🫀 Heart Disease Classification Project

📌 Project Overview

Heart disease classifier based on CDC dataset (**319,795 samples**). The project implements a classification pipeline to predict heart disease occurrence based on medical indicators and lifestyle factors.

🎯 Project Objective

Create an efficient classification model predicting heart disease occurrence in population based on medical and demographic data.

📊 Key Performance Indicators (KPI)

- **F1-score (weighted) ≥ 0.75** (metric saved in MLflow)
- **LogLoss** as probabilistic classification quality indicator
- Metric stability in cross-validation (**std ≤ 0.05**)
- Automated hyperparameter tuning using **Optuna**
- Code and model change tracking via **MLflow + Git**

⚠️ Risk Assessment

- **Data imbalance risk:** class imbalance (91.4% "No", 8.6% "Yes") -- requires balancing (class_weight, stratified sampling)
- **Model dependency risk:** use of specific framework (CatBoost)
- **Overfitting:** controlled by early stopping and cross-validation
- **Reproducibility risk:** minimized through version control (joblib, MLflow, git hash, random_state)

📂 Dataset Description

Dataset consists of **319,795 samples and 18 columns** (17 features + 1 label).

Columns:

Column	Description
HeartDisease	Heart disease diagnosis (Yes/No) -- **target variable**
BMI	Body Mass Index
Smoking	Smoking status (Yes/No)
AlcoholDrinking	Alcohol consumption (Yes/No)
Stroke	Previous stroke (Yes/No)
PhysicalHealth	Days of poor physical health (0--30)
MentalHealth	Days of poor mental health (0--30)
DiffWalking	Walking difficulty (Yes/No)
Sex	Gender (Male/Female)
AgeCategory	Age category
Race	Race/ethnicity
Diabetic	Diabetes status (Yes/No/Borderline/Yes during pregnancy)
PhysicalActivity	Physical activity (Yes/No)

GenHealth	General health (Excellent/Very good/Good/Fair/Poor)
SleepTime	Hours of sleep per day
Asthma	Asthma (Yes/No)
KidneyDisease	Kidney disease (Yes/No)
SkinCancer	Skin cancer (Yes/No)

Target Variable Distribution

- No (no heart disease): **292,422 samples (91.4%)**
- Yes (heart disease): **27,373 samples (8.6%)**

⚠️ Warning: Clear class imbalance -- applied class_weights in CatBoost and stratified sampling.

🤖 Model Description

Model: **CatBoostClassifier (gradient boosting)**

CatBoost Features

- Native support for categorical data (no one-hot encoding)
- Ordered boosting -- reduces overfitting risk
- Imbalanced class handling (class_weights)

🔧 Hyperparameter Tuning

- Implementation: **Optuna** + mlflow.start_run(nested=True)
- Best parameters saved in best_params.pkl and logged in MLflow
- Key optimization parameters:
  - iterations
  - learning_rate
  - depth
  - l2_leaf_reg
  - class_weights

🔁 Cross-Validation

- Implementation: catboost.cv with 5-fold stratified shuffle
- Monitored metrics: **F1, LogLoss, AUC-ROC**
- Results visualized with error bands using **Plotly**

🗂️ Project Structure

```
├── .devcontainer/          # Codespaces / Docker configuration
├── .github/workflows/      # CI/CD pipeline
│   ├── ci.yml              # code linting and formatting
│   └── lint-code.yml       # additional PR tests
├── ARISA_DSML/             # main source code
```

```
│   ├── __init__.py
│   ├── config.py          # project configuration
│   ├── preproc.py         # data processing
│   ├── train.py           # model training
│   ├── predict.py         # predictions and monitoring
│   ├── resolve.py         # model management
│   └── helpers.py         # utility functions
├── data/                  # project data
│   ├── raw/               # raw data from Kaggle
│   ├── interim/           # intermediate data
│   ├── processed/         # processed data
│   └── external/          # external data
├── models/                # models and artifacts
├── reports/               # reports and visualizations
│   └── figures/           # charts and diagrams
├── results/               # EDA/DS experiment results
├── mlruns/                # MLflow tracking
├── mlartifacts/           # MLflow artifacts
├── notebooks/             # Jupyter notebooks
│   ├── 01-Before_MLOps.ipynb
│   └── 02-Model_version.ipynb
├── tests/                 # unit tests
├── docs/                  # documentation
├── Makefile               # task automation
├── README.md              # project description
├── pyproject.toml         # package configuration
├── setup.cfg              # tool configuration
├── requirements.txt       # Python dependencies
```

└── .gitignore   # git exclusions

⚙️ Prerequisites

- Python 3.11+
- Pandas & NumPy
- Scikit-learn
- Matplotlib & Plotly
- Jupyter Notebook
- MLflow
- Git & GitHub

🚀 Installation and Setup

Clone repository

git clone https://github.com/Pawel20240101/PZ_ARISA_MLOps_Final.git

cd PZ_ARISA_MLOps_Final

Create environment

python -m venv .venv

Windows

.\.venv\Scripts\activate

Linux/Mac

source .venv/bin/activate

Install dependencies

pip install -r requirements.txt

Input data

Data is automatically downloaded from Kaggle via API in preproc.py

Start MLflow UI

mlflow ui

Or

mlflow ui --host 127.0.0.1 --port 5000

🔄 MLOps Pipeline

Makefile - Local testing and development

```
# Full local pipeline (development and debugging)
make test             # Complete test: linting + ML pipeline

# Individual stages
make preprocess       # Data download and processing
make train            # Model training with hyperopt
make predict          # Predictions and drift monitoring

# Code quality
make lint             # Format checking
make format           # Automatic formatting
```

Note: make test runs the full ML pipeline (preprocess → train → predict) and is used locally in CodeSpace for development and debugging.

GitHub Actions - CI/CD

ci.yml (runs on push/PR to main and test)

- Format checking: black --check
- Linting: flake8
- Import sorting: isort --check-only
- Unit tests: pytest -v

lint-code.yml (runs on PR to main)

- Dependency installation: make requirements
- Linting: make lint

Key difference:

- **GitHub Actions** = code quality only (fast, no ML data)
- **Makefile** = full ML pipeline (local environment)

🖌️ Data Processing

- Target conversion (Yes/No → 1/0)
- Train/test split (stratified)
- Categorical column validation
- Class balancing: **class_weight + stratified sampling**

📈 Evaluation Metrics

Monitored:

- **F1-score (weighted)**
- Precision and Recall for positive class
- AUC-ROC
- Confusion Matrix
- LogLoss

📡 Monitoring and Support

- **MLflow** -- metric and model version tracking
- **NannyML** -- data drift detection
- **Git** -- version control

🧪 Code Quality

- **Formatting:** black (line-length 99)
- **Linting:** flake8
- **Import sorting:** isort
- **Unit tests:** pytest (in tests/ directory)
- **Documentation:** docstrings + README.md

🔒 Git Management

.gitignore - Exclusions

```
# Large ML files (not synchronized)

data/               # Data downloaded via Kaggle API

models/             # Models generated during training

mlruns/             # MLflow tracking data

mlartifacts/        # MLflow artifacts

reports/            # Reports and charts


# System files

__pycache__/

*.pyc
```

.venv/

.pytest_cache/

Benefits:

- Lightweight repository (few MB instead of hundreds)
- Fast cloning and CI/CD
- Reproducible data via make preprocess

📊 Experiment Results

Cross-Validation (N=5)

- Mean F1 Score: (requires improvement - imbalanced classes)
- Mean LogLoss: ~0.49 (after convergence)
- Standard deviation ≪ 0.05 (no overfitting signs)

SHAP Analysis

- Most important features: AgeCategory, GenHealth, Stroke, BMI
- Low impact features: Race → model does not discriminate

📜 Conclusions and Recommendations

Observations:

- F1-score below target KPI (≥0.75)
- Strong class imbalance requires additional techniques

Improvement recommendations:

- Apply SMOTE/ADASYN for synthetic sample generation
- Optimize classification threshold
- Ensemble methods (voting, stacking)
- Feature engineering (feature interactions)

Medical insights:

- Most important risk factors: older age, poor health status, stroke history, high BMI
- Interpretability ensured through SHAP
- Model requires further optimization for clinical applications