K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) adalah algoritma supervised learning yang dapat digunakan untuk masalah klasifikasi dan regresi. KKN bekerja dengan membandingkan data baru dengan **k** tetangga terdekatnya dalam ruang fitur. Tetangga terdekat adalah titik data yang paling mirip dengan data baru berdasarkan jarak tertentu (misalnya, jarak Euclidean).

# Cara Kerja KNN (Klasifikasi):

1. **Hitung Jarak:** Hitung jarak antara data baru dan semua titik data dalam data latih.
2. **Pilih Tetangga Terdekat:** Pilih **k** titik data dengan jarak terdekat sebagai tetangga terdekat.
3. **Hitung Mayoritas:** Tentukan kelas dari data baru berdasarkan kelas mayoritas dari **k** tetangga terdekatnya.

# Memilih Nilai K

Nilai **K** adalah hyperparameter penting dalam KNN. Nilai **k** yang terlalu kecil dapat membuat model sensitif terhadap noise, sedangkan nilai **k** yang terlalu besar dapat membuat model tidak dapat menangkap pola lokal dalam data. Anda dapat mencoba beberapa nilai **k** yang berbeda dan menggunakan cross-validation untuk menemukan nilai **k** yang optimal.

# Keunggulan KNN:

* **Sederhana dan Mudah Dipahami:** Konsep dan implementasinya mudah dipahami.
* **Tidak Ada Asumsi tentang Distribusi Data:** Tidak memerlukan asumsi tentang distribusi data, seperti normalitas.
* **Dapat Digunakan untuk Klasifikasi dan Regresi:** Fleksibel dan dapat digunakan untuk berbagai jenis masalah.

# Kelemahan KNN:

* **Lambat pada Data Besar:** Kinerja menurun saat dataset bertambah besar karena perlu menghitung jarak dengan semua titik data.
* **Sensitif terhadap Skala Fitur:** Fitur dengan rentang nilai yang besar akan lebih mendominasi perhitungan jarak.
* **Sensitif terhadap Noise dan Outlier:** Outlier dapat mempengaruhi hasil prediksi.

**Tugas Hari 29:**

Menggunakan dataset “Iris”

**Tugas:**

1. **Membuat Model KNN:**

* Gunakan KNeighboarsClassifier dari scikit-learn untuk membuat model KNN dengan k = 3.
* Bagi data menjadi data latih (80%) dan data uji (20%).
* Latih model pada data latih.

1. **Prediksi dan Evaluasi:**

* Gunakan model untuk memprediksi label kelas pada data uji.
* Hitung akurasi model pada data uji.

1. **Optimasi K (Opsional):**

* Coba beberapa nilai k yang berbeda (misalnya, 1, 5, 7) dan bandingkan kinerjanya.
* Gunakan cross-validation untuk menemukan nilai k yang optimal.

**Contoh Kode (Scikit-learn)**

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import cross\_val\_score

# ... (kode untuk membaca dan mempersiapkan dataset Iris)

# Membuat model KNN (k=3)

model = KNeighborsClassifier(n\_neighbors=3)

# Melatih model

model.fit(X\_train, y\_train)

# Prediksi pada data uji

y\_pred = model.predict(X\_test)

# Evaluasi akurasi

accuracy = accuracy\_score(y\_test, y\_pred)

print(f'Akurasi model (k=3): {accuracy:.2f}')

# Cross-validation untuk mencari k terbaik (opsional)

k\_values = [1, 3, 5, 7]

for k in k\_values:

model = KNeighborsClassifier(n\_neighbors=k)

scores = cross\_val\_score(model, X, y, cv=5, scoring='accuracy')

print(f'Akurasi rata-rata (k={k}): {scores.mean():.2f} (+/- {scores.std():.2f})')

**Catatan:**

Coba bereksperimen dengan nilai k dan matrik jarak yang berbeda untuk melihat bagimana hal tersebut mempengaruhi kinerja model.

**Selamat Mengerjakan Tugas! 🙂**

**Tugas:**

1. **Membuat Model KNN:**

**from sklearn.neighbors import KNeighborsClassifier**

**from sklearn.model\_selection import cross\_val\_score**

**# ...(Kode untuk Membaca dan Persiapan dataset)**

**# Membuat Model**

**model = KNeighborsClassifier(n\_neighbors=3)**

**# Melatih Model**

**model.fit(X\_train, y\_train)**

1. **Prediksi dan Evaluasi:**

**# Prediksi pada data uji**

**y\_pred = model.predict(X\_test)**

**# Evaluasi**

**accuracy = accuracy\_score(y\_test, y\_pred)**

**print(f'\nAkurasi Model (K=3): {accuracy:.2f}')**

1. **Optimasi K (Opsional):**

**# Cross-Validation untuk mencari k terbaik**

**k\_values = [1, 3, 5, 7]**

**for k in k\_values:**

**model = KNeighborsClassifier(n\_neighbors=k)**

**scores = cross\_val\_score(model, X, y, cv=5, scoring='accuracy')**

**print(f'\nAkurasi Rata-Rata (k={k}): {scores.mean():.2f} (+/- {scores.std():.2f})')**

# Penjelasan Kode:

1. **Membuat Model KNN:**
   * model = KNeighborsClassifier(n\_neighbors=3): Membuat objek model KNN dengan n\_neighbors=3, yang berarti menggunakan 3 tetangga terdekat untuk membuat prediksi.
   * model.fit(X\_train, y\_train): Melatih model pada data latih.
2. **Prediksi dan Evaluasi:**
   * y\_pred = model.predict(X\_test): Melakukan prediksi pada data uji menggunakan model KNN yang sudah dilatih.
   * accuracy = accuracy\_score(y\_test, y\_pred): Menghitung akurasi model dengan membandingkan prediksi (y\_pred) dengan label sebenarnya (y\_test).
   * print(f'\nAkurasi Model (K=3): {accuracy:.2f}'): Mencetak akurasi model dengan format dua angka di belakang koma.
3. **Optimasi K (Opsional):**
   * k\_values = [1, 3, 5, 7]: Mendefinisikan beberapa nilai k yang akan dicoba.
   * for k in k\_values: Melakukan perulangan untuk setiap nilai k.
     + model = KNeighborsClassifier(n\_neighbors=k): Membuat model KNN baru dengan nilai k yang berbeda.
     + scores = cross\_val\_score(model, X, y, cv=5, scoring='accuracy'): Melakukan 5-fold cross-validation untuk mengevaluasi model dengan nilai k yang berbeda.
     + print(f'\nAkurasi Rata-Rata (k={k}): {scores.mean():.2f} (+/- {scores.std():.2f})'): Mencetak rata-rata dan standar deviasi akurasi dari cross-validation.

**Interpretasi Hasil:**

* **Akurasi Model (k=3):** Menunjukkan akurasi model KNN dengan k=3 pada data uji. Nilai ini memberikan gambaran awal tentang kinerja model.
* **Akurasi Rata-Rata (k={k}):** Menunjukkan rata-rata akurasi model KNN dengan nilai k yang berbeda-beda dari hasil cross-validation. Standar deviasi menunjukkan seberapa bervariasi akurasi pada setiap fold. Anda dapat memilih nilai k yang memberikan akurasi rata-rata tertinggi dan standar deviasi terendah.

**Catatan:**

* Hasil cross-validation dapat bervariasi tergantung pada pembagian data latih dan uji, serta nilai random state.
* Anda dapat mencoba nilai k yang lebih banyak dan lebih beragam dalam cross-validation untuk menemukan nilai k yang optimal.