1. Cara kerja algoritma SVM dari scratch pada repository ini

a. Inisialisasi parameter

Algoritma SVM diinisialisasi dengan beberapa parameter penting seperti learning_rate, num_iterations, lambda_param (parameter regulasi), kernel (jenis kernel), degree (untuk kernel polynomial), dan gamma (untuk kernel RBF). Selain itu, bobot model (**w**) diinisialisasi sebagai array nol dengan panjang sesuai jumlah sampel pelatihan, dan bias (**b**) diatur ke nol. Selama proses ini, X_train juga disimpan dalam objek model untuk digunakan dalam perhitungan kernel pada saat prediksi.

b. Pemilihan kernel

Ada tiga jenis kernel didukung dalam implementasi ini, yaitu linear, polynomial, dan RBF (Radial Basis Function). Pemilihan kernel menentukan fungsi yang digunakan untuk menghitung jarak atau kesamaan antara sampel.

- **Linear Kernel**

  Menggunakan dot product antara dua vektor

$$K(X_i, X_j) = X_i \cdot X_j$$

- **Polynomial Kernel**

  Menaikkan hasil dot product ke suatu derajat tertentu

$$K(X_i, X_j) = (1 + X_i \cdot X_j)^d$$

- **RBF (Gaussian) Kernel**

  Menggunakan fungsi eksponensial untuk mengukur kesamaan antara dua titik dengan gamma sebagai parameter

$$K(X_i, X_j) = \exp(-\gamma \|X_i - X_j\|^2)$$

c. Pembentukan kernel matrix

Selama pelatihan, model pertama-tama menghitung kernel matrix *K* antara X_train dengan dirinya sendiri. Kernel matrix ini adalah representasi dari jarak atau kesamaan antar data, yang digunakan untuk mengubah data ke dalam dimensi yang lebih tinggi, memungkinkan pemisahan yang lebih baik untuk data yang tidak dapat dipisahkan secara linear.

d. Pelatihan model menggunakan gradient descent

Model dilatih menggunakan iterasi gradient descent untuk memperbarui bobot dan bias. Setiap iterasi mencakup semua sampel, dan untuk setiap sampel, model memeriksa apakah kondisi margin terpenuhi. Jika suatu sampel berada di luar margin, bobot diperbarui hanya dengan mempertimbangkan penalti regularisasi. Jika tidak, bobot dan bias diperbarui untuk mengurangi kesalahan klasifikasi.

e. Menghitung Loss
Pada interval tertentu, model menghitung hinge loss untuk memantau performa. Hinge loss diukur untuk menentukan seberapa baik model memisahkan data dengan margin yang benar. Formula untuk hinge loss adalah:

$$L(w,b) = \frac{1}{2}\|w\|^2 + \lambda \sum \max(0, 1 - y_i(\sum(\alpha_i y_i K(X_i, X_j)) + b))$$

Di mana L(w, b) adalah total loss, dan $\|w\|^2$ adalah penalti regularisasi.

f. Prediksi
Setelah pelatihan selesai, model dapat digunakan untuk memprediksi kelas dari data baru (X_test). Kernel matrix dihitung antara X_test dan X_train untuk setiap sampel. Model kemudian menggunakan kernel matrix ini bersama dengan bobot (**w**) dan bias (**b**) yang sudah dipelajari untuk menghasilkan prediksi.

4. Perbandingan hasil evaluasi model

Berdasarkan hasil evaluasi, terdapat perbedaan kinerja antara model SVM yang diimplementasikan dari scratch dan model SVM yang menggunakan pustaka scikit-learn, baik dengan kernel linear maupun polynomial.

a. **Akurasi dan F1 Score**: Model SVM dari scikit-learn, baik untuk kernel linear maupun polynomial, menunjukkan performa yang lebih baik dibandingkan dengan implementasi dari scratch. Pada set uji, SVM_Sklearn dengan kernel linear memiliki akurasi 0.81 dan F1 Score 0.61, sementara SVM_Scratch dengan kernel linear memiliki akurasi 0.79 dan F1 Score 0.49. Perbedaan serupa juga terlihat pada kernel polynomial, di mana scikit-learn unggul dalam akurasi dan F1 Score.
b. **Cross-Validation**: Hasil K-Fold Cross-Validation juga menunjukkan bahwa model scikit-learn lebih stabil dan konsisten. SVM_Sklearn (Linear Kernel) mencapai Cross-Validation Accuracy 0.80 ± 0.01 dan F1 Score 0.59 ± 0.02, sementara SVM_Scratch (Linear Kernel) hanya mencapai 0.78 ± 0.01 untuk akurasi dan 0.42 ± 0.18 untuk F1 Score. Hasil ini juga konsisten untuk kernel polynomial.
c. **Alasan Perbedaan**
    1. **Teknik Optimasi yang Berbeda**

       Implementasi SVM dari scratch menggunakan gradient descent yang mungkin cenderung kurang efisien dan stabil dibandingkan dengan metode optimasi seperti Sequential Minimal Optimization (SMO) yang digunakan oleh scikit-learn.

    2. **Efisiensi Penggunaan Kernel**

Meskipun SVM dari scratch mendukung kernel polynomial dan RBF, cara implementasi dan penggunaan kernel tersebut mungkin kurang optimal. Saya cukup yakin jika, scikit-learn punya implementasi kernel dan regularisasi yang lebih baik, sehingga lebih mampu menangani data yang kompleks dan memberikan hasil yang lebih baik dalam hal akurasi dan stabilitas model

5. Improvement yang bisa dilakukan

Berikut adalah beberapa ide improvisasi yang mungkin bisa dilakukan

Pertama, mengganti metode optimasi yang digunakan. Alih-alih menggunakan gradient descent, yang bisa kurang efisien dan stabil, optimasi lain seperti Sequential Minimal Optimization (SMO) bisa dicoba untuk digunakan. Metode ini sepertinya lebih cepat dan lebih stabil, terutama pada dataset yang besar atau ketika data tidak bisa dipisahkan secara linear. Dengan optimasi yang lebih baik, model bisa lebih akurat dan lebih cepat mencapai hasil yang diinginkan.

Kedua, menambahkan lebih banyak jenis kernel yang lebih kompleks. Saat ini, model mendukung kernel linear, polynomial, dan RBF (Gaussian), tetapi bisa ditingkatkan dengan menambahkan kernel lain seperti sigmoid atau kombinasi beberapa kernel. Kernel yang lebih beragam akan memungkinkan model untuk lebih fleksibel dalam menangani berbagai jenis data.

Selain itu, penggunaan regularisasi yang lebih canggih atau hyperparameter tuning seperti pencarian grid atau pencarian acak untuk menemukan kombinasi parameter yang optimal juga bisa membantu meningkatkan kinerja model.