CART (Decision Tree)

1. Cara kerja algoritma Decision Tree

Decision Tree adalah metode *machine learning* yang digunakan untuk klasifikasi dan regresi dengan cara membagi data menjadi subset yang lebih kecil berdasarkan fitur-fitur tertentu. Algoritma ini membuat keputusan dengan mengikuti struktur pohon, di mana setiap node mewakili pertanyaan tentang fitur, setiap cabang mewakili hasil dari pertanyaan tersebut, dan setiap daun (leaf) mewakili label klasifikasi atau nilai prediksi. Langkah-langkah utama dalam algoritma Gaussian Naive Bayes adalah sebagai berikut.

Inisialisasi:

- Mulai dari root node, di mana semua data pelatihan tersedia.
- Pada setiap node, pilih fitur dan threshold yang membagi data dengan cara yang terbaik berdasarkan metrik seperti Information Gain.

Pemilihan Fitur dan Threshold Terbaik:

- Untuk setiap fitur yang dipilih (jika menggunakan subset fitur), uji berbagai nilai threshold untuk menentukan pemisahan terbaik.
- Information Gain (atau metrik lain) digunakan untuk mengevaluasi seberapa baik fitur dan threshold membagi data. Metrik ini mengukur pengurangan ketidakpastian (entropy) dalam data setelah pemisahan.

Pembagian Data:

- Data dibagi menjadi dua subset berdasarkan threshold yang dipilih: satu subset untuk nilai yang lebih kecil atau sama dengan threshold dan satu subset untuk nilai yang lebih besar.

Rekursi:

- Proses ini diulang secara rekursif untuk setiap subset data yang terbagi, membangun subtree untuk masing-masing subset hingga kriteria pemberhentian tercapai (seperti kedalaman maksimum pohon atau jumlah sampel minimal).

Penentuan Daun (Leaf Node):

- Jika kondisi pemberhentian tercapai atau tidak ada pemisahan yang signifikan, buatlah node daun yang berisi label paling umum atau nilai prediksi untuk subset data tersebut.

Prediksi:

- Untuk melakukan prediksi, traversing pohon dimulai dari root node dan mengikuti cabang berdasarkan fitur dari data yang diprediksi sampai mencapai node daun.

4. Hasil Evaluasi Model

From Scratch	From Scikit-Learn
Accuracy: 0.8136645962732919 precision recall f1-score support 0 0.83 0.97 0.90 399 1 0.31 0.06 0.10 84 accuracy 0.81 483 macro avg 0.57 0.52 0.50 483 weighted avg 0.74 0.81 0.76 483	Accuracy: 0.8240165631469979 precision recall f1-score support 0 0.85 0.95 0.90 399 1 0.49 0.21 0.30 84 accuracy 0.82 483 macro avg 0.67 0.58 0.60 483 weighted avg 0.79 0.82 0.79 483

Model Decision Tree yang diimplementasikan dari awal (scratch) menghasilkan akurasi sebesar 81.37%, sedangkan model dari scikit-learn mencapai akurasi 82.40%. Selain itu, model scikit-learn menunjukkan kinerja yang lebih baik pada metrik precision, recall, dan f1-score, terutama pada kelas minoritas (kelas 1).

Hal ini terjadi karena implementasi keduanya mungkin memiliki perbedaan dalam cara perhitungan entropi, informasi gain, atau pemilihan threshold yang dapat mempengaruhi kinerja model. Misalnya, implementasi from scratch bisa jadi tidak mengoptimalkan pemilihan threshold dengan sebaik mungkin dibandingkan dengan algoritma yang sudah dioptimalkan dalam scikit-learn. Pada model scikit-learn, pemilihan fitur dan split dilakukan dengan optimasi yang lebih baik, sedangkan pada implementasi scratch, pemilihan fitur mungkin menggunakan teknik acak yang lebih sederhana. Selain itu, Implementasi from scratch mungkin mengalami masalah dengan stabilitas numerik atau efisiensi komputasi yang tidak terjadi pada implementasi scikit-learn yang sudah teruji dan dioptimalkan.

5. *Improvement*

Dalam hal optimasi model, penyetelan parameter yang cermat seperti max_depth, min_samples_split, min_samples_leaf, dan max_features dapat dilakukan menggunakan teknik seperti Grid Search atau Random Search untuk menemukan konfigurasi yang paling efektif. Selain itu, penerapan teknik pruning, baik pada tahap pembuatan pohon atau setelahnya, dapat mengurangi risiko overfitting dan meningkatkan kemampuan generalisasi model. Selain itu, pemilihan fitur yang relevan dengan metode seleksi fitur, seperti Recursive Feature Elimination (RFE), dapat memperbaiki kualitas model dengan mengurangi fitur yang kurang berkontribusi.

Dalam perbaikan implementasi algoritma, memperbaiki kriteria pemisahan dengan menggunakan metode yang lebih canggih seperti impurity Gini atau gain ratio, daripada sekadar entropy, dapat menghasilkan pembagian yang lebih informatif dan stabil. Peningkatan stabilitas numerik dalam perhitungan informasi gain dan entropi juga penting untuk menghindari masalah dengan angka ekstrem.