Chula
Chulalongkorn University

DECISION TREE

Nguyen Phan Xuan Truong
Tran Nguyen Trong Phu

Content

1/ Decision Trees

2/ Pruning a Decision Tree

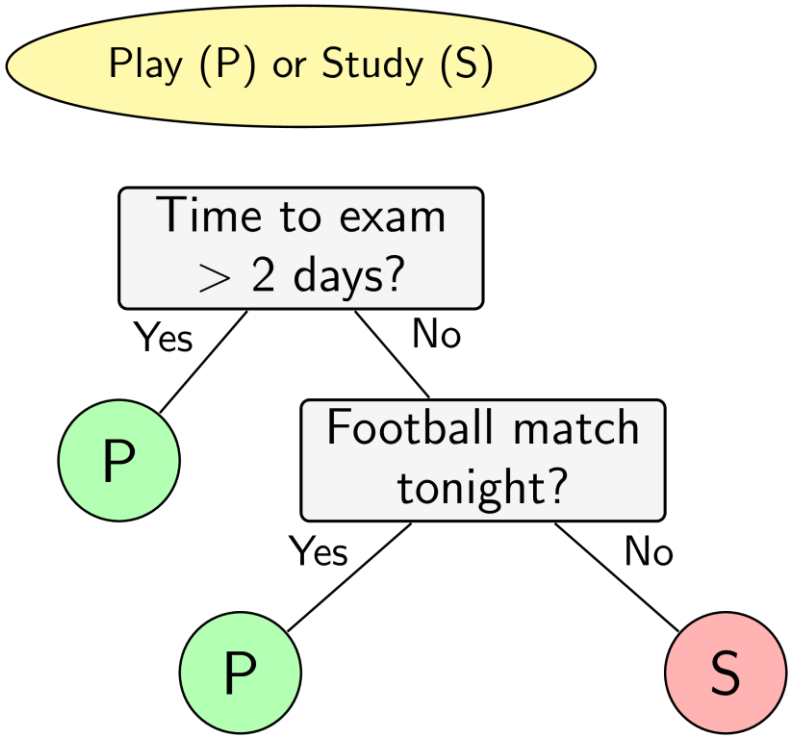
3/ Bootstrap Aggregation (Bagging) and Random Forests

4/ Boosting

5/ Lab: Decision Trees

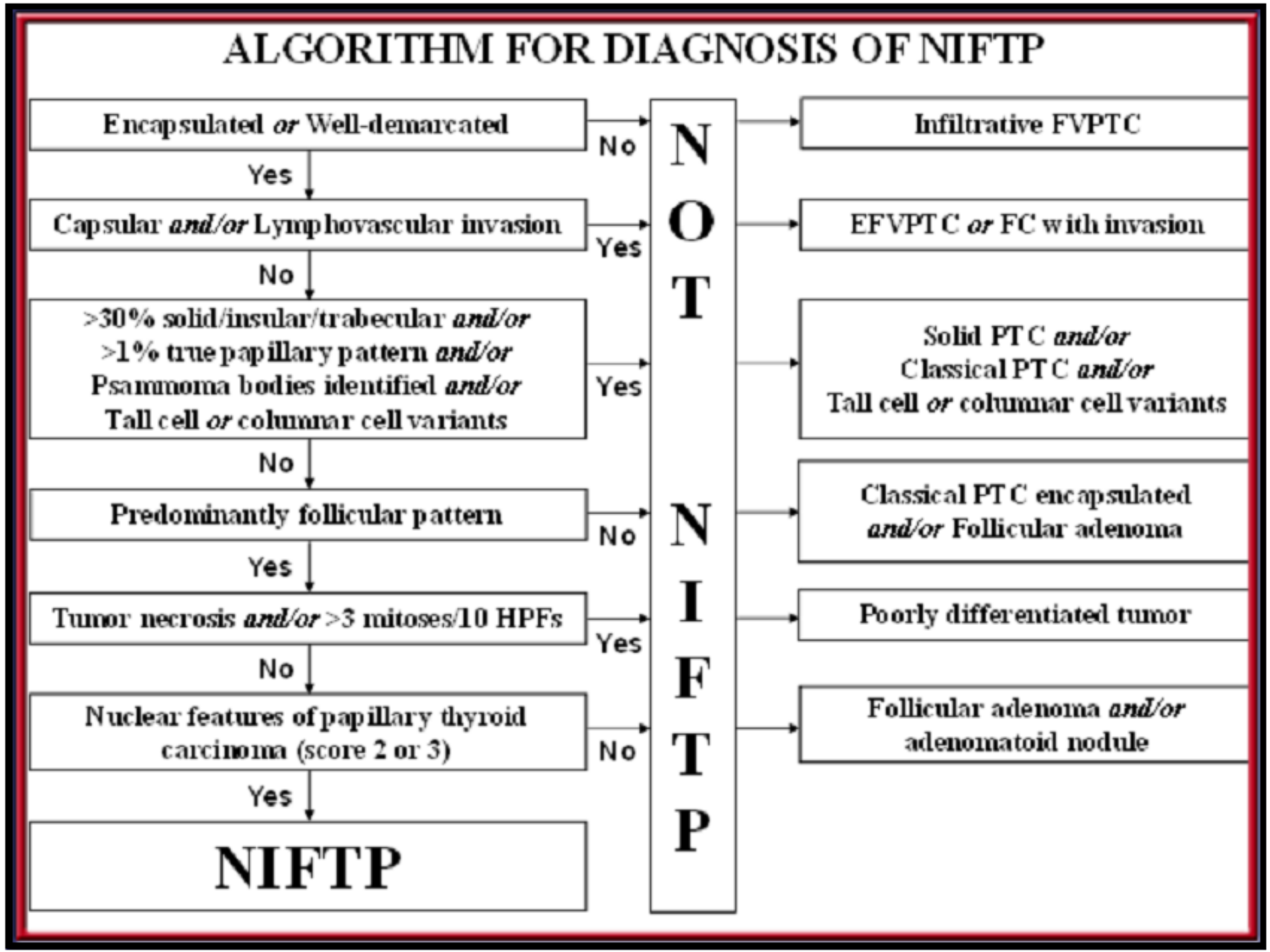
6/ Exercises

1. Decision Trees



When a Decision Tree classifies things into categories -- > Classification Tree

When a Decision predicts numeric values-- > Regression Tree


ALGORITHM FOR DIAGNOSIS OF NIFTP
Encapsulated *or* Well-demarcated
No
Infiltrative FVPTC
Yes
Capsular *and/or* Lymphovascular invasion
Yes
EFVPTC *or* FC with invasion
No
>30% solid/insular/trabecular *and/or*
>1% true papillary pattern *and/or*
Psammoma bodies identified *and/or*
Tall cell *or* columnar cell variants
Yes
Solid PTC *and/or*
Classical PTC *and/or*
Tall cell *or* columnar cell variants
No
Predominantly follicular pattern
No
Classical PTC encapsulated
and/or Follicular adenoma
Yes
Tumor necrosis *and/or* >3 mitoses/10 HPFs
Yes
Poorly differentiated tumor
No
Nuclear features of papillary thyroid carcinoma (score 2 or 3)
No
Follicular adenoma *and/or*
adenomatoid nodule
Yes
NIFTP
NOT NIFTP

Tree-based Methods

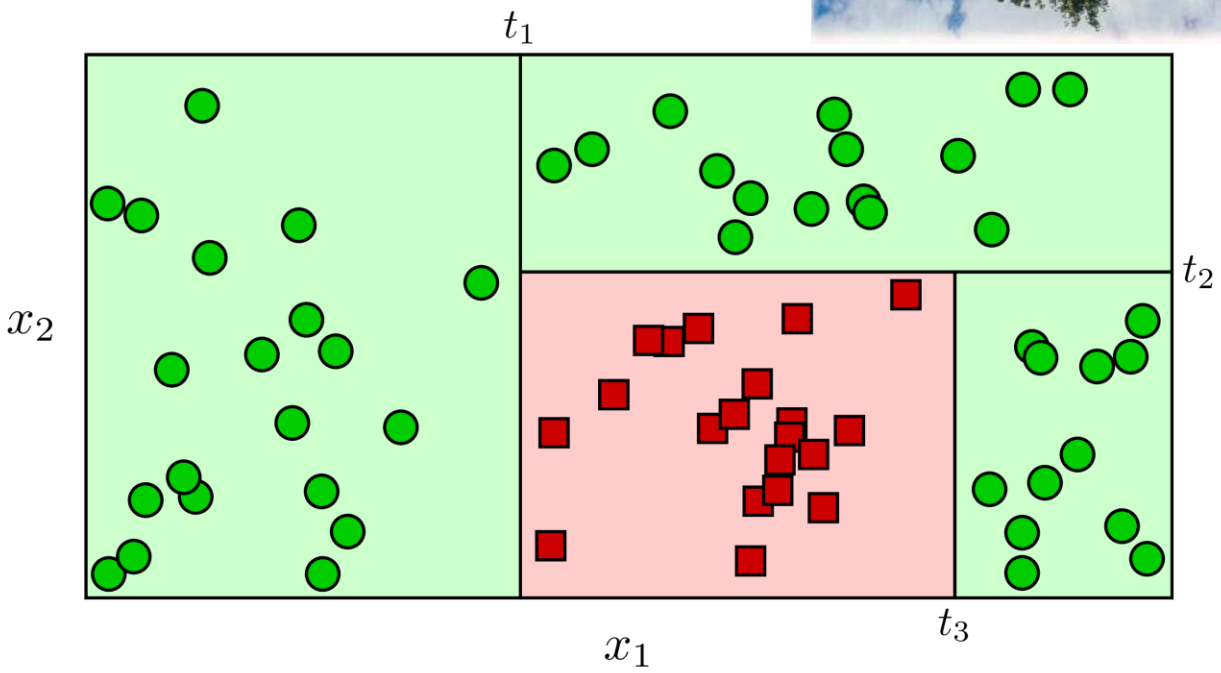
- Here we describe tree-based methods for regression and classfication
- These involve stratifying or segmenting the predictor space into a number of simple regions
- Since the set of splitting rules used to segment the predictor space can be summarized in a tree, these types of approaches are known as decision-tree methods

Advantages and Disadvantages of Trees

- ▲ Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression!
- ▲ Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches seen in previous chapters.
- ▲ Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).
- ▲ Trees can easily handle qualitative predictors without the need to create dummy variables.
- ▼ Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches seen in this book.

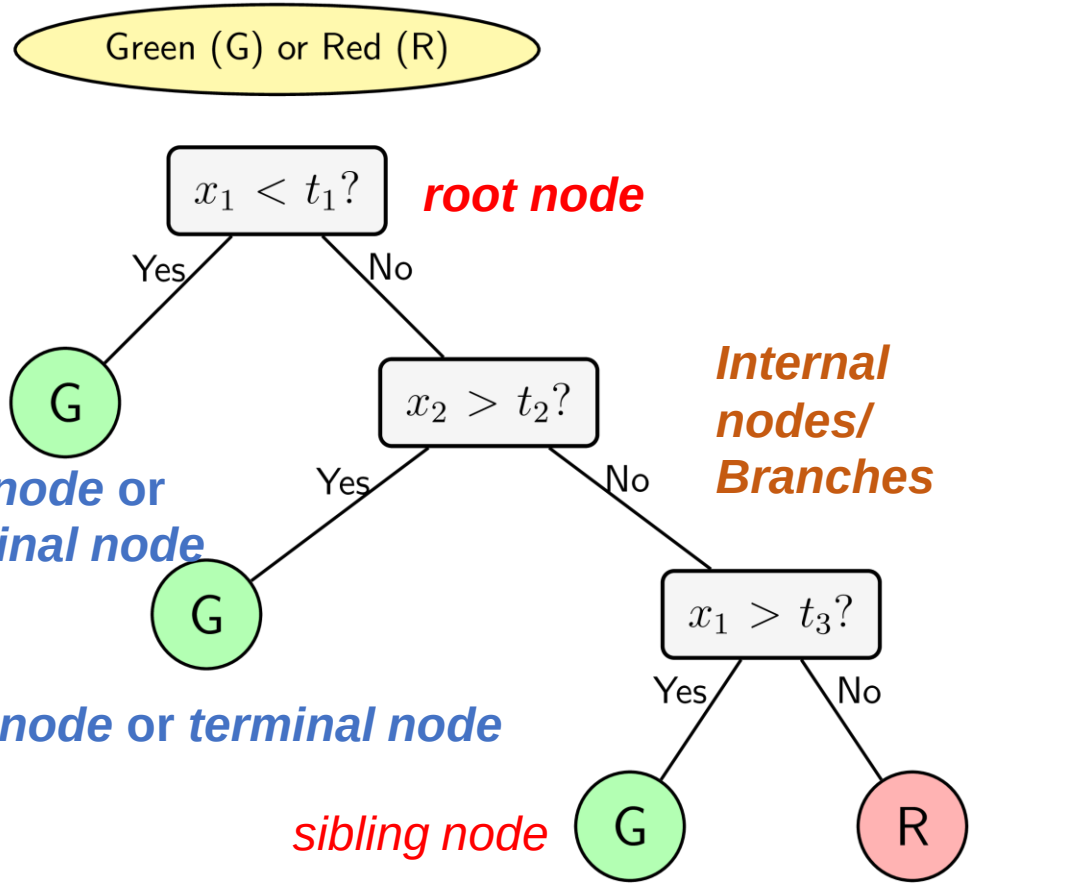
However, by aggregating many decision trees, the predictive performance of trees can be substantially improved. We introduce these concepts next.

Interpretation of Tree



(a)

Binary decision tree



(b)

Classification Tree

But probability and statistics deal with many different kinds of data, continuous and discrete, and we have dealt with at least the following during the semester:

- Real Numbers -- Height, weight, time, $X \sim N(\mu, \sigma^2)$, ….
- Integers -- Number of emails, Cards, $X \sim B(N,p)$, ….
- Binary – Heads/Tails, Red/Black, $X \sim Bern(p)$, …

Continuous

Discrete

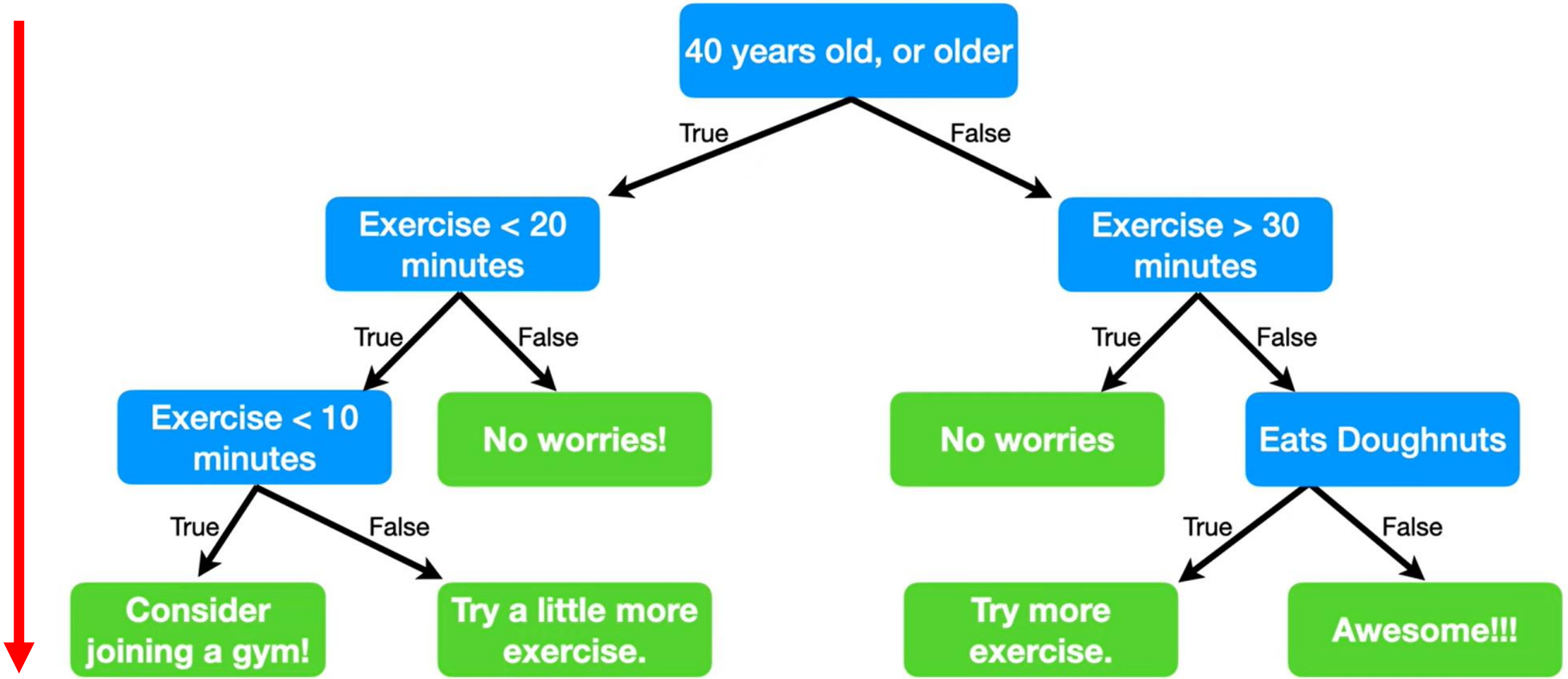
And there are other kinds of (discrete) data which statisticians must consider:

Categorical -- A finite list of unordered categories or labels, e.g.,

- Political parties (Republican, Democrat, Independent, Green. …)
- Blood type (A, B, AB, O, …)
- State lived in (MA, VT, ….)

Ordinal – Like categorical, but with an explicit ordering, e.g.,

- Class year (freshman, sophomore, …)
- Grades (A, A-, B+, …)
- Likert Scale (disagree strongly, disagree, neutral, agree, agree strongly)


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!!!

Gini index and Deviance

- The *Gini index* is defined by

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

  a measure of total variance across the K classes. The Gini index takes on a small value if all of the $\hat{p}_{mk}$'s are close to zero or one.
- For this reason the Gini index is referred to as a measure of node *purity* — a small value indicates that a node contains predominantly observations from a single class.
- An alternative to the Gini index is *cross-entropy*, given by

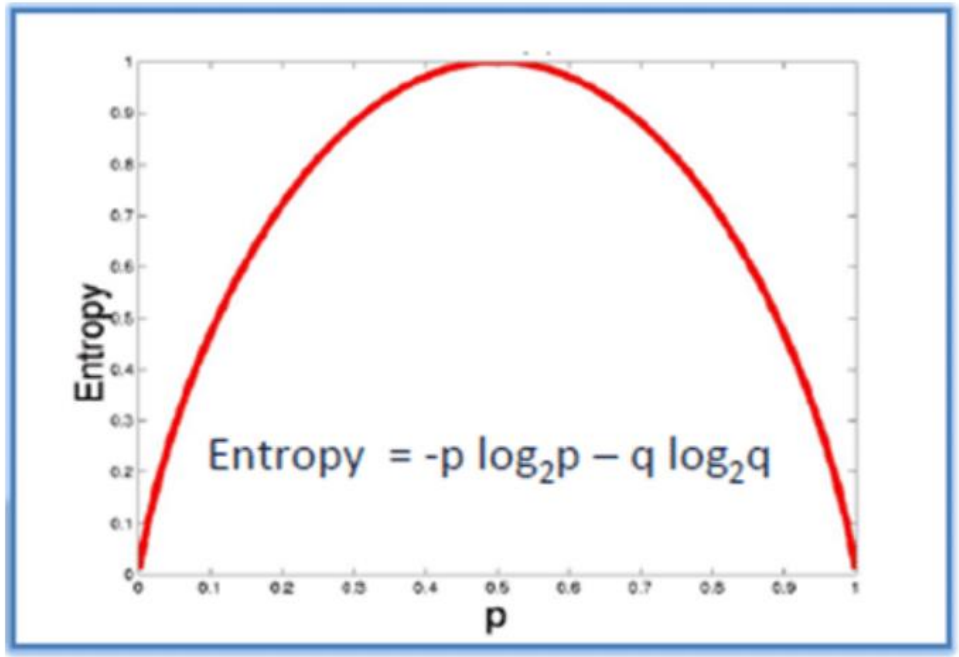
$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.$$

- It turns out that the Gini index and the cross-entropy are very similar numerically.

D= $H(p) = -\sum_{n=1}^{n} p_i \log(p_i)$

Giả sử bạn tung một đồng xu, entropy sẽ được tính như sau:

$$H = -[0.5 \ln(0.5) + 0.5 \ln(0.5)]$$



$$\text{Entropy} = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

- P tinh khiết: $p_i = 0$ hoặc $p_i = 1$
- P vẩn đục: $p_i = 0.5$, khi đó hàm Entropy đạt đỉnh cao nhất

https://www.youtube.com/watch?v=YtebGVx-Fxw&list=PLblh5JKOoLUICTaGLRoHQDuF_7q2GfuJF&index=6

https://trituenhantao.io/kien-thuc/decision-tree/

Để xác định các nút trong mô hình cây quyết định, ta thực hiện tính Infomation Gain tại mỗi nút theo trình tự sau:

•**Bước 1**: Tính toán hệ số Entropy của biến mục tiêu S có N phần tử với N_c phần tử thuộc lớp c cho trước:

$$H(S) = -\sum_{c=1}^{C} (N_c/N) \log(N_c/N)$$

•**Bước 2**: Tính hàm số Entropy tại mỗi thuộc tính: với thuộc tính x, các điểm dữ liệu trong S được chia ra K child node $S_1, S_2, ..., S_K$ với số điểm trong mỗi child node lần lượt là $m_1, m_2, ..., m_K$, ta có:

$$H(x, S) = \sum_{k=1}^{K} (m_k / N) * H(S_k)$$

Bước 3: Chỉ số Gain Information được tính bằng:

$$\mathbf{G(x, S) = H(S) - H(x,S)}$$

Trong ID3, tại mỗi node, thuộc tính được chọn được xác định dựa trên:

$$x^* = \arg\max_x G(x, \mathcal{S}) = \arg\min_x H(x, \mathcal{S})$$

tức thuộc tính khiến cho *information gain* đạt giá trị lớn nhất.

https://trituenhantao.io/kien-thuc/decision-tree/

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

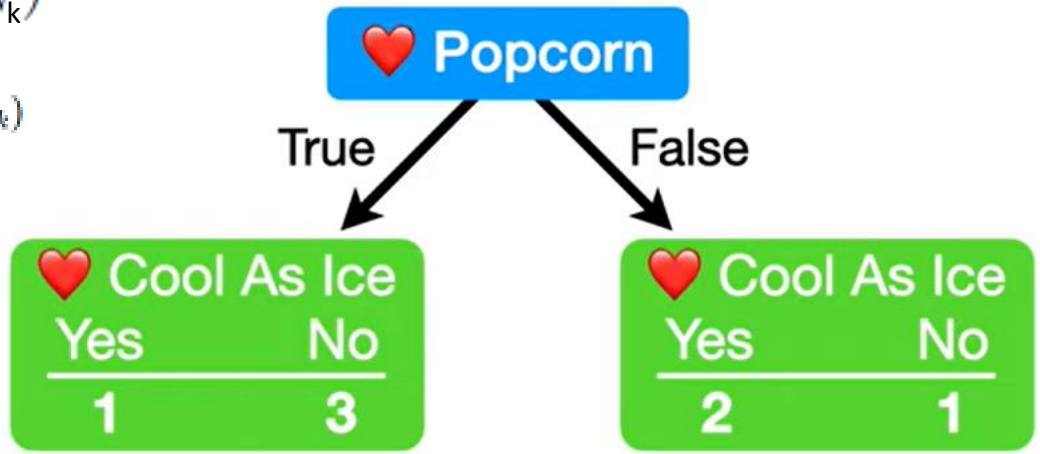
https://www.youtube.com/watch?v=_L39rN6gz7Y&list=PLblh5JKOoLUICTaGLRoHQDuF_7q2GfuJF&index=43

Example

$$H(\mathcal{S}_k) = -\sum_{c=1}^{C} \frac{N_c}{N_k} \log\left(\frac{N_c}{N_k}\right)$$

$$H(x, \mathcal{S}) = \sum_{k=1}^{K} \frac{m_k}{N} H(\mathcal{S}_k)$$

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



$$H(True) = -\frac{1}{4}\log\frac{1}{4} - \frac{3}{4}\log\frac{3}{4} = 0.2442191$$

$$H(False) = -\frac{2}{3}\log\frac{2}{3} - \frac{1}{3}\log\frac{1}{3} = 0.2764346$$

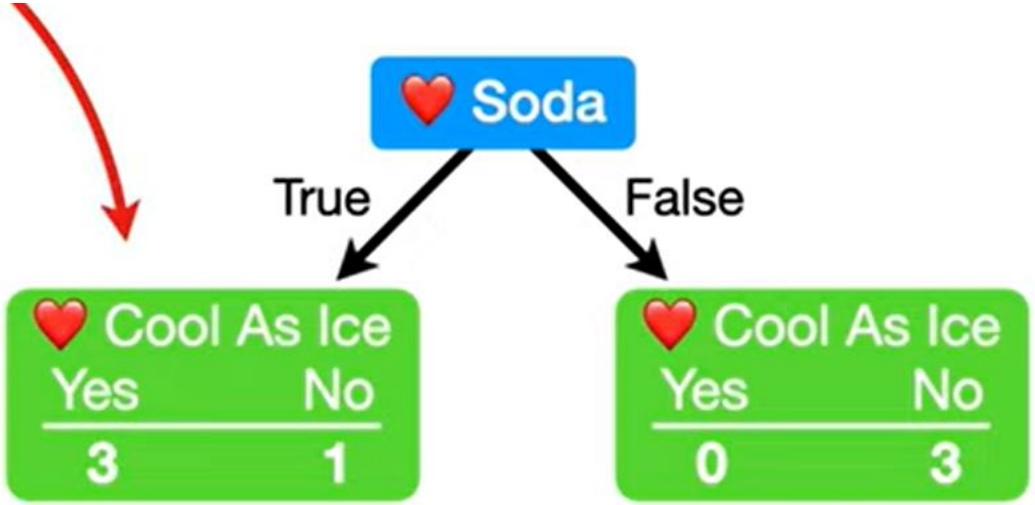
$$H(Popcorn, S) = \frac{4}{7}x\ H(True) + \frac{3}{7}x\ H(False) = 0.2580257$$

Example

$$H(\mathcal{S}_k) = -\sum_{c=1}^{C} \frac{N_c}{N_k} \log\left(\frac{N_c}{N_k}\right)$$

$$H(x, \mathcal{S}) = \sum_{k=1}^{K} \frac{m_k}{N} H(\mathcal{S}_k)$$

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



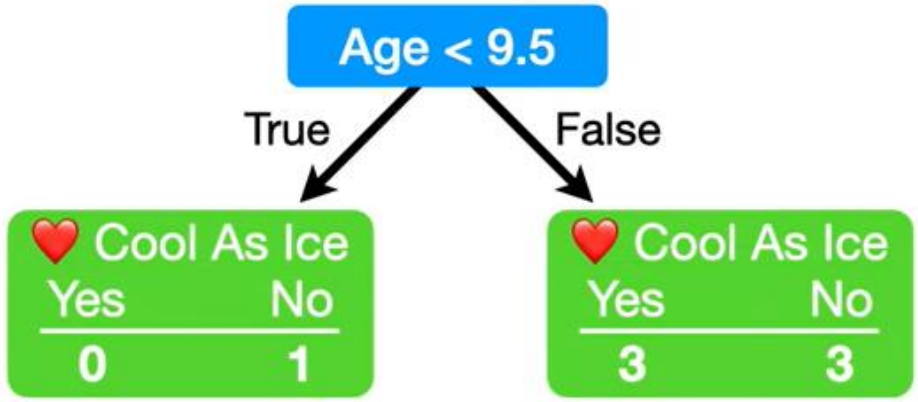
Example

$$H(\mathcal{S}_k) = -\sum_{c=1}^{C} \frac{N_c}{N_k} \log\left(\frac{N_c}{N_k}\right)$$

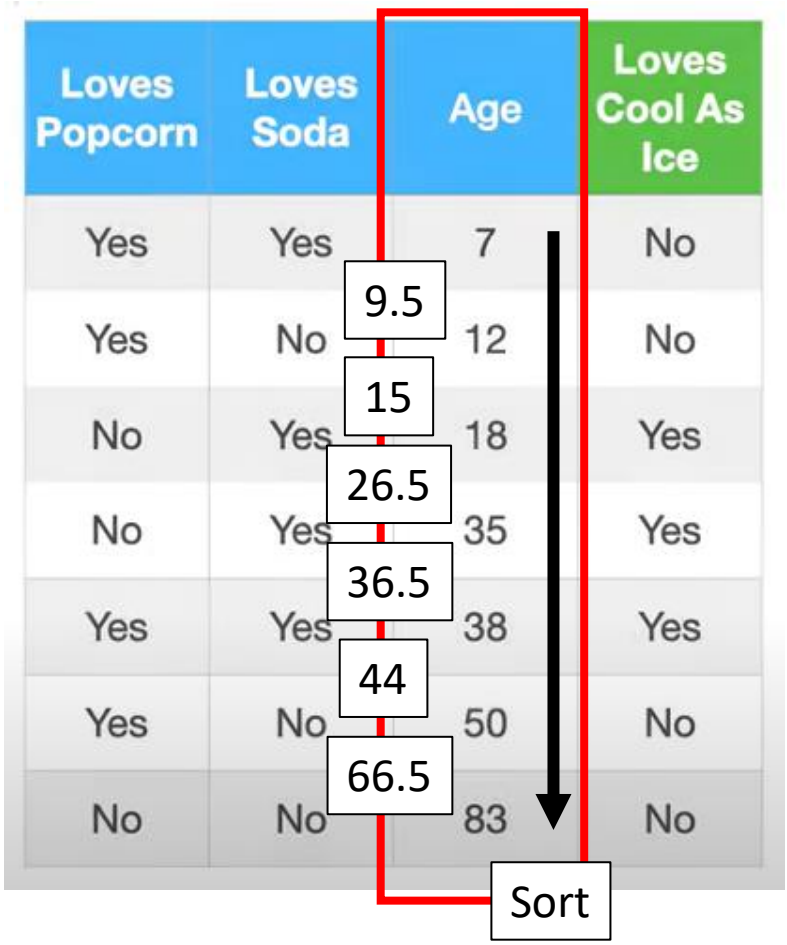
$$H(x, \mathcal{S}) = \sum_{k=1}^{K} \frac{m_k}{N} H(\mathcal{S}_k)$$

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

9.5
15
26.5
36.5
44
66.5
Sort



Example



$H(Age < 9.5, S) = 0.2580257$

$H(Age < 15, S) = 0.2087752$

$H(Age < 26.5, S) = 0.2904891$

$H(Age < 36.5, S) = 0.2904891$

$H(Age < 44, S) = 0.2087752$

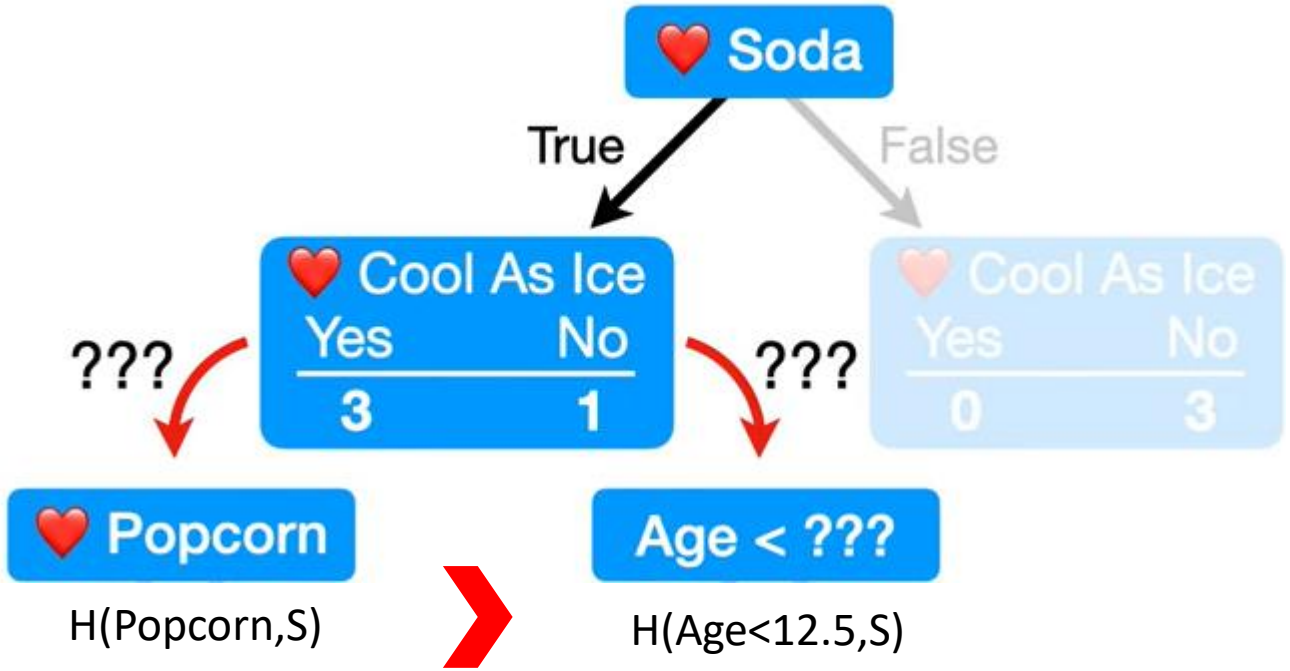
$H(Age < 66.5, S) = 0.2580257$

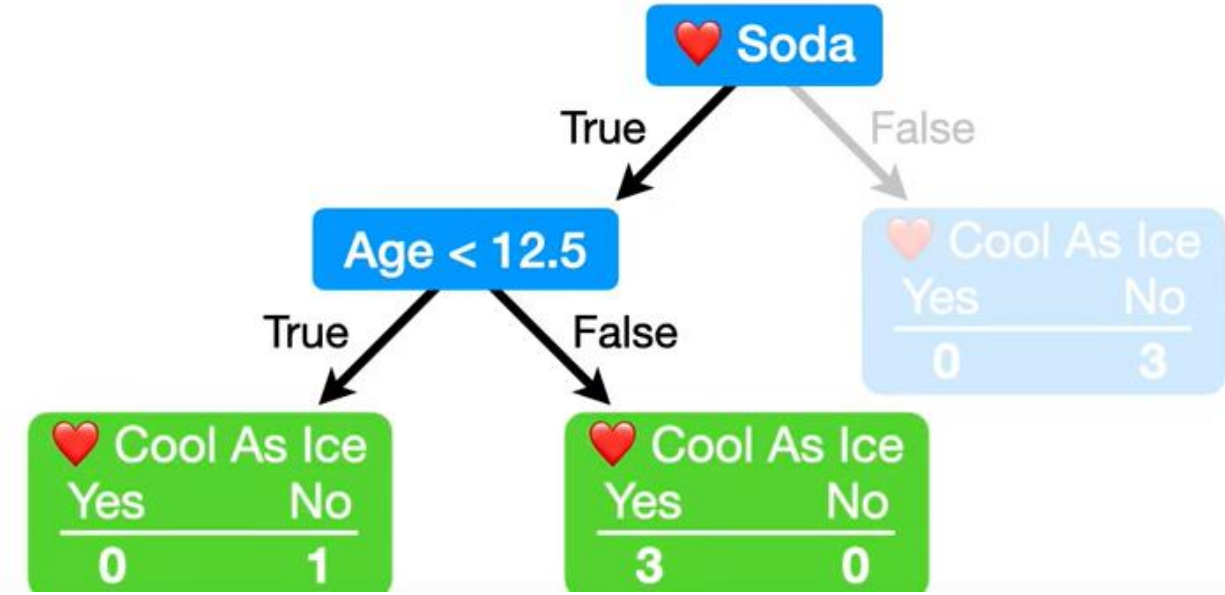
$H(Popcorn, S) = 0.2580257$

$H(Soda, S) = 0.1395538$

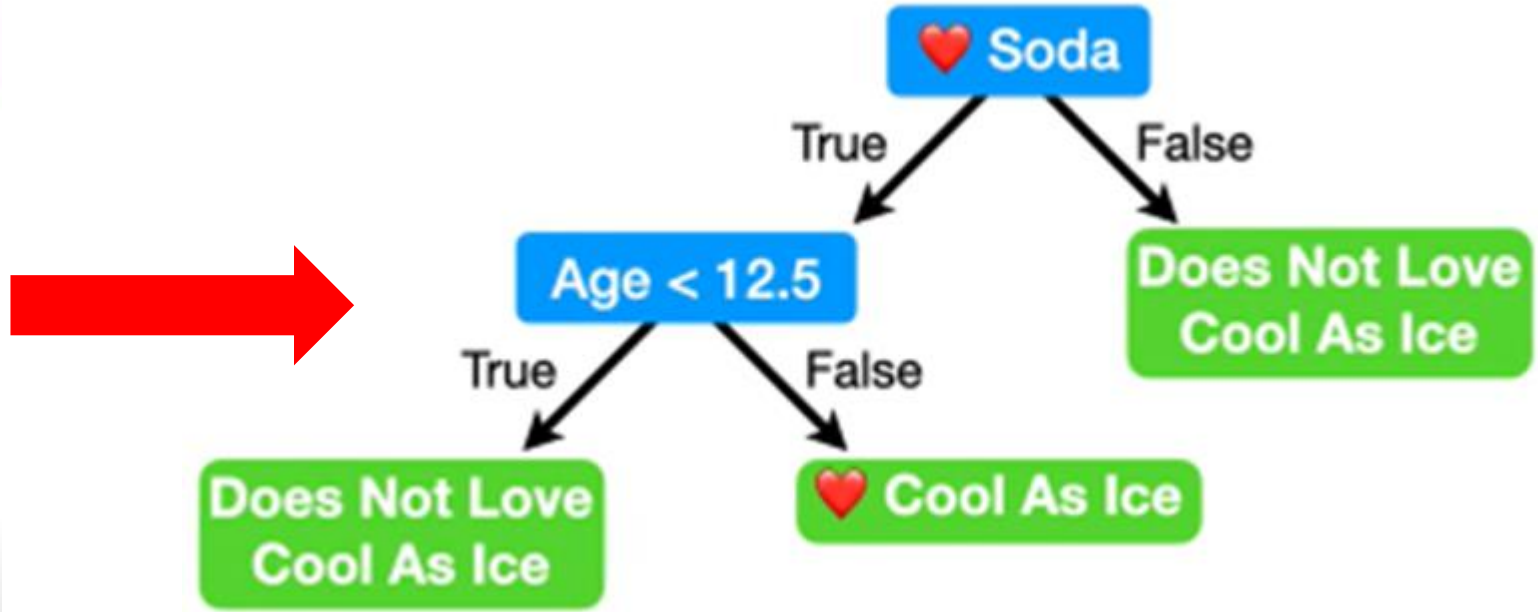
$H(Age < 15, S) = 0.2087752$

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



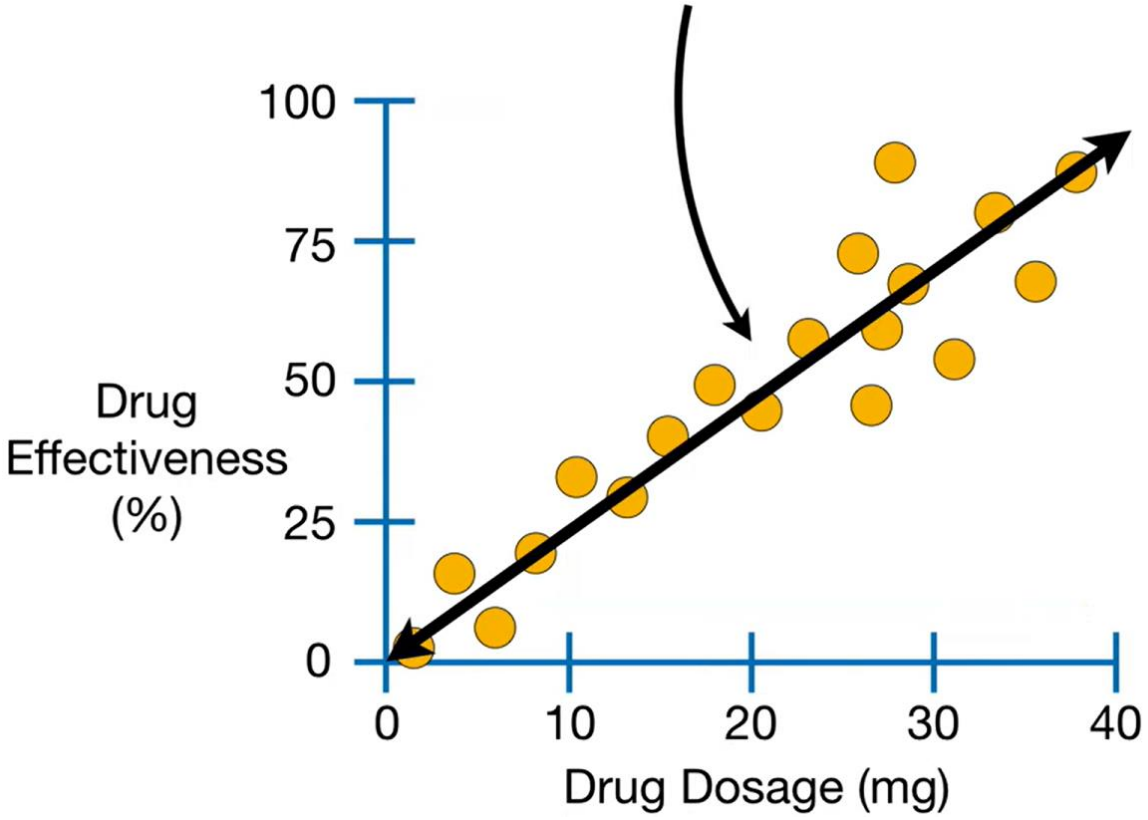


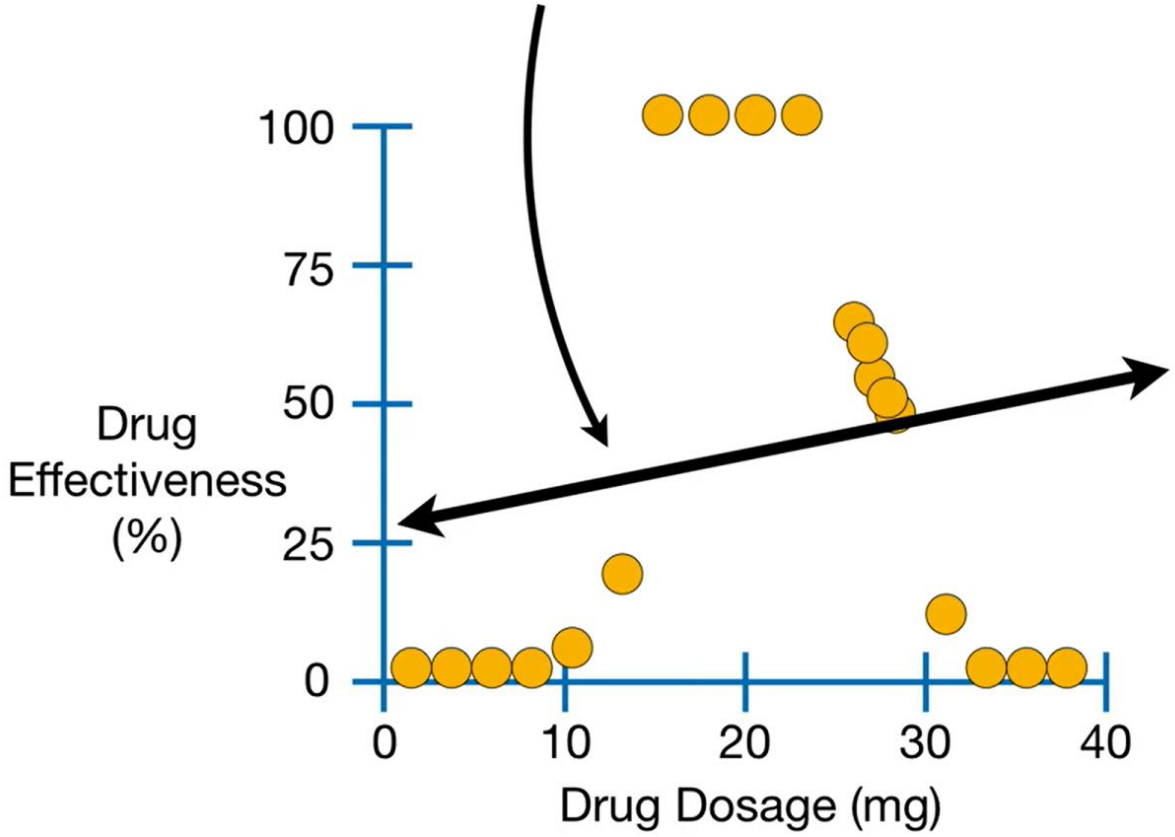
Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



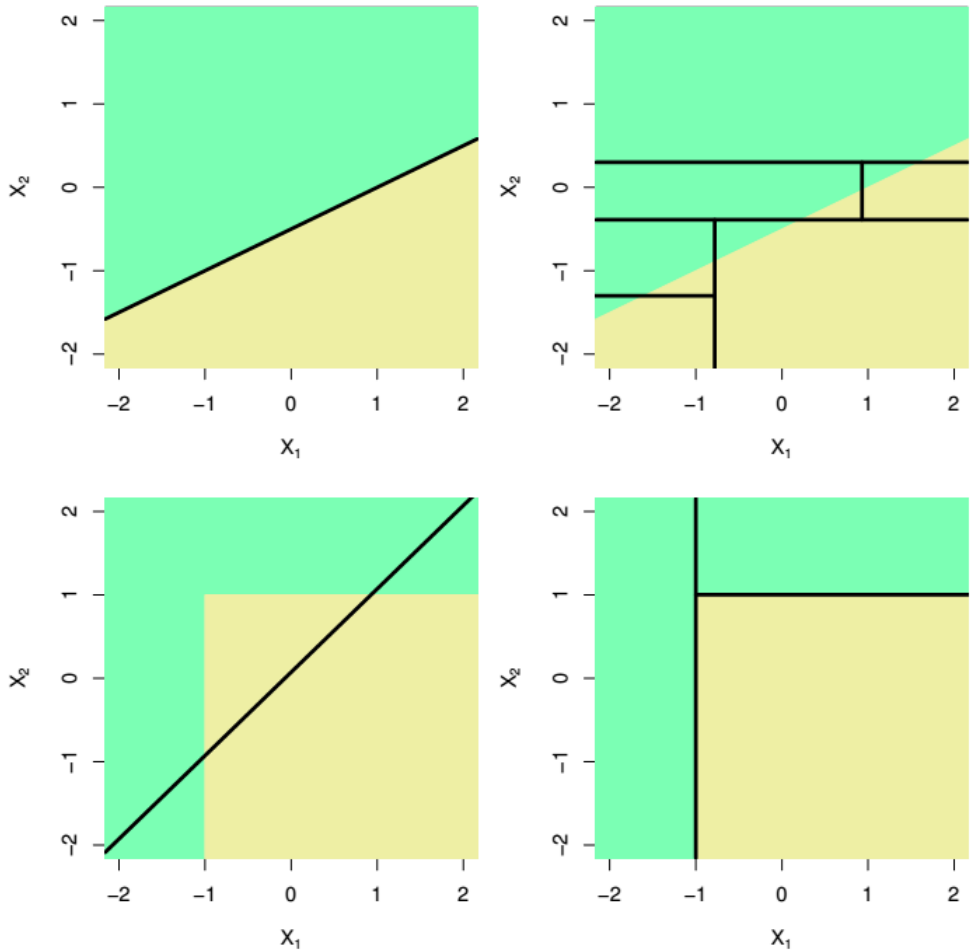
Regression Tree

Linear Model





Trees Versus Linear Models

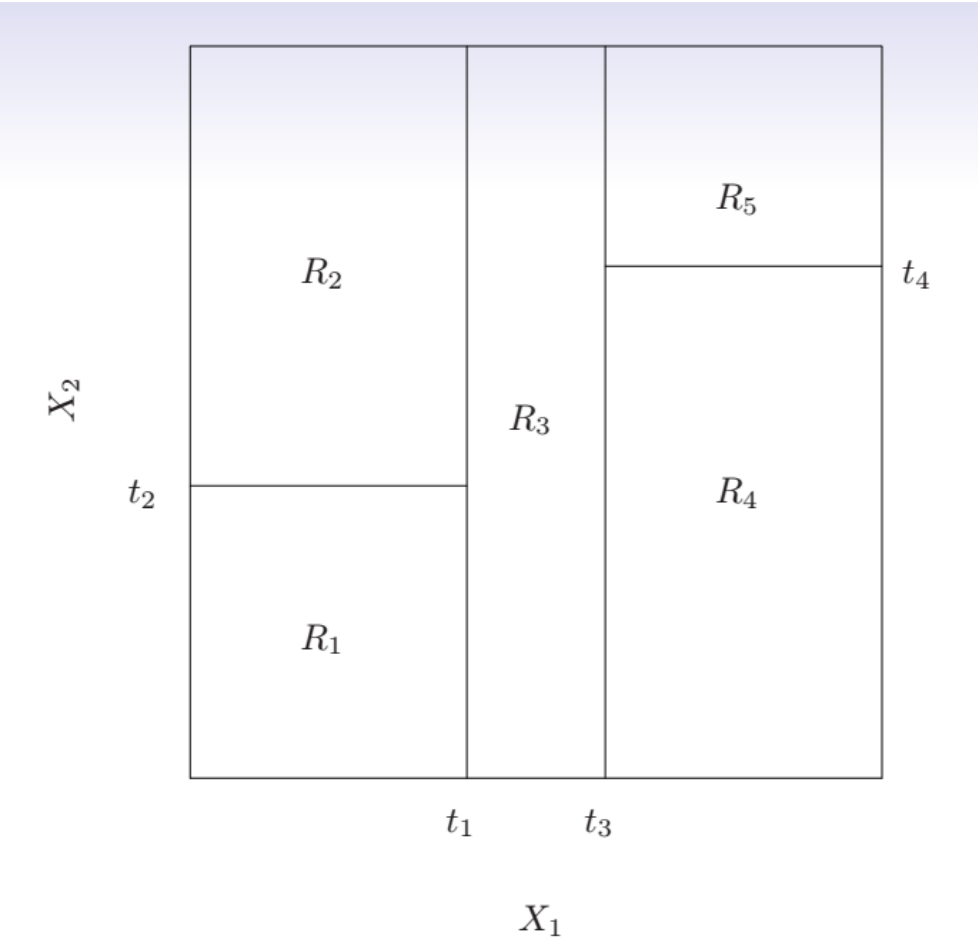


Top Row: True linear boundary; Bottom row: true non-linear boundary.

Left column: linear model; Right column: tree-based model

Details of the tree-building process

- We divide the predictor space - that is, the set of possible values for X1; X2; …. ; Xp| into J distinct and non-overlapping regions, R1; R2;…; RJ
- For every observation that falls into the region Rj, we make the same prediction, which is simply the mean of the response values for the training observations in Rj

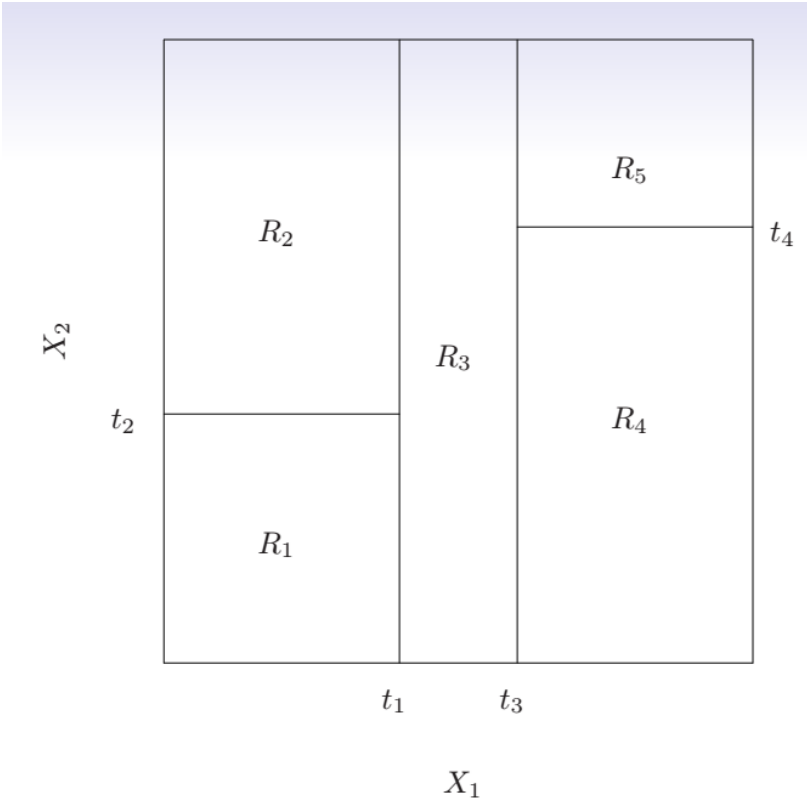


Details of the tree-building process

- In theory, the regions could have any shape. However, we choose to divide the predictor space into high-dimensional rectangles, or *boxes*, for simplicity and for ease of interpretation of the resulting predictive model.
- The goal is to find boxes $R_1, \ldots, R_J$ that minimize the SSR , given by

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2,$$

where $\hat{y}_{R_j}$ is the mean response for the training observations within the jth box.



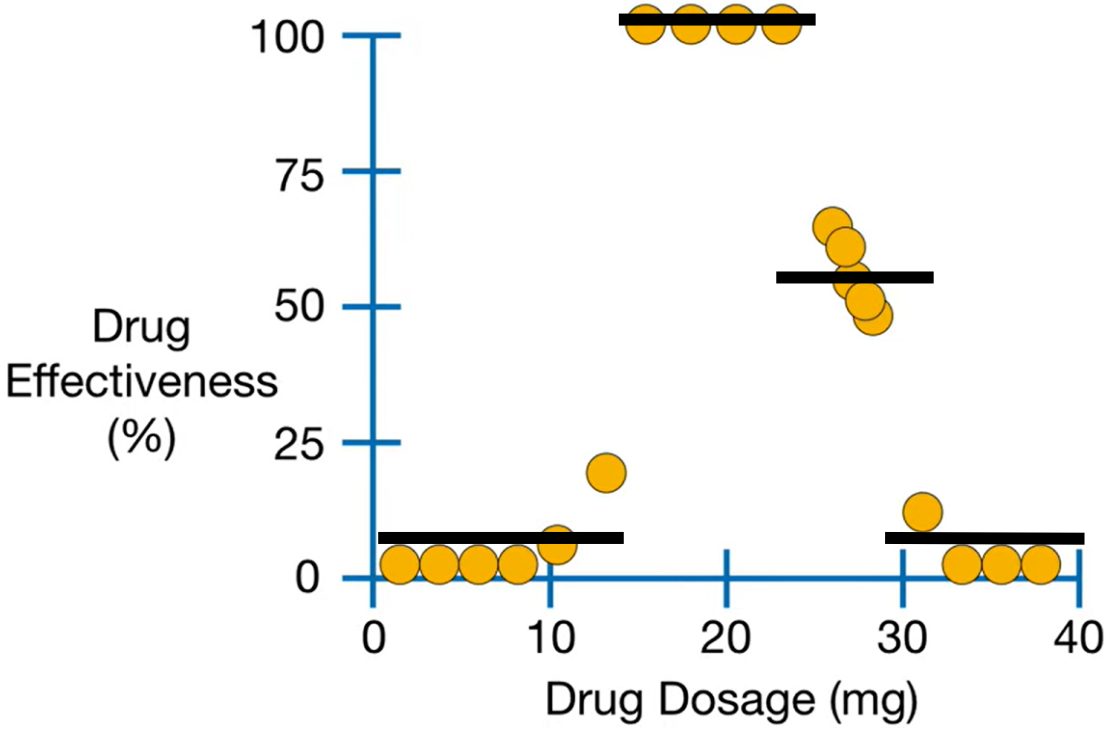
More details of the tree-building process

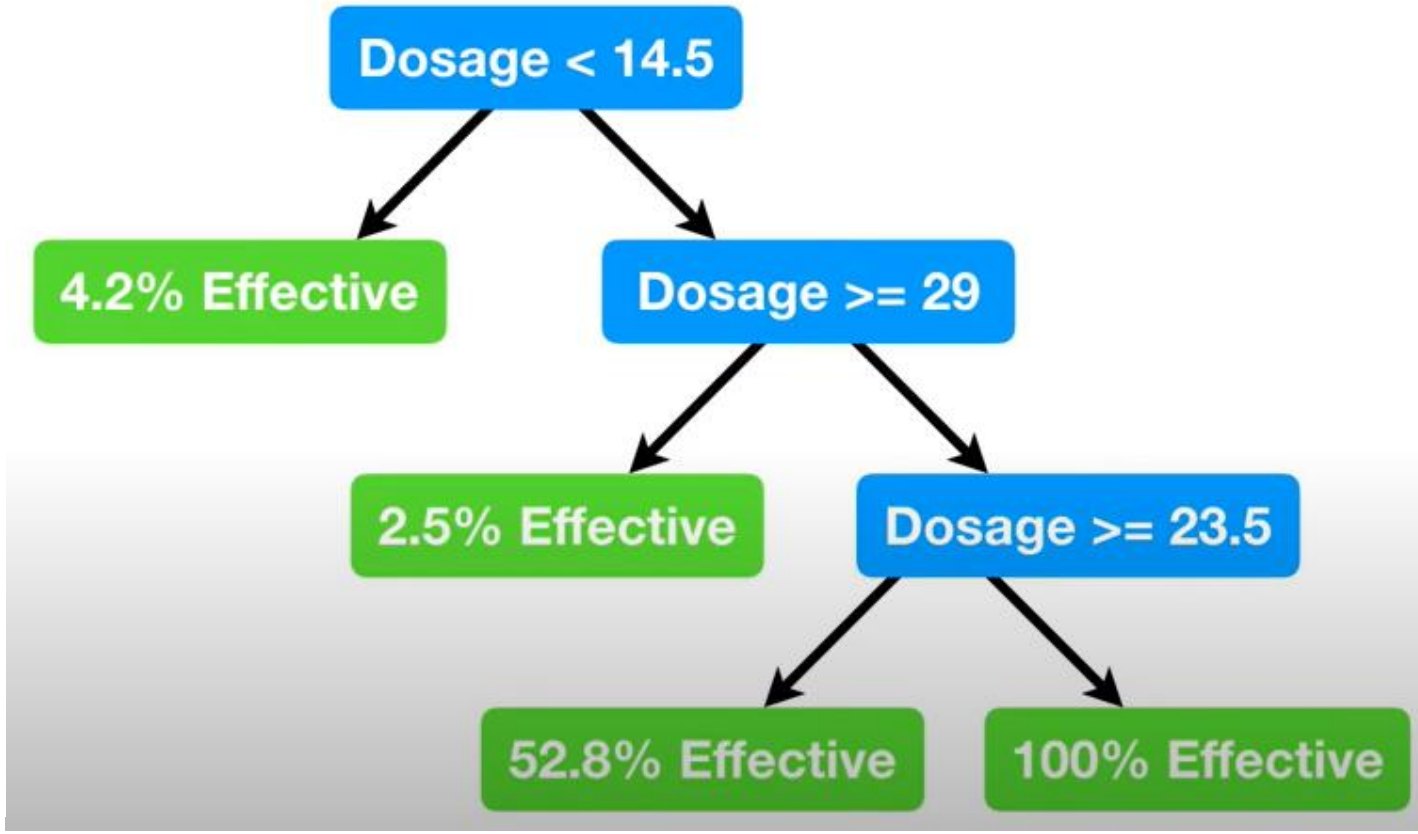
- Unfortunately, it is computationally infeasible to consider every possible partition of the feature space into J boxes.
- For this reason, we take a *top-down*, *greedy* approach that is known as recursive binary splitting.
- The approach is *top-down* because it begins at the top of the tree and then successively splits the predictor space; each split is indicated via two new branches further down on the tree.
- It is *greedy* because at each step of the tree-building process, the *best* split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.

Details— Continued

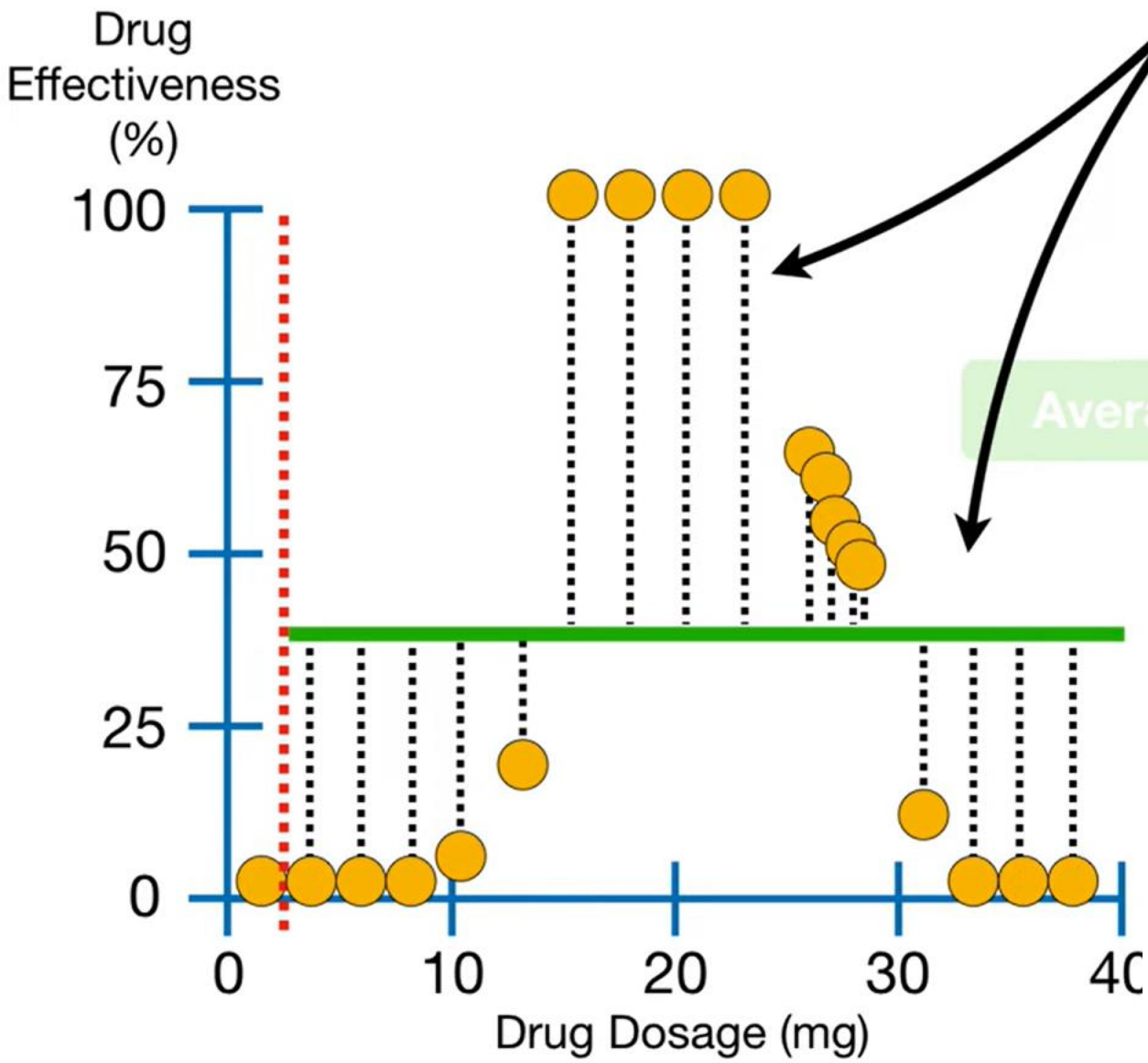
- We first select the predictor X_j and the cutpoint s such that splitting the predictor space into the regions $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$ leads to the greatest possible reduction in SSR
- Next, we repeat the process, looking for the best predictor and best cutpoint in order to split the data further so as to minimize the SSR within each of the resulting regions.
- However, this time, instead of splitting the entire predictor space, we split one of the two previously identified regions. We now have three regions.
- Again, we look to split one of these three regions further, so as to minimize the SSR The process continues until a stopping criterion is reached; for instance, we may continue until no region contains more than five observations.

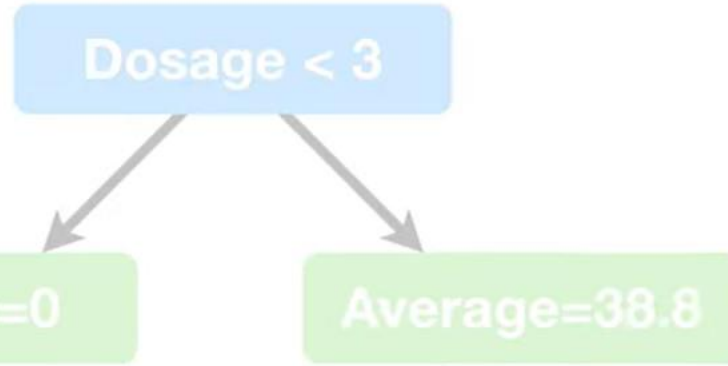
Ex:

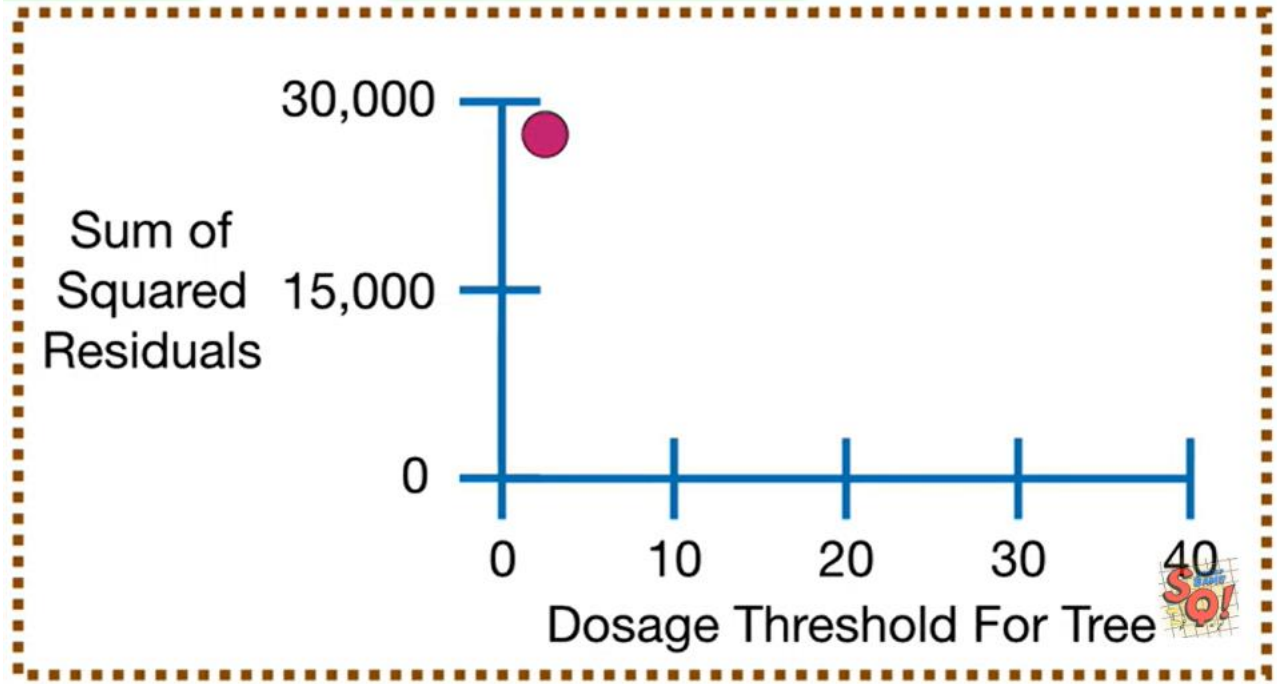


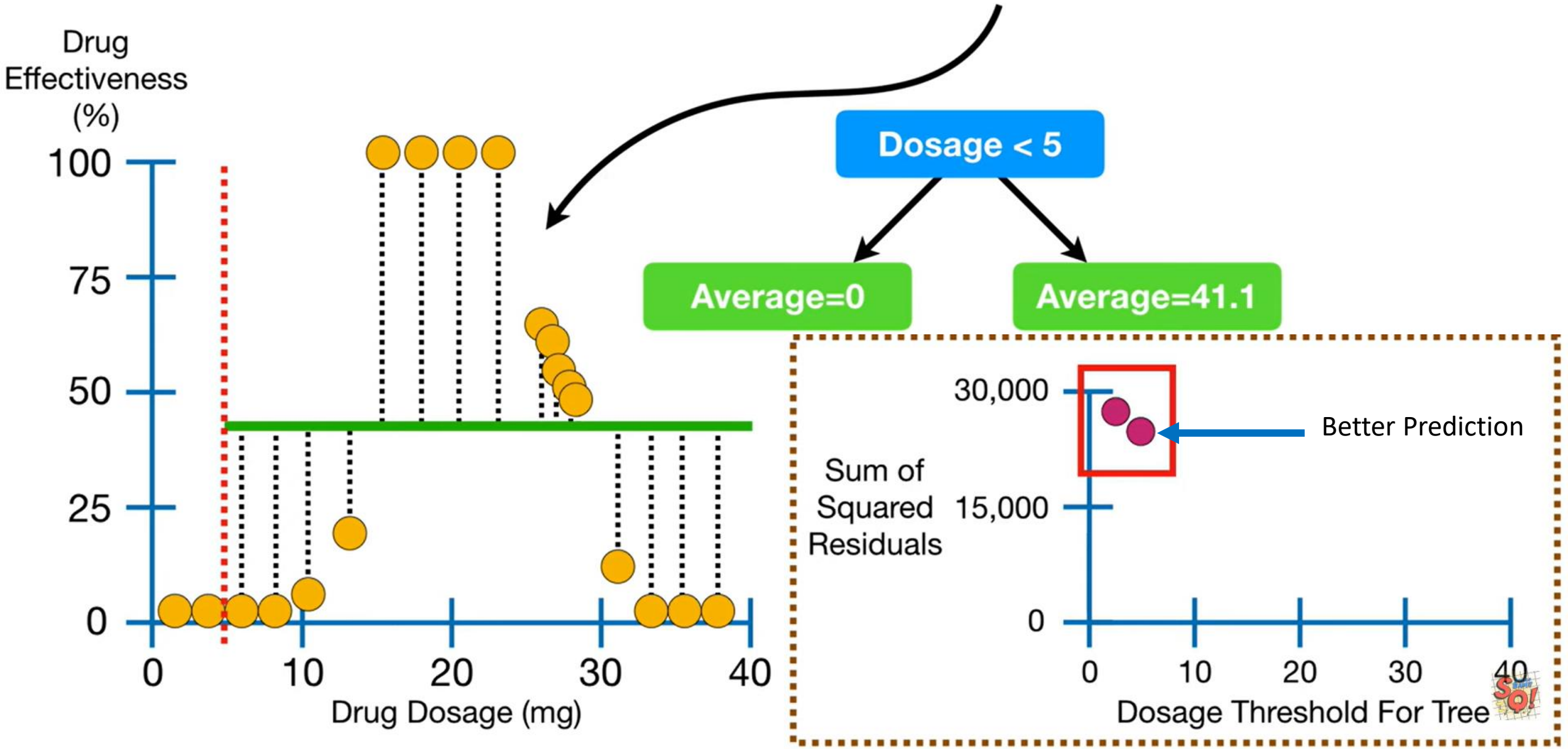


https://www.youtube.com/watch?v=g9c66TUylZ4&list=PLblh5JKOoLUICTaGLRoHQDuF_7q2GfuJF&index=46

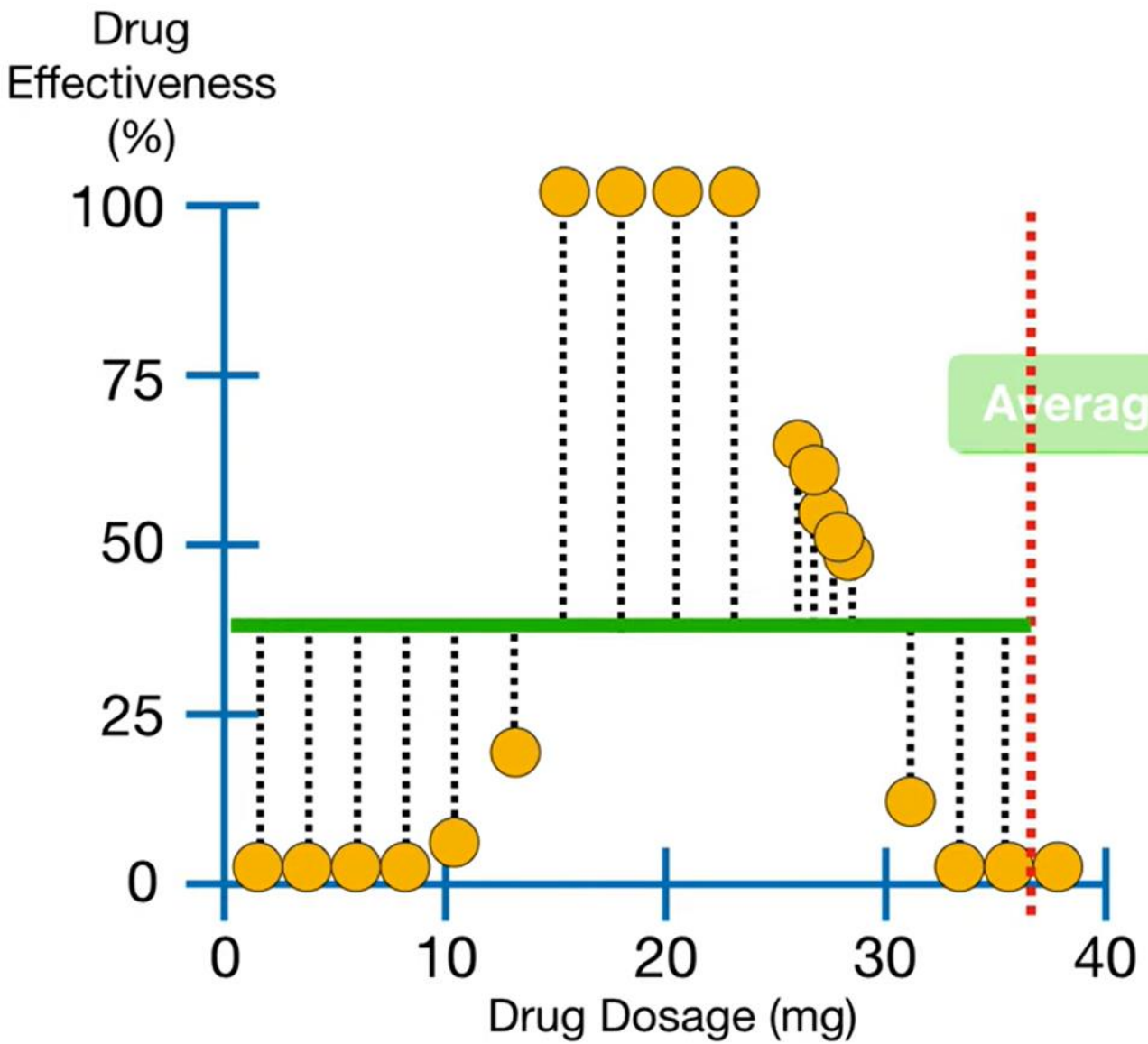

Drug Effectiveness (%)
100
75
50
25
0
0
10
20
30
40
Drug Dosage (mg)

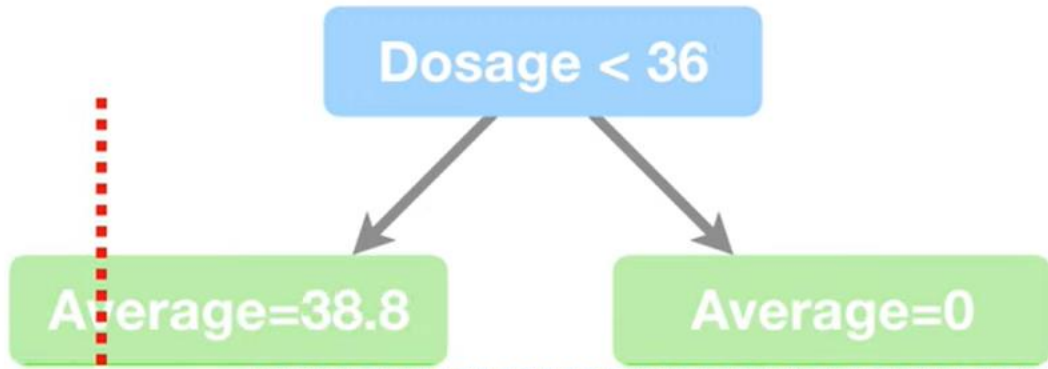

Dosage < 3
Average=0
Average=38.8

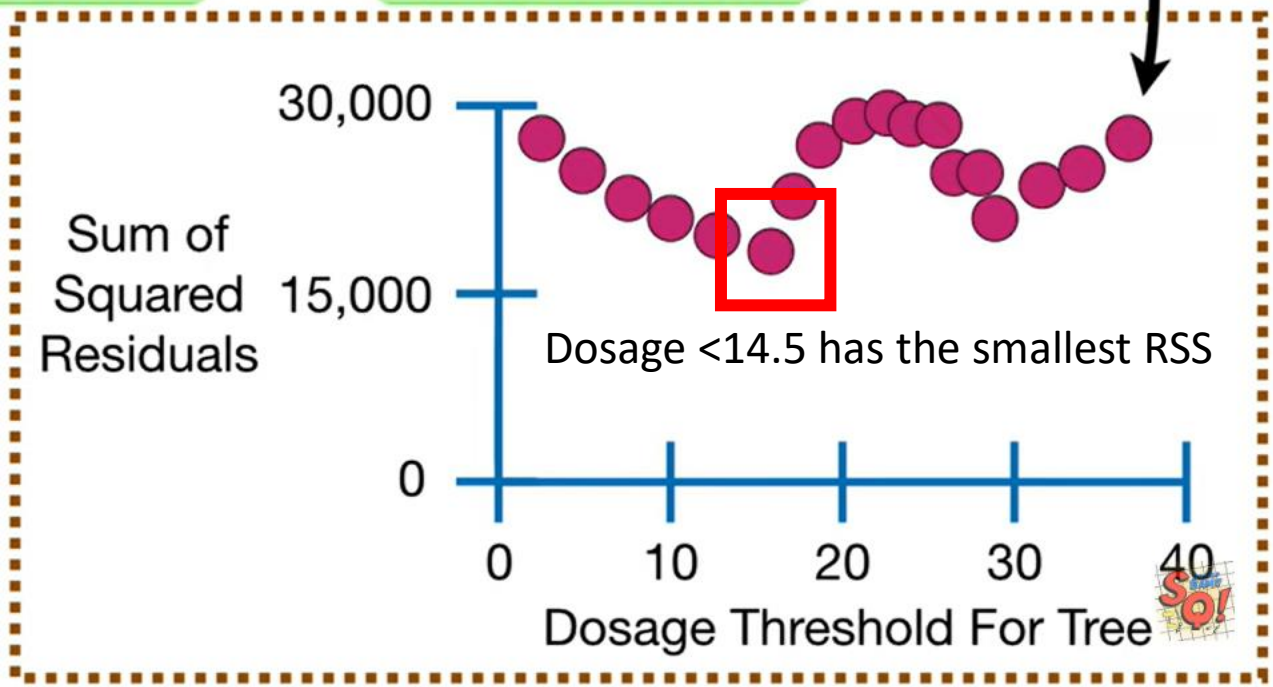

Sum of Squared Residuals
30,000
15,000
0
0
10
20
30
40
Dosage Threshold For Tree

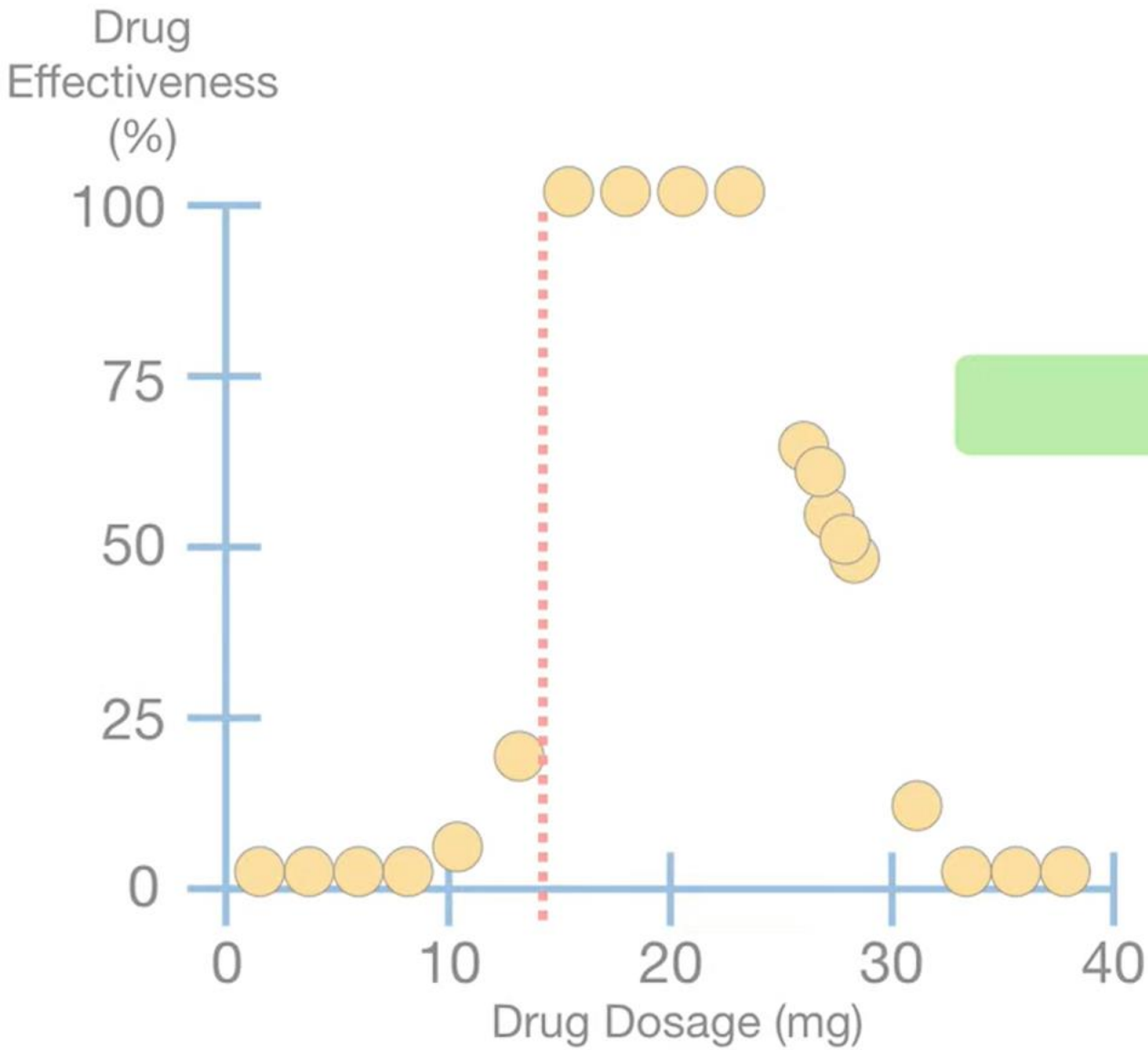

Drug Effectiveness (%)
100
75
50
25
0
0
10
20
30
40
Drug Dosage (mg)
Dosage < 5
Average=0
Average=41.1
Sum of Squared Residuals
30,000
15,000
0
Better Prediction
0
10
20
30
40
Dosage Threshold For Tree

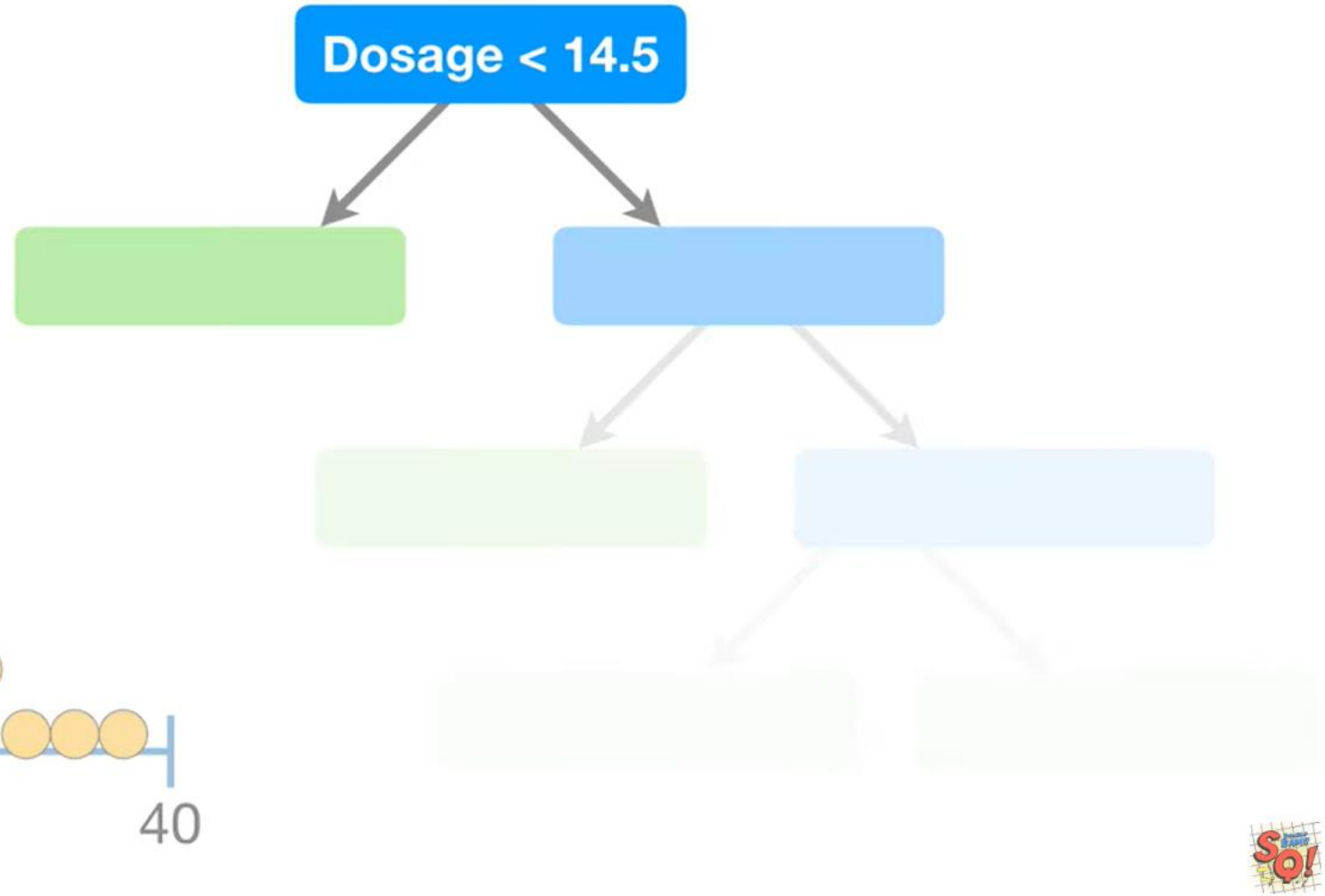
And we repeat until we have calculated the sum of squared residuals for all of the remaining thresholds.

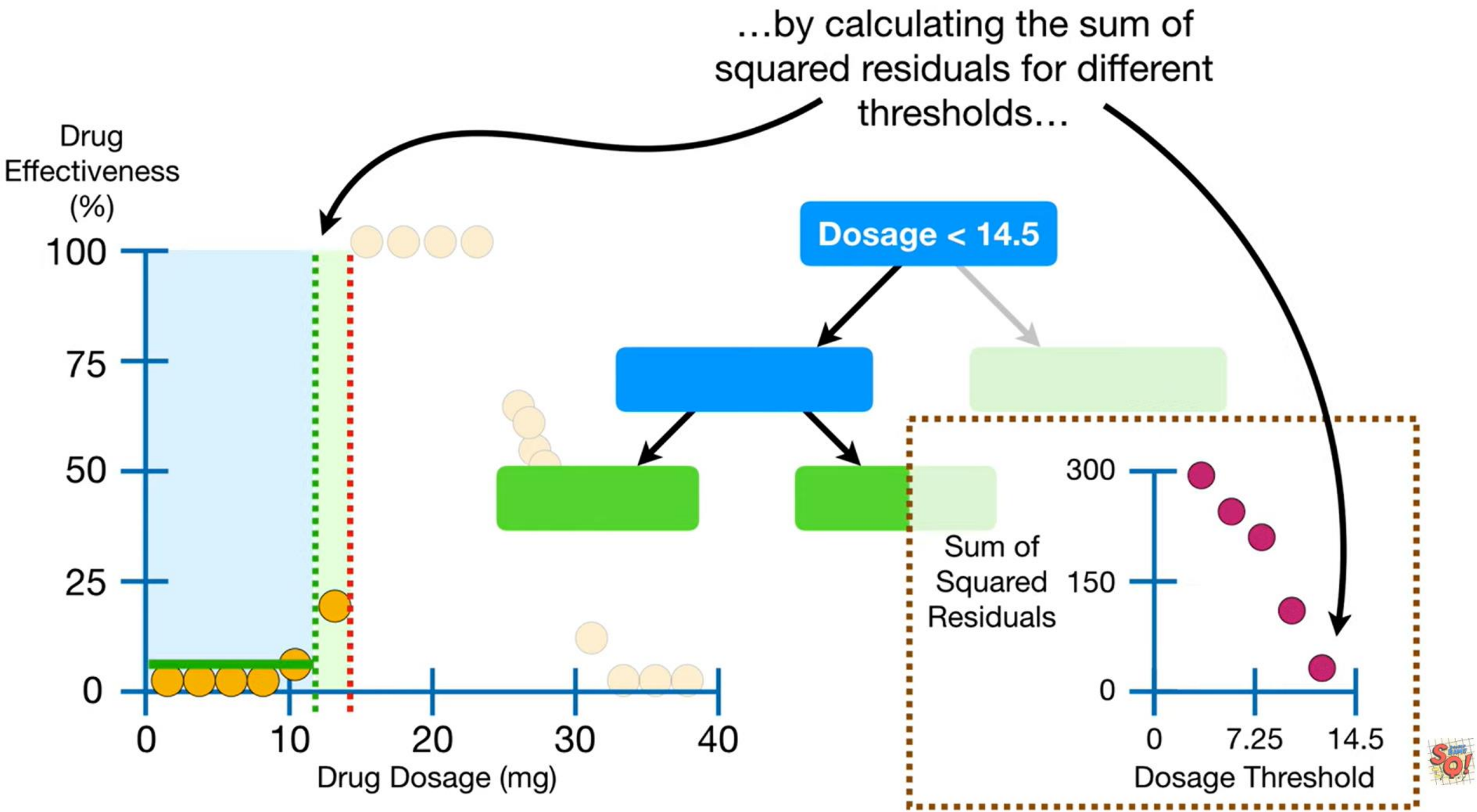






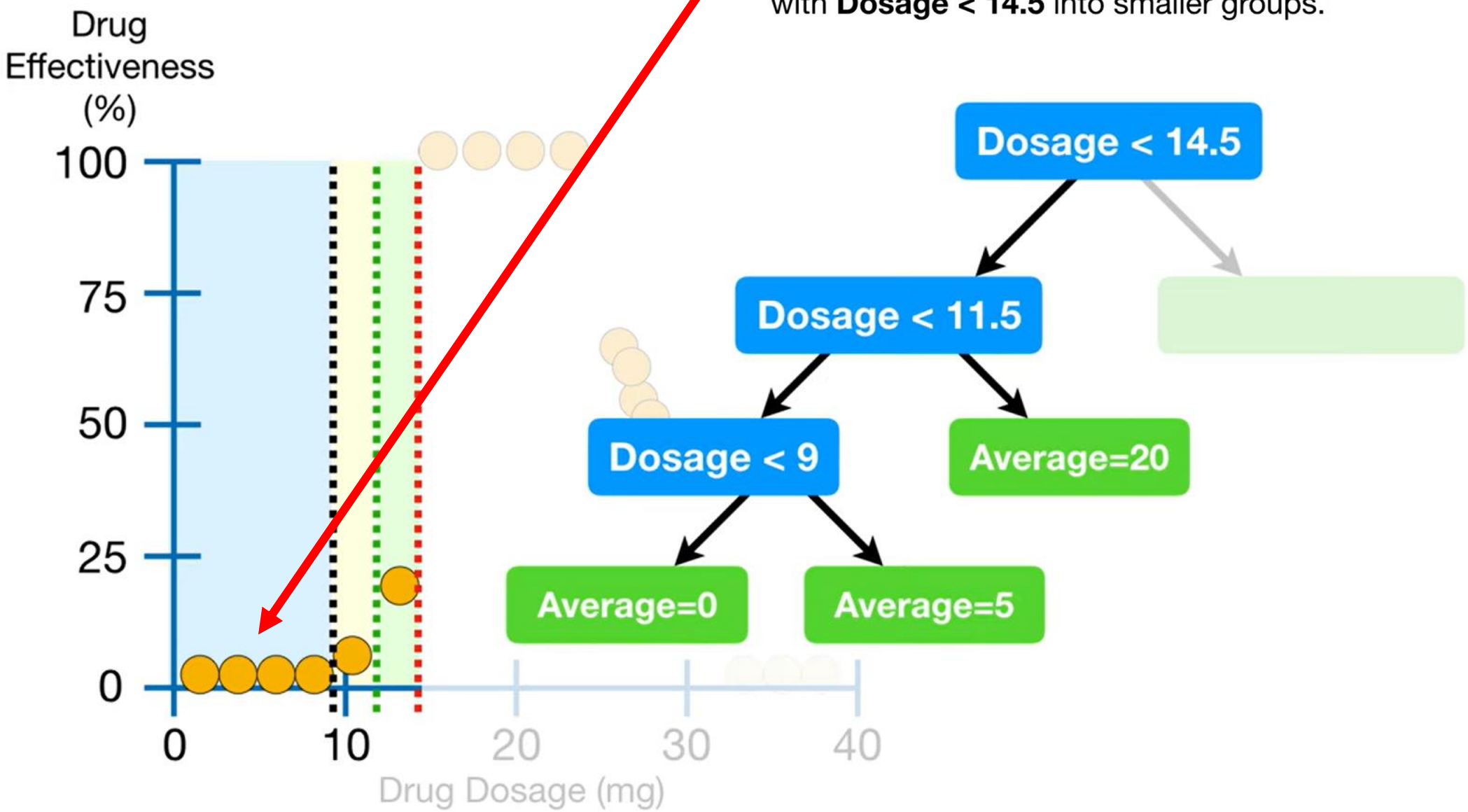

Drug
Effectiveness
(%)
100
75
50
25
0
0
10
20
30
40
Drug Dosage (mg)


Dosage < 14.5

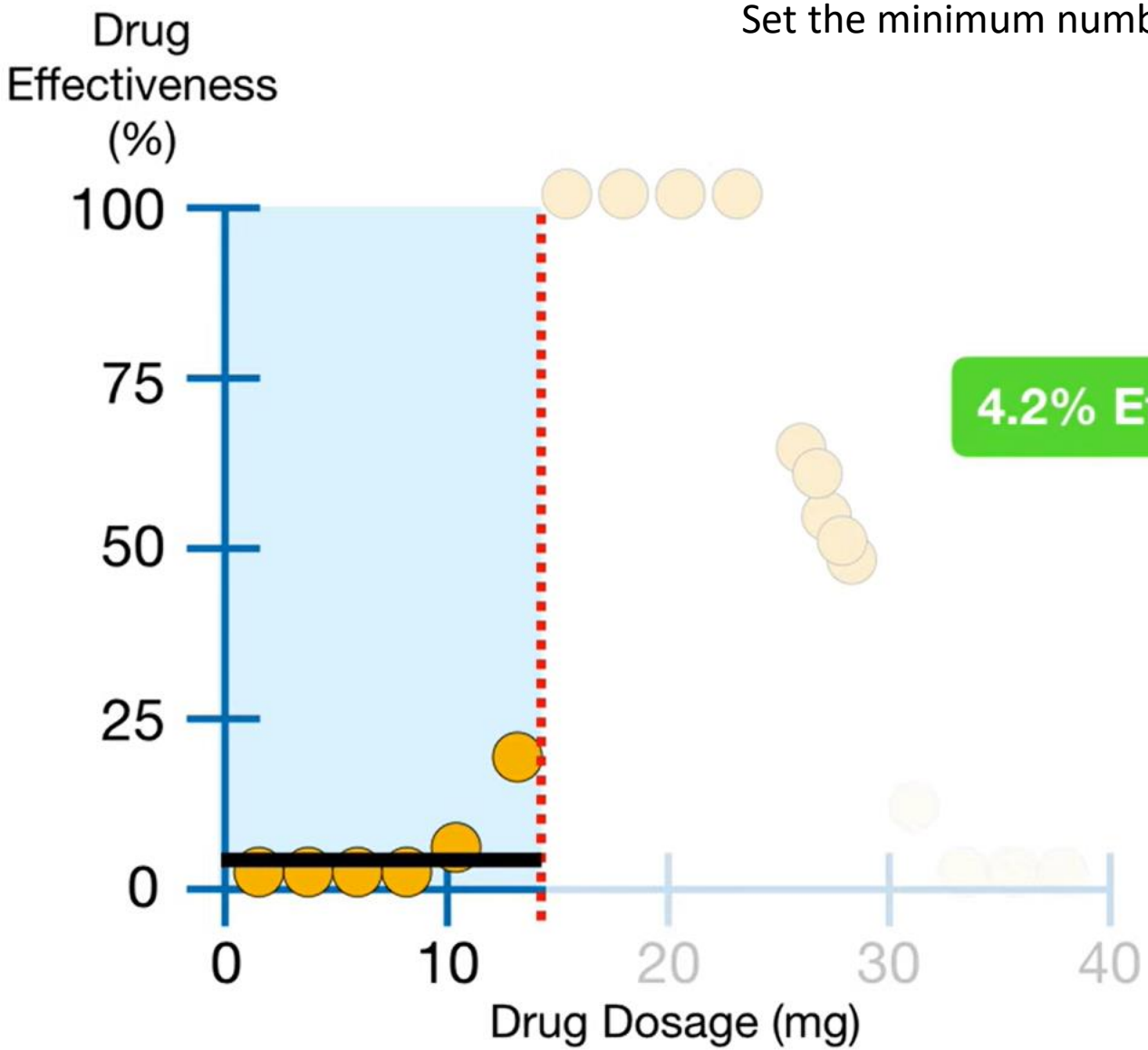

…by calculating the sum of squared residuals for different thresholds…
Drug Effectiveness (%)
100
75
50
25
0
0
10
20
30
40
Drug Dosage (mg)
Dosage < 14.5
Sum of Squared Residuals
300
150
0
0
7.25
14.5
Dosage Threshold

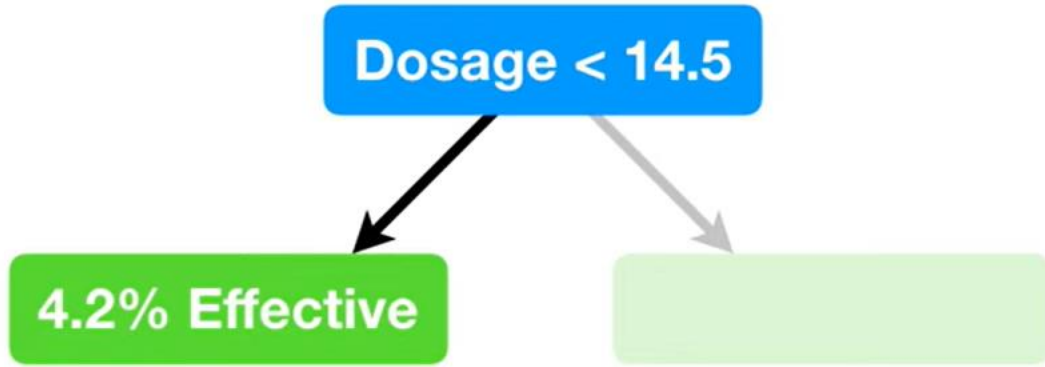
That said, those **4** observations all have the same **Drug Effectiveness**, so we don't need to split them into smaller groups.

So we are done splitting the observations with **Dosage < 14.5** into smaller groups.

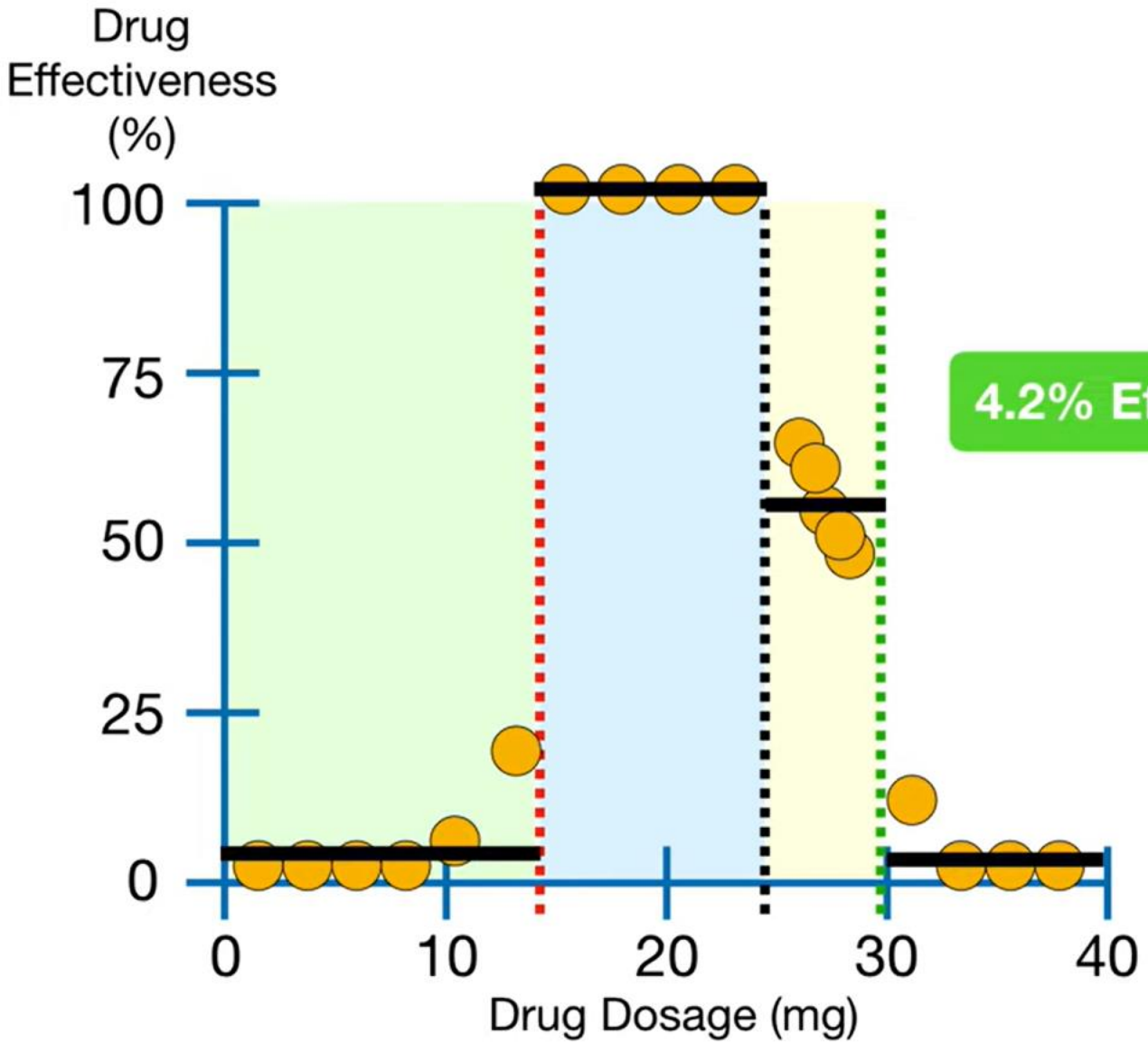


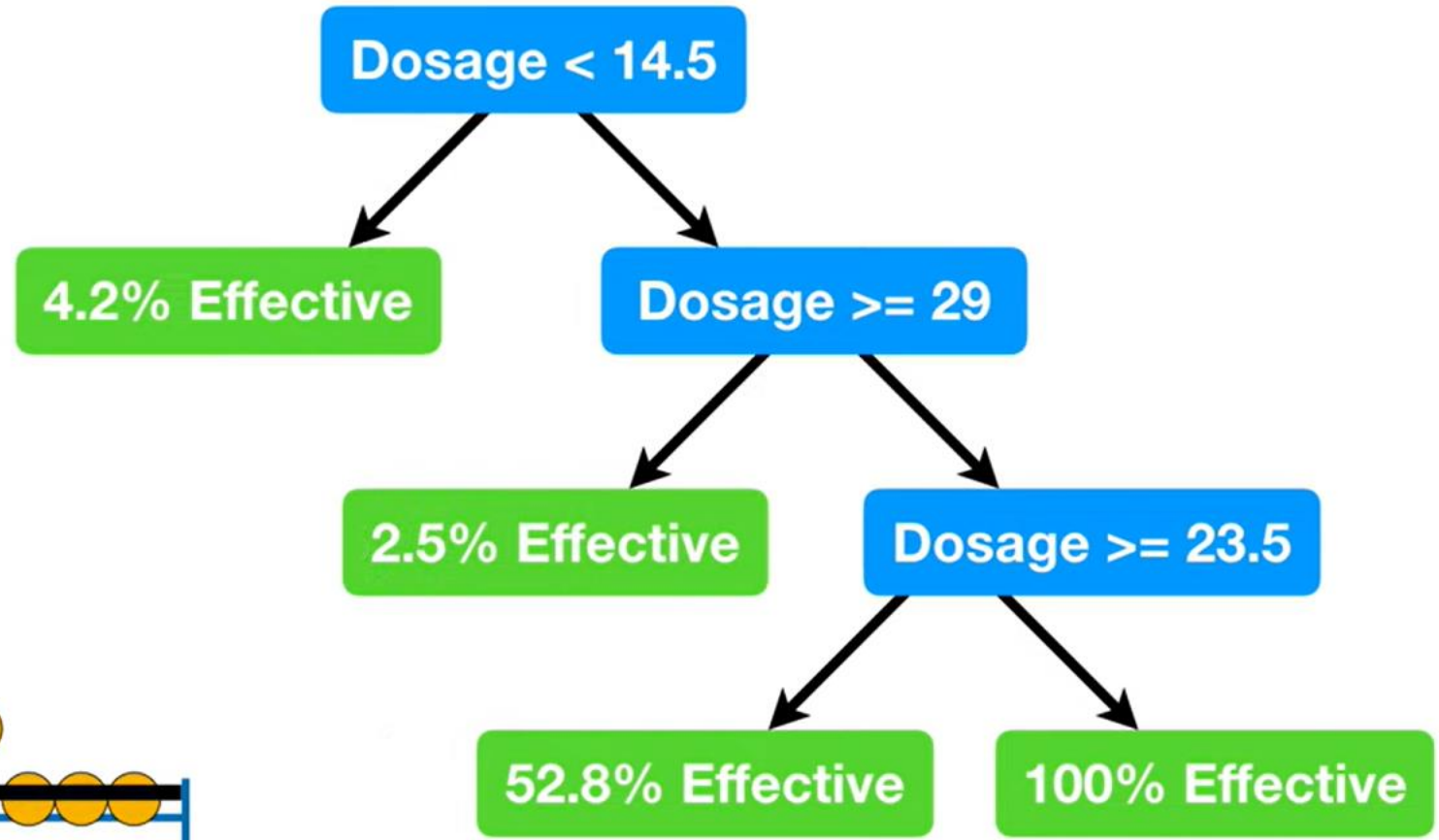
Set the minimum number of observations to allow for a split is 7 (usually 20)





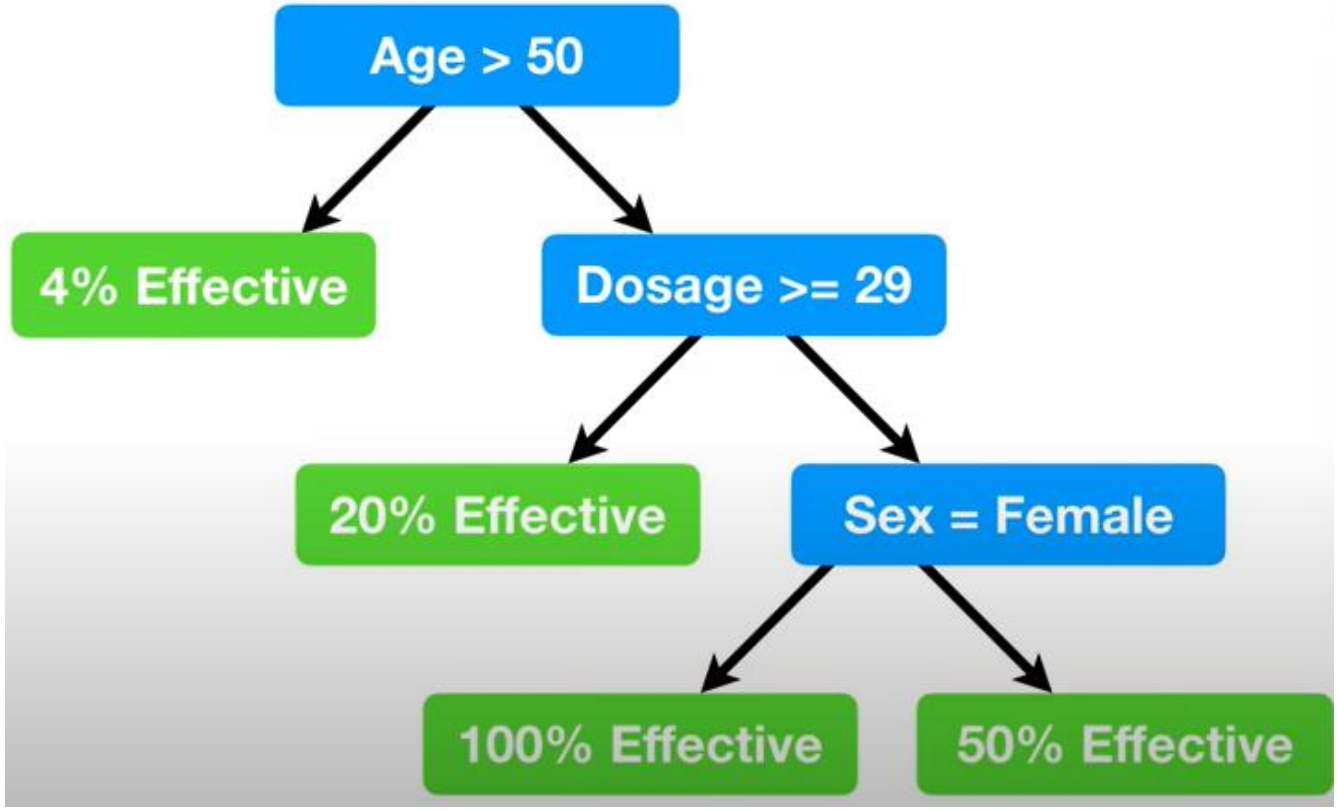
Since no leaf has more than **7** observations in it,…

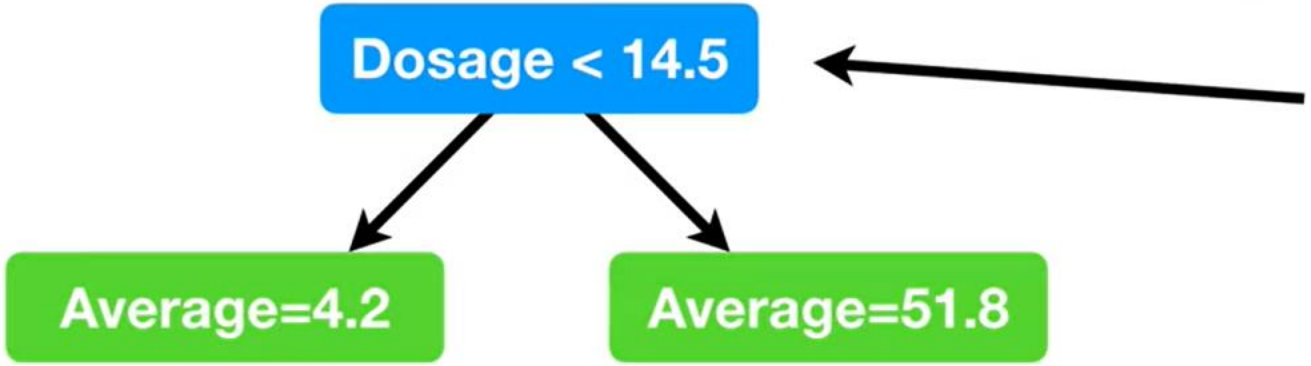




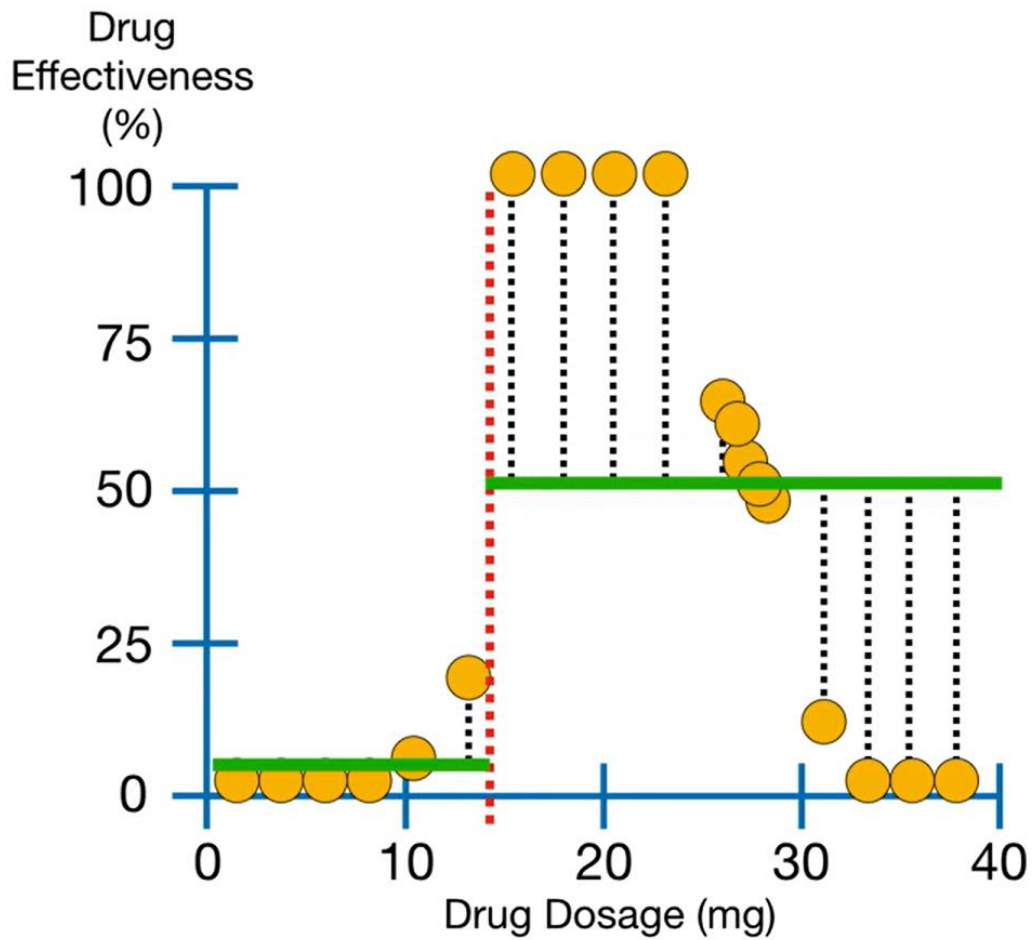
More complicated!!!

Dosage	Age	Sex	Etc.	Drug Effect.
10	25	Female	…	98
20	73	Male	…	0
35	54	Female	…	100
5	12	Male	…	44
etc…	etc…	etc…	etc…	etc…

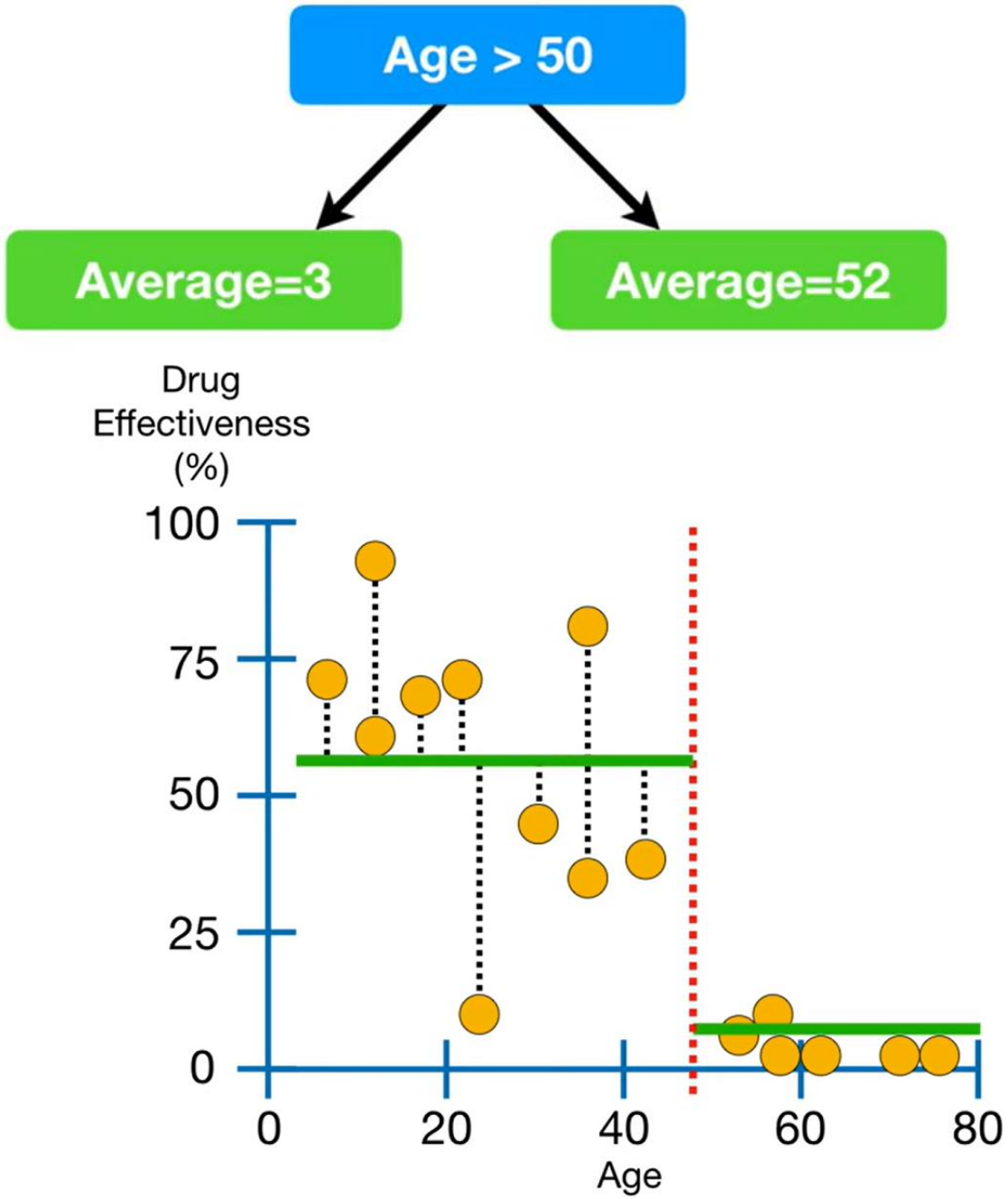




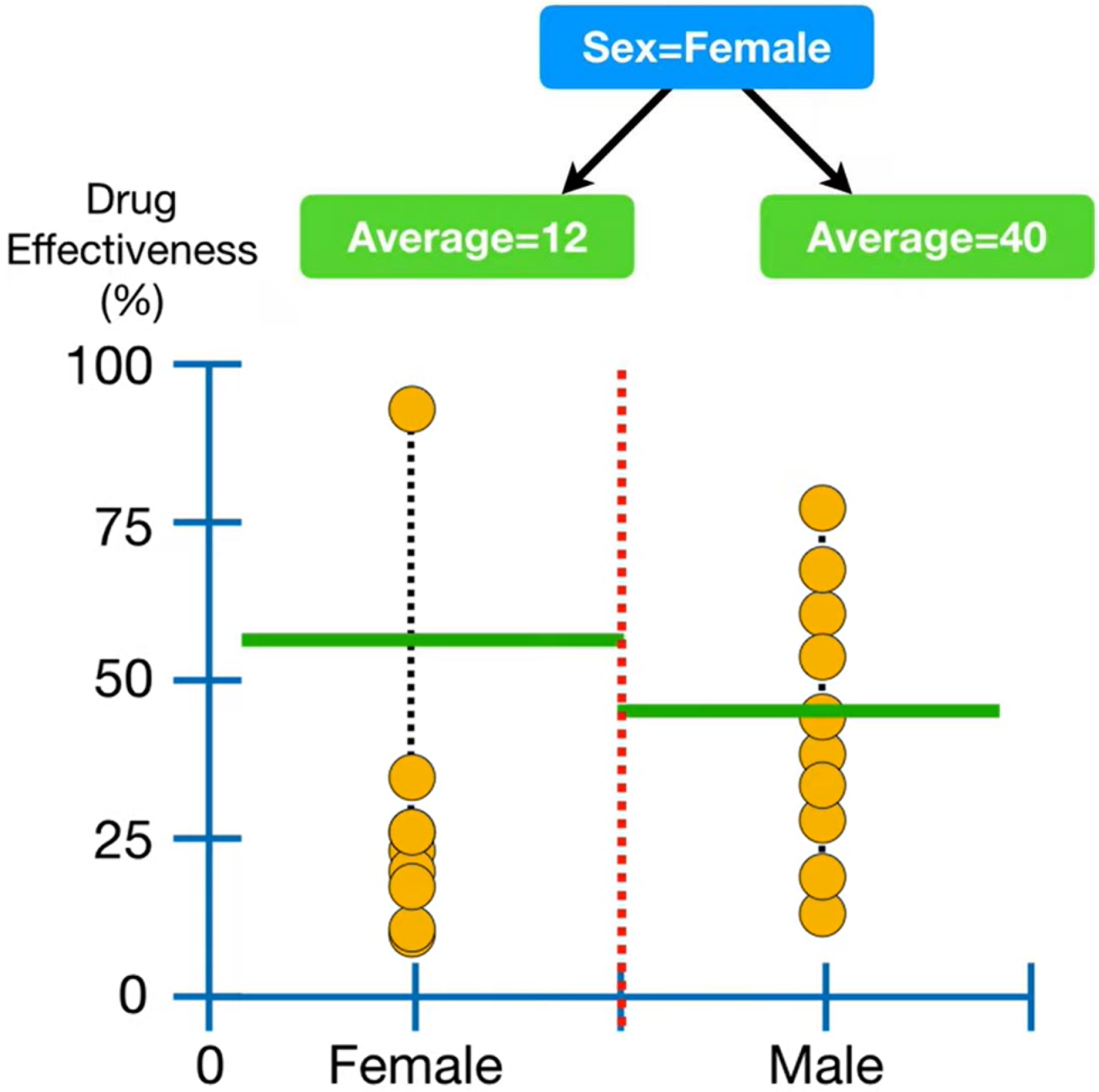
The best threshold becomes a *candidate* for the root.



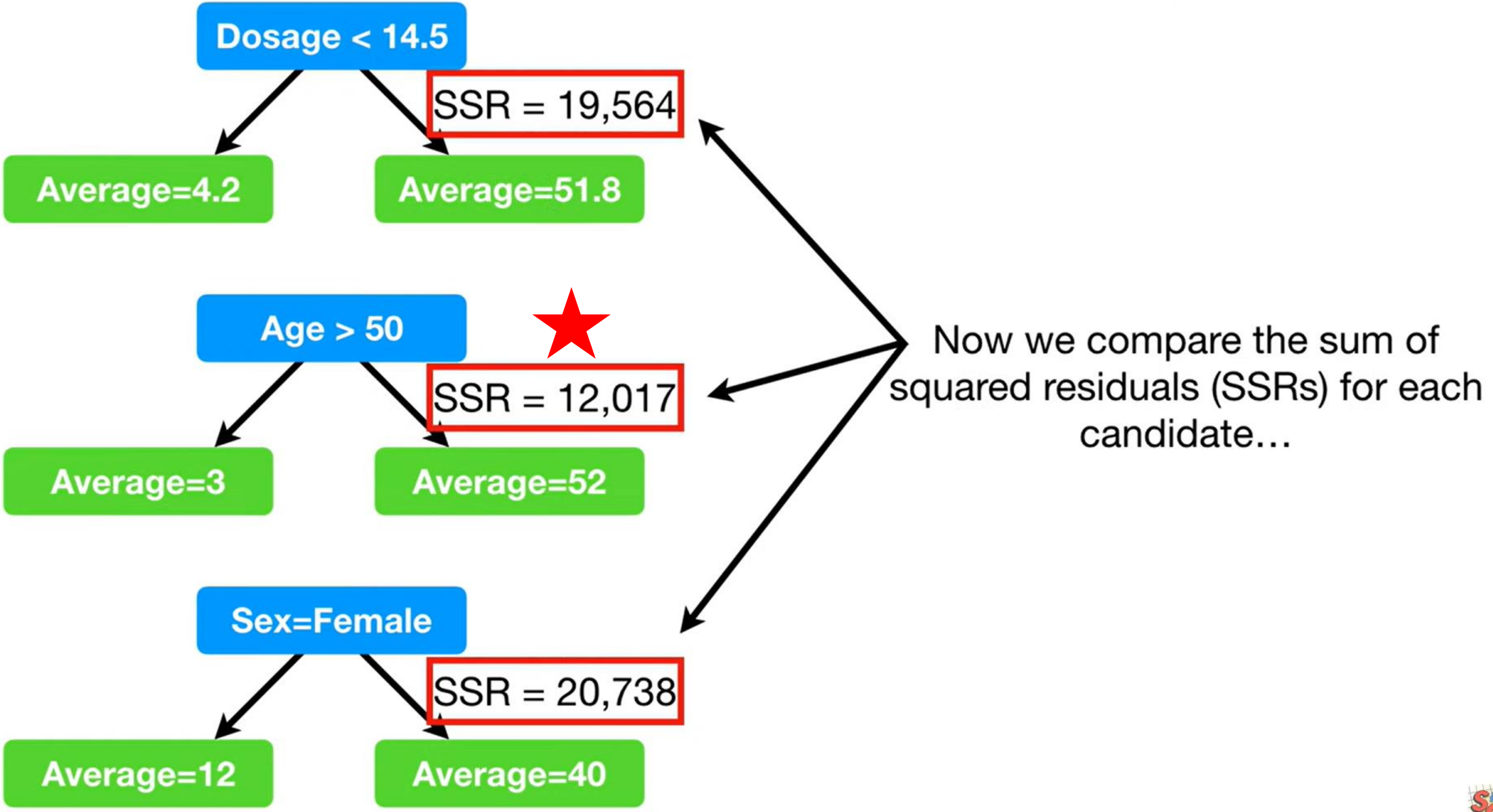
Dosage	Age	Sex	Drug Effect.
10	25	Female	98
20	73	Male	0
35	54	Female	6
5	12	Male	44
etc…	etc…	etc…	etc…



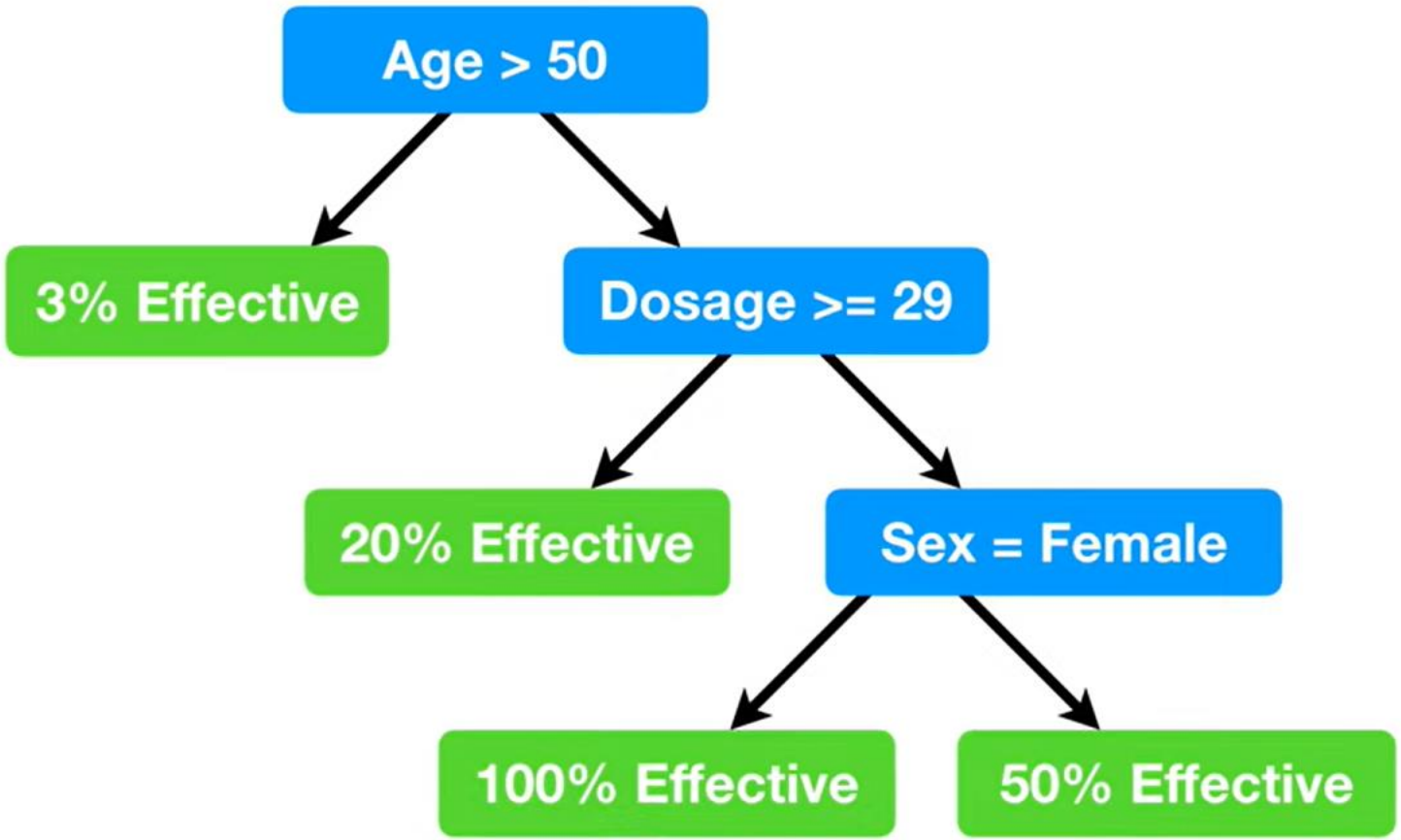
Dosage	Age	Sex	Drug Effect.
10	25	Female	98
20	73	Male	0
35	54	Female	6
5	12	Male	44
etc…	etc…	etc…	etc…



Dosage	Age	Sex	Drug Effect.
10	25	Female	98
20	73	Male	0
35	54	Female	6
5	12	Male	44
etc…	etc…	etc…	etc…


Dosage < 14.5
SSR = 19,564
Average=4.2
Average=51.8
Age > 50
SSR = 12,017
Average=3
Average=52
Sex=Female
SSR = 20,738
Average=12
Average=40
Now we compare the sum of squared residuals (SSRs) for each candidate…

Then we grow the tree just like before, except now we compare the lowest sum of squared residuals from each predictor.



Dosage	Age	Sex	Drug Effect.
10	25	Female	98
20	73	Male	0
35	54	Female	6
5	12	Male	44
etc…	etc…	etc…	etc…

But probability and statistics deal with many different kinds of data, continuous and discrete, and we have dealt with at least the following during the semester:

- Real Numbers -- Height, weight, time, $X \sim N(\mu, \sigma^2)$, ….
- Integers -- Number of emails, Cards, $X \sim B(N,p)$, ….
- Binary – Heads/Tails, Red/Black, $X \sim Bern(p)$, …

Continuous

Discrete

And there are other kinds of (discrete) data which statisticians must consider:

Categorical -- A finite list of unordered categories or labels, e.g.,

- Political parties (Republican, Democrat, Independent, Green. …)
- Blood type (A, B, AB, O, …)
- State lived in (MA, VT, ….)

Ordinal – Like categorical, but with an explicit ordering, e.g.,

- Class year (freshman, sophomore, …)
- Grades (A, A-, B+, …)
- Likert Scale (disagree strongly, disagree, neutral, agree, agree strongly)

Rank my jokes...	Likes StatQuest
1	Yes
1	No
3	Yes
1	Yes
etc...	etc...

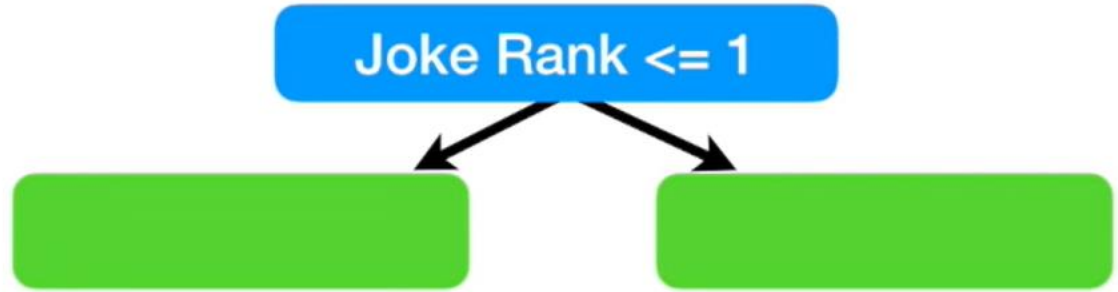
Ranked data is similar to numeric data, except instead now we calculate impurity scores for all of the possible ranks.

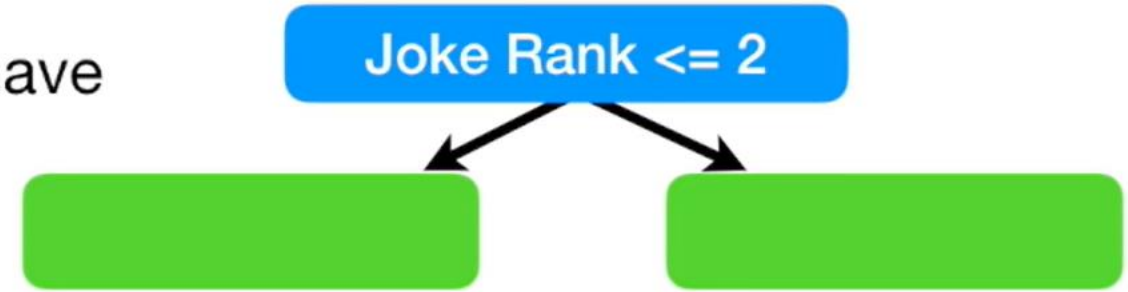
So if people could rank my jokes from 1 to 4 (4 being the funniest), we could calculate the following impurity scores...

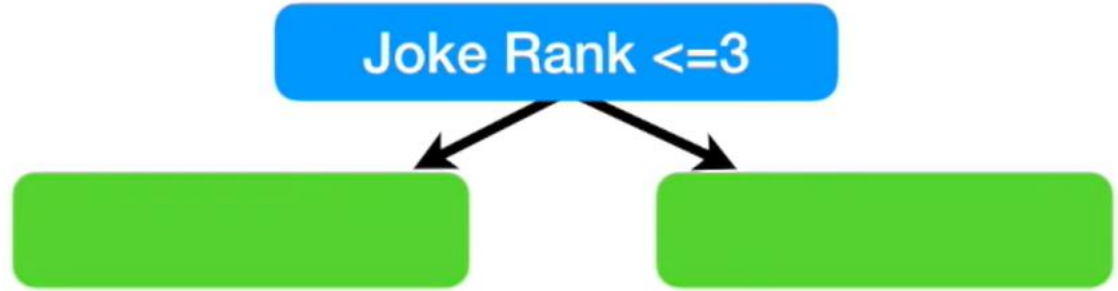
https://www.youtube.com/watch?v=7VeUPuFGJHk&list=PLBq2sVJiEBvA9rPo3IEQsJNI4IJbn81tB&index=1

Rank my jokes…	Likes StatQuest
1	Yes
1	No
3	Yes
1	Yes
etc…	etc…

NOTE: We don't have to calculate an impurity score for Joke Rank <= 4 because that would include everyone.



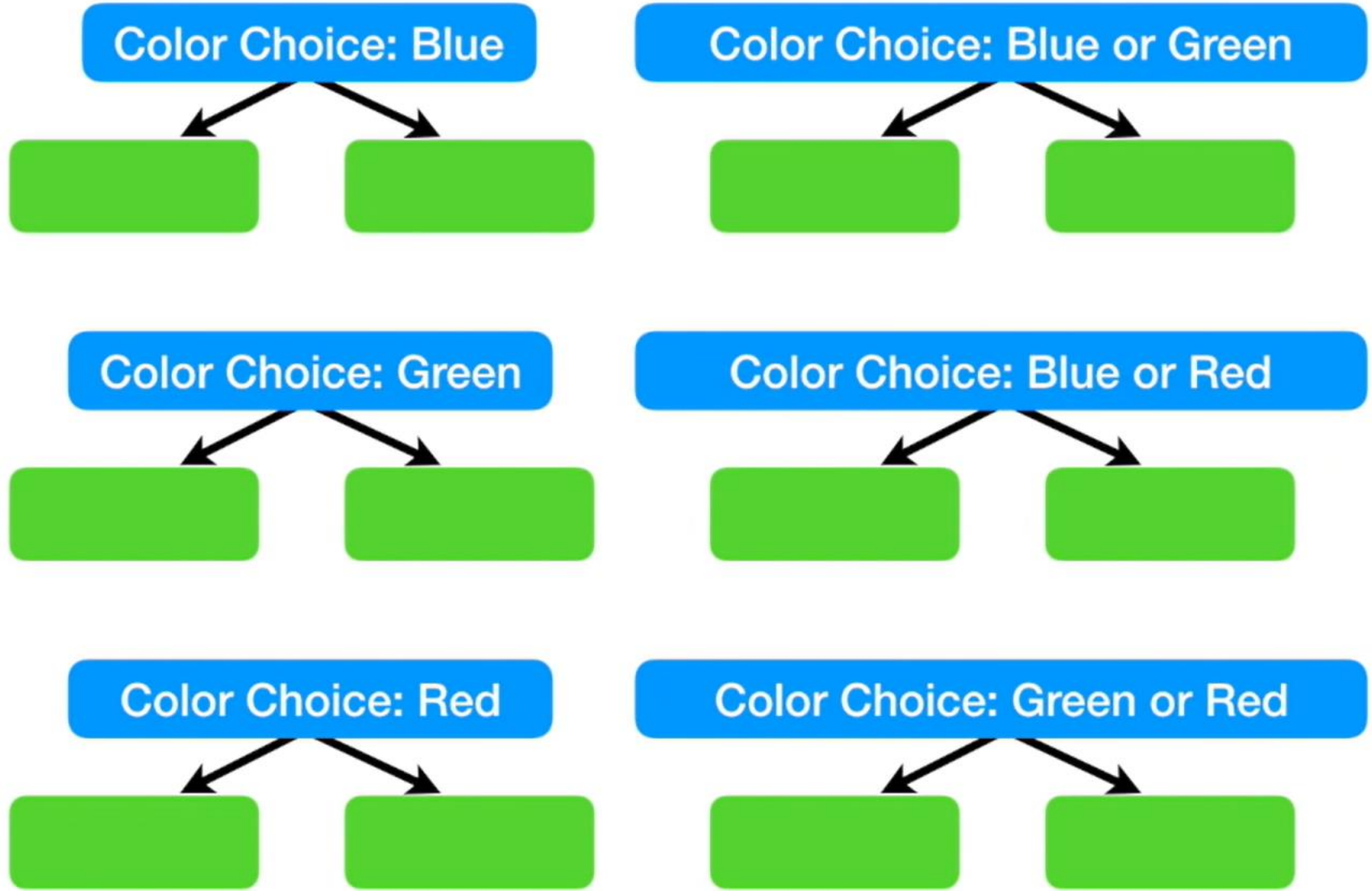




Color Choice	Likes StatQuest
Green	Yes
Blue	No
Red	Yes
Green	Yes
etc…	etc…

When there are **multiple choices**, like "**color choice can be blue, green or red**", you calculate an impurity score for each one as well as each possible combination.

Color Choice	Likes StatQuest
Green	Yes
Blue	No
Red	Yes
Green	Yes
etc…	etc…



Classification Tree

Gini Index

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

OR

Cross Entropy

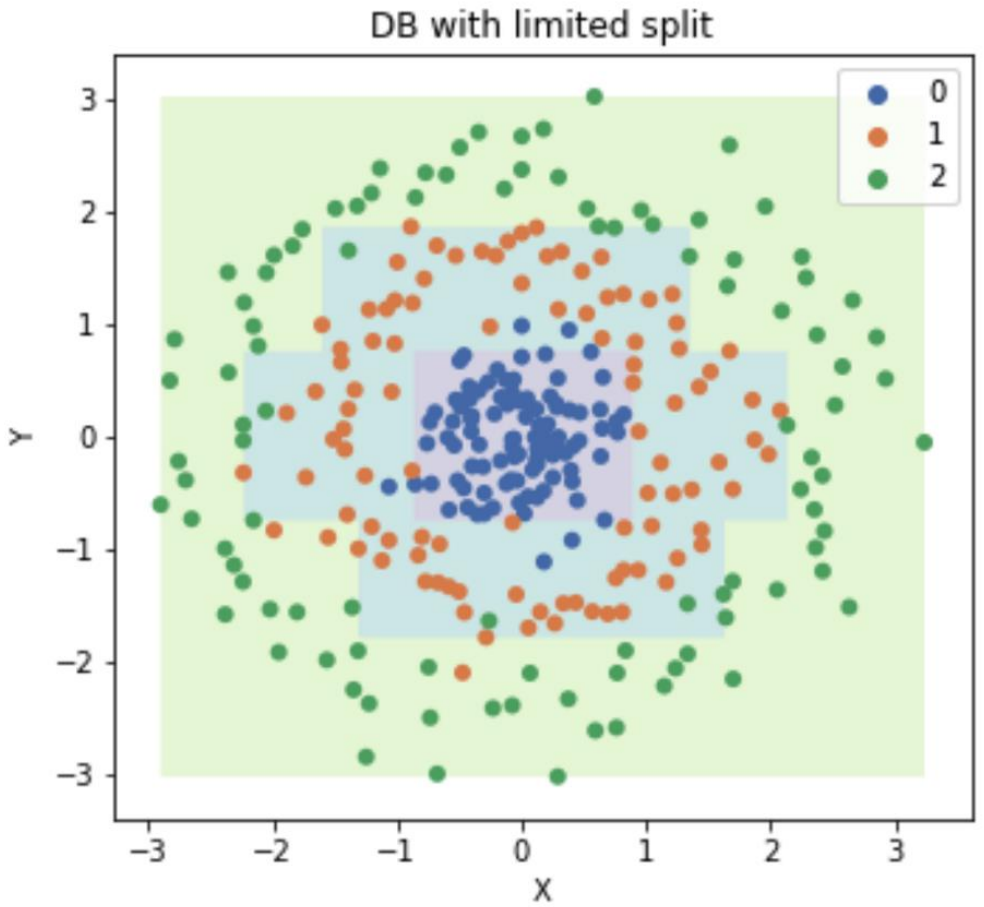
$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.$$

Regression Tree

RSS= $$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2,$$

Tiêu chuẩn dừng

Trong các thuật toán Decision tree, với phương pháp chia trên, ta sẽ chia mãi các node nếu nó chưa tinh khiết. Như vậy, ta sẽ thu được một tree mà mọi điểm trong tập huấn luyện đều được dự đoán đúng (giả sử rằng không có hai input giống nhau nào cho output khác nhau). Khi đó, cây có thể sẽ rất phức tạp (nhiều node) với nhiều leaf node chỉ có một vài điểm dữ liệu. Như vậy, nhiều khả năng **overfitting** sẽ xảy ra.




```
|---feature_1 <= 1.87
|   |---feature_1 <= -0.74
|   |   |---feature_1 <= -1.79
|   |   |   |---feature_1 <= -2.1
|   |   |   |   |---Class: 2
|   |   |   |---feature_1 > -2.1
|   |   |   |   |---Class: 2
|   |   |---feature_1 > -1.79
|   |   |   |---feature_0 <= 1.62
|   |   |   |   |---feature_0 <= -1.31
|   |   |   |   |   |---Class: 2
|   |   |   |   |---feature_0 > -1.31
|   |   |   |   |   |---feature_1 <= -1.49
|   |   |   |   |   |   |---Class: 1
|   |   |   |   |   |---feature_1 > -1.49
|   |   |   |   |   |   |---Class: 1
|   |   |   |---feature_0 > 1.62
|   |   |   |   |---Class: 2
|   |---feature_1 > -0.74
|   |   |---feature_1 <= 0.76
|   |   |   |---feature_0 <= 0.89
|   |   |   |   |---feature_0 <= -0.86
|   |   |   |   |   |---feature_0 <= -2.24
|   |   |   |   |   |   |---Class: 2
|   |   |   |   |   |---feature_0 > -2.24
|   |   |   |   |   |   |---Class: 1
|   |   |   |   |---feature_0 > -0.86
|   |   |   |   |   |---Class: 0
|   |   |   |---feature_0 > 0.89
|   |   |   |   |---feature_0 <= 2.13
|   |   |   |   |   |---Class: 1
|   |   |   |   |---feature_0 > 2.13
|   |   |   |   |   |---Class: 2
|   |   |---feature_1 > 0.76
|   |   |   |---feature_0 <= -1.6
|   |   |   |   |---Class: 2
|   |   |   |---feature_0 > -1.6
|   |   |   |   |---feature_0 <= 1.35
|   |   |   |   |   |---feature_1 <= 1.66
|   |   |   |   |   |   |---Class: 1
|   |   |   |   |   |---feature_1 > 1.66
|   |   |   |   |   |   |---Class: 1
|   |   |   |   |---feature_0 > 1.35
|   |   |   |   |   |---Class: 2
|---feature_1 > 1.87
|   |---Class: 2
```

https://towardsdatascience.com/construct-a-decision-tree-and-how-to-deal-with-overfitting-f907efc1492d

Để tránh trường hợp này, ta có thể dừng cây theo một số phương pháp sau đây:

- nếu node đó có entropy bằng 0, tức mọi điểm trong node đều thuộc một class.
- nếu node đó có số phần tử nhỏ hơn một ngưỡng nào đó. Trong trường hợp này, ta chấp nhận có một số điểm bị phân lớp sai để tránh overfitting. Class cho leaf node này có thể được xác định dựa trên **class chiếm đa số trong node**.
- nếu khoảng cách từ node đó đến root node đạt tới một giá trị nào đó. Việc **hạn chế *chiều sâu của tree*** này làm giảm độ phức tạp của tree và phần nào giúp tránh overfitting.
- nếu tổng số leaf node vượt quá một ngưỡng nào đó.
- Ngoài ra, ta còn có phương pháp **cắt tỉa cây (TREE PRUNING)**.

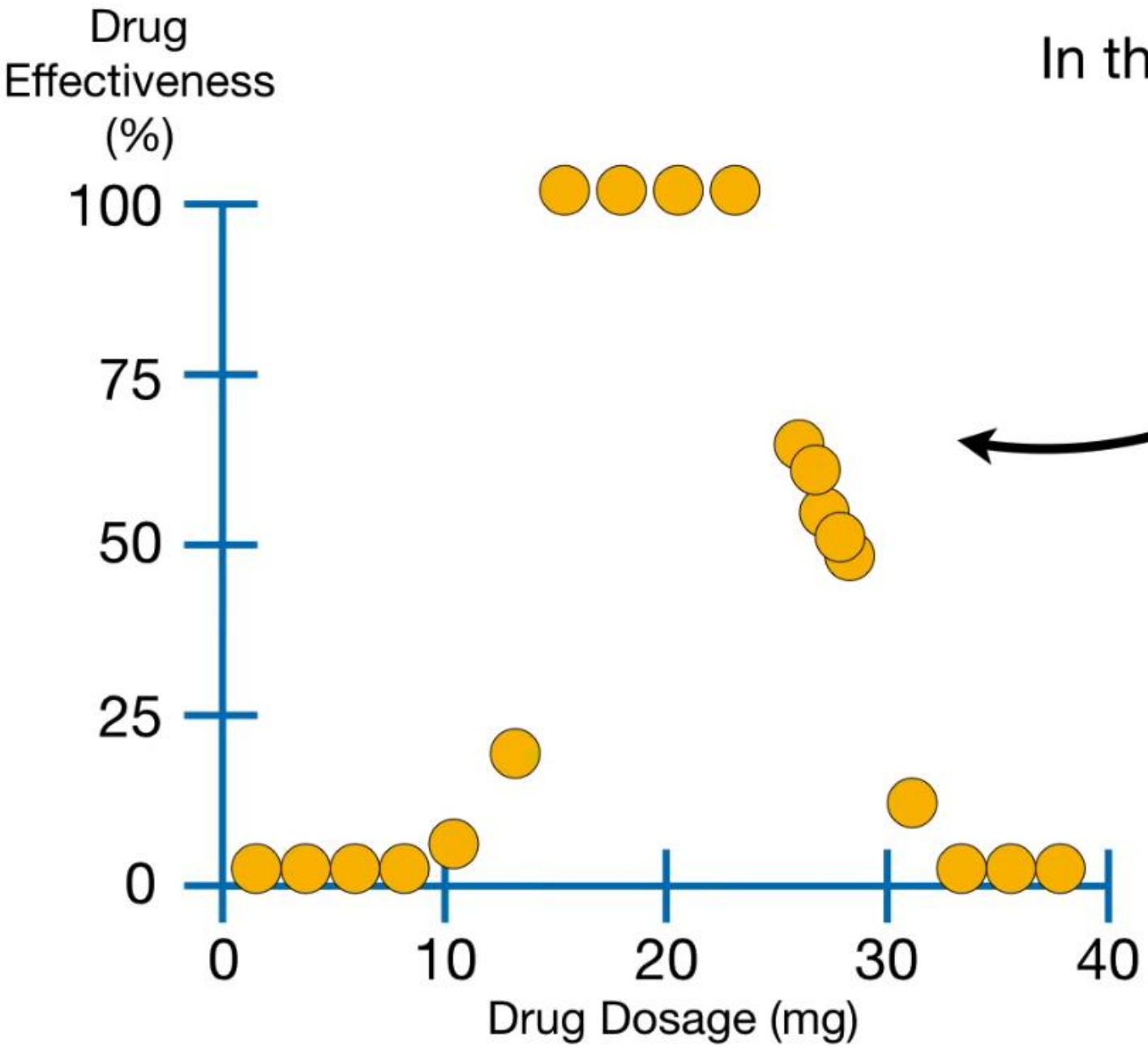
TREE PRUNING

TREE PRUNING

- 1. Why we need to prune the tree?
- 2. What is cost complexity pruning?
- 3. How to choose the optimal cost complexity parameter?

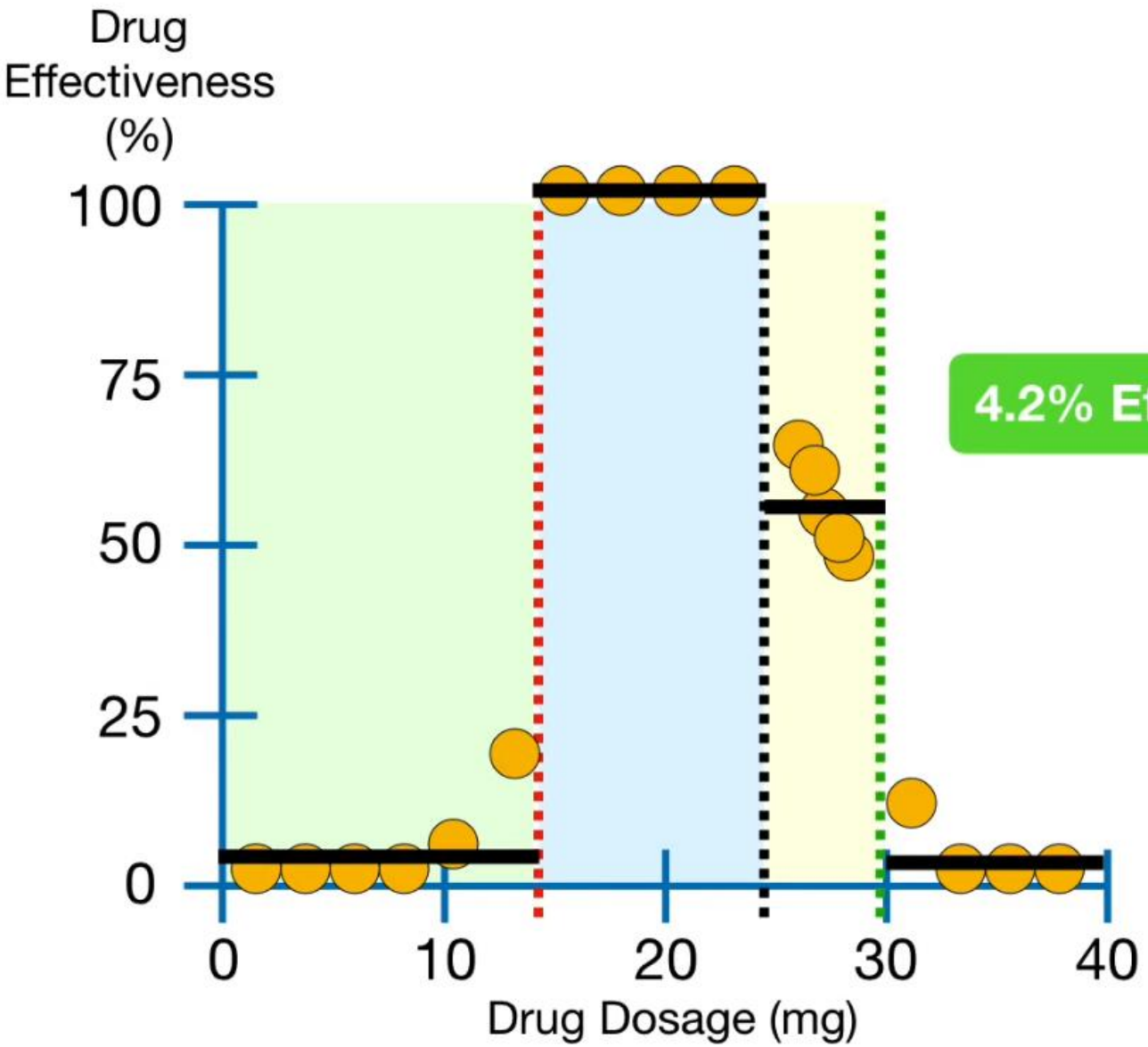
1. Why we need to prune the tree

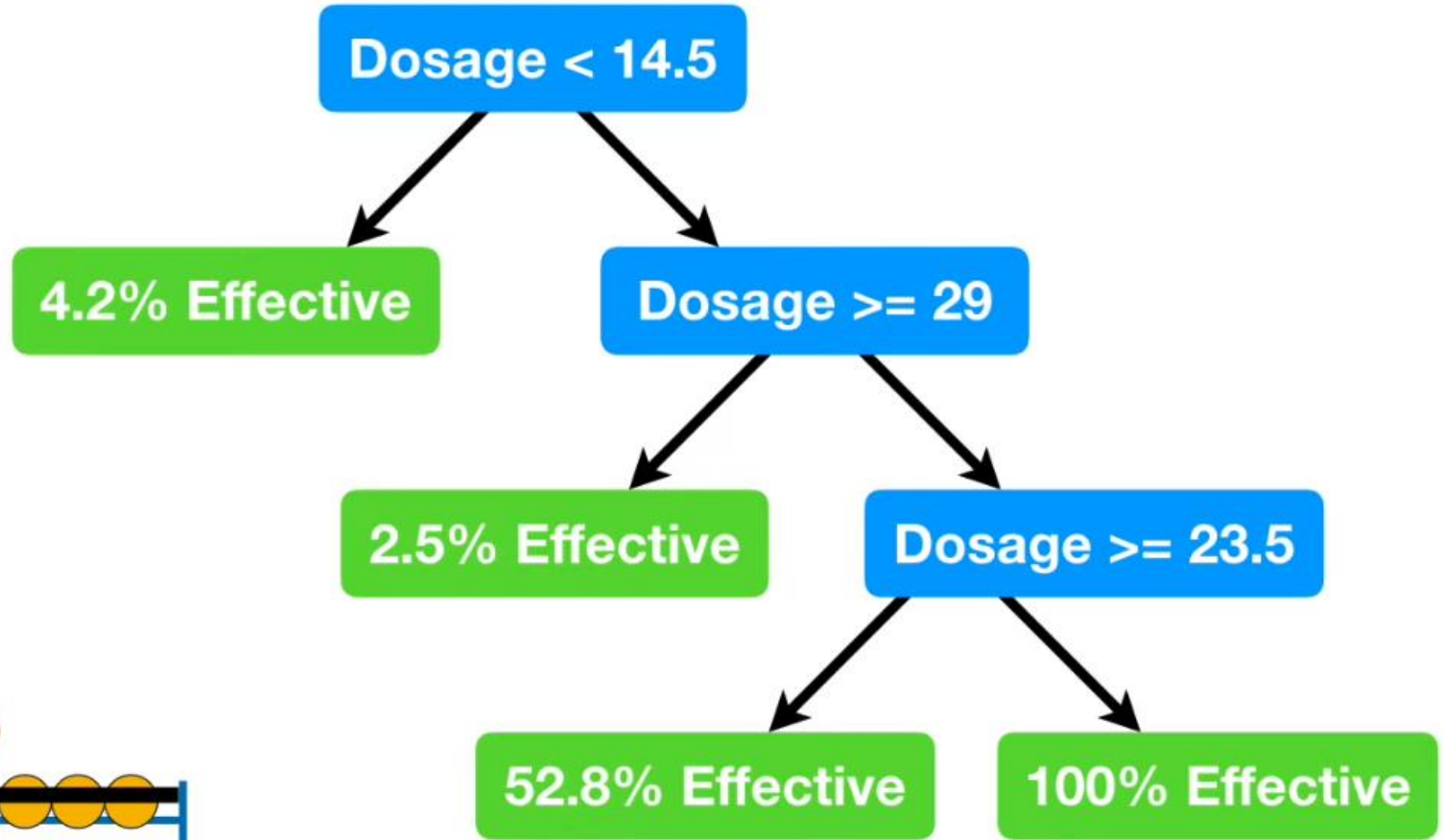
- The process described above may produce good predictions on the training set, but is likely to *overfit* the data, leading to poor test set performance.
- A smaller tree with fewer splits (that is, fewer regions *R*1*; : : : ; RJ*) might lead to lower variance and better interpretation at the cost of a little bias.



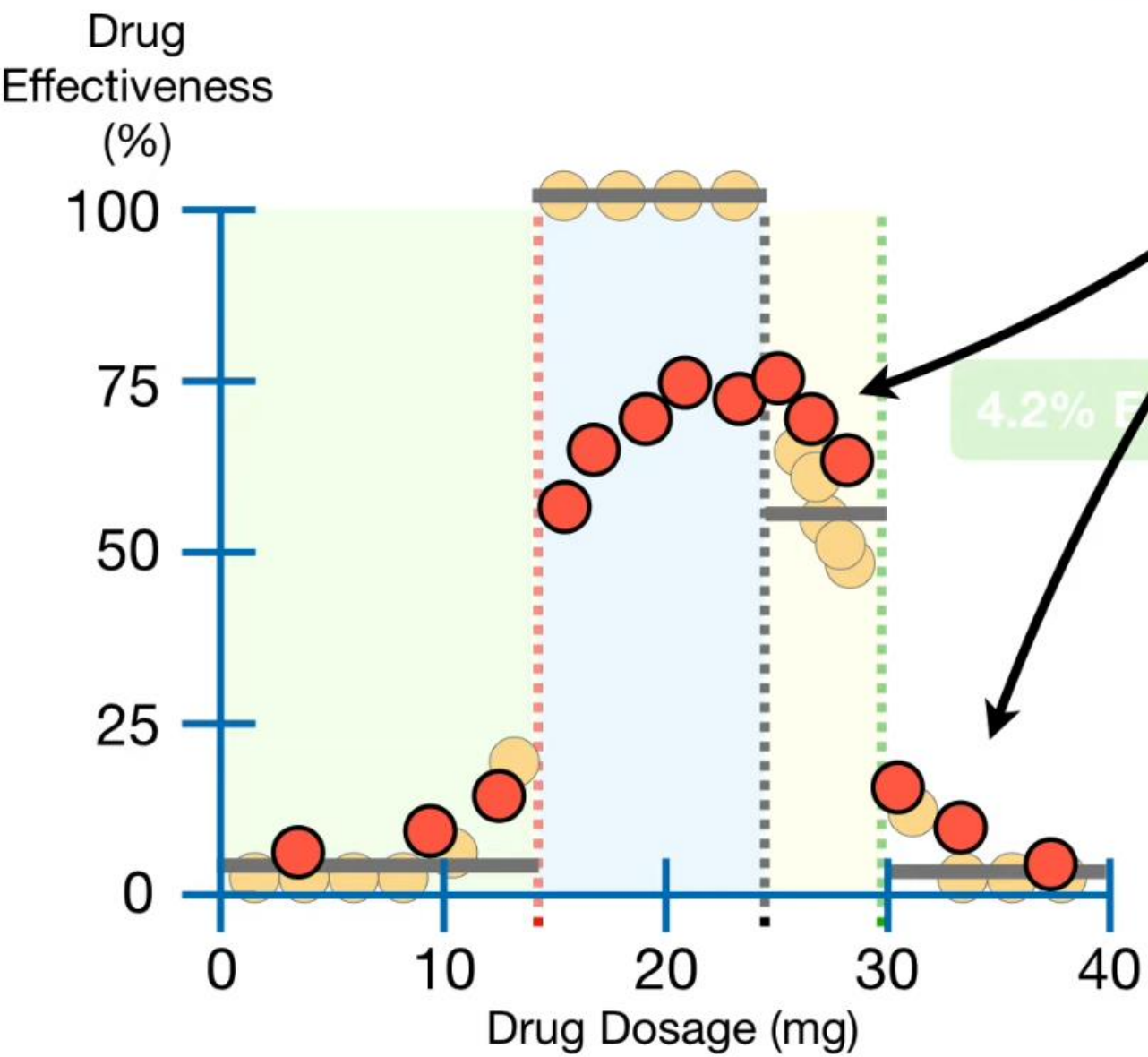
In the **StatQuest** on **Regression Trees**,
we had this data.

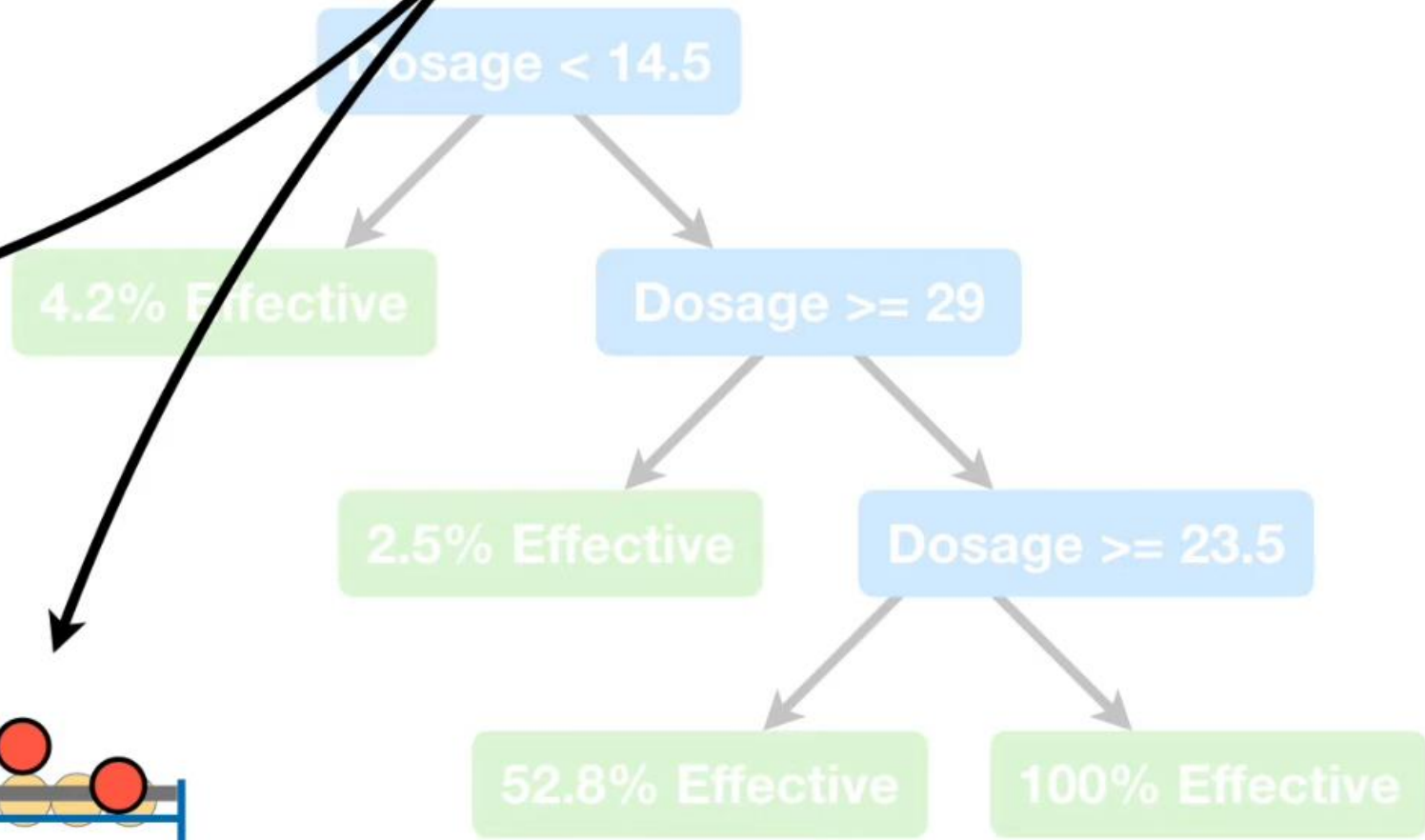
We then fit a **Regression Tree** to the data…



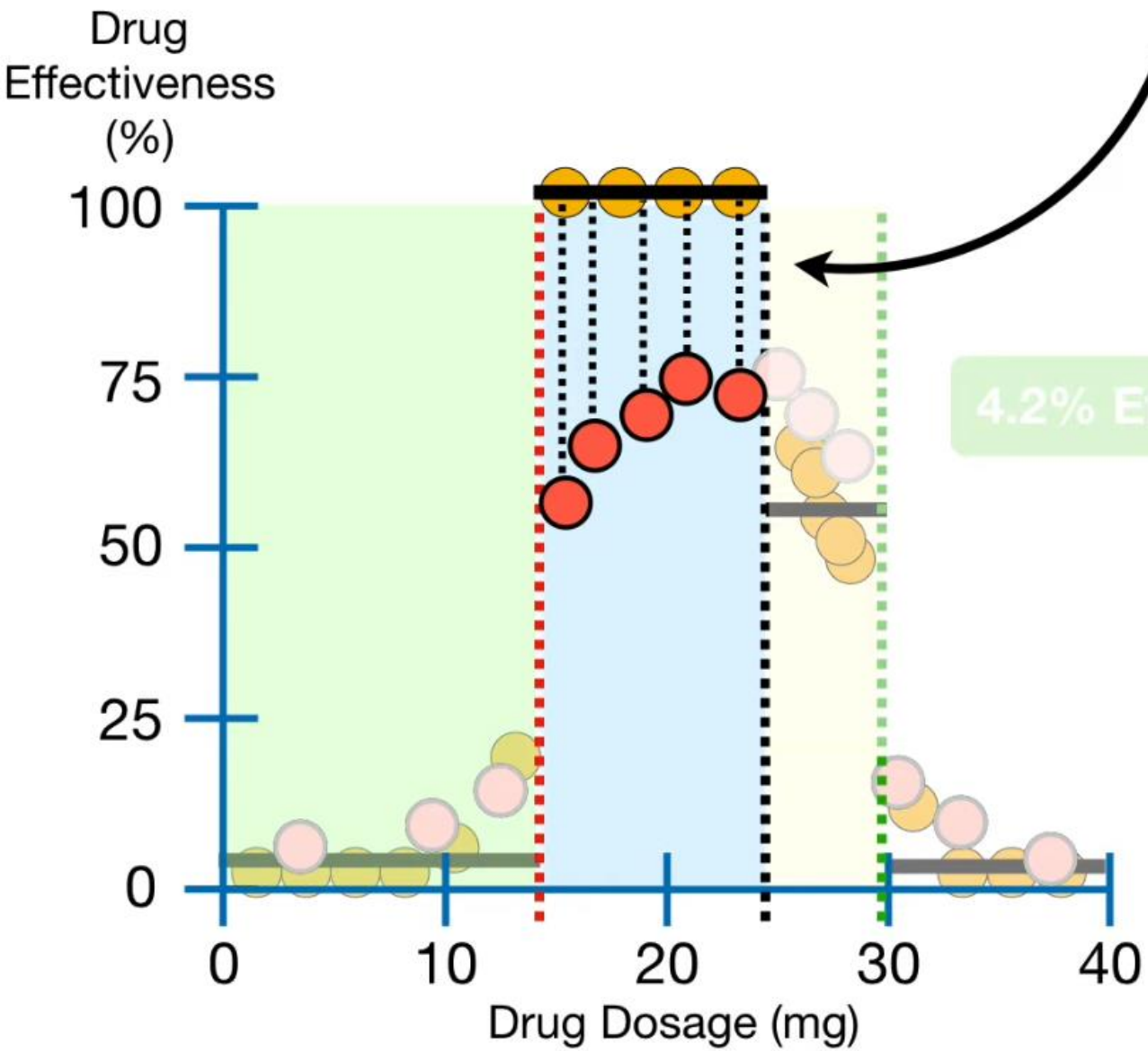


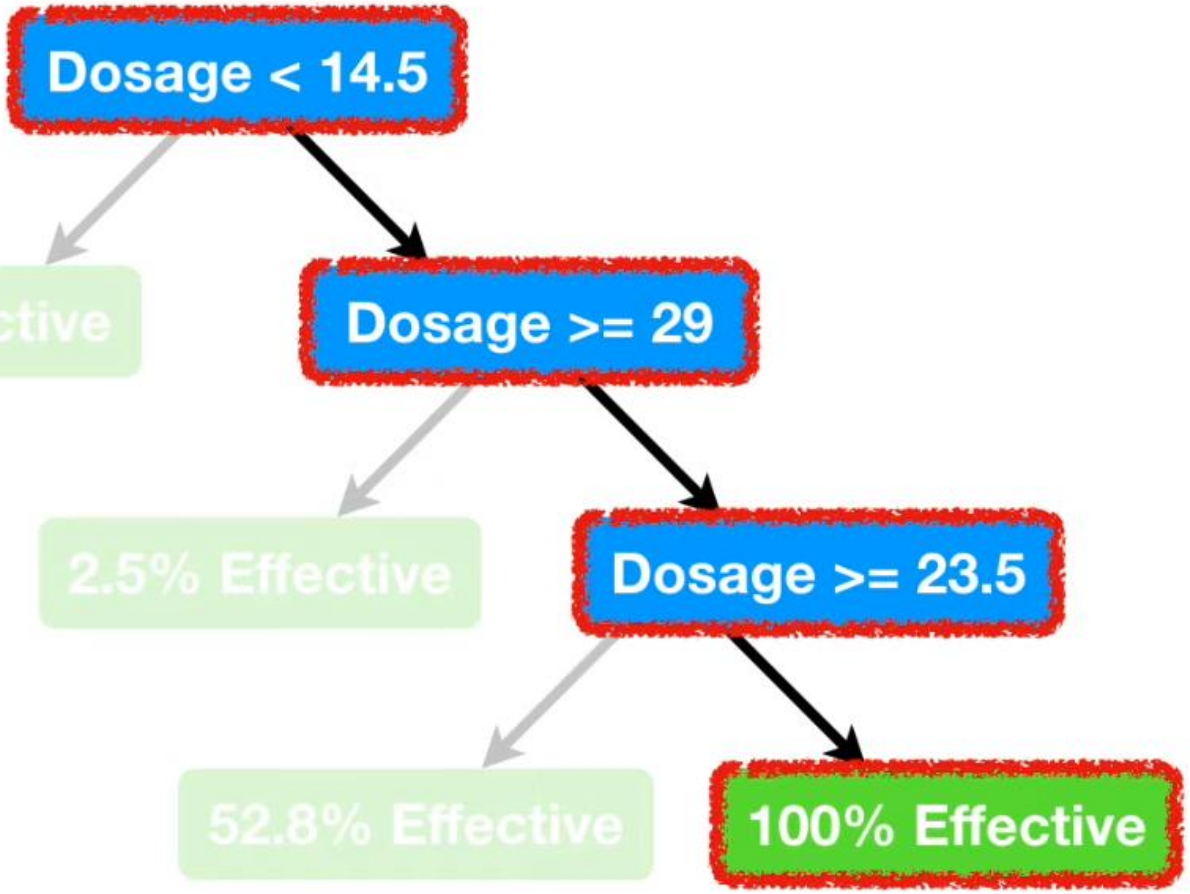
However, what if these **Red Circles** were **Testing Data**?



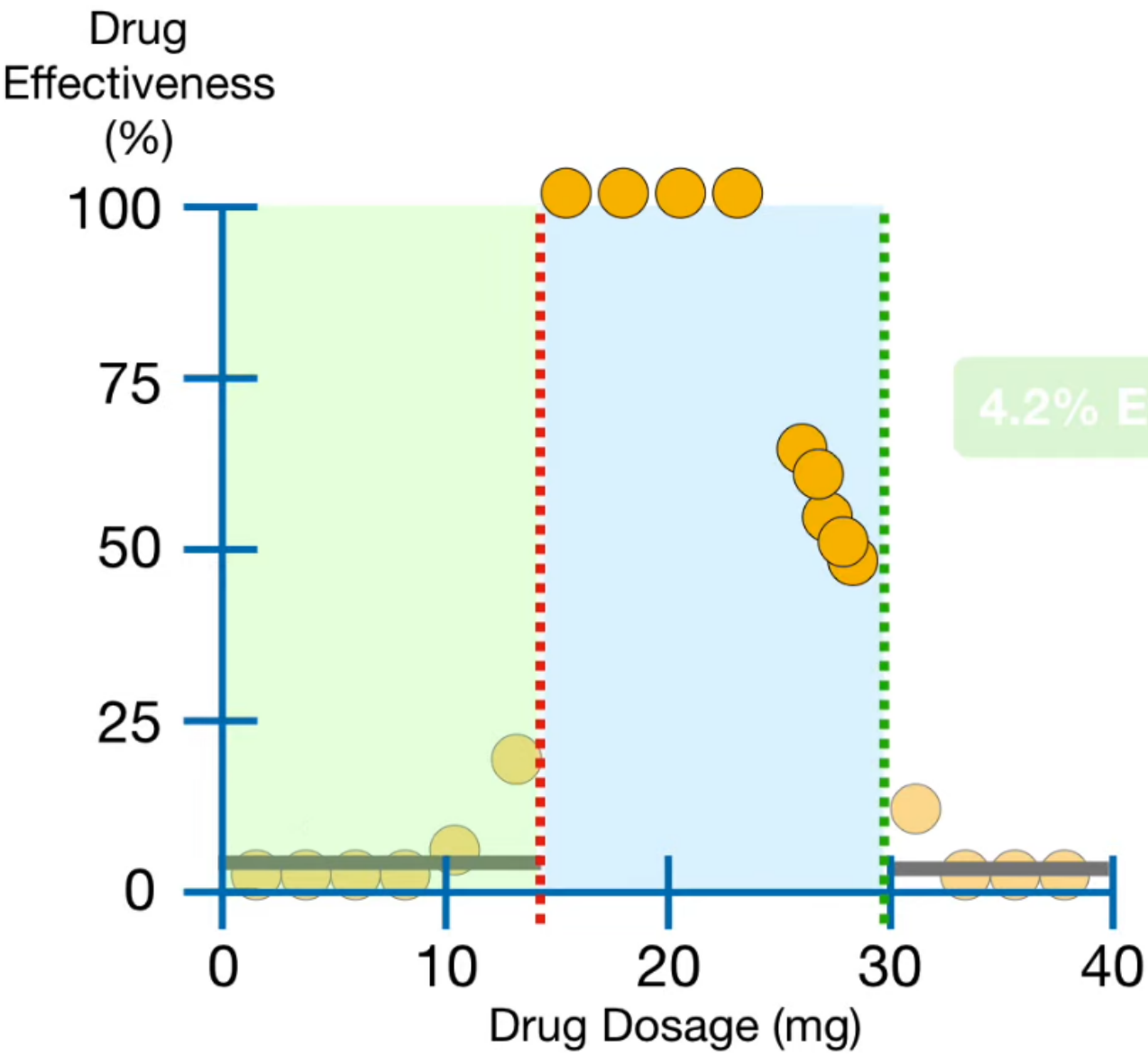


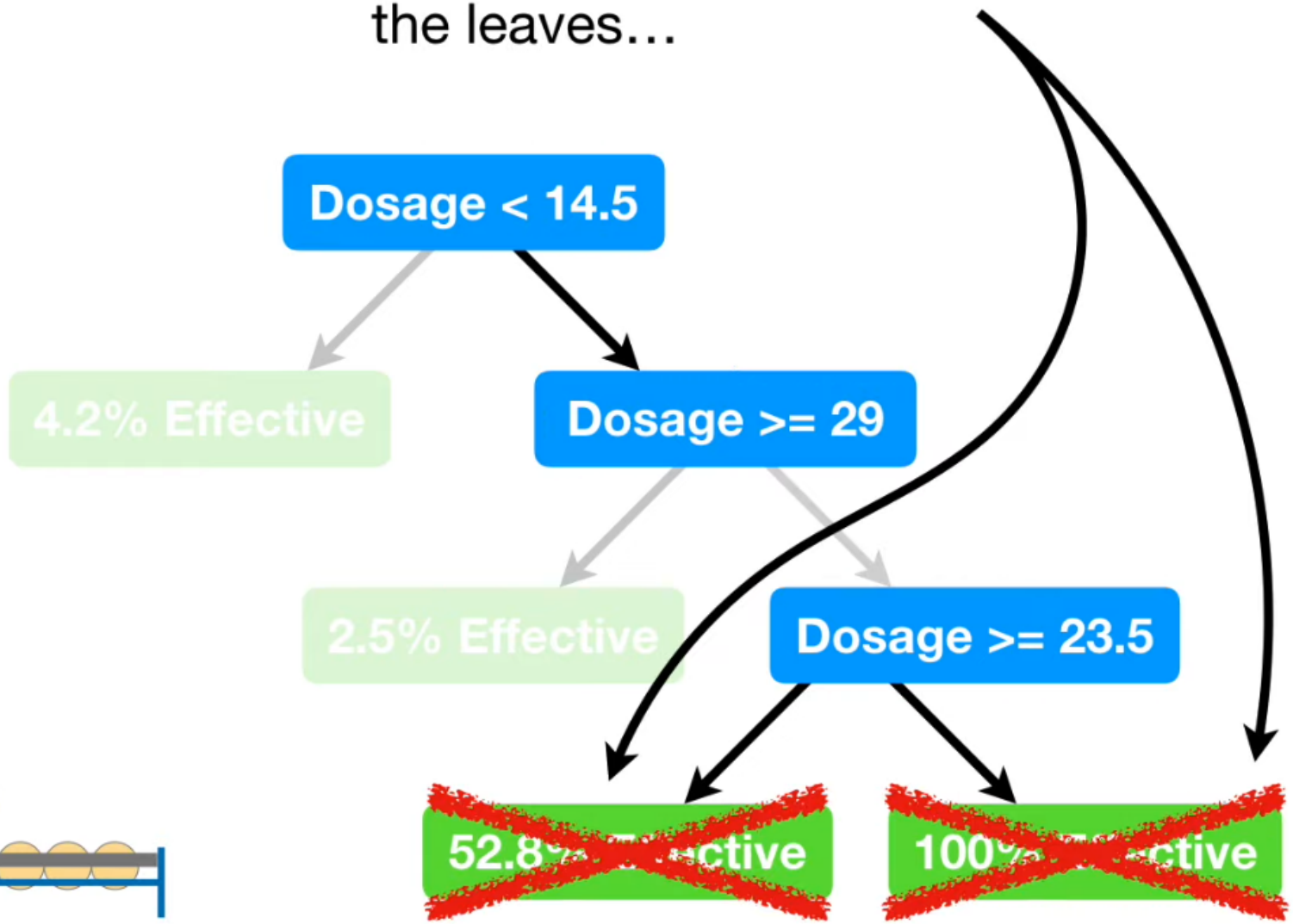
And the **Residuals** for these **Observations** are much larger.



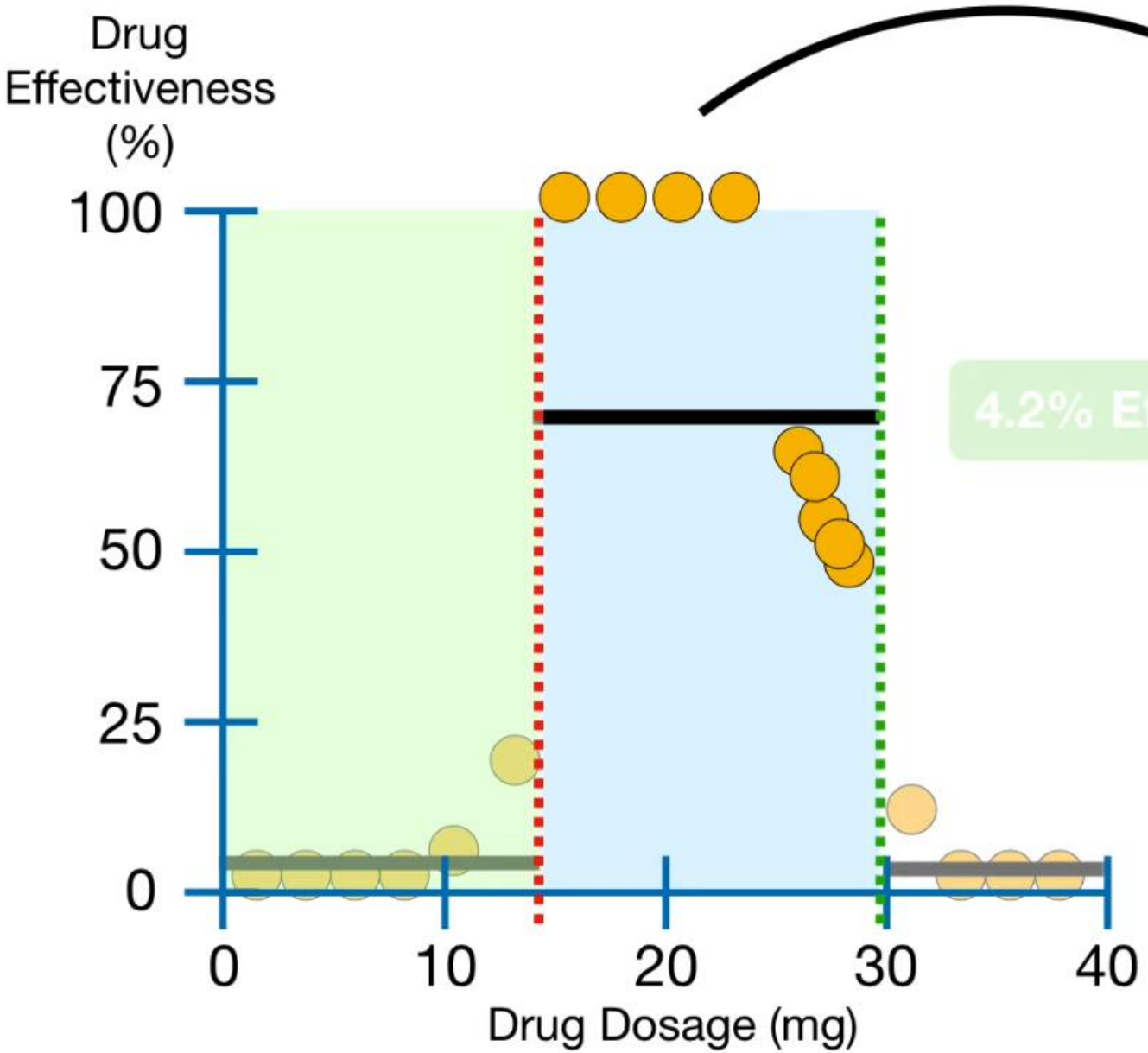


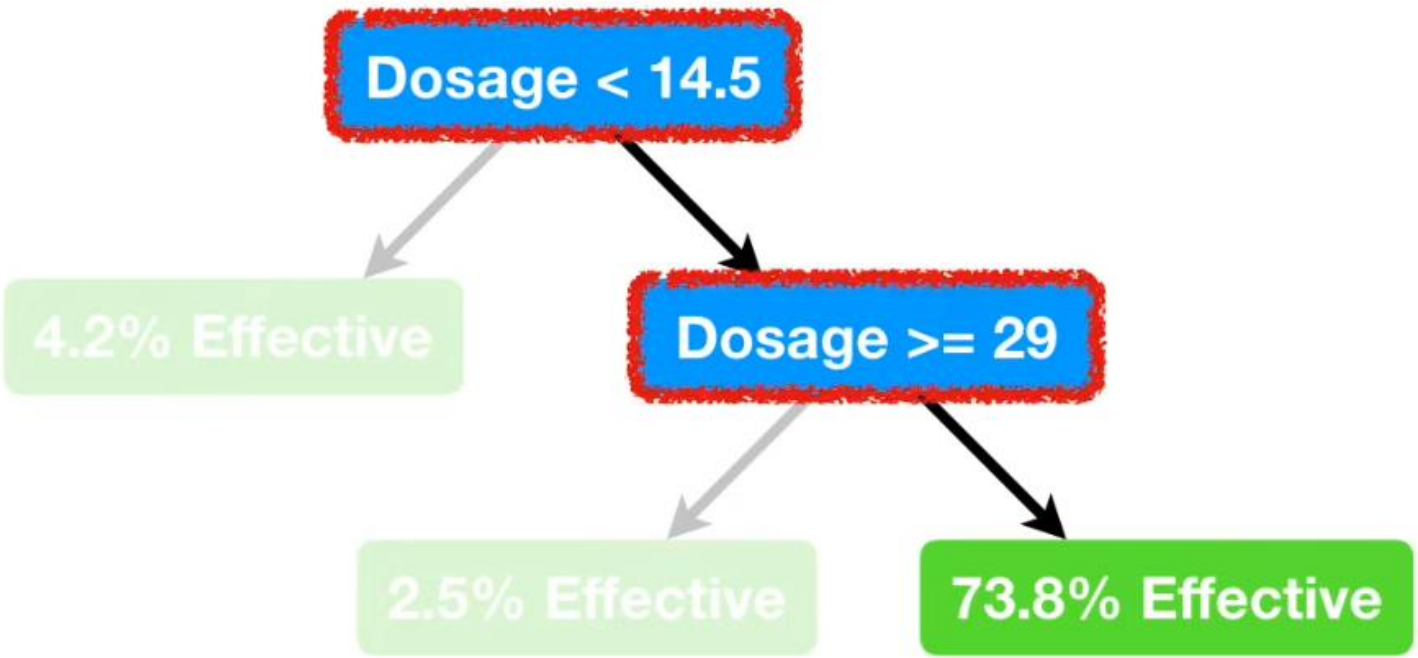
One way to prevent **Overfitting** a **Regression Tree** to the **Training Data** is to remove some of the leaves…



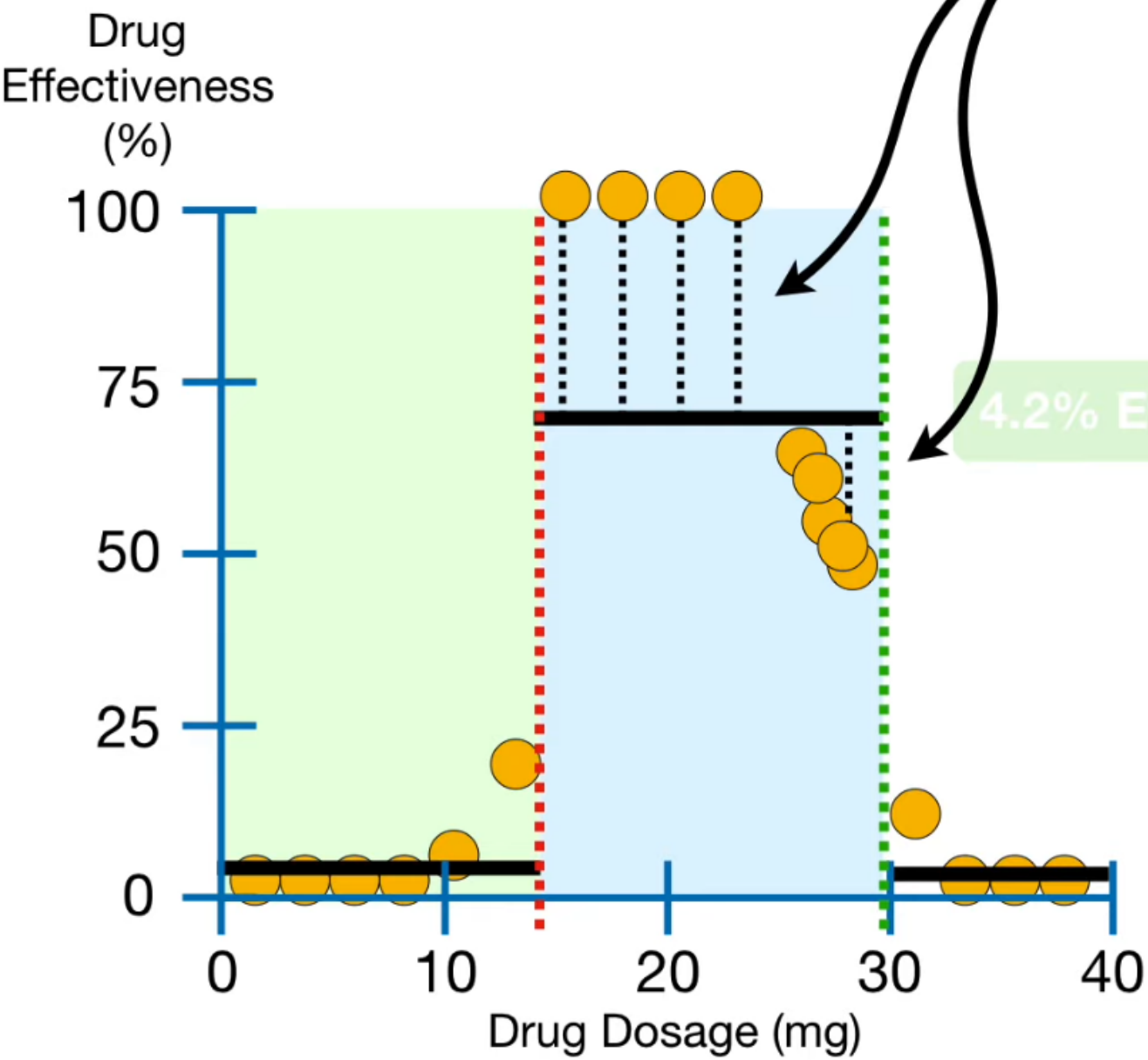


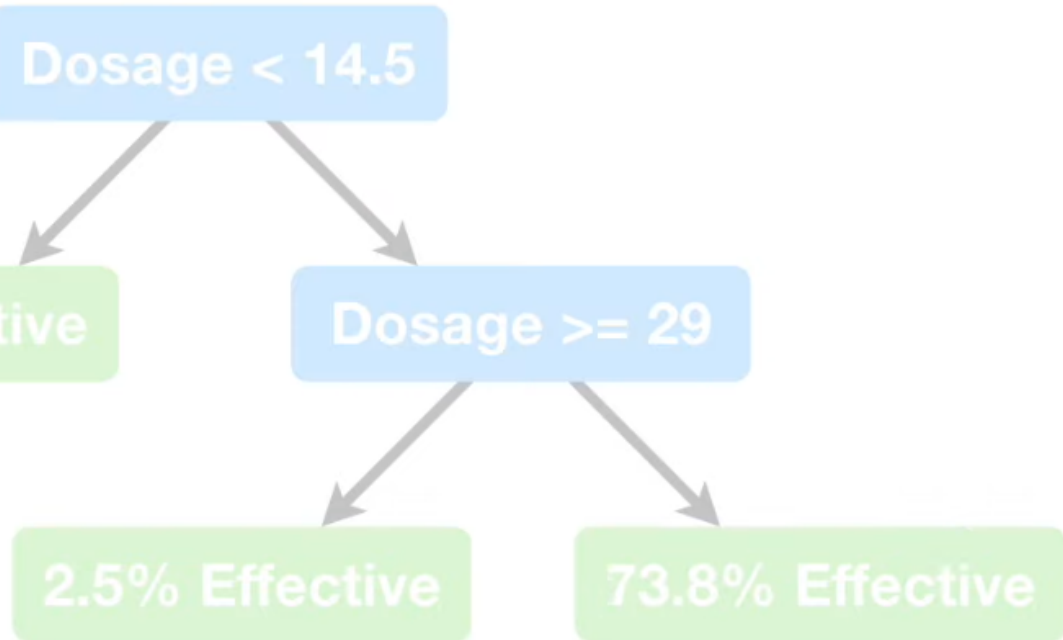
Now all of the observations between **14.5** and **29** go to the leaf on the far right.



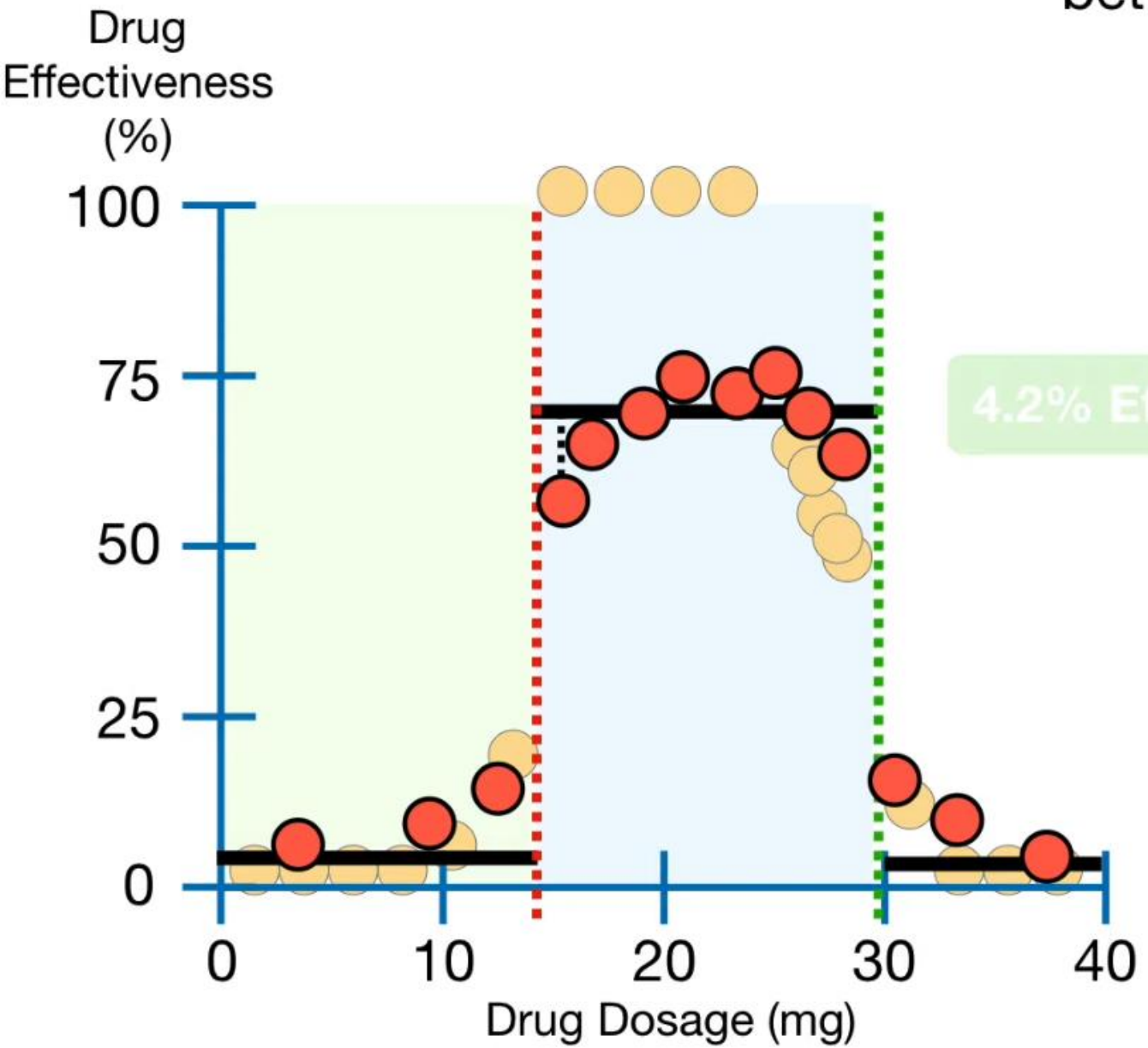


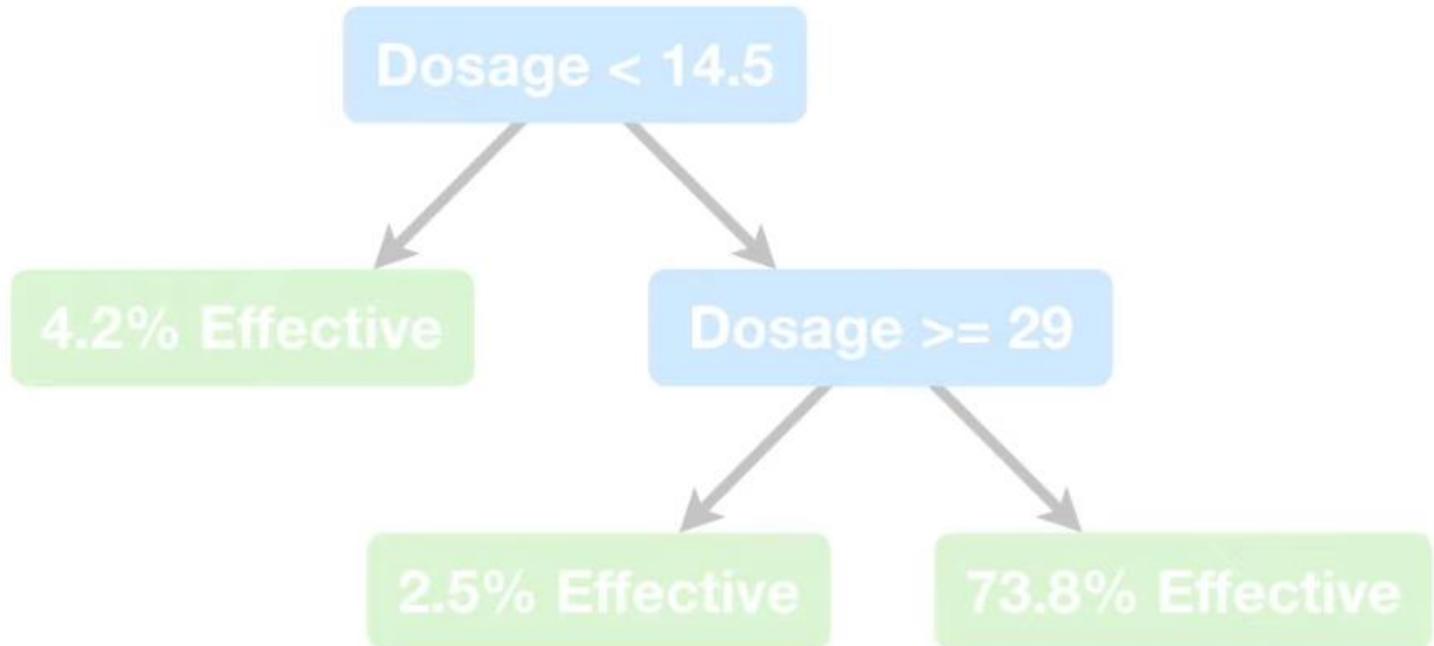
The large **Residuals** tell us that the new tree doesn't fit the **Training Data** as well as before…



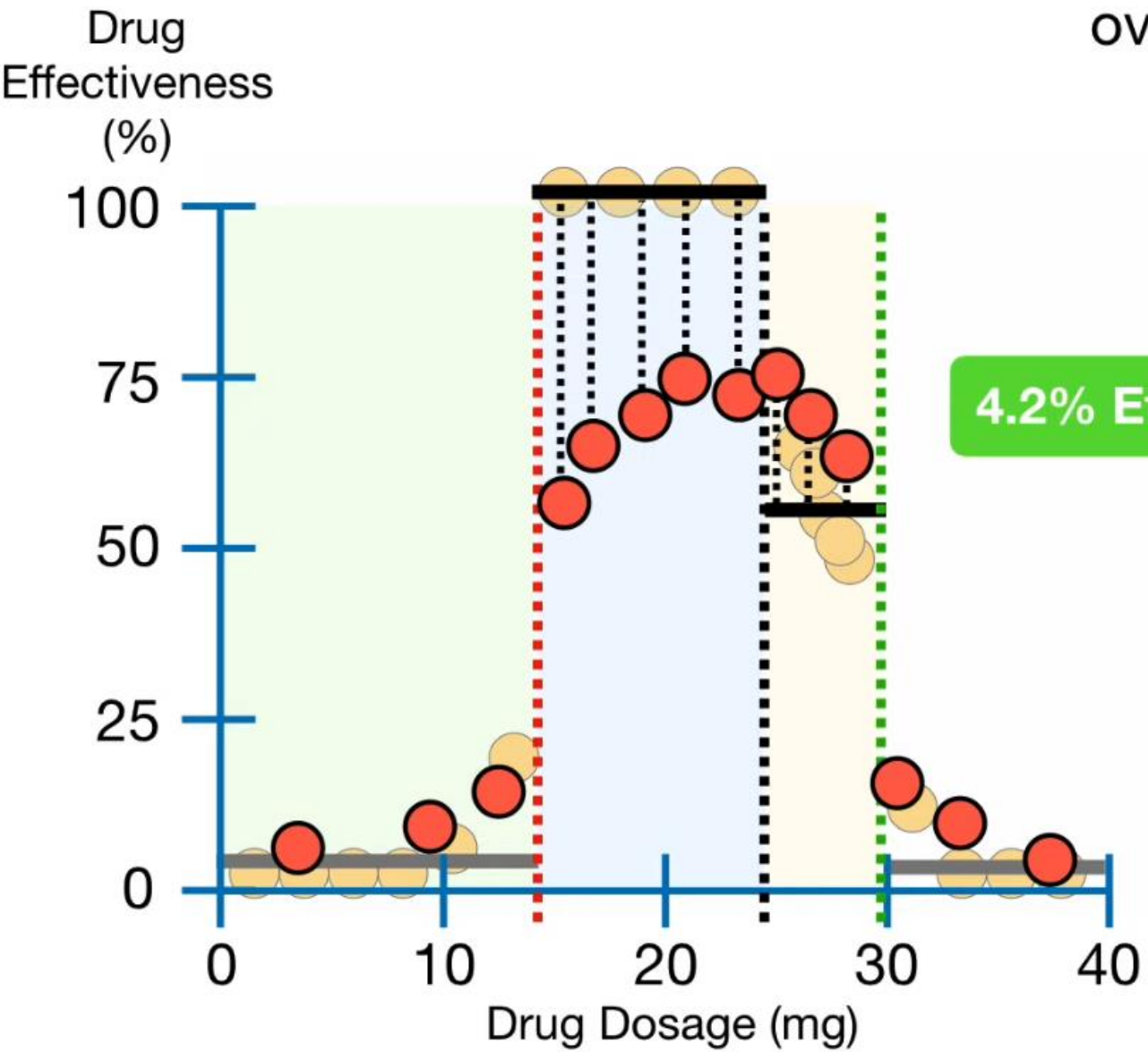


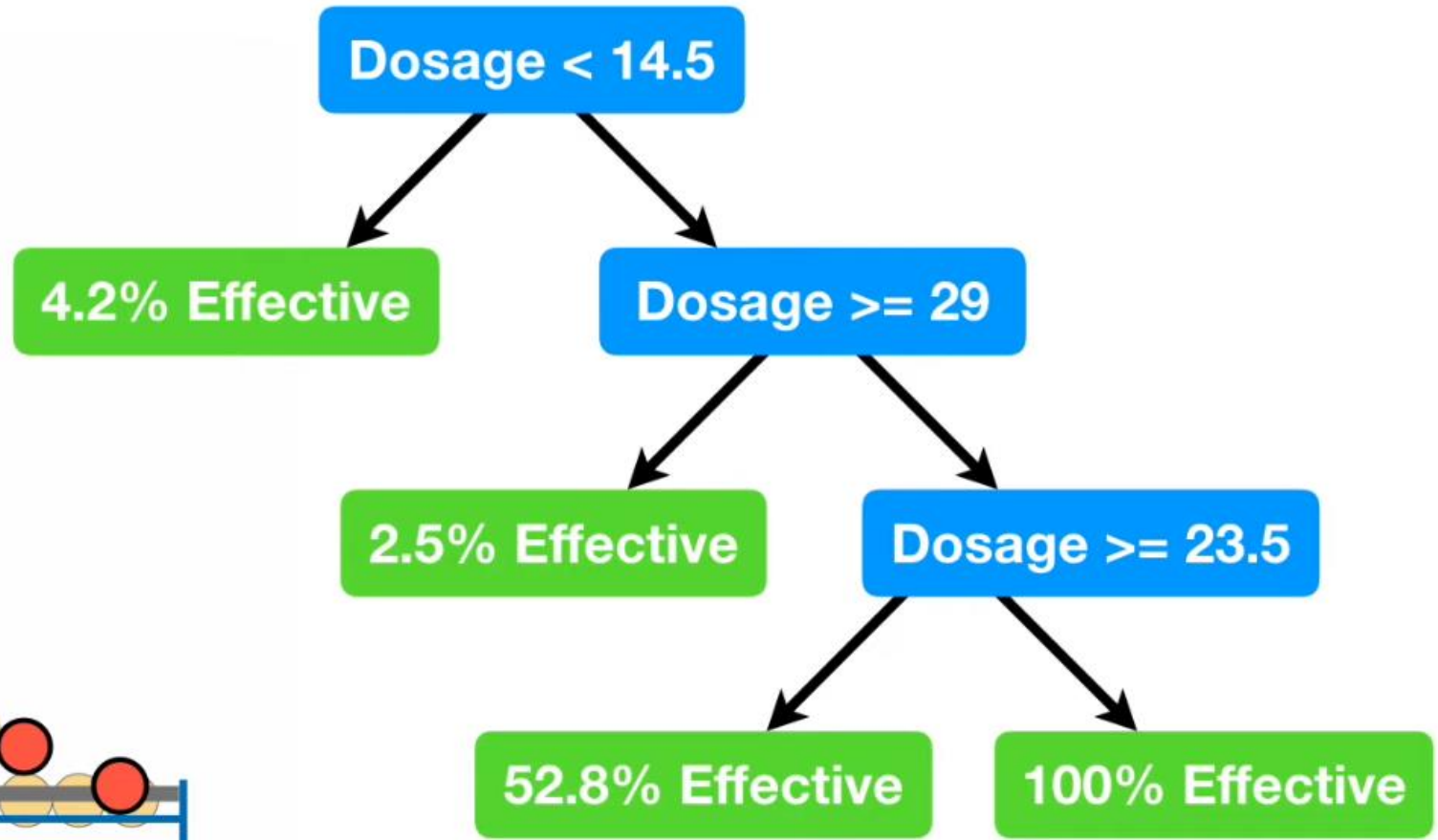
…but the new sub-tree does a much better job with the **Testing Data**.



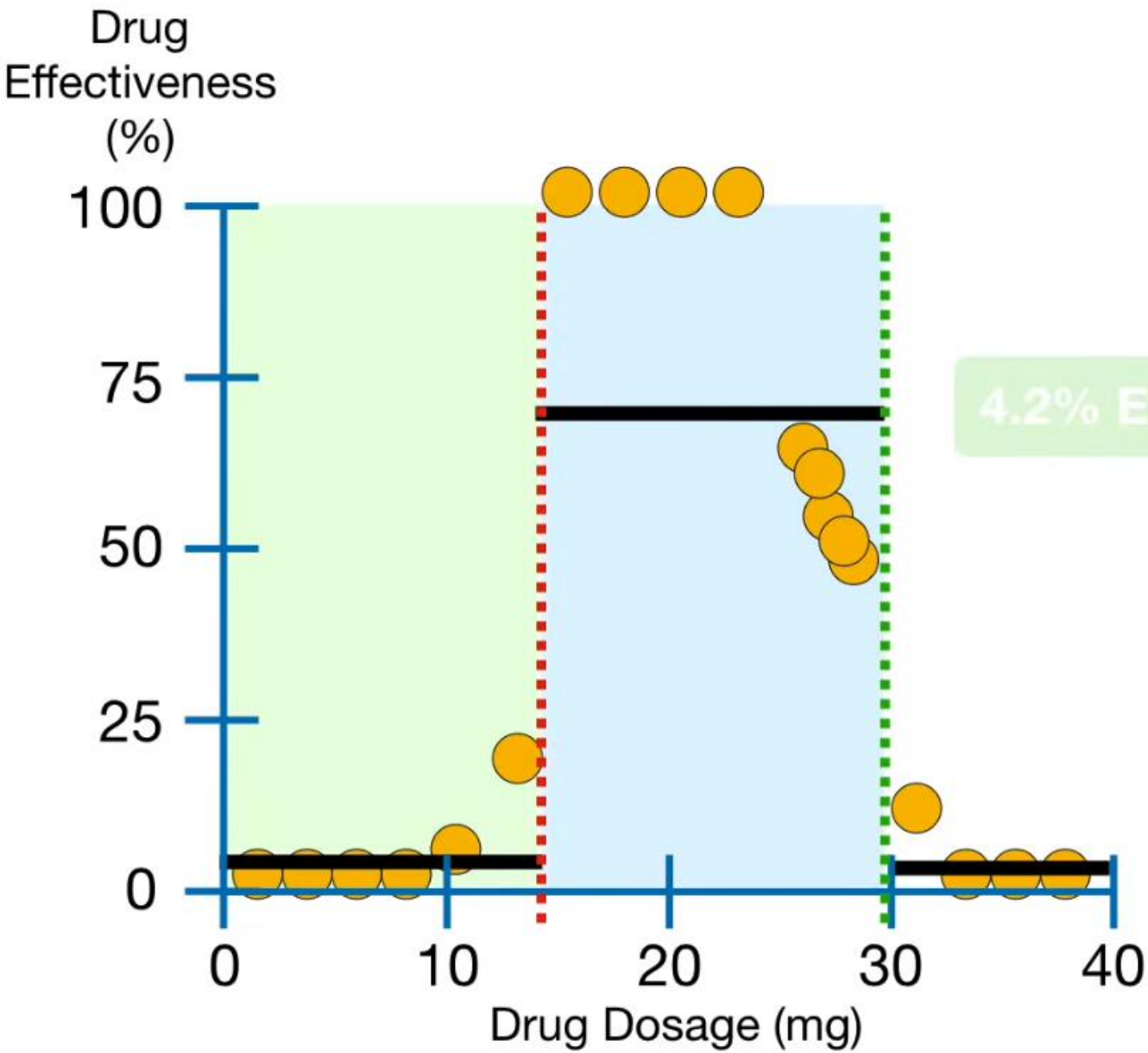


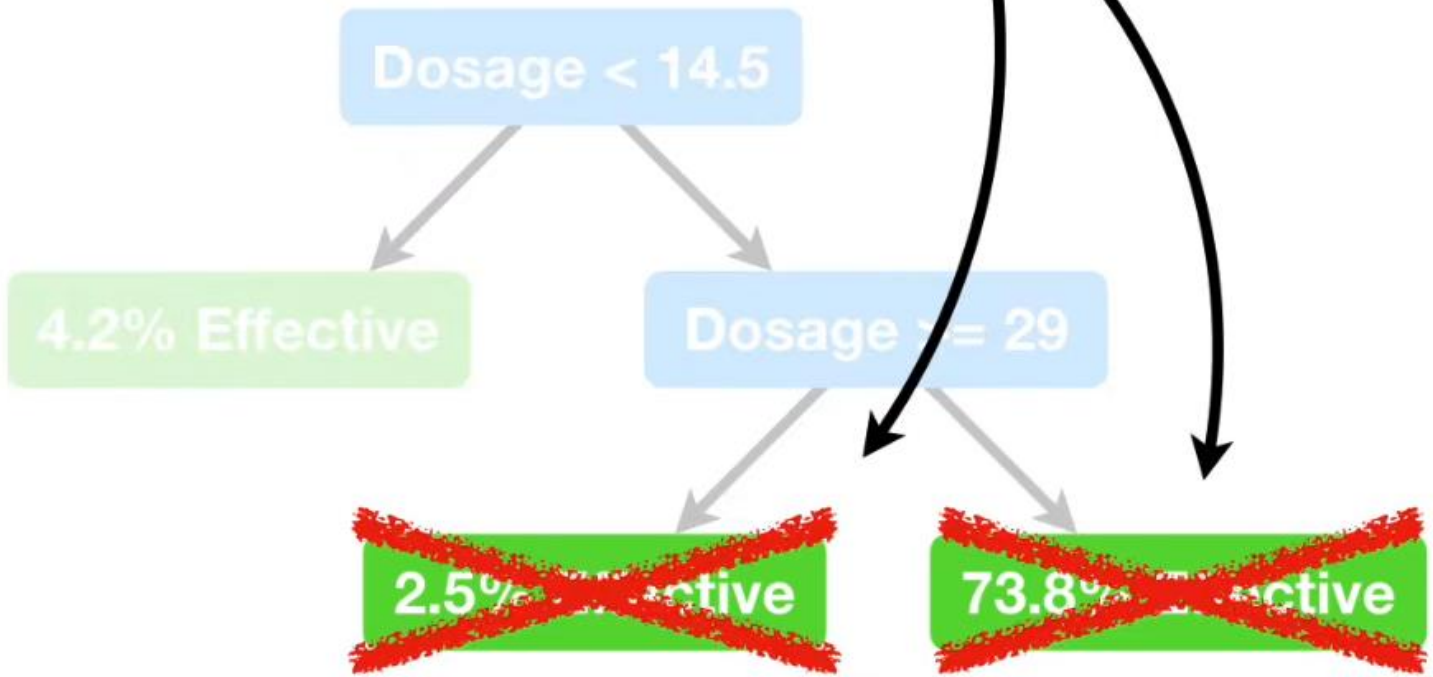
Thus, the main idea behind pruning a **Regression Tree** is to prevent overfitting the **Training Data**…



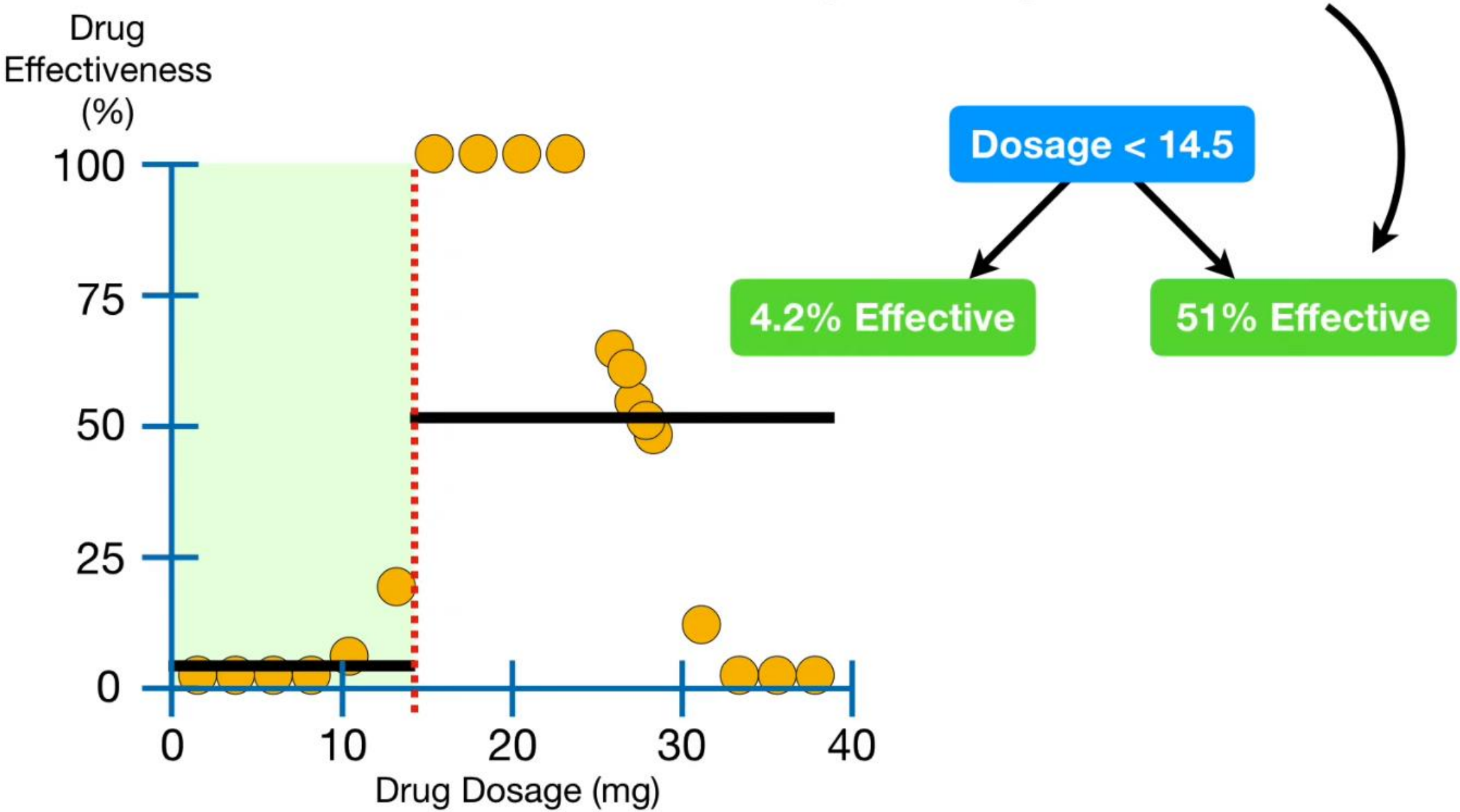


NOTE: If we wanted to prune the tree more, we could remove these two leaves…

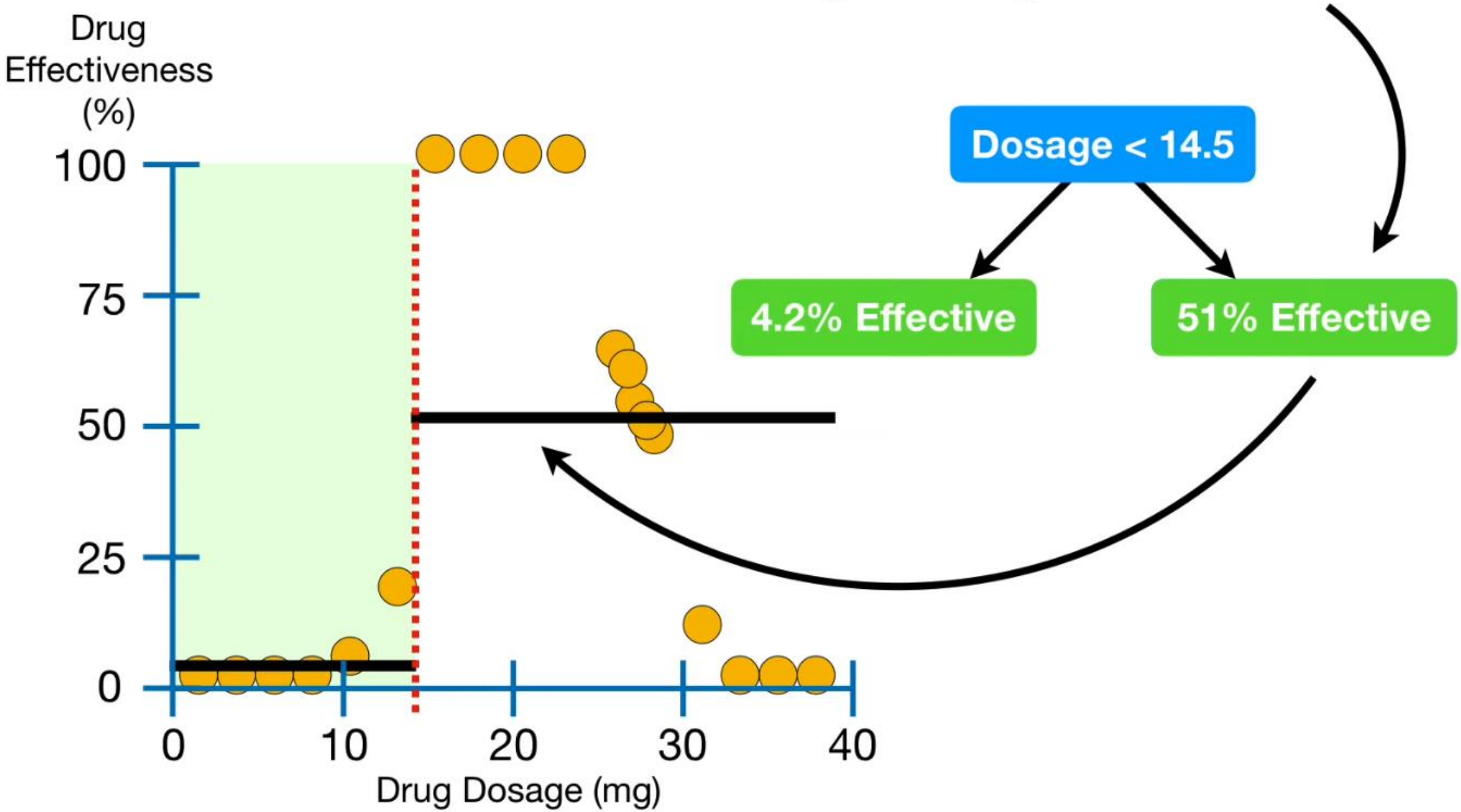


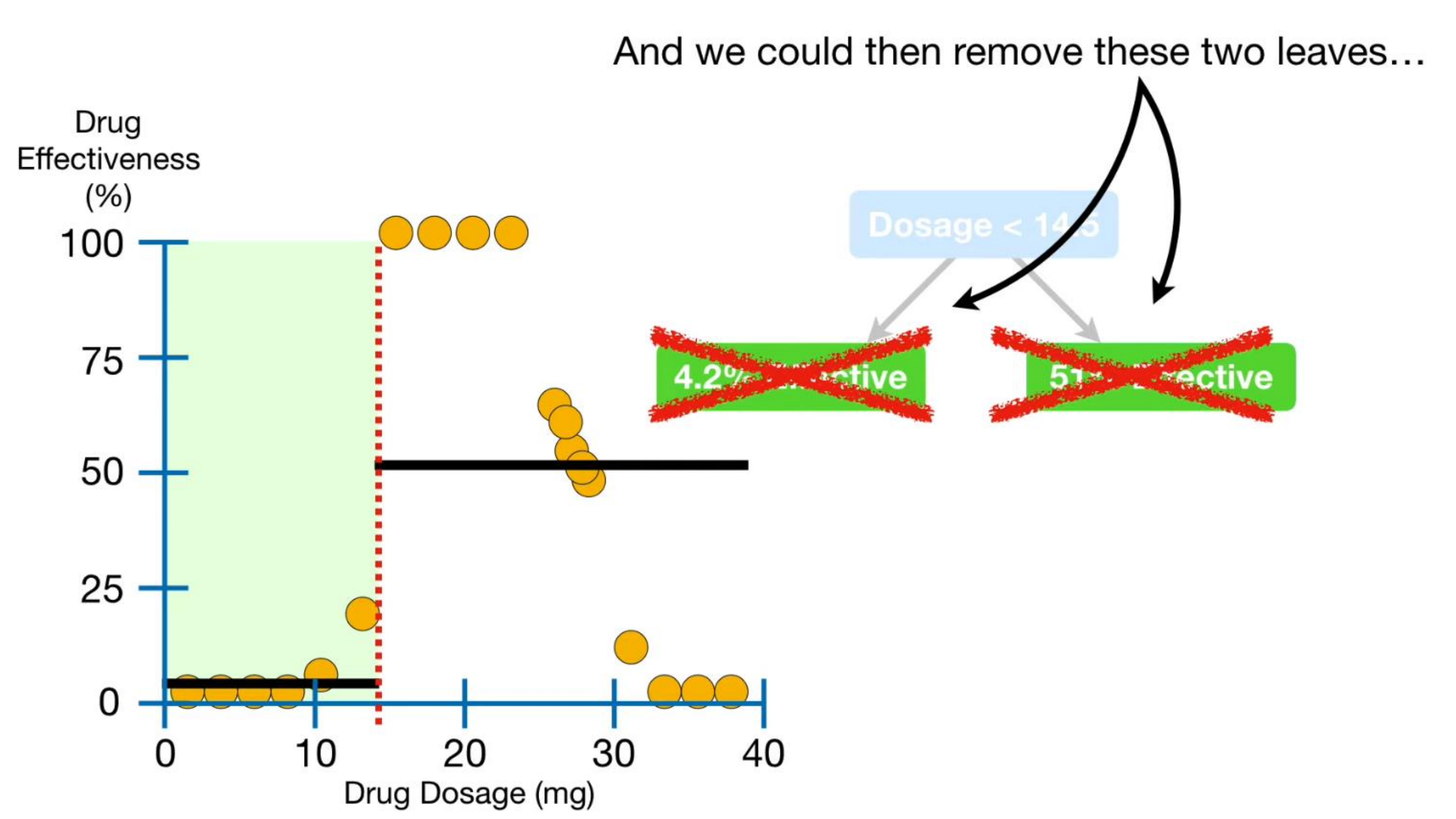


…and replace the split with a leaf that is the average of a larger number of observations.

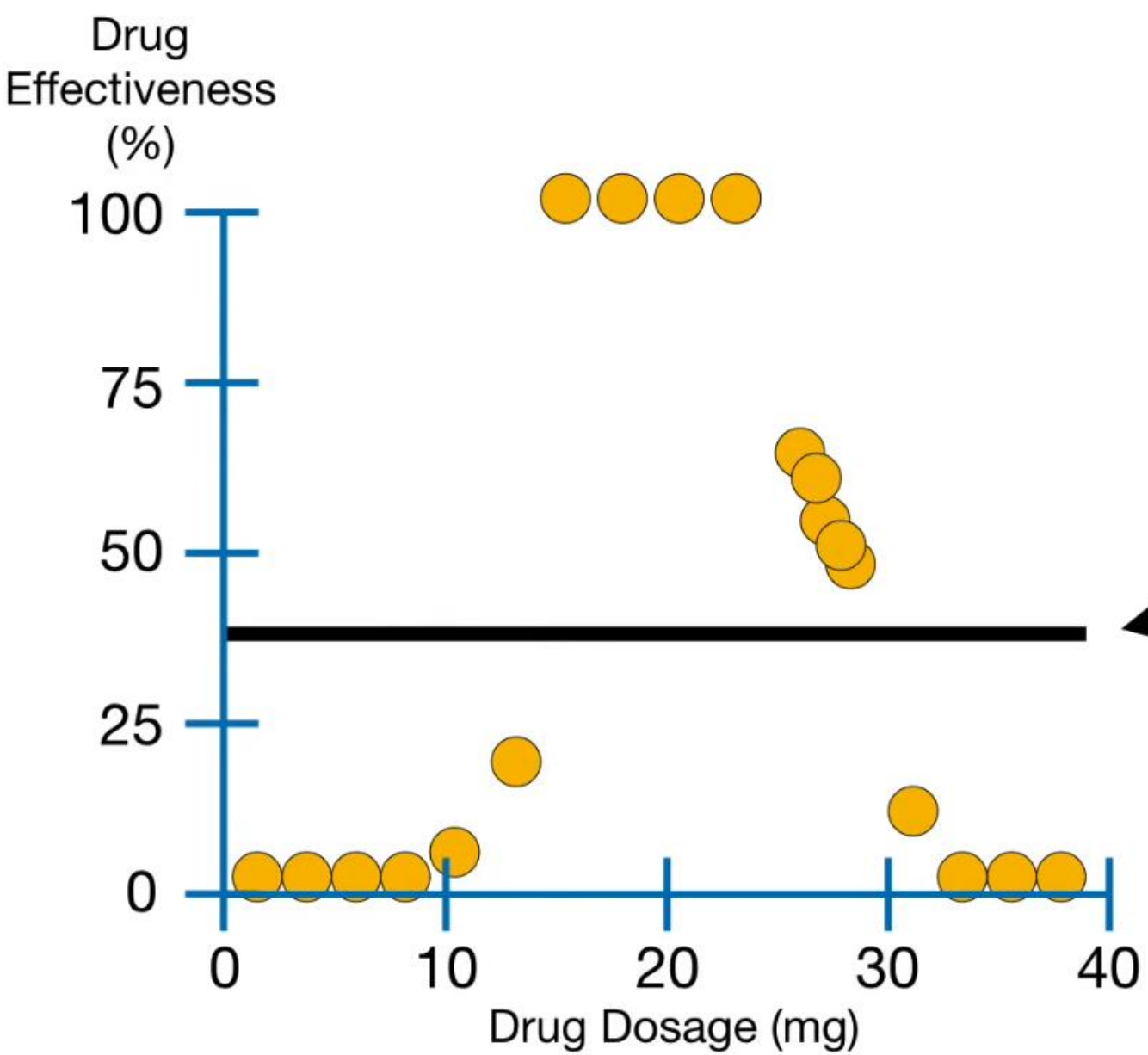


…and replace the split with a leaf that is the average of a larger number of observations.



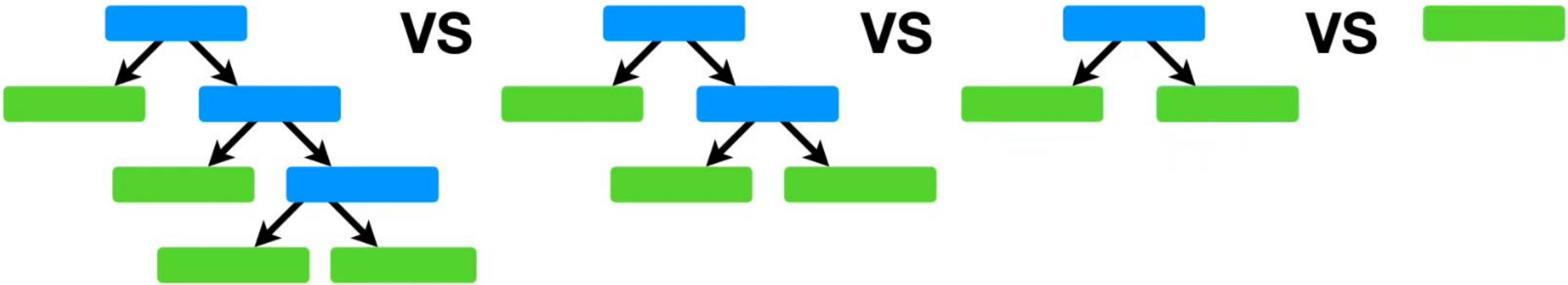

And we could then remove these two leaves…
Drug Effectiveness (%)
100
75
50
25
0
0
10
20
30
40
Drug Dosage (mg)
Dosage < 14.5
4.2%
ctive
51
ective

…and replace the split with a leaf that is the average of all of the observations.



So the question is:

“How do we decide which tree to use?”



2. Tree pruning

Sum of Squared Residuals = $$\sum_{m=1}^{|T|} \sum_{i:\, x_i \in R_m} (y_i - \hat{y}_{R_m})^2$$

2. Tree pruning

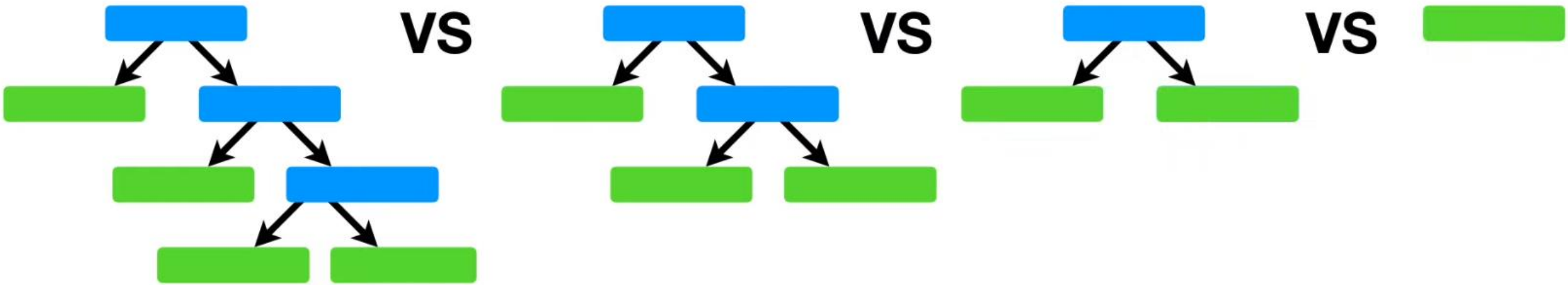
- A better strategy is to grow a very large tree *T*0, and then *prune* it back in order to obtain a *subtree*
- *Cost complexity pruning* | also known as *weakest link pruning* | is used to do this
- we consider a sequence of trees indexed by a nonnegative tuning parameter α. For each value of α there corresponds a subtree $T \subset T0$ such that

$$\sum_{m=1}^{|T|} \sum_{i:\, x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

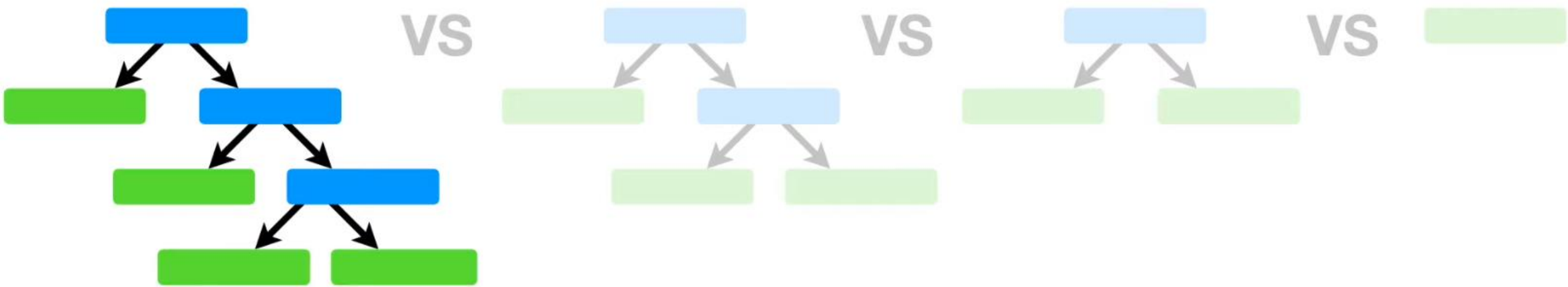
is as small as possible. Here |T| indicates the number of terminal nodes of the tree *T*, R_m is the rectangle (i.e. the subset of predictor space) corresponding to the *m*th terminal node, and ^ y_{Rm} is the mean of the training observations in R_m.

So the question is:

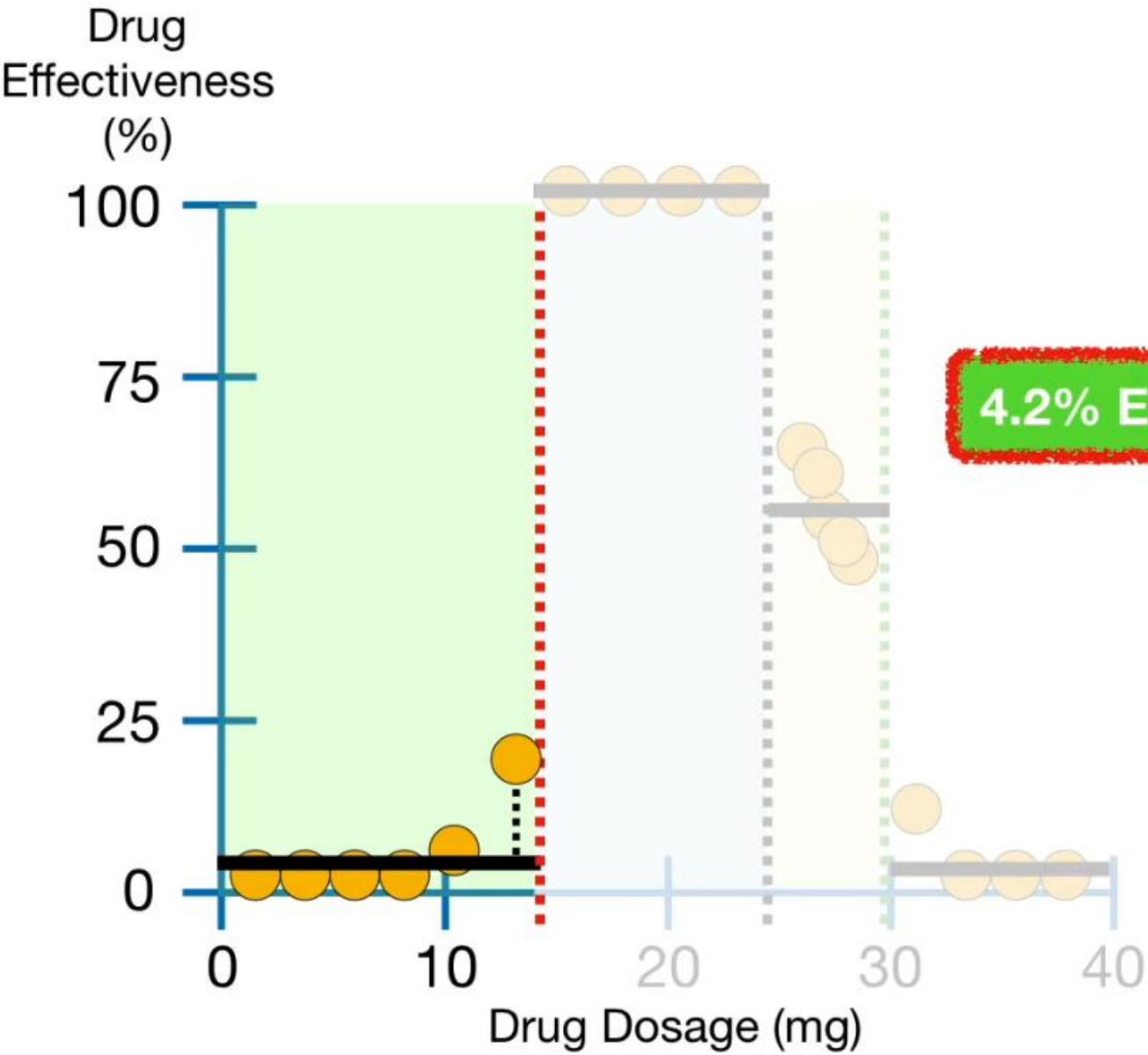
“How do we decide which tree to use?”



The first step in **Cost Complexity Pruning** is to calculate the **Sum of Squared Residuals** for each tree.



The **Sum of Squared Residuals** for the **Observations** with **Dosages < 14.5** is…



Dosage < 14.5

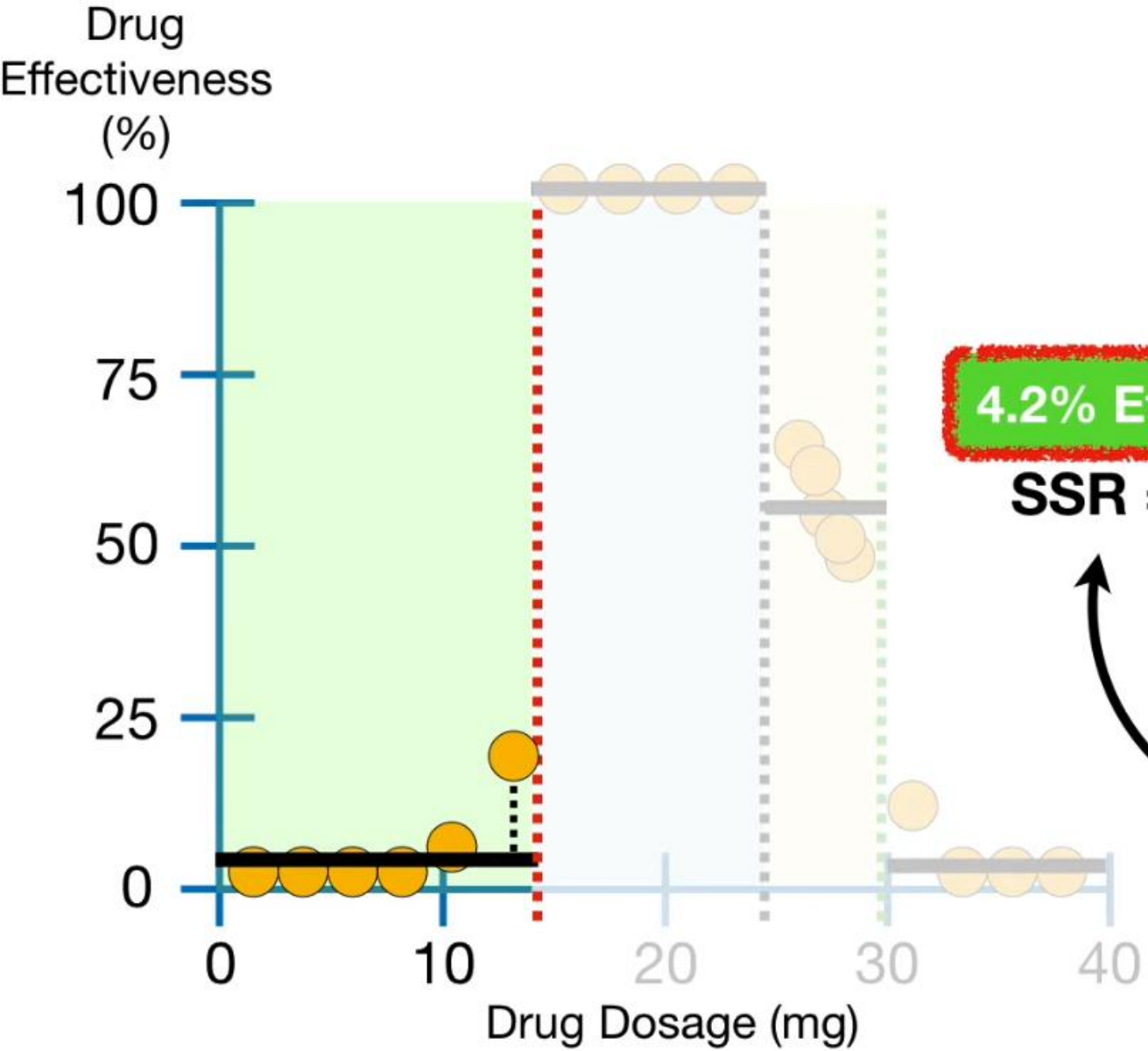
4.2% Effective

$$(0 - 4.2)^2 + (0 - 4.2)^2 + (0 - 4.2)^2 + (0 - 4.2)^2$$

$$+ (5 - 4.2)^2 + (20 - 4.2)^2$$

$$= 320.8$$

So we'll save that **Sum of Squared Residuals** (**SSR**) underneath the corresponding leaf.



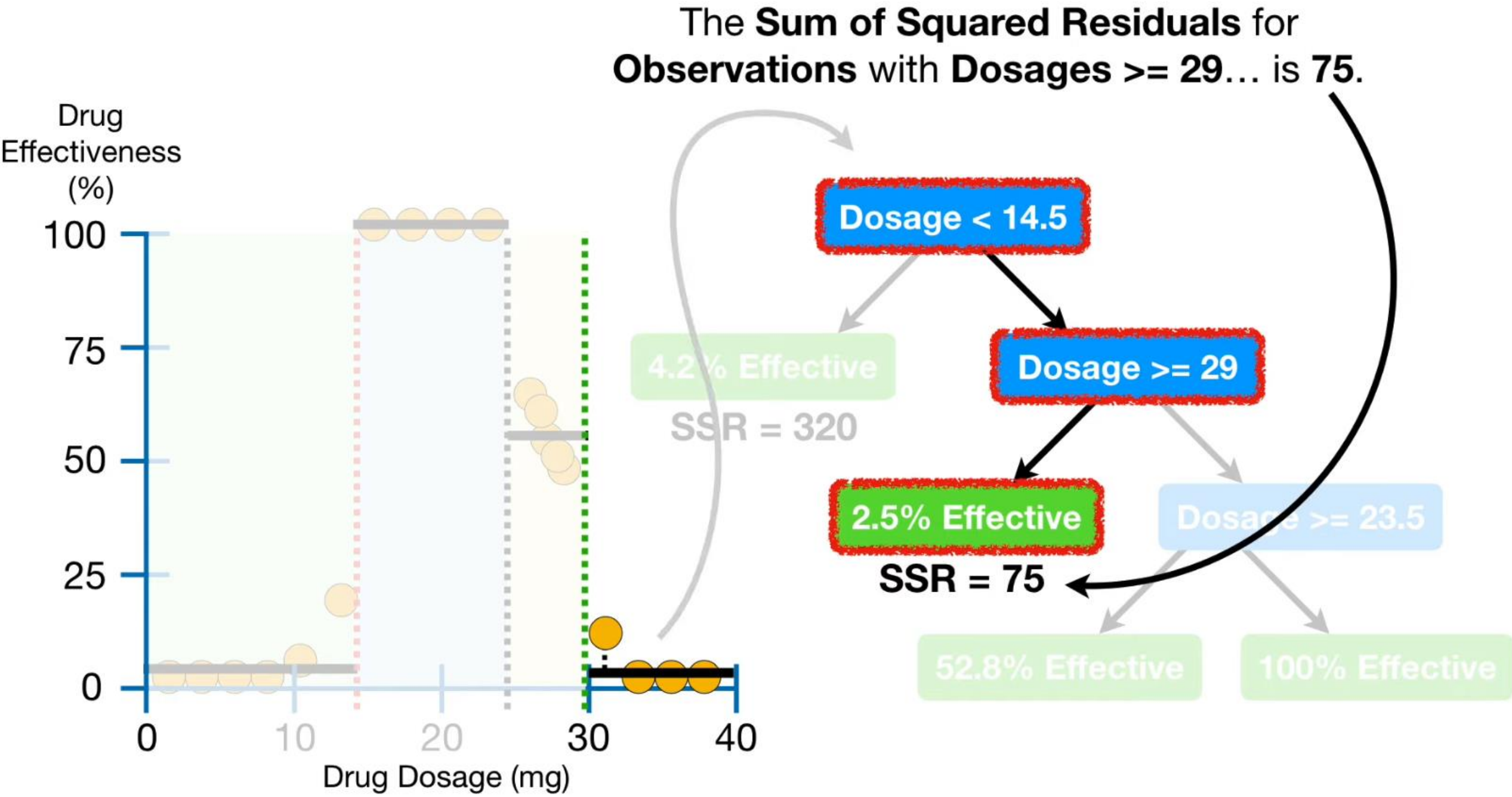
Dosage < 14.5

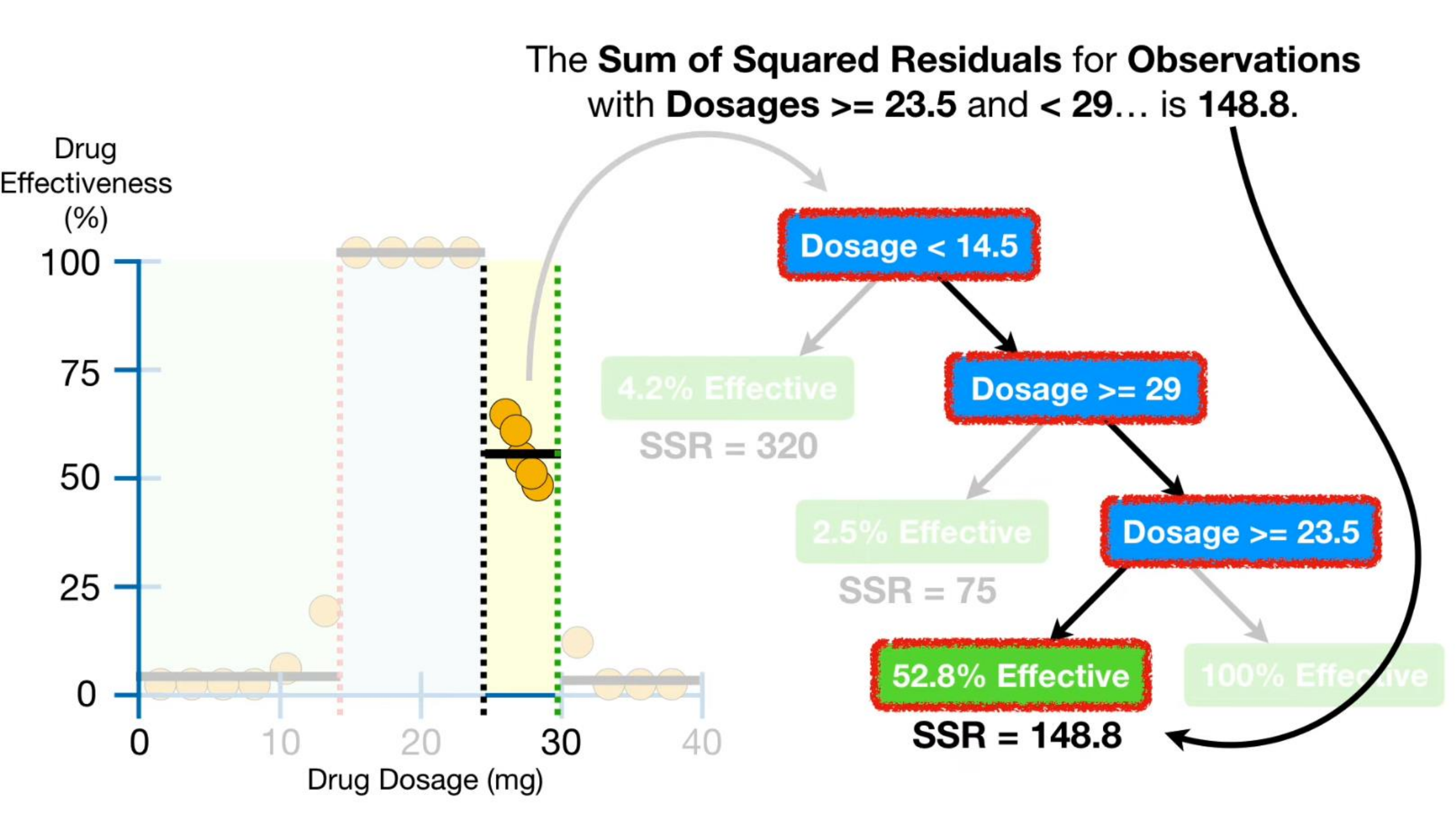
4.2% Effective

SSR = 320

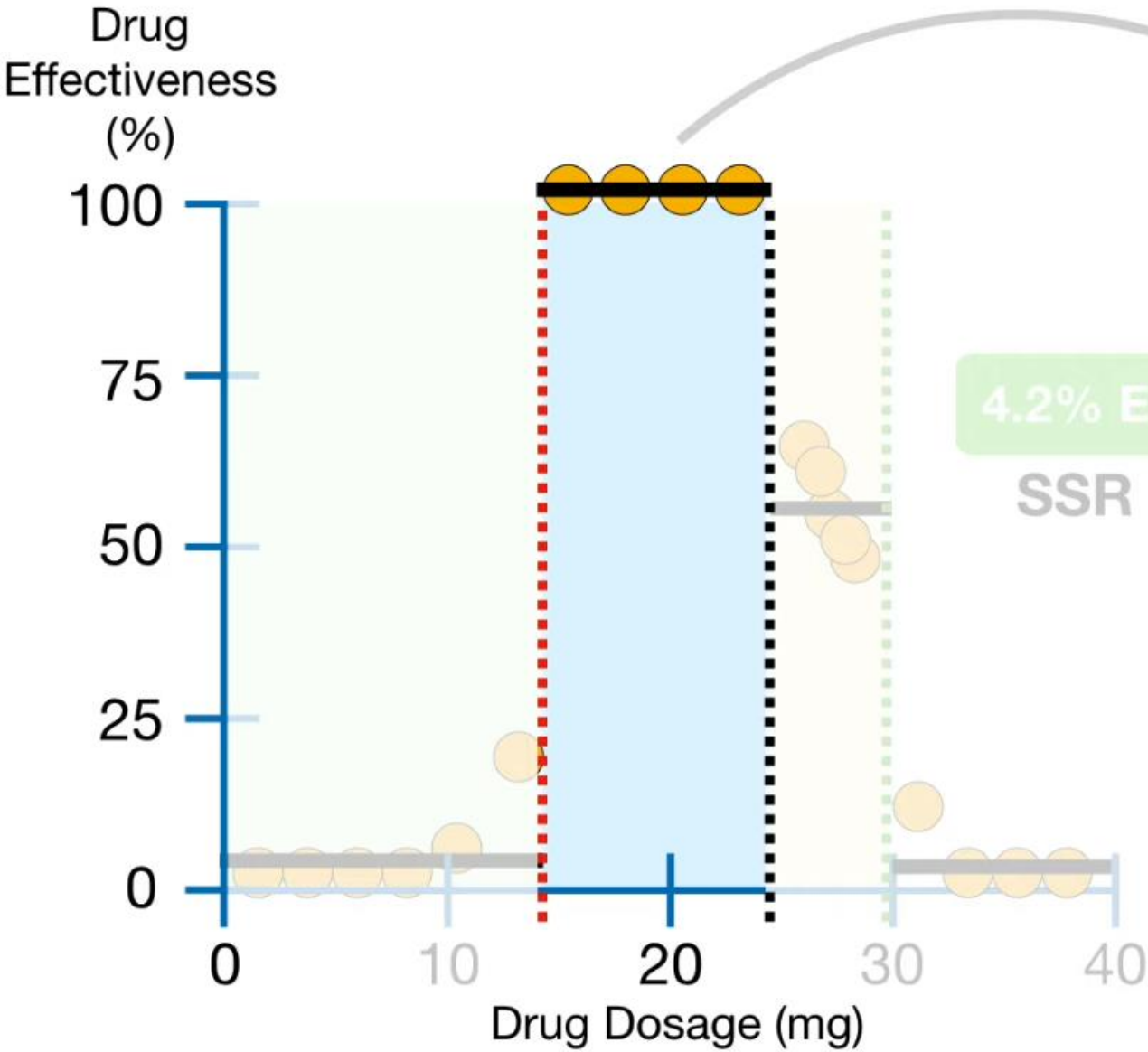
$$(0 - 4.2)^2 + (0 - 4.2)^2 + (0 - 4.2)^2 + (0 - 4.2)^2 + (5 - 4.2)^2 + (20 - 4.2)^2$$

= 320.8


The **Sum of Squared Residuals** for **Observations** with **Dosages >= 29**... is **75**.
Drug Effectiveness (%)
100
75
50
25
0
0
10
20
30
40
Drug Dosage (mg)
Dosage < 14.5
4.2% Effective
SSR = 320
Dosage >= 29
2.5% Effective
SSR = 75
Dosage >= 23.5
52.8% Effective
100% Effective


The Sum of Squared Residuals for Observations with Dosages >= 23.5 and < 29... is 148.8.
Drug Effectiveness (%)
100
75
50
25
0
0
10
20
30
40
Drug Dosage (mg)
Dosage < 14.5
4.2% Effective
SSR = 320
Dosage >= 29
2.5% Effective
SSR = 75
Dosage >= 23.5
52.8% Effective
100% Effective
SSR = 148.8

And the **Sum of Squared Residuals** for **Observations** with **Dosages >= 14.5** and **< 23.5**… is **0**.



Dosage < 14.5

4.2% Effective

SSR = 320

Dosage >= 29

2.5% Effective

SSR = 75

Dosage >= 23.5

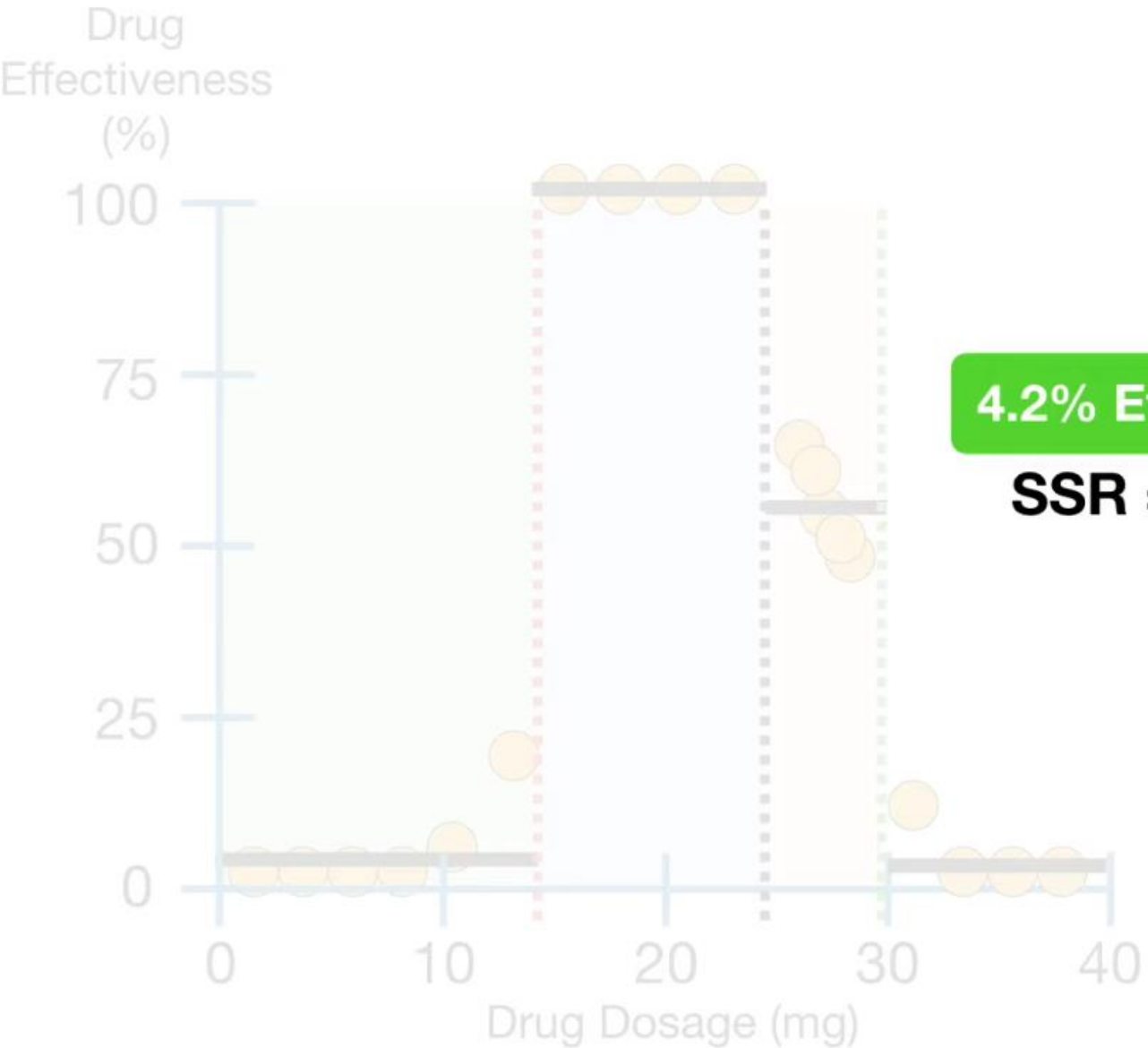
52.8% Effective

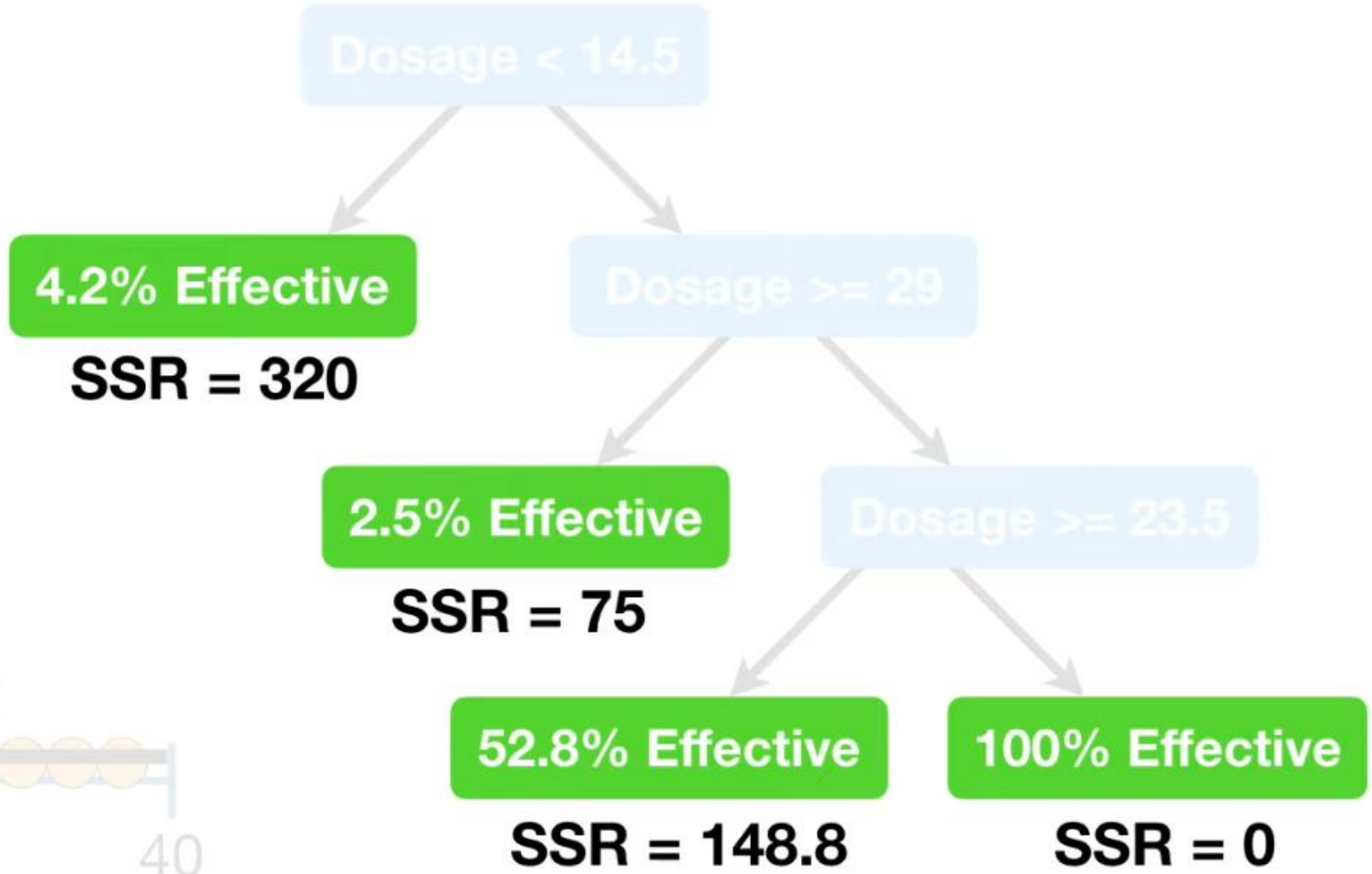
SSR = 148.8

100% Effective

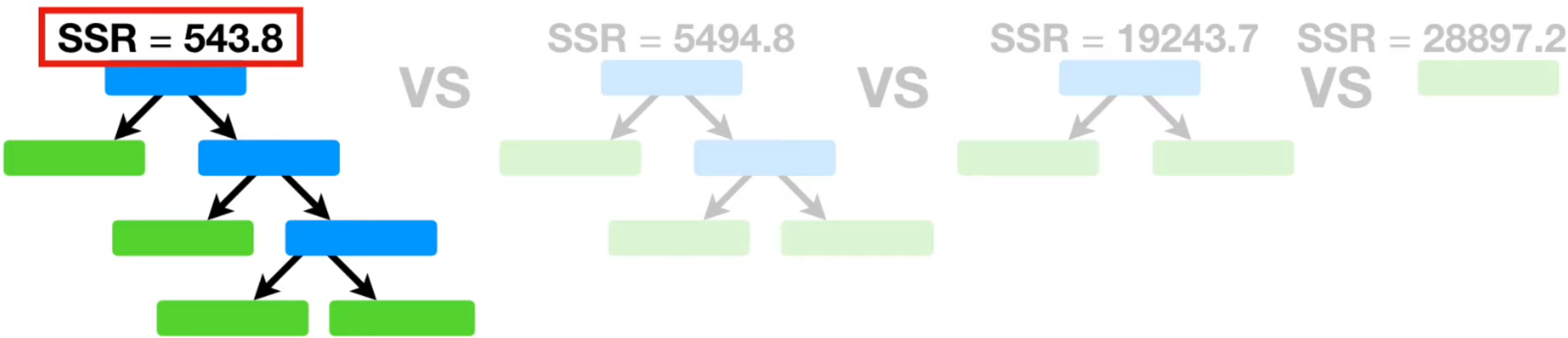
SSR = 0

Thus, the total **Sum of Squared Residuals** (SSR) for the whole tree is 320 + 75 + 148.8 + 0 = **543.8**

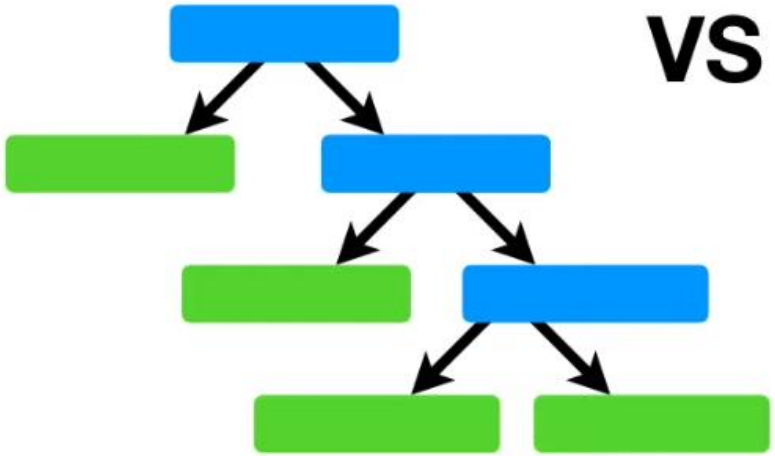




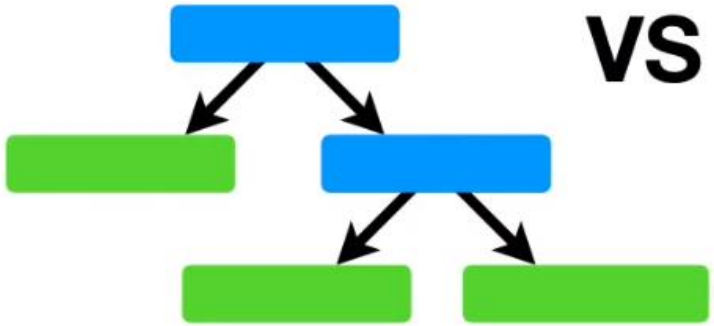
NOTE: The **Sum of Squared Residuals** is relatively small for the original, full sized tree…



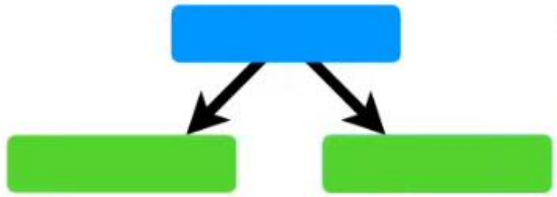
SSR = 543.8



SSR = 5494.8



SSR = 19243.7



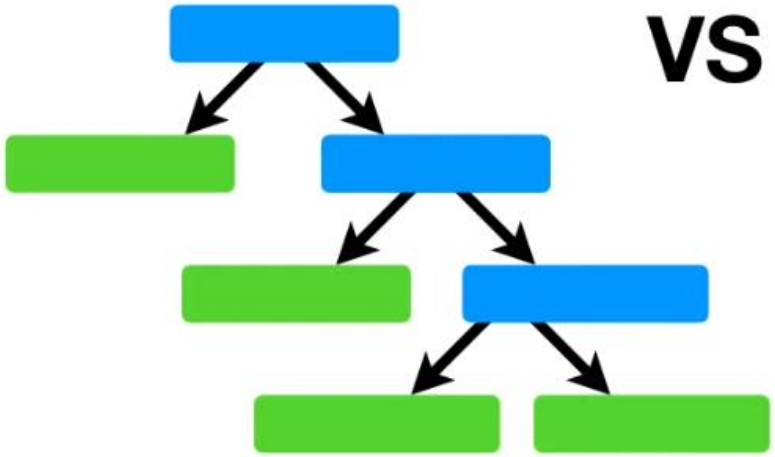
SSR = 38897.2

- Tree score = SSR + α * |T|
- α = 0 -> +∞

Subtree	SSR	Tree score (SSR + alpha*number of leaves)
4 leaves	543.8	
3 leaves	5494.8	
2 leaves	19243.7	
1 leaf	38897.2	

WHICH IS THE OPTIMAL ALPHA PARAMETER?

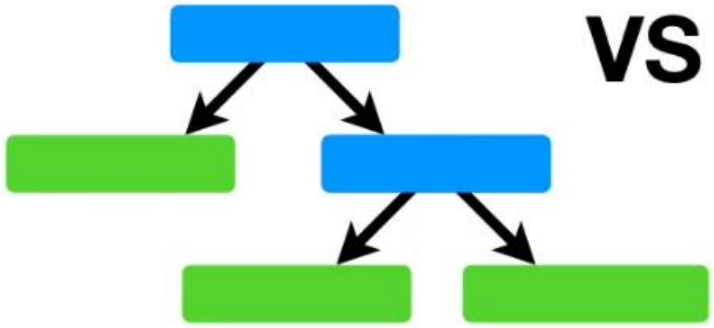
SSR = 543.8



$\alpha = 0$

VS

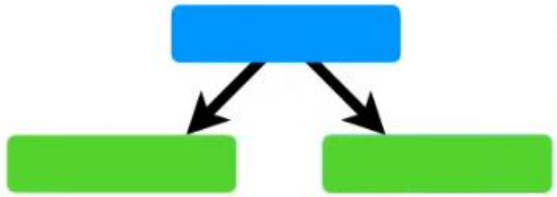
SSR = 5494.8



$\alpha = 10000$

VS

SSR = 19243.7



$\alpha = 15000$

VS

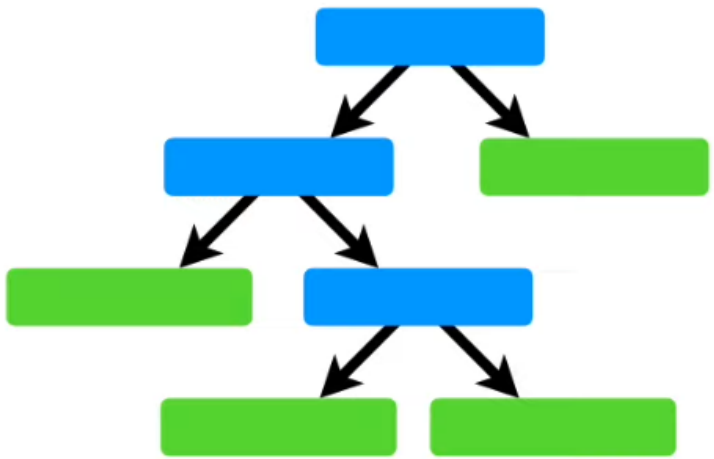
SSR = 38897.2

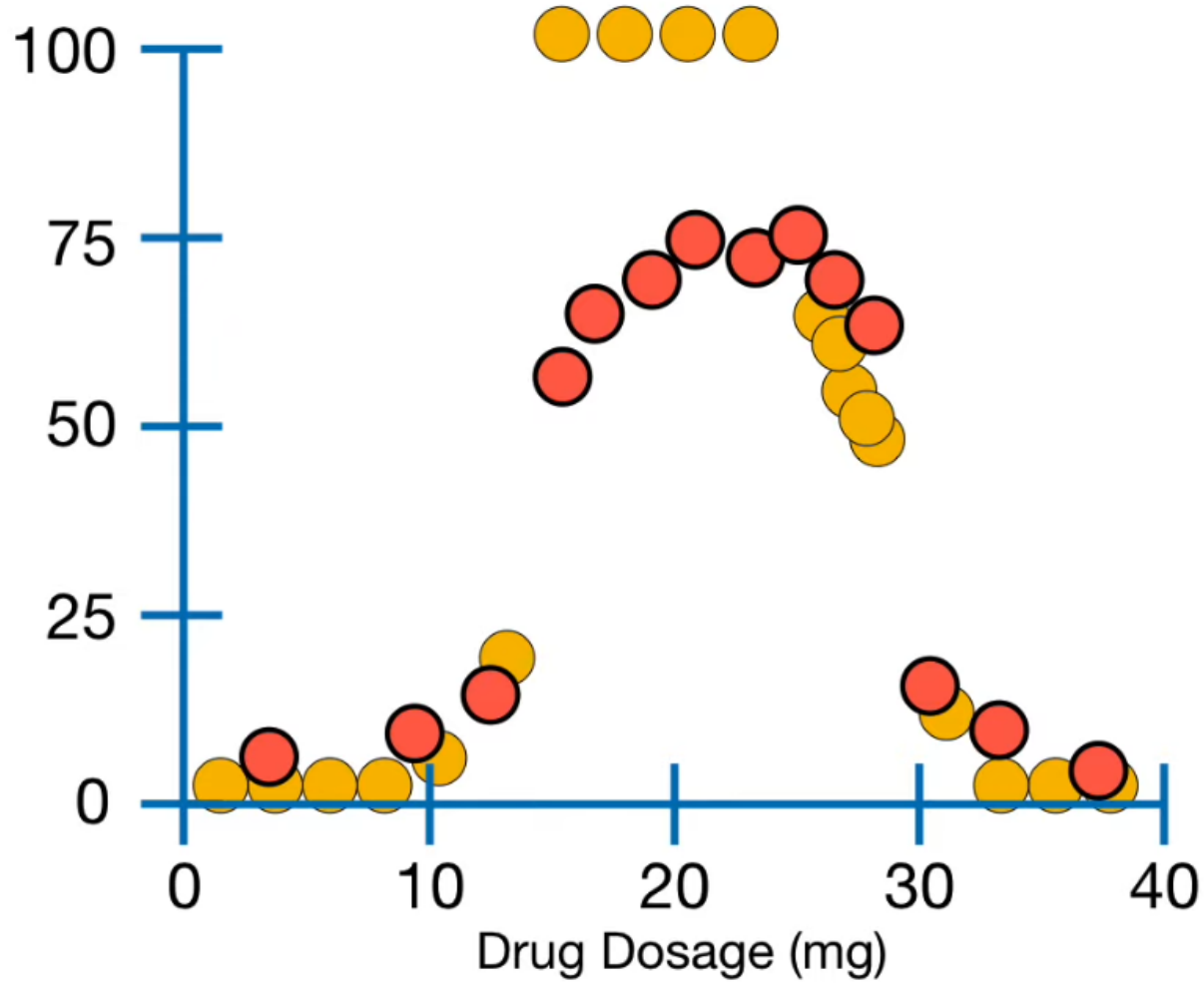
$\alpha = 22000$

2. Tree pruning

- The tuning parameter $\hat{\alpha}$ controls a trade-off between the subtree's complexity and its fit to the training data.
- We select an optimal value using **cross-validation.**
- We then return to the full data set and obtain the subtree corresponding to $\hat{\alpha}$

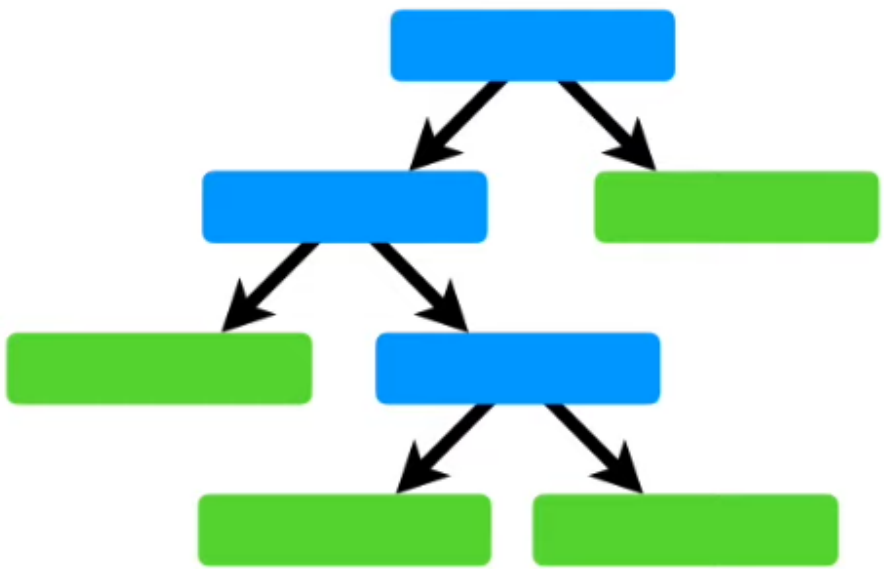
…build a full sized **Regression Tree**.





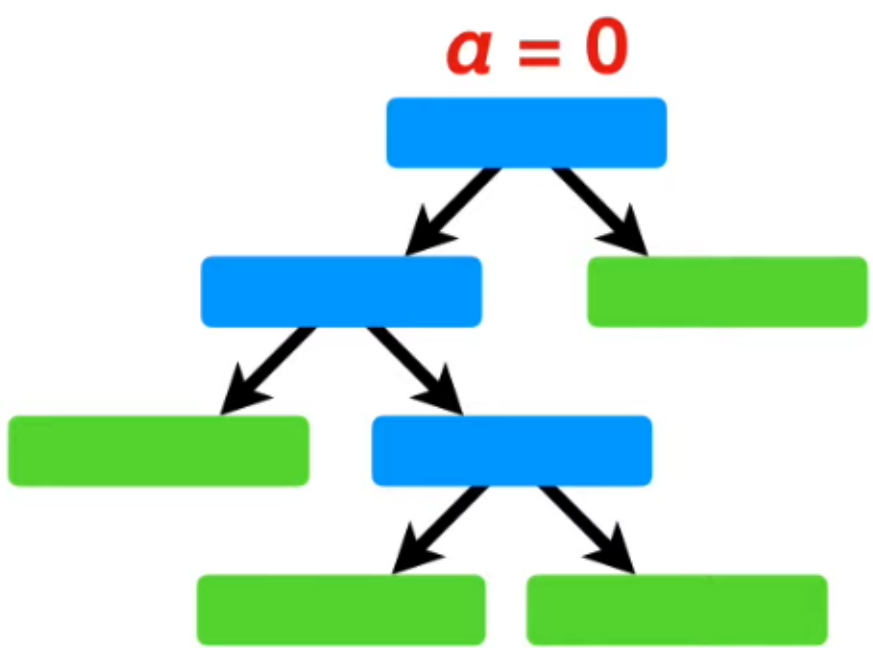
ALSO NOTE: This full sized tree has the lowest **Tree Score** when $\alpha = 0$.

Tree Score = **SSR** + αT



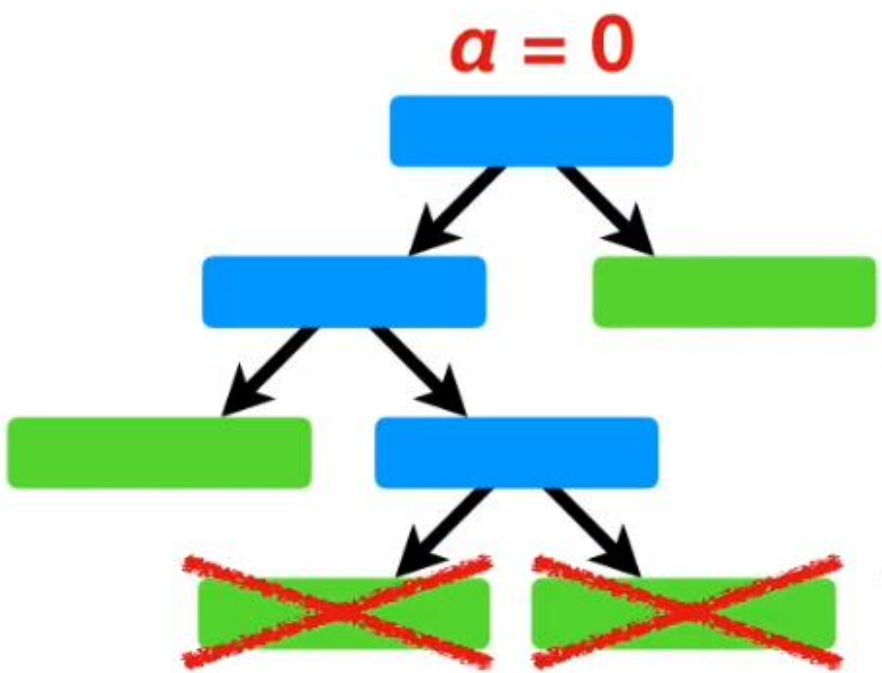
So let's put $\alpha = 0$ here, to remind us that this tree has the lowest **Tree Score** when $\alpha = 0$.

Tree Score = SSR



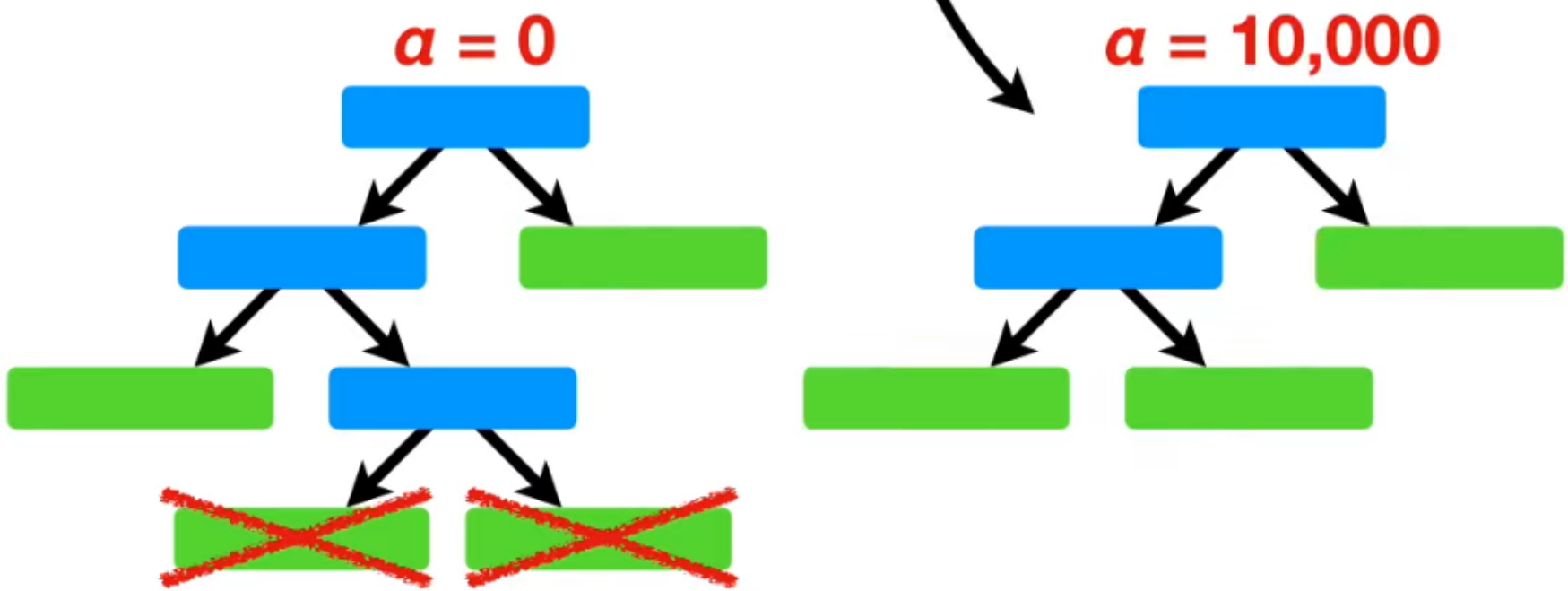
Now we increase α until pruning leaves will give us a lower **Tree Score**.

Tree Score = **SSR** + αT



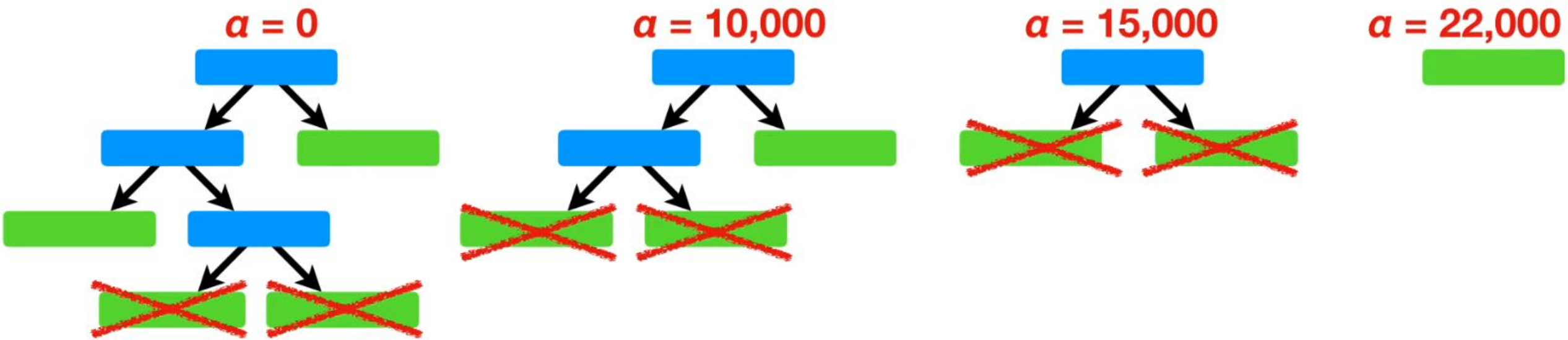
…and use this sub-tree.

Tree Score = SSR + αT

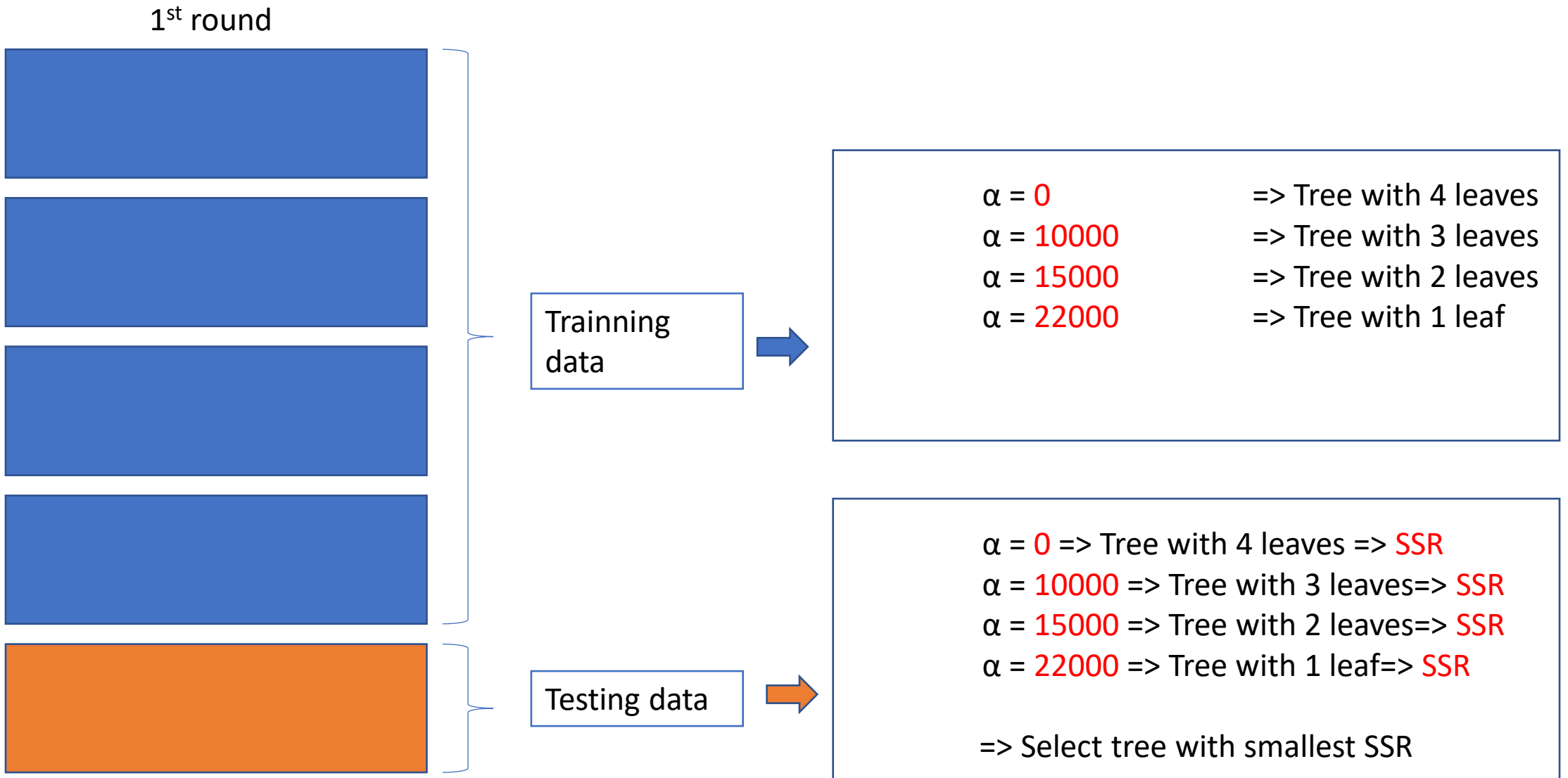


…and use this sub-tree instead.

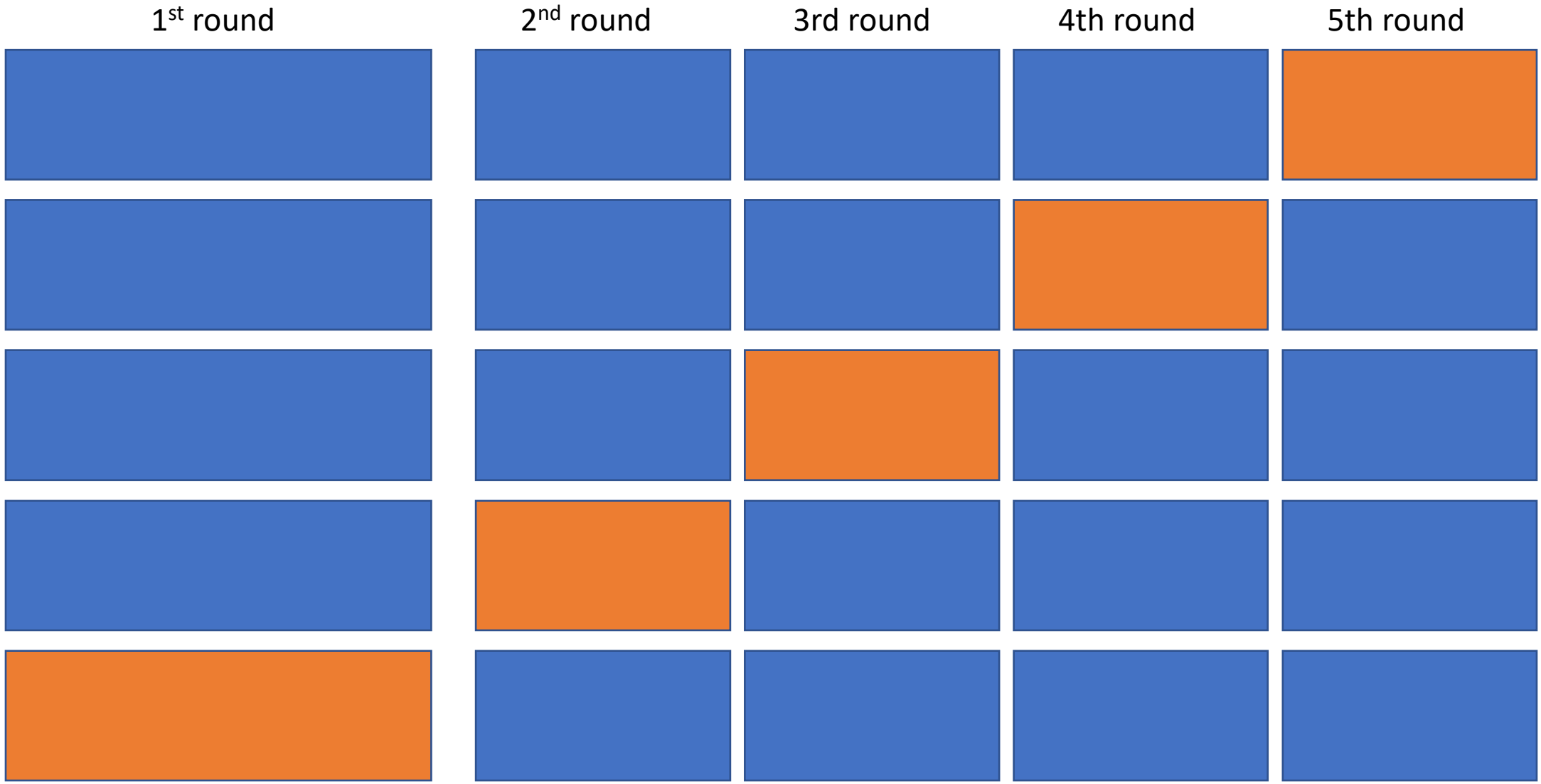
Tree Score = SSR + αT



k-fold cross-validation



k-fold cross-validation



k-fold cross-validation

1st round

α = 0 => Tree with 4 leaves => SSR
α = 10000 => Tree with 3 leaves=> SSR
α = 15000 => Tree with 2 leaves=> SSR
α = 22000 => Tree with 1 leaf=> SSR

2nd round

α = 0 => Tree with 4 leaves => SSR
α = 10000 => Tree with 3 leaves=> SSR
α = 15000 => Tree with 2 leaves=> SSR
α = 22000 => Tree with 1 leaf=> SSR

3rd round

α = 0 => Tree with 4 leaves => SSR
α = 10000 => Tree with 3 leaves=> SSR
α = 15000 => Tree with 2 leaves=> SSR
α = 22000 => Tree with 1 leaf=> SSR

4th round

α = 0 => Tree with 4 leaves => SSR
α = 10000 => Tree with 3 leaves=> SSR
α = 15000 => Tree with 2 leaves=> SSR
α = 22000 => Tree with 1 leaf=> SSR

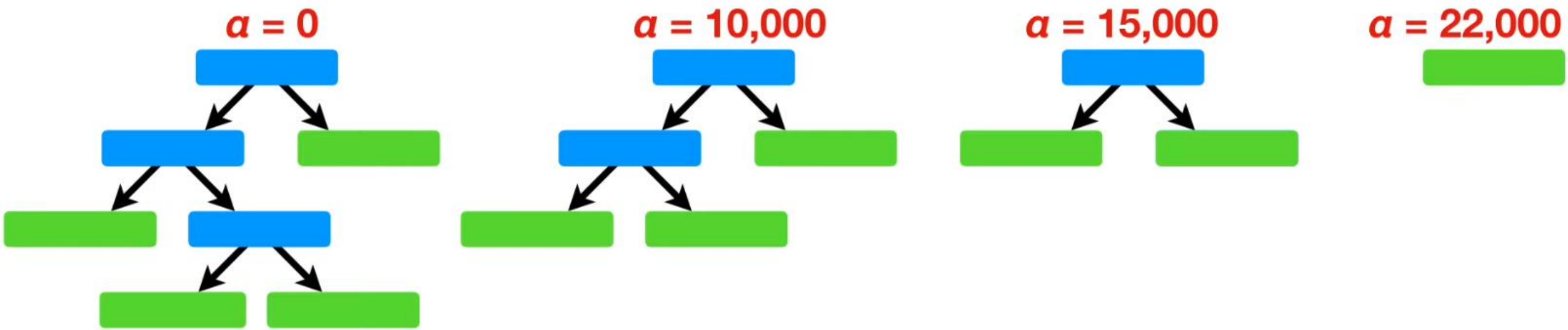
5th round

α = 0 => Tree with 4 leaves => SSR
α = 10000 => Tree with 3 leaves=> SSR
α = 15000 => Tree with 2 leaves=> SSR
α = 22000 => Tree with 1 leaf=> SSR

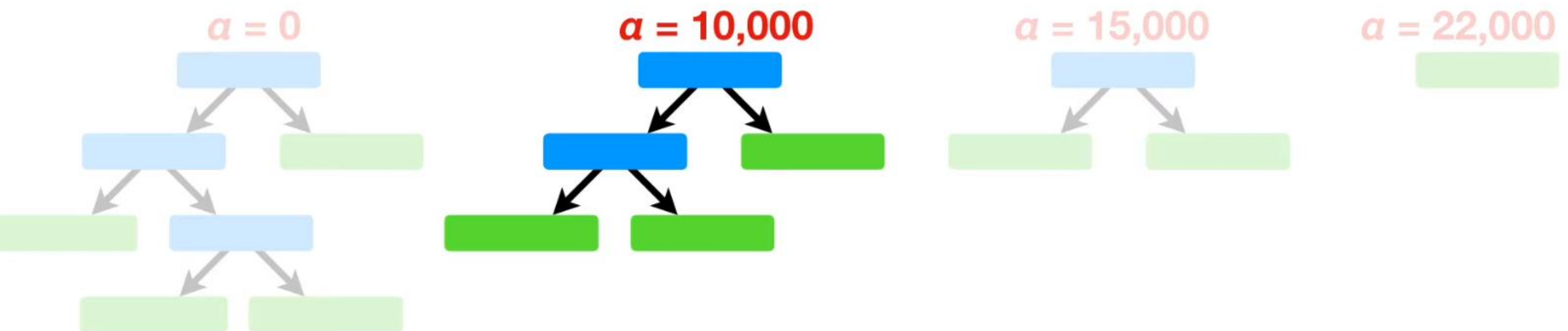
$$\hat{\alpha} = 10000$$

Select the α that, on average, gave us the lowest **Sum of Squared Residuals** with the **Testing data**

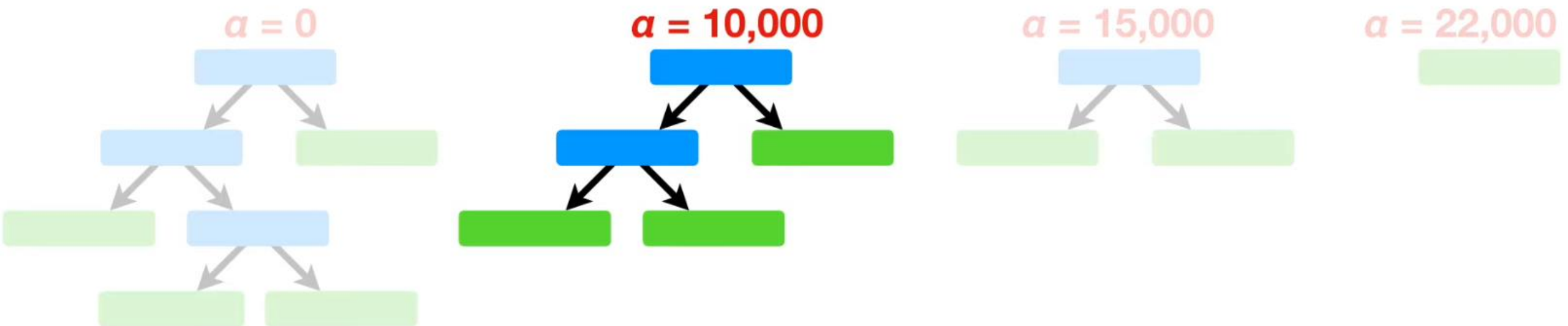
Lastly, we go back to the original trees and sub-trees made from the full data.



…and pick the tree that corresponds to the value for α that we selected (α = 10,000).



This sub-tree will be the final, pruned tree.



2. Tree pruning: summary

1. Use recursive binary splitting to **grow a large tree on the training data**, stopping only when each terminal node has fewer than some minimum number of observations

2. Apply **cost complexity pruning** to the large tree in order to obtain a sequence of best subtrees, as a function of α

3. Use **K-fold cross-validation** to choose α. For each $k = 1; \ldots ; K$:

    3.1 Repeat Steps 1 and 2 on the $K/(K-1)$th fraction of the training data, excluding the kth fold.

    3.2 Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

Average the results, and pick α to minimize the average error.

Return the subtree from Step 2 that corresponds to the chosen value of α.

COST COMPLEXITY PRUNE FOR CLASSIFICATION

1.10.8. Minimal Cost-Complexity Pruning

Minimal cost-complexity pruning is an algorithm used to prune a tree to avoid over-fitting, described in Chapter 3 of [BRE]. This algorithm is parameterized by $\alpha \geq 0$ known as the complexity parameter. The complexity parameter is used to define the cost-complexity measure, $R_\alpha(T)$ of a given tree T:

$$R_\alpha(T) = R(T) + \alpha|\widetilde{T}|$$

where $|\widetilde{T}|$ is the number of terminal nodes in T and $R(T)$ is traditionally defined as the total misclassification rate of the terminal nodes. Alternatively, scikit-learn uses the total sample weighted impurity of the terminal nodes for $R(T)$. As shown above, the impurity of a node depends on the criterion. Minimal cost-complexity pruning finds the subtree of T that minimizes $R_\alpha(T)$.

A natural alternative to RSS is the *classification error rate*. this is simply the fraction of the training observations in that region that do not belong to the most common class:

$$E = 1 - \max_k(\hat{p}_{mk}).$$

https://scikit-learn.org/stable/modules/tree.html#minimal-cost-complexity-pruning