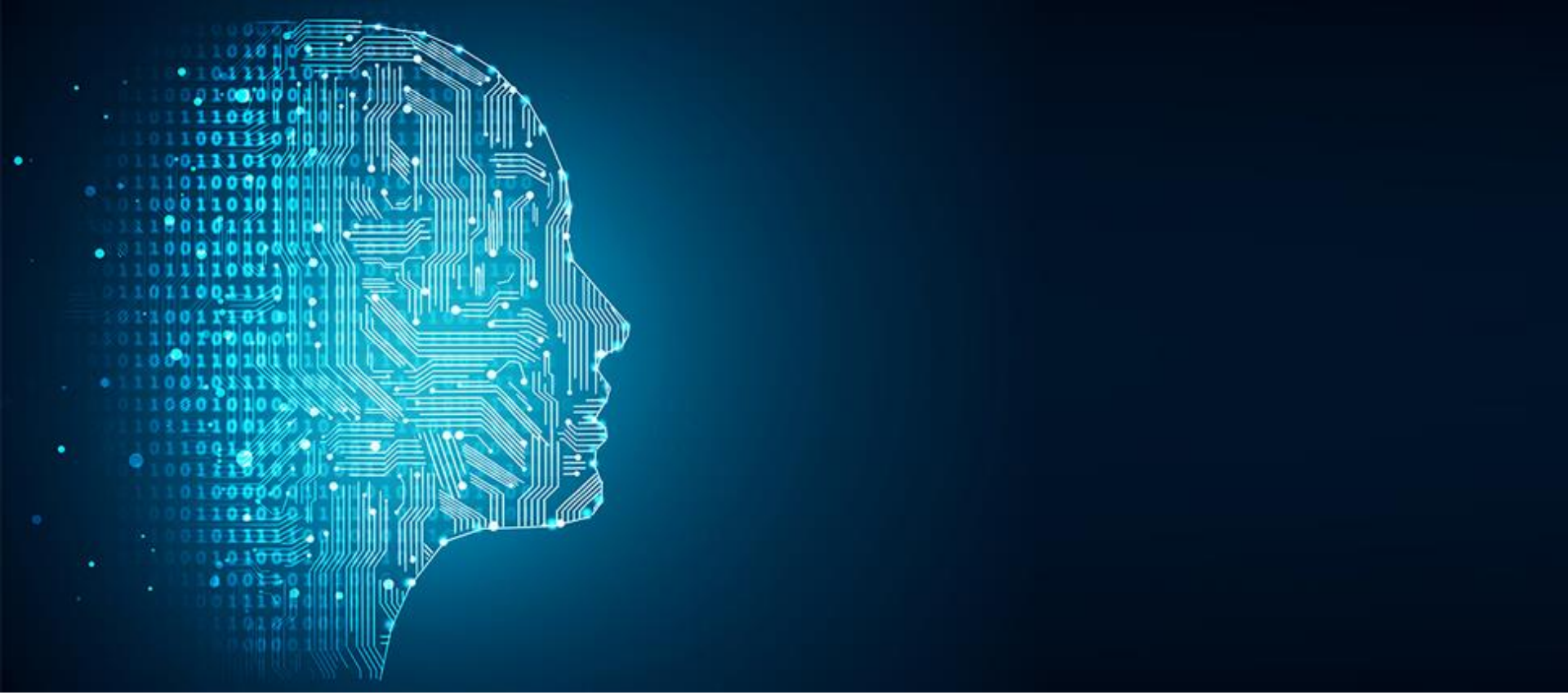
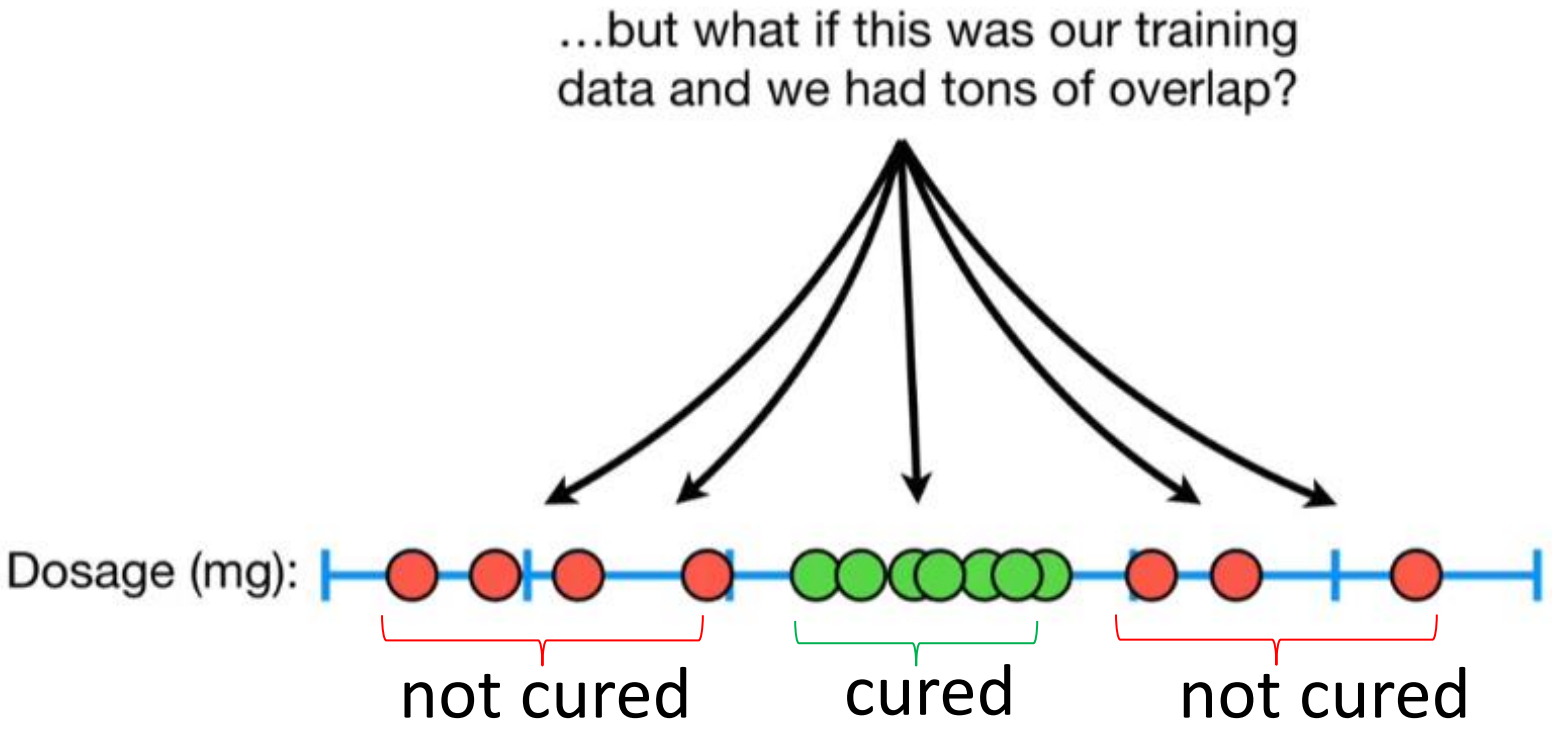


COMP SCI 1400
AI Technologies

Machine Learning Basis – Decision tree, Kmeans, PCA

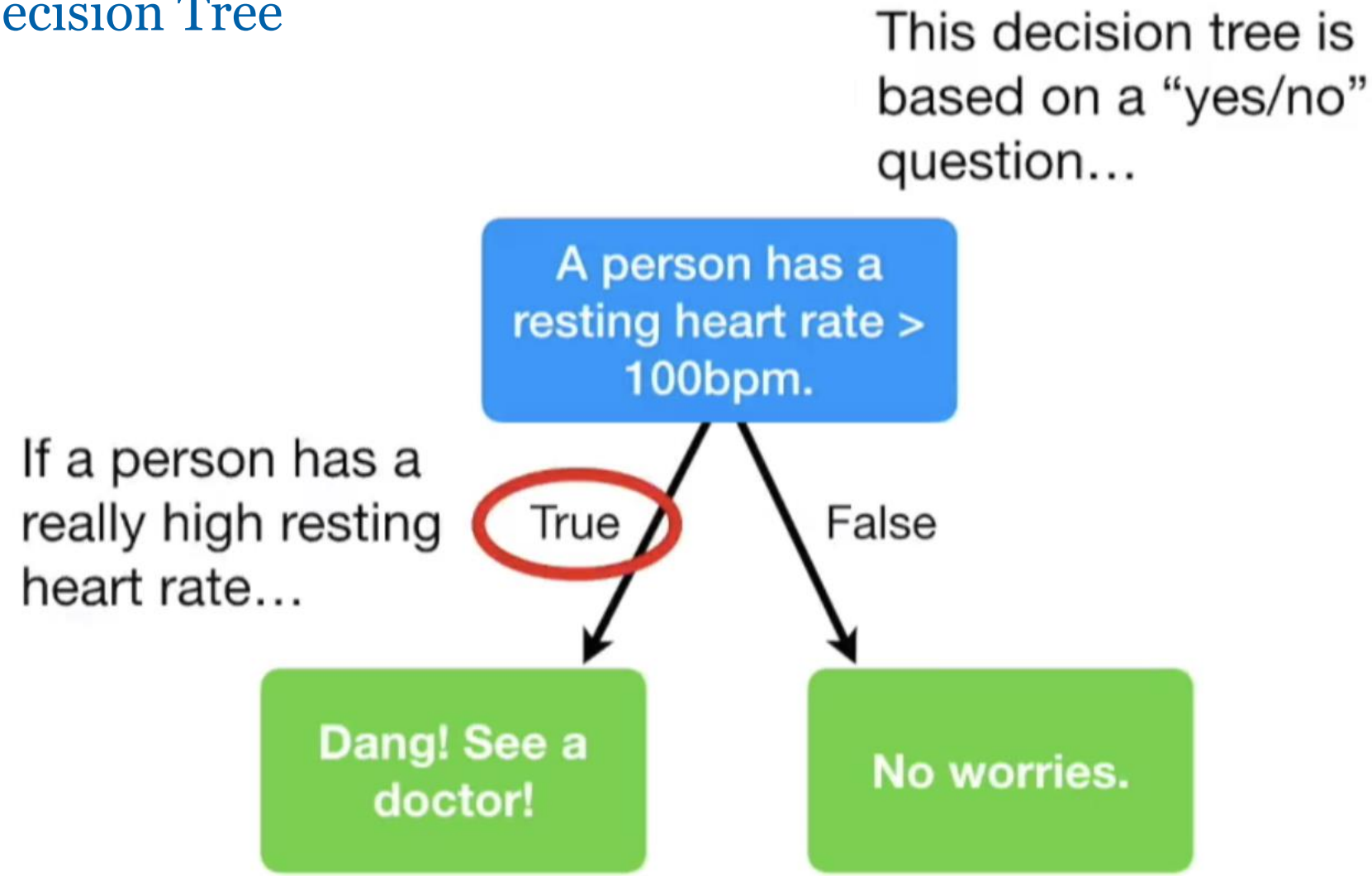
Dr Kamal Mammadov

Review this case

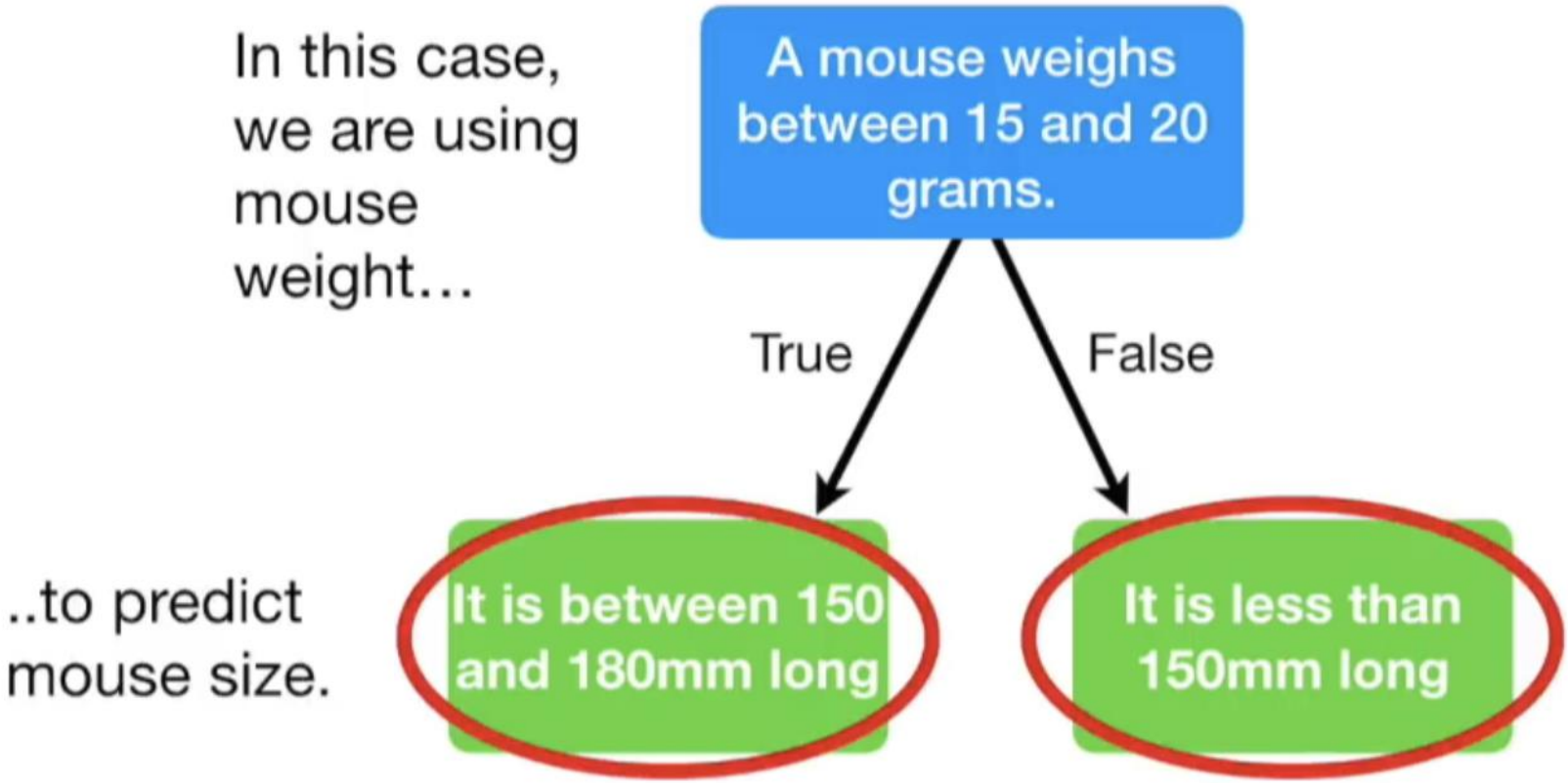


Decision Tree

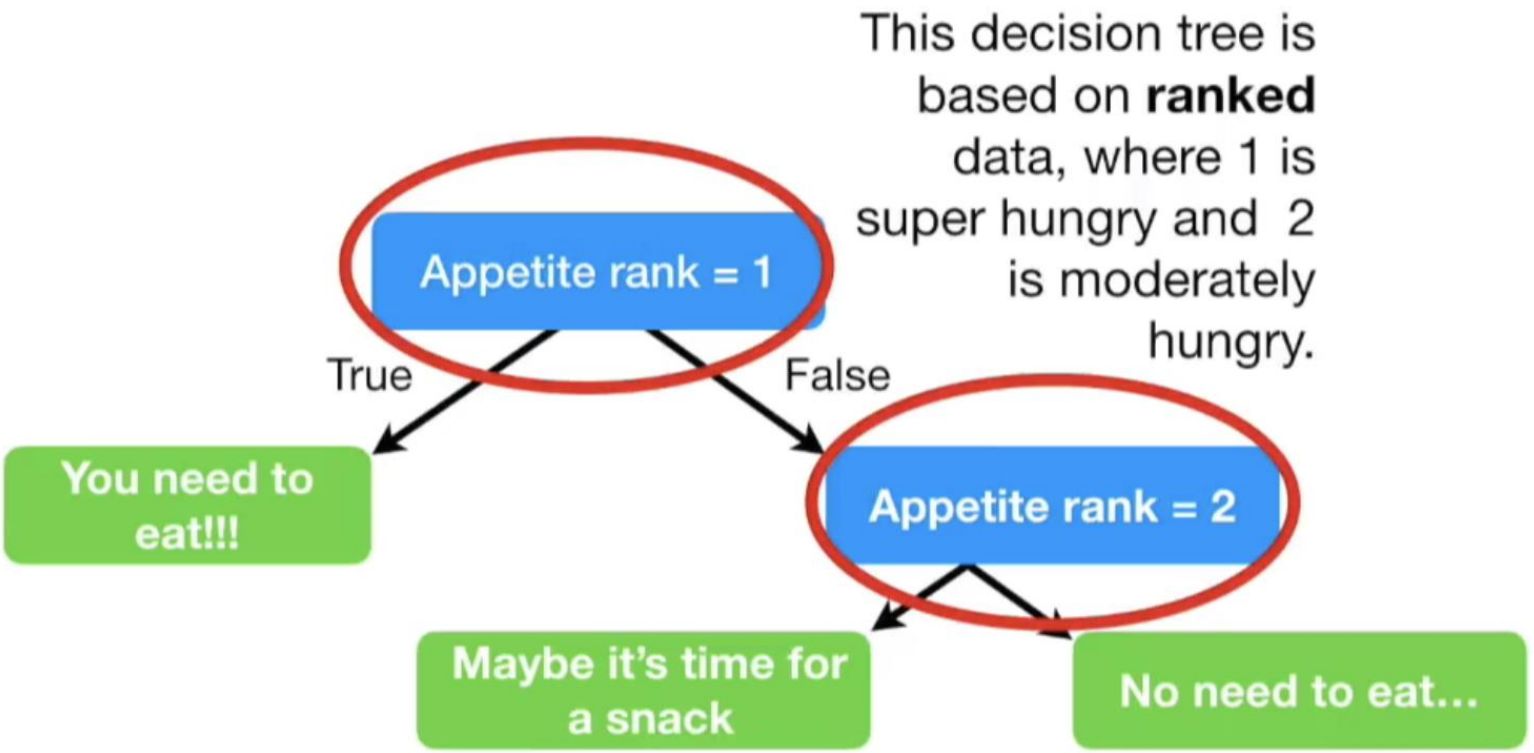
- Decision Tree



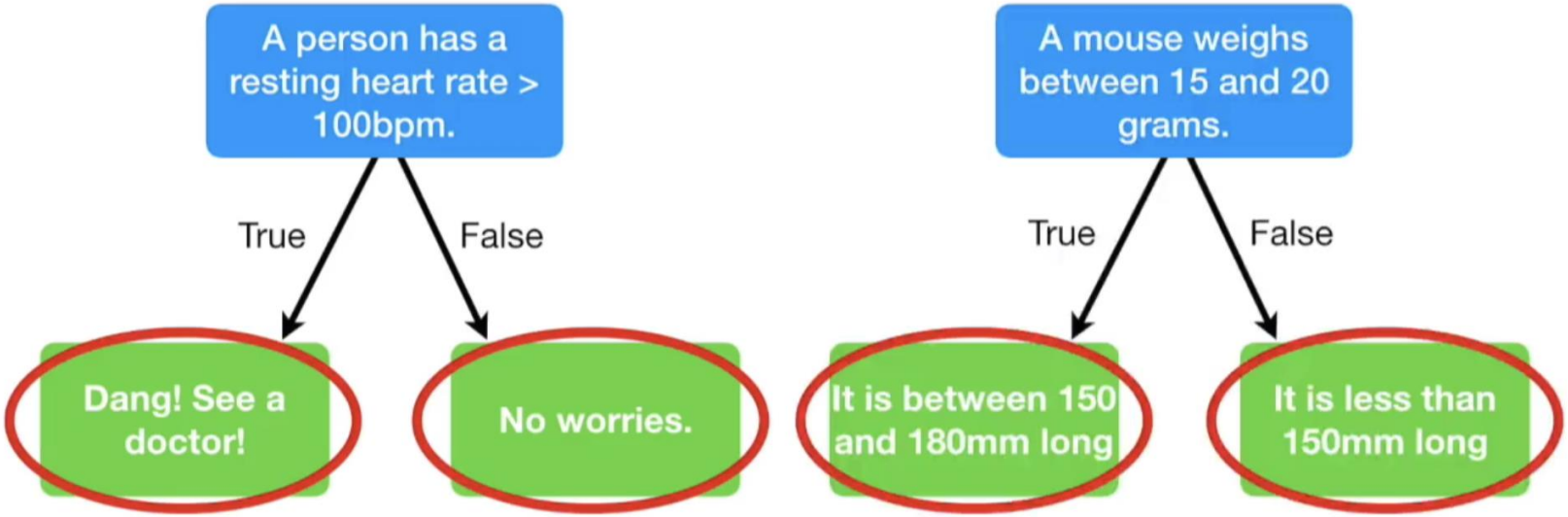
- # Decision Tree



Decision Tree



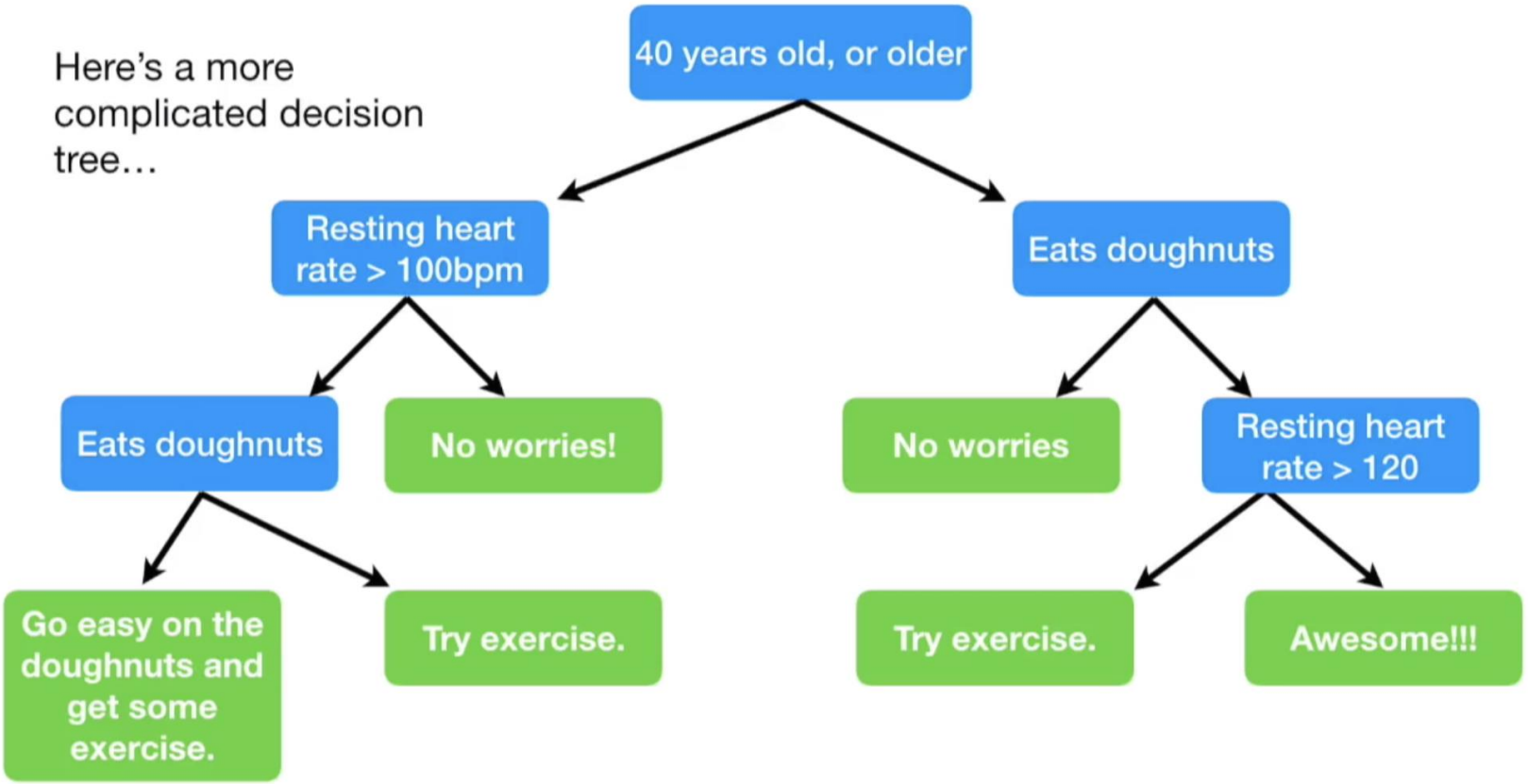
Decision Tree



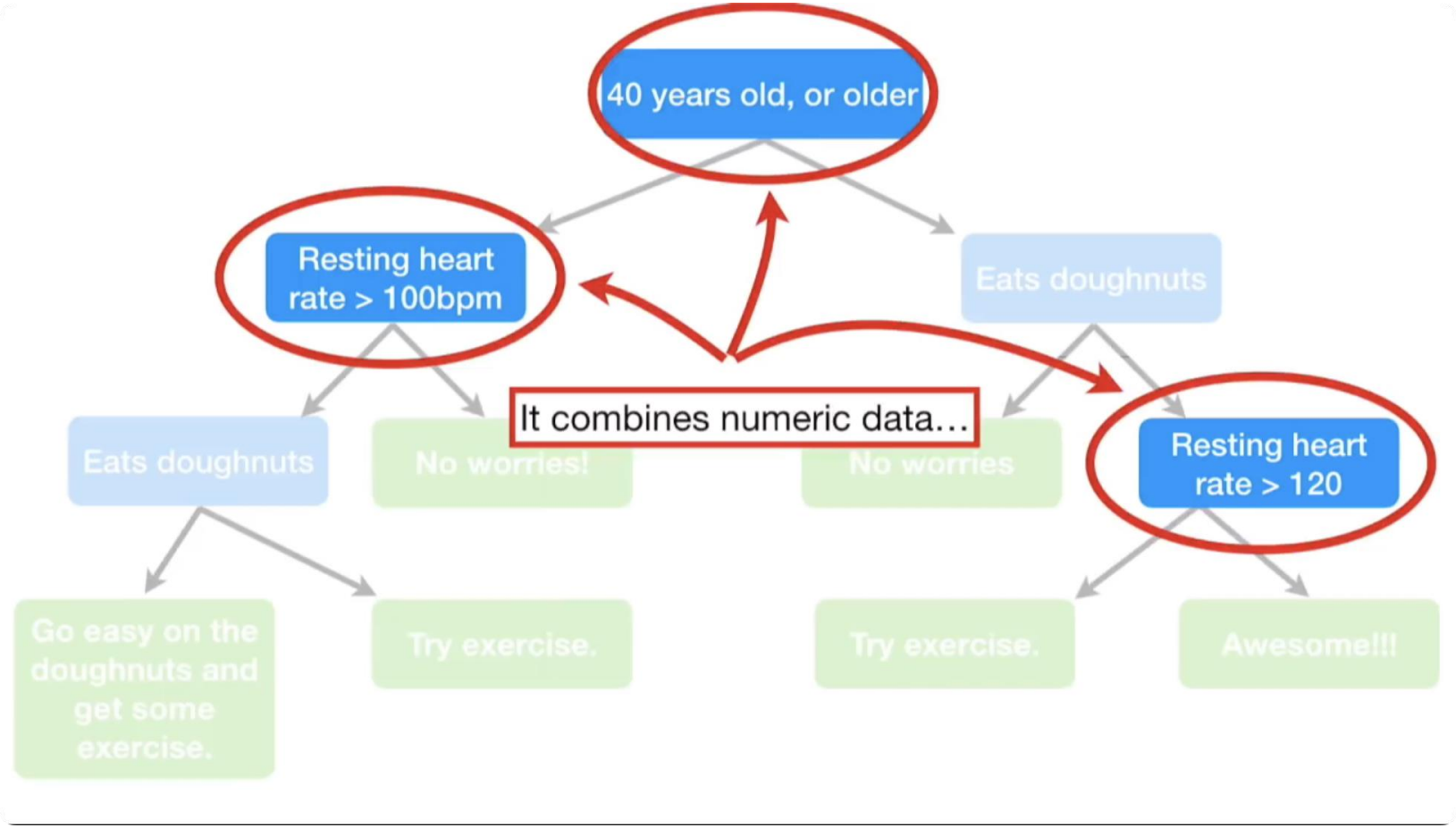
NOTE: The classification can be categories…

…or numeric.

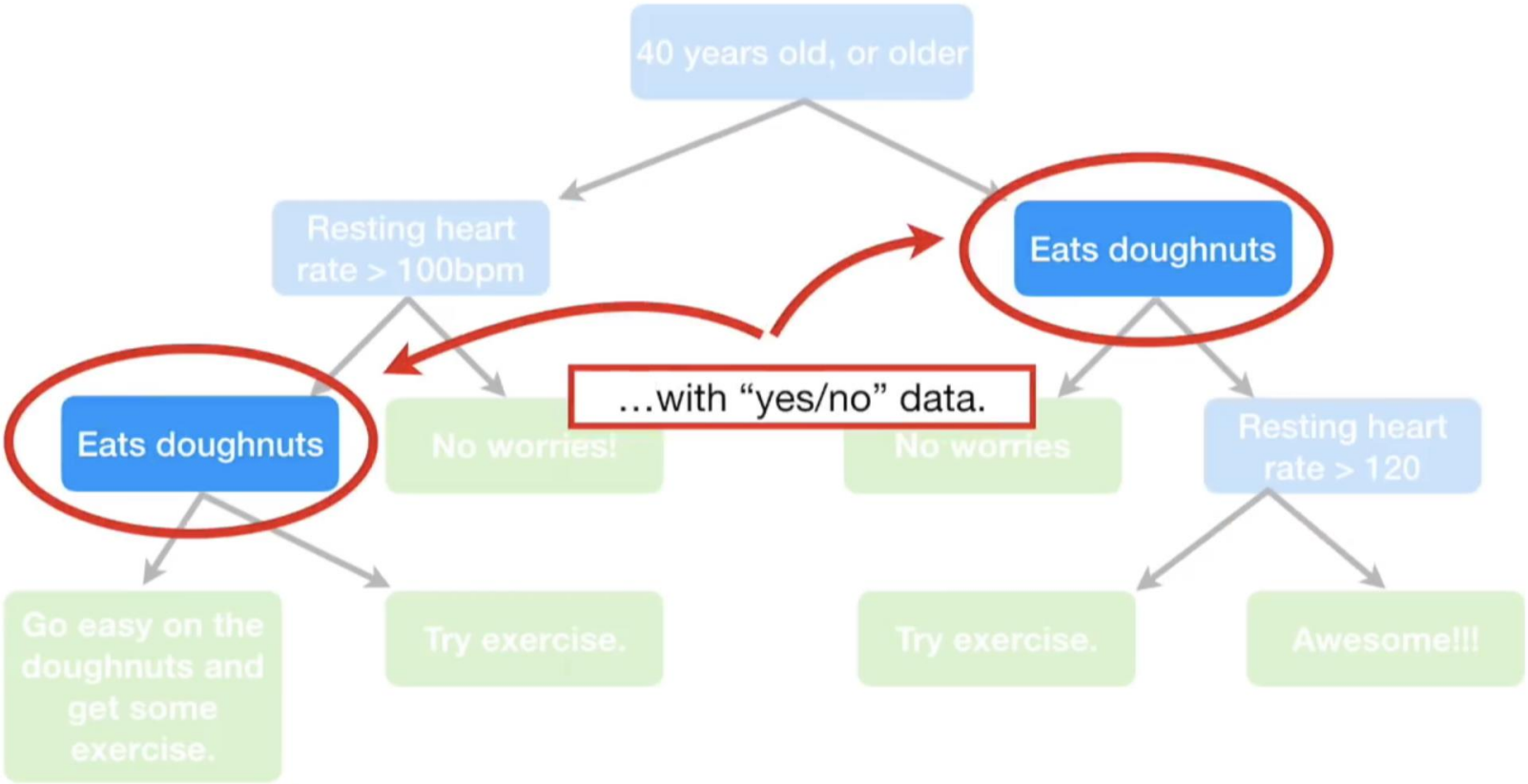
Decision Tree



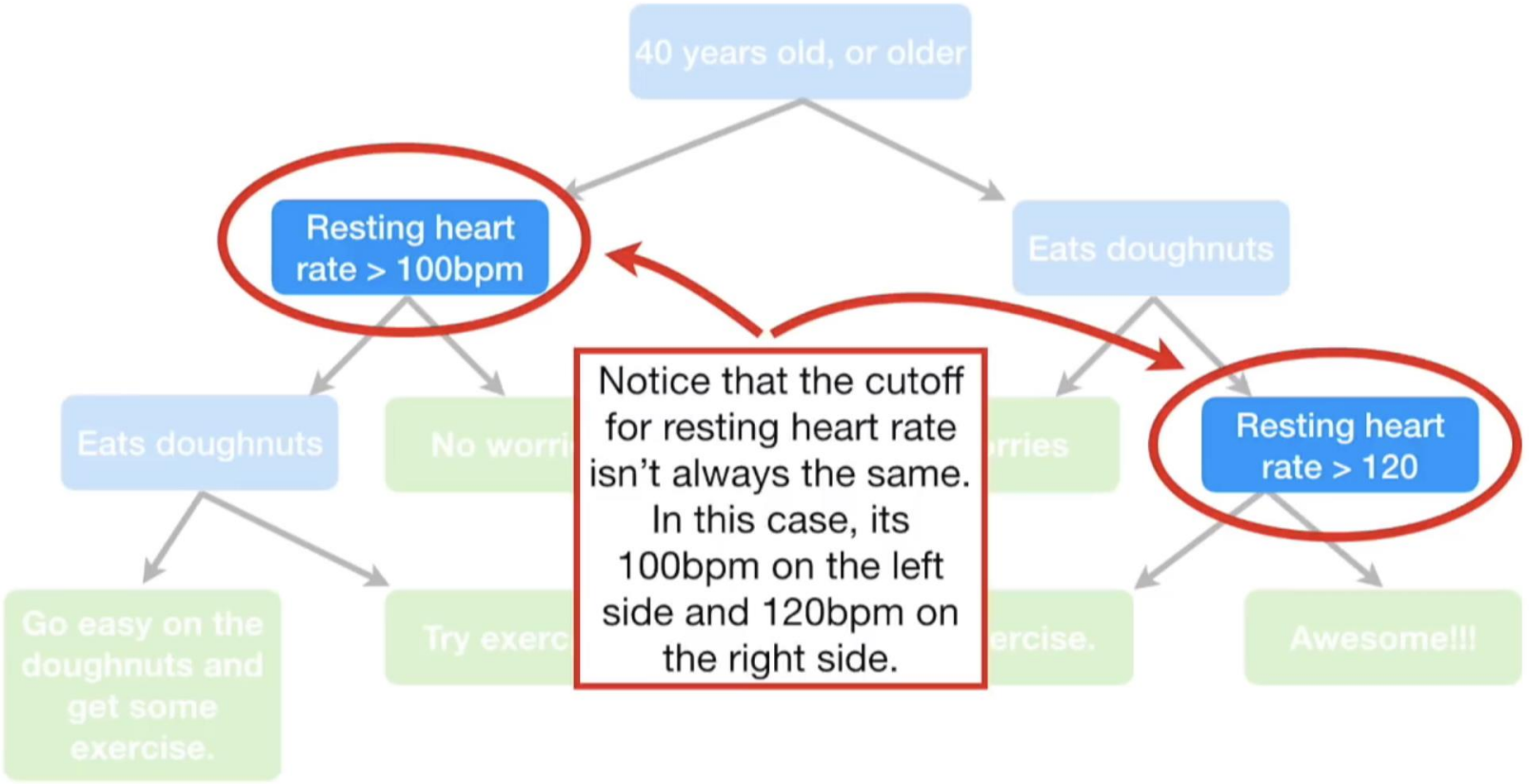
Decision Tree



Decision Tree

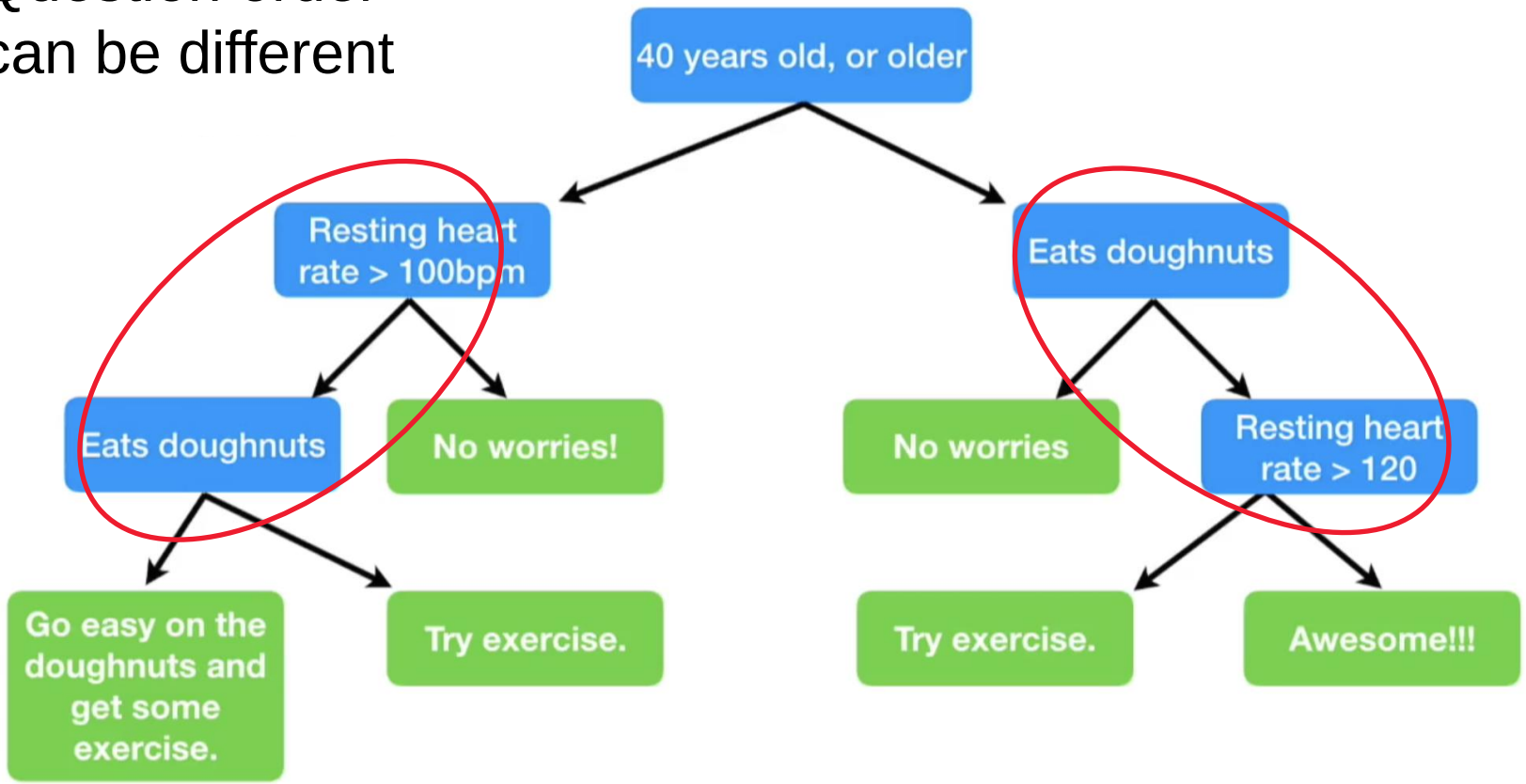


Decision Tree

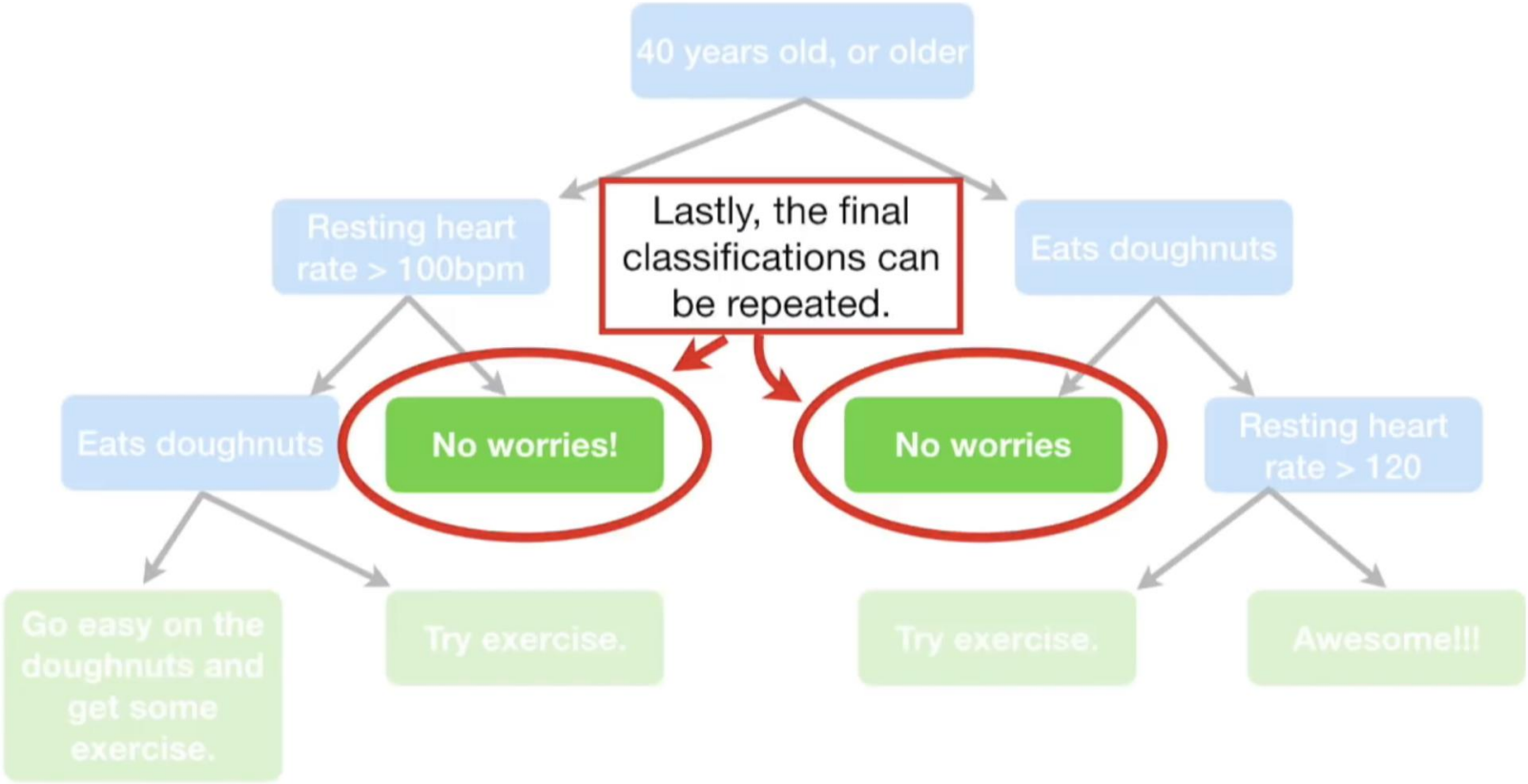


Decision Tree

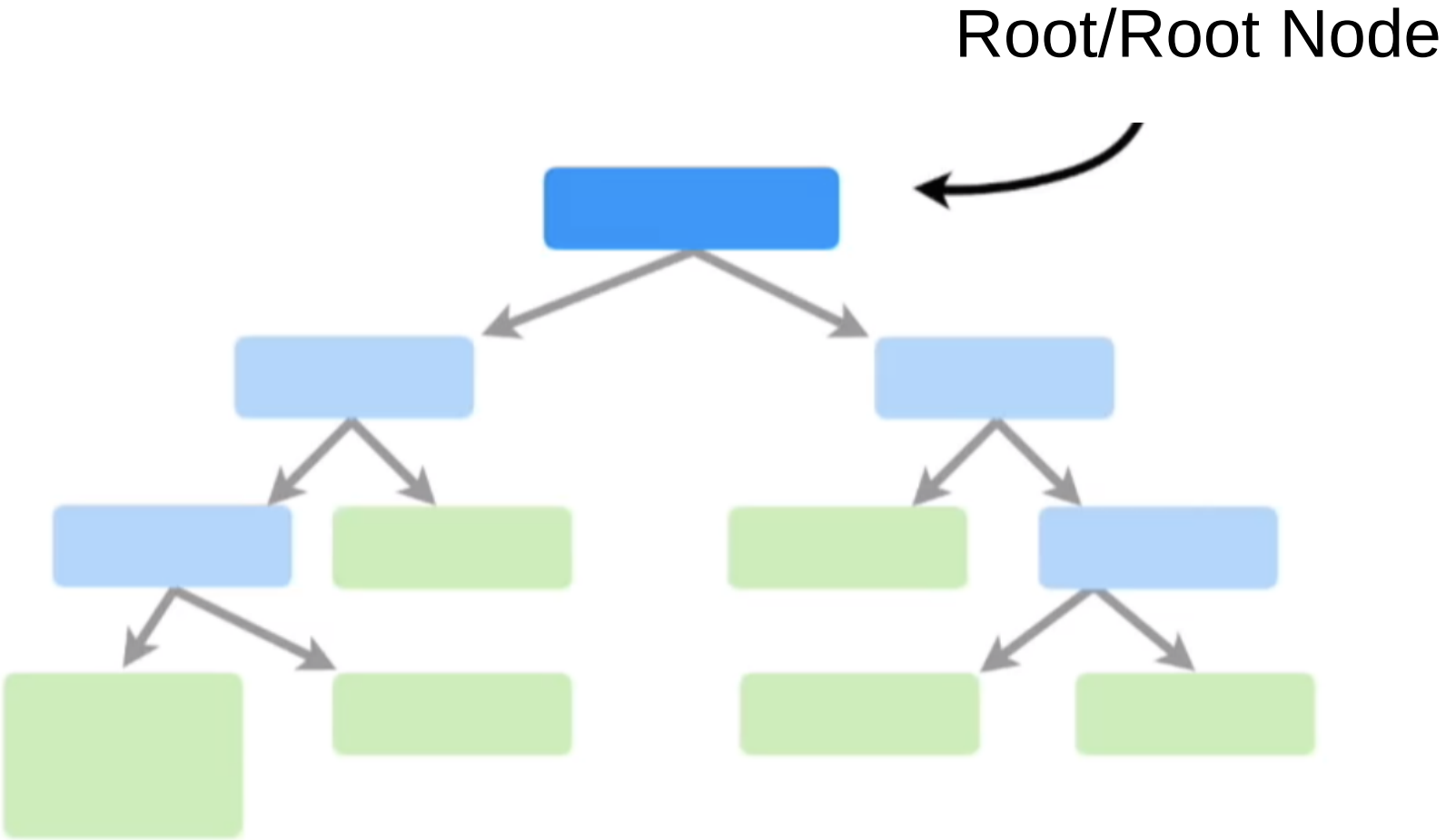
Question order can be different



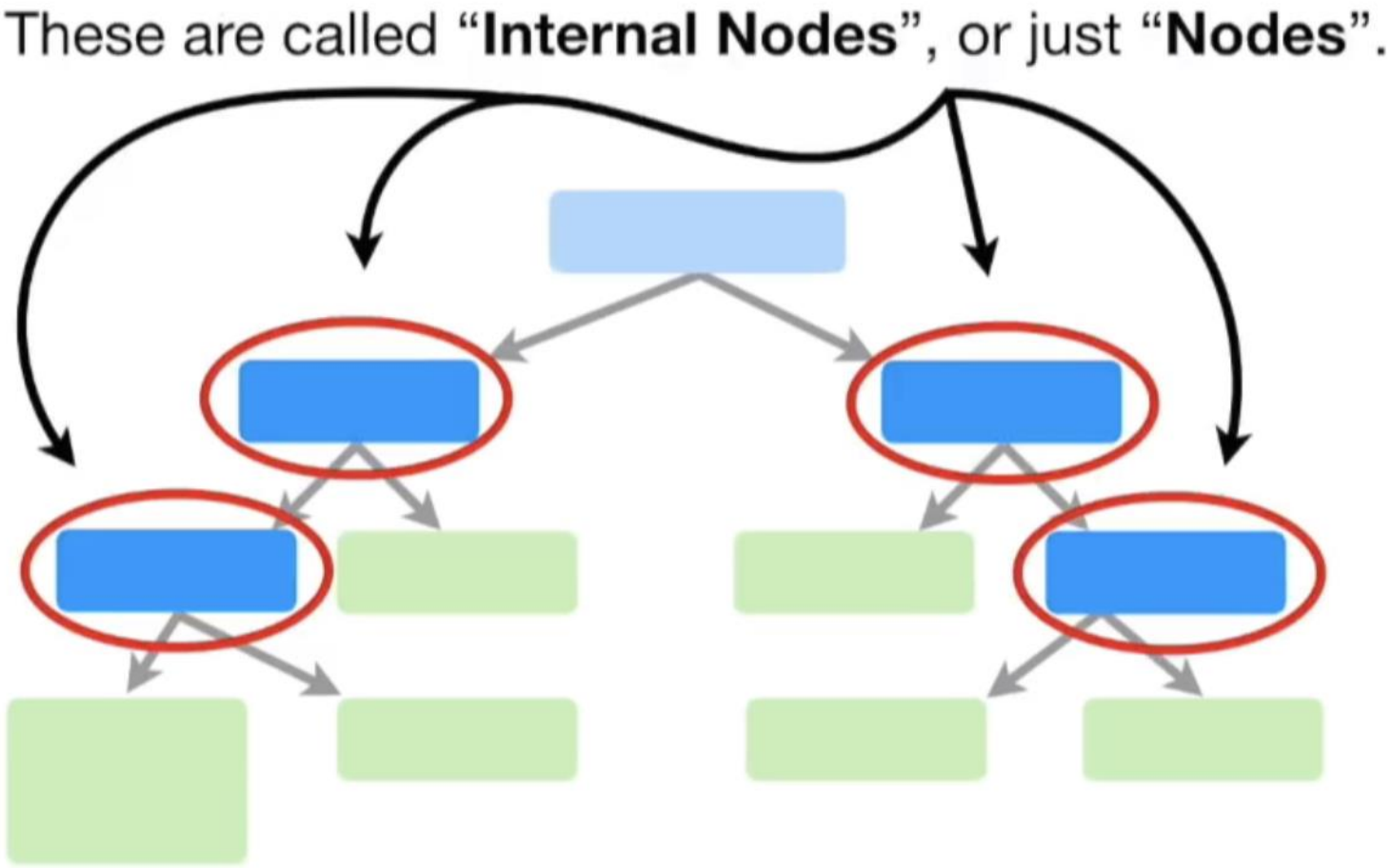
Decision Tree



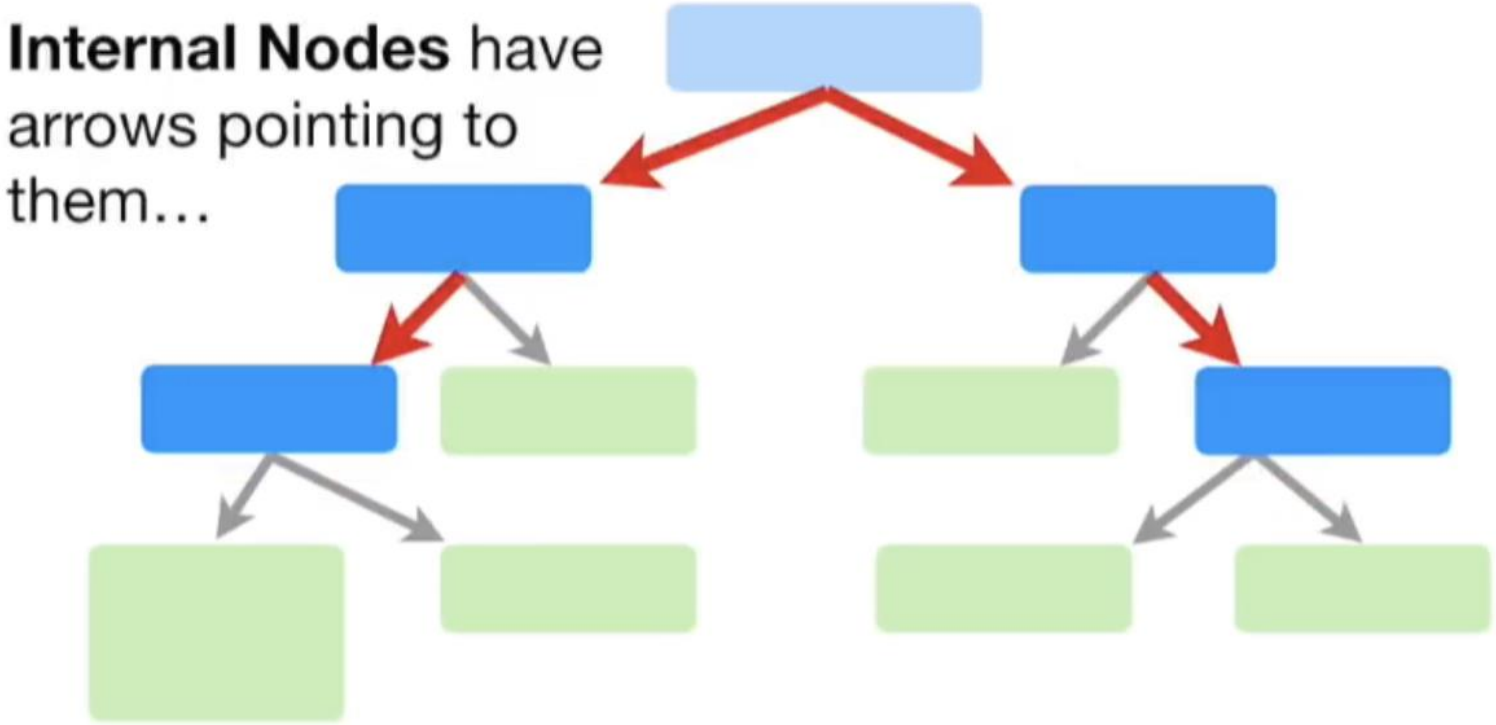
Decision Tree



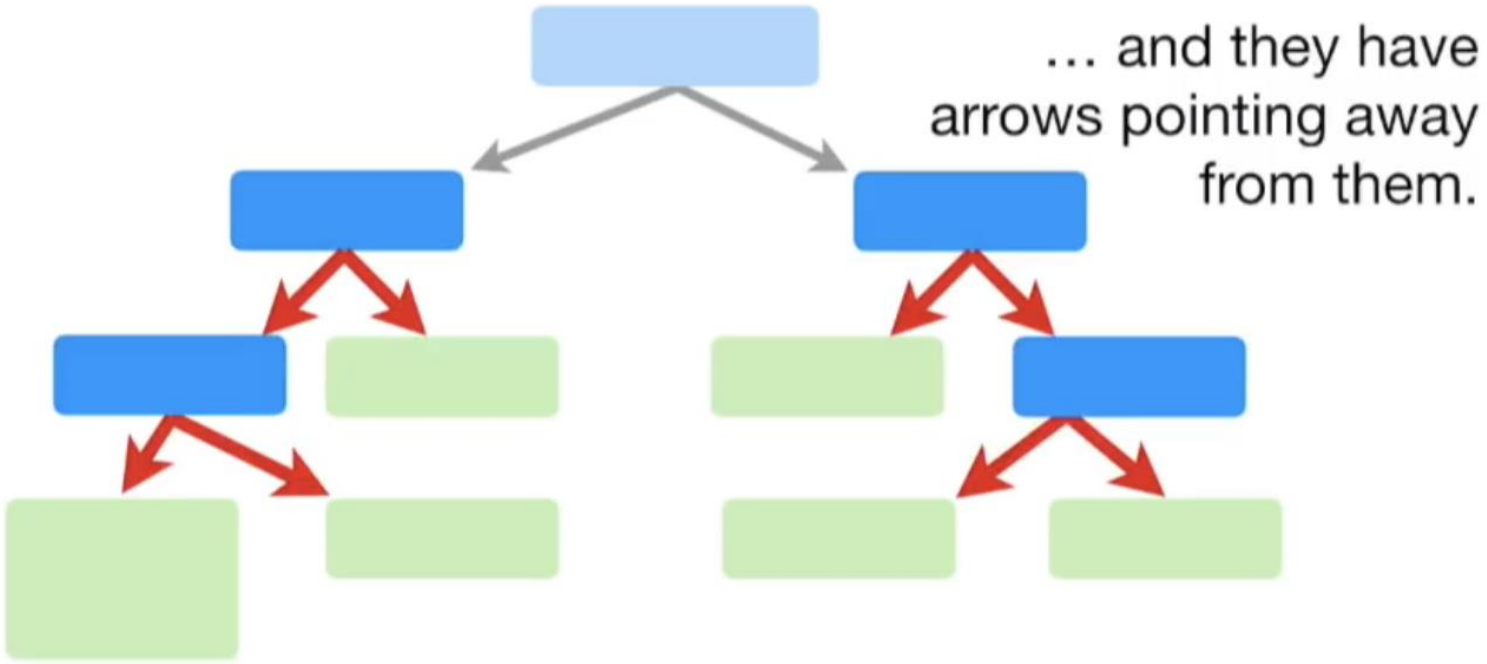
Decision Tree



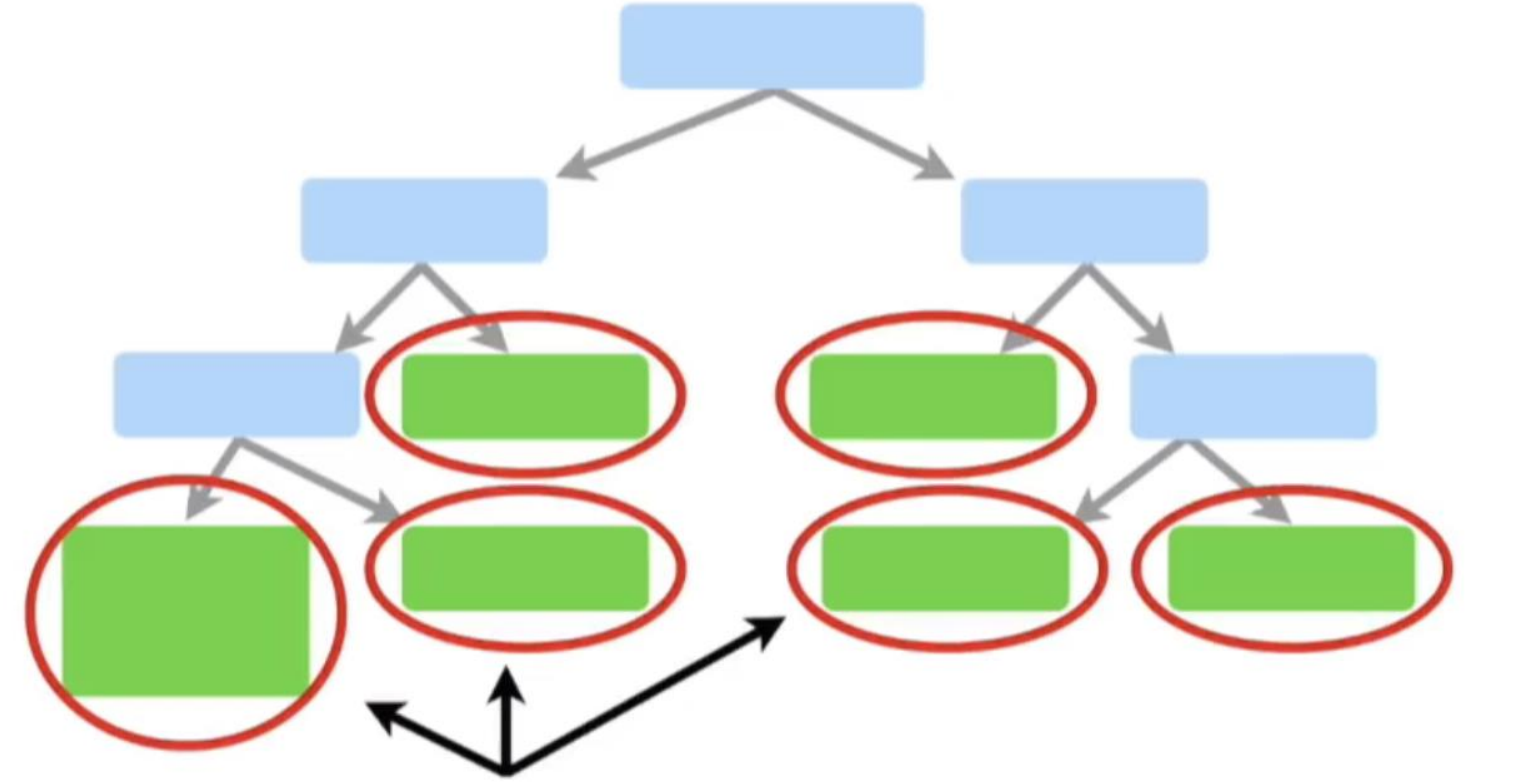
Decision Tree



Decision Tree



Decision Tree



Lastly, these are called “**Leaf Nodes**”, or just “**Leaves**”

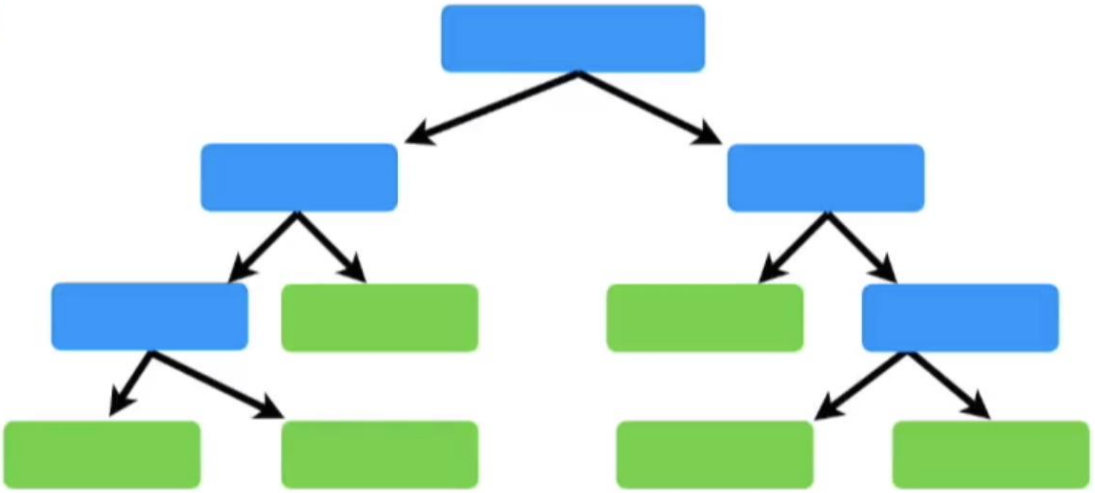
Decision Tree

Build a decision tree that uses chest pain, blood circulation status, and artery status to predict whether a heart disease present

Now we are ready to talk about how to go from a raw table of data…

Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc…	etc…	etc…	etc…

…to a decision tree!!!



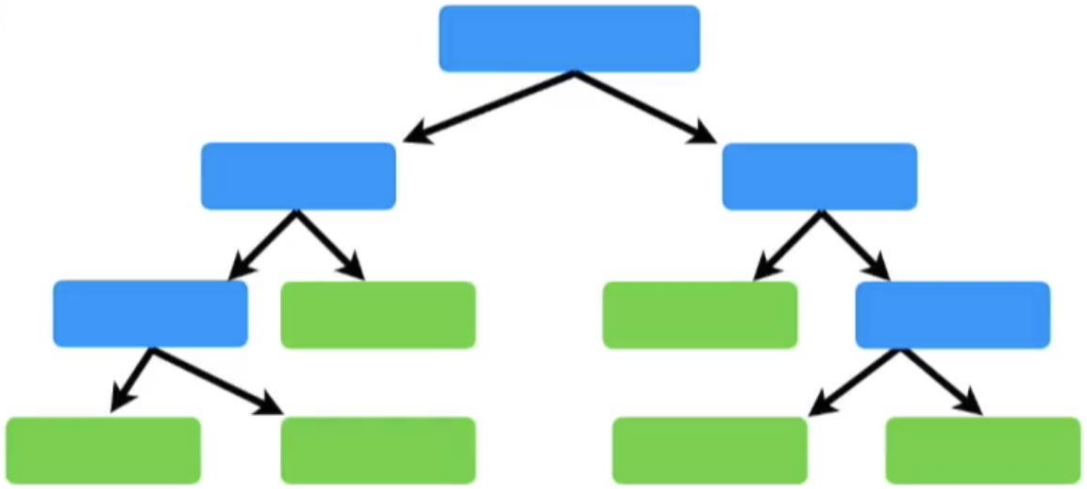
Decision Tree

Which variable/column should be the root?

Now we are ready to talk about how to go from a raw table of data…

Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc…	etc…	etc…	etc…

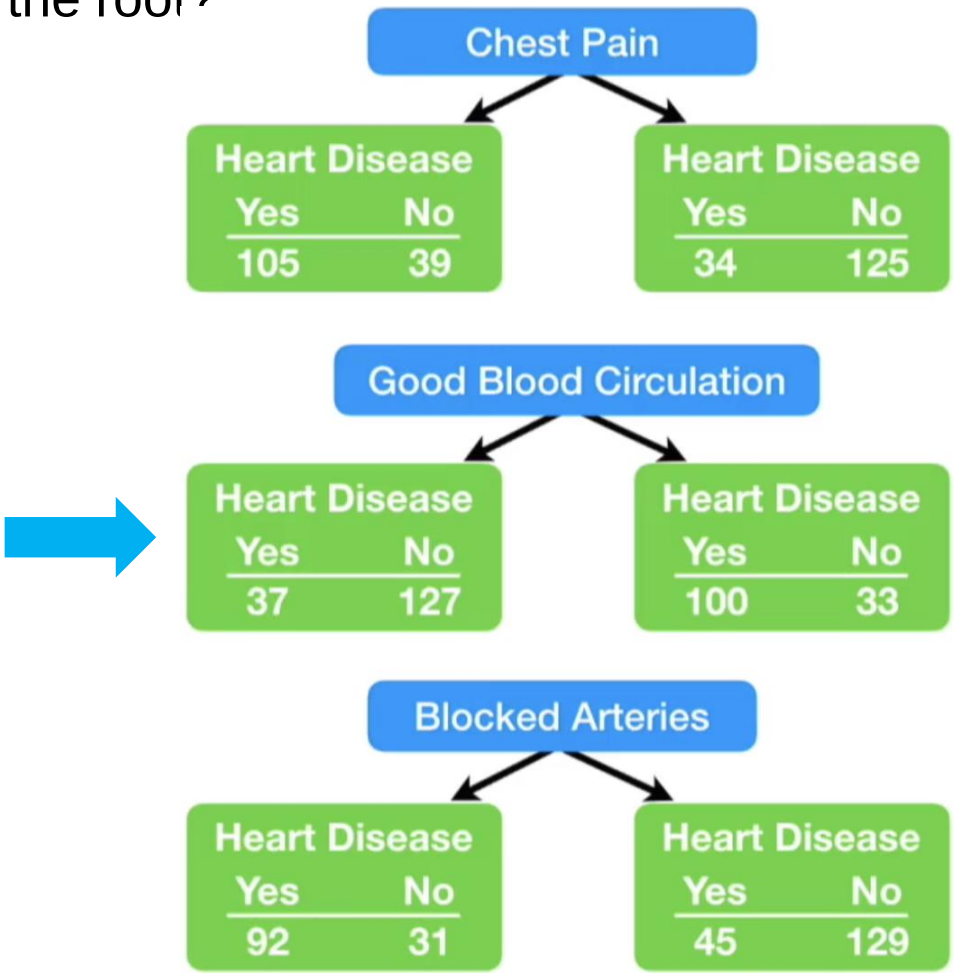
…to a decision tree!!!



Decision Tree

Which variable/column should be the root?

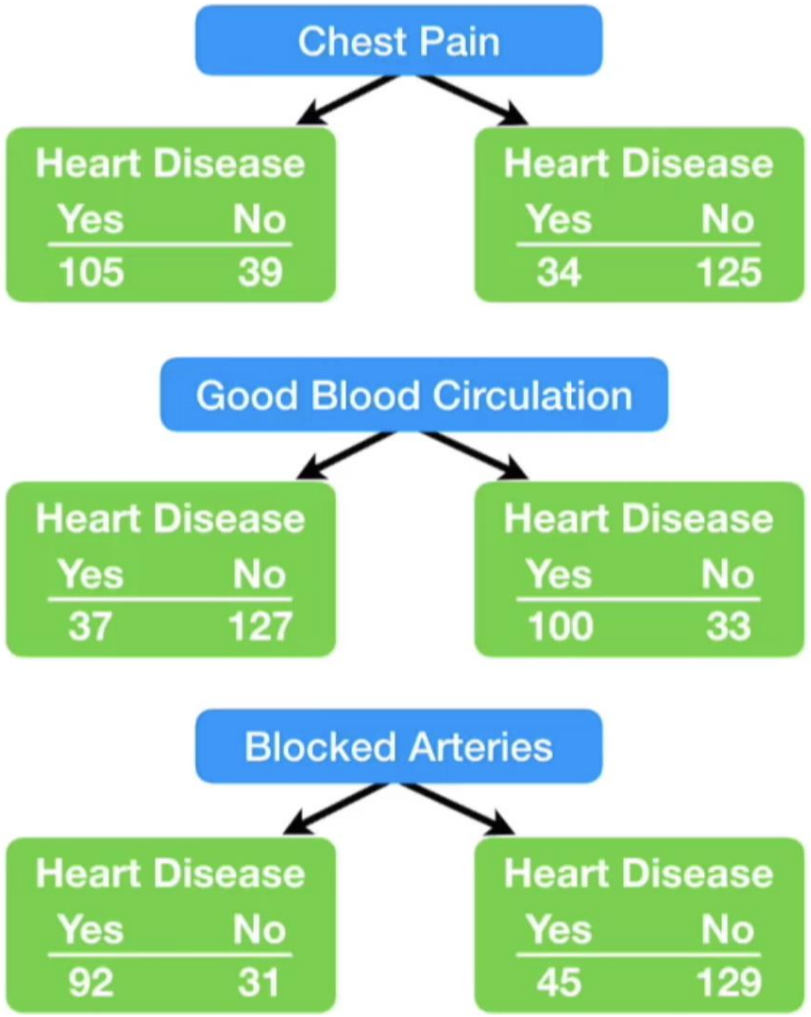
Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc…	etc…	etc…	etc…



Decision Tree – Root Node

Because none of the leaf nodes are 100% “YES Heart Disease” or 100% “NO Heart Disease”, they are all considered “**impure**”.

To determine which separation is best, we need a way to measure and compare “**impurity**”.



Decision Tree – Gini Impurity

- Then the Gini Impurity of the dataset D is defined as:

$$Gini(D) = 1 - \sum_{i=1}^{k} p_i^2$$

  - D: the data set
  - k: number of classes
  - p_i: The proportion of elements in the dataset D that belong to class i

- ## Gini Impurity Theorem

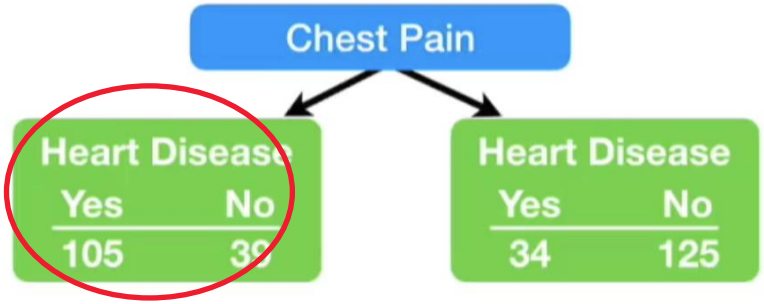
Given a random classifier that classifies a data point according to the class distribution, ie

$$P(\text{classify data point } x \text{ as class } i \mid \text{dataset } D) = p_i$$

Prove that the probability of misclassification equals the Gini Impurity $Gini(D)$.

Decision Tree – Root Node

$$Gini(D) = 1 - \sum_{i=1}^{k} p_i^2$$



Gini impurity = 0.395

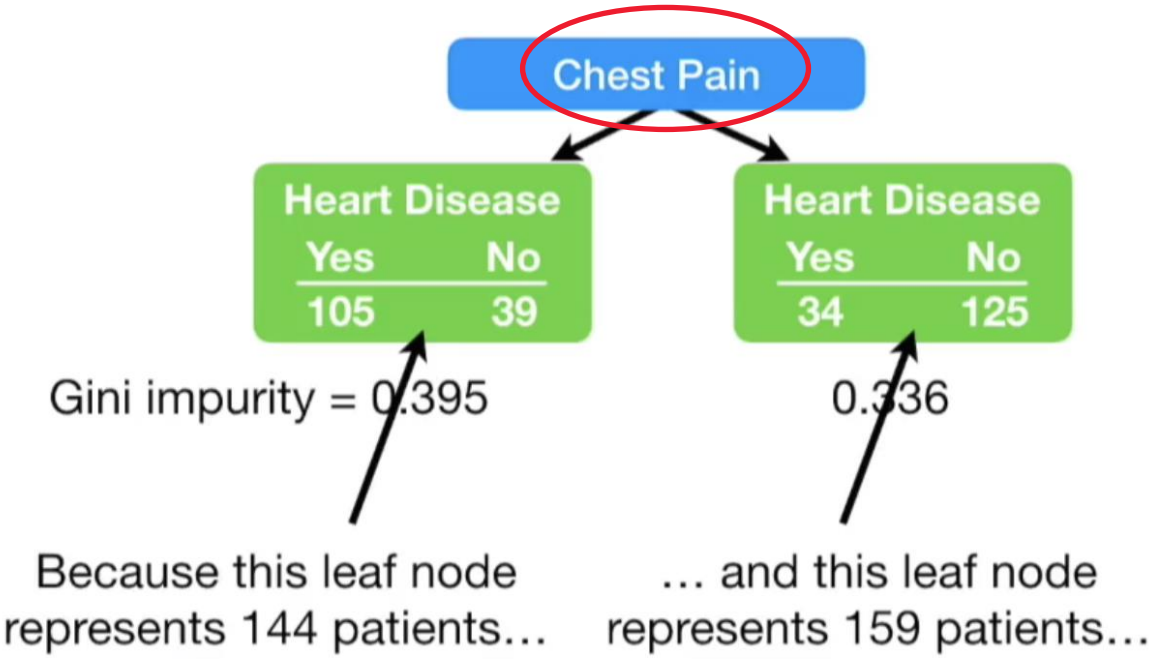
0.336

For this leaf, the Gini impurity = 1 - (the probability of "yes")^2 - (the probability of "no")^2

$$= 1 - \left(\frac{105}{105 + 39}\right)^2 - \left(\frac{39}{105 + 39}\right)^2$$

$$= 0.395$$

Decision Tree – Root Node



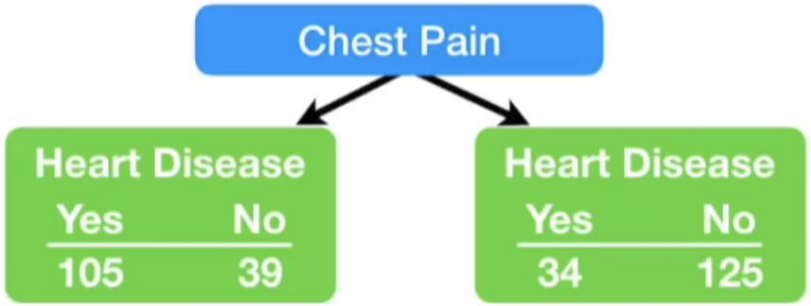
Gini impurity for Chest Pain = weighted average of Gini impurities for the leaf nodes

$$= \left(\frac{144}{144 + 159}\right) 0.395 + \left(\frac{159}{144 + 159}\right) 0.336$$

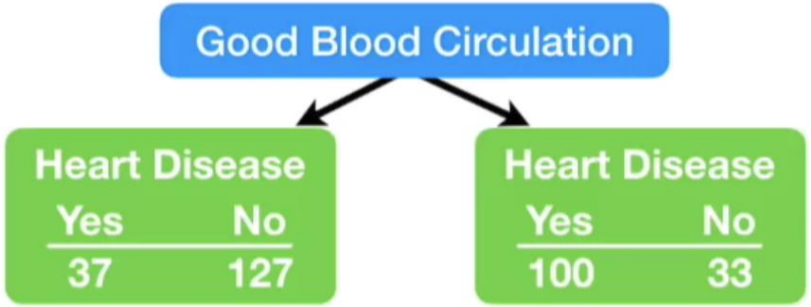
$$= 0.364$$

Decision Tree – Root Node

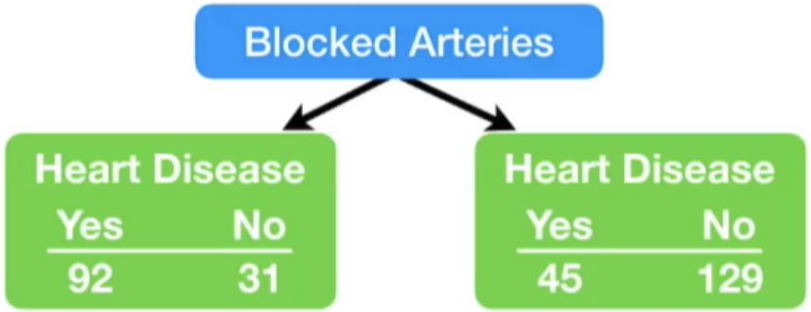
Gini impurity for Chest Pain = 0.364



Gini impurity for Good Blood Circulation = 0.360



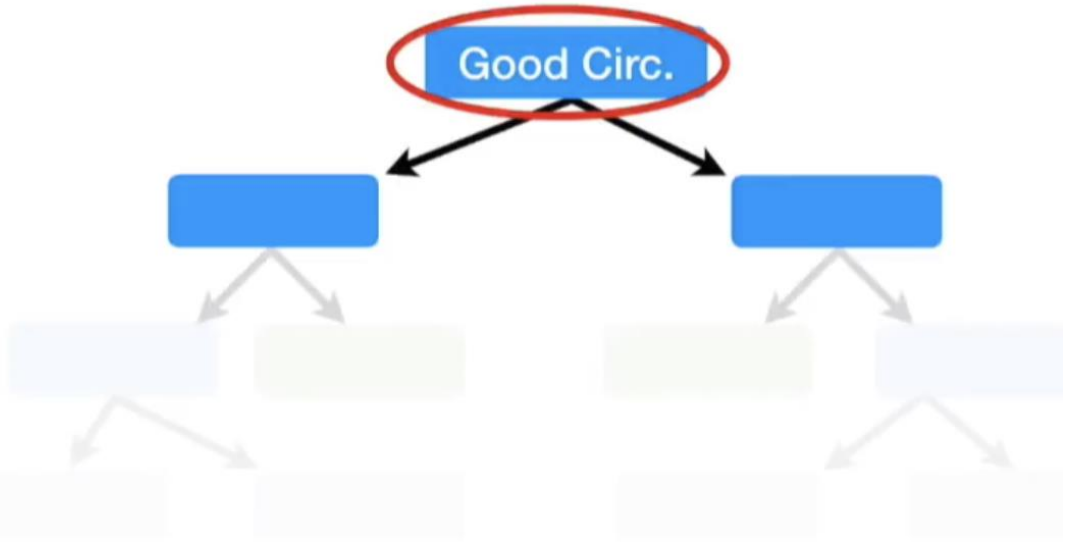
Gini impurity for Blocked Arteries = 0.381



Decision Tree – Root Node

Gini impurity for Chest Pain = 0.364

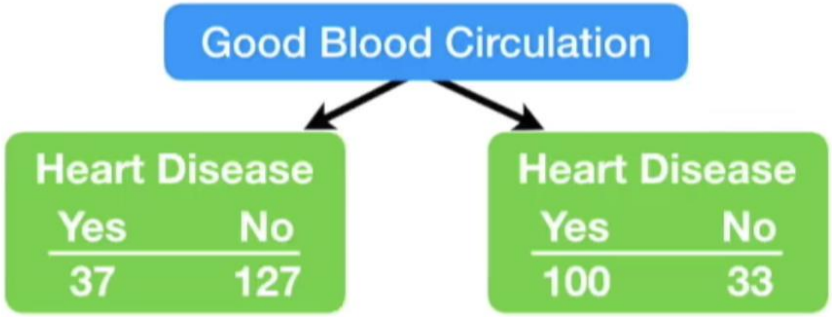
…so we will use it at the root of the tree.



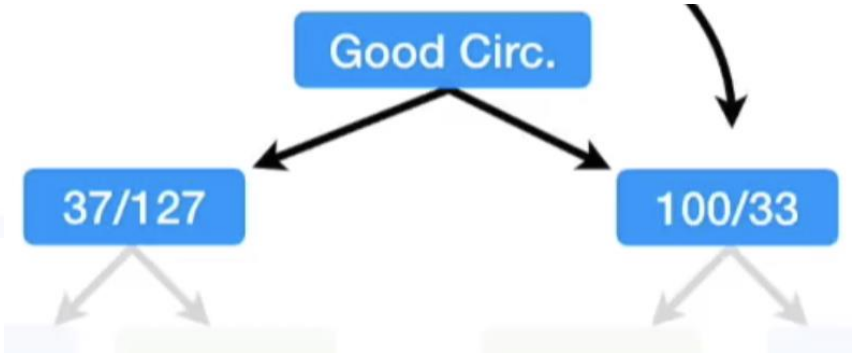
Gini impurity for Good Blood Circulation = 0.360

Gini impurity for Blocked Arteries = 0.381

Decision Tree – Root Node

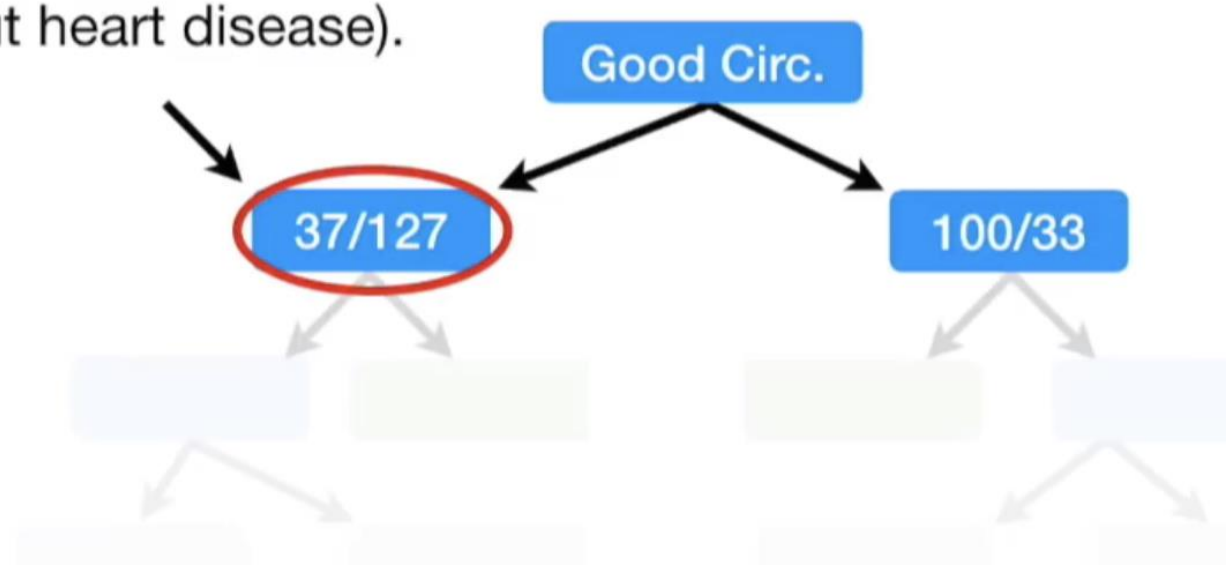


When we divided all of the patients using **Good Blood Circulation**, we ended up with "impure" leaf nodes.

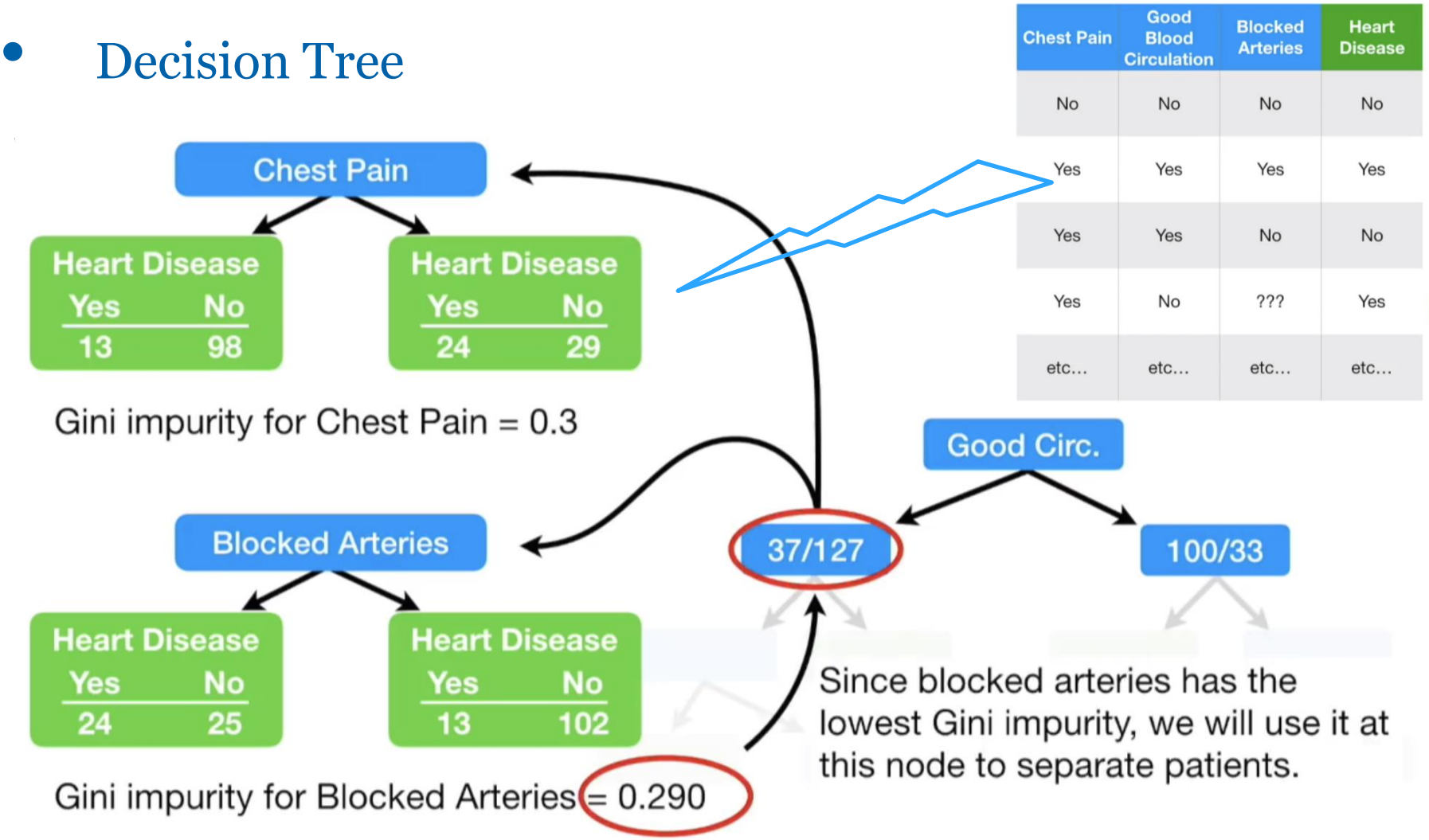


Decision Tree

Now we need to figure how well **chest pain** and **blocked arteries** separate these 164 patients (37 with heart disease and 127 without heart disease).

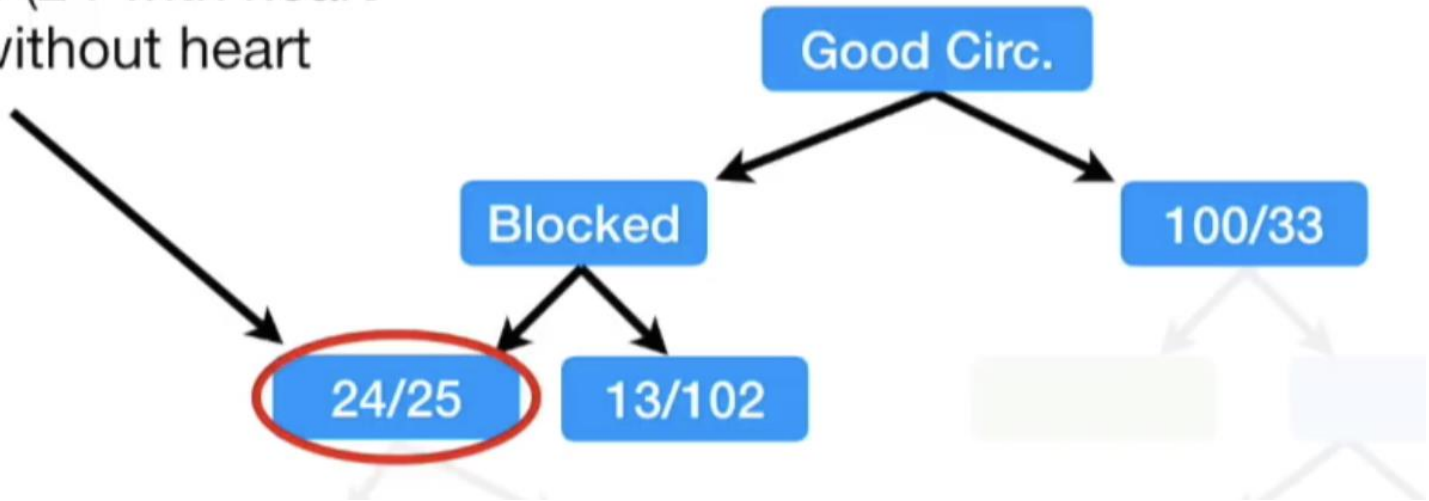


Decision Tree

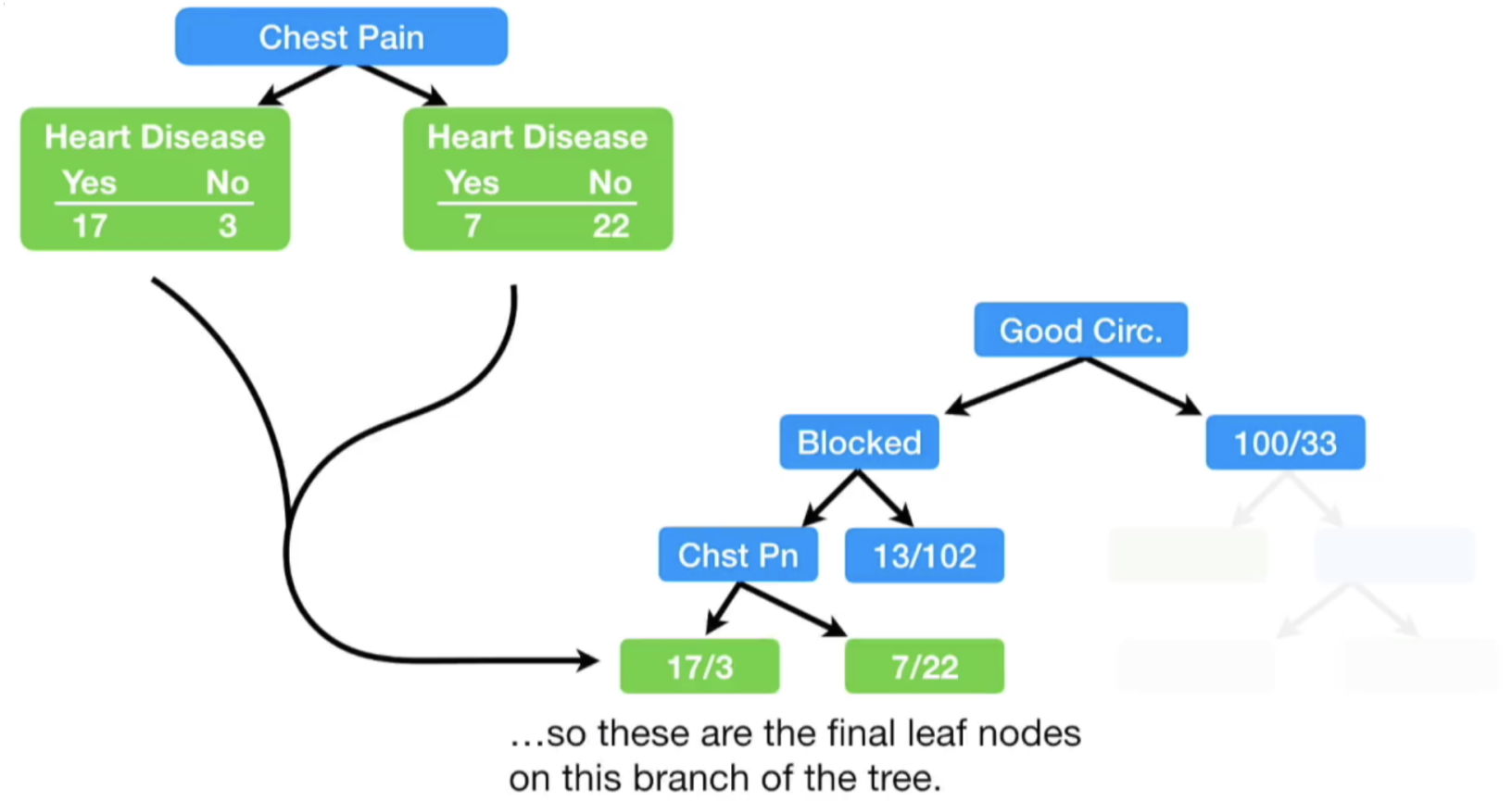


- # Decision Tree

All we have left is Chest Pain, so first we'll see how well it separates these 49 patients (24 with heart disease and 25 without heart disease).

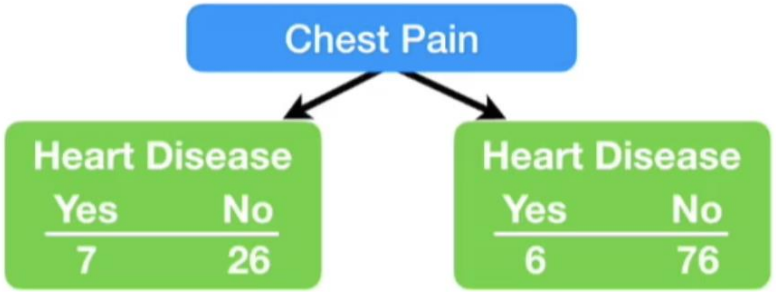


Decision Tree



Decision Tree

Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes
Yes	Yes	No	No
Yes	No	???	Yes
etc…	etc…	etc…	etc…



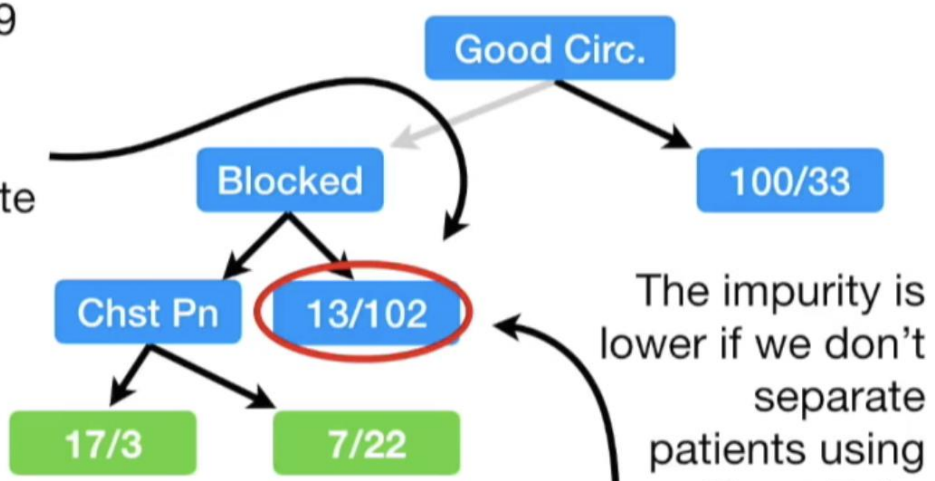
Gini impurity for Chest Pain = 0.29

The Gini impurity for this node, before using chest pain to separate patients is…

= 1 - (the probability of "yes")^2
- (the probability of "no")^2

$$= 1 - \left(\frac{13}{13 + 102}\right)^2 - \left(\frac{102}{13 + 102}\right)^2$$

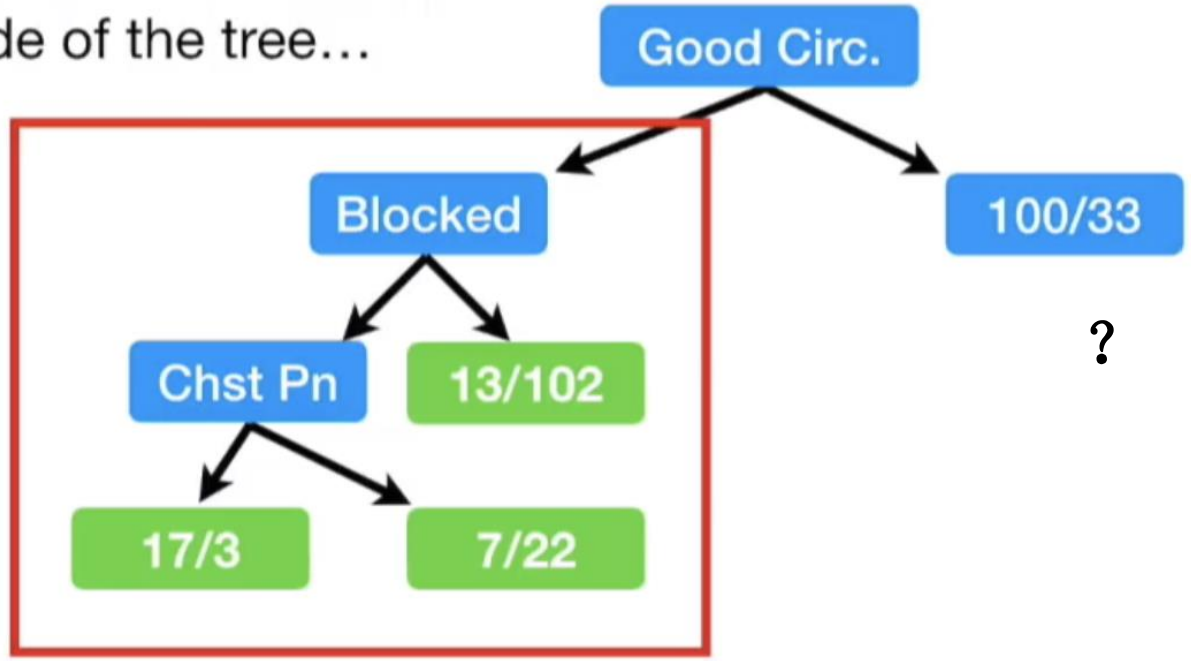
= 0.2



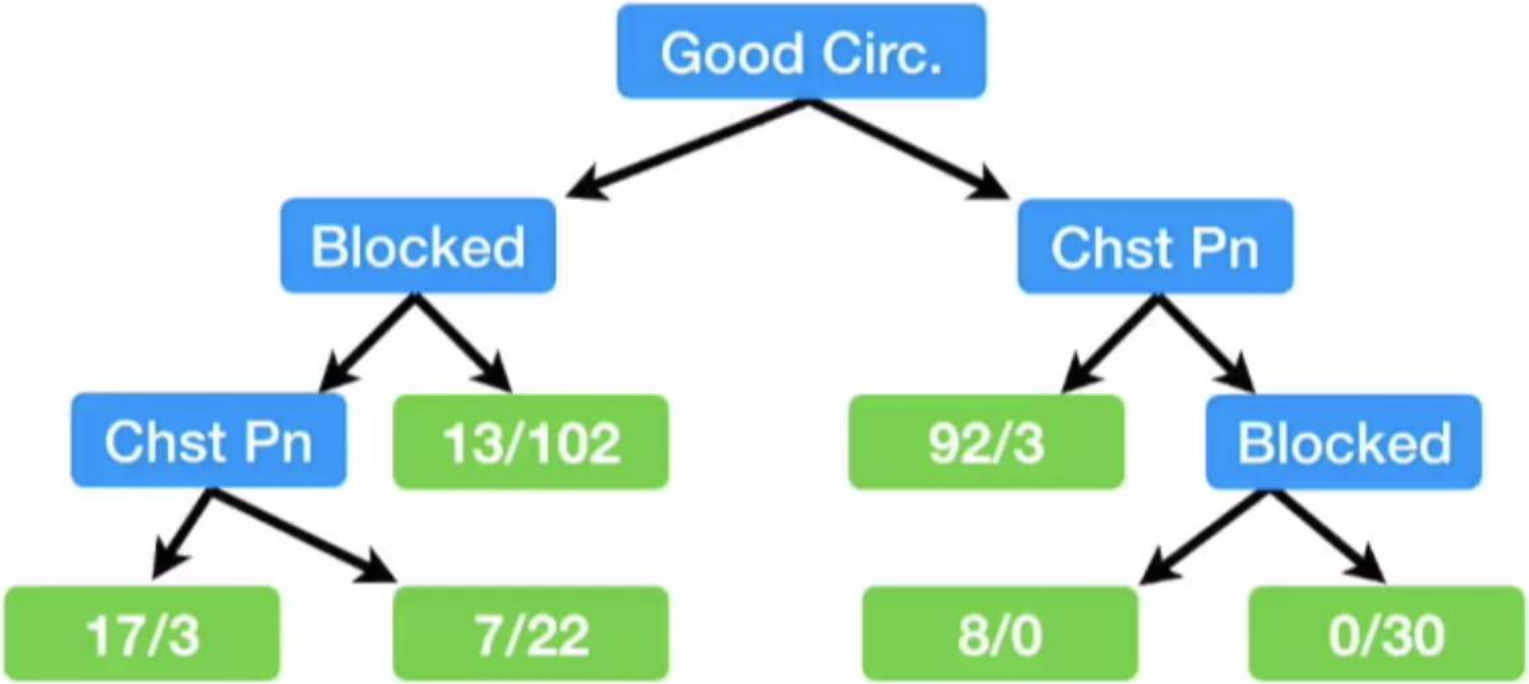
The impurity is lower if we don't separate patients using Chest Pain.

Decision Tree

OK, at this point we've worked out the entire left side of the tree…



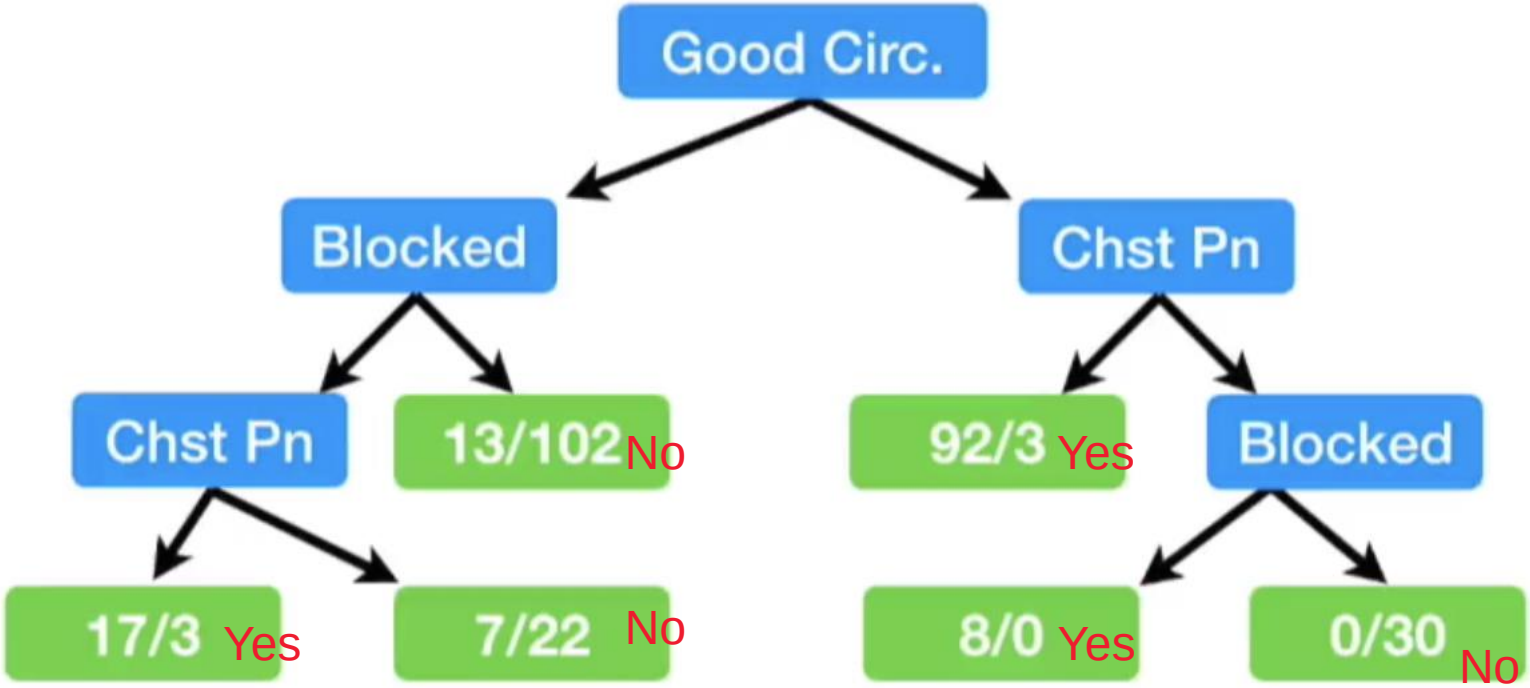
- Decision Tree



- Decision Tree

prediction

Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes



- Decision Tree

Weight	Heart Disease
220	Yes
180	Yes
225	Yes
190	No
155	No

Chest Pain	Good Blood Circulation	Blocked Arteries	Heart Disease
No	No	No	No
Yes	Yes	Yes	Yes

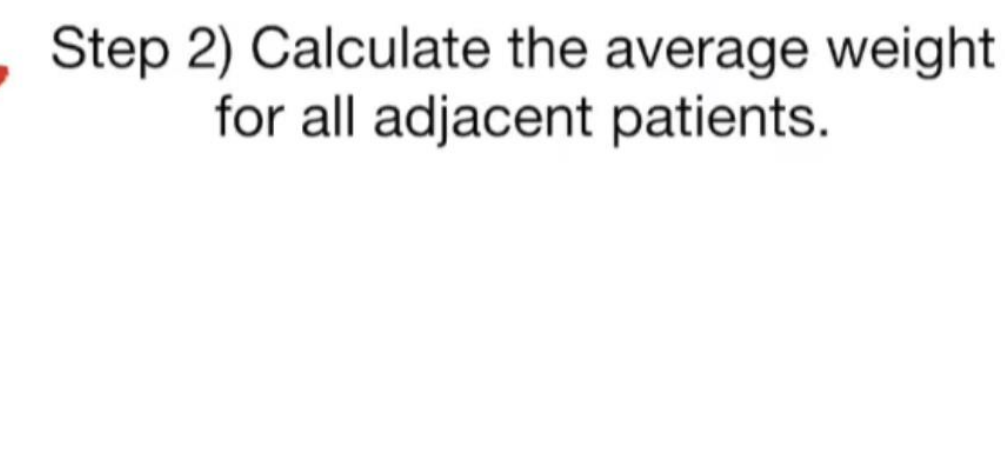
How do we determine what's the best weight to use to divide the patients?

Decision Tree – Root Node

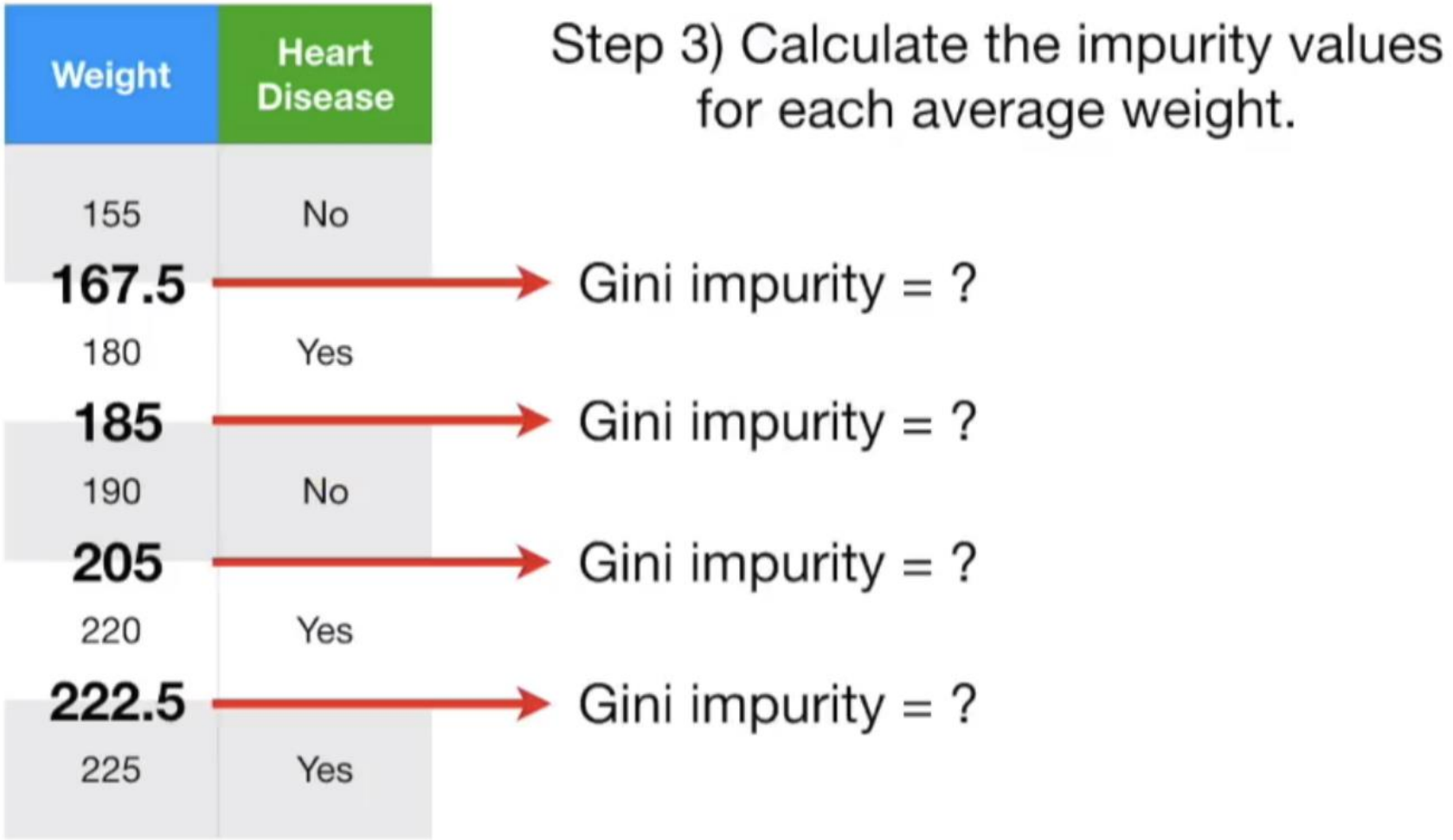
Step 1) Sort the patients by weight, lowest to highest.

Decision Tree – Root Node

Weight	Heart Disease
155	No
167.5	
180	Yes
185	
190	No
205	
220	Yes
222.5	
225	Yes

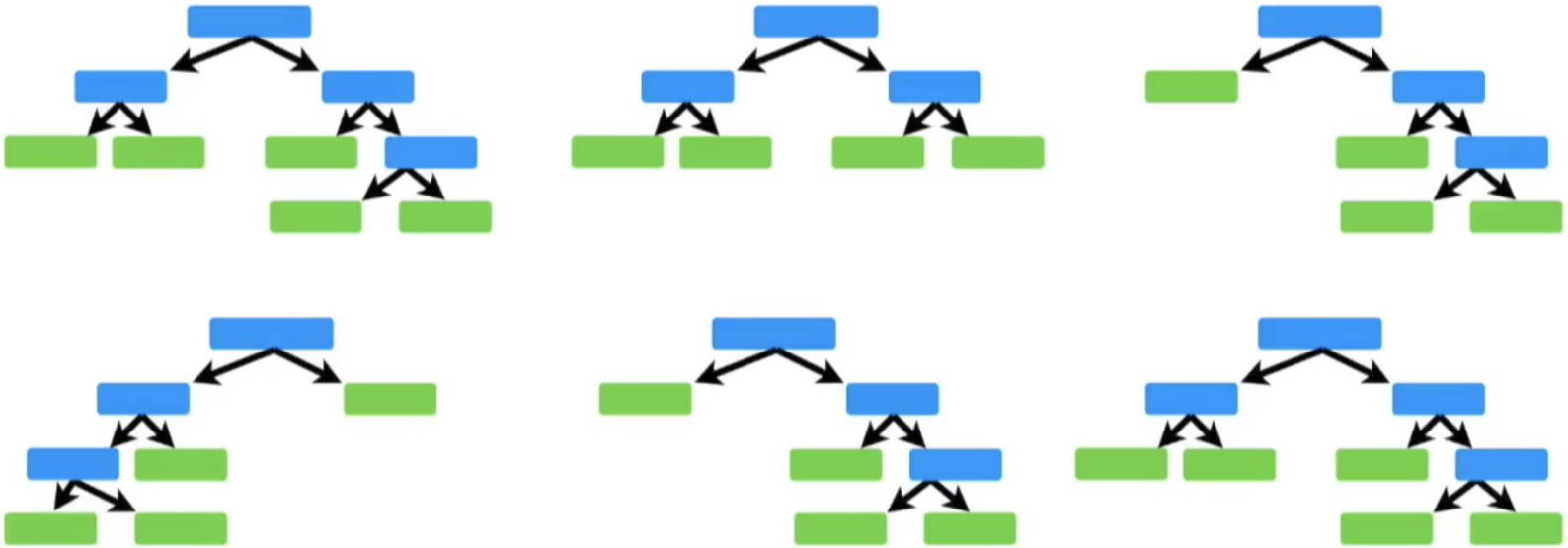


Decision Tree – Root Node



Random Forest

Random Forest



Random Forest

1. Build **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

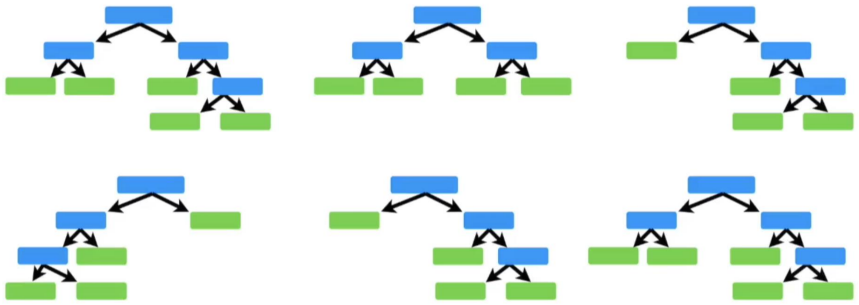
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease

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.

- # Random Forest

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

- # Random Forest

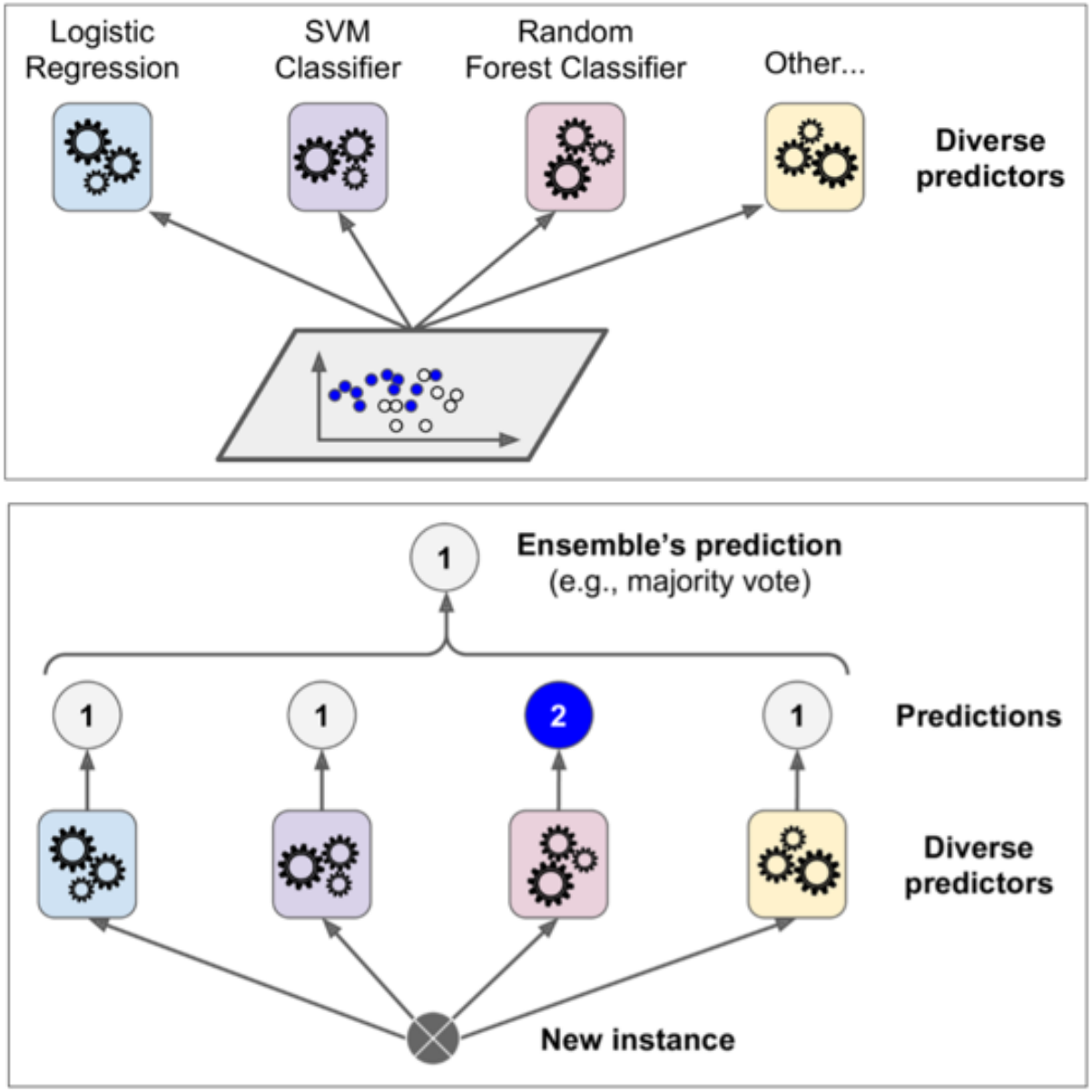
Testing

Vote by each tree!

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.

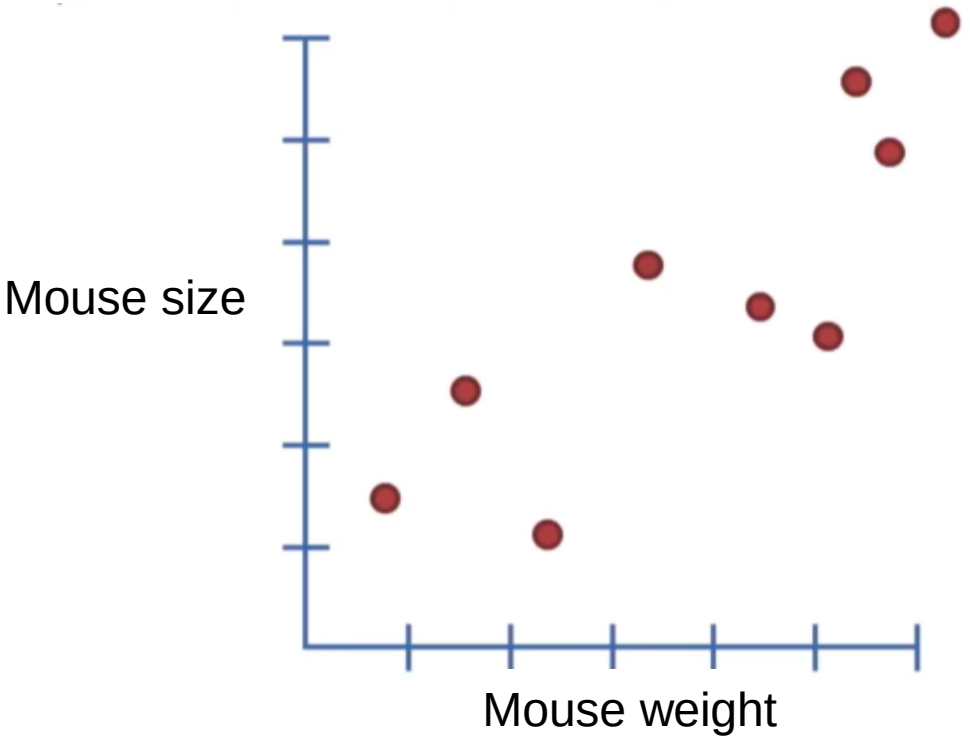
Ensemble Learning



Regression

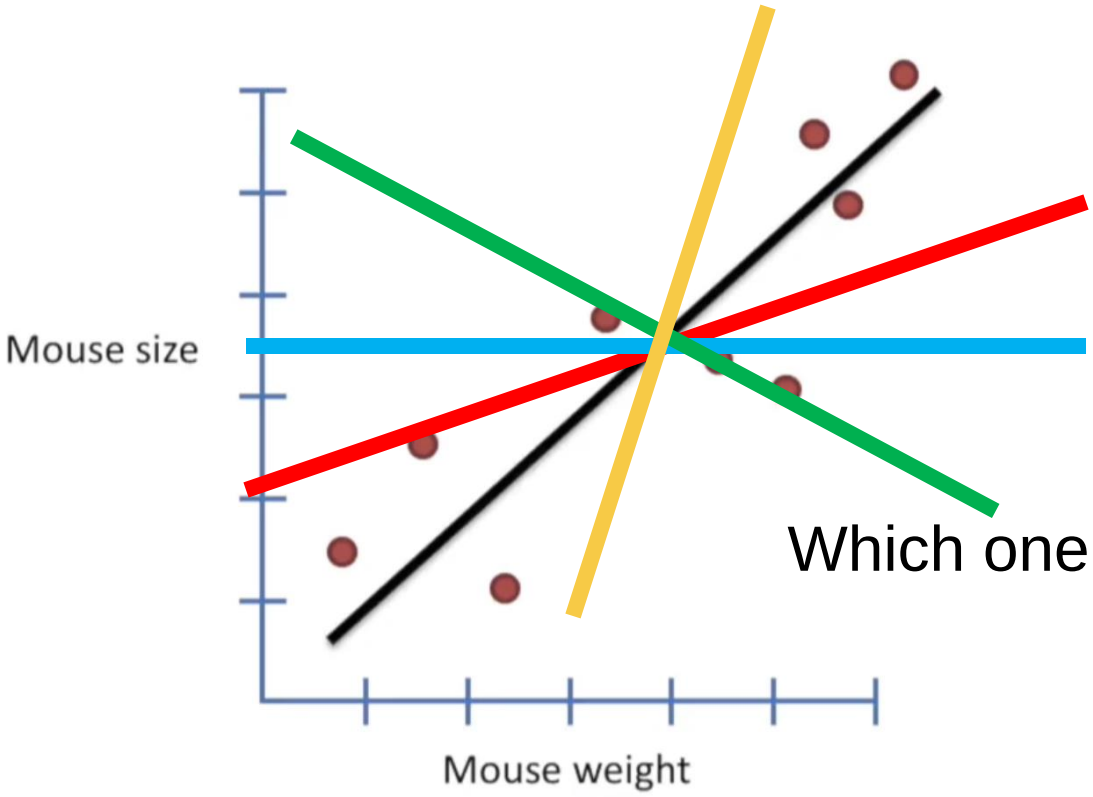
- # Linear Regression

Goal: Fit a line to the data



- **Linear Regression**

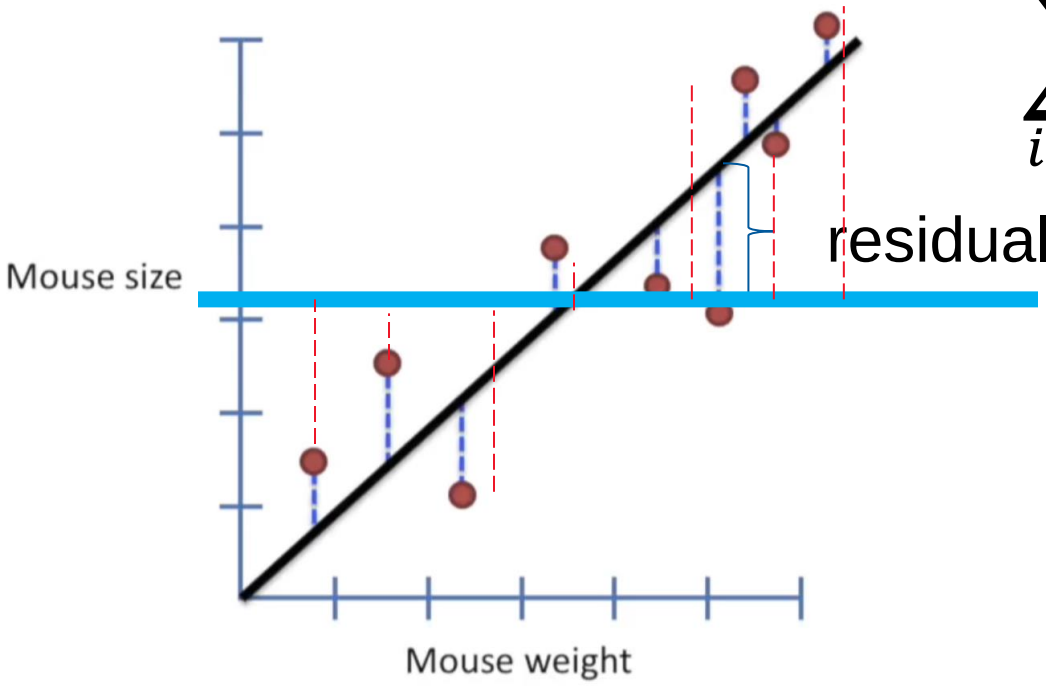
Fit a line to the data



Which one is the best?

- **Linear Regression**

Find the best fit which has the **least square error**

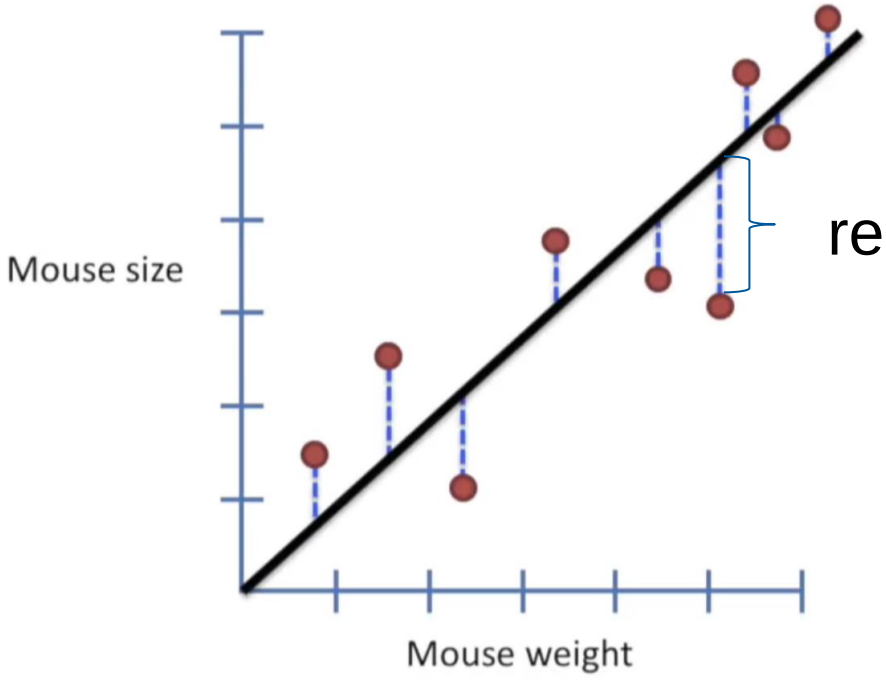


$$\sum_{i=1}^{N}(y-\hat{y})^2$$

residual (error)

- **Linear Regression**

Alternative:
Find the best fit which has the **least absolute difference**

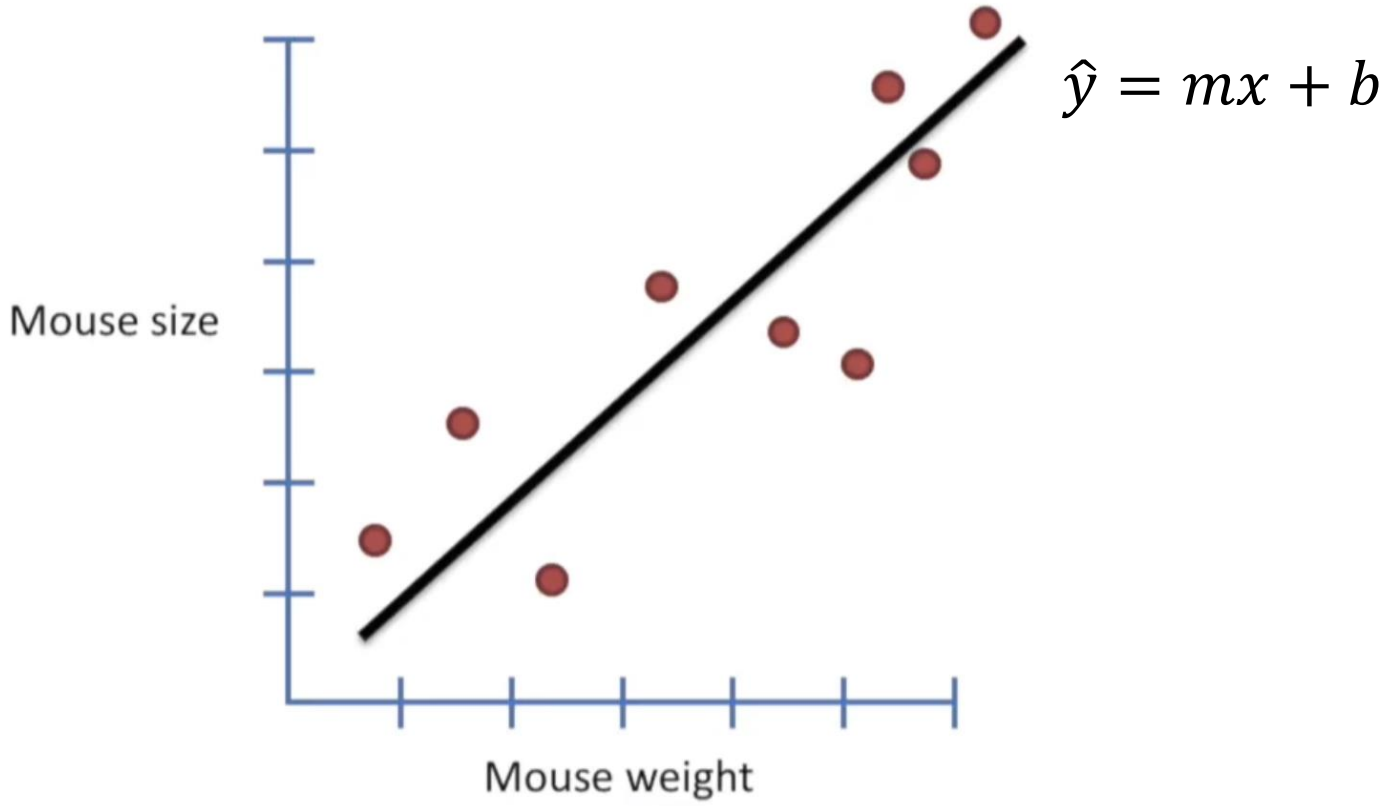


$$\sum_{i=1}^{N} |y - \hat{y}_i|$$

residual (error)

- **Linear Regression**

Use least-squares to fit a line to the data



- **Regression**

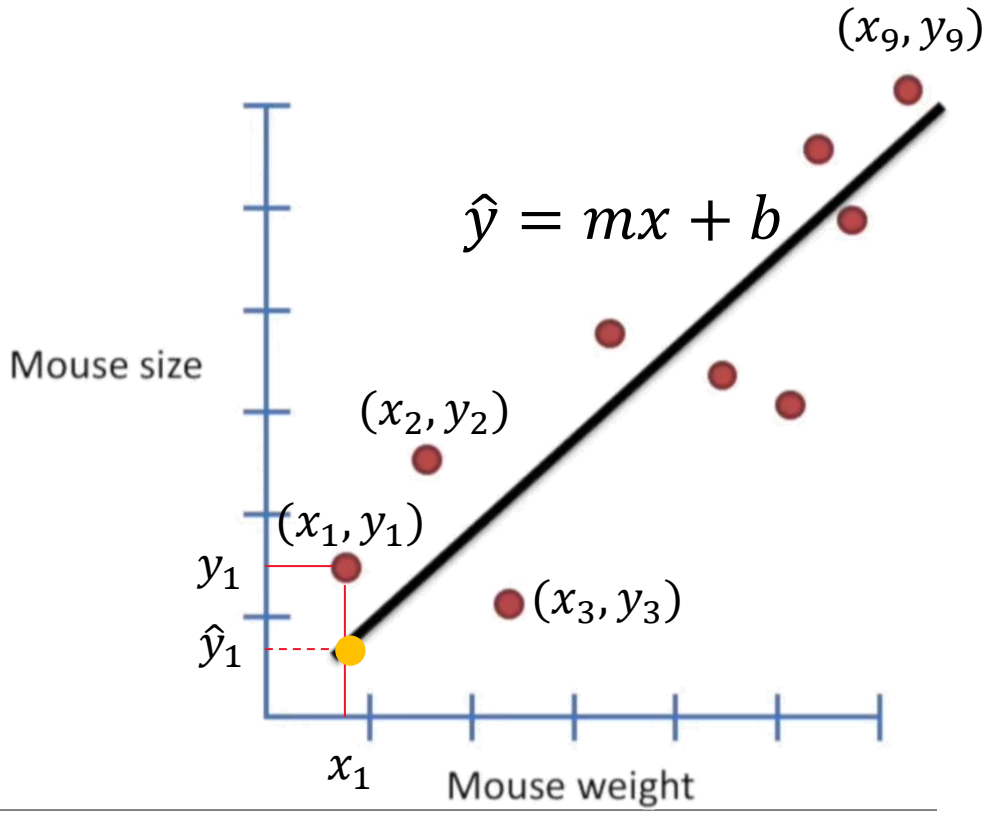
Goal $arg \min_{m,b} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$ ← L2 loss

where $\hat{y}_i = mx_i + b$

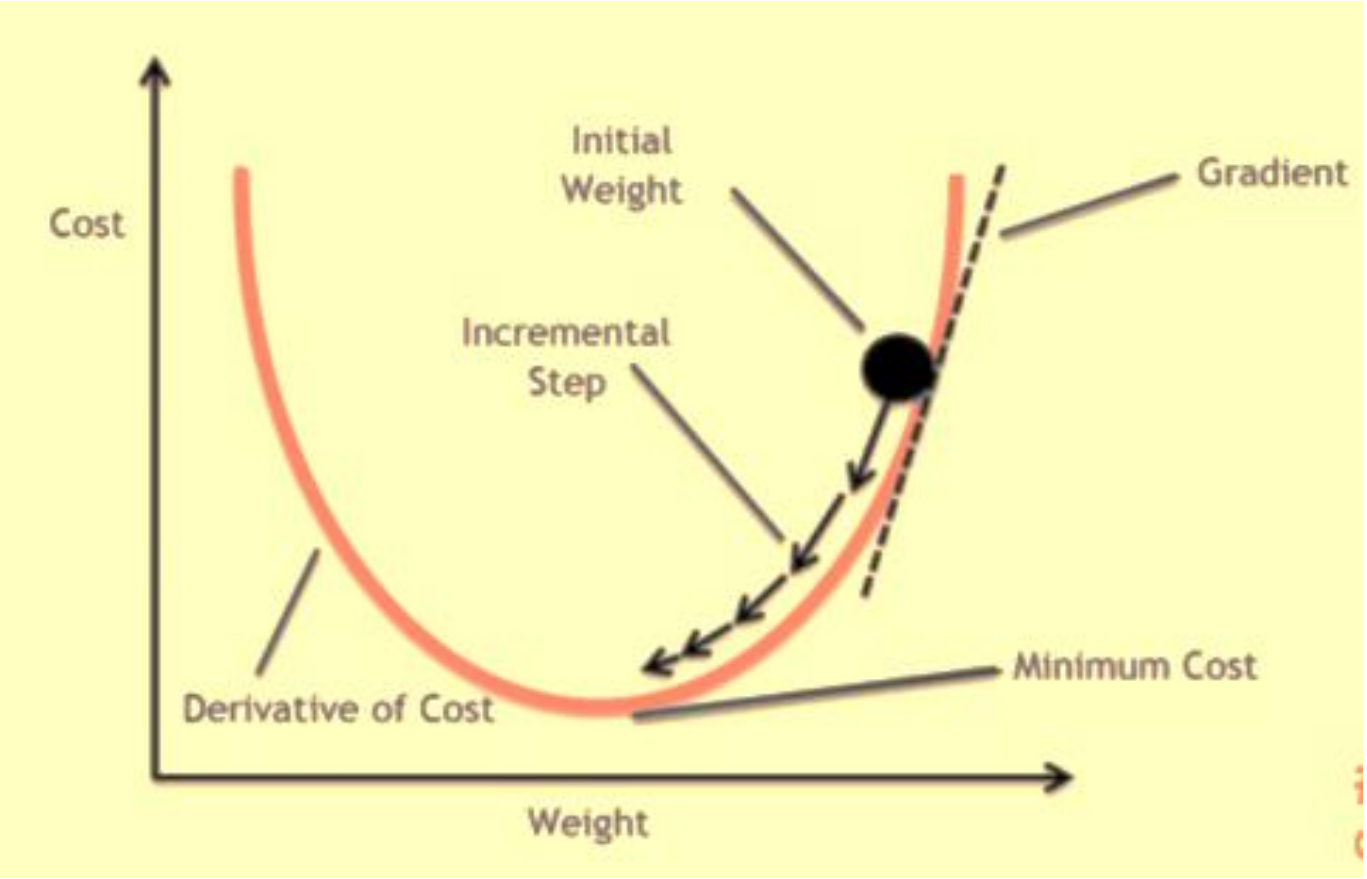
$(x, y) \in \{(x_1, y_1), \dots (x_n, y_n)\}$

$arg \min_{m,b} \sum_{i=1}^{N} |y_i - \hat{y}_i|$

L1 loss



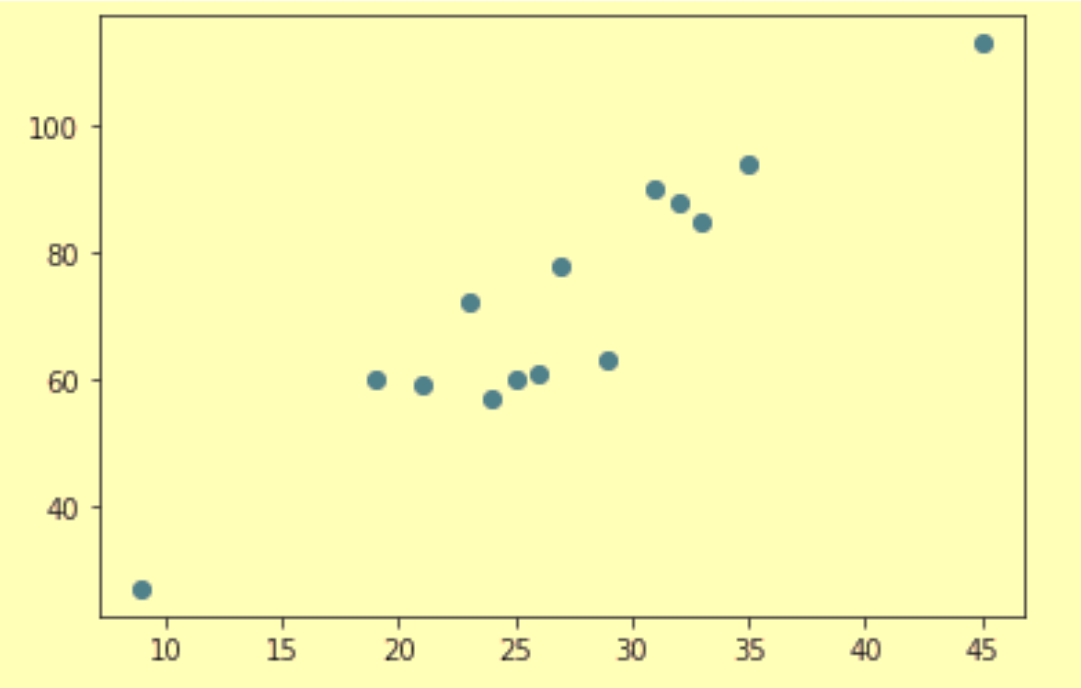
Optimization: Gradient Descent



Programming Example

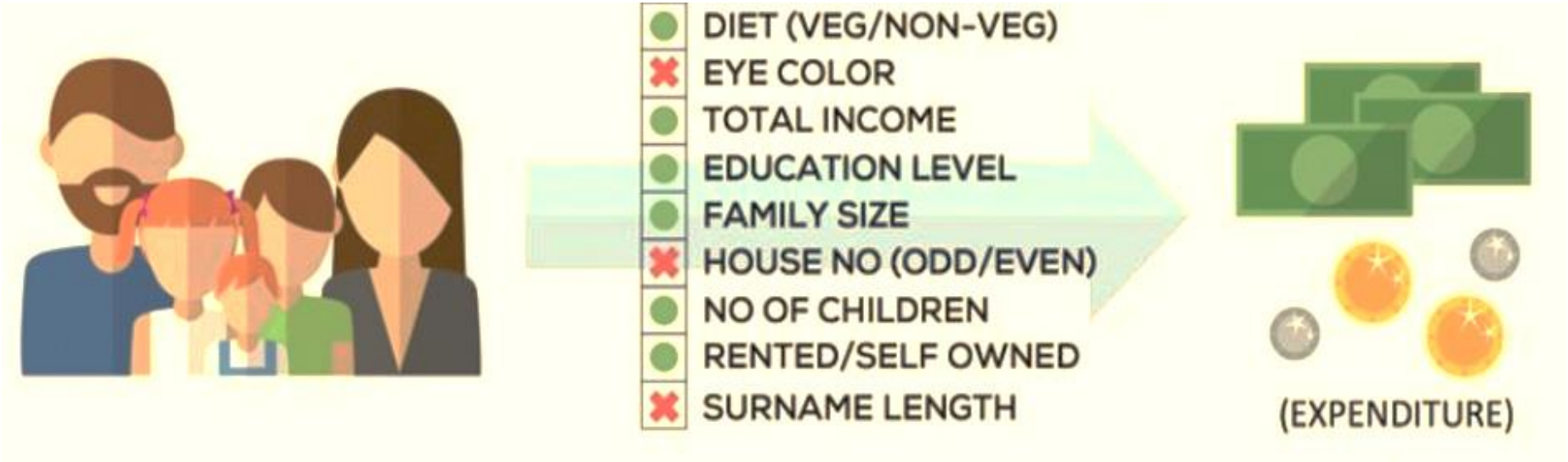
Regression

id	income	expenditure
1	19	60
2	45	113
3	35	94
4	31	90
5	25	60
6	32	88
7	21	59
8	26	61
9	24	57
10	27	78
11	9	27
12	23	72
13	33	85
14	29	63



Regression

Multiple variable regression



$$\hat{y}_i = b + w_1 x_{i1} + w_2 x_{i2} + \cdots + w_p x_{ip}$$

$X_1 = (x_{11}, x_{12}, \ldots, x_{1p})$

$X_N = (x_{N1}, x_{N2}, \ldots, x_{Np})$

Diet	Income	Edu	Size	Children	Rent/loan	Expense
1	4600	12	3	1	2000	800
0	6000	15	5	2	1500	1500

Clustering

Training data

Test data

Image:

K-Means

Imagine we have some 1D data like this, it can be weight, size, etc.

Task: divide them into different groups

How many groups are there?

K-Means

But, rather than rely on our eye, let's see if we can get a computer to identify the same 3 clusters.

To do this, we'll use K-means clustering.

K-Means --- process

1. Select the number of clusters you want to identify in your data, namely “k” in k-means

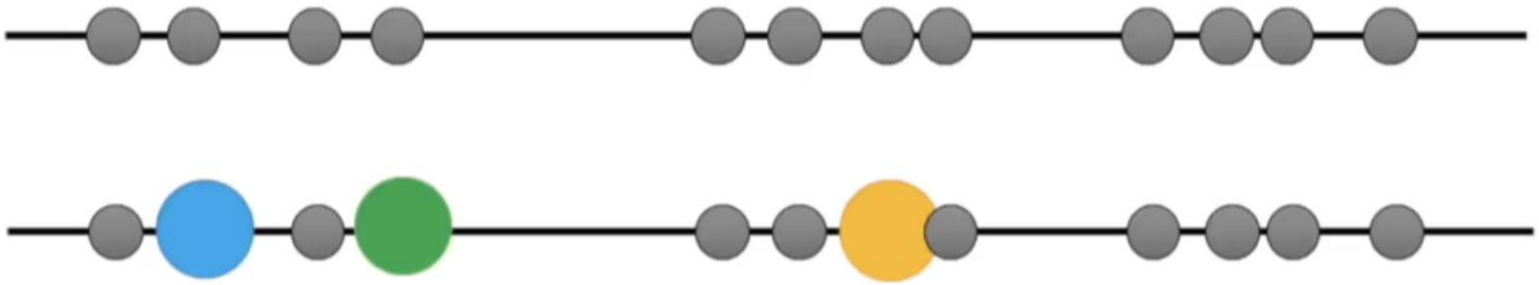
K-Means --- process

1. Select the number of clusters you want to identify in your data, namely “k” in k-means

$$k = 3$$

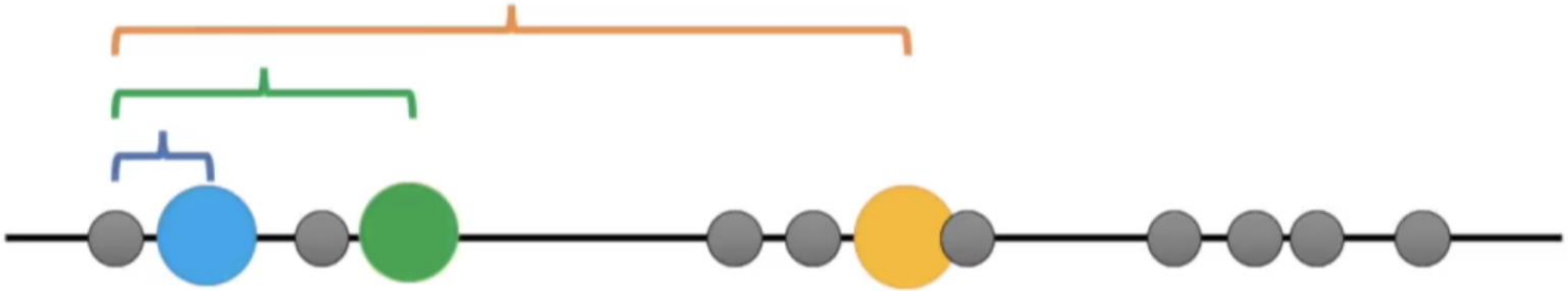
K-Means --- process

2. Randomly select 3 distinct data points



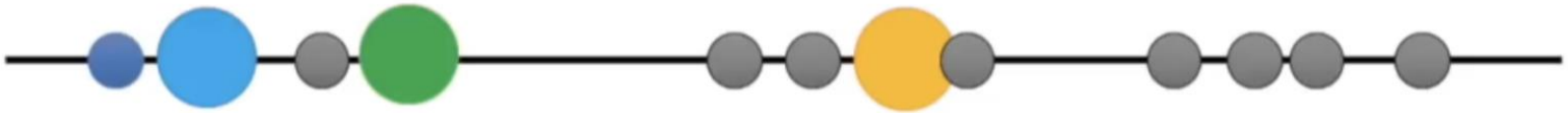
K-Means --- process

3. Measure the distance of the 1st data to all cluster centers



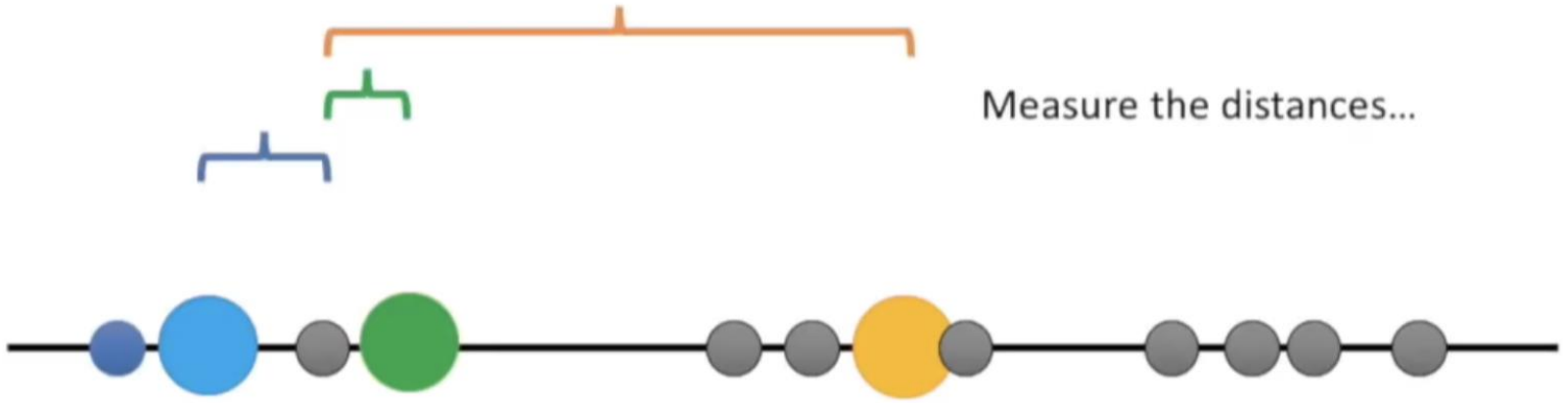
K-Means --- process

4. Assign it to the nearest cluster



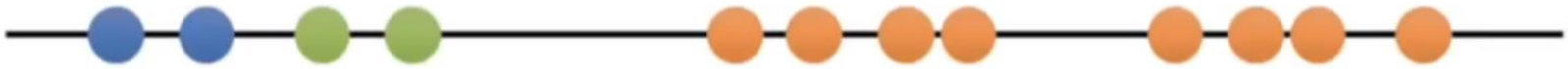
K-Means --- process

5. Do the same thing to the remaining points



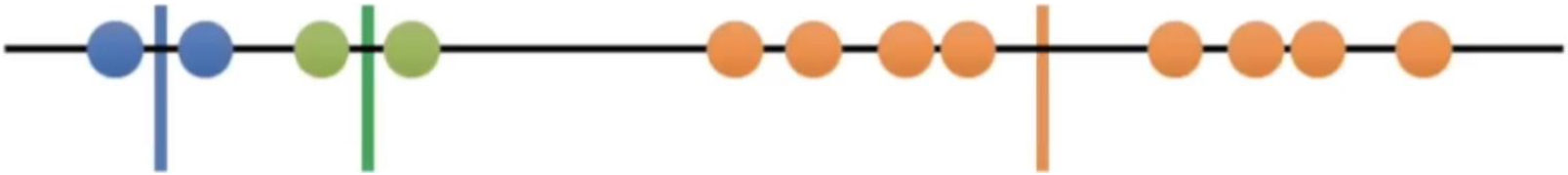
K-Means --- process

6. After loop all data points …



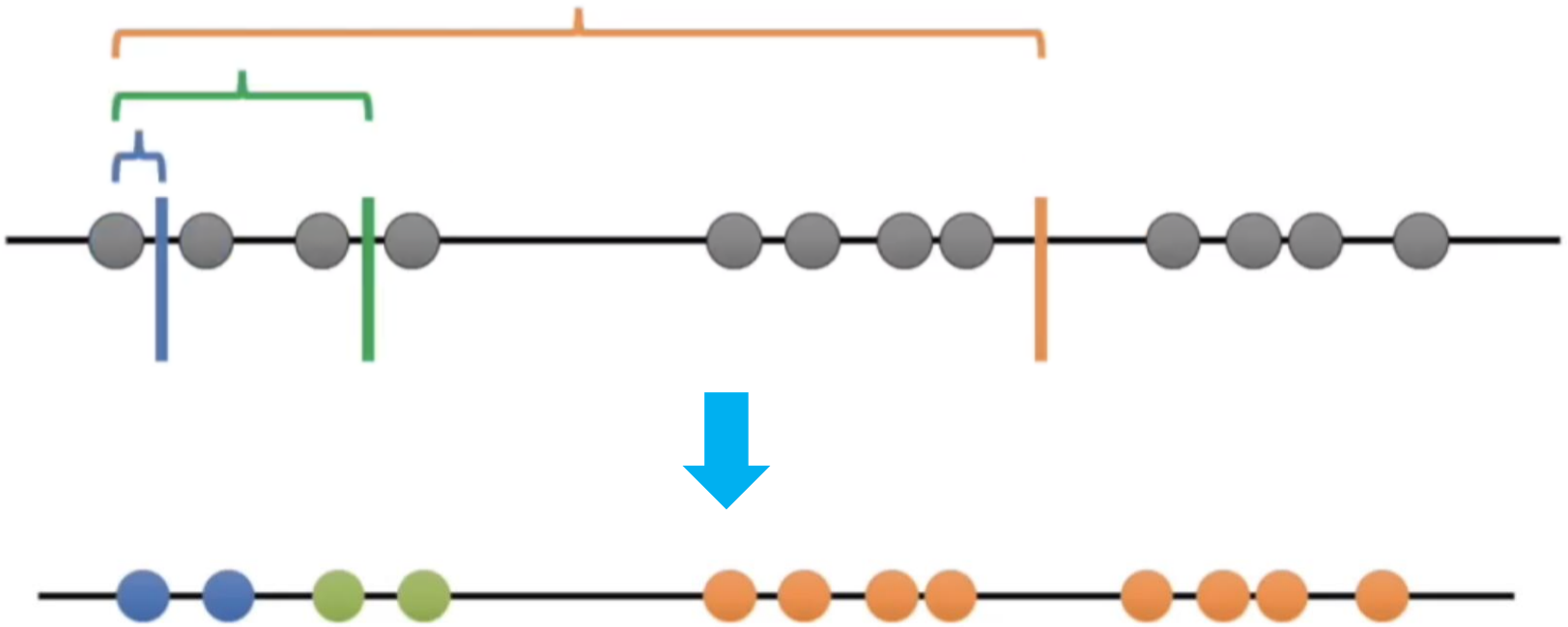
K-Means --- process

7. Calculate the mean of each cluster, getting the new cluster centers



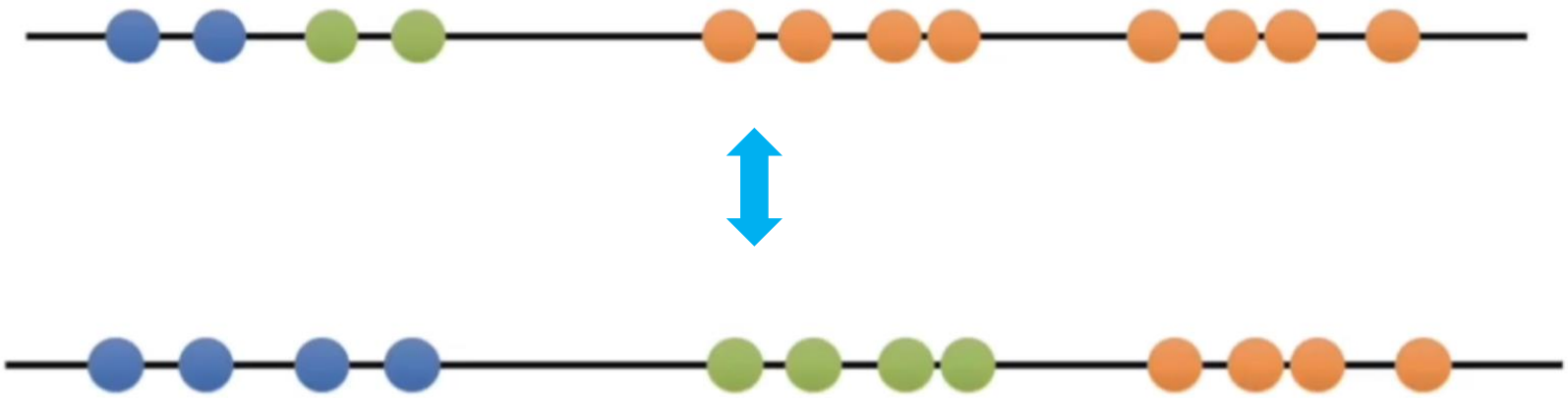
K-Means --- process

8. Repeat the above steps, until cluster centers do not change.



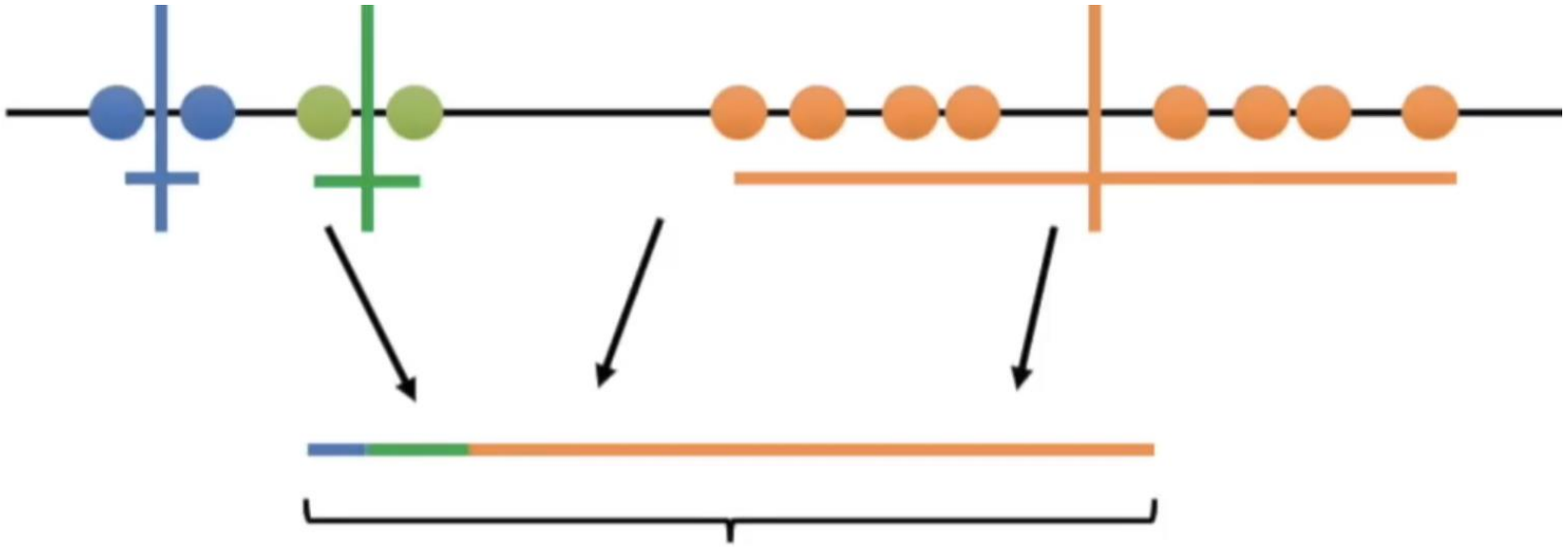
K-Means

It is pretty terrible compared to we did by eye



K-Means

Measure variation

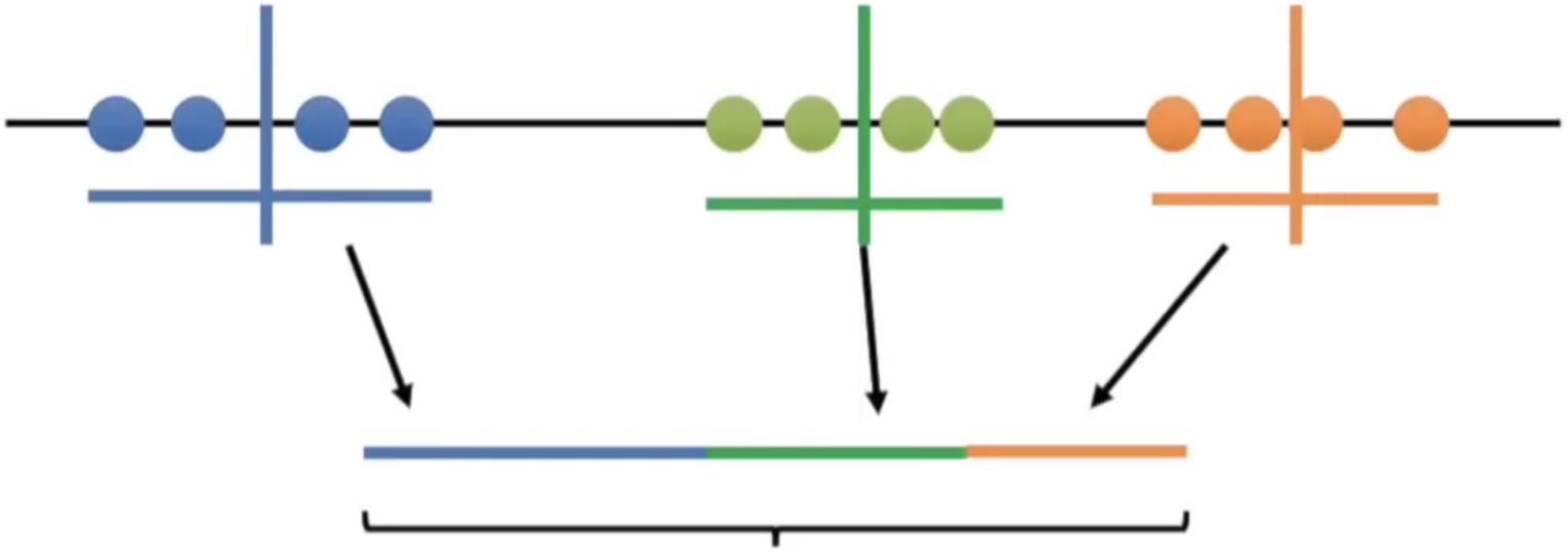


total variation within the clusters

K-Means

Redo k-means, starting from different initial clusters

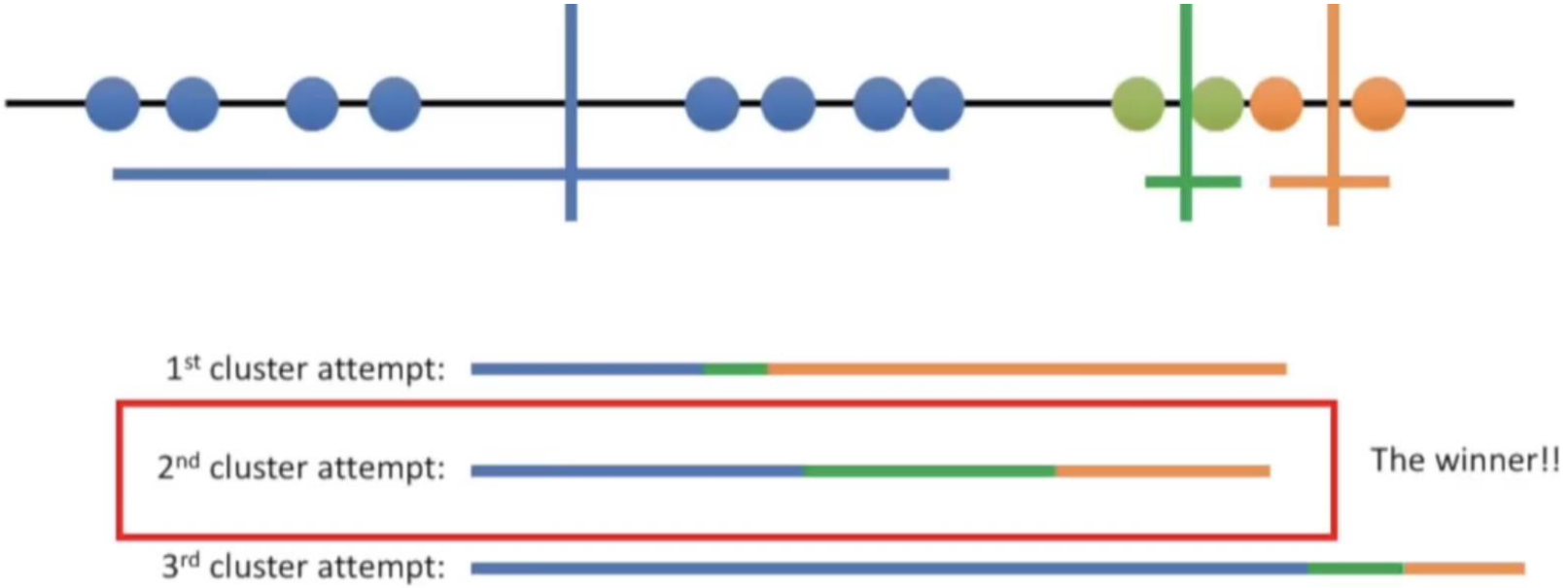
K-Means



total variation within the clusters

K-Means

Compare variations

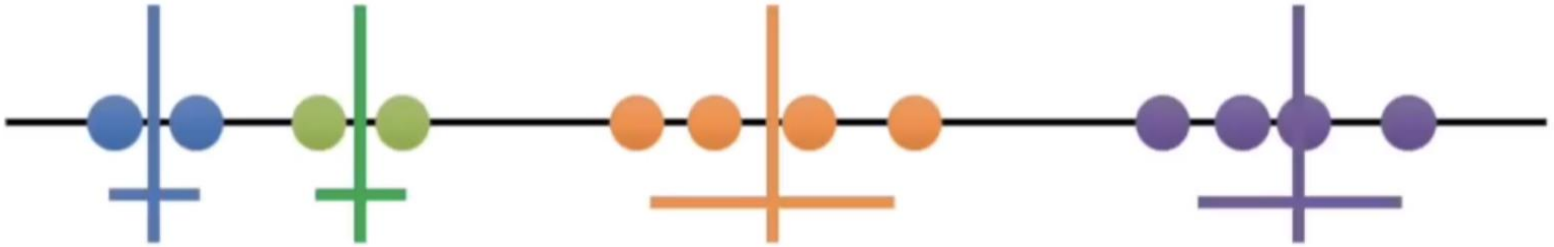


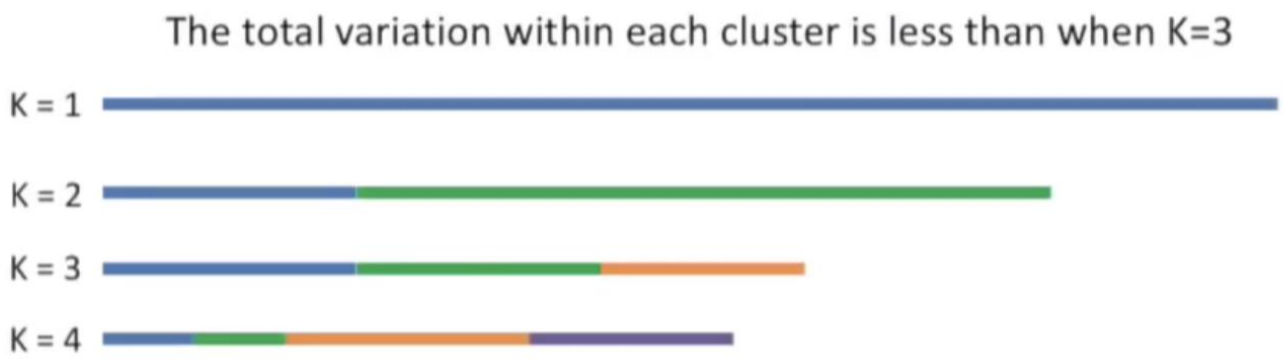
K-Means

How to figure out what value to use for “K”?

Just try different values for K

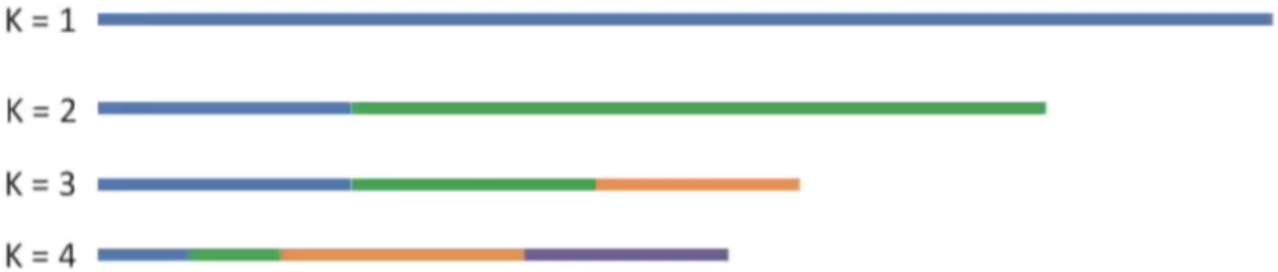
K-Means

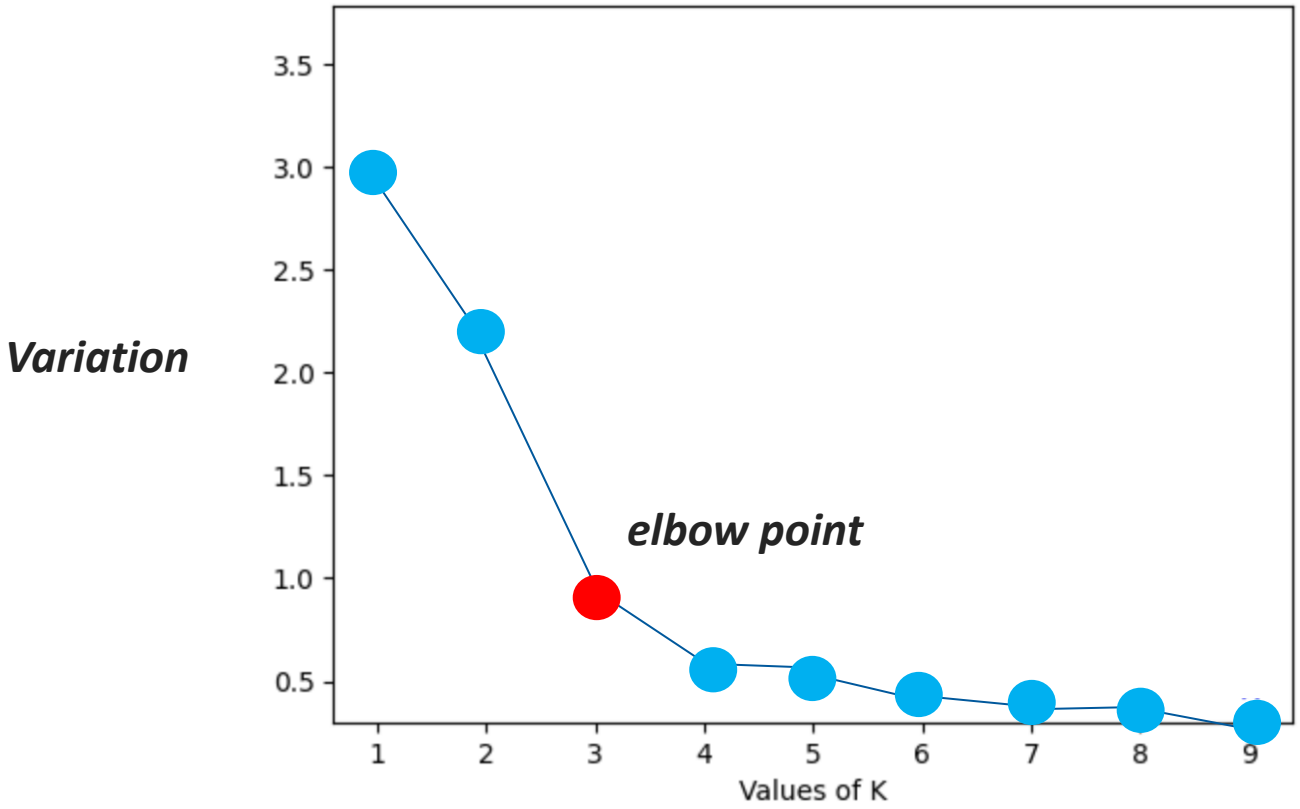




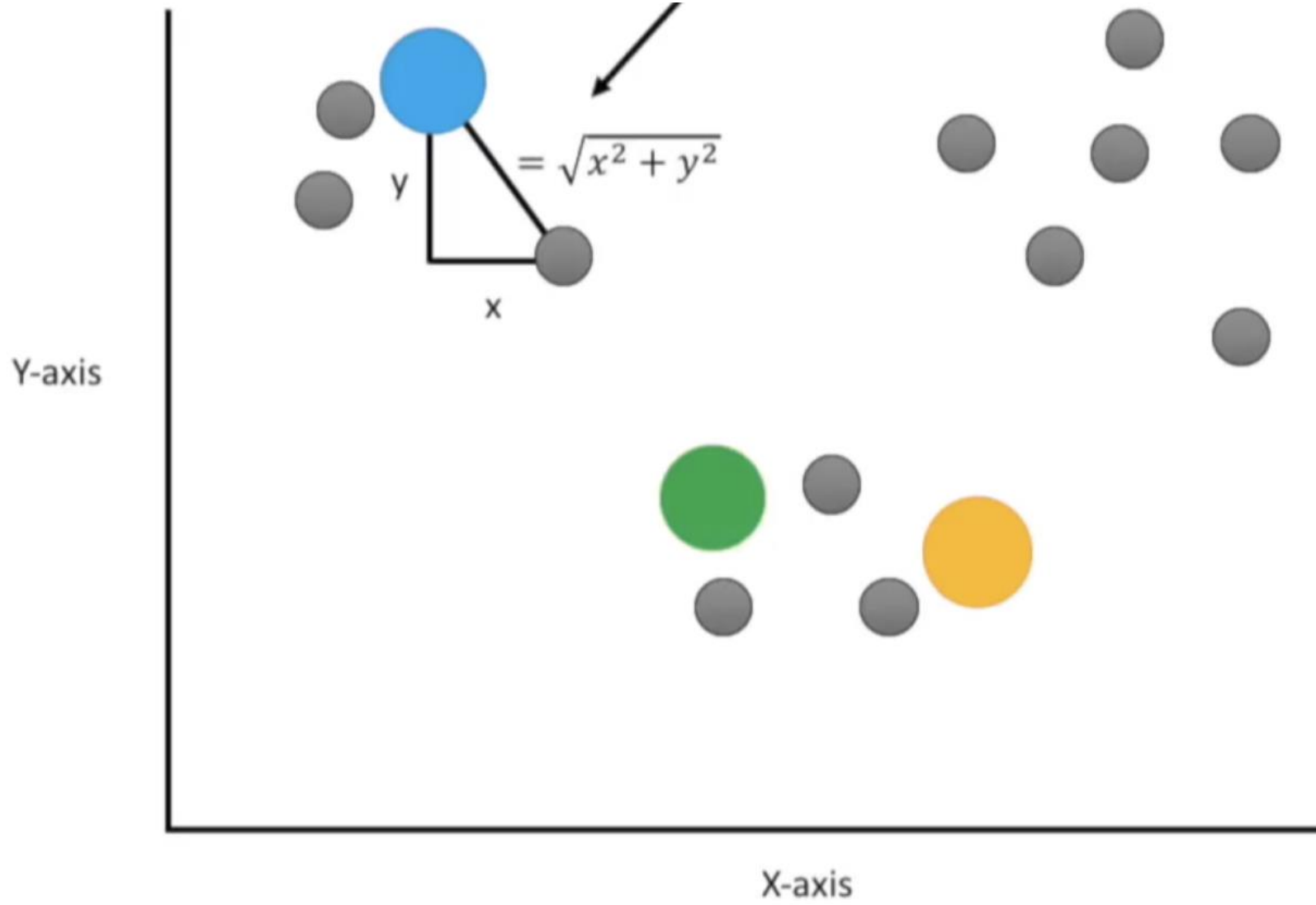
When we increase K, the variations become smaller

K-

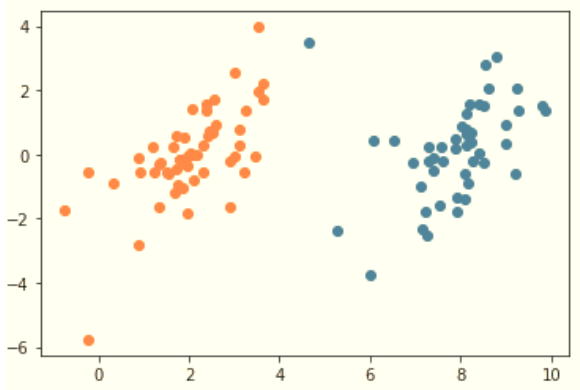

K = 1
K = 2
K = 3
K = 4


Variation
elbow point
3.5
3.0
2.5
2.0
1.5
1.0
0.5
1
2
3
4
5
6
7
8
9
Values of K

K-Means – 2D



K-Means --- Implementation

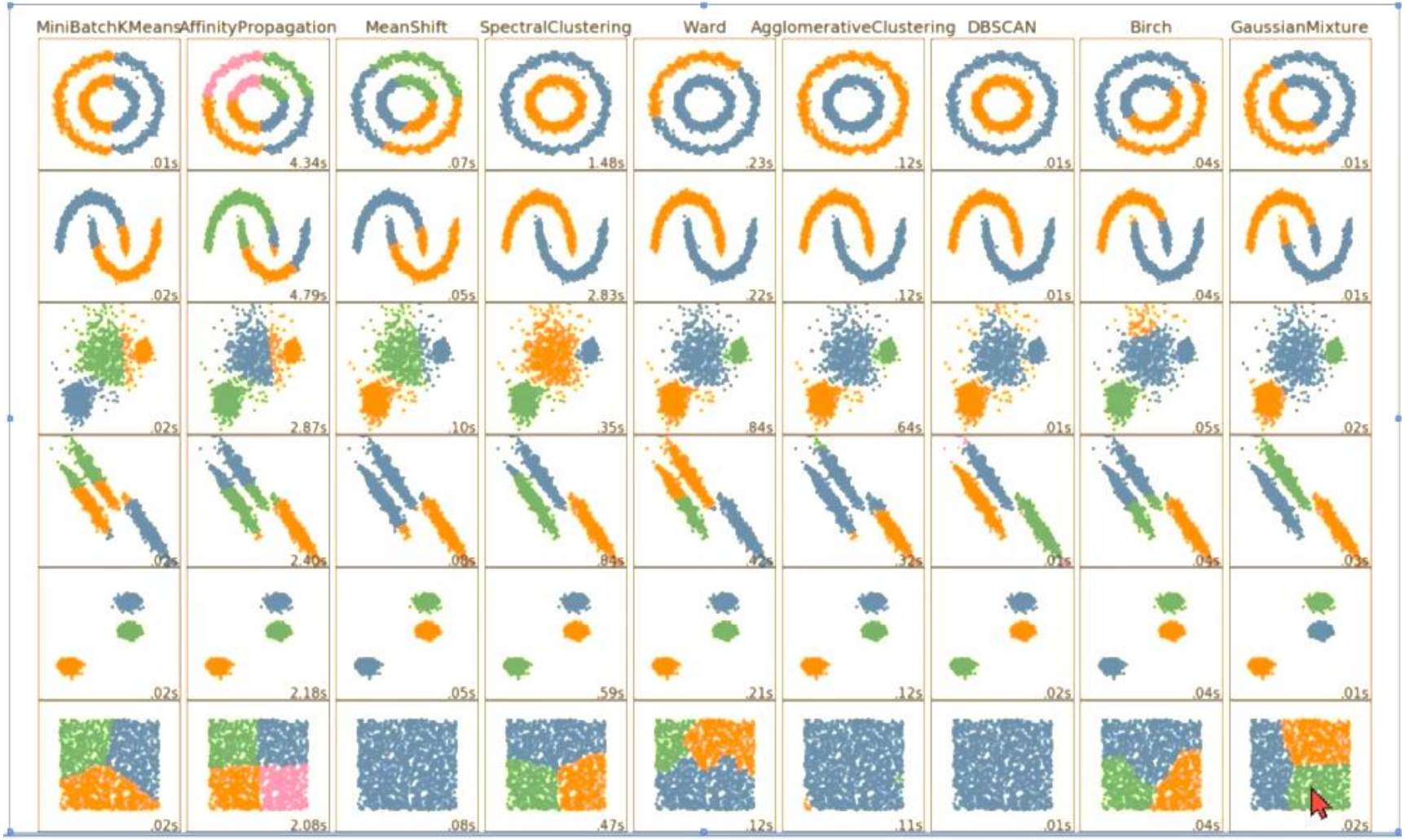



```
epoch = 5
for _ in range(epoch):
    for i in range(k):
        clusters[i]=[]

    # Calculate the distance from all points to the k cluster centers
    for i in range(x.shape[0]):
        xi = x[i]
        distances = np.sum((cluster_center-xi)**2,axis=1)
        # add the point to the cluster that is closer
        c = np.argmin(distances)
        clusters[c].append(i)

    # Recalculate the cluster centers of k clusters (all points in each cluster are added up and averaged)
    for i in range(k):
        cluster_center[i] = np.sum(x[clusters[i]],axis=0)/len(clusters[i])
```

Other Clustering Algorithms

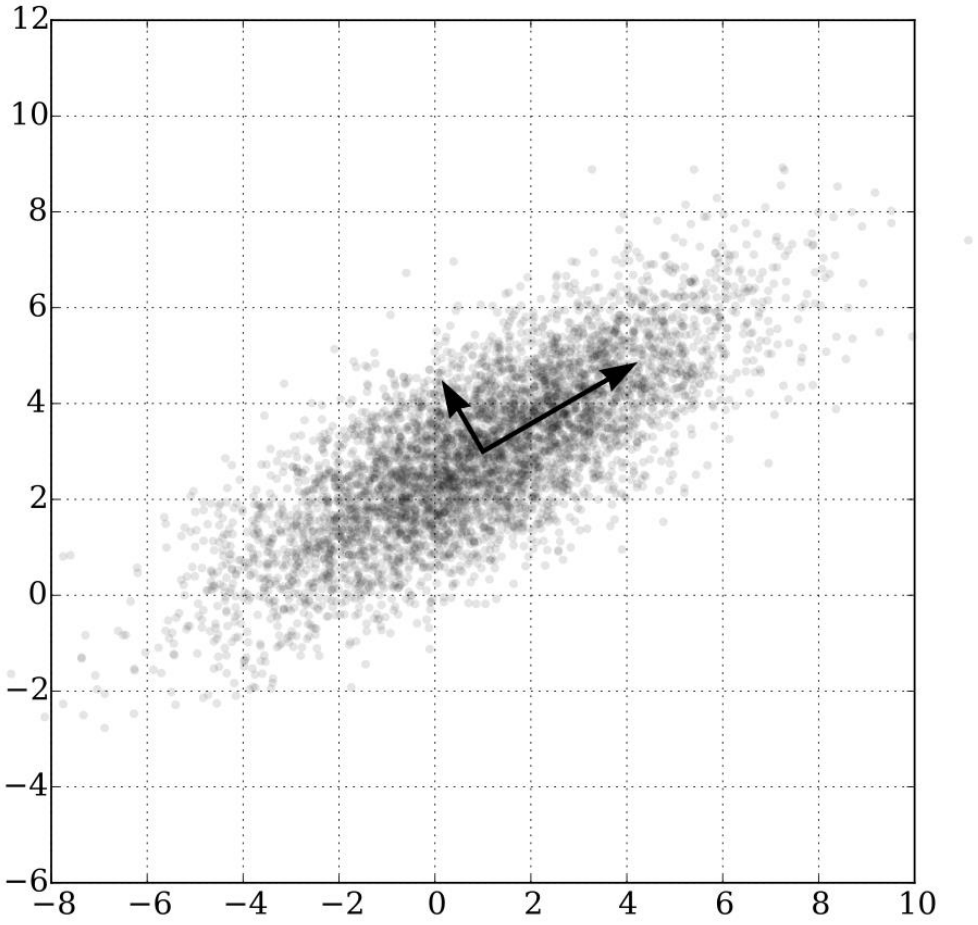


PCA --- Principal component analysis

PCA --- goal

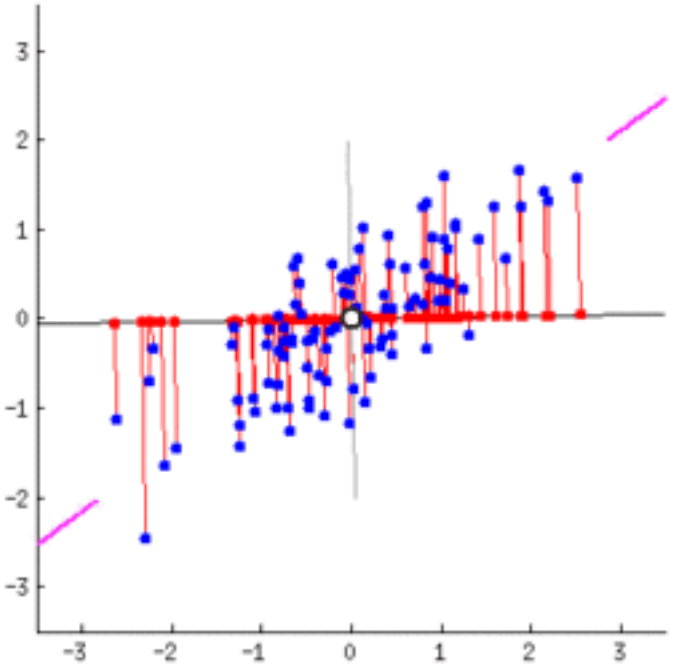
1. Reduce data dimension (get new representation)
2. Find out the variable that impacts the result the most

PCA --- example



PCA --- Process

Maximize the variance after projection on the potential new axis



https://en.wikipedia.org/wiki/Principal_component_analysis

Reference

- PCA Wikipidia

https://en.wikipedia.org/wiki/Principal_component_analysis

- Decision tree explained:

https://www.youtube.com/watch?v=7VeUPuFGJHk

- Clustering

https://www.youtube.com/watch?v=4b5d3muPQmA&t=43s