

유전적 친족 관계 분류기 (Genomic Relatedness Classifier)

IBD 지표와 분포 통계(feature)로 친족 관계를 예측하는 모델을 구축하고 평가합니다. 데이터셋(cM_1, cM_3, cM_6)별로 하나의 통합 리포트를 생성합니다.

프로젝트 구조

- `data/raw/` — 입력 원본 데이터
 - `model_input_with_kinship_filtered_cM_*.csv` — cM 임계값(1, 3, 6)별 라벨 데이터
 - `merged_info.out` (및 `merged_info.out.zip`) — 분포 통계 원본
- `data/processed/` — 전처리/중간 산출물
 - `merged_cM_*.csv` — 모델 입력용 병합 데이터
 - `top_features_*.pkl`, `scaler_*.pkl` — 특성 선택/스케일러
 - `evaluation_results_*.json` — 모드별 평가 JSON (통합 후에는 참조용)
- `models/<dataset>/<mode>/` — 학습된 모델 가중치(`mlp.pth`, `cnn.pth`) — git 무시
- `reports/<dataset>/` — 통합 리포트와 플롯 — git 무시
 - `results.json`, `results.md`, `results.pdf`
 - `results_KR.md`, `results_KR.pdf`
 - `assets/<scenario>/feature_importance_<dataset>_<scenario>.svg|png`
 - `assets/<scenario>/kinship_distribution_<dataset>_<scenario>.svg|png`
 - `plots/confusion/<scenario>/<mode>/confusion_matrix_*.svg|png`
- `scripts/` — 파이프라인 스크립트
 - `run_all.py` — 전체 파이프라인 오케스트레이션
 - `data_prep.py`, `eda.py`, `feature_selection.py` — 전처리/EDA/특성선택
 - `train_models.py`, `evaluate_models.py`, `build_report.py` — 학습/평가/리포트
- `docs/` — 문서 (`plan.md`, `plan_KR.md`)

준비 (Windows / PowerShell)

필수 조건:

- Python 3.11 (레포 .venv 기준)
- NVIDIA GPU 및 호환 CUDA 드라이버

단계:

1. uv 설치

```
pip install uv
```

2. 레포 루트에 가상환경 생성/활성화

```
uv venv  
.venv\Scripts\activate
```

3. pyproject/uv.lock 기반 의존성 설치 (재현 가능)

```
uv lock ; uv sync
```

4. PyTorch(CUDA) 별도 설치 (CPU 훨 방지 목적)

CUDA 12.1 예시:

```
uv pip install --index-url https://download.pytorch.org/whl/cu121 `  
torch==2.5.1+cu121 torchvision==0.20.1+cu121 torchaudio==2.5.1+cu121
```

5. (선택) PDF 출력용 Node.js 설치 (npx md-to-pdf 활성화)

사용법

데이터셋 단일 실행:

```
.\.venv\Scripts\python.exe scripts\run_all.py cM_1 --epochs 1 --train-device cuda --eval-device cuda
```

모든 데이터셋 실행:

```
.\.venv\Scripts\python.exe scripts\run_all.py all --epochs 1 --train-device cuda --eval-device cuda
```

옵션:

- `--epochs <int>` : 에포크 수 (기본값 1 또는 `$env:TRAIN_EPOCHS`)
- `--special-epochs <int>` : UN 포함 + 과샘플링(smote/overunder)인 경우에만 별도 에포크 적용
- `--train-device cuda` : 학습 디바이스 (정책상 CPU 비허용)
- `--eval-device cuda` : 평가 디바이스 (정책상 CPU 비허용)

출력물

`reports/<dataset>/`에 통합 리포트를 생성합니다.

- `results.md` / `results_KR.md` (+ 선택적 PDF)
- 시나리오 플롯: `assets/<scenario>/feature_importance_*.svg|png`, `kinship_distribution_*.svg|png`
- 혼동 행렬: `plots/confusion/<scenario>/<mode>/confusion_matrix_*.svg|png`

시나리오와 모드

- 시나리오:
 - `included` : UN 라벨을 포함하여 영향 평가
 - `noUN` : 학습/검증 전 UN 삭제
- 모드(불균형 대응):
 - `zero` : 재균형 없음(기준선, 다수 편향 가능)
 - `weighted` : 클래스 가중 손실
 - `smote` : 학습 세트 SMOTE 과샘플링

- o overunder : SMOTE 이후 ENN/Tomek 정리(과샘플 + 언더샘플)

리포트는 시나리오 플롯을 2열로 배치하며, 텍스트 선명도를 위해 SVG를 우선 사용합니다. 모드/모델별 혼동 행렬도 포함됩니다.

모델 가중치는 `models/<dataset>/<mode>/{mlp.pth, cnn.pth}`에 저장됩니다.

참고 (클래스 불균형)

- 'zero' 모드는 재균형 미적용 기준선으로, 다수 클래스에 편향될 수 있습니다. 매크로/가중치 지표와 클래스별 결과를 함께 보세요.
- 'weighted'는 클래스 가중 손실, 'smote'는 학습 세트 과샘플링(검증은 원본 분포)입니다.
- AUC는 OvR 방식으로 강건하게 계산되며, 정의 불가 상황에서는 종립값(0.5)로 대체되어 N/A가 발생하지 않습니다.

재현성 (pyproject.toml + uv.lock)

- 레포에는 정확한 파이썬 버전 범위와 핵심 의존성이 정의된 `pyproject.toml` 과, `uv lock` 으로 생성한 `uv.lock` 이 포함됩니다.
- PyTorch GPU 훈련은 CPU 훈련 설치를 피하기 위해 `pyproject.toml` 에 명시하지 않았습니다. CUDA 전용 인덱스로 설치하세요(준비 단계 4 참고).
- `.venv/` 는 git 무시 대상입니다.

Git 트래킹

- `models/`, `reports/`, `data/processed/*.csv` 등 재생성 가능한/대용량 산출물은 git에서 무시됩니다.
- 필요 시 LFS를 설정해 산출물을 트래킹할 수 있습니다.