



Steel Plate Defect Prediction

Playground Series - Season 4, Episode 3

Playground · 2199 Teams · 2 years ago

S4E3 Steel Plate Defect Prediction

生成AIと表形式の基盤モデルで、Late Summer
Private 0.8904 まで

2026-09-24

背景の欠陥画像と寸法はイメージです（実物の



効いたのは「問題の定式化」と「違う種類のモデルを混ぜること」。EDA から出したアイデアはスコアを上げなかったが、上限と理由を説明した。

コンペとデータ

タスク	鋼板の表面欠陥7種類の確率を予測
入力	欠陥の画像から取り出した数値の特徴量27個（画像はない）
評価	7列それぞれの ROC AUC の平均（7ターゲットが同じ重み）
データ	train 19,219行、test 12,814行（合成）、元データ UCI 1,941行
上位	Private 0.8892~0.8898 に密集（1位 0.88977）

7ターゲットは同じ重み → 件数の少ない欠陥も同じだけ効く

7種類の欠陥の姿

上段：元データの数値から描いた大きさと明るさ（共通の縮尺）

下段：名前からのイメージ（実物の画像ではない）

Pastry



10×30 px 面積 209
平均輝度 105 最暗 80
銅種A300 31% (n=158)

Z_Scratch



15×20 px 面積 146
平均輝度 107 最暗 93
銅種A300 91% (n=190)

K_Scratch



155×67 px 面積 6281
平均輝度 108 最暗 41
銅種A300 0% (n=391)

Stains



6×4 px 面積 16
平均輝度 128 最暗 115
銅種A300 1% (n=72)

Dirtiness



10×35 px 面積 145
平均輝度 117 最暗 107
銅種A300 16% (n=55)

Bumps

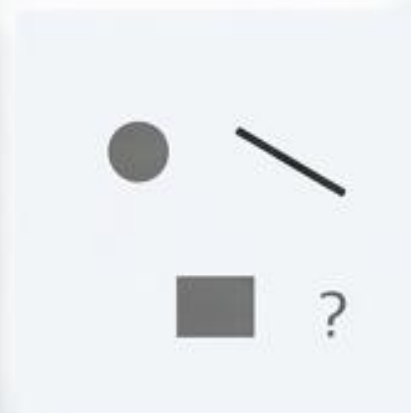


12×14 px 面積 120
平均輝度 110 最暗 91
銅種A300 69% (n=402)

Other_Faults



14×18 px 面積 146
平均輝度 113 最暗 97
銅種A300 40% (n=673)



データの中身といちばん大事なデータの性質

id	横×縦 (px)	面積	明るさ 最小/最大	鋼種	板厚	正解
0	6×5	16	113/140	A400	50	Stains
1	8×22	433	70/111	A300	80	Other_Faults
2	153×68	11388	29/141	A400	40	K_Scatch
3	8×27	210	114/134	A400	40	K_Scatch
4	20×45	521	82/111	A400	300	Other_Faults

ラベル分布 (train)



**形式はマルチラベルだが、
中身はほぼ多クラス分類。**

8クラス（7欠陥+欠陥なし）で分類し、
7列の確率に戻す（P@Q 変換）。

ターゲットの陽性率



Other_Faults (34%)、Bumps (25%) が多く、
Stains (3%)、Dirtiness (2.5%) が少ない。

元データとは
Z_Scratch・Bumps の比率が少しずれる。

生成AIによる調査

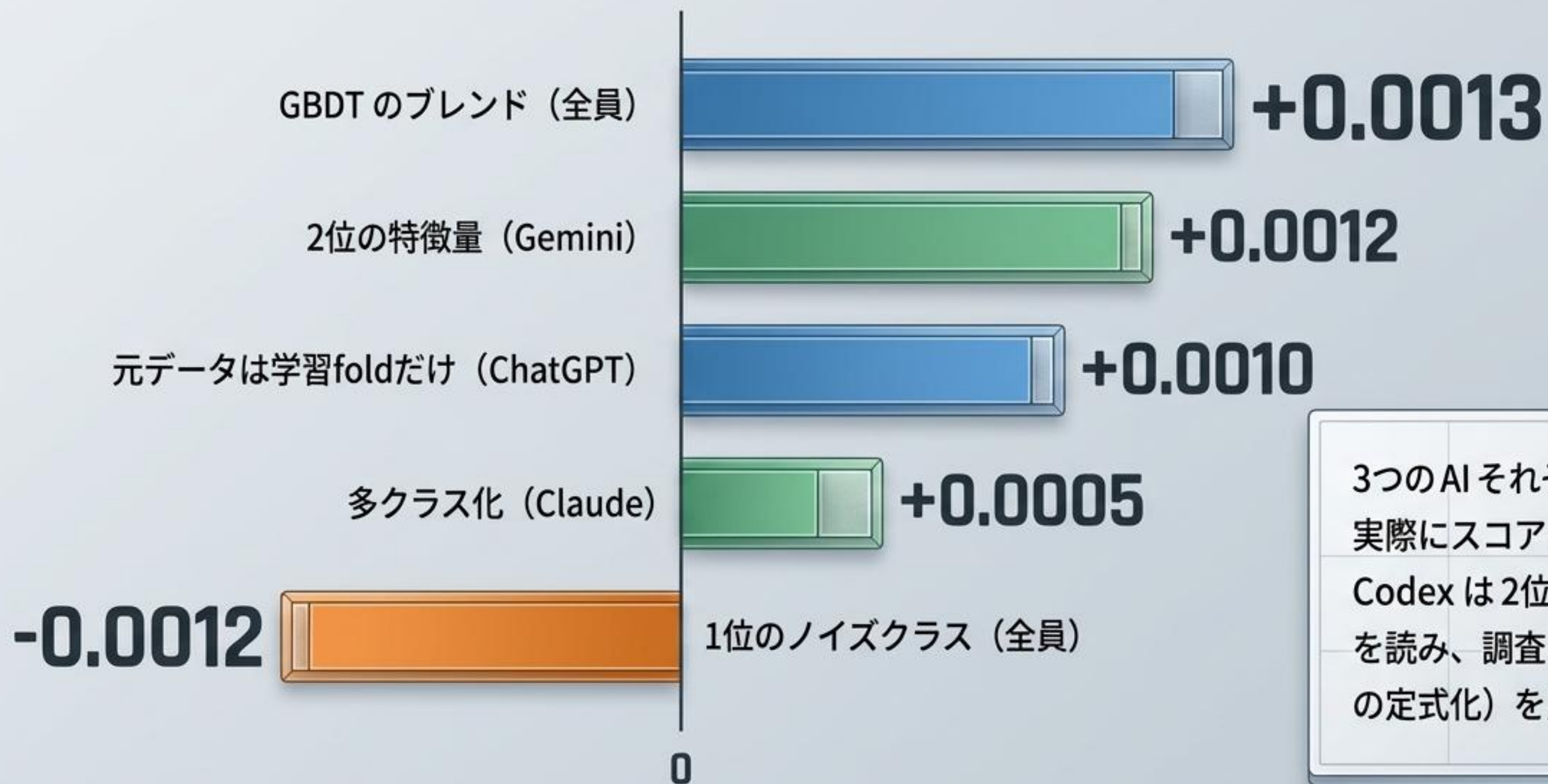
3つの生成AIで上位解法を調べ、統合した



AI Model	正確性	網羅性	実用性	独自貢献
Claude	◎	◎	○	定式化の比較、P@Q
ChatGPT	△ (スコア取違い)	△	◎ (検証設計)	fold固定、元データは学習側だけ
Gemini	× (別コンペ混入)	△	△	2位 writeup の発見

共通ルール: fold固定 / 元データは学習側のみ / OOFで判断、LBで調整しない / 上書き禁止

調査の主張は、実行で確かめられたか



3つのAIそれぞれの貢献が、実際にスコアを上げた。

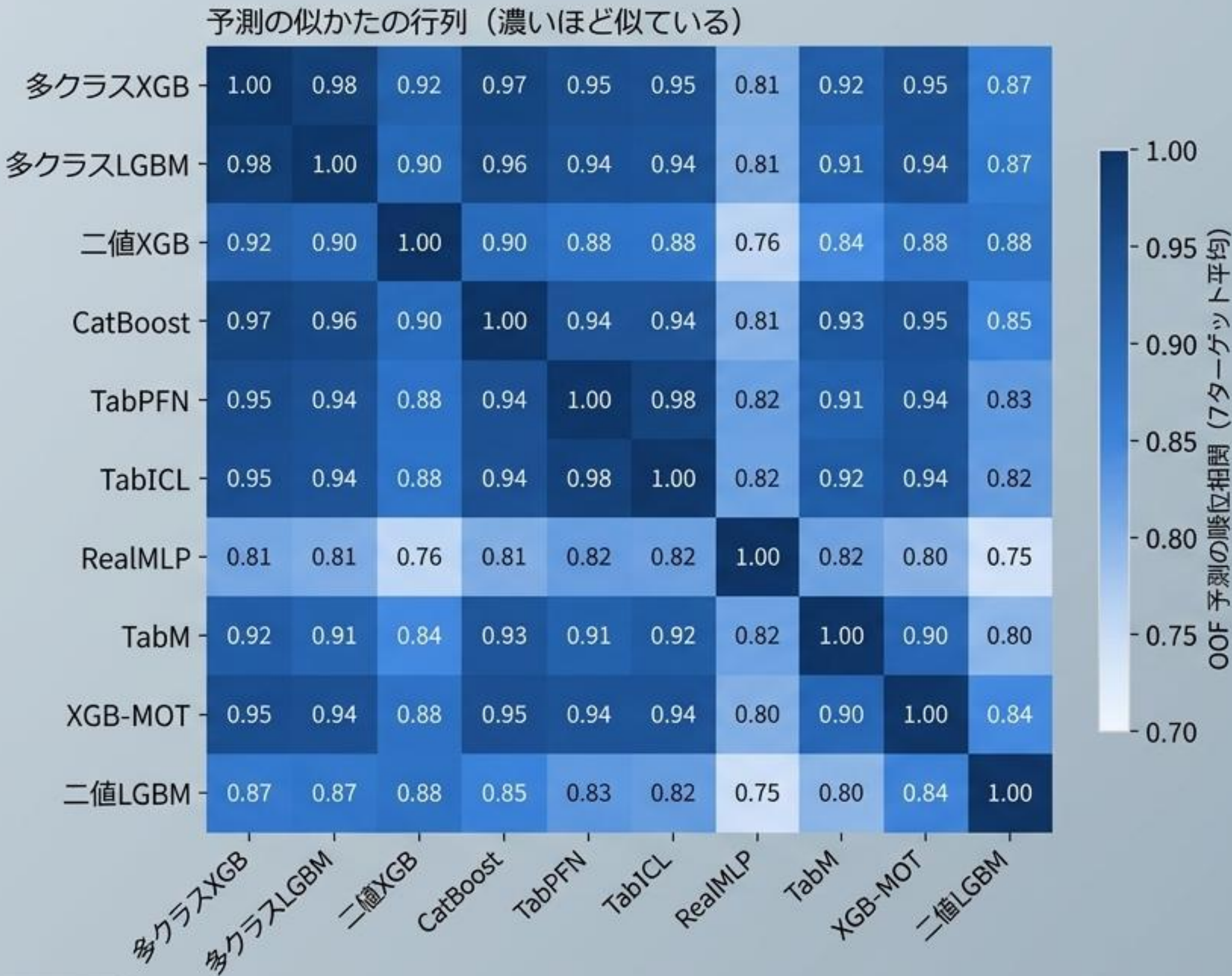
Codex は2位の公開コードを読み、調査の誤り (XGBの定式化) を見つけた。

第1ラウンド (Codex) : 積み上げ

提出 : Private 0.88894
(目標 0.889 に 0.00006 届かず)



第2ラウンドの狙い：GBDT どうしは似すぎている

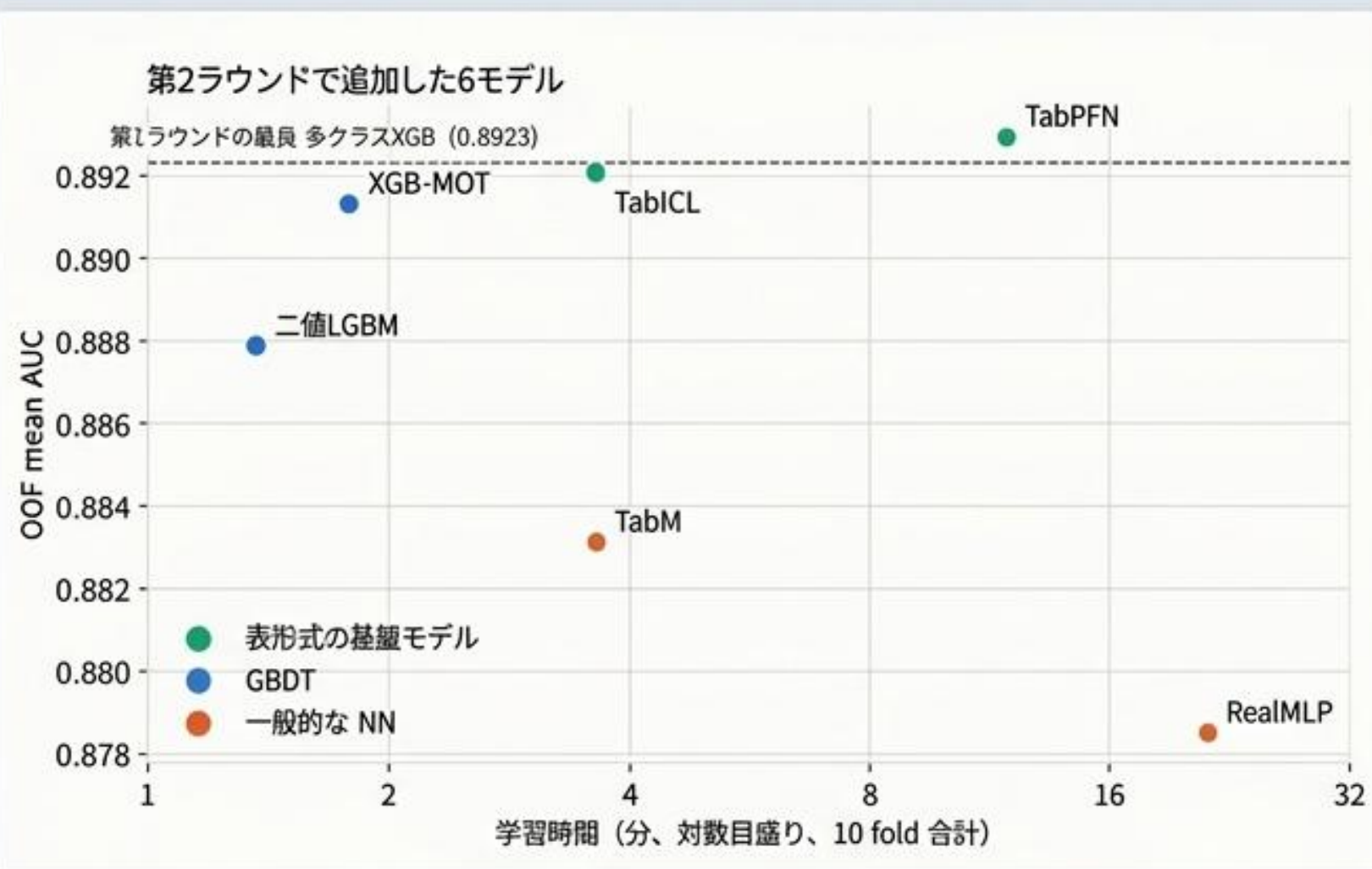


第1ラウンドの
GBDT 4種は互いに
0.90~0.98 と
極めて近い。

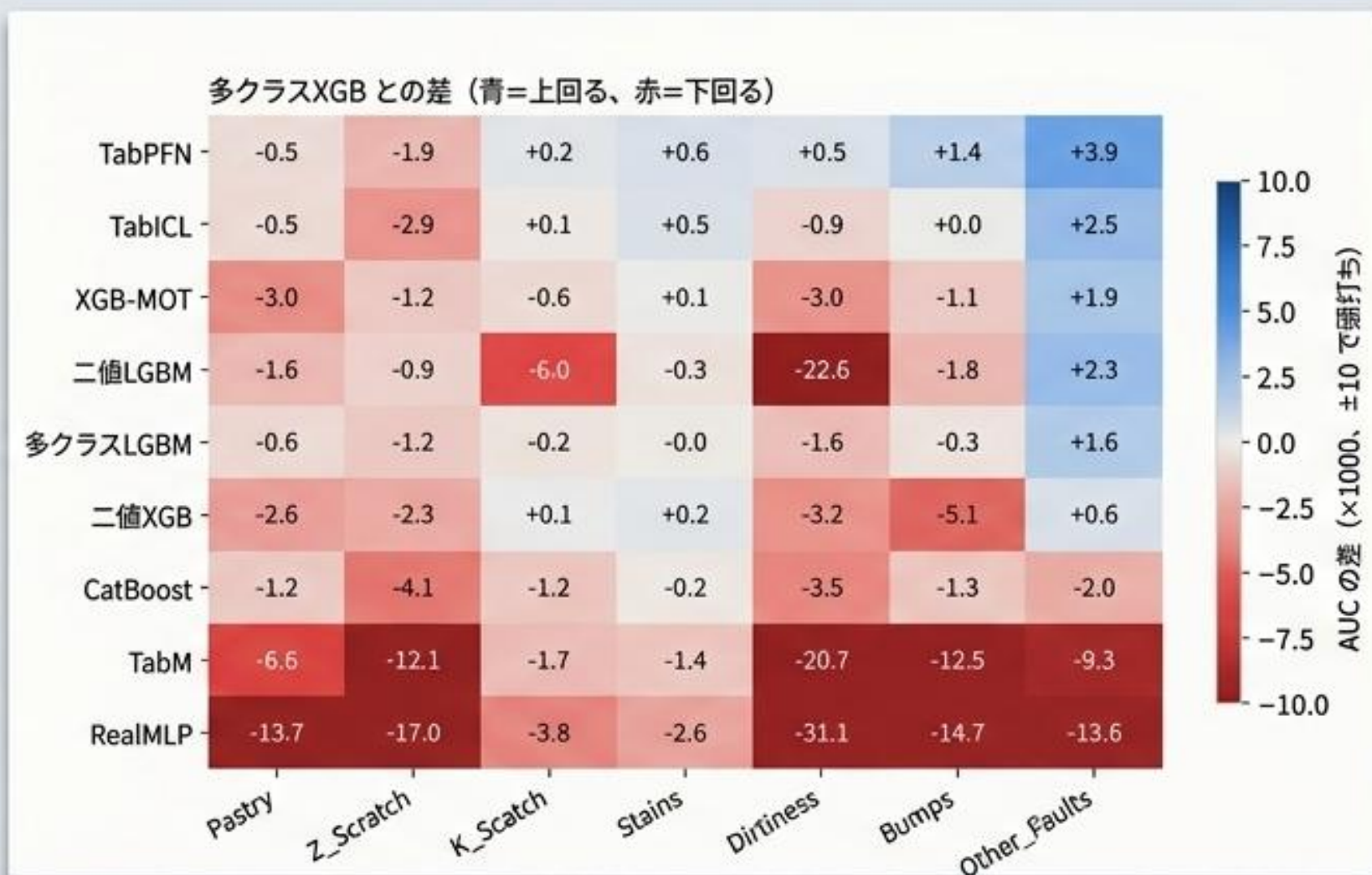
ACTION / COUNTERMEASURE

対策：違う仕組みの
モデル（基盤モデル、
NN、二値分類）を足
して多様性を増やす。

第2ラウンド：6つのモデルを追加とターゲット別比較



TabPFN-3.5: OOF 0.8929 (12分). 調整も早期停止もなしで、GBDTを上回って単体最高。一般的なNN (RealMLP/TabM) は既定の設定では弱い。



TabPFNはOther_Faults・BumpsでGBDTを上回る。二値LGBMは全体では弱いですが、最も難しいOther_Faultsでは上回る。

ブレンドの方法と「違い」の重要性

Methodology Comparison

Method	OOF Score
幾何平均	0.89386
全体 Nelder-Mead	0.89467
ターゲット別 Nelder-Mead	0.89458
Ridge	0.89429
Optuna+Ridge	0.89452

選び方の基準はブレンドの前に決めた（前回を
+0.0003 以上、差が小さければ単純な手法を選択）。

Final Blend Weights & Key Insight



二値LGBM は単体では弱い（0.8879）のに重みは 0.12。第2ラウンドで追加したモデルの重みの合計は約6割。

提出結果



OOF の改善 (+0.0011) が、そのまま Private の改善 (+0.0014) に表れた。
全データでの再学習の効果はごく小さい (+0.00007)。

どのターゲットが伸びたか



伸びたのは AUC が低い4つ。すでに高い3つはほとんど変わらない。
「なぜこの4つのターゲットが伸びたのか？」を EDA で調べる。

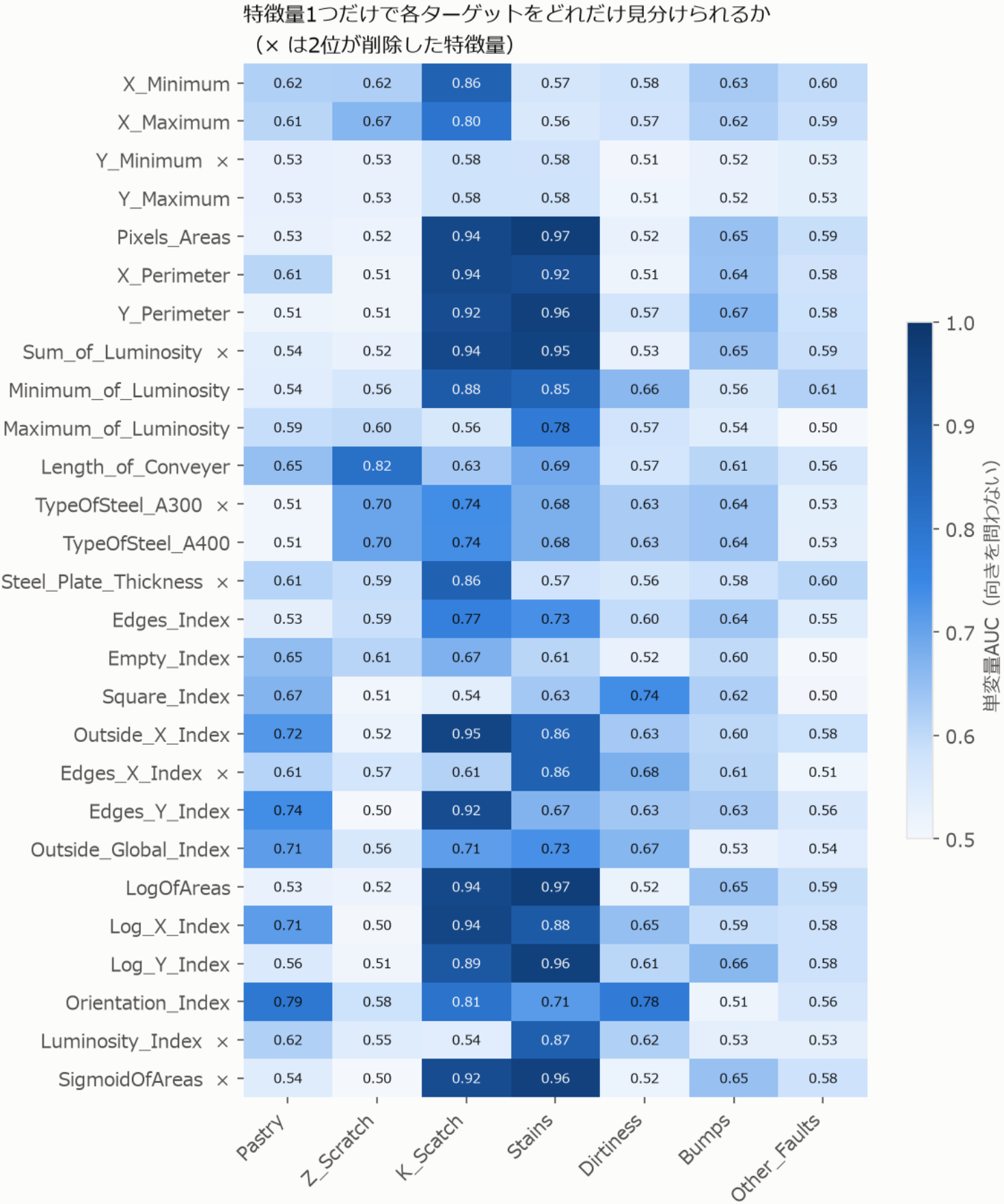
EDA①

特徴量1つでどこまで見分けられるか

- Stains・K_Scratchは面積や形1つではほぼ決まる
- Bumps・Other_Faultsには決め手になる特徴量がない

単変量AUC トップの特徴量

ターゲット	最も効く特徴量	AUC
Stains	LogOfAreas	0.970
K_Scratch	Outside_X_Index	0.948
Bumps	Y_Perimeter	0.667
Other_Faults	Minimum_of_Luminosity	0.615



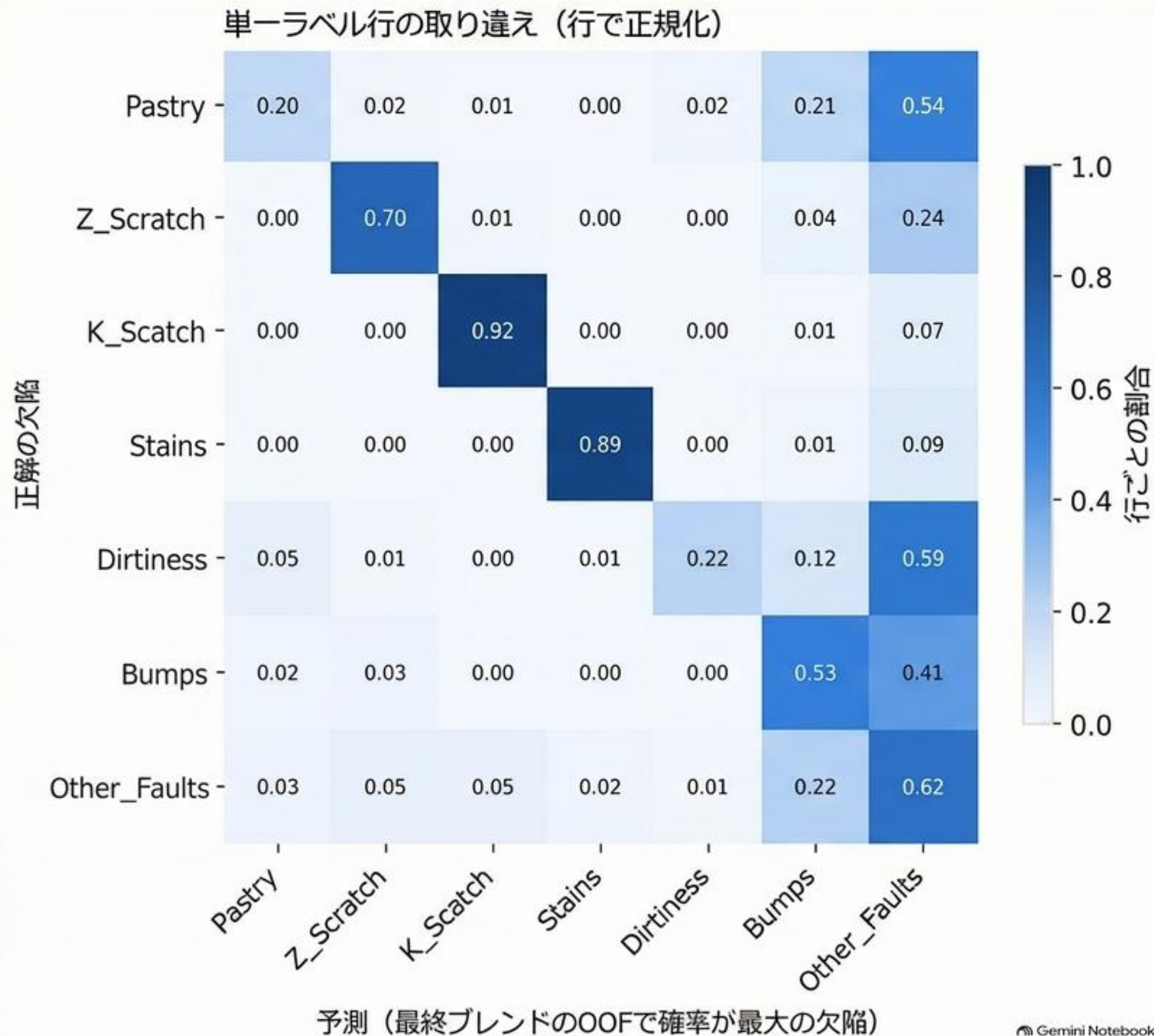
EDA②

どの欠陥とどの欠陥
を間違えるか

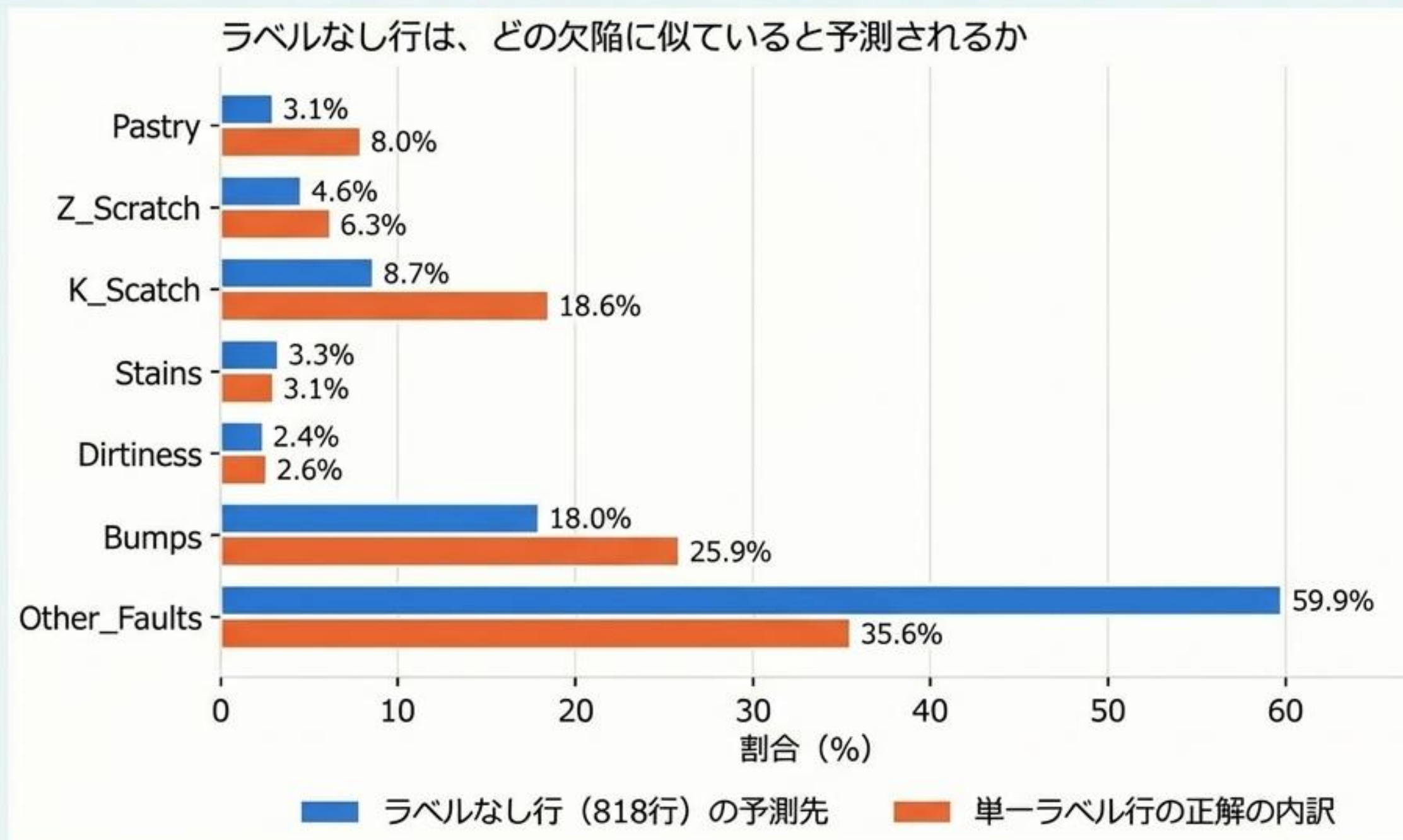
間違いの行き先はほとんどが
Other_Faults と Bumps

結論

**Other_Faults ・ Bumps ・
Pastry ・ Dirtiness が
「見分けにくいかたまり」**



EDA③ ラベルなしの818行は何に似ているか



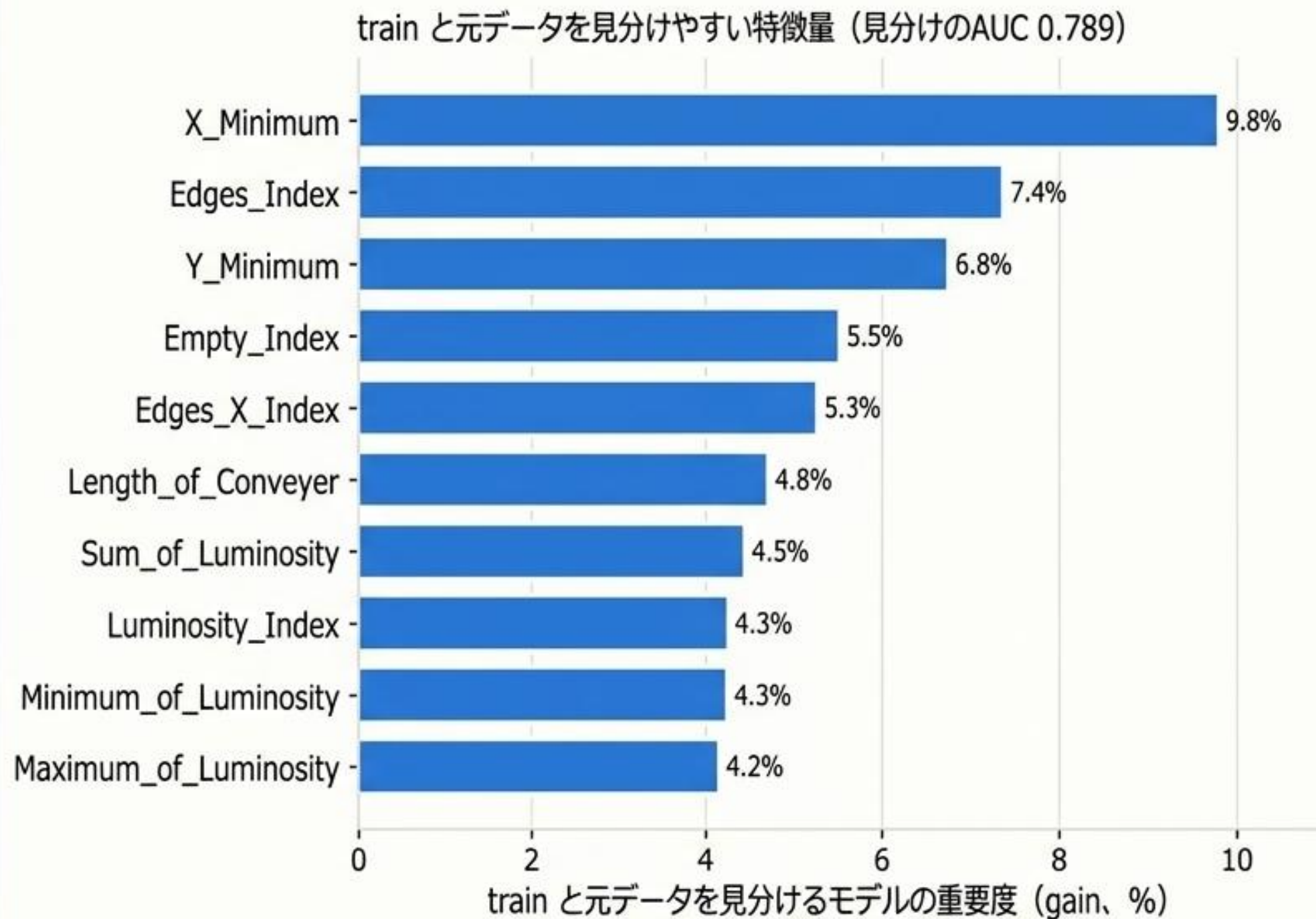
- 60% が Other_Faults、18% が Bumps と予測される（確信度も低い）
- 元データには存在しない行 → 合成の過程でできた「あいまいな行」

EDA④ 分布のずれ

- train と test は同じ分布
→ OOF をそのまま信用できる
- 元データとの違いは主に
X の位置

見分けの AUC

比較	AUC
train と test	0.501 (同じ分布)
train と元データ	0.789 (違う分布)



EDA⑤ 合成データの異常値

- 「最大 < 最小」の行が約4% (test にも同じ割合)
- 異常のある行でも予測の精度は落ちていない

異常データ件数

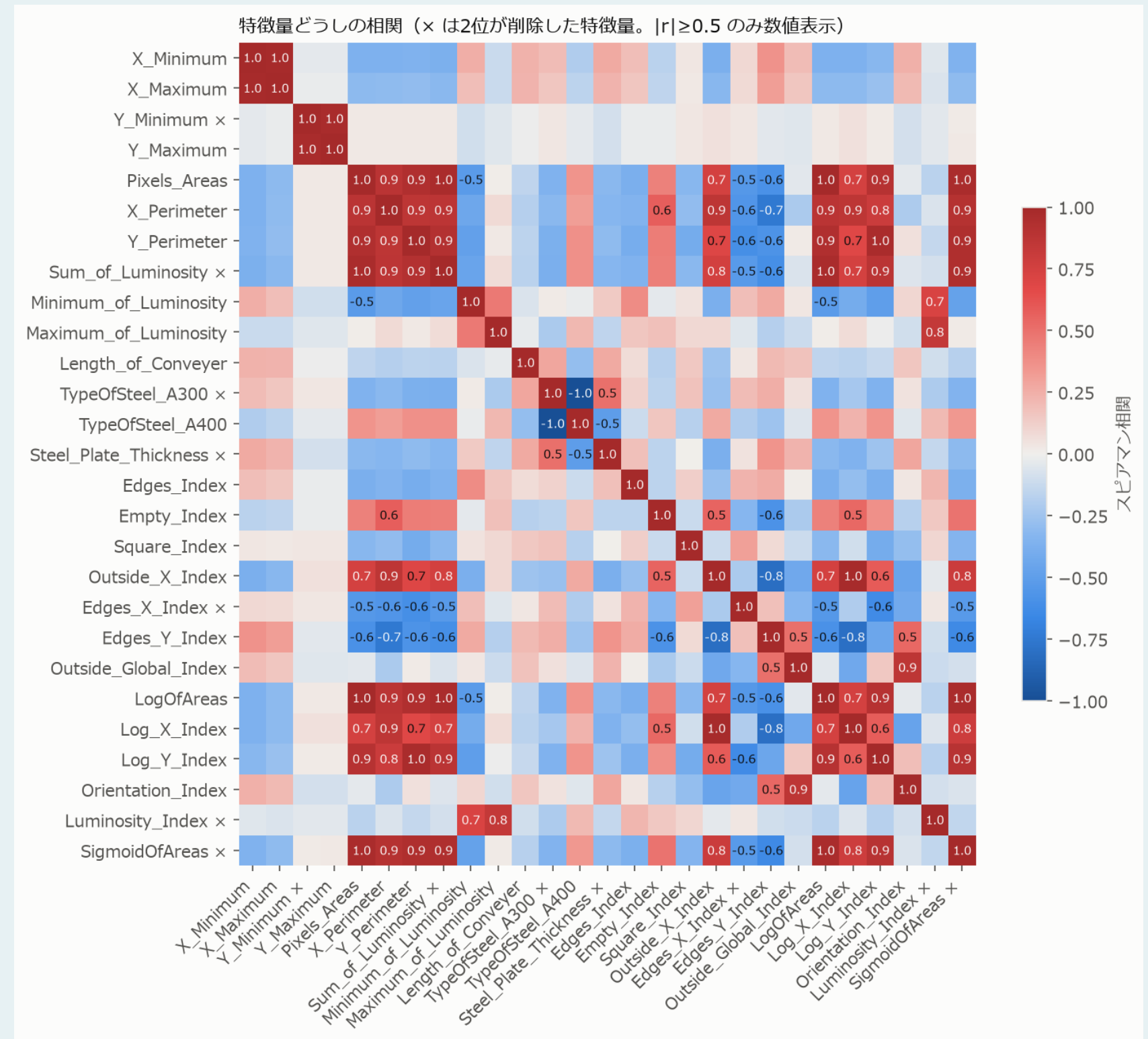
異常	train	test	元データ
横の幅がマイナス	48行	35行	0
縦の幅がマイナス	690行	477行	0
縦の幅が1,000 px 超	412行	264行	1

モデルごとの精度 (OOF)

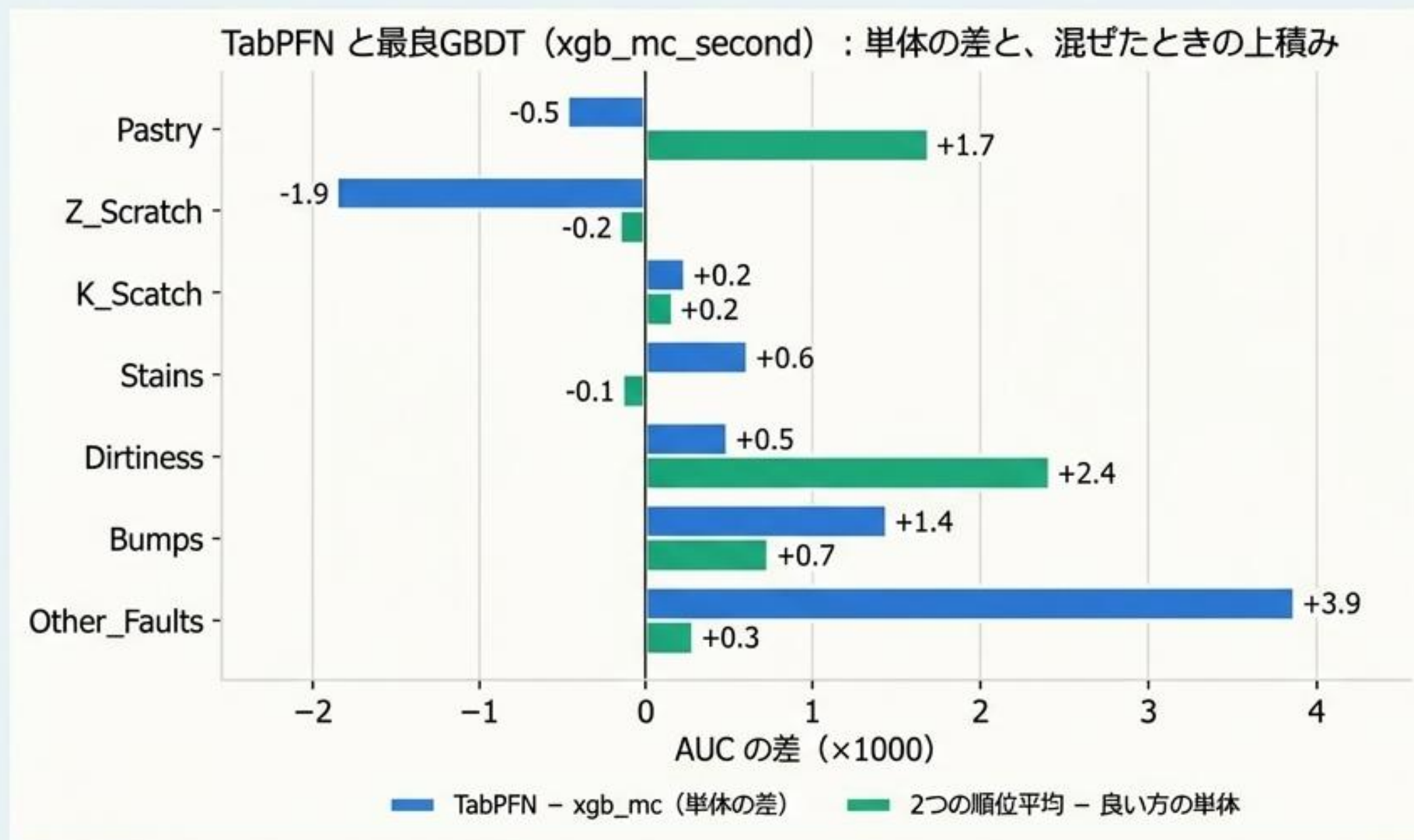
モデル	異常のある行	異常のない行 (同数の無作為抽出の範囲)
最終ブレンド	0.8992	0.8944 (0.885~0.905)
TabPFN	0.8991	0.8926 (0.881~0.903)

EDA⑥ 特徴量どうしの相関と、2位が削除した列

- 削除した7列のうち5列は、残した列とほぼ同じ情報 ($|r|$ 0.83~0.998)
- 面積系の4列は互いに0.94 以上



EDA⑦ TabPFN はなぜ効いたか



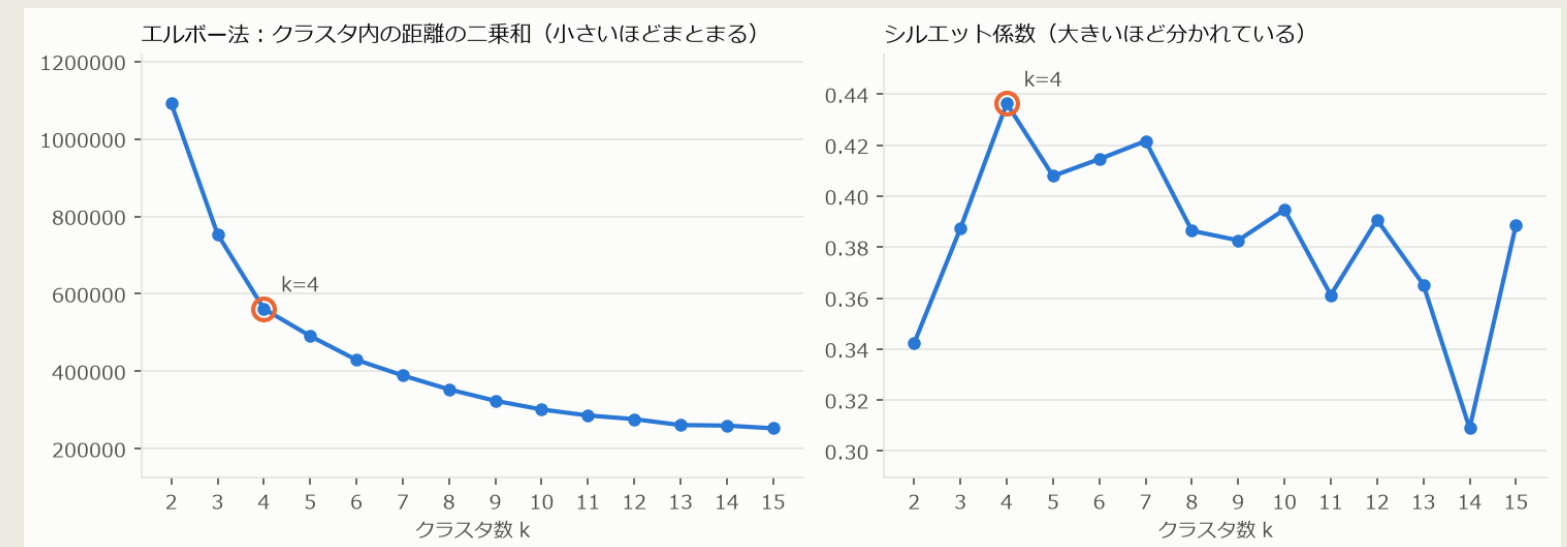
- TabPFN は Other_Faults (+0.0039) と Bumps(+0.0014) で GBDT より強い
- 混ぜると Dirtiness +0.0024、Pastry +0.0017 : 違う行で違う間違いをする

EDA⑧ k-means : クラスタ数はいくつか

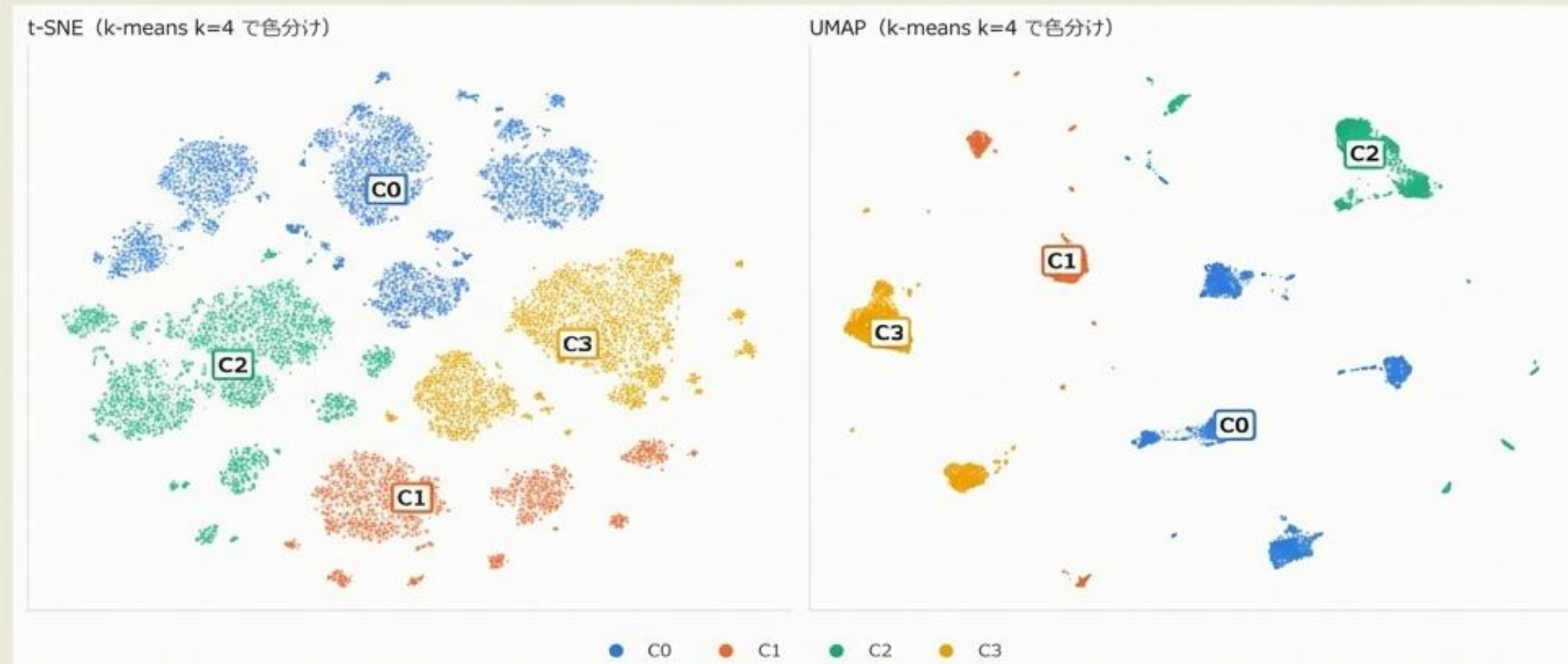
- 欠陥の数（7）がクラスタ数になるとは限らない
→ 指標で決めた
- 適切なクラスタ数は前処理で変わる

前処理と最適なk

前処理	最適な k	シルエット係数
分位点変換	4	0.436
対数変換 + 標準化	2	0.329



EDA⑨ クラスタの正体

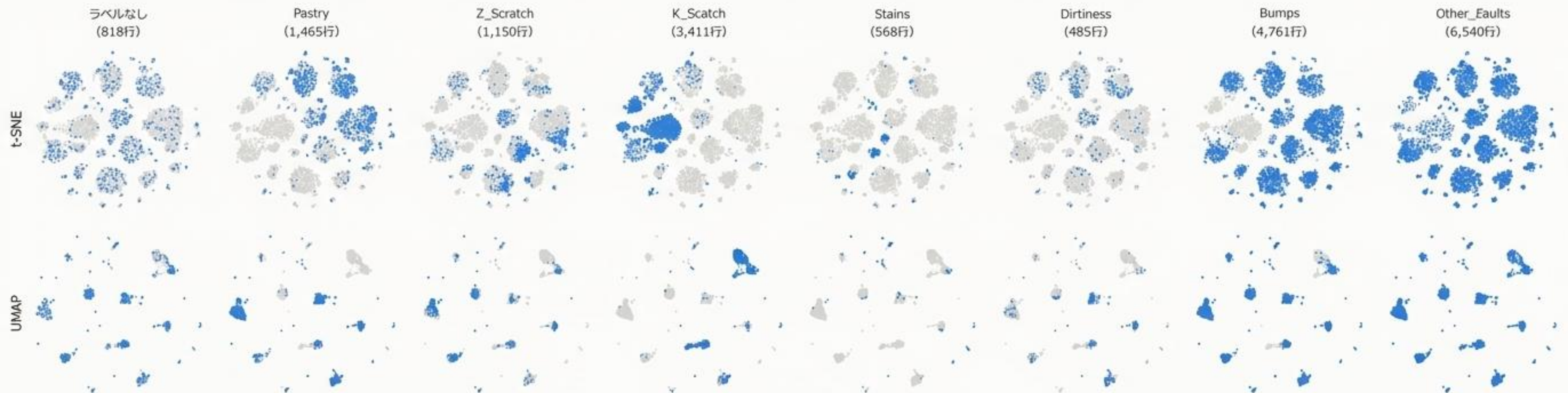


クラスタは欠陥の種類ではなく、製造条件で決まる（ラベルとの一致 ARI 0.08）

クラスタ	銅種	Outside_Global_Index	主な欠陥
C0	A400	1	混在
C1	A300	0	混在
C2	A400	0	K_Scatch
C3	A300	1	混在

EDA⑩ 欠陥ごとに見ると (t-SNE・UMAP)

欠陥の種類ごとの位置 (青 = その欠陥、灰色 = それ以外)



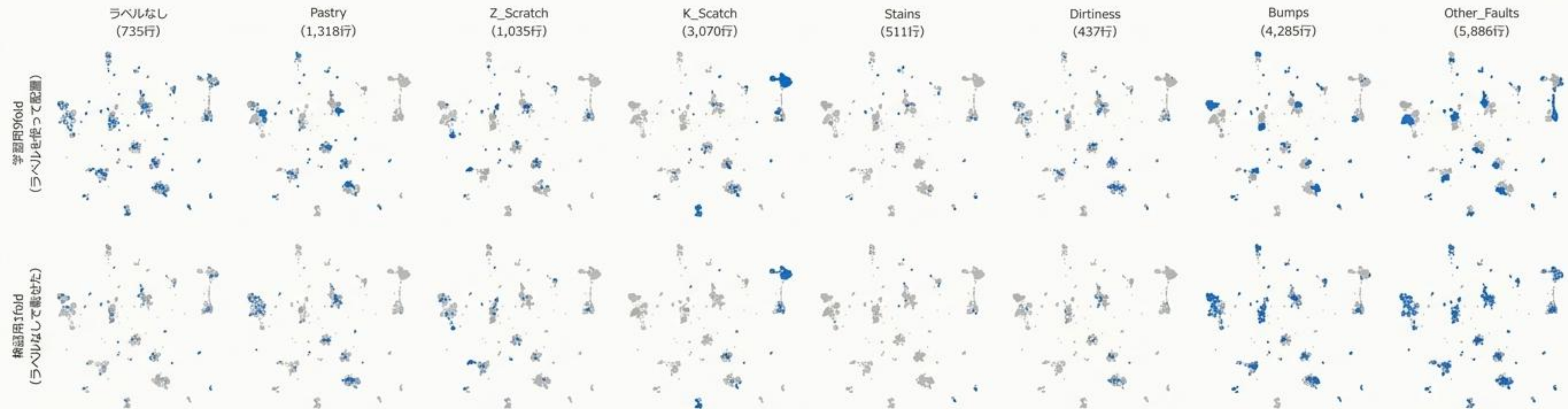
K_Scratch・Stains はまとまる。

Z_Scratch は鋼種 A300 の島に偏る

Bumps・Other_Faults・ラベルなしは、どこにでもいる

EDA⑪ ラベルを入れた UMAP は信じてよいか

教師ありUMAP (fold 0) ; 青 = その欠陥, 灰色 = 学習用の全点



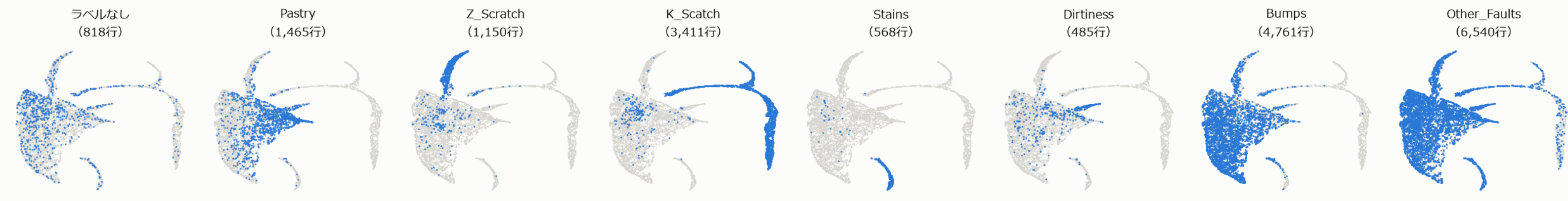
ラベルを入れた UMAP の島の位置・距離は、そのままでは信用できない

方法 (近くの15点の多数決)	正解率
ラベルを入れた UMAP (2次元)	0.449
元の特徴量 (19次元)	0.537
参考: 最終ブレンド	0.620

EDA⑫ モデルから見た欠陥の地図

最終ブレンドの OOF 予測（7列）を UMAP で2次元に

最終ブレンドのOOF予測（7列の確率）のUMAP：青＝その欠陥（正解ラベル）、灰色＝全点



大きなかたまり1つ＋3本の腕（K_Scratch・Z_Scratch・Stains）

青＝その欠陥（正解ラベル）、灰色＝全点。腕の3つはモデルが見分けられる欠陥、かたまりは見分けにくい欠陥

EDAのまとめ

結果で分かったこと	EDA での裏付け
多クラス化が効いた	95.6% が1ラベル、ラベルなし行は Other_Faults に近い
元データの追加が効いた	分布は違うが、学習側だけに使った
2位の特徴量が効いた	削除した列は、残した列とほぼ同じ情報
伸びたのは難しい4ターゲット	見分けにくいかたまり（取り違え・UMAP）
TabPFN が最大の重み	難しいターゲットで GBDT と違う間違いをする
OOF どおりに Private が伸びた	train と test は同じ分布

結論：伸びしろは「見分けにくいかたまり」を割れるかどうかにかかない

第3ラウンド：EDA のヒントを試した

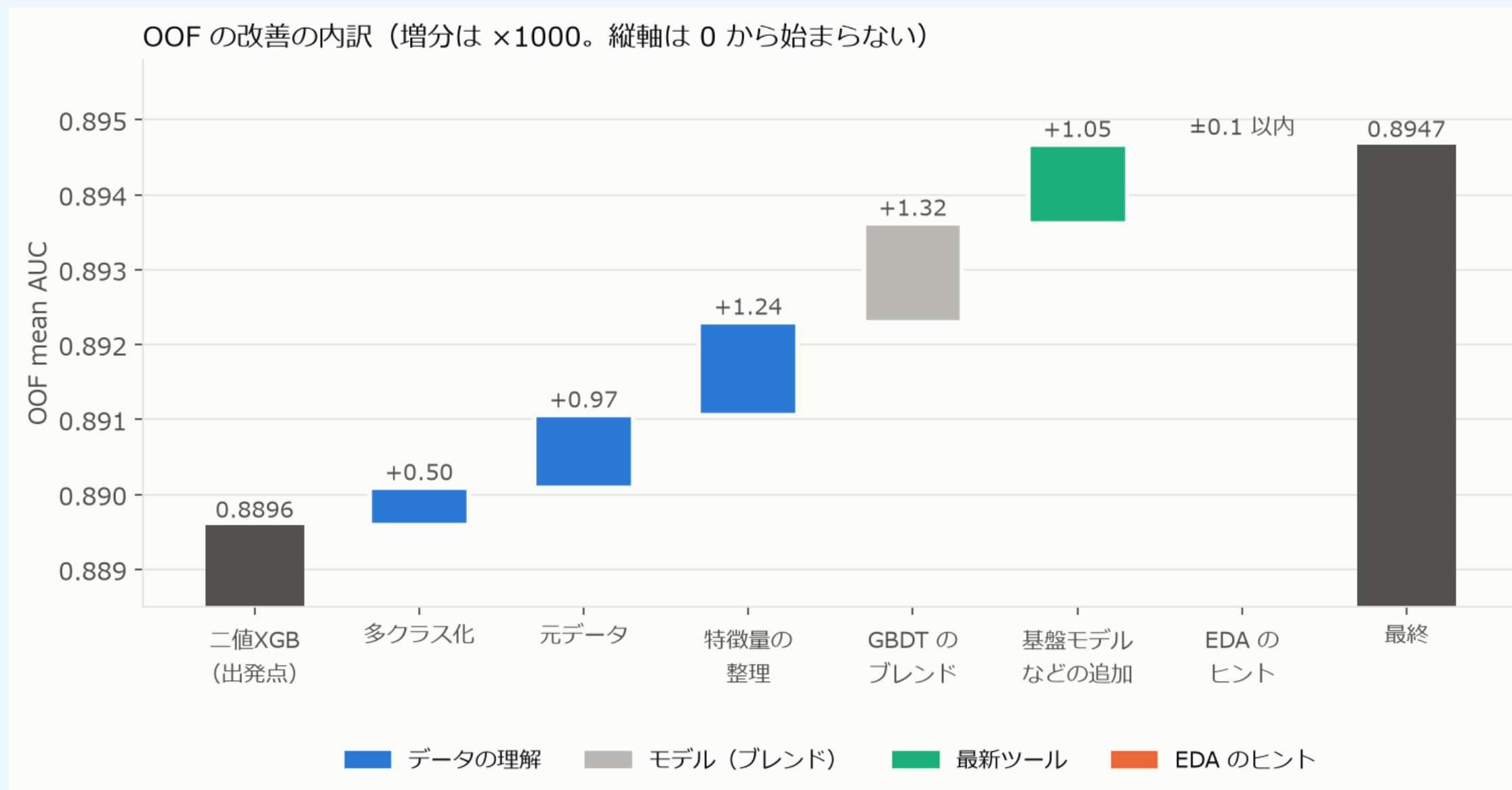
どれも基準（+0.00003）に遠く届かず → 提出なし

EDA の見立てどおり：かたまりには決め手になる特徴量がない

既存10モデルのブレンドに1つ足したときの OOF の変化

追加したもの	ヒント	変化
TabPFN (seed 7)	seed 平均	+0.000008
TabICL (特徴量追加)	特徴量	+0.000004
多クラスXGB (ラベルなし行を除外)	ラベルなし行	+0.000003
2段階目モデル	かたまり専用	-0.000002
多クラスXGB (特徴量追加)	特徴量	-0.000003
ノイズクラス 比率1.0	ノイズ	-0.000005

EDA か、最新ツールか



改善の半分以上はデータの理解から（上位入賞者の EDA の成果を、調査で受け取った）
第3ラウンドの EDA が効かなかったのは、手がかりがもう残っていなかったから

これからの EDA の役割

EDA の役割は「特徴量を作る」から「問題の立て方と検証を決め、伸びしろを見極める」へ。(※1つのコンペからの考察)

変わること	変わらないこと
• 小さく整った表データでは、手で特徴量を作る価値は下がる • 基盤モデルが特徴量どうしの関係を学ぶ	• 問題をどう定式化するか • 検証をどう設計するか • データに異常・リーク・ずれはないか • 伸びしろはどこにあり、上限はどこか

AIエージェントと進めて分かったこと

項目	観察
引き継ぎ	ルール（AGENTS.md）と作業ログで、Codex → Claude Code に途中で交代でできた
再現性	fold 固定と上書き禁止で、別のエージェントがやり直しても 1e-8 の精度で一致
提出	Codex は自動で提出、Claude Code は確認を求め、ユーザーが手動で提出
注意点	ライブラリの自動ダウンロードも事前確認が必要。 利用枠の消費はエージェントで大きく違う

まとめ



- 目標 Private 0.889 を達成 (0.89042)
- 効いたのは定式化と、違う種類のモデルの組み合わせ
- EDA は伸びしろの上限と理由を示した。次の一手は TabPFN のファインチューニング (時間がかかり、効果は不確か)