신용카드 사기 거래 탐지: 1등 풀이 파이프라인(완전 해석) + 검증/시각화 결과 정밀 해설

기반 산출물: viz_checkpoints/20260112_114353 (fig/tables/logs) 및 1등 노트북(ipynb) 실행 흐름

본 보고서는 '코드가 실제로 무엇을 했는지'와 '왜 그렇게 하면 점수가 올라가는지'를 수학적 직관/데이터 관점/검증 근거(그림·표)까지 연결해 완전 해석 형태로 기술한다.

핵심 결론(한 문장): 라벨 없는 데이터에서 사기 비율이 매우 희소하므로, '고순도 후보 생성(Recall) → 후보 내부 정제(Precision)'로 문제를 분해하고, 임베딩(구조 드러내기) + 트리 기반 이상치 점수(경계 만들기) + 비선형 정제(후보 내부 FP 제거)로 private score를 끌어올린다.

목차

Placeholder for table of contents 0

0. 문제 구조 재정의

대회 데이터는 train/test가 라벨이 없고, val에만 라벨(Class)이 존재한다. 규칙상 val로 학습(모델 파라미터 피팅)을 하면 안 되므로, 정통적인 지도학습(Train=라벨, Val=검증) 구조가 불가능하다. 따라서 1등 코드는 이 문제를 다음과 같이 재정의한다.

- train/test = '거의 정상'이지만 라벨이 없는 대규모 거래 집합
- val = 라벨이 있는 소규모 집합(양성 30개) → 학습 금지이므로 '검증/설계 기준'으로만 사용
- 목표 = test에서 사기(양성)만 적당한 개수로 뽑아 제출(극단적 불균형 + private 평가)

이 재정의에서 가장 중요한 관찰은 불균형 규모다. val에서 사기는 30개로 fraud rate는 0.001054이다. 즉 '조금만 틀려도' Precision/Recall이 크게 흔들린다. 그래서 1등 해법은 '한 번에 완벽 분류'가 아니라, 후보를 만들고(candidate generation) 후보를 정제(refinement)하는 다단계 파이프라인을 택한다.

1. 1등 파이프라인 요약(왜 이런 구조인가)

1등 해법의 전체 구조는 아래 3줄로 요약된다.
(A) 정상 분포 학습: train에서 '정상' 패턴을 학습하는 비지도 모델 구축
(B) 고순도 후보 생성: val/test에서 이상치 점수 상위 극소수만 후보로 채택(Recall 확보)
(C) 후보 내부 정제: 후보에서 남아있는 FP를 후보 내부 구조로 제거(Precision 확보)

이 구조가 private score에 유리한 이유:
- 사기 비율이 극단적으로 낮기 때문에, 전체 데이터에서 '조금만 과민'해도 FP가 폭증한다.
- 반대로 너무 보수적이면 FN이 늘어 사기를 놓친다.
- 그러므로 Stage1/Stage3은 '사기를 포함한 후보 집합'을 만들고, 1010은 '후보 내부에서 FP만 깎는 칼' 역할을 한다.

2. 반복형 실행 흐름(처리/학습 ↔ 검증/시각화)

요청하신 구성대로, 본문은 처리/학습 단계 설명 후 바로 이어서 검증/시각화 결과를 해석한다. 각 블록은 다음 질문에 답하도록 작성한다.

- 이 단계의 입력/출력은 무엇인가?
- 이 단계의 핵심 메커니즘은 무엇이며, 왜 필요한가?
- 어떤 실패 모드가 있고, 무엇을 보면 정상 동작인지 확인할 수 있는가?
- 이번 실행 결과(그림/표)는 위 기준을 만족하는가?

3. 블록 1: 데이터 정합성 확보

3.1 (처리/학습) 로딩/스키마/수치 안정성

모든 변환(PaCMAP, KernelPCA)과 모델(IF)은 입력이 수치적으로 깨끗하다는 전제 위에서만 동작한다. 따라서 1등 코드는 로딩 직후 아래를 강제 확인한다.

- 스키마: train/val/test 모두 V1..V30 동일 여부
- 결측/Inf/NaN: 변환/점수 계산이 실패하지 않도록 0이어야 함
- 중복행: 오류는 아니나 밀도 기반/트리 기반 이상치 탐지에서 빈도 효과가 있을 수 있어 확인
- val 라벨 분포: 이후 contamination 및 후보 개수 스케일링의 기준

표 1. 스플릿별 기본 통계(정합성)

split	n	null_total	dup_rows
train	113842	0	245
val	28462	0	11
test	142503	0	294

해석/검증 포인트
- null_total=0이면 전처리 추가(결측 대체)가 필요 없고, 변환이 안정적으로 수행될 조건을 만족한다.
- 중복행이 존재하더라도 반드시 나쁜 것은 아니나, 후보가 '중복 패턴'을 이상치로 오해하지 않는지 후속 단계에서 확인해야 한다.

표 2. Inf/NaN(비유한) 점검

split	finite_ratio
train	1
val	1
test	1

해석/검증 포인트
- finite_ratio=1.0이면 Inf/NaN이 없다는 의미로, PaCMAP/KernelPCA가 '수치 오염' 없이 실행 가능하다.

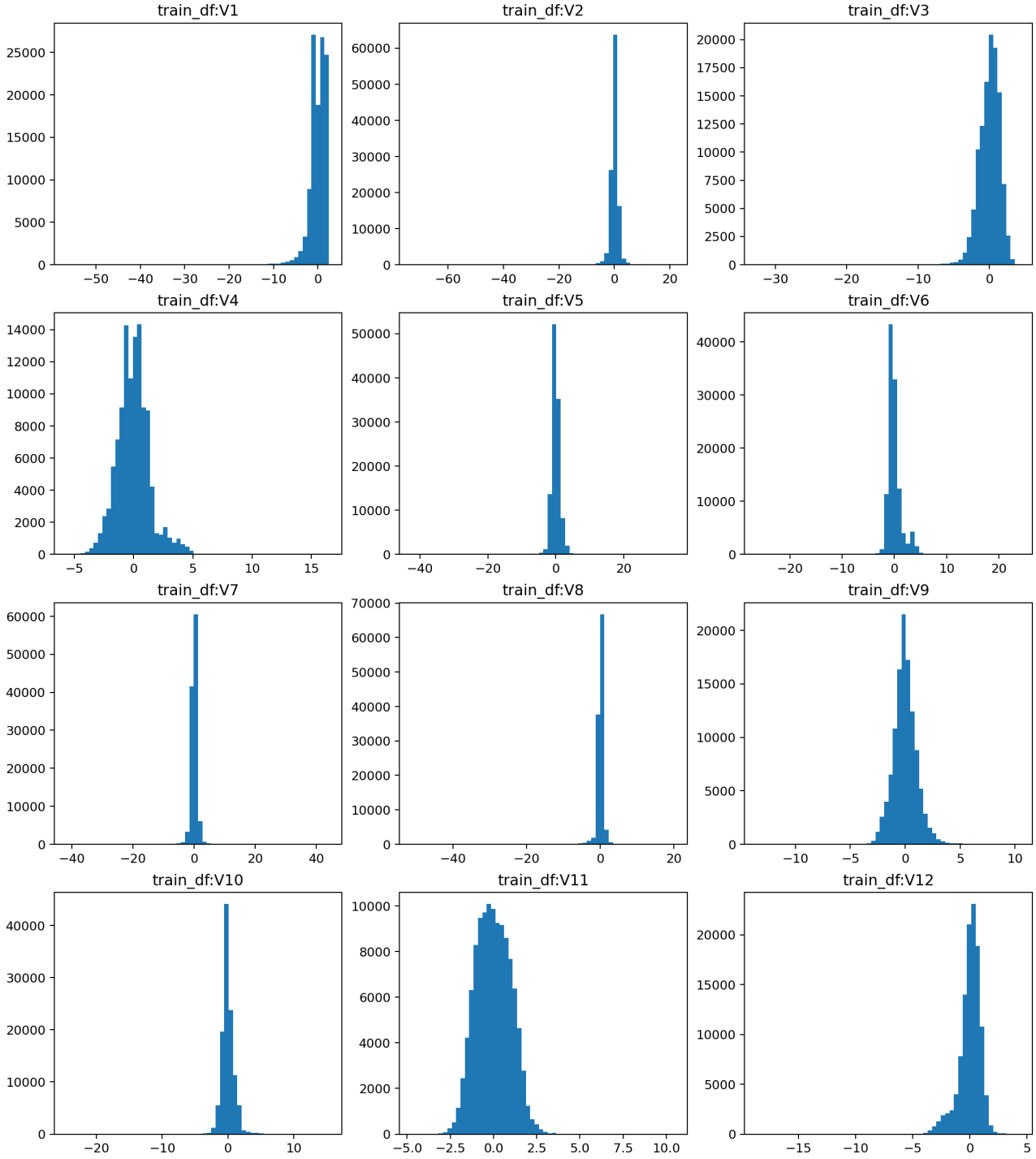
표 3. Validation 라벨 분포(극불균형)

Class	count
0	28432
1	30

해석/검증 포인트
- 양성 30개는 모델이 '확률 예측'처럼 부드럽게 동작하기 어렵게 만든다.
- 따라서 '후보 생성→정제' 구조로 문제를 쪼개는 것이 합리적이다.

3.2 (검증/시각화) 원시 분포 히스토그램

그림 1. Train 원시 분포(상위 12개 피처)

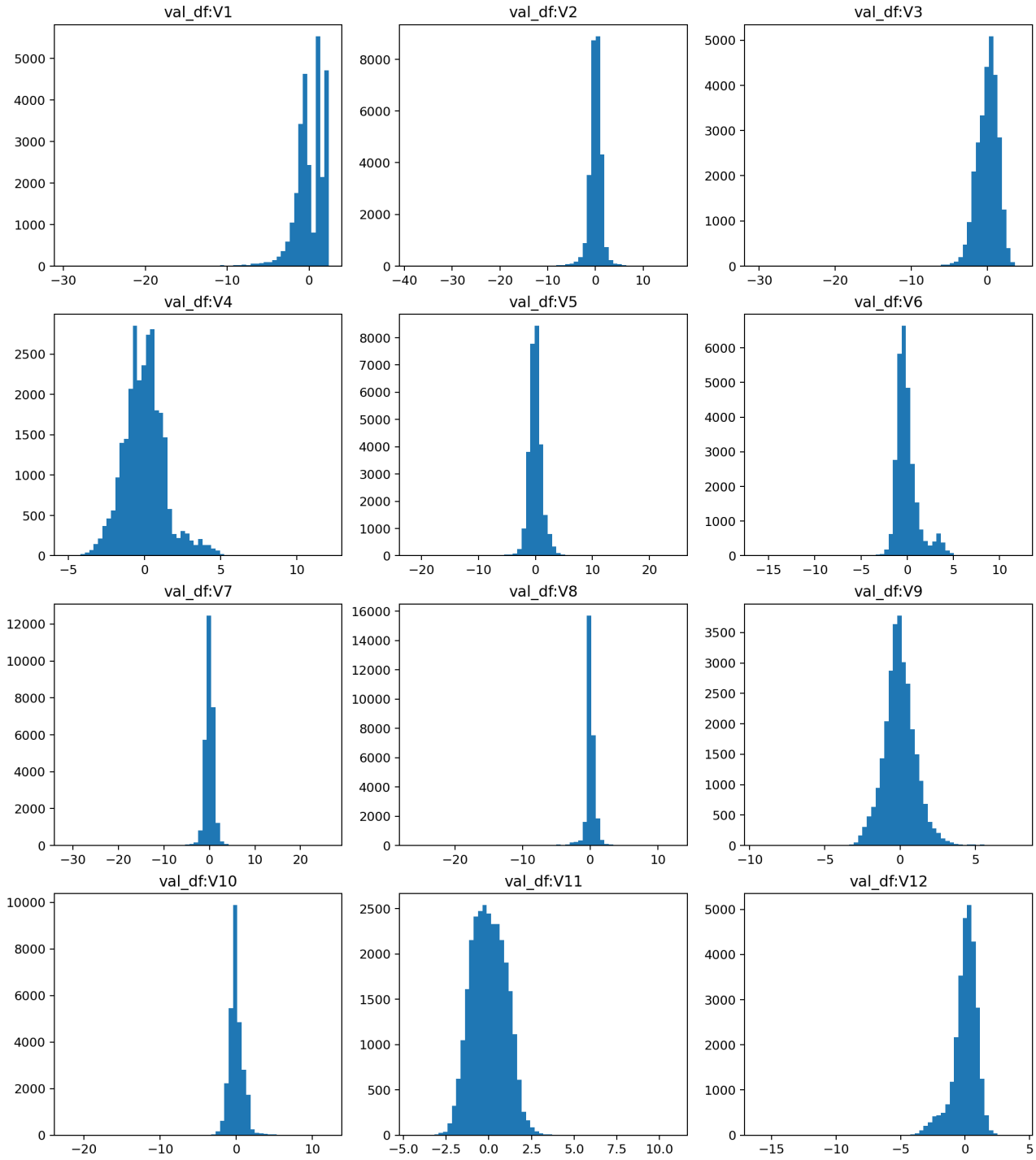


파일: fig/train_df_raw_hist_top12.png

해석/검증 포인트

- 대부분 피처가 0 근처에 몰리고 꼬리가 길어도 정상(익명화된 카드 거래 데이터에서 흔함).
- 이 단계에서 '이상치가 눈에 띄지 않는 것'이 오히려 정상이다(비율이 매우 낮기 때문).

그림 2. Validation 원시 분포(상위 12개 피처)

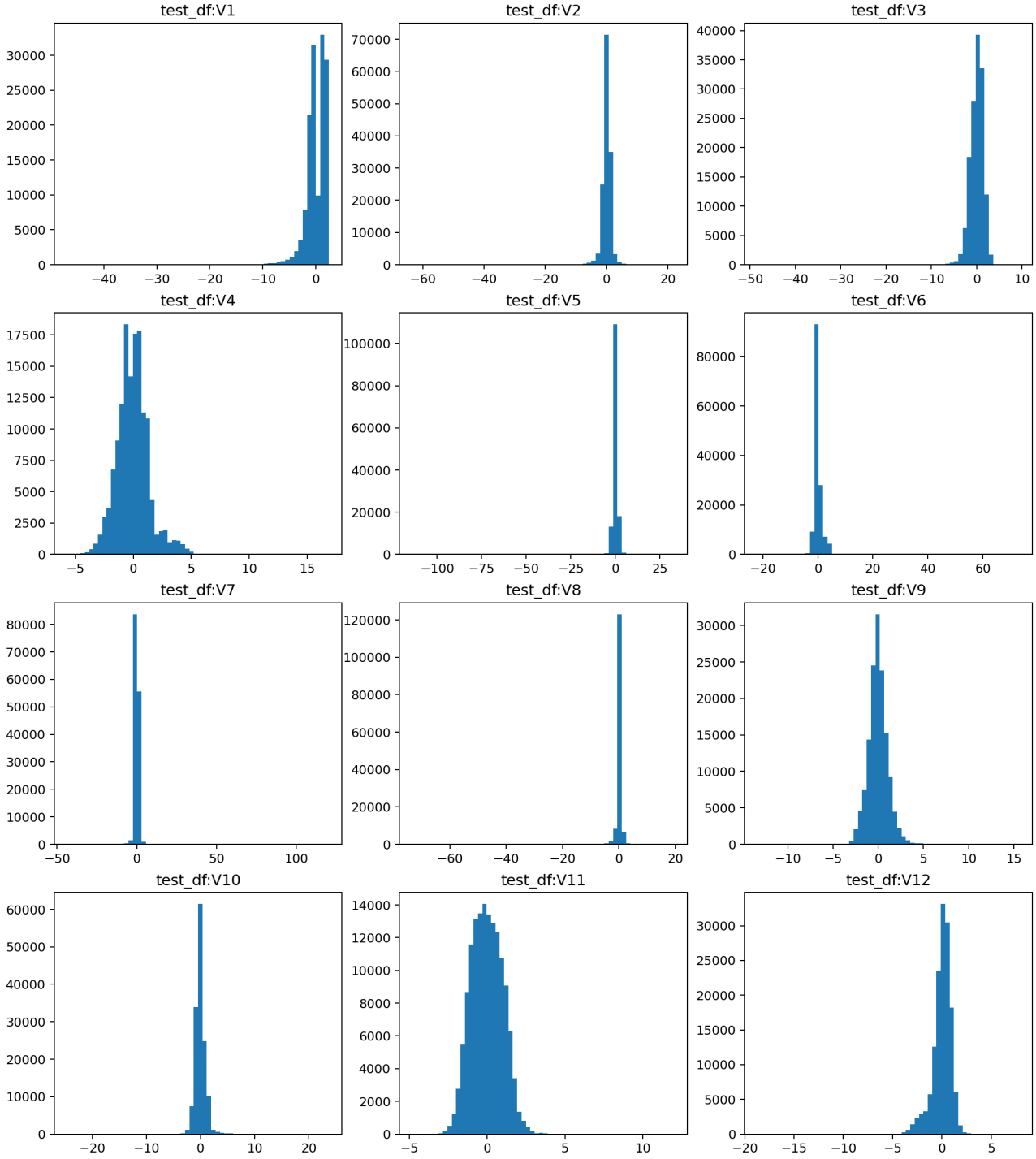


파일: fig/val_df_raw_hist_top12.png

해석/검증 포인트

- train과 형태가 크게 다르지 않다면 train 기반 정상 학습이 val에도 적용될 가능성이 높다.
- val은 학습에 쓰지 않지만, 분포가 심하게 다르면 이후 점수 해석이 깨질 수 있으므로 여기서 1차 확인한다.

그림 3. Test 원시 분포(상위 12개 피처)



파일: fig/test_df_raw_hist_top12.png

해석/검증 포인트

- test 분포가 train과 유사하면 제출에서 FP 폭증 위험이 감소한다.
- 만약 특정 피처가 test에서만 스케일이 다르면(전처리 누락 등), 최종 양성비가 비정상적으로 튈 수 있다.

4. 블록 2: 분포 쉬프트(Train↔Val/Test) 검증

4.1 (처리/학습) 쉬프트 지표의 의미

비지도 이상치 탐지는 'train=정상' 가정이 핵심이다. 그래서 train과 다른 분포를 가진 val/test를 입력하면 정상 샘플도 outlier로 오인(FP 증가)할 수 있다. 이를 막기 위해 PSI/KS/Wasserstein을 계산한다.

- PSI: 구간별 비율 변화(운영 데이터 안정성 점검에 자주 사용)
- KS: 누적분포 함수 차이(분포 형태 변화 탐지)
- Wasserstein: 분포 간 '이동 비용'(꼬리/중심 위치 변화 반영)

4.2 (검증/시각화) 쉬프트 테이블 + 대표 피처 비교

표 4. Train vs Val 쉬프트(PSI 상위 12개)

feature	psi	ks_stat	ks_pvalue	wasserstein
V26	0.000956667	0.00503132	0.609862	0.00397627
V9	0.000941138	0.0094534	0.0339528	0.0104739
V25	0.000876785	0.00604823	0.374036	0.00601702
V14	0.000862909	0.0102568	0.0164994	0.0118392
V19	0.000854638	0.00536829	0.526122	0.00749366
V5	0.000603827	0.00571345	0.44545	0.0169533
V3	0.000579181	0.00810018	0.100231	0.018289
V21	0.000566389	0.00738851	0.165584	0.00707285
V8	0.000467853	0.00313368	0.978215	0.0145554
V17	0.000466016	0.00803337	0.105276	0.00871262
V6	0.000413047	0.00895121	0.0517381	0.024637
V22	0.000384792	0.00459053	0.721557	0.00370509

해석/검증 포인트

- PSI 값이 전반적으로 매우 작으면(0.001 미만 수준) train 정상 경계가 val에 그대로 적용될 여지가 크다.
- 상위 12개만 봐도 큰 값이 없으면, '특정 피처만 붕괴' 같은 위험이 낮다.

표 5. Train vs Test 쉬프트(PSI 상위 12개)

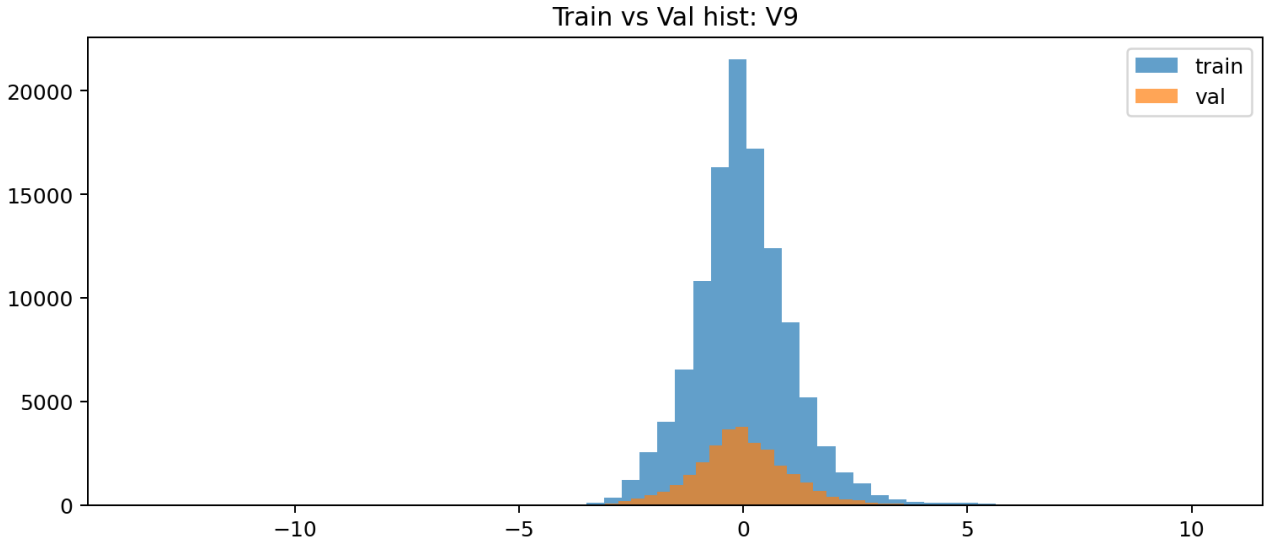
feature	psi	ks_stat	ks_pvalue	wasserstein
V8	0.000223748	0.00453978	0.146777	0.00853673
V3	0.000219325	0.00476251	0.112927	0.017969
V24	0.000200619	0.00306729	0.589821	0.00212194
V6	0.000191788	0.00337825	0.464543	0.00758424
V28	0.000172989	0.00332403	0.485475	0.00386829
V14	0.000172055	0.00369801	0.351436	0.011631

feature	psi	ks_stat	ks_pvalue	wasserstein
V7	0.000170507	0.00321969	0.526958	0.0144874
V25	0.000145875	0.00521471	0.0637869	0.00410991
V19	0.000145317	0.00385588	0.302778	0.00437952
V26	0.000144936	0.00293676	0.644952	0.00199269
V22	0.00014365	0.00245833	0.838041	0.00372027
V10	0.00014301	0.00337966	0.464006	0.00989872

해석/검증 포인트

- train-test 쉬프트가 작으면 제출(test)에서 양성비가 '너무 높거나 0이 되는' 극단 현상이 줄어든다.

그림 4. Train vs Val 히스토그램 비교(V9)

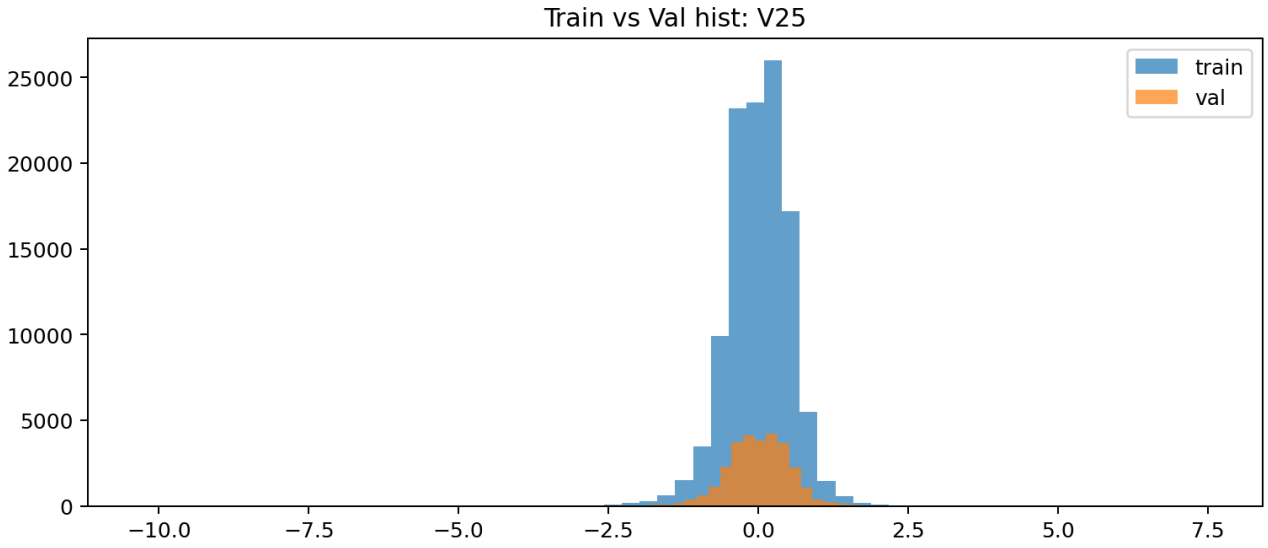


파일: fig/hist_train_vs_val_V9.png

해석/검증 포인트

- 두 히스토그램이 크게 어긋나지 않으면, 해당 피처는 '정상 기준'으로 사용 가능하다.
- 만약 특정 구간에서 val만 튀면, 그 구간은 outlier 점수가 과대평가될 수 있다.

그림 5. Train vs Val 히스토그램 비교(V25)

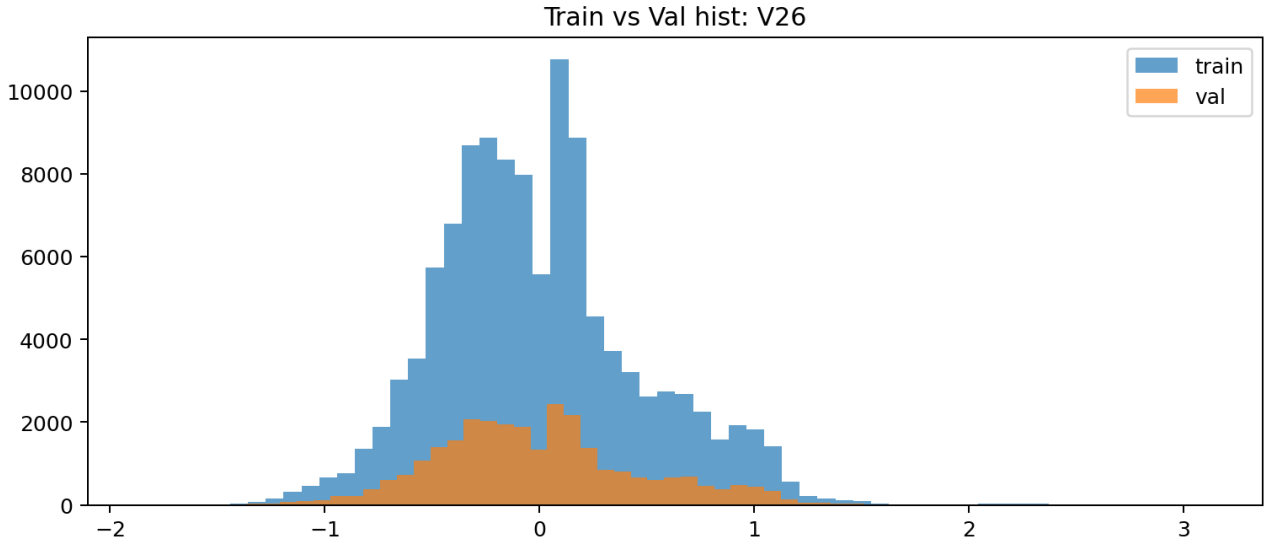


파일: fig/hist_train_vs_val_V25.png

해석/검증 포인트

- 두 히스토그램이 크게 어긋나지 않으면, 해당 피처는 '정상 기준'으로 사용 가능하다.
- 만약 특정 구간에서 val만 튀면, 그 구간은 outlier 점수가 과대평가될 수 있다.

그림 6. Train vs Val 히스토그램 비교(V26)



파일: fig/hist_train_vs_val_V26.png

해석/검증 포인트

- 두 히스토그램이 크게 어긋나지 않으면, 해당 피처는 '정상 기준'으로 사용 가능하다.
- 만약 특정 구간에서 val만 튀면, 그 구간은 outlier 점수가 과대평가될 수 있다.

5. 블록 3: 피처 선택(라벨 기반 통계) + contamination 스케일링

5.1 (처리/학습) 라벨을 '학습'이 아닌 '피처 선택'에만 쓰는 이유

val 라벨을 모델 피팅에 쓰면 규칙 위반이 될 수 있고, 또한 극소수 양성(30개)은 과적합 위험이 높다.
하지만 피처가 어떤 방향으로 사기와 차이가 나는지(분포 차이)는 val 라벨로 '통계적으로' 판단할 수 있다.
1등 코드는 이런 목적에서 Wilcoxon 같은 비모수 검정을 통해 피처를 그룹화한다.

핵심 직관: 비지도 모델(IF)은 라벨을 모르지만, 입력 피처가 Class0/1을 잘 벌려주면(분포가 다르면) 정상 학습 대비 이탈(outlier) 점수가 더 선명해진다.

5.2 (처리/학습) contamination 결정의 의미(후보 수 스케일)

표 6. fraud rate 기반 contamination/기대 사기 수

item	value
val fraud_rate	0.00105404
contamination(used)	0.00105515
train expected fraud (≈rate*n)	119.994
test expected fraud (≈rate*n)	150.203

해석/검증 포인트

- contamination은 IF의 outlier 컷오프에 직접 영향 → 후보 개수(양성 예측 수)의 1차 스케일을 만든다.
- 기대 사기 수가 '수만 개'로 나오면 contamination이 너무 큰 것이고, '0~수 개'면 너무 작은 것이다.

5.3 (검증/시각화) 피처 그룹(선택 결과)와 분포 차이

표 7. superior_var1: 선택된 피처 목록

idx	col
1	V2
2	V3
3	V4
6	V7
8	V9
9	V10
10	V11
11	V12
13	V14
15	V16
16	V17
17	V18

해석/검증 포인트
- 피처 수가 너무 적으면 모델이 정보를 못 담고, 너무 많으면 노이즈가 섞일 수 있다.

표 8. superior_var2: 선택된 피처 목록

idx	col
3	V4
9	V10
10	V11
11	V12
13	V14
15	V16
16	V17

표 9. inferior_var2: 비교용 피처 목록

idx	col
0	V1
1	V2
2	V3
4	V5
5	V6
6	V7
7	V8
8	V9
12	V13
14	V15
17	V18
18	V19
19	V20
20	V21
21	V22
22	V23
23	V24
24	V25
25	V26
26	V27

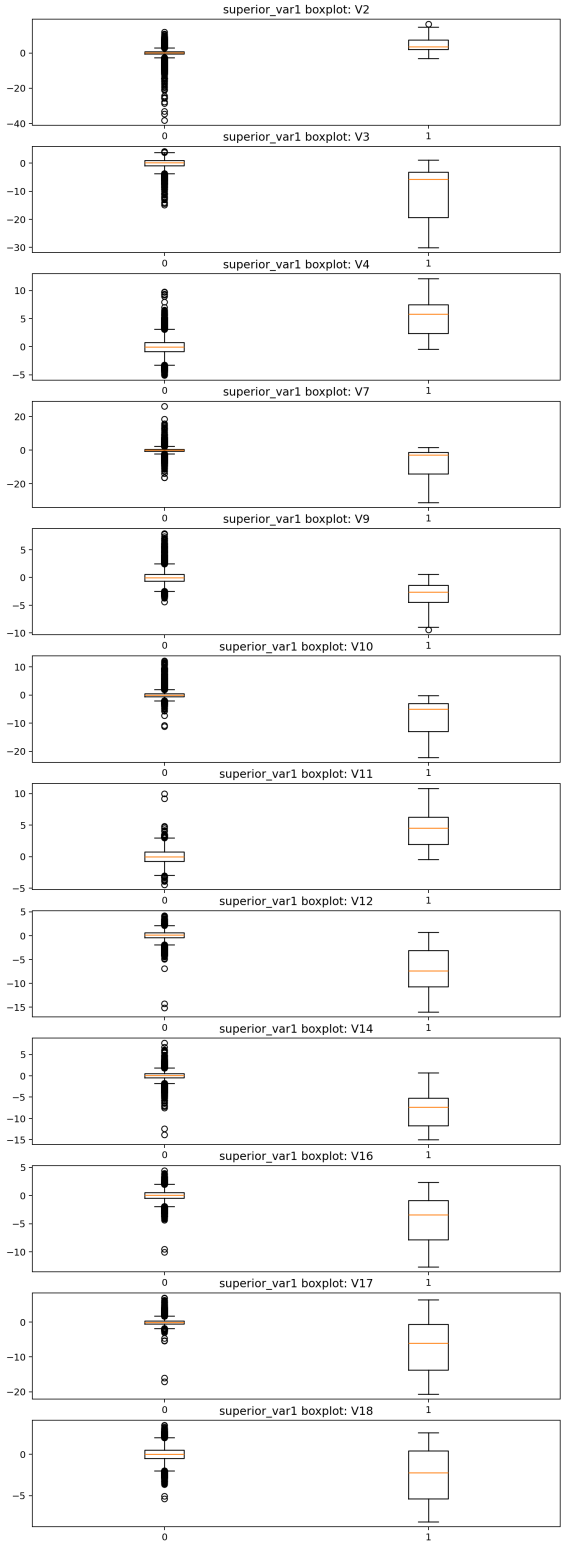
표 10. superior_var1: Class별 통계(상위 12)

feature	mean_0	std_0	mean_1	std_1	delta_mean
V4	-0.00386507	1.39025	5.45856	3.39891	5.46242
V2	-0.00303429	1.59221	4.78657	4.34981	4.7896
V11	-0.000684565	1.0023	4.45015	2.89608	4.45084
V18	0.00833401	0.823085	-2.45956	3.25429	-2.4679
V9	-0.00140853	1.08283	-3.31825	2.584	-3.31685
V16	0.00967232	0.839691	-4.50607	4.48055	-4.51574
V12	0.0101802	0.947581	-7.29493	5.18709	-7.30511
V17	0.0153025	0.748009	-7.33768	7.4835	-7.35299
V10	0.00569154	1.03752	-7.5751	6.1909	-7.5808
V7	0.0136854	1.10186	-7.6993	8.8213	-7.71299
V14	0.020558	0.893263	-7.81511	4.30124	-7.83567
V3	0.0117582	1.43806	-9.80242	8.81549	-9.81418

해석/검증 포인트

- Class1 평균/중앙값이 Class0와 체계적으로 다르면 '분리 단서'가 되는 피처다.
- 표준편차가 지나치게 크면 꼬리값(outlier)이 많아, 스케일링/임베딩에서 영향이 커질 수 있다.

그림 7. superior_var1 박스플롯(Class별)

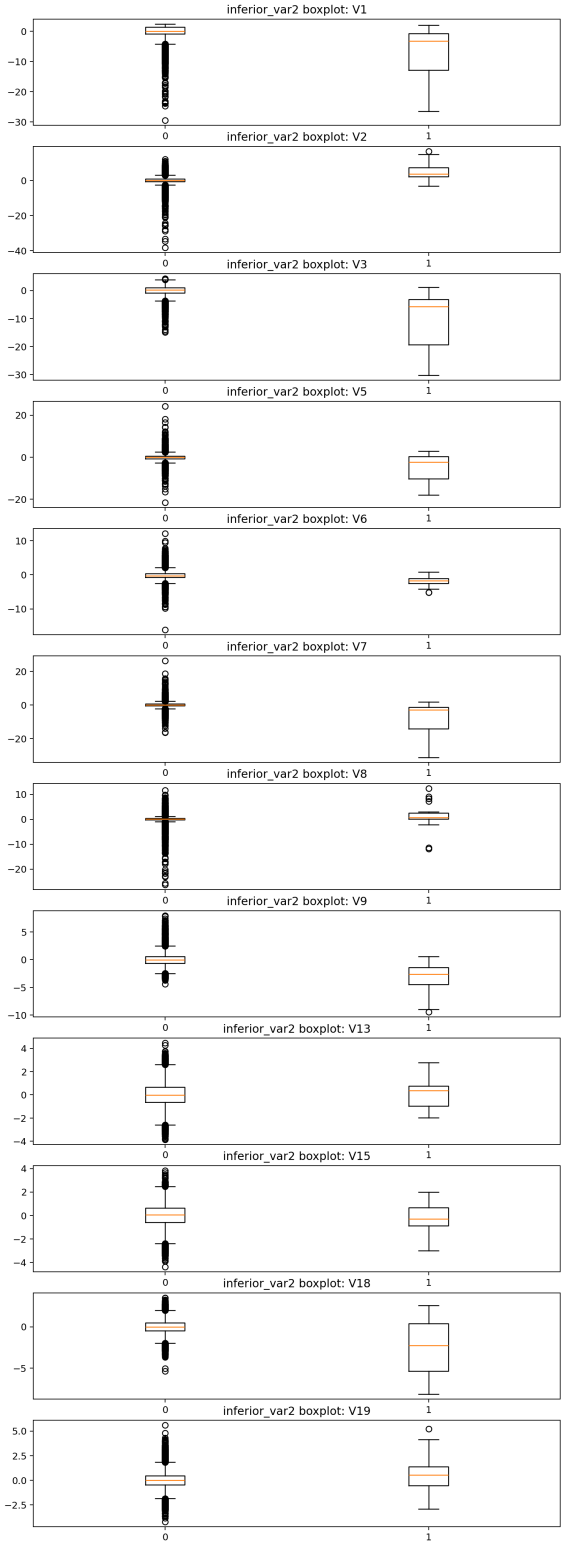


파일: fig/superior_var1_boxplots_top12.png

해석/검증 포인트

- 상자 위치가 분리되거나 꼬리 방향이 다른 피처가 많을수록, 비지도 점수가 사기를 상위로 올릴 가능성이 커진다.

그림 8. inferior_var2 박스플롯(Class별)



파일: fig/inferior_var2_boxplots_top12.png

해석/검증 포인트

- 겹침이 큰 피처는 단독 분리력은 낮지만, 조합/비선형 변환에서 보조 신호가 될 수 있다.

6. 블록 4: Stage1 후보 생성(PaCMAP 임베딩 + IsolationForest)

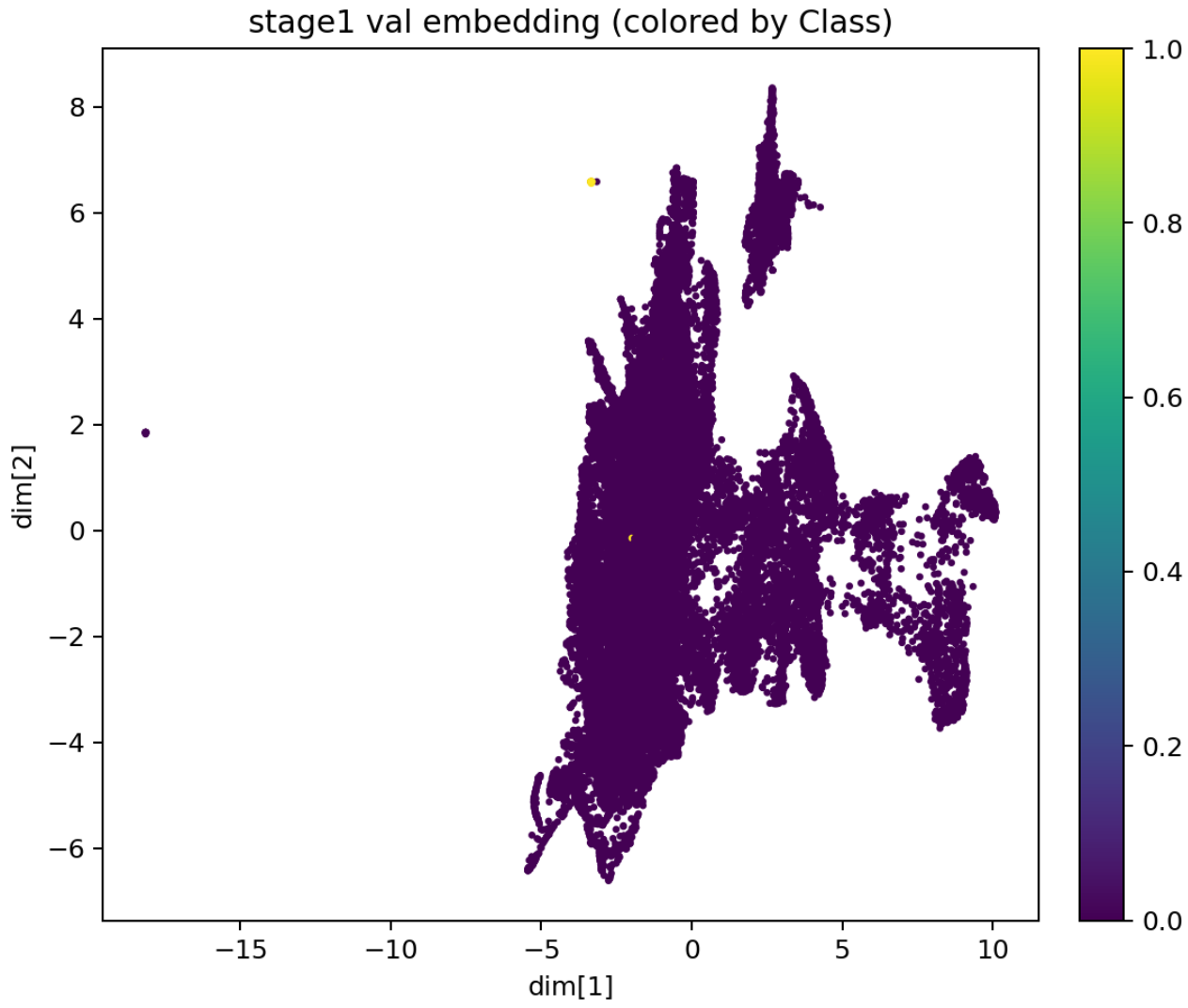
6.1 (처리/학습) Stage1 메커니즘을 '수식/직관'으로 풀어쓰기

PaCMAP은 고차원 데이터를 2D/3D로 내리면서, ① 근접점(로컬)과 ② 멀리 있는 점(글로벌)의 관계를 동시에 보존하려고 한다. 이 과정에서 '군집 바깥'에 있는 점이 시각적으로 드러날 가능성이 높다. IsolationForest(IF)는 데이터를 랜덤 분할하는 트리를 여러 개 만들고, 어떤 샘플이 '평균적으로 얼마나 빨리 고립(isolate)되는지'로 이상치 점수를 준다. 대략적으로, 고립 깊이가 얕을수록(outlier) 점수가 커진다.

Stage1은 train으로 IF를 피팅하고, val/test에서는 점수만 계산한다. 그리고 contamination에 의해 결정되는 컷오프(상위 극소수)를 1로 두어 후보를 만든다.

6.2 (검증/시각화) Stage1 결과가 '정상 동작'인지 판단

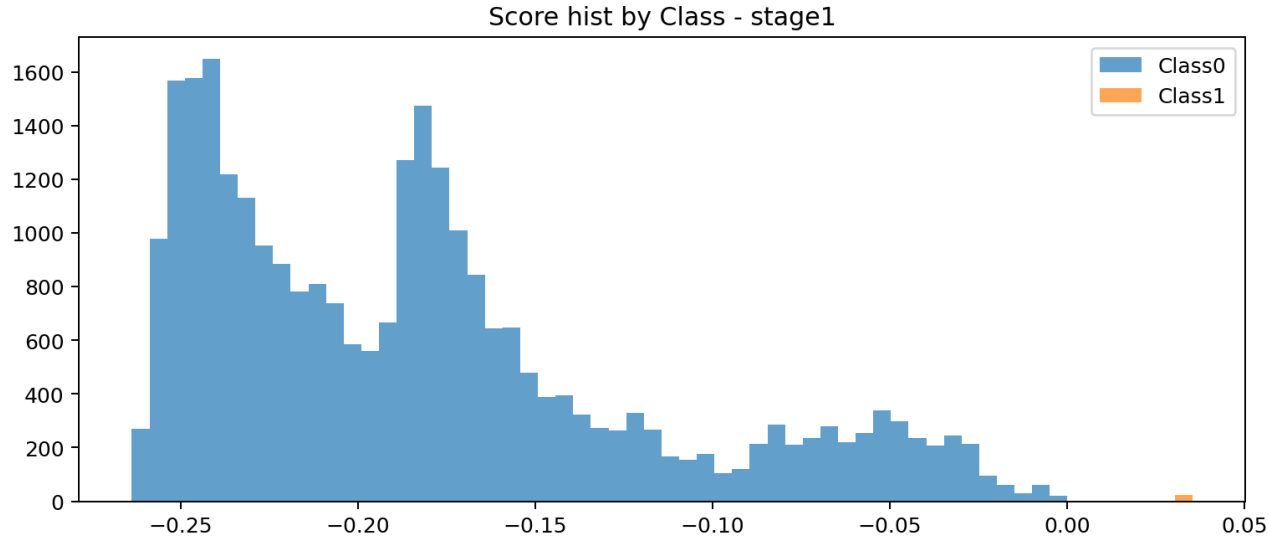
그림 9. Stage1 val 임베딩 2D(색=Class)



파일: fig/stage1_val_scatter2d_byClass.png

해석/검증 포인트
- 이상적이면 Class1 점들이 '밀집 정상 군집 바깥' 또는 경계부에 더 많이 위치한다.
- 겹침이 있어도 정상이다(사기 30개는 매우 적고, feature가 완벽히 분리하지 못함).
- 중요한 건 '후보로 찍힌 점'이 이탈 영역에 주로 위치하는지(후속 점수/예측 분포로 확인).

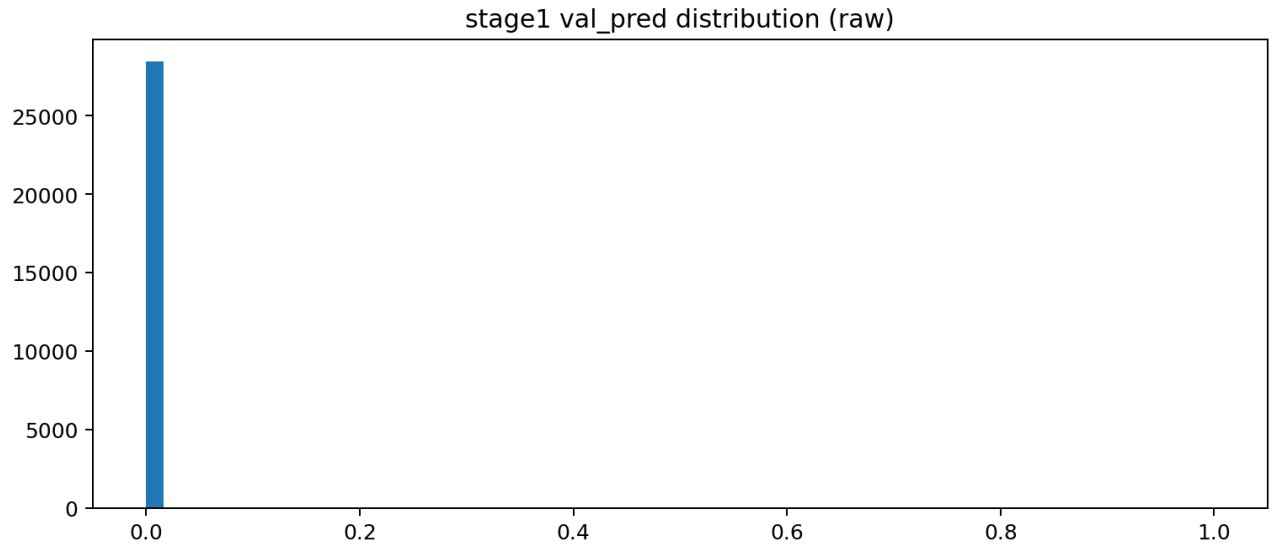
그림 10. Stage1 점수 분포(Class별)



파일: fig/stage1_score_hist_by_class.png

해석/검증 포인트
- 사기(Class1)의 점수 분포가 더 높은 영역으로 치우치면, 점수가 '올바른 순위'를 제공한다.
- 이 그림은 AUC와 연결된다: 분포가 분리될수록 AUC가 올라간다.

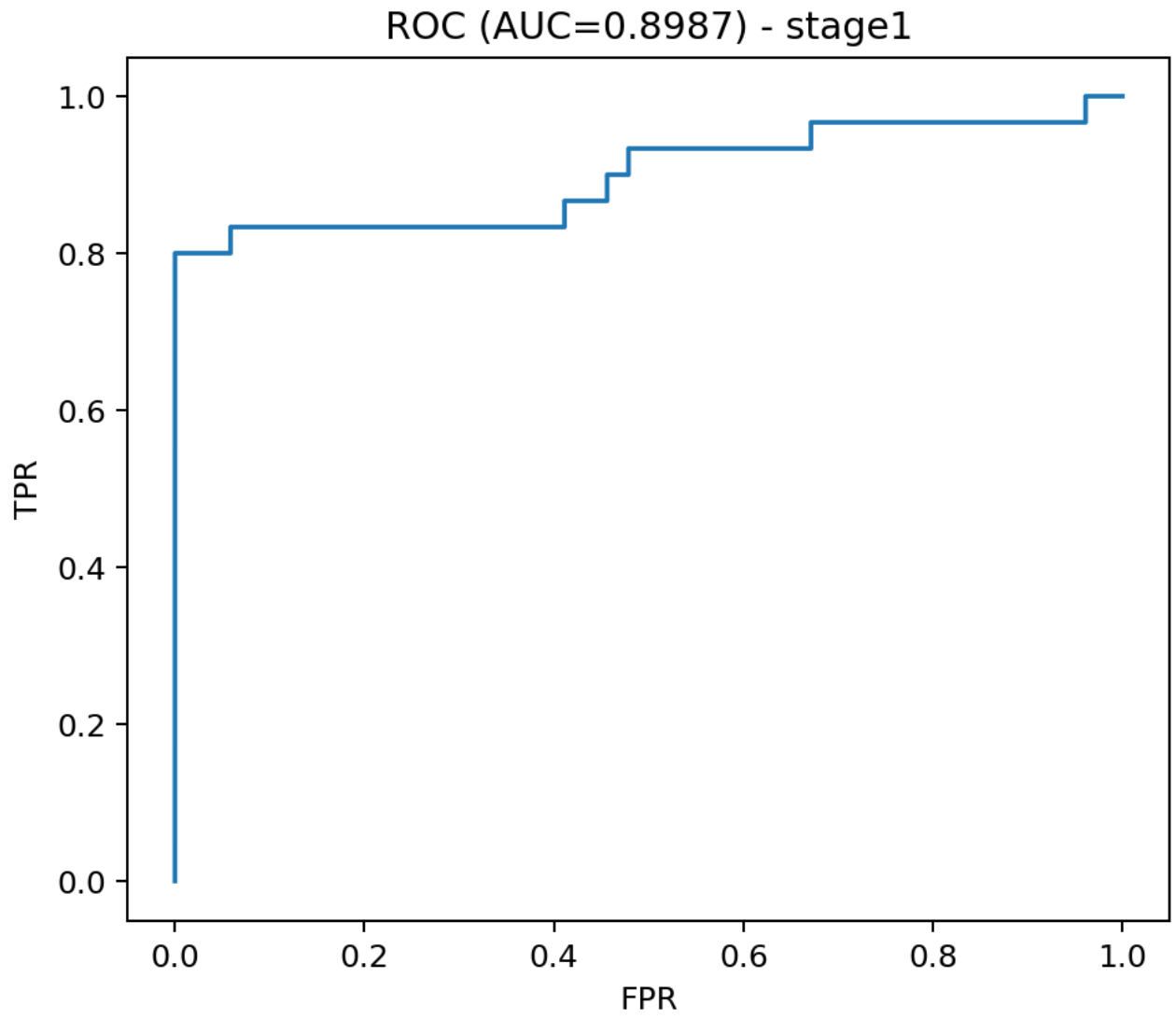
그림 11. Stage1 val 예측값 분포(raw)



파일: fig/stage1_val_pred_hist.png

해석/검증 포인트
- 대부분이 0이고 극소수만 1이면 '후보 생성'이 과민하지 않다는 신호다.
- 여기서 1이 너무 많으면 FP가 폭증하고, 너무 적으면 FN이 증가한다.

그림 12. Stage1 ROC(AUC)

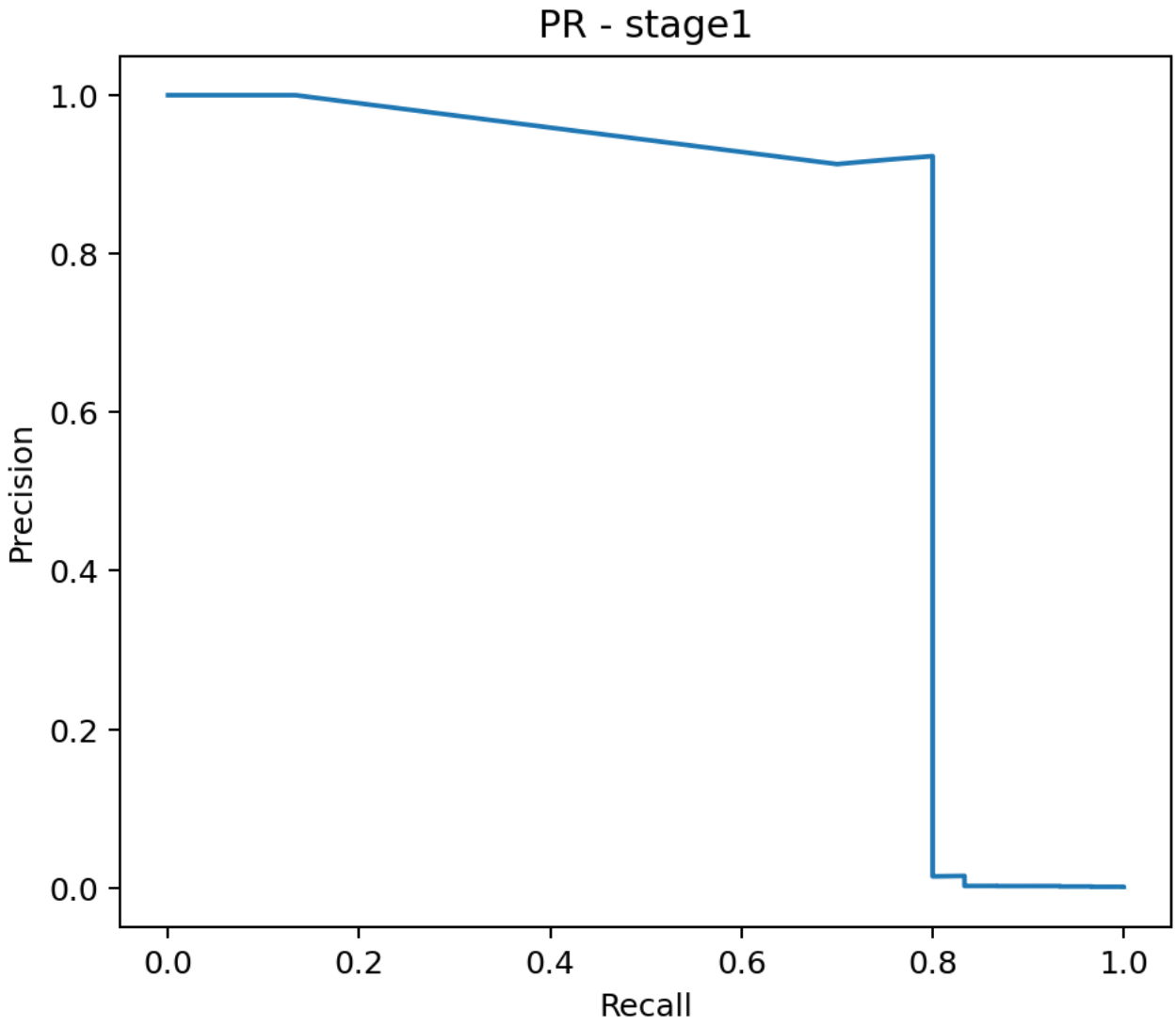


파일: fig/stage1_roc.png

해석/검증 포인트

- 이번 실행에서 AUC=0.8987. 극불균형에서도 점수의 순위 분리력이 존재함을 의미한다.

그림 13. Stage1 PR



파일: fig/stage1_pr.png

해석/검증 포인트

- PR은 '양성을 얼마나 깨끗하게 뽑는가'를 보여준다.
- 후보 생성 단계에서는 Recall을 확보하면서 Precision을 최대화하는 절충이 중요하다.

표 11. Stage1 혼동행렬(전체 val)

true	pred0	pred1
true0	28429	3
true1	6	24

해석/검증 포인트

- TP=24, FN=6: 사기 30개 중 24개를 후보로 포함(Recall=0.800).
- FP=3: 정상 28432개 중 3개만 후보로 포함(정상 기준 FP rate≈0.000106).
- 후보 27개 중 사기 비율은 0.889로 매우 높아(≈0.889), '후보 생성' 목표를 달성.

6.3 (검증) 저장/재로딩 일관성(재현성, 인덱스 정렬)

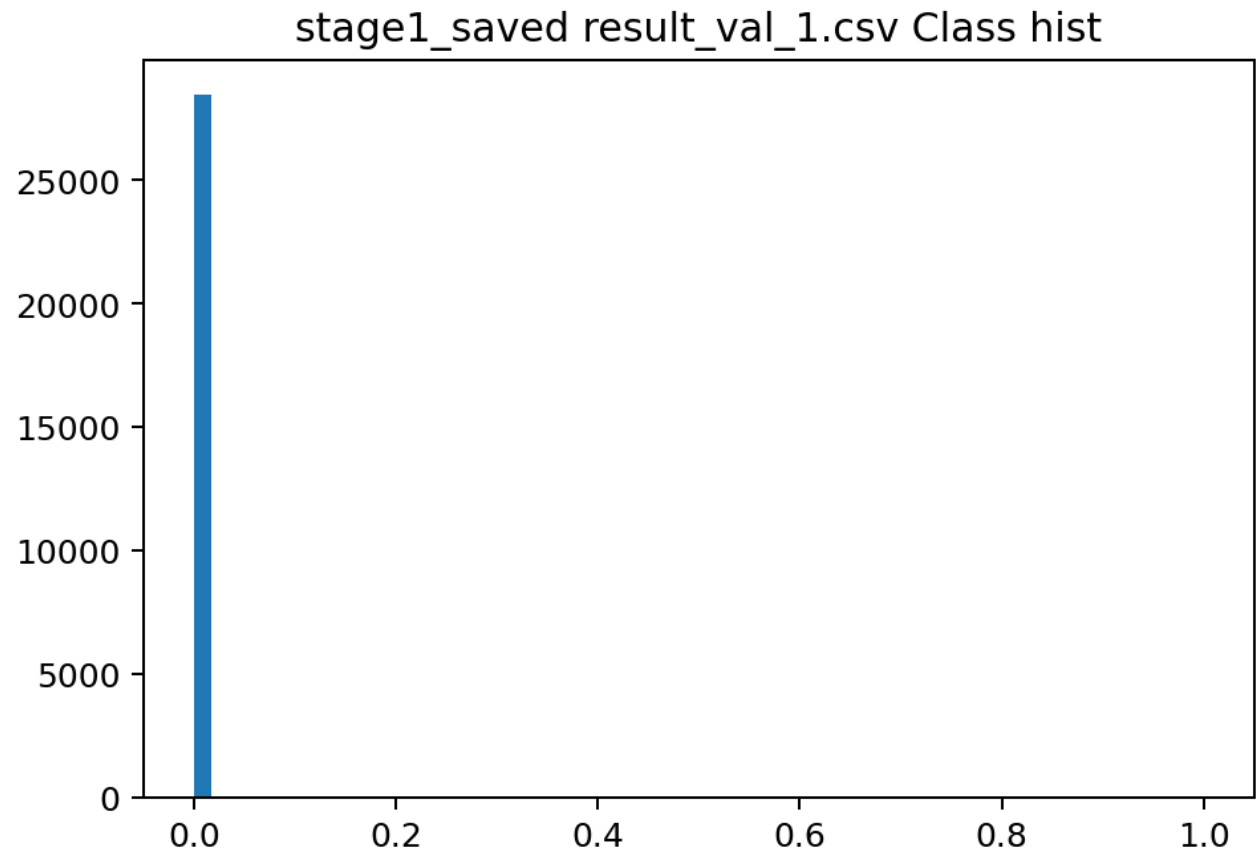
표 12. Stage1 저장/재로딩 결과 동일성

file	equal_to_memory
result_train_1.csv	True
result_val_1.csv	True
result_test_1.csv	True

해석/검증 포인트

- True이면 디스크 저장/로딩 과정에서 컬럼 순서, dtype, 인덱스 정렬이 깨지지 않았다는 의미다.
- 이 검증이 없으면 Stage3/1010에서 '길이 불일치', '다른 샘플에 라벨 매핑' 같은 치명적 버그가 발생할 수 있다.

그림 14. Stage1 저장 결과(val) Class 분포



파일: fig/stage1_saved_result_val_1.csv_hist.png

해석/검증 포인트

- 저장 파일에서도 후보의 희소성이 유지되면(대부분 0), 저장/로딩으로 임계값이 바뀌지 않았음을 시사한다.

7. 블록 5: Stage3 안정화(피처 보강 + 보팅)

7.1 (처리/학습) Stage3 메커니즘: 랜덤성을 '보팅'으로 상쇄

PaCMAP은 시드/초기화에 따라 임베딩이 조금씩 달라질 수 있다. 이 변동은 '후보가 바뀌는 현상'으로 이어질 수 있다. Stage3는 이 문제를 다음 두 방법으로 완화한다.

- 피처 보강: new_born_idx, dhk_add_list 등 추가 피처를 더해 사기/정상의 구조적 차이를 강화
- 다회 실행 + 보팅: 여러 번 실행했을 때 반복적으로 outlier로 뽑히는 샘플만 최종 후보로 확정

7.2 (검증/시각화) 추가 피처가 실제로 분리 단서를 제공하는가

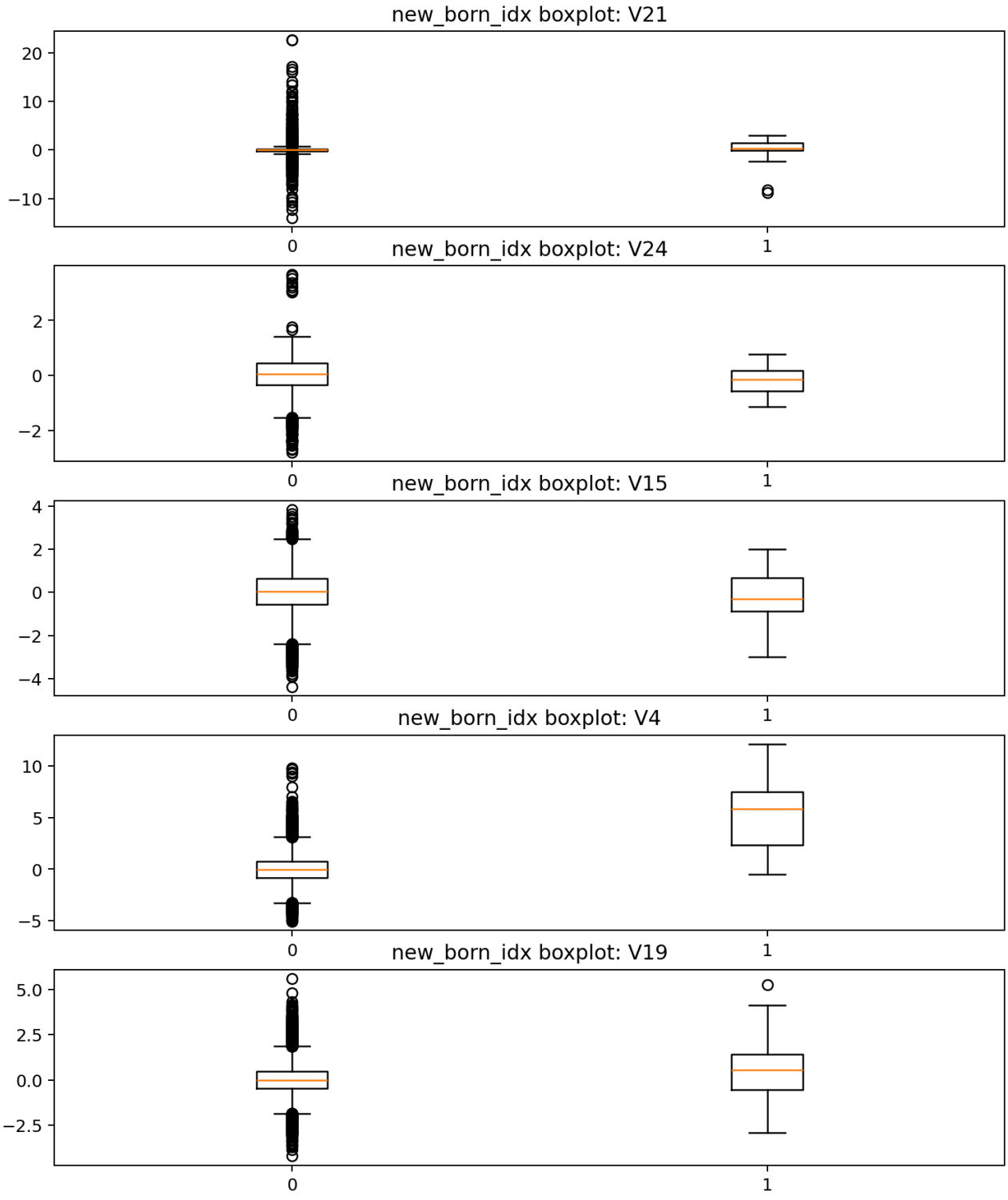
표 13. new_born_idx 피처 매핑

idx	col
20	V21
23	V24
14	V15
3	V4
18	V19

표 14. dhk_add_list 피처 매핑

idx	col
23	V24

그림 15. new_born_idx 박스플롯(Class별)

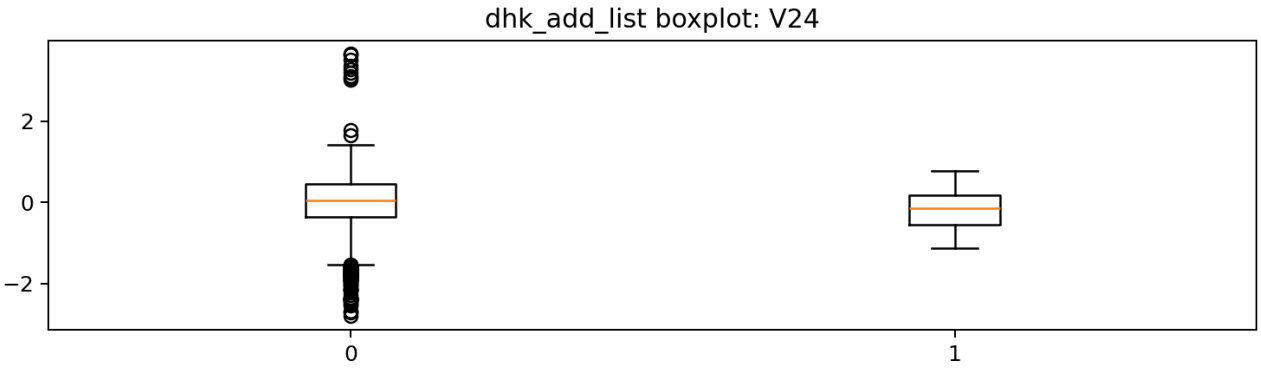


파일: fig/new_born_idx_boxplots.png

해석/검증 포인트

- Class별 중앙값/사분위 범위가 체계적으로 다르면 Stage3 입력 강화로 작동한다.

그림 16. dhk_add_list 박스플롯(Class별)



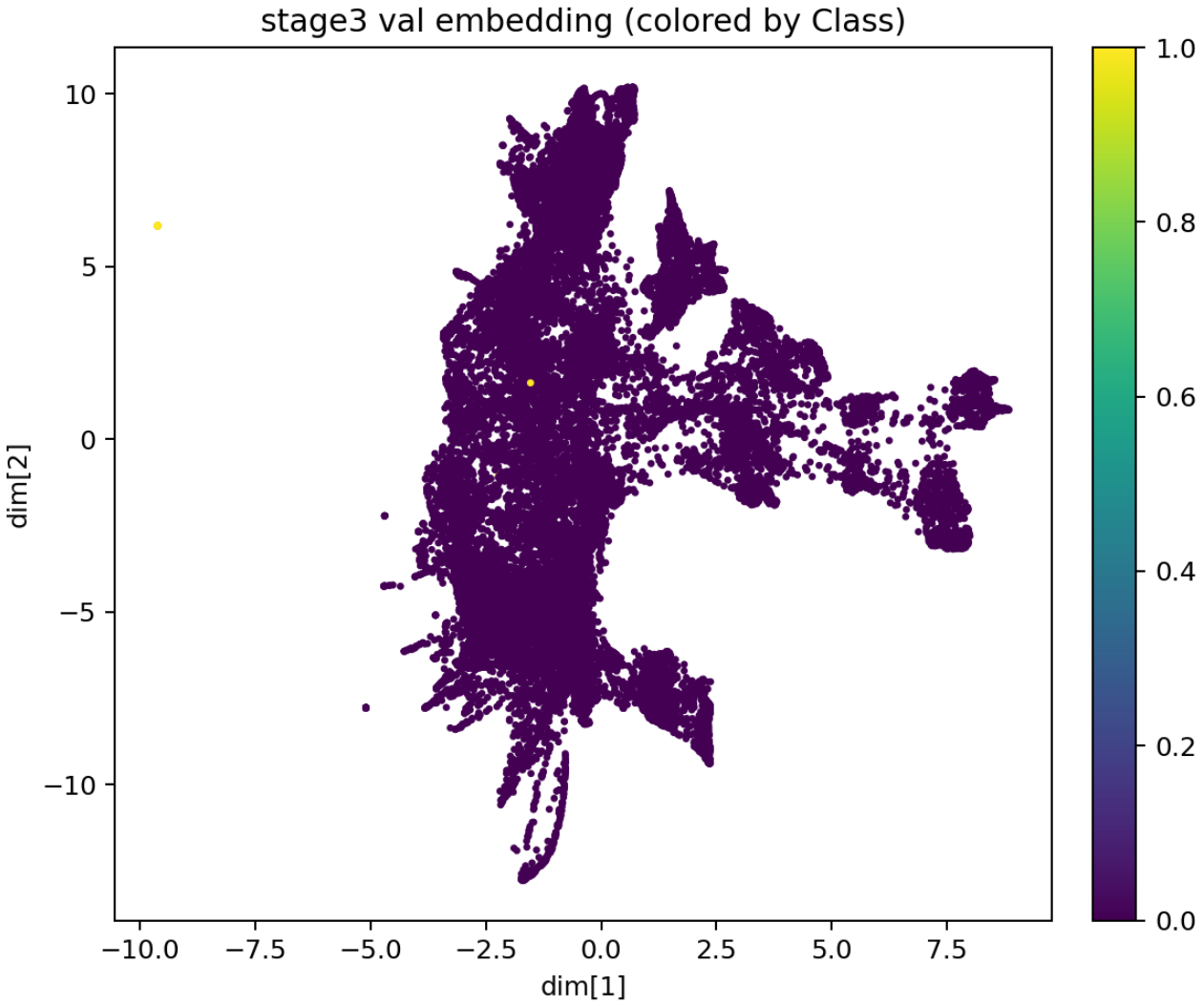
파일: fig/dhk_add_list_boxplots.png

해석/검증 포인트

- 특정 피처(예: V24)가 사기에서 꼬리 분포가 다르면 후보 순위가 더 안정화될 수 있다.

7.3 (검증/시각화) Stage3 후보/점수/성능: Stage1과 무엇이 달라졌나

그림 17. Stage3 val 임베딩 2D(색=Class)

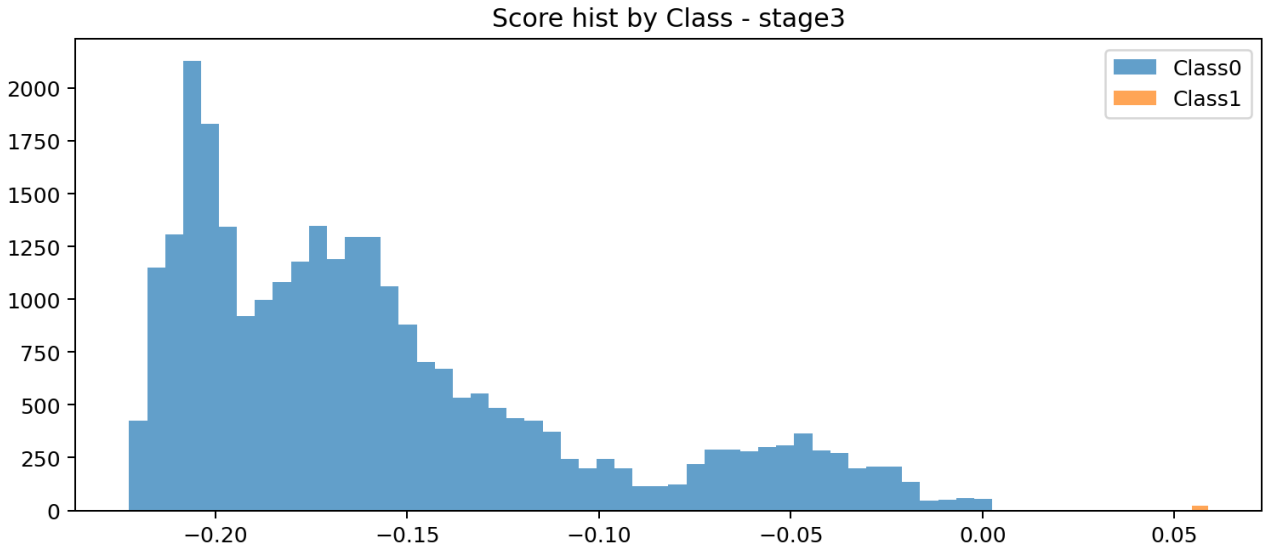


파일: fig/stage3_val_scatter2d_byClass.png

해석/검증 포인트

- Stage1과 비교해 outlier 영역이 더 분명해지거나, 후보가 더 안정적으로 군집 밖에 위치하면 개선 신호다.

그림 18. Stage3 점수 분포(Class별)

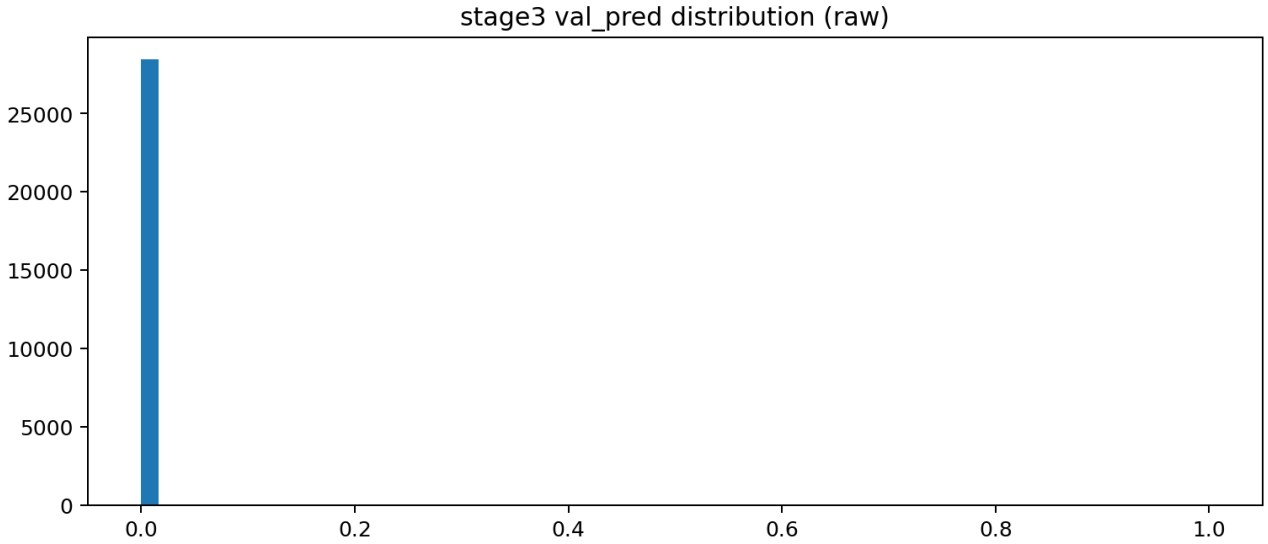


파일: fig/stage3_score_hist_by_class.png

해석/검증 포인트

- Stage1 대비 Class 분포가 더 벌어지면(사기 점수↑), 후보 생성 순위가 더 좋아질 수 있다.

그림 19. Stage3 val 예측값 분포(raw)

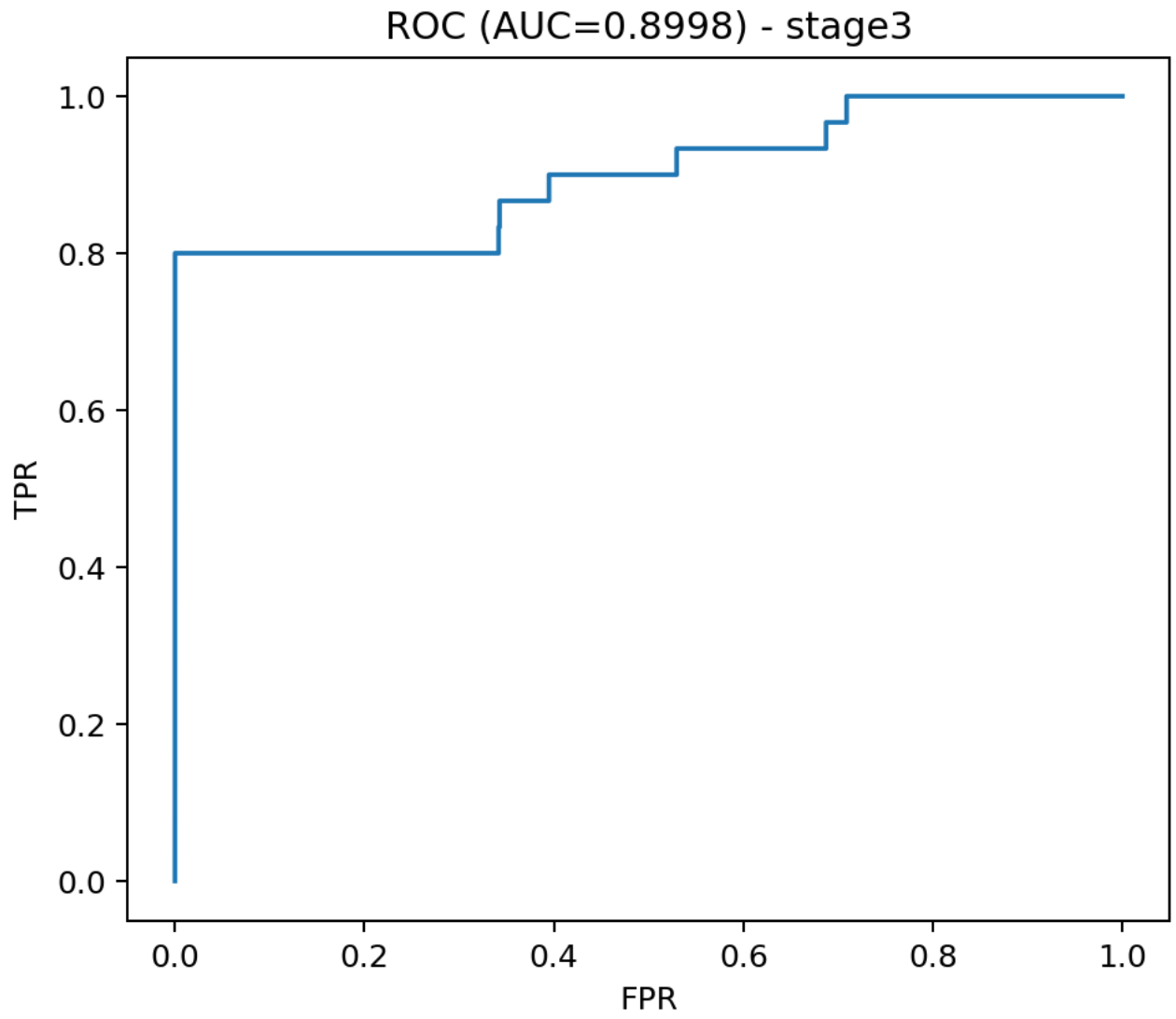


파일: fig/stage3_val_pred_hist.png

해석/검증 포인트

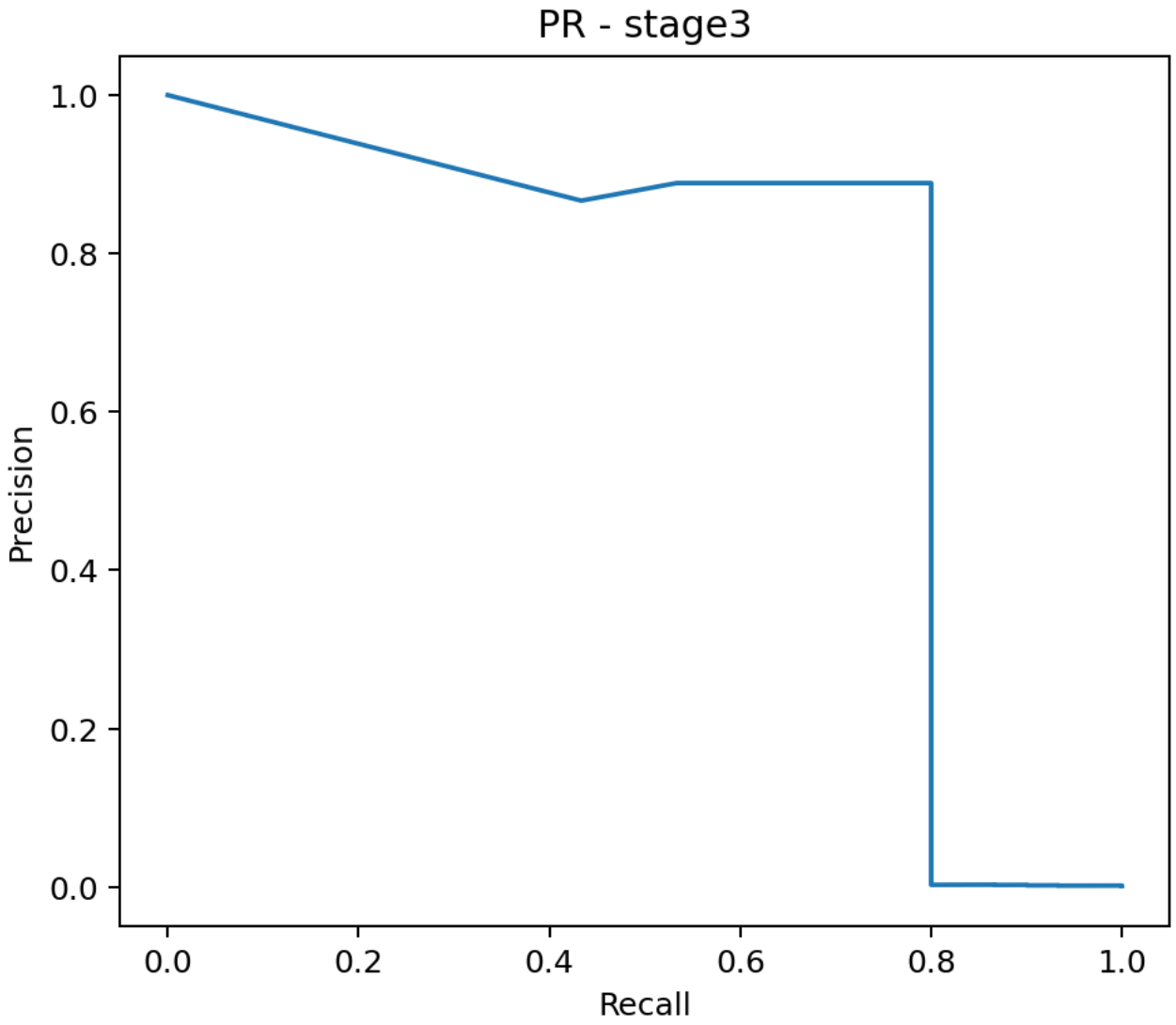
- 후보 수가 비슷하게 유지되면서(희소), 사기 포함률이 유지/향상되면 안정화에 성공한 것이다.

그림 20. Stage3 ROC(AUC)



파일: fig/stage3_roc.png

그림 21. Stage3 PR



파일: fig/stage3_pr.png

표 15. Stage3 혼동행렬(전체 val)

true	pred0	pred1
true0	28429	3
true1	6	24

해석/검증 포인트

- TP=24, FP=3, FN=6로 Stage1과 동일. 이는 두 가지 의미를 갖는다.
- (1) Stage1 결과가 이미 안정적이어서 Stage3 보팅이 바꾸지 않아도 됨, 또는
- (2) Stage3가 Stage1 후보를 그대로 '안정적으로 재현'했음을 의미(랜덤성 문제 없음).
- private 점수 관점에서는 (2)가 특히 중요: 재현성 있는 후보 구조는 과적합보다 더 안전하다.

8. 블록 6: 1010 후보 정제(KernelPCA 기반 FP 제거)

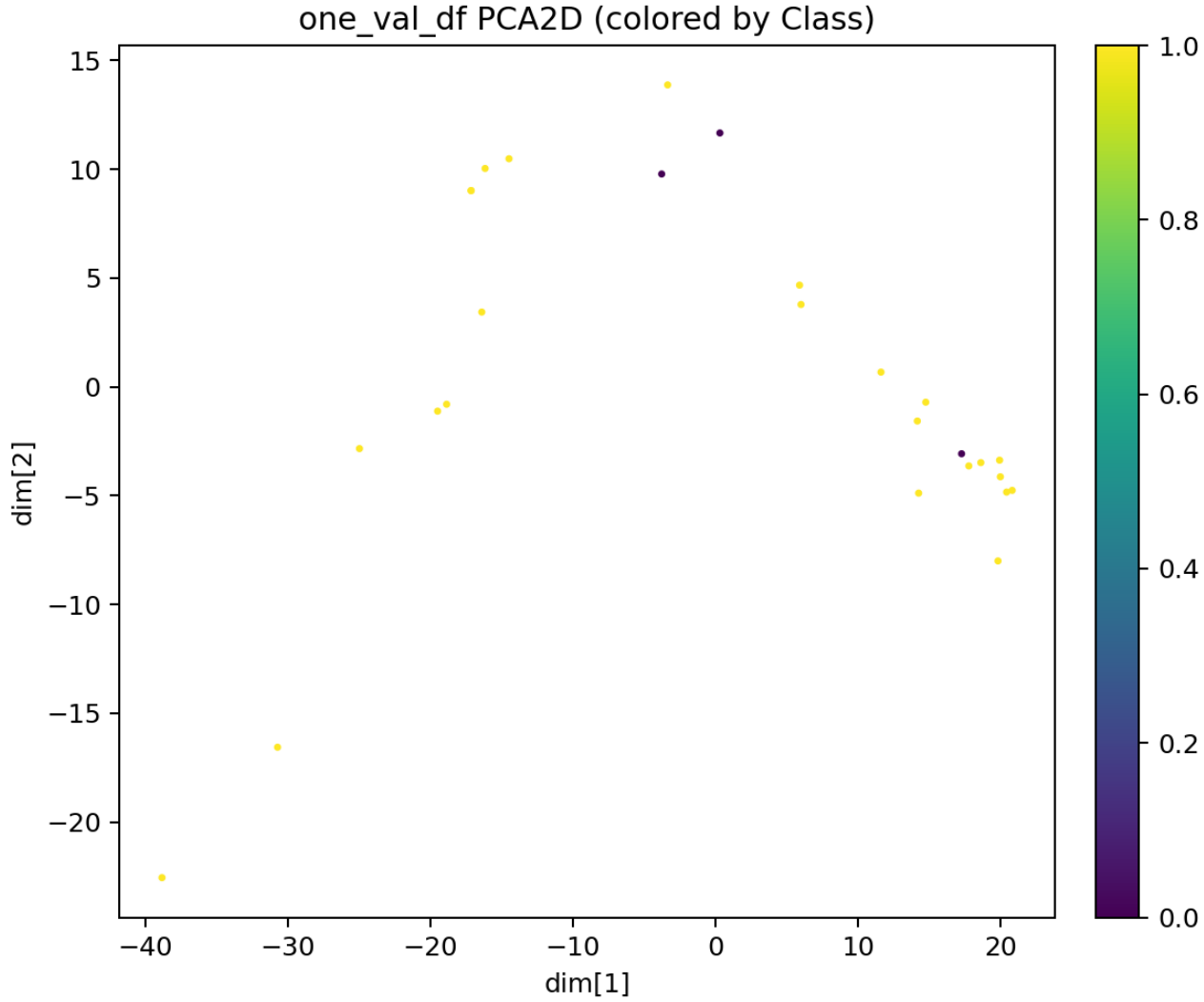
8.1 (처리/학습) 왜 후보 내부에서 다시 비선형 변환을 하나

Stage1/3 후보(27개)는 사기 비율이 매우 높은 고순도 집합이지만, FP=3이 남아 있다. 이 3개를 전체 공간에서 다시 분류하려 하면, 극불균형과 쉬프트로 인해 오히려 FP/FN이 흔들릴 수 있다. 그래서 1등 코드는 '후보 subset'이라는 작은 공간에서 구조를 다시 본다.

KernelPCA 직관: 선형 PCA는 직선으로만 공간을 펼치지만, KernelPCA는 커널을 통해 비선형 관계(곡선/고리/복잡한 경계)를 펼칠 수 있다. 후보 내부에서 정상 FP가 사기와 다른 비선형 구조를 갖는다면, KernelPCA 좌표에서 더 깔끔하게 분리될 수 있다.

8.2 (검증/시각화) 후보 subset 구조 및 pred1010 분포

그림 22. 후보 subset(one_val_df) 2D 좌표(비선형 구조 확인)



파일: fig/one_val_pca2d.png

해석/검증 포인트

- 후보 내부에서 몇 개 점이 분리된 군집/꼬리로 보이면, 그 점들이 FP일 가능성을 의심할 수 있다.
- 단, 이것만으로 FP라고 단정하면 안 되고, pred1010 분포 및 혼동행렬로 확인해야 한다.

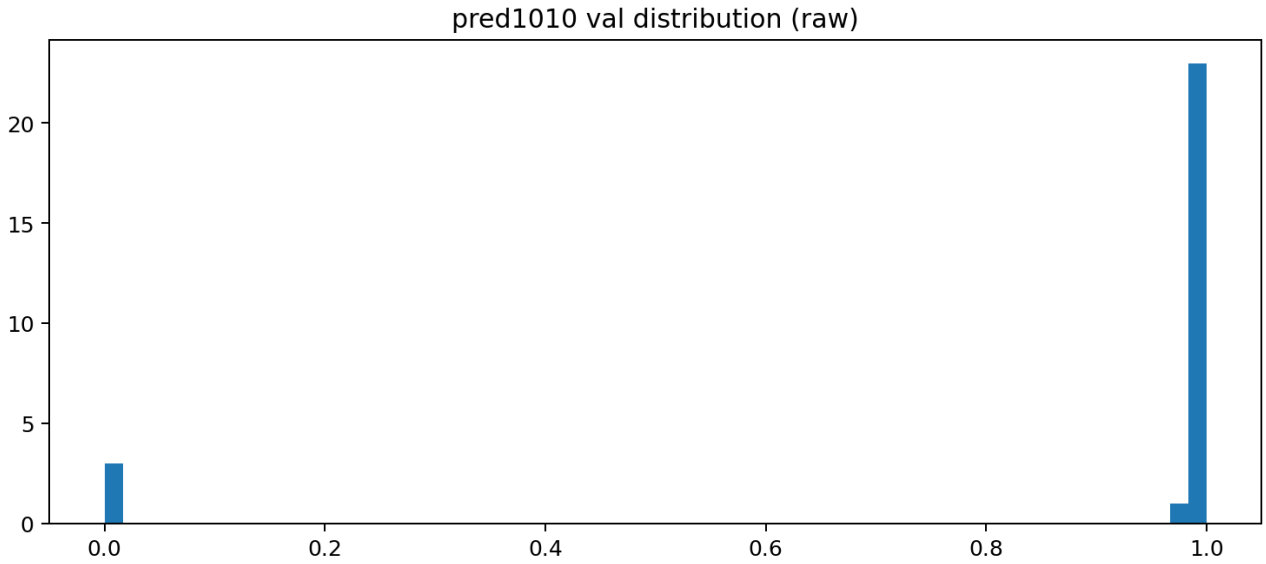
표 16. 후보 subset 내 Class 분포

count	count.1
0	3
1	24

해석/검증 포인트

- Stage1/3 후보의 '고순도'를 수치로 확인한다(사기 24개, 정상 3개).
- 후보 생성이 제대로 됐기 때문에 정제 단계가 '작은 문제'로 축소된다.

그림 23. pred1010 분포(val subset)



파일: fig/pred1010_val_hist.png

해석/검증 포인트

- 값이 0/1 근처로 몰리면 라운딩 기반 규칙이 안정적으로 동작할 수 있다.
- 값이 0.4~0.6에 많이 몰리면 임계값 설정에 민감해져 오히려 불안정해질 수 있다.

표 17. 1010 subset 혼동행렬(후보 27개 내부)

true	pred0	pred1
true0	3	0
true1	0	24

해석/검증 포인트

- FP(정상 3개)를 0으로 모두 제거하면서 TP(사기 24개)를 유지 → 정제 단계의 목적(Precision 향상) 달성.
- 이 결과가 private score에 강하게 기여: 잘못 찍힌 정상(FP)을 제거하면 제출에서 불필요한 양성 예측이 줄어 든다.

표 18. 전체 val 확장 평가(1010 후처리까지 적용)

true	pred0	pred1
true0	28432	0
true1	6	24

해석/검증 포인트

- FP가 3 → 0로 감소하고, TP=24 유지. 즉, Recall 손실 없이 Precision만 개선.
- 이 패턴은 '후보 생성→정제' 설계가 의도대로 작동했음을 보여주는 가장 강한 증거다.

표 19. 1010 단계 요약 지표

item	value
subset_len	27
val_all_len	28462
subset_f1_macro	1
pipeline_f1_macro	0.944392
pipeline_pos_rate	0.00084323

해석/검증 포인트

- subset_f1_macro가 1에 가까우면(또는 매우 높으면) 후보 내부에서 완전 분리가 이루어진 것이다.
- pipeline_pos_rate는 전체 val에서 '양성으로 찍힌 비율'로, 지나치게 커지면 과민(FP), 너무 작으면 보수(FN) 신호다.

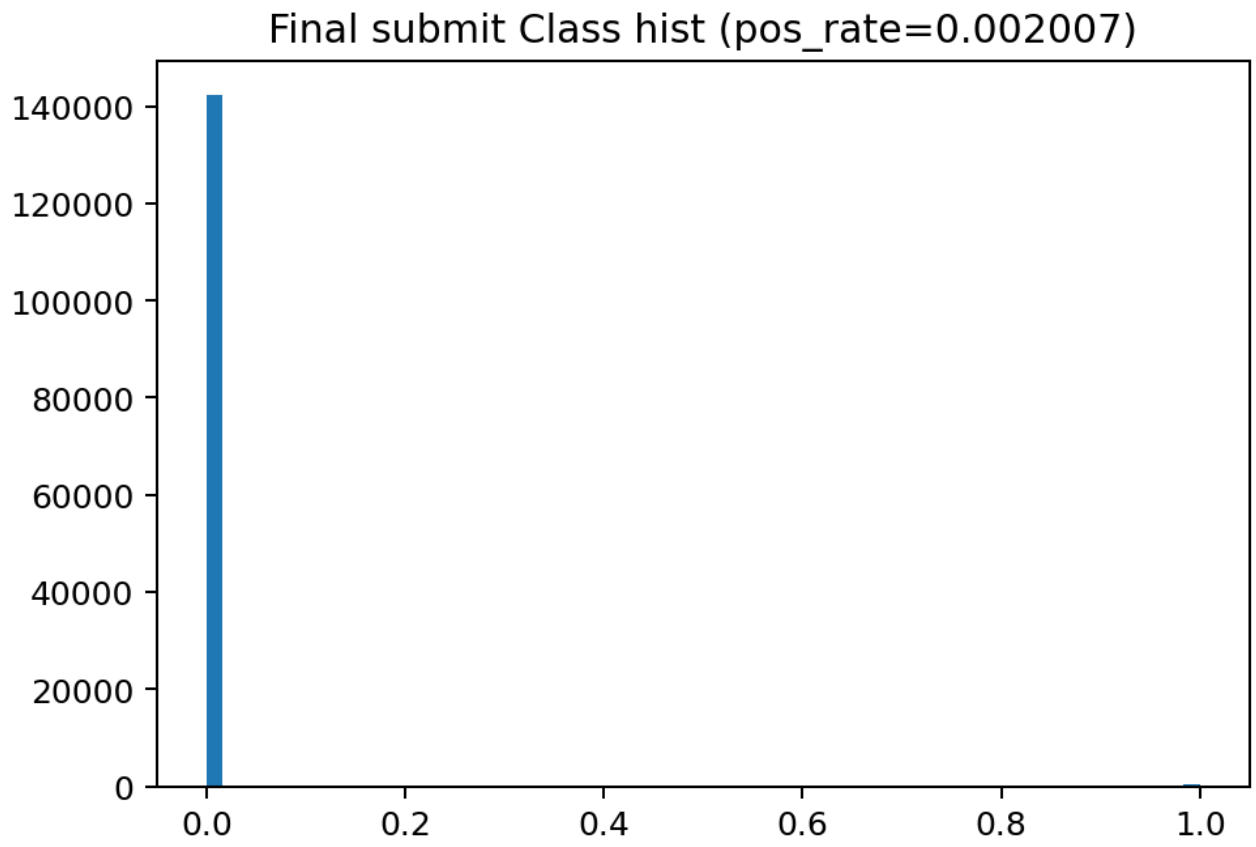
9. 블록 7: 최종 제출 생성(인덱스/ID 매핑의 안전성)

9.1 (처리/학습) 제출 생성에서 가장 흔한 실패 모드

- 인덱스 정렬 깨짐: 점수/예측 배열이 ID와 다른 순서로 섞이면 완전히 잘못된 제출이 된다.
- 후보 subset 확장 오류: subset 결과를 전체에 매핑할 때 길이/인덱스가 불일치하면 오류 또는 silent bug 발생.
- dtype/형식 오류: Class가 float으로 남거나, 0/1이 아닌 값이 들어가면 제출이 비정상.

9.2 (검증/시각화) 최종 Class 분포

그림 24. 최종 제출 Class 분포



파일: fig/final_submit_class_hist.png

해석/검증 포인트
- 양성 비율이 0이면 탐지기가 과도하게 보수적이거나 임계값이 잘못됐을 가능성이 높다.
- 양성 비율이 지나치게 크면 쉬프트/전처리/임계값 오류로 FP 폭증 가능성이 높다.

표 20. 최종 제출 요약

item	value
test_n	142503
final_pos_rate	0.00200698
estimated_positive_count	286

해석/검증 포인트
- 이번 실행에서 양성 약 286개. val fraud rate(0.001054)보다 높지만,

- 후보 생성/정제 구조 특성상 test에서 약간 더 보수/공격적으로 나올 수 있다.
- 핵심은 '극단적으로 튀지 않고' 후보/후처리 로직으로 설명 가능한 범위인지 여부다.

10. 왜 이 방식이 1등(private)에서 강한가

이 파이프라인의 강점은 '라벨을 학습에 쓰지 않으면서도' 성능을 끌어올리는 구조적 설계에 있다.

- 라벨 사용 최소화: val 라벨은 모델 피팅이 아니라 '피처 선택/검증'에만 사용 → 규칙 준수 + 과적합 완화
- 희소 사건 분해: 전체에서 바로 분류하지 않고, 고순도 후보를 만든 뒤 후보 내부 정제로 FP를 제거
- 랜덤성 대응: PaCMAP 변동성은 Stage3 보팅으로 안정화(재현성 강화)
- 정량 검증 루프: 저장/재로딩 동일성, 쉬프트 지표, 혼동행렬로 매 단계 '정상 동작'을 증명

이번 실행에서 Stage1/3는 TP=24(사기 30개 중 24개)까지 끌어올렸고, 1010은 FP=3을 0으로 만들면서 TP를 유지했다. 즉, Recall을 확보한 다음 Precision만 올리는 이상적인 분업이 구현되어 있다.

10.1 남은 FN=6(미탐 사기)의 의미와 원인 가설

혼동행렬에서 FN=6은 '사기인데 정상 군집 내부에 섞여 들어간 샘플'일 가능성이 높다. 비지도 탐지 관점에서 FN은 흔히 다음 이유로 발생한다.

- Feature non-separability: 선택된 피처에서 해당 사기는 정상과 분포가 거의 동일
- Embedding collapse: 임베딩 과정에서 사기 샘플이 정상 군집에 압축되어 outlier로 드러나지 않음
- Contamination/threshold trade-off: 후보 수를 늘리면 FN이 줄지만 FP가 늘어 private score가 악화될 수 있어, 의도적으로 FN을 감수했을 가능성
- Subset-refinement side effect: 1010은 후보 내부 정제만 하므로 FN을 되살리지는 못함(설계 의도상).

따라서 FN=6이 남았다는 사실은 '학습이 실패했다'가 아니라, private 점수를 위해 선택된 Precision-우선 절충의 결과로 해석할 수 있다.

부록 A. 후보 subset(27개) 기술통계(Describe)

표 21. one_val_df(후보 subset) describe

0	count	mean	std	min	25%	50%	75%	
	27	-7.4402	7.93101	-26.4577	-13.3654	-3.89658	-1.44576	1.3
	27	5.69778	3.98264	1.28938	2.68608	4.14199	7.50879	16
	27	-11.3521	8.51972	-30.1773	-20.3135	-7.77054	-4.9952	-0.
	27	6.33087	2.92278	1.41185	5.46081	6.04745	7.62471	12
	27	-5.72983	6.12861	-17.8926	-10.914	-4.01178	0.159404	1.5
	27	-2.03973	1.55642	-5.141	-2.92784	-1.96844	-1.1861	0.8
	27	-9.36847	8.53081	-31.1973	-14.4666	-7.73993	-2.16168	-0.
	27	1.68824	5.14693	-11.9196	0.303476	1.43101	2.88986	12
	27	-3.83837	2.37965	-9.46257	-4.73093	-3.24952	-2.16561	-0.
	27	-8.95157	5.97855	-22.1871	-13.8396	-6.60046	-3.72658	-0.
	27	5.47746	2.86192	0.301369	3.36274	5.14835	7.61082	10
	27	-9.22654	4.66597	-16.0603	-13.8018	-9.00191	-5.17537	-1.
	27	0.231471	1.05344	-1.95679	-0.530209	0.386352	0.757968	2.7
	27	-9.08409	3.99964	-14.953	-13.2479	-9.05799	-6.19543	-1.
	27	-0.4947	1.06238	-2.99243	-1.05018	-0.536801	0.194798	1.2
	27	-5.91597	4.14971	-12.6753	-9.51953	-6.80989	-1.61152	-0.
	27	-9.74981	6.91624	-20.7407	-15.7658	-11.8038	-2.99723	1.5
	27	-3.32865	3.0332	-8.15367	-5.62485	-4.7413	-0.662471	1.8
	27	1.04827	1.65562	-1.80801	-0.0380684	0.832574	1.77814	5.2
	27	0.410102	1.22855	-1.85047	-0.204668	0.353898	0.995165	3.8

해석/검증 포인트

- 후보 subset의 평균/표준편차가 원본(val)과 크게 다르면 '정상 군집 바깥'을 잡았다는 정황.
- 특정 피처에서 극단값이 두드러지면, IF 점수나 KernelPCA 분리에 크게 기여했을 수 있다.

표 22. ori_val_df(전체 val) describe

0	count	mean	std	min	25%	50%	75%	
	28462	0.00496679	1.93064	-29.5161	-0.915525	0.0235863	1.31558	2.4
	28462	0.00201413	1.6052	-38.3053	-0.598053	0.0754701	0.803463	16
	28462	0.00141373	1.49975	-30.1773	-0.873022	0.175784	1.01156	4.2
	28462	0.00189253	1.40514	-5.07124	-0.852444	-0.0216193	0.739044	12
	28462	-0.00396881	1.33515	-21.577	-0.69801	-0.05278	0.598712	24
	28462	-0.0177296	1.29221	-16.1726	-0.77492	-0.280742	0.377266	12
	28462	0.00555563	1.16513	-31.1973	-0.54629	0.0462796	0.566825	26

0	count	mean	std	min	25%	50%	75%	
	28462	0.00902301	1.10354	-26.278	-0.210941	0.0226217	0.323836	12
	28462	-0.00490461	1.09084	-9.46257	-0.64276	-0.0666225	0.596308	7.9
	28462	-0.0022989	1.08456	-22.1871	-0.540463	-0.0993862	0.452763	12
	28462	0.00400678	1.01651	-4.45385	-0.755867	-0.0335268	0.743699	10
	28462	0.00248036	0.990731	-16.0603	-0.408285	0.13737	0.618721	4.2
	28462	0.00417031	1.00056	-3.84894	-0.638741	-0.0123175	0.666392	4.4
	28462	0.0122989	0.938752	-14.953	-0.411374	0.0596796	0.50019	7.6
	28462	0.00317739	0.913747	-4.39131	-0.572378	0.0536062	0.64831	3.8
	28462	0.00491257	0.864289	-12.6753	-0.461673	0.0709308	0.521601	4.4
	28462	0.00755218	0.821528	-20.7407	-0.474047	-0.066174	0.399337	6.9
	28462	0.00573275	0.83328	-8.15367	-0.491693	0.00089036	0.509625	3.5
	28462	0.000146352	0.808897	-4.19715	-0.463059	0.000396637	0.462772	5.5
	28462	0.00192798	0.729583	-18.2923	-0.212876	-0.0624809	0.131592	24

부록 B. ipynb 구조(헤딩/단계 표시)

표 23. 노트북 셀 아웃라인(상위 40개)

cell	lvl	title
3	3	(code) PaCMAP/IsolationForest 단계
4	3	(code) PaCMAP/IsolationForest 단계
4	3	(code) KernelPCA(1010) 단계
4	3	(code) 통계 기반 피처 선택
5	3	(code) PaCMAP/IsolationForest 단계
9	3	(code) 통계 기반 피처 선택
14	3	(code) PaCMAP/IsolationForest 단계
15	3	(code) PaCMAP/IsolationForest 단계
16	3	(code) PaCMAP/IsolationForest 단계
17	3	(code) PaCMAP/IsolationForest 단계
19	3	(code) PaCMAP/IsolationForest 단계
19	3	(code) 통계 기반 피처 선택
21	3	(code) PaCMAP/IsolationForest 단계
22	3	(code) PaCMAP/IsolationForest 단계
24	3	(code) PaCMAP/IsolationForest 단계
25	3	(code) PaCMAP/IsolationForest 단계
27	3	(code) 통계 기반 피처 선택
30	3	(code) PaCMAP/IsolationForest 단계
30	3	(code) KernelPCA(1010) 단계
30	3	(code) 통계 기반 피처 선택

해석/검증 포인트

- 본 보고서의 블록 구분은 노트북의 '단계별 계산→검증' 흐름을 기반으로 재정리한 것이다.
- 특히 PaCMAP/IF, KernelPCA(1010)가 각각 독립 블록으로 반복 등장하는 것이 핵심 패턴이다.