Dostrajanie klasyfikatora FUZZY LOGIC z użyciem wybranej procedury optymalizacji inspirowanej naturą

Dokumentacja projektu

Maksym Kazhaiev, Zuzanna Sulima, Viktoriia Kinash

02 lipca 2023

Spis treści:

1. Wstęp

Analiza została dokonana w ramach projektu zaliczeniowego na przedmiot: Metody Inteligencji Obliczeniowej. Projekt ma na celu zaimplementowanie logiki dostrajania klasyfikatora FUZZY LOGIC z użyciem wybranej procedury optymalizacji inspirowanej naturą.

Niektóre wykorzystane pojęcia:

- Fuzzy Logic (logika rozmyta) - metoda analizy, która umożliwia wyrażanie nieprecyzyjnych lub rozmytych danych.
- Algorytm genetyczny - metaheurystyczna metoda optymalizacji, która naśladuje proces ewolucji w naturze. Opiera się na zasadzie selekcji naturalnej, krzyżowania i mutacji genetycznej, aby znaleźć najlepsze rozwiązanie problemu.

2. Implementacja

Jako zbiór danych wykorzystaliśmy dataset umieszczony pod tym linkiem, który dotyczy osiągnięć uczniów w edukacji średniej dwóch portugalskich szkół. Dane zostały zebrane za pomocą raportów szkolnych i kwestionariuszy. Atrybuty obejmują oceny uczniów, dane demograficzne, społeczne i związane ze szkołą, a właściwie:

- school - szkoła ucznia
- sex- płeć ucznia
- age - wiek ucznia
- adres - rodzaj adresu domowego ucznia

- famsize - wielkość rodziny
- Pstatus - status wspólnego zamieszkiwania rodziców
- Medu - wykształcenie matki
- Fedu - wykształcenie ojca
- Mjob - zawód matki
- Fjob - zawód ojca
- powód - powód wyboru tej szkoły
- opiekun - opiekun ucznia
- traveltime - czas podróży z domu do szkoły
- studytime - tygodniowy czas nauki
- failures - liczba niepowodzeń w przeszłości
- schoolsup - dodatkowe wsparcie edukacyjne
- famsup - rodzinne wsparcie edukacyjne
- paid - dodatkowe płatne zajęcia z przedmiotu kursu
- activities - zajęcia pozalekcyjne
- nursery - uczęszczał do przedszkola
- higher - chce kontynuować wyższe wykształcenie
- internet - dostęp do Internetu w domu
- romantyczny - w związku romantycznym
- famrel - jakość relacji rodzinnych
- freetime - wolny czas po szkole
- wychodzić - wychodzenie z przyjaciółmi
- Dalc - spożycie alkoholu w dni powszednie
- Walc - spożycie alkoholu w weekend
- health - obecny stan zdrowia
- absences - liczba nieobecności w szkole

Do analizy zostały wykorzystane atrybuty zaznaczone na różowo.

W celu zoptymalizowania parametrów utworzonego układu Fuzzy Logic, wykorzystano algorytm genetyczny. Jest to algorytm inspirowany procesem ewolucji w przyrodzie, który polega na tym, że najlepiej przystosowane organizmy mają większe szanse na przetrwanie i reprodukcję. Podobnie, w przypadku algorytmu genetycznego utworzona jest populacja rozwiązań i wykorzystane są operatory genetyczne, takie jak selekcja, krzyżowanie i mutacja, aby rozwiązania ewoluowały w stronę optymalności.

W tym przypadku, populacją były przedziały zmiennych, które są używane do konfiguracji kontrolera rozmytego. Algorytm ewoluował poprzez generowanie nowych pokoleń populacji i zastosowanie operatorów genetycznych:

Selekcja - w każdej iteracji algorytmu, populacja była poddawana procesowi selekcji, gdzie najlepsze osobniki (rozwiązania) miały większe szanse na zostanie wybranymi jako rodzice kolejnej generacji

Krzyżowanie - polegało na łączeniu cech rodziców w celu stworzenia nowych osobników

Mutacja - polegała na wprowadzeniu losowych zmian w genotypach osobników

Klasa '**Solution**' reprezentuje pojedyncze rozwiązanie w algorytmie genetycznym. Opis poszczególnych zmiennych i metod:

- **antecedent_universes** - zmienna określająca zbiory rozmyte dla zmiennych wejściowych
- **consequent_universe** - zmienna określająca zbiór rozmyty dla zmiennej wyjściowej
- **antecedent_ranges** - zmienna określająca przedziały zmiennych wejściowych
- **consequent_range** - zmienna określająca przedział zmiennej wyjściowej
- **fitness** - zmienna określająca wartość dopasowania danego rozwiązania
- **fzCtrl** - zmienna określająca układ sterowania rozmytego
- **__init__(self, fzCtrl)** - inicjalizuje nowy obiekt klasy i jego zmienne. Przyjmuje jako argument obiekt kontrolera rozmytego.
- **initialize_ranges(self)** - inicjalizuje przedziały dla zmiennych wejściowych i wyjściowej. Losowo generuje trzy wartości dla każdego z przedziałów, a następnie sortuje je w kolejności rosnącej.
- **update_fuzzy_control_system(self)** - aktualizuje system kontrolny rozmyty oraz ustawia funkcje przynależności dla zmiennych wejściowych i wyjściowej na podstawie bieżących przedziałów.
- **evaluate_fitness(self, attributes_train, grades_train)** - oblicza dopasowanie dla danego rozwiązania, aktualizuje system kontrolny rozmyty oraz przewiduje oceny na podstawie atrybutów i porównuje je z rzeczywistymi ocenami. Dopasowanie jest obliczane jako średnia kwadratowa różnicy między przewidywanymi ocenami a rzeczywistymi ocenami.
- **predict_grade(self, attributes)** - przewiduje ocenę na podstawie atrybutów, aktualizuje system kontrolny rozmyty, ustawia wartości atrybutów i oblicza wynik na podstawie reguł logiki rozmytej. Zwraca przewidywaną ocenę.
- **mutate(self, mutation_rate)** - metoda przeprowadzająca mutowanie rozwiązania. Dla każdej zmiennej wejściowej i wyjściowej, z określonym prawdopodobieństwem mutacji, losuje nowe wartości przedziałów i aktualizuje przedziały zmiennej wejściowej.
- **crossover(self, other)** - metoda przeprowadzająca krzyżowanie rozwiązań. Tworzy dwa nowe potomne rozwiązania, gdzie dla każdej zmiennej wejściowej i wyjściowej losuje punkt krzyżowania oraz wymienia przedziały między rodzicami, a na końcu zwraca te dwa potomne rozwiązania.

Do procesu selekcji (funkcja **perform_selection**) wykorzystano **metodę turniejową**. Polega ona na tym, że losowo wybierana jest określona liczba osobników z populacji, a następnie spośród nich wybierany jest zwycięzca na podstawie ich wartości funkcji przystosowania (fitness). W tym przypadku, liczba kandydatów w turnieju jest określona przez zmienną **tournament_size**, a zwycięzcą jest osobnik o najwyższej wartości funkcji przystosowania. Proces ten powtarzany był przez ___(ilość)___ generacji, aż zostały osiągnięte satysfakcjonujące wyniki.

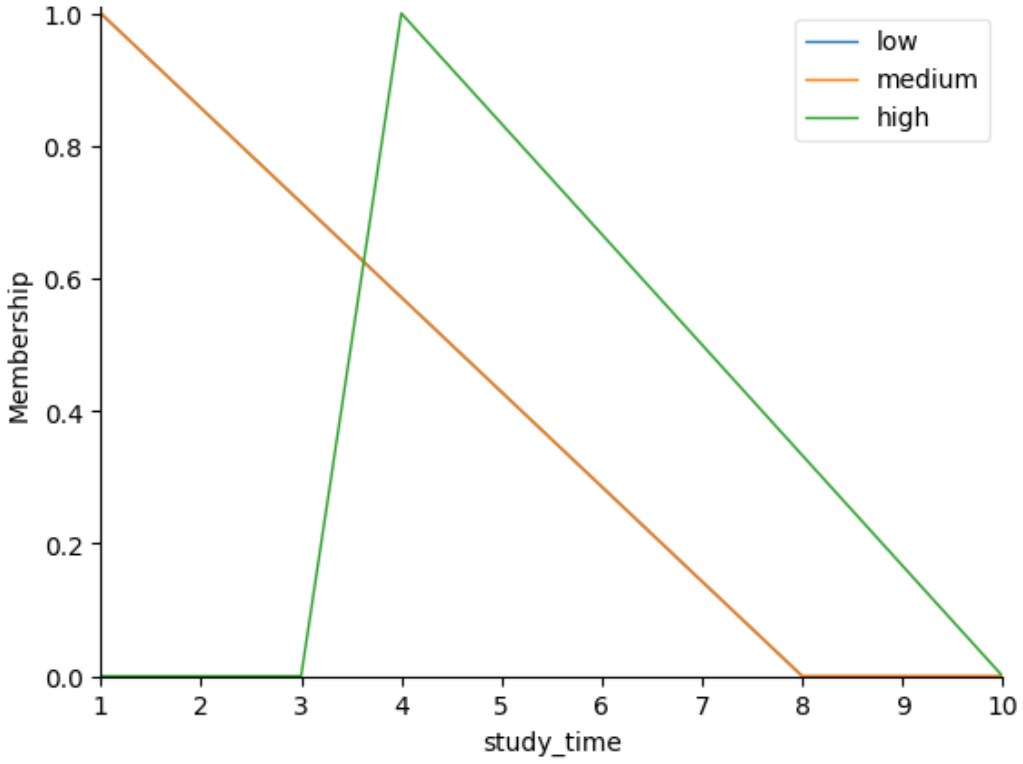
3. Przeprowadzenie trenowania

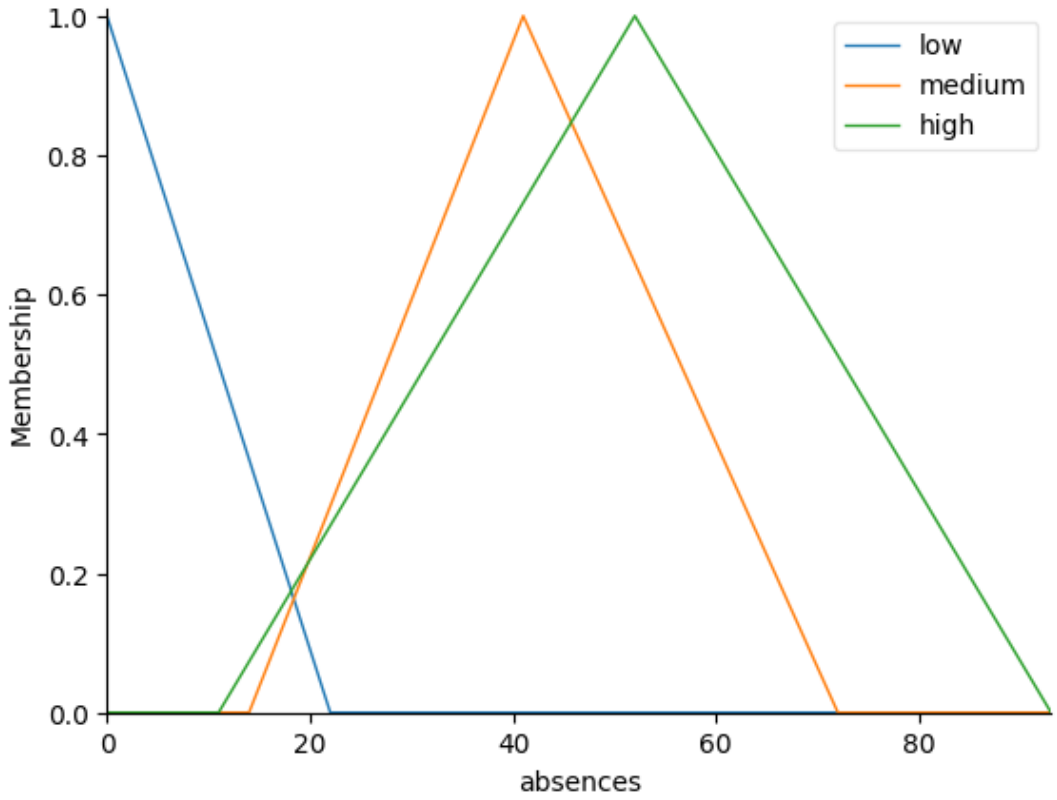
Na początku stworzyliśmy Fuzzy Logic dla wybranych atrybutów: studytime w zakresie od 0 do 10, health w zakresie od 0 do 93, absences w zakresie od 0 do 5. Później wczytano dane dotyczące studentów oraz ich ocen z pliku .csv oraz podzielono dane na uczące i testujące. Następnie przeprowadzono inicjalizację populacji przy użyciu klasy Solution, która reprezentuje osobniki w algorytmie genetycznym. Utworzono 50 osobników, z których każdy reprezentuje rozwiązanie. Algorytm genetyczny został uruchomiony przez ___(ilość)____ pokoleń. W każdej iteracji, dla każdego osobnika w populacji, obliczone zostało dopasowanie przy użyciu funkcji **evaluate_fitness,** która wykorzystuje dane treningowe. W kolejnym kroku, wybierany jest najlepszy osobnik z populacji, populacja jest poddawana selekcji turniejowej, a następnie tworzona jest nowa populacja potomków przez krzyżowanie i mutację.

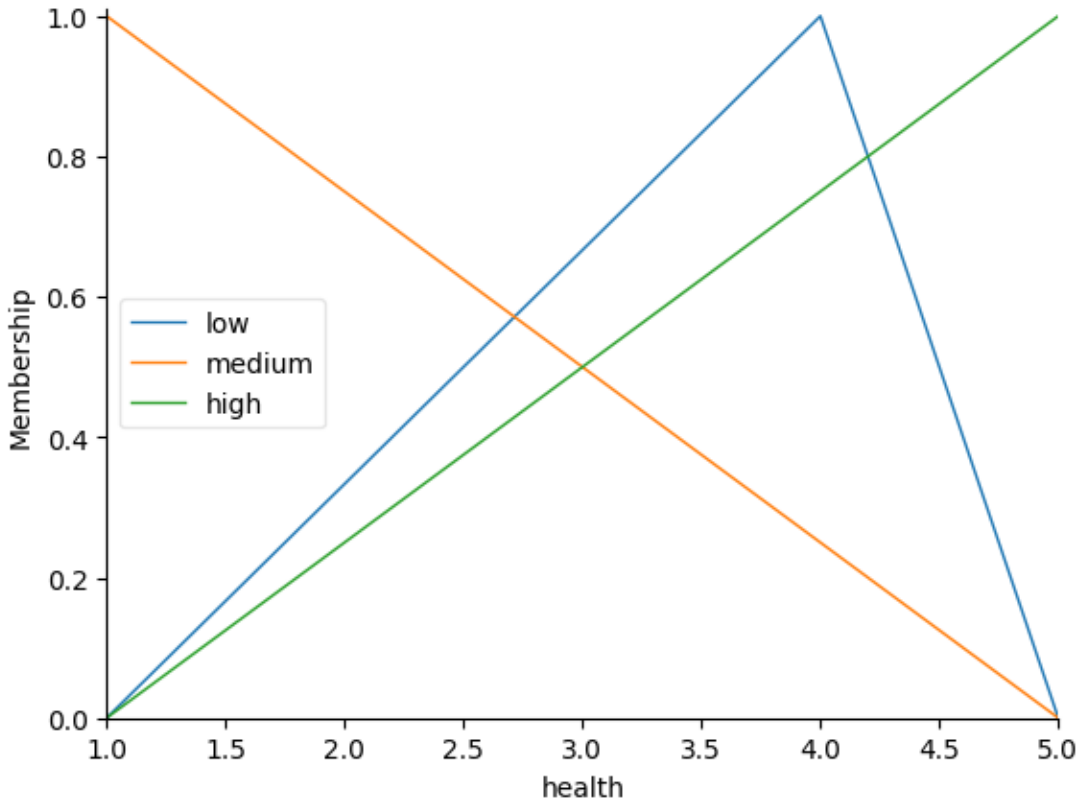
Po przejściu przez wszystkie pokolenia, najlepszy osobnik jest stosowany do zbioru testowego. Dla każdego zestawu cech w zbiorze testowym, przewidywana ocena jest obliczana przy użyciu logiki rozmytej.

4. Wyniki

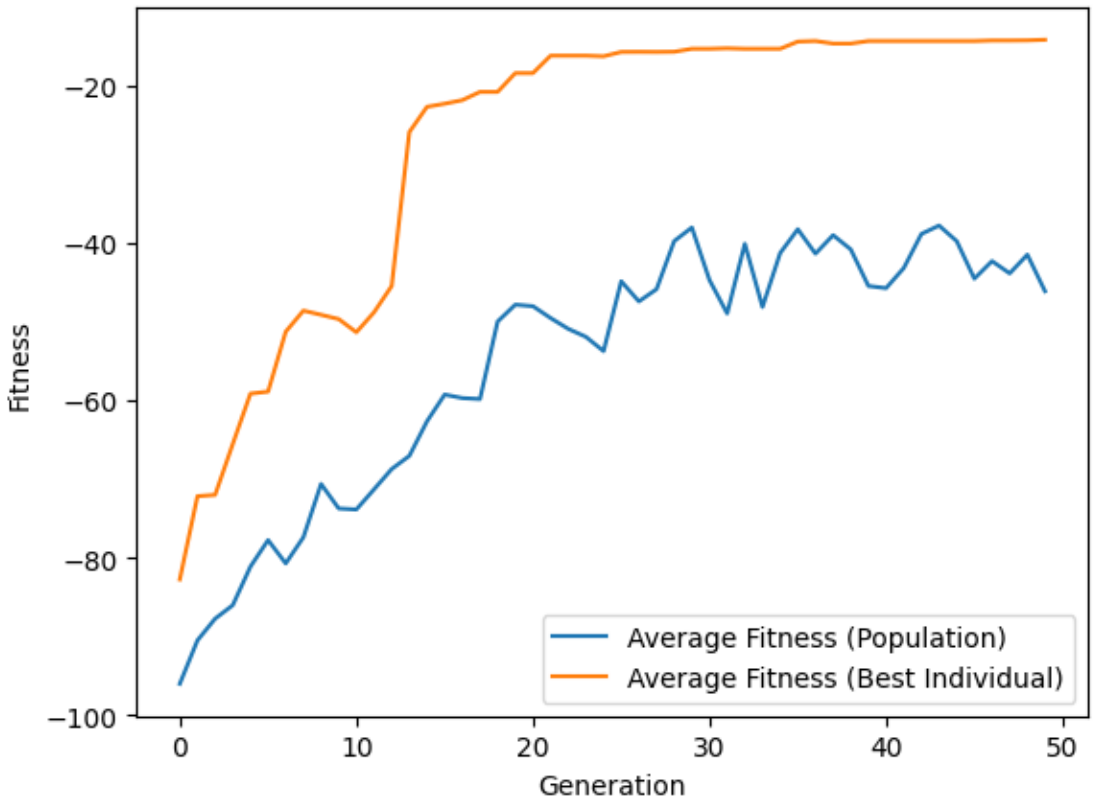
Funkcje przynależności dla każdego atrybutu:




low
medium
high
Membership
absences
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80


low
medium
high
Membership
health
1.0
0.8
0.6
0.4
0.2
0.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

Wykres wydajności algorytmu genetycznego w kolejnych generacjach. Konkretnie, wykres przedstawia średnią wartość funkcji dopasowania dla całej populacji oraz średnią wartość funkcji dopasowania dla najlepszego osobnika w każdej generacji.



Najlepsza wartość funkcji dopasowania (fitness) najlepszego osobnika znalezionego podczas procesu ewolucji algorytmu genetycznego to: -45.320445657547346, przy tym średnia kwadratowa różnica między przewidywanymi wartościami a rzeczywistymi wartościami w zbiorze testowym wynosiła: 5.9558542934461345.

5. Wnioski

Przygotowany model określa oceny studenta z pewnym odchyleniem, ale ogólnie predykcje są dość bliskie rzeczywistym wartościom. Istnieje kilka powodów tego odchylenia: po pierwsze, dany zbiór jest szacowany bardzo dużą ilością parametrów i jest ciężko określić, które mają największy wpływ. Możliwe, że uwzględniamy zbyt mało parametrów żeby system mógł działać tak, jak w przykładowym zbiorze. Zdefiniowane reguły też mają bardzo duży wpływ na poprawność algorytmu, a jesteśmy w stanie je dobrać jedynie w oparciu o swoje doświadczenie i modyfikować metodą prób i błędów przez co nie da się idealnie określić te reguły.