1) **Apa yang dimaksud dengan hold-out validation dan k-fold cross-validation?**

Hold-out validation adalah pembagian tunggal dataset menjadi train–validasi–(opsional) test, melatih pada train dan menilai pada validasi. Metode ini cepat dan mencerminkan alur produksi yang langsung, tetapi estimasinya ber-varian tinggi karena bergantung pada satu irisan data—bila split "beruntung" atau "sial", metrik bisa bias. **k-fold cross-validation** membagi data menjadi *k* lipatan, melatih *k* kali dengan tiap lipatan bergiliran menjadi validasi, lalu merata-ratakan skor. Ini mengurangi varians estimasi dan memaksimalkan pemakaian data latih, namun biaya komputasinya kira-kira *k* kali lebih besar dan implementasi ceroboh (mis. preprocessing di luar loop fold) mudah menimbulkan kebocoran.

---

2) **Kapan hold-out lebih baik dibanding k-fold, dan sebaliknya?**

Hold-out lebih tepat ketika satu split sudah representatif—misalnya pada data sangat besar—sehingga k-fold hanya menambah waktu tanpa manfaat material; saat sumber daya komputasi terbatas atau model mahal dilatih; serta pada skenario non-IID seperti deret waktu, di mana pemisahan **temporal** (latih masa lalu -> validasi masa depan) wajib agar kausalitas terjaga. **k-fold lebih unggul** pada data kecil–menengah, hasil model yang labil, atau kelas tak seimbang; rata-rata lintas fold memberi estimasi lebih stabil untuk pemilihan model/hyperparameter. Pada data dengan unit berulang (pasien/pengguna/sekolah), gunakan variasi *group-aware*; untuk deret waktu gunakan **TimeSeriesSplit**—"k-fold acak" bukan solusi universal.

---

3) **Apa yang dimaksud dengan data leakage?**

Data leakage adalah masuknya informasi yang **tidak akan tersedia saat prediksi** ke proses pelatihan/validasi. Bentuk umum: *target leakage* (fitur menyiratkan label/masa depan), *preprocessing leakage* (normalisasi/imputasi/seleksi fitur dihitung dari seluruh data sebelum split atau di luar loop CV), kontaminasi lintas split (duplikasi/entitas sama di train & validasi), serta *tuning leakage* (test set dipakai berulang untuk memilih hyperparameter). Semua ini membuat evaluasi terlihat lebih baik daripada kemampuan generalisasi yang sesungguhnya.

---

4) **Bagaimana dampak data leakage terhadap kinerja model?**

Kebocoran hampir selalu menghasilkan metrik validasi yang **terlalu optimistis** dan *ranking* model yang menipu. Akibatnya, kita memilih model/hyperparameter yang salah, lalu performa ambruk saat berhadapan dengan data nyata. Secara statistik, terjadi **overestimation** pada akurasi/ROC-AUC/F1, varians tersembunyi tidak terukur, dan stabilitas model menurun—yang berujung pada keputusan bisnis/riset yang keliru serta hasil sulit direproduksi.

---

5) **Solusi untuk mengatasi data leakage**

Pegang prinsip **“fit hanya pada data latih”** dan bungkus seluruh preprocessing di dalam **pipeline** yang dieksekusi per-split/per-fold. Praktiknya: lakukan split terlebih dahulu; tempatkan *scaler*, imputasi, seleksi fitur, dan *resampling* (SMOTE/oversampling) di pipeline yang **di-fit pada train saja** di setiap fold. Pilih skema split yang sesuai struktur data: **Stratified** untuk ketidakseimbangan kelas, **GroupKFold** untuk entitas berulang, **TimeSeriesSplit** untuk deret waktu (sertakan *temporal gap* bila perlu). Jaga **test set** benar-benar perawan dan pakai sekali di akhir; untuk tuning agresif, gunakan **nested CV** (inner untuk tuning, outer untuk estimasi generalisasi). Lengkapi dengan audit kausalitas fitur (hindari sinyal masa depan), pembersihan duplikasi lintas split, serta *logging* seed/versi data/konfigurasi split agar bisa diaudit dan direproduksi.