Politechnika
Śląska

PRACA MAGISTERSKA

Zastosowanie metod sztucznej inteligencji do detekcji arytmii na podstawie sygnałów PPG

Jakub KULA
Nr albumu: 296849

Kierunek: Informatyka
Specjalność: Internet i technologie sieciowe

PROWADZĄCY PRACĘ
dr hab. inż. Pander Tomasz, prof. PŚ
KATEDRA Cybernetyki, Nanotechnologii i Przetwarzania Danych
Wydział Automatyki, Elektroniki i Informatyki

Gliwice 2025

Tytuł pracy

Zastosowanie metod sztucznej inteligencji do detekcji arytmii na podstawie sygnałów PPG

Streszczenie

(Streszczenie pracy - odpowiednie pole w systemie APD powinno zawierać kopię tego streszczenia.)

Słowa kluczowe

(2-5 slow (fraz) kluczowych, oddzielonych przecinkami)

Thesis title

Application of artificial intelligence methods for arrhythmia detection based on PPG signal

Abstract

(Thesis abstract - to be copied into an appropriate field during an electronic submission - in English.)

Key words

(2-5 keywords, separated by commas)

Spis treści

Rozdział 1

Wstęp

1.1 Cel i zakres pracy

1.2 Aktualny stan wiedzy

1.3 Charakterystyka rozdziałów

Rozdział 2

Charakterystyka arytmii serca i sygnału PPG

2.1 Klasyfikacja i mechanizmy arytmii serca

2.1.1 Arytmie nadkomorowe

2.1.2 Arytmie komorowe

2.1.3 Migotanie przedsionków

2.2 Fotopletyzmografia - zasada działania i zastosowania

Rozdział 3

Metody uczenia maszynowego

3.1 Klasyczne metody

Metoda najbliższych sąsiadów (KNN)

Klasyfikator opierający swoje predykcje na analizie k najbliższych sąsiadów ze zbioru treningowego. Metoda ta wykorzystuje metryki odległości do wyznaczenia próbek najbardziej zbliżonych do klasyfikowanego punktu, a następnie przypisuje mu etykietę tej klasy, która najczęściej występuje wśród wybranych k sąsiadów.

Najczęściej stosowaną metryką odległości jest odległość euklidesowa, zdefiniowana wzorem:

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^2 \right)^{\frac{1}{2}} \tag{3.1}$$

gdzie $x = (x_1, x_2, \ldots, x_n)$ oraz $y = (y_1, y_2, \ldots, y_n)$ to dwa punkty w n-wymiarowej przestrzeni cech. Generalizacją odległości Euklidesowej jest odległości Minkowskiego zdefiniowana jako:

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^q \right)^{\frac{1}{q}} \tag{3.2}$$

gdzie q>0 [1]. Można zauważyc, że dla q=2 otrzymuje odległość Euklidesową, a dla q=1 odległość Manhatan. Ze względu na to, że algorytm KNN opiera się na obliczaniu odległości między próbkami, istotne jest zachowanie jednolitej skali cech. W przypadku, gdy poszczególne cechy mają różne zakresy wartości, cecha o największej rozpiętości może zdominować obliczenia odległości, a tym samym nieproporcjonalnie wpłynąć na wynik predykcji.

W przypadku tego klasyfikatora kluczowe jest odpowiednie dobranie liczby sąsiadów. Zbyt mała wartość parametru k może prowadzić do przeuczenia, w którym model nadmiernie dopasowuje się do danych treningowych i traci zdolność generalizacji. Z kolei zbyt duża liczba sąsiadów skutkuje zjawiskiem niedouczenia, w którym model nie jest w stanie uchwycić istotnych zależności w danych[2].

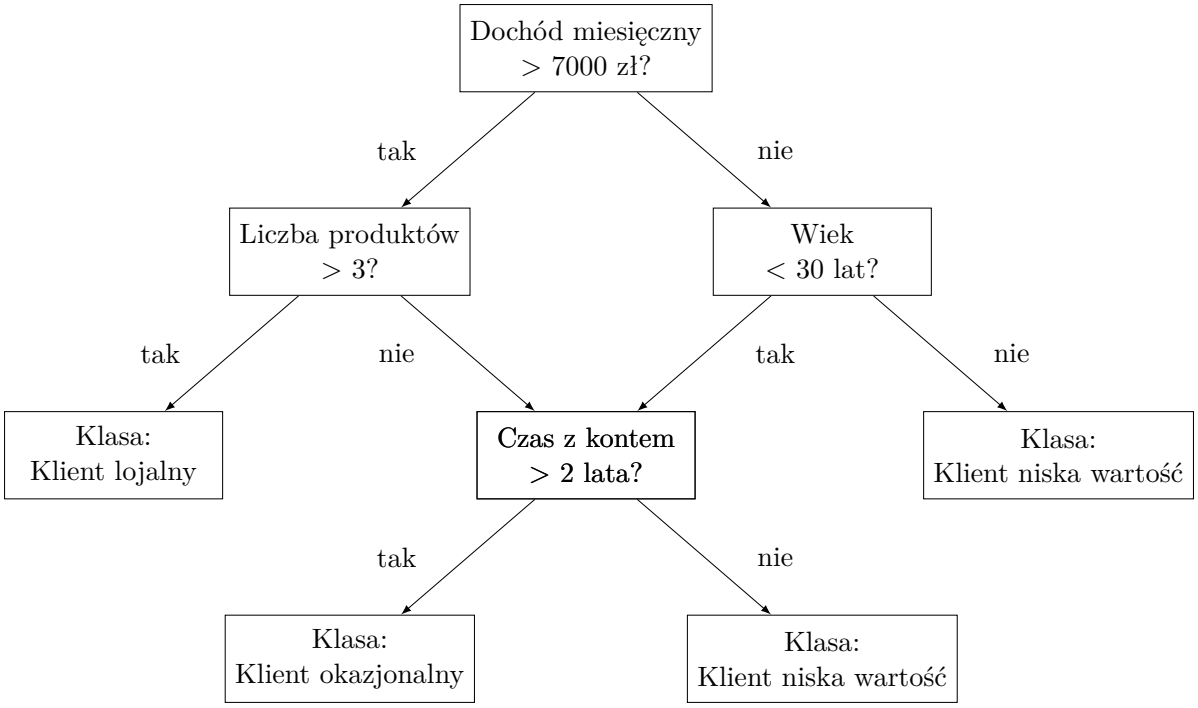
Drzewo decyzyjne

Model drzewa decyzyjnego opiera się na zagnieżdżonych regułach typu „jeżeli-wtedy", które dzielą przestrzeń cech na coraz mniejsze podobszary. Struktura drzewa składa się z:

- korzenia - punktu początkowego, od którego rozpoczyna się proces podejmowania decyzji,
- węzłów decyzyjnyc - zawierających testy logiczne dzielące dane na podzbiory na podstawie wartości cech,
- gałęzi - reprezentujących możliwe wyniki testów prowadzące do kolejnych węzłów,
- liści - końcowych węzłów, w których przypisywana jest wartość przewidywanej klasy.

Proces uczenia drzewa decyzyjnego polega na iteracyjnym wyborze cech oraz odpowiadających im wartości progowych, które najlepiej rozdzielają dane.

Rysunek 3.1: Przykład drzewa decyzyjnego



Drzewa decyzyjne, dzięki prostocie procesu uczenia i działania, cechują się wysoką interpretowalnością. Mimo że drzewo może zawierać wiele węzłów decyzyjnych, użytkownik jest w stanie z łatwością prześledzić krok po kroku ciąg decyzji prowadzących do końcowej predykcji. Największą wadą drzew decyzyjnych jest ich skłonność do przeuczania, co oznacza, że model traci zdolność do uogólniania i zbyt dokładnie dopasowuje się do danych ze zbioru uczącego. Zjawisko to szczególnie często występuje w przypadku bardzo rozbudowanych drzew.

Podczas procesu budowy drzewa decyzyjnego możliwe jest pełne kontrolowanie jego struktury poprzez określenie między innymi ograniczeń związanych z maksymalną głębokością drzewa oraz minimalną liczbą próbek wymaganą do utworzenia węzła decyzyjnego. Pozwala to na regulację złożoności modelu i ograniczenie ryzyka przeuczenia.

Las losowy

Klasyfikator oparty na zbiorze drzew decyzyjnych, których predykcje są agregowane w celu podjęcia decyzji końcowej, zwykle przez głosowanie większościowe. Model ten wykorzystuje dwa kluczowe mechanizmy w procesie uczenia:

- Bootstrap aggregation - dla każdego drzewa decyzyjnego generowany jest losowy podzbiór próbek (z powtórzeniami) ze zbioru treningowego.
- Losowy wybór cech - na każdym etapie podziału drzewa rozważana jest tylko losowo wybrana podgrupa cech.

Zastosowanie powyższych technik sprawia, że lasy losowe cechują się większą odpornością na przeuczenie w porównaniu do pojedynczych drzew decyzyjnych. Dodatkową zaletą jest możliwość oszacowania niepewności predykcji na podstawie rozrzutu odpowiedzi poszczególnych drzew. Wadą tej metody jest natomiast ograniczona interpretowalność oraz zwiększone wymagania obliczeniowe i pamięciowe względem pojedynczych modeli drzewiastych.

Naiwny klasyfikator Bayesa

Klasyfikator oparty na teorii Bayesa, który dokonuje klasyfikacji poprzez ocenę prawdopodobieństwa przynależności wektora cech do poszczególnych klas. Model ten nie analizuje bezpośrednich zależności między cechami a klasą, lecz na podstawie danych treningowych estymuje rozkłady prawdopodobieństwa cech w każdej klasie. Przy klasyfikacji nowych próbek oblicza się prawdopodobieństwo przynależności wektora cech do każdej klasy, a przypisanie następuje do klasy o największym prawdopodobieństwie. Regułę Bayesa można zapisać wzorem:

$$P(C_k \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid C_k) \cdot P(C_k)}{P(\mathbf{x})}, \tag{3.3}$$

gdzie:

- $\mathbf{x}$ - wektor cech opisujących próbkę,
- C_k - klasa, do której próbka może należeć,
- $P(C_k \mid \mathbf{x})$ - prawdopodobieństwo, że próbka o cechach $\mathbf{x}$ należy do klasy C_k,

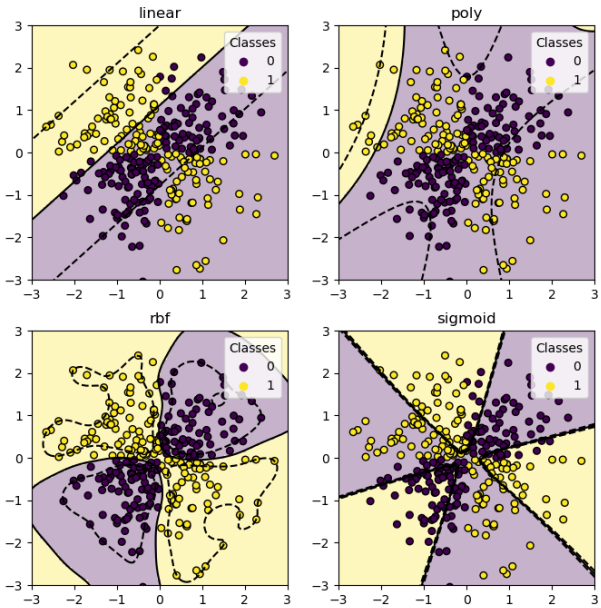
- $P(\mathbf{x} \mid C_k)$ - prawdopodobieństwo zaobserwowania cech $\mathbf{x}$, pod warunkiem, że próbka należy do klasy C_k,
- $P(C_k)$ - prawdopodobieństwo wystąpienia klasy C_k,
- $P(\mathbf{x})$ - całkowite prawdopodobieństwo wystąpienia cech $\mathbf{x}$.

Klasyfikator nazywany jest „naiwnym” ze względu na przyjęte założenie o niezależności cech, które znacząco uproszczają obliczenia statystyczne, jednak jest jest ono zazwyczaj nierealistyczne.

Maszyna wektorów nośnych (SVM)

Maszyna wektorów nośnych to klasyfikator, którego celem jest wyznaczenie optymalnej granicy decyzyjnej oddzielającej dane należące do różnych klas. W sytuacji, gdy dane są liniowo separowalne, istnieje nieskończenie wiele możliwych hiperpłaszczyzn rozdzielających. SVM wybiera tę, która maksymalizuje tzw. margines, czyli minimalną odległość między granicą decyzyjną a najbliższymi punktami obu klas. Takie podejście zapewnia dobrą generalizację klasyfikatora.

W przypadku, gdy dane nie są liniowo separowalne, stosuje się funkcję jądra, która przekształca przestrzeń cech do przestrzeni o wyższym wymiarze, w której możliwe jest liniowe rozdzielenie danych. Wykorzystanie funkcji jądra pozwala SVM na modelowanie nieliniowych granic decyzyjnych bez jawnego przekształcania danych.



Rysunek 3.2: Przykłady granic decyzyjnych w przestrzeni cech na zbiorze XOR[3]

LS-SVM

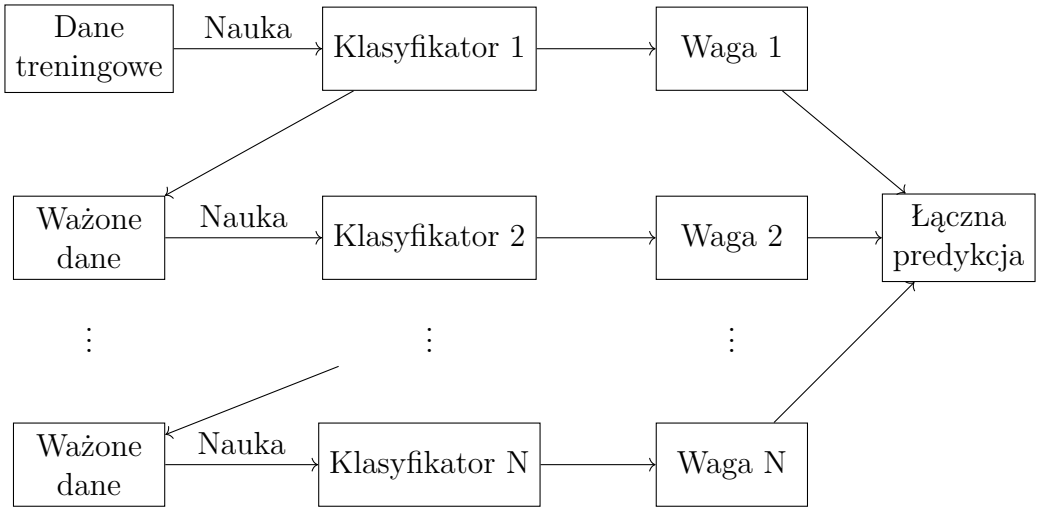
LS-SVM stanowi modyfikację klasycznego SVM, w której problem optymalizacji kwadratowej zastąpiono rozwiązaniem układu równań liniowych [4]. Kluczową różnicą w stosunku do standardowej wersji jest zastosowanie funkcji kosztu w postaci sumy kwadratów błędów oraz wykorzystanie równości jako ograniczeń optymalizacji, zamiast nierówności.

Podejście to znacząco upraszcza obliczenia, czyniąc algorytm bardziej efektywnym pod względem czasu uczenia, szczególnie dla dużych zbiorów danych. Jednakże, uproszczona forma LS-SVM charakteryzuje się większą wrażliwością na obserwacje odstające, co może wpływać na stabilność klasyfikatora w obecności szumu lub nietypowych próbek.

Modele typu Boosting

Koncepcja polegajaca na połaczeniu wielu modeli posiadających małą skuteczność w celu stworzenia jednego modelu o dużej dokładności[5, 6]. W ogólnym przypadku schemat działania boostingu polega na sekwencyjnym uczeniu modeli, gdzie każdy kolejny model otrzymuje informację o błędach popełnionych przez poprzednie i na tej podstawie stara się poprawić jakość predykcji całego systemu.

Rysunek 3.3: Schemat przetwarznia wstępnego



Największą wadą boostingu jest wysokie ryzyko przeuczenia, wynikające z jego mechanizmu uczenia się na błędach. Takie podejście sprzyja nadmiernemu skupianiu się na przykładach trudnych do sklasyfikowania, co może prowadzić do zbyt mocnego dopasowania modelu do danych treningowych i utraty zdolności generalizacji.

Zdecydowano się na wybór trzech modeli typu boosting, opartych na technice *Gradient Boosting*, czyli metodzie polegającej na sekwencyjnym minimalizowaniu funkcji straty poprzednich modeli przy użyciu algorytmu spadku gradientowego. Wybrane modele:

- **XGBoost** - wykorzystuje modele drzew decyzyjnych, w każdej iteracji dodając kolejne drzewo, które uczy się na podstawie błędów poprzednich. W celu minimalizacji

funkcji straty stosowane jest rozwinięcie Taylora drugiego rzędu, a także regularyzacja L1 i L2 w celu zapobiega przeuczeniu.

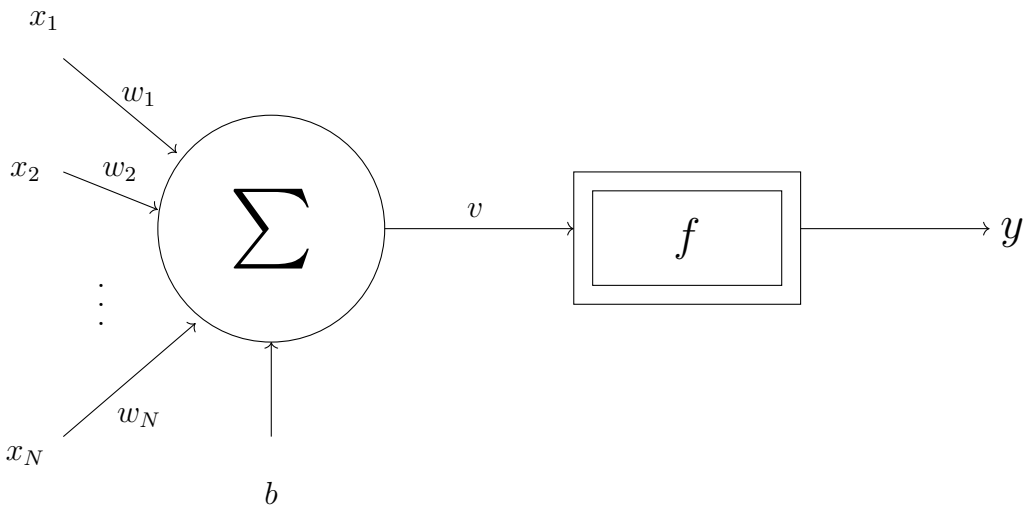
- **CatBoost** - podobnie jak XGBoost opiera się na drzewach decyzyjnych i metodzie gradient boosting, jednak dodatkowo optymalizuje przetwarzanie cech kategorycznych i zmniejsza ryzyko przeuczania wprowadzajac mechanizm uporządkowane wzmocnienie

- **LightBoost** - również bazuje na drzewach decyzyjnych, jednak w procesie uczenia nie tworzy całych nowych drzew, lecz rozwija istniejące liście tych drzew, które posiadają najwyższą wartość funkcji straty.

3.2 Sieci neuronowe

Neuron, będący podstawowym elementem sztucznej sieci neuronowej, jest inspirowany budową i działaniem neuronów biologicznych występujących w ośrodkowym układzie nerwowym. Komórki nerwowe pełnią funkcję przekaźników impulsów elektrycznych - odbierają, przetwarzają i przekazują sygnały.

Neuron matematyczny jest układem typu MISO (Multiple Input, Single Output), który sumuje otrzymane sygnały wejściowe po uprzednim przemnożeniu ich przez odpowiadające im wagi. Do takiej zsumowanej wartości dodawany jest składnik stały, zwany biasem. Ostatecznie wynik ten podawany jest na wejście funkcji aktywacji, której celem jest wprowadzenie nieliniowości do modelu[7].



Rysunek 3.4: Model neuronu

Sygnał wyjściowy z pojedyńczego neuronu można wyznaczyć za pomocą wzoru:

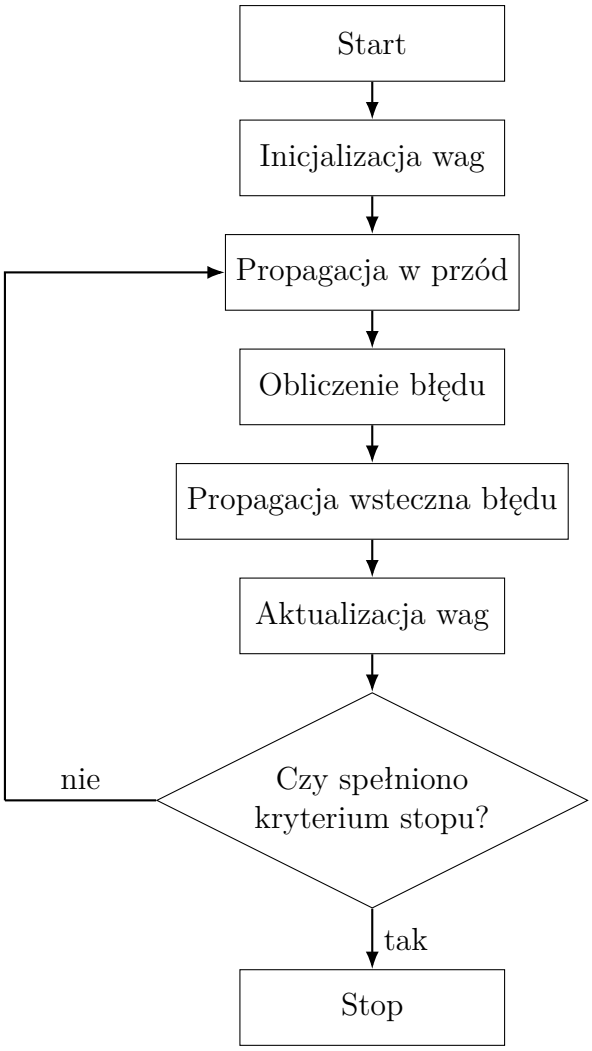
$$y = f\left(\sum_{i=1}^{\mathrm{N}} w_i x_i + \mathrm{b}\right), \tag{3.4}$$

gdzie:

- y - sygnał wyjściowy,
- $f(\cdot)$ - funkcja aktywacji,
- N - liczba wejść,
- x_i - sygnał wejściowy dla i-tego wejścia,
- w_i - waga przypisana do i-tego sygnału wejściowego,
- b - składnik stały,

Organizując neurony warstwowo, tworzymy sieć neuronową. Sieć posiada warstwę wejściową, która przyjmuje dane - liczba neuronów w tej warstwie odpowiada liczbie wymiarów danych wejściowych. Kolejne warstwy nazywane są warstwami ukrytymi, ponieważ nie mamy bezpośredniego dostępu do ich wyjść. Ostatnia warstwa to warstwa wyjściowa, w której liczba neuronów odpowiada liczbie klas w problemie klasyfikacji lub liczbie wartości przewidywanych w problemie regresji.

Nauka sieci neuronowych polega na iteracyjnej aktualizacji wag neuronów w celu minimalizacji funkcji błędu. Proces ten rozpoczyna się od przypisania początkowych wartości wag, które zazwyczaj wybierane są quasi-losowo - jako niewielkie liczby o rozkładzie normalnym. Taki wybór ma na celu zapewnienie różnorodności w początkowym uczeniu się neuronów. Istnieją jednak również bardziej zaawansowane metody inicjalizacji, które uwzględniają właściwości funkcji aktywacji w poszczególnych warstwach sieci.
W każdym kroku procesu uczenia dane ze zbioru treningowego są przetwarzane przez sieć neuronową za pomocą algorytmu propagacji w przód. Na podstawie uzyskanych wyników obliczana jest wartość błędu przy użyciu funkcji straty, która określa sposób kwantyfikacji rozbieżności między przewidywaniami modelu a rzeczywistymi wartościami. Następnie wykorzystywany jest optymalizator, który - na podstawie gradientów obliczanych przy pomocy algorytmu propagacji wstecz wyznacza nowe wartości wag neuronów. Ostatecznym etapem jest aktualizacja wag w całej sieci. Proces ten jest powtarzany przez ustaloną liczbę epok lub do momentu spełnienia zdefiniowanego warunku stopu.



Rysunek 3.5: Schemat procesu uczenia sieci neuronowej

Wraz z rozwojem sieci neuronowych powstały bardziej złożone sposoby przetwarzania danych. W badaniach wykorzystano następujące typy warstw sieci neuronowych:

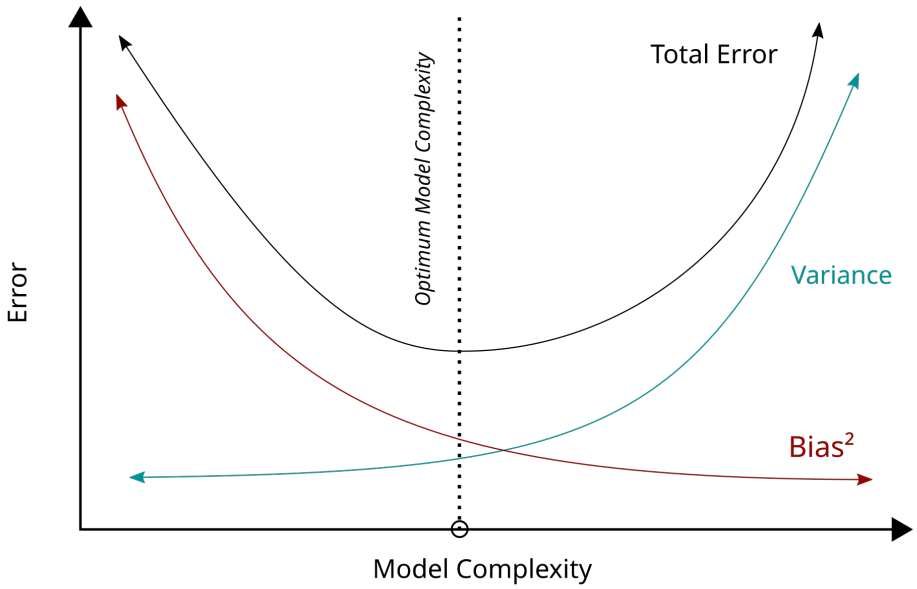
- Warstwy gęste - klasyczne warstwy w pełni połączone, w których każdy neuron jest połączony z każdym neuronem warstwy kolejnej. Są najczęściej stosowane w prostych sieciach oraz jako warstwy końcowe w bardziej złożonych architekturach.
- Warstwy konwolucyjne - przetwarzają dane wejściowe za pomocą filtrów konwolucyjnych, które działają lokalnie w obrębie małych fragmentów danych wejściowych, zgodnie z przesuwającym się jądrem. Podczas procesu uczenia, wartości wag jądra są aktualizowane, co pozwala na automatyczne wykrywanie cech. Często stosowanym mechanizmem pomocniczym jest warstwa poolingowa, której zadaniem jest redukcja wymiarowości danych oraz zwiększenie odporności na przesunięcia i zakłócenia. Zastosowanie warstw konwolucyjnych umożliwiło rozwój głębokiego uczenia, eliminując konieczność ręcznego projektowania wektora cech przez inżynierów.
- Warswy LSTM - specjalny typ rekurencyjnych warstw neuronowych zaprojektowany

w celu ograniczenia problemu zanikających gradientów, czyli zjawiska, w którym wartości gradientów w początkowych warstwach stają się bardzo małe, co prowadzi do niestabilności i spowolnienia procesu uczenia. Warstwy LSTM wprowadzają pamięć krótkotrwałą, która jest kontrolowana przez trzy rodzaje bramek: wejściową, wyjściową oraz zapominającą. Bramki te regulują przepływ informacji do, z oraz w obrębie komórki pamięci, co umożliwia skuteczne przechowywanie i aktualizowanie informacji w czasie. Dzięki temu model jest w stanie uczyć się zależności sekwencyjnych na dłuższych dystansach czasowych. Choć mechanizm LSTM znacząco ogranicza problem zanikania gradientów, nadal może wystąpić zjawisko eksplodujących gradientów. Warstwy te są szczególnie przydatne podczas przewidywania szeregów czasowych.

Tworząc sieć neuronową, inżynier musi dostosować hiperparametry, które determinują zarówno strukturę sieci, jak i sposób jej uczenia. Hiperparametry dzielą się na dwie główne kategorie:

1. Hiperparametry związane z architekturą sieci:
   (a) Liczba warstw sieci,
   (b) Liczba neuronów w poszczególnych warstwach,
   (c) Funkcje aktywacji zastosowane w warstwach,
   (d) Typy warstw.
2. Hiperparametry związane z procesem uczenia:
   (a) Liczba epok - warunek stopu związany z ilością prztworzeń zbiorzu uczącego,
   (b) Funkcja strat - miara błędu wykorzystywana przez optymalizator do aktualizacji wag,
   (c) Optymalizator - algorytm obliczający gradienty, wyznacza nowe wagi podczas procesu nauki.
   (d) Wielkość wsadu - liczba próbek przetwarzanych jednocześnie w jednej iteracji,
   (e) Tempo uczenia - parametr regulujący krok aktualizacji wag.

Proces wyboru hiperparametrów jest kluczowy dla uzyskania modelu o wysokiej skuteczności. Wraz ze zwiększaniem liczby warstw sieci oraz liczby neuronów, rośnie zdolność modelu do reprezentowania złożonych zależności, co prowadzi do zmniejszenia błędu biasu. Jednak bardziej złożone modele posiadają większy rozrzut wyników co skutkuje wzrostem wariancji



Rysunek 3.6: Zależność pomiędzy wariancją a biasem w funkcji złożoności modelu[8]

Wykres przedstawia 3.6 kompromis obciążeniowo-wariacyjny (bias-variance tradeoff) w funkcji złożoności modelu. Przy niskiej złożoności model nie jest w stanie dobrze odwzorować zależności w zbiorze uczącym - mamy wówczas do czynienia z niedouczeniem oraz wysokim błędem obciążenia. Wraz ze wzrostem złożoności modelu błąd obciążenia maleje, a jakość predykcji rośnie. Jednak dalsze zwiększanie złożoności prowadzi do wzrostu błędu wariancji, ponieważ model zaczyna uczyć się również szumu obecnego w danych, co skutkuje przeuczeniem i utratą zdolności generalizacji. W takiej sytuacji mimo niskiego błędu obciążenia, wysoka wariancja obniża skuteczność modelu. Optymalny punkt znajduje się tam, gdzie suma błędu obciążenia i wariancji jest najmniejsza - w tym miejscu model osiąga najlepszą równowagę między niedouczeniem a przeuczeniem.

3.3 Miary jakościowe klasyfikatorów

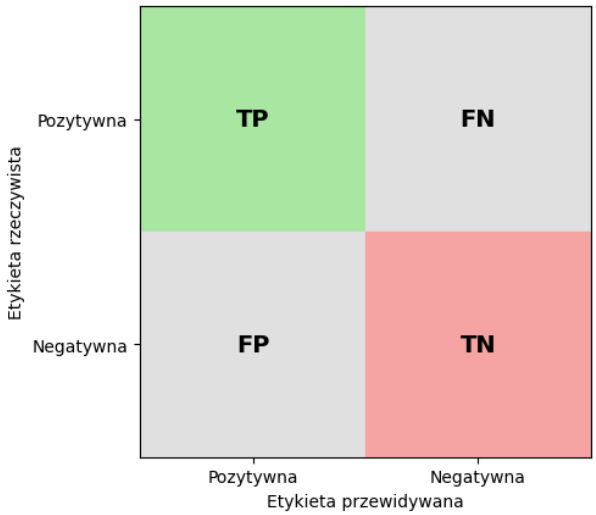
W celu efektywnego porównywania modeli predykcyjnych konieczne jest zastosowanie odpowiednich miar oceny jakości, które umożliwiają liczbową ocenę skuteczności działania danego modelu w określonym kontekście. Dobór metryk powinien być dostosowany do charakteru problemu - inne miary stosuje się w przypadku klasyfikacji, inne przy lokalizacji obiektów, a jeszcze inne przy zadaniach regresyjnych.

Aby właściwie zrozumieć sposób działania metryk stosowanych w klasyfikacji binarnej, należy w pierwszej kolejności zdefiniować cztery możliwe wyniki predykcji:

- **True Positive (TP)** - poprawna klasyfikacja przypadku z arytmią serca jako przypadek chorobowy.

- **False Positive (FP)** - błędna klasyfikacja przypadku zdrowego jako przypadek z arytmią. Nazywana również błędem pierwszego rodzaju.
- **True Negative (TN)** - poprawna klasyfikacja przypadku zdrowego jako niechorobowego.
- **False Negative (FN)** - błędna klasyfikacja przypadku z arytmią jako przypadek zdrowy. Nazywana również błędem drugiego rodzaju.

Dzięki wyznaczeniu 4 przypadów, jestesmy w stanie w graficzny sposób przedstawić wyniki klasyfikacji w postaci macierzy pomyłek.



Rysunek 3.7: Struktura macierzy pomyłek dla klasyfikacji binarnej

Wykorzystując wymieniane przypadki, jesteśmy w stanie wyznaczyć metryki jakościowe, które pozwalają na ocenę skuteczności klasyfikatora.

- **Dokładność** - metryka określająca jaki procent przypadków został poprawnie sklasyfikowany. Jest to metryka która jest mocno zalezna od rozkładu klasy. W sytuacji gdy jedna klasas jest znacznie liczniejsza, może ona mocno zakrzywić wyniki. W przypadku klasyfykacji binarnej możemy ją wyznaczyć jako:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{3.5}$$

- **Precyzja** - miara jakościowa określającą jaki procent przypadków sklasyfikowanych jako osoby z chorobą rzeczywiscie posiadaja arytmię. Prezycja nie zwraca uwagi na przypadki zdrowę, zwraca uwagę jedynie na przypadki z arytmią. Omawianą miarę możemy obliczyć jako:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.6}$$

- **Swoistość** - określa umiejętność detekcji zdrowych pacejntów bez arytmi. Matematycznie możemy zapisać to jako:

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{3.7}$$

- **Czułość** - prawdopodobieństwo ze klasyfikacja będzie poprawna pod warunkiem że dana próbka pochodzi od osoby chorej. Można ją obliczyć ze wzoru:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.8}$$

- **F-miara** - metryka która łaczy prezycję i czułość. Wyznacza się jako średnią harmoniczną owych metryk. Ze względu na to, jest ona bardzo wrażliwa na przypadki gdy jedna z metryk jest bardzo niska. Jednak jest ona odporna na nierówny rozkład klas. Wartość F-miary możemy wyznaczyć jako:

$$\text{F}_1 = \frac{1}{2}\left(\frac{1}{\text{precision}} + \frac{1}{\text{recall}}\right) = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \tag{3.9}$$

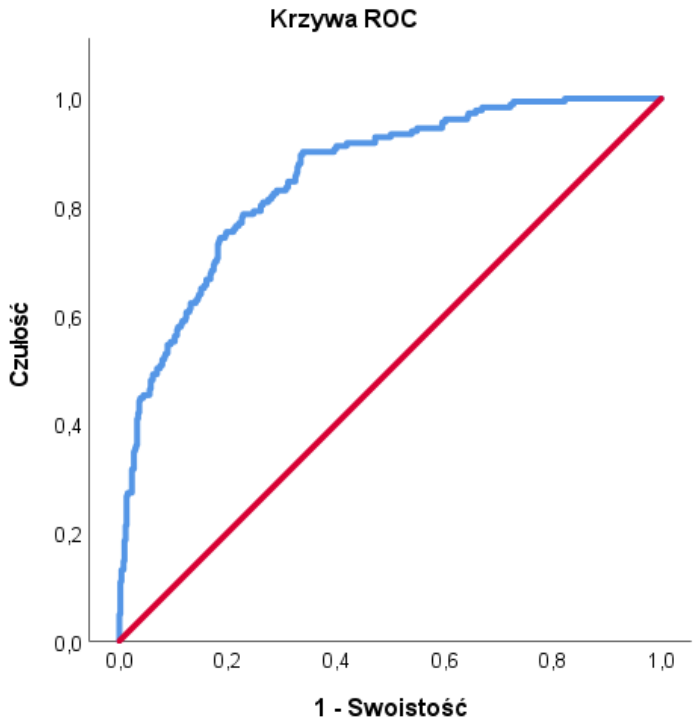
- **współczynnika korelacji Matthews'a (MCC)** - miara korelacji binarnej wprowadzona do uczenia maszynowego przez Briana W. Matthewsa [9]. Zakres tego współczynnika wynosi od -1 do 1, gdzie 1 oznacza perfekcyjną klasyfikację, 0 klasyfikację losową, a -1 - całkowicie odwrotną predykcję. MCC jest wskaźnikiem jakościowym odpornym na niezbalansowanie klas i można go obliczyć według wzoru:

$$\text{MCC} = \frac{\text{TP} \cdot \text{TN} - \text{FP} \cdot \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}} \tag{3.10}$$

- **Krzywa ROC** [10, 11, 12] - graficzne przedstawienie wydajności modelu klasyfikacyjnego w zadaniu klasyfikacji binarnej, w zależności od wartości progu decyzyjnego. Wykres ten ukazuje zależność między czułością a 1 - swoistością.

  Krzywa ROC służy do oceny oraz porównywania jakości klasyfikatorów, a także do wyboru odpowiedniego progu decyzyjnego, od którego próbki są przypisywane do klasy pozytywnej.

  W najlepszym przypadku krzywa ROC przechodzi przez punkt (0,0) i (0,1), co oznacza, że klasyfikator osiąga 100% czułości (brak fałszywie negatywnych wyników) oraz 100% swoistości (brak fałszywie pozytywnych wyników). Taki model zawsze dokonuje poprawnych predykcji i uznawany jest za idealny.



Rysunek 3.8: Wykres krzywej ROC [13]

Wraz z pogorszeniem jakości klasyfikatora krzywa ROC zbliża się do przekątnej łączącej punkty (0,0) i (1,1). Taka linia reprezentuje model losowy - jego predykcje są statystycznie równoważne zgadywaniu, niezależnie od przyjętego progu decyzyjnego. W takiej sytuacji model uznaje się za nieskuteczny.

W skrajnie niekorzystnym przypadku, gdy krzywa ROC znajduje się poniżej tej przekątnej, oznacza to, że klasyfikator częściej się myli, niż przewiduje poprawnie. Paradoksalnie, odwrócenie jego predykcj poprawia skuteczność klasyfikacji.

Wraz ze zmniejszaniem wartości progu decyzyjnego rośnie czułość modelu, co oznacza spadek liczby błędów drugiego rodzaju. Jednocześnie jednak rośnie liczba błędów pierwszego rodzaju - fałszywie pozytywnych wyników. Z kolei zwiększanie progu skutkuje sytuacją odwrotną: poprawia się swoistość, ale kosztem spadku czułości.

Taką sytuacje określa się jako kompromis między czułością a swoistością. Optymalny próg zależy od charakterystyki problemu oraz kosztów błędnych decyzji. W zastosowaniach medycznych często preferuje się modele o wysokiej czułości, aby ograniczyć liczbę niewykrytych przypadków choroby, nawet kosztem większej liczby fałszywych alarmów.

Wykorzystując krzywą ROC, możliwe jest oblicznie pod nią. Taka matryka nazywa się **AUC**. Jest ona zależna i od czułości i od swoistości, wiec jest nie ma nią wpływu niezbalansowanie klas. Jednak, AUC może prowadzić do mylących wniosków w przypadku przecinających się krzywych ROC, gdzie jednej model może być

lepsza w jednym zakresie wartości progu decyzyjnego, a gorsza w innym. W takich sytuacjach wykorzystuje się analizę fragmentaryczną[14]

- **Funkcja celu (FC)** - w celu ułatwienia porównań pomiędzy różnymi modelami klasyfikacyjnymi zaproponowano autorską metrykę, będącą średnią ważoną trzech niezależnych miar jakości klasyfikacji: swoistości, F-miary oraz współczynnika korelacji Matthews'a (MCC).

  Ze względu na odmienny zakres wartości MCC (-1 do 1), dokonano jego normalizacji do przedziału [0,1] według wzoru:

$$\mathrm{nMCC} = \frac{\mathrm{MCC} + 1}{2} \tag{3.11}$$

  Zaproponowano następujące wagi: 0.5 dla F-miary, 0.3 dla swoistości oraz 0.2 dla znormalizowanego MCC. Wszystkie składowe funkcji celu przyjmują wartości z zakresu [0,1], co zapewnia ich porównywalność oraz umożliwia interpretację FC jako skali ogólnej jakości klasyfikatora.

  Funkcję celu definiuje wzór:

$$\mathrm{FC} = 0.5 \cdot F_1 + 0.3 \cdot \mathrm{Spec} + 0.2 \cdot \mathrm{nMCC} \tag{3.12}$$

  Opisana funkcja celu, zostanie także wykorzystana jako parametr optymalizacji hiperparametrów, który będzie maksymalizowano podczas eksperymentów.

Rozdział 4

Przetwarzanie i wybrane zbiory danych

4.1 Przegląd wykorzystanych zbiorów danych

TODO - napisać coś tutaj

4.1.1 MIMIC PERform AF Dataset

Zbiór danych MIMIC PERform AF[15] został pozyskany z bazy MIMIC-III Waveform Database Matched Subset[16]. Zawiera on 20-minutowe zapisy sygnałów fotopletyzmograficznych oraz elektrokardiograficznych, zarejestrowane u 35 ciężko chorych dorosłych pacjentów hospitalizowanych na oddziałach intensywnej terapii. U 19 z tych pacjentów zdiagnozowano migotanie przedsionków, natomiast u pozostałych zapis sygnałów nie wykazywał nieprawidłowości w rytmie serca. Dane zostały zebrane z wykorzystaniem monitora przyłóżkowego, z częstotliwością próbkowania wynoszącą 125 Hz.

4.1.2 Zbiór PPG według Liu, Zengding i Zhou et al.

Dane wykorzystywane w tym zbiorze[17] zostały zebrane w okresie od marca 2020 do marca 2021 w szpitalu Fuwai w Pekinie, będącym wyspecjalizowaną placówką w zakresie chorób sercowo-naczyniowych. Sygnał fotopletyzmograficzny został początkowo próbkowany z częstotliwością 250 Hz, a następnie poddany procesowi resamplingu do 100 Hz. W celu usunięcia niepożądanych zakłóceń zastosowano filtrację pasmową w zakresie 0.5 - 50 Hz. Przefiltrowany sygnał został następnie podzielony na segmenty 10-sekundowe, przy czym fragmenty zawierające istotne zakłócenia zostały odrzucone.

Cały zbiór danych obejmuje zapisy od 228 pacjentów, co odpowiada łącznie 118 217 segmentom 10-sekundowym. Na potrzeby badań publicznych udostępniono jedynie 46 827 z tych fragmentów. Dane zostały skategoryzowane według pięciu typów arytmii: przed-

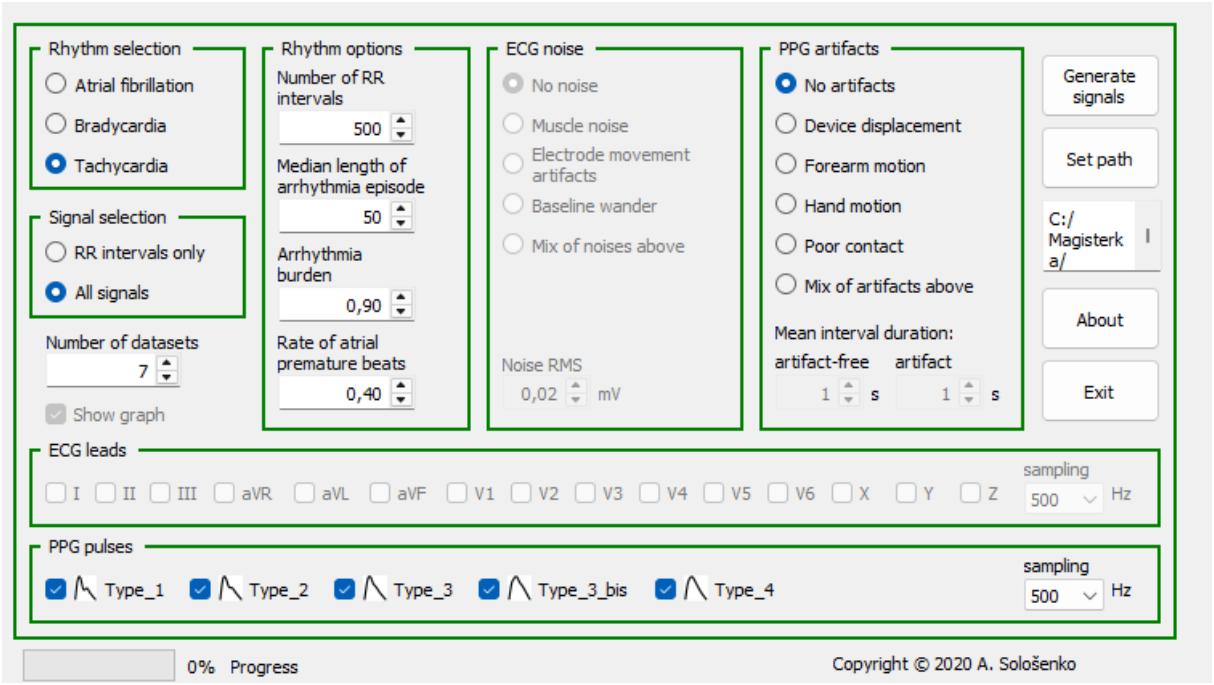
wczesny skurcz komorowy, przedwczesny skurcz przedsionkowy, tachykardia komorowa, tachykardia nadkomorowa oraz migotanie przedsionków.

4.1.3 PhysioNet/CinC Challenge 2015

test[18] Jest zbiorem użytym w konkursie majacym na celu stworzenie algorytmu redukującego ilość fałszych alarmów podczas detekcji arytmii serca. Dane pochodzą z czterech losowo wybranch, szpitali w Stanach Zjednoczonych oraz Europie. Udostępniona część zbioru zawiera 750 nagrań. Sygnały zostały próbkowane z częstotliwością 250Hz oraz przefiltrowane filtrem pasmowym 0.05 - 40Hz

4.1.4 Dane syntetyczne

W celu wzbogacenia zbiorów danych, zdecydowano użyć się danych syntetycznych wygenerowanych przy użyciu generatora sygnałów PPG i ECG z epizodami arrytmi[19]. Zastosowany symulator umożliwia pełną kontrolę nad procesem generowania sygnałów. Użytkownik ma możliwość definiowania szeregu parametrów determinujących charakterystykę sygnału, w tym: rodzaj arrytmi, mediane długości epizodów arytmi, obciążenie arytmią które determinuje łączny czas kiedy występuje arytmia, wskaźnik przedwczesnychskurczów przedsionkowych czy rodzaj pulsu zdefiniowany przez Dawber et al[20].



Rysunek 4.1: GUI użytego symulatora

W ramach eksperymentu wygenerowano pięć syntetycznych podzbiorów danych, z których każdy został zasymulowany z częstotliwością próbkowania równą 500 Hz oraz

zawierał sygnały odpowiadające 500 kolejnym interwałom RR. Każdy ze zbiorów zawierał równą ilośc przypadków bradycardii, tachycardii i migotania przedsionków. Zawierał się w nich także każdy rodzaj pulsu PPG. Zdecydowano się nie wykorzystywać możliwości generowania sygnałów z szumem.

Poszczególne podzbiory różniły się parametrami charakterystycznymi dla przebiegu arytmii:

Tabela 4.1: Porównanie parametrów podzbiorów danych syntetycznych

Zbiór	Mediana długości epizodu arytmii	Obciążenie arytmią	Wskaźnik przedwczesnych skurczów przedsionkowych
1	3	0.1	0.05
2	10	0.3	0.10
3	5	0.5	0.10
4	30	0.4	0.20
5	50	0.9	0.40

Każdy ze zbiorów ma symulować inny rodzaj arytmi:

- Zbiór 1 - Rzadkie występowanie krótkich epizodów arytmi
- Zbiór 2 - Umiarkowane występowanie z umiarkowaną liczbą przedwczesnych skórczy
- Zbiór 3 - Wysokie występowanie arytmi z krótkimi epizodami
- Zbiór 4 - Długie, częste epizody arytmi.
- Zbiór 5 - Niemal ciągła arytmia

Dzięki zastosowanemu symulatorowi, wygenerowane dane syntetyczne mogą stanowić cenne źródło informacji dla przyszłych modeli klasyfikacyjnych. Umożliwiają one nie tylko lepsze zrozumienie przypadków odstających, ale również wspierają proces uczenia modeli w scenariuszach bardziej typowych, odpowiadających klasycznym przypadkom klinicznym.

4.2 Przetwarzanie wstępne

Pomimo wysokiej jakości wybranych zbiorów danych, ich bezpośrednie wykorzystanie w procesie klasyfikacji wymaga zastosowania wieloetapowego przetwarzania wstępnego. Głównym celem tych transformacji jest ujednolicenie charakterystyki sygnałów pochodzących z różnych źródeł oraz przygotowanie reprezentatywnego wektora cech, który może zostać wykorzystany w procesie uczenia klasyfikatorów.

Pierwszym etapem przetwarzania wstępnego sygnału jest jego filtracja. W tym celu zastosowano filtr pasmowo-przepustowy o częstotliwościach granicznych 0.5 Hz oraz 40 Hz. Głównym celem tak dobranego filtra jest eliminacja zakłóceń pochodzących spoza pasma istotnego z punktu widzenia analizy sygnałów PPG. Dolne ograniczenie pozwala zmniejszyć wpływ ruchu pacjenta czy zmiane położenia czujnika, natomiast górna granica redukuje wpływ wysokoczęstotliwościowych zakłóceń, takich jak zakłócenia elektromagnetyczne.

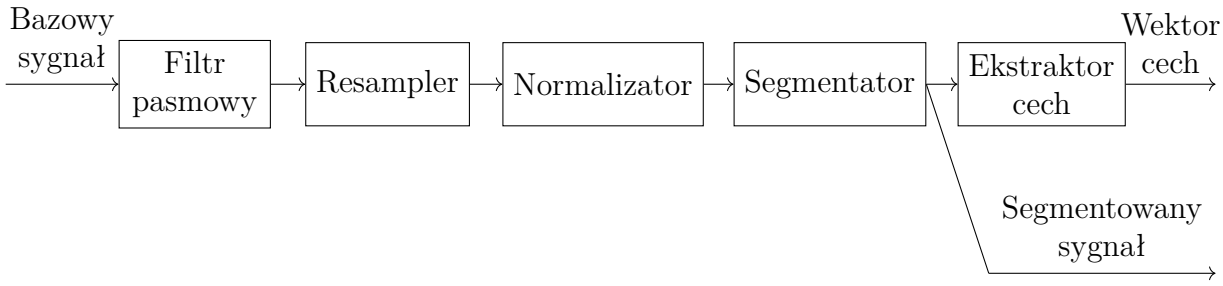
Drugim istotnym etapem przetwarzania wstępnego była zmiana częstotliwości próbkowania sygnałów. Zdecydowano się na jednolitą częstotliwość 100 Hz, co odpowiada najniższej wartości występującej wśród wykorzystanych zbiorów danych. Ujednolicenie próbkowania do tej wartości niesie ze sobą kilka korzyści. Po pierwsze, redukcja liczby próbek skutkuje zmniejszeniem rozmiaru danych, co przekłada się na krótszy czas ich przetwarzania jak i nauki modeli w sposób bezcechowych.

Kolejnym krokiem w procesie przetwarzania wstępnego była normalizacja sygnału do przedziału $[0, 1]$. Wykorzystane zbiory danych charakteryzowały się zróżnicowanym zakresem wartości amplitud, co mogłoby negatywnie wpłynąć zarówno na proces uczenia cechowego, jak i bezcechowego. Niejednorodna skala danych może prowadzić do dominacji niektórych sygnałów lub cech w trakcie uczenia. W celu minimalizacji tego efektu, zdecydowano się na niezależną normalizację każdego sygnału osobno.

Z uwagi na fakt, że zbiór danych opracowany przez Liu, Zengding i Zhou et al.[17] składa się z 10-sekundowych segmentów sygnału, zdecydowano o ujednoliceniu długości wszystkich pozostałych próbek poprzez ich podział na fragmenty o identycznym czasie trwania. Segmentacja została przeprowadzona po wcześniejszym przeskalowaniu częstotliwości próbkowania do 100 Hz, co oznacza, że każdy 10-sekundowy segment zawiera dokładnie 1000 próbek. Próbki te, zostaną wykorzytane do bezcechowej nauki klasyfikatorów.

Ostatnim etapem przetwarznia wstępnego jest wybranie oraz ekstrakcja cech. Proces ten zostanie omówiony dokładnie w rozdziale 4.3

Rysunek 4.2: Schemat przetwarznia wstępnego



4.3 Ekstrakcja cech

Zdecydowano się na wybór czterech głównych domen analizy, z których każda interpretuje sygnał PPG z innej perspektywy, umożliwiając wydobycie zróżnicowanych informacji. Niezależnie od wybranej domeny, we wszystkich przypadkach zastosowano zestaw podstawowych miar statystycznych, które zostały wyekstrahowane z przekształconych sygnałów:

- Średnia arytmetyczna,
- Mediana,
- Odchylenie standardowe,
- Wariancja,
- Rozstęp międzykwartylowy,

4.3.1 Cechy z domeny czasu

Pierwszą z wybranych domen, interpretuje sygnał jako funkcję ciągłą w czasie. Domena czasu trakuje kolejne próbki czasowe jako kolejne wartości funkcji aplitudy. Domena ta bezpośrednio analizuje sygnału w jej pierwotnej postaci, gdzie $x = \{x_1, x_2, \ldots, x_n\}$. W tej przestrzeni, zdecydowano się na wybranie następujących cech:

- **Skośność** - miara asymetrii sygnału:

$$\frac{1}{N}\sum_{i=1}^{N}\left(\frac{x_i - \bar{x}}{\sigma}\right)^3 \tag{4.1}$$

  gdzie:

  - $\bar{x}$ - średnia arytmetyczna.
  - σ - odchylenie standardowe sygnału.

- **Współczynnik zmienności** - względna miara zróżnicowania sygnału:

$$\mathrm{CV} = \frac{\sigma}{\bar{x}} \tag{4.2}$$

- **Średnie odchylenie bezwzględne** - bezwzgledna miara zróżnicowania sygnału:

$$\text{MAD} = \frac{1}{n}\sum_{i=1}^{n}|x_i - \bar{x}| \tag{4.3}$$

- **Entropia Shannona** - miara niepewności sygnału:

$$H = -\sum_{i=1}^{N} p_i \log_2 p_i \tag{4.4}$$

gdzie:

  - p_i - prawdopodobieństwo wystąpienia i-tej wartości sygnału.

4.3.2 Cechy różnicowe

Kolejną z wybranych domen jest domena cech różnicowych, w której sygnał nie jest bezpośrednio intereptowany jako kolejne wartości amplitudy, tylko poprzez różnice zmian między kolejnymi próbkami $\Delta x = \{x_2 - x_1,\, x_3 - x_2,\, \ldots,\, x_n - x_{n-1}\}$. Cechy wyekstrachowane z tej domeny:

- **Procent dodatnich różnic** - odsetek dodatkich róznic kolejnych wartości sygnału

$$\frac{100}{N-1}\sum_{i=1}^{N-1}\delta(\Delta x > 0) \tag{4.5}$$

gdzie:

  - $\delta(\cdot)$ - funkcja wskaźnikowa, równa 1 gdy warunek jest spełniony, 0 w przeciwnym wypadku.

- **Średnia wartość bezwzględna różnic:**

$$\frac{1}{N-1}\sum_{i=1}^{N-1}|\Delta x| \tag{4.6}$$

- **Pierwiastek średniokwadratowy różnic** - miara zmienności rytmu serca, która odzwierciedla aktywność układu przywspółczulnego:[21]

$$\text{RMSSD} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N-1}(\Delta x)^2} \tag{4.7}$$

- **Znormalizowane średnie odchylenie bezwzględne różnic** - ocenia lokalną dynamikę zmian sygnału:

$$\frac{1}{N-1}\sum_{i=1}^{N-1}\frac{|\Delta x|}{\bar{x}} \tag{4.8}$$

- **Znormalizowana suma bezwzględnych różnic:**

$$\frac{\sum_{i=1}^{N-1} |\Delta x|}{\sum_{i=1}^{N} |x_i|} \tag{4.9}$$

4.3.3 Cechy częstotliwościowe

Ostatnią wybraną domeną jest domena częstotliwościowa. W tej domenie, sygnał jest przekszłcany przy użyciu transoframty Fouriera, aby uzyskać jego rozkład mocny w zależności częstotliwości $P_{xx}(f)$, gdzie f oznacza częstotliwość. Analiza cech częstotliwościowych, pozwala identyfikować składowe dominujące jak i nieregularności. W tej domenie zdecydowano się na ekstrakcję następujących cech:

- **Maksymalny szczyt widmowy** - określa częstotliwość która posiada największą ilość energii

$$f_{\max} = f\Big(\arg\max_{f} P_{xx}(f)\Big) \tag{4.10}$$

- **Kurtoza widma** - spłaszczenie widma sygnału:

$$\operatorname{kurt}(P_{xx}) = \frac{1}{M}\sum_{i=1}^{M}\left(\frac{P_{xx}(f_i) - \overline{P_{xx}}}{\sigma_{P_{xx}}}\right)^4 \tag{4.11}$$

- **Energia spektralna** - suma mocy sygnału w całym paśmie częstotliwości:

$$E = \sum_{i=1}^{M} P_{xx}(f_i) \tag{4.12}$$

- **Udział wysokich szczytów widma**

$$\frac{\sum_{i=1}^{M} \delta(P_{xx}(f_i) > \overline{P_{xx}})}{M} \tag{4.13}$$

4.3.4 Implementacja

Wstępne przetwarzanie zbiorów danych zostało zrealizowane z wykorzystaniem środowiska MATLAB w wersji R2024b. Wybór ten podyktowany był faktem, iż większość analizowanych zbiorów danych dostarczana była w postaci plików z rozszerzeniem .mat, będących binarnym formatem przechowywania danych specyficznym dla środowiska MATLAB.

Proces wstępnego przekształcania wymagał indywidualnego dostosowania skryptów dla każdego zbioru danych, co wynikało z różnic w sposobie reprezentacji danych. W części przypadków dane zapisane były jako pojedyncze sygnały, natomiast w innych przyj-

mowały postać struktur zawierających dodatkowe informacje kontekstowe, takie jak metadane dotyczące sygnału oraz dane pacjenta.

Proces zmiany częstotliwości próbkowania sygnału został zrealizowany przy użyciu funkcji `resample()`. Funkcja ta najpierw przeprowadza interpolację sygnału, zwiększając jego częstotliwość próbkowania, następnie sygnał jest filtrowany za pomocą dolnoprzepustowego filtra FIR, a ostatecznie następuje redukcja częstotliwości próbkowania do wartości docelowej [22].

Proces wyodrębniania wektora cech został również przeprowadzony w środowisku MATLAB. Do obliczenia podstawowych miar statystycznych wykorzystano funkcje dostępne w standardowym pakiecie języka. W przypadku bardziej złożonych cech zastosowano narzędzia udostępniane w ramach rozszerzeń Statistics and Machine Learning Toolbox oraz Signal Processing Toolbox.

Rozdział 5

Detekcja arytmii serca

Całość skryptów dotyczących detekcji arytmii serca z wykorzystaniem metod uczenia maszynowego została zaimplementowana w języku Python w wersji 3.11. W celu ułatwienia procesu budowy, trenowania oraz ewaluacji klasyfikatorów tradycyjnych wykorzystano bibliotekę scikit-learn, która oferuje szeroki zestaw gotowych algorytmów klasyfikacyjnych, funkcji oceny jakości modeli oraz narzędzi do przetwarzania i transformacji danych. W przypadku implementacji modeli opartych na sztucznych sieciach neuronowych zastosowano bibliotekę TensorFlow.

Cały proces nauki i testowania modeli został przeprowadzony na komputerze osobistym o następującej specyfikacji sprzętowej: procesor Intel Core i5-13400F, 32 GB pamięci RAM oraz karta graficzna NVIDIA GeForce RTX 4070.

5.1 Architektura i konfiguracja modeli

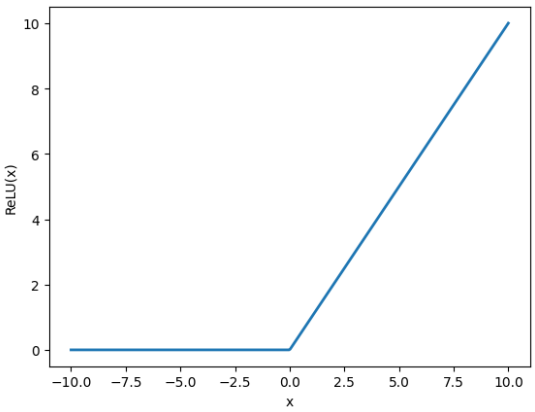
W pierwszym etapie nauki zdecydowano się na jedynie nieznaczną modyfikację domyślnych parametrów klasycznych klasyfikatorów. Dla algorytmu K najbliższych sąsiadów zastosowano metrykę Minkowskiego z parametrem q=2, co odpowiada wykorzystaniu klasycznej metryki euklidesowej. Liczbę sąsiadów ustalono na k=50. W przypadku drzewa decyzyjnego ograniczono maksymalną głębokość drzewa do 30 testów logicznych, co miało na celu redukcję ryzyka przeuczenia modelu. Dla klasyfikatora typu las losowy wykorzystano 100 drzew decyzyjnych, z których każde ograniczono do maksymalnie 10 testów logicznych.

Naiwny klasyfikator Bayesa, ze względu na swoje czysto statystyczne działanie, nie posiada klasycznych hiperparametrów wpływających na proces uczenia. Jedynym parametrem regulacyjnym jest wartość wariancji dodawanej do wszystkich cech w celu stabilizacji obliczeń numerycznych, którą ustawiono na 10^{-8}. Dopierajac hiperparametry dla metody wektorów nośnych, wybrano jądro rbf które w celu obliczenia odległości punktów od hierpłaszczyzny wykorzystuje klasyczną odległość euklidesowsą.

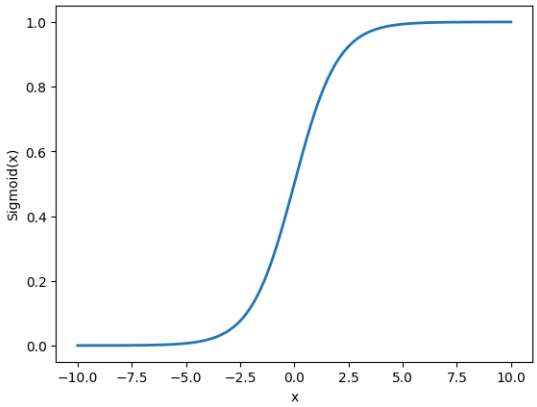
W kontekście trzech modeli boostingowych, zdecydowano się nie zmieniać żadnych parametrów podstawowych.

W przypadku nauki przy użyciu cech, została użyta tylko jedna architektóra sieci neuronowej, oparta na warstwach gęstych. Pierwszą warstwą była warstwa wejściowa posiadają 26 neuronów, odpowiadającym cechom wejściowych. Następnie 4 warstwy ukryte, każda z nich posiadała funkcje aktywacji ReLU. Warstwy posiadały 1024, 512, 256, 128 ilość neuronów. Ostatnia warstwa posiadała tylko jeden neuron, a funkcją aktywacji był sigmoid.

Rysunek 5.1: Wykresy wybranych funckji aktywacji



(a) ReLU



(b) Sigmoid

W prezentowanej architekturze zastosowano funkcję aktywacji ReLU w warstwach ukrytych, ze względu na jej wysoką efektywność obliczeniową oraz zdolność do wspomagania procesu uczenia głębokich sieci neuronowych. ReLU także minimalizuje problem zanikania gradientu.

W warstwie wyjściowej wykorzystano funkcję aktywacji sigmoidalną, której właściwości umożliwiają interpretację wyjścia modelu jako prawdopodobieństwa przynależności próbki do klasy pozytywnej. W związku z tym zastosowano jeden neuron w warstwie wyjściowej, którego wartość aktywacji reprezentuje stopień pewności modelu co do przypisania obserwacji do klasy pozytywnej.

Do optymalizacji wag modelu wykorzystano algorytm ADAM, który łączy zalety metod opartych na momentum oraz RMSProp. Optymalizator ten adaptacyjnie dostosowuje szybkość uczenia dla każdej z wag, uwzględniając zarówno średnie wartości gradientów z poprzednich epok, jak i ich skale[23, 24].

Ze względu na rodzaj problemu, wybraną funkcją strat została binarna entropia krzy-

żowa. Funkcję tą można opisać przy pomocy wzoru:

$$\mathcal{L}_{\text{avg}} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)\right) \tag{5.1}$$

gdzie:

- N liczba próbek w zbiorze,
- y_i prawdziwa etykieta i-tej próbki,
- $\hat{y}_i$ prawdopodobieństwo przypisane klasie pozytywnej dla i-tej próbki.

W eksperymencie przyjęto, że proces uczenia modelu będzie przebiegać przez 10 epok z rozmiarem partii równym 64. Zastosowano również dynamiczną regulacje kroku uczenia, w której początkowa wartość kroku uczenia wynosiła 0.001

Od piątej epoki zastosowano wykładniczy schemat zmniejszania kroku uczenia, zgodny z równaniem:

$$Lr(epoka) = \begin{cases} Lr_0 & \text{jeżeli } epoka < 5 \\ Lr(epoka - 1) \times e^{-0.1} & \text{jeżeli } epoka \geq 5 \end{cases} \tag{5.2}$$

gdzie:

- Lr - krok uczenia
- Lr_0 - początkowy krok uczenia

W przypadku uczenia bezcechowego, zdecydowano się na przetestowanie czterech różnych architektur sieci neuronowych, zróżnicowanych pod względem rodzaju zastosowanych warstw.

Pierwsza architektura odpowiadała strukturze wykorzystanej w eksperymencie z uczeniem cechowym, co umożliwia bezpośrednie porównanie efektywności obu metod względem tego samego problemu klasyfikacyjnego.

Druga architektura wykorzystywała warstwy konwolucyjne jednowymiarowe , przeplatane warstwami MaxPooling1D, których zadaniem było redukowanie wymiarowości sygnału poprzez wybór wartości maksymalnej w lokalnym sąsiedztwie. Po zakończeniu ekstrakcji cech konwolucyjnych, zastosowano warstwę Flatten, która przekształca dane o strukturze wielowymiarowej do postaci wektora jednowymiarowego. Operacja ta jest niezbędna, aby możliwe było podłączenie kolejnych warstw gęstych, które przetwarzają wektor cech wygenerowany przez wcześniejsze warstwy konwolucyjne.

Trzecia rozważana architektura opiera się na dwóch sekwencyjnie połączonych warstwach typu LSTM, które są zdolne do modelowania zależności czasowych w sygnale.

Warstwy te zostały uzupełnione o końcowe warstwy gęste, umożliwiające dokonanie klasyfikacji na podstawie reprezentacji uzyskanej z przetworzenia sekwencji czasowej.

Czwarta architektura stanowi połączenie podejść zastosowanych we wszystkich trzech wcześniejszych modelach. W pierwszym etapie wykorzystywane są warstwy konwolucyjne jednowymiarowe wraz z warstwami MaxPooling1D, których zadaniem jest lokalna ekstrakcja cech oraz redukcja wymiarowości danych. Wynikowy tensor jest następnie przekazywany do warstw LSTM, które modelują zależności czasowe. Ostatecznie, przetworzony wektor cech trafia do warstw gęstych, które realizują końcową klasyfikację.

We wszystkich opisanych architekturach przyjęto identyczny zestaw hiperparametrów procesu uczenia, jak w przypadku modelu uczącego się na danych cechowych. Dzięki temu możliwe było porównanie skuteczności różnych architektur przy zachowaniu spójnych warunków eksperymentalnych.

Zdecydowano się także, na przeprowadzenie procesu optymalizacji hiperparametrów modeli. Proces ten polegał na iteracyjnym zmianie hiperparametrów i testowaniu modeli. Całość procesu została wykonania przy pomocy bibloteki Optuna. Wykorzystuje ona strategie sekwencyjnej optymalizacji bayesowskiej. Metoda ta jest znacznie szybka niż wykorzystanie siatki parametrów lub losowego sprawdzania parametrów. Metoda ta, buduje modele probabilistyczne które uczą się procesu wybierania kolejnych zestawów hiperparametrów, balansując eksploracje nowych obszarów jak i ulepszanie znanych.

Nazwy hiperparametrów zamieszczone w tabeli 5.1 zostały zaczerpnięte z dokumentacji technicznej wykorzystanej biblioteki [3].

W przypadku algorytmu K najbliższych sąsiadów zdecydowano się na optymalizację dwóch kluczowych hiperparametrów. Pierwszym z nich była liczba sąsiadów k, wykorzystywana do predykcji klasy nowych próbek. W ramach procedury wyszukiwania optymalnych wartości badano wartości K z przedziału od 5 do 200. Drugim z optymalizowanych parametrów był parametr p w funkcji odległości Minkowskiego, który wpływa na sposób obliczania dystansu pomiędzy próbkami. W tym przypadku analizowano wartości całkowite z zakresu od 1 do 5.

Drzewo decyzyjne dysponuje większą liczbą hiperparametrów, które mogą podlegać procesowi optymalizacji. Parametr criterion odpowiada za sposób oceny jakości podziału danych w węzłach drzewa. Domyślną wartością tego parametru jest "gini", który opiera się na analizie proporcji klas. Alternatywnie, rozważono również wartość "entropy", która wykorzystuje entropię do określenia niepewności rozkładu klas w danym węźle.

Parametr max features definiuje maksymalną liczbę cech rozważanych podczas wyboru cechy, według której następuje podział. Natomiast parametry min samples split oraz min

samples leaf kontrolują odpowiednio minimalną liczbę próbek wymaganą do dokonania podziału oraz minimalną liczbę próbek w liściu. Wartości tych trzech parametrów przeszukiwano w zbiorze liczb całkowitych od 1 do 20.

Ostatnim optymalizowanym parametrem było max depth, które określa maksymalną głębokość drzewa decyzyjnego, a tym samym maksymalną liczbę testów logicznych prowadzących od korzenia do liścia. Zakres wartości tego parametru obejmował liczby całkowite z przedziału od 3 do 20.

Las losowy dziedziczy wszystkie parametry drzewa decyzyjnego ze względu na swoje działanie oparte o równoległą naukę tych modeli. Dlatego parametry criterion, max features, min samples split, min samples leaf oraz max depth są optymalizowane w takich samych przedziałach jak w przypadku drzewa decyzyjnego.

Unikatowym hiperparametrem, który podlegał optymalizacji, jest n estimators, określający liczbę drzew decyzyjnych wchodzących w skład lasu losowego. Wartości tego parametru przeszukiwano w zbiorze liczb całkowitych z zakresu od 1 do 200.

Najważniejszym parametrem dla klasyfikatora SVM jest jądro, przy pomocy którego dokonywane są wewnętrzne przekształcenia przestrzeni cech. Zdecydowano się sprawdzić wszystkie cztery możliwości: linear, który oblicza iloczyn skalarny dwóch wektorów; poly, który rozszerza dane do przestrzeni wielomianowej; rbf, który wykorzystuje eksponentę kwadratu odległości euklidesowej; oraz sigmoid, który używa funkcji tangensa hiperbolicznego. Porównanie działania poszczególnych jąder przedstawia Rysunek 3.2.

Jądra wykorzystywane w metodzie SVM korzystają z dwóch wspólnych parametrów. Pierwszy z nich, gamma, określa wpływ pojedynczego punktu treningowego na przebieg granicy decyzyjnej. W niniejszym eksperymencie gamma może przyjmować dwie wartości: scale, która uwzględnia wariancję oraz liczbę cech, oraz auto, która opiera się wyłącznie na liczbie cech. Drugim parametrem jest C, który kontroluje siłę regularyzacji L2 podczas treningu. Parametr ten może przyjmować wartości zmiennoprzecinkowe z przedziału od 10^{-5} do 10^{5}.

Ze względu na zróżnicowane działanie poszczególnych jąder, istnieją również parametry charakterystyczne dla niektórych z nich. Parametr degree określa rząd wielomianu, do którego rozwijane jest jądro poly, jego wartość będzie sprawdzana z liczb całkowitych od 2 do 5. Z kolei parametr coeff0 stanowi wartość przesunięcia dla jąder typu poly oraz sigmoid. Sprawdzanymi wartościami tego parametru będą liczby zmiennoprzecinkowe od -1 do 1. Jądra linear oraz rbf nie posiadają dodatkowych, charakterystycznych parametrów poza gamma oraz C.

Tabela 5.1: Testowane zakresy hiperparametrów

Hiperparametr	**Typ**	**Wartości**
	KNN	
K	Liczba całkowita	(5, 200)
p	Liczba całkowita	(1, 5)
	DRZEWO DECYZYJNE	
Criterion	Kategoryczny	gini, entropy
Max features	Kategoryczny	sqrt, log2, None
Min samples split	Liczba całkowita	(1, 20)
Min samples leaf	Liczba całkowita	(1, 20)
Max depth	Liczba całkowita	(3, 20)
	LAS LOSOWY	
Criterion	Kategoryczny	gini, entropy
Max features	Kategoryczny	sqrt, log2, None
Min samples split	Liczba całkowita	(1, 20)
Min samples leaf	Liczba całkowita	(1, 20)
Max depth	Liczba całkowita	(3, 20)
n estimators	Liczba całkowita	(1, 200)
	SVM	
Kernel	Kategoryczny	linear, poly, rbf, sigmoid
Gamma	Kategoryczny	scale, auto
Degree	Liczba całkowita	(2, 5)
Coef0	Liczba dziesiętna	(-1. 1)
C	Liczba dziesiętna	$(10^{-5}, 10^{5})$

5.2 Walidacja wyników

W celu oceny jakości predykcji modeli zastosowano dwie niezależne metody walidacji krzyżowej. TODO DOPISAĆ COŚ TUTAJ

5.2.1 Walidacja holdout

Pierwszą z użytych metod jest walidacja krzyżowa metodą holdout polega na jednorazowym podziale dostępnych danych na dwa rozłączne podzbiory: zbiór treningowy oraz zbiór testowy. Zbiór treningowy wykorzystywany jest do uczenia modelu, natomiast zbiór testowy służy do oceny jego zdolności generalizacji na nieznanych wcześniej danych. Najczęściej stosowaną proporcją podziału jest 80:20, gdzie 80% danych trafia do zbioru treningowego, a pozostałe 20% do testowego. Należy jednak podkreślić, że nie jest to wartość sztywno ustalona - wybór proporcji zależy od liczby dostępnych próbek oraz charakterystyki problemu.

W niektórych przypadkach, szczególnie przy zastosowaniu złożonych modeli wymagających doboru hiperparametrów, ze zbioru treningowego dodatkowo wydziela się zbiór walidacyjny. Wówczas zbiór treningowy dzielony jest na dwa podzbiory: zbiór uczący oraz zbiór walidacyjny. Zbiór walidacyjny służy do oceny jakości modelu podczas procesu dobór optymalnych hiperparametrów bez ryzyka przecieku informacji ze zbioru testowego. Jego wykorzystanie pozwala na kontrolowanie lepsze procesu uczenia, np. poprzez mechanizm wczesnego zatrzymania, który przerywa naukę modelu w momencie, gdy jakość predykcji na zbiorze walidacyjnym przestają się poprawiać, co ogranicza ryzyko przeuczenia.

RYSUNEK

Aby uniknać przecieku informacji, polegającego na tym, że model ma dostęp do danych testowych już na etapie treningu, z decydowano się użyć zbiorów X Y Z jako zbioru treningowego oraz zbioru ABC jako zbiór testujący. Przeciek taki prowadzi do sztucznego zawyżenia wartości metryk jakościowych, co z kolei fałszuje rzeczywistą skuteczność modelu w warunkach niezależnych od zbioru treningowego. W kontekście wdrożeniowym, gdzie modele są stosowane do danych nieobecnych podczas treningu, taki przeciek znacząco obniża przewidywaną skuteczność modeli. Z tego względu procedura walidacji została przeprowadzona w sposób rygorystyczny, z zachowaniem pełnej separacji danych treningowych i testowych.

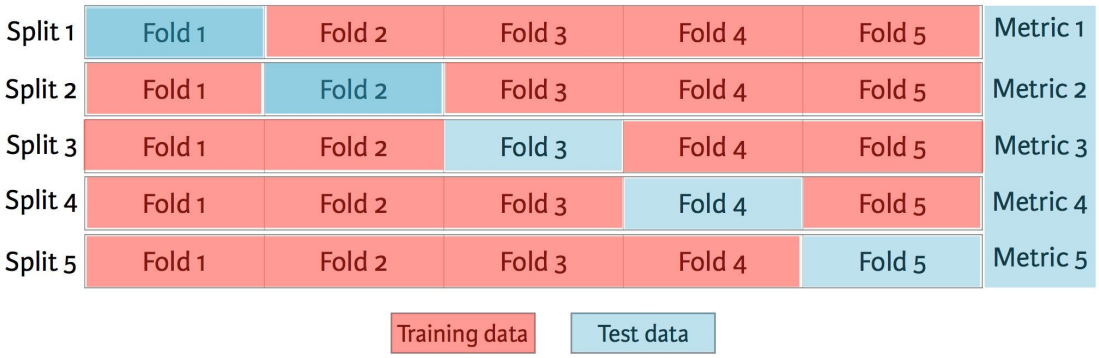
Nauca cechowa

Nauca bezcechowa

Optymalizacjia hiperparametrów

5.2.2 Walidacja K-Fold

Drugą zastosowaną metodą oceny jakości modelu jest walidacja krzyżowa K-krotna. Procedura ta polega na K-krotnym podziale dostępnego zbioru danych na K zbliżonych rozmiarowo części, zwanych foldami. W każdej iteracji jeden z foldów pełni rolę zbioru testującego, natomiast pozostałe K-1 foldy służą do trenowania modelu. Proces ten jest powtarzany K razy, za każdym razem z innym foldem jako zestawem testowym. Dzięki temu każda obserwacja jest wykorzystana dokładnie raz do walidacji oraz K-1 razy do uczenia modelu. Zdecydowano się na wykonanie walidacji przy pomocy 5 foldów.



Rysunek 5.2: Wizualizacja walidacji Kfold[25]

W przypadku zastosowania walidacji K-krotnej, w celu uniknięcia przecieku danych, zdecydowano się na ręczny przydział pacjentów do poszczególnych foldów. Dzięki temu próbki pochodzące od tego samego pacjenta nie występowały jednocześnie w zbiorze treningowym i walidacyjnym w żadnej z iteracji, co mogłoby prowadzić do sztucznego zawyżenia ocenianych metryk. Dodatkowo, w celu zapewnienia wiarygodności wyników, dane syntetyczne zostały wykluczone z procesu testowania, lecz były uwzględniane w trakcie uczenia modelu w każdej iteracji.

Ze względu na charakter metody walidacji krzyżowej K-Fold, w ramach jednego procesu walidacyjnego model jest trenowany i testowany K razy. Skutkuje to otrzymaniem K niezależnych wyników dla każdej z metryk oceny modelu. W celu ich przejrzystej prezentacji zdecydowano się na przedstawienie rezultatów w postaci wartości średniej $\pm$ odchylenie standardowe. Taki sposób raportowania umożliwia nie tylko ocenę przeciętnej skuteczności modelu, ale również pozwala na analizę rozrzutu wyników między poszczególnymi foldami, dając pełniejszy obraz jakości modelu.

Nauca cechowa

Tabela 5.2: Wartości miar jakościowych dla nauki cechowej dla walidacji Kfold

	Dokładność	**Spec**	**F-miara**	**nMCC**	**FC**
KNN	0.83 ± 0.01	0.74 ± 0.03	0.87 ± 0.01	0.82 ± 0.01	0.82 ± 0.01
Drzewo Decyzyjne	0.83 ± 0.01	0.77 ± 0.02	0.86 ± 0.00	0.83 ± 0.01	0.83 ± 0.01
Las Losowy	0.84 ± 0.01	0.72 ± 0.03	0.87 ± 0.01	0.83 ± 0.01	0.82 ± 0.01
Bayes	0.50 ± 0.04	0.33 ± 0.28	0.57 ± 0.17	0.47 ± 0.02	0.48 ± 0.01
SVM	0.79 ± 0.01	0.61 ± 0.03	0.84 ± 0.01	0.78 ± 0.01	0.76 ± 0.01
LLSVM	0.78 ± 0.01	0.61 ± 0.02	0.83 ± 0.01	0.76 ± 0.01	0.75 ± 0.01
XGBoost	0.86 ± 0.01	0.81 ± 0.02	0.89 ± 0.01	0.86 ± 0.01	0.86 ± 0.01
CatBoost	0.86 ± 0.01	0.81 ± 0.02	0.89 ± 0.01	0.86 ± 0.01	0.86 ± 0.01
LightBoost	0.85 ± 0.01	0.78 ± 0.03	0.88 ± 0.01	0.84 ± 0.01	0.84 ± 0.01
Sieć Neuronowa	0.82 ± 0.01	0.73 ± 0.05	0.85 ± 0.01	0.81 ± 0.01	0.81 ± 0.02

Analizując dane przedstawione w tabeli 5.2, można zaobserwować, że modele boostingowe najlepiej poradziły sobie z zadaniem wykrywania arytmii. Osiągnęły one najwyższe wartości we wszystkich miarach jakościowych. Modele XGBoost i CatBoost uzyskały identyczne wyniki, co wynika z faktu, że wykorzystują one zbliżone techniki modelowania zależności pomiędzy cechami a klasą. Charakteryzują się one wysokim poziomem równowagi pomiędzy precyzją a czułością, co potwierdza wysoka wartość F-miary. Jednocześnie nie tracą zdolności do wykrywania klasy zdrowych pacjentów, co potwierdza średnia swoistość na poziomie 81%

Drugim najlepszym modelem pod względem wartości funkcji celu okazało się drzewo decyzyjne, którego wynik był zaledwie o 0,03 punktu procentowego niższy niż dla najlepszego modelu w przypadku funkcji celu, nMCC, F-miary oraz dokładności.

Las losowy, pomimo agregowania wielu drzew decyzyjnych, osiągnął nieco niższe wartości wskaźników jakościowych, z wyjątkiem F-miary, która była wyższa o 0,01 punktu procentowego. Może to sugerować, że model ten wymaga dokładniejszego strojenia hiperparametrów.

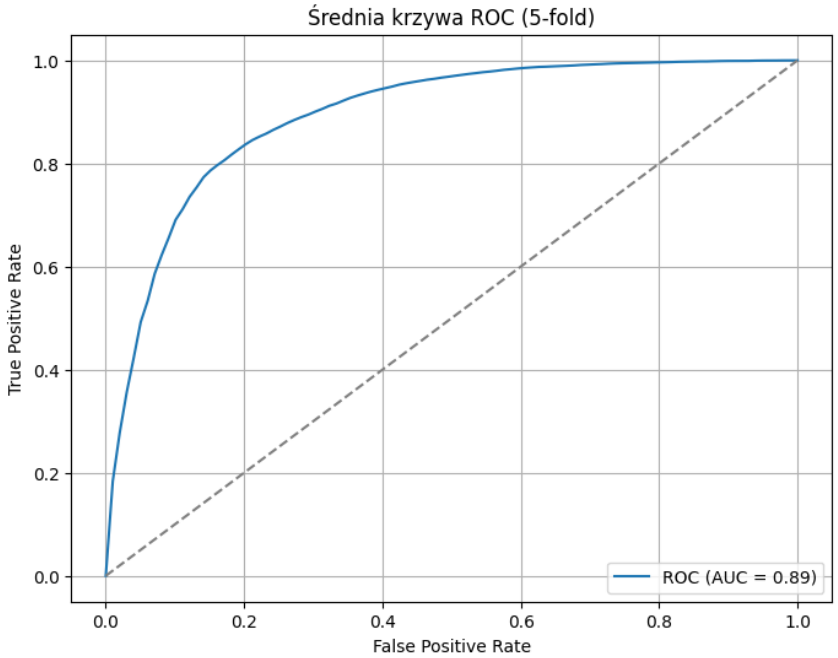
W przypadku klasyfikatora KNN wartości miar jakościowych były porównywalne do wyników uzyskanych przez las losowy.

Model SVM i LLSVM poradził sobie porównywalnie dobrze, ale SVM posiada każdy wskaźnij jakościowy nie gorszy niż LSSVM. Wciąż będać mniej wydajnym niż wcześnij omawiane modele.

Najgorzej z zadaniem wykrywania arytmii poradził sobie naiwny klasyfikator Bayesa. Uzyskał on najniższe wartości miar jakościowych: swoistość na poziomie 0.33 oraz nMCC

wynoszące 0.47, co przełożyło się na ujemną wartość współczynnika MCC równą -0.06. Ponadto model ten cechował się największym rozrzutem wyników, podczas gdy odchylenie standardowe innych klasyfikatorów nie przekraczało 0.05, w tym przypadku osiągnęło nawet 0.28.

Sięc neuronowa, osiąngęła wyniki porównywalne z dobrymi klasyfikatorami, które są lepsze niż modele maszyn nośnych, ale gorsze niż las losowy, osiągajac wartość funkcji celu i nMCC na poziomie 0.81.



Rysunek 5.3: Średnia krzywa ROC

Średnia krzywa ROC, wyznaczona na podstawie pięciu foldów walidacyjnych, wskazuje na bardzo dobrą zdolność modelu do klasyfikacji, osiągając wartość AUC równą 0,89. Krzywa ta wykazuje wyraźne wygięcie w kierunku lewego górnego narożnika wykresu oraz płaskie podejście do prawego górnego narożnika, co świadczy o wysokiej skuteczności klasyfikatora. Brakuje w niej fragmentów przebiegających blisko przekątnej. Krzywa jest ponadto stabilna i regularna, co dodatkowo potwierdza jej jakość.

Nauca bezcechowa

Tabela 5.3: Wartości miar jakościowych dla nauki bezcechowej dla walidacji Kfold

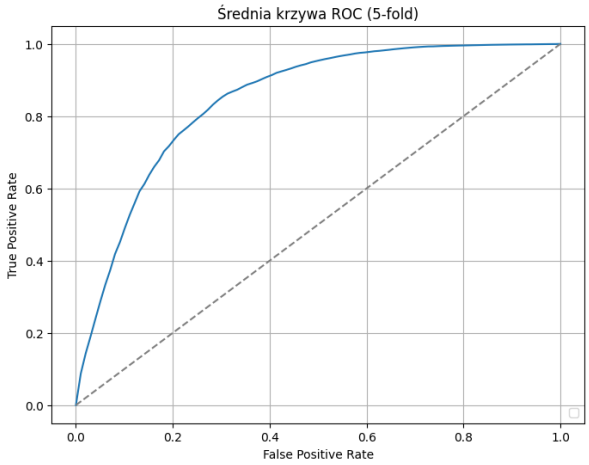
	Dokładność	**Spec**	**F-miara**	**nMCC**	**FC**
Gęsta	0.79 ± 0.01	0.61 ± 0.07	0.84 ± 0.01	0.77 ± 0.01	0.76 ± 0.02
CNN	0.78 ± 0.01	0.61 ± 0.07	0.83 ± 0.00	0.77 ± 0.01	0.75 ± 0.02
LSTM	0.63 ± 0.03	0.16 ± 0.13	0.76 ± 0.01	0.56 ± 0.07	0.53 ± 0.07
CNN-LSTM	0.61 ± 0.00	0.00 ± 0.00	0.75 ± 0.00	0.50 ± 0.00	0.48 ± 0.00

Analizując tabelę 5.3, można zauważyć, że dwa pierwsze modele, sieć gęsta oraz CNN, poradziły sobie porównywalnie dobrze z zadaniem klasyfikacji, osiągając wartość znormalizowanego współczynnika korelacji równą 0,77. Pozostałe miary jakości również prezentują zadowalający poziom. Jedynym wskaźnikiem wyraźnie odbiegającym od reszty jest swoistość, która wyniosła zaledwie 0,61 i charakteryzowała się największym rozrzutem wartości.

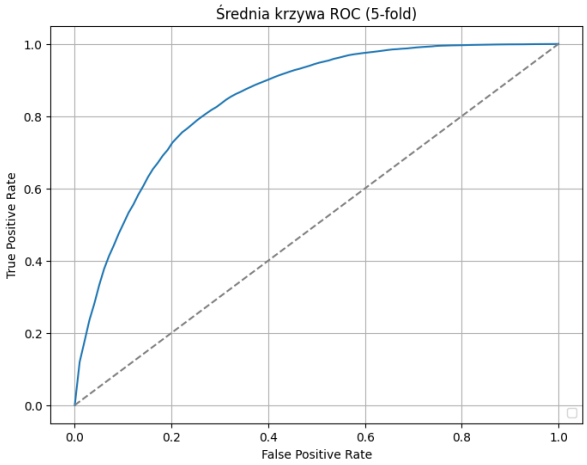
Dodanie warstw LSTM doprowadziło do pogorszenia skuteczności klasyfikatorów. W skrajnym przypadku, sieć o architekturze CNN-LSTM klasyfikowała wszystkich pacjentów jako chorych, o czym świadczy zerowa wartość swoistości oraz nMCC wynoszący jedynie 0,50. Model ten uzyskał także najniższą wartość funkcji celu, równą zaledwie 0.48. Sieć LSTM bez warstw konwolucyjnych osiągnęła wyniki nieznacznie lepsze niż CNN-LSTM posiadajac swoistość na poziomie 0.16 oraz nMCC równe 0.56.

Analiza krzywych ROC przedstawionych na rysunku 5.4 potwierdza wcześniej sformułowane wnioski. Można zaobserwować istotne różnice pomiędzy krzywymi ROC dla sieci gęstej (5.4a) i sieci konwolucyjnej (5.4b), a krzywymi dla modeli zawierających warstwy LSTM (5.4c) oraz CNN-LSTM (5.4d). Dwa pierwsze modele charakteryzują się krzywymi silnie wygiętymi w kierunku lewego górnego rogu wykresu oraz znacznym oddaleniem od przekątnej a ich pola pod krzywą wynoszą około 0.85.W przeciwieństwie do tego, krzywe ROC modeli zawierających warstwy LSTM są zbliżone do przekątnej wykresua ich wartości AUC oscylują wokół 0.5, co sugeruje niemal losowe prognozowanie klasy.

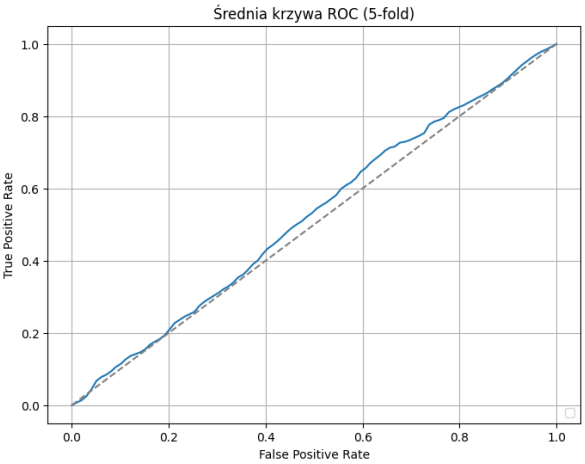
Rysunek 5.4: Średnie krzywe ROC dla nauki bezcechowej



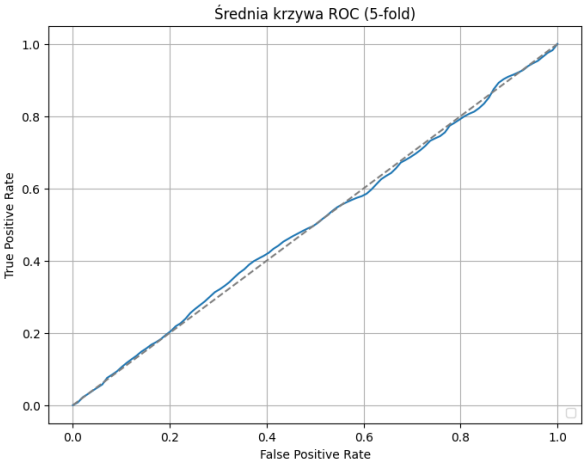
(a) Krzywa ROC sieci gęstej



(b) Krzywa ROC sieci CNN



(c) Krzywa ROC sieci LSTM



(d) Krzywa ROC sieci CNN-LSTM

Optymalizacja hiperparametrów

Tabela 5.4: Wartości miar jakościowych po optymlizacji

	Dokładność	**Spec**	**F-miara**	**nMCC**	**FC**
KNN	0.83 ± 0.00	0.73 ± 0.00	0.87 ± 0.00	0.82 ± 0.00	0.82 ± 0.00
Drzewo Decyzyjne	0.83 ± 0.01	0.77 ± 0.02	0.86 ± 0.00	0.83 ± 0.01	0.83 ± 0.01
Las Losowy	0.84 ± 0.01	0.72 ± 0.03	0.87 ± 0.01	0.83 ± 0.01	0.82 ± 0.01
SVM	0.79 ± 0.01	0.61 ± 0.03	0.84 ± 0.01	0.78 ± 0.01	0.76 ± 0.01

5.3 Analiza wyników

Rozdział 6

Podsumowanie i wnioski

- syntetyczny opis wykonanych prac
- wnioski
- możliwość rozwoju, kontynuacji prac, potencjalne nowe kierunki
- Czy cel pracy zrealizowany?

Bibliografia

[1] Bing Liu. *Web Data Mining*. Springer Berlin / Heidelberg, 2007.

[2] Max Kuhn i Kjell Johnson. *Applied Predictive Modeling*. Springer, 2013.

[3] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt i Gaël Varoquaux. „API design for machine learning software: experiences from the scikit-learn project". W: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, s. 108–122.

[4] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor i J. Vandewalle. *Least Squares Support Vector Machines*. Singapore: World Scientific Pub. Co., 2002. ISBN: 981-238-151-1.

[5] Leslie Valiant. „A Theory of the Learnable". W: *Communications of the ACM* 27 (1984), s. 1134–1142.

[6] Michael Kearns i Leslie Valiant. „Cryptographic Limitations on Learning Boolean Formulae and Finite Automata". W: *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*. 1989.

[7] J. Hertz, A. Krogh i R. G. Palmer. *Wstęp do teorii obliczeń neuronowych*. Warszawa: WNT, 1993.

[8] Bigbossfarin. *Bias and variance contributing to total error*. `https://en.wikipedia.org/wiki/File:Bias_and_variance_contributing_to_total_error.svg`. Own work. CC0 license. Accessed: 2025-07-28. 2021.

[9] B. W. Matthews. „Comparison of the Predicted and Observed Secondary Structure of T4 Phage Lysozyme". W: *Biochimica et Biophysica Acta (BBA) - Protein Structure* 405.2 (1975), s. 442–451. DOI: `10.1016/0005-2795(75)90109-9`.

[10] D. G. Altman i J. M. Bland. „Diagnostic Tests 3: Receiver Operating Characteristic Plots". W: *British Medical Journal* 309.6948 (1994), s. 188. DOI: `10.1136/bmj.309.6948.188`.

[11] C. Brown i H. Davis. „Receiver Operating Characteristics Curves and Related Decision Measures: A Tutorial". W: *Chemometrics and Intelligent Laboratory Systems* 80.1 (2006), s. 24–38.

[12] Tom Fawcett. „An Introduction to ROC Analysis". W: *Pattern Recognition Letters* 27.8 (2006), s. 861–874. DOI: 10.1016/j.patrec.2005.10.010.

[13] Przemysław Solecki. *Krzywe ROC i ocena jakości klasyfikacji*. Dostęp: 29 lipca 2025. Predictive Solutions. 2025. URL: https://predictivesolutions.pl/krzywe-roc-i-ocena-jakosci-klasyfikacji.

[14] David K. McClish. „Analyzing a Portion of the ROC Curve". W: *Medical Decision Making* 9 (1989), s. 190–195. DOI: 10.1177/0272989X8900900307.

[15] MIMIC PERform AF Dataset contributors. *MIMIC PERform AF Dataset*. https://physionet.org/content/mimic-perform-af/. 2020.

[16] Benjamin Moody, George Moody, Mauricio Villarroel, Gari D. Clifford i Ikaro Silva. *MIMIC-III Waveform Database Matched Subset*. https://physionet.org/content/mimic3wdb-matched/1.0/. Wer. 1.0. Published: April 7, 2020. 2020.

[17] Zengding Liu, Bin Zhou, Zhiming Jiang, Xi Chen, Ye Li, Min Tang i Fen Miao. „Multiclass Arrhythmia Detection and Classification From Photoplethysmography Signals Using a Deep Convolutional Neural Network". W: *Journal of the American Heart Association* 11.7 (2022), e023555.

[18] Gari D. Clifford, Ikaro Silva, Benjamin Moody, Chengyu Liu i Roger G. Mark. *PhysioNet/Computing in Cardiology Challenge 2015: Reducing False Arrhythmia Alarms in the ICU*. https://archive.physionet.org/challenge/2015/. Accessed July 2025. Dataset provided as part of the PhysioNet/CinC Challenge 2015. 2015.

[19] A. Solosenko, A. Petrenas, B. Paliakaite, V. Marozas i L. Sornmo. *Model for Simulating ECG and PPG Signals with Arrhythmia Episodes*. 2022. DOI: 10.13026/s32e-sv15. URL: https://doi.org/10.13026/s32e-sv15.

[20] Thomas R. Dawber, H. E. Thomas i P. M. Mcnamara. „Characteristics of the dicrotic notch of the arterial pulse wave in coronary heart disease". W: *Angiology* 24.4 (1973), s. 244–255. DOI: 10.1177/000331977302400407.

[21] Maryam Farokhipour i Farzaneh Ketabchi. „The Validity of Heart Rate Variability Obtained from Electrocardiography and Blood Pressure in Rats Subjected to Isoproterenol-Induced Heart Ischemia". W: *Journal of Tehran University Heart Center* 18.1 (sty. 2023), s. 33–38. DOI: 10.18502/jthc.v18i1.12579. URL: https://doi.org/10.18502/jthc.v18i1.12579.

[22] MathWorks. *resample*. https://www.mathworks.com/help/releases/R2024b/signal/ref/resample.html. Version R2024b. 2025. (Term. wiz. 31. 07. 2025).

[23] Diederik P. Kingma i Jimmy Ba. „Adam: A Method for Stochastic Optimization". W: *arXiv preprint arXiv:1412.6980* (2014). URL: https://arxiv.org/abs/1412.6980.

[24] Sebastian Ruder. „An overview of gradient descent optimization algorithms". W: *arXiv preprint arXiv:1609.04747* (2016). URL: https://arxiv.org/abs/1609.04747.

[25] Jaz Allibhai. *Hold-out vs. Cross-validation in Machine Learning*. Accessed: 2025-08-01. 2018. URL: https://medium.com/@jazallibhai/hold-out-vs-cross-validation-in-machine-learning-92fd1e7b7f1c.

Dodatki

Spis skrótów i symboli

PPG	Sygnał fotopletyzmograficzny
$\bar{x}$	Średnia wartość sygnału
σ	Odchylenie standardowe
H	Entropie Shannona
E	Energia
RMSSD	Pierwiastek średniokwadratowy różnic
kurt	Kurtoza widma
MVC	model – widok – kontroler (ang. *model–view–controller*)
μ	stopnień przyleżności do zbioru
$\mathbb{E}$	zbiór krawędzi grafu
$\mathcal{L}$	transformata Laplace'a

Lista dodatkowych plików, uzupełniających tekst pracy

W systemie do pracy dołączono dodatkowe pliki zawierające:

- źródła programu,
- zbiory danych użyte w eksperymentach,
- film pokazujący działanie opracowanego oprogramowania lub zaprojektowanego i wykonanego urządzenia,
- itp.

Spis rysunków

Spis tabel