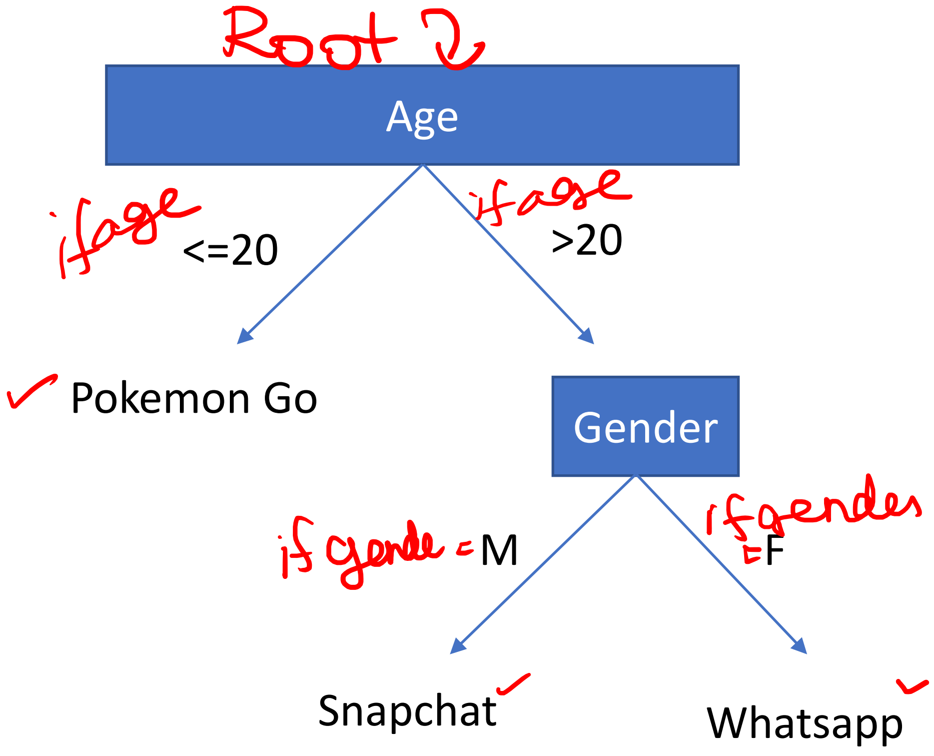
Decision Tree

What is a Decision Tree?

Recommending Apps

Gender	Age	App
F	15	Pokemon Go
F	25	Whatsapp
M	32	Snapchat
F	40	Whatsapp
M	12	Pokemon Go
M	14	Pokemon Go



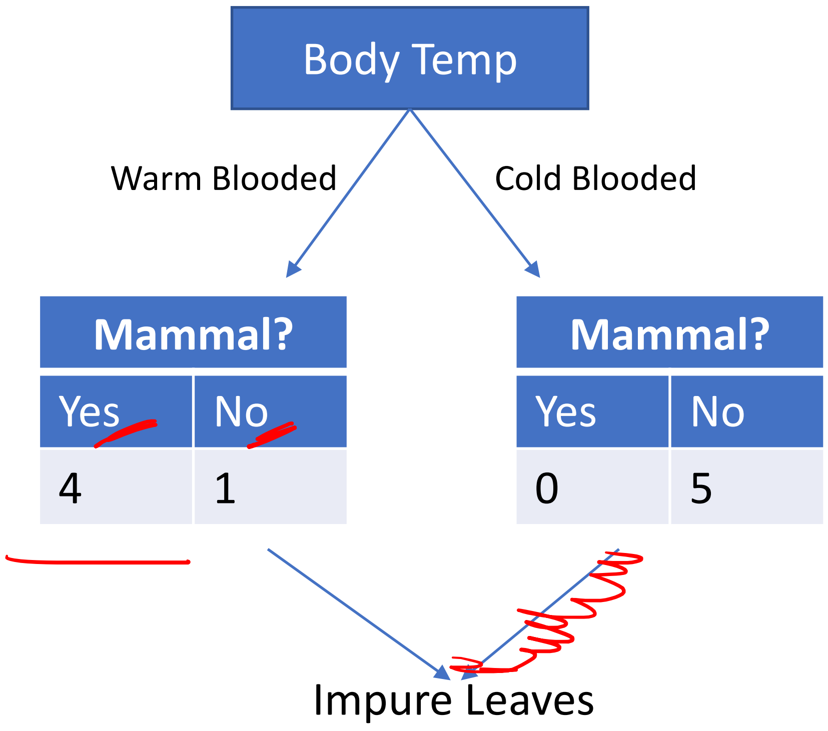
Let's take the toy dataset

Independent / Predictors ? dependent

Body Temp	Gives Birth	Four-legged	Hibernates	Mammal?
Warm blooded	Yes	No	No	Yes
Warm blooded	No	No	No	No
Warm blooded	Yes	Yes	No	Yes
Cold Blooded	Yes	No	No	No
Cold Blooded	No	Yes	No	No
Cold Blooded	No	No	No	No
Cold Blooded	No	No	No	No
Warm blooded	Yes	No	No	Yes
Warm blooded	No	Yes	Yes	Yes
Cold blooded	No	Yes	Yes	No

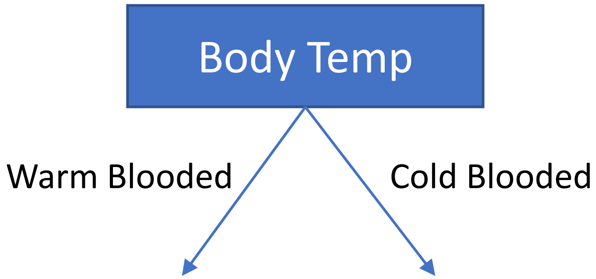
Let's take the toy dataset

Body Temp	Gives Birth	Four-legged	Hibernates	Mammal?
Warm blooded	Yes	No	No	Yes
Warm blooded	No	No	No	No
Warm blooded	Yes	Yes	No	Yes
Cold Blooded	Yes	No	No	No
Cold Blooded	No	Yes	No	No
Cold Blooded	No	No	No	No
Cold Blooded	No	No	No	No
Warm blooded	Yes	No	No	Yes
Warm blooded	No	Yes	Yes	Yes
Cold blooded	No	Yes	Yes	No



Identifying how impure a leaf is

Gini Impurity = 1 – (probability of yes)^2 – (the probability of no)^2



Mammal?	
Yes	No
4	1

Gini(WB) = 1 – $(4/5)^2$ – $(1/5)^2$
= 0.32

Mammal?	
Yes	No
0	5

Gini(CB) = 1 – $(0/5)^2$ – $(5/5)^2$
= 0

We have the impurity for both leaves.

We now have to identify how impure the feature called ‘Body Temp’ is.

So can combine the weighted average using:

Total Weighted Gini = (5/10) x 0.32 + (5/10) x 0
= 0.16

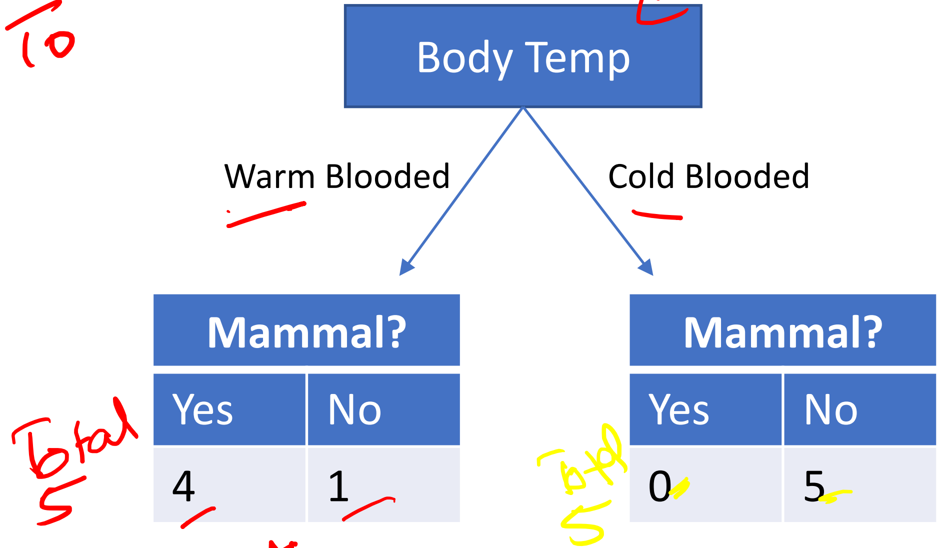
Let’s take the toy dataset

10 rows

$\frac{5}{10} \times 0.32 + \frac{5}{10} \times 0$

0.16

Body Temp	Gives Birth	Four-legged	Hibernates	Mammal?
Warm blooded	Yes	No	No	Yes
Warm blooded	No	No	No	No
Warm blooded	Yes	Yes	No	Yes
Cold Blooded	Yes	No	No	No
Cold Blooded	No	Yes	No	No
Cold Blooded	No	No	No	No
Cold Blooded	No	No	No	No
Warm blooded	Yes	No	No	Yes
Warm blooded	No	Yes	Yes	Yes
Cold blooded	No	Yes	Yes	No



Gini(WB) = $1 - (4/5)^2 - (1/5)^2$
= 0.32

Gini(CB) = $1 - (0/5)^2 - (5/5)^2$
= 0

Total Weighted Gini = (5/10) x 0.32 + (5/10) x 0
= 0.16

Let’s take the toy dataset

Body Temp	Gives Birth	Four-legged	Hibernates	Mammal?
Warm blooded	Yes	No	No	Yes
Warm blooded	No	No	No	No
Warm blooded	Yes	Yes	No	Yes
Cold Blooded	Yes	No	No	No
Cold Blooded	No	Yes	No	No
Cold Blooded	No	No	No	No
Cold Blooded	No	No	No	No
Warm blooded	Yes	No	No	Yes
Warm blooded	No	Yes	Yes	Yes
Cold blooded	No	Yes	Yes	No

Rows = 10

Gives Birth

Yes

4

Mammal?	
Yes	No
3	1

No

6

Mammal?	
Yes	No
1	5

NM

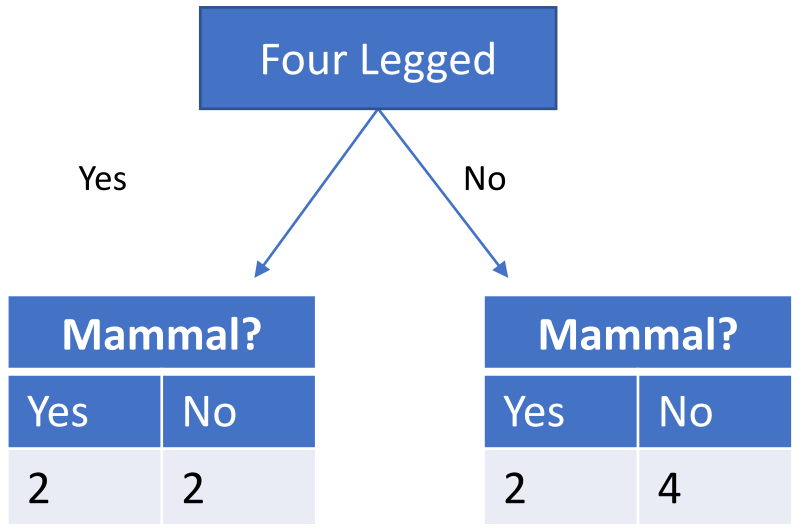
Gini(Yes) = $1 - (3/4)^2 - (1/4)^2$
= 0.25

Gini(No) = $1 - (1/6)^2 - (5/6)^2$
= 0.278

Total Weighted Gini = (4/10) x 0.25 + (6/10) x 0.278
= 0.2668

Let's take the toy dataset

Body Temp	Gives Birth	Four-legged	Hibernates	Mammal?
Warm blooded	Yes	No	No	Yes
Warm blooded	No	No	No	No
Warm blooded	Yes	Yes	No	Yes
Cold Blooded	Yes	No	No	No
Cold Blooded	No	Yes	No	No
Cold Blooded	No	No	No	No
Cold Blooded	No	No	No	No
Warm blooded	Yes	No	No	Yes
Warm blooded	No	Yes	Yes	Yes
Cold blooded	No	Yes	Yes	No



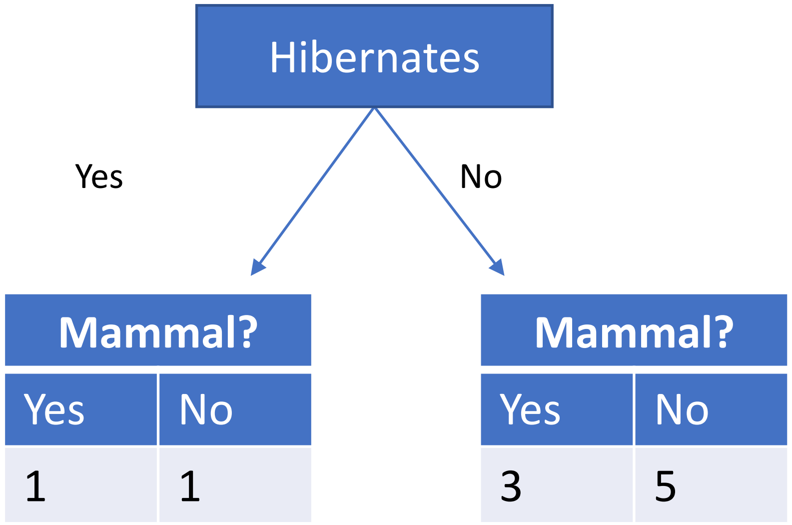
Gini(Yes) $= 1 - (2/4)^2 - (2/4)^2$
$= 0.5$

Gini(No) $= 1 - (2/6)^2 - (4/6)^2$
$= 0.44$

Total Weighted Gini $= (4/10) \times 0.5 + (6/10) \times 0.44$
$= 0.464$

Let’s take the toy dataset

Body Temp	Gives Birth	Four-legged	Hibernates	Mammal?
Warm blooded	Yes	No	No	Yes
Warm blooded	No	No	No	No
Warm blooded	Yes	Yes	No	Yes
Cold Blooded	Yes	No	No	No
Cold Blooded	No	Yes	No	No
Cold Blooded	No	No	No	No
Cold Blooded	No	No	No	No
Warm blooded	Yes	No	No	Yes
Warm blooded	No	Yes	Yes	Yes
Cold blooded	No	Yes	Yes	No



Gini(Yes) = $1 - (1/2)^2 - (1/2)^2$
= 0.5

Gini(No) = $1 - (3/8)^2 - (5/8)^2$
= 0.469

Total Weighted Gini = (2/10) x 0.5 + (8/10) x 0.469
= 0.475

- Gini Impurity for :
  - Body Temp = 1.6
  - Gives Birth = 0.2668 least impure
  - Four Legged = 0.464
  - Hibernates = 0.475

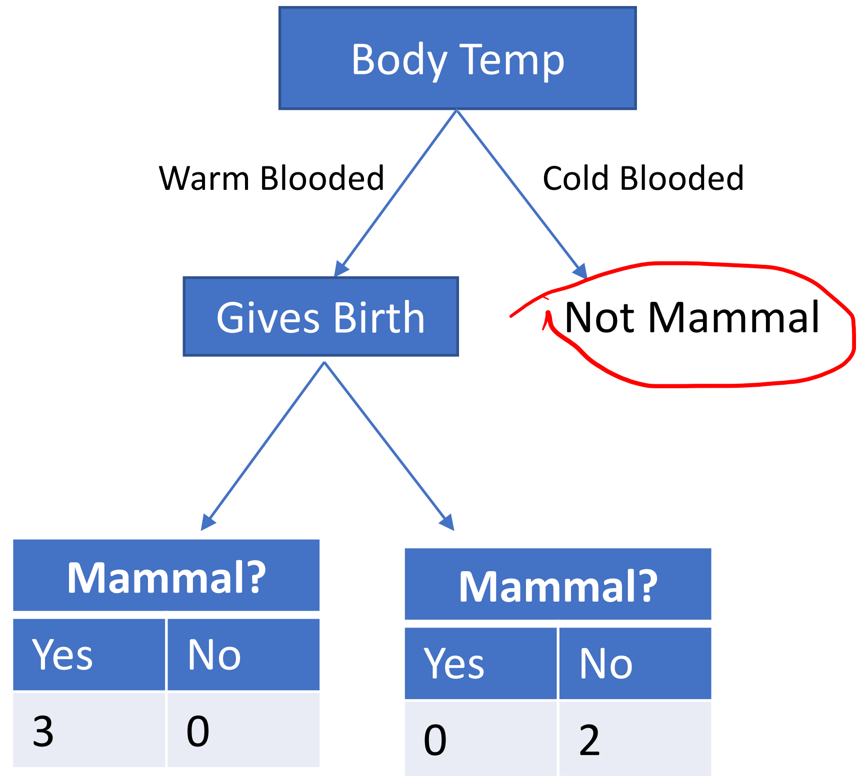
most

Body Temp is the ~~least~~ 'impure' feature, ~~so that'll be the root node~~

Body Temp	Gives Birth	Four-legged	Hibernates	Mammal?
Warm blooded	Yes	No	No	Yes
Warm blooded	No	No	No	No
Warm blooded	Yes	Yes	No	Yes
Cold Blooded	Yes	No	No	No
Cold Blooded	No	Yes	No	No
Cold Blooded	No	No	No	No
Cold Blooded	No	No	No	No
Warm blooded	Yes	No	No	Yes
Warm blooded	No	Yes	Yes	Yes
Cold blooded	No	Yes	Yes	No

Let’s take the toy dataset

Body Temp	Gives Birth	Four-legged	Hibernates	Mammal?
Warm blooded	Yes	No	No	Yes
Warm blooded	No	No	No	No
Warm blooded	Yes	Yes	No	Yes
Cold Blooded	Yes	No	No	No
Cold Blooded	No	Yes	No	No
Cold Blooded	No	No	No	No
Cold Blooded	No	No	No	No
Warm blooded	Yes	No	No	Yes
Warm blooded	No	Yes	Yes	Yes
Cold blooded	No	Yes	Yes	No



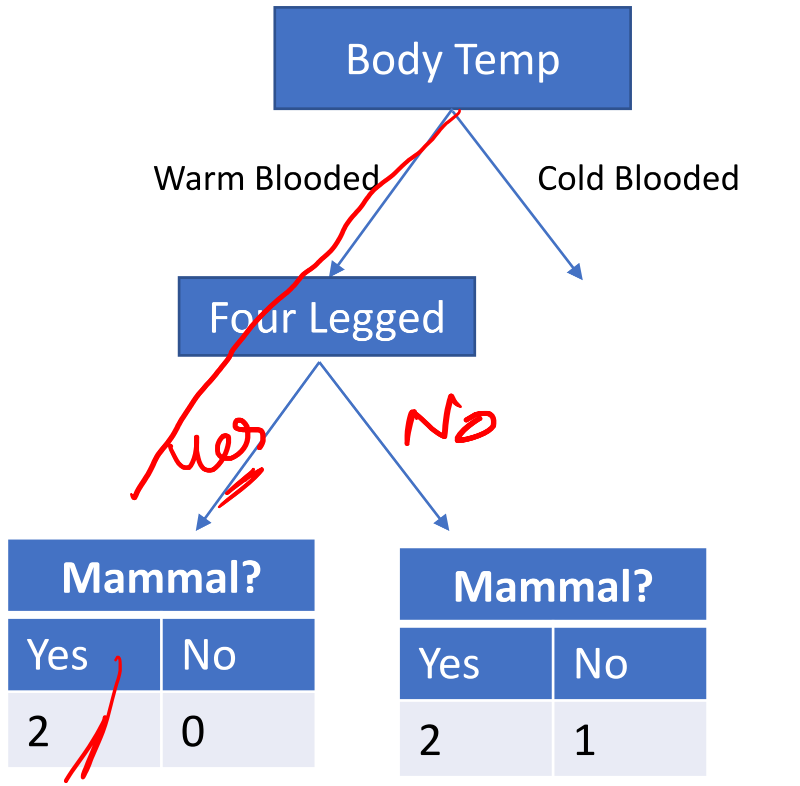
Gini(Yes) = $1 - (3/3)^2 - (0/3)^2$
$= 0$

Gini(Yes) = $1 - (0/2)^2 - (2/2)^2$
$= 0$

Total Weighted Gini = $(3/5) \times 0 + (2/5) \times 0$
$= 0$

Let's take the toy dataset

Body Temp	Gives Birth	Four-legged	Hibernates	Mammal?
Warm blooded	Yes	No	No	Yes
Warm blooded	No	No	No	No
Warm blooded	Yes	Yes	No	Yes
Cold Blooded	Yes	No	No	No
Cold Blooded	No	Yes	No	No
Cold Blooded	No	No	No	No
Cold Blooded	No	No	No	No
Warm blooded	Yes	No	No	Yes
Warm blooded	No	Yes	Yes	Yes
Cold blooded	No	Yes	Yes	No



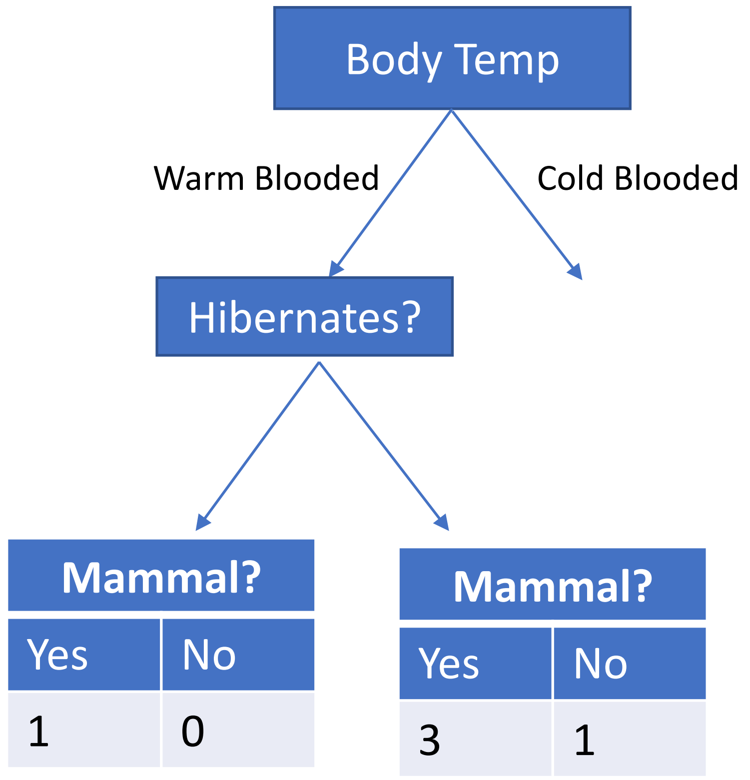
Gini(Yes) = $1 - (2/2)^2 - (0/2)^2$
= 0

Gini(Yes) = $1 - (2/3)^2 - (1/3)^2$
= 0.44

Total Weighted Gini = $(2/5) \times 0 + (3/5) \times 0.44$
= 0.264

Let’s take the toy dataset

Body Temp	Gives Birth	Four-legged	Hibernates	Mammal?
Warm blooded	Yes	No	No	Yes
Warm blooded	No	No	No	No
Warm blooded	Yes	Yes	No	Yes
Cold Blooded	Yes	No	No	No
Cold Blooded	No	Yes	No	No
Cold Blooded	No	No	No	No
Cold Blooded	No	No	No	No
Warm blooded	Yes	No	No	Yes
Warm blooded	No	Yes	Yes	Yes
Cold blooded	No	Yes	Yes	No



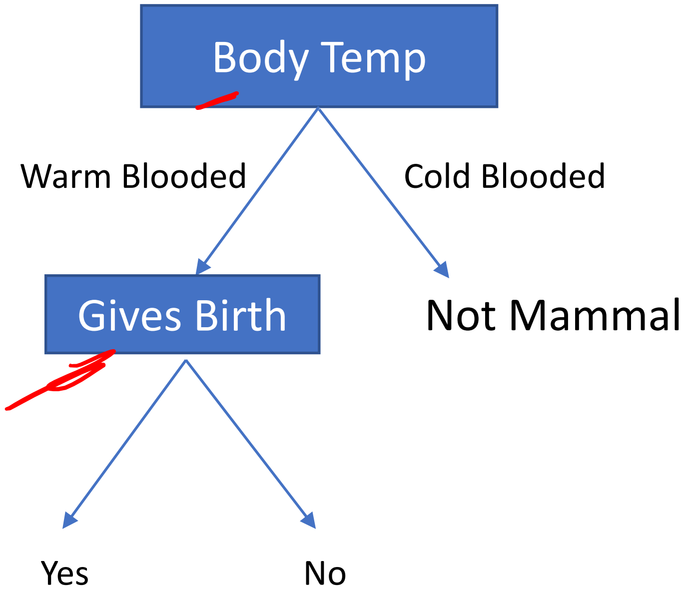
Gini(Yes) $= 1 - (1/1)^2 - (0/1)^2$
$= 0$

Gini(Yes) $= 1 - (3/4)^2 - (1/4)^2$
$= 0.375$

Total Weighted Gini $= (1/5) \times 0 + (4/5) \times 0.375$
$= 0.3$

Let’s take the toy dataset

Body Temp	Gives Birth	Four-legged	Hibernates	Mammal?
Warm blooded	Yes	No	No	Yes
Warm blooded	No	No	No	No
Warm blooded	Yes	Yes	No	Yes
Cold Blooded	Yes	No	No	No
Cold Blooded	No	Yes	No	No
Cold Blooded	No	No	No	No
Cold Blooded	No	No	No	No
Warm blooded	Yes	No	No	Yes
Warm blooded	No	Yes	Yes	Yes
Cold blooded	No	Yes	Yes	No



What if the dataset had a column with numerical data?

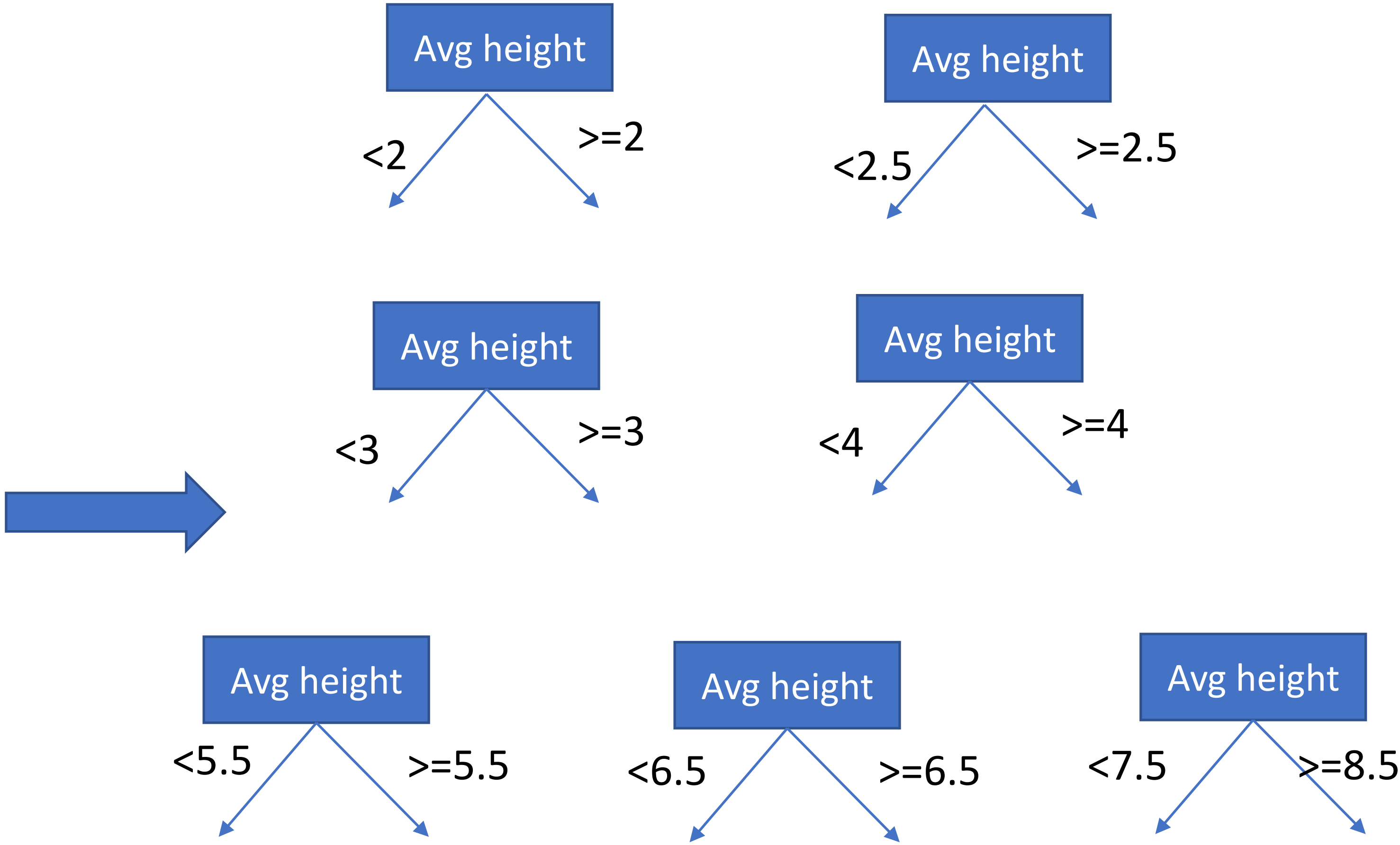
Body Temp	Gives Birth	Four-legged	Average height	Hibernates	Mammal?
Warm blooded	Yes	No	7	No	Yes
Warm blooded	No	No	3	No	No
Warm blooded	Yes	Yes	6	No	Yes
Cold Blooded	Yes	No	3	No	No
Cold Blooded	No	Yes	2	No	No
Cold Blooded	No	No	5	No	No
Cold Blooded	No	No	2	No	No
Warm blooded	Yes	No	9	No	Yes
Warm blooded	No	Yes	8	Yes	Yes
Cold blooded	No	Yes	4	Yes	No

Body Temp	Gives Birth	Four-legged	Average height	Hibernates	Mammal?
Warm blooded	Yes	No	7	No	Yes
Warm blooded	No	No	3	No	No
Warm blooded	Yes	Yes	6	No	Yes
Cold Blooded	Yes	No	3	No	No
Cold Blooded	No	Yes	2	No	No
Cold Blooded	No	No	5	No	No
Cold Blooded	No	No	2	No	No
Warm blooded	Yes	No	9	No	Yes
Warm blooded	No	Yes	8	Yes	Yes
Cold blooded	No	Yes	4	Yes	No

Body Temp	Gives Birth	Four-legged	Average height	Hibernates	Mammal?
Cold Blooded	No	Yes	2	No	No
Cold Blooded	No	No	2	No	No
Warm blooded	No	No	3	No	No
Cold Blooded	Yes	No	3	No	No
Cold Blooded	No	No	5	No	No
Warm blooded	Yes	Yes	6	No	Yes
Warm blooded	Yes	No	7	No	Yes
Warm blooded	No	Yes	8	Yes	Yes
Warm blooded	Yes	No	9	No	Yes

Body Temp	Gives Birth	Four-legged	Average height	Hibernates	Mammal?
Cold Blooded	No	Yes	2	No	No
Cold Blooded	No	No	2	No	No
Warm blooded	No	No	3	No	No
Cold Blooded	Yes	No	3	No	No
Cold Blooded	No	No	5	No	No
Warm blooded	Yes	Yes	6	No	Yes
Warm blooded	Yes	No	7	No	Yes
Warm blooded	No	Yes	8	Yes	Yes
Warm blooded	Yes	No	9	No	Yes

2
2.5
3
4
5.5
6.5
7.5
8.5



We generate multiple trees from one column, select the one with least gini impurity and then compare it with columns for split.

Validation sets

Validation
= test

- We need a way to calculate how the model that we build will perform on the real world.
- Each time we are given a dataset, we must split the dataset randomly, into the train set and the validation set.
- The validation set is assumed to be a representative of real-world data.
- The model looks into the train set and trains itself. Then, we pass the validation set's inputs into the model and make the model to do predictions. Since, we already know what the correct output is, we compare the predictions with the actual outputs, so validate how well the model will perform in real world.

An analogy

- We study for our math tests using sample problems given in a book. These sample problems are the train set.
- Now, if we ask the same sample problems from the book in an exam, the student can simply memorize all the problems in the book without having to understand the conceptual underpinnings.
- So, instead of testing the students with the same problems from the book, we ask slightly different questions, but based on the same concepts.
- This is what we do with validation sets. The model learns the patterns required to make predictions from the train set. In order to if the model has just memorized the data instead of learning the patterns, we check how well it performs on the validation set.

So, what now?

k-Fold cross validation

70% 30%

- We first split the dataset like this. Train Set (80%) | Dev Set (20%)
- Then, we first develop the model based on the train set.
- The Dev/Validation set also has the input and their correct ground truth values. So, we take the inputs of the Dev set, get the predictions from the model, compare them with the ground truth values for the corresponding input and predicted value and check how many times our model got the output right.
- Let's look at an example for how to calculate the accuracy of the model on the Dev set.

We first split the dataset like this

Train Set (80%)	Validation Set (20%)

Validation set

Body Temp	Gives Birth	Four-legged	Hibernates	Predicted	Actual
Warm blooded	Yes	Yes	No	No	Yes
Warm blooded	No	No	Yes	No	No
Warm blooded	Yes	Yes	Yes	Yes	Yes
Cold Blooded	No	No	No	No	No
Cold Blooded	No	Yes	No	No	No

Accuracy = (4/5) * 100 = 80%

The Confusion Matrix

Let's take a new dataset, with the following features

False Positive

No Prediction Yes

Yes No

Train Set

Chest Pain	Good blood circ	Blocked Arteries	Weight	Heart Disease
No	No	No	125	No
Yes	Yes	Yes	180	Yes
…	…	…	…	…

We can train a model for this dataset using a Decision Tree, or any other model.

Validation Set

Chest Pain	Good blood circ	Blocked Arteries	Weight	Heart Disease
Yes	No	Yes	167	Yes
…	…	…	…	…

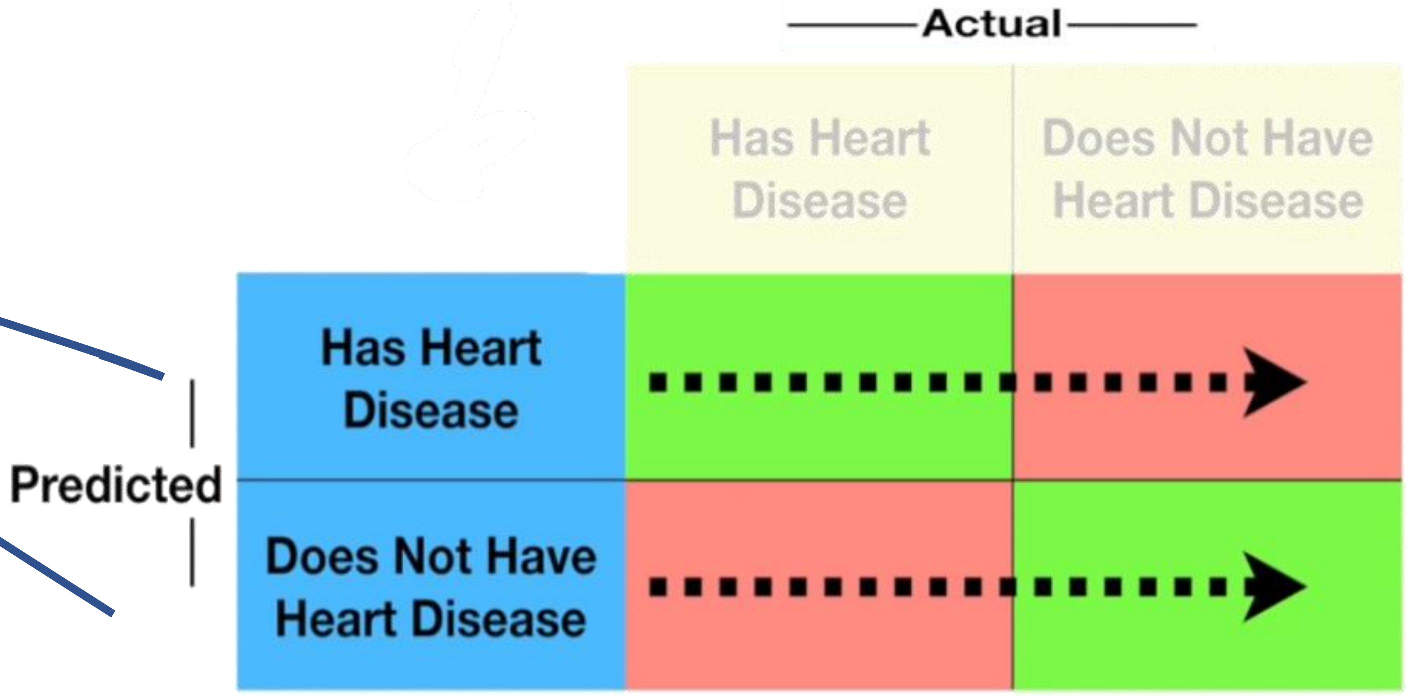
After training the decision tree, we validate the model and let's assume we get an accuracy of 88%.

Is there any other way to evaluate the model?

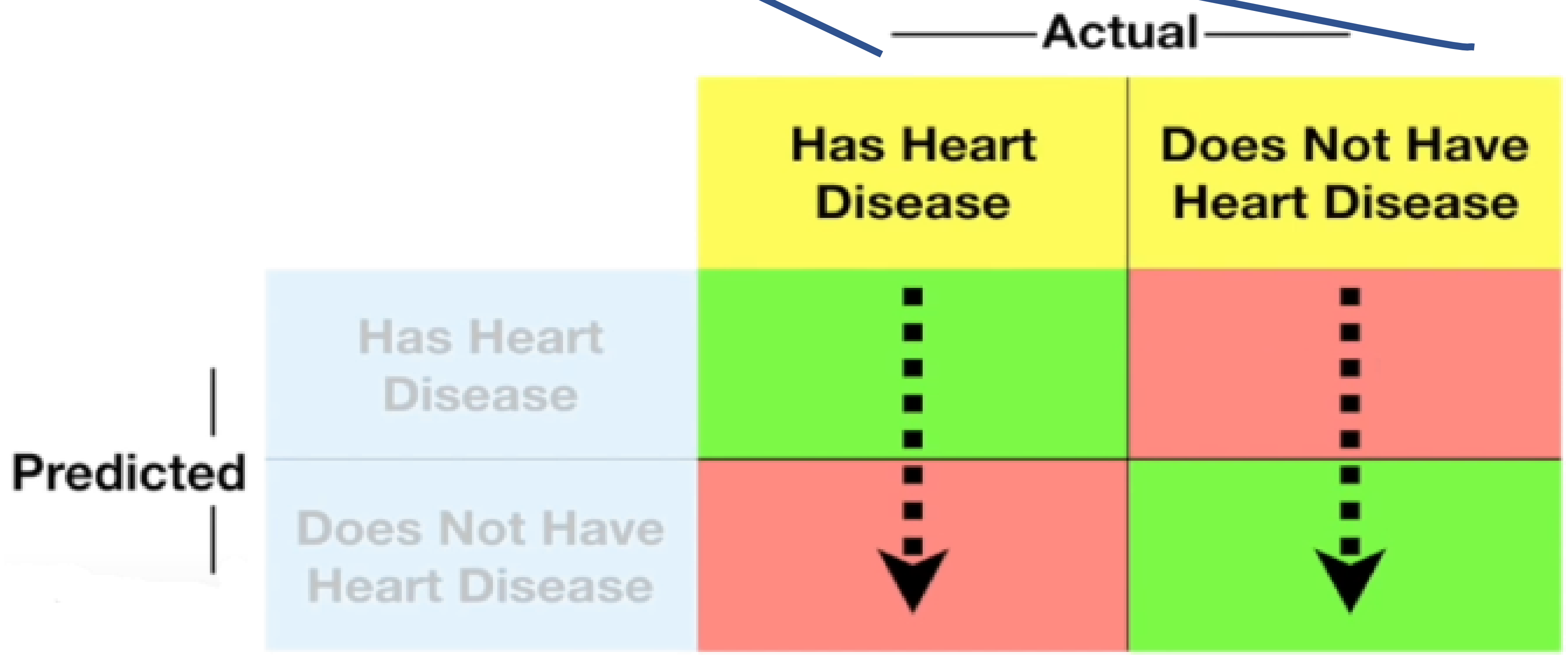
Enter Confusion Matrix

		Actual	
		Has Heart Disease	Does Not Have Heart Disease
Predicted	Has Heart Disease	20	22 FP
	Does Not Have Heart Disease	FN 12	80

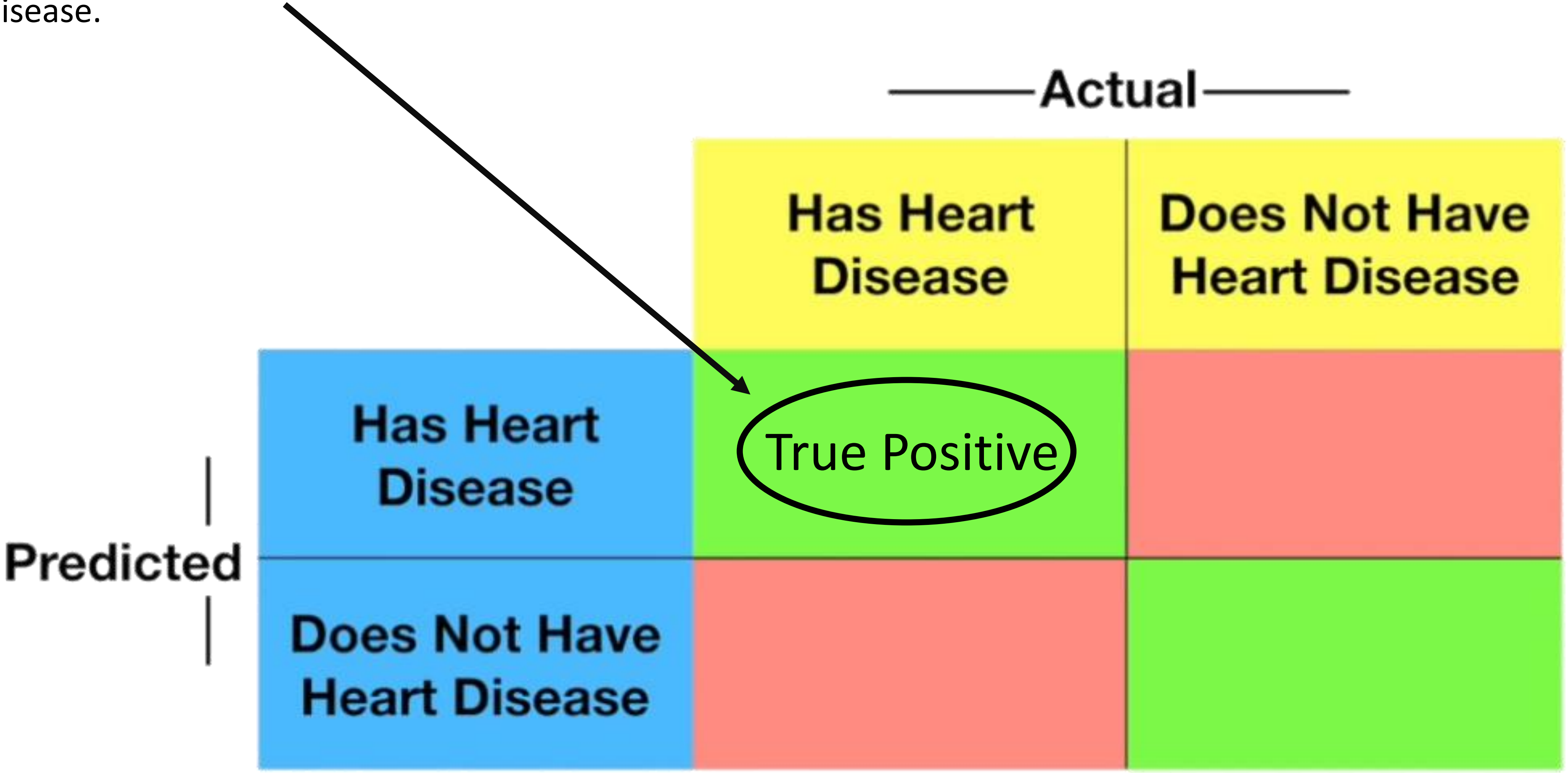
Rows correspond to what the model has predicted



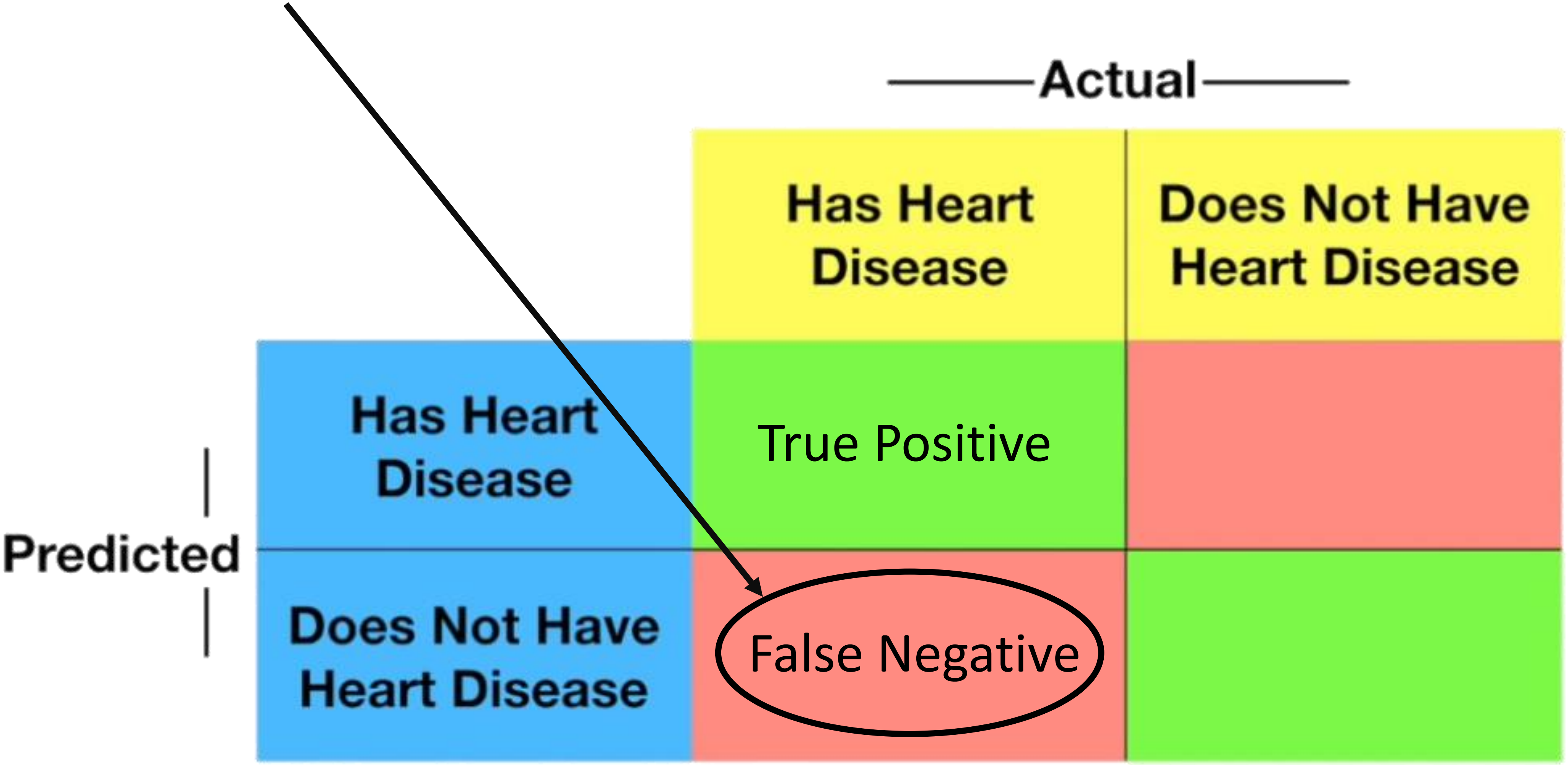
Columns correspond to what the ground truth is



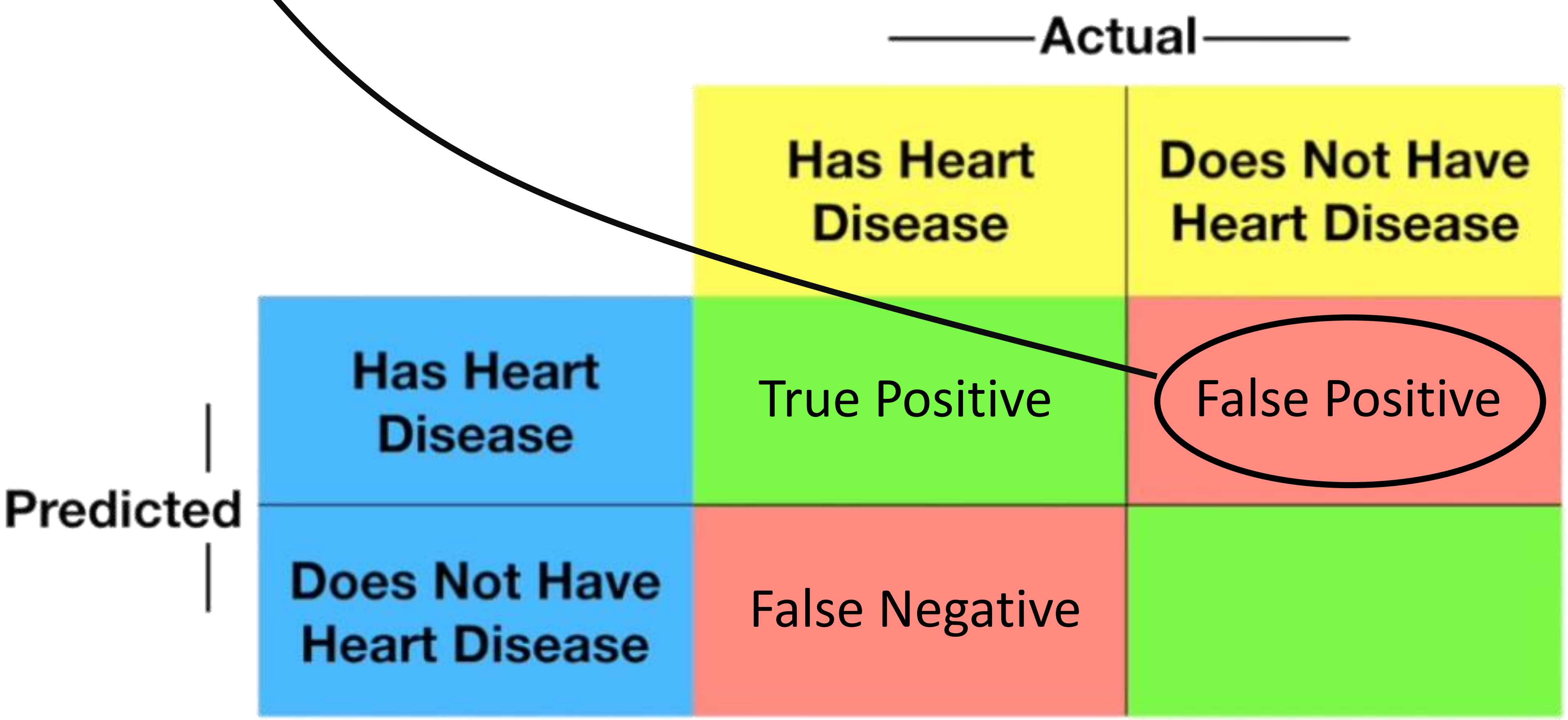
True Positive is when the patient **HAS** a heart disease, and the model also predicts that the patient **HAS** a heart disease.



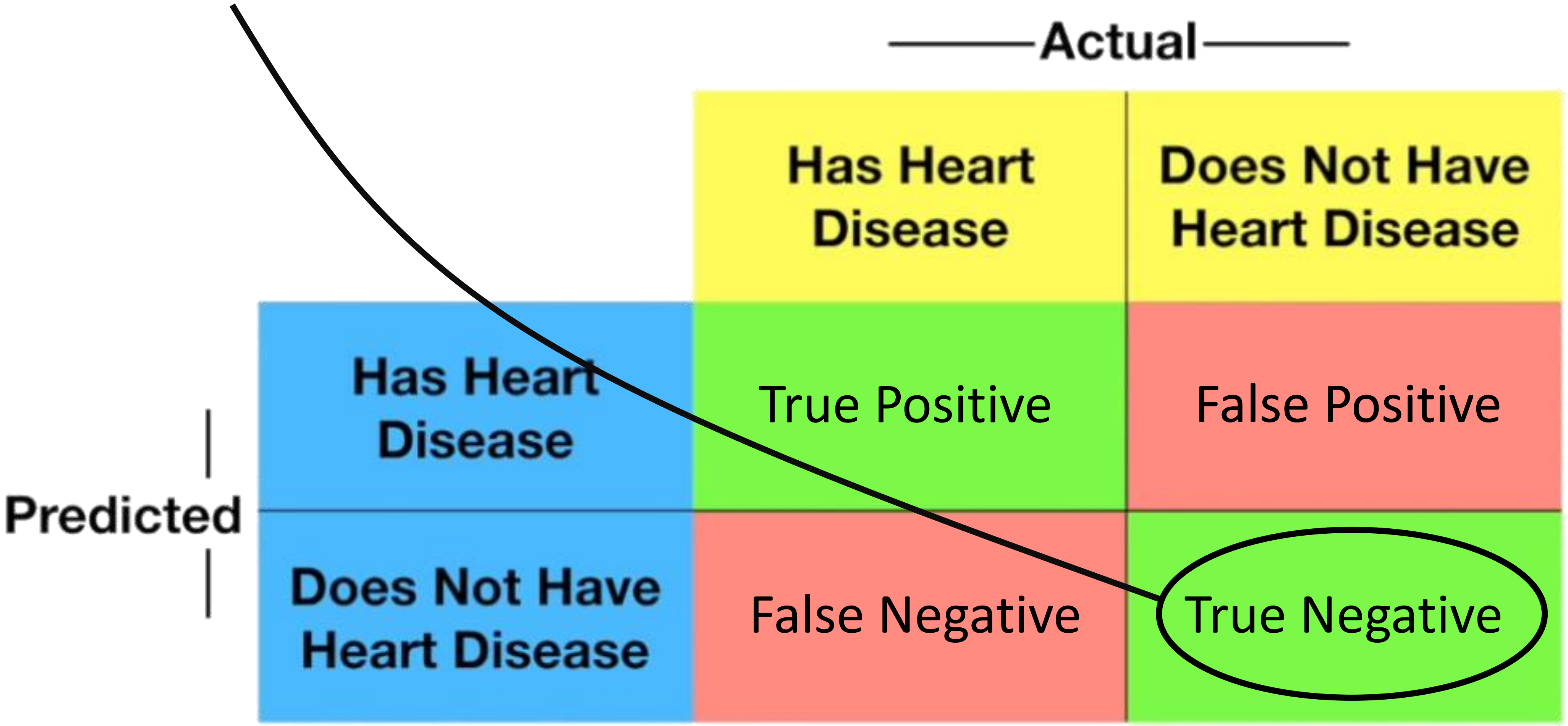
False Negative is when the patient **HAS** a heart disease, and the model also predicts that the patient **DOESN'T** have a heart disease.



False Positive is when the patient **DOESN'T** have a heart disease, but the model predicts that the patient **HAS** a heart disease.



True Negative is when the patient **DOESN'T** have a heart disease, and the model also **predicts** that the patient **HAS** a heart disease.



Sum of the numbers on the green boxes indicate how many times the model got the answer right in the validation set.

FP →

FN →

		Actual	
		Has Heart Disease	Does Not Have Heart Disease
Predicted	Has Heart Disease	142	22
	Does Not Have Heart Disease	29	110

Comparing models

So, let's assume that we have two models, one decision tree and another classification algorithm named Support Vector Machine (SVM). We'll get into how SVM works later.

We train both models on the train set. So, we want to evaluate how well the model is performing on different classes to see which one to deploy.

Decision Tree

		Actual: Has Heart Disease	Actual: Does Not Have Heart Disease
Predicted	Has Heart Disease	142	22
	Does Not Have Heart Disease	29	110

SVM

		Actual: Has Heart Disease	Actual: Does Not Have Heart Disease
Predicted	Has Heart Disease	107	53
	Does Not Have Heart Disease	64	79

Decision Tree has a good accuracy and is also producing lower false positives and false negatives. So, **in this case**, decision tree model is better for real time use.

So, what if the accuracy is same between, but they have different errors on false negatives and false positives like below

Decision Tree

		Actual	
		Has Heart Disease	Does Not Have Heart Disease
Predicted	Has Heart Disease	142	9
	Does Not Have Heart Disease	42	110

SVM

		Actual	
		Has Heart Disease	Does Not Have Heart Disease
Predicted	Has Heart Disease	140	55
	Does Not Have Heart Disease	3	105

Here the accuracies are almost similar. But We can see that the Decision Tree model gives a lot false negatives and that can be fatal to a lot of people. SVM gives lesser number of false negatives. So, in this case, even if the accuracy of SVM is slightly lower, since it is good at identifying people with a heart disease than the decision tree, we can say that SVM is a better model.

Multi-class Confusion Matrix

Jurassic Park	Run for your Wife	Out Cold	Howard the Duck	Favorite movie
Liked	Didn't Like	Liked	Liked	Troll 2
Didn't like	Liked	Didn't Like	Like	Gore Police
Like	Like	Didn't Like	Like	Cool as ice
…	…	…	…	…

		Actual		
		Troll 2	Gore Police	Cool as Ice
Predicted	Troll 2	12	102	93
	Gore Police	112	23	77
	Cool as Ice	83	92	17

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

In this case, sensitivity tells us what percentage of patients WITH heart disease were correctly identified

		Actual	
		Has Heart Disease	**Does Not Have Heart Disease**
Predicted	**Has Heart Disease**	True Positive	False Positive
	Does Not Have Heart Disease	False Negative	True Negative

$$\text{Specificity} = \frac{TN}{\text{TN} + FP}$$

In this case, sensitivity tells us what percentage of patients WITHOUT heart disease were correctly identified

		Actual	
		Has Heart Disease	**Does Not Have Heart Disease**
Predicted	**Has Heart Disease**	True Positive	False Positive
	Does Not Have Heart Disease	False Negative	True Negative

Let’s assume a model has the following confusion matrix.

Sensitivity = (139)/(139+32) = 0.81

Specificity = (112)/(112 + 20) = 0.85

		Actual	
		Has Heart Disease	Does Not Have Heart Disease
Predicted	Has Heart Disease	139	20
	Does Not Have Heart Disease	32	112

Let’s assume that another model has the following confusion matrix.

		Actual: Has Heart Disease	Actual: Does Not Have Heart Disease
Predicted	Has Heart Disease	142	20
	Does Not Have Heart Disease	32	112

Sensitivity = (142)/(142 + 29) = 0.83

Specificity = (110)/(110 + 22) = 0.83

Let’s compare the two models

Model 1

Sensitivity = (139)/(139+32) = 0.81

Specificity = (112)/(112 + 20) = 0.85

Model 2

Sensitivity = (142)/(142 + 29) = 0.83

Specificity = (110)/(110 + 22) = 0.83

Based on sensitivity, we can infer that Model 2 is better at predicting whether a person has heart disease. Based on Specificity, we can infer that Model 1 is better at identifying people who don’t have a heart disease.