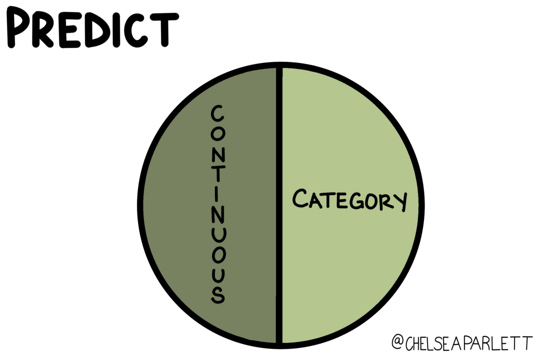


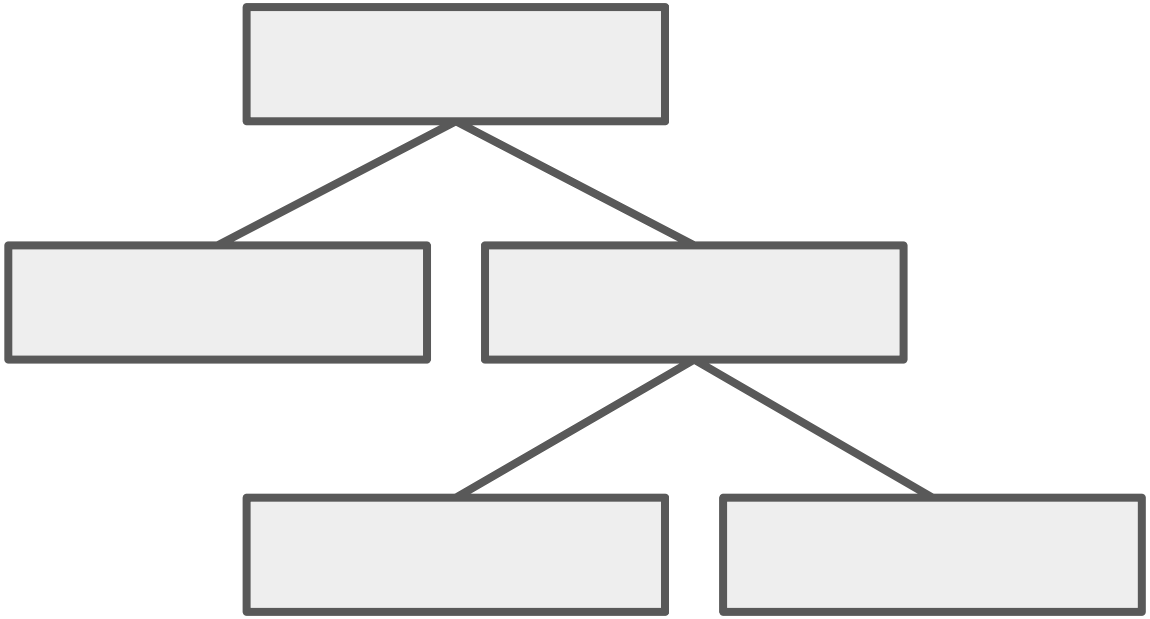
🌲 Based Models

Dr. Chelsea Parlett-Pelleriti

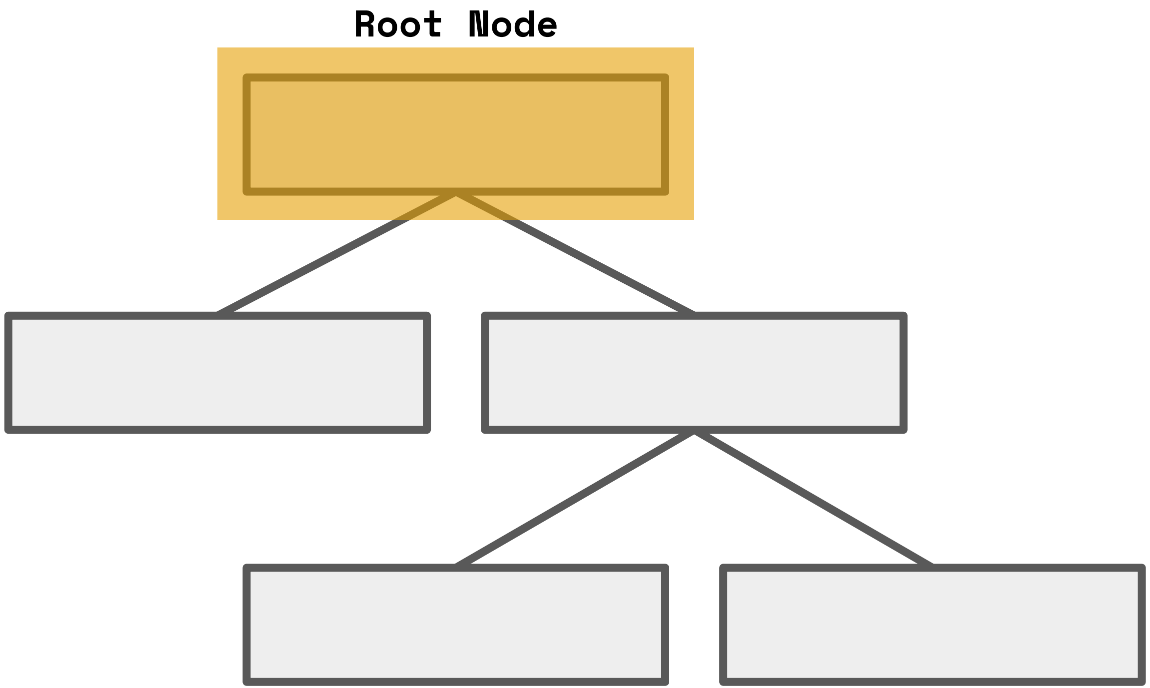
Decision Trees


I BE-LEAF
IN YOU
@CHELSEAPARLETT

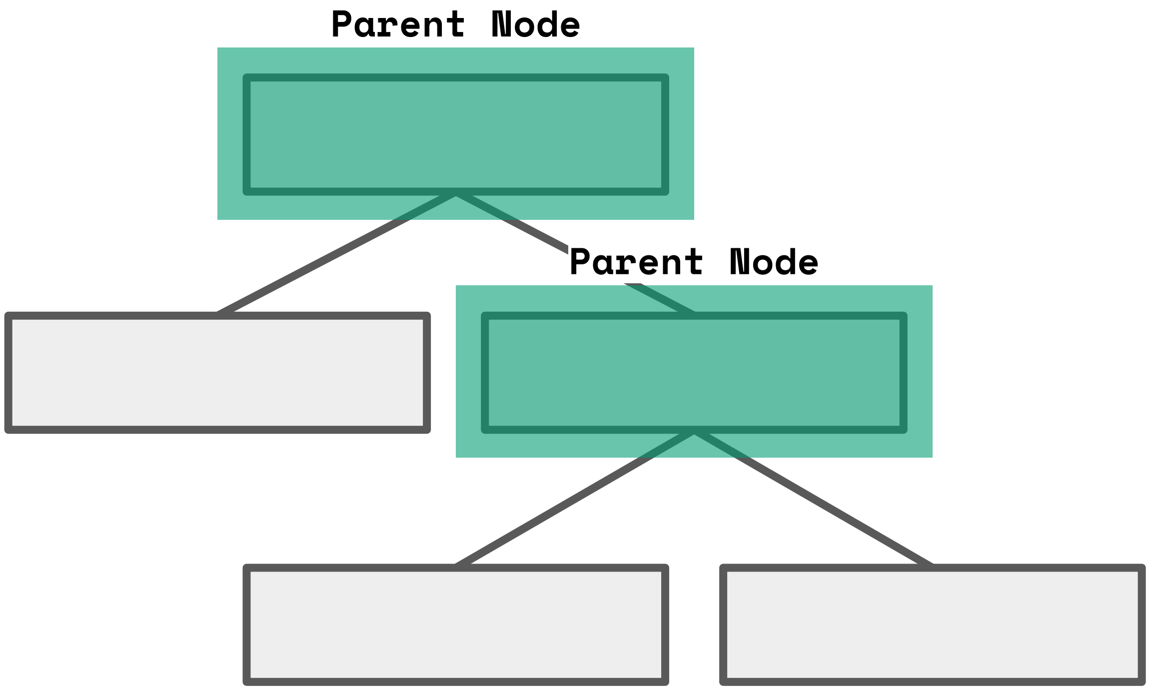
Tree Vocabulary



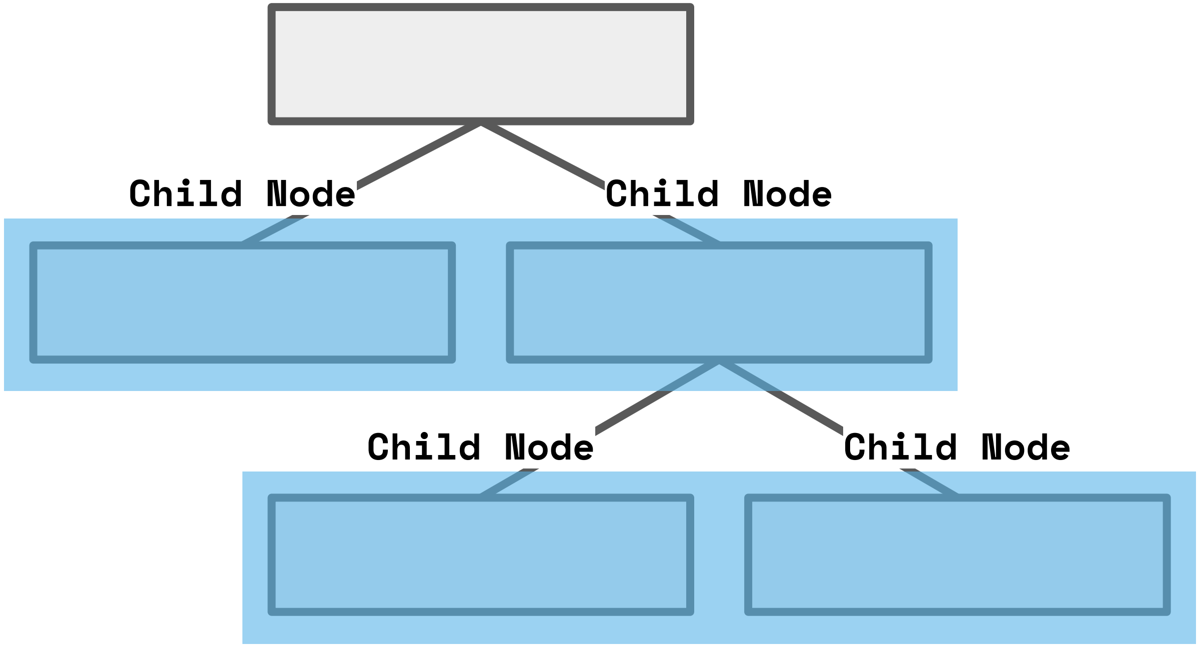
Tree Vocabulary



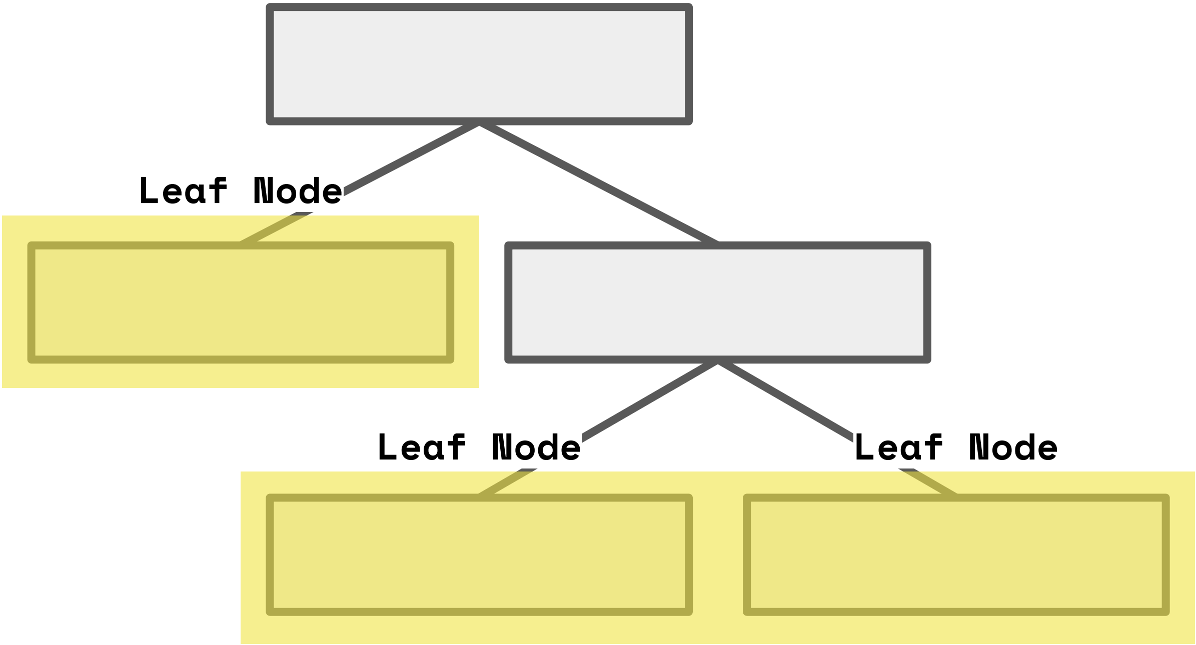
Tree Vocabulary



Tree Vocabulary

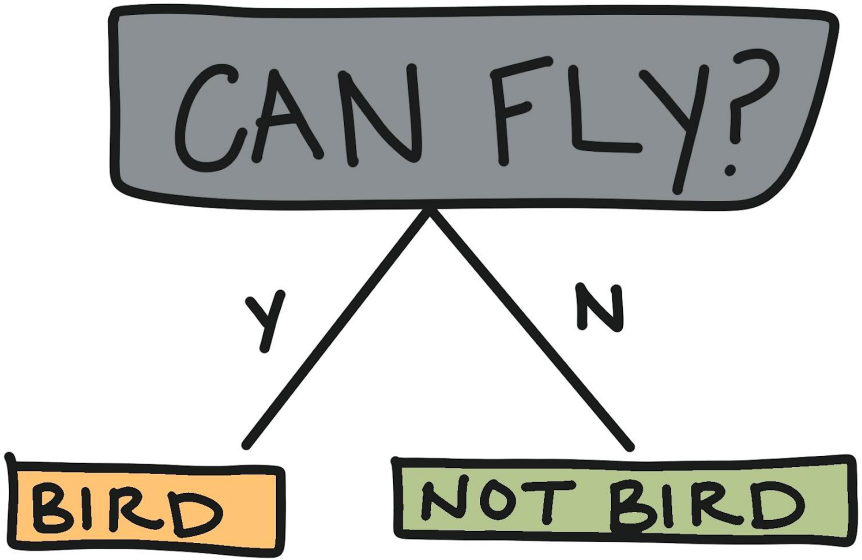


Tree Vocabulary

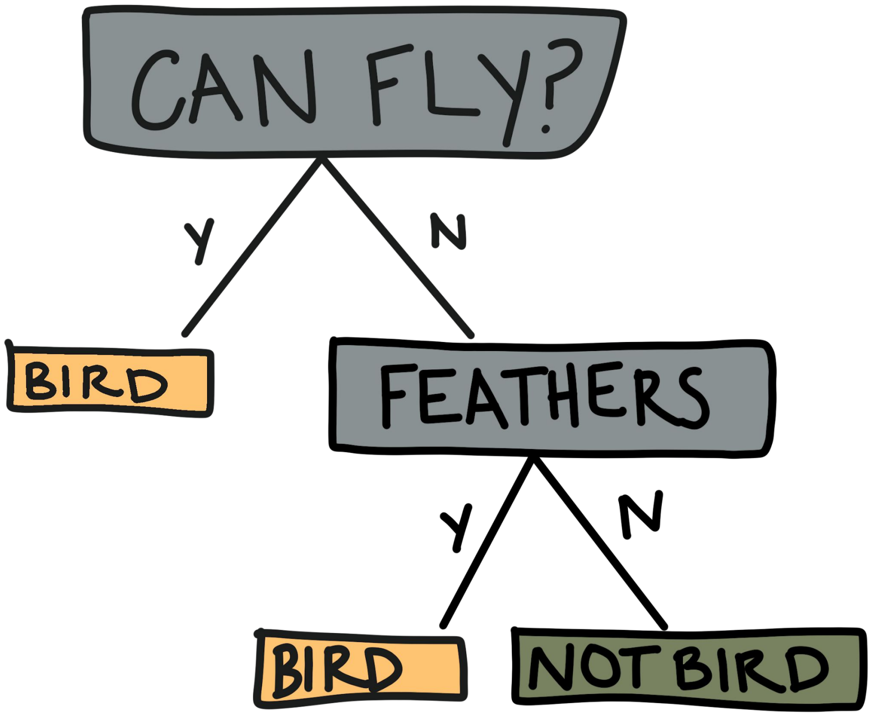


Twenty Questions

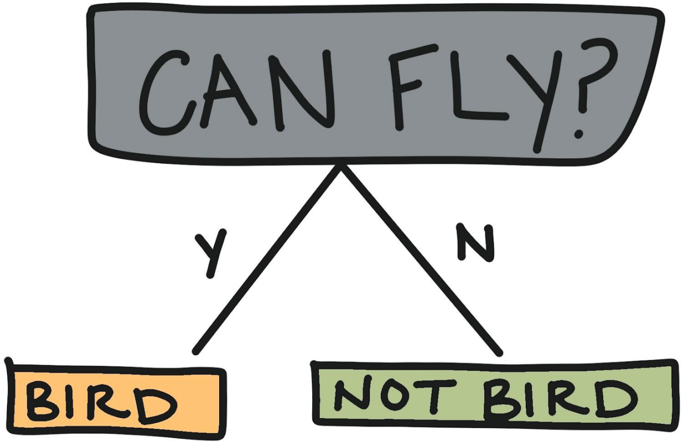
Simple Tree

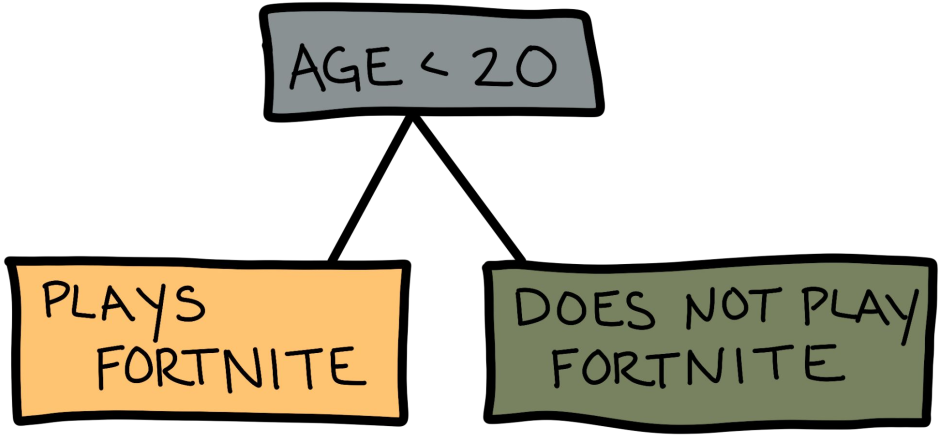


More Complicated Tree



Data Types


CAN FLY?
Y
N
BIRD
NOT BIRD

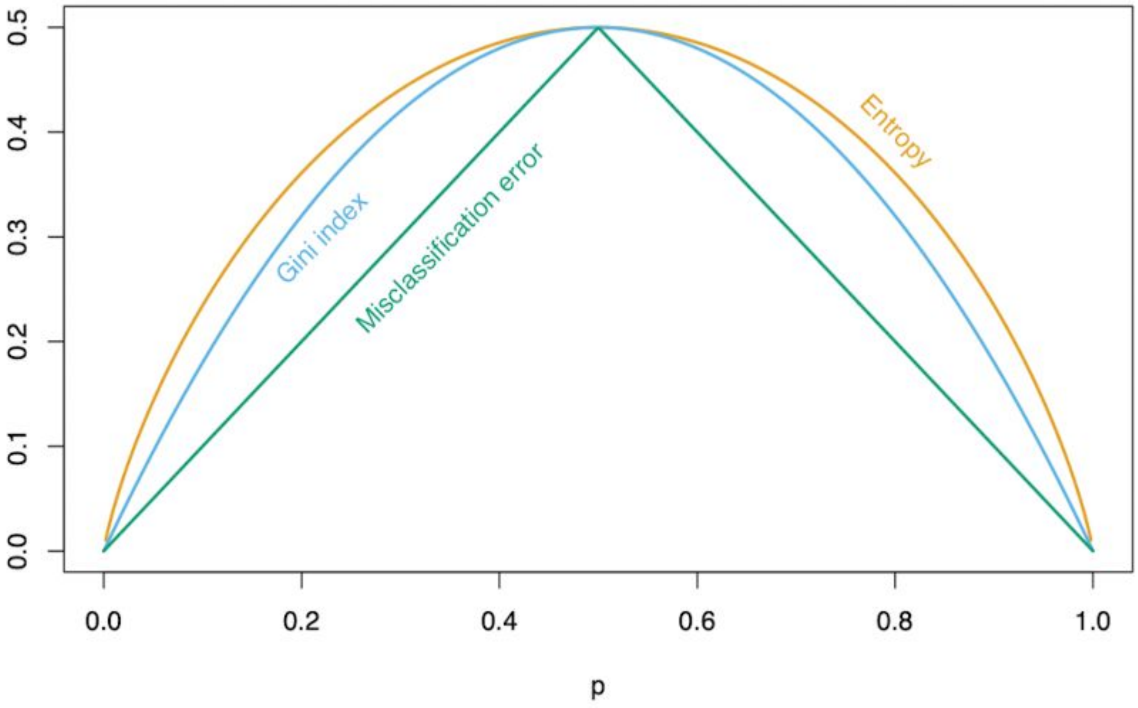

AGE < 20
PLAYS FORTNITE
DOES NOT PLAY FORTNITE

Gini Impurity and Entropy

$$\mathrm{GI} = 1 - \sum_{i=1}^{n} (p_i)^2 \qquad \mathrm{E} = -\sum_{i=1}^{n} p_i * log(p_i)$$

Goal: choose split so that GI (or entropy) is minimized

Gini Impurity and Entropy



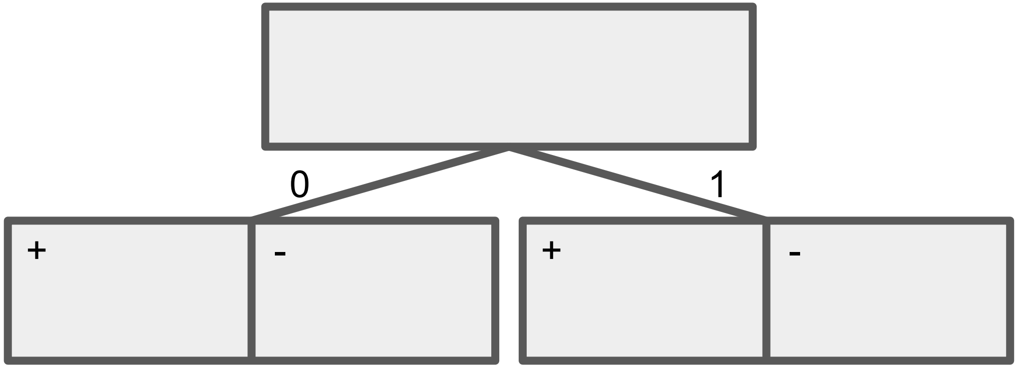
$$\mathrm{GI} = 1 - \sum_{i=1}^{n} (p_i)^2 \qquad \mathrm{E} = - \sum_{i=1}^{n} p_i * log(p_i)$$

Figure from the Elements of Statistical Learning (Hastie et al)

$$1 - \sum_{i=1}^{n} (p_i)^2$$

Categorical

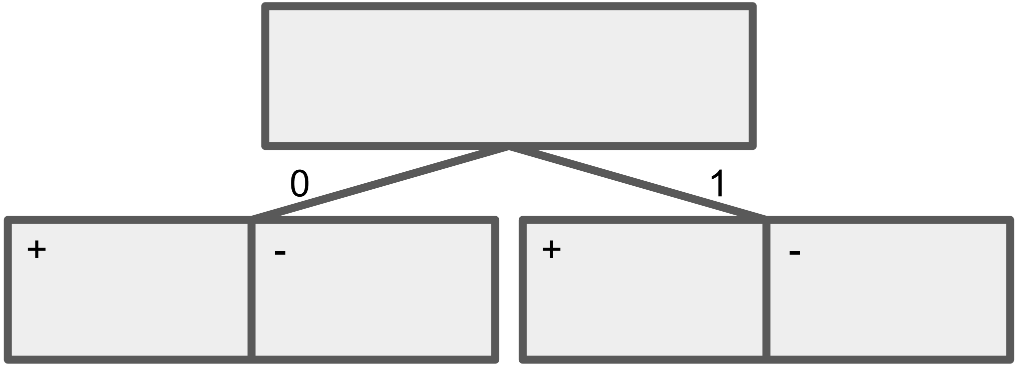
cats	pet	wfh	children	income
1	0	1	1	34
0	1	0	1	58.3
1	1	1	0	71.5
0	0	0	1	74.9
0	0	0	1	75.3
1	0	0	1	75.6
0	0	0	1	81
1	1	1	0	82.3
1	1	1	0	85.6
1	1	1	1	95.4



Categorical

$$1 - \sum_{i=1}^{n} (p_i)^2$$

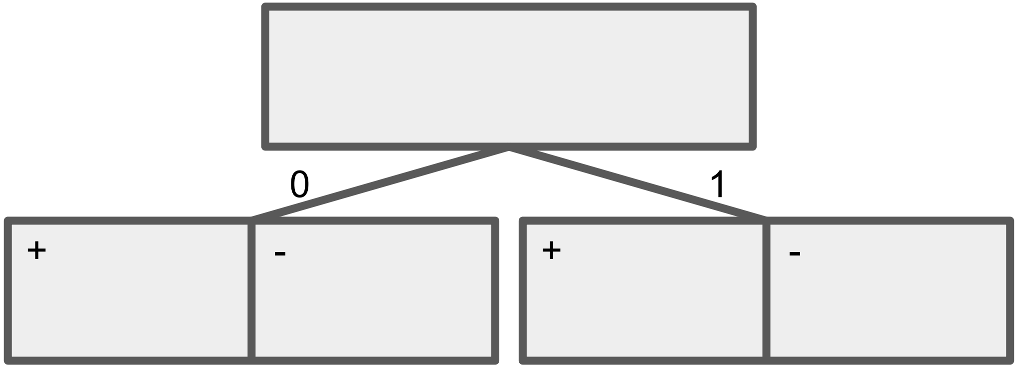
cats	pet	wfh	children	income
1	0	1	1	34
0	1	0	1	58.3
1	1	1	0	71.5
0	0	0	1	74.9
0	0	0	1	75.3
1	0	0	1	75.6
0	0	0	1	81
1	1	1	0	82.3
1	1	1	0	85.6
1	1	1	1	95.4



Continuous

$$1 - \sum_{i=1}^{n} (p_i)^2$$

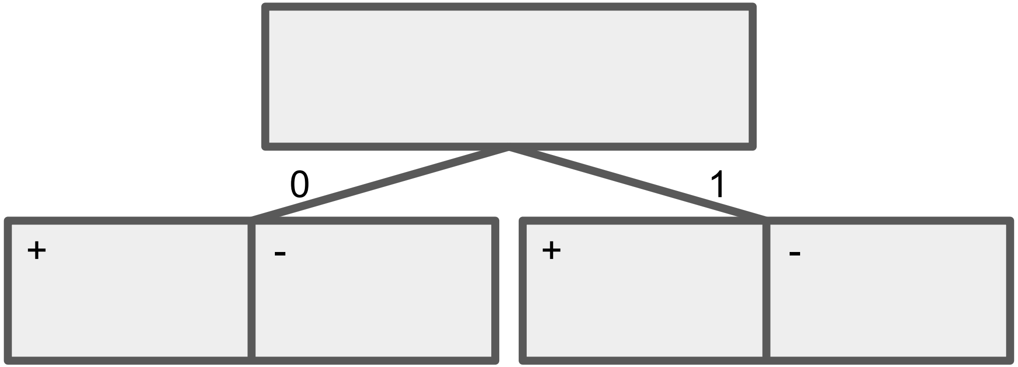
cats	pet	wfh	children	income
1	0	1	1	34
0	1	0	1	58.3
1	1	1	0	71.5
0	0	0	1	74.9
0	0	0	1	75.3
1	0	0	1	75.6
0	0	0	1	81
1	1	1	0	82.3
1	1	1	0	85.6
1	1	1	1	95.4



Continuous

$$1 - \sum_{i=1}^{n} (p_i)^2$$

cats	pet	wfh	children	income
1	0	1	1	34
0	1	0	1	58.3
1	1	1	0	71.5
0	0	0	1	74.9
0	0	0	1	75.3
1	0	0	1	75.6
0	0	0	1	81
1	1	1	0	82.3
1	1	1	0	85.6
1	1	1	1	95.4



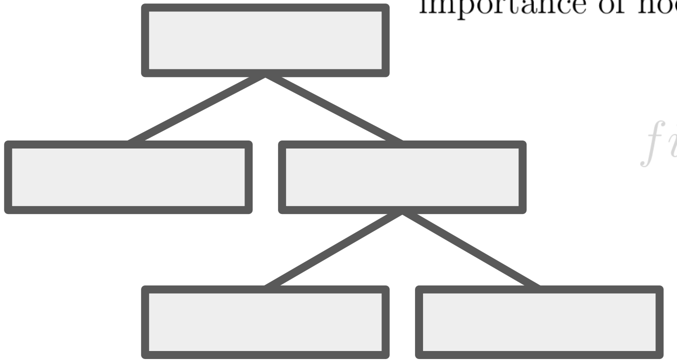
Basic Steps

1. Calculate Gini Impurity (or Entropy/Information Gain) for each node
2. Choose Node with lowest score
3. If the parent node has the lowest score, it is a leaf.

Variable Importance

1. How much does this feature reduce node impurity?
2. If we shuffle the values of this feature, how much does it reduce the performance?

$$\underbrace{\text{imp}_j}_{\text{importance of node } j} = \overbrace{w_j C_j}^{\text{weighted parent node impurity}} - \underbrace{(w_{\text{left}_j} C_{\text{left}_j} + w_{\text{right}_j} C_{\text{right}_j})}_{}$$

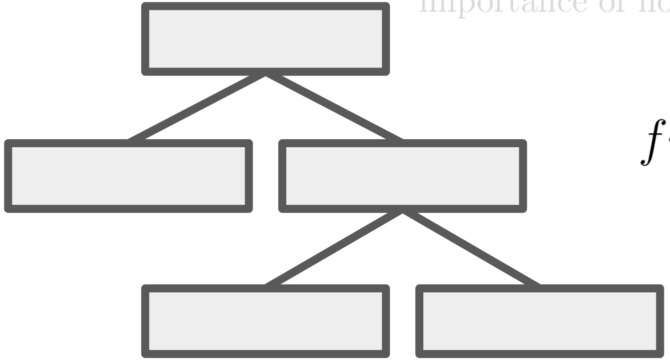


$$fi_i = \frac{\sum_{j \in S_i} \text{imp}_j}{\sum_{k \in S_{all}} \text{imp}_k}; S_i \text{ is set of all nodes that split on feature}_i$$

Variable Importance

1. How much does this feature reduce node impurity?
2. If we shuffle the values of this feature, how much does it reduce the performance?

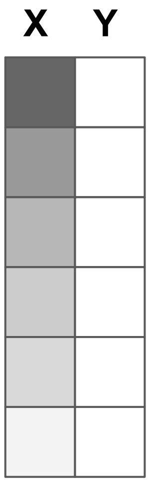
$$\underbrace{\mathrm{imp}_j}_{\text{importance of node } j} = \overbrace{w_j C_j}^{\text{weighted parent node impurity}} - \underbrace{(w_{\text{left}_j} C_{\text{left}_j} + w_{\text{right}_j} C_{\text{right}_j})}$$



$$fi_i = \frac{\sum_{j \in S_i} \mathrm{imp}_j}{\sum_{k \in S_{all}} \mathrm{imp}_k}; S_i \text{ is set of all nodes that split on feature}_i$$

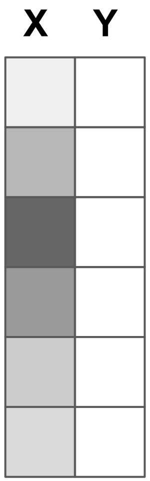
Variable Importance

1. How much does this feature reduce node impurity?
2. If we shuffle the values of this feature, how much does it reduce the performance?



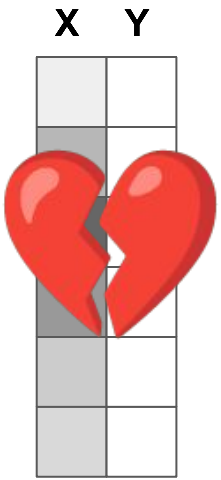
Variable Importance

1. How much does this feature reduce node impurity?
2. If we shuffle the values of this feature, how much does it reduce the performance?



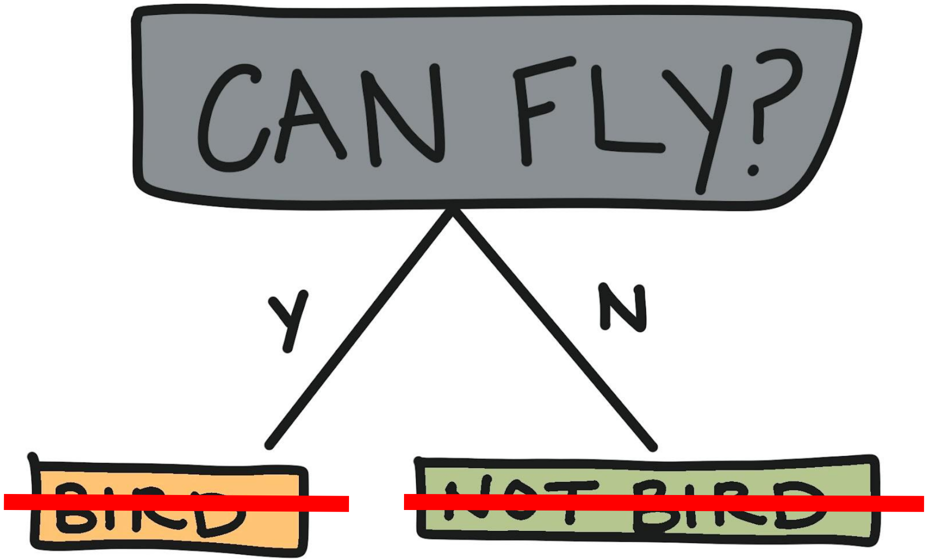
Variable Importance

1. How much does this feature reduce node impurity?
2. If we shuffle the values of this feature, how much does it reduce the performance?



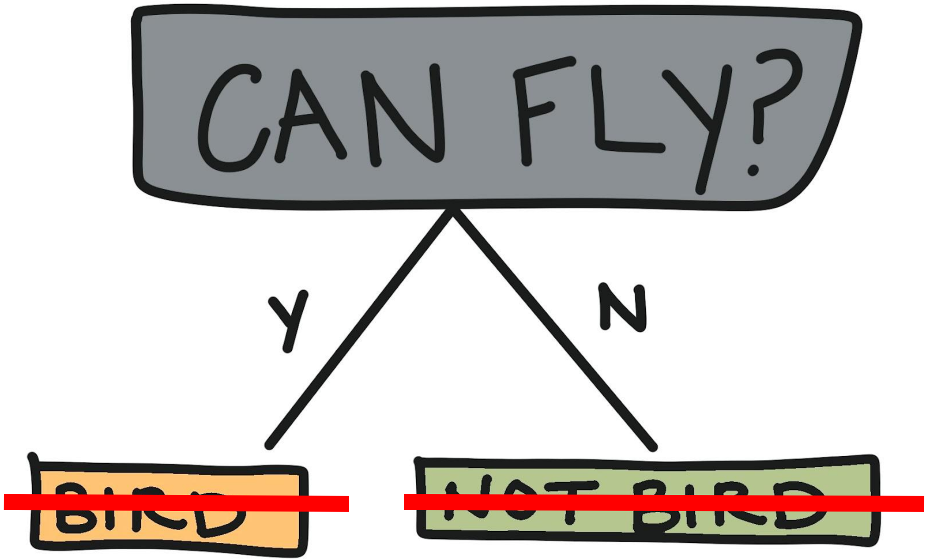
Regression Trees

Regression Trees



Regression Trees

Instead of checking if splits decrease Gini Impurity, we check if they decrease MSE



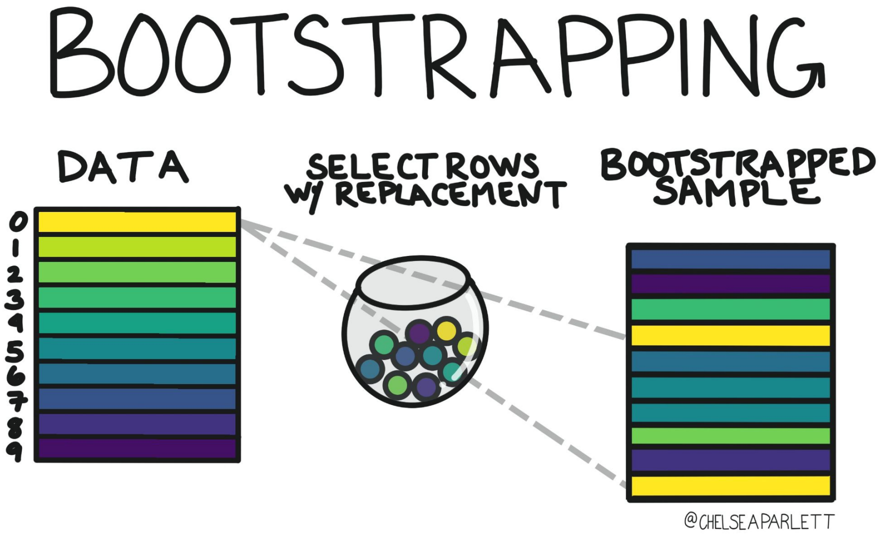
Random Forests

ONE vs. MANY

Random Forests

- Bootstrap Aggregating (**Bagging**)
- **Random Feature Selection**

Random Forests



Random Forests

- Bootstrap Aggregating (**Bagging**)
- **Random Feature Selection**

Random Forests

Random Forests

Random Forests

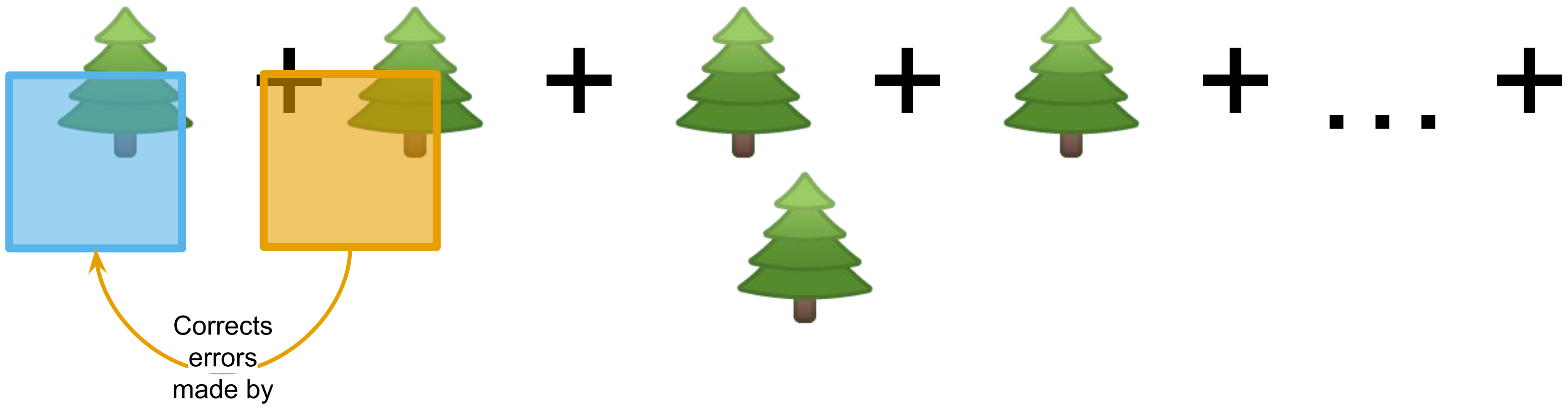
Important Hyperparameters

- # of trees
- # of features per tree

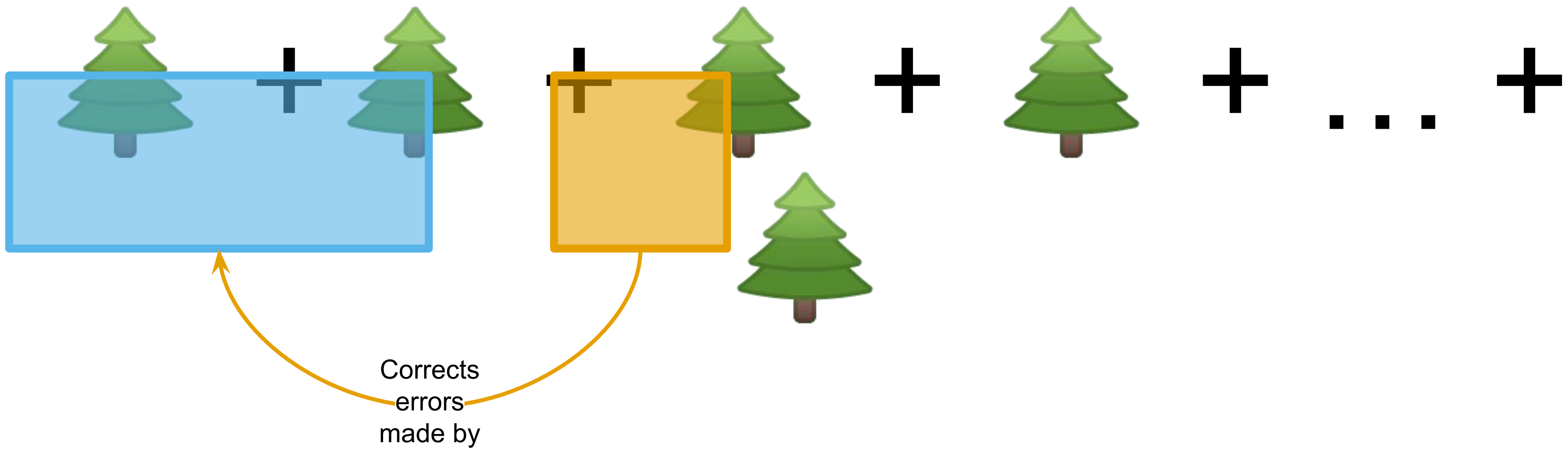
Gradient Boosting Trees

Gradient Boosting Trees

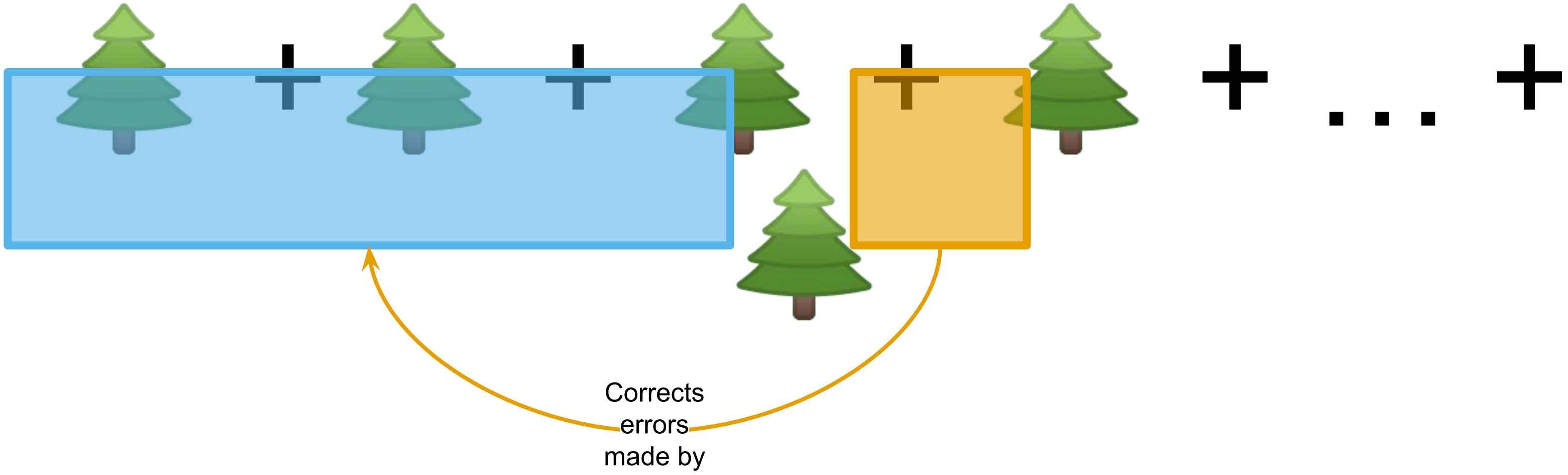
Gradient Boosting Trees



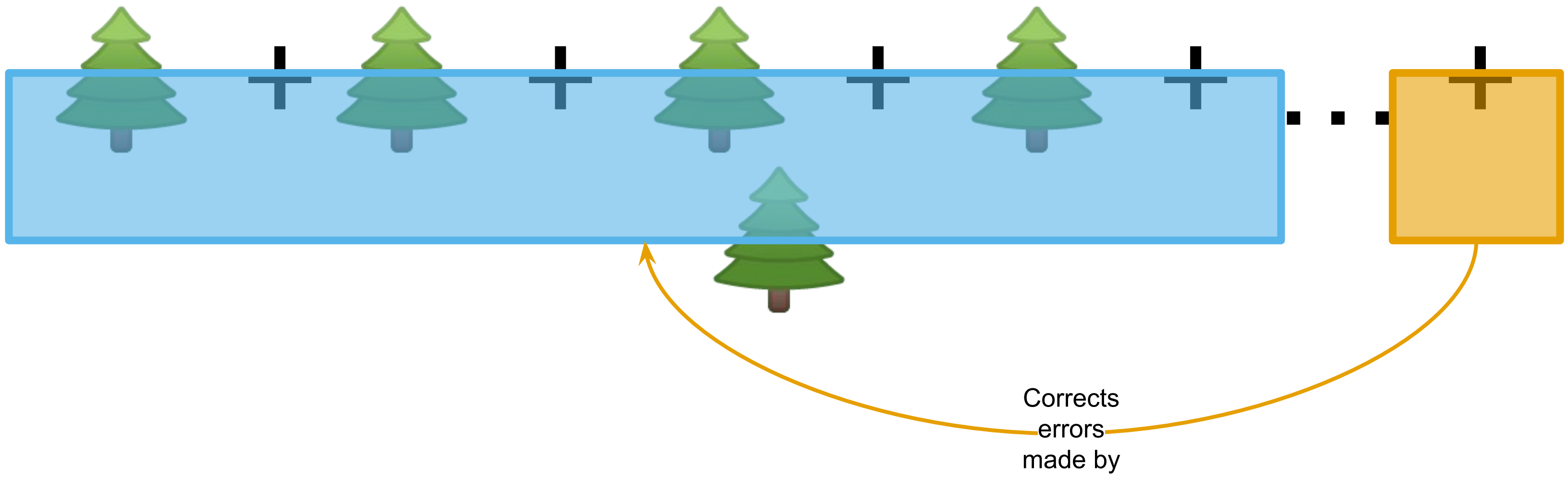
Gradient Boosting Trees



Gradient Boosting Trees



Gradient Boosting Trees



Gradient Boosting Tree

	Age	Initial Guess	Residual
Person 1	20		
Person 2	19		
Person 3	21		
Person 4	20		

Gradient Boosting Tree

	Age	Initial Guess	Residual
Person 1	20	20	
Person 2	19	20	
Person 3	21	20	
Person 4	20	20	

Gradient Boosting Tree

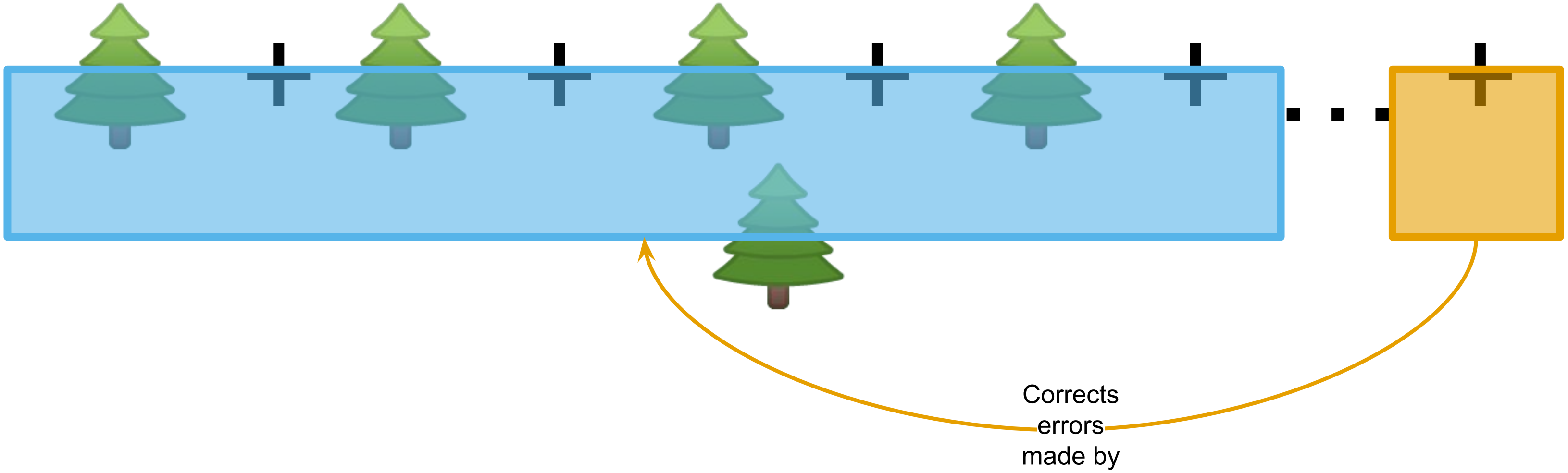
	Age	Initial Guess	Residual
Person 1	20	20	0
Person 2	19	20	-1
Person 3	21	20	1
Person 4	20	20	0

Gradient Boosting Tree

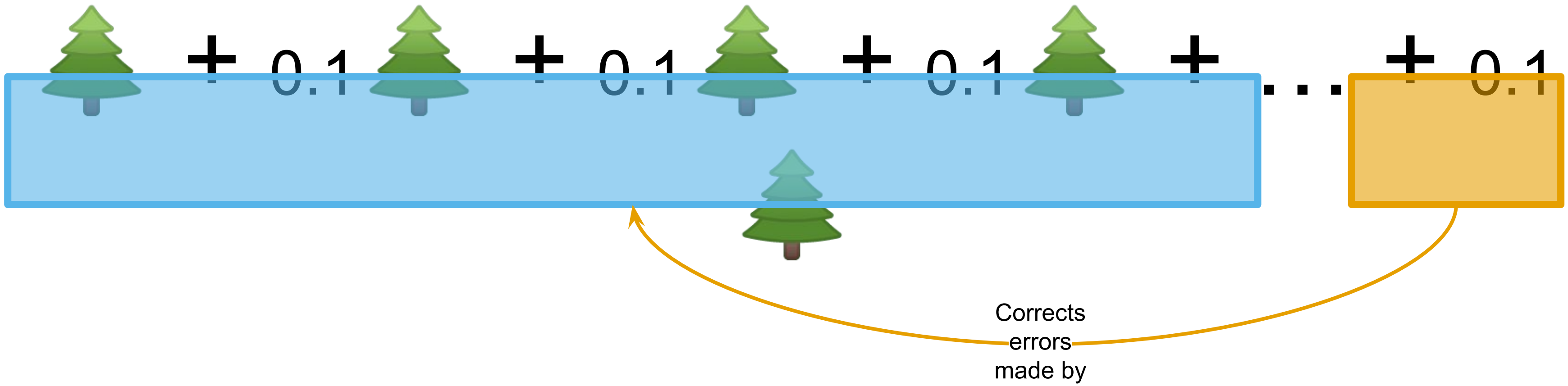
Actual Value = Prediction + Residual

	Age	Initial Guess	Residual
Person 1	20	20	0
Person 2	19	20	-1
Person 3	21	20	1
Person 4	20	20	0

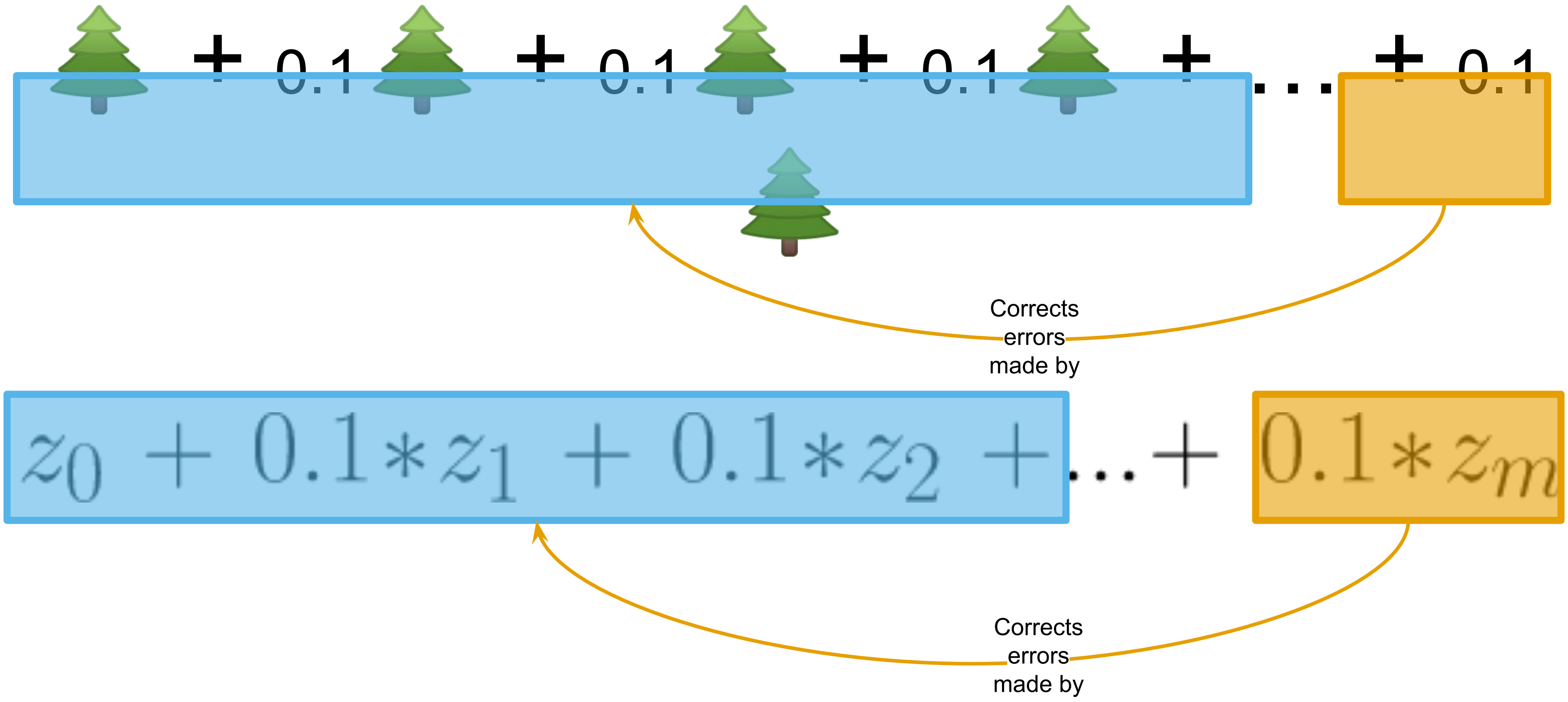
Gradient Boosting Trees



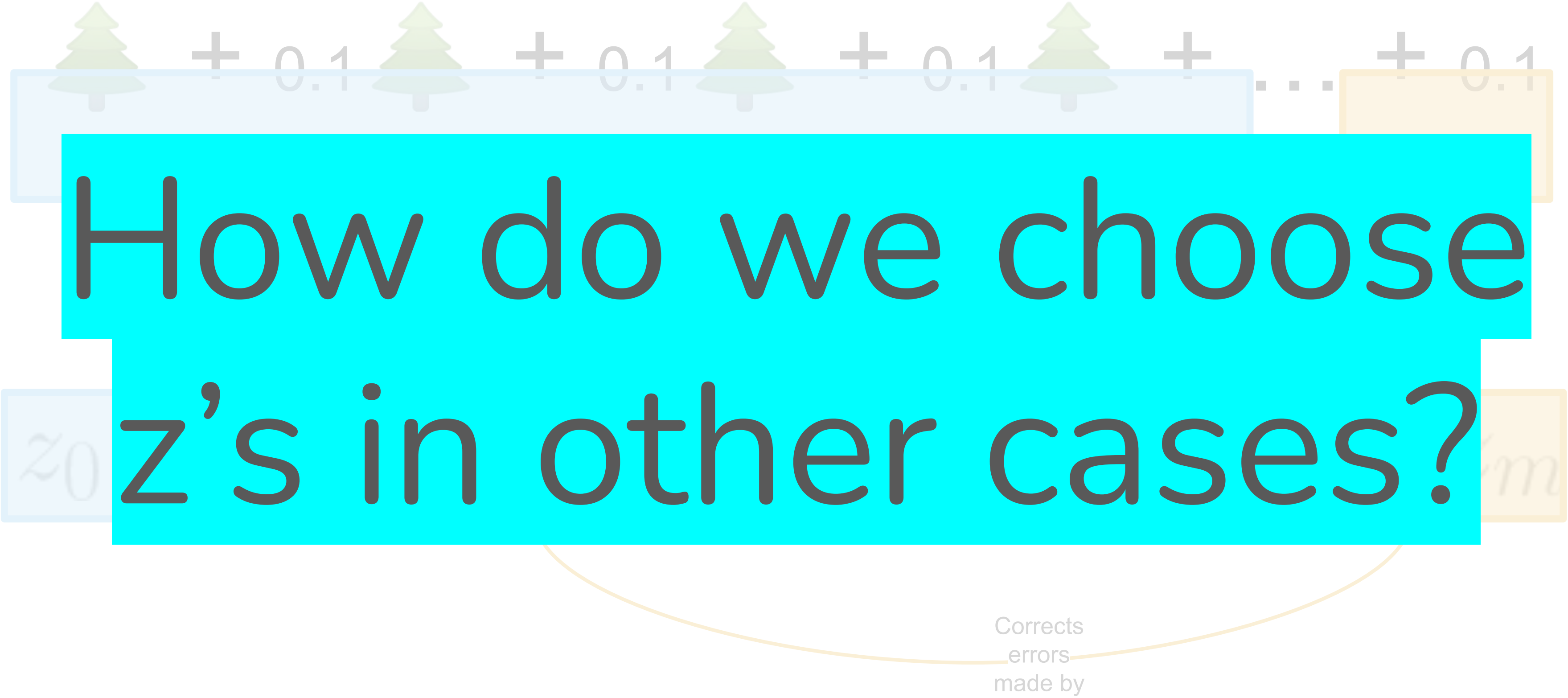
Gradient Boosting Trees



Gradient Boosting Trees



Gradient Boosting Trees



How do we choose z’s in other cases?

Gradient Boosting Trees: Some Extra Math

$$z_i = -\frac{\partial L(y, F_i)}{\partial F_i}$$

Let's generalize that math! Having future trees predict the error works for a regression tree using **(y-p)^2** as it's loss.

Gradient Boosting Trees: Some Extra Math

$$z_i = -\frac{\partial L(y, F_i)}{\partial F_i}$$

But in all cases, subsequent trees will predict the **negative gradient** (z_i) of the Loss with respect to the Ensemble prediction

Gradient Boosting Trees: Some Extra Math

$$z_i = -\frac{\partial L(y, F_i)}{\partial F_i}$$

But in all cases, subsequent trees will predict the **negative gradient** (z_i) of the Loss with respect to the Ensemble prediction

Baseline prediction

$$F_i = \sum_{t=0}^{i} z_t \qquad F_2 = \boxed{z_0} + \boxed{z_1} + \boxed{z_2}$$

Errors

Gradient Boosting Trees: Some Extra Math

$$z_i = -\frac{\partial L(y, F_i)}{\partial F_i}$$

But in all cases, subsequent trees will predict the **negative gradient** (z_i) of the Loss with respect to the Ensemble prediction

$$F_i = \sum_{t=0}^{i} z_t$$

$$F_1 = z_0 + z_1$$
$$F_2 = z_0 + z_1 + z_2$$

$$F_i = F_{i-1} + z_i$$

Gradient Boosting Trees: Some Extra Math

$$z_i = \boxed{-\frac{\partial L(y, F_i)}{\partial F_i}}$$

Negative Gradient of Loss w.r.t. Ensemble Prediction

Gradient Boosting Trees: Some Extra Math

$$z_i = \boxed{-\frac{\partial L(y, F_i)}{\partial F_i}}$$

Negative Gradient of Loss w.r.t. Ensemble Prediction

The Negative Gradient tell us what *adjustments* we should make to our prediction (**F_i**) in order to *decrease* our loss.

Gradient Boosting Trees: Some Extra Math

$$z_i = \boxed{-\frac{\partial L(y, F_i)}{\partial F_i}}$$

Negative Gradient of Loss w.r.t. Ensemble Prediction

The Negative Gradient tell us what *adjustments* we should make to our prediction (**F_i**) in order to *decrease* our loss.

$$L(y,p) = \boxed{(y-p)^2}$$

Let's choose Squared Error as our Loss

$$-\frac{\partial L(y,p)}{\partial p} = \boxed{2(y-p)}$$

Negative Gradient

Gradient Boosting Trees: Some Extra Math

$$z_i = \boxed{-\frac{\partial L(y, F_i)}{\partial F_i}}$$

Negative Gradient of Loss w.r.t. Ensemble Prediction

The Negative Gradient tell us what *adjustments* we should make to our prediction (**F_i**) in order to *decrease* our loss.

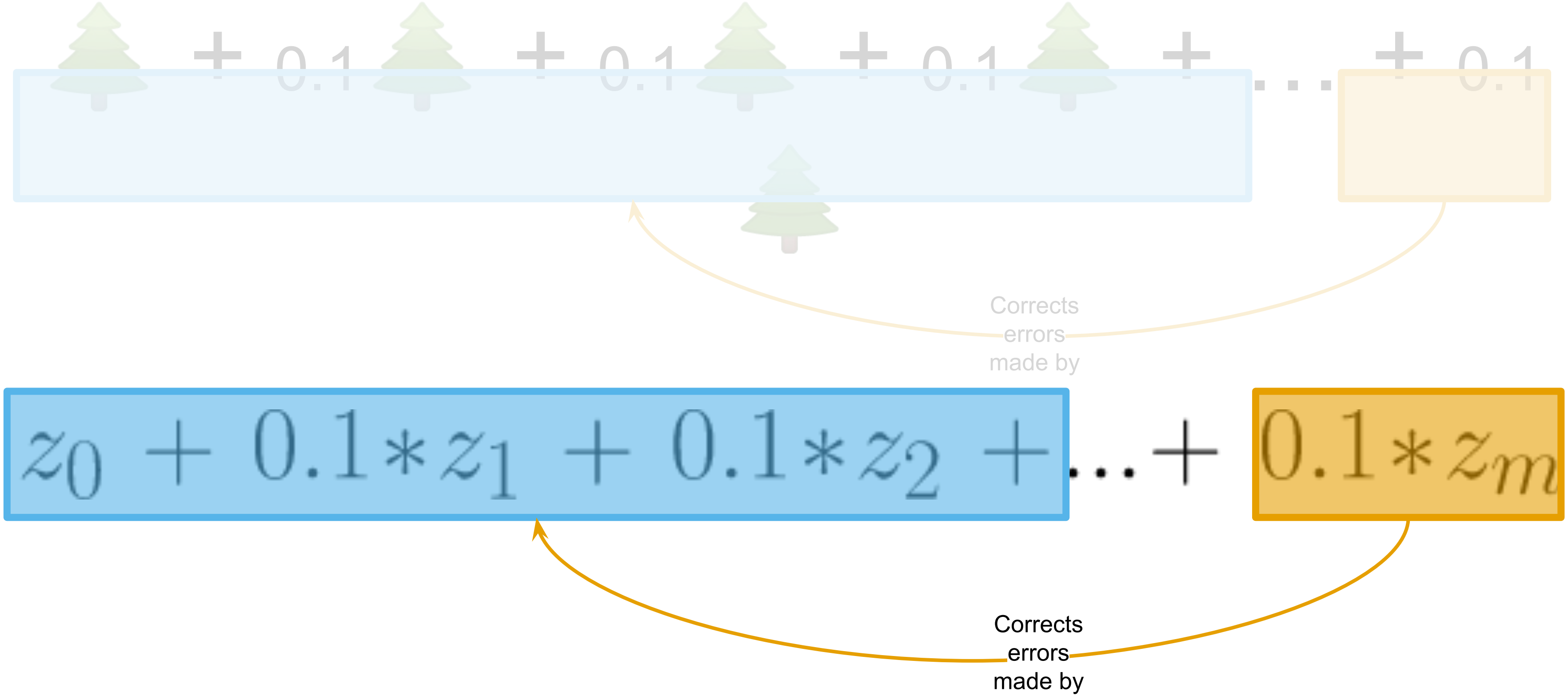
$$L(y,p) = \boxed{(y-p)^2}$$

Let's choose Squared Error as our Loss

$$-\frac{\partial L(y,p)}{\partial p} = 2\boxed{(y-p)}$$

Just the Error!

Gradient Boosting Trees



Gradient Boosting Trees: Some Extra Math

$$z_i = -\frac{\partial L(y, F_i)}{\partial F_i}$$

With squared loss, **error *is* the negative gradient**, but the negative gradient will work in other situations