Random Forest Classifier

Decision Tree based ensemble Algorithm

Table of contents

Overview

Random Forest????!

Overview

What is a Random Forest ?

- A Random Forest is an ensemble learning algorithm used for both classification and regression tasks in machine learning.
- It operates by constructing a multitude of decision trees during training and outputs the mode (classification) or mean (regression) of the individual trees for predictions.
- It is called Random because the algorithm contains a random factor via the bagging process, and random feature selection and Forest since it constructs a multitude of decision trees.

Overview

Why use Random forest rather than Normal DTs?

The Random forest algorithm came to handle some issues with normal Decision Trees, which are as follows:

1. Overfitting : Decision trees are prone to overfitting, meaning they can capture noise in the training data and perform poorly on unseen data. Random Forests mitigate this issue by training multiple trees on different subsets of the data (through bootstrap sampling) and combining their predictions. This ensemble approach tends to generalize better to new, unseen data.

2. Bias : The ensemble nature of Random Forests, combining predictions from multiple trees, can help reduce bias introduced by individual trees. If a particular tree in the ensemble is biased in some way, the impact on the overall prediction is mitigated when aggregated with predictions from other trees.

How it works

How do Random Forests work

How Random Forests Work

01 Bootstrapping

First we take the original dataset and create subsets of it. This is achieved through a process called **bootstrap sampling**, where random samples (with replacement) are drawn from the original dataset to create multiple training sets.

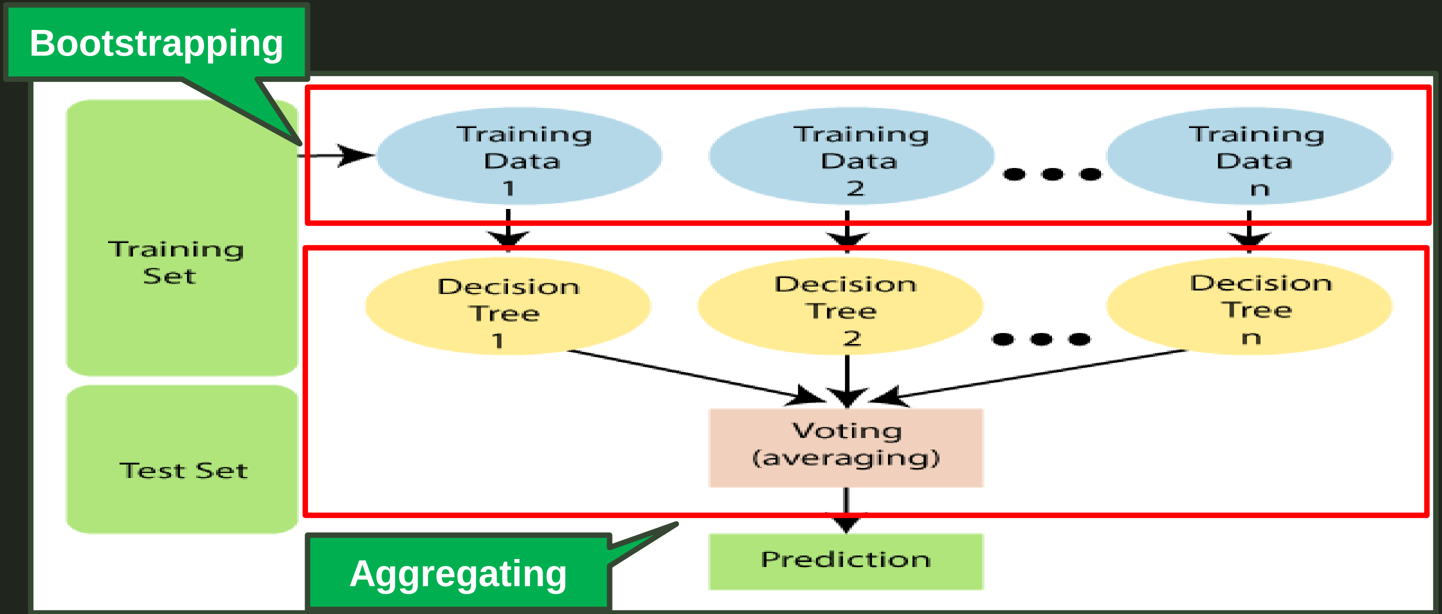
02 Feature Randomization & Tree Construction

We start constructing out Random Forest, however, introduces **randomness in the selection of features** for each split in a decision tree. Instead of considering all features at every split, only a random subset of features is considered. This helps in creating diverse and uncorrelated trees.

03 Voting Mechanism

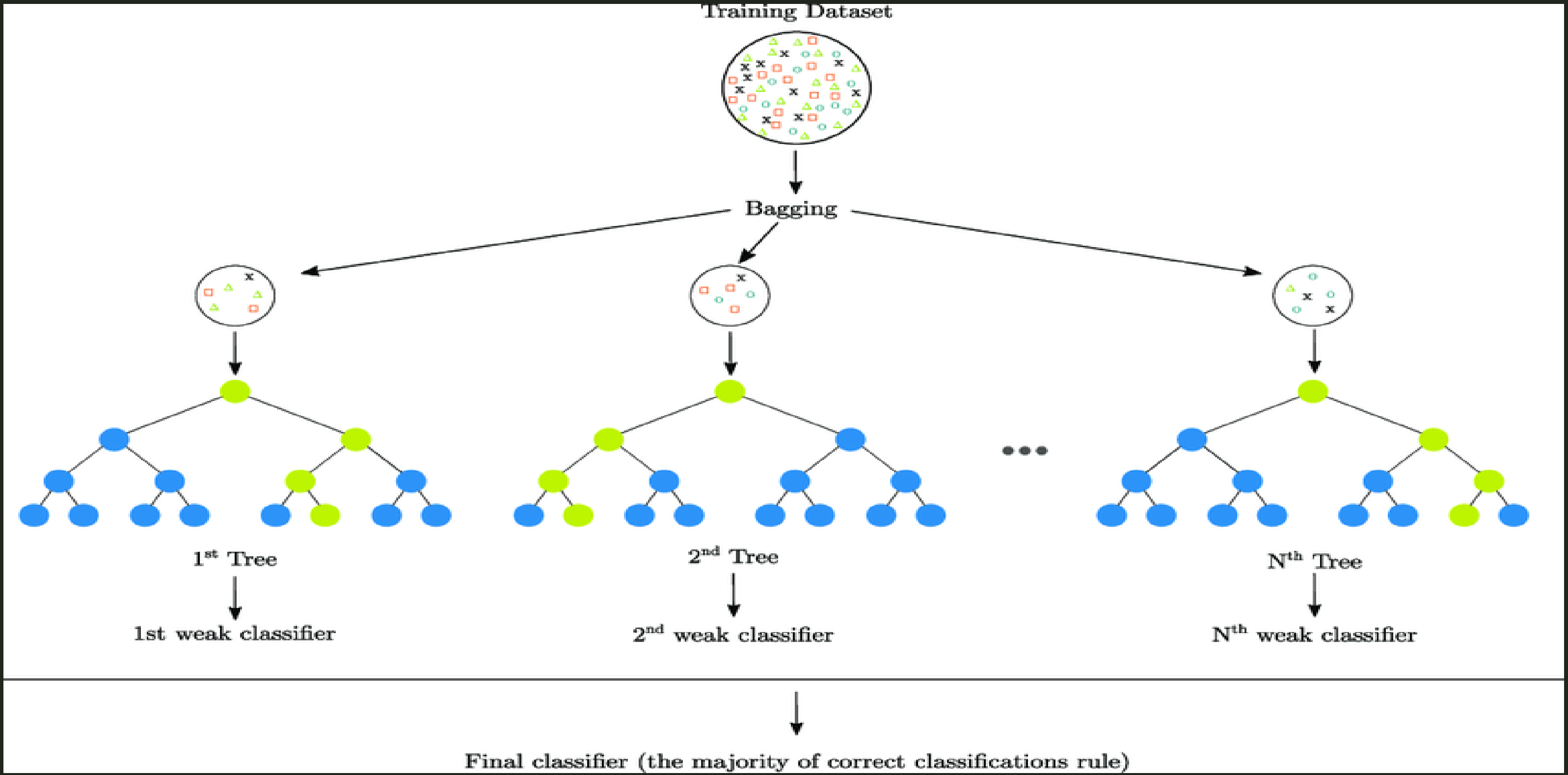
Once the individual decision trees are trained, they "vote" on the predicted class for a new input. For classification tasks, the class that receives the **majority** of votes becomes the final prediction.

How Random Forests Work



Random Forest Algorithm is uses the **bagging process**, which is short for **bootstrapping** the dataset and **aggregating** the predictions.

How Random Forests Work

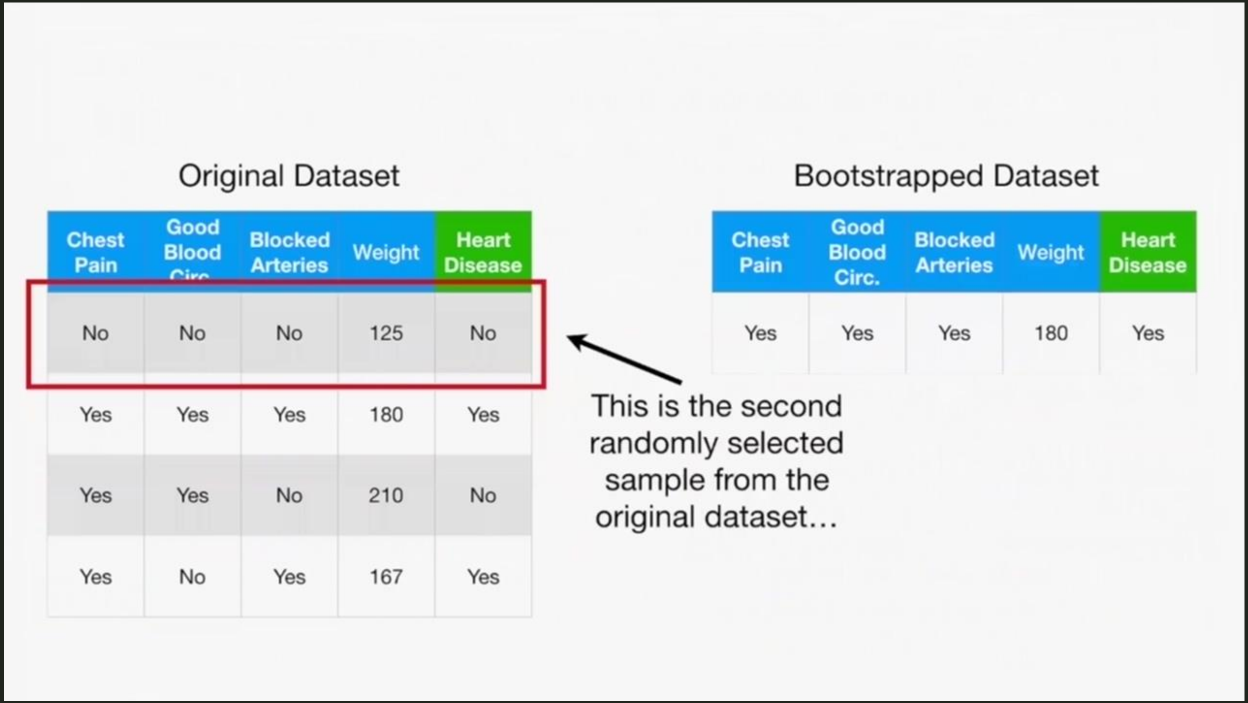


Example

How do Random Forests work, with an Example step-by-step

Step 1 - Bootstrapping the Data

We take the Original Dataset and we create N subsets (where N is the number of Trees the forest will contain)



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

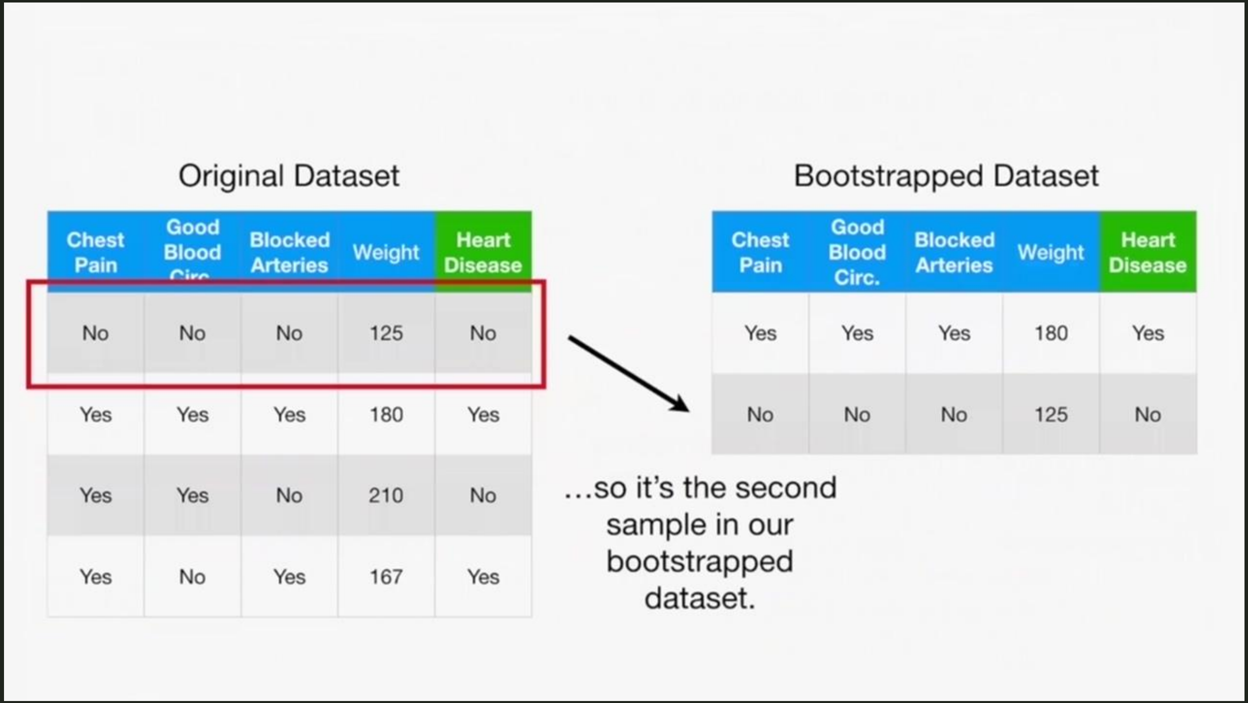
This is the second randomly selected sample from the original dataset...

Bootstrapped Dataset

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

Step 1 - Bootstrapping the Data

We take the Original Dataset and we create N subsets (where N is the number of Trees the forest will contain)



Original Dataset

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

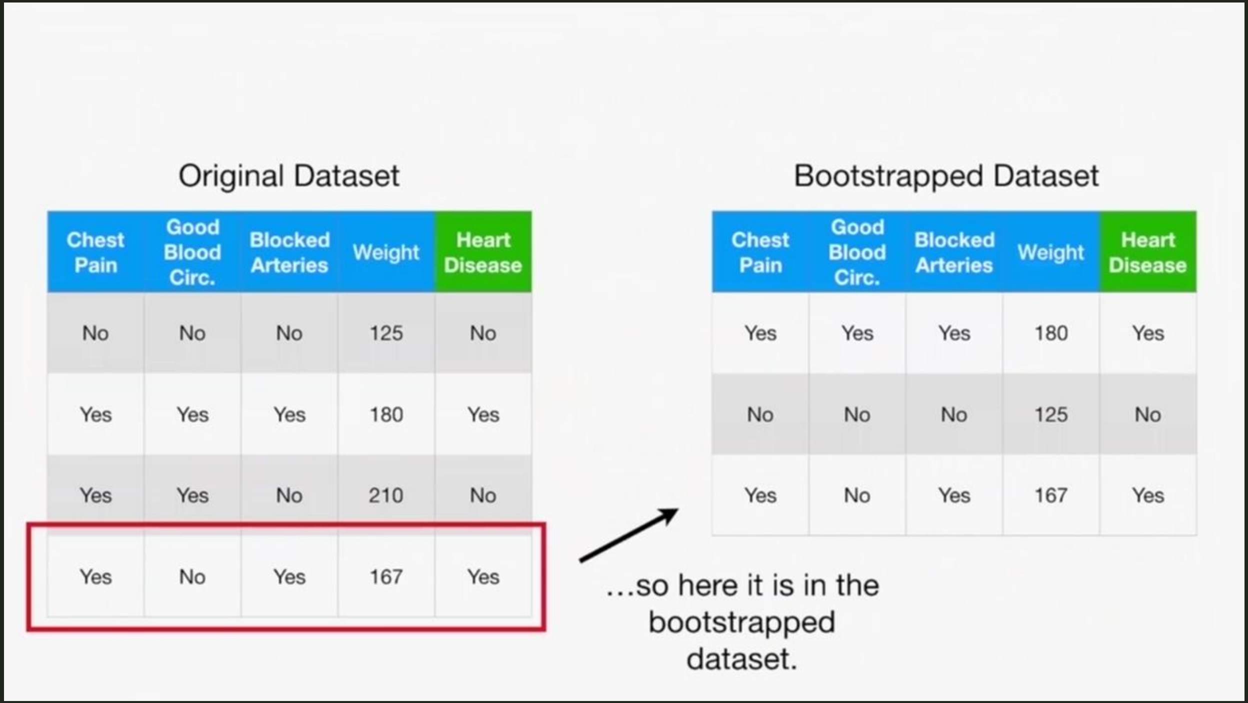
Bootstrapped Dataset

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

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

Step 1 - Bootstrapping the Data

We take the Original Dataset and we create N subsets (where N is the number of Trees the forest will contain)



Original Dataset

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

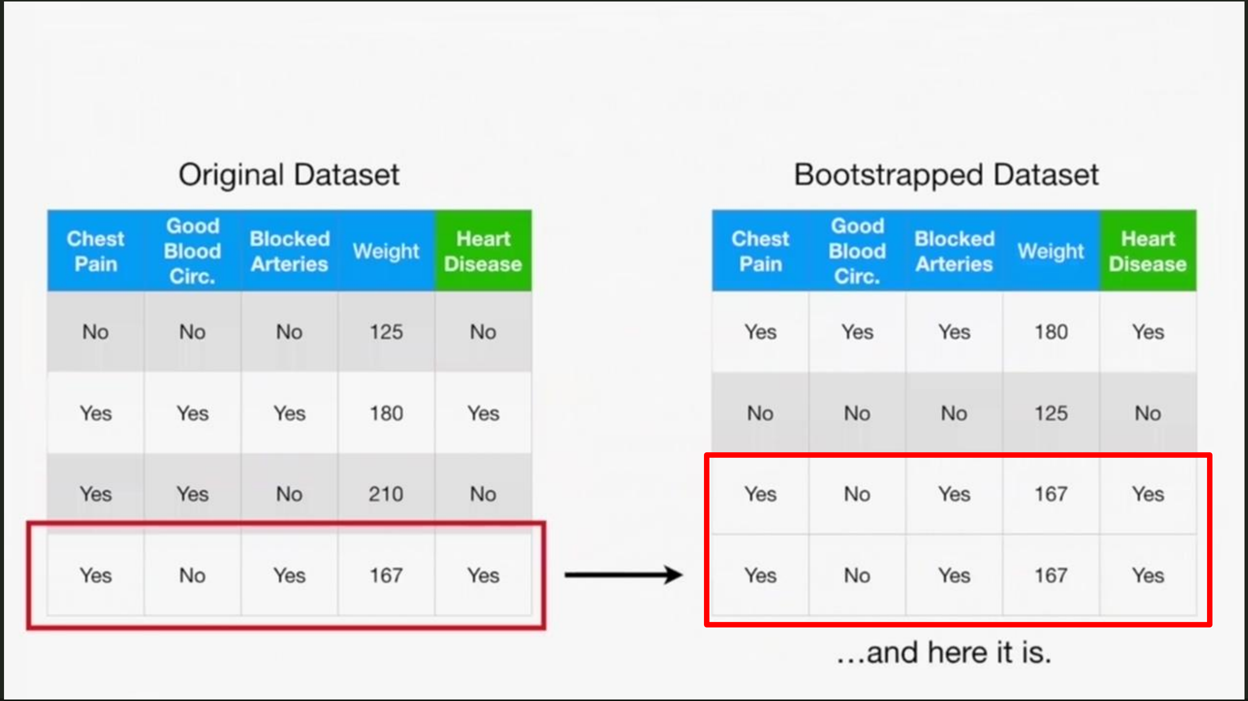
Bootstrapped Dataset

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

…so here it is in the bootstrapped dataset.

Step 1 - Bootstrapping the Data

Note: we can have duplicate rows because it is random selection of rows with **replacement**.



Original Dataset

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

Bootstrapped Dataset

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

…and here it is.

Step 1 - Bootstrapping the Data

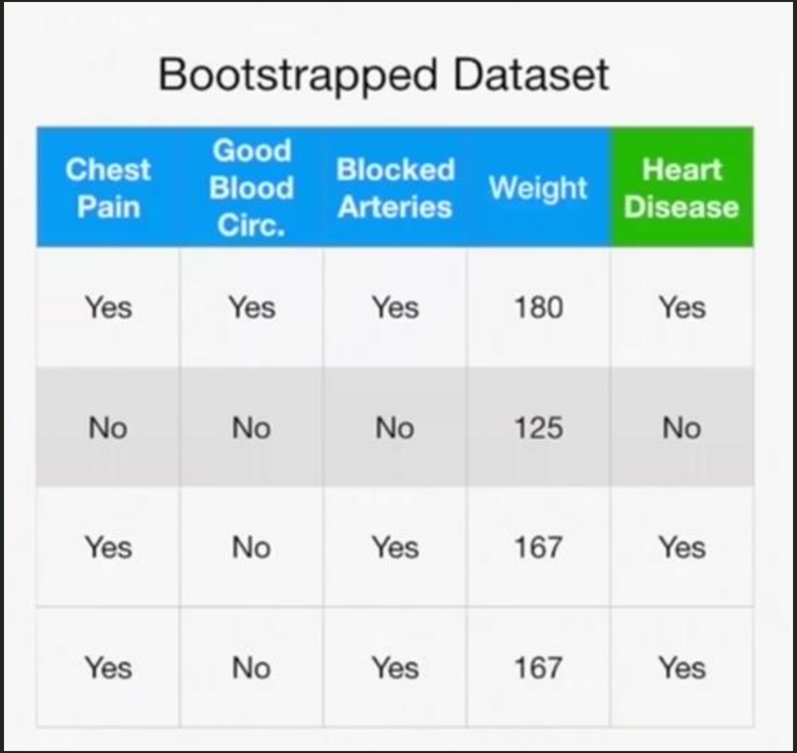
Note: the number of records in bootstrapped dataset can be less or equal to the number of records in the original dataset.

Tip : it was found the best size for the bootstrapped dataset around 60% of the original dataset

Note: All the subsets are of equal length

Note: About 1/3 of the original dataset does not get selected using bootstrapping these records are called "Out-Of-Bag" sample

Step Output : N subset (bootstrapped Datasets)

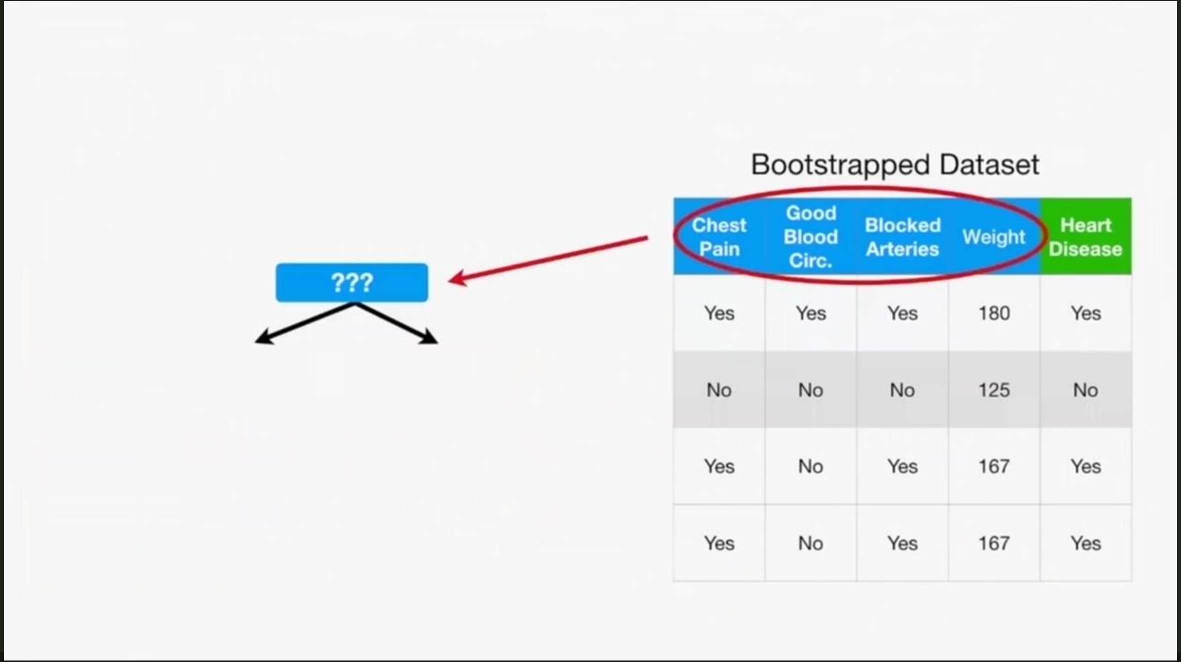


Bootstrapped Dataset

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

Step 2 - Feature Randomization & Tree Construction

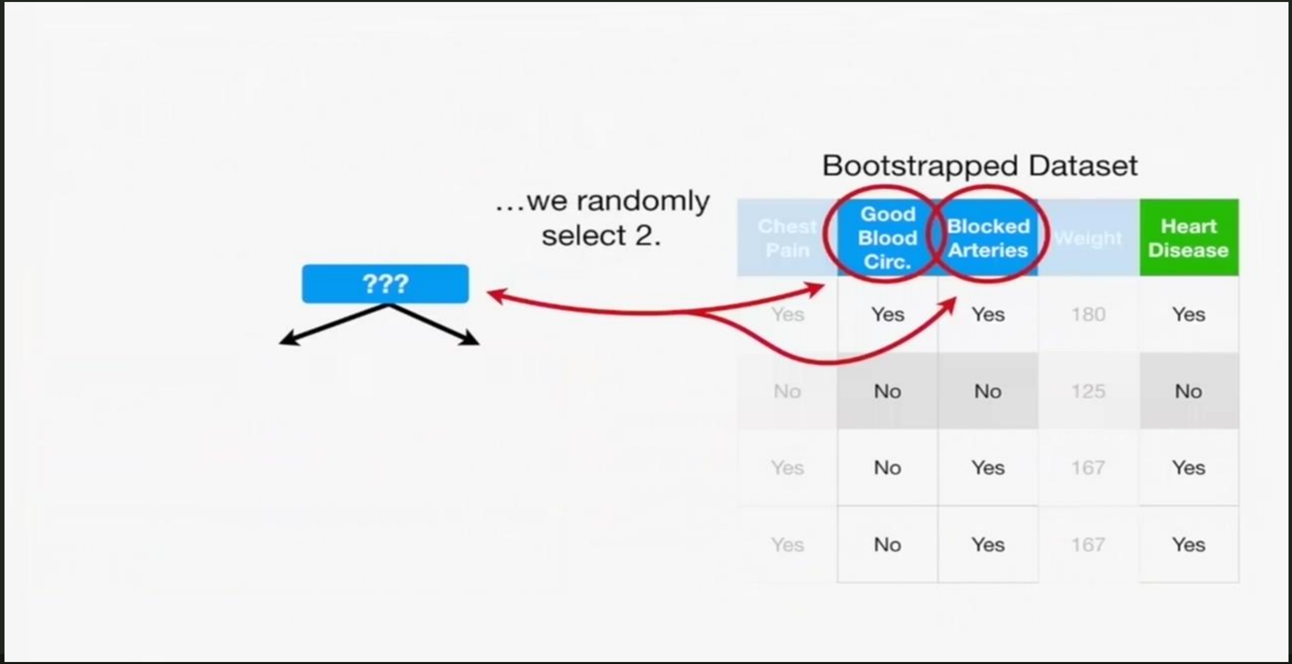
We create a decision tree for each of our bootstrapped datasets, but unlike a normal tree we will not consider all the features when selecting a splitting attribute, but a subset of features every time we pick a splitting attribute



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

Step 2 - Feature Randomization & Tree Construction

We create a decision tree for each of our bootstrapped datasets, but unlike a normal tree we will not consider all the features when selecting a splitting attribute, but a subset of features every time we pick a splitting attribute

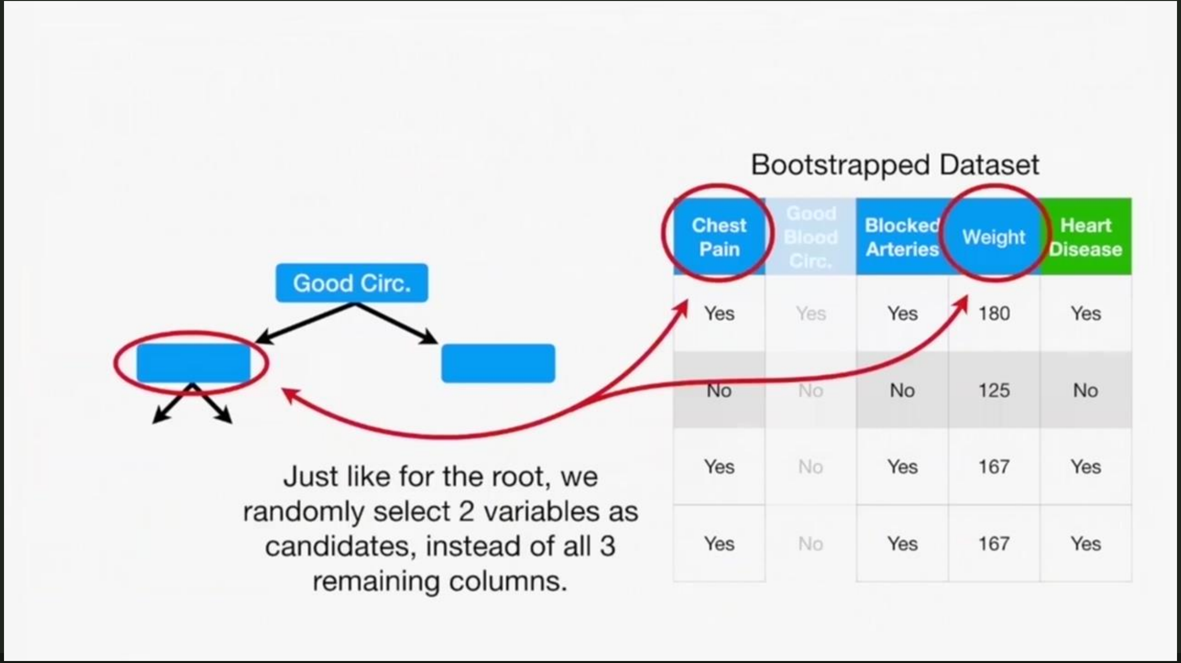


Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

Step 2 - Feature Randomization & Tree Construction

We create a decision tree for each of our bootstrapped datasets, but unlike a normal tree we will not consider all the features when selecting a splitting attribute, but a subset of features every time we pick a splitting attribute

Note: we keep going till we create a decision tree.

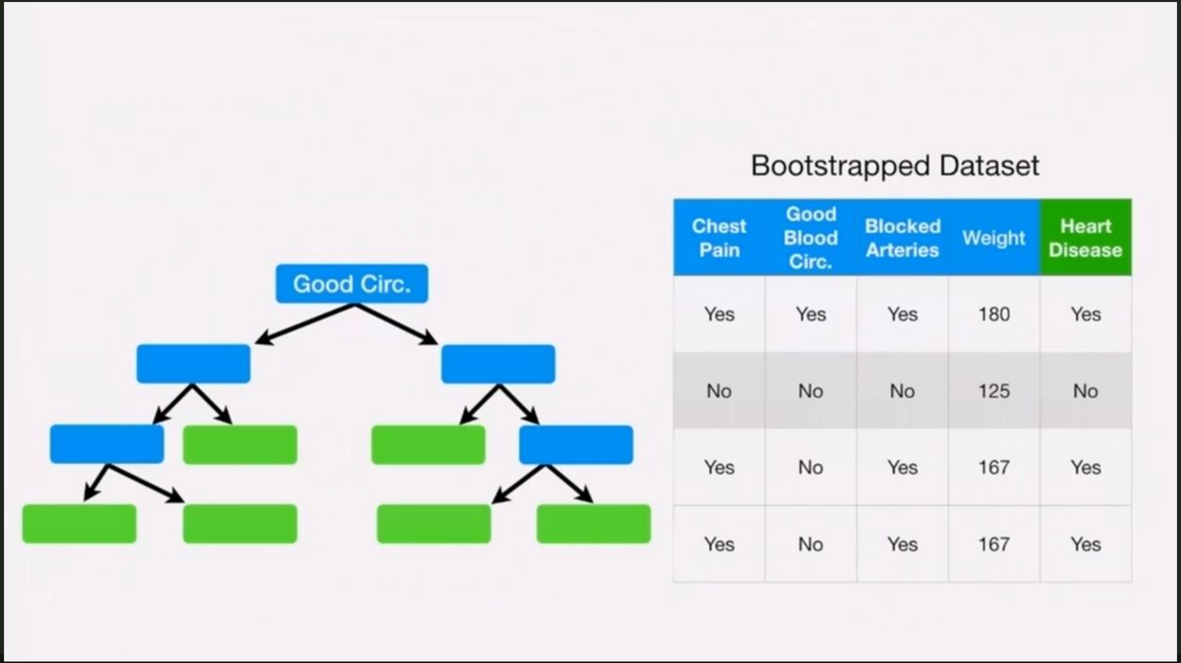


Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

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

Step 2 - Feature Randomization & Tree Construction

We create a decision tree for each of our bootstrapped datasets, but unlike a normal tree we will not consider all the features when selecting a splitting attribute, but a subset of features every time we pick a splitting attribute



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

Step 2 - Feature Randomization & Tree Construction

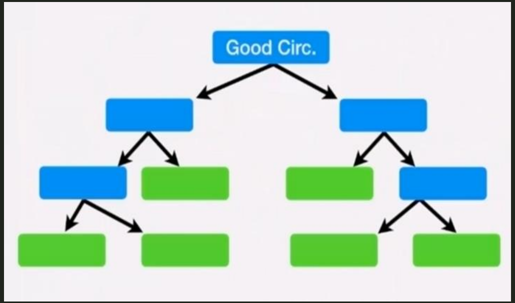
How do we calculate the best splitting Attribute?

- **Entropy / Information Gain:**
  - ***Entropy*** is a measure of the impurity or uncertainty of a set of data. It ranges from 0 (completely pure) to 1 (completely impure). When building a decision tree, the entropy of a set is calculated before and after a split, and the change in entropy is used to determine the information gain.
  - $y = -\sum_{i=1}^{k} p_i log_k(p_i)$ where ***k*** is the number of classes and ***pi*** is the probability of an element being classified as class ***i***.
  - **Information gain** is a measure of the difference in entropy between the set before and after a split. The attribute that provides the highest information gain is chosen as the split attribute.

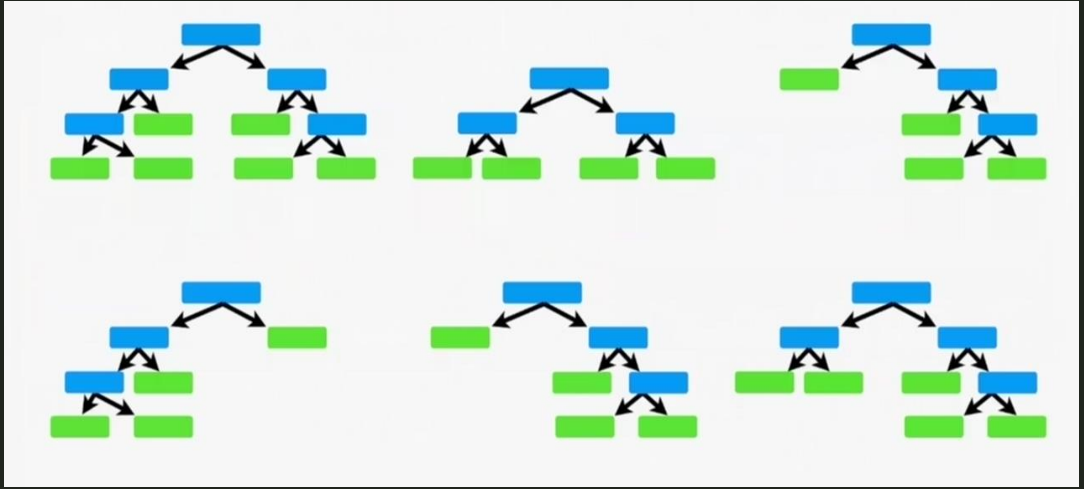
$$G(S,Q) = E(S) - \sum_{i=1}^{k} p_i E(S,Q_i)$$

Step 2 - Feature Randomization & Tree Construction

Note: Typically the number of features selected at each step, is the **square of the number of features**.
Note: Sometimes keep doing the 3 steps and increasing the number of features every time till we get better results



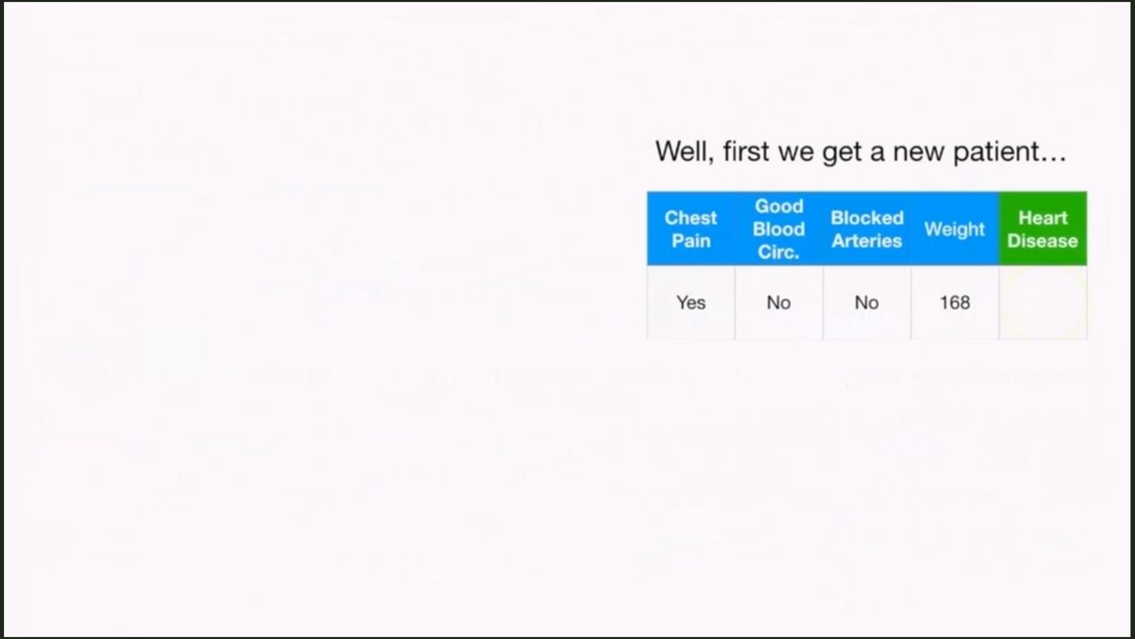
Step Output : N Decision Trees (our random forest)



Note: Due to Bootstrapping and Random Feature selection we get a variety of different trees which **overcome overfitting** and **increase accuracy**

Step 3 - Voting mechanism

After creating N Trees, we have our forest. Now we pass a record (patient) to the forest which will pass it among all the N Trees and keep track of the decisions

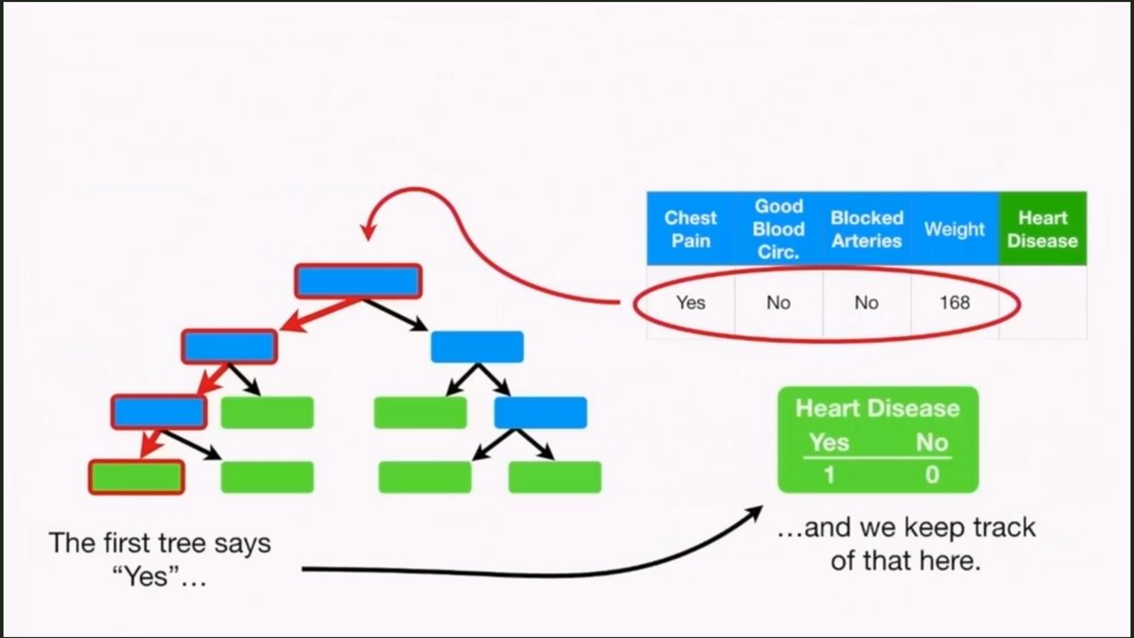


Well, first we get a new patient...

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

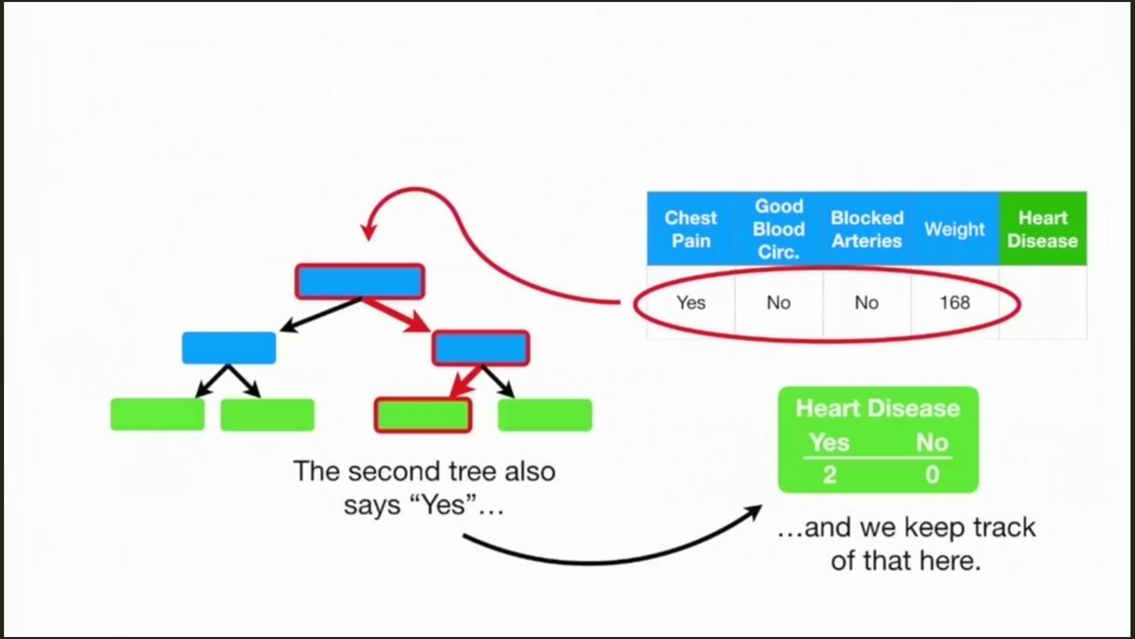
Step 3 - Voting mechanism

After creating N Trees, we have our forest. Now we pass a record (patient) to the forest which will pass it among all the N Trees and keep track of the decisions



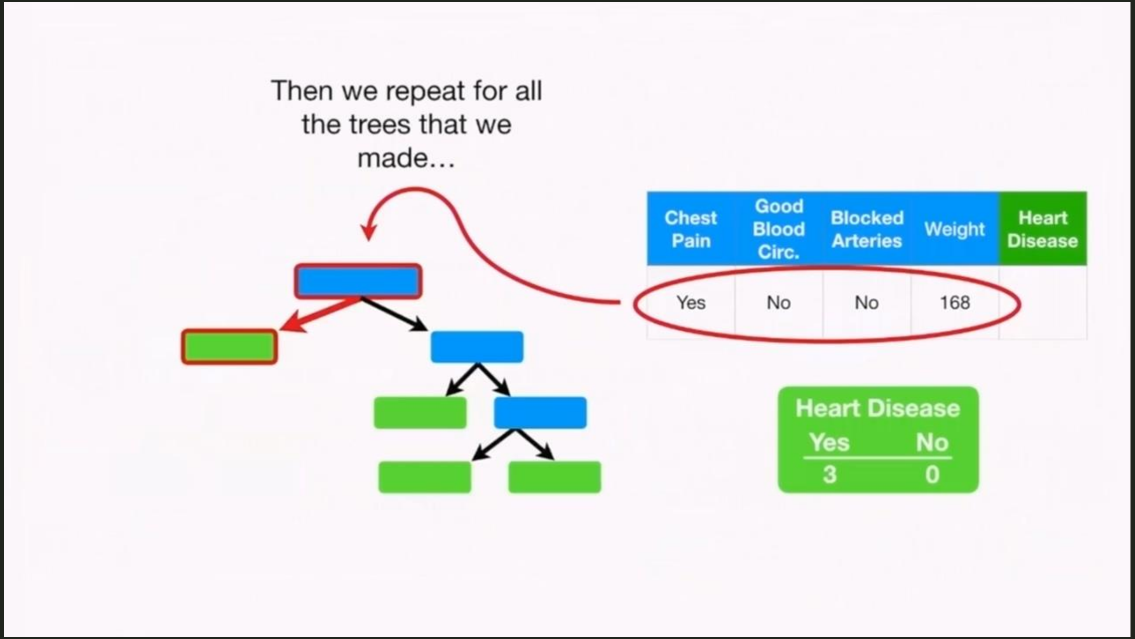
Step 3 - Voting mechanism

After creating N Trees, we have our forest. Now we pass a record (patient) to the forest which will pass it among all the N Trees and keep track of the decisions



Step 3 - Voting mechanism

After creating N Trees, we have our forest. Now we pass a record (patient) to the forest which will pass it among all the N Trees and keep track of the decisions

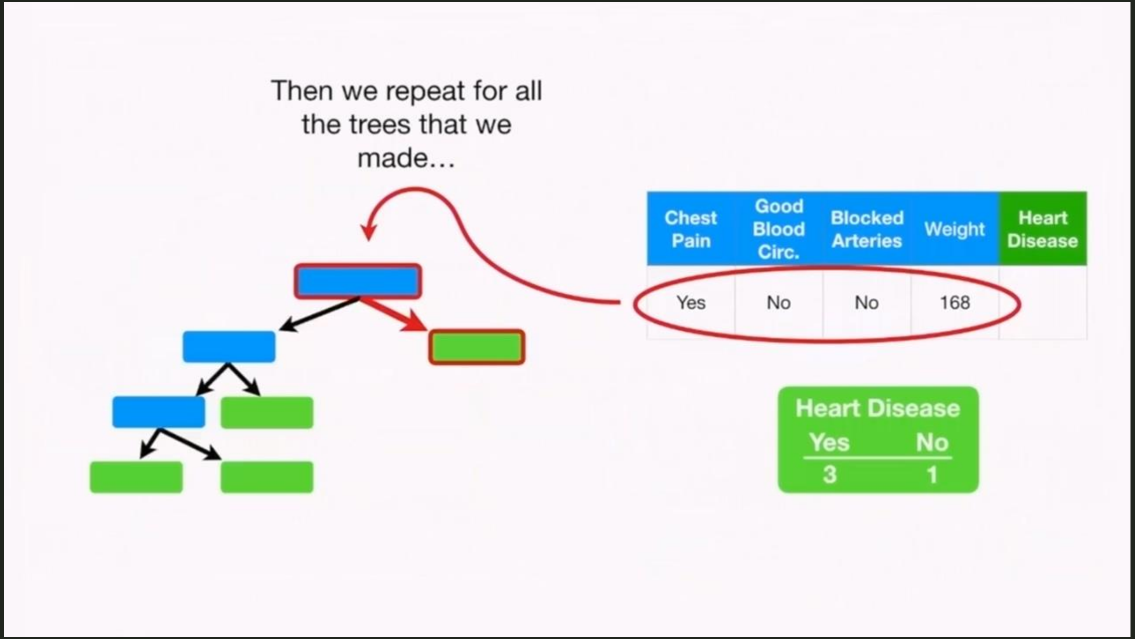


Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	No	168	

Heart Disease	
Yes	No
3	0

Step 3 - Voting mechanism

After creating N Trees, we have our forest. Now we pass a record (patient) to the forest which will pass it among all the N Trees and keep track of the decisions

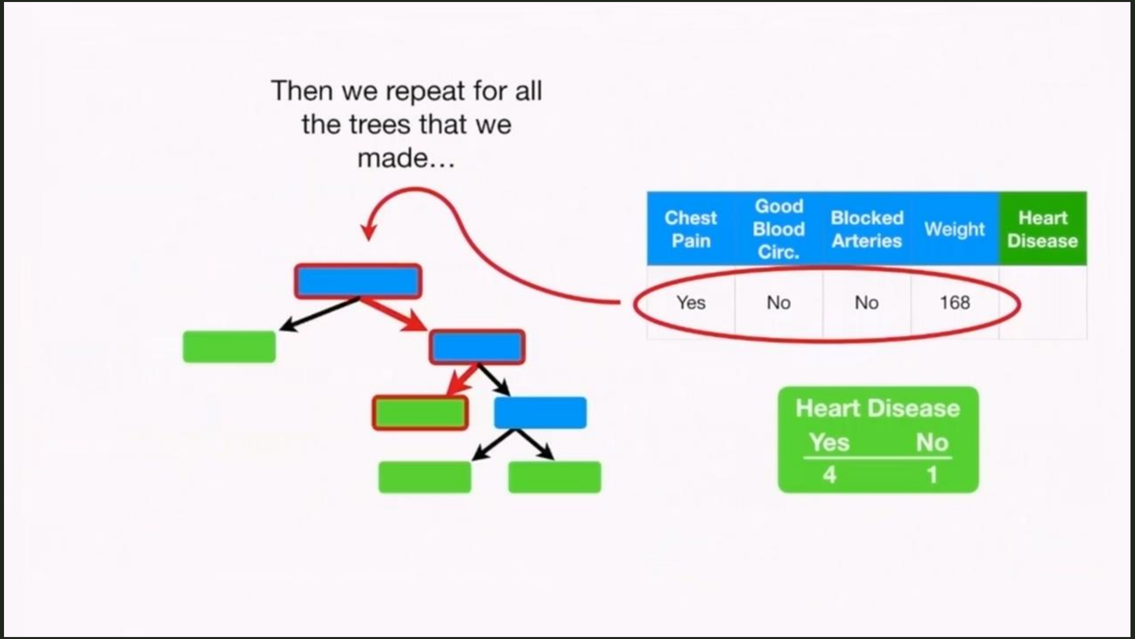


Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	No	168	

Heart Disease	
Yes	No
3	1

Step 3 - Voting mechanism

After creating N Trees, we have our forest. Now we pass a record (patient) to the forest which will pass it among all the N Trees and keep track of the decisions

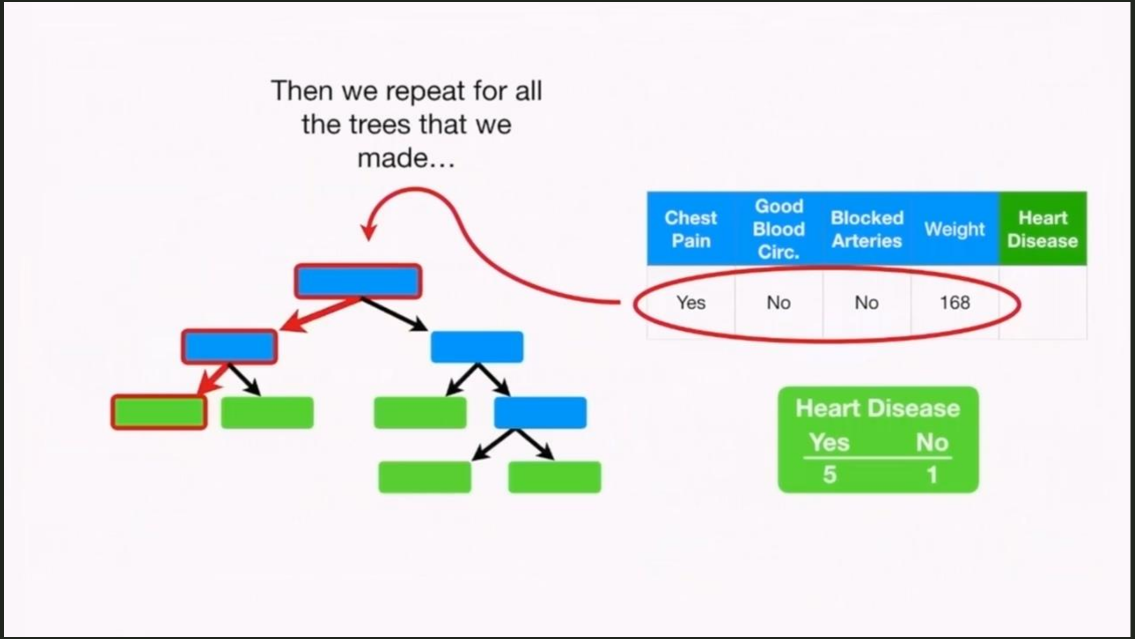


Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	No	No	168	

Heart Disease	
Yes	No
4	1

Step 3 - Voting mechanism

After creating N Trees, we have our forest. Now we pass a record (patient) to the forest which will pass it among all the N Trees and keep track of the decisions



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

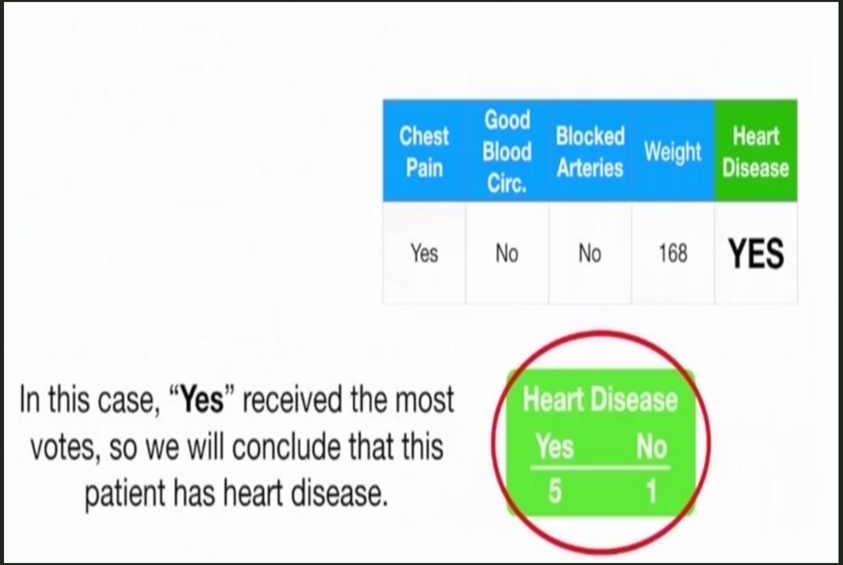
Heart Disease	
Yes	No
5	1

Step 3 - Voting mechanism

After all the trees have made there decisions, the class label with the majority votes will be the final decision of the Random Forest algorithm.

Note:

- For **classification** we take the majority of votes
- while in **regression** we take the average.



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

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

Heart Disease	
Yes	No
5	1

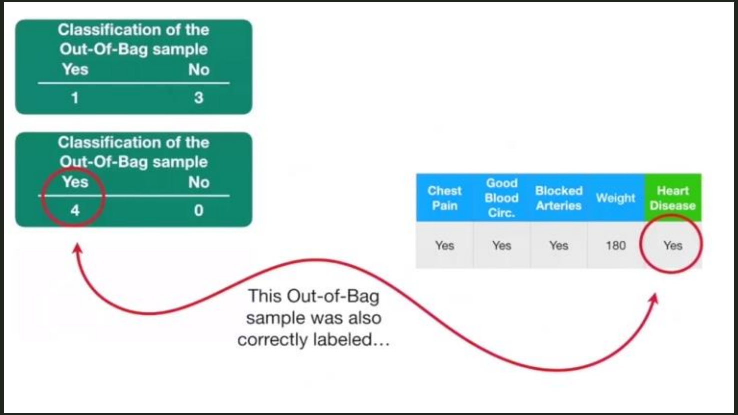
Final Notes

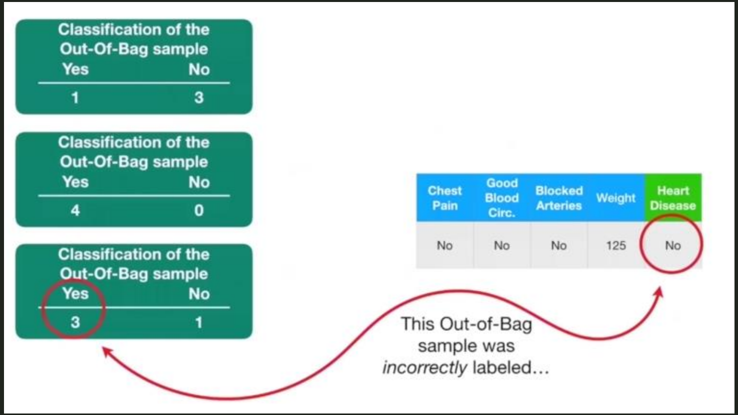
Note: About 1/3 of the original dataset does not get selected using bootstrapping these records are called “**Out-Of-Bag**” sample

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

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

Note: sometimes it is split into test, and train


Classification of the Out-Of-Bag sample
Yes No
1 3
Classification of the Out-Of-Bag sample
Yes No
4 0
Chest Pain
Good Blood Circ.
Blocked Arteries
Weight
Heart Disease
Yes Yes Yes 180 Yes
This Out-of-Bag sample was also correctly labeled...


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
Chest Pain
Good Blood Circ.
Blocked Arteries
Weight
Heart Disease
No No No 125 No
This Out-of-Bag sample was *incorrectly* labeled...

Advantages

Advantages

What makes the Random Forest Great?

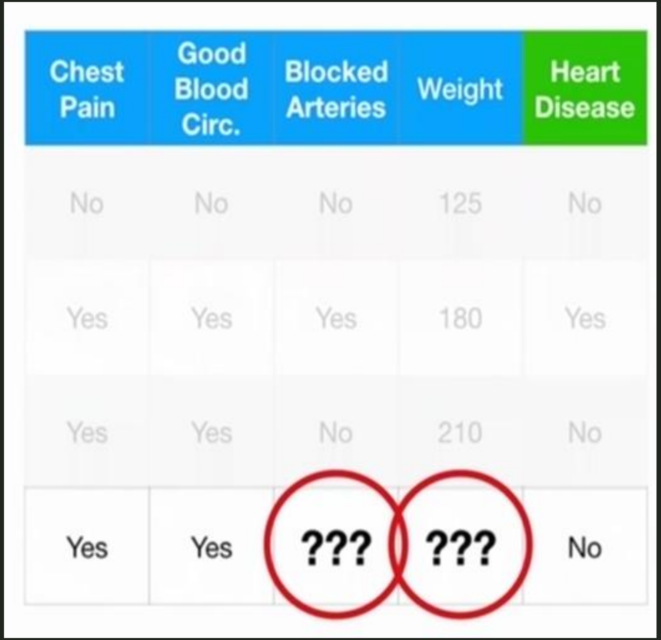
- **Robustness**: Random Forests are less prone to overfitting compared to individual decision trees.
- **Accuracy**: They often provide higher accuracy because they combine the predictions of multiple trees.
- **Handling Missing Values**: Random Forests can handle missing values in the dataset.

Advantages

How does it Handle Missing values

It handles missing values via a proximity matrix.

First, we make an initial guess which will probable be wrong then we will try to refine it.

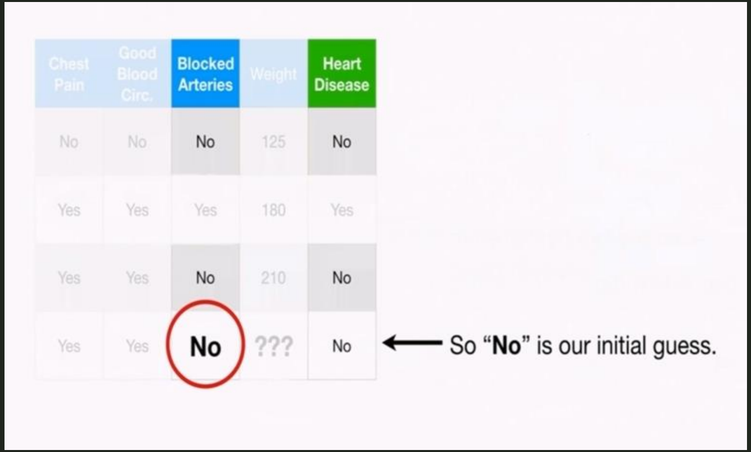


Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	???	???	No

Advantages

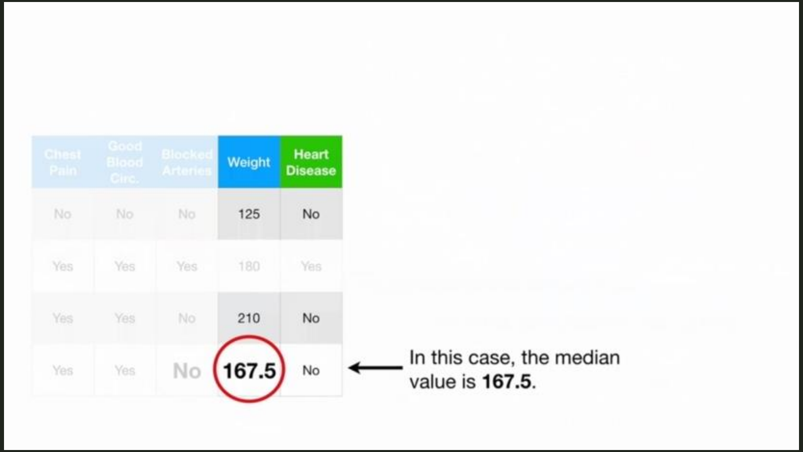
How does it Handle Missing values

First, we make an initial guess which will probable be wrong then we will try to refine it.



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	Yes	No	???	No

So "**No**" is our initial guess.



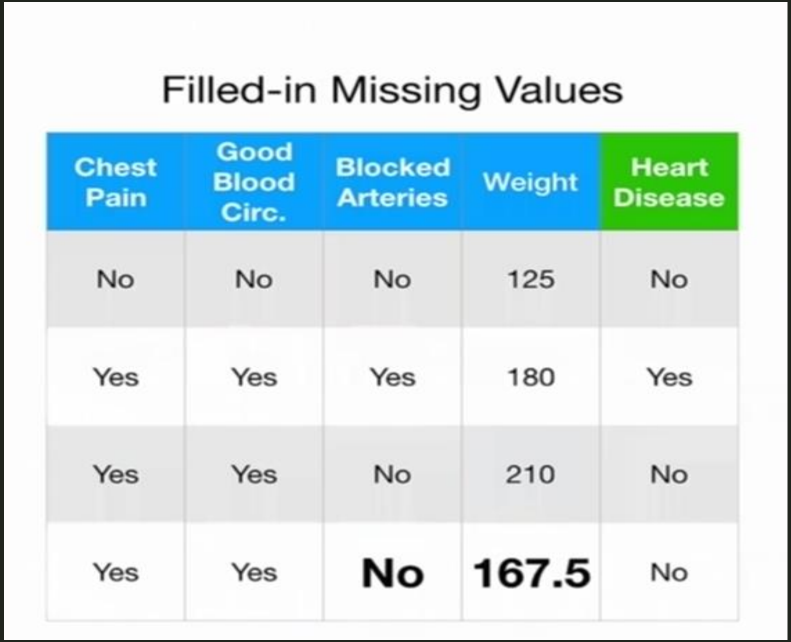
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	No	167.5	No

In this case, the median value is **167.5**.

Advantages

How does it Handle Missing values

First, we make an initial guess which will probable be wrong then we will try to refine it.



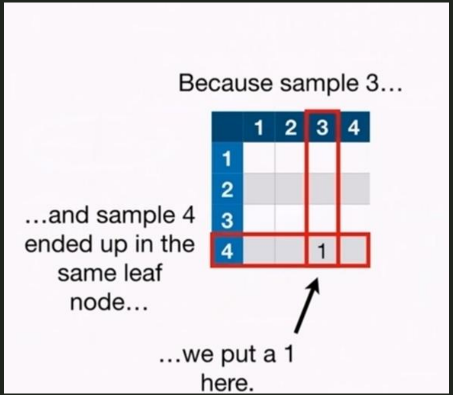
Filled-in Missing Values

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	Yes	**No**	**167.5**	No

Advantages

How does it Handle Missing values

Second, we create a proximity matrices by marking 1 for every pair of records that end up in the same leaf node.

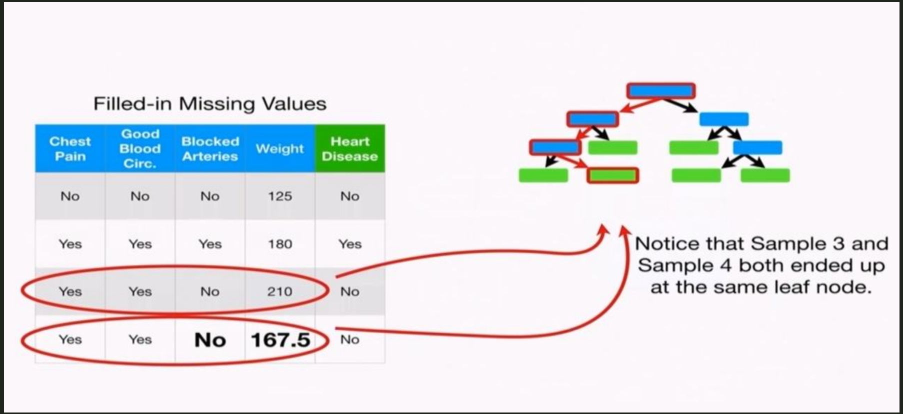


Because sample 3…

…and sample 4 ended up in the same leaf node…

	1	2	3	4
1				
2				
3				
4			1	

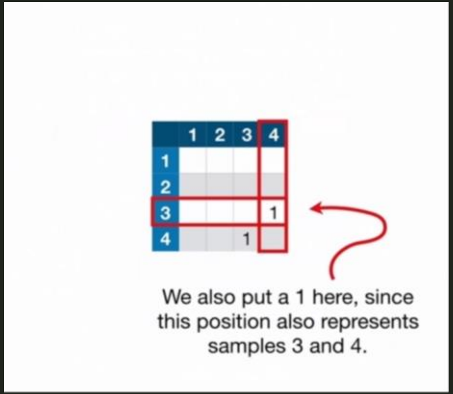
…we put a 1 here.



Filled-in Missing Values

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	Yes	**No**	**167.5**	No

Notice that Sample 3 and Sample 4 both ended up at the same leaf node.



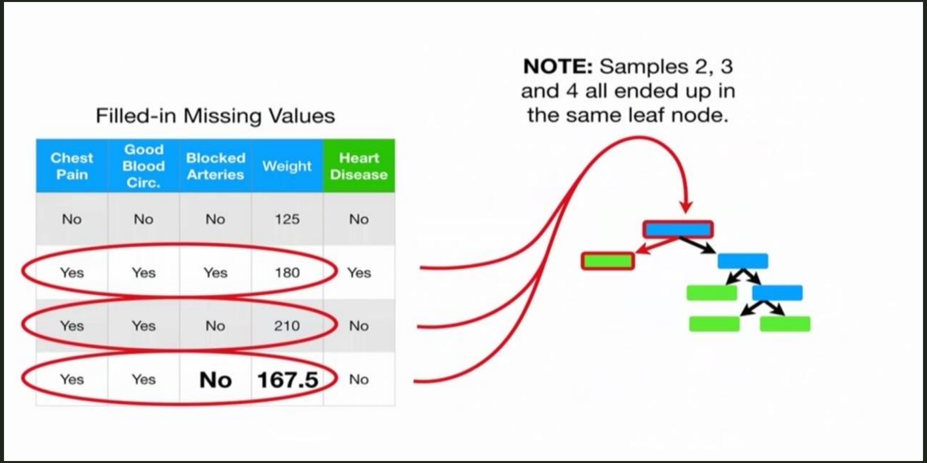
	1	2	3	4
1				
2				
3				1
4			1	

We also put a 1 here, since this position also represents samples 3 and 4.

Advantages

How does it Handle Missing values

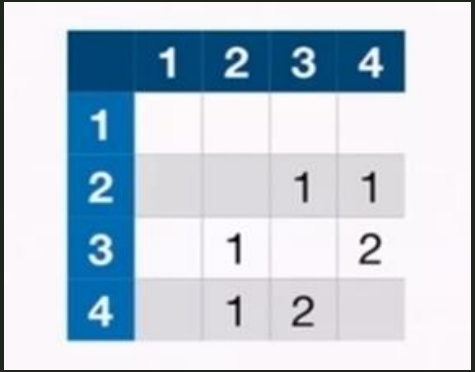
Third, we Keep doing for all the trees and updating the proximity Metrix



Filled-in Missing Values

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	Yes	**No**	**167.5**	No

NOTE: Samples 2, 3 and 4 all ended up in the same leaf node.

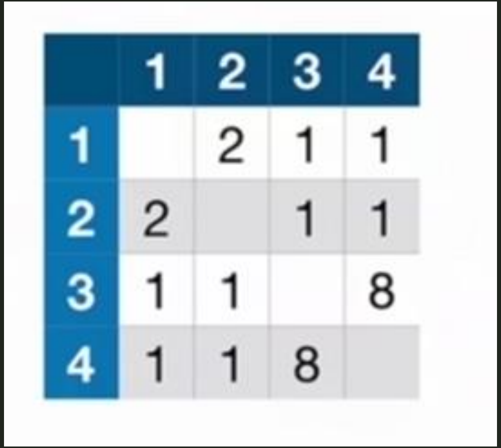


	1	2	3	4
1				
2			1	1
3		1		2
4		1	2	

Advantages

How does it Handle Missing values

Third, We will end up with the complete proximity matric

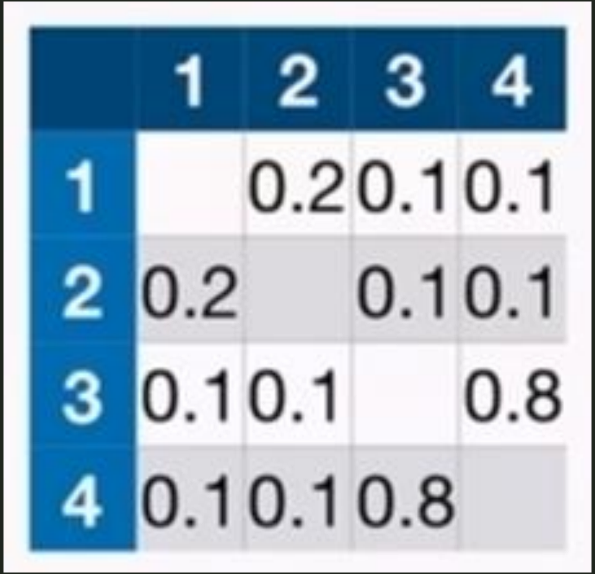


	1	2	3	4
1		2	1	1
2	2		1	1
3	1	1		8
4	1	1	8	

Advantages

How does it Handle Missing values

Forth, We divide the values by the total number of trees in this example we assume it to be 10

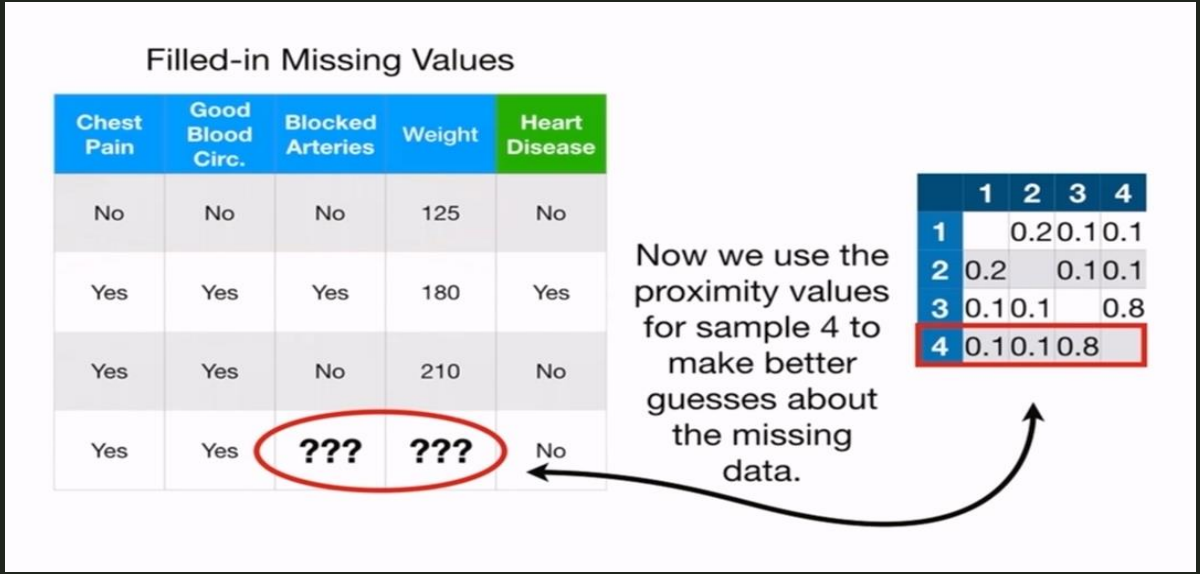


	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Advantages

How does it Handle Missing values

Fifth, We use the proximity matrices to make better guesses



Filled-in Missing Values

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	Yes	???	???	No

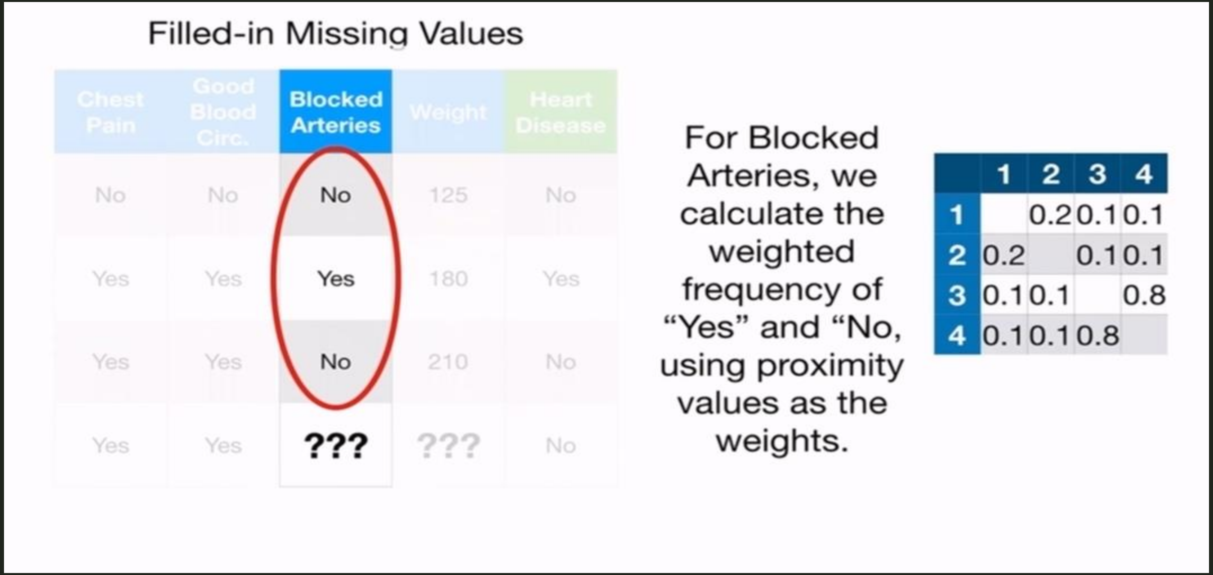
Now we use the proximity values for sample 4 to make better guesses about the missing data.

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Advantages

How does it Handle Missing values

Fifth, We use the proximity matrices to make better guesses



Filled-in Missing Values

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	Yes	???	???	No

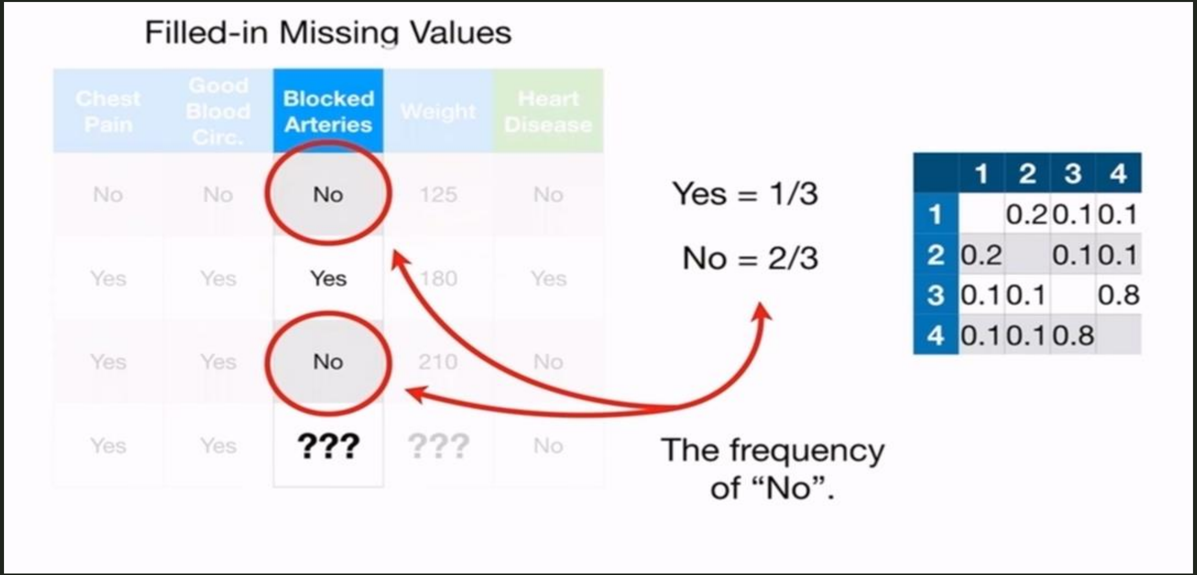
For Blocked Arteries, we calculate the weighted frequency of "Yes" and "No, using proximity values as the weights.

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Advantages

How does it Handle Missing values

Fifth, We use the proximity matrices to make better guesses



Filled-in Missing Values

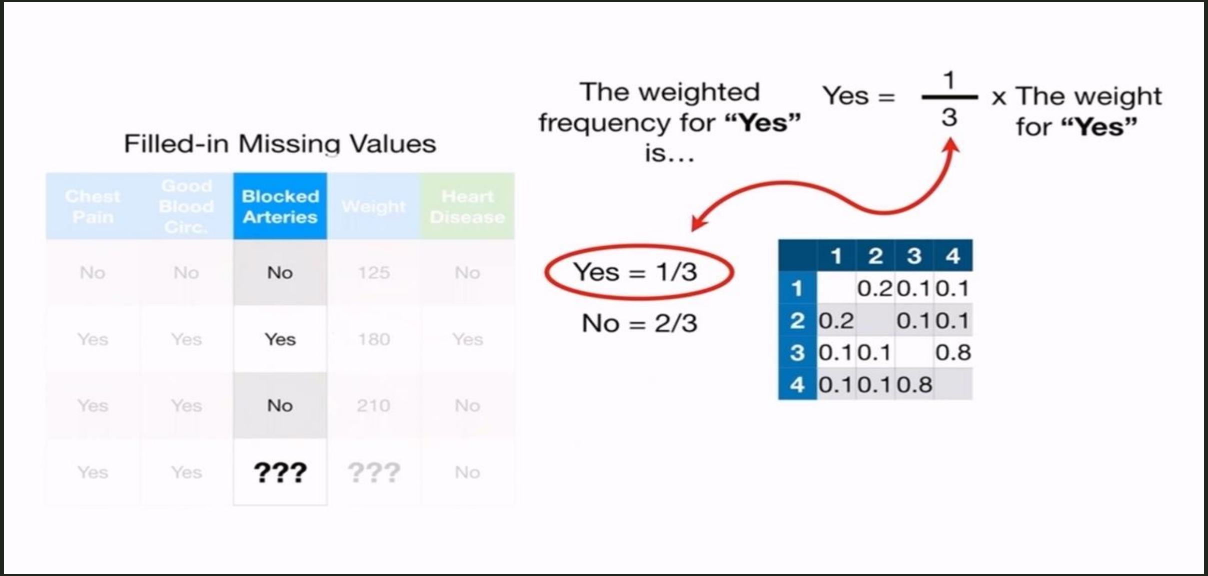
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	???	???	No

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Advantages

How does it Handle Missing values

Fifth, We use the proximity matrices to make better guesses



Filled-in Missing Values

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	Yes	???	???	No

The weighted frequency for **"Yes"** is...

$$\text{Yes} = \frac{1}{3} \times \text{The weight for "Yes"}$$

Yes = 1/3

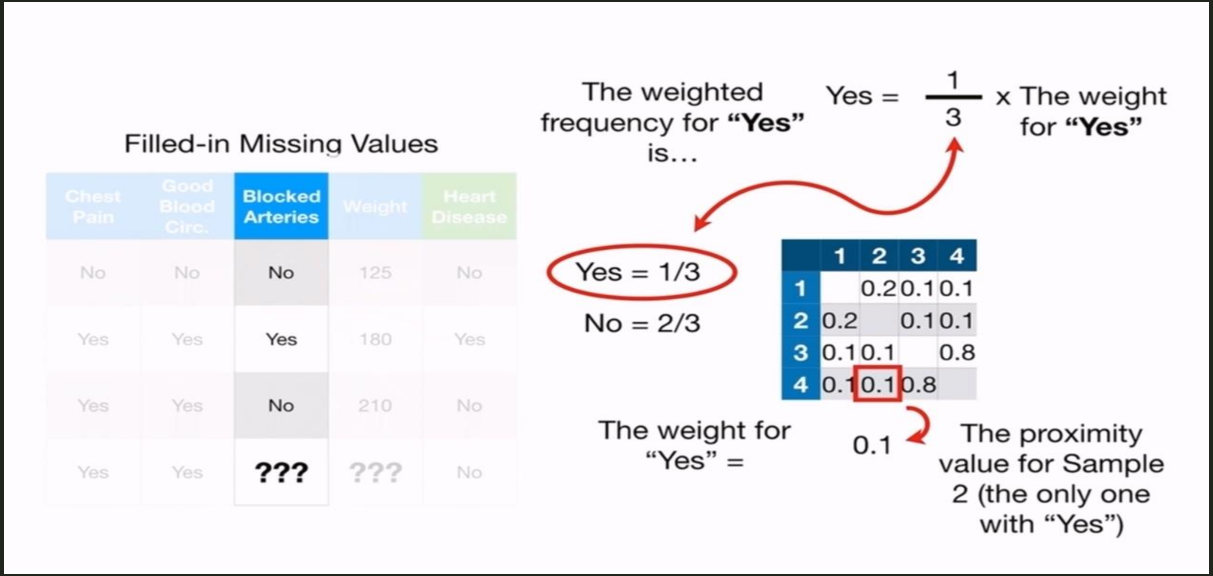
No = 2/3

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Advantages

How does it Handle Missing values

Fifth, We use the proximity matrices to make better guesses



Filled-in Missing Values

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	Yes	???	???	No

The weighted frequency for **"Yes"** is…

Yes = $\frac{1}{3}$ x The weight for **"Yes"**

Yes = 1/3

No = 2/3

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

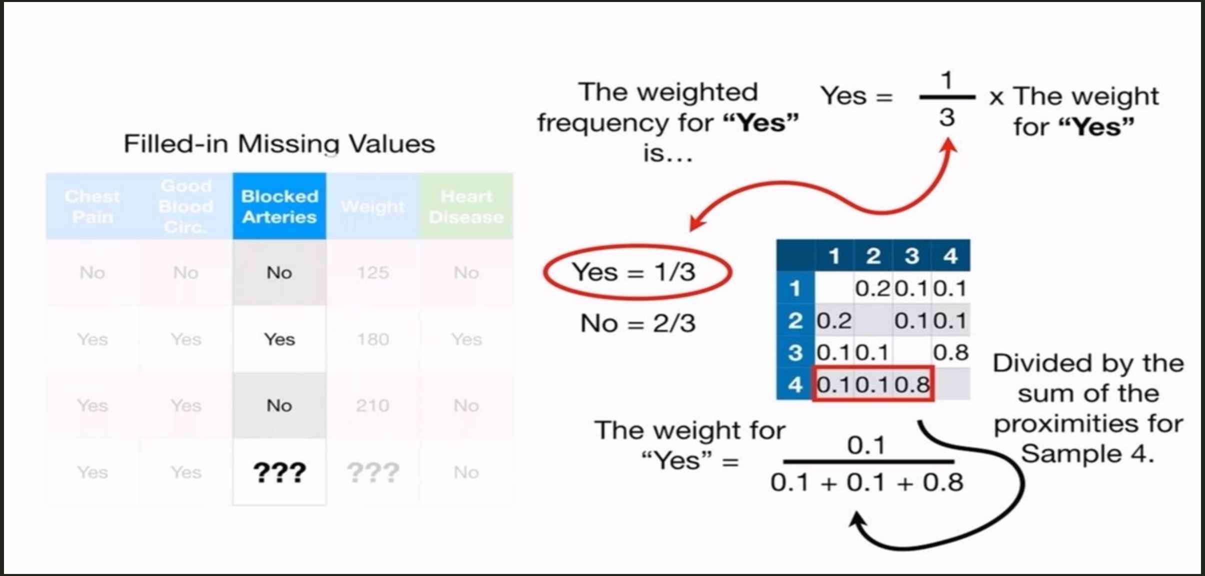
The weight for "Yes" = 0.1

The proximity value for Sample 2 (the only one with "Yes")

Advantages

How does it Handle Missing values

Fifth, We use the proximity matrices to make better guesses



Filled-in Missing Values

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	Yes	???	???	No

The weighted frequency for **"Yes"** is…

$$\text{Yes} = \frac{1}{3} \times \text{The weight for "Yes"}$$

Yes = 1/3

No = 2/3

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

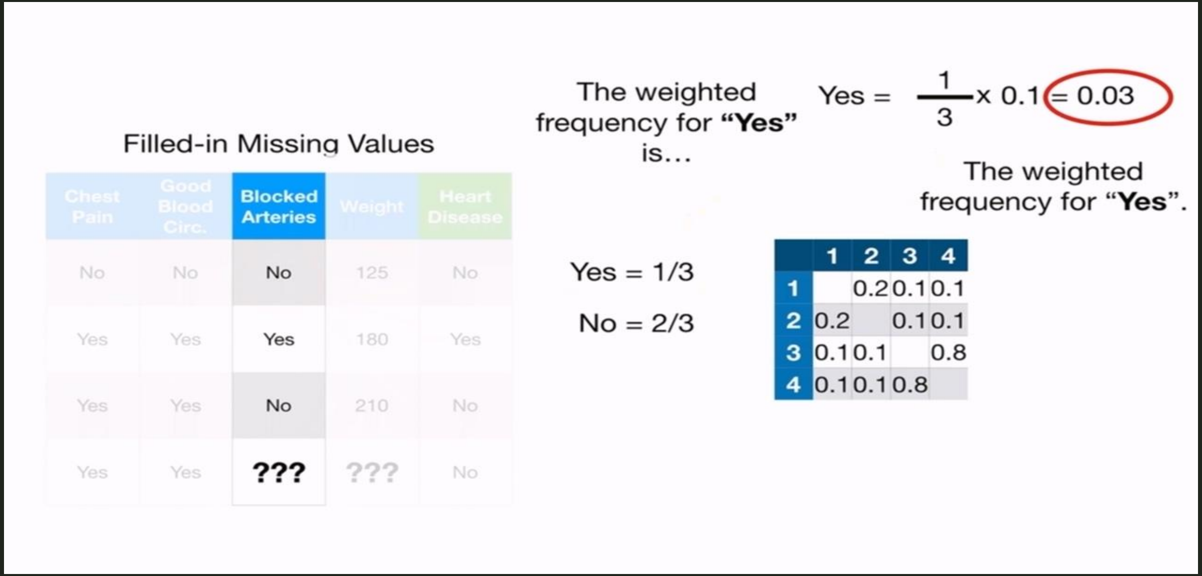
$$\text{The weight for "Yes"} = \frac{0.1}{0.1 + 0.1 + 0.8}$$

Divided by the sum of the proximities for Sample 4.

Advantages

How does it Handle Missing values

Fifth, We use the proximity matrices to make better guesses



Filled-in Missing Values

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	Yes	???	???	No

The weighted frequency for **"Yes"** is…

Yes = 1/3

No = 2/3

$$\text{Yes} = \frac{1}{3} \times 0.1 = 0.03$$

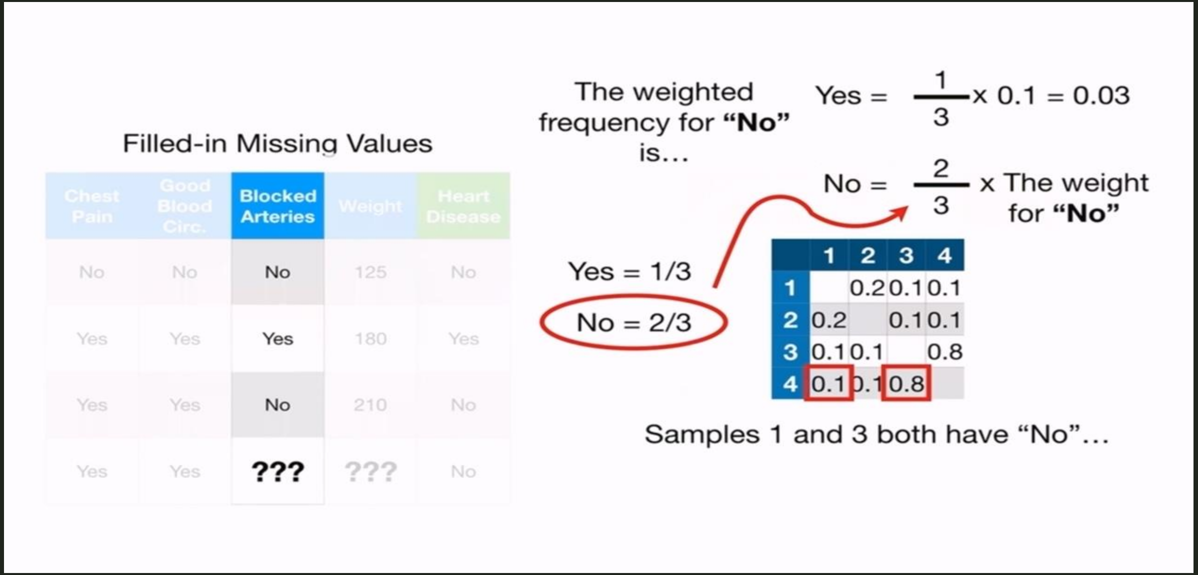
The weighted frequency for **"Yes"**.

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Advantages

How does it Handle Missing values

Fifth, We use the proximity matrices to make better guesses



Filled-in Missing Values

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	Yes	???	???	No

The weighted frequency for **"No"** is...

$$\text{Yes} = \frac{1}{3} \times 0.1 = 0.03$$

$$\text{No} = \frac{2}{3} \times \text{The weight for "No"}$$

Yes = 1/3

No = 2/3

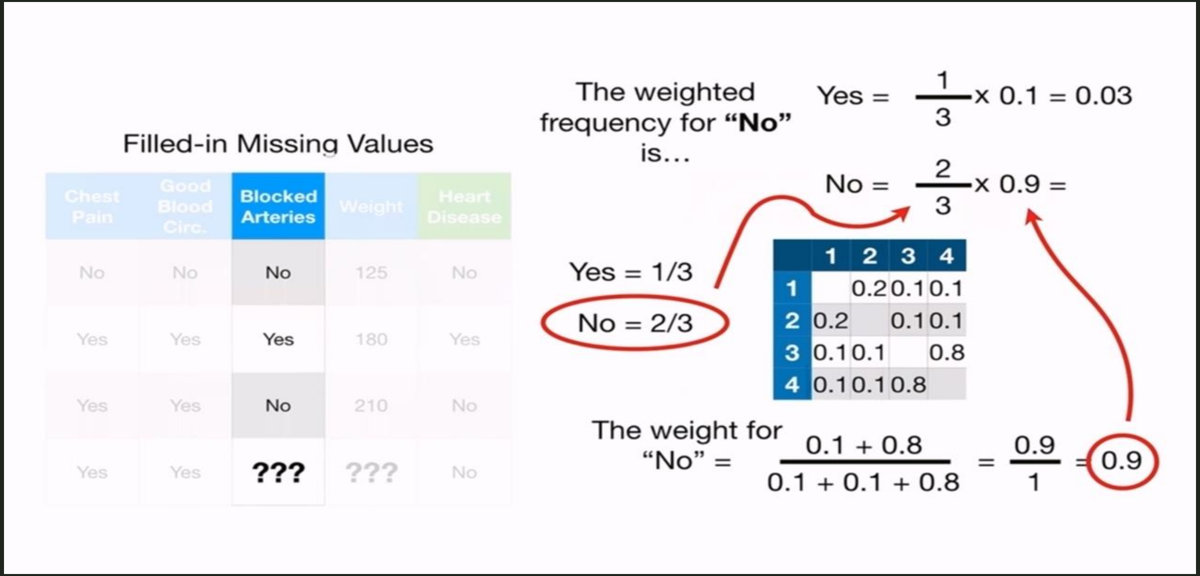
	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Samples 1 and 3 both have "No"...

Advantages

How does it Handle Missing values

Fifth, We use the proximity matrices to make better guesses



Filled-in Missing Values

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	Yes	???	???	No

The weighted frequency for **"No"** is...

$$\text{Yes} = \frac{1}{3} \times 0.1 = 0.03$$

$$\text{No} = \frac{2}{3} \times 0.9 =$$

$\text{Yes} = 1/3$

$\boxed{\text{No} = 2/3}$

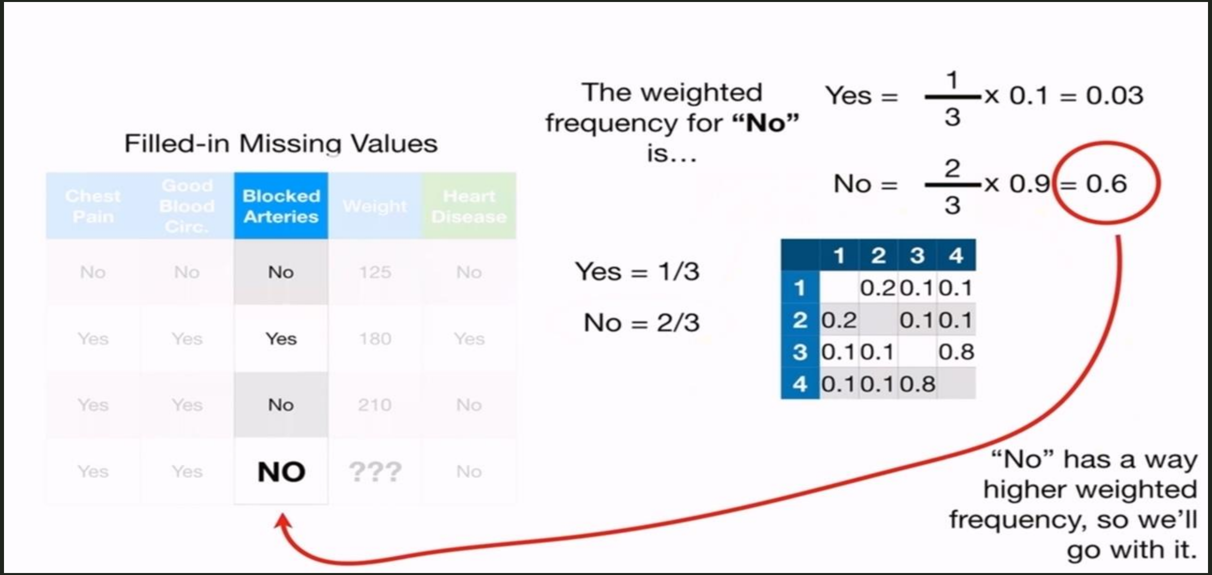
	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

$$\text{The weight for "No"} = \frac{0.1 + 0.8}{0.1 + 0.1 + 0.8} = \frac{0.9}{1} = \boxed{0.9}$$

Advantages

How does it Handle Missing values

Sixth, we choose the value with the higher frequency



Filled-in Missing Values

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	Yes	NO	???	No

The weighted frequency for **"No"** is...

Yes = 1/3

No = 2/3

$$\text{Yes} = \frac{1}{3} \times 0.1 = 0.03$$

$$\text{No} = \frac{2}{3} \times 0.9 = 0.6$$

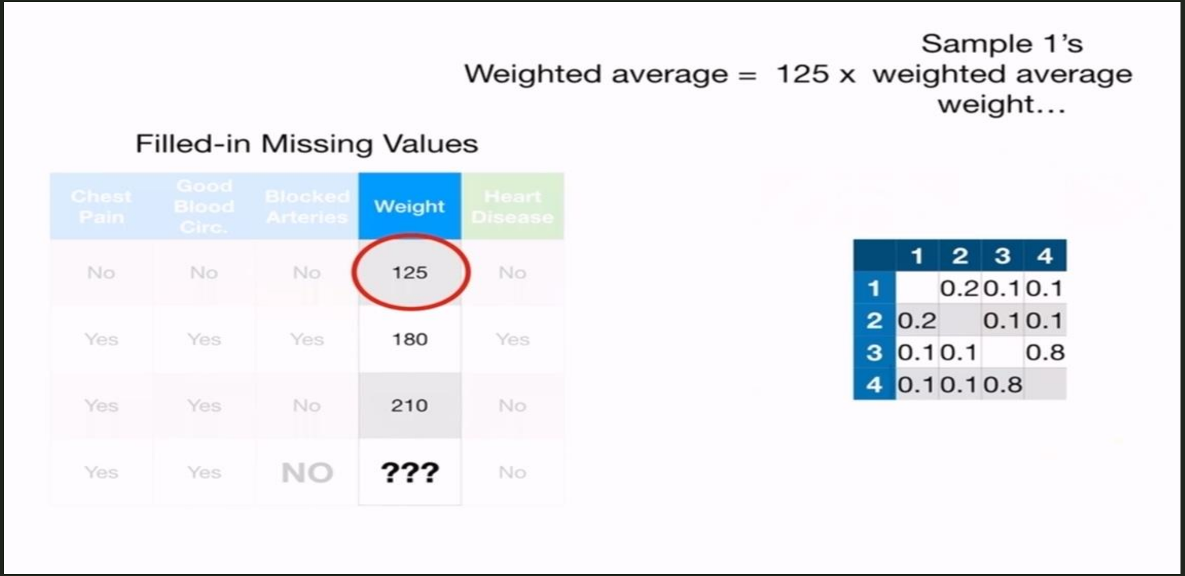
	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

"No" has a way higher weighted frequency, so we'll go with it.

Advantages

How does it Handle Missing values

Seventh, for numeric values we find the average



Weighted average = 125 x Sample 1's weighted average weight...

Filled-in Missing Values

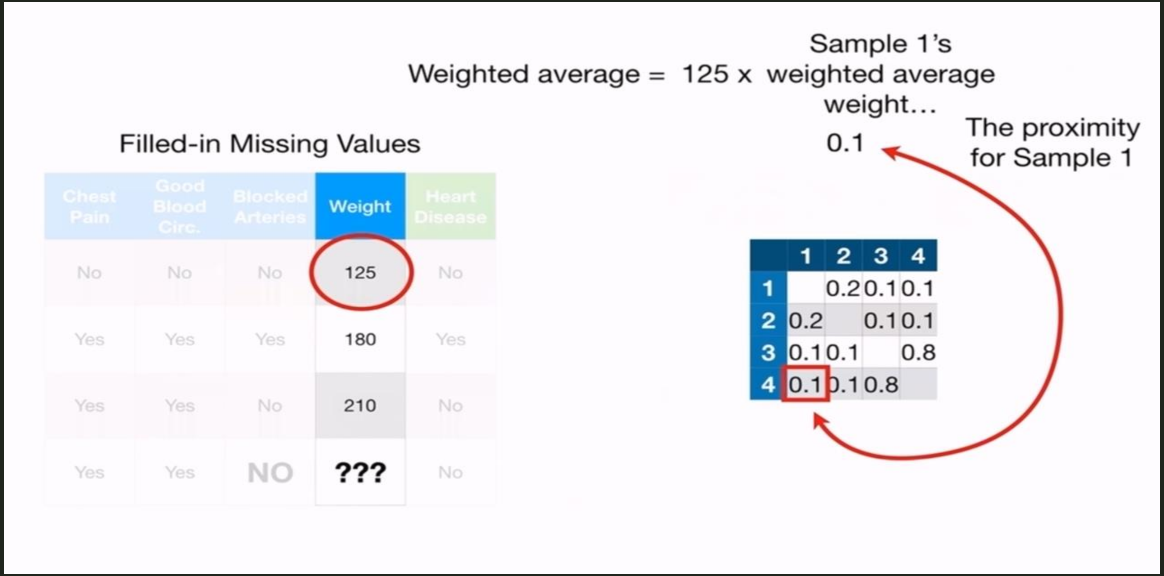
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	???	No

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Advantages

How does it Handle Missing values

Seventh, for numeric values we find the average



Weighted average = 125 x Sample 1's weighted average weight...

0.1

The proximity for Sample 1

Filled-in Missing Values

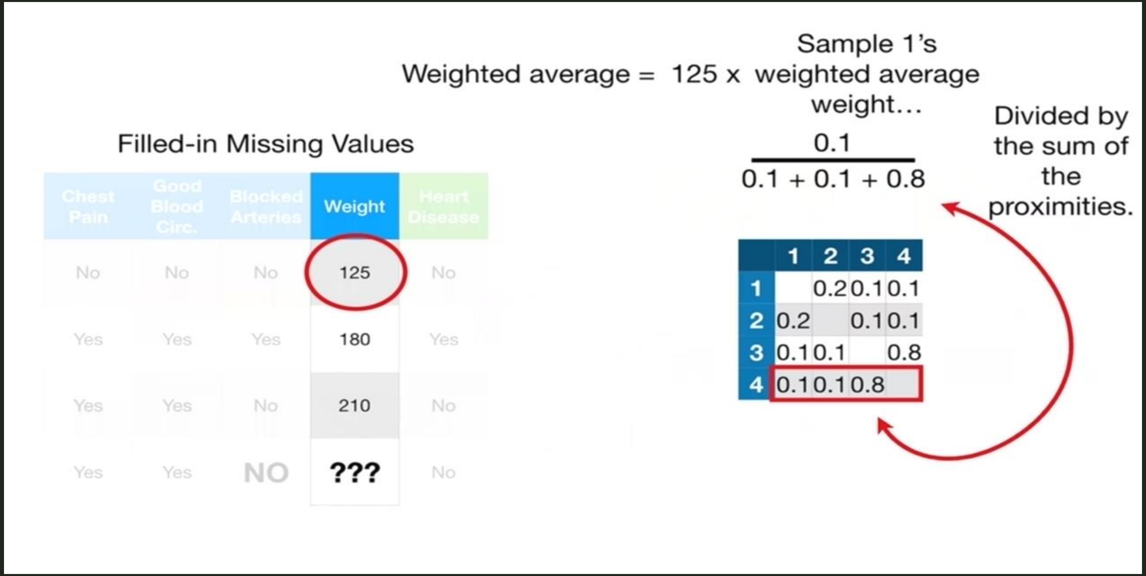
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	???	No

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Advantages

How does it Handle Missing values

Seventh, for numeric values we find the average



Weighted average = 125 x Sample 1's weighted average weight...

$$\frac{0.1}{0.1 + 0.1 + 0.8}$$

Divided by the sum of the proximities.

Filled-in Missing Values

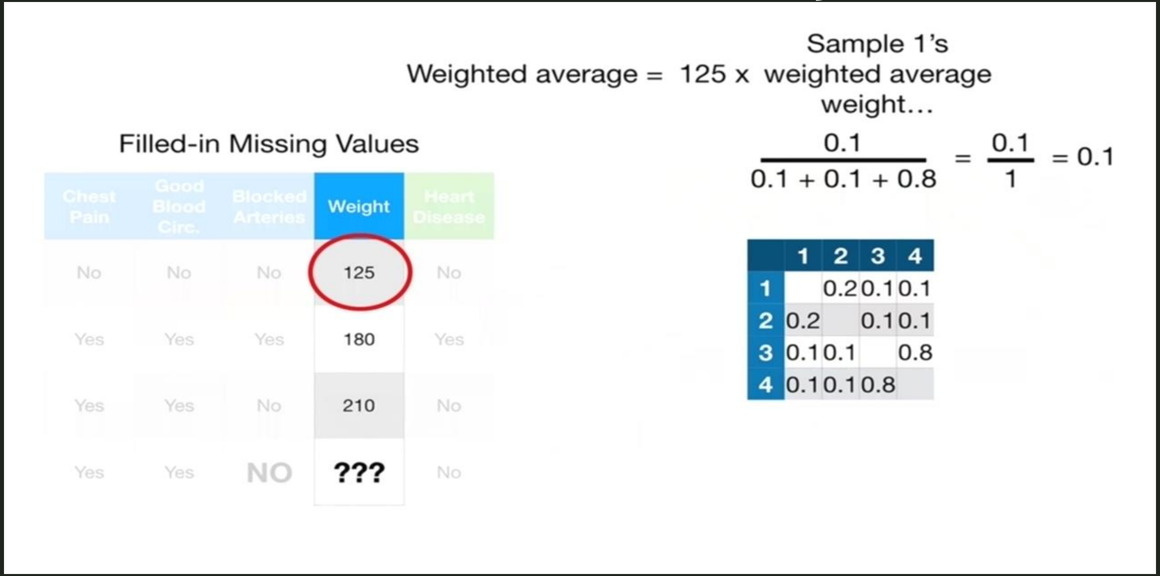
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	???	No

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Advantages

How does it Handle Missing values

Seventh, for numeric values we find the average



Weighted average = 125 x Sample 1's weighted average weight...

$$\frac{0.1}{0.1 + 0.1 + 0.8} = \frac{0.1}{1} = 0.1$$

Filled-in Missing Values

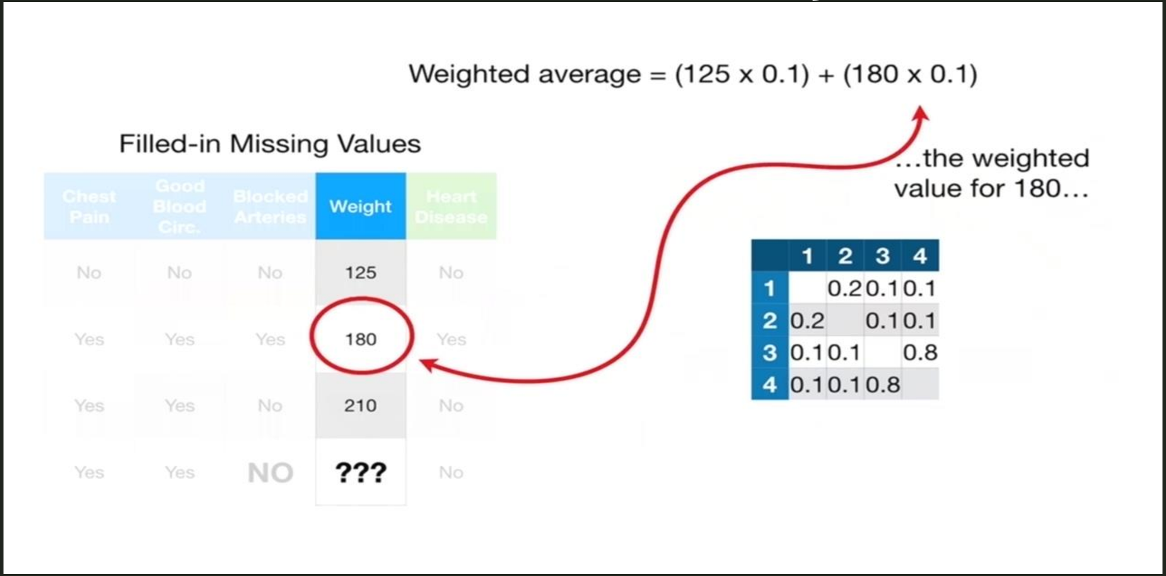
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	???	No

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Advantages

How does it Handle Missing values

Seventh, for numeric values we find the average



Weighted average = (125 x 0.1) + (180 x 0.1)

...the weighted value for 180...

Filled-in Missing Values

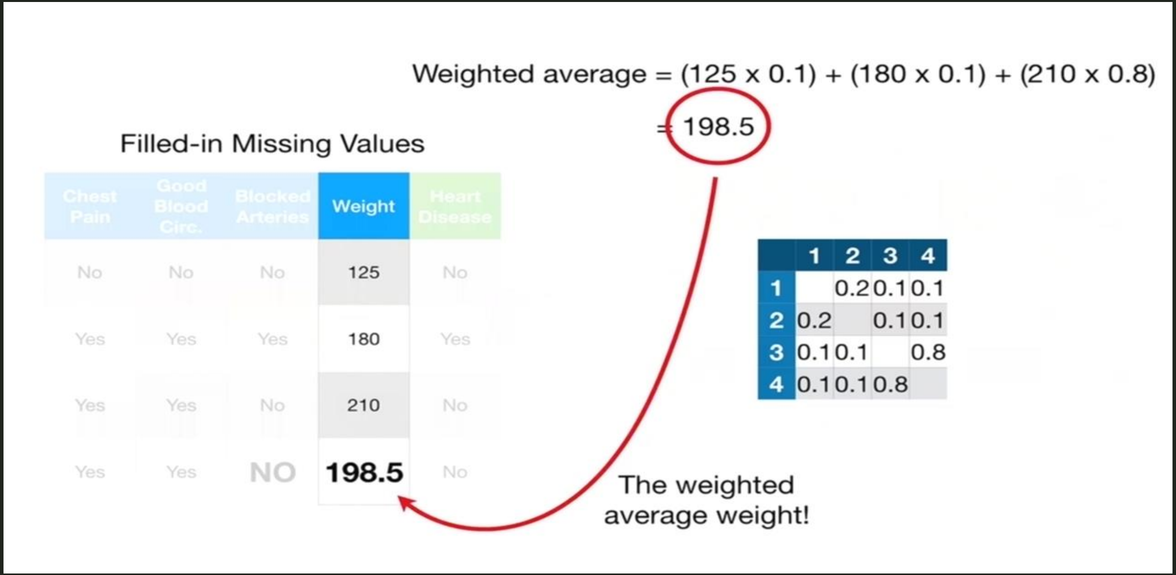
Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
Yes	Yes	No	210	No
Yes	Yes	NO	???	No

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

Advantages

How does it Handle Missing values

Seventh, for numeric values we find the average



Weighted average = (125 x 0.1) + (180 x 0.1) + (210 x 0.8)

= 198.5

Filled-in Missing Values

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	Yes	NO	**198.5**	No

	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		0.8
4	0.1	0.1	0.8	

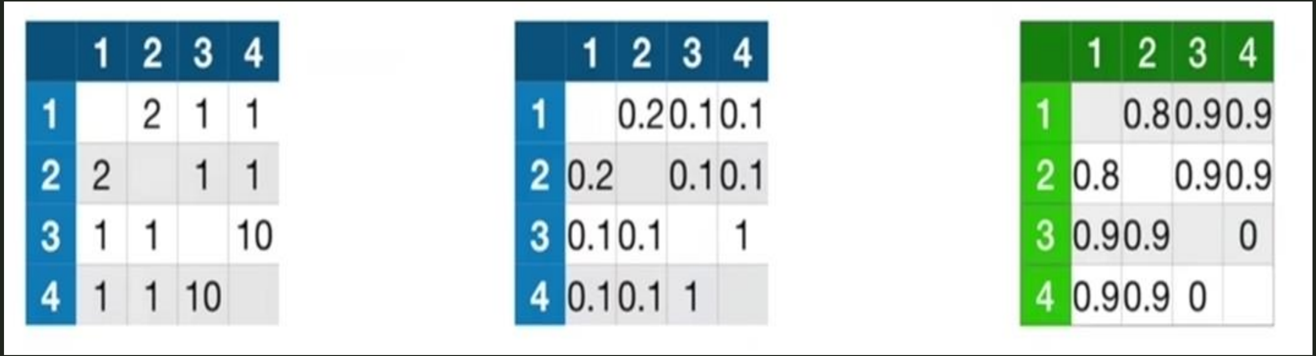
The weighted average weight!

Advantages

How does it Handle Missing values

Finally, We repeat this process till no values are converged

Note, we can use the proximity metrics to find a distance metrics



	1	2	3	4
1		2	1	1
2	2		1	1
3	1	1		10
4	1	1	10	

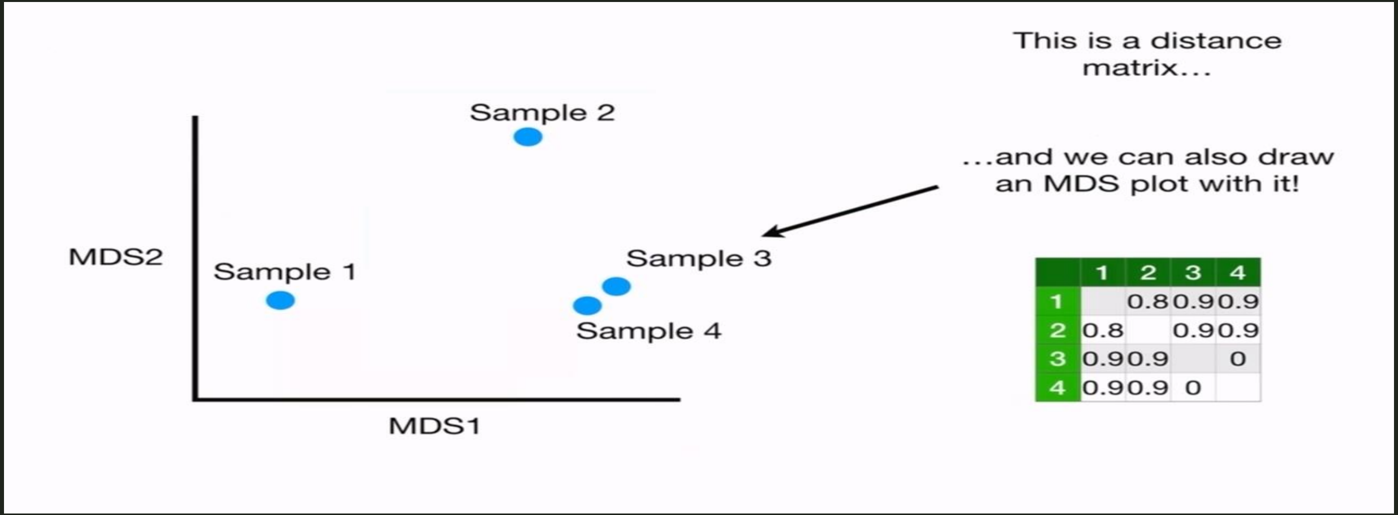
	1	2	3	4
1		0.2	0.1	0.1
2	0.2		0.1	0.1
3	0.1	0.1		1
4	0.1	0.1	1	

	1	2	3	4
1		0.8	0.9	0.9
2	0.8		0.9	0.9
3	0.9	0.9		0
4	0.9	0.9	0	

Advantages

Distance metrics

Which can be used to visualize the algorithm in a lot of charts



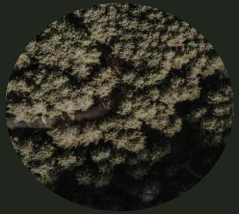
This is a distance matrix...

...and we can also draw an MDS plot with it!

	1	2	3	4
1		0.8	0.9	0.9
2	0.8		0.9	0.9
3	0.9	0.9		0
4	0.9	0.9	0	

Limitations

Limitations



What are the limitations of a Random Forest?

- **Interpretability**: Random Forests can be challenging to interpret, especially when the ensemble includes a large number of trees. Understanding the contribution of each feature to the overall prediction becomes complex, making it less intuitive than a single decision tree.

- **Computational Complexity**: Training a Random Forest with a large number of trees and deep trees can be computationally expensive and time-consuming. While the training process can be parallelized, it may still be slower than some other algorithms for large datasets. **Why?** because large number of decision-trees are used to make predictions. All the trees in the forest have to make a prediction for the same input and then perform voting on it. So, it is a time-consuming process.

- **Memory Usage**: Random Forests, especially with a large number of trees, can consume significant memory. Storing and maintaining multiple decision trees may be resource-intensive, limiting their applicability in memory-constrained environments.

Ali Ahmed Yousef 21_16_0038
Ahmed Maher Al-Maqtari 21_16_0049
Muhannad Al-Mozaiger 21_16_0067

Thanks!