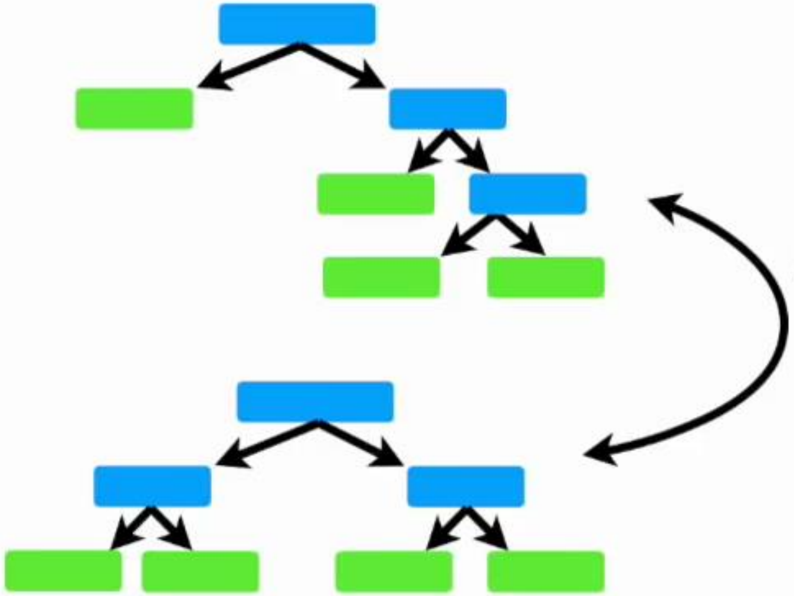
Gradient Boost

DISCLAIMER

The following content/slides/animations are from YouTube channel named

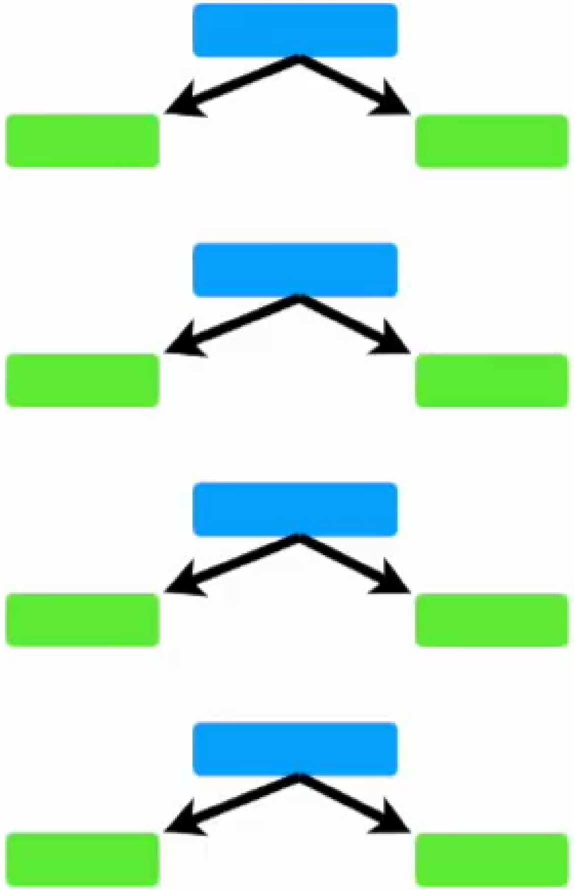
StatQuest with Josh Starmer

In a **Random Forest**, each time you make a tree, you make a full sized tree.

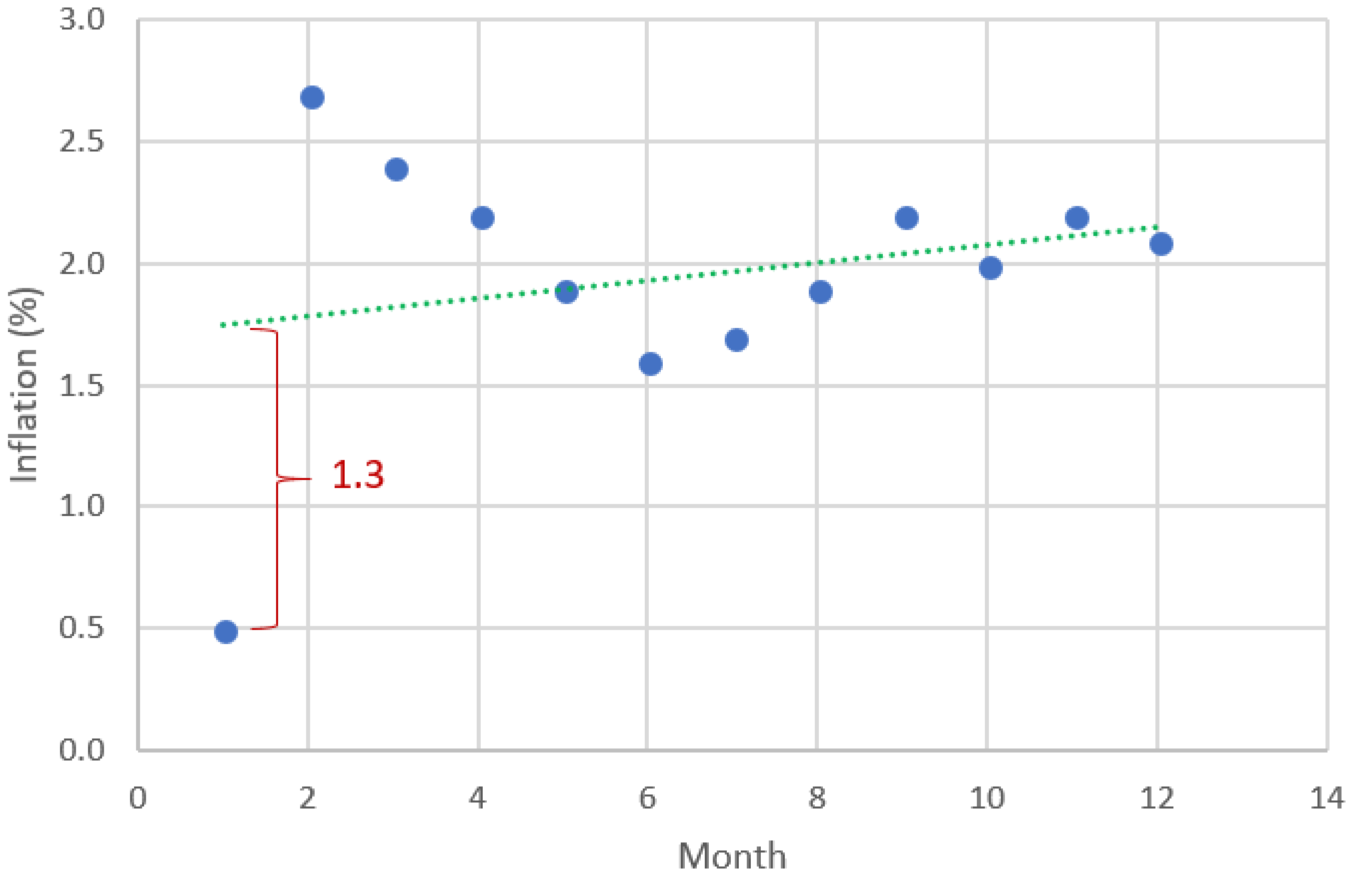


Some trees might be bigger than others, but there is no predetermined maximum depth.

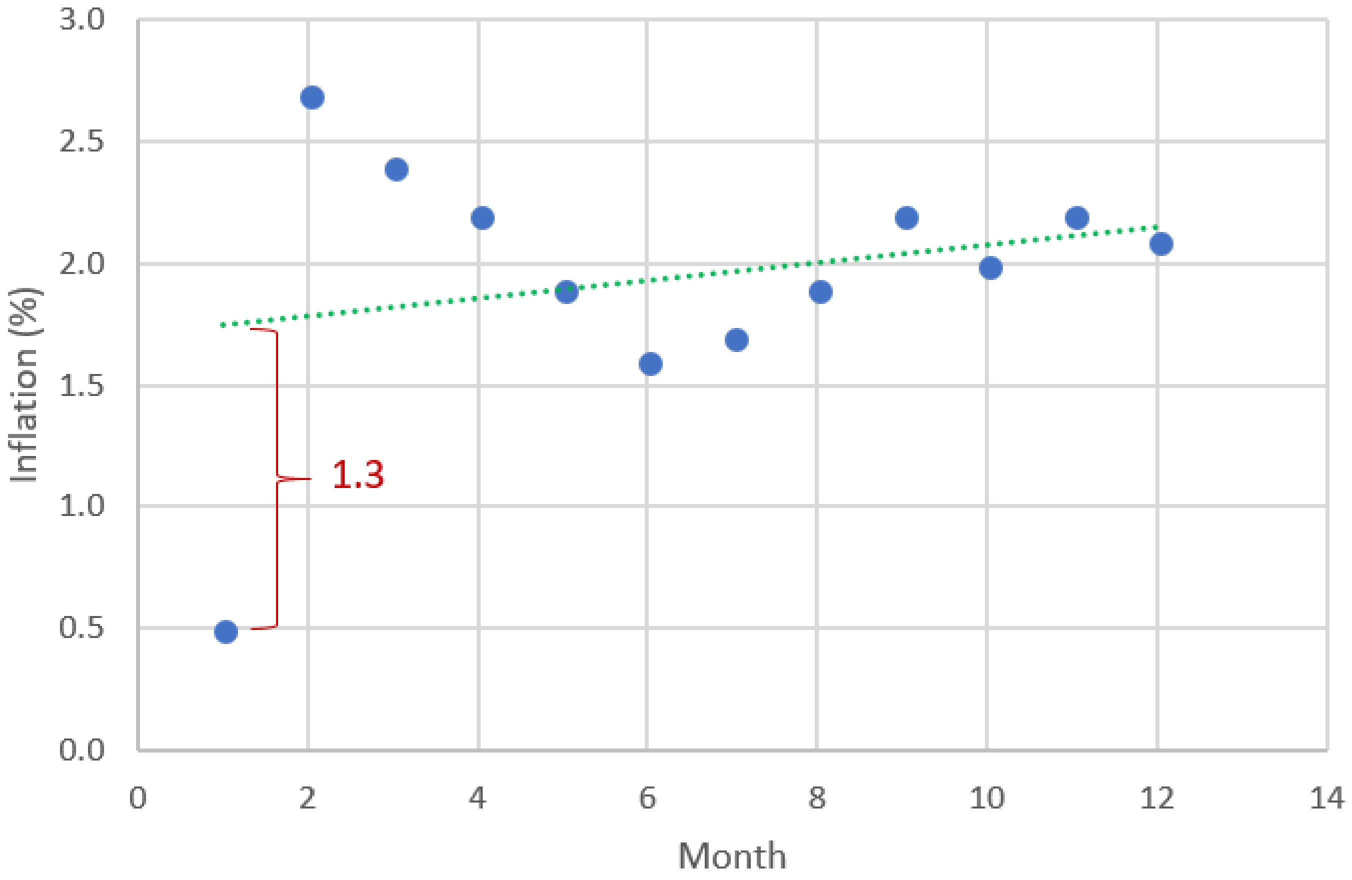
In contrast, in a **Forest of Trees** made with **AdaBoost**, the trees are usually just a **node** and two **leaves**.



Residual



Residual Learning

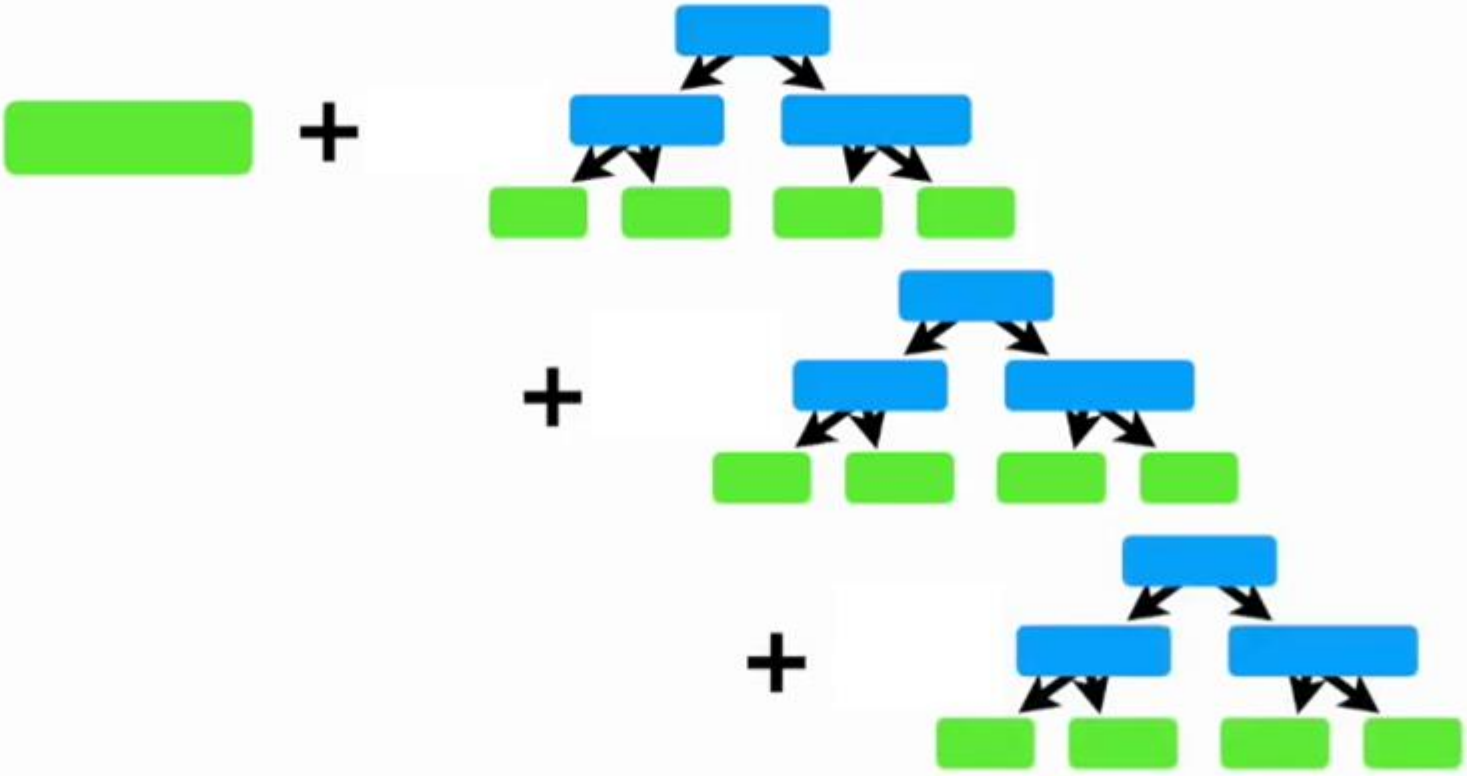


Gradient Boost Regression

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57

Gradient Boost

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57

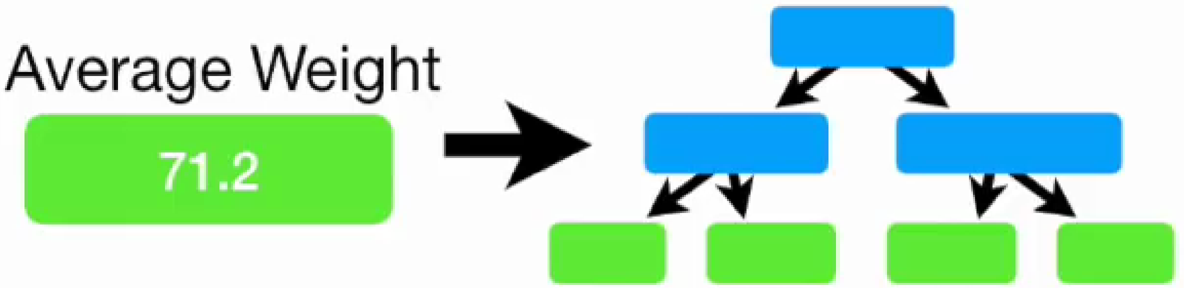


Average Weight

71.2

The first thing we do is calculate the average **Weight.**

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57



Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57

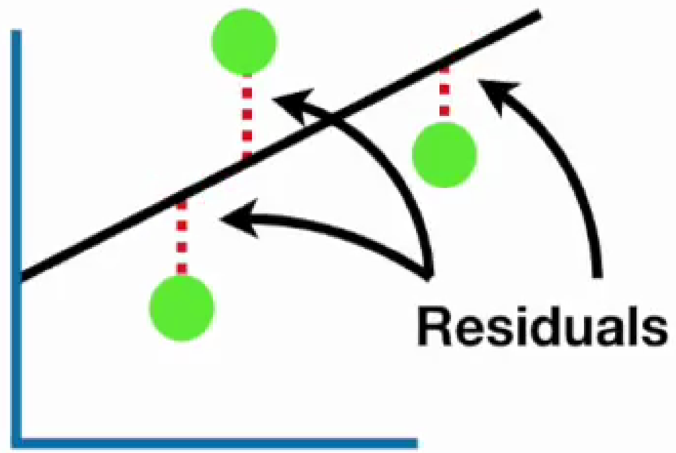
The next thing we do is build a tree based on the errors from the first tree.

Average Weight

71.2

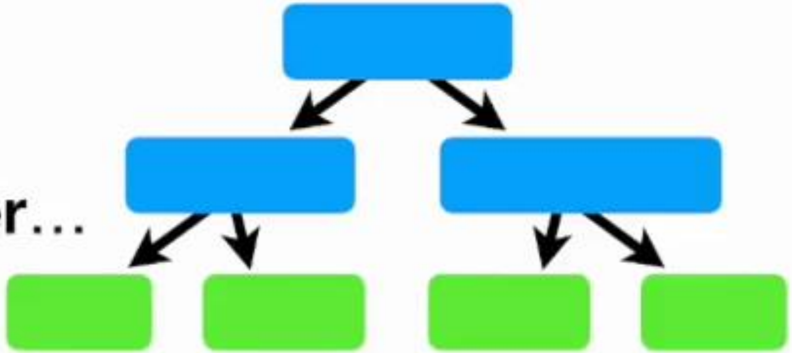
NOTE: The term **Pseudo Residual** is based on **Linear Regression**, where the difference between the **Observed** values and the **Predicted** values results in **Residuals**.

Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	16.8
1.6	Green	Female	76	
1.5	Blue	Female	56	
1.8	Red	Male	73	
1.5	Green	Male	77	
1.4	Blue	Female	57	



The "**Pseudo**" part of **Pseudo Residual** is a reminder that we are doing **Gradient Boost**, not **Linear Regression**, and is something I'll talk more about in **Part 2** of this series when we go through the math.

Now we will build a **Tree**, using **Height**, **Favorite Color** and **Gender**…



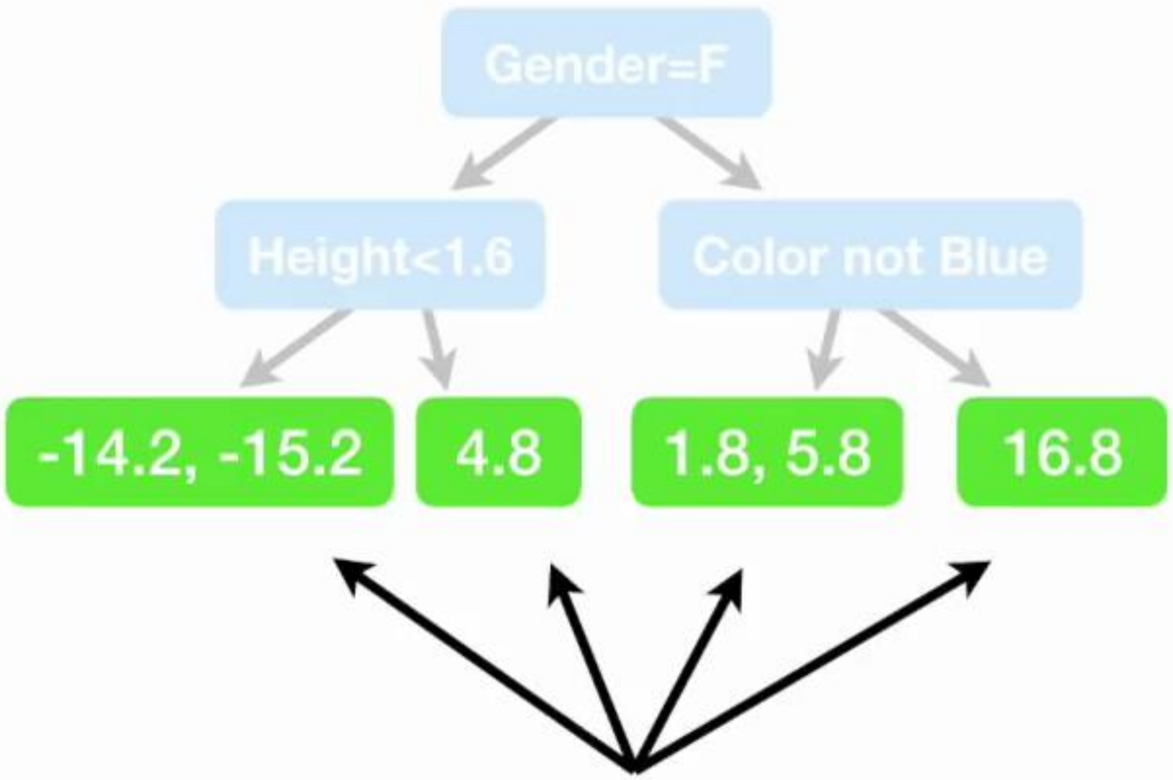
Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	16.8
1.6	Green	Female	76	4.8
1.5	Blue	Female	56	-15.2
1.8	Red	Male	73	1.8
1.5	Green	Male	77	5.8
1.4	Blue	Female	57	-14.2

…to **Predict** the **Residuals**.

Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	16.8
1.6	Green	Female	76	4.8
1.5	Blue	Female	56	-15.2
1.8	Red	Male	73	1.8
1.5	Green	Male	77	5.8
1.4	Blue	Female	57	-14.2

By restricting the total number of leaves, we get fewer leaves than **Residuals**.

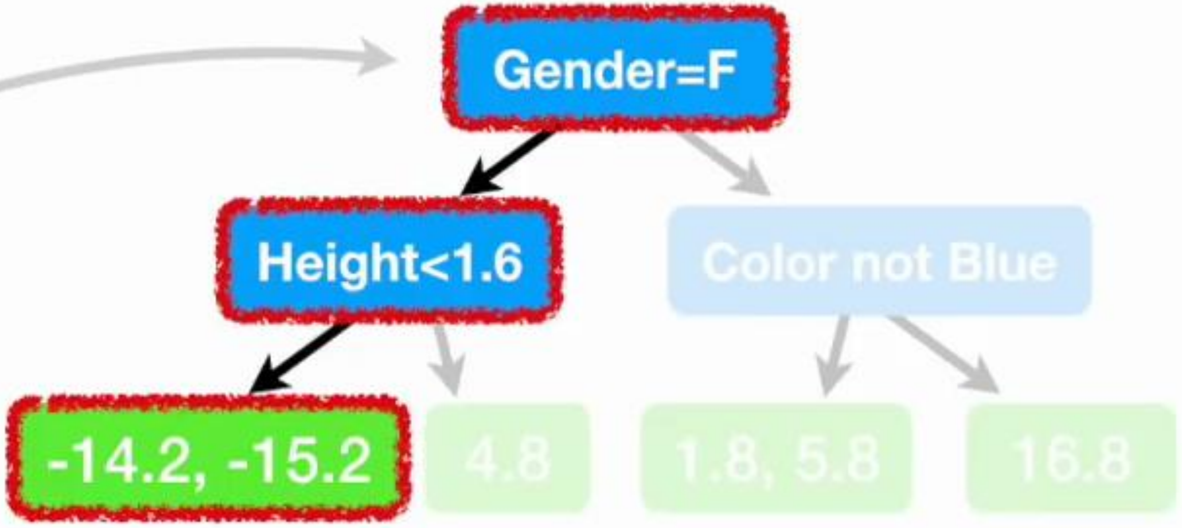
Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	16.8
1.6	Green	Female	76	4.8
1.5	Blue	Female	56	-15.2
1.8	Red	Male	73	1.8
1.5	Green	Male	77	5.8
1.4	Blue	Female	57	-14.2



Remember, in this example we are only allowing up to four leaves…

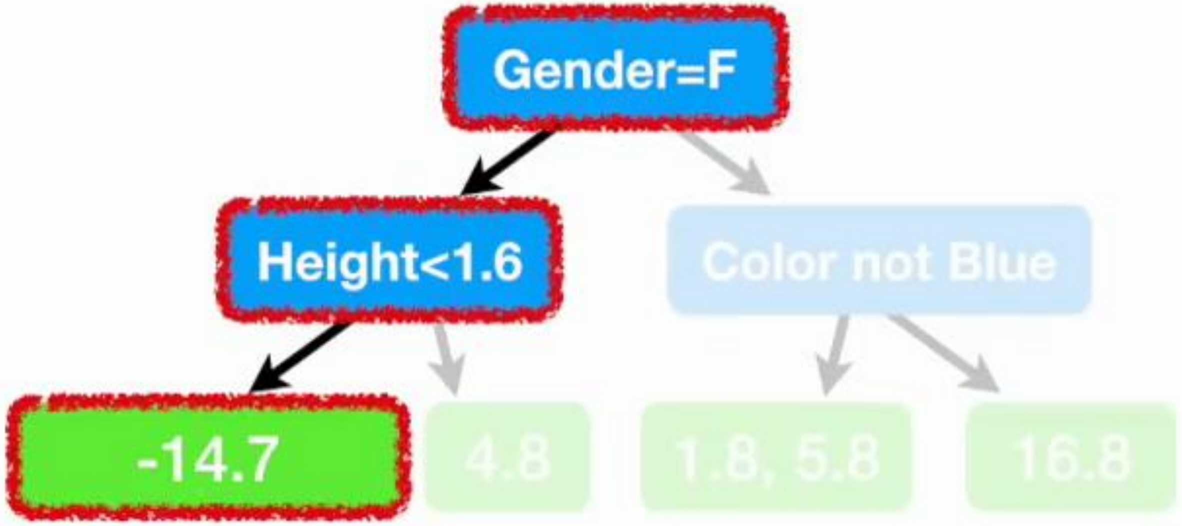
…but when using a larger dataset, it is common to allow anywhere from **8** to **32**.

Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	[illegible]	16.8
1.6	Green	Female	76	4.8
1.5	Blue	Female	56	-15.2
1.8	Red	Male	73	1.8
1.5	Green	Male	77	5.8
1.4	Blue	Female	57	-14.2



As a result, these two rows of data go to the same leaf.

Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	16.8
1.6	Green	Female	76	4.8
1.5	Blue	Female	56	-15.2
1.8	Red	Male	73	1.8
1.5	Green	Male	77	5.8
1.4	Blue	Female	57	-14.2



So we replace these residuals with their average.

$$\frac{(-14.2 + -15.2)}{2} = -14.7$$

Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	16.8
1.6	Green	Female	76	4.8
1.5	Blue	Female	56	-15.2
1.8	Red	Male	73	1.8
1.5	Green	Male	77	5.8
1.4	Blue	Female	57	-14.2

So we replace these residuals with their average.

$$\frac{(1.8 + 5.8)}{2} = 3.8$$

Average Weight

71.2

We start with the initial **Prediction**, **71.2**…

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88

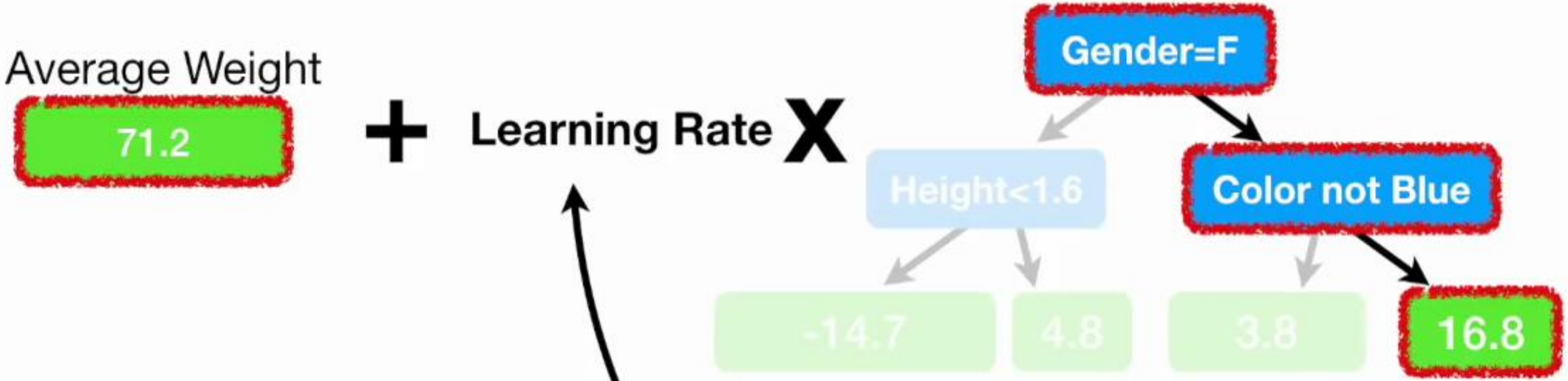
Average Weight

71.2

+

…so the **Predicted Weight** = 71.2 + 16.8 = 88

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88



Gradient Boost deals with this problem by using a **Learning Rate** to scale the contribution from the new tree.

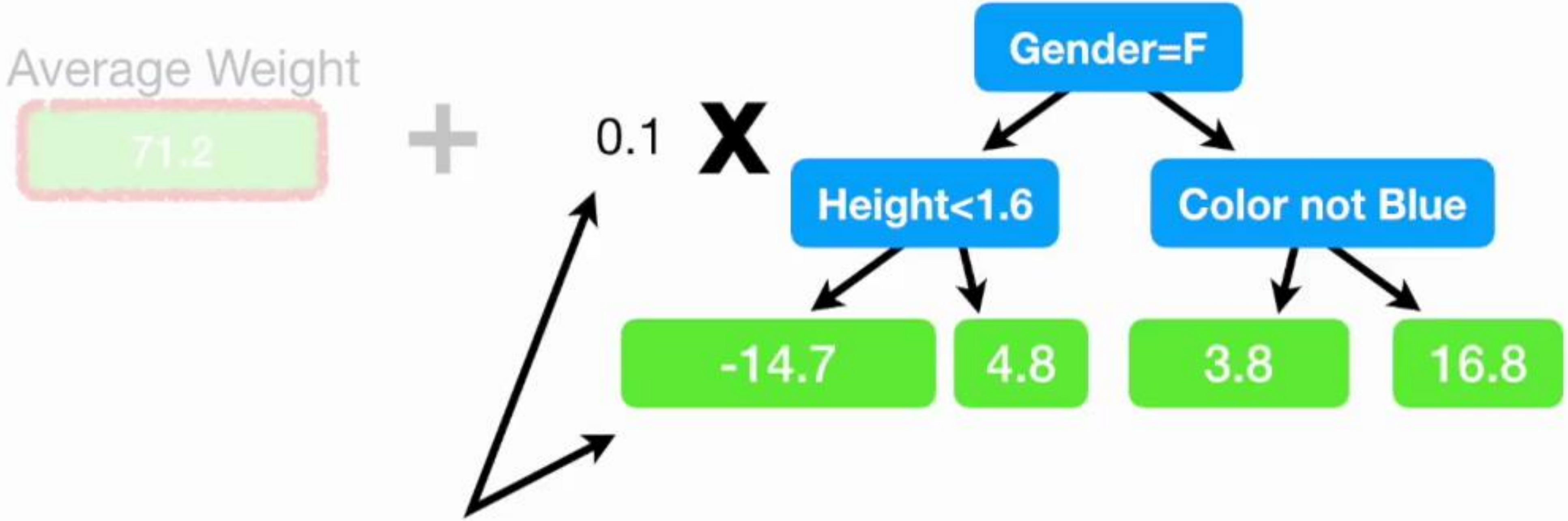
The **Learning Rate** is a value between **0** and **1**.

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88

Predicted Weight = 71.2 + (0.1 × 16.8) $\boxed{= 72.9}$

Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88

With the **Learning Rate** set to **0.1**, the new **Prediction** isn't as good as as it was before…



In other words, scaling the tree by the **Learning Rate** results in a small step in the right direction.

empirical evidence shows that taking lots of small steps in the right direction results in better **Predictions** with a **Testing Dataset**, i.e. lower **Variance**.

Average Weight

71.2 + 0.1 X

Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	15.1
1.6	Green	Female	76	
1.5	Blue	Female	56	
1.8	Red	Male	73	
1.5	Green	Male	77	
1.4	Blue	Female	57	

Residual = (**88** - (71.2 + 0.1 × 16.8))

$\boxed{= 15.1}$

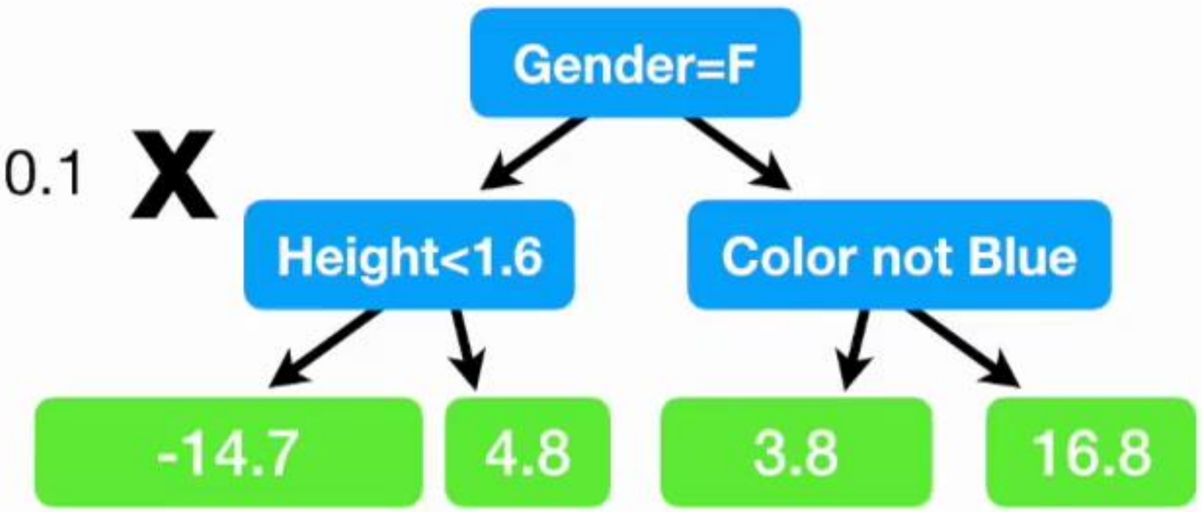
…and we save that in the column for **Pseudo Residuals**.

Average Weight

71.2

+

0.1 X

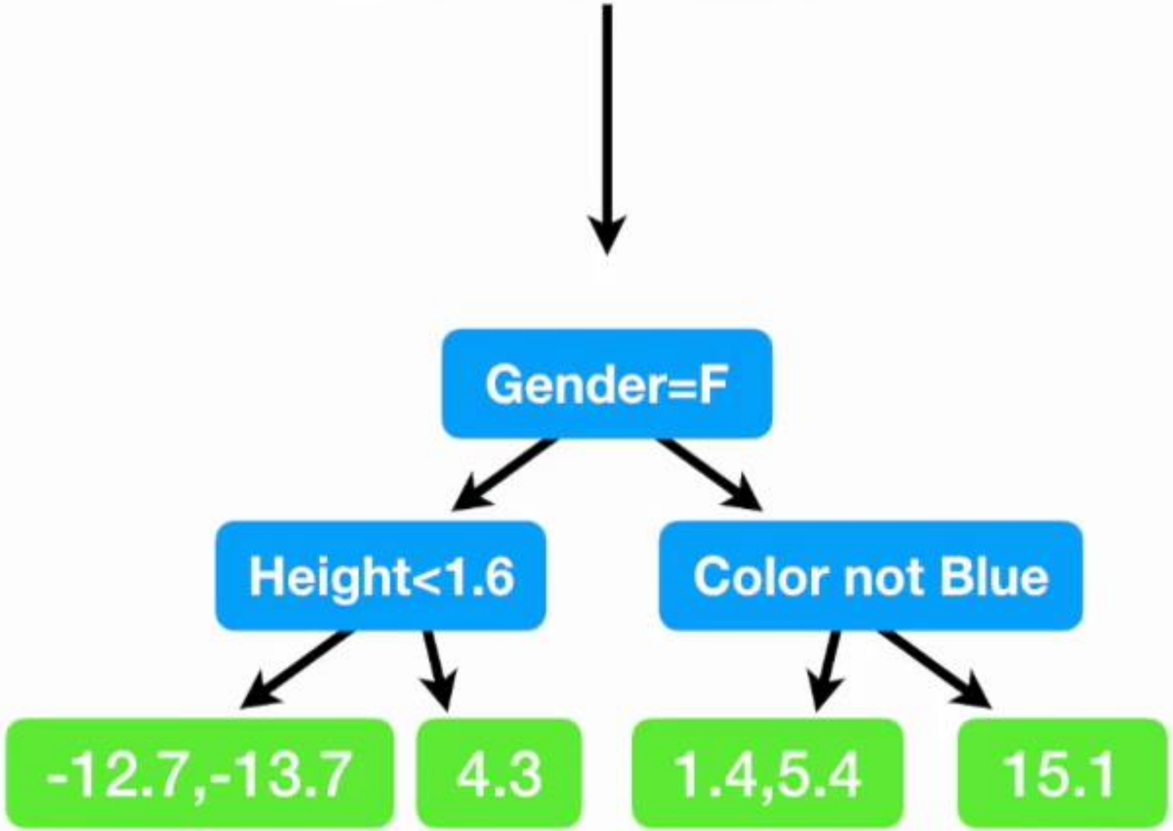


Residual	Residual
16.8	15.1
4.8	4.3
-15.2	-13.7
1.8	1.4
5.8	5.4
-14.2	-12.7

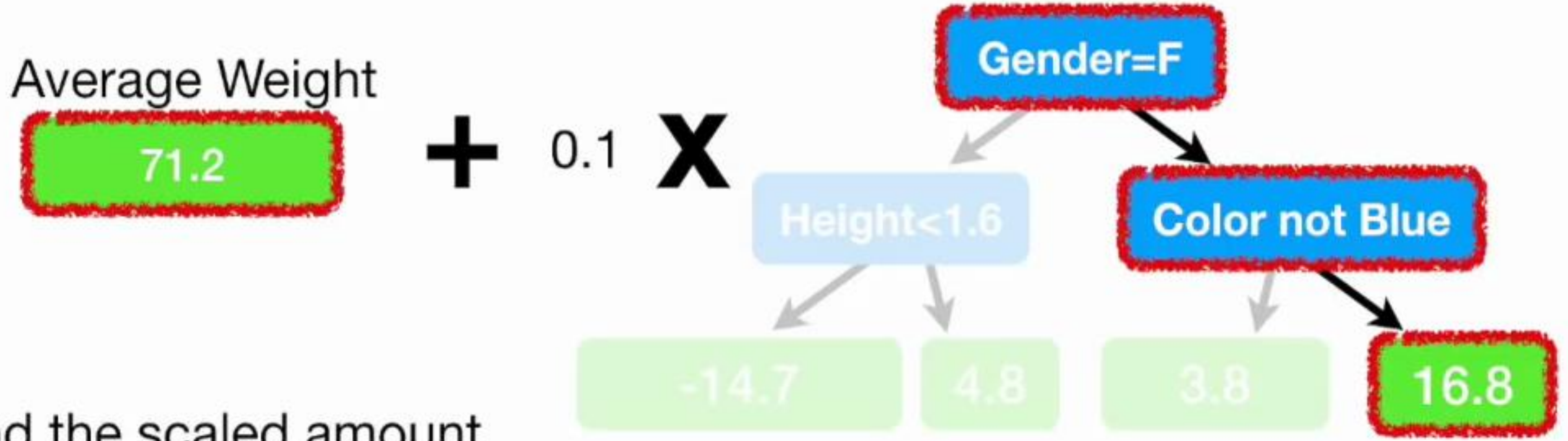
...and these are the **Residuals** after adding the new tree scaled by the **Learning Rate**.

The new **Residuals** are all smaller than before, so we've taken a small step in the right direction.

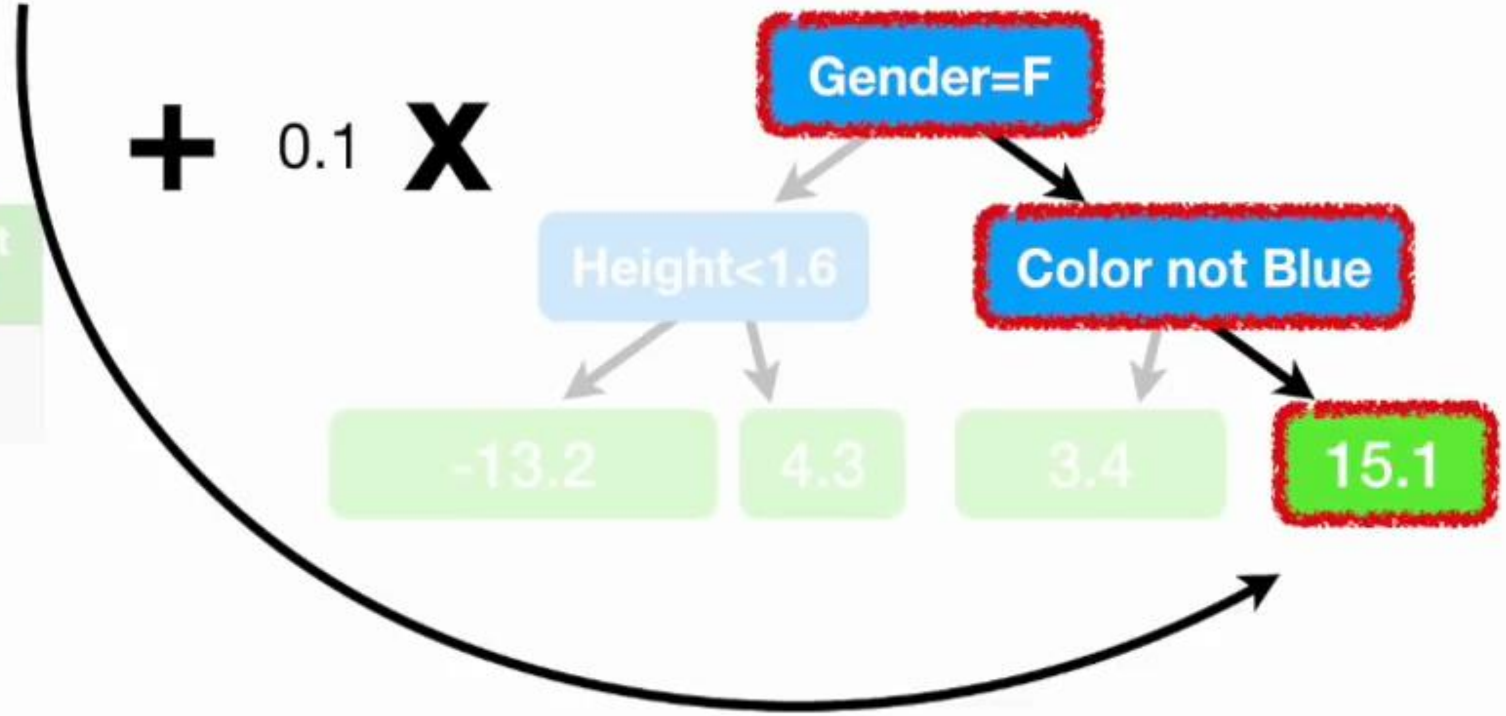
And here's the new tree!



Height (m)	Favorite Color	Gender	Weight (kg)	Residual
1.6	Blue	Male	88	15.1
1.6	Green	Female	76	4.3
1.5	Blue	Female	56	-13.7
1.8	Red	Male	73	1.4
1.5	Green	Male	77	5.4
1.4	Blue	Female	57	-12.7



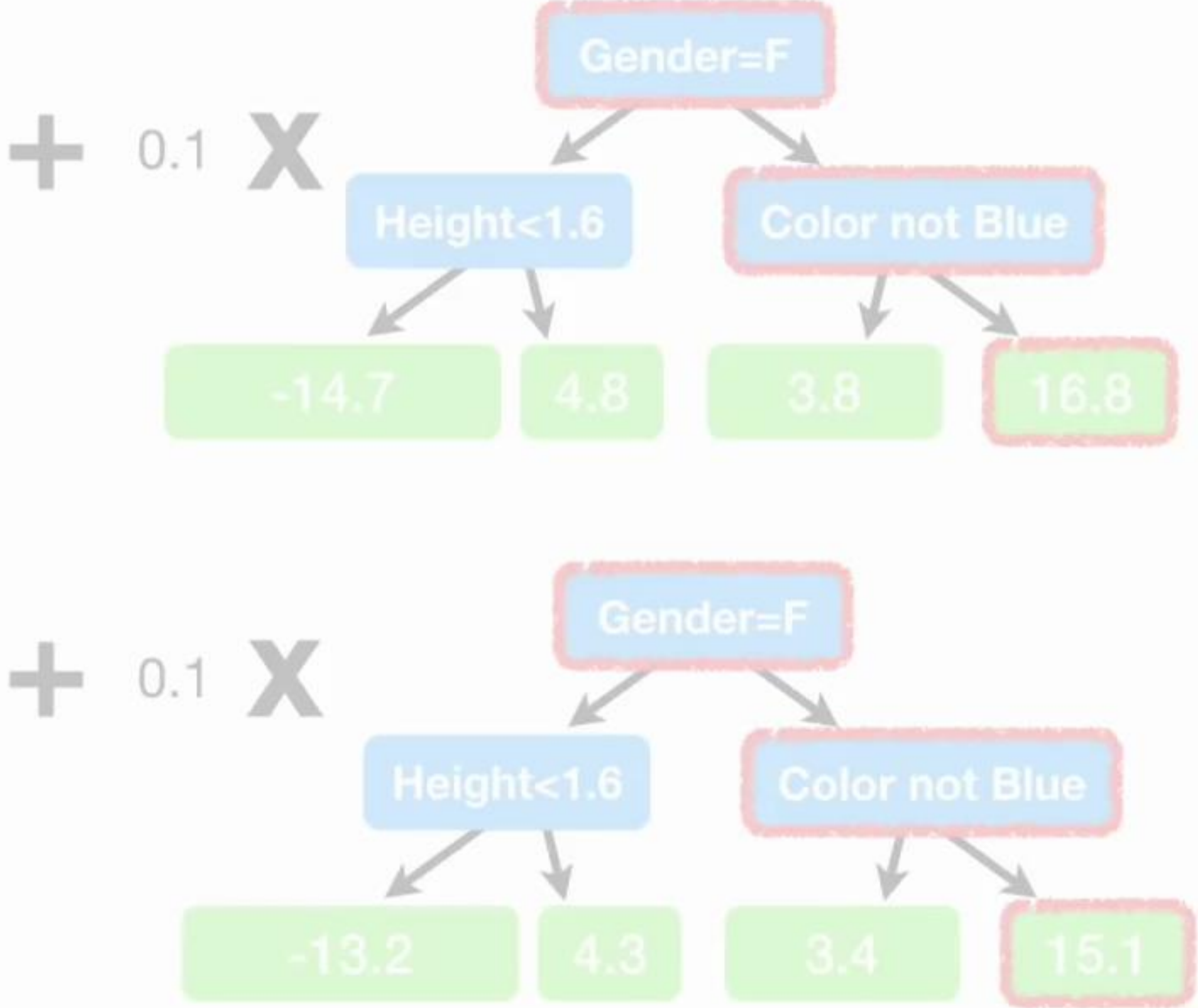
...and the scaled amount
from the second **Tree**.



Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88

Average Weight

71.2

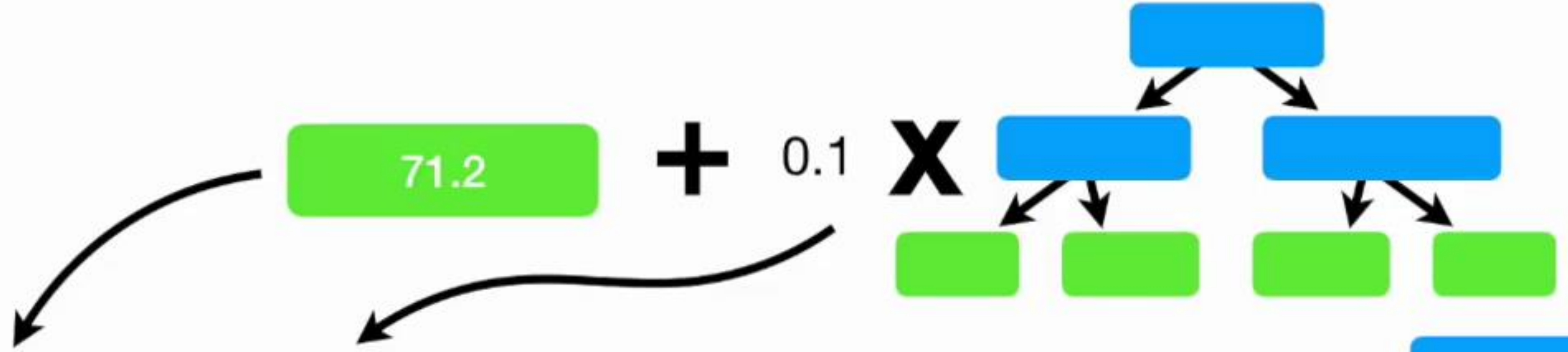


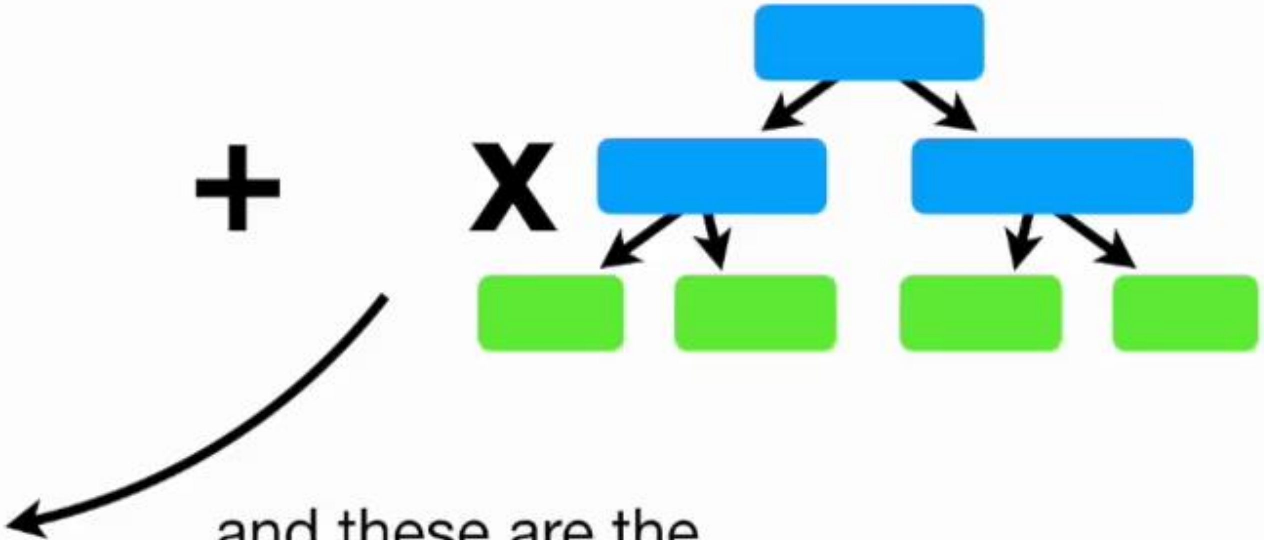
Which is another small step
closer to the **Observed Weight**.

$$71.2 + (0.1 \times 16.8) + (0.1 \times 15.1)$$

$$\boxed{= 74.4}$$

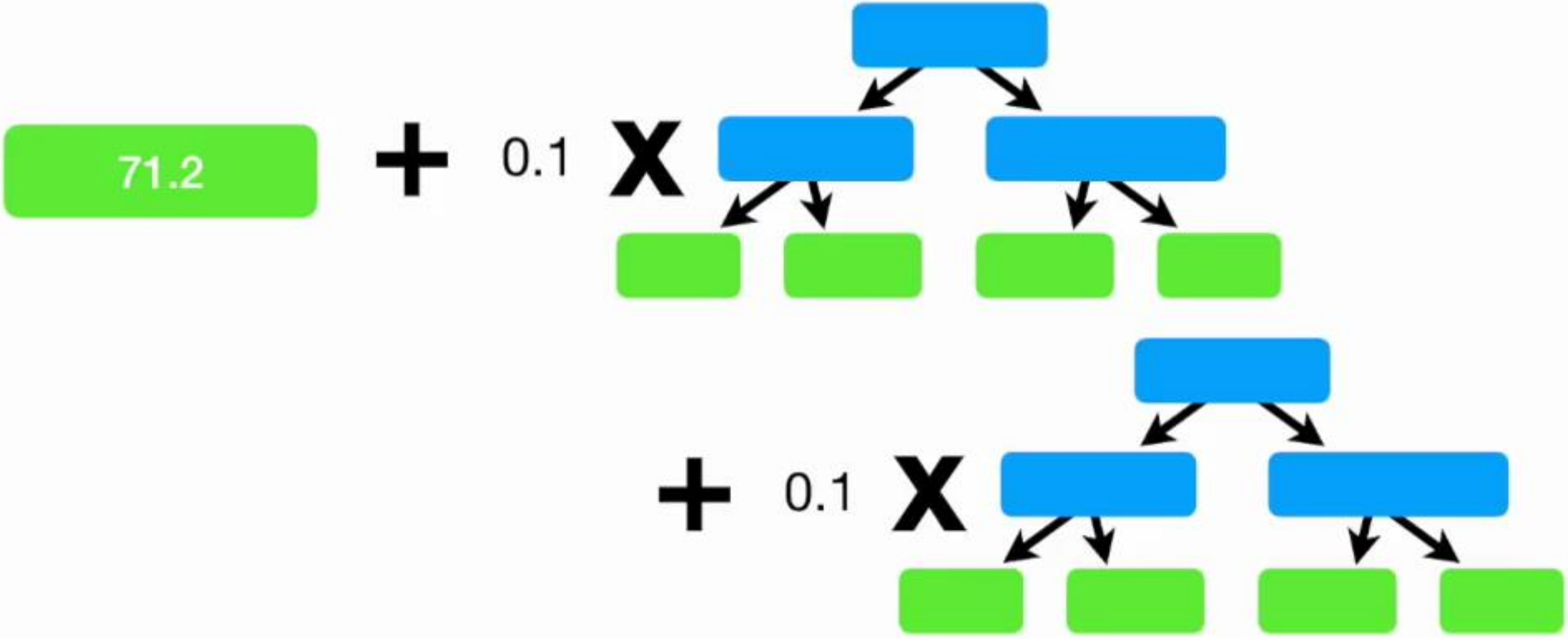
Height (m)	Favorite Color	Gender	Weight (kg)
1.6	Blue	Male	88



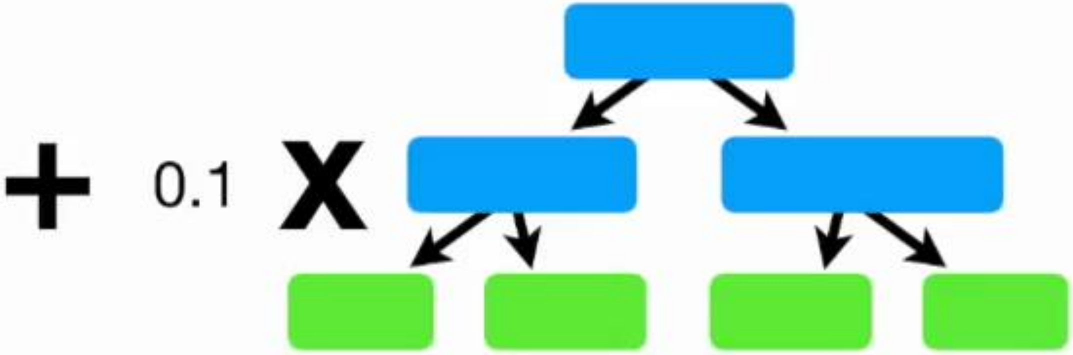


Residual	Residual	Residual
16.8	15.1	13.6
4.8	4.3	3.9
-15.2	-13.7	-12.4
1.8	1.4	1.1
5.8	5.4	5.1
-14.2	-12.7	-11.4

…and these are the **Residuals** after we added the second **Tree** to the **Prediction**…



...and we keep making trees until we reach the maximum specified, or adding additional trees does not significantly reduce the size of the **Residuals**.



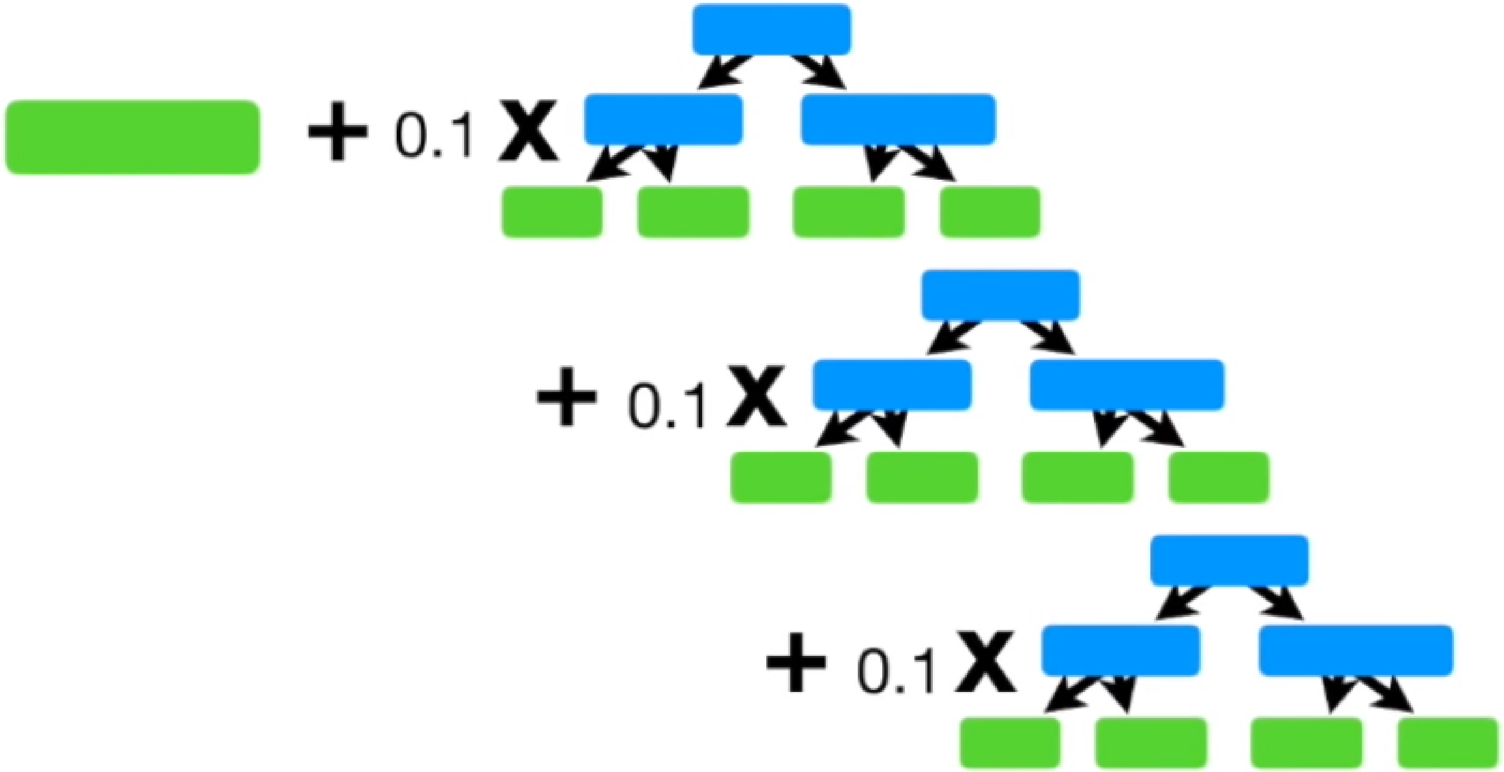
...etc...etc...etc...

Gradient Boost Classification

Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes

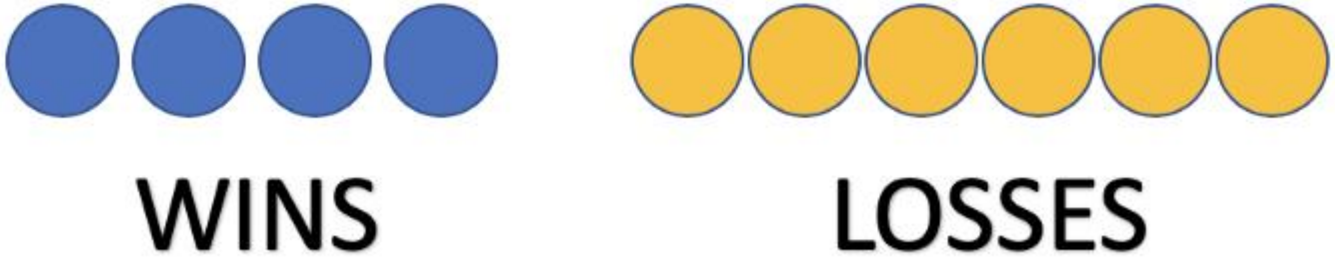
...and walk through, step-by-step, the most common way that **Gradient Boost** fits a model to this **Training Data**.

Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes

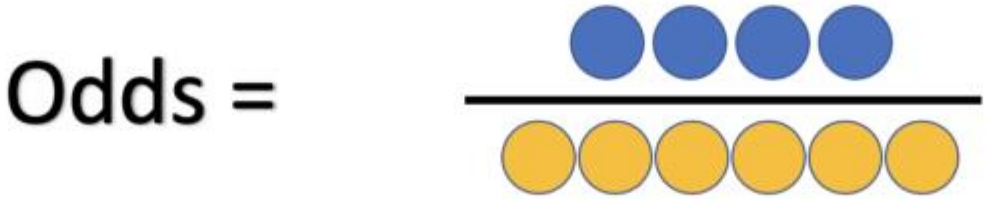


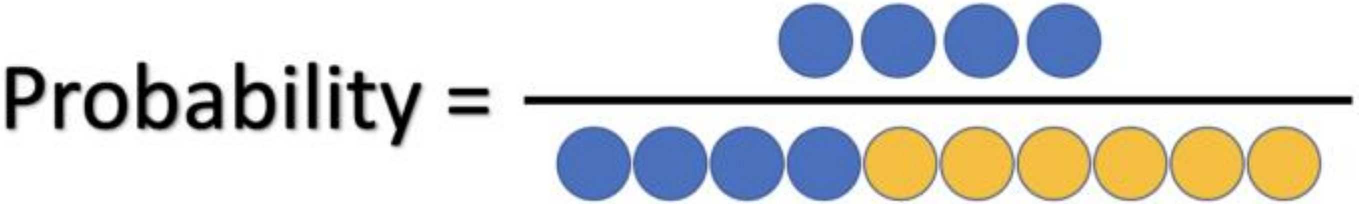
Gradient Boost Classification

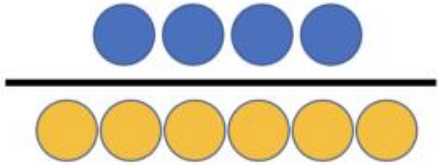
Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes

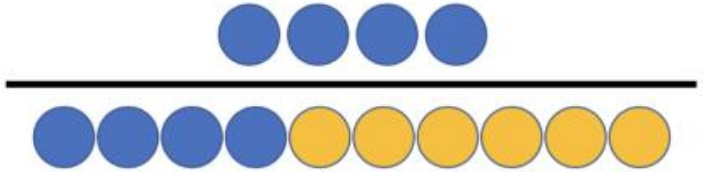


Odds are the *ratio of something happening to something not happening*.





Odds = 

Probability = 

$$\frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}}$$

$\log(4/2) = 0.7$

Just like with **Logistic Regression**, the easiest way to use the **log(odds)** for **Classification** is to convert it to a **Probability**…

Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes

…and we do that with a **Logistic Function**.

$$\textbf{Probability of Loving Troll 2} = \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}}$$

log(4/2) = 0.7

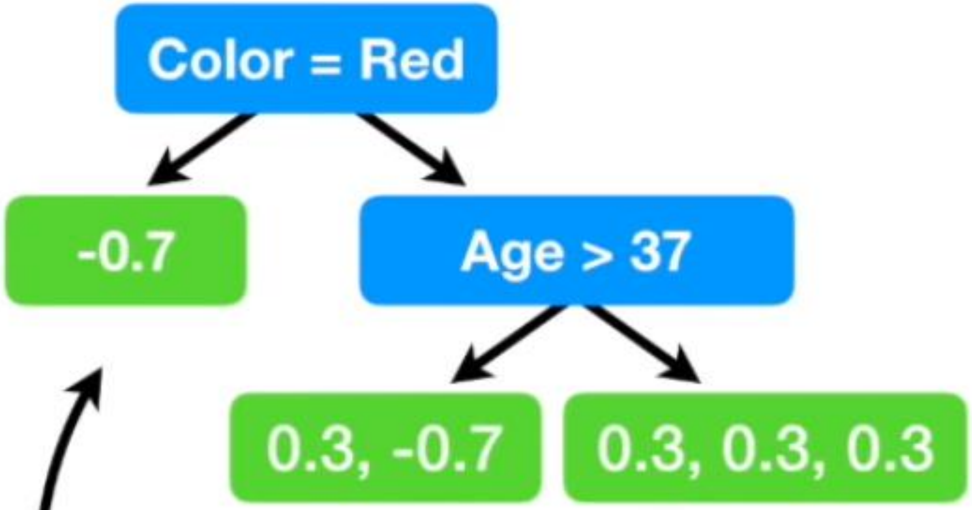
Probability of **Loving Troll 2** = 0.7

And let's save that up here for now.

Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes

$$\textbf{Probability of Loving Troll 2} = \frac{e^{\log(4/2)}}{1 + e^{\log(4/2)}} = \boxed{0.7}$$

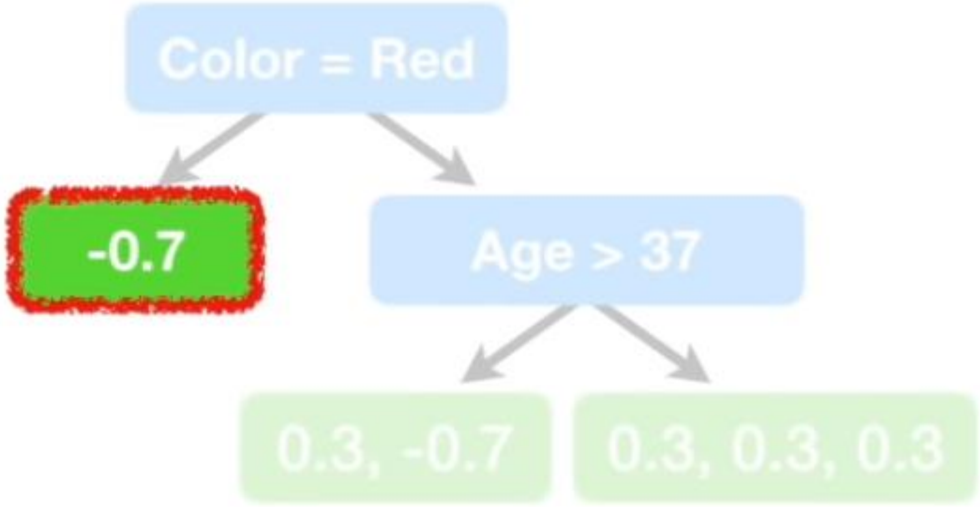
Likes Popcorn	Age	Favorite Color	Loves Troll 2	Residual
Yes	12	Blue	Yes	0.3
Yes	87	Green	Yes	0.3
No	44	Blue	No	-0.7
Yes	19	Red	No	-0.7
No	32	Green	Yes	0.3
No	14	Blue	Yes	0.3



And here's the tree!

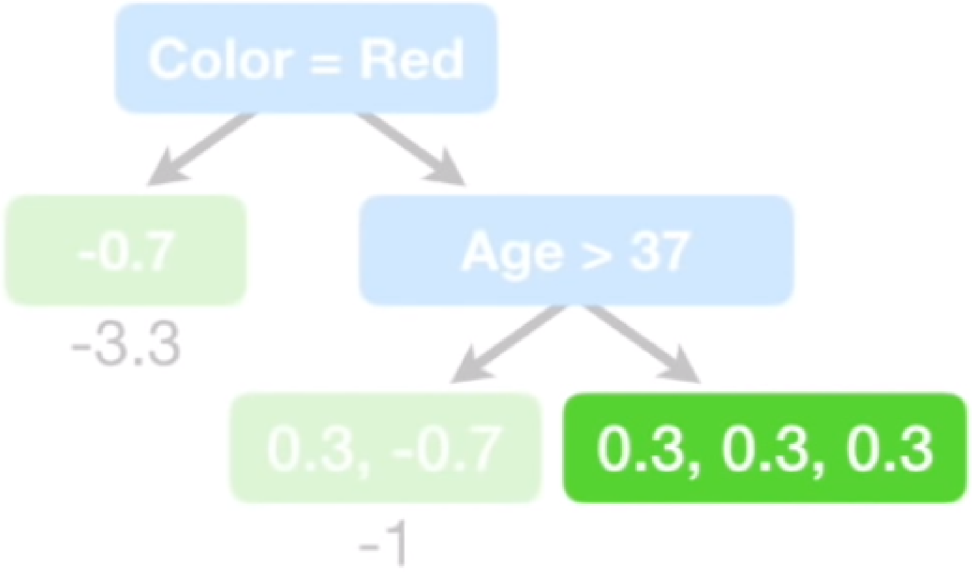
log(4/2) = 0.7

Probability of Loving Troll 2 = 0.7



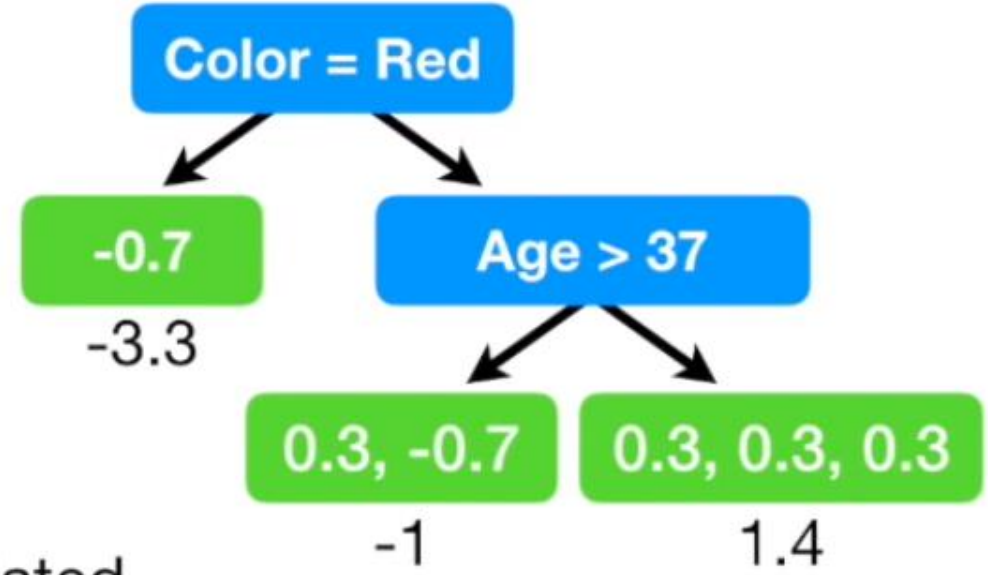
So we plug in the **Residual** from the leaf…

$$\frac{\sum \text{Residual}_i}{\sum \left[\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)\right]}$$



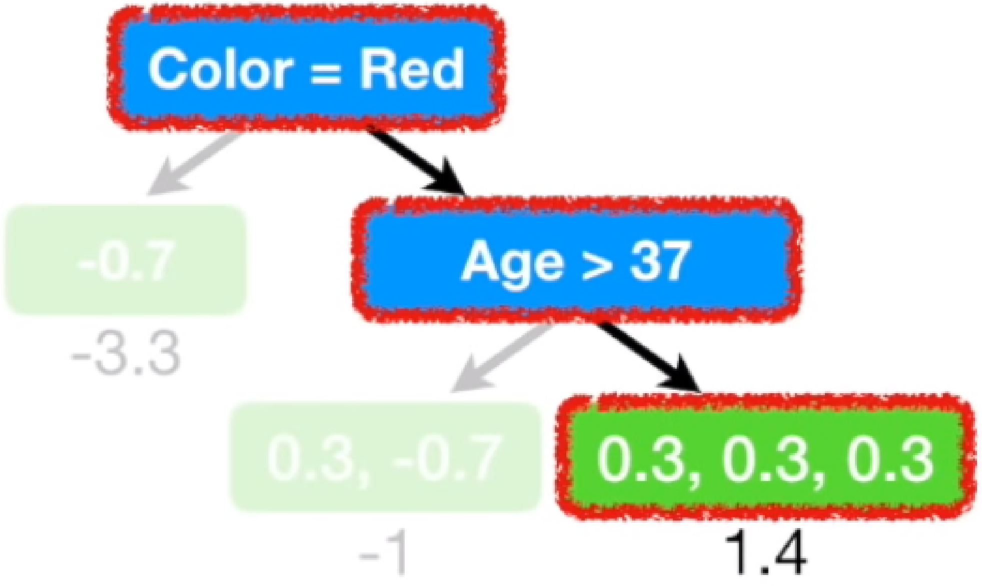
$$\frac{0.3 + 0.3 + 0.3}{(0.7 \times (1 - 0.7)) + (0.7 \times (1 - 0.7)) + (0.7 \times (1 - 0.7))}$$

…and do the math…



Hooray!!! We've calculated **Output Values** for all three leaves in the tree!

log(4/2) = 0.7 + 0.8 X



Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes

…and the new **log(odds)**
Prediction = 1.8.

$$\text{log(odds) Prediction} = 0.7 + (0.8 \times 1.4) = \boxed{1.8}$$

$\log(4/2) = 0.7$

Initial Probability of **Loving Troll 2** $= 0.7$

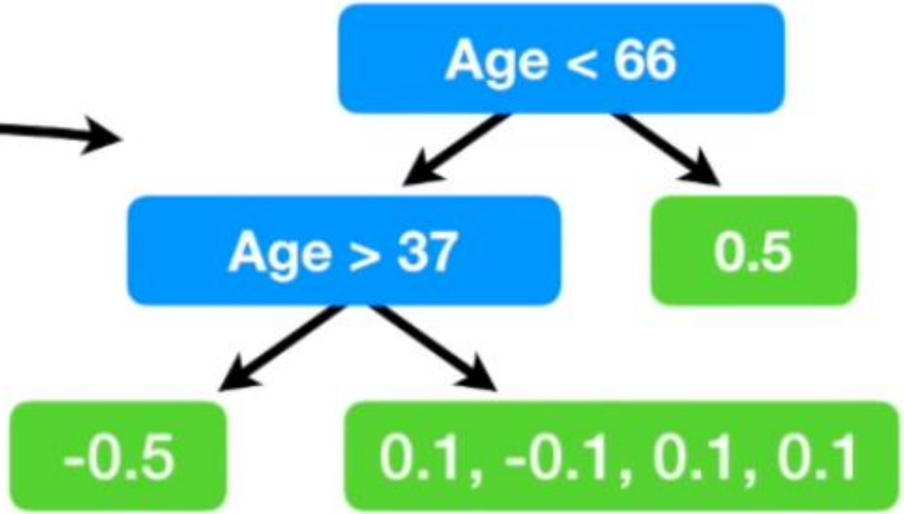
Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	12	Blue	Yes
Yes	87	Green	Yes
No	44	Blue	No
Yes	19	Red	No
No	32	Green	Yes
No	14	Blue	Yes

…so we are taking a small step in the right direction since this person **Loves Troll 2**.

$$\text{Probability} = \frac{e^{1.8}}{1 + e^{1.8}} = \boxed{0.9}$$

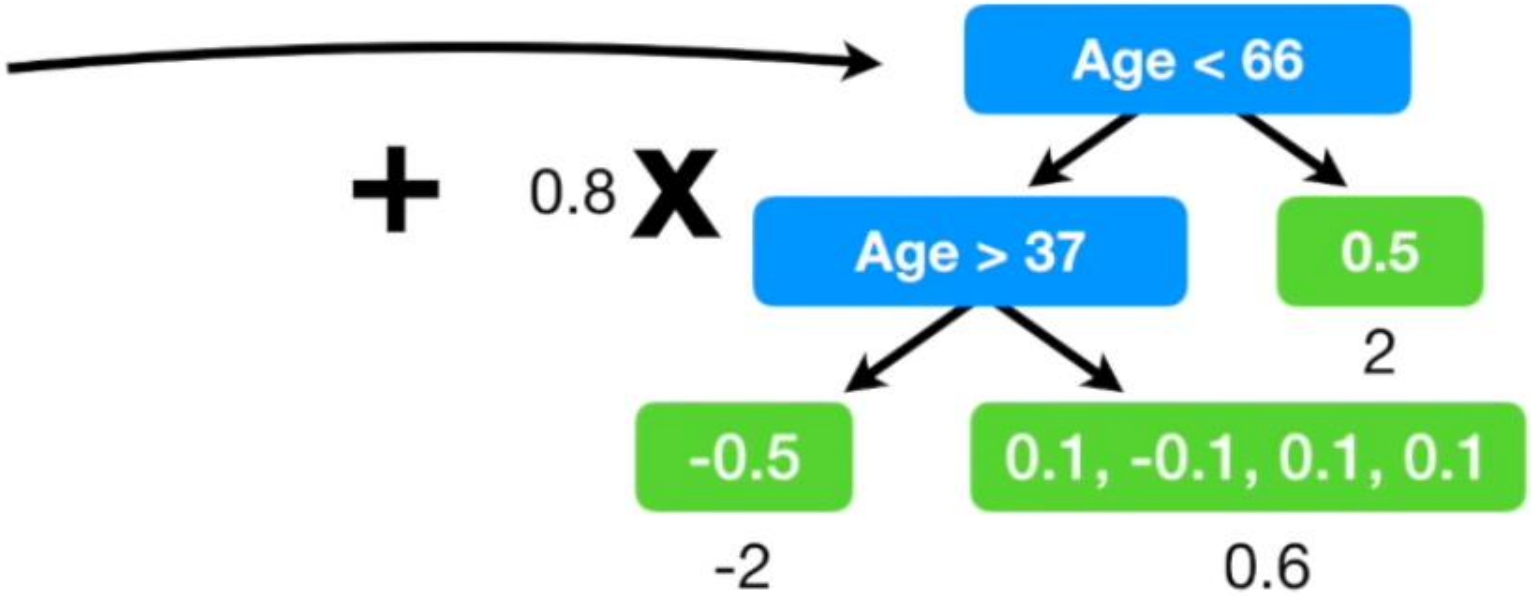
$$\text{log(odds) Prediction} = 0.7 + (0.8 \times 1.4) = 1.8$$

Now that we have the **Residuals**, we can build a new tree…



Likes Popcorn	Age	Favorite Color	Loves Troll 2	Predicted Prob.	Residual
Yes	12	Blue	Yes	0.9	0.1
Yes	87	Green	Yes	0.5	0.5
No	44	Blue	No	0.5	-0.5
Yes	19	Red	No	0.1	-0.1
No	32	Green	Yes	0.9	0.1
No	14	Blue	Yes	0.9	0.1

Then we built another tree based on the new **Residuals**, the difference between the **Observed** values and the values **Predicted** by the leaf ***and*** the first tree…



...and the **Predicted Probability** that this individual will **Love Troll 2** is **0.9**.

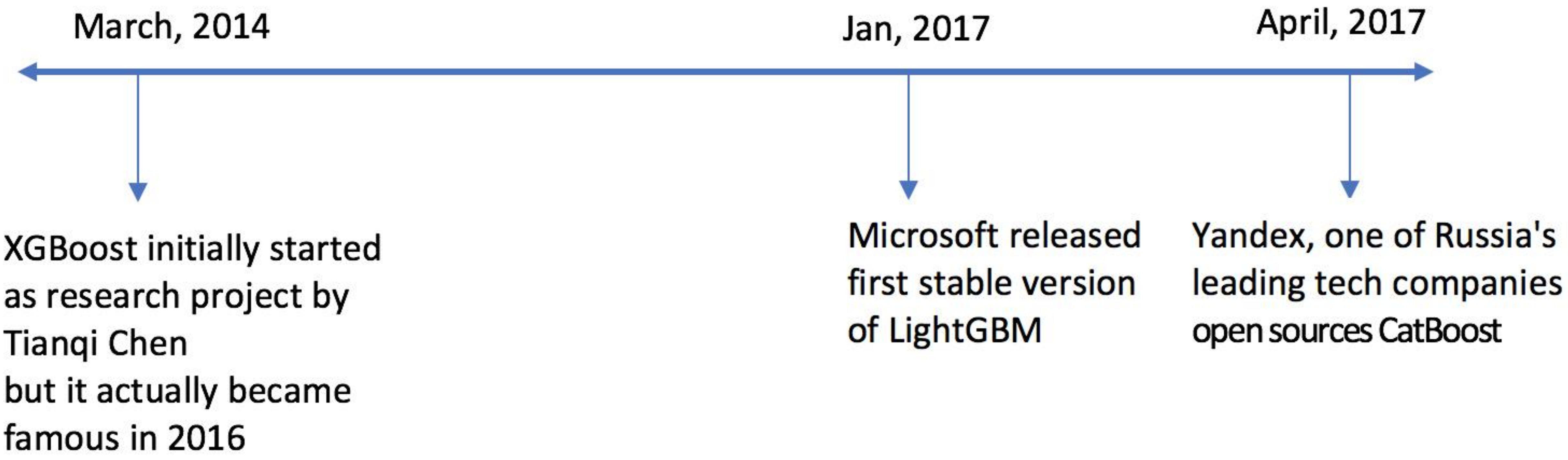
Log(odds) Prediction that someone **Loves Troll 2**: $= 0.7 + (0.8 \times 1.4) + (0.8 \times 0.6) = 2.3$

$$\text{Probability} = \frac{e^{2.3}}{1 + e^{2.3}} = \boxed{0.9}$$

Likes Popcorn	Age	Favorite Color	Loves Troll 2
Yes	25	Green	???

Gradient Boosting in Sklearn

- class sklearn.ensemble.GradientBoostingClassifier(*, loss='deviance', learning_rate=0.1, n_estimators=100, subsample=1.0, criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_depth=3, min_impurity_decrease=0.0, min_impurity_split=None, init=None, random_state=None, max_features=None, verbose=0, max_leaf_nodes=None, warm_start=False, validation_fraction=0.1, n_iter_no_change=None, tol=0.0001, ccp_alpha=0.0)
- class sklearn.ensemble.GradientBoostingRegressor(*, loss='ls', learning_rate=0.1, n_estimators=100, subsample=1.0, criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_depth=3, min_impurity_decrease=0.0, min_impurity_split=None, init=None, random_state=None, max_features=None, alpha=0.9, verbose=0, max_leaf_nodes=None, warm_start=False, validation_fraction=0.1, n_iter_no_change=None, tol=0.0001, ccp_alpha=0.0)


March, 2014
Jan, 2017
April, 2017
XGBoost initially started as research project by Tianqi Chen but it actually became famous in 2016
Microsoft released first stable version of LightGBM
Yandex, one of Russia's leading tech companies open sources CatBoost

	XGBoost	Light BGM		CatBoost	
Parameters Used	max_depth: 50 learning_rate: 0.16 min_child_weight: 1 n_estimators: 200	max_depth: 50 learning_rate: 0.1 num_leaves: 900 n_estimators: 300		depth: 10 learning_rate: 0.15 l2_leaf_reg= 9 iterations: 500 one_hot_max_size = 50	
Training AUC Score	0.999	Without passing indices of categorical features	Passing indices of categorical features	Without passing indices of categorical features	Passing indices of categorical features
		0.992	0.999	0.842	0.887
Test AUC Score	0.789	0.785	0.772	0.752	0.816
Training Time	970 secs	153 secs	326 secs	180 secs	390 secs
Prediction Time	184 secs	40 secs	156 secs	2 secs	14 secs
Parameter Tuning Time (for 81 fits, 200 iteration)	500 minutes	200 minutes		120 minutes	


நன்றி


Boost
gsk