kNN • Naive Bayes • k-Fold CV

Fokus:

1. k-Nearest Neighbors (kNN)
2. Naive Bayes (NB)
3. k-Fold Cross-Validation

Tujuan Sesi

- Memahami **intuisi & rumus inti** kNN dan NB
- Mampu **menghitung manual** contoh kecil
- Menjalankan **kode minimal** di scikit-learn
- Memahami **k-Fold CV** untuk evaluasi yang adil

Bagian A — kNN

kNN: Intuisi Singkat

- **Tidak benar-benar "melatih"** model; kNN **mengingat data latih**.
- Prediksi titik baru:
    i. Hitung **jarak** ke semua data latih
    ii. Ambil **k tetangga terdekat**
    iii. **Voting** mayoritas kelas tetangga (atau bobot jarak)

Parameter penting

- **k** (mis. 3, 5) → kecil sensitif noise; besar lebih stabil
- **Metode jarak**: Euclidean untuk angka
- **Scaling** fitur numerik **sangat membantu**

kNN: Rumus Jarak (Euclidean)

Untuk titik (p=(x_p,y_p)) dan (q=(x_q,y_q)):

$$d(p,q) = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}$$

Umum ke dimensi (n):

$$d(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$

kNN: Hitung Manual (2D, k=3)

Data latih (kelas 0 & 1):

Titik	x	y	Kelas
A	1	1	0
B	2	2	0
C	1	2	0
D	5	5	1
E	6	5	1
F	5	6	1

Titik uji: **T = (3,3)**.
Hitung jarak → urut:

- B: $\sqrt{2} \approx 1.414$ (0)
- C: $\sqrt{5} \approx 2.236$ (0)
- A: $\sqrt{8} \approx 2.828$ (0)
- D: $\sqrt{8} \approx 2.828$ (1)
- E/F: $\sqrt{13} \approx 3.606$ (1)

k=3 → {B,C,A} ⇒ mayoritas **0** → **Prediksi 0**.

kNN: Kode Minimal (Pipeline)

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report

X, y = load_iris(return_X_y=True)
Xtr, Xte, ytr, yte = train_test_split(X, y, test_size=0.2,
                                      random_state=42, stratify=y)

clf = make_pipeline(StandardScaler(), KNeighborsClassifier(n_neighbors=5))
clf.fit(Xtr, ytr)
yp = clf.predict(Xte)

print("Accuracy:", accuracy_score(yte, yp))
print(classification_report(yte, yp))
```

Catatan: `StandardScaler()` mencegah fitur besar “mendominasi” jarak.

Bagian B — Naive Bayes

Naive Bayes: Intuisi

- Dasar **Teorema Bayes**:
  $P(y|X) \propto P(X|y), P(y)$
- “Naive” = **anggap fitur saling independen bersyarat** pada kelas → memudahkan hitung:
  $P(X|y) = \prod_i P(x_i|y)$
- Varian populer:
  - **Multinomial/Bernoulli NB** (teks: hitung kata/biner)
  - **Gaussian NB** (fitur numerik $(\sim)$ normal)

NB Multinomial: Hitung Manual (Teks Mini)

Kelas: **Spam (S)** vs **Ham (H)**
Korpus mini (3 spam, 3 ham):

- Spam: “**gratis** hadiah”, “**gratis** kupon”, “hadiah besar **gratis**”
- Ham: “rapat **besok**”, “tugas **besok** dikumpulkan”, “jadwal kuliah **besok**”

Kata unik: gratis, hadiah, kupon, besar, rapat, besok, tugas, dikumpulkan, jadwal, kuliah

Prior:

- (P(S)=3/6=0.5), (P(H)=0.5)

Hitung **likelihood** per kata dengan **laplace smoothing** (α=1):

$$P(w|S) = \frac{\text{count}(w \text{ di Spam}) + \alpha}{\sum_w \text{count} + \alpha V}$$

$$P(w|H) = \frac{\text{count}(w \text{ di Ham}) + \alpha}{\sum_w \text{count} + \alpha V}$$

Misal total token Spam=6, Ham=6, (V=10).

Contoh:

- `gratis` di Spam=3 → $(P(\text{gratis}|S) = (3+1)/(6+10) = 4/16 = 0.25)$
- `gratis` di Ham=0 → $(P(\text{gratis}|H) = (0+1)/(6+10) = 1/16)$

Kalimat uji: "**gratis** hadiah"

Skor (abaikan konstanta):

- $(P(S|X) \propto P(S) \cdot P(\text{gratis}|S) \cdot P(\text{hadiah}|S))$
- $(P(H|X) \propto P(H) \cdot P(\text{gratis}|H) \cdot P(\text{hadiah}|H))$

Jika kata `hadiah` lebih sering di Spam, skor Spam > Ham ⇒ **prediksi Spam**.

NB Gaussian: Rumus Inti (Numerik)

Untuk fitur numerik (x_i) diasumsikan Gaussian di tiap kelas (y):

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_{iy}^2}}\exp!\left(-\frac{(x_i-\mu_{iy})^2}{2\sigma_{iy}^2}\right)$$

Log-probabilitas total:

$$\log P(y|X) \propto \log P(y) + \sum_i \log P(x_i|y)$$

Bandingkan semua kelas → ambil yang terbesar.

NB: Kode Minimal (Teks & Numerik)

Teks (MultinomialNB)

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

texts = ["gratis hadiah", "gratis kupon", "hadiah besar gratis",
         "rapat besok", "tugas besok dikumpulkan", "jadwal kuliah besok"]
y = [1,1,1, 0,0,0]  # 1=Spam, 0=Ham

X = CountVectorizer().fit_transform(texts)
Xtr, Xte, ytr, yte = train_test_split(X, y, test_size=0.33, random_state=42, stratify=y)

clf = MultinomialNB()
clf.fit(Xtr, ytr)
print(classification_report(yte, clf.predict(Xte)))
```

Numerik (GaussianNB)

```
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

X, y = load_iris(return_X_y=True)
Xtr, Xte, ytr, yte = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

gnb = GaussianNB()
gnb.fit(Xtr, ytr)
print("Accuracy:", accuracy_score(yte, gnb.predict(Xte)))
```

Bagian C — k-Fold Cross-Validation

k-Fold CV: Intuisi

- Satu split train/test terkadang **beruntung/sial**.
- **k-Fold CV** membagi data jadi **k lipatan (fold)**:
  - Ulangi k kali: **(k-1) fold untuk train**, **1 fold untuk validasi**
  - **Rata-ratakan** metrik → estimasi performa **lebih stabil**

Tip praktis

- Klasifikasi: gunakan **StratifiedKFold** agar proporsi kelas terjaga.
- Contoh umum: **k = 5** atau **10**.

k-Fold CV: Ilustrasi Manual (k=3)

Data indeks 0…8 → bagi 3 fold:

- Fold1: val = {0,1,2}, train = {3–8}
- Fold2: val = {3,4,5}, train = {0–2,6–8}
- Fold3: val = {6,7,8}, train = {0–5}

Latih-nilai di tiap fold → dapat 3 skor (mis. accuracy):

$$\text{cv\_score} = \frac{s_1+s_2+s_3}{3}$$

k-Fold CV: Kode Minimal (sklearn)

```
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import load_iris

X, y = load_iris(return_X_y=True)
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

scores = cross_val_score(KNeighborsClassifier(n_neighbors=5),
                         X, y, cv=cv, scoring='accuracy')
print("Fold scores:", scores)
print("Mean ± std:", scores.mean(), "±", scores.std())
```

Catatan: Gunakan **pipeline** bila ada scaling/encoding agar **anti-leakage** selama CV.

Kesalahan Umum & Solusi

- ❌ **Leakage**: scaling/encoding sebelum split/CV → **Gunakan Pipeline**
- ❌ Hanya lihat **accuracy** saat data timpang → **lihat juga P/R/F1**
- ❌ k terlalu kecil/besar di kNN → **coba 3,5,7** dan bandingkan via CV
- ❌ NB Multinomial tanpa smoothing → **laplace α=1** default sudah aman

Ringkasan

- **kNN**: tetangga terdekat + jarak; butuh scaling
- **Naive Bayes**: Bayes + asumsi independensi; cepat & kuat untuk teks
- **k-Fold CV**: evaluasi rata-rata yang stabil
- **Pipeline**: kunci anti-leakage saat praproses + evaluasi