Kevin Alif Bagaskara

1103210075

UTS Machine Learning

1. Jika model Machine Learning menunjukkan AUC-ROC tinggi (0.92) tetapi Presisi sangat rendah (15%) pada dataset tersebut, jelaskan faktor penyebab utama ketidaksesuaian ini! Bagaimana strategi tuning hyperparameter dapat meningkatkan Presisi tanpa mengorbankan AUC-ROC secara signifikan? Mengapa Recall menjadi pertimbangan kritis dalam konteks ini, dan bagaimana hubungannya dengan cost false negative?

AUC-ROC yang tinggi menunjukkan bahwa model mampu membedakan dengan baik antara dua kelas secara umum, tetapi Presisi yang sangat rendah berarti banyak prediksi positif yang salah atau banyak false positive. Ini sering terjadi pada dataset dengan class imbalance tinggi di mana model cenderung lebih 'berani' memberikan prediksi positif padahal mayoritas data negatif. Strategi tuning hyperparameter seperti menaikkan threshold keputusan (misalnya dari 0.5 ke 0.7) bisa meningkatkan Presisi karena hanya prediksi dengan probabilitas tinggi yang dikategorikan sebagai positif, walau ini bisa menurunkan Recall. Untuk mempertahankan AUC-ROC, tuning bisa dilakukan lewat teknik seperti penalizing false positives dalam fungsi loss atau menerapkan class weight balancing. Recall menjadi kritis dalam kasus seperti fraud detection atau diagnosis penyakit karena false negative jauh lebih mahal secara biaya atau konsekuensi. Di sinilah trade-off antara Presisi dan Recall jadi nyata, dan pemilihan metrik tergantung pada konsekuensi kesalahan. false negative bisa menyebabkan kerugian besar, sehingga walau Presisi rendah, Recall juga tetap harus dijaga.

2. Sebuah fitur kategorikal dengan 1000 nilai unik (high-cardinality) digunakan dalam model machine learning. Jelaskan dampaknya terhadap estimasi koefisien dan stabilitas Presisi! Mengapa target encoding berisiko menyebabkan data leakage dalam kasus dataset tersebut, dan alternatif encoding apa yang lebih aman untuk mempertahankan AUC-ROC?

Fitur kategorikal dengan banyak nilai unik, seperti ID merchant atau ZIP code dapat mengacaukan model terutama model linier dan pohon jika tidak diencode dengan bijak. One-hot encoding akan menghasilkan ribuan kolom dan menyebabkan overfitting, terutama pada data kecil. Ini memperburuk estimasi koefisien karena model 'menghafal' daripada belajar. Target encoding (mengganti kategori dengan rata-rata target-nya) memang lebih kompak, tetapi sangat berisiko menyebabkan kebocoran data jika diterapkan sebelum split karena informasi target bocor ke fitur. Ini menyebabkan Presisi dan metrik lainnya tampak tinggi di validasi, tapi buruk di testing. Alternatif yang lebih aman adalah frequency encoding (mengganti dengan jumlah kemunculan), hashing trick (fitur randomisasi terbatas), atau target encoding yang dilakukan dengan skema K-fold cross-validation di training set saja agar tidak bocor. Teknik ini menjaga stabilitas AUC-ROC tanpa mengorbankan Presisi terlalu besar.

3. Setelah normalisasi Min-Max, model SVM linear mengalami peningkatan Presisi dari 40% ke 60% tetapi Recall turun 20%. Analisis dampak normalisasi terhadap decision boundary dan margin kelas minoritas! Mengapa scaling yang sama mungkin memiliki efek berlawanan jika diterapkan pada model Gradient Boosting?

SVM sangat bergantung pada jarak antar titik untuk menentukan margin. Kenaikan Presisi menandakan bahwa decision boundary menjadi lebih 'rapi' setelah fitur berada dalam skala

seragam, tetapi penurunan Recall menunjukkan bahwa margin terlalu sempit atau terlalu menghindari kelas minoritas. Ini umum jika minoritas terdesak oleh mayoritas setelah scaling. Pada sisi lain, Gradient Boosting tidak sensitif terhadap skala karena ia berbasis pohon keputusan. Bahkan, scaling bisa melemahkan kekuatan model boosting dalam mengenali threshold alami fitur. Jadi, normalisasi yang memperbaiki margin pada SVM bisa berdampak sebaliknya pada model boosting yang sebenarnya mengandalkan raw value distribusi. Hal ini menunjukkan bahwa preprocessing harus mempertimbangkan sifat dasar algoritma yang digunakan.

4. Eksperimen feature interaction dengan menggabungkan dua fitur melalui perkalian meningkatkan AUC-ROC dari 0.75 ke 0.82. Jelaskan mekanisme matematis di balik peningkatan ini dalam konteks decision boundary non-linear! Mengapa uji statistik seperti chi-square gagal mendeteksi interaksi semacam ini, dan metode domain knowledge apa yang dapat digunakan sebagai alternatif?

Ketika dua feature dikalikan seperti X1 * X2, kita dengan tidak langsung memperkenalkan non-linearitas ke dalam model, bahkan jika model dasarnya linier. Jika hubungan target tidak aditif tetapi interaktif (misalnya hanya terjadi pada kondisi spesifik gabungan fitur), maka interaksi memungkinkan model membentuk decision boundary yang melengkung atau berubah tergantung konteks. AUC-ROC naik karena model jadi lebih akurat membedakan kelas dengan feature baru ini. Namun, uji statistik seperti Chi-square gagal mendeteksi interaksi semacam ini karena uji tersebut mengukur asosiatif linear atau marginal antar variabel dan target, bukan konjungsi antar feature. Untuk menangkap interaksi seperti ini, domain knowledge menjadi sangat penting. misalnya mengetahui bahwa risiko penipuan tinggi hanya jika device = mobile dan transaction amount > $1000. Teknik seperti decision trees shallow atau partial dependence plots juga bisa digunakan untuk mendeteksi efek interaksi.

5. Dalam pipeline preprocessing, penggunaan oversampling sebelum pembagian train-test menyebabkan data leakage dengan AUC-ROC validasi 0.95 tetapi AUC-ROC testing 0.65. Jelaskan mengapa temporal split lebih aman untuk fraud detection, dan bagaimana stratified sampling dapat memperparah masalah ini! Bagaimana desain preprocessing yang benar untuk memastikan evaluasi metrik Presisi/Recall yang realistis?

Oversampling seharusnya hanya diterapkan pada training set karena jika dilakukan sebelum train-test split, informasi dari target label tersuntik ke model secara tidak sah. Inilah penyebab validasi AUC-ROC tampak tinggi (karena model ‘curang’) tapi testing jeblok karena generalisasi buruk. Temporal split jauh lebih aman untuk domain seperti fraud detection karena pola penipuan sering berubah seiring waktu. Jika kita shuffle data sebelum split, informasi dari masa depan bisa bocor ke pelatihan, yang tidak mungkin terjadi di realita. Stratified sampling memperburuk ini karena memaksa proporsi kelas di test dan train serupa. Padahal dalam praktik, kasus fraud biasanya langka dan datang secara tak beraturan. Maka, preprocessing yang benar mencakup: membagi data secara temporal, melakukan oversampling hanya di training set, dan melakukan cross-validation berbasis waktu untuk menjaga evaluasi tetap realistis terhadap metrik seperti Presisi dan Recall.