

# Regression:

## A machine learning perspective

Emily Fox & Carlos Guestrin  
Machine Learning Specialization  
University of Washington

# Part of a specialization

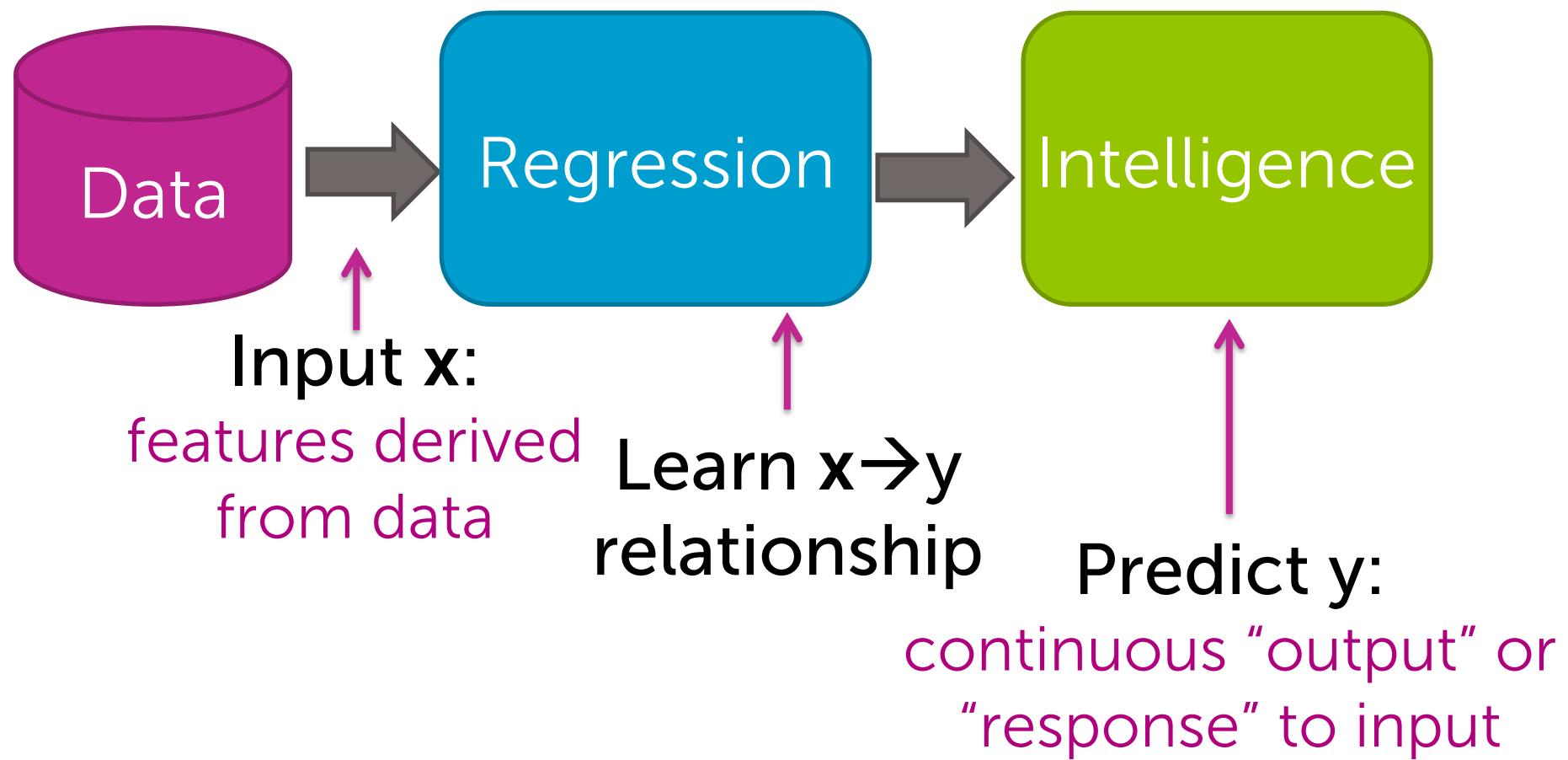
# This course is a part of the Machine Learning Specialization



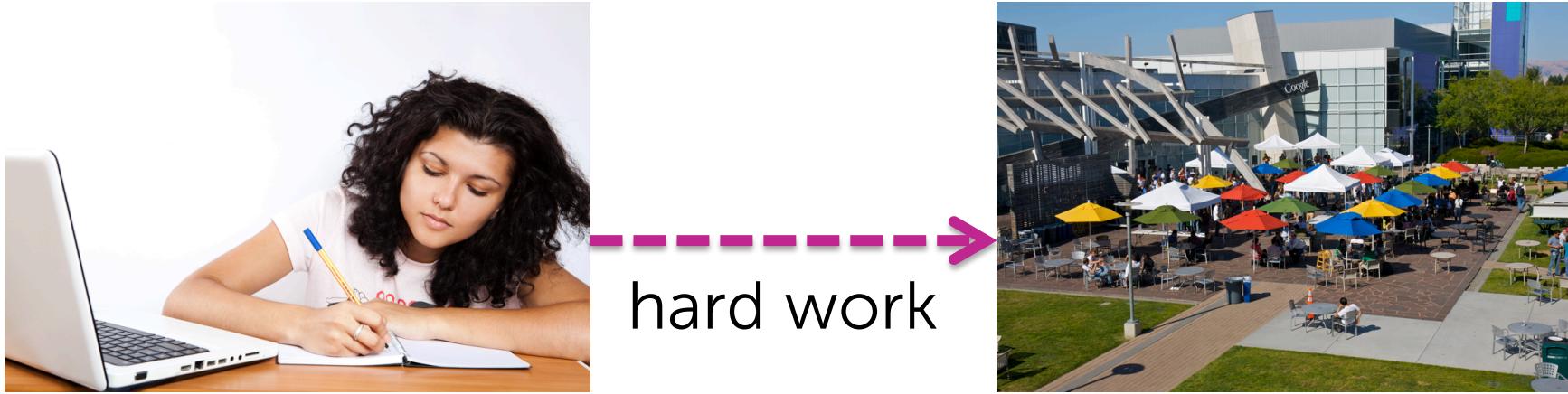
# What is the course about?

# What is regression?

From features to predictions



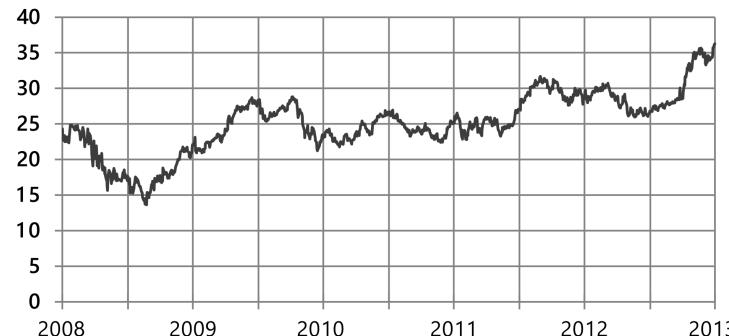
# Salary after ML specialization



- How much will your salary be? ( $y = \text{ $$}$ )
- Depends on  $x = \text{ performance in courses, quality of capstone project, \# of forum responses, ...}$

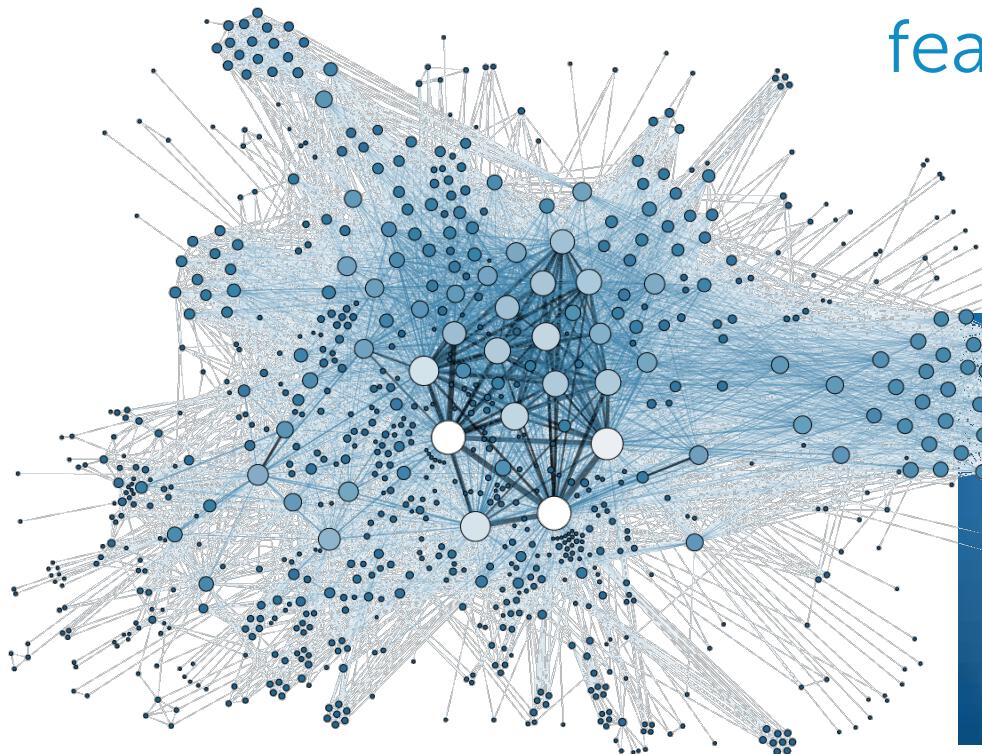
# Stock prediction

- Predict the price of a stock ( $y$ )
- Depends on  $x =$ 
  - Recent history of stock price
  - News events
  - Related commodities

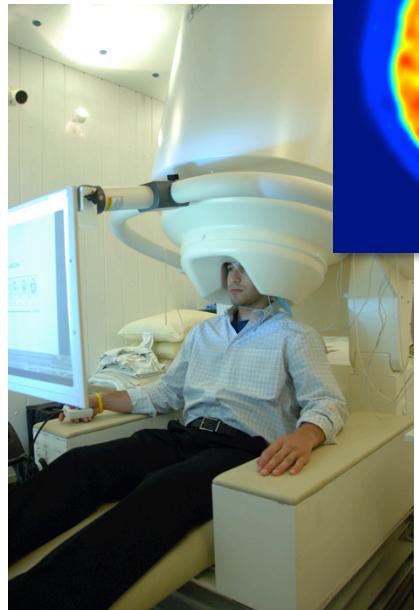
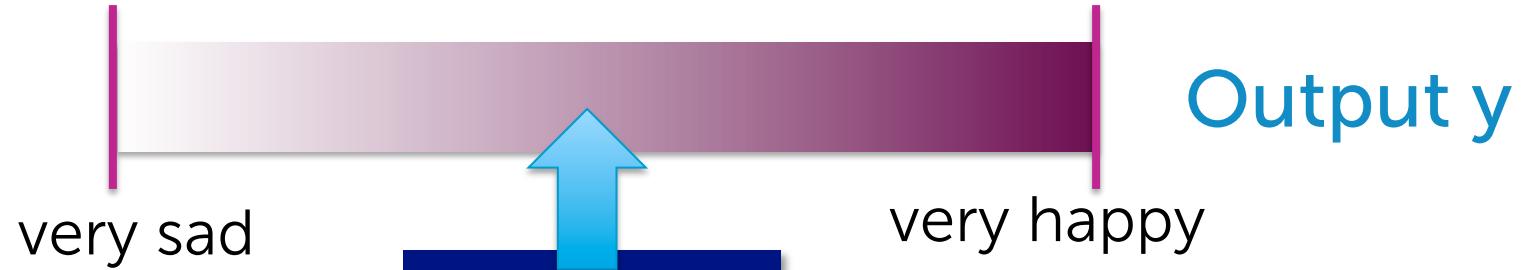


# Tweet popularity

- How many people will retweet your tweet? (y)
- Depends on  $x = \# \text{ followers}$ ,  
 $\# \text{ of followers of followers}$ ,  
 $\text{features of text tweeted}$ ,  
 $\text{popularity of hashtag}$ ,  
 $\# \text{ of past retweets}$ ,...

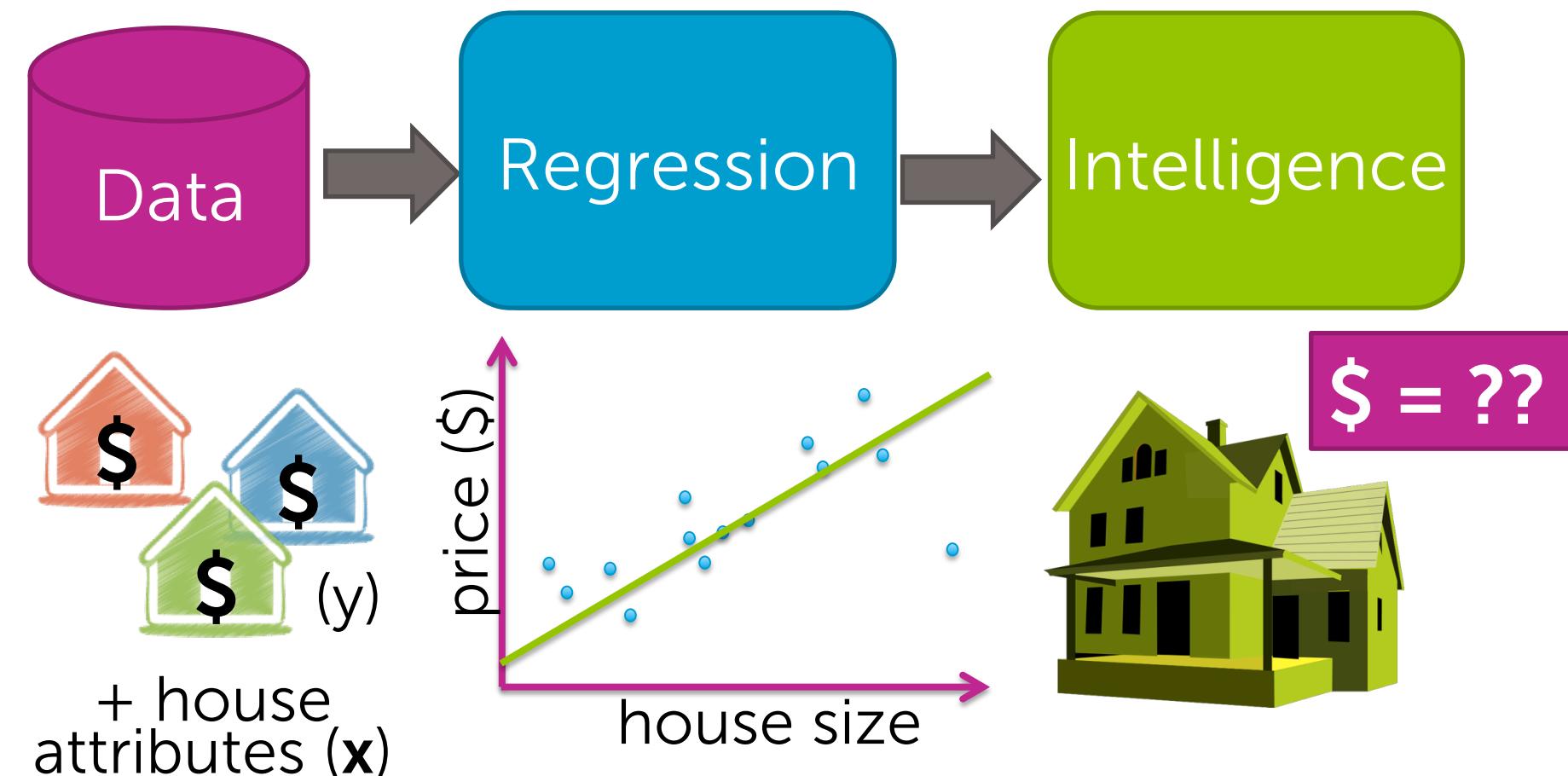


# Reading your mind



Inputs  $x$  are  
brain region  
intensities

# Case Study: Predicting house prices



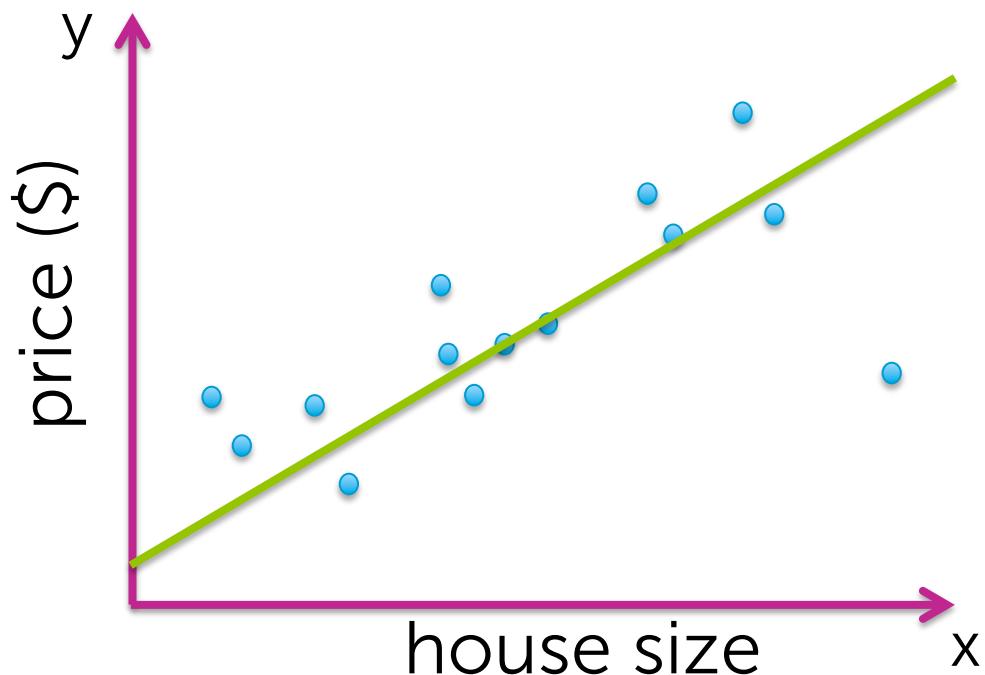
# Impact of regression

# Course outline

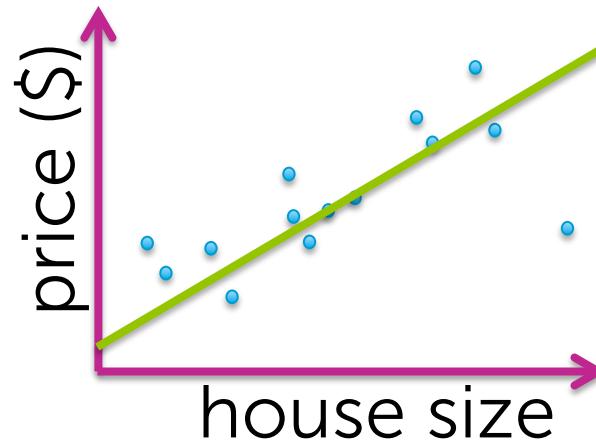
# Module 1: Simple Regression

What makes it simple?

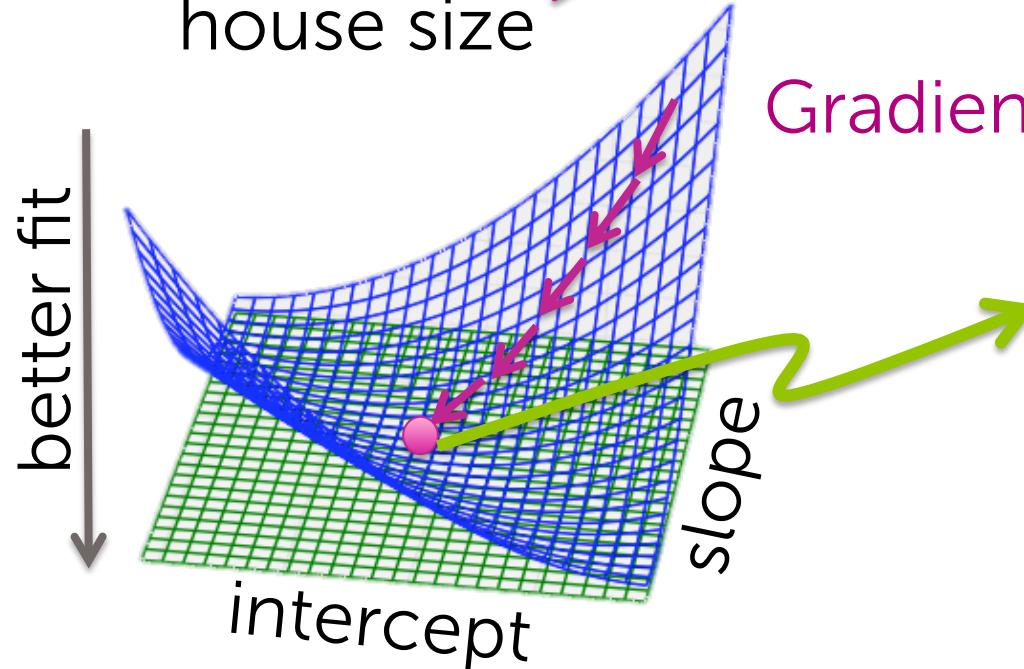
1 input and just fit a line to data



# Module 1: Simple Regression



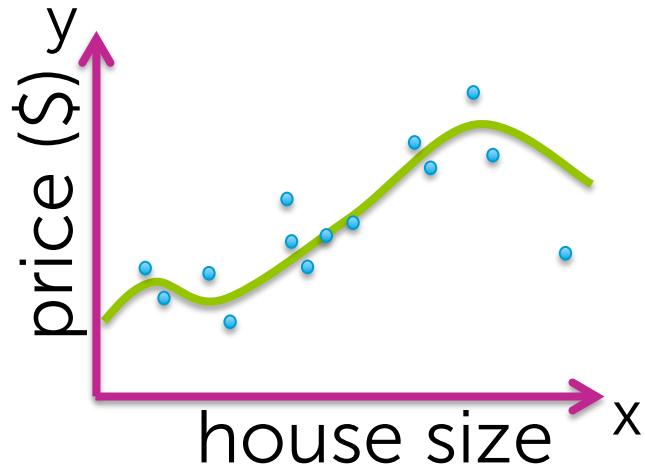
Define **goodness-of-fit** metric for each possible line



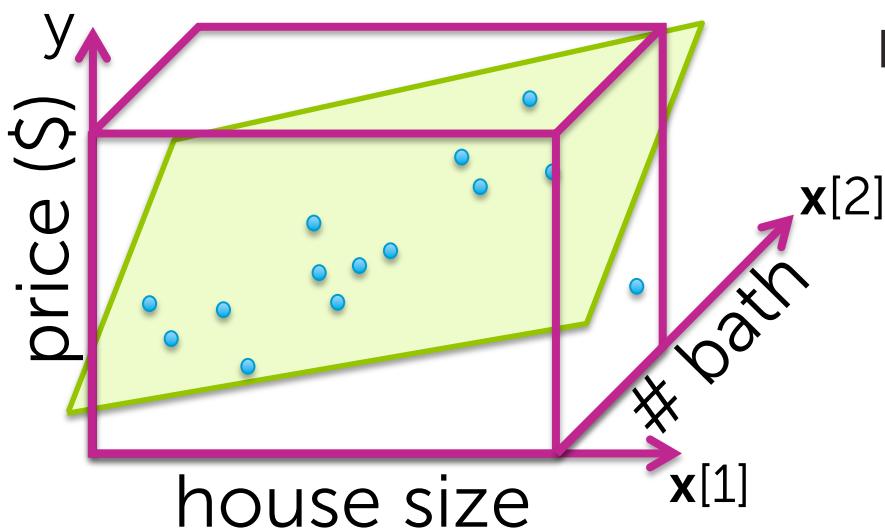
Gradient descent algorithm

Get estimated parameters  
– interpret  
– use to form predictions

# Module 2: Multiple Regression



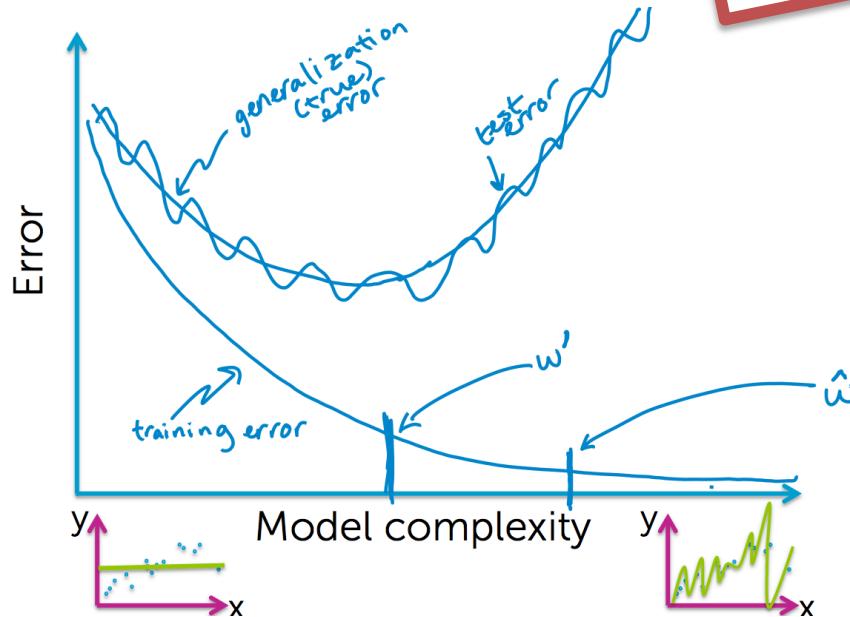
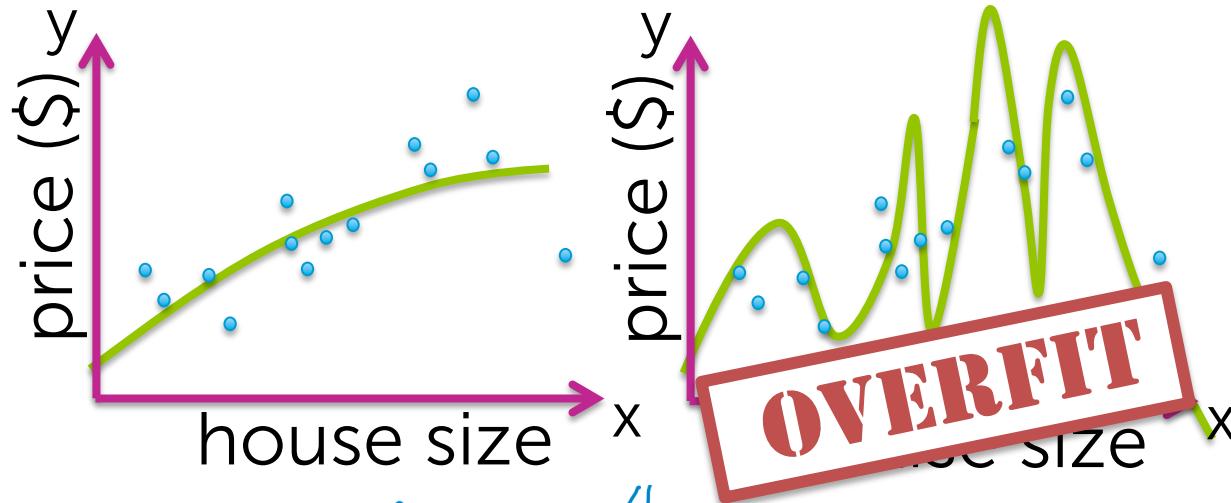
Fit **more complex relationships** than just a line



Incorporate more inputs

- Square feet
- # bathrooms
- # bedrooms
- Lot size
- Year built
- ...

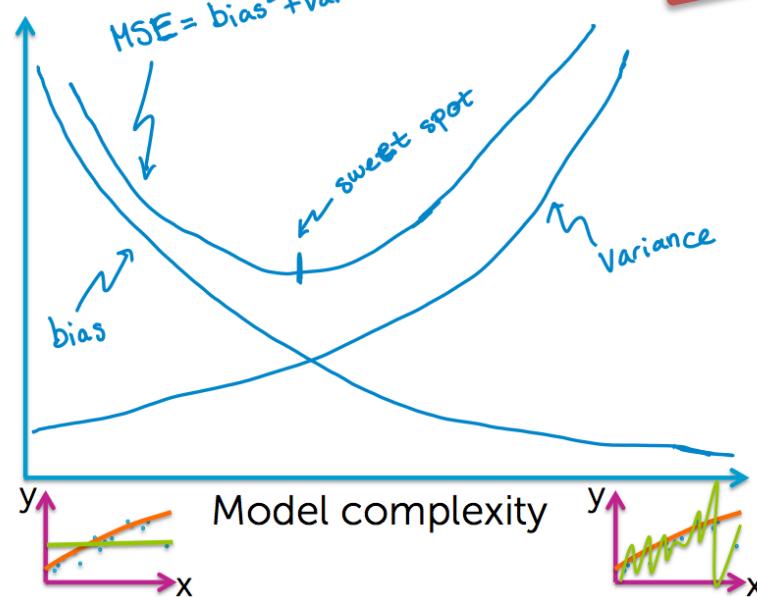
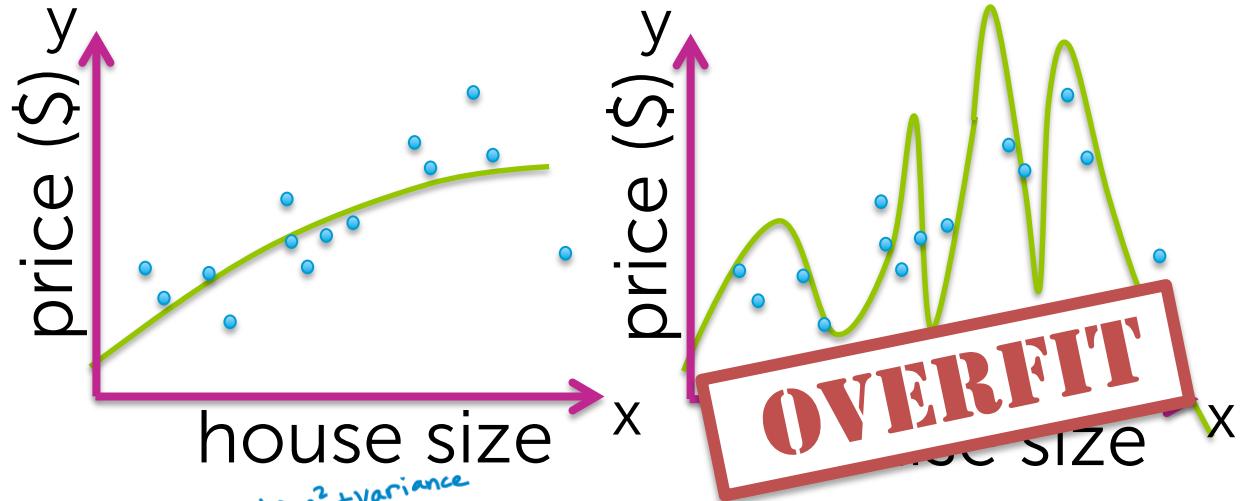
# Module 3: Assessing Performance



Measures of error:

- Training
- Test
- True (generalization)

# Module 3: Assessing Performance



Bias-variance  
tradeoff

# Module 4: Ridge Regression



Ridge total cost =  
measure of fit + measure of  
model complexity

bias-variance tradeoff

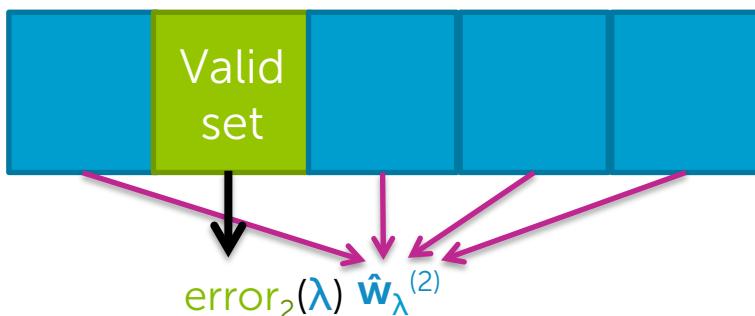
A diagram below the text shows a triangle with three arrows originating from its vertices. Two arrows point from the top-left vertex to the words 'measure of fit' and 'measure of model complexity' respectively. A third arrow points from the bottom vertex to the text 'bias-variance tradeoff'.

# Module 4: Ridge Regression

How to choose balance?  
(i.e., model complexity)

measure of fit + measure of  
model complexity

Cross validation



# Module 5: Feature Selection & Lasso Regression



Useful for **efficiency**  
of predictions and  
**interpretability**

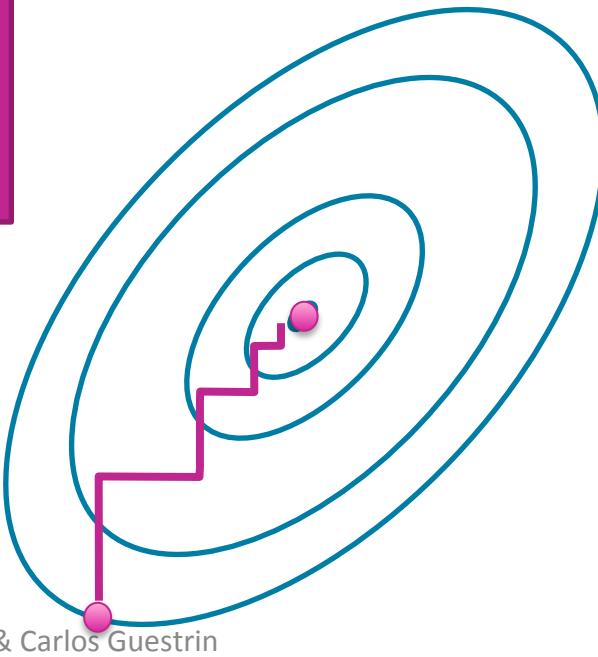
Lot size	Dishwasher
Single Family	Garbage disposal
Year built	Microwave
Last sold price	Range / Oven
Last sale price/sqft	Refrigerator
Finished sqft	Washer
Unfinished sqft	Dryer
Finished basement sqft	Laundry location
# floors	Heating type
Flooring types	Jetted Tub
Parking type	Deck
Parking amount	Fenced Yard
Cooling	Lawn
Heating	Garden
Exterior materials	Sprinkler System
Roof type	:
Structure style	:

# Module 5: Feature Selection & Lasso Regression

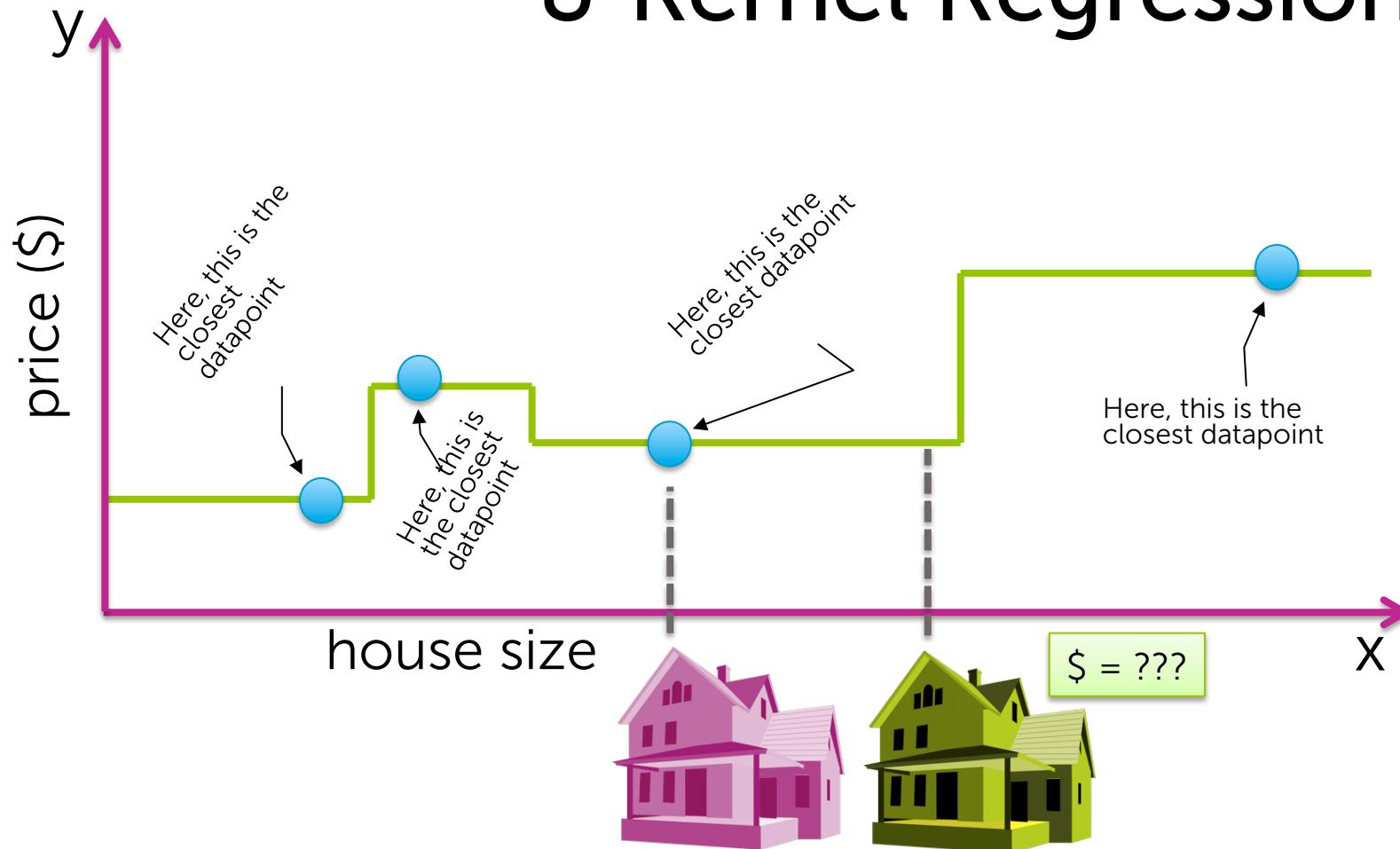
Lasso total cost =  
measure of fit + (different) measure of  
model complexity

knocks out certain features...  
“sparsity”

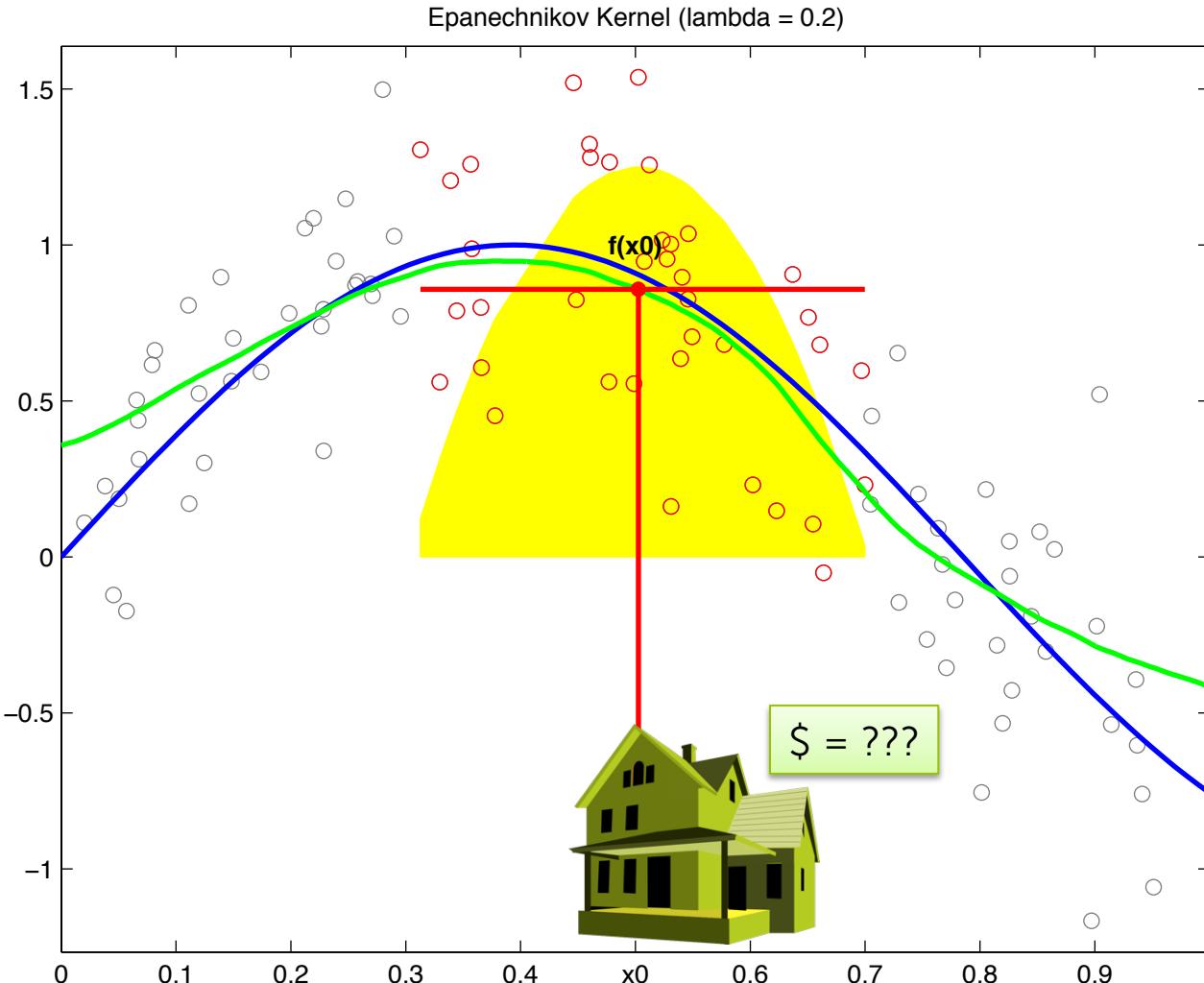
Coordinate descent algorithm



# Module 6: Nearest Neighbor & Kernel Regression



# Module 6: Nearest Neighbor & Kernel Regression



# Summary of what's covered

## Models

- Linear regression
- Regularization: Ridge (L2), Lasso (L1)
- Nearest neighbor and kernel regression

## Algorithms

- Gradient descent
- Coordinate descent

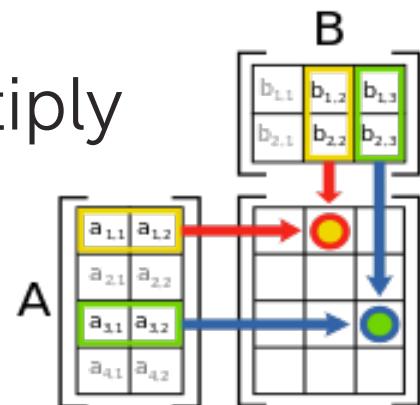
## Concepts

- Loss functions, bias-variance tradeoff, cross-validation, sparsity, overfitting, model selection, feature selection

# Assumed background

# Math background

- Basic calculus
  - Concept of derivatives
- Basic linear algebra
  - Vectors
  - Matrices
  - Matrix multiply



# Programming experience

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

```
get_user(self, user):
    """
    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

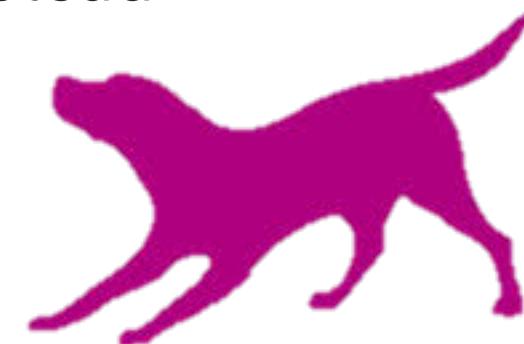
def are_connected(self, user1, user2):
    if self.filter(from_user=user1, to_user=user2).count() > 0:
        return True
    if self.filter(from_user=user2, to_user=user1).count() > 0:
        return True
    return False

def remove(self, user1, user2):
    """
    Deletes proper object regardless of the order of users in argument
    """
    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 yas)---
```



# Reliance on GraphLab Create

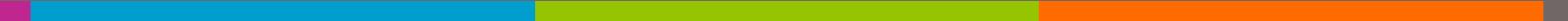
- SFrames will be used, though not required
  - open source project of Dato  
(creators of GraphLab Create)
  - can use pandas and numpy instead
- Assignments will:
  1. Use GraphLab Create to explore high-level concepts
  2. Ask you to implement *all* algorithms without GraphLab Create
- Net result: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