Gaussian Naive Bayes

1. Cara kerja algoritma Gaussian Naive Bayes

Gaussian Naive Bayes adalah algoritma klasifikasi yang didasarkan pada Teorema Bayes dengan asumsi bahwa fitur-fitur bersifat independen satu sama lain (asumsi naif) dan mengikuti distribusi Gaussian (Normal). Algoritma ini digunakan untuk memprediksi probabilitas suatu data termasuk dalam salah satu kelas yang telah ditentukan berdasarkan nilai-nilai fitur yang dimiliki oleh data tersebut. Langkah-langkah utama dalam algoritma Gaussian Naive Bayes adalah sebagai berikut.

Inisialisasi dan Training:

- Ambil data input X dan target y.
- Identifikasi kelas unik dalam y dan simpan sebagai classes.
- Untuk setiap kelas c dalam classes:
    - Ambil data subset X_c dimana data memiliki target c.
    - Hitung mean (mean) dari fitur-fitur dalam X_c dan simpan.
    - Hitung variansi (var) dari fitur-fitur dalam X_c dan simpan.
    - Hitung prior probability (priors) dengan membagi jumlah data dalam X_c dengan total data.

Prediksi Data Baru:

- Untuk setiap data baru x:
    - Inisialisasi array kosong posteriors.
    - Untuk setiap kelas c dalam classes:
        - Hitung prior log dari priors untuk kelas c.
        - Untuk setiap fitur j pada data x:
            - Hitung likelihood dengan formula Gaussian menggunakan mean dan variansi dari kelas c.
            - Tambahkan log dari likelihood ke prior log.
        - Simpan hasil perhitungan ini ke posteriors.
    - Pilih kelas dengan nilai tertinggi dalam posteriors sebagai prediksi.

4. Hasil Evaluasi Model

From Scratch	From Scikit-Learn
Accuracy: 0.8012422360248447 precision recall f1-score support 0 0.88 0.88 0.88 399 1 0.43 0.43 0.43 84 accuracy 0.80 483 macro avg 0.65 0.65 0.65 483 weighted avg 0.80 0.80 0.80 483	Accuracy: 0.8012422360248447 precision recall f1-score support 0 0.88 0.88 0.88 399 1 0.43 0.43 0.43 84 accuracy 0.80 483 macro avg 0.65 0.65 0.65 483 weighted avg 0.80 0.80 0.80 483

Hasil evaluasi menunjukkan bahwa performa model memiliki akurasi yang identik. Precision, Recall, F1-Score juga sama untuk masing-masing kelas pada kedua implementasi. Perbandingan ini menunjukkan bahwa model dari kedua pendekatan memberikan performa yang serupa, yang merupakan indikator bahwa implementasi scratch Anda sudah baik.

5. *Improvement*

Untuk meningkatkan performa model dapat dilakukan Implementasikan smoothing untuk menangani variansi nol dalam fitur, yang dapat menghindari masalah ketika variansi fitur tertentu adalah nol. Misalnya, dengan menambahkan epsilon kecil pada varian. Selain itu, dapat dilakukan feature engineering dengan membuat fitur baru yang relevan atau transformasi fitur yang ada untuk meningkatkan asumsi Gaussian, seperti interaksi antar fitur, log transformation, atau polynomial features. Fitur yang relevan dapat meningkatkan kemampuan model untuk belajar dari data.