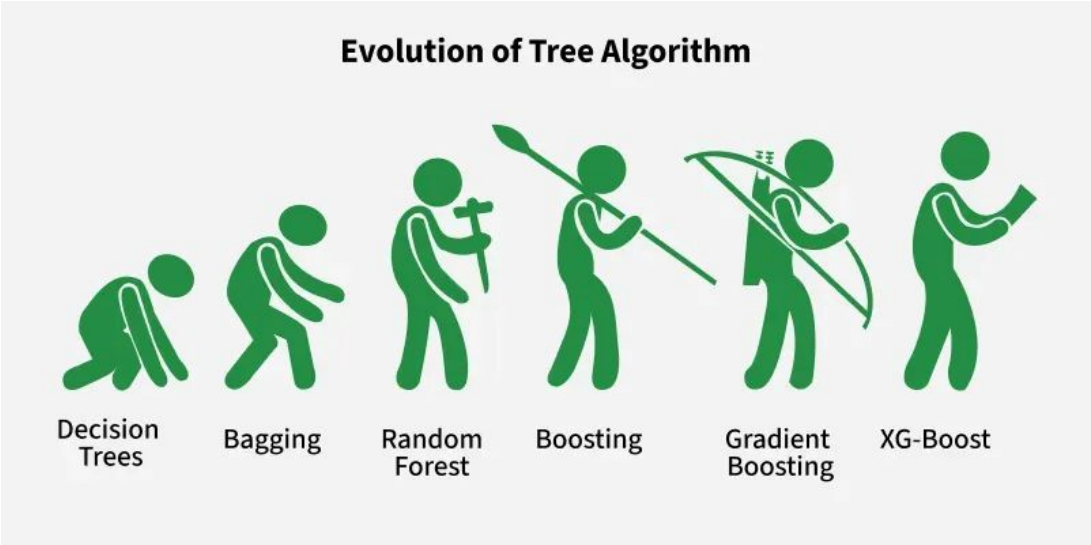
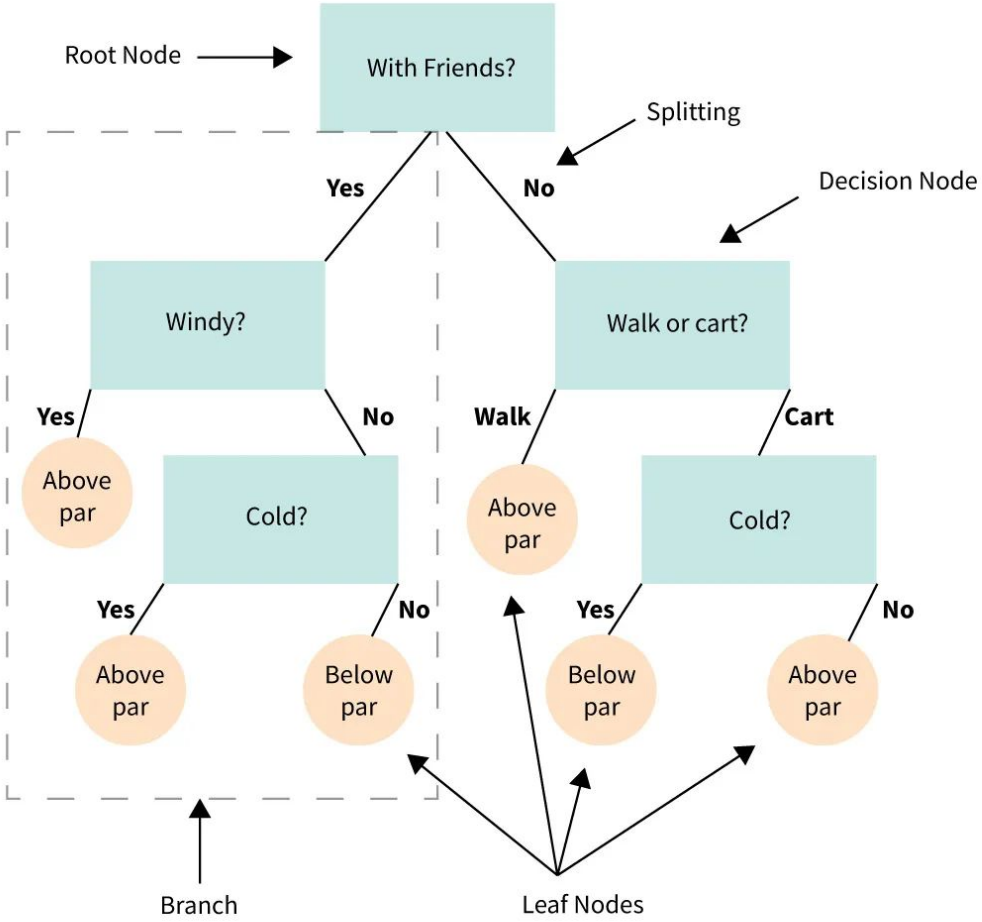
XGBoost


Evolution of Tree Algorithm
Decision Trees
Bagging
Random Forest
Boosting
Gradient Boosting
XG-Boost

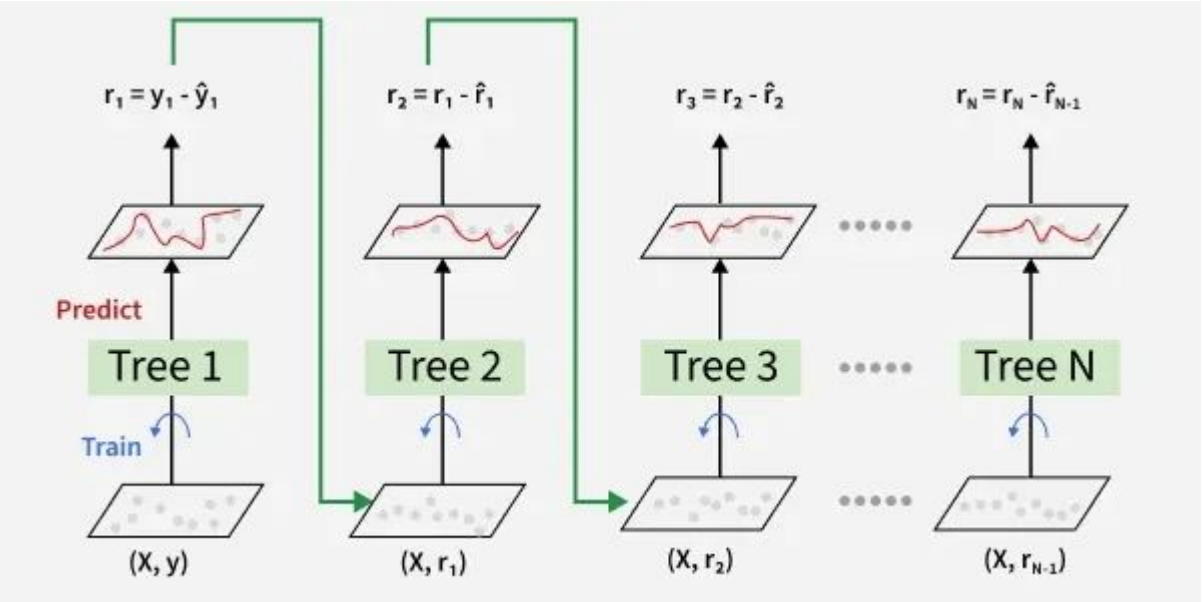
Decision trees


Root Node
With Friends?
Splitting
Yes
No
Decision Node
Windy?
Walk or cart?
Yes
No
Walk
Cart
Above par
Cold?
Above par
Cold?
Yes
No
Yes
No
Above par
Below par
Below par
Above par
Branch
Leaf Nodes

Single regression trees are often **unstable**; used for more powerful **ensembles**:

- **Random Forests:** trains trees on different random subsets of the data and features, and then **averages** their predictions.
- **Gradient Boosting (e.g., XGBoost):** builds trees, with each new tree trained to **correct** the errors made by previous ones.

Gradient boosting



Where $r_1, r_2, \ldots, r_N$ are the errors predicted by each tree.

$$y_{pred} = y_1 + \eta \cdot r_1 + \eta \cdot r_2 + \cdots + \eta \cdot r_N$$

XGBoost Model Objective

$$obj(\theta) = \sum_i^n l(y_i, \hat{y}_i) + \sum_{k=1}^K \Omega(f_k)$$

Where:

- $l(y_i, \hat{y}_i)$ is the loss function which computes the difference between the true value y_i and the predicted value $\hat{y}_i$,
- $\Omega(f_k)$

  is the regularization term which discourages overly complex trees.

Regularization

The regularization term $\Omega(f_t)$ simplify complex trees by penalizing the number of leaves in the tree and the size of the leaf. It is defined as:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

Where:

- T is the number of leaves in the tree
- γ is a regularization parameter that controls the complexity of the tree
- λ is a parameter that penalizes the squared weight of the leaves w_j

What Makes XGBoost "eXtreme"?

Extreme optimization

- Parallel, cache-aware
- Best splits: Evaluates every possible split for each feature at each level to minimizes loss

Extreme regularization

- L1, L2, Pruning
- Second-Order Derivatives (Newton Boosting)

Optimization

Parallel Computing: Unlike traditional boosting, which builds trees sequentially, XGBoost parallelizes the tree-building process.

Out-of-Core Computing: For massive datasets that don't fit in memory, XGBoost uses disk-mapping and cache-aware pre-fetching to handle data larger than RAM

Ex: Cache-Aware Prefetching

Rearrange calculations (e.g., tree construction, gradient stats) to make data always available when needed.

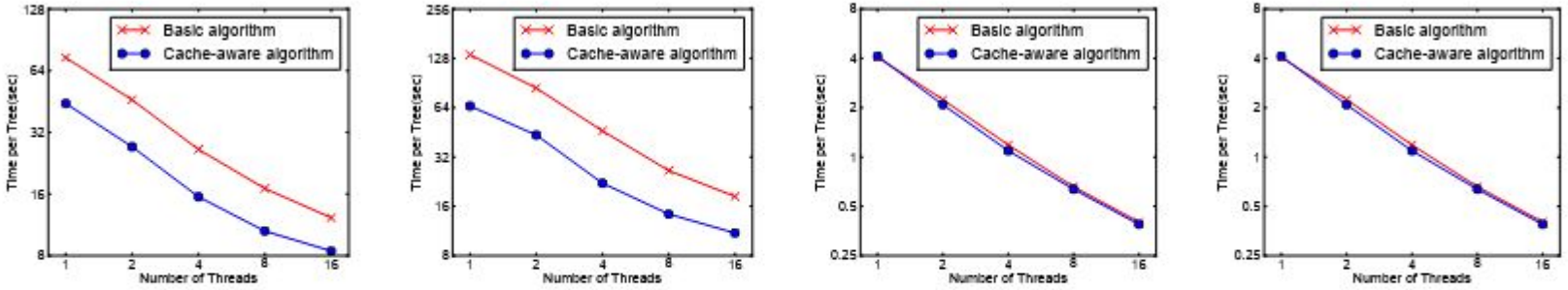


Figure 7: Impact of cache-aware prefetching in exact greedy algorithm. We find that the cache-miss effect impacts the performance on the large datasets (10 million instances). Using cache aware prefetching improves the performance by factor of two when the dataset is large.

L1 L2 Regularization

L1 Regularization

$$\text{Cost} = \sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2 + \lambda \sum_{j=0}^{M} |W_j|$$

L2 Regularization

$$\text{Cost} = \sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2 + \lambda \sum_{j=0}^{M} W_j^2$$

Loss function

Regularization Term

L1: is this step even worth taking?

$$\text{objective} \approx g\,w + \tfrac{1}{2}h\,w^2 + \underbrace{\lambda w^2}_{L2} + \underbrace{\alpha|w|}_{L1}$$

L2: Discourages extreme contributions

Before you take this step, ask: is the step itself too aggressive?

Newtonian Boosting

While traditional GBM uses only the gradient to minimize loss, XGBoost uses a second-order derivative in the loss function.

$$\text{GBM: } \ell \approx \ell_0 + g\Delta \quad \text{vs.} \quad \text{XGBoost: } \ell \approx \ell_0 + g\Delta + 1/2\, h\, \Delta^2$$

Gradient no longer on a straight line, rather fitting a local parabola!

Advantages

- Scalable for large datasets with **millions** of records
- Supports parallel processing and **GPU acceleration**
- Offers customizable parameters and regularization for **fine-tuning**
- Includes feature importance analysis (**sparsity** modeling)

Disadvantages

Blackbox: trades semantic interpretability for predictive power.

Overfit w/ small datasets: hessian aware loss can give false confidence in noisy gradients

Resource intensive: Large datasets + many features → RAM pressure and engineering overhead.

resources

Chen, T., & Guestrin, C. (2016).
XGBoost: A Scalable Tree Boosting System.
Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
arXiv:1603.02754.
https://arxiv.org/pdf/1603.02754

GeeksforGeeks.
XGBoost (eXtreme Gradient Boosting).
https://www.geeksforgeeks.org/machine-learning/xgboost/

ChatGPT (OpenAI).
Intuitive explanations of XGBoost concepts including second-order Taylor expansion, Hessian usage, L1/L2 regularization, and cache-aware prefetching.
Conversation with the author, February 10, 2026.