Politechnika Wrocławska
Wydział Informatyki i Telekomunikacji
Kierunek: Informatyczne Systemy Automatyki

Podstawy Sieci Neuronowych

Sprawozdanie końcowe z projektu

Temat:
Predykcja niewydolności serca z wykorzystaniem sztucznych sieci neuronowych (Wariant #1)

Autorzy:
Michał Łachut (280106)
Dawid Pajor (280067)

Data oddania:
22 stycznia 2026

Spis treści

1 Analiza problemu i wybór architektury **2**
1.1 Definicja problemu 2
1.2 Uzasadnienie wyboru architektury 2

2 Przygotowanie danych (Data Preparation) **2**
2.1 Analiza wstępna i braki danych 2
2.2 Korelacja cech 3
2.3 Preprocessing 4

3 Architektura sieci neuronowej **4**

4 Eksperymenty i dobór struktury **5**
4.1 Analiza eksperymentu 6

5 Wyniki modelu optymalnego **6**
5.1 Przebieg uczenia 6
5.2 Ewaluacja końcowa 7
5.3 Macierz pomyłek 7

6 Dyskusja i ocena realizacji **8**
6.1 Środowisko i dobrane narzędzia 8
6.2 Napotkane problemy i rozwiązania (Sukcesy i porażki) 9
6.3 Porównanie wyników z oczekiwaniami 9

7 Podsumowanie i wnioski **9**

1. Analiza problemu i wybór architektury

1.1. Definicja problemu

Celem projektu jest stworzenie modelu uczenia maszynowego zdolnego do przewidywania wystąpienia choroby serca na podstawie danych klinicznych pacjenta. Problem zdefiniowano jako zadanie **klasyfikacji binarnej**:

- Klasa 0: Pacjent zdrowy,
- Klasa 1: Pacjent z chorobą serca.

1.2. Uzasadnienie wyboru architektury

Ze względu na ustrukturyzowany, tabelaryczny charakter danych, wybrano architekturę **MLP (Multi-Layer Perceptron)**. Sieci gęste (Dense) są w tym przypadku bardziej adekwatne niż sieci konwolucyjne (CNN) czy rekurencyjne (RNN), które służą do analizy obrazu lub sekwencji.

2. Przygotowanie danych (Data Preparation)

2.1. Analiza wstępna i braki danych

Pierwszym krokiem była weryfikacja jakości danych. Przeprowadzono analizę brakujących wartości (Missing Values Analysis). Jak wykazano na Rysunku 1, zbiór danych jest kompletny i nie wymagał imputacji danych.

```
============================================================
MISSING VALUES ANALYSIS
============================================================
```

✅ No Missing Values found in the dataset!

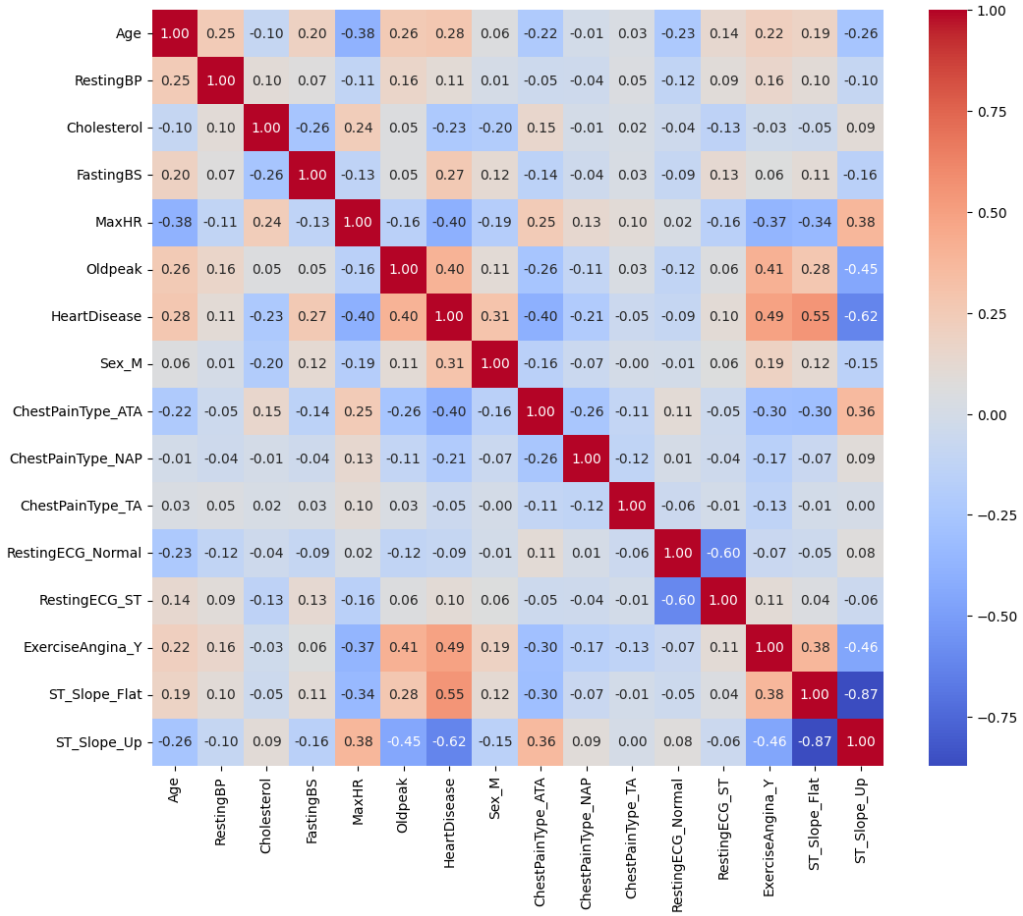
	Column	Missing Count	Missing %	Present Count	Present %
1	Age	0	0.0	918	100.0
2	RestingBP	0	0.0	918	100.0
3	Cholesterol	0	0.0	918	100.0
4	FastingBS	0	0.0	918	100.0
5	MaxHR	0	0.0	918	100.0
6	Oldpeak	0	0.0	918	100.0
7	HeartDisease	0	0.0	918	100.0
8	Sex_M	0	0.0	918	100.0
9	ChestPainType_ATA	0	0.0	918	100.0
10	ChestPainType_NAP	0	0.0	918	100.0
11	ChestPainType_TA	0	0.0	918	100.0
12	RestingECG_Normal	0	0.0	918	100.0
13	RestingECG_ST	0	0.0	918	100.0
14	ExerciseAngina_Y	0	0.0	918	100.0
15	ST_Slope_Flat	0	0.0	918	100.0
16	ST_Slope_Up	0	0.0	918	100.0

Rysunek 1: Raport brakujących danych - potwierdzenie kompletności zbioru

2.2. Korelacja cech

W celu zrozumienia zależności między cechami wygenerowano macierz korelacji (Heatmap). Analiza Rysunku 2 pozwala zauważyć, że:

- Najsilniejszą pozytywną korelację ze zmienną celu (`HeartDisease`) wykazuje cecha `ST_Slope_Flat` (0.55) oraz `Oldpeak` (0.40).
- Silną korelację ujemną wykazuje `ST_Slope_Up` (-0.62) oraz maksymalne tętno `MaxHR` (-0.40).



Rysunek 2: Macierz korelacji zmiennych numerycznych

2.3. Preprocessing

Zastosowano następujące techniki przygotowania danych:

1. **One-Hot Encoding:** Zmienne kategoryczne (np. `ChestPainType`) zamieniono na kolumny binarne (funkcja `get_dummies`), zwiększając wymiarowość wejścia.
2. **Standaryzacja:** Zastosowano `StandardScaler` dla cech numerycznych, sprowadzając je do wspólnej skali (średnia 0, odchylenie 1), co jest kluczowe dla zbieżności algorytmu *Adam*.
3. **Podział danych:**
   - Zbiór treningowy: 80% (z czego 20% wydzielono dynamicznie na walidację).
   - Zbiór testowy: 20% (184 próbki).

3. Architektura sieci neuronowej

Zaprojektowano i zaimplementowano sieć o następującej strukturze (zgodnie z Rysunkiem 3):

Warstwa	Liczba Neuronów	Aktywacja	Parametry
Dense (Input)	64	ReLU	1,024
Dropout	-	-	0
Dense (Hidden)	32	ReLU	2,080
Dense (Output)	1	Sigmoid	33
Suma			**3,137**

Tabela 1: Szczegóły modelu optymalnego

Layer (type)	Output Shape	Param #
dense_3 (Dense)	(None, 64)	1,024
dropout_1 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 32)	2,080
dense_5 (Dense)	(None, 1)	33

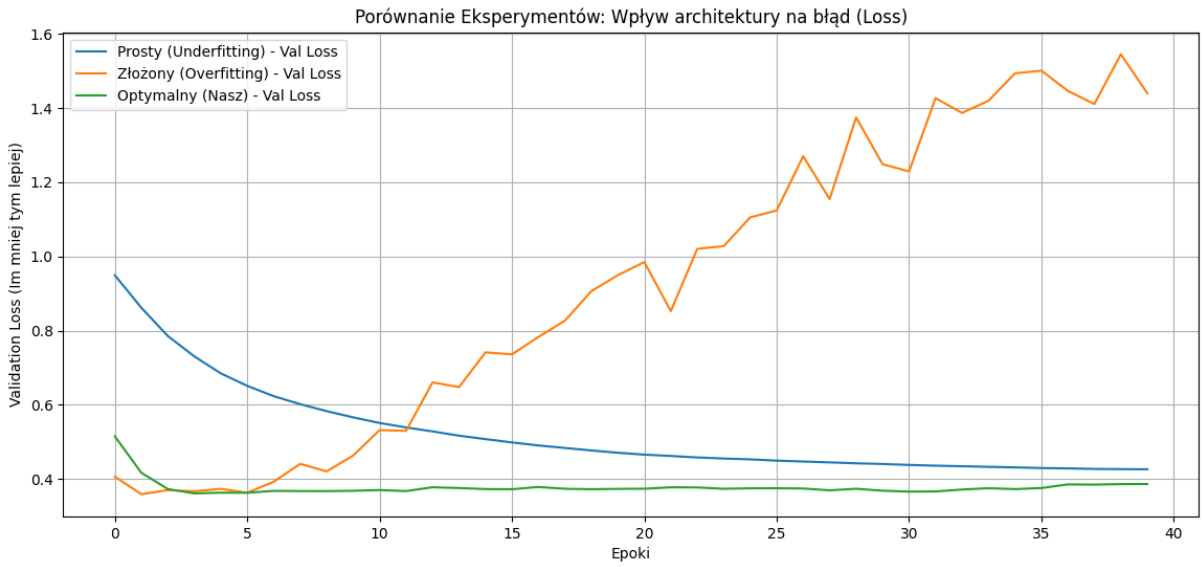
Total params: 3,137 (12.25 KB)
Trainable params: 3,137 (12.25 KB)
Non-trainable params: 0 (0.00 B)

Rysunek 3: Podsumowanie architektury modelu (Model Summary)

Zastosowano warstwę **Dropout (0.3)**, która losowo "wyłącza"30% neuronów w pierwszej warstwie ukrytej podczas każdej epoki treningu, co wymusza redundancję w sieci i zapobiega przeuczeniu.

4. Eksperymenty i dobór struktury

Przeprowadzono kluczowy eksperyment porównujący trzy architektury sieci, aby empirycznie dobrać najlepsze rozwiązanie. Wyniki przedstawiono na wykresie zbiorczym (Rys. 4).



Rysunek 4: Wpływ architektury na błąd walidacji (Validation Loss)

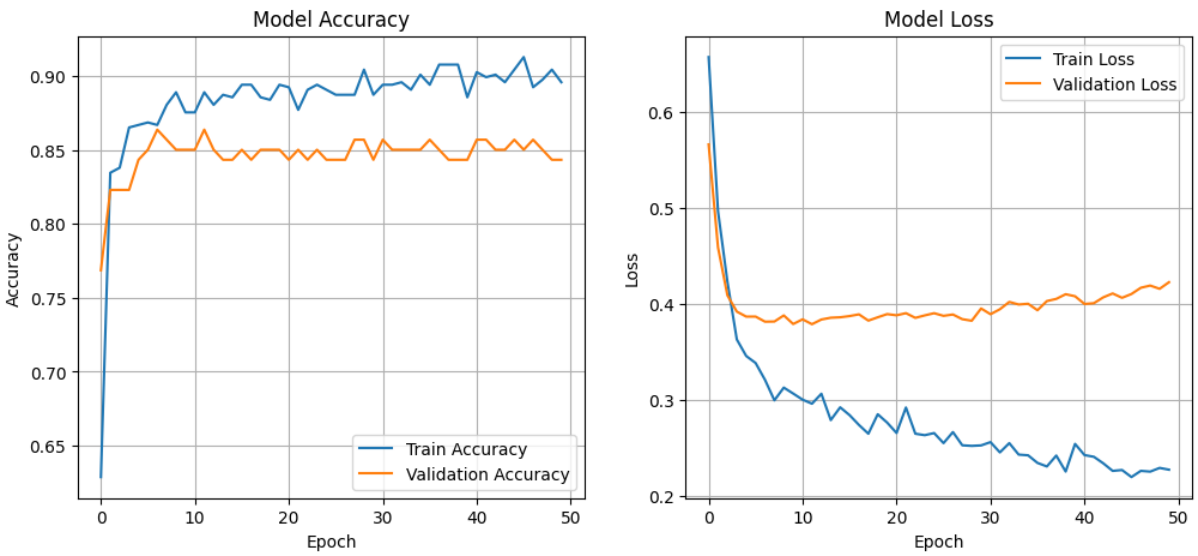
4.1. Analiza eksperymentu

- **Model Prosty (Niebieska linia):** Osiągnął wysoki błąd początkowy i wolno się uczył. Jest to przykład *underfittingu* (niedouczenia) – model był zbyt prosty, by wykryć zależności.
- **Model Złożony (Pomarańczowa linia):** Wykazał klasyczny przykład katastrofalnego *overfittingu* (przeuczenia). Do około 5. epoki błąd spadał, po czym zaczął gwałtownie rosnąć, osiągając wartość > 1.4. Model "nauczył się na pamięć"zbioru treningowego, tracąc zdolność generalizacji.
- **Model Optymalny (Zielona linia):** Utrzymał stabilny, niski poziom błędu walidacji (ok. 0.38 - 0.40) przez cały proces uczenia, nie wykazując tendencji wzrostowej.

5. Wyniki modelu optymalnego

5.1. Przebieg uczenia

Model trenowano przez 50 epok. Jak widać na Rysunku 5, dokładność (Accuracy) na zbiorze treningowym wzrosła do ok. 90%, a na walidacyjnym ustabilizowała się w okolicach 85%, co świadczy o zdrowym procesie uczenia.



Rysunek 5: Krzywe uczenia (Accuracy i Loss) dla modelu optymalnego

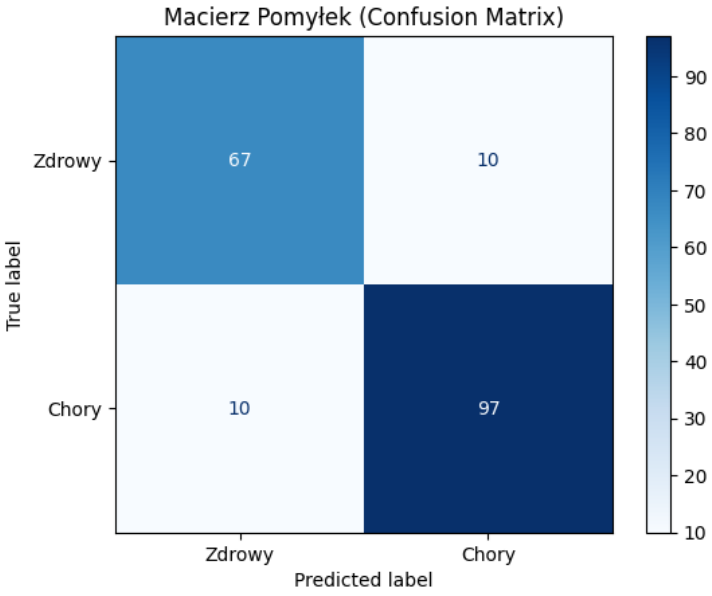
5.2. Ewaluacja końcowa

Ostateczną weryfikację przeprowadzono na niezależnym zbiorze testowym liczącym 184 pacjentów. Uzyskano następujące wyniki (Raport Klasyfikacji):

- **Accuracy (Dokładność):** 89%
- **Precision (Precyzja dla klasy Chory):** 0.90
- **Recall (Czułość dla klasy Chory):** 0.91

5.3. Macierz pomyłek

Szczegółowy rozkład błędów przedstawia Macierz Pomyłek (Rys. 6).



Rysunek 6: Macierz pomyłek na zbiorze testowym

Interpretacja wyników:

- **True Positive (97):** Poprawnie zdiagnozowano 97 osób chorych.
- **True Negative (67):** Poprawnie zdiagnozowano 67 osób zdrowych.
- **False Negative (10):** Model pominął 10 osób chorych (błąd II rodzaju - najbardziej krytyczny w medycynie).
- **False Positive (10):** Model błędnie uznał 10 osób zdrowych za chore.

6. Dyskusja i ocena realizacji

6.1. Środowisko i dobrane narzędzia

Zgodnie z wymaganiami projektowymi, implementację zrealizowano w języku **Python 3.10**. Wykorzystano następujący stos technologiczny:

- **TensorFlow / Keras:** Budowa i trening sieci neuronowej (API Sequential).
- **Pandas & NumPy:** Manipulacja danymi tabelarycznymi i obliczenia wektorowe.
- **Scikit-Learn:** Preprocessing (skalowanie, encoding) oraz metryki ewaluacyjne.
- **Matplotlib & Seaborn:** Wizualizacja danych i wyników (wykresy strat, macierze).

6.2. Napotkane problemy i rozwiązania (Sukcesy i porażki)

Podczas realizacji projektu zidentyfikowano kilka kluczowych wyzwań:

1. **Problem przeuczenia (Overfitting):** Wstępne modele o dużej pojemności (ponad 256 neuronów) osiągały niemal 100% dokładności na zbiorze treningowym, ale drastycznie traciły skuteczność na zbiorze walidacyjnym (Loss > 1.5).

2. **Rozwiązanie:** Problem rozwiązano poprzez redukcję liczby neuronów oraz implementację warstwy `Dropout` (0.3), co ustabilizowało proces uczenia.

3. **Niezbalansowanie klas:** Zauważono lekką przewagę klasy chorobowej w danych, co mogło wpłynąć na metrykę Accuracy. Analiza macierzy pomyłek potwierdziła jednak, że model zachowuje wysoki *Recall* (0.91), co uznajemy za sukces w kontekście medycznym.

6.3. Porównanie wyników z oczekiwaniami

Wstępne założenia projektowe zakładały osiągnięcie dokładności na poziomie min. 85% (typowy próg dla prostych modeli w literaturze dla tego zbioru). Ostateczny wynik modelu optymalnego (**89%**) przewyższył oczekiwania. Uzyskano model, który nie tylko spełnia założenia dokładności, ale cechuje się niskim odsetkiem wyników fałszywie ujemnych (False Negatives), co było priorytetem projektowym.

7. Podsumowanie i wnioski

Projekt zakończył się sukcesem. Osiągnięto wysoką dokładność **89%** na zbiorze testowym.

Najważniejszym wnioskiem z procesu projektowego jest **rola regularyzacji**. Eksperymenty dobitnie pokazały, że samo zwiększanie liczby neuronów (Model Złożony) prowadzi do drastycznego pogorszenia wyników na nowych danych. Dopiero zastosowanie zbalansowanej architektury z warstwą `Dropout` pozwoliło na stworzenie modelu, który nie tylko pamięta dane treningowe, ale potrafi skutecznie diagnozować nowych pacjentów.

Wysoki współczynnik *Recall* (0.91) oznacza, że model jest bezpieczny w zastosowaniach wstępnej diagnozy, minimalizując ryzyko nie wykrycia choroby.