1. Jelaskan cara kerja dari algoritma tersebut!
Jawab:
CART adalah algoritma yang digunakan untuk membangun decision tree dalam klasifikasi dan regresi. Decision tree bekerja dengan cara mempartisi dataset ke dalam subset-subset yang lebih kecil berdasarkan nilai fitur tertentu. Partisi ini dilakukan secara berulang hingga dataset telah cukup terpisah, yang kemudian digunakan untuk memprediksi target variabel. Dalam konteks klasifikasi, tujuan dari algoritma CART adalah untuk menemukan pembagian optimal dari fitur-fitur dataset, sehingga pada setiap node di pohon, subset yang dihasilkan terdiri dari data yang homogen (semua sampel dalam node memiliki label yang sama). Cara kerjanya adalah sebagai berikut:

- Inisialisasi CART dimulai dari seluruh dataset pada root node.
- Memeriksa setiap fitur untuk menemukan titik pemisahan (split point) yang maksimal mengurangi impurity. Metrik impurity yang digunakan adalah Gini Index.

  $$\text{Gini Index} = 1 - \sum_{i=1}^{n} (p_i)^2$$

  p_i adalah probabilitas relatif dari kelas *i* pada node.

  Semakin kecil nilai Gini, semakin homogen pembagian node tersebut.
- Dataset dipisah berdasarkan aturan pemisahan (split rule) terbaik, menghasilkan dua subset. CART melanjutkan proses ini secara rekursif pada subset yang dihasilkan untuk membuat node baru.
- Pohon akan berhenti berkembang ketika salah satu kondisi berikut terpenuhi:
  - Impurity tidak bisa diturunkan lebih jauh (subset yang dihasilkan sudah homogen).
  - Jumlah sampel dalam node terlalu sedikit (misalnya, di bawah ambang batas yang telah ditentukan).
  - Maksimal kedalaman pohon tercapai.
- Untuk memprediksi sampel baru, algoritma CART melewati sampel tersebut melalui pohon dari root hingga mencapai daun (leaf node). Nilai prediksi diambil dari mayoritas label dalam daun tersebut.

4. Bandingkan hasil evaluasi model pada nomor 2 dan 3, bagaimana hasil perbandingannya? Jika ada perbedaan, jelaskan alasannya!
Jawab:

Menggunakan Implementasi From Scratch

```
import importlib
import cart
importlib.reload(cart)

from cart import DecisionTreeCART

cart = DecisionTreeCART(max_depth=5, min_samples_split=8)

# Using holdout validation
cart.fit(X_train, y_train)
y_pred = cart.predict(X_test)
ho_accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {ho_accuracy}")

# Using k-fold cross-validation
cv_accuracy = cross_val_score(cart, X, y, cv=kfold, scoring="accuracy").mean()
print(f"Cross-validation Accuracy: {cv_accuracy}")
```

✓ 0.6s

```
Accuracy: 0.7049180327868853
Cross-validation Accuracy: 0.7254098360655739
```

Menggunakan Implementasi dari Library

```
cart_lib = DecisionTreeClassifier(criterion='gini', max_depth=5, random_state=42)

# Using holdout validation
cart_lib.fit(X_train, y_train)
y_pred_lib = cart_lib.predict(X_test)
ho_accuracy_lib = accuracy_score(y_test, y_pred_lib)
print(f'Accuracy: {ho_accuracy_lib}')

# Using k-fold cross-validation
cv_accuracy_lib = cross_val_score(cart_lib, X, y, cv=kfold, scoring="accuracy").mean()
print(f'Cross-validation Accuracy: {cv_accuracy_lib}')
```

✓ 0.0s

```
Accuracy: 0.7049180327868853
Cross-validation Accuracy: 0.7251912568306011
```

Kedua model memberikan hasil akurasi yang sama pada holdout validation dan sangat mendekati pada cross-validation. Perbedaan kecil mungkin disebabkan oleh optimasi internal library, seperti efisiensi penggunaan memori atau perbedaan kecil dalam perhitungan angka floating-point.

5. Jelaskan improvement apa saja yang bisa Anda lakukan untuk mencapai hasil yang lebih baik dibandingkan dengan hasil yang Anda punya saat ini! Improvement yang dimaksud tidak terbatas pada bagaimana algoritma diimplementasikan, namun juga mencakup tahap sebelum modeling and validation.

Jawab:

- Melakukan pemilihan fitur yang lebih baik.
- Melakukan tuning hyperparameter.