Uniwersytet Łódzki
Wydział Fizyki i Informatyki Stosowanej

SPRAWOZDANIE Z LABORATORIUM
Zaawansowane metody obliczeniowe

Regresja wielowymiarowa liniowa i logistyczna
implementacja własna i biblioteczna

Autor: Katarzyna Stańczyk
Kierunek studiów: Informatyka
Data wykonania: 24 stycznia 2026
Rok akademicki: 2025/2026

Łódź, 2026

1 Wstęp

Celem ćwiczenia było dopasowanie modelu regresyjnego do zadanego zbioru danych oraz porównanie implementacji własnej z rozwiązaniem bibliotecznym. Wykorzystano zbiór *Banknote Authentication* (UCI), w którym zmienna docelowa przyjmuje wartości $y \in \{0, 1\}$, dlatego rozważany problem ma charakter klasyfikacji i zastosowano regresję logistyczną.

W pracy przeprowadzono uczenie modelu w dwóch wariantach: (1) implementacja własna uczona metodą spadku gradientowego oraz (2) model referencyjny `LogisticRegression` z biblioteki `scikit-learn`. Ocenę jakości przeprowadzono na zbiorze testowym, analizując zarówno poprawność klasyfikacji, jak i jakość predykcji probabilistycznych. Wyniki oraz wykresy zapisano w katalogu uruchomienia, a pełne podsumowanie eksperymentu umieszczono w pliku `summary.json`.

1.1 Regresja

1.1.1 Regresja liniowa

Regresja liniowa jest używana, gdy zmienna docelowa y jest ciągła (np. cena, temperatura). Model ma postać:

$$\hat{y}(x) = w^T x + b, \tag{1}$$

gdzie w to wagi, a b to wyraz wolny. Taki model zwraca dowolną liczbę rzeczywistą, więc nie nadaje się bezpośrednio do przewidywania prawdopodobieństwa. W tym ćwiczeniu dane mają klasy 0/1, dlatego właściwym wyborem jest regresja logistyczna.

1.1.2 Regresja logistyczna

Regresja logistyczna jest stosowana, gdy zmienna docelowa jest binarna, czyli $y \in \{0, 1\}$. Najpierw liczymy wartość liniową:

$$z = w^T x + b, \tag{2}$$

a następnie zamieniamy ją na prawdopodobieństwo klasy 1 za pomocą funkcji sigmoidalnej:

$$p(y = 1 \mid x) = \sigma(z), \qquad \sigma(z) = \frac{1}{1 + e^{-z}}. \tag{3}$$

Dzięki temu wynik zawsze jest w przedziale $[0, 1]$ i można go interpretować jako prawdopodobieństwo.

Aby zamienić prawdopodobieństwo na klasę, stosujemy próg decyzyjny, np.:

$$\hat{y} = \begin{cases} 1 & \text{gdy } p \geq 0.5, \\ 0 & \text{gdy } p < 0.5. \end{cases} \tag{4}$$

W pracy przyjęto próg 0.5, ponieważ jest to najczęściej stosowana wartość domyślna.

1.1.3 Funkcja straty cross-entropy (log-loss)

Do uczenia regresji logistycznej używa się funkcji straty opartej o entropię krzyżową (log-loss):

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log(p_i) + (1-y_i)\log(1-p_i)\right), \tag{5}$$

gdzie p_i to przewidywane prawdopodobieństwo klasy 1 dla próbki i.
Intuicyjnie: model jest karany, gdy jest pewny, ale się myli (np. daje $p \approx 1$ dla próbki z klasą 0). W implementacji praktycznej ogranicza się p do przedziału $[\varepsilon, 1-\varepsilon]$, żeby uniknąć problemu z $\log(0)$.

1.1.4 Gradient i aktualizacja wag

W implementacji własnej zastosowano spadek gradientowy w wersji macierzowej (uczenie pełnobatchowe). Dla zbioru uczącego $X \in \mathbb{R}^{n\times d}$ oraz wektora wag $w \in \mathbb{R}^d$:

$$p = \sigma(Xw), \tag{6}$$

gdzie σ działa elementowo.
Gradient funkcji straty względem wag ma postać:

$$\nabla_w \mathcal{L} = \frac{1}{n} X^T (p - y). \tag{7}$$

Aktualizacja wag w każdej epoce:

$$w \leftarrow w - \alpha \nabla_w \mathcal{L}, \tag{8}$$

gdzie α to współczynnik uczenia.
W kodzie wyraz wolny b został uwzględniony przez dodanie do macierzy X dodatkowej kolumny jedynek (tzw. *bias* w macierzy).

1.1.5 Standaryzacja cech

Przed uczeniem wykonano standaryzację cech (`StandardScaler`): każda cecha jest przekształcana tak, aby miała średnią 0 i odchylenie standardowe 1. To ważne, bo przy różnych skalach cech spadek gradientowy może działać wolniej albo niestabilnie.

Standaryzację dopasowano tylko na zbiorze treningowym, a następnie zastosowano do zbioru testowego. Dzięki temu nie dochodzi do "przecieku" informacji ze zbioru testowego do treningu.

1.1.6 Metryki oceny jakości

Do oceny jakości klasyfikacji użyto następujących miar:

- **accuracy** – odsetek poprawnych klasyfikacji,
- **precision** – jaki procent przewidzianych jedynek jest poprawny,
- **recall** – jaki procent prawdziwych jedynek został wykryty,
- **F1** – średnia harmoniczna precision i recall,
- **log-loss** – jakość przewidywanych prawdopodobieństw (im mniejszy, tym lepiej),
- **ROC-AUC** – pole pod krzywą ROC (im bliżej 1, tym lepiej).

Dodatkowo pokazano **macierz pomyłek**, która podsumowuje liczbę trafień i pomyłek dla klas 0 oraz 1.

1.2 Model

1.2.1 Parametry i hiperparametry

W modelu regresji logistycznej wyróżniono:

- **parametry modelu** (wynik uczenia): wagi w oraz wyraz wolny b,
- **hiperparametry** (ustawiane przed uczeniem): współczynnik uczenia α, liczba epok, sposób inicjalizacji wag oraz próg klasyfikacji.

W eksperymentach użyto następujących ustawień:

- podział danych: 20% na test (`test_size`=0.2), ziarno losowości `seed`=123,
- uczenie manualne: $\alpha = 0.05$, liczba epok `EPOCHS`=2000, inicjalizacja `zeros`, regularyzacja $L2 = 0.0$,
- klasyfikacja: próg decyzyjny `THRESHOLD`=0.5,
- model biblioteczny: `LogisticRegression` z `max_iter`=5000 oraz $C = 10^6$ (minimalna regularyzacja).

1.3 Współczynnik uczenia

Współczynnik uczenia α określa, jak duży krok wykonywany jest podczas aktualizacji wag w metodzie spadku gradientowego. Gdy α jest zbyt małe, uczenie jest wolne (strata spada bardzo wolno). Gdy α jest zbyt duże, uczenie może być niestabilne (strata może rosnąć lub oscylować). W tym ćwiczeniu dobrano $\alpha = 0.05$ jako kompromis pomiędzy szybkością uczenia a stabilnością zbieżności, co można ocenić na wykresie straty w funkcji epok.

2 Rozwinięcie

2.1 Środowisko i konfiguracja programu

Program został napisany w języku **Python 3.12** i uruchamiany lokalnie w środowisku **PyCharm 2024.3.4**. Kod realizuje pełny przebieg eksperymentu: pobranie danych, podział na zbiory treningowy i testowy, standaryzację cech, trening modelu w wersji manualnej i bibliotecznej, obliczenie metryk oraz zapis wyników do plików.

Parametry eksperymentu (m.in. `LR`, `EPOCHS`, `TEST_SIZE`, `SEED`, `THRESHOLD`) zostały zdefiniowane bezpośrednio w kodzie, dzięki czemu uruchomienie programu nie wymaga podawania argumentów z linii poleceń. Reprodukowalność wyników zapewniono przez ustawienie stałego ziarna losowości `SEED` oraz zapis pełnego podsumowania do pliku `summary.json`.

Dane wejściowe pochodzą ze zbioru **Banknote Authentication** (UCI). Program automatycznie pobiera plik `data_banknote_authentication.txt` przy pierwszym uruchomieniu (jeśli nie znajduje się on lokalnie).

2.2 Wykorzystane biblioteki i ich rola

W projekcie wykorzystano następujące biblioteki:

- **`os`** – obsługa plików i katalogów: sprawdzanie, czy plik danych już istnieje, tworzenie folderu wynikowego `runs/...` oraz katalogu `plots/`.
- **`json`** – zapis podsumowania eksperymentu do pliku `summary.json` (parametry uruchomienia, metryki, wagi, macierze pomyłek).
- **`time`** – generowanie unikalnej nazwy folderu wynikowego na podstawie czasu uruchomienia (np. `20260117_114050`).
- **`urllib.request`** – pobieranie danych z internetu (UCI) bez ręcznego ściągania pliku.
- **`numpy (np)`** – obliczenia macierzowe i wektorowe: wczytanie danych, przygotowanie macierzy cech, implementacja funkcji sigmoidalnej, obliczanie gradientu i aktualizacja wag w manualnej regresji logistycznej.
- **`scikit-learn (sklearn)`** – część biblioteczna oraz narzędzia pomocnicze:
  - `train_test_split` – podział danych na zbiory trening/test, z opcją `stratify` w celu zachowania proporcji klas,
  - `StandardScaler` – standaryzacja cech (średnia 0, odchylenie standardowe 1); skaler dopasowywany jest wyłącznie na zbiorze treningowym, a następnie stosowany do zbioru testowego,
  - `LogisticRegression` – gotowa implementacja regresji logistycznej (punkt odniesienia do porównania),
  - `sklearn.metrics` – obliczanie metryk: *accuracy*, *precision*, *recall*, *F1*, *log-loss*, *ROC-AUC* oraz wyznaczanie krzywych ROC i Precision–Recall.
- **`matplotlib`** – generowanie wykresów i zapis do plików PNG. Ustawiono backend `Agg`, aby wykresy mogły zapisywać się do plików bez otwierania okna graficznego.

2.3 Struktura plików wynikowych

Po uruchomieniu programu tworzony jest katalog wynikowy `runs/<data_uruchomienia>/`, a w nim:

- `summary.json` – podsumowanie eksperymentu (parametry, metryki train/test, wagi, macierze pomyłek oraz miary porównania prawdopodobieństw),
- `plots/` – folder zawierający wygenerowane wykresy w formacie PNG.

2.4 Opis danych i przygotowanie wejścia

Zbiór **Banknote Authentication** zawiera cztery cechy liczbowe: `variance`, `skewness`, `curtosis`, `entropy` oraz etykietę klasy `class` równą 0 lub 1. Ponieważ zmienna docelowa jest binarna, zadanie ma charakter klasyfikacji i zastosowano **regresję logistyczną**.

Przed uczeniem wykonano:

- **podział danych na train/test** w proporcji `(1-TEST_SIZE)/TEST_SIZE` z użyciem stratyfikacji po klasie (`stratify=y`),
- **standaryzację cech** (`StandardScaler`): parametry standaryzacji wyznaczono na zbiorze treningowym, a następnie zastosowano tę samą transformację dla zbioru testowego, co zapobiega „przeciekowi informacji" ze zbioru testowego.

2.5 Przetwarzanie danych wejściowych i przebieg eksperymentu

Eksperyment obejmuje dwa modele:

- **model manualny** – implementacja regresji logistycznej uczonej metodą pełnobatchowego spadku gradientowego (parametry: `LR`, `EPOCHS`, `INIT`, opcjonalnie `L2` oraz próg decyzyjny `THRESHOLD`),
- **model biblioteczny (sklearn)** – `LogisticRegression` z limitem iteracji `max_iter` oraz parametrem `C=SKLEARN_C` ustawionym na dużą wartość w celu ograniczenia wpływu regularyzacji.

Dla obu modeli obliczono predykcje prawdopodobieństw oraz etykiety klas (na podstawie progu `THRESHOLD`), a następnie wyznaczono metryki jakości osobno dla zbioru treningowego i testowego. Na końcu wygenerowano wykresy oraz zapisano podsumowanie eksperymentu do pliku `summary.json`.

2.6 Wyniki

Wyniki eksperymentu zapisano w pliku `summary.json` oraz w postaci wykresów w katalogu `plots/`. W tabeli 1 zestawiono metryki dla zbioru treningowego i testowego dla obu podejść: implementacji własnej oraz bibliotecznej (`sklearn`).

Tabela 1: Metryki jakości dla regresji logistycznej: implementacja własna vs `sklearn`.

	Manual		**sklearn**	
Metryka	**Train**	**Test**	**Train**	**Test**
Accuracy	9.7450	9.9270	9.7900	9.9270
Precision	9.5100	9.8390	9.5860	9.8390
Recall	9.9390	1.0000	9.9590	1.0000
F1	9.7190	9.9190	9.7690	9.9190
Log-loss	8.3500	7.2500	5.0600	4.0900
ROC-AUC	9.9920	9.9970	9.9960	9.9990

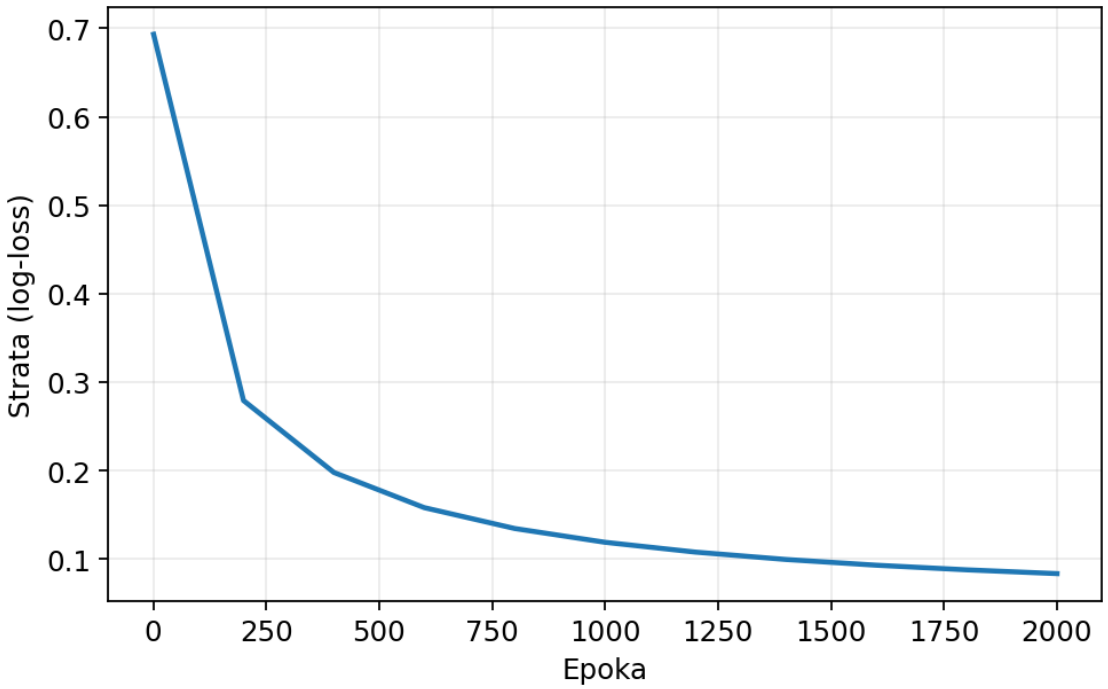
Dodatkowo, na zbiorze testowym dla obu metod uzyskano taką samą macierz pomyłek:

- TN = 151, FP = 2,
- FN = 0, TP = 122.

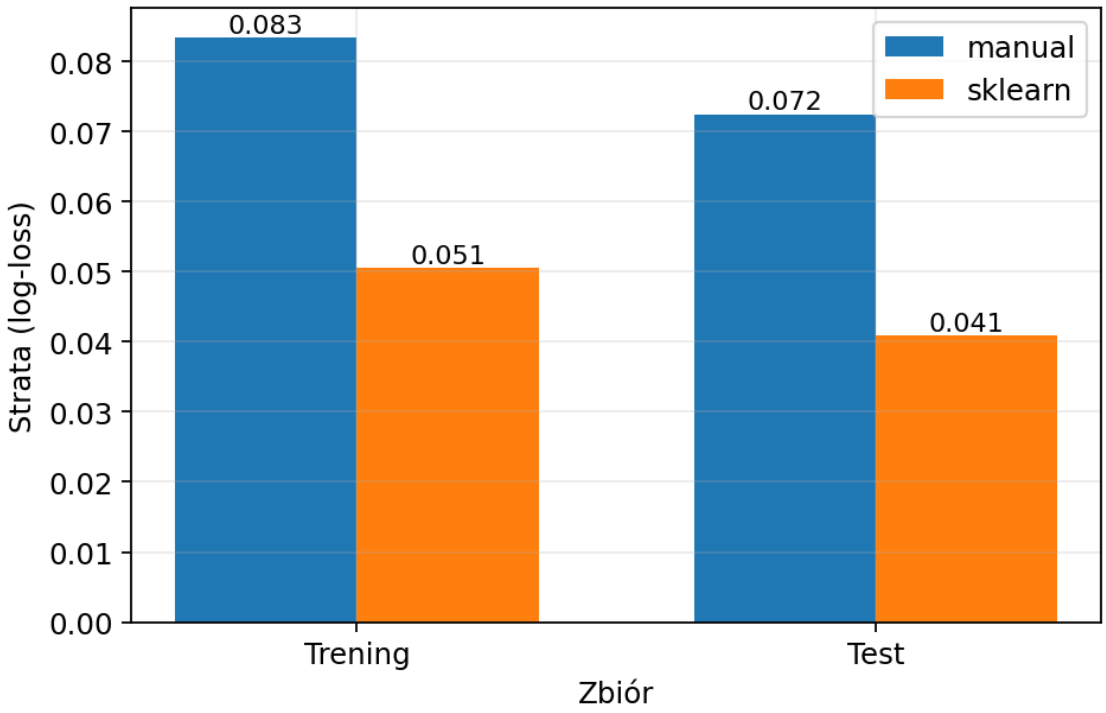
Wygenerowane wykresy (katalog `plots/`) obejmują:

- `loss_manual.png` – przebieg straty (log-loss) podczas uczenia implementacji własnej,
- `loss_porownanie.png` – porównanie log-loss na zbiorze treningowym i testowym (manual vs sklearn),
- `cm_manual.png`, `cm_sklearn.png` – macierze pomyłek,
- `roc_manual.png`, `roc_sklearn.png` – krzywe ROC,
- `pr_manual.png`, `pr_sklearn.png` – krzywe Precision–Recall,
- `proba_scatter.png` – porównanie prawdopodobieństw manual vs sklearn (wykres punktowy),
- `proba_diff_hist.png` – histogram różnic prawdopodobieństw (manual − sklearn),
- `wagi_porownanie.png` – porównanie wag cech (po standaryzacji i normalizacji wag).

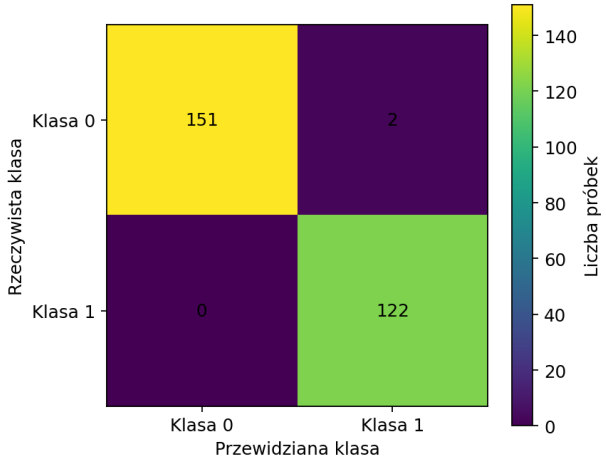
Wyraz wolny (*bias*/intercept) wypisywany jest osobno w konsoli, ponieważ w porównaniu wag porównywane są tylko wagi cech.



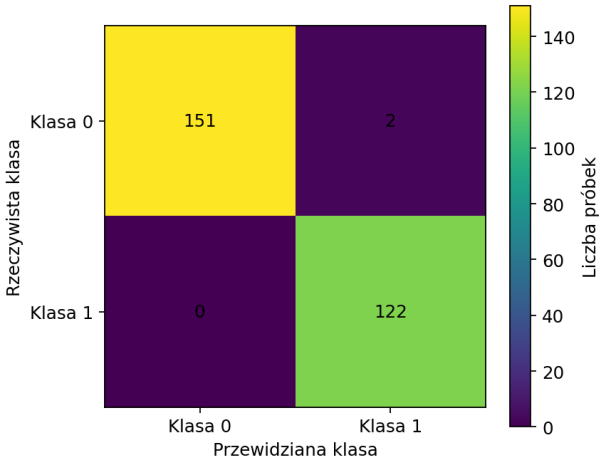
Rysunek 1: Przebieg funkcji straty (log-loss) podczas uczenia implementacji własnej (manual) w kolejnych epokach.



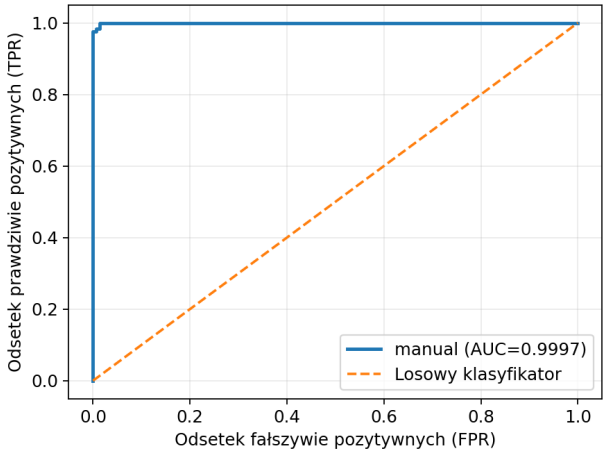
Rysunek 2: Porównanie wartości log-loss dla zbioru treningowego i testowego: implementacja własna (manual) vs `sklearn`.



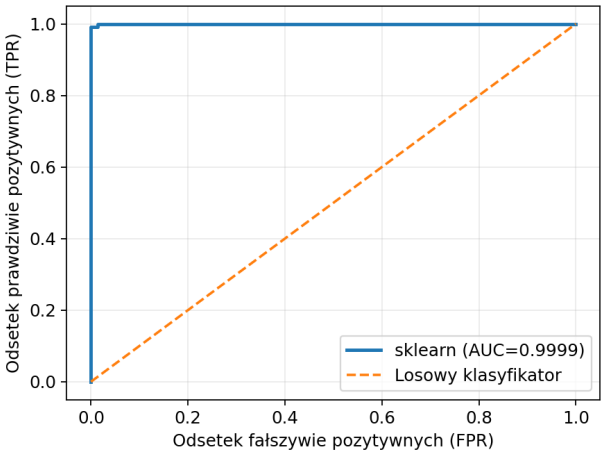
Rysunek 3: Macierz pomyłek dla implementacji własnej (manual) na zbiorze testowym.



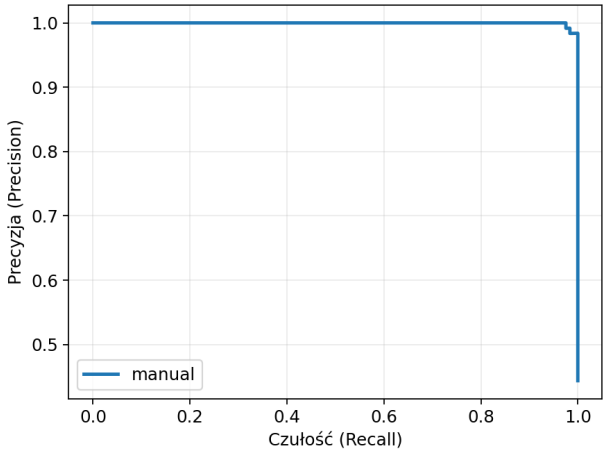
Rysunek 4: Macierz pomyłek dla modelu `sklearn` na zbiorze testowym.



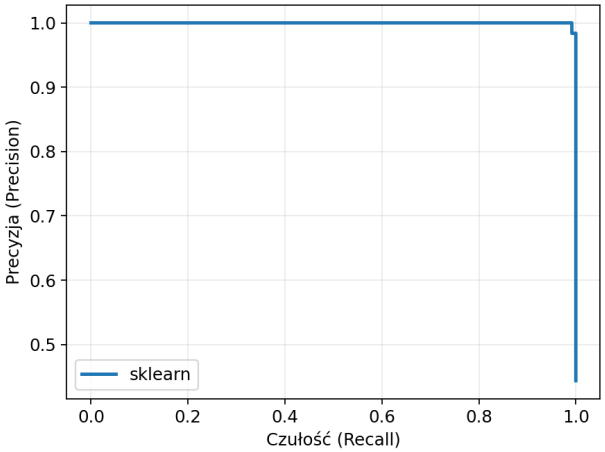
Rysunek 5: Krzywa ROC dla implementacji własnej (manual) na zbiorze testowym.



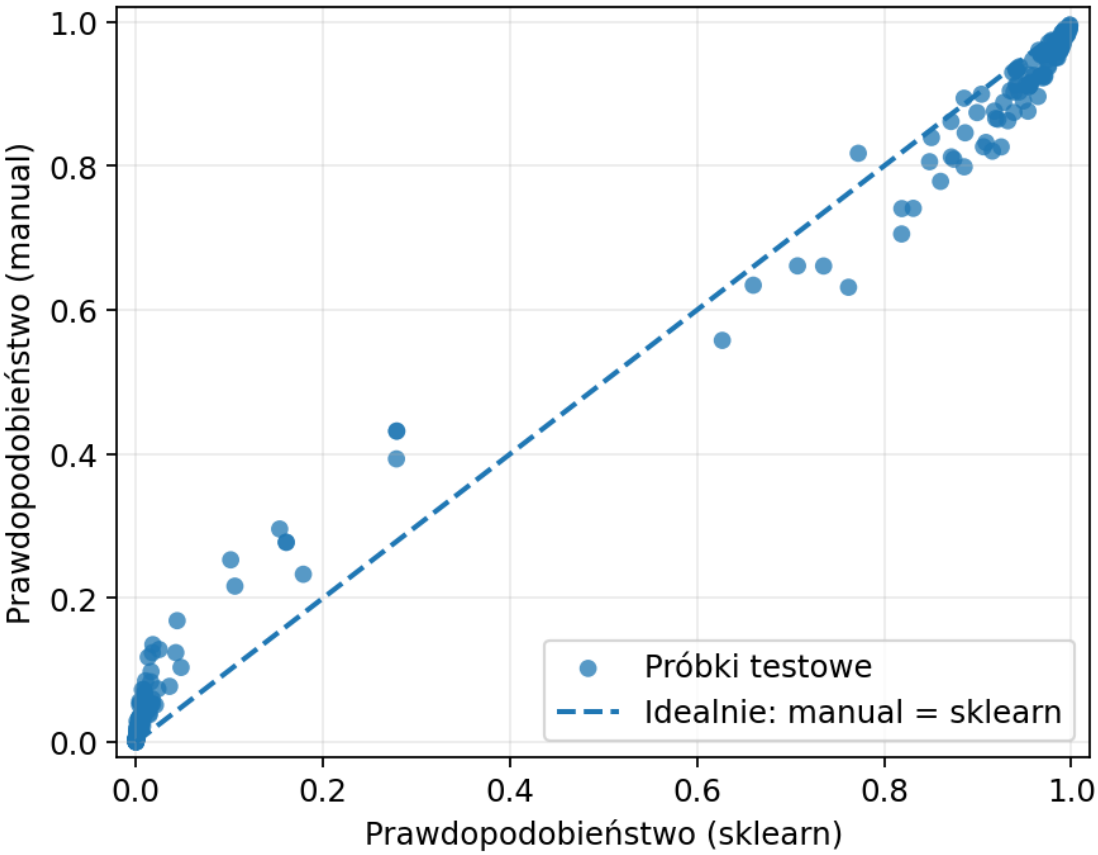
Rysunek 6: Krzywa ROC dla modelu `sklearn` na zbiorze testowym.



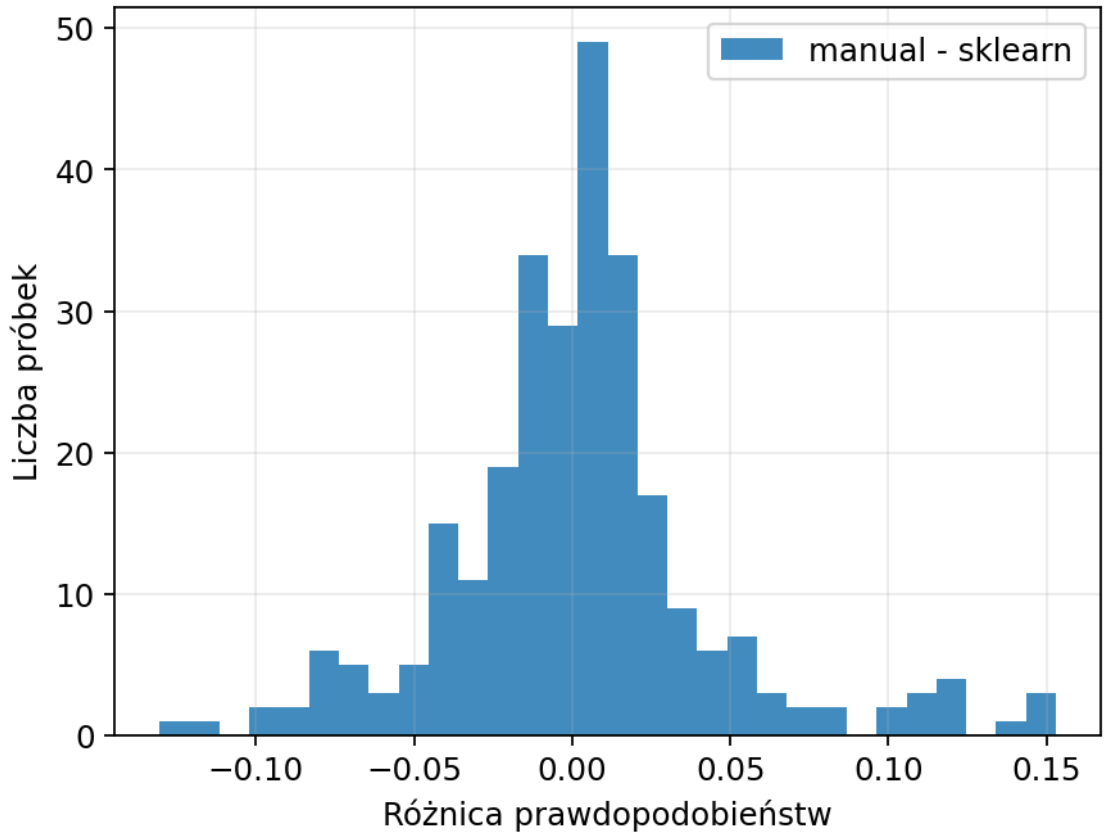
Rysunek 7: Krzywa Precision–Recall dla implementacji własnej na zbiorze testowym.



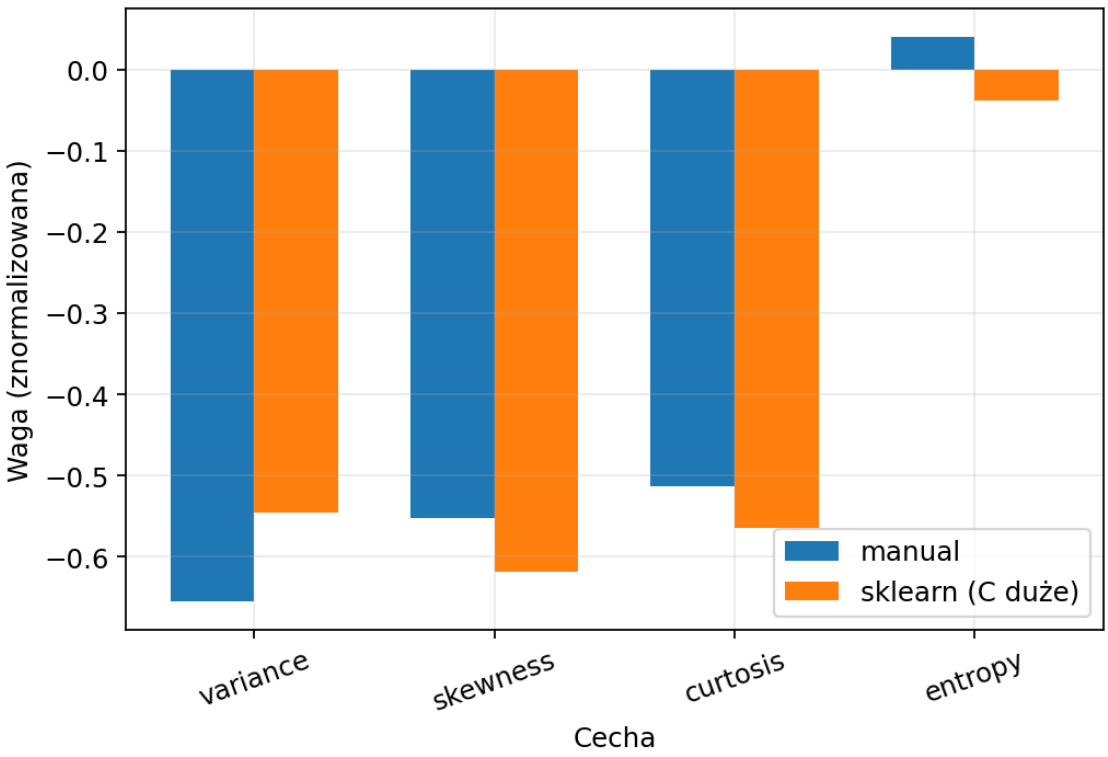
Rysunek 8: Krzywa Precision–Recall dla modelu `sklearn` na zbiorze testowym.



Rysunek 9: Porównanie prawdopodobieństw na zbiorze testowym: implementacja własna (manual) vs `sklearn`. Linia przerywana oznacza idealną zgodność ($p_{\text{manual}} = p_{\text{sklearn}}$).



Rysunek 10: Histogram różnic prawdopodobieństw na zbiorze testowym ($p_{\text{manual}} - p_{\text{sklearn}}$). Wartości blisko zera oznaczają zgodne przewidywania obu modeli.



Rysunek 11: Porównanie wag cech po standaryzacji. Wagi zostały znormalizowane, aby porównać względny wpływ cech w obu modelach.

2.7 Opis porównania: manual vs biblioteczny

Porównano dwa podejścia do regresji logistycznej:

- **Implementacja własna (manual)**:
  - uczenie metodą pełnobatchowego spadku gradientowego (gradient liczony z całego zbioru treningowego),
  - stała liczba epok (`EPOCHS`) oraz stały współczynnik uczenia (`LR`),
  - inicjalizacja wag: zera lub małe wartości losowe (w tym eksperymencie: `zeros`),
  - aktualizacja wag na podstawie gradientu funkcji log-loss.
- **Implementacja biblioteczna (`sklearn`)**:
  - gotowy model `LogisticRegression`, który używa własnej metody optymalizacji oraz kryterium stopu,
  - domyślna regularyzacja jest obecna, ale w eksperymencie została ograniczona przez ustawienie dużego parametru `C`,
  - parametry uczenia (np. `max_iter`) są obsługiwane wewnętrznie przez bibliotekę.

Porównanie wykonano na trzy sposoby:

- **Metryki jakości** na zbiorze treningowym i testowym (accuracy, precision, recall, F1, log-loss, ROC-AUC),
- **Porównanie prawdopodobieństw**:
  - wykres `proba_scatter.png`,
  - histogram różnic `proba_diff_hist.png`,
  - miary różnic: korelacja, MAE, RMSE.
- **Porównanie wag cech** po standaryzacji:
  - wagi cech znormalizowano (dla czytelności wykresu),
  - wyraz wolny (*bias*/intercept) podano osobno w logach konsoli.

2.8 Komentarz do wyników

Obie metody osiągnęły bardzo podobny poziom jakości klasyfikacji, co widać po takich samych metrykach typu *accuracy* na zbiorze testowym oraz identycznej macierzy pomyłek (tylko 2 błędy typu FP).

Jednocześnie widać różnicę w *log-loss*: metoda biblioteczna (`sklearn`) ma wyraźnie mniejszą stratę niż implementacja własna. To oznacza, że oba modele podejmują podobne decyzje klasowe (stąd podobne accuracy), ale **różnią się jakością oszacowania prawdopodobieństw**. Innymi słowy: model biblioteczny jest lepiej „skalibrowany" probabilistycznie (częściej daje pewniejsze i bardziej trafne prawdopodobieństwa).

Na wykresie `proba_scatter.png` punkty są blisko przekątnej, co oznacza wysoką zgodność prawdopodobieństw manual vs sklearn. Jednak histogram `proba_diff_hist.png` pokazuje, że różnice nadal występują (szczególnie dla niektórych próbek blisko granicy decyzyjnej), co tłumaczy różnice w log-loss.

Standaryzacja cech miała kluczowe znaczenie, ponieważ:

- ułatwia i stabilizuje uczenie metodą spadku gradientowego,
- sprawia, że wagi cech są bardziej porównywalne między sobą,
- poprawia działanie metod optymalizacji w bibliotece.

Wykres `wagi_porownanie.png` porównuje wagi cech po normalizacji (dla czytelności). Należy pamiętać, że *bias*/intercept jest osobnym parametrem i dlatego został wypisany osobno w konsoli.

3 Zakończenie

W ramach ćwiczenia zaimplementowano regresję logistyczną w dwóch wariantach: własnym (manual) oraz bibliotecznym z wykorzystaniem `scikit-learn`. Model został przetestowany na zbiorze Banknote Authentication, a cały proces obejmował pobranie danych, podział na zbiory treningowy i testowy, standaryzację cech, trening modeli, obliczenie metryk oraz wygenerowanie wykresów.

Oba podejścia osiągnęły bardzo podobne wyniki klasyfikacji (m.in. accuracy, precision, recall i F1), co potwierdza poprawność implementacji własnej. Różnice były lepiej widoczne w metryce *log-loss* oraz w porównaniu prawdopodobieństw: model biblioteczny uzyskał niższą wartość *log-loss*, co oznacza lepszą jakość i „pewność" przewidywanych prawdopodobieństw (lepszą kalibrację), mimo podobnych etykiet końcowych po zastosowaniu progu decyzyjnego.

Na podstawie eksperymentu można stwierdzić, że:

- implementacja własna działa poprawnie i daje wyniki porównywalne z metodą biblioteczną,
- standaryzacja cech jest ważna dla stabilnego uczenia i porównywalnych wag,
- metryki oparte na etykietach (np. accuracy) nie pokazują wszystkiego — do oceny jakości prawdopodobieństw potrzebny jest np. *log-loss* oraz analiza wykresów (ROC/PR, porównanie probabilistyczne).

Wyniki, wagi oraz wykresy zapisano do plików, co ułatwia weryfikację eksperymentu i jego powtarzanie dla innych ustawień hiperparametrów (np. liczby epok, współczynnika uczenia lub inicjalizacji wag).

Literatura