KNN

1. Cara kerja algoritma KNN

Algoritma K-Nearest Neighbors (KNN) adalah metode machine learning yang digunakan untuk klasifikasi dan regresi dengan memanfaatkan kesamaan antara data point. Algoritma ini bersifat non parametrik, sehingga tidak memerlukan asumsi mengenai distribusi data. Dalam KNN, prediksi dibuat dengan mencari K tetangga terdekat dari titik data tertentu menggunakan metrik jarak seperti Euclidean, Manhattan, atau Minkowski. Setelah K tetangga terdekat ditemukan, kelas atau nilai titik data tersebut ditentukan melalui suara mayoritas untuk klasifikasi, atau rata-rata nilai untuk regresi. Pendekatan ini membuat KNN fleksibel dalam mengenali pola dan menghasilkan prediksi berdasarkan struktur data lokal. Lebih runutnya, berikut *pseudocode* cara kerja algoritma KNN.

- Input:
  - Data point baru (*new_point*)
  - Dataset pelatihan (*training_data*)
  - Nilai K (jumlah tetangga)
  - Metrik jarak (distance_metric: Euclidean, Manhattan, Minkowski)
- Output:
  - Prediksi kelas atau nilai (classification/regression)
- Langkah:
  - Step 1: Inisialisasi nilai K dan metrik jarak.
  - Step 2: Untuk setiap data point dalam dataset pelatihan:
    a. Jarak dihitung antara setiap titik data dalam kumpulan data dan titik target. Hitung jarak antara new_point dan data point saat ini menggunakan distance_metric:
       - *Euclidean*: jarak kartesius antara dua titik yang berada di bidang.

$$\mathrm{dist}(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

       - *Manhattan*: jumlah selisih absolut antara koordinat titik-titik dalam dimensi-n.

$$distance = \sum_{1}^{n} |p_i - q_i|$$

       - *Minkowski*: metrik dalam ruang vektor bernorma yang digunakan untuk kesamaan jarak vektor.

$$d(x,y) = (\sum_{i=1}^{n}(x_i - y_i)^p)^{\frac{1}{p}}$$

         Jika p = 1, maka Minkowski distance sama dengan Manhattan distance. Jika p = 2, maka Minkowski distance sama dengan Euclidean distance. Sementara

jika p > 3, jaraknya akan lebih sensitif terhadap perbedaan besar antara komponen x1 dan x2.

b. Simpan data point dan jaraknya dalam list.

- Step 3: Urutkan list berdasarkan jarak dalam urutan menaik.
- Step 4: Pilih K data point terdekat dari list yang telah diurutkan.
- Step 5: Jika KNN digunakan untuk klasifikasi:
    a. Lakukan pemilihan suara mayoritas pada kelas dari K tetangga terdekat.
    b. Tentukan kelas dengan jumlah suara terbanyak sebagai prediksi untuk new_point.
- Step 6: Jika KNN digunakan untuk regresi:
    a. Hitung rata-rata nilai target dari K tetangga terdekat.
    b. Tentukan nilai rata-rata sebagai prediksi untuk new_point.
- Step 7: Kembalikan prediksi kelas atau nilai.
- End Algorithm

4. Hasil Evaluasi Model

From Scratch	From Scikit-Learn
Accuracy: 0.8327814569536424 precision recall f1-score support 0 0.86 0.95 0.90 504 1 0.49 0.23 0.31 100 accuracy 0.83 604 macro avg 0.68 0.59 0.61 604 weighted avg 0.80 0.83 0.81 604	Accuracy: 0.8327814569536424 precision recall f1-score support 0 0.86 0.95 0.90 504 1 0.49 0.23 0.31 100 accuracy 0.83 604 macro avg 0.68 0.59 0.61 604 weighted avg 0.80 0.83 0.81 604

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 modeling dengan memanfaatkan parameter tuning dengan menggunakan teknik pencarian hyperparameter seperti Grid Search atau Random Search untuk menemukan nilai k yang optimal dan parameter lain yang relevan. Selain itu, dapat dilakukan feature engineering dengan membuat fitur baru yang relevan atau transformasi fitur yang ada, seperti interaksi antar fitur, log transformation, atau polynomial features. Fitur yang relevan dapat meningkatkan kemampuan model untuk belajar dari data.