

# CART (Classification and Regression Tree)

Adinda Putri - 13523071

**CART** merupakan decision tree yang digunakan untuk mempelajari hubungan dan pola dalam dataset dan membuat struktur berdasarkan nilai variabel dalam dataset tersebut. Saat proses konstruksi struktur tree, model memilih variabel dan nilai threshold yang paling baik untuk membagi data dan melakukan classification atau regression. Pada kali ini, akan dibahas hanya CART untuk classification. Kelebihan utama dari CART adalah model ini dapat mengukur pengaruh dan relevansi variabel dalam dataset. Hal ini dapat meningkatkan akurasi model dan bekerja baik untuk dataset nonlinear.

## Cara Kerja

1. Buat struktur **tree-like** yang terdiri dari **nodes** dan **branches**. Node merepresentasikan decision point, branch merepresentasikan possible outcomes dari berbagai decision tersebut. Terdapat juga **leaf nodes** yang berisi nilai untuk variabel target.
2. Hitung probabilitas kelas di node dengan rumus berikut

$$p_i = \frac{\text{jumlah data dengan kelas } i}{\text{total data di node}}$$

Dengan  $y_i$  merupakan label class ke- $i$

3. Untuk setiap fitur  $X_j$ , ambil semua nilai unik dan hitung kandidat thresholdnya (median)

$$\text{thresholds} = \frac{x_k + x_{k+1}}{2}$$

4. Hitung Gini impurity untuk tiap split

$$Gini = 1 - \sum_{i=1}^K p_i^2$$

dengan K adalah jumlah class unik pada node tersebut.

5. Hitung weighted Gini untuk split menjadi dua subset

$$Gini_{split} = \frac{N_{left}}{N} Gini(D_{left}) + \frac{N_{right}}{N} Gini(D_{right})$$

6. Pilih fitur + thresholdnya dengan **weighted Gini terkecil** sebagai best split

7. Buat node baru untuk menyimpan fitur terbaik, threshold terbaik dan probabilitas kelas di node tersebut

8. Split data menjadi dua subset dengan subset kiri nilai fitur  $\leq$  threshold dan subset kanan nilai fitur  $>$  threshold

9. Lakukan langkah 4-8 secara rekursif hingga max\_depth tercapai, min\_samples\_split bernilai False atau semua node pure (hanya satu kelas)

10. Leaf node menyimpan probabilitas kelas dan tidak bisa di-split lagi.

11. Dilakukan prediksi untuk tiap sampel.

## Perbandingan model dari scratch dan dari Scikit-Learn

The screenshot shows a Jupyter Notebook with two sections. The first section, titled 'CART: Classification', contains a code cell for 'K-Fold Cross-Validation CART' from scratch. The code calls `evaluate_model_classification(cart_classifier_pipe, X_train, y_train, kf)` and takes 25.1s to execute. The output shows performance metrics: Accuracy 74.30%, Precision 77.97%, Recall 74.20%, and F1-Score 75.77%. The second section, titled 'K-Fold Cross-Validation CART dari Scikit-Learn', contains a code cell that calls `evaluate_model_classification(cart_classifier_scikit_pipe, X_train, y_train, kf)` and takes 1.5s to execute. The output shows slightly lower performance metrics: Accuracy 73.75%, Precision 77.83%, Recall 73.51%, and F1-Score 75.29%.

```
evaluate_model_classification(cart_classifier_pipe, X_train, y_train, kf)
```

✓ 25.1s Python

--- Evaluating Model with K-Fold Cross Validation ---  
Rata-rata Akurasi: 74.30%  
Rata-rata Presisi: 77.97%  
Rata-rata Recall: 74.20%  
Rata-rata F1-Score: 75.77%

```
evaluate_model_classification(cart_classifier_scikit_pipe, X_train, y_train, kf)
```

✓ 1.5s Python

--- Evaluating Model with K-Fold Cross Validation ---  
Rata-rata Akurasi: 73.75%  
Rata-rata Presisi: 77.83%  
Rata-rata Recall: 73.51%  
Rata-rata F1-Score: 75.29%

Gambar 1. K-Fold Cross-Validation CART dari Scratch dan CART dari Sklearn

Sumber: Penulis

The screenshot shows a Jupyter Notebook with a code cell for 'Hold-Out Validation CART' from scratch. The code creates a `CARTClassifier` with `max_depth=100` and `min_samples_leaf=1`, fits it on `X_train` and `y_train`, predicts on `X_test`, and prints the classification report. It also creates a `DecisionTreeClassifier` with the same parameters. The code takes 2.7s to execute. Below the code, a classification report is displayed as a table.

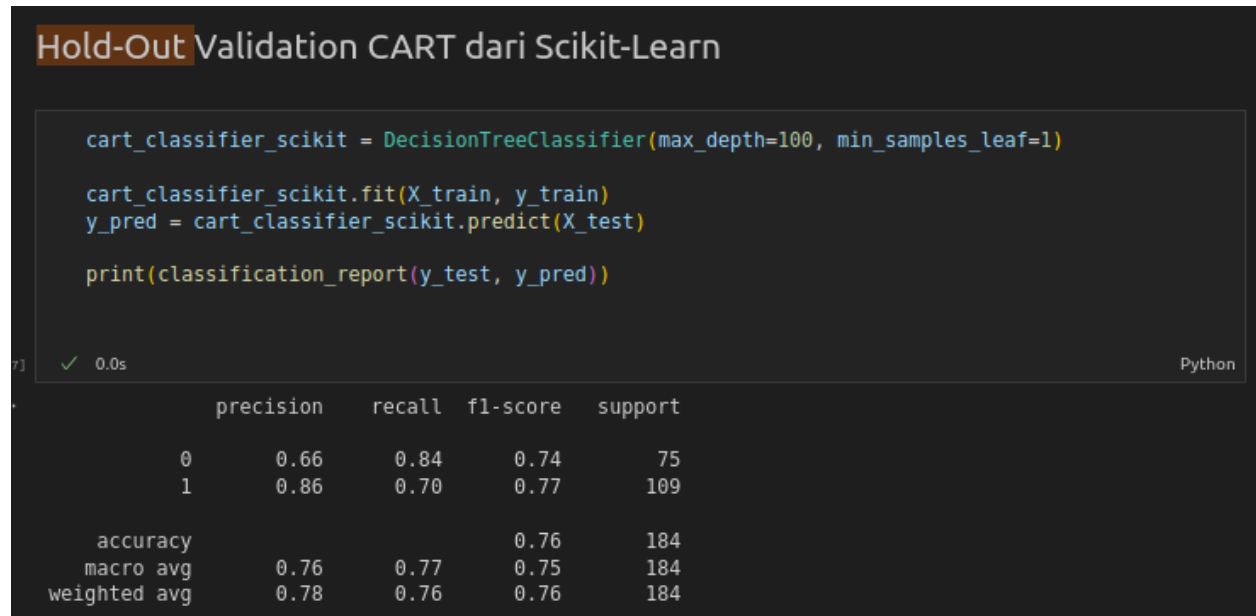
```
cart_classifier = CARTClassifier(max_depth=100, min_samples_leaf=1)
cart_classifier.fit(X_train, y_train)
y_pred = cart_classifier.predict(X_test)
print(classification_report(y_test, y_pred))
cart_classifier_scikit = DecisionTreeClassifier(max_depth=100, min_samples_leaf=1)
```

✓ 2.7s Python

	precision	recall	f1-score	support
0	0.68	0.85	0.76	75
1	0.88	0.72	0.79	109
accuracy			0.78	184
macro avg	0.78	0.79	0.78	184
weighted avg	0.80	0.78	0.78	184

Gambar 2. Hold-Out Validation CART dari Scratch

Sumber: Penulis



Gambar 3. Hold-Out Validation Cart dari Sklearn

Sumber: Penulis

Terdapat perbedaan hasil pada model yang mungkin terjadi karena implementasi algoritma yang berbeda. Meskipun keduanya tidak menggunakan pruning, dan mendapatkan nilai parameter yang sama. Akan tetapi, implementasi pencarian nilai thresholdnya mungkin berbeda. Selain itu algoritma CART versi Scikit-Learn sudah optimized.

### Ruang Improvement:

- Optimisasi pemilihan threshold dengan binary search, quicksort, mergesort dengan optimization.
- Eksperimen dengan exhaustive search untuk melakukan pencarian threshold lebih komprehensif
- Model dari scratch belum menggunakan pruning. Dapat diterapkan cost-complexity pruning untuk menghindari overfitting
- Optimisasi tree operation seperti memaksimalkan vectorized operation dan gunakan in-place operations dan data struktur yang efisien.

### Referensi:

[1] *Decision Trees in Machine Learning: CART and Advanced Trees*, Medium oleh Kaan Erden, 16 April 2023. [Daring]. Tersedia:

<https://medium.com/@kaanerdenn/decision-trees-in-machine-learning-cart-and-advanced-trees-8b3fe375e9f9>. [Diakses: 2 September 2025].

[2] *Decision Tree (CART) from scratch. Full tutorial*, Kaggle Notebook oleh Egazakharenko, 2025. [Daring]. Tersedia: <https://www.kaggle.com/code/egazakharenko/decision-tree-cart-from-scratch-full-tutorial>. [Diakses: 2 September 2025].