Boosting

Boosting

- The concept of **boosting** is not actually a machine learning algorithm, it is methodology *applied* to an existing machine learning algorithm, most commonly applied to the decision tree.
- Let's explore this idea of a meta-learning algorithm by reviewing a simple application and formula.

Boosting

- Main formula for boosting:

$$F_T(x) = \sum_{t=1}^{T} f_t(x) \qquad f_t(x) = \alpha_t h(x)$$

- Implies that a combination of **estimators** with an applied **coefficient** could act as an effective **ensemble estimator**.

Boosting

- Main formula for boosting:

$$F_T(x) = \sum_{t=1}^{T} f_t(x) \qquad f_t(x) = \alpha_t h(x)$$

- Note **h(x)** can in theory be **any** machine learning algorithm (estimator/learner).

Boosting

- Main formula for boosting:

$$F_T(x) = \sum_{t=1}^{T} f_t(x) \qquad f_t(x) = \alpha_t h(x)$$

- Can an ensemble of **weak learners** (very simple models) be a **strong learner** when combined?

Boosting

- Main formula for boosting:

$$F_T(x) = \sum_{t=1}^{T} f_t(x) \qquad f_t(x) = \alpha_t h(x)$$

- For decision tree models, we can use simple trees in place of h(x) and combine them with the coefficients on each model.

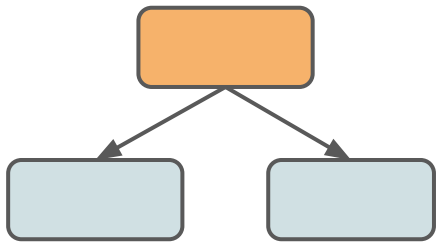
AdaBoost

Boosting

- AdaBoost (Adaptive Boosting) works by using an ensemble of **weak learners** and then combining them through the use of a weighted sum.
- Adaboost adapts by using previously created **weak learners** in order to adjust misclassified instances for the next created **weak learner.**

Boosting

- What is a **weak learner**?
  - A weak model is a model that is too simple to perform well on its own.
  - The weakest decision tree possible would be a **stump**, one node and two leaves!



Boosting

- Unlike a single decision tree which fits to all the data at once (*fitting the data hard*), AdaBoost aggregates multiple weak learners, allowing the overall **ensemble** model to *learn slowly* from the features.
- Let's first understand how this works from a data perspective!

AdaBoost

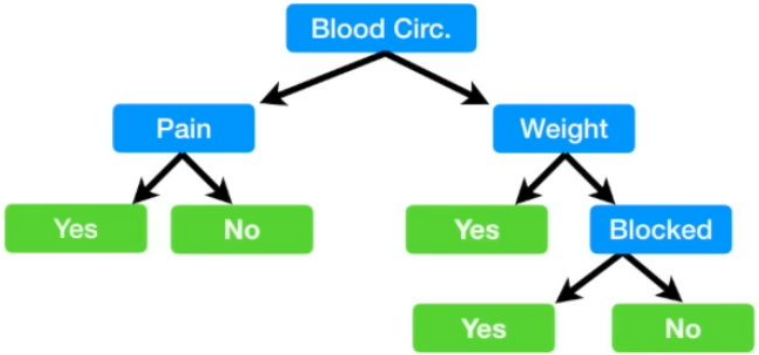
For example, if we were using this data to determine if someone had heart disease or not…

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

AdaBoost

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

…then a full sized **Decision Tree** would take advantage of all **4** variables that we measured (**Chest Pain**, **Blood Circulation**, **Blocked Arteries** and **Weight**) to make a decision…



AdaBoost

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

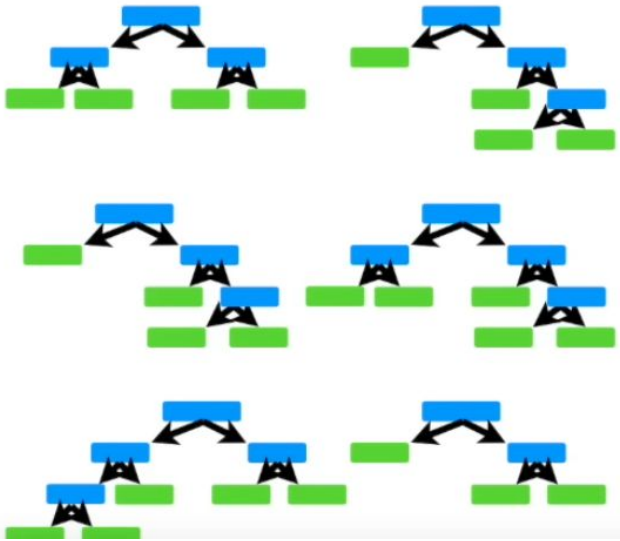
…but a **Stump** can only use one variable to make a decision.

Thus, **Stumps** are technically "weak learners".

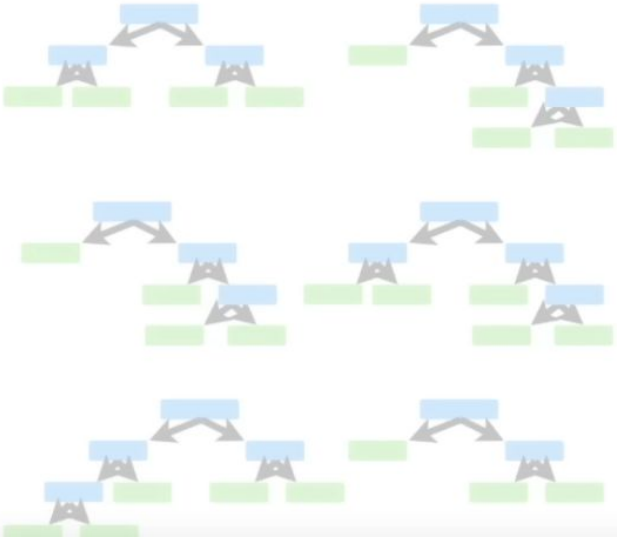
However, that's the way **AdaBoost** likes it, and it's one of the reasons why they are so commonly combined.

AdaBoost

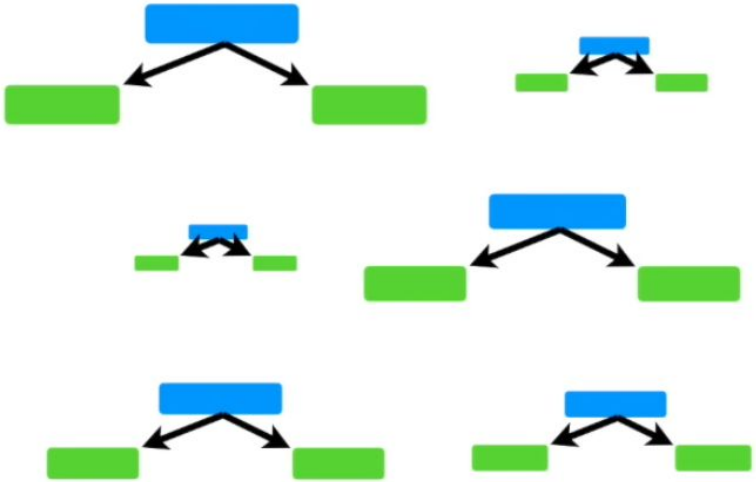
In a **Random Forest**, each tree has an equal vote on the final classification.



AdaBoost

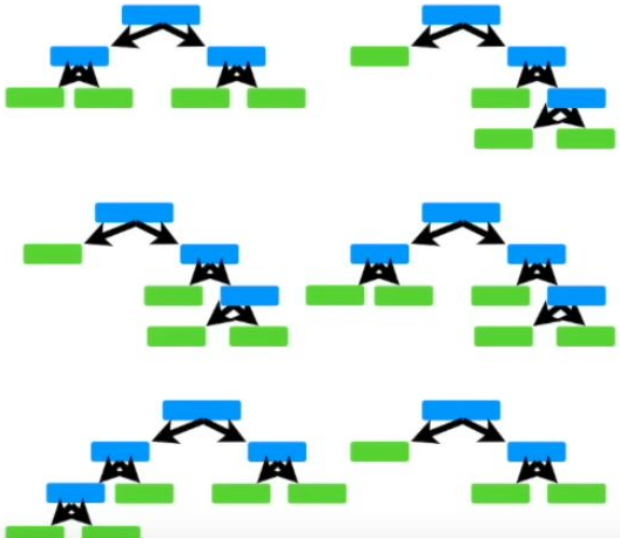


In contrast, in a **Forest of Stumps** made with **AdaBoost**, some stumps get more say in the final classification than others.

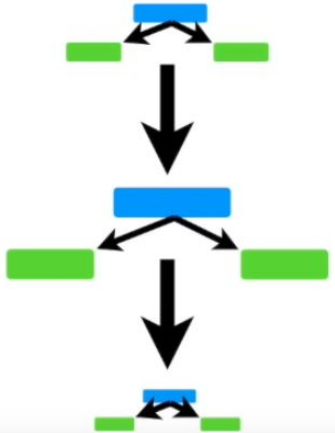


AdaBoost

Lastly, in a **Random Forest**, each decision tree is made independently of the others.



In contrast, in a **Forest of Stumps** made with **AdaBoost**, order is important.



AdaBoost

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.

AdaBoost

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

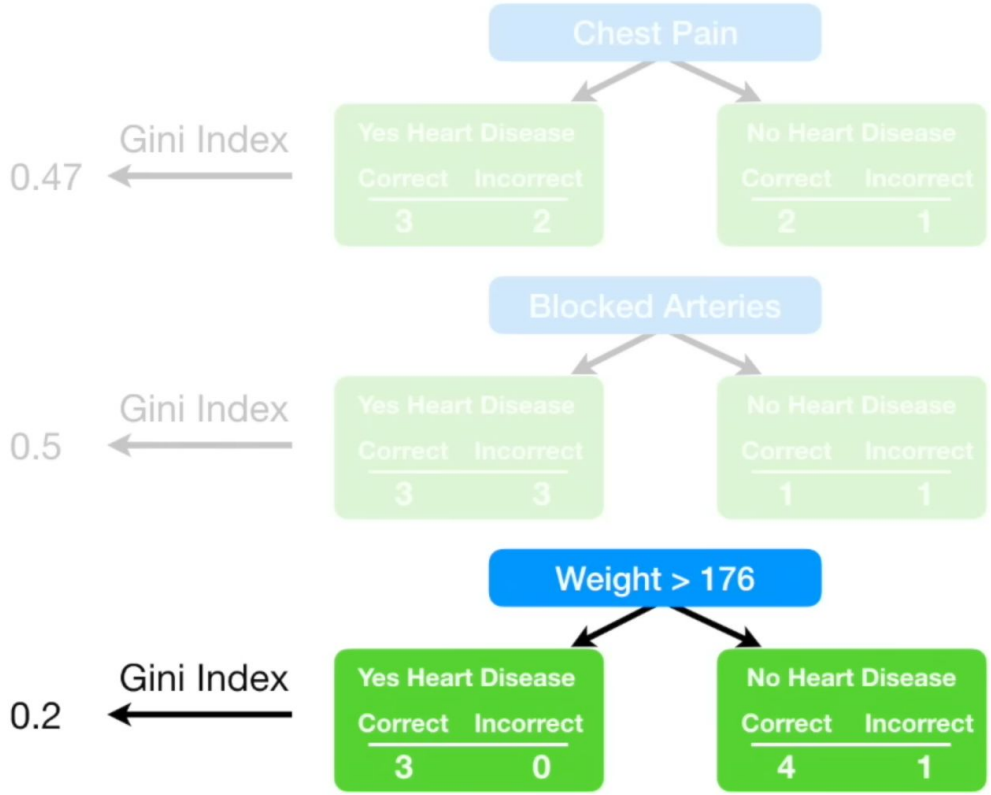
At the start, all samples get the same weight…

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

…and that makes the samples all equally important.

AdaBoost

Now we calculate the **Gini Index** for the three stumps.



AdaBoost

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

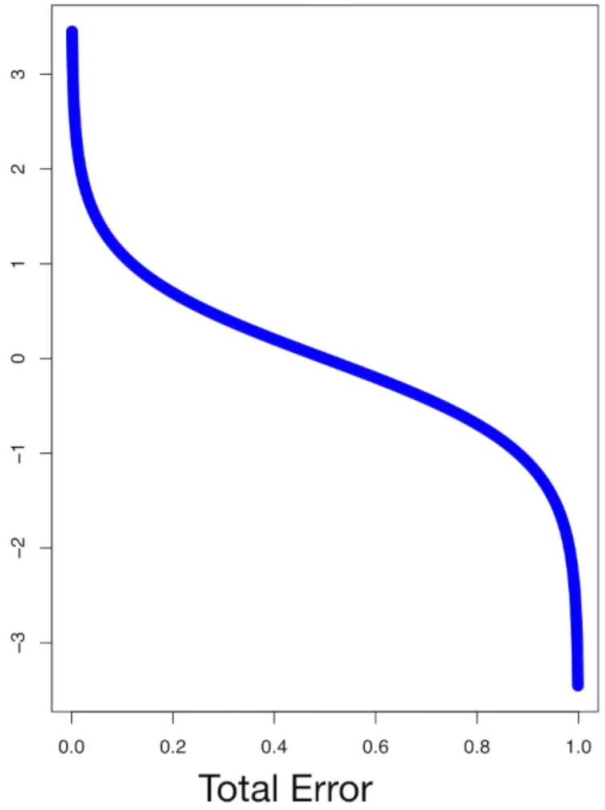
We determine how much say a stump has in the final classification based on how well it classified the samples.

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

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

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

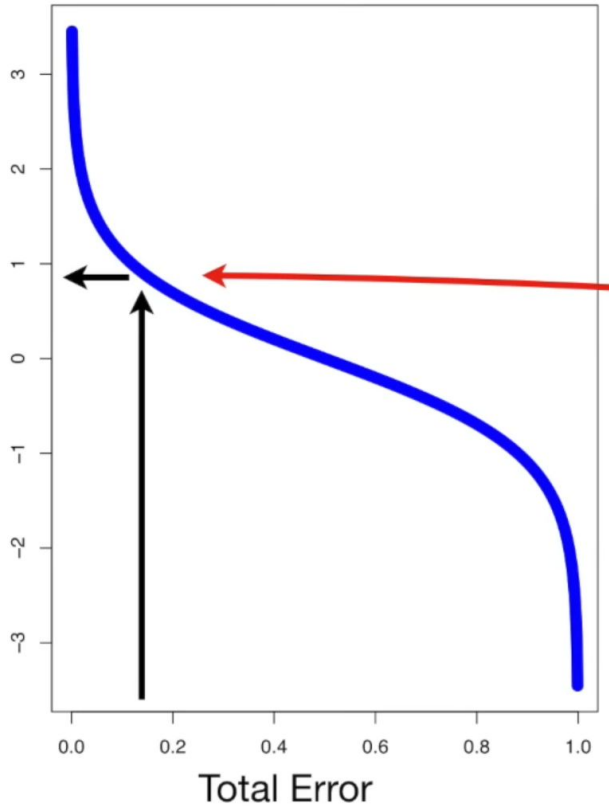
AdaBoost



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

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

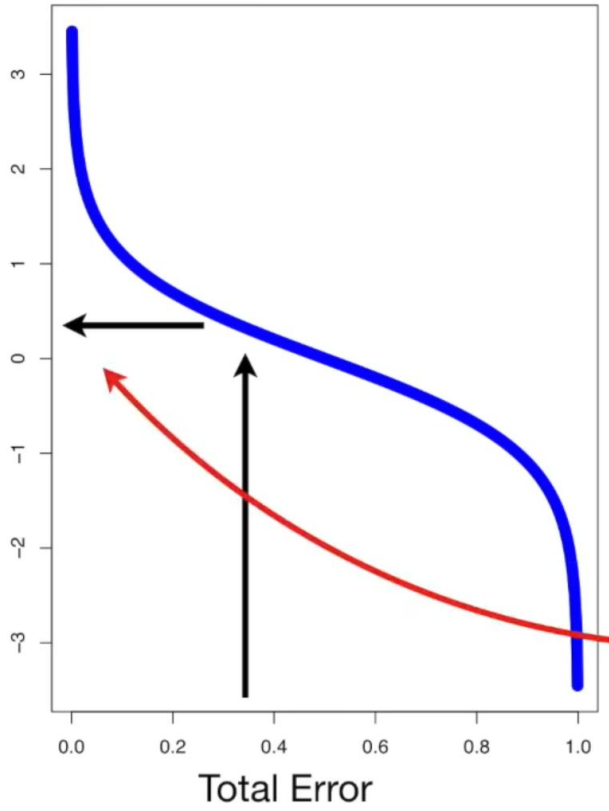
AdaBoost

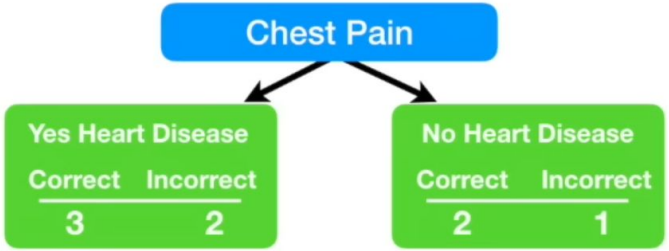


…and the **Amount of Say** that this stump has on the final classification is **0.97**.

$$\text{Amount of Say} = \frac{1}{2}\log(7) = 0.97$$

AdaBoost





$$\text{Total Error} = \frac{1}{8} + \frac{1}{8} + \frac{1}{8} = \frac{3}{8}$$

So we are expecting the **Amount of Say** to be between 0 and 0.5.

AdaBoost

Now we need to learn how to modify the weights so that the next stump will take the errors that the current stump made into account.

AdaBoost

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 will emphasize the need for the next stump to correctly classify it by increasing its **Sample Weight**…

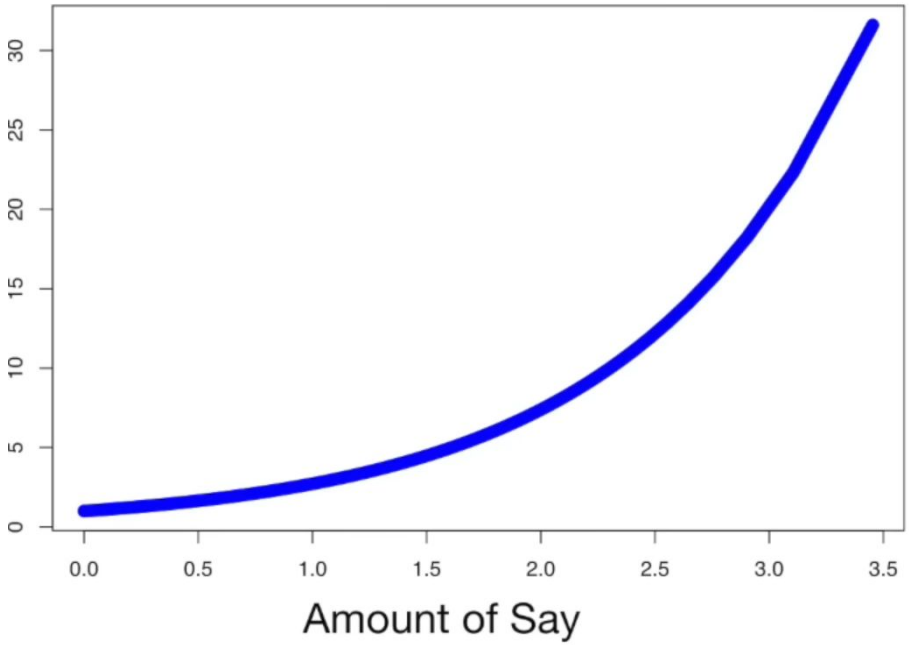
AdaBoost

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

$$\text{New Sample Weight} = \text{sample weight} \times e^{\text{amount of say}}$$

This is the formula we will use to *increase* the **Sample Weight** for the sample that was *incorrectly* classified.

AdaBoost

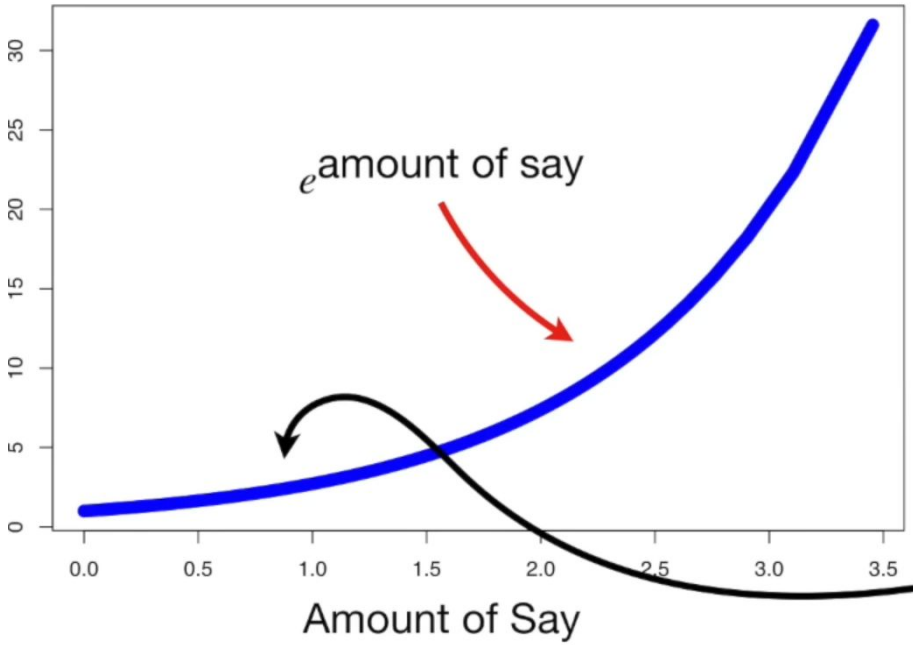


$$\text{New Sample Weight} = \text{sample weight} \times e^{\text{amount of say}}$$

$$= \frac{1}{8} e^{\text{amount of say}}$$

To get a better understanding of how this part will scale the previous **Sample Weight**, let's draw a graph.

AdaBoost



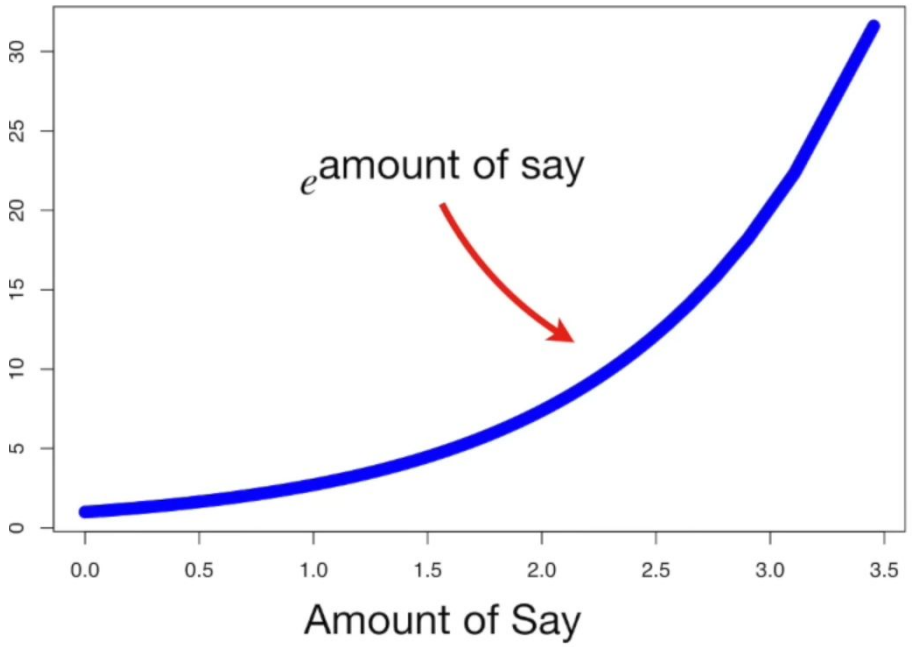
$$\text{New Sample Weight} = \text{sample weight} \times e^{\text{amount of say}}$$

$$= \frac{1}{8} e^{\text{amount of say}}$$

$$= \frac{1}{8} e^{0.97} = \frac{1}{8} \times 2.64$$

…and $e^{0.97} = 2.64$

AdaBoost



$$\text{New Sample Weight} = \text{sample weight} \times e^{\text{amount of say}}$$

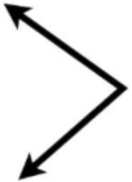
$$= \frac{1}{8} e^{\text{amount of say}}$$

$$= \frac{1}{8} e^{0.97} = \frac{1}{8} \times 2.64 = 0.33$$

That means the new **Sample Weight** is **0.33**, which is *more* than the old one (**1/8** = **0.125**).

AdaBoost

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 *decrease* the **Sample Weights** for all of the *correctly* classified samples.

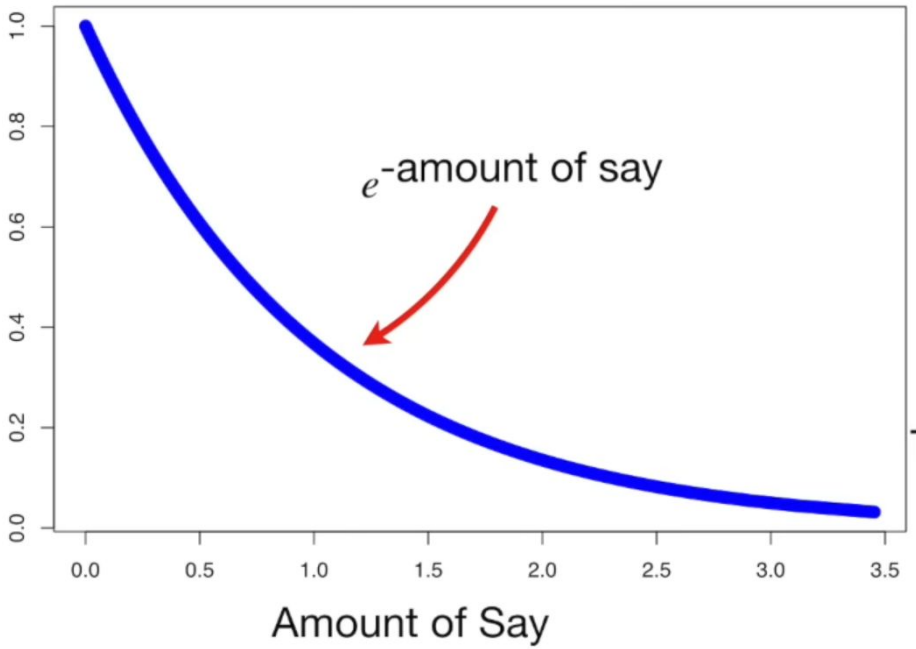
AdaBoost

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

$$\text{New Sample Weight} = \text{sample weight} \times e^{\text{-amount of say}}$$

This is the formula we will use to *decrease* the **Sample Weights**.

AdaBoost



$$\text{New Sample Weight} = \text{sample weight} \times e^{\text{-amount of say}}$$

$$= \frac{1}{8} e^{\text{-amount of say}}$$

$$= \frac{1}{8} e^{-0.97} = \frac{1}{8} \times 0.38 = 0.05$$

The new **Sample Weight** is **0.05**, which is *less* than the old one (**1/8** = **0.125**).

AdaBoost

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight	New Weight
Yes	Yes	205	Yes	1/8	0.05
No	Yes	180	Yes	1/8	0.05
Yes	No	210	Yes	1/8	0.05
Yes	Yes	167	Yes	1/8	0.33
No	Yes	156	No	1/8	0.05
No	Yes	125	No	1/8	0.05
Yes	No	168	No	1/8	0.05
Yes	Yes	172	No	1/8	0.05

Now we need to normalize the **New Sample Weights** so that they will add up to 1.

AdaBoost

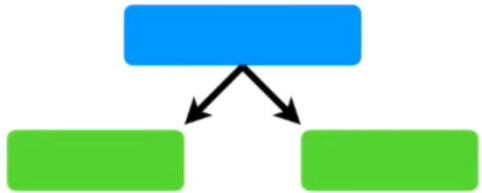
Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight	New Weight	Norm. Weight
Yes	Yes	205	Yes	1/8	0.05	0.07
No	Yes	180	Yes	1/8	0.05	0.07
Yes	No	210	Yes	1/8	0.05	0.07
Yes	Yes	167	Yes	1/8	0.33	0.49
No	Yes	156	No	1/8	0.05	0.07
No	Yes	125	No	1/8	0.05	0.07
Yes	No	168	No	1/8	0.05	0.07
Yes	Yes	172	No	1/8	0.05	0.07

So we divide each **New Sample Weight** by **0.68** to get the normalized values.

AdaBoost

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	0.07
No	Yes	180	Yes	0.07
Yes	No	210	Yes	0.07
Yes	Yes	167	Yes	0.49
No	Yes	156	No	0.07
No	Yes	125	No	0.07
Yes	No	168	No	0.07
Yes	Yes	172	No	0.07

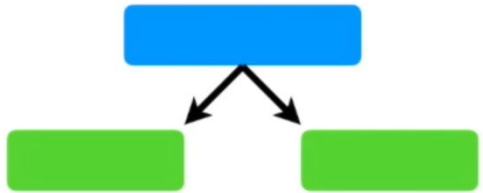
Now we can use the modified **Sample Weights** to make the second **stump** in the forest.



AdaBoost

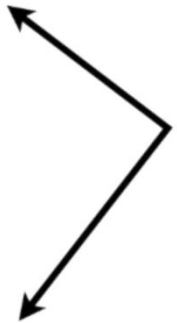
Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	0.07
No	Yes	180	Yes	0.07
Yes	No	210	Yes	0.07
Yes	Yes	167	Yes	0.49
No	Yes	156	No	0.07
No	Yes	125	No	0.07
Yes	No	168	No	0.07
Yes	Yes	172	No	0.07

In theory, we could use the **Sample Weights** to calculate **Weighted Gini Indexes** to determine which variable should split the next stump.



AdaBoost

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	0.07
No	Yes	180	Yes	0.07
Yes	No	210	Yes	0.07
Yes	Yes	167	Yes	0.49
No	Yes	156	No	0.07
No	Yes	125	No	0.07
Yes	No	168	No	0.07
Yes	Yes	172	No	0.07



Alternatively, instead of using a **Weighted Gini Index**, we can make a new collection of samples that contains duplicate copies of the samples with the largest **Sample Weights**.

AdaBoost

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	0.07
No	Yes	180	Yes	0.07
Yes	No	210	Yes	
Yes	Yes	167	Yes	
No	Yes	156	No	
No	Yes	125	No	0.07
Yes	No	168	No	0.07
Yes	Yes	172	No	0.07

So we start by making a new, but empty, dataset that is the same size as the original…

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease

AdaBoost

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	0.07
No	Yes	180	Yes	0.07
Yes	No	210	Yes	0.07
Yes	Yes	167	Yes	0.49
No	Yes	156	No	0.07
No	Yes	125	No	0.07
Yes	No	168	No	0.07
Yes	Yes	172	No	0.07

Then we pick a random number between 0 and 1…

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease

AdaBoost

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	0.07
No	Yes	180	Yes	0.07
Yes	No	210	Yes	0.07
Yes	Yes	167	Yes	0.49
No	Yes	156	No	0.07
No	Yes	125	No	0.07
Yes	No	168	No	0.07
Yes	Yes	172	No	0.07

…and we see where that number falls when we use the **Sample Weights** like a distribution.

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease

AdaBoost

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	0.07
No	Yes	180	Yes	0.07
Yes	No	210	Yes	0.07
Yes	Yes	167	Yes	0.49
No	Yes	156	No	0.07
No	Yes	125	No	0.07
Yes	No	168	No	0.07
Yes	Yes	172	No	0.07

If the number is between **0** and **0.07**, then we would put this sample into the new collection of samples…

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease

AdaBoost

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	0.07
No	Yes	180	Yes	0.07
Yes	No	210	Yes	0.07
Yes	Yes	167	Yes	0.49
No	Yes	156	No	0.07
No	Yes	125	No	0.07
Yes	No	168	No	0.07
Yes	Yes	172	No	0.07

…and if the number is between **0.07** and **0.14** (**0.07** + **0.07** = **0.14**), then we would put this sample into the new collection of samples…

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease

AdaBoost

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	0.07
No	Yes	180	Yes	0.07
Yes	No	210	Yes	0.07
Yes	Yes	167	Yes	0.49
No	Yes	156	No	0.07
No	Yes	125	No	0.07
Yes	No	168	No	0.07
Yes	Yes	172	No	0.07

…and if the number is between **0.21** and **0.70** (**0.21** + **0.49** = **0.70**), then we would put this sample into the new collection of samples…

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease

AdaBoost

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	0.07
No	Yes	180	Yes	0.07
Yes	No	210	Yes	0.07
Yes	Yes	167	Yes	0.49
No	Yes	156	No	0.07
No	Yes	125	No	0.07
Yes	No	168	No	0.07
Yes	Yes	172	No	0.07

For example, imagine the first number I picked was **0.72**...

...then I would put this sample into my new collection of samples...

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease

AdaBoost

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	0.07
No	Yes	180	Yes	0.07
Yes	No	210	Yes	0.07
Yes	Yes	167	Yes	0.49
No	Yes	156	No	0.07
No	Yes	125	No	0.07
Yes	No	168	Yes	0.07
Yes	Yes	172	No	0.07

Ultimately, this sample was added to the new collection of samples **4** times, reflecting its larger **Sample Weight**.

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

AdaBoost

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

Lastly, we give all the samples equal **Sample Weights**, just like before.

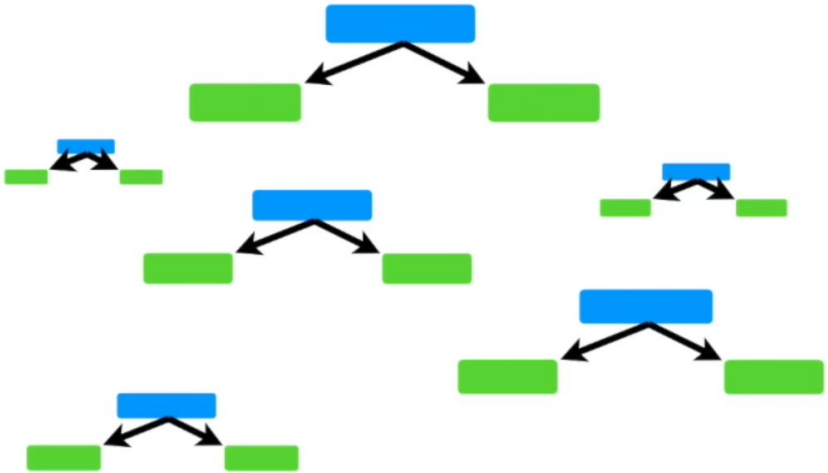
AdaBoost

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

Because these samples are all the same, they will be treated as a block, creating a large penalty for being misclassified.

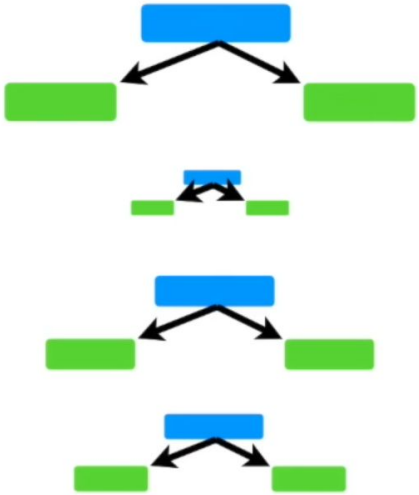
AdaBoost

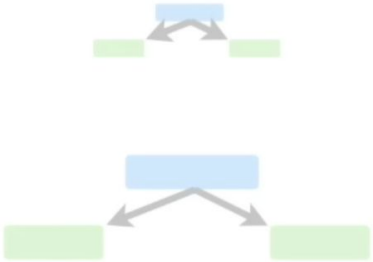
Now we need to talk about how a forest of stumps created by **AdaBoost** makes classifications…



AdaBoost

Imagine that these stumps classified a patient as **Has Heart Disease**…

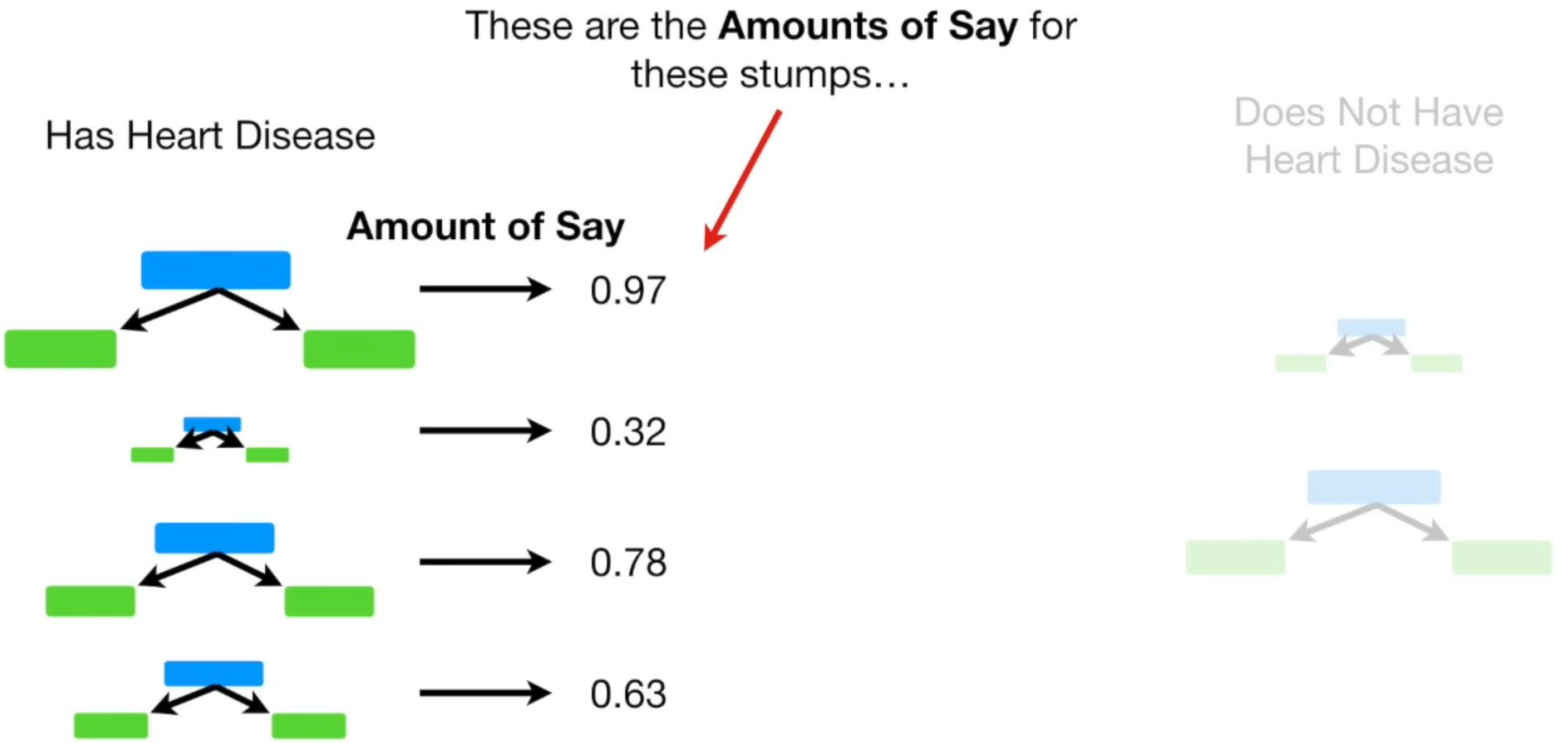




AdaBoost

…and these stumps classified the patient as **Does Not Have Heart Disease**.

AdaBoost

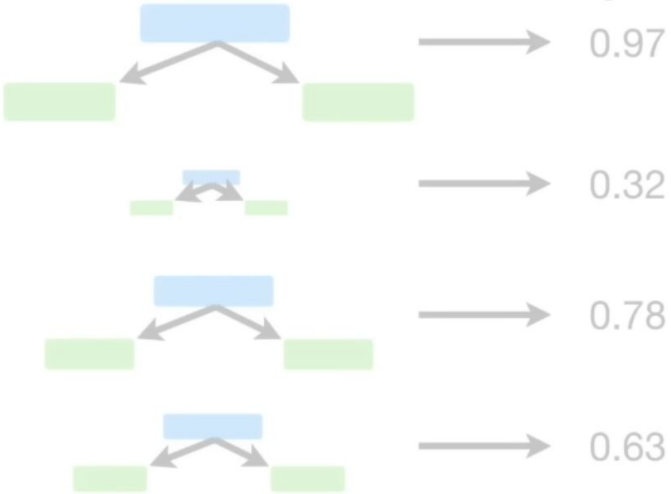

These are the **Amounts of Say** for these stumps…
Has Heart Disease
Amount of Say
0.97
0.32
0.78
0.63
Does Not Have Heart Disease

AdaBoost

…and these are the **Amounts of Say** for these stumps…

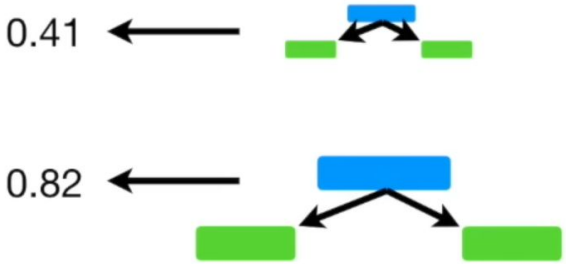
Has Heart Disease

Amount of Say



Does Not Have Heart Disease

Amount of Say

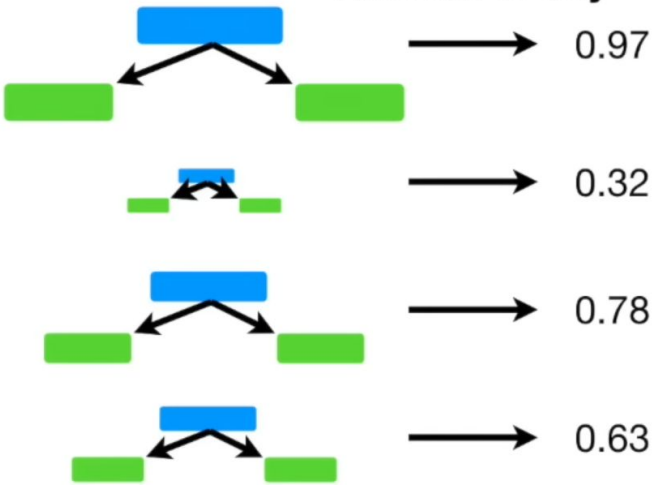


AdaBoost

Now we add up the **Amounts of Say** for this group of stumps…

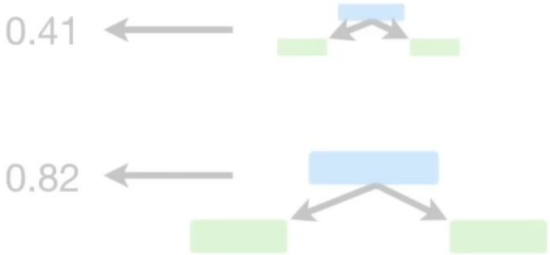
Has Heart Disease **Total = 2.7**

Amount of Say

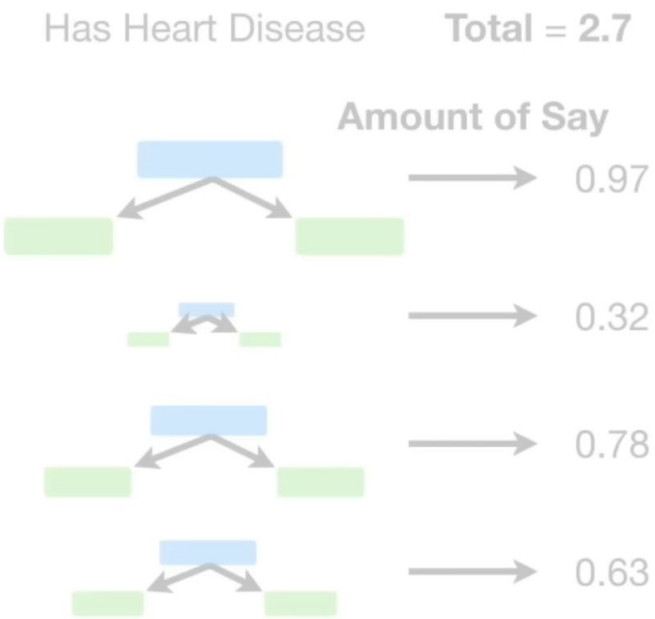


Does Not Have Heart Disease

Amount of Say



AdaBoost

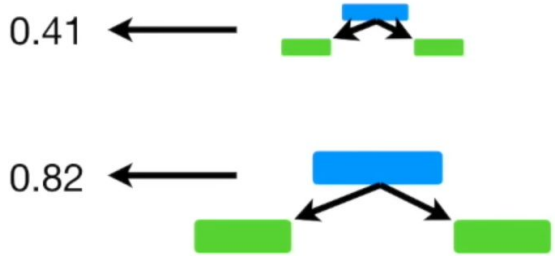


…and for this group of stumps…

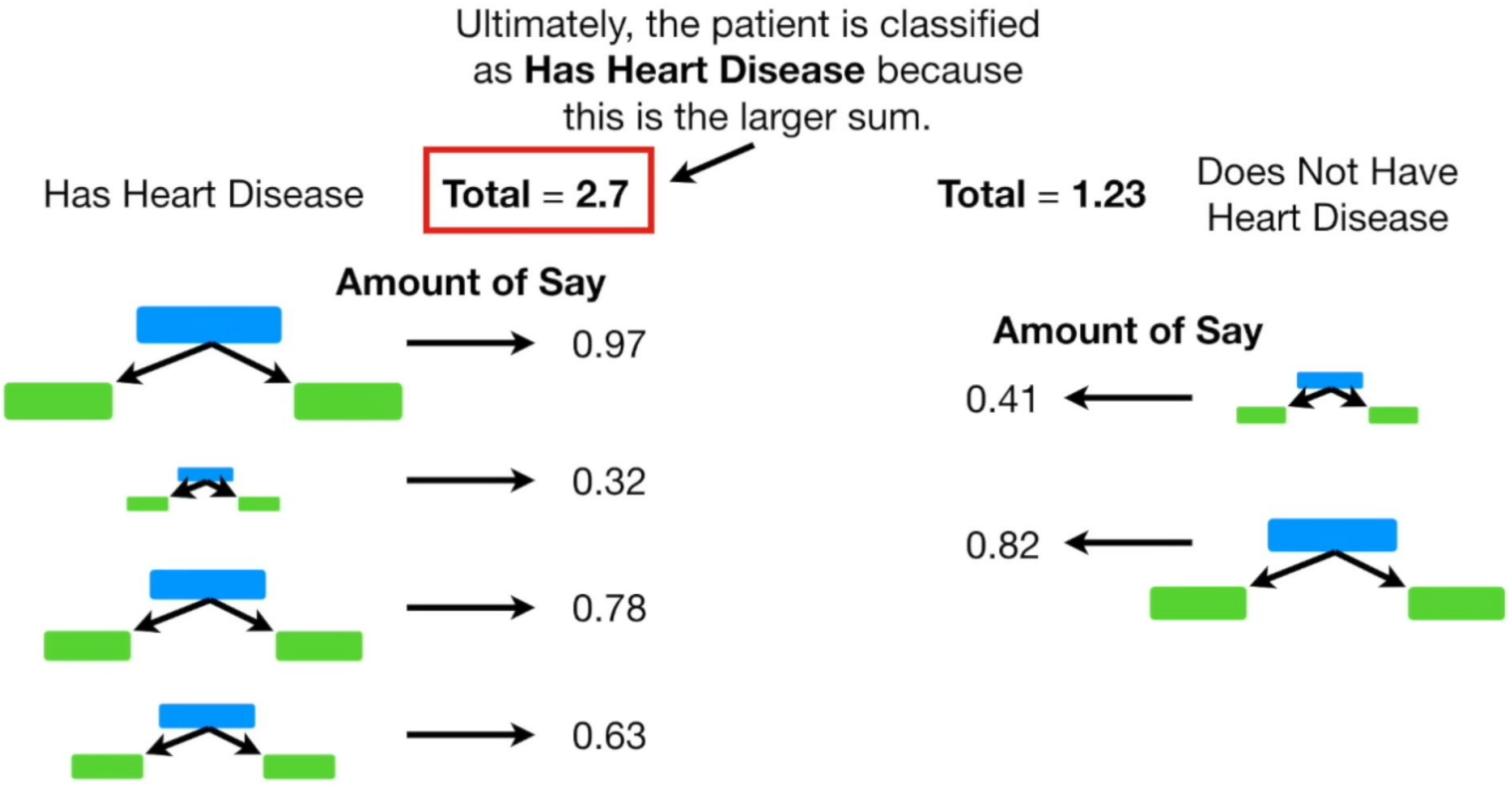
Total = 1.23

Does Not Have Heart Disease

Amount of Say



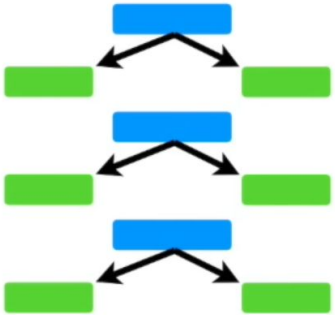
AdaBoost


Ultimately, the patient is classified as **Has Heart Disease** because this is the larger sum.
Has Heart Disease
Total = 2.7
Total = 1.23
Does Not Have Heart Disease
Amount of Say
0.97
0.32
0.78
0.63
Amount of Say
0.41
0.82

AdaBoost

To review, the three ideas behind **AdaBoost** are…

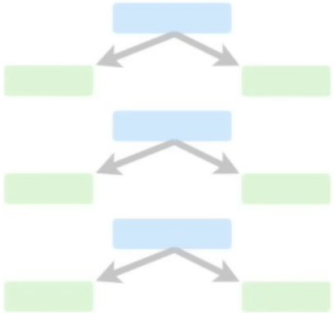
1) **AdaBoost** combines a lot of "weak learners" to make classifications. The weak learners are almost aways **stumps**.



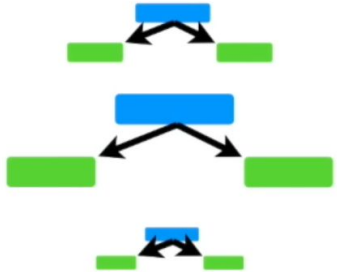
AdaBoost

To review, the three ideas behind **AdaBoost** are…

1) **AdaBoost** combines a lot of "weak learners" to make classifications. The weak learners are almost aways **stumps**.



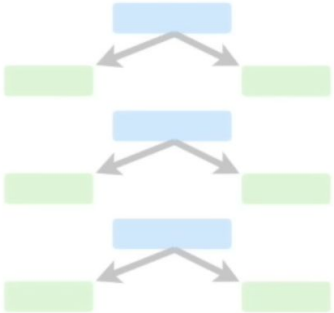
2) Some **stumps** get more say in the classification than others.



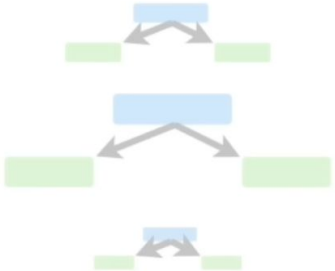
AdaBoost

To review, the three ideas behind **AdaBoost** are…

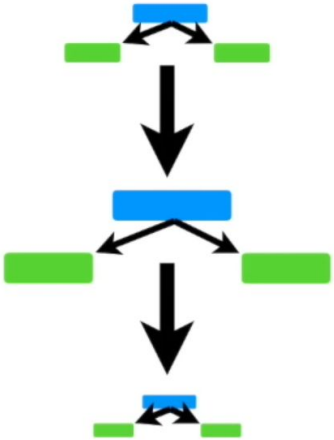
1) **AdaBoost** combines a lot of "weak learners" to make classifications. The weak learners are almost aways **stumps**.



2) Some **stumps** get more say in the classification than others.



3) Each **stump** is made by taking the previous **stump's** mistakes into account.



Gradient Boosting

Theory and Intuition

Boosting

- Gradient Boosting is a very similar idea to AdaBoost, where weak learners are created in series in order to produce a strong ensemble model.
- Gradient Boosting makes use of the residual error for learning.

Boosting

- Gradient Boosting vs. Adaboost:
  - Larger Trees allowed in Gradient Boosting.
  - Learning Rate coefficient same for all weak learners.
  - Gradual series learning is based on training on the **residuals** of the previous model.

Boosting

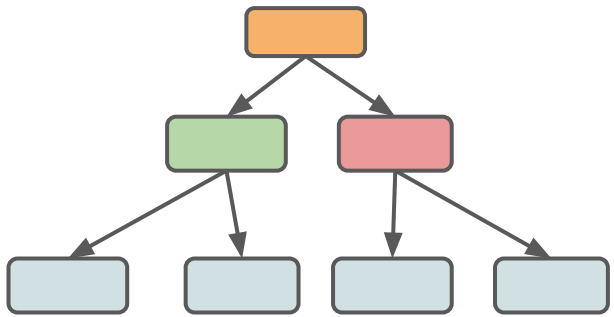
- Gradient Boosting Regression Example

Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$462,000
230	3	3	$565,000

Boosting

- Train a decision tree on data

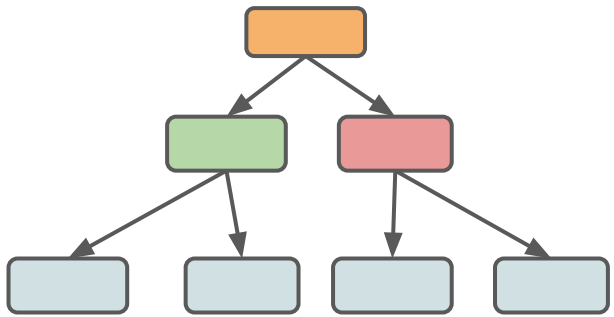
Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$462,000
230	3	3	$565,000



Boosting

- Note - not just a stump!

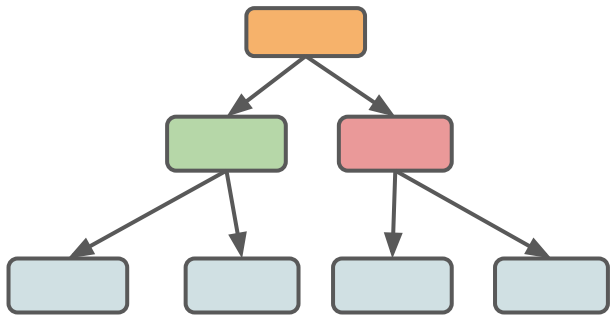
Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$462,000
230	3	3	$565,000



Boosting

- Get predicted $\hat{y}$ value

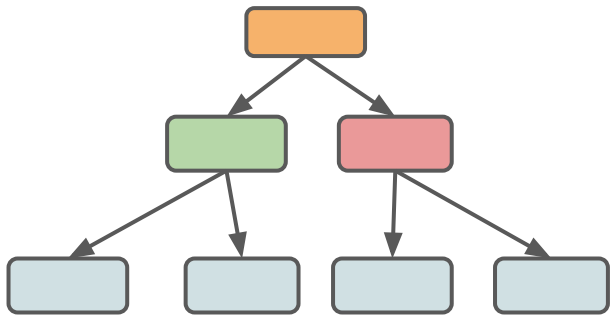
Area m^2	Bedrooms	Bathrooms	Price
200	3	2	$500,000
190	2	1	$462,000
230	3	3	$565,000



Boosting

- Get predicted $\hat{y}$ value

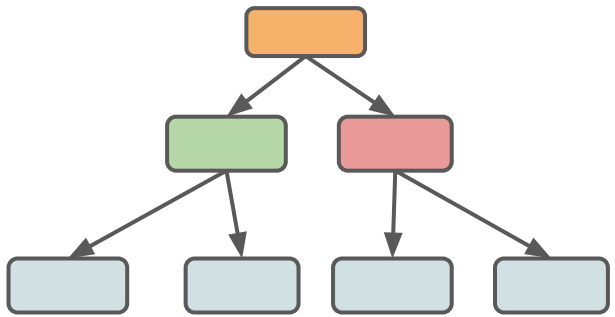
Area m^2	Bedrooms	Bathrooms	y
200	3	2	$500,000
190	2	1	$462,000
230	3	3	$565,000



Boosting

- Get predicted **ŷ** value

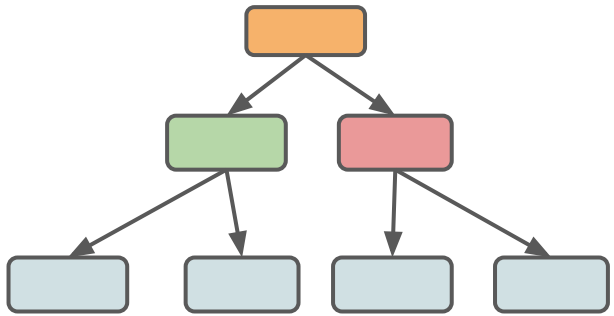
y	ŷ
$500,000	$509,000
$462,000	$509,000
$565,000	$509,000



Boosting

- Calculate residual: **e = y-ŷ**

y	ŷ	e
$500,000	$509,000	-$9,000
$462,000	$509,000	-$47,000
$565,000	$509,000	$56,000



Boosting

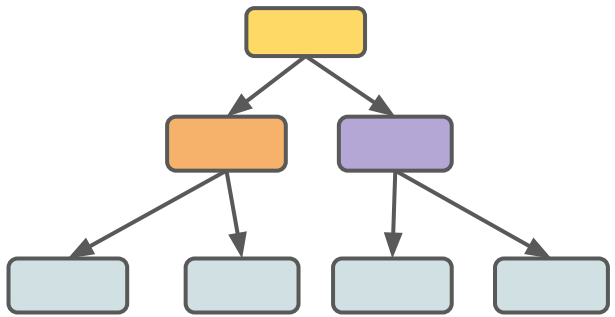
- Create new model to predict the **error**

y	ŷ	e
$500,000	$509,000	-$9,000
$462,000	$509,000	-$47,000
$565,000	$509,000	$56,000

Boosting

- Create new model to predict the **error**

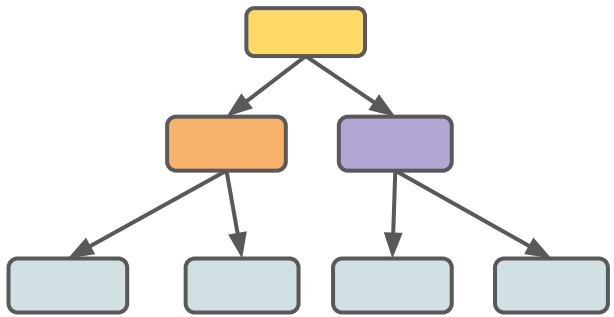
y	ŷ	e
$500,000	$509,000	-$9,000
$462,000	$509,000	-$47,000
$565,000	$509,000	$56,000



Boosting

- Create new model to predict the **error**

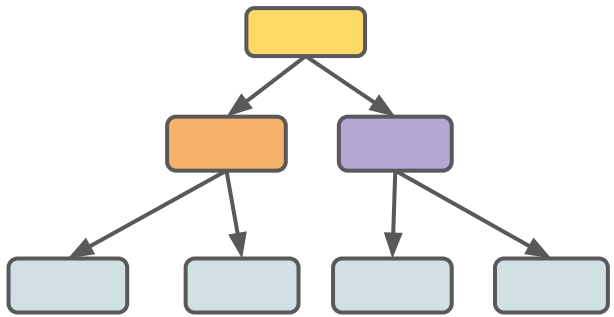
y	ŷ	e	f1
$500,000	$509,000	-$9,000	-$8,000
$462,000	$509,000	-$47,000	-$50,000
$565,000	$509,000	$56,000	$50,000



Boosting

- Create new model to predict the **error**

y	ŷ	e	f1
$500,000	$509,000	-$9,000	-$8,000
$462,000	$509,000	-$47,000	-$50,000
$565,000	$509,000	$56,000	$50,000



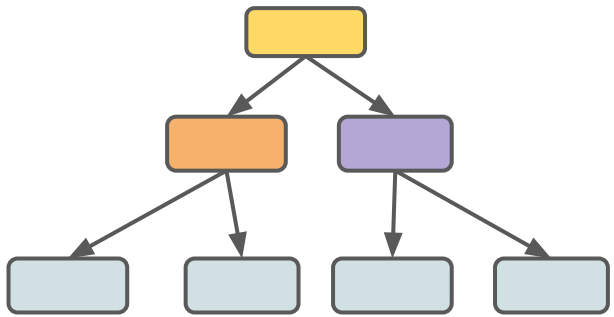
Area m^2	Bedrooms	Bathrooms
200	3	2
190	2	1
230	3	3

Boosting

- Create new model to predict the **error**

y	ŷ	e	f1
$500,000	$509,000	-$9,000	-$8,000
$462,000	$509,000	-$47,000	-$50,000
$565,000	$509,000	$56,000	$50,000

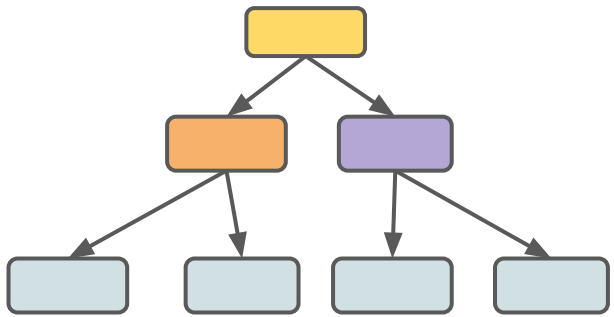
Area m^2	Bedrooms	Bathrooms
200	3	2
190	2	1
230	3	3



Boosting

- Update prediction using **error prediction**

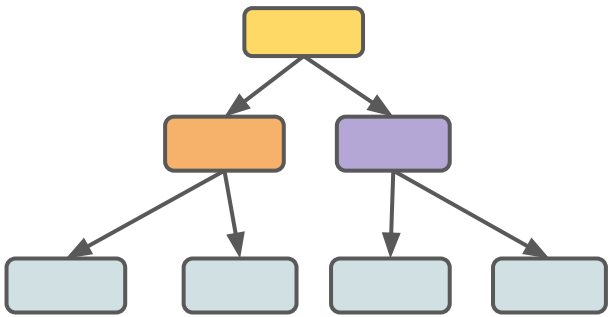
y	ŷ	e	f1
$500,000	$509,000	-$9,000	-$8,000
$462,000	$509,000	-$47,000	-$50,000
$565,000	$509,000	$56,000	$50,000



Boosting

- Update prediction using **error prediction**

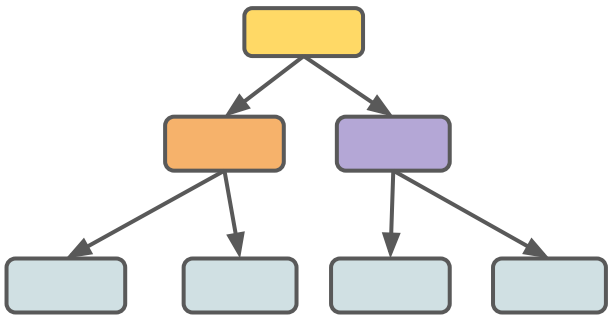
y	ŷ	e	f1	F1 = ŷ + f1
$500,000	$509,000	-$9,000	-$8,000	
$462,000	$509,000	-$47,000	-$50,000	
$565,000	$509,000	$56,000	$50,000	



Boosting

- Update prediction using **error prediction**

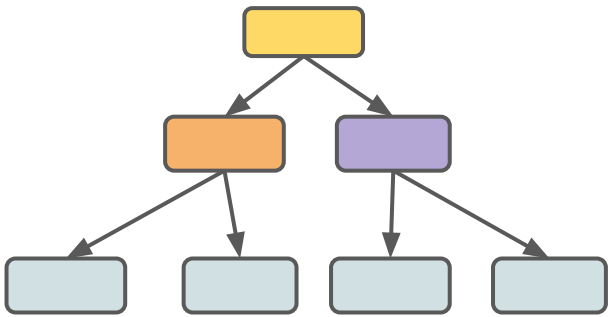
y	ŷ	e	f1	F1 = ŷ + f1
$500,000	$509,000	-$9,000	-$8,000	$501,000
$462,000	$509,000	-$47,000	-$50,000	$459,000
$565,000	$509,000	$56,000	$50,000	$559,000



Boosting

- We can continue this process in series

y	ŷ	e	f1	F1 = ŷ + f1
$500,000	$509,000	-$9,000	-$8,000	$501,000
$462,000	$509,000	-$47,000	-$50,000	$459,000
$565,000	$509,000	$56,000	$50,000	$559,000



Boosting

- Gradient Boosting Process

$$F_m = F_{m-1} + f_m$$

Boosting

- Gradient Boosting Process

$$F_m = F_{m-1} + (\text{learning rate} * f_m)$$

Boosting

- Gradient Boosting Process
  - Create initial model: f_0
  - Train another model on error
    - $e = y - f_0$
  - Create new prediction
    - $F_1 = f_0 + \eta f_1$
  - Repeat as needed
    - $F_m = f_{m-1} + \eta f_m$