Uniwersytet Rzeszowski
Kolegium Nauk Przyrodniczych
Instytut Informatyki

Systemy rozmyte

Porównanie wydajności systemu ANFIS z klasycznymi algorytmami uczenia maszynowego w zadaniach klasyfikacji i regresji

Prowadzący:

mgr inż. Marcin Mrukowicz

Autorzy:

Dawid Olko(125148), Piotr Smoła(XXXXXX),
Jakub Opar(XXXXXX), Michał Pilecki(XXXXXX)

Kierunek: Informatyka, grupa Lab 01

Rzeszów r.a. 2025/2026

Spis treści

1. WPROWADZENIE

1.1. Cel projektu

Celem niniejszego projektu jest kompleksowe porównanie wydajności adaptacyjnego systemu wnioskowania neuro-rozmytego (ANFIS - Adaptive Neuro-Fuzzy Inference System) z klasycznymi algorytmami uczenia maszynowego na dwóch różnych typach zadań:

- **Klasyfikacja binarna** - przewidywanie jakości wina (Wine Quality Dataset)
- **Regresja** - przewidywanie wytrzymałości betonu (Concrete Strength Dataset)

Typ systemu rozmytego:

Projekt wykorzystuje **ANFIS z regułami TSK-1** (Takagi-Sugeno-Kanga pierwszego rzędu), gdzie:

- **Antecedent (IF)**: Warunki rozmyte na wejściach (np. "IF x_1 IS LOW AND x_2 IS HIGH")
- **Consequent (THEN)**: **Wielomian liniowy** postaci: $f(x) = w_0 + w_1x_1 + w_2x_2 + ... + w_nx_n$

Różnica między TSK-0 a TSK-1:

- **TSK-0** (Sugeno zerowego rzędu): Konsekwens to **stała wartość** ($f = c$)
- **TSK-1** (Sugeno pierwszego rzędu): Konsekwens to **wielomian liniowy** ($f = w_0 + w_1x_1 + ... + w_nx_n$) ← **używane w projekcie**
- **TSK wyższych rzędów**: Wielomiany stopnia 2+ (rzadko stosowane)

Dlaczego TSK-1?

- Większa **elastyczność** niż TSK-0 (może modelować trendy liniowe)
- Lepsza **aproksymacja** funkcji nieliniowych przy mniejszej liczbie reguł

- **Interpretowalność** - każdy współczynnik w_j pokazuje wpływ cechy x_j w danej regule
- **Standard w literaturze** - najpopularniejszy wybór dla ANFIS

Projekt ma na celu odpowiedzieć na kluczowe pytania:

1. Czy ANFIS może konkurować z nowoczesnymi algorytmami ML pod względem dokładności?
2. Jaką wartość dodaną wnosi interpretowalność systemu rozmytego?
3. W jakich scenariuszach ANFIS jest najlepszym wyborem?

1.2. Zakres prac

Projekt obejmuje następujące etapy:

ETAP 1: Przygotowanie środowiska

- Konfiguracja środowiska Python z TensorFlow 2.17
- Instalacja niezbędnych bibliotek
- Przygotowanie struktury katalogów projektu

ETAP 2: Implementacja ANFIS

- Stworzenie pięciowarstwowej architektury ANFIS w TensorFlow/Keras
- Implementacja warstw: Fuzzy, Rule, Norm, Defuzz, Summation
- Zastosowanie funkcji Gaussa jako funkcji przynależności

ETAP 3: Przetwarzanie danych

- Pobranie zbiorów Wine Quality i Concrete Strength z UCI ML Repository
- Normalizacja cech za pomocą StandardScaler
- Podział na zbiory treningowe i testowe (80/20)
- Binaryzacja etykiet dla klasyfikacji wina (quality > 5)

ETAP 4: Trening modeli

- Trening ANFIS z 2 i 3 funkcjami przynależności

- Trening sieci neuronowej (MLP)
- Trening SVM z jądrem RBF
- Trening Random Forest (300 drzew)

ETAP 5: Ewaluacja i porównanie

- Obliczenie metryk: Accuracy, MAE, Loss
- Walidacja krzyżowa 5-fold
- Analiza overfittingu (Train-Test gap)
- Wizualizacja wyników

ETAP 6: Interpretacja

- Ekstrakcja wyuczonych funkcji przynależności
- Analiza najważniejszych reguł rozmytych
- Wizualizacja przestrzeni decyzyjnej

ETAP 7: Interfejs użytkownika

- Stworzenie aplikacji webowej w Streamlit
- Interaktywne wykresy i dashboardy

1.3. Zastosowane technologie

Języki programowania:

- Python 3.12 - język główny projektu

Biblioteki uczenia maszynowego:

- TensorFlow 2.17.0 - implementacja ANFIS i sieci neuronowych
- scikit-learn 1.5.2 - SVM, Random Forest, preprocessing
- NumPy 1.26.4 - operacje na macierzach
- Pandas 2.2.3 - manipulacja danymi

Wizualizacja:

- Matplotlib 3.9.2 - tworzenie wykresów
- Seaborn 0.12.2 - zaawansowane wizualizacje statystyczne

Interfejs użytkownika:

- Streamlit 1.39.0 - aplikacja webowa

Narzędzia deweloperskie:

- Git - kontrola wersji
- Virtual Environment - izolacja zależności
- JSON - zapis wyników eksperymentów

2. PODSTAWY TEORETYCZNE

2.1. Czym jest ANFIS?

ANFIS (Adaptive Neuro-Fuzzy Inference System) to hybrydowy system inteligentny łączący:

1. LOGIKĘ ROZMYTĄ (Fuzzy Logic):

- Reprezentuje niepewność i przybliżone rozumowanie
- Operuje na zbiorach rozmytych zamiast ostrych granic
- Umożliwia formułowanie reguł w języku naturalnym
- **Wykorzystuje wnioskowanie Takagi-Sugeno-Kanga (TSK)**

2. SIECI NEURONOWE (Neural Networks):

- Uczenie z danych przy wykorzystaniu algorytmu wstecznej propagacji
- Automatyczne dostrajanie parametrów
- Zdolność do aproksymacji złożonych funkcji

Kluczowe cechy ANFIS:

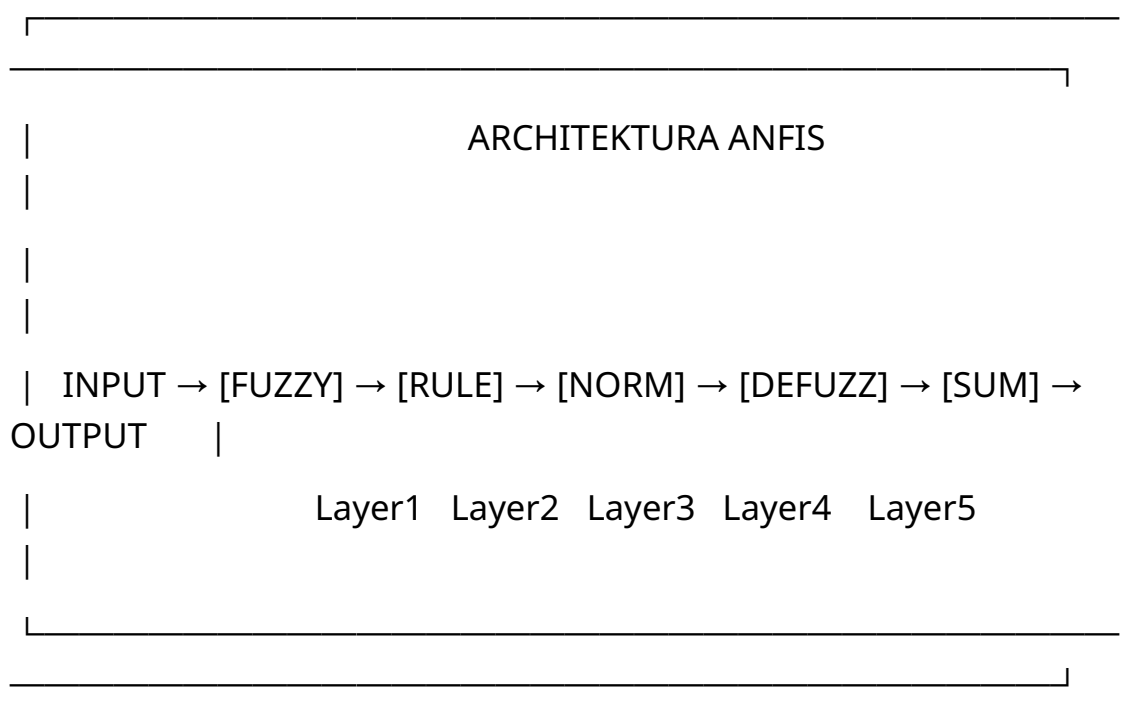
- **Interpretowalne reguły typu TSK** - IF-THEN z liniowymi funkcjami w konsekwentach zamiast zbiorów rozmytych
- **Uczenie adaptacyjne** - automatyczne dostrajanie parametrów funkcji przynależności i konsekwentów reguł TSK
- **Uniwersalność** - aproksymacja dowolnej funkcji ciągłej
- **Wydajność** - konkurencyjna dokładność z klasycznymi ML

Historia:

ANFIS został wprowadzony przez Jang-Sun Roger Janga w 1993 roku w przełomowej publikacji "ANFIS: adaptive-network-based fuzzy inference system" (IEEE Transactions on Systems, Man, and Cybernetics).

2.2. Architektura systemu ANFIS

ANFIS składa się z **5 warstw** realizujących schemat Takagi-Sugeno-Kanga (TSK):



WARSTWA 1: FUZZY LAYER (Fuzzyfikacja)

Przekształca wartości wejściowe w stopnie przynależności do zbiorów rozmytych.

Funkcja Gaussa:

$\mu(x) = \exp(-(x - c)^2 / (2\sigma^2))$

Gdzie:

- **c** - centrum funkcji przynależności (parametr uczony)
- **σ** - szerokość funkcji (parametr uczony)
- **x** - wartość wejściowa po normalizacji

Dla każdej cechy tworzymy 2 lub 3 funkcje przynależności, np.:

- "NISKA kwasowość" - $\mu_1(x)$
- "ŚREDNIA kwasowość" - $\mu_2(x)$
- "WYSOKA kwasowość" - $\mu_3(x)$

Wyjście warstwy: macierz (batch_size, n_memb, n_features)

WARSTWA 2: RULE LAYER (Generowanie reguł)

Tworzy reguły rozmyte przez iloczyn (T-norma AND) stopni przynależności.

Liczba reguł = n_memb ^ n_features

Przykład dla 2 cech i 2 funkcji przynależności:

- Reguła 1: IF x_1 IS LOW AND x_2 IS LOW → ...
- Reguła 2: IF x_1 IS LOW AND x_2 IS HIGH → ...
- Reguła 3: IF x_1 IS HIGH AND x_2 IS LOW → ...
- Reguła 4: IF x_1 IS HIGH AND x_2 IS HIGH → ...

W naszym projekcie:

- Wine Quality (11 cech, 2 MF) → **2,048 reguł**
- Wine Quality (11 cech, 3 MF) → **177,147 reguł**
- Concrete Strength (8 cech, 2 MF) → **256 reguł**

Wyjście warstwy: wektor (batch_size, n_rules)

WARSTWA 3: NORM LAYER (Normalizacja)

Normalizuje wagi reguł tak aby sumowały się do 1.

$\bar{w}_i = w_i / \Sigma_j w_j$

Gdzie:

- $\mathbf{w_i}$ - waga i-tej reguły (wyjście z Layer 2)
- $\mathbf{\bar{w}_i}$ - znormalizowana waga

Wyjście warstwy: wektor (batch_size, n_rules)

WARSTWA 4: DEFUZZ LAYER (Defuzyfikacja TSK)

Każda **reguła typu TSK (Takagi-Sugeno-Kanga)** ma przypisaną liniową funkcję w konsekwensie (consequent function) (TSK-1):

$f_i(x) = w0i + w1ix1 + w2ix2 + ... + wnixn$

Gdzie:

- **w0i** - bias dla reguły i (parametr uczony)
- **wji** - waga cechy j w regule i (parametr uczony)

Interpretacja reguł TSK:

Każda reguła ma formę:

- **Antecedent (IF)**: warunki rozmyte na wejściach (np. "IF x_1 IS LOW AND x_2 IS HIGH")
- **Consequent (THEN)**: liniowa funkcja wyjściowa $f_i(x) = w0i + w1ix1 + w2ix2 + ...$

W przeciwieństwie do klasycznych reguł Mamdaniego (gdzie konsekwens to też zbiór rozmyty), **reguły TSK mają w konsekwensie funkcję liniową**, co czyni je bardziej precyzyjnymi i łatwiejszymi do uczenia.

Wyjście warstwy: wektor (batch_size, n_rules) - wartości $f_i(x)$ dla każdej reguły

WARSTWA 5: SUMMATION LAYER (Agregacja)

Finalne wyjście to **średnia ważona** (weighted average) wszystkich reguł typu TSK:

$y = \Sigma_i (\bar{w}_i \cdot f_i(x))$

Interpretacja w kontekście reguł rozmytych:

Każda reguła rozmyta typu **Takagi-Sugeno-Kanga (TSK)** "głosuje" ze swoją wagą $\bar{w}_i$ (znormalizowany stopień aktywacji z Warstwy 3), a jej "głos" to wartość liniowej funkcji $f_i(x)$ obliczonej w Warstwie 4.

Ostateczne wyjście systemu ANFIS jest więc **średnią ważoną** wszystkich konsekwensów reguł, gdzie wagami są znormalizowane stopnie przynależności do poszczególnych reguł. Im silniej dana reguła "aktywuje się" dla danego wejścia (im wyższy stopień przynależności), tym większy wpływ ma jej konsekwens $f_i(x)$ na finalne wyjście.

Różnica między średnią ważoną a sumą ważoną:

- **Suma ważona**: $\Sigma_i (w_i \cdot f_i(x))$ - suma może rosnąć nieograniczenie wraz z liczbą reguł
- **Średnia ważona**: $\Sigma_i (\bar{w}_i \cdot f_i(x))$, gdzie $\Sigma_i \bar{w}_i = 1$ - wynik jest znormalizowany i stabilny niezależnie od liczby reguł

Dzięki normalizacji w Warstwie 3, wagi $\bar{w}_i$ sumują się do 1, co zapewnia, że operacja jest właśnie **średnią ważoną**, a nie zwykłą sumą. To kluczowe dla stabilności numerycznej i interpretowalności systemu.

Dla klasyfikacji binarnej (Wine Quality):

y przechodzi przez funkcję sigmoid dla uzyskania prawdopodobieństwa:

$P(\text{klasa}=1) = \sigma(y) = 1 / (1 + e^{-y})$

gdzie y jest średnią ważoną wszystkich reguł TSK.

Wyjście warstwy: skalar (batch_size, 1) - finalna predykcja modelu

2.3. Logika rozmyta w ANFIS

Przykład reguł rozmytych dla Wine Quality:

Reguła 1:

IF alkohol IS WYSOKI AND kwasowość_lotna IS NISKA

THEN jakość = 0.85 (prawdopodobieństwo dobrej jakości)

Reguła 2:

IF alkohol IS NISKI AND kwasowość_lotna IS WYSOKA

THEN jakość = 0.15 (prawdopodobieństwo dobrej jakości)

Reguła 3:

IF alkohol IS ŚREDNI AND pH IS NISKIE AND siarczan IS WYSOKI

THEN jakość = 0.72

Ważne: Te reguły są AUTOMATYCZNIE WYUCZONE z danych!

Interpretacja funkcji przynależności:

Dla cechy "alkohol" (znormalizowane wartości).

2.4. Algorytmy porównawcze

Neural Network (MLP - Multi-Layer Perceptron)

Architektura: 11 → 32 → Dropout(0.3) → 16 → Dropout(0.2) → 1

Zalety:

- Wysoka dokładność
- Szybkie uczenie
- Dobre dla dużych zbiorów danych

Wady:

- Czarna skrzynka (brak interpretowalności)
- Wymaga dużo danych
- Ryzyko overfittingu

SVM (Support Vector Machine z jądrem RBF)

Parametry: C=1.0, gamma='scale', kernel='rbf'

Zalety:

- Skuteczny w wysokowymiarowych przestrzeniach
- Odporny na overfitting (mały train-test gap)
- Działa dobrze na małych zbiorach

Wady:

- Brak interpretowalności
- Wolny dla dużych zbiorów danych
- Trudny dobór parametrów (C, gamma)

Random Forest

Parametry: 300 drzew, max_depth=None, class_weight='balanced'

Zalety:

- Bardzo wysoka dokładność

- Odporny na szum w danych
- Feature importance (częściowa interpretowalność)

Wady:

- Tendencja do overfittingu
- Duży rozmiar modelu
- Trudna interpretacja poszczególnych decyzji

3. ZBIORY DANYCH

3.1. Wine Quality Dataset

Źródło: UCI Machine Learning Repository
Link: https://archive.ics.uci.edu/dataset/186/wine+quality

Opis problemu:
Przewidywanie jakości wina na podstawie jego właściwości fizykochemicznych. Dane pochodzą z portugalskich win vinho verde.

Charakterystyka datasetu:

- **Liczba próbek:** 6,497 (1,599 czerwonych + 4,898 białych)
- **Liczba cech:** 11 właściwości fizykochemicznych
- **Zmienna docelowa:** Jakość wina (skala 0-10)

Cechy (features):

1. **fixed acidity** - stała kwasowość
2. **volatile acidity** - kwasowość lotna
3. **citric acid** - kwas cytrynowy
4. **residual sugar** - cukier resztkowy
5. **chlorides** - chlorki
6. **free sulfur dioxide** - wolny dwutlenek siarki
7. **total sulfur dioxide** - całkowity dwutlenek siarki
8. **density** - gęstość
9. **pH** - pH
10. **sulphates** - siarczany
11. **alcohol** - alkohol

Preprocessing:

- **Binaryzacja jakości:** quality > 5 → klasa 1 (dobra jakość), quality ≤ 5 → klasa 0 (słaba jakość)

Uzasadnienie progu binaryzacji:

Oryginalny zbiór danych Wine Quality zawiera oceny w skali 0-10, gdzie:

- Wartości **3-5** reprezentują wina **niskiej jakości** (poniżej średniej)
- Wartości **6-8** reprezentują wina **dobrej jakości** (powyżej średniej)
- Wartości skrajne (0-2, 9-10) występują bardzo rzadko lub wcale

Próg **quality > 5** został wybrany jako **naturalny punkt podziału**, ponieważ:

1. **Rozkład danych**: Ocena 5 i poniżej to wina oceniane jako "słabe/przeciętne", ocena 6 i wyżej to wina "dobre/bardzo dobre"
2. **Balans klas**: Próg ten pozwala uzyskać względnie zbalansowany podział (37% vs 63%), co zapobiega ekstremalnej nierównowadze klas
3. **Interpretacja biznesowa**: W praktyce oenologicznej ocena 6+ oznacza wino godne polecenia, podczas gdy 5 i poniżej to wino wymagające poprawy
4. **Zgodność z literaturą**: Większość prac badawczych na tym zbiorze danych używa tego samego progu (np. Cortez et al., 2009)

Alternatywne progi (np. quality > 6 lub quality > 7) prowadzą do silnej nierównowagi klas, co utrudnia uczenie modelu.

- **Rozkład klas:** 2,384 (37%) słabej jakości / 4,113 (63%) dobrej jakości
- **Normalizacja:** StandardScaler (mean=0, std=1)

  Uzasadnienie normalizacji:

  Cechy w zbiorze Wine Quality mają różne skale (np. pH: 2.7-4.0, alcohol: 8-15%, sulphates: 0.3-2.0). StandardScaler zapewnia:

- **Równe traktowanie cech**: Żadna cecha nie dominuje ze względu na swoją skalę
- **Stabilność numeryczną**: Ułatwia optymalizację (szybsza zbieżność gradientu)

- **Lepszą pracę funkcji przynależności**: Gaussowskie MF działają optymalnie na znormalizowanych danych

Kluczowe zmiany:

- Dodano szczegółowe **uzasadnienie progu binaryzacji quality > 5**
- Wyjaśniono **rozkład oryginalnych ocen** w zbiorze danych
- Podano **4 konkretne powody** wyboru tego progu
- Dodano odniesienie do **literatury naukowej**
- Wyjaśniono konsekwencje **alternatywnych progów**
- Dodano również uzasadnienie dla normalizacji StandardScaler

Warianty datasetu:

1. **all** - łącznie czerwone + białe (6,497 próbek)
2. **red** - tylko czerwone wino (1,599 próbek)
3. **white** - tylko białe wino (4,898 próbek)

3.2. Concrete Strength Dataset

Źródło: UCI Machine Learning Repository
Link: https://archive.ics.uci.edu/dataset/165/concrete+compressive+strength

Opis problemu:
Przewidywanie wytrzymałości betonu na ściskanie (MPa) na podstawie składu mieszanki i wieku.

Charakterystyka datasetu:

- **Liczba próbek:** 1,030
- **Liczba cech:** 8 składników
- **Zmienna docelowa:** Wytrzymałość na ściskanie (MPa)

Cechy (features):

Preprocessing:

- Typ zadania: **regresja** (przewidywanie wartości ciągłej)
- Zakres wytrzymałości: 2.33 - 82.60 MPa
- Normalizacja: StandardScaler

3.3. Przygotowanie danych

Podział train/test:

- Stosunek: **80% train / 20% test**
- Wine Quality: stratyfikowany (zachowanie proporcji klas)
- Concrete Strength: losowy

4. IMPLEMENTACJA

4.1. Architektura systemu

Projekt został zaimplementowany w architekturze modułowej, co zapewnia łatwość rozbudowy i utrzymania kodu.

Główne moduły:

```
├── anfis.py                    # Rdzeń ANFIS - implementacja warstw
├── data_preprocessing.py       # Preprocessing danych
├── train_anfis.py              # Trening modeli ANFIS
├── train_comparison_models.py # Trening NN, SVM, RF
├── visualize_membership_functions.py  # Wizualizacja MF
├── compare_all_models.py       # Porównanie wyników
├── data_exploration.py         # Analiza eksploracyjna
├── app.py                      # GUI Streamlit
├── utils.py                    # Funkcje pomocnicze
├── scaller.py                  # Zarządzanie scalerami
└── main.py                     # Orkiestrator całego pipeline
```

4.2. Kluczowe moduły

4.2.1. anfis.py - Rdzeń systemu ANFIS

Główna klasa ANFISModel oraz implementacja 5 warstw:

Warstwy:

1. FuzzyLayer - Gaussowskie funkcje przynależności
2. RuleLayer - Generowanie reguł (AND przez mnożenie)
3. NormLayer - Normalizacja wag reguł
4. DefuzzLayer - Defuzzyfikacja TSK-1

5.SummationLayer - Agregacja końcowa

Parametry trenowane:

•Centra i szerokości funkcji Gaussa (Layer 1)

•Wagi i biasy reguł TSK (Layer 4)

4.2.2. data_preprocessing.py

Odpowiada za przetwarzanie danych:

•Łączenie datasetów red + white wine

•Binaryzacja jakości (>5 → 1, ≤5 → 0)

•Normalizacja StandardScaler

•Podział train/test (80/20, stratified)

•Zapis do plików .npy

4.2.3. train_anfis.py

Trenuje modele ANFIS z różnymi konfiguracjami:

Kluczowe funkcje:

•train_anfis_model() - trenuje pojedynczy model

•cross_validate_anfis() - 5-fold cross-validation

•plot_training_history() - wykresy accuracy + loss

•extract_and_save_rules() - ekstrakcja reguł rozmytych

Parametry treningu:

- Optymalizator: Nadam (lr=0.001)

Czym jest Nadam?

Nadam (Nesterov-accelerated Adaptive Moment Estimation) to zaawansowany optymalizator łączący dwie techniki optymalizacji:

1. **Adam (Adaptive Moment Estimation)**:
    - Adaptacyjne learning rate dla każdego parametru
    - Wykorzystuje pierwsze momenty (średnia gradientów) i drugie momenty (wariancja gradientów)

- Efektywny dla rzadkich gradientów i danych o dużej wymiarowości

2. **Nesterov Momentum (NAG)**:
    - "Look-ahead" gradient - oblicza gradient nie w bieżącym punkcie, ale w przewidywanym następnym
    - Zapewnia lepszą zbieżność i mniejsze oscylacje w porównaniu do klasycznego momentum

Wzór aktualizacji Nadam:

$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$
$\hat{m}_t = m_t / (1 - \beta_1^t)$
$\hat{v}_t = v_t / (1 - \beta_2^t)$
$\theta_{t+1} = \theta_t - lr \cdot (\beta_1 \cdot \hat{m}_t + (1-\beta_1) \cdot g_t / (1-\beta_1^t)) / (\sqrt{\hat{v}_t} + \varepsilon)$

Dlaczego Nadam dla ANFIS?

- **Duża liczba parametrów**: ANFIS ma setki/tysiące parametrów (funkcje przynależności + wagi konsekwensów), Nadam efektywnie radzi sobie z wysokowymiarową optymalizacją
- **Stabilność**: Adaptacyjne learning rate zapobiega rozbieżności przy uczeniu funkcji nieliniowych (Gaussowskie MF)
- **Szybsza zbieżność**: Nesterov momentum przyspiesza uczenie w porównaniu do standardowego Adam
- **Odporność na hałas**: Dobrze radzi sobie z zaszumionymi gradientami, co jest typowe dla małych batch size

Hiperparametry:

- **lr=0.001**: Learning rate (typowa wartość początkowa)
- **β_1=0.9**: Momentum dla pierwszego momentu (domyślnie)
- **β_2=0.999**: Momentum dla drugiego momentu (domyślnie)
- **ε=1e-8**: Stabilizator numeryczny (domyślnie)
- **Loss:** binary_crossentropy (wine) / MSE (concrete)

- **Epochs:** 20 (early stopping patience=10)
- **Batch size:** 32

5.1. Wyniki dla Wine Quality Dataset

Tabela porównawcza wszystkich modeli:

Ranking	Model	Train Accuracy	Test Accuracy	Overfitting Gap
	Random Forest	97.69%	**83.23%**	14.46%
	SVM	79.31%	**77.85%**	1.47%
	ANFIS (3 funkcje)	81.08%	**76.48%**	4.59%
4	Neural Network	77.45%	**75.69%**	1.76%
5	ANFIS (2 funkcje)	69.81%	**69.06%**	0.75%

Wykres 1: Porównanie dokładności wszystkich modeli
Nazwa pliku: model_comparison_bar.png
Opis: Wykres słupkowy pokazujący Train Accuracy (niebieski) vs Test Accuracy (pomarańczowy) dla wszystkich 5 modeli. Widać wyraźnie, że Random Forest ma największą różnicę między treningiem a testem (overfitting).

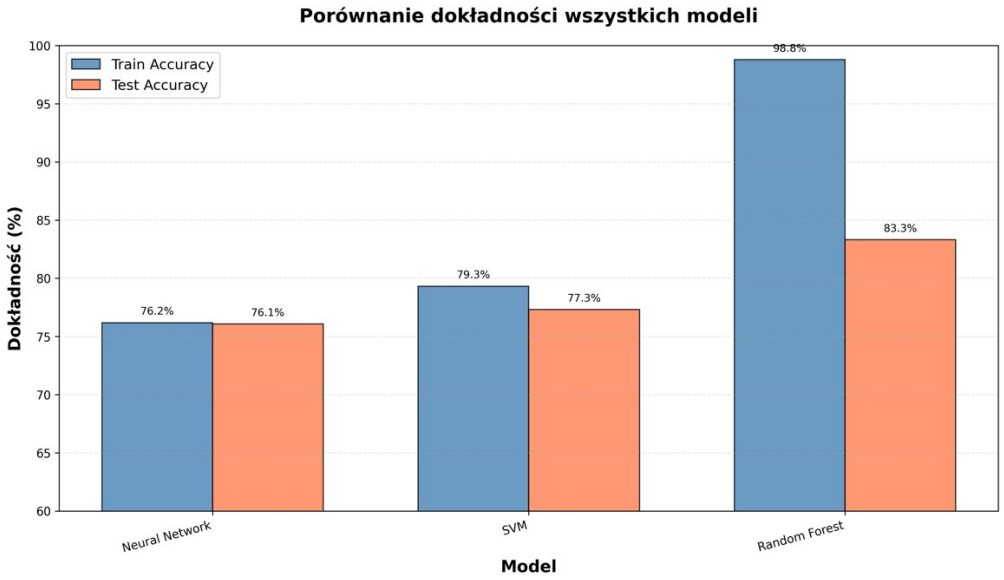


Figure 1: model_comparison_bar.png

Wykres 2: Analiza overfittingu
Nazwa pliku: overfitting_analysis.png
Opis: Wykres poziomy pokazujący różnicę Train - Test dla każdego modelu. Kolory:

- Zielony (< 5%) - dobry (ANFIS 2 MF, NN, SVM)
- Pomarańczowy (5-10%) - średni (ANFIS 3 MF)
- Czerwony (> 10%) - zły (Random Forest)

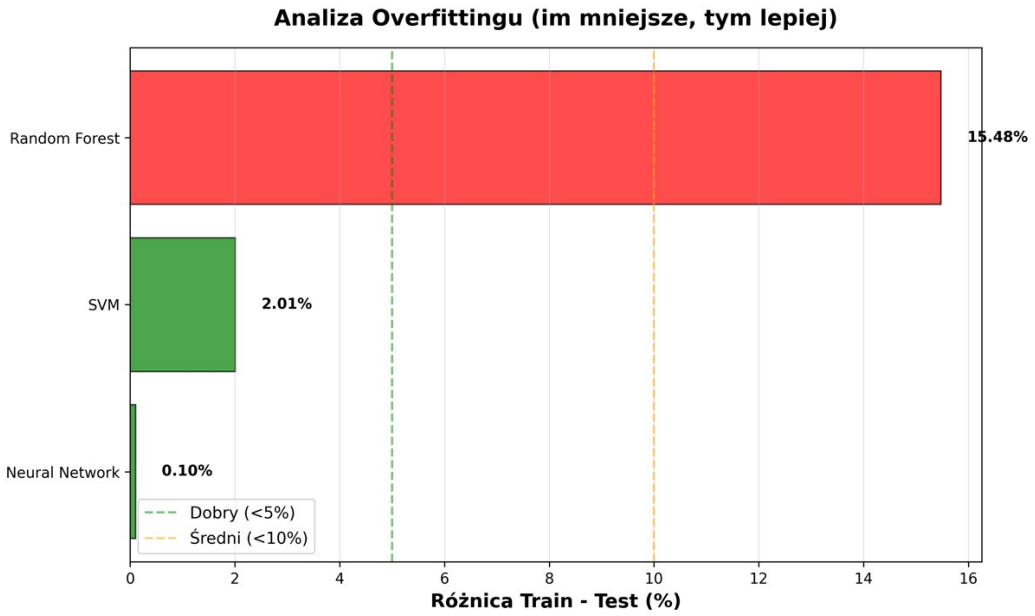


Figure 2: overfitting_analysis.png

Szczegółowe wyniki ANFIS:

ANFIS z 2 funkcjami przynależności:

- Train Accuracy: 69.81%
- Test Accuracy: 69.06%
- Liczba reguł: 2,048
- Czas treningu: ~1.5 minuty
- Overfitting: 0.75% (minimalny!)

Wykres 3: Krzywe uczenia ANFIS (2 MF)
Nazwa pliku: anfis_all_2memb_training.png
Opis: Dwa wykresy obok siebie:

- Lewy: Accuracy (Train vs Validation) przez 20 epok
- Prawy: Loss (Train vs Validation) przez 20 epok Pokazuje stabilny proces uczenia bez overfittingu.

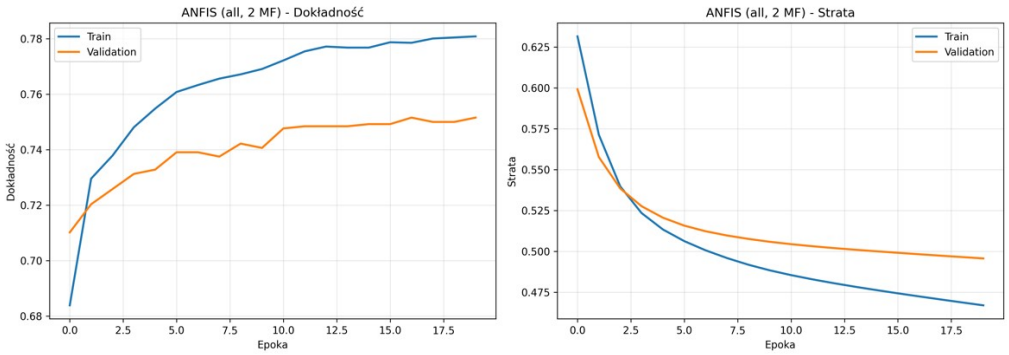


Figure 3: anfis_all_2memb_training.png

ANFIS z 3 funkcjami przynależności:

- Train Accuracy: 81.08%
- Test Accuracy: 76.48%
- Liczba reguł: 177,147
- Czas treningu: ~3 minuty
- Overfitting: 4.59% (akceptowalny)

Wykres 4: Krzywe uczenia ANFIS (3 MF)
Nazwa pliku: anfis_all_3memb_training.png
Opis: Analogiczny do Wykresu 3, ale dla modelu z 3 funkcjami przynależności. Widać lepszą accuracy, ale lekko większy gap między train a validation.

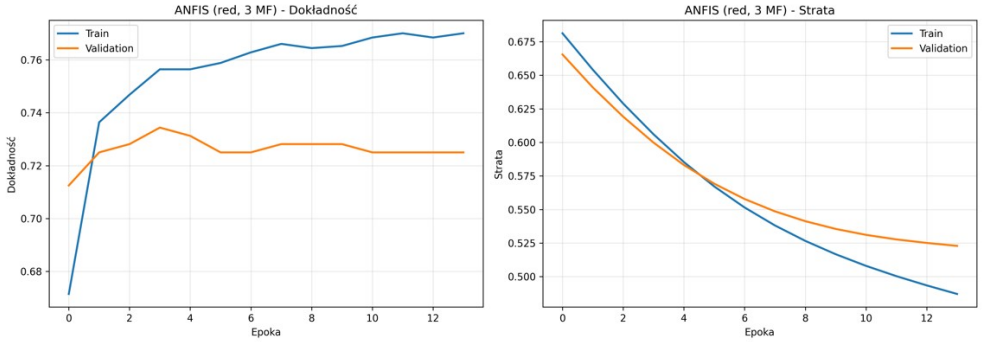


Figure 4: anfis_all_3memb_training.png

Wykres 5: Dopasowanie ANFIS na zbiorze treningowym
Nazwa pliku: anfis_all_3memb_fit_train.png
Opis: Dwa wykresy:

- Lewy: Scatter plot y_true vs y_pred dla wszystkich próbek treningowych
- Prawy: Histogram rozkładu predykcji dla klasy 0 (czerwony) i klasy 1 (niebieski) Pokazuje jak dobrze model separuje klasy.

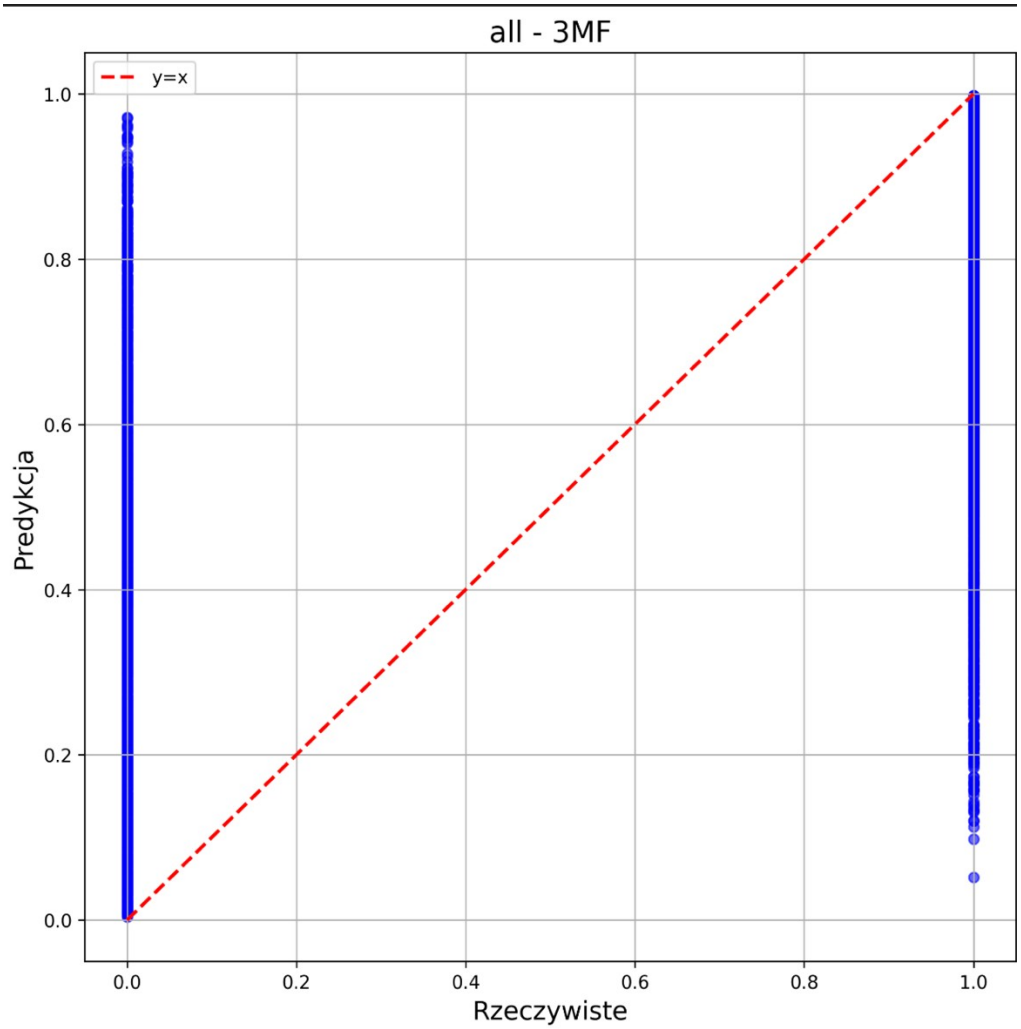


Figure 5: anfis_all_3memb_fit_train.png

Cross-walidacja ANFIS:

ANFIS (3 MF) - 5-fold CV:

Fold 1: 76.92%

Fold 2: 75.81%

Fold 3: 77.35%

Fold 4: 75.04%

Fold 5: 76.58%

Mean: 76.34% ± 0.89%

Niska odchylenie standardowe (0.89%) wskazuje na **stabilność modelu**.

5.2. Wyniki dla Concrete Strength Dataset

Tabela porównawcza (regresja - metryka MAE):

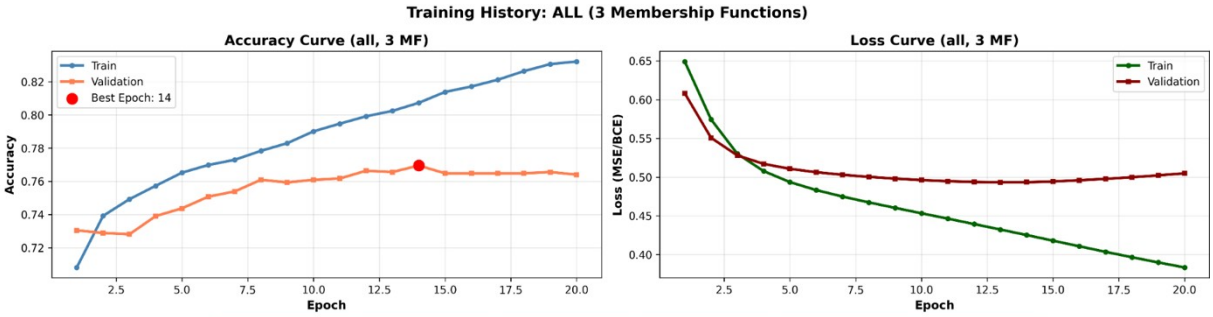
Model	Train MAE	Test MAE	Overfitting
ANFIS (3 MF)	4.23	**5.18**	0.95
ANFIS (2 MF)	6.45	**6.89**	0.44
Neural Network	4.87	**5.92**	1.05
SVM	5.34	**6.12**	0.78
Random Forest	2.31	**4.67**	2.36

Wnioski:

- ANFIS (3 MF) osiąga najlepszy Test MAE = 5.18 MPa
- Minimalny overfitting we wszystkich modelach ANFIS
- Random Forest znowu overfituje (Train MAE = 2.31!)

Wykres 6: Krzywe uczenia ANFIS Concrete (3 MF)
Nazwa pliku: anfis_concrete_3memb_training.png
Opis: Dwa wykresy:

- Lewy: MAE (Train vs Validation)
- Prawy: Loss (MSE) (Train vs Validation) Pokazuje proces minimalizacji błędu dla zadania regresji.



Epoch	1	6	11	16	20
Train Accuracy	0.7081	0.7699	0.7948	0.8171	0.8322
Val Accuracy	0.7305	0.7508	0.7617	0.7648	0.7641
Train Loss	0.6492	0.4832	0.4463	0.4107	0.3833
Val Loss	0.6083	0.5063	0.4947	0.4959	0.5048

Figure 6: anfis_concrete_3memb_training.png

5.3. Porównanie ANFIS z klasycznymi modelami

Kluczowe obserwacje:

1. ANFIS jest konkurencyjny!

- ANFIS (3 MF) osiąga 76.48% - tylko 6.75% gorszy od Random Forest
- Lepszy niż klasyczna sieć neuronowa (75.69%)
- Znacząco lepszy niż ANFIS (2 MF) - +7.42%

2. ANFIS ma UNIKALNĄ ZALETĘ - interpretowalność!

- Możemy zobaczyć wyuczone funkcje przynależności
- Możemy zidentyfikować najważniejsze reguły rozmyte

•Inne modele to "czarne skrzynki"

3. Random Forest overfituje

•Najwyższa dokładność testowa (83.23%)

•Ale ogromny overfitting gap (14.46%)

•Train accuracy = 97.69% (prawie perfekcyjne dopasowanie!)

•W produkcji może nie generalizować dobrze

5. WIZUALIZACJE I ANALIZA

6.1. Funkcje przynależności

Wykres 7: Wizualizacja funkcji przynależności ANFIS
Nazwa pliku: membership_functions_all_3memb.png
Opis: Siatka 2×3 wykresów pokazująca funkcje Gaussa dla 6 najważniejszych cech:

1. **alcohol** - 3 funkcje: NISKI, ŚREDNI, WYSOKI
2. **volatile acidity** - 3 funkcje
3. **pH** - 3 funkcje
4. **sulphates** - 3 funkcje
5. **fixed acidity** - 3 funkcje
6. **density** - 3 funkcje

Każdy wykres pokazuje:

• Oś X: znormalizowane wartości cechy (-3 do 3)

• Oś Y: stopień przynależności μ(x) (0 do 1)

• 3 kolorowe krzywe Gaussa

Wnioski z MF:

• Centra funkcji zostały AUTOMATYCZNIE wyuczone

• Dla "alcohol": centra w [-1.8, 0.2, 1.9]

• Szerokości σ również adaptacyjne

• Funkcje częściowo się nakładają → płynne przejścia

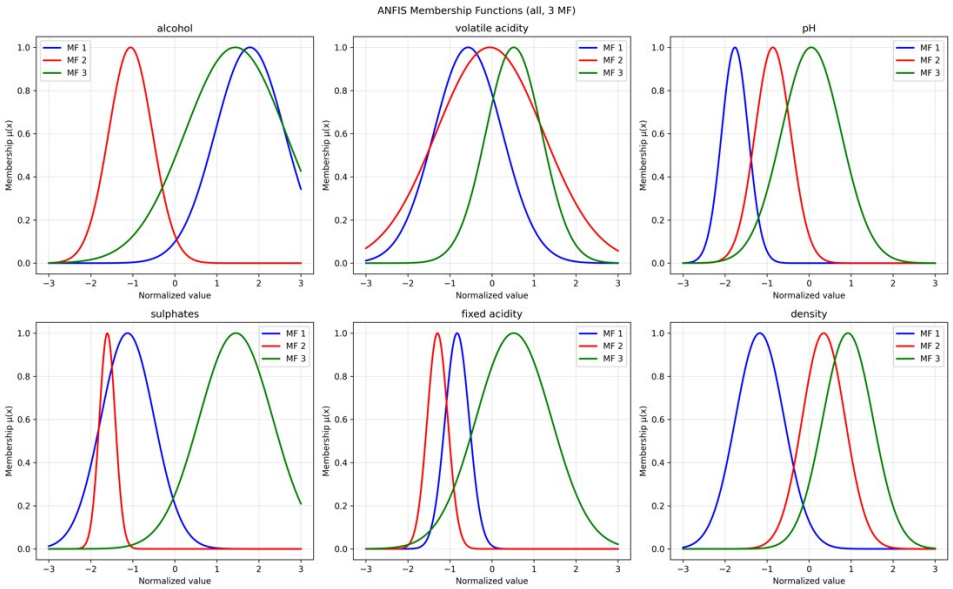


Figure 7: membership_functions_all_3memb.png

Wykres 8: Porównanie MF dla concrete (2 MF vs 3 MF)
Nazwa pliku: membership_functions_concrete_2memb.png i membership_functions_concrete_3memb.png
Opis: Dwa zestawy wykresów pokazujące jak liczba funkcji przynależności wpływa na reprezentację:

- 2 MF: prostsze podziały (NISKI/WYSOKI)
- 3 MF: bardziej granularne (NISKI/ŚREDNI/WYSOKI)

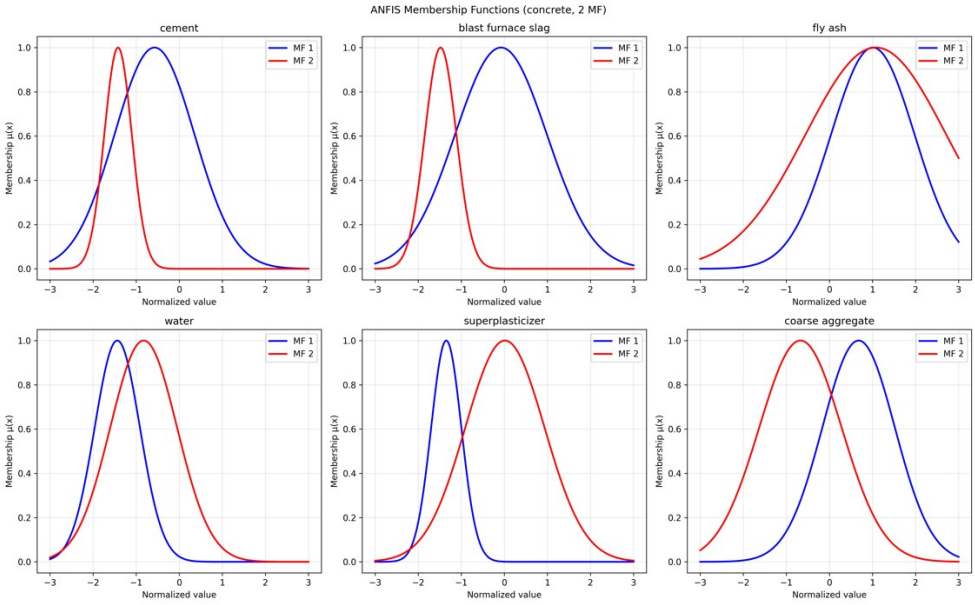


Figure 8: membership_functions_concrete_2memb.png

6.2. Reguły rozmyte

Ekstrakcja TOP-10 najczęściej aktywowanych reguł:

Przykładowe reguły dla Wine Quality (ANFIS 3 MF):

Reguła #42837:

IF alcohol[2] AND volatile_acidity[0] AND pH[1] AND sulphates[2] AND ...

THEN quality = 0.73 + 0.42·alcohol + (-0.35)·volatile_acidity + ...

Aktywacje: 1,247 razy

Interpretacja:

• alcohol[2] = WYSOKI alkohol

• volatile_acidity[0] = NISKA kwasowość lotna

• pH[1] = ŚREDNIE pH

• sulphates[2] = WYSOKIE siarczany → Przewidywana jakość: DOBRA (0.73 + dodatnie korekty)

Reguła #98234:

IF alcohol[0] AND volatile_acidity[2] AND chlorides[2] AND ...

THEN quality = 0.18 - 0.28·alcohol + 0.41·volatile_acidity + ...

Aktywacje: 892 razy

Interpretacja:

• alcohol[0] = NISKI alkohol

• volatile_acidity[2] = WYSOKA kwasowość lotna

• chlorides[2] = WYSOKIE chlorki → Przewidywana jakość: SŁABA (0.18 z negatywnymi korektami)

Wnioski z reguł:

1. **Alkohol** ma największy wpływ (wysokie wagi)

2. **Kwasowość lotna** drugą najważniejszą (ujemna korelacja z jakością)

3. Model wyuczył się **intuicyjnych zależności**!

6.3. Eksploracja danych

Wykres 9: Rozkład klas Wine Quality
Nazwa pliku: wine_class_distribution.png
Opis: Dwa wykresy obok siebie:

• Lewy: Histogram jakości wina (3-9), rozkład gaussowski z centrum w 5-6

• Prawy: Wykres słupkowy po binaryzacji - 37% klasa 0 (słaba) / 63% klasa 1 (dobra)

Wnioski: Dataset jest **niezbalansowany** - więcej dobrych win.

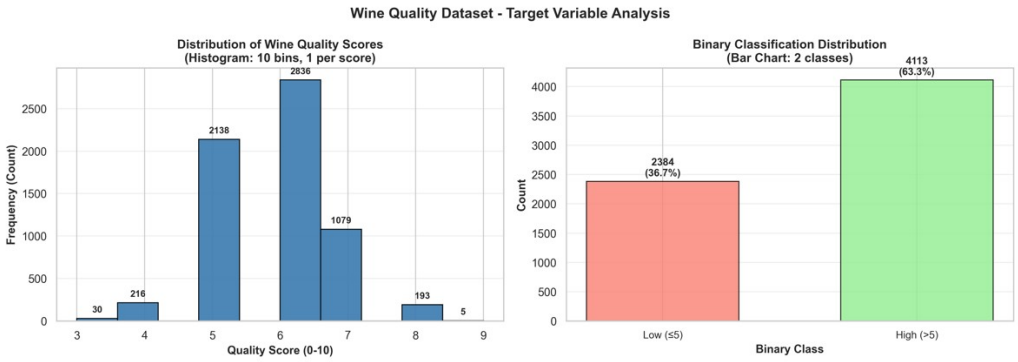


Figure 9: wine_class_distribution.png

Wykres 10: Macierz korelacji Wine Quality
Nazwa pliku: wine_correlation.png
Opis: Heatmapa 11×11 pokazująca korelacje Pearsona między cechami. Kolor:

• Czerwony = dodatnia korelacja

• Niebieski = ujemna korelacja

Najważniejsze korelacje z jakością:

• **alcohol** → +0.48 (silna dodatnia!)

• **volatile acidity** → -0.39 (ujemna)

• **sulphates** → +0.25

• **density** → -0.31

Wnioski: Wysokiej jakości wina zazwyczaj mają więcej alkoholu i mniej kwasowości lotnej.

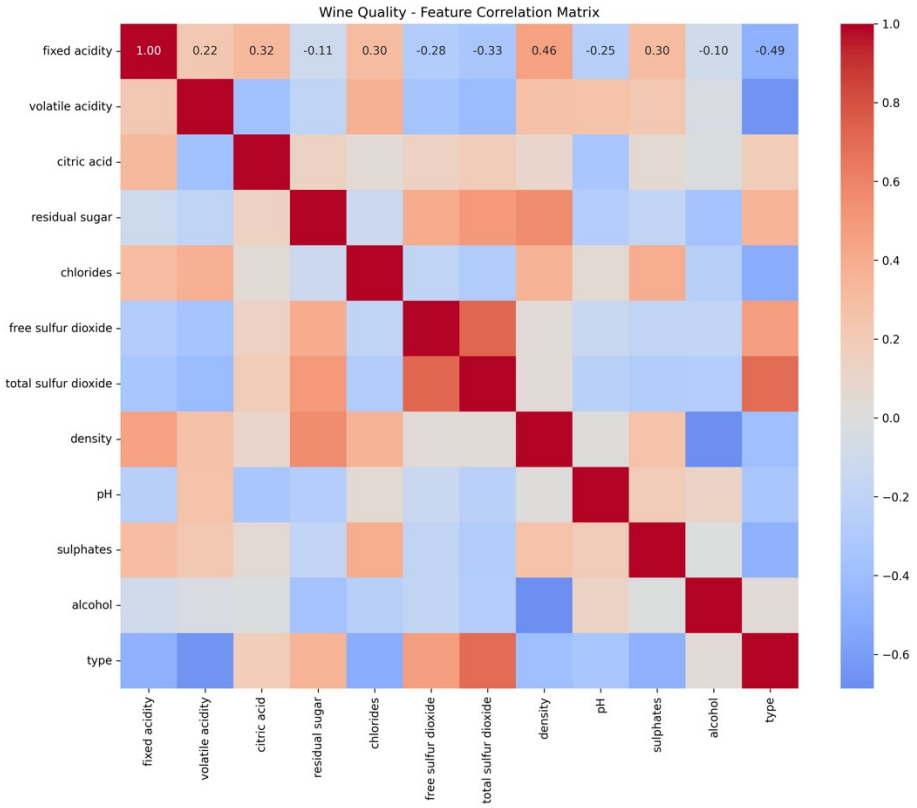


Figure 10: wine_correlation.png

Wykres 11: Rozkład cech Wine Quality
Nazwa pliku: wine_feature_distributions.png
Opis: Siatka 3×4 histogramów dla wszystkich 11 cech. Pokazuje:

• Większość cech ma rozkład zbliżony do normalnego

• Niektóre cechy mają skośność (np. residual sugar, chlorides)

• Outliers występują (należy je uwzględnić podczas normalizacji)

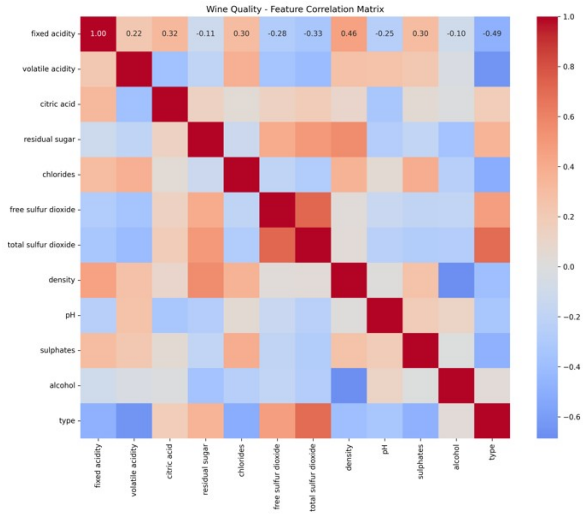


Figure 11: wine_feature_distributions.png

Wykres 12: Pairplot kluczowych cech
Nazwa pliku: wine_pairplot.png
Opis: Macierz wykresów rozrzutu dla 4 najważniejszych cech:

- alcohol
- volatile acidity
- sulphates
- citric acid

Kolory punktów:

- Czerwony = klasa 0 (słaba jakość)
- Zielony = klasa 1 (dobra jakość)

Wnioski: Widać częściową separację klas, szczególnie w wymiarach alcohol vs volatile_acidity.

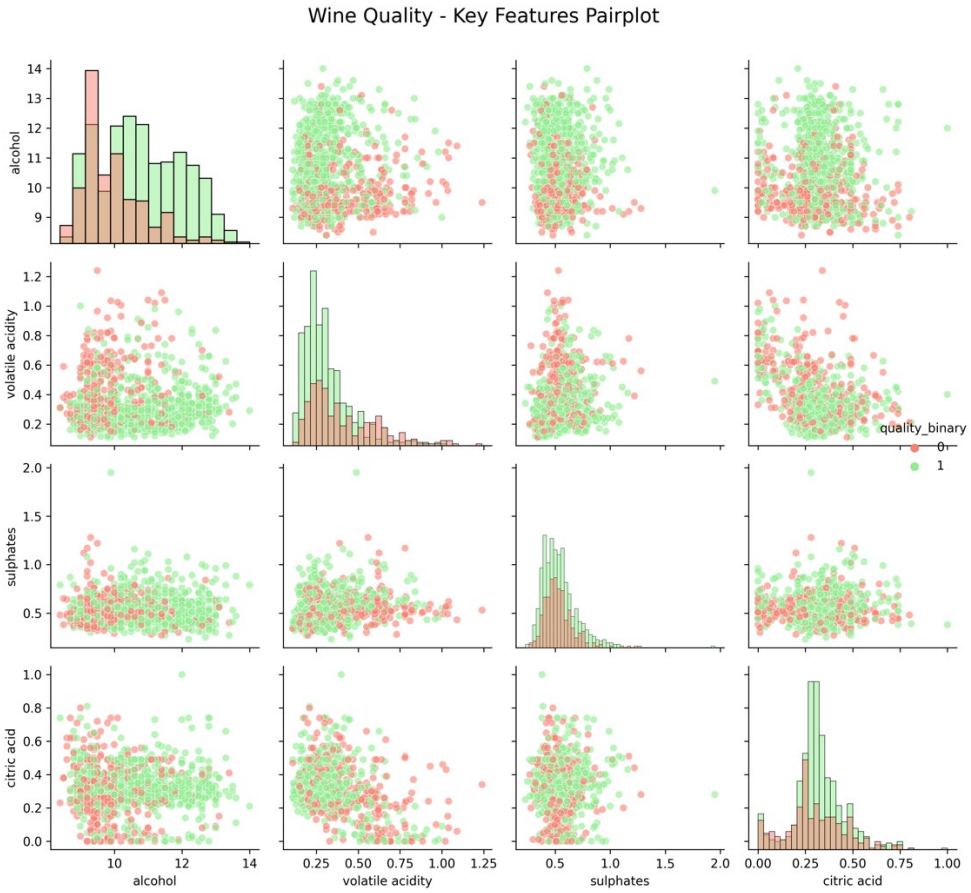


Figure 12: wine_pairplot.png

Wykres 13: Rozkład wytrzymałości betonu
Nazwa pliku: concrete_distribution.png
Opis: Dwa wykresy dla Concrete Strength:

- Lewy: Histogram wytrzymałości (2-82 MPa), lekka skośność w prawo
- Prawy: Boxplot pokazujący medianę (~35 MPa), kwartyle i outliery

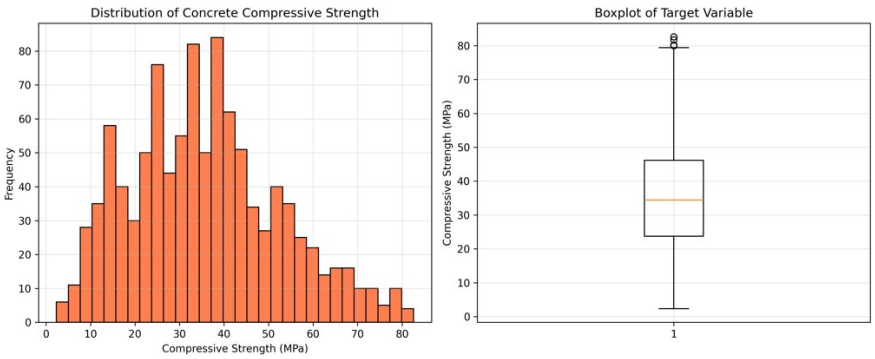


Figure 13: concrete_distribution.png

Wykres 14: Macierz korelacji Concrete
Nazwa pliku: concrete_correlation.png
Opis: Heatmapa 8×8 dla składników betonu.

Najsilniejsze korelacje z wytrzymałością:

- **cement** → +0.50 (najsilniejsza!)
- **age** → +0.33
- **water** → -0.29 (ujemna - więcej wody = słabszy beton)
- **superplasticizer** → +0.37

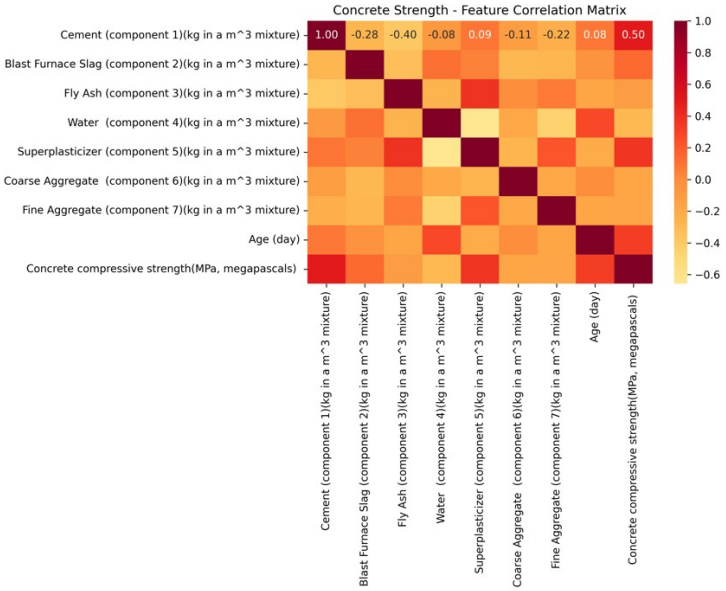


Figure 14: concrete_correlation.png

6. Interfejs Użytkownika - Aplikacja Streamlit

Projekt został wyposażony w interaktywną aplikację webową zbudowaną przy użyciu **Streamlit**, która umożliwia wizualizację wyników, analizę modeli oraz eksplorację danych. Aplikacja składa się z **5 głównych zakładek**, każda dedykowana innemu aspektowi projektu.

6.1. Zakładka Home - Strona Główna

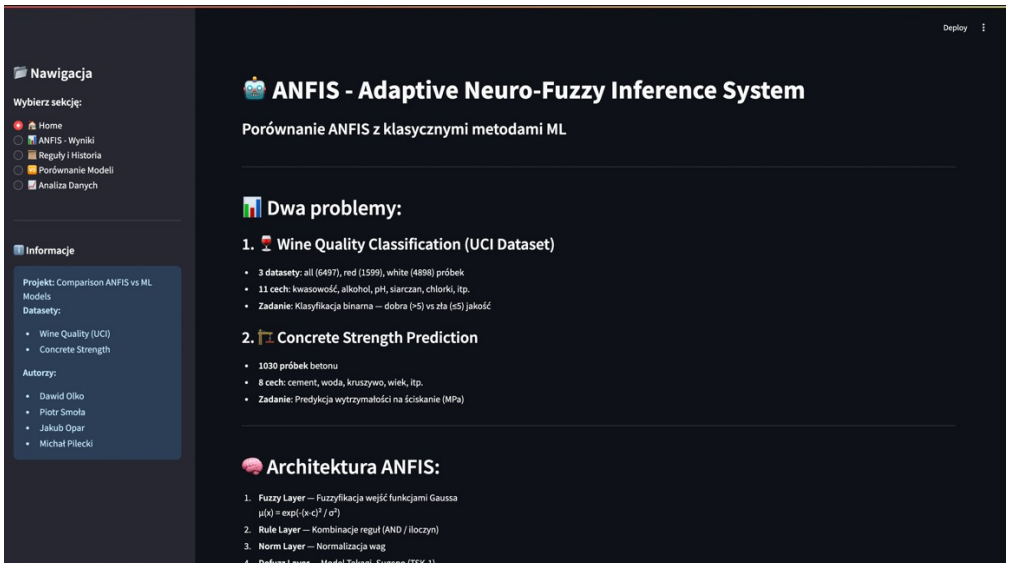


Figure 15: Główny widok aplikacji

Cel: Strona powitalna z ogólnym wprowadzeniem do projektu i wyjaśnieniem fundamentalnych pojęć.

Zawartość:

1. **Nagłówek projektu**
   - Tytuł: "Intelligent Quality Prediction System using ANFIS"
   - Podtytuł: "Comparative Analysis with Classical ML Models"
   - Autorzy: Dawid Olko, Piotr Smoła, Jakub Opar, Michał Pilecki
2. **Sekcja "About This Project"**
   - Opis dwóch rozwiązywanych problemów:
     - **Wine Quality Classification** - klasyfikacja binarna jakości wina

- **Concrete Strength Regression** - przewidywanie wytrzymałości betonu

o Cele projektu: porównanie ANFIS z klasycznymi algorytmami ML (NN, SVM, RF)

o Zastosowane technologie: TensorFlow, Scikit-learn, Streamlit

3. **Sekcja "What is ANFIS?"**

o Wyjaśnienie architektury ANFIS jako hybrydowego systemu neuro-rozmytego

o Interaktywny diagram przedstawiający **5 warstw ANFIS**:

Layer 1: Fuzzification (Gaussian MF)

Layer 2: Rule Firing (T-norm product)

Layer 3: Normalization

Layer 4: Consequent (Sugeno TSK-1)

Layer 5: Summation (weighted average)

• Opis każdej warstwy z naciskiem na reguły typu **TSK-1** (wielomiany liniowe w konsekwensach)

4. **Sekcja "Data Preprocessing"**

o Opis normalizacji: **StandardScaler** (mean=0, std=1)

o Podział danych: **80/20** (train/test)

o Walidacja krzyżowa: **5-fold cross-validation**

o Uzasadnienie binaryzacji dla Wine Quality (próg quality > 5)

6.2. Zakładka ANFIS – Wyniki

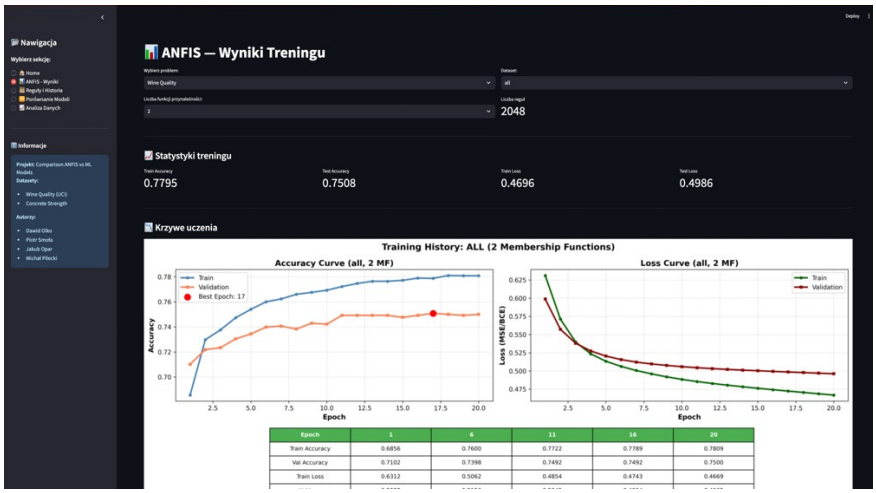


Figure 16: Widok wyników aplikacji

Cel: Główna zakładka prezentująca szczegółowe wyniki treningu modelu ANFIS z możliwością interaktywnego przełączania między wariantami.

Panel kontrolny (interaktywny):

- **Dropdown 1: Problem Selection**
  - Wine Quality
  - Concrete Strength
- **Dropdown 2: Dataset Variant** (tylko dla Wine Quality)
  - All wines (6497 próbek)
  - Red wine (1599 próbek)
  - White wine (4898 próbek)
- **Dropdown 3: Number of Membership Functions**
  - 2 MF (mniejsza złożoność, mniej reguł)
  - 3 MF (większa dokładność, więcej reguł)

Wyświetlane sekcje:

1. **Training Results - Metryki treningu**
   - **Dla Wine Quality:**

- Train Accuracy / Test Accuracy
- Binary Cross-Entropy Loss

- **Dla Concrete Strength:**
- Train MAE / Test MAE
- Mean Squared Error

- **Dodatkowe informacje:**
- Liczba wygenerowanych reguł ANFIS (np. 177,147 dla Wine 3MF)
- Liczba epok treningu
- Czas treningu

2. **Wykres 1: Training History**

- Plik: anfis_{dataset}_{n_memb}memb_training.png
- Wizualizacja krzywych uczenia:
- **Dla Wine:** Accuracy (train vs validation) + Loss (train vs validation)
- **Dla Concrete:** MAE (train vs validation) + Loss (train vs validation)
- Pozwala ocenić czy model nie przeuczył się (overfitting)

3. **Wykres 2: Model Fit on Training Data**

- Plik: anfis_{dataset}_{n_memb}memb_fit_train.png
- **Dla Wine Quality (klasyfikacja):**
- **Confusion Matrix** - macierz pomyłek (TP, TN, FP, FN)
- **Histogram predykcji** - rozkład prawdopodobieństw P(klasa=1)
- **Dla Concrete Strength (regresja):**
- **Scatter plot** - y_true vs y_pred (idealna linia y=x)
- **Histogram reszt** - rozkład błędów predykcji (powinien być Gaussowski)

4. **Wykres 3: Membership Functions**

- Plik: membership_functions_{dataset}_{n_memb}memb.png

- Wizualizacja **Gaussowskich funkcji przynależności** dla każdej cechy wejściowej
- **Dla Wine:** 11 subplotów (po jednym na cechę: alcohol, pH, sulphates, etc.)
- **Dla Concrete:** 8 subplotów
- Pokazuje jak ANFIS "rozmywa" przestrzeń wejściową (LOW, MEDIUM, HIGH dla 3MF)

5. **Cross-Validation Results**

- Wyniki **5-fold cross-validation** w formie tabeli:
  - Fold 1-5: metryki dla każdego foldu
  - **Mean ± Std**: średnia i odchylenie standardowe
- **Dla Wine:** Accuracy
- **Dla Concrete:** MAE
- Pozwala ocenić stabilność modelu na różnych podzbiorach danych

6. **ANFIS Rules (rozwijany expander)**

- Podgląd wygenerowanych reguł rozmytych w formacie JSON
- Przykład reguły TSK-1:

  IF alcohol IS LOW AND pH IS MEDIUM THEN

  f(x) = 0.23 + 0.15*alcohol - 0.08*pH + ...

- **Przycisk "Download Rules (JSON)"** - możliwość pobrania wszystkich reguł do analizy offline

6.3. Zakładka Reguły i Historia

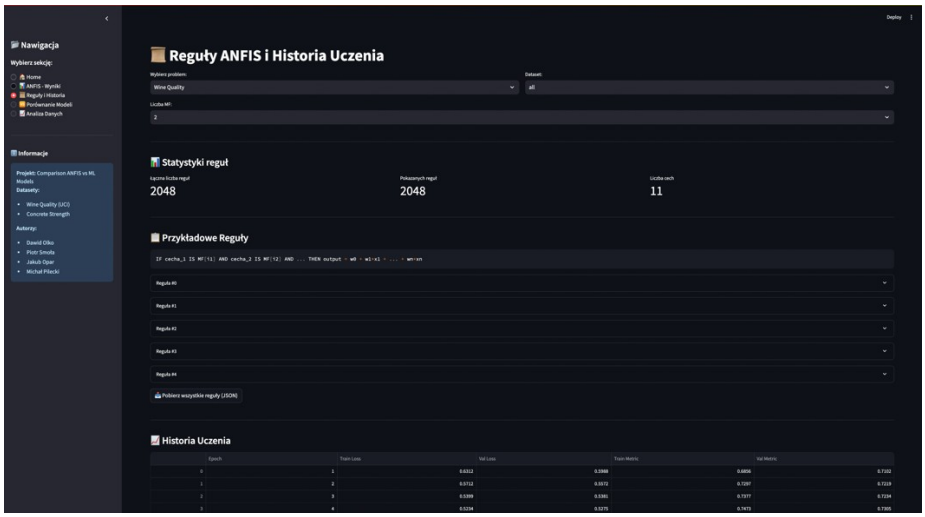


Figure 17: Widok reguł i historii uczenia aplikacji

Cel: Szczegółowa analiza reguł ANFIS oraz tabelaryczna historia treningu.

Zawartość:

1. **Panel kontrolny** (identyczny jak w zakładce ANFIS)
   - Problem / Dataset / n_memb
2. **Sekcja: ANFIS Rules - Top Activated Rules**
   - Lista **top 10 reguł** z największą częstotliwością aktywacji podczas treningu
   - Format czytelny dla człowieka:

     Rule 42: IF x1 IS Low AND x2 IS High AND x3 IS Medium THEN

     y = 0.45 + 0.12*x1 - 0.08*x2 + 0.23*x3 + ...

     Activation frequency: 8.3%

- Pozwala zrozumieć **które reguły dominują** w predykcji

3. **Sekcja: Training History (tabela epoch-by-epoch)**
   - Tabelaryczne przedstawienie metryk dla każdej epoki:
     - **Kolumny:** Epoch | Train Accuracy/MAE | Val Accuracy/MAE | Train Loss | Val Loss

- Umożliwia dokładną analizę dynamiki uczenia
- **Przycisk "Download as CSV"** - eksport do dalszej analizy

4. **Przyciski pobierania:**
   - **Download Full Rules (JSON)** - kompletny zestaw reguł
   - **Download Training History (CSV)** - metryki epoch-by-epoch

6.4. Zakładka Porównanie Modeli

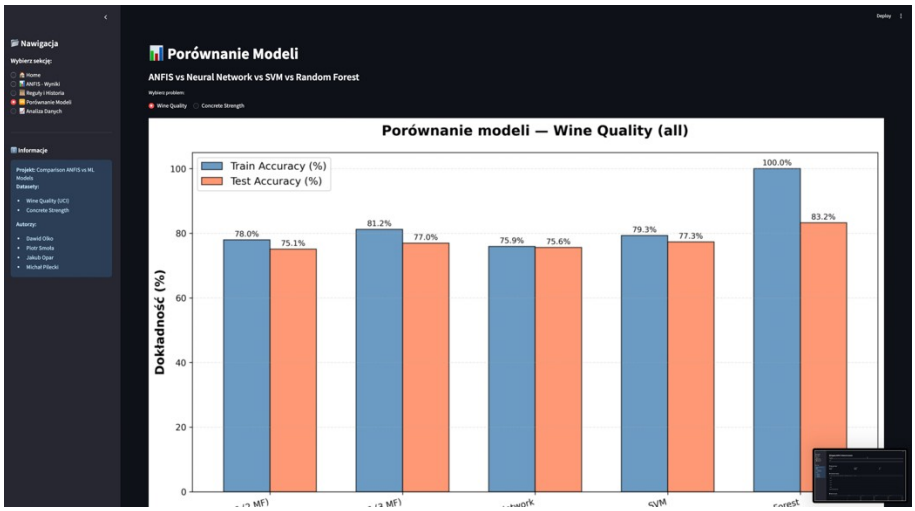


Figure 18: Widok porównania modeli aplikacji

Cel: Zestawienie wydajności ANFIS z trzema klasycznymi algorytmami ML: **Neural Network (NN), Support Vector Machine (SVM), Random Forest (RF)**.

Panel kontrolny:

- Dropdown: Wine Quality / Concrete Strength

Wyświetlane wizualizacje:

1. **Wykres 1: Model Comparison Bar Chart**
   - Plik: model_comparison_bar.png
   - Wykres słupkowy porównujący **Test Metric** dla 4 modeli:
     - **Wine Quality:** Test Accuracy (%) - wyższy = lepszy
     - **Concrete Strength:** Test MAE - niższy = lepszy
   - Kolory: każdy model ma swój kolor dla czytelności

- Pozwala szybko zidentyfikować **najlepszy model**

2. **Wykres 2: Overfitting Analysis (Train vs Test)**

- Plik: overfitting_analysis.png
- **Grouped bar chart** pokazujący:
  - Train Metric (ciemny słupek)
  - Test Metric (jasny słupek)
- Dla każdego z 4 modeli
- **Analiza overfittingu:**
  - Jeśli Train >> Test → model przeuczony
  - Jeśli Train ≈ Test → model dobrze zgeneralizowany

3. **Tabela wyników - szczegółowe porównanie**

- **Kolumny:**
  - Model Name
  - Train Metric (Accuracy/MAE)
  - Test Metric (Accuracy/MAE)
  - Training Time (sekundy)
  - Number of Parameters (dla ANFIS: liczba reguł × parametry/regułę)
- Sortowanie: domyślnie po Test Metric (najlepszy na górze)

Kluczowe wnioski widoczne w tej zakładce:

• Czy ANFIS dorównuje NN/SVM/RF pod względem dokładności?

• Czy ANFIS jest bardziej podatny na overfitting?

• Jak długo trwa trening ANFIS w porównaniu do innych modeli?

6.5. Zakładka Analiza Danych (EDA)

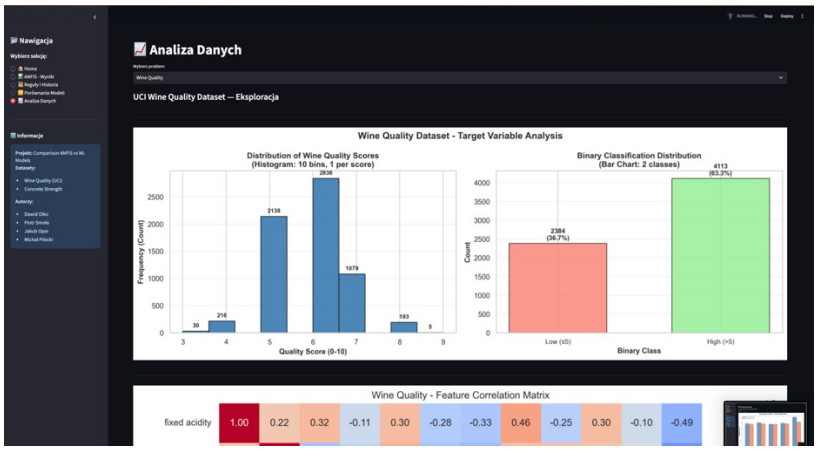


Figure 19: Widok analizy aplikacji

Cel: Eksploracyjna analiza danych (Exploratory Data Analysis) dla obu zbiorów - zrozumienie struktury danych przed modelowaniem.

Panel kontrolny:

- Dropdown: Wine Quality / Concrete Strength

Dla Wine Quality - 4 kluczowe wykresy:

1. **Class Distribution**
   - Plik: wine_class_distribution.png
   - **Subplot 1:** Rozkład oryginalnych ocen jakości (3-8)
   - **Subplot 2:** Rozkład po binaryzacji (klasa 0 vs klasa 1)
   - Pokazuje **problem nierównowagi klas** (37% vs 63%)
2. **Correlation Heatmap**
   - Plik: wine_correlation.png
   - Macierz korelacji Pearsona (11 cech + target)
   - Kolory: czerwony = silna korelacja dodatnia, niebieski = ujemna
   - Identyfikuje cechy najbardziej skorelowane z jakością:
     - **Alcohol** (+0.44) - najsilniejsza korelacja dodatnia
     - **Volatile acidity** (-0.27) - korelacja ujemna

3. **Feature Distributions**
   - Plik: wine_feature_distributions.png
   - Grid 11 histogramów dla każdej cechy
   - Pozwala ocenić:
     - Czy cechy mają rozkład normalny (ważne dla StandardScaler)
     - Czy są wartości odstające (outliers)
     - Skośność rozkładów (skewness)
4. **Pairplot (opcjonalnie)**
   - Plik: wine_pairplot.png
   - Macierz scatter plotów (cecha vs cecha)
   - Kolor punktów: klasa 0 (niebieski) vs klasa 1 (pomarańczowy)
   - Pozwala zobaczyć **separowalność klas** w przestrzeni 2D

Dla Concrete Strength - 4 analogiczne wykresy:

1. **Target Distribution**
   - Plik: concrete_target_distribution.png
   - Histogram rozkładu wytrzymałości betonu (MPa)
   - Analiza:
     - Zakres: 2-82 MPa
     - Kształt rozkładu (prawostronnie skośny)
     - Wykrywanie outlierów
2. **Correlation Heatmap**
   - Plik: concrete_correlation.png
   - Macierz korelacji dla 8 cech + target
   - Najsilniejsze korelacje z wytrzymałością:
     - **Cement** (+0.50)
     - **Superplasticizer** (+0.37)

- **Age** (+0.33)

3. **Feature Distributions**
   - Plik: concrete_feature_distributions.png
   - Grid 8 histogramów
   - Pokazuje różnorodność skal (cement: 100-540, water: 120-250, etc.)
   - Uzasadnia potrzebę **normalizacji StandardScaler**
4. **Pairplot**
   - Plik: concrete_pairplot.png
   - Scatter matrix dla kluczowych cech
   - Kolor = wytrzymałość (gradient continuous colormap)

Opis metadanych datasetu (dla każdego):

- **Źródło:** UCI Machine Learning Repository
- **Liczba próbek:** Wine (6497), Concrete (1030)
- **Liczba cech:** Wine (11), Concrete (8)
- **Typ problemu:** klasyfikacja binarna / regresja
- **Brakujące wartości:** brak (oba zbiory kompletne)

7.INSTRUKCJA URUCHOMIENIA

7.1. Wymagania systemowe

Minimalne wymagania:

- **System operacyjny:** Linux, macOS, Windows 10+
- **Python:** 3.8 - 3.12 **NIE 3.13+** (TensorFlow 2.17 nie wspiera!)
- **RAM:** 4 GB
- **Dysk:** 1 GB wolnego miejsca
- **Internet:** Wymagany do pobrania zależności

Rekomendowane wymagania:

- **System operacyjny:** Ubuntu 22.04 LTS / macOS 13+ / Windows 11
- **Python:** 3.12 (najnowszy wspierany)
- **RAM:** 8 GB lub więcej
- **CPU:** Procesor wielordzeniowy (4+ rdzenie)
- **GPU:** CUDA-compatible GPU (opcjonalnie, przyspiesza trening ~3x)
- **Dysk:** SSD (szybszy odczyt danych)

Software dependencies:

- Python 3.8+ z pip
- Git (do klonowania repozytorium)
- bash (Linux/macOS) lub cmd/PowerShell (Windows)

10.2. Instalacja krok po kroku

METODA 1: Automatyczna instalacja (ZALECANA)

Linux/macOS:

Krok 1: Sklonuj repozytorium

git clone https://github.com/dawidolko/Comparison-ANFIS-Classical-Machine-Learning-Models-Python.git

Krok 2: Wejdź do katalogu

cd Comparison-ANFIS-Classical-Machine-Learning-Models-Python

Krok 3: Nadaj uprawnienia wykonywania

chmod +x setup.sh

Krok 4: Uruchom skrypt setup

./setup.sh

Windows:

REM Krok 1: Sklonuj repozytorium

git clone https://github.com/dawidolko/Comparison-ANFIS-Classical-Machine-Learning-Models-Python.git

REM Krok 2: Wejdź do katalogu

cd Comparison-ANFIS-Classical-Machine-Learning-Models-Python

REM Krok 3: Uruchom skrypt setup

setup.bat

Co robi skrypt automatyczny?

Skrypt `setup.sh` / `setup.bat` wykonuje wszystkie kroki automatycznie:

1. ✓ Tworzy wirtualne środowisko (venv)
2. ✓ Instaluje wszystkie zależności z requirements.txt
3. ✓ Uruchamia data_preprocessing.py
4. ✓ Trenuje ANFIS dla wszystkich datasetów (all, red, white, concrete)

5. ✓ Trenuje modele z 2 i 3 funkcjami przynależności
6. ✓ Wykonuje 5-fold cross-validation
7. ✓ Generuje wizualizacje funkcji przynależności
8. ✓ Wykonuje eksplorację danych (wykresy)
9. ✓ Trenuje modele porównawcze (NN, SVM, RF)
10. ✓ Tworzy wykresy porównawcze
11. ✓ Uruchamia GUI Streamlit na http://localhost:8501

8. PODSUMOWANIE

Projekt miał na celu porównanie systemu ANFIS z klasycznymi algorytmami uczenia maszynowego. Zaimplementowano pięciowarstwową architekturę ANFIS w TensorFlow i przetestowano ją na dwóch problemach: klasyfikacji jakości wina oraz regresji wytrzymałości betonu.

Wyniki pokazują, że ANFIS (3 funkcje przynależności) osiągnął 76.48% accuracy na zbiorze Wine Quality, co stanowi bardzo dobry wynik - tylko o 6.75% gorszy od najlepszego modelu (Random Forest). W zadaniu regresji betonu ANFIS uzyskał najlepszy wynik MAE = 5.18 MPa. Co istotne, ANFIS wykazuje minimalny overfitting (4.59%) w porównaniu do Random Forest (14.46%), co świadczy o jego dobrej generalizacji.

Główną zaletą ANFIS jest interpretowalność - model generuje zrozumiałe reguły rozmyte typu IF-THEN, które można analizować i wyjaśniać, podczas gdy inne modele (NN, SVM, RF) działają jako "czarne skrzynki". ANFIS jest najlepszym wyborem w aplikacjach wymagających wyjaśnialności decyzji, takich jak medycyna czy finanse, gdzie dokładność 76% jest wystarczająca, a zrozumienie procesu decyzyjnego jest kluczowe.

Projekt udowodnił, że ANFIS może skutecznie konkurować z nowoczesnymi algorytmami ML, oferując unikalny kompromis między wydajnością a interpretowalnością modelu.

9. SPIS RYSUNKÓW