XGBoost and Competitors: Tree-Based Methods for Tabular Data

28.11.2025

Adéla Kubíková

What is tabular data?

- most common type
- transactional data
- heterogeneous
- no local structure
- quality of datasets
- deep learning?

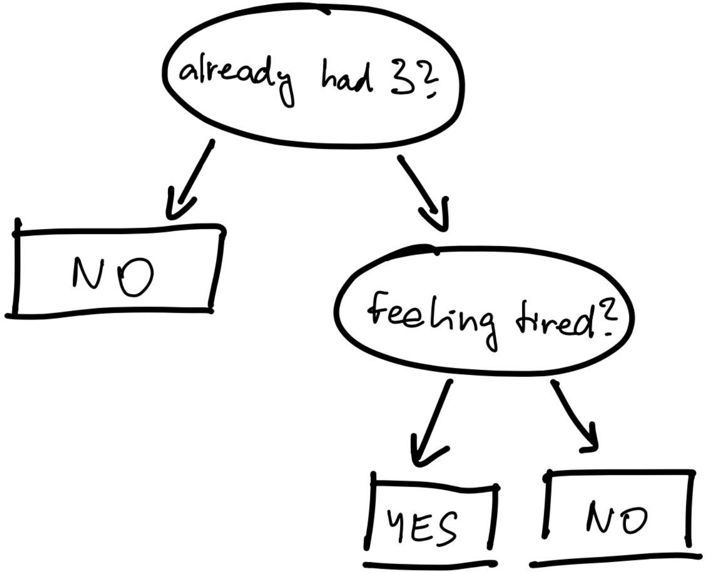
Customer ID	Age	Gender	Income ($)	Purchased
101	25	Male	40000	No
102	32	Female	55000	Yes
103	45	Male	72000	No
104	28	Female	60000	Yes
105	29	Male	50000	Yes

Contents

Evolution of Tree Algorithm

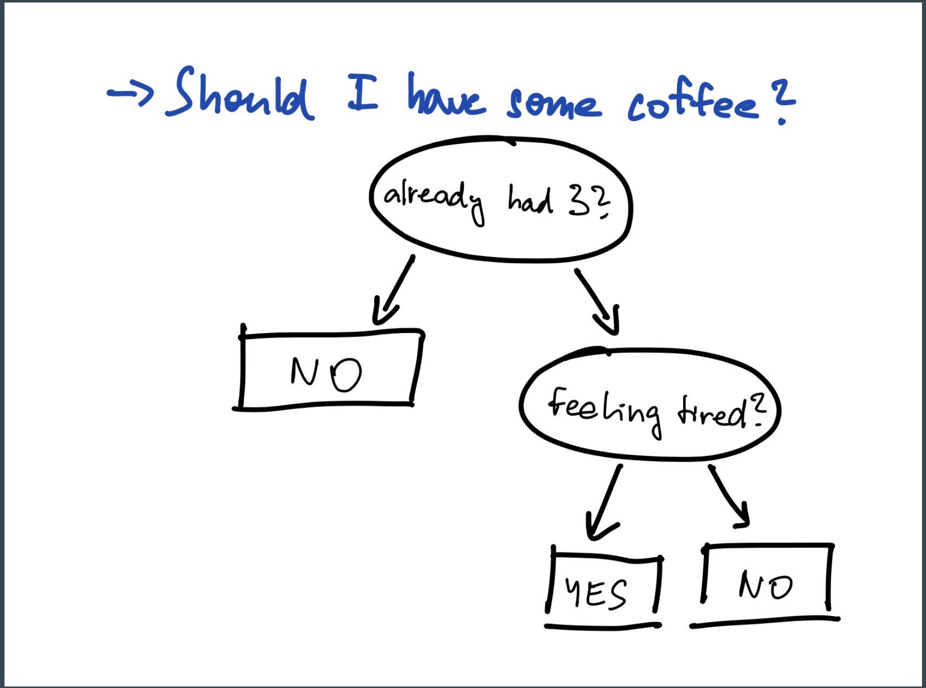
Base learner: Decision trees

→ Should I have some coffee?

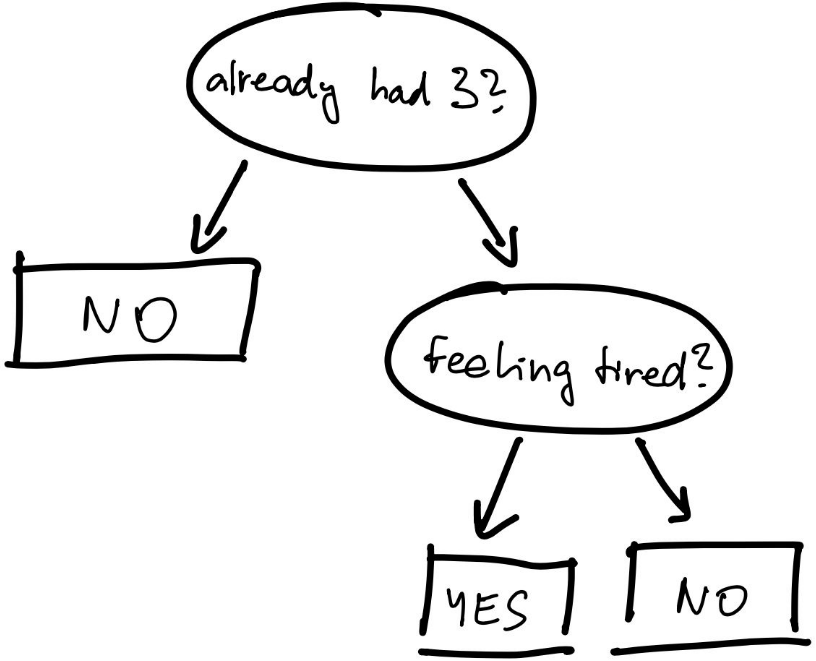


What is a decision tree?

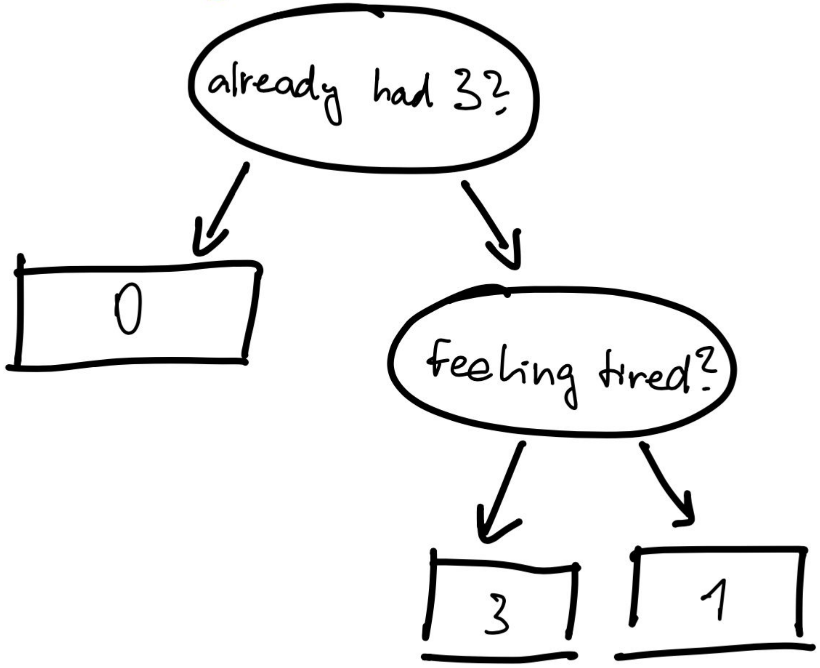
- simple model
- set of rules leading to a decision
- feature <= threshold
- classification trees
- regression trees



→ Should I have some coffee?



→ How many cups should I drink?

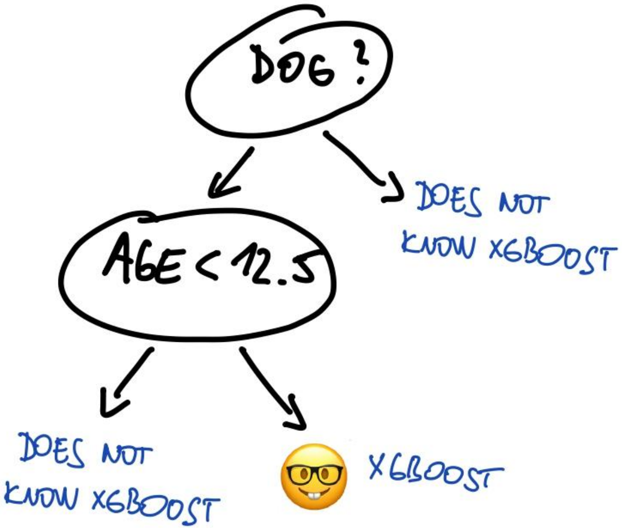


How to build a decision tree?

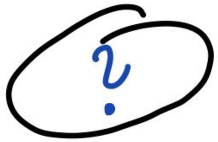
- classification: minimizing Gini Impurity (Entropy, Information Gain)

- regression: minimizing prediction error (MSE, MAE)

Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO



Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO



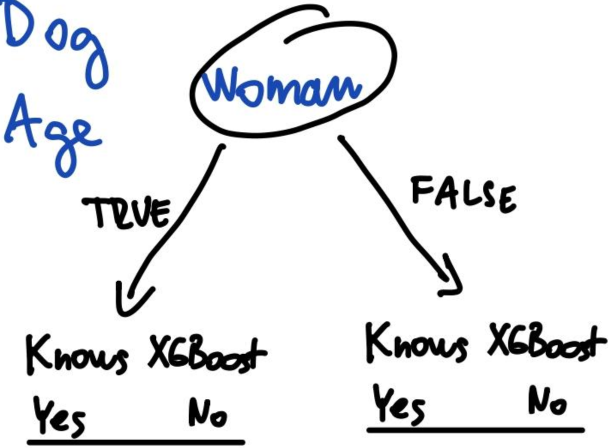
Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO

Woman
Dog
Age

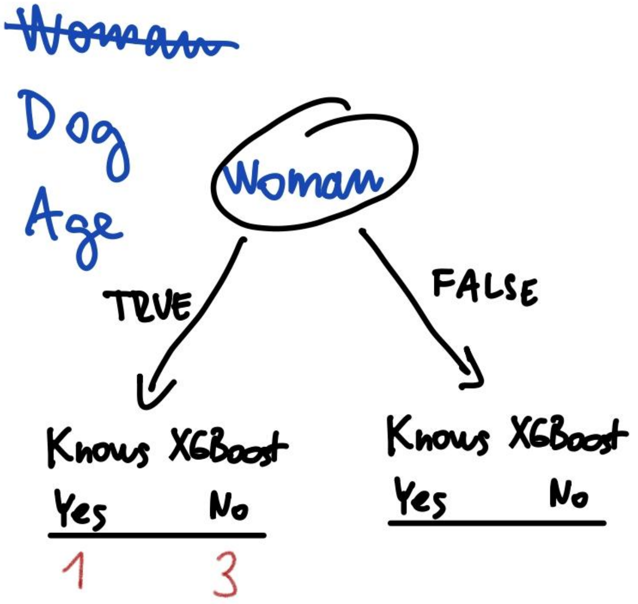
?

Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO

~~Woman~~
Dog
Age

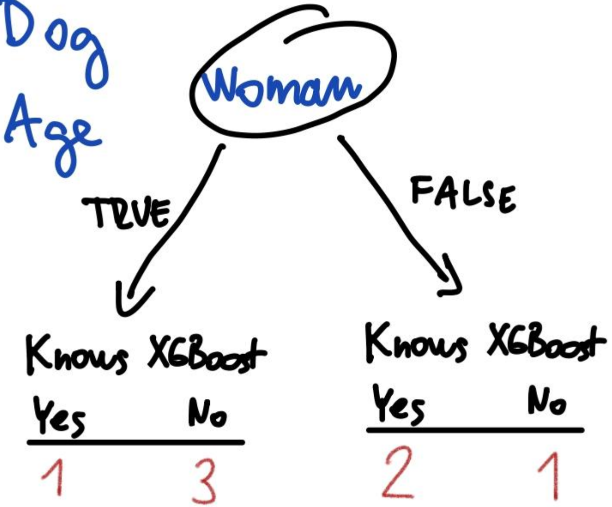


Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO



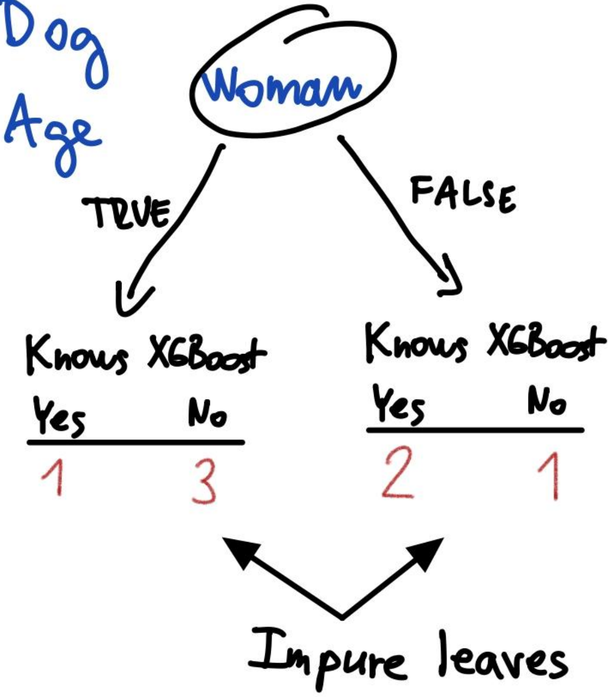
Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO

~~Woman~~
Dog
Age



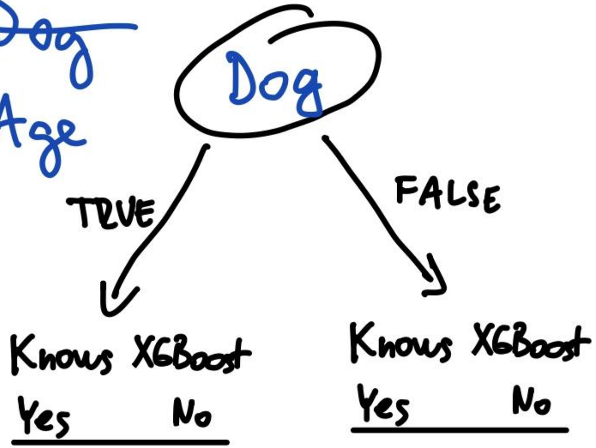
Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO

~~Woman~~
Dog
Age



Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO

~~Woman~~
~~Dog~~
Age

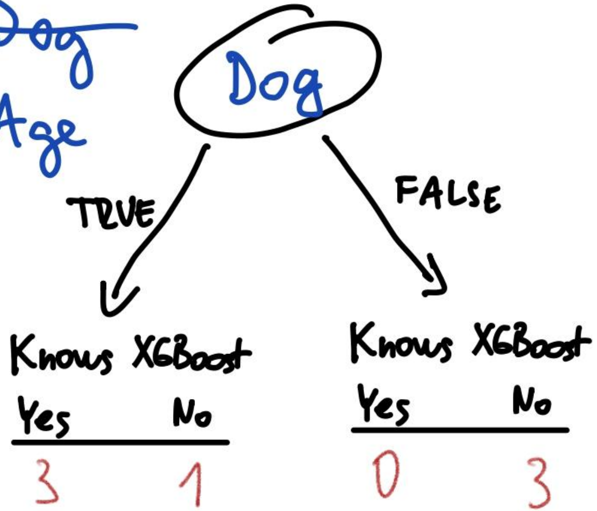


Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO

~~Woman~~

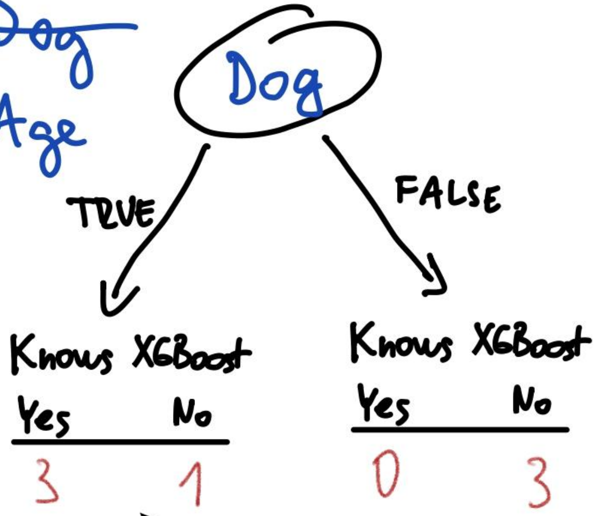
~~Dog~~

Age



Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO

~~Woman~~
~~Dog~~
Age



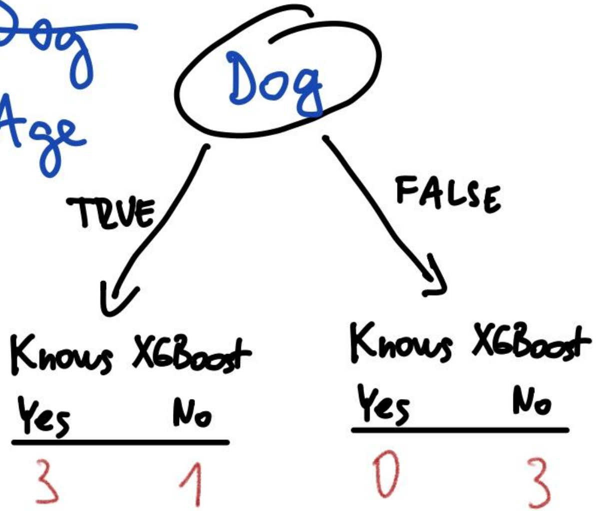
Only this leaf is impure

Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO

~~Woman~~

~~Dog~~

Age



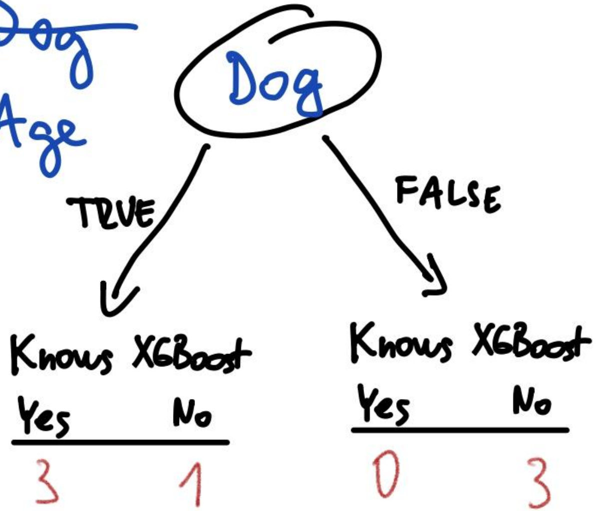
How to quantify which predictor is better?

Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO

~~Woman~~

~~Dog~~

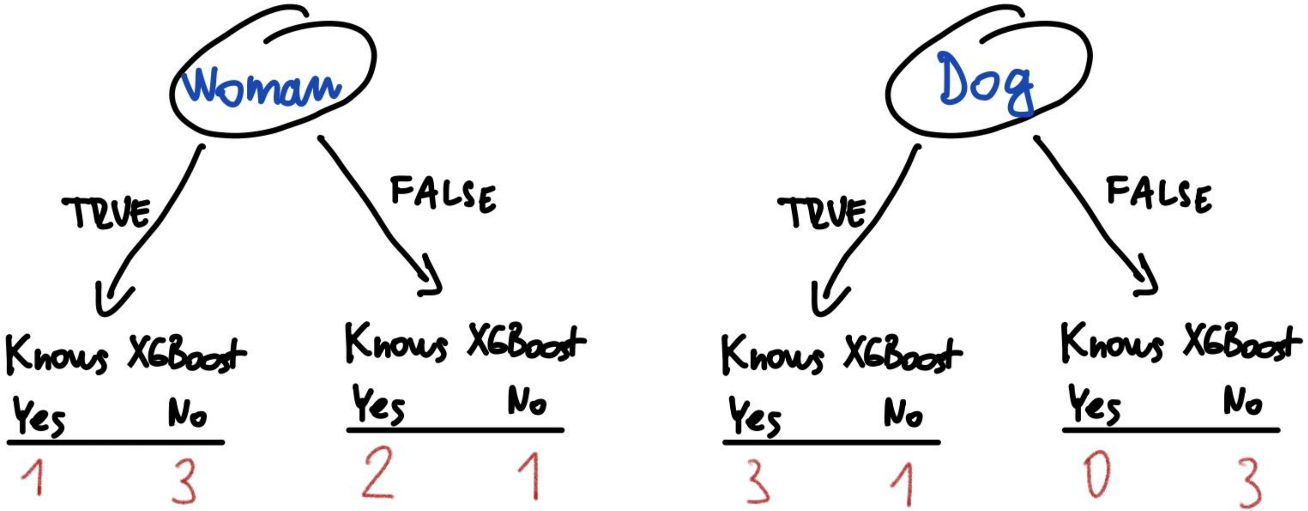
Age



→ Gini impurity

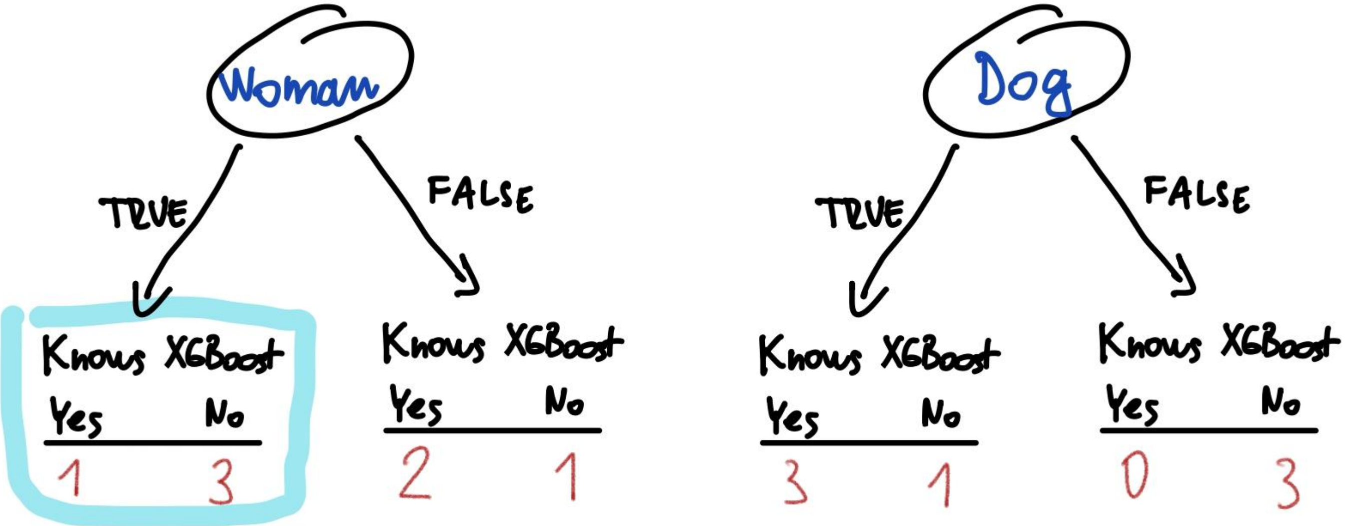
(Entropy, Information Gain)

CALCULATING GINI IMPURITY



G.I. for a Leaf = $1 - (P[Yes])^2 - (P[No])^2$

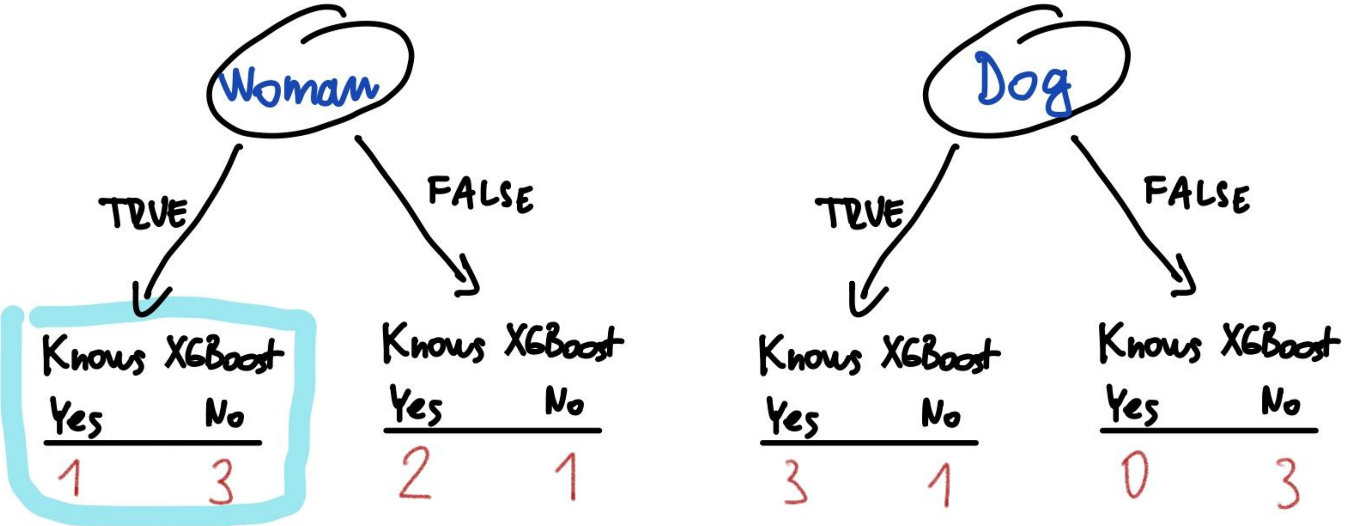
CALCULATING GINI IMPURITY



G.I. for a Leaf = 1 - (P[Yes])^2 - (P[No])^2

$$= 1 - \left(\frac{1}{1+3}\right)^2 - \left(\frac{3}{1+3}\right)^2$$

CALCULATING GINI IMPURITY

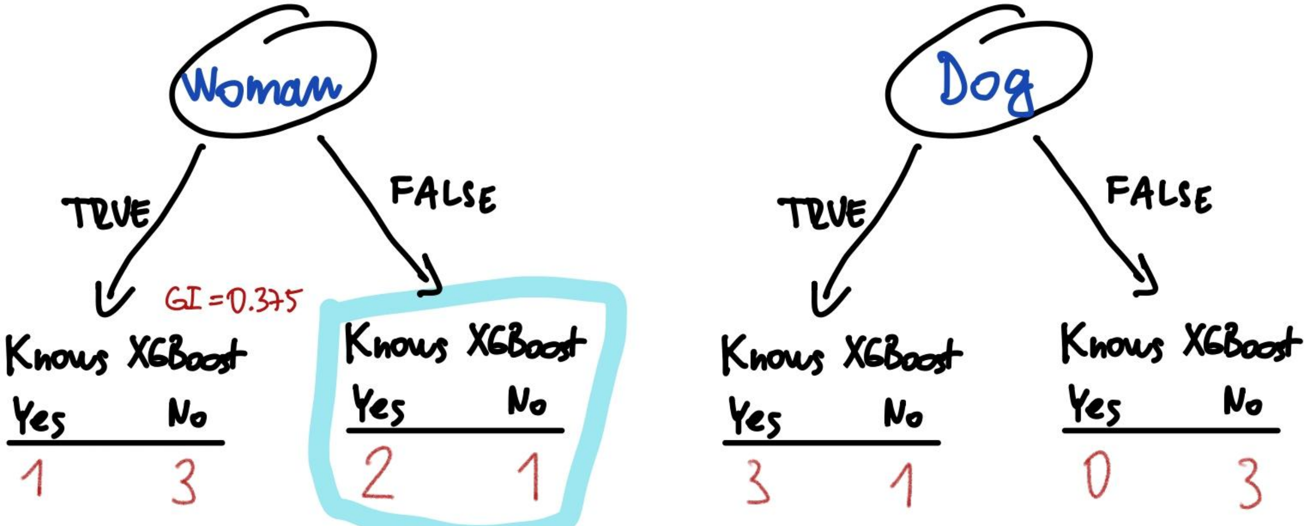


G.I. for a Leaf = $1 - (P[\text{Yes}])^2 - (P[\text{No}])^2$

$$= 1 - \left(\frac{1}{1+3}\right)^2 - \left(\frac{3}{1+3}\right)^2$$

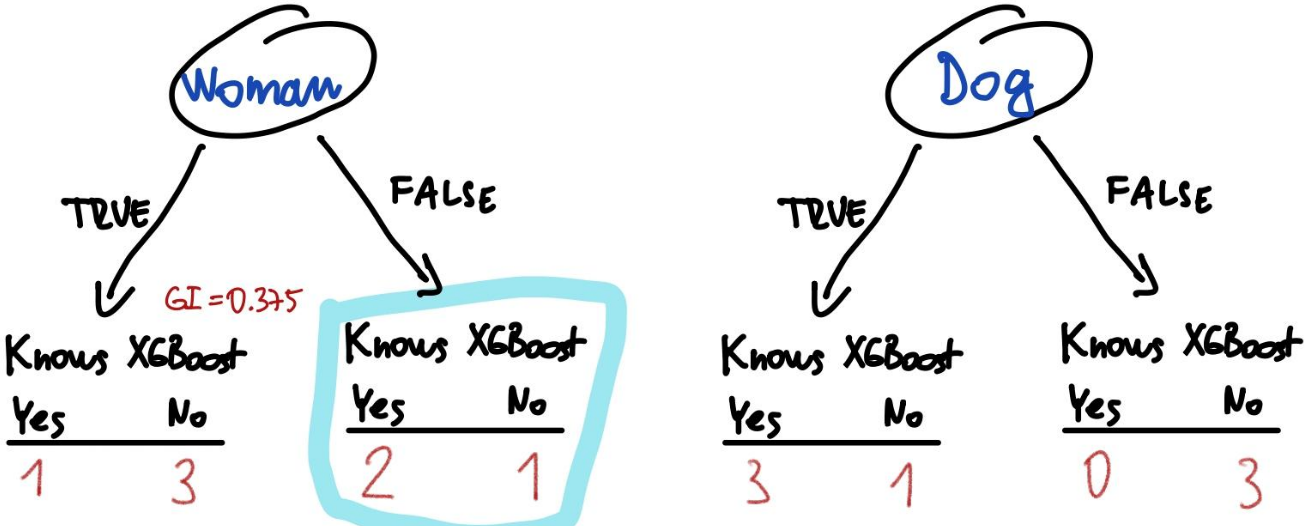
$$= \underline{0.375}$$

CALCULATING GINI IMPURITY



G.I. for a Leaf = $1 - (P[Yes])^2 - (P[No])^2$

CALCULATING GINI IMPURITY

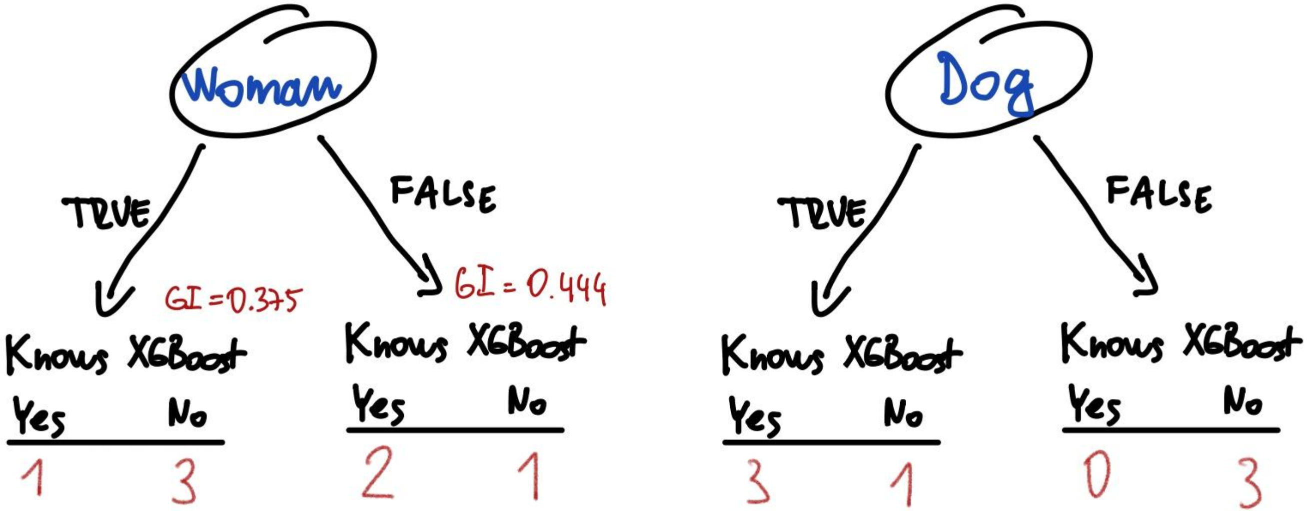


$$\text{G.I. for a Leaf} = 1 - (P[\text{Yes}])^2 - (P[\text{No}])^2$$

$$= 1 - \left(\frac{2}{2+1}\right)^2 - \left(\frac{1}{2+1}\right)^2$$

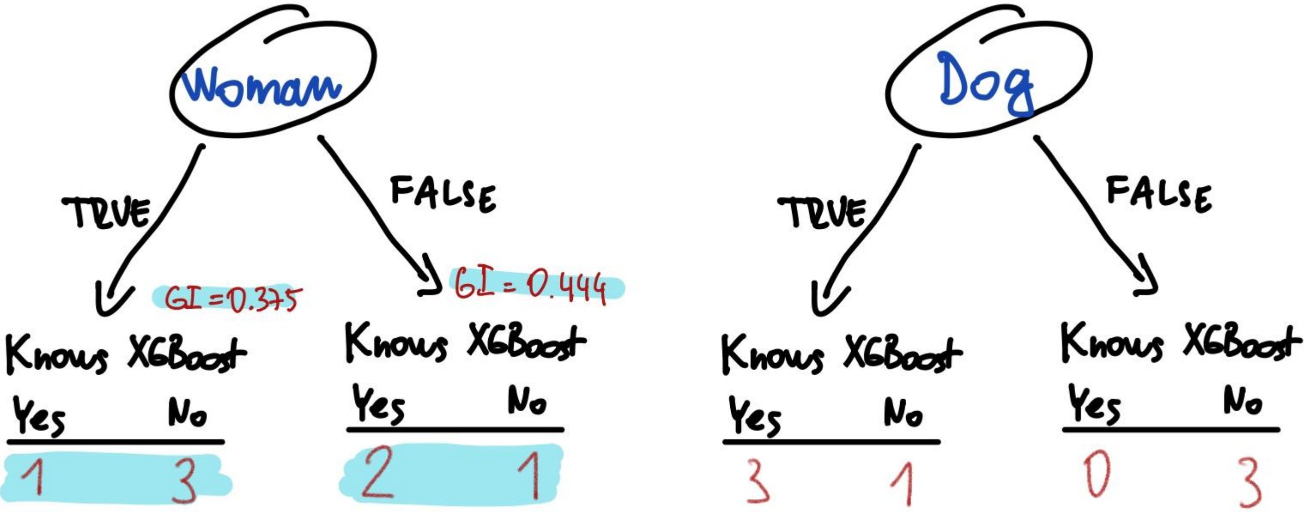
$$= 0.444$$

CALCULATING GINI IMPURITY



Total Gini impurity = weighted average of Gini Impurities for the Leaves

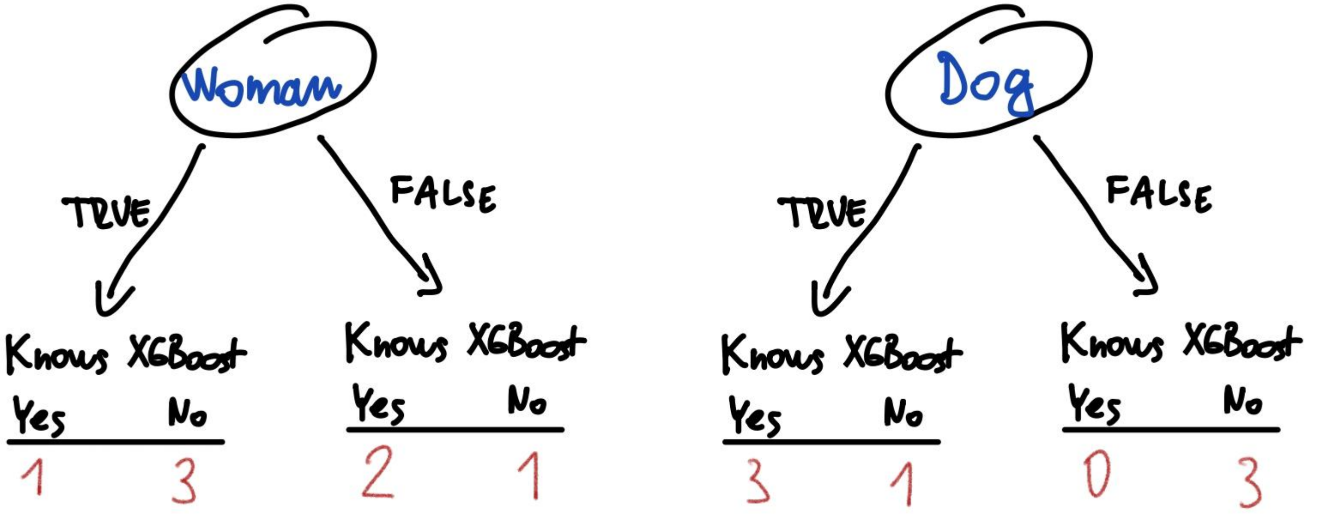
CALCULATING GINI IMPURITY



Total Gini impurity = weighted average of Gini Impurities for the Leaves

$$= \frac{4}{4+3} \cdot 0.375 + \frac{3}{4+3} \cdot 0.444 = \underline{0.405}$$

CALCULATING GINI IMPURITY



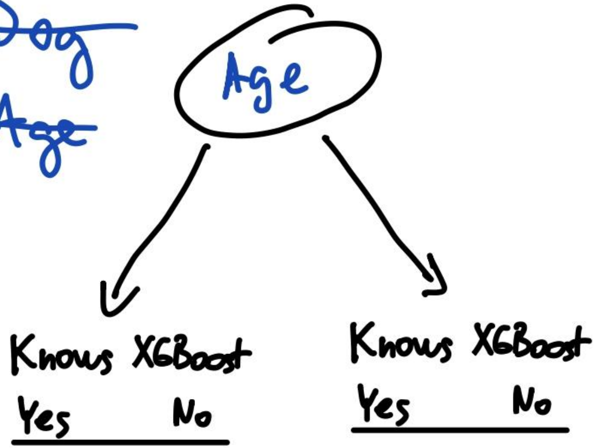
$GI_{Woman} = 0.405$ $GI_{Dog} = 0.214$

Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO

~~Woman~~

~~Dog~~

~~Age~~

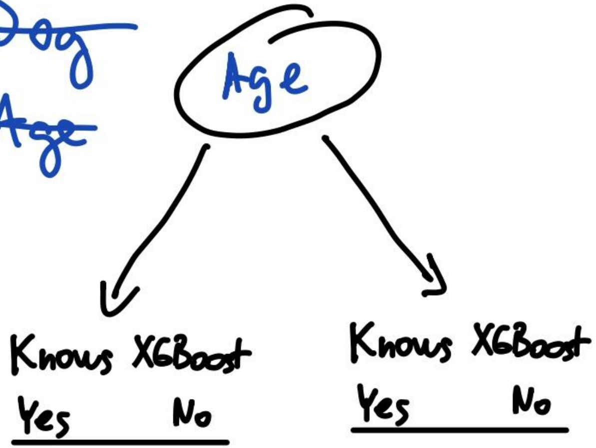


Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO

~~Woman~~

~~Dog~~

~~Age~~



How to calculate Gini Impurity with numerical values?

1. SORT AGE IN ASCENDING ORDER

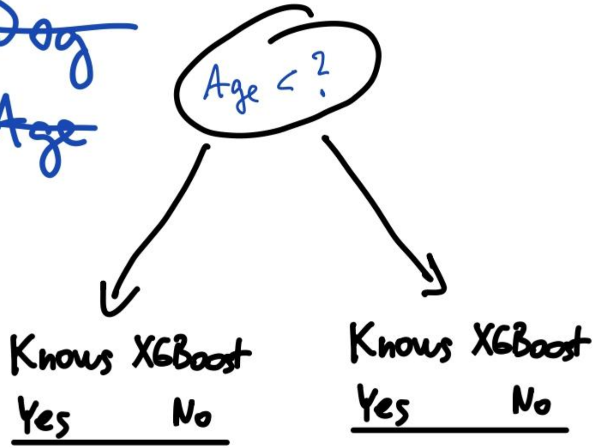
2. CALCULATE AVG FOR ADJACENT PEOPLE

Age	Knows XGBoost
7	NO
12	NO
18	YES
35	YES
38	YES
50	NO
83	NO

~~Woman~~

~~Dog~~

~~Age~~



1. SORT AGE IN ASCENDING ORDER

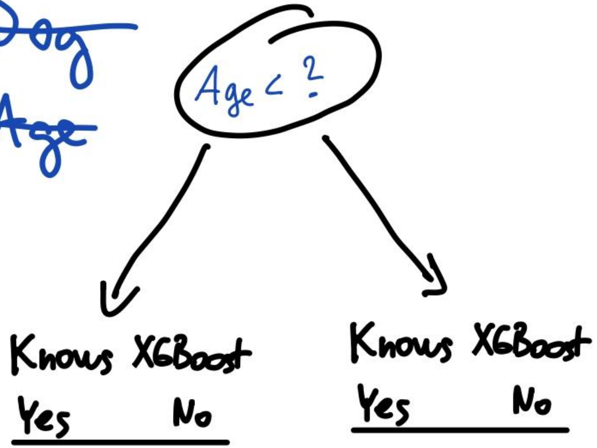
2. CALCULATE AVG FOR ADJACENT PEOPLE

	Age	Knows XGBoost
	7	NO
9.5		
	12	NO
15		
	18	YES
26.5		
	35	YES
36.5		
	38	YES
44		
	50	NO
66.5		
	83	NO

~~Woman~~

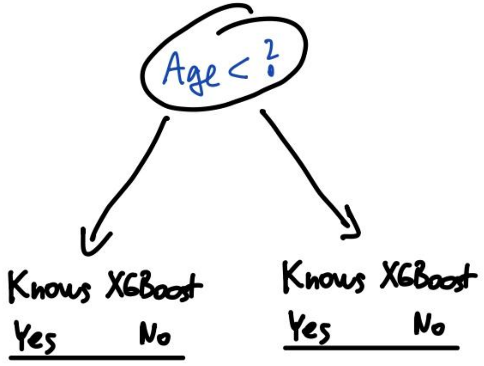
~~Dog~~

~~Age~~



1. SORT AGE IN ASCENDING ORDER
2. CALCULATE AVG FOR ADJACENT PEOPLE
3. CALCULATE GINI IMPURITY FOR ALL AVERAGE AGES.

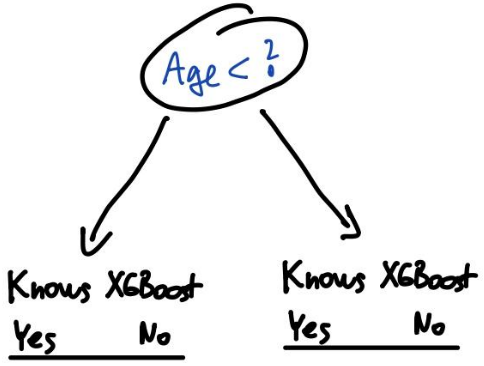
	Age	Knows XGBoost	GI
	7	NO	
9.5			0.429
	12	NO	
15			0.343
	18	YES	
26.5			0.476
	35	YES	
36.5			0.476
	38	YES	
44			0.343
	50	NO	
66.5			0.429
	83	NO	



1. SORT AGE IN ASCENDING ORDER
2. CALCULATE AVG FOR ADJACENT PEOPLE
3. CALCULATE GINI IMPURITY FOR ALL AVERAGE AGES.

Age	Knows XGBoost
7	NO
12	NO
18	YES
35	YES
38	YES
50	NO
83	NO

	GI
9.5	0.429
15	0.343
26.5	0.476
36.5	0.476
44	0.343
66.5	0.429

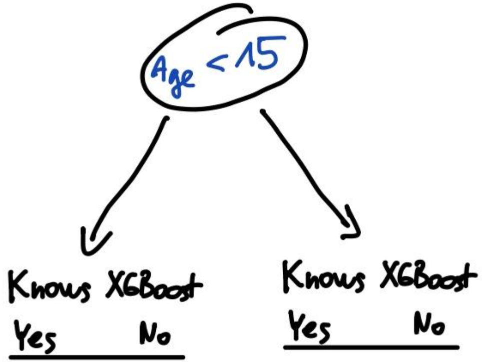


TWO CANDIDATES FOR THRESHOLD

1. SORT AGE IN ASCENDING ORDER
2. CALCULATE AVG FOR ADJACENT PEOPLE
3. CALCULATE GINI IMPURITY FOR ALL AVERAGE AGES.

Age	Knows XGBoost
7	NO
12	NO
18	YES
35	YES
38	YES
50	NO
83	NO

	GI
9.5	0.429
15	0.343
26.5	0.476
36.5	0.476
44	0.343
66.5	0.429



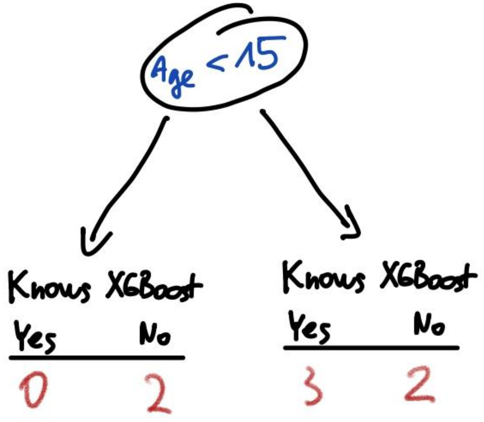
1. SORT AGE IN ASCENDING ORDER

2. CALCULATE AVG FOR ADJACENT PEOPLE

3. CALCULATE GINI IMPURITY FOR ALL AVERAGE AGES.

Age	Knows XGBoost
7	NO
12	NO
18	YES
35	YES
38	YES
50	NO
83	NO

Average Age	GI
9.5	0.429
15	0.343
26.5	0.476
36.5	0.476
44	0.343
66.5	0.429



$GI_{Age} = 0.343$

Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO

Woman
Dog
Age

?

WHICH PREDICTOR SHOULD WE CHOOSE?

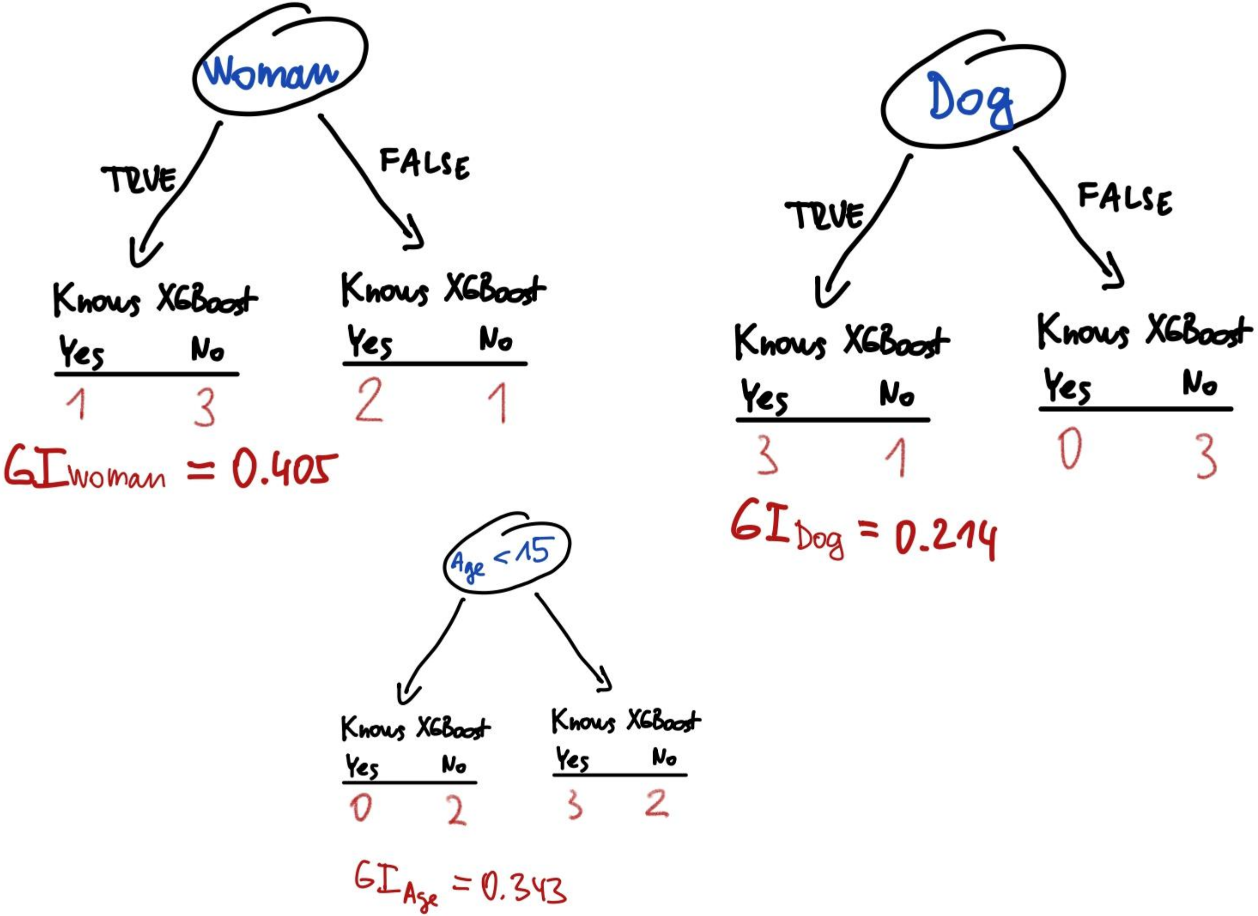
Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO

Woman
Dog
Age

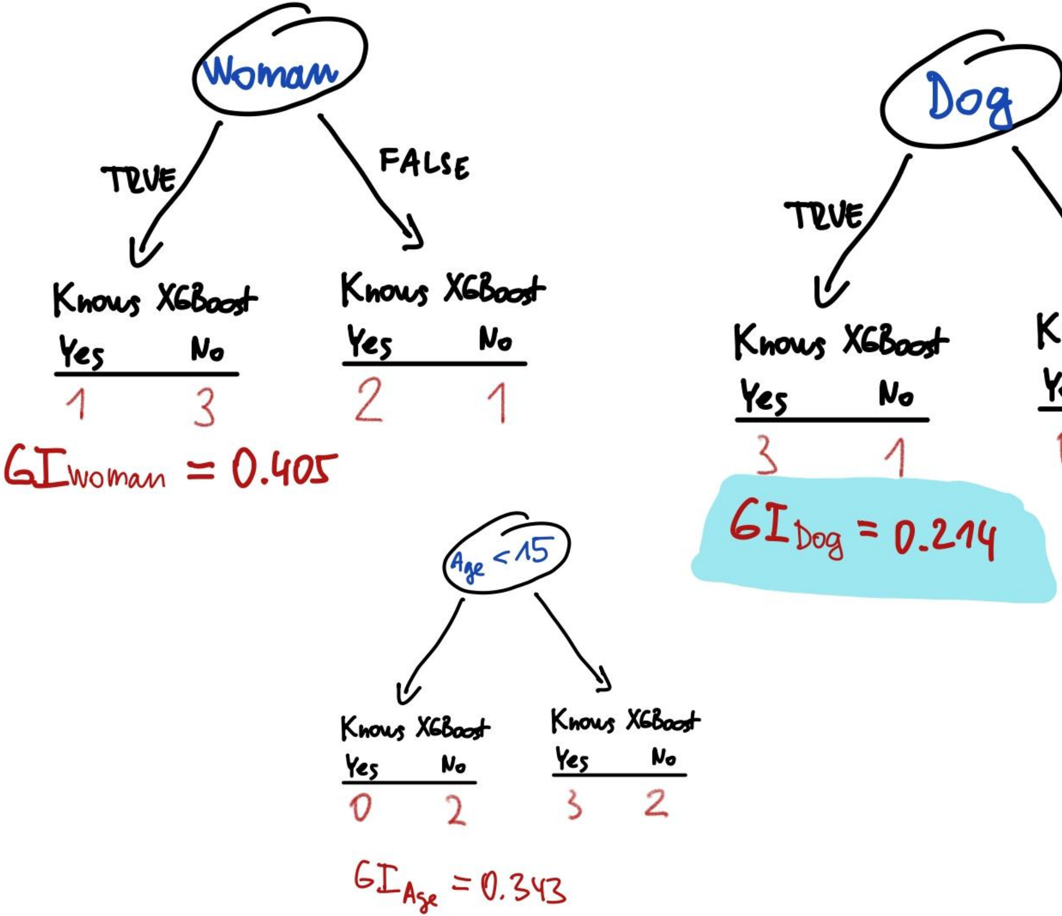
?

WHICH PREDICTOR SHOULD WE CHOOSE?

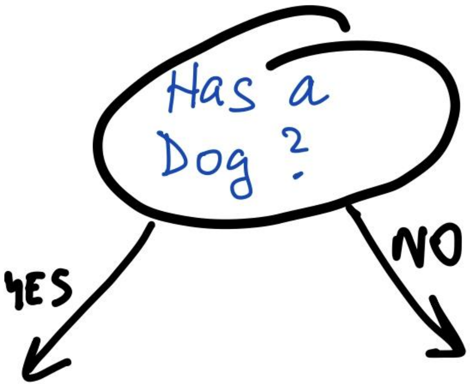
→ THE ONE WITH THE LOWEST GINI IMPURITY



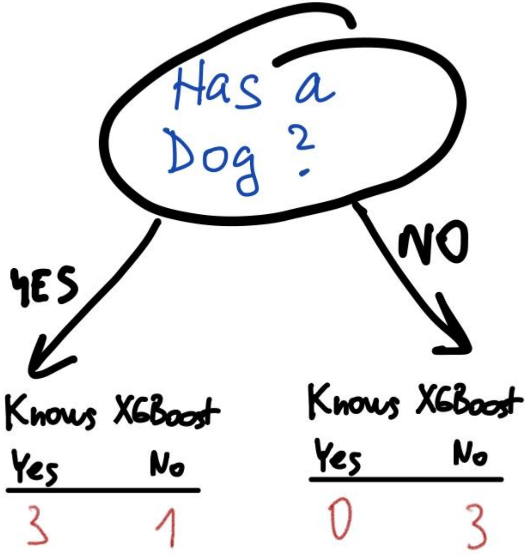
Woman
TRUE
FALSE
Knows XGBoost
Yes No
1 3
Knows XGBoost
Yes No
2 1
GI woman = 0.405
Dog
TRUE
FALSE
Knows XGBoost
Yes No
3 1
Knows XGBoost
Yes No
0 3
GI Dog = 0.214
Age < 15
Knows XGBoost
Yes No
0 2
Knows XGBoost
Yes No
3 2
GI Age = 0.343


Woman
TRUE
FALSE
Knows XGBoost
Yes No
1 3
Knows XGBoost
Yes No
2 1
GI woman = 0.405
Dog
TRUE
FALSE
Knows XGBoost
Yes No
3 1
Knows XGBoost
Yes No
0 3
GI Dog = 0.214
Age < 15
Knows XGBoost
Yes No
0 2
Knows XGBoost
Yes No
3 2
GI Age = 0.343

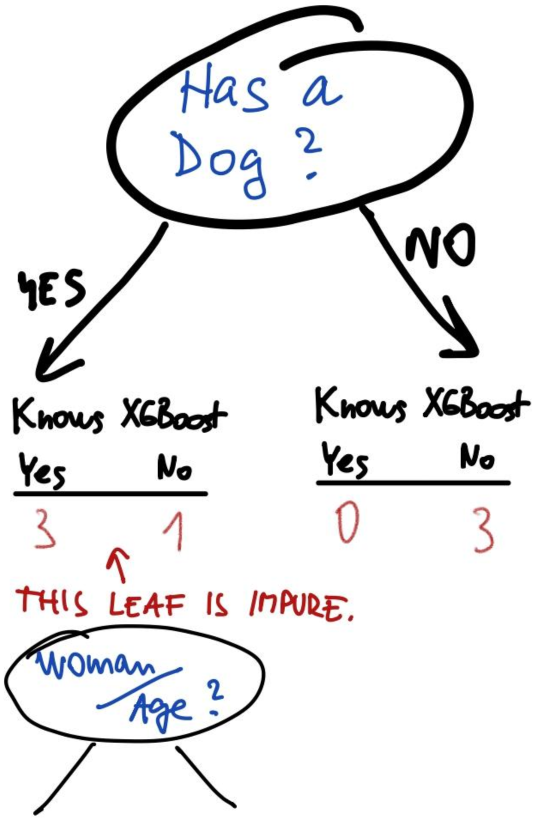
Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO



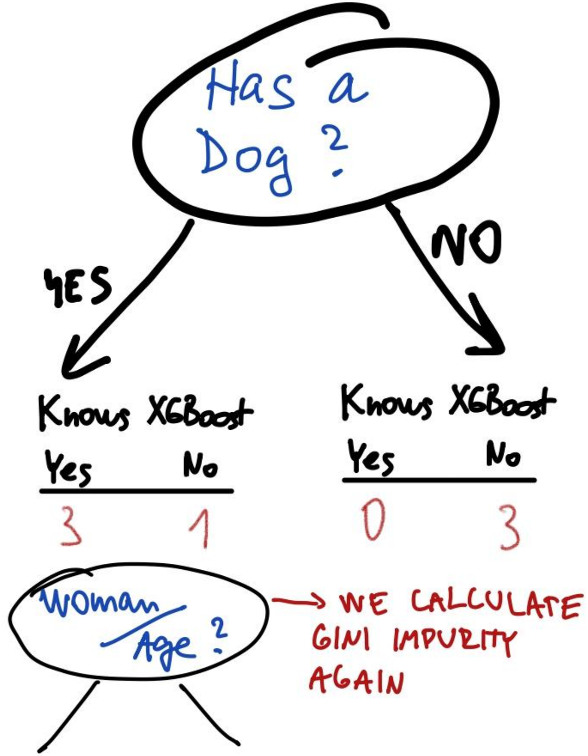
Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO



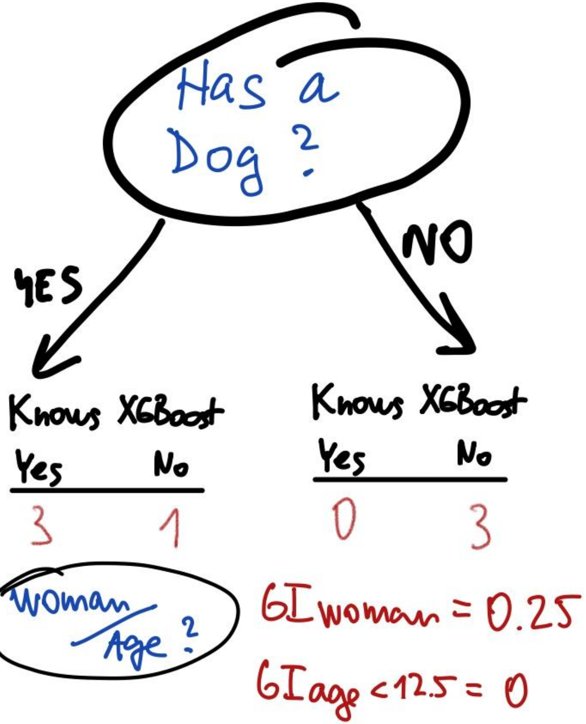
Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO



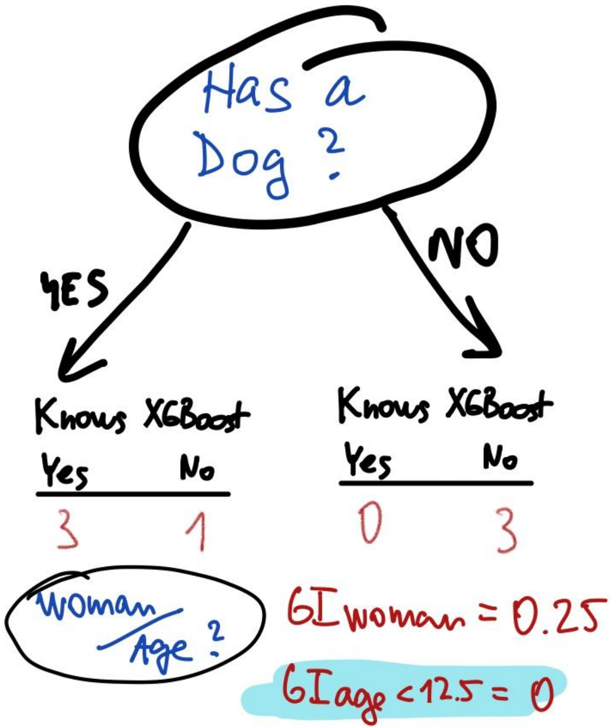
Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO



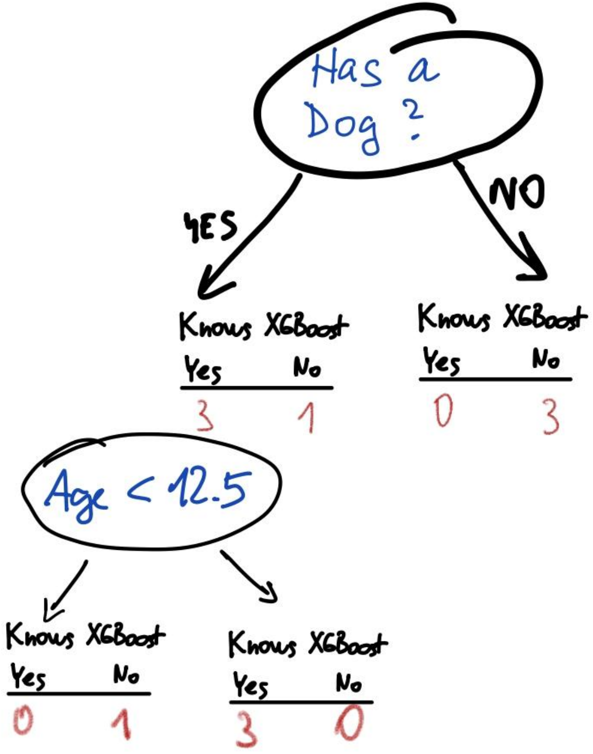
Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO



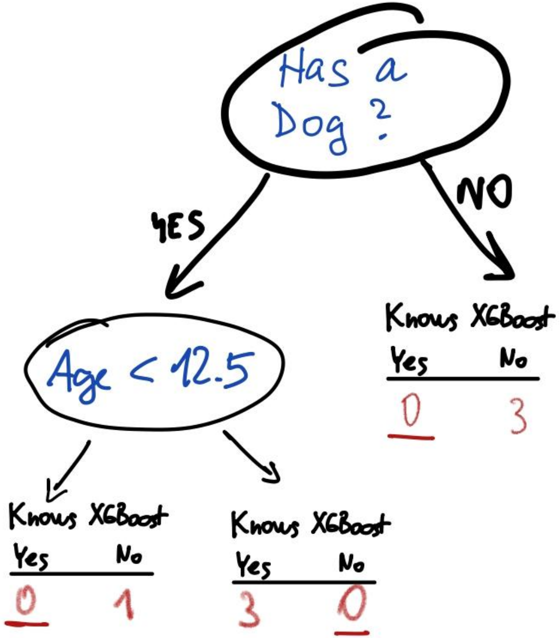
Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO



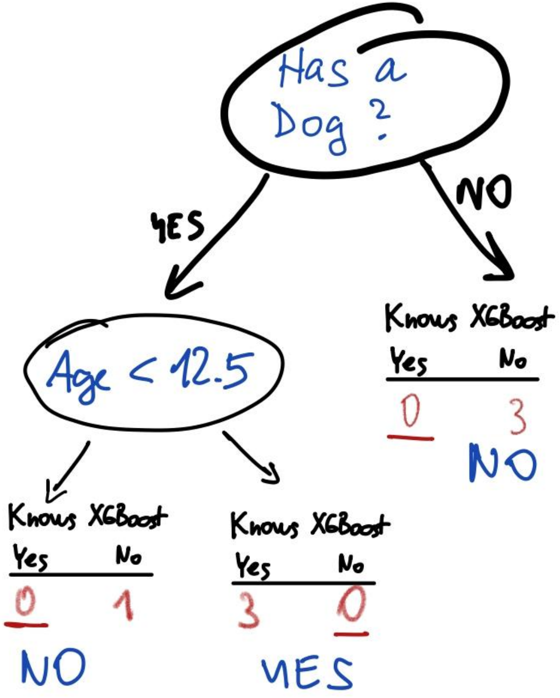
Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO



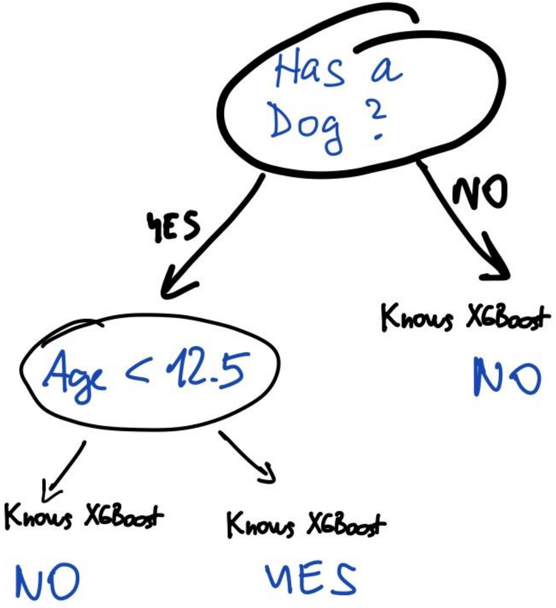
Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO



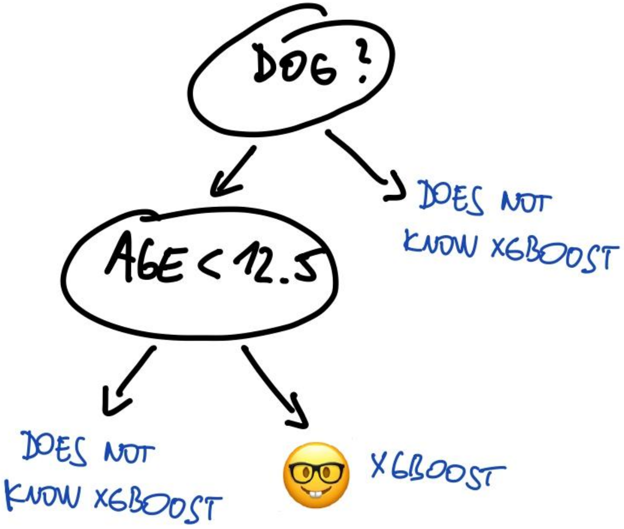
Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO



Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO



Is a woman	Has a dog	Age	Knows XGBoost
YES	YES	7	NO
YES	NO	12	NO
NO	YES	18	YES
NO	YES	35	YES
YES	YES	38	YES
YES	NO	50	NO
NO	NO	83	NO



Building trees (summary)

Greedy algorithm:

1. consider all features
2. consider all possible thresholds
3. compute impurity or loss reduction
4. pick the best split
5. recurse on left and right branch
6. stop when: max depth OR not enough samples OR improvement small

Pros and Cons of Decision Trees

- interpretable
- no scaling
- both numerical and categorical
- fast
- nonlinear relationships

✅ 👍

- overfitting
- high variance
- instability
- not very accurate

☹️ ❌

Pros and Cons of Decision Trees

- overfitting
- high variance
- instability
- not very accurate

Solution?
-> Ensemble methods

☹️ ❌

Ensemble Methods: From Weak Learners to Robust Predictors

Motivation

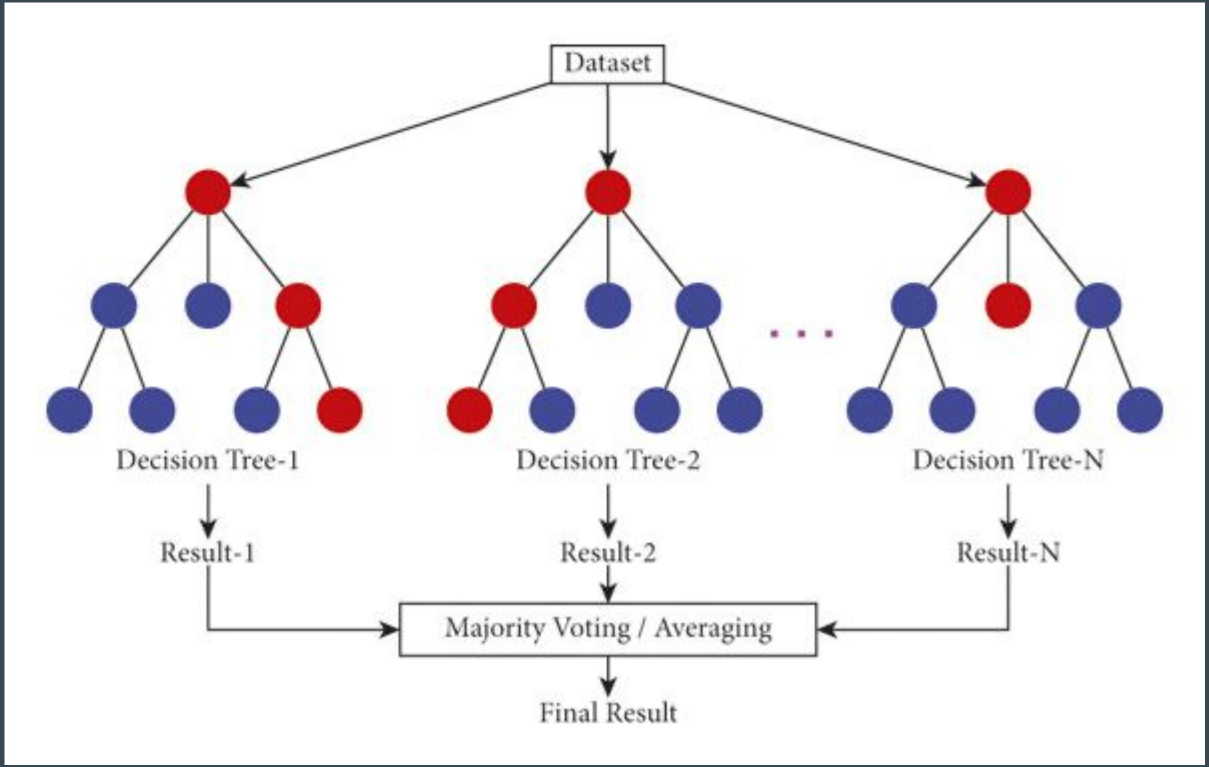
- Reduce variance – single tree highly unstable
- Reduce bias –
    - shallow trees → too simple, high bias
    - deep trees → overfit, overly complex

- Bagging (Random Forests)
- Boosting (XGBoost, LightGBM, CatBoost)

Bagging (Random Forest)

= bootstrap aggregating

idea: lower variance by averaging multiple results



Building Random Forest

1. create a bootstrapped dataset
2. build decision tree using random subset of features at each step

CREATE BOOTSTRAPPED DATASET

Watch TV	Weight	Is a woman	Has a dog	Age	Knows XGBoost
YES	90	YES	YES	7	NO
NO	108	YES	NO	12	NO
NO	172	NO	YES	18	YES
NO	186	NO	YES	35	YES
NO	152	YES	YES	38	YES
YES	149	YES	NO	50	NO
YES	207	NO	NO	83	NO

CREATE BOOTSTRAPPED DATASET

	Watch TV	Weight	Is a woman	Has a dog	Age	Knows XGBoost
1	YES	90	YES	YES	7	NO
2	NO	108	YES	NO	12	NO
3	NO	172	NO	YES	18	YES
4	NO	186	NO	YES	35	YES
5	NO	152	YES	YES	38	YES
6	YES	149	YES	NO	50	NO
7	YES	207	NO	NO	83	NO

= SAMPLING WITH REPLACEMENT N. SAMPLES = # rows

E.G. 7, 2, 1, 4, 7, 3, 1

CREATE BOOTSTRAPPED DATASET

	Watch TV	Weight	Is a woman	Has a dog	Age	Knows XGBoost
1	YES	90	YES	YES	7	NO
2	NO	108	YES	NO	12	NO
3	NO	172	NO	YES	18	YES
4	NO	186	NO	YES	35	YES
~~5~~	~~NO~~	~~152~~	~~YES~~	~~YES~~	~~38~~	~~YES~~
~~6~~	~~YES~~	~~149~~	~~YES~~	~~NO~~	~~50~~	~~NO~~
7	YES	207	NO	NO	83	NO

= SAMPLING WITH REPLACEMENT N. SAMPLES = # rows

E.G. 7, 2, 1, 4, 7, 3, 1

CREATE BOOTSTRAPPED DATASET

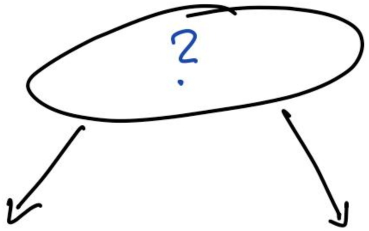
	Watch TV	Weight	Is a woman	Has a dog	Age	Knows XGBoost
1	YES	90	YES	YES	7	NO
2	NO	108	YES	NO	12	NO
3	NO	172	NO	YES	18	YES
4	NO	186	NO	YES	35	YES
7	YES	207	NO	NO	83	NO
1	YES	90	YES	YES	7	NO
7	YES	207	NO	NO	83	NO

= SAMPLING WITH REPLACEMENT N. SAMPLES = #rows

E.G. 7, 2, 1, 4, 7, 3, 1

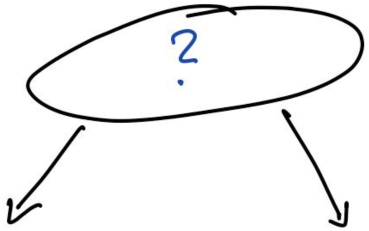
BUILD DECISION TREE

Watch TV	Weight	Is a woman	Has a dog	Age	Knows XGBoost
YES	90	YES	YES	7	NO
NO	108	YES	NO	12	NO
NO	172	NO	YES	18	YES
NO	186	NO	YES	35	YES
YES	207	NO	NO	83	NO
YES	90	YES	YES	7	NO
YES	207	NO	NO	83	NO



BUILD DECISION TREE

Watch TV	Weight	Is a woman	Has a dog	Age	Knows XGBoost
YES	90	YES	YES	7	NO
NO	108	YES	NO	12	NO
NO	172	NO	YES	18	YES
NO	186	NO	YES	35	YES
YES	207	NO	NO	83	NO
YES	90	YES	YES	7	NO
YES	207	NO	NO	83	NO

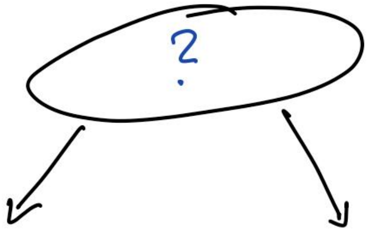


→ CHOOSE RANDOM SUBSET OF VARIABLES

E.G., [Has a Dog, Age]

BUILD DECISION TREE

Watch TV	Weight	Is a woman	Has a dog	Age	Knows XGBoost
YES	90	YES	YES	7	NO
NO	108	YES	NO	12	NO
NO	172	NO	YES	18	YES
NO	186	NO	YES	35	YES
YES	207	NO	NO	83	NO
YES	90	YES	YES	7	NO
YES	207	NO	NO	83	NO



→ CHOOSE RANDOM SUBSET OF VARIABLES

E.G., [Has a Dog, Age]

→ DETERMINE THE BEST CANDIDATE BY CALCULATING GINI IMPURITY

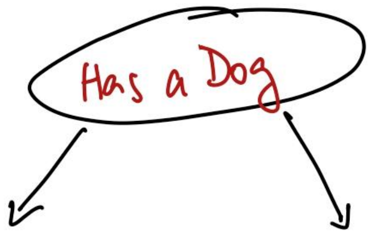
$GI_{Dog} = \dots$ $GI_{Age} = \dots$

BUILD DECISION TREE

Watch TV	Weight	Is a woman	Has a dog	Age	Knows XGBoost
YES	90	YES	YES	7	NO
NO	108	YES	NO	12	NO
NO	172	NO	YES	18	YES
NO	186	NO	YES	35	YES
YES	207	NO	NO	83	NO
YES	90	YES	YES	7	NO
YES	207	NO	NO	83	NO

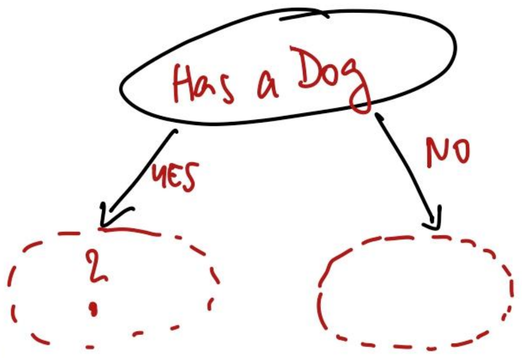
→ DETERMINE THE BEST CANDIDATE BY CALCULATING GINI IMPURITY

Assume $GI_{Dog} < GI_{Age}$



BUILD DECISION TREE

Watch TV	Weight	Is a woman	Has a dog	Age	Knows XGBoost
YES	90	YES	YES	7	NO
NO	108	YES	NO	12	NO
NO	172	NO	YES	18	YES
NO	186	NO	YES	35	YES
YES	207	NO	NO	83	NO
YES	90	YES	YES	7	NO
YES	207	NO	NO	83	NO



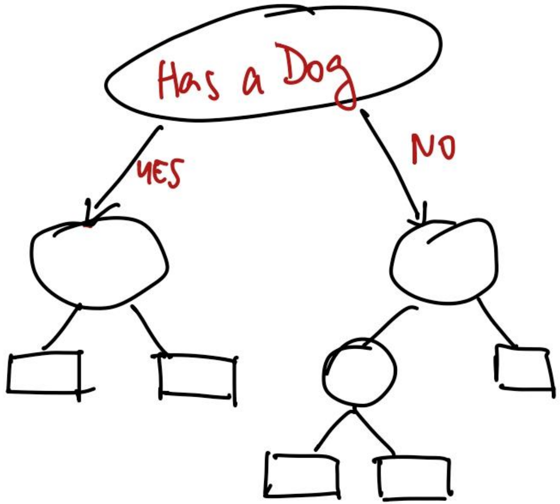
→ CHOOSE RANDOM SUBSET OF VARIABLES AGAIN

→ SELECT THE BEST SPLIT

→ BUILD THE FULL TREE

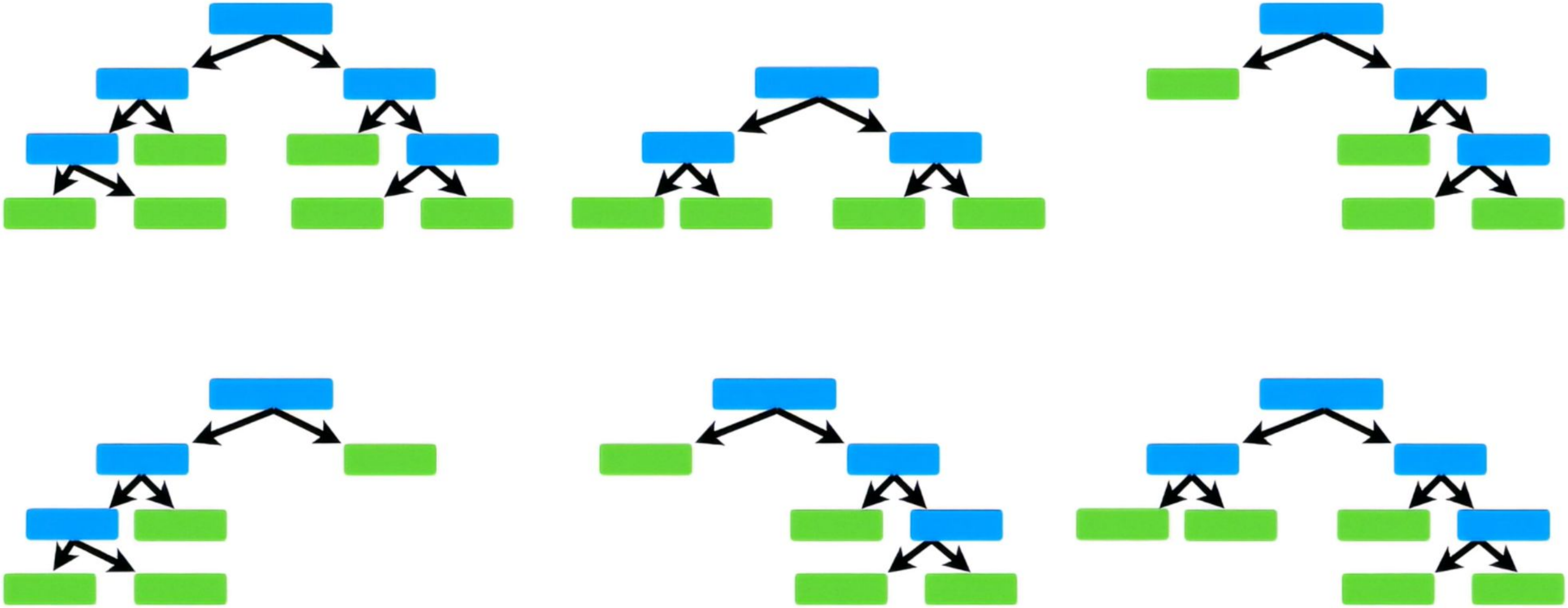
BUILD DECISION TREE

Watch TV	Weight	Is a woman	Has a dog	Age	Knows XGBoost
YES	90	YES	YES	7	NO
NO	108	YES	NO	12	NO
NO	172	NO	YES	18	YES
NO	186	NO	YES	35	YES
YES	207	NO	NO	83	NO
YES	90	YES	YES	7	NO
YES	207	NO	NO	83	NO



Building Random Forest

1. create a bootstrapped dataset
2. build decision tree using random subset of features at each step
3. repeat for n=number of trees times



Obtaining Prediction

- we evaluate all individual trees in random forest
- aggregate the result
  - majority vote / average value

- bootstrapped dataset + aggregating = bagging

Limitations of Bagging

✅ Pros of Random Forests

- variance reduction
- + stability

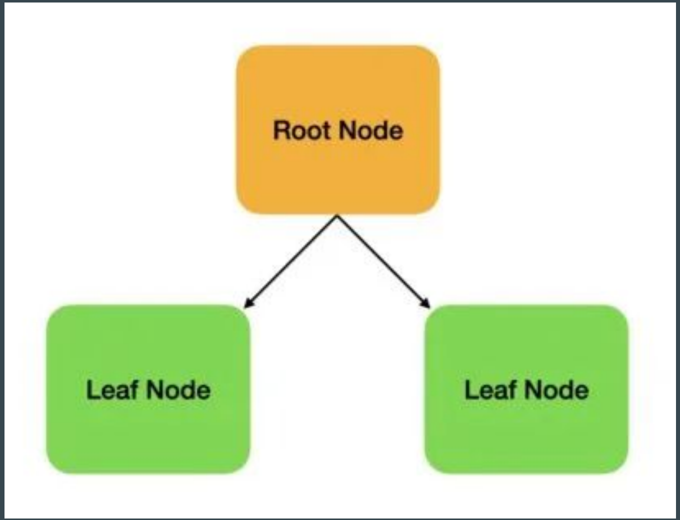
❌ Limitations of Random Forests

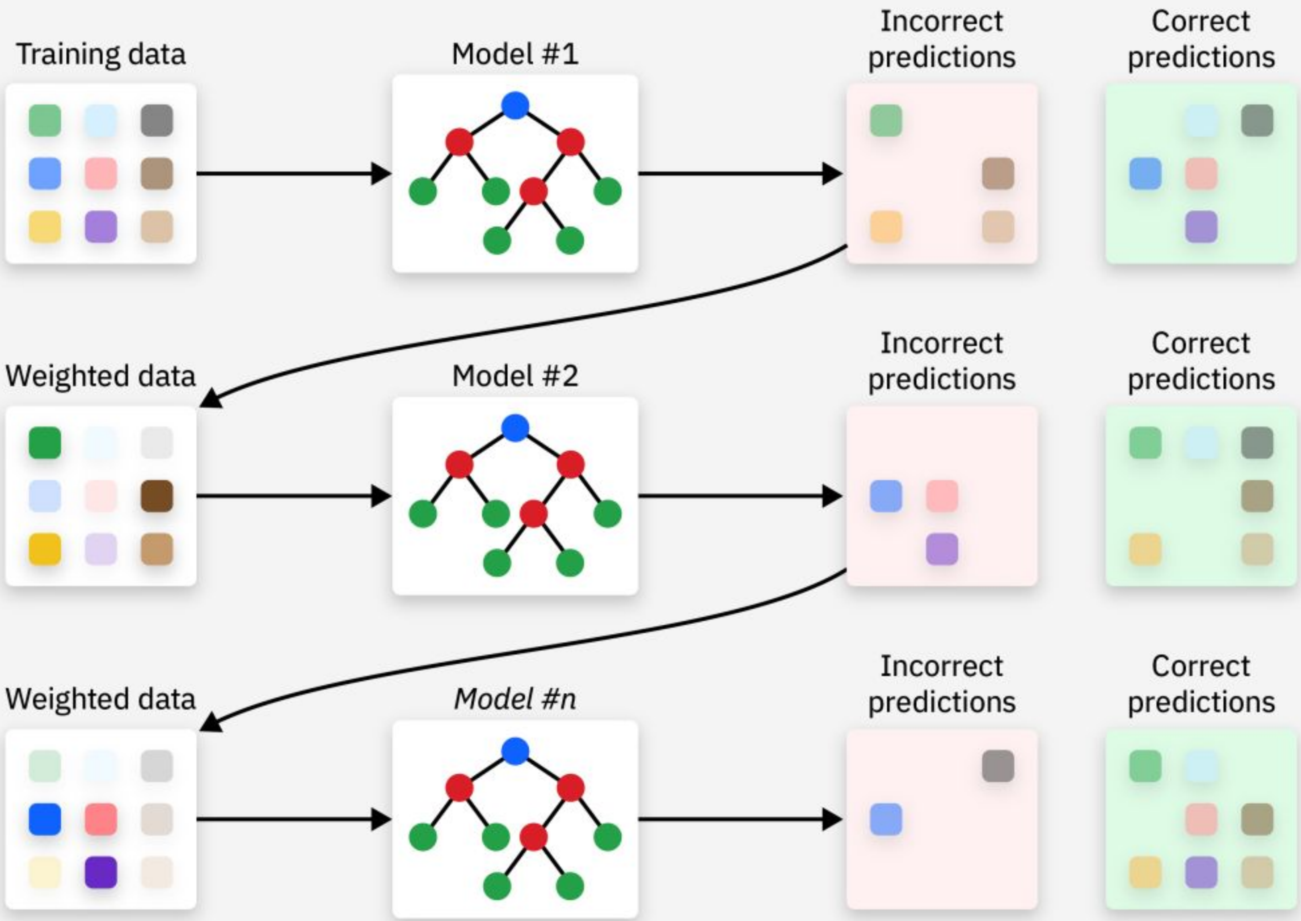
- averaging does not reduce **bias**
- slower inference
- trees built independently: they don't learn from previous errors

- What if we could build trees that *learn from each other*?

Boosting

- idea: sequentially build trees correcting error of previous model
- increase weight of misclassified examples
- usually shallow trees / stumps
- combine predictions
- AdaBoost, Gradient Boosting




Training data
Model #1
Incorrect predictions
Correct predictions
Weighted data
Model #2
Incorrect predictions
Correct predictions
Weighted data
Model #n
Incorrect predictions
Correct predictions

Gradient Boosting

- gradient descent to minimize loss function
- -> gradient-guided correction

Algorithm 1: Gradient_Boost

$F_0(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^{N} L\left(y_i, \rho\right)$

For $m = 1$ to M do:

$\quad \tilde{y}_i = -\left[\frac{\partial L(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}, \; i = 1, N$

$\quad \mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^{N} [\tilde{y}_i - \beta h(\mathbf{x}_i; \mathbf{a})]^2$

$\quad \rho_m = \arg\min_\rho \sum_{i=1}^{N} L\left(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i; \mathbf{a}_m)\right)$

$\quad F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$

endFor

end Algorithm

Step 1: initial prediction (avg)

Step 2: calculate pseudo-residuals

Step 3: build decision tree (8 to 32 leaves) fitting residuals

-> Iterate 2, 3 until stopping criterion

Age	Category	Purchase weight (kg)	Amount
25	Electronics	2.5	123.45
34	Clothing	1.3	56.78
42	Electronics	5.0	345.67
19	Hardware	3.2	98.01

$$MSE = \frac{1}{2}(\text{Observed} - \text{Predicted})^2$$

Step 1: Initial prediction

$$\frac{dL}{d\hat{y}} = (\text{predicted} - \text{observed})$$

$$\frac{123.45 + 56.78 + 345.67 + 98.01}{4} = \underline{156}$$

$$MSE = \frac{1}{2}(\text{Observed} - \text{Predicted})^2$$

Age	Category	Purchase weight (kg)	Amount	PSEUDO RESIDUALS
25	Electronics	2.5	123.45	-32.55
34	Clothing	1.3	56.78	-99.22
42	Electronics	5.0	345.67	189.45
19	Hardware	3.2	98.01	-58.01

Step 1: Initial prediction

$$\frac{dL}{d\hat{y}} = (\text{predicted} - \text{observed})$$

$$\frac{123.45 + 56.78 + 345.67 + 98.01}{4} = \underline{156}$$

Step 2: Calculate pseudo-residuals

$$MSE = \frac{1}{2}(\text{Observed} - \text{Predicted})^2$$

Age	Category	Purchase weight (kg)	Amount	PSEUDO RESIDUALS
25	Electronics	2.5	123.45	-32.55
34	Clothing	1.3	56.78	-99.22
42	Electronics	5.0	345.67	189.45
19	Hardware	3.2	98.01	-58.01

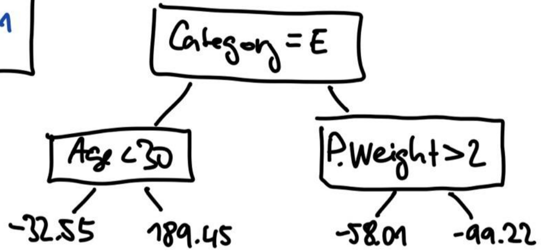
Step 1: Initial prediction

$$\frac{dL}{d\hat{y}} = (\text{predicted} - \text{observed})$$

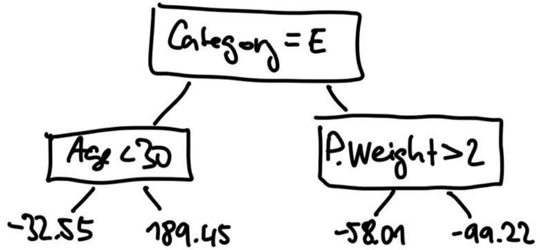
$$\frac{123.45 + 56.78 + 345.67 + 98.01}{4} = \underline{156}$$

Step 2: Calculate pseudo-residuals

Step 3: Build a weak learner



Step 3: Build a weak learner + Calculate predictions for each row



Age	Category	Purchase weight (kg)	Amount	PSEUDO RESIDUALS	NEW PREDICTIONS
25	Electronics	2.5	123.45	-32.55	152.745
34	Clothing	1.3	56.78	-99.22	146.078
42	Electronics	5.0	345.67	189.45	174.945
19	Hardware	3.2	98.01	-58.01	150.199

Initial guess   Leaf value

$$= \underbrace{156}_{\text{Initial guess}} + \underbrace{0.1}_{\text{Learning rate}} \cdot (\underbrace{-32.55}_{\text{Leaf value}})$$

Learning rate

Step 4: Iterate

→ calculate new pseudo-residuals

→ build new another weak learner

$\hat{y}$ = Initial guess + $e \cdot (r_1 + r_2 + \ldots + r_i)$ after i iterations

Age	Category	Purchase weight (kg)	Amount	PSEUDO RESIDUALS	NEW PREDICTIONS
25	Electronics	2.5	123.45		152.745
34	Clothing	1.3	56.78		146.078
42	Electronics	5.0	345.67		174.945
19	Hardware	3.2	98.01		150.199

XGBoost

- eXtreme Gradient BOOSTting
- GB with several modifications:

XGBoost

- eXtreme Gradient BOOSTting
- GB with several modifications:

Objective function

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

$$\text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

Obj = Loss + Regularization

T = number of leaves in the tree
w_j = weight (output value) of leaf j
γ = penalty for adding a leaf = pruning
λ = L2 regularization on leaf weights

XGBoost

- eXtreme Gradient BOOSTting
- GB with several modifications:

Deciding the split

-> maximizing Gain

$$\text{Gain} = \frac{1}{2}\left(\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right) - \gamma$$

G .... sum of gradients
H .... sum of hessians

CART: **unregularized** -> reduce MSE or impurity
XGBoost: regularized -> max Gain

$$\text{Gain} = \frac{1}{2}\left(\underbrace{\frac{G_L^2}{H_L+\lambda}}_{\text{LEFT NODE}} + \underbrace{\frac{G_R^2}{H_R+\lambda}}_{\text{RIGHT NODE}} - \underbrace{\frac{(G_L+G_R)^2}{H_L+H_R+\lambda}}_{\text{ROOT}}\right) - \lambda$$

MSE: $L = \frac{1}{2}(y_i - \hat{y}_i)^2$

GRADIENT $g_i = \hat{y}_i - y_i$ (residual)

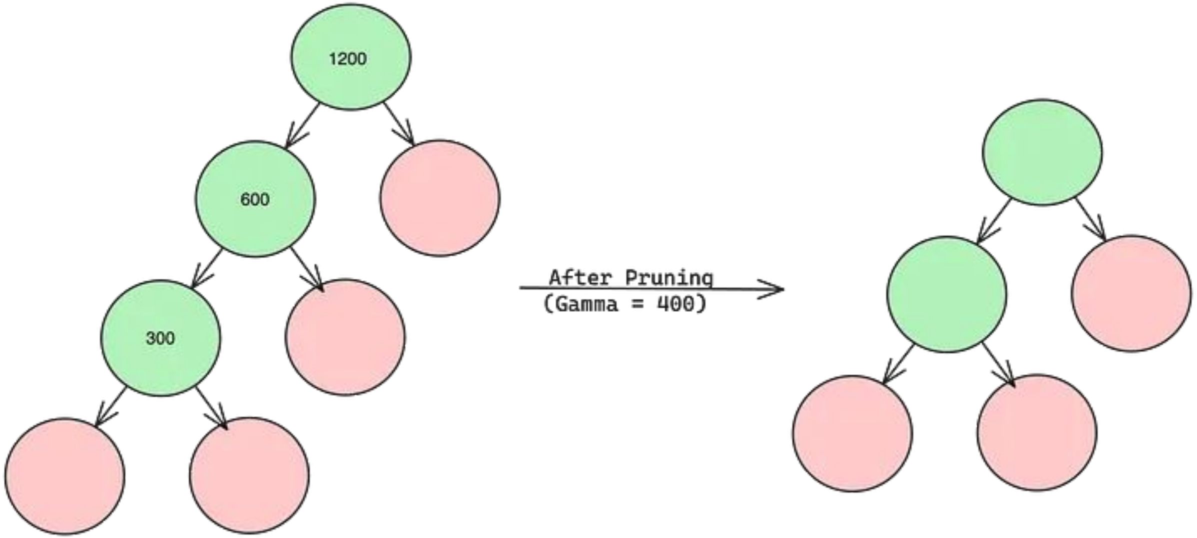
HESSIAN $h_i = 1$ (constant)

square of sum of residuals

NODE SCORE (LEFT): $\frac{G_L^2}{H_L+\lambda} \approx \frac{(\sum r_i)^2}{n_L + \lambda} = \frac{(\text{sum of residuals})^2}{\text{number of samples} + \lambda}$

CART: $\text{SCORE} = \sum r_i^2(\text{root}) - \left(\sum r_i^2(\text{left}) + \sum r_i^2(\text{right})\right)$

sum of squared residuals


1200
600
300
After Pruning
(Gamma = 400)

XGBoost

- eXtreme Gradient BOOSTting
- GB with several modifications:

Output value of a Leaf

$$w_j = -\frac{G_j}{H_j + \lambda}$$

MSE and λ=0 : average of residuals

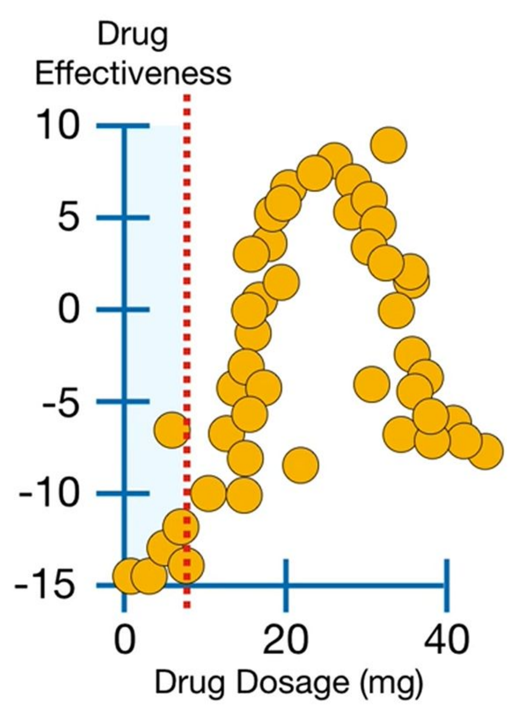
XGBoost

- eXtreme Gradient BOOSTting
- GB with several modifications
- computing predictions: same as GB

XGBoost Optimizations

Approximate Greedy Algorithm

- avoids testing every threshold
- approximation: using quantiles
    - Weighted Quantiles Sketch alg.



XGBoost Optimizations

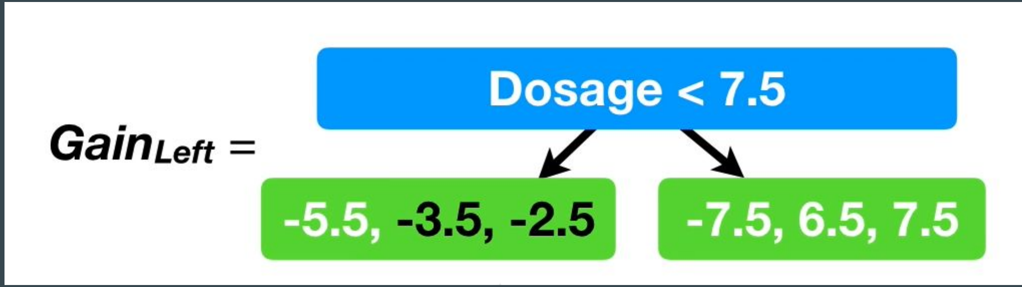
Approximate Greedy Algorithm

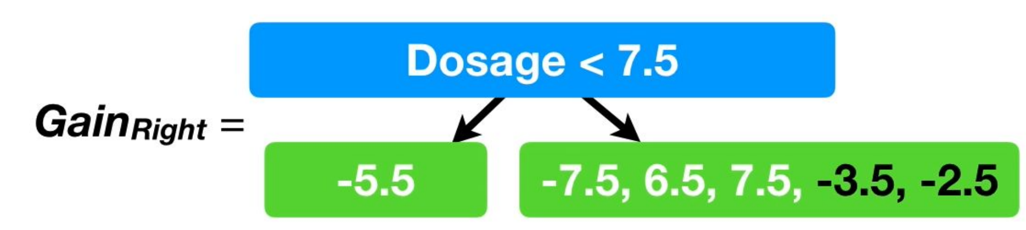
Parallel Learning

Building trees with missing data - Sparsity-Aware Split Finding

Predicted Drug Effectiveness

0.5





Dosage	Drug Effectiveness	Residuals
10	-7	-7.5
???	-3	-3.5
21	7	6.5
25	8	7.5
5	-5	-5.5
???	-2	-2.5

XGBoost Optimizations

Approximate Greedy Algorithm

Parallel Learning

Sparsity-Aware Split Finding

Cache-Aware access

-storing gradients and hessians in Cache -> speedup

XGBoost Optimizations

Approximate Greedy Algorithm

Parallel Learning

Sparsity-Aware Split Finding

Cache-Aware access

-> other memory optimizations

-> subsampling (rows and columns)

CatBoost

- = CATegorical BOOSTing
- designed for categorical data
- uses Ordered Target Encoding
- symmetric trees - same decision on the same level
    - can be fitter in parallel, very fast with GPU
- -> best for categorical-heavy datasets

LightGBM

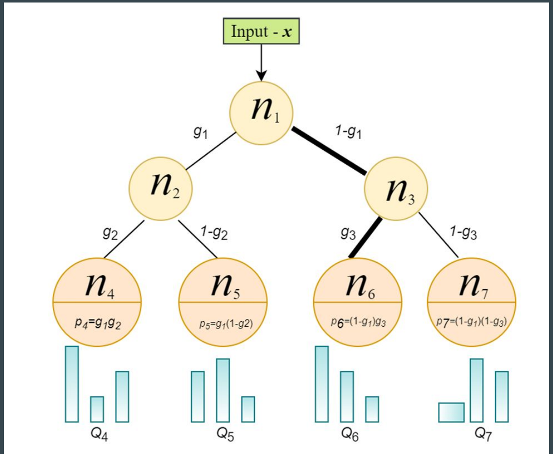
- faster
- leaf-wise tree building - splits the node with the largest split gain first
  - -> unbalanced trees
- low memory usage
- better categorical features handling than XGBoost
- tend to overfit
- -> best for fast training on big datasets

Deep Neural Models for Tabular Data

Two main categories of Deep Models

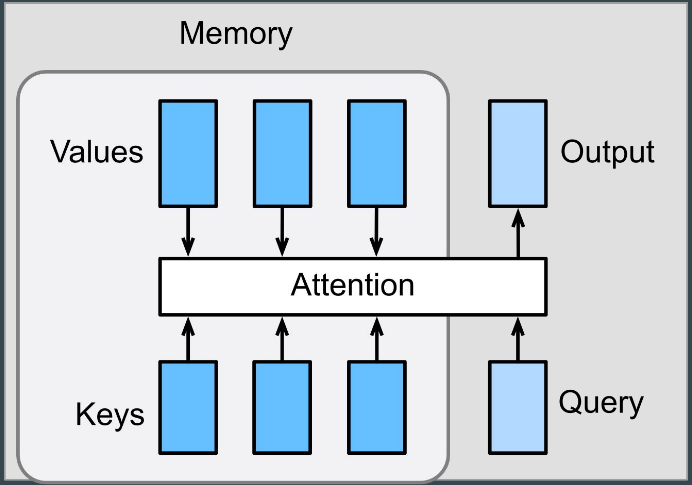
Differentiable Trees

- decision trees are not differentiable
- smoothing the decision function
- softmax, sparsemax



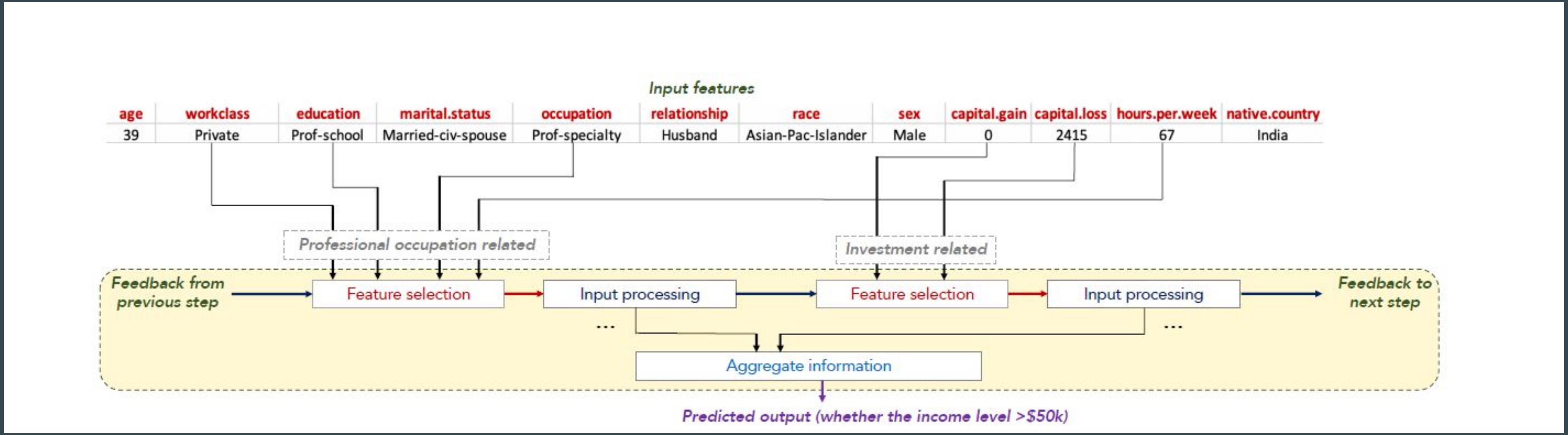
Attention-Based Models

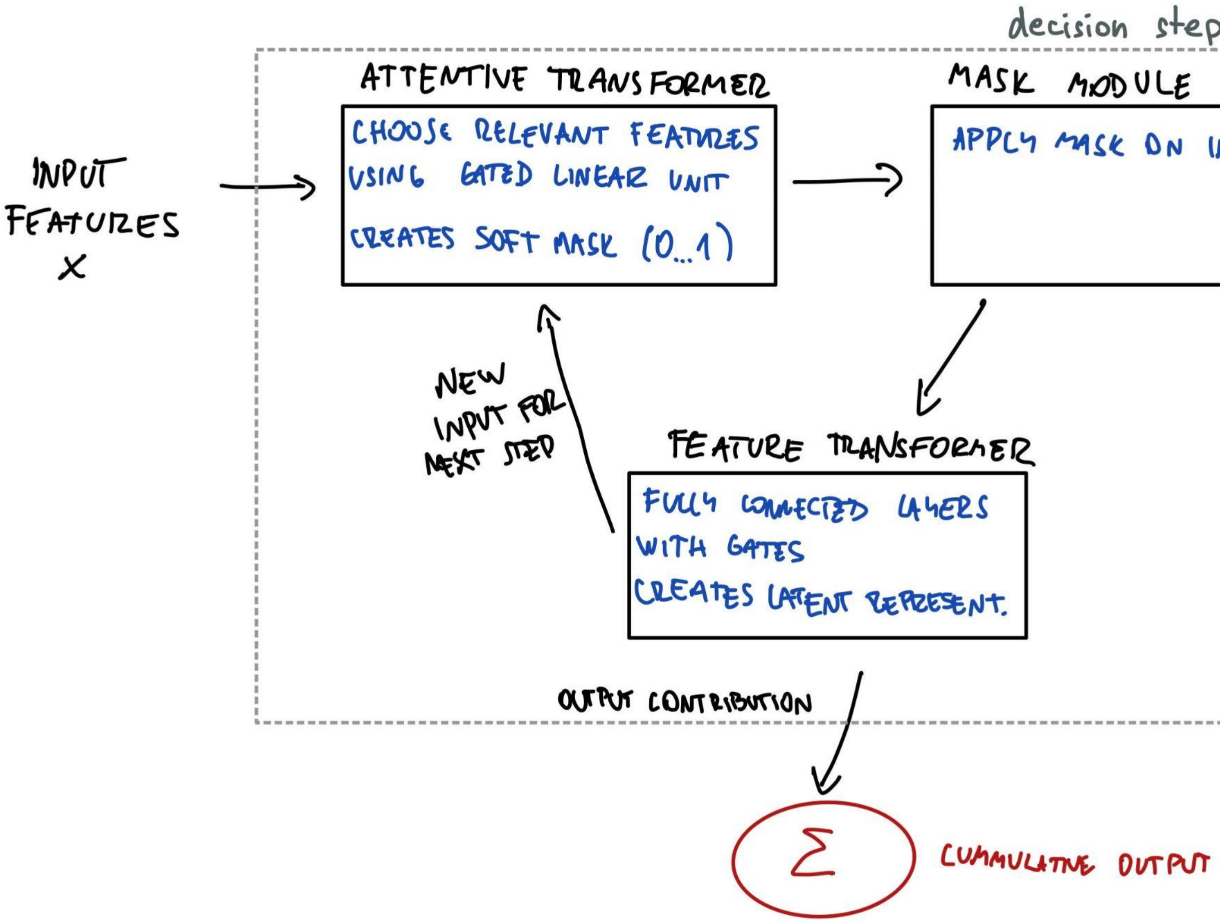
- using attention mechanism detect interactions between features or samples



TabNet

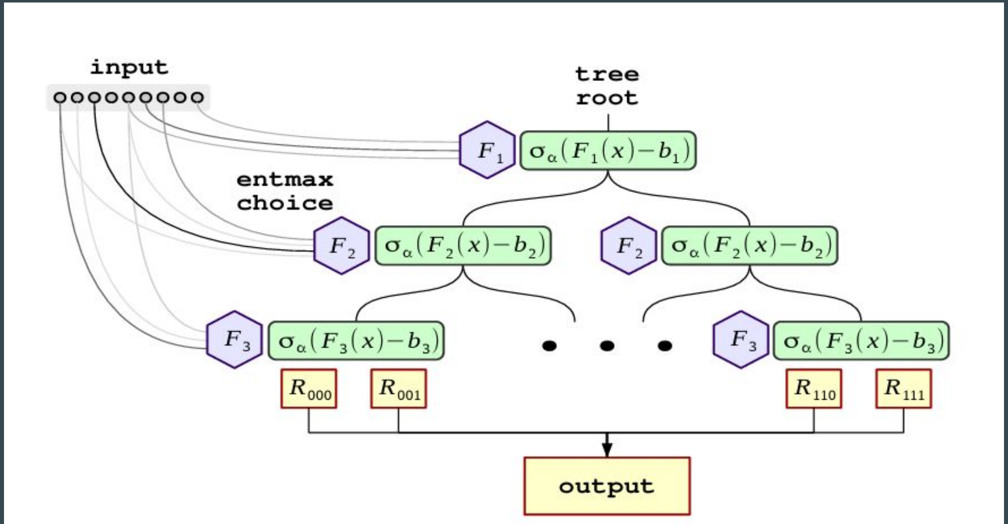
- choosing relevant features for each row using mask and attention
- different relevant features for each data input
- usually 4-10 “decision” steps




decision step
INPUT FEATURES X
ATTENTIVE TRANSFORMER
CHOOSE RELEVANT FEATURES USING GATED LINEAR UNIT
CREATES SOFT MASK (0...1)
MASK MODULE
APPLY MASK ON INPUT
FEATURE TRANSFORMER
FULLY CONNECTED LAYERS WITH GATES
CREATES LATENT REPRESENT.
NEW INPUT FOR NEXT STEP
OUTPUT CONTRIBUTION
Σ
CUMMULATIVE OUTPUT

Neural Oblivious Decision Ensembles (NODE)

- inspired by CatBoost
- ensemble of **differentiable** "trees"
- not a CART but neural layer
- ODT: all nodes in same depth use the same split rule
- soft routing: softmax instead of 0/1



DNF-Net

- simulating DNFs
- DNF-Net = ensemble of DNNF blocks
- soft AND and soft OR

$$\mathrm{OR}(\mathbf{x}) \triangleq \tanh\left(\sum_{i=1}^{d} \mathbf{x}_i + d - 1.5\right)$$

$$\mathrm{AND}(\mathbf{x}) \triangleq \tanh\left(\sum_{i=1}^{d} \mathbf{x}_i - d + 1.5\right)$$

$$(A \wedge \neg B \wedge \neg C) \vee (\neg D \wedge E \wedge F \wedge D \wedge F)$$

INPUT : $x \in \mathbb{R}^d$

LEARNABLE

DNNF Block

LEARN EMBEDDINGS $L(x)$

$L(x) = \tanh(x^T w + b) \in \mathbb{R}^m$

#LITERALS $= m$

↓

CONJUNCTIONS OVER L AND

#CONJUNCTIONS $= k$

↓

OUTPUT - NEURAL OR GATE

$$DNNF(x) = OR([AND_{c^1}(L(x)), AND_{c^2}(L(x)), \dots AND_{c^k}(L(x))])$$

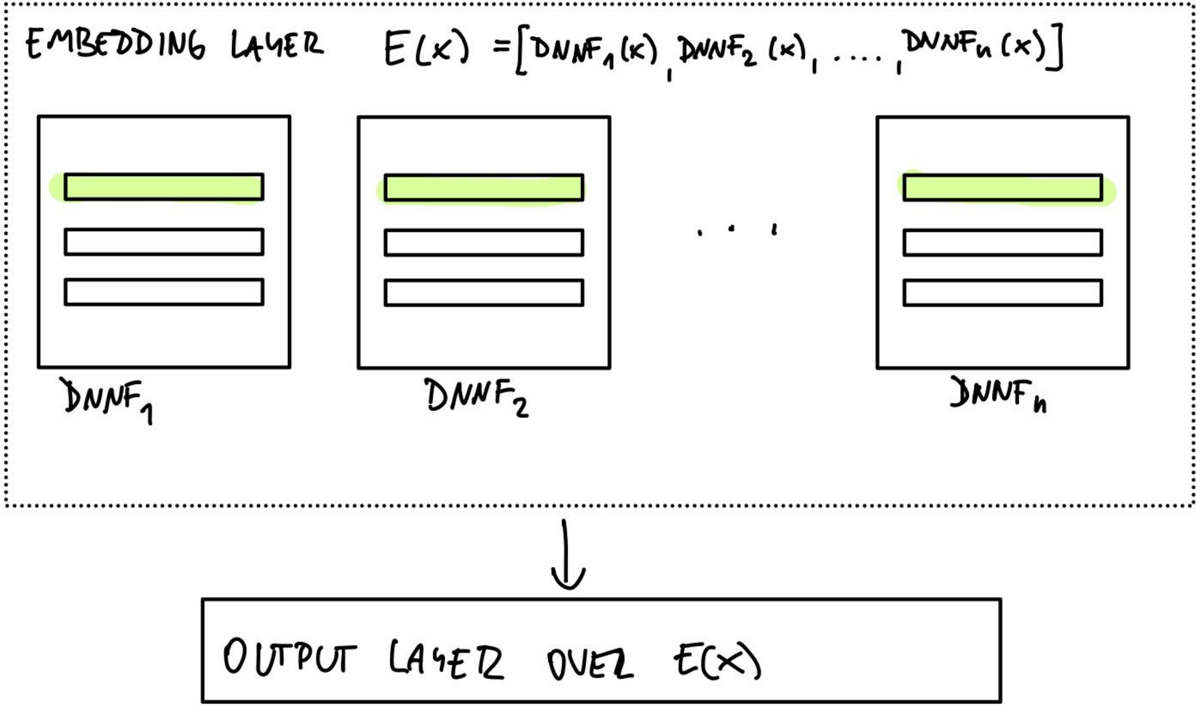
$c^1, c^2, \dots c^k$ FIXED MASKING VECTORS

↓

OUTPUT : HOW WELL x SATISFIES THE DNF FORMULA

DNF-Net

LEARNABLE



E.G. BINARY CLASSIFICATION $\text{DNF-Net}(x) = \sigma\left(\sum_{i=1}^{n} w_i \, DNNF_i(x) + b_i\right)$

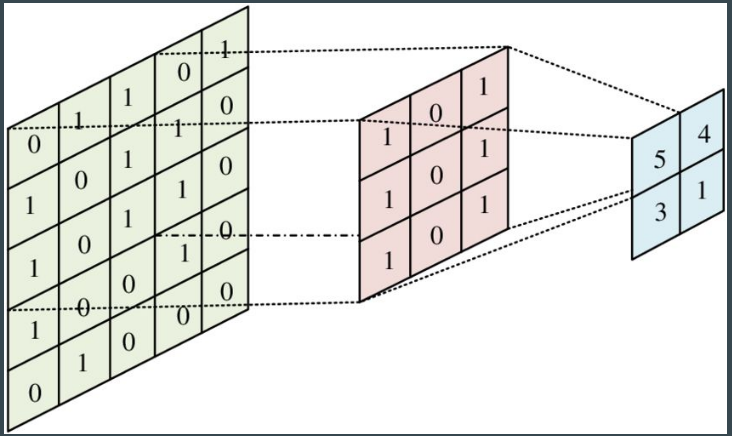
$\downarrow$

OUTPUT (E.G. [0,1])

1D-CNN

- kaggle multi-label classification, probability prediction
- CNN for feature extraction
- expect local correlation between features
- order of columns usually does not matter

- idea: re-order features to make them correlated
- expand dimensionality -> soft-ordering
- apply CNNs



Performance Comparison: Deep Models and XGBoost

Data Description, Training and Evaluation Metrics

- datasets from DNF-Net, NODE, TabNet papers + 2 Kaggle
- 11 classification and regression tasks
- cross-entropy loss, RMSE
- testing significance of performance differences

Dataset	Features	Classes	Samples	Source	Paper
Gesture Phase	32	5	9.8k	OpenML	DNF-Net
Gas Concentrations	129	6	13.9k	OpenML	DNF-Net
Eye Movements	26	3	10.9k	OpenML	DNF-Net
Epsilon	2000	2	500k	PASCAL Challenge 2008	NODE
YearPrediction	90	1	515k	Million Song Dataset	NODE
Microsoft (MSLR)	136	5	964k	MSLR-WEB10K	NODE
Rossmann Store Sales	10	1	1018K	Kaggle	TabNet
Forest Cover Type	54	7	580k	Kaggle	TabNet
Higgs Boson	30	2	800k	Kaggle	TabNet
Shrutime	11	2	10k	Kaggle	New dataset
Blastchar	20	2	7k	Kaggle	New dataset

Table 1: Description of the tabular datasets

Requirements for Real-World Applications

- high accuracy
- efficient training and inference
- short optimization time

Testing Deep Models on Unseen Datasets

- datasets not included in original paper
- models perform worse than datasets original models
- no deep model consistently outperformed others
- XGBoost generally outperformed deep models (Eye, Gesture, MSLR, Shrutime, Blastchar)
- XGBoost significantly better than not original models in 8 of 11 cases
- ensemble of deep models and XGBoost usually outperformed other models

Model Name	Rossman	CoverType	Higgs	Gas	Eye	Gesture
XGBoost	490.18 ± 1.19	3.13 ± 0.09	21.62 ± 0.33	2.18 ± 0.20	**56.07**± 0.65	80.64 ± 0.80
NODE	488.59 ± 1.24	4.15 ± 0.13	21.19 ± 0.69	2.17 ± 0.18	68.35 ± 0.66	92.12 ± 0.82
DNF-Net	503.83 ± 1.41	3.96 ± 0.11	23.68 ± 0.83	**1.44** ± 0.09	68.38 ± 0.65	86.98 ± 0.74
TabNet	**485.12**± 1.93	3.01 ± 0.08	**21.14**± 0.20	1.92 ± 0.14	67.13 ± 0.69	96.42 ± 0.87
1D-CNN	493.81 ± 2.23	3.51 ± 0.13	22.33 ± 0.73	1.79 ± 0.19	67.9 ± 0.64	97.89 ± 0.82
Simple Ensemble	488.57 ± 2.14	3.19 ± 0.18	22.46 ± 0.38	2.36 ± 0.13	58.72 ± 0.67	89.45 ± 0.89
Deep Ensemble w/o XGBoost	489.94 ± 2.09	3.52 ± 0.10	22.41 ± 0.54	1.98 ± 0.13	69.28 ± 0.62	93.50 ± 0.75
Deep Ensemble w XGBoost	485.33 ± 1.29	**2.99** ± 0.08	22.34 ± 0.81	1.69 ± 0.10	59.43 ± 0.60	**78.93** ± 0.73
	TabNet			DNF-Net		

Model Name	YearPrediction	MSLR	Epsilon	Shrutime	Blastchar
XGBoost	77.98 ± 0.11	$55.43 \pm$2e-2	$11.12 \pm$3e-2	13.82 ± 0.19	20.39 ± 0.21
NODE	76.39 ± 0.13	$55.72 \pm$3e-2	**10.39**$\pm$1e-2	14.61 ± 0.10	21.40 ± 0.25
DNF-Net	81.21 ± 0.18	$56.83 \pm$3e-2	$12.23 \pm$4e-2	16.8 ± 0.09	27.91 ± 0.17
TabNet	83.19 ± 0.19	$56.04 \pm$1e-2	$11.92 \pm$3e-2	$14.94 \pm, 0.13$	23.72 ± 0.19
1D-CNN	78.94 ± 0.14	$55.97 \pm$4e-2	$11.08 \pm$6e-2	15.31 ± 0.16	24.68 ± 0.22
Simple Ensemble	78.01 ± 0.17	$55.46 \pm$4e-2	$11.07 \pm$4e-2	$13.61 \pm, 0.14$	21.18 ± 0.17
Deep Ensemble w/o XGBoost	78.99 ± 0.11	$55.59 \pm$3e-2	$10.95 \pm$1e-2	14.69 ± 0.11	24.25 ± 0.22
Deep Ensemble w XGBoost	**76.19** ± 0.21	**55.38**$\pm$1e-2	$11.18 \pm$1e-2	**13.10**± 0.15	**20.18**± 0.16
	NODE			New datasets	

Direct Comparison

- calculating relative performance to the best model
- average relative performance per model on unseen datasets
- best: Ensemble w XGBoost
- best single model: XGBoost
- deep models: last 4 places

Name	Average Relative Performance (%)
XGBoost	3.34
NODE	14.21
DNF-Net	11.96
TabNet	10.51
1D-CNN	7.56
Simple Ensemble	3.15
Deep Ensemble w/o XGBoost	6.91
Deep Ensemble w XGBoost	**2.32**

Why are the Deep Models Worse?

- selection bias
- hyperparameters optimization (more extensive search could lead to better performance)

Do we need both?

- ensemble of deep models with XGBoost were the best
- which component of ensemble is mandatory?
- do we need deep models?
- (XGBoost, SVM, CatBoost)
- (deep model ensemble)
- (deep models + XGBoost)

Comparison of Ensemble Models

- deep models ensemble did not perform well
- classical ensemble better than deep models
- however, deep models combined with XGBoost are significantly better than classical ensemble or XGBoost itself

Name	Average Relative Performance (%)
XGBoost	3.34
NODE	14.21
DNF-Net	11.96
TabNet	10.51
1D-CNN	7.56
Simple Ensemble	3.15
Deep Ensemble w/o XGBoost	6.91
Deep Ensemble w XGBoost	**2.32**

Performance & Computation Demand Trade-off

- \+ performance
- \+ expensive computation
- Do we need all models in the ensemble?
- How to choose the subset?

- 3 models ≈ full ensemble
- significant differences between selection approaches

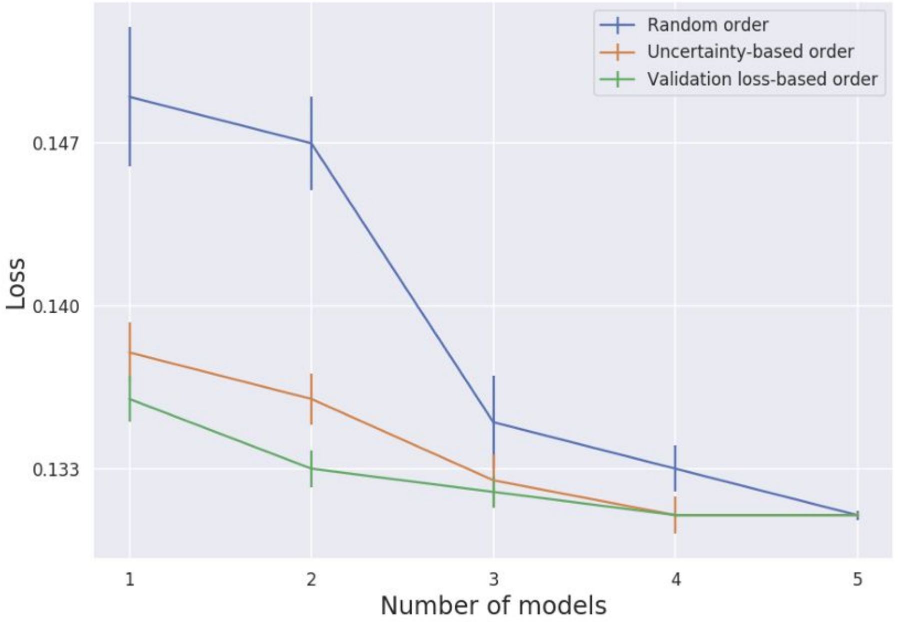
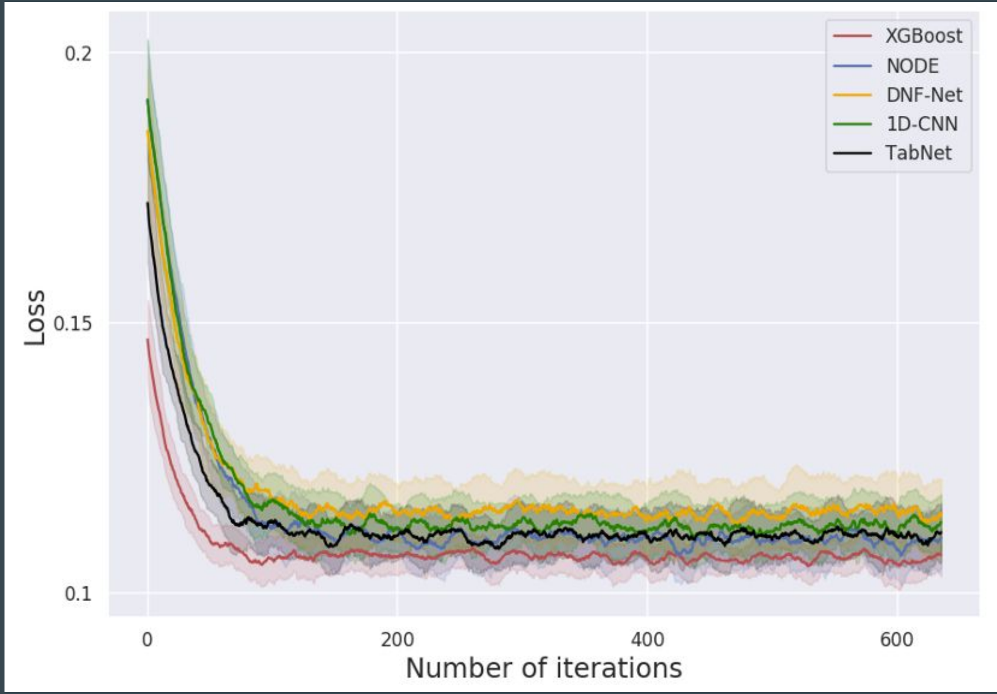


Figure 1: **The impact of selecting a subset of models in the ensemble.**

How difficult is the hyperparameter optimization?

- limited time for training in real applications
- FLOPS not suitable - different architectures
- runtime affected by software optimizations (XGBoost)
- proxy measure: number of optimization iterations
- XGBoost much faster than deep models
-



Summary

- Deep models often underperform on unseen datasets.
- XGBoost remains a strong baseline.

Summary

- Deep models often underperform on unseen datasets.
- XGBoost remains a strong baseline.
- Ensembles combining DL with XGBoost outperform individual models.

Summary

- Deep models often underperform on unseen datasets.
- XGBoost remains a strong baseline.
- Ensembles combining DL with XGBoost outperform individual models.
- DL models harder to optimize

Summary

- Deep models often underperform on unseen datasets.
- XGBoost remains a strong baseline.
- Ensembles combining DL with XGBoost outperform individual models.
- DL models harder to optimize

Limited time? -> XGBoost

Maximal performance? -> Add deep models to an ensemble

Grinsztajn et al.: Why do tree-based models still outperform deep learning on typical tabular data?

Random Forest

XGBoost

GBTree

SAINT

Resnet

FT_Transformer

MLP

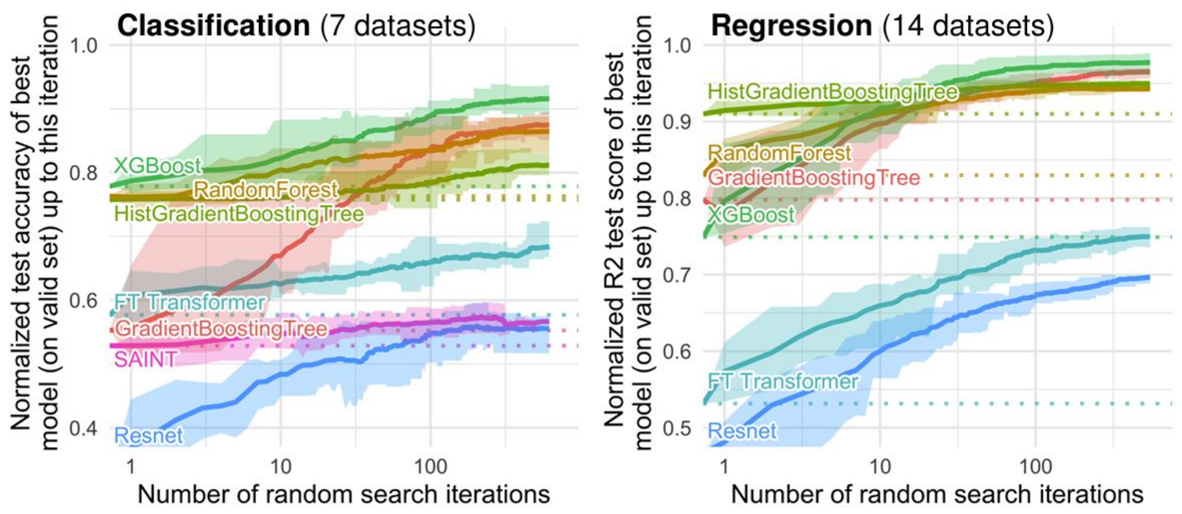


Figure 2: **Benchmark on medium-sized datasets, with both numerical and categorical features**. Dotted lines correspond to the score of the default hyperparameters, which is also the first random search iteration. Each value corresponds to the test score of the best model (on the validation set) after a specific number of random search iterations, averaged on 15 shuffles of the random search order. The ribbon corresponds to the minimum and maximum scores on these 15 shuffles.