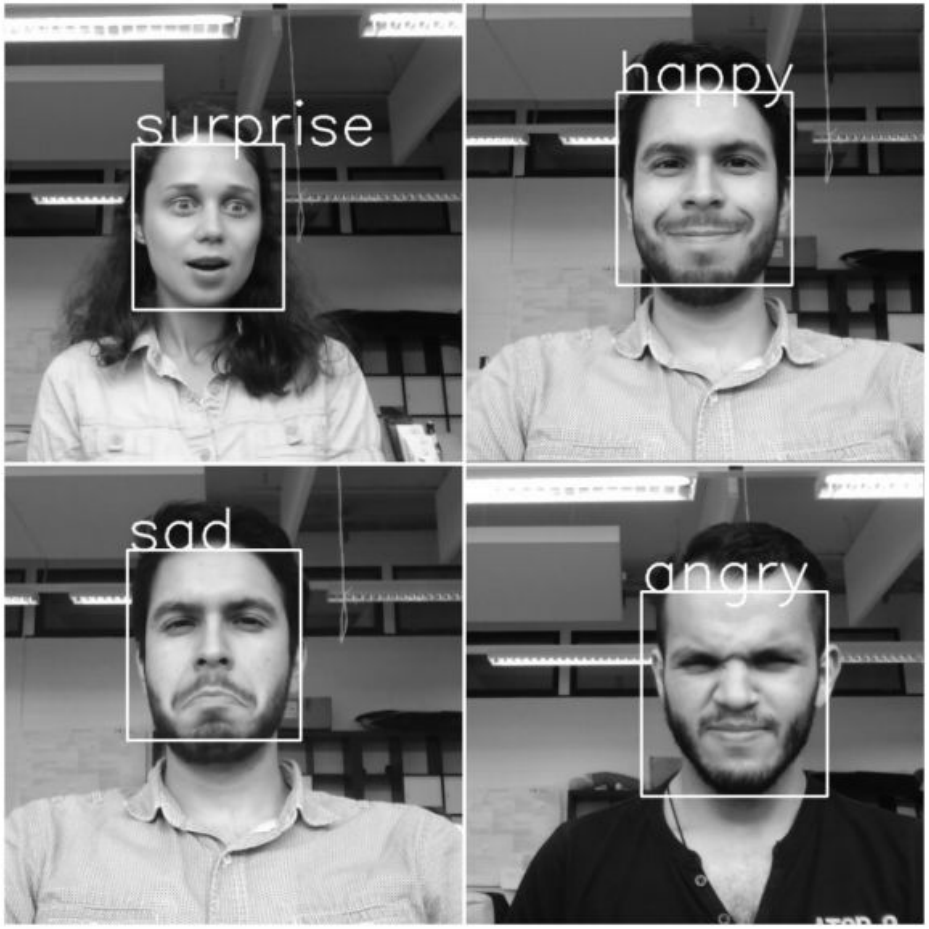
Course Reminders

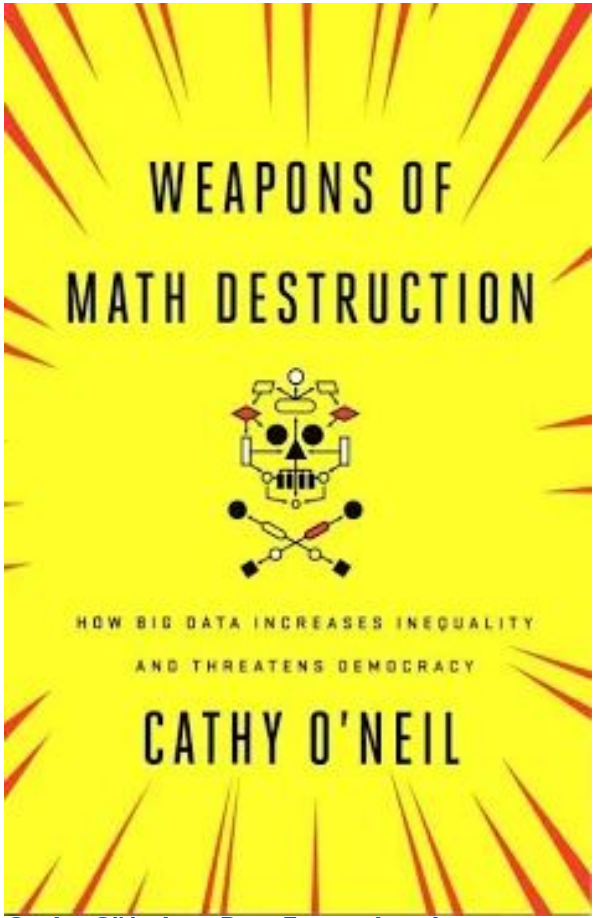
Due Dates:

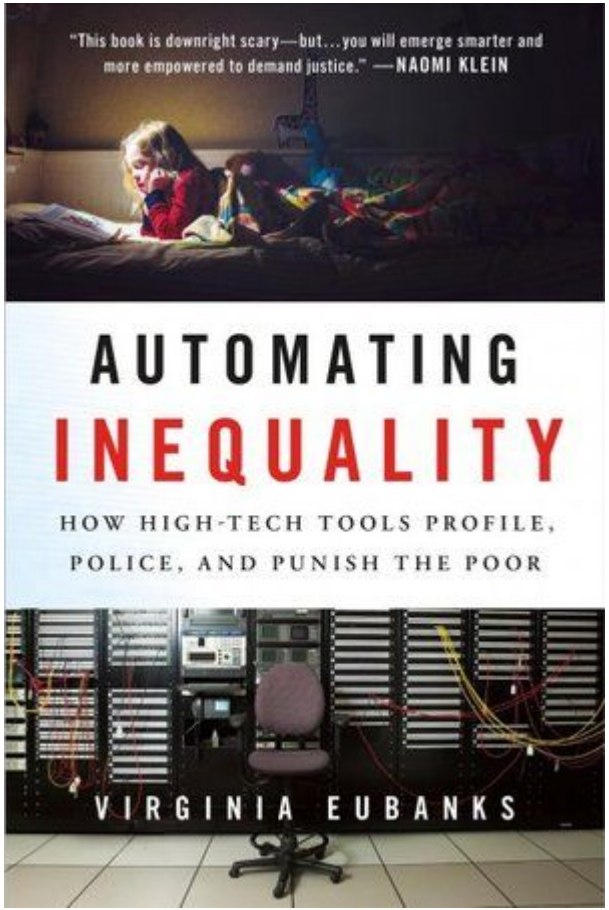
- A4 due next Sunday (11:59 PM)

https://twitter.com/nke_ise/status/897756900753891328



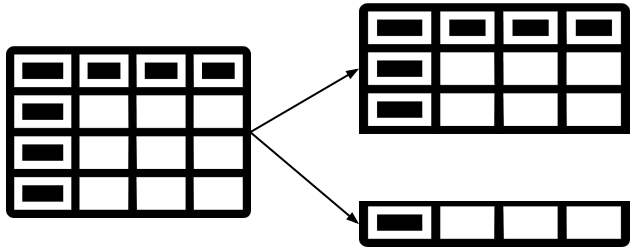
https://github.com/oarriaga/face_classification



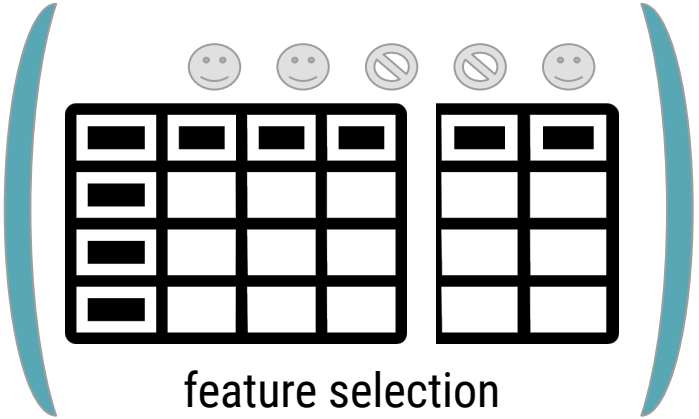


[Cathy O'Neil on DataFramed podcast]

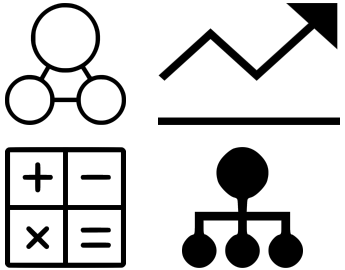
Basic Steps to Prediction



data
partitioning



feature selection



model selection

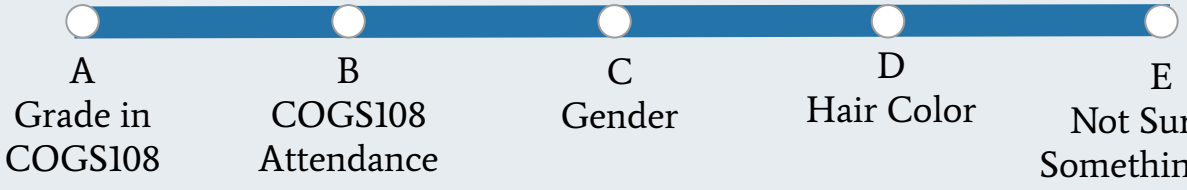
model assessment

Data Science Question

Based on data I have about you all, can I predict who in this course will be successful?

Prediction Hypothesis

Which would be the most predictive of your future success?



N = 254

train the model:
N = 178
(70% of the data)

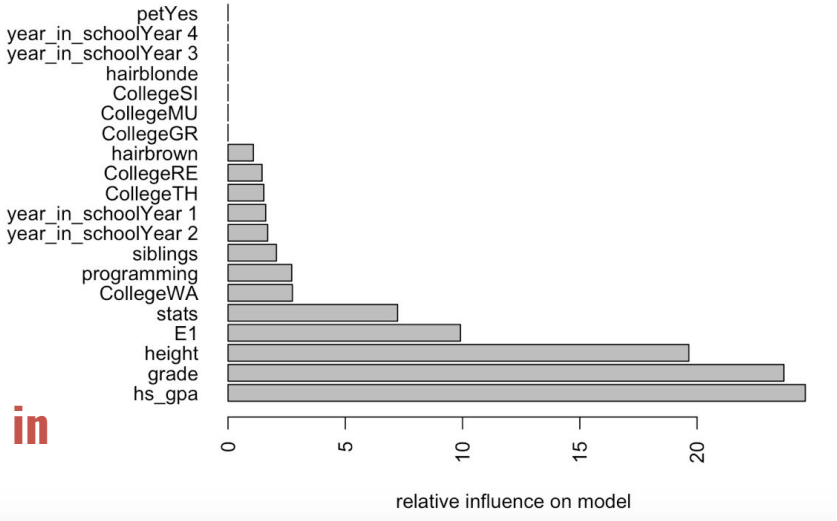
test the model:
N = 76
(30% of the data)

train the model

predicted success in test set

	Accuracy	Sensitivity	Specificity
training set	71.2%	76%	67%
test set	49.1%	40%	60%

Assess Prediction Model



N = 254

train the model:
N = 178
(70% of the data)

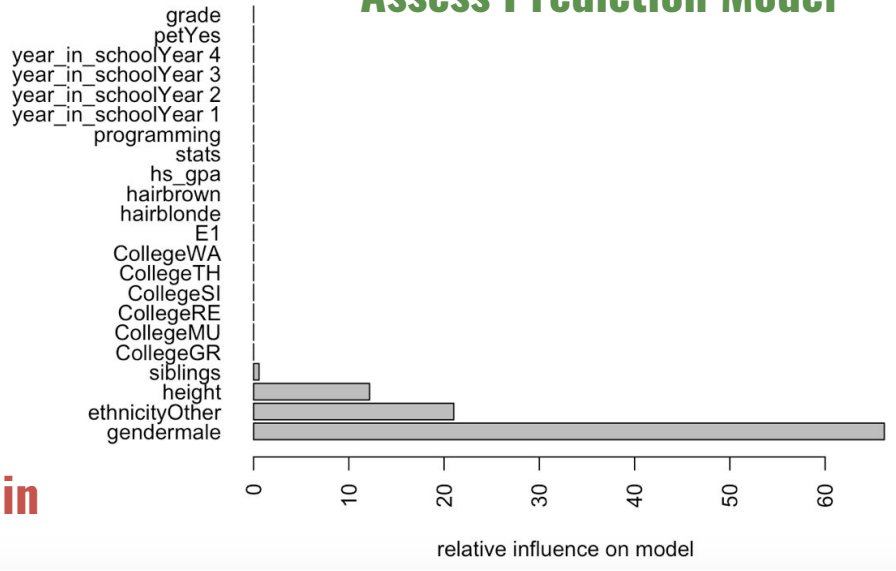
test the model:
N = 76
(30% of the data)

train the model

predicted success in test set

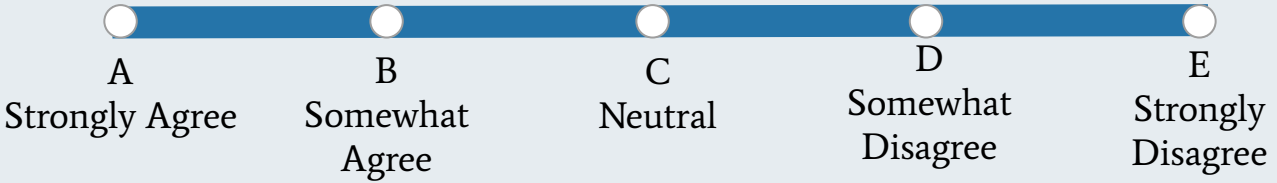
	Accuracy	Sensitivity	Specificity
training set	100%	100%	100%
test set	100%	100%	100%

Assess Prediction Model



Prediction Thoughts

Professor Ellis should use this model each quarter in her courses.



What if I were using these data to determine who I should write recommendation letters for?

Or to determine which students I focus my attention on?

Or whose projects I read?

Or who I allow to come to office hours?

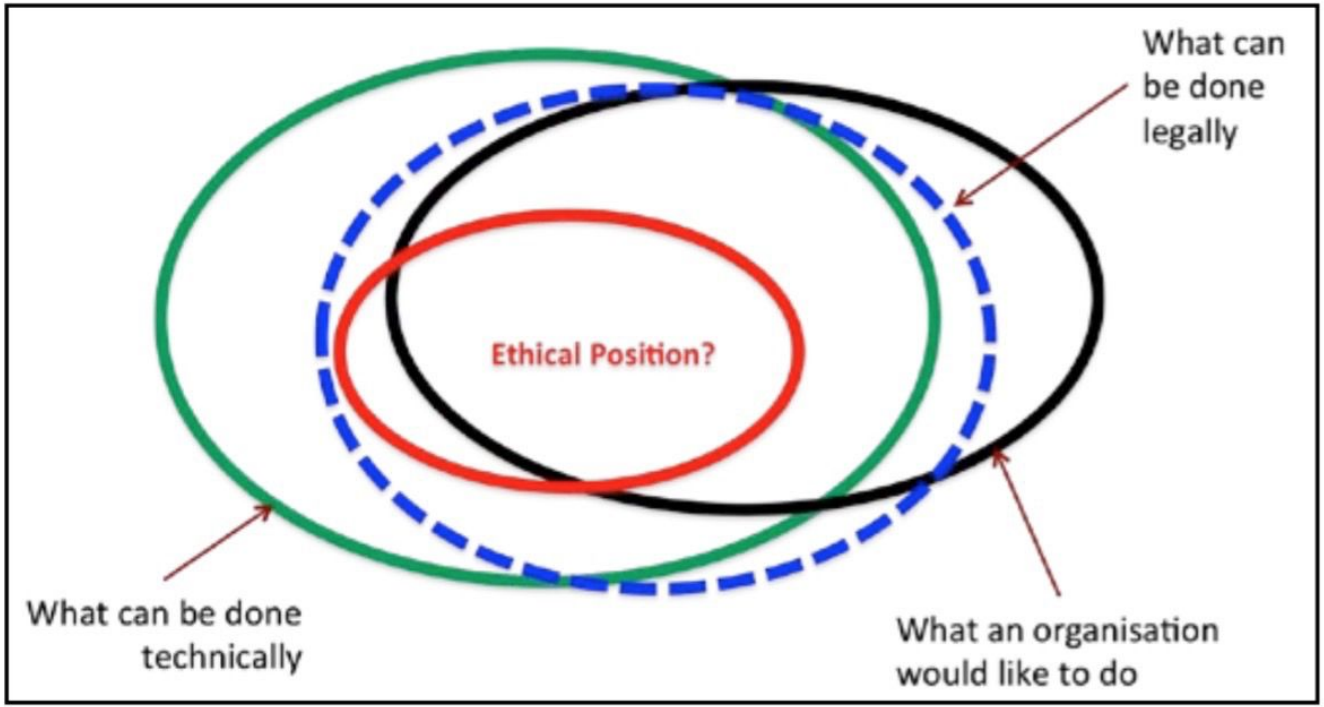
Or who UCSD allows to be certain majors?

What to do about bias...

1. Anticipate and plan for potential biases before model generation. Check for bias after.
2. Have diverse teams.
3. Use machine learning to improve lives rather than for punitive purposes.
4. Revisit your models. Update your algorithms.
5. You are responsible for the models you put out into the world, unintended consequences and all.

Think about whether the models you're building should even be built.

Big Data Ethics



Source: IBM

adapted from Brad Voytek

Predictive algorithms should (*at a minimum*) be FAT

Fair: lacking biases which create unfair and discriminatory outcomes

- For whom does this algorithm fail?
- Steps to take:
  1. Verify data about individual is correct
  2. Carry out “sensitivity test”

Accountable/Accurate: answerable to the people subject to them

- Correct data used? Is there a mechanism for appeal?

Transparent: open about how and why particular decisions were made

- Think *carefully* about what transparency is (Handing over source code likely isn’t the answer)

Sources: https://www.datacamp.com/community/podcast/weapons-math-destruction & https://ironholds.org/resources/papers/mulching.pdf

A Mulching Proposal

Analysing and Improving an Algorithmic System for Turning the Elderly into High-Nutrient Slurry

Os Keyes
Department of Human Centered Design & Engineering
University of Washington
Seattle, WA, USA
okeyes@uw.edu

Jevan Hutson
School of Law
University of Washington
Seattle, WA, USA
jevanh@uw.edu

Meredith Durbin
Department of Astronomy
University of Washington
Seattle, WA, USA
mdurbin@uw.edu

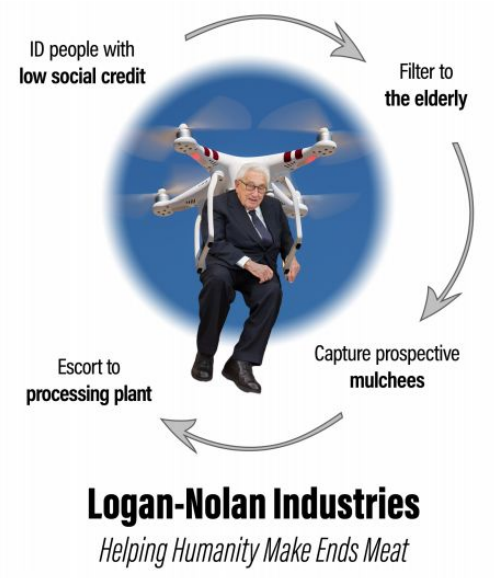
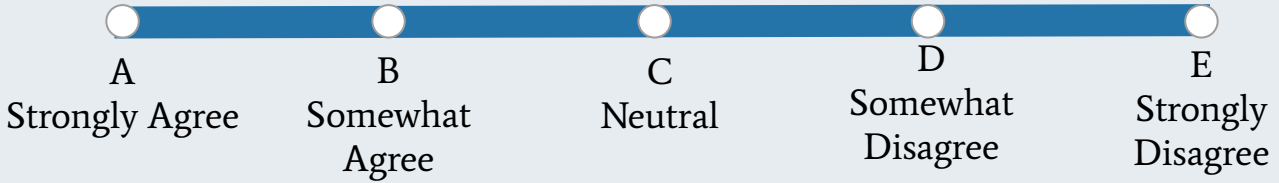


Figure 1: A publicity image for the project, produced by Logan-Nolan Industries

https://dl.acm.org/citation.cfm?id=3310433 (*CHI* 2019)

Prediction Thoughts

We should start using this algorithm to mulch up the elderly



A Mulching Proposal

FAIR - equally considers all elderly individuals

ACCURATE - pre- has mechanism for appeal; post - compensation

TRANSPARENT - website with all features ; testable

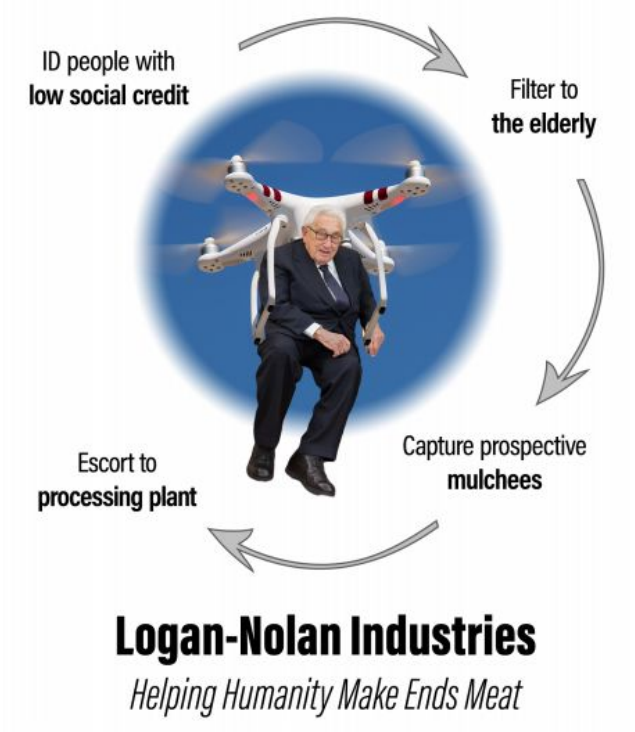


Figure 1: A publicity image for the project, produced by Logan-Nolan Industries

Checklists are helpful, but they’re not and excuse for thoughtlessness.

Machine Learning: Examples

Shannon E. Ellis, Ph.D
UC San Diego

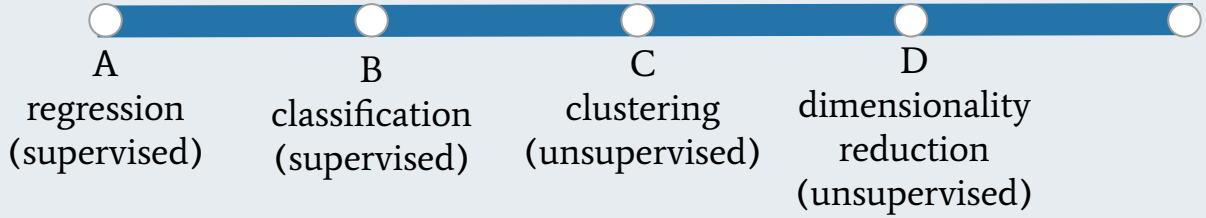
Department of Cognitive Science
sellis@ucsd.edu

Worked Example I:

Predicting whether a home is in San Francisco, CA or New York, NY

Prediction Approach

What type of machine learning task is “Predicting whether a home is in San Francisco, CA or New York, NY?”



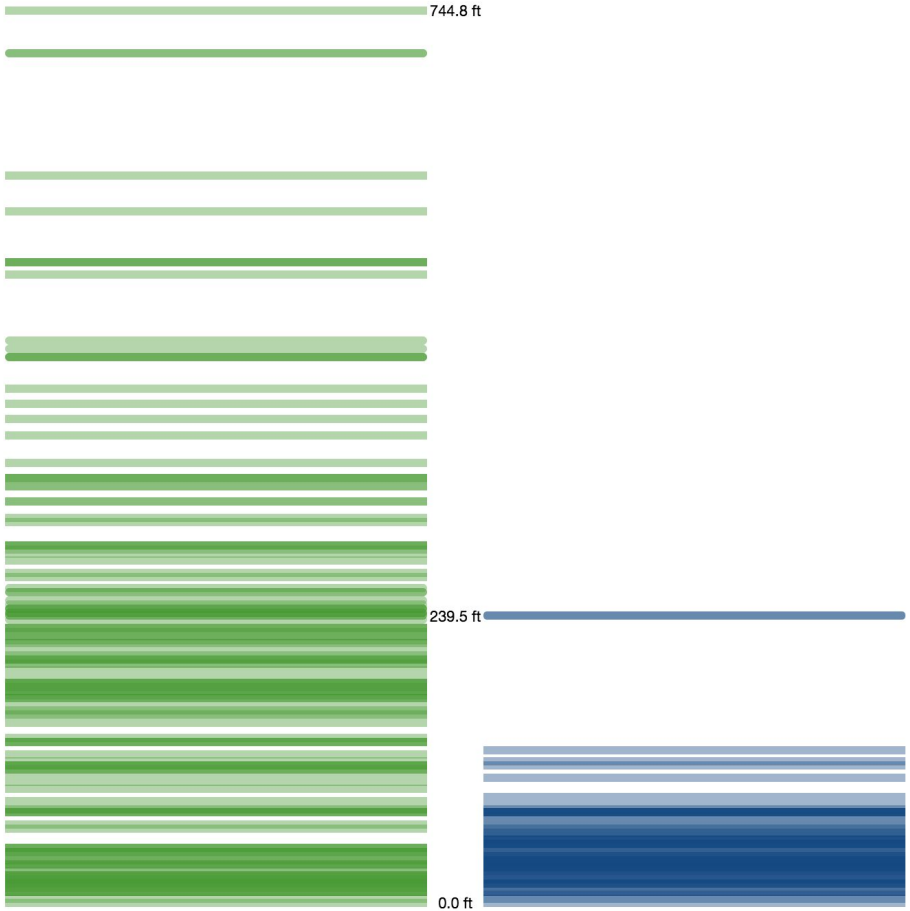
What features distinguish a house in New York from a house in San Francisco?

First, some intuition

Let's say you had to determine whether a home is in **San Francisco** or in **New York**. In machine learning terms, categorizing data points is a **classification** task.

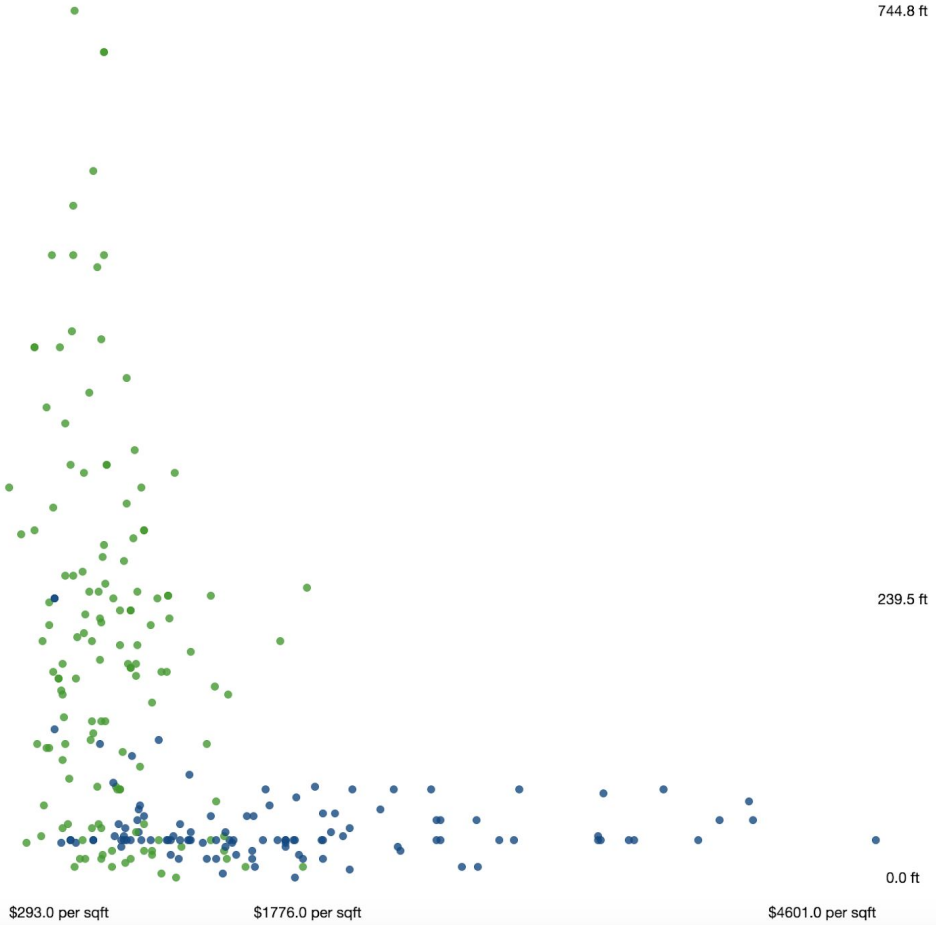
San Fran is hilly ...so elevation may be a helpful feature.

With the data here, homes > ~73m should be classified as San Fran homes



Adding nuance

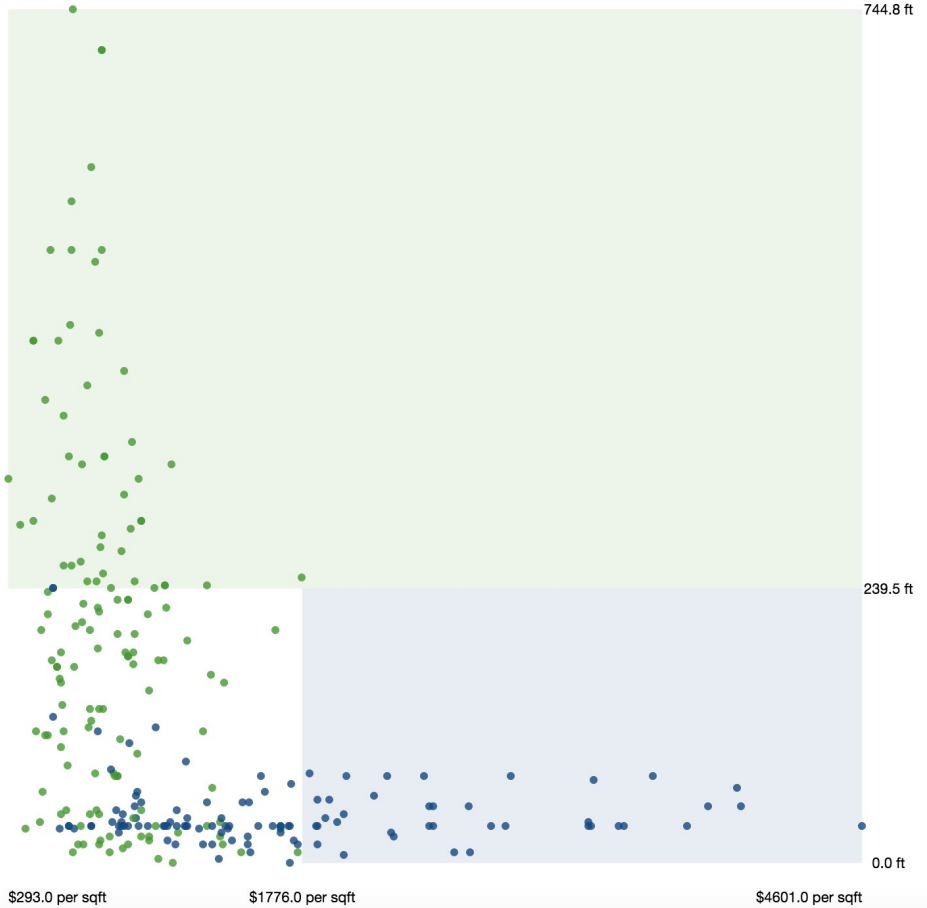
Elevation isn't a perfect feature for classification, so we can look at its relationship to other features, like *price per square foot*



http://www.r2d3.us/visual-intro-to-machine-learning-part-1/?from=@

Drawing boundaries

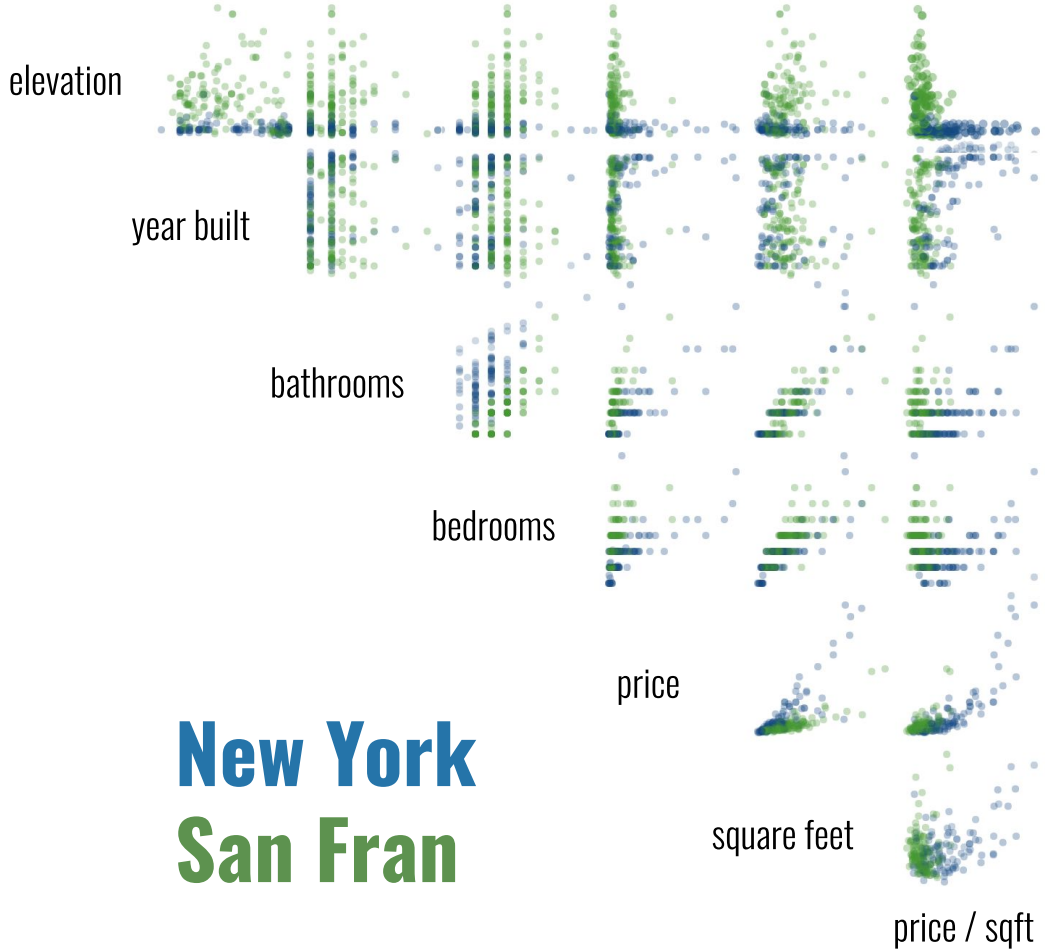
Boundaries can be drawn so that if a house falls in the green box, it's classified as a San Fran home. Blue box, New York. Statistical learning figures out how to best draw these boxes.



http://www.r2d3.us/visual-intro-to-machine-learning-part-1/?from=@

Our training set will use 7 different **features**. At the right we see the **scatterplot matrix** of the relationship between these features.

Patterns are clear, but boundaries for delineation are not obvious.



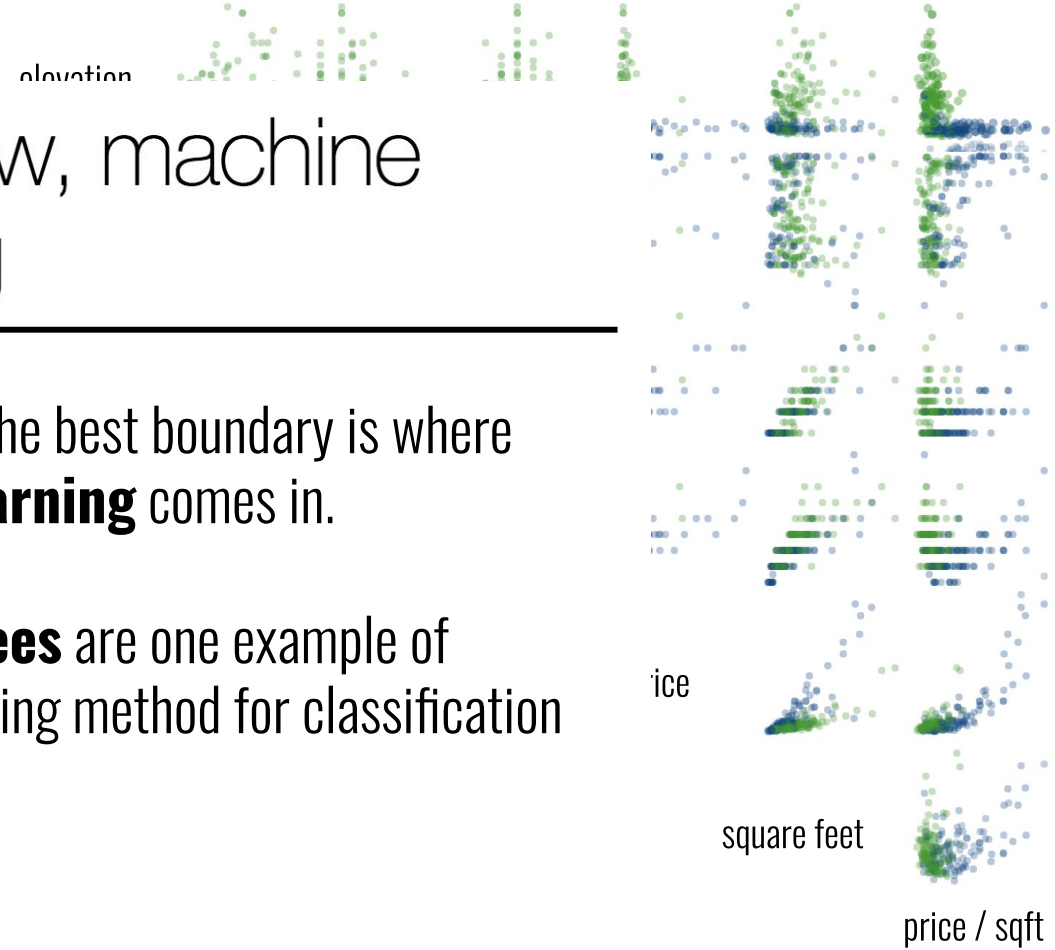
And now, machine learning

Determining the best boundary is where **machine learning** comes in.

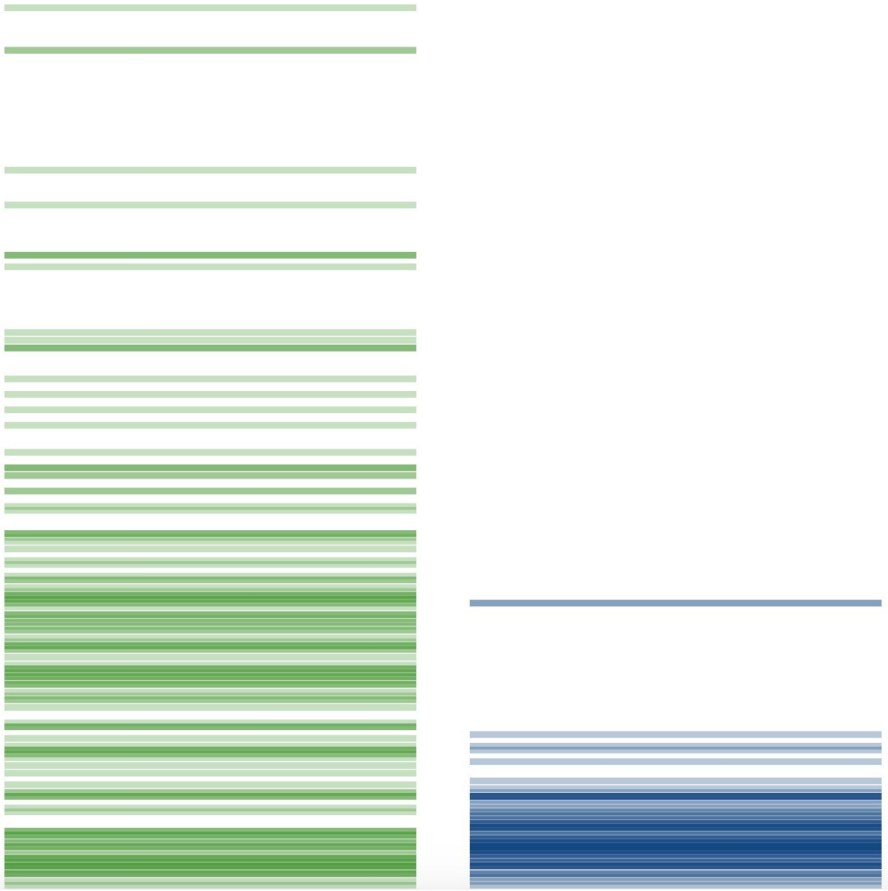
Decision trees are one example of machine learning method for classification tasks.

Our training set will u
features. At the righ
scatterplot matrix
relationship between

Patterns are clear, bu
delineation are not ob



http://www.r2d3.us/visual-intro-to-machine-learning-part-1/?from=@



Finding better boundaries

We guessed ~73m before. Let's improve on that guess...

A **histogram** helps display frequency of homes by elevation more easily.

73m is the highest home in New York, but most of them have lower elevations

http://www.r2d3.us/visual-intro-to-machine-learning-part-1/?from=@

Your first fork

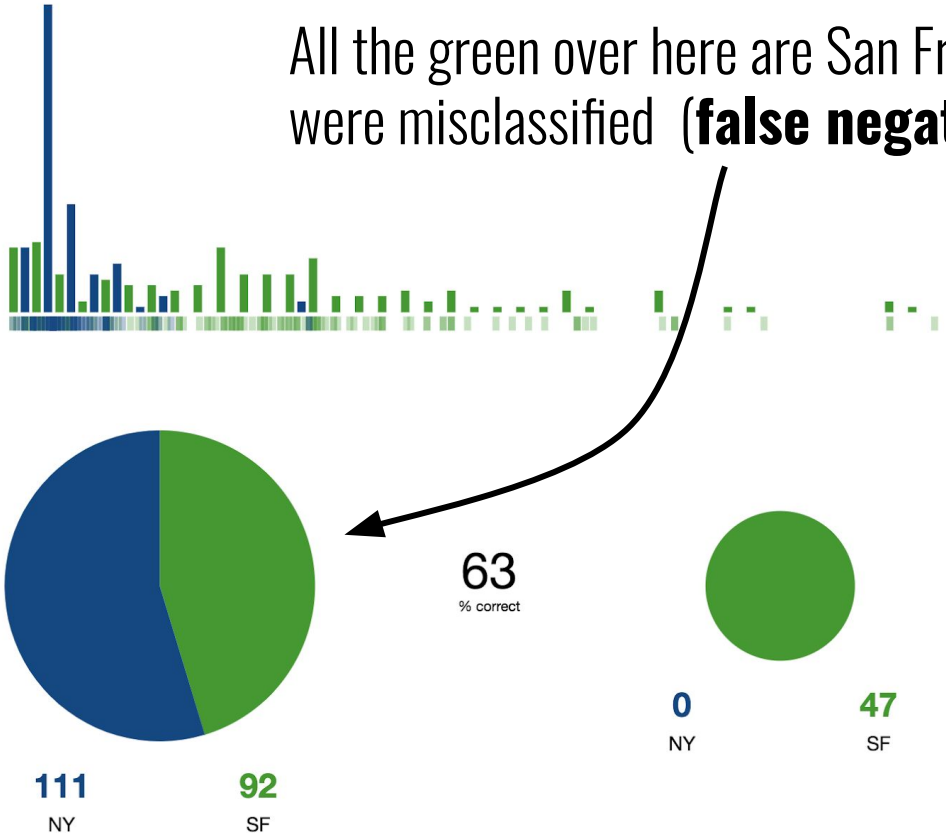
A decision tree uses if-then statements to define patterns in the data.

In machine learning, the splits are called **forks** and they split the data into **branches** based on some value.

The value that splits the branches is the **split point.** Homes to the left get categorized differently than those on the right.

http://www.r2d3.us/visual-intro-to-machine-learning-part-1/?from=@

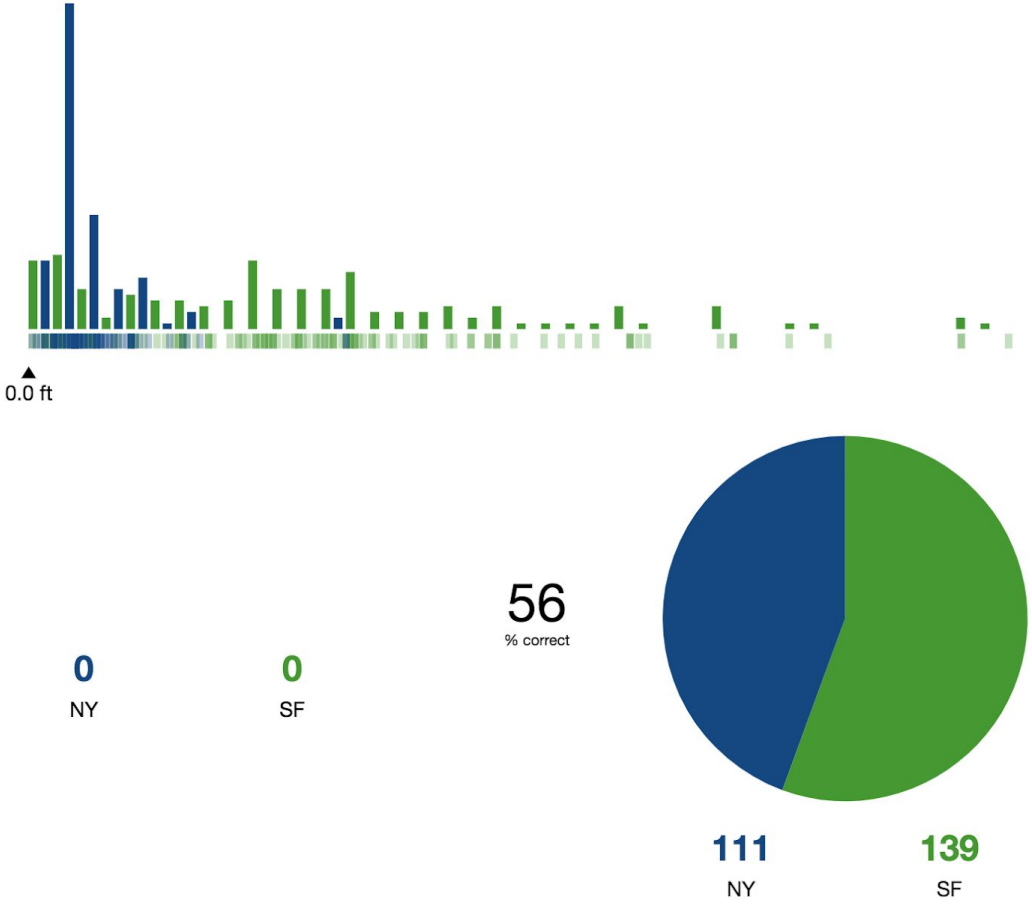
All the green over here are San Fran homes that were misclassified (**false negatives**)



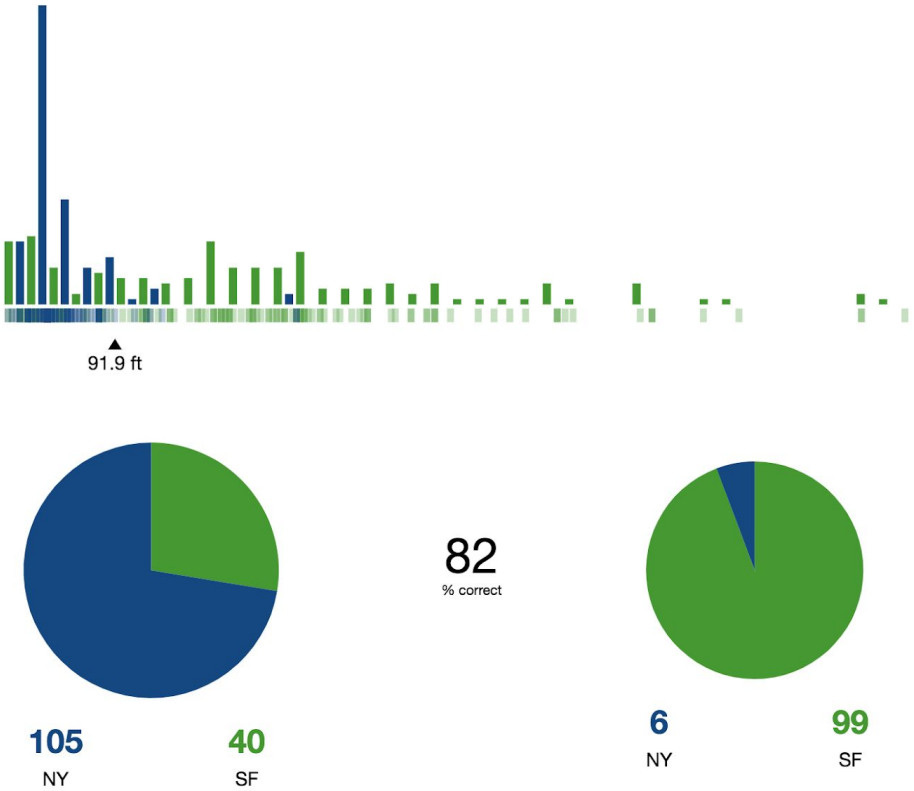
Tradeoffs

Splitting at ~73m incorrectly classifies some San Francisco homes as New York homes.

http://www.r2d3.us/visual-intro-to-machine-learning-part-1/?from=@

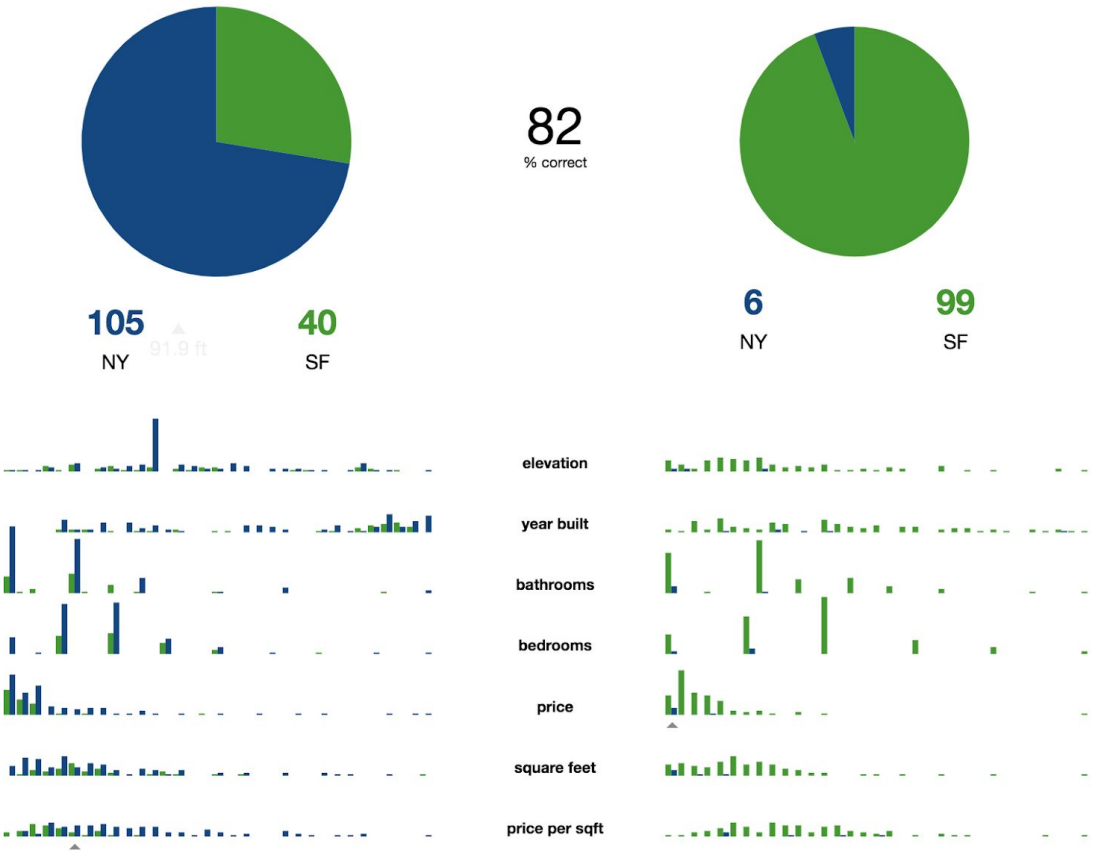


If you split to capture *every* home in San Fran, you'll also get a bunch of New York homes (**false positives**)



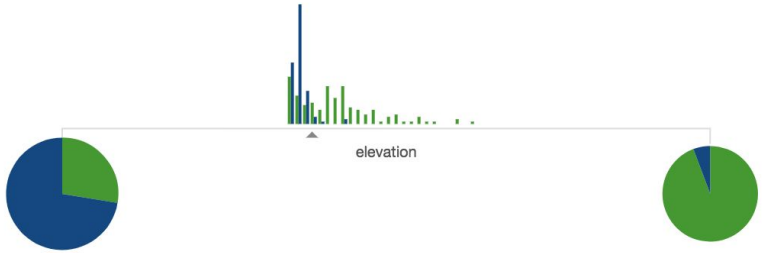
The best split

The best split point aims for branches that are as homogenous (pure) as possible



Recursion

Additional split points are determined through repetition (**recursion**)



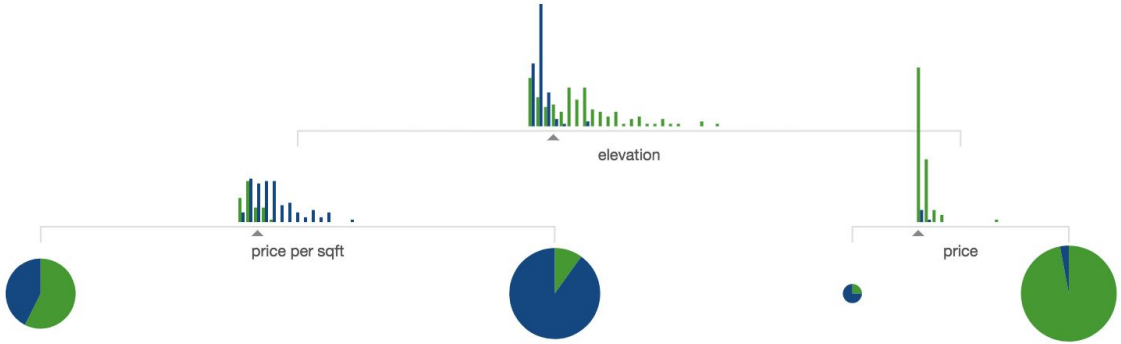
Growing a tree

Additional forks add new information to improve **prediction accuracy**.

Accuracy: 82%

Growing a tree

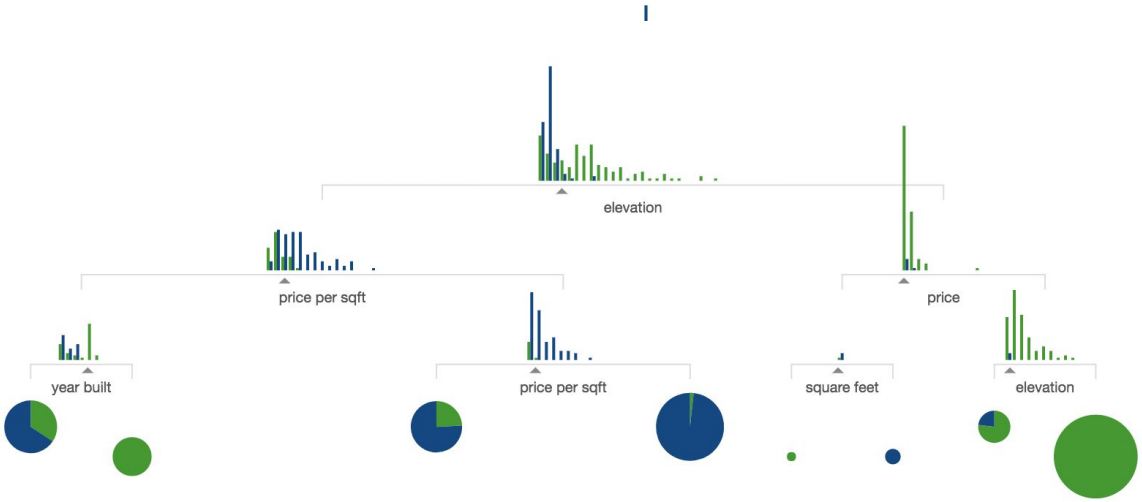
Additional forks add new information to improve **prediction accuracy**.



Accuracy: 86%

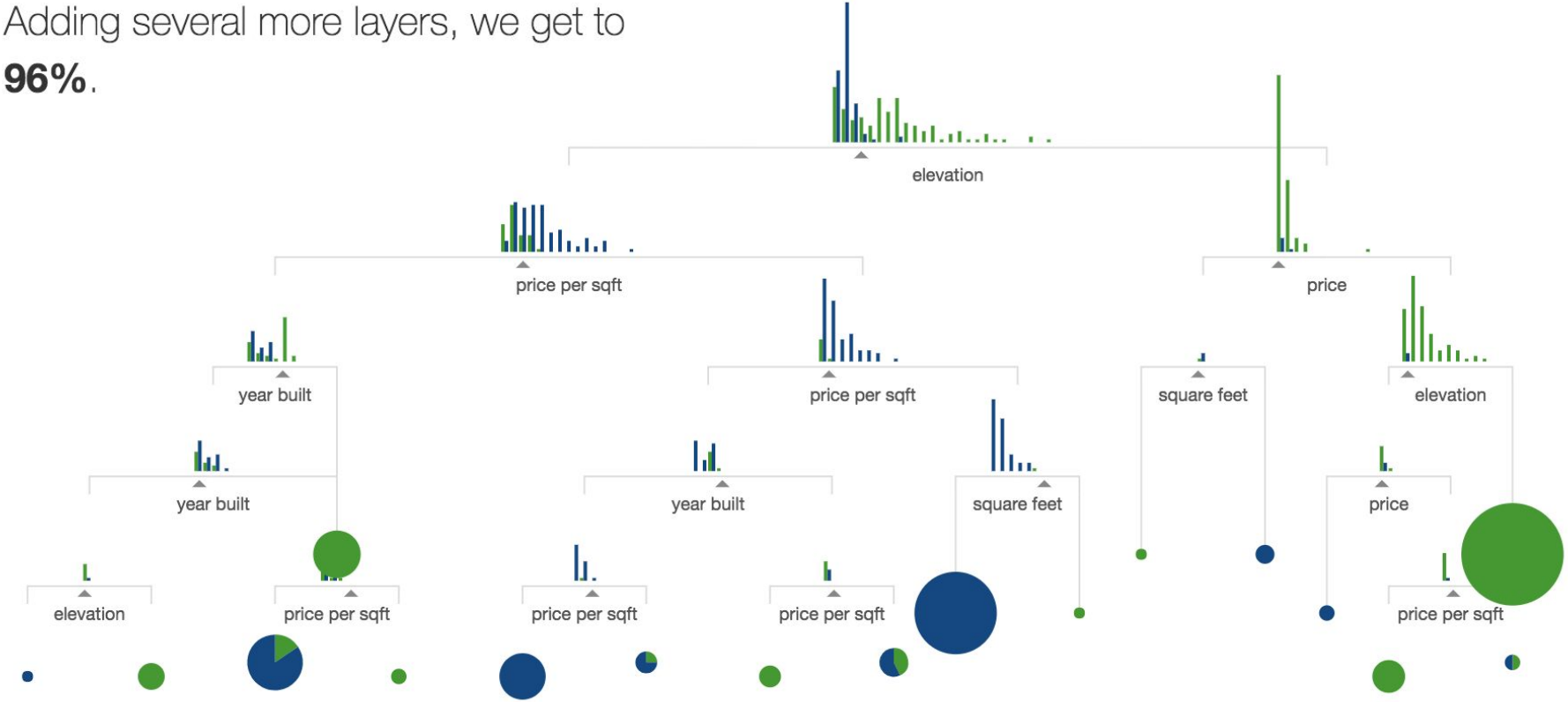
Growing a tree

Additional forks add new information to improve **prediction accuracy**.



http://www.r2d3.us/visual-intro-to-machine-learning-part-1/?from=@

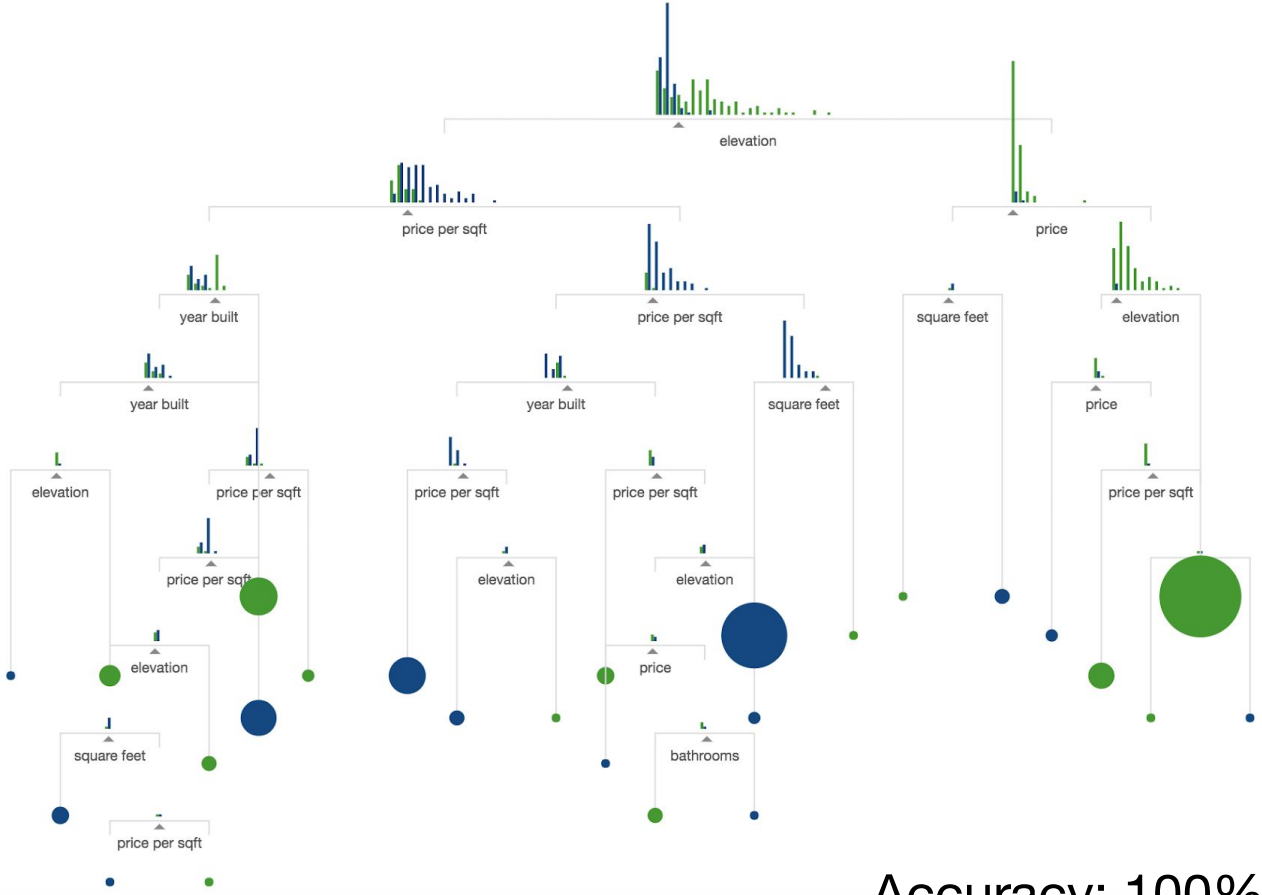
Adding several more layers, we get to **96%**.



Accuracy: 96%

http://www.r2d3.us/visual-intro-to-machine-learning-part-1/?from=@

It’s possible to add branches until your model is **100% accurate**.



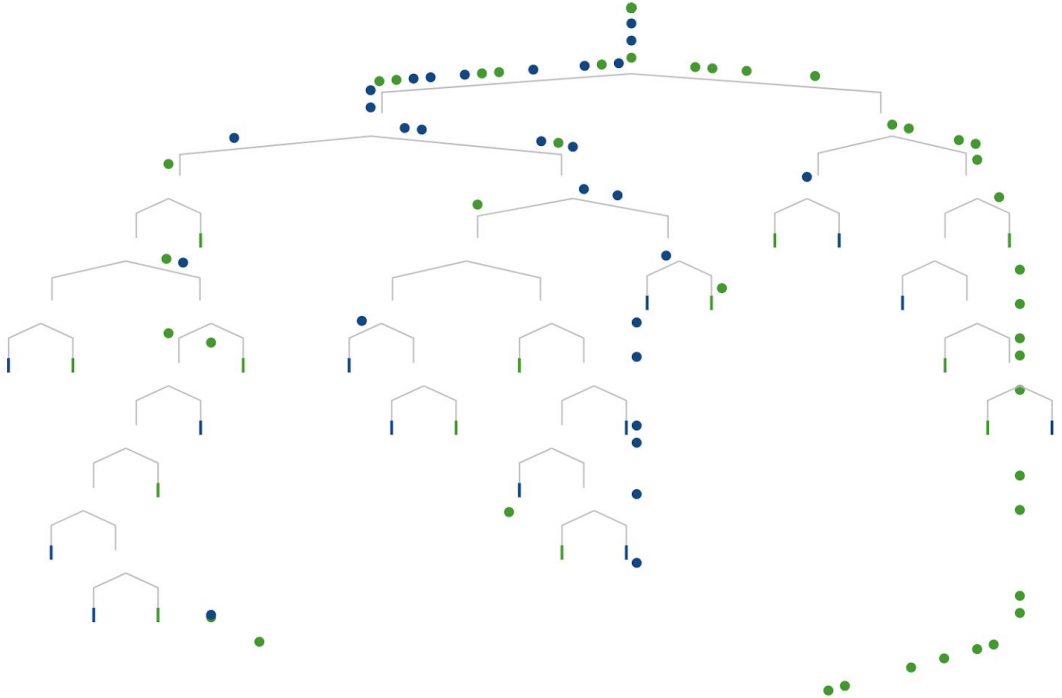
Accuracy: 100%

http://www.r2d3.us/visual-intro-to-machine-learning-part-1/?from=@

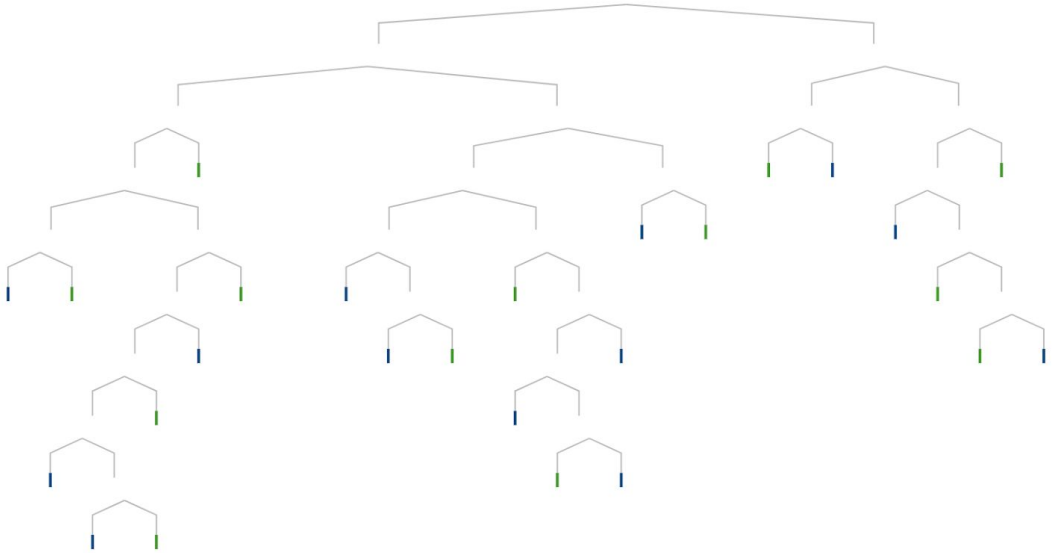
Making predictions

The decision tree **model** can then predict which homes are in which city.

Here, we're using the **training data**.



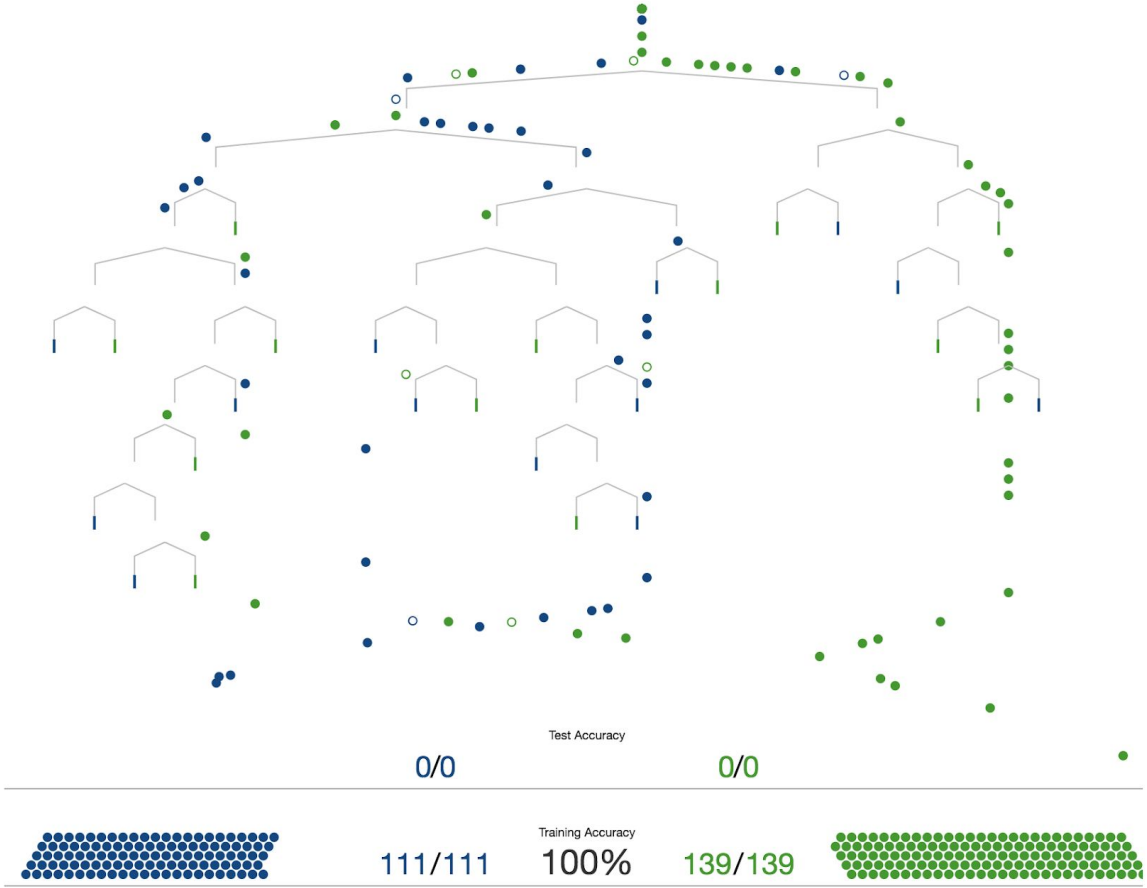
Because our tree was trained on this data and we grew the tree to 100% accuracy, each house is perfectly sorted



Reality check

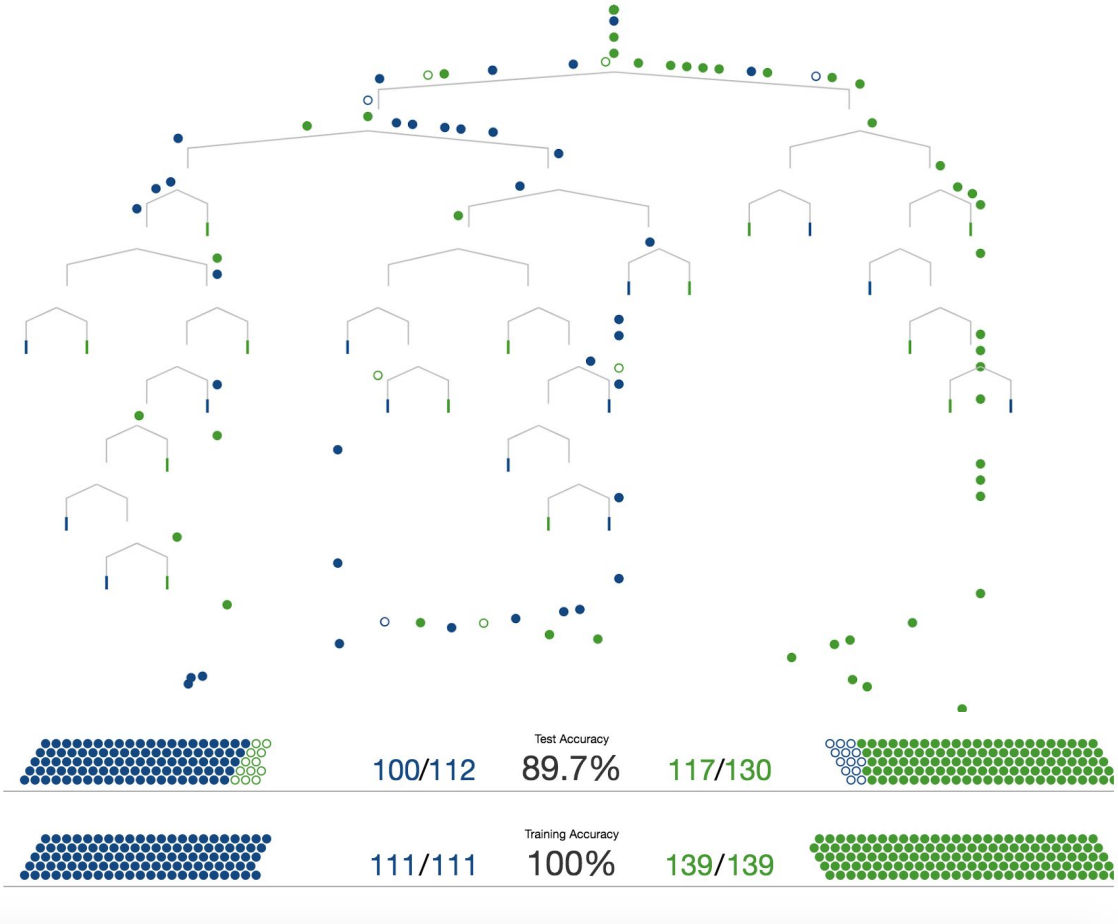
But…how does this tree on data that the model hasn't seen before?

The **test set** then makes it way through the decision tree.



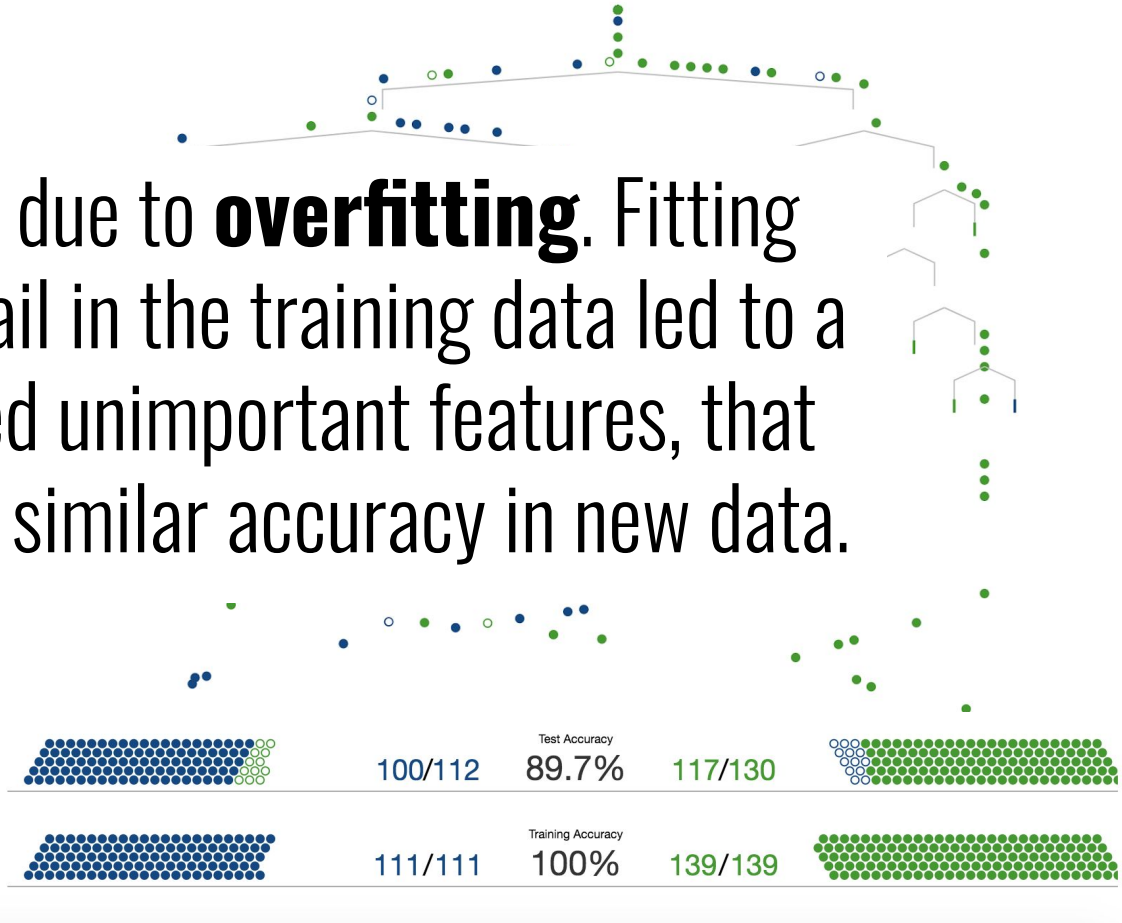
http://www.r2d3.us/visual-intro-to-machine-learning-part-1/?from=@

Ideally the tree should perform similarly on both known and unknown data



http://www.r2d3.us/visual-intro-to-machine-learning-part-1/?from=@

These errors are due to **overfitting**. Fitting every single detail in the training data led to a tree that modeled unimportant features, that did not allow for similar accuracy in new data.



http://www.r2d3.us/visual-intro-to-machine-learning-part-1/?from=@

Recap

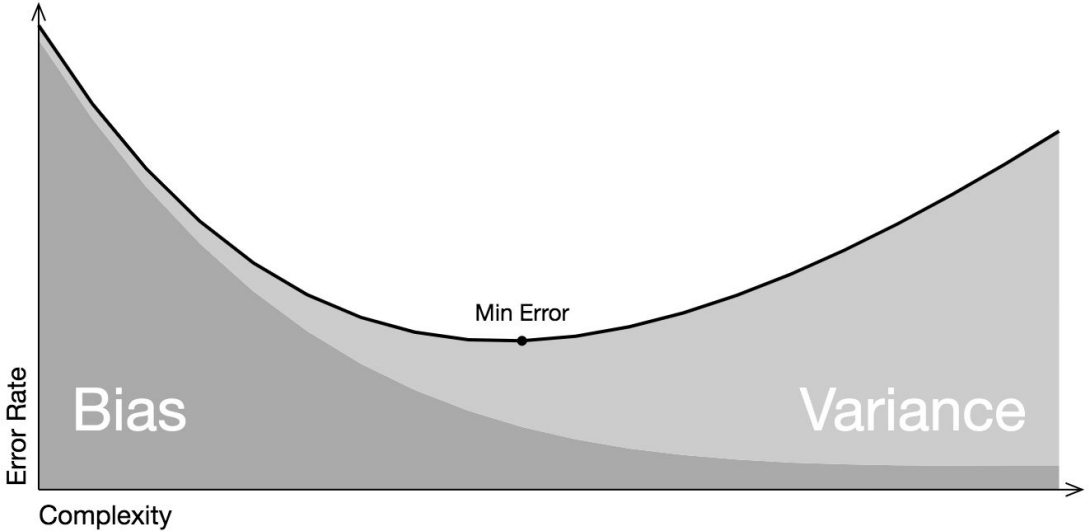
1. Machine learning identifies patterns using **statistical learning** and computers by unearthing **boundaries** in data sets. You can use it to make predictions.

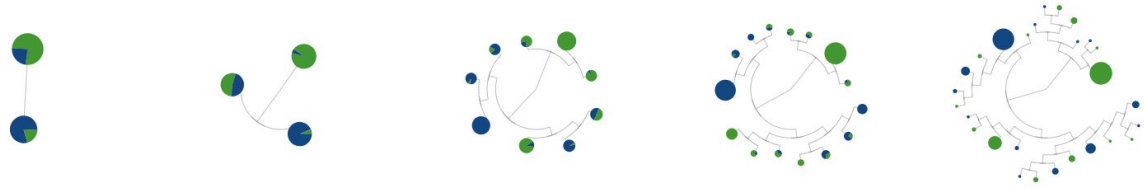
2. One method for making predictions is called a decision trees, which uses a series of if-then statements to identify boundaries and define patterns in the data.

3. **Overfitting** happens when some boundaries are based on on *distinctions that don't make a difference*. You can see if a model overfits by having test data flow through the model.

So...what can we do about overfitting?

Bias-variance tradeoff

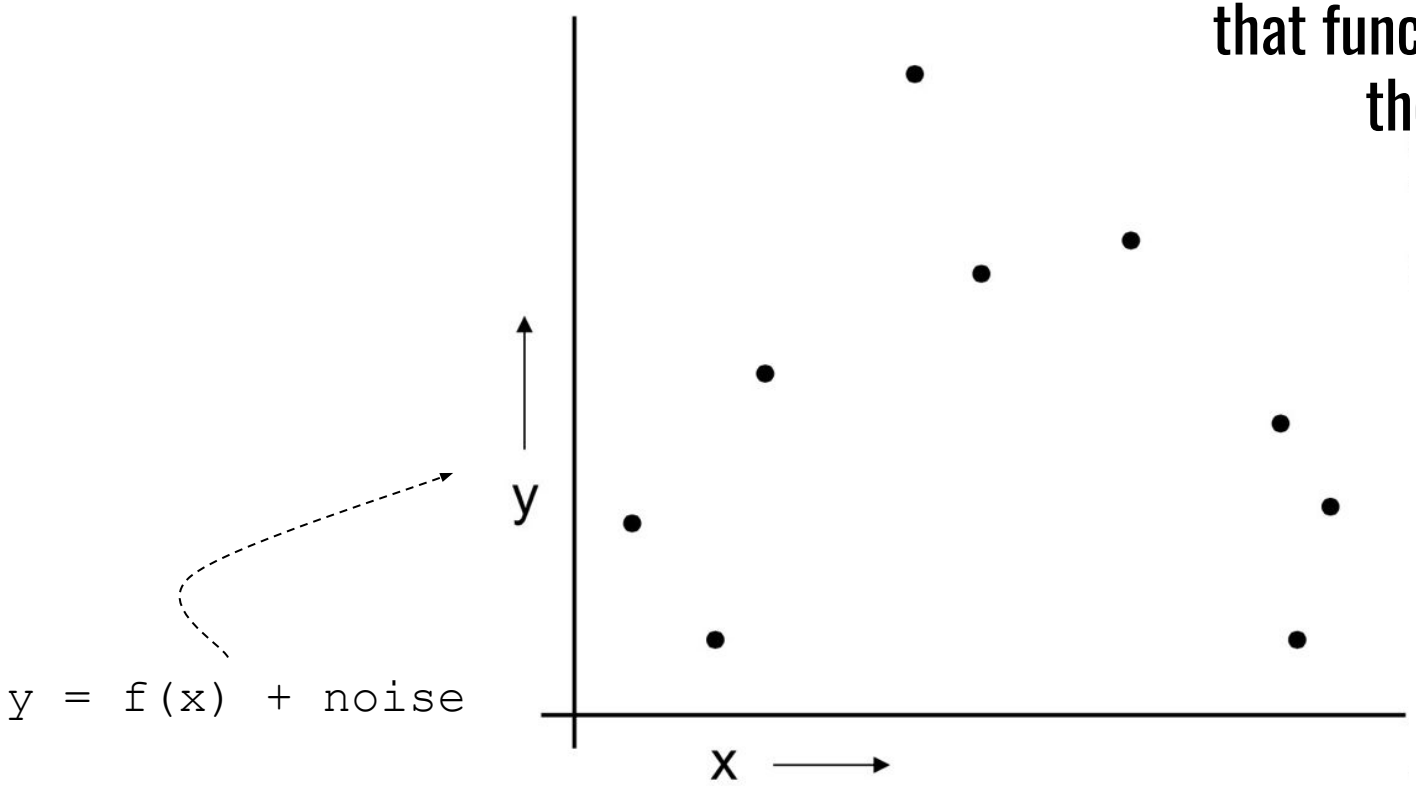




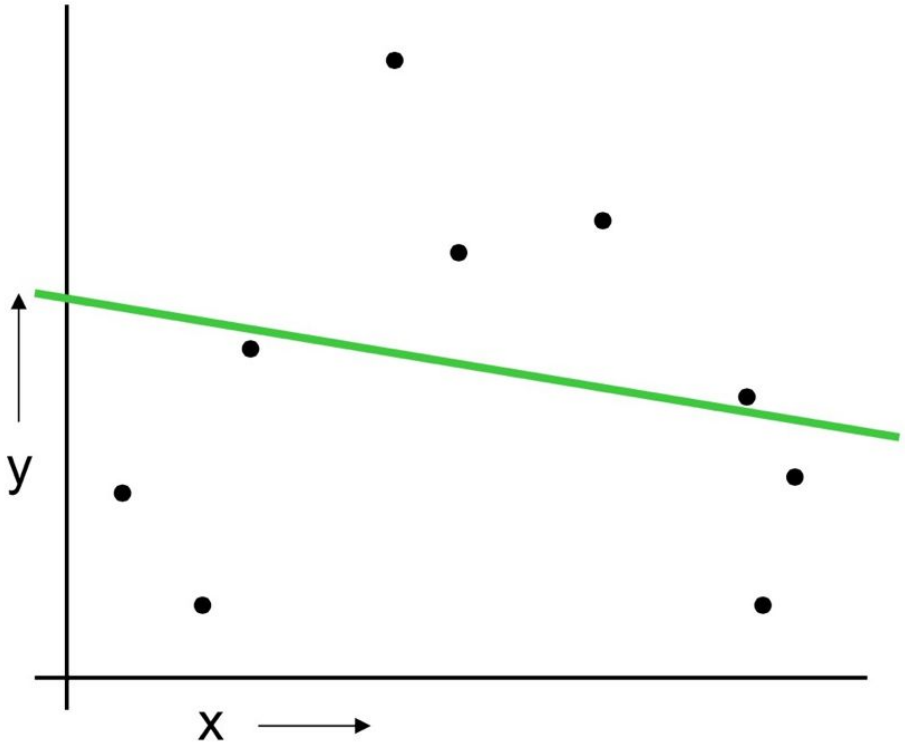
Bias-variance tradeoff

- **High variance** models make mistakes in *inconsistent* ways.
- **Biased models** tend to be overly simple and not reflect reality
- What to do:
  - Consider tuning parameters in the model
    - can avoid overfitting by setting minimum node size threshold (fewer splits; variance decreased)
  - Changing model approach
    - Bagging, boosting, & ensemble methods
  - Re-consider data splitting approach
    - Training + test?
    - LOOCV
    - K-fold CV

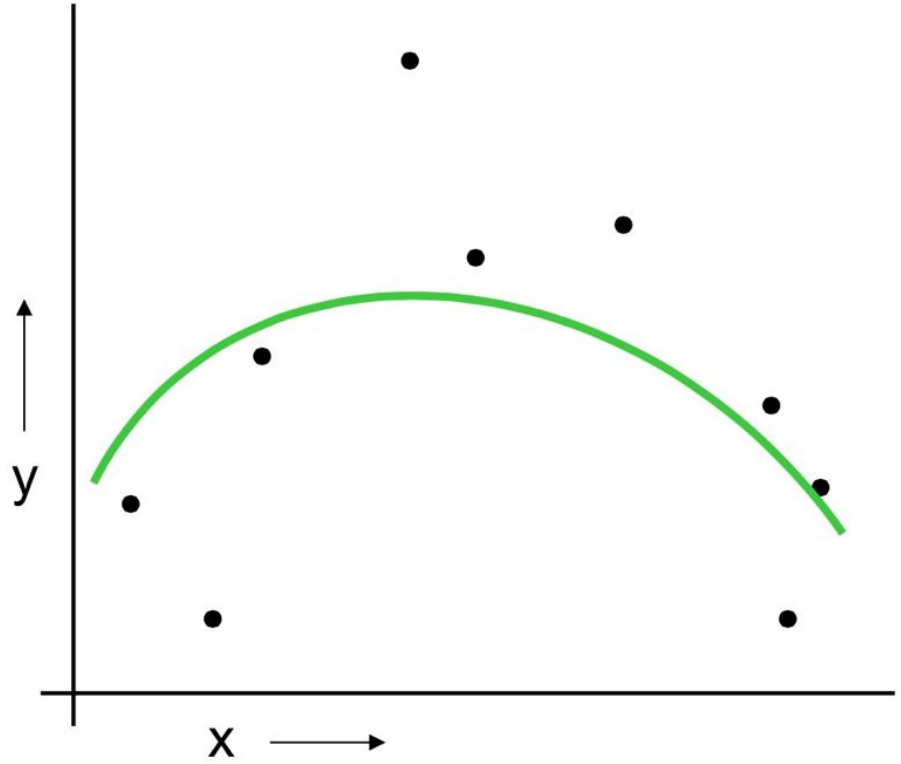
Can we determine what that function (`f`) *is* using these data?



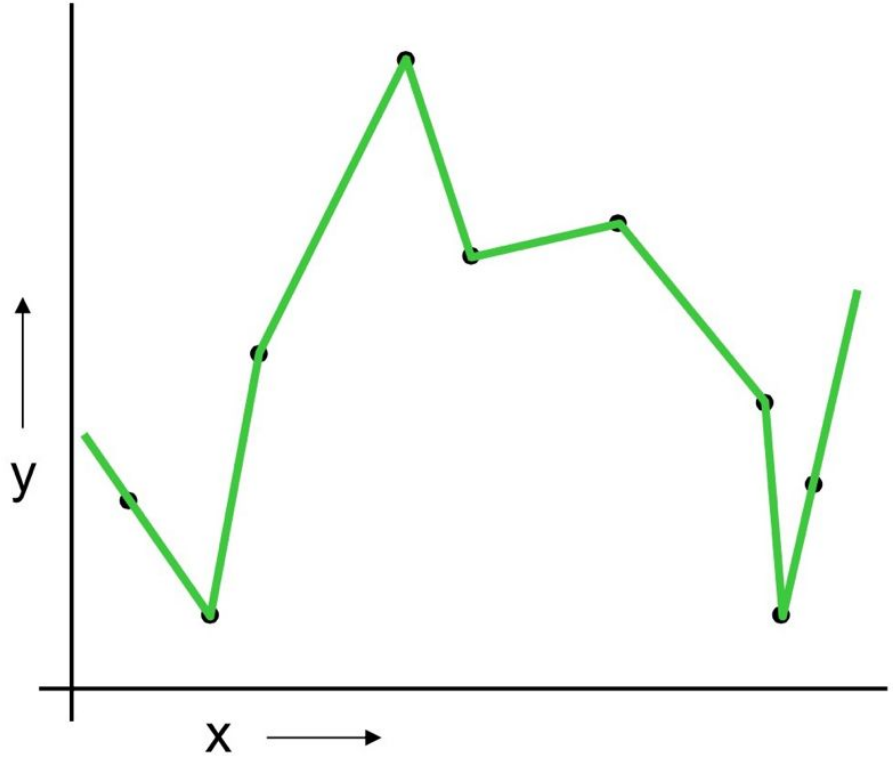
Linear regression



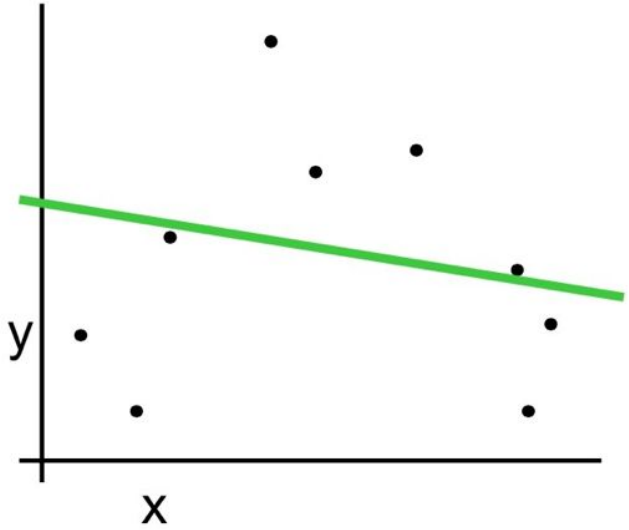
Quadratic regression

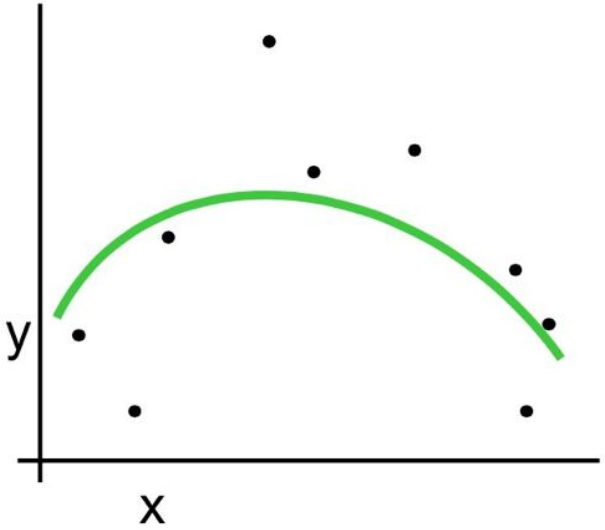


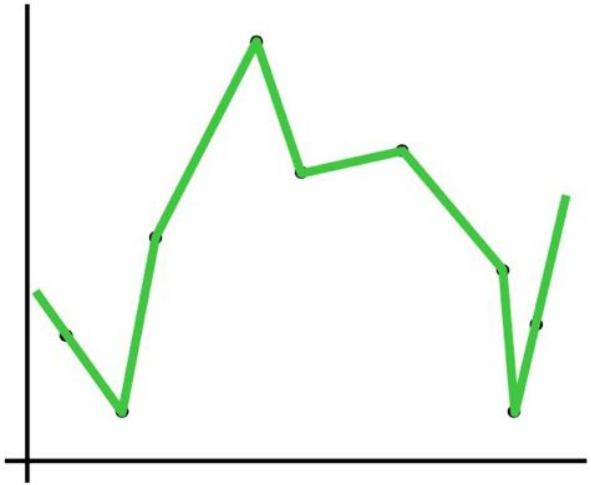
Piecewise linear nonparametric regression



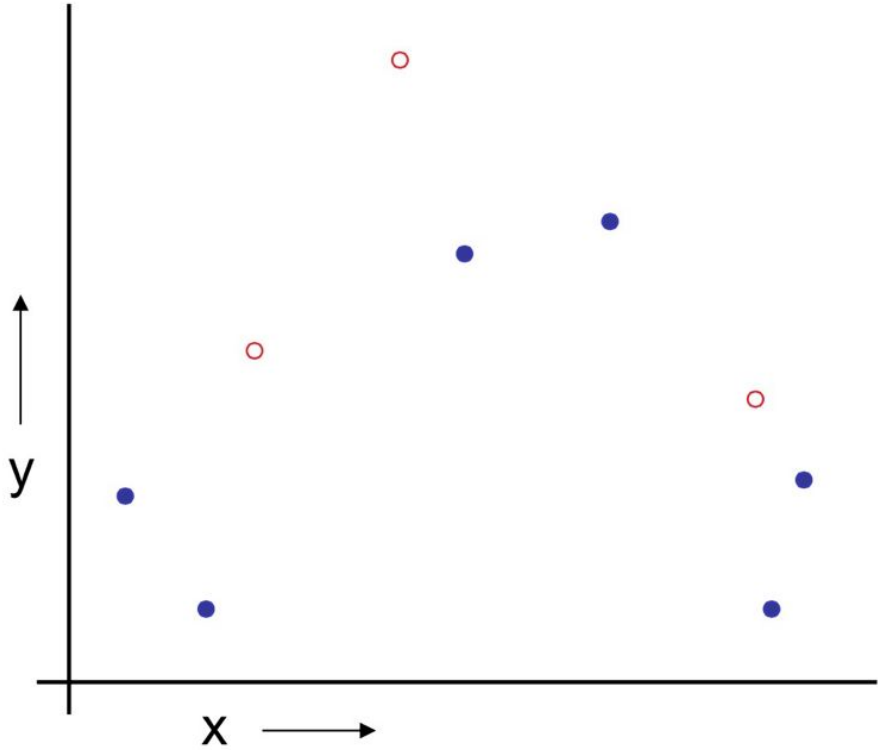
Which to choose?







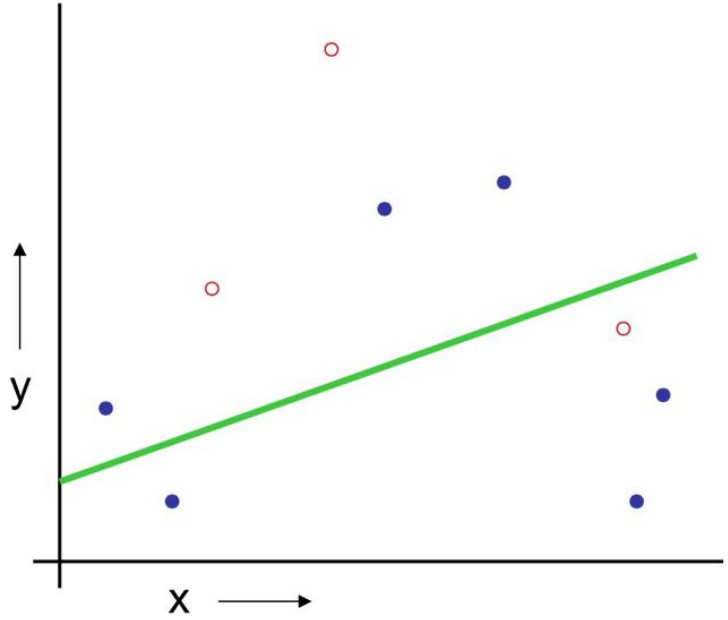
The data partition method



1. Randomly choose 30% of the data to be in a test set

2. The remainder is a training set

Train the model on your training set



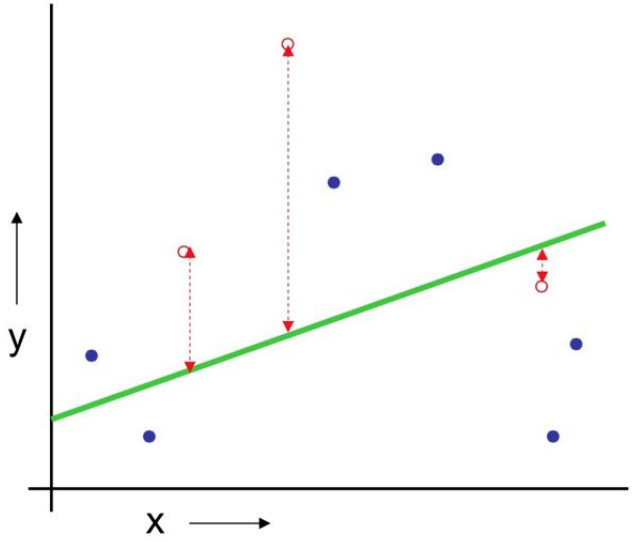
(Linear regression example)

1. Randomly choose 30% of the data to be in a test set

2. The remainder is a training set

3. Perform your regression on the training set

Assess future performance using the test set



(Linear regression example)

Mean Squared Error = 2.4

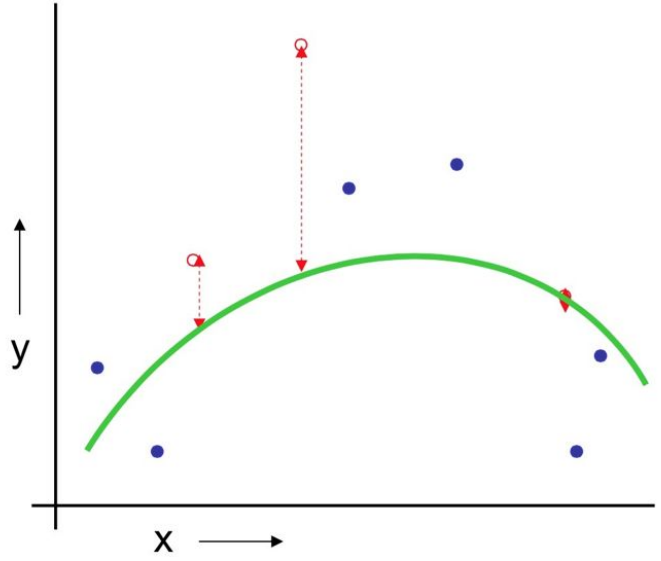
1. Randomly choose 30% of the data to be in a test set

2. The remainder is a training set

3. Perform your regression on the training set

4. Estimate your future performance with the test set

Go through this process for each possible model



(Quadratic regression example)

Mean Squared Error = 0.9

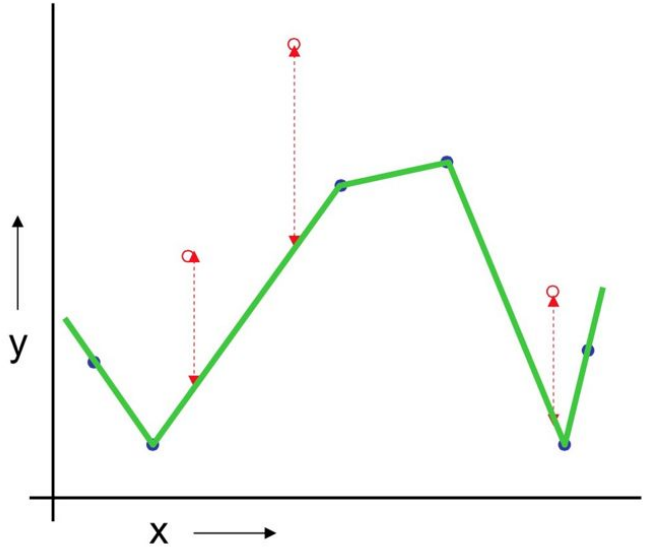
1. Randomly choose 30% of the data to be in a test set

2. The remainder is a training set

3. Perform your regression on the training set

4. Estimate your future performance with the test set

Go through this process for each possible model



(Join the dots example)

Mean Squared Error = 2.2

1. Randomly choose 30% of the data to be in a test set

2. The remainder is a training set

3. Perform your regression on the training set

4. Estimate your future performance with the test set

Pros and cons of data partitioning

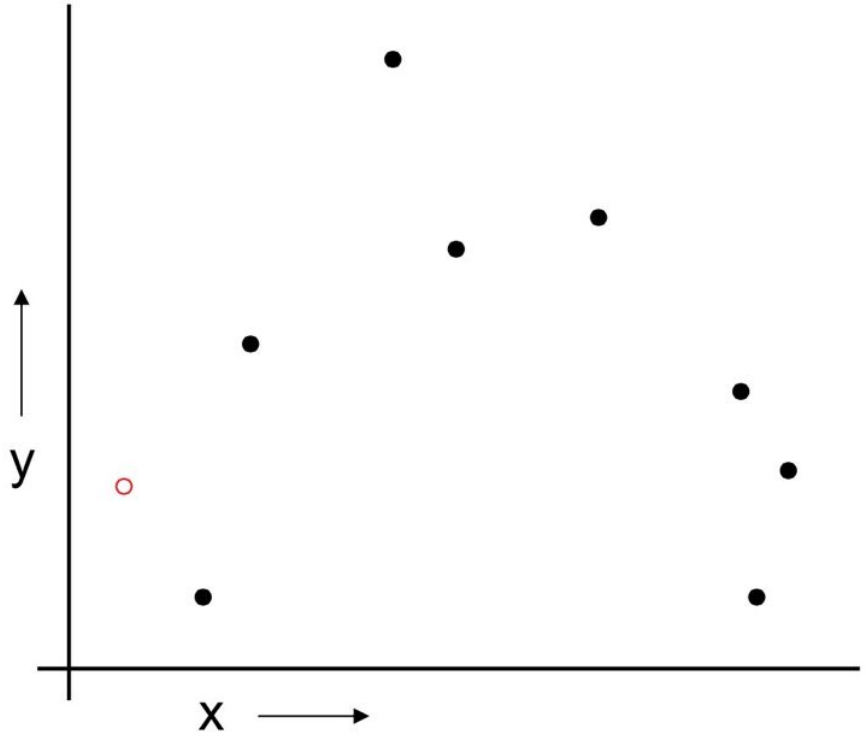
Pros:

- Simple approach
- Can choose model with best test-set score

Cons:

- Model fit on 30% less data than you have
- Without a large data set, removing 30% of the data could bias prediction

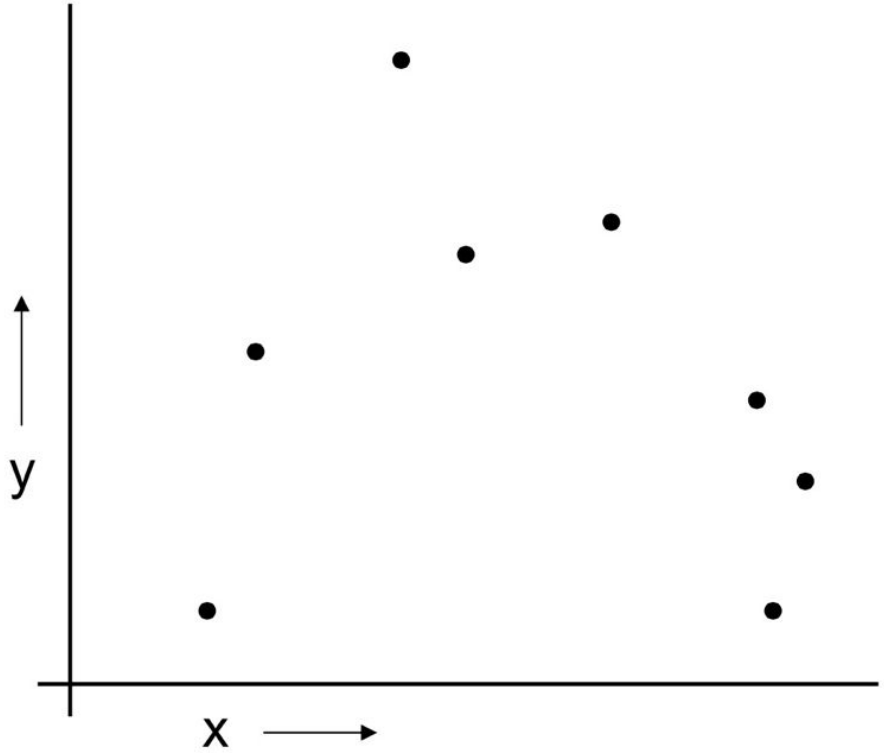
Leave one out cross validation (LOOCV)



For k=1 to R

1. Let (x_k, y_k) be the k^{th} record

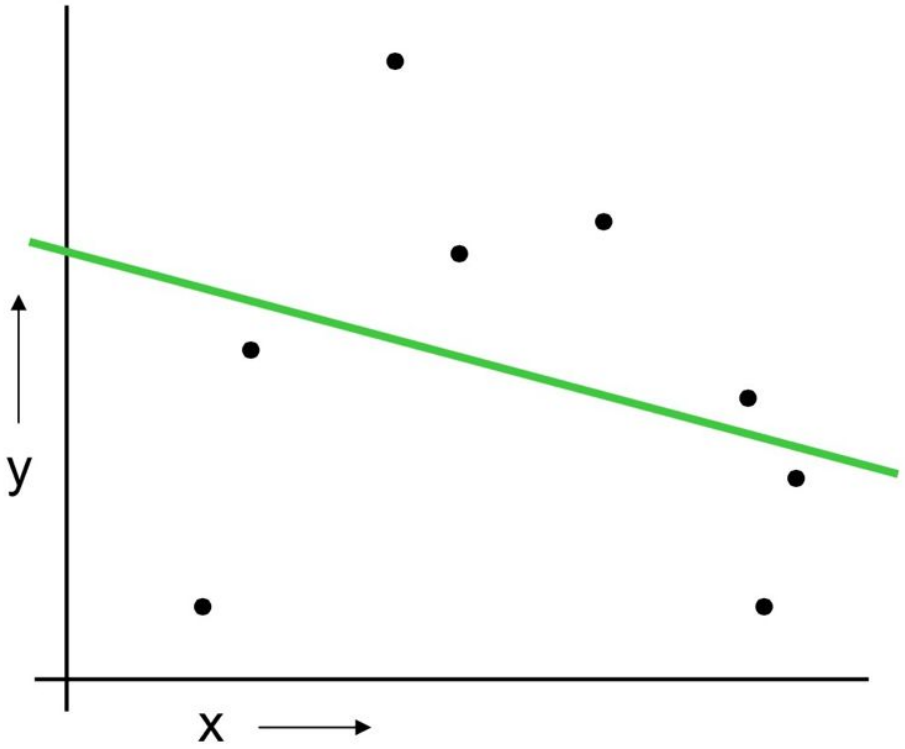
Leave one out cross validation (LOOCV)



For k=1 to R

1. Let (x_k, y_k) be the k^{th} record
2. Temporarily remove (x_k, y_k) from the dataset

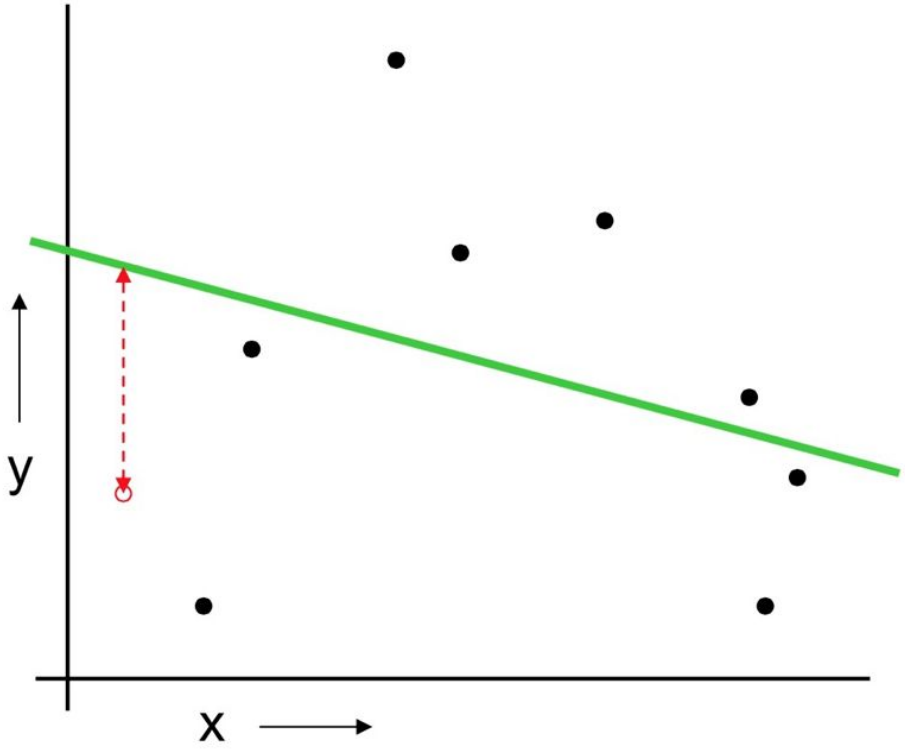
Leave one out cross validation (LOOCV)



For k=1 to R

1. Let (x_k, y_k) be the k^{th} record
2. Temporarily remove (x_k, y_k) from the dataset
3. Train on the remaining R-1 datapoints

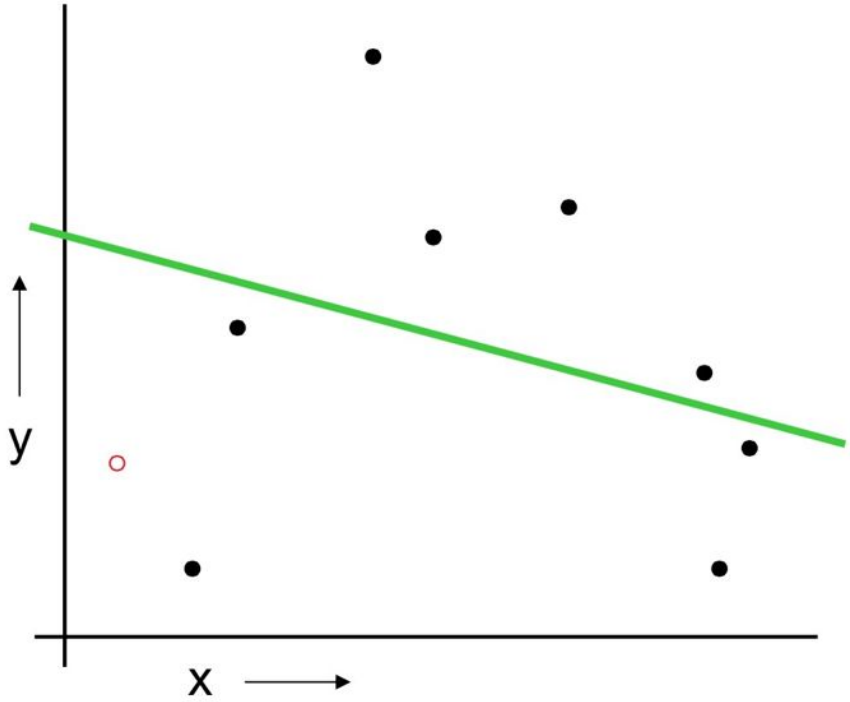
Leave one out cross validation (LOOCV)



For k=1 to R

1. Let (x_k, y_k) be the k^{th} record
2. Temporarily remove (x_k, y_k) from the dataset
3. Train on the remaining R-1 datapoints
4. Note your error (x_k, y_k)

Leave one out cross validation (LOOCV)

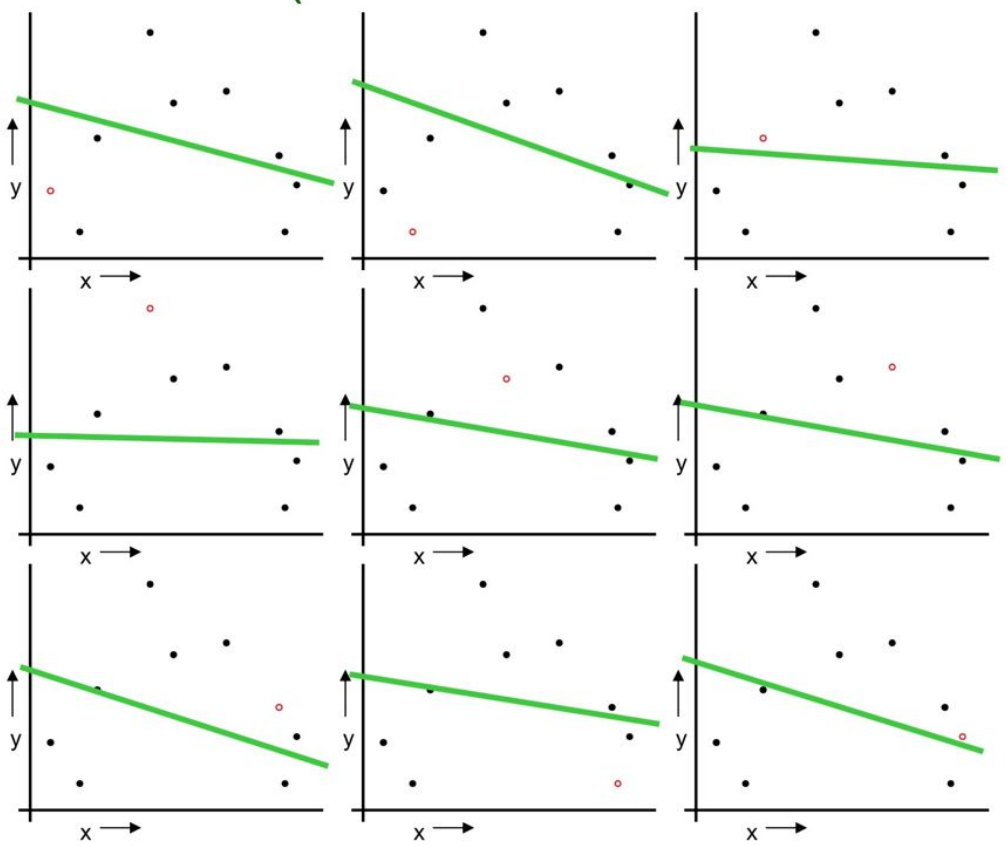


For k=1 to R

1. Let (x_k, y_k) be the k^{th} record
2. Temporarily remove (x_k, y_k) from the dataset
3. Train on the remaining R-1 datapoints
4. Note your error (x_k, y_k)

When you've done all points, report the mean error.

Leave one out cross validation (LOOCV)



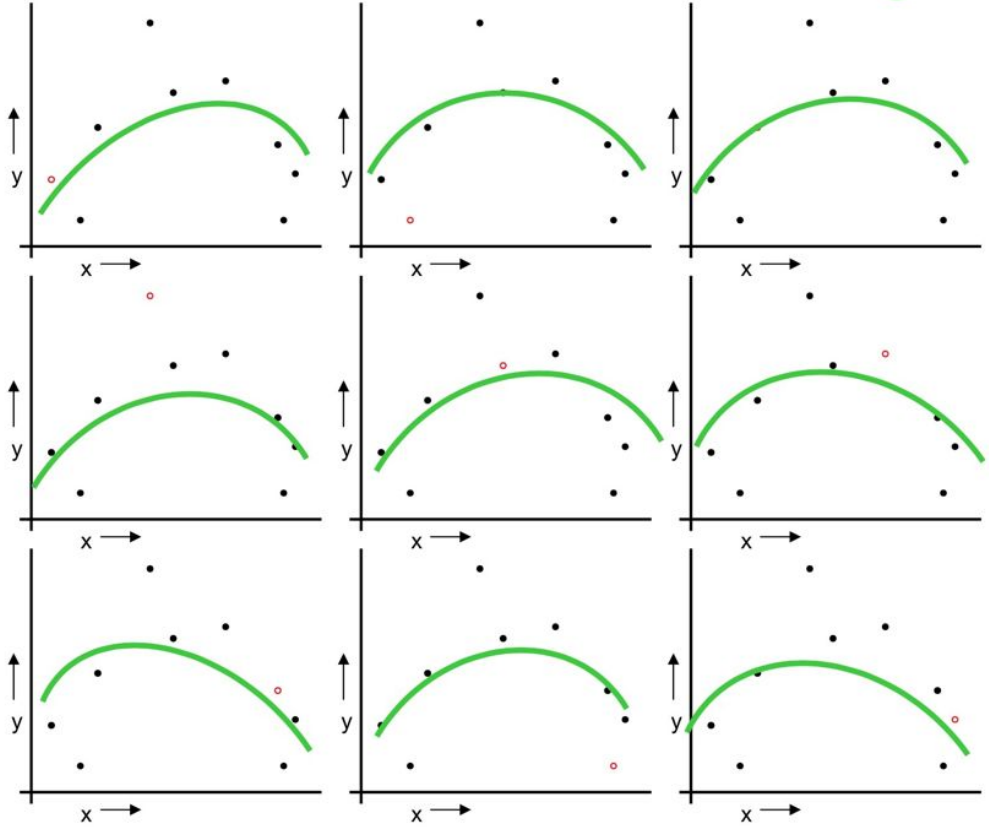
For k=1 to R

1. Let (x_k, y_k) be the k^{th} record
2. Temporarily remove (x_k, y_k) from the dataset
3. Train on the remaining R-1 datapoints
4. Note your error (x_k, y_k)

When you've done all points, report the mean error.

$MSE_{LOOCV} = 2.12$

Leave one out cross validation (LOOCV)



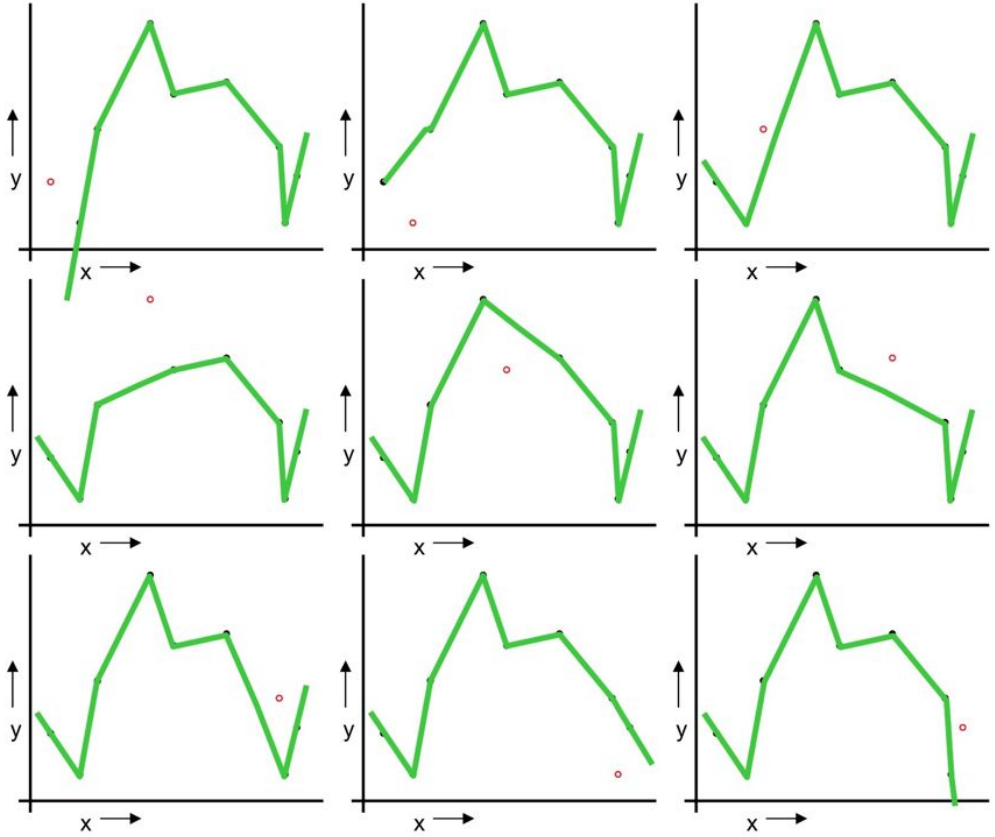
For k=1 to R

1. Let (x_k, y_k) be the k^{th} record
2. Temporarily remove (x_k, y_k) from the dataset
3. Train on the remaining R-1 datapoints
4. Note your error (x_k, y_k)

When you've done all points, report the mean error.

$MSE_{LOOCV} = 0.962$

Leave one out cross validation (LOOCV)



For k=1 to R

1. Let (x_k, y_k) be the k^{th} record
2. Temporarily remove (x_k, y_k) from the dataset
3. Train on the remaining R-1 datapoints
4. Note your error (x_k, y_k)

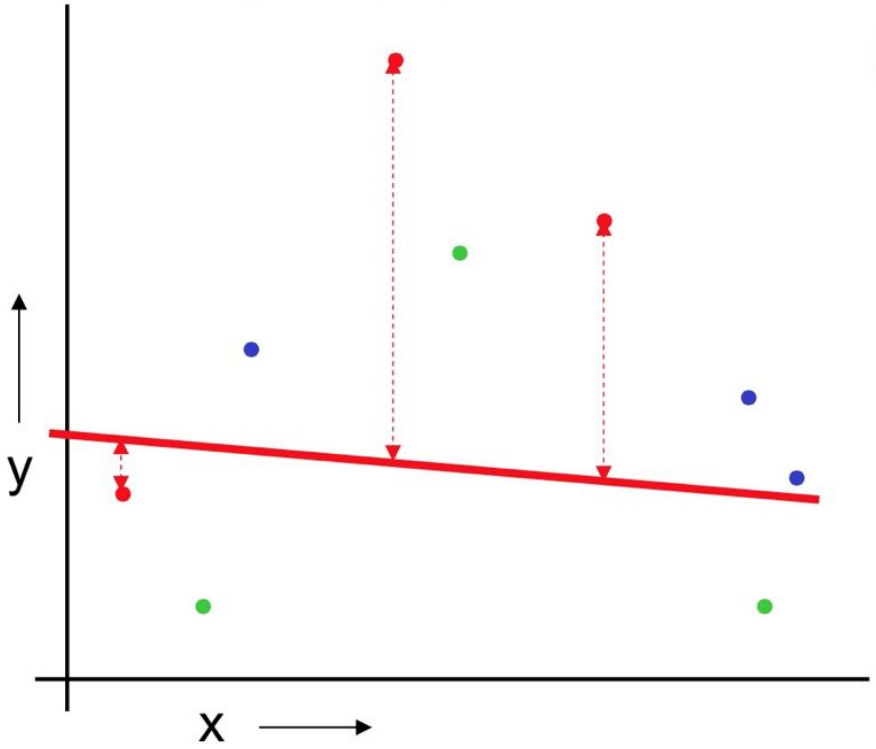
When you've done all points, report the mean error.

$MSE_{LOOCV} = 3.33$

Method Comparison

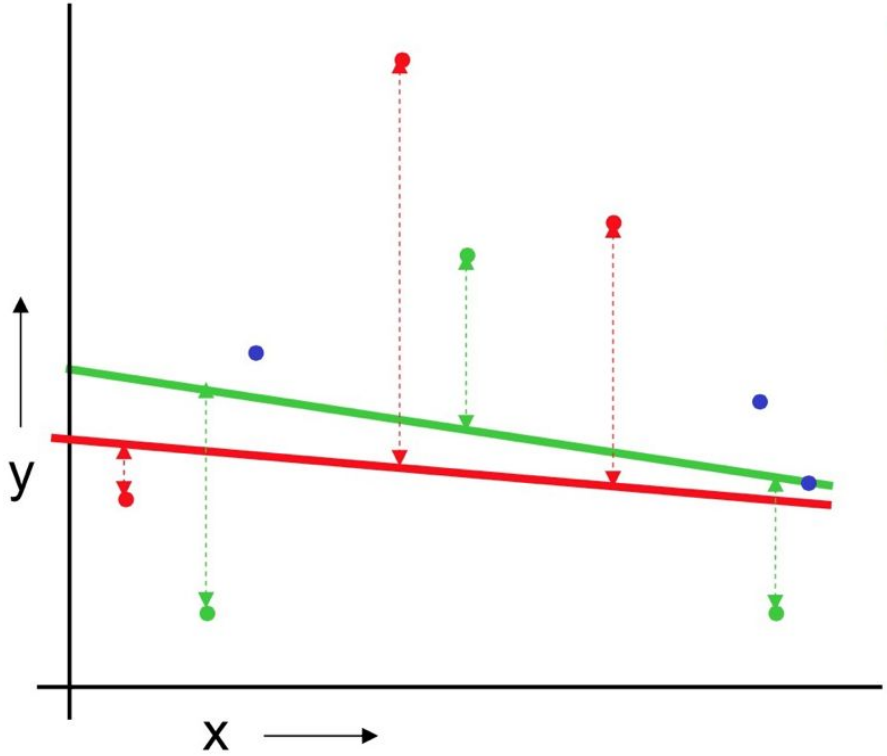
	Cons	Pros
Data partitioning	Variance: unreliable estimate of future performance	Cheap
LOOCV	Computationally expensive; has weird behavior	Uses all your data

k-fold cross validation



For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

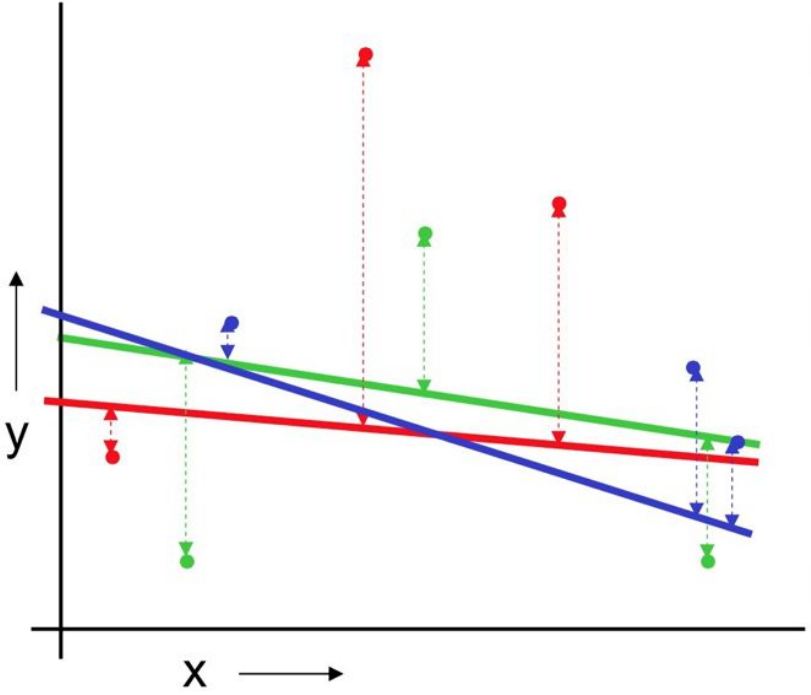
k-fold cross validation



For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

k-fold cross validation

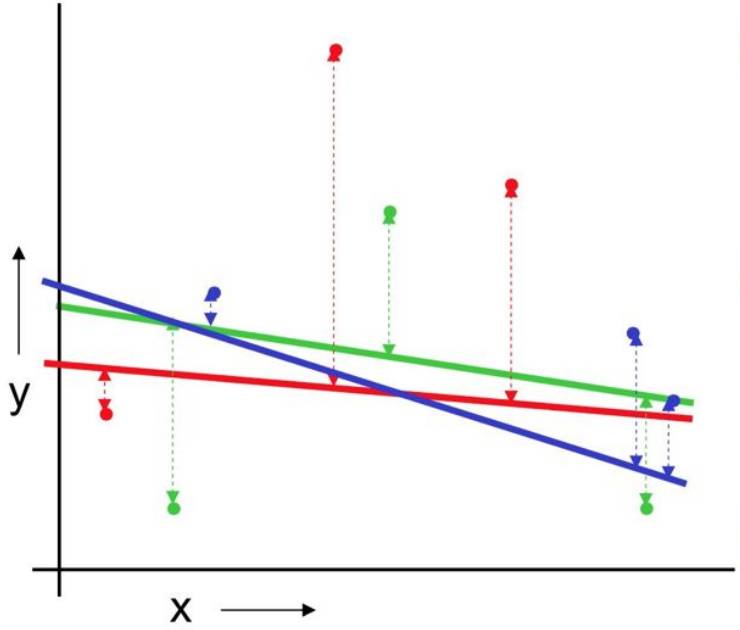


For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

For the blue partition: Train on all the points not in the blue partition. Find the test-set sum of errors on the blue points.

k-fold cross validation



Linear Regression

$MSE_{3FOLD}=2.05$

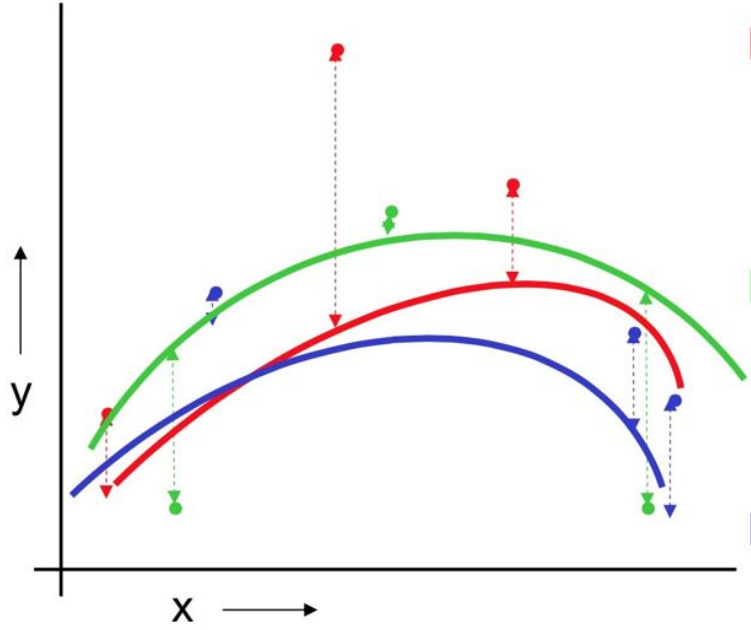
For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

For the blue partition: Train on all the points not in the blue partition. Find the test-set sum of errors on the blue points.

Then report the mean error

k-fold cross validation



Quadratic Regression

$MSE_{3FOLD}=1.11$

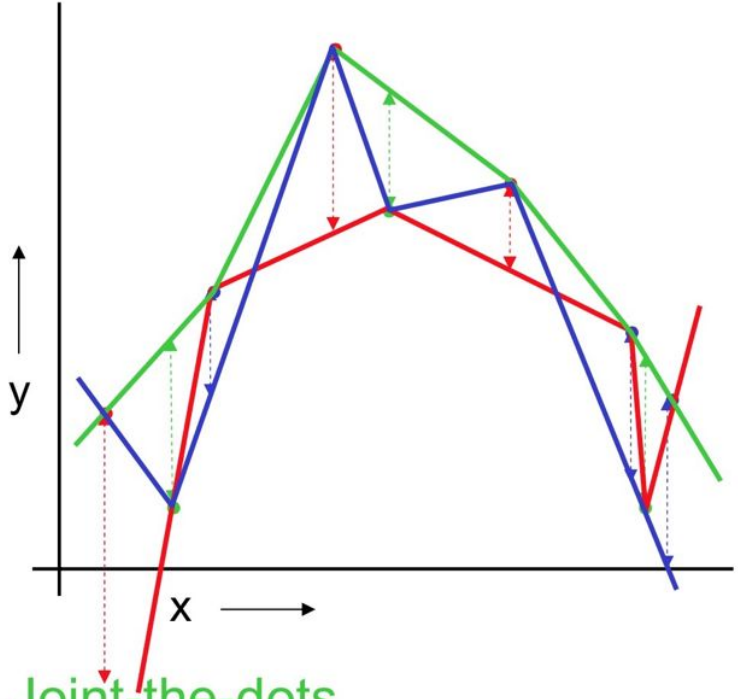
For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

For the blue partition: Train on all the points not in the blue partition. Find the test-set sum of errors on the blue points.

Then report the mean error

k-fold cross validation



Joint-the-dots

$MSE_{3FOLD}=2.93$

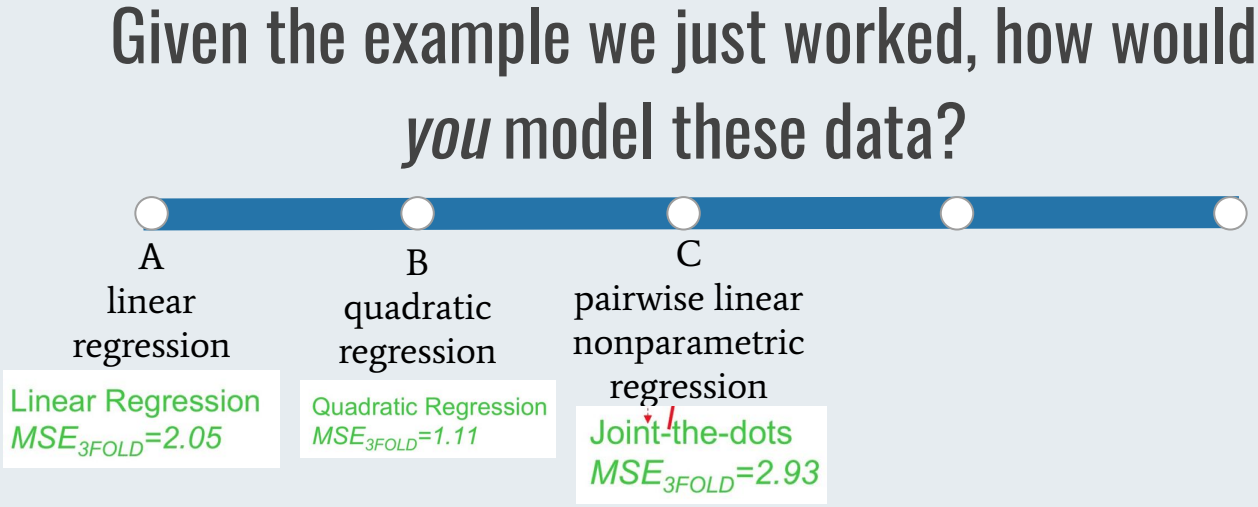
For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

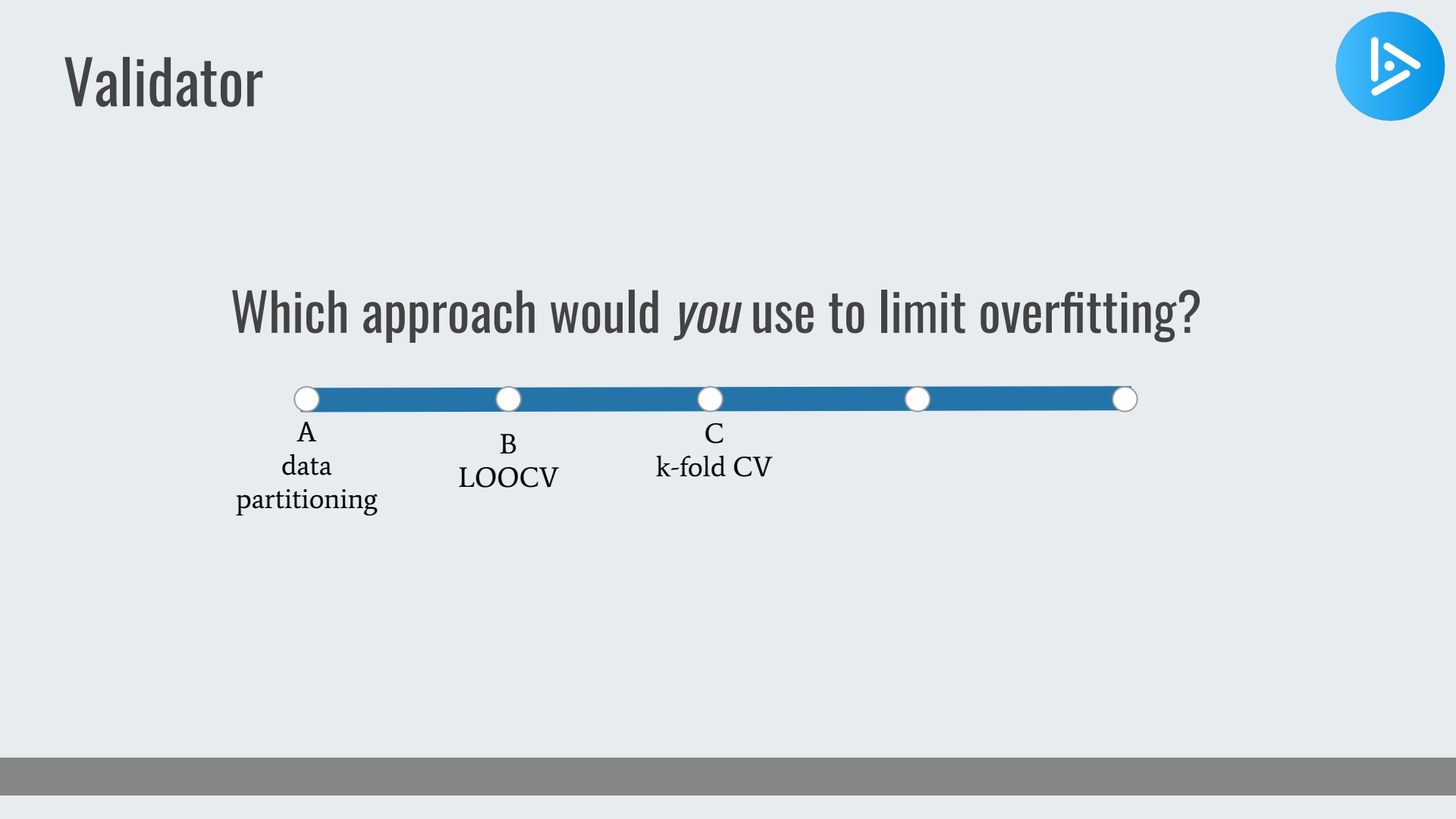
For the blue partition: Train on all the points not in the blue partition. Find the test-set sum of errors on the blue points.

Then report the mean error

Validator


Given the example we just worked, how would *you* model these data?
A
linear regression
Linear Regression
MSE_{3FOLD}=2.05
B
quadratic regression
Quadratic Regression
MSE_{3FOLD}=1.11
C
pairwise linear nonparametric regression
Joint-the-dots
MSE_{3FOLD}=2.93

Validator

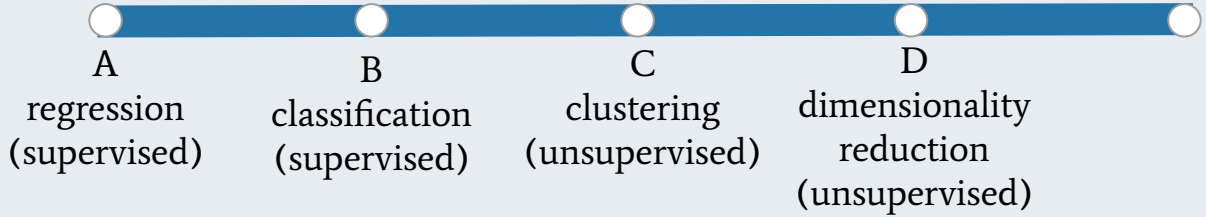

Which approach would *you* use to limit overfitting?
A
data
partitioning
B
LOOCV
C
k-fold CV

Worked Example II:

Categorizing Listing Photos at Airbnb

Prediction Approach

What type of machine learning task is "Categorizing Listing Photos at Airbnb?"



Shijing Yao Follow
Senior Machine Learning Scientist @ Airbnb
May 2 · 11 min read

Categorizing Listing Photos at Airbnb

Large-scale deep learning models are changing the way we think about images of homes on our platform.

Authors: *Shijing Yao, Qiang Zhu, Phillippe Siclait*

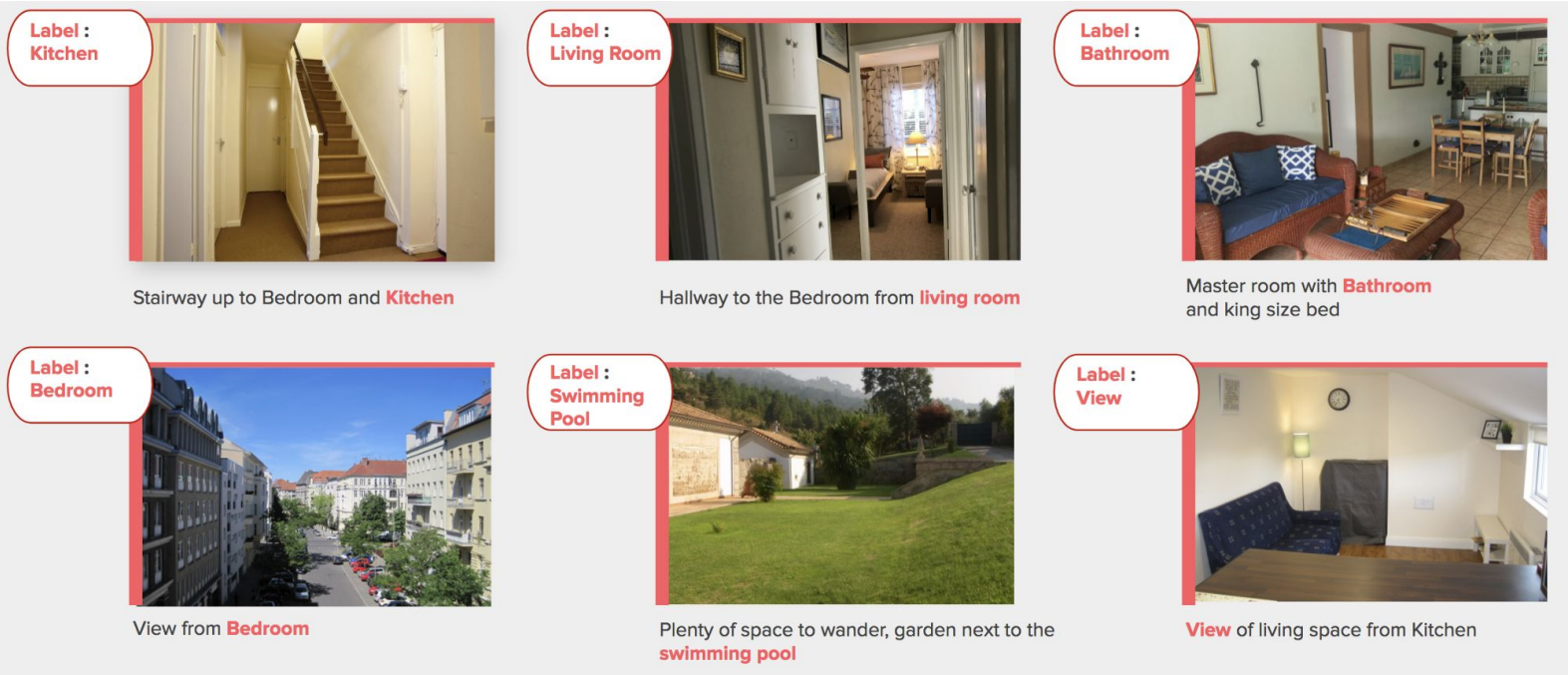
https://medium.com/airbnb-engineering/categorizing-listing-photos-at-airbnb-f9483f3ab7e3

The Case for Why

1. Ease of home tour
    - optimizing the user experience
    - photos with the same room type can be grouped together
    - Allows airbnb to rank/group - display photos users will want to see first
2. Efficient listing validation
    - validate the number of certain rooms
    - check whether the basic room information submitted is correct
    - Ensure listings abide by our marketplace’s high standards

https://medium.com/airbnb-engineering/categorizing-listing-photos-at-airbnb-f9483f3ab7e3

Why can't you just trust what people typed on their own photos and train a supervised model?



"A tempting method to extract room-type label from image caption is as follows: If a certain room type keyword is found in the caption of an image, the image will be labeled as that type."

https://medium.com/airbnb-engineering/categorizing-listing-photos-at-airbnb-f9483f3ab7e3

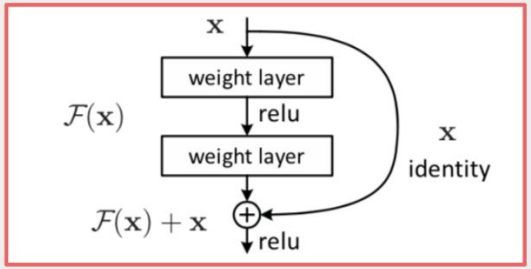
Getting labels for model assessment

"On one side, we *asked vendors to label relatively small number of photos*, usually in thousands or tens of thousands. This chunk of labeled data would be used as a golden set for us to evaluate models.We used random sampling to get this golden set and ensured the data was unbiased."

"On the other side, we *leveraged image captions created by hosts as a proxy for room-type information* and extracted labels out of it. This idea was huge for us because it made the expensive labeling task essentially free. We only needed a judicious way to ensure that room-type labels extracted from image caption were accurate and reliable."

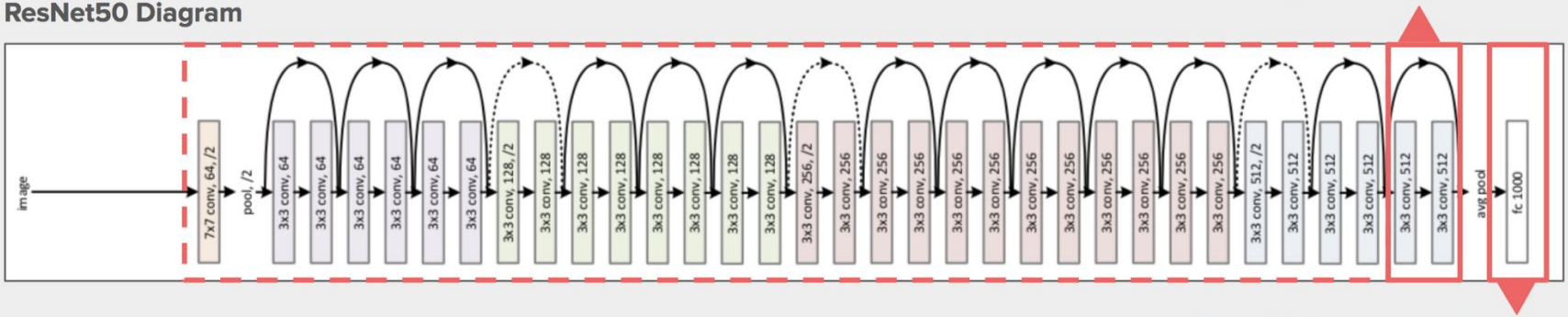
https://medium.com/airbnb-engineering/categorizing-listing-photos-at-airbnb-f9483f3ab7e3

Retrain ResNet50

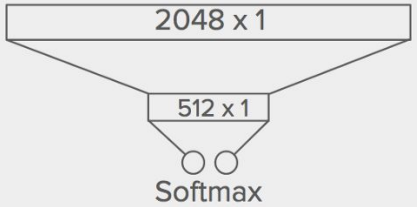


Residual Learning Block

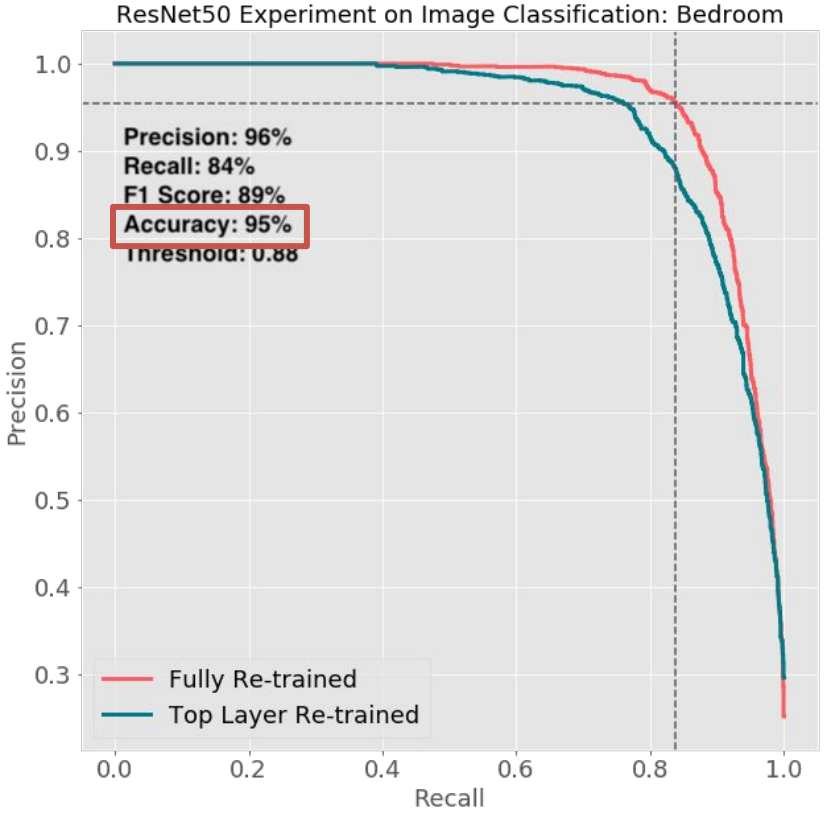
ResNet50 Diagram

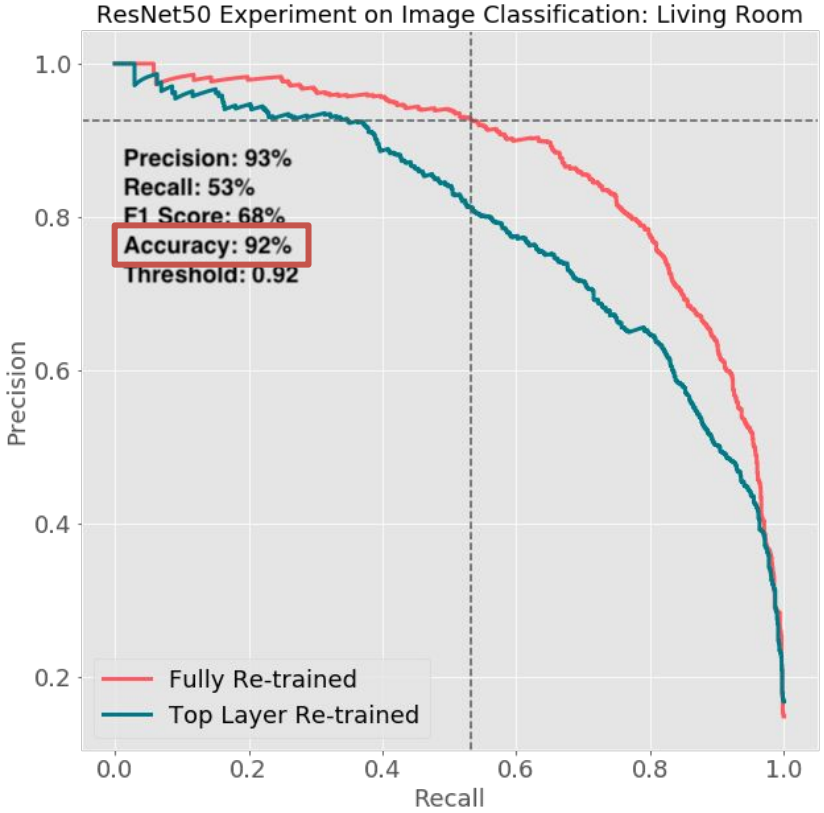


Re-architect fully-connected layers



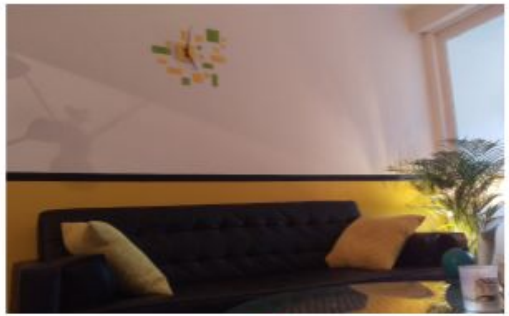
https://medium.com/airbnb-engineering/categorizing-listing-photos-at-airbnb-f9483f3ab7e3





https://medium.com/airbnb-engineering/categorizing-listing-photos-at-airbnb-f9483f3ab7e3

Bedroom Or Not?



“The left two photos were correctly predicted as bedrooms; The right two photos were correctly predicted NOT as bedrooms.”

https://medium.com/airbnb-engineering/categorizing-listing-photos-at-airbnb-f9483f3ab7e3

Bathroom Or Not?

https://medium.com/airbnb-engineering/categorizing-listing-photos-at-airbnb-f9483f3ab7e3

Pool Or Not?

https://medium.com/airbnb-engineering/categorizing-listing-photos-at-airbnb-f9483f3ab7e3