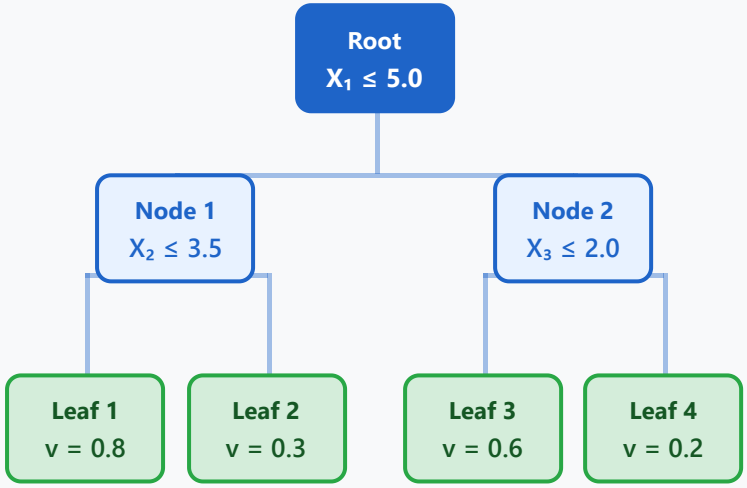
TreeSHAP: Optimized for Tree-Based Models

Polynomial time algorithm with exact Shapley values

Tree Traversal Example



Complexity Analysis

$$O(TLD^2) \text{ vs } O(2^M)$$

T: trees, L: leaves, D: max depth, M: features

Speed Advantage

10-100x faster than KernelSHAP for tree models

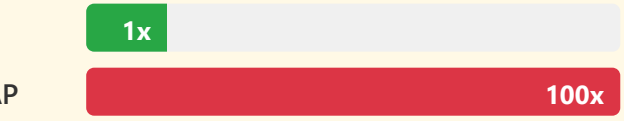
Exact Values

Computes exact Shapley values using tree structure

Model Support

Random Forest, XGBoost, LightGBM, CatBoost

Speed Comparison



TreeSHAP Computation Principles

1 Tree Structure Exploitation

TreeSHAP efficiently computes Shapley values by leveraging the hierarchical structure of tree-based models. It tracks decision paths at each node while reusing computations for feature combinations that share the same subtree.

$$\varphi_i = \Sigma_{S \subseteq M \setminus \{i\}} \, |S|!(M-|S|-1)! \, / \, M! \times [f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S)]$$

Original Shapley value definition

2 Recursive Computation

The algorithm recursively traverses the tree from root to leaves, tracking two pieces of information at each node:

🔑 Tracked Information

- **Coverage**: The proportion of possible feature combinations passing through the current node
- **Contribution**: The impact each feature has on the prediction value

$$\texttt{RECURSE}(j, m, p_z, p_o, v)$$

j: current node, m: unique path, p_z: zero probability, p_o: one probability, v: node value

3 Path Integration

When the same feature is used across multiple paths, the final Shapley value is calculated by taking a weighted average of each path's contribution. This process obtains exact values without explicitly enumerating all possible paths in the tree.

$$\varphi_i = \Sigma_{\text{paths}} \, w_{\text{path}} \times \Delta v_{i,\text{path}}$$

w_{path}: path weight, $\Delta v_{i,path}$: contribution of feature i along the path

4 Ensemble Integration

For ensemble models like Random Forest or Gradient Boosting, the Shapley values are computed independently for each tree and then averaged to generate the final explanation.

$$\varphi_i^{ensemble} = (1/T) \times \Sigma_{t=1}^{T} \varphi_i^{(t)}$$

T: number of trees, $\varphi_i^{(t)}$: Shapley value of feature i for the t-th tree

✓ TreeSHAP vs KernelSHAP

TreeSHAP

Computation: Direct use of tree structure

Complexity: $O(TLD^2)$

Accuracy: Exact Shapley values

Speed: Very fast (polynomial time)

Application: Tree-based models only

KernelSHAP

Computation: Sampling-based approximation

Complexity: $O(2^M)$

Accuracy: Approximate values

Speed: Slow (exponential time)

Application: Any model

💡 Key Insights

- TreeSHAP reduces exponential complexity to polynomial complexity by efficiently computing conditional expectations in trees

- It leverages dynamic programming principles to eliminate redundant computations and maximize memory efficiency
- Unlike model-agnostic KernelSHAP, it enables accurate and fast computation by exploiting the model's internal structure