Bab 07 — Supervised Learning

Status: Draft lengkap v0.3 — struktur subbab mendalam Bagian buku: Level D — Cara Model Belajar dari Data Target pembaca: Pembaca yang sudah memahami fondasi ML Bab 6 dan siap membandingkan keluarga algoritma supervised learning secara intuitif, teknis, dan praktis.

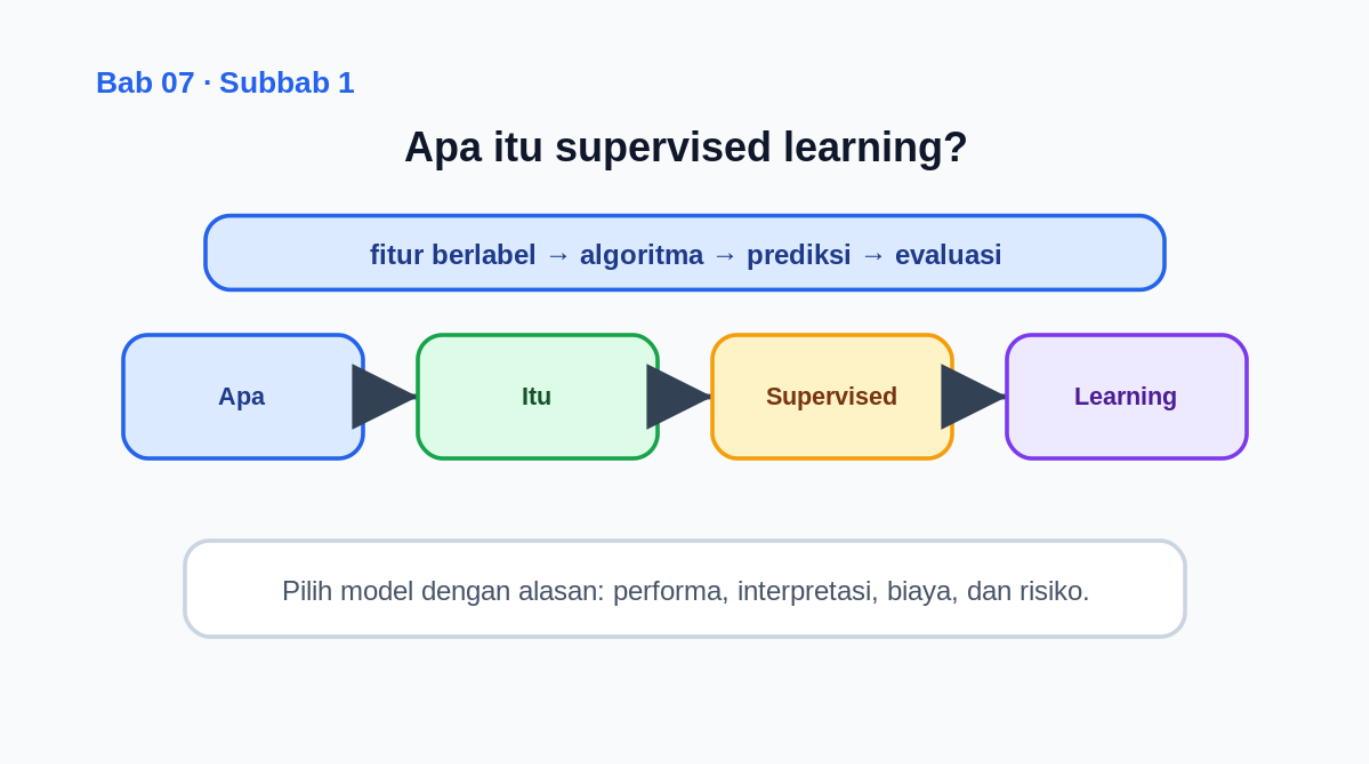
Cara membaca bab ini

Bab ini tetap memakai struktur subbab, bukan label halaman. Tujuannya agar setiap konsep dibahas sepanjang yang diperlukan. Supervised learning punya banyak algoritma, tetapi jangan dibaca sebagai daftar hafalan. Bacalah sebagai peta karakter: model mana yang sederhana, mana yang probabilistik, mana yang berbasis jarak, mana yang berbasis aturan, mana yang ensemble, dan mana yang mencari margin.

Janji Bab 7: setelah membaca bab ini, pembaca bisa menjelaskan perbedaan regresi dan klasifikasi, memahami intuisi beberapa algoritma utama, menjalankan perbandingan model kecil, dan memilih model dengan alasan yang lebih dewasa daripada “yang paling canggih”.

Motivasi:
Algoritma supervised learning seperti alat di bengkel. Tidak semua pekerjaan butuh bor terbesar. Praktisi yang baik tahu kapan memakai obeng sederhana, kapan memakai bor, dan kapan berhenti untuk mengukur ulang.

Subbab 1 — Apa itu supervised learning?



Apa itu supervised learning?

Supervised learning adalah keluarga metode ML yang belajar dari contoh berlabel. Setiap contoh punya input X dan jawaban historis y. Model belajar hubungan antara X dan y agar bisa membuat prediksi pada contoh baru. Jika Bab 6 membangun kebiasaan kerja ML yang jujur, Bab 7 mulai membahas alat-alat utama untuk membuat prediksi: regresi, klasifikasi, kNN, Naive Bayes, logistic

regression, decision tree, random forest, boosting, dan SVM.

Contoh Indonesia: koperasi ingin memprediksi apakah pengajuan pinjaman perlu review tambahan; sekolah ingin memprediksi siswa yang butuh bantuan; warung ingin memperkirakan jumlah penjualan besok. Selama ada contoh masa lalu dan target yang jelas, supervised learning bisa membantu. Tetapi kata “membantu” penting: model bukan pengganti pemahaman domain dan tanggung jawab manusia.

Inti Bab 7 bukan menghafal nama algoritma. Intinya memahami karakter tiap model: apa asumsi dasarnya, kapan cocok, kapan berbahaya, bagaimana membaca error, dan bagaimana memilih model sederhana sebelum model kompleks.

Ide teknis / latihan kecil

> Supervised learning: belajar dari pasangan (fitur, label).
> X = fitur, y = jawaban historis, model = fungsi prediksi.

Pendalaman sejarah dan makna teknis

Akar supervised learning dapat ditelusuri ke statistik, teori keputusan, pattern recognition, dan cybernetics. Regresi linear sudah dipakai jauh sebelum istilah AI populer. Naive Bayes berakar pada teorema Bayes abad ke-18. Perceptron muncul pada 1950-an sebagai model awal jaringan saraf linear. Decision tree berkembang sebagai cara membuat aturan prediktif yang dapat dibaca manusia. SVM populer pada 1990-an karena teori margin dan kernel. Random forest dan boosting makin kuat ketika komputer mampu melatih banyak model sekaligus.

Sejarah ini penting karena menunjukkan bahwa supervised learning bukan satu algoritma tunggal. Ia adalah keluarga cara berpikir: sebagian berasal dari statistik, sebagian dari geometri, sebagian dari pencarian aturan, sebagian dari ensemble. Jika pembaca memahami asal idenya, nama algoritma tidak terasa seperti daftar asing.

Persamaan dasar supervised learning

> Dataset latih: $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$
> Model: $\hat{y} = f(x; \theta)$
> Tujuan: pilih θ agar loss rata-rata kecil pada data baru

x adalah fitur, y adalah label/target, θ adalah parameter model. Training mencari parameter yang membuat prediksi ■ mendekati y. Generalisasi berarti model tidak hanya bagus di data latih, tetapi juga pada contoh baru.

Contoh hitung terstruktur

Misalkan ada tiga data latihan:

x: jam belajar	y: lulus?
1	0
3	1
4	1

Aturan manual sederhana: jika jam belajar ≥ 2, prediksi lulus. Prediksi untuk x=1 adalah 0, x=3 adalah 1, x=4 adalah 1. Semua benar pada data kecil ini. Tetapi supervised learning bertanya lebih jauh: apakah ambang 2 tetap bagus untuk data baru? Jika ada siswa belajar 2 jam tetapi tidak lulus karena faktor lain, model harus diuji ulang. Inilah alasan evaluasi jujur lebih penting daripada cocok pada contoh kecil.

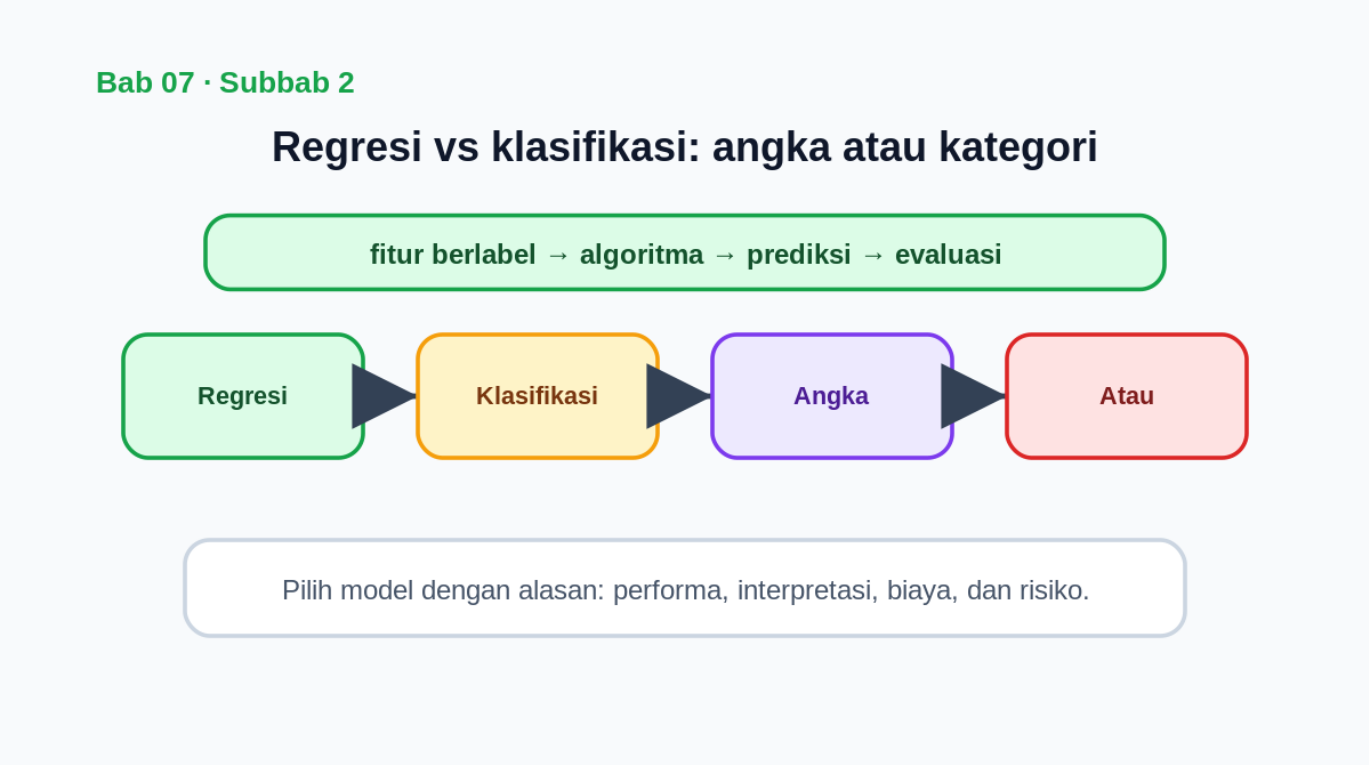
Tes cepat subbab 1

1. Apa definisi supervised learning dengan notasi X dan y?

2. Mengapa sejarah supervised learning berasal dari beberapa tradisi, bukan satu algoritma?

3. Buat dataset 3 baris untuk kasus sekolah atau UMKM.

Subbab 2 — Regresi vs klasifikasi: angka atau kategori



Regresi vs klasifikasi: angka atau kategori

Dua tugas supervised learning paling umum adalah regresi dan klasifikasi. Regresi memprediksi angka: penjualan, harga, waktu tiba, nilai ujian, atau jumlah permintaan. Klasifikasi memprediksi kategori: spam/tidak, lulus/tidak, risiko rendah/sedang/tinggi.

Kesalahan memilih bentuk target dapat merusak proyek. Jika target stok sebenarnya angka, mengubahnya menjadi “tinggi/rendah” mungkin membuang informasi. Sebaliknya, jika keputusan operasional hanya butuh prioritas “perlu review/tidak”, klasifikasi mungkin lebih praktis daripada regresi yang terlalu detail.

Pertanyaan yang harus diajukan: keputusan akhirnya apa? Jika keputusan membutuhkan jumlah, gunakan regresi. Jika membutuhkan pilihan kelas, gunakan klasifikasi. Jika membutuhkan urutan tindakan, mungkin ranking lebih tepat. Model dipilih mengikuti keputusan, bukan mengikuti tren algoritma.

Ide teknis / latihan kecil

```
# regresi: prediksi angka
prediksi_penjualan = 42
# klasifikasi: prediksi kategori
perlu_review = True
```

Pendalaman matematika regresi dan klasifikasi

Regresi biasanya memakai loss berbasis jarak angka. Untuk satu data, squared error adalah (■ - y)². Untuk banyak data, MSE adalah rata-ratanya. Klasifikasi memakai loss yang menghukum probabilitas kelas benar yang rendah, misalnya cross-entropy.

```
Regresi:     y ∈ bilangan nyata, contoh y = 42 gelas
Klasifikasi: y ∈ kelas, contoh y ∈ {beli, tidak beli}
MSE:         (1/n) Σ(ŷᵢ - yᵢ)²
```

Contoh hitung regresi

Aktual penjualan tiga hari: `[40, 50, 60]`. Prediksi model: `[42, 47, 55]`.

```
error = [2, -3, -5]
absolute error = [2, 3, 5]
MAE = (2 + 3 + 5) / 3 = 3,33
```

```
squared error = [4, 9, 25]
MSE = (4 + 9 + 25) / 3 = 12,67
```

Contoh hitung klasifikasi

Empat prediksi beli/tidak: aktual `[1, 0, 1, 1]`, prediksi `[1, 0, 0, 1]`. Benar pada data ke-1, ke-2, ke-4; salah pada data ke-3. Accuracy = 3/4 = 0,75. Tetapi jika data positif penting, kesalahan data ke-3 adalah false negative dan perlu dibaca lebih serius.

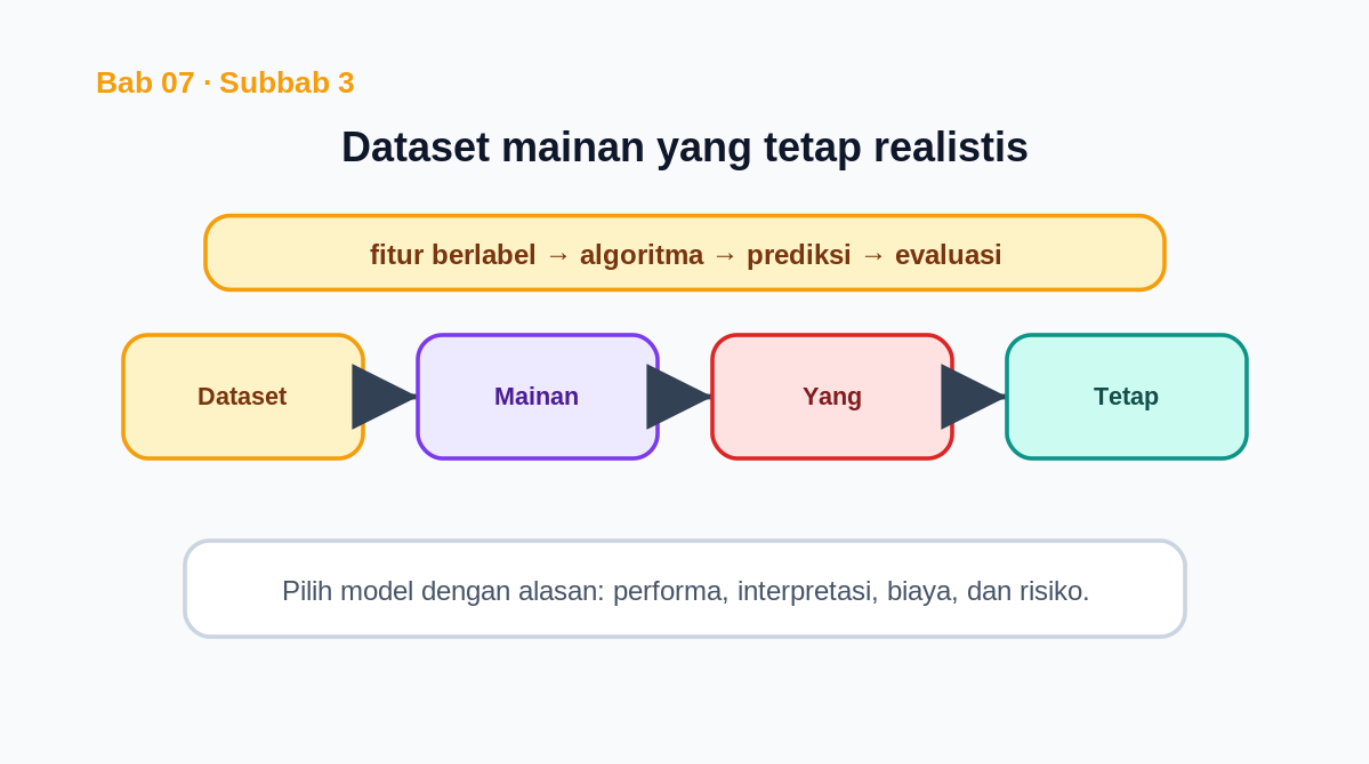
Tes cepat subbab 2

1. Hitung MAE untuk aktual [10,20] dan prediksi [12,15].

2. Kapan regresi lebih tepat daripada klasifikasi?

3. Buat contoh false negative pada klasifikasi layanan publik.

Subbab 3 — Dataset mainan yang tetap realistis



Dataset mainan yang tetap realistis

Untuk memahami supervised learning, kita memakai dataset kecil yang bisa dibaca manusia. Dataset praktikum Bab 7 berisi contoh pelanggan/produk dengan fitur seperti harga relatif, rating, diskon, jumlah klik, dan apakah produk dibeli. Untuk regresi, targetnya bisa jumlah permintaan. Untuk klasifikasi, targetnya beli/tidak.

Dataset kecil bukan berarti main-main. Justru dataset kecil membuat pembaca bisa melihat setiap baris dan memahami mengapa model salah. Dalam proyek besar, prinsipnya sama: baris adalah contoh, fitur adalah informasi, label adalah jawaban historis.

Kita sengaja memakai Python standard library agar fokus pada konsep. Setelah paham, pembaca akan lebih siap memakai scikit-learn di proyek nyata. Library mempercepat kerja, tetapi pemahaman dasar membuat kita tidak buta saat hasil model aneh.

Ide teknis / latihan kecil

```
Contoh fitur: harga_rel, rating, diskon, klik
Label klasifikasi: beli/tidak
Target regresi: jumlah permintaan
```

Kenapa dataset kecil tetap berguna untuk belajar?

Dalam pembelajaran teknis, dataset kecil seperti mikroskop. Kita bisa melihat setiap baris dan memahami mengapa model salah. Dataset besar lebih realistis, tetapi sering membuat pemula hanya menekan `fit()` tanpa memahami mekanisme.

Contoh dataset mini

rating	diskon	klik	harga_rel	beli
4.8	0.30	10	0.40	1
3.2	0.05	1	0.90	0
4.1	0.20	6	0.50	1
3.8	0.00	2	0.80	0

Sebelum modeling, baca tabel ini secara manusiawi. Rating tinggi membantu. Diskon membantu. Klik tinggi memberi sinyal minat. Harga relatif tinggi mungkin menurunkan peluang beli. Model supervised learning mencoba mengubah intuisi ini menjadi aturan atau bobot.

Latihan hitung kecil

Jika kita membuat skor manual:

$$\text{skor} = \text{rating} + 5\times\text{diskon} + 0{,}2\times\text{klik} - 2\times\text{harga\_rel}$$

Untuk baris pertama: $4{,}8 + 5\times0{,}30 + 0{,}2\times10 - 2\times0{,}40 = 4{,}8 + 1{,}5 + 2 - 0{,}8 = 7{,}5$. Untuk baris kedua: $3{,}2 + 0{,}25 + 0{,}2 - 1{,}8 = 1{,}85$. Aturan threshold 4 akan memprediksi baris pertama beli dan baris kedua tidak beli. Ini bukan model final, tetapi membantu memahami ide skor.

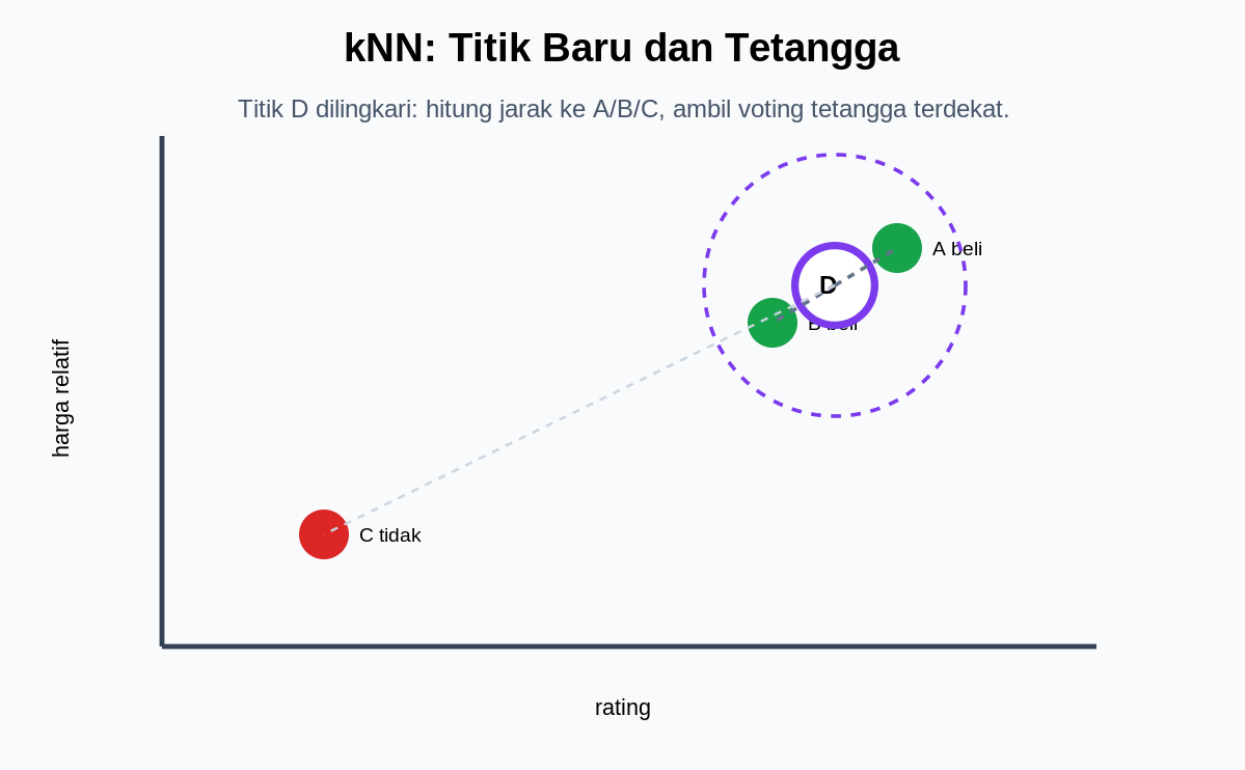
Tes cepat subbab 3

1. Hitung skor manual untuk satu produk memakai rumus pada subbab.

2. Mengapa dataset kecil membantu belajar mekanisme?

3. Sebutkan fitur yang mungkin bocor dalam dataset pembelian.

Subbab 4 — k-Nearest Neighbors: belajar dari tetangga terdekat



k-Nearest Neighbors: belajar dari tetangga terdekat

kNN adalah algoritma yang sangat intuitif: untuk memprediksi contoh baru, lihat k contoh terdekat dalam data latih, lalu ambil voting atau rata-rata. Jika pelanggan baru mirip dengan pelanggan yang membeli, prediksi kemungkinan beli. Jika rumah mirip dengan rumah mahal, prediksi harga juga tinggi.

kNN hampir tidak punya proses training berat. Ia menyimpan data, lalu bekerja saat prediksi. Kelebihannya mudah dipahami dan bisa menangkap pola lokal. Kekurangannya lambat pada data besar, sensitif terhadap skala fitur, dan bisa buruk jika fitur tidak relevan.

Karena kNN memakai jarak, scaling penting. Fitur “harga dalam rupiah” bisa mendominasi fitur “rating 1–5” jika tidak dinormalisasi. Ini menghubungkan Bab 4 tentang jarak vektor dengan Bab 7: geometri data memengaruhi prediksi.

Ide teknis / latihan kecil

> Prediksi kNN = lihat k tetangga terdekat → voting/rata-rata.
> Jarak umum: Euclidean distance.

Cara membaca gambar/graph kNN

Pada gambar kNN, titik-titik data dapat dibaca sebagai graph geometris sederhana. Setiap titik adalah contoh data. Titik baru diberi lingkaran tebal agar pembaca tahu mana yang sedang diprediksi. Garis putus-putus dari titik baru ke tetangga terdekat menunjukkan jarak yang sedang dihitung. Lingkaran radius di sekitar titik baru menunjukkan “wilayah pencarian tetangga”.

Urutan membaca gambar:

1. Temukan titik baru yang dilingkari.

2. Lihat tiga garis terpendek menuju titik latih.

3. Baca label warna tetangga: beli/tidak.

4. Lakukan voting.

Dengan cara ini, pembaca tidak hanya tahu rumus jarak, tetapi juga melihat keputusan kNN sebagai proses memilih tetangga pada ruang fitur.

Contoh hitung kNN secara manual

Misalkan fitur hanya dua: rating dan harga_rel setelah diskalakan sederhana. Data latih:

Produk	rating	harga_rel	beli
A	5	0,2	1
B	4	0,4	1
C	2	0,9	0

Produk baru D punya rating 4,5 dan harga_rel 0,3. Hitung jarak Euclidean:

```
jarak(D,A) = sqrt((4,5-5)² + (0,3-0,2)²)
           = sqrt(0,25 + 0,01) = sqrt(0,26) ≈ 0,51
jarak(D,B) = sqrt((4,5-4)² + (0,3-0,4)²)
           = sqrt(0,25 + 0,01) ≈ 0,51
jarak(D,C) = sqrt((4,5-2)² + (0,3-0,9)²)
           = sqrt(6,25 + 0,36) ≈ 2,57
```

Jika k=1, tetangga terdekat bisa A atau B, keduanya beli. Jika k=3, voting `[1,1,0]` menghasilkan prediksi beli. Dari contoh ini, pembaca melihat bahwa kNN tidak “belajar bobot”, tetapi menyimpan contoh dan memakai kedekatan.

Catatan skala fitur

Jika rating 1–5 tetapi harga ditulis dalam rupiah jutaan, jarak bisa didominasi harga. Karena itu, scaling bukan detail kosmetik; scaling menentukan siapa yang dianggap “tetangga”.

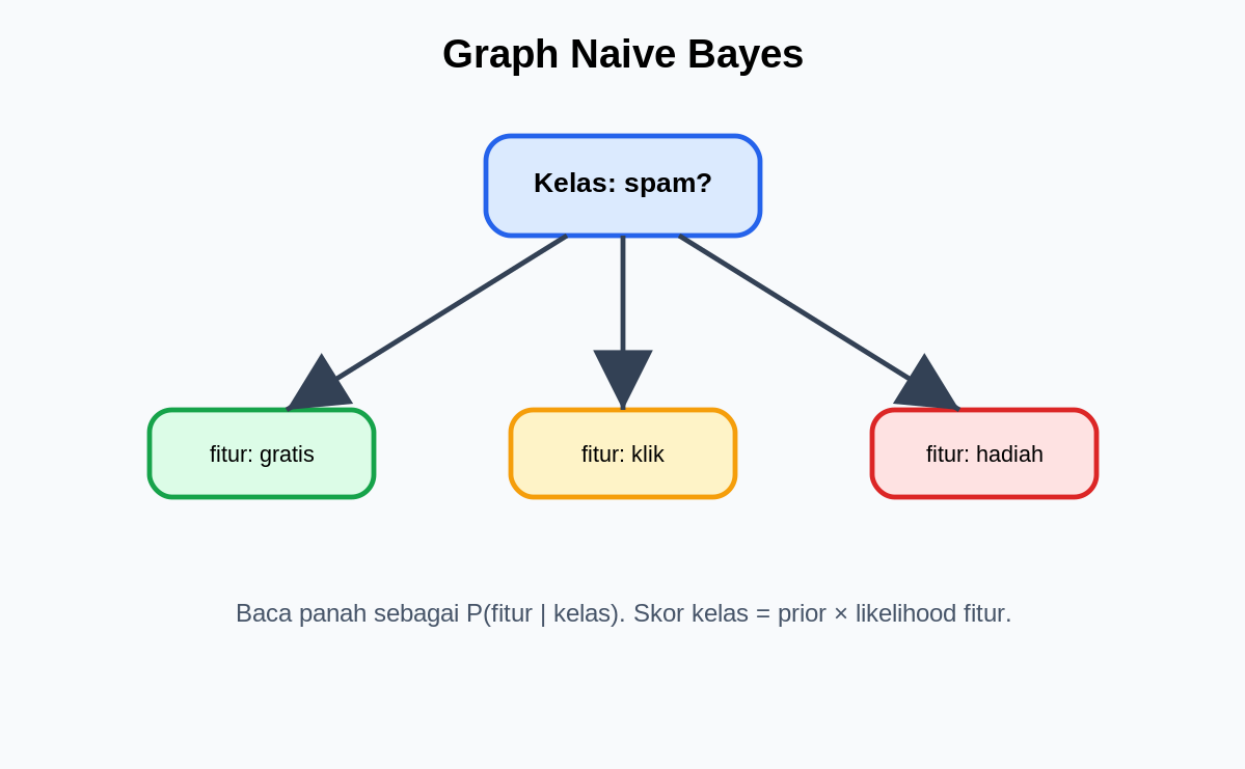
Tes cepat subbab 4

1. Hitung jarak Euclidean dua titik: A=(1,2), B=(4,6).

2. Mengapa scaling penting untuk kNN?

3. Apa efek k terlalu kecil dan k terlalu besar?

Subbab 5 — Naive Bayes: probabilitas sederhana yang sering kuat



Naive Bayes: probabilitas sederhana yang sering kuat

Naive Bayes memakai probabilitas untuk menghitung kelas paling mungkin berdasarkan fitur. Kata “naive” muncul karena model mengasumsikan fitur saling independen jika kelas diketahui. Asumsi ini sering tidak benar sempurna, tetapi model tetap sering berguna, terutama untuk teks, spam, dan baseline probabilistik.

Bayangkan klasifikasi pesan spam. Kata “gratis”, “hadiah”, dan “klik” mungkin meningkatkan peluang spam. Naive Bayes menggabungkan bukti dari kata-kata itu secara sederhana. Ia cepat, mudah, dan bekerja baik pada banyak fitur diskret/teks.

Kelemahannya: jika fitur sangat saling bergantung, probabilitas bisa terlalu percaya diri. Smoothing juga penting agar fitur yang belum pernah muncul tidak membuat peluang menjadi nol.

Ide teknis / latihan kecil

$$P(\text{kelas} \mid \text{fitur}) \propto P(\text{kelas}) \times P(\text{fitur1}|\text{kelas}) \times P(\text{fitur2}|\text{kelas}) \times \ldots$$

Cara membaca graph Naive Bayes

Graph Naive Bayes biasanya digambar sebagai satu node kelas di atas dan beberapa node fitur di bawah. Panah dari kelas menuju fitur berarti probabilitas fitur dihitung dengan syarat kelas: `P(fitur | kelas)`. Ini bukan berarti kelas secara fisik menyebabkan kata muncul, tetapi cara model menyusun perhitungan.

Urutan membaca gambar:

1. Mulai dari node kelas: spam atau bukan spam.

2. Turun ke node fitur: “gratis”, “klik”, dan fitur lain.

3. Ambil peluang setiap fitur bersyarat pada kelas.

4. Kalikan prior kelas dan likelihood fitur.

5. Bandingkan skor antar kelas.

Gambar ini membantu pembaca memahami kenapa model disebut probabilistik: setiap jalur kelas → fitur menyumbang bukti numerik.

Contoh hitung Naive Bayes dengan tabel kecil

Misalkan kita klasifikasi pesan promo sebagai spam/tidak. Fitur: apakah ada kata “gratis” dan “klik”. Data latih ringkas:

kelas	jumlah pesan	gratis muncul	klik muncul
spam	10	8	7
bukan spam	20	2	3

Prior:

$$P(\text{spam}) = 10/30 = 0{,}333$$
$$P(\text{bukan}) = 20/30 = 0{,}667$$

Untuk pesan baru berisi “gratis” dan “klik”:

$$\text{skor\_spam} \propto P(\text{spam}) \times P(\text{gratis}|\text{spam}) \times P(\text{klik}|\text{spam})$$
$$= 0{,}333 \times 8/10 \times 7/10$$
$$= 0{,}333 \times 0{,}8 \times 0{,}7 = 0{,}186$$

$$\text{skor\_bukan} \propto P(\text{bukan}) \times P(\text{gratis}|\text{bukan}) \times P(\text{klik}|\text{bukan})$$
$$= 0{,}667 \times 2/20 \times 3/20$$
$$= 0{,}667 \times 0{,}1 \times 0{,}15 = 0{,}010$$

Karena skor spam lebih besar, prediksi spam. Dalam praktik, kita memakai smoothing agar peluang tidak menjadi nol ketika kata belum pernah muncul di kelas tertentu.

Turunan makna

Naive Bayes bukan menebak dari satu kata. Ia menggabungkan bukti. Namun asumsi independensi berarti ia memperlakukan “gratis” dan “klik” seolah bukti terpisah setelah kelas diketahui. Ini sering berguna, tetapi bisa terlalu percaya diri jika fitur sangat berkorelasi.

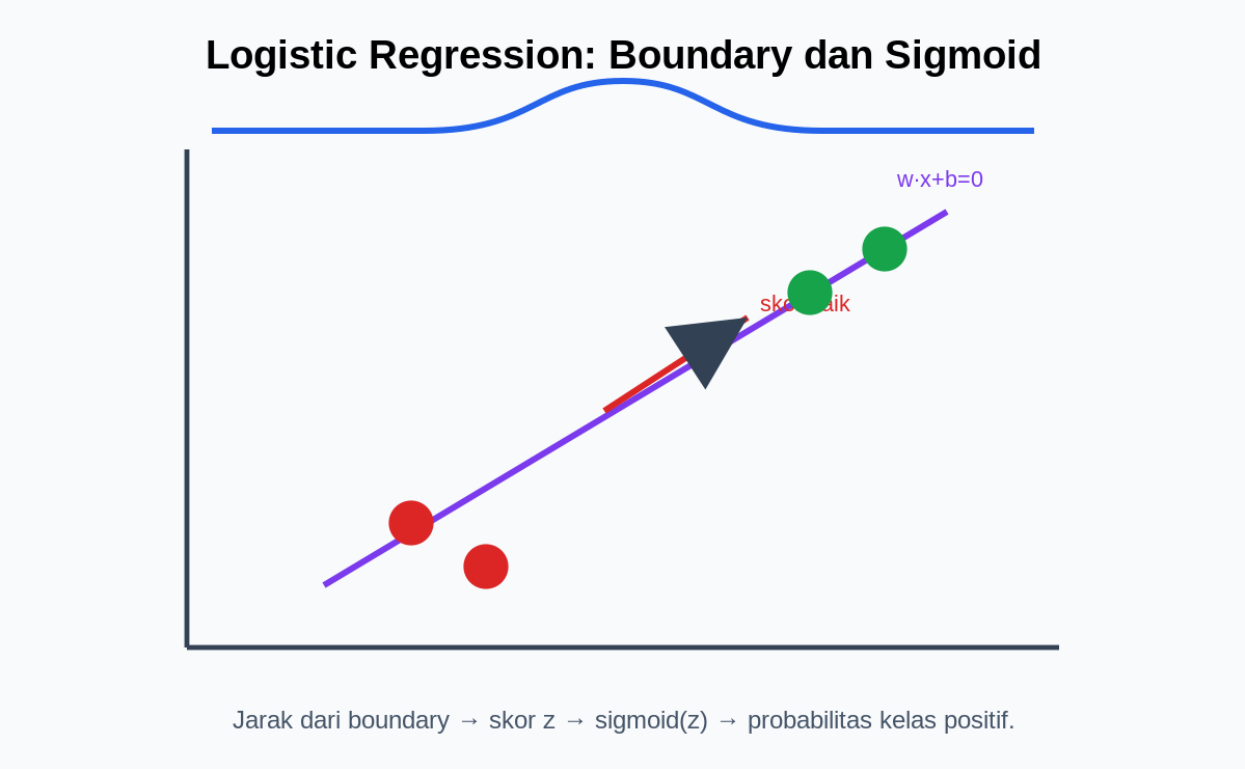
Tes cepat subbab 5

1. Hitung skor Naive Bayes sederhana untuk dua kelas dengan prior berbeda.

2. Apa arti asumsi naive?

3. Mengapa smoothing diperlukan?

Subbab 6 — Logistic regression: garis keputusan dengan probabilitas



Logistic regression: garis keputusan dengan probabilitas

Logistic regression adalah model klasifikasi linear. Meskipun namanya regression, ia dipakai untuk klasifikasi. Model menghitung skor linear dari fitur, lalu melewatkannya ke fungsi sigmoid agar menjadi probabilitas 0 sampai 1.

Intuisinya dekat dengan Bab 4 dan 5: skor = w·x + b. Jika skor besar, probabilitas kelas positif tinggi. Jika skor kecil, probabilitas rendah. Training mencari bobot w dan bias b yang membuat probabilitas cocok dengan label.

Kelebihan logistic regression: relatif mudah dijelaskan, cepat, dan menjadi baseline kuat. Kekurangannya: batas keputusan dasarnya linear. Jika pola data sangat melengkung atau interaksi fitur rumit, ia perlu fitur tambahan atau model lain.

Ide teknis / latihan kecil

```
skor = w·x + b
probabilitas = sigmoid(skor) = 1 / (1 + e^-skor)
```

Cara membaca gambar logistic regression

Pada gambar logistic regression, garis lurus adalah decision boundary: semua titik di garis itu memiliki `w·x+b=0` dan probabilitas sekitar 0,5. Titik di satu sisi garis punya skor positif dan cenderung diprediksi kelas 1. Titik di sisi lain punya skor negatif dan cenderung kelas 0.

Panah tegak lurus garis menunjukkan arah kenaikan skor. Semakin jauh titik bergerak ke arah panah, semakin besar z, dan sigmoid mengubahnya menjadi probabilitas makin dekat ke 1. Sebaliknya, semakin jauh ke arah berlawanan, probabilitas makin dekat ke 0.

Urutan membaca gambar:

1. Temukan garis batas keputusan.

2. Lihat posisi titik terhadap garis.

3. Baca arah panah skor.

4. Hubungkan jarak dari garis dengan keyakinan model.

Logistic regression: dari skor linear ke probabilitas

Mulai dari skor:

$$z = w_1x_1 + w_2x_2 + \ldots + b$$
$$p = \text{sigmoid}(z) = 1 / (1 + e^{-z})$$

Jika $z=0$, maka $p=0,5$. Jika $z=2$, $p\approx0,88$. Jika $z=-2$, $p\approx0,12$. Jadi bobot linear diubah menjadi probabilitas.

Contoh hitung prediksi

Misalkan model punya:

$$z = 1,2\times\text{rating} + 2,0\times\text{diskon} + 0,3\times\text{klik} - 1,5\times\text{harga\_rel} - 5$$

Untuk produk rating 4,5, diskon 0,2, klik 6, harga_rel 0,4:

$$z = 1,2\times4,5 + 2\times0,2 + 0,3\times6 - 1,5\times0,4 - 5$$
$$= 5,4 + 0,4 + 1,8 - 0,6 - 5 = 2,0$$
$$p = \text{sigmoid}(2,0) \approx 0,88$$

Jika threshold 0,5, prediksi beli.

Sedikit turunan gradient

Untuk logistic regression dengan binary cross-entropy:

$$\text{Loss} = -[y \log(p) + (1-y) \log(1-p)]$$
$$p = \text{sigmoid}(w \cdot x + b)$$

Hasil turunan pentingnya sangat elegan:

$$\partial\text{Loss}/\partial w_j = (p - y)\, x_j$$
$$\partial\text{Loss}/\partial b = (p - y)$$

Artinya jika model memprediksi p terlalu besar dibanding y, bobot untuk fitur aktif akan diturunkan. Jika p terlalu kecil, bobot akan dinaikkan. Ini menyambung langsung ke Bab 5 tentang gradient.

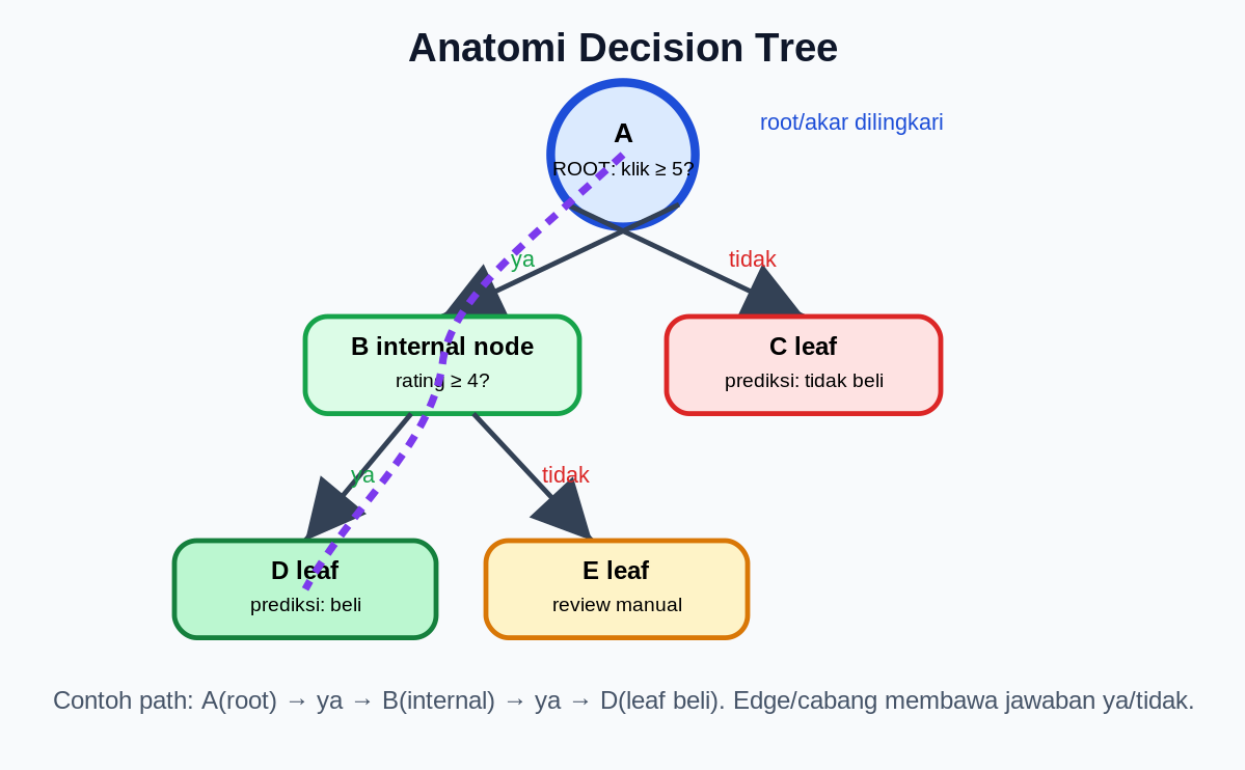
Tes cepat subbab 6

1. Hitung sigmoid(0), sigmoid(2) secara perkiraan.

2. Apa makna gradient (p-y)x pada logistic regression?

3. Mengapa logistic regression tetap disebut model linear?

Subbab 7 — Decision tree: serangkaian pertanyaan ya/tidak



Decision tree: serangkaian pertanyaan ya/tidak

Decision tree memprediksi dengan membuat pertanyaan bertingkat. Contoh: “apakah klik > 5?”, “apakah diskon > 20%?”, “apakah rating > 4?”. Setiap jawaban membawa data ke cabang berikutnya sampai daun yang berisi prediksi.

Kelebihannya: mudah divisualisasikan dan cocok untuk campuran fitur. Pohon bisa menangkap interaksi non-linear tanpa banyak preprocessing. Kekurangannya: pohon tunggal mudah overfit jika terlalu dalam. Ia bisa menghafal detail data latih.

Pohon keputusan mengajarkan ide penting: model bisa menjadi kumpulan aturan yang dipelajari dari data. Tetapi aturan yang terlalu spesifik harus dicurigai. Batasi kedalaman, minimum sampel, atau gunakan ensemble.

Ide teknis / latihan kecil

Jika klik tinggi dan rating tinggi → kemungkinan beli.
Jika klik rendah dan harga tinggi → kemungkinan tidak beli.

Anatomi graph decision tree: root, node, edge, leaf, dan path

Decision tree adalah graph berbentuk pohon. Ia punya bagian-bagian penting:

• Root node / akar: node paling atas; pertanyaan pertama. Di gambar, root diberi label A dan dilingkari tebal karena semua contoh masuk dari sini.

• Internal node: node tengah yang masih berisi pertanyaan lanjutan.

• Edge / cabang: garis penghubung antar node, biasanya diberi label jawaban seperti “ya” atau “tidak”.

• Leaf / daun: node akhir yang berisi prediksi.

• Path / jalur keputusan: rute dari root sampai leaf untuk satu contoh.

Kenapa mulai dari root A yang dilingkari? Karena A adalah pertanyaan pertama yang dipilih model untuk memecah data. Model memilih root yang paling mengurangi impurity, misalnya pertanyaan “klik ≥ 5?”. Semua data harus melewati pertanyaan ini sebelum menuju cabang berikutnya.

Contoh membaca path

Misalkan root A bertanya: `klik ≥ 5`?

• Produk X punya klik=7 → jawab ya, bergerak ke node B.

• Node B bertanya: `rating ≥ 4`?

• Produk X punya rating=4,3 → jawab ya, bergerak ke leaf D.

• Leaf D berisi prediksi: beli.

Jadi jalurnya: `A(root) → ya → B(internal) → ya → D(leaf beli)`. Inilah alasan decision tree mudah dijelaskan: prediksi bisa dibaca sebagai rute keputusan.

Decision tree dan ukuran impurity

Pohon memilih pertanyaan yang membuat data setelah split menjadi lebih “murni”. Salah satu ukuran impurity adalah Gini:

$$Gini = 1 - \Sigma\, p\_k^2$$

Untuk klasifikasi biner dengan proporsi positif p dan negatif 1-p:

$$Gini = 1 - p^2 - (1-p)^2$$

Jika node berisi 5 positif dan 5 negatif, p=0,5:

$$Gini = 1 - 0{,}25 - 0{,}25 = 0{,}5$$

Jika node berisi 9 positif dan 1 negatif, p=0,9:

$$Gini = 1 - 0{,}81 - 0{,}01 = 0{,}18$$

Node kedua lebih murni.

Contoh split

Misalkan 10 data dibagi oleh pertanyaan “klik ≥ 5”. Sebelum split: 5 positif, 5 negatif, Gini=0,5. Setelah split:

• kiri: 4 positif, 1 negatif → $Gini = 1 - (4/5)^2 - (1/5)^2 = 0{,}32$

• kanan: 1 positif, 4 negatif → $Gini = 0{,}32$

Weighted Gini = `5/10×0,32 + 5/10×0,32 = 0,32`. Penurunan impurity = `0,5 - 0,32 = 0,18`. Pohon memilih split dengan penurunan impurity terbesar.

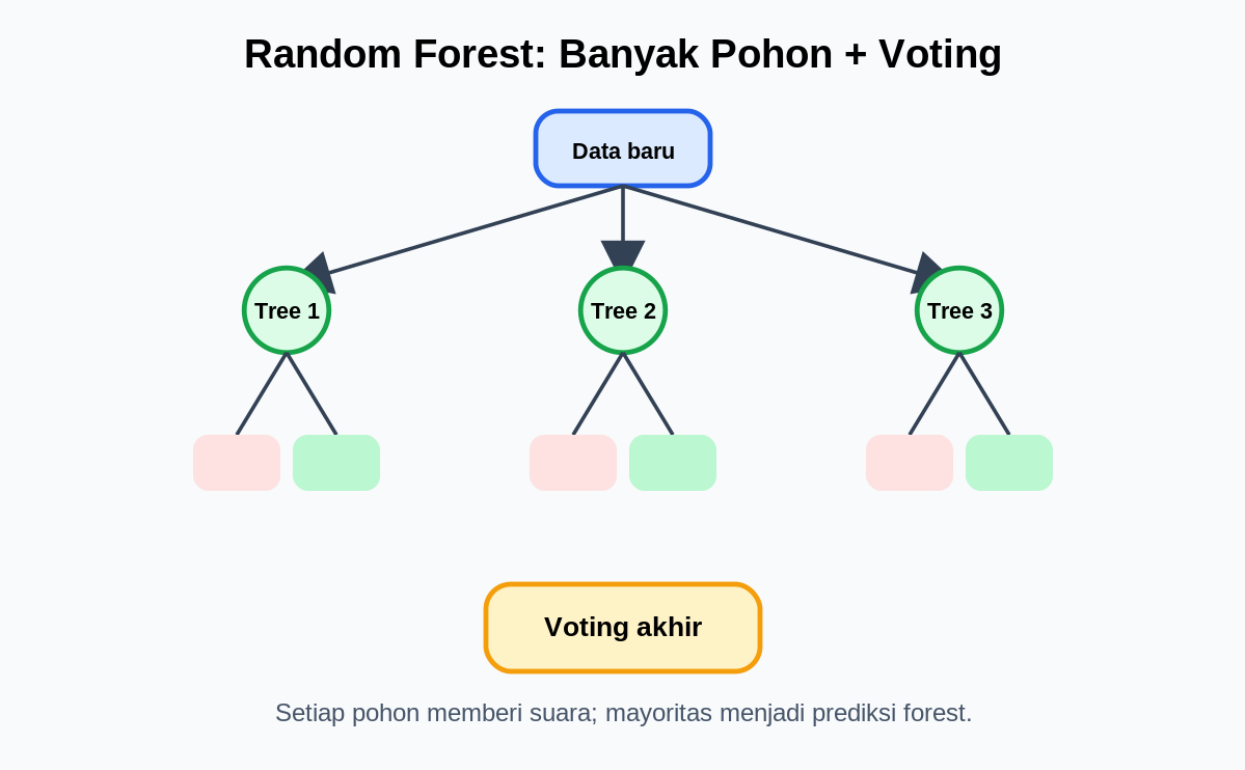
Tes cepat subbab 7

1. Hitung Gini untuk node berisi 3 positif dan 1 negatif.

2. Mengapa pohon terlalu dalam bisa overfit?

3. Buat satu aturan decision tree untuk kasus produk.

Subbab 8 — Random forest: banyak pohon agar lebih stabil



Random forest: banyak pohon agar lebih stabil

Random forest membuat banyak decision tree, masing-masing dilatih pada sampel/fitur yang agak berbeda, lalu menggabungkan prediksi. Ide utamanya: satu pohon bisa terlalu percaya diri, tetapi banyak pohon yang berbeda dapat saling menstabilkan.

Random forest mengurangi variance dibanding pohon tunggal. Ia sering menjadi model tabular yang kuat dan relatif mudah dipakai. Kelemahannya: lebih sulit dijelaskan daripada satu pohon, lebih berat, dan tidak selalu terbaik untuk data sangat besar atau pola tertentu.

Dalam praktikum standard library, kita tidak membangun random forest penuh. Kita membahas idenya dan memakai beberapa stump sederhana sebagai gambaran ensemble. Tujuannya agar pembaca memahami mengapa “banyak model sederhana” bisa lebih stabil.

Ide teknis / latihan kecil

```
Random forest = banyak pohon + voting/rata-rata.
Tujuan: mengurangi overfitting pohon tunggal.
```

Cara membaca graph random forest

Gambar random forest menampilkan beberapa pohon kecil berdampingan. Setiap pohon punya root sendiri, node sendiri, dan leaf sendiri. Satu data baru dimasukkan ke semua pohon. Setiap pohon memberi suara. Suara-suara itu masuk ke kotak voting akhir.

Urutan membaca gambar:

1. Data baru masuk ke Tree 1, Tree 2, Tree 3, dan seterusnya.

2. Setiap tree menghasilkan prediksi leaf.

3. Prediksi dikumpulkan.

4. Mayoritas vote menjadi prediksi forest.

Jadi random forest bukan satu pohon raksasa, tetapi kumpulan graph pohon yang masing-masing agak berbeda.

Mengapa random forest mengurangi variance?

Satu decision tree mudah berubah jika data latih sedikit berubah. Random forest melatih banyak pohon pada bootstrap sample dan subset fitur, lalu voting. Jika kesalahan antar pohon tidak sepenuhnya sama, voting dapat mengurangi fluktuasi.

Contoh voting

Lima pohon memprediksi: `[1, 1, 0, 1, 0]`. Jumlah suara positif = 3, negatif = 2, maka prediksi akhir = 1.

Jika satu pohon salah karena terlalu mengikuti noise, empat pohon lain bisa menahan keputusan. Ini mirip bertanya ke beberapa ahli yang melihat bukti dari sudut sedikit berbeda.

Catatan matematis ringan

Jika kita merata-ratakan banyak prediksi yang error-nya tidak berkorelasi sempurna, variance rata-rata bisa turun. Namun jika semua pohon terlalu mirip, manfaat ensemble berkurang. Karena itu random forest memperkenalkan randomness: sampel berbeda dan fitur berbeda.

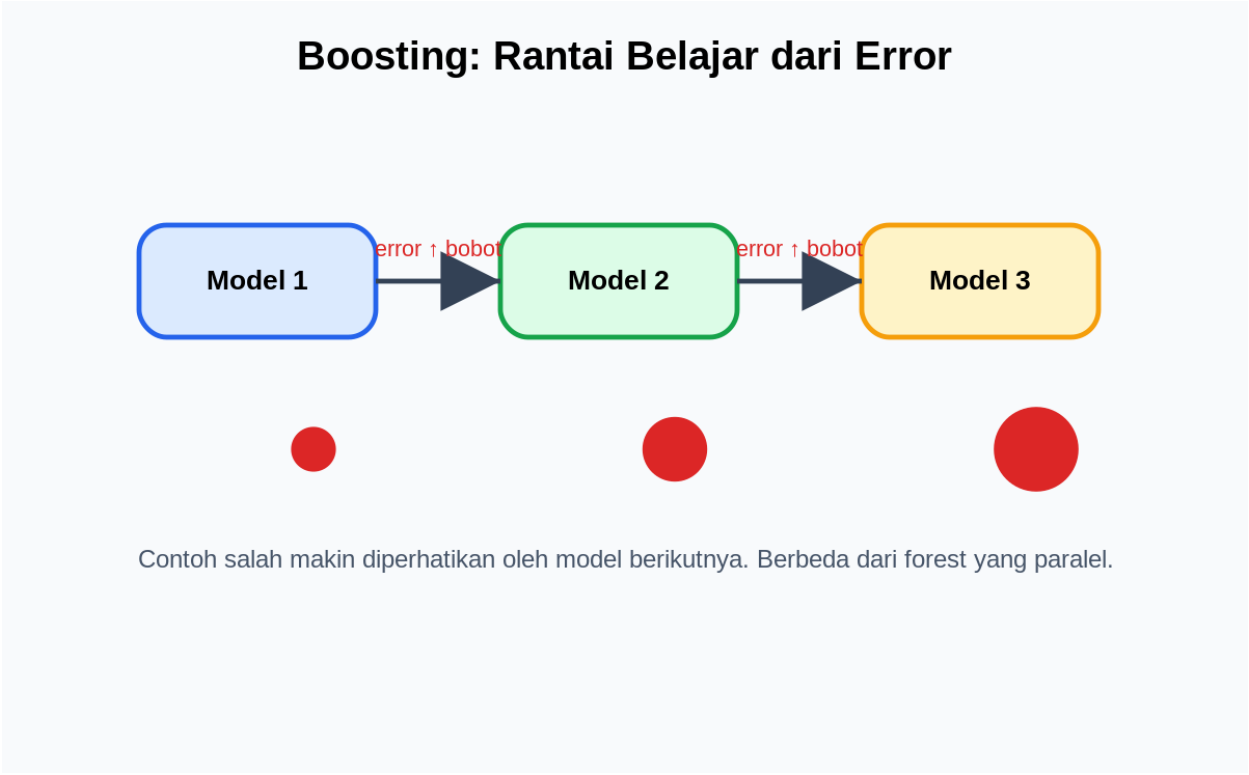
Tes cepat subbab 8

1. Lakukan voting dari prediksi [1,0,1,1,0].

2. Mengapa random forest lebih stabil daripada satu tree?

3. Kapan random forest kurang cocok?

Subbab 9 — Boosting: belajar dari kesalahan sebelumnya



Boosting: belajar dari kesalahan sebelumnya

Boosting juga ensemble, tetapi logikanya berbeda dari random forest. Model dibangun berurutan. Model berikutnya memberi perhatian lebih pada contoh yang sebelumnya salah. Dengan cara ini, kumpulan weak learners bisa menjadi model kuat.

Analogi belajar: setelah latihan pertama, guru melihat soal mana yang sering salah, lalu latihan berikutnya fokus pada kelemahan itu. Boosting sering sangat kuat pada data tabular. Namun ia juga bisa sensitif terhadap noise dan outlier jika terlalu agresif.

Algoritma modern seperti Gradient Boosting, XGBoost, LightGBM, dan CatBoost sangat populer. Bab ini hanya memberi fondasi intuisi agar nanti nama-nama itu tidak terasa seperti sihir.

Ide teknis / latihan kecil

```
Boosting = model 1 salah → model 2 fokus ke error → model 3 lanjut memperbaiki.
```

Cara membaca graph boosting

Boosting digambar sebagai rantai model: model pertama → error → model kedua → error → model ketiga. Panah antar model menunjukkan bahwa model berikutnya belajar dari kesalahan model sebelumnya. Contoh yang salah diberi bobot lebih besar, sehingga lebih “terlihat” pada iterasi berikutnya.

Urutan membaca gambar:

1. Model 1 membuat prediksi awal.

2. Contoh yang salah ditandai merah.

3. Bobot contoh salah dinaikkan.

4. Model 2 dilatih dengan perhatian lebih pada contoh merah.

5. Prediksi akhir menggabungkan suara model-model tersebut.

Graph ini berbeda dari random forest: random forest paralel, boosting berurutan.

Boosting dengan contoh bobot sederhana

Boosting memberi perhatian lebih pada contoh yang salah. Contoh sangat sederhana: ada 5 data dengan bobot awal sama, masing-masing 0,2. Model pertama salah pada data ke-2 dan ke-5. Pada iterasi berikutnya, bobot data ke-2 dan ke-5 dinaikkan agar model berikutnya lebih fokus ke sana.

```
awal:  [0,2, 0,2, 0,2, 0,2, 0,2]
setelah salah di 2 dan 5: [0,15, 0,275, 0,15, 0,15, 0,275]  # lalu dinormalisasi
```

Dalam AdaBoost, bobot model juga bergantung pada error. Model yang error-nya kecil diberi suara lebih besar. Secara ringkas:

```
alpha = 0,5 × ln((1-error)/error)
```

Jika error=0,2:

```
alpha = 0,5 × ln(0,8/0,2) = 0,5 × ln(4) ≈ 0,693
```

Model yang lebih akurat mendapat alpha lebih besar. Ide ini membantu memahami mengapa boosting kuat tetapi bisa sensitif terhadap label salah: contoh noisy terus diberi perhatian.

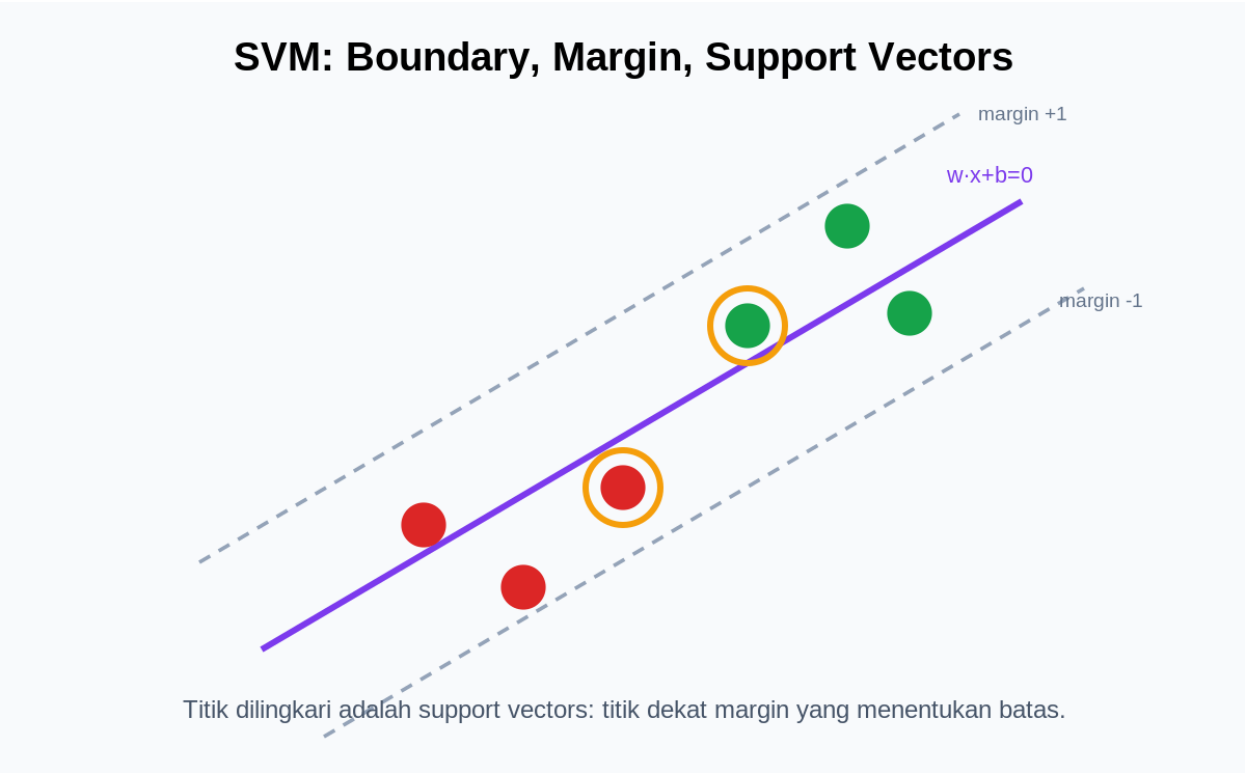
Tes cepat subbab 9

1. Hitung alpha AdaBoost jika error=0,25.

2. Mengapa boosting sensitif terhadap label salah?

3. Jelaskan boosting dengan analogi latihan soal.

Subbab 10 — SVM: mencari margin yang lebar



SVM: mencari margin yang lebar

Support Vector Machine mencari batas keputusan yang memisahkan kelas dengan margin selebar mungkin. Margin adalah jarak aman antara batas keputusan dan contoh terdekat. Ide ini elegan: bukan hanya memisahkan data latih, tetapi mencari pemisah yang tidak terlalu mepet.

Untuk data yang bisa dipisahkan hampir linear, SVM bisa sangat kuat. Dengan kernel, SVM dapat menangani pola non-linear dengan memetakan data ke ruang fitur lain. Namun kernel dan parameter perlu dipilih hati-hati.

Dalam buku pemula, SVM cukup dipahami sebagai model yang peduli pada batas dan margin. Detail optimisasi formal bisa dipelajari nanti. Yang penting: margin lebar sering membantu generalisasi.

Ide teknis / latihan kecil

SVM mencari decision boundary dengan margin besar.
Support vectors = titik-titik penting dekat batas.

Cara membaca gambar SVM

Gambar SVM memiliki tiga garis penting: garis tengah sebagai decision boundary, dan dua garis sejajar sebagai margin. Titik yang menyentuh atau paling dekat dengan margin disebut support vectors. Titik-titik inilah yang paling menentukan posisi batas.

Urutan membaca gambar:

1. Temukan garis tengah `w·x+b=0`.

2. Lihat dua garis margin di kedua sisi.

3. Cari titik yang dilingkari: itulah support vectors.

4. Jika titik berada di sisi benar dan di luar margin, hinge loss 0.

5. Jika titik masuk margin atau salah sisi, hinge loss positif.

Dengan visual ini, pembaca melihat bahwa SVM bukan hanya mencari garis pemisah, tetapi garis yang punya jarak aman.

SVM, margin, dan hinge loss

Untuk klasifikasi linear, decision boundary dapat ditulis:

$$w \cdot x + b = 0$$

Prediksi positif jika $w \cdot x + b \geq 0$, negatif jika kurang dari 0. SVM mencari w dan b yang tidak hanya memisahkan kelas, tetapi memberi margin lebar. Untuk label $y \in \{-1, +1\}$, kondisi margin ideal:

$$y_i(w \cdot x_i + b) \geq 1$$

Jika nilainya besar, titik berada di sisi benar dan jauh dari batas. Jika kurang dari 1, titik melanggar margin.

Hinge loss:

$$\text{Loss}_i = \max(0, 1 - y_i(w \cdot x_i + b))$$

Contoh hitung

Jika $y=+1$ dan skor $w \cdot x+b=0,3$:

$$\text{Loss} = \max(0, 1 - 1 \times 0,3) = 0,7$$

Jika $y=+1$ dan skor$=1,4$:

$$\text{Loss} = \max(0, 1 - 1,4) = 0$$

Titik kedua sudah benar dan cukup jauh dari margin.

Tes cepat subbab 10

1. Hitung hinge loss untuk $y=-1$ dan skor$=0,5$.

2. Apa itu support vector?

3. Mengapa margin lebar membantu generalisasi?

Subbab 11 — Model linear vs non-linear: kapan batas lurus tidak cukup

Model linear vs non-linear: kapan batas lurus tidak cukup

Model linear seperti logistic regression membuat batas keputusan lurus dalam ruang fitur asli. Ini sering cukup jika pola sederhana atau fitur sudah dirancang baik. Model non-linear seperti tree, forest, boosting, kNN, atau kernel SVM dapat menangkap pola yang lebih rumit.

Namun non-linear bukan otomatis lebih baik. Model fleksibel bisa mengejar noise. Jika data kecil, label berisik, atau evaluasi bocor, model kompleks mungkin tampak hebat tetapi rapuh.

Strategi sehat: mulai dari baseline dan model sederhana. Jika error analysis menunjukkan pola yang tidak tertangkap, baru naikkan kompleksitas. Jangan memilih model rumit hanya karena terdengar canggih.

Ide teknis / latihan kecil

> Sederhana dulu → lihat error → tambah kompleksitas jika memang perlu.

Contoh pola XOR: mengapa non-linear dibutuhkan

Ada pola klasik: XOR. Misalkan dua fitur biner x1 dan x2. Label positif jika salah satu bernilai 1, tetapi bukan keduanya.

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0

Tidak ada satu garis lurus yang memisahkan positif dan negatif di ruang 2D asli. Model linear akan kesulitan. Namun jika kita menambah fitur interaksi $x1 \times x2$, pola bisa lebih mudah. Tree juga dapat memecah ruang dengan aturan bertahap.

Pelajaran: kadang masalahnya bukan “model bodoh”, tetapi representasi fitur tidak cukup. Feature engineering dapat mengubah masalah non-linear menjadi lebih mudah untuk model sederhana.

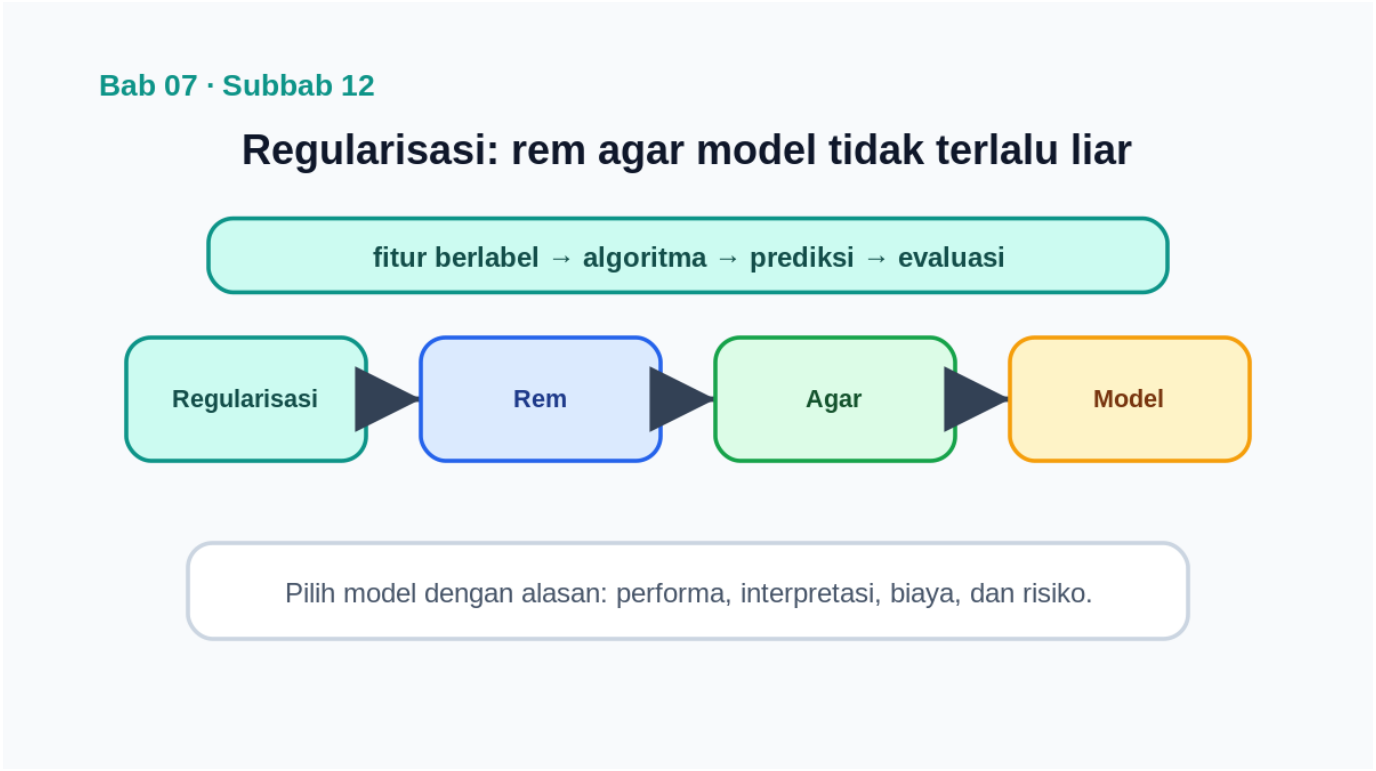
Tes cepat subbab 11

1. Jelaskan mengapa XOR tidak bisa dipisahkan satu garis lurus.

2. Apa peran fitur interaksi?

3. Beri contoh pola non-linear di marketplace.

Subbab 12 — Regularisasi: rem agar model tidak terlalu liar



Regularisasi: rem agar model tidak terlalu liar

Regularisasi adalah cara memberi penalti pada model yang terlalu kompleks. Pada model linear, regularisasi bisa mengecilkan bobot agar model tidak terlalu bergantung pada satu fitur. Pada tree, regularisasi bisa berupa batas kedalaman atau minimum sampel per daun.

Intuisinya seperti rem. Model perlu cukup fleksibel untuk belajar, tetapi tidak boleh terlalu liar mengikuti noise. Regularisasi membantu mengurangi overfitting.

Regularisasi bukan hukuman sembarangan. Ia adalah cara memasukkan preferensi: model yang lebih sederhana dan stabil sering lebih dipercaya jika performanya sebanding.

Ide teknis / latihan kecil

> Loss total = loss prediksi + penalti kompleksitas

L1 dan L2 regularization

Dua regularisasi umum pada model linear:

> L2: loss_total = loss + $\lambda \Sigma w_j^2$
> L1: loss_total = loss + $\lambda \Sigma |w_j|$

L2 mendorong bobot mengecil halus. L1 dapat mendorong sebagian bobot menjadi nol sehingga membantu seleksi fitur. λ dibaca lambda, mengatur kekuatan penalti.

Contoh hitung L2

Jika loss prediksi = 0,40, bobot `[2, -1, 0,5]`, dan λ=0,1:

> $\Sigma w^2 = 2^2 + (-1)^2 + 0{,}5^2 = 4 + 1 + 0{,}25 = 5{,}25$
> penalti = 0,1 × 5,25 = 0,525
> loss_total = 0,40 + 0,525 = 0,925

Model dengan bobot sangat besar dihukum. Ini membantu mengurangi overfitting dan membuat model lebih stabil.

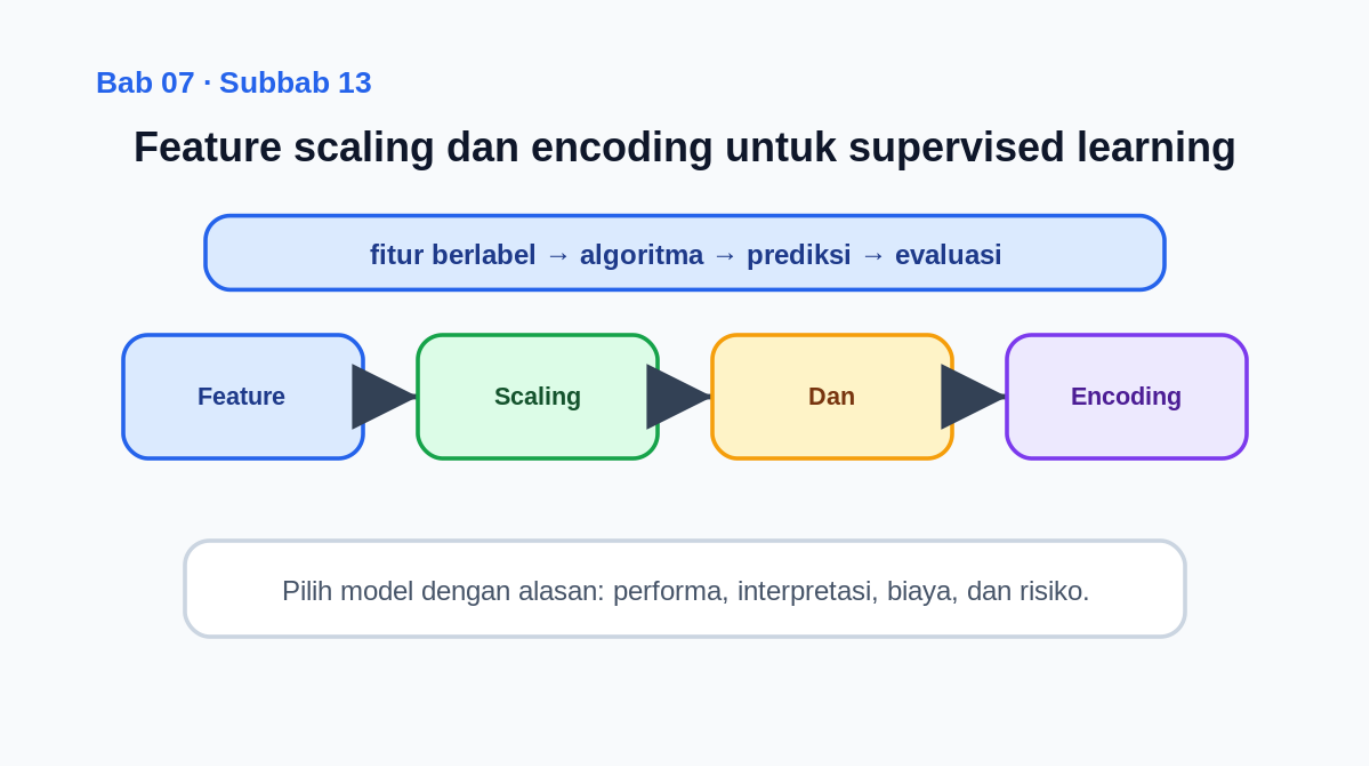
Tes cepat subbab 12

1. Hitung penalti L2 untuk bobot [1,2] dan lambda 0,1.

2. Apa beda intuisi L1 dan L2?

3. Bagaimana regularisasi membantu overfitting?

Subbab 13 — Feature scaling dan encoding untuk supervised learning



Feature scaling dan encoding untuk supervised learning

Beberapa algoritma sangat sensitif terhadap skala fitur: kNN, SVM, logistic regression berbasis gradient. Tree-based model biasanya lebih tahan terhadap skala, tetapi tetap butuh encoding untuk kategori.

Scaling mengubah fitur numerik agar berada pada rentang sebanding. Encoding mengubah kategori menjadi angka. Contoh: kota, jenis produk, atau metode pembayaran. Namun encoding harus hati-hati agar tidak menciptakan urutan palsu.

Seperti Bab 6, preprocessing harus dipelajari dari train saja lalu diterapkan ke validation/test. Jika tidak, kita bisa membocorkan informasi evaluasi.

Ide teknis / latihan kecil

```
x = [10, 20, 30]
scaled = [(v-min(x))/(max(x)-min(x)) for v in x]
print(scaled)
```

Scaling dengan angka nyata

Misalkan fitur klik berada pada rentang 0–1000, sedangkan rating 1–5. Untuk kNN, selisih klik 100 dapat mengalahkan selisih rating 2. Min-max scaling mengubah nilai ke rentang 0–1:

```
x_scaled = (x - min) / (max - min)
```

Jika klik minimum 0 dan maksimum 1000, klik=250 menjadi:

```
(250 - 0) / (1000 - 0) = 0,25
```

Jika rating minimum 1 dan maksimum 5, rating=4 menjadi:

```
(4 - 1) / (5 - 1) = 0,75
```

Sekarang keduanya sebanding. Tetapi scaling harus fit pada train saja. Jika min/max dihitung dari test, informasi evaluasi ikut bocor.

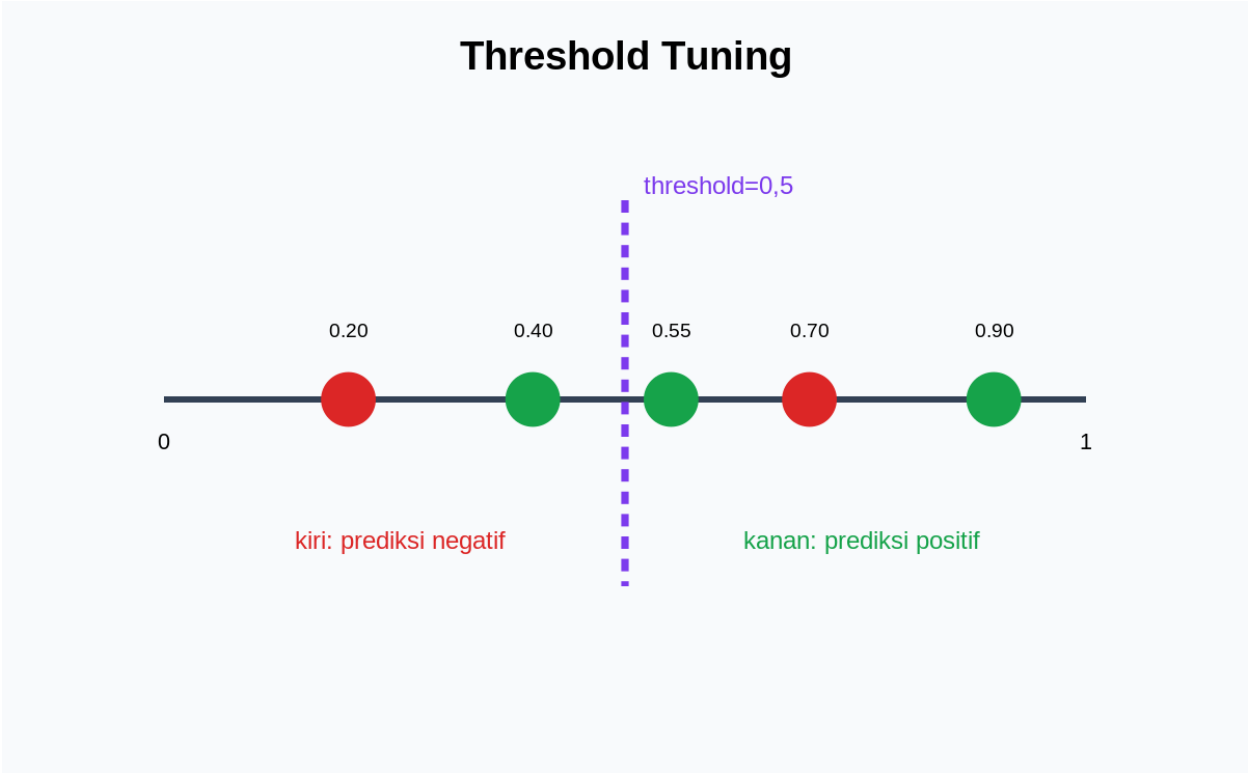
Tes cepat subbab 13

1. Hitung min-max scaling untuk x=75, min=50, max=100.

2. Algoritma mana yang sensitif terhadap skala?

3. Mengapa scaling harus fit pada train?

Subbab 14 — Threshold tuning: probabilitas menjadi keputusan



Threshold tuning: probabilitas menjadi keputusan

Banyak classifier menghasilkan skor atau probabilitas. Untuk membuat keputusan, kita memilih threshold. Jika probabilitas di atas threshold, prediksi positif. Threshold 0,5 sering default, tetapi tidak selalu tepat.

Jika false negative mahal, threshold bisa diturunkan agar recall naik. Jika false positive mahal, threshold bisa dinaikkan agar precision naik. Ini keputusan produk, bukan hanya matematika.

Contoh: skrining siswa butuh bantuan mungkin memilih recall tinggi agar sedikit siswa terlewat. Tetapi jika kapasitas guru sangat terbatas, precision dan ranking juga penting.

Ide teknis / latihan kecil

```
prob >= threshold → positif
threshold turun → recall cenderung naik, precision bisa turun
```

Cara membaca graph threshold

Pada gambar threshold tuning, probabilitas model dapat dibayangkan sebagai garis horizontal dari 0 sampai 1. Threshold adalah pagar vertikal. Titik di kanan pagar menjadi prediksi positif; titik di kiri menjadi negatif. Ketika pagar digeser ke kiri, lebih banyak titik menjadi positif. Recall biasanya naik, tetapi false positive bisa ikut naik.

Urutan membaca gambar:

1. Urutkan skor probabilitas dari kecil ke besar.

2. Letakkan threshold.

3. Tandai prediksi kanan sebagai positif.

4. Bandingkan dengan label aktual untuk menghitung TP/FP/TN/FN.

Latihan threshold dengan confusion matrix

Misalkan classifier memberi probabilitas untuk 5 pelanggan:

pelanggan	probabilitas beli	aktual beli
A	0,90	1
B	0,70	0
C	0,55	1
D	0,40	1
E	0,20	0

Threshold 0,5 menghasilkan prediksi `[1,1,1,0,0]`. Dibanding aktual `[1,0,1,1,0]`:

TP: A,C = 2
FP: B = 1
TN: E = 1
FN: D = 1
precision = 2/(2+1) = 0,667
recall = 2/(2+1) = 0,667

Jika threshold diturunkan ke 0,35, D menjadi positif. Recall naik menjadi 1, tetapi FP tetap perlu dicek. Latihan ini menunjukkan threshold adalah keputusan operasional.

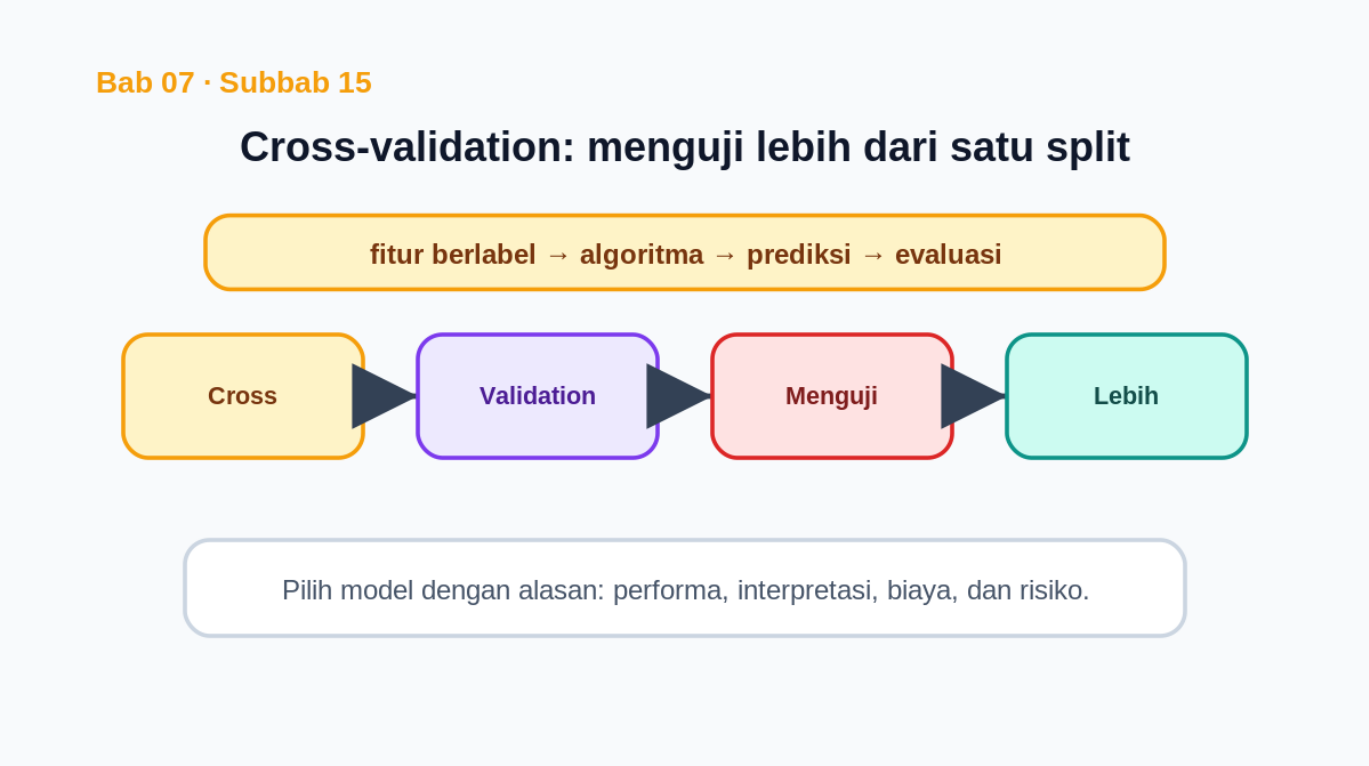
Tes cepat subbab 14

1. Dari tabel probabilitas, hitung precision dan recall untuk threshold 0,5.

2. Apa efek threshold diturunkan?

3. Kapan precision lebih penting daripada recall?

Subbab 15 — Cross-validation: menguji lebih dari satu split



Cross-validation: menguji lebih dari satu split

Satu split train/test bisa kebetulan menguntungkan atau merugikan. Cross-validation membagi data menjadi beberapa fold, melatih dan mengevaluasi berulang pada fold berbeda, lalu merata-ratakan skor.

Ini membantu saat data terbatas. Namun cross-validation harus tetap menghormati struktur data. Untuk data waktu, jangan mengacak masa depan ke train. Untuk data pengguna, jangan sampai data pengguna yang sama bocor ke train dan test.

Cross-validation bukan pengganti test final. Ia alat untuk memilih model lebih stabil sebelum evaluasi akhir.

Ide teknis / latihan kecil

K-fold CV: bagi data jadi K bagian → latih K kali → rata-rata skor.

Contoh 5-fold cross-validation

Misalkan ada 100 data dan K=5. Data dibagi menjadi 5 fold masing-masing 20 data. Prosesnya:

Putaran 1: validasi fold 1, train fold 2-5
Putaran 2: validasi fold 2, train fold 1,3,4,5
...
Putaran 5: validasi fold 5, train fold 1-4

Jika skor F1 lima putaran adalah `[0,70, 0,75, 0,72, 0,78, 0,74]`, rata-ratanya:

(0,70+0,75+0,72+0,78+0,74)/5 = 0,738

Kita juga melihat variasi skor. Jika skor antar fold sangat jauh, model mungkin tidak stabil atau data terlalu kecil/beragam. Cross-validation memberi gambaran lebih kaya daripada satu split.

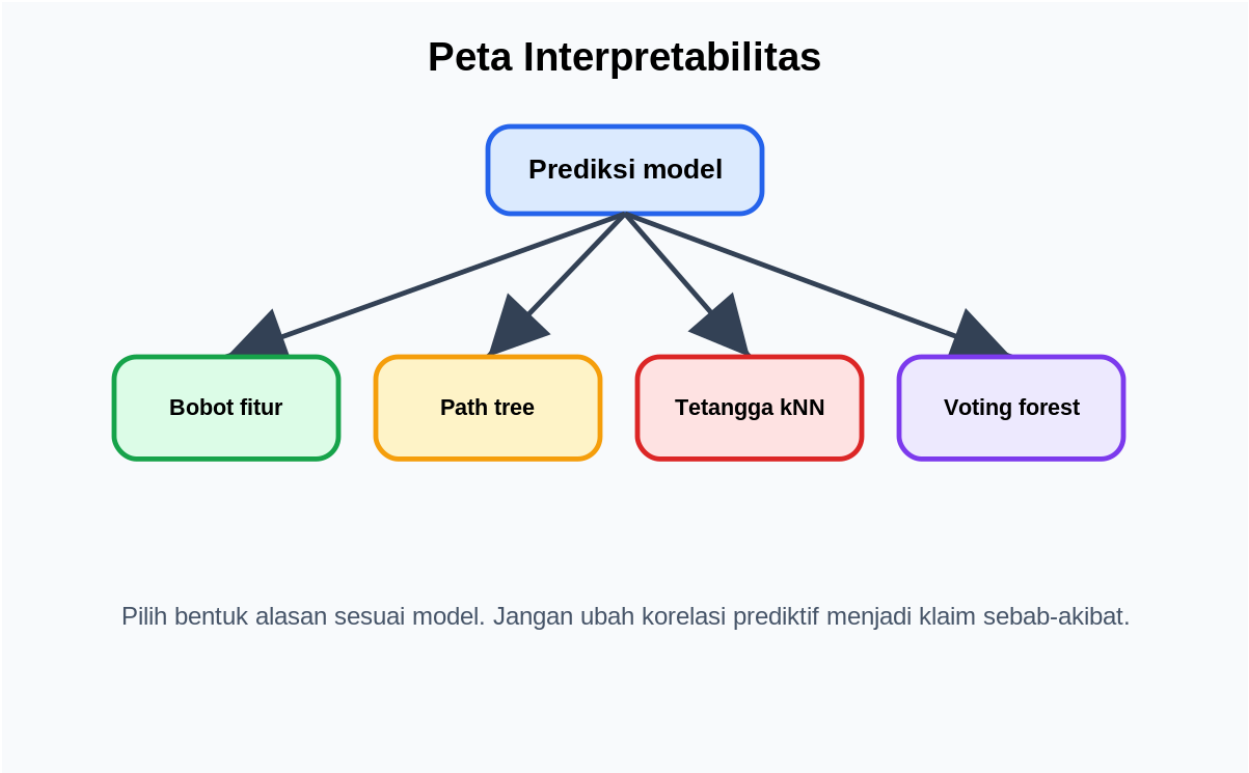
Tes cepat subbab 15

1. Hitung rata-rata skor CV [0,8,0,7,0,9].

2. Mengapa CV tidak boleh melanggar urutan waktu?

3. Apa tanda model tidak stabil dari hasil fold?

Subbab 16 — Interpretabilitas: bisa menjelaskan bukan sekadar memprediksi



Interpretabilitas: bisa menjelaskan bukan sekadar memprediksi

Dalam banyak konteks Indonesia—pendidikan, kredit, kesehatan, layanan publik—prediksi saja tidak cukup. Pengguna perlu alasan. Decision tree mudah dijelaskan, logistic regression bisa memberi bobot fitur, random forest/boosting perlu alat interpretasi tambahan.

Interpretabilitas bukan berarti semua model harus sederhana. Tetapi semakin besar dampak keputusan, semakin besar kebutuhan penjelasan, audit, dan jalur banding.

Penjelasan model harus jujur. Jangan membuat cerita palsu setelah prediksi. Jika model tidak bisa dijelaskan dengan cukup, batasi penggunaannya pada konteks yang risikonya rendah atau sediakan manusia sebagai peninjau.

Ide teknis / latihan kecil

> Model berdampak tinggi → perlu alasan, audit, dan koreksi manusia.

Cara membaca graph interpretabilitas

Graph interpretabilitas bukan hanya gambar model, tetapi peta alasan. Untuk logistic regression, peta alasan berupa bobot fitur: panah positif menaikkan skor, panah negatif menurunkan skor. Untuk tree, alasan berupa path dari root ke leaf. Untuk kNN, alasan berupa tetangga terdekat.

Urutan membaca alasan model:

1. Tanyakan model apa yang dipakai.

2. Pilih bentuk penjelasan yang sesuai: bobot, path, tetangga, atau vote.

3. Bedakan alasan prediktif dan sebab-akibat.

4. Tulis batasan penjelasan agar tidak terlihat lebih pasti daripada kenyataan.

Interpretabilitas dengan contoh bobot

Logistic regression dapat memberi petunjuk lewat bobot. Misalkan bobot:

```
rating: +1,2
diskon: +2,0
klik: +0,3
harga_rel: -1,5
```

Tanda positif berarti fitur menaikkan skor beli; tanda negatif menurunkan skor. Namun interpretasi harus hati-hati: bobot bergantung pada scaling, korelasi antar fitur, dan data latih. Jangan mengatakan “diskon pasti menyebabkan pembelian” hanya karena bobot positif. Itu korelasi prediktif, bukan bukti kausal.

Untuk tree, interpretasi berupa aturan. Untuk kNN, interpretasi berupa tetangga terdekat. Untuk ensemble, interpretasi lebih sulit dan membutuhkan alat tambahan seperti permutation importance atau SHAP pada tahap lanjut.

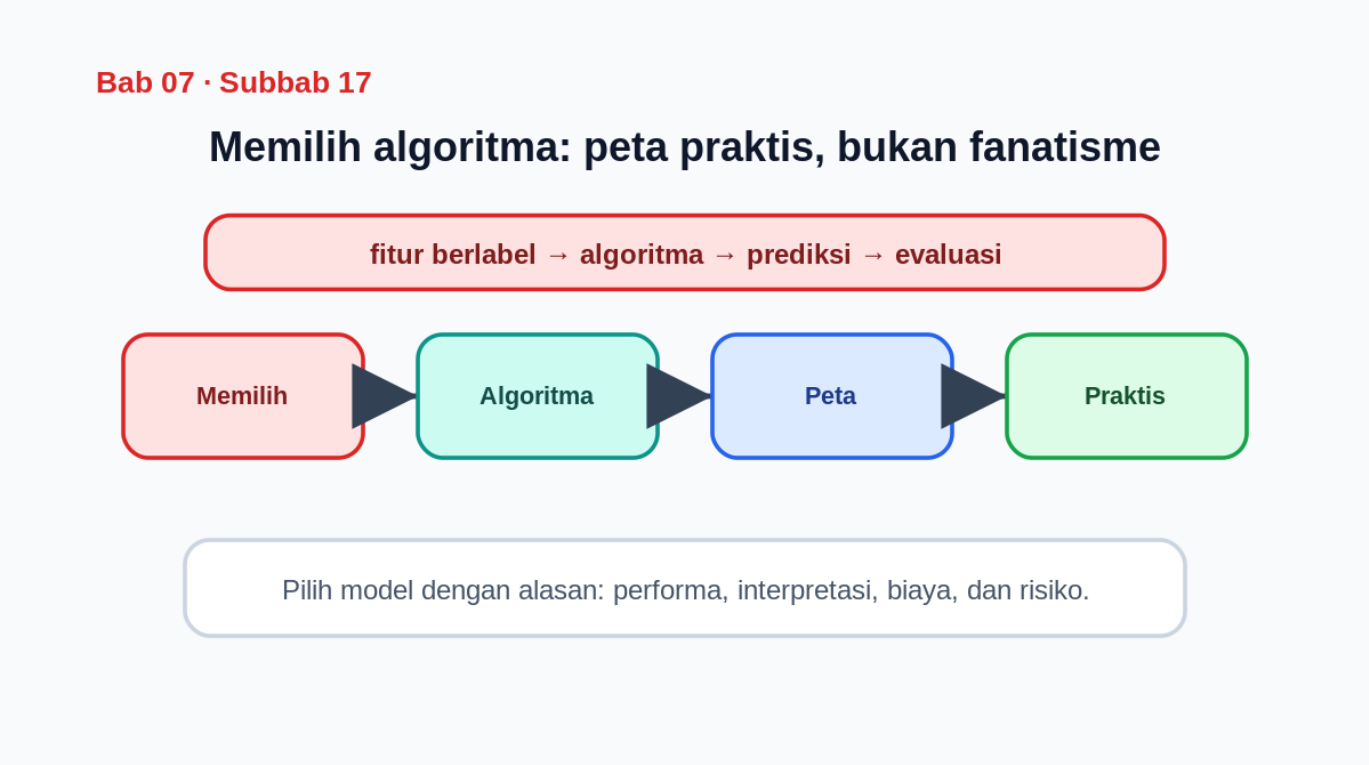
Tes cepat subbab 16

1. Interpretasikan bobot harga_rel negatif.

2. Mengapa bobot bukan bukti kausal?

3. Model mana paling mudah dijelaskan di bab ini?

Subbab 17 — Memilih algoritma: peta praktis, bukan fanatisme



Memilih algoritma: peta praktis, bukan fanatisme

Tidak ada algoritma terbaik untuk semua masalah. kNN cocok untuk intuisi jarak dan data kecil. Naive Bayes cocok sebagai baseline teks. Logistic regression kuat sebagai baseline linear. Tree mudah dijelaskan. Random forest stabil. Boosting kuat pada tabular. SVM berguna untuk margin dan dataset ukuran tertentu.

Pilihan algoritma sebaiknya mengikuti data, tujuan, metrik, ukuran dataset, kebutuhan interpretasi, dan biaya komputasi. Jika dataset kecil dan butuh penjelasan, tree/logistic regression mungkin lebih cocok. Jika performa tabular penting dan interpretasi masih bisa dikelola, boosting bisa dicoba.

Sikap terbaik: bukan fanatik model, tetapi ilmiah. Uji baseline, bandingkan, lihat error, dokumentasikan.

Ide teknis / latihan kecil

Pilih model berdasarkan: data + metrik + interpretasi + biaya + risiko.

Tabel pemilihan praktis

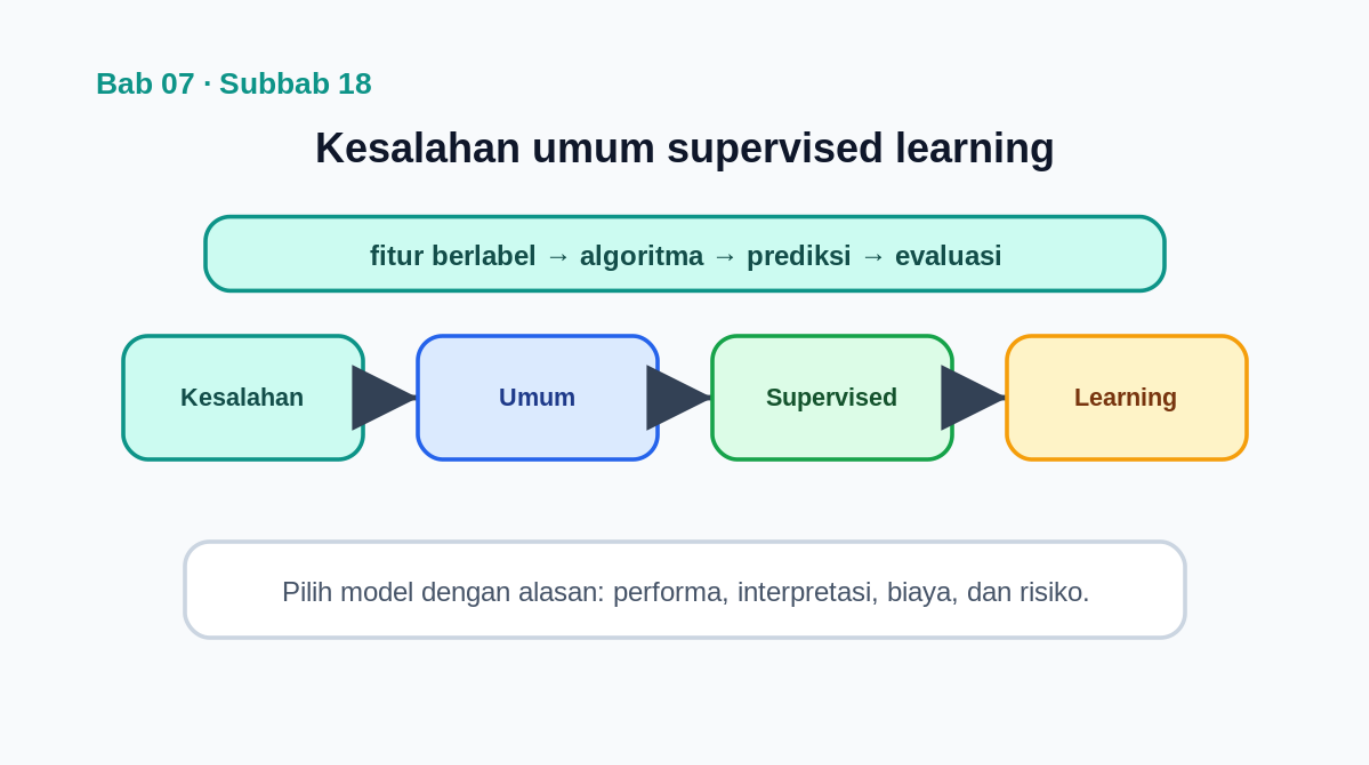
Kondisi	Model awal yang masuk akal	Alasan
Butuh baseline cepat	Logistic regression / Naive Bayes	Cepat dan mudah dibandingkan
Data kecil, jarak bermakna	kNN	Intuitif dan lokal
Butuh aturan mudah dibaca	Decision tree	Jalur keputusan jelas
Data tabular, performa kuat	Random forest / boosting	Ensemble sering stabil/kuat
Butuh margin linear	SVM / linear model	Cocok untuk batas tegas

Tabel ini bukan hukum. Ia hanya peta awal. Keputusan final tetap harus berdasarkan eksperimen, metrik, error analysis, dan kebutuhan produk.

Tes cepat subbab 17

1. Pilih model awal untuk data teks spam dan jelaskan alasannya.
2. Pilih model awal untuk data tabular UMKM dan jelaskan alasannya.
3. Mengapa tidak ada algoritma terbaik universal?

Subbab 18 — Kesalahan umum supervised learning



Kesalahan umum supervised learning

Kesalahan umum pertama: mengejar accuracy tanpa melihat class imbalance. Kedua: membandingkan model tanpa baseline. Ketiga: memakai test set berkali-kali. Keempat: lupa preprocessing fit hanya di train. Kelima: mengabaikan error analysis. Keenam: menganggap model kompleks pasti lebih baik.

Kesalahan lain: label tidak jelas, fitur bocor, split salah untuk data waktu, dan metrik tidak sesuai aksi. Semua ini lebih sering merusak proyek daripada pilihan algoritma.

Bab 7 ingin menanam mental debugging. Jika performa aneh, jangan panik. Periksa data, target, split, metrik, baseline, leakage, distribusi, dan contoh error.

Ide teknis / latihan kecil

> Jika skor aneh: cek data → label → split → baseline → metric → leakage → error.

Checklist debugging supervised learning

Saat model buruk, ikuti urutan ini:

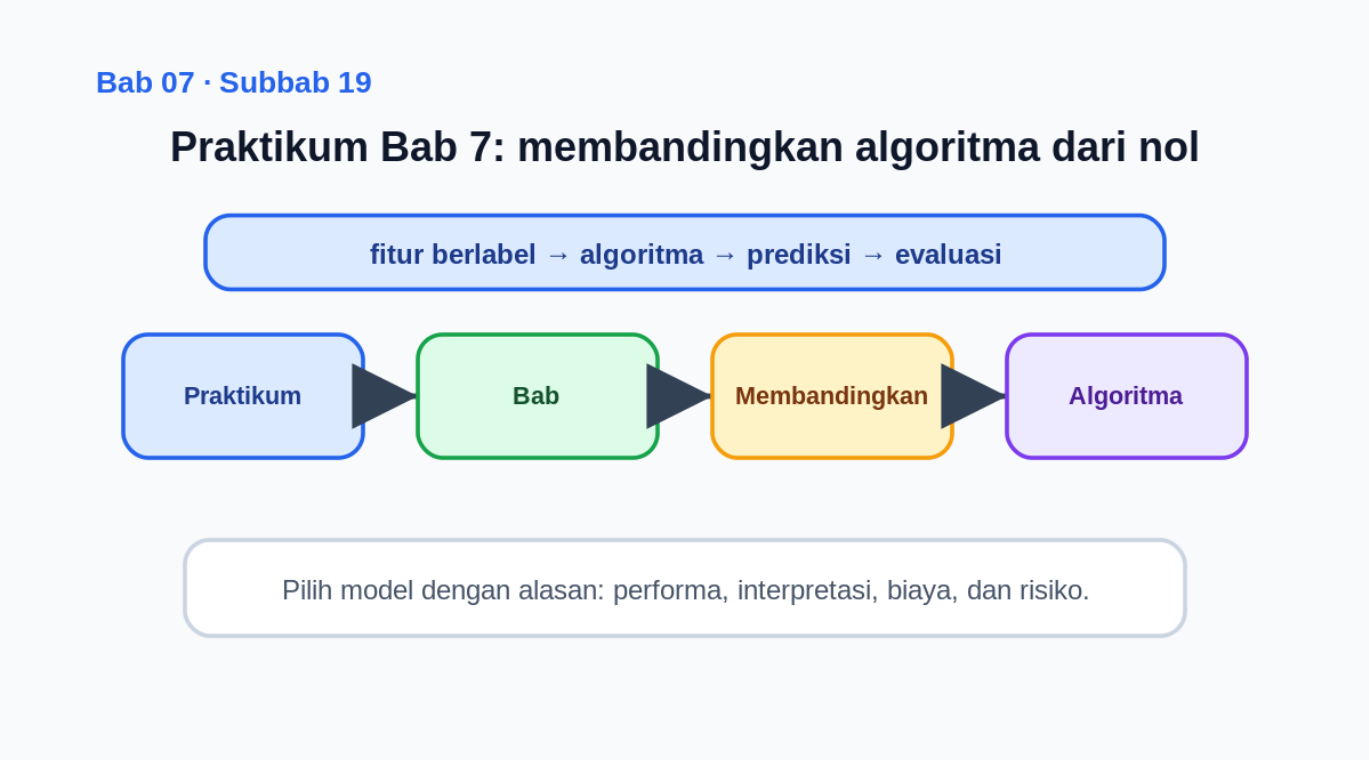
1. Cek label: apakah definisinya jelas?
2. Cek split: apakah train/test bocor?
3. Cek baseline: apakah model mengalahkannya?
4. Cek distribusi: apakah train dan test mirip?
5. Cek metrik: apakah sesuai konsekuensi?
6. Cek error: false positive/false negative terjadi pada pola apa?
7. Cek fitur: apakah ada fitur tidak tersedia saat prediksi?
8. Cek kompleksitas: underfit atau overfit?

Urutan ini mencegah kita langsung mengganti algoritma tanpa diagnosis.

Tes cepat subbab 18

1. Ikuti checklist debugging untuk kasus accuracy tinggi tetapi recall nol.
2. Mengapa baseline wajib sebelum model kompleks?
3. Apa indikasi data leakage?

Subbab 19 — Praktikum Bab 7: membandingkan algoritma dari nol



Praktikum Bab 7 membandingkan beberapa model standard library: majority baseline, kNN, Gaussian Naive Bayes, logistic regression kecil, decision stump, ensemble stumps, dan perceptron sebagai intuisi linear margin. Tujuannya bukan menggantikan scikit-learn, tetapi membuat konsep terasa nyata.

Pembaca akan melihat setiap model menghasilkan prediksi dan metrik. Setelah itu, pembaca diminta mengubah fitur, threshold, k, learning rate, dan data agar memahami perilaku model.

Jika pembaca bisa menjelaskan mengapa model A lebih baik pada dataset tertentu dan model B lebih mudah dijelaskan, ia sudah mulai berpikir seperti praktisi ML, bukan sekadar pengguna library.

Ide teknis / latihan kecil

```
python3 supervised_learning_playground.py
```

Soal praktik terstruktur untuk pembaca

Gunakan output script Bab 7. Buat tabel seperti ini:

Model	Accuracy	Precision	Recall	F1	Catatan
Baseline	...	...	...	...	...
kNN	...	...	...	...	...
Naive Bayes	...	...	...	...	...
Logistic	...	...	...	...	...

Lalu jawab:

1. Model mana F1 tertinggi?

2. Model mana recall tertinggi?

3. Model mana paling mudah dijelaskan ke pemilik UMKM?

4. Jika false negative lebih mahal, model mana yang kamu pilih?

5. Jika penjelasan wajib, apakah pilihanmu berubah?

Dengan latihan ini, pembaca belajar bahwa “model terbaik” selalu bergantung pada tujuan dan batasan.

Tes cepat subbab 19

1. Isi tabel perbandingan model dari output script.

2. Pilih model jika false negative lebih mahal.

3. Pilih model jika interpretabilitas wajib.

Pendalaman teknis sebelum praktikum

Ada beberapa prinsip yang perlu dipegang ketika membandingkan algoritma. Pertama, metrik yang sama belum tentu berarti kualitas yang sama. Dua model bisa punya F1 serupa tetapi jenis error berbeda. Untuk produk pendidikan, model dengan recall lebih tinggi mungkin lebih disukai meskipun precision sedikit turun. Untuk tindakan mahal, precision mungkin lebih penting. Karena itu, setiap tabel skor harus dibaca bersama confusion matrix dan contoh error.

Kedua, model yang menghasilkan probabilitas belum tentu probabilitasnya terkalibrasi. Jika model berkata 0,8, idealnya dari banyak kasus dengan skor 0,8 sekitar 80% benar-benar positif. Banyak model menghasilkan skor yang berguna untuk ranking tetapi tidak otomatis bagus sebagai

probabilitas keputusan. Threshold tuning dan calibration menjadi topik lanjut, tetapi fondasinya perlu dikenal sejak sekarang.

Ketiga, interpretasi model harus disesuaikan dengan risiko. Untuk rekomendasi produk hiburan, model kompleks yang sulit dijelaskan mungkin masih diterima. Untuk kredit, pendidikan, kesehatan, atau layanan publik, penjelasan dan audit jauh lebih penting. Jangan hanya bertanya “model mana paling akurat?”, tetapi juga “model mana bisa dipertanggungjawabkan?”.

Keempat, supervised learning selalu bergantung pada label masa lalu. Jika label masa lalu bias, model bisa belajar bias itu. Jika label masa lalu berasal dari keputusan manusia yang tidak konsisten, model bisa meniru ketidakkonsistenan. Data berlabel adalah aset besar, tetapi juga membawa sejarah proses yang perlu diperiksa.

Kelima, algoritma adalah bagian dari sistem. Model yang bagus di notebook bisa gagal jika fitur di produksi dihitung berbeda, jika data terlambat masuk, jika distribusi pengguna berubah, atau jika hasil model tidak ditindaklanjuti dengan benar. Maka supervised learning yang matang selalu kembali ke Bab 6: framing, split jujur, baseline, error analysis, reproducibility, dan monitoring.

Keenam, jangan membandingkan model hanya dari satu angka. Buat tabel yang memuat accuracy, precision, recall, F1, jumlah false positive, jumlah false negative, waktu training, kemudahan penjelasan, dan risiko operasional. Model dengan F1 tertinggi mungkin bukan pilihan terbaik jika sangat sulit dijelaskan pada pengguna terdampak. Sebaliknya, model sederhana dengan performa sedikit lebih rendah bisa lebih tepat jika keputusan sensitif dan membutuhkan audit.

Ketujuh, pisahkan eksplorasi dan keputusan. Saat eksplorasi, kita boleh mencoba banyak model untuk belajar. Saat membuat keputusan, kita harus menetapkan kriteria: metrik utama, batas minimum performa, batas interpretabilitas, dan risiko yang tidak boleh dilanggar. Tanpa kriteria, pemilihan model mudah berubah menjadi “model mana yang terlihat keren”.

Kedelapan, supervised learning bukan hanya untuk klasifikasi biner. Banyak proyek nyata memakai multiclass classification, multi-label classification, ordinal classification, regresi multi-output, atau ranking. Bab ini fokus dasar karena pembaca perlu fondasi kuat. Setelah fondasi ini matang, variasi lain lebih mudah dipelajari: semuanya tetap kembali ke fitur, target, loss, metrik, dan evaluasi jujur.

Kesembilan, catat keterbatasan implementasi manual. Script Bab 7 sengaja menulis algoritma dari nol agar pembaca memahami mekanisme. Untuk produksi, gunakan library teruji, pipeline yang benar, validasi statistik, monitoring, dan review keamanan. Manual code adalah kaca pembesar untuk belajar; library produksi adalah alat kerja yang sudah diuji lebih luas.

Kesepuluh, model yang berbeda bisa benar dengan cara berbeda. kNN benar karena tetangga mirip. Naive Bayes benar karena bukti probabilistik cukup kuat. Logistic regression benar karena kombinasi linear fitur memisahkan kelas. Tree benar karena aturan bertahap menangkap interaksi. Ensemble benar karena banyak model saling menutupi kelemahan. Memahami “cara benar” ini membuat pembaca lebih siap melakukan error analysis.

Kesebelas, supervised learning perlu dibaca bersama distribusi data. Jika data train berisi pelanggan kota besar tetapi test berisi pelanggan daerah, performa bisa turun. Jika train berisi harga normal tetapi masa depan punya promo besar, model bisa meleset. Jika kelas positif jarang, threshold default bisa salah. Karena itu, sebelum menyalahkan algoritma, selalu lihat distribusi fitur dan label.

Keduabelas, jangan mengabaikan biaya prediksi. kNN sederhana saat training tetapi bisa lambat saat prediksi karena harus membandingkan dengan banyak data. Logistic regression cepat saat prediksi. Tree juga cepat dan mudah dibaca. Ensemble lebih berat tetapi sering lebih akurat. Dalam produk nyata, pilihan model juga mempertimbangkan latensi, memori, kemudahan deploy, dan frekuensi update.

Ketigabelas, model supervised learning punya siklus hidup. Hari ini data dilabeli, minggu ini model dilatih, bulan depan pola pengguna berubah. Model yang dulu baik dapat menurun. Maka hasil Bab 7 harus disambung dengan kebiasaan monitoring: cek metrik berkala, cek drift, cek keluhan pengguna, dan siapkan proses retraining jika diperlukan.

Praktikum terpadu Bab 7

File utama:

- `code/supervised_learning_playground.py`
- `code/supervised_learning_playground.ipynb`

Jalankan dari terminal:

```
cd zero-to-hero-menaklukkan-ai/chapters/07-supervised-learning/code
python3 supervised_learning_playground.py
```

Eksperimen wajib:

1. Bandingkan baseline, kNN, Naive Bayes, logistic regression, decision stump, ensemble stump, dan perceptron.
2. Ubah nilai k pada kNN.
3. Ubah learning rate logistic regression.
4. Ubah threshold probabilitas.
5. Tambahkan noise pada data dan lihat model mana yang paling stabil.
6. Tulis model mana yang paling mudah dijelaskan dan model mana yang paling baik pada metrik.

Ringkasan Bab 7

- Supervised learning belajar dari contoh berlabel.
- Regresi memprediksi angka; klasifikasi memprediksi kategori.
- kNN memakai tetangga terdekat dan sensitif terhadap skala fitur.
- Naive Bayes cepat dan probabilistik, tetapi memakai asumsi independensi sederhana.
- Logistic regression adalah baseline linear yang kuat untuk klasifikasi probabilistik.
- Decision tree mudah dijelaskan tetapi bisa overfit.
- Random forest dan boosting memakai banyak model sederhana untuk performa lebih kuat.
- SVM menekankan margin.
- Model sederhana sering menjadi awal terbaik.
- Pemilihan algoritma harus mengikuti data, metrik, interpretasi, biaya, dan risiko.

Referensi utama bab

[R1] James, Witten, Hastie, Tibshirani, Taylor. *An Introduction to Statistical Learning*. Supervised learning, classification, regression, tree-based methods; tidak disalin. [R2] Hastie, Tibshirani, Friedman. *The Elements of Statistical Learning*. Model linear, SVM, boosting, random forest; tidak disalin. [R3] Géron. *Hands-On Machine Learning*. Workflow praktis supervised learning; tidak disalin. [R4] Mitchell. *Machine Learning*. kNN, Naive Bayes, decision tree, generalisasi; tidak disalin. [R5] scikit-learn documentation. API dan metrik pembanding; tidak disalin.

Catatan validasi internal v0.3

Aspek	Status	Catatan
Struktur	Sesuai Bab 6	Subbab tanpa label halaman.
Kedalaman	Draft lengkap	19 subbab algoritma/konsep plus pendalaman teknis dan praktikum.
Praktikum	Siap v0.3	Perbandingan model standard library.
Risiko	Terbuka	Implementasi manual untuk edukasi, bukan pengganti library produksi.