

# ML: Example & Ethics

C. Alex Simpkins Jr., Ph.D  
UC San Diego, RDPRobotics LLC



Department of Cognitive Science  
[rdrobotics@gmail.com](mailto:rdrobotics@gmail.com)  
[csimpkinsjr@ucsd.edu](mailto:csimpkinsjr@ucsd.edu)

Lectures : <https://github.com/COGS108/Lectures-Wi23>

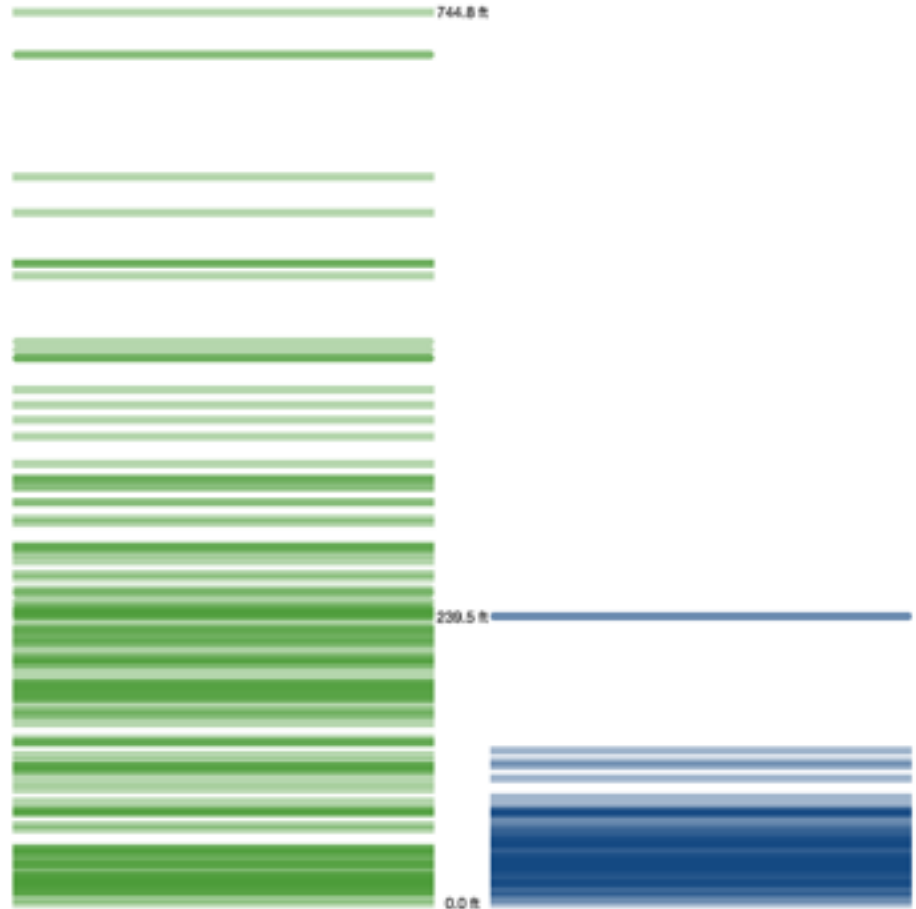
What features distinguish  
a house in New York from  
a house in San Francisco?

---

# 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  $> \sim 73\text{m}$  should be classified as San Fran homes



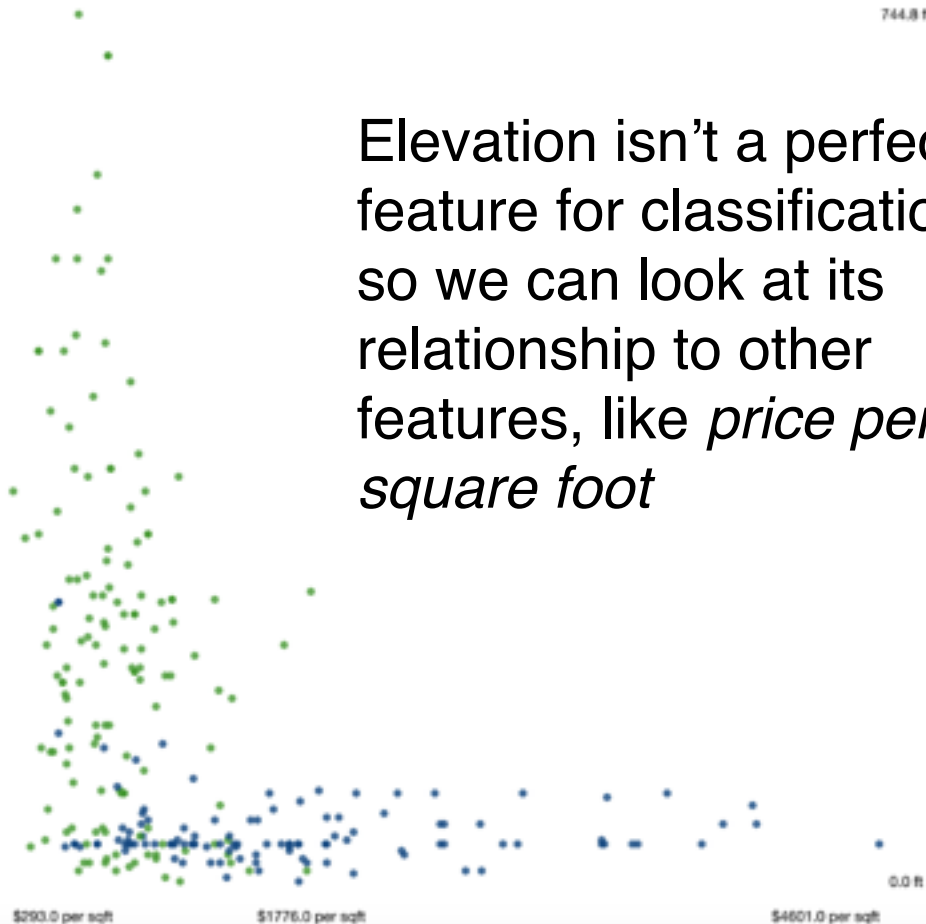
# Adding nuance

Adding another **dimension** allows for more nuance. For example, New York apartments can be extremely expensive per square foot.

So visualizing elevation *and* price per square foot in a **scatterplot** helps us distinguish lower-elevation homes.

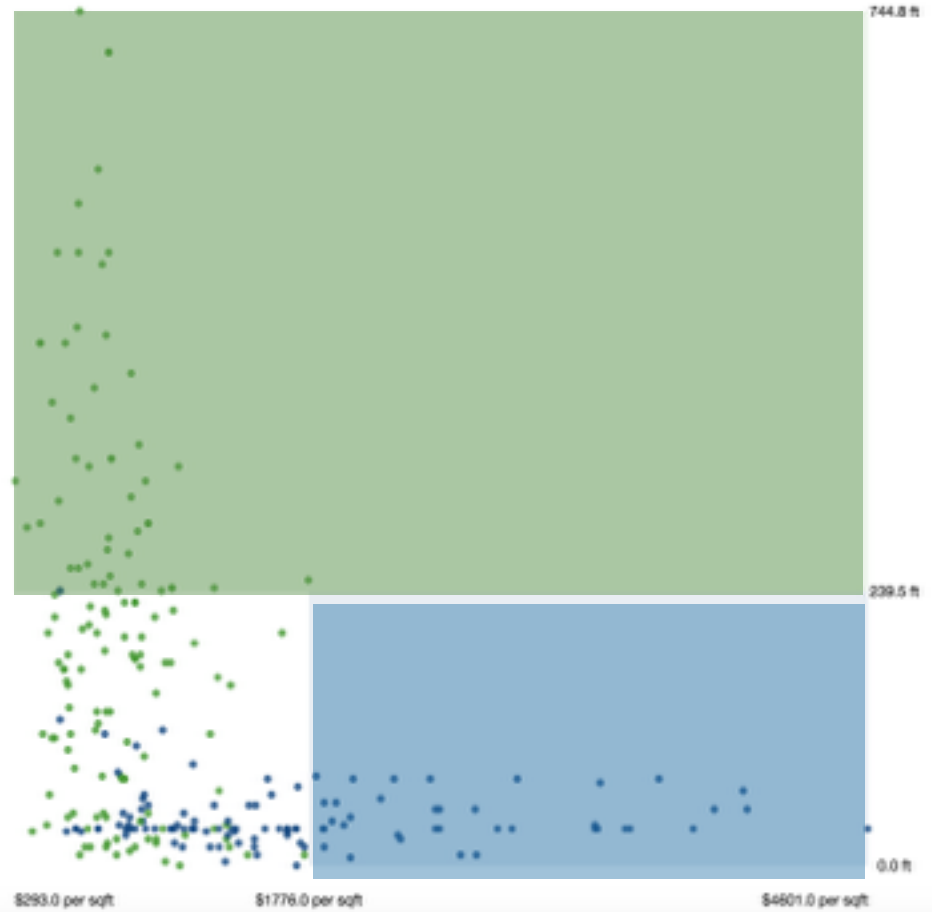
The data suggests that, among homes at or below 73 meters, those that cost more than \$19,116.7 per square meter are in New York City.

Dimensions in a data set are called **features**, **predictors**, or **variables**. 1



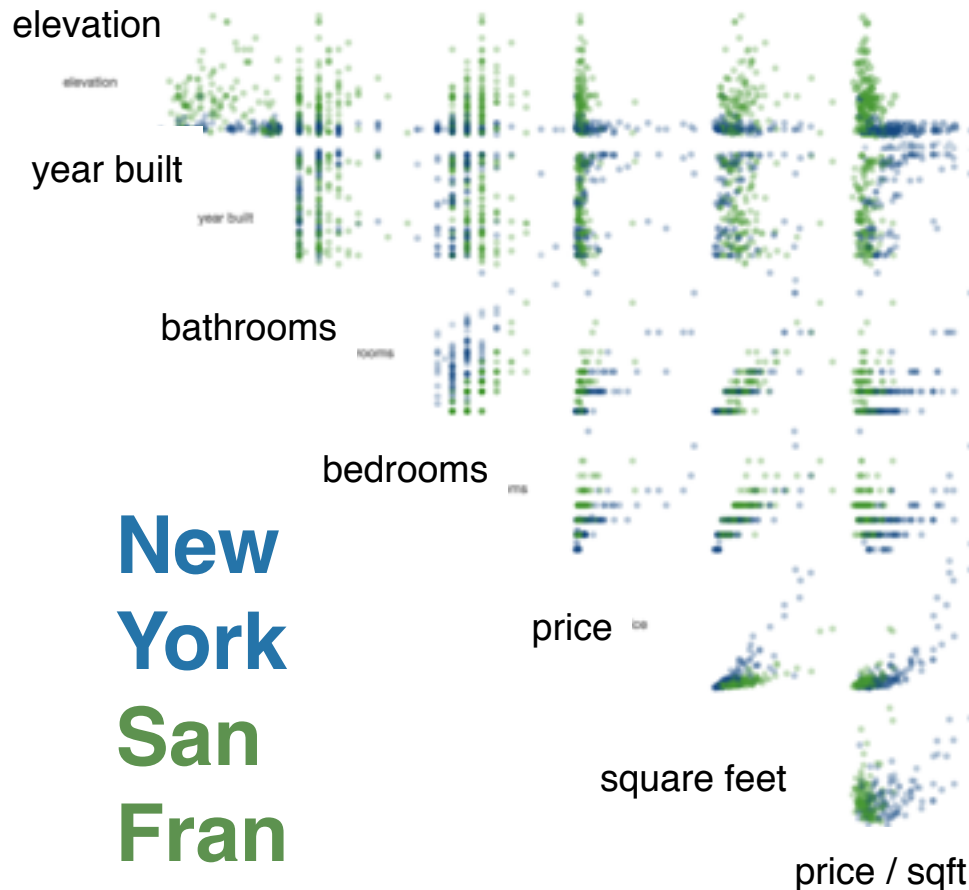
## Drawing boundaries

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.



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.



Our training set has  
different **features**  
we see the **scatter plot**  
**matrix** of the relationships  
between these features.  
Patterns are clear  
boundaries for decision  
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.

train





Finding better  
boundaries

---

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



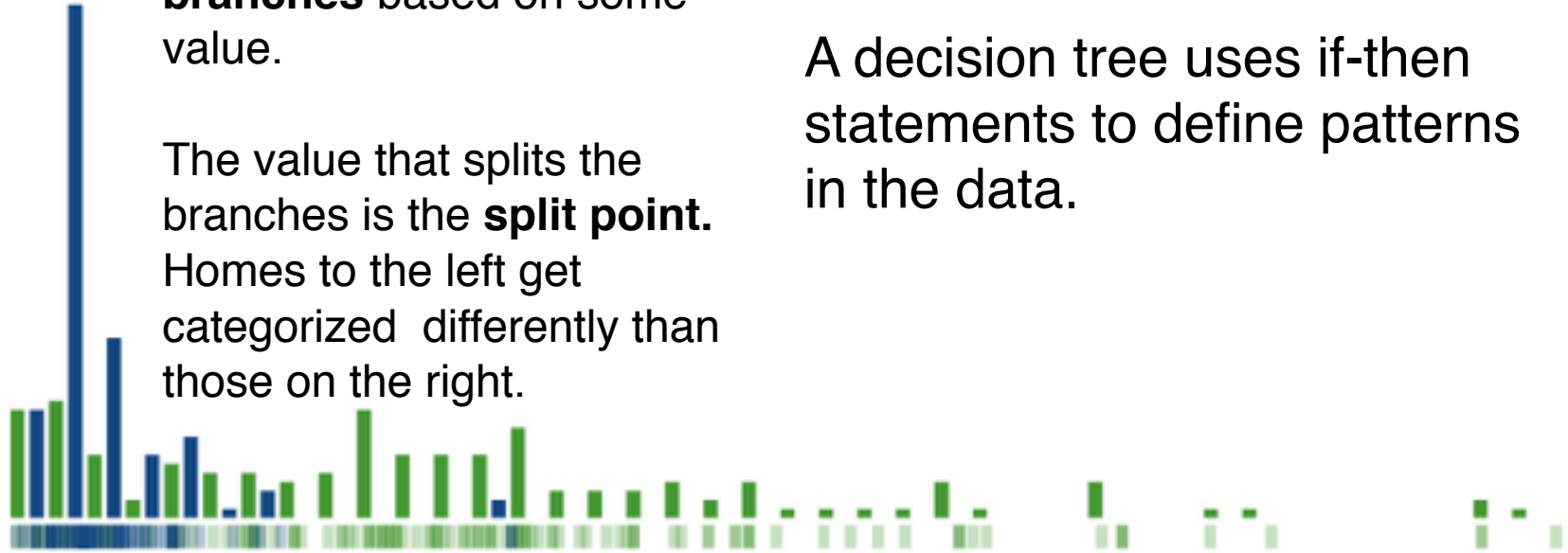
A **histogram** helps display frequency of homes by elevation more easily.

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



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.

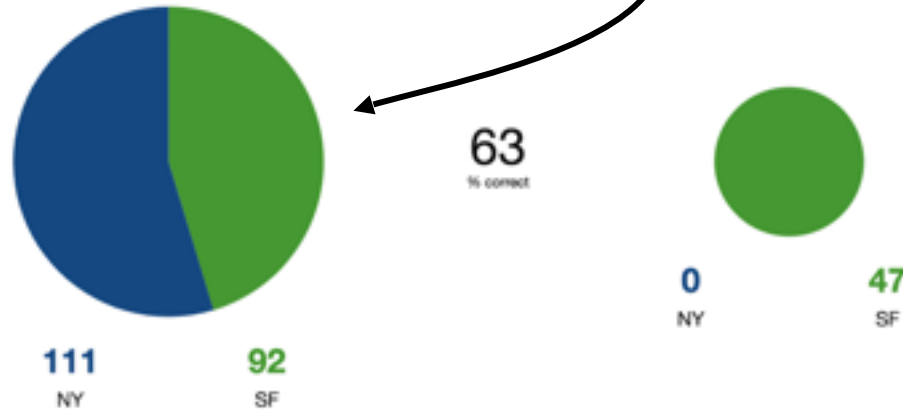
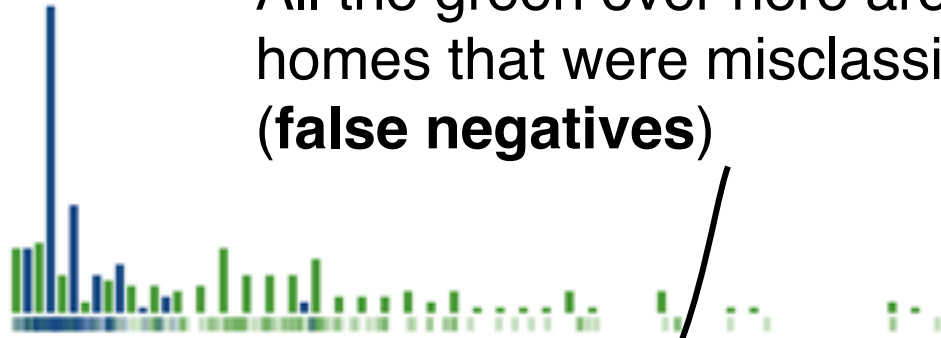


## Your first fork

---

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

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



## Tradeoffs

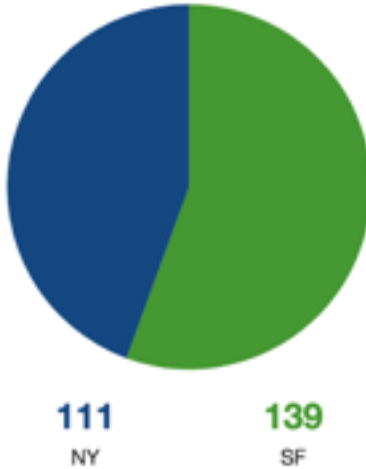
Splitting at ~73m  
incorrectly  
classifies some  
San Francisco  
homes as New  
York homes.



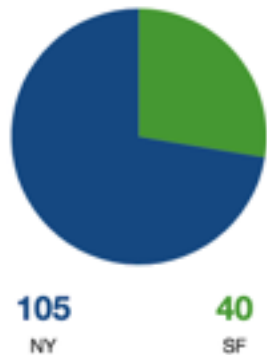
0  
NY

0  
SF

56  
% correct



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

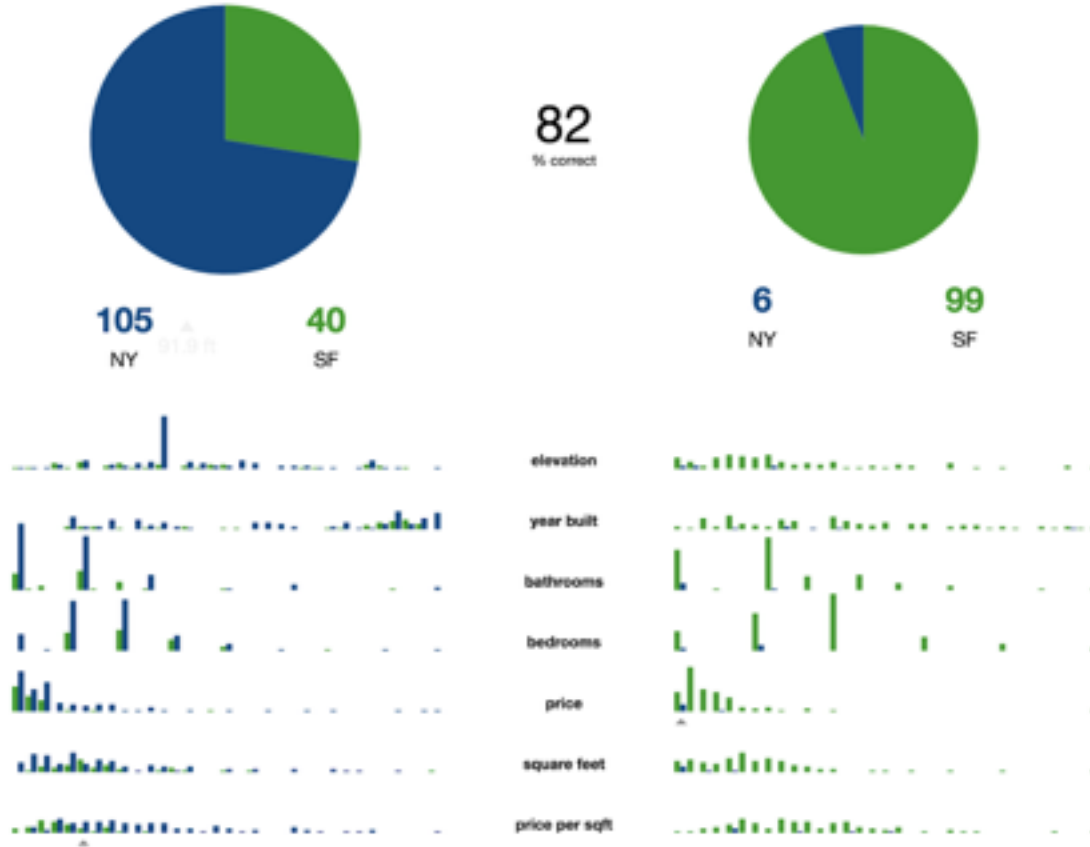


82  
% correct



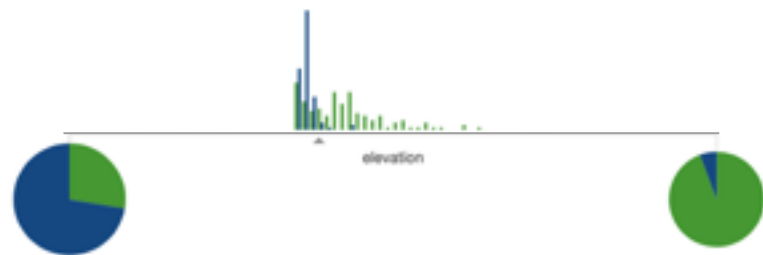
## The best split

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%

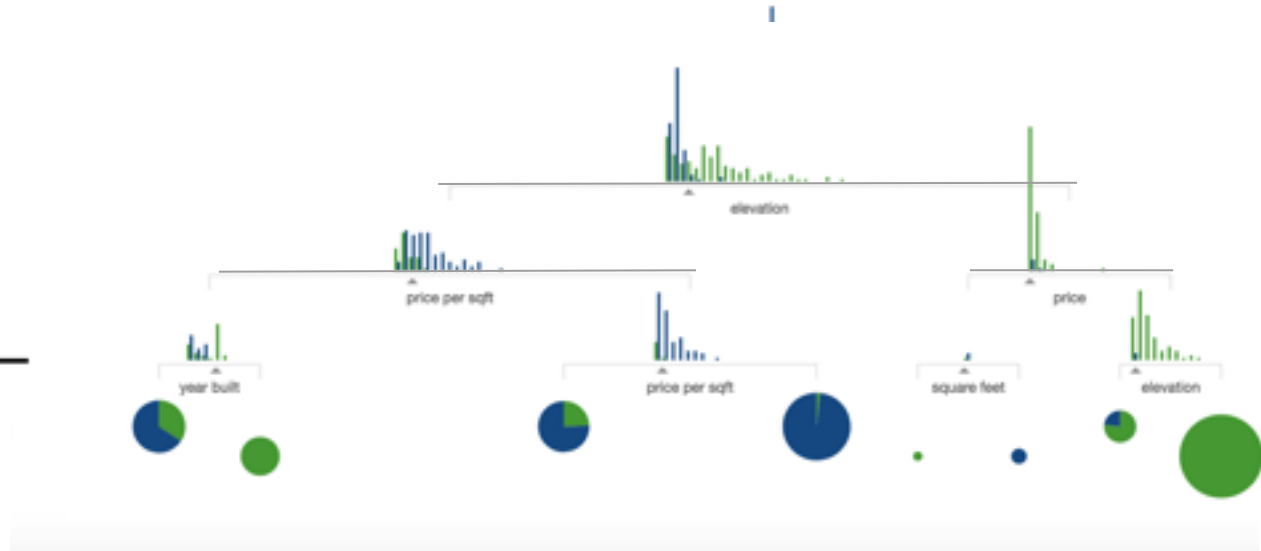
Additional forks  
add new  
information to  
improve  
**prediction  
accuracy.**



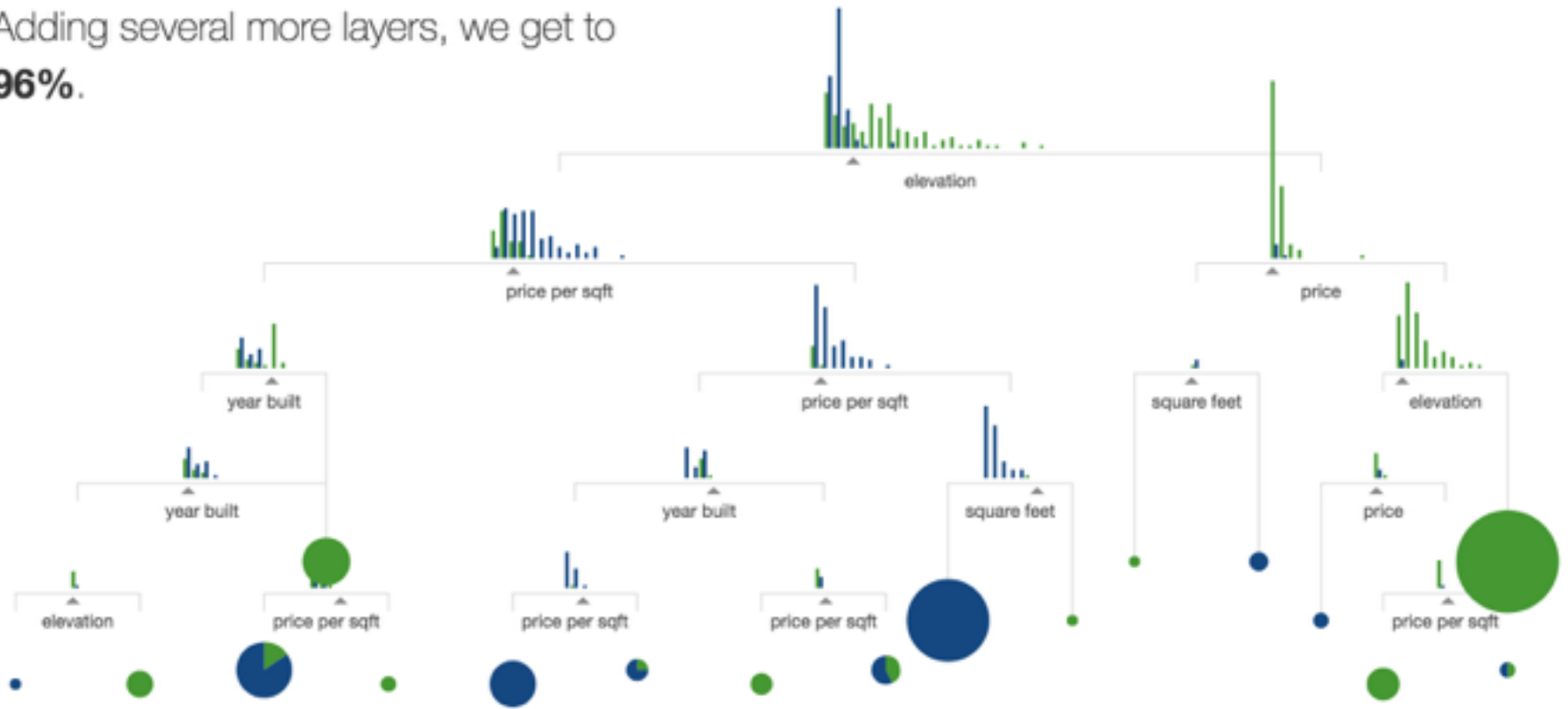


## Growing a tree

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

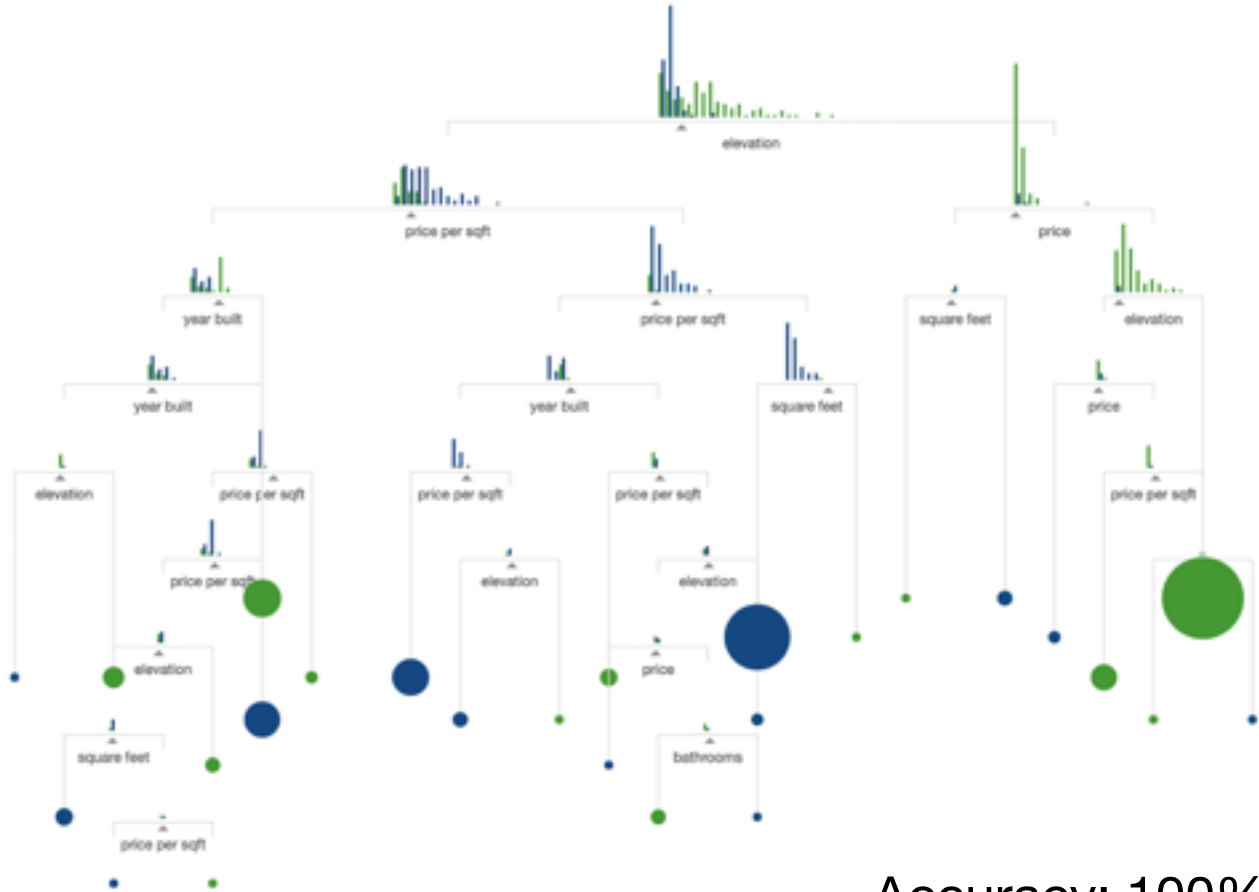


Adding several more layers, we get to **96%**.



Accuracy: 96%

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

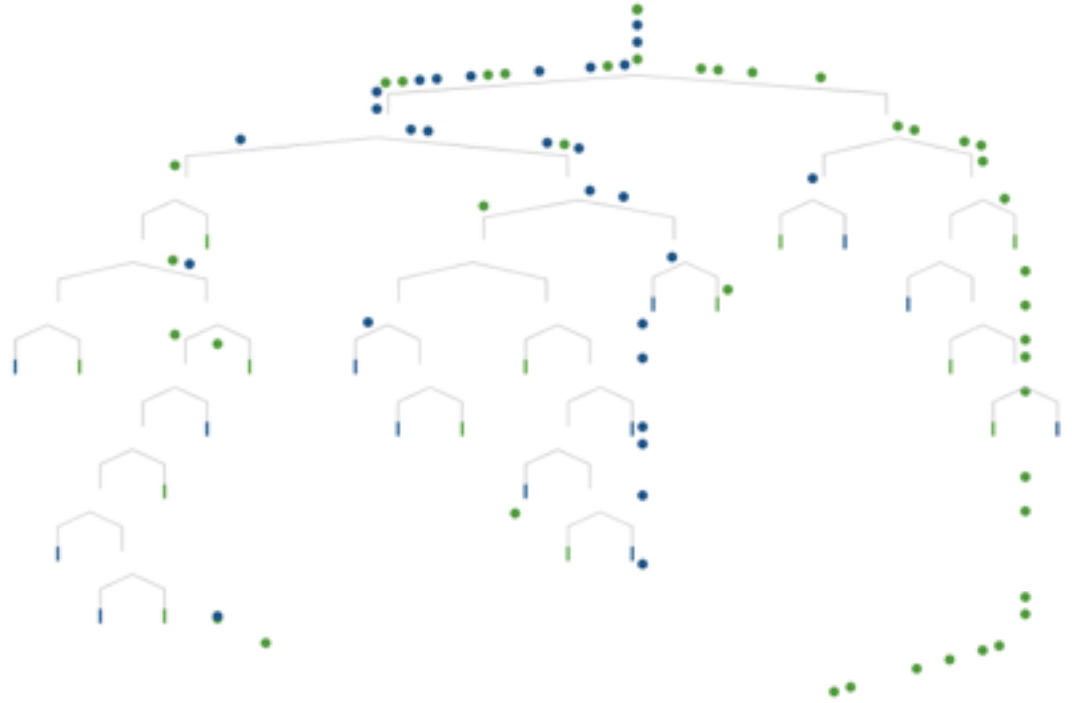


Accuracy: 100%

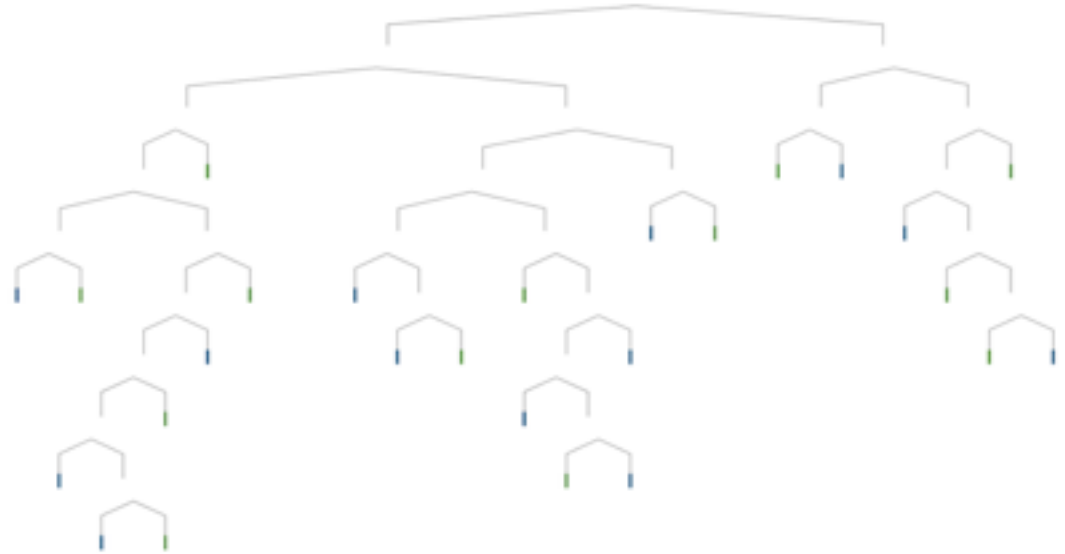
## Making predictions

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



111/111

Training Accuracy

100%

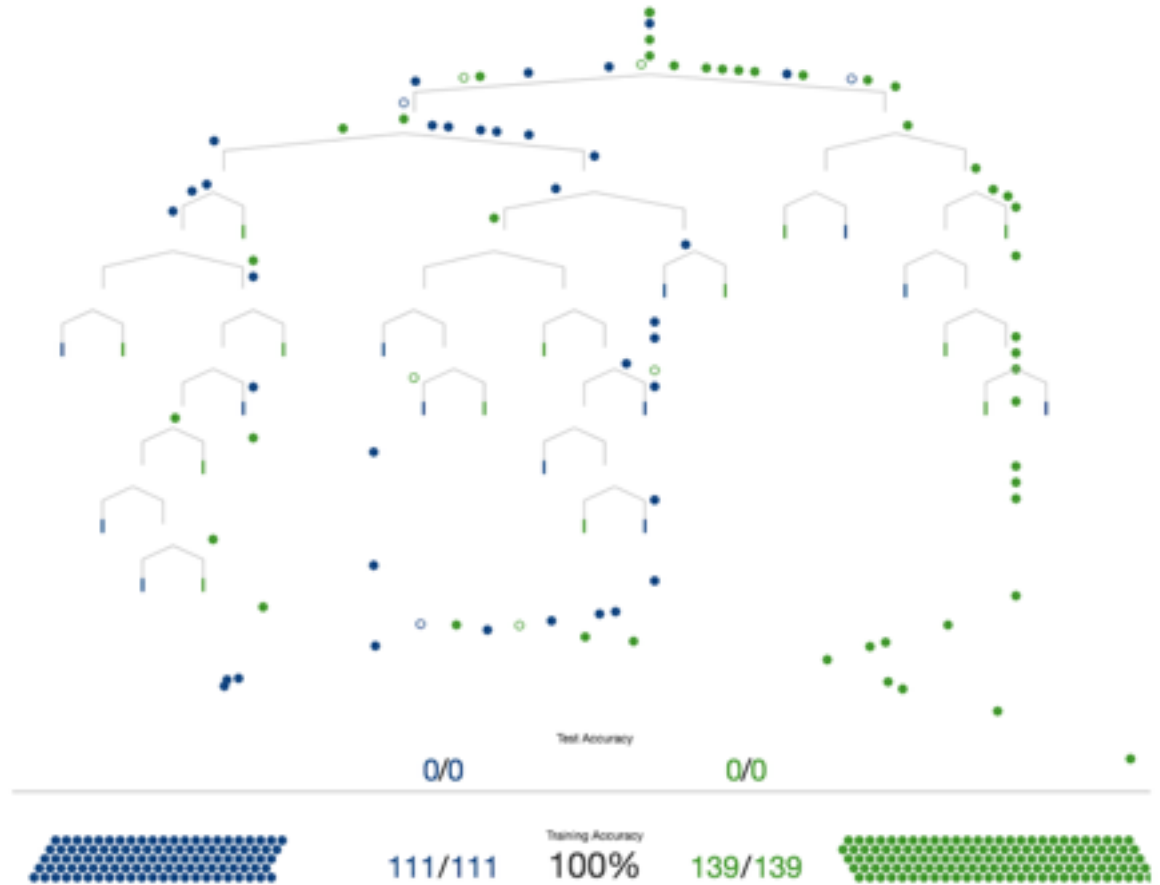
139/139



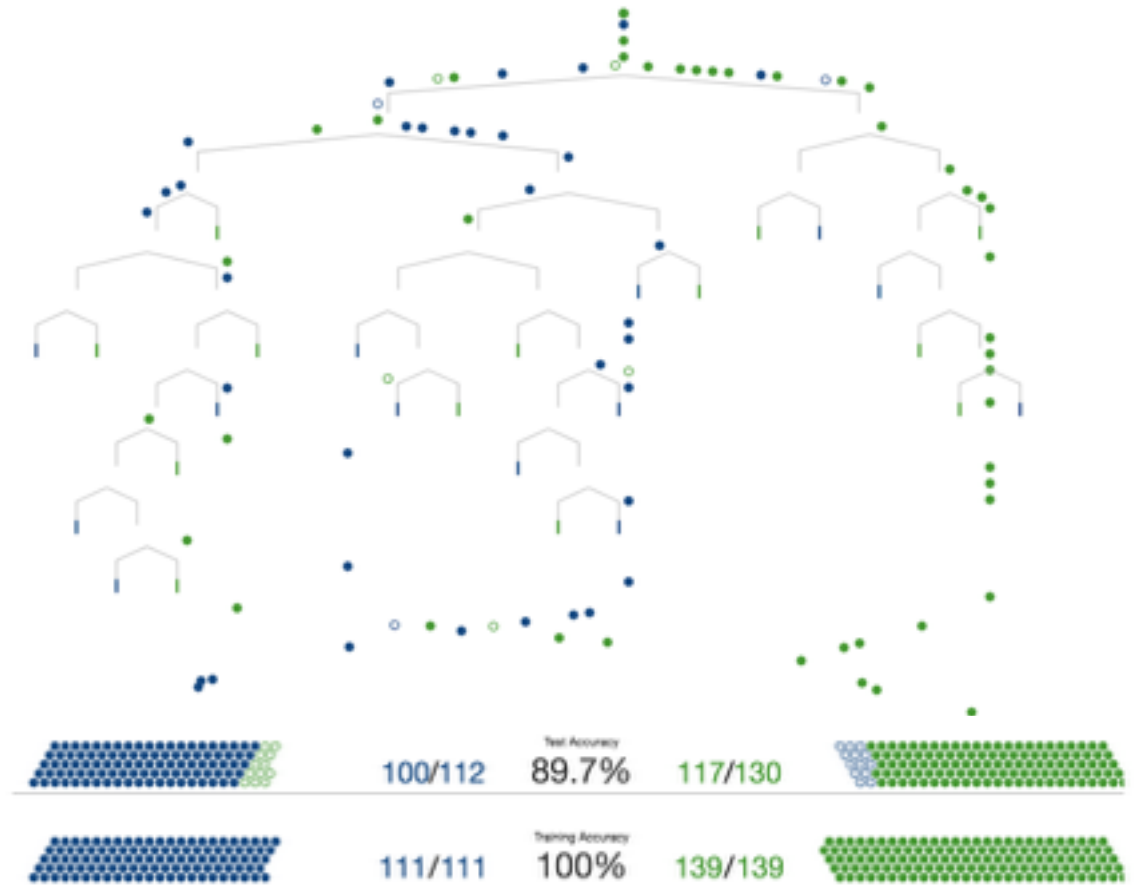
## Reality check

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

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



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



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.





# Recap

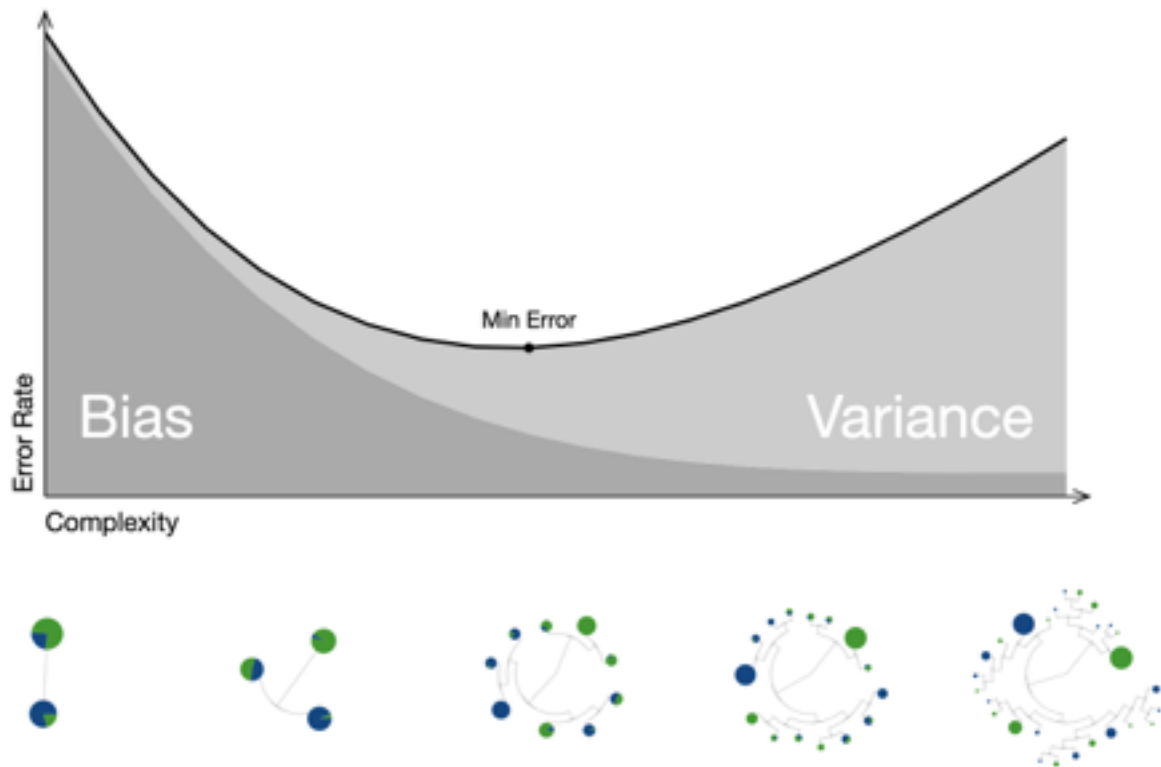
---

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?

---

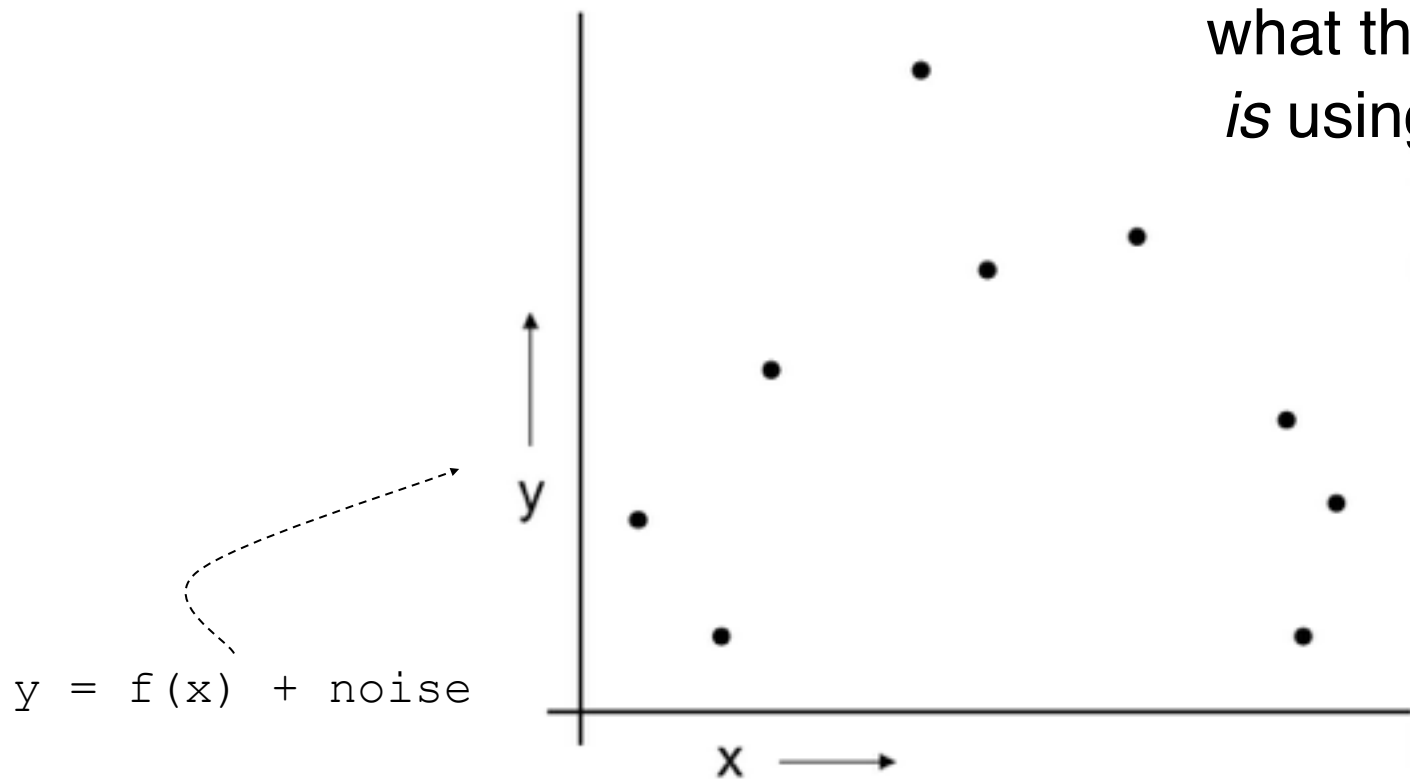
# Bias-variance tradeoff



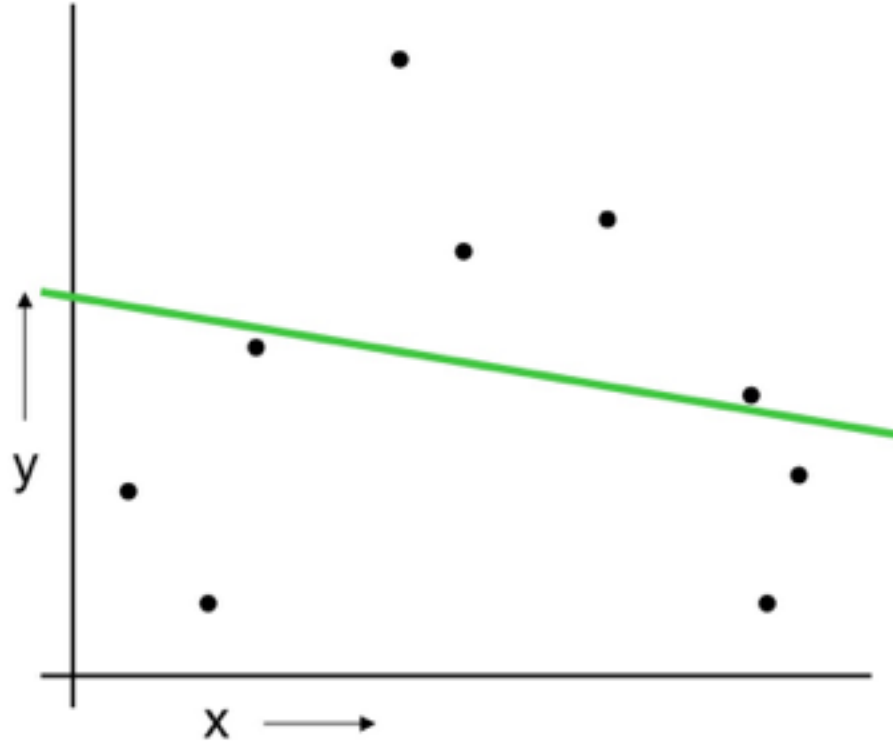
# Bias-variance tradeoff

- **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

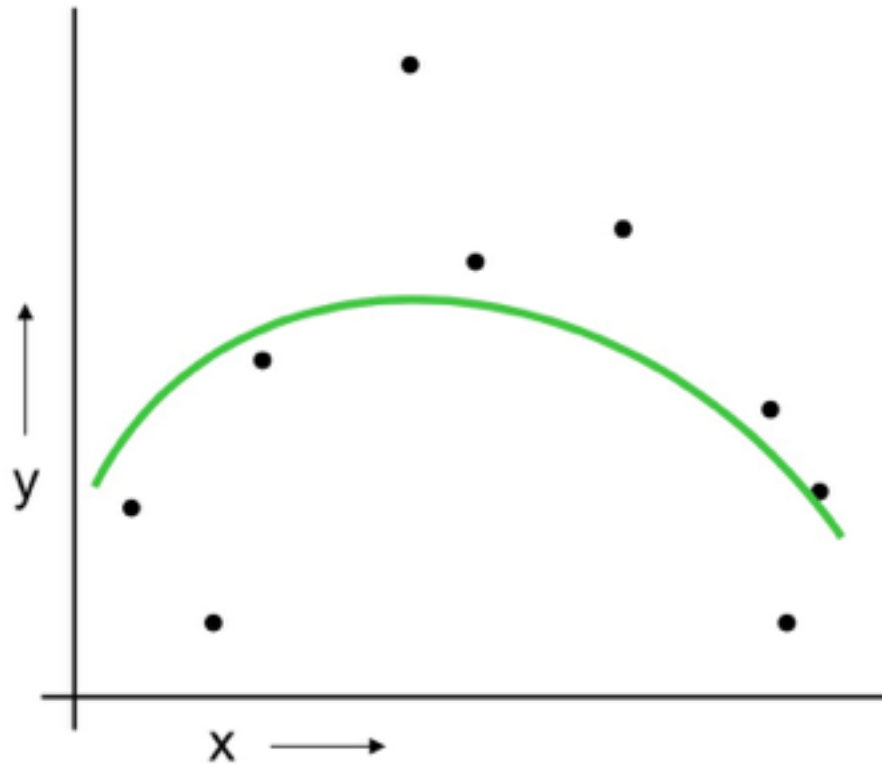
Can we determine  
what that function ( $f$ )  
is using these data?



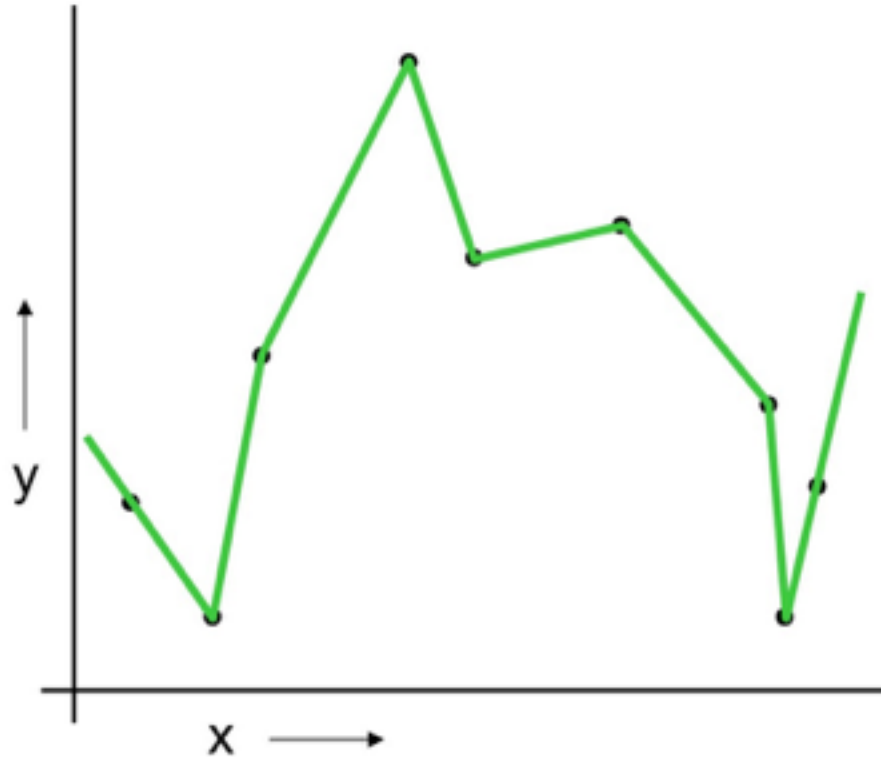
# Linear regression



# Quadratic regression

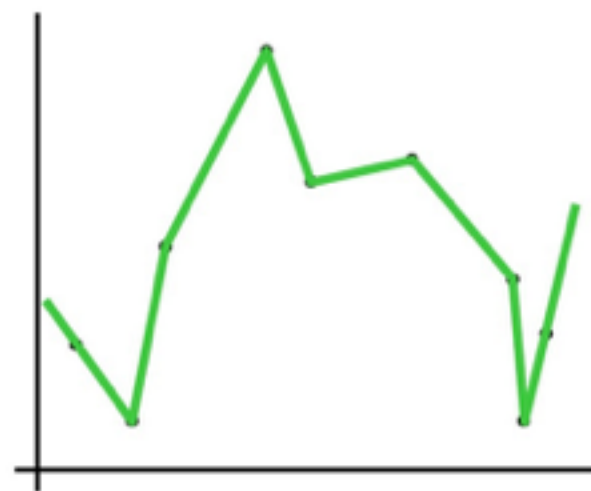
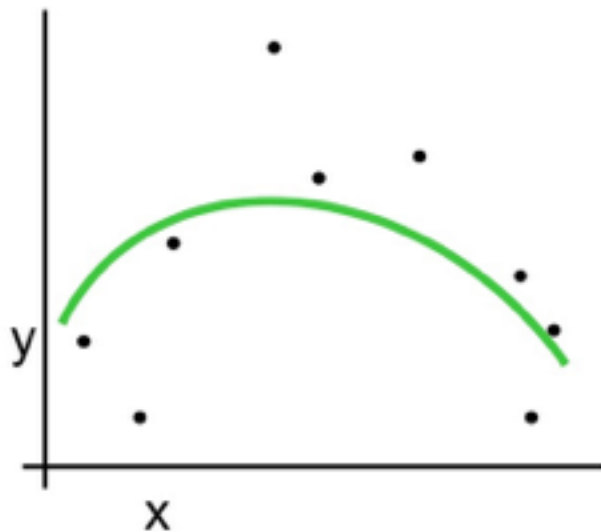
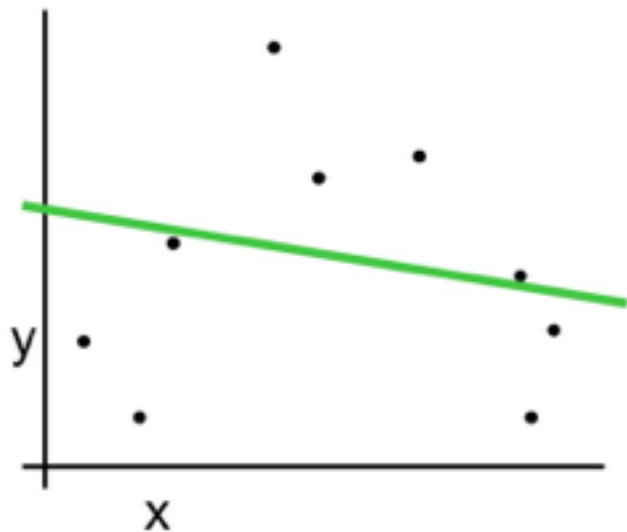


# Piecewise linear nonparametric regression

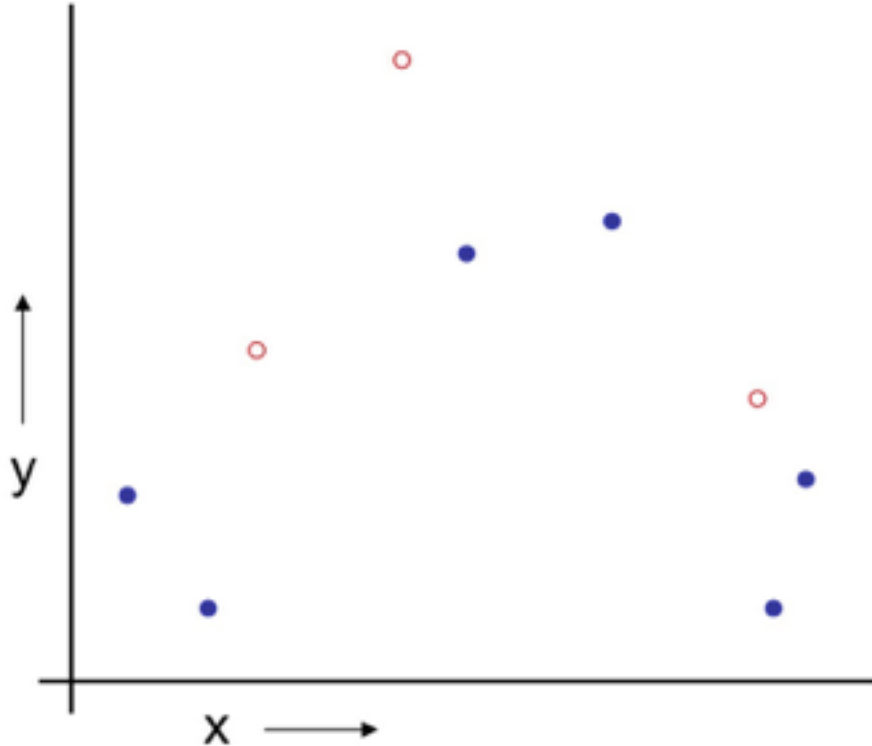




Which to choose?

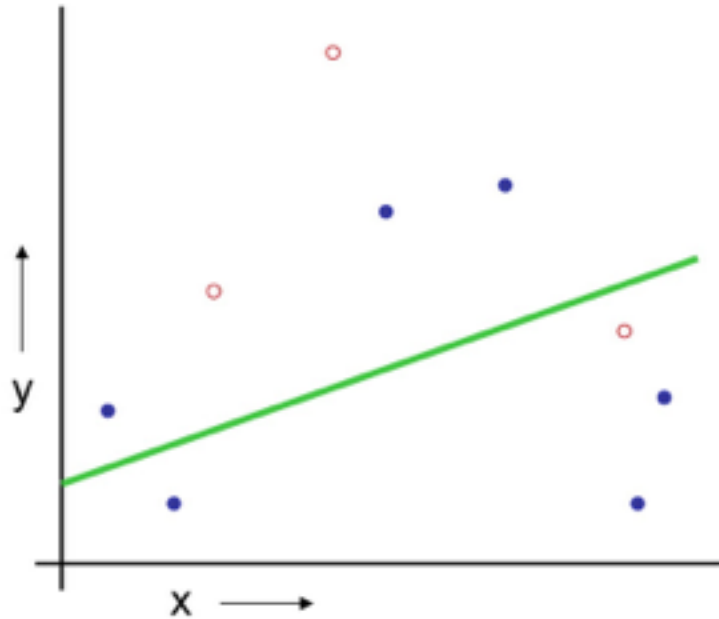


# The data partition method



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



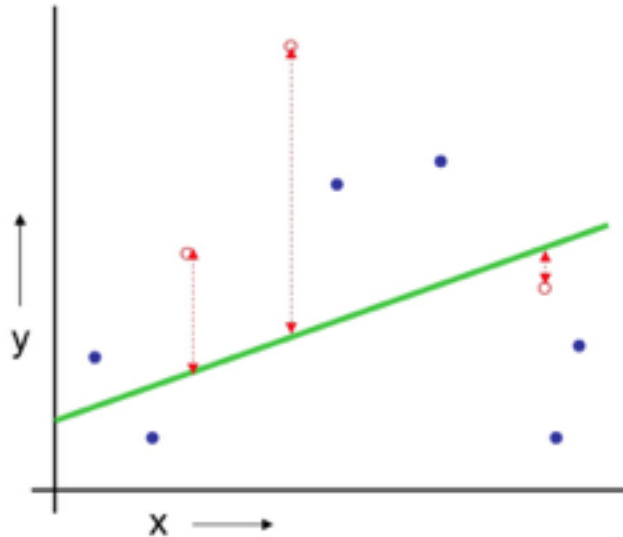
(Linear regression example)

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

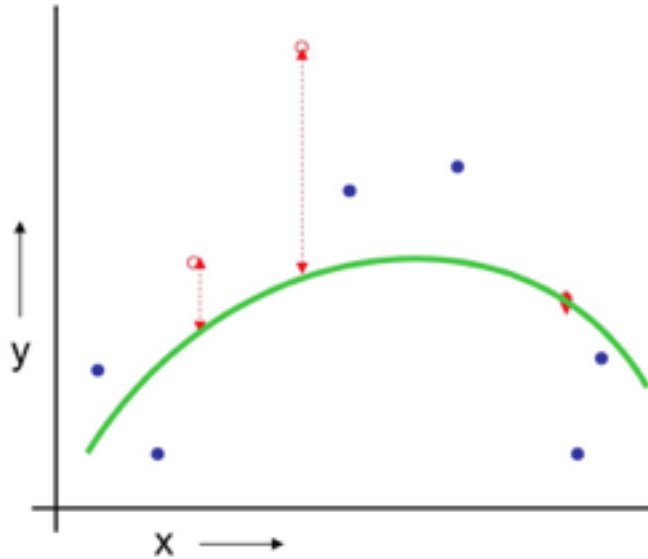
1. Randomly choose  
30% of the data to be in a  
**test set**

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

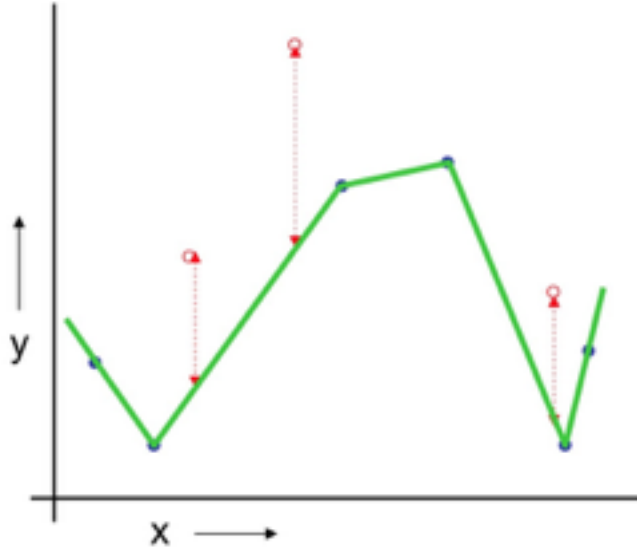
1. Randomly choose  
30% of the data to be in a  
**test set**

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

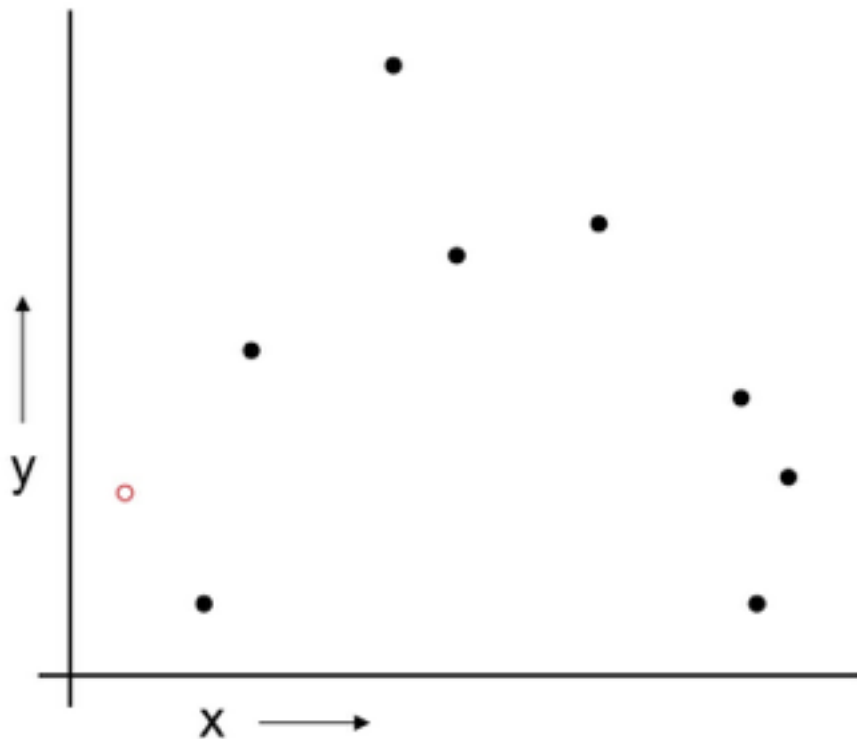
## Cons:

- 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)

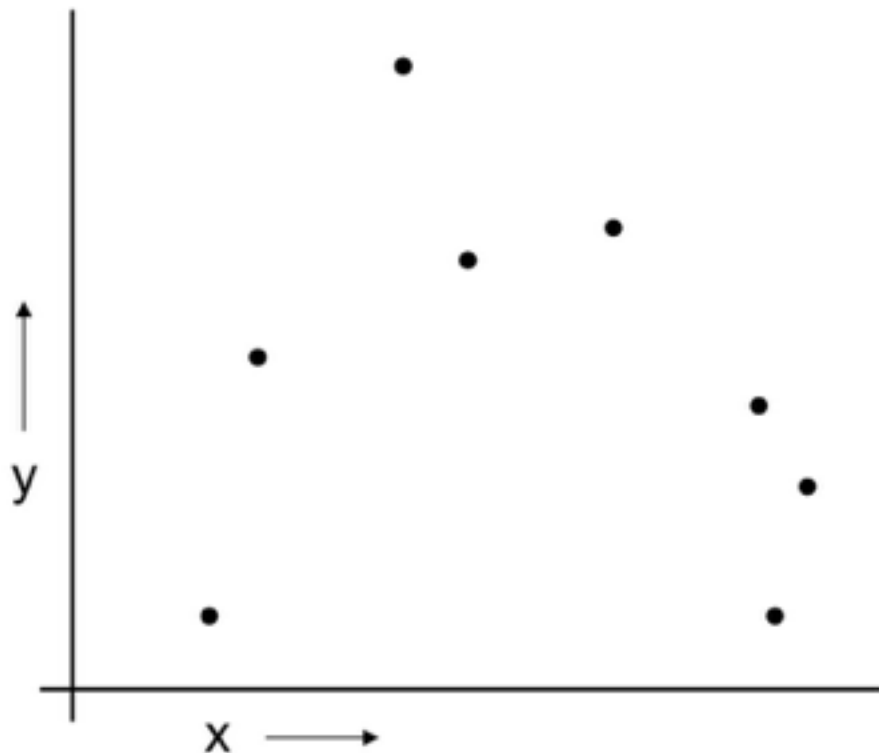
For  $k=1$  to  $R$

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





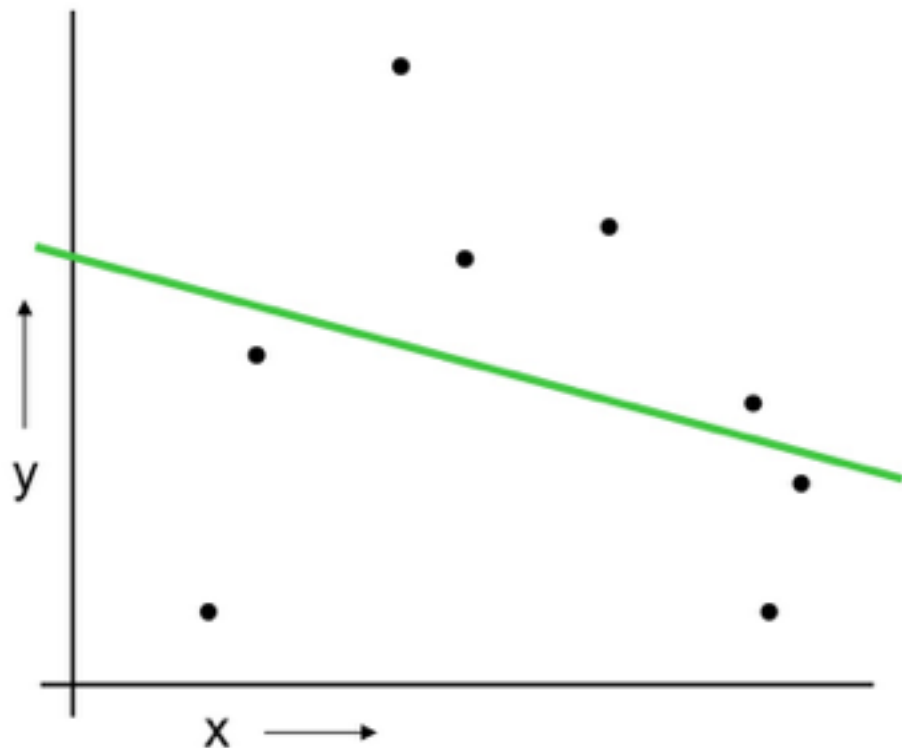
# Leave one out cross validation (LOOCV)



For  $k=1$  to  $R$

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

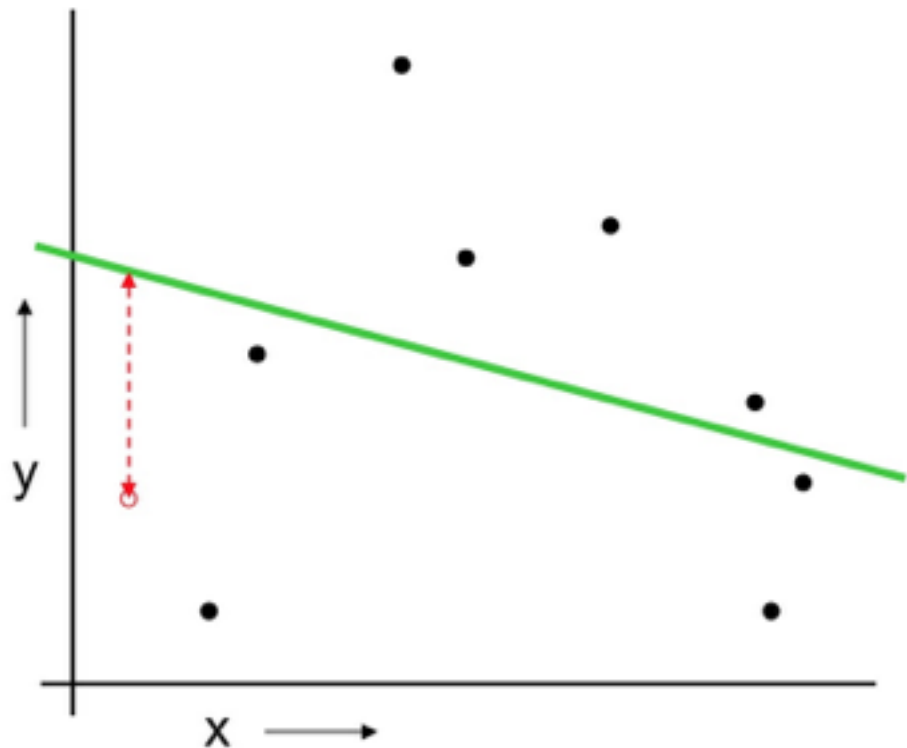
# Leave one out cross validation (LOOCV)



For  $k=1$  to  $R$

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

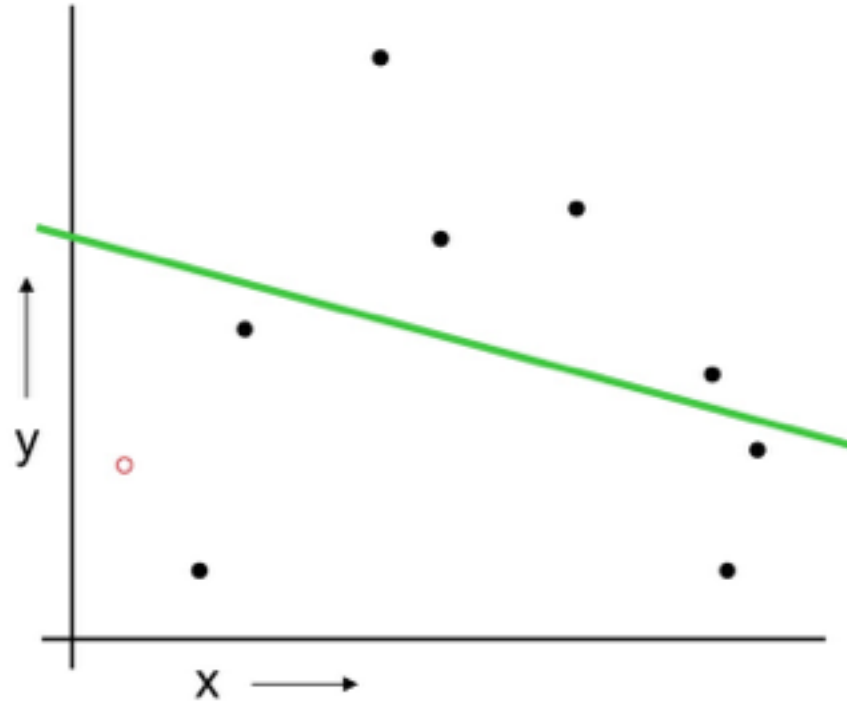
# Leave one out cross validation (LOOCV)



For  $k=1$  to  $R$

1. Let  $(x_k, y_k)$  be the  $k^{\text{th}}$  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)

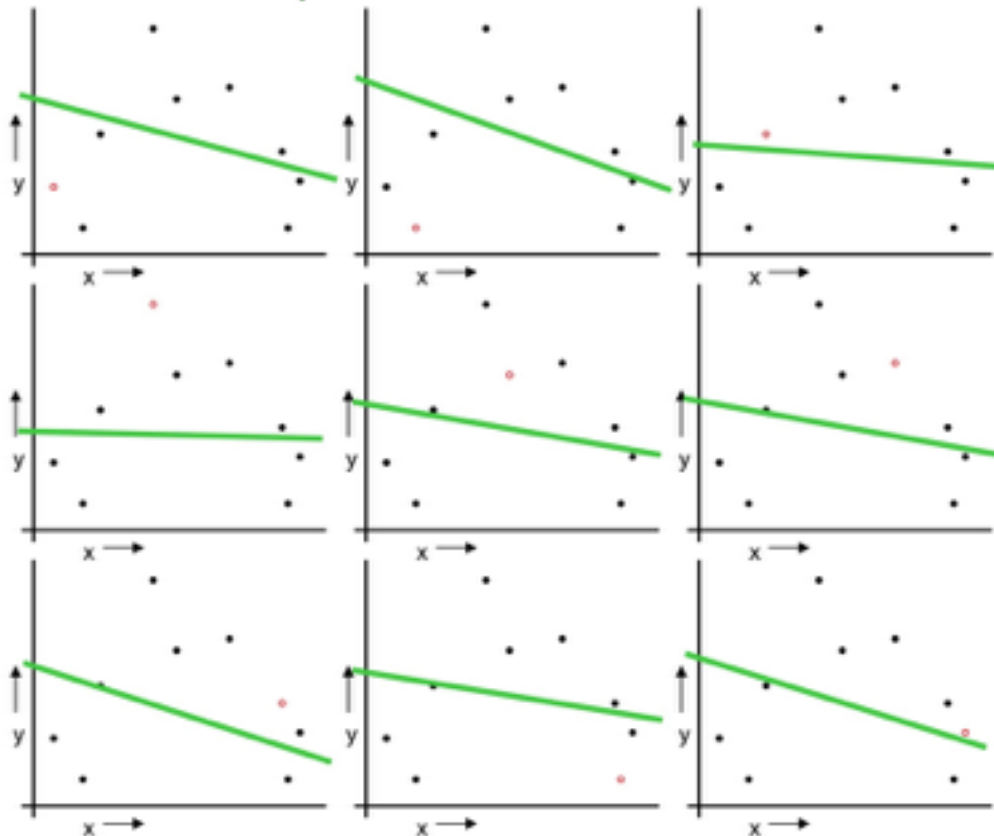


For  $k=1$  to  $R$

1. Let  $(x_k, y_k)$  be the  $k^{\text{th}}$  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)



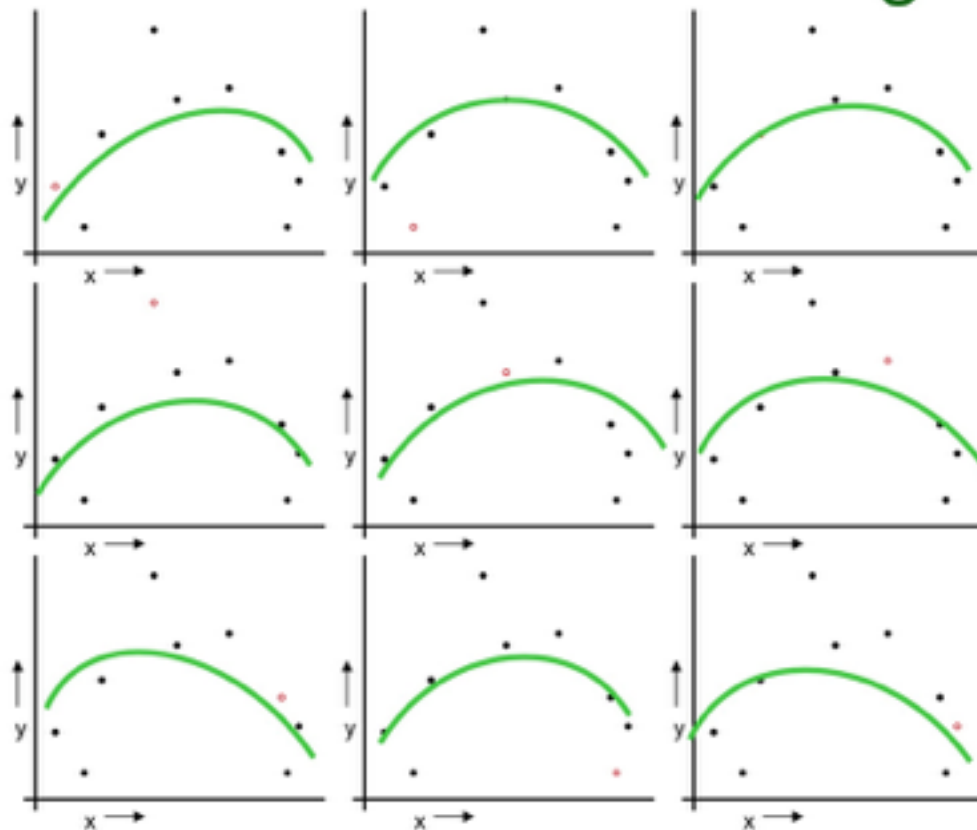
For  $k=1$  to  $R$

1. Let  $(x_k, y_k)$  be the  $k^{\text{th}}$  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)



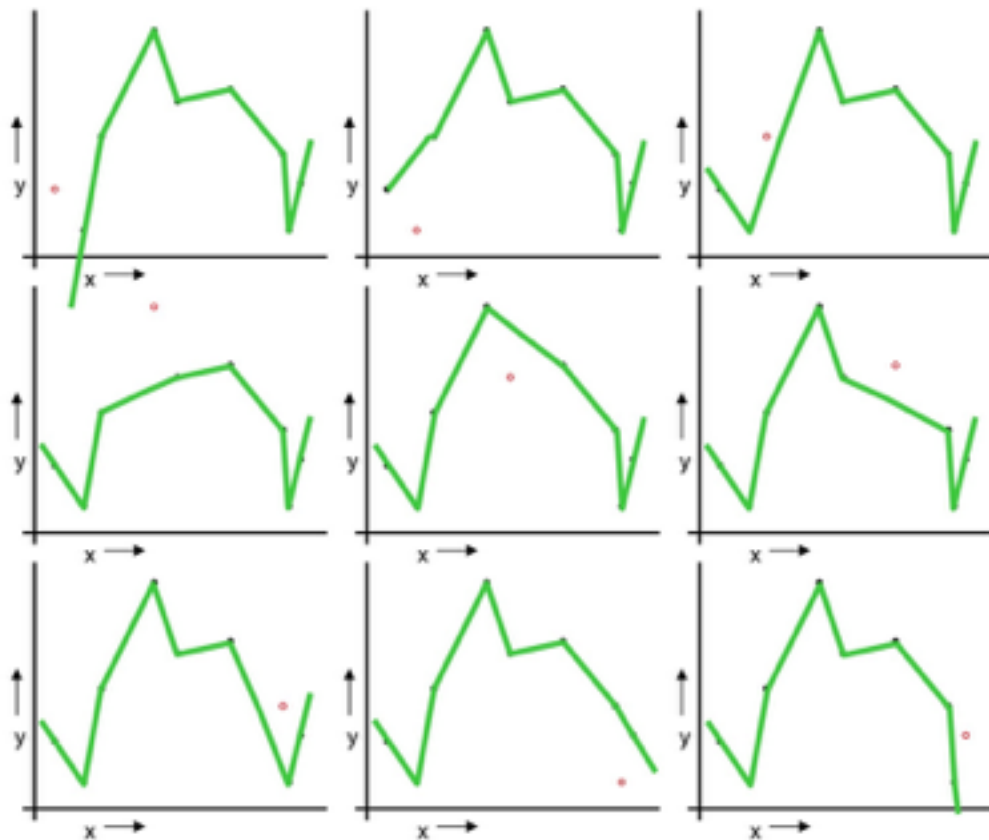
For  $k=1$  to  $R$

1. Let  $(x_k, y_k)$  be the  $k^{\text{th}}$  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  $k^{\text{th}}$  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.

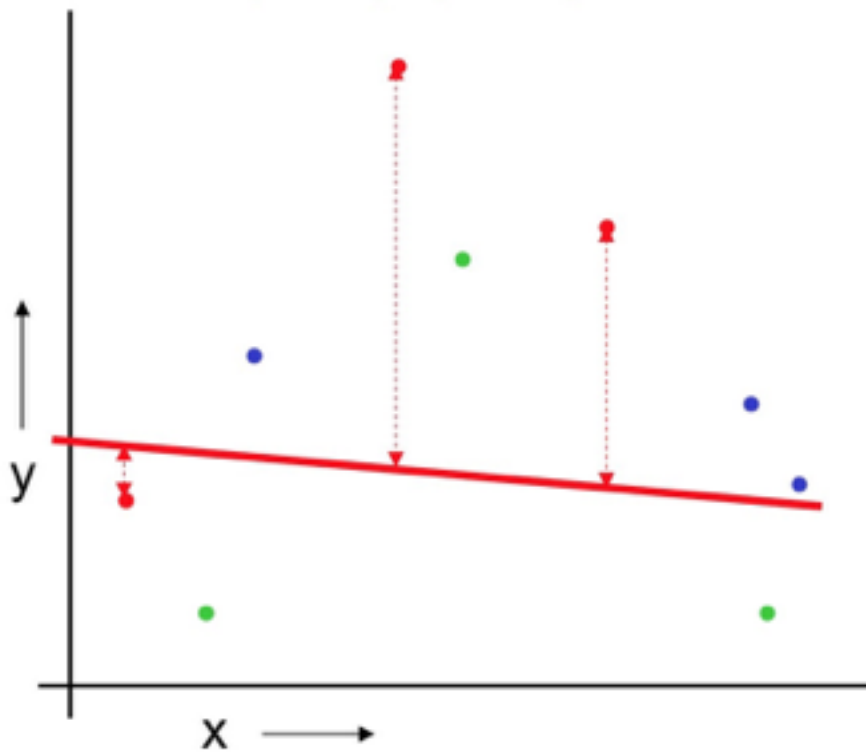
$$MSE_{LOOCV} = 3.33$$

# Method Comparison

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

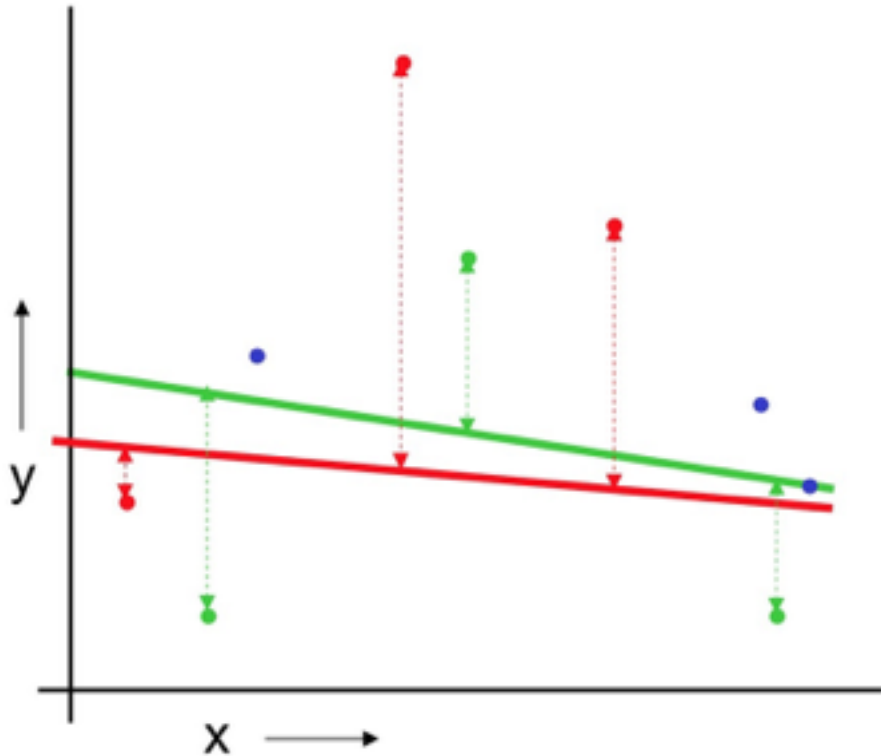


# $k$ -fold cross validation



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.

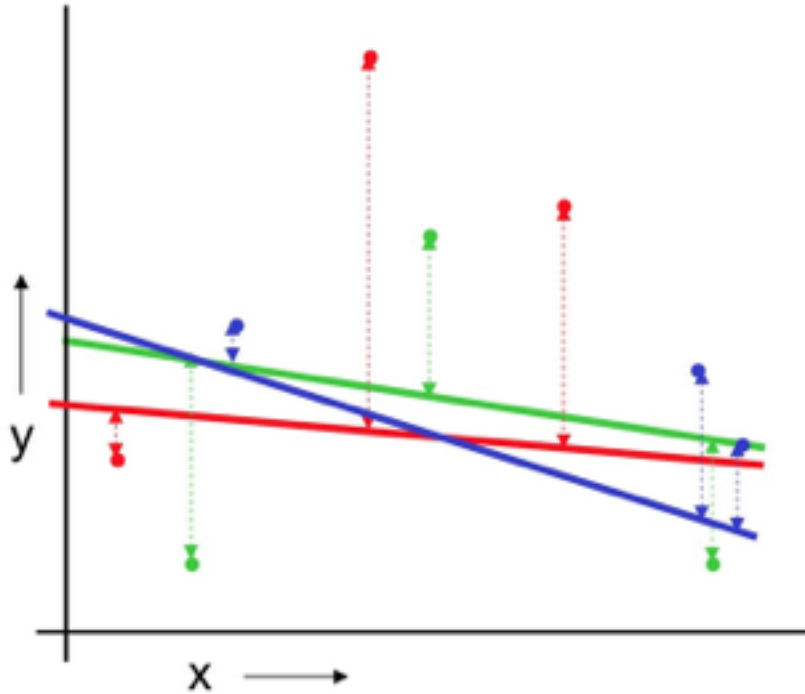
# $k$ -fold cross validation



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

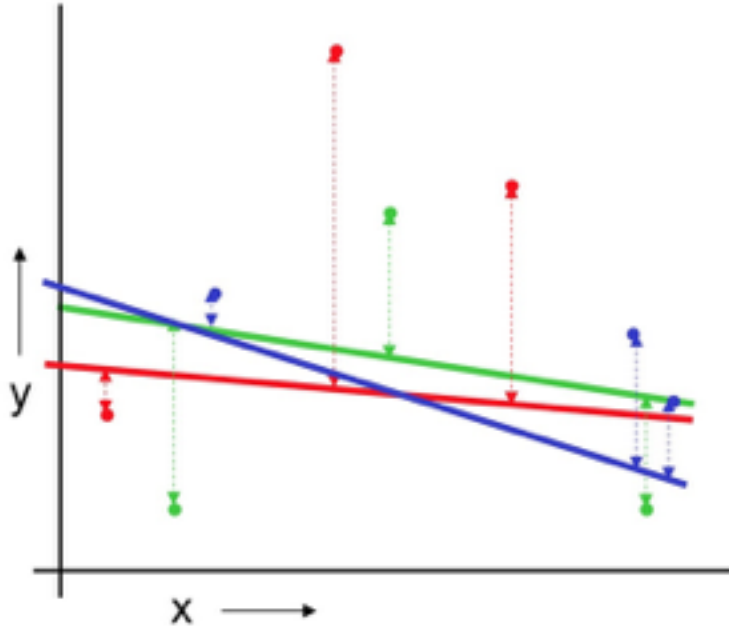


For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

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  
 $MSE_{3FOLD}=2.05$

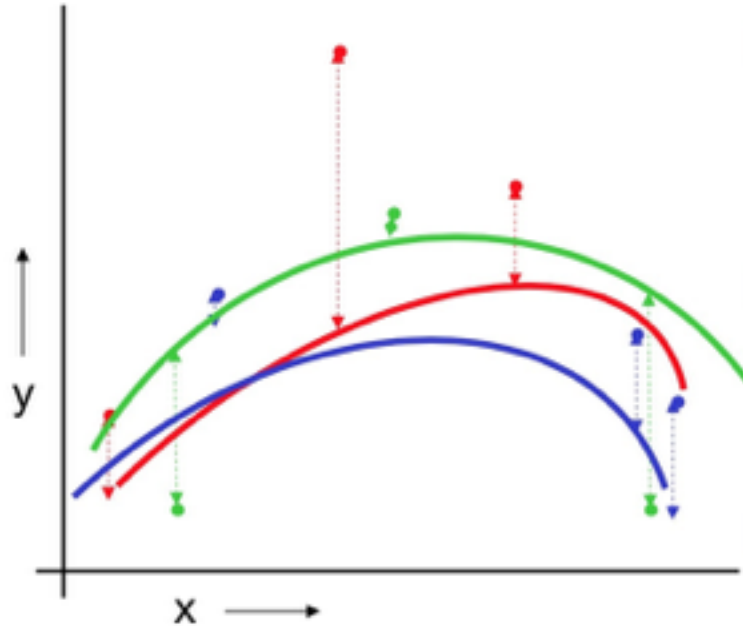
For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

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  
 $MSE_{3FOLD}=1.11$

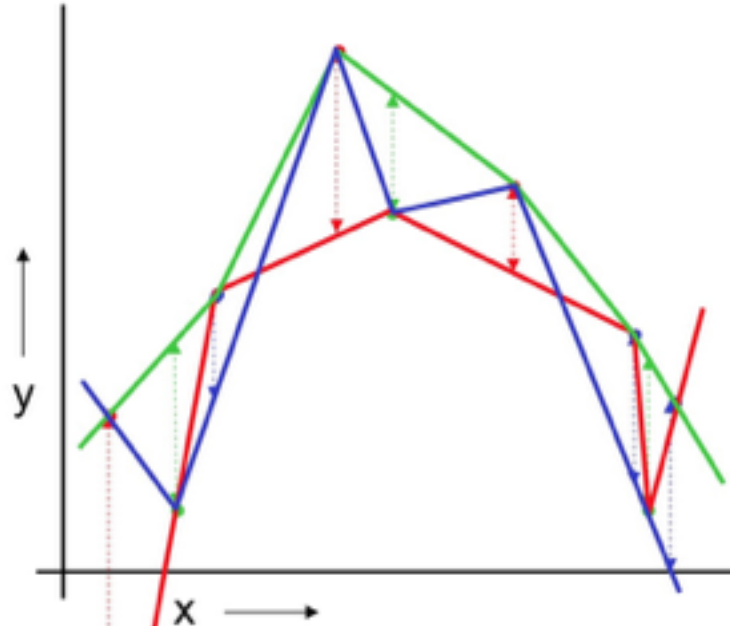
For the red partition: Train on all the points not in the red partition. Find the test-set sum of errors on the red points.

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



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.

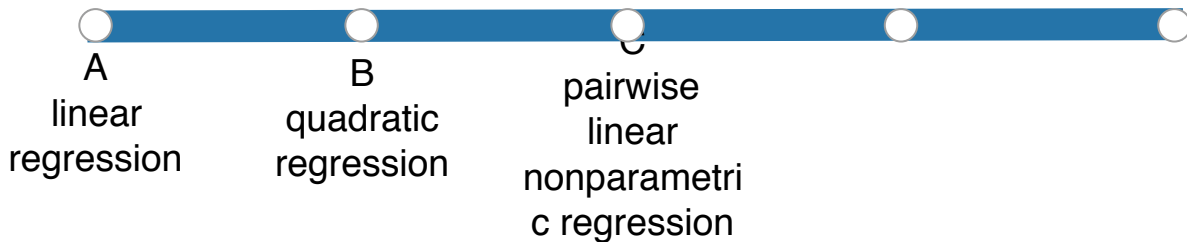
For the green partition: Train on all the points not in the green partition. Find the test-set sum of errors on the green points.

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



Given the example we just worked, how would *you* model these data?



Linear Regression  
 $MSE_{3FOLD}=2.05$

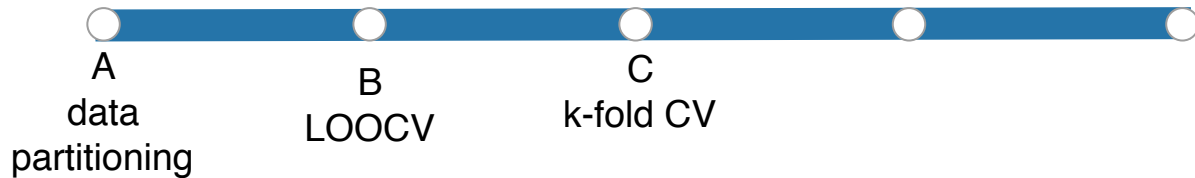
Quadratic Regression  
 $MSE_{3FOLD}=1.11$

Joint-the-dots  
 $MSE_{3FOLD}=2.93$

# Validator



Which approach would *you* use to limit overfitting?





When models are trained on  
historical data, predictions will  
perpetuate historical biases



Predictive Analysis Ethics

# Amazon scraps secret AI recruiting tool that showed bias against women

Jeffrey Dastin

8 MIN READ



SAN FRANCISCO (Reuters) - Amazon.com Inc's (AMZN.O) machine-learning specialists uncovered a big problem: their new recruiting engine did not like women.

---

{\* ARTIFICIAL INTELLIGENCE \*}

# MIT apologizes, permanently pulls offline huge dataset that taught AI systems to use racist, misogynistic slurs

Top uni takes action after *EI Reg* highlights concerns by academics

Katyanna Quach Wed 1 Jul 2020 // 10:55 UTC

SHARE







Chukwuemeka Afigbo

@nke\_ise

Follow

If you have ever had a problem grasping the importance of diversity in tech and its impact on society, watch this video



5:48 AM · 16 Aug 2017

155,234 Retweets 215,762 Likes



[https://twitter.com/nke\\_ise/status/897756900753891328](https://twitter.com/nke_ise/status/897756900753891328)



**Janice Wyatt-Ross, Ed.D**

@JaniceWyattRoss



Daughter 1 was taking an exam today being proctored by some type of software that apparently was not tested on dark skin. She had to open her window, turn on the lights, and then shine a flashlight over her head to be detectable. 😡😡😡

6:01 PM • Feb 22, 2021 • Twitter for iPhone

**7,030** Retweets   **939** Quote Tweets   **34.6K** Likes

# 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.

# Discussed so far...

- data partitioning
- feature selection
- supervised & unsupervised machine learning
  - Continuous variables: regression (supervised) and dimensionality reduction (unsupervised)
  - Categorical variables: classification (supervised; decision trees) or clustering (unsupervised)
- model assessment
  - Continuous: RMSE (& Accuracy)
  - Categorical: Accuracy, Sensitivity, Specificity, AUC
- biased data can & will lead to biased predictions



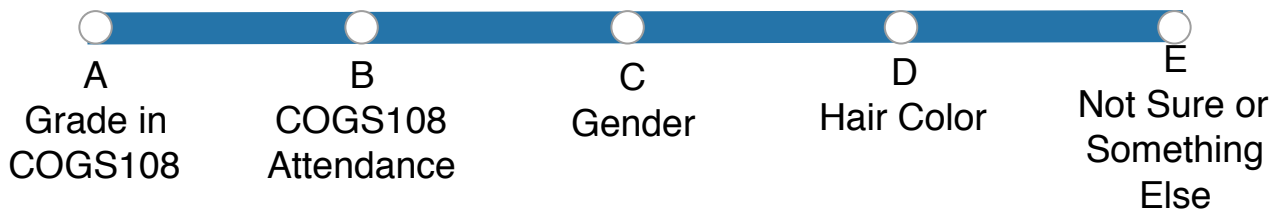
## Data Science Question

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

# Prediction Approach



Which would be the most predictive of your future success?



**N = 254**

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

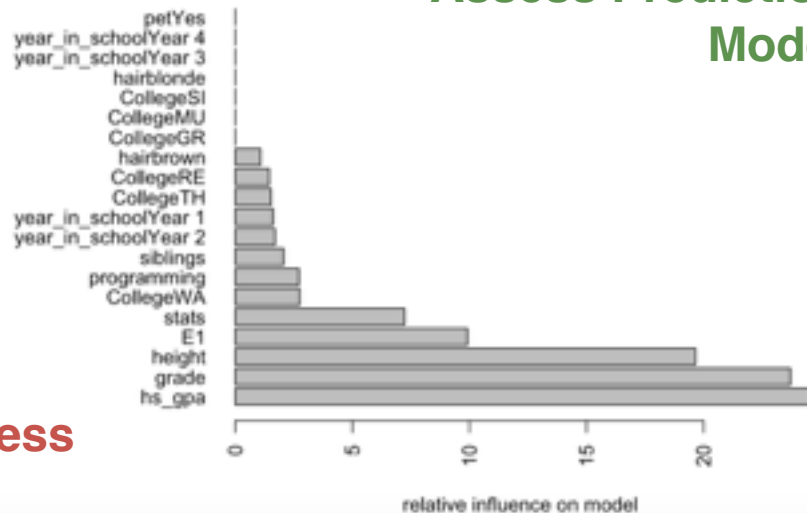
**train the model**

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

**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)

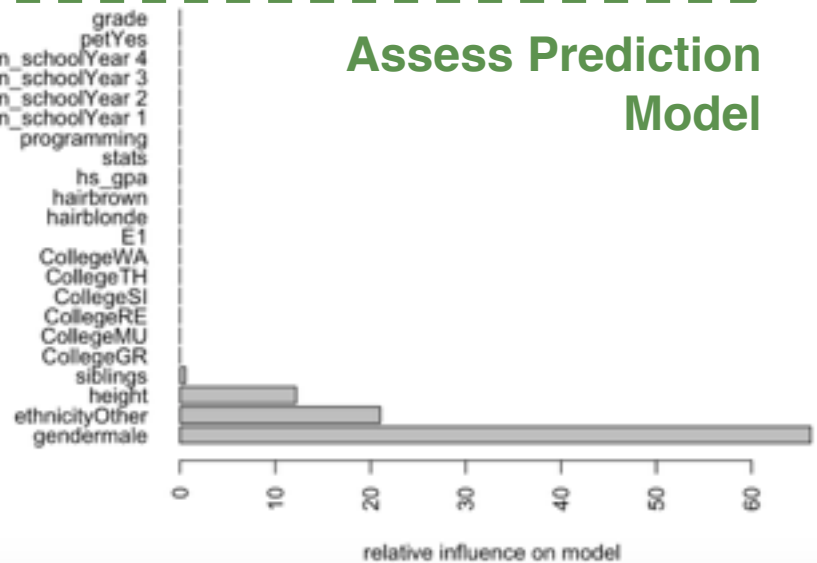
**train the model**

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

**predicted success  
in test set**

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

**Assess Prediction  
Model**



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 data science majors?



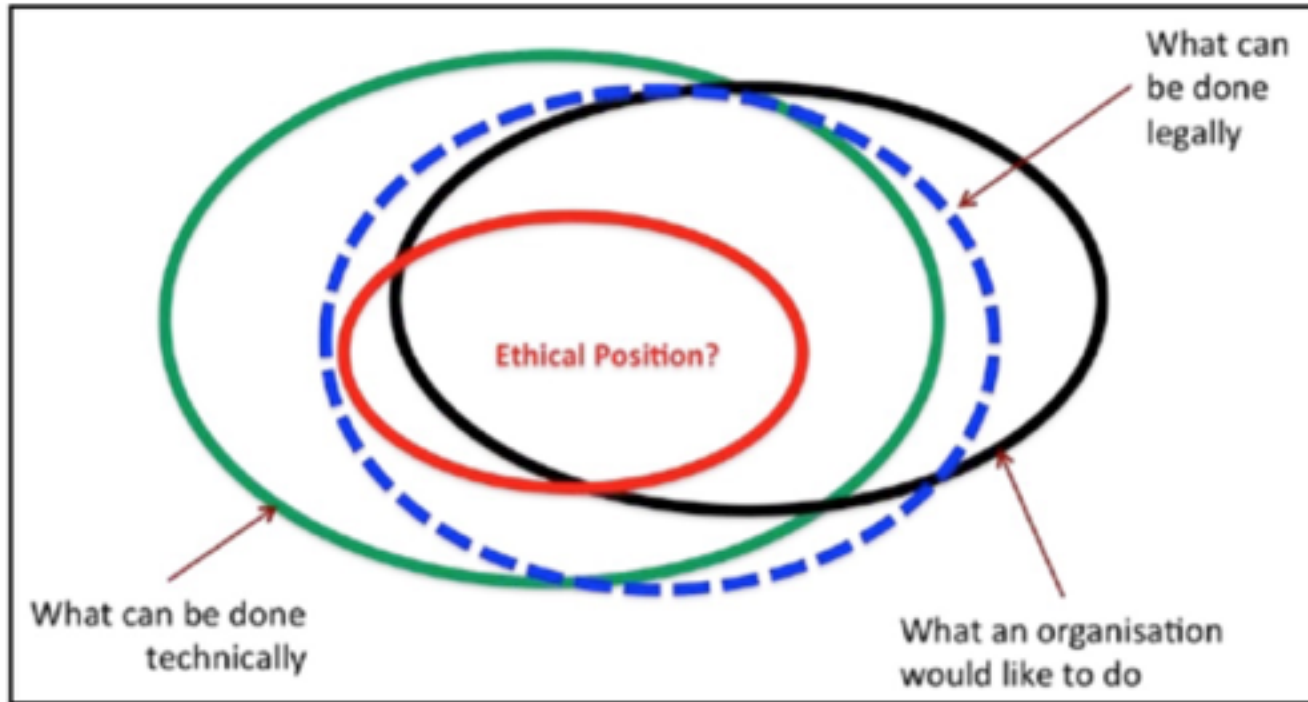
# What to do about bias...

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





# 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)

---

# 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

## Meredith Durbin

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

## Jevan Hutson

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



**Logan-Nolan Industries**

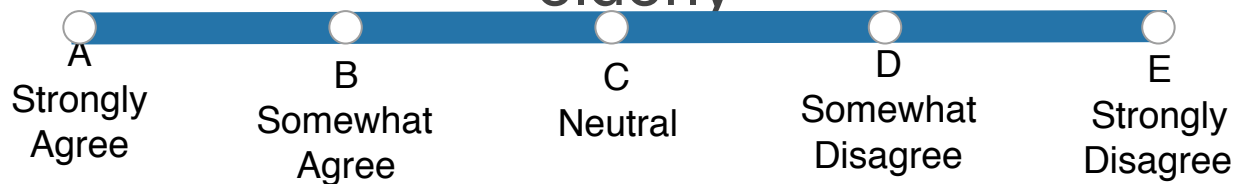
*Helping Humanity Make Ends Meat*

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

# Prediction Thoughts



We should start using this algorithm to mulch up the  
elderly

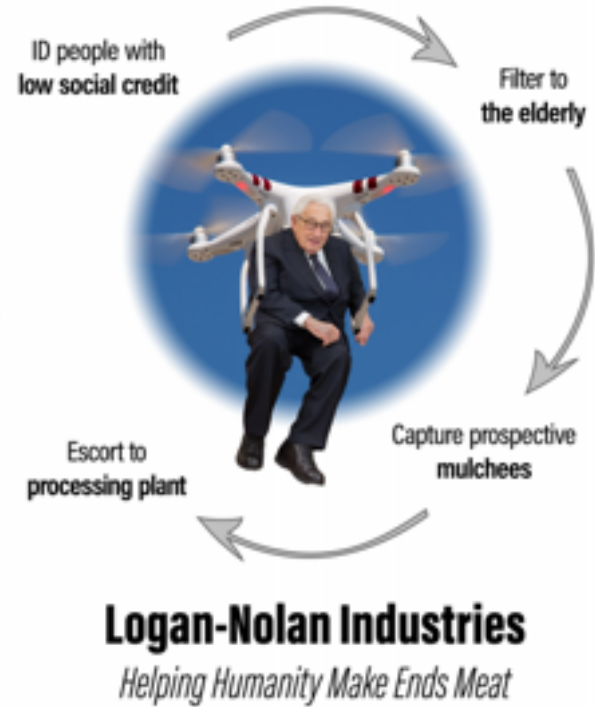


# A Mulching Proposal

**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 an excuse for thoughtlessness.