

# Machine Learning Hierarchy & Application Strategy

---

## Level 1: The Fundamental Unit

### Decision Tree

- **Concept:** A single flowchart structure representing sequential decisions.
- **Characteristics:** Weak and unstable on its own; prone to overfitting.
- **Role:** The basic building block used to construct robust ensemble models.

## Level 2: The Strategy (Ensemble Learning)

*Goal: Combine multiple trees to create a powerful tool for regression or classification.*

### Strategy A: Bagging (Bootstrap Aggregating)

- **Logic:** Parallel execution.
- **Mechanism:** Trains  $N$  trees independently on random data subsets. Final result is an average.
- **Key Algorithm:** Random Forest.
- **Weakness for Physics:** Averaging "dilutes" hard constraints. If a level matches in energy but fails a spin veto, averaging might still assign a high probability (e.g., 90% Match + 10% Veto  $\approx 90\%$  Match).

### Strategy B: Boosting

- **Logic:** Sequential execution.
- **Mechanism:** Iterative correction. Tree  $N$  targets the errors of Tree  $N-1$ .
- **Key Algorithms:**
  - AdaBoost: Adjusts **sample weights** (focuses on hard-to-classify data points).
  - Gradient Boosting: Uses **gradient descent** to minimize error residuals (focuses on reducing loss).

## Level 3: Software Implementations (The Packages)

*Libraries implementing the Gradient Boosting algorithm.*

Package	NaN Handling	Growth Strategy	Best Data Scale	Verdict for Nuclear Data
Scikit-learn GradientBoosting	Fails (Crashes)	Level-wise	Small/Medium	Reject (Requires imputation, creating bias).
LightGBM	Native / Safe	Leaf-wise	Huge ( $>100k$ )	Reject (Risk of overfitting small data).
Scikit-learn HistGradientBoosting	Native / Safe	Leaf-wise	Medium/Large	Acceptable (Good, but less tunable than XGBoost).

Package	NaN Handling	Growth Strategy	Best Data Scale	Verdict for Nuclear Data
XGBoost	Native / Safe	Level-wise	Any (Excel for Small)	Best (Stable, robust regularization).
CatBoost	Native / Safe	Oblivious Trees	Categorical-Heavy	Specialized (Overkill for numerical energy data).

- **Leaf-wise (LightGBM):** Grows deep branches quickly. Great for massive data, but overfits noise in small nuclear datasets.
- **Level-wise (XGBoost):** Grows balanced trees. More stable and conservative for small datasets with experimental uncertainties.

## Level 4: Application to Nuclear Level Matching

### *Final Recommendation*

- **Concept:** Decision Trees.
- **Strategy: Boosting**
  - *Reasoning:* Boosting handles **physics constraints** (e.g., Spin/Parity vetoes) effectively by applying strong negative corrections to invalid matches. Bagging (Random Forest) tends to "average out" these critical vetoes.
- **Algorithm:** Gradient Boosting.
- **Recommended Package: XGBoost**
- **Why it is the Best:**
  1. **Sparsity Awareness:** Nuclear data is full of missing values (unknown \$J^\pi\$). XGBoost handles **NaN** natively by learning the optimal "default direction" for missing data, rather than requiring dangerous guesses (imputation).
  2. **Stability on Small Data:** Unlike LightGBM, XGBoost's **level-wise growth** and advanced **regularization** (\$L\_1\$/\$L\_2\$) prevent the model from "memorizing" experimental noise in small datasets (\$N < 500\$).
  3. **Physics Compliance:** The Boosting strategy effectively enforces hard vetoes (e.g., Selection Rules) better than Bagging methods.