Machine Learning L+Pr

Béla J. Szekeres, PhD

Lecture 2 / II

ELTE Faculty of Informatics, Szombathely, Hungary

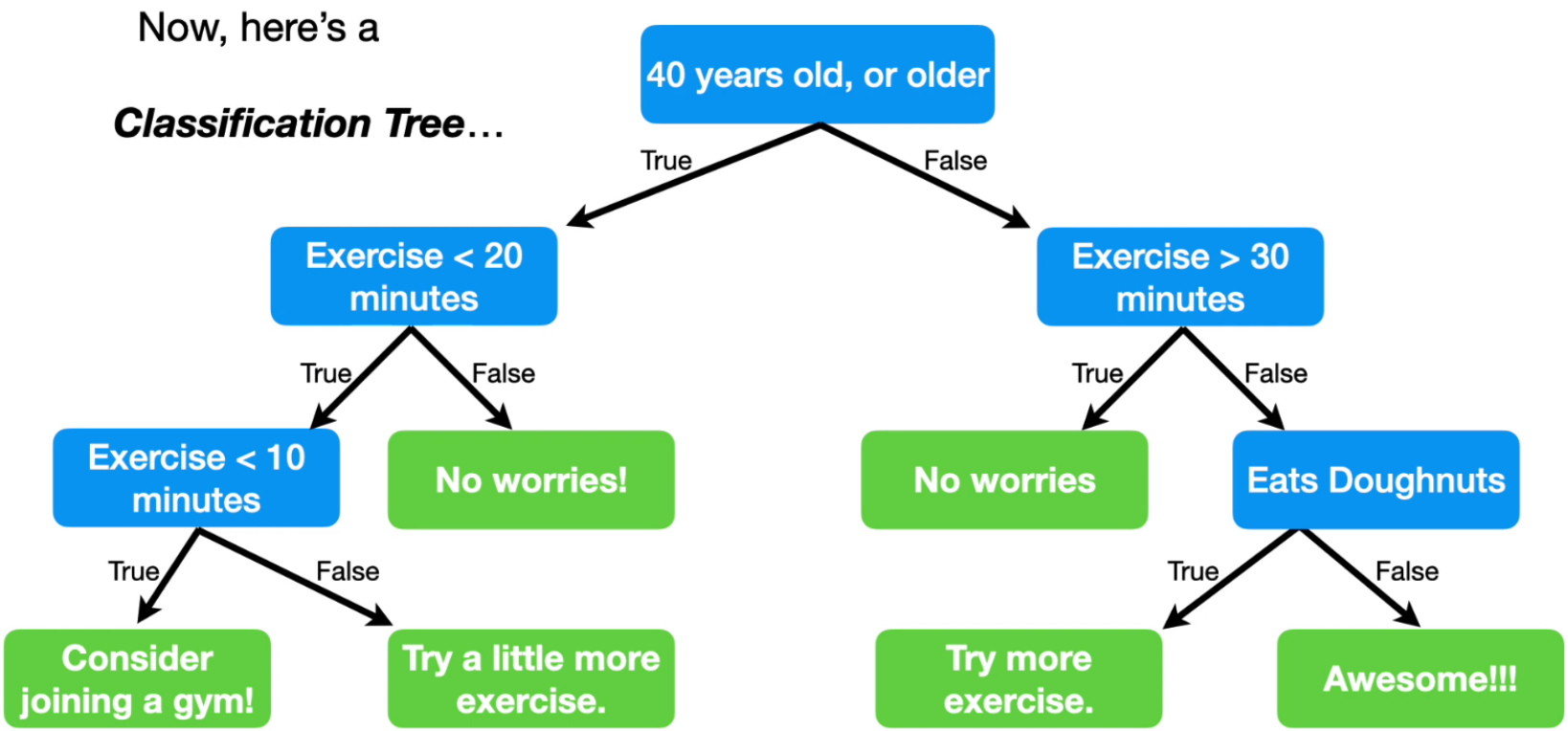
Topics of this day

- Decision Trees: Classification trees, Regression trees
- Random Forests
- SVM
- Practice: Scikit-learn

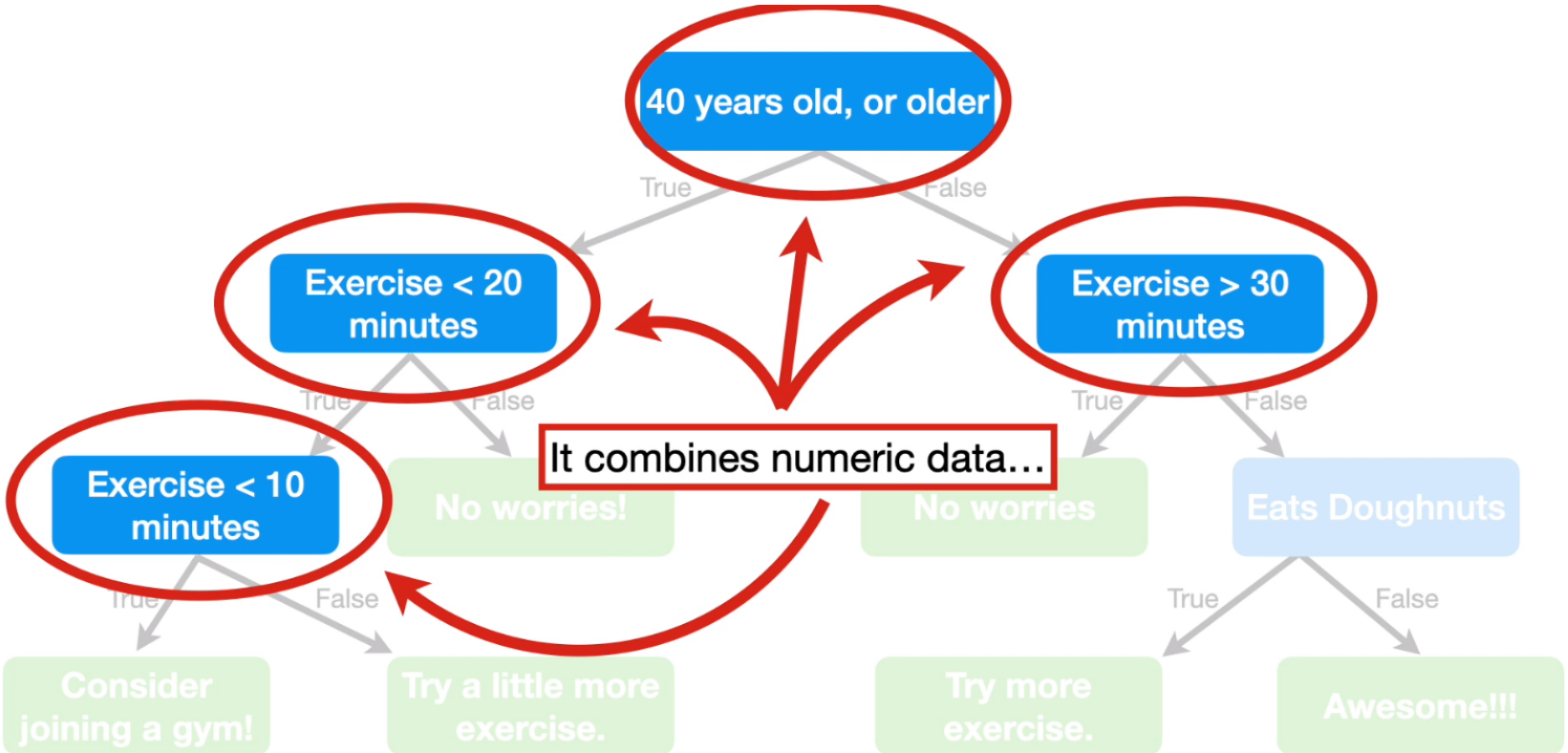
Decision Trees

- it is a type of supervised learning approaches: mostly used in classification problems but it can be used for regression as weel
- works fine for both categorical variables and continouous input as well
- in every step it splits the data into two sets based on significant splitter in input variables - Gini-index
- assume that all data is numerical (we can convert every categorical value to numerical)
- when a decision tree classifies things into categories, it's called a **Classification Tree**
- and when a decision tree predicts numerical values, it's called a **Regression tree**

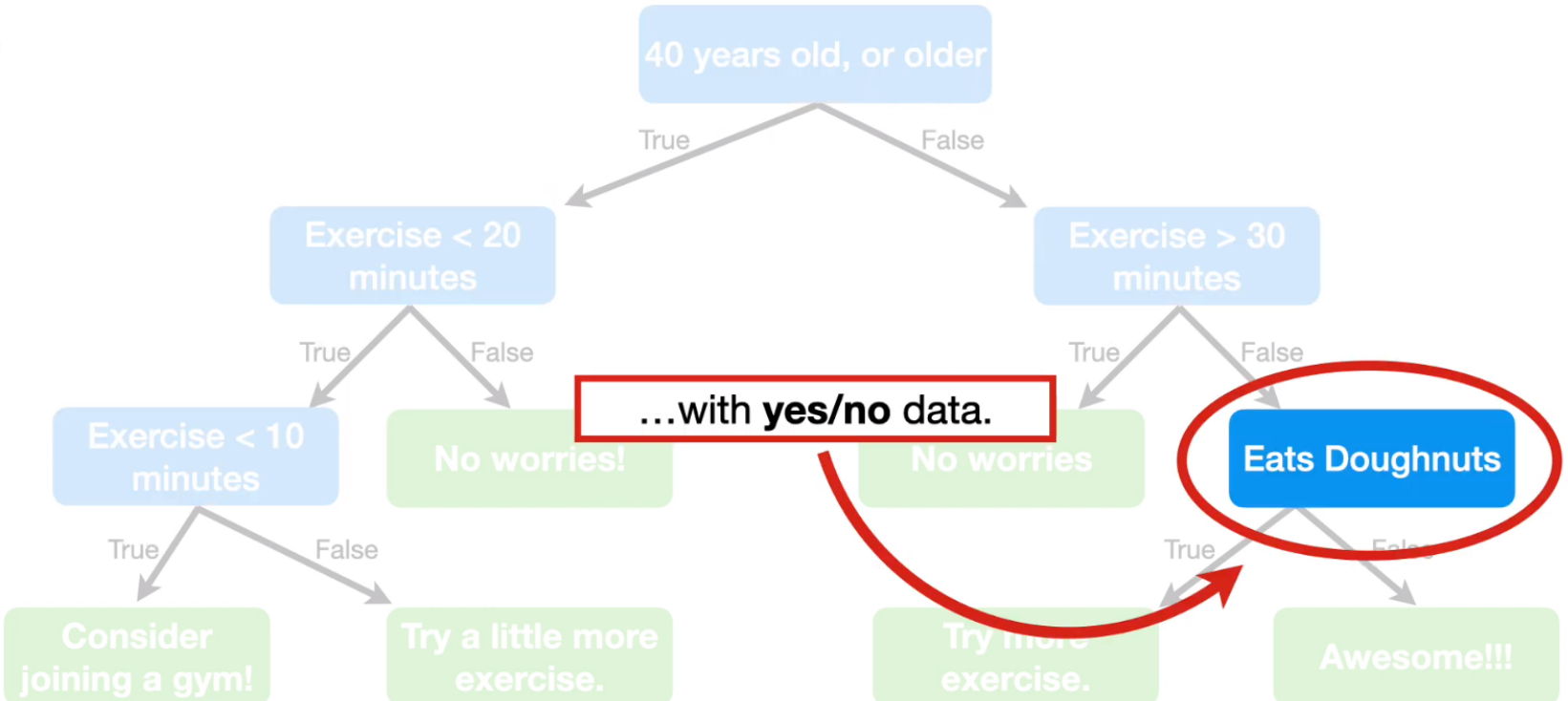
Example


Now, here's a
Classification Tree...
40 years old, or older
True
False
Exercise < 20 minutes
Exercise > 30 minutes
True
False
True
False
Exercise < 10 minutes
No worries!
No worries
Eats Doughnuts
True
False
True
False
Consider joining a gym!
Try a little more exercise.
Try more exercise.
Awesome!!!

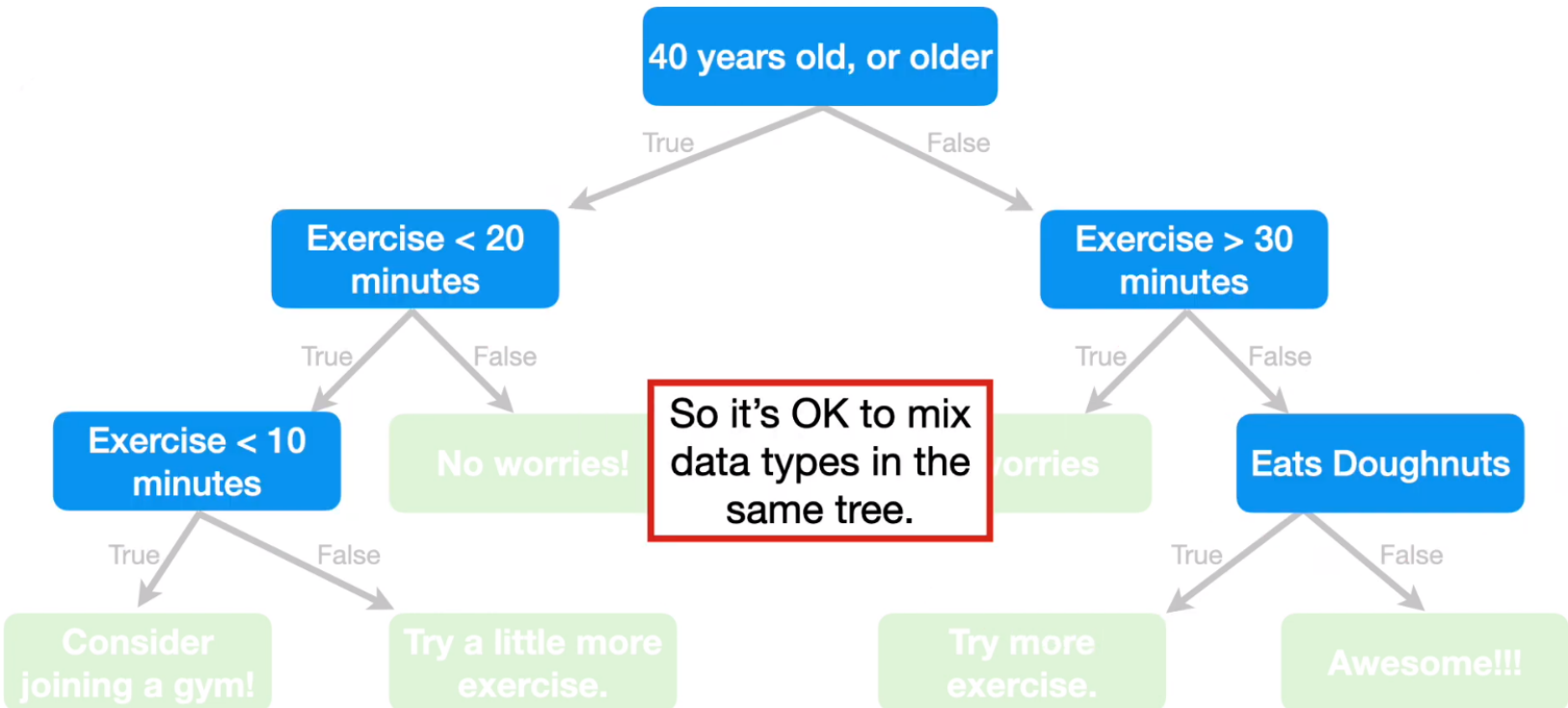
Example


40 years old, or older
True
False
Exercise < 20 minutes
Exercise > 30 minutes
True
False
True
False
Exercise < 10 minutes
No worries!
It combines numeric data...
No worries
Eats Doughnuts
True
False
True
False
Consider joining a gym!
Try a little more exercise.
Try more exercise.
Awesome!!!

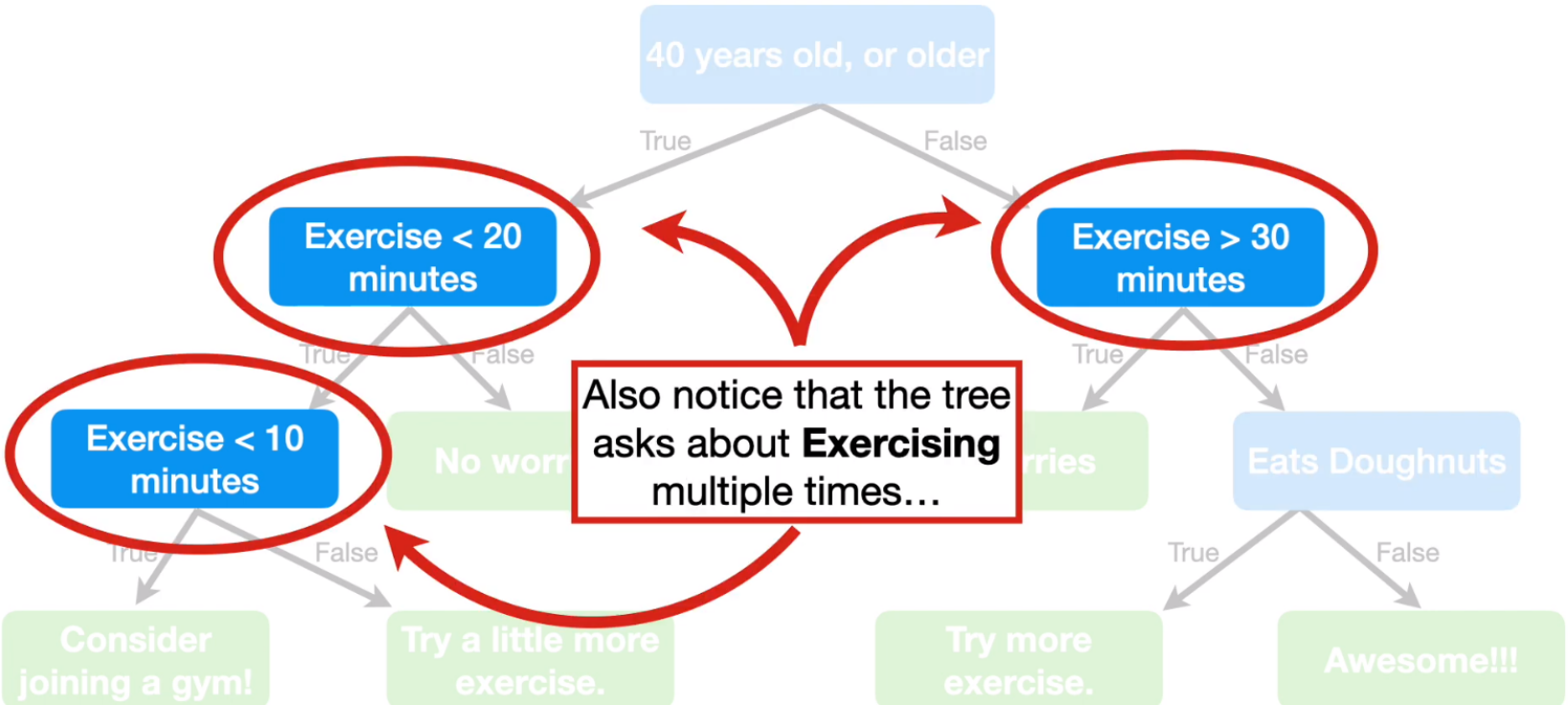
Example


40 years old, or older
True
False
Exercise < 20 minutes
Exercise > 30 minutes
True
False
True
False
Exercise < 10 minutes
No worries!
No worries
Eats Doughnuts
…with yes/no data.
True
False
True
Consider joining a gym!
Try a little more exercise.
Awesome!!!

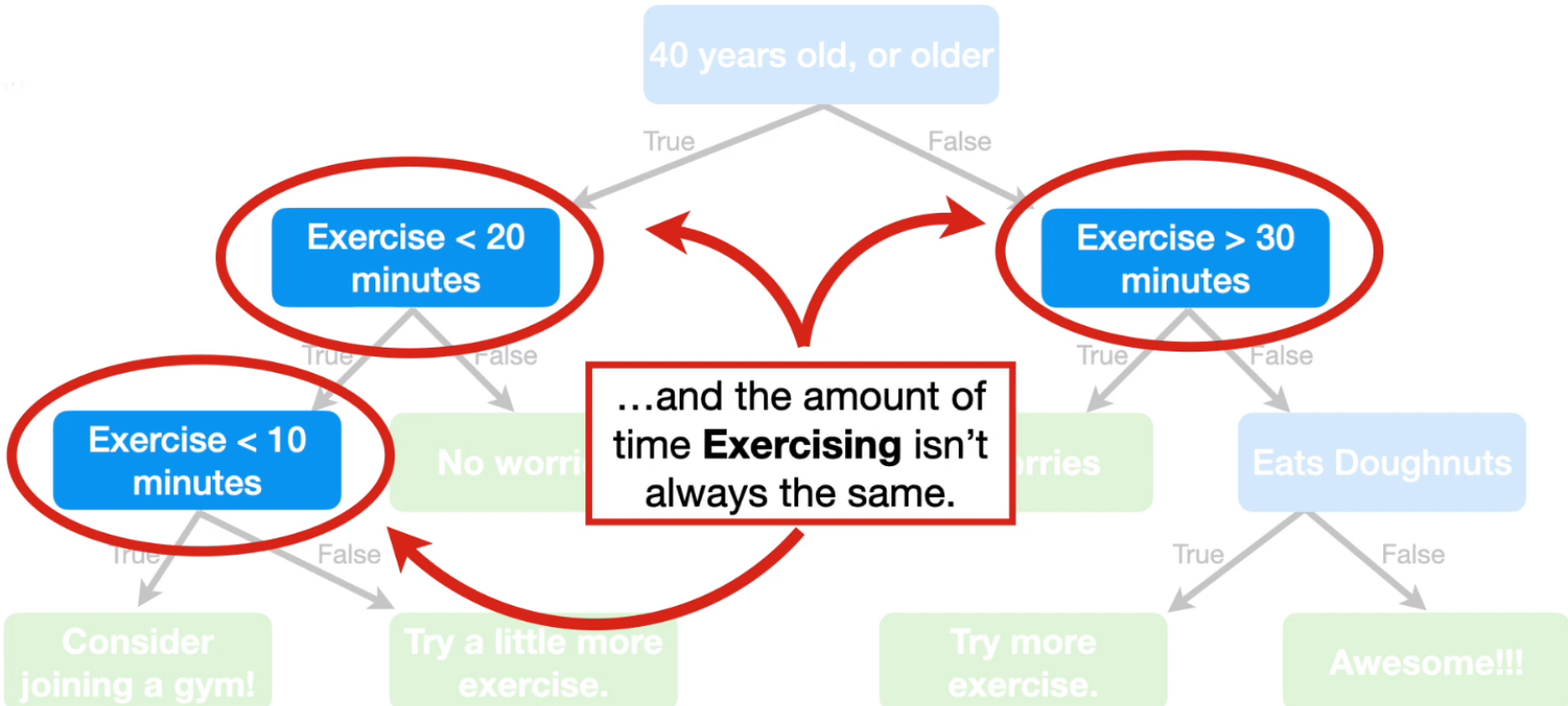
Example


40 years old, or older
True
False
Exercise < 20 minutes
Exercise > 30 minutes
True
False
True
False
Exercise < 10 minutes
No worries!
So it's OK to mix data types in the same tree.
Eats Doughnuts
True
False
True
False
Consider joining a gym!
Try a little more exercise.
Try more exercise.
Awesome!!!

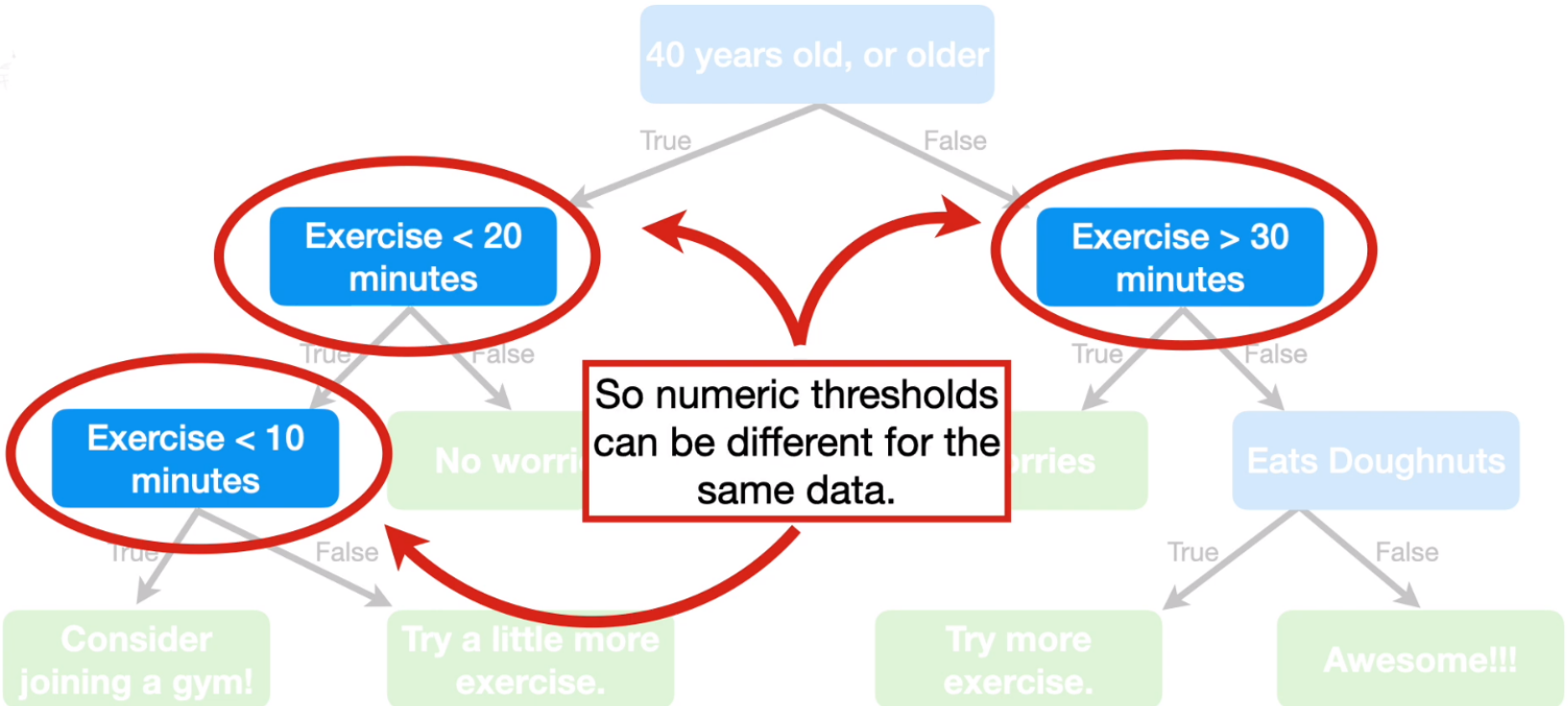
Example


40 years old, or older
True
False
Exercise < 20 minutes
Exercise > 30 minutes
True
False
True
False
Exercise < 10 minutes
Eats Doughnuts
Also notice that the tree asks about **Exercising** multiple times…
True
False
True
False
Consider joining a gym!
Try a little more exercise.
Try more exercise.
Awesome!!!

Example


40 years old, or older
True
False
Exercise < 20 minutes
Exercise > 30 minutes
True
False
True
False
Exercise < 10 minutes
No worri
...and the amount of time **Exercising** isn't always the same.
rries
Eats Doughnuts
True
False
True
False
Consider joining a gym!
Try a little more exercise.
Try more exercise.
Awesome!!!

Example


40 years old, or older
True
False
Exercise < 20 minutes
Exercise > 30 minutes
True
False
True
False
Exercise < 10 minutes
So numeric thresholds can be different for the same data.
Eats Doughnuts
True
False
True
False
Consider joining a gym!
Try a little more exercise.
Try more exercise.
Awesome!!!

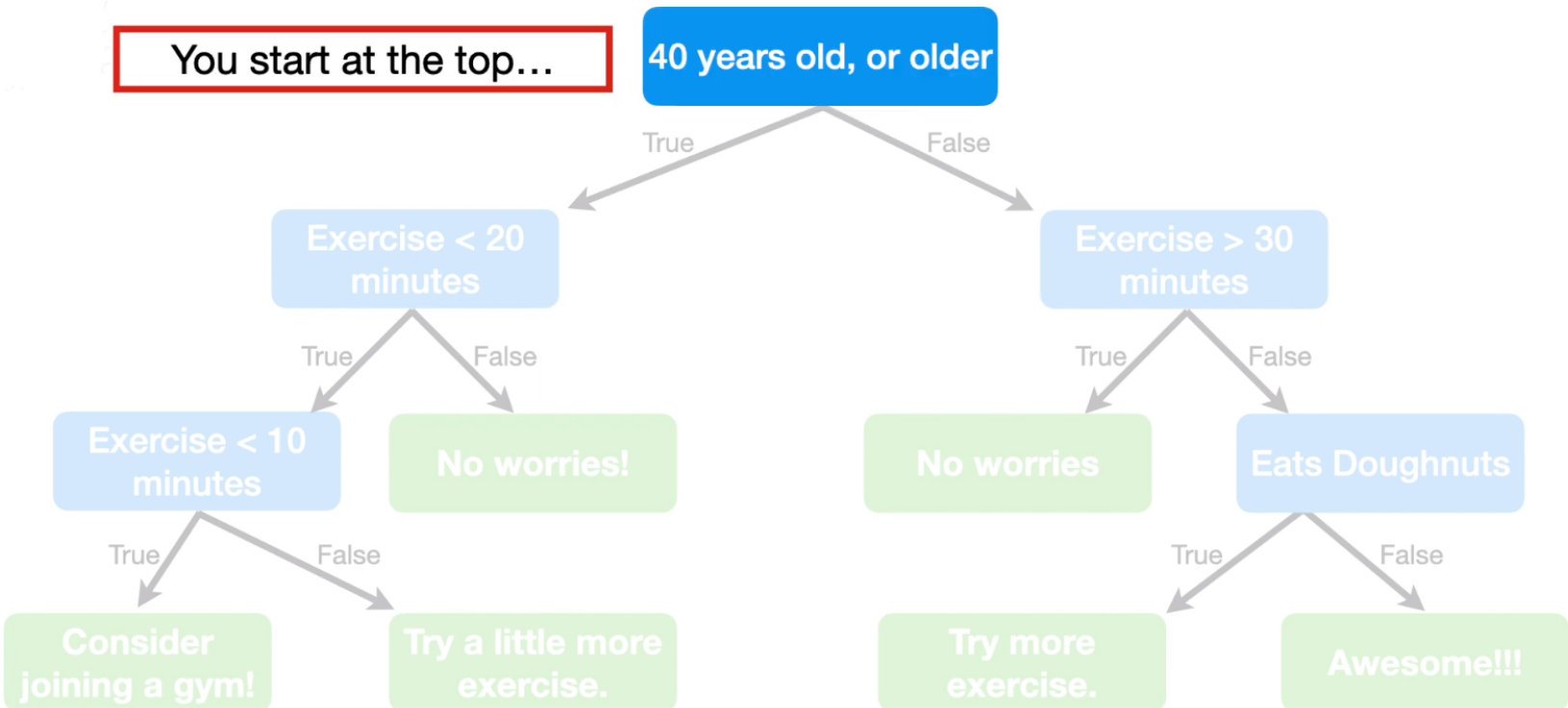
Example


40 years old, or older
True
False
Exercise < 20 minutes
Lastly, the final classifications can be repeated.
Exercise > 30 minutes
True
False
True
False
Exercise < 10 minutes
No worries!
No worries
Eats Doughnuts
True
False
True
False
Consider joining a gym!
Try a little more exercise.
Try more exercise.
Awesome!!!

Example


40 years old, or older
True
False
Exercise < 20 minutes
Exercise > 30 minutes
For the most part, Classification Trees are pretty easy to work with.
True
False
True
False
Exercise < 10 minutes
No worries!
No worries
Eats Doughnuts
True
False
True
False
Consider joining a gym!
Try a little more exercise.
Try more exercise.
Awesome!!!

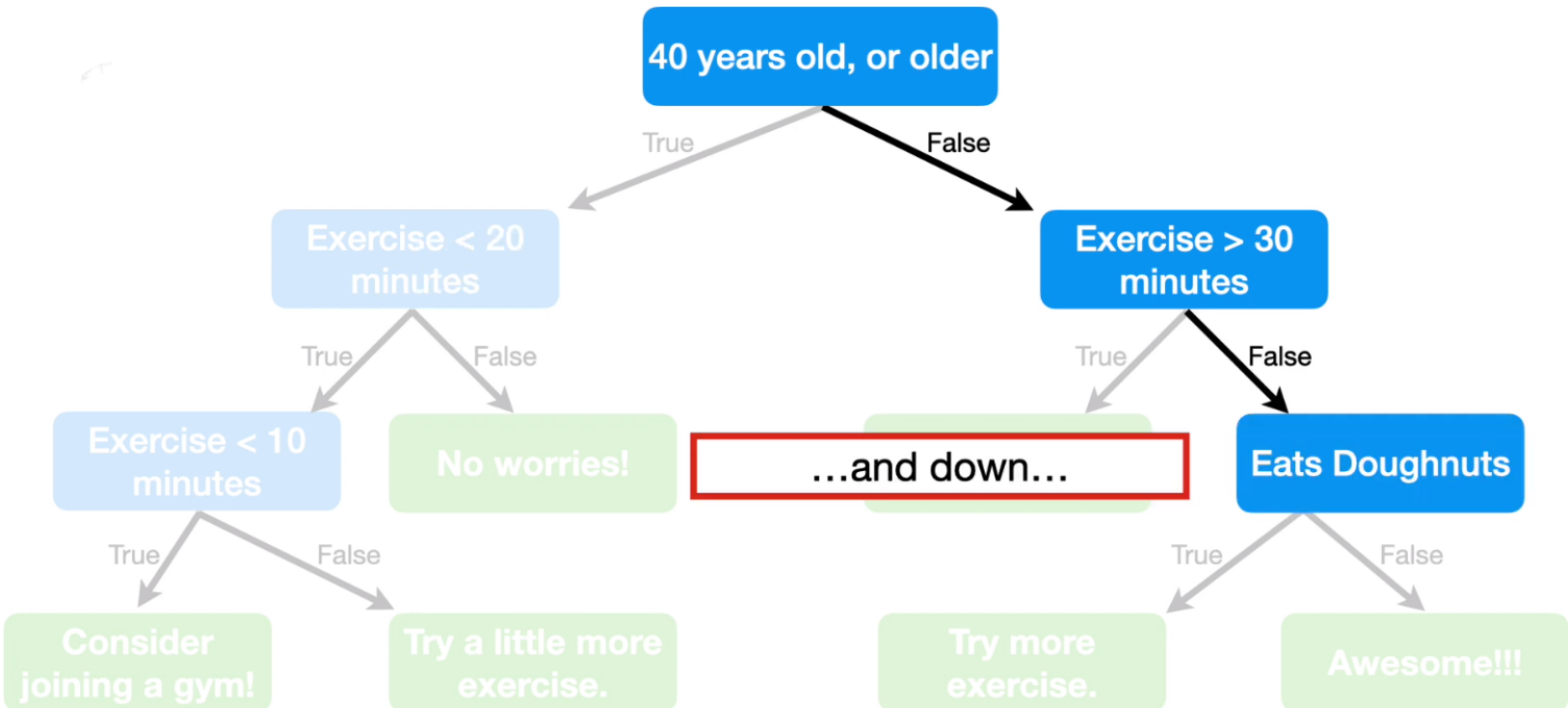
Example


You start at the top...
40 years old, or older
True
False
Exercise < 20 minutes
Exercise > 30 minutes
True
False
True
False
Exercise < 10 minutes
No worries!
No worries
Eats Doughnuts
True
False
True
False
Consider joining a gym!
Try a little more exercise.
Try more exercise.
Awesome!!!

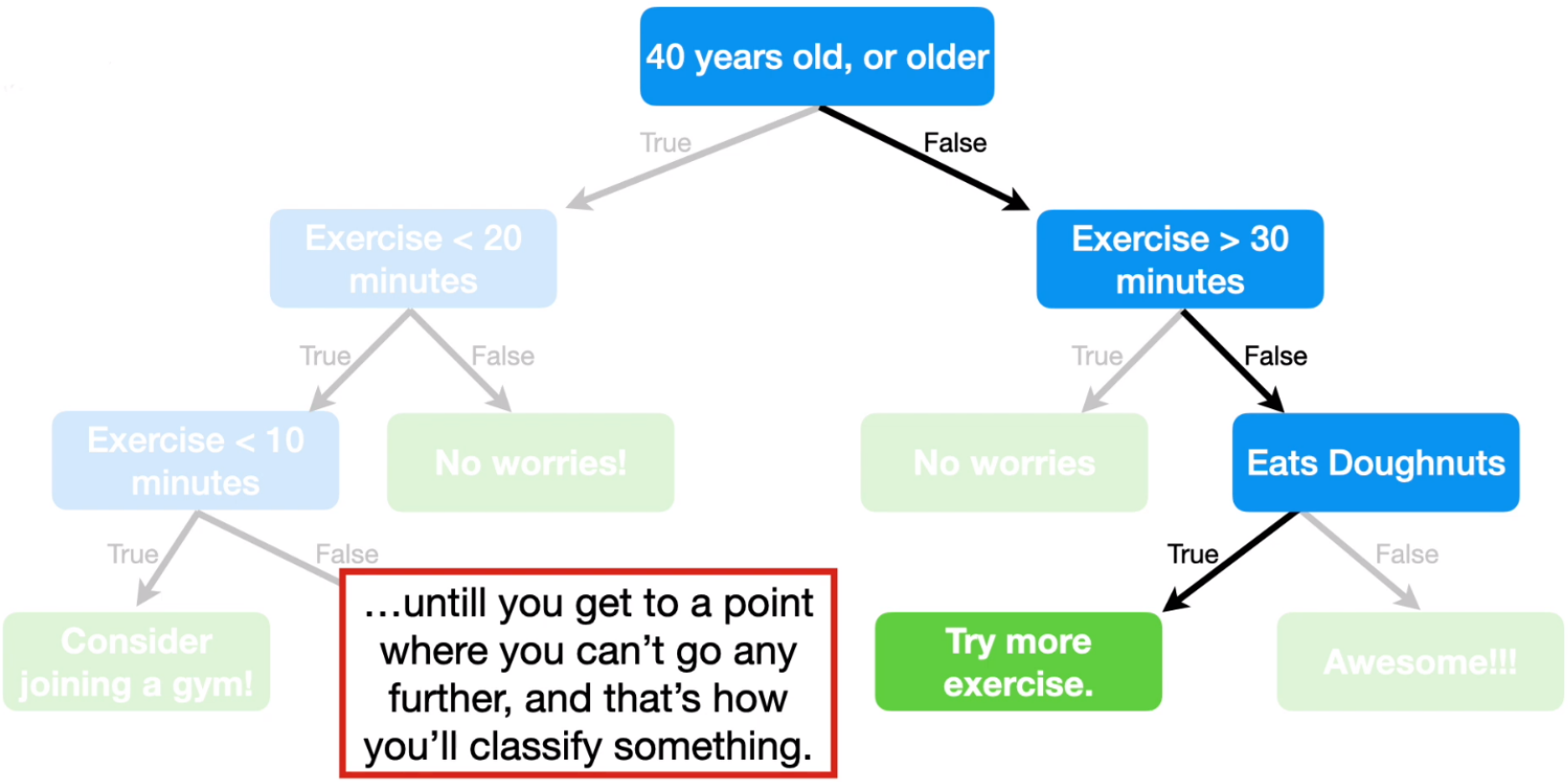
Example


40 years old, or older
True
False
…and work your way down…
Exercise > 30 minutes
True
False
True
False
Exercise < 10 minutes
No worries!
No worries
Eats Doughnuts
True
False
True
False
Consider joining a gym!
Try a little more exercise.
Try more exercise.
Awesome!!!

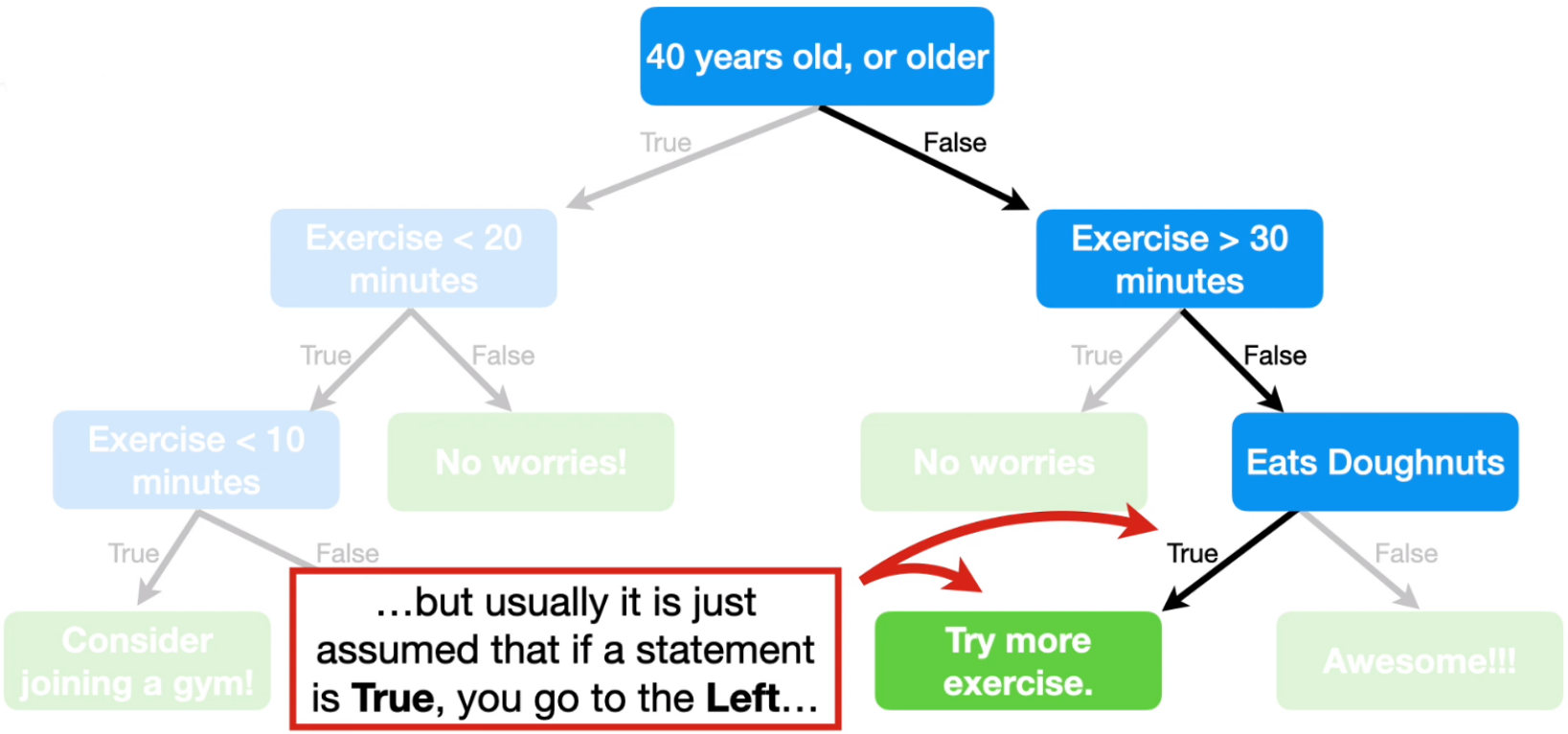
Example


40 years old, or older
True
False
Exercise < 20 minutes
Exercise > 30 minutes
True
False
True
False
Exercise < 10 minutes
No worries!
…and down…
Eats Doughnuts
True
False
True
False
Consider joining a gym!
Try a little more exercise.
Try more exercise.
Awesome!!!

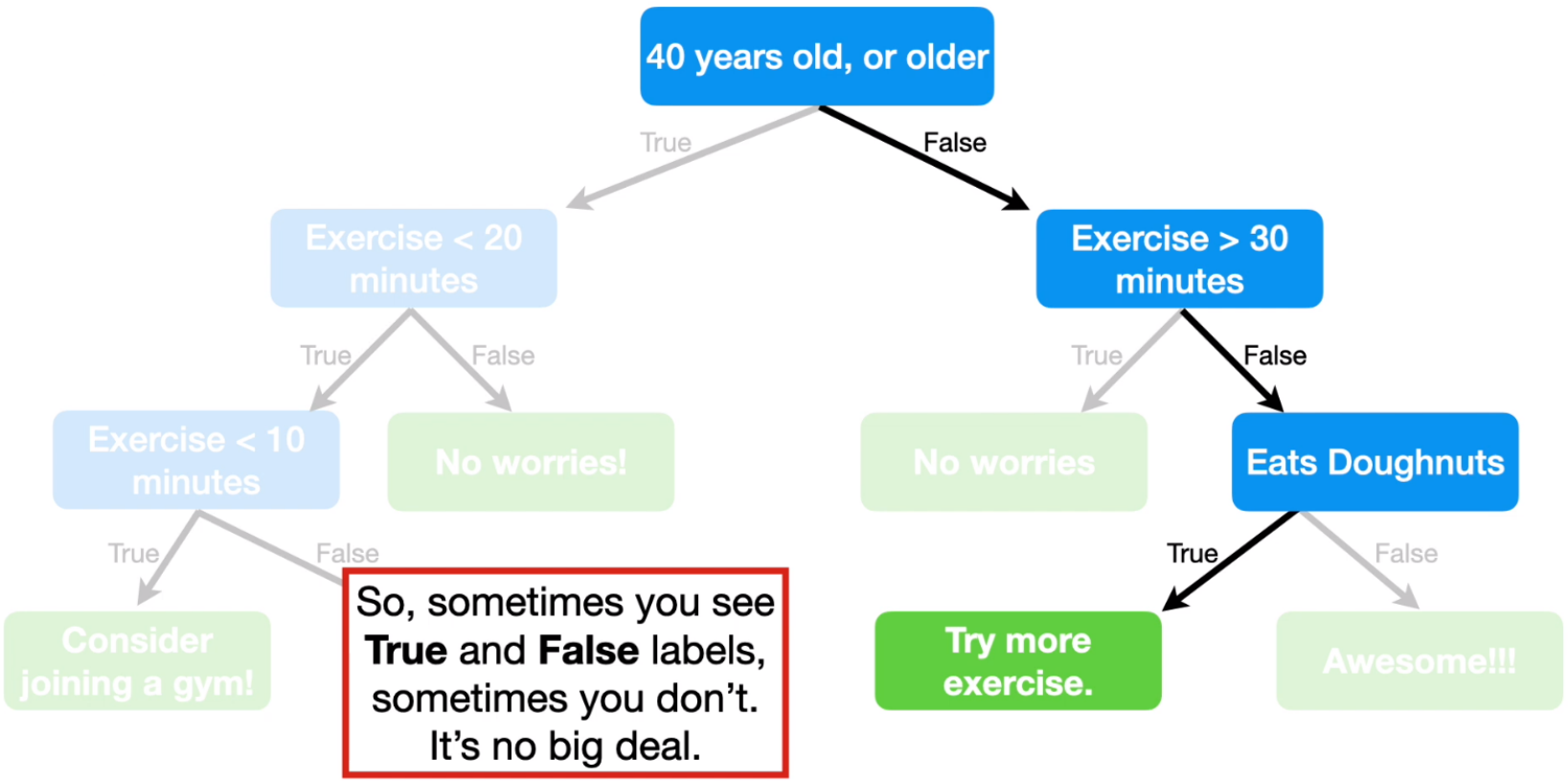
Example


40 years old, or older
True
False
Exercise < 20 minutes
Exercise > 30 minutes
True
False
True
False
Exercise < 10 minutes
No worries!
No worries
Eats Doughnuts
True
False
True
False
Consider joining a gym!
…untill you get to a point where you can't go any further, and that's how you'll classify something.
Try more exercise.
Awesome!!!

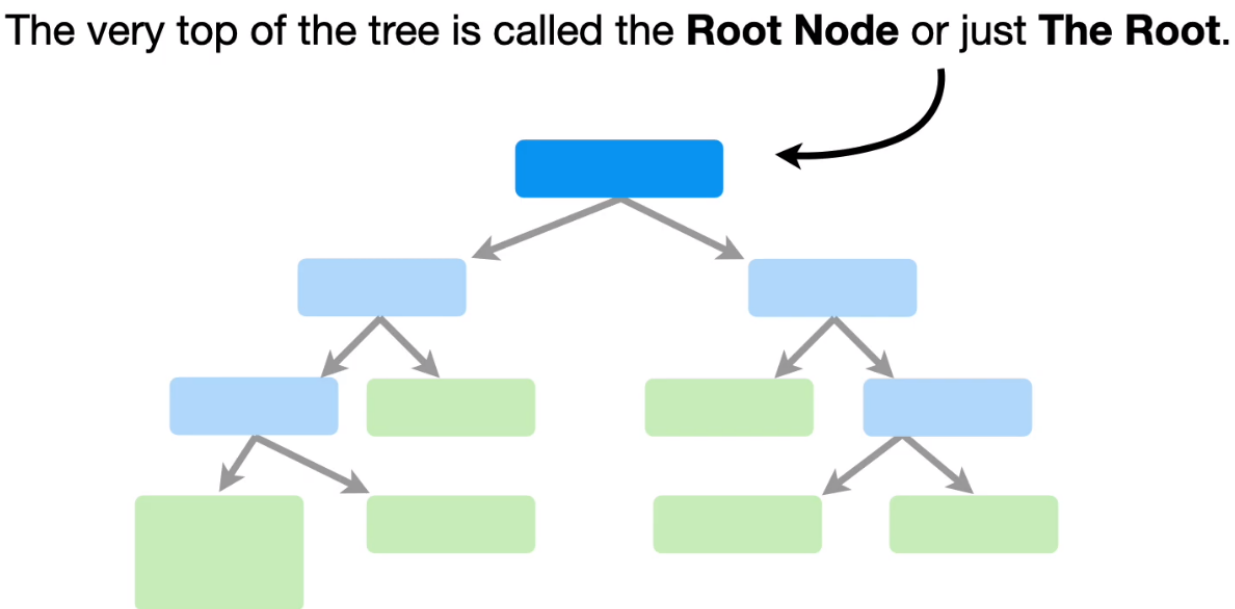
Example


40 years old, or older
True
False
Exercise < 20 minutes
Exercise > 30 minutes
True
False
True
False
Exercise < 10 minutes
No worries!
No worries
Eats Doughnuts
True
False
True
False
Consider joining a gym!
…but usually it is just assumed that if a statement is True, you go to the Left…
Try more exercise.
Awesome!!!

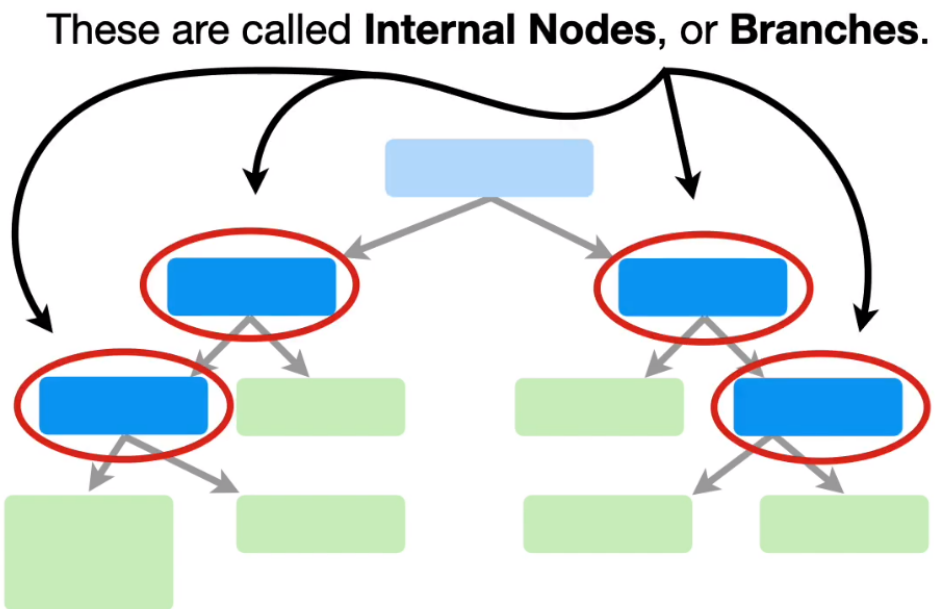
Example


40 years old, or older
True
False
Exercise < 20 minutes
Exercise > 30 minutes
True
False
True
False
Exercise < 10 minutes
No worries!
No worries
Eats Doughnuts
True
False
True
False
Consider joining a gym!
So, sometimes you see True and False labels, sometimes you don't. It's no big deal.
Try more exercise.
Awesome!!!

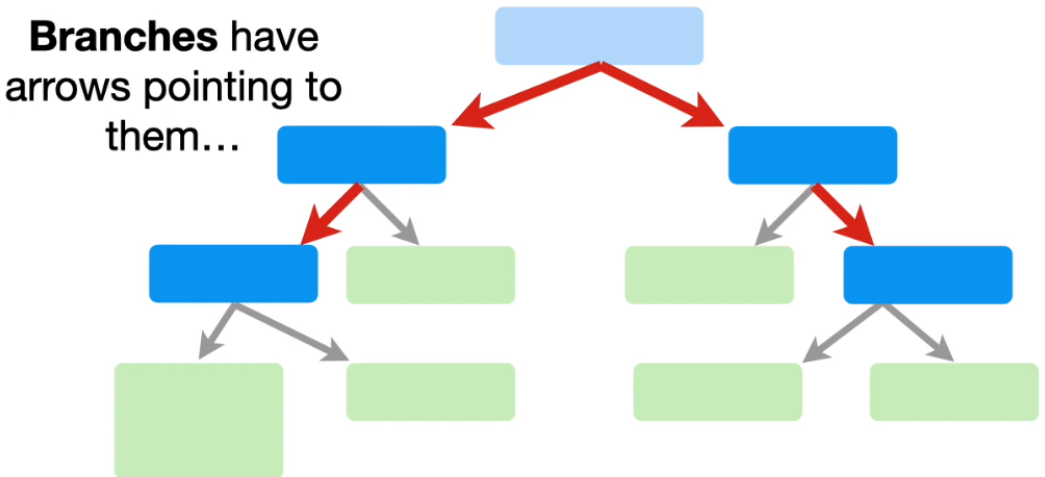
Example


The very top of the tree is called the **Root Node** or just **The Root.**

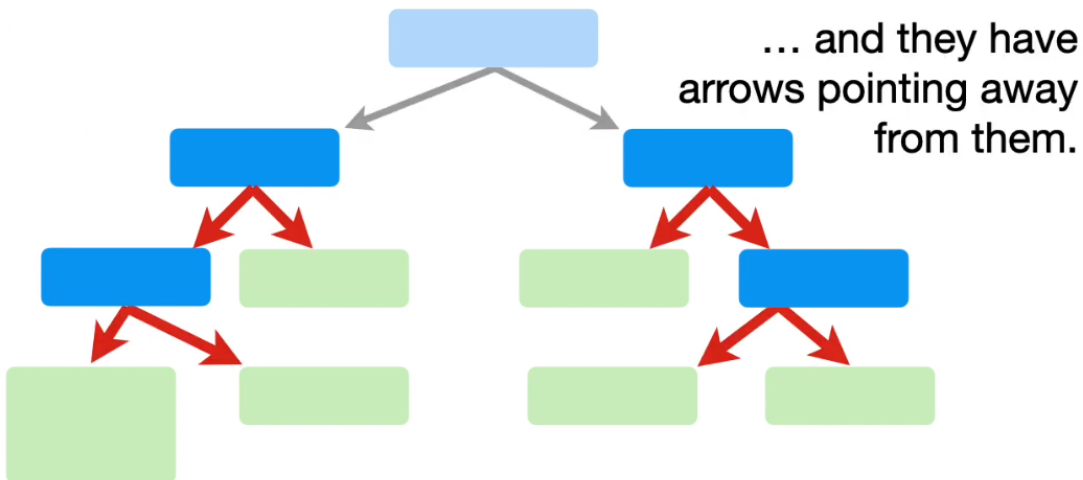
Example



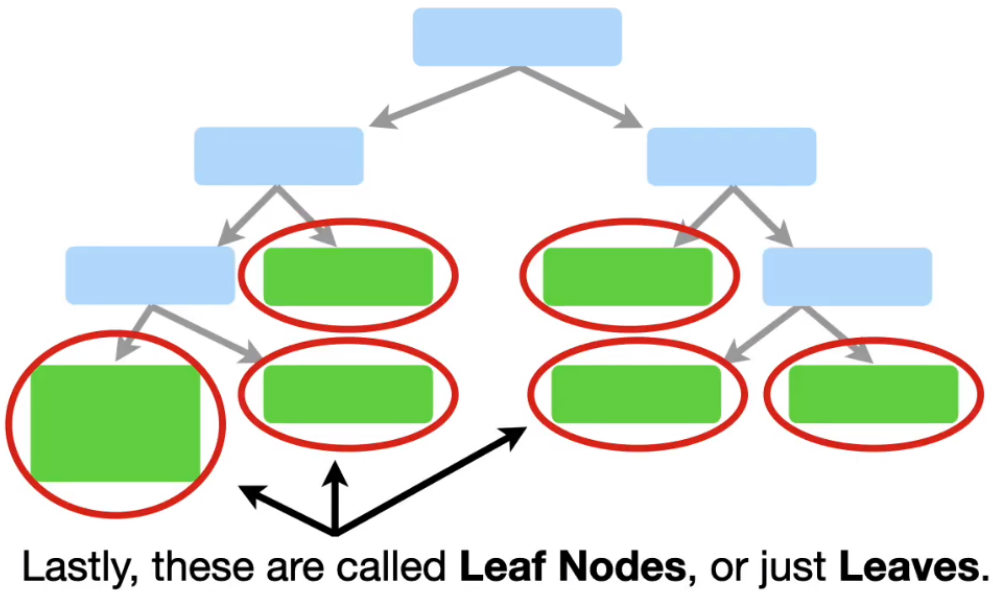
Example



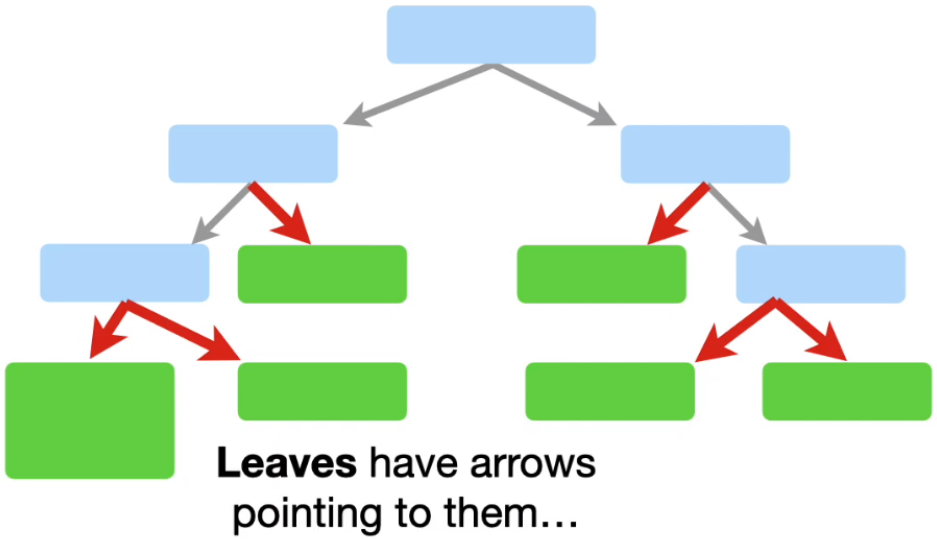
Example



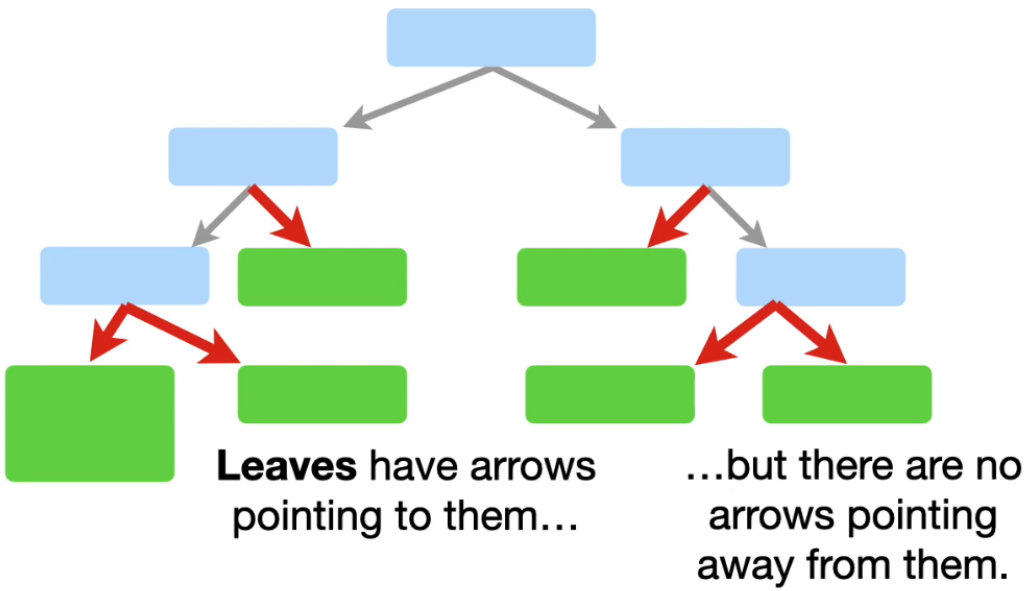
Example



Example

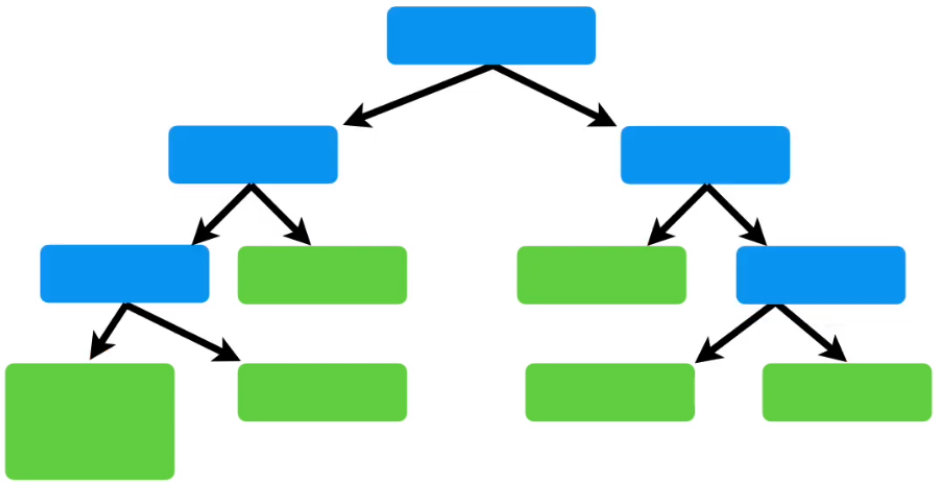


Example


Leaves have arrows pointing to them…
…but there are no arrows pointing away from them.

Example

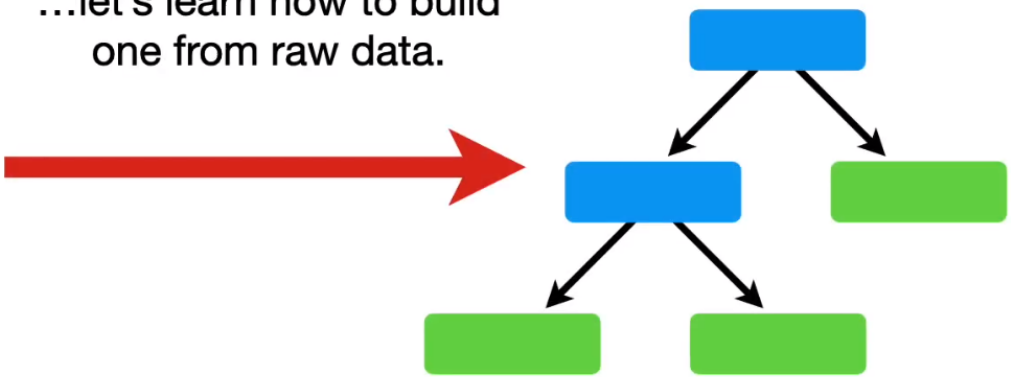
Now that we know how to use and interpret **Classification Trees**...



Example

Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

...let's learn how to build one from raw data.



At the board.

Example

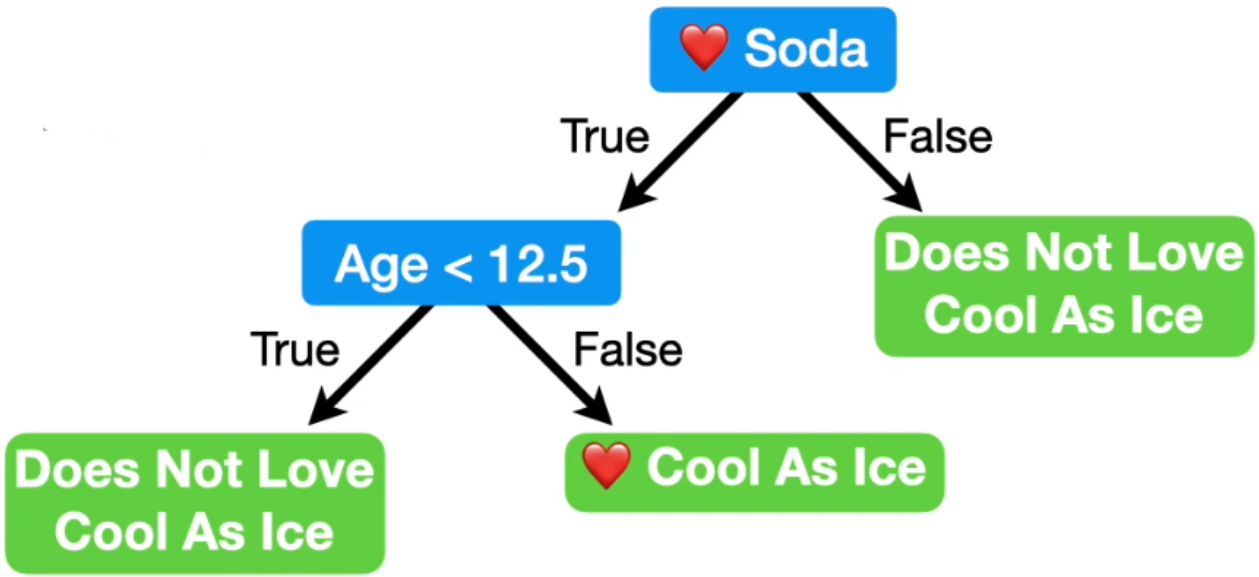
Which splitting is the best? How to construct the tree?

- **Gini Index** approach
- Information entropy
- Variance reduction

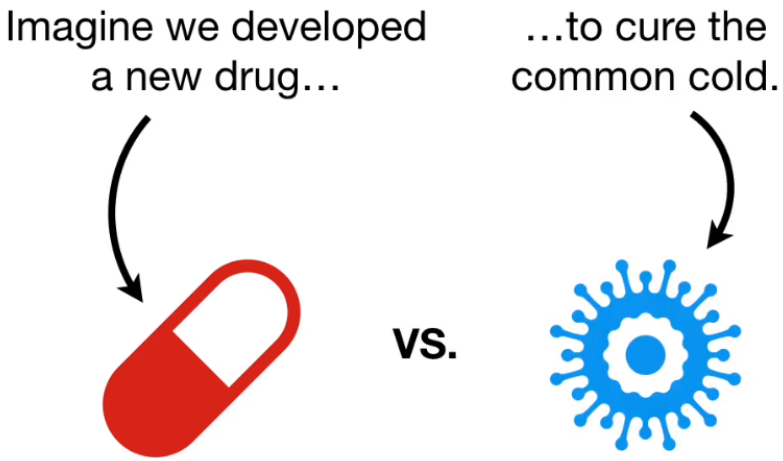
Gini Index

- let $p(i|t)$ denote the fraction of records belonging to class i at a given node t
- $\Rightarrow G(t) = 1 - p(0|t)^2 - p(1|t)^2$ is the Gini Index of the **Leaf** t
- At the parent, or at the **question** node t: $\frac{N_0}{N_0+N_1} G(t_0) + \frac{N_0}{N_0+N_1} G(t_1)$
- We should examine every possible question. What are the possible questions?

Example: solution



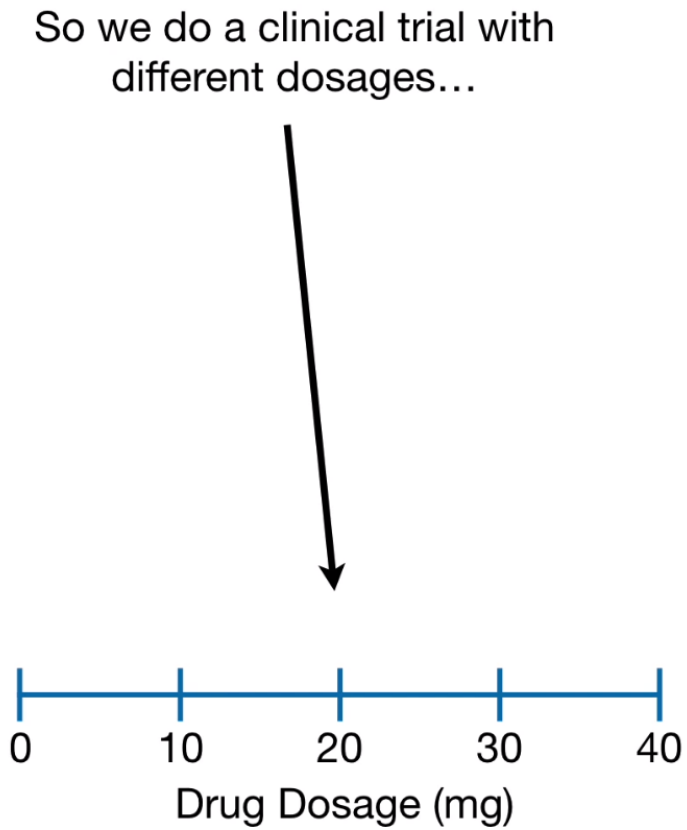
Regression Trees



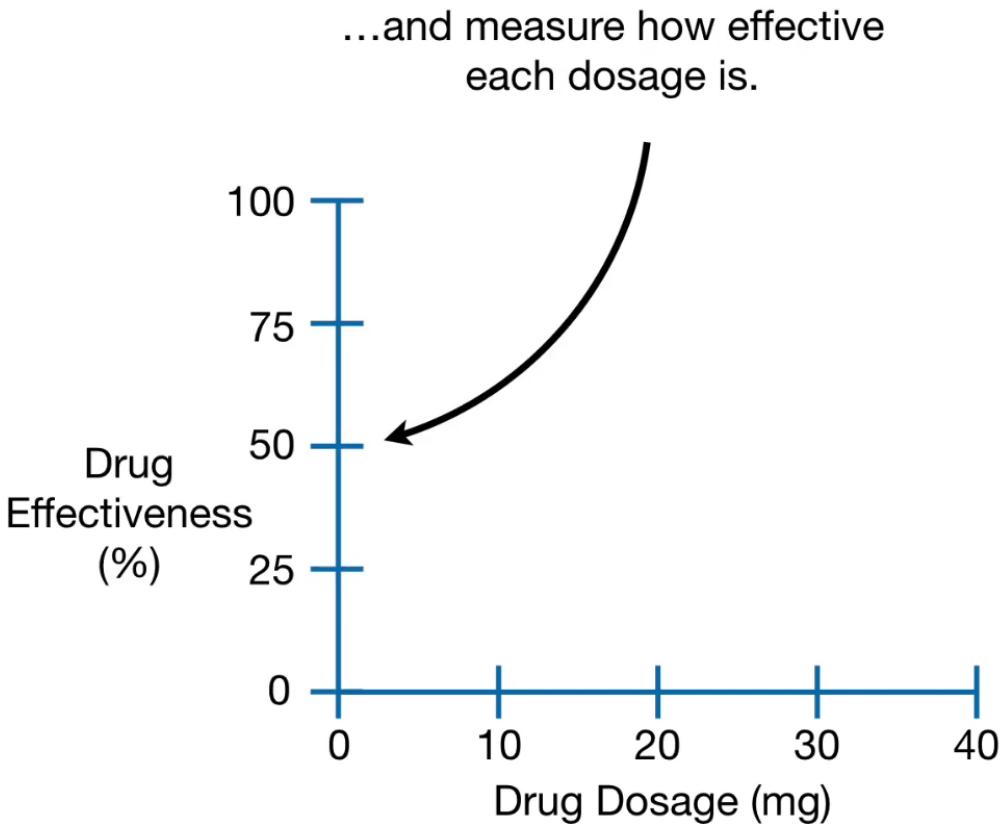
Regression Trees

However, we don't know the optimal dosage to give to patients.

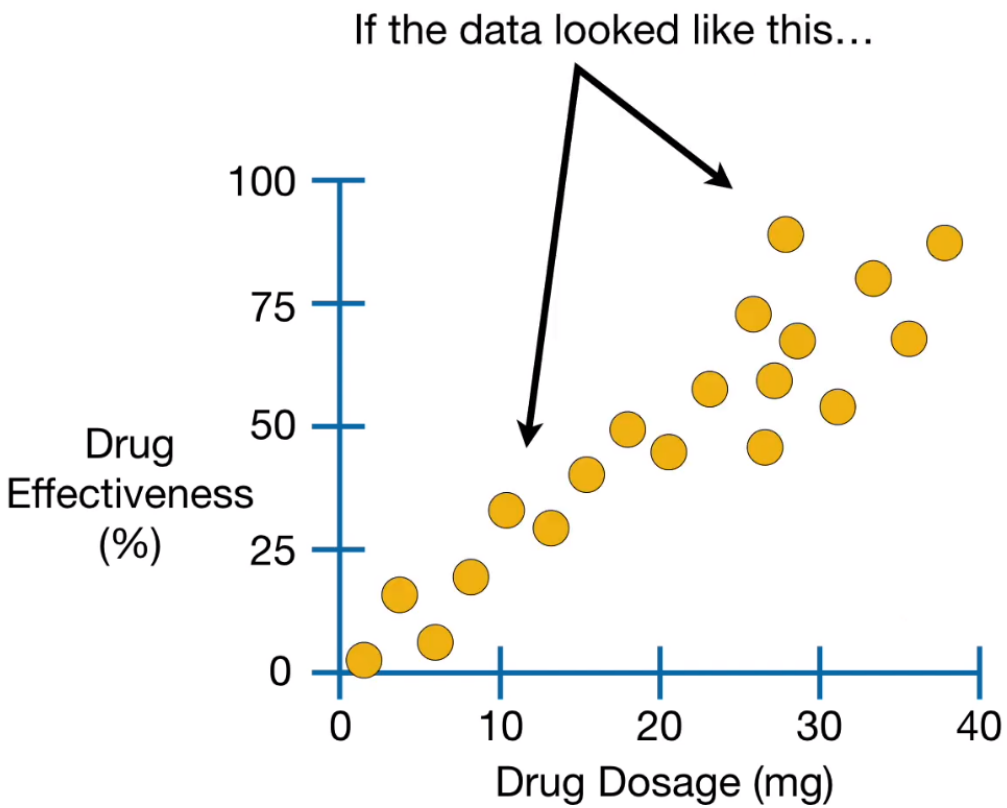
Regression Trees



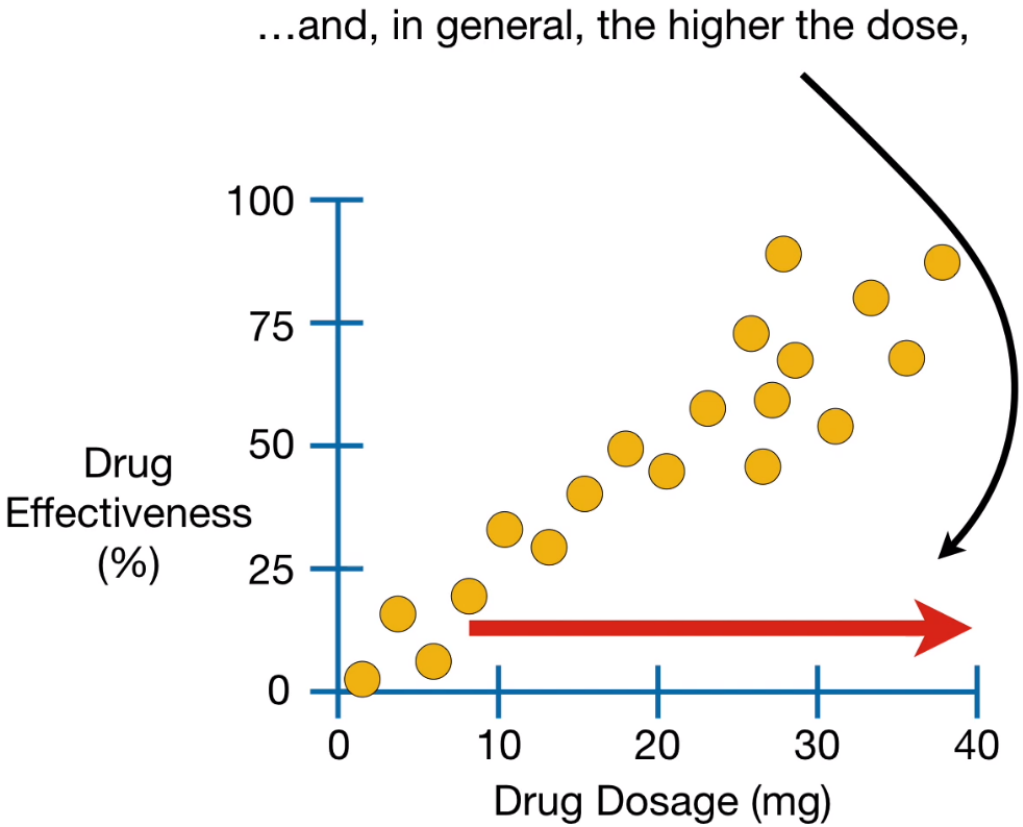
Regression Trees



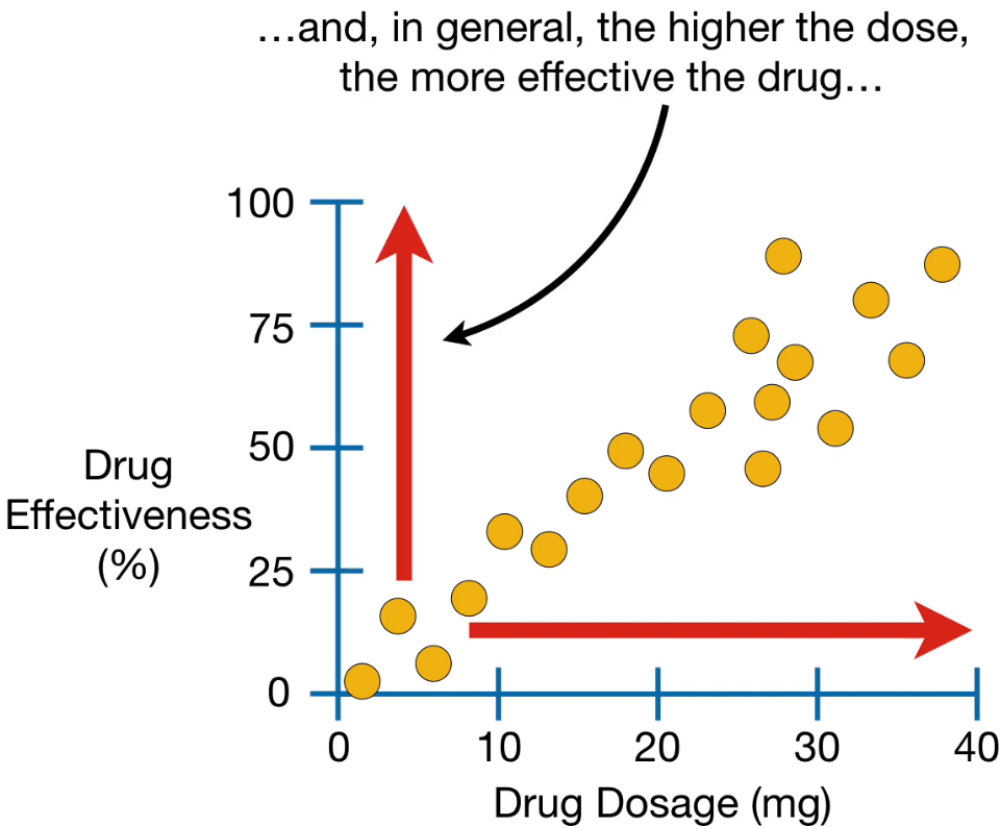
Regression Trees



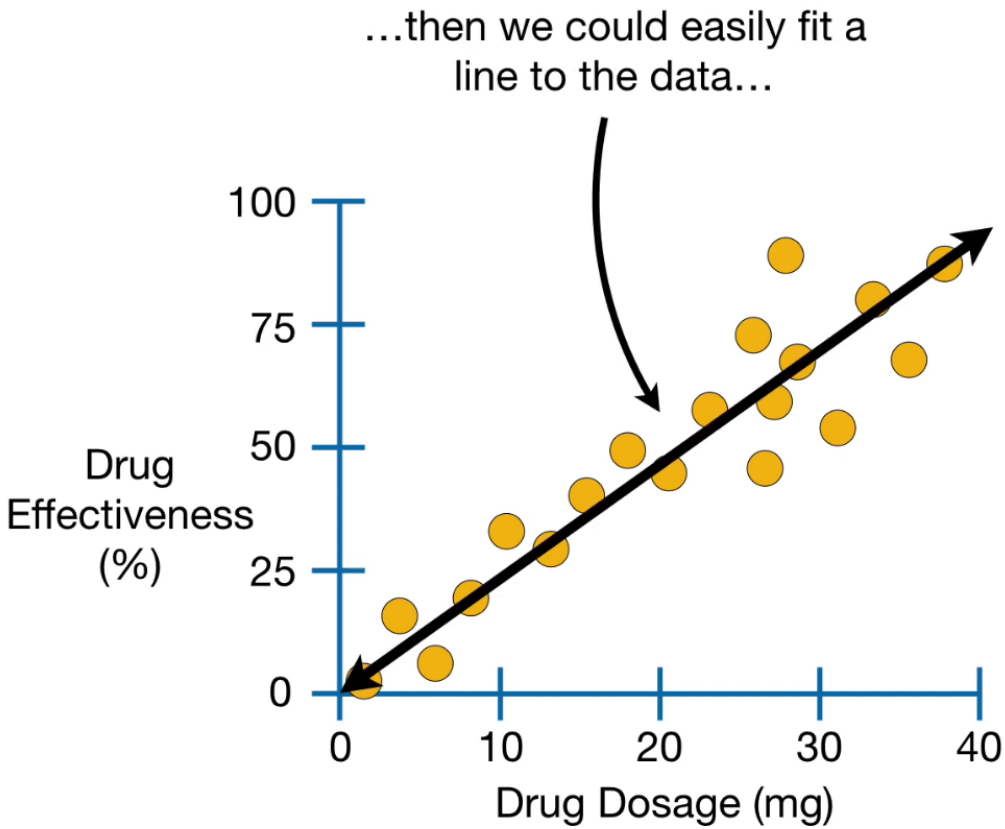
Regression Trees



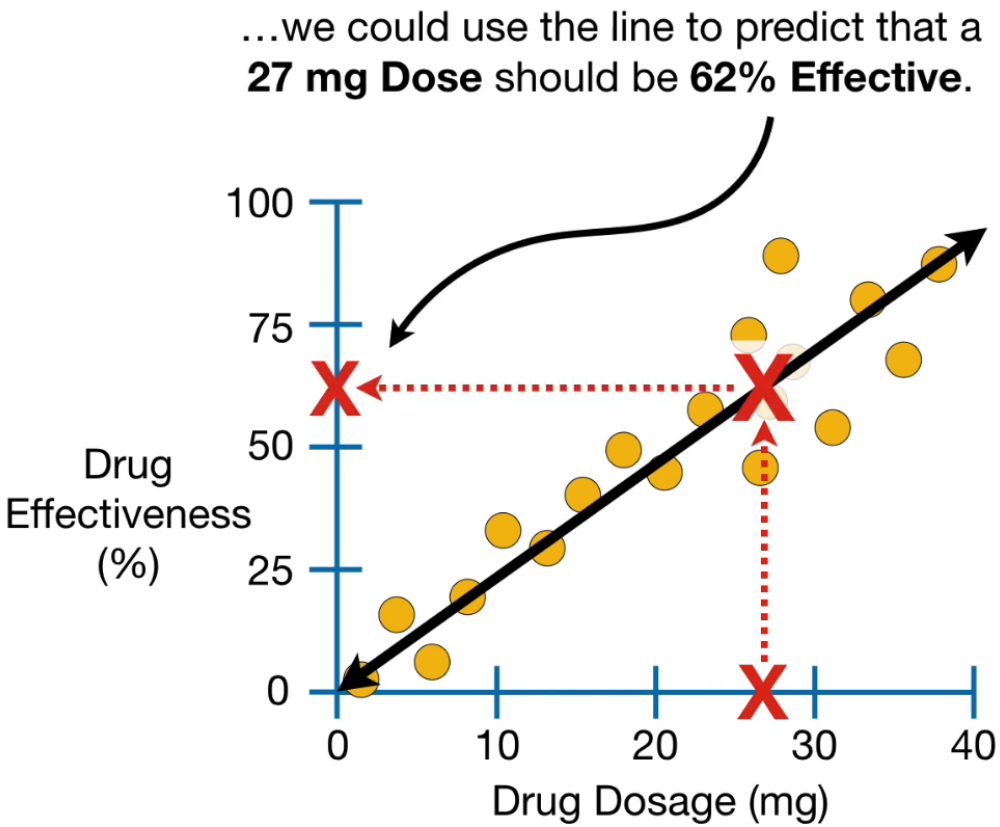
Regression Trees



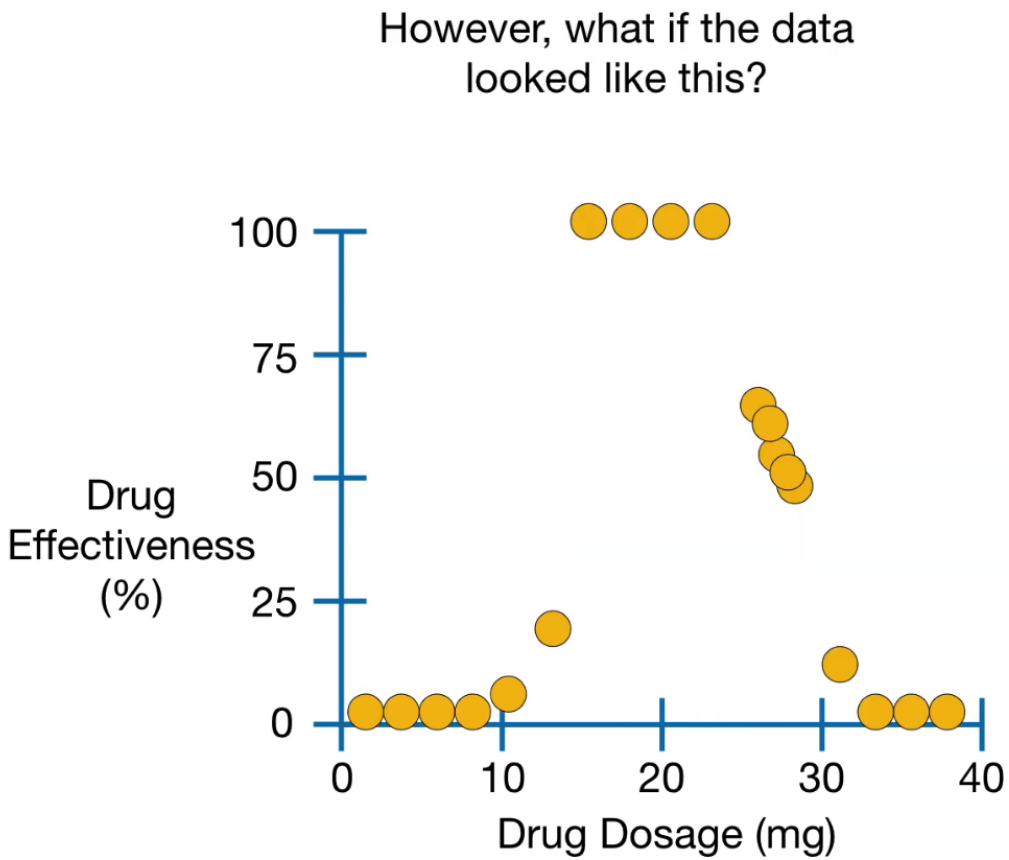
Regression Trees



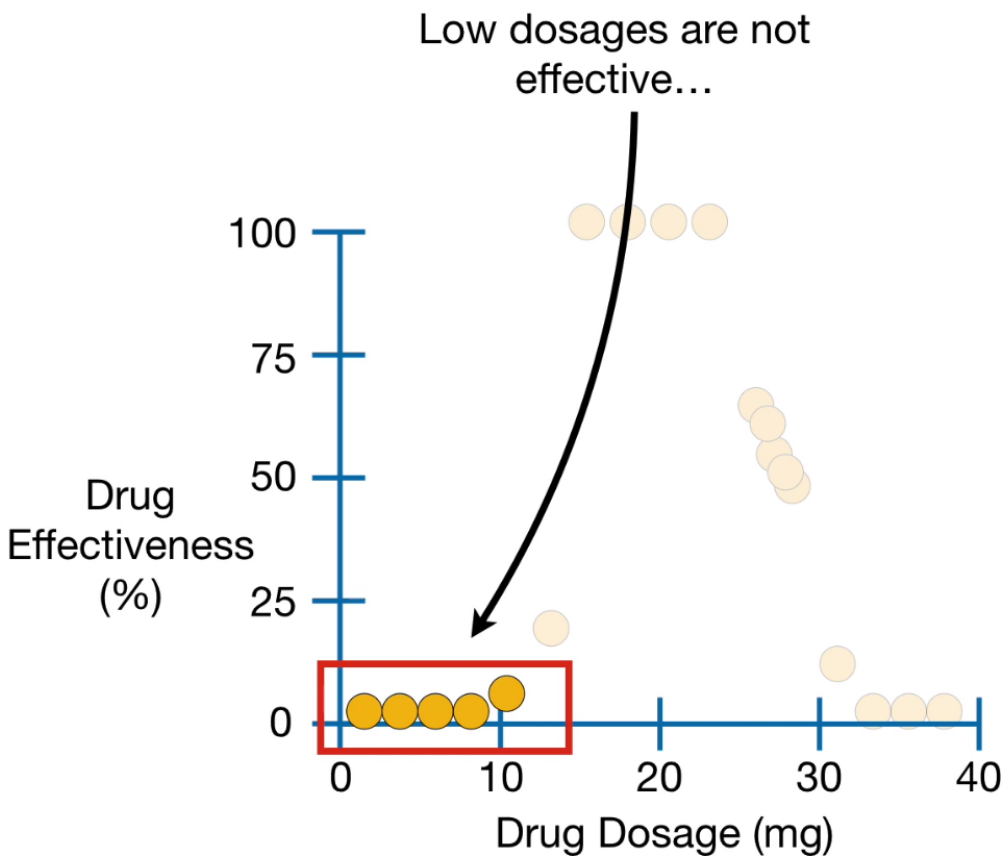
Regression Trees



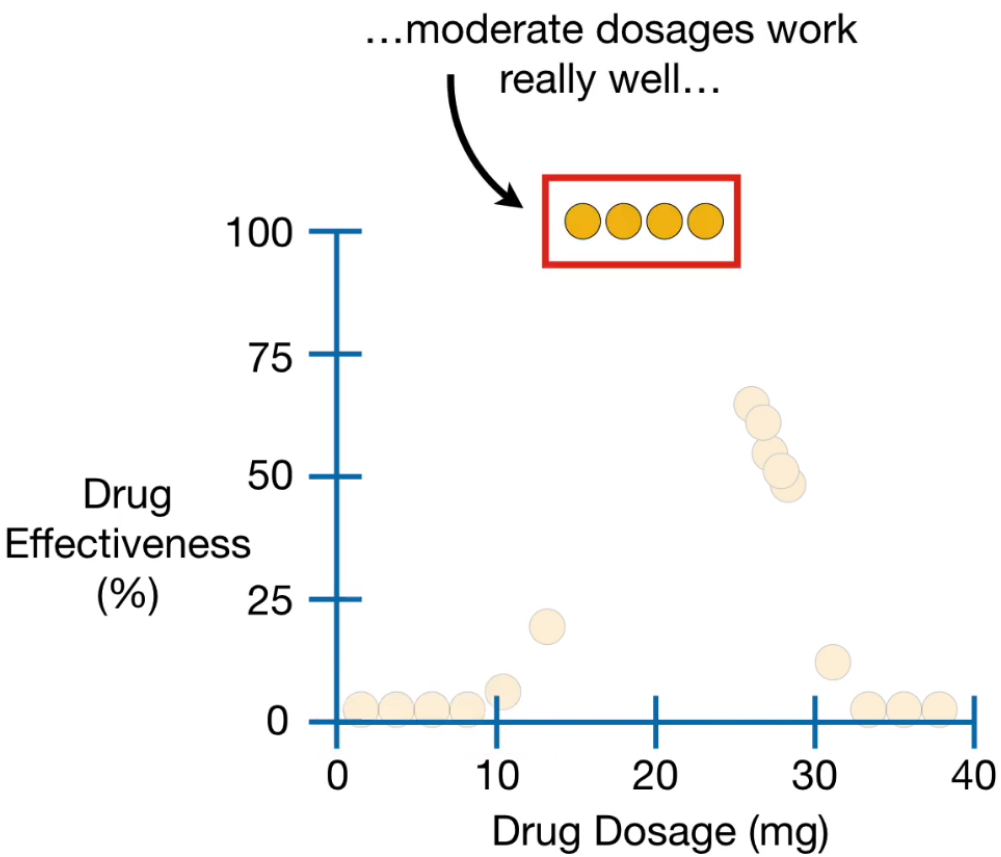
Regression Trees



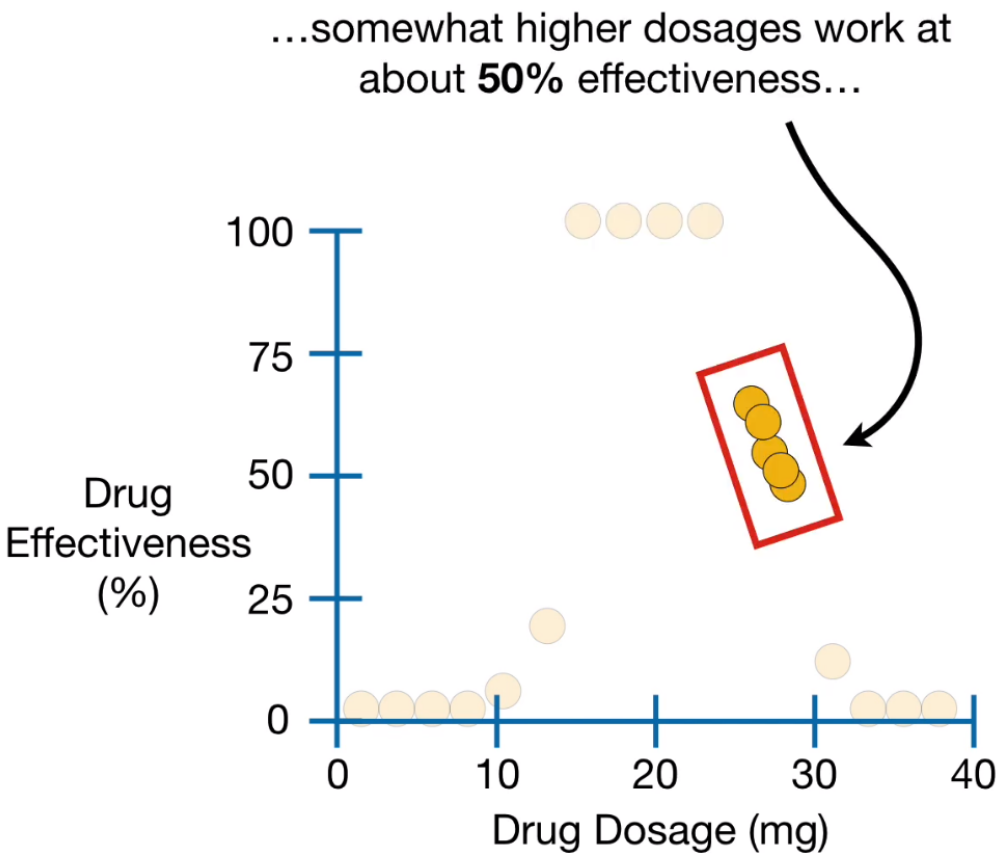
Regression Trees



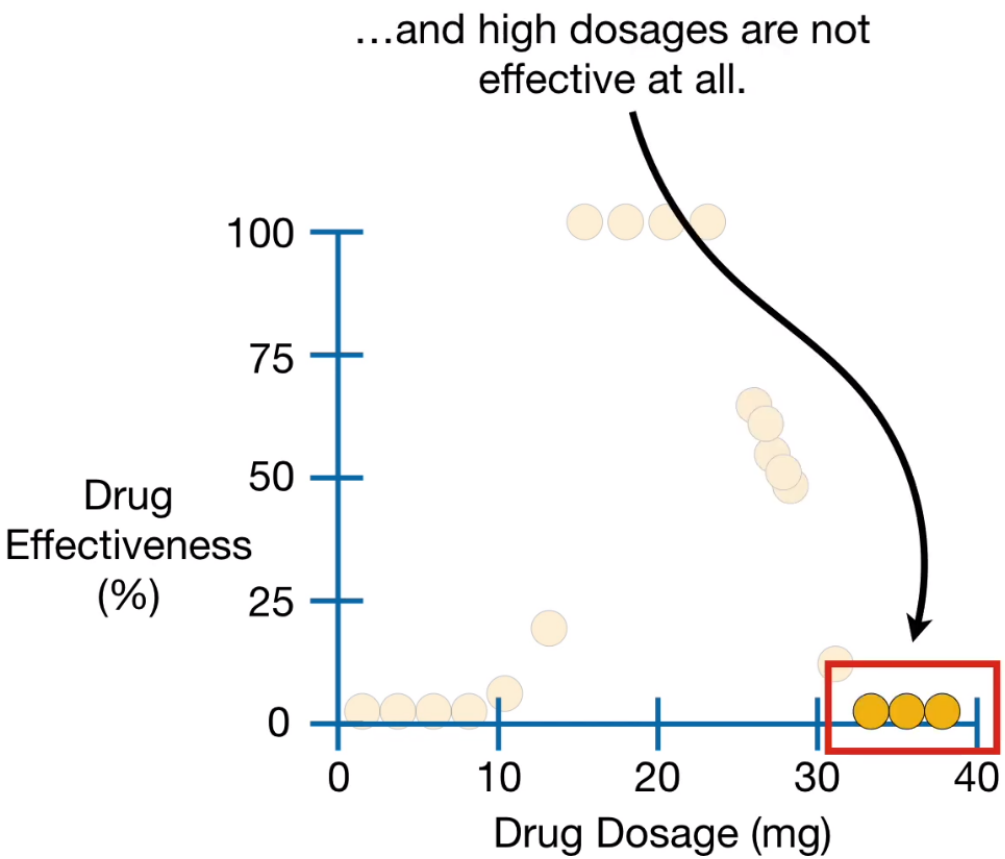
Regression Trees



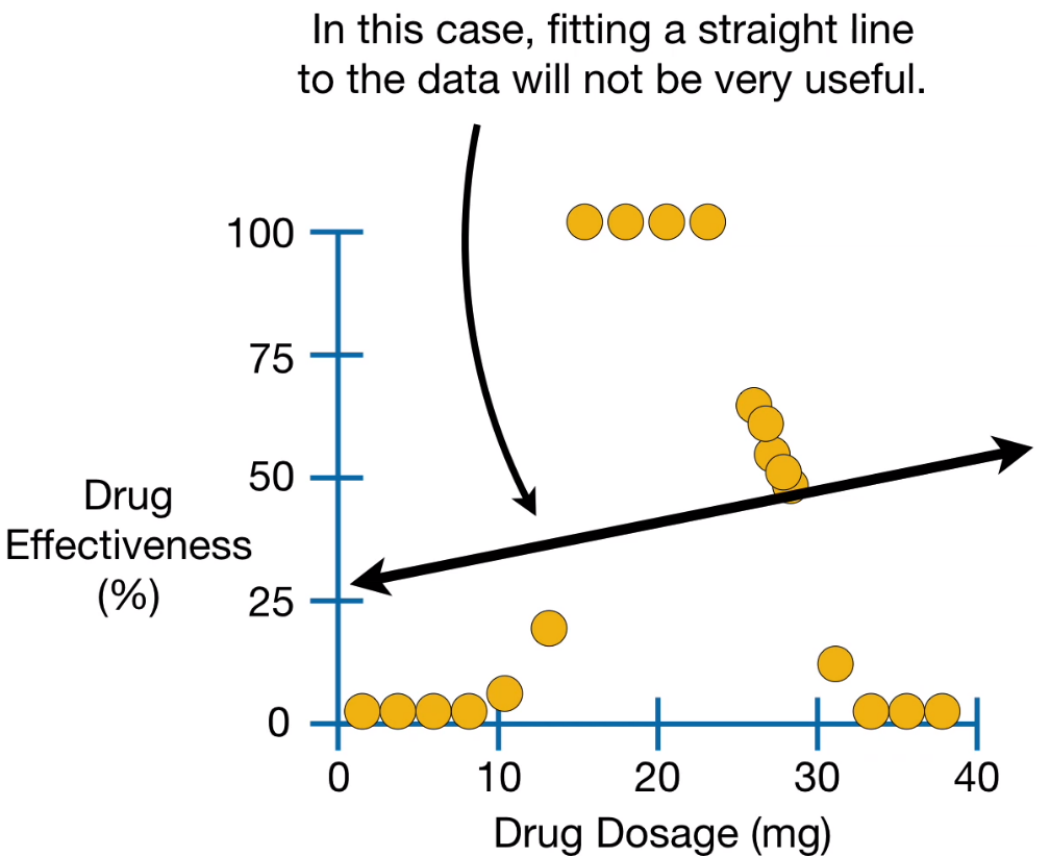
Regression Trees



Regression Trees

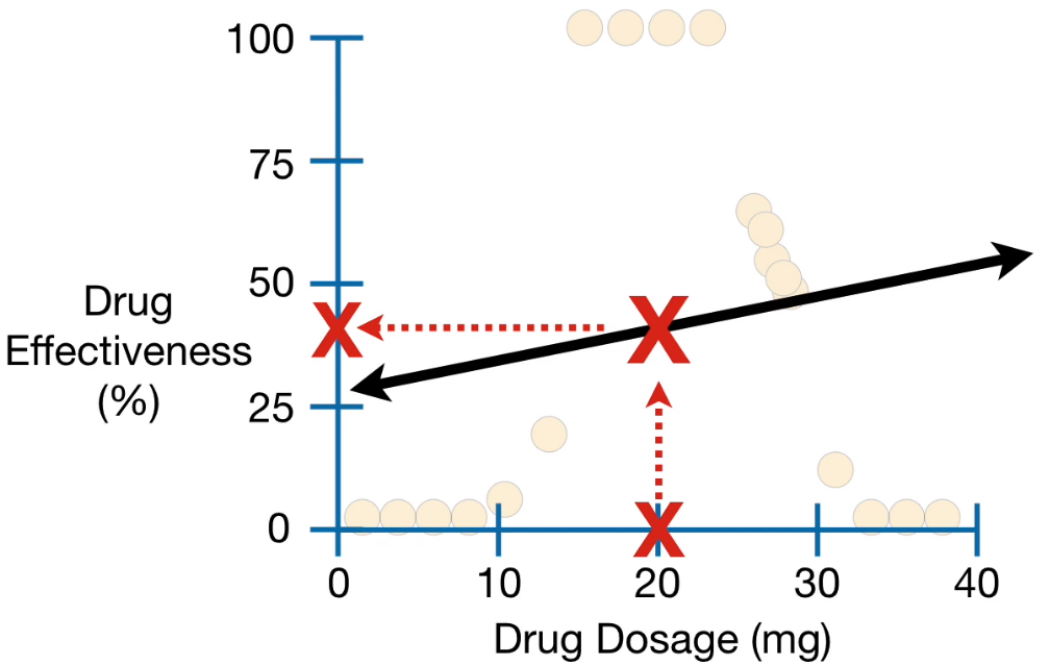


Regression Trees

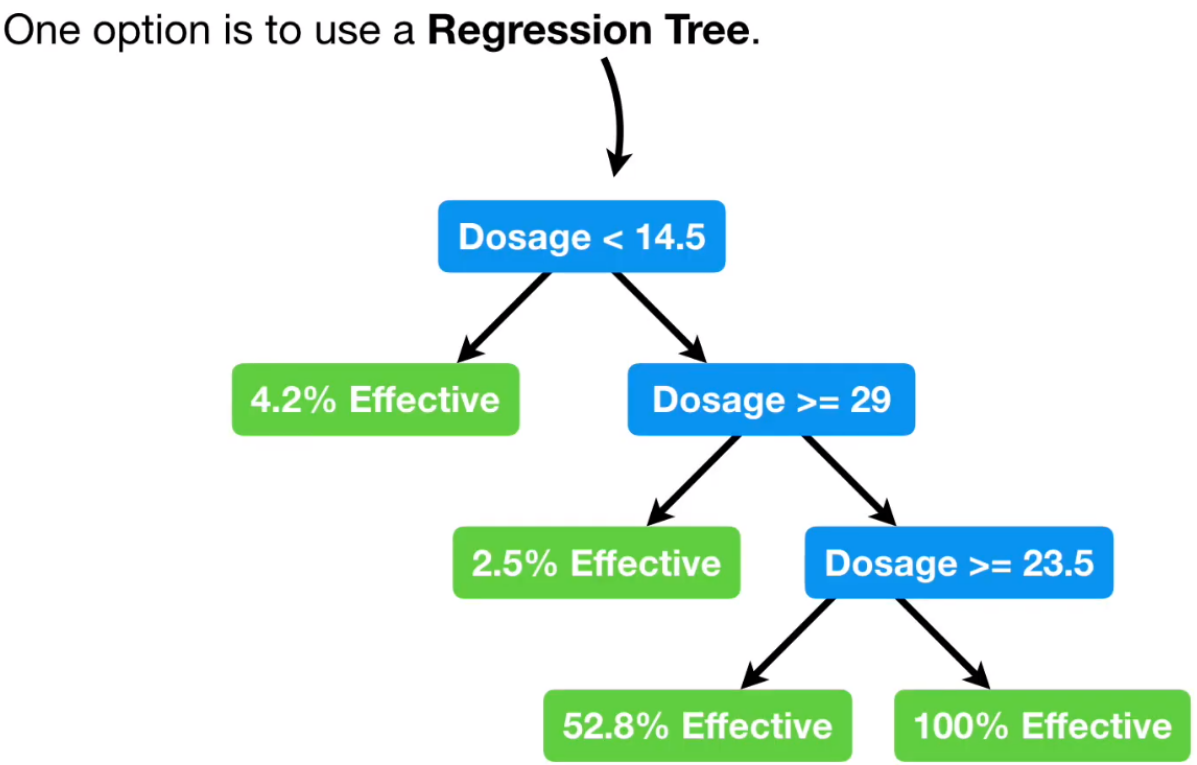


Regression Trees

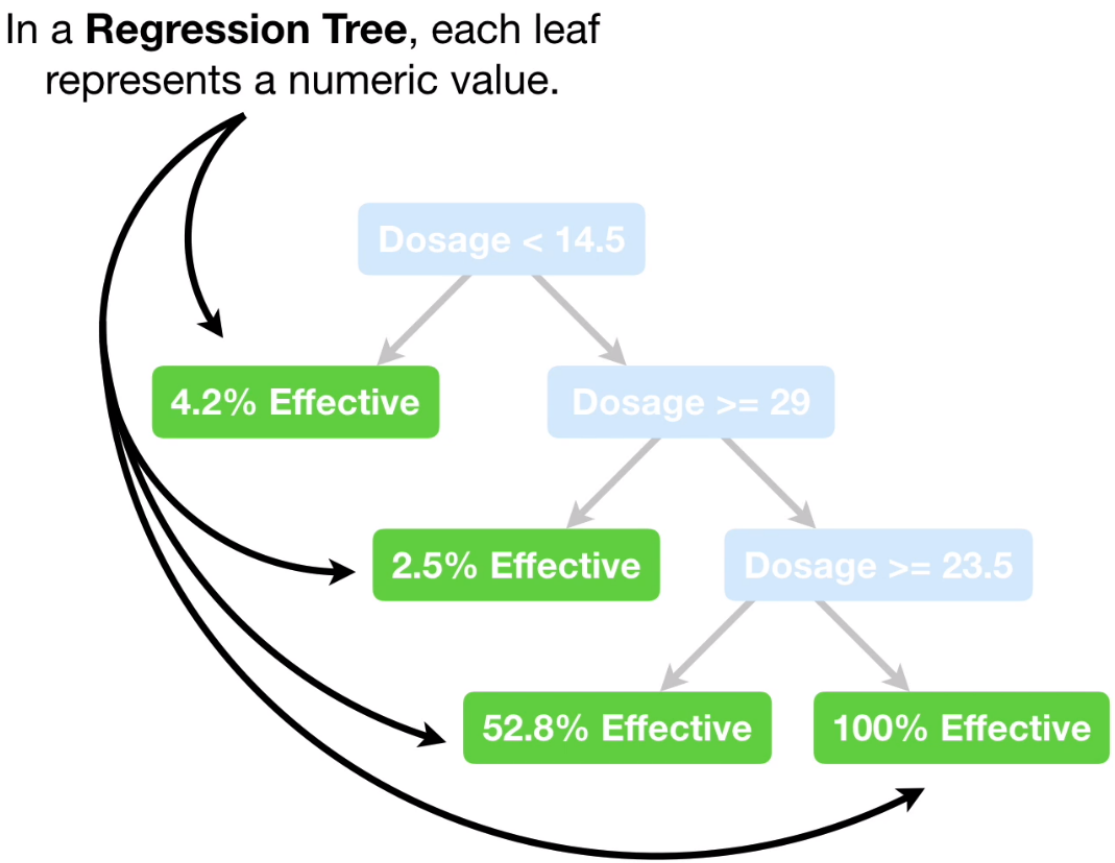
So we need to use something other than a straight line to make predictions.



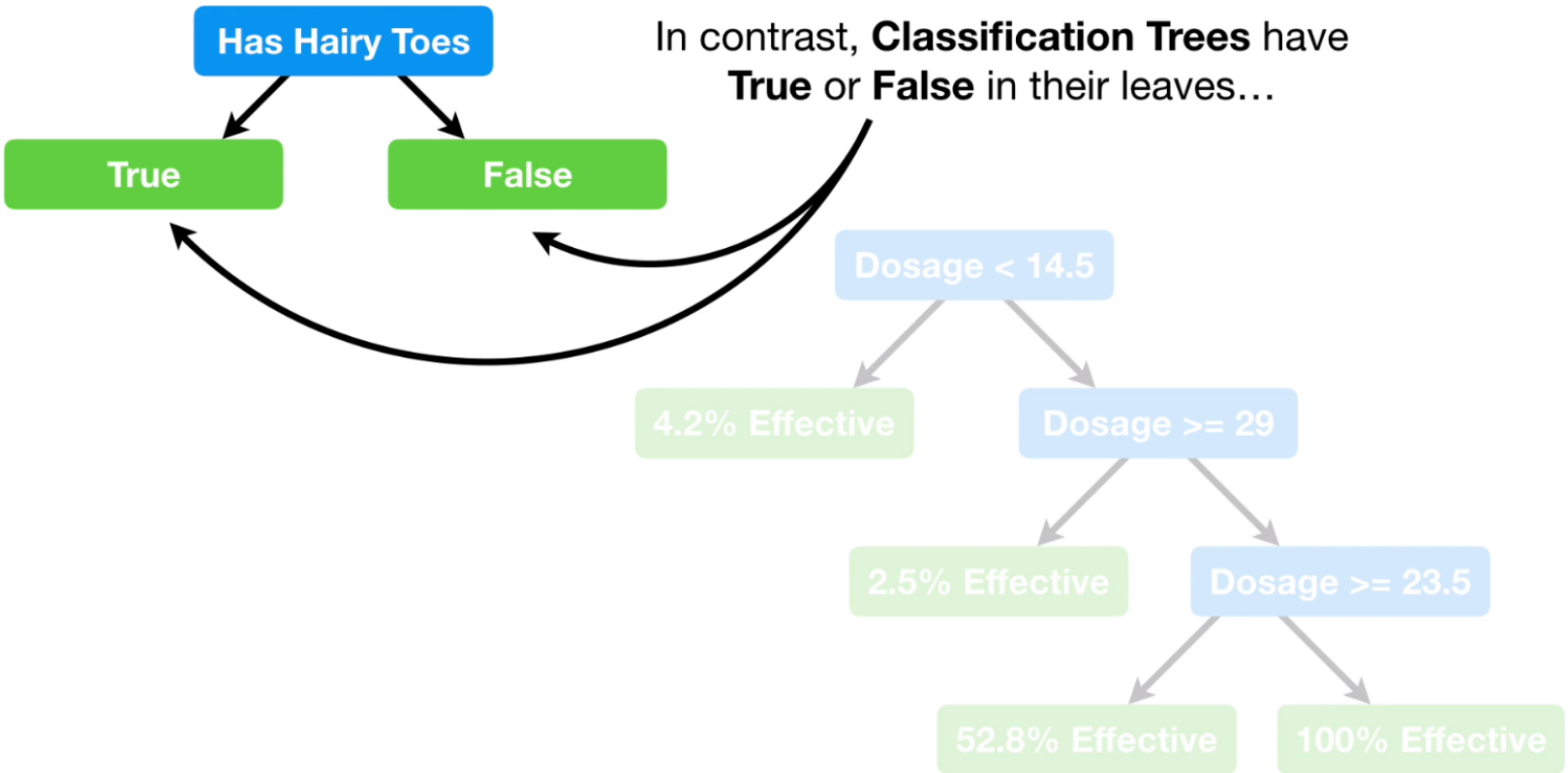
Regression Trees



Regression Trees

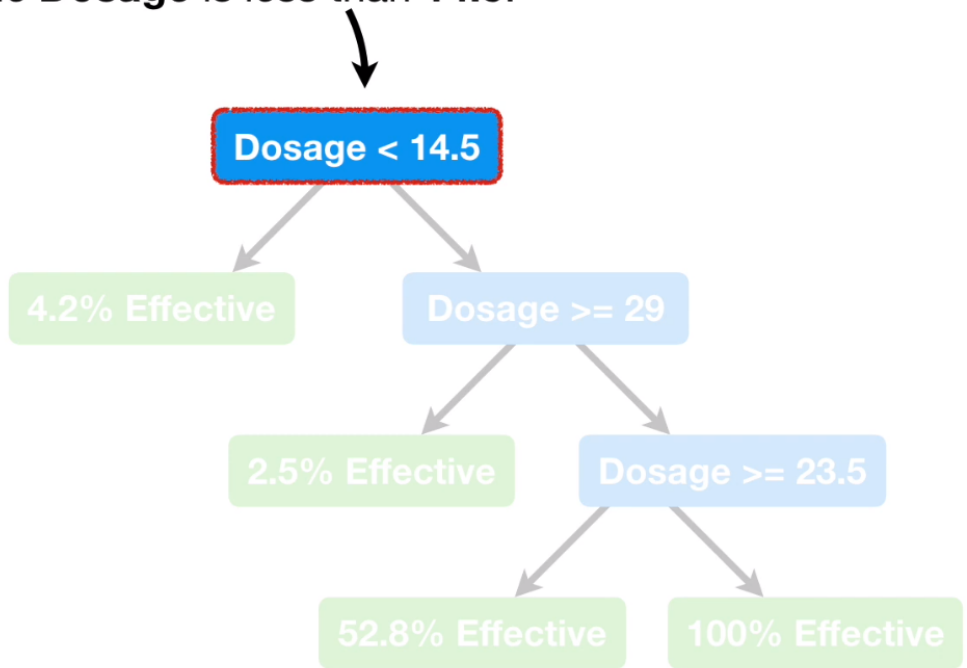


Regression Trees

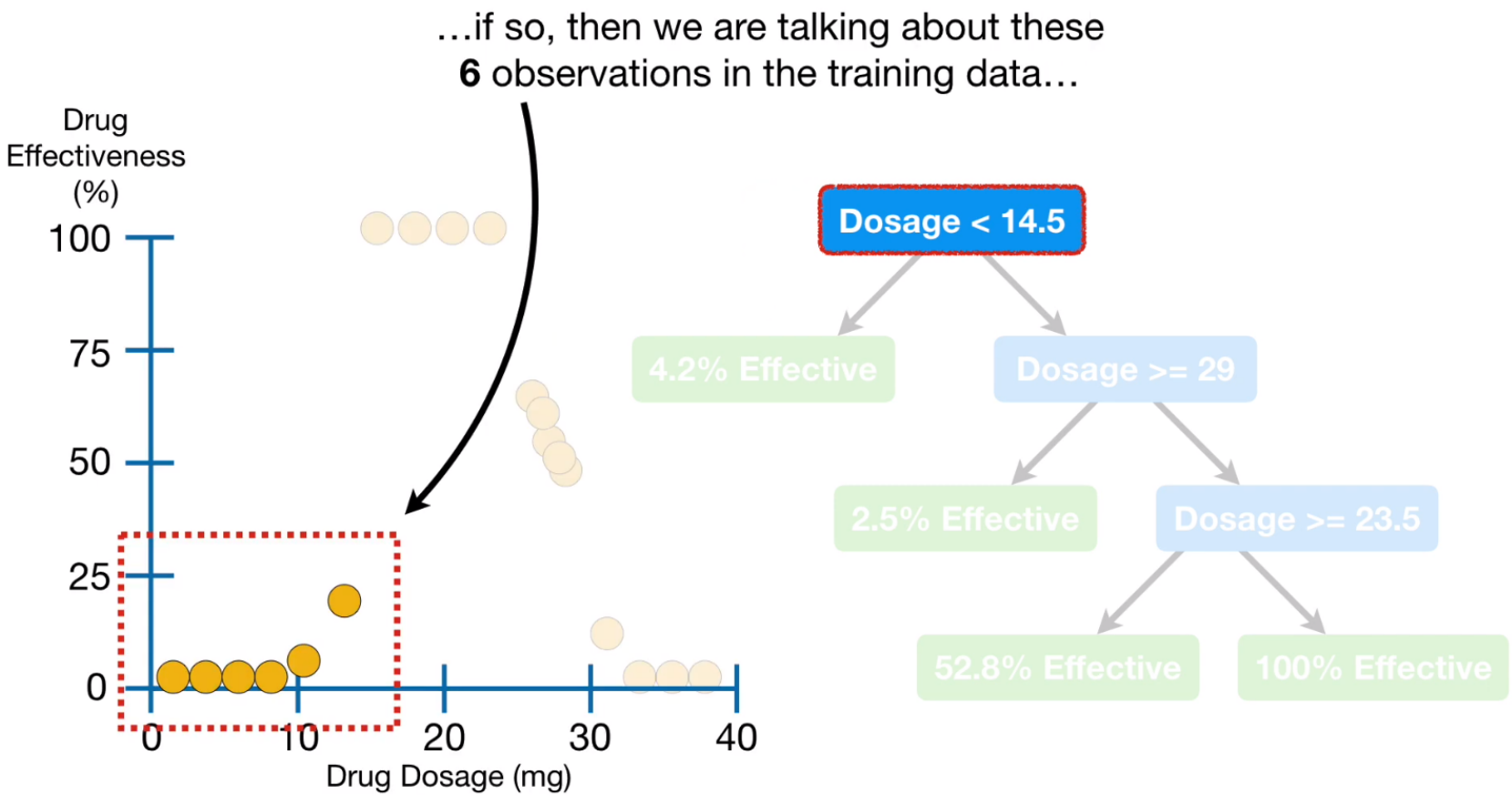


Regression Trees

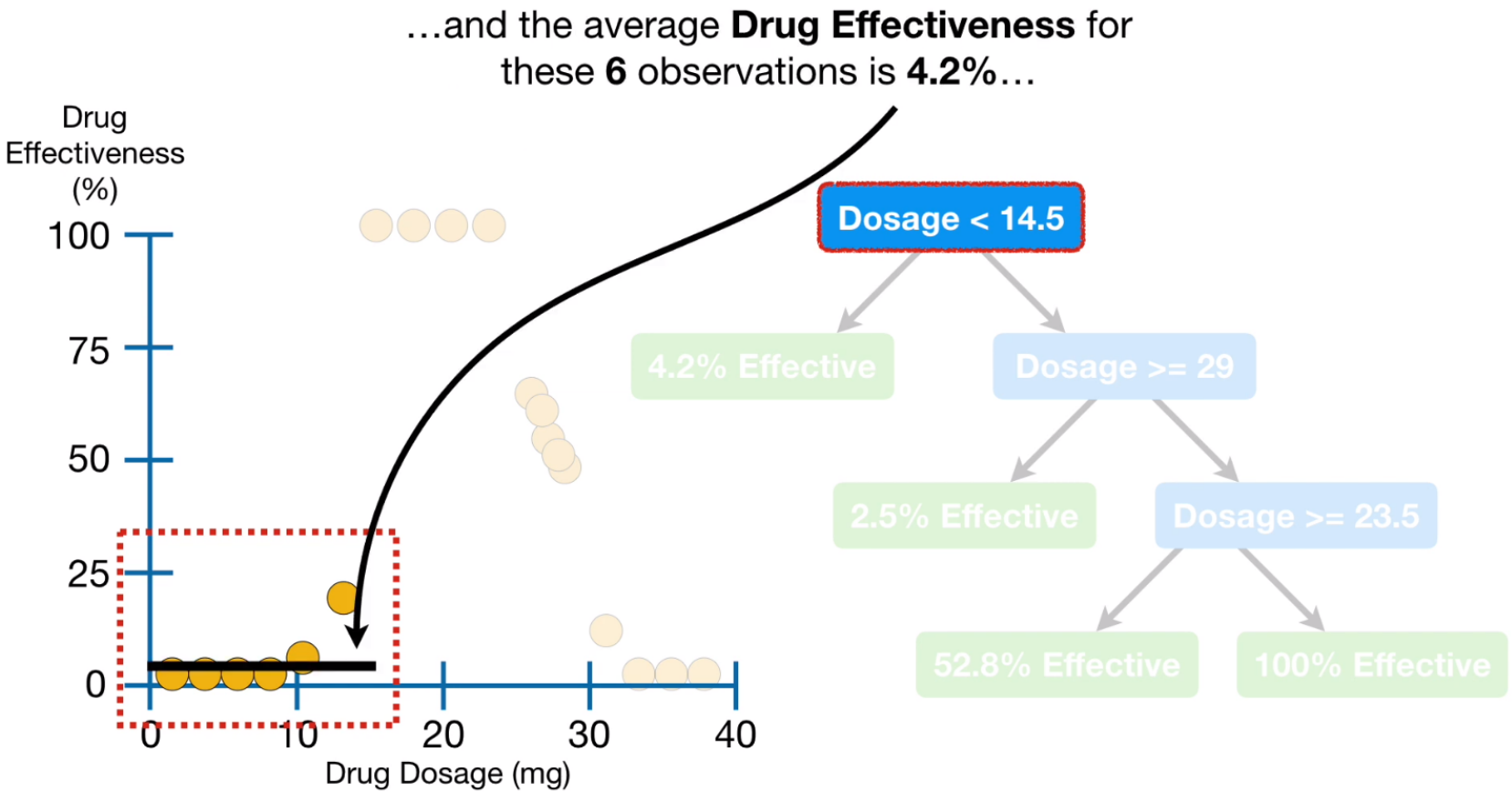
With *this* **Regression Tree**, we start by asking if the **Dosage** is less than **14.5**.



Regression Trees

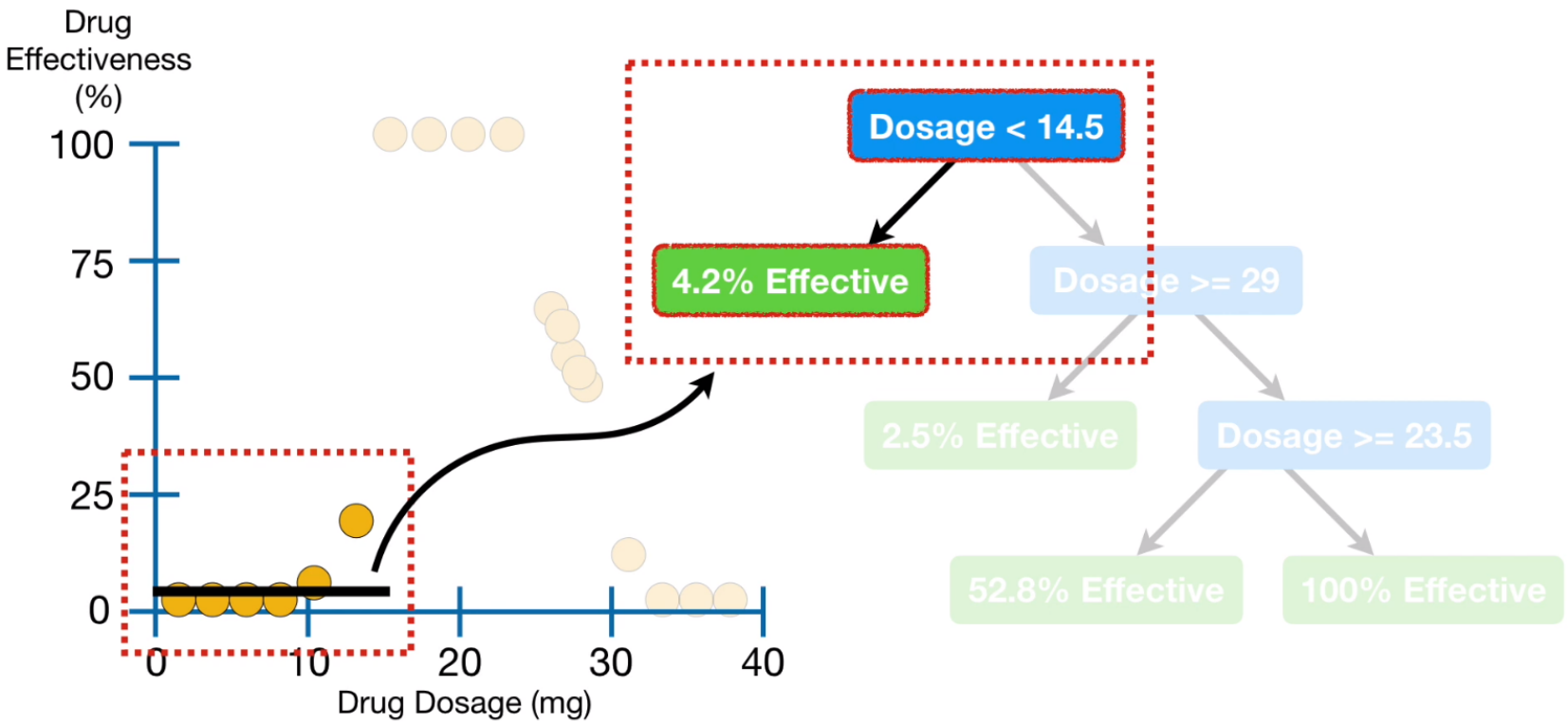


Regression Trees

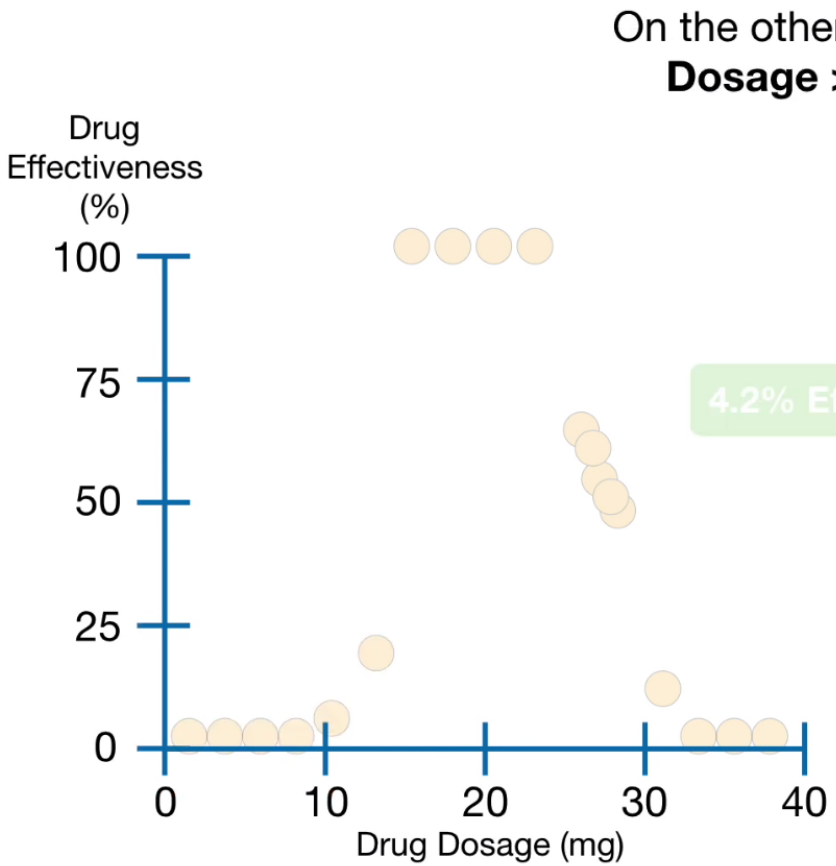


Regression Trees

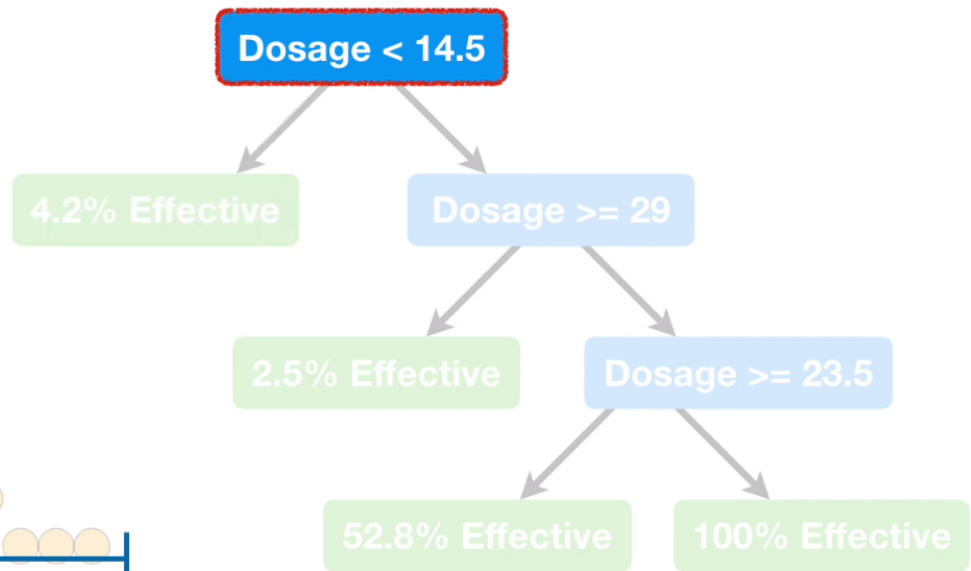
…so the tree uses the average value, **4.2%**, as its prediction for people with **Dosages < 14.5**.



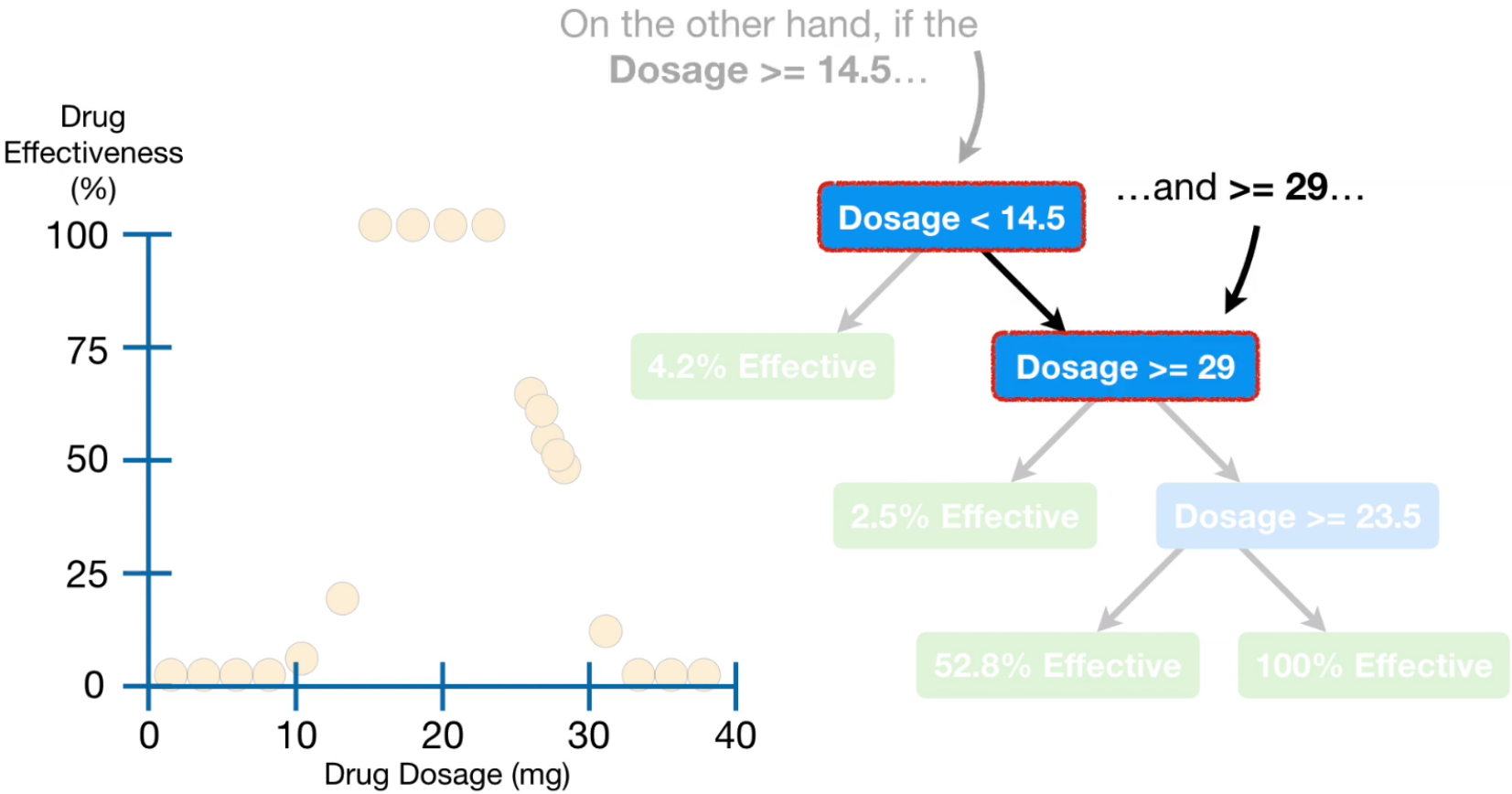
Regression Trees



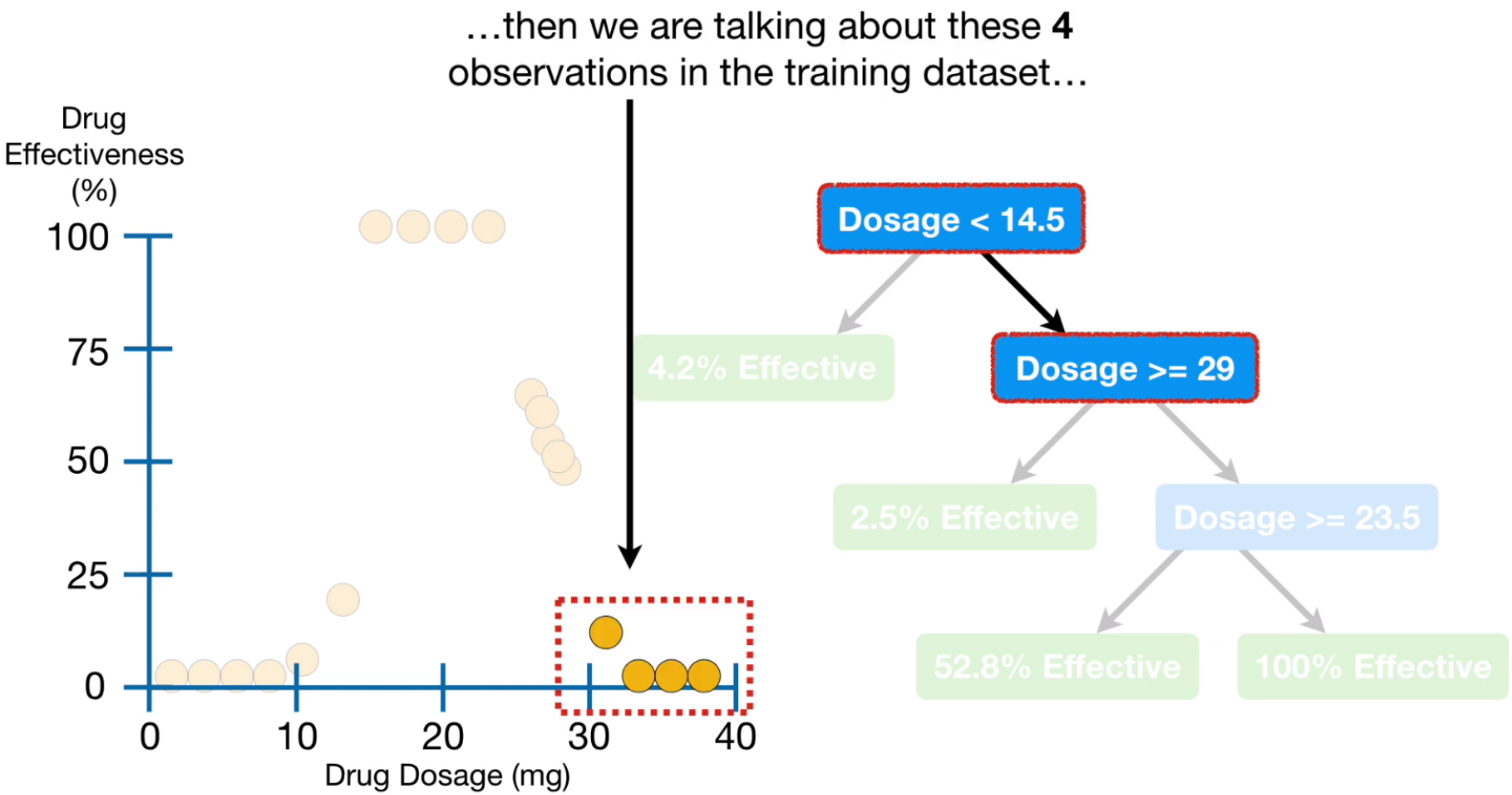
On the other hand, if the **Dosage >= 14.5**...



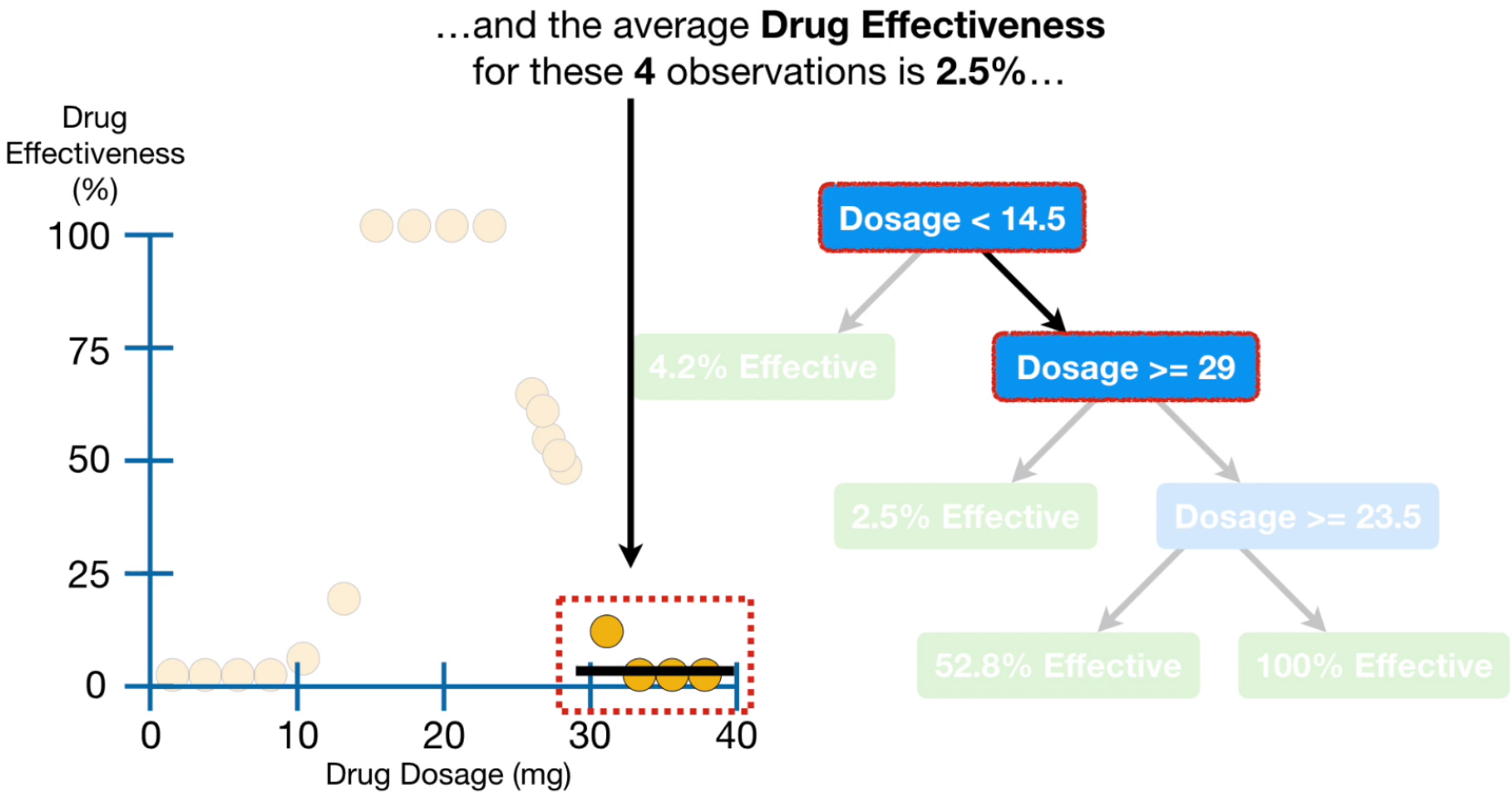
Regression Trees



Regression Trees

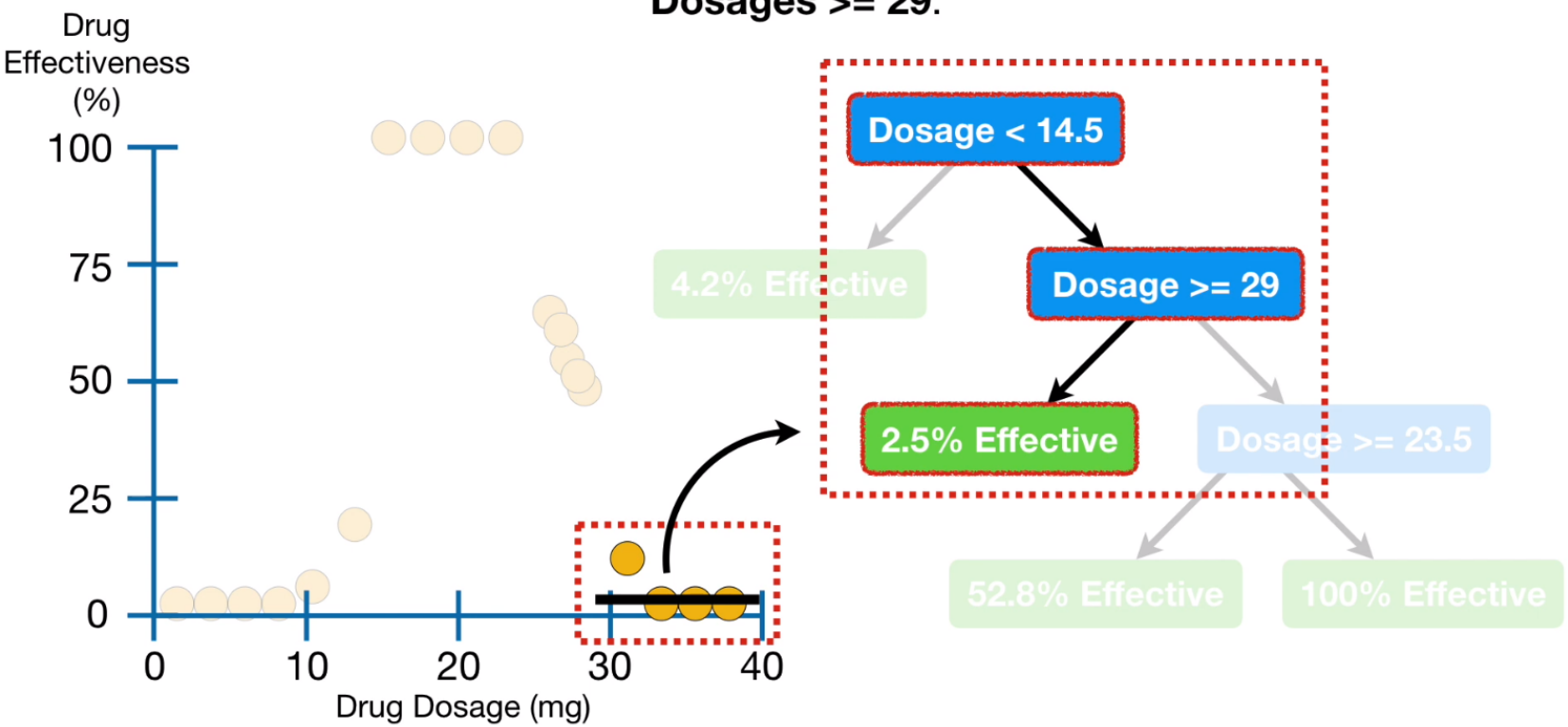


Regression Trees

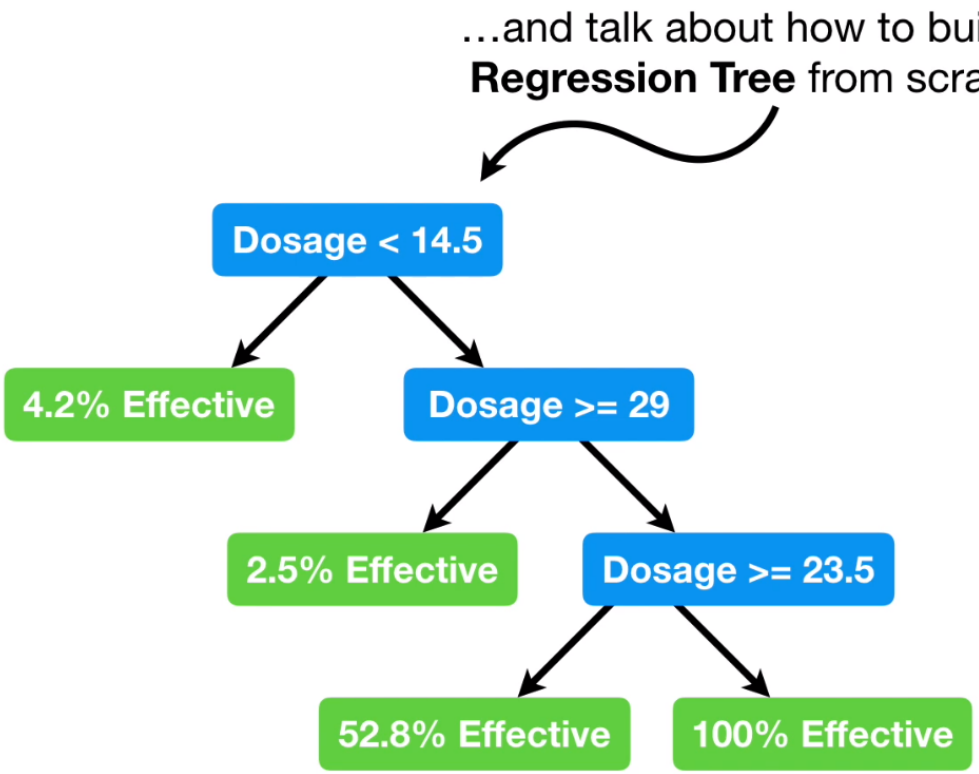


Regression Trees

…so the tree uses the average value, **2.5%**, as its prediction for people with **Dosages >= 29**.

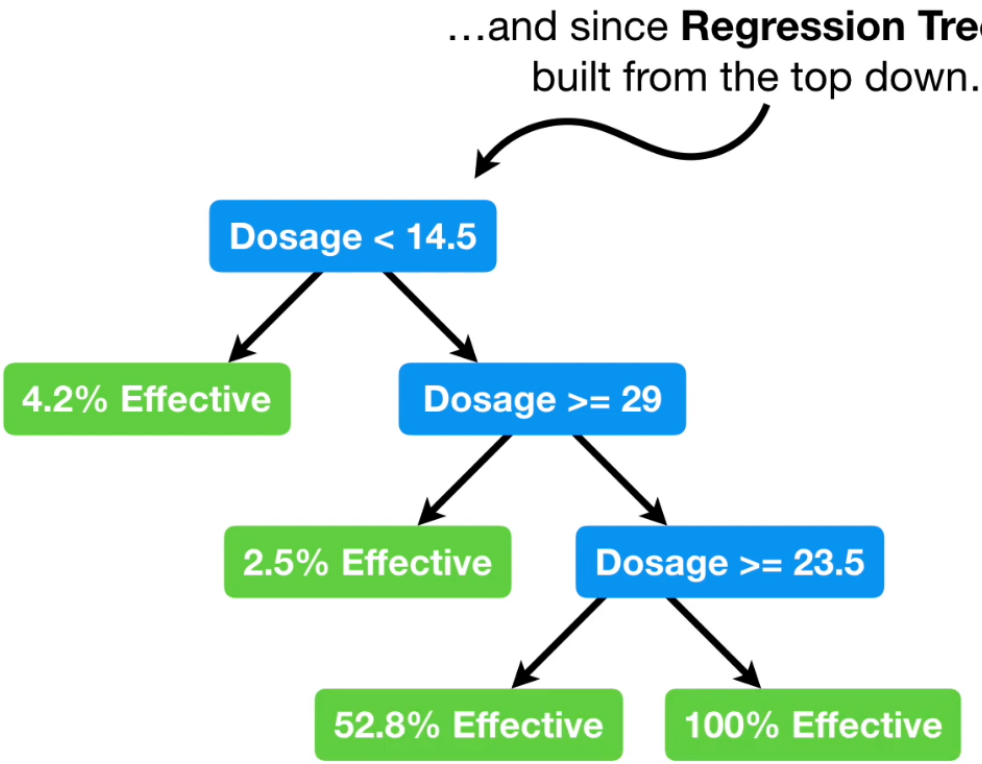


Regression Trees



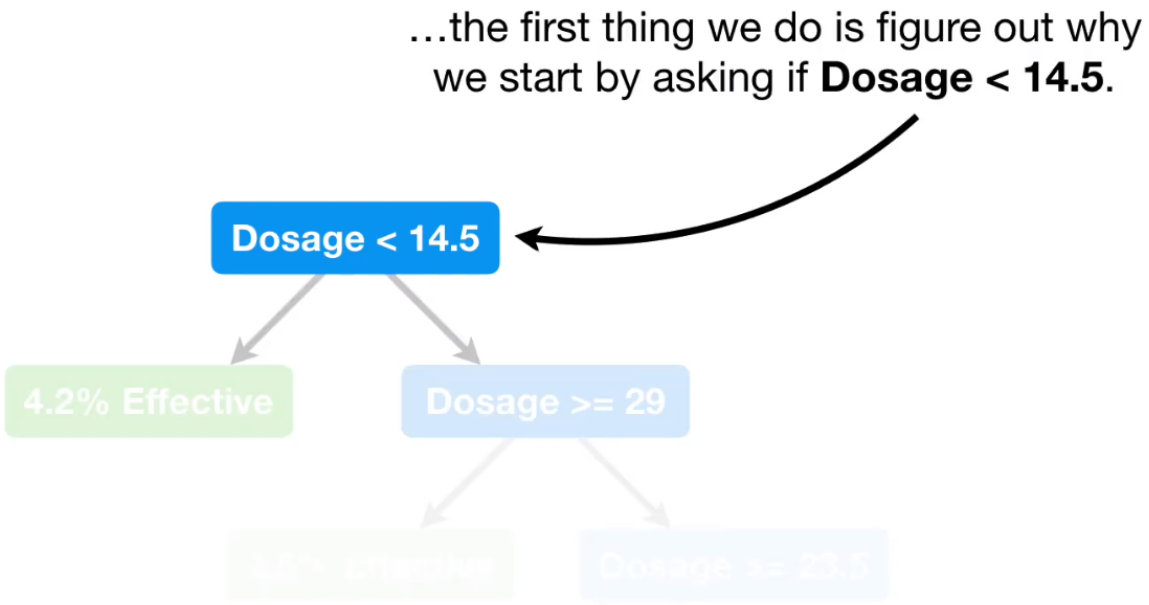
Dosage	Drug Effect.
10	58
20	60
35	57
5	44
etc...	etc...

Regression Trees



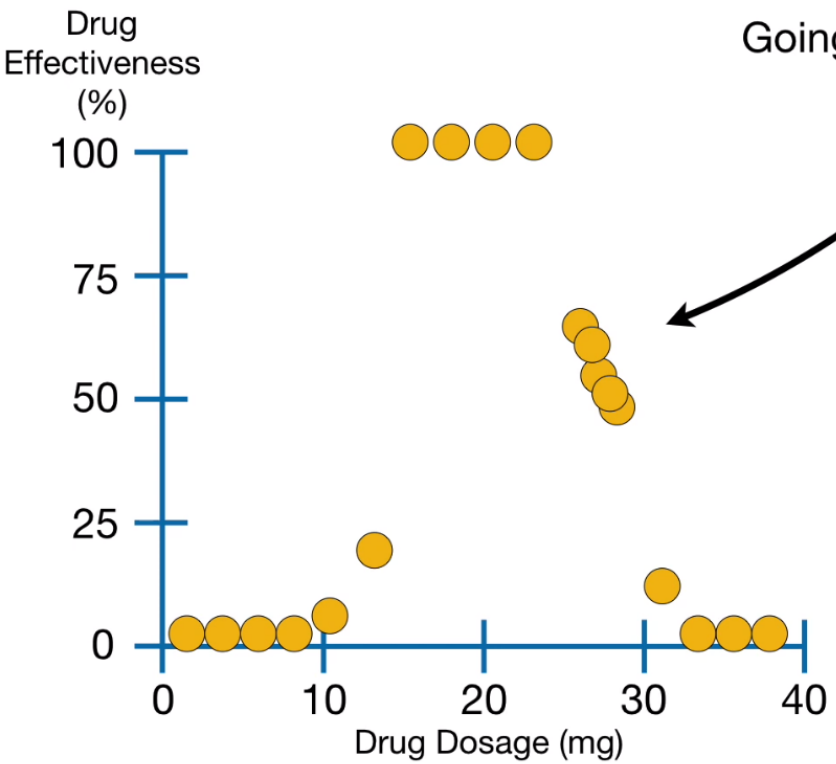
Dosage	Drug Effect.
10	58
20	60
35	57
5	44
etc...	etc...

Regression Trees



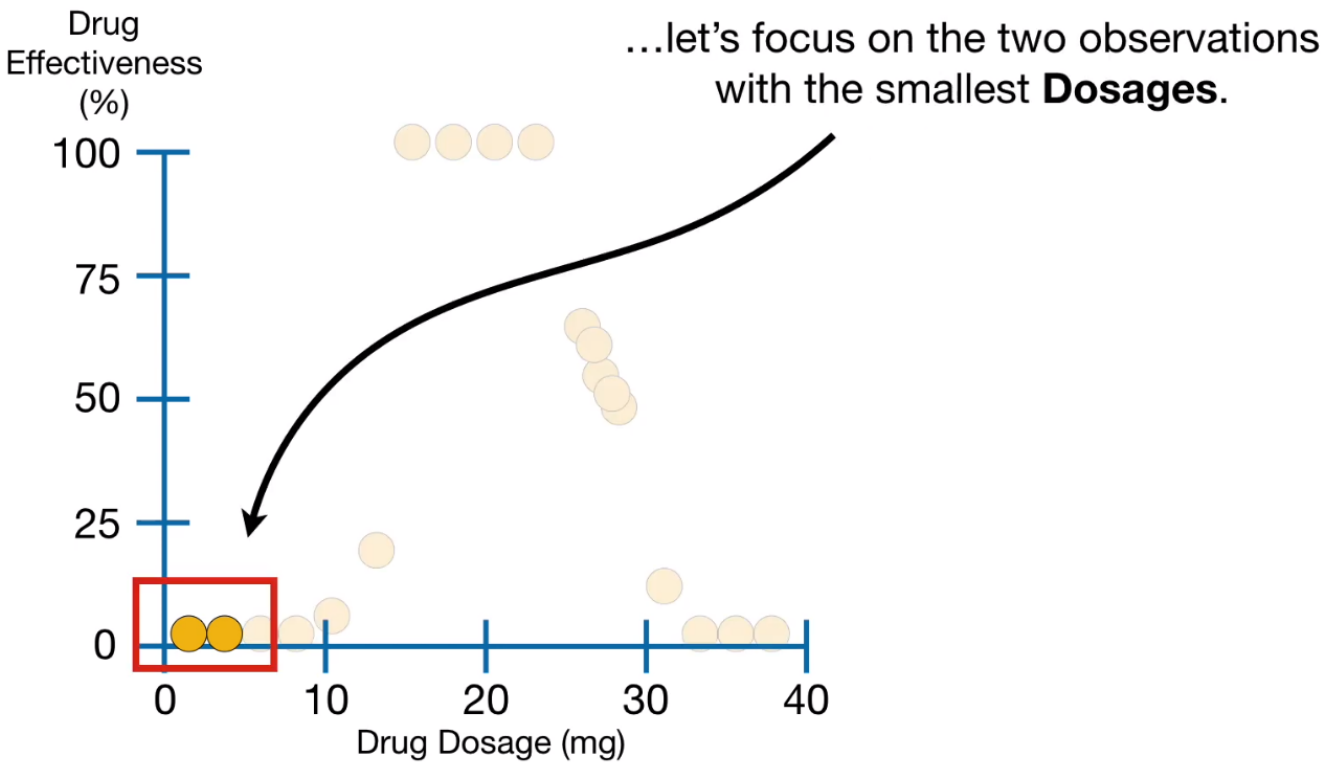
Dosage	Drug Effect.
10	58
20	60
35	57
5	44
etc...	etc...

Regression Trees

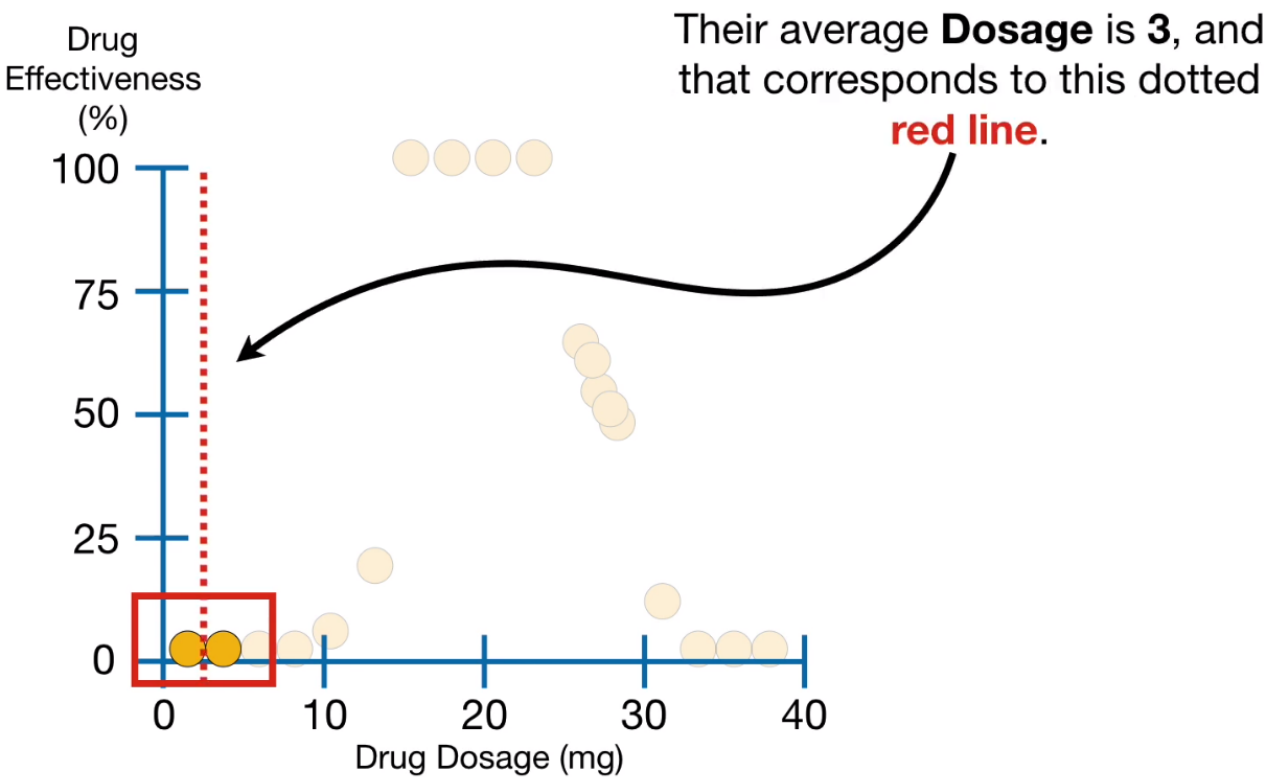


Dosage	Drug Effect.
10	58
20	60
35	57
5	44
etc…	etc…

Regression Trees

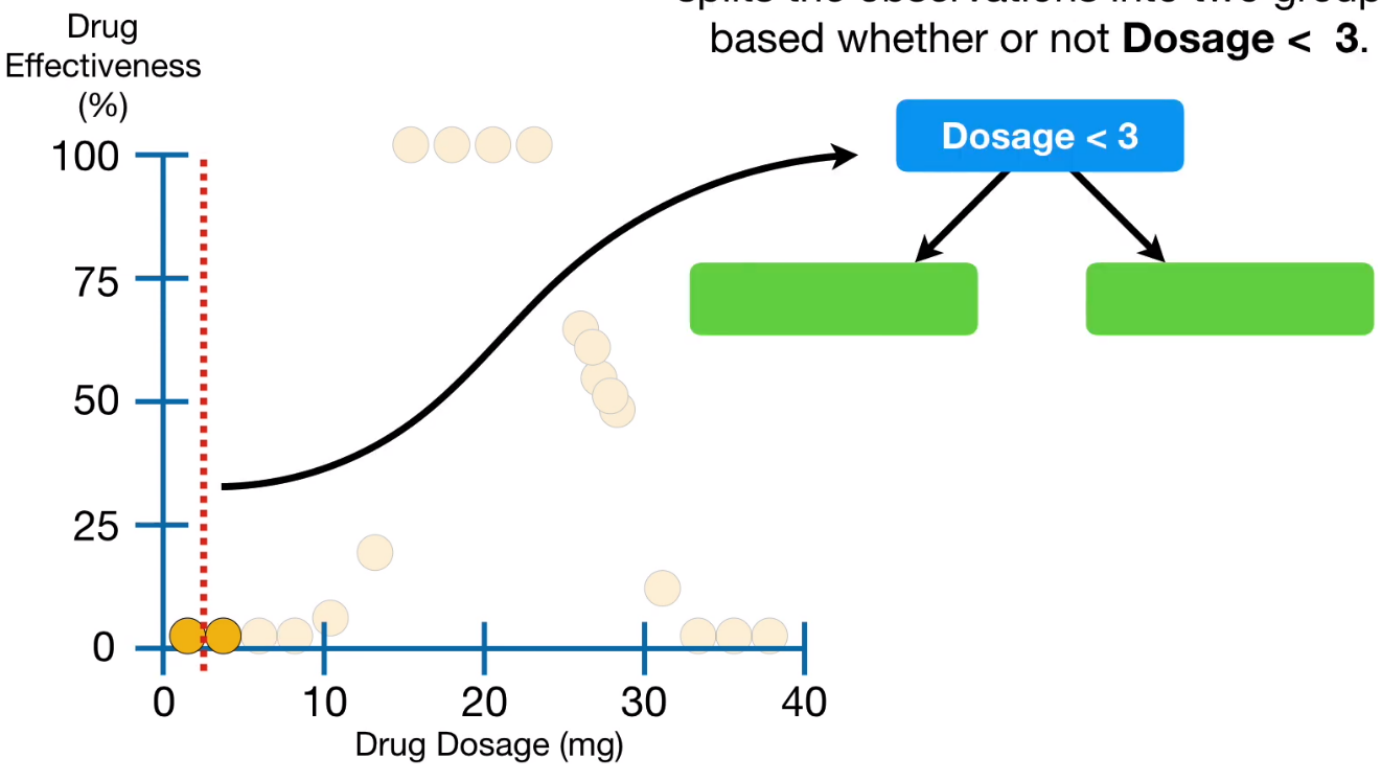


Regression Trees

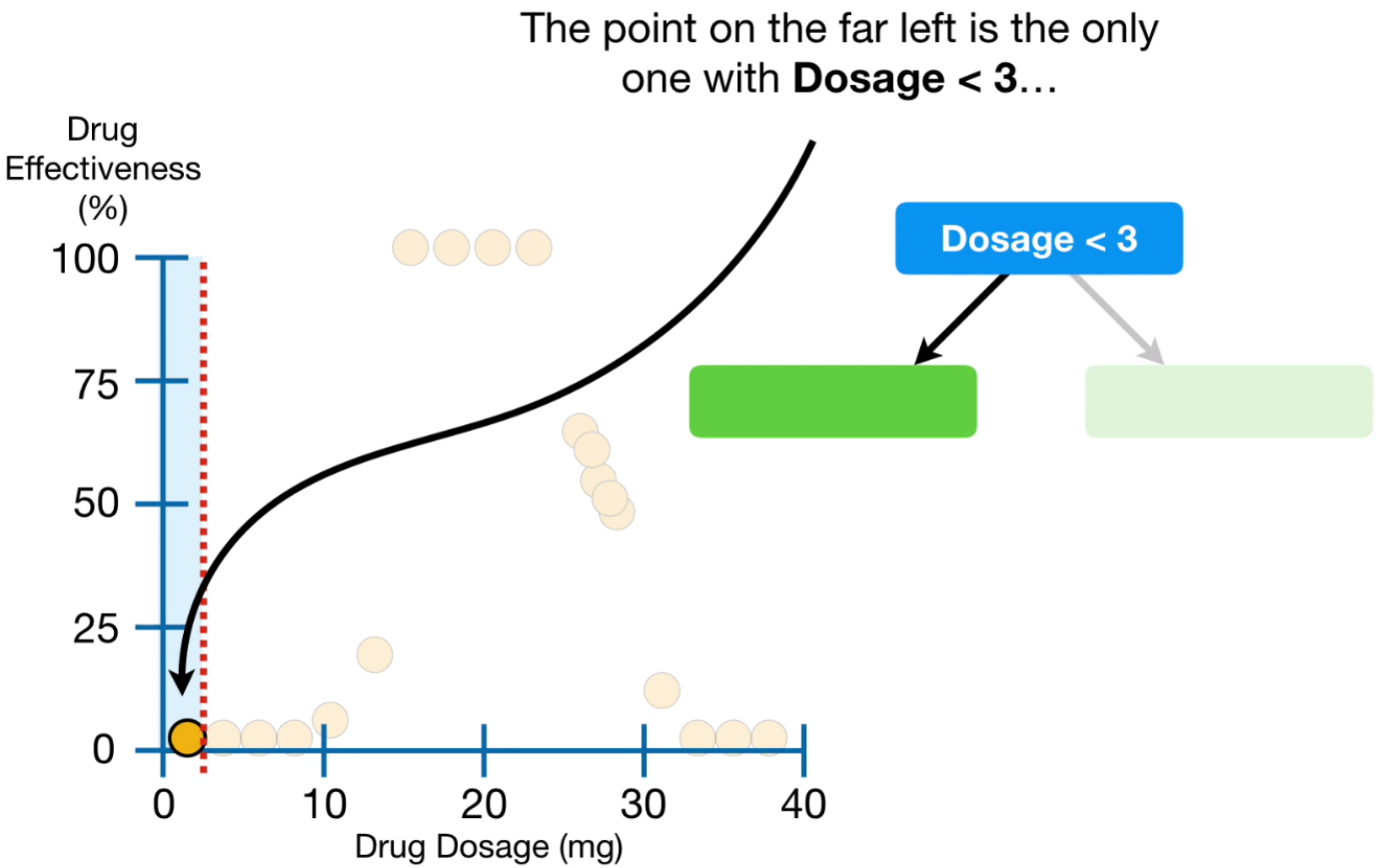


Regression Trees

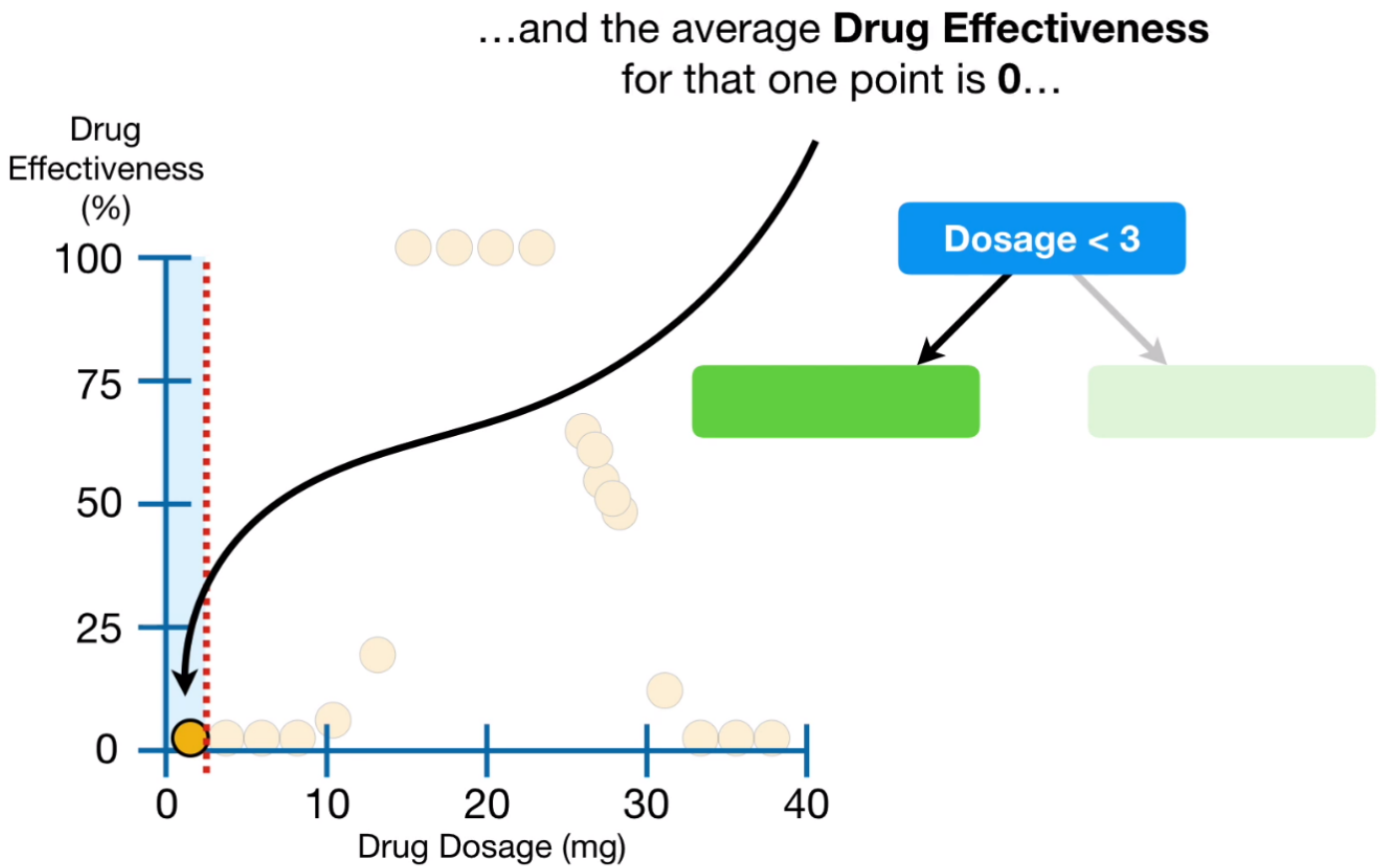
Now we can build a very simple tree that splits the observations into two groups based whether or not **Dosage < 3**.



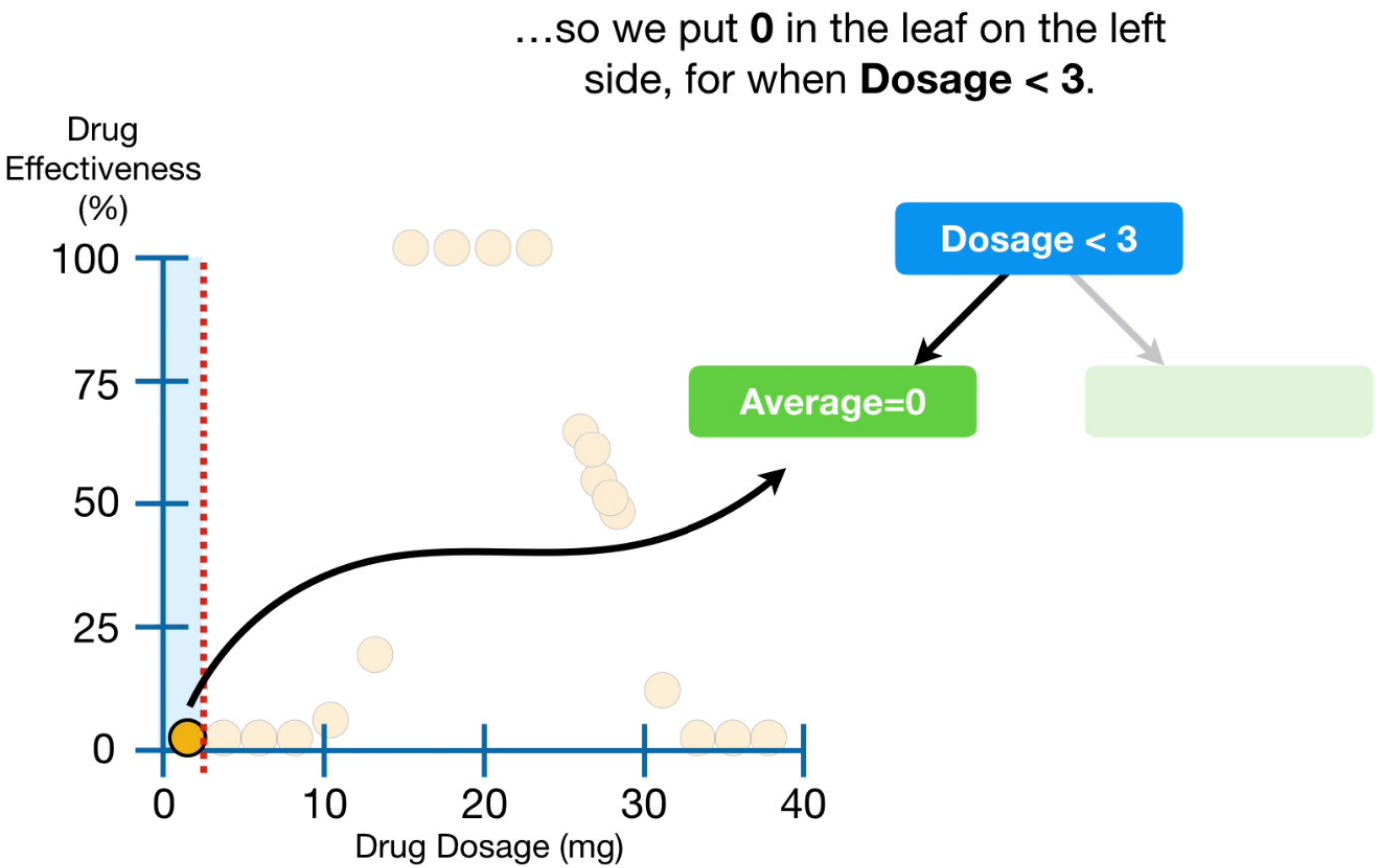
Regression Trees



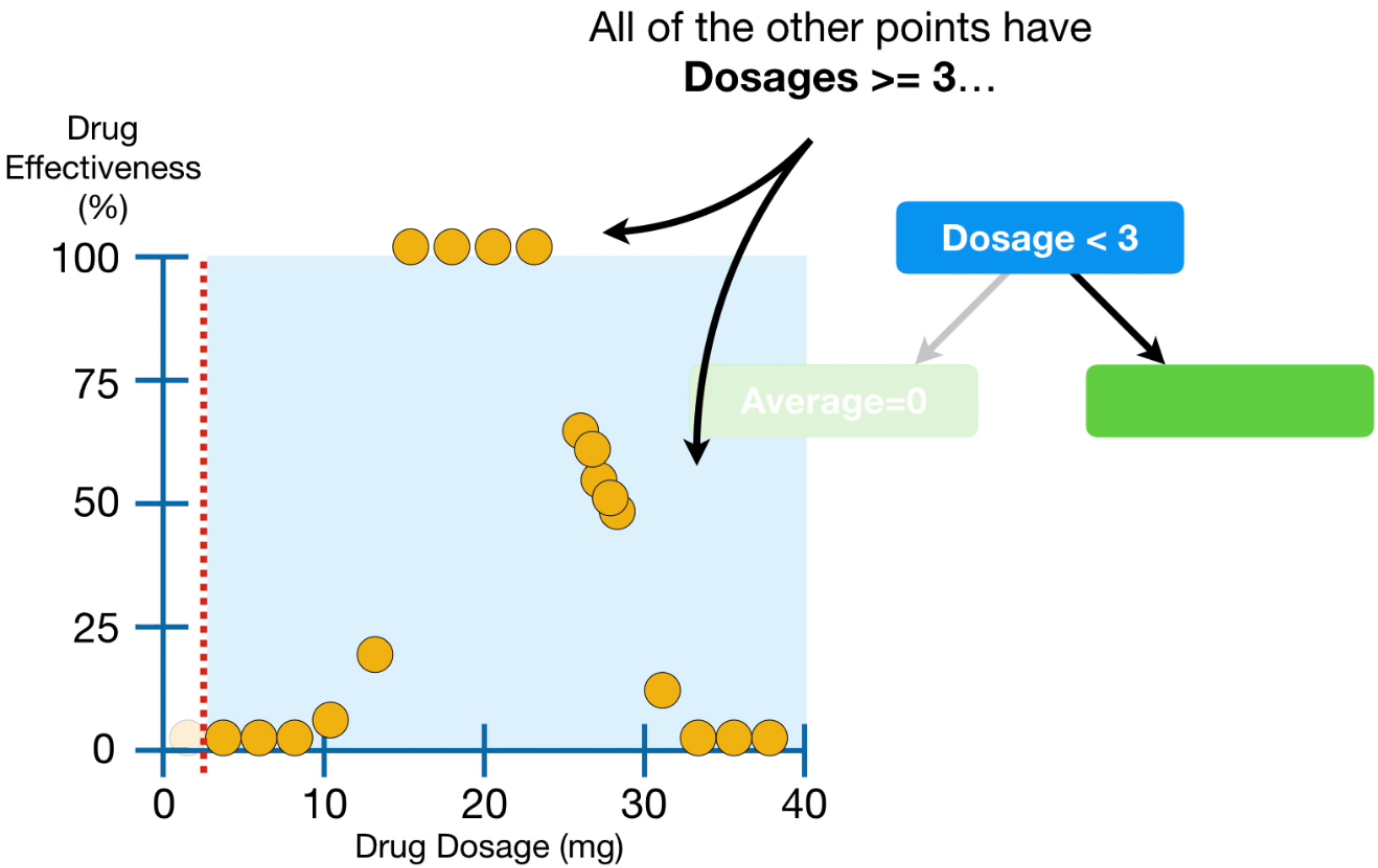
Regression Trees



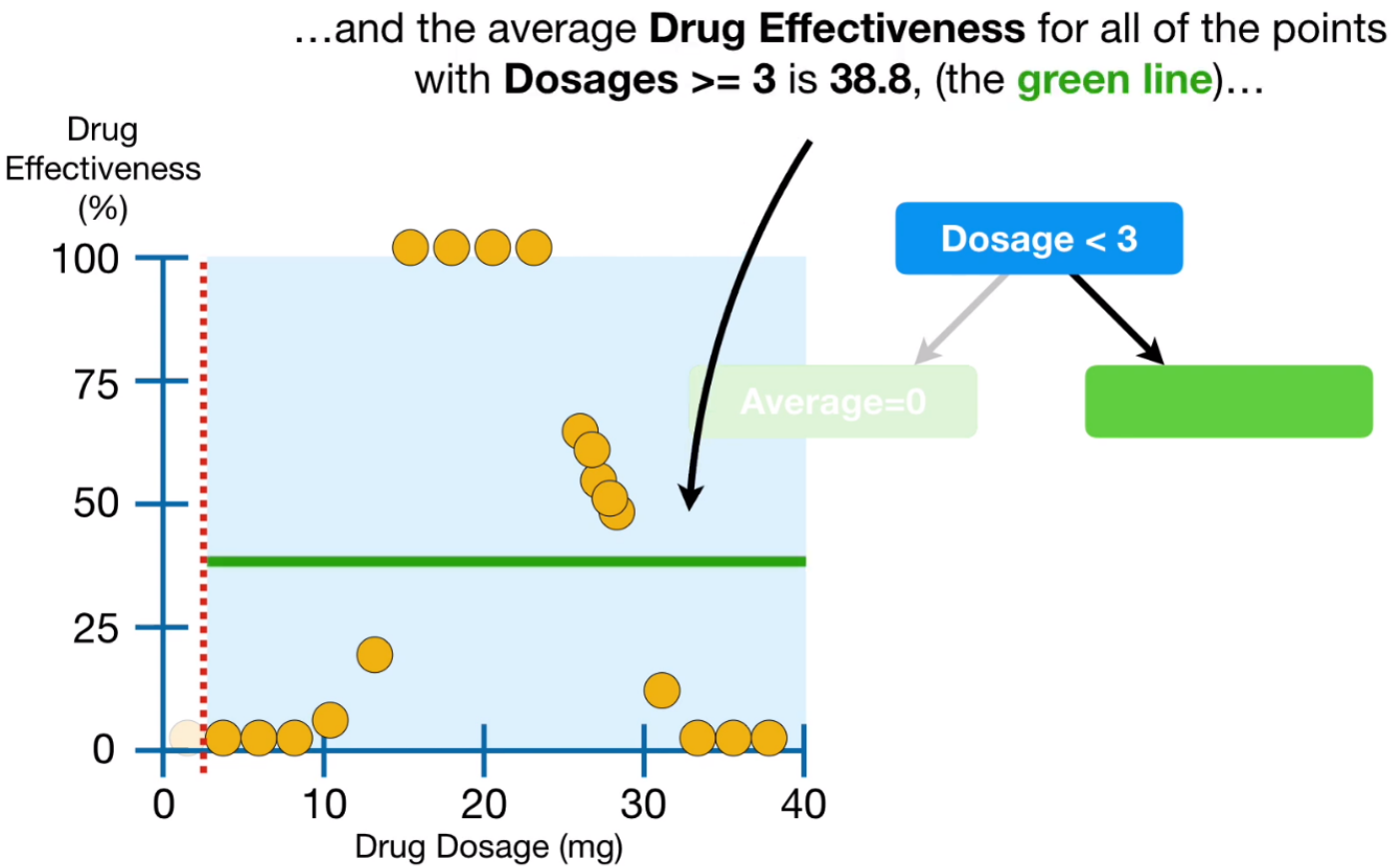
Regression Trees



Regression Trees

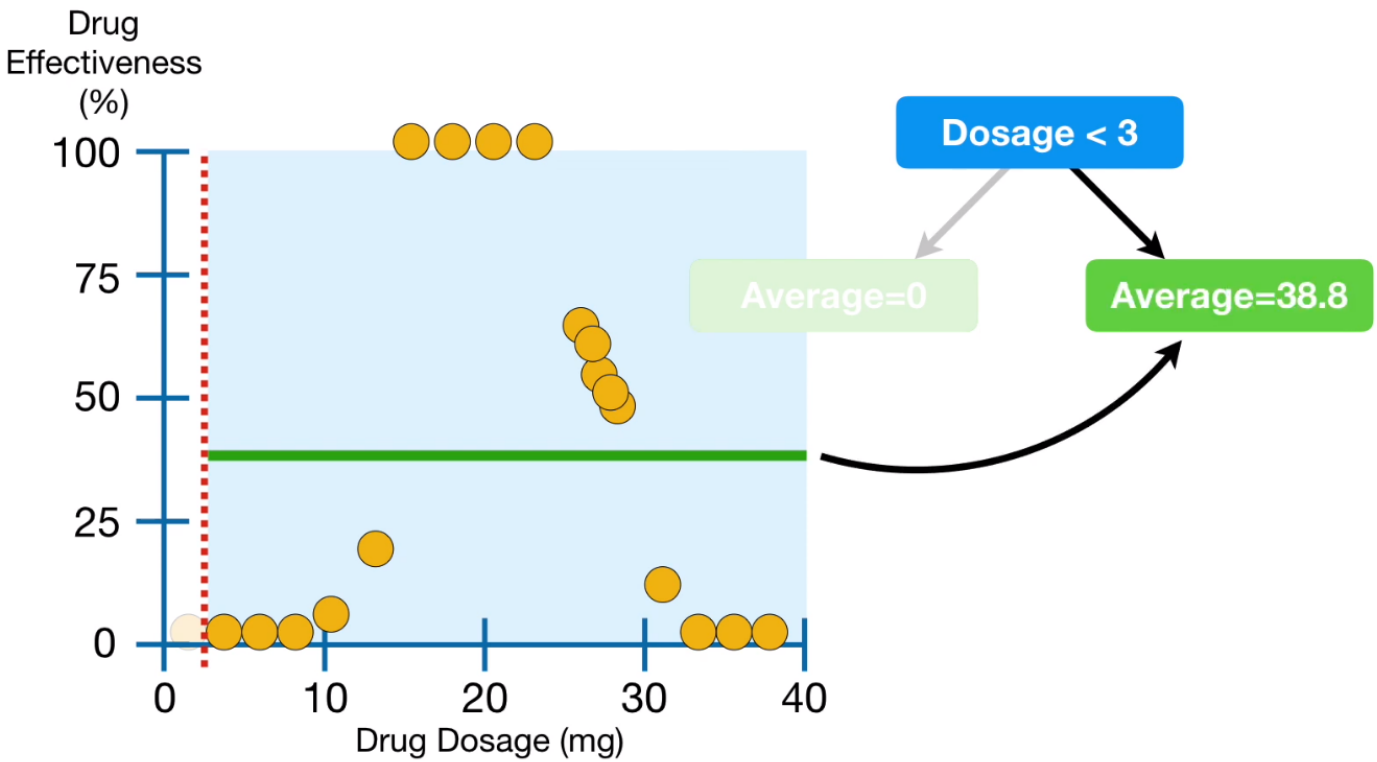


Regression Trees

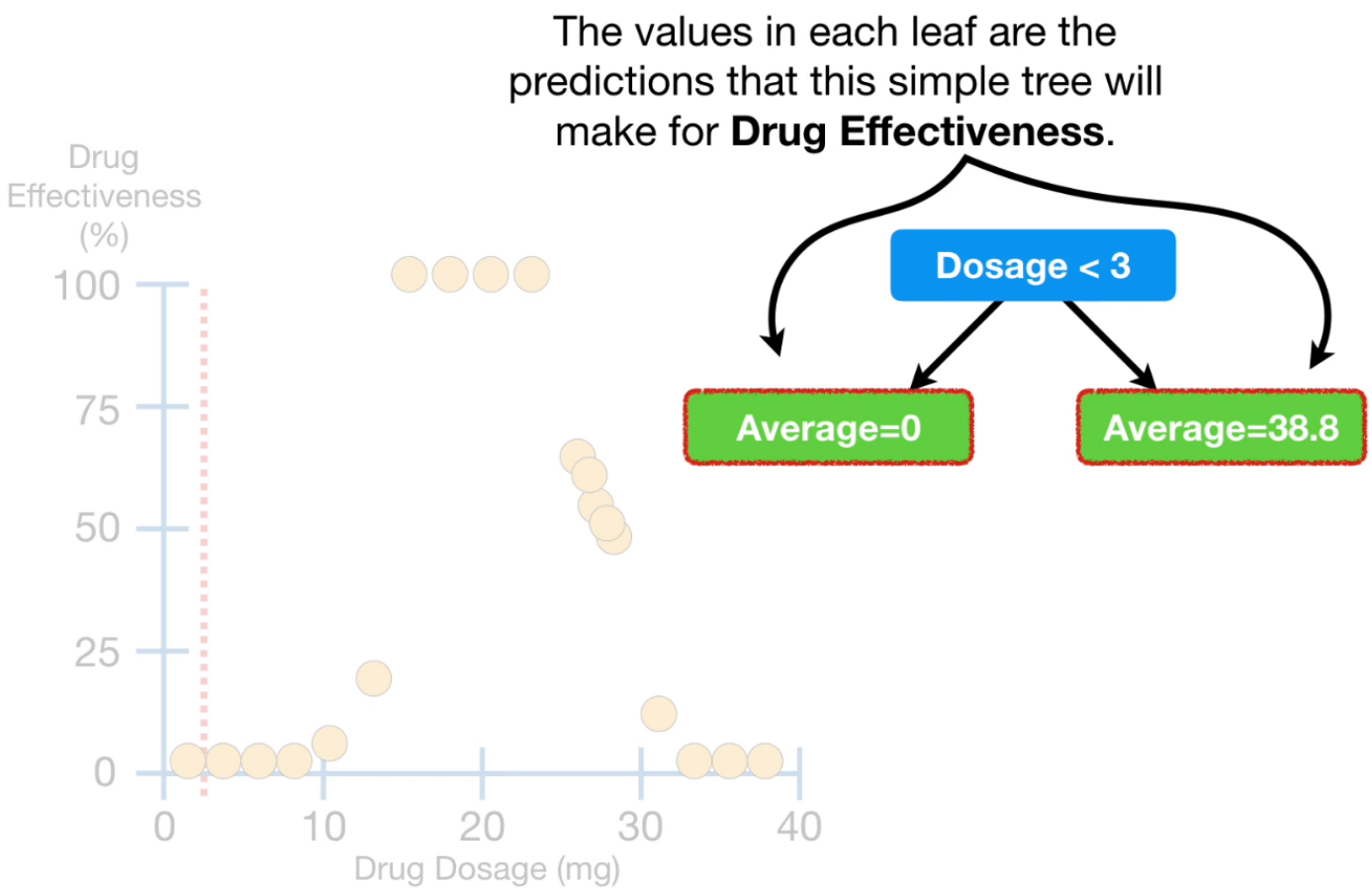


Regression Trees

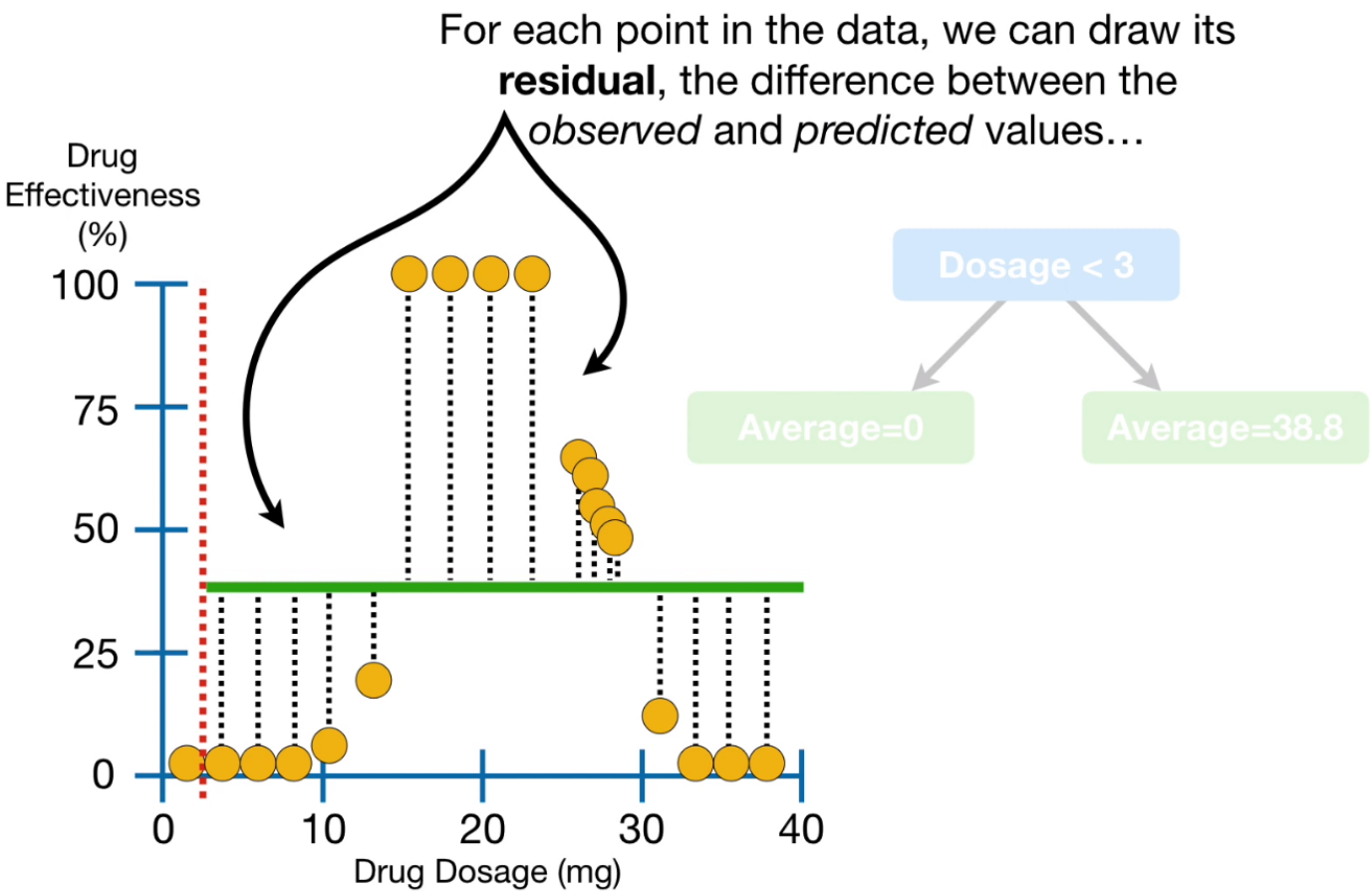
…so we put **38.8** in the leaf on the right side, for when the **Dosage >= 3**.



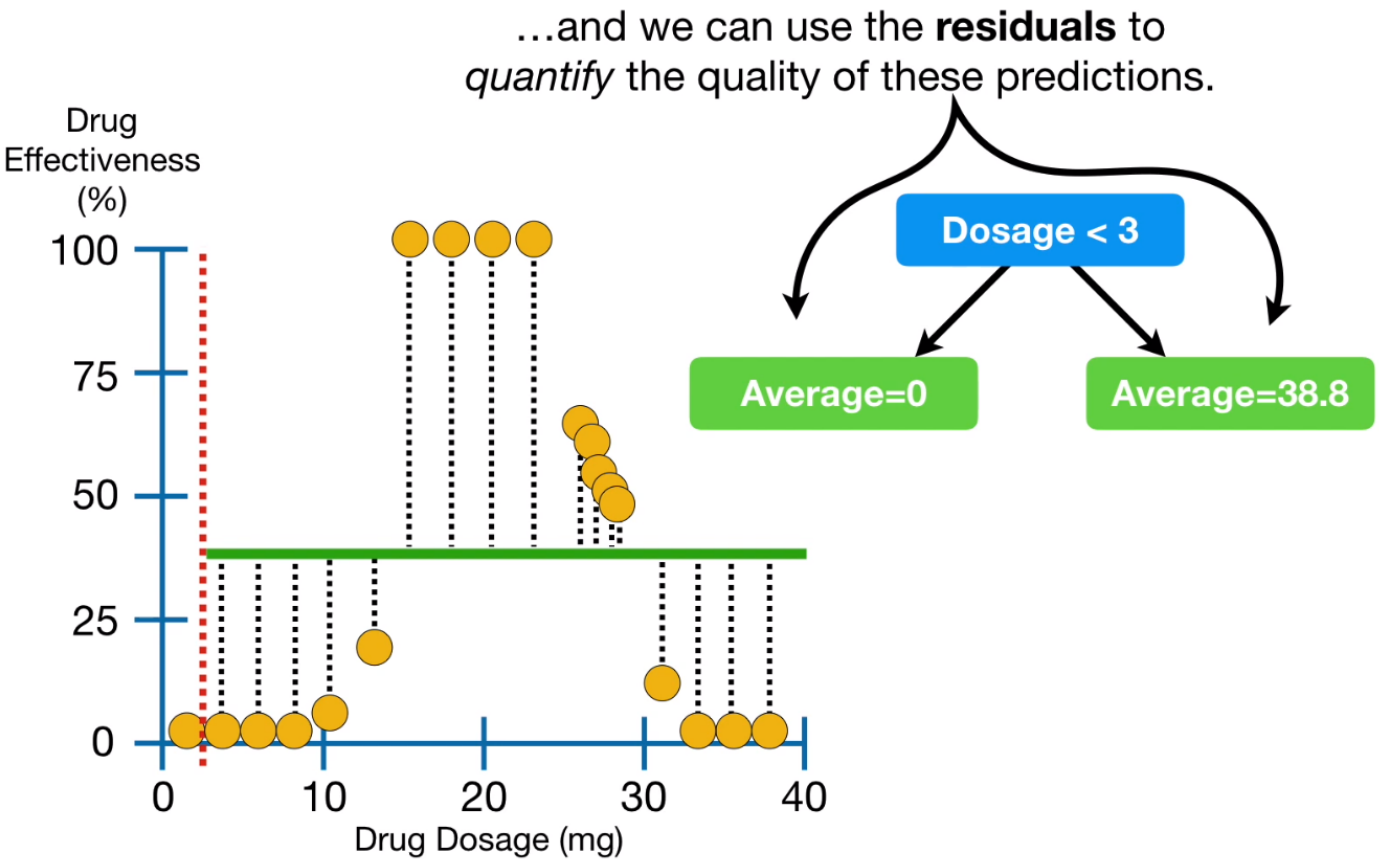
Regression Trees



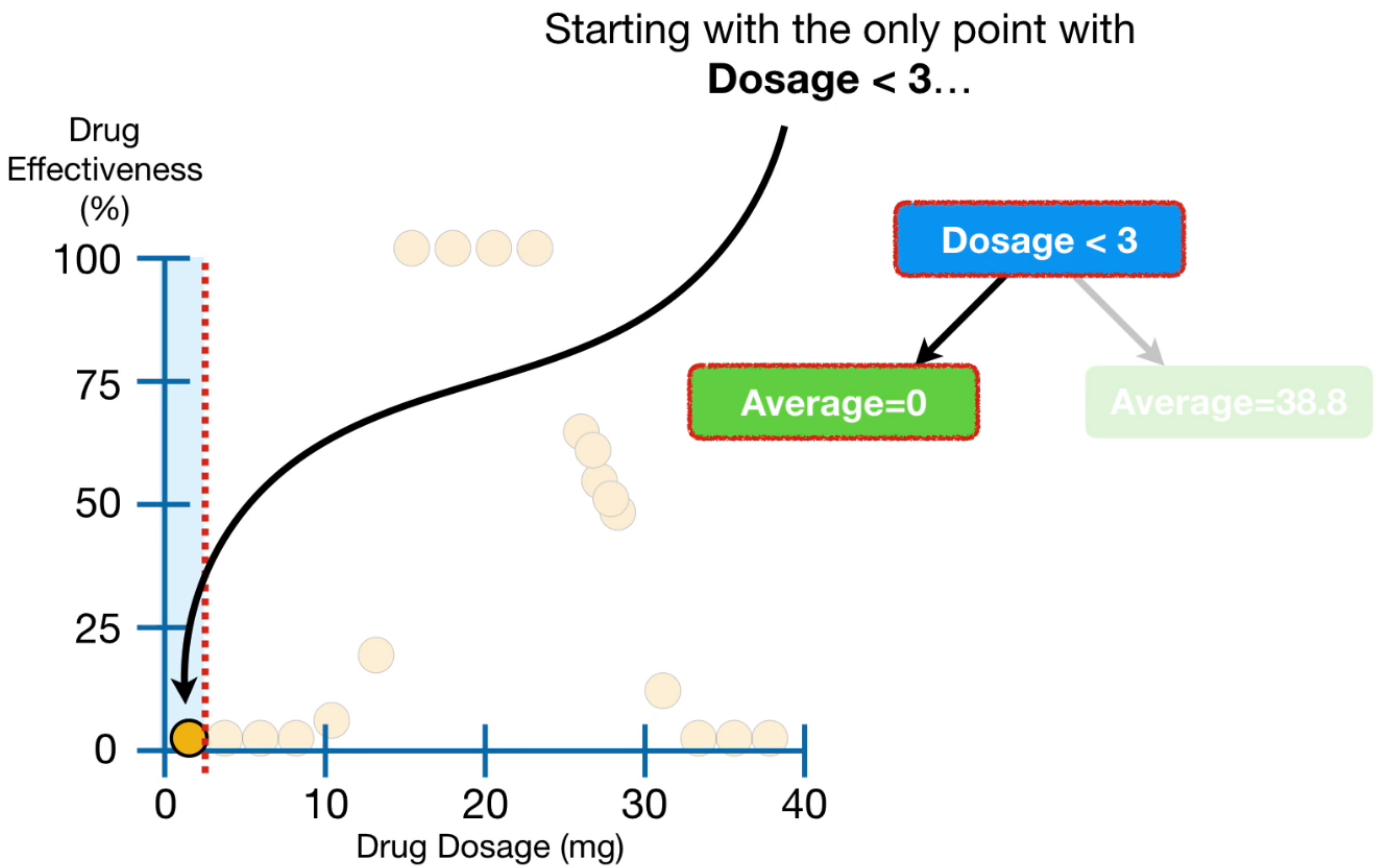
Regression Trees



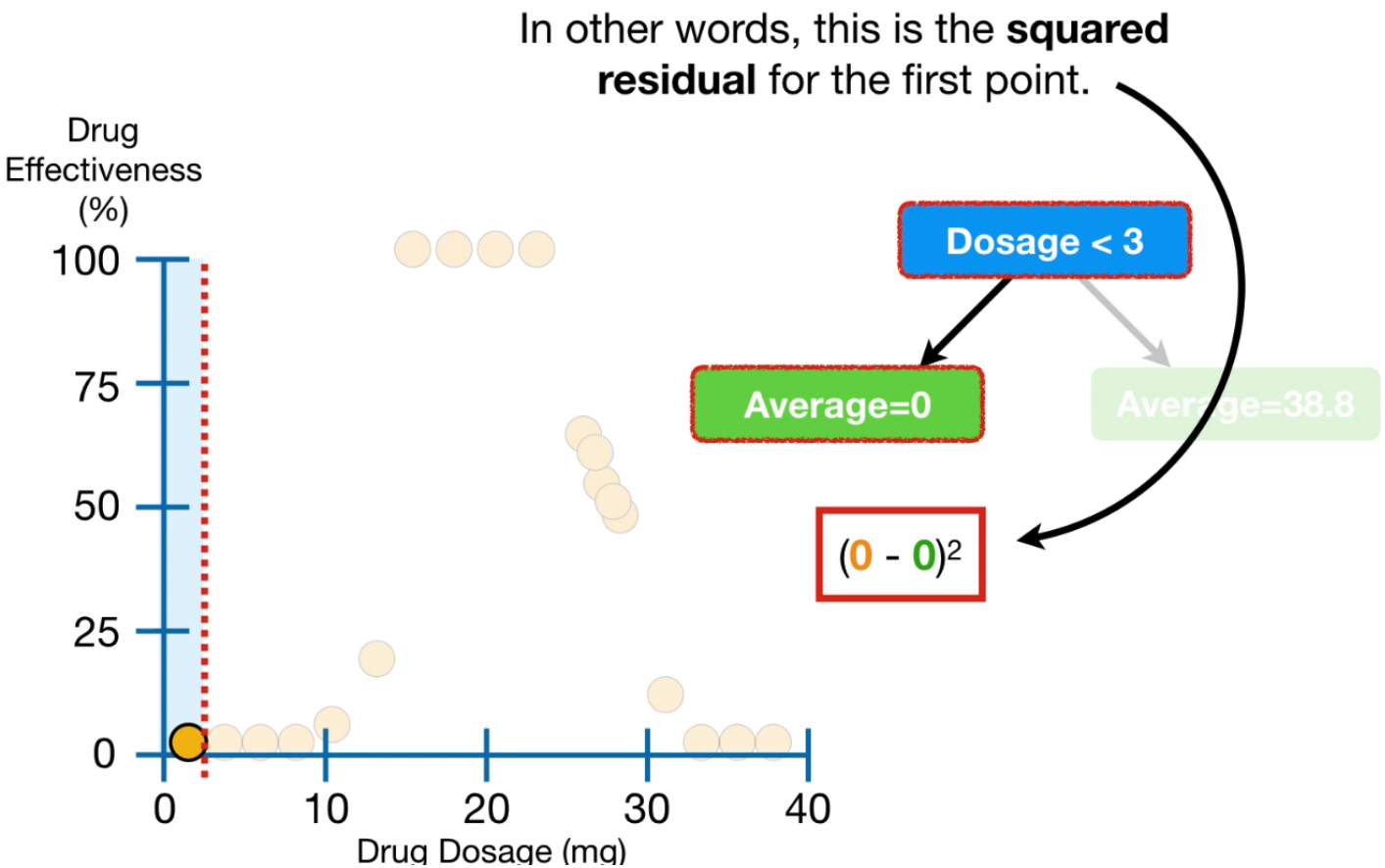
Regression Trees



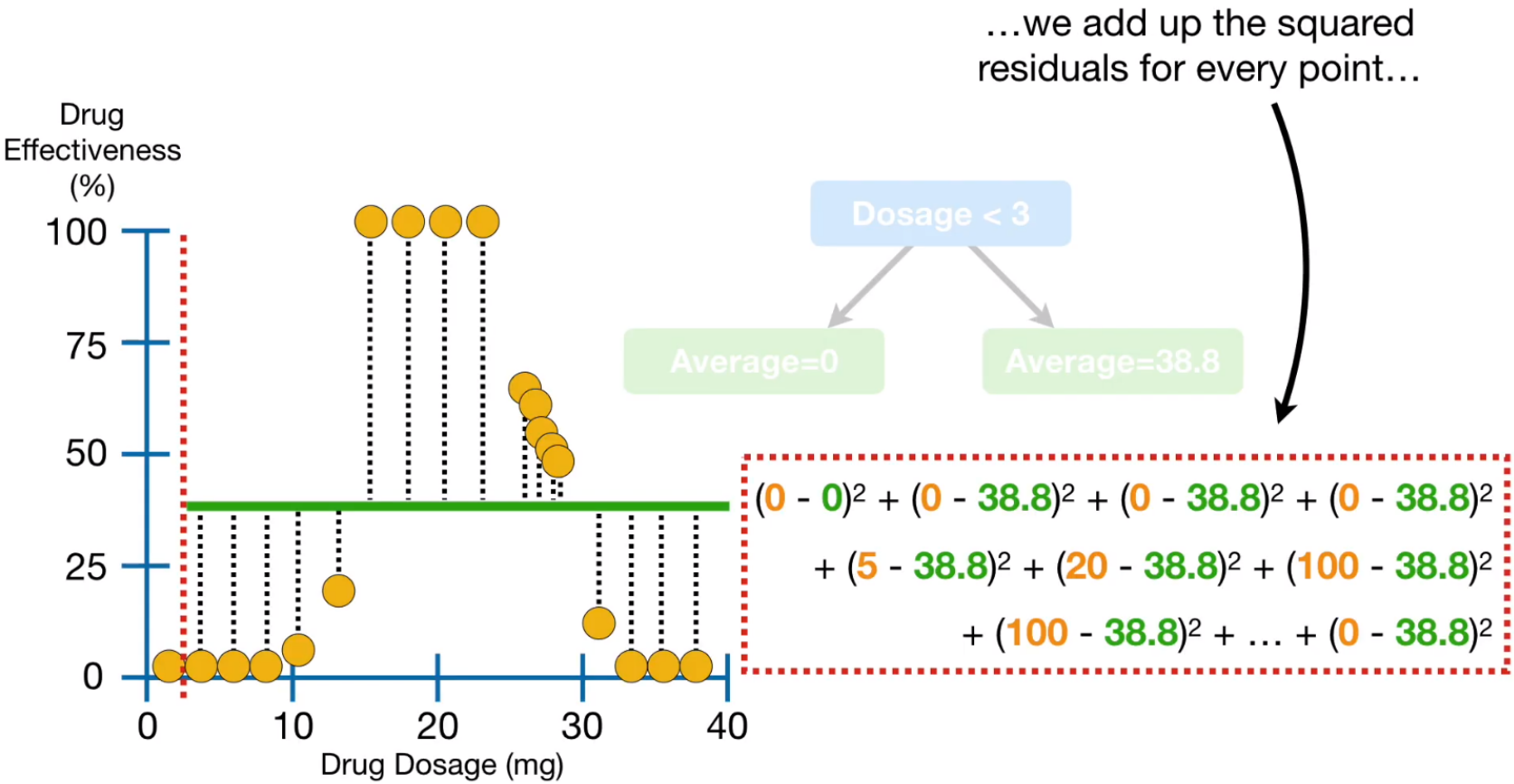
Regression Trees



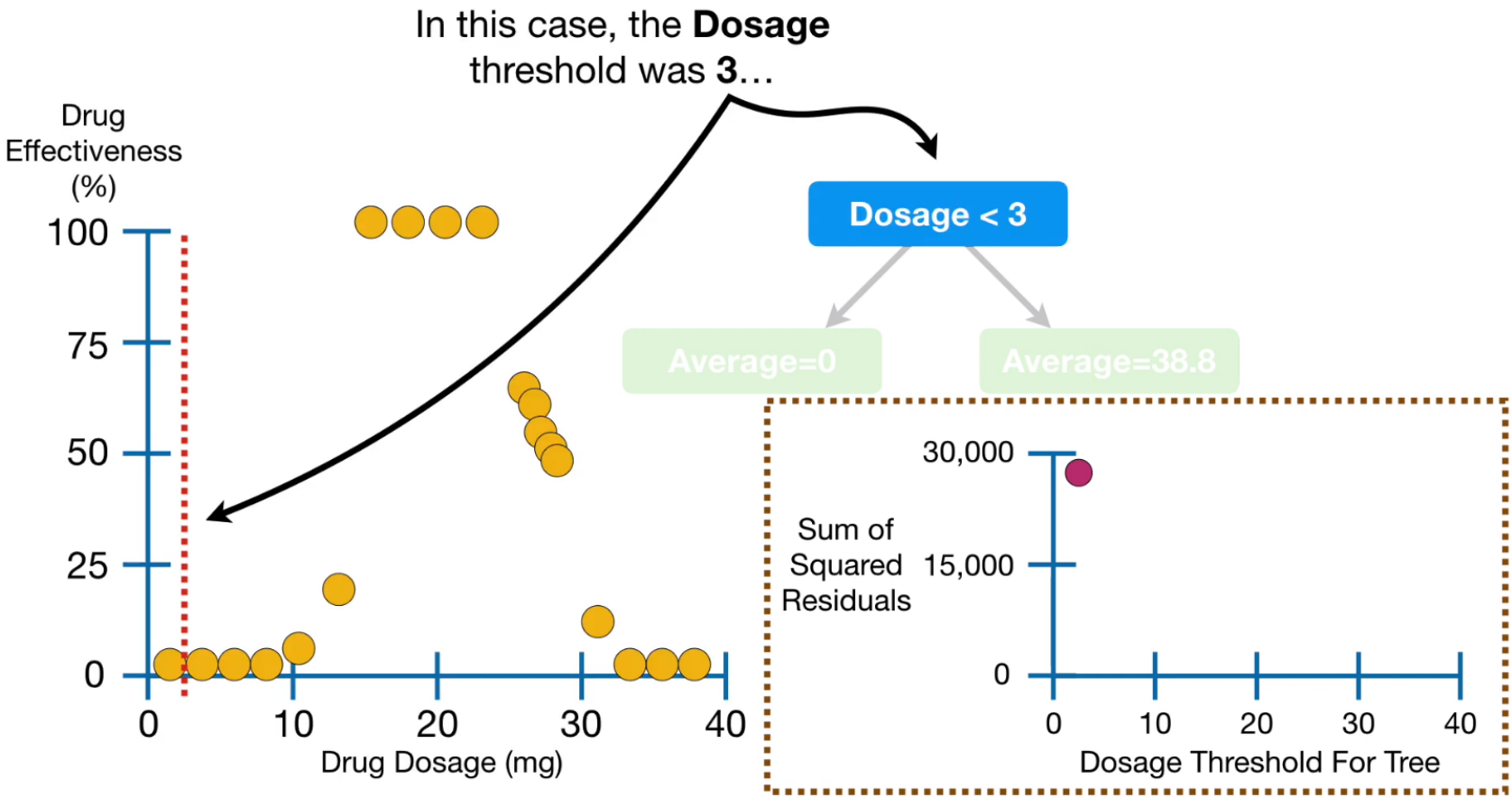
Regression Trees



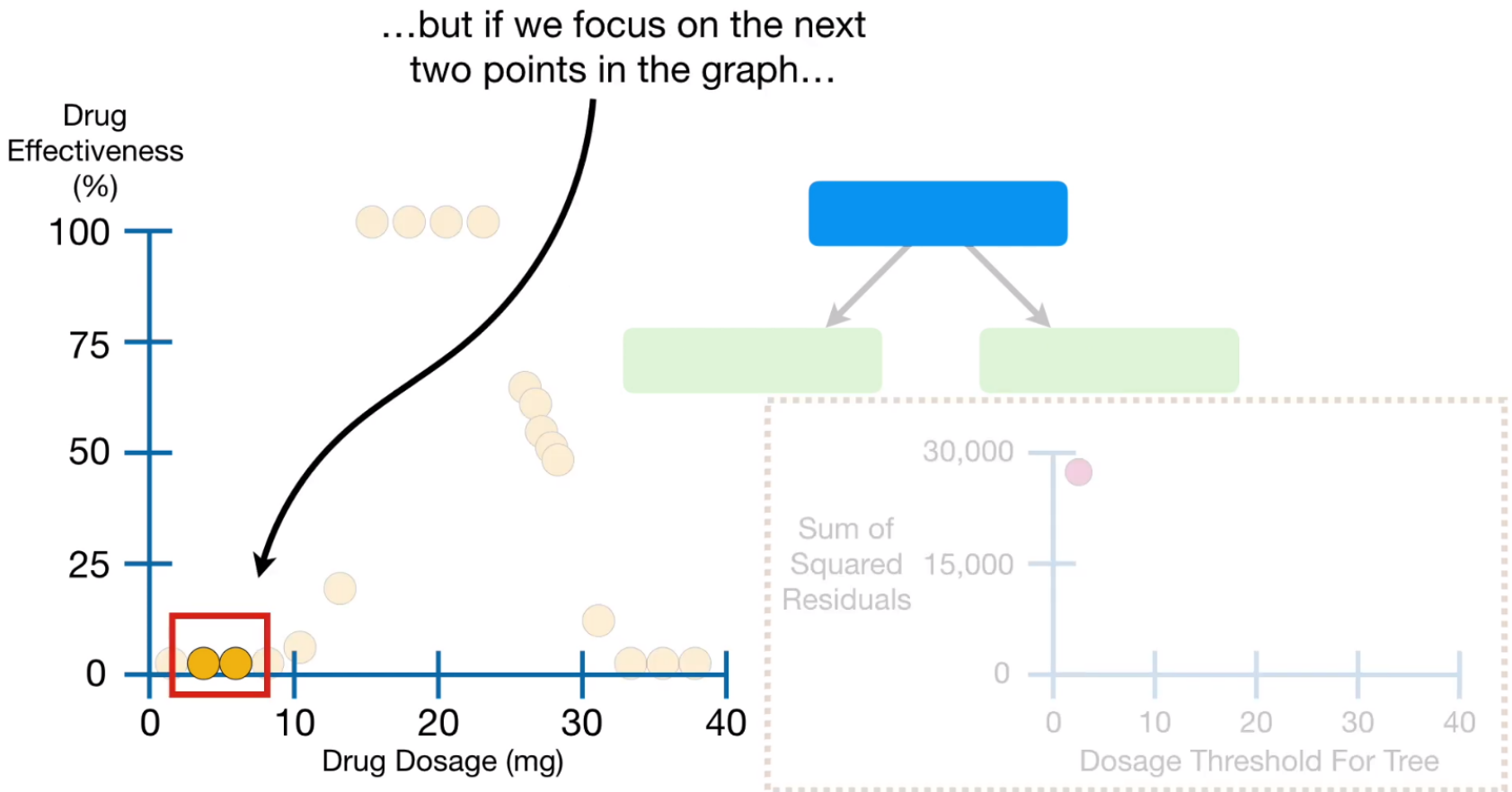
Regression Trees



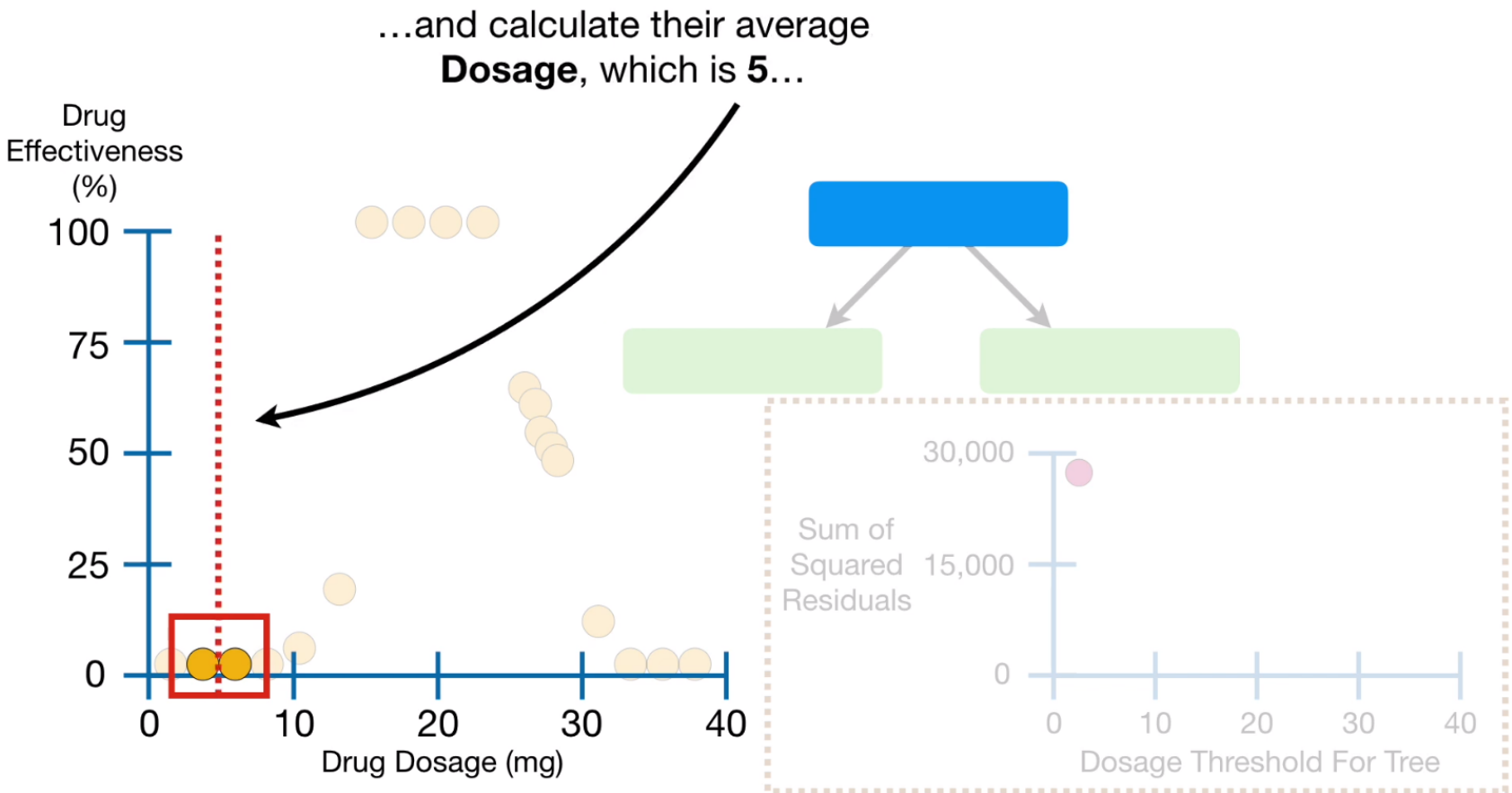
Regression Trees



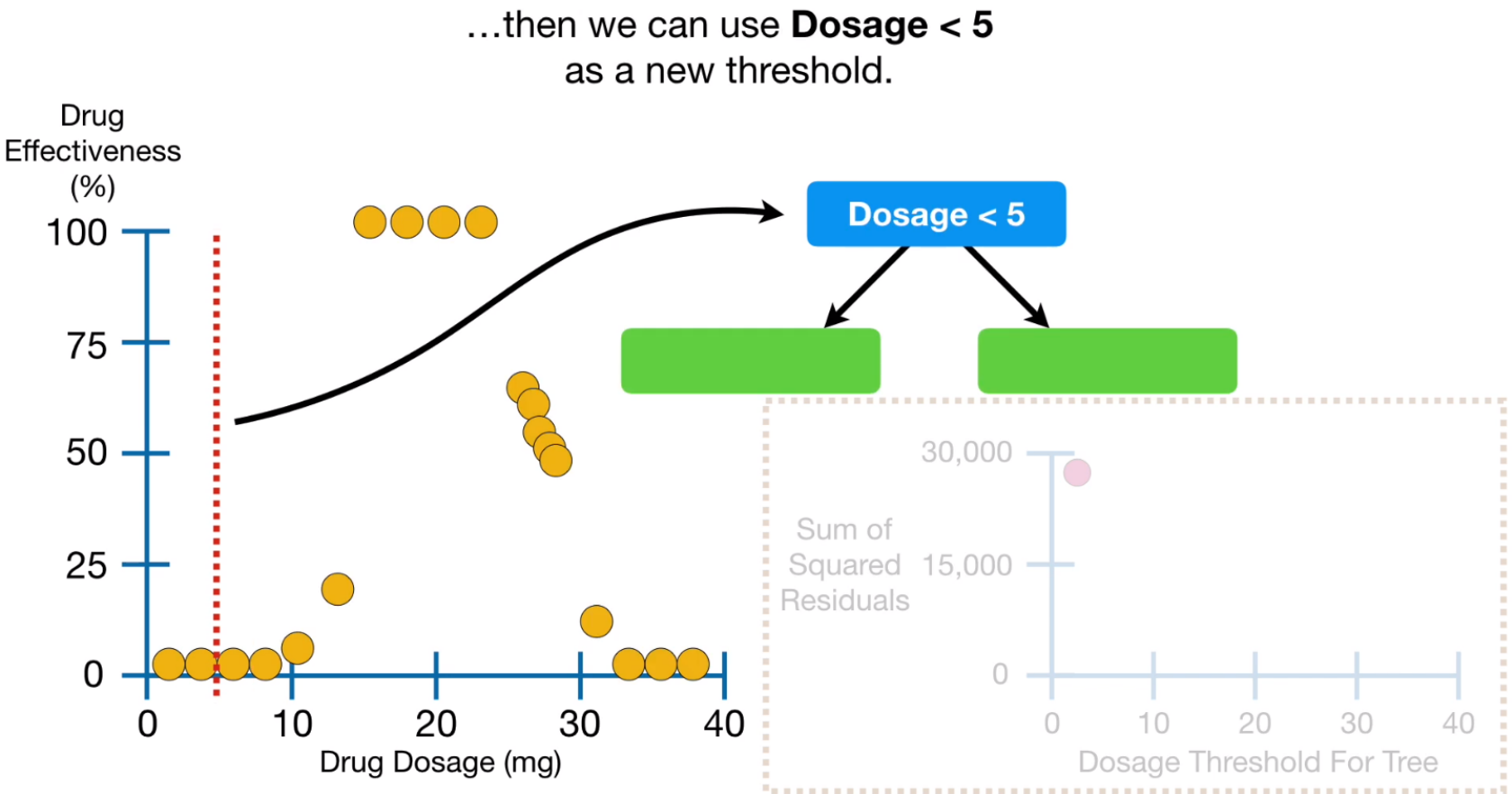
Regression Trees



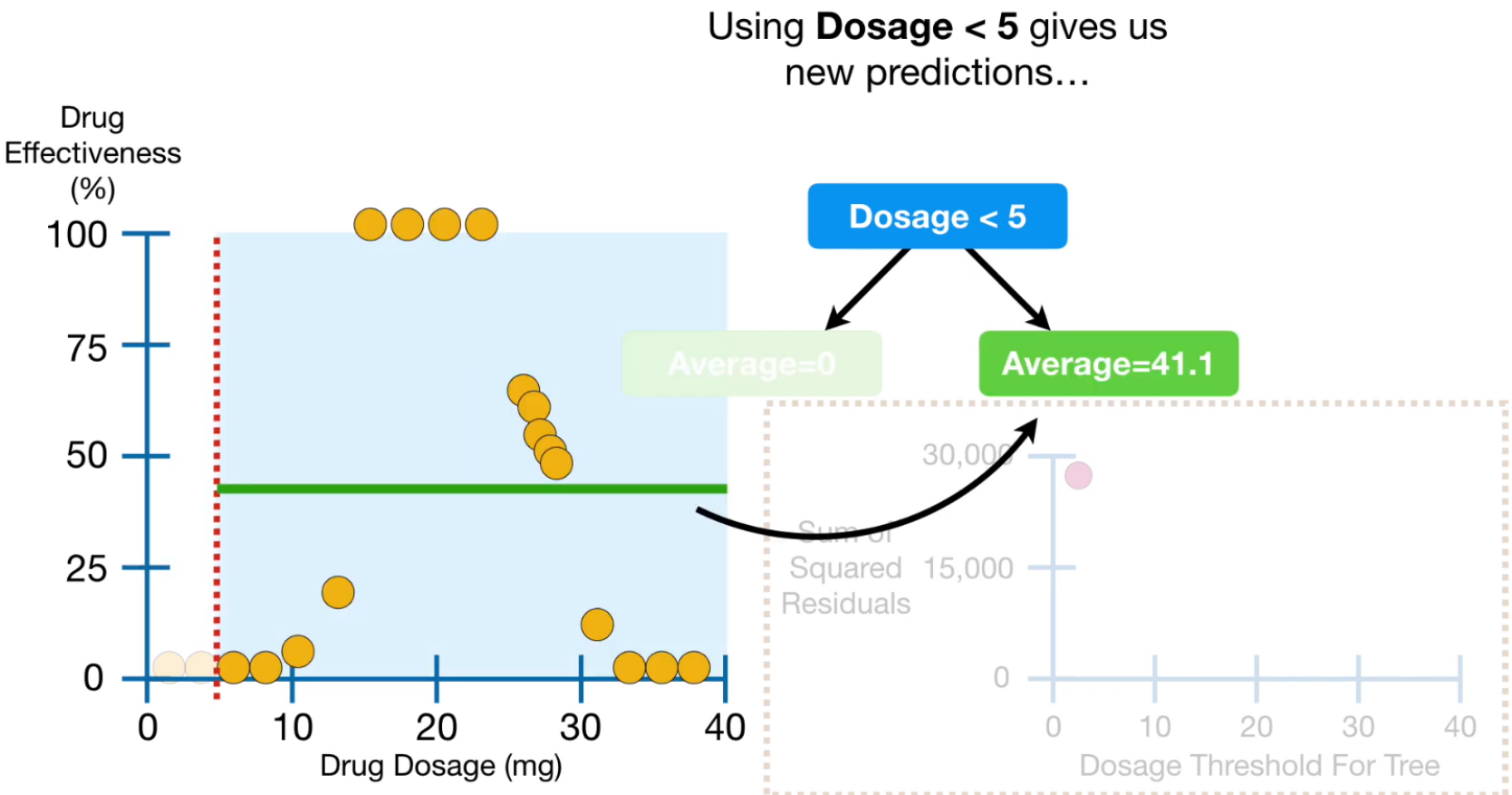
Regression Trees



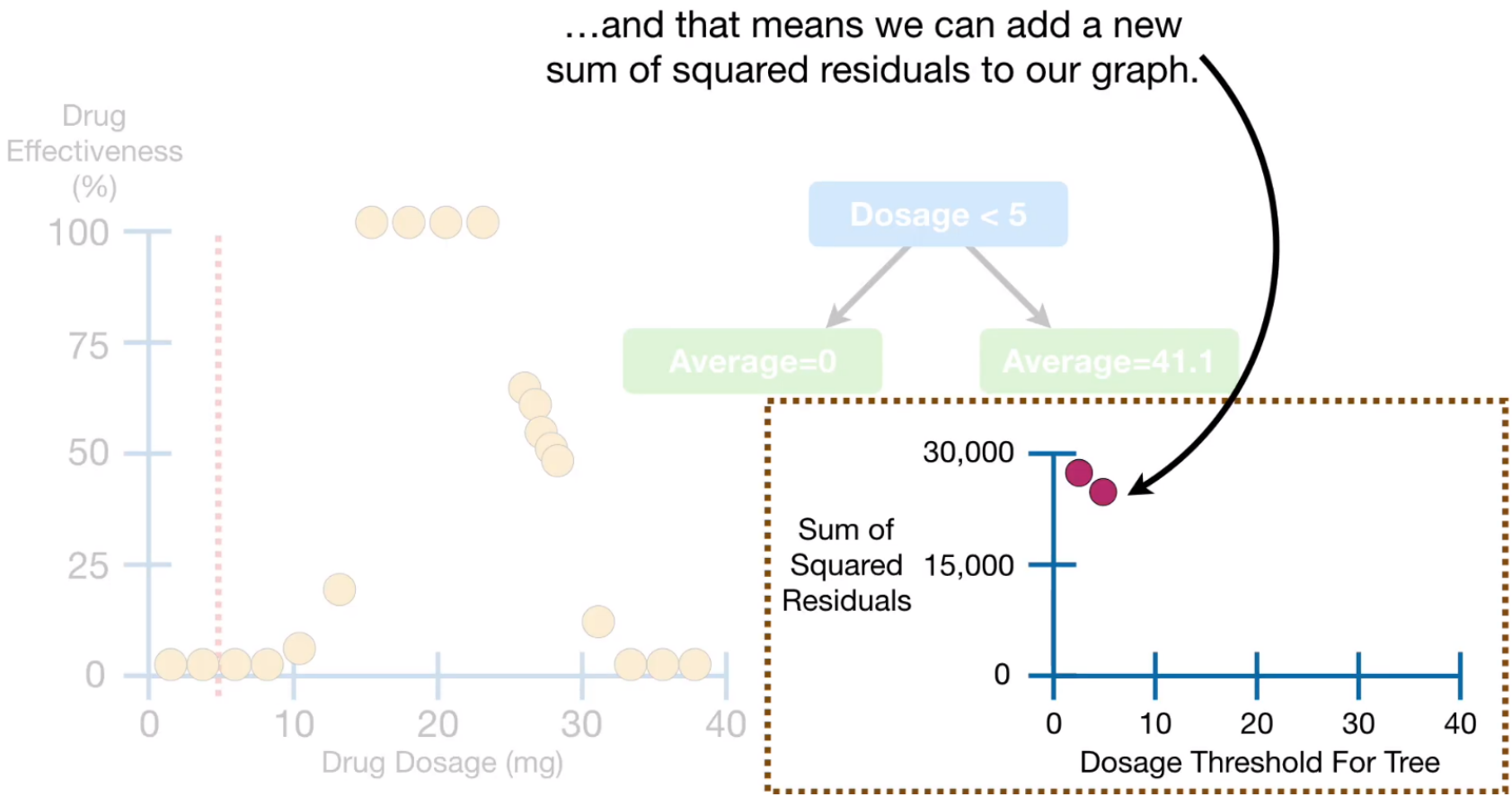
Regression Trees



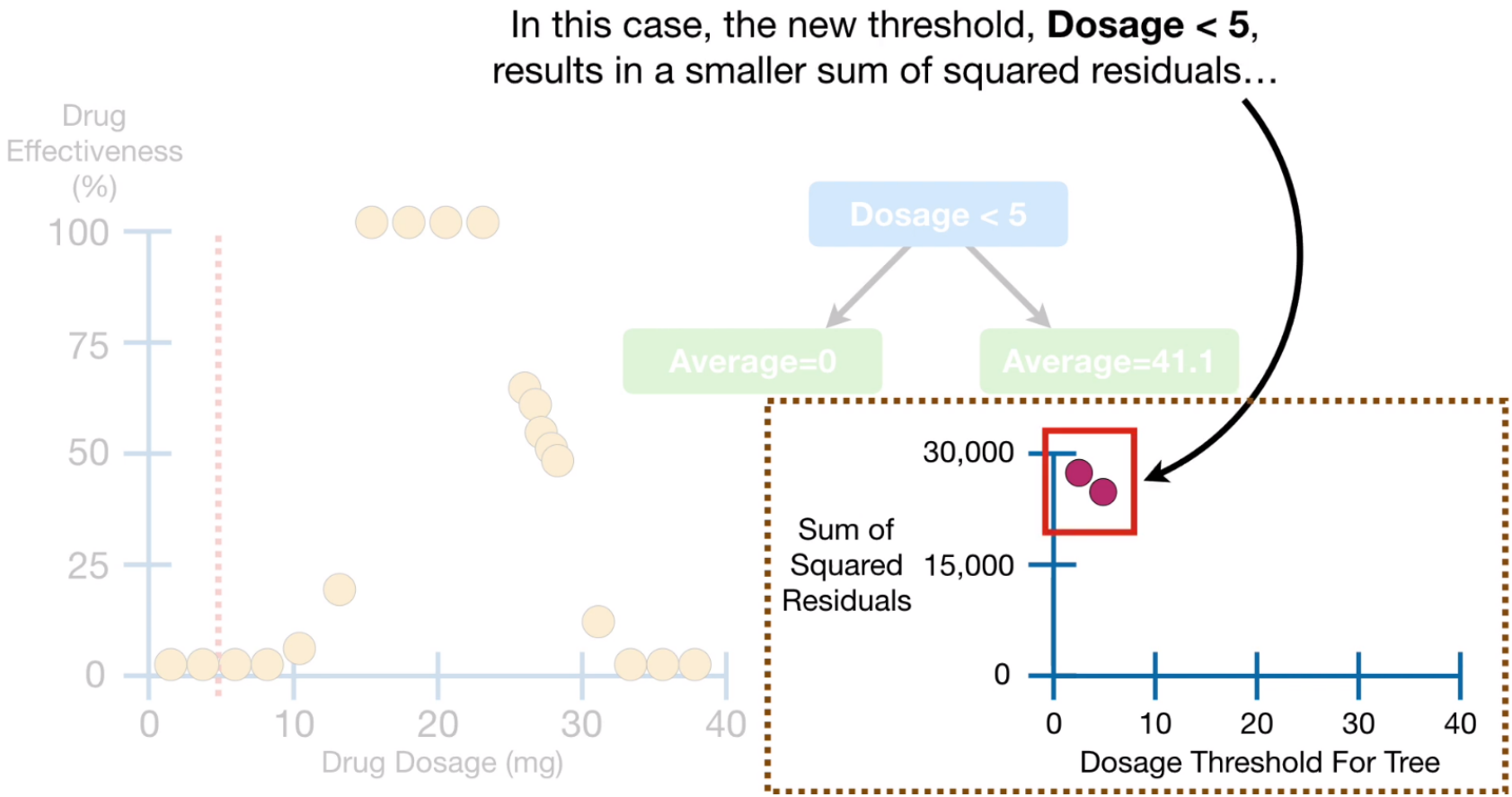
Regression Trees



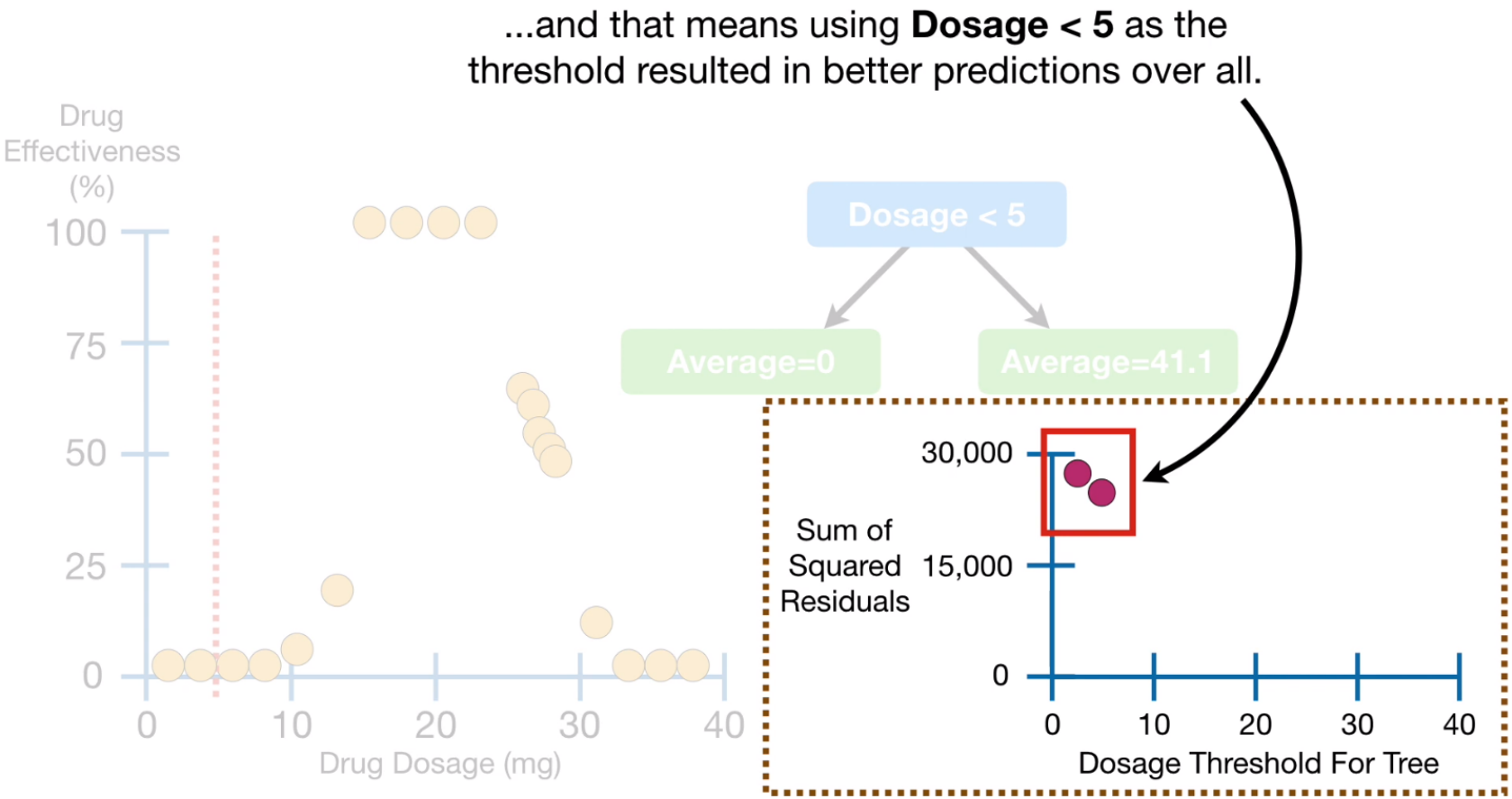
Regression Trees



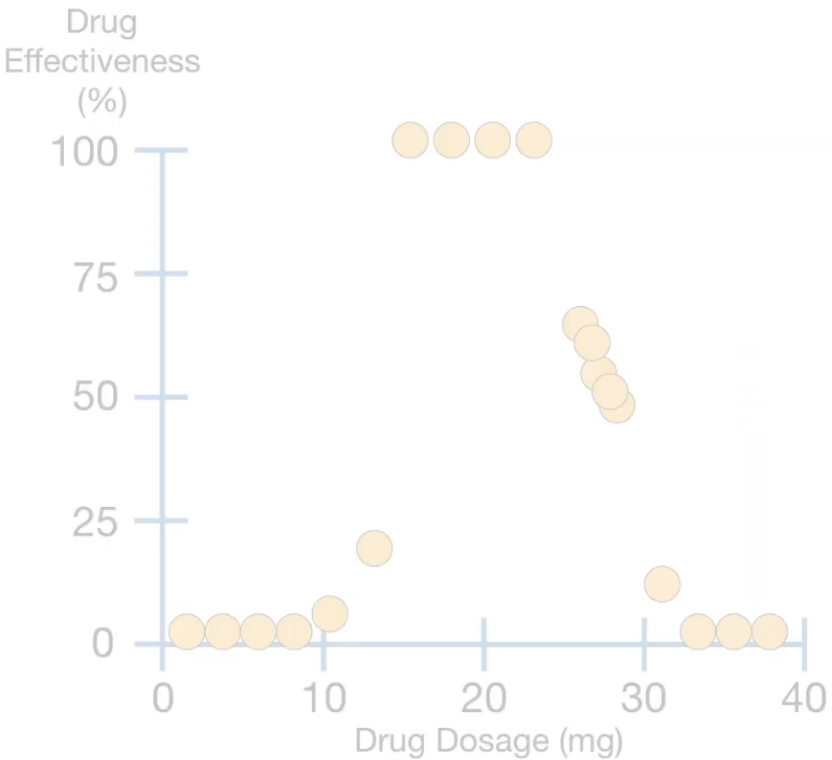
Regression Trees



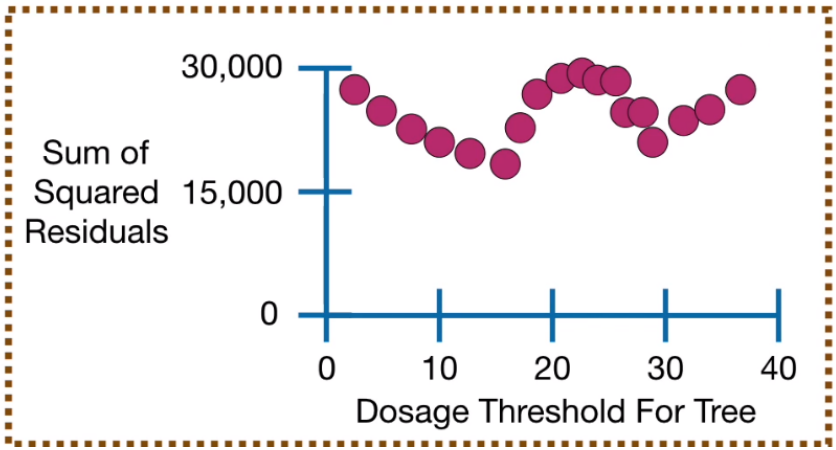
Regression Trees



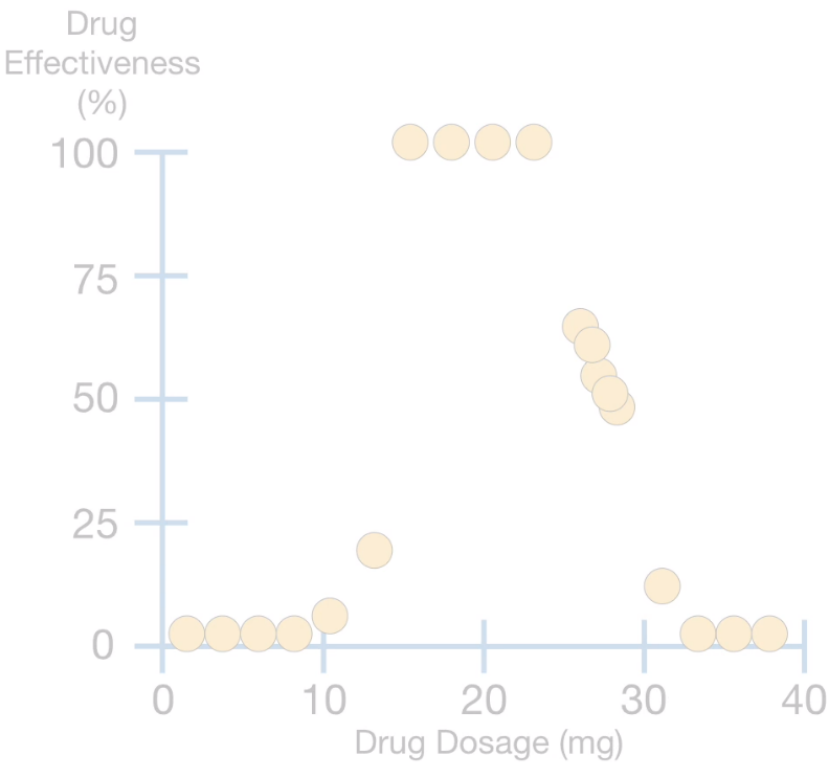
Regression Trees

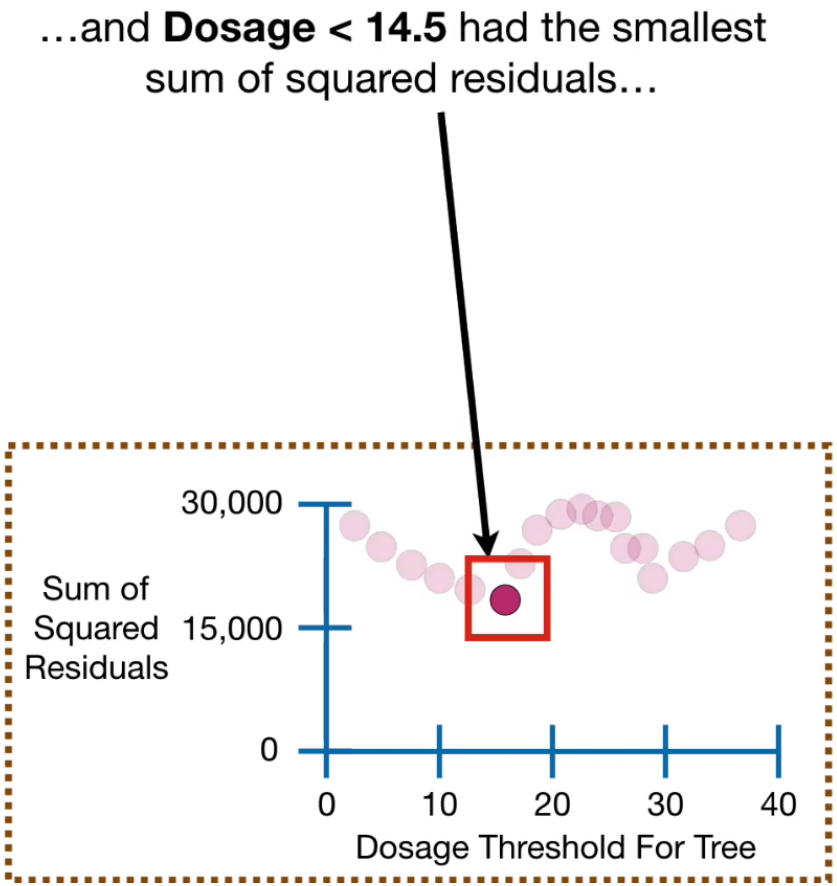


Now we can see the sum of squared residuals for all of the thresholds...

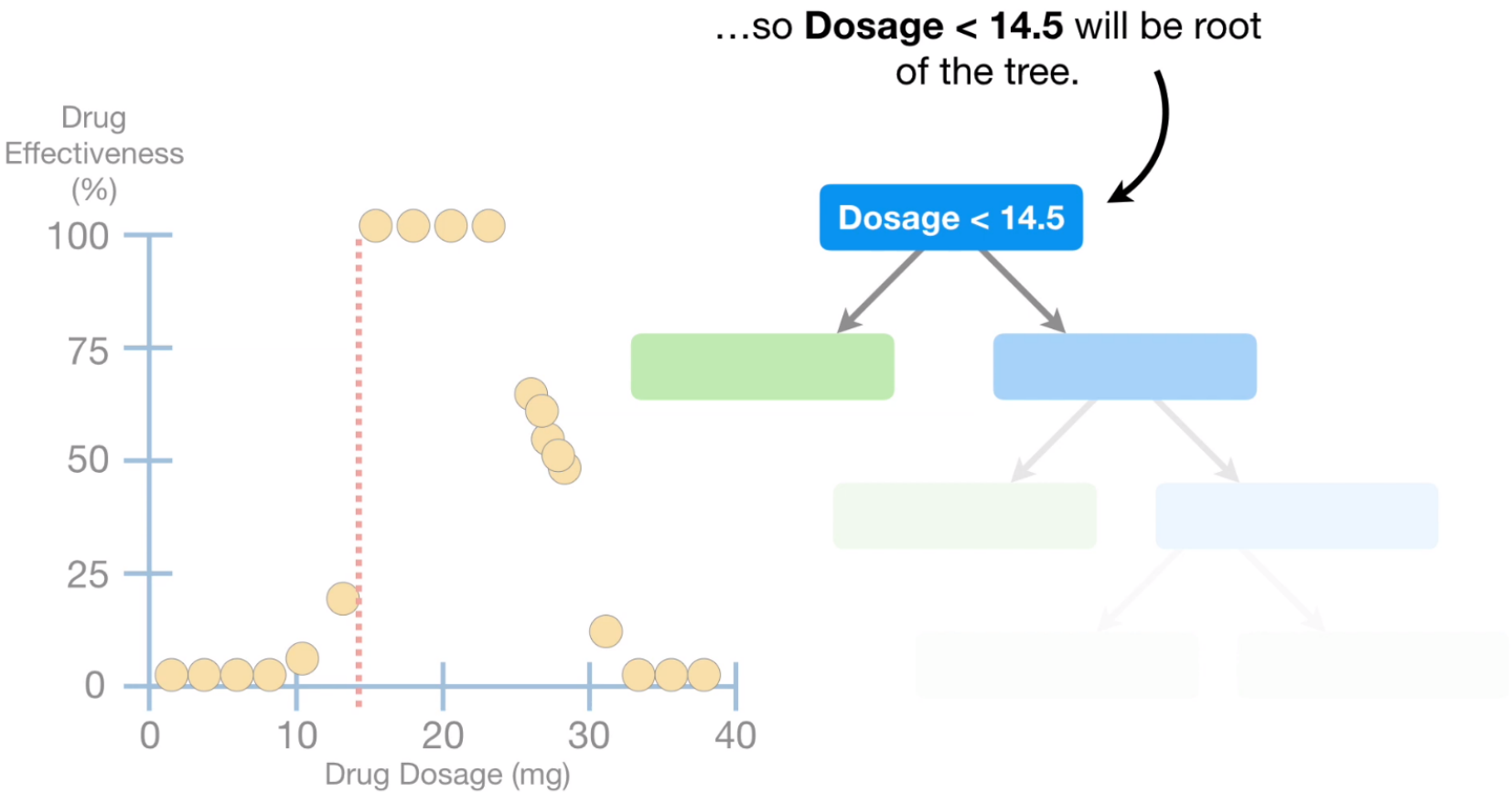


Regression Trees

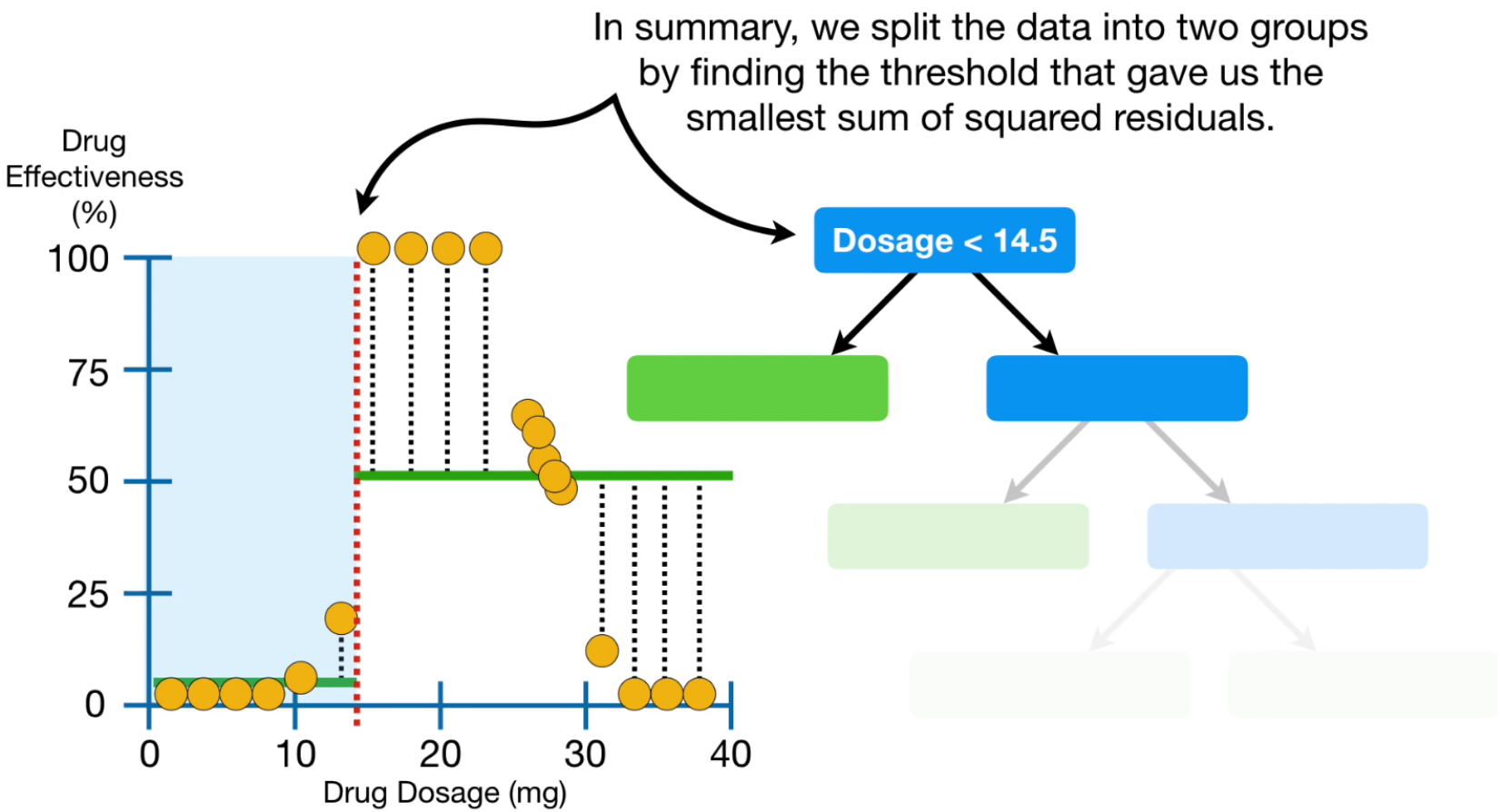




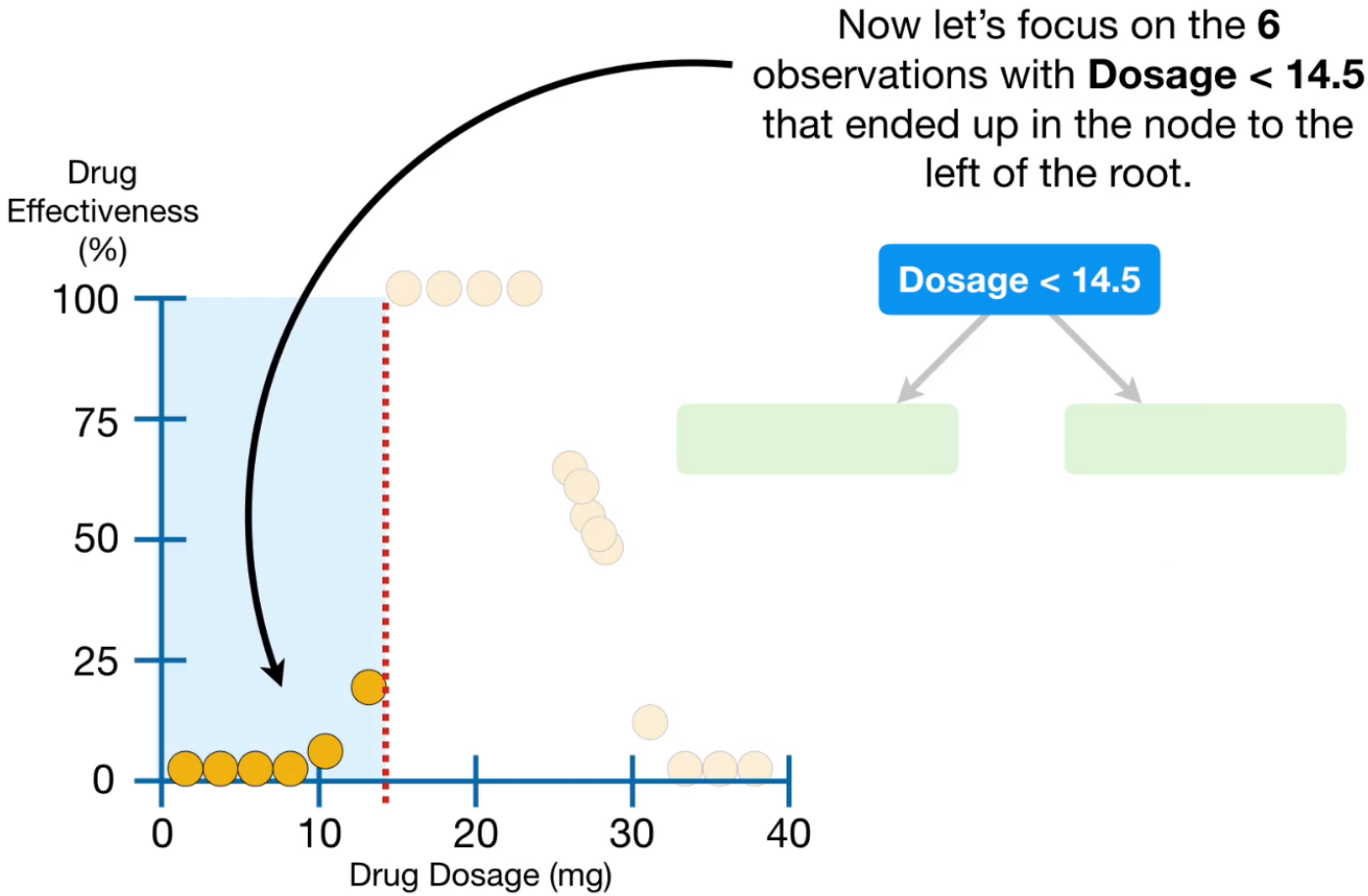
Regression Trees



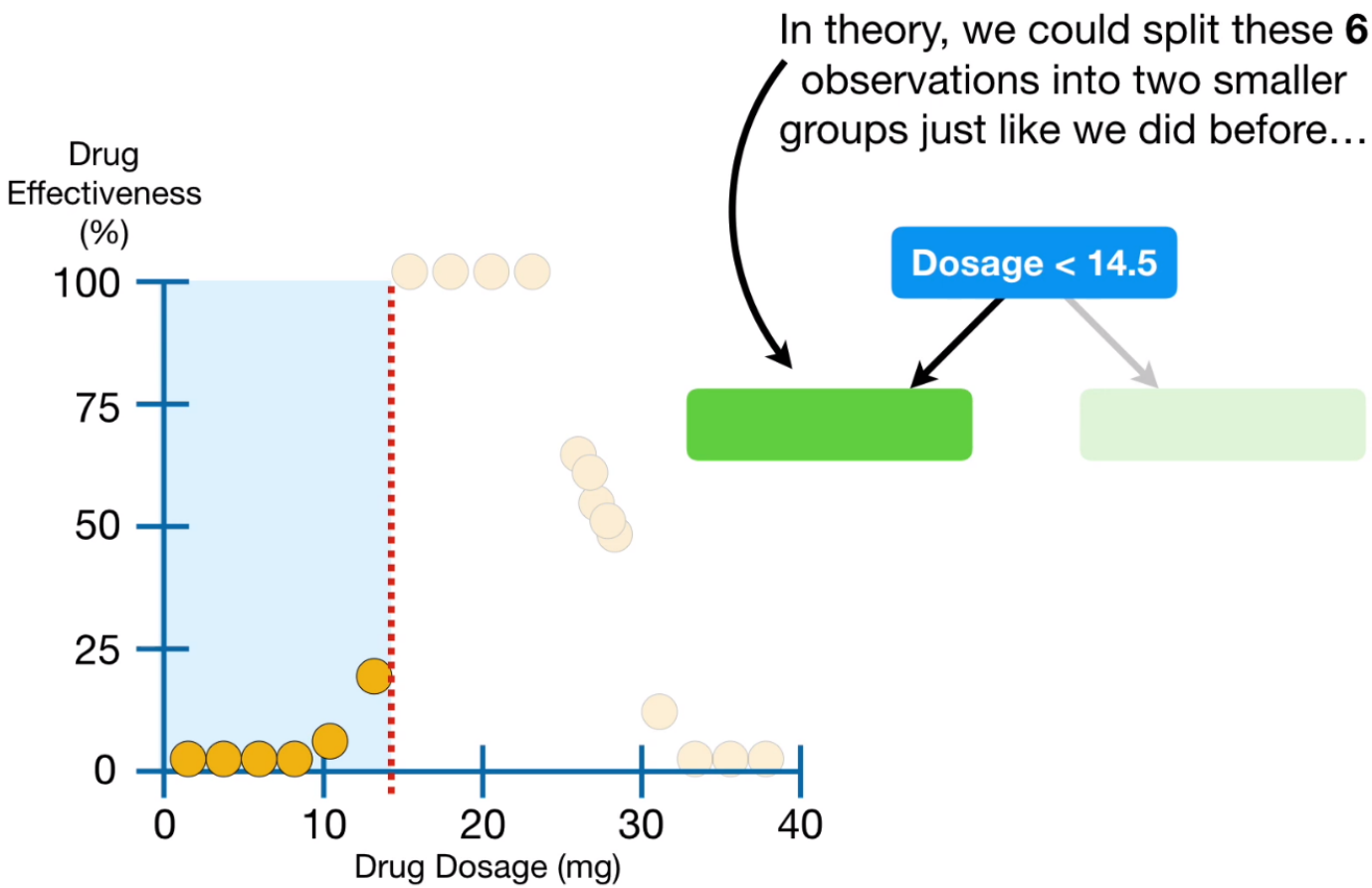
Regression Trees



Regression Trees



Regression Trees



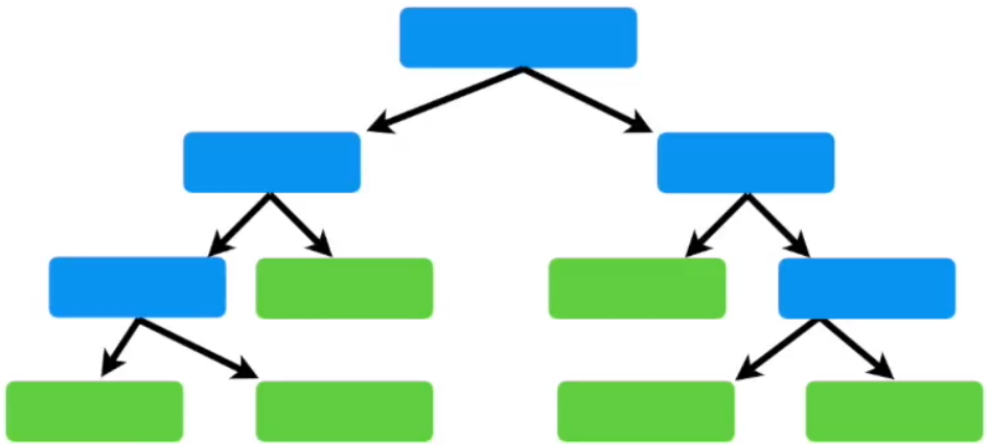
Summary

Properties

- very simple methods
- very fast
- **But:** usually overfit the data
- $\Rightarrow$ solution: Pruning, **Random Forests**

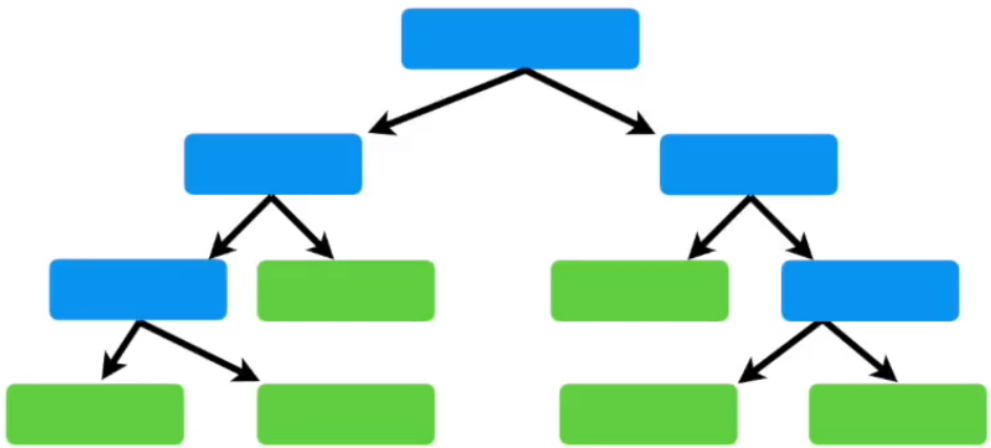
Building random forests

Decision Trees are easy to build, easy to use and easy to interpret...



Building random forests

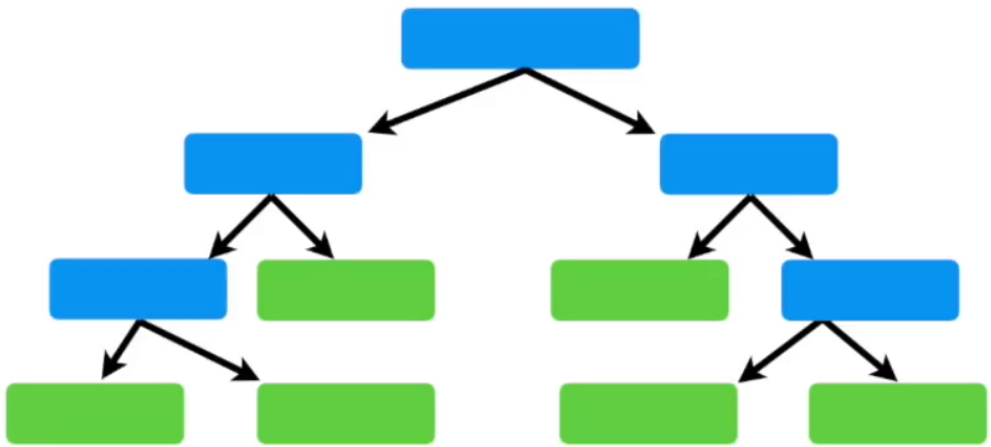
Decision Trees are easy to build, easy to use and easy to interpret...



...but in practice they are not that awesome.

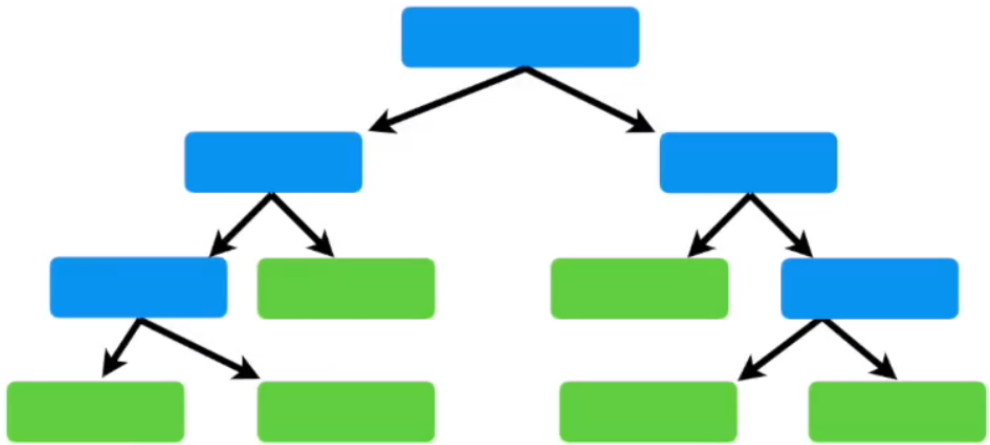
Building random forests

To quote from ***The Elements of Statistical Learning*** (aka The Bible of Machine Learning), "Trees have one aspect that prevents them from being the ideal tool for predictive learning, namely **inaccuracy**."



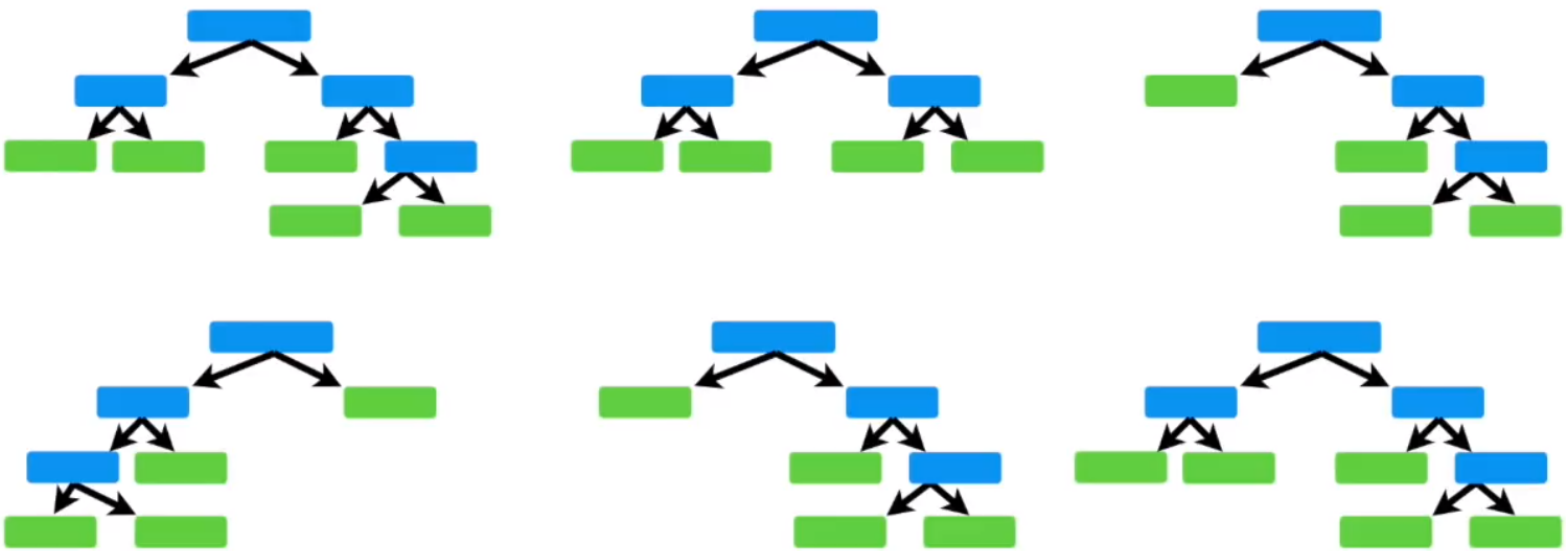
Building random forests

In other words, they work great with the data used to create them, but **they are not flexible when it comes to classifying new samples.**



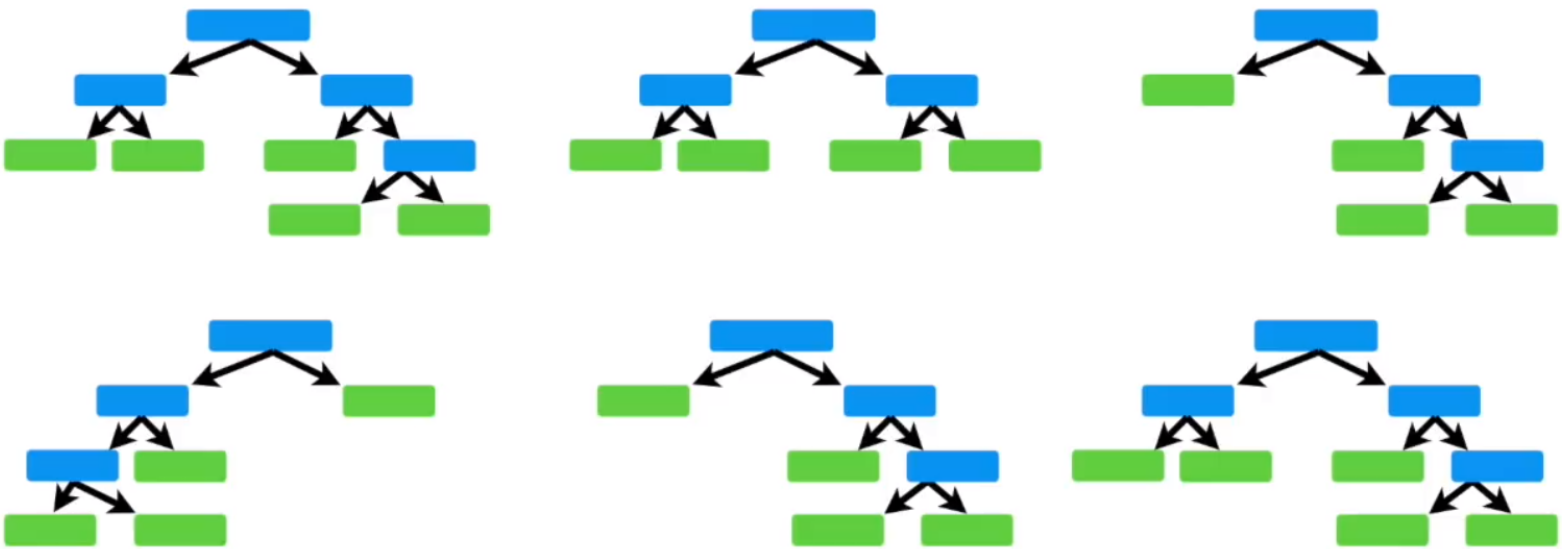
Building random forests

The good news is that **Random Forests** combine the simplicity of decision trees with flexibility resulting in a vast improvement in accuracy.



Building random forests

So let's make a Random Forest!!!



Building random forests

Step 1: Create a "bootstrapped" dataset.

Building random forests

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

Imagine that these 4 samples are the entire dataset that we are going to build a tree from…

Building random forests

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease

To create a bootstrapped dataset that is the same size as the original, we just randomly select samples from the original dataset.

Building random forests

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease

To create a bootstrapped dataset that is the same size as the original, we just randomly select samples from the original dataset.

The important detail is that we're allowed to pick the same sample more than once.

Building random forests

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes

…so it's the first sample in our bootstrapped dataset.

Building random forests

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No

...so it's the second sample in our bootstrapped dataset.

Building random forests

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes

...so here it is in the bootstrapped dataset.

Building random forests

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

...and here it is.

Building random forests

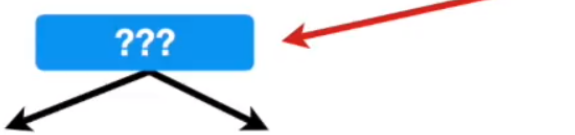
Step 2: Create a decision tree using the bootstrapped dataset, but only use a random subset of variables (or columns) at each step.

Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

Building random forests

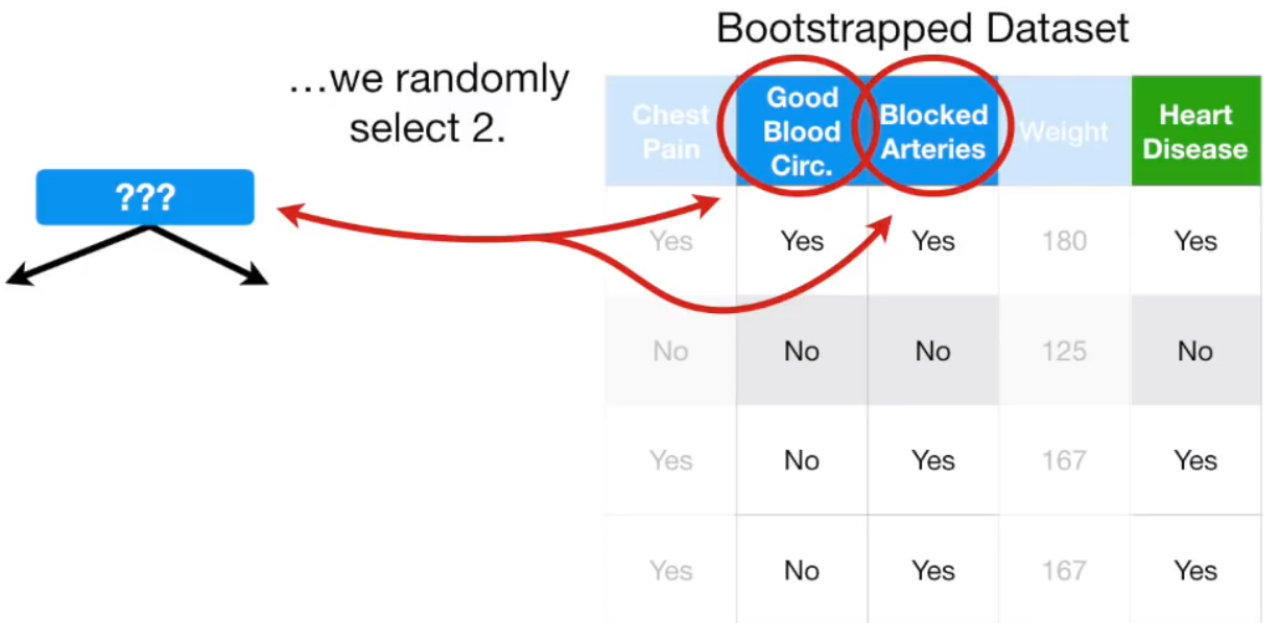
Thus, instead of considering all 4 variables to figure out how to split the root node...



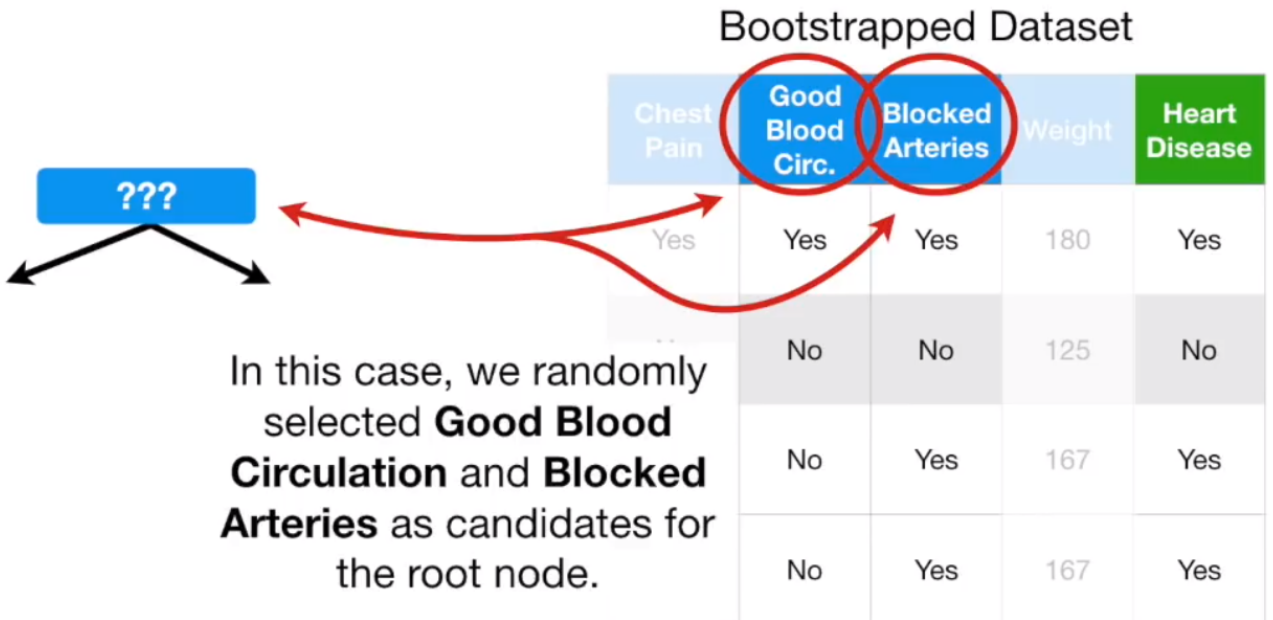
Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

Building random forests

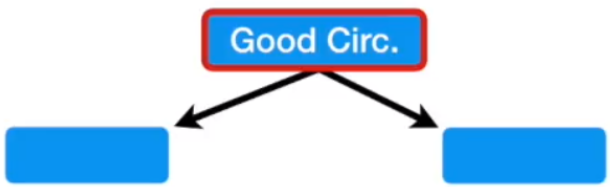


Building random forests



Building random forests

Just for the sake of the example, assume that **Good Blood Circulation** did the best job separating the samples.

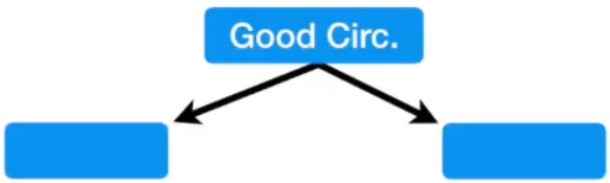


Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

Building random forests

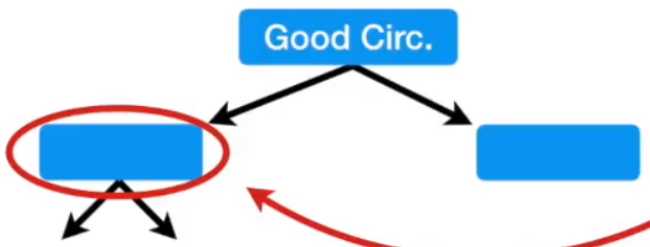
Since we used **Good Blood Circulation**, I'm going to grey it out so that we focus on the remaining variables.



Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

Building random forests

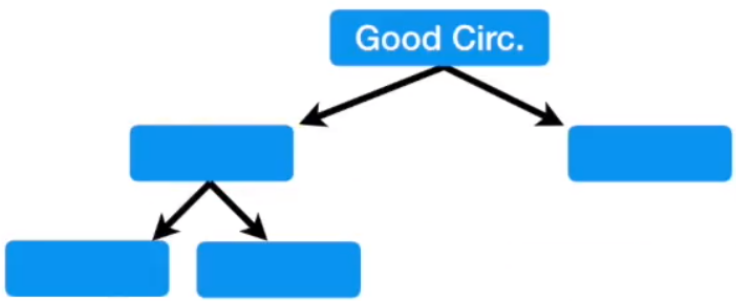


Just like for the root, we randomly select 2 variables as candidates, instead of all 3 remaining columns.

Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

Building random forests



Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

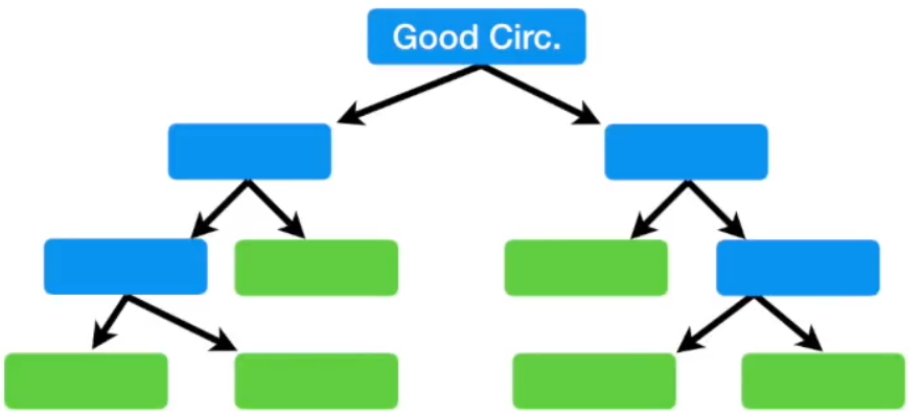
And we just build the tree as usual, but only considering a random subset of variables at each step.

Building random forests

We built a tree...

1) Using a bootstrapped dataset

2) Only considering a random a subset of variables at each step.

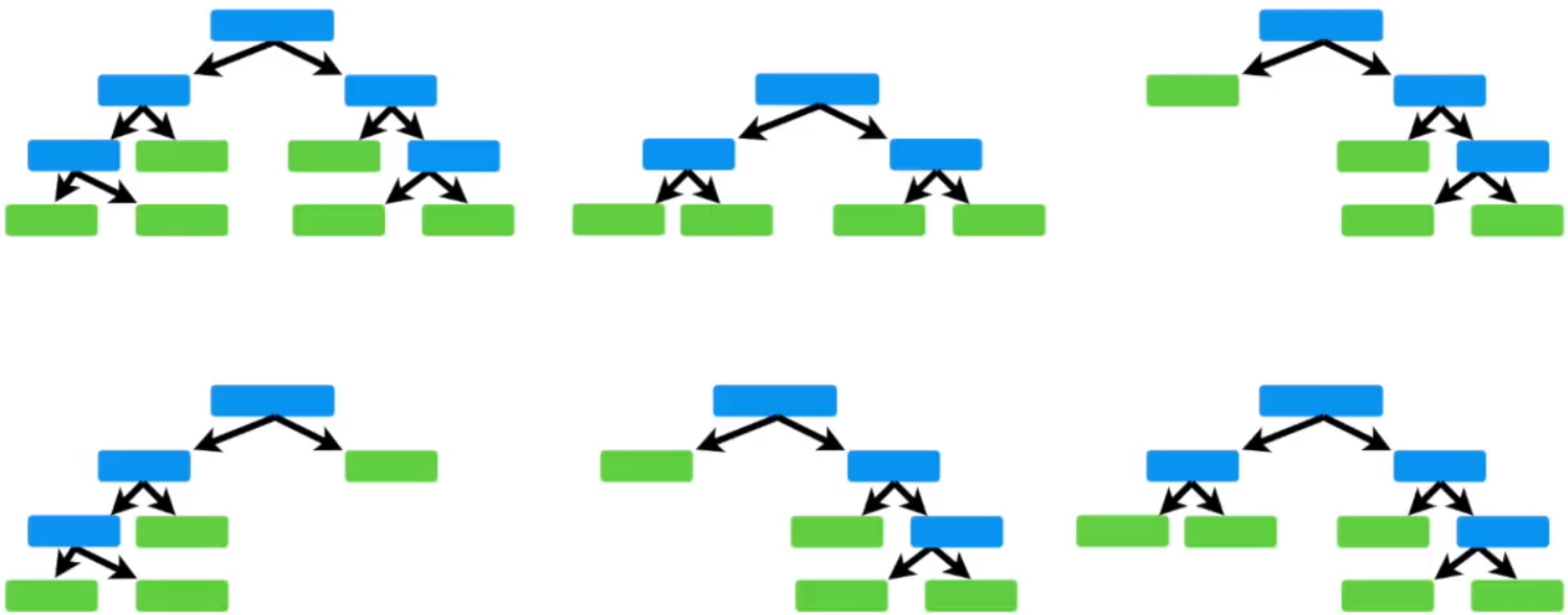


Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

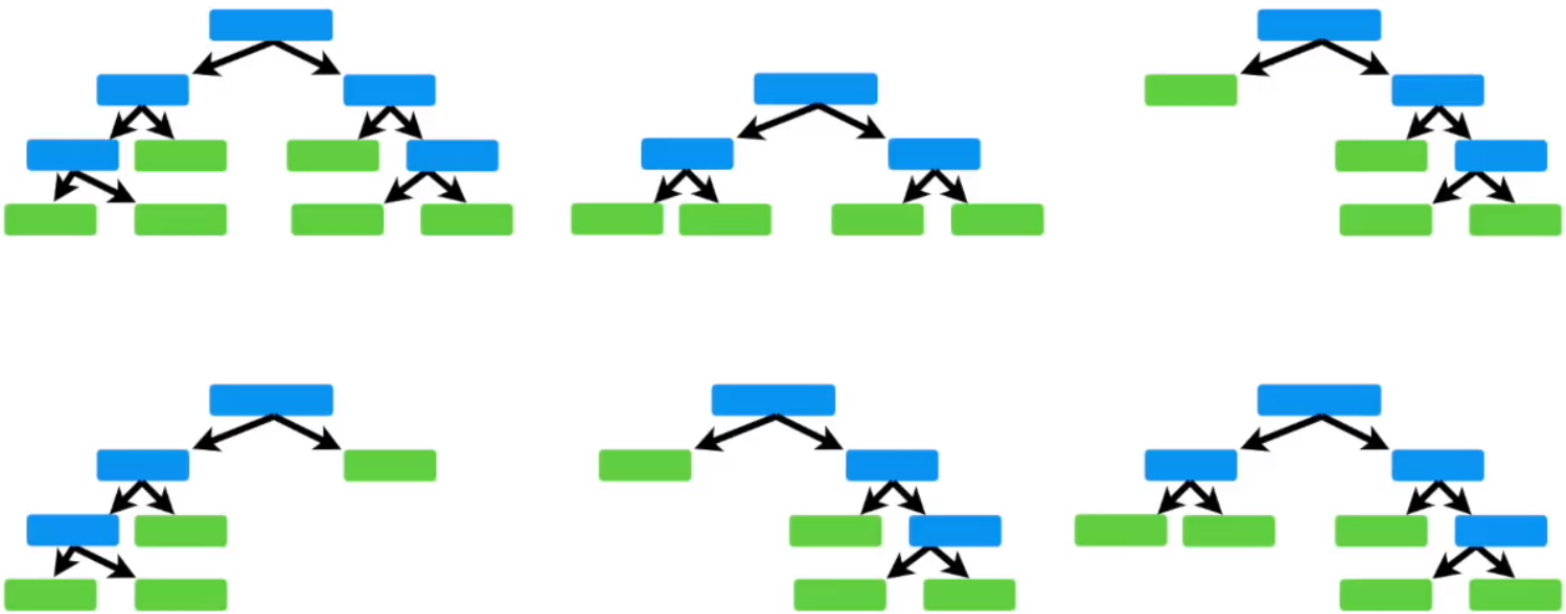
Building random forests

Now go back to Step 1 and repeat: Make a new bootstrapped dataset and build a tree considering a subset of variables at each step.



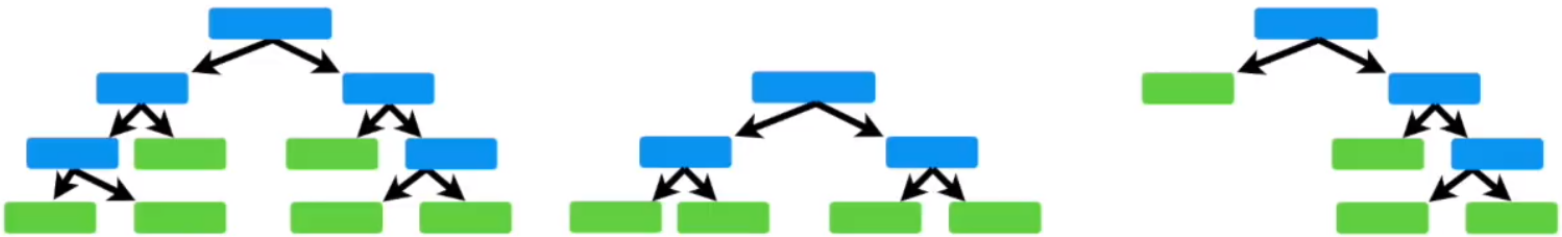
Building random forests

Ideally, you'd do this 100's of times, but we only have space to show 6... but you get the idea.

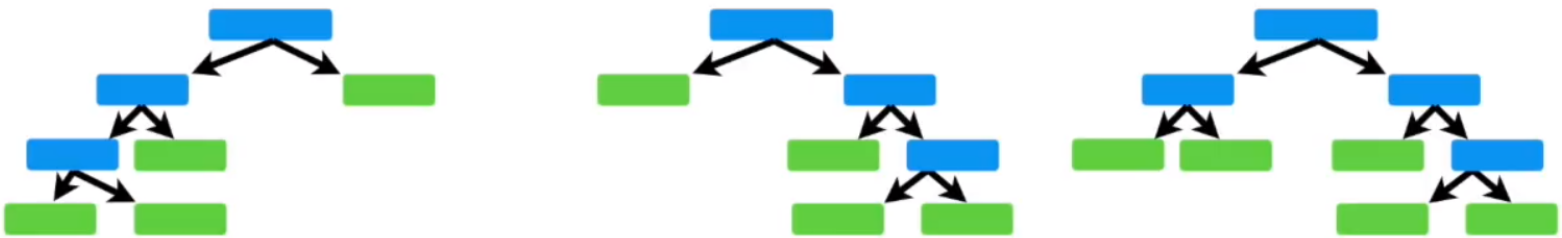


Building random forests

Using a bootstrapped sample and considering only a subset of the variables at each step results in a wide variety of trees.

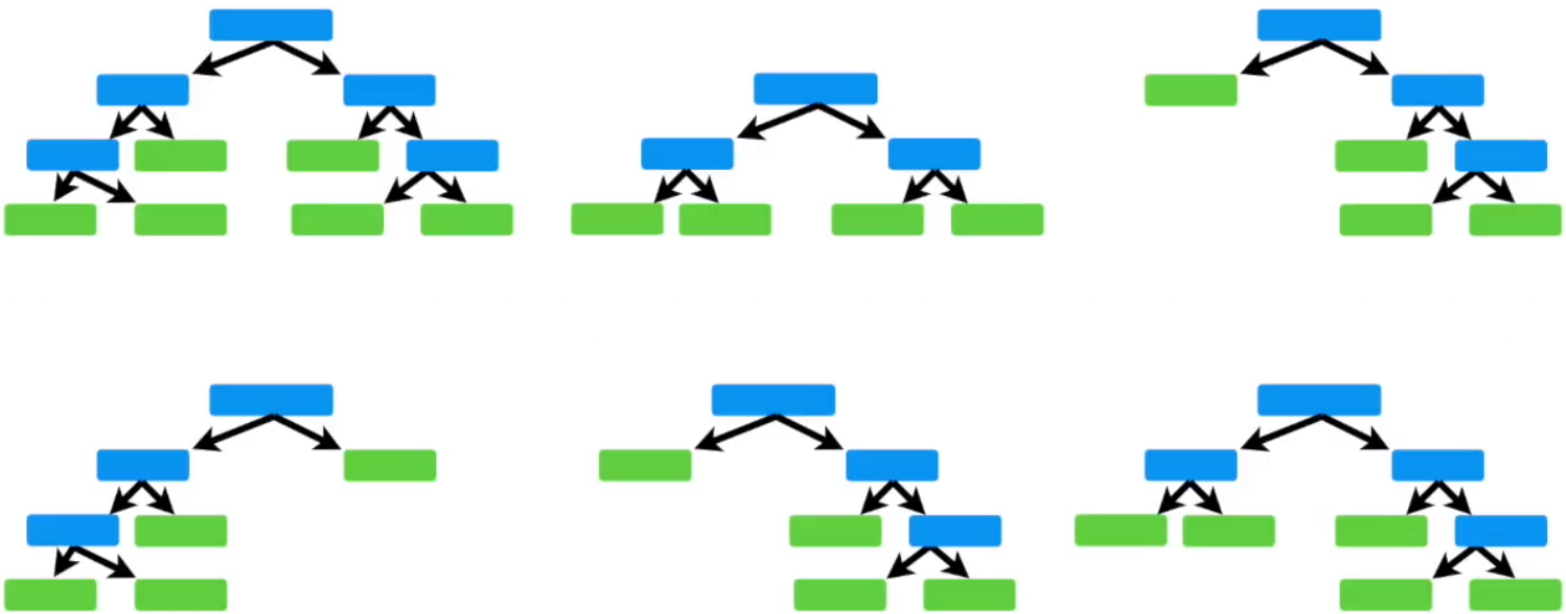


The variety is what makes random forests more effective than individual decision trees.



Evaluating a random forest

Sweet!!! Now that we've created a random forest, how do we use it?



Evaluating a random forest

Well, first we get a new patient...

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	No	168	

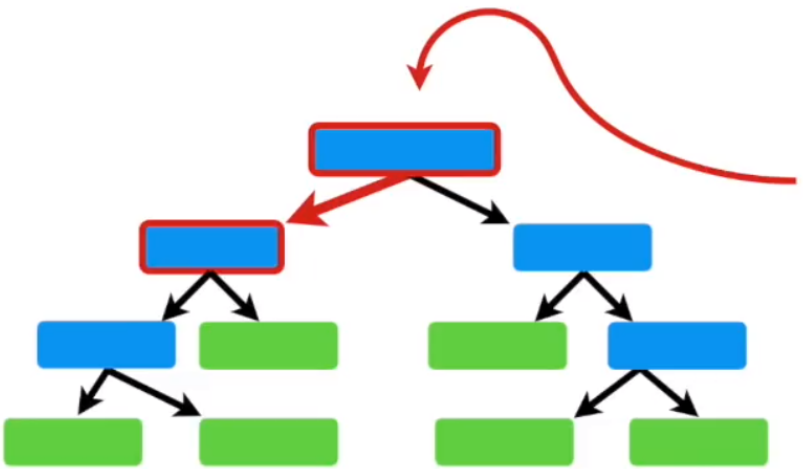
Evaluating a random forest

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	No	168	

...and now we want to know if they have heart disease or not.

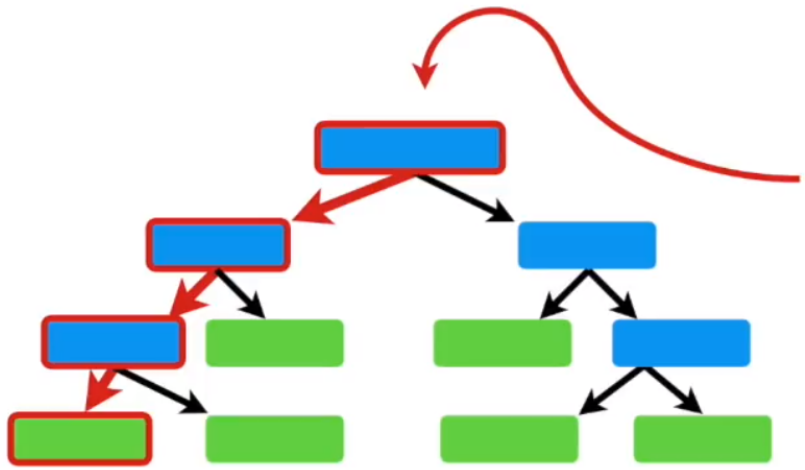
Evaluating a random forest

So we take the data and run it down the first tree that we made...



Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	No	168	

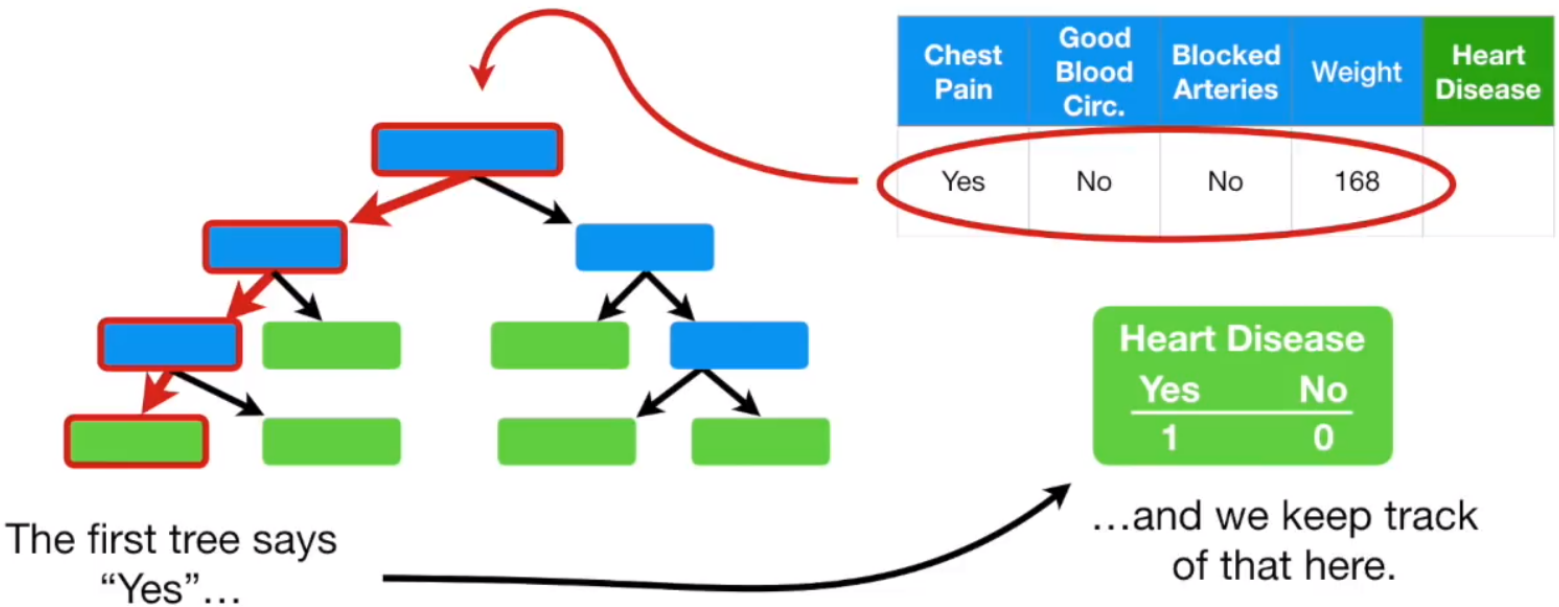
Evaluating a random forest



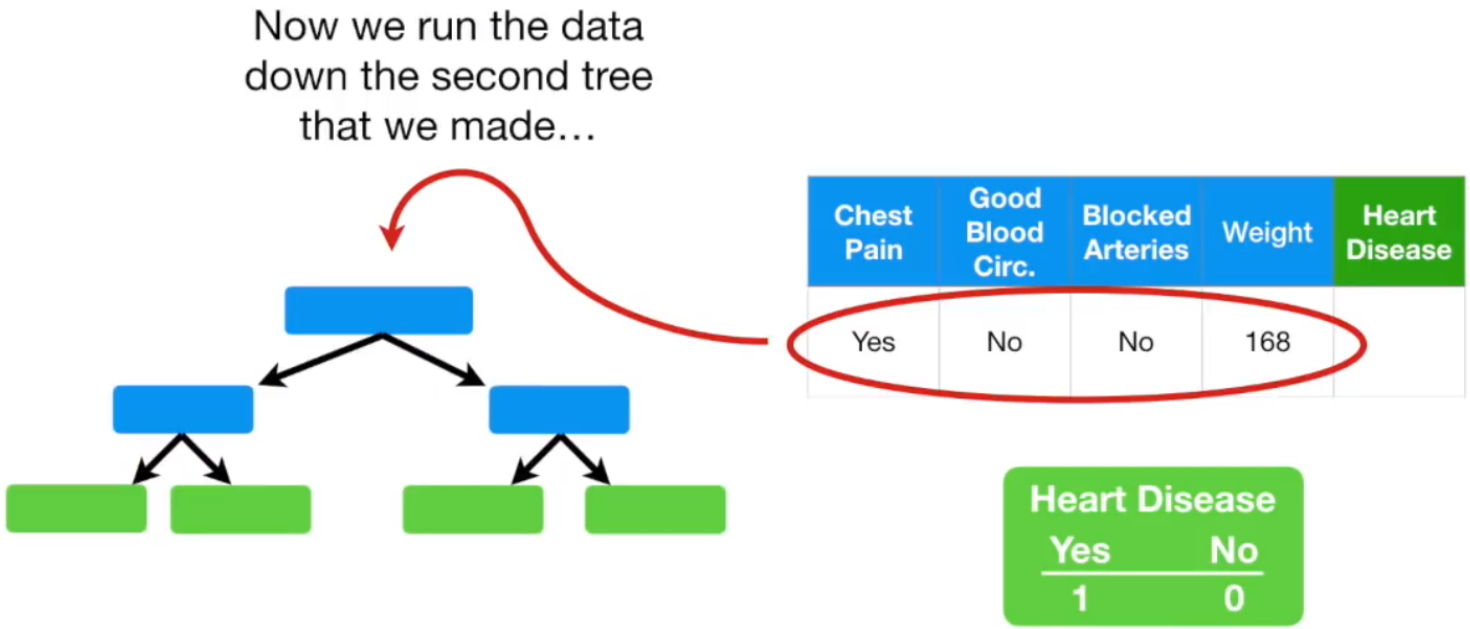
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	No	168	

The first tree says "Yes"...

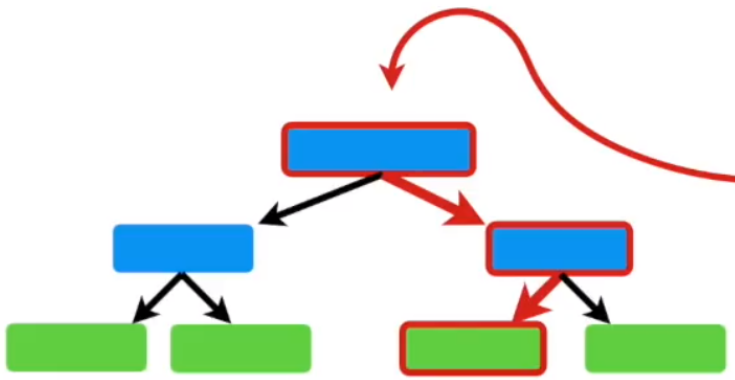
Evaluating a random forest



Evaluating a random forest



Evaluating a random forest



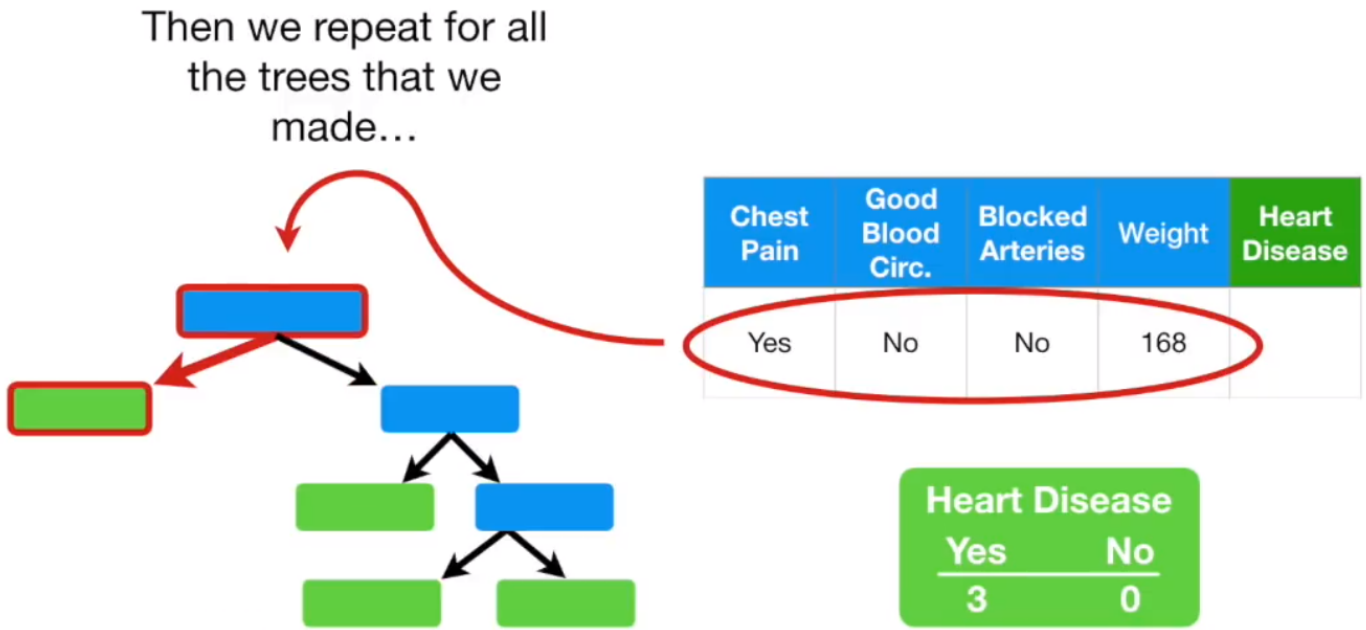
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	No	168	

The second tree also says "Yes"...

Heart Disease	
Yes	No
2	0

...and we keep track of that here.

Evaluating a random forest



Evaluating a random forest

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	No	168	

After running the data down all of the trees in the random forest, we see which option received more votes.

Heart Disease	
Yes	No
5	1

Evaluating a random forest

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	No	168	**YES**

In this case, "**Yes**" received the most votes, so we will conclude that this patient has heart disease.

Heart Disease	
Yes	No
5	1

Evaluating a random forest

Terminology Alert!!!

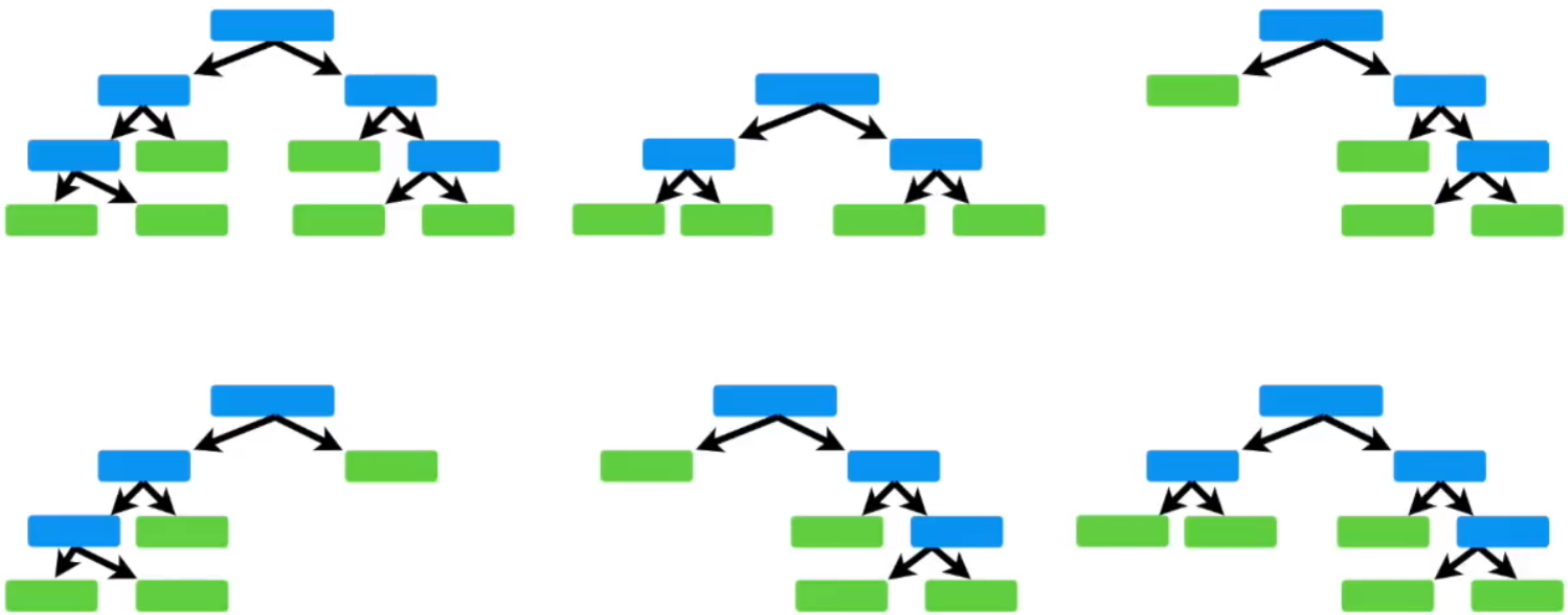
Bootstrapping the data plus using the **aggregate** to make a decision is called **"Bagging"**

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	No	168	**YES**

Heart Disease	
Yes	No
5	1

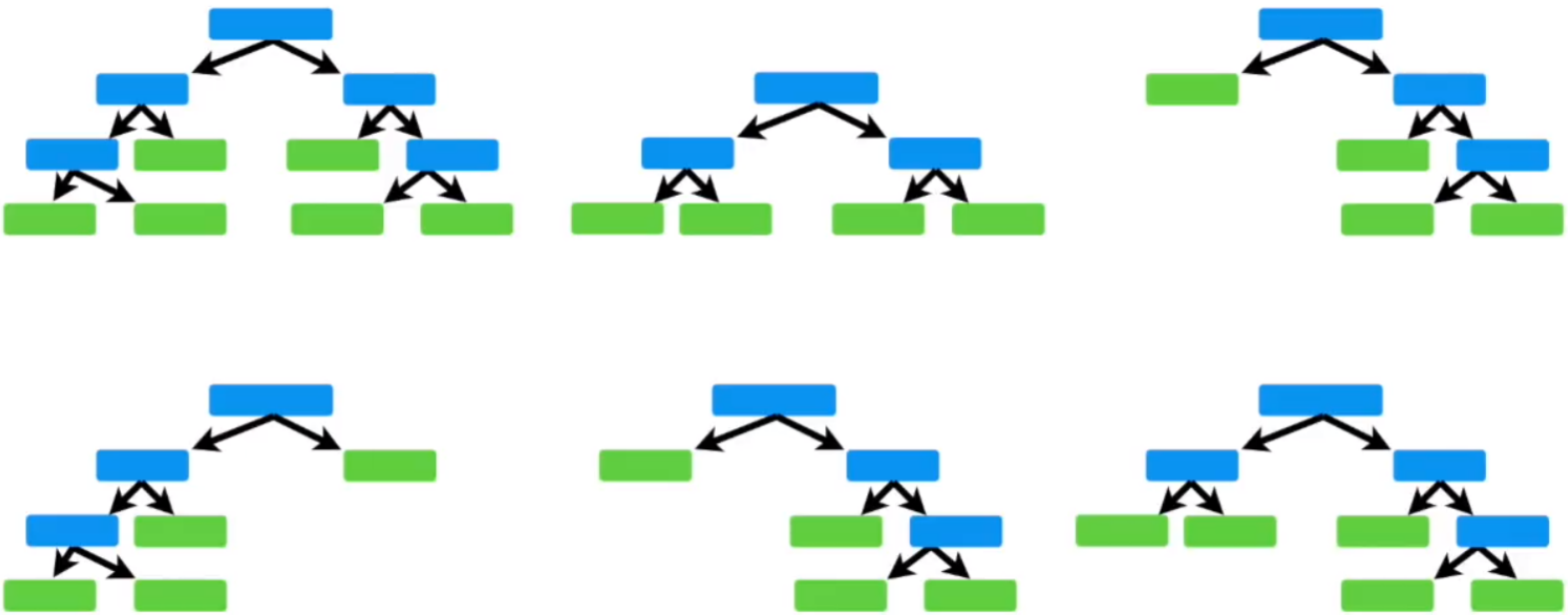
Evaluating a random forest

OK, now we've seen how to create and use a random forest...



Out-of-bag-error

How do we know if it's any good?



Out-of-bag-error

As a result, this entry was not included in the bootstrapped dataset.

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

Out-of-bag-error

Typically, about 1/3 of the original data does not end up in the bootstrapped dataset.

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

Out-of-bag-error

Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

Here is the entry that didn't end up in the bootstrapped dataset..

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	No	210	No

Out-of-bag-error

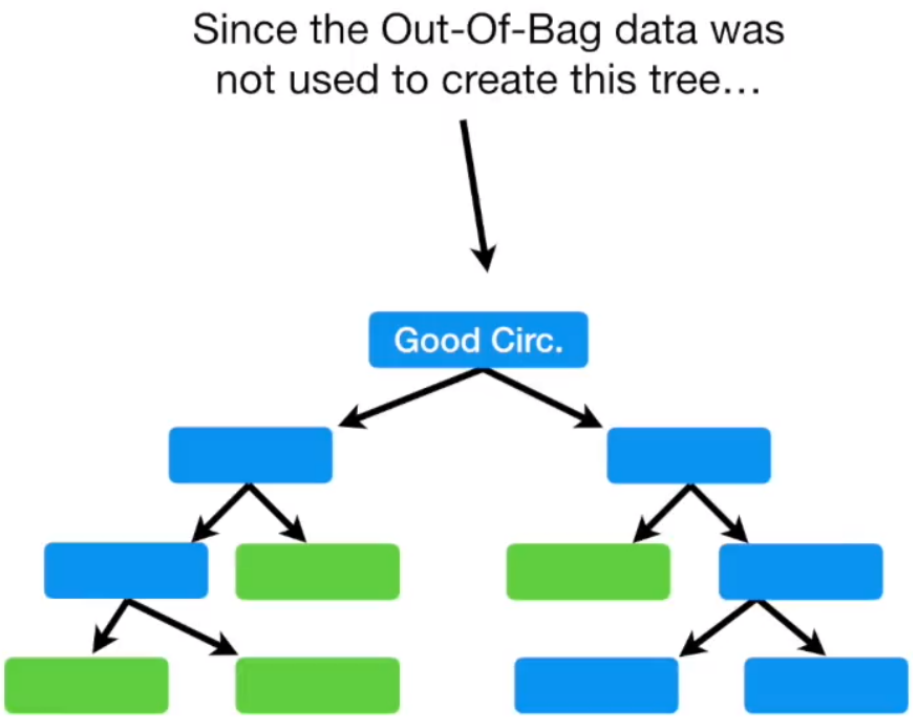
Original Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	No	Yes	167	Yes

This is called the
"Out-Of-Bag Dataset"

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	No	210	No

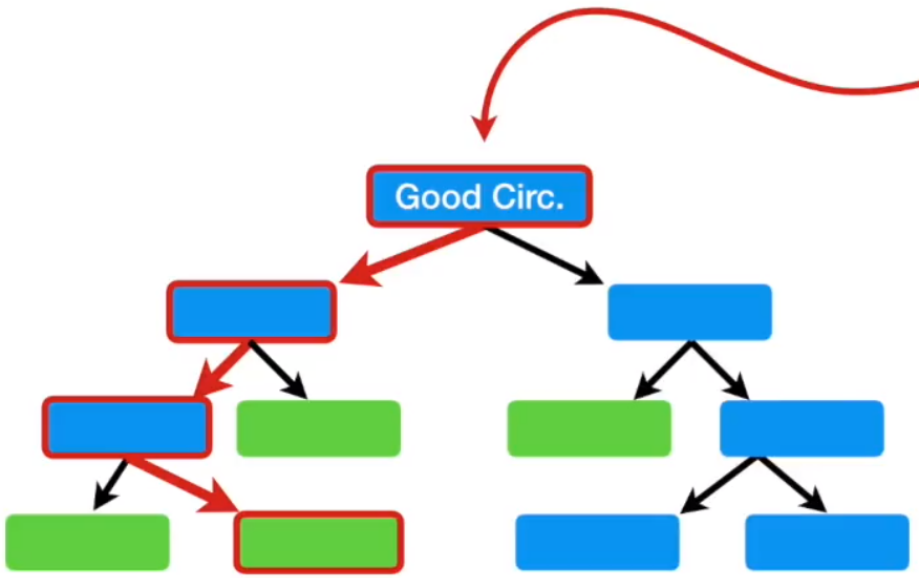
Out-of-bag-error



Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	No	210	No

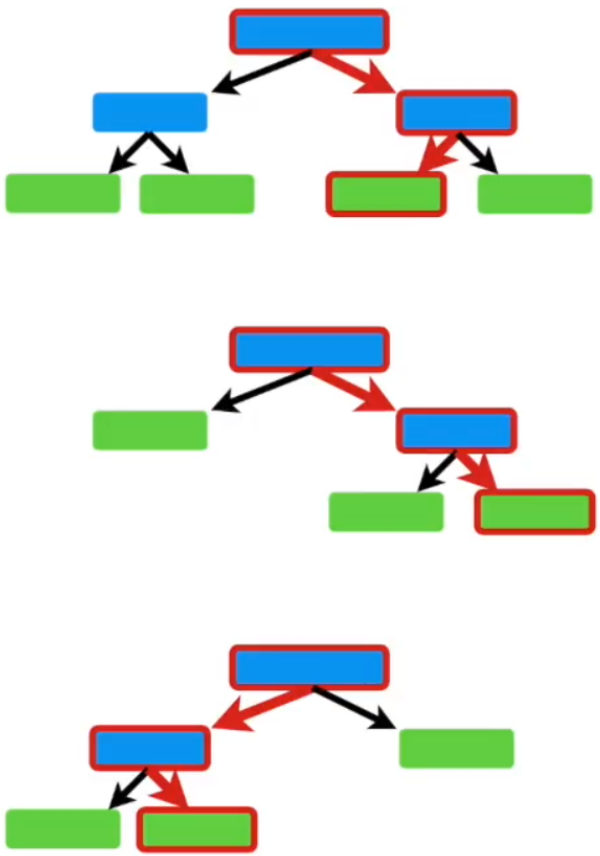
Out-of-bag-error

…we can run it through and see if it correctly classifies the sample as "No Heart Disease"



Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	No	210	No

Out-of-bag-error



Then we run this Out-Of-Bag sample through all of the other trees that were built without it...

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	No	210	No

Out-of-bag-error

Classification of the Out-Of-Bag sample	
Yes	No
1	3

Since the label with the most votes wins, it is the label that we assign this Out-of-Bag sample.

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	No	210	No

Out-of-bag-error

Classification of the Out-Of-Bag sample	
Yes	No
1	3

Classification of the Out-Of-Bag sample	
Yes	No
4	0

Classification of the Out-Of-Bag sample	
Yes	No
3	1

etc... etc... etc...

Ultimately, we can measure how accurate our random forest is by the proportion of Out-Of-Bag samples that were correctly classified by the Random Forest.

Out-of-bag-error

Classification of the Out-Of-Bag sample	
Yes	No
1	3

Classification of the Out-Of-Bag sample	
Yes	No
4	0

Classification of the Out-Of-Bag sample	
Yes	No
3	1

etc... etc... etc...

Ultimately, we can measure how accurate our random forest is by the proportion of Out-Of-Bag samples that were correctly classified by the Random Forest.

The proportion of Out-Of-Bag samples that were *incorrectly* classified is the "**Out-Of-Bag Error**"