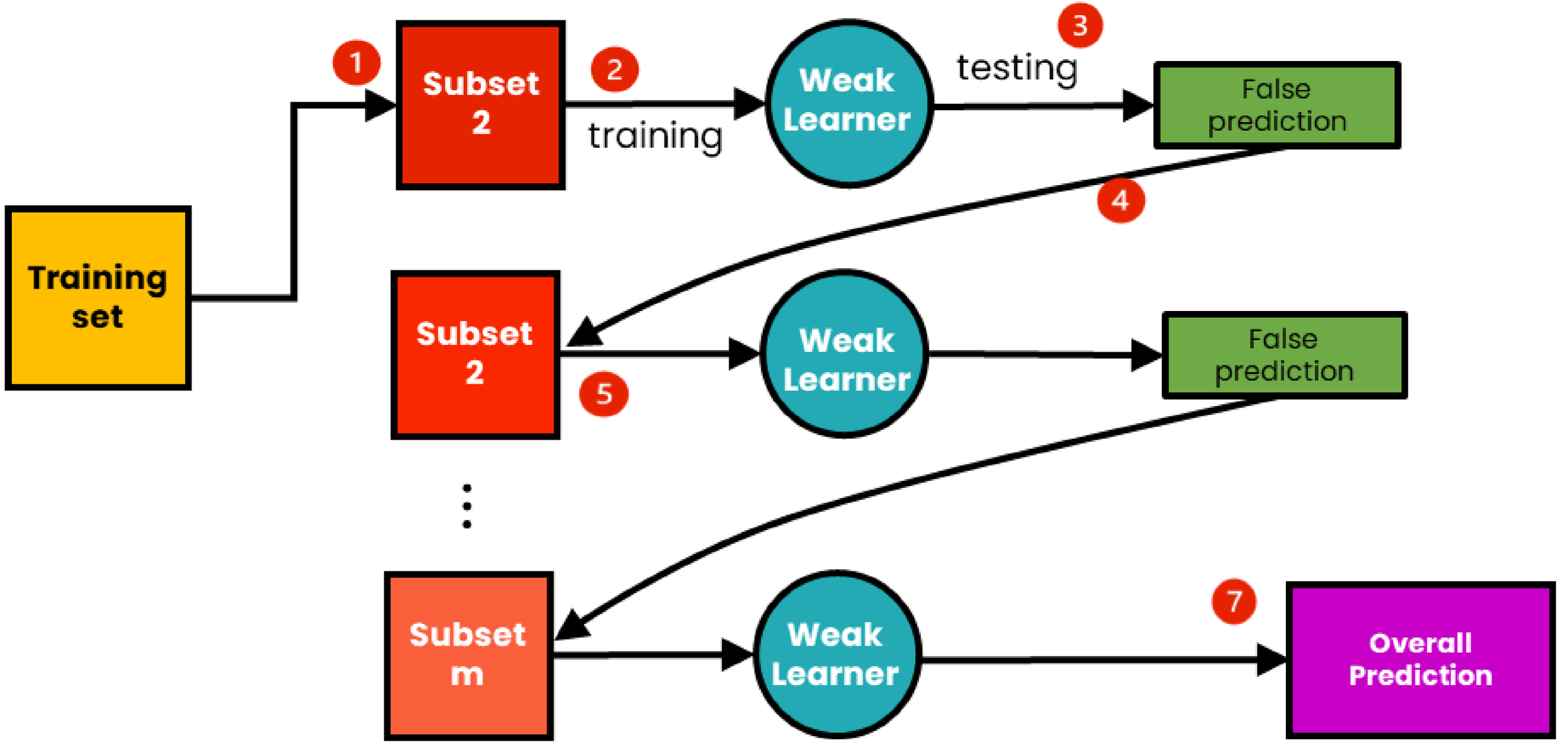
UNIT - III

ENSEMBLE TECHNIQUES & UNSUPERVISED LEARNING

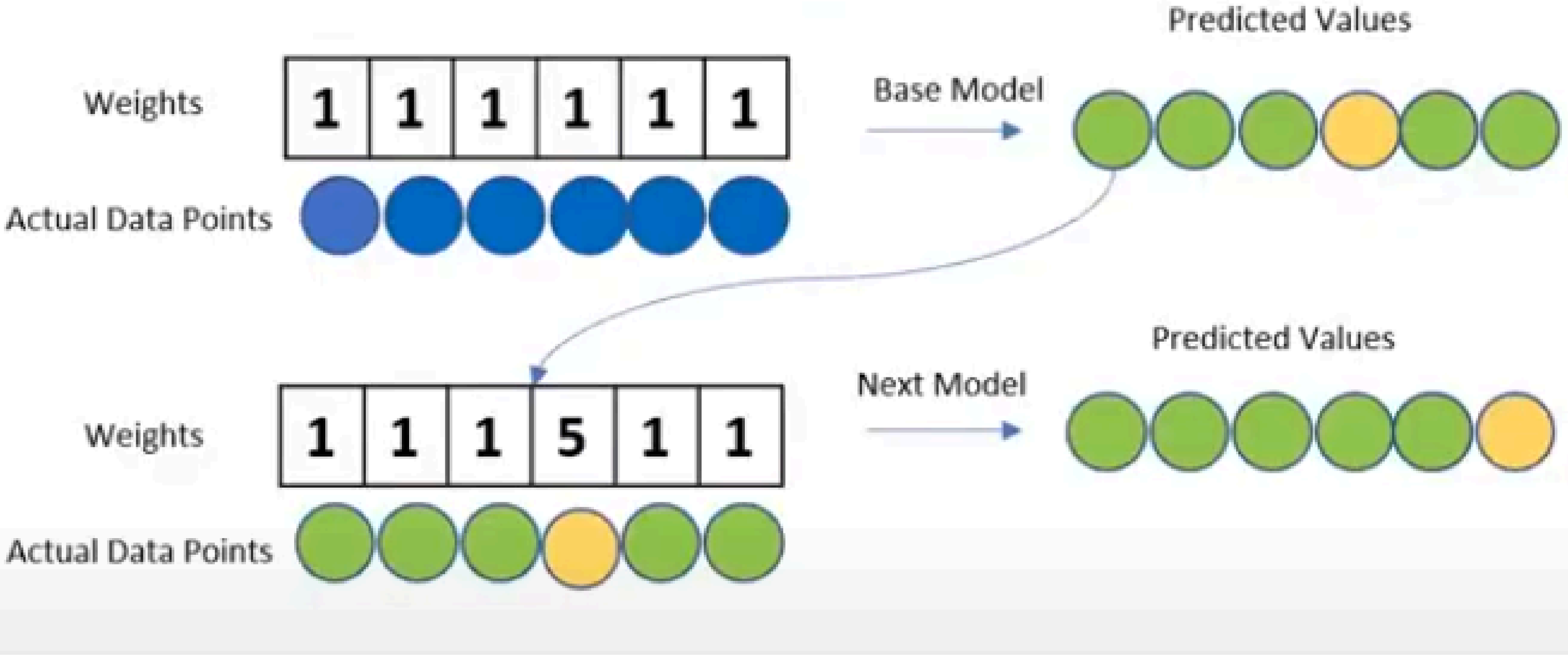
UNIT III ENSEMBLE TECHNIQUES & UNSUPERVISED LEARNING 9

Combining multiple learners: Model combination schemes, Voting, Ensemble Learning - bagging, boosting, stacking, Unsupervised learning: K-means, Instance-Based Learning: KNN, Gaussian mixture models and Expectation maximization

The Process of Boosting



- Boosting creates an ensemble model by combining several weak learners sequentially
- It assigns equal weight to each data sample. It feeds the data to the first machine model, called the base algorithm. The base algorithm makes predictions for each data sample
- Next, it assesses model predictions and increases the weight of samples with a more significant error
- The algorithm passes the weighted data to the next model
- The algorithm repeats steps 2 and 3 until instances of training errors are below a certain threshold


Predicted Values
Weights
1 1 1 1 1 1
Base Model
Actual Data Points
Predicted Values
Next Model
Weights
1 1 1 5 1 1
Actual Data Points

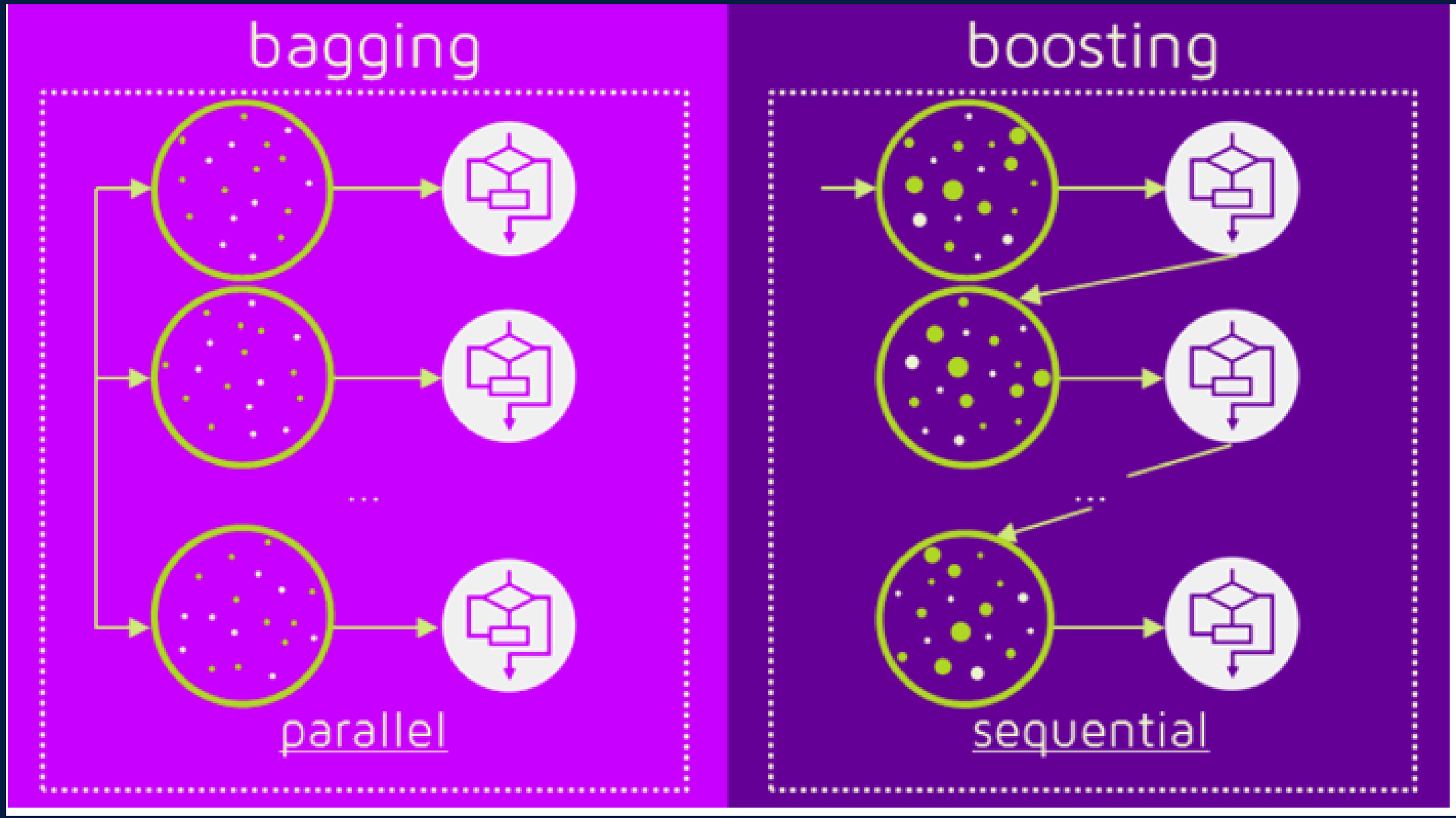
Bagging vs Boosting

Bagging

- All weak learners are built in parallel
- Equal weight to each weak learner
- Independent samples
- Can help reduce the variance of the model
- Example: Bagging Classifier, Random Forest

Boosting

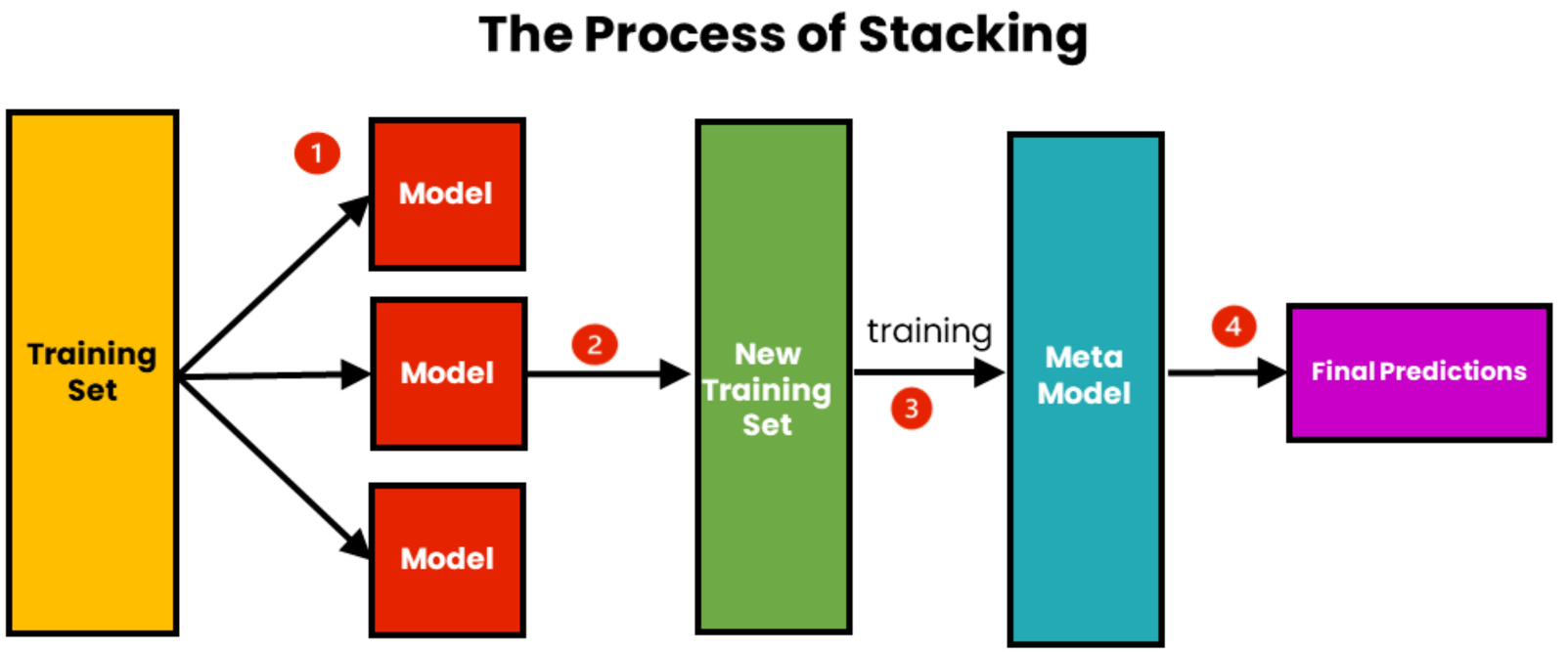
- Successive learners to improve accuracy from the prior weak learners
- More weight to those weak learners with better performance
- Subsequent samples have more of those observations which had relatively higher errors in previous weak learner
- Can help reduce the bias of the model
- Example: AdaBoost, Gradient Boosting Classifier


bagging
boosting
parallel
sequential

STACKING

STACKING

- Stacking is an ensemble learning technique that uses predictions from multiple models (for example decision tree, knn or svm) to build a new model.
- It is also known as stacked ensembles or stacked generalization.
- It entails training numerous base models on the same training dataset, then feeding their predictions into a higher-level model, also known as a **meta-model** or second-level model, to make the final prediction.


The Process of Stacking
Training Set
1
Model
Model
Model
2
New Training Set
training
3
Meta Model
4
Final Predictions

How Stacking works?

Preparing the data

Model Selection

Training the base models

Predictions on the validation set

Developing a Meta Model

Training the Meta Model

Making Test Set Predictions

Model Evaluation

Step 1: Split the training dataset into two parts:

Dataset	Description
Training Data	Used to train the base models
Hold-Out Validation Data	Used to train the meta-model

Step 2: Train several base models on the training data:

Model	Algorithm
Base Model 1	Decision Tree
Base Model 2	Neural Network
Base Model 3	Support Vector Machine

Step 3: Make predictions using the base models on the hold-out validation data:

Model	Base Model 1	Base Model 2	Base Model 3
Prediction	Prediction 1	Prediction 2	Prediction 3

Step 4: Train the meta-model on the hold-out validation data using the predictions from the base models as input features:

Model	Algorithm
Meta-Model	Logistic Regression

Step 5: To make a prediction for new data:

Model	Algorithm
Base Model 1	Decision Tree
Base Model 2	Neural Network
Base Model 3	Support Vector Machine
Meta-Model	Logistic Regression

To make a prediction for new data

Final step: Evaluate the performance of the stacked model on a separate test dataset that was not used during training.

Advantages of Stacking

Improved Predictive Performance

Interpretability

Model Diversity

Flexibility

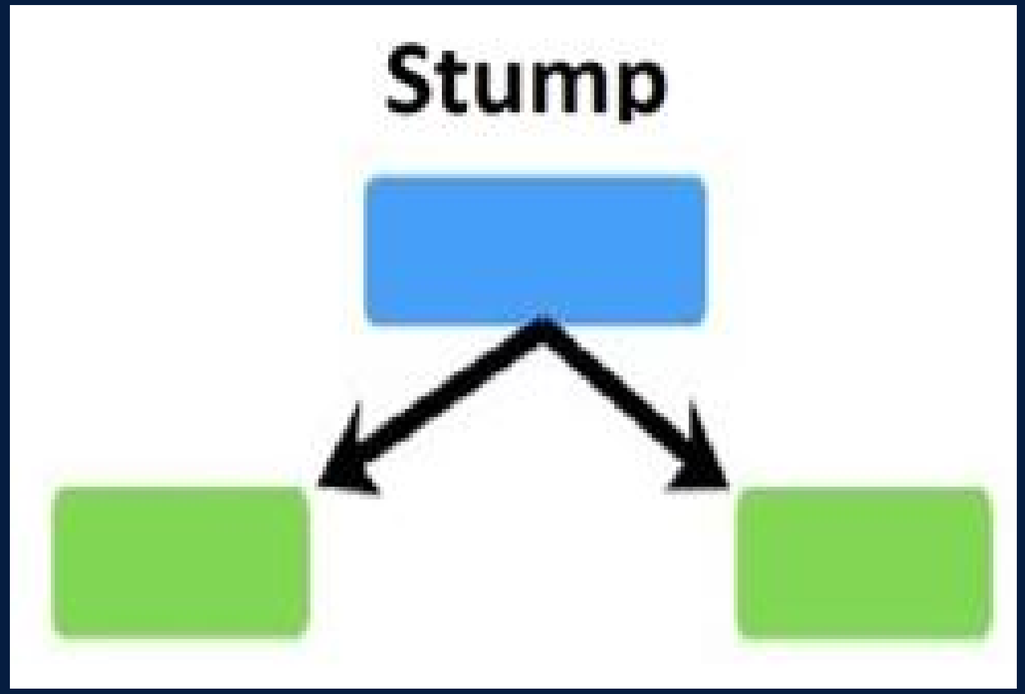
AdaBoost

AdaBoost

- AdaBoost is a **boosting technique** and it is a **sequential learning process**.
- In the sequential process, one tree is dependent on the previous tree which means if there are multiple models implemented say ML model 1, ML model 2, ML model 3, and so on.
- Each of them has a process of ensembling then ML **model 2** will depend on the output of ML **model 1** and similarly ML **model 3** will depend on the output of ML **model 2**.

AdaBoost

- In Adaboost, the trees are not fully grown.
- Rather the trees are just one root and two leaves.
- They are called **stumps** in the language of Adaboost.
- Stumps are nothing but **one root node** and **two leaf nodes**.



AGE	BMI	GENDER
25	24	F
41	31	F
56	28	M
78	26	F
62	30	M

- The very first thing it does is, it will assign a weight to all these records called initial weights.
- The initial weights would be a sum equal to 1.

AGE	BMI	GENDER	INITIAL WEIGHTS
25	24	F	1/5
41	31	F	1/5
56	28	M	1/5
78	26	F	1/5
62	30	M	1/5

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No

First, we'll start with some data.

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No

We create a **Forest of Stumps** with **AdaBoost** to predict if a patient has heart disease.

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
Yes	Yes	205	Yes
No	Yes	180	Yes
Yes	No	210	Yes
Yes	Yes	167	Yes
No	Yes	156	No
No	Yes	125	No
Yes	No	168	No
Yes	Yes	172	No

We will make these predictions based on a patient's **Chest Pain** and **Blocked Artery** status and their **Weight**.

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	
No	Yes	180	Yes	
Yes	No	210	Yes	
Yes	Yes	167	Yes	
No	Yes	156	No	
No	Yes	125	No	
Yes	No	168	No	
Yes	Yes	172	No	

The first thing we do is give each sample a weight that indicates how important it is to be correctly classified.

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	
No	Yes	180	Yes	
Yes	No	210	Yes	
Yes	Yes	167	Yes	
No	Yes	156	No	
No	Yes	125	No	
Yes	No	168	No	
Yes	Yes	172	No	

NOTE: The **Sample Weight** is different from the **Patient Weight**, and I'l do the best I can to be clear about which of the two I'm talking about.

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

At the start, all samples get the same weight…

$$\frac{1}{\text{total number of samples}} = \frac{1}{8}$$

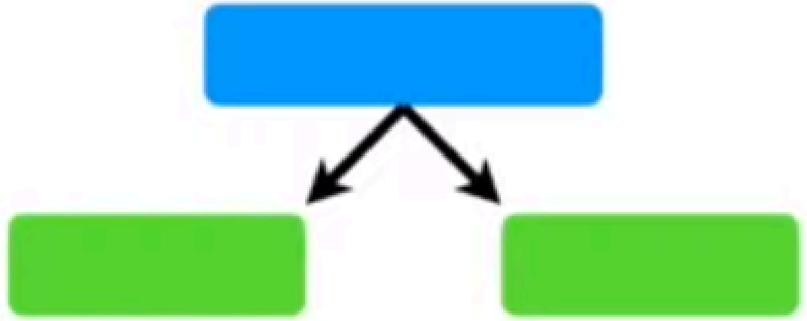
…and that makes the samples all equally important.

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

However, after we make the first stump, these weights will change in order to guide how the next stump is created.

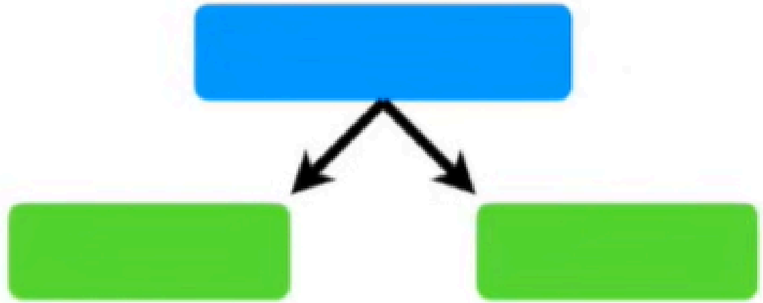
Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

Now we need to make the first **stump** in the forest.



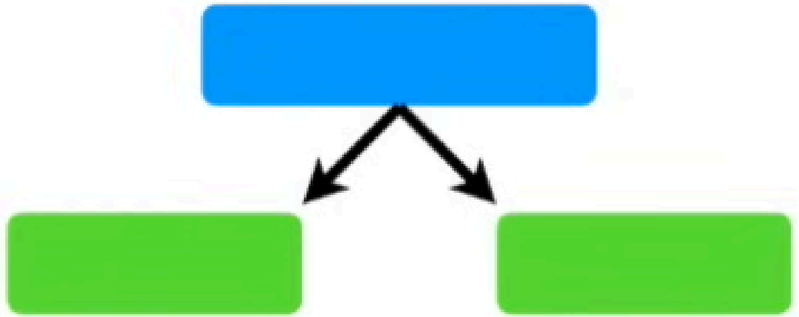
Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

This is done finding the variable, **Chest Pain**, **Blocked Arteries** or **Patient Weight**, that does the best job classifying the samples.



Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

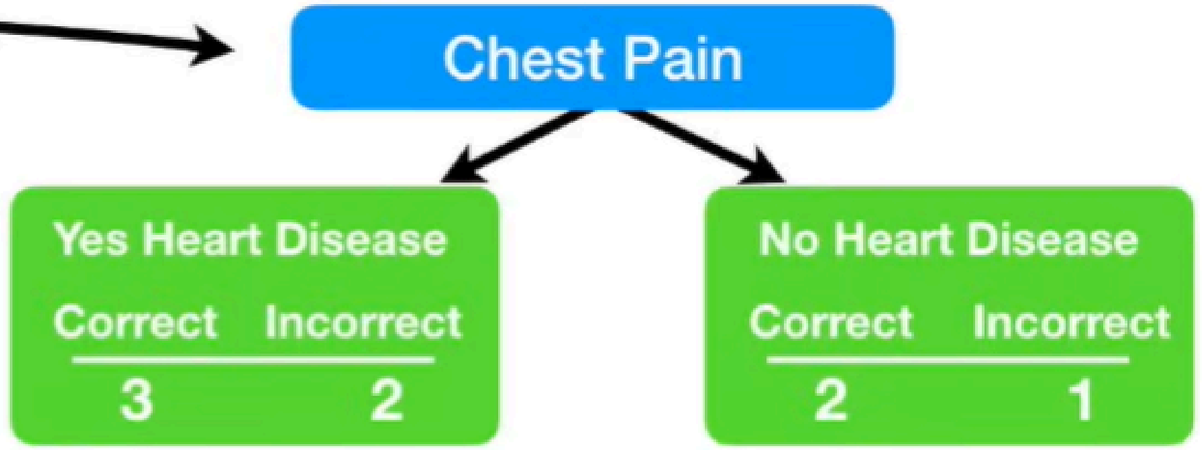
NOTE: Because all of the weights are the same, we can ignore them right now.



Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

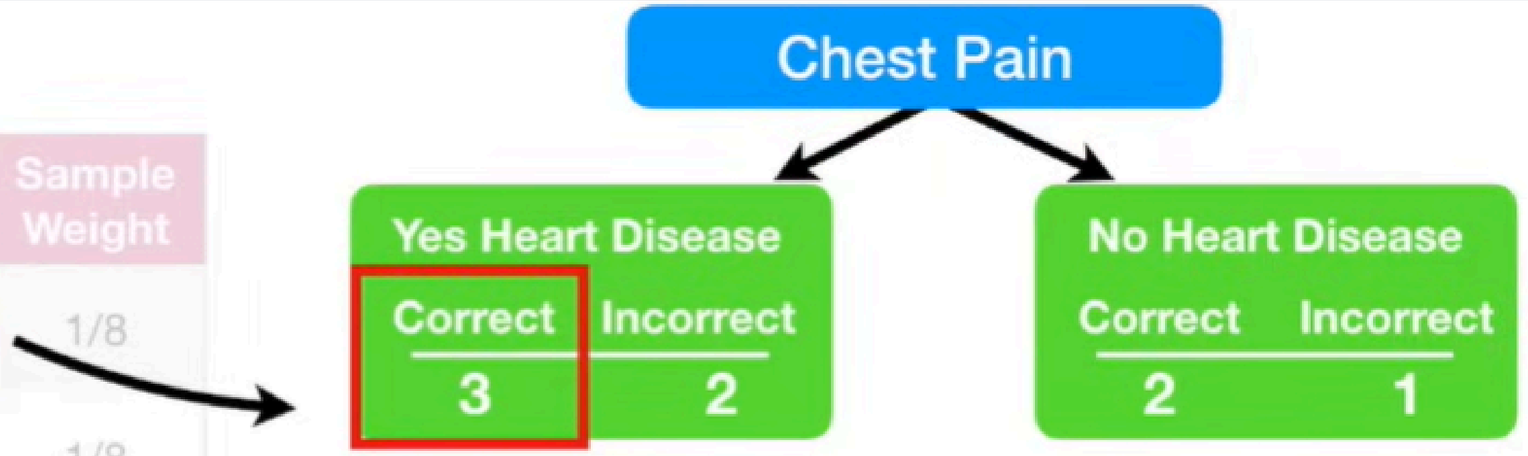
We start by seeing how well **Chest Pain** classifies the samples.

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8



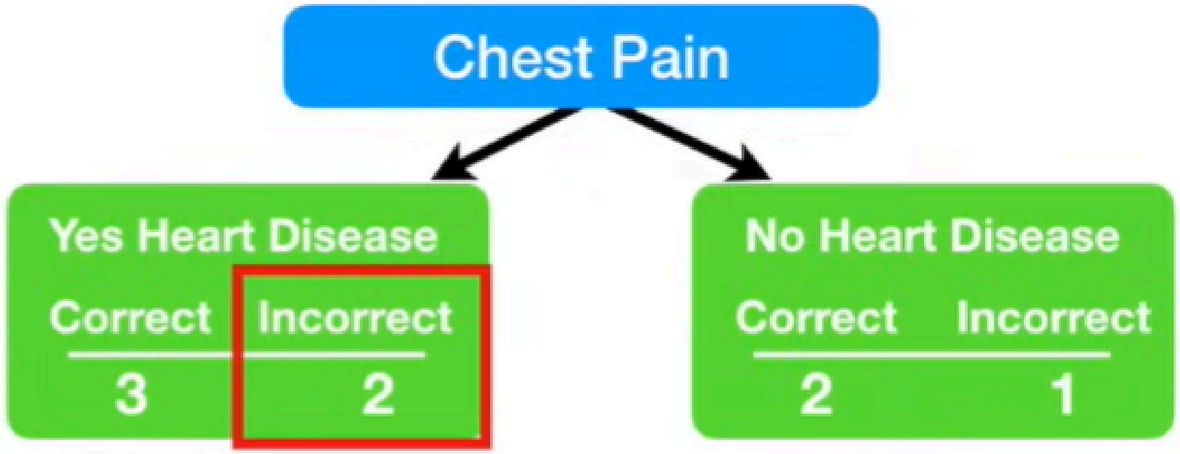
We start by seeing how well **Chest Pain** classifies the samples.

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8



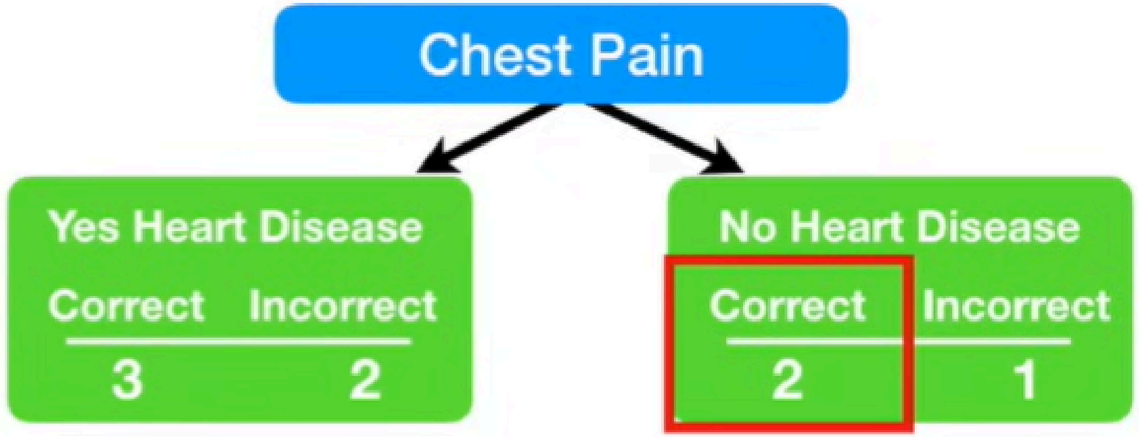
Of the 5 samples with **Chest Pain**, 3 were *correctly* classified as having **Heart Disease**…

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8



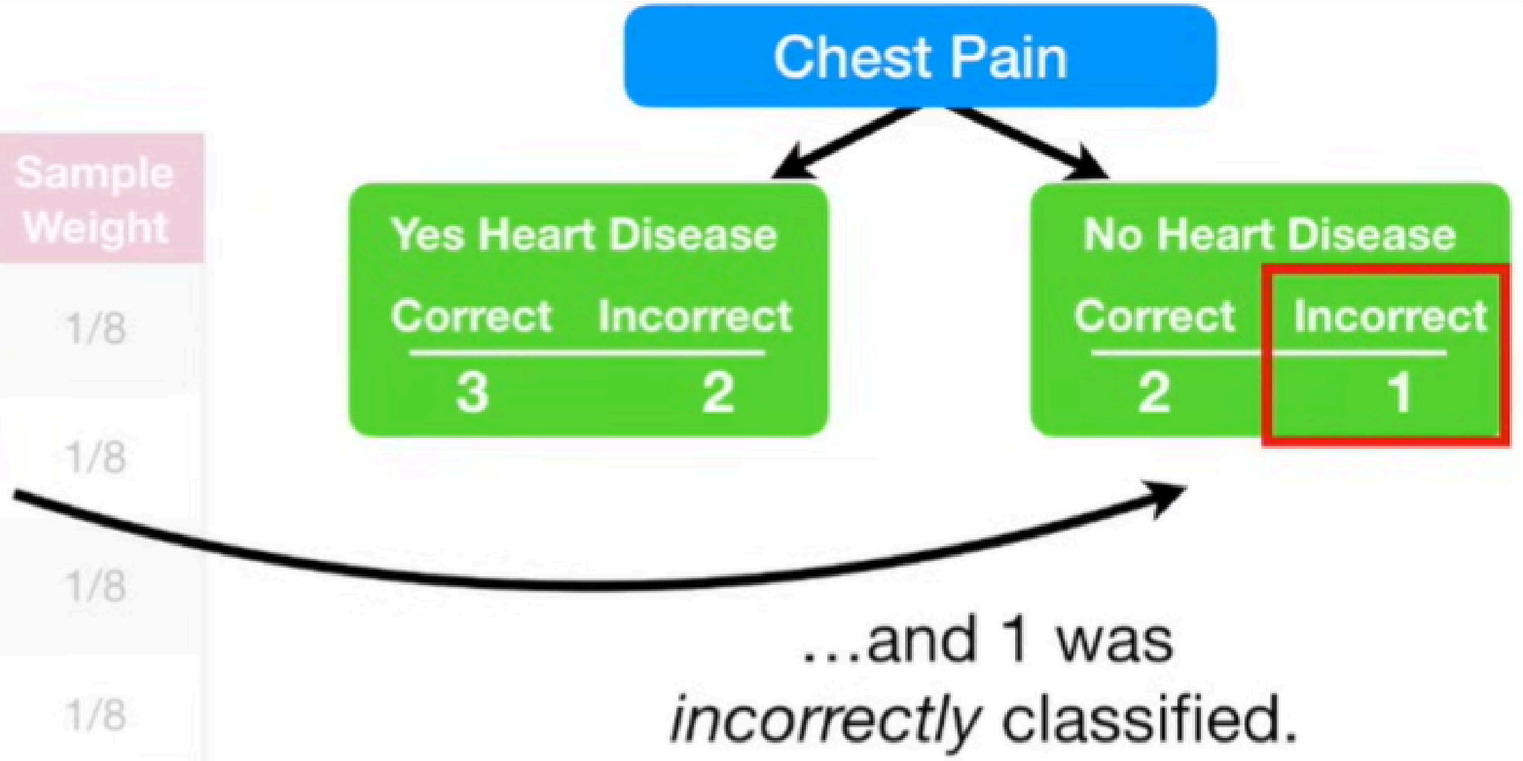
…and 2 were *incorrectly* classified.

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8



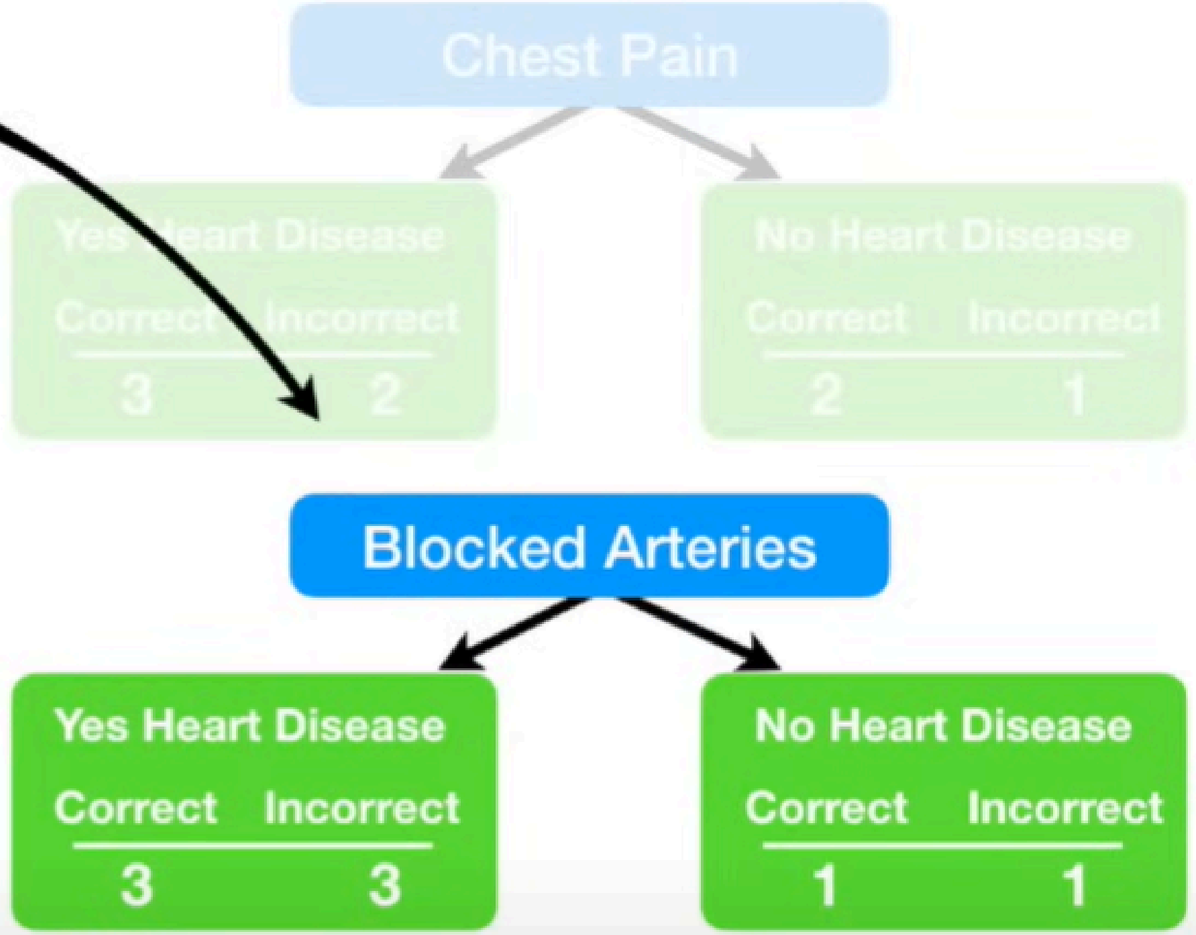
Of the 3 samples *without* **Chest Pain**, 2 were *correctly* classified as *not* having **Heart Disease**…

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8



…and 1 was *incorrectly* classified.

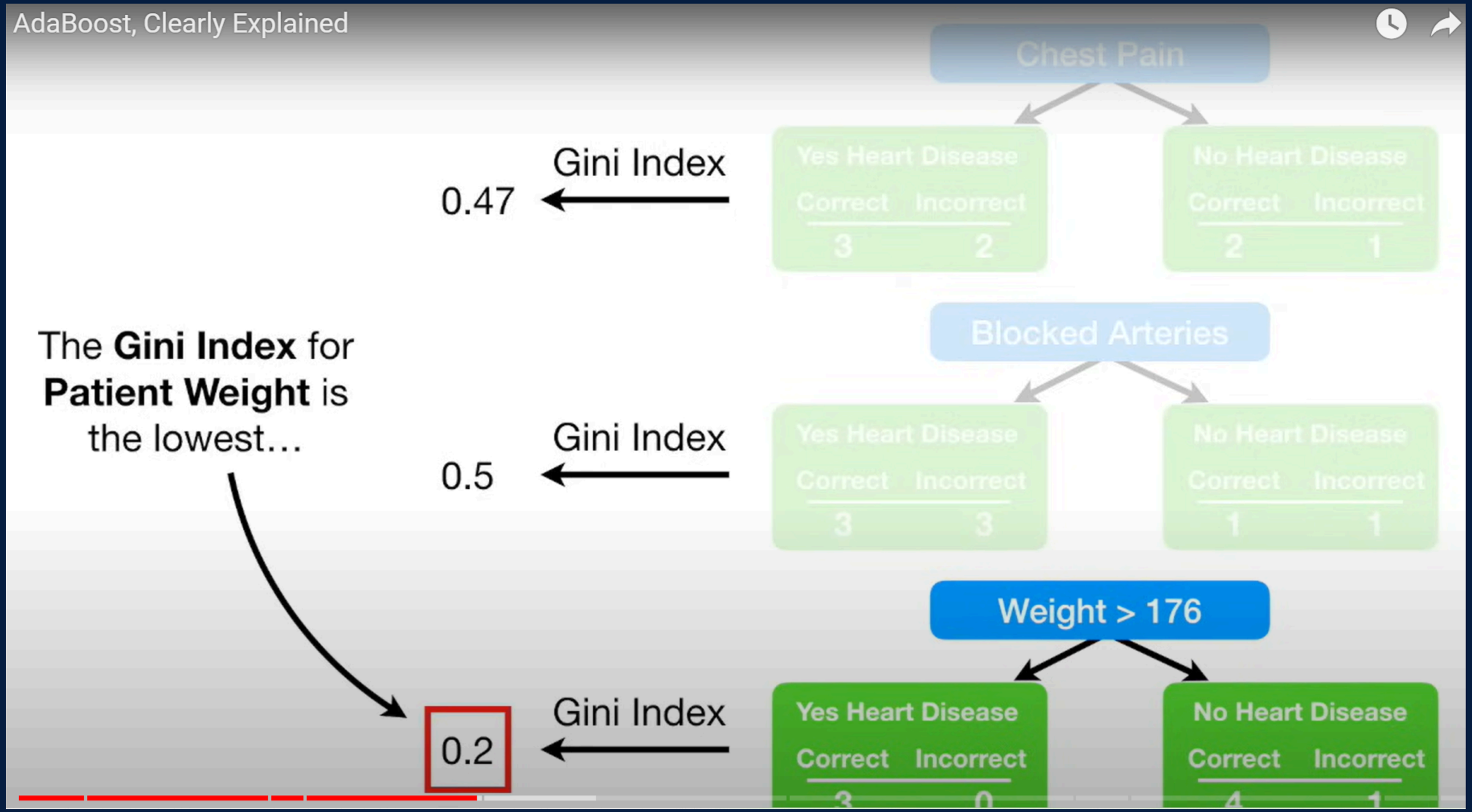
Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8



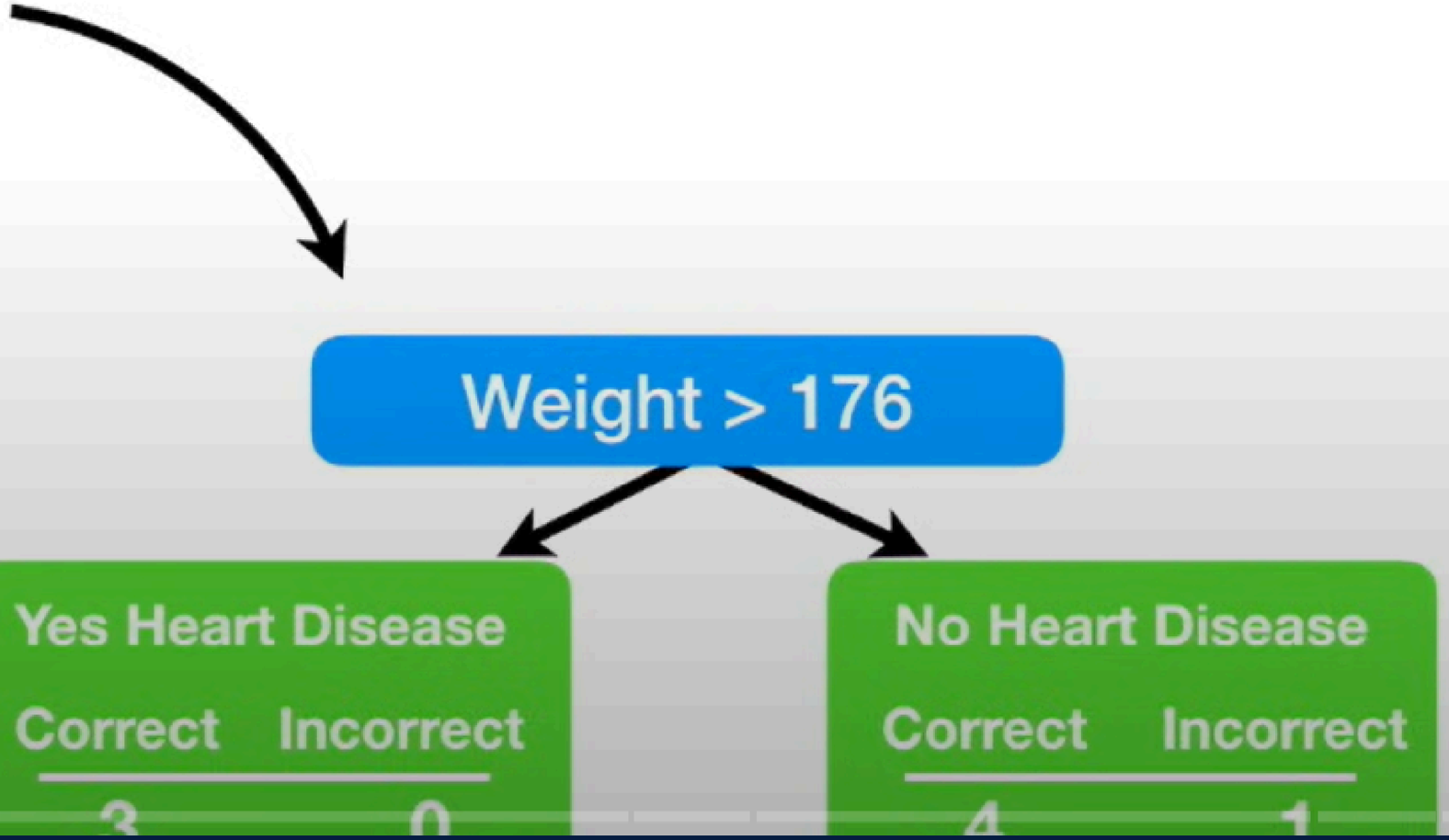
Now we do the same thing for **Blocked Arteries**…

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

NOTE: We used the techniques described in the **Decision Tree StatQuest** to determine that **176** was the best weight to separate the patients.


AdaBoost, Clearly Explained
Chest Pain
Yes Heart Disease
Correct Incorrect
3 2
No Heart Disease
Correct Incorrect
2 1
Gini Index
0.47
Blocked Arteries
Yes Heart Disease
Correct Incorrect
3 3
No Heart Disease
Correct Incorrect
1 1
Gini Index
0.5
The Gini Index for Patient Weight is the lowest…
Weight > 176
Yes Heart Disease
Correct Incorrect
3 0
No Heart Disease
Correct Incorrect
4 1
Gini Index
0.2

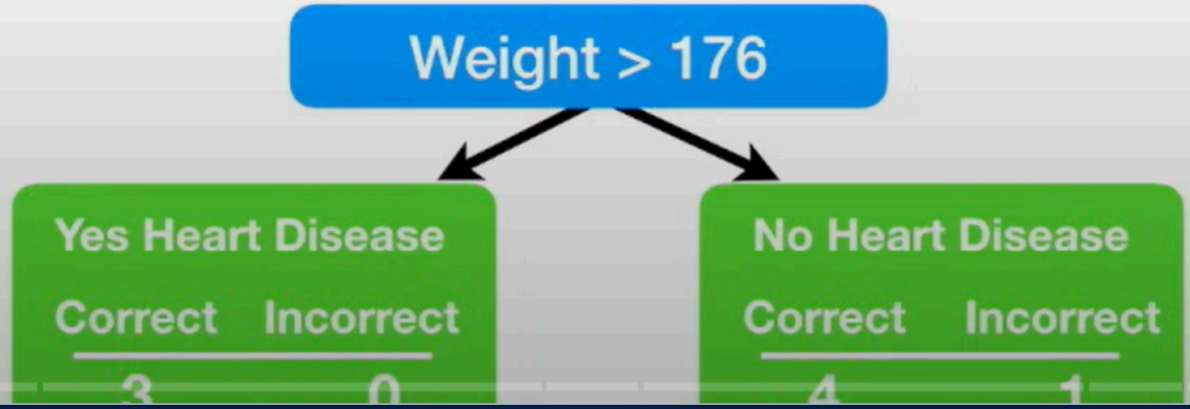
…so this will be the first stump in the forest.



We determine how much say a stump has in the final classification based on how well it classified the samples.

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

This patient, who weighs less than 176, *has* heart disease, but the stump says they do not.



Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

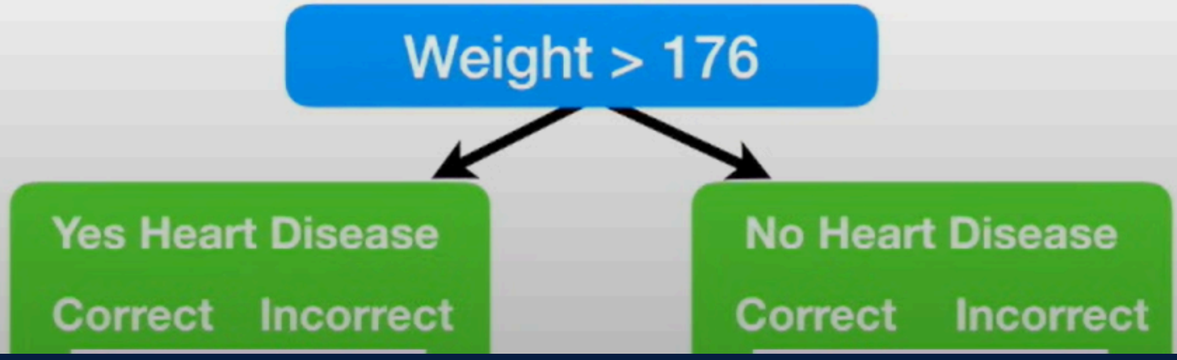
The **Total Error** for a stump is the sum of the weights associated with the *incorrectly* classified samples.

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

Thus, in this case, the **Total Error** is **1/8**.

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

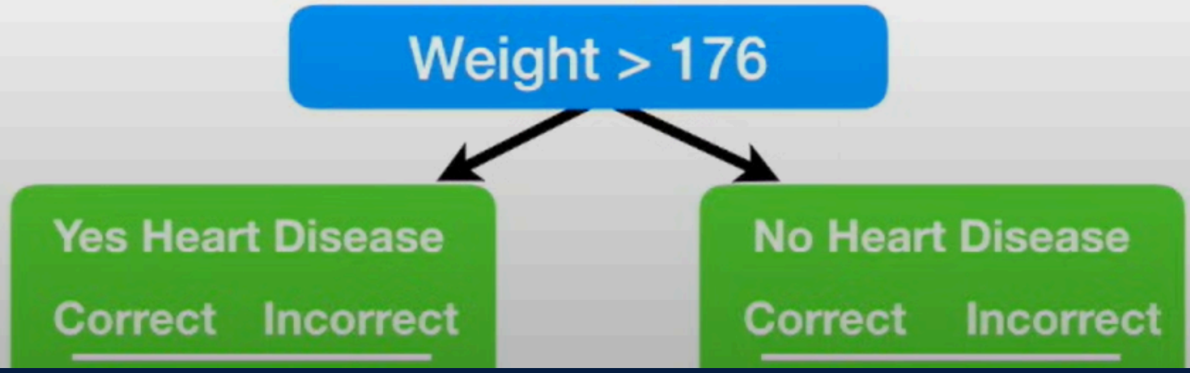
NOTE: Because all of the **Sample Weights** add up to **1**, **Total Error** will always be between **0**, for a perfect stump, and **1**, for a horrible stump.

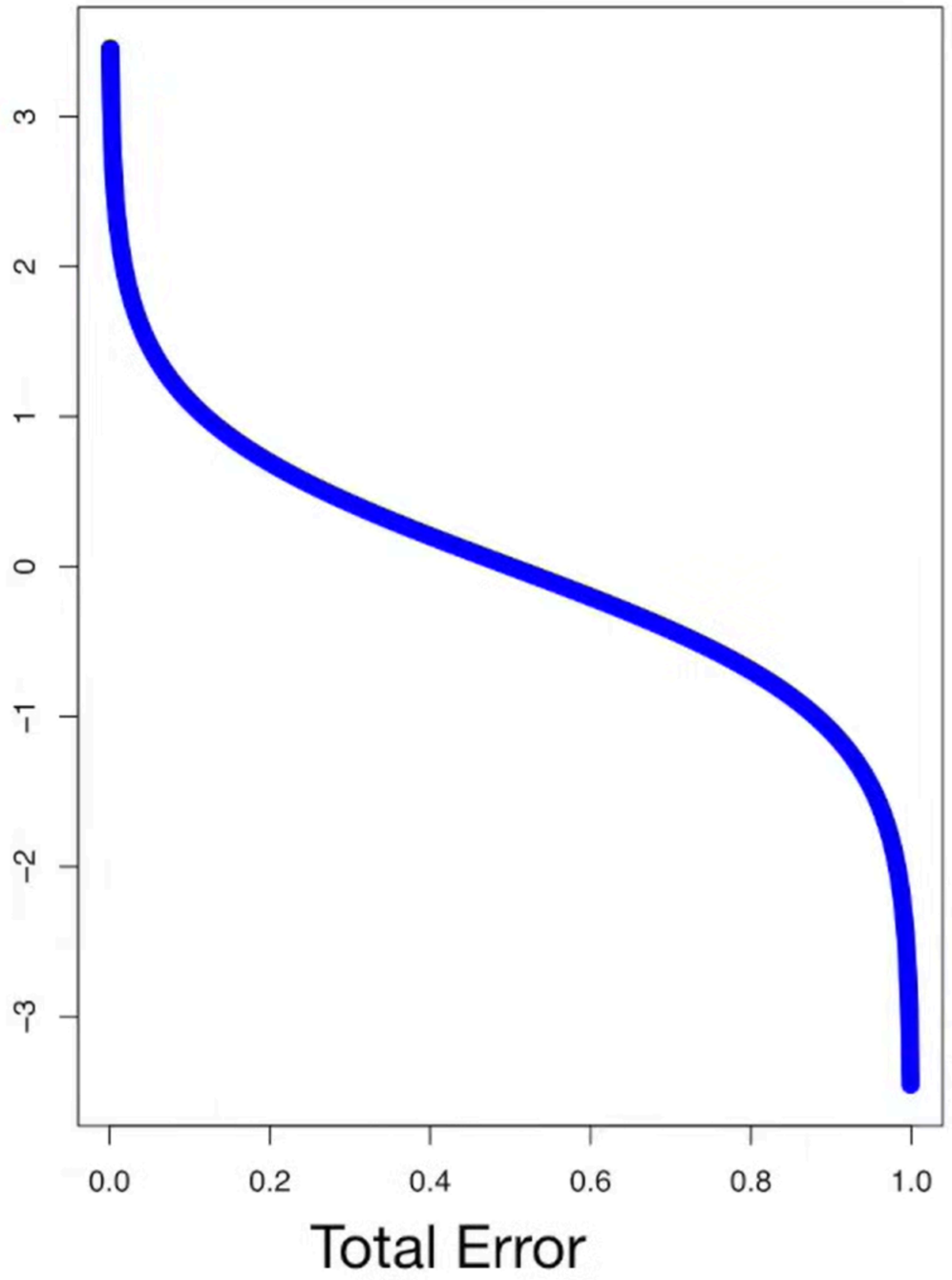


Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

We use the **Total Error** to determine **Amount of Say** this stump has in the final classification with the following formula:

$$\text{Amount of Say} = \frac{1}{2}\log\left(\frac{1 - \text{Total Error}}{\text{Total Error}}\right)$$





We can draw a graph of the **Amount of Say** by plugging in a bunch of numbers between **0** and **1** for **Total Error**.

$$\text{Amount of Say} = \frac{1}{2}\log\left(\frac{1 - \text{Total Error}}{\text{Total Error}}\right)$$

Advantages

- It is less prone to overfitting
- Less parameters tweaking
- Helps in reducing bias and variance
- Accuracy of weak classifiers can be improved using this method
- Easy to use

Disadvantages

- It needs a quality dataset
- Very sensitive to outliers and noise
- Slower than XGBoost
- Hyperparameter optimization is much more difficult