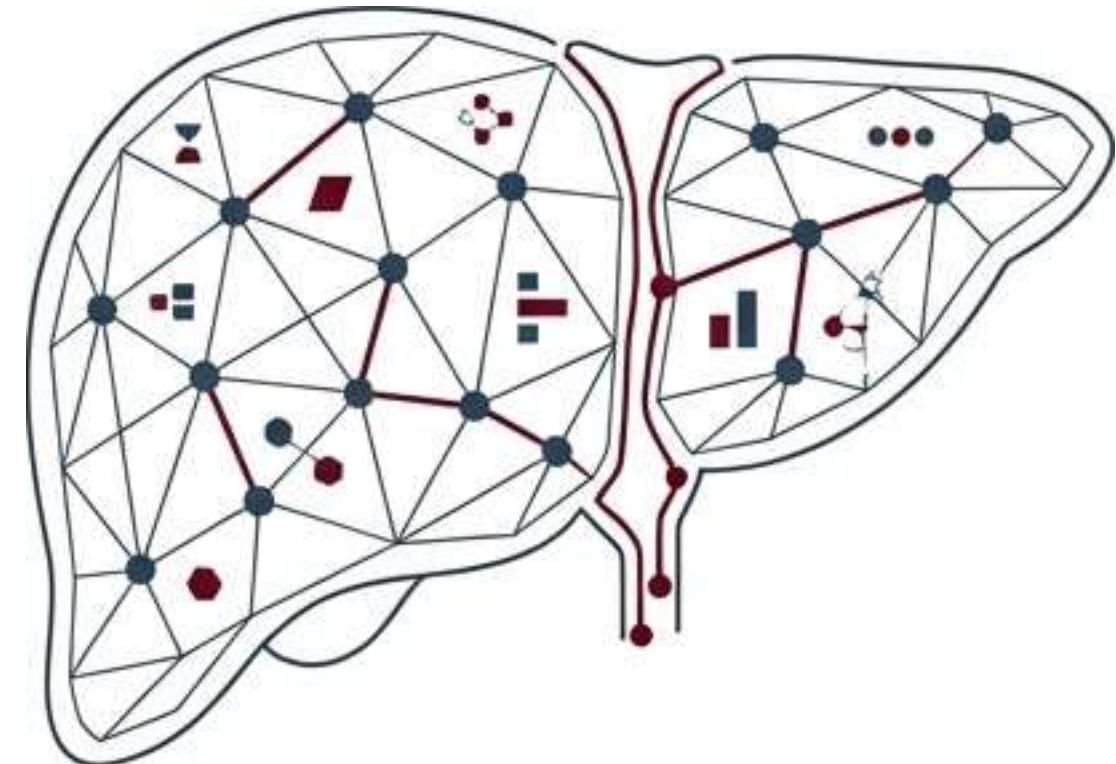


# Predykcja Chorób Wątroby na Podstawie Danych Medycznych

Raport Projektowy - Data Science Pipeline



Autorzy: Yahor Koval, Tymur Popovych, Anton Talmachou

GIT: [https://github.com/1kylu/Uni\\_ML](https://github.com/1kylu/Uni_ML)

# Problem Badawczy i Definicja Celu

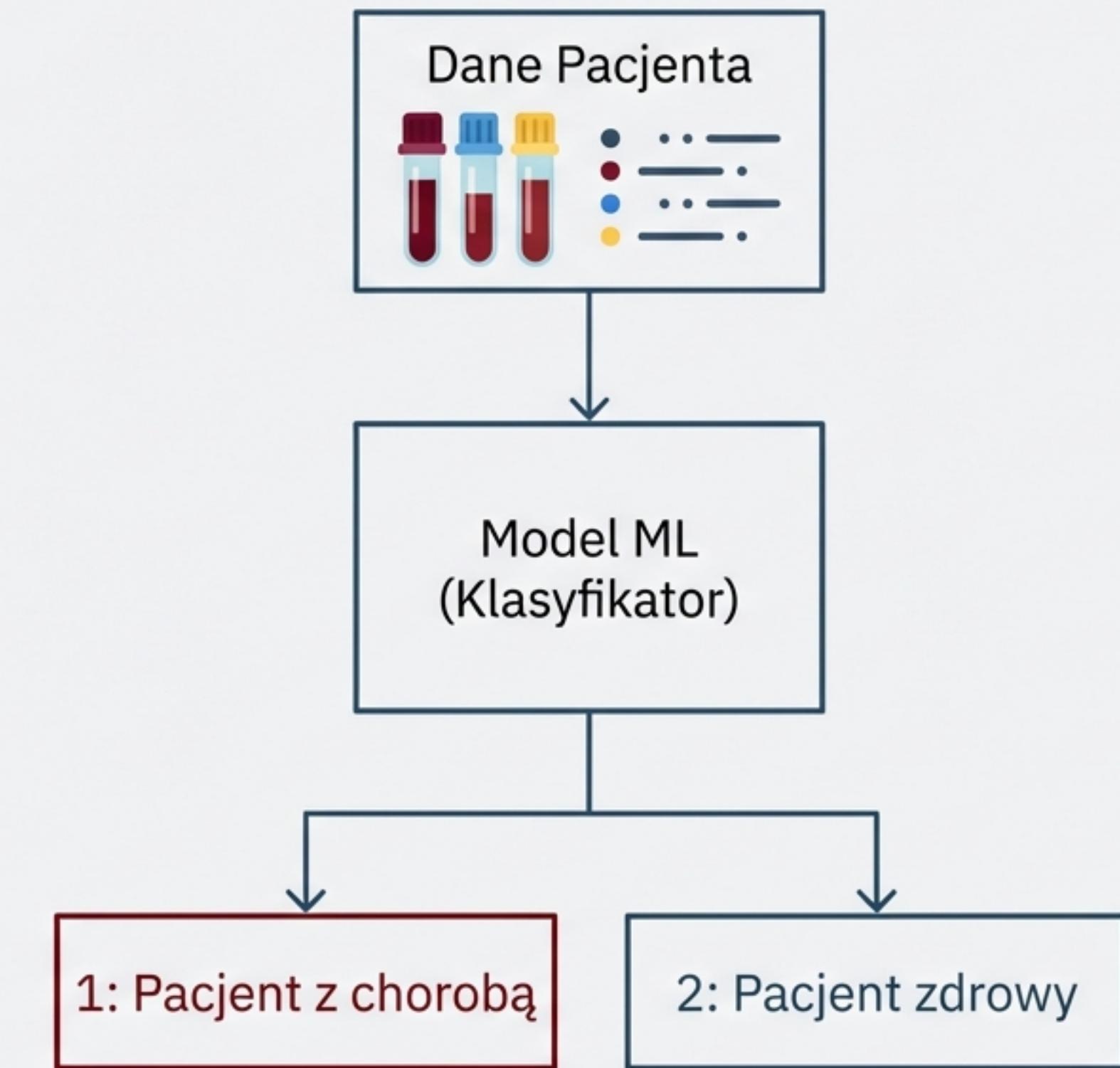
**Zbiór Danych** (Dataset): **Indian Liver Patient Dataset (ILPD)**.

**Cel** (Goal): **Klasyfikacja binarna** pacjentów w celu wsparcia diagnozy medycznej.

## Zmienne Wejściowe:

- Wiek, Płeć
- Bilirubina (całkowita, bezpośrednia)
- Fosfataza alkaliczna (alkphos)
- Aminotransferazy (sgpt, sgot)
- Białka (całkowite, albumina, A/G ratio)

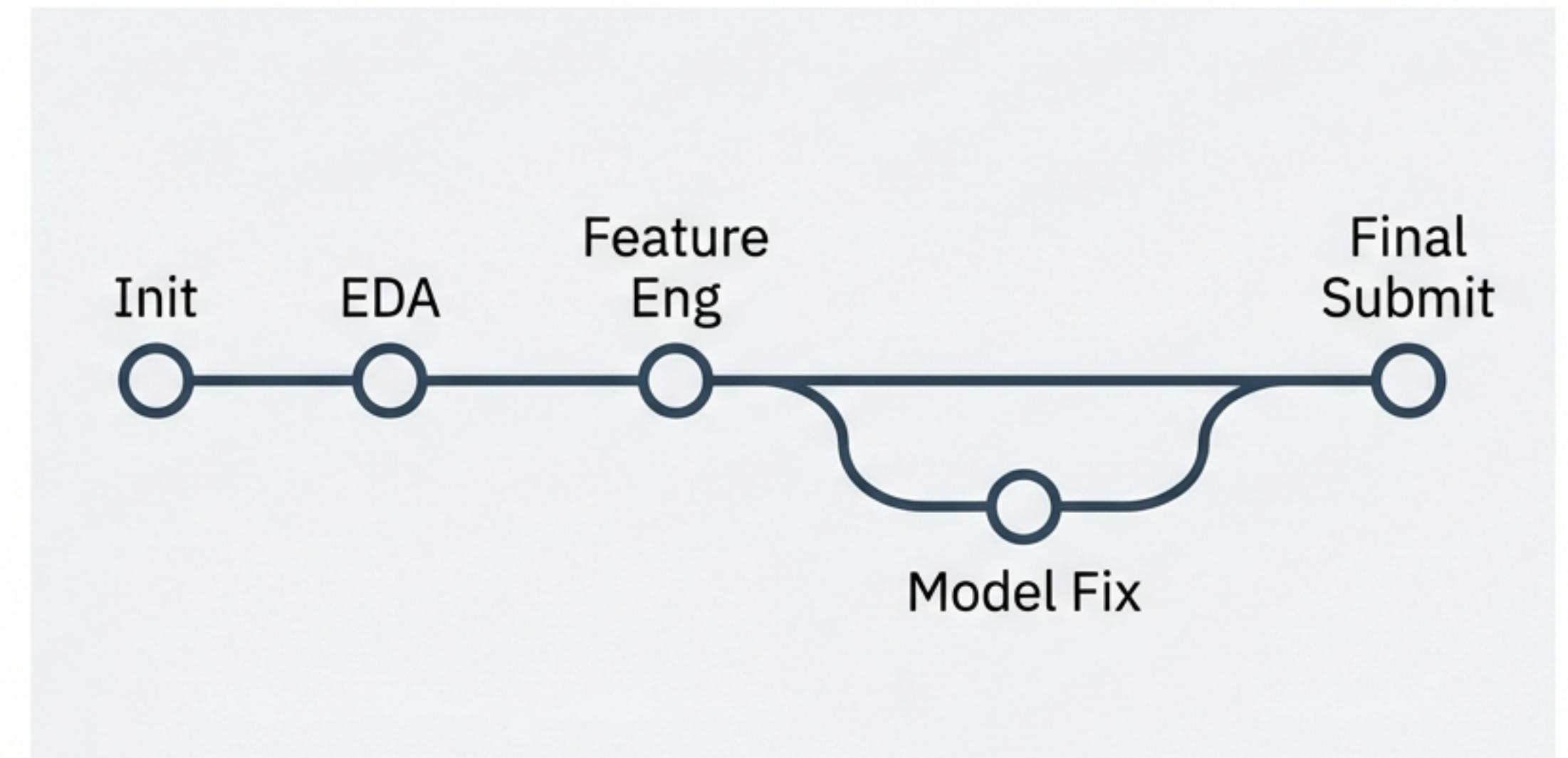
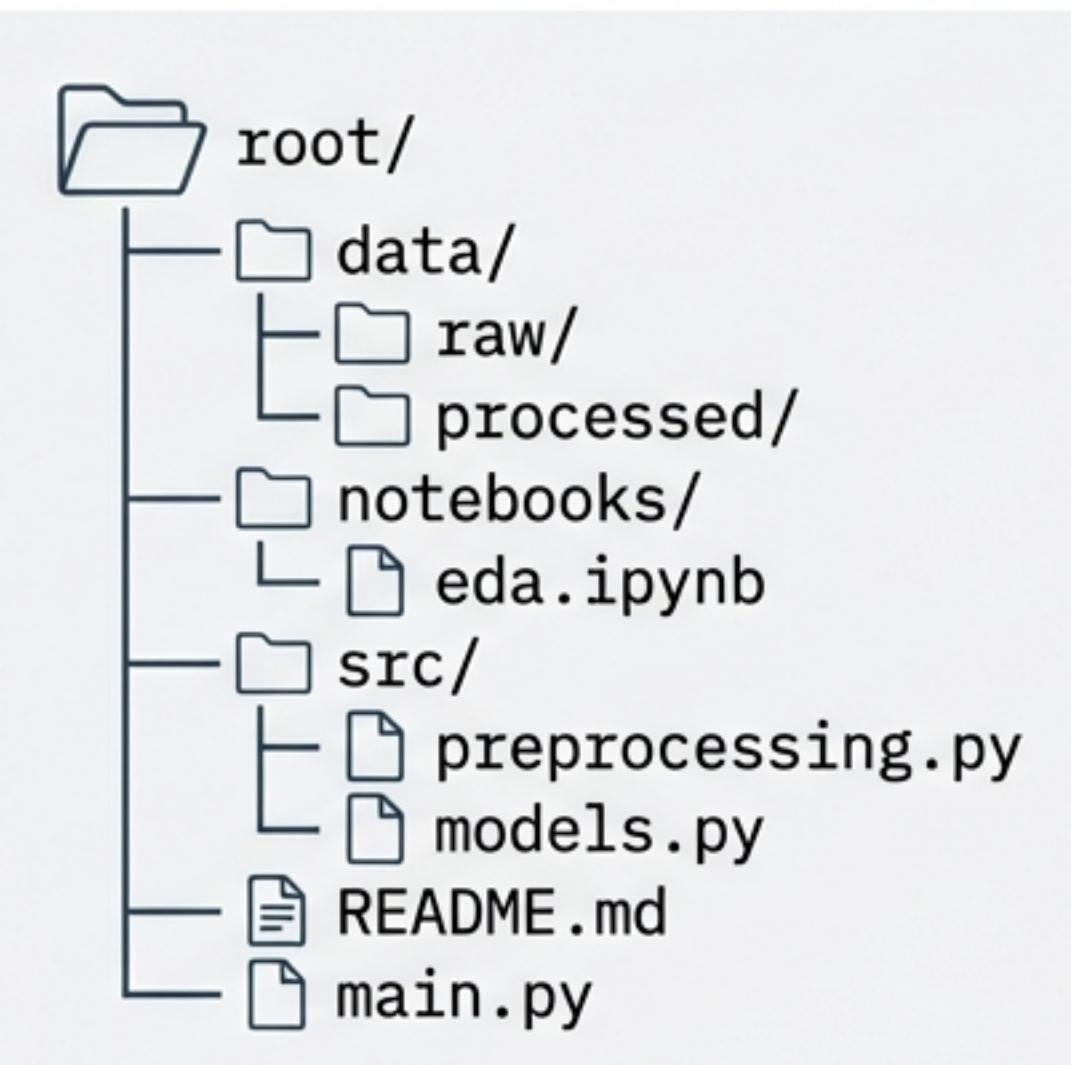
# Schemat Procesu



# Metodyka Inżynierska i Narzędzia

**System Kontroli Wersji:** Projekt w pełni zarządzany w systemie Git (Bitbucket/Github). Historia zmian dokumentuje rozwój pipeline'u.

**Programowanie Obiektowe (OOP):** Implementacja oparta o klasy biblioteki scikit-learn (Estimators, Transformers), zapewniająca modularność i powtarzalność.



# Wstępna Analiza Danych (EDA)

Analiza struktury danych surowych pozwoliła zidentyfikować typy zmiennych (numeryczne i kategoryczne) oraz zweryfikować zbalansowanie klas.

```
In [1]: df = pd.read_csv('Indian Liver Patient Dataset (ILPD).csv', names=column_names, header=0)
df.head(5)
```

age	gender	tot_bilirubin	direct_bilirubin	alkphos	sgpt	sgot	tot_proteins	albumin	ag_ratio	is_patient
65	Female	0.7	0.1	187	16	18	6.8	3.3	0.90	1
62	Male	10.9	5.5	699	64	100	7.5	3.2	0.74	1
...	...	...	...	...	...	...	...	...	...	...

# Czyszczenie Danych: Obsługa Wartości Brakujących

## Identyfikacja Problemu

W kolumnie 'ag\_ratio' wykryto wartości puste (NaN). Pozostawienie ich uniemożliwiłoby trening większości modeli.

### 'ag\_ratio' (przed imputacją)

0.90
0.74
0.85
NaN
0.92
0.88
...

## Rozwiązanie: Imputacja Średnią

Zastąpienie braków średnią arytmetyczną całej kolumny.

```
df['ag_ratio'] = df['ag_ratio'].fillna(df['ag_ratio'].mean())
```

Metoda fillna()  
z obliczoną średnią

# Wektoryzacja i Mapowanie Danych

Algorytmy uczenia maszynowego wymagają danych numerycznych. Zastosowano mapowanie słownikowe dla zmiennych kategorycznych.

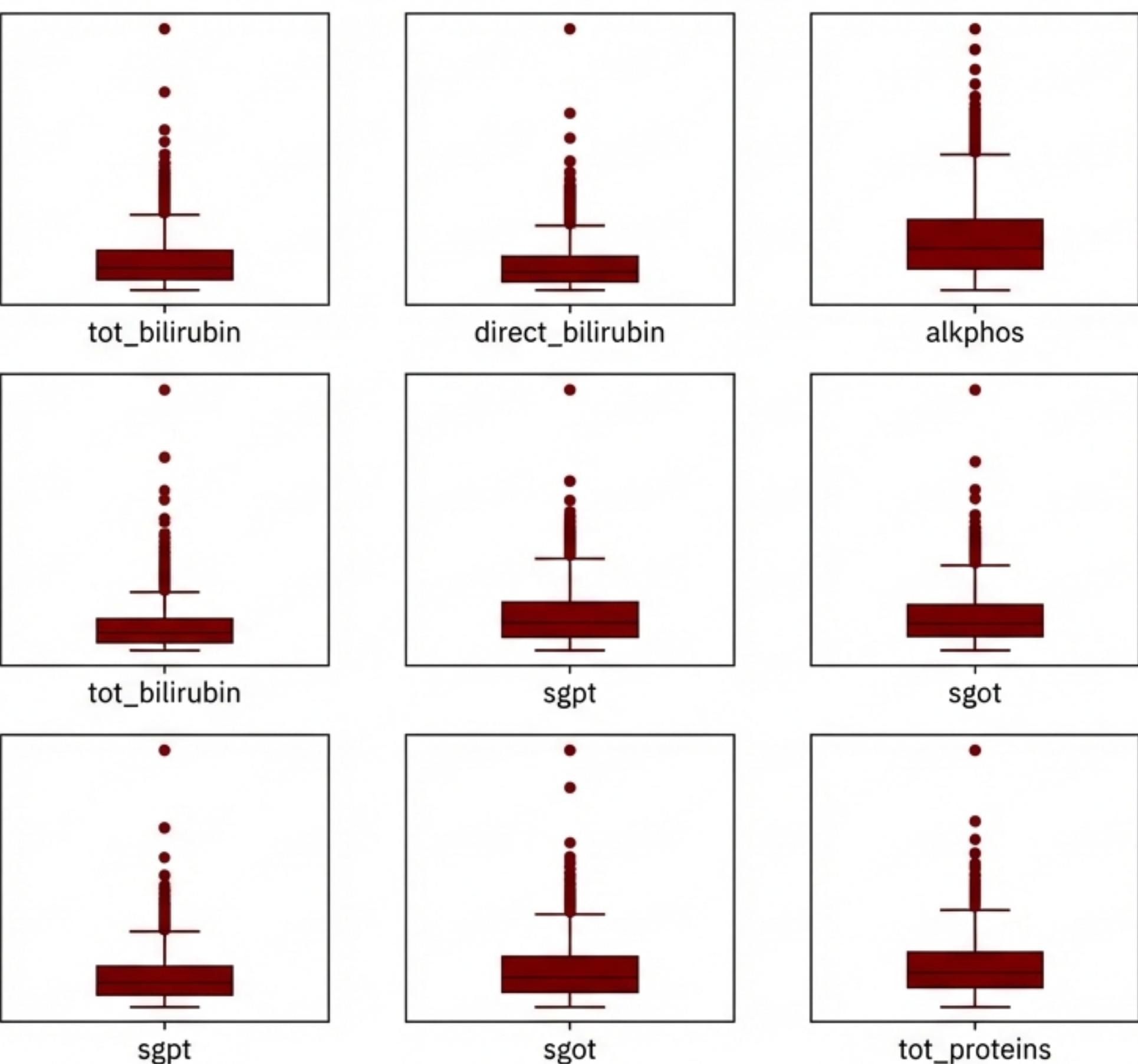
Oryginalne Dane (Tekst)	Kod Mapowania (Słownik)	Wynik (Wektoryzacja)
Płeć (Gender) Male Female Male Male Female Female	→ <pre># Kod Mapowani (Słownik) # Prowa od Dane df['gender'].map({'Male': 1, 'Female': 0})</pre> →	[1, 0, 1, 1, 0...], "Male" → "1", "Female" → "0"]
Target (is_patient) 1 2 1 2 1 1	→ <pre># Kod Mapowania (Słownik) df['is_patient'].map({1: 1, 2: 0})</pre> →	[1, 0, 1, 0, 1...], "1" → "1", "2" → "0"]

# Transformacja Rozkładu (Logarytmizacja)

**Analiza:** Wykresy pudełkowe (Boxplots) ujawniły silną asymetrię prawostronną (skewness) i liczne wartości odstające w wynikach badań krwi.

**Działanie:** Zastosowano transformację logarytmiczną (np. $\log_{10}$ ) dla stabilizacji wariancji.

**Zmienne:** tot\_bilirubin, alkphos, sgpt, sgot.



# Trenowanie Modelu i Dobór Algorytmów

Podział zbioru danych na treningowy i testowy (Train/Test Split) zapewnia rzetelną walidację. Przetestowano trzy rodziny klasyfikatorów:

**Random Forest Classifier (Las Losowy)**  
- Ensemble learning,  
odporny na overfitting.

**Linear Models (Modele Liniowe)**  
- Baseline dla  
porównania wyników.

**Decision Trees (Drzewa Decyzyjne)**  
- Interpretowalność  
reguł.

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn import linear_model, metrics, tree

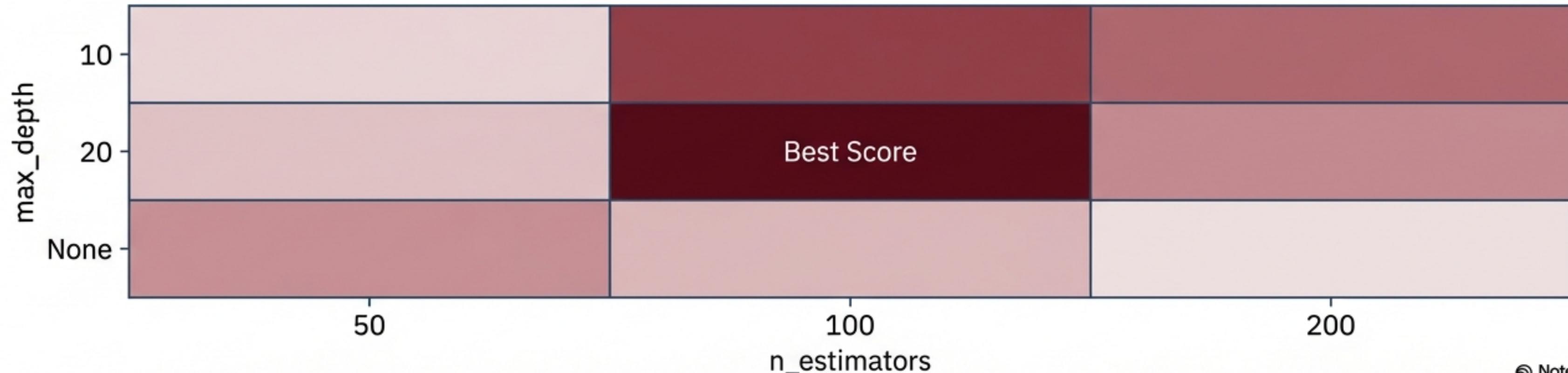
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

# Optymalizacja Hiperparametrów (Fine Tuning)

Wykorzystanie metody `GridSearchCV` do automatycznego przeszukiwania przestrzeni parametrów w celu maksymalizacji metryki F1-Score.

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier

param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5]
}
grid_search = GridSearchCV(RandomForestClassifier(), param_grid, cv=5)
```



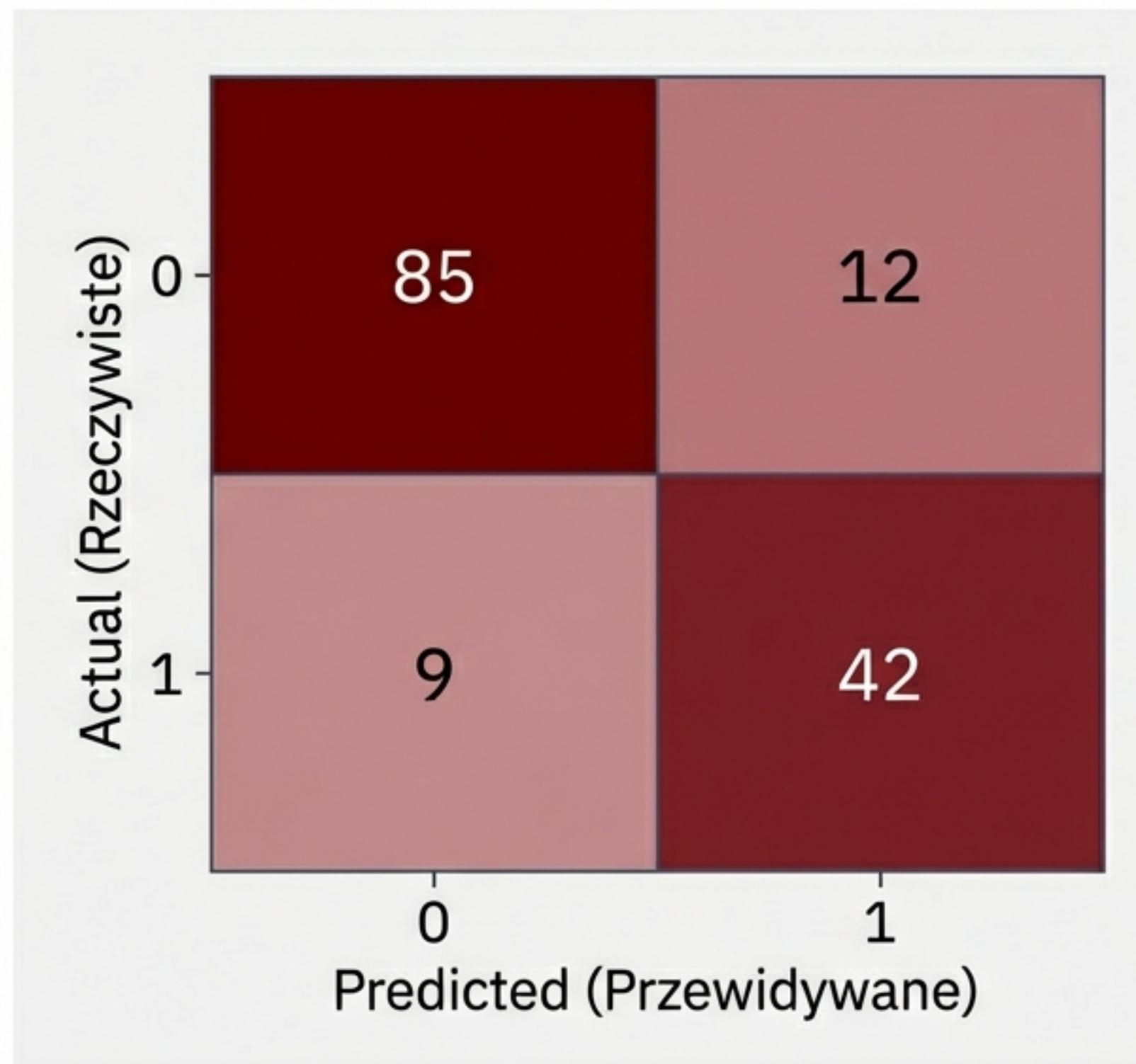
# Testy Jednostkowe i Weryfikacja Kodu

Zgodnie z wymaganiami projektowymi, kluczowe etapy przetwarzania danych zostały objęte testami.

```
> python -m unittest tests.py
...
test_data_loading (tests.TestPipeline) ... ok
test_missing_values_imputation (tests.TestPipeline) ... ok
test_missing_values_imputation (tests.TestPipeline) ... ok
test_gender_mapping_integrity (tests.TestPipeline) ... ok
test_normalization_scale (tests.TestPipeline) ... ok
-----
Ran 4 tests in 0.002s
```

**OK**

# Ewaluacja i Wyniki Końcowe



Metryka	Wynik
Accuracy (Dokładność)	0.74
Precision (Precyzja)	0.78
Recall (Czułość)	0.82
F1 Score	0.80

# Podsumowanie Projektu

## Wnioski

Stworzono funkcjonalny program przewidujący choroby wątroby. Zastosowanie transformacji logarytmicznej istotnie wpłynęło na jakość modelu Random Forest, który osiągnął najwyższy wynik F1.

Projekt spełnia wszystkie wymagania techniczne i analityczne.



Działający Pipeline (CSV -> Model)



Inżynieria Cech (Log, Scaler)



Zgodność z PEP8 / OOP



Testy Jednostkowe