Classification

Lecture 05

Classification

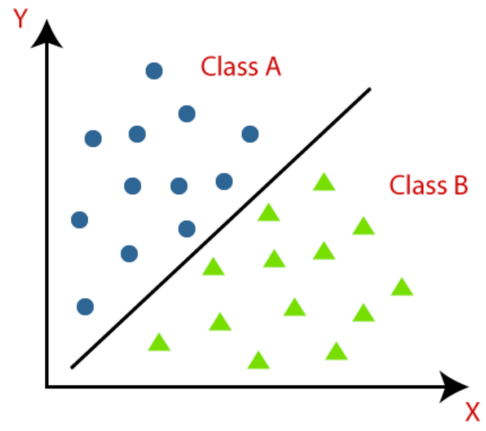
- The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data.
- In Classification, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups.
- Such as, Yes or No, 0 or 1, Spam or Not Spam, cat or dog, etc.
- Classes can be called as targets/labels or categories.

Purpose

- Prediction

  (Classify emails as either "spam" or "not spam.")
- Decision making

  (In medical diagnosis, classification models help in determining whether a patient has a particular disease.)



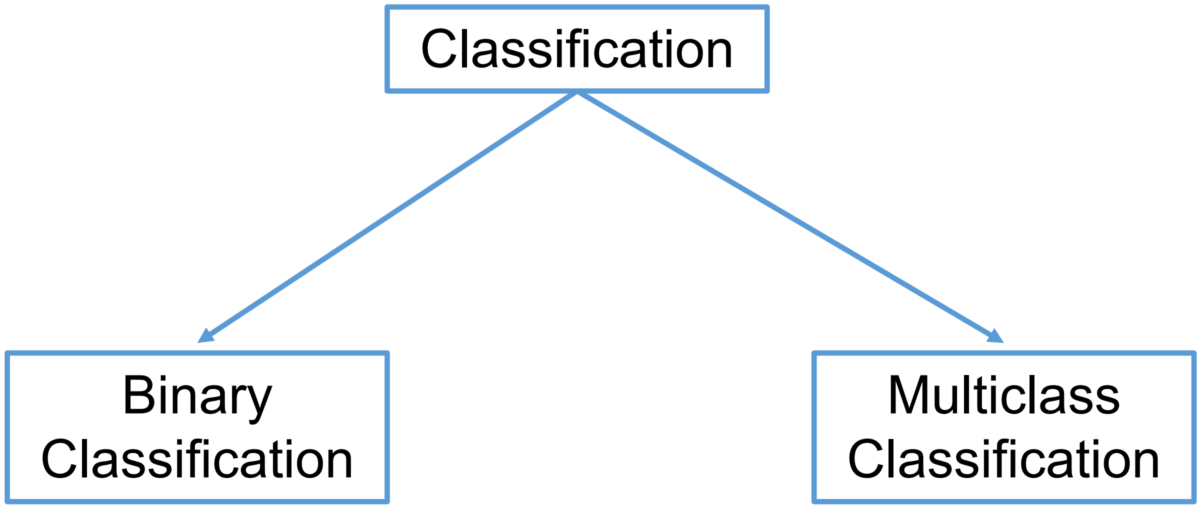
- Data Segmentation

  (customers are classified into different categories based on their purchasing behaviour)
- Risk Management

(in credit scoring, classification models help in determining whether a loan applicant is likely to default)

Types of Classification

(Based on number of output classes)



Binary Classification

Binary classification is a type of classification where the model predicts one of two possible outcomes. The two outcomes are typically labelled as 0 and 1, true and false, or positive and negative.

Example:

1. Spam Detection: Classifying emails as "Spam" or "Not Spam."

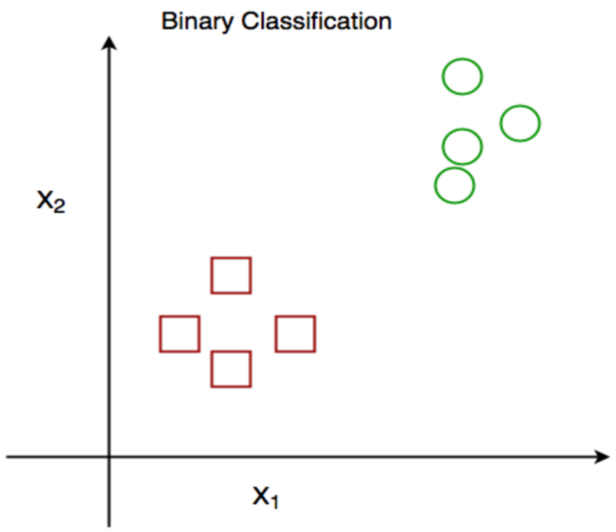
- **Input:** Email content
- **Output:** "Spam" (1) or "Not Spam" (0)

2. Medical Diagnosis: Predicting whether a patient has a certain disease.

- **Input:** Patient data like age, symptoms, and test results
- **Output:** "Disease Present" (1) or "Disease Absent" (0)

❑ **Common Algorithms:**

- Logistic Regression
- Support Vector Machines (SVM)
- Decision Trees
- Naive Bayes
- Random Forests



Multiclass Classification

Multiclass classification is a type of classification where the model predicts one of three or more possible outcomes. Unlike binary classification, the output can be any one of multiple classes.

Example:

1. Image Classification: Classifying images into categories like "Cat," "Dog," and "Bird."

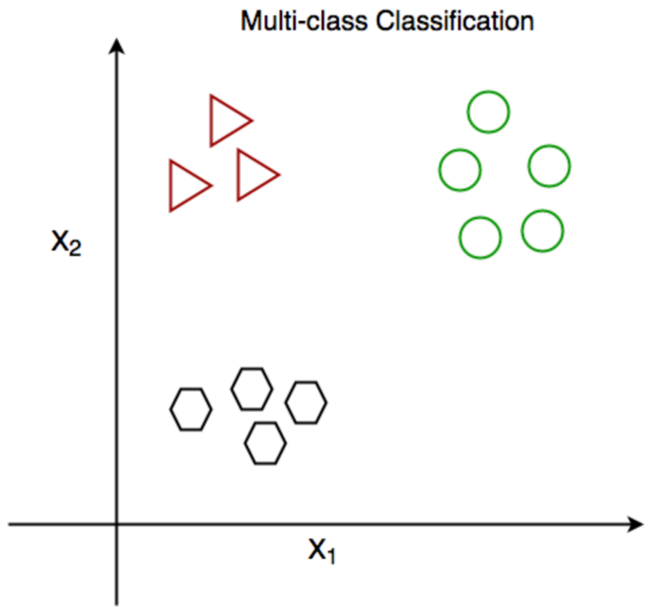
- **Input:** Image pixels
- **Output:** "Cat" (0), "Dog" (1), or "Bird" (2)

2. Sentiment Analysis: Classifying customer reviews as "Positive," "Negative," or "Neutral."

- **Input:** Text of the review
- **Output:** "Positive" (2), "Negative" (0), or "Neutral" (1)

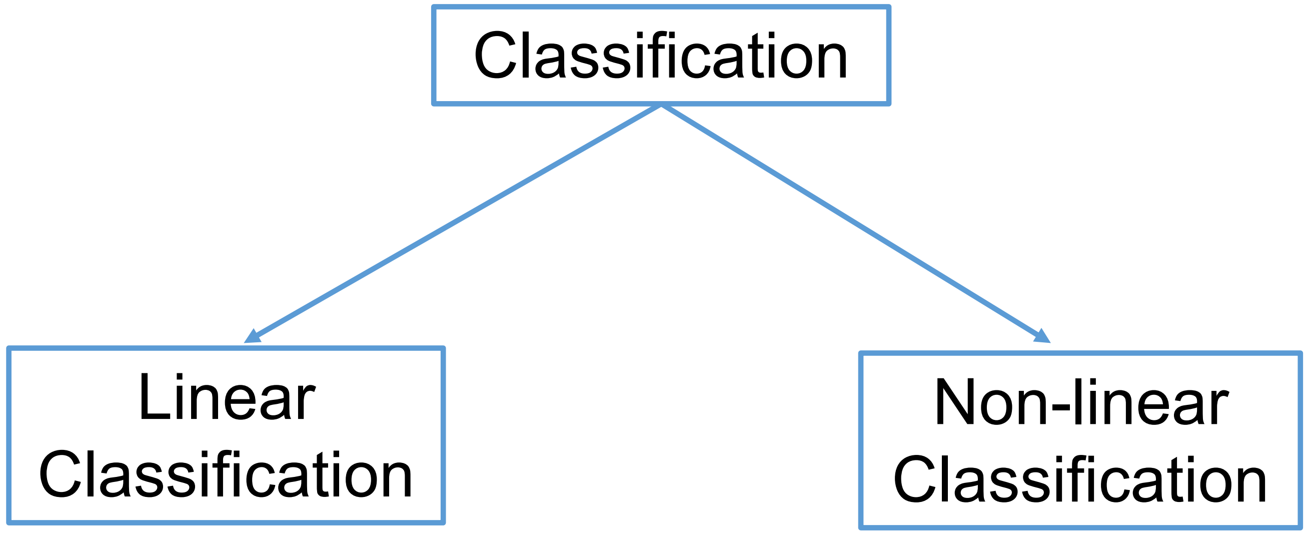
Common Algorithms:

- Multinomial Logistic Regression
- Decision Trees
- Random Forests
- Support Vector Machines (SVM)
- Neural Networks



Types of Classification

(Based on nature of decision boundary)



Linear Classification

Linear classification refers to the process of classifying data points using a linear decision boundary. In a linear classifier, the decision boundary is a straight line (in 2D) or a hyperplane (in higher dimensions) that separates different classes.

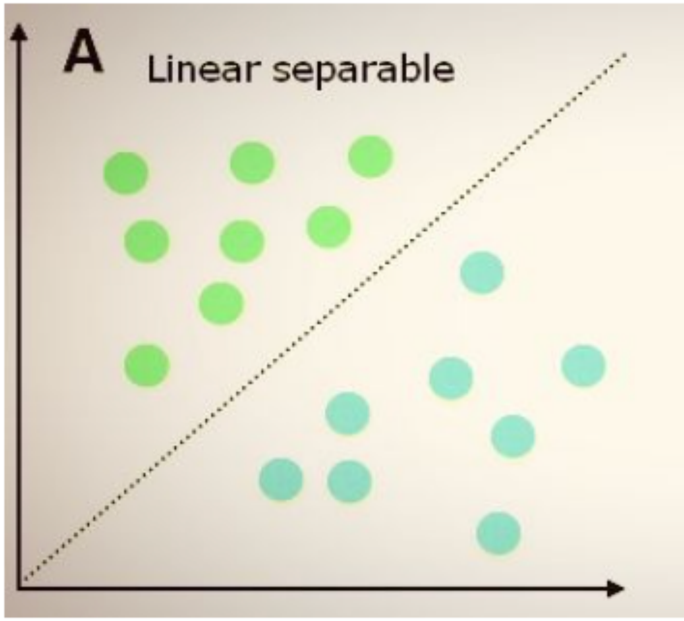
Example:

Email Spam Detection:

Input: Features like the frequency of certain words in the email.

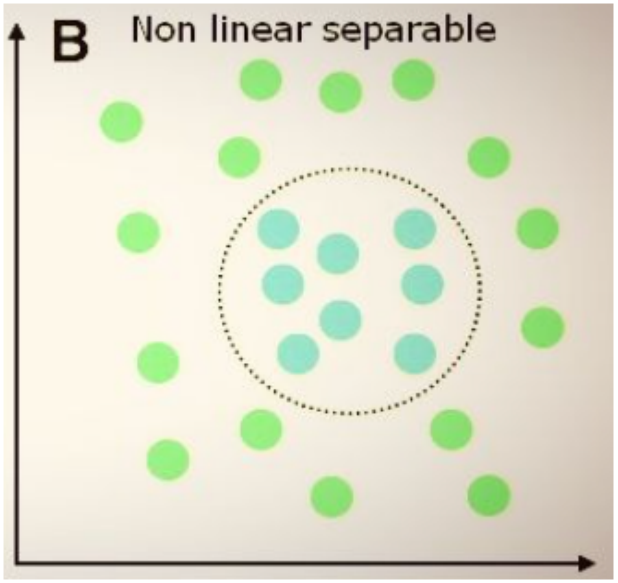
Output: "Spam" or "Not Spam."

Explanation: A linear classifier might separate emails based on a threshold of word frequency. For example, if a particular word appears more than 5 times, the email is classified as spam.



Non-linear Classification

Non-linear classification involves classifying data points using a non-linear decision boundary. The decision boundary can be curved, allowing for the separation of more complex data distributions.

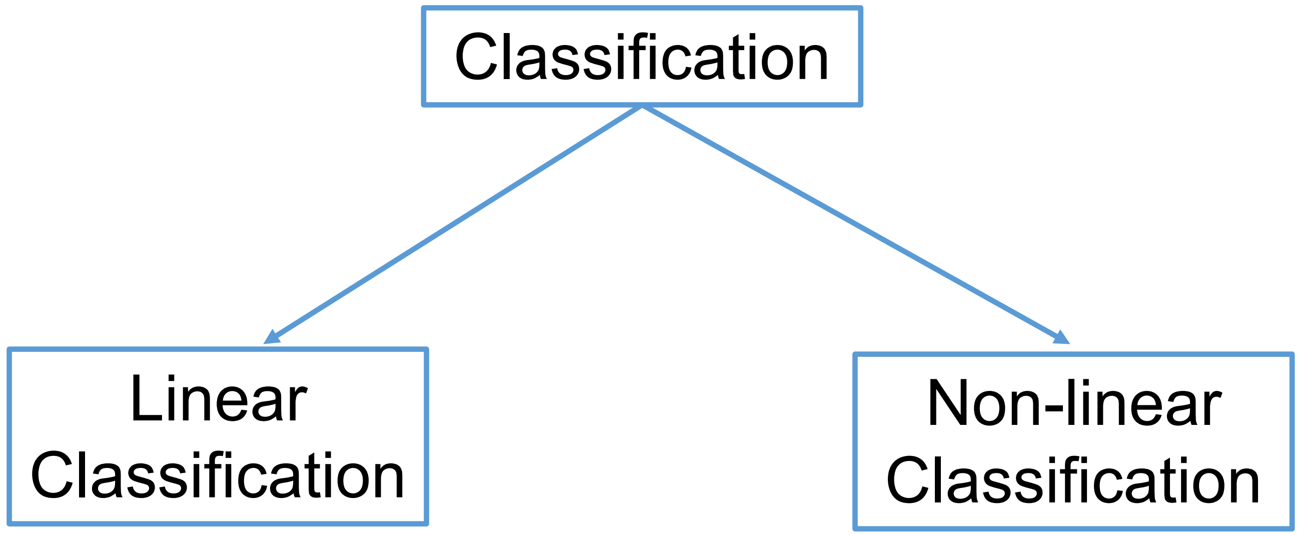


Example:

Image Recognition (e.g., Handwritten Digit Classification):

- **Input:** Pixel values of images.
- **Output:** Digits 0-9.
- **Explanation:** The relationship between pixel intensities and the digit label is complex and often non-linear. Non-linear classifiers can capture these complexities better than linear ones.

Algorithms to follow in Classification



- Logistic Regression
- Support Vector Machines

- K-Nearest Neighbours
- Kernel SVM
- Naïve Bayes
- Decision Tree Classification
- Random Forest Classification

Decision Tree Classification

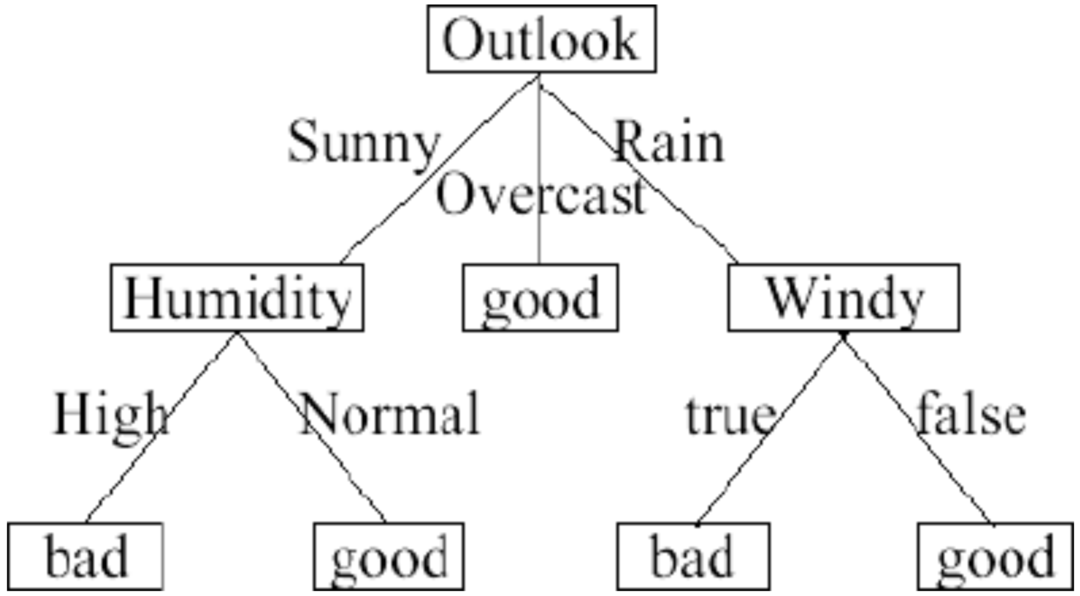
A decision tree is a type of supervised machine learning algorithm used for classification tasks.
It works by splitting the dataset into subsets based on the most significant feature at each node, creating a tree-like structure where each internal node represents a "decision" based on a feature, and each leaf node represents the outcome or label.

Root Node: The topmost node in the tree, representing the entire dataset. It splits into two or more homogeneous sets.

Decision Nodes: Intermediate nodes that split further based on different features.

Leaf Nodes: Terminal nodes that represent the final classification or decision.

Branches: The links between nodes, representing the outcome of a decision rule.



Decision Tree

Now, lets see how to make or create a decision tree with this data.

Account age > 1	Violations > 0	Balance > 50000	Preferred
Yes	Yes	Yes	No
No	Yes	No	No
Yes	No	No	Yes
No	Yes	No	No
Yes	No	Yes	No
No	Yes	No	Yes
Yes	Yes	No	Yes
No	No	No	No

Gini Impurity

To calculate the Gini Impurity for preferred customer, we start calculating the Gini Impurity for the individual Leaves

$$G = 1 - \sum_{i}^{c} \left(p_i\right)^2$$

Gini Impurity (G) for the Leaf on the left (for "account age")

Gini Impurity for a Leaf = 1 - (the probability of Yes)^2 – (the probability of No)^2

$$G = 1 - \left(\left(2/4\right)^2 + \left(2/5\right)^2 \right) = 0.5$$

Account age > 1	Violations > 0	Balance > 50000	Preferred
Yes	Yes	Yes	No
No	Yes	No	No
Yes	No	No	Yes
No	Yes	No	No
Yes	No	Yes	No
No	Yes	No	Yes
Yes	Yes	No	Yes
No	No	No	No

Gini Impurity (G) for the Leaf on the Right (for "account age")

Gini Impurity for a Leaf = 1 - (the probability of Yes)^2 – (the probability of No)^2

$$G = 1 - \left(\left(1/4\right)^2 + \left(3/4\right)^2 \right) = 0.375$$

Account age > 1	Violations > 0	Balance > 50000	Preferred
Yes	Yes	Yes	No
No	Yes	No	No
Yes	No	No	Yes
No	Yes	No	No
Yes	No	Yes	No
No	Yes	No	Yes
Yes	Yes	No	Yes
No	No	No	No

The Total Gini Impurity (G) is the Weighted Average of the Leaf Impurities

Total Gini Impurity = weighted average of Gini Impurities of the Leaves

$$G = \left(4/8\right) \times 0.5 + \left(4/8\right) \times 0.375 = 0.4375$$

Account age > 1	Violations > 0	Balance > 50000	Preferred
Yes	Yes	Yes	No
No	Yes	No	No
Yes	No	No	Yes
No	Yes	No	No
Yes	No	Yes	No
No	Yes	No	Yes
Yes	Yes	No	Yes
No	No	No	No

Gini Impurity (G) for the Leaf on the left (for "Violations")

Gini Impurity for a Leaf = 1 -(the probability of Yes)^2 – (the probability of No)^2

$$G = 1 - \left(\left(3/5\right)^2 + \left(2/5\right)^2 \right) = 0.48$$

Account age > 1	Violations > 0	Balance > 50000	Preferred
Yes	Yes	Yes	No
No	Yes	No	No
Yes	No	No	Yes
No	Yes	No	No
Yes	No	Yes	No
No	Yes	No	Yes
Yes	Yes	No	Yes
No	No	No	No

Gini Impurity (G) for the Leaf on the Right (for “Violations”)

Gini Impurity for a Leaf = 1 -(the probability of Yes)^2 – (the probability of No)^2

$$G = 1 - \left(\left(1/3\right)^2 + \left(2/3\right)^2 \right) = 0.44$$

Account age > 1	Violations > 0	Balance > 50000	Preferred
Yes	Yes	Yes	No
No	Yes	No	No
Yes	$\boxed{\text{No}}$	No	$\boxed{\text{Yes}}$
No	Yes	No	No
Yes	$\boxed{\text{No}}$	Yes	$\boxed{\text{No}}$
No	Yes	No	Yes
Yes	Yes	No	Yes
No	$\boxed{\text{No}}$	No	$\boxed{\text{No}}$

The Total Gini Impurity (G) is the Weighted Average of the Leaf Impurities

(for "Violations")

Total Gini Impurity = weighted average of Gini Impurities of the Leaves

$$G = \left(5/8\right) \times 0.48 + \left(3/8\right) \times 0.44 = 0.465$$

Account age > 1	Violations > 0	Balance > 50000	Preferred
Yes	Yes	Yes	No
No	Yes	No	No
Yes	No	No	Yes
No	Yes	No	No
Yes	No	Yes	No
No	Yes	No	Yes
Yes	Yes	No	Yes
No	No	No	No

Gini Impurity (G) for the Leaf on the left (for "Balance")

Gini Impurity for a Leaf = 1 -(the probability of Yes)^2 – (the probability of No)^2

$$G = 1 - \left(\left(0/2\right)^2 + \left(2/2\right)^2 \right) = 0$$

Account age > 1	Violations > 0	Balance > 50000	Preferred
Yes	Yes	$\boxed{\text{Yes}}$	$\boxed{\text{No}}$
No	Yes	No	No
Yes	No	No	Yes
No	Yes	No	No
Yes	No	$\boxed{\text{Yes}}$	$\boxed{\text{No}}$
No	Yes	No	Yes
Yes	Yes	No	Yes
No	No	No	No

Gini Impurity (G) for the Leaf on the Right (for “Balance”)

Gini Impurity for a Leaf = 1 -(the probability of Yes)^2 – (the probability of No)^2

$$G = 1 - \left(\left(3/6\right)^2 + \left(3/6\right)^2 \right) = 0.5$$

Account age > 1	Violations > 0	Balance > 50000	Preferred
Yes	Yes	Yes	No
No	Yes	No	No
Yes	No	No	Yes
No	Yes	No	No
Yes	No	Yes	No
No	Yes	No	Yes
Yes	Yes	No	Yes
No	No	No	No

The Total Gini Impurity (G) is the Weighted Average of the Leaf Impurities

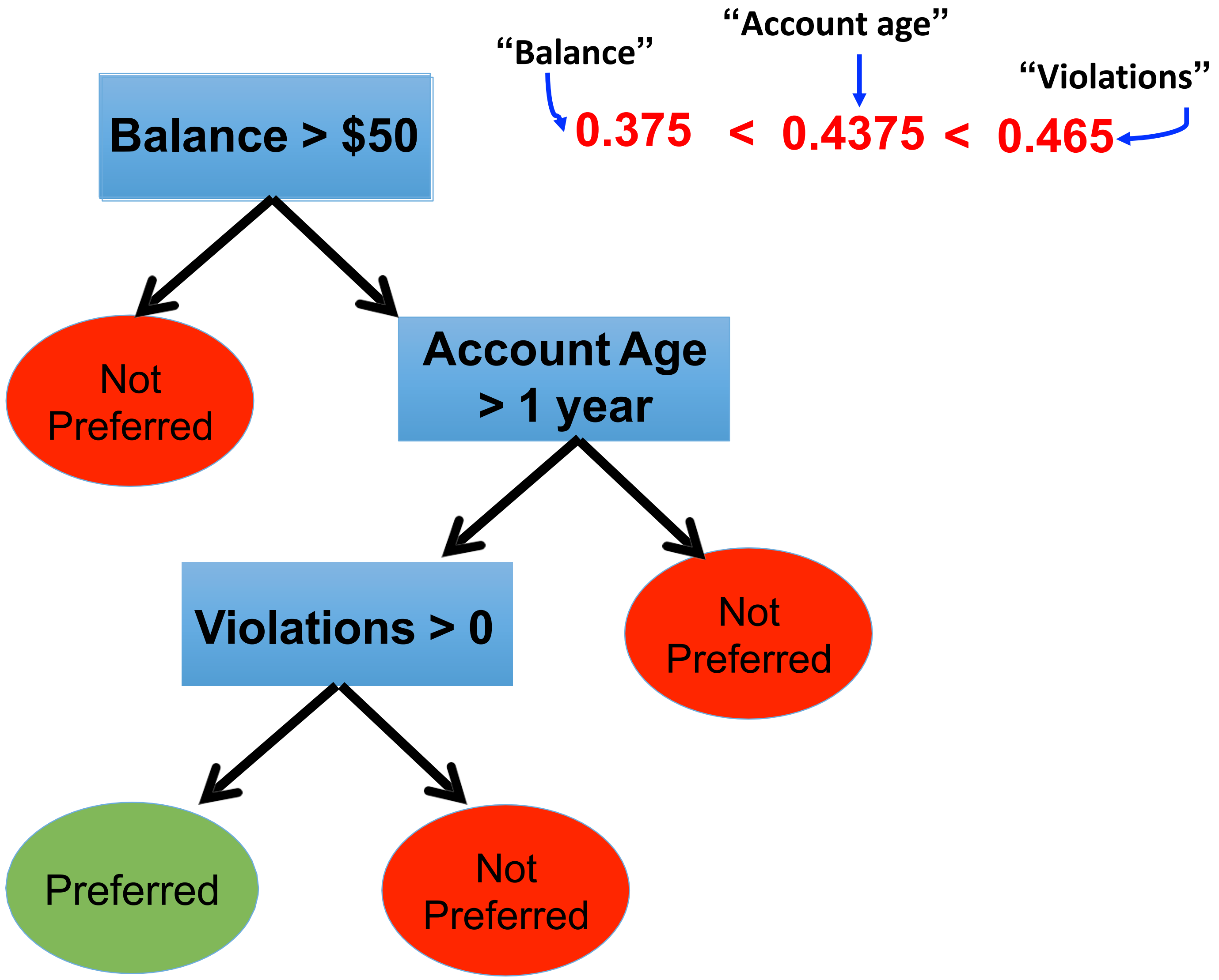
(for “Balance”)

Total Gini Impurity = weighted average of Gini Impurities of the Leaves

$$G = (2/8)\times 0 + (6/8)\times 0.5 = 0.375$$

Account age > 1	Violations > 0	Balance > 50000	Preferred
Yes	Yes	Yes	No
No	Yes	No	No
Yes	No	No	Yes
No	Yes	No	No
Yes	No	Yes	No
No	Yes	No	Yes
Yes	Yes	No	Yes
No	No	No	No

Decision Tree



Numerical

Here, we have provided a dataset of 6 patients with **CANCER** and its symptoms as observed in these patients.

Blurry vision	Weight Loss	Cancer
Yes	No	Yes
Yes	Yes	No
Yes	Yes	No
No	No	Yes
Yes	Yes	No
Yes	No	No

Calculate the total Gini impurity value of **Weight Loss** symptoms for separating the patients with or without **Cancer**.

Numerical

Make the decision tree on basis of the following data.

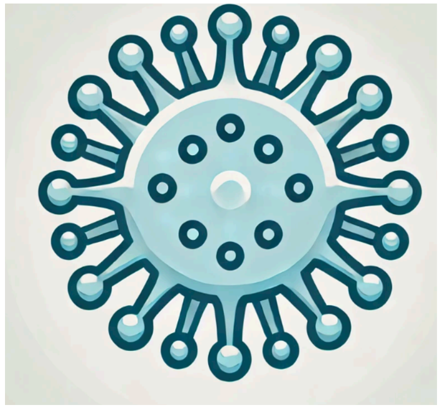
Supplies	Weather	Worked	Shopped
Low	Sunny	Yes	Yes
High	Sunny	Yes	No
Med	Cloudy	Yes	No
Low	Raining	Yes	No
Low	Cloudy	No	Yes
High	Sunny	No	No
High	Raining	No	No
Med	Cloudy	Yes	No
Low	Raining	Yes	No
Low	Raining	No	Yes
Med	Sunny	No	Yes
High	Sunny	Yes	No

Predict Drug Effectiveness

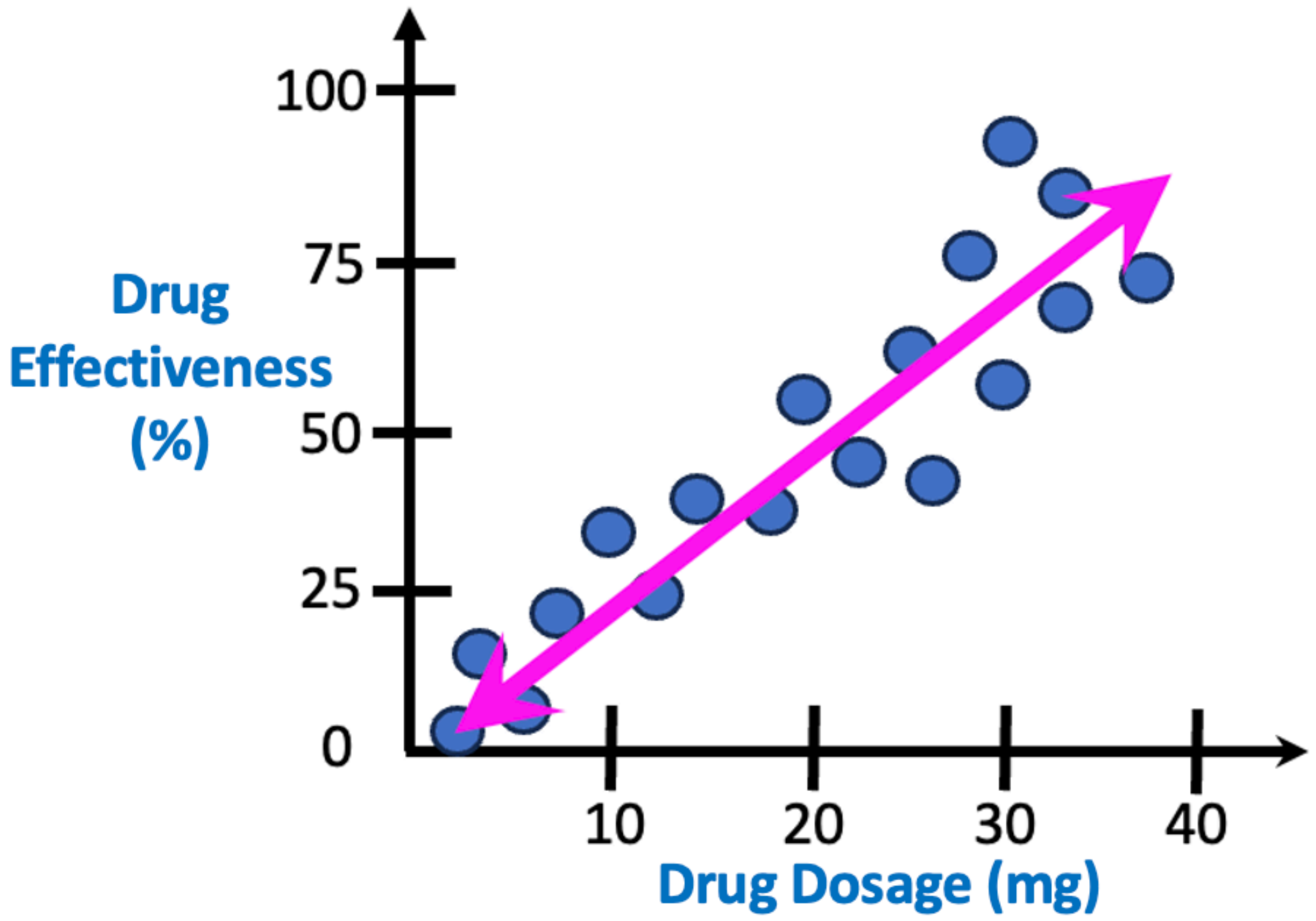
Imagine we develop new drug

To cure the coronavirus

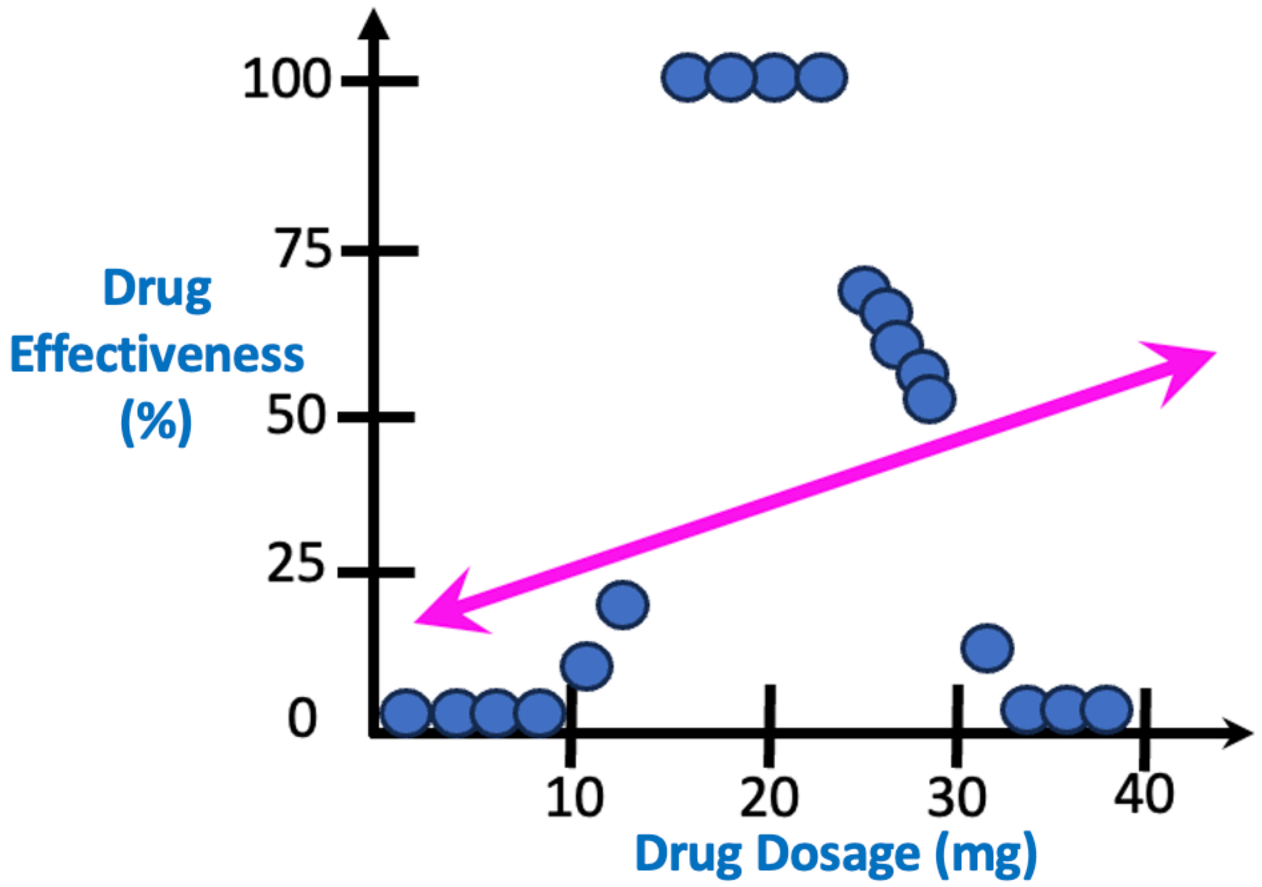
vs



We could use the line to predict that a 27 mg Dose should be 62% Effective

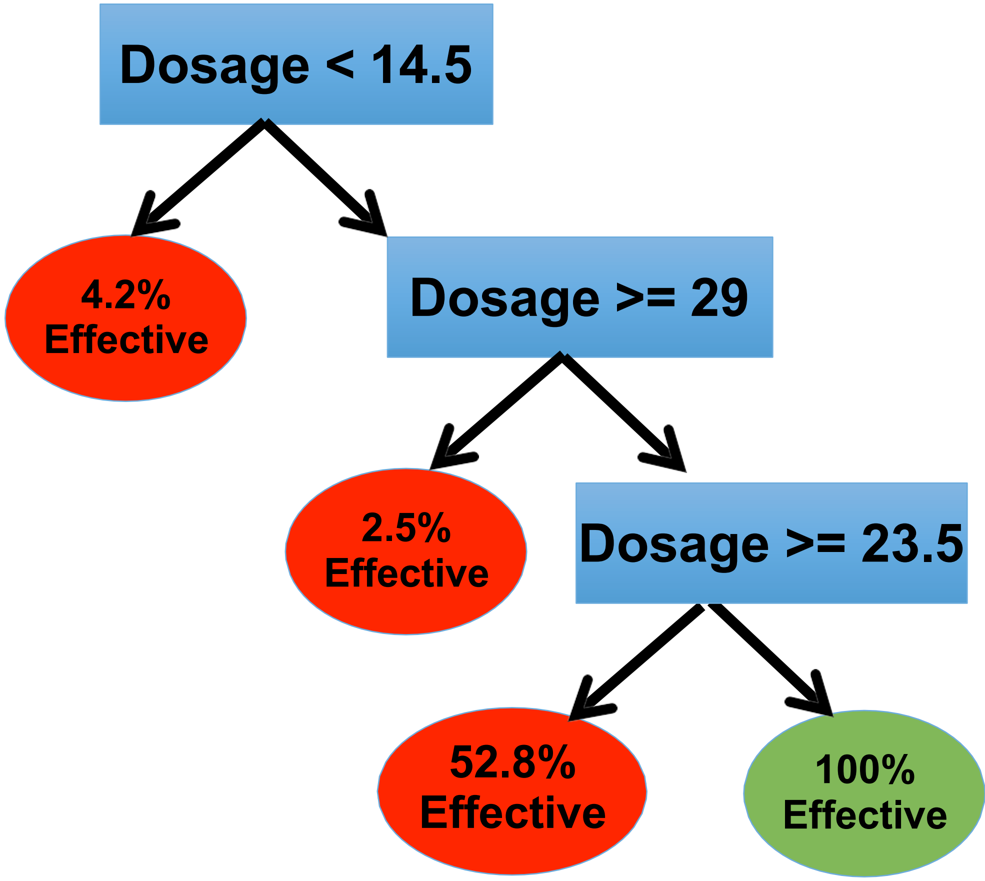


However, What if the data looked like this

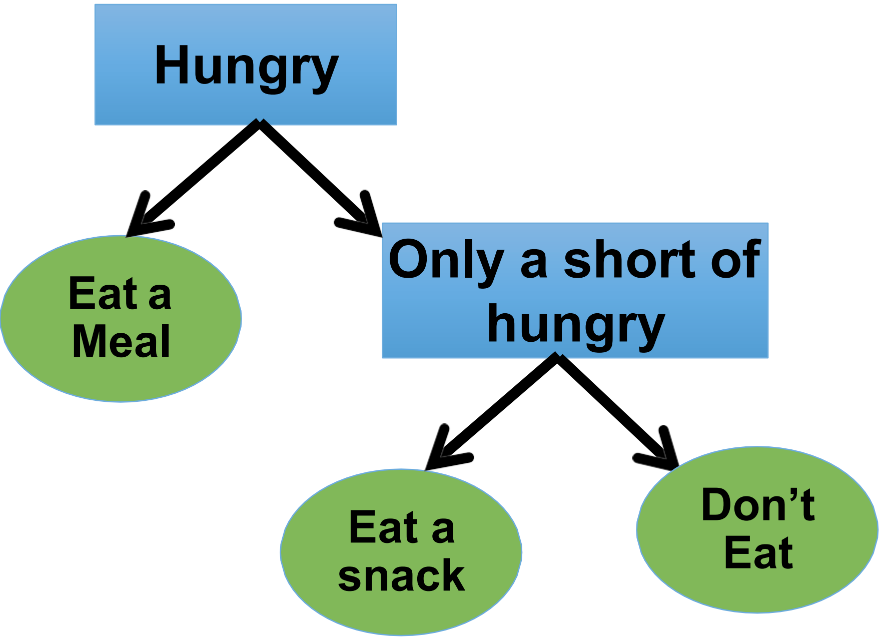


Regression Tree

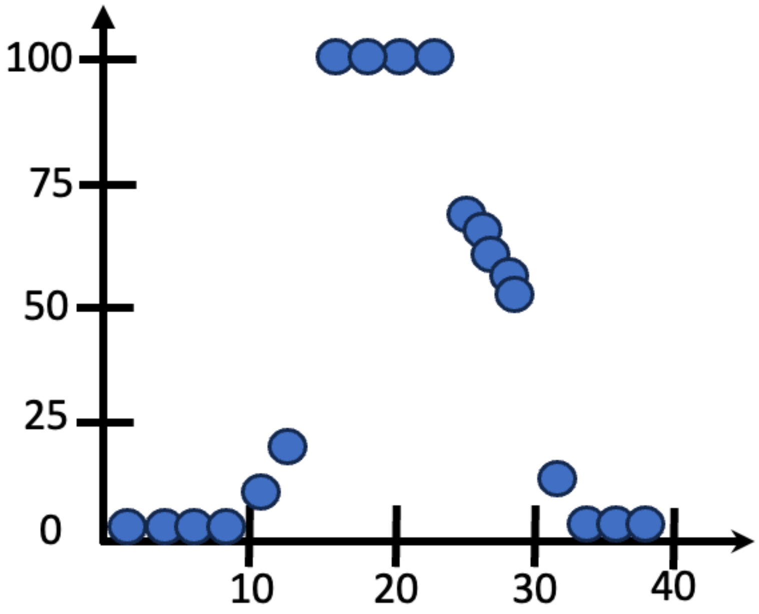
- Regression Trees are a type of decision Tree
- In a Regression Trees, each leaf represents a numeric value

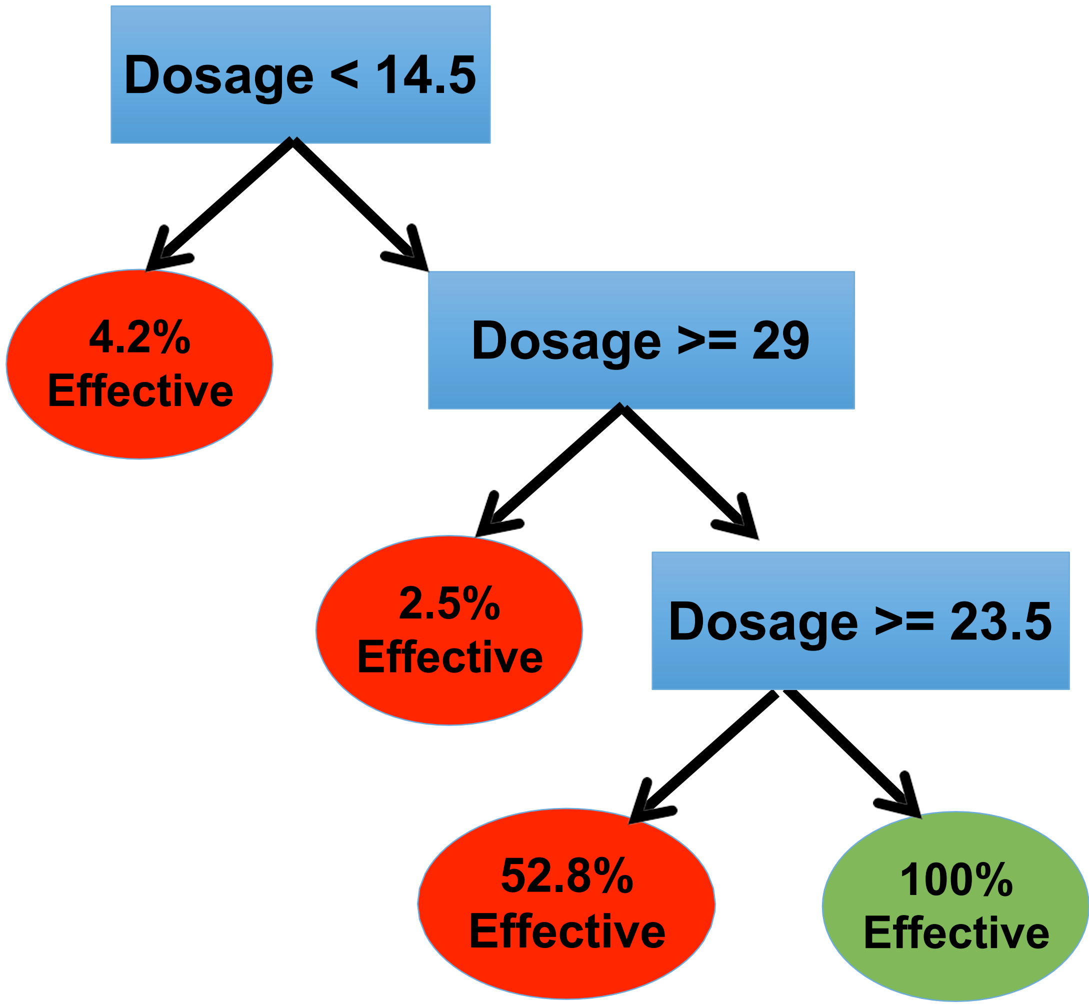


In contrast, Classification Trees have True or False in their leaves



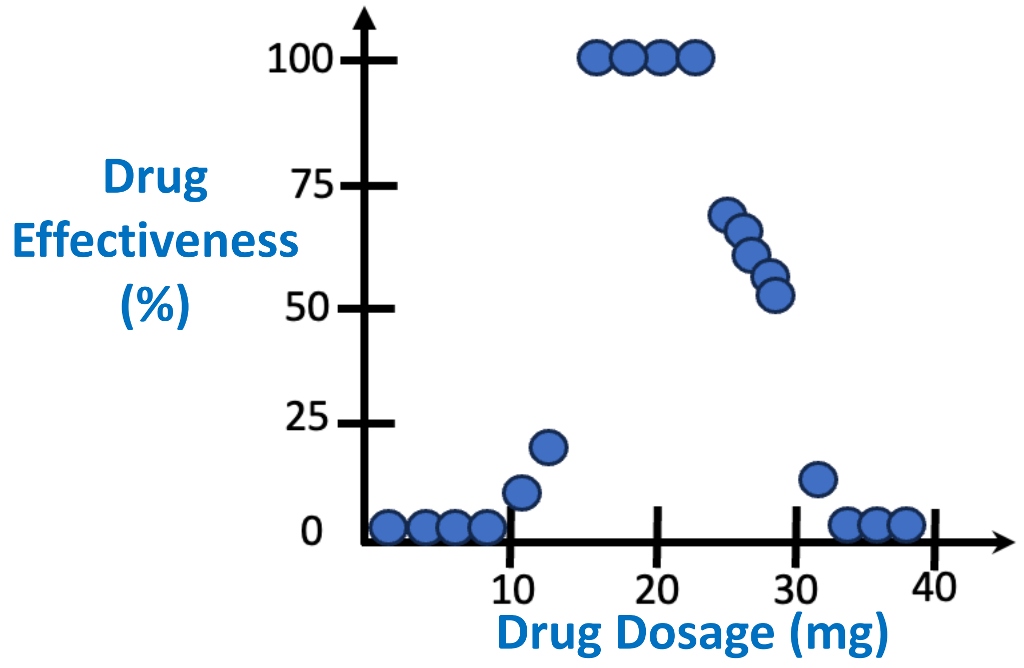
However, we are talking about these Observations in the training dataset





When the data are super simple and we are only using one predictor

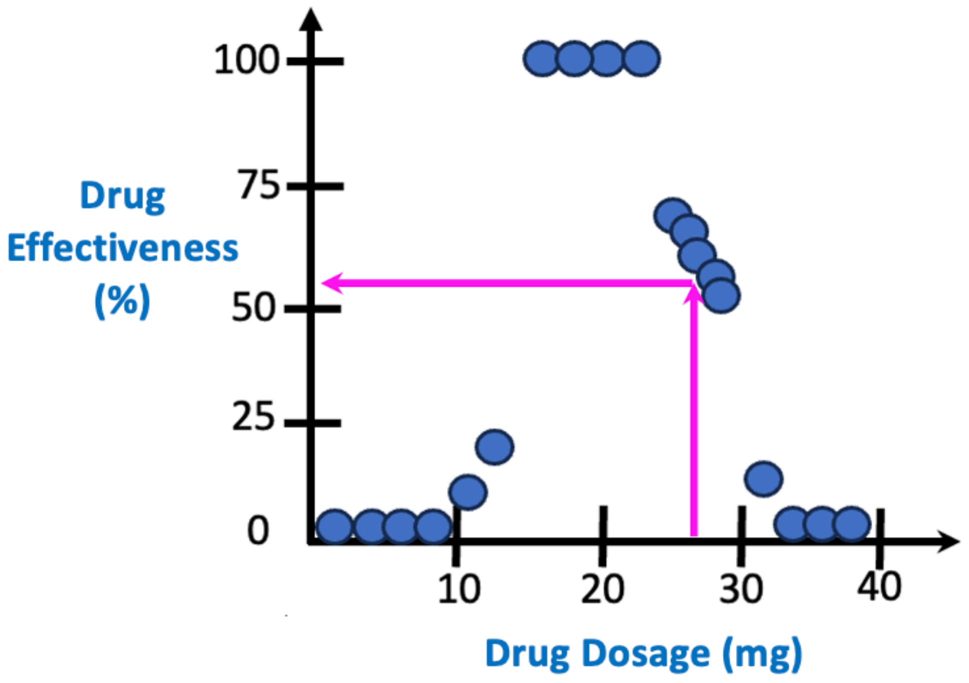
Dosage to predict Drug Effectiveness, making predictions by eye isn't terrible



Dosage	Drug Effect
10	98
20	0
35	100
5	44
Etc.	Etc.

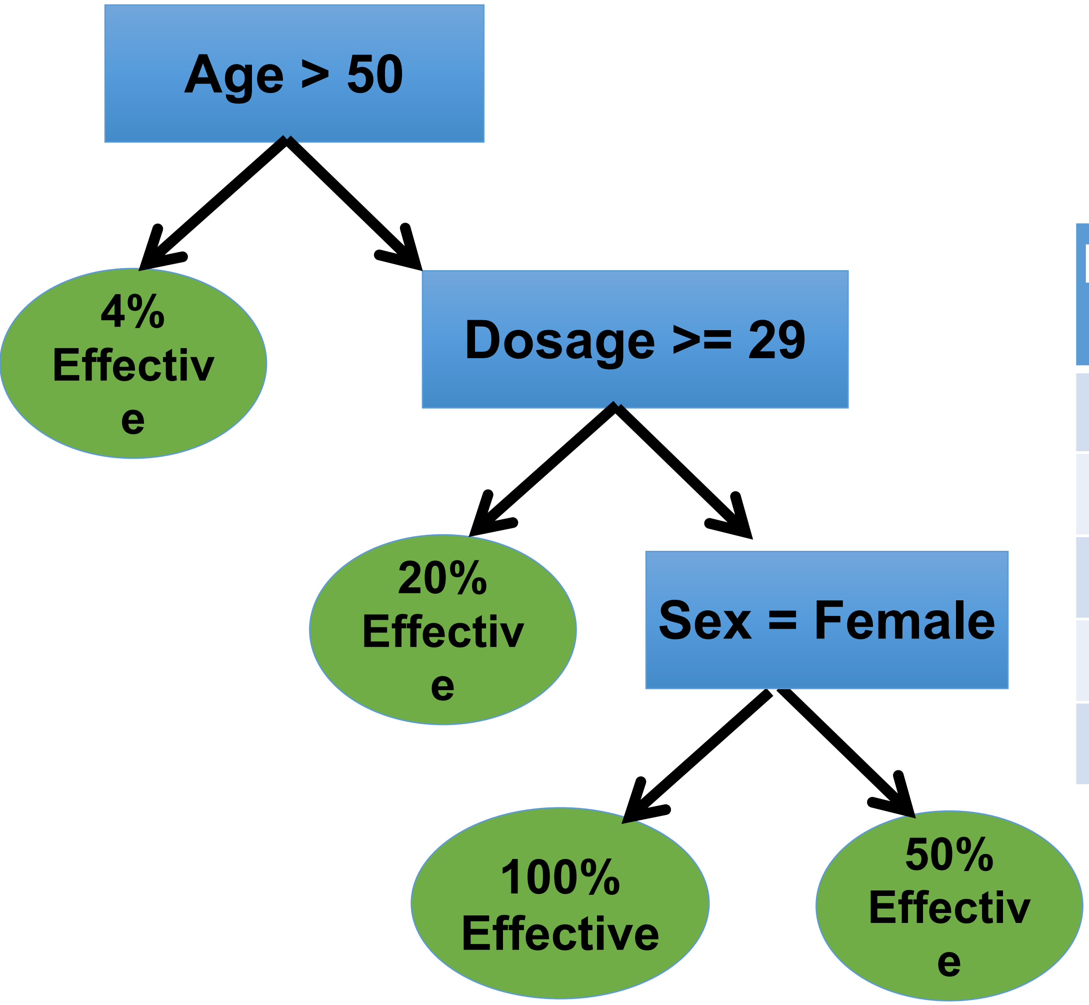
But when we have 3 or more predictors, like Dosage, Age, and Sex to predict Drug Effectiveness

Regression Tree easily accommodates the additional predictors



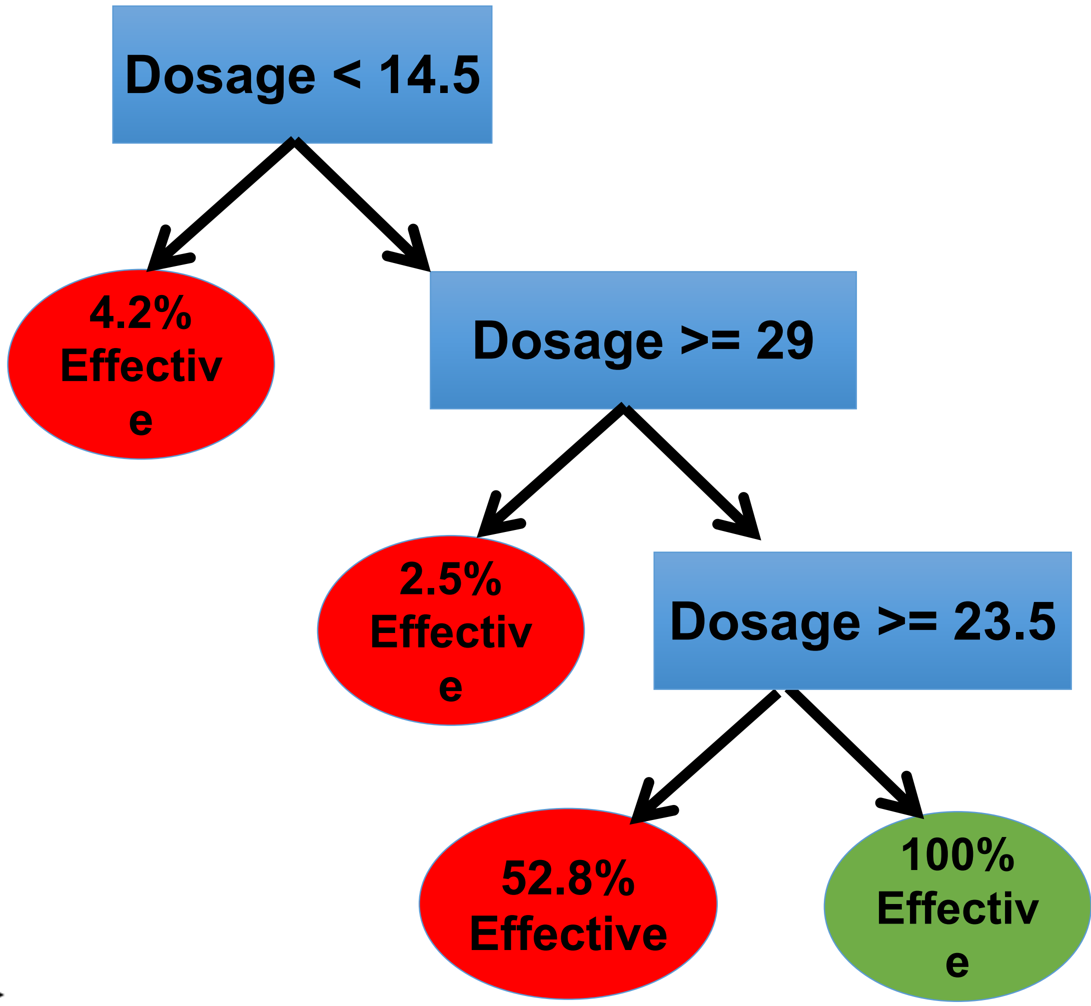
Dosage	Age	Sex	Etc.	Drug Effect
10	25	Female	…	98
20	73	Male	…	0
35	54	Female	…	100
5	12	Male	…	44
Etc.	Etc.	Etc.	Etc.	Etc.

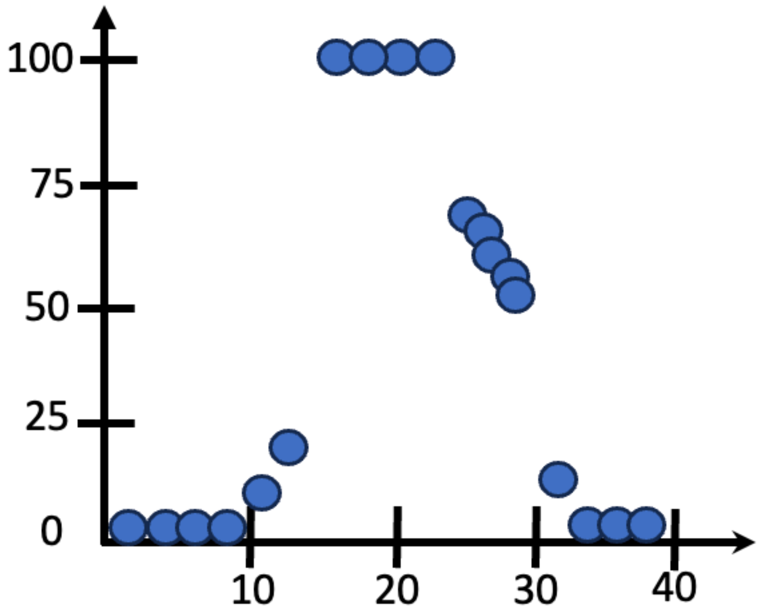
Regression Tree easily accommodates the additional predictors



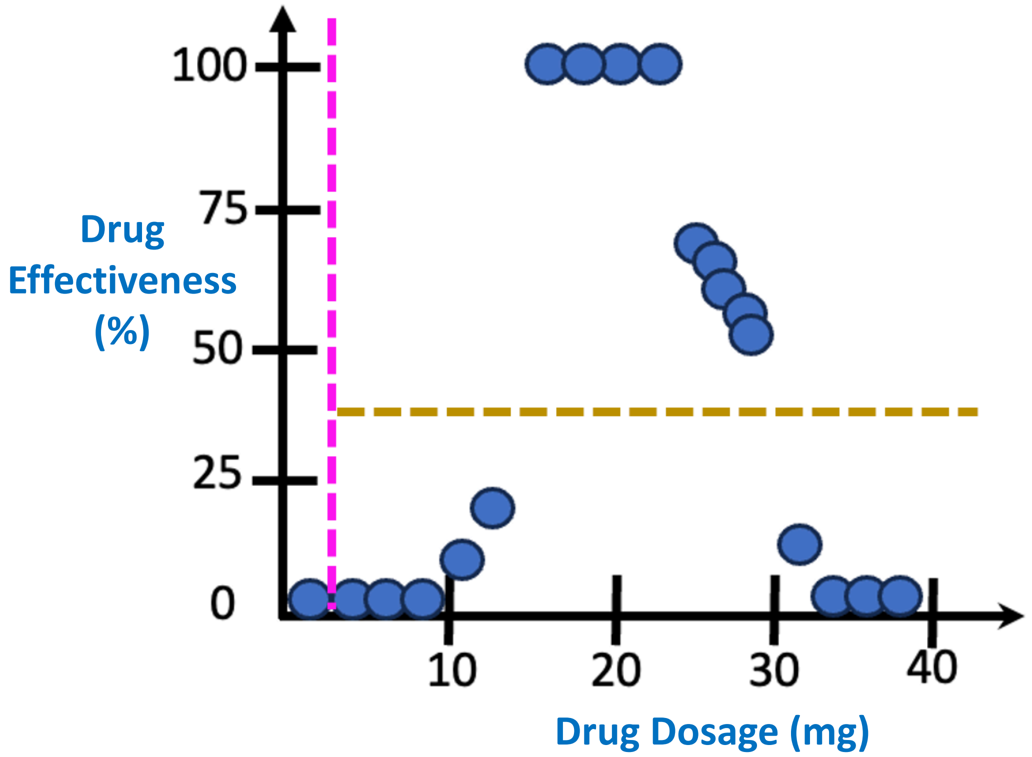
Dosage	Age	Sex	Etc.	Drug Effect
10	25	Female	…	98
20	73	Male	…	0
35	54	Female	…	100
5	12	Male	…	44
Etc.	Etc.	Etc.	Etc.	Etc.

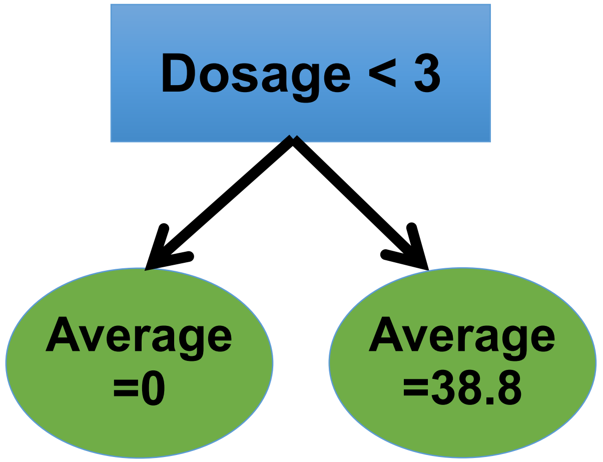
The first thing we do is to figure out why we start by asking if Dosage < 14.5



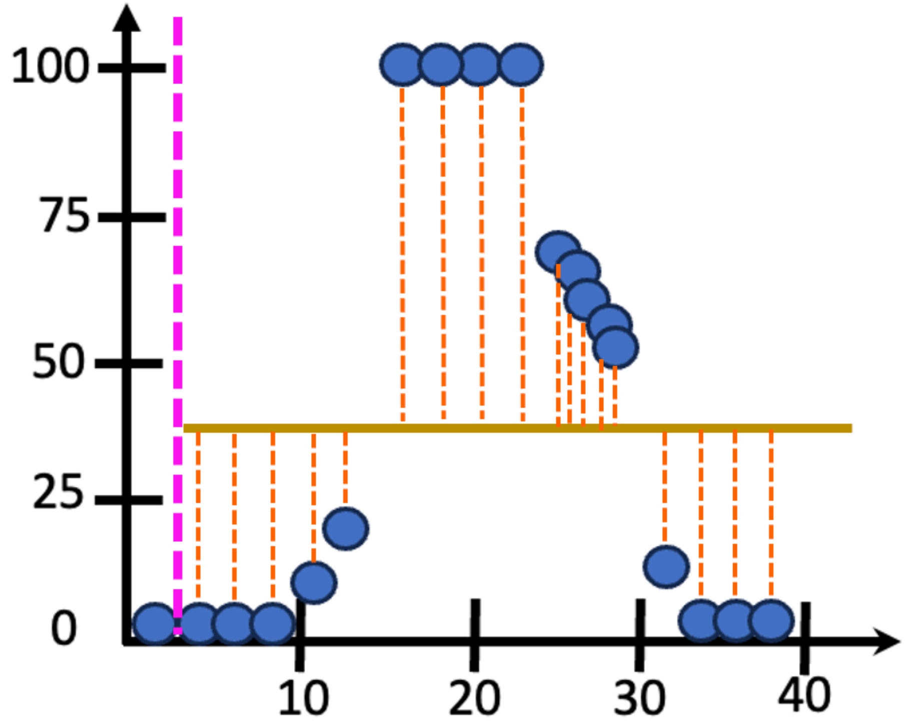


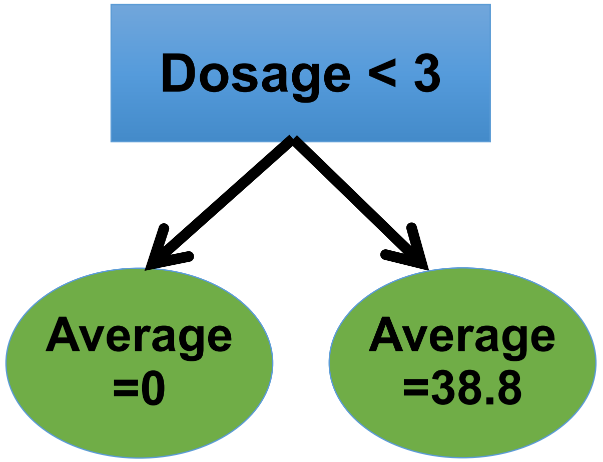
The average Drug Effectiveness for all of the points with Dosage >= 3 is 38.8 (the green line)





For each point in the data, we can draw its residual, the difference between the observed and predicted values



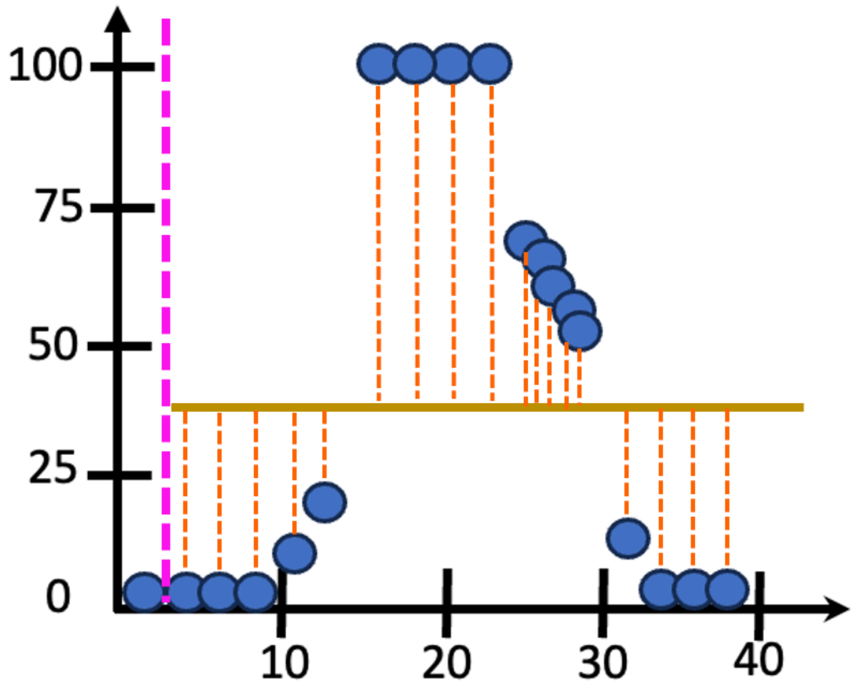


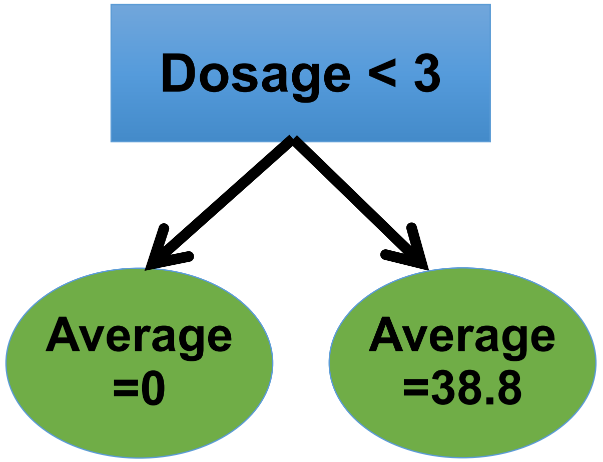
Sum of Squared residual (SSR) for all points=

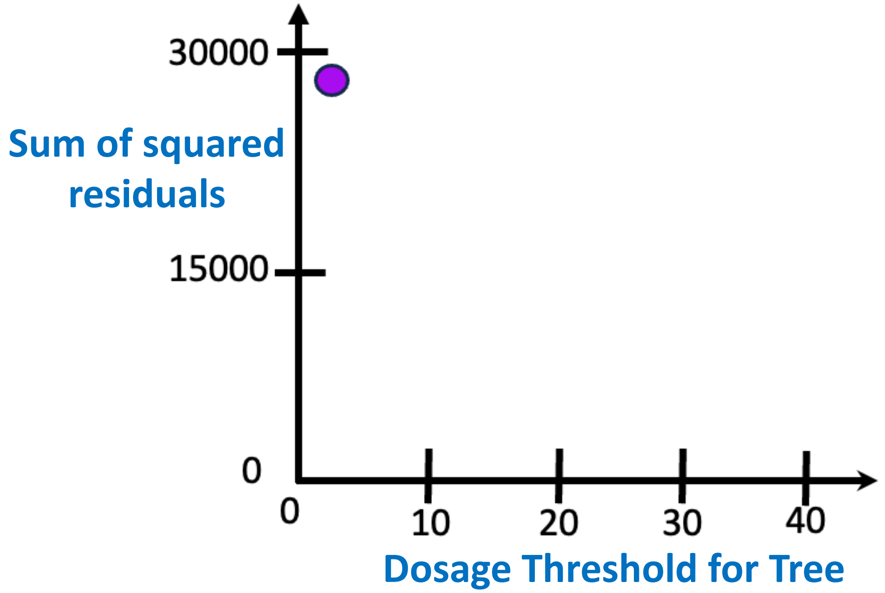
$(0 - 0)^2 + (0 - 38.8)^2 + (0 - 38.8)^2 + (0 - 38.8)^2 + (5 - 38.8)^2 + (20 - 38.8)^2 + (100 - 38.8)^2 + (100 - 38.8)^2 + (100 - 38.8)^2 + (100 - 38.8)^2 + \ldots\ldots (0 - 38.8)^2 +$

$\boxed{= 27{,}468.5}$

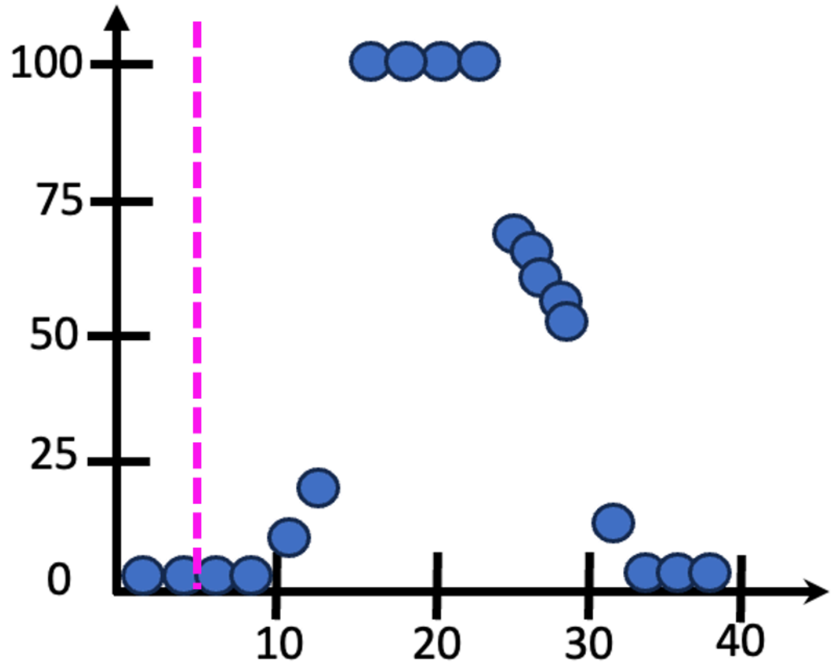
In this case, dosage threshold was 3

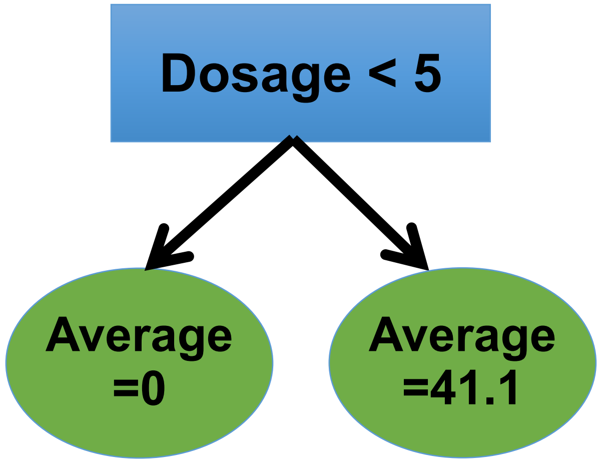


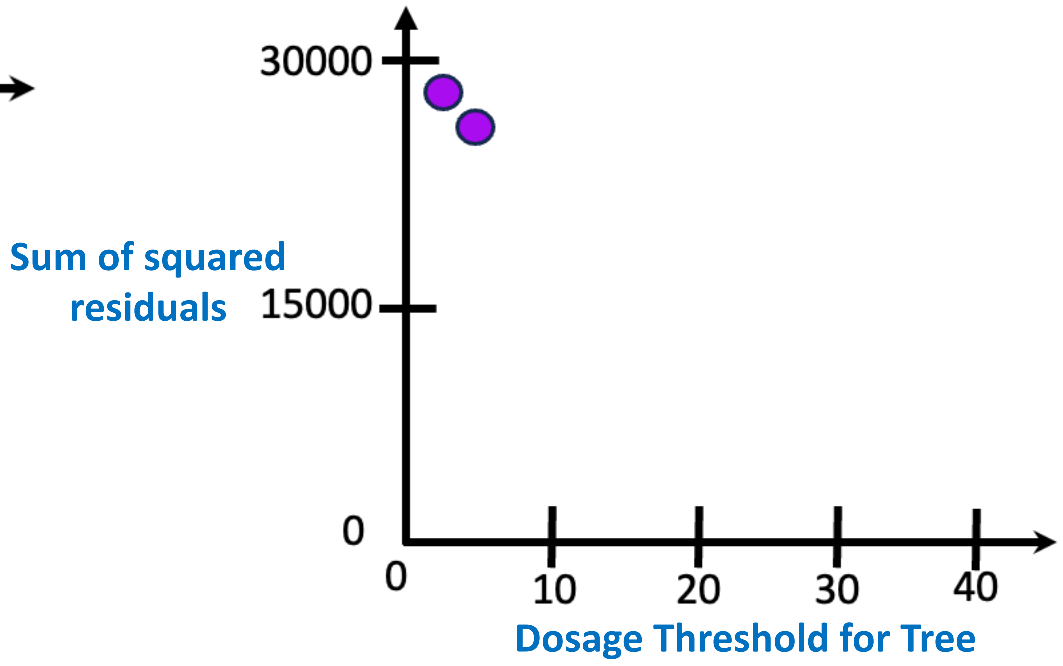




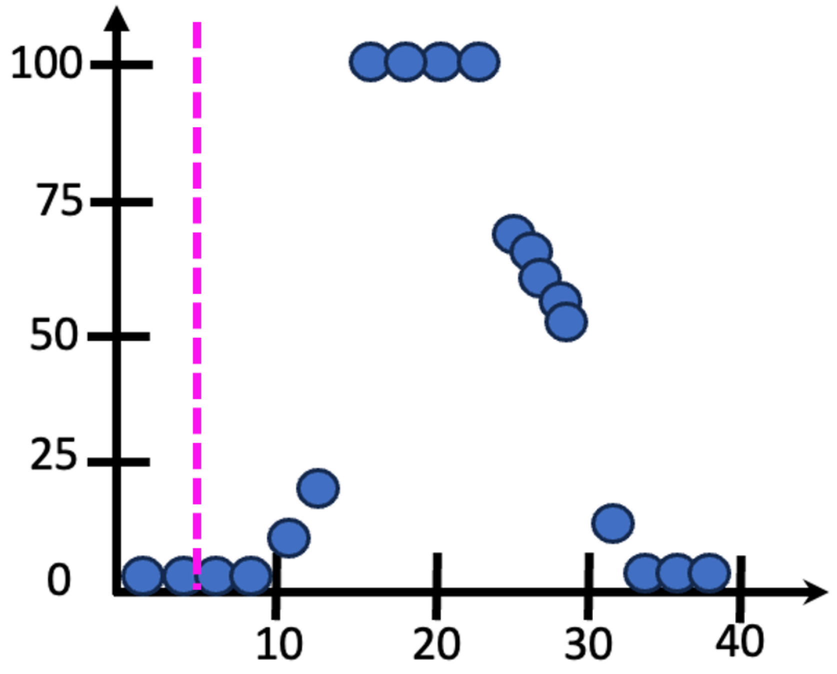
In this case, dosage threshold was 5

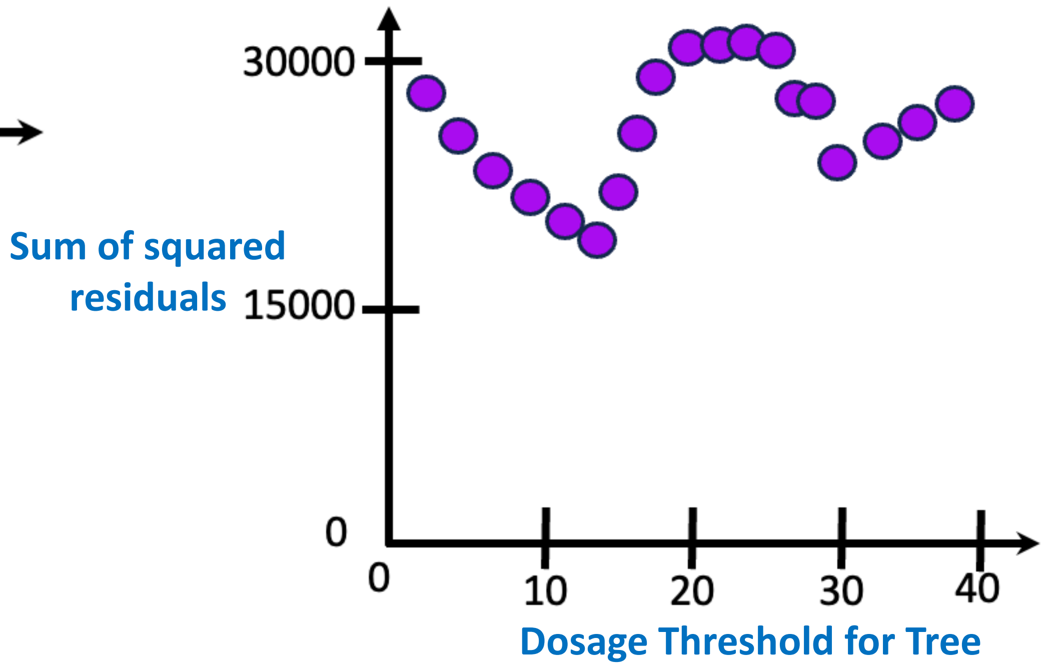






The sum of squared residuals for all of the threshold





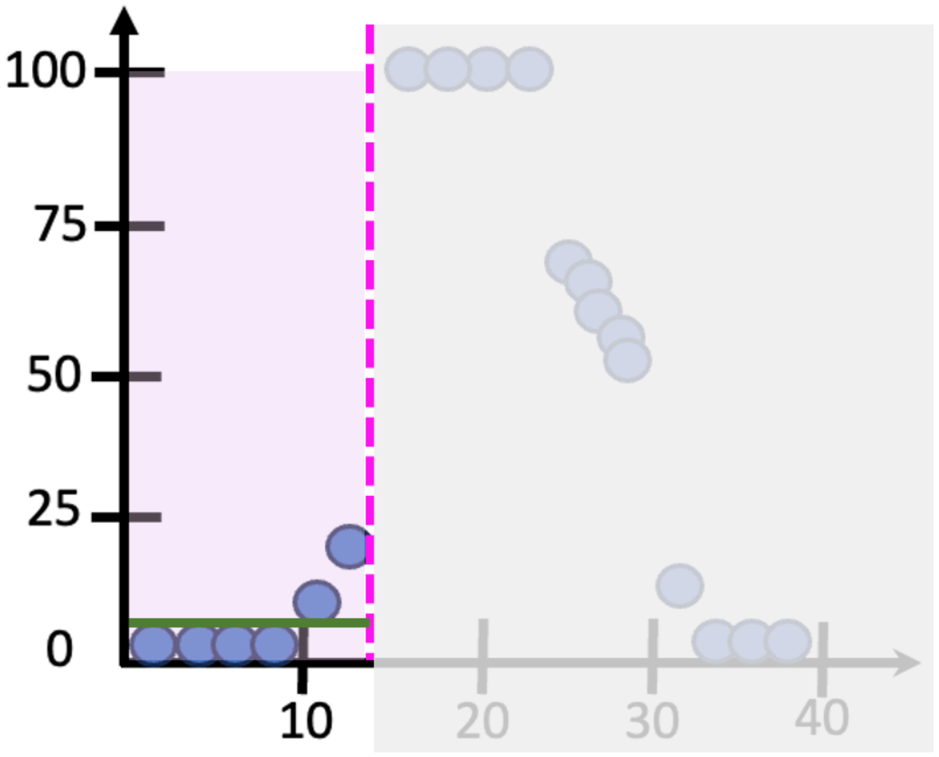
When to split observations

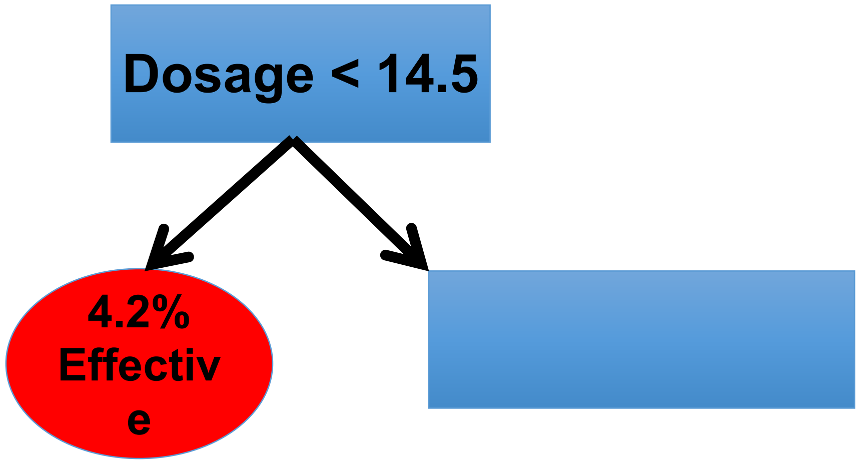
The simplest is to only split observations when there are more than some minimum number

Typically, the minimum number of observations to allow for a split is 20

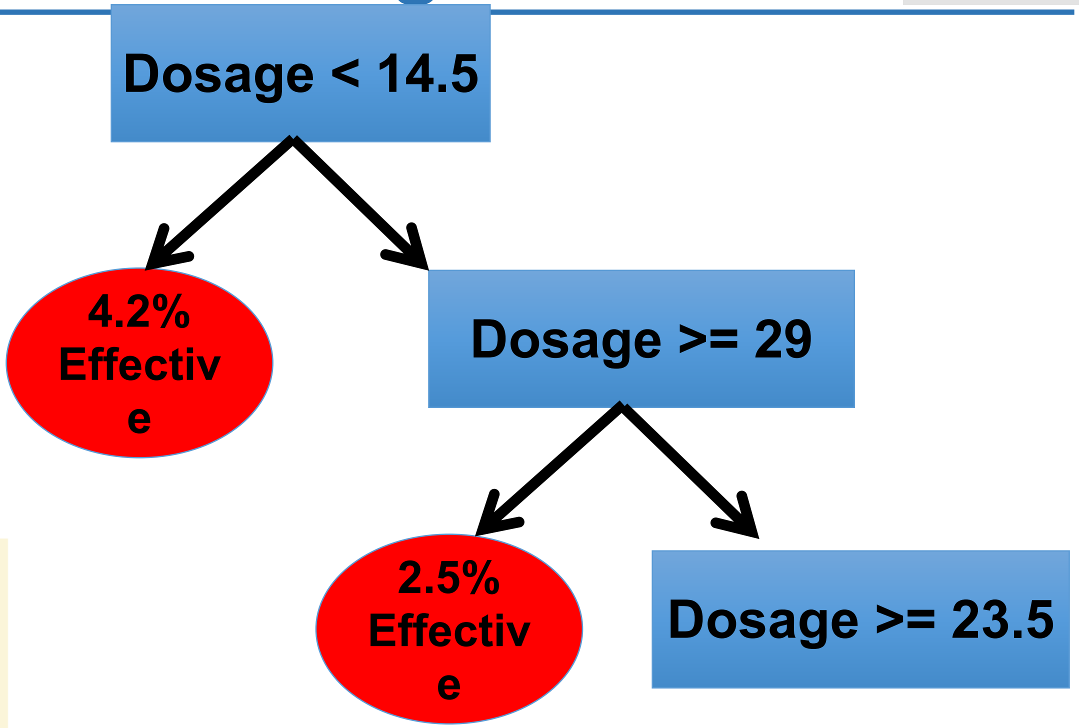
However, since this example doesn't have many observations, we set the minimum to 7

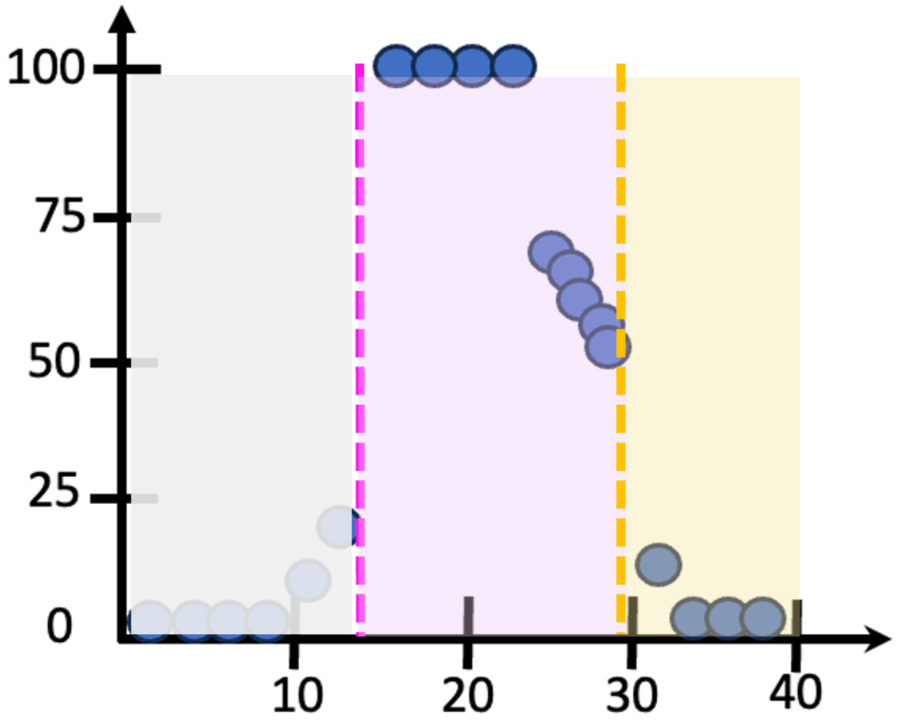
Drug Effectiveness for the 6 observations with Dosage < 14.5, 4.2%





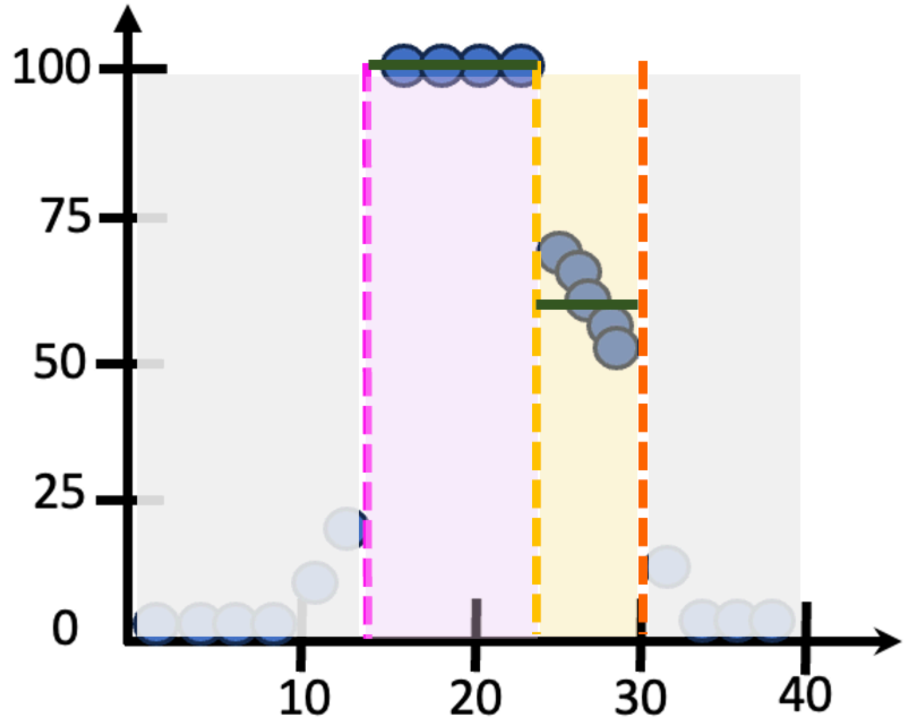
Drug Effectiveness for more than 7 observations on the right side with Dosage >= 14.5

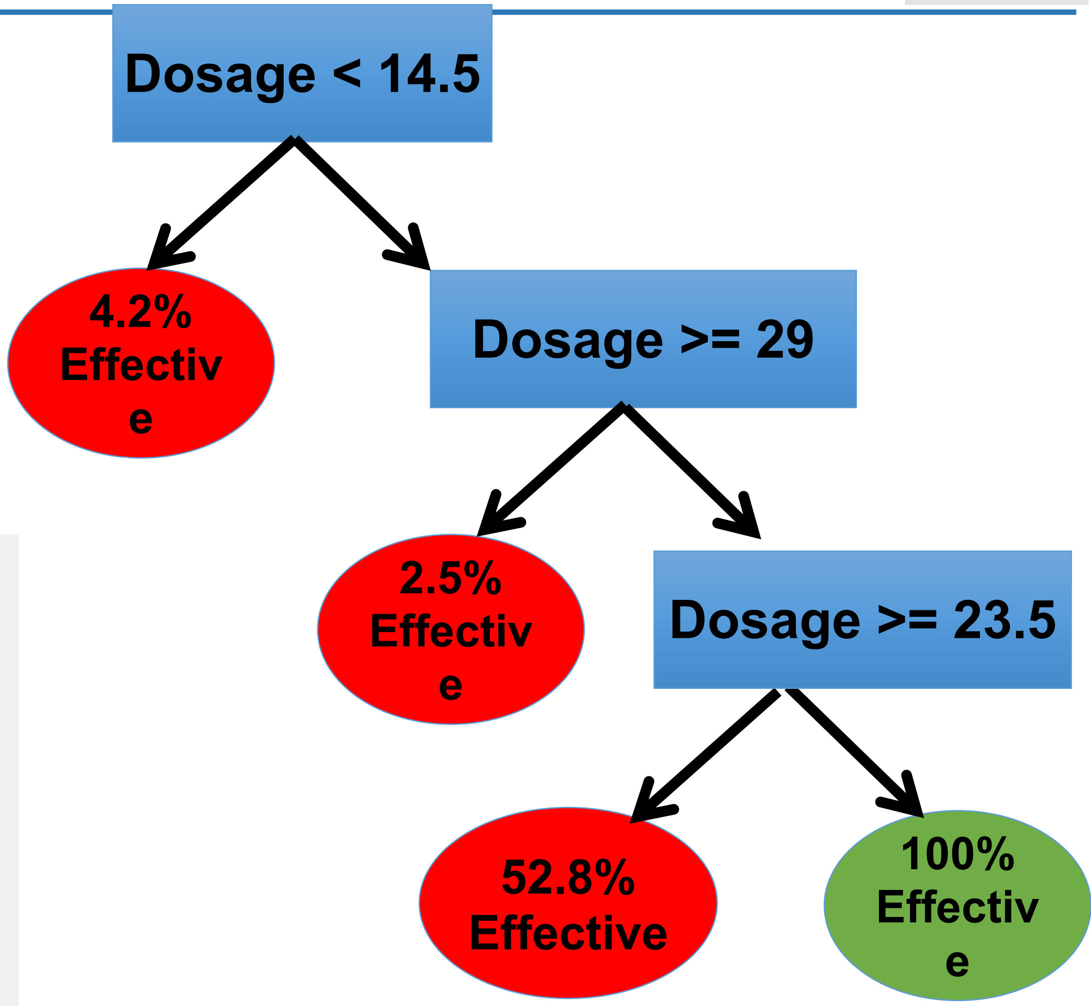




We can split them into two groups and we do that by finding the threshold that gives us the smallest sum of squared residuals

The average Drug Effectiveness for observations with Dosage between 23.5 & 29: 52.8%, as the output for leaf on the left





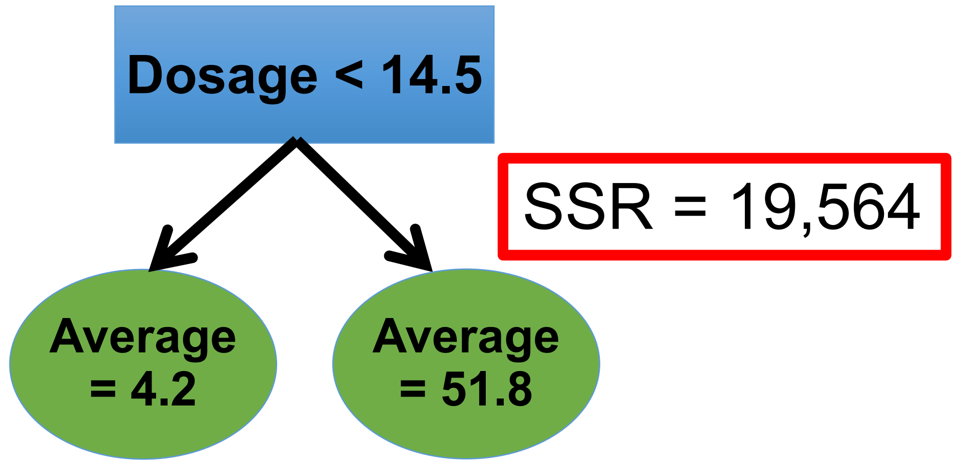
Each leaf corresponds to the average Drug Effectiveness from a cluster of observations

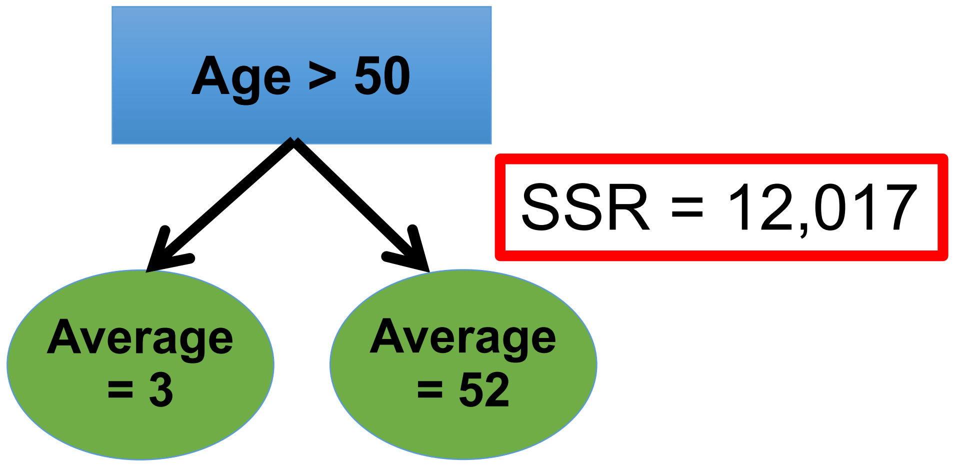
How to build a tree to predict Drug Effectiveness using a bunch of predictors

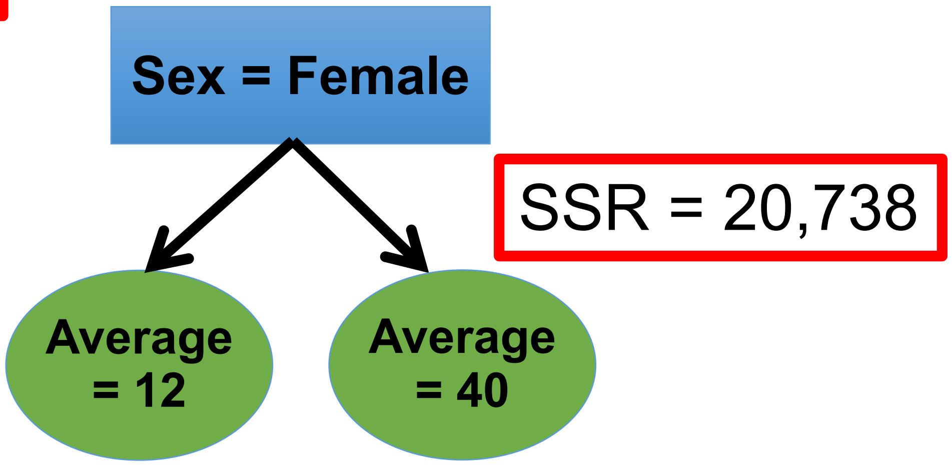
1. Build tree by using Dosage to predict Drug Effectiveness.
2. We will try different thresholds for Dosage and calculate the sum of squared residuals at each step and pick the minimum sum of squared residuals.
3. The best threshold becomes a candidate for the root
4. We focus on using Age to predict Drug Effectiveness and repeat steps 2 and 3.
5. Now we focus on Sex to predict Drug Effectiveness and repeat steps 2 and 3.

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.

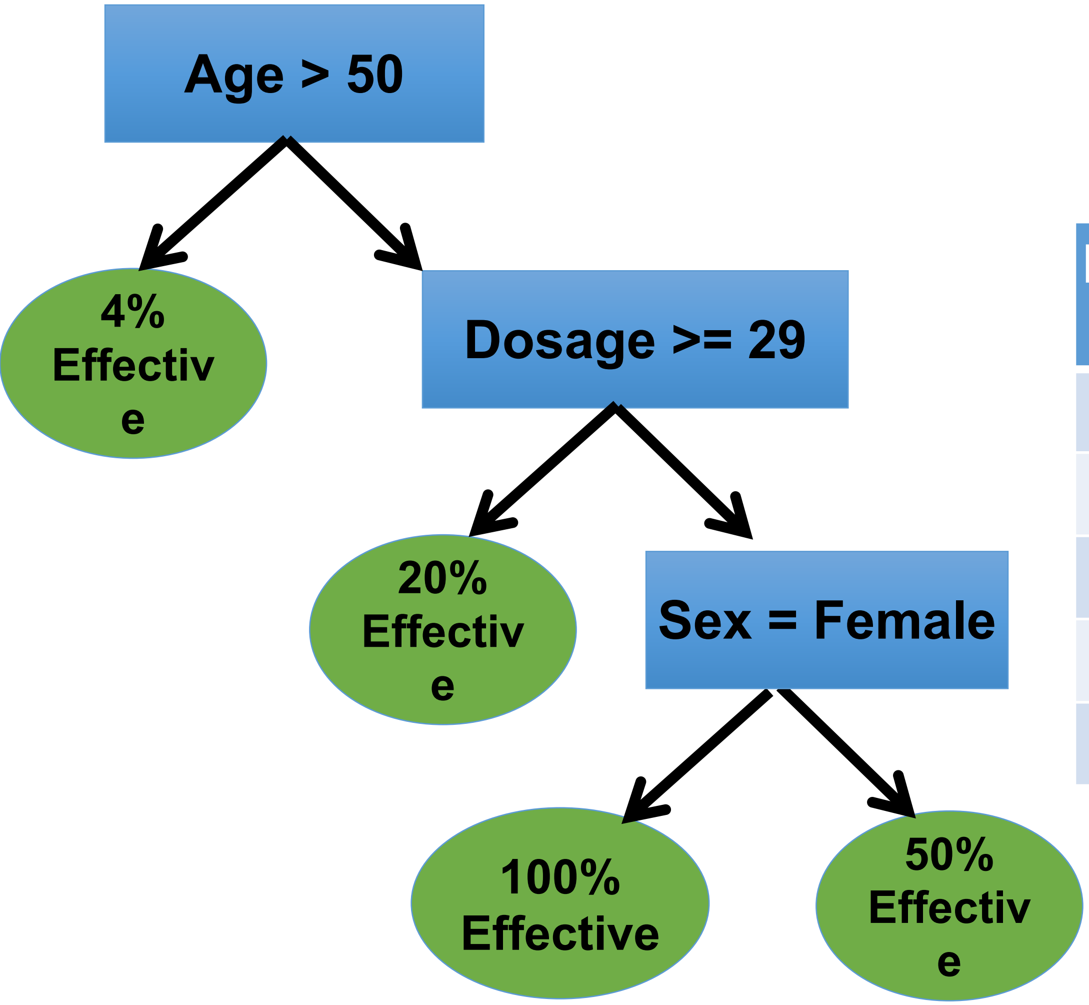
Comparison of squared residuals (SSRs) For each of the candidates







Regression Tree easily accommodates the additional predictors



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.

Numerical

Below table represents the drug effectiveness corresponding to the provided dosage.

Dosage	Drug effectiveness
2	0
8	11
10	15
13	18
16	56
20	72
22	98
25	100

1. Considering the threshold of the dosage > 3, calculate the squared residuals (SSR) of the dosage.

2. Considering the threshold of the dosage > 15, calculate the squared residuals (SSR) of the dosage.

Summary

- Regression Trees are a type of decision Trees
- In a Regression Tree, each leaf represents a numeric value
- We determine how to divide the observations by trying different thresholds and calculating the sum of squared residuals at each steps
- We determine how to divide the observations by trying different thresholds and calculating the sum of squared residuals at each step.
- The threshold with the smallest sum of squared residuals becomes a candidate for the root of the tree.
- If we have more than one predictor, we find the optimal threshold for each one and pick the candidate with the smallest sum of squared residuals to be the root.
- When we have fewer than some minimum number of observations in a node (7 in this example, but more commonly 20), then that node becomes a leaf otherwise we repeat the procedure to split the remaining observations until we can no longer split the observations into smaller groups and then we are done

Random Forests

- Random Forests made out of decision tress
- Decision Trees are easy to build, easy to use and easy to interpret but in practice they are not that awesome.
- Decision trees work great with the data used to create them, but they are not flexible when it comes to classifying new samples
- The good news is that Random Forests combine the simplicity of decision tress with flexibility resulting in a vast improvement in accuracy

Create a "bootstrapped" 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

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 are allowed to pick the sample more than once

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

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

In this example, we will only consider two variables (columns) at each step

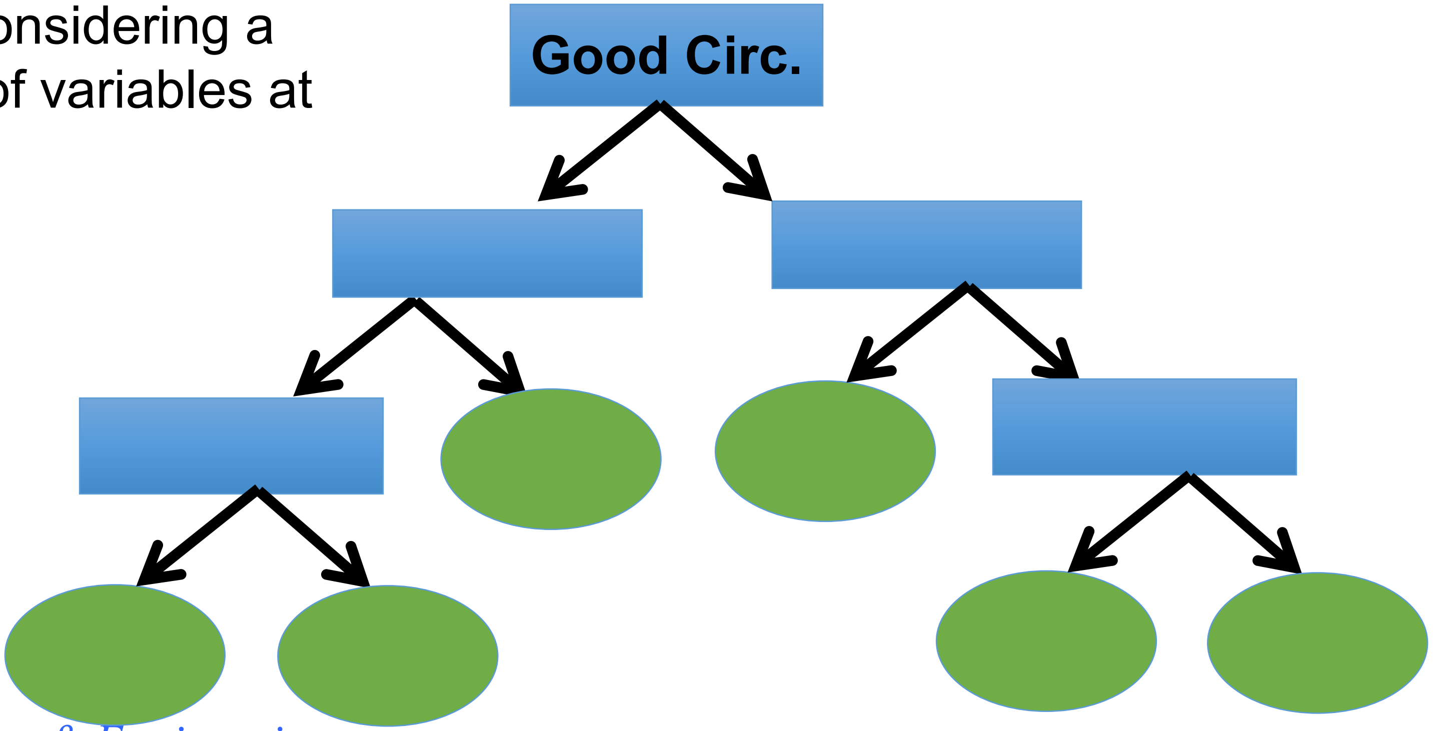
Thus instead of considering all 4 variables to figure out how to split the root node, we randomly select 2.

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

- In this case, we randomly selected Good Blood Circulation and Blocked Arteries as candidates for the root node.
- Just for the sake of example, assume the Good Blood Circulation did the best job separating the samples
- And we just build the tree as usual but only considering a random subset of variables at each step

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

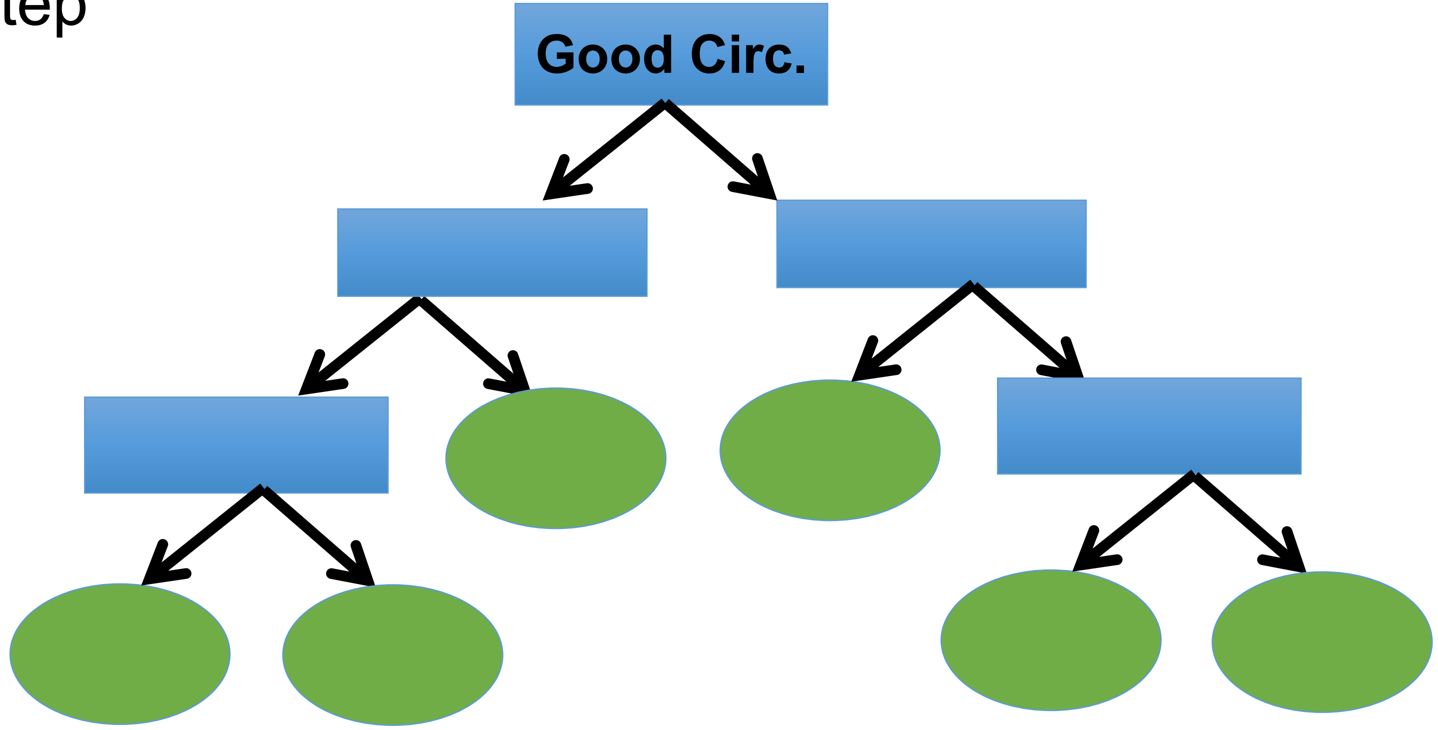


Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

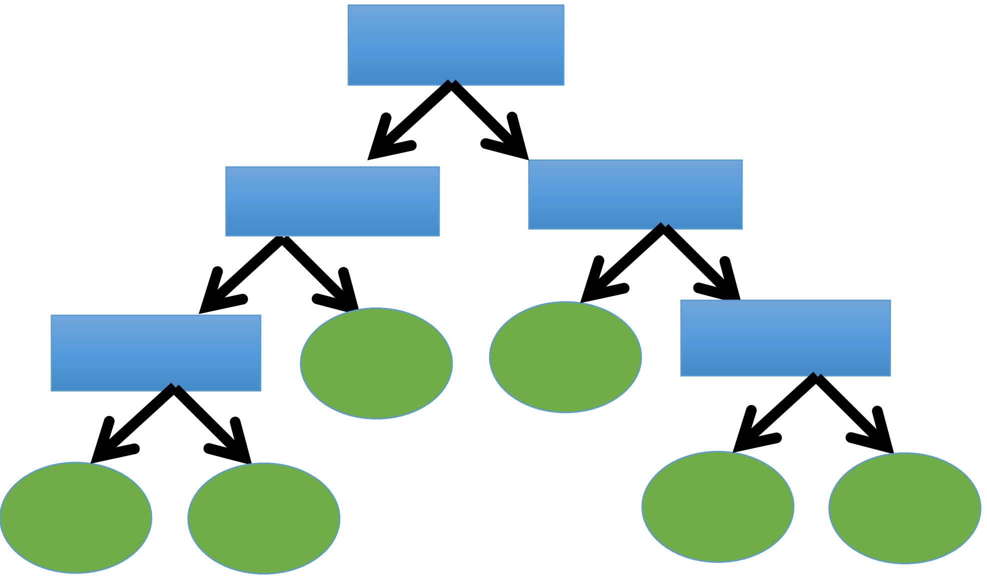
We built a tree:

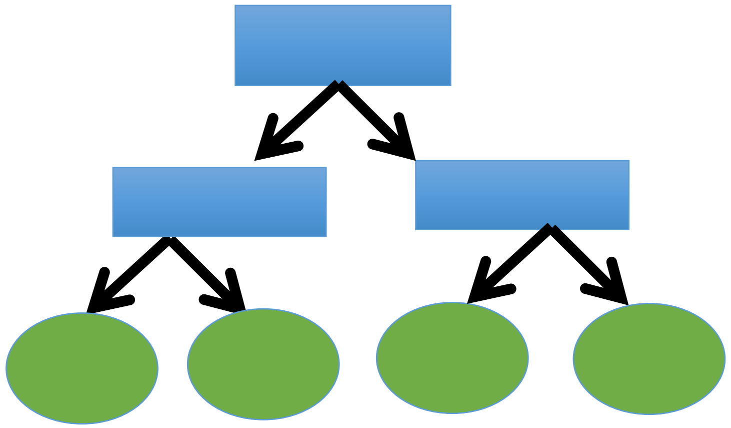
Using a bootstrapped dataset

Only considering a random subset of variables at each step

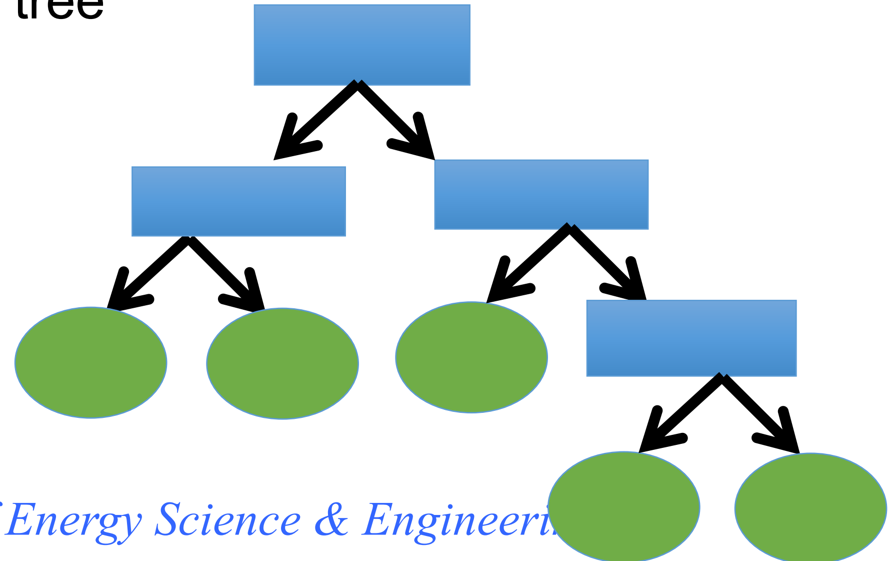


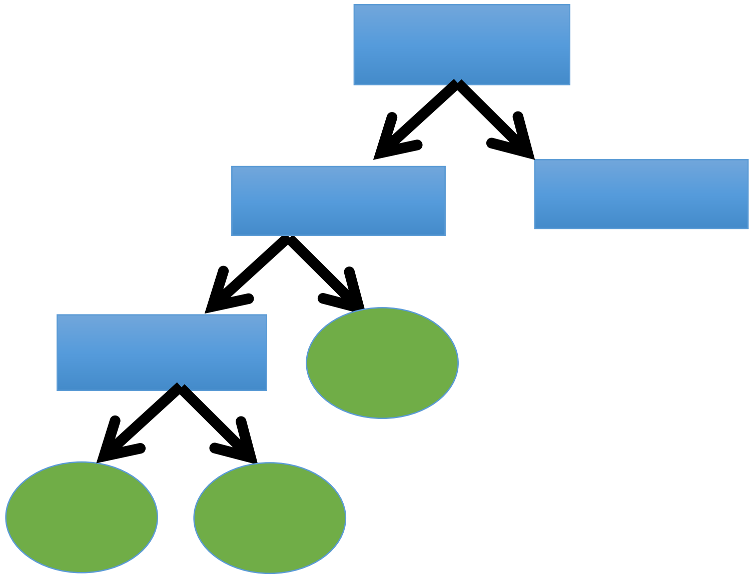
Using bootstrapped sample and considering only a subset of the variables at each step results in wide variety of trees





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

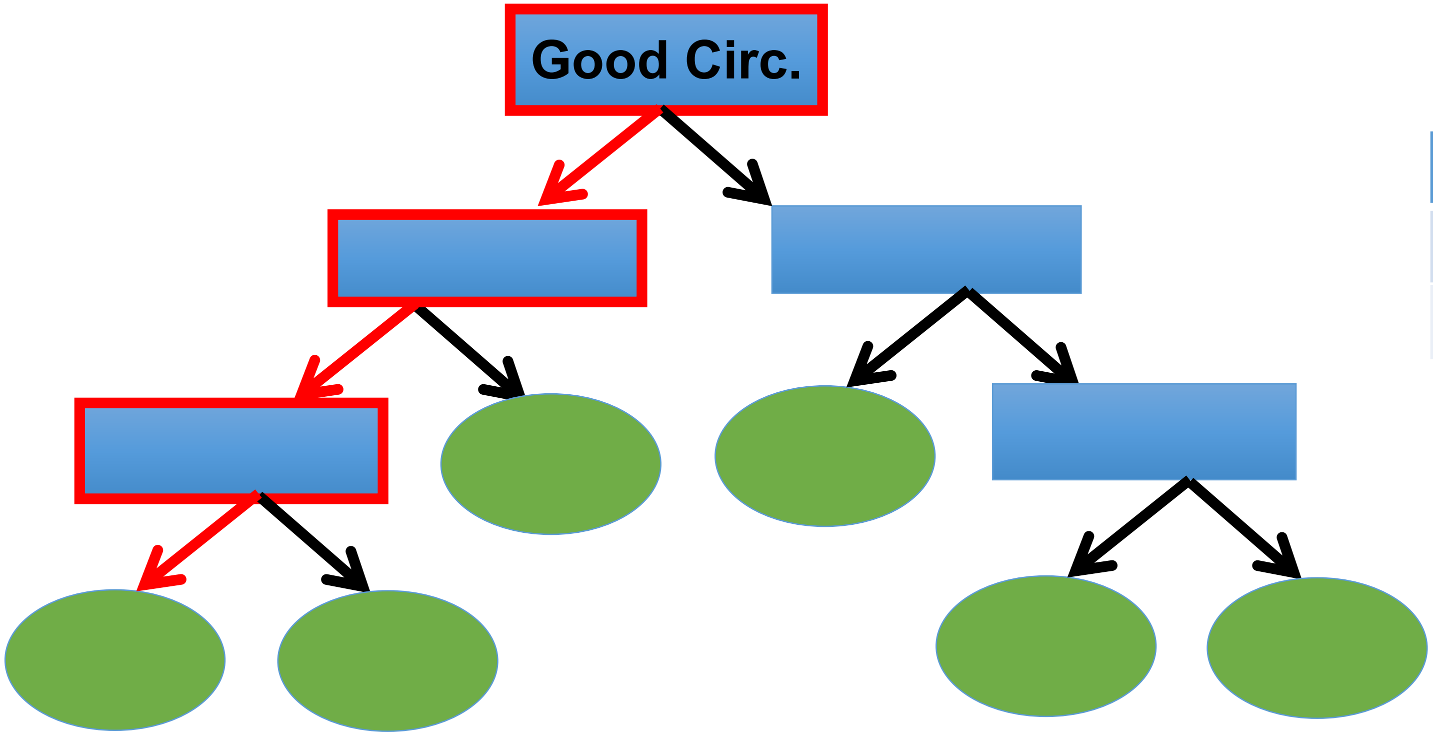




We have created Random Forest, how do we use it?

We get a new patient and got the measurements: we want to know if they have heart disease or not

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

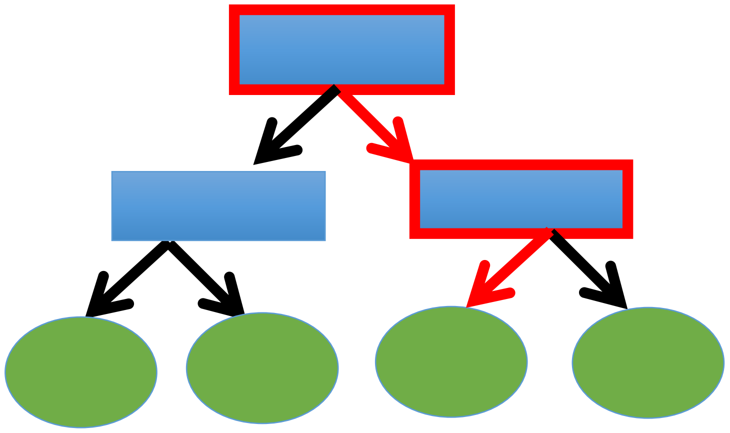


Heart Disease	
Yes	No
1	0

The first tree says "Yes"

We get a new patient and perform measurements: we want to know if they have heart disease or not

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



Heart Disease	
Yes	No
2	0

The second tree also says "Yes"

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**

Then we repeat for all the trees that we have made

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

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

Remember when we created 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

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

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

Typically, about 1/3 of the original data doesn't end up in the bootstrapped dataset

Out-Of-Bag 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

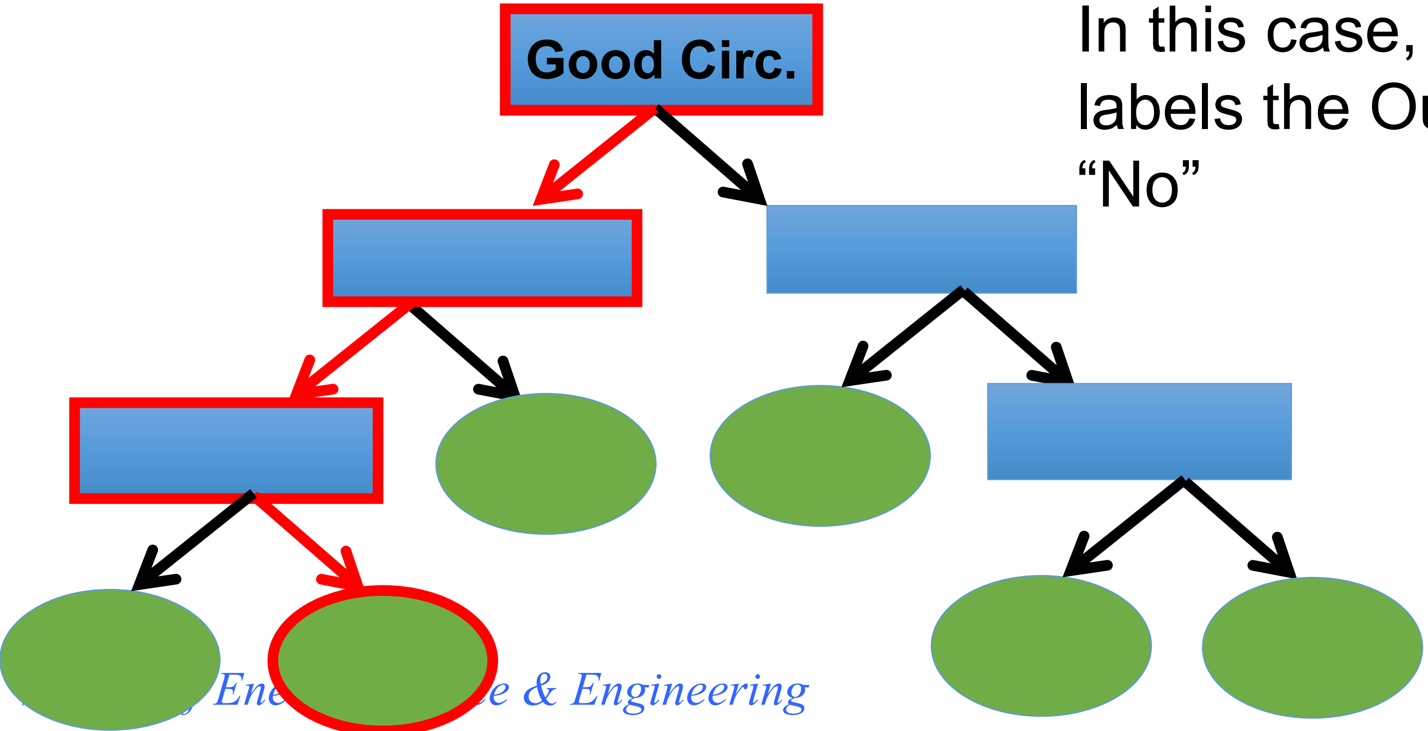
This is called the “Out-Of-Bag Dataset”

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

Out-Of-Bag Data was not used to create tree

We can runt it through and see if it correctly classifies the samples as “No Heart Disease”

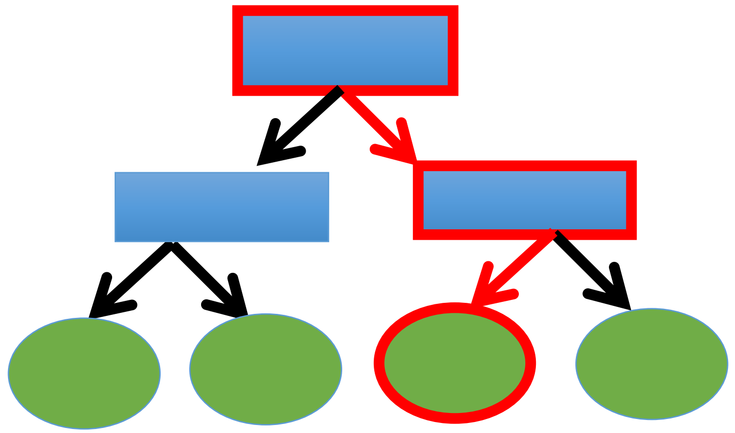
Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Diseas e
Yes	Yes	No	210	No



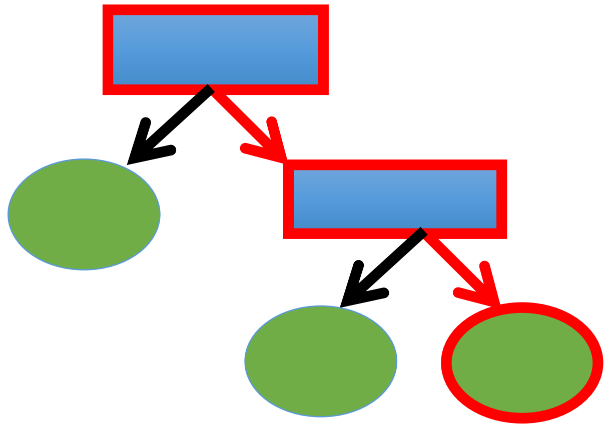
In this case, the tree correctly labels the Out-of-Bag sample “No”

We run this Out-Of-Bag samples through all of the other trees that were built without it

Chest Pain	Good Blood Circ	Blocked Arteries	Weight	Heart Diseas e
Yes	Yes	No	210	No



This tree incorrectly labeled the Out-of-Bag sample "Yes"



This tree correctly labeled the Out-of-Bag sample "No"

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

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

The proportion of Out-Of-Bag samples that were incorrectly classified is the “Out-Of-Bag Error”

Summary

We now know how to:

- Build a Random Forest
- Use Random Forest
- Estimate the accuracy of a Random Forest

Remember when we built our first tree and we only used 2 variables to make a decision at each step

We can compare the Out-Of-Bag error for a random forest built using 2 variables per step to random forest built using three variables per step and we test a bunch of different settings and choose the most accurate random forest