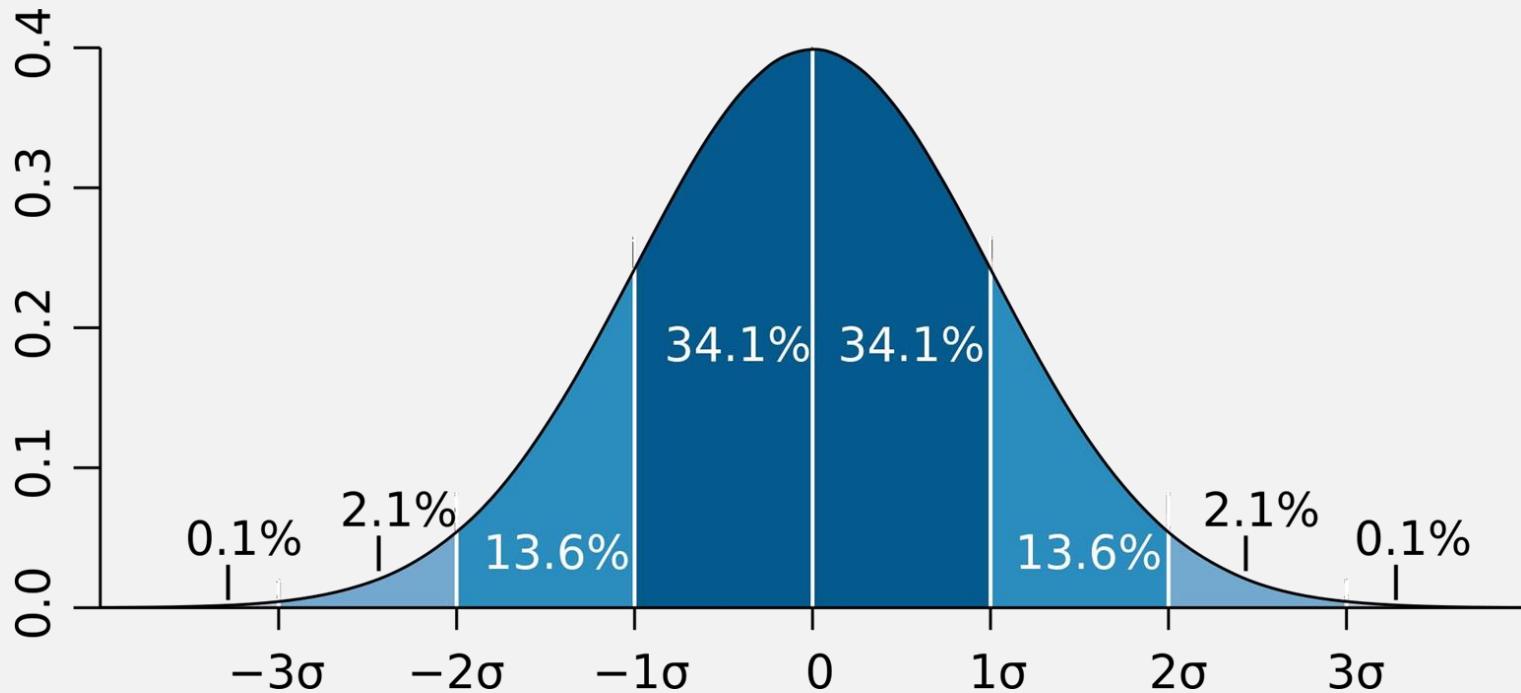
- Sample variance:

$$s^2 = \frac{\sum (X - M)^2}{N-1}$$

# Probability Density Functions

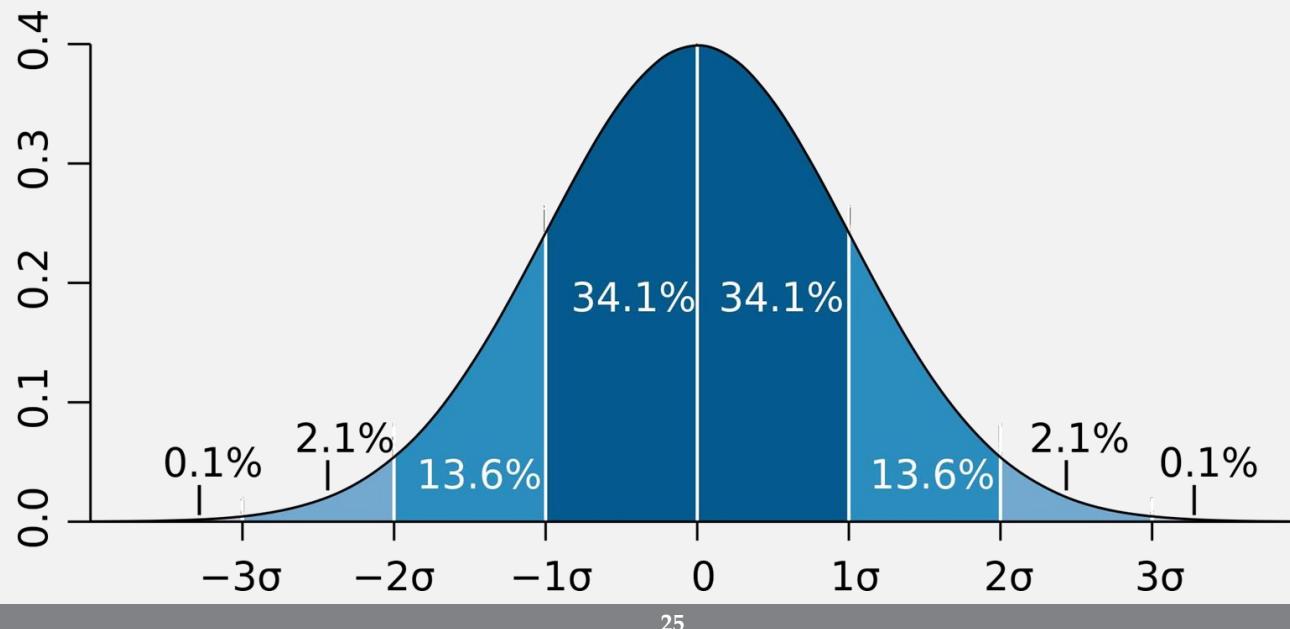


# Example: a “normal distribution”

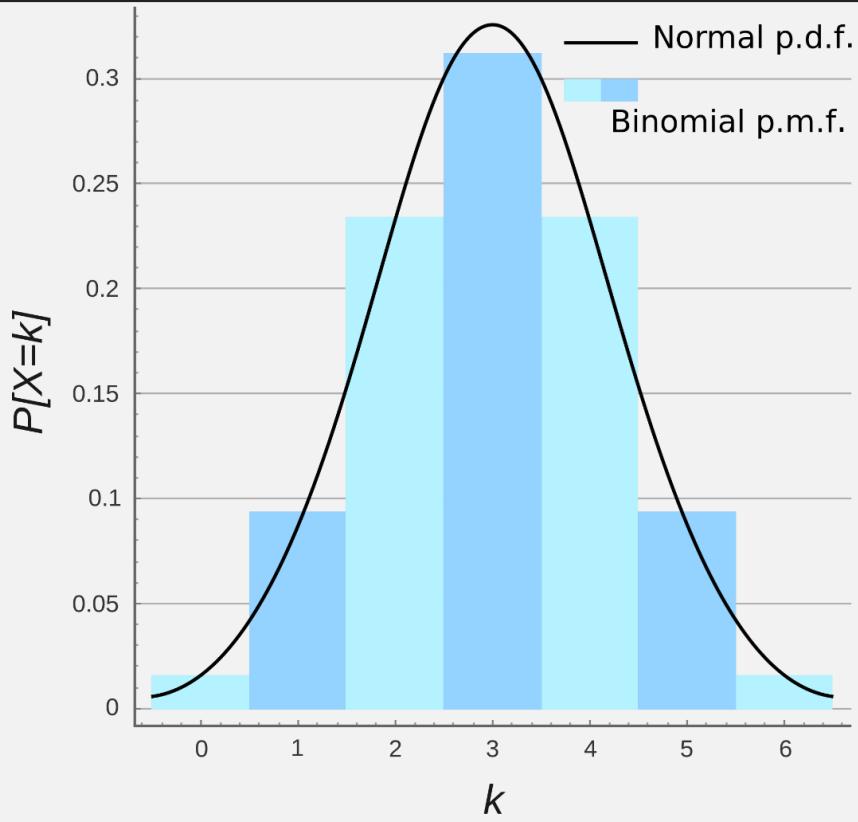


# Example: a “normal distribution”

- Gives you the probability of a data point falling within some given range of a given value.



# Probability Mass Function

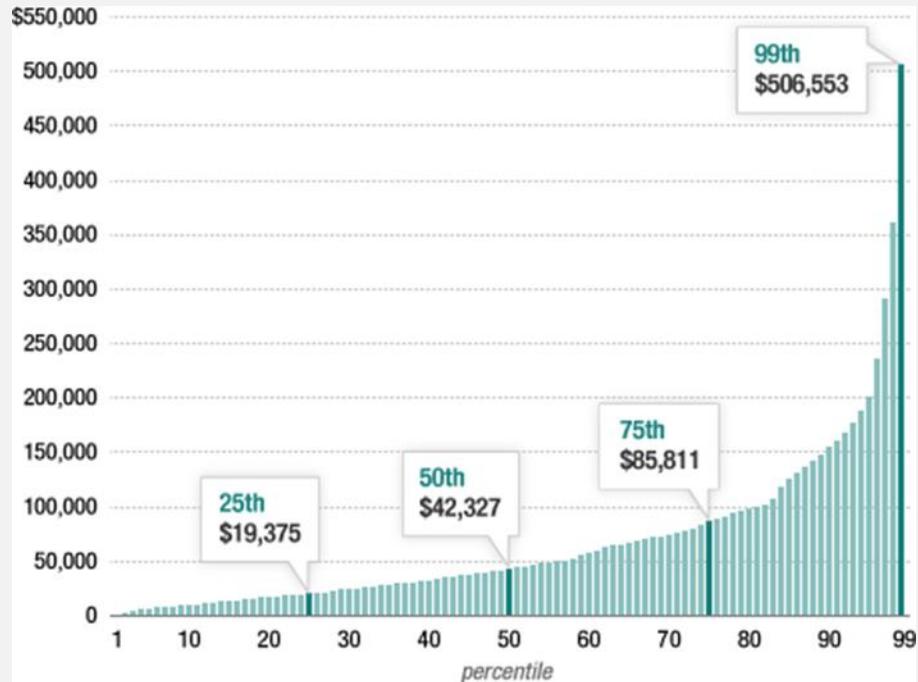


# Percentiles and Moments

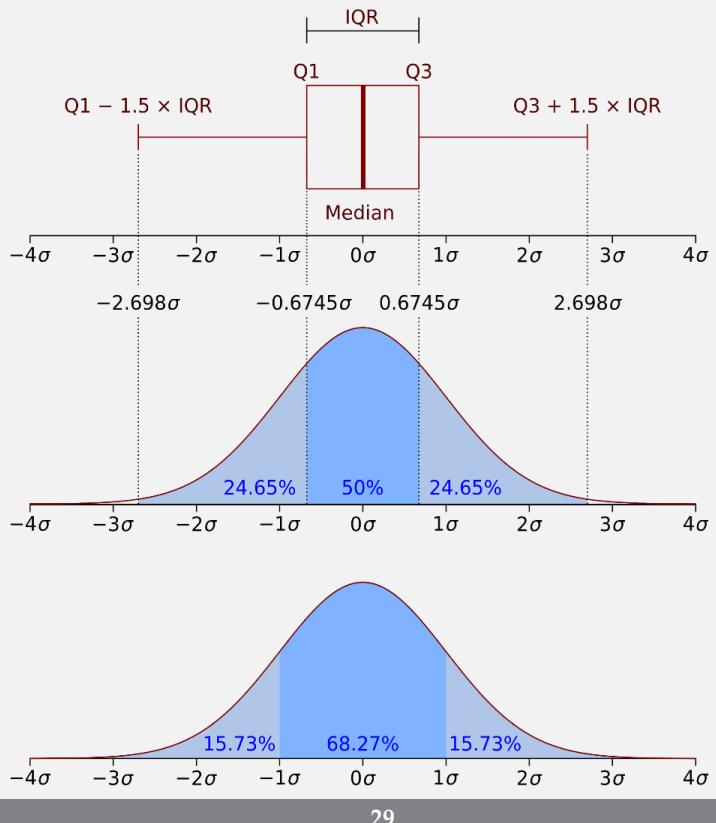


# Percentiles

- In a data set, what's the point at which X% of the values are less than that value?
- Example: income distribution



# Percentiles in a normal distribution

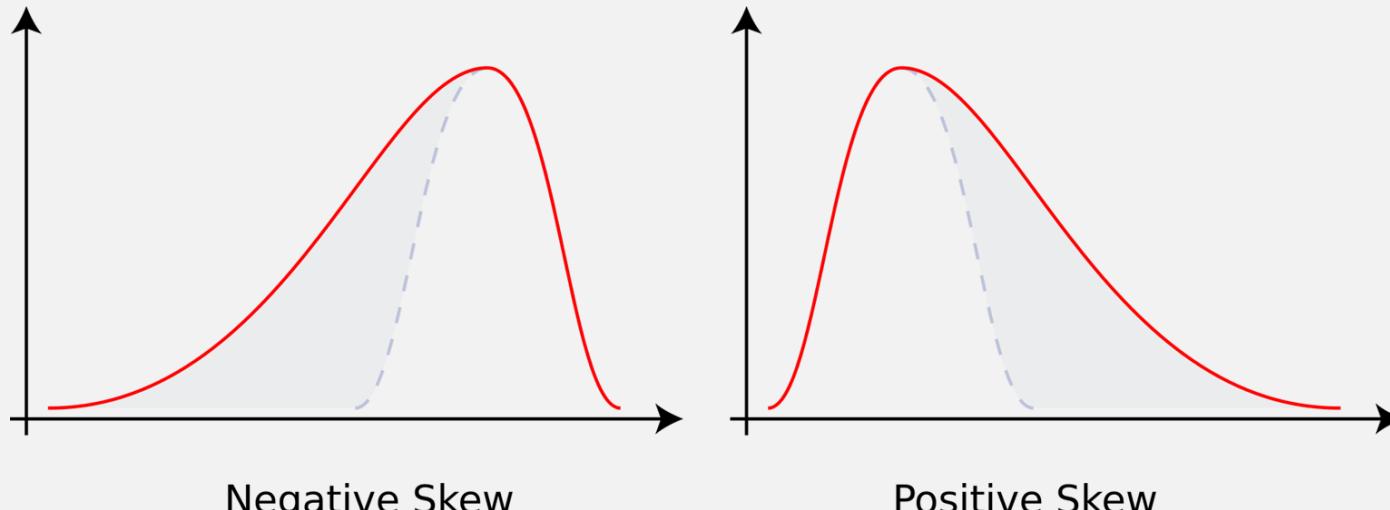


# Moments

- Quantitative measures of the shape of a probability density function
- Mathematically they are a bit hard to wrap your head around:
  - $\mu = \int_{-\infty}^{\infty} (x - c)^n f(x) dx$  (for moment  $n$  around value  $c$ )
- But intuitively, it's a lot simpler in statistics.

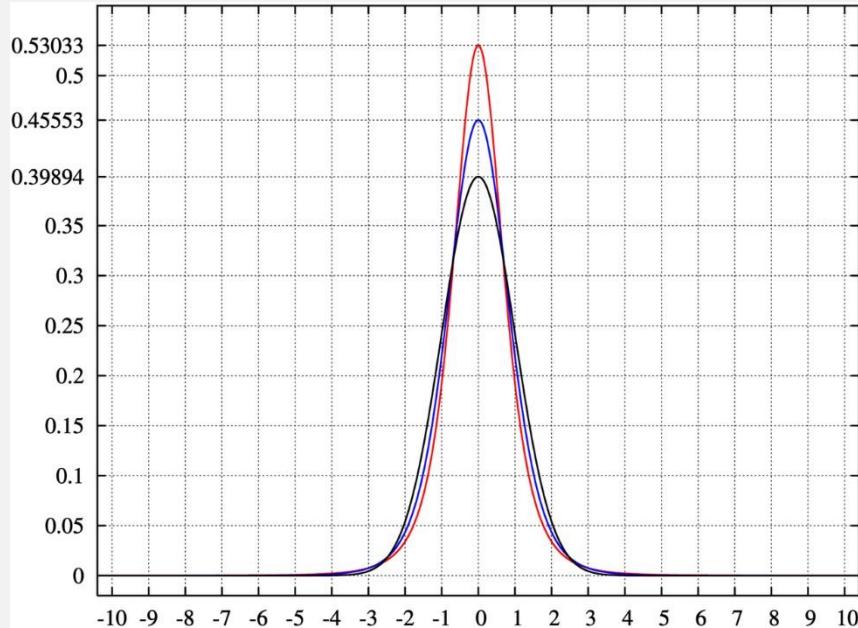
# The third moment is “skew” ( $\gamma$ )

- How “lopsided” is the distribution?
- A distribution with a longer tail on the left will be skewed left, and have a negative skew.



# The fourth moment is “kurtosis”

- How thick is the tail, and how sharp is the peak, compared to a normal distribution?
- Example: higher peaks have higher kurtosis

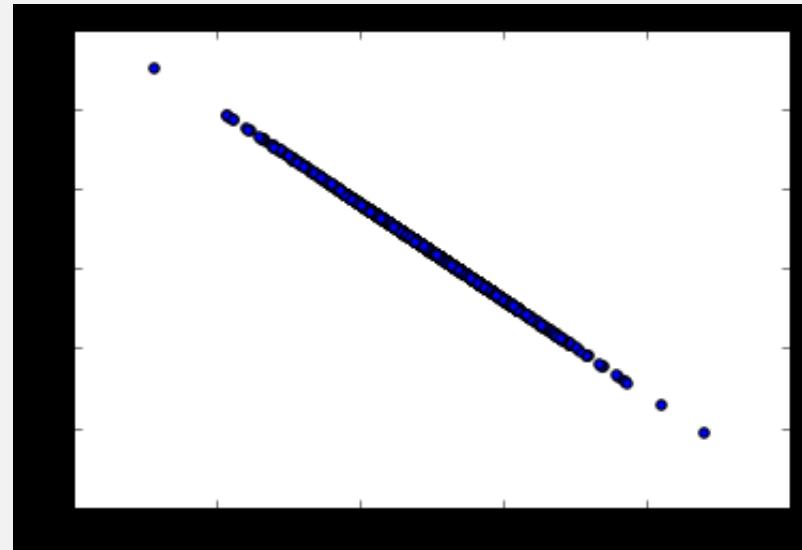
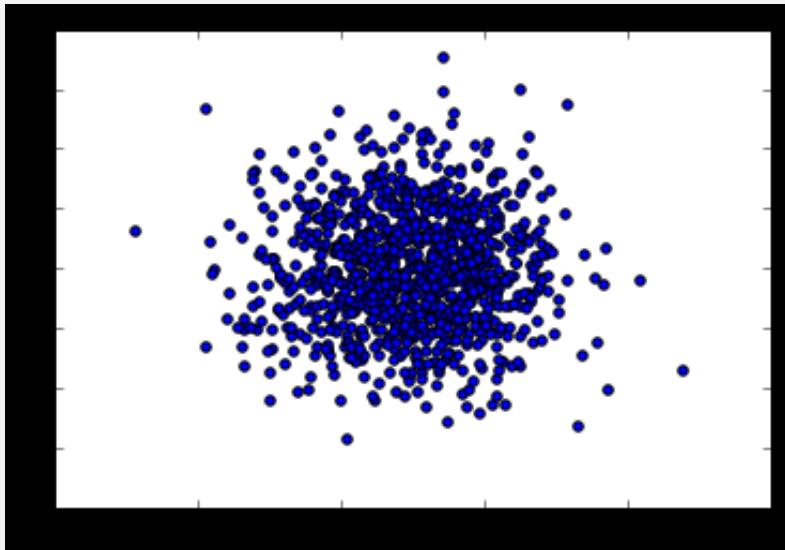


# Covariance and Correlation



# Covariance

- Measures how two variables vary in tandem from their means.



# Measuring covariance

- Think of the data sets for the two variables as high-dimensional vectors
- Convert these to vectors of variances from the mean
- Take the dot product (cosine of the angle between them) of the two vectors
- Divide by the sample size

# Interpreting covariance is hard

- We know a small covariance, close to 0, means there isn't much correlation between the two variables.
- And large covariances – that is, far from 0 (could be negative for inverse relationships) mean there is a correlation
- But how large is “large”?

# That's where correlation comes in!

- Just divide the covariance by the standard deviations of both variables, and that normalizes things.
- So a correlation of -1 means a perfect inverse correlation
- Correlation of 0: no correlation
- Correlation 1: perfect correlation

# Remember: correlation does not imply causation!

- Only a controlled, randomized experiment can give you insights on causation.
- Use correlation to decide what experiments to conduct!

# Conditional Probability



# Conditional Probability

- If I have two events that depend on each other, what's the probability that both will occur?
- Notation:  $P(A,B)$  is the probability of A and B both occurring
- $P(B|A)$  : Probability of B given that A has occurred
- We know:

$$P(B|A) = \frac{P(A,B)}{P(A)}$$

# For example

- I give my students two tests. 60% of my students passed both tests, but the first test was easier – 80% passed that one. What percentage of students who passed the first test also passed the second?
- A = passing the first test, B = passing the second test
- So we are asking for  $P(B|A)$  – the probability of B given A •  $P(B|A) = \frac{P(A,B)}{P(A)} = \frac{0.6}{0.8} = 0.75$

# Bayes' Theorem



# Bayes' Theorem

- Now that you understand conditional probability, you can understand Bayes' Theorem:

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)}$$

# Bayes' Theorem to the rescue

- Drug testing is a common example. Even a “highly accurate” drug test can produce more false positives than true positives.
- Let's say we have a drug test that can accurately identify users of a drug 99% of the time, and accurately has a negative result for 99% of non- users. But only 0.3% of the overall population actually uses this drug.



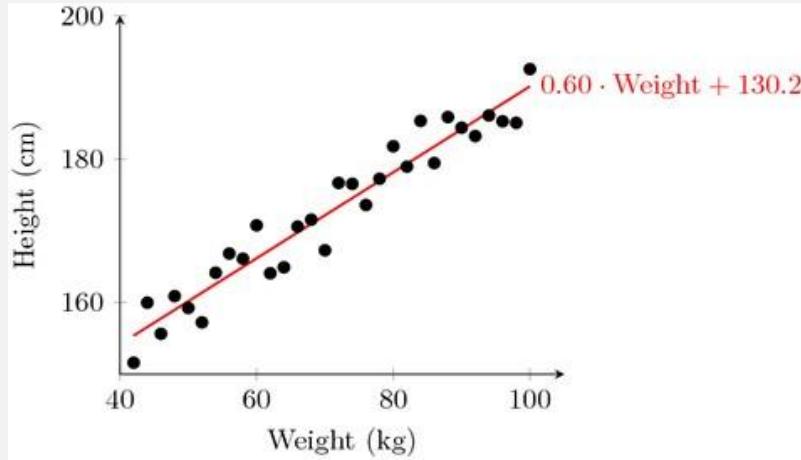
# Bayes' Theorem to the rescue

- Event A = Is a user of the drug, Event B = tested positively for the drug.
- We can work out from that information that  $P(B)$  is 1.3% ( $0.99 * 0.003 + 0.01 * 0.997$  – the probability of testing positive if you do use, plus the probability of testing positive if you don't.)

$$\bullet P(A|B) = \frac{P(A)P(B|A)}{P(B)} = \frac{0.003 * 0.99}{0.013} = 22.8\%$$

# Linear Regression

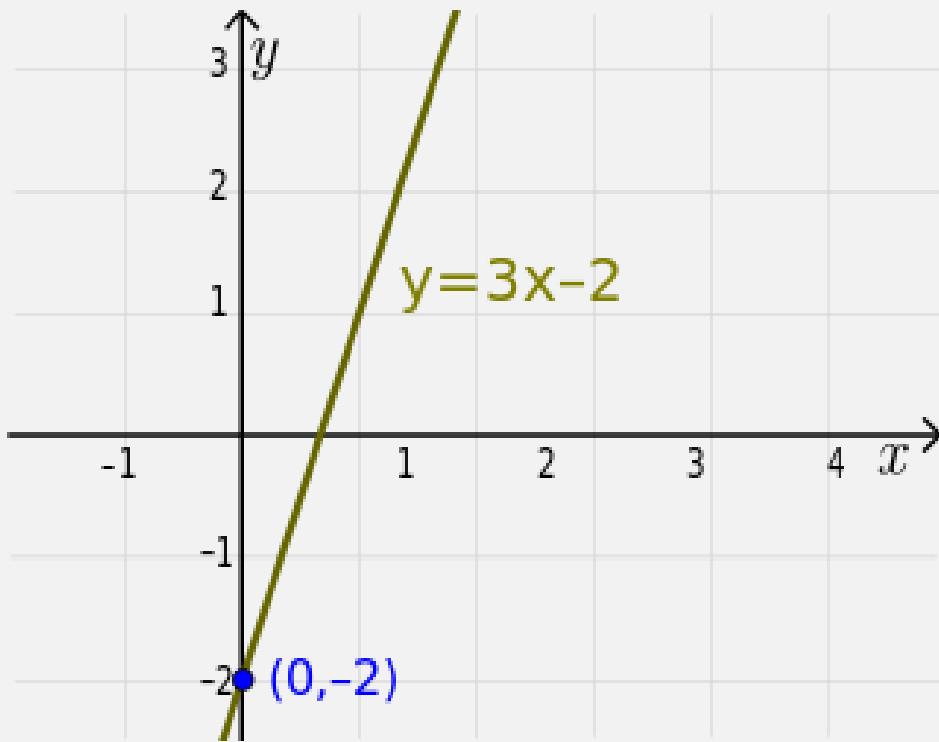
- Fit a line to a data set of observations
- Use this line to predict unobserved values
- I don't know why they call it "regression." It's really misleading



# Linear Regression: How does it work?

- Usually using “least squares”
- Minimizes the squared-error between each point and the line
- Remember the slope-intercept equation of a line?  
 $y=mx+b$
- The slope is the correlation between the two variables times the standard deviation in Y, all divided by the standard deviation in X.
  - Neat how standard deviation how some real mathematical

# Linear Regression: How does it work?

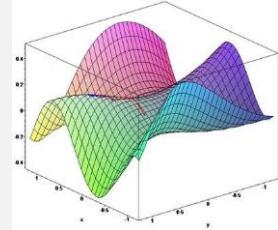


# Linear Regression: How does it work?

- Least squares minimizes the sum of squared errors.
- This is the same as maximizing the likelihood of the observed data if you start thinking of the problem in terms of probabilities and probability distribution functions
- This is sometimes called “maximum likelihood estimation”

# More than one way to do it

- Gradient Descent is an alternate method to least squares.
- Basically iterates to find the line that best follows the contours defined by the data.
- Can make sense when dealing with 3D data
- Easy to try in Python and just compare the results to least squares



But usually least squares is a <sup>50</sup> perfectly good choice.

# Measuring error with r-squared

- How do we measure how well our line fits our data?
- R-squared (aka coefficient of determination) measures:

# Computing r-squared

$$1.0 - \frac{\text{sum of squared errors}}{\text{sum of squared variation from mean}}$$

# Interpreting r-squared

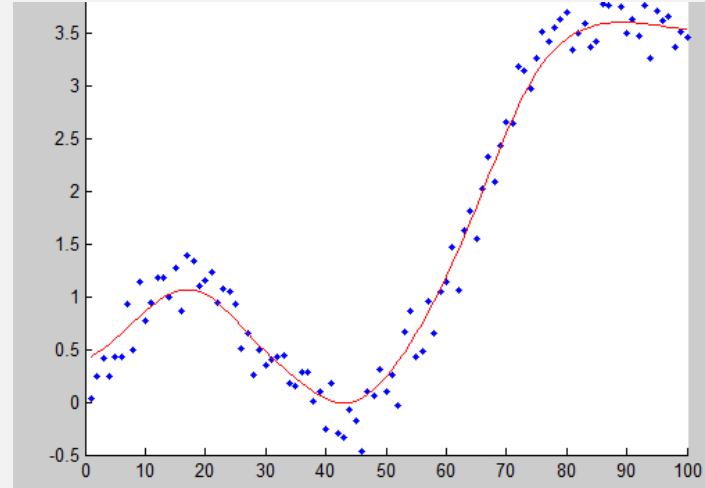
- Ranges from 0 to 1
- 0 is bad (none of the variance is captured), 1 is good (all of the variance is captured).

# Polynomial Regression



# Why limit ourselves to straight lines?

- Not all relationships are linear.
- Linear formula:  $y = mx + b$ 
  - This is a “first order” or “first degree” polynomial, as the power of  $x$  is 1
- Second order polynomial:  $y = ax^2 + bx + c$
- Third order:  $y = ax^3 + bx^2 + cx + d$



# Beware overfitting

- Don't use more degrees than you need
- Visualize your data first to see how complex of a curve there might really be
- Visualize the fit – is your curve going out of its way to accommodate outliers?
- A high r-squared simply means your curve fits your *training data* well; but it may not be a good predictor.
- Later we'll talk about more principled ways to detect overfitting (train/test)

# Multiple Regression



# Multiple Regression

- What if more than one variable influences the one you're interested in?
- Example: predicting a price for a car based on its many attributes (body style, brand, mileage, etc.)



# Still uses least squares

- We just end up with coefficients for each factor.
  - For example,  $price = \alpha + \beta_1 \text{mileage} + \beta_2 \text{age} + \beta_3 \text{doors}$
  - These coefficients imply how important each factor is (if the data is all normalized!)
  - Get rid of ones that don't matter!
- Can still measure fit with r-squared
- Need to assume the different factors are not themselves dependent on each other.

# Multi-Level Models



# Multi-Level Models

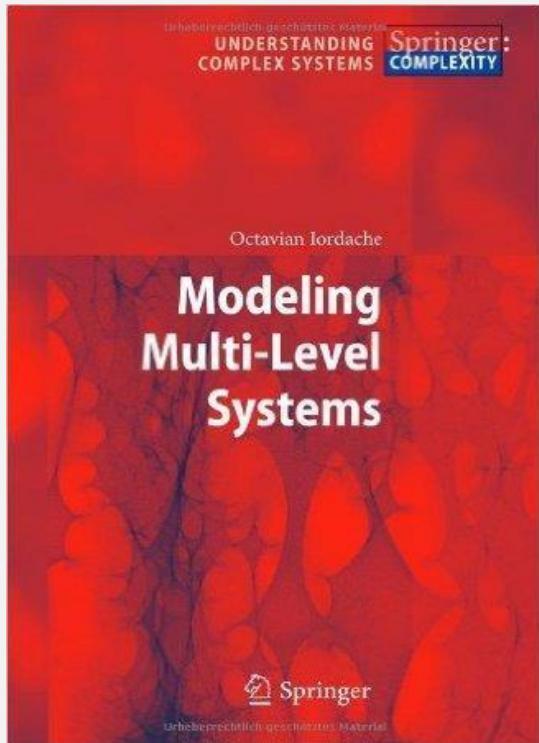
- The concept is that some effects happen at various levels.
- Example: your health depends on a hierarchy of the health of your cells, organs, you as a whole, your family, your city, and the world you live in.

# Modeling multiple levels

- You must identify the factors that affect the outcome you're trying to predict at each level.
- For example – SAT scores might be predicted based on the genetics of individual children, the home environment of individual children, the crime rate of the neighborhood they live in, the quality of the teachers in their school, the funding of their school district, and the education policies of their state.

# Doing this is hard.

- I just want you to be aware of the concept, as multi-level models showed up on some data science job requirements I've seen.



# Supervised and Unsupervised Machine Learning



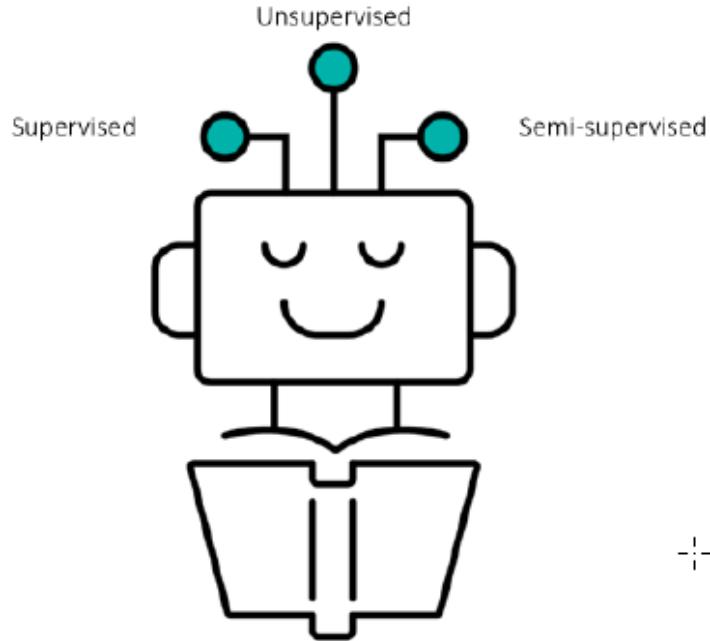
# What is machine learning?

- Algorithms that can learn from observational data, and can make predictions based on it.



# What is machine learning?

## Types of Machine Learning



# Unsupervised Learning

- The model is not given any “answers” to learn from; it must make sense of the data just given the observations themselves.
- Example: group (cluster) some objects together into 2 different sets. But I don’t tell you what the “right” set is for any object ahead of time.

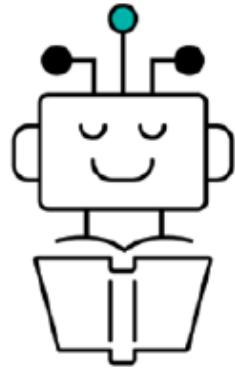


# Unsupervised Learning

- Unsupervised learning sounds awful! Why use it?
- Maybe you don't know what you're looking for – you're looking for *latent variables*.
- Example: clustering users on a dating site based on their information and behavior.  
Perhaps you'll find there are groups of people that emerge that don't conform to your known stereotypes.

# Unsupervised Learning

## Unsupervised Learning Defined



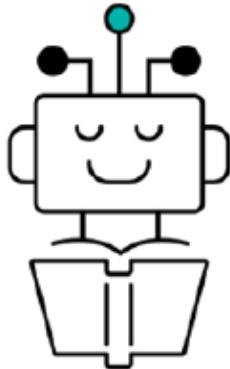
Unsupervised  
Learning

Unsupervised learning uses raw, unlabeled data and has no knowledge of the output label.



# Unsupervised Learning

## Unsupervised Learning: How it Works

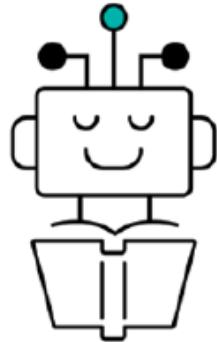


Unsupervised  
Learning

- 1 Load unlabeled raw data
- 2 Algorithm infers patterns from the data on its own
- 3 Algorithm identifies groups of data that exhibit similar patterns
- 4 Provides output

# Unsupervised Learning

## Unsupervised Learning: Pros and Cons



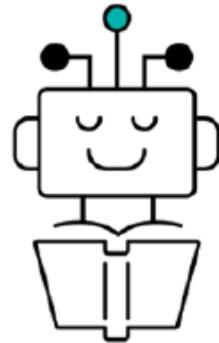
Unsupervised  
Learning

| Pros                | Cons                          |
|---------------------|-------------------------------|
| Very fast to start  | Difficult to measure accuracy |
| Disruptive insights | Requires more experience      |
|                     | Curse of dimensionality       |

- +

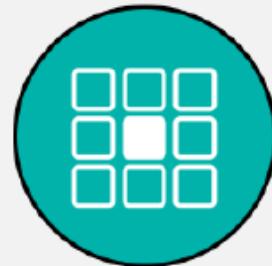
# Unsupervised Learning

## Unsupervised Algorithms: Common Use Case



Unsupervised  
Learning

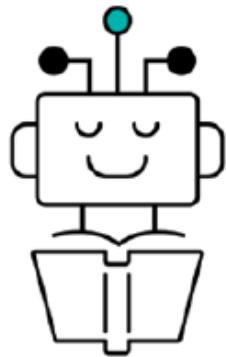
Use case example for an unsupervised learning problem includes cluster analysis



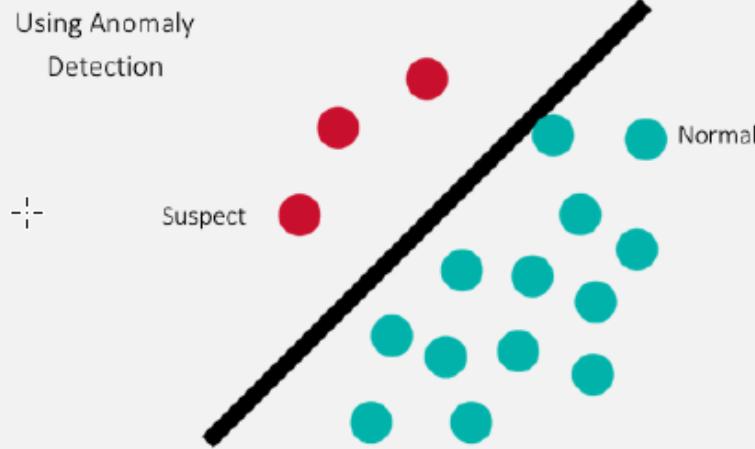
Security Anomaly  
Detection

# Unsupervised Learning

## Unsupervised Learning: Use Case Example

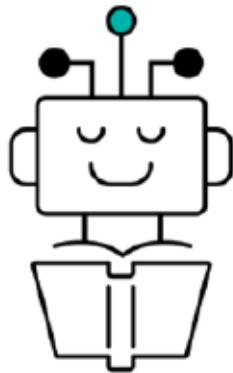


Unsupervised  
Learning



# Unsupervised Learning

## Unsupervised Algorithms Table



Unsupervised  
Learning

| Type         | Algorithm or Task                              |
|--------------|--|
| Unsupervised | K-Means: Cluster Analysis                      |
| Unsupervised | Association Rule Learning                      |
| Unsupervised | Dimensionality Reduction Techniques (PCA, SVD) |

-+-

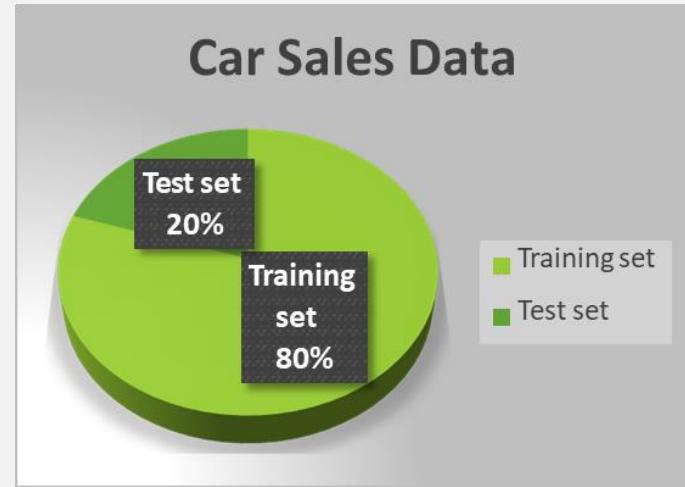
# Supervised Learning

- In supervised learning, the data the algorithm “learns” from comes with the “correct” answers.
- The model created is then used to predict the answer for new, unknown values.



# Evaluating Supervised Learning

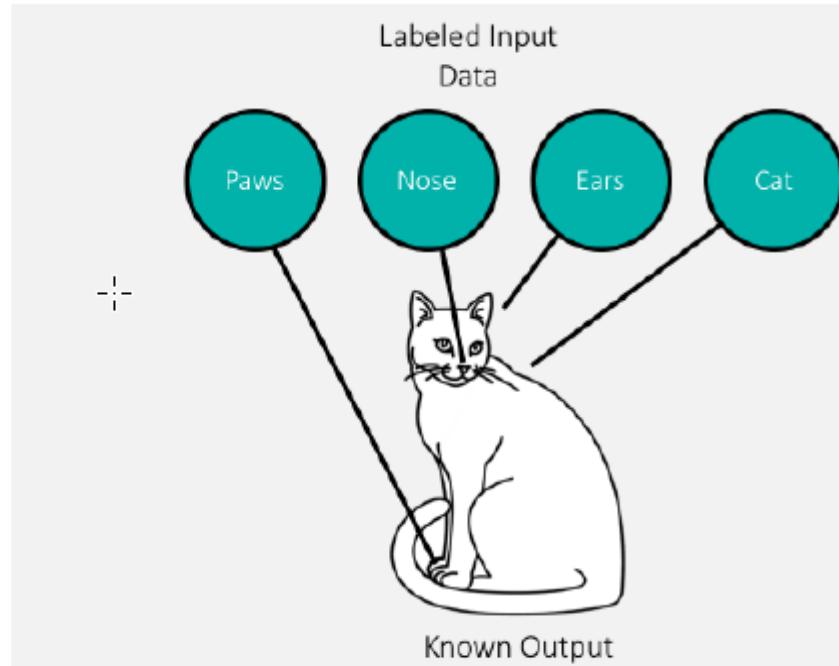
- If you have a set of training data that includes the value you're trying to predict – you don't have to guess if the resulting model is good or not.
- If you have enough training data, you can split it into two parts: a *training* set and a *test* set.



# Supervised Learning

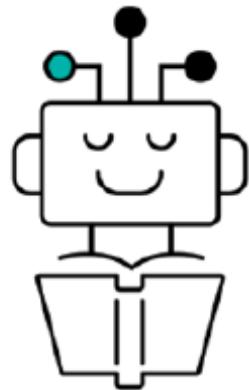
## Supervised Learning: Defined

- Uses labeled training data
- Learns relationships between given inputs to a given output
- Desired output must be part of the labeled data, to be known



# Supervised Learning

## Supervised Learning: How it Works



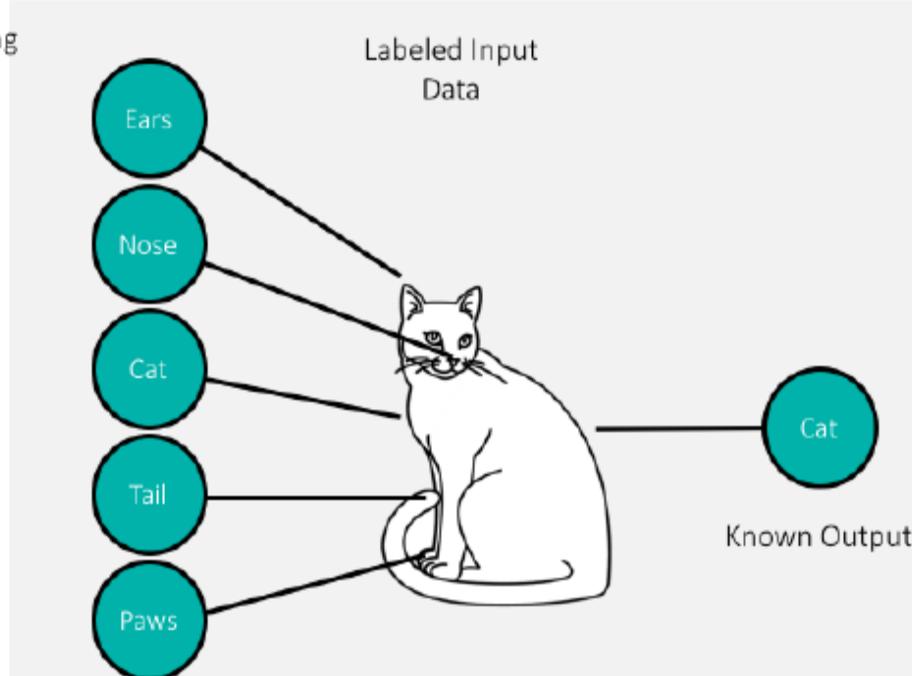
Supervised  
Learning

- 1 Load labeled input data
- 2 Train model on the data: connection to input variables and output is made
- 3 Apply new data to algorithm
- 4 Provides output

# Supervised Learning

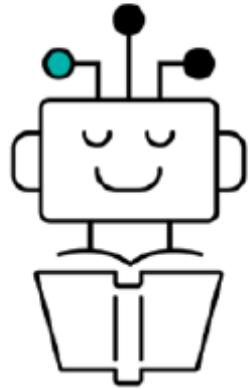
## Supervised Learning: How it Works Example

Supervised learning uses labeled training data to train machines to learn relationships between given inputs to a given output



# Supervised Learning

## Supervised Learning: Pros and Cons



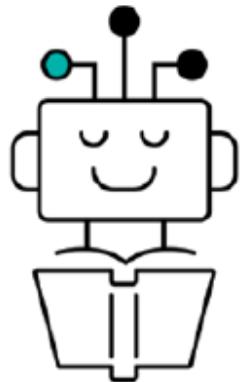
Supervised  
Learning

| Pros                     | Cons                      |
|--------------------------|---------------------------|
| Very clear objective     | Often labor-intensive     |
| Easy to measure accuracy | Limited data to work with |
| Controlled training      | Limited insights          |

-|-

# Supervised Learning

## Supervised Algorithms: Classification



Supervised  
Learning

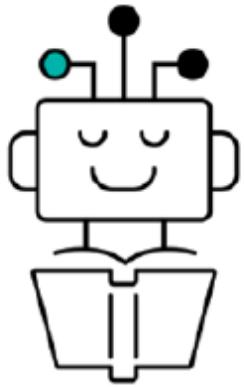
Two general supervised learning categories:

| Type       | Algorithm or Task   |
|------------|---|
| Supervised | Classification (used to predict a categorical result)               |
| Supervised | Regression (used to predict the output value given the input value) |

-+-

# Supervised Learning

## Supervised Algorithms: Classification Use Cases



Supervised  
Learning

Use case examples for supervised learning algorithms include these simple binary classification problems:



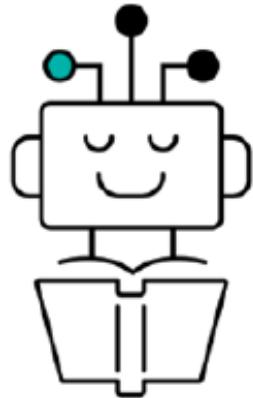
Fraud Detection



Spam Filtering

# Supervised Learning

## Supervised Algorithms: Regression



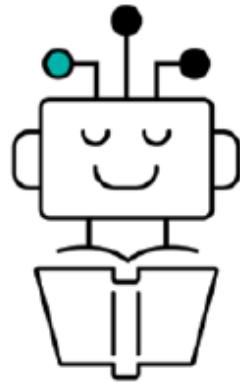
Supervised  
Learning

Regression algorithms are used to predict a continuous numeric result.

| Type       | Algorithm or Task   |
|------------|---|
| Supervised | Classification (used to predict a categorical result)               |
| Supervised | Regression (used to predict the output value given the input value) |

# Supervised Learning

## Supervised Algorithms: Regression Use Cases



Supervised  
Learning

Use case example for a supervised learning regression problem include:



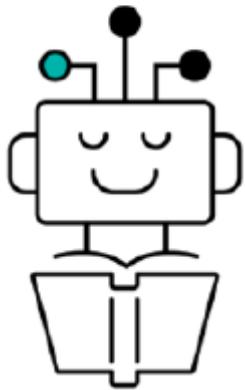
Credit Card  
Model Assessment



Customer/Employee  
Churn

# Supervised Learning

## Supervised Algorithms Table

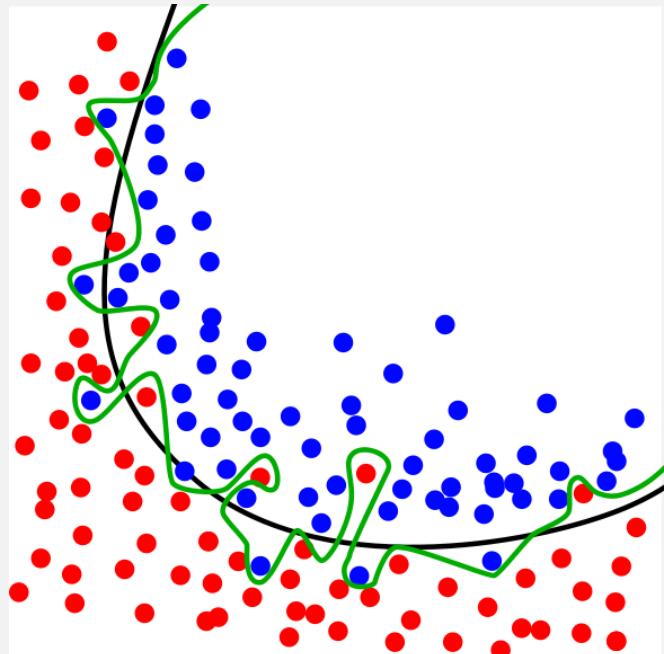


Supervised  
Learning

| Type           | Algorithm or Task                  |
|----------------|------------------------------------|
| Classification | Naïve Bayes                        |
| Classification | Logistic Regression                |
| Classification | Support Vector Machines (SVM)      |
| Regression     | Linear Regression                  |
| Both           | Decision Trees/Random Forest       |
| Both           | K-Nearest Neighbor (KNN)           |
| Both           | Gradient Boosting Algorithms (GBA) |

# Train / Test in practice

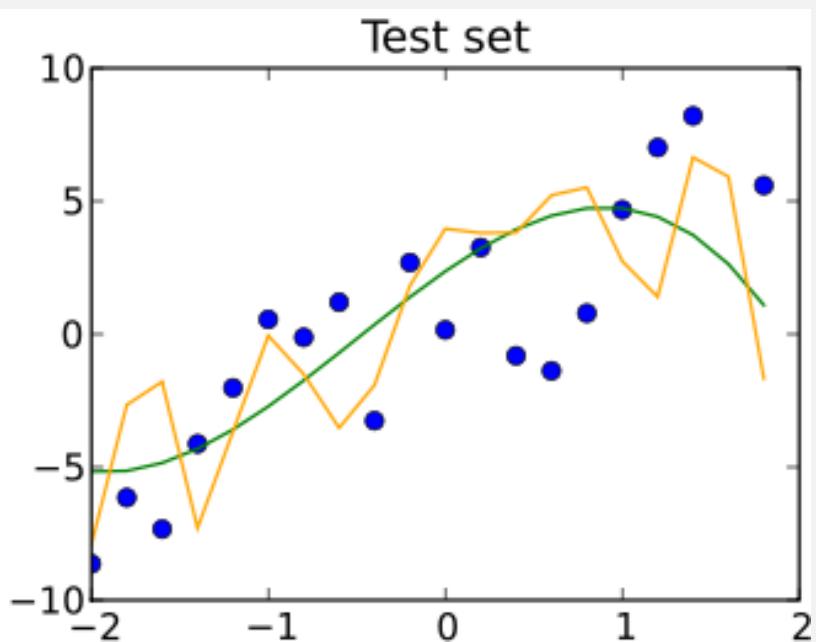
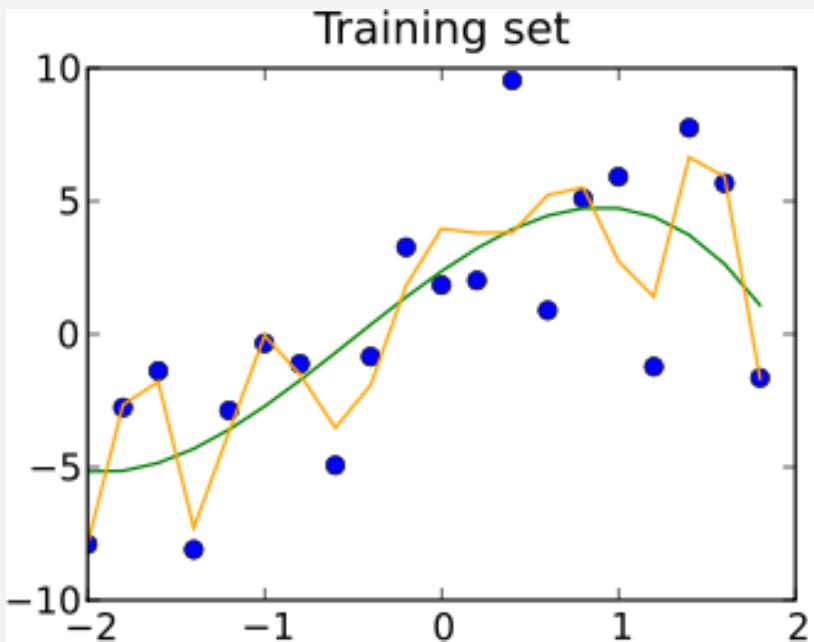
- Need to ensure both sets are large enough to contain representatives of all the variations and outliers in the data you care about
- The data sets must be selected randomly
- Train/test is a great way to guard against *overfitting*



# Train/Test is not Infallible

- Maybe your sample sizes are too small
- Or due to random chance your train and test sets look remarkably similar
- Overfitting can still happen

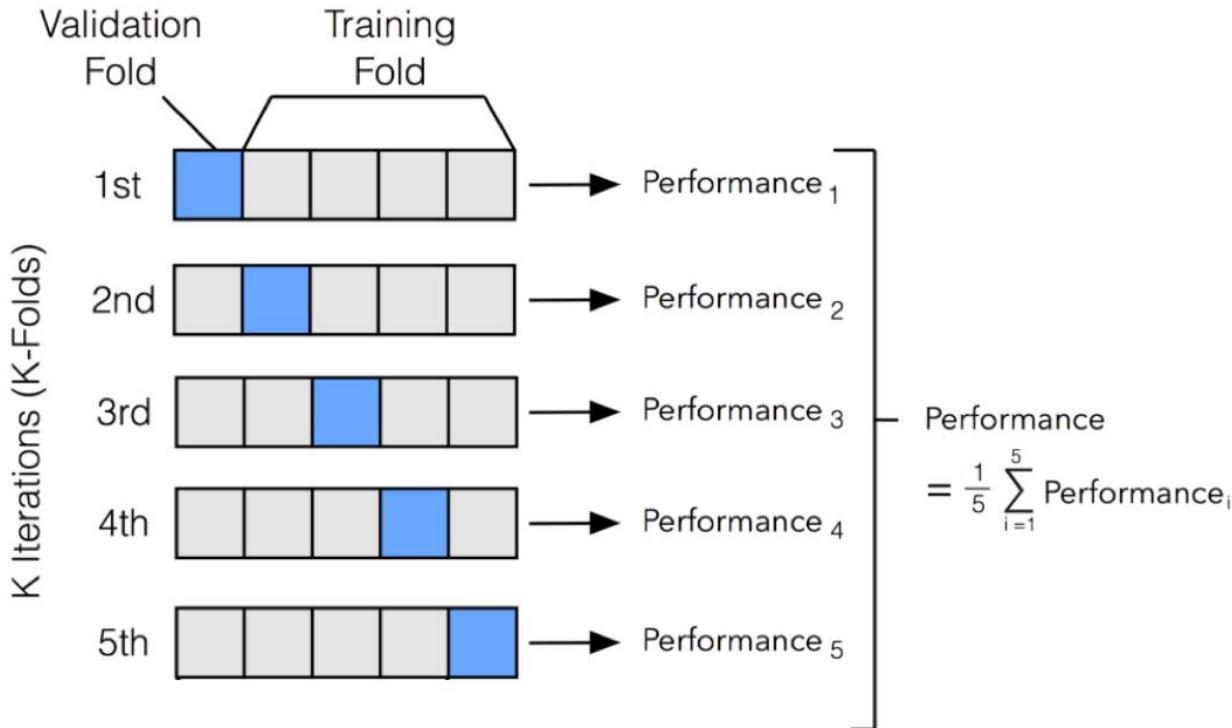
# Train/Test is not Infallible



# K-fold Cross Validation

- One way to further protect against overfitting is *K-fold cross validation*
- Sounds complicated. But it's a simple idea:
  - Split your data into K randomly-assigned segments
  - Reserve one segment as your test data
  - Train on each of the remaining K-1 segments and measure their performance against the test set
  - Take the average of the K-1 r-squared scores

# K-fold Cross Validation



# Multivariate Regression

- **Multivariate Regression** is a method used to measure the degree at which more than one independent variable (**predictors**) and more than one dependent variable (**responses**), are linearly related.
- The method is broadly used to predict the behavior of the response variables associated to changes in the predictor variables, once a desired degree of relation has been established.

- Can a supermarket owner maintain stock of water, ice cream, frozen foods, canned foods and meat as a function of temperature, tornado chance and gas price during tornado season in June?
  - From this question, several obvious assumptions can be drawn:
    - If it is too hot, ice cream sales increase;
    - If a tornado hits, water and canned foods sales increase while ice cream, frozen foods and meat will decrease;
    - If gas prices increase, prices on all goods will increase.
    - A mathematical model, based on multivariate regression analysis will address this and other more complicated questions.

- The **Simple Regression** model, relates one predictor and one response.

# Bayesian Methods



# Remember Bayes' Theorem?

$$\bullet P(A|B) = \frac{P(A)P(B|A)}{P(B)}$$

- Let's use it for machine learning! I want a spam classifier.
- Example: how would we express the probability of an email being spam if it contains the word “free”?

$$\bullet P(Spam | Free) = \frac{P(Spam)P(Free | Spam)}{P(Free)}$$

# Remember Bayes' Theorem?

$$P(c | x) = \frac{P(x | c)P(c)}{P(x)}$$

Diagram illustrating the components of Bayes' Theorem:

- Likelihood:  $P(x | c)$
- Class Prior Probability:  $P(c)$
- Posterior Probability:  $P(c | x)$
- Predictor Prior Probability:  $P(x)$

```
graph TD; A[P(x | c)] --> C[P(c | x)]; A --> D[P(x)]; B[P(c)] --> C; B --> D
```

$$P(c | X) = P(x_1 | c) \times P(x_2 | c) \times \cdots \times P(x_n | c) \times P(c)$$

# Remember Bayes' Theorem?

$$\frac{\text{Old probability}}{P(\text{spam})} \times \frac{\text{Strength of new evidence}}{P(\text{"enlarge"}|\text{spam})} = \frac{\text{New probability}}{P(\text{spam}|\text{"enlarge"})}$$
$$\frac{P(\text{spam})}{P(\text{not spam})} \times \frac{P(\text{"enlarge"}|\text{spam})}{P(\text{"enlarge"}|\text{not spam})} = \frac{P(\text{spam}|\text{"enlarge"})}{P(\text{not spam}|\text{"enlarge"})}$$

$$\frac{0.8}{0.2} \times \frac{0.1}{0.0001} = 4,000:1 \text{ odds of an email being spam, if it contains the word "enlarge"}$$

# What about all the other words?

- We can construct  $P(\text{Spam} \mid \text{Word})$  for every (meaningful) word we encounter during training
- Then multiply these together when analyzing a new email to get the probability of it being spam.



# Sounds like a lot of work.

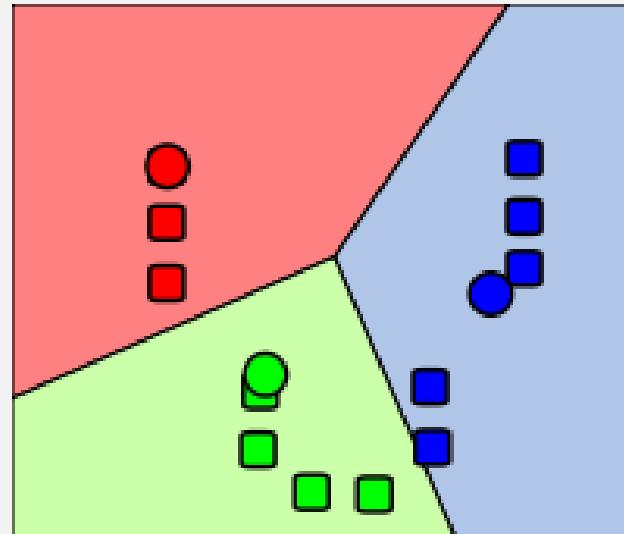
- Scikit-learn to the rescue!
- The CountVectorizer lets us operate on lots of words at once, and MultinomialNB does all the heavy lifting on Naïve Bayes.
- We'll train it on known sets of spam and “ham” (non-spam) emails
  - So this is supervised learning!
- Let's do this

# K-Means Clustering



# K-Means Clustering

- Attempts to split data into K groups that are closest to K centroids
- Unsupervised learning – uses only the positions of each data point



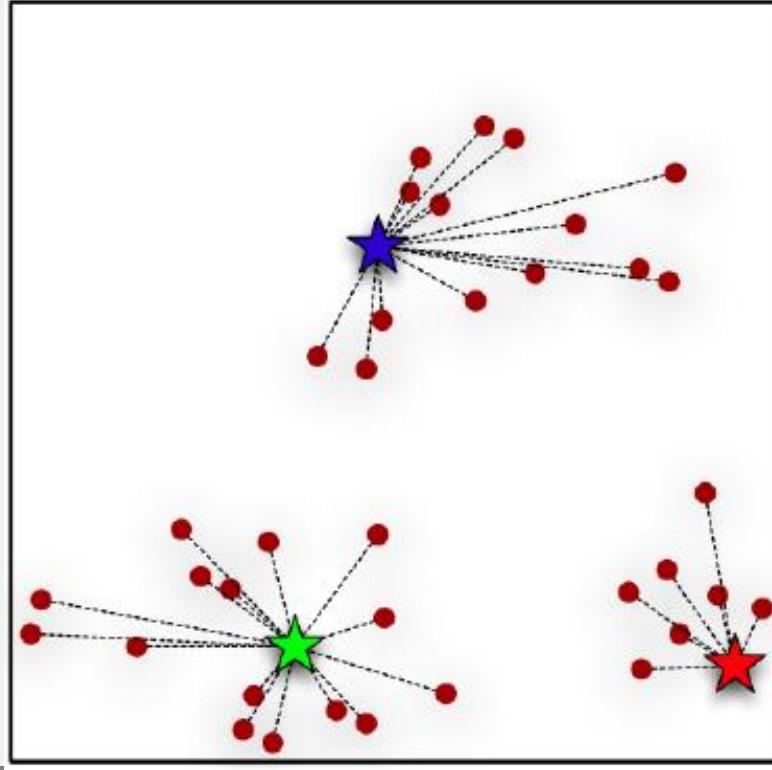
# K-Means Clustering

- Sounds fancy! Wow! Unsupervised machine learning!  
Clusters! K!
- Actually how it works is really simple.
  - Randomly pick K centroids (k-means)
  - Assign each data point to the centroid it's closest to
  - Recompute the centroids based on the average position of each centroid's points
  - Iterate until points stop changing assignment to centroids

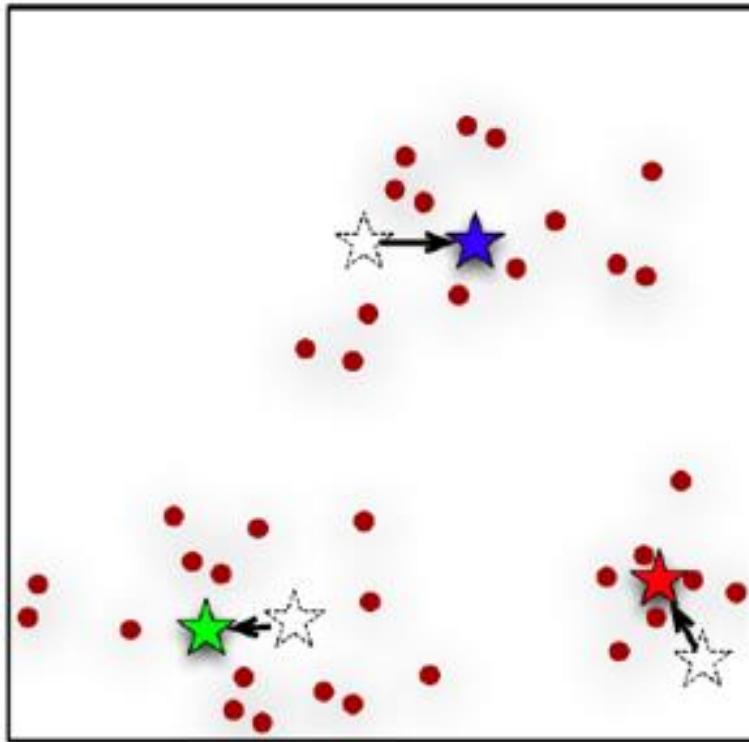
# Graphical example



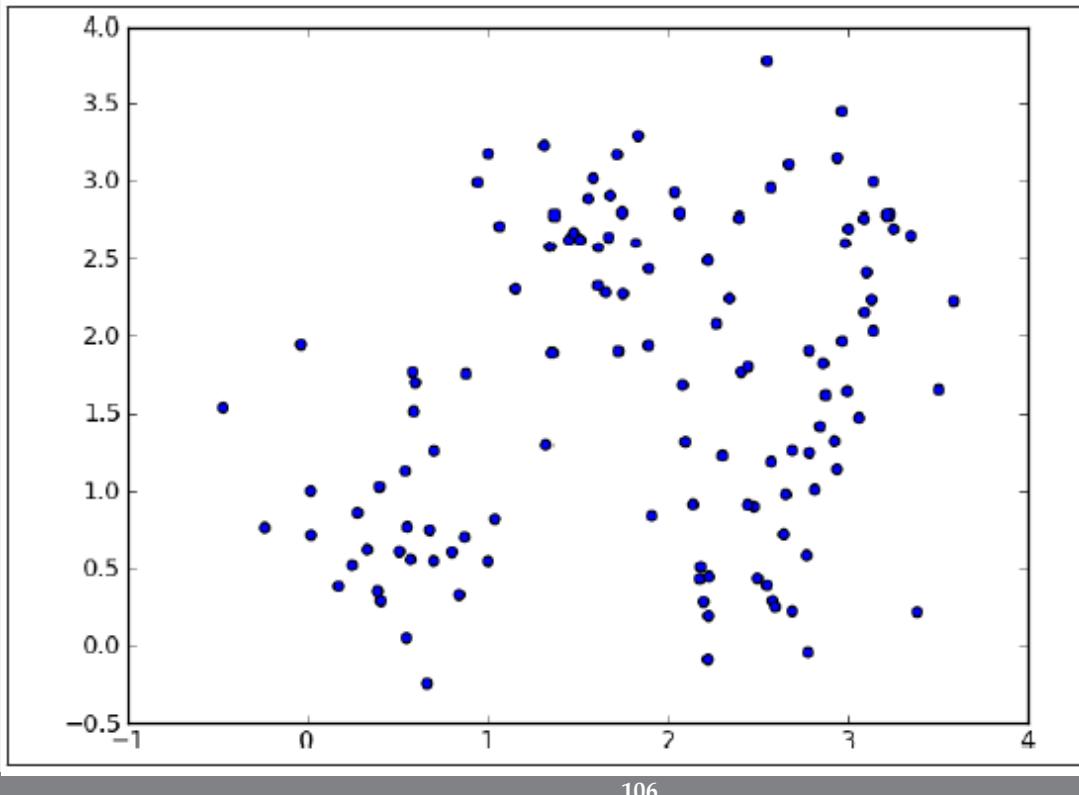
# Graphical example



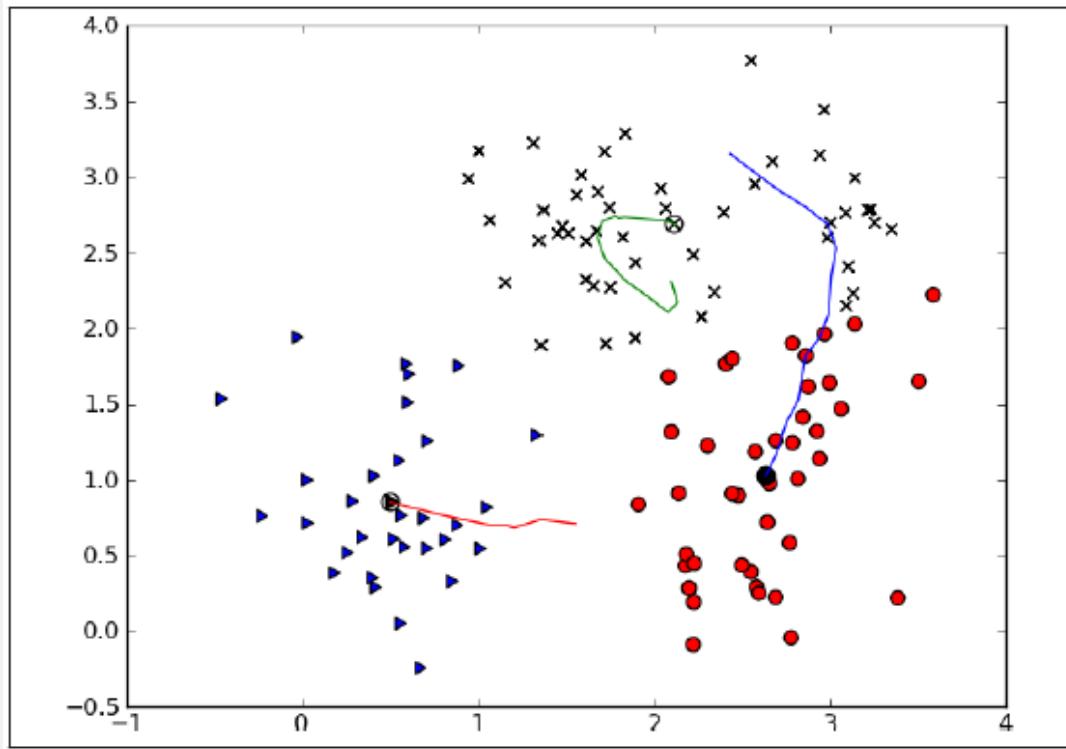
# Graphical example



# Graphical example



# Graphical example



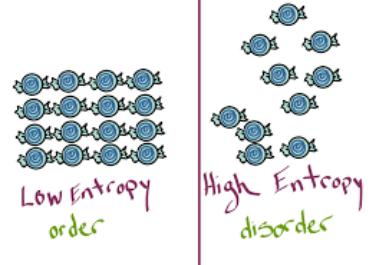
# K-Means Clustering Gotchas

- Choosing K
- Avoiding local minima
- Labeling the clusters

# Entropy



# Entropy

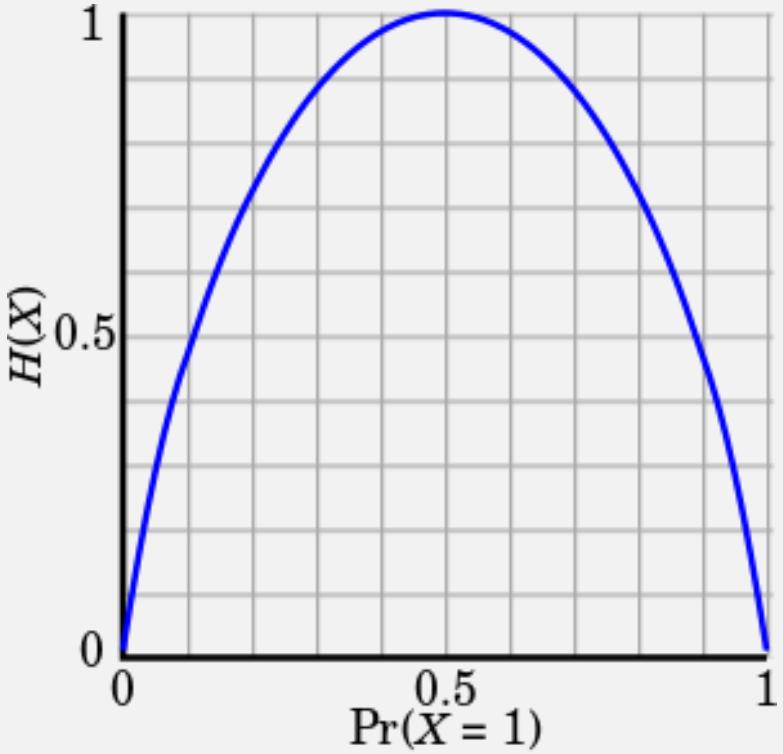


- A measure of a data set's disorder – how same or different it is.
- If we classify a data set into N different classes (example: a data set of animal attributes and their species)
  - The entropy is 0 if all of the classes in the data are the same (everyone is an iguana)
  - The entropy is high if they're all different



# Computing Entropy

- $H(S) = -p_1 \ln p_1 - \dots - p_n \ln p_n$
- $p_i$  represents the proportion of the data labeled for each class
- Each term looks like this:



# Decision Trees

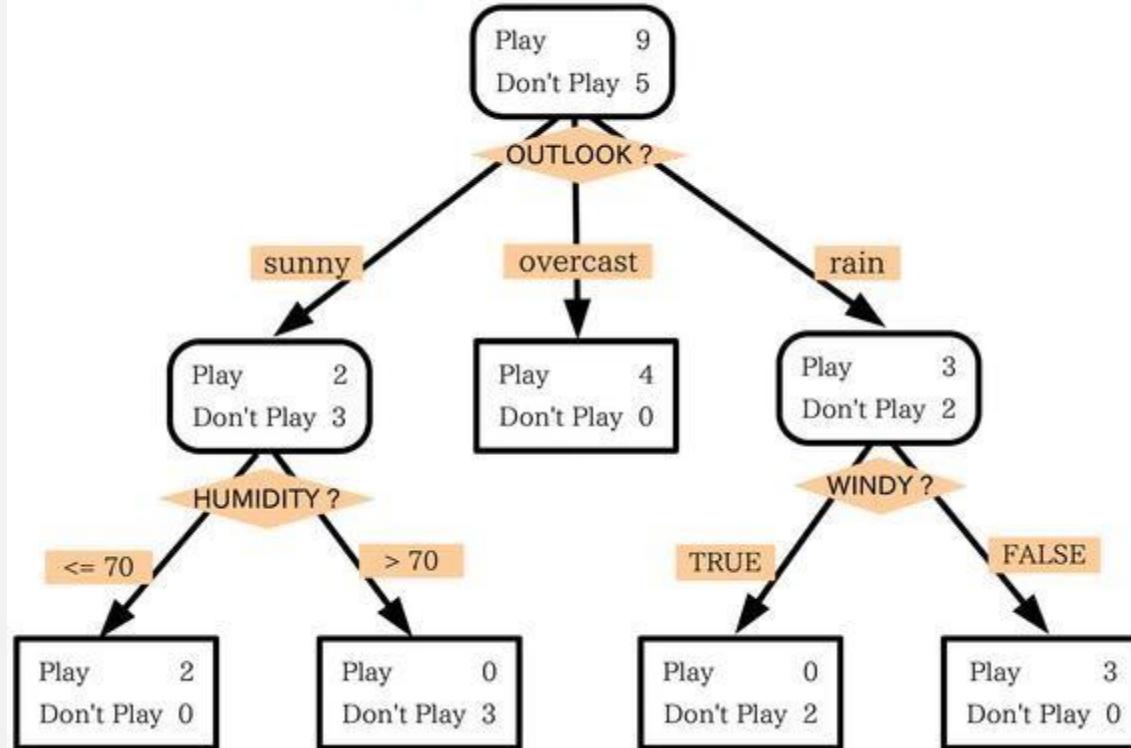


# Decision Trees

- You can actually construct a flowchart to help you decide a classification for something with machine learning, this is called a Decision Tree
- Another form of supervised learning
  - Give it some sample data and the resulting classifications
  - Out comes a tree!

# Decision Trees

Dependent variable: PLAY



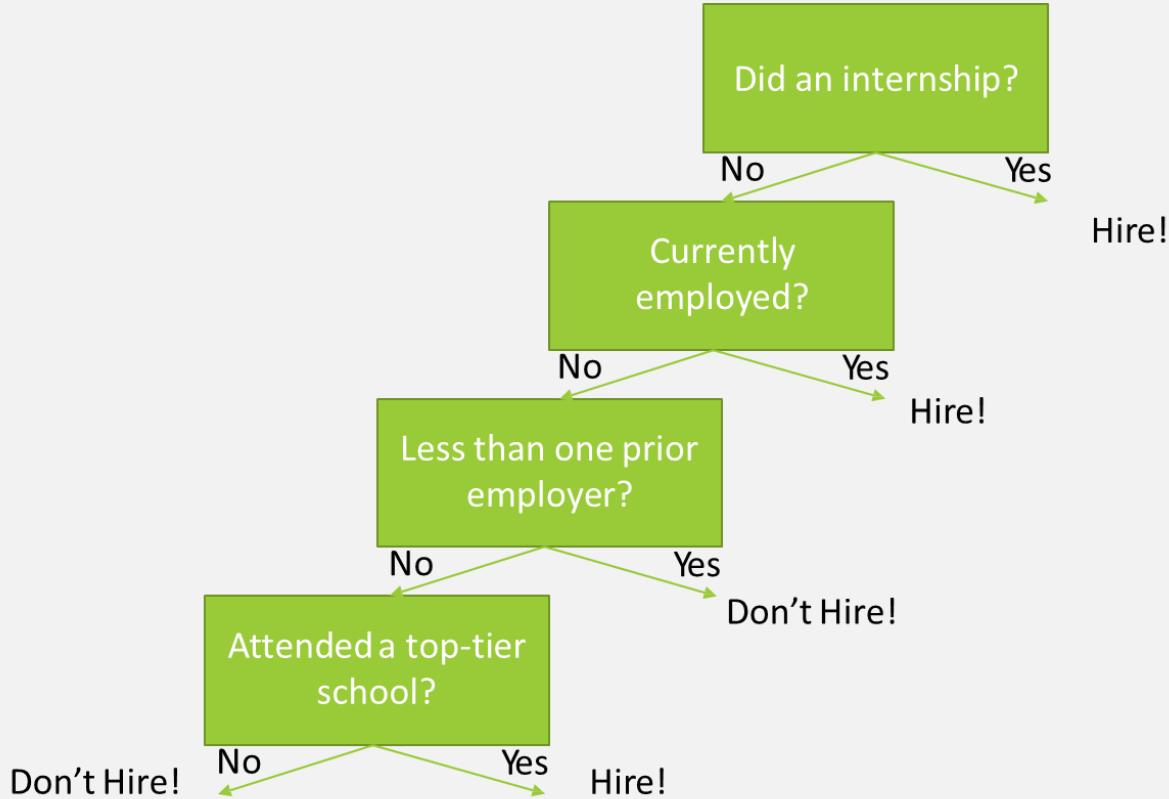
# Decision Tree example

- You want to build a system to filter out resumes based on historical hiring data
- You have a database of some important attributes of job candidates, and you know which ones were hired and which ones weren't
- You can train a decision tree on this data, and arrive at a system for predicting whether a candidate will get hired based on it!

# Totally Fabricated Hiring Data

| Candidate ID | Years Experience | Employed? | Previous employers | Level of Education | Top-tier school | Interned | Hired |
|--------------|------------------|-----------|--------------------|--------------------|-----------------|----------|-------|
| 0            | 10               | 1         | 4                  | 0                  | 0               | 0        | 1     |
| 1            | 0                | 0         | 0                  | 0                  | 1               | 1        | 1     |
| 2            | 7                | 0         | 6                  | 0                  | 0               | 0        | 0     |
| 3            | 2                | 1         | 1                  | 1                  | 1               | 0        | 1     |
| 4            | 20               | 0         | 2                  | 2                  | 1               | 0        | 0     |

# Totally Fabricated Should-I-Hire-This-Person Tree



# How Decision Trees Work

- At each step, find the attribute we can use to partition the data set to minimize the *entropy* of the data at the next step
- Fancy term for this simple algorithm: ID3
- It is a *greedy algorithm* – as it goes down the tree, it just picks the decision that reduce entropy the most at that stage.

# Random Forests

- Decision trees are very susceptible to overfitting
- To fight this, we can construct several alternate decision trees and let them “vote” on the final classification
  - Randomly re-sample the input data for each tree (fancy term for this: *bootstrap aggregating* or *bagging*)
  - Randomize a subset of the attributes each step is allowed to choose from

# Ensemble Learning



# Ensemble Learning

- Random Forests was an example of *ensemble learning*
- It just means we use multiple models to try and solve the same problem, and let them vote on the results.

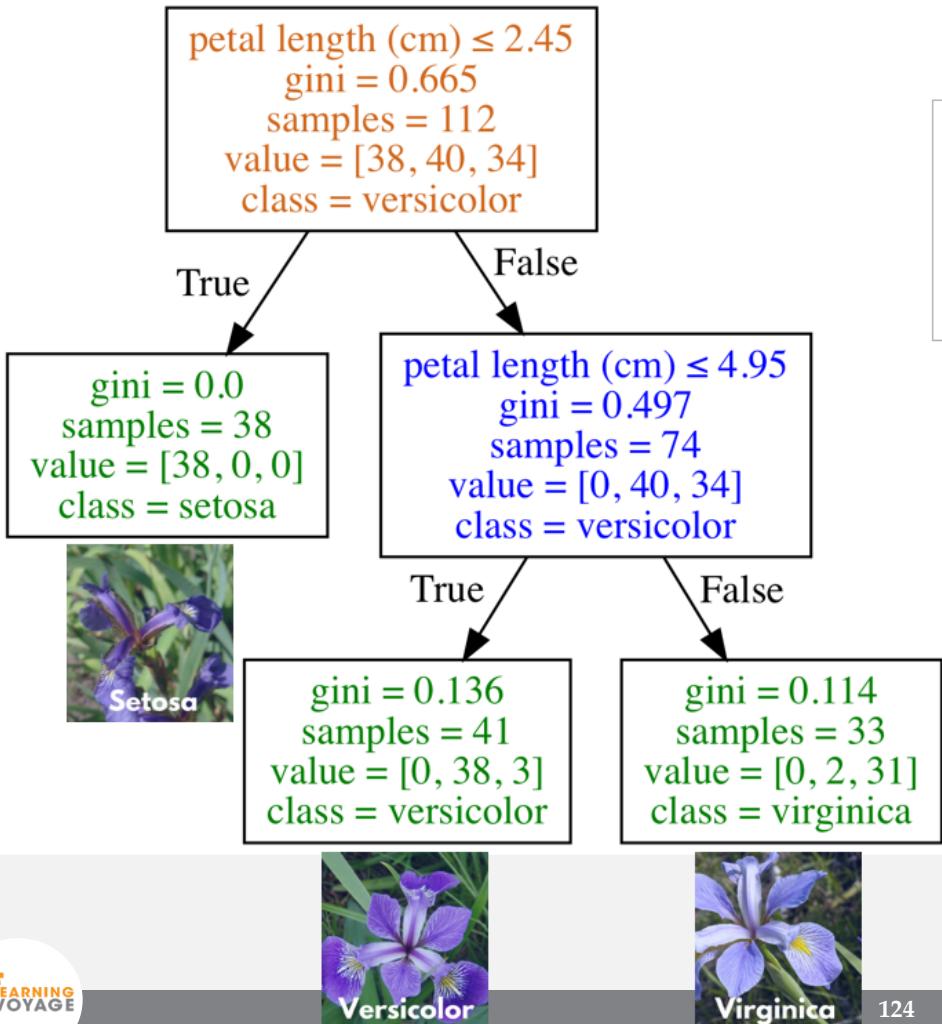


# Ensemble Learning

- Random Forests uses *bagging* (bootstrap aggregating) to implement ensemble learning
  - Many models are built by training on randomly-drawn subsets of the data
- *Boosting* is an alternate technique where each subsequent model in the ensemble boosts attributes that address data mis-classified by the previous model

# Advanced Ensemble Learning: Ways to Sound Smart

- Bayes Optimal Classifier
- Bayesian Parameter Averaging
- Bayesian Model Combination



## Type of Node

- █ Root + Decision Node
- █ Decision Node
- █ Leaf/Terminal Node

# Support Vector Machines

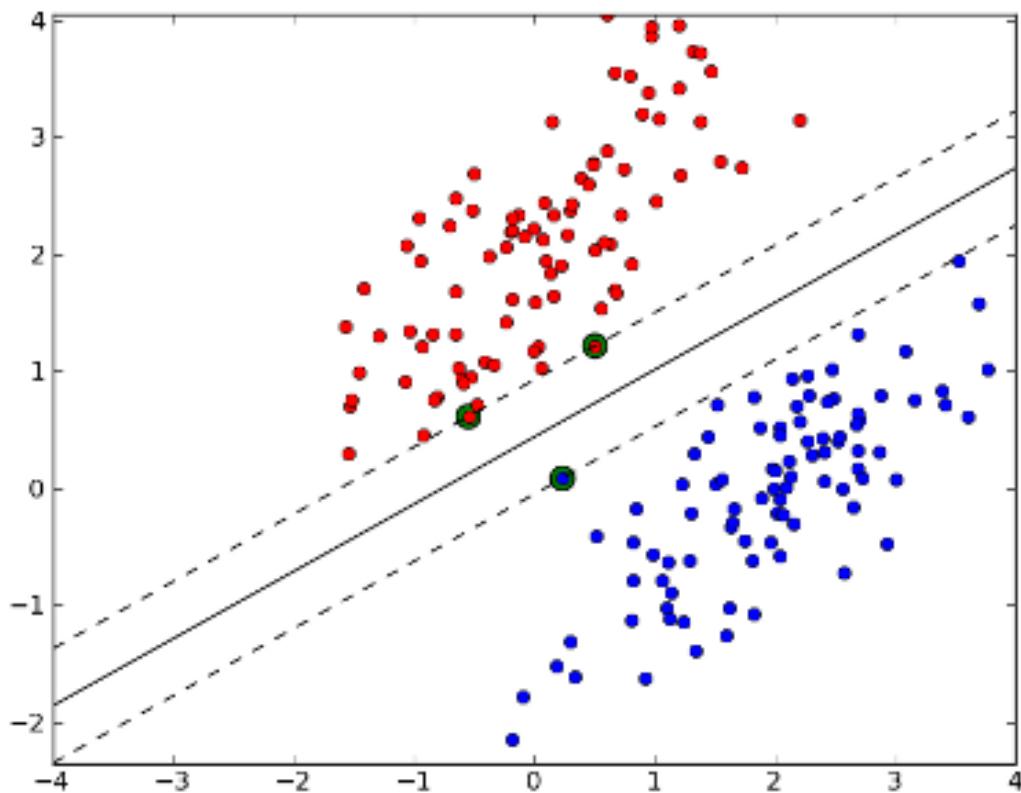


# Support Vector Machines

- Works well for classifying higher-dimensional data (lots of features)
- Finds higher-dimensional *support vectors* across which to divide the data (mathematically, these support vectors define hyperplanes. Needless to say I'm not going to get into the mathematical details!)

# Higher dimensions? Hyperplanes? Huh?

- The important point is that SVM's employ some advanced mathematical trickery to cluster data, and it can handle data sets with lots of features.
- It's also fairly expensive – the “kernel trick” is the only thing that makes it possible.

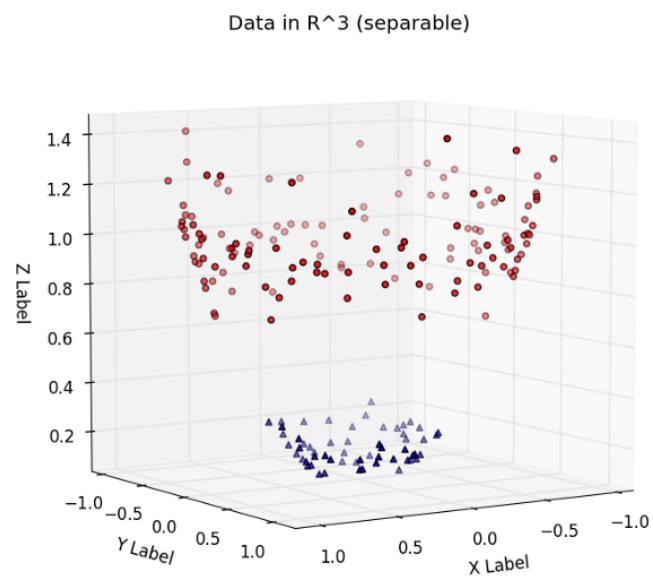
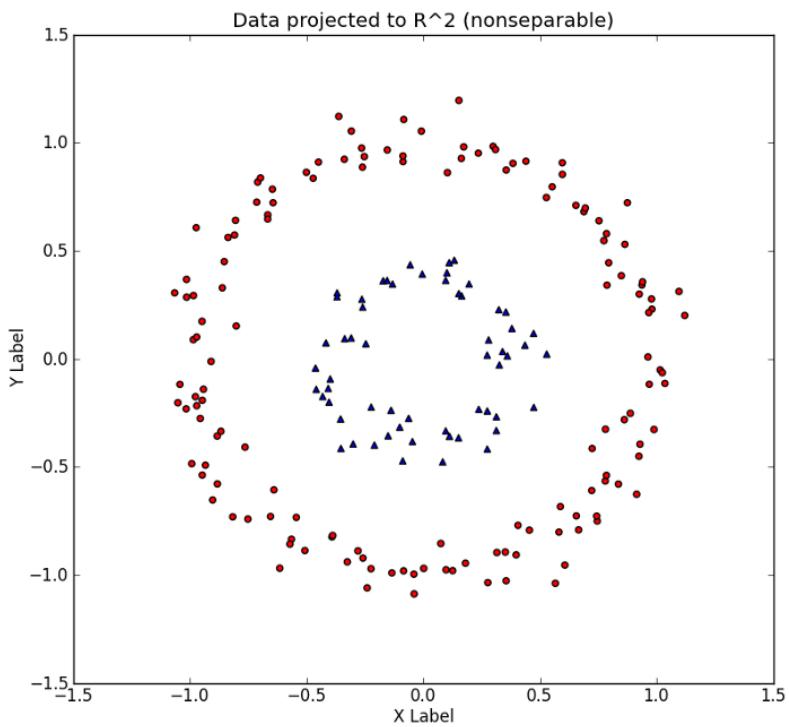


minimize:

$$W(\alpha) = - \sum_{i=1}^{\ell} \alpha_i + \frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} y_i y_j \alpha_i \alpha_j \mathbf{x}_i \mathbf{x}_j$$

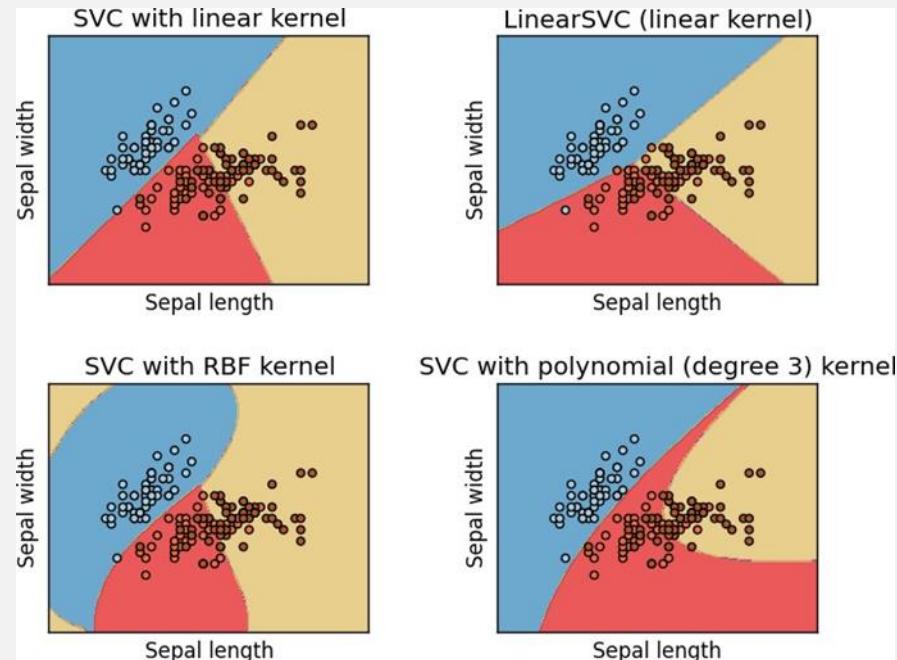
subject to:

$$\sum_{\substack{i=1 \\ 0 \leq \alpha_i \leq C}}^{\ell} y_i \alpha_i = 0 \quad (4)$$



# Support Vector Classification

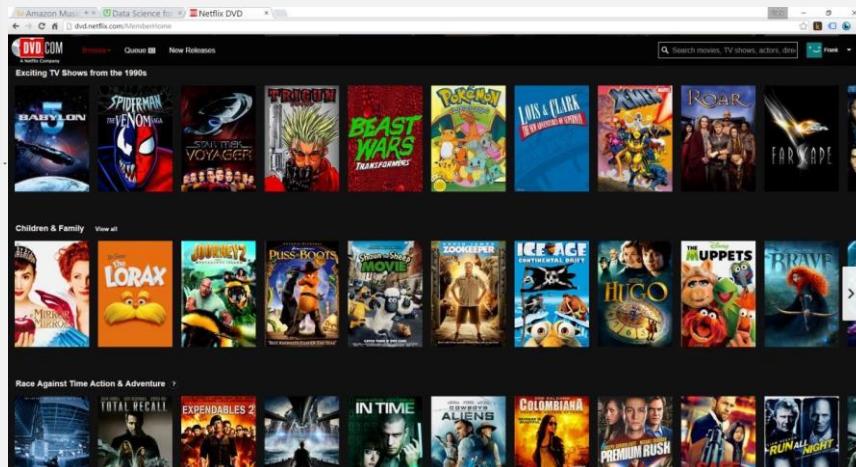
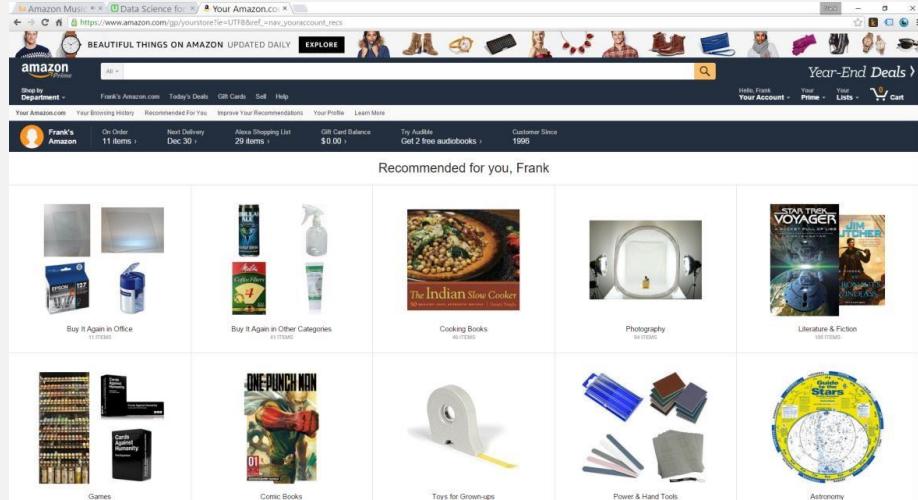
- In practice you'll use something called SVC to classify data using SVM.
- You can use different “kernels” with SVC. Some will work better than others for a given data set.



# Recommender Systems



# What are recommender systems?



# User-Based Collaborative Filtering

- Build a matrix of things each user bought/viewed/rated
- Compute similarity scores between users
- Find users similar to you
- Recommend stuff they bought/viewed/rated that you haven't yet.

# User-Based Collaborative Filtering



# User-Based Collaborative Filtering



# Problems with User-Based CF

- People are fickle; tastes change
- There are usually many more people than things
- People do bad things

# What if we based recommendations on relationships between things instead of people?

- A movie will always be the same movie – it doesn't change
- There are usually fewer things than people (less computation to do)
- Harder to game the system

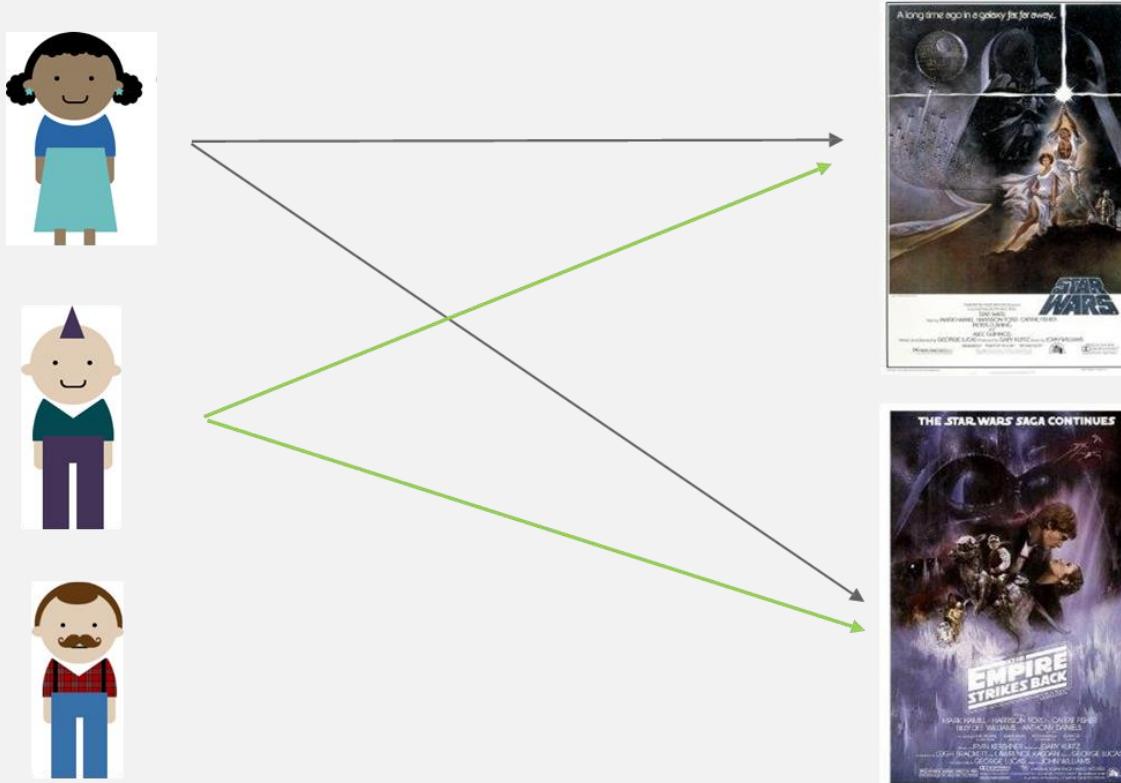
# Item-Based Collaborative Filtering

- Find every pair of movies that were watched by the same person
- Measure the similarity of their ratings across all users who watched both
- Sort by movie, then by similarity strength
- (This is just one way to do it!)

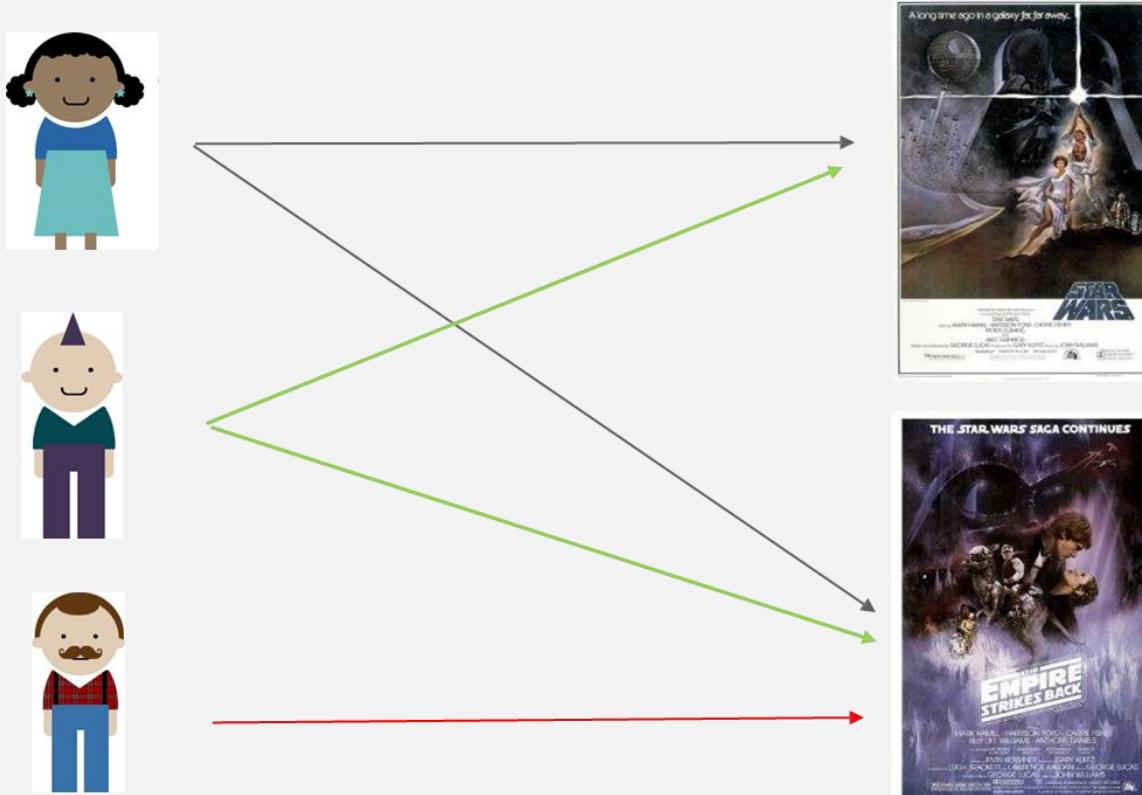
# Item-Based Collaborative Filtering



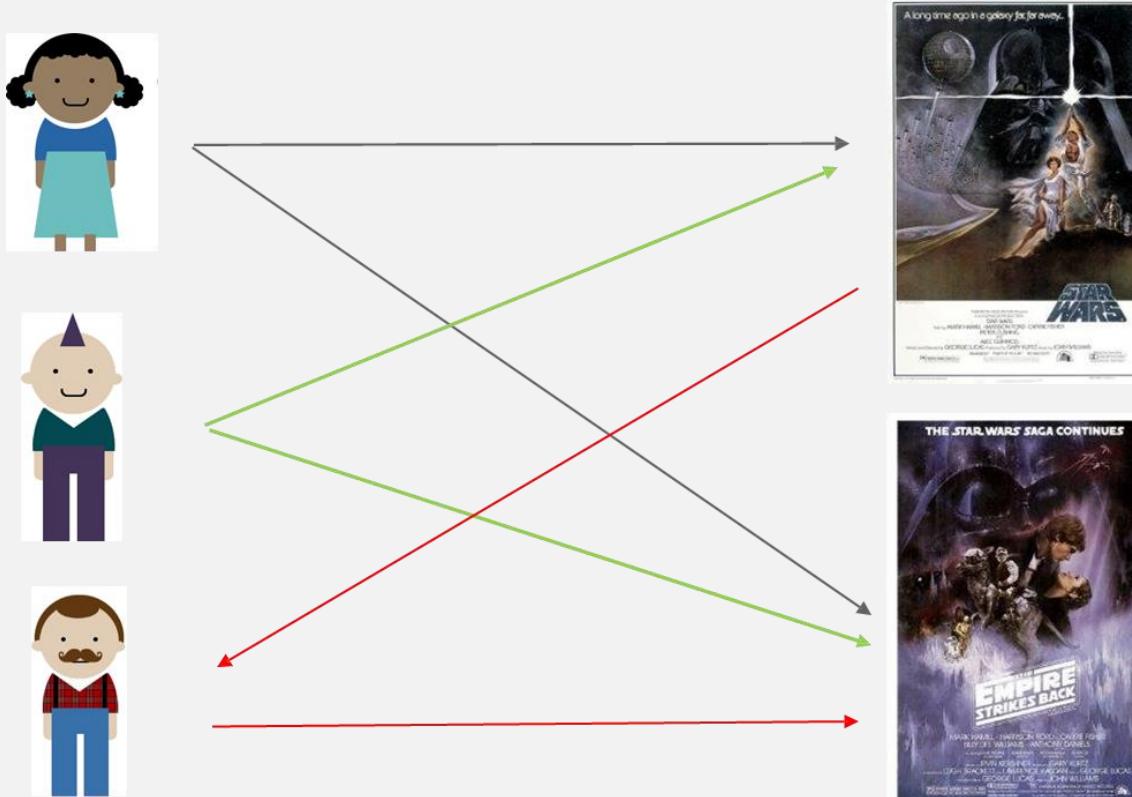
# Item-Based Collaborative Filtering



# Item-Based Collaborative Filtering



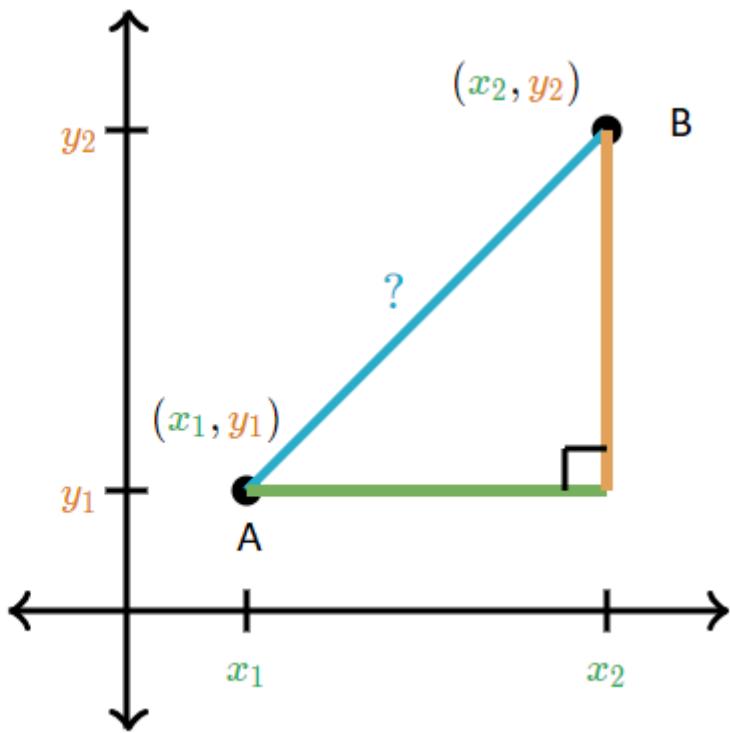
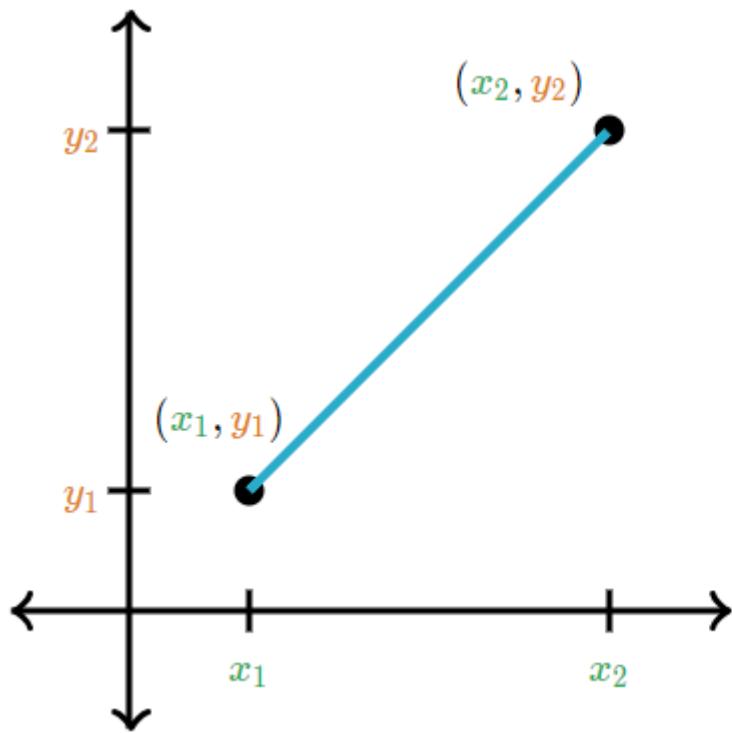
# Item-Based Collaborative Filtering



# K-Nearest Neighbor



# Euclidean



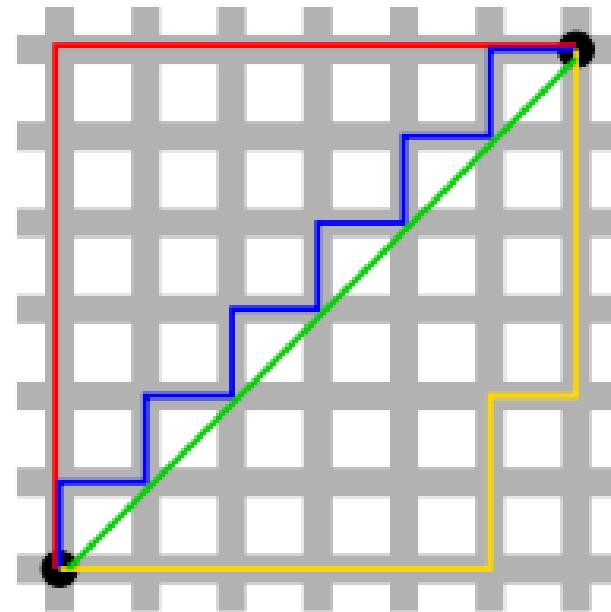
$$? = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

# Minkowski Distance

$$\left( \sum_{i=1}^n |x_i - y_i|^p \right)^{1/p}$$

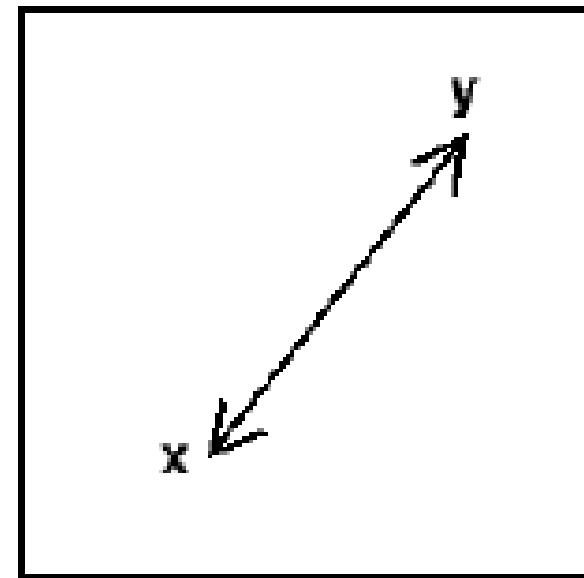
# Minkowski Distance ( $p=1$ ) --> Manhattan Distance

$$\left( \sum_{i=1}^n |x_i - y_i|^p \right)^{1/p}$$



# Minkowski Distance (p=2) --> Euclidean Distance

$$\left( \sum_{i=1}^n |x_i - y_i|^p \right)^{1/p}$$



## Euclidean

# Cosine Distance

$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \|\vec{b}\| \cos \theta$$

$$\cos \theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$$

$$\begin{aligned}\cos 0' &= 1 & \cos 90' &= 0 \\ \cos 180' &= -1\end{aligned}$$

# Mahalanobis Distance

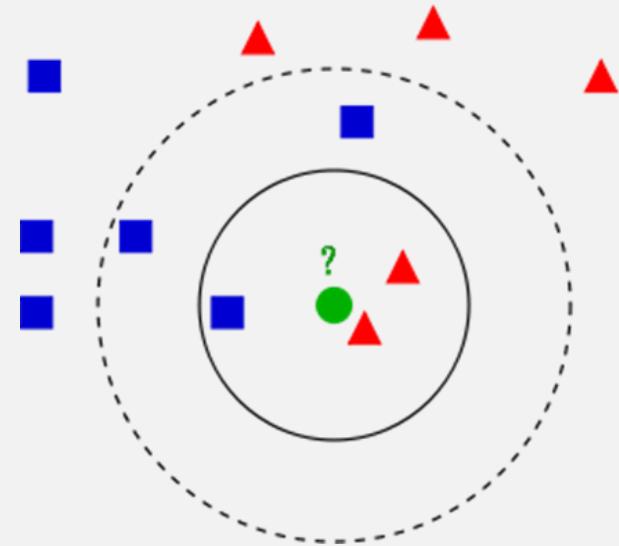
- *The **Mahalanobis distance** is a measure of the distance between a point  $P$  and a distribution  $D$ . The idea of measuring is, how many standard deviations away  $P$  is from the mean of  $D$ .*

$$D_M(\vec{x}) = \sqrt{(\vec{x} - \vec{\mu})^T S^{-1} (\vec{x} - \vec{\mu})}.$$

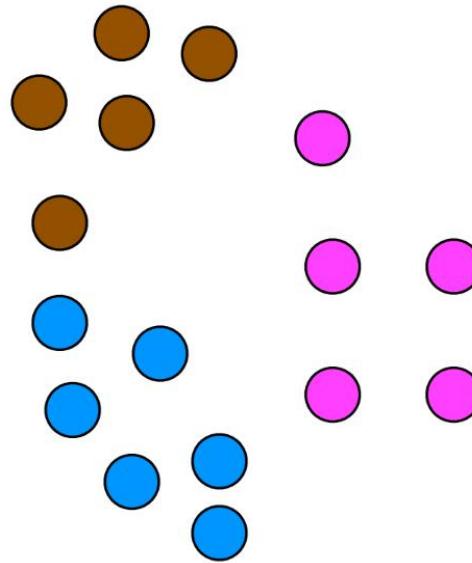
# K-Nearest Neighbor (KNN)

- Used to classify new data points based on “distance” to known data
- Find the K nearest neighbors, based on your distance metric
- Let them all vote on the classification

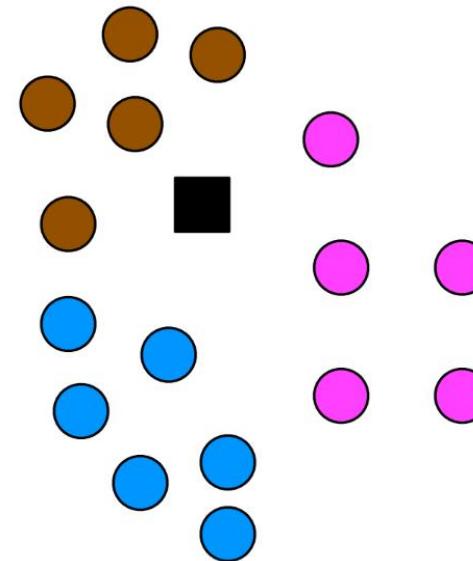
That's it!



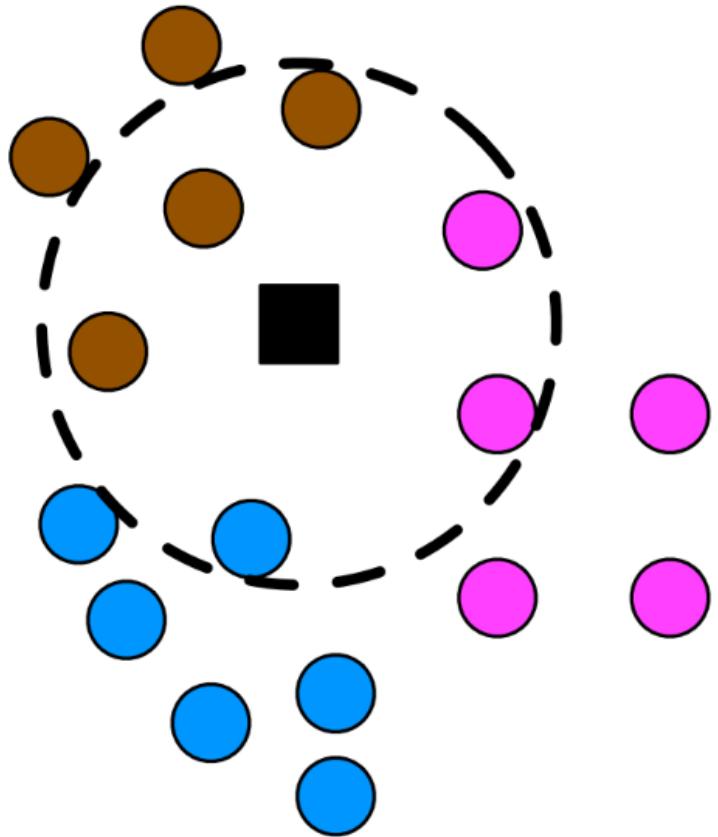
<http://bit.ly/2PCWvPL>



#1



#2



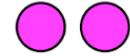
- ● ● 3 from class 1
- 1 from class 2
- ● 2 from class 3



3 from class 1



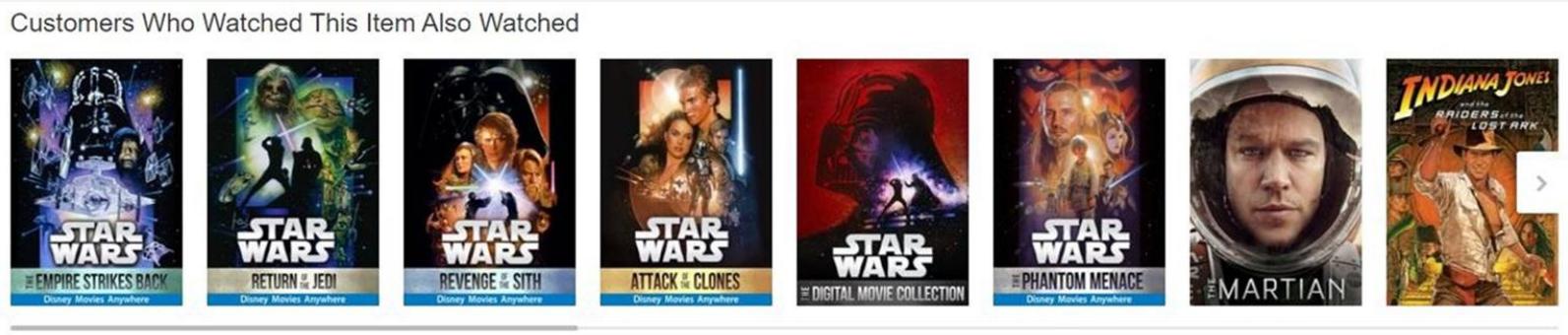
1 from class 2



2 from class 3

# It's Really That Simple

- Although it's one of the simplest machine learning models there is – it still qualifies as “supervised learning”. <http://bit.ly/KNNPCA38>
- But let's do something more complex with it
- Movie similarities just based on metadata!



# Discrete Choice Models



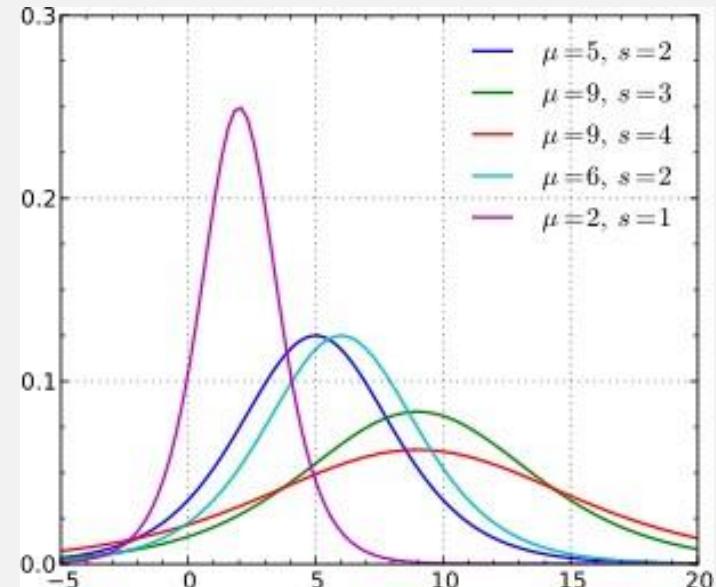
# Discrete Choice Models

- Predict some choice people have between discrete alternatives
  - Do I take the train, bus, or car to work today? (Multinomial choice)
  - Which college will I go to? (Multinomial)
  - Will I cheat on my spouse? (Binary)



# Discrete Choice Models

- Use some sort of regression on the relevant attributes
  - Attributes of the people
  - Variables of the alternatives



# Example

- Will my spouse cheat on me?



# The Curse of Dimensionality



# What is the curse of dimensionality?

- Many problems can be thought of as having a huge number of “dimensions”
- For example, in recommending movies, the ratings vector for each movie may represent a dimension – every movie is its own dimension!
- That makes your head hurt. It’s tough to visualize.

# Remember K-Means Clustering?

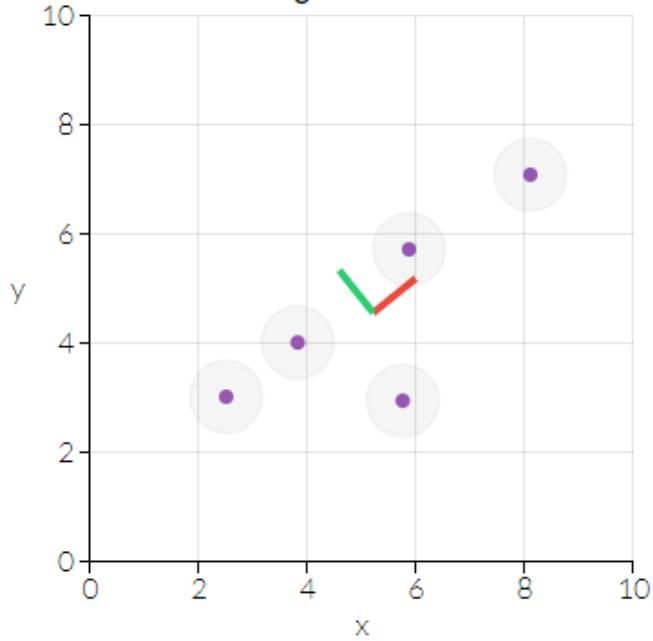
- This is an example of a dimensionality reduction algorithm.
- It reduces data down to K dimensions.



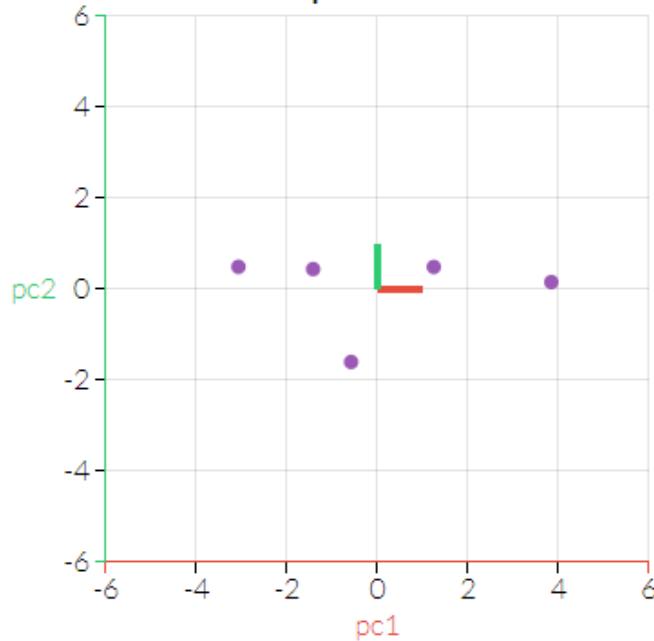
# Another way: Principal Component Analysis (PCA)

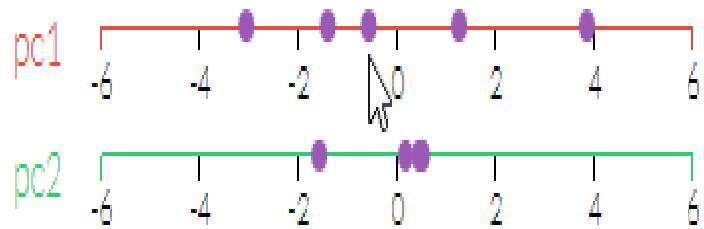
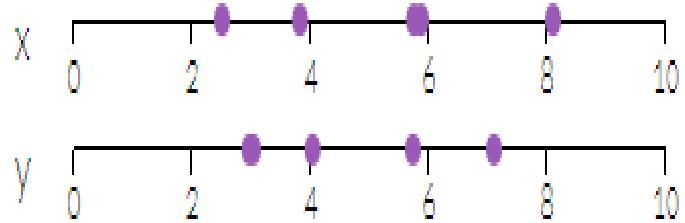
- Involves fancy math – but at a high level:
- Finds “eigenvectors” in the higher dimensional data
  - These define hyperplanes that split the data while preserving the most variance in it
  - The data gets projected onto these hyperplanes, which represent the lower dimensions you want to represent
  - A popular implementation of this is called Singular Value Decomposition (SVD)

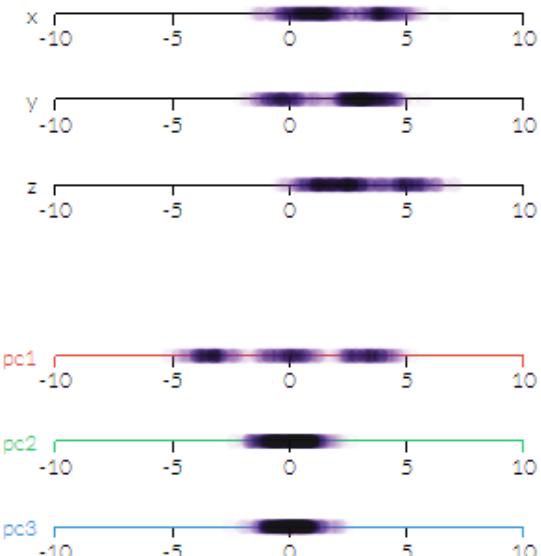
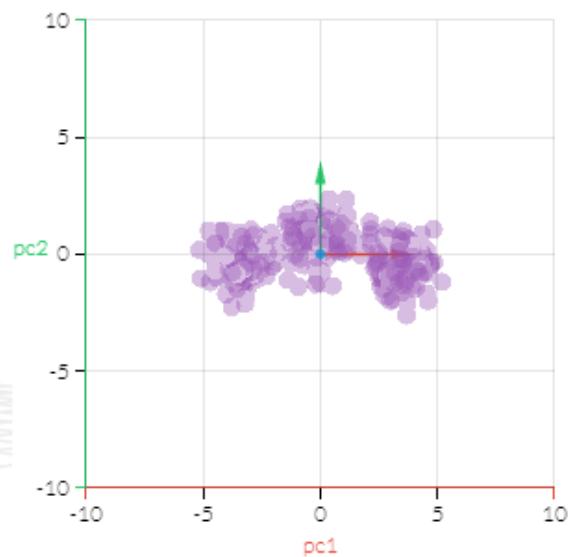
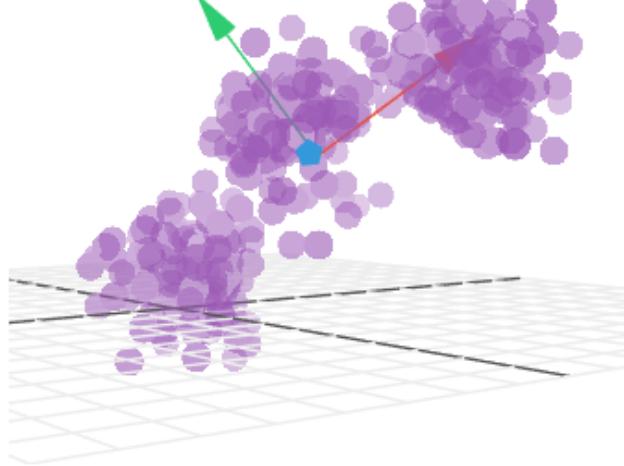
original data set



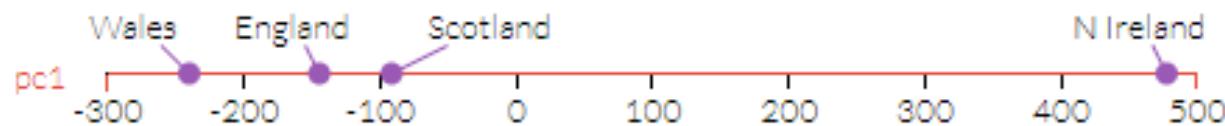
output from PCA







|                    | England | N Ireland | Scotland | Wales |
|--------------------|---------|-----------|----------|-------|
| Alcoholic drinks   | 375     | 135       | 458      | 475   |
| Beverages          | 57      | 47        | 53       | 73    |
| Carcase meat       | 245     | 267       | 242      | 227   |
| Cereals            | 1472    | 1494      | 1462     | 1582  |
| Cheese             | 105     | 66        | 103      | 103   |
| Confectionery      | 54      | 41        | 62       | 64    |
| Fats and oils      | 193     | 209       | 184      | 235   |
| Fish               | 147     | 93        | 122      | 160   |
| Fresh fruit        | 1102    | 674       | 957      | 1137  |
| Fresh potatoes     | 720     | 1033      | 566      | 874   |
| Fresh Veg          | 253     | 143       | 171      | 265   |
| Other meat         | 685     | 586       | 750      | 803   |
| Other Veg          | 488     | 355       | 418      | 570   |
| Processed potatoes | 198     | 187       | 220      | 203   |
| Processed Veg      | 360     | 334       | 337      | 365   |
| Soft drinks        | 1374    | 1506      | 1572     | 1256  |
| Sugars             | 156     | 139       | 147      | 175   |



# Example: Visualizing 4-D Iris Flower Data

- The “Iris dataset” comes with scikit-learn
- An Iris flower has petals and sepals (the lower, supportive part of the flower.)



# ETL and ELT



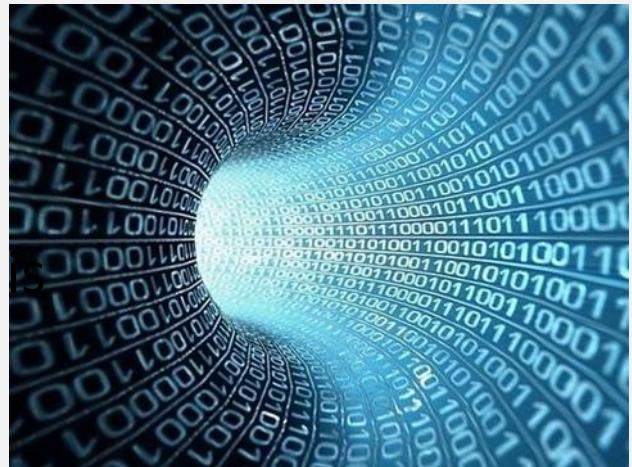
# What is Data Warehousing?

- A large, centralized database that contains information from many sources
- Often used for business analysis in large corporations or organizations
- Queried via SQL or tools (i.e. Tableau)



# ETL: Extract, Transform, Load

- ETL and ELT refer to how data gets into a data warehouse.
- Traditionally, the flow was Extract, Transform, Load:
  - Raw data from operational systems is first periodically *extracted*



# ELT: Extract, Load, Transform

- Today, a huge Oracle instance isn't the only choice for a large data warehouse
- Things like Hive let you host massive databases on a Hadoop cluster
- Or, you might store it in a large, distributed NoSQL data store
  - ...and query it using things like Spark or MapReduce



# Lots more to explore

- Data warehousing is a discipline in itself, too big to cover here
- We will cover Spark in more depth later in this course.

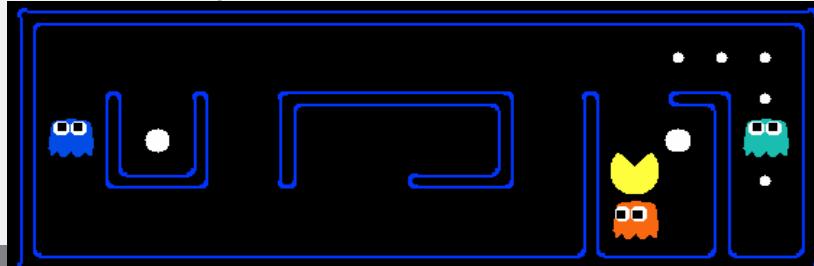


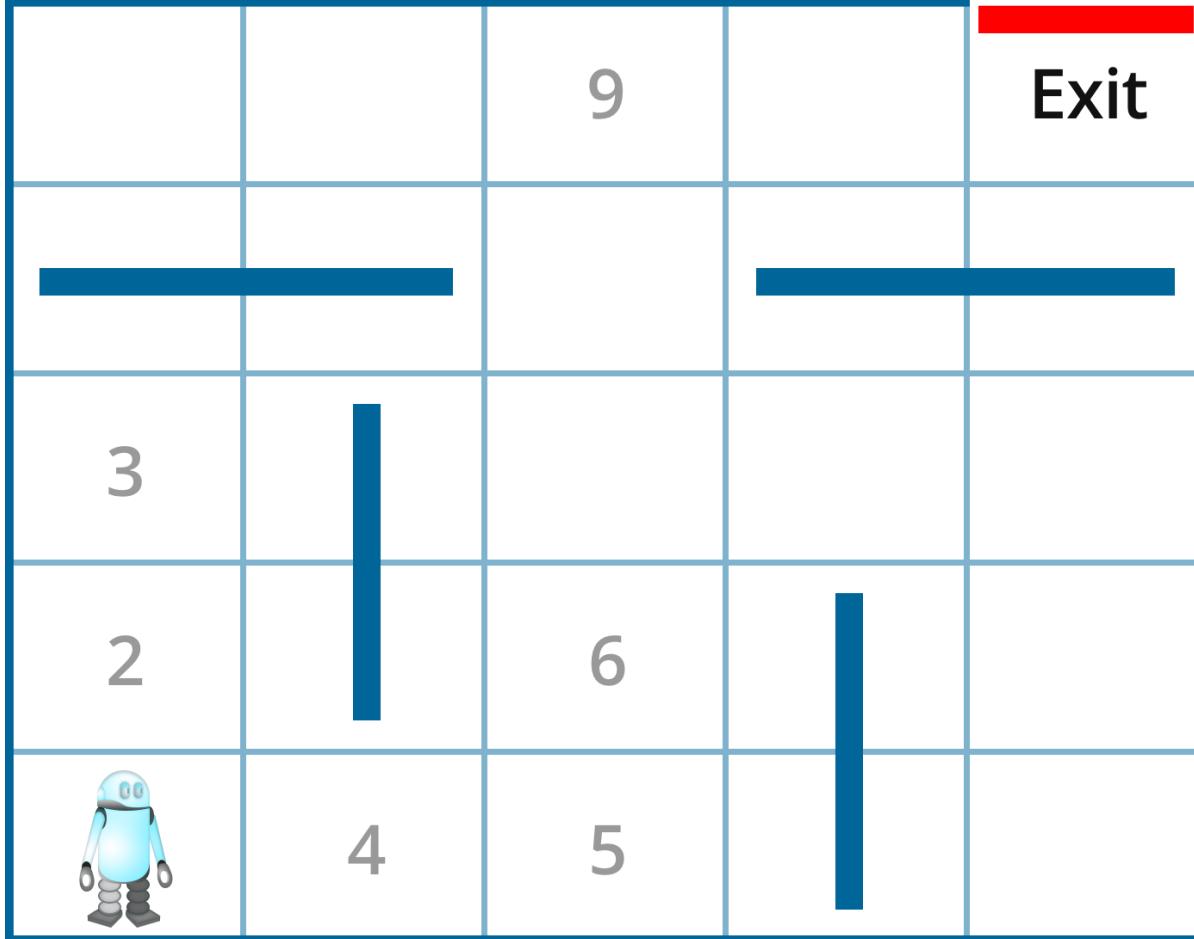
# Reinforcement Learning



# Reinforcement Learning

- You have some sort of agent that “explores” some space
- As it goes, it learns the value of different state changes in different conditions
- Those values inform subsequent behavior of the agent
- Examples: Pac-Man, Cat & Mouse game
- Yields fast on-line performance once the space has been explored



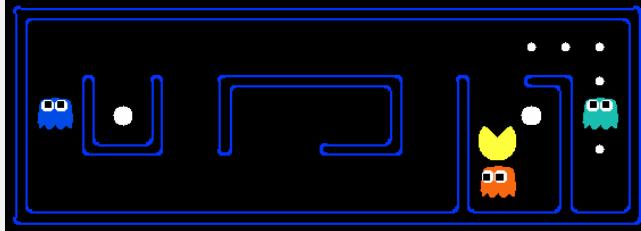


# Q-Learning

- A specific implementation of reinforcement learning
- You have:
  - A set of environmental states  $s$
  - A set of possible actions in those states  $a$
  - A value of each state/action Q
- Start off with Q values of 0
- Explore the space

# Q-Learning

- What are some state/actions here?
  - Pac-man has a wall to the West
  - Pac-man dies if he moves one step South
  - Pac-man just continues to live if going North or East
- You can “look ahead” more than one step by using a discount factor when computing Q (here  $s$  is previous state,  $s'$  is current state)
  - $Q(s,a) += \text{discount} * (\text{reward}(s,a) + \max(Q(s')) - Q(s,a))$



# The exploration problem

- How do we efficiently explore all of the possible states?
  - Simple approach: always choose the action for a given state with the highest Q. If there's a tie, choose at random
    - But that's really inefficient, and you might miss a lot of paths that way

# Fancy Words

- Markov Decision Process
  - From Wikipedia: **Markov decision processes (MDPs)** provide a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision maker.
  - Sound familiar? MDP's are just a way to describe what we just did using mathematical notation.
  - States are still described as  $s$  and  $s'$
  - State transition functions are described as  $P_a(s, s')$   
Our "Q" values are described as a reward function  $R_a(s, s')$

# More Fancy Words

- Dynamic Programming
  - From Wikipedia: **dynamic programming** is a method for solving a complex problem by breaking it down into a collection of simpler subproblems, solving each of those subproblems just once, and storing their solutions - ideally, using a memory-based data structure.

# So to recap

- You can make an intelligent Pac-Man in a few steps:
  - Have it semi-randomly explore different choices of movement (actions) given different conditions (states)
  - Keep track of the reward or penalty associated with each choice for a given state/action ( $Q$ )
  - Use those stored  $Q$  values to inform its future choices



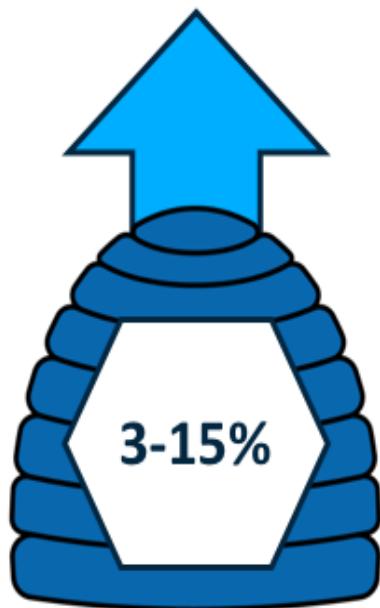
# Implementing Reinforcement Learning

- Python Markov Decision Process Toolbox:
  - <http://pymdptoolbox.readthedocs.org/en/latest/api/mdp.html>
- Cat & Mouse Example:
  - <https://github.com/studywolf/blog/tree/master/RL/Cat%20vs%20Mouse%20exploration>
- Pac-Man Example:
  - <https://inst.eecs.berkeley.edu/~cs188/sp12/projects/reinforcement/reinforcement.html>

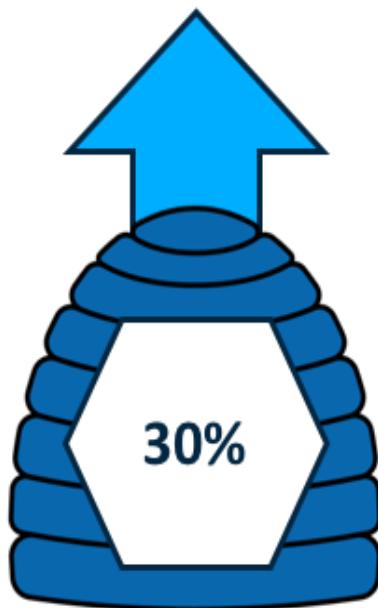
# The Data Science Process



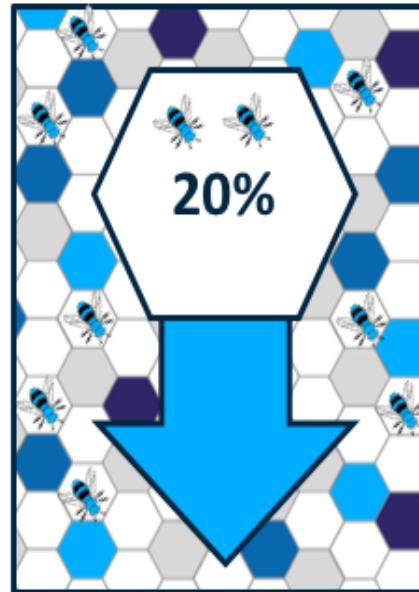
# Artificial Intelligence Adopters



Profit Advantage

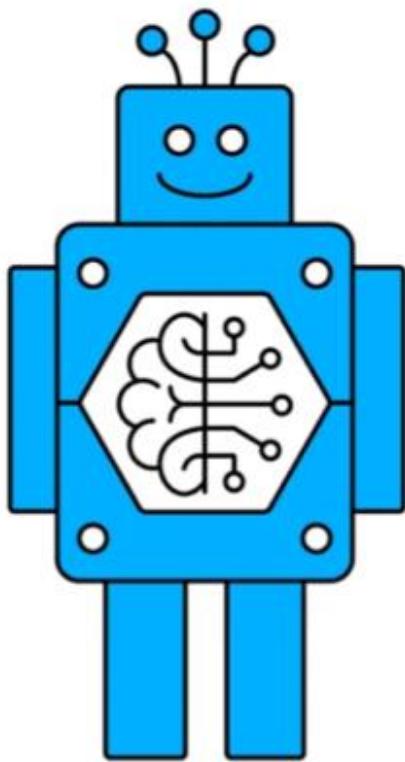


Overall Revenue

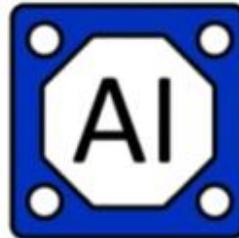


Production

# The Data Science Landscape



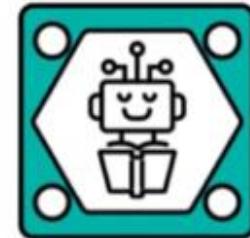
AI  
Planning



Symbolic  
Logic  
Expert System



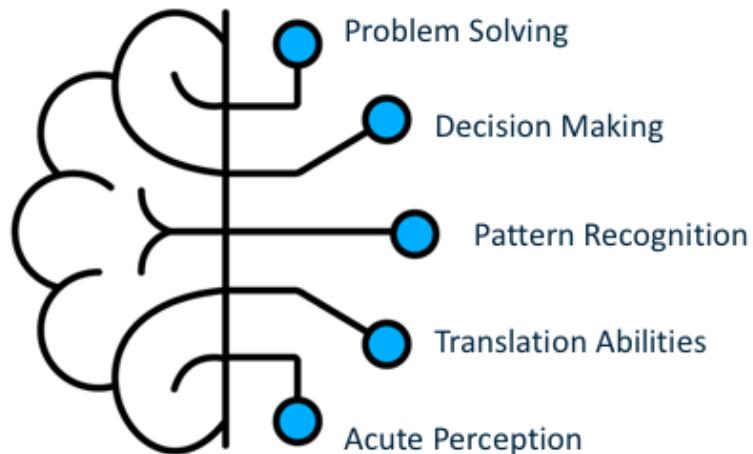
Machine  
Learning



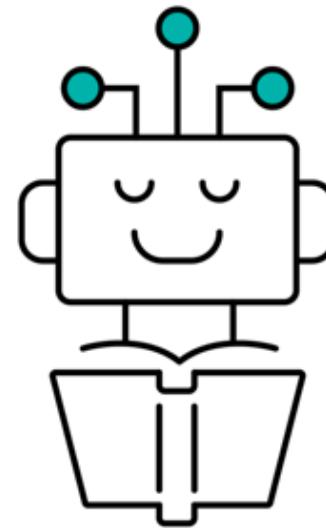


## Review: AI vs. ML

### Artificial Intelligence



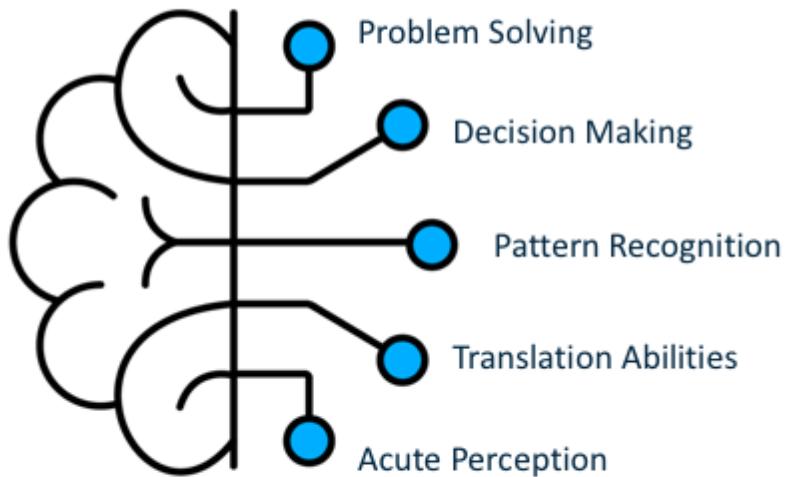
### Machine Learning





# Review: AI vs. ML

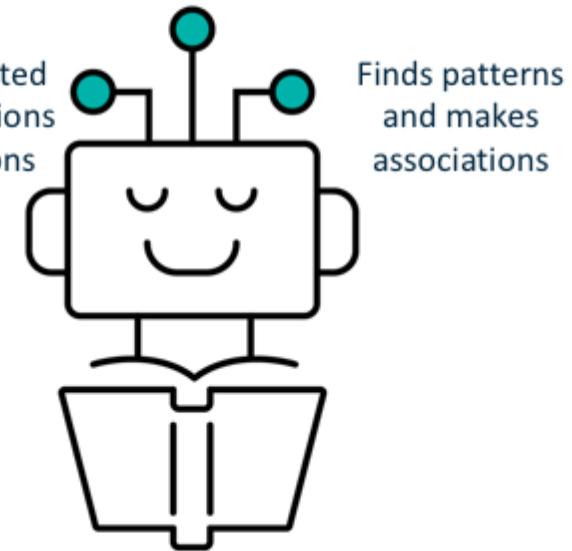
## Artificial Intelligence



## Machine Learning

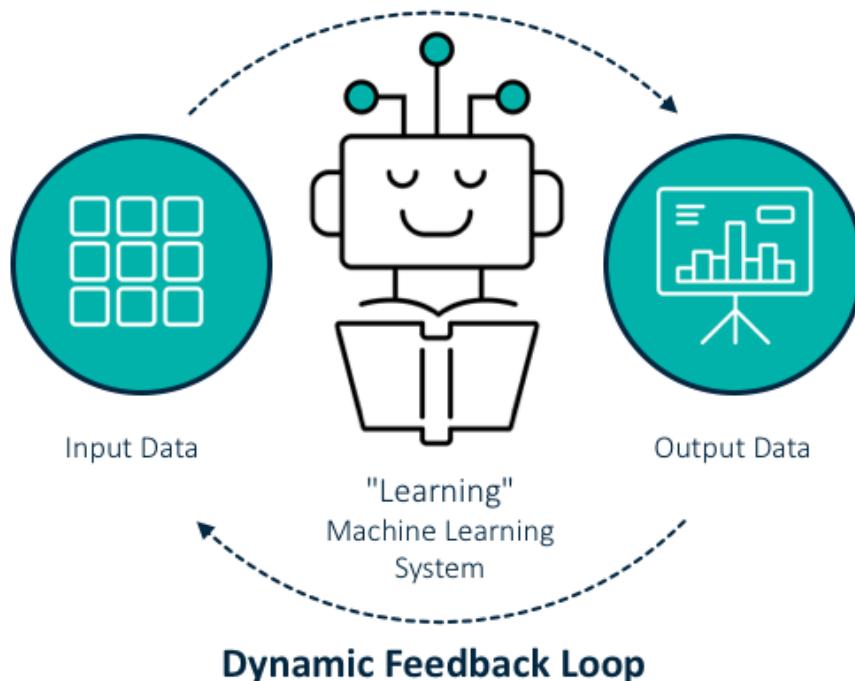
Taught to learn from data and make decisions

Makes calculated recommendations and predictions

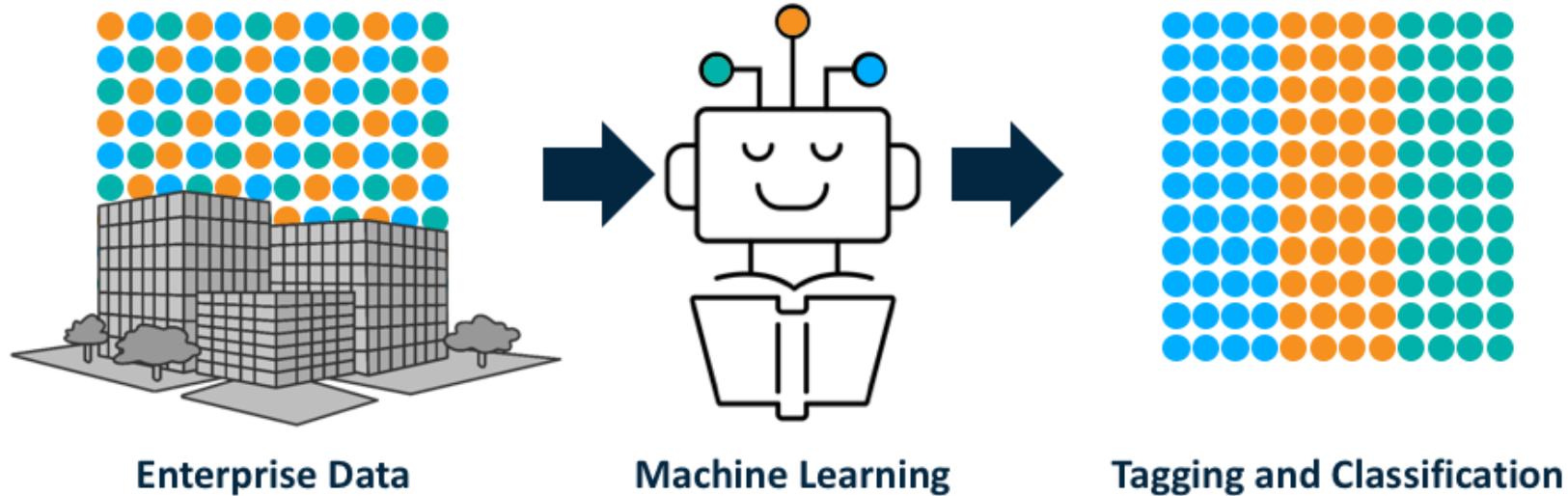




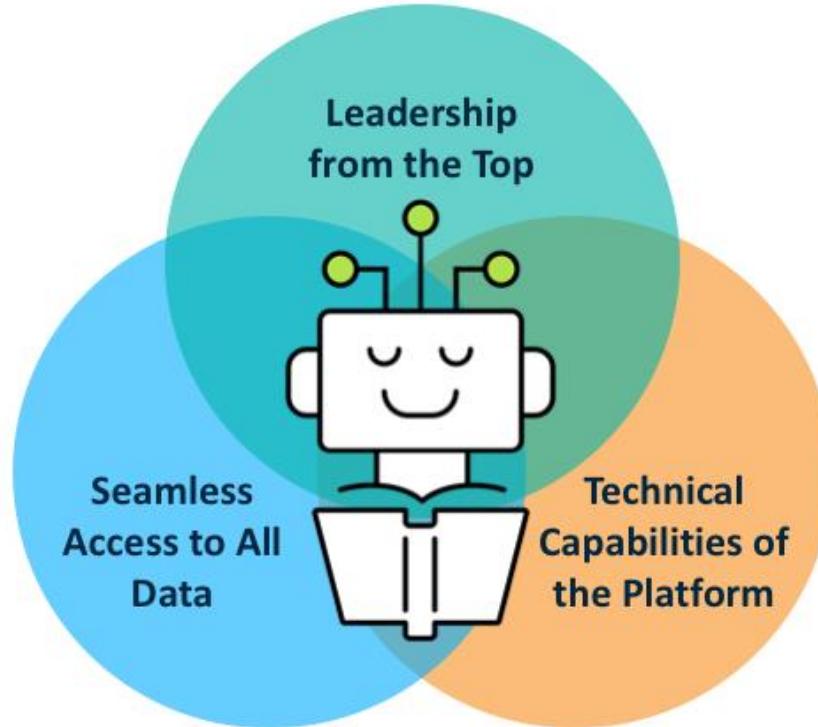
## | Review: AI vs. ML



# Machine Learning Today



## A Strong Foundation



## Data Management Logistics

**90% of Machine Learning  
effort** is all about  
**Data Logistics**

If not done well, it can easily cause you to fail.

# Project Plan Workflow

| Part I (Preparation)                             |                         |  | Part II (Implementation)         |                          |                               |
|--|-------------------------|--|----------------------------------|--------------------------|-------------------------------|
| Plan   | Prepare                 | Features                                   | Framework                        | Model                    | Production                    |
| 1  | 2                       | 3  | 4                                | 5                        | 6                             |
| Identify business need and create a project plan | Manage and prepare data | Select and engineer features for the model | Select algorithms and frameworks | Train and validate model | Implement and monitor project |
|  |                         |  |                                  |                          |                               |

# 1

## Identify Business Need

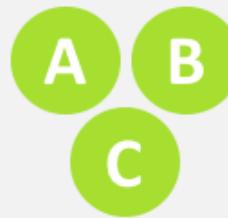
**How will the business derive value?**

- Define project goal
- Break down broad goals to make them specific
- Determines success criteria

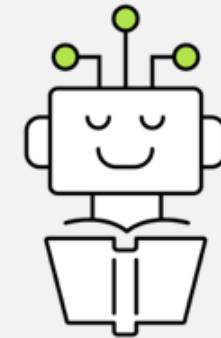
Determine the following:



Broad Goal



Measurable Outcomes



Success Criteria

# 1 Examples by Industry



Healthcare



Manufacturing



Finance



Retail



## 1 Examples by Industry - Healthcare



Improve patient care

Improve accuracy of medical diagnoses

Improve accuracy detection  
on skin disorders

Improve accuracy detection using  
image recognition on skin lesions  
by X%

## 1 Examples by Industry - Manufacturing



Improve business efficiency and reduce costs

Detect and prevent unexpected machine line maintenance

Install new IoT sensors to monitor performance, identifying potential instances of performance degradation

Reduce machine line downtime by X%

## 1 Examples by Industry - Finance



Reduce operational costs and improve customer experience

Reduce response times of reported fraud behavior with program automation

Create ML application that automates system communications to report anomalous behavior, then block/reverse charges upon verification

Reduce total processing time by X%

## 1 Examples by Industry – Retail



Increase sales and customer engagement

Offer product recommendations and extend cross-promotional sales

Create a robust recommendation engine

Increase recommended product checkout rates by X%

## 1 Example Use Case – AllPets Retail Supplier



# 1

## Example Use Case – Recommendation Engines

### Content-Based Engines

- Item attribute data

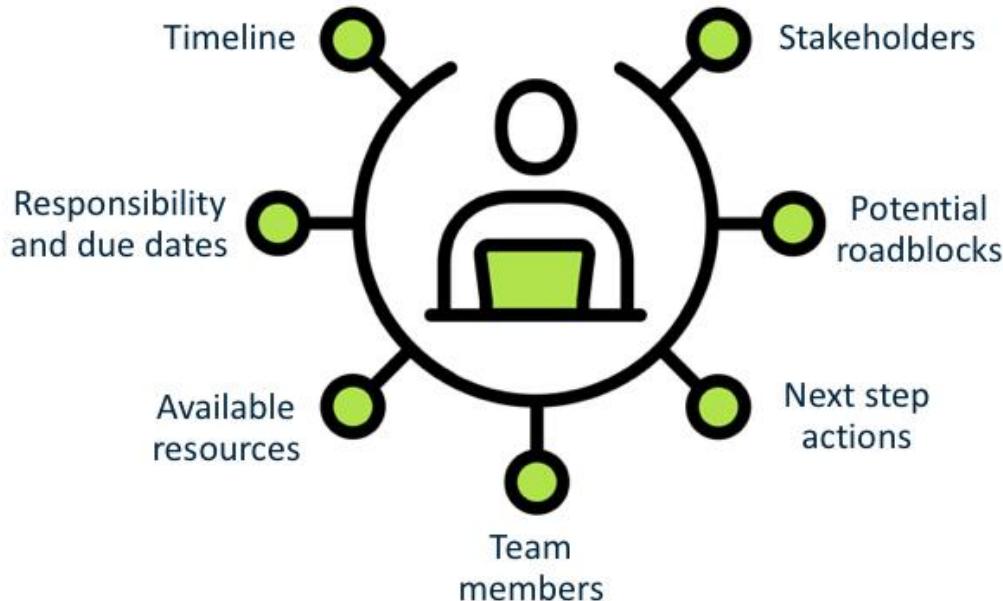
### Collaborative Filtering Engines

- User interactions



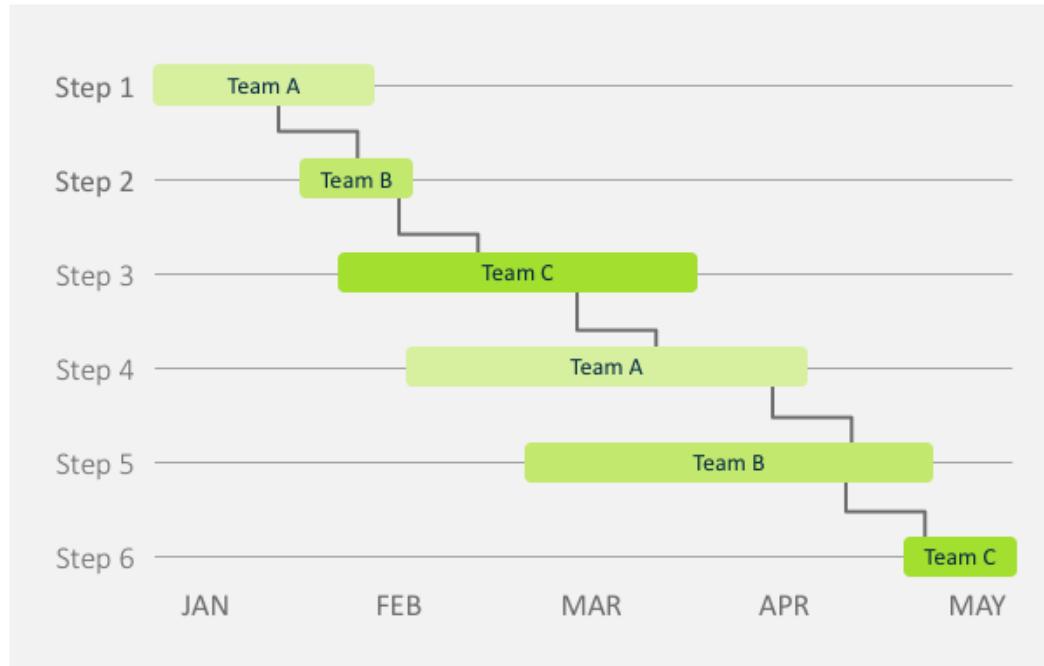
# 1

## Establish a Solid Project Plan



# 1 Establish a Solid Project Plan

- Pre-defined launch tasks
- Incorporate all post-launch requirements



# 1

## Machine Learning Planning Specifics



What tools will produce the best **results**?

What methods will **deliver** the most value?

**Allocate enough time for:**

- Selection of algorithms and frameworks
- Feature engineering
- Data preparation and transformation

# 1 Other ML Considerations



## Privacy and Regulations

- What are the privacy concerns?
- Do you need to document reasoning behind decisions?



## Cold Start Projects

- Starting without the required data
- Can buy or simulate data



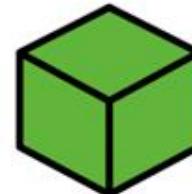
## Surrogate or Approximation Models

- Exploration with data you might want to model
- Expected outcomes are unknown

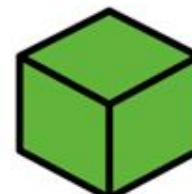
# Project Plan Workflow: Step 2

| Part I (Preparation)  |   |   |
|---|---|---|
| Plan  | Prepare   | Features  |
| 1   | 2   | 3   |
| Identify business need and create a project plan                                  | Manage and prepare data   | Select and engineer features for the model  |
|  |  |  |

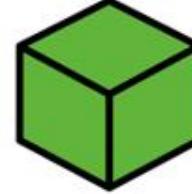
## Main Areas of Data Management



- Collect
- Explore
- Assess

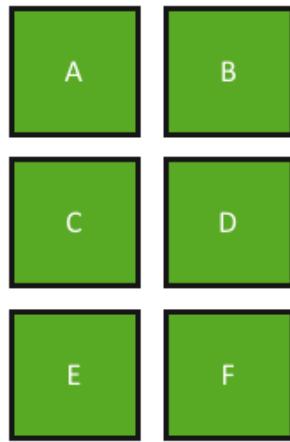


- Pre-process
- Clean
- Format

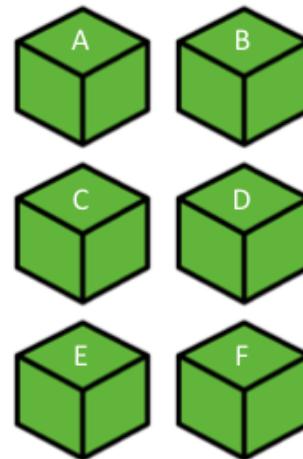


- Transform

## 2 Data Management – Collect Data



Identify datasets  
from various sources



Compile and retrieve  
datasets



Organize datasets

## 2

## Data Management – Explore and Assess Data

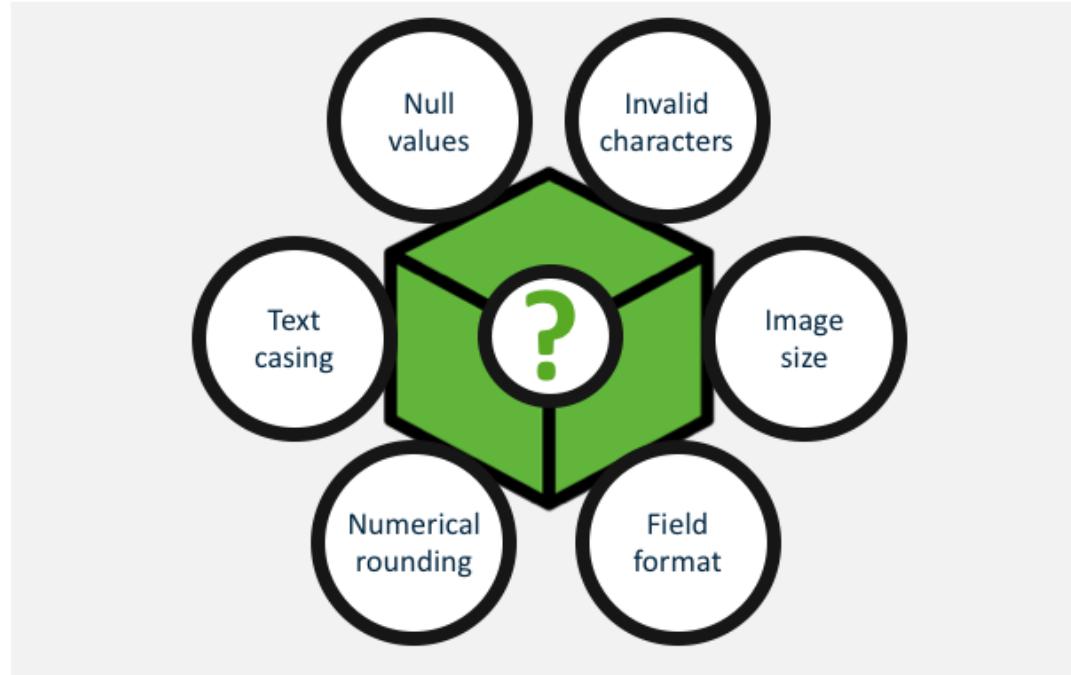


### Considerations:

- Unstructured vs. Structured data formats
- Missing values, invalid characters % ! ^ ;\*\_
- Numerical rounding or text casing
- Image/video resolution, rotation, sizes

## 2 Data Preparation – Pre-Process Data

- Addresses questions raised during exploration phase
- Integrity of original data should be protected
- Never overwrite original datasets
- Use snapshots, mirrors, or version control to protect raw data



## 2 Data Pre-processing – Example Use Case

| Client Name | Transaction ID | Purchase Date | Purchase TimeStamp | Address Field 1           | State      | Zip   | Client Email  |
|-------------|----------------|---------------|--------------------|---------------------------|------------|-------|---------------|
| Sara Kendy  | 10294..        | 10/24/18      | 12:44:32           | 1011 First A.             | California | 94402 | smkend@y...   |
| sara kendy  | 10283..        | 06/4/17       | 2:50:42%00         | 1011 1 <sup>st</sup> Av.. | CA         | 94402 | smkend@y...   |
| Chip Ray    | 10291..        | 2018/1/4      | 04:03:33>          | 22 Beverly..              | ND         | -     | chips@gma..   |
| D. Little   | 10290..        | 2/22/17       | 12:54:02           | 848 Gamin..               | CO         | -     | lild@hotmai.. |
| Jon Allen   | 10292..        | 4/5/2018      | 08:06:55           | 447 Pieland.              | CAL        | 90523 | jonizkewl@h.. |
| Jon allen   | 10293..        | 09/16/18      | ^7:^3:20           | 7 Butan Ro..              | MN         | 40528 | jonizkewl@h.. |

## 2 Data Pre-processing – Example Use Case

| Client Name | Transaction ID | Purchase Date | Purchase TimeStamp | Address Field 1           | State | Zip   | Client Email  |
|-------------|----------------|---------------|--------------------|---------------------------|-------|-------|---------------|
| sara kendy  | 10294..        | 10/24/18      | 12:44:32           | 1011 First A.             | CA    | 94402 | smkend@y...   |
| sara kendy  | 10283..        | 06/04/17      | 02:50:42           | 1011 1 <sup>st</sup> Av.. | CA    | 94402 | smkend@y...   |
| chip ray    | 10291..        | 01/04/18      | 04:03:33           | 22 Beverly..              | ND    | 0     | chips@gma..   |
| d little    | 10290..        | 02/22/17      | 12:54:02           | 848 Gamin..               | CO    | 0     | lild@hotmai.. |
| jon allen   | 10292..        | 04/05/18      | 08:06:55           | 447 Pieland.              | CA    | 90523 | jonizkewl@h.. |
| jon allen   | 10293..        | 09/16/18      | 07:03:20           | 7 Butan Ro..              | MN    | 40528 | jonizkewl@h.. |

## 2 Data Transformations: Unstructured to Structured Data

### Data Type: Text

Transformation: Vectorizing



### Data Type: Image

Transformation: Labeling



### Data Type: Audio

Transformation: Signal Processing



### Data Type: Video

Transformation: Labeling

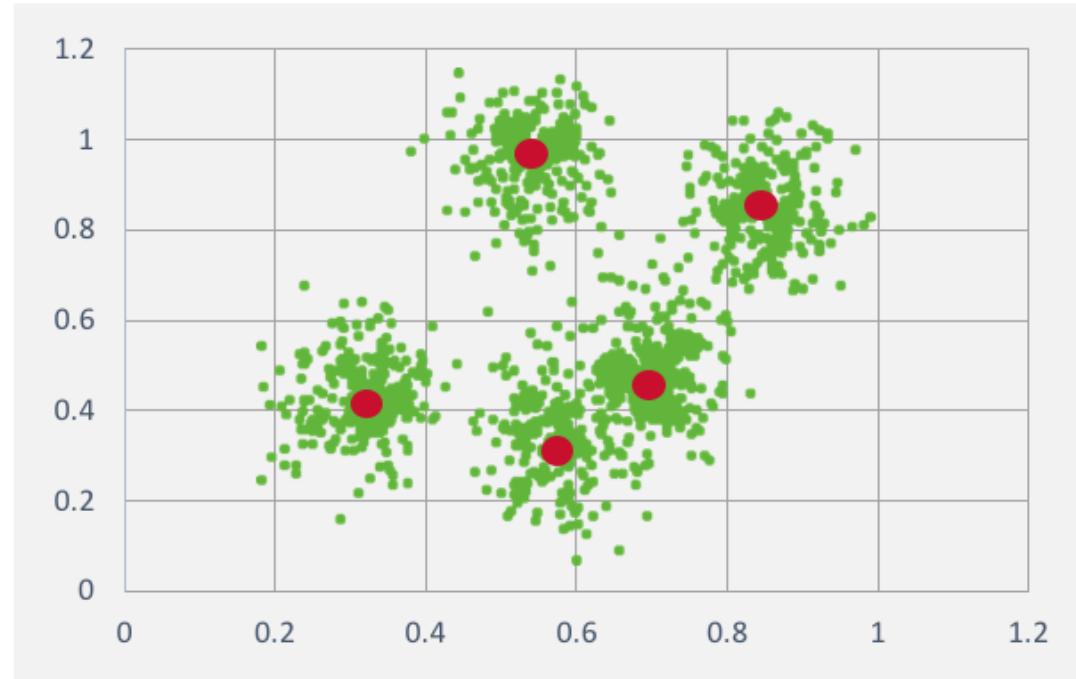


## 2

# Data Transformations: Dimensionality Reduction

## Dimensionality Reduction Algorithms:

- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Singular value Decomposition (SVD)

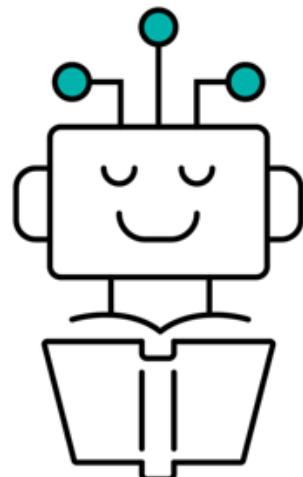


# Project Plan Workflow: Step 3

| Part I (Preparation)  |   |   |
|---|---|---|
| Plan  | Prepare   | Features  |
| 1   | 2   | 3   |
| Identify business need and create a project plan                                  | Manage and prepare data   | Select and engineer features for the model  |
|  |  |  |

## 3 Feature Selection

Select the features from your data that will provide the most value to your project. The model may work perfectly, but if the wrong features were selected for training, it won't produce useful results.



Consider the intention of your original project goal.

Examples of goals that may require very different features for selection:

- Increasing cross-sells
- Increasing click rates
- Increasing page visit durations

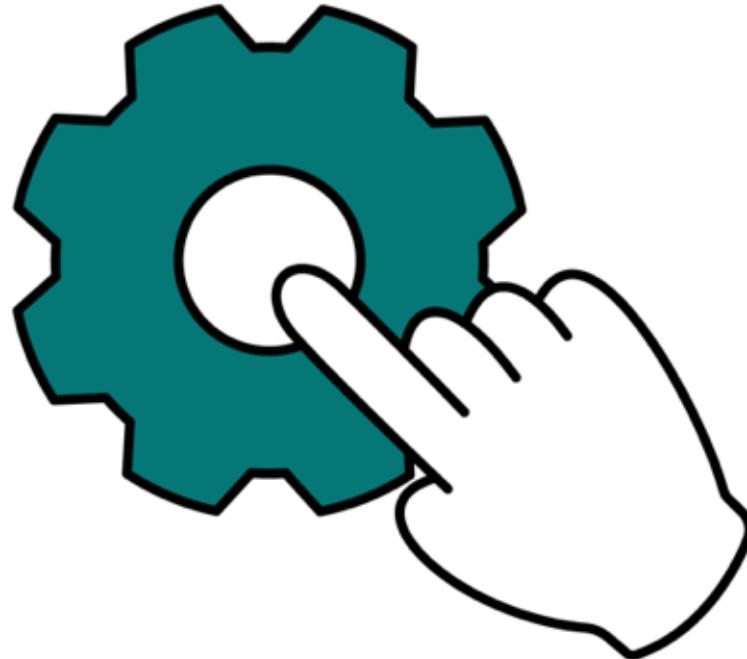
## 3 Feature Selection

| Feature Selection Tips      | Examples   |
|-----------------------------|--|
| Useful vs. useless features | Do you need to keep all physical characteristic data, or just some? Do you need all user data like shipping information? |
| Avoid redundant features    | Measurement fields showing in multiple formats:<br>in/cm, lbs/kg   |
| Keep it simple              | Age vs. date of birth fields, latitude/longitude coordinates<br>vs. simplified address or zip code                       |

### 3 Feature Selection

| Client Name | Transaction ID | Purchase Date | Purchase TimeStamp | Address Field 1           | State | Zip   | Client Email   | Product(s)         |
|-------------|----------------|---------------|--------------------|---------------------------|-------|-------|----------------|--------------------|
| sara kendy  | 10294..        | 10/24/18      | 12:44:32           | 1011 First A.             | CA    | 94402 | smkend@y...    | Dog XL Bed Com..   |
| sara kendy  | 10283..        | 06/04/17      | 02:50:42           | 1011 1 <sup>st</sup> Av.. | CA    | 94402 | smkend@y...    | 10Pk Bone Flavo..  |
| chip ray    | 10291..        | 01/04/18      | 04:03:33           | 22 Beverly..              | ND    | 0     | chips@gma..    | Premium Wate..     |
| d little    | 10290..        | 02/22/17      | 12:54:02           | 848 Gamin..               | CO    | 0     | lild@hotmail.. | AquaSafe Fish B..  |
| jon allen   | 10292..        | 04/05/18      | 08:06:55           | 447 Pieland.              | CAL   | 40528 | jonizkewl@h..  | PurrSnickety toy   |
| jon allen   | 10293..        | 09/16/18      | 07:03:20           | 7 Butan Ro..              | MN    | 40528 | jonizkewl@h..  | Cat XL Bee Costu.. |

### 3 Feature Selection - Example Use Case

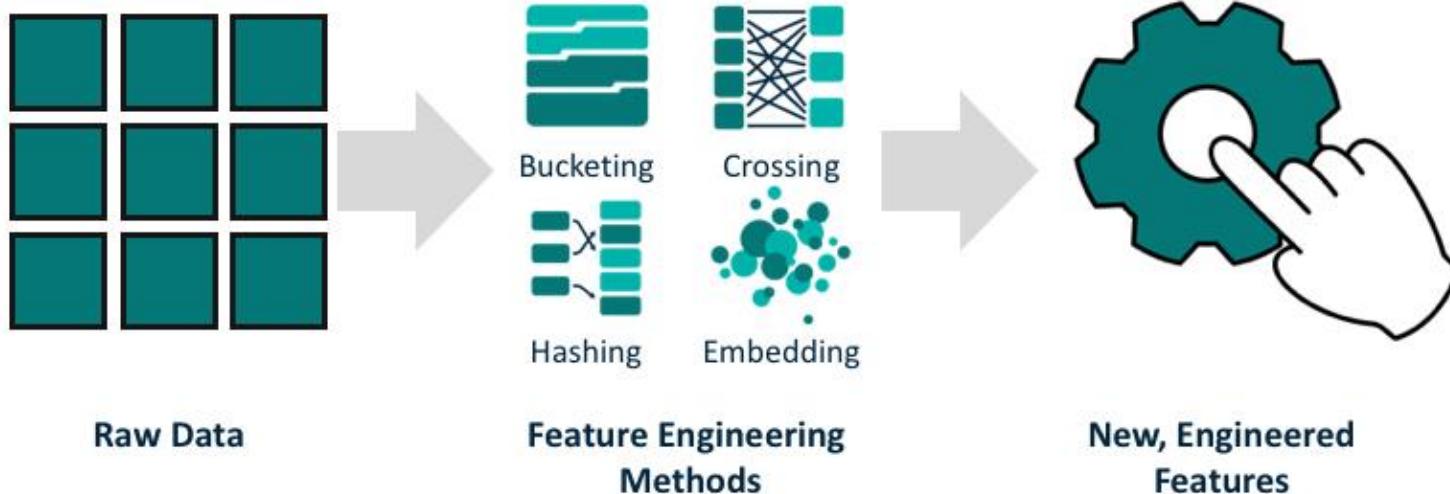


#### Potential Features to Train Recommendation Model

- Unique users and items
- User purchase history
- Product run rates
- Any available ratings or product reviews by user
- User viewing history (web logs)

### 3 Feature Engineering

Feature engineering is the transformation of raw data into inputs for your algorithm, along with creating composite features.



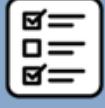
 | Full Project Plan Workflow

| Part I (Preparation)                             |                         |  | Part II (Implementation)         |                          |                               |
|--|-------------------------|--|----------------------------------|--------------------------|-------------------------------|
| Plan   | Prepare                 | Features                                   | Framework                        | Model                    | Production                    |
| 1  | 2                       | 3  | 4                                | 5                        | 6                             |
| Identify business need and create a project plan | Manage and prepare data | Select and engineer features for the model | Select algorithms and frameworks | Train and validate model | Implement and monitor project |
|  |                         |  |                                  |                          |                               |

# Project Plan Workflow: Step 4

| Part II (Implementation)  |   |   |
|---|---|---|
| Framework   | Model   | Production  |
| <b>4</b>  | <b>5</b>  | <b>6</b>  |
| Select algorithms and frameworks  | Train and validate model  | Implement and monitor project   |
|  |  |  |

# Project Plan Workflow

| Part II (Implementation)  |   |   |  |
|---|---|---|--|
| Framework   | Model   | Framework   | Production   |
| 4   | 5   | 4   | 6  |
| Select algorithms and frameworks  | Train and validate model  | Revisit algorithms and frameworks   | Implement and monitor project  |
|  |  |  |  |

- Not always linear
- Steps may intermingle or become iterative

## 4 Algorithm and Framework Selection

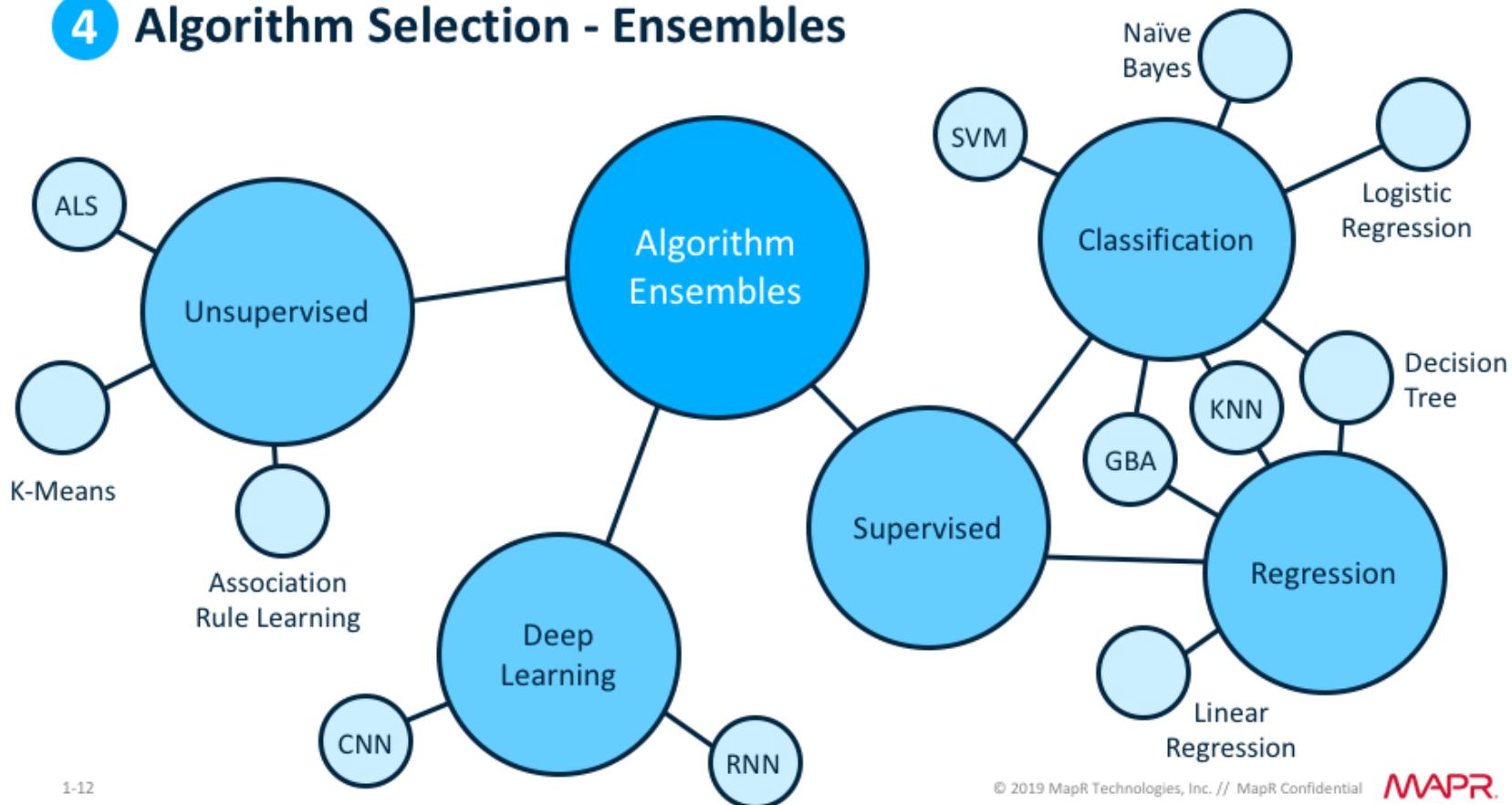
| Learning Method | Algorithm Type            | Algorithm Name                     |
|-----------------|---------------------------|------------------------------------|
| Supervised      | Classification            | Naïve Bayes                        |
| Supervised      | Classification            | Logistic Regression                |
| Supervised      | Classification            | Support Vector Machines (SVM)      |
| Supervised      | Regression                | Linear Regression                  |
| Supervised      | Classification/Regression | Decision Trees/Random Forest       |
| Supervised      | Classification/Regression | K-Nearest Neighbor (KNN)           |
| Supervised      | Classification/Regression | Gradient Boosting Algorithms (GBA) |
| Unsupervised    | Unsupervised              | K-Means: Cluster Analysis          |
| Unsupervised    | Unsupervised              | Association Rule Learning          |
| Unsupervised    | Unsupervised              | Alternating Least Squares (ALS)    |

## 4

## Algorithm and Framework Selection

| Algorithm Type | Learning Method         | Algorithm Name                     |
|----------------|-------------------------|------------------------------------|
| Deep Learning  | Supervised/Unsupervised | Recurrent Neural Network (RNN)     |
| Deep Learning  | Supervised/Unsupervised | Convolutional Neural Network (CNN) |

## 4 Algorithm Selection - Ensembles



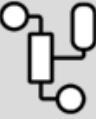
## 4

## Framework Selection

- Determine which ML frameworks to use
- Tools and frameworks should allow for flexibility in your pipeline
- This is only a subset of available libraries
- Will continue to change and grow as the field evolves



# Project Plan Workflow: Step 5

| Part II (Implementation)  |   |   |
|---|---|---|
| Framework   | Model   | Production  |
| <b>4</b>  | <b>5</b>  | <b>6</b>  |
| Select algorithms and frameworks  | Train and validate model  | Implement and monitor project   |
|  |  |  |

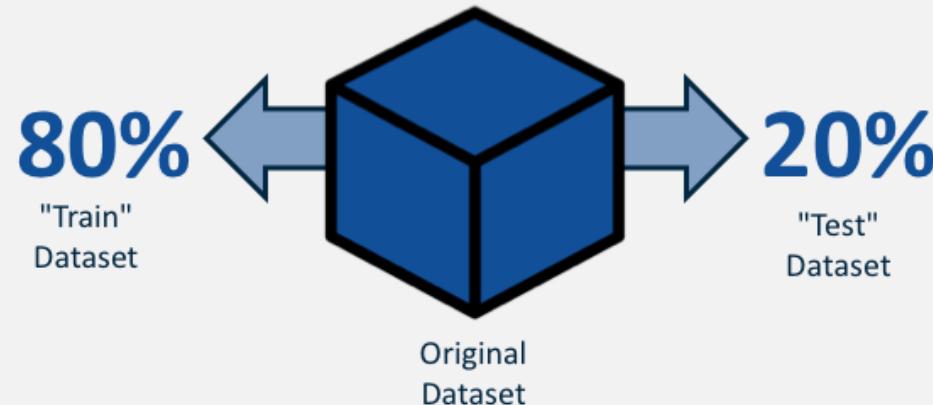
## Steps to Train and Validate Model

- A. Split data into Train vs. Test datasets:  
Load "Train" dataset
- B. Create decoy model
- C. Model is trained on "Train" dataset
- D. Validate and test the model by loading the "Test" dataset

## 5

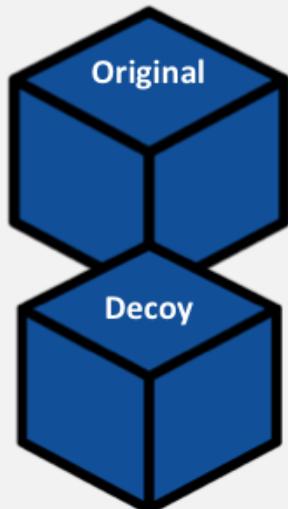
## Supervised Model Training and Validation

- A. Split data into Train vs. Test datasets:  
Load "Train" dataset
- B. Create decoy model
- C. Model is trained on "Train" dataset
- D. Validate and test the model by loading the "Test" dataset



## 5 Supervised Model Training and Validation

- A. Split data into Train vs. Test datasets:  
Load "Train" dataset
- B. **Create decoy model**
- C. Model is trained on "Train" dataset
- D. Validate and test the model by loading the "Test" dataset



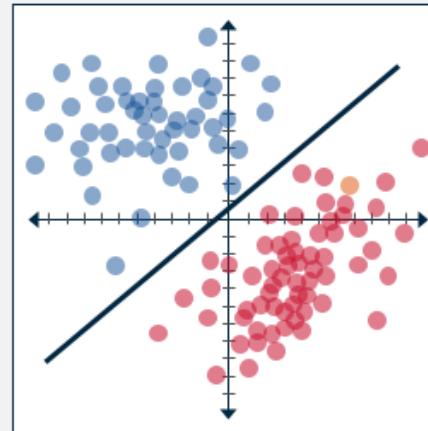
### Decoy "Training" Model

- Preserves specific parameters of the original training of a model
- Accepts exact same training data as the model, but it doesn't actually do anything.
- Saves the information as an archive that can be referenced later, if needed.

## 5

# Supervised Model Training and Validation

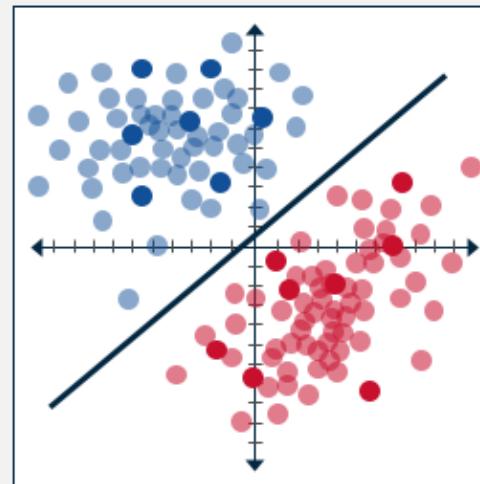
- A. Split data into Train vs. Test datasets:  
Load "Train" dataset
- B. Create decoy model
- C. **Model is trained on "Train" dataset**
- D. Validate and test the model by loading the "Test" dataset



## 5

# Supervised Model Training and Validation

- A. Split data into Train vs. Test datasets:  
Load "Train" dataset
- B. Create decoy model
- C. Model is trained on "Train" dataset
- D. **Validate and test the model by loading the "Test" dataset**

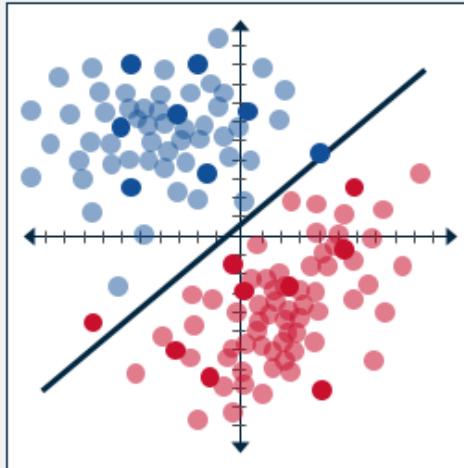


## 5 Supervised Model Training and Validation

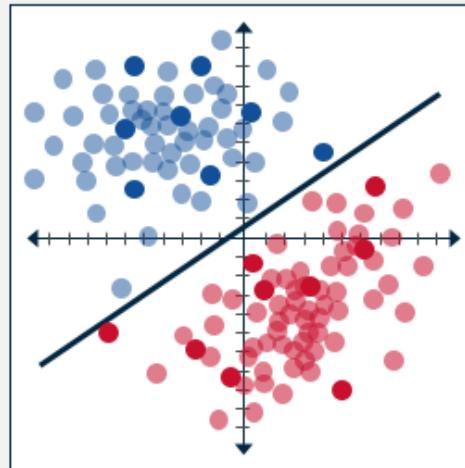
- A. Split data into Train vs. Test datasets:  
Load "Train" dataset
- B. Create decoy model
- C. Model is trained on "Train" dataset
- D. **Validate and test the model by loading the "Test" dataset**

Validate: Retrain/update model parameters until required accuracy levels are met.

Trained Model

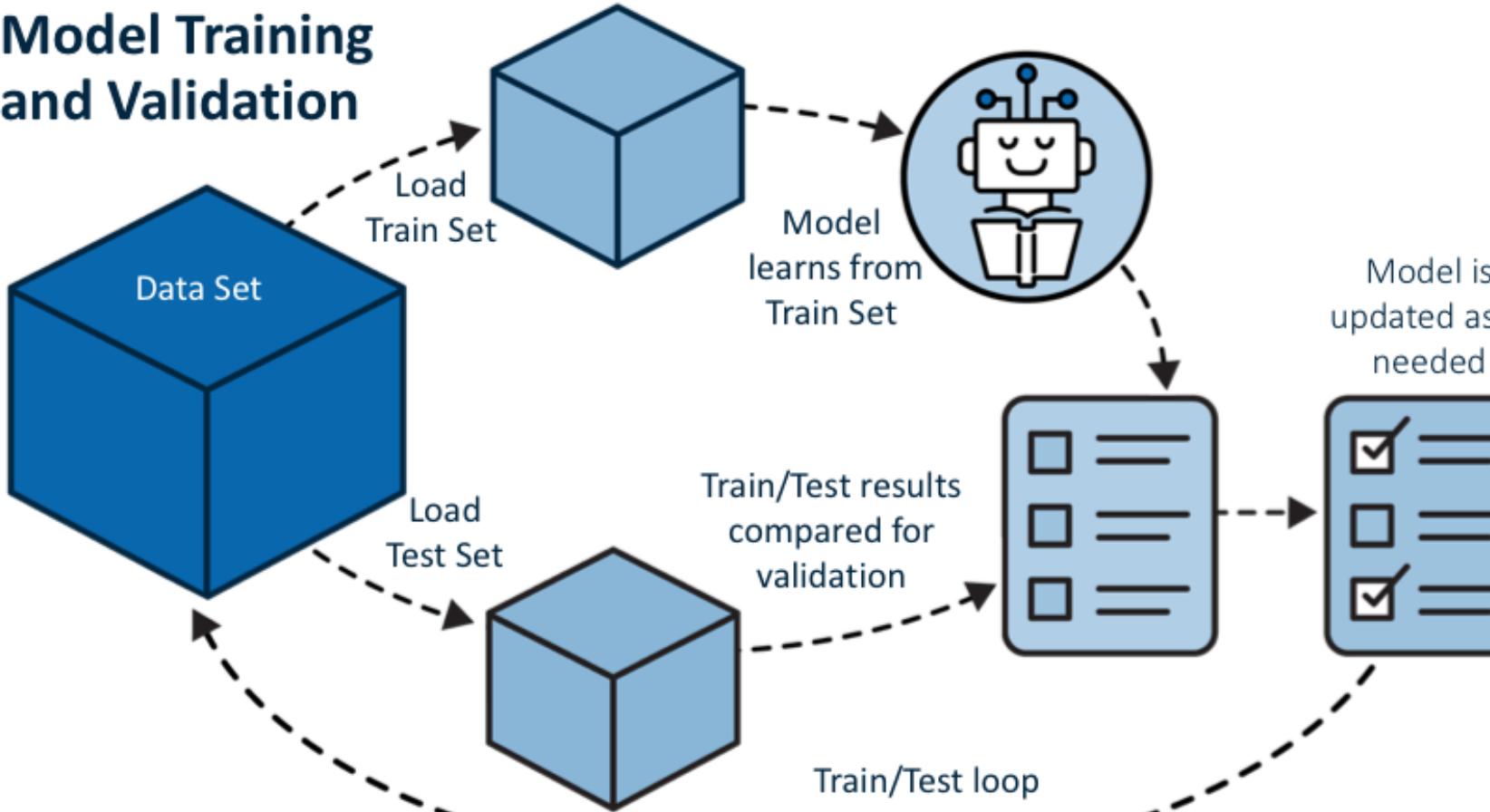


Validated Model



## 5

# Model Training and Validation



# Project Plan Workflow: Step 6

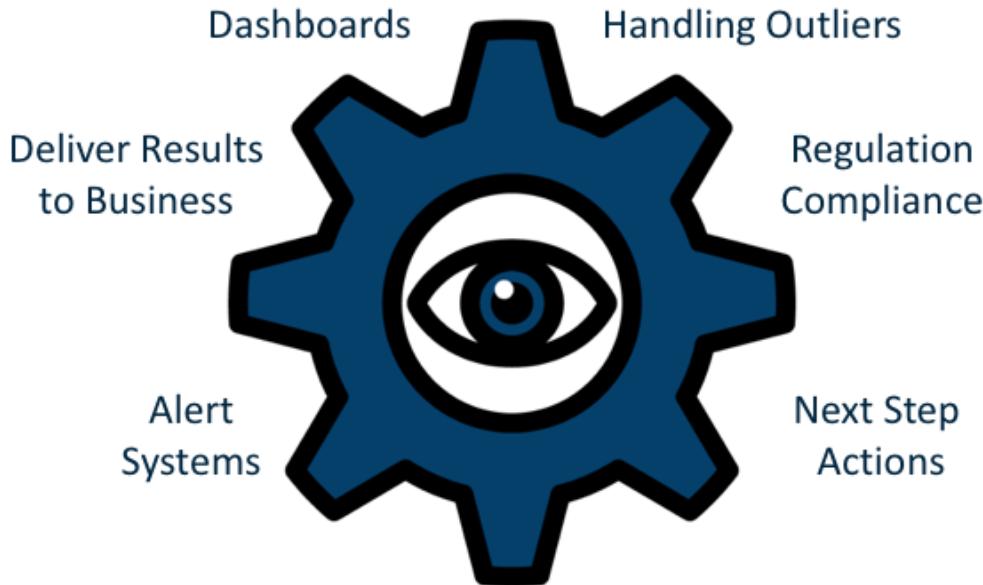
| Part II (Implementation)  |   |   |
|---|---|---|
| Framework   | Model   | Production  |
| 4   | 5   | 6   |
| Select algorithms and frameworks  | Train and validate model  | Implement and monitor project   |
|  |  |  |

# The Full Project Plan Workflow

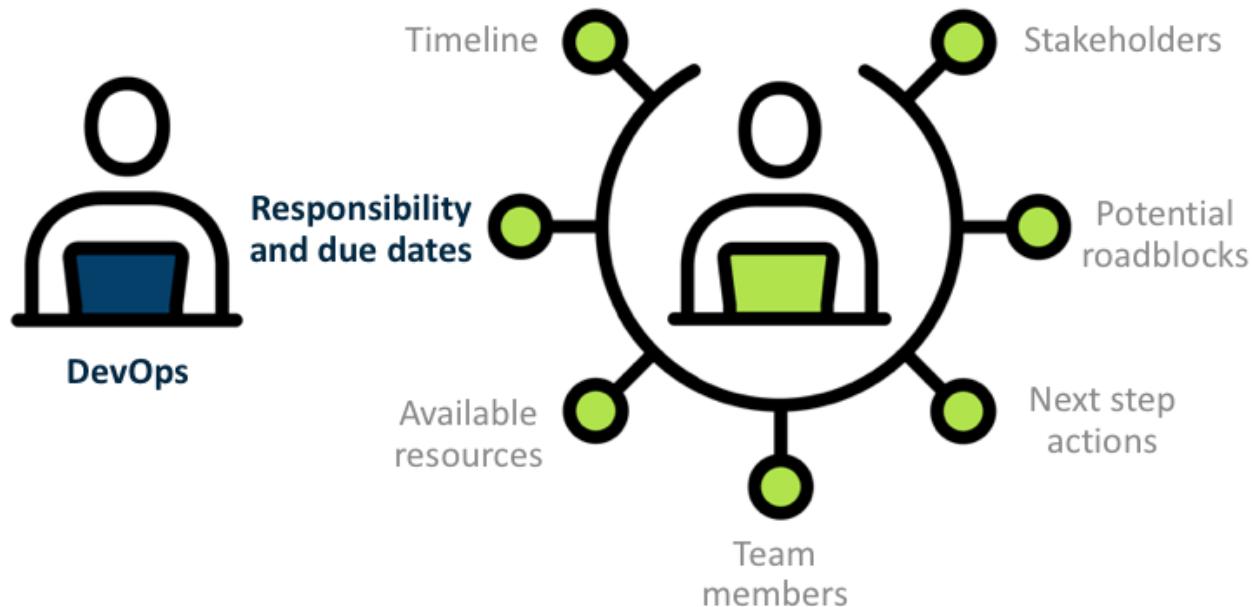
| Part I (Preparation)                             |                         |  | Part II (Implementation)         |                          |                               |
|--|-------------------------|--|----------------------------------|--------------------------|-------------------------------|
| Plan   | Prepare                 | Features                                   | Framework                        | Model                    | Production                    |
| 1  | 2                       | 3  | 4                                | 5                        | 6                             |
| Identify business need and create a project plan | Manage and prepare data | Select and engineer features for the model | Select algorithms and frameworks | Train and validate model | Implement and monitor project |
|  |                         |  |                                  |                          |                               |

## 6

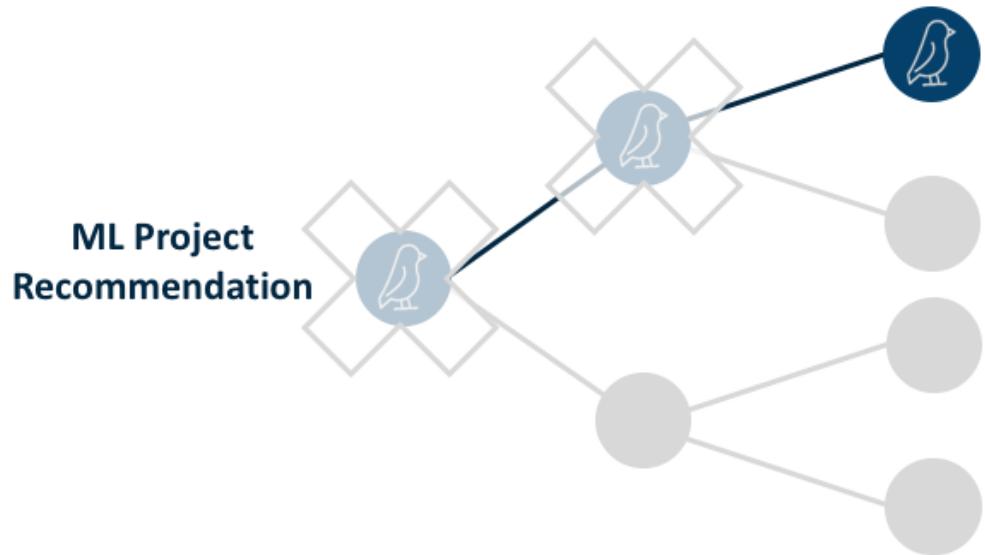
# Implementation to Production and Monitoring



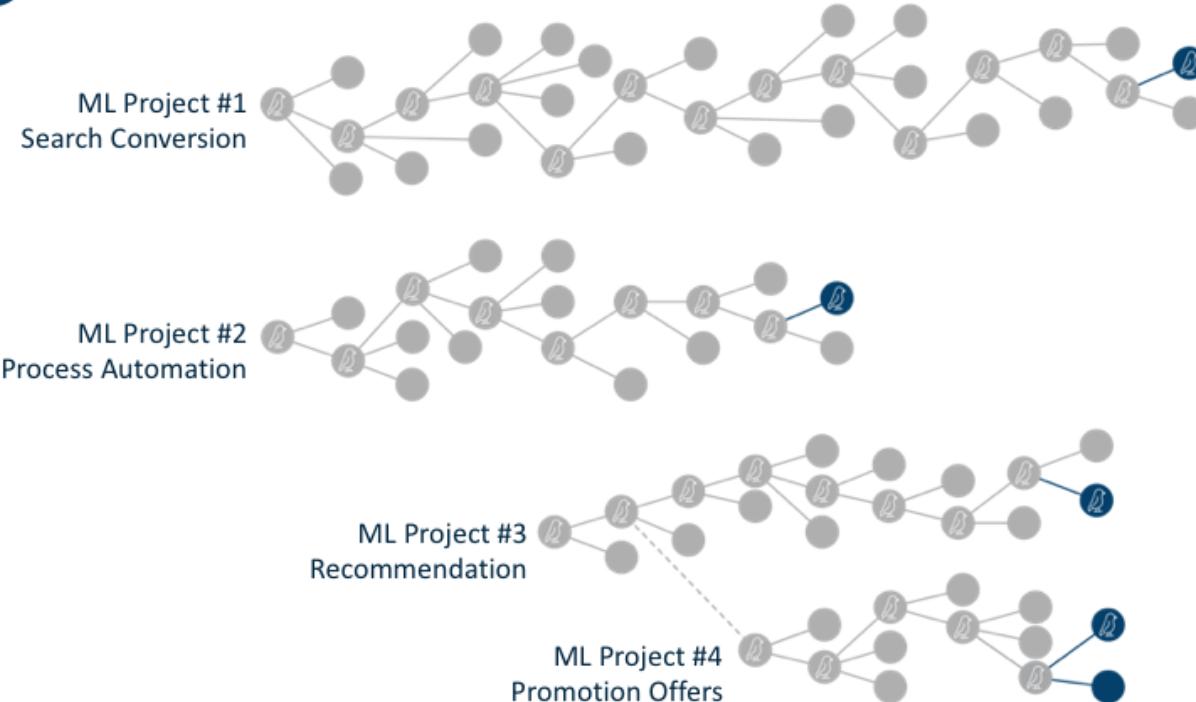
## 6 Implementation: Launch Plan



## 6 Monitoring Results



## 6 Monitoring Results







# Confusion Matrix



# Sometimes accuracy doesn't tell the whole story

- A test for a rare disease can be 99.9% accurate by just guessing “no” all the time
- We need to understand true positives and true negative, as well as false positives and false negatives.
- A confusion matrix shows this.



# Binary confusion matrix

|               |                 | Actual YES      | Actual NO |
|---------------|-----------------|-----------------|-----------|
| Predicted YES | TRUE POSITIVES  | FALSE POSITIVES |           |
| Predicted NO  | FALSE NEGATIVES | TRUE NEGATIVE   |           |

# Image has cat?

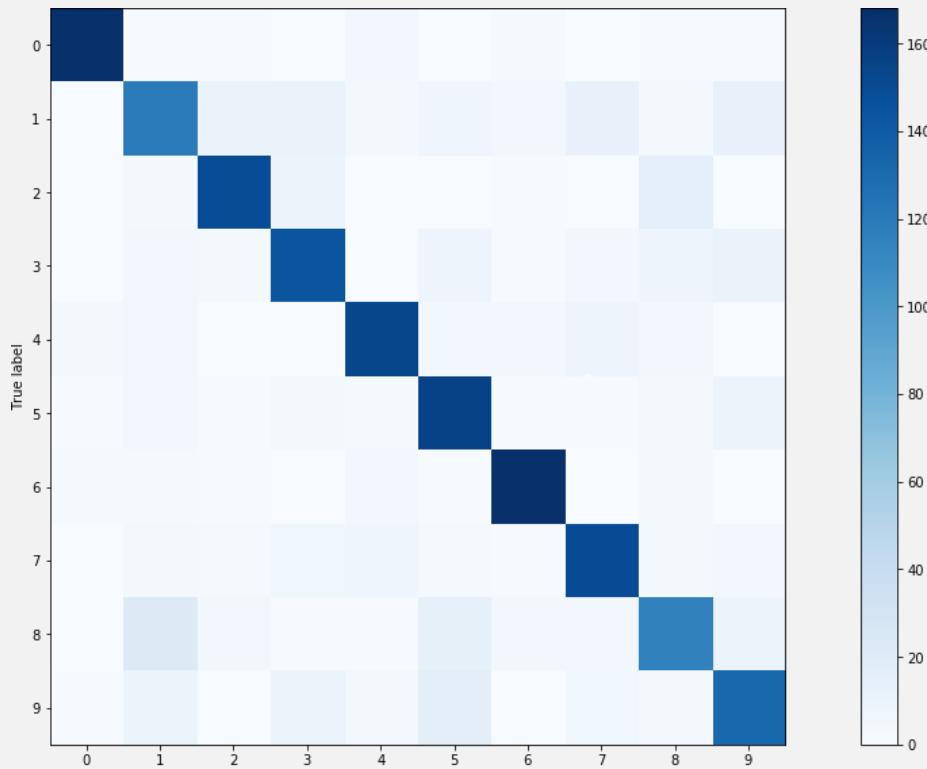
|                   |    | Actual cat | Actual not cat |
|-------------------|----|------------|----------------|
| Predicted cat     | 50 | 5          |                |
| Predicted not cat | 10 | 100        |                |



# Another format

|            | Predicted NO | Predicted YES |     |
|------------|--------------|---------------|-----|
| Actual NO  | 50           | 5             | 55  |
| Actual YES | 10           | 100           | 110 |
|            | 60           | 105           |     |

# Multi-class confusion matrix + heat map



# Measuring your Models



# Remember our friend the confusion matrix

|               |                 | Actual YES      | Actual NO |
|---------------|-----------------|-----------------|-----------|
| Predicted YES | TRUE POSITIVES  | FALSE POSITIVES |           |
| Predicted NO  | FALSE NEGATIVES | TRUE NEGATIVE   |           |

# Recall

*TRUE POSITIVES*

---

*TRUE POSITIVES+FALSE NEGATIVES*

- AKA Sensitivity, True Positive rate, Completeness
- Percent of negatives wrongly predicted
- Good choice of metric when you care a lot about false negatives
  - i.e., fraud detection

# Recall example

|                     | Actual fraud | Actual not fraud |
|---------------------|--------------|------------------|
| Predicted fraud     | 5            | 20               |
| Predicted not fraud | 10           | 100              |

$$\text{Recall} = \text{TP}/(\text{TP}+\text{FN})$$

$$\text{Recall} = 5/(5+10) = 5/15 = 1/3 = 33\%$$

# Precision

*TRUE POSITIVES*

---

*TRUE POSITIVES+FALSE POSITIVES*

- AKA Correct Positives
- Percent of relevant results
- Good choice of metric when you care a lot about false positives
  - i.e., medical screening, drug testing

# Precision example

|                     |    | Actual fraud | Actual not fraud |
|---------------------|----|--------------|------------------|
|                     |    | 5            | 20               |
| Predicted fraud     | 5  | 20           |                  |
| Predicted not fraud | 10 | 100          |                  |

$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$

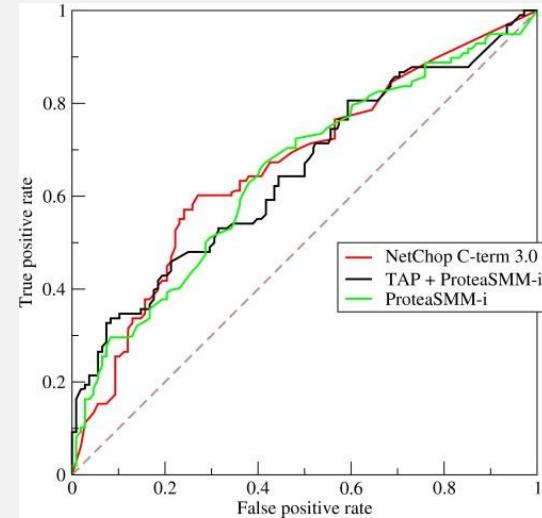
$\text{Precision} = \frac{5}{5+20} = \frac{5}{25} = \frac{1}{5} = 20\%$

# Other metrics

- Specificity =  $\frac{TN}{TN+FP}$  = “True negative rate”
- F1 Score
  - $\frac{2TP}{2TP+FP+FN}$
  - $2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$
  - Harmonic mean of precision and sensitivity
  - When you care about precision AND recall

# ROC Curve

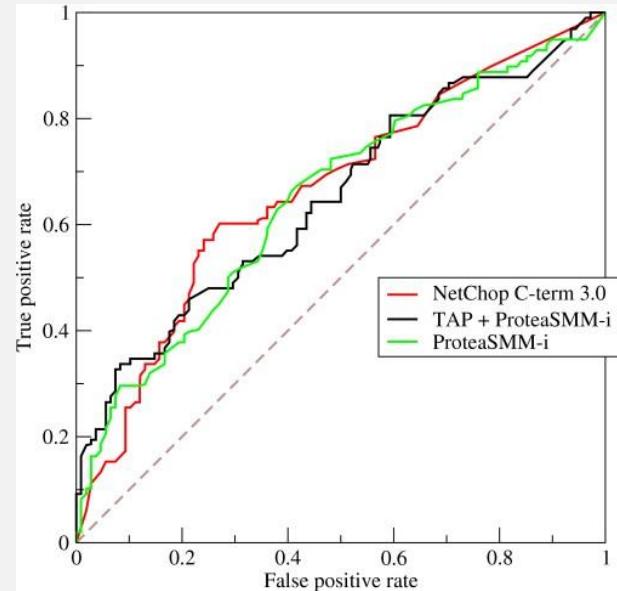
- Receiver Operating Characteristic Curve
- Plot of true positive rate (recall) vs. false positive rate at various threshold settings.
- Points above the diagonal represent good classification (better than random)
- Ideal curve would just be a point in the upper-left corner



BOR at the English language Wikipedia [CC BY-SA 3.0  
(<http://creativecommons.org/licenses/by-sa/3.0/>)]

# AUC

- The area under the ROC curve is... wait for it..
- Area Under the Curve (AUC)
- Equal to probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one
- ROC AUC of 0.5 is a useless classifier, 1.0 is perfect

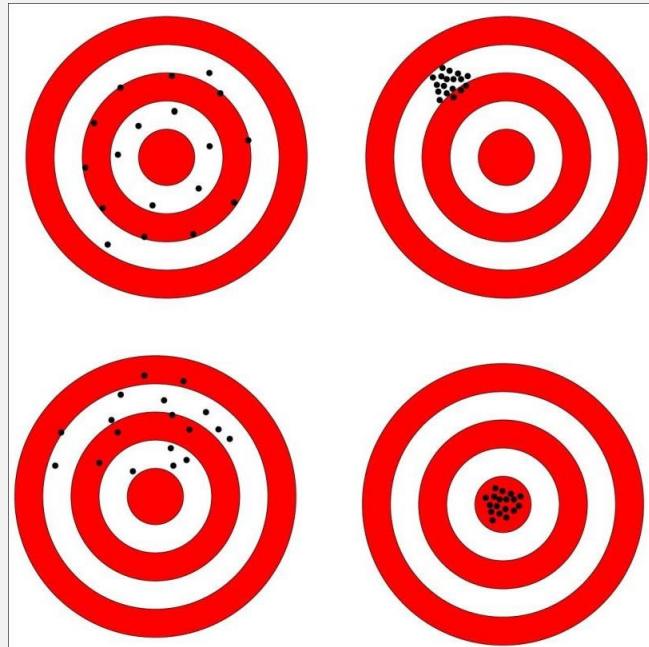


# The Bias / Variance Tradeoff



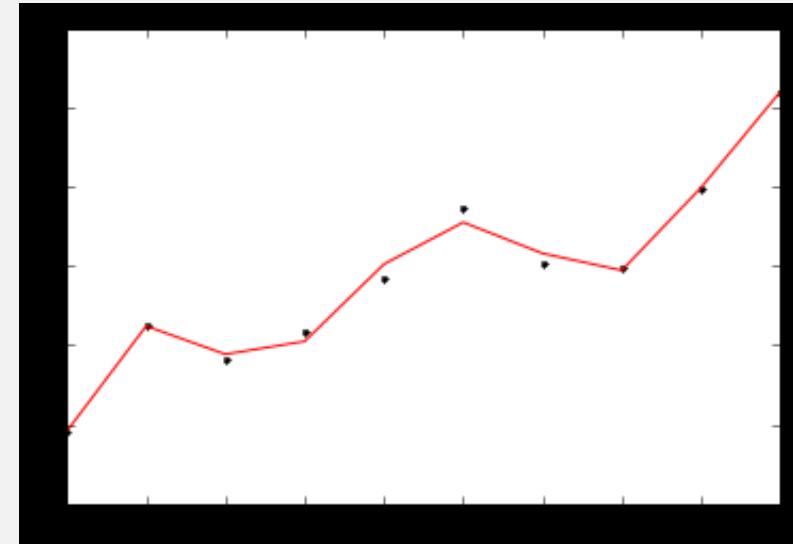
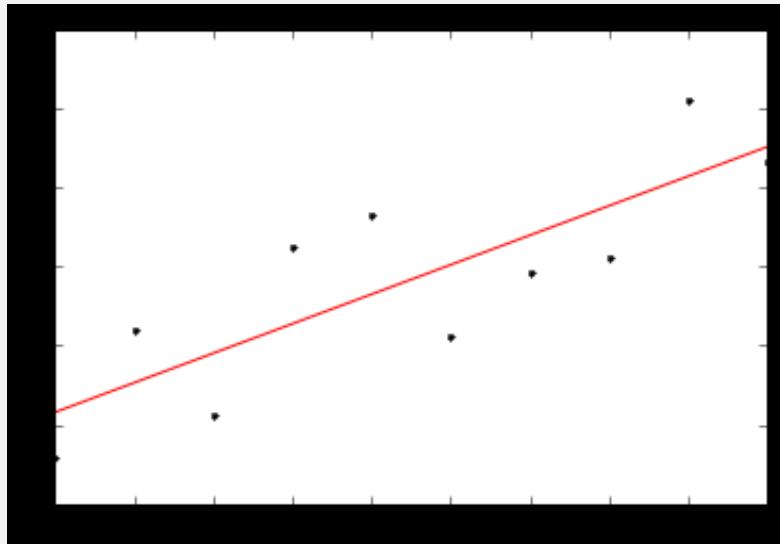
# Bias and Variance

- *Bias* is how far removed the mean of your predicted values is from the “real” answer
- *Variance* is how scattered your predicted values are from the “real” answer



# Often you need to choose between bias and variance

- It comes down to overfitting vs underfitting your data



# But what you really care about is error

- Bias and variance both contribute to error
  - $Error = Bias^2 + Variance$
- But it's error you want to minimize, not bias or variance specifically
- A complex model will have high variance and low bias
- A too-simple model will have low variance and high bias
- But both may have the same error – the optimal complexity is in the middle

# Tying it to earlier lessons

- Increasing K in K-Nearest-Neighbors decreases variance and increases bias (by averaging together more neighbors)
- A single decision tree is prone to overfitting – high variance
  - But a random forest decreases that variance.

# Avoiding Overfitting



# Review: K-Fold Cross Validation

- One way to further protect against overfitting is *K-fold cross validation*
- Sounds complicated. But it's a simple idea:
  - Split your data into K randomly-assigned segments
  - Reserve one segment as your test data
  - Train on each of the remaining K-1 segments and measure their performance against the test set
  - Take the average of the K-1 r-squared scores

# Using K-Fold Cross Validation

- Scikit-learn makes this really easy. Even easier than just a single train/test split.
- In practice, you need to try different variations of your model and measure the mean accuracy using K-Fold Cross validation until you find a sweet spot

# Let's Play

- Use K-Fold Cross Validation with a SVC model of Iris classification.
- We'll see that without K-Fold, we could overfit the model.



# Cleaning Your Data



# Cleaning your Data

- The reality is, much of your time as a data scientist will be spent preparing and “cleaning” your data
  - Outliers
  - Missing Data
  - Malicious Data
  - Erroneous Data
  - Irrelevant Data
  - Inconsistent Data



# Garbage In, Garbage Out

- Look at your data! Examine it!
- Question your results!
  - And *always* do this – not just when you don't get a result that you like!



# Let's analyze some web log data

- All I want is the most-popular pages on my non-profit news website.
- How hard can that be?
- <http://bit.ly/2RGZUzW>

DECEMBER 29, 2015

Local News: Automatic ▾



News that won't crush your soul.

HEADLINES US/WORLD AUSTRALIA COMICS + MORE WEATHER SCIENCE TECHNOLOGY  
ENTERTAINMENT TRAVEL SPORTS BUSINESS ABOUT

## Technology

### [The Best WIRED Photo Stories of the Year](#)



From the coldest place on earth to the secret home of the Internet, photography took us some weird and wonderful places this year. The post The Best WIRED Photo Stories of the Year appeared first on WIRED.

[...]

Tue, Dec 29, 2015

Source: Wired

### [How a Nation of Tech Copycats Transformed Into a Hub for Innovation](#)



China, once known more for manufacturing stuff for

### Latest Stories

Reds deal Chapman to Dodgers for 2 prospects

*CBS Sports*

Ben Silverman Talks 'Dream and Harsh Nightmare' of His NBC Tenure

*Variety*

Iraqi state TV: Prime Minister Haider al-Abadi is in Ramadi to hail city's liberation from IS

*Washington Post World*

Bennet Omalu, doctor who raised alarm bells about NFL head injuries, on racism in U.S. science

*Washington Post National*

Valeant CEO takes medical leave of absence

*CNN Money*

# Normalizing Numerical Data



# The importance of normalizing data

- If your model is based on several numerical attributes – are they comparable?
  - Example: ages may range from 0-100, and incomes from 0-billions
  - Some models may not perform well when different attributes are on very different scales
  - It can result in some attributes counting more than others
  - Bias in the attributes can also be a problem.

# Examples

- Scikit-learn's PCA implementation has a “whiten” option that does this for you. Use it.
- Scikit-learn has a preprocessing module with handy normalize and scale functions
- Your data may have “yes” and “no” that needs to be converted to “1” and “0”

# Read the docs

- Most data mining and machine learning techniques work fine with raw, un-normalized data
- But double check the one you're using before you start.
- Don't forget to re-scale your results when you're done!

# Dealing with Outliers



# Dealing with Outliers

- Sometimes it's appropriate to remove outliers from your training data
- Do this responsibly! Understand why you are doing this.
- For example: in collaborative filtering, a single user who rates thousands of movies could have a big effect on everyone else's ratings.



That may not be desirable.

# Dealing with Outliers

- Our old friend standard deviation provides a principled way to classify outliers.
- Find data points more than some multiple of a standard deviation in your training data.
- What multiple? You just have to use common sense.

# Feature Engineering



# What is feature engineering?

- Applying your knowledge of the data – and the model you’re using - to create better features to train your model with.
  - Which features should I use?
  - Do I need to transform these features in some way?
  - How do I handle missing data?
  - Should I create new features from the existing ones?
- You can’t just throw in raw data and expect good results

# The Curse of Dimensionality

- Too many features can be a problem – leads to sparse data
- Every feature is a new dimension
- Much of feature engineering is selecting the features most relevant to the problem at hand
  - This often is where domain knowledge comes into play



# Imputing Missing Data: Mean Replacement

- Replace missing values with the mean value from the rest of the column (columns, not rows! A column represents a single feature; it only makes sense to take the mean from other samples of the same feature.)

```
In [3]: import pandas as pd  
  
masses_data = pd.read_csv('mammographic_masses.data.txt',  
masses_data.head()  
  
Out[3]:
```

|   | BI-RADS | age  | shape | margin | density | severity |
|---|---------|------|-------|--------|---------|----------|
| 0 | 5.0     | 67.0 | 3.0   | 5.0    | 3.0     | 1        |
| 1 | 4.0     | 43.0 | 1.0   | 1.0    | NaN     | 1        |
| 2 | 5.0     | 58.0 | 4.0   | 5.0    | 3.0     | 1        |
| 3 | 4.0     | 28.0 | 1.0   | 1.0    | 3.0     | 0        |
| 4 | 5.0     | 74.0 | 1.0   | 5.0    | NaN     | 1        |

```
In [6]: mean_imputed = masses_data.fillna(masses_data.mean())  
mean_imputed.head()  
  
Out[6]:
```

|   | BI-RADS | age  | shape | margin | density  | severity |
|---|---------|------|-------|--------|----------|----------|
| 0 | 5.0     | 67.0 | 3.0   | 5.0    | 3.000000 | 1        |
| 1 | 4.0     | 43.0 | 1.0   | 1.0    | 2.910734 | 1        |
| 2 | 5.0     | 58.0 | 4.0   | 5.0    | 3.000000 | 1        |
| 3 | 4.0     | 28.0 | 1.0   | 1.0    | 3.000000 | 0        |
| 4 | 5.0     | 74.0 | 1.0   | 5.0    | 2.910734 | 1        |

# Imputing Missing Data: Dropping

- If not many rows contain missing data...
  - ...and dropping those rows doesn't bias your data...
  - ...and you don't have a lot of time...
  - ...maybe it's a reasonable thing to do.

```
In [3]: import pandas as pd  
  
masses_data = pd.read_csv('mammographic_masses.data').  
masses_data.head()
```

Out[3]:

|   | BI-RADS | age  | shape | margin | density | severity |
|---|---------|------|-------|--------|---------|----------|
| 0 | 5.0     | 67.0 | 3.0   | 5.0    | 3.0     | 1        |
| 1 | 4.0     | 43.0 | 1.0   | 1.0    | NaN     | 1        |
| 2 | 5.0     | 58.0 | 4.0   | 5.0    | 3.0     | 1        |
| 3 | 4.0     | 28.0 | 1.0   | 1.0    | 3.0     | 0        |
| 4 | 5.0     | 74.0 | 1.0   | 5.0    | NaN     | 1        |

```
In [7]: mean_imputed = masses_data.dropna()  
mean_imputed.head()
```

Out[7]:

|    | BI-RADS | age  | shape | margin | density | severity |
|----|---------|------|-------|--------|---------|----------|
| 0  | 5.0     | 67.0 | 3.0   | 5.0    | 3.0     | 1        |
| 2  | 5.0     | 58.0 | 4.0   | 5.0    | 3.0     | 1        |
| 3  | 4.0     | 28.0 | 1.0   | 1.0    | 3.0     | 0        |
| 8  | 5.0     | 57.0 | 1.0   | 5.0    | 3.0     | 1        |
| 10 | 5.0     | 76.0 | 1.0   | 4.0    | 3.0     | 1        |

# Imputing Missing Data: Machine Learning

- KNN: Find K “nearest” (most similar) rows and average their values
- Deep Learning
- Regression

# Imputing Missing Data: Just Get More Data

- What's better than imputing data? Getting more real data!
- Sometimes you just have to try harder or collect more data

# Handling Unbalanced Data



# What is unbalanced data?

- Large discrepancy between “positive” and “negative” cases
  - i.e., fraud detection. Fraud is rare, and most rows will be not-fraud
  - Don’t let the terminology confuse you; “positive” doesn’t mean “good”
    - It means the thing you’re testing for is what happened.
    - If your machine learning model is made to detect fraud, then fraud is the “positive”



# Oversampling

- Duplicate samples from the minority class
- Can be done at random



# Under sampling

- Instead of creating more positive samples, remove negative ones
- Throwing data away is usually not the right answer
  - Unless you are specifically trying to avoid “big data” scaling issues



# SMOTE

- Synthetic Minority Over-sampling TTechnique
- Artificially generate new samples of the minority class using nearest neighbors
  - Run K-nearest-neighbors of each sample of the minority class
  - Create a new sample from the KNN result (mean of the neighbors)

# Adjusting thresholds

- When making predictions about a classification (fraud / not fraud), you have some sort of threshold of probability at which point you'll flag something as the positive case (fraud)



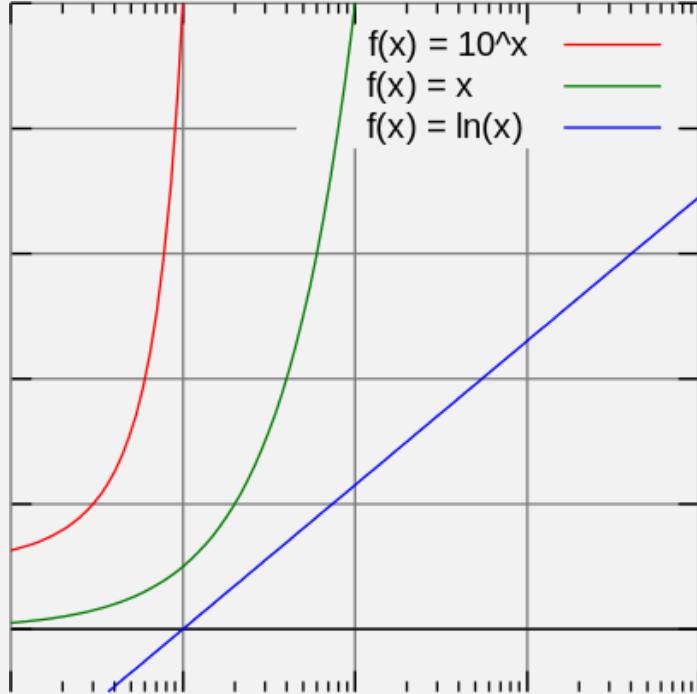
# Binning

- Bucket observations together based on ranges of values.
- Example: estimated ages of people
  - Put all 20-somethings in one classification, 30-somethings in another, etc.



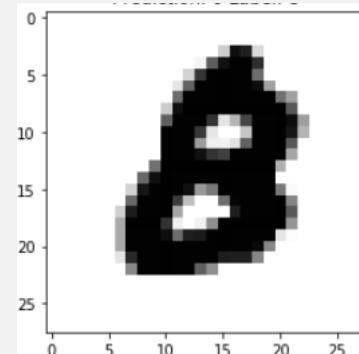
# Transforming

- Feature data with an exponential trend may benefit from a logarithmic transform
- Applying some function to a feature to make it better suited for training



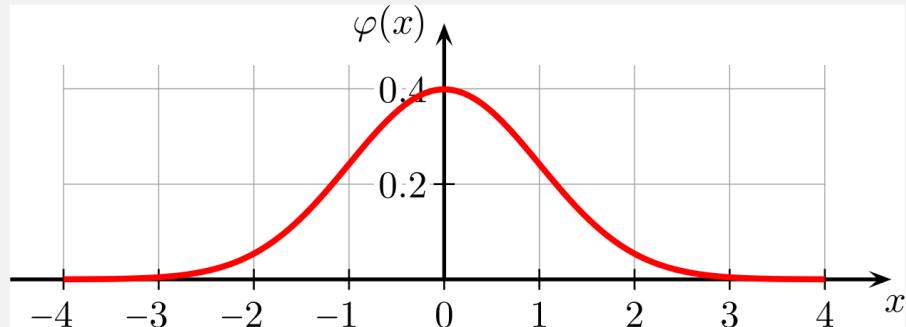
# Encoding

- Transforming data into some new representation required by the model
- One-hot encoding
  - Create “buckets” for every category
  - The bucket for your category has a 1, all others have a 0
  - Very common in deep learning, where categories are represented by individual output “neurons”



# Scaling / Normalization

- Some models prefer feature data to be normally distributed around 0 (most neural nets)
- Most models require feature data to at least be scaled to comparable values



# Shuffling

- Many algorithms benefit from shuffling their training data
- Otherwise they may learn from residual signals in the training data resulting from the order in which they were collected



# Installing Apache Spark on Windows



# Installing Spark on Windows

- Install a JDK
- Install Python (but you should already have this)
- Install a pre-built version of Spark for Hadoop
- Create a conf/log4j.properties file to change the warning level
- Add a SPARK\_HOME environment variable
- Add %SPARK\_HOME%\bin to your PATH
- Set HADOOP\_HOME to c:\winutils
- Install winutils.exe to c:\winutils\bin



# Installing Spark on other OS's

- Pretty much the same, but look up how to set environment variables on your OS
- Winutils.exe not needed of course

# Spark Introduction

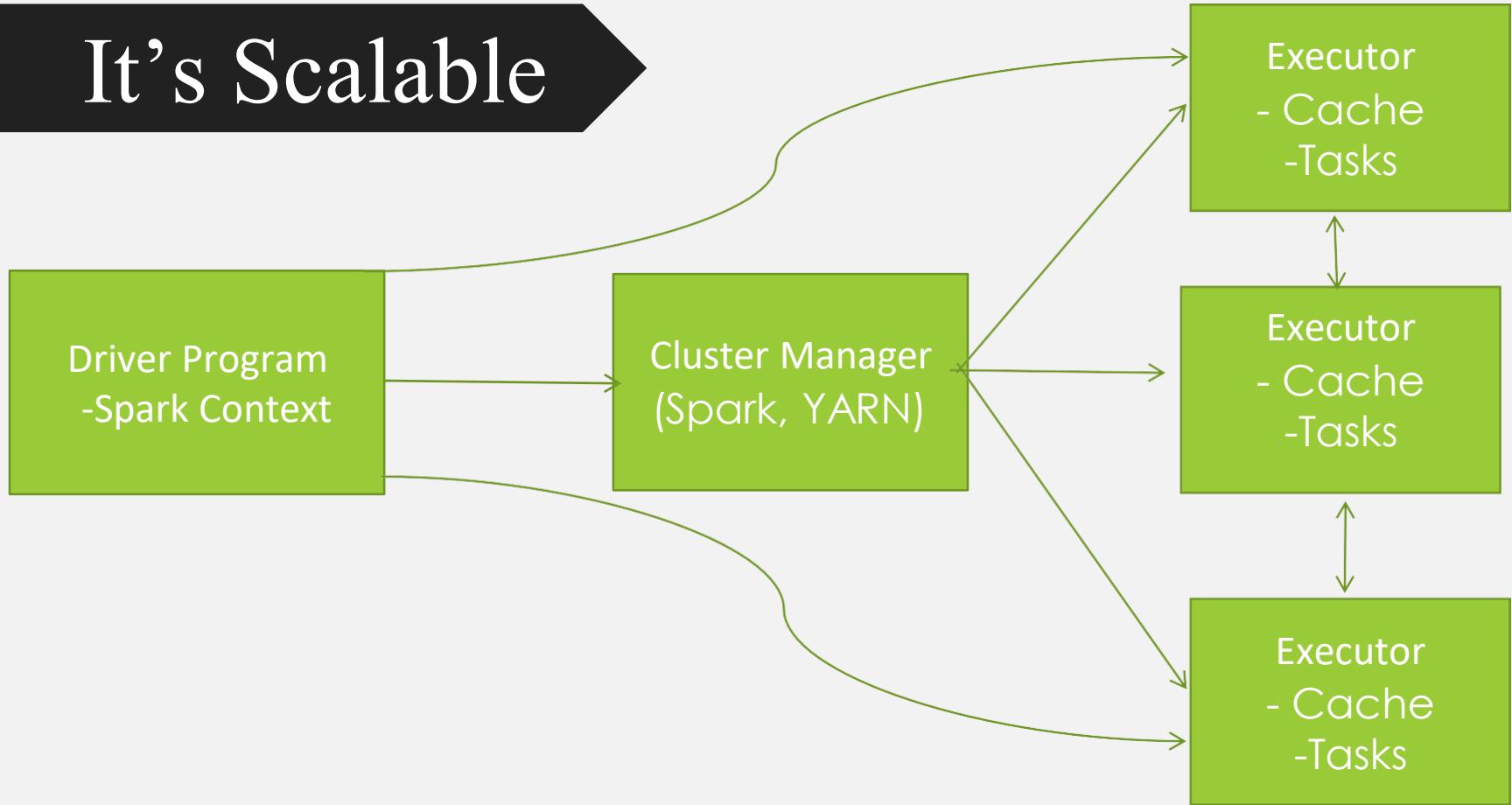


# What is Spark?



- "A fast and general engine for large-scale data processing"

# It's Scalable



# It's Fast

- "Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk."
- DAG Engine (directed acyclic graph) optimizes workflows

# It's Hot

- Amazon
- Ebay: log analysis and aggregation
- NASA JPL: Deep Space Network
- Groupon
- TripAdvisor
- Yahoo
- Many others:  
<https://cwiki.apache.org/confluence/display/SPARK/Power+ed+By+Spark>

# It's Not That Hard

- Code in Python, Java, or Scala
- Built around one main concept: the Resilient Distributed Dataset (RDD)

# Components of Spark

Spark Streaming

Spark SQL

MLLib

GraphX

SPARK CORE

# Python vs. Scala

- Why Python?
  - No need to compile, manage dependencies, etc.
  - Less coding overhead
  - You already know Python
  - Lets us focus on the concepts instead of a new language

# Fear Not

- Python and Scala look very similar in Spark.

**Python code to square numbers in a data set:**

```
nums = sc.parallelize([1, 2, 3, 4])
squared = nums.map(lambda x: x *
x).collect()
```

**Scala code to square numbers in a data set:**

```
val nums = sc.parallelize(List(1, 2, 3, 4))
val squared = nums.map(x => x * x).collect()
```

# Resilient Distributed Datasets (RDDs)



# RDD

Resilient

Distributed

Dataset

# The Spark Context

Created by  
your driver  
program

Is responsible  
for making  
RDD's  
resilient and  
distributed!

Creates  
RDD's

The Spark  
shell creates  
a "sc" object  
for you

# Creating RDD's

- `nums = parallelize([1, 2, 3, 4])`
- `sc.textFile("file:///c:/users/frank/gobs-o-text.txt")`
  - or `s3n://`, `hdfs://`
- `hiveCtx = HiveContext(sc)`   `rows = hiveCtx.sql("SELECT name, age FROM users")`
- Can also create from:
  - JDBC
  - Cassandra
  - HBase
  - Elasticsearch
  - JSON, CSV, sequence files, object files, various compressed formats

# Transforming RDD's

- map
- flatmap
- filter
- distinct
- sample
- union, intersection, subtract,  
cartesian

# Map() example

- `rdd = sc.parallelize([1, 2, 3, 4])`
- `rdd.map(lambda x: x*x)`
- This yields 1, 4, 9, 16

# What's that lambda thing?

- Many RDD methods accept a *function* as a parameter
- `rdd.map(lambda x: x*x)`
- Is the same thing as
- ```
def squareIt(x):  
    return x*x
```
- `rdd.map(squareIt)`
- There, you now understand functional programming.

# RDD Actions

- collect
- count
- countByValue
- take
- top
- reduce
- ... and more ...

# Lazy Evaluation

- Nothing actually happens in your driver program until an action is called!

# Introducing MLLib



# Some MLlib Capabilities

- Feature extraction
  - Term Frequency / Inverse Document Frequency useful for search
- Basic statistics
  - Chi-squared test, Pearson or Spearman correlation, min, max, mean, variance
- Linear regression, logistic regression

# Special MLlib Data Types

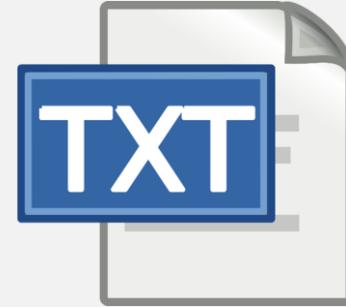
- Vector (dense or sparse)
- Labeled Point
- Rating

# TF-IDF



# TF-IDF

- Stands for *Term Frequency* and *Inverse Document Frequency*
- Important data for search – figures out what terms are most relevant for a document
- Sounds fancy!



# TF-IDF Explained

- *Term Frequency* just measures how often a word occurs in a document
  - A word that occurs frequently is probably important to that document's meaning
- *Document Frequency* is how often a word occurs in an entire set of documents, i.e., all of Wikipedia or every web page
  - This tells us about common words that just appear everywhere no matter what the topic, like “a”, “the”, “and”, etc.

# TF-IDF Explained

- So a measure of the relevancy of a word to a document might be:

$$\frac{\text{Term Frequency}}{\text{Document Frequency}}$$

Or: Term Frequency \* Inverse Document Frequency

# TF-IDF In Practice

- We actually use the log of the IDF, since word frequencies are distributed exponentially. That gives us a better weighting of a words overall popularity
- TF-IDF assumes a document is just a “bag of words”
  - Parsing documents into a bag of words can be most of the work
  - Words can be represented as a hash value (number) for efficiency
  - What about synonyms? Various tenses?

# Using TF-IDF

- A very simple search algorithm could be:
  - Compute TF-IDF for every word in a corpus
  - For a given search word, sort the documents by their TF-IDF score for that word
  - Display the results

# Let's use TF-IDF on Wikipedia

**WIKIPEDIA**  
*The Free Encyclopedia*



# Deploying models for real-time use

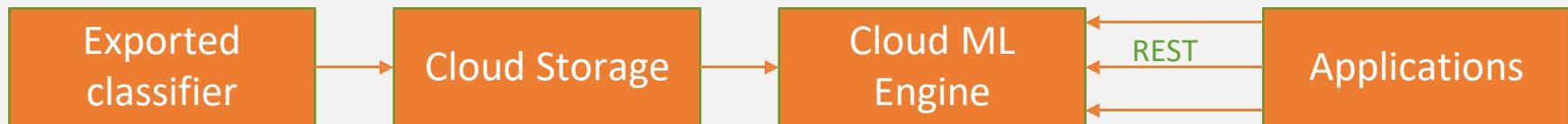


# How do I use my model in an app?

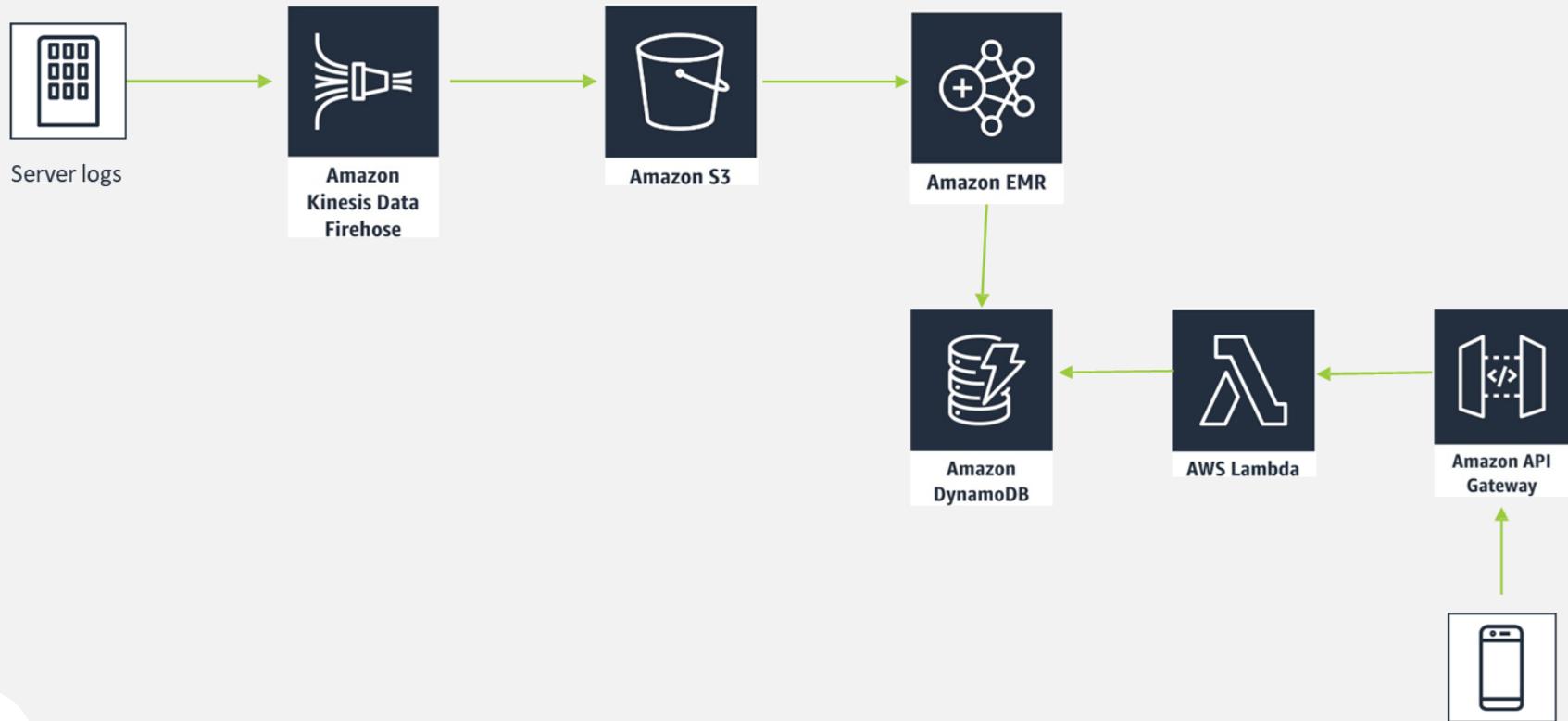
- Your external apps can't just run a notebook!
- Separate your training from your predictions
  - Train the model periodically offline
  - Push the model – or its results – to a web service
  - Your app calls the web service

# Example: Google Cloud ML

- Dump your trained classifier using `sklearn.externals`
  - ```
from sklearn.externals import joblib
joblib.dump(clf, 'model.joblib')
```
- Upload `model.joblib` to Google cloud storage, specifying the scikit-learn framework
- Cloud ML Engine exposes a REST API that you can call to make predictions in real-time



# Example: AWS (recommender system)



# Other approaches

- Roll your own web service with Flask or another framework
  - Then you have servers to provision and maintain  
:/

- Go all-in with a platform



# A/B Tests



# What is an A/B test?

- A controlled experiment, usually in the context of a website
- You test the performance of some change to your website (the variant) and measure conversion relative to your unchanged site (the control.)

## PURCHASE

SALES INFORMATION FOR SILVERLINING AND TRITON SDK LICENSES

### PRO LICENSES

Our software library licenses feature:

PURCHASE A  
LICENSE

- Royalty-free distribution linked into one title or project on one platform
- 3 months of technical support and maintenance
- A personalized license code to unlock our trial SDK's for your project

## PURCHASE

SALES INFORMATION FOR SILVERLINING AND TRITON SDK LICENSES

### PRO LICENSES

Our software library licenses feature:

PURCHASE A  
LICENSE

- Royalty-free distribution linked into one title or project on one platform
- 3 months of technical support and maintenance
- A personalized license code to unlock our trial SDK's for your project

# What sorts of things can you test?

1. Design changes
2. UI flow
3. Algorithmic changes
4. Pricing changes
5. You name it

# How do you measure conversion

- Ideally choose what you are trying to influence
  - Order amounts
  - Profit
  - Ad clicks
  - Order quantity
- But attributing actions downstream from your change can be hard
  - Especially if you're running more than one experiment

# Variance is your Enemy

- Common mistake:
  - Run a test for some small period of time that results in a few purchases to analyze
  - You take the mean order amount from A and B, and declare victory or defeat

# Variance is your Enemy

- Sometimes you need to also look at conversion metrics with less variance
- Order quantities vs. order dollar amounts, for example

# T-Tests and P-Values



# Determining significance

- So, how do we know if a result is likely to be “real” as opposed to just random variation?
- T-tests and P-values

# The T-Statistic

- A measure of the difference between the two sets expressed in units of standard error
- The size of the difference relative to the variance in the data
- A high t value means there's probably a real difference between the two sets

# The P-Value

- Think of it as the probability of A and B satisfying the “null hypothesis”
- So, a low P-Value implies significance.
- It is the probability of an observation lying at an extreme t-value assuming the null hypothesis

# Using P-values

- Choose some threshold for “significance” before your experiment
  - 1%? 5%?
- When your experiment is over:
  - Measure your P-value
  - If it's less than your significance threshold, then you can reject the null hypothesis
    - If it's a positive change, roll it out
    - If it's a negative change, discard it before you lose more money.

# How Long Do I Run an Experiment?



# How do I know when I'm done with an A/B test?

- You have achieved significance (positive or negative)
- You no longer observe meaningful trends in your p-value
  - That is, you don't see any indication that your experiment will “converge” on a result over time
- You reach some pre-established upper bound on time

# A/B Test Gotchas



# Correlation does not imply causation

- Even your low p-value score on a well-designed experiment does not imply causation!
  - It could still be random chance
  - Other factors could be at play
  - It's your duty to ensure business owners understand this

# Novelty Effects

- Changes to a website will catch the attention of previous users who are used to the way it used to be
  - They might click on something simply because it is new
  - But this attention won't last forever
- Good idea to re-run experiments much later and validate their impact
  - Often the “old” website will outperform the “new” one after awhile, simply because it is a change

# Seasonal Effects

- An experiment run over a short period of time may only be valid for that period of time



# Selection Bias

- Sometimes your random selection of customers for A or B isn't really random
  - For example: assignment is based somehow on customer ID
  - But customers with low ID's are better customers than ones with high ID's
- Run an A/A test periodically to check
- Audit your segment assignment algorithms

# Data Pollution

- Are robots (both self-identified and malicious) affecting your experiment?
  - Good reason to measure conversion based on something that requires spending real money

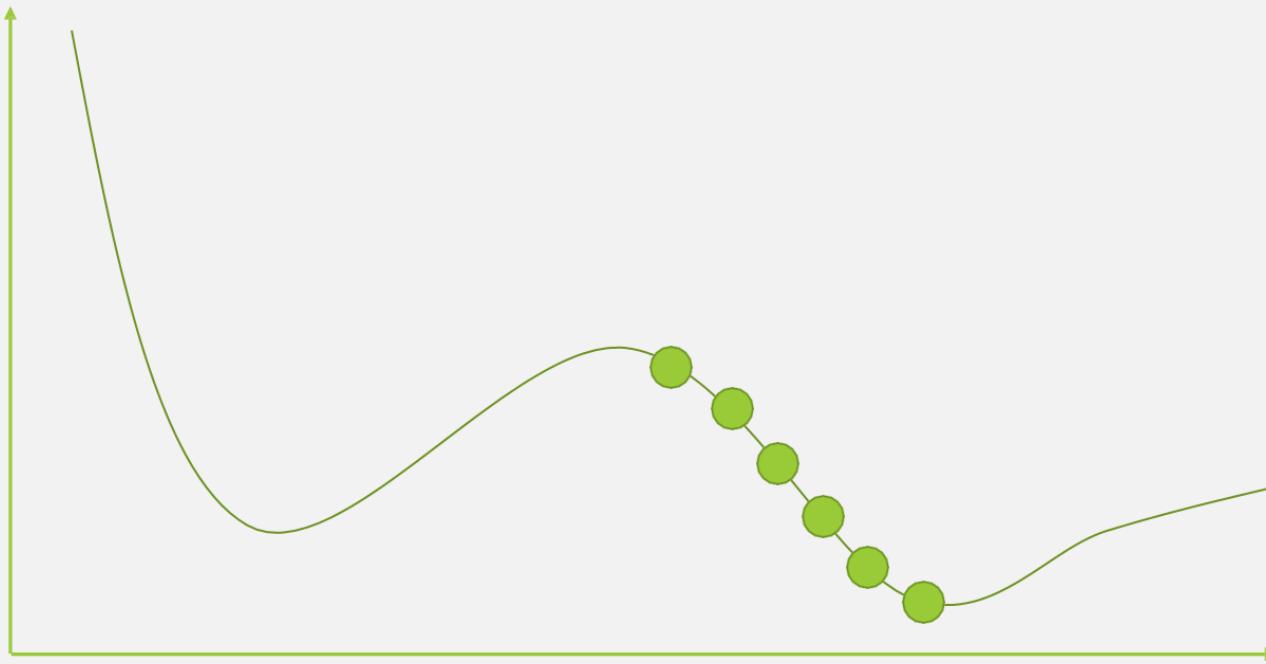


# Attribution Errors

- Often there are errors in how conversion is attributed to an experiment
- Using a widely used A/B test platform can help mitigate that risk
  - If your is home-grown, it deserves auditing

# deep learning pre- requisites

# gradient descent



# autodiff

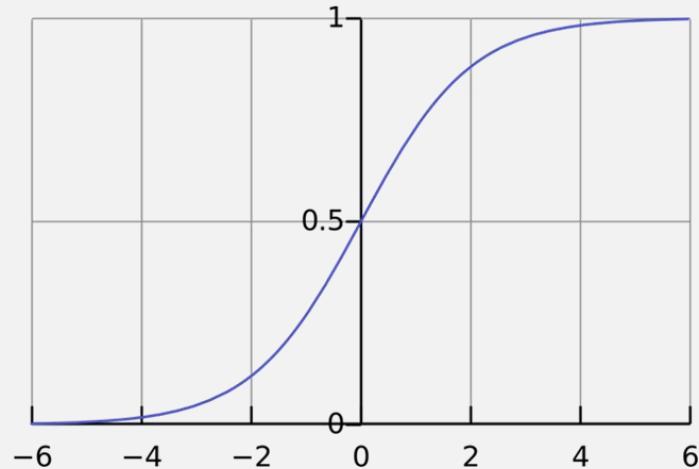
- Gradient descent requires knowledge of, well, the gradient from your cost function (MSE)
- Mathematically we need the first partial derivatives of all the inputs
  - This is hard and inefficient if you just throw calculus at the problem
- Reverse-mode autodiff to the rescue!

# SoftMax

- Used for classification
  - Given a score for each class
  - It produces a probability of each class
  - The class with the highest probability is the “answer” you get

$$h_{\theta}(x) = \frac{1}{1 + \exp(-\theta^T x)},$$

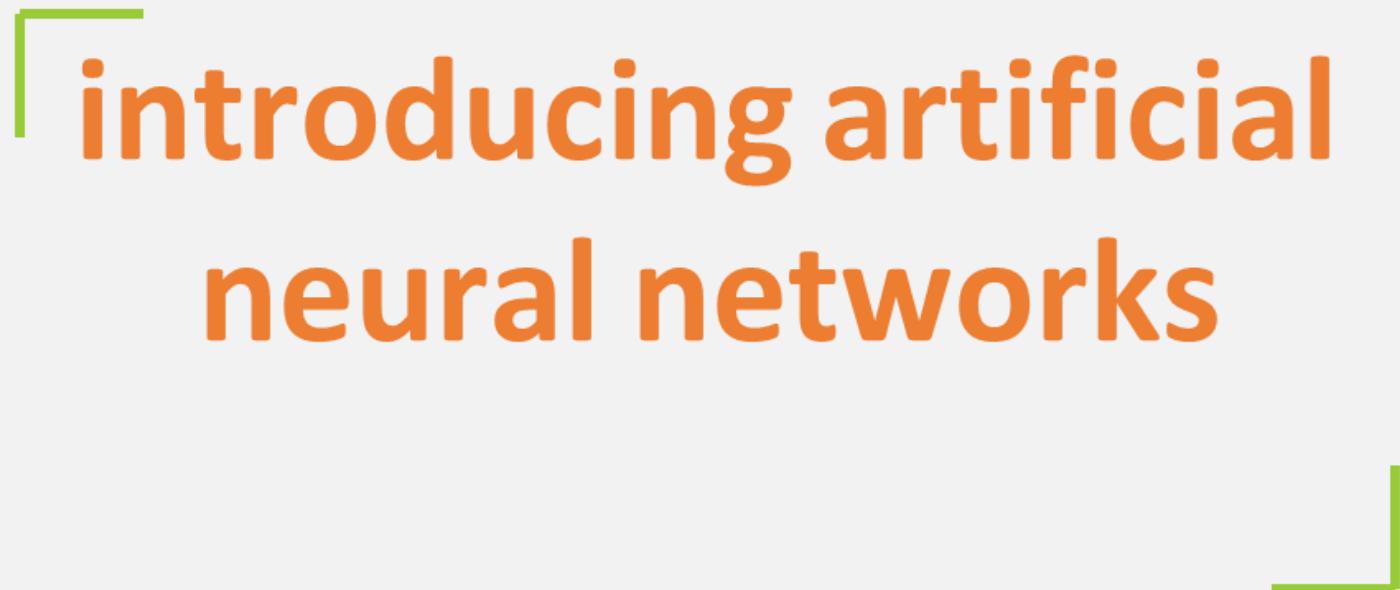
$x$  is a vector of input values  
 $\theta$  is a vector of weights



# in review

- Gradient descent is an algorithm for minimizing error over multiple steps
- Autodiff is a calculus trick for finding the gradients in gradient descent
- Softmax is a function for choosing the most probable classification given several input values





# introducing artificial neural networks

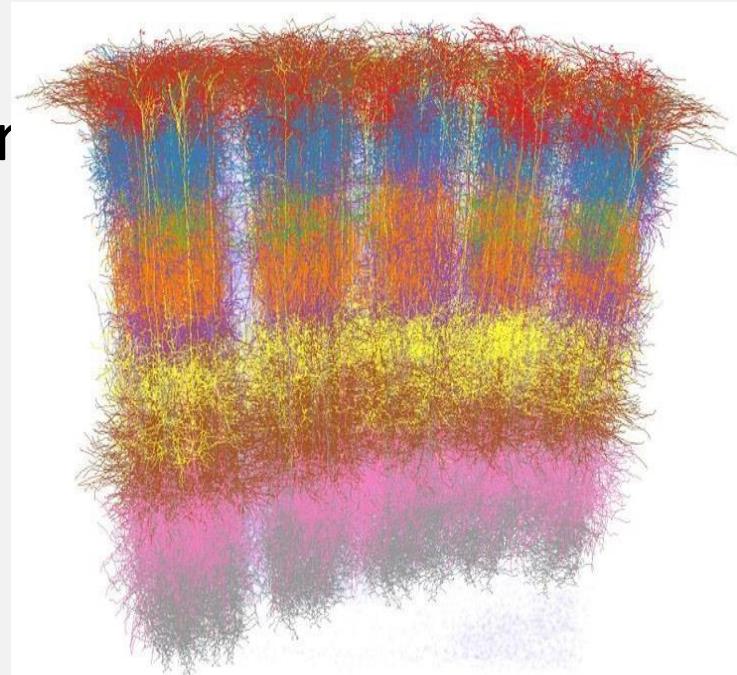
# The biological inspiration

- Neurons in your cerebral cortex are connected via axons
- A neuron “fires” to the neurons it’s connected to, when enough of its input signals are activated.



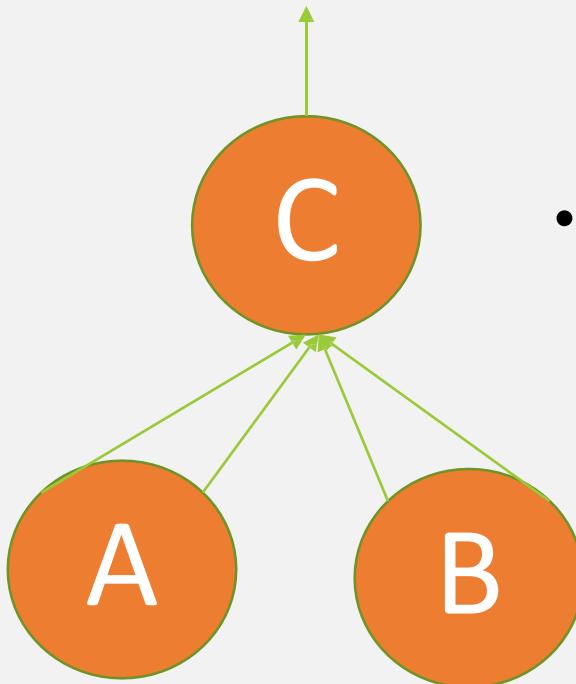
# Cortical columns

- Neurons in your cortex seem to be arranged into many stacks, or “columns” that process information in parallel
- “mini-columns” of around 100 neurons are organized into larger “hyper- columns”. There are 100 million mini- columns in your cortex



# The first artificial neurons

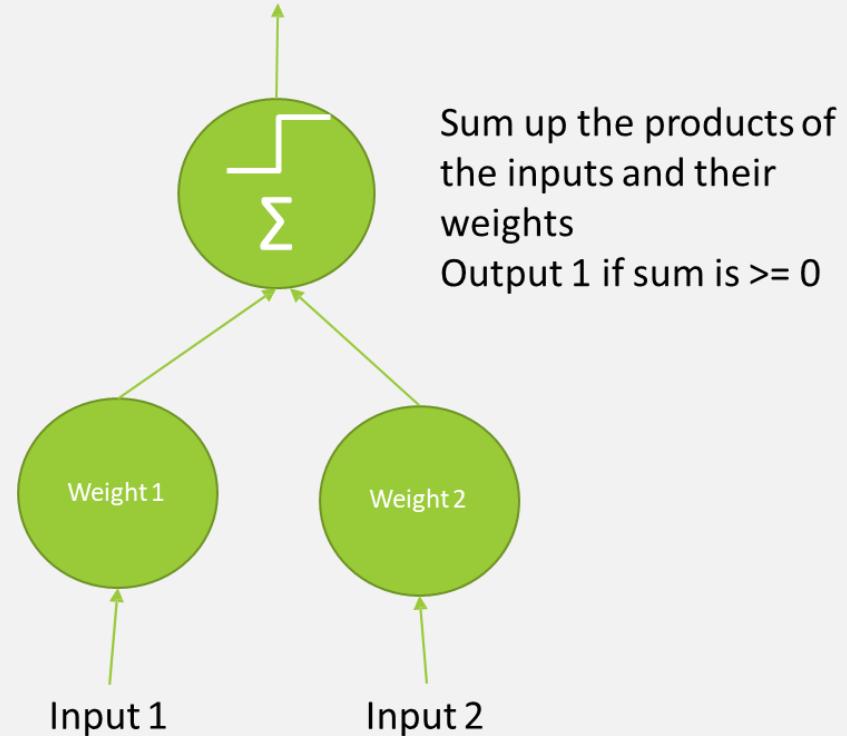
- 1943!!



- This example would implement  $C = A \text{ OR } B$  if the threshold is 2 inputs being active.

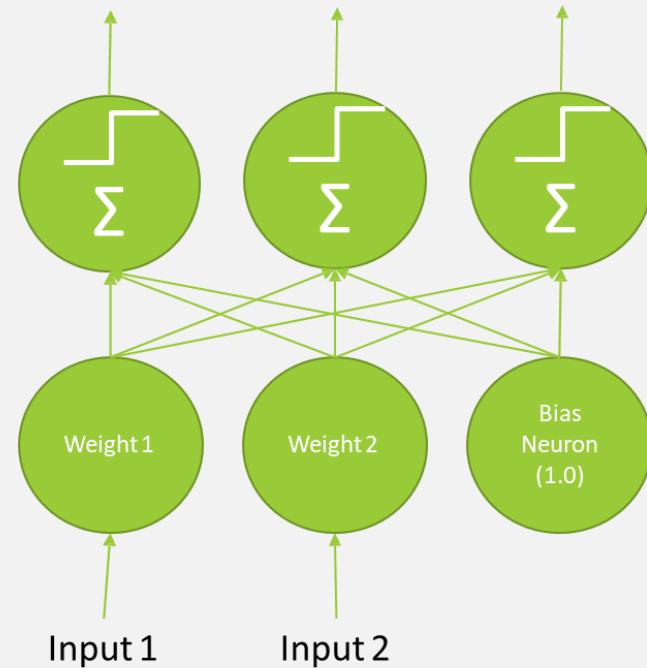
# The linear threshold unit (ltu)

- 1957!
- Adds weights to the inputs; output is given by a step function



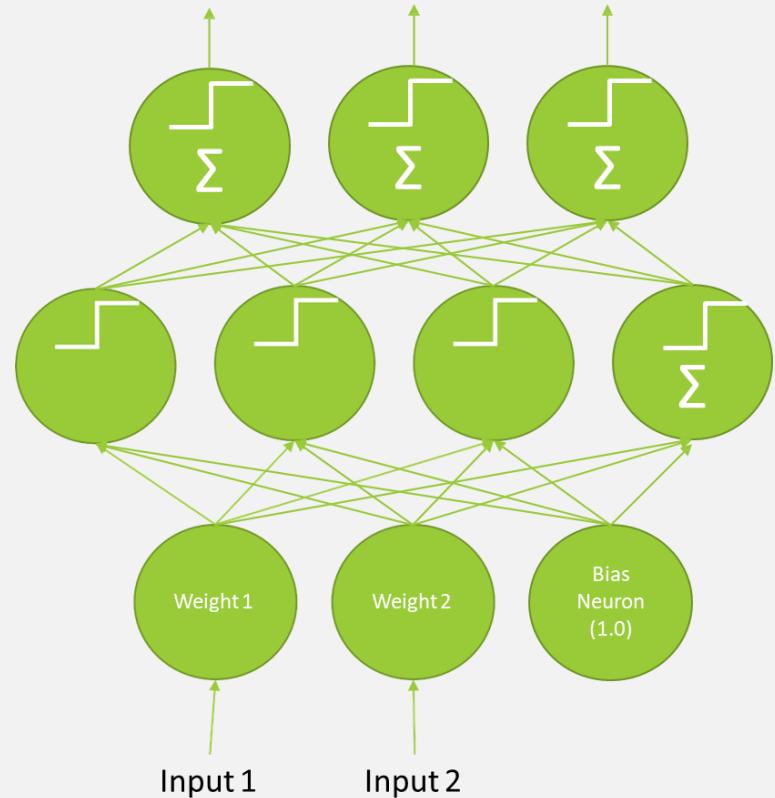
# The perceptron

- A layer of LTU's
- A perceptron can learn by reinforcing weights that lead to correct behavior during training
- This too has a biological basis ("cells that fire together, wire together")



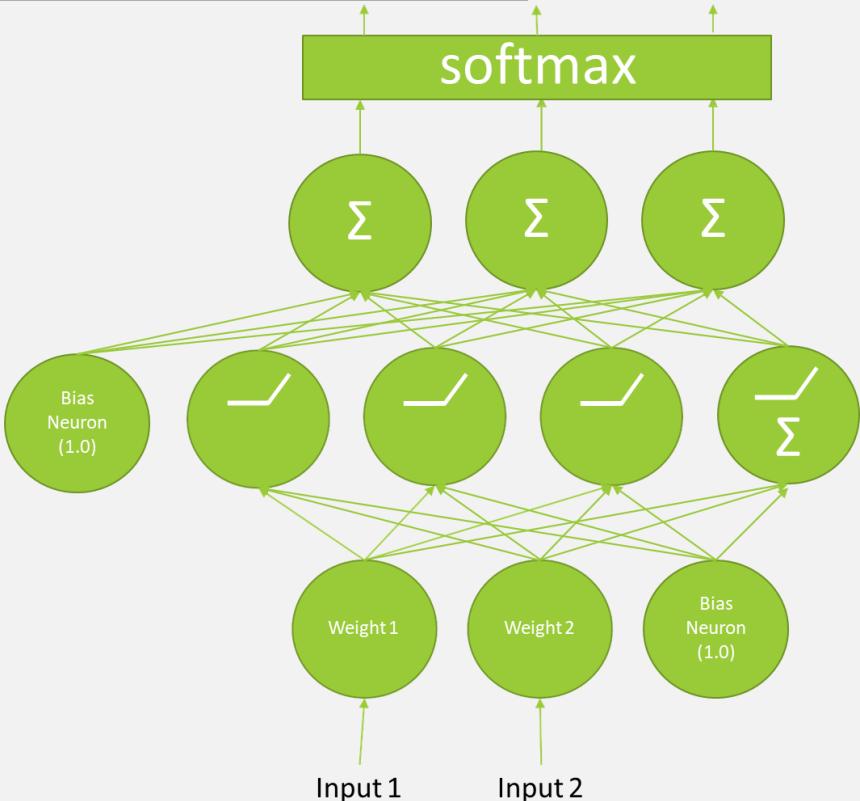
# Multi-layer perceptron

- Addition of “hidden layers”
- This is a Deep Neural Network
- Training them is trickier
  - but we’ll talk about that.



# A modern deep neural network

- Replace step activation function with something better
- Apply softmax to the output
- Training using gradient descent



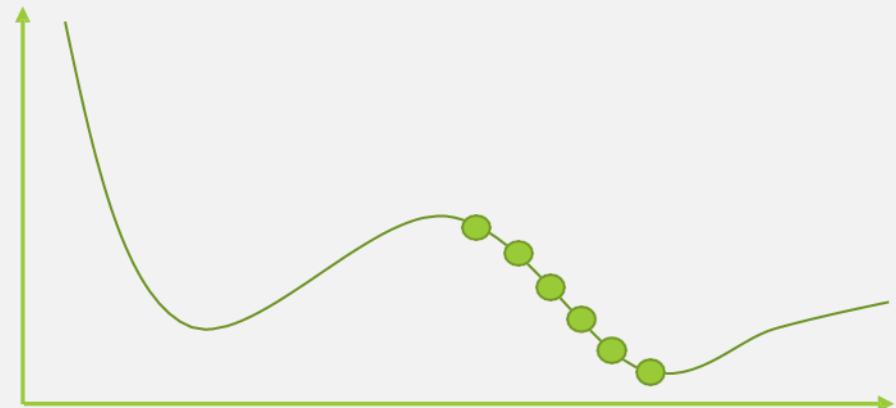


let's play

# deep learning

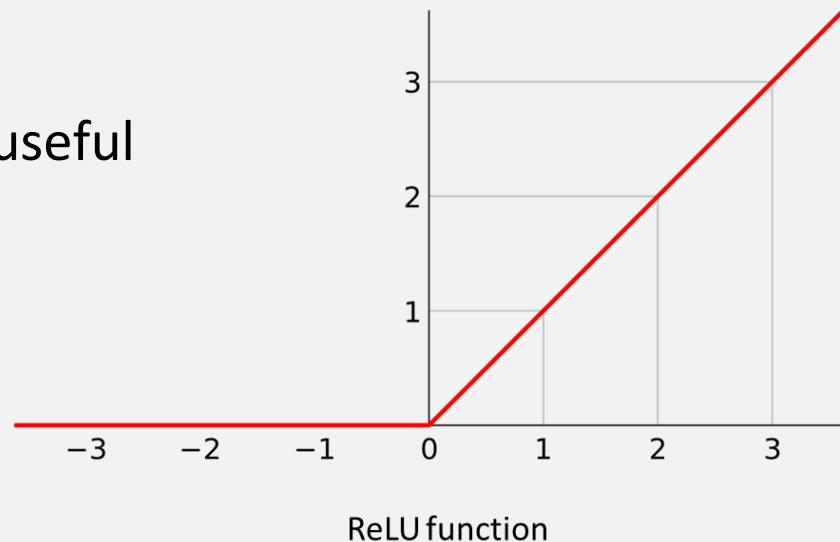
# Backpropagation

- How do you train a MLP's weights? How does it **learn**?
- Backpropagation... or more specifically:  
Gradient Descent using reverse-mode autodiff!



# activation functions (aka rectifier)

- Step functions don't work with gradient descent – there is no gradient!
  - Mathematically, they have no useful derivative.
- Alternatives:
  - Logistic function
  - Hyperbolic tangent function
  - Exponential linear unit (ELU)
  - ReLU function (Rectified Linear Unit)

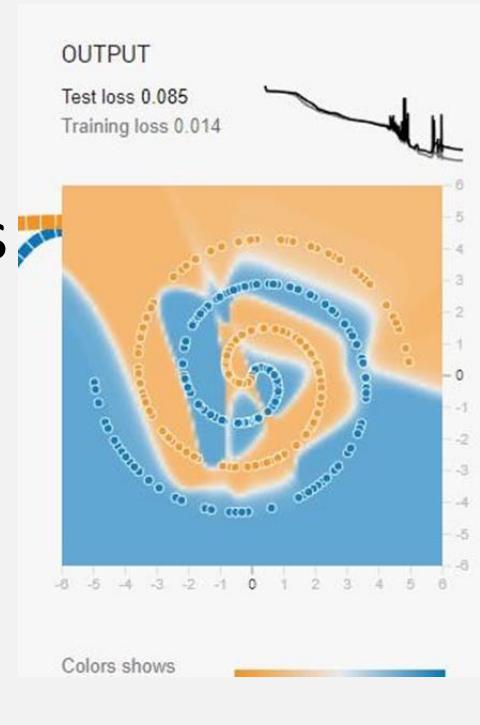


# Optimization functions

- There are faster (as in faster learning) optimizers than gradient descent
  - Momentum Optimization
    - Introduces a momentum term to the descent, so it slows down as things start to flatten and speeds up as the slope is steep
  - Nesterov Accelerated Gradient
    - A small tweak on momentum optimization – computes momentum based on the gradient slightly ahead of you, not where you are
  - RMSProp
    - Adaptive learning rate to help point toward the minimum
  - Adam
    - Adaptive moment estimation – momentum + RMSProp combined
    - Popular choice today – easy to use

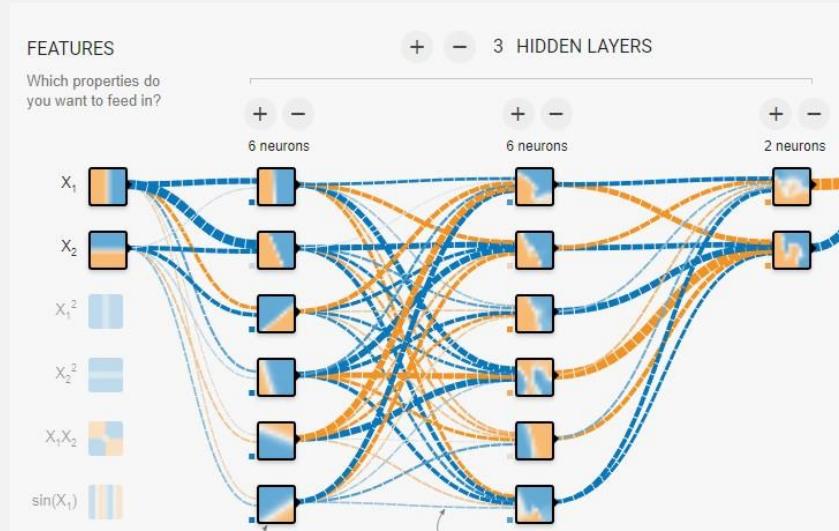
# Avoiding overfitting

- With thousands of weights to tune, overfitting is a problem
- Early stopping (when performance starts dropping)
- Regularization terms added to cost function during training
- Dropout – ignore say 50% of all neurons randomly at each training step



# Tuning your topology

- Trial & error is one way
  - Evaluate a smaller network with less neurons in the hidden layers
  - Evaluate a larger network with more layers
    - Try reducing the size of each layer as you progress – form a funnel





# tensorflow

# Why TensorFlow?

- It's not specifically for neural networks— it's more generally an architecture for executing a graph of numerical operations
- Tensorflow can optimize the processing of that graph, and distribute its processing across a network
  - Sounds a lot like Apache Spark, eh?



It can also distribute work across GPU's!

# TensorFlow basics

- Install with `conda install TensorFlow` or  
`conda install tensorflow-gpu`
- A tensor is just a fancy name for an array or matrix of values
- To use TensorFlow, you:
  1. Construct a graph to compute your tensors
  2. Initialize your variables
  3. Execute that graph – nothing actually happens until then



World's simplest Tensorflow app:

```
import tensorflow as tf

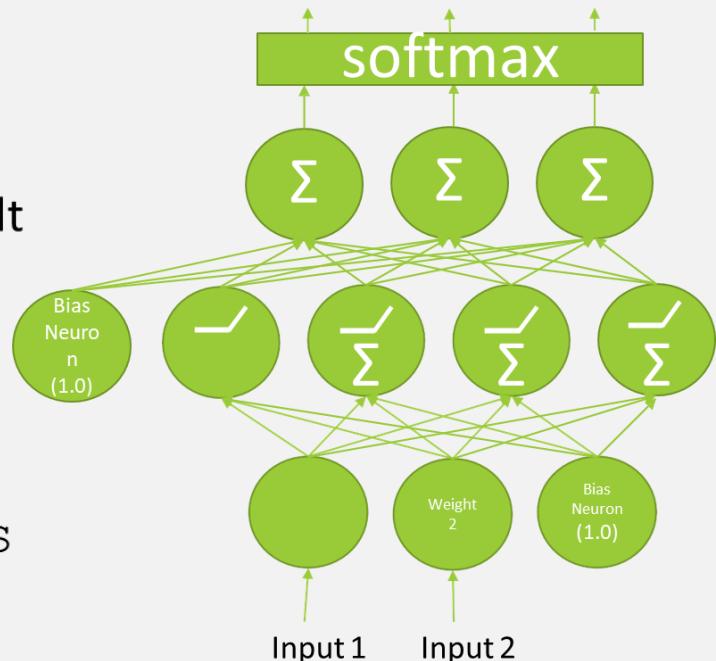
a = tf.Variable(1, name="a")
b = tf.Variable(2, name="b")
f = a + b

init = tf.global_variables_initializer()
with tf.Session() as s:
    init.run()
    print( f.eval() )
```

# Creating a neural network with TensorFlow

- Mathematical insights:
  - All those interconnected arrows multiplying weights can be thought of as a big matrix multiplication
  - The bias term can just be added onto the result of that matrix multiplication
- So in Tensorflow, we can define a layer of a neural network as:

```
output =  
tf.matmul(previous_layer,  
layer_weights) + layer_biases
```
- By using Tensorflow directly we're kinda doing this the "hard way."



# Creating a neural network with TensorFlow

- Load up our training and testing data
  - Construct a graph describing our neural network
- Use **placeholders** for the input data and target labels
  - This way we can use the same graph for training and testing!
- Use **variables** for the learned weights for each connection and learned biases for each neuron
  - Variables are preserved across runs within a Tensorflow

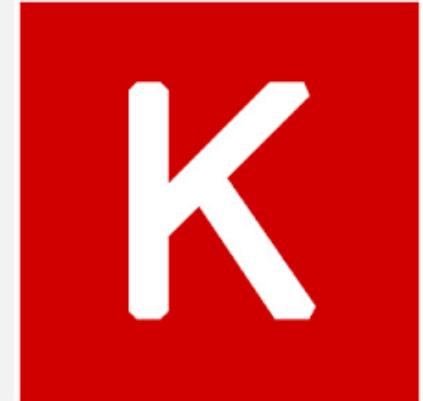


# Make sure your features are normalized

- Neural networks usually work best if your input data is normalized.
  - That is, 0 mean and unit variance
  - The real goal is that every input feature is comparable in terms of magnitude
- scikit\_learn's StandardScaler can do this for you
- Many data sets are normalized to begin with – such as the one we're about to use.

# Let's try it out





# keras



# Why keras?



- Easy and fast prototyping
  - Runs on top of TensorFlow (or CNTK, or Theano)
  - scikit\_learn integration
  - Less to think about – which often yields better results without even trying
  - This is really important! The faster you can experiment, the better your results.



# let's dive in



# Example: multi-class classification

- MNIST is an example of multi-class classification.

```
model = Sequential()
```

```
model.add(Dense(64, activation='relu', input_dim=20))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9,
           nesterov=True)
model.compile(loss='categorical_crossentropy',
               optimizer=sgd, metrics=['accuracy'])
```

# Example: binary classification

```
model = Sequential()
model.add(Dense(64, input_dim=20,
activation='relu')) model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy',
optimizer='rmsprop', metrics=['accuracy'])
```

# Integrating keras with scikit-learn

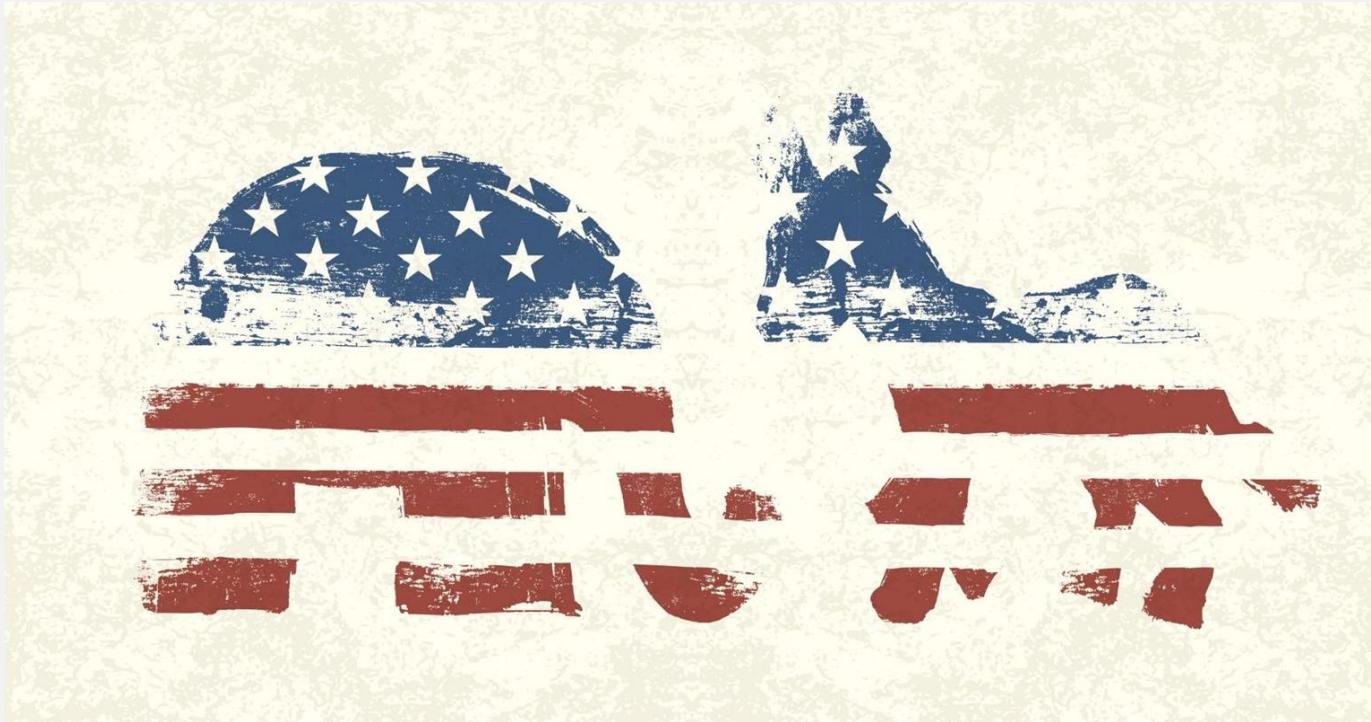
```
from keras.wrappers.scikit_learn import KerasClassifier

def create_model():
    model = Sequential()
    model.add(Dense(6, input_dim=4, kernel_initializer='normal', activation='relu'))
    model.add(Dense(4, kernel_initializer='normal', activation='relu'))
    model.add(Dense(1, kernel_initializer='normal', activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='rmsprop', metrics=['accuracy'])
    return model

estimator = KerasClassifier(build_fn=create_model, nb_epoch=100, verbose=0)

cv_scores = cross_val_score(estimator, features, labels, cv=10)
print(cv_scores.mean())
```

# let's try it out



# convolutional neural networks

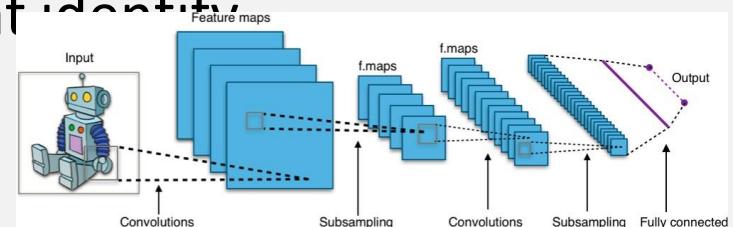
# cnn's: what are they for?

- When you have data that doesn't neatly align into columns
  - Images that you want to find features within
  - Machine translation
  - Sentence classification
  - Sentiment analysis



# cnn's: how do they work?

- Inspired by the biology of the visual cortex
  - Local receptive fields are groups of neurons that only respond to a part of what your eyes see (subsampling)
  - They overlap each other to cover the entire visual field (convolutions)
  - They feed into higher layers that identify increasingly complex images



# How do we “know” that’s a stop sign?

- Individual local receptive fields scan the image looking for edges, and pick up the edges of the stop sign in a layer
- Those edges in turn get picked up by a higher level convolution that identifies the stop sign’s shape (and letters, too)



# cnn's with keras

- Source data must be of appropriate dimensions
  - ie width x length x color channels
- Conv2D layer type does the actual convolution on a 2D image
  - Conv1D and Conv3D also available – doesn't have to be image data
- MaxPooling2D layers can be used to reduce a 2D layer down by taking the maximum value in a given block

# cnn's are hard

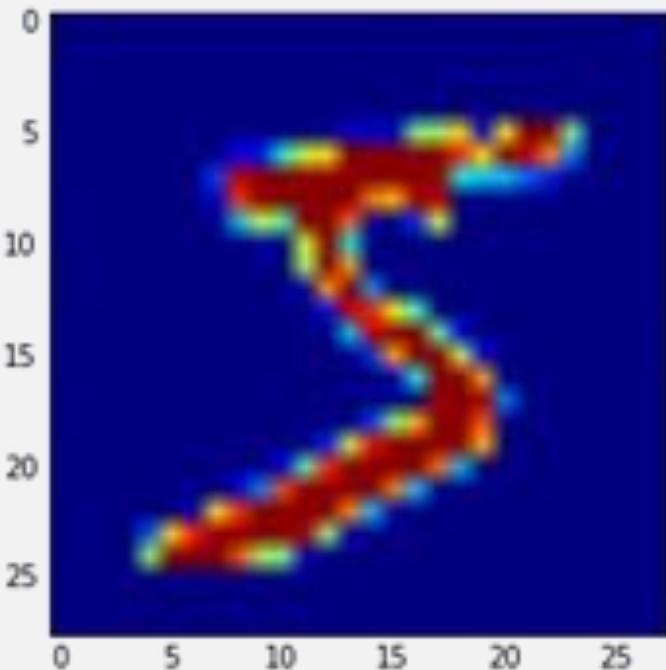
- Very resource-intensive (CPU, GPU, and RAM)
- Lots of hyperparameters
  - Kernel sizes, many layers with different numbers of units, amount of pooling... in addition to the usual stuff like number of layers, choice of optimizer



# Specialized cnn architectures

- Defines specific arrangement of layers, padding, and hyperparameters
- LeNet-5
  - Good for handwriting recognition
- AlexNet
  - Image classification, deeper than LeNet
- GoogLeNet
  - Even deeper, but with better performance
  - Introduces *inception modules* (groups of convolution layers)
- ResNet (Residual Network)
  - Even deeper – maintains performance via *skip connections*.

let's try it out



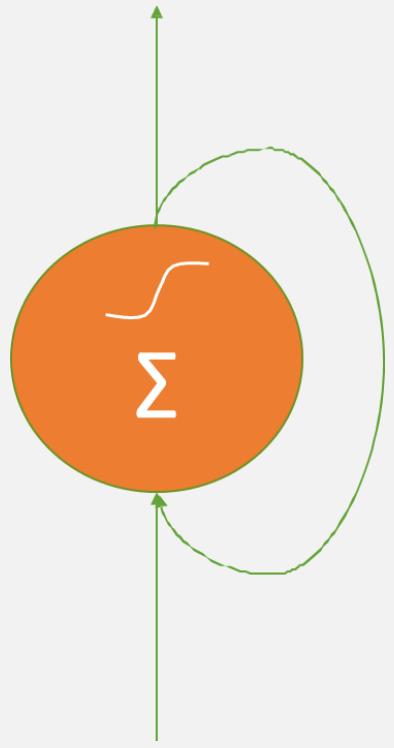
# recurrent neural networks

# rnn's: what are they for?

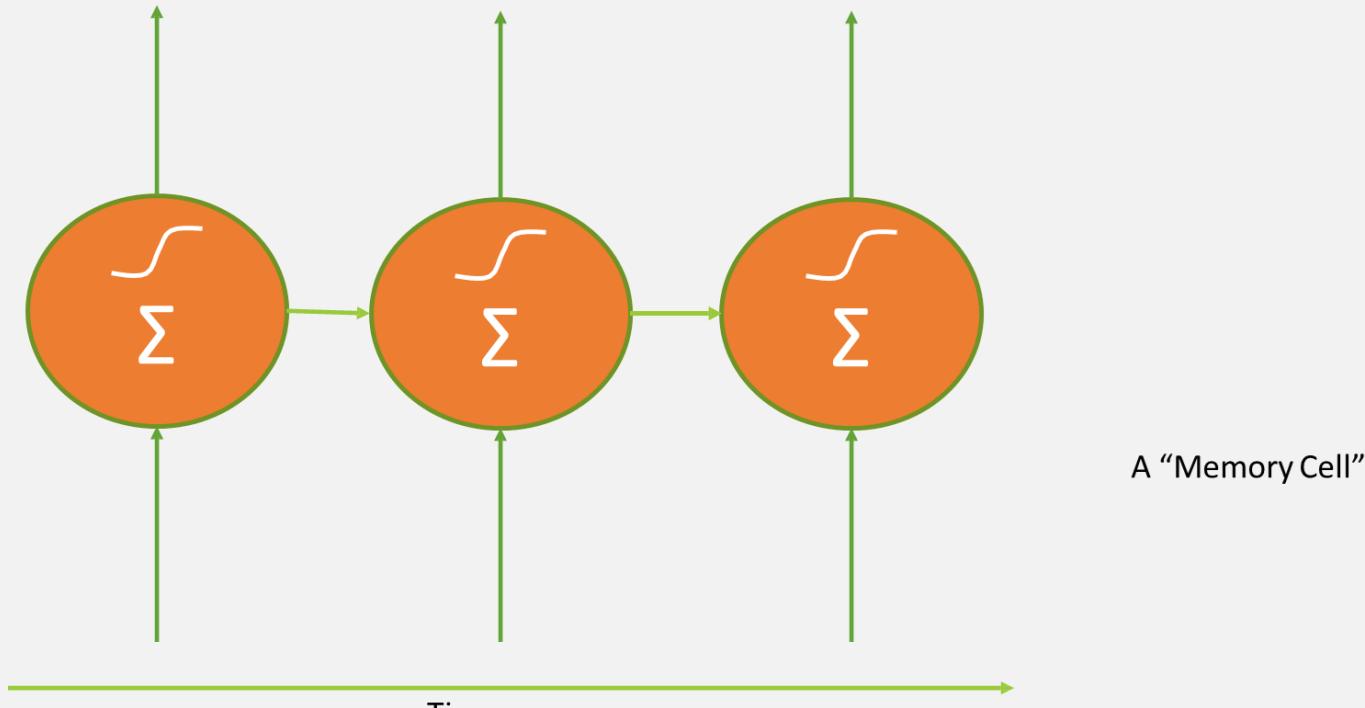
- Time-series data
  - When you want to predict future behavior based on past behavior
  - Web logs, sensor logs, stock trades
  - Where to drive your self-driving car based on past trajectories
- Data that consists of sequences of arbitrary length
  - Machine translation
  - Image captions
  - Machine-generated music



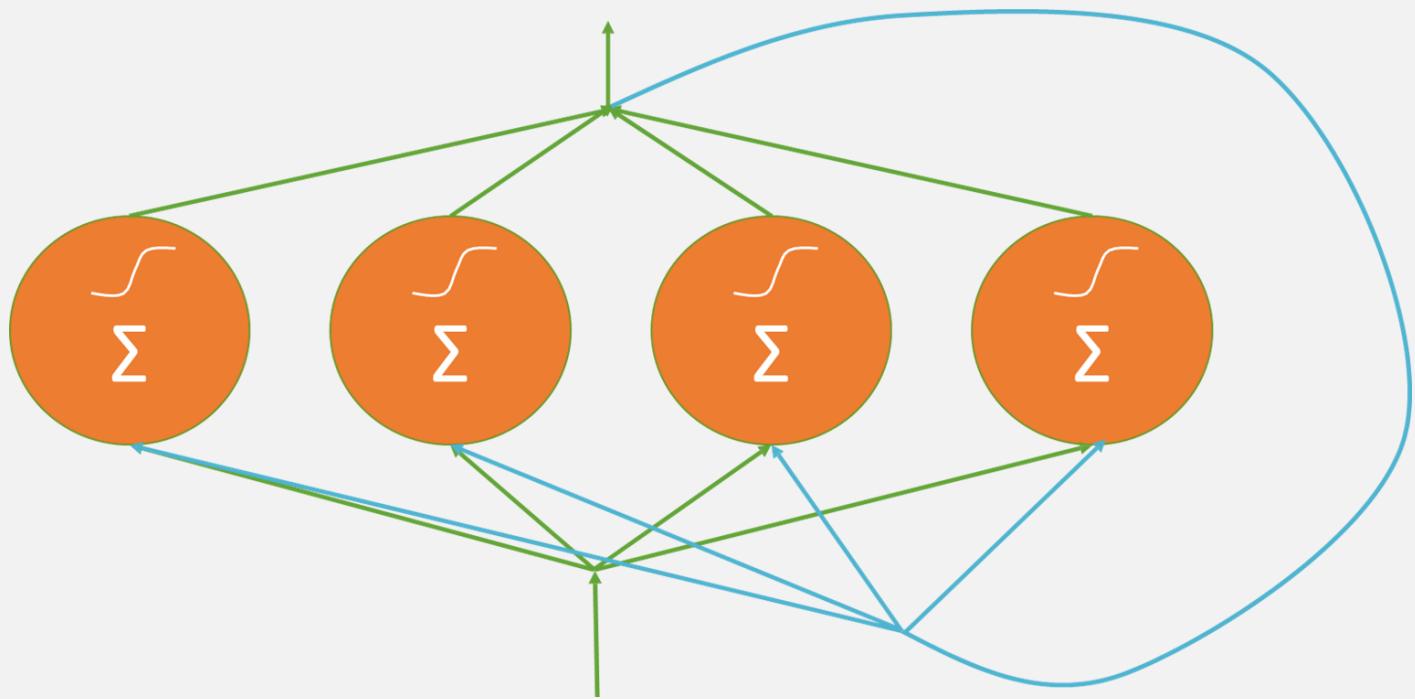
# A recurrent neuron



# Another way to look at it



# A layer of recurrent neurons



# rnn topologies



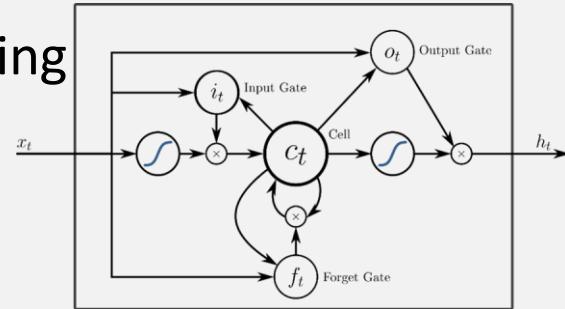
- Sequence to sequence
  - i.e., predict stock prices based on series of historical data
- Sequence to vector
  - i.e., words in a sentence to sentiment
- Vector to sequence
  - i.e., create captions from an image
- Encoder -> Decoder
  - Sequence -> vector -> sequence
  - i.e., machine translation

# training rnn's

- Backpropagation through time
  - Just like backpropagation on MLP's, but applied to each time step.
- All those time steps add up fast
  - Ends up looking like a really, really deep neural network.
  - Can limit backpropagation to a limited number of time steps (truncated backpropagation through time)

# training rnn's

- State from earlier time steps get diluted over time
  - This can be a problem, for example when learning sentence structures
- LSTM Cell
  - Long Short-Term Memory Cell
  - Maintains separate short-term and long-term states
- GRU Cell
  - Gated Recurrent Unit
  - Simplified LSTM Cell that performs about as well



# training rnn's

- It's really hard
  - Very sensitive to topologies, choice of hyperparameters
  - Very resource intensive
  - A wrong choice can lead to a RNN that doesn't converge at all.



# let's run an example

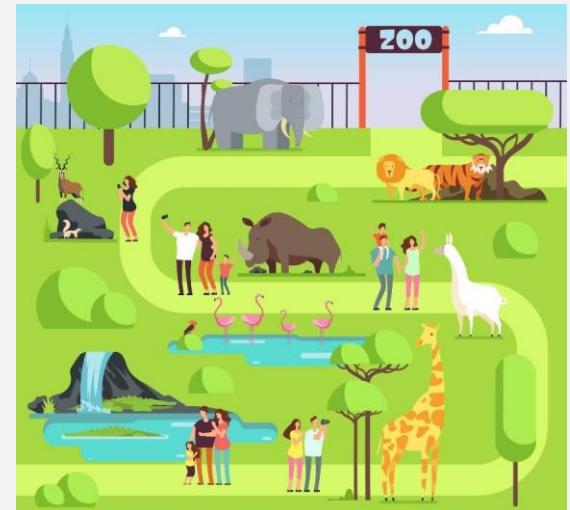


# Transfer Learning



# Re-using trained models

- For many common problems, you can import pre-trained models and just use them.
  - Image classification (ResNet, Inception, MobileNet, Oxford VGG)
  - NLP (word2vec, GloVe)
- Use them as-is, or tune them for your application
- Model Zoos
  - Caffe Model Zoo



# Tuning Neural Networks



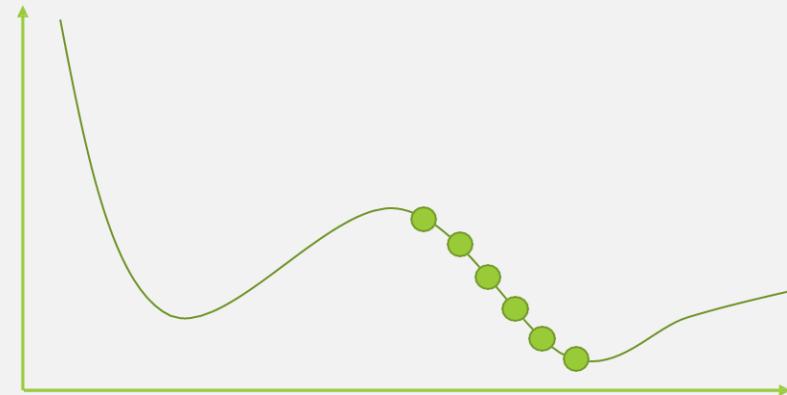
# Learning Rate

- Neural networks are trained by gradient descent (or similar means)
- We start at some random point, and sample different solutions (weights) seeking to minimize some cost function, over many *epochs*



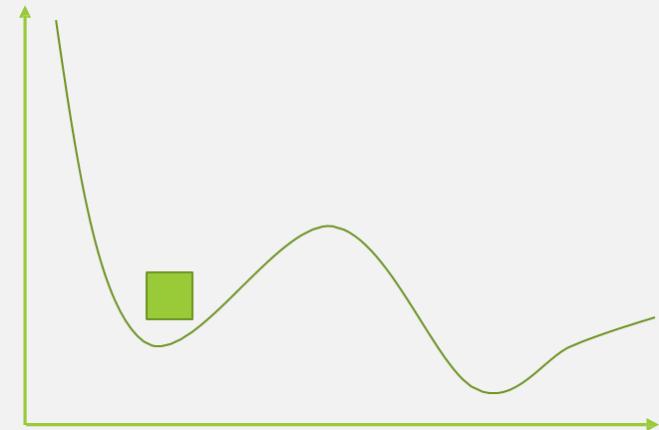
# Effect of learning rate

- Too high a learning rate means you might overshoot the optimal solution!
- Too small a learning rate will take too long to find the optimal solution
- Learning rate is an example of a *hyperparameter*



# Batch Size

- How many training samples are used within each epoch
- Somewhat counter-intuitively:
  - Smaller batch sizes can work their way out of “local minima” more easily
  - Batch sizes that are too large can end up getting stuck in the wrong solution
  - Random shuffling at each epoch can make this look like very inconsistent results from run to run



# To Recap

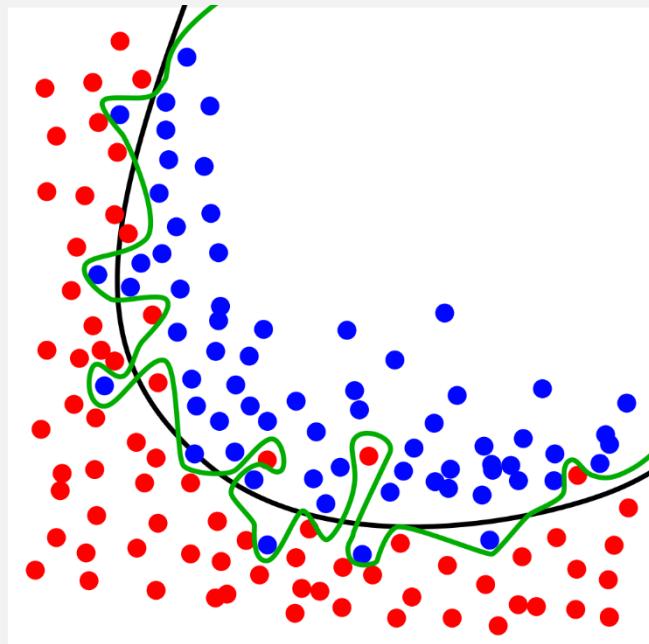
- Small batch sizes tend to not get stuck in local minima
- Large batch sizes can converge on the wrong solution at random
- Large learning rates can overshoot the correct solution
- Small learning rates increase training time

# Neural Network Regularization Techniques

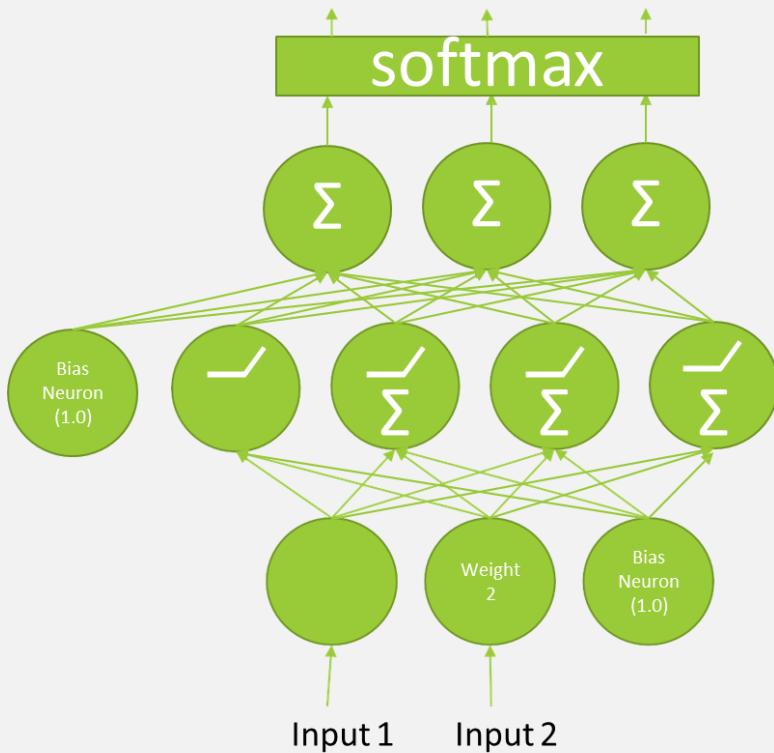


# What is regularization?

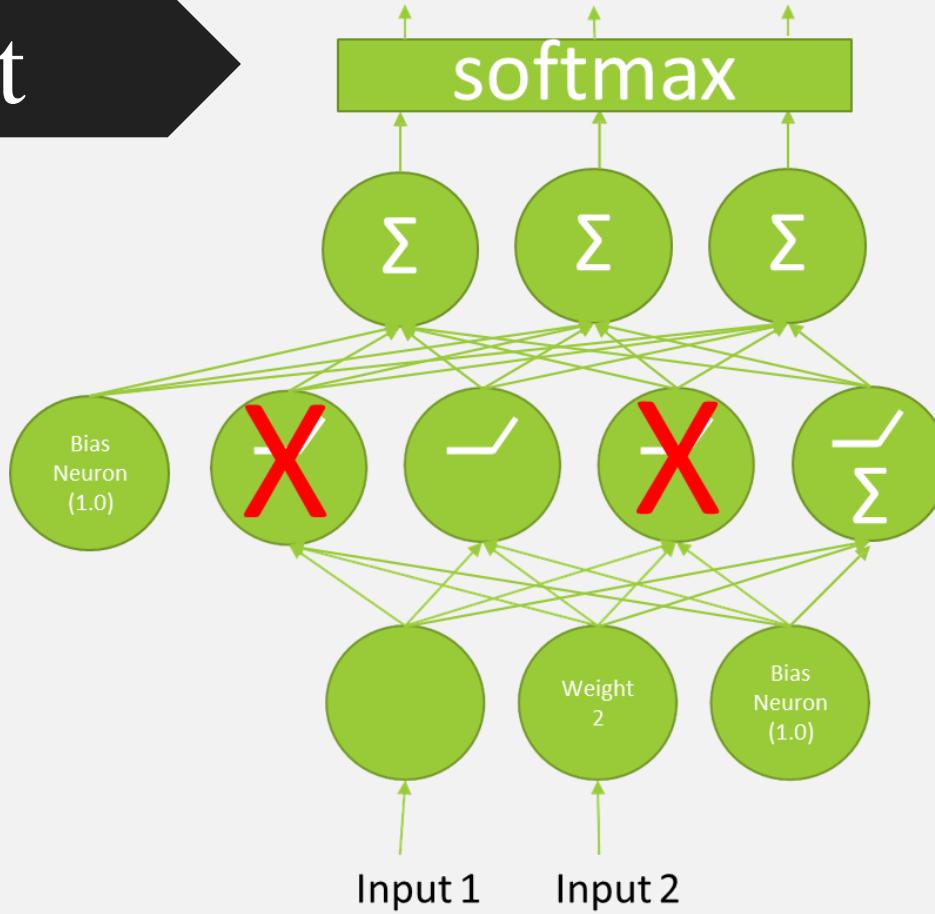
- Preventing *overfitting*
  - Models that are good at making predictions on the data they were trained on, but not on new data it hasn't seen before
  - Overfitted models have learned patterns in the training data that don't generalize to the real world
  - Often seen as high accuracy on training data set, but lower accuracy on test or evaluation data set.
    - When training and evaluating a model, we use *training*, *evaluation*, and *testing* data sets.



# Too many layers? Too many neurons?



# Dropout



# Early Stopping

```
Epoch 1/10
- 4s - loss: 0.2406 - acc: 0.9302 - val_loss: 0.1437 - val_acc: 0.9557
Epoch 2/10
- 2s - loss: 0.0971 - acc: 0.9712 - val_loss: 0.0900 - val_acc: 0.9725
Epoch 3/10
- 2s - loss: 0.0653 - acc: 0.9803 - val_loss: 0.0725 - val_acc: 0.9786
Epoch 4/10
- 2s - loss: 0.0471 - acc: 0.9860 - val_loss: 0.0689 - val_acc: 0.9795
Epoch 5/10
- 2s - loss: 0.0367 - acc: 0.9890 - val_loss: 0.0675 - val_acc: 0.9808
Epoch 6/10
- 2s - loss: 0.0266 - acc: 0.9919 - val_loss: 0.0680 - val_acc: 0.9796
Epoch 7/10
- 2s - loss: 0.0208 - acc: 0.9937 - val_loss: 0.0678 - val_acc: 0.9811
Epoch 8/10
- 2s - loss: 0.0157 - acc: 0.9953 - val_loss: 0.0719 - val_acc: 0.9810
Epoch 9/10
- 2s - loss: 0.0130 - acc: 0.9960 - val_loss: 0.0707 - val_acc: 0.9825
Epoch 10/10
- 2s - loss: 0.0097 - acc: 0.9972 - val_loss: 0.0807 - val_acc: 0.9805
```

# The Ethics of Deep Learning



# Types of errors



- Accuracy doesn't tell the whole story
- Type 1: False positive
  - Unnecessary surgery
  - Slam on the brakes for no reason
- Type 2: False negative
  - Untreated conditions
  - You crash into the car in front of you

# Hidden biases



- Just because your model isn't human doesn't mean it's inherently fair
- Example: train a model on what sort of job applicants get hired, use it to screen resumes
  - Past biases toward gender / age / race will be reflected in your model, because it was reflected in the data you trained the model with.

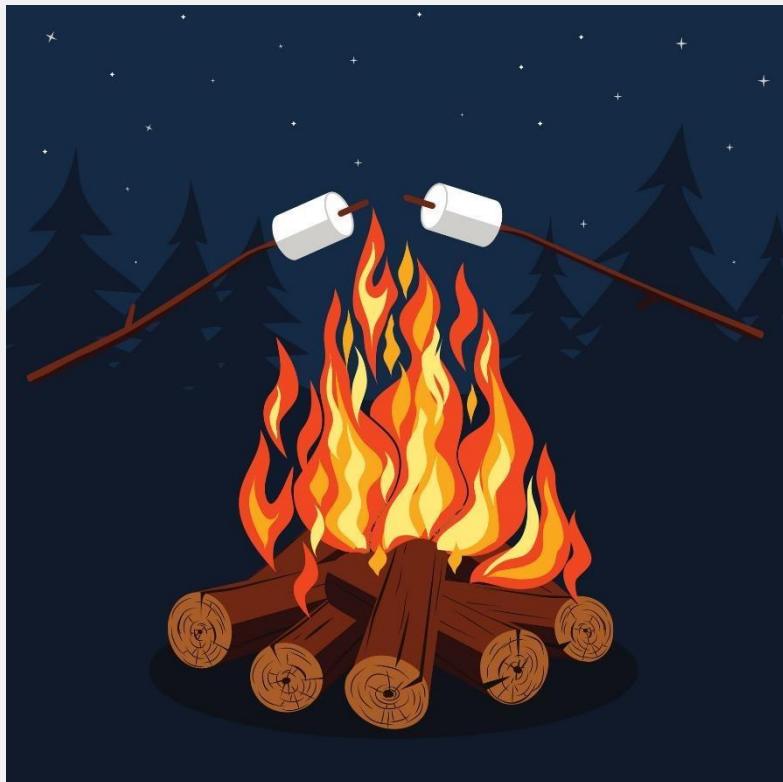
# Is it really better than a human?

- Don't oversell the capabilities of an algorithm in your excitement
- Example: medical diagnostics that are almost, but not quite, as good as a human doctor
- Another example: self-driving cars that can kill people



# Unintended applications of your research

- Gather 'round the fire while Uncle E! tells you a story.



# Learning More about Deep Learning



# Learning more



# Final Project

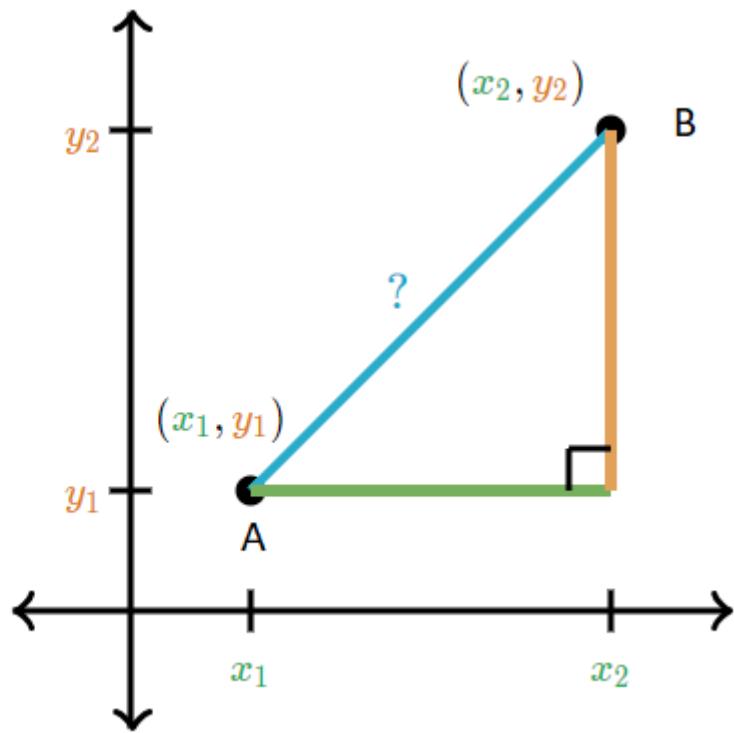
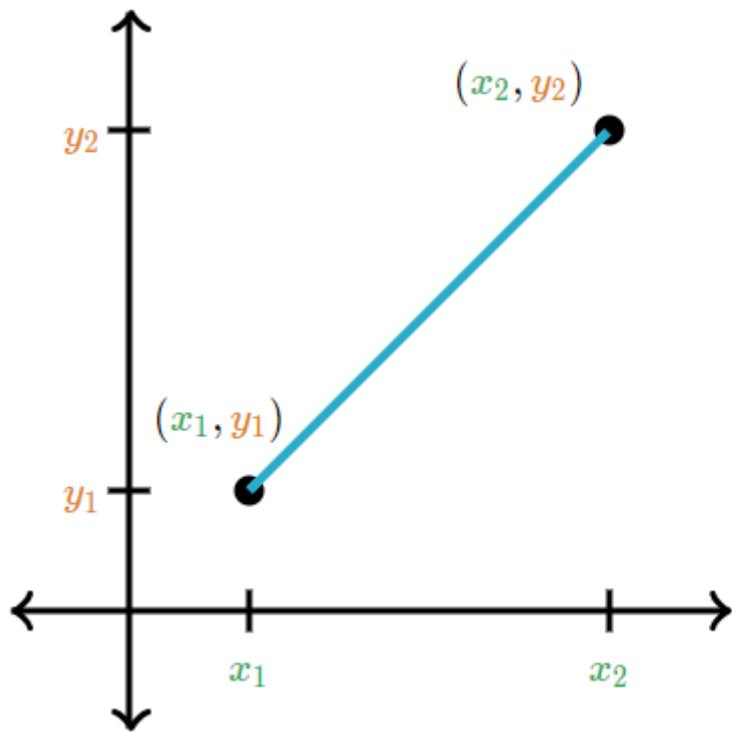


# Your Assignment

- Predict if a mass detected in a mammogram is benign or malignant, using the best supervised machine learning model you can find.



# Euclidean



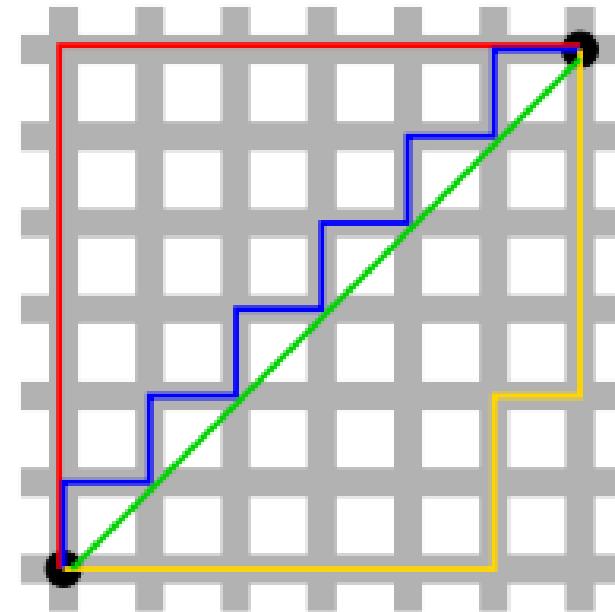
$$? = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

# Minkowski Distance

$$\left( \sum_{i=1}^n |x_i - y_i|^p \right)^{1/p}$$

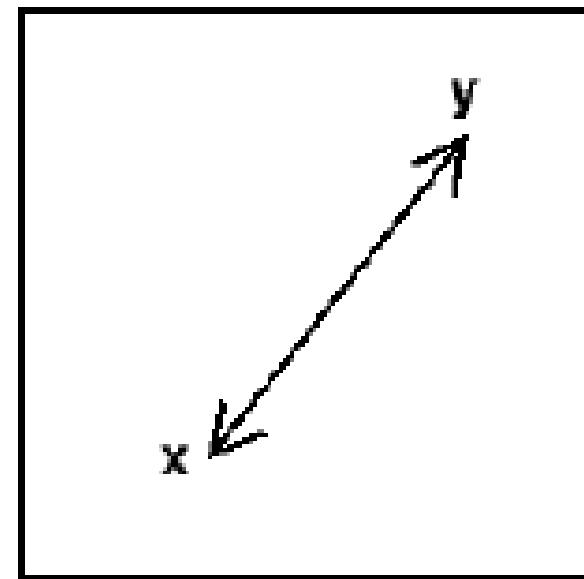
# Minkowski Distance ( $p=1$ ) --> Manhattan Distance

$$\left( \sum_{i=1}^n |x_i - y_i|^p \right)^{1/p}$$



# Minkowski Distance (p=2) --> Euclidean Distance

$$\left( \sum_{i=1}^n |x_i - y_i|^p \right)^{1/p}$$



## Euclidean

# Cosine Distance

$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \|\vec{b}\| \cos \theta$$

$$\cos \theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}$$

$$\begin{aligned}\cos 0' &= 1 & \cos 90' &= 0 \\ \cos 180' &= -1\end{aligned}$$

# Mahalanobis Distance

- *The **Mahalanobis distance** is a measure of the distance between a point  $P$  and a distribution  $D$ . The idea of measuring is, how many standard deviations away  $P$  is from the mean of  $D$ .*

$$D_M(\vec{x}) = \sqrt{(\vec{x} - \vec{\mu})^T S^{-1} (\vec{x} - \vec{\mu})}.$$

