Master Degree in Artificial Intelligence

genTree: A Different Approach to evtree Using Cython

2025 Statistical Learning

Gianni Moretti

General Pseudocode

1. Initialize a population of trees.
2. Evaluate each individual (fitness function).
3. Repeat until termination condition is met:
   (a) Select parents.
   (b) Apply variation operators (mutation, crossover).
   (c) Evaluate new solutions.
   (d) Select survivors (deterministic crowding).

Initialization and Parent Selection

Initialization

- Each individual: a tree with one root node and one valid random split.
- Uniform selection of variable and split point.
- Repeat until a valid split is found (node size threshold).

Parent Selection

- Each tree is selected at least once for variation.
- In crossover, a second parent is chosen at random.

Variation Operators

Overview

- **Split**: transforms a terminal node into an internal node.
- **Prune**: collapses an internal node with two leaves back to a terminal node.
- **Major Mutation**: changes the variable and/or split point of an internal node.
- **Minor Mutation**: slightly adjusts only the split point.
- **Crossover**: swaps subtrees between two trees.

Variation Operators

Split and Prune

Split:

- Select a random terminal node.
- Assign a valid split (variable + cutpoint).
- Repeat up to P attempts if invalid.

Prune:

- Select an internal node with two leaf children.
- Collapse it to a leaf, removing the split criterion.

Variation Operators

Mutation

Major Mutation:

- Select a random internal node.
- With 50% probability change the split variable or only the cutpoint.
- If invalid, revert and retry up to 3 times.

Minor Mutation:

- Slightly modify the cutpoint (10% of range or adjacent category).
- Preserve the same topological structure.

Variation Operators

Crossover

- Select two trees and one internal node in each at random.
- Swap the subtrees rooted at those nodes.
- Discard any resulting invalid nodes.

Fitness Evaluation

Classification:

$$\text{loss} = 2N \cdot MC(Y, f(X, \theta)), \quad \text{comp} = \alpha M \log N$$

Regression:

$$\text{loss} = N \log MSE(Y, f(X, \theta)), \quad \text{comp} = \alpha 4(M + 1) \log N$$

Where M is the number of leaves, N is the number of samples, and α controls complexity penalty. Minimize loss $+$ comp (accuracy vs. complexity trade-off).

Survivor Selection and Termination

Survivor Selection

- Deterministic crowding: each parent competes with its most similar offspring.
- Maintains population diversity.

Termination Criteria

- Convergence: top 5% tree quality stable for 100 iterations.
- Minimum of 1000 iterations.
- Otherwise stop after a specified number of iterations.

genTree

Why?

*"A common strategy is the **($\mu + \lambda$) selection**, where μ survivors for the next generation are selected from the union of μ parents and λ offsprings.*

*An alternative approach is the **(μ, λ) strategy** where μ survivors for the next generation are selected from λ offsprings.*

*Our algorithm uses a **deterministic crowding approach**, where each parent solution competes with its most similar offspring for a place in the population."*

General Pseudocode (genTree)

1. Initialize a population of random trees.
2. Evaluate each individual using a loss function (accuracy + complexity).
3. Repeat for a fixed number of generations:
   (a) Select parents based on fitness (probabilistic selection).
   (b) Apply variation operators (crossover, mutation).
   (c) Evaluate new solutions.
   (d) Select survivors (elitism + probabilistic replacement).

Initialization and Parent Selection

Initialization

- Each individual: a randomly generated tree (recursive random splits).
- Splits chosen randomly among features and valid thresholds.
- Trees must respect max depth and min samples per leaf.

Variation Operators

Identical except for the node selection:

- **Split Random Leaf**: transforms a random leaf into an internal node with a valid split.
- **Prune Random Leaf**: collapses an internal node (with leaf children) into a leaf.
- **Major Split**: changes the split variable and/or threshold of an internal node.
- **Minor Split**: slightly adjusts only the split threshold.

Variation Operators

Additional methods:

- **Prune Random Node**: collapses an internal node into a leaf.
- **Change Root Split**: changes the split at the root node.

Crossover

- Select two trees and a random internal node in each.
- Swap the subtrees of the second tree in the node of the first.
- Fix tree integrity (prune invalid splits if necessary).

Survivor Selection and Termination

Survivor Selection

- Elitism: best individuals always survive.
- Remaining population filled by offspring (crossover + mutation).
- Selection probabilities based on fitness.

Termination Criteria

- Fixed number of generations.
- (Optionally) convergence of best fitness.

genTree Parameters

- **max_depth**: maximum depth of the trees.
- **min_samples_leaf**: minimum number of samples required in a leaf.
- **is_regression**: boolean, True for regression, False for classification.
- **expand_prob**: probability to expand a node.
- **pop_size**: number of individuals in the population.
- **n_generations**: number of evolutionary generations.
- **alpha**: Complexity penalty coefficient in the loss function (default: 0.25).
- **importance**: Exponent for fitness-based parent selection (default: 1).
- **mutation_prob**: Probability of mutation per offspring (default: 0.1).
- **best_sel**: Fraction of best individuals preserved by elitism (default: 0.1).

The `_fit` Method

1. Set the complexity penalty parameter α and number of classes.
2. Create an initial random population of trees.
3. For each generation:
   (a) Evaluate the loss (fitness) of each tree.
   (b) Sort the population by fitness.
   (c) Preserve the top fraction (`best_sel`) as elites.
   (d) Fill the rest of the population by:
      - Selecting parents with probability inversely proportional to loss.
      - Applying crossover to generate offspring.
      - Optionally mutating offspring (with probability `mutation_prob`).
   (e) Update the population for the next generation.

Experiments

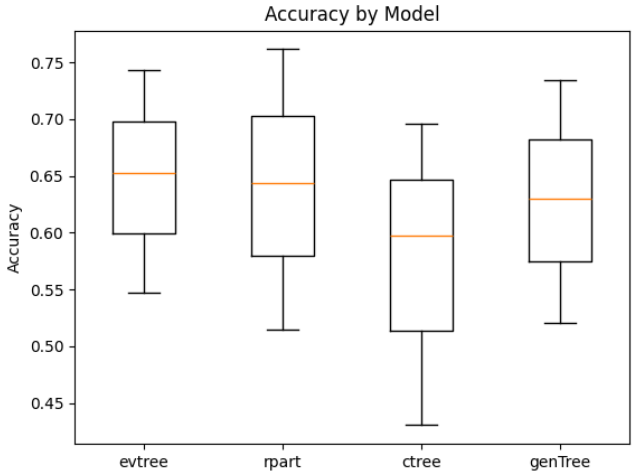
Glass Identification

Each row represents a glass sample described by 9 numerical variables (percentages of chemical elements such as Na, Mg, Al, etc.) and a categorical variable (‘Type‘) indicating the glass type (e.g., window, container, etc.).

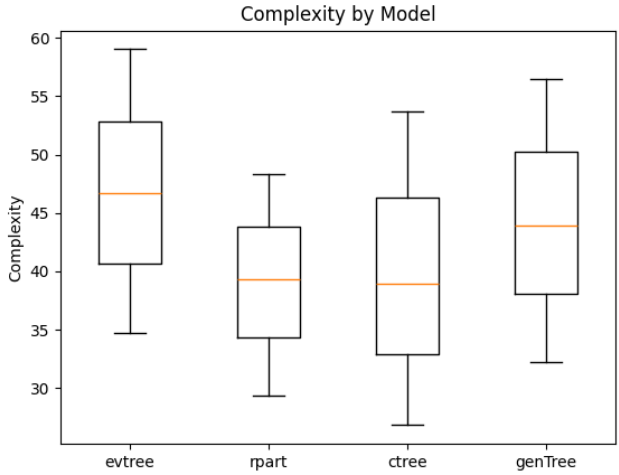
Objective: Predict the glass type based on its chemical composition.

- 100 boostrap run
- Accuracy OOB
- complexity $= \alpha M \log N, \alpha = 1$

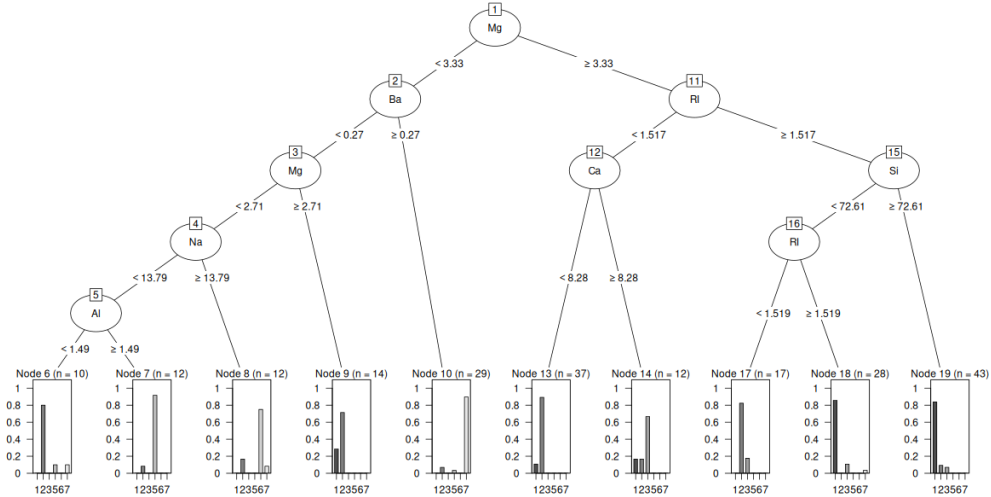
Glass Identification - Accuracy



Glass Identification - Complexity



Glass Identification - evTree Graph



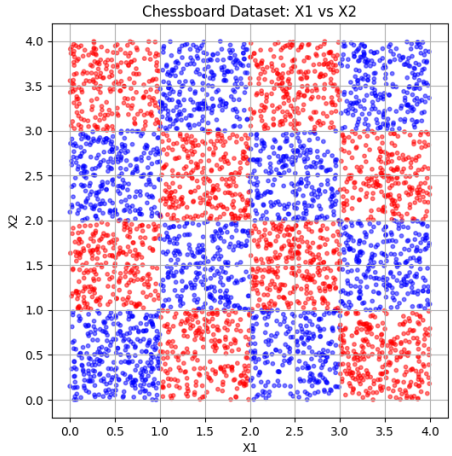
Glass Identification - genTree Graph

Chessboard

The dataset consists of 4,000 samples with 8 features (X1–X8). The class label is assigned in a 4x4 chessboard pattern based on the integer parts of X1 and X2, while X3–X8 are random noise features.

- 2000 Point Train set, 2000 point Test set
- Accuracy OOB
- complexity $= \alpha M \log N, \alpha = 1$

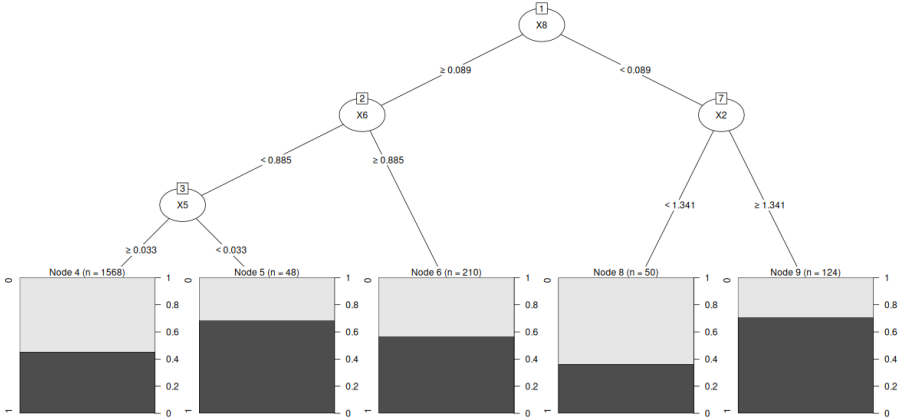
Chessboard



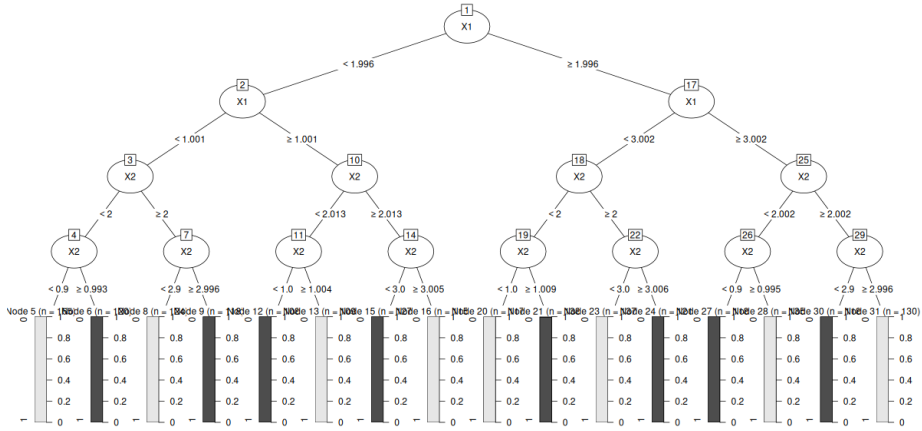
Chessboard - result

Model	Set	Accuracy	Complexity	Evaluation
evtree	Train	0.9995	121.614439	123.6144
evtree	Test	0.9940	121.614439	145.6144
rpart	Train	0.5655	38.004512	1776.0045
rpart	Test	0.4885	38.004512	2084.0045
ctree	Train	0.5165	7.600902	1941.6009
ctree	Test	0.4925	7.600902	2037.6009
gentree	Train	0.6490	68.408122	1472.4081
gentree	Test	0.6140	68.408122	—

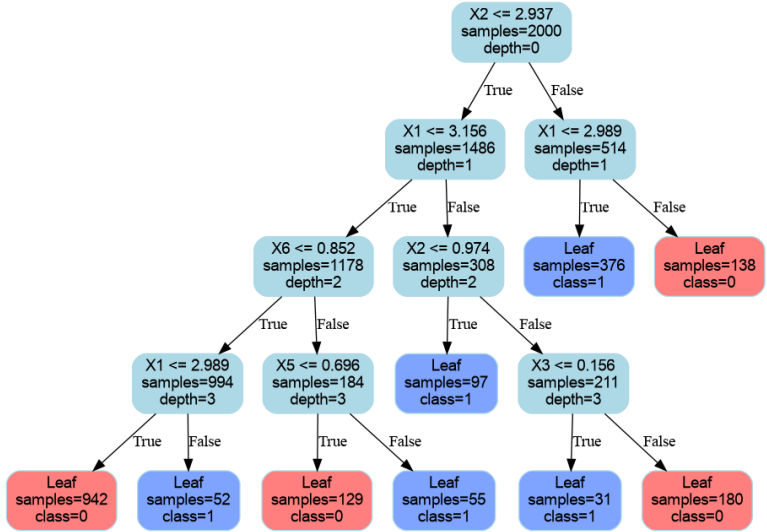
Chessboard - rpart Graph



Chessboard - evtree Graph



Chessboard - genTree Graph



Conclusions

Conclusions - Summary

- **Performance:** genTree is significantly slower than the other methods.
- **Accuracy:** When allowed to run as long as needed, it finds optimal solutions that outperform rpart and ctree, and in most cases also evtree (albeit at a higher cost).
- **Complexity:** In general, it tends to find solutions as complex as those of rpart and ctree, but much more complex than those found by evtree.
- **Initialization:** The initial population has a strong influence on the final result.

Conclusions - Future Work

- **Initialization:** Instead of starting with complete trees, start from the root only, as done in evtree.
- **Performance:** Parallelizing the generation of the new population could help reduce computation time.
- **Tuning:** Investigate how different parameters influence the discovery of optimal solutions.

References

- *evtree:* Evolutionary Learning of Globally Optimal Classification and Regression Trees in R
- *ChatGPT* - visuals & tips
- *Github Copilot* - Microsoft

Thanks for your attention

Gianni Moretti