Metody aproksymacji rozmytej w procesach decyzyjnych

Metody Sztucznej Inteligencji

Damian Opoka

Piotr Kryczka

Jakub Michalak

Wojciech Podmokły

Adam Ryl

Abstrakt

W poniższym raporcie opisano wykorzystanie aproksymacji rozmytej w procesie decyzyjnym, w którym dostarczone informacje są niepełne, niejednoznaczne i wymagają dokładniejszej analizy.

Rozwiązanie przedstawionego poniżej problemu wymaga matematycznego przedstawienia danych wejściowych w postaci zbiorów rozmytych. Przeprowadzane obliczenia wymagają zastosowania relacji rozmytych, różnych rodzajów norm trójkątnych i implikacji rozmytych, aproksymacji zbiorów rozmytych w oparciu o relacje oraz odległości między zbiorami.

Logika rozmyta i jej zastosowanie

Teoria logiki rozmytej opiera się na obserwacji względnej częściowej przynależności. Idea ta jest wynikiem obserwacji procesu ludzkiej percepcji i poznania. Lotfi A. Zahed opublikował swoją pierwszą słynną pracę naukową o zbiorach rozmytych w 1965 roku.

Logika rozmyta radzi sobie z poznaniem i doświadczeniem, które jest niepewne, nieprecyzyjne, częściowo prawdziwe lub nie posiada wyraźnych granic. Logika rozmyta pozwala na użycie niejasnych ocen dokonywanych przez człowieka w komputerowych obliczeniach. Zapewnia również efektywne sposoby rozwiązywania problemów o wielu kryteriach i lepszą ocenę dostępnych opcji rozwiązania.

Nowe metody obliczeniowe oparte na logice rozmytej mogą być używane w rozwoju inteligentnych systemów podejmowania decyzji, identyfikacji, rozpoznawania wzorców, optymalizacji i kontroli. Logika rozmyta jest niezwykle przydatna dla wielu osób zaangażowanych w badania i rozwój, w tym inżynierów, matematyków, twórców oprogramowania i badaczy, naukowców specjalizujących się w naukach przyrodniczych i medycznych, socjologów, ekonomistów, analityków biznesowych i prawników.

Rzeczywiście, wdrożenie logiki rozmytej, która kiedyś była jedynie mało znaną ciekawostką matematyczną, jest widoczne szczególnie w inżynierii i pracach naukowo-badawczych. Logika rozmyta ma liczne zastosowania w rozpoznawaniu rysów twarzy, urządzeniach takich jak klimatyzatory, pralki, odkurzacze, przeciwpoślizgowe układy hamulcowe, układy transmisyjne, helikoptery bezzałogowe, w systemach do wielokryterialnej optymalizacji systemów elektroenergetycznych, prognozowania pogody, modelowania zmiany cen nowych produktów, oceny ryzyka projektów, diagnostyki medycznej i planów leczenia. Sprawdziła się w wielu dziedzinach, takich jak inżynieria systemów sterowania, przetwarzanie obrazów, energetyka, automatyka i robotyka, elektronika użytkowa i optymalizacja.

Logika rozmyta jest gałęzią matematyki, która tchnęła nowe życie w naukę i rozwój cywilizacji. Setki badaczy pracując z całym matematycznym aparatem logiki rozmytej tworzy patenty i przełomowe artykuły naukowe. Więcej informacji, danych statystycznych oraz źródeł wiedzy na ten temat można odnaleźć w pracy [4].

Problem

Celem projektu jest rozwiązanie poniżej opisanego problemu w oparciu o matematyczny aparat zagadnienia zbiorów rozmytych.

Pewna firma prowadzi rekrutację na daną liczbę stanowisk $P = \{p_1, ..., p_n\}$, gdzie $n \geq 20$. Każde spośród n stanowisk ma przypisany znany zbiór stopni umiejętności $S = \{s_1, ..., s_m\}$, gdzie $m \geq 10$, wymaganych w różnym stopniu na to stanowisko.

Do pracy w firmie aplikuje k kandydatów $C = \{c_1, ..., c_k\}$, z których każdy wykazuje w różnym stopniu umiejętności ze zbioru S.

Dla każdego $p \in P$, $s \in S$ oraz $c \in C$ znane są wartości:

- $R(p, s) \in [0, 1]$ - stopień, w jakim na stanowisko p wymagana jest umiejętność s, oszacowany np. przez *hiring managera* lub właściciela firmy,
- $Q(c, s) \in [0, 1]$ - stopień, w jakim kandydat c wykazuje umiejętność s, oszacowany np. przez rekrutera lub specjalistę od HR.

Aby problem decyzyjny został rozwiązany, należy wyznaczyć najlepszego kandydata na każde ze stanowisk. Może się zdarzyć także sytuacja, że dany kandydat okaże się najlepszym kandydatem na więcej niż jedno stanowisko.

Powyższy problem zostanie rozwiązany z wykorzystaniem zbiorów rozmytych. Rozwiązanie jest oparte na pracy [1], w której został przedstawiony analogiczny problem z dziedziny medycyny i etapy jego rozwiązania. W ramach realizacji projektu został wykonany opis teoretyczny przedstawiający matematyczne zagadnienia związane z aproksymacją rozmytą. Została także stworzona aplikacja uruchamiana w systemie Windows, która pozwoli na łatwe wprowadzanie i modyfikacje danych wejściowych oraz rozwiązanie problemu w oparciu o wybraną normę trójkątną oraz odpowiadającą jej implikację rozmytą.

Wprowadzenie teoretyczne

Przed rozwiązaniem problemu niezbędne będzie zaznajomienie się z kilkoma wspomnianymi już zagadnieniami teoretycznymi i definicjami dotyczącymi aproksymacji rozmytej.

Zbiór rozmyty

Zbiorem rozmytym A w pewnym zbiorze rozważań X, co zapisujemy jako $A \subseteq X$, nazywamy zbiór par:

$$A = (\mu_A(x), x),$$

dla każdego $x \in X$. Przekształcenie $\mu_A : X \to [0, 1]$ nazywamy funkcją przynależności rozmytego zbioru A, która każdemu elementowi $x \in X$ przypisuje jego stopień przynależności do A, $\mu_A(x) \in [0, 1]$.
Rodzinę wszystkich zbiorów rozmytych na X oznaczamy jako $\mathcal{F}(X)$.

Relacja rozmyta

Relacja rozmyta R na zbiorach rozmytych X i Y, to zbiór rozmyty z $X \times Y$. Dla $x \in X$ i dla $y \in Y$, R(x,y) to stopień w jakim x jest związana z y. Niech $\mathcal{R}(X, Y)$ oznacza rodzinę wszystkich relacji rozmytych na X i Y. Dla $R \in \mathcal{R}(X, Y)$ relacja odwrotna, to $R^{-1}(y, x) = R(x, y)$. Dla każdego $R \in \mathcal{R}(X, Y)$ i dla każdego $x \in X$, xR oznacza zbiór rozmyty Y definiowany jako $(xR)(y) = R(x, y)$. Oznaczenie Ry jest definiowane analogicznie.

Przykłady relacji, które zostaną użyte później:

- relacja związania:
  - dla obiektów: x_1 jest związany z x_2, jeśli wszystkie właściwości x_1 są także właściwościami x_2.
  - dla właściwości: y_1 jest związana z y_2, jeśli wszystkie obiekty mające właściwość y_1 mają także właściwość y_2.
- relacja kompatybilności:
  - dla obiektów: x_1 i x_2 są kompatybilne, jeśli oba mają pewną właściwość y.
  - dla właściwości: y_1 i y_2 są kompatybilne, jeśli pewien obiekt x ma obie te właściwości.

Normy trójkątne

Normy trójkątne (t-normy) to przekształcenie $\otimes : [0, 1]^2 \to [0, 1]$, które spełnia następujące warunki:

- przemienność ($x \otimes y = y \otimes x$),
- łączność ($x \otimes (y \otimes z) = (x \otimes y) \otimes z$),
- monotoniczność (dla każdego $x \leq z$ spełnione jest $x \otimes y \leq z \otimes y$ oraz $y \otimes x \leq y \otimes z$),
- element neutralny 1 (dla każdego $x \in [0, 1]$ $x \otimes 1 = x$).

Przy rozwiązaniu uwzględnione zostaną trzy t-normy:

- standardowa t-norma: $x \otimes_Z y = min(x, y)$,
- produktowa t-norma: $x \otimes_P y = xy$,
- t-norma Łukasiewicza: $x \otimes_L y = max(0, x + y - 1)$.

Implikacja rozmyta

Implikacja rozmyta to przekształcenie $\rightarrow$: $[0,1]^2 \rightarrow [0,1]$, które spełnia warunki:

- dla każdego $x, y, z \in [0,1]$, jeśli $x \leq z$ to $z \rightarrow y \leq x \rightarrow y$ i $y \rightarrow x \leq y \rightarrow z$,
- $1 \rightarrow 1 = 0 \rightarrow 0 = 0 \rightarrow 1 = 1$ i $1 \rightarrow 0 = 0$.

W rozwiązaniu wykorzystane zostaną implikacje rezydualne, które definiowane są w sposób następujący:

$$x \rightarrow y = sup\{z \in [0,1] : x \otimes z \leq y\}.$$

T-normom odpowiadają kolejno następujące implikacje rezydualne:

- implikacja Gödela: $x \rightarrow_{GD} y = \begin{cases} 1 & \text{gdy } x \leq y \\ y & \text{gdy } x > y \end{cases}$,
- implikacja Goguen-Gainesa: $x \rightarrow_{GG} y = \begin{cases} 1 & \text{gdy } x \leq y \\ \frac{y}{x} & \text{gdy } x > y \end{cases}$,
- implikacja Łukasiewicza: $x \rightarrow_{L} y = min(1, 1 - x + y)$.

Aproksymacje zbiorów rozmytych oparte na relacjach

Niech X, Y będą niepustymi zbiorami rozmytymi oraz niech $R \subseteq X \times Y$ będzie relacją rozmytą. Niech $A \in \mathcal{F}(Y)$. Zdefiniujmy następujące operatory dla każdego $x \in X$:

$$([R]_{\otimes}A)(x) = \inf_{y \in Y}(R(x,y) \rightarrow A(y))$$
$$(\langle R \rangle_{\otimes}A)(x) = \sup_{y \in Y}(R(x,y) \otimes A(y))$$

Powyższe operatory to odpowiednio rozmyta konieczność oraz rozmyta możliwość. Relację R można sobie wyobrazić jako podobieństwa obiektów na podstawie ich właściwości. Wtedy te operatory są odpowiednio operatorami rozmytych dolnych i górnych aproksymacji. Obrazowo, mając $A \in \mathcal{F}(Y)$ oraz $x \in X$:

- $([R]_{\otimes}A)(x)$ jest stopniem w jakim obiekt x jest związany z A,
- $(\langle R \rangle_{\otimes}A)(x)$ jest stopniem w jakim obiekt x jest kompatybilny z A.

Analogicznie, rozważając $B \in \mathcal{F}(X)$ reprezentujące decyzję eksperta, dla dowolnego $y \in Y$, $([R^{-1}]_{\otimes}B)(y)$ jest stopniem, w jakim właściwość y jest związana z decyzją B i $(\langle R^{-1} \rangle_{\otimes}B)(y)$ jest stopniem, w jakim y jest kompatybilny z B.
Teraz, na podstawie powyższych operatorów możemy zdefiniować dwa przekształcenia $\blacktriangle^R_{\otimes}$, $\blacktriangledown^R_{\otimes}$: $\mathcal{F}(Y) \rightarrow \mathcal{F}(Y)$, dla każdego $A \in \mathcal{F}(Y)$:

- $\blacktriangle^R_{\otimes}A = \langle R^{-1} \rangle_{\otimes}[R]_{\otimes}A$,
- $\blacktriangledown^R_{\otimes}A = [R^{-1}]_{\otimes}\langle R \rangle_{\otimes}A$.

Obrazowo, mając $A \in \mathcal{F}(Y)$ oraz $y \in Y$:

- $(\blacktriangle^R_{\otimes}A)(y)$ - stopień, w jakim jakiś obiekt charakteryzowany przez właściwość y jest związany z obiektem A,
- $(\blacktriangledown^R_{\otimes}A)(y)$ - stopień, w jakim wszystkie obiekty charakteryzowane przez właściwość y są kompatybilne z obiektem A.

Wiemy, że te operatory to odpowiednio dolne oraz górne ograniczenie A. Zatem dla każdego $y \in Y$, $(\blacktriangle^R_{\otimes}A)(y)$ może być opisane jako stopień, w którym y co najmniej (z pewnością) należy do A, i $(\blacktriangle^R_{\otimes}A)(y)$ może być interpretowane jako stopień, w którym y co najwyżej (prawdopodobnie) należy do A.

Odległość między zbiorami rozmytymi

Niech $X = \{x_1, \ldots, x_n\}$ oraz niech $A = \{(\mu_A(x), x) : x \in X\}$, $B = \{(\mu_B(x), x) : x \in X\}$ oznaczają dwa zbiory rozmyte na X. Dana jest również relacja relacje $R(a, b) \in \mathcal{R}(A, B)$.

Znormalizowaną odległością Hamminga pomiędzy zbiorem A i B nazywamy:

$$\delta_H(A, B) = \frac{1}{2n} \sum_{i=1}^{n} d_i,$$

gdzie

$$d_i = |(\blacktriangle_{\otimes}^{R} A)(x_i) - (\blacktriangle_{\otimes}^{R} B)(x_i)| + |(\blacktriangledown_{\otimes}^{R} A)(x_i) - (\blacktriangledown_{\otimes}^{R} B)(x_i)|.$$

Znormalizowaną odległość euklidesową pomiędzy zbiorem A i B nazywamy:

$$\delta_H(A, B) = \sqrt{\frac{1}{2n} \sum_{i=1}^{n} e_i},$$

gdzie

$$e_i = [(\blacktriangle_{\otimes}^{R} A)(x_i) - (\blacktriangle_{\otimes}^{R} B)(x_i)]^2 + [(\blacktriangledown_{\otimes}^{R} A)(x_i) - (\blacktriangledown_{\otimes}^{R} B)(x_i)]^2.$$

Rozwiązanie problemu

Algorytm składa się z trzech kroków.

Krok pierwszy

Dla każdego stanowiska p z n stanowisk $P = \{p_1, ..., p_n\}$, $n \geq 20$ aproksymujemy umiejętności $S = \{s_1, ..., s_m\}$, $m \geq 10$ potrzebne na dane stanowisko ze względu na relację $Q(c, s) \in [0, 1]$, oznaczającą stopień, w jakim kandydat c ze zbioru $C = \{c_1, ..., c_k\}$ wykazuje się umiejętnością s ze zbioru S. W tym celu wyznaczane są ograniczenia $(\blacktriangle_L^Q, \blacktriangledown_L^Q)$ zbioru pR. Relacja $R(p, s) \in [0, 1]$ oznacza stopień, w jakim na stanowisko $p \in P$ wymagana jest umiejętność $s \in S$.

Krok drugi

Dla każdego kandydata $c \in C$, aproksymujemy jego umiejętności ze względu na relację Q. Wyznaczamy zatem $(\blacktriangle_L^Q, \blacktriangledown_L^Q)$ zbioru cQ.

Krok trzeci

Dla każdego stanowiska p i dla każdego kandydata c obliczamy odległość pomiędzy pR, a cQ. Najmniejsza wartość odległości wskazuje najlepszego kandydata.

Przykład z zastosowaniem różnych norm

Przedstawienie problemu

Pewna firma informatyczna ogłosiła rekrutację na stanowiska: Web Developer, Game Developer, DevOps, .NET Developer oraz Big Data. Umiejętności mające wpływ na ocenę kandydata to: C++, C#, Java, Python, Linux, Javascript i PHP. Do firmy zgłosiło się sześć osób: Jan, Kacper, Krzysztof, Maciej, Aleksander oraz Mateusz.
Policzymy najlepsze dopasowanie kandydatów do stanowisk wcześniej opisanym algorytmem, z wykorzystaniem wszystkich wymienionych wcześniej norm wraz z odpowiadającymi im implikacjami, korzystając z odległości Hamminga oraz Euklidesowej.

Dane wejściowe

Na wejściu mamy następujące dane:

- tabelę relacji R, czyli stopień, w jakim dane umiejętności są pożądane na dane stanowisko,

R	C++	C#	Java	Python	Linux	Javascript	PHP
Web Developer	0	0,4	0,8	0,7	0,4	1	0,9
Game Developer	0,9	0,7	0,6	0,3	0,2	0,1	0
DevOps	0,8	0,3	0,5	0,6	1	0,3	0,1
.NET Developer	0,8	1	0,7	0,5	0,2	0,1	0
Big Data	0,4	0,5	0,6	1	0,6	0,7	0,2

- tabelę relacji Q, czyli umiejętności kandydatów.

Q	C++	C#	Java	Python	Linux	Javascript	PHP
Jan	0,6	0,7	0	0,1	1	0,7	0,4
Kacper	0,3	0,8	0,2	0,4	0,7	0,8	0,9
Krzysztof	0,5	0,1	0,4	0,9	0,5	0,3	0,1
Maciej	0,3	0,2	0,6	0,7	0,9	1	0,5
Aleksander	0,6	0	0,1	0,2	0,8	0,9	0,6
Mateusz	1	0,4	0,8	1	0,5	0,3	0,2

Obliczenia

Norma standardowa i odpowiadająca jej implikacja Gödela

Po kroku 1. dostajemy:

$(\blacktriangle_L^Q, \blacktriangledown_L^Q)$ pR	C++	C#	Java	Python	Linux	Javascript	PHP
Web Developer	(0,8; 0)	(1; 0)	(1; 0)	(0,7; 0)	(0,7; 0)	(1; 0)	(1; 0)
Game Developer	(0,9; 0)	(0,7; 0)	(1; 0)	(0,5; 0)	(0,6; 0)	(0,6; 0)	(0,7; 0)
DevOps	(0,8; 0,5)	(0,7; 0,1)	(1; 0,4)	(0,6; 0,6)	(1; 0,5)	(0,7; 0,3)	(0,7; 0,1)
.NET Developer	(0,8; 0)	(1; 0)	(1; 0)	(0,5; 0)	(0,6; 0)	(0,6; 0)	(0,8; 0)
Big Data	(1; 0,4)	(0,7; 0,4)	(1; 0,4)	(1; 0,4)	(0,7; 0,4)	(0,7; 0,3)	(0,7; 0,2)

Tabela 1: Ograniczenia zbioru pR.

Po kroku 2. dostajemy:

$(\blacktriangle_L^Q, \blacktriangledown_L^Q)$ cQ	C++	C#	Java	Python	Linux	Javascript	PHP
Jan	(0,6; 0,6)	(0,7; 0,7)	(0,6; 0)	(0,5; 0,1)	(1; 1)	(0,7; 0,7)	(0,7; 0,4)
Kacper	(0,5; 0,3)	(1; 0,8)	(0,5; 0,2)	(0,5; 0,4)	(0,7; 0,7)	(0,8; 0,8)	(1; 0,9)
Krzysztof	(0,5; 0,5)	(0,5; 0,1)	(1; 0,4)	(0,9; 0,9)	(0,5; 0,5)	(0,5; 0,3)	(0,5; 0,1)
Maciej	(0,7; 0,3)	(1; 0,2)	(0,7; 0,6)	(0,7; 0,7)	(0,9; 0,9)	(1; 1)	(0,8; 0,5)
Aleksander	(0,6; 0,6)	(1; 0)	(0,6; 0,1)	(0,5; 0,2)	(0,8; 0,8)	(0,9; 0,9)	(0,8; 0,6)
Mateusz	(1; 1)	(0,5; 0,4)	(1; 0,8)	(1; 1)	(0,5; 0,5)	(0,5; 0,3)	(0,5; 0,2)

Tabela 2: Ograniczenia zbioru cQ.

Po kroku 3. Ostatecznie dostajemy:

	Jan	Kacper	Krzysztof	Maciej	Aleksander	Mateusz
Web Developer	0,393	0,379	0,357	0,357	**0,314**	0,457
Game Developer	0,336	0,421	**0,3**	0,429	0,343	0,386
DevOps	0,25	0,336	**0,143**	0,243	0,271	0,243
.NET Developer	0,357	0,386	0,321	0,393	**0,307**	0,421
Big Data	0,286	0,314	**0,179**	0,293	0,321	**0,179**

Tabela 3: Odległości między zbiorami rozmytymi (odległość Hamminga).

	Jan	Kacper	Krzysztof	Maciej	Aleksander	Mateusz
Web Developer	0,471	0,485	**0,423**	0,48	0,424	0,541
Game Developer	0,457	0,501	**0,378**	0,504	0,439	0,503
DevOps	0,326	0,394	**0,214**	0,295	0,312	0,298
.NET Developer	0,461	0,486	**0,396**	0,495	0,427	0,523
Big Data	0,338	0,376	**0,238**	0,328	0,352	0,274

Tabela 4: Odległości między zbiorami rozmytymi (odległość Euklidesowa).

Norma produktowa i odpowiadająca jej implikacja Goguen-Gainesa

Po kroku 1. dostajemy:

$(\blacktriangle_L^Q, \blacktriangledown_L^Q)$ pR	C++	C#	Java	Python	Linux	Javascript	PHP
Web Developer	(0,7; 0)	(1; 0)	(0,875; 0)	(0,7; 0)	(0,7; 0)	(1; 0)	(0,9; 0)
Game Developer	(0,9; 0)	(0,7; 0)	(0,6; 0)	(0,5; 0)	(0,4; 0)	(0,36; 0)	(0,622; 0)
DevOps	(0,8; 0,5)	(0,875; 0,2)	(1; 0,4)	(0,6; 0,6)	(1; 0,333)	(0,875; 0,2)	(0,778; 0,1)
.NET Developer	(0,8; 0)	(1; 0)	(0,7; 0)	(0,5; 0)	(0,467; 0)	(0,42; 0)	(0,8; 0)
Big Data	(1; 0,4)	(0,7; 0,35)	(1; 0,32)	(1; 0,72)	(0,6; 0,5)	(0,7; 0,4)	(0,622; 0,2)

Tabela 5: Ograniczenia zbioru pR.

Po kroku 2. dostajemy:

$(\blacktriangle_L^Q, \blacktriangledown_L^Q)$ cQ	C++	C#	Java	Python	Linux	Javascript	PHP
Jan	(0,6; 0,6)	(0,875; 0,7)	(0,75; 0)	(0,556; 0,1)	(1; 1)	(0,875; 0,7)	(0,778; 0,4)
Kacper	(0,4; 0,3)	(1; 0,8)	(0,5; 0,2)	(0,4; 0,4)	(0,7; 0,7)	(0,8; 0,8)	(0,9; 0,9)
Krzysztof	(0,667; 0,5)	(0,45; 0,1)	(1; 0,4)	(0,9; 0,9)	(0,5; 0,5)	(0,444; 0,3)	(0,4; 0,1)
Maciej	(0,7; 0,3)	(1; 0,2)	(0,875; 0,6)	(0,7; 0,7)	(0,9; 0,9)	(1; 1)	(0,889; 0,5)
Aleksander	(0,6; 0,6)	(0,9; 0)	(0,75; 0,1)	(0,444; 0,2)	(0,8; 0,8)	(0,9; 0,9)	(0,8; 0,6)
Mateusz	(1; 1)	(0,5; 0,4)	(1; 0,8)	(1; 1)	(0,571; 0,5)	(0,5; 0,3)	(0,444; 0,2)

Tabela 6: Ograniczenia zbioru cQ.

Po kroku 3. Ostatecznie dostajemy:

	Jan	Kacper	Krzysztof	Maciej	Aleksander	Mateusz
Web Developer	0,324	0,377	0,355	0,315	**0,291**	0,465
Game Developer	0,389	0,437	**0,321**	0,47	0,359	0,421
DevOps	0,247	0,335	**0,202**	0,218	0,253	0,284
.NET Developer	0,353	0,394	0,332	0,413	**0,316**	0,446
Big Data	0,321	0,341	**0,148**	0,267	0,32	0,151

Tabela 7: Odległości między zbiorami rozmytymi (odległość Hamminga).

	Jan	Kacper	Krzysztof	Maciej	Aleksander	Mateusz
Web Developer	0,439	0,48	0,426	0,469	**0,41**	0,541
Game Developer	0,486	0,506	**0,386**	0,536	0,452	0,516
DevOps	0,334	0,396	**0,268**	0,305	0,319	0,32
.NET Developer	0,468	0,485	**0,406**	0,511	0,432	0,531
Big Data	0,348	0,394	**0,177**	0,3	0,345	0,238

Tabela 8: Odległości między zbiorami rozmytymi (odległość Euklidesowa).

Norma Łukasiewicza i odpowiadająca jej implikacja Łukasiewicza

Po kroku 1. dostajemy:

$(\blacktriangle_L^Q, \blacktriangledown_L^Q)$ pR	C++	C#	Java	Python	Linux	Javascript	PHP
Web Developer	(0,7; 0)	(1; 0,4)	(0,9; 0,1)	(0,7; 0,4)	(0,7; 0,4)	(1; 0,5)	(0,9; 0,5)
Game Developer	(0,9; 0,3)	(0,7; 0)	(0,6; 0,1)	(0,5; 0,3)	(0,3; 0,2)	(0,2; 0,1)	(0,6; 0)
DevOps	(0,8; 0,6)	(0,9; 0,3)	(1; 0,4)	(0,6; 0,6)	(1; 0,6)	(0,9; 0,3)	(0,8; 0,1)
.NET Developer	(0,8; 0,5)	(1; 0)	(0,7; 0,3)	(0,5; 0,5)	(0,4; 0,2)	(0,3; 0,1)	(0,8; 0)
Big Data	(1; 0,4)	(0,7; 0,3)	(1; 0,3)	(1; 0,8)	(0,6; 0,6)	(0,7; 0,7)	(0,6; 0,2)

Tabela 9: Ograniczenia zbioru pR.

Po kroku 2. dostajemy:

$(\blacktriangle_L^Q, \blacktriangledown_L^Q)$ cQ	C++	C#	Java	Python	Linux	Javascript	PHP
Jan	(0,6; 0,6)	(0,9; 0,7)	(0,8; 0)	(0,6; 0,1)	(1; 1)	(0,9; 0,7)	(0,8; 0,4)
Kacper	(0,4; 0,3)	(1; 0,8)	(0,6; 0,2)	(0,4; 0,4)	(0,7; 0,7)	(0,8; 0,8)	(0,9; 0,9)
Krzysztof	(0,7; 0,5)	(0,5; 0,1)	(1; 0,4)	(0,9; 0,9)	(0,5; 0,5)	(0,4; 0,3)	(0,4; 0,1)
Maciej	(0,7; 0,3)	(1; 0,2)	(0,9; 0,6)	(0,7; 0,7)	(0,9; 0,9)	(1; 1)	(0,9; 0,5)
Aleksander	(0,6; 0,6)	(0,9; 0)	(0,8; 0,1)	(0,4; 0,2)	(0,8; 0,8)	(0,9; 0,9)	(0,8; 0,6)
Mateusz	(1; 1)	(0,6; 0,4)	(1; 0,8)	(1; 1)	(0,6; 0,5)	(0,6; 0,3)	(0,5; 0,2)

Tabela 10: Ograniczenia zbioru cQ.

Po kroku 3. Ostatecznie dostajemy:

	Jan	Kacper	Krzysztof	Maciej	Aleksander	Mateusz
Web Developer	0,221	0,207	0,314	**0,179**	0,214	0,35
Game Developer	0,393	0,379	**0,257**	0,421	0,329	0,364
DevOps	**0,2**	0,3	0,207	**0,2**	0,221	0,243
.NET Developer	0,357	0,343	**0,221**	0,343	0,293	0,343
Big Data	0,3	0,314	0,164	0,243	0,293	**0,157**

Tabela 11: Odległości między zbiorami rozmytymi (odległość Hamminga).

	Jan	Kacper	Krzysztof	Maciej	Aleksander	Mateusz
Web Developer	0,282	**0,255**	0,363	0,269	0,273	0,437
Game Developer	0,462	0,476	**0,288**	0,489	0,416	0,422
DevOps	0,275	0,359	0,276	**0,262**	0,287	0,285
.NET Developer	0,439	0,446	**0,269**	0,431	0,386	0,363
Big Data	0,34	0,38	**0,195**	0,259	0,326	0,248

Tabela 12: Odległości między zbiorami rozmytymi (odległość Euklidesowa).

Wnioski

W zależności od wybranej normy z implikacją oraz sposobu liczenia odległości, otrzymano różne wyniki.

		Jan		Kacper		Krzysztof		Maciej		Aleksander		Mateusz	
		H.	E.	H.	E.	H.	E.	H.	E.	H.	E.	H.	E.
Web Developer	standardowa												
	produktowa												
	Łukasiewicza												
Game Developer	standardowa												
	produktowa												
	Łukasiewicza												
DevOps	standardowa												
	produktowa												
	Łukasiewicza												
.NET Developer	standardowa												
	produktowa												
	Łukasiewicza												
Big Data	standardowa												
	produktowa												
	Łukasiewicza												

Tabela 13: Najlepsze dopasowania kandydatów na stanowisko, w zależności od wybranej normy wraz z odpowiadającą jej implikacją oraz odległościami: H. - Hamminga, E. - Euklidesowa. Kolor zielony - najlepsze dopasowanie, kolor żółty - najlepsze ex aequo.

Można zatem wnioskować, że na następujące stanowiska, najlepszymi kandydatami są:

- **Web Developer**
  W przypadku normy produktowej, obie odległości wskazały **Aleksandra**. Wygrał on również przy normie standardowej i odległości Hamminga. Przy normie standardowej i odległości Euklidesowej zwyciężył jednak **Krzysztof**. Norma Łukasiewicza wskazała natomiast zupełnie innych kandydatów - przy odległości Hamminga **Kacpra**, a Euklidesowej **Macieja**.
- **Game Developer**
  W tym przypadku wszystkie metody wskazały **Krzysztofa**.
- **DevOps**
  Przy zastosowaniu normy standardowej i produktowej najlepszym kandydatem okazał się **Krzysztof**, jednak przy normie Łukasiewicza w przypadku odległości Euklidesowej był to **Maciej**, a Hamminga ex aequo **Maciej** z **Janem**.
- **.NET Developer**
  Korzystając z odległości Euklidesowej przy każdej normie wygrał **Krzysztof**. W przypadku odległości Hamminga przy normie Łukasiewicza również wygrał **Krzysztof**, jednak przy pozostałych normach **Aleksander**.
- **Big Data**
  W tym przypadku również korzystając z odległości Euklidesowej przy każdej normie wygrał **Krzysztof**. W przypadku odległości Hamminga przy normie produktowej również wygrał **Krzysztof**, natomiast przy standardowej ex aequo **Krzysztof** z **Mateuszem**, a przy Łukasiewicza **Mateusz**.

Podsumowując, przy jednym stanowisku - **Game Developer**, wszystkie metody jednoznacznie wskazały jednego kandydata.
Przy innych stanowiskach sytuacje się różniły. W przypadku stanowiska **DevOps**, dwie normy - standardowa i produktowa wskazały tego samego kandydata, natomiast ostatnia norma wskazała niejednoznacznie innych.
W przypadku dwóch stanowisk - **.NET Developer** i **Big Data** - Euklidesowa metoda liczenia odległości wskazały jednoznacznie Krzysztofa na obu stanowiskach, natomiast odległość Hamminga nie była w tym przypadku jednoznaczna.
Przy stanowisku **Web Developer** sytuacja była zupełnie inna. W zależności od użytej normy i odległości, wskazywani byli różni kandydaci, jednak najczęściej został wskazany jeden.

Opis techniczny rozwiązania

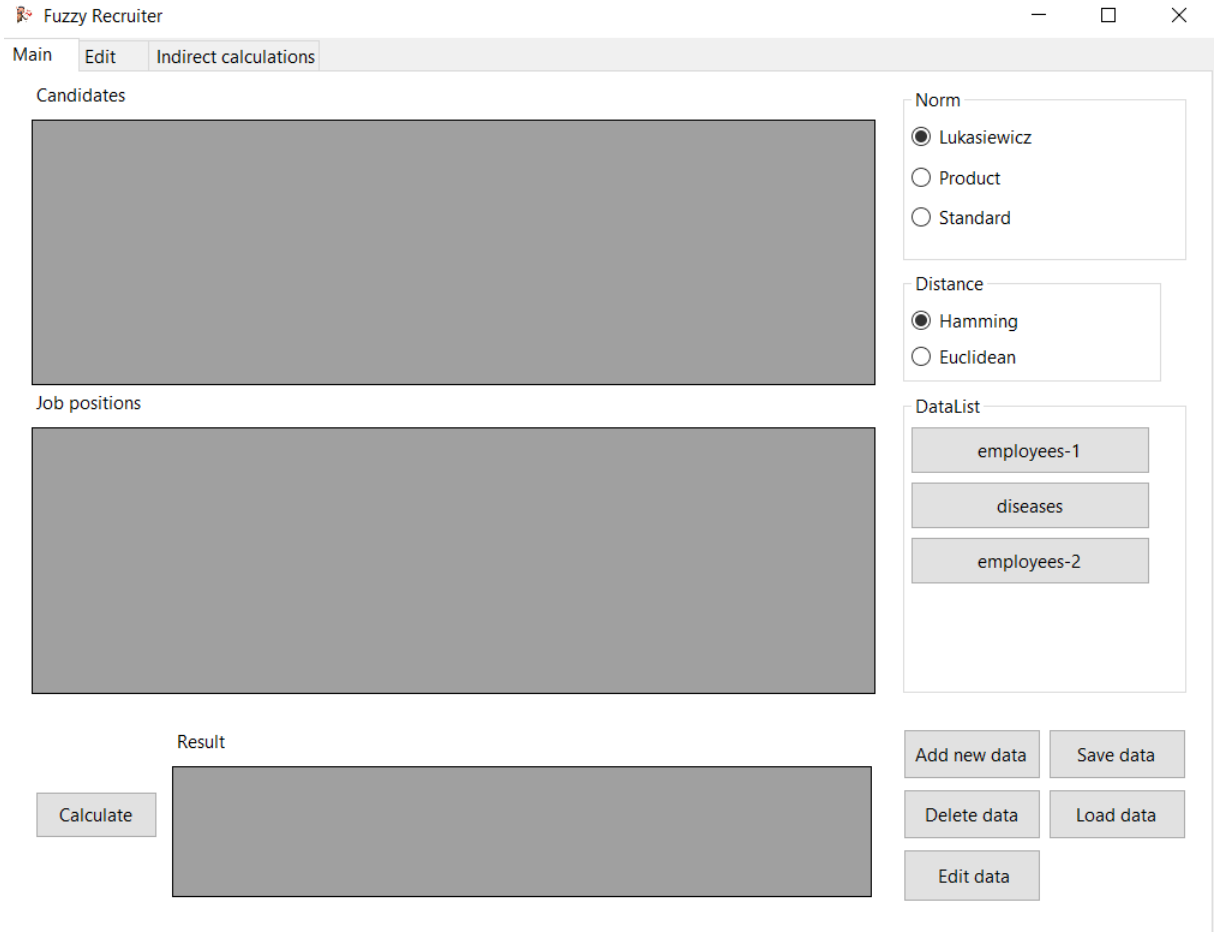
Program o nazwie *Fuzzy Recruiter* rozwiązujący zadany problem decyzyjny z zastosowaniem techniki wnioskowania rozmytego został napisany w języku C#, natomiast interfejs graficzny, przedstawiony użytkownikowi stworzono w technologii *Windows Forms*.

Program korzysta z napisanej na potrzeby zadania biblioteki *FuzzySetLib*. Zawartość biblioteki prezentuje się następująco:

- *Approximations.cs* - klasa zawierająca funkcje obliczające aproksymację dolną i górną przy użyciu rozmytej konieczności i rozmytej możliwości oraz funkcje pozwalające na wykonanie kroku 3. algorytmu:
  - struktura *Bounds* została stworzona w celu reprezentacji ograniczeń zbioru,
  - funkcja *Transpose* dla relacji R wyznacza R^{-1},
  - funkcja *FuzzyPossibility* dla danej normy, relacji Q, zbioru A oraz elementu x, wyznacza $(\langle Q \rangle_{\otimes} A)(x)$,
  - funkcja *FuzzyNecessity* dla danej implikacji, relacji Q, zbioru A oraz elementu x, wyznacza $([Q]_{\otimes} A)(x)$,
  - funkcja *ObjectUpperApprox* dla danej normy, implikacji, relacji Q oraz zbioru A, wyznacza $(\blacktriangle_{\otimes}^{Q} A)$,
  - funkcja *ObjectLowerApprox* dla danej normy, implikacji, relacji Q oraz zbioru A, wyznacza $(\blacktriangledown_{\otimes}^{Q} A)$,
  - funkcja *PropertyBoundApproximation* dla danej normy, implikacji oraz relacji Q, wyznacza $(\blacktriangle_{\otimes}^{Q}, \blacktriangledown_{\otimes}^{Q})$ zbiorów cQ,
  - funkcja *ObjectBoundApproximation* dla danej normy, implikacji, relacji Q oraz relacji R wyznacza $(\blacktriangle_{\otimes}^{Q}, \blacktriangledown_{\otimes}^{Q})$ zbiorów pR,
  - funkcja *FuzzyRelationBasedApproximation* dla danej normy, implikacji, odległości, relacji R oraz relacji Q, wyznacza macierz odległości pomiędzy zbiorami rozmytymi,
- *Distances.cs* - klasa zawiera funkcje obliczające odległości Hamminga i Euklidesową,
- *Implications.cs* - klasa definiująca implikacje: Gödela, Goguen-Gainesa i Łukasiewicza,
- *Norms.cs* - klasa definiująca normy: standardową, produktową i Łukasiewicza.

Funkcjonalności biblioteki *FuzzySetLib* zostały przetestowane przy użyciu testów jednostkowych. Znajdują się one w projekcie *FuzzySetLibTests*.

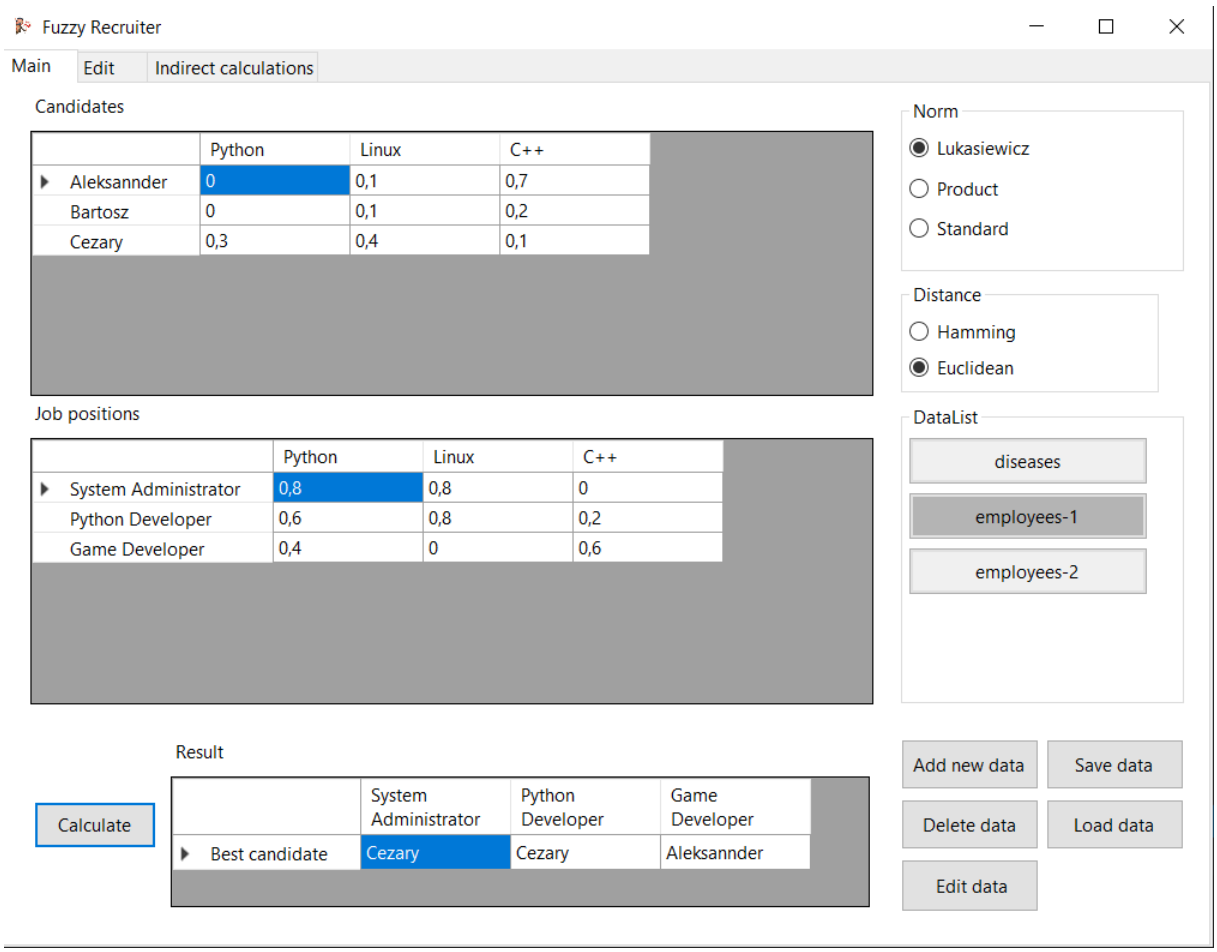
Fuzzy Recruiter zawiera przyjazny użytkownikowi i prosty w obsłudze interfejs graficzny.



Aplikacja składa się z trzech widoków udostępnionych użytkownikowi w zakładkach:

- *Main*,
- *Edit*,
- *Indirect calculations*.

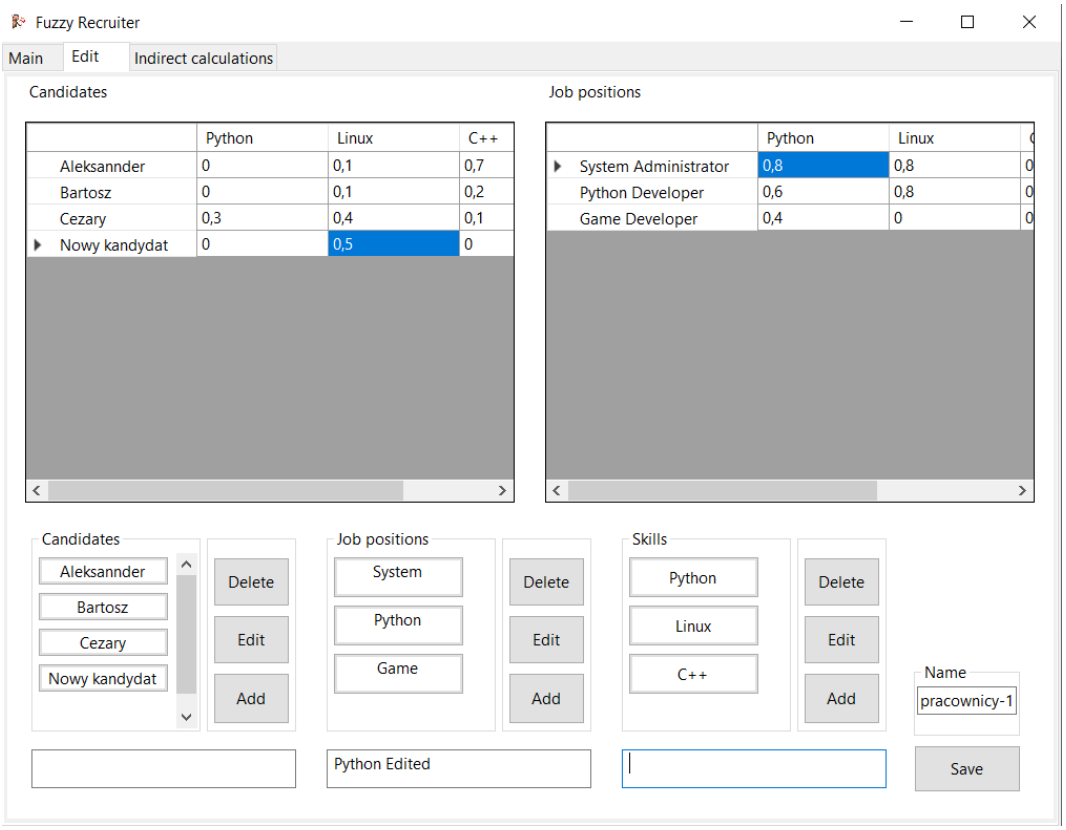
Zakładka **Main** jest domyślnym widokiem programu. Użytkownik ma możliwość wyboru normy (w panelu *Norm*) oraz odległości (w panelu *Distance*), przy pomocy których mają być wykonywane niezbędne obliczenia. W panelu *DataList* dostępne są przyciski do wczytania przykładowych danych wejściowych, które można użyć do obliczeń - tu również pojawi się opcja wyboru danych stworzonych przez użytkownika, jeśli tylko zapisze zmiany lokalnie w programie. Po kliknięciu odpowiedniego przycisku, do obszarów *Candidates* i *Job positions* zostają załadowane dane na temat umiejętności kandydatów i wymagań na poszczególne stanowiska.



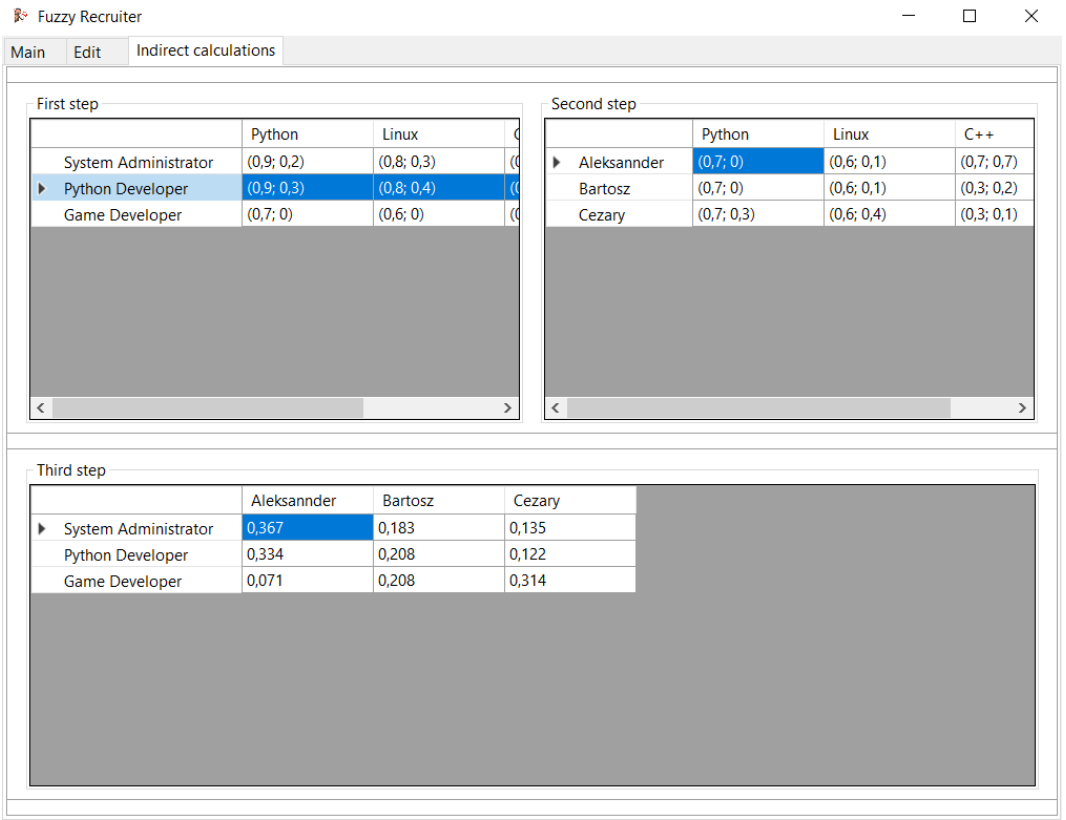
W panelu **Main** dostępne są także następujące funkcje:

- *Calculate* - obliczanie najlepszych kandydatów na stanowiska na podstawie wczytanych danych,
- *Add new data* - stworzenie nowych danych w zakładce edycji **Edit**,
- *Delete data* - lokalne usunięcie wybranego zestawu danych z listy,
- *Edit data* - edytowanie wybranego zestawu danych w zakładce **Edit**,
- *Save data* - zapisanie zestawu danych na dysku jako pliku *.json*,
- *Load data* - wczytanie zestawu danych z dysku.

Zakładka **Edit** umożliwia zarządzanie tabelami z danymi na temat stanowisk, kandydatów oraz ich umiejętności. Wybrane rekordy z danymi można usuwać i edytować, a także dodawać nowe. Kliknięcie przycisku *Save* umożliwia zapisanie edytowanych danych lub dopiero co utworzonych lokalnie w programie.



Po wykonaniu obliczeń dopasowujących kandydatów na stanowiska w zakładce **Indirect calculations** dostępne są obliczenia pośrednie dla trzech kroków opisanego w poprzednich rozdziałach algorytmu wnioskowania rozmytego. Funkcjonalność ta jest przydatna podczas analizy danych lub wyszukiwania potencjalnych błędów w obliczeniach „ręcznych".



Prawidłowe działanie aplikacji jest usprawnione przez system prostych ostrzeżeń np. w przypadku niezaładowania danych lub niewprowadzenia wymaganej wartości w pewnym polu. Dodatkowo w programie zapewniono responsywność, co znacznie zwiększa czytelność interfejsu użytkownika.

Literatura

[1] Anna M. Radzikowska, *Fuzzy relation-based approximation techniques in supporting medical diagnosis*, Journal of Automation, Mobile Robotics & Intelligent Systems 11, no 1, pp. 21–29, 2017.

[2] Anna M. Radzikowska, Etienne E Kerre, *A comparative study off fuzzy rough sets*, Fuzzy Sets and Systems 126, pp. 137–155, 2002.

[3] T. Calvo, R. Mesiar, *Continuous generated associative aggregation operators*, Fuzzy Sets and Systems, Volume 126, Issue 2, 2002, Pages 191-197, [link].

[4] Harpreet Singh, Madan M. Gupta, Thomas Meitzler, Zeng-Guang Hou, Kum Kum Garg, Ashu M. G. Solo, Lotfi A. Zadeh, *Real-Life Applications of Fuzzy Logic*, Advances in Fuzzy Systems, vol. 2013, Article ID 581879, 3 pages, 2013, [link].