GENOME: 유전체 및 역학 데이터 기반 질병 예측 (MILTON 활용)

이 프로젝트는 MILTON 프레임워크를 사용하여 UK Biobank 형식의 유전체 및 역학 데이터를 기반으로 질병 예측 모델을 개발하고 평가하는 것을 목표로 합니다.

1. 설치

MILTON 라이브러리 및 필요한 의존성 설치를 위해 다음 단계를 따릅니다.

```
# 1. 프로젝트 클론
git clone https://github.com/iKnowLab-Projects/GENOME.git
cd GENOME

# 2. MILTON 코드 디렉토리로 이동
cd code/milton-release

# 3. (권장) Conda 환경 생성 및 활성화 (예: milton_env 이름 사용)
# conda create -n milton_env python=3.9 # 필요시 파이썬 버전 지정
# conda activate milton_env

# 4. 의존성 설치 및 MILTON 설치 (편집 가능 모드)
pip install -r requirements.txt
pip install -e .

# 5. 프로젝트 루트 디렉토리로 복귀
cd ../../
```

2. 데이터 준비

분석을 실행하기 전에 필요한 데이터를 올바른 형식과 위치에 준비해야 합니다.

2.1. 데이터 구조

다음과 같은 디렉토리 구조로 데이터를 구성하는 것을 권장합니다. (경로는 예시이며, 실제 경로에 맞게 조정해야 합니다.)

```
DATA_FOLDER/
├── UKB_genome/ # 유전체 데이터 디렉토리
│ ├── UKB_genome.bed # PLINK BED 파일
│ ├── UKB_genome.bim # PLINK BIM 파일
│ └── UKB_genome.fam # PLINK FAM 파일
├── total_demo.txt # 역학 데이터 파일 (.txt 또는 .parquet)
└── (ukb.parquet/) # 스크립트 실행 시 생성될 전처리된 역학 데이터 디렉토리
```

- **유전체 데이터 (`UKB_genome/`)**: PLINK 형식의 `.bed`, `.bim`, `.fam` 파일이 포함된 디렉토리입니다. 모든 파일의 기본 이름(예: `UKB_genome`)은 동일해야 합니다.
- **역학 데이터 (`total_demo.txt`)**: 대상자들의 인구통계학적 정보, 임상 측정치 등이 포함된 텍스트 파일 (`.txt`) 또는 Parquet 파일 (`.parquet`)입니다.
    - **필수 컬럼**: `eid` (Subject ID) 컬럼이 반드시 포함되어야 합니다.
    - **컬럼 형식**: MILTON은 UK Biobank 데이터 필드 ID 형식을 기대합니다 (예: `31.0.0`은 성별). 만약 데이터가 다른 형식의 컬럼명을 사용한다면 `process_ukb_data.py` 스크립트가 특정 변환을 시도하지만, 예상대로 동작하지 않을 수 있습니다. 가급적 UKB 형식(숫자ID.인스턴스.배열인덱스)과 유사하게 맞춰주는 것이 좋습니다.
    - **구분자**: `.txt` 파일의 경우 탭(`\t`) 또는 쉼표(`,`)를 구분자로 사용해야 합니다. 스크립트가 자동으로 감지합니다.

2.2. 스크립트 내 경로 설정

`process_ukb_data.py` 스크립트 상단에서 데이터 및 결과 경로를 **절대 경로**로 올바르게 수정해야 합니다.

```
# GENOME/process_ukb_data.py 상단 경로 수정

# .bed, .fam, .bim 파일이 있는 *디렉토리* 경로
GENOME_DATA_PATH = "/home/soyeon/workspace/data/GENOME/UKB_genome"
# 역학 데이터 파일 경로 (.txt 또는 .parquet)
DEMO_DATA_PATH = "/home/soyeon/workspace/data/GENOME/total_demo.txt"
# 최종 분석 결과가 저장될 기본 디렉토리 경로
OUTPUT_BASE_PATH = "./results" # 현재 디렉토리 아래 results 폴더에 저장
```

중요: `GENOME_DATA_PATH`는 유전체 파일들이 위치한 **디렉토리 경로**를 지정해야 합니다. 스크립트는 이 경로를 `UKB_DATA_LOCATION` 환경 변수로 설정하여 MILTON이 유전체 데이터를 찾도록 합니다.

3. 분석 실행

모든 설정이 완료되면 프로젝트 루트 디렉토리 (`GENOME/`)에서 다음 명령어를 실행하여 분석을 시작합니다.

```
python process_ukb_data.py
```

실행 과정

스크립트는 다음 단계들을 자동으로 수행합니다.

1. **유전체 데이터 확인**: `GENOME_DATA_PATH`에서 `.bed`, `.bim`, `.fam` 파일 존재 여부를 확인합니다.
2. **역학 데이터 로드 및 전처리**:
    - `DEMO_DATA_PATH`에서 역학 데이터를 로드합니다.
    - 파일 크기에 따라 전체 데이터를 로드할지, 일부만 로드할지 사용자에게 질문합니다.
    - 컬럼명 형식을 MILTON이 인식 가능한 형태로 변환하려고 시도합니다 (예: `X31-0.0` -> `X31.0.0`).
    - 데이터 타입을 'category'로 변환합니다.
    - 전처리된 데이터를 **10개의 Parquet 파티션 파일** (`part-0.parquet` ~ `part-9.parquet`)로 나누어 원본 역학 데이터와 **같은 디렉토리 내의 `ukb.parquet/` 폴더**에 저장합니다. MILTON은 이 Parquet 파

일들을 사용합니다.
    - UKB 샘플 목록 (`ukb-sample-list.txt`)을 생성합니다.
3. **MILTON 세션 시작**: Dask 로컬 클러스터를 설정합니다.
4. **분석 설정 생성 (`make_v16_config`)**:
    - `process_ukb_data.py` 내에 하드코딩된 값들 (컨트롤 비율, 시간 모델, 특성 집합 등)을 사용하여 MILTON 분석 설정을 구성합니다. 필요시 이 부분을 직접 수정하여 분석 파라미터를 변경할 수 있습니다.
    - 분석할 질병 코드 (`code = 'E11'`)를 설정합니다.
    - 결과 저장 경로 (`out_dir`)를 구성합니다.
5. **MILTON 분석 실행 (`run_milton_analysis`)**:
    - 설정된 질병 코드 (`code`)에 대한 케이스/컨트롤 정의 (`patients.spec`).
    - `Evaluator` 객체 생성 및 `run()` 메서드 호출.
        - 데이터 로딩 (`load_data`).
        - 필요시 특성 선택 (`_select_features`).
        - 모델 훈련 및 평가 (`_fit_all_models`, `_evaluate`).
    - 결과 저장 (`save_report`).

4. 결과 확인

분석이 성공적으로 완료되면 `OUTPUT_BASE_PATH` 아래에 지정된 하위 디렉토리 (예: `./results/E11/67bm/time_agnostic/`)에 다음과 같은 결과 파일들이 생성됩니다.

- `report.html`: 분석 결과 요약 및 시각화 (Bokeh 차트 포함).
- `metrics.csv`: 모델 성능 지표 (AUC, F1 등).
- `model_coeffs.csv`: 특성 중요도 또는 로지스틱 회귀 계수.
- `scores.parquet`: 각 대상자에 대한 예측 점수 및 코호트 할당 정보.
- `qv_significance.parquet`: (QV 분석 활성화 시) 유전자 기반 희귀 변이 분석 결과.
- `stuff.pickle`: ROC/PR 곡선 데이터 등 추가 정보.

5. 주요 디버깅 기록 및 이슈 사항

분석 파이프라인 구축 과정에서 발생했던 주요 문제들과 해결 과정을 기록합니다.

- **모델 훈련/평가 오류 (Classifier `fit` 실패)**:
    - **증상**: 모델 훈련 (`MiltonPipeline.fit`) 또는 평가(`_eval_replica`) 단계에서 `ValueError('Invalid classes inferred from unique values of y. Expected: [0], got [1]')` 오류 발생. 이는 타겟 변수 `y` (코호트 정보)에 컨트롤 그룹(클래스 `0`) 없이 케이스 그룹(클래스 `1`)만 전달되었기 때문입니다.
    - **원인 추정**: 파이프라인 앞단의 코호트 생성 (`UkbPatientSelector`) 시 컨트롤 그룹 샘플링/추가 (`_add_controls`, `_stratified_sample`) 단계에서 문제가 발생하여 컨트롤 그룹이 누락되는 것으로 파악되었습니다. 특히, 층화 샘플링 요인(`sampling_factors`) 로딩 (`_load_factors`) 실패가 컨트롤 그룹 매칭 실패로 이어질 수 있습니다.
    - **임시 해결**: `classification.py`의 `MiltonPipeline._get_estimator` 함수 내 `resample` 호출 시 `same_size=True`를 `same_size=False`로 변경하여 클래스 불균형을 허용하도록 수정했습니다. 이는 근본적인 컨트롤 누락 문제를 해결하는 것은 아니지만, `resample` 함수 자체의 오류는 방지합니다. (근본 원인 해결을 위한 추가 디버깅 필요)
- **데이터 로딩 시 Empty DataFrame 반환**:

    - **증상**: 데이터 로딩 및 전처리 후 실제 Classifier에 전달되기 전 단계 (`data_source.py`의 특정 지점)에서 DataFrame의 내용은 비어있고 컬럼 정보만 남는 현상 발생. Dask 객체의 경우 `.compute()` 실행 시 빈 DataFrame 반환.
    - **원인**: `ParquetDataSet` 초기화 시 `opt_outs` (분석 제외 대상자) 목록을 읽어오는 과정에서 발생. `data_info.py`의 `UKB_OPT_OUT_SUBJECTS` 설정 파일 (`ukb-opt-outs.csv`)에 실수로 **모든 대상자 ID**를 넣어, `ParquetDataSet._load_cached()` 내부의 `read_parquet` 호출 시 `excl_ids` 인자에 의해 모든 데이터가 필터링되었습니다.
    - **해결**: `data_info.py`에서 `UKB_OPT_OUT_SUBJECTS = None`으로 설정하여 해당 제외 로직을 비활성화했습니다.

- **컬럼명 형식 불일치 문제**:
    - **문제**: 원본 Parquet 파일의 컬럼명 (예: `X31.0.0`)과 MILTON 내부 로직(특히 데이터 사전 기반 처리)에서 기대하는 처리된 필드 ID (예: `31`) 형식이 혼용되어 데이터 접근 및 처리에 오류 발생.
    - **해결**: 데이터 로딩 시점에는 원본 컬럼명을 유지하고 (`data_source.py` 수정), 각 함수에서 필요에 따라 필드 ID를 기반으로 해당 원본 컬럼을 찾아 사용하도록 로직을 수정했습니다 (`patsel.py`, `utils.py` 등). 예를 들어, `patsel.py`의 `_load_factors`는 필드 ID로 원본 컬럼(`X[ID].0.0` 등)을 찾아 처리하고, 최종 결과는 원본 컬럼명으로 반환하도록 수정했습니다. (`utils.py`의 `process_column_names`는 최종적으로 컬럼명을 변경해야 할 때 사용될 수 있습니다.)

- **Categorical 데이터 타입 강제**:
    - **문제**: 일부 데이터 처리 또는 모델링 단계에서 명시적으로 Categorical 데이터 타입을 요구하는 경우 발생.
    - **해결**: `utils.py`에 `convert_df_dtypes_to_categorical` 함수를 추가하고, `process_ukb_data.py` 등 필요한 지점에서 호출하여 DataFrame 전체 또는 특정 컬럼의 타입을 'category'로 일괄 변환했습니다.

(선택 사항) 데이터 탐색

`eda.ipynb` 노트북 파일을 사용하여 로드된 역학 데이터를 탐색해 볼 수 있습니다. (노트북 사용법은 해당 파일 내 설명 참조)