DATA 2060

CART

For Classification

Sibo, Cynthia, Shiyu, and Zhaocheng
Data Science Institute, Brown

https://github.com/cynthiayry/DATA2060-Machine-Learning-Algorithm-Project--CART

INTRODUCTION

The CART (Classification and Regression Tree) is a fundamental decision tree learning algorithm that can be used for both classification and regression predictive modeling problems. It was introduced by Leo Breiman and his colleagues in 1984.

01

Math

Behind CART Classification

Representation

The model is represented as a **binary tree** T. Every split divides a node into exactly two children.

- **Nodes**: partition the data based on a splitting rule $x_j \leq t$ (next slides)
- **Leaves**: assign a class label based on the majority vote of samples falling into that region. (pick the class with the largest count)

For an input vector x, the prediction function f(x) traverses the tree from root to leaf:

$$f(\boldsymbol{x}) = \operatorname{argmax}_k \sum_{i \in R_{\text{leaf}(\boldsymbol{x})}} \mathbb{1}(y_i = k)$$

where:

- $R_{\text{leaf}(\mathbf{x})}$ is the set of indices of training points that fall into the same leaf as x.
- k indexes the class labels.

The tree predicts the majority class of the training samples in that leaf.

Loss Function

CART uses Gini Impurity to measure how mixed the classes are in a node.

Goal: find the split that produces the purest child node.

Gini impurity measures the probability of misclassifying a randomly chosen element from the set if it were randomly labeled according to the distribution of labels in the set.

For a node m with observations from K classes, let p_{mk} be the proportion of training observations in node m that belong to class k:

$$p_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$$

The Gini Impurity $H(Q_m)$ for node m is defined as:

$$H(Q_m) = \sum_{k=1}^{K} p_{mk}(1 - p_{mk}) = 1 - \sum_{k=1}^{K} p_{mk}^2$$

- If all samples belong to a single class, p_{mk} is 1 for that class and 0 for others. $1 - 1^2 = 0$. Impurity is minimal (Pure).
- If samples are evenly distributed, impurity is maximal.

Optimizer

CART uses a **greedy approach** (recursive binary splitting). It does not attempt to find the optimal tree; instead, it looks for the best single split at the current step.

At each step, the algorithm considers splitting a node m using a feature j and a threshold t. This splits the data into two subsets:

1. $Q_{left} = \{x | x^{(j)} \leq t\}$
2. $Q_{right} = \{x | x^{(j)} > t\}$

t is the midpoint between two consecutive unique values of feature j:

$$t = \frac{x_{(i)} + x_{(i+1)}}{2}$$

The quality of the split is measured by the weighted impurity of the children. We seek to minimize the cost function : $J(j, t)$

$$J(j,t) = \frac{N_{left}}{N_{total}} H(Q_{left}) + \frac{N_{right}}{N_{total}} H(Q_{right})$$

Where:

- N_{total} is the number of samples in the parent node.
- $H(Q)$ is the Gini impurity defined before.

How CART Works

Steps:

1. Start with all data at the root.
2. Iterate through every feature j and its every possible unique value to decide the candidate threshold t.
3. Calculate the cost function J(j, t).
4. Find the best split (j, t) that minimizes J across all features and thresholds.
5. Recurse on the left and right children until a stopping criterion is met.

02
Algorithm

Algorithm: CART Tree Construction

Input: Dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$, stopping criteria
Output: Binary decision tree T

Function `BUILD_TREE`(D, *depth*):
  if *stopping_criterion*(D, *depth*) **then**
    return LEAF_NODE(majority_class(D))
  best_gain $\leftarrow 0$;
  best_split $\leftarrow$ None;
  for *each feature* $j \in \{1, \ldots, d\}$ **do**
    thresholds $\leftarrow$ unique_sorted_values($D[:, j]$);
    for *each threshold* $\tau \in$ *thresholds* **do**
      $D_{\text{left}} \leftarrow \{(x, y) \in D : x_j \leq \tau\}$;
      $D_{\text{right}} \leftarrow \{(x, y) \in D : x_j > \tau\}$;
      $gain \leftarrow \text{Gini}(D) - \frac{|D_{\text{left}}|}{|D|}\text{Gini}(D_{\text{left}}) - \frac{|D_{\text{right}}|}{|D|}\text{Gini}(D_{\text{right}})$;
      if $gain >$ *best_gain* **then**
        best_gain $\leftarrow$ *gain*;
        best_split $\leftarrow (j, \tau)$;
  if *best_gain* $= 0$ **then**
    return LEAF_NODE(majority_class(D))
  node $\leftarrow$ INTERNAL_NODE(*best_split*);
  node.left $\leftarrow$ `BUILD_TREE`(D_{left}, *depth* $+ 1$);
  node.right $\leftarrow$ `BUILD_TREE`(D_{right}, *depth* $+ 1$);
  return *node*;

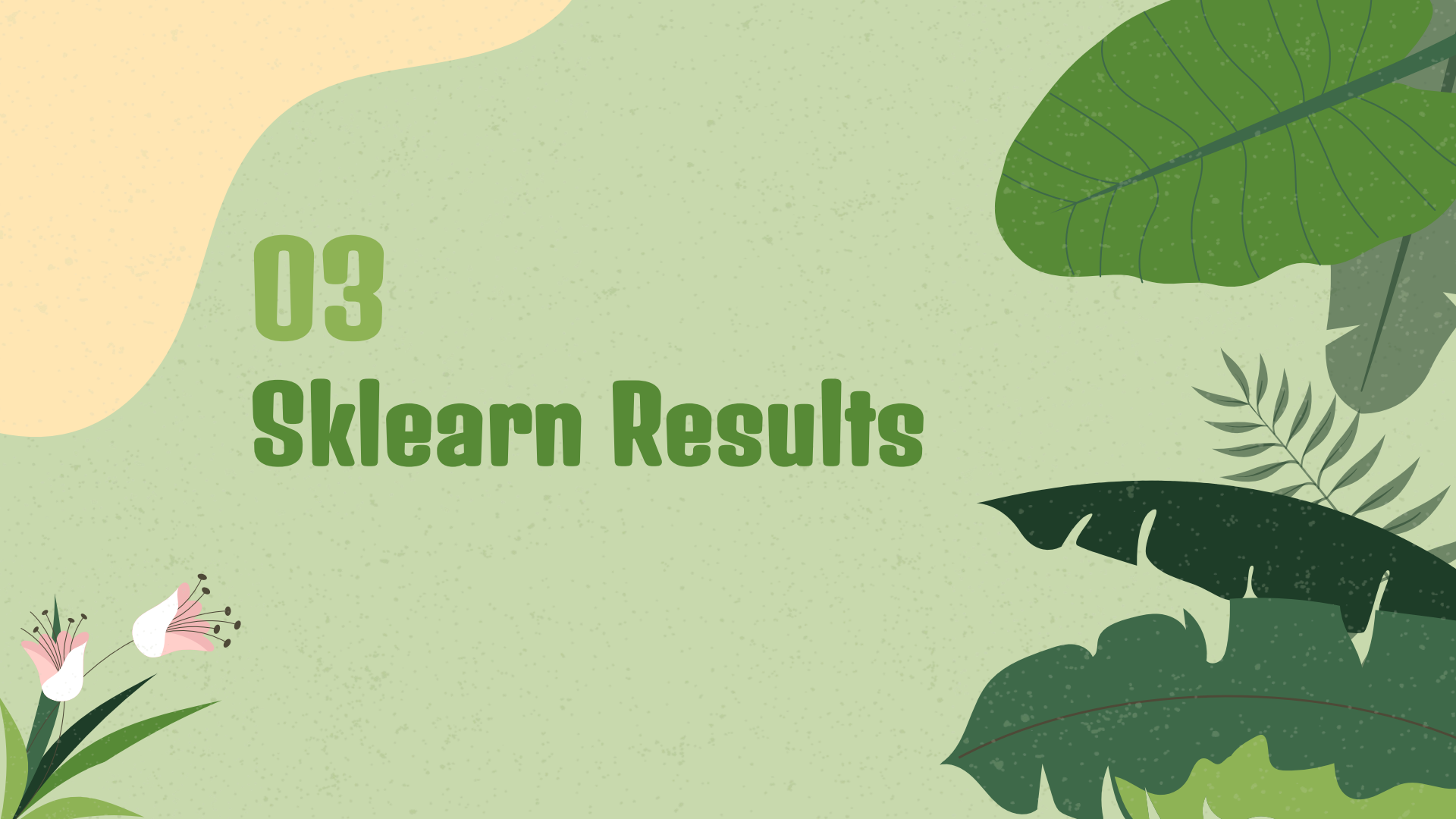
Stopping Criteria

- Maximum depth
- Minimum samples required to split
- Minimum samples at leaf node
- No improvement in Gini gain
- All samples belong to the same class

Complexity

- n_features = d, n_samples = N
- Training: O(d * NlogN * depth),
- Prediction: O(depth)

03
Sklearn Results



Validating Our CART Implementation

Dataset: Wisconsin Breast Cancer (569 rows, 30 columns) from UCI Datasets

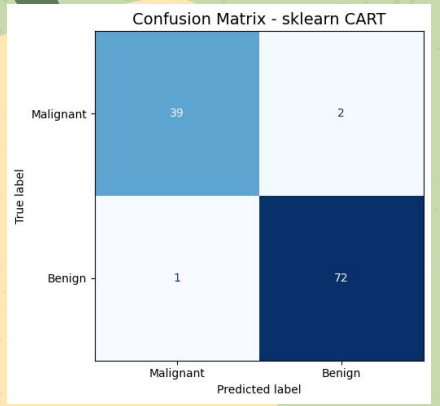
Train Data: 455; Test Data: 114

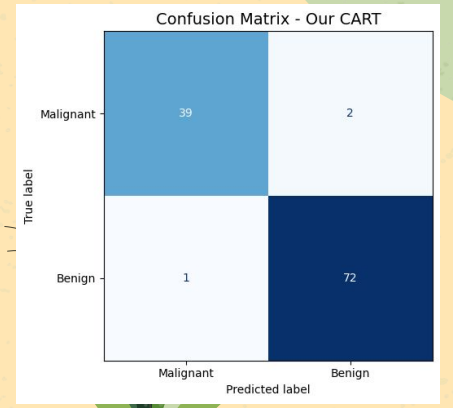
Binary classification: Malignant vs Benign

Hyperparameter Tuning: 3-fold stratified CV on training data

- max_depth: [2, **3,** 4, 5, 6, 7, 8]
- min_samples_split: [**2**, 5]
- min_samples_leaf: [**1,** 2, 5]
- BEST COMBO:
    - max_depth=3, split=2, leaf=1
    - Mean Validation Accuracy=0.94

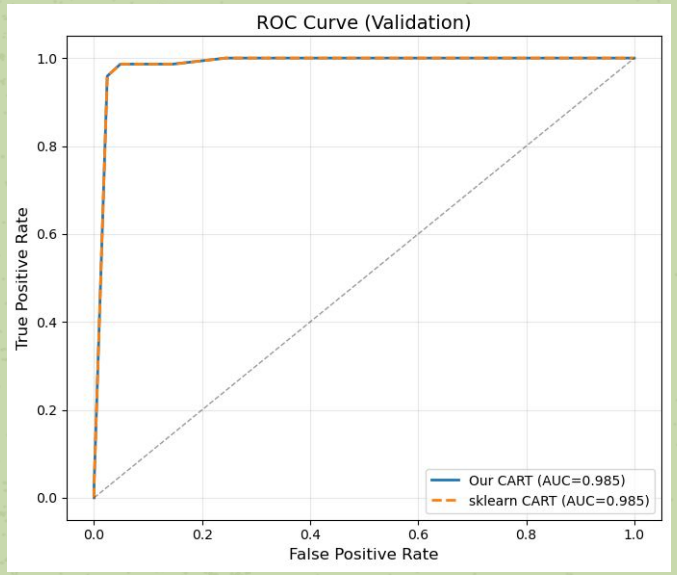
Implementation Comparison Results

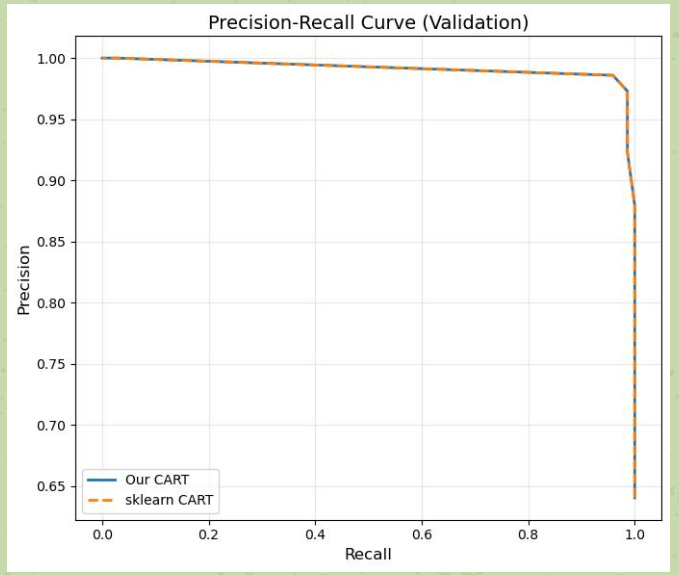




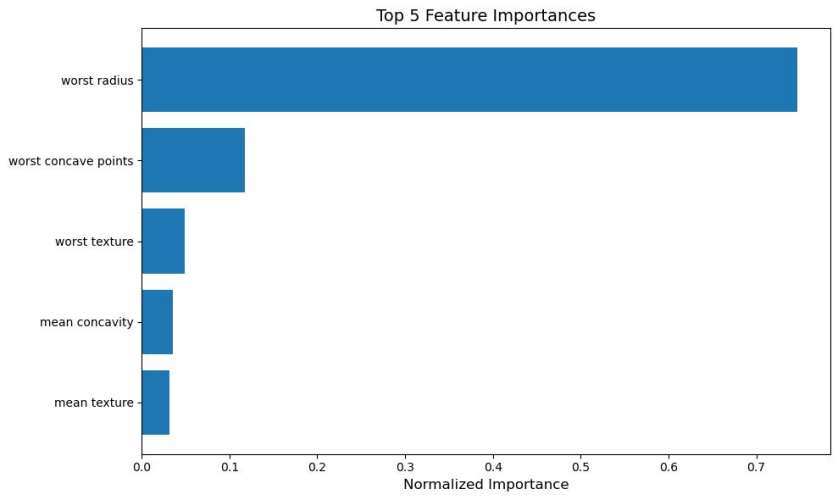
Metric	Our CART	Sklearn CART	Difference
Training Accuracy	98.20%	98.20%	0.000%
Test Accuracy	97.37%	97.37%	0.000%
Tree Depth	3	3	0
Number of Leaves	8	8	0

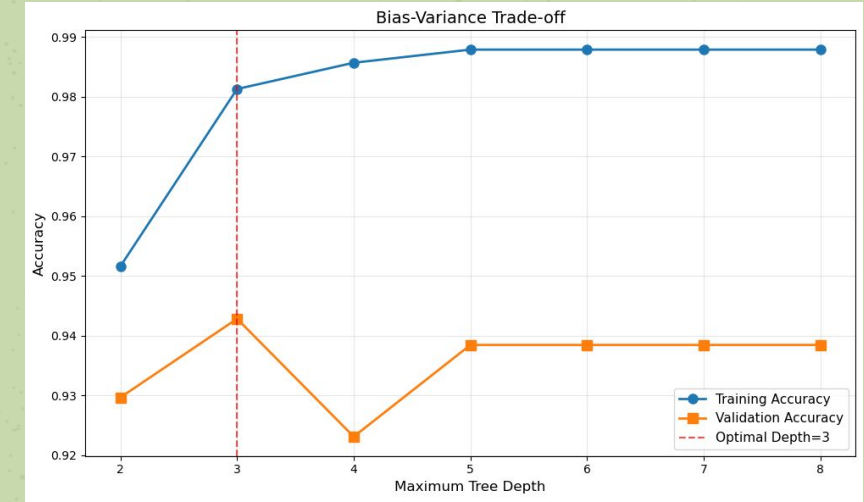
ROC & Precision-Recall Curves





Additional Model Insights





04

SUMMARY

What Did We Do?

WHAT IS CART

- Binary decision tree algorithm for classification
- Uses Gini impurity
- Greedy recursive partitioning to build tree structure

PROJECT GOAL

- Implemented CART algorithm
- Using only NumPy
- Validated against sklearn's DecisionTreeClassifier

OUR IMPLEMENTATION

- 21 methods
- All core functionality: fit, predict, predict_proba, score
- Feature importance and other visuals/analysis
- Hyperparameters tuning with CV

VALIDATION RESULTS

- 14 unit tests covering all methods
- Tested on Breast Cancer Wisconsin (114 samples, 30 features)
- Exact match with sklearn: 97.37% val accuracy

INTERESTING DISCOVERIES

Extreme Feature Concentration

- Top 3 features = ~90% of predictive power
- Single feature ("worst radius") > 70%

Binary Trees Surprisingly Expressive

- Only 8 leaves → 97.37% accuracy
- Expected larger tree

CV Reveals Non-Monotonic Behavior

- Optimal depth 3 (94% CV), but 4 drops to 92%
- Depths 5-8 plateau at 93.85% (CV finding nuanced patterns)

CHALLENGES OVERCOME

Threshold Selection

- Problem: How to efficiently find optimal split thresholds?
- Solution: Use midpoints between sorted unique values (Python's np.unique() sorts: O(N log N))

Exact sklearn Matching

- Problem: Reproduce sklearn's exact behavior
- Solution: Source code study for tie-breaking details ("first wins" tie-breaking)

Thank You!

Do you have any questions?

References

Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (1984) *Classification and Regression Trees*. Wadsworth, Belmont, CA.

Loh, W.Y. (2011) 'Classification and regression trees', *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1), pp. 14–23.

Pedregosa, F. *et al.* (2011) 'Scikit-learn: Machine learning in Python', *Journal of Machine Learning Research*, 12, pp. 2825–2830.

Dua, D. and Graff, C. (2019) 'UCI Machine Learning Repository'. University of California, Irvine, School of Information and Computer Science. Available at: https://archive.ics.uci.edu/ml (Accessed: 6 December 2025).

Street, W.N., Wolberg, W.H. and Mangasarian, O.L. (1993) 'Nuclear feature extraction for breast tumor diagnosis', *Biomedical Image Processing and Biomedical Visualization*, pp. 861–870.