XGBoost

Bagging vs boosting

	Bagging	Boosting
Similarities	Ensembles,combine weak learners of the same type	
Difference	Models are built independently of each other	Each next model influenced by the performance of the previous one
Types	Random forests	Gradient boosting
	● Fully grown decision trees ● Low bias, high variance	● Shallow trees ● High bias, low variance

Boosting Algorithms

- Adaboost
- Gradient Boosting
- XGBoost

Adaboost

1. Starts by building a short tree (stump)
2. Builds the next stump based on errors the previous stamp made
3. Each of the subsequent stumps is weighted
    a. The amount of say that this has on the final output is how good it compensated for the previous errors
4. Until it makes the number of stumps we ask for or it has the perfect fit

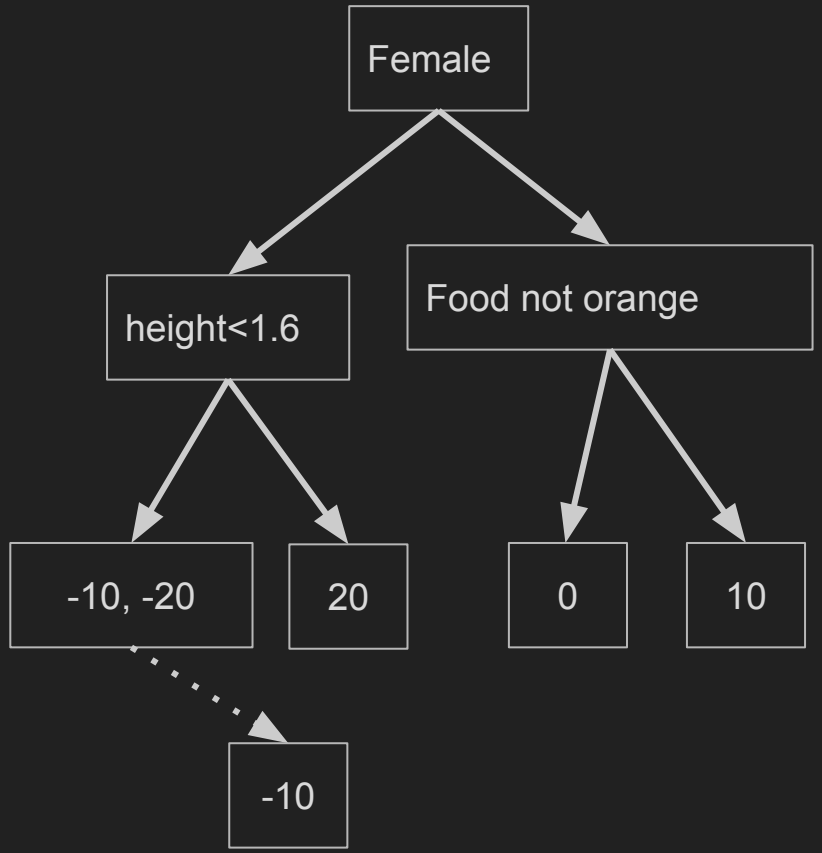
Gradient Boost

1. Starts by making just a leaf
2. Then GB builds a tree which like adaboost is based on the errors of previous leaf **but unlike adaboost this tree is larger than a stump**
3. GB boost scales the trees like adaboost, **but unlike adaboost it scales them all by the same amount**
4. Trees from here on predict the residuals
5. GB also multiplies the next trees with a learning rate to deal with the overfitting
6. We keep adding trees till we reach max size or no significant error improvement

Gradient Boosting

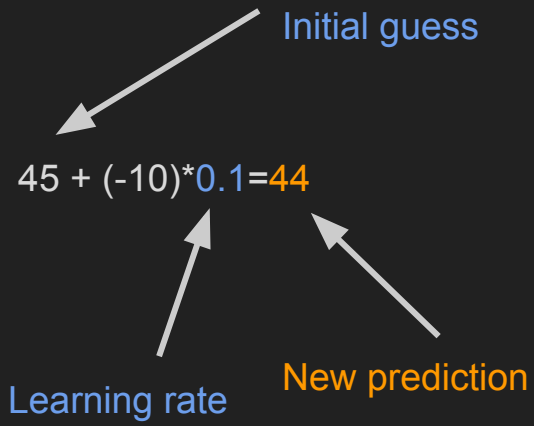
height	food	gender	weight	residuals
1.6	banana	male	55	10
1.6	orange	male	45	0
1.6	orange	female	65	20
1.5	orange	female	35	-10
1.4	orange	female	25	-20

Initial prediction
average=45



Gradient Boosting-0 Second iteration

height	food	gender	weight	residuals	residuals_2
1.6	banana	male	55	10	
1.6	orange	male	45	0	
1.6	orange	female	65	20	
1.5	orange	female	35	-10	-9
1.4	orange	female	25	-20	-19



Gradient boosting

- Input data $\left\{\left(x_i, y_i\right)\right\}_{i=1}^{n}$ and a loss function $L = (y_i, F(x))$
- Initialize a model with constant value $F_0 x = argmin_\gamma \sum_{i=1}^{n} L(y_i, \gamma)$, our first leaf
- For $m = 1 \; to \; M$:
  - $r_{im} = - \left[\frac{\partial L(y_i, F((x_i))}{\partial F((x_i)}\right]_{F(x) = F_{m-1}(x)} for \; i \; 1, .., n$
  - Fit a regression tree to r_{im} and create terminal regions for R_{jm}
  - For $j = 1 ... . j_m$ compute $\gamma_{jm} = argmin_\gamma \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i + \gamma))$
  - Update $F_m(x) = F_{m-1}(x) + \sum_{j=1}^{j_m} \gamma_m \mathrm{I}(x \in R_{jm})$
- Output $F_m(x)$

XGboost

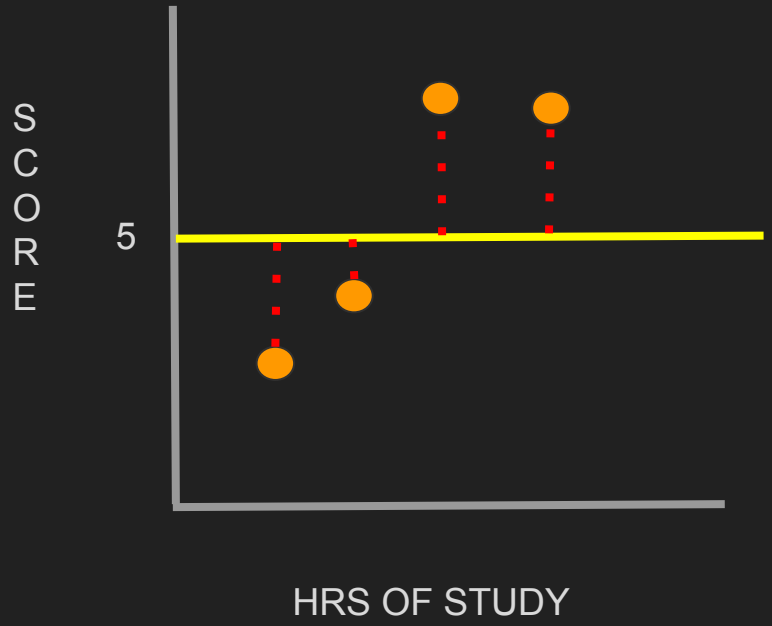
1. Gradient boost
2. Regularization
3. Regression tree
4. Approximate greedy algorithm
5. Weighted quantile sketch
6. Sparsity aware split finding
7. Parallel Learning
8. Cache aware process
9. Blocks for out of core computation

Regression tree

1. Start with our initial prediction and calculate residuals
2. All of them go in one leaf
3. We calculate a similarity score for the residuals (sum of residuals, squared/ (no. of residuals+l))
4. Now comes the questions if we can cluster residuals better by splitting them and if the leaves cluster similar residuals better than the root
    a. For this we calculate the gain (Leaf l -leaf R- root)
    b. We want the gain value maximized
5. We compare the gain at lowest leaf of the tree with a user defined tree complexity (*gamma*)
    a. If the difference is positive we don't prune otherwise we are done
    b. The higher the lamda the more pruning by shrinking similarity scores
6. The only difference between regression and classification is the loss functions

XGboost

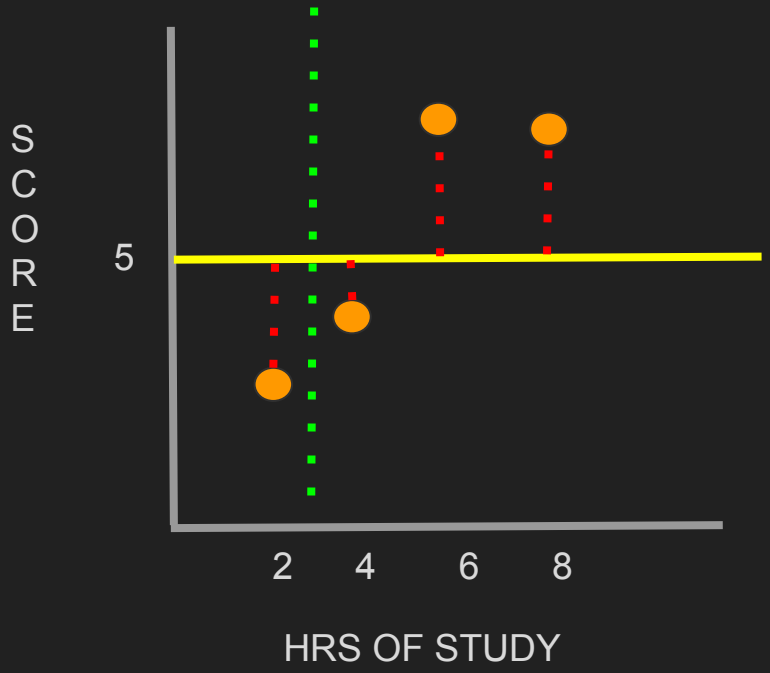
3 , 1, -4, -3 Similarity 4

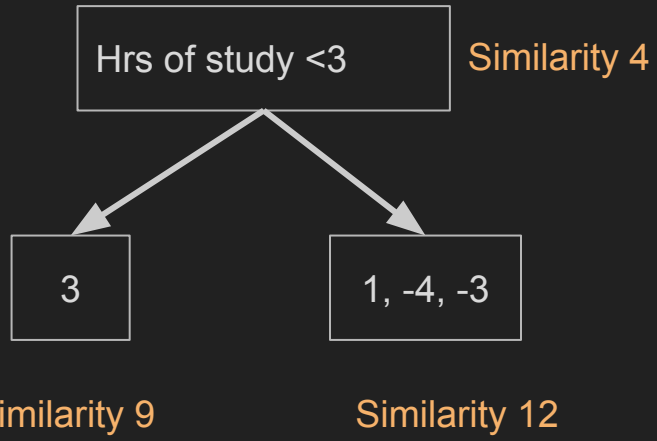


Similarity score =(3+1-4-3) ^ 2 /4 = 2.25

Can we cluster residuals better ???

XGboost

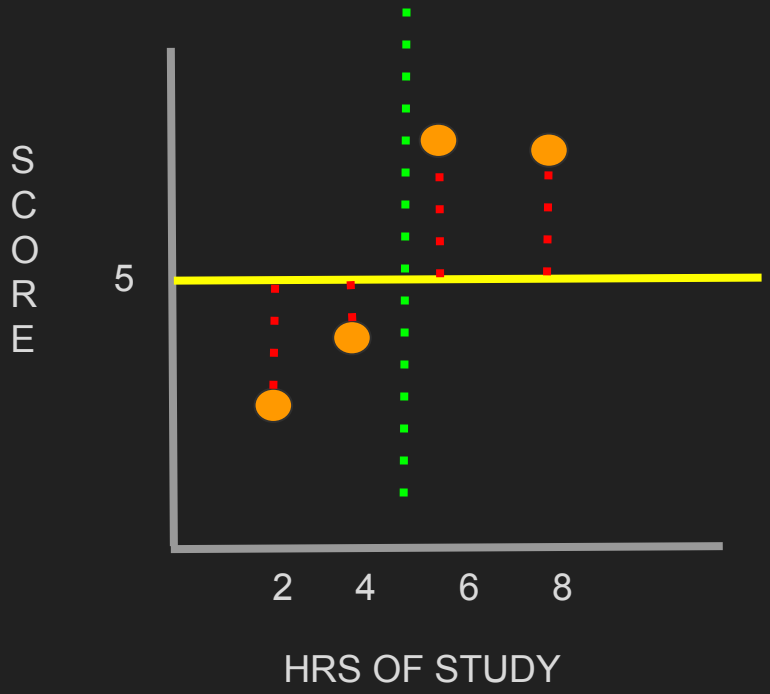


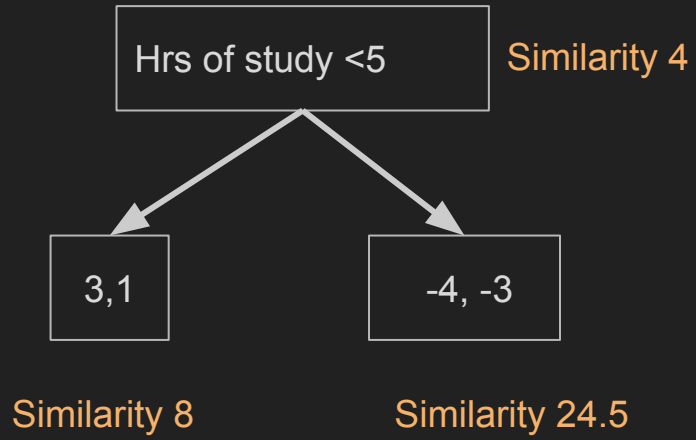


How much better the leaves cluster residuals???

Gain = Left + Right - Node = 17

XGboost





Gain = Left + Right - Node = 28.5

Approximate greedy algorithm

- Instead of testing all possible thresholds based on specific values we test quantiles
- We need to find the threshold with the biggest gain
- By default the approximate greedy algorithm uses about 33 quantiles

Weighted quantile sketch

- Large dataset: split amongst different cores or threads
- Sketch algorithm combines the values to make an approximate histogram which is then used by the greedy algorithm to calculate quantiles
- Weighted quantiles?
  - Normally, in histograms u have the same set of observations within the quantile
  - Each observation has corresponding weight and the sum of weights is the same within quantiles
  - The weight for each observation is the second derivative of the loss function
  - But for regression they are one, which means that they are like normal quantiles

Cache aware process

- CPU has a small amount of cache memory
- XGBoost puts gradients and hessians into the cache so it can compute faster

Blocks for out-of-core

- When the dataset is too big for cache and main, some of it must be put into the hard drive
- Because reading and writing from the hard is super slow, xgboost tries to minimize this by compressing the data

Sources

- Based on **Statquest's videos**