1. Cara kerja algoritma algoritma Gaussian Naive Bayes pada repository ini

a. Inisialisasi parameter

Proses dimulai dengan menginisialisasi parameter model, yaitu daftar kelas (classes), mean (mean), variansi (var), dan prior (priors). Parameter ini diinisialisasi sebagai array kosong yang akan diisi dengan nilai yang dihitung dari data pelatihan.

b. Menghitung *mean*, *variance*, dan *prior* untuk setiap kelas

Saat metode fit dipanggil, model menghitung statistik dasar untuk setiap kelas dalam data pelatihan. Ini mencakup *mean* (rata-rata) dan *variance* (varian) dari setiap fitur untuk setiap kelas. *Prior probability* dihitung sebagai proporsi jumlah sampel dari seti ap kelas dibandingkan dengan total jumlah sampel. Prior ini menunjukkan probabilitas dasar dari setiap kelas tanpa melihat fitur.

c. Menghitung *likelihood*:

Untuk setiap fitur dalam sampel baru yang akan diklasifikasikan, model menghitung *likelihood* berdasarkan distribusi Gaussian yang telah dihitung sebelumnya. *Likelihood* ini mengukur seberapa besar kemungkinan suatu fitur bernilai tertentu, diberikan kelas tertentu. *Likelihood* untuk distribusi Gaussian dihitung menggunakan rumus:

$$P(x_i|c) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right)$$

di mana :

- x_i adalah nilai fitur untuk sampel tersebut
- μ adalah *mean* dari fitur untuk kelas tersebut
- σ^2 adalah varians dari fitur untuk kelas tersebut

d. Menghitung *posterior probability*

Model kemudian menghitung *posterior probability* untuk setiap kelas menggunakan **Teorema Bayes**. Posterior ini menunjukkan probabilitas bahwa sampel termasuk dalam kelas tertentu, diberikan fitur yang diamati. Posterior dihitung dengan mengalikan prior dengan *likelihood* dari semua fitur:

$$P(c|x) \propto P(x|c) \cdot P(c)$$

Namun, untuk stabilitas numerik, formula berikut yang digunakan:

$$\log P(c|x) = \log P(c) + \sum_i \log P(x_i|c)$$

e. Prediksi kelas

Setelah menghitung *posterior probability* untuk semua kelas, model memilih kelas dengan probabilitas tertinggi sebagai prediksi. Proses ini diulang untuk setiap sampel dalam dataset uji.

f. Menghindari masalah numerik:

Nilai epsilon kecil pada perhitungan likelihood dan prior, untuk mencegah masalah numerik seperti *underflow* saat menghitung logaritma dari nilai kecil.

g. Menghasilkan prediksi
   Setelah seluruh data uji diproses, model mengembalikan prediksi untuk setiap sampel berdasarkan kelas yang memiliki posterior tertinggi.

4. Perbandingan hasil evaluasi model

Berdasarkan hasil evaluasi model, *Gaussian Naive Bayes* dari *Scratch* dan *Gaussian Naive Bayes* dari scikit-learn menunjukkan hasil yang hampir identik. Kedua model memiliki Akurasi 0.75 dan F1 Score 0.62 pada set pengujian. Hasil K-Fold Cross-Validation juga konsisten dengan Cross-Validation Accuracy 0.74 ± 0.01 dan F1 Score 0.62 ± 0.02 untuk kedua model.

Kesamaan hasil ini mungkin disebabkan oleh kesederhanaan algoritma *Gaussian Naive Bayes*. Kedua implementasi menggunakan rumus dasar yang sama tanpa optimisasi khusus, sehingga hasil evaluasi cenderung sama. Selain itu, karena model *Gaussian Naive Bayes* sepertinya bersifat stabil pada dataset yang sesuai dengan asumsi distribusinya, tidak ada perbedaan signifikan antara implementasi dari scratch dan versi scikit-learn

5. Improvement yang bisa dilakukan

Untuk meningkatkan kinerja *Gaussian Naive Baye***s**, ada beberapa perbaikan yang mungkin bisa dilakukan pada tingkat algoritma. Salah satunya adalah mengatur nilai minimum untuk variansi guna menghindari pembagian dengan nol dan membuat perhitungan likelihood lebih stabil. Selain itu, metode ensemble seperti *Bagging* atau *Boosting* mungkin juga dapat digunakan untuk menggabungkan prediksi dari beberapa model, meningkatkan akurasi, dan mengurangi *overfitting*. Alternatif lain adalah penerapan *Bayesian Model Averaging* yang mungkin juga dapat meningkatkan estimasi probabilitas dengan memperhitungkan ketidakpastian dari beberapa model.