K-Nearest Neighbors (KNN) Classifier

1. Pendahuluan

K-Nearest Neighbors (KNN) adalah algoritma non-parametrik untuk klasifikasi yang berdasarkan pada prinsip bahwa suatu data x akan diprediksi sebagai kelas mayoritas dari k tetangga terdekatnya dalam ruang fitur. Algoritma ini tidak melakukan proses pelatihan eksplisit, melainkan menyimpan data pelatihan dan melakukan prediksi dengan membandingkan data baru terhadap seluruh data pelatihan.

2. Formulasi Matematis

Diberikan:

- Dataset pelatihan: $\mathrm{D} = \{(x_i, y_i)\}_{i=1}^{n}$
- Titik input baru: $x \in \mathrm{R}^d$
- Fungsi jarak: $d : \mathrm{R}^d \times \mathrm{R}^d \to \mathrm{R}$

Langkah prediksi:

1. Hitung jarak $d(x, x_i)$ untuk seluruh $i \in \{1, \ldots, n\}$
2. Ambil $\mathrm{N}_k(x) \subset \mathrm{D}$, yaitu k data pelatihan dengan jarak terkecil terhadap x
3. Prediksi kelas berdasarkan mayoritas:

$$\hat{y} = \arg\max_{c \in \mathrm{C}} \sum_{(x_i, y_i) \in \mathrm{N}_k(x)} 1[y_i = c]$$

3. Parameter dan Pengaturan

- k: Jumlah tetangga yang dipertimbangkan dalam prediksi.
- **distance_metric**: Jenis fungsi jarak yang digunakan:
  - **Euclidean**: $d(x, x') = \sqrt{\sum_j (x_j - x'_j)^2}$
  - **Manhattan**: $d(x, x') = \sum_j |x_j - x'_j|$
  - **Minkowski (p=2)**: $d(x, x') = \left(\sum_j |x_j - x'_j|^p\right)^{1/p}$

4. Algoritma

4.1. Training (fit)

Fungsi `fit(X, y)` menyimpan data pelatihan tanpa proses parameter learning. Data disimpan dalam:

$$X_{\text{train}} \in \mathrm{R}^{n \times d}, \quad y_{\text{train}} \in \mathrm{R}^n$$

4.2. Prediksi

Untuk setiap sampel uji x:

1. Hitung jarak terhadap semua $x_i \in X_{\text{train}}$
2. Pilih k sampel dengan jarak terkecil
3. Ambil label terbanyak sebagai hasil prediksi $\hat{y}$

$$\hat{y}(x) = \text{mode}(y_i \mid (x_i, y_i) \in \text{N}_k(x))$$

4.3. Tiebreaking

Jika terjadi jumlah kelas yang sama, digunakan aturan `np.argmax` pada `np.unique(..., return_counts=True)` untuk mengambil label dengan urutan indeks terkecil dalam `unique_labels`.

5. Kompleksitas

- **Training time**: $\text{O}(1)$
- **Prediction time**: $\text{O}(n \cdot d)$ per sampel uji
- **Memory usage**: $\text{O}(n \cdot d)$