



# Classification:

## A machine learning perspective

Emily Fox & Carlos Guestrin  
Machine Learning Specialization  
University of Washington



# Part of a specialization

---

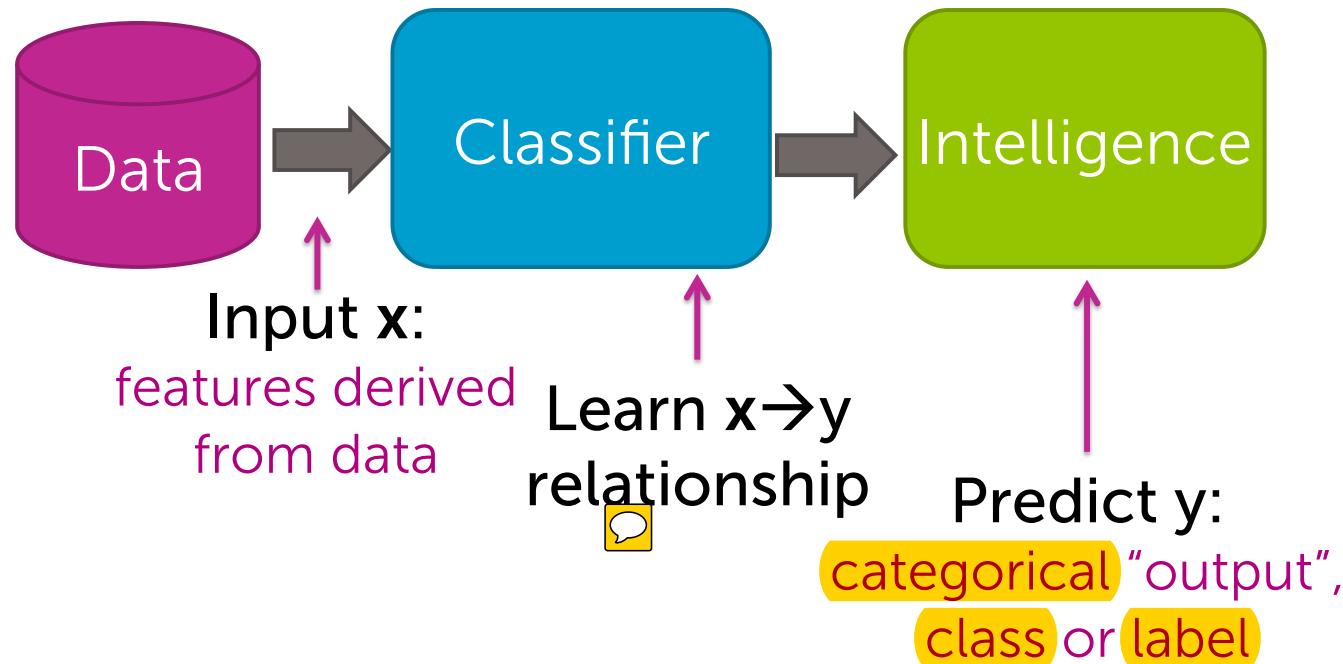
# This course is a part of the Machine Learning Specialization



# What is the course about?

# What is classification?

From features to predictions



# Sentiment classifier



Input  $x$ : Easily best sushi in Seattle.

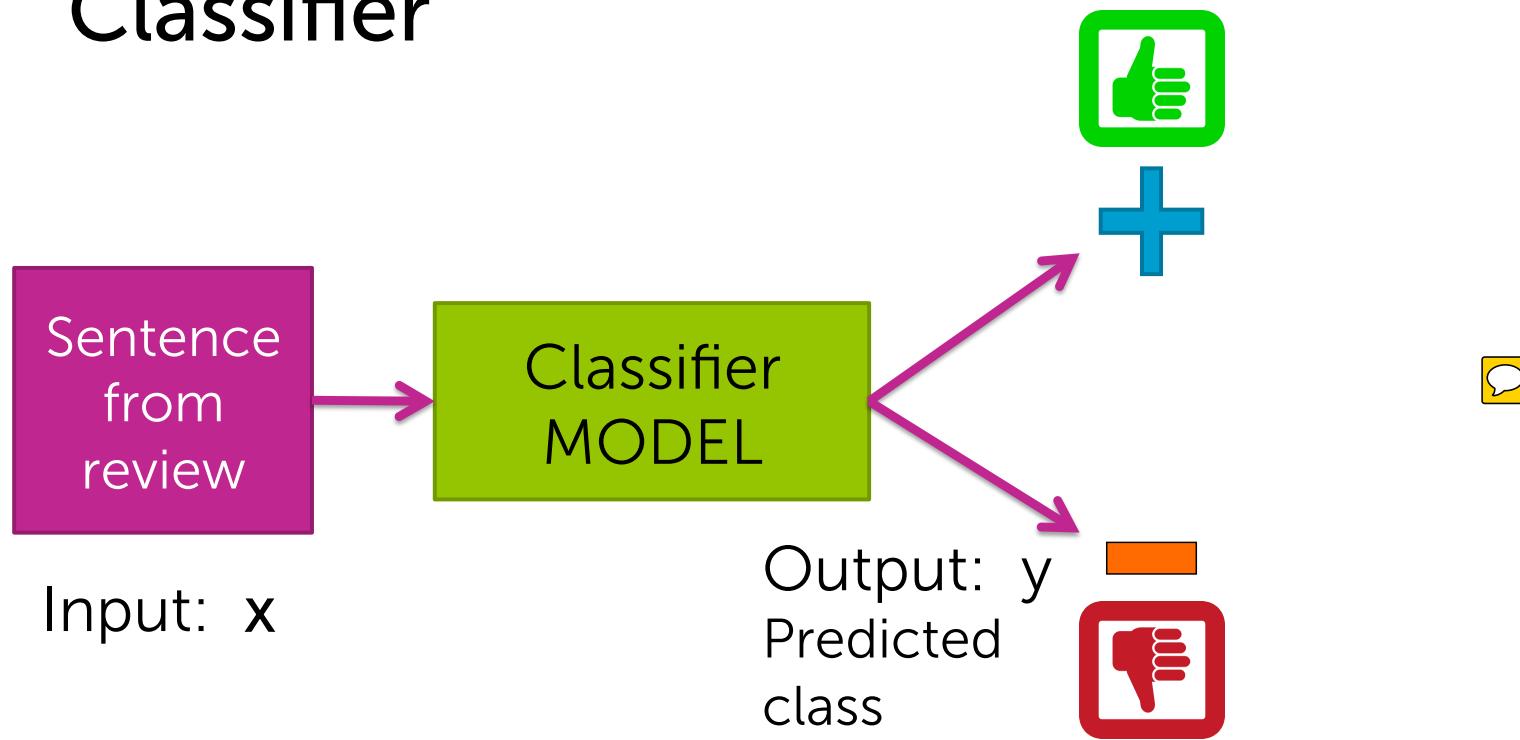


Sentence Sentiment  
Classifier

Output:  $y$   
Sentiment

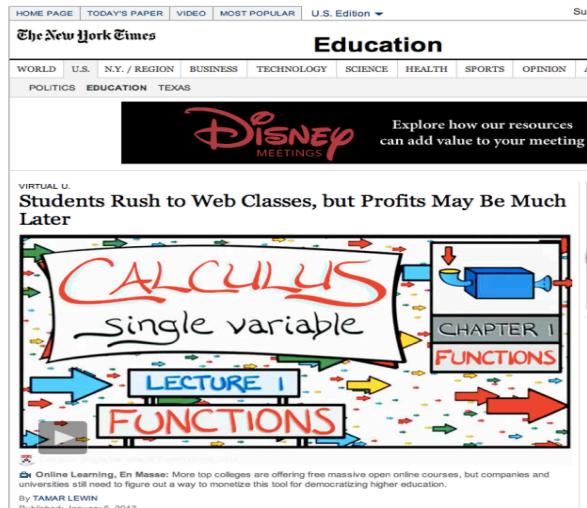


# Classifier

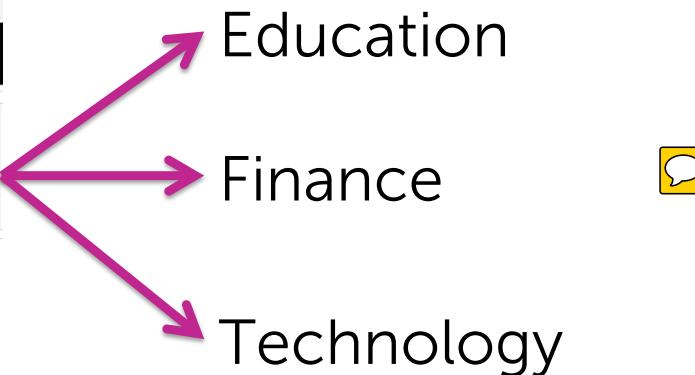


# Example multiclass classifier

*Output  $y$  has more than 2 categories*

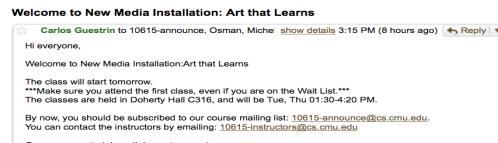


Input:  $x$   
Webpage



Output:  $y$

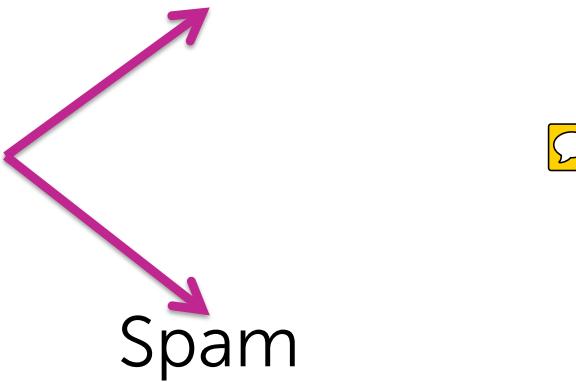
# Spam filtering



Input:  $x$

Text of email,  
sender, IP,...

Not spam



Output:  $y$

# Image classification

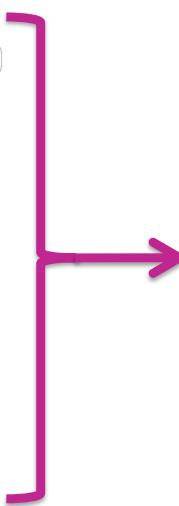
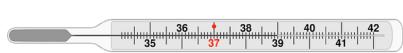


Input:  $x$   
Image pixels

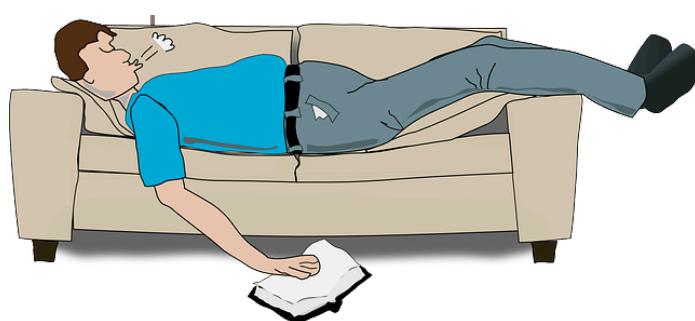
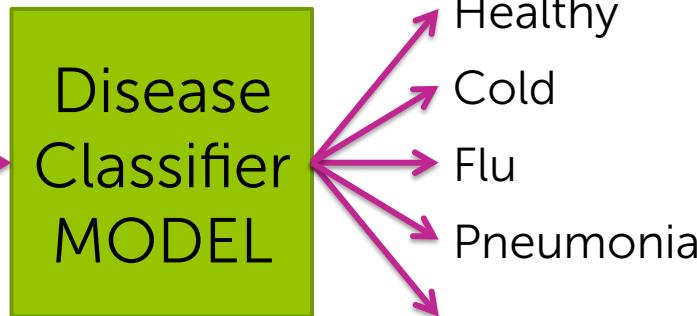
Output:  $y$   
Predicted object

# Personalized medical diagnosis

Input:  $x$

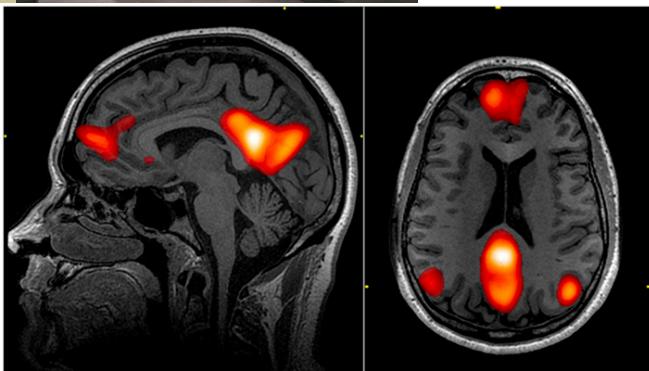


Output:  $y$



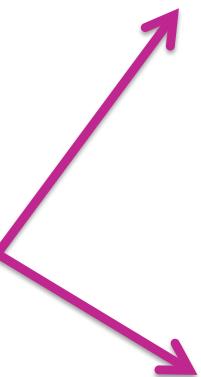
# Reading your mind

Inputs  $x$  are  
brain region  
intensities



Output  $y$

“Hammer”



“House”

# Impact of classification



# Impact of classification



# Course overview

---

# Course philosophy: Always use case studies & ...

Core  
concept

Visual

Algorithm

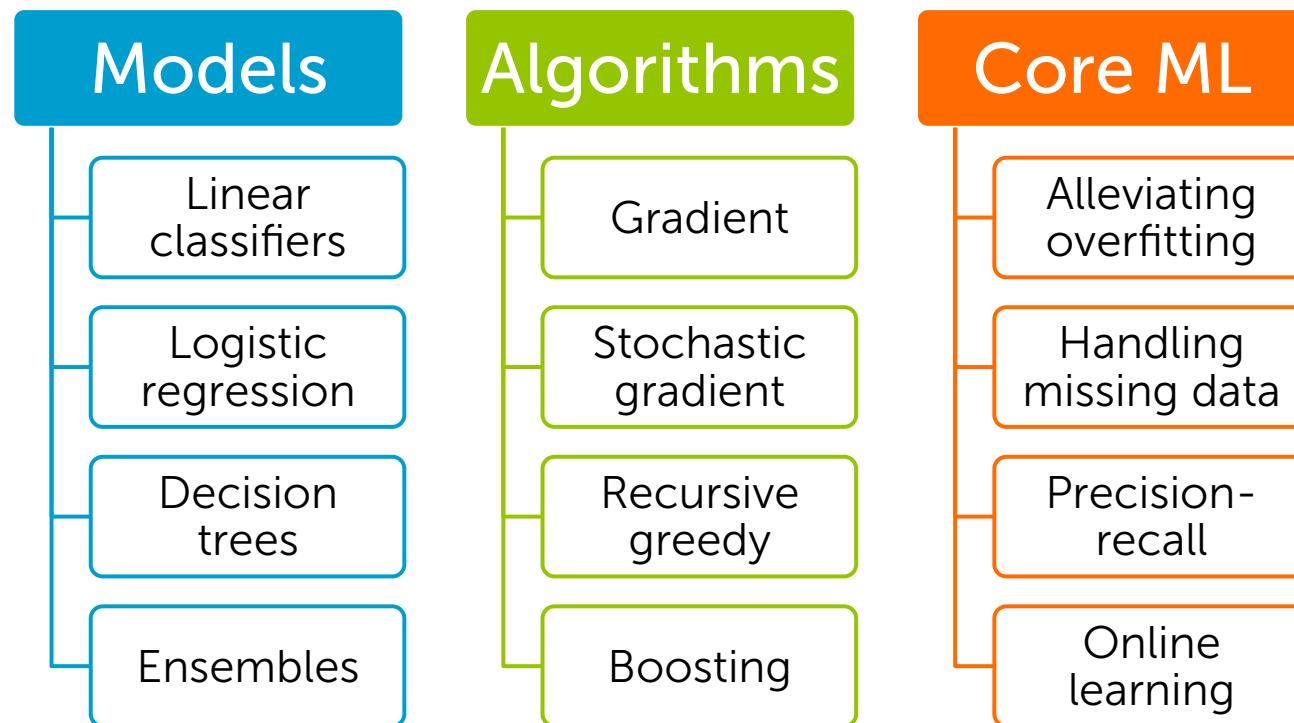
Practical

Implement

Advanced  
topics

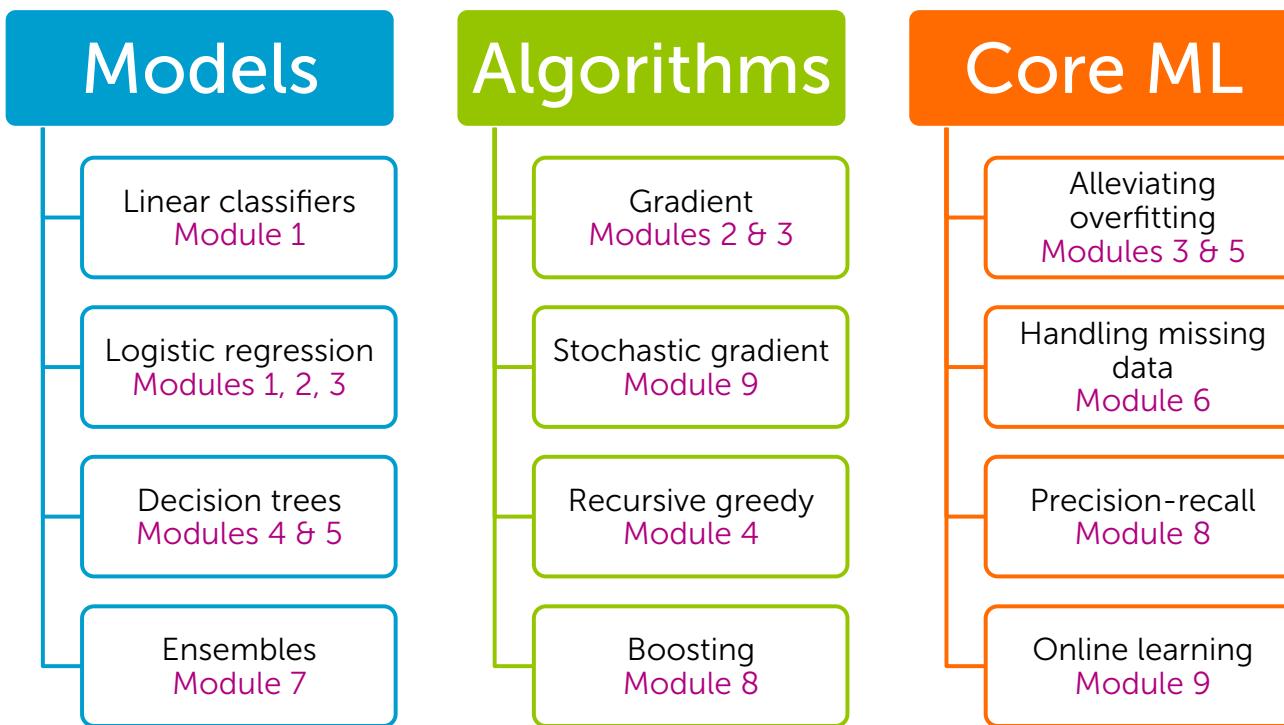
OPTIONAL

# Overview of content



# Course outline

# Overview of modules

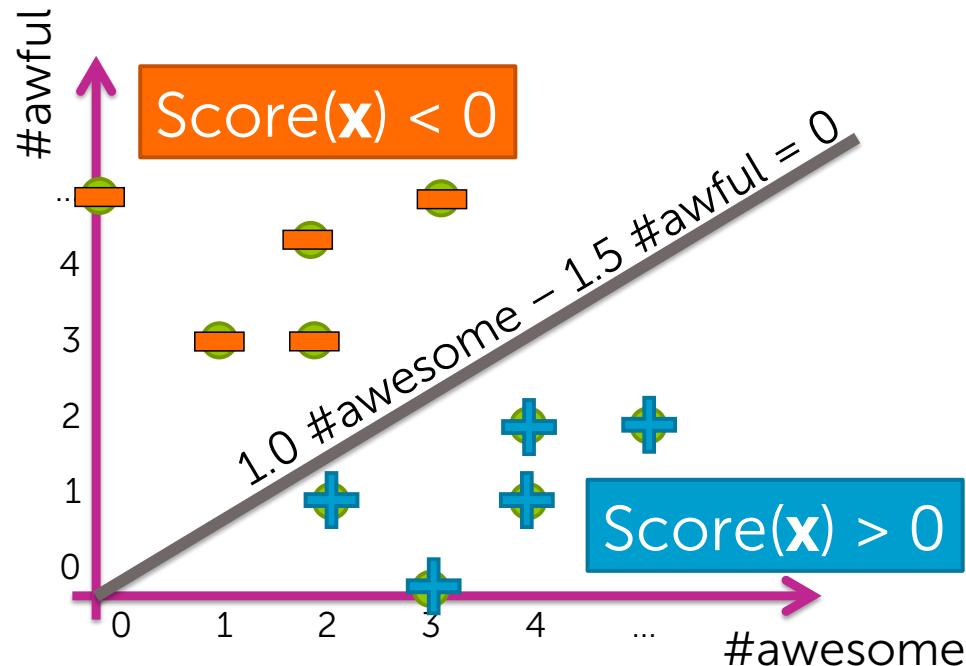


# Module 1: Linear classifiers

Word	Coefficient
#awesome	1.0
#awful	-1.5

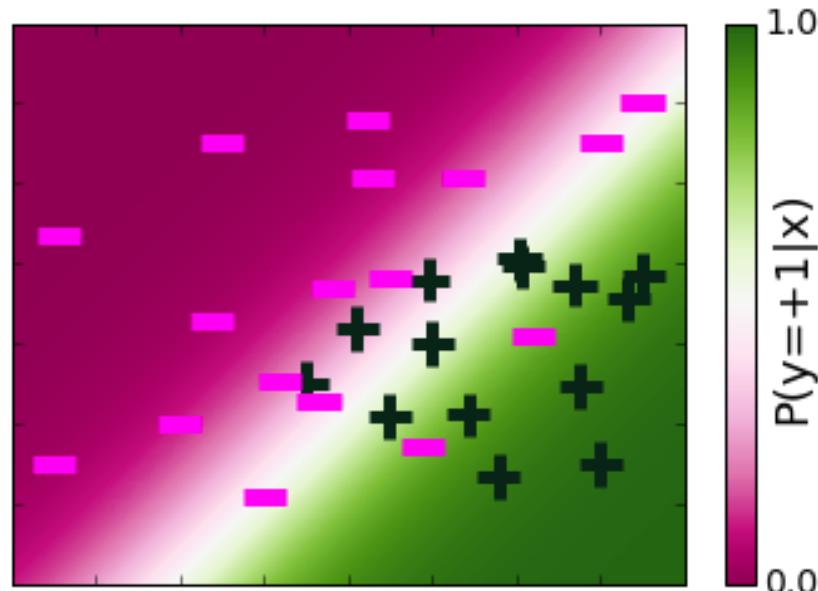


$$\text{Score}(x) = 1.0 \# \text{awesome} - 1.5 \# \text{awful}$$



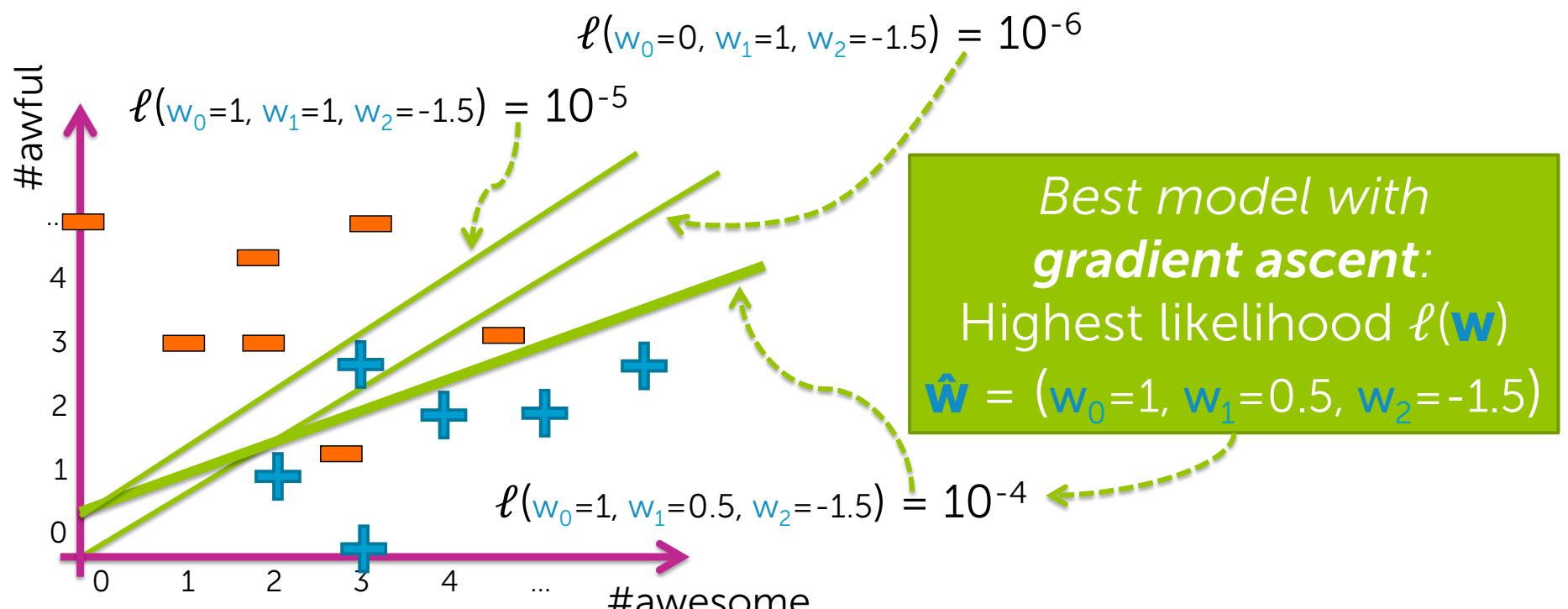
## Module 1: Logistic regression represents probabilities

$$\hat{P}(y=+1|x, \hat{w}) = \frac{1}{1 + e^{-\hat{w}^T h(x)}}$$

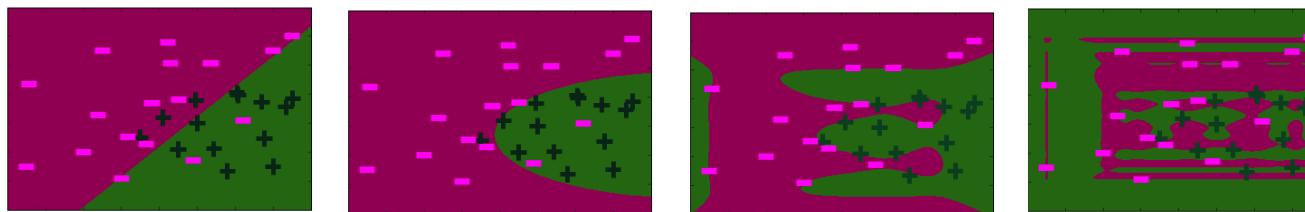
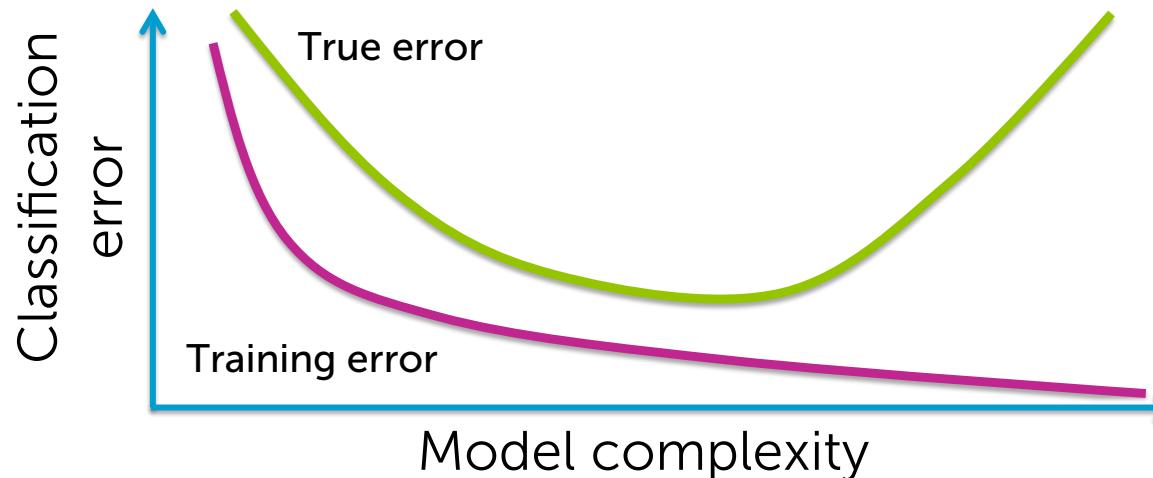


## Module 2: Learning “best” classifier

Maximize likelihood over all possible  $w_0, w_1, w_2$

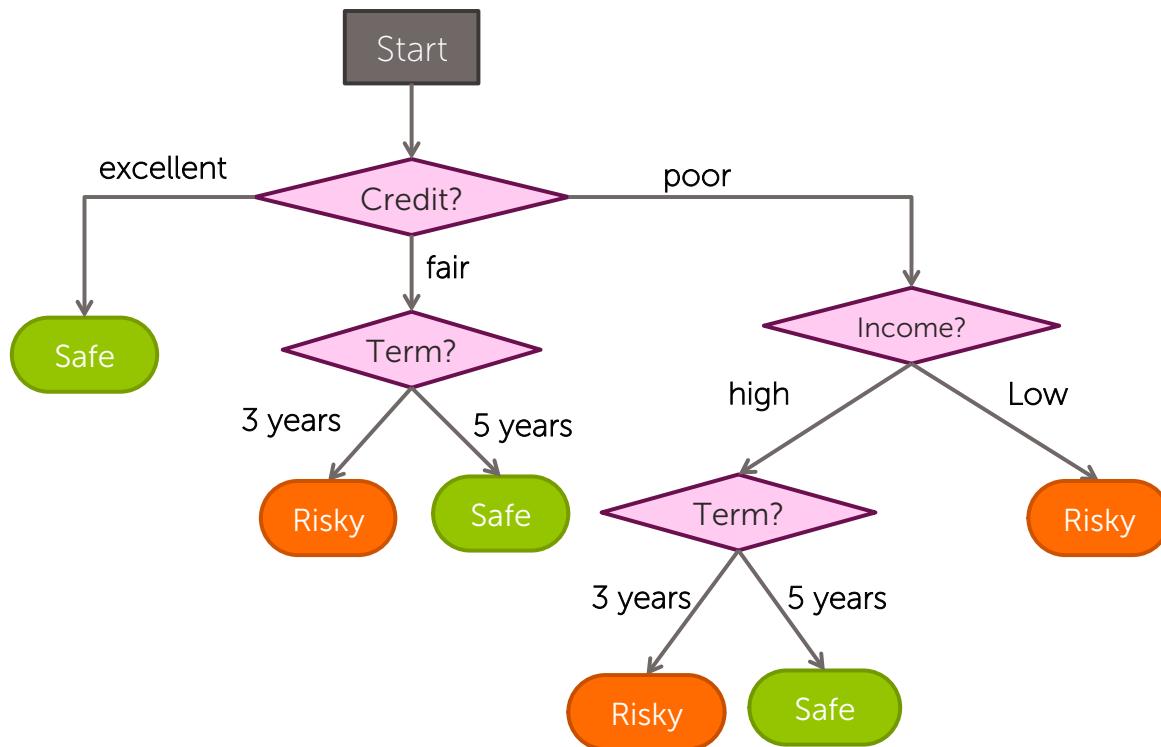


# Module 3: Overfitting & regularization



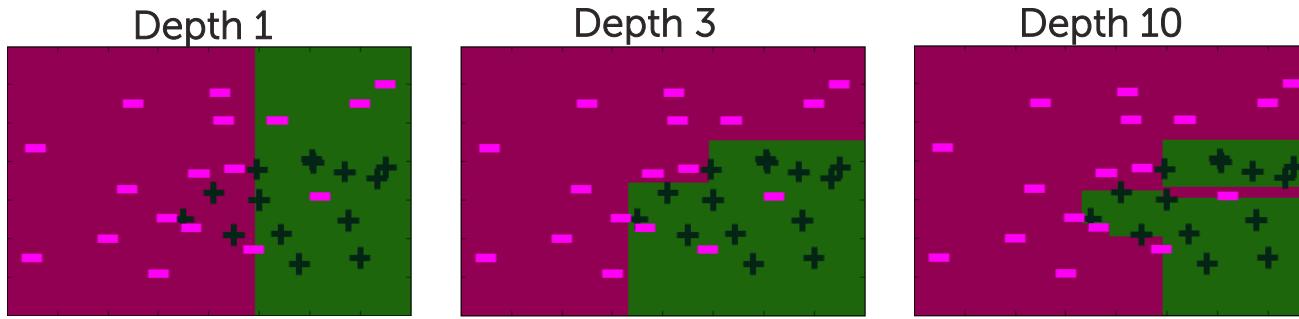
Use regularization penalty  $\ell(\mathbf{w}) - \lambda \|\mathbf{w}\|_2^2$  to mitigate overfitting

# Module 4: Decision trees

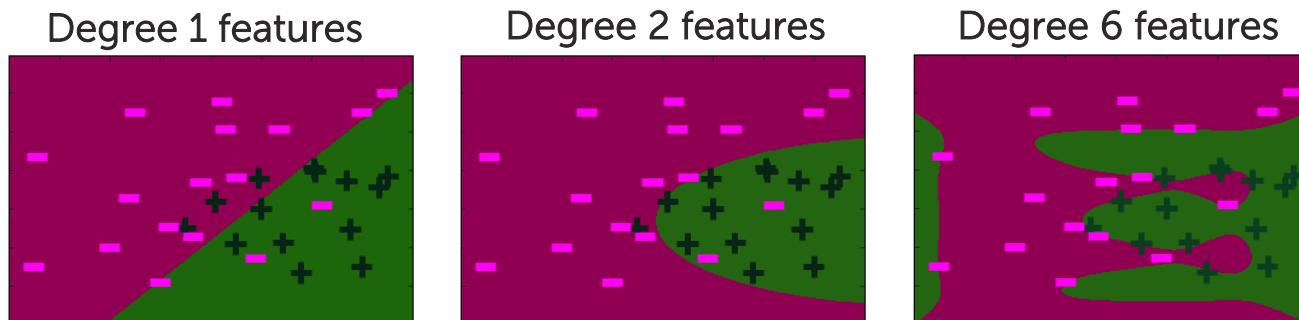


# Module 5: Overfitting in decision trees

## Decision Tree



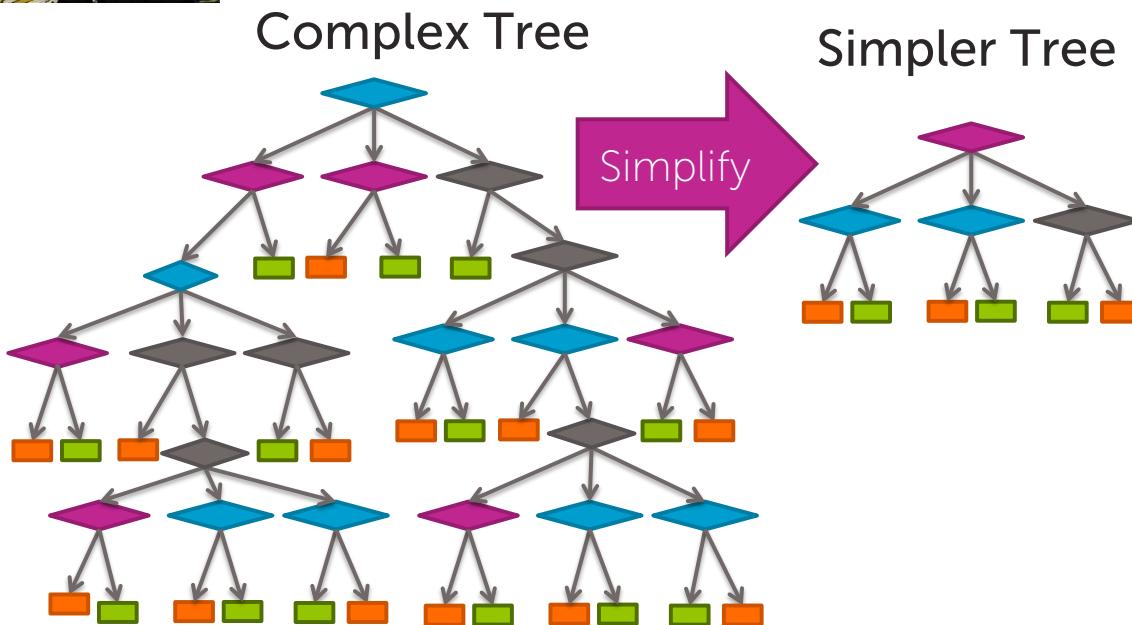
## Logistic Regression



## Module 5: Alleviate overfitting by learning simpler trees

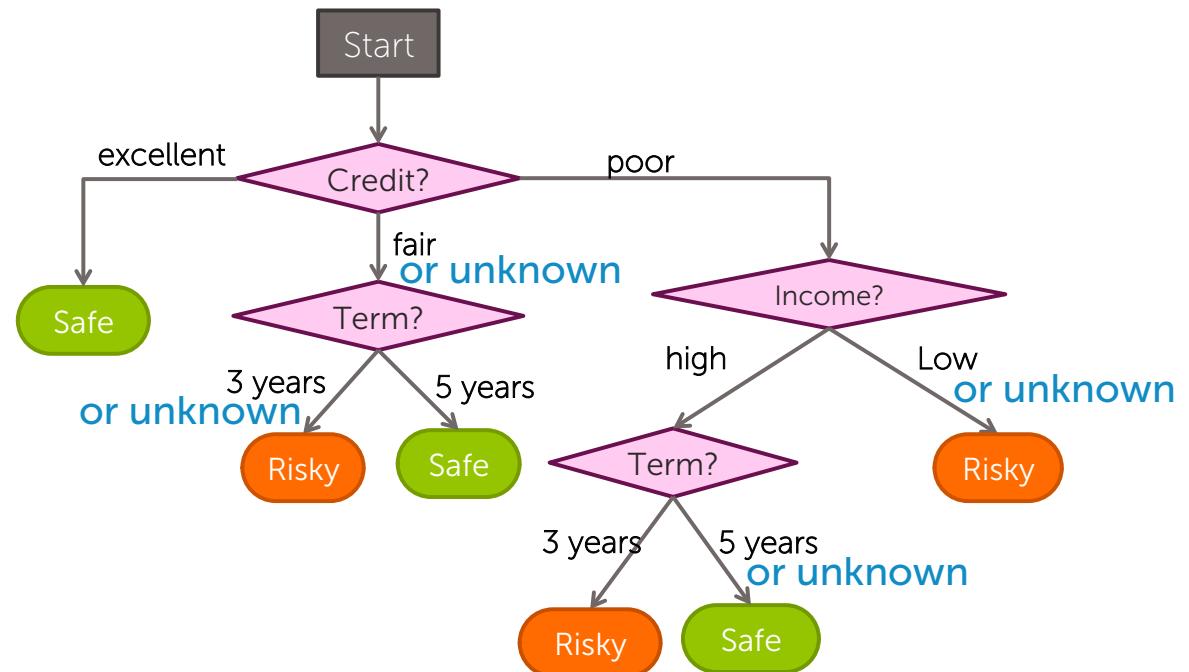


Occam's Razor: "Among competing hypotheses,  
the one with fewest assumptions should be  
selected", William of Occam, 13<sup>th</sup> Century



# Module 6: Handling missing data

Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	?	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	high	safe
poor	?	high	risky
poor	5 yrs	low	safe
fair	?	high	safe



# Module 7: Boosting question

"Can a set of weak learners be combined to create a stronger learner?" *Kearns and Valiant (1988)*



Yes! *Schapire (1990)*

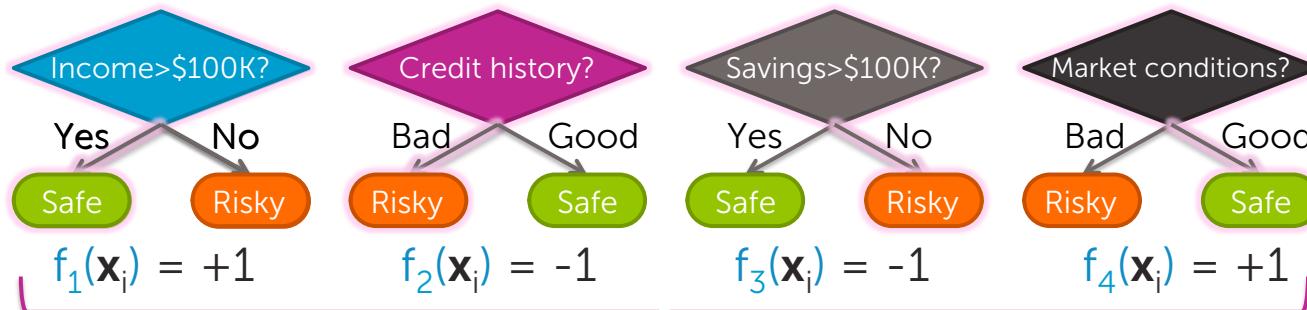


Boosting



**Amazing impact:** • simple approach • widely used in industry • wins most Kaggle competitions

# Module 7: Boosting using AdaBoost



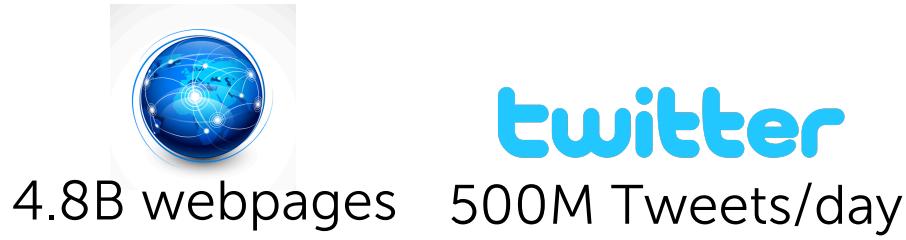
**Ensemble:** Combine votes from many simple classifiers to learn complex classifiers



# Module 8: Precision-recall

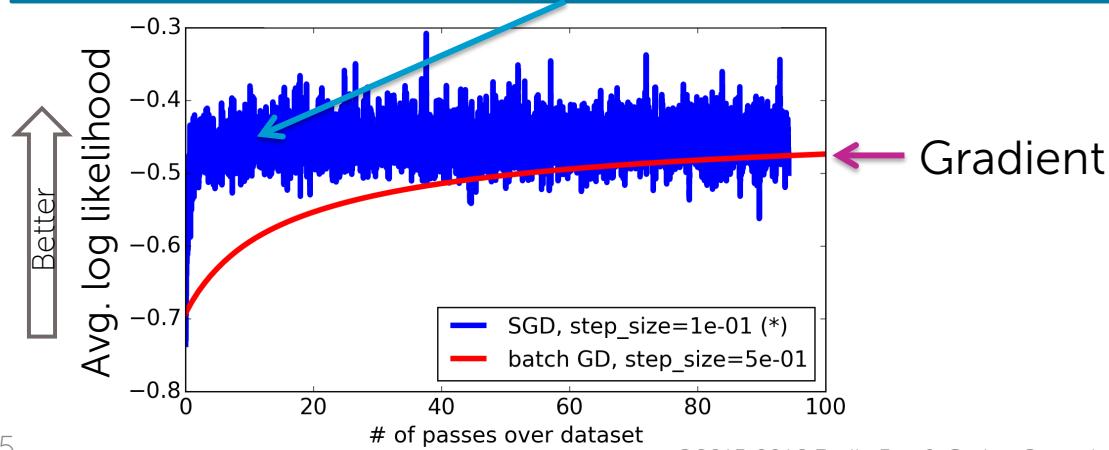


# Module 9: Scaling to huge datasets & online learning



You Tube  
5B views/day

**Stochastic gradient:** tiny modification to gradient,  
a lot faster, but annoying in practice



# Assumed background



# Courses 1 & 2 in this ML Specialization

- Course 1: Foundations
  - Overview of ML case studies
  - Black-box view of ML tasks
  - Programming & data manipulation skills
- Course 2: Regression
  - Data representation (input, output, features)
  - Linear regression model
  - Basic ML concepts:
    - ML algorithm
    - Gradient descent
    - Overfitting
    - Validation set and cross-validation
    - Bias-variance tradeoff
    - Regularization

# Math background

- Basic calculus
  - Concept of derivatives
- Basic vectors
- Basic functions
  - Exponentiation  $e^x$
  - Logarithm

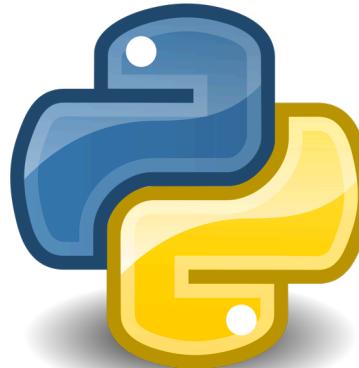


# Programming experience

- Basic Python used
  - Can pick up along the way if knowledge of other language



```
    """  
    Returns a QuerySet of connections for user.  
    """  
    set1 = self.filter(from_user=user).select_related(depth=1)  
    set2 = self.filter(to_user=user).select_related(depth=1)  
    return set1 | set2  
  
def are_connected(self, user1, user2):  
    """  
    If self.filter(from_user=user1, to_user=user2).count() > 0:  
        return True  
    if self.filter(from_user=user2, to_user=user1).count() > 0:  
        return True  
    return False  
  
def remove(self, user1, user2):  
    """  
    Deletes proper object regardless of the order of users in arguments  
    connection = self.filter(from_user=user1, to_user=user2)  
    if not connection:  
        connection = self.filter(from_user=user2, to_user=user1)  
    connection.delete()  
--:---  
models.py   Top L1   (Python AC yes)---
```



# Reliance on GraphLab Create

- SFrames will be used, though not required
  - open source project of Dato  
(creators of GraphLab Create)
  - can use pandas and numpy instead
- Assignments will:
  1. Use GraphLab Create to explore high-level concepts
  2. Ask you to implement *all* algorithms without GraphLab Create
- Net result:
  - learn how to code methods in Python



# Computing needs

- Basic 64-bit desktop or laptop
- Access to internet
- Ability to:
  - Install and run Python (and GraphLab Create)
  - Store a few GB of data





# Let's get started!