Projekt Klasyfikacji Chorób Serca

Opis Projektu

Klasyfikator chorób serca oparty na zbiorze danych CDC (**319,795 próbek**).
Projekt implementuje pipeline klasyfikacyjny do przewidywania występowania chorób serca na podstawie zestawu wskaźników medycznych i stylu życia.

Cel projektu

Stworzenie wydajnego modelu klasyfikacyjnego przewidującego występowanie chorób serca w populacji na podstawie danych medycznych i demograficznych.

Kluczowe Wskaźniki Wydajności (KPI)

- **F1-score (weighted) ≥ 0.75** (metryka zapisywana w MLflow)
- **LogLoss** jako wskaźnik jakości klasyfikacji probabilistycznej
- Stabilność metryk w walidacji krzyżowej (**std ≤ 0.05**)
- Automatyzacja doboru hiperparametrów przy użyciu **Optuna**
- Śledzenie zmian kodu i modeli poprzez **MLflow + Git**

⚠ Ocena Ryzyka

- **Ryzyko dysproporcji danych:** niezbalansowanie klas (91.4% „No”, 8.6% „Yes”) – wymaga zastosowania balansowania (class_weight, stratified sampling)
- **Ryzyko zależności od modelu:** użycie konkretnego frameworka (CatBoost)
- **Overfitting:** kontrolowane przez early stopping i walidację krzyżową
- **Ryzyko reprodukowalności:** minimalizowane przez kontrolę wersji (joblib, MLflow, git hash, random_state)

Opis Zbioru Danych

Zbiór składa się z **319,795 próbek i 18 kolumn** (17 cech + 1 etykieta).

Kolumny:

Kolumna	Opis
HeartDisease	Diagnoza choroby serca (Yes/No) – **zmienna docelowa**
BMI	Wskaźnik masy ciała
Smoking	Status palenia (Yes/No)
AlcoholDrinking	Spożywanie alkoholu (Yes/No)
Stroke	Przebyta udar (Yes/No)
PhysicalHealth	Dni złego zdrowia fizycznego (0–30)
MentalHealth	Dni złego zdrowia psychicznego (0–30)
DiffWalking	Trudności w chodzeniu (Yes/No)
Sex	Płeć (Male/Female)
AgeCategory	Kategoria wiekowa
Race	Rasa/pochodzenie
Diabetic	Status cukrzycy (Yes/No/Borderline/Yes during

	pregnancy)
PhysicalActivity	Aktywność fizyczna (Yes/No)
GenHealth	Ogólny stan zdrowia (Excellent/Very good/Good/Fair/Poor)
SleepTime	Godziny snu na dobę
Asthma	Astma (Yes/No)
KidneyDisease	Choroba nerek (Yes/No)
SkinCancer	Rak skóry (Yes/No)

Rozkład zmiennej docelowej

- No (brak choroby serca): **292,422 próbek (91.4%)**
- Yes (choroba serca): **27,373 próbek (8.6%)**

⚠ Wyraźna nierównowaga klas – zastosowano class_weights w CatBoost i stratified sampling.

🤖 Opis Modelu

Model: **CatBoostClassifier (gradient boosting)**

Cechy CatBoost

- natywne wsparcie dla danych kategorycznych (bez one-hot encodingu)
- ordered boosting – redukcja ryzyka overfittingu
- obsługa niezbalansowanych klas (class_weights)

🔧 Dostrajanie Hiperparametrów

- Implementacja: **Optuna** + mlflow.start_run(nested=True)
- Najlepsze parametry zapisywane w best_params.pkl i logowane w MLflow
- Kluczowe parametry optymalizacji:
  - iterations
  - learning_rate
  - depth
  - l2_leaf_reg
  - class_weights

🔁 Walidacja Krzyżowa

- Implementacja: catboost.cv z 5-fold stratified shuffle
- Monitorowane metryki: **F1, LogLoss, AUC-ROC**
- Wyniki wizualizowane z pasmami błędów przy użyciu **Plotly**

🏗 Struktura Projektu

├── .devcontainer/ # konfiguracja Codespaces / Dockera

├── .github/workflows/ # pipeline CI/CD │

```
├── data/                # dane projektu, dane pośrednie,przetworzone itp
├── models/              # zapisane modele i artefakty monitoringu
├── docs/                # dokumentacja
├── reports/             # raporty, wizualizacje
├── results/             # wyniki eksperymentów / predykcji
├── notebooks/           # notatniki Jupyter (EDA, eksperymenty)
├── ARISA_DSML/          # główny kod źródłowy
├── tests/               # testy jednostkowe / integracyjne
├── Makefile             # automatyzacja (lint, test, train)
├── README.md            # opis projektu
├── pyproject.toml       # konfiguracja pakietu i zależności
├── setup.cfg            # config narzędzi (flake8, black)
├── requirements.txt     # lista zależności
└── .gitignore           # ignorowane pliki i katalogi W ARISA_DSML
```

❁ Wymagania wstępne

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

🚀 Instalacja i Uruchomienie

1. Klonowanie repozytorium

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

   cd PZ_ARISA_MLOps_Final
   ```

2. Tworzenie środowiska

   ```
   python -m venv .venv

   # Windows

   .\.venv\Scripts\activate
   ```

Linux/Mac

source .venv/bin/activate

3. Instalacja zależności

   pip install -r requirements.txt

4. Dane wejściowe

   skopiuj heart_2020_cleaned.csv do data/raw/

5. Start MLflow UI

   mlflow ui

   Lub

   mlflow ui --host 127.0.0.1 --port 5000

🖌️ Przetwarzanie Danych

- Konwersja targetu (Yes/No → 1/0)
- Podział train/test (stratyfikowany)
- Walidacja kolumn kategorycznych
- Balansowanie klas: **class_weight + stratified sampling**

📈 Metryki Ewaluacji

Monitorowane:

- **F1-score (weighted)**
- Precision i Recall dla klasy pozytywnej
- AUC-ROC
- Confusion Matrix
- LogLoss

📡 Monitorowanie i Wsparcie

- **MLflow** – śledzenie metryk i wersji modeli
- **NannyML** – wykrywanie driftu danych
- **Git** – kontrola wersji

🔄 CI/CD Pipeline

Automatyczne workflowy:

- lint-code.yml – linting przy każdym PR na main
- ci.yml – linting, formatowanie i testy przy push/PR na main i test

Zabezpieczenia:

- ochrona gałęzi main
- wymagane code review
- pre-commit hooks (flake8, black)

🩹 Jakość Kodu

- Linting: flake8 + black + isort
- Testy jednostkowe: (do implementacji)
- Dokumentacja: docstringi + README.md

📊 Wyniki Eksperymentów

Cross-Validation (N=5)

- Mean F1 Score: ~0.77 (stabilny po ~50 iteracjach)
- Mean LogLoss: ~0.49 (po zbieżności)
- Standard deviation $\ll$ 0.05 (brak oznak overfittingu)

Analiza SHAP

- Najważniejsze cechy: AgeCategory, GenHealth, Stroke, BMI
- Cechy o małym wpływie: Race → model nie dyskryminuje

📜 Wnioski Medyczne

- Najważniejsze czynniki ryzyka: starszy wiek, słaby stan zdrowia, historia udaru, wysokie BMI
- Model osiągnął KPI (F1 ≈ 0.77, LogLoss ≈ 0.49)
- Interpretowalność zapewniona dzięki SHAP