

Systemy Sztucznej Inteligencji

dokumentacja projektu *Przewidywanie cukrzycy*

Kamil Giziński, Bartosz Rolnik, Dominik Sigulski
Informatyka, semestr IV
Wydział Matematyki Stosowanej
Politechnika Śląska

10 czerwca 2022

Spis treści

1	Część I	2
1.1	Opis programu	2
1.2	Instrukcja obsługi	2
1.3	Analiza bazy danych	3
1.4	Wprowadzenie od strony medycznej	7
1.5	Dodatkowe informacje	9
2	Część II	10
2.1	Opis działania	10
2.2	Implementacja	13
2.2.1	Klasyfikator miękki nr 1 - Klasa SoftSetClassifierMean	13
2.2.2	Klasyfikator miękki nr 2 - Klasa SoftSetClassifierPercentage	16
2.2.3	Klasyfikator rozmyty - Klasa Fuzzy	19
2.3	Testy	28
2.3.1	Test metod klasy ProcessingData	28
2.3.2	Test metod klasy NaiveBayes	31
2.3.3	Test metod klasy KNN	32
2.3.4	Test metod klasy SoftSetClassifierMean	33
2.3.5	Test metod klasy SoftSetClassifierPercentage	34
2.4	Eksperymenty	35
3	Pełen kod aplikacji	45
3.1	Klasa ProcessingData	45
3.2	Klasa NaiveBayes	47
3.3	Klasa KNN	50
3.4	Klasa SoftSetClassifierMean	52
3.5	Klasa SoftSetClassifierPercentage	55
3.6	Klasa FuzzyClassifier	58
3.7	Dodawanie poprzedników i konsekwencji	61
3.8	Dodawanie reguł	62
3.9	Testowanie klasyfiaktora zbioru rozmytego	106

1 Część I

1.1 Opis programu

Klasyfikator miękki dla przewidywania cukrzycy u pacjentów.

W projekcie należy zaproponować przynajmniej trzy różne sposoby na tworzenie tabeli zbioru miękkiego na podstawie zbioru treningowego oraz porównać otrzymane wyniki.

Poprawka 1:

Po konsultacji z prowadzącym laboratorium zdecydowaliśmy się na stworzenie dwóch klasyfikatorów zbiorów miękkich i jednego klasyfikatora zbiorów rozmytych.

1.2 Instrukcja obsługi

Podstawą projektu jest baza danych zawierająca dane dotyczące stanu zdrowia pacjentów. W szczególności wszyscy pacjenci w tym zbiorze danych to kobiety w wieku co najmniej 21 lat pochodzące z indiańskiego plemienia Pima. Dane zostały zebrane przez Narodowy Instytut Cukrzycy oraz Chorób Układu Pokarmowego i Nerek (ang. National Institute of Diabetes and Digestive and Kidney Diseases - NIDDK) w Stanach Zjednoczonych. Bazę można pobrać ze strony [kaggle.com](https://www.kaggle.com).

Kolumny zbioru danych to kolejno:

1. **Pregnancies** (ciąże) — liczba ciąż danej pacjentki;
2. **Glucose** (glukoza) — stężenie glukozy w osoczu w doustnym teście tolerancji glukozy;
3. **BloodPressure** (ciśnienie krwi) — rozkurczowe ciśnienie krwi (mm Hg);
4. **SkinThickness** (grubość skóry) — grubość fałdu skórniego na tricepsie (mm);
5. **Insulin** (insulina) — insulina w surowicy 2-godzinnej (mj./ml);
6. **BMI** — wskaźnik masy ciała (waga w kg / (wzrost w m)²);
7. **DiabetesPedigreeFunction** (funkcja metryki cukrzycy) — funkcja oceniająca prawdopodobieństwo wystąpienia cukrzycy na podstawie wywiadu rodzinnego;
8. **Age** (wiek) — wiek pacjentki w latach;
9. **Outcome** (wynik) — zmienna klasy (0 lub 1). 268 z 768 to 1, pozostałe to 0;

Program uruchamiamy za pomocą środowiska Jupyter. Po jego uruchomieniu program zacznie wyliczać dokładność wszystkich klasyfikatorów, które poznaliśmy w ciągu tego semestru, oraz tych, które sami zaprogramowaliśmy. Wszystkie wyżej wymienione algorytmy służą do przewidywania na podstawie zebranych danych, czy pacjent choruje na cukrzycę, czy jest zdrowy. Na samym początku wyliczana jest dokładność klasyfikatorów: Bayesa oraz KNN. Następnie program wyliczy dokładność dwóch zaprojektowanych przez nas klasyfikatorów miękkich. W osobnym pliku znajduje się Klasyfikator wnioskujący ze zbioru rozmytego,

którego dokładność jest wyświetlana przy pomocy macierzy pomyłek. To oznacza, że otrzymujemy dokładne dane, jaki procent pacjentów został zdiagnozowanych poprawnie jako chorzy, fałszywie jako chorzy, poprawnie jako zdrowi oraz fałszywie jako zdrowi.

1.3 Analiza bazy danych

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Pregnancies                          768 non-null    int64
1   Glucose                             768 non-null    int64
2   BloodPressure                       768 non-null    int64
3   SkinThickness                      768 non-null    int64
4   Insulin                            768 non-null    int64
5   BMI                                768 non-null    float64
6   DiabetesPedigreeFunction            768 non-null    float64
7   Age                                768 non-null    int64
8   Outcome                             768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

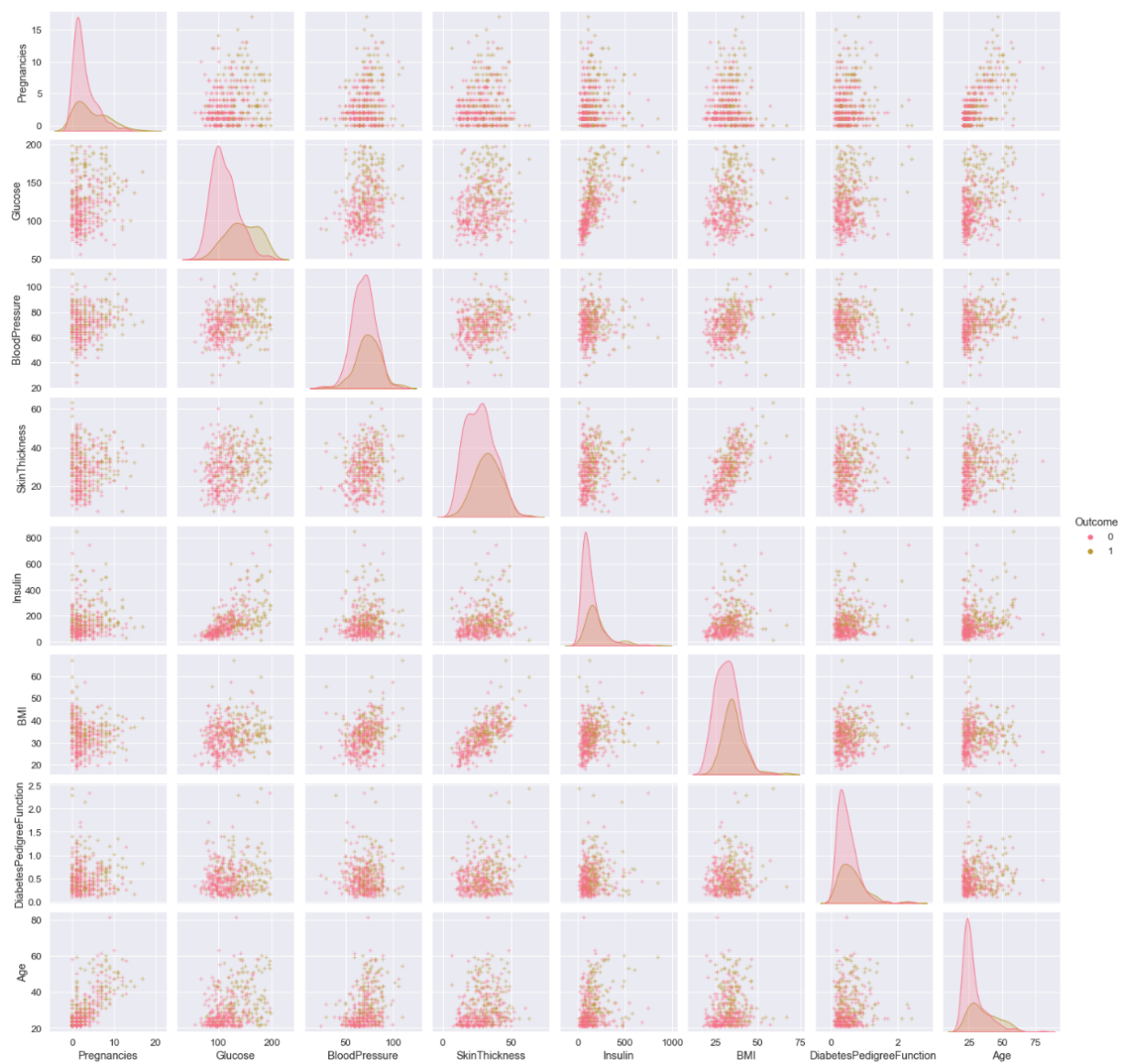
Rysunek 1: Wyświetlenie podstawowych informacji o bazie

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

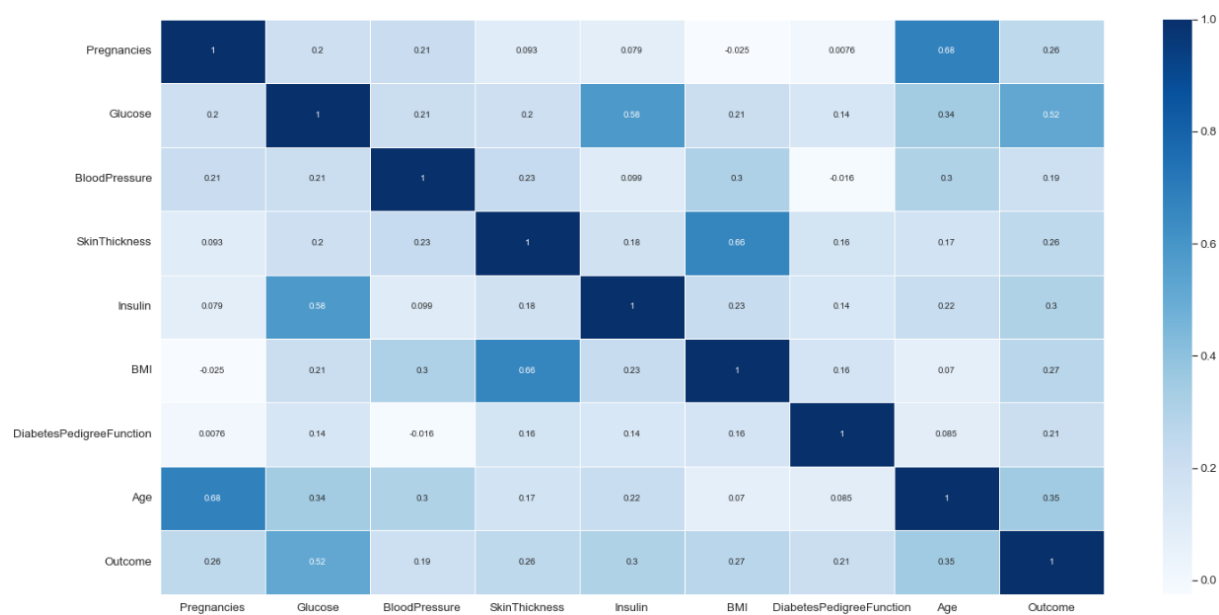
Rysunek 2: Wyświetlenie podstawowej statystyki bazy

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000
mean	3.301020	122.627551	70.663265	29.145408	156.056122	33.086224	0.523046	30.864796	0.331633
std	3.211424	30.860781	12.496092	10.516424	118.841690	7.027659	0.345488	10.200777	0.471401
min	0.000000	56.000000	24.000000	7.000000	14.000000	18.200000	0.085000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	21.000000	76.750000	28.400000	0.269750	23.000000	0.000000
50%	2.000000	119.000000	70.000000	29.000000	125.500000	33.200000	0.449500	27.000000	0.000000
75%	5.000000	143.000000	78.000000	37.000000	190.000000	37.100000	0.687000	36.000000	1.000000
max	17.000000	198.000000	110.000000	63.000000	846.000000	67.100000	2.420000	81.000000	1.000000

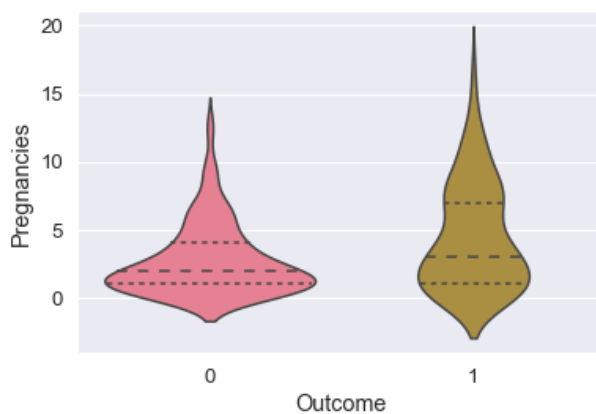
Rysunek 3: Wyświetlenie podstawowej statystyki bazy z wyeliminowanymi niektórymi wartościami równymi 0



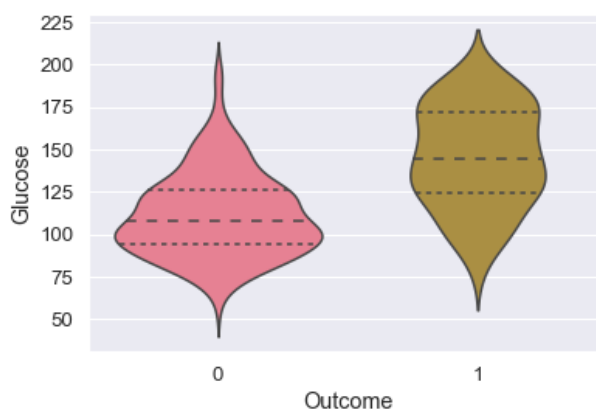
Rysunek 4: Wykres rozkładu Wyników dla par cech



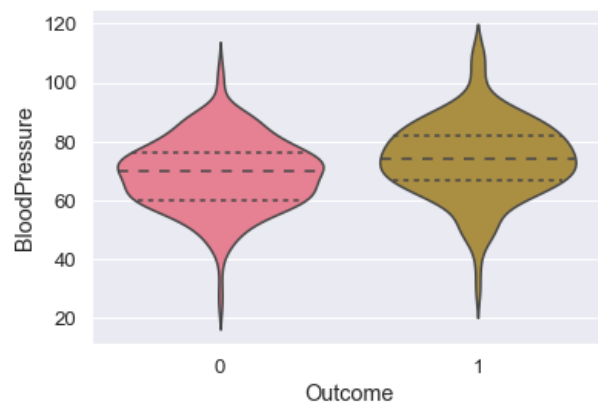
Rysunek 5: Wykres korelacji wartości



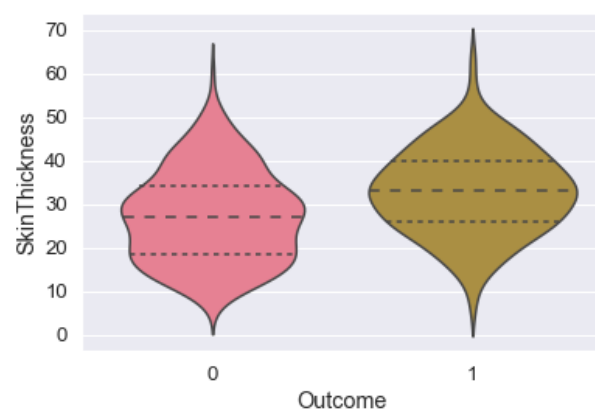
Rysunek 6: Rozkład Cięża - Wynik



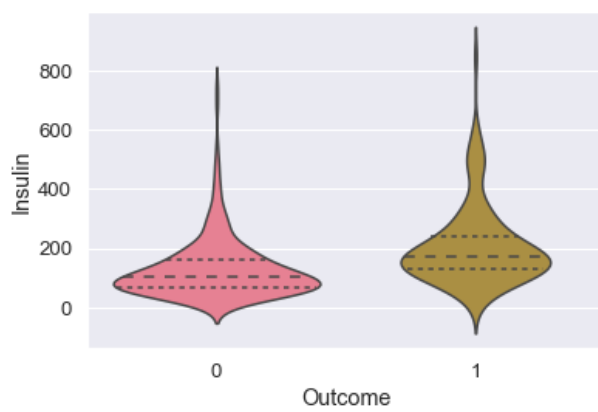
Rysunek 7: Rozkład Glukoza - Wynik



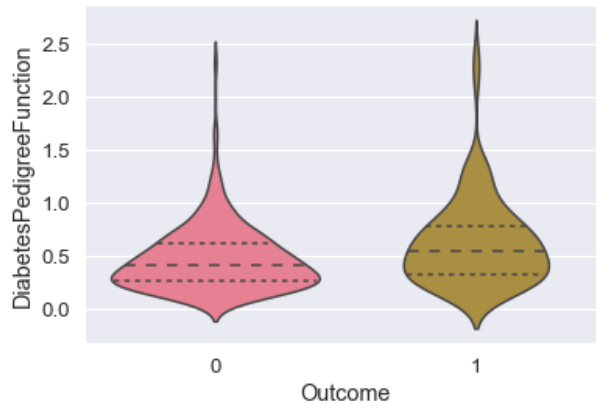
Rysunek 8: Rozkład CiśnienieKrw - Wynik



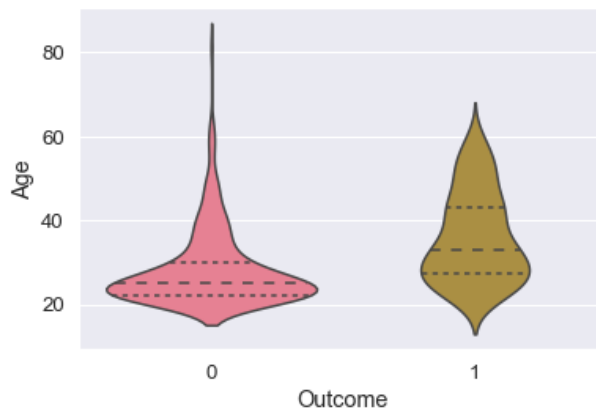
Rysunek 9: Rozkład GrubośćSkóry - Wynik



Rysunek 10: Rozkład Insulina - Wynik



Rysunek 11: Rozkład Funkcja metryki cukrzycy - Wynik



Rysunek 12: Rozkład Wiek - Wynik

1.4 Wprowadzenie od strony medycznej

W tej części zajmiemy się przybliżeniem informacji o kolumnach w zbiorze danych, z którego korzystamy.

Głukoza to cukier prosty, który jest podstawowym źródłem energii w organizmie człowieka. Oznaczenie jej stężenia we krwi (czyli glikemii) na czczo pomaga określić, czy pacjent ma cukrzycę.

Orientacyjna norma stężenia glukozy we krwi wynosi:

- **70-99 mg/dl** — wartości prawidłowe
- **100-125 mg/dl** — wartości nieprawidłowe, wymagają wykonania doustnego testu tolerancji glukozy
- **ponad 126 mg/dl** — wartości nieprawidłowe, wymagają powtórzenia badania, po dwukrotnym otrzymaniu takiego wyniku rozpoznaje się cukrzycę

Kolejna kolumna to **ciśnienie krwi**. Ciśnienie tętnicze to siła nacisku, jaką płynąca krew wywiera na ściany tętnic. Pomiaru dokonuje się za pomocą ciśnieniomierza, a otrzymane wartości przedstawia się w mm Hg, czyli milimetrach słupa rtęci. Prawidłowe ciśnienie krwi jest niższe niż 90 mm Hg.

Ciśnienie tętnicze w zakresie normy dzieli się na:

- **normalne:** <80 mm Hg;
- **podwyższone faza 1:** 80–89 mm Hg;
- **podwyższone faza 2:** >90 mm Hg.

Grubość fałdu skórno na tricepsie jest jednym z wyznaczników wartości tkanki tłuszczowej organizmu. Według danych Amerykańskiej agencji CDC ([Basic data on anthropometric measurements and angular measurements of the hip and knee joints for selected age groups 1-74 years of age](#)) na podstawie badań w latach 1971-1975, mediana grubości fałdu skórno na tricepsie dla kobiet nie przekracza **25 mm**. Jednakowoż nie udało nam się znaleźć informacji na temat konkretnych norm (i czy takie normy dla całej populacji w ogóle istnieją).

Insulina to hormon, który obniża poziom cukru we krwi. Jej norma 2 godziny po posiłku to **do 30 mIU/ml** (za: www.doz.pl). W zbiorze danych, z którego korzystamy, wartości zdecydowanie wykraczają poza normę - mediana wynosiła 125.5 mU/ml, co może sugerować użycie innych jednostek, a nie jedynie literówkę w ich nazwie. Nie udało się nam znaleźć normy dla jednostek 'mU/ml'. Podwyższone stężenie insuliny zazwyczaj jest objawem cukrzycy typu 2, natomiast wynik poniżej normy wskazuje na cukrzycę typu 1. Wyniki, które przekraczają normę lub znajdują się poniżej przyjętych wartości, wymagają konsultacji u diabetologa.

Funkcja metryki cukrzycy

Niestety nie udało nam się dotrzeć do szczegółów dotyczących wzoru Funkcji metryki cukrzycy. Jak podaje strona [kaggle.com](https://www.kaggle.com), jest to "funkcja oceniająca prawdopodobieństwo wystąpienia cukrzycy na podstawie wywiadu rodzinnego". Możemy się jedynie domyślać, że im wyższa wartość, tym większe ryzyko.

BMI to wskaźnik masy ciała. Jest wskaźnikiem, który jest obliczany przez porównanie wzrostu z masy ciała. Jego wartość jest pomocna w ocenie ryzyka wystąpienia chorób związanych z nadwagą takich jak miażdżycy, cukrzycy lub choroba niedokrwienna serca. Im mniejsza wartość BMI, tym ryzyko wystąpienia chorób jest mniejsze.

Normy BMI za [CDC](#):

- **niedowaga:** <18.5;

- **waga prawidłowa:** 18.5 - 24.9;
- **nadwaga:** 25.0 - 29.9;
- **otyłość:** >30;

Wiek

Jak podaje amerykańska agencja CDC, pacjent jest w grupie ryzyka, jeśli jego wiek wynosi co najmniej **45 lat**.

1.5 Dodatkowe informacje

Wymagania:

- Zaproponowanie dwóch klasyfikatorów zbiorów miękkich i jednego klasyfikatora zbioru rozmytego;
- Porównanie wyników wszystkich klasyfikatorów.

2 Część II

2.1 Opis działania

- **ProcessingData:**

Jest to klasa, która służy do prawidłowego sformatowanie bazy danych. Posiada ona metodę do tasowania zbioru, do normalizacji próbek oraz finalnie do podzielenia bazy na zbiór treningowy i walidacyjny (w naszym programie wszystkie zbiory będą dzielone w proporcjach 70:30).

- **Naiwny Klasyfikator Bayesa:**

Naiwny klasyfikator Bayesowski, bazujący na twierdzeniu Bayesa, nadaje się szczególnie do problemów o bardzo wielu wymiarach na wejściu. W wyniku działania algorytmu otrzymujemy listę prawdopodobieństw przynależności obiektu do danych klasy. Ostatecznym rezultatem klasyfikacji jest wartość zwrócona przez metodę maksimum prawdopodobieństwa spośród wyników dla poszczególnych klas.

- **Klasyfikator KNN:**

Przy użyciu tego algorytmu przydzielamy dany obiekt do klasy, gdzie należy największa liczba jego sąsiadów, gdzie liczba sąsiadów jest z góry ustaloną liczbą 'k'. W przypadku naszego programu do liczenia odległości między obiektami wykorzystaliśmy metrykę euklidesową.

- **Zbiory miękkie:**

Zbiory miękkie są w rzeczywistości uogólnieniem zbiorów rozmytych (pewnym ich uproszczeniem), zostały stworzone, żeby poradzić sobie z niepewnością w sposób parametryczny. Zbiór miękki to sparametryzowana rodzina zbiorów – intuicyjnie jest to „miękkie”, ponieważ granica zbioru zależy od parametrów.

- **Zbiory rozmyte:**

W matematyce zbiory rozmyte (tzw. zbiory niepewne) to zbiory, których elementy mają stopnie przynależności i jest to rozszerzenie klasycznego pojęcia zbioru. W klasycznej teorii mnogości przynależność elementów do zbioru ocenia się zero-jedynkowo — element albo należy, albo nie należy do zbioru. Z kolei teoria zbiorów rozmytych pozwala na stopniową ocenę przynależności elementów do zbioru. Oceny przynależności elementu dokonuje się przy pomocy funkcji przynależności o wartości w rzeczywistym przedziale jednostkowym $[0, 1]$.

- **Wzory:**

(1) ProcessingData - normalizacja std

$$\frac{x - \bar{x}}{\sigma}$$

* σ - odchylenie standardowe

* \bar{x} - średnia

(2) ProcessingData - normalizacja minmax

$$\frac{x - \min}{\max - \min}$$

- * \min - wartość minimalna
- * \max - wartość maksymalna

(3) NaiveBayes - twierdzenie Bayesa

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

- * A, B - zdarzenia
- * $P(A|B)$ - oznacza prawdopodobieństwo warunkowe, tj. prawdopodobieństwo zajścia zdarzenia A , o ile zajdzie zdarzenie B
- * $P(B|A)$ - oznacza prawdopodobieństwo zajścia zdarzenia B , o ile zajdzie zdarzenie A

(4) NaiveBayes - gęstość (Gauss)

$$\frac{1}{\sigma\sqrt{2\pi}} * e^{-\frac{(x-m)^2}{2\sigma^2}}$$

- * σ - odchylenie standardowe
- * x - wynik obserwacji
- * m - wartość oczekiwana zmiennej

(5) KNN - Euklidesowa odległość między wektorami

$$\sqrt{\sum_{i=1}^n |x_i - y_i|^2}$$

- * x_i - współrzędna wektora x
- * y_i - współrzędna wektora y

(6) Rozmyte - funkcja typu trójkąt

$$\mu_{\text{triangular}}(x; a, b, c) = \begin{cases} 0, & \text{if } x \leq a. \\ \frac{x-a}{b-a}, & \text{if } a < x \leq b. \\ \frac{c-x}{c-b}, & \text{if } b < x \leq c. \\ 0, & \text{if } c < x. \end{cases}$$

- * a - początek podstawy trójkąta, członkostwo przyjmuje wartość 0

- * b - wysokość trójkąta, członkostwo przyjmuje wartość 1
- * c - długość podstawy trójkąta, członkostwo przyjmuje wartość 0

(7) Rozmyte - funkcja typu trapez

$$\mu_{trapezoidal}(x; a, b, c, d) = \begin{cases} 0, & \text{if } x \leq a. \\ \frac{x-a}{b-a}, & \text{if } a < x \leq b. \\ 1, & \text{if } b < x \leq c. \\ \frac{d-x}{d-c}, & \text{if } c < x \leq d. \\ 0, & \text{if } d < x. \end{cases}$$

- * a - początek podstawy trapezu, członkostwo przyjmuje wartość 0
- * zakres od b do c reprezentuje najwyższą wartość członkostwa równą 1
- * d - długość podstawy dolnej trapezu, członkostwo przyjmuje wartość 0

(8) Stopień spełnienia przesłanki

$$\mu_i = \max\{ssp\}$$

- * ssp - stopnie spełnienia etykiet przesłanki (np.: $ssp=(0,6, 0,75)$)

(9) Stopień spełnienia reguły dla wszystkich przesłanek:

$$\mu_{cale}(x) = \min\{\mu_1, \mu_2, \dots, \mu_i\}$$

- * μ_i - stopień spełnienia danej przesłanki

(10) Środek ciężkości trójkąta

$$S = \frac{a + b + c}{3}$$

- * a, b, c - współrzędne wierzchołków w trójkącie

(11) Środek ciężkości figury

$$h = \frac{\sum_i \mu_i A_i c_i}{\sum_i \mu_i A_i}$$

- * A_i - powierzchnia zbioru i
- * μ_i - stopień przynależności zbioru i
- * c_i - środek ciężkości zbioru i

Przykłady zasad dla zbiorów rozmytych:

```

1      # Same niskie wartosc lub niskie | srednie sprawiaja ze nie
      przewidujemy cukrzycy
2      fuzzy.add_rule({'Pregnancies': 'low | medium',
3                      'Glucose': 'low | medium_low | medium',
4                      'BloodPressure': 'low | medium_low | medium',
5                      'SkinThickness': 'low | medium_low | medium',
6                      'Insulin': 'low | medium',
7                      'BMI': 'underweight | healthy_weight',
8                      'DiabetesPedigreeFunction': 'low | medium',
9                      'Age': 'low | medium',
10                     'Outcome': 'low'})
11
12     # Wartosci oscylujace miedzy niskimi i srednimi z wyjatkiem
      grubosci skory i ilosci ciaz czy wieku wciaz sa przewidywane
      jako brak cukrzycy
13     fuzzy.add_rule({'Pregnancies': 'medium | high',
14                     'Glucose': 'low | medium_low | medium',
15                     'BloodPressure': 'low | medium_low | medium',
16                     'SkinThickness': 'medium | medium_high | high',
17                     'Insulin': 'low | medium',
18                     'BMI': 'underweight | healthy_weight',
19                     'DiabetesPedigreeFunction': 'low | medium',
20                     'Age': 'medium | high',
21                     'Outcome': 'low'})
22
23     # sporo srednich, srednich niskich lub wysokich czynnikow
      sklaniaja nas do przewidzenia cukrzycy
24     fuzzy.add_rule({'Pregnancies': 'medium | high',
25                     'Glucose': 'medium | medium_high | high',
26                     'BloodPressure': 'low | medium_low | medium',
27                     'SkinThickness': 'low | medium_low | medium',
28                     'Insulin': 'low | medium',
29                     'BMI': 'underweight | healthy_weight',
30                     'DiabetesPedigreeFunction': 'medium | high',
31                     'Age': 'medium | high',
32                     'Outcome': 'high'})

```

2.2 Implementacja

2.2.1 Klasyfikator miękki nr 1 - Klasa SoftSetClassifierMean

Klasa SoftSetClassifierMean zawiera klasyfikator miękki korzystający ze średniej arytmetycznej oraz metody pomocnicze.

```

1  # lista etykiet
2  labels: list[str] = []
3  # lista par 0, 1 dla kazdej kolumny i etykiety
4  pairs: list[list[list[int, int]]] = []
5  # lista srednich dla kazdej kolumny i etykiety
6  means: list[list[float]] = []

```

Opis list pomocniczych *labels*, *pairs*, *means* przechowujących kolejno etykiety, pary 0, 1 w odpowiedniej kolejności dla każdej kolumny i wiersza, średnie wartości każdej kolumny i etykiety.

```

1 @staticmethod
2 def mean(column: list[float]) -> float:
3     """
4     Receives a column of the dataframe with numerical values and returns
5     its mean value
6     :param column: list[float]
7     :return mean value of the values in a dataframe column: float
8     """
9     # czyli zwykła suma przez ilość składników
10    return sum(column) / len(column)

```

Metoda pomocnicza obliczająca średnie wartości kolumn. Czyli suma wartości dzielona przez ilość wartości.

```

1 def calculate(self, df: pd.DataFrame, label: str) -> None:
2     """
3     Receives a dataframe and the label of the column we want to predict
4     and decides whether pairs are [0,1] or [1,0]
5     :param df: pd.DataFrame
6     :param label: str
7     :return: None
8     """
9     self.pairs = []
10    self.means = []
11    # lista unikalnych wartosci z podanej kolumny 'label'
12    # ma byc zawsze ta sama kolejnosc wiec sortujemy
13    self.labels = sorted(df[label].unique())
14    # 'category' to 0 i 1
15    for category in self.labels:
16        # wartosci ktorych unikalna wartosc kolumny 'label' to 'category'
17        # czyli podzial na czesci ktore maja 0 i 1
18        category_values = df[df[label] == category].drop(labels=label,
19        axis=1)
20        temp_mean = []
21        temp_pair = []
22        # item to wszystkie wartosci z danej kolumny
23        for _, item in category_values.items():
24            # srednia z kolumny, dodajemy ja do listy srednich
25            mean = self.mean(item)
26            temp_mean.append(mean)
27            len_lower = 0
28            len_upper = 0
29            # zliczamy elementy wieksze od sredniej i mniejsze od
30            # sredniej
31            for value in item:
32                if value > mean:
33                    len_upper += 1
34                else:
35                    len_lower += 1
36
37        # jesli jest tyle samo powyzej sredniej co ponizej to

```

```

35         # wybieramy losowo pare [0, 1] lub [1, 0]
36         if len_upper == len_lower:
37             temp_pair.append(random.choice(([0, 1],[1, 0])))
38         # jesli wiecej powyzej sredniej to zapisujemy pare [1, 0]
39         elif len_upper > len_lower:
40             temp_pair.append([1, 0])
41         # jesli wiecej ponizej sredniej to zapisujemy pare [0, 1]
42         else:
43             temp_pair.append([0, 1])
44
45         # uzupełniamy listy wszystkich 'srednich' i wszystkich 'par'
46         self.means.append(temp_mean)
47         self.pairs.append(temp_pair)

```

Metoda tworząca tabelę ważoną zbioru miękkiego. Dzielimy zbiór na część, w której Wynik badania oznacza diabetyka oraz część, w której Wynik badania oznacza osobę zdrową. Dla każdej kolumny w każdej części obliczamy wartość średnią. Zliczamy ilość elementów większych od średniej oraz elementów mniejszych od średniej. Tworzymy tabelę ważoną zbioru z par [0, 1] i [1, 0]. Pary [0, 1] dodajemy do tabeli ważonej w przypadku, gdy jest więcej elementów poniżej średniej. Pary [1, 0] dodajemy do tabeli ważonej w przypadku, gdy jest więcej elementów powyżej średniej. Jeżeli elementów jest tyle samo, to losujemy czy dodać parę [1, 0] czy [0, 1].

```

1 def predict(self, sample: pd.Series, label: str) -> str:
2     """
3     Receives a sample in the form of pd.Series and returns its predicted
4     label
5     :param sample: pd.Series
6     :param label: str
7     :return predicted label of the sample: str
8     """
9
10    # liczenie elementow dla sampla dla kazdej z klas (0, 1)
11
12    sample = sample.drop(labels=label)
13    probabilities = []
14    # bierzemy po kolei odpowiednie pary i srednie
15    for category, mean in zip(self.pairs, self.means):
16        sample_pairs = []
17        # bierzemy odpowiednia srednia i wartosc z sample'a
18        for m, x in zip(mean, sample):
19            # jesli element rowny sredniej
20            # wybieramy losowo pare [0, 1] lub [1, 0]
21            if x == m:
22                sample_pairs.append(random.choice(([0, 1],[1, 0])))
23            # jesli element wiekszy od sredniej to zapisujemy pare [1,
24            # 0]
25            elif x > m:
26                sample_pairs.append([1, 0])
27            # jesli element mniejszy od sredniej to zapisujemy pare [0,
28            # 1]
29            else:
30                sample_pairs.append([0, 1])
31
32    temp = []

```



```

28
29         # liczenie prawdopodobienstwa dla kazdej klasy
30         for x, y in zip(category, sample_pairs):
31             temp.append(x[0] * y[0] + x[1] * y[1])
32         probabilities.append(sum(temp))
33
34     # naszym indeksem jest indeks największego prawdopodobienstwa
35     idx = max(range(len(probabilities)), key=probabilities.__getitem__)
36     # zwracamy najprawdopodobniejszy wynik z 'labels' (u nas 0 lub 1)
37     return self.labels[idx]

```

Metoda przewidująca etykietę klasy próbki. Znowu tworzymy listę par [0, 1] i [1, 0] jednak teraz bierzemy pod uwagę odpowiednią (wcześniej obliczoną) średnią oraz kolejne elementy z kolejnych kolumn w naszej próbce. Schemat dobierania par jest identyczny do wyżej opisanego podczas opisu metody *calculate()*. Następnie liczymy prawdopodobieństwo wystąpienia każdej etykiety. Opiera się to na zwykłym iloczynie skalarnym wektorów. W tym przypadku naszymi 'wektorami' są indeksy etykiet oraz pary [0, 1] i [1, 0]. Na koniec zwracamy przewidywaną etykietę (czyli tę, która uzyskała największą wartość na liście prawdopodobieństw).

```

1 def test(self, validation_set: pd.DataFrame, label: str) -> None:
2     """
3     Test how accurate the prediction method is
4     :param validation_set: pd.DataFrame - a record from the validation
      dataset
5     :param label: str
6     :return: None
7     """
8     correct = 0
9     # iterrows() jest uzywane do iteracji wiesz dataframe'a
10    for _, row in validation_set.iterrows():
11        # jesli poprawnie przewidzimy etykiety
12        if self.predict(sample=row, label=label) == row[label]:
13            correct += 1
14    # informacja podsumowujaca
15    # print(f'Accuracy: {correct / len(validation_set) * 100:.2f}%; {
      correct}/{len(validation_set)}')
16    return correct / len(validation_set) * 100

```

Metoda testująca dokładność działania klasyfikatora miękkiego. Prosty mechanizm, który w pętli wywołuje metodę *predict()* oraz porównuje wynik z wynikiem poprawnym. Zwrócona zostaje skuteczność przewidywania w procentach.

2.2.2 Klasyfikator miękki nr 2 - Klasa `SoftSetClassifierPercentage`

Klasa `SoftSetClassifierPercentage` zawiera klasyfikator miękki korzystający z procentów oraz metody pomocnicze.

```

1 # lista etykiet
2 labels: list[str] = []
3 # lista par 0, 1 dla kazdej kolumny i kazdej etykiety
4 pairs: list[list[list[int, int]]] = []

```

```
5 # lista min'ow i max'ow dla kazdej kolumny i kazdej etykiety
6 minmaxs: list[list[list[float, float]]] = []
```

Opis list pomocniczych *labels*, *pairs*, *minmax* przechowujących kolejno etykiety, pary 0, 1 w odpowiedniej kolejności dla każdej kolumny i wiersza, minima i maksima każdej kolumny i etykiety.

```
1 @staticmethod
2 def mean(column: list[float]) -> float:
3     """
4     Receives a column of the dataframe with numerical values and returns
5     its mean value
6     :param column: list[float]
7     :return mean value of the values in a dataframe column: float
8     """
9     # czyli zwykła suma przez ilość składników
10    return sum(column) / len(column)
```

Metoda pomocnicza obliczająca średnie wartości kolumn. Czyli suma wartości dzielona przez ilość wartości.

```
1 def calculate(self, df: pd.DataFrame, label: str) -> None:
2     """
3     Receives a dataframe and the label of the column we want to predict
4     and decides whether pairs are [0,1] or [1,0]
5     :param df: pd.DataFrame
6     :param label: str
7     :return: None
8     """
9     self.pairs = []
10    self.minmaxs = []
11    # lista unikalnych wartosci z podanej kolumny 'label'
12    # ma byc zawsze ta sama kolejnosc wiec sortujemy
13    self.labels = sorted(df[label].unique())
14
15    # 'category' to 0 i 1
16    for category in self.labels:
17        # wartosci ktorych unikalna wartosc kolumny 'label' to 'category'
18        # czyli podzial na czesci ktore maja 0 i 1
19        category_values = df[df[label] == category].drop(labels=label,
20        axis=1)
21        temp_minmax = []
22        temp_pair = []
23        # item to wszystkie wartosci z danej kolumny
24        for _, item in category_values.items():
25            # minimum z kolumny
26            min_t = min(item)
27            # maximum z kolumny
28            max_t = max(item)
29            temp_minmax.append([min_t, max_t])
30        # srednia
```

```

29         mean = self.mean(item)
30         len_lower = 0
31         len_upper = 0
32         # zliczamy elementy wieksze od sredniej i mniejsze od
           sredniej
33         for value in item:
34             if value > mean:
35                 len_upper += 1
36             else:
37                 len_lower += 1
38
39         # jesli jest tyle samo powyzej sredniej co ponizej to
40         # wybieramy losowo pare [0, 1] lub [1, 0]
41         if len_upper == len_lower:
42             temp_pair.append(random.choice(([0, 1],[1, 0])))
43         # jesli wiecej powyzej sredniej to zapisujemy pare [1, 0]
44         elif len_upper > len_lower:
45             temp_pair.append([1, 0])
46         # jesli wiecej ponizej sredniej to zapisujemy pare [0, 1]
47         else:
48             temp_pair.append([0, 1])
49         # uzupełniamy listy wszystkich 'minmaxow' i wszystkich 'par'
50         self.minmaxs.append(temp_minmax)
51         self.pairs.append(temp_pair)

```

Metoda tworząca tabelę ważoną zbioru miękkiego. Dzielimy zbiór na część, w której Wynik badania oznacza diabetyka oraz część, w której Wynik badania oznacza osobę zdrową. Dla każdej kolumny w każdej części obliczamy wartość średnią, wartość minimalną oraz wartość maksymalną. Zliczamy ilość elementów większych od średniej oraz elementów mniejszych od średniej. Tworzymy tabelę ważoną zbioru z par [0, 1] i [1, 0]. Pary [0, 1] dodajemy do tabeli ważonej w przypadku, gdy jest więcej elementów poniżej średniej. Pary [1, 0] dodajemy do tabeli ważonej w przypadku, gdy jest więcej elementów powyżej średniej. Jeżeli elementów jest tyle samo, to losujemy czy dodać parę [1, 0] czy [0, 1].

```

1 def predict(self, sample: pd.Series, label: str) -> str:
2     """
3     Receives a sample in the form of pd.Series and returns its predicted
        label
4     :param sample: pd.Series
5     :param label: str
6     :return predicted label of the sample: str
7     """
8     # liczenie elementow dla sampla dla kazdej z klas (0, 1)
9
10    sample = sample.drop(labels=label)
11    probabilities = []
12
13    # bierzemy po kolei odpowiednie pary i srednie
14    for category, minmax in zip(self.pairs, self.minmaxs):
15        sample_pairs = []
16        # bierzemy odpowiednia pare minmax i wartosc z sample'a
17        for pair, x in zip(minmax, sample):
18            # proportion = (wartosc_sampla - min) / (max - min)

```

```

19         proportion = (x - pair[0]) / (pair[1] - pair[0])
20         # dodajemy pare [1 - prop, prop] do listy wszystkich par
21         sample_pairs.append([1 - proportion, proportion])
22
23         temp = []
24         # liczenie prawdopodobienstwa dla kazdej klasy
25         for x, y in zip(category, sample_pairs):
26             temp.append(x[0] * y[0] + x[1] * y[1])
27         probabilities.append(sum(temp))
28     # naszym indeksem jest indeks największego prawdopodobienstwa
29     idx = max(range(len(probabilities)), key=probabilities.__getitem__)
30     # zwracamy najprawdopodobniejszy wynik z 'labels' (u nas 0 lub 1)
31     return self.labels[idx]

```

Metoda przewidująca etykietę klasy próbki. Znowu tworzymy listę par, jednak teraz bierzemy pod uwagę odpowiednie pary [min, max] oraz kolejne elementy z kolejnych kolumn w naszej próbce. Liczymy proporcję ze wzoru $(wartosc_sampla - min) / (max - min)$ i na jej podstawie dodajemy parę $[1 - proporcja, proporcja]$ do listy wszystkich par. Następnie na podstawie tych par liczymy prawdopodobieństwo wystąpienia każdej etykiety. Opiera się to na zwykłym iloczynie skalarnym wektorów. W tym przypadku naszymi 'wektorami' są indeksy etykiet oraz pary $[1 - proporcja, proporcja]$. Na koniec zwracamy przewidywaną etykietę (czyli tę, która uzyskała największą wartość na liście prawdopodobieństw).

```

1 def test(self, validation_set: pd.DataFrame, label: str) -> None:
2     """
3     Test how accurate the prediction method is
4     :param validation_set: pd.DataFrame - a record from the validation
5       dataset
6     :param label: str
7     :return: None
8     """
9     correct = 0
10    # iterrows() jest uzywane do iteracji wieszy dataframe'a
11    for _, row in validation_set.iterrows():
12        # jesli poprawnie przewidzimy etykiety
13        if self.predict(sample=row, label=label) == row[label]:
14            correct += 1
15    # informacja podsumowujaca
16    # print(f'Accuracy: {correct / len(validation_set) * 100:.2f}%; {
17      correct}/{len(validation_set)}')
18    return correct / len(validation_set) * 100

```

Metoda testująca dokładność działania klasyfikatora miękkiego. Prosty mechanizm, który w pętli wywołuje metodę *predict()* oraz porównuje wynik z wynikiem poprawnym. Zwrócona zostaje skuteczność przewidywania w procentach.

2.2.3 Klasyfikator rozmyty - Klasa Fuzzy

Klasa Fuzzy zawiera klasyfikator rozmyty oraz metody pomocnicze.

```

1 def __init__(self):

```

```

2     # słownik słowników rozmycia dla każdej kolumny
3     self.ancecedents: dict[str: dict[str: tuple[float]]] = {}
4     # słownik zawierający nazwę etykiety klasy oraz słowników z jej
        rozmyciem
5     self.consequent: dict[str: dict[str: tuple[float]]] = {}
6     # lista słowników; każda reguła to jeden słownik
7     self.rules: list[dict[str: str]] = []

```

Opis atrybutów pomocniczych **ancecedents**, **consequent**, **rules** przechowujących słownik słowników rozmycia dla każdej kolumny, słownik zawierający nazwę etykiety klasy oraz słowników z jej rozmyciem, lista słowników, w których każda reguła to jeden słownik.

```

1 def add_ancecedent(self, parameter: str, linguistic_value: str, *args)
    -> None:
2     if parameter not in self.ancecedents:
3         self.ancecedents[parameter] = {}
4     self.ancecedents[parameter][linguistic_value] = args

```

Metoda pozwalająca na dodanie poprzednika do słownika **ancecedens**, gdzie każdy poprzednik to słownik.

```

1 def add_consequent(self, parameter: str, linguistic_value: str, *args)
    -> None:
2     if parameter not in self.consequent:
3         self.consequent[parameter] = {}
4     self.consequent[parameter][linguistic_value] = args

```

Metoda pozwalająca na dodanie konsekwencji do słownika **consequent**, gdzie każda konsekwencja to słownik.

```

1 def add_rule(self, rule: dict[str: str]) -> None:
2     assert set(rule.keys()) == set.union(set(self.ancecedents.keys()),
        set(self.consequent.keys())), print("Nieprawidłowa liczba
        kategorii w regule.")
3     self.rules.append(rule)

```

Metoda służąca do dodawania zasad do słownika **rules**, gdzie każda zasada to słownik.

```

1 @staticmethod
2 def triangular_function(n: float, a: float, b: float, c: float) -> float
    :
3     assert a <= b <= c, print("Nieprawidłowe wartosci")
4     if n == a == b or n == b == c:
5         return 1
6     if n <= a:
7         return 0
8     if a < n < b:
9         return (n - a) / (b - a)
10    if n == b:
11        return 1

```

```

12     if b < n < c:
13         return (c - n) / (c - b)
14     if n >= c:
15         return 0

```

Metoda obliczająca przynależność dla funkcji trójkątnej.

```

1 @staticmethod
2 def trapezoidal_function(n: float, a: float, b: float, c: float, d:
   float) -> float:
3     assert a <= b <= c <= d, print("Nieprawidłowe wartosci")
4     if c == d and n >= d:
5         return 1
6     if a == b and n <= a:
7         return 1
8     if n <= a:
9         return 0
10    if a < n < b:
11        return (n - a) / (b - a)
12    if b <= n <= c:
13        return 1
14    if c < n < d:
15        return (d - n) / (d - c)
16    if n >= d:
17        return 0

```

Metoda obliczająca przynależność dla funkcji trapezoidalnej.

```

1 def membership_function(self, n: float, *args) -> float:
2     assert len(args) in (3, 4), print("Nieprawidłowa liczba argumentow")
3     if len(args) == 3:
4         return self.triangular_function(n, *args)
5     else:
6         return self.trapezoidal_function(n, *args)

```

Metoda wywołująca funkcję przynależności. W zależności od liczby podanych argumentów odwoła się ona do funkcji trójkątnej lub trapezoidalnej.

```

1 def fuzzify(self, column: str, n: float) -> dict[str, float]:
2     return {antecedent: self.membership_function(n, *self.antecedents[
        column][antecedent]) for antecedent in self.antecedents[column].
        keys()}

```

Metoda służąca do rozmycia danych wejściowych. Dla każdej kolumny zwraca słownik, gdzie jest informacja, w jakim stopniu i jakie wartości są dla danej próbki. Z konkretnej wartości liczbowej przechodzimy w rozmyte.

```

1 @staticmethod
2 def rule_fulfillment(rule: dict[str: str], fuzzy_values: dict[str: dict[
   str: float]]) -> tuple[str, float]:

```

```

3     label = list(rule.keys())[-1]
4     minimum = float('inf')
5     # dla kazdej wartosci w danej zasadzie
6     for name, linguistic_value in list(rule.items())[:-1]:
7         maximum = float('-inf')
8         # jesli mamy wartosci rozdzielone 'or', to bierzemy ich maksimum
9         for lvs in linguistic_value.split(' | '):
10            maximum = max(fuzzy_values[name][lvs], maximum)
11            # bierzemy minimum stopni spelnienia wartosci danej zasady
12            minimum = min(minimum, maximum)
13     return rule[label], minimum

```

Metoda obliczająca stopień spełnienia danej zasady.

```

1 @staticmethod
2 def area(a: float, c: float) -> float:
3     return (c - a) / 2

```

Metoda obliczająca powierzchnię danego wyniku.

```

1 @staticmethod
2 def cog(a: float, b: float, c: float) -> float:
3     return (a + b + c) / 3

```

Metoda obliczająca środek ciężkości.

```

1 def aggregate(self, label: str, outputs: defaultdict) -> float:
2     # licznik i mianownik dla funkcji agregujacej
3     counter = []
4     denominator = []
5     # obliczanie powierzchni oraz srodka ciezkosci kazdej etykiety
6     # funkcji przynaloznosci
7     for name, membership in outputs.items():
8         if not membership:
9             continue
10        a, b, c = self.consequent[label][name]
11        area = self.area(a, c)
12        cog = self.cog(a, b, c)
13        counter.append(membership * area * cog)
14        denominator.append(membership * area)
15    return round(sum(counter) / sum(denominator), 3)

```

Metoda agregująca, wystrzajająca wartość końcową za pomocą metody środka ciężkości.

```

1 def compute(self, sample: pd.Series) -> tuple[str, float]:
2     # rozmycie probki
3     fuzzy_values = {column: self.fuzzify(column, sample[column]) for
4                     column in self.antecedents.keys()}
5
6     # obliczenie stopnia spelnienia zasad
7     fulfillments = [self.rule_fulfillment(rule, fuzzy_values) for rule
8                    in self.rules]

```

```

7
8     # wybranie maksymalnego spełnienia reguły dla danych wartosci
9     outputs: defaultdict = defaultdict(lambda: 0)
10    for linguistic_value, fulfillment in fulfillments:
11        outputs[linguistic_value] = max(outputs[linguistic_value],
12                                         fulfillment)
13    outputs = dict(outputs)
14    label = list(self.consequent.keys())[0]
15
16    # wybor przewidzianej etykiety klasy: w przypadku remisu dla dwóch
17    # etykiet wybieramy chorobe
18    output_names = sorted(outputs.keys(), key=lambda x: (-1 * outputs[x
19    ], x))
20
21    # print(sample['Outcome'], self.aggregate(label, outputs), outputs)
22    return output_names[0], self.aggregate(label, outputs)

```

Metoda przewidująca etykietę klasy.

```

1 def view(self, parameter: str) -> None:
2     # lista kolorow do wykresow
3     colors: dict[int: tuple] = {2: ('#00b100', '#ff5c38'),
4                                     3: ('#00b100', '#d5c731', '#ff5c38'),
5                                     4: ('#00b100', '#adcd27', '#f1b438', '#
6                                     ff5c38'),
7                                     5: ('#00b100', '#93cb20', '#d5c731', '#
8                                     f9a539', '#ff5c38')}
9
10    # globalne parametry dla wykresow matplotlib.pyplot
11    plt.rcParams["figure.figsize"] = (14, 8)
12    if parameter in self.consequent.keys():
13        source = self.consequent[parameter]
14    else:
15        source = self.antecedents[parameter]
16
17    # stworzenie dziedziny
18    minimum: int = floor(list(source.values())[0][0])
19    maximum: int = ceil(list(source.values())[-1][-1])
20    x = np.linspace(minimum, maximum, num=(maximum - minimum) * 100 + 1)
21    # stworzenie wykresow funkcji przynaleznosci
22    for k, v, color in zip(source.keys(), source.values(), colors[len(
23    source.keys()))):
24        y = np.array([self.membership_function(i, *v) for i in x])
25        plt.plot(x, y, color, label=k)
26
27    # nadanie parametrow wykresu
28    plt.rc('font', size=16)
29    plt.rc('axes', titlesize=14)
30    plt.rc('axes', labelszize=14)
31    plt.rc('xtick', labelszize=14)
32    plt.rc('ytick', labelszize=14)
33    plt.rc('legend', fontsize=14)
34    plt.rc('figure', titlesize=16)
35    plt.title(parameter)
36    plt.xlabel('Membership')
37    plt.ylabel(parameter)

```

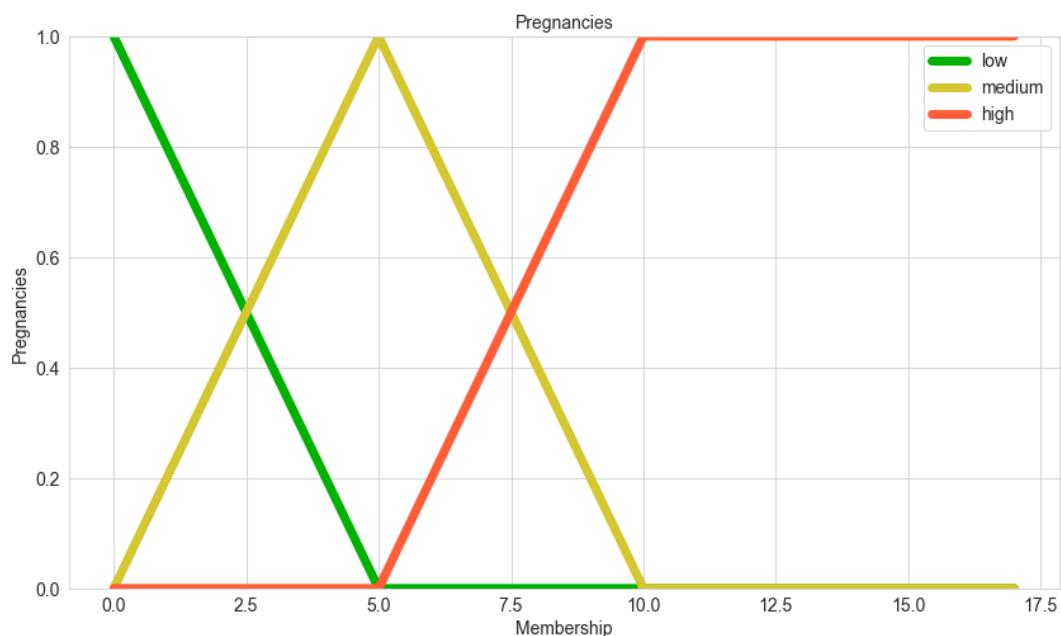


```

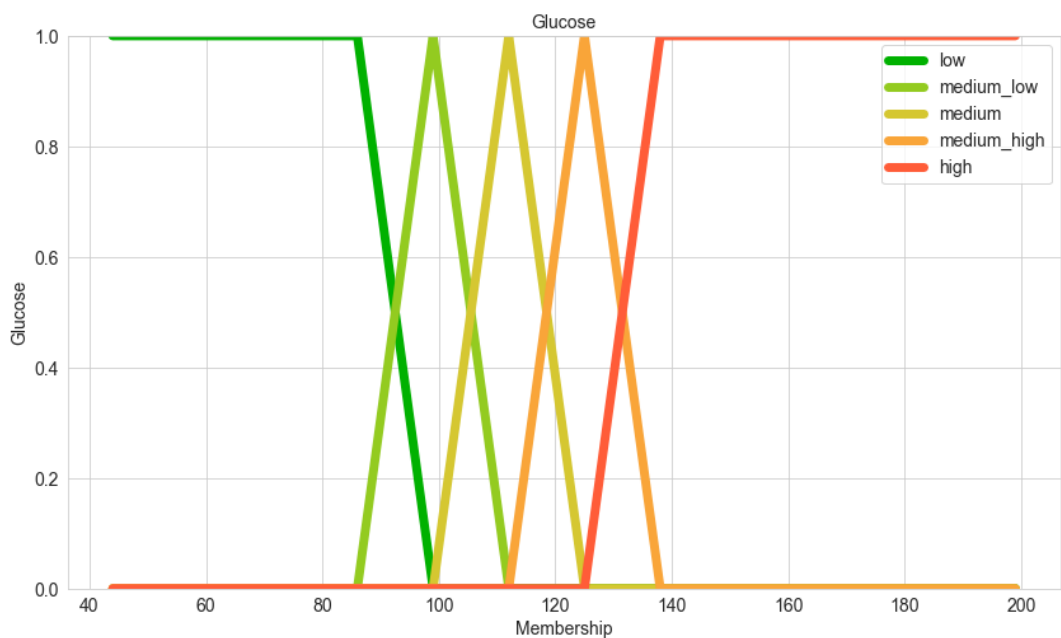
34 plt.legend(loc="upper right")
35 plt.ylim(0, 1)
36 plt.show()

```

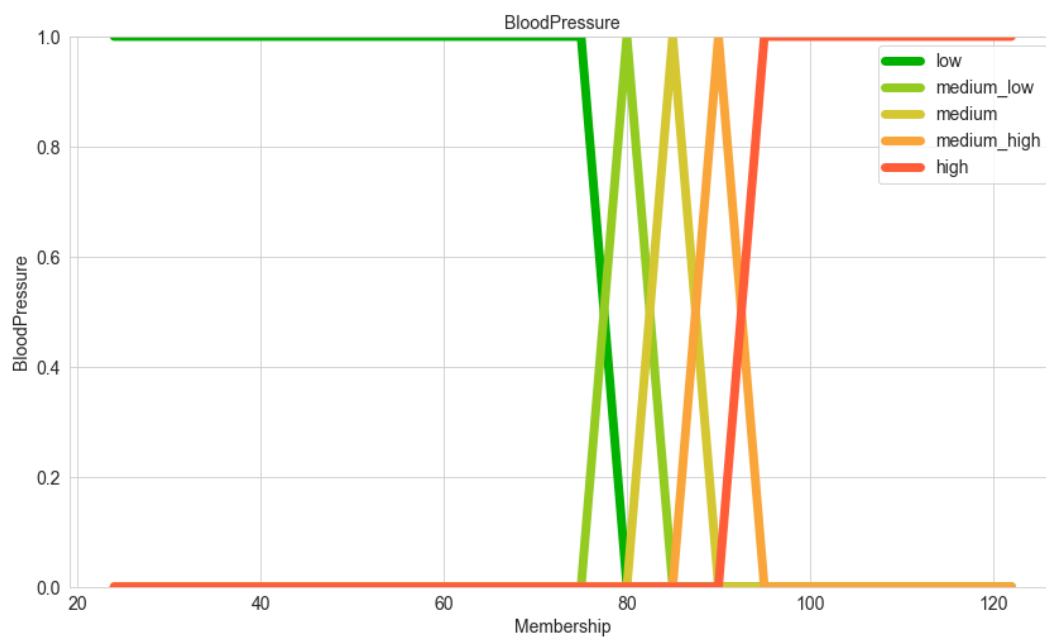
Metoda pomocnicza wyświetlająca wykres rozmycia cechy.
Poniżej wykresy wygenerowane przy pomocy metody *view()*.



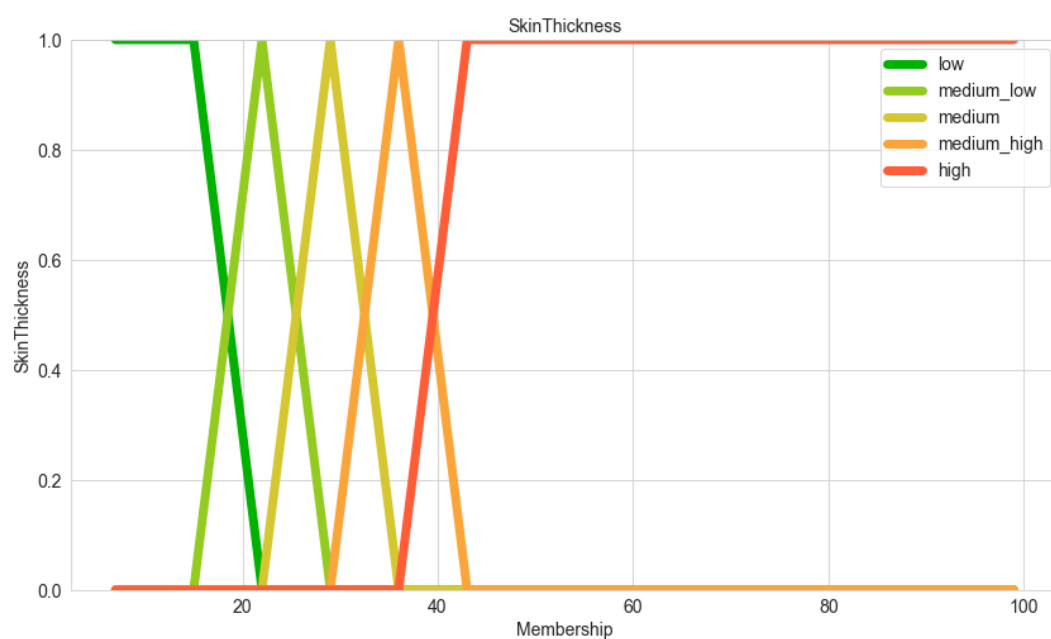
Rysunek 13: Wykres funkcji przynależności dla kolumny Pregnancies



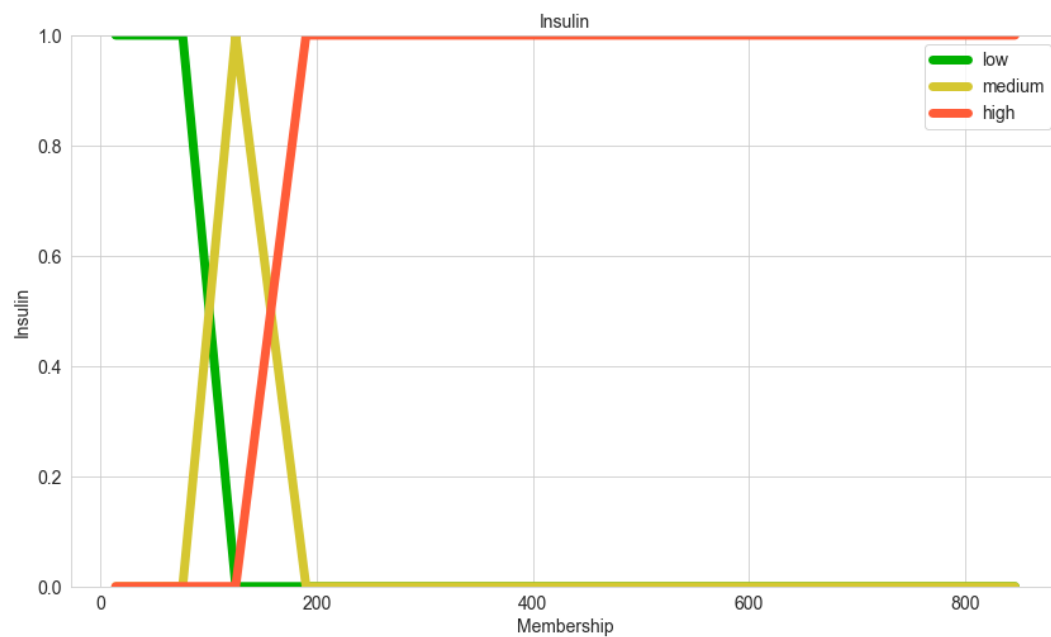
Rysunek 14: Wykres funkcji przynależności dla kolumny Glucose



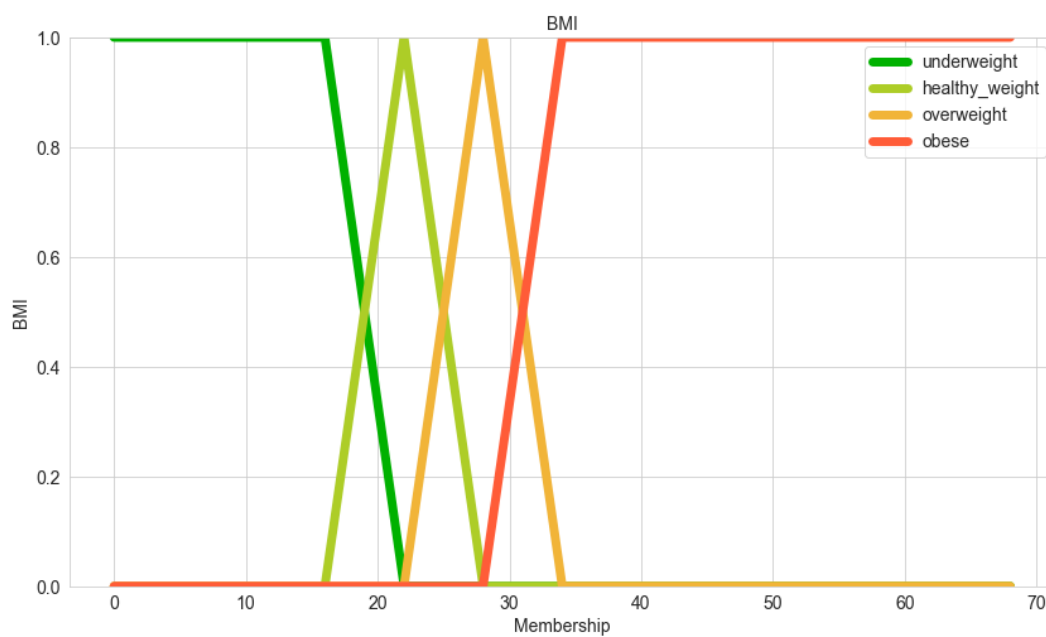
Rysunek 15: Wykres funkcji przynależności dla kolumny BloodPressure



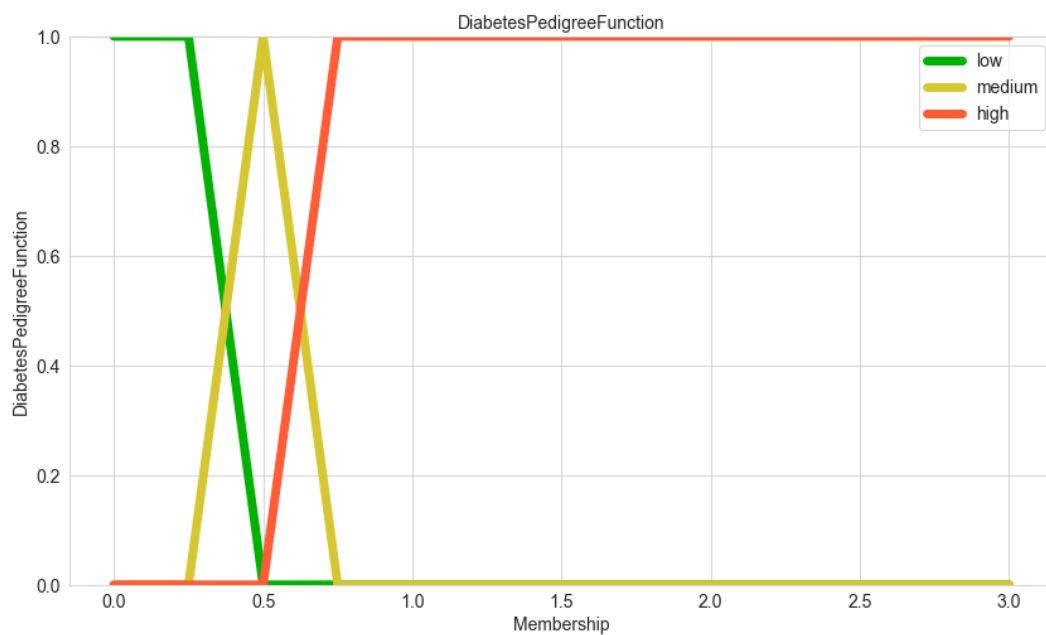
Rysunek 16: Wykres funkcji przynależności dla kolumny SkinThickness



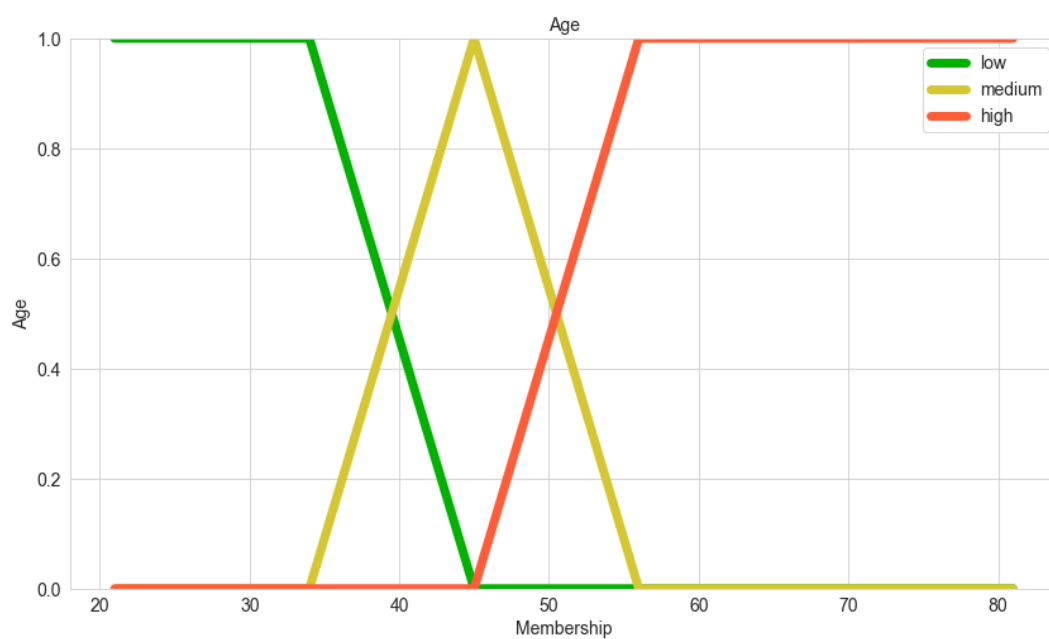
Rysunek 17: Wykres funkcji przynależności dla kolumny Insulin



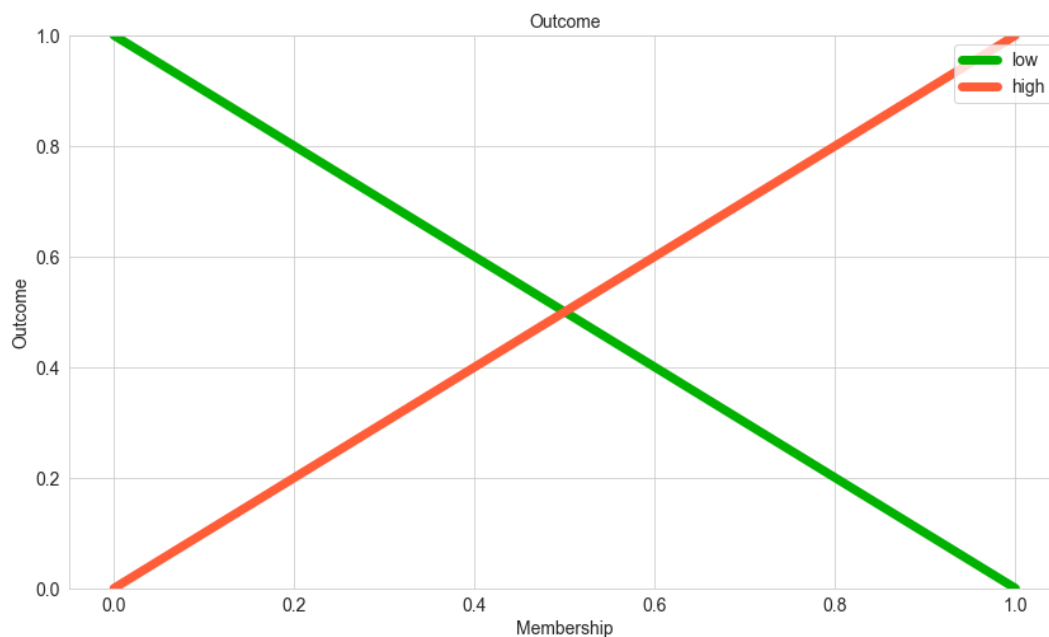
Rysunek 18: Wykres funkcji przynależności dla kolumny BMI



Rysunek 19: Wykres funkcji przynależności dla kolumny DiabetesPedigreeFunction



Rysunek 20: Wykres funkcji przynależności dla kolumny Age



Rysunek 21: Wykres funkcji przynależności dla kolumny Outcome

2.3 Testy

2.3.1 Test metod klasy ProcessingData

```
shuffled_data = ProcessingData.shuffle(data)
shuffled_data
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	4	84	90	23	56	39.5	0.159	25	0
1	8	126	88	36	108	38.5	0.349	49	0
2	3	115	66	39	140	38.1	0.150	28	0
3	0	177	60	29	478	34.6	1.072	21	1
4	1	79	80	25	37	25.4	0.583	22	0
...
387	0	124	56	13	105	21.8	0.452	21	0
388	2	144	58	33	135	31.6	0.422	25	1
389	9	154	78	30	100	30.9	0.164	45	0
390	1	84	64	23	115	36.9	0.471	28	0
391	6	165	68	26	168	33.6	0.631	49	0

Rysunek 22: Wyświetlenie przetasowanych danych przy pomocy metody *shuffle()*

```
normalized_data_std = ProcessingData.normalize_std(shuffled_data, 'Outcome')
normalized_data_std
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	0.217654	-1.251671	1.547423	-0.584363	-0.841928	0.912647	-1.053715	-0.574936	0
1	1.463207	0.109279	1.387373	0.651799	-0.404371	0.770353	-0.503768	1.777826	0
2	-0.093734	-0.247160	-0.373178	0.937067	-0.135105	0.713435	-1.079765	-0.280841	0
3	-1.027899	1.761862	-0.853328	-0.013827	2.709015	0.215403	1.588924	-0.967063	1
4	-0.716511	-1.413689	0.747172	-0.394184	-1.001804	-1.093710	0.173534	-0.869031	0
...
387	-1.027899	0.044472	-1.173428	-1.535257	-0.429615	-1.605972	-0.205639	-0.967063	0
388	-0.405123	0.692544	-1.013378	0.366531	-0.177178	-0.211482	-0.292473	-0.574936	1
389	1.774596	1.016580	0.587122	0.081263	-0.471687	-0.311089	-1.039243	1.385699	0
390	-0.716511	-1.251671	-0.533228	-0.584363	-0.345469	0.542681	-0.150645	-0.280841	0
391	0.840431	1.373019	-0.213128	-0.299095	0.100502	0.073108	0.312468	1.777826	0

Rysunek 23: Wyświetlenie znormalizowanych danych przy pomocy metody *normalize_std()*

```
normalized_data_minmax = ProcessingData.normalize_minmax(shuffled_data, 'Outcome')
normalized_data_minmax
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	0.235294	0.197183	0.767442	0.285714	0.050481	0.435583	0.031692	0.066667	0
1	0.470588	0.492958	0.744186	0.517857	0.112981	0.415133	0.113062	0.466667	0
2	0.176471	0.415493	0.488372	0.571429	0.151442	0.406953	0.027837	0.116667	0
3	0.000000	0.852113	0.418605	0.392857	0.557692	0.335378	0.422698	0.000000	1
4	0.058824	0.161972	0.651163	0.321429	0.027644	0.147239	0.213276	0.016667	0
...
387	0.000000	0.478873	0.372093	0.107143	0.109375	0.073620	0.157173	0.000000	0
388	0.117647	0.619718	0.395349	0.464286	0.145433	0.274029	0.144325	0.066667	1
389	0.529412	0.690141	0.627907	0.410714	0.103365	0.259714	0.033833	0.400000	0
390	0.058824	0.197183	0.465116	0.285714	0.121394	0.382413	0.165310	0.116667	0
391	0.352941	0.767606	0.511628	0.339286	0.185096	0.314928	0.233833	0.466667	0

Rysunek 24: Wyświetlenie znormalizowanych danych przy pomocy metody *normalize_minmax()*

```
# Test dla normalizacji std
normalized_training_data_std, normalized_validation_data_std = ProcessingData.split(normalized_data_std, 0.7)
normalized_training_data_std
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	0.217654	-1.251671	1.547423	-0.584363	-0.841928	0.912647	-1.053715	-0.574936	0
1	1.463207	0.109279	1.387373	0.651799	-0.404371	0.770353	-0.503768	1.777826	0
2	-0.093734	-0.247160	-0.373178	0.937067	-0.135105	0.713435	-1.079765	-0.280841	0
3	-1.027899	1.761862	-0.853328	-0.013827	2.709015	0.215403	1.588924	-0.967063	1
4	-0.716511	-1.413689	0.747172	-0.394184	-1.001804	-1.093710	0.173534	-0.869031	0
...
269	-0.716511	-0.700810	-1.653578	-1.345078	-1.010219	-1.264464	0.008550	-0.476904	0
270	-0.405123	1.113791	0.267022	0.556709	2.389262	0.898418	-1.126076	-0.084777	0
271	1.463207	1.729459	1.547423	0.461620	1.211224	0.087337	-0.162222	2.660112	1
272	0.217654	-0.376774	0.106972	1.697782	0.428670	0.571140	2.509361	2.464048	1
273	-0.405123	-0.020335	-0.853328	-1.059810	-0.421200	-0.467613	0.561392	-0.869031	0

274 rows × 9 columns [Open in new tab](#)

```
normalized_validation_data_std
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	-1.027899	-0.668407	1.227323	-1.154899	-0.429615	-0.538760	0.497714	-0.378873	0
1	-0.716511	0.174087	-1.813628	1.507603	0.319281	1.054942	0.260368	-0.672968	1
2	-0.405123	0.141683	-1.013378	-0.489274	1.000860	-0.766432	3.117196	-0.574936	0
3	-0.093734	-0.506389	-0.693278	-1.535257	-0.909244	-1.449448	0.448508	-0.771000	1
4	-0.716511	0.238894	-0.053078	-1.535257	-0.429615	-1.022563	-0.147750	-0.869031	0
...
113	-1.027899	0.044472	-1.173428	-1.535257	-0.429615	-1.605972	-0.205639	-0.967063	0
114	-0.405123	0.692544	-1.013378	0.366531	-0.177178	-0.211482	-0.292473	-0.574936	1
115	1.774596	1.016580	0.587122	0.081263	-0.471687	-0.311089	-1.039243	1.385699	0
116	-0.716511	-1.251671	-0.533228	-0.584363	-0.345469	0.542681	-0.150645	-0.280841	0
117	0.840431	1.373019	-0.213128	-0.299095	0.100502	0.073108	0.312468	1.777826	0

Rysunek 25: Wyświetlenie zbioru treningowego i walidacyjnego podzielonych przy pomocy metody *split()* dla danych znormalizowanych metodą *normalize_std()*

```
# Test dla normalizacji minmax
normalized_training_data_minmax, normalized_validation_data_minmax = ProcessingData.split(normalized_data_minmax, 0.7)
normalized_training_data_minmax
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	0.235294	0.197183	0.767442	0.285714	0.050481	0.435583	0.031692	0.066667	0
1	0.470588	0.492958	0.744186	0.517857	0.112981	0.415133	0.113062	0.466667	0
2	0.176471	0.415493	0.488372	0.571429	0.151442	0.406953	0.027837	0.116667	0
3	0.000000	0.852113	0.418605	0.392857	0.557692	0.335378	0.422698	0.000000	1
4	0.058824	0.161972	0.651163	0.321429	0.027644	0.147239	0.213276	0.016667	0
...
269	0.058824	0.316901	0.302326	0.142857	0.026442	0.122699	0.188865	0.083333	0
270	0.117647	0.711268	0.581395	0.500000	0.512019	0.433538	0.020985	0.150000	0
271	0.470588	0.845070	0.767442	0.482143	0.343750	0.316973	0.163597	0.616667	1
272	0.235294	0.387324	0.558140	0.714286	0.231971	0.386503	0.558887	0.583333	1
273	0.117647	0.464789	0.418605	0.196429	0.110577	0.237219	0.270664	0.016667	0

274 rows × 9 columns [Open in new tab](#)

```
normalized_validation_data_minmax
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	0.000000	0.323944	0.720930	0.178571	0.109375	0.226994	0.261242	0.100000	0
1	0.058824	0.507042	0.279070	0.678571	0.216346	0.456033	0.226124	0.050000	1
2	0.117647	0.500000	0.395349	0.303571	0.313702	0.194274	0.648822	0.066667	0
3	0.176471	0.359155	0.441860	0.107143	0.040865	0.096115	0.253961	0.033333	1
4	0.058824	0.521127	0.534884	0.107143	0.109375	0.157464	0.165739	0.016667	0
...
113	0.000000	0.478873	0.372093	0.107143	0.109375	0.073620	0.157173	0.000000	0
114	0.117647	0.619718	0.395349	0.464286	0.145433	0.274029	0.144325	0.066667	1
115	0.529412	0.690141	0.627907	0.410714	0.103365	0.259714	0.033833	0.400000	0
116	0.058824	0.197183	0.465116	0.285714	0.121394	0.382413	0.165310	0.116667	0
117	0.352941	0.767606	0.511628	0.339286	0.185096	0.314928	0.233833	0.466667	0

Rysunek 26: Wyświetlenie zbioru treningowego i walidacyjnego podzielonych przy pomocy metody *split()* dla danych znormalizowanych metodą *normalize_minmax()*

2.3.2 Test metod klasy NaiveBayes

```
Accuracy: 76.27%; 90/118
Accuracy: 73.73%; 87/118
Accuracy: 76.27%; 90/118
Accuracy: 77.12%; 91/118
Accuracy: 75.42%; 89/118
```

Rysunek 27: Test klasyfikatora *NaiveBayes* dla danych znormalizowanych przy pomocy metody *normalize_std()*

Testy dokładności przewidywań zostały przeprowadzone dla 50 próbek. Średnia dokładność wyniosła 74.98%

```
Accuracy: 74.58%; 88/118
Accuracy: 73.73%; 87/118
Accuracy: 72.88%; 86/118
Accuracy: 77.97%; 92/118
Accuracy: 78.81%; 93/118
```

Rysunek 28: Test klasyfikatora *NaiveBayes* dla danych znormalizowanych przy pomocy metody *normalize_minmax()*

Testy dokładności przewidywań zostały przeprowadzone dla 50 próbek. Średnia dokładność wyniosła 75.97%

```
Accuracy: 74.58%; 88/118
Accuracy: 76.27%; 90/118
Accuracy: 82.20%; 97/118
Accuracy: 77.97%; 92/118
Accuracy: 73.73%; 87/118
```

Rysunek 29: Test klasyfikatora *NaiveBayes* dla danych nieznormalizowanych

Testy dokładności przewidywań zostały przeprowadzone dla 50 próbek. Średnia dokładność wyniosła 75.56%

2.3.3 Test metod klasy KNN

```
Accuracy for k=2: 67.80%, correct predictions: 80/118
Accuracy for k=3: 66.95%, correct predictions: 79/118
Accuracy for k=4: 59.32%, correct predictions: 70/118
Accuracy for k=2: 80.51%, correct predictions: 95/118
Accuracy for k=3: 77.12%, correct predictions: 91/118
Accuracy for k=4: 72.88%, correct predictions: 86/118
```

Rysunek 30: Test klasyfikatora *KNN* dla danych znormalizowanych przy pomocy metody *normalize_std()*

Testy dokładności przewidywań zostały przeprowadzone dla 50 próbek. Średnia dokładność wyniosła 71.31%

```
Accuracy for k=2: 70.34%, correct predictions: 83/118
Accuracy for k=3: 69.49%, correct predictions: 82/118
Accuracy for k=4: 67.80%, correct predictions: 80/118
Accuracy for k=2: 68.64%, correct predictions: 81/118
Accuracy for k=3: 65.25%, correct predictions: 77/118
Accuracy for k=4: 63.56%, correct predictions: 75/118
```

Rysunek 31: Test klasyfikatora *KNN* dla danych znormalizowanych przy pomocy metody *normalize_minmax()*

Testy dokładności przewidywań zostały przeprowadzone dla 50 próbek. Średnia dokładność wyniosła 71.51%

```
Accuracy for k=2: 63.56%, correct predictions: 75/118
Accuracy for k=3: 59.32%, correct predictions: 70/118
Accuracy for k=4: 54.24%, correct predictions: 64/118
Accuracy for k=2: 67.80%, correct predictions: 80/118
Accuracy for k=3: 66.10%, correct predictions: 78/118
Accuracy for k=4: 58.47%, correct predictions: 69/118
```

Rysunek 32: Test klasyfikatora *KNN* dla danych nieznormalizowanych

Testy dokładności przewidywań zostały przeprowadzone dla 50 próbek. Średnia dokładność wyniosła 65.59%

2.3.4 Test metod klasy SoftSetClassifierMean

```
Accuracy: 43.22%; 51/118
Accuracy: 45.76%; 54/118
Accuracy: 54.24%; 64/118
Accuracy: 67.80%; 80/118
Accuracy: 61.86%; 73/118
```

Rysunek 33: Test *klasyfikatora miękkiego* bazującego na średniej dla danych znormalizowanych przy pomocy metody *normalize_std()*

Testy dokładności przewidywań zostały przeprowadzone dla 50 próbek. Średnia dokładność wyniosła 48.44%

```
Accuracy: 34.75%; 41/118
Accuracy: 34.75%; 41/118
Accuracy: 41.53%; 49/118
Accuracy: 39.83%; 47/118
Accuracy: 53.39%; 63/118
```

Rysunek 34: Test *klasyfikatora miękkiego* bazującego na średniej dla danych znormalizowanych przy pomocy metody *normalize_minmax()*

Testy dokładności przewidywań zostały przeprowadzone dla 50 próbek. Średnia dokładność wyniosła 50.08%

```
Accuracy: 36.44%; 43/118
Accuracy: 50.85%; 60/118
Accuracy: 52.54%; 62/118
Accuracy: 53.39%; 63/118
Accuracy: 63.56%; 75/118
```

Rysunek 35: Test *klasyfikatora miękkiego* bazującego na średniej dla danych nieznormalizowanych

Testy dokładności przewidywań zostały przeprowadzone dla 50 próbek. Średnia dokładność wyniosła 50.12%

2.3.5 Test metod klasy SoftSetClassifierPercentage

Accuracy: 27.97%; 33/118
Accuracy: 41.53%; 49/118
Accuracy: 70.34%; 83/118
Accuracy: 33.05%; 39/118
Accuracy: 30.51%; 36/118

Rysunek 36: Test *klasyfikatora miękkiego* bazującego na procentach dla danych znormalizowanych przy pomocy metody *normalize_std()*

Testy dokładności przewidywań zostały przeprowadzone dla 50 próbek. Średnia dokładność wyniosła 55.12%

Accuracy: 71.19%; 84/118
Accuracy: 44.92%; 53/118
Accuracy: 29.66%; 35/118
Accuracy: 64.41%; 76/118
Accuracy: 46.61%; 55/118

Rysunek 37: Test *klasyfikatora miękkiego* bazującego na procentach dla danych znormalizowanych przy pomocy metody *normalize_minmax()*

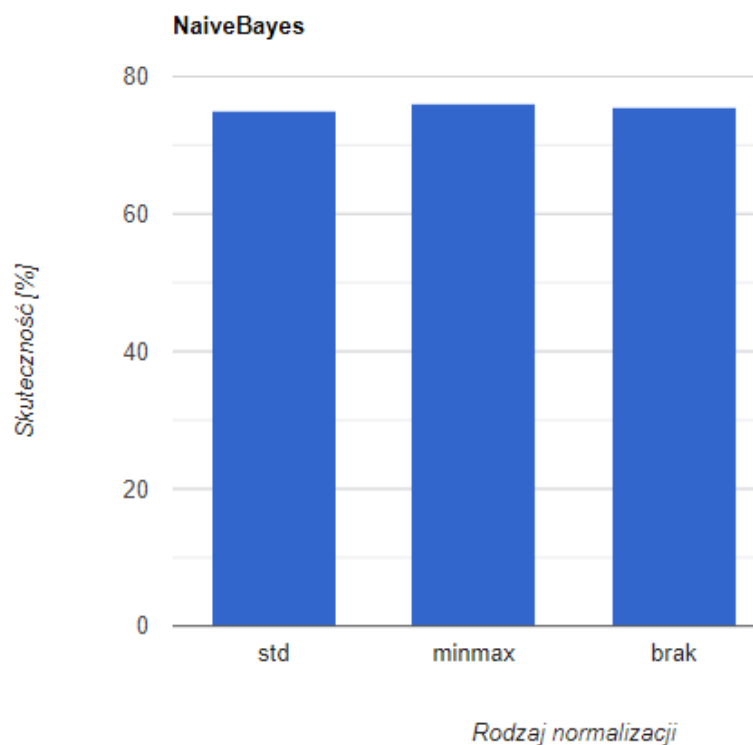
Testy dokładności przewidywań zostały przeprowadzone dla 50 próbek. Średnia dokładność wyniosła 53.32%

Accuracy: 68.64%; 81/118
Accuracy: 58.47%; 69/118
Accuracy: 61.02%; 72/118
Accuracy: 33.90%; 40/118
Accuracy: 68.64%; 81/118

Rysunek 38: Test *klasyfikatora miękkiego* bazującego na procentach dla danych nieznormalizowanych

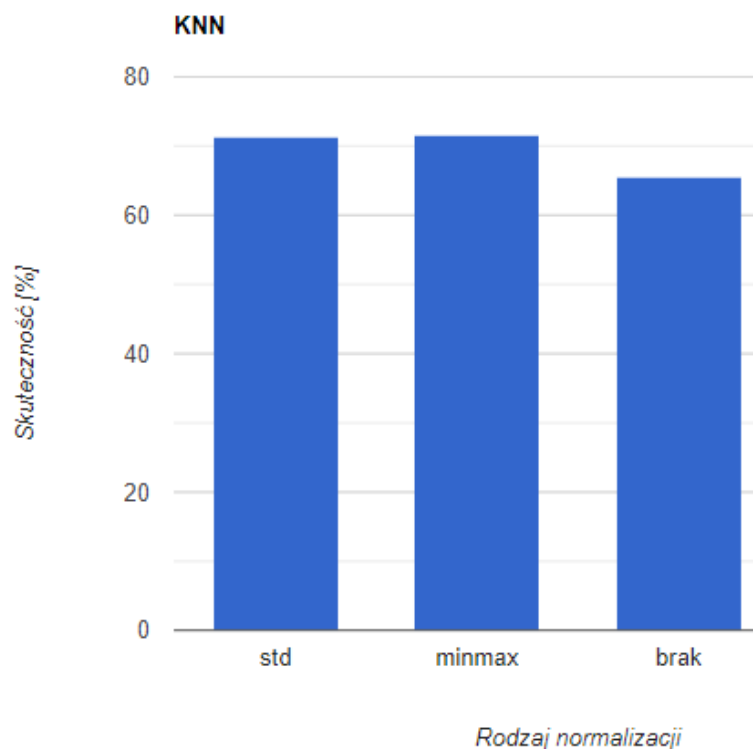
Testy dokładności przewidywań zostały przeprowadzone dla 50 próbek. Średnia dokładność wyniosła 52.93%

2.4 Eksperymenty



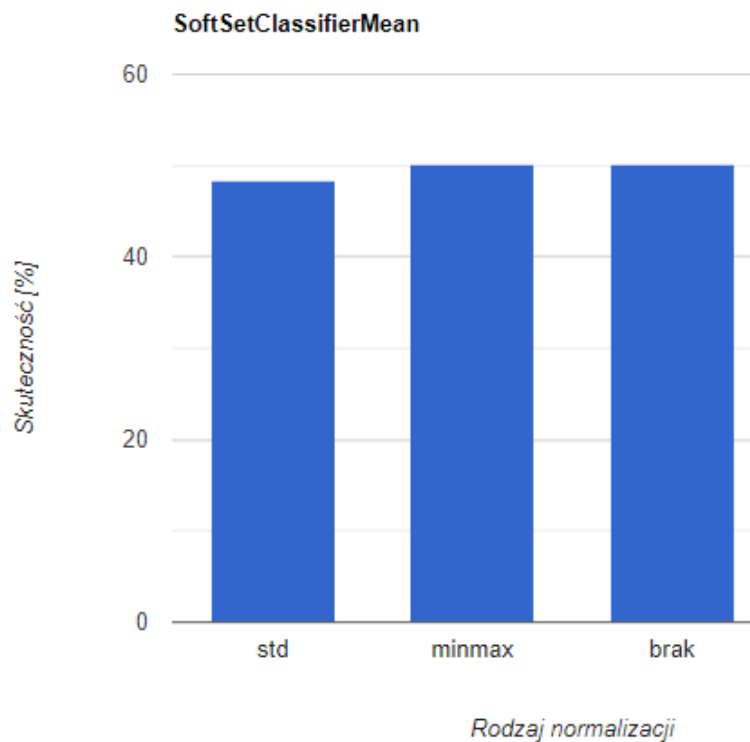
Rysunek 39: Podsumowanie wyników dokładności klasy NaiveBayes

Najlepszą dokładność przewidywanych wyników za pomocą klasy NaiveBayes uzyskaliśmy dla danych znormalizowanych przy użyciu wzoru opierającego się na minimum i maksimum - 75.97%. Niewiele mniej, bo 75.56% dla danych nieznormalizowanych. Najgorszy wynik uzyskaliśmy dla danych znormalizowanych przy użyciu wzoru opierającego się na odchyleniu standardowym - 74.98%. Jednocześnie należy pamiętać, że są to najlepsze wyniki spośród wszystkich zaprezentowanych klasyfikatorów.



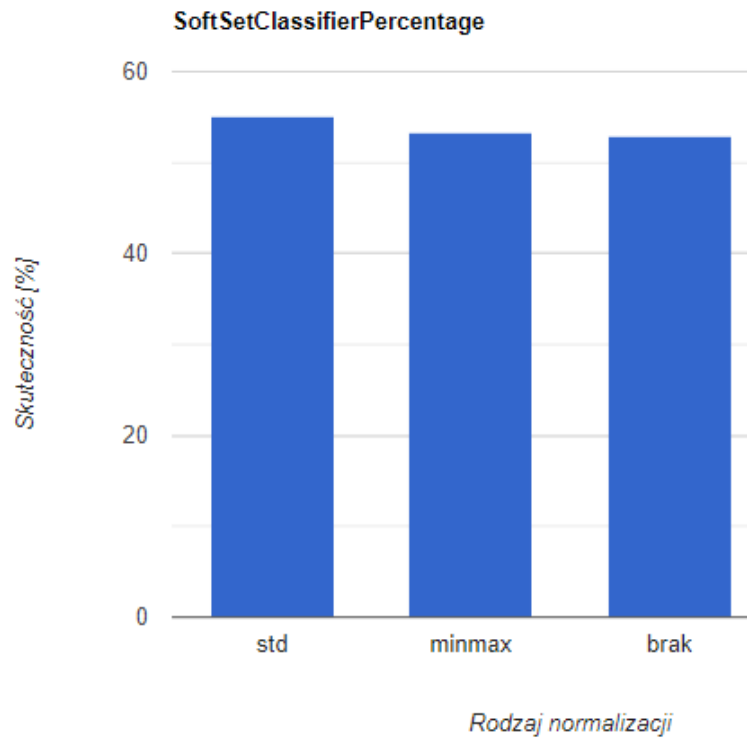
Rysunek 40: Podsumowanie wyników dokładności klasy KNN

Najlepszą dokładność przewidywanych wyników za pomocą klasy KNN uzyskaliśmy dla danych znormalizowanych przy użyciu wzoru opierającego się na minimum i maksimum - 71.51%. Wynik niemal identyczny - 71.31% - dla danych znormalizowanych przy użyciu wzoru opierającego się na odchyleniu standardowym. Najgorszy wynik uzyskaliśmy dla danych nieznormalizowanych - 65.59%. Uzyskane wyniki plasują klasyfikator KNN na drugim miejscu zaraz za klasyfikatorem NaiveBayes w naszym osobistym rankingu testowanych klasyfikatorów.



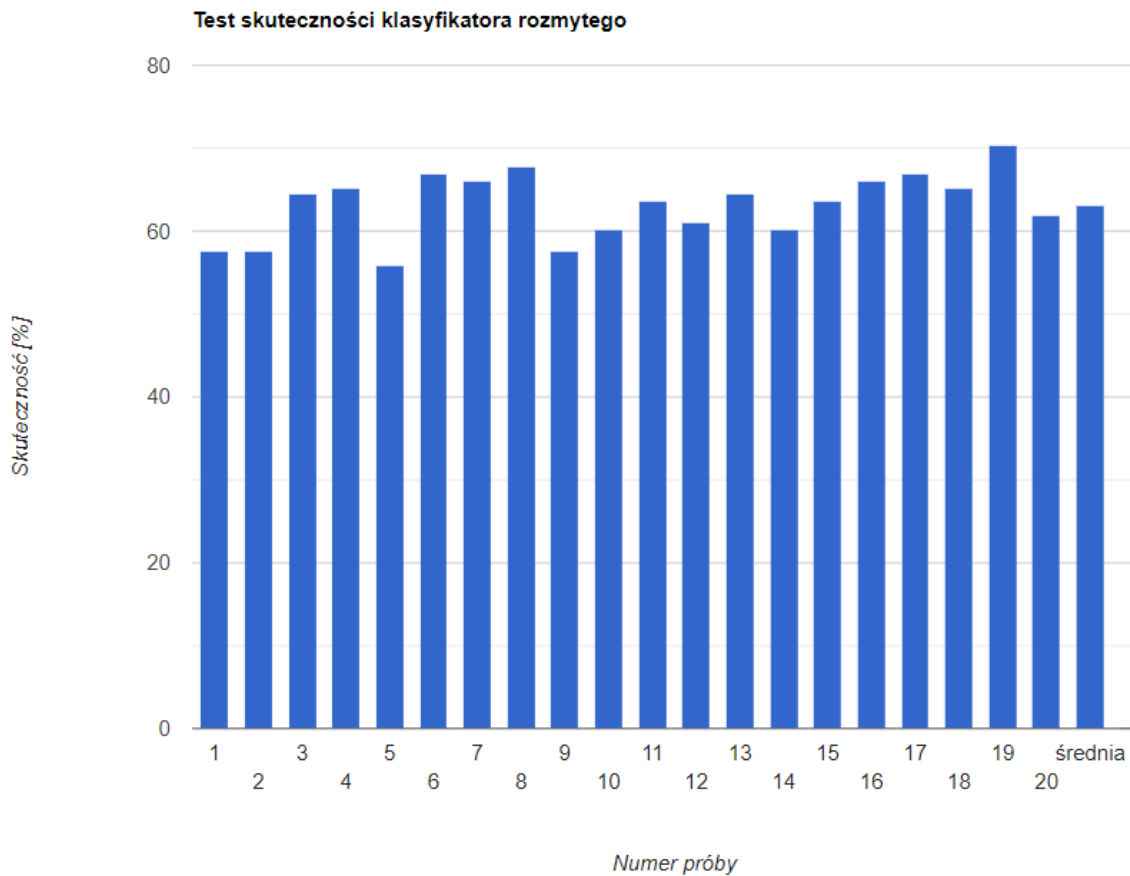
Rysunek 41: Podsumowanie wyników dokładności klasy SoftSetClassifierMean

Najlepszą dokładność przewidywanych wyników za pomocą klasy SoftSetClassifierMean uzyskaliśmy dla danych nieznormalizowanych - 50.12%. Wynik niemal identyczny - 50.08% - dla danych znormalizowanych przy użyciu wzoru opierającego się na minimum i maksimum. Najgorszy wynik uzyskaliśmy dla danych znormalizowanych przy użyciu wzoru opierającego się na odchyleniu standardowym - 48.44%. Wyniki są niezadowalające ze względu na użycie średniej w klasyfikatorze. Można powiedzieć, że przypominają losowe przewidywania ze względu na skuteczność bliską 50%. Jest to najmniej efektywny klasyfikator spośród zaprezentowanych.



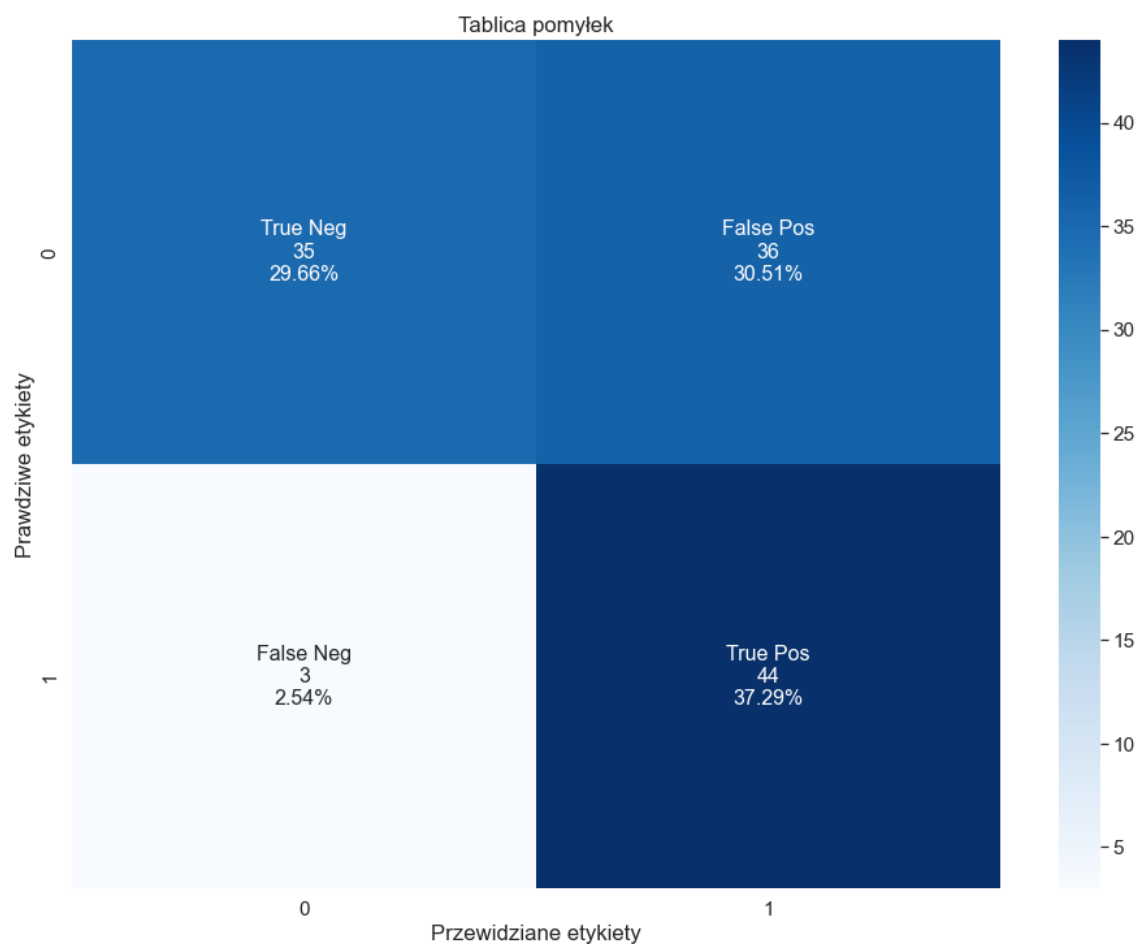
Rysunek 42: Podsumowanie wyników dokładności klasy SoftSetClassifierPercentage

Najlepszą dokładność przewidywanych wyników za pomocą klasy SoftSetClassifierPercentage uzyskaliśmy dla danych znormalizowanych przy użyciu wzoru opierającego się na odchyleniu standardowym - 55.12%. Niemal o 2 punkty procentowe gorszy wynik został uzyskany dla danych znormalizowanych przy użyciu wzoru opierającego się na minimum i maksimum - 53.32%. Najgorszy wynik uzyskaliśmy dla danych nieznormalizowanych - 52.93%. Uzyskane wyniki plasują w naszym rankingu klasyfikator SoftSetClassifierPercentage na przedostatnim miejscu zaraz za klasyfikatorem rozmytym.



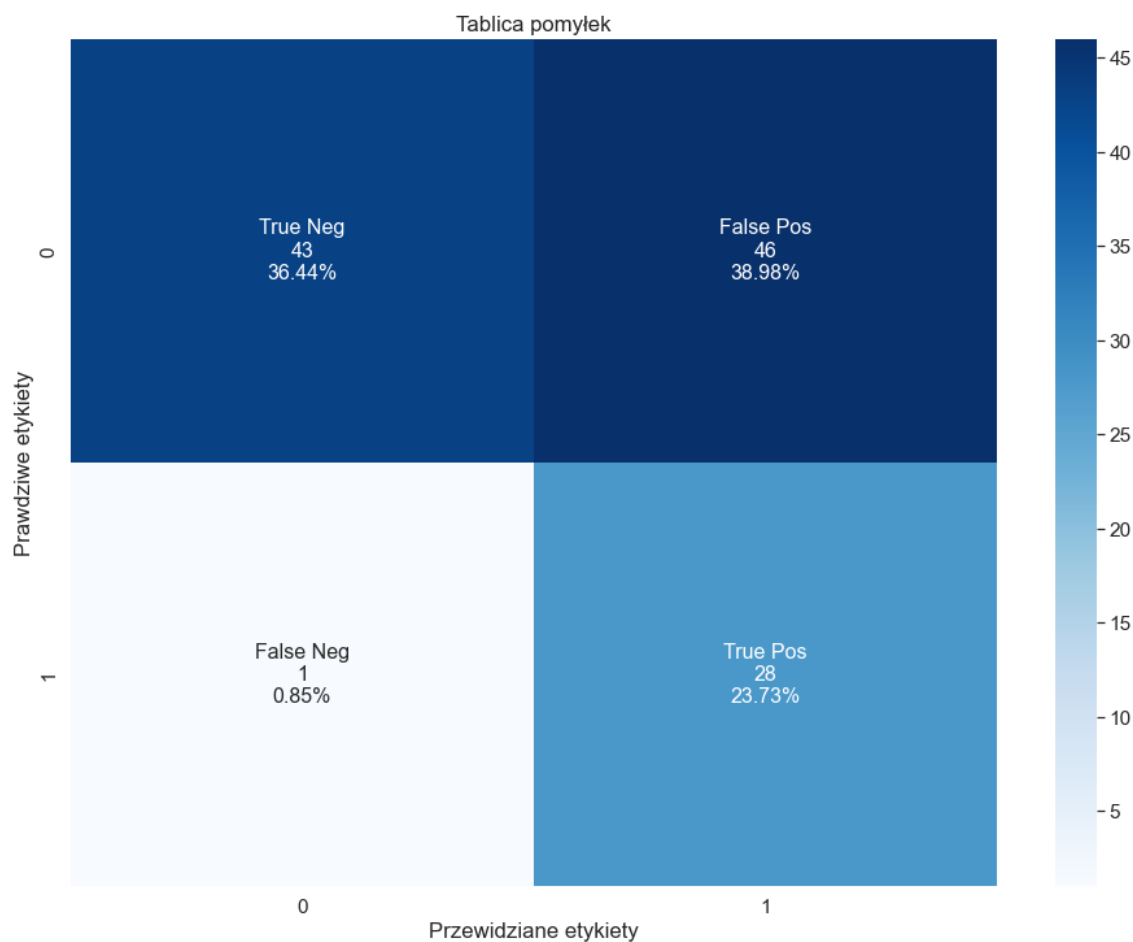
Rysunek 43: Podsumowanie wyników dokładności Klasyfikatora rozmytego

Najlepszy spośród zaproponowanych przez nas klasyfikatorów i jednocześnie trzecim spośród wszystkich poznanych okazał się klasyfikator rozmyty ze średnią dokładnością przewidywań równą 63.14%. Wyniki oscylują między 55% a 75%, co w niektórych przypadkach oznacza zrównanie się dokładnością z klasyfikatorem Naiwnym Bayesa.

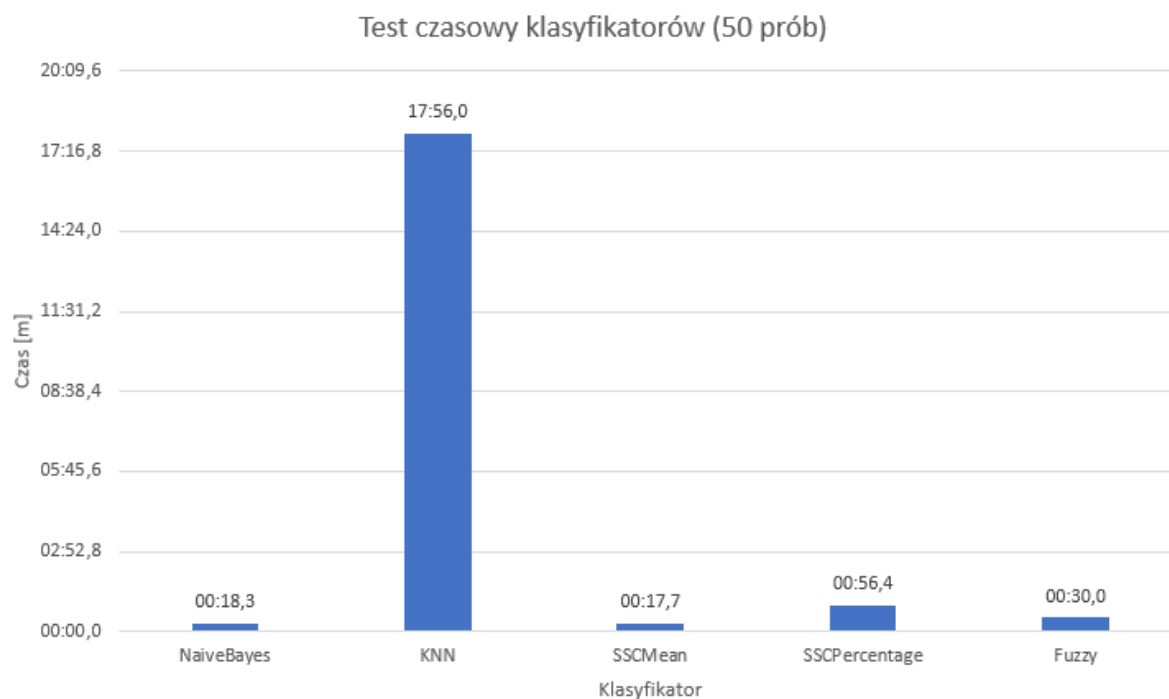


Rysunek 44: Przykładowa tablica pomyłek nr 1

- True Positive (Actual: True, Predicted: True)
- False Positive (Actual: False, Predicted: True)
- True Negative (Actual: False, Predicted: False)
- False Negative (Actual: True, Predicted: False)

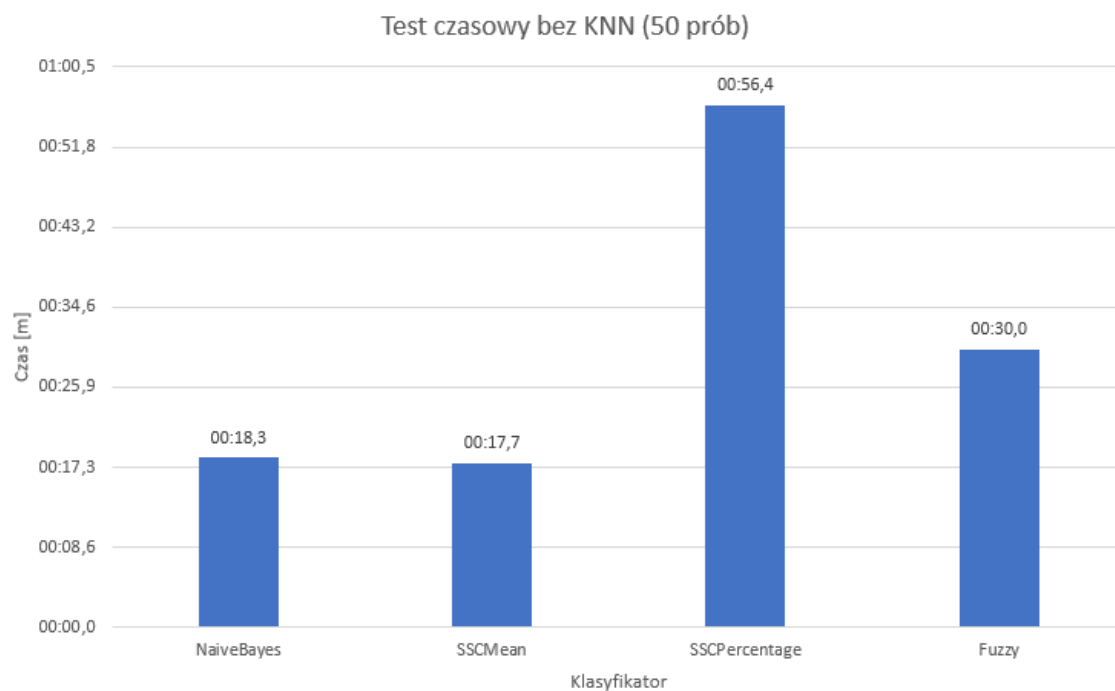


Rysunek 45: Przykładowa tablica pomyłek nr 2



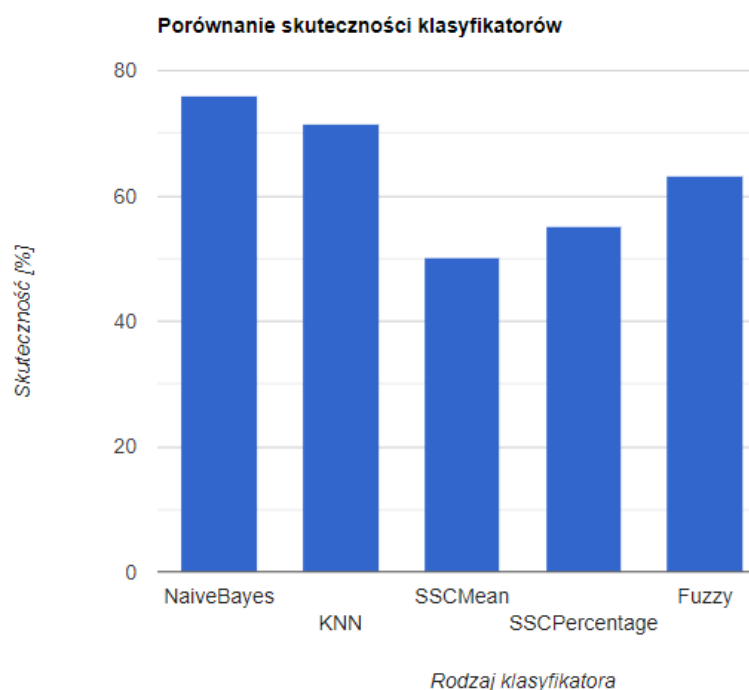
Rysunek 46: Podsumowanie czasowe klasyfikatorów

Czas był mierzony za pomocą biblioteki *datetime* i metody *datetime.now()*. Klasyfikator KNN pod względem szybkości obliczeniowej nie może konkurować z pozostałymi testowanymi przez nas klasyfikatorami. 50 prób klasyfikacji zajęło klasyfikatorowi KNN dokładnie 17 min 56 sek.



Rysunek 47: Podsumowanie czasowe klasyfikatorów bez KNN

Dodatkowy wykres podsumowania szybkości klasyfikatorów bez KNN. Najlepszy wynik osiągnął klasyfikator SSCMean - 17.68 sek. Zaraz za nim klasyfikator Naiwny Bayesa z wynikiem 18.25 sek. Klasyfikator zbiorów rozmytych osiągnął wynik 30.00 sek. Przedostatni, zaraz za KNN procentowy klasyfikator zbiorów miękkich - 56.35 sek.



Rysunek 48: Podsumowanie ogólne

Wyjaśnienie:

- **NaiveBayes**: wybrany został najkorzystniejszy przypadek, czyli dla danych znormalizowanych minmax.
- **KNN**: wybrany został najkorzystniejszy przypadek, czyli dla danych znormalizowanych minmax.
- **SoftSetClassifierMean**: wybrany został najkorzystniejszy przypadek, czyli dla danych nieznormalizowanych.
- **SoftSetClassifierPercentage**: wybrany został najkorzystniejszy przypadek, czyli dla danych znormalizowanych std.

3 Pełen kod aplikacji

3.1 Klasa ProcessingData

```
1 # Klasa statyczna zawierajaca metody przydatne do przetwarzania danych
2 class ProcessingData:
3
4     # metoda tasujaca wiersze pd.DataFrame
5     @staticmethod
6     def shuffle(df: pd.DataFrame) -> pd.DataFrame:
7         """
8         Receives a DataFrame, randomly shuffles it and returns a
9         shuffled version.
10        :param df: pd.DataFrame
11        :return df: pd.DataFrame
12        """
13        # kazdy element od konca do poczatku
14        for idx in range(len(df) - 1, 0, -1):
15            # wybieramy losowo 'rand_idx', czyli nowe miejsce elementu
16            rand_idx = random.randint(0, idx)
17            # zamieniamy
18            df.iloc[idx], df.iloc[rand_idx] = df.iloc[rand_idx], df.iloc
19            [idx]
20            # zwracamy potasowany zbior ze zresetowanymi indeksami
21            return df.reset_index(drop=True)
22
23 # metoda normalizujaca kolumny pd.DataFrame za pomoca normalizacji z
24 # uzcieniem odchylenia standardowego
25 @staticmethod
26 def normalize_std(df: pd.DataFrame, label: str) -> pd.DataFrame:
27     """
28     Receives a DataFrame and a class label to skip during
29     normalization, normalizes data using std normalization and
30     returns the normalized version.
31     :param label: str
32     :param df: pd.DataFrame
33     :return df: pd.DataFrame
34     """
35     # usuwamy kolumny z klasami
36     normalized = df.drop(labels=label, axis=1)
37     # normalizujemy wg wzoru z odchyleniem standardowym i zwracamy
38     # zbior razem z wczeniej usunieta kolumna
39     normalized = (normalized - normalized.mean()) / (normalized.std())
40     return normalized.join(df[label])
41
42 # metoda normalizujaca kolumny pd.DataFrame za pomoca normalizacji
43 # minmax
44 @staticmethod
45 def normalize_minmax(df: pd.DataFrame, label: str) -> pd.DataFrame:
46     """
47     Receives a DataFrame and a class label to skip during
48     normalization, normalizes data using min-max normalization
49     and returns the normalized version.
50     :param label: str
```

```

42         :param df: pd.DataFrame
43         :return df: pd.DataFrame
44         """
45         # usuwamy kolumnę z klasami
46         normalized = df.drop(labels=label, axis=1)
47         # normalizujemy wg wzoru z 'min' i 'max' i zwracamy zbior razem
          z wcześniej usunięta kolumną
48         normalized = (normalized-normalized.min())/(normalized.max()-
          normalized.min())
49         return normalized.join(df[label])
50
51     # metoda dzielacza pd.DataFrame na zbior treningowy oraz walidacyjny
52     @staticmethod
53     def split(df: pd.DataFrame, ratio: float) -> tuple[pd.DataFrame, pd.
          DataFrame]:
54         """
55         Receives a DataFrame and returns two dataframes, split into
          training and validation DataFrames.
56         :param df: pd.DataFrame
57         :param ratio: float
58         :return training_df, validation_df: tuple[pd.DataFrame, pd.
          DataFrame]
59         """
60         # miejsce elementu dzielacego zbior w odpowiedniej proporcji wg
          'ratio'
61         cutoff = int(len(df) * ratio)
62         # zwracamy zbior treningowy i walidacyjny
63         return df[:cutoff].reset_index(drop=True), df[cutoff:].
          reset_index(drop=True)

```

3.2 Klasa NaiveBayes

```
1 # Klasa NaiveBayes zawierajaca klasyfikator Bayesa oraz metody
   pomocnicze
2 from functools import reduce
3
4
5 class NaiveBayes:
6     """
7     labels - stores all the labels existing within the column that we
        want to predict
8     values - stores the mean and standard deviation values of all the
        columns and labels into the class attribute values
9     """
10    # labels - unikalne wartosci z podanej kolumny, ktora chcemy
        przewidziec (czyli u nas 0 i 1)
11    labels: list[float] = []
12    values: list[list[list[float], float]] = []
13
14    # metoda pomocnicza liczaca srednia wartosc w kolumnie
15    @staticmethod
16    def mean(column: list[float]) -> float:
17        """
18        Receives a column of the dataframe with numerical values and
        returns its mean value
19        :param column: list[float]
20        :return mean value of the values in a dataframe column: float
21        """
22        # czyli suma wszystkich wartosci w kolumnie przez dlugosc
        kolumny
23        return sum(column) / len(column)
24
25    # metoda pomocnicza liczaca odchylenie standardowe wartosci w
        kolumnie
26    @staticmethod
27    def std_dev(column: list[float], m: float) -> float:
28        """
29        Receives a column of the dataframe and its mean value and
        returns its standard deviation.
30        :param column: list[float]
31        :param m: float
32        :return standard deviation of the values in a dataframe column:
        float
33        """
34        return math.sqrt(sum(map(lambda x: (x - m) ** 2, column)) / len(
            column))
35
36    # metoda pomocnicza obliczajaca gestosc prawdopodobienstwa
37    @staticmethod
38    def density(x: float, m: float, sd: float) -> float:
39        """
40        Receives a value, plus a mean value and a standard deviation of
        the values in a certain column, and returns its Gaussian
        probability.
41        :param x: float
```



```

42         :param m: float
43         :param sd: float
44         :return Gaussian probability: float
45         """
46         # rozklad normalny / Gaussa
47         return math.exp(-(x - m) ** 2 / (2 * sd ** 2)) / (math.sqrt(2 *
            math.pi) * sd)
48
49     # metoda obliczajaca wartosci przechowywane w tablicach pomocniczych
    klasy
50     def calculate(self, df: pd.DataFrame, label: str) -> None:
51         """
52         Receives a dataframe and the label of the column we want to
            predict and loads mean and standard deviation values into the
            class attribute values.
53         :param df: pd.DataFrame
54         :param label: str
55         :return: None
56         """
57         # lista wartosci oraz lista unikalnych wartosci z podanej
            kolumny 'label'
58         self.values = []
59         self.labels = df[label].unique()
60
61         # dla kazdej unikalnej wartosci z przewidywanej kolumny (czyli u
            nas dla 0 i 1)
62         for category in self.labels:
63             # wartosci ktorych unikalna wartosc kolumny 'label' to '
                category', czyli podzial na czesci ktore maja 0 i 1
64             category_values = df[df[label] == category].drop(labels=
                label, axis=1)
65             # pary: [srednia, odchylenie std] dla kazdej kolumny dla
                kazdej kategorii 'category_values'
66             column = [[self.mean(item), self.std_dev(item, self.mean(
                item))]] for _, item in category_values.items()
67             # dodanie do 'values' wszystkich par [srednia, odchylenie
                std]
68             self.values.append(column)
69
70     # metoda przewidyujaca etykiety klas probek
71     def predict(self, sample: pd.Series, label: str) -> float:
72         """
73         Receives a sample in the form of pd.Series and returns its
            predicted label.
74         :param sample: pd.Series
75         :param label: str - column with class labels
76         :return predicted label of the sample: str
77         """
78         sample = sample.drop(labels=label)
79         probabilities = []
80         # dla kazdej kategorii w 'values'
81         for category in self.values:
82             category_pairs = []
83             # pair to para [srednia, odchylenie]
84             # x to odpowiednia wartosc w 'sample'

```

```

85         for pair, x in zip(category, sample):
86             # dodajemy kolejne obliczone Gaussy
87             category_pairs.append(self.density(x, pair[0], pair[1]))
88             # liczymy prawdopodobienstwa czyli wymnazamy ze soba kolejne
               odpowiednie Gaussy
89             probabilities.append(reduce(lambda z, y: z * y,
               category_pairs))
90         # naszym indeksem jest indeks najwiekszego prawdopodobienstwa
91         idx = max(range(len(probabilities)), key=probabilities.
               __getitem__)
92         # zwracamy przewidywany wynik z 'labels' (u nas 0 lub 1)
93         return self.labels[idx]
94
95     # metoda testujaca dokladnosc przewidywan
96     def test(self, validation_set: pd.DataFrame, label: str) -> None:
97         """
98         Test how accurate the prediction method is
99         :param validation_set: pd.DataFrame - a record from the
               validation dataset
100        :param label: str - column with class labels
101        :return: None
102        """
103        correct = 0
104        # iterrows() jest uzywane do iteracji wierszy dataframe'a
105        for _, row in validation_set.iterrows():
106            # jesli poprawnie przewidzimy etykiety (czyli 0 lub 1)
107            if self.predict(sample=row, label=label) == row[label]:
108                correct += 1
109        # informacja podsumowujaca
110        print(f'Accuracy: {correct / len(validation_set) * 100:.2f}%; {
               correct}/{len(validation_set)}')

```

3.3 Klasa KNN

```
1 # Klasa statyczna KNN zawierajaca klasyfikator KNN oraz metody
   pomocnicze
2 from collections import Counter
3
4 class KNN:
5
6     # metoda obliczajaca odleglosc miedzy probkami za pomoca metryki
7     @staticmethod
8     def distance(x: pd.Series, y: pd.Series, m: int, label: str) ->
       float:
9         """
10         :param x: pd.Series - a record from a dataset to calculate the
              distance
11         :param y: pd.Series - a record from a dataset to calculate the
              distance
12         :param m: int - to calculate the mth root in Euclidean metric
13         :param label: str - column with class labels
14         :return: float - the distance between two records
15         """
16         s = 0
17         x = x.drop(labels=label)
18         y = y.drop(labels=label)
19         # odleglosc miedzy dwoma wektorami, czyli pierwiastek sumy
              wartosci bezwzgleдных kwadratow roznic kolejnych
              wspolrzednych
20         for xi, yi in zip(x, y):
21             s += pow(abs(xi - yi), m)
22         return pow(s, 1 / m)
23
24     # metoda przewidujaca etykiety klasy probki na podstawie etykiet jej
       k-najblizszych sasiadow
25     @staticmethod
26     def cluster(sample: pd.Series, training_set: pd.DataFrame, k: int, m
       : int, label: str) -> str:
27         """
28         :param sample: pd.Series - a record from the validation dataset
29         :param training_set: pd.DataFrame - training dataset
30         :param k: int - k-nearest elements
31         :param m: int - to pass into KNN.distance
32         :param label: str - name of the label that we want to predict
33         :return: str - predicted label of the object
34         """
35         # pary (dystans, outcome) dla zbioru treningowego
36         distances = [(KNN.distance(sample, training_set.iloc[idx], m,
              label), training_set.iloc[idx][label]) for idx in
              training_set.index]
37         # wybieramy 'k' najblizszych wynikow
38         distances = sorted(distances, key=lambda d: (d, random.random()))
             )[:k]
39         # zwracamy etykiety, ktora wytypowalismy najwiecej razy
40         return max(dict(Counter(elem[1] for elem in distances)))
41
42     # metoda testujaca dokladnosc dzialania
```

```

43     @staticmethod
44     def test(training_set: pd.DataFrame, validation_set: pd.DataFrame,
45             k0: int, kn: int, m: int, label: str) -> None:
46         """
47         :param training_set: pd.Series - a record from the validation
48             dataset
49         :param validation_set: pd.DataFrame - training dataset
50         :param k0: int - starting value of k to pass into KNN.cluster
51         :param kn: int - end value of k to pass into KNN.cluster
52         :param m: int - to pass into KNN.cluster
53         :param label: str - name of the label that we want to predict
54         :return: None
55         """
56         # testujemy dla 'k' od 'k0' do 'kn'
57         for k in range(k0, kn + 1):
58             correct = 0
59             # bierzemy kazda probkie (sample) ze zbioru walidacyjnego
60             for sample in validation_set.iloc:
61                 x = KNN.cluster(sample, training_set, k, m, label)
62                 # sprawdzamy czy przewidziany jest taki jak rzeczywisty
63                 if x == sample[label]:
64                     correct += 1
65             # informacja podsumowujaca
66             print(f'Accuracy for {k=}: {float(correct) / len(
67                 validation_set) * 100:.2f}%', correct predictions: {
68                 correct}/{len(validation_set)}')

```

3.4 Klasa SoftSetClassifierMean

```
1 # Klasa SoftSetClassifierMean zawierajaca klasyfikator miękki
   korzystajacy ze sredniej arytmetycznej oraz metody pomocnicze
2 class SoftSetClassifierMean:
3     """
4     labels - stores all the labels existing within the column that we
       want to predict
5     pairs - stores 0 and 1 pairs for every label for every column
6     means - stores mean values for every label for every column
7     """
8     # lista etykiet
9     labels: list[str] = []
10    # lista par 0, 1 dla kazdej kolumny i kazdej cechy
11    pairs: list[list[list[int, int]]] = []
12    # lista srednich dla kazdej kolumny i kazdej cechy
13    means: list[list[float]] = []
14
15    # metoda pomocnicza obliczajaca srednie kolumn
16    @staticmethod
17    def mean(column: list[float]) -> float:
18        """
19        Receives a column of the dataframe with numerical values and
           returns its mean value
20        :param column: list[float]
21        :return mean value of the values in a dataframe column: float
22        """
23        # czyli zwykła suma przez ilość składników
24        return sum(column) / len(column)
25
26    # metoda tworząca tabele wazona zbioru miękkiego
27    def calculate(self, df: pd.DataFrame, label: str) -> None:
28        """
29        Receives a dataframe and the label of the column we want to
           predict and decides whether pairs are [0,1] or [1,0]
30        :param df: pd.DataFrame
31        :param label: str
32        :return: None
33        """
34        self.pairs = []
35        self.means = []
36        # lista unikalnych wartosci z podanej kolumny 'label'
37        # ma byc zawsze ta sama kolejnosc wiec sortujemy
38        self.labels = sorted(df[label].unique())
39
40        # 'category' to 0 i 1
41        for category in self.labels:
42            # wartosci ktorych unikalna wartosc kolumny 'label' to '
               category'
43            # czyli podzial na czesci ktore maja 0 i 1
44            category_values = df[df[label] == category].drop(labels=
               label, axis=1)
45            temp_mean = []
46            temp_pair = []
47            # item to wszystkie wartosci z danej kolumny
```

```

48         for _, item in category_values.items():
49             # srednia z kolumny, dodajemy ja do listy srednich
50             mean = self.mean(item)
51             temp_mean.append(mean)
52             len_lower = 0
53             len_upper = 0
54             # zliczamy elementy wieksze od sredniej i mniejsze od
               sredniej
55             for value in item:
56                 if value > mean:
57                     len_upper += 1
58                 else:
59                     len_lower += 1
60
61             # jesli jest tyle samo powyzej sredniej co ponizej to
62             # wybieramy losowo pare [0, 1] lub [1, 0]
63             if len_upper == len_lower:
64                 temp_pair.append(random.choice(([0, 1],[1, 0])))
65             # jesli wiecej powyzej sredniej to zapisujemy pare [1,
               0]
66             elif len_upper > len_lower:
67                 temp_pair.append([1, 0])
68             # jesli wiecej ponizej sredniej to zapisujemy pare [0,
               1]
69             else:
70                 temp_pair.append([0, 1])
71
72             # uzupełniamy listy wszystkich 'srednich' i wszystkich 'par'
73             self.means.append(temp_mean)
74             self.pairs.append(temp_pair)
75
76 # metoda przewidujaca etykiety klasy probki
77 def predict(self, sample: pd.Series, label: str) -> str:
78     """
79     Receives a sample in the form of pd.Series and returns its
       predicted label
80     :param sample: pd.Series
81     :param label: str
82     :return predicted label of the sample: str
83     """
84     # liczenie elementow dla probki (sample) dla kazdej z klas (0,
       1)
85
86     sample = sample.drop(labels=label)
87     probabilities = []
88
89     # bierzemy po kolei odpowiednie pary i srednie
90     for category, mean in zip(self.pairs, self.means):
91         sample_pairs = []
92         # bierzemy odpowiednia srednia i wartosc z probki (sample)
93         for m, x in zip(mean, sample):
94             # jesli element rowny sredniej
95             # wybieramy losowo pare [0, 1] lub [1, 0]
96             if x == m:
97                 sample_pairs.append(random.choice(([0, 1],[1, 0])))

```

```

98         # jesli element wiekszy od sredniej to zapisujemy pare
          [1, 0]
99     elif x > m:
100         sample_pairs.append([1, 0])
101     # jesli element mniejszy od sredniej to zapisujemy pare
          [0, 1]
102     else:
103         sample_pairs.append([0, 1])
104     temp = []
105
106     # liczenie prawdopodobienstwa dla kazdej klasy
107     for x, y in zip(category, sample_pairs):
108         temp.append(x[0] * y[0] + x[1] * y[1])
109     probabilities.append(sum(temp))
110
111     # naszym indeksem jest indeks najwiekszego prawdopodobienstwa
112     idx = max(range(len(probabilities)), key=probabilities.
        __getitem__)
113     # zwracamy najprawdopodobniejszy wynik z 'labels' (u nas 0 lub
        1)
114     return self.labels[idx]
115
116 # metoda testujaca dokladnosc dzialania Klasyfikatora miemkiego
117 def test(self, validation_set: pd.DataFrame, label: str) -> None:
118     """
119     Test how accurate the prediction method is
120     :param validation_set: pd.DataFrame - a record from the
        validation dataset
121     :param label: str
122     :return: None
123     """
124     correct = 0
125     # iterrows() jest uzywane do iteracji wieszy dataframe'a
126     for _, row in validation_set.iterrows():
127         # jesli poprawnie przewidzimy etykiety
128         if self.predict(sample=row, label=label) == row[label]:
129             correct += 1
130     # informacja podsumowujaca
131     print(f'Accuracy: {correct / len(validation_set) * 100:.2f}%; {
        correct}/{len(validation_set)}')

```

3.5 Klasa SoftSetClassifierPercentage

```
1 # Klasa SoftSetClassifierPercentage zawierajaca klasyfikator miekki
  korzystajacy z procentow oraz metody pomocnicze
2 class SoftSetClassifierPercentage:
3     """
4     labels - stores all the labels existing within the column that we
        want to predict
5     pairs - stores pairs of values between 0 and 1 for every label for
        every column
6     minmaxs - stores min and max pairs of values for every label for
        every column
7     """
8     # lista etykiet
9     labels: list[str] = []
10    # lista par 0, 1 dla kazdej kolumny i kazdej cechy
11    pairs: list[list[list[int, int]]] = []
12    # lista minow i maxow dla kazdej kolumny i kazdej cechy
13    minmaxs: list[list[list[float, float]]] = []
14
15    # metoda pomocnicza obliczajaca srednie kolumn
16    @staticmethod
17    def mean(column: list[float]) -> float:
18        """
19        Receives a column of the dataframe with numerical values and
            returns its mean value
20        :param column: list[float]
21        :return mean value of the values in a dataframe column: float
22        """
23        # czyli suma przez ilosc skladnikow
24        return sum(column) / len(column)
25
26    # metoda tworzaca tabele wazona zbioru miekkiego
27    def calculate(self, df: pd.DataFrame, label: str) -> None:
28        """
29        Receives a dataframe and the label of the column we want to
            predict and decides whether pairs are [0,1] or [1,0]
30        :param df: pd.DataFrame
31        :param label: str
32        :return: None
33        """
34        self.pairs = []
35        self.minmaxs = []
36        # lista unikalnych wartosci z podanej kolumny 'label'
37        # ma byc zawsze ta sama kolejnosc wiec sortujemy
38        self.labels = sorted(df[label].unique())
39
40        # 'category' to 0 i 1
41        for category in self.labels:
42            # wartosci ktorych unikalna wartosc kolumny 'label' to '
                category'
43            # czyli podzial na czesci ktore maja 0 i 1
44            category_values = df[df[label] == category].drop(labels=
                label, axis=1)
45            temp_minmax = []
```



```

46         temp_pair = []
47         # item to wszystkie wartosci z danej kolumny
48         for _, item in category_values.items():
49             # minimum z kolumny
50             min_t = min(item)
51             # maximum z kolumny
52             max_t = max(item)
53             temp_minmax.append([min_t, max_t])
54             # srednia
55             mean = self.mean(item)
56             len_lower = 0
57             len_upper = 0
58             # zliczamy elementy wieksze od sredniej i mniejsze od
              sredniej
59             for value in item:
60                 if value > mean:
61                     len_upper += 1
62                 else:
63                     len_lower += 1
64
65             # jesli jest tyle samo powyzej sredniej co ponizej to
66             # wybieramy losowo pare [0, 1] lub [1, 0]
67             if len_upper == len_lower:
68                 temp_pair.append(random.choice(([0, 1],[1, 0])))
69             # jesli wiecej powyzej sredniej to zapisujemy pare [1,
              0]
70             elif len_upper > len_lower:
71                 temp_pair.append([1, 0])
72             # jesli wiecej ponizej sredniej to zapisujemy pare [0,
              1]
73             else:
74                 temp_pair.append([0, 1])
75             # uzupełniamy listy wszystkich 'minmaxow' i wszystkich 'par'
76             self.minmaxs.append(temp_minmax)
77             self.pairs.append(temp_pair)
78
79         # metoda przewidujaca etykiety klasy probki
80         def predict(self, sample: pd.Series, label: str) -> str:
81             """
82             Receives a sample in the form of pd.Series and returns its
              predicted label
83             :param sample: pd.Series
84             :param label: str
85             :return predicted label of the sample: str
86             """
87             # liczenie elementow dla probki (sample) dla kazdej z klas (0,
              1)
88
89             sample = sample.drop(labels=label)
90             probabilities = []
91
92             # bierzemy po kolei odpowiednie pary i srednie
93             for category, minmax in zip(self.pairs, self.minmaxs):
94                 sample_pairs = []
95                 # bierzemy odpowiednia pare minmax i wartosc z probki (

```

```

    sample)
96     for pair, x in zip(minmax, sample):
97         # proportion = (wartosc_sampla - min) / (max - min)
98         proportion = (x - pair[0]) / (pair[1] - pair[0])
99         # dodajemy liste [1 - prop, prop] do listy wszystkich
           par
100         sample_pairs.append([1 - proportion, proportion])
101
102     temp = []
103     # liczenie prawdopodobienstwa dla kazdej klasy
104     for x, y in zip(category, sample_pairs):
105         temp.append(x[0] * y[0] + x[1] * y[1])
106     probabilities.append(sum(temp))
107 # naszym indeksem jest indeks największego prawdopodobienstwa
108 idx = max(range(len(probabilities)), key=probabilities.
           __getitem__)
109 # zwracamy najprawdopodobniejszy wynik z 'labels' (u nas 0 lub
           1)
110 return self.labels[idx]
111
112 # metoda testujaca dokladnosc dzialania Klasyfikatora miękkiego
113 def test(self, validation_set: pd.DataFrame, label: str) -> None:
114     """
115     Test how accurate the prediction method is
116     :param validation_set: pd.DataFrame - a record from the
           validation dataset
117     :param label: str
118     :return: None
119     """
120     correct = 0
121     # iterrows() jest uzywane do iteracji wieszy dataframe'a
122     for _, row in validation_set.iterrows():
123         # jesli poprawnie przewidzimy etykiety
124         if self.predict(sample=row, label=label) == row[label]:
125             correct += 1
126
127     # informacja podsumowujaca
128     print(f'Accuracy: {correct / len(validation_set) * 100:.2f}%; {
           correct}/{len(validation_set)}')

```

3.6 Klasa FuzzyClassifier

```
1 class Fuzzy:
2     def __init__(self):
3         # slownik slownikow rozmycia dla kazdej kolumny
4         self.antecedents: dict[str: dict[str: tuple[float]]] = {}
5         # slownik zawierajaca nazwe etykiety klasy oraz slownikow z jej
6         # rozmyciem
7         self.consequent: dict[str: dict[str: tuple[float]]] = {}
8         # lista slownikow; kazda regula to jeden slownik
9         self.rules: list[dict[str: str]] = []
10
11     # metoda pozwalajaca na dodanie poprzednika
12     def add_antecedent(self, parameter: str, linguistic_value: str, *
13         args) -> None:
14         if parameter not in self.antecedents:
15             self.antecedents[parameter] = {}
16         self.antecedents[parameter][linguistic_value] = args
17
18     # metoda pozwalajaca na dodanie konsekwencji
19     def add_consequent(self, parameter: str, linguistic_value: str, *
20         args) -> None:
21         if parameter not in self.consequent:
22             self.consequent[parameter] = {}
23         self.consequent[parameter][linguistic_value] = args
24
25     # metoda pozwalajaca dodac regule
26     def add_rule(self, rule: dict[str: str]) -> None:
27         assert set(rule.keys()) == set.union(set(self.antecedents.keys()),
28             set(self.consequent.keys())), print("Nieprawidlowa liczba
29             kategorii w regule.")
30         self.rules.append(rule)
31
32     # metoda obliczajaca przynaleznosc dla funkcji typu trojkat
33     @staticmethod
34     def triangular_function(n: float, a: float, b: float, c: float) ->
35         float:
36         assert a <= b <= c, print("Nieprawidlowe wartosci")
37         if n == a == b or n == b == c:
38             return 1
39         if n <= a:
40             return 0
41         if a < n < b:
42             return (n - a) / (b - a)
43         if n == b:
44             return 1
45         if b < n < c:
46             return (c - n) / (c - b)
47         if n >= c:
48             return 0
49
50     # metoda obliczajaca przynaleznosc dla funkcji typu trapez
51     @staticmethod
52     def trapezoidal_function(n: float, a: float, b: float, c: float, d:
53         float) -> float:
```

```

47         assert a <= b <= c <= d, print("Nieprawidlowe wartosci")
48         if c == d and n >= d:
49             return 1
50         if a == b and n <= a:
51             return 1
52         if n <= a:
53             return 0
54         if a < n < b:
55             return (n - a) / (b - a)
56         if b <= n <= c:
57             return 1
58         if c < n < d:
59             return (d - n) / (d - c)
60         if n >= d:
61             return 0
62
63     # metoda wywołująca funkcje przynależności na podstawie liczby
        argumentow(dla 3: funkcje typu trojkat, a dla 4: funkcja typu
        trapez)
64     def membership_function(self, n: float, *args) -> float:
65         assert len(args) in (3, 4), print("Nieprawidlowa liczba
            argumentow")
66         if len(args) == 3:
67             return self.triangular_function(n, *args)
68         else:
69             return self.trapezoidal_function(n, *args)
70
71     # metoda rozmywająca dane wejściowe
72     def fuzzify(self, column: str, n: float) -> dict[str, float]:
73         return {antecedent: self.membership_function(n, *self.
            antecedents[column][antecedent]) for antecedent in self.
            antecedents[column].keys()}
74
75     # metoda obliczająca stopień spełnienia danej zasady
76     @staticmethod
77     def rule_fulfillment(rule: dict[str: str], fuzzy_values: dict[str:
        dict[str: float]]) -> tuple[str, float]:
78         label = list(rule.keys())[-1]
79         minimum = float('-inf')
80         # dla każdej wartości w danej zasadzie
81         for name, linguistic_value in list(rule.items())[:-1]:
82             maximum = float('-inf')
83             # jeśli mamy wartości rozdzielone 'or', to bierzemy ich
                maksimum
84             for lvs in linguistic_value.split(' | '):
85                 maximum = max(fuzzy_values[name][lvs], maximum)
86             # bierzemy minimum stopni spełnienia wartości danej zasady
87             minimum = min(minimum, maximum)
88         return rule[label], minimum
89
90     # metoda obliczająca powierzchnię danego wyniku
91     @staticmethod
92     def area(a: float, c: float) -> float:
93         return (c - a) / 2
94

```

```

95     # metoda srodka ciezkosci
96     @staticmethod
97     def cog(a: float, b: float, c: float) -> float:
98         return (a + b + c) / 3
99
100    # metoda agregujaca, wyostrzajaca wartosc koncowa za pomoca metody
    srodka ciezkosci
101    def aggregate(self, label: str, outputs: defaultdict) -> float:
102        # licznik i mianownik dla funkcji agregujacej
103        counter = []
104        denominator = []
105        # obliczanie powierzchni oraz srodka ciezkosci kazdej etykiety
        funkcji przynaleznosci
106        for name, membership in outputs.items():
107            if not membership:
108                continue
109            a, b, c = self.consequent[label][name]
110            area = self.area(a, c)
111            cog = self.cog(a, b, c)
112            counter.append(membership * area * cog)
113            denominator.append(membership * area)
114        return round(sum(counter) / sum(denominator), 3)
115
116    # metoda przewidujaca etykiety klasy
117    def compute(self, sample: pd.Series) -> tuple[str, float]:
118        # rozmycie probki
119        fuzzy_values = {column: self.fuzzify(column, sample[column]) for
        column in self.antecedents.keys()}
120
121        # obliczenie stopnia spelnienia zasad
122        fulfillments = [self.rule_fulfillment(rule, fuzzy_values) for
        rule in self.rules]
123
124        # wybranie maksymalnego spelnienia reguly dla danych wartosci
125        outputs: defaultdict = defaultdict(lambda: 0)
126        for linguistic_value, fulfillment in fulfillments:
127            outputs[linguistic_value] = max(outputs[linguistic_value],
        fulfillment)
128        outputs = dict(outputs)
129        label = list(self.consequent.keys())[0]
130        # wybor przewidzianej etykiety klasy: w przypadku remisu dla
        dwoch etykiet wybieramy chorobe
131        output_names = sorted(outputs.keys(), key=lambda x: (-1 *
        outputs[x], x))
132        # print(sample['Outcome'], self.aggregate(label, outputs),
        outputs)
133        return output_names[0], self.aggregate(label, outputs)
134
135    # metoda pomocnicza wyswietlajaca wykres rozmycia cechy
136    def view(self, parameter: str) -> None:
137        # lista kolorow do wykresow
138        colors: dict[int: tuple] = {2: ('#00b100', '#ff5c38'),
139                                     3: ('#00b100', '#d5c731', '#ff5c38'),
140                                     4: ('#00b100', '#adcd27', '#f1b438'),

```

```

141         '#ff5c38'),
142         5: ('#00b100', '#93cb20', '#d5c731',
143            '#f9a539', '#ff5c38')}]
144
145 # globalne parametry dla wykresow matplotlib.pyplot
146 plt.rcParams["figure.figsize"] = (14, 8)
147 if parameter in self.consequent.keys():
148     source = self.consequent[parameter]
149 else:
150     source = self.antecedents[parameter]
151
152 # stworzenie dziedziny
153 minimum: int = floor(list(source.values())[0][0])
154 maximum: int = ceil(list(source.values())[-1][-1])
155 x = np.linspace(minimum, maximum, num=(maximum - minimum) * 100
156                + 1)
157
158 # stworzenie wykresow funkcji przynaleznosci
159 for k, v, color in zip(source.keys(), source.values(), colors[
160     len(source.keys())]):
161     y = np.array([self.membership_function(i, *v) for i in x])
162     plt.plot(x, y, color, label=k)
163
164 # nadanie parametrow wykresu
165 plt.rc('font', size=16)
166 plt.rc('axes', titlesize=14)
167 plt.rc('axes', labelszize=14)
168 plt.rc('xtick', labelszize=14)
169 plt.rc('ytick', labelszize=14)
170 plt.rc('legend', fontsize=14)
171 plt.rc('figure', titlesize=16)
172 plt.title(parameter)
173 plt.xlabel('Membership')
174 plt.ylabel(parameter)
175 plt.legend(loc="upper right")
176 plt.ylim(0, 1)
177 plt.show()

```

3.7 Dodawanie poprzedników i konsekwencji

```

1 # Dodawanie poprzednikow i konsekwencji
2 fuzzy = Fuzzy()
3
4 fuzzy.add_antecedent('Pregnancies', 'low', 0, 0, 5)
5 fuzzy.add_antecedent('Pregnancies', 'medium', 0, 5, 10)
6 fuzzy.add_antecedent('Pregnancies', 'high', 5, 10, 17, 17)
7
8 fuzzy.add_antecedent('Glucose', 'low', 44, 44, 86, 99)
9 fuzzy.add_antecedent('Glucose', 'medium_low', 86, 99, 112)
10 fuzzy.add_antecedent('Glucose', 'medium', 99, 112, 125)
11 fuzzy.add_antecedent('Glucose', 'medium_high', 112, 125, 138)
12 fuzzy.add_antecedent('Glucose', 'high', 125, 138, 199, 199)
13
14 fuzzy.add_antecedent('BloodPressure', 'low', 24, 24, 75, 80)
15 fuzzy.add_antecedent('BloodPressure', 'medium_low', 75, 80, 85)

```

```

16     fuzzy.add_antecedent('BloodPressure', 'medium', 80, 85, 90)
17     fuzzy.add_antecedent('BloodPressure', 'medium_high', 85, 90, 95)
18     fuzzy.add_antecedent('BloodPressure', 'high', 90, 95, 122, 122)
19
20     fuzzy.add_antecedent('SkinThickness', 'low', 7, 7, 15, 22)
21     fuzzy.add_antecedent('SkinThickness', 'medium_low', 15, 22, 29)
22     fuzzy.add_antecedent('SkinThickness', 'medium', 22, 29, 36)
23     fuzzy.add_antecedent('SkinThickness', 'medium_high', 29, 36, 43)
24     fuzzy.add_antecedent('SkinThickness', 'high', 36, 43, 99, 99)
25
26     fuzzy.add_antecedent('Insulin', 'low', 14, 14, 76, 125)
27     fuzzy.add_antecedent('Insulin', 'medium', 76, 125, 190)
28     fuzzy.add_antecedent('Insulin', 'high', 125, 190, 846, 846)
29
30     fuzzy.add_antecedent('BMI', 'underweight', 0, 0, 16, 22)
31     fuzzy.add_antecedent('BMI', 'healthy_weight', 16, 22, 28)
32     fuzzy.add_antecedent('BMI', 'overweight', 22, 28, 34)
33     fuzzy.add_antecedent('BMI', 'obese', 28, 34, 67.1, 67.1)
34
35     fuzzy.add_antecedent('DiabetesPedigreeFunction', 'low', 0, 0,
36                           0.25, 0.5)
37     fuzzy.add_antecedent('DiabetesPedigreeFunction', 'medium', 0.25,
38                           0.5, 0.75)
39     fuzzy.add_antecedent('DiabetesPedigreeFunction', 'high', 0.5,
40                           0.75, 2.42, 2.42)
41
42     fuzzy.add_antecedent('Age', 'low', 21, 21, 34, 45)
43     fuzzy.add_antecedent('Age', 'medium', 34, 45, 56)
44     fuzzy.add_antecedent('Age', 'high', 45, 56, 81, 81)
45
46     for elem in fuzzy.antecedents:
47         fuzzy.view(elem)
48
49     fuzzy.add_consequent('Outcome', 'low', 0, 0, 1)
50     fuzzy.add_consequent('Outcome', 'high', 0, 1, 1)
51     fuzzy.view('Outcome')

```

3.8 Dodawanie reguł

```

1     # Dodawanie reguł
2     # Większość niskich
3     # Same niskie (1) 1
4     fuzzy.add_rule({'Pregnancies': 'low | medium',
5                     'Glucose': 'low | medium_low | medium',
6                     'BloodPressure': 'low | medium_low | medium',
7                     'SkinThickness': 'low | medium_low | medium',
8                     'Insulin': 'low | medium',
9                     'BMI': 'underweight | healthy_weight',
10                    'DiabetesPedigreeFunction': 'low | medium',
11                    'Age': 'low | medium',
12                    'Outcome': 'low'})
13
14     # Pojedyncze wysokie (8) 9
15     fuzzy.add_rule({'Pregnancies': 'medium | high',

```

```

15         'Glucose': 'low | medium_low | medium',
16         'BloodPressure': 'low | medium_low | medium',
17         'SkinThickness': 'low | medium_low | medium',
18         'Insulin': 'low | medium',
19         'BMI': 'underweight | healthy_weight',
20         'DiabetesPedigreeFunction': 'low | medium',
21         'Age': 'low | medium',
22         'Outcome': 'low'})
23     fuzzy.add_rule({'Pregnancies': 'low | medium',
24                    'Glucose': 'medium | medium_high | high',
25                    'BloodPressure': 'low | medium_low | medium',
26                    'SkinThickness': 'low | medium_low | medium',
27                    'Insulin': 'low | medium',
28                    'BMI': 'underweight | healthy_weight',
29                    'DiabetesPedigreeFunction': 'low | medium',
30                    'Age': 'low | medium',
31                    'Outcome': 'low'})
32     fuzzy.add_rule({'Pregnancies': 'low | medium',
33                    'Glucose': 'low | medium_low | medium',
34                    'BloodPressure': 'medium | medium_high | high',
35                    'SkinThickness': 'low | medium_low | medium',
36                    'Insulin': 'low | medium',
37                    'BMI': 'underweight | healthy_weight',
38                    'DiabetesPedigreeFunction': 'low | medium',
39                    'Age': 'low | medium',
40                    'Outcome': 'low'})
41     fuzzy.add_rule({'Pregnancies': 'low | medium',
42                    'Glucose': 'low | medium_low | medium',
43                    'BloodPressure': 'low | medium_low | medium',
44                    'SkinThickness': 'medium | medium_high | high',
45                    'Insulin': 'low | medium',
46                    'BMI': 'underweight | healthy_weight',
47                    'DiabetesPedigreeFunction': 'low | medium',
48                    'Age': 'low | medium',
49                    'Outcome': 'low'})
50     fuzzy.add_rule({'Pregnancies': 'low | medium',
51                    'Glucose': 'low | medium_low | medium',
52                    'BloodPressure': 'low | medium_low | medium',
53                    'SkinThickness': 'low | medium_low | medium',
54                    'Insulin': 'medium | high',
55                    'BMI': 'underweight | healthy_weight',
56                    'DiabetesPedigreeFunction': 'low | medium',
57                    'Age': 'low | medium',
58                    'Outcome': 'low'})
59     fuzzy.add_rule({'Pregnancies': 'low | medium',
60                    'Glucose': 'low | medium_low | medium',
61                    'BloodPressure': 'low | medium_low | medium',
62                    'SkinThickness': 'low | medium_low | medium',
63                    'Insulin': 'low | medium',
64                    'BMI': 'obese | overweight',
65                    'DiabetesPedigreeFunction': 'low | medium',
66                    'Age': 'low | medium',
67                    'Outcome': 'low'})
68     fuzzy.add_rule({'Pregnancies': 'low | medium',
69                    'Glucose': 'low | medium_low | medium',

```



```

70         'BloodPressure': 'low | medium_low | medium',
71         'SkinThickness': 'low | medium_low | medium',
72         'Insulin': 'low | medium',
73         'BMI': 'underweight | healthy_weight',
74         'DiabetesPedigreeFunction': 'medium | high',
75         'Age': 'low | medium',
76         'Outcome': 'low'})
77     fuzzy.add_rule({'Pregnancies': 'low | medium',
78                    'Glucose': 'low | medium_low | medium',
79                    'BloodPressure': 'low | medium_low | medium',
80                    'SkinThickness': 'low | medium_low | medium',
81                    'Insulin': 'low | medium',
82                    'BMI': 'underweight | healthy_weight',
83                    'DiabetesPedigreeFunction': 'low | medium',
84                    'Age': 'medium | high',
85                    'Outcome': 'low'})
86     # Pary Pregnancies (7) 16
87     fuzzy.add_rule({'Pregnancies': 'medium | high',
88                    'Glucose': 'medium | medium_high | high',
89                    'BloodPressure': 'low | medium_low | medium',
90                    'SkinThickness': 'low | medium_low | medium',
91                    'Insulin': 'low | medium',
92                    'BMI': 'underweight | healthy_weight',
93                    'DiabetesPedigreeFunction': 'low | medium',
94                    'Age': 'low | medium',
95                    'Outcome': 'low'})
96     fuzzy.add_rule({'Pregnancies': 'medium | high',
97                    'Glucose': 'low | medium_low | medium',
98                    'BloodPressure': 'medium | medium_high | high',
99                    'SkinThickness': 'low | medium_low | medium',
100                   'Insulin': 'low | medium',
101                   'BMI': 'underweight | healthy_weight',
102                   'DiabetesPedigreeFunction': 'low | medium',
103                   'Age': 'low | medium',
104                   'Outcome': 'low'})
105     fuzzy.add_rule({'Pregnancies': 'medium | high',
106                    'Glucose': 'low | medium_low | medium',
107                    'BloodPressure': 'low | medium_low | medium',
108                    'SkinThickness': 'medium | medium_high | high',
109                    'Insulin': 'low | medium',
110                    'BMI': 'underweight | healthy_weight',
111                    'DiabetesPedigreeFunction': 'low | medium',
112                    'Age': 'low | medium',
113                    'Outcome': 'low'})
114     fuzzy.add_rule({'Pregnancies': 'medium | high',
115                    'Glucose': 'low | medium_low | medium',
116                    'BloodPressure': 'low | medium_low | medium',
117                    'SkinThickness': 'low | medium_low | medium',
118                    'Insulin': 'medium | high',
119                    'BMI': 'underweight | healthy_weight',
120                    'DiabetesPedigreeFunction': 'low | medium',
121                    'Age': 'low | medium',
122                    'Outcome': 'low'})
123     fuzzy.add_rule({'Pregnancies': 'medium | high',
124                    'Glucose': 'low | medium_low | medium',

```

```

125         'BloodPressure': 'low | medium_low | medium',
126         'SkinThickness': 'low | medium_low | medium',
127         'Insulin': 'low | medium',
128         'BMI': 'obese | overweight',
129         'DiabetesPedigreeFunction': 'low | medium',
130         'Age': 'low | medium',
131         'Outcome': 'low'})
132     fuzzy.add_rule({'Pregnancies': 'medium | high',
133                    'Glucose': 'low | medium_low | medium',
134                    'BloodPressure': 'low | medium_low | medium',
135                    'SkinThickness': 'low | medium_low | medium',
136                    'Insulin': 'low | medium',
137                    'BMI': 'underweight | healthy_weight',
138                    'DiabetesPedigreeFunction': 'medium | high',
139                    'Age': 'low | medium',
140                    'Outcome': 'low'})
141     fuzzy.add_rule({'Pregnancies': 'medium | high',
142                    'Glucose': 'low | medium_low | medium',
143                    'BloodPressure': 'low | medium_low | medium',
144                    'SkinThickness': 'low | medium_low | medium',
145                    'Insulin': 'low | medium',
146                    'BMI': 'underweight | healthy_weight',
147                    'DiabetesPedigreeFunction': 'low | medium',
148                    'Age': 'medium | high',
149                    'Outcome': 'low'})
150     # Pary Glucose (6) 22
151     fuzzy.add_rule({'Pregnancies': 'low | medium',
152                    'Glucose': 'medium | medium_high | high',
153                    'BloodPressure': 'medium | medium_high | high',
154                    'SkinThickness': 'low | medium_low | medium',
155                    'Insulin': 'low | medium',
156                    'BMI': 'underweight | healthy_weight',
157                    'DiabetesPedigreeFunction': 'low | medium',
158                    'Age': 'low | medium',
159                    'Outcome': 'low'})
160     fuzzy.add_rule({'Pregnancies': 'low | medium',
161                    'Glucose': 'medium | medium_high | high',
162                    'BloodPressure': 'low | medium_low | medium',
163                    'SkinThickness': 'medium | medium_high | high',
164                    'Insulin': 'low | medium',
165                    'BMI': 'underweight | healthy_weight',
166                    'DiabetesPedigreeFunction': 'low | medium',
167                    'Age': 'low | medium',
168                    'Outcome': 'low'})
169     fuzzy.add_rule({'Pregnancies': 'low | medium',
170                    'Glucose': 'medium | medium_high | high',
171                    'BloodPressure': 'low | medium_low | medium',
172                    'SkinThickness': 'low | medium_low | medium',
173                    'Insulin': 'medium | high',
174                    'BMI': 'underweight | healthy_weight',
175                    'DiabetesPedigreeFunction': 'low | medium',
176                    'Age': 'low | medium',
177                    'Outcome': 'low'})
178     fuzzy.add_rule({'Pregnancies': 'low | medium',
179                    'Glucose': 'medium | medium_high | high',

```

```

180         'BloodPressure': 'low | medium_low | medium',
181         'SkinThickness': 'low | medium_low | medium',
182         'Insulin': 'low | medium',
183         'BMI': 'obese | overweight',
184         'DiabetesPedigreeFunction': 'low | medium',
185         'Age': 'low | medium',
186         'Outcome': 'low'})
187 fuzzy.add_rule({'Pregnancies': 'low | medium',
188                 'Glucose': 'medium | medium_high | high',
189                 'BloodPressure': 'low | medium_low | medium',
190                 'SkinThickness': 'low | medium_low | medium',
191                 'Insulin': 'low | medium',
192                 'BMI': 'underweight | healthy_weight',
193                 'DiabetesPedigreeFunction': 'medium | high',
194                 'Age': 'low | medium',
195                 'Outcome': 'low'})
196 fuzzy.add_rule({'Pregnancies': 'low | medium',
197                 'Glucose': 'medium | medium_high | high',
198                 'BloodPressure': 'low | medium_low | medium',
199                 'SkinThickness': 'low | medium_low | medium',
200                 'Insulin': 'low | medium',
201                 'BMI': 'underweight | healthy_weight',
202                 'DiabetesPedigreeFunction': 'low | medium',
203                 'Age': 'medium | high',
204                 'Outcome': 'low'})
205 # Pary BloodPressure (5) 27
206 fuzzy.add_rule({'Pregnancies': 'low | medium',
207                 'Glucose': 'low | medium_low | medium',
208                 'BloodPressure': 'medium | medium_high | high',
209                 'SkinThickness': 'medium | medium_high | high',
210                 'Insulin': 'low | medium',
211                 'BMI': 'underweight | healthy_weight',
212                 'DiabetesPedigreeFunction': 'low | medium',
213                 'Age': 'low | medium',
214                 'Outcome': 'low'})
215 fuzzy.add_rule({'Pregnancies': 'low | medium',
216                 'Glucose': 'low | medium_low | medium',
217                 'BloodPressure': 'medium | medium_high | high',
218                 'SkinThickness': 'low | medium_low | medium',
219                 'Insulin': 'medium | high',
220                 'BMI': 'underweight | healthy_weight',
221                 'DiabetesPedigreeFunction': 'low | medium',
222                 'Age': 'low | medium',
223                 'Outcome': 'low'})
224 fuzzy.add_rule({'Pregnancies': 'low | medium',
225                 'Glucose': 'low | medium_low | medium',
226                 'BloodPressure': 'medium | medium_high | high',
227                 'SkinThickness': 'low | medium_low | medium',
228                 'Insulin': 'low | medium',
229                 'BMI': 'obese | overweight',
230                 'DiabetesPedigreeFunction': 'low | medium',
231                 'Age': 'low | medium',
232                 'Outcome': 'low'})
233 fuzzy.add_rule({'Pregnancies': 'low | medium',
234                 'Glucose': 'low | medium_low | medium',

```

```

235         'BloodPressure': 'medium | medium_high | high',
236         'SkinThickness': 'low | medium_low | medium',
237         'Insulin': 'low | medium',
238         'BMI': 'underweight | healthy_weight',
239         'DiabetesPedigreeFunction': 'medium | high',
240         'Age': 'low | medium',
241         'Outcome': 'low'})
242     fuzzy.add_rule({'Pregnancies': 'low | medium',
243         'Glucose': 'low | medium_low | medium',
244         'BloodPressure': 'medium | medium_high | high',
245         'SkinThickness': 'low | medium_low | medium',
246         'Insulin': 'low | medium',
247         'BMI': 'underweight | healthy_weight',
248         'DiabetesPedigreeFunction': 'low | medium',
249         'Age': 'medium | high',
250         'Outcome': 'low'})
251     # Pary SkinThickness (4) 31
252     fuzzy.add_rule({'Pregnancies': 'low | medium',
253         'Glucose': 'low | medium_low | medium',
254         'BloodPressure': 'low | medium_low | medium',
255         'SkinThickness': 'medium | medium_high | high',
256         'Insulin': 'medium | high',
257         'BMI': 'underweight | healthy_weight',
258         'DiabetesPedigreeFunction': 'low | medium',
259         'Age': 'low | medium',
260         'Outcome': 'low'})
261     fuzzy.add_rule({'Pregnancies': 'low | medium',
262         'Glucose': 'low | medium_low | medium',
263         'BloodPressure': 'low | medium_low | medium',
264         'SkinThickness': 'medium | medium_high | high',
265         'Insulin': 'low | medium',
266         'BMI': 'obese | overweight',
267         'DiabetesPedigreeFunction': 'low | medium',
268         'Age': 'low | medium',
269         'Outcome': 'low'})
270     fuzzy.add_rule({'Pregnancies': 'low | medium',
271         'Glucose': 'low | medium_low | medium',
272         'BloodPressure': 'low | medium_low | medium',
273         'SkinThickness': 'medium | medium_high | high',
274         'Insulin': 'low | medium',
275         'BMI': 'underweight | healthy_weight',
276         'DiabetesPedigreeFunction': 'medium | high',
277         'Age': 'low | medium',
278         'Outcome': 'low'})
279     fuzzy.add_rule({'Pregnancies': 'low | medium',
280         'Glucose': 'low | medium_low | medium',
281         'BloodPressure': 'low | medium_low | medium',
282         'SkinThickness': 'medium | medium_high | high',
283         'Insulin': 'low | medium',
284         'BMI': 'underweight | healthy_weight',
285         'DiabetesPedigreeFunction': 'low | medium',
286         'Age': 'medium | high',
287         'Outcome': 'low'})
288     # Pary Insulin (3) 34
289     fuzzy.add_rule({'Pregnancies': 'low | medium',

```

```

290         'Glucose': 'low | medium_low | medium',
291         'BloodPressure': 'low | medium_low | medium',
292         'SkinThickness': 'low | medium_low | medium',
293         'Insulin': 'medium | high',
294         'BMI': 'obese | overweight',
295         'DiabetesPedigreeFunction': 'low | medium',
296         'Age': 'low | medium',
297         'Outcome': 'low'})
298 fuzzy.add_rule({'Pregnancies': 'low | medium',
299                'Glucose': 'low | medium_low | medium',
300                'BloodPressure': 'low | medium_low | medium',
301                'SkinThickness': 'low | medium_low | medium',
302                'Insulin': 'medium | high',
303                'BMI': 'underweight | healthy_weight',
304                'DiabetesPedigreeFunction': 'medium | high',
305                'Age': 'low | medium',
306                'Outcome': 'low'})
307 fuzzy.add_rule({'Pregnancies': 'low | medium',
308                'Glucose': 'low | medium_low | medium',
309                'BloodPressure': 'low | medium_low | medium',
310                'SkinThickness': 'low | medium_low | medium',
311                'Insulin': 'medium | high',
312                'BMI': 'underweight | healthy_weight',
313                'DiabetesPedigreeFunction': 'low | medium',
314                'Age': 'medium | high',
315                'Outcome': 'low'})
316 # Pary BMI (2) 36
317 fuzzy.add_rule({'Pregnancies': 'low | medium',
318                'Glucose': 'low | medium_low | medium',
319                'BloodPressure': 'low | medium_low | medium',
320                'SkinThickness': 'low | medium_low | medium',
321                'Insulin': 'low | medium',
322                'BMI': 'obese | overweight',
323                'DiabetesPedigreeFunction': 'medium | high',
324                'Age': 'low | medium',
325                'Outcome': 'low'})
326 fuzzy.add_rule({'Pregnancies': 'low | medium',
327                'Glucose': 'low | medium_low | medium',
328                'BloodPressure': 'low | medium_low | medium',
329                'SkinThickness': 'low | medium_low | medium',
330                'Insulin': 'low | medium',
331                'BMI': 'obese | overweight',
332                'DiabetesPedigreeFunction': 'low | medium',
333                'Age': 'medium | high',
334                'Outcome': 'low'})
335 # Pary DiabetesPedigreeFunction (1) 37
336 fuzzy.add_rule({'Pregnancies': 'low | medium',
337                'Glucose': 'low | medium_low | medium',
338                'BloodPressure': 'low | medium_low | medium',
339                'SkinThickness': 'low | medium_low | medium',
340                'Insulin': 'low | medium',
341                'BMI': 'underweight | healthy_weight',
342                'DiabetesPedigreeFunction': 'medium | high',
343                'Age': 'medium | high',
344                'Outcome': 'low'})

```

```

345 # Trojki Pregnancies i Glucose (6) 43
346 fuzzy.add_rule({'Pregnancies': 'medium | high',
347                 'Glucose': 'medium | medium_high | high',
348                 'BloodPressure': 'medium | medium_high | high',
349                 'SkinThickness': 'low | medium_low | medium',
350                 'Insulin': 'low | medium',
351                 'BMI': 'underweight | healthy_weight',
352                 'DiabetesPedigreeFunction': 'low | medium',
353                 'Age': 'low | medium',
354                 'Outcome': 'low'})
355 fuzzy.add_rule({'Pregnancies': 'medium | high',
356                 'Glucose': 'medium | medium_high | high',
357                 'BloodPressure': 'low | medium_low | medium',
358                 'SkinThickness': 'medium | medium_high | high',
359                 'Insulin': 'low | medium',
360                 'BMI': 'underweight | healthy_weight',
361                 'DiabetesPedigreeFunction': 'low | medium',
362                 'Age': 'low | medium',
363                 'Outcome': 'low'})
364 fuzzy.add_rule({'Pregnancies': 'medium | high',
365                 'Glucose': 'medium | medium_high | high',
366                 'BloodPressure': 'low | medium_low | medium',
367                 'SkinThickness': 'low | medium_low | medium',
368                 'Insulin': 'medium | high',
369                 'BMI': 'underweight | healthy_weight',
370                 'DiabetesPedigreeFunction': 'low | medium',
371                 'Age': 'low | medium',
372                 'Outcome': 'low'})
373 fuzzy.add_rule({'Pregnancies': 'medium | high',
374                 'Glucose': 'medium | medium_high | high',
375                 'BloodPressure': 'low | medium_low | medium',
376                 'SkinThickness': 'low | medium_low | medium',
377                 'Insulin': 'low | medium',
378                 'BMI': 'overweight | obese',
379                 'DiabetesPedigreeFunction': 'low | medium',
380                 'Age': 'low | medium',
381                 'Outcome': 'low'})
382 fuzzy.add_rule({'Pregnancies': 'medium | high',
383                 'Glucose': 'medium | medium_high | high',
384                 'BloodPressure': 'low | medium_low | medium',
385                 'SkinThickness': 'low | medium_low | medium',
386                 'Insulin': 'low | medium',
387                 'BMI': 'underweight | healthy_weight',
388                 'DiabetesPedigreeFunction': 'medium | high',
389                 'Age': 'low | medium',
390                 'Outcome': 'low'})
391 fuzzy.add_rule({'Pregnancies': 'medium | high',
392                 'Glucose': 'medium | medium_high | high',
393                 'BloodPressure': 'low | medium_low | medium',
394                 'SkinThickness': 'low | medium_low | medium',
395                 'Insulin': 'low | medium',
396                 'BMI': 'underweight | healthy_weight',
397                 'DiabetesPedigreeFunction': 'low | medium',
398                 'Age': 'medium | high',
399                 'Outcome': 'low'})

```

```

400 # Trojki Pregnancies i BloodPressure (5) 48
401 fuzzy.add_rule({'Pregnancies': 'medium | high',
402                 'Glucose': 'low | medium_low | medium',
403                 'BloodPressure': 'medium | medium_high | high',
404                 'SkinThickness': 'medium | medium_high | high',
405                 'Insulin': 'low | medium',
406                 'BMI': 'underweight | healthy_weight',
407                 'DiabetesPedigreeFunction': 'low | medium',
408                 'Age': 'low | medium',
409                 'Outcome': 'low'})
410 fuzzy.add_rule({'Pregnancies': 'medium | high',
411                 'Glucose': 'low | medium_low | medium',
412                 'BloodPressure': 'medium | medium_high | high',
413                 'SkinThickness': 'low | medium_low | medium',
414                 'Insulin': 'medium | high',
415                 'BMI': 'underweight | healthy_weight',
416                 'DiabetesPedigreeFunction': 'low | medium',
417                 'Age': 'low | medium',
418                 'Outcome': 'low'})
419 fuzzy.add_rule({'Pregnancies': 'medium | high',
420                 'Glucose': 'low | medium_low | medium',
421                 'BloodPressure': 'medium | medium_high | high',
422                 'SkinThickness': 'low | medium_low | medium',
423                 'Insulin': 'low | medium',
424                 'BMI': 'overweight | obese',
425                 'DiabetesPedigreeFunction': 'low | medium',
426                 'Age': 'low | medium',
427                 'Outcome': 'low'})
428 fuzzy.add_rule({'Pregnancies': 'medium | high',
429                 'Glucose': 'low | medium_low | medium',
430                 'BloodPressure': 'medium | medium_high | high',
431                 'SkinThickness': 'low | medium_low | medium',
432                 'Insulin': 'low | medium',
433                 'BMI': 'underweight | healthy_weight',
434                 'DiabetesPedigreeFunction': 'medium | high',
435                 'Age': 'low | medium',
436                 'Outcome': 'low'})
437 fuzzy.add_rule({'Pregnancies': 'medium | high',
438                 'Glucose': 'low | medium_low | medium',
439                 'BloodPressure': 'medium | medium_high | high',
440                 'SkinThickness': 'low | medium_low | medium',
441                 'Insulin': 'low | medium',
442                 'BMI': 'underweight | healthy_weight',
443                 'DiabetesPedigreeFunction': 'low | medium',
444                 'Age': 'medium | high',
445                 'Outcome': 'low'})
446 # Trojki Pregnancies i SkinThickness (4) 52
447 fuzzy.add_rule({'Pregnancies': 'medium | high',
448                 'Glucose': 'low | medium_low | medium',
449                 'BloodPressure': 'low | medium_low | medium',
450                 'SkinThickness': 'medium | medium_high | high',
451                 'Insulin': 'medium | high',
452                 'BMI': 'underweight | healthy_weight',
453                 'DiabetesPedigreeFunction': 'low | medium',
454                 'Age': 'low | medium',

```

```

455         'Outcome': 'low'})
456 fuzzy.add_rule({'Pregnancies': 'medium | high',
457                 'Glucose': 'low | medium_low | medium',
458                 'BloodPressure': 'low | medium_low | medium',
459                 'SkinThickness': 'medium | medium_high | high',
460                 'Insulin': 'low | medium',
461                 'BMI': 'overweight | obese',
462                 'DiabetesPedigreeFunction': 'low | medium',
463                 'Age': 'low | medium',
464                 'Outcome': 'low'})
465 fuzzy.add_rule({'Pregnancies': 'medium | high',
466                 'Glucose': 'low | medium_low | medium',
467                 'BloodPressure': 'low | medium_low | medium',
468                 'SkinThickness': 'medium | medium_high | high',
469                 'Insulin': 'low | medium',
470                 'BMI': 'underweight | healthy_weight',
471                 'DiabetesPedigreeFunction': 'medium | high',
472                 'Age': 'low | medium',
473                 'Outcome': 'low'})
474 fuzzy.add_rule({'Pregnancies': 'medium | high',
475                 'Glucose': 'low | medium_low | medium',
476                 'BloodPressure': 'low | medium_low | medium',
477                 'SkinThickness': 'medium | medium_high | high',
478                 'Insulin': 'low | medium',
479                 'BMI': 'underweight | healthy_weight',
480                 'DiabetesPedigreeFunction': 'low | medium',
481                 'Age': 'medium | high',
482                 'Outcome': 'low'})
483 # Trojki Pregnancies i Insulin (3) 55
484 fuzzy.add_rule({'Pregnancies': 'medium | high',
485                 'Glucose': 'low | medium_low | medium',
486                 'BloodPressure': 'low | medium_low | medium',
487                 'SkinThickness': 'low | medium_low | medium',
488                 'Insulin': 'medium | high',
489                 'BMI': 'overweight | obese',
490                 'DiabetesPedigreeFunction': 'low | medium',
491                 'Age': 'low | medium',
492                 'Outcome': 'low'})
493 fuzzy.add_rule({'Pregnancies': 'medium | high',
494                 'Glucose': 'low | medium_low | medium',
495                 'BloodPressure': 'low | medium_low | medium',
496                 'SkinThickness': 'low | medium_low | medium',
497                 'Insulin': 'medium | high',
498                 'BMI': 'underweight | healthy_weight',
499                 'DiabetesPedigreeFunction': 'medium | high',
500                 'Age': 'low | medium',
501                 'Outcome': 'low'})
502 fuzzy.add_rule({'Pregnancies': 'medium | high',
503                 'Glucose': 'low | medium_low | medium',
504                 'BloodPressure': 'low | medium_low | medium',
505                 'SkinThickness': 'low | medium_low | medium',
506                 'Insulin': 'medium | high',
507                 'BMI': 'underweight | healthy_weight',
508                 'DiabetesPedigreeFunction': 'low | medium',
509                 'Age': 'medium | high',

```



```

510         'Outcome': 'low'})
511 # Trojki Pregnancies i BMI (2) 57
512 fuzzy.add_rule({'Pregnancies': 'medium | high',
513                'Glucose': 'low | medium_low | medium',
514                'BloodPressure': 'low | medium_low | medium',
515                'SkinThickness': 'low | medium_low | medium',
516                'Insulin': 'low | medium',
517                'BMI': 'obese | overweight',
518                'DiabetesPedigreeFunction': 'medium | high',
519                'Age': 'low | medium',
520                'Outcome': 'low'})
521 fuzzy.add_rule({'Pregnancies': 'medium | high',
522                'Glucose': 'low | medium_low | medium',
523                'BloodPressure': 'low | medium_low | medium',
524                'SkinThickness': 'low | medium_low | medium',
525                'Insulin': 'low | medium',
526                'BMI': 'obese | overweight',
527                'DiabetesPedigreeFunction': 'low | medium',
528                'Age': 'medium | high',
529                'Outcome': 'low'})
530 # Trojki Pregnancies i DiabetesPedigreeFunction (1) 58
531 fuzzy.add_rule({'Pregnancies': 'medium | high',
532                'Glucose': 'low | medium_low | medium',
533                'BloodPressure': 'low | medium_low | medium',
534                'SkinThickness': 'low | medium_low | medium',
535                'Insulin': 'low | medium',
536                'BMI': 'underweight | healthy_weight',
537                'DiabetesPedigreeFunction': 'medium | high',
538                'Age': 'medium | high',
539                'Outcome': 'low'})
540 # Trojki Glucose i BloodPressure (5) 63
541 fuzzy.add_rule({'Pregnancies': 'low | medium',
542                'Glucose': 'medium | medium_high | high',
543                'BloodPressure': 'medium | medium_high | high',
544                'SkinThickness': 'medium | medium_high | high',
545                'Insulin': 'low | medium',
546                'BMI': 'underweight | healthy_weight',
547                'DiabetesPedigreeFunction': 'low | medium',
548                'Age': 'low | medium',
549                'Outcome': 'low'})
550 fuzzy.add_rule({'Pregnancies': 'low | medium',
551                'Glucose': 'medium | medium_high | high',
552                'BloodPressure': 'medium | medium_high | high',
553                'SkinThickness': 'low | medium_low | medium',
554                'Insulin': 'medium | high',
555                'BMI': 'underweight | healthy_weight',
556                'DiabetesPedigreeFunction': 'low | medium',
557                'Age': 'low | medium',
558                'Outcome': 'low'})
559 fuzzy.add_rule({'Pregnancies': 'low | medium',
560                'Glucose': 'medium | medium_high | high',
561                'BloodPressure': 'medium | medium_high | high',
562                'SkinThickness': 'low | medium_low | medium',
563                'Insulin': 'low | medium',
564                'BMI': 'overweight | obese',

```

```

565         'DiabetesPedigreeFunction': 'low | medium',
566         'Age': 'low | medium',
567         'Outcome': 'low'})
568 fuzzy.add_rule({'Pregnancies': 'low | medium',
569                 'Glucose': 'medium | medium_high | high',
570                 'BloodPressure': 'medium | medium_high | high',
571                 'SkinThickness': 'low | medium_low | medium',
572                 'Insulin': 'low | medium',
573                 'BMI': 'underweight | healthy_weight',
574                 'DiabetesPedigreeFunction': 'medium | high',
575                 'Age': 'low | medium',
576                 'Outcome': 'low'})
577 fuzzy.add_rule({'Pregnancies': 'low | medium',
578                 'Glucose': 'medium | medium_high | high',
579                 'BloodPressure': 'medium | medium_high | high',
580                 'SkinThickness': 'low | medium_low | medium',
581                 'Insulin': 'low | medium',
582                 'BMI': 'underweight | healthy_weight',
583                 'DiabetesPedigreeFunction': 'low | medium',
584                 'Age': 'medium | high',
585                 'Outcome': 'low'})
586 # Trojki Glucose i SkinThickness (4) 67
587 fuzzy.add_rule({'Pregnancies': 'low | medium',
588                 'Glucose': 'medium | medium_high | high',
589                 'BloodPressure': 'low | medium_low | medium',
590                 'SkinThickness': 'medium | medium_high | high',
591                 'Insulin': 'medium | high',
592                 'BMI': 'underweight | healthy_weight',
593                 'DiabetesPedigreeFunction': 'low | medium',
594                 'Age': 'low | medium',
595                 'Outcome': 'low'})
596 fuzzy.add_rule({'Pregnancies': 'low | medium',
597                 'Glucose': 'medium | medium_high | high',
598                 'BloodPressure': 'low | medium_low | medium',
599                 'SkinThickness': 'medium | medium_high | high',
600                 'Insulin': 'low | medium',
601                 'BMI': 'overweight | obese',
602                 'DiabetesPedigreeFunction': 'low | medium',
603                 'Age': 'low | medium',
604                 'Outcome': 'low'})
605 fuzzy.add_rule({'Pregnancies': 'low | medium',
606                 'Glucose': 'medium | medium_high | high',
607                 'BloodPressure': 'low | medium_low | medium',
608                 'SkinThickness': 'medium | medium_high | high',
609                 'Insulin': 'low | medium',
610                 'BMI': 'underweight | healthy_weight',
611                 'DiabetesPedigreeFunction': 'medium | high',
612                 'Age': 'low | medium',
613                 'Outcome': 'low'})
614 fuzzy.add_rule({'Pregnancies': 'low | medium',
615                 'Glucose': 'medium | medium_high | high',
616                 'BloodPressure': 'low | medium_low | medium',
617                 'SkinThickness': 'medium | medium_high | high',
618                 'Insulin': 'low | medium',
619                 'BMI': 'underweight | healthy_weight',

```

```

620         'DiabetesPedigreeFunction': 'low | medium',
621         'Age': 'medium | high',
622         'Outcome': 'low'})
623 # Trojki Glucose i Insulin (3) 70
624 fuzzy.add_rule({'Pregnancies': 'low | medium',
625                 'Glucose': 'medium | medium_high | high',
626                 'BloodPressure': 'low | medium_low | medium',
627                 'SkinThickness': 'low | medium_low | medium',
628                 'Insulin': 'medium | high',
629                 'BMI': 'overweight | obese',
630                 'DiabetesPedigreeFunction': 'low | medium',
631                 'Age': 'low | medium',
632                 'Outcome': 'low'})
633 fuzzy.add_rule({'Pregnancies': 'low | medium',
634                 'Glucose': 'medium | medium_high | high',
635                 'BloodPressure': 'low | medium_low | medium',
636                 'SkinThickness': 'low | medium_low | medium',
637                 'Insulin': 'medium | high',
638                 'BMI': 'underweight | healthy_weight',
639                 'DiabetesPedigreeFunction': 'medium | high',
640                 'Age': 'low | medium',
641                 'Outcome': 'low'})
642 fuzzy.add_rule({'Pregnancies': 'low | medium',
643                 'Glucose': 'medium | medium_high | high',
644                 'BloodPressure': 'low | medium_low | medium',
645                 'SkinThickness': 'low | medium_low | medium',
646                 'Insulin': 'medium | high',
647                 'BMI': 'underweight | healthy_weight',
648                 'DiabetesPedigreeFunction': 'low | medium',
649                 'Age': 'medium | high',
650                 'Outcome': 'low'})
651 # Trojki Glucose i BMI (2) 72
652 fuzzy.add_rule({'Pregnancies': 'low | medium',
653                 'Glucose': 'medium | medium_high | high',
654                 'BloodPressure': 'low | medium_low | medium',
655                 'SkinThickness': 'low | medium_low | medium',
656                 'Insulin': 'low | medium',
657                 'BMI': 'obese | overweight',
658                 'DiabetesPedigreeFunction': 'medium | high',
659                 'Age': 'low | medium',
660                 'Outcome': 'low'})
661 fuzzy.add_rule({'Pregnancies': 'low | medium',
662                 'Glucose': 'medium | medium_high | high',
663                 'BloodPressure': 'low | medium_low | medium',
664                 'SkinThickness': 'low | medium_low | medium',
665                 'Insulin': 'low | medium',
666                 'BMI': 'obese | overweight',
667                 'DiabetesPedigreeFunction': 'low | medium',
668                 'Age': 'medium | high',
669                 'Outcome': 'low'})
670 # Trojki Glucose i DiabetesPedigreeFunction (1) 73
671 fuzzy.add_rule({'Pregnancies': 'low | medium',
672                 'Glucose': 'medium | medium_high | high',
673                 'BloodPressure': 'low | medium_low | medium',
674                 'SkinThickness': 'low | medium_low | medium',

```

```

675         'Insulin': 'low | medium',
676         'BMI': 'underweight | healthy_weight',
677         'DiabetesPedigreeFunction': 'medium | high',
678         'Age': 'medium | high',
679         'Outcome': 'low'})
680 # Trojki BloodPressure i SkinThickness (4) 77
681 fuzzy.add_rule({'Pregnancies': 'low | medium',
682                 'Glucose': 'low | medium_low | medium',
683                 'BloodPressure': 'medium | medium_high | high',
684                 'SkinThickness': 'medium | medium_high | high',
685                 'Insulin': 'medium | high',
686                 'BMI': 'underweight | healthy_weight',
687                 'DiabetesPedigreeFunction': 'low | medium',
688                 'Age': 'low | medium',
689                 'Outcome': 'low'})
690 fuzzy.add_rule({'Pregnancies': 'low | medium',
691                 'Glucose': 'low | medium_low | medium',
692                 'BloodPressure': 'medium | medium_high | high',
693                 'SkinThickness': 'medium | medium_high | high',
694                 'Insulin': 'low | medium',
695                 'BMI': 'overweight | obese',
696                 'DiabetesPedigreeFunction': 'low | medium',
697                 'Age': 'low | medium',
698                 'Outcome': 'low'})
699 fuzzy.add_rule({'Pregnancies': 'low | medium',
700                 'Glucose': 'low | medium_low | medium',
701                 'BloodPressure': 'medium | medium_high | high',
702                 'SkinThickness': 'medium | medium_high | high',
703                 'Insulin': 'low | medium',
704                 'BMI': 'underweight | healthy_weight',
705                 'DiabetesPedigreeFunction': 'medium | high',
706                 'Age': 'low | medium',
707                 'Outcome': 'low'})
708 fuzzy.add_rule({'Pregnancies': 'low | medium',
709                 'Glucose': 'low | medium_low | medium',
710                 'BloodPressure': 'medium | medium_high | high',
711                 'SkinThickness': 'medium | medium_high | high',
712                 'Insulin': 'low | medium',
713                 'BMI': 'underweight | healthy_weight',
714                 'DiabetesPedigreeFunction': 'low | medium',
715                 'Age': 'medium | high',
716                 'Outcome': 'low'})
717 # Trojki BloodPressure i Insulin (3) 80
718 fuzzy.add_rule({'Pregnancies': 'low | medium',
719                 'Glucose': 'low | medium_low | medium',
720                 'BloodPressure': 'medium | medium_high | high',
721                 'SkinThickness': 'low | medium_low | medium',
722                 'Insulin': 'medium | high',
723                 'BMI': 'overweight | obese',
724                 'DiabetesPedigreeFunction': 'low | medium',
725                 'Age': 'low | medium',
726                 'Outcome': 'low'})
727 fuzzy.add_rule({'Pregnancies': 'low | medium',
728                 'Glucose': 'low | medium_low | medium',
729                 'BloodPressure': 'medium | medium_high | high',

```

```

730         'SkinThickness': 'low | medium_low | medium',
731         'Insulin': 'medium | high',
732         'BMI': 'underweight | healthy_weight',
733         'DiabetesPedigreeFunction': 'medium | high',
734         'Age': 'low | medium',
735         'Outcome': 'low'})
736 fuzzy.add_rule({'Pregnancies': 'low | medium',
737                 'Glucose': 'low | medium_low | medium',
738                 'BloodPressure': 'medium | medium_high | high',
739                 'SkinThickness': 'low | medium_low | medium',
740                 'Insulin': 'medium | high',
741                 'BMI': 'underweight | healthy_weight',
742                 'DiabetesPedigreeFunction': 'low | medium',
743                 'Age': 'medium | high',
744                 'Outcome': 'low'})
745 # Trojki BloodPressure i BMI (2) 82
746 fuzzy.add_rule({'Pregnancies': 'low | medium',
747                 'Glucose': 'low | medium_low | medium',
748                 'BloodPressure': 'medium | medium_high | high',
749                 'SkinThickness': 'low | medium_low | medium',
750                 'Insulin': 'low | medium',
751                 'BMI': 'obese | overweight',
752                 'DiabetesPedigreeFunction': 'medium | high',
753                 'Age': 'low | medium',
754                 'Outcome': 'low'})
755 fuzzy.add_rule({'Pregnancies': 'low | medium',
756                 'Glucose': 'low | medium_low | medium',
757                 'BloodPressure': 'medium | medium_high | high',
758                 'SkinThickness': 'low | medium_low | medium',
759                 'Insulin': 'low | medium',
760                 'BMI': 'obese | overweight',
761                 'DiabetesPedigreeFunction': 'low | medium',
762                 'Age': 'medium | high',
763                 'Outcome': 'low'})
764 # Trojki BloodPressure i DiabetesPedigreeFunction (1) 83
765 fuzzy.add_rule({'Pregnancies': 'low | medium',
766                 'Glucose': 'low | medium_low | medium',
767                 'BloodPressure': 'medium | medium_high | high',
768                 'SkinThickness': 'low | medium_low | medium',
769                 'Insulin': 'low | medium',
770                 'BMI': 'underweight | healthy_weight',
771                 'DiabetesPedigreeFunction': 'medium | high',
772                 'Age': 'medium | high',
773                 'Outcome': 'low'})
774 # Trojki SkinThickness i Insulin (3) 86
775 fuzzy.add_rule({'Pregnancies': 'low | medium',
776                 'Glucose': 'low | medium_low | medium',
777                 'BloodPressure': 'low | medium_low | medium',
778                 'SkinThickness': 'medium | medium_high | high',
779                 'Insulin': 'medium | high',
780                 'BMI': 'overweight | obese',
781                 'DiabetesPedigreeFunction': 'low | medium',
782                 'Age': 'low | medium',
783                 'Outcome': 'low'})
784 fuzzy.add_rule({'Pregnancies': 'low | medium',

```

```

785         'Glucose': 'low | medium_low | medium',
786         'BloodPressure': 'low | medium_low | medium',
787         'SkinThickness': 'medium | medium_high | high',
788         'Insulin': 'medium | high',
789         'BMI': 'underweight | healthy_weight',
790         'DiabetesPedigreeFunction': 'medium | high',
791         'Age': 'low | medium',
792         'Outcome': 'low'})
793 fuzzy.add_rule({'Pregnancies': 'low | medium',
794                 'Glucose': 'low | medium_low | medium',
795                 'BloodPressure': 'low | medium_low | medium',
796                 'SkinThickness': 'medium | medium_high | high',
797                 'Insulin': 'medium | high',
798                 'BMI': 'underweight | healthy_weight',
799                 'DiabetesPedigreeFunction': 'low | medium',
800                 'Age': 'medium | high',
801                 'Outcome': 'low'})
802 # Trojki SkinThickness i BMI (2) 88
803 fuzzy.add_rule({'Pregnancies': 'low | medium',
804                 'Glucose': 'low | medium_low | medium',
805                 'BloodPressure': 'low | medium_low | medium',
806                 'SkinThickness': 'medium | medium_high | high',
807                 'Insulin': 'low | medium',
808                 'BMI': 'obese | overweight',
809                 'DiabetesPedigreeFunction': 'medium | high',
810                 'Age': 'low | medium',
811                 'Outcome': 'low'})
812 fuzzy.add_rule({'Pregnancies': 'low | medium',
813                 'Glucose': 'low | medium_low | medium',
814                 'BloodPressure': 'low | medium_low | medium',
815                 'SkinThickness': 'medium | medium_high | high',
816                 'Insulin': 'low | medium',
817                 'BMI': 'obese | overweight',
818                 'DiabetesPedigreeFunction': 'low | medium',
819                 'Age': 'medium | high',
820                 'Outcome': 'low'})
821 # Trojki SkinThickness i DiabetesPedigreeFunction (1) 89
822 fuzzy.add_rule({'Pregnancies': 'low | medium',
823                 'Glucose': 'low | medium_low | medium',
824                 'BloodPressure': 'low | medium_low | medium',
825                 'SkinThickness': 'medium | medium_high | high',
826                 'Insulin': 'low | medium',
827                 'BMI': 'underweight | healthy_weight',
828                 'DiabetesPedigreeFunction': 'medium | high',
829                 'Age': 'medium | high',
830                 'Outcome': 'low'})
831 # Trojki Insulin i BMI (2) 91
832 fuzzy.add_rule({'Pregnancies': 'low | medium',
833                 'Glucose': 'low | medium_low | medium',
834                 'BloodPressure': 'low | medium_low | medium',
835                 'SkinThickness': 'low | medium_low | medium',
836                 'Insulin': 'medium | high',
837                 'BMI': 'obese | overweight',
838                 'DiabetesPedigreeFunction': 'medium | high',
839                 'Age': 'low | medium',

```

```

840         'Outcome': 'low'})
841 fuzzy.add_rule({'Pregnancies': 'low | medium',
842               'Glucose': 'low | medium_low | medium',
843               'BloodPressure': 'low | medium_low | medium',
844               'SkinThickness': 'low | medium_low | medium',
845               'Insulin': 'medium | high',
846               'BMI': 'obese | overweight',
847               'DiabetesPedigreeFunction': 'low | medium',
848               'Age': 'medium | high',
849               'Outcome': 'low'})
850 # Trojki Insulin i DiabetesPedigreeFunction (1) 92
851 fuzzy.add_rule({'Pregnancies': 'low | medium',
852               'Glucose': 'low | medium_low | medium',
853               'BloodPressure': 'low | medium_low | medium',
854               'SkinThickness': 'low | medium_low | medium',
855               'Insulin': 'medium | high',
856               'BMI': 'underweight | healthy_weight',
857               'DiabetesPedigreeFunction': 'medium | high',
858               'Age': 'low | medium',
859               'Outcome': 'low'})
860 # Trojki BMI i DiabetesPedigreeFunction (1) 93
861 fuzzy.add_rule({'Pregnancies': 'low | medium',
862               'Glucose': 'low | medium_low | medium',
863               'BloodPressure': 'low | medium_low | medium',
864               'SkinThickness': 'low | medium_low | medium',
865               'Insulin': 'low | medium',
866               'BMI': 'obese | overweight',
867               'DiabetesPedigreeFunction': 'medium | high',
868               'Age': 'medium | high',
869               'Outcome': 'low'})
870
871 # 4 wysokie, 4 niskie
872 # Czworkki Pregnancies i Glucose i BloodPressure (5) 98
873 fuzzy.add_rule({'Pregnancies': 'medium | high',
874               'Glucose': 'medium | medium_high | high',
875               'BloodPressure': 'medium | medium_high | high',
876               'SkinThickness': 'medium | medium_high | high',
877               'Insulin': 'low | medium',
878               'BMI': 'underweight | healthy_weight',
879               'DiabetesPedigreeFunction': 'low | medium',
880               'Age': 'low | medium',
881               'Outcome': 'high'})
882 fuzzy.add_rule({'Pregnancies': 'medium | high',
883               'Glucose': 'medium | medium_high | high',
884               'BloodPressure': 'medium | medium_high | high',
885               'SkinThickness': 'low | medium_low | medium',
886               'Insulin': 'medium | high',
887               'BMI': 'underweight | healthy_weight',
888               'DiabetesPedigreeFunction': 'low | medium',
889               'Age': 'low | medium',
890               'Outcome': 'high'})
891 fuzzy.add_rule({'Pregnancies': 'medium | high',
892               'Glucose': 'medium | medium_high | high',
893               'BloodPressure': 'medium | medium_high | high',
894               'SkinThickness': 'low | medium_low | medium',

```

```

895         'Insulin': 'low | medium',
896         'BMI': 'overweight | obese',
897         'DiabetesPedigreeFunction': 'low | medium',
898         'Age': 'low | medium',
899         'Outcome': 'high'})
900 fuzzy.add_rule({'Pregnancies': 'medium | high',
901                 'Glucose': 'medium | medium_high | high',
902                 'BloodPressure': 'medium | medium_high | high',
903                 'SkinThickness': 'low | medium_low | medium',
904                 'Insulin': 'low | medium',
905                 'BMI': 'underweight | healthy_weight',
906                 'DiabetesPedigreeFunction': 'medium | high',
907                 'Age': 'low | medium',
908                 'Outcome': 'high'})
909 fuzzy.add_rule({'Pregnancies': 'medium | high',
910                 'Glucose': 'medium | medium_high | high',
911                 'BloodPressure': 'medium | medium_high | high',
912                 'SkinThickness': 'low | medium_low | medium',
913                 'Insulin': 'low | medium',
914                 'BMI': 'underweight | healthy_weight',
915                 'DiabetesPedigreeFunction': 'low | medium',
916                 'Age': 'medium | high',
917                 'Outcome': 'high'})
918 # Czworki Pregnancies i Glucose i SkinThickness (4) 102
919 fuzzy.add_rule({'Pregnancies': 'medium | high',
920                 'Glucose': 'medium | medium_high | high',
921                 'BloodPressure': 'low | medium_low | medium',
922                 'SkinThickness': 'medium | medium_high | high',
923                 'Insulin': 'medium | high',
924                 'BMI': 'underweight | healthy_weight',
925                 'DiabetesPedigreeFunction': 'low | medium',
926                 'Age': 'low | medium',
927                 'Outcome': 'high'})
928 fuzzy.add_rule({'Pregnancies': 'medium | high',
929                 'Glucose': 'medium | medium_high | high',
930                 'BloodPressure': 'low | medium_low | medium',
931                 'SkinThickness': 'medium | medium_high | high',
932                 'Insulin': 'low | medium',
933                 'BMI': 'overweight | obese',
934                 'DiabetesPedigreeFunction': 'low | medium',
935                 'Age': 'low | medium',
936                 'Outcome': 'high'})
937 fuzzy.add_rule({'Pregnancies': 'medium | high',
938                 'Glucose': 'medium | medium_high | high',
939                 'BloodPressure': 'low | medium_low | medium',
940                 'SkinThickness': 'medium | medium_high | high',
941                 'Insulin': 'low | medium',
942                 'BMI': 'underweight | healthy_weight',
943                 'DiabetesPedigreeFunction': 'medium | high',
944                 'Age': 'low | medium',
945                 'Outcome': 'high'})
946 fuzzy.add_rule({'Pregnancies': 'medium | high',
947                 'Glucose': 'medium | medium_high | high',
948                 'BloodPressure': 'low | medium_low | medium',
949                 'SkinThickness': 'medium | medium_high | high',

```



```

950         'Insulin': 'low | medium',
951         'BMI': 'underweight | healthy_weight',
952         'DiabetesPedigreeFunction': 'low | medium',
953         'Age': 'medium | high',
954         'Outcome': 'high'})
955 # Czworki Pregnancies i Glucose i Insulin (3) 105
956 fuzzy.add_rule({'Pregnancies': 'medium | high',
957                 'Glucose': 'medium | medium_high | high',
958                 'BloodPressure': 'low | medium_low | medium',
959                 'SkinThickness': 'low | medium_low | medium',
960                 'Insulin': 'medium | high',
961                 'BMI': 'overweight | obese',
962                 'DiabetesPedigreeFunction': 'low | medium',
963                 'Age': 'low | medium',
964                 'Outcome': 'high'})
965 fuzzy.add_rule({'Pregnancies': 'medium | high',
966                 'Glucose': 'medium | medium_high | high',
967                 'BloodPressure': 'low | medium_low | medium',
968                 'SkinThickness': 'low | medium_low | medium',
969                 'Insulin': 'medium | high',
970                 'BMI': 'underweight | healthy_weight',
971                 'DiabetesPedigreeFunction': 'medium | high',
972                 'Age': 'low | medium',
973                 'Outcome': 'high'})
974 fuzzy.add_rule({'Pregnancies': 'medium | high',
975                 'Glucose': 'medium | medium_high | high',
976                 'BloodPressure': 'low | medium_low | medium',
977                 'SkinThickness': 'low | medium_low | medium',
978                 'Insulin': 'medium | high',
979                 'BMI': 'underweight | healthy_weight',
980                 'DiabetesPedigreeFunction': 'low | medium',
981                 'Age': 'medium | high',
982                 'Outcome': 'high'})
983 # Czworki Pregnancies i Glucose i BMI (2) 107
984 fuzzy.add_rule({'Pregnancies': 'medium | high',
985                 'Glucose': 'medium | medium_high | high',
986                 'BloodPressure': 'low | medium_low | medium',
987                 'SkinThickness': 'low | medium_low | medium',
988                 'Insulin': 'low | medium',
989                 'BMI': 'overweight | obese',
990                 'DiabetesPedigreeFunction': 'medium | high',
991                 'Age': 'low | medium',
992                 'Outcome': 'high'})
993 fuzzy.add_rule({'Pregnancies': 'medium | high',
994                 'Glucose': 'medium | medium_high | high',
995                 'BloodPressure': 'low | medium_low | medium',
996                 'SkinThickness': 'low | medium_low | medium',
997                 'Insulin': 'low | medium',
998                 'BMI': 'overweight | obese',
999                 'DiabetesPedigreeFunction': 'low | medium',
1000                 'Age': 'medium | high',
1001                 'Outcome': 'high'})
1002 # Czworki Pregnancies i Glucose i DiabetesPedigreeFunction (1)
1003     10
1004 fuzzy.add_rule({'Pregnancies': 'medium | high',

```

```

1004         'Glucose': 'medium | medium_high | high',
1005         'BloodPressure': 'low | medium_low | medium',
1006         'SkinThickness': 'low | medium_low | medium',
1007         'Insulin': 'low | medium',
1008         'BMI': 'underweight | healthy_weight',
1009         'DiabetesPedigreeFunction': 'medium | high',
1010         'Age': 'medium | high',
1011         'Outcome': 'high'})
1012 # Czworki Pregnancies i BloodPressure i SkinThickness (4) 112
1013 fuzzy.add_rule({'Pregnancies': 'medium | high',
1014                 'Glucose': 'low | medium_low | medium',
1015                 'BloodPressure': 'medium | medium_high | high',
1016                 'SkinThickness': 'medium | medium_high | high',
1017                 'Insulin': 'medium | high',
1018                 'BMI': 'underweight | healthy_weight',
1019                 'DiabetesPedigreeFunction': 'low | medium',
1020                 'Age': 'low | medium',
1021                 'Outcome': 'high'})
1022 fuzzy.add_rule({'Pregnancies': 'medium | high',
1023                 'Glucose': 'low | medium_low | medium',
1024                 'BloodPressure': 'medium | medium_high | high',
1025                 'SkinThickness': 'medium | medium_high | high',
1026                 'Insulin': 'low | medium',
1027                 'BMI': 'underweight | healthy_weight',
1028                 'DiabetesPedigreeFunction': 'low | medium',
1029                 'Age': 'low | medium',
1030                 'Outcome': 'high'})
1031 fuzzy.add_rule({'Pregnancies': 'medium | high',
1032                 'Glucose': 'low | medium_low | medium',
1033                 'BloodPressure': 'medium | medium_high | high',
1034                 'SkinThickness': 'medium | medium_high | high',
1035                 'Insulin': 'low | medium',
1036                 'BMI': 'underweight | healthy_weight',
1037                 'DiabetesPedigreeFunction': 'medium | high',
1038                 'Age': 'low | medium',
1039                 'Outcome': 'high'})
1040 fuzzy.add_rule({'Pregnancies': 'medium | high',
1041                 'Glucose': 'low | medium_low | medium',
1042                 'BloodPressure': 'medium | medium_high | high',
1043                 'SkinThickness': 'medium | medium_high | high',
1044                 'Insulin': 'low | medium',
1045                 'BMI': 'underweight | healthy_weight',
1046                 'DiabetesPedigreeFunction': 'low | medium',
1047                 'Age': 'medium | high',
1048                 'Outcome': 'high'})
1049 # Czworki Pregnancies i BloodPressure i Insulin (3) 115
1050 fuzzy.add_rule({'Pregnancies': 'medium | high',
1051                 'Glucose': 'low | medium_low | medium',
1052                 'BloodPressure': 'medium | medium_high | high',
1053                 'SkinThickness': 'low | medium_low | medium',
1054                 'Insulin': 'medium | high',
1055                 'BMI': 'overweight | obese',
1056                 'DiabetesPedigreeFunction': 'low | medium',
1057                 'Age': 'low | medium',
1058                 'Outcome': 'high'})

```

```

1059     fuzzy.add_rule({'Pregnancies': 'medium | high',
1060                    'Glucose': 'low | medium_low | medium',
1061                    'BloodPressure': 'medium | medium_high | high',
1062                    'SkinThickness': 'low | medium_low | medium',
1063                    'Insulin': 'medium | high',
1064                    'BMI': 'underweight | healthy_weight',
1065                    'DiabetesPedigreeFunction': 'medium | high',
1066                    'Age': 'low | medium',
1067                    'Outcome': 'high'})
1068     fuzzy.add_rule({'Pregnancies': 'medium | high',
1069                    'Glucose': 'low | medium_low | medium',
1070                    'BloodPressure': 'medium | medium_high | high',
1071                    'SkinThickness': 'low | medium_low | medium',
1072                    'Insulin': 'medium | high',
1073                    'BMI': 'underweight | healthy_weight',
1074                    'DiabetesPedigreeFunction': 'low | medium',
1075                    'Age': 'medium | high',
1076                    'Outcome': 'high'})
1077     # Czworki Pregnancies i BloodPressure i BMI (2) 117
1078     fuzzy.add_rule({'Pregnancies': 'medium | high',
1079                    'Glucose': 'low | medium_low | medium',
1080                    'BloodPressure': 'medium | medium_high | high',
1081                    'SkinThickness': 'low | medium_low | medium',
1082                    'Insulin': 'low | medium',
1083                    'BMI': 'overweight | obese',
1084                    'DiabetesPedigreeFunction': 'medium | high',
1085                    'Age': 'low | medium',
1086                    'Outcome': 'high'})
1087     fuzzy.add_rule({'Pregnancies': 'medium | high',
1088                    'Glucose': 'low | medium_low | medium',
1089                    'BloodPressure': 'medium | medium_high | high',
1090                    'SkinThickness': 'low | medium_low | medium',
1091                    'Insulin': 'low | medium',
1092                    'BMI': 'overweight | obese',
1093                    'DiabetesPedigreeFunction': 'low | medium',
1094                    'Age': 'medium | high',
1095                    'Outcome': 'high'})
1096     # Czworki Pregnancies i BloodPressure i DiabetesPedigreeFunction
1097     (1) 118
1098     fuzzy.add_rule({'Pregnancies': 'medium | high',
1099                    'Glucose': 'low | medium_low | medium',
1100                    'BloodPressure': 'medium | medium_high | high',
1101                    'SkinThickness': 'low | medium_low | medium',
1102                    'Insulin': 'low | medium',
1103                    'BMI': 'underweight | healthy_weight',
1104                    'DiabetesPedigreeFunction': 'medium | high',
1105                    'Age': 'medium | high',
1106                    'Outcome': 'high'})
1107     # Czworki Pregnancies i SkinThickness i Insulin (3) 121
1108     fuzzy.add_rule({'Pregnancies': 'medium | high',
1109                    'Glucose': 'low | medium_low | medium',
1110                    'BloodPressure': 'low | medium_low | medium',
1111                    'SkinThickness': 'medium | medium_high | high',
1112                    'Insulin': 'medium | high',
1113                    'BMI': 'overweight | obese',

```

```

1113         'DiabetesPedigreeFunction': 'low | medium',
1114         'Age': 'low | medium',
1115         'Outcome': 'high'})
1116 fuzzy.add_rule({'Pregnancies': 'medium | high',
1117                 'Glucose': 'low | medium_low | medium',
1118                 'BloodPressure': 'low | medium_low | medium',
1119                 'SkinThickness': 'medium | medium_high | high',
1120                 'Insulin': 'medium | high',
1121                 'BMI': 'underweight | healthy_weight',
1122                 'DiabetesPedigreeFunction': 'medium | high',
1123                 'Age': 'low | medium',
1124                 'Outcome': 'high'})
1125 fuzzy.add_rule({'Pregnancies': 'medium | high',
1126                 'Glucose': 'low | medium_low | medium',
1127                 'BloodPressure': 'low | medium_low | medium',
1128                 'SkinThickness': 'medium | medium_high | high',
1129                 'Insulin': 'medium | high',
1130                 'BMI': 'underweight | healthy_weight',
1131                 'DiabetesPedigreeFunction': 'low | medium',
1132                 'Age': 'medium | high',
1133                 'Outcome': 'high'})
1134 # Czworki Pregnancies i SkinThickness i BMI (2) 123
1135 fuzzy.add_rule({'Pregnancies': 'medium | high',
1136                 'Glucose': 'low | medium_low | medium',
1137                 'BloodPressure': 'low | medium_low | medium',
1138                 'SkinThickness': 'medium | medium_high | high',
1139                 'Insulin': 'low | medium',
1140                 'BMI': 'overweight | obese',
1141                 'DiabetesPedigreeFunction': 'medium | high',
1142                 'Age': 'low | medium',
1143                 'Outcome': 'high'})
1144 fuzzy.add_rule({'Pregnancies': 'medium | high',
1145                 'Glucose': 'low | medium_low | medium',
1146                 'BloodPressure': 'low | medium_low | medium',
1147                 'SkinThickness': 'medium | medium_high | high',
1148                 'Insulin': 'low | medium',
1149                 'BMI': 'overweight | obese',
1150                 'DiabetesPedigreeFunction': 'low | medium',
1151                 'Age': 'medium | high',
1152                 'Outcome': 'high'})
1153 # Czworki Pregnancies i SkinThickness i DiabetesPedigreeFunction
1154         (1) 124
1155 fuzzy.add_rule({'Pregnancies': 'medium | high',
1156                 'Glucose': 'low | medium_low | medium',
1157                 'BloodPressure': 'low | medium_low | medium',
1158                 'SkinThickness': 'medium | medium_high | high',
1159                 'Insulin': 'low | medium',
1160                 'BMI': 'underweight | healthy_weight',
1161                 'DiabetesPedigreeFunction': 'medium | high',
1162                 'Age': 'medium | high',
1163                 'Outcome': 'high'})
1164 # Czworki Pregnancies i Insulin i BMI (2) 126
1165 fuzzy.add_rule({'Pregnancies': 'low | medium',
1166                 'Glucose': 'low | medium_low | medium',
1167                 'BloodPressure': 'medium | medium_high | high',

```

```

1167         'SkinThickness': 'low | medium_low | medium',
1168         'Insulin': 'medium | high',
1169         'BMI': 'overweight | obese',
1170         'DiabetesPedigreeFunction': 'medium | high',
1171         'Age': 'low | medium',
1172         'Outcome': 'high'})
1173 fuzzy.add_rule({'Pregnancies': 'low | medium',
1174               'Glucose': 'low | medium_low | medium',
1175               'BloodPressure': 'medium | medium_high | high',
1176               'SkinThickness': 'low | medium_low | medium',
1177               'Insulin': 'medium | high',
1178               'BMI': 'overweight | obese',
1179               'DiabetesPedigreeFunction': 'low | medium',
1180               'Age': 'medium | high',
1181               'Outcome': 'high'})
1182 # Czworki Pregnancies i Insulin i DiabetesPedigreeFunction (1)
1183     127
1184 fuzzy.add_rule({'Pregnancies': 'low | medium',
1185               'Glucose': 'low | medium_low | medium',
1186               'BloodPressure': 'medium | medium_high | high',
1187               'SkinThickness': 'low | medium_low | medium',
1188               'Insulin': 'medium | high',
1189               'BMI': 'underweight | healthy_weight',
1190               'DiabetesPedigreeFunction': 'medium | high',
1191               'Age': 'medium | high',
1192               'Outcome': 'high'})
1193 # Czworki Pregnancies i BMI i DiabetesPedigreeFunction (1) 128
1194 fuzzy.add_rule({'Pregnancies': 'medium | high',
1195               'Glucose': 'low | medium_low | medium',
1196               'BloodPressure': 'low | medium_low | medium',
1197               'SkinThickness': 'low | medium_low | medium',
1198               'Insulin': 'low | medium',
1199               'BMI': 'obese | overweight',
1200               'DiabetesPedigreeFunction': 'medium | high',
1201               'Age': 'medium | high',
1202               'Outcome': 'high'})
1203 # Czworki Glucose i BloodPressure i SkinThickness (4) 132
1204 fuzzy.add_rule({'Pregnancies': 'low | medium',
1205               'Glucose': 'medium | medium_high | high',
1206               'BloodPressure': 'medium | medium_high | high',
1207               'SkinThickness': 'medium | medium_high | high',
1208               'Insulin': 'medium | high',
1209               'BMI': 'underweight | healthy_weight',
1210               'DiabetesPedigreeFunction': 'low | medium',
1211               'Age': 'low | medium',
1212               'Outcome': 'high'})
1213 fuzzy.add_rule({'Pregnancies': 'low | medium',
1214               'Glucose': 'medium | medium_high | high',
1215               'BloodPressure': 'medium | medium_high | high',
1216               'SkinThickness': 'medium | medium_high | high',
1217               'Insulin': 'low | medium',
1218               'BMI': 'overweight | obese',
1219               'DiabetesPedigreeFunction': 'low | medium',
1220               'Age': 'low | medium',

```

```

1221         'Outcome': 'high'})
1222 fuzzy.add_rule({'Pregnancies': 'low | medium',
1223                'Glucose': 'medium | medium_high | high',
1224                'BloodPressure': 'medium | medium_high | high',
1225                'SkinThickness': 'medium | medium_high | high',
1226                'Insulin': 'low | medium',
1227                'BMI': 'underweight | healthy_weight',
1228                'DiabetesPedigreeFunction': 'medium | high',
1229                'Age': 'low | medium',
1230                'Outcome': 'high'})
1231 fuzzy.add_rule({'Pregnancies': 'low | medium',
1232                'Glucose': 'medium | medium_high | high',
1233                'BloodPressure': 'medium | medium_high | high',
1234                'SkinThickness': 'medium | medium_high | high',
1235                'Insulin': 'low | medium',
1236                'BMI': 'underweight | healthy_weight',
1237                'DiabetesPedigreeFunction': 'low | medium',
1238                'Age': 'medium | high',
1239                'Outcome': 'high'})
1240 # Czworki Glucose i BloodPressure i Insulin (3) 135
1241 fuzzy.add_rule({'Pregnancies': 'low | medium',
1242                'Glucose': 'medium | medium_high | high',
1243                'BloodPressure': 'medium | medium_high | high',
1244                'SkinThickness': 'low | medium_low | medium',
1245                'Insulin': 'medium | high',
1246                'BMI': 'overweight | obese',
1247                'DiabetesPedigreeFunction': 'low | medium',
1248                'Age': 'low | medium',
1249                'Outcome': 'high'})
1250 fuzzy.add_rule({'Pregnancies': 'low | medium',
1251                'Glucose': 'medium | medium_high | high',
1252                'BloodPressure': 'medium | medium_high | high',
1253                'SkinThickness': 'low | medium_low | medium',
1254                'Insulin': 'medium | high',
1255                'BMI': 'underweight | healthy_weight',
1256                'DiabetesPedigreeFunction': 'medium | high',
1257                'Age': 'low | medium',
1258                'Outcome': 'high'})
1259 fuzzy.add_rule({'Pregnancies': 'low | medium',
1260                'Glucose': 'medium | medium_high | high',
1261                'BloodPressure': 'medium | medium_high | high',
1262                'SkinThickness': 'low | medium_low | medium',
1263                'Insulin': 'medium | high',
1264                'BMI': 'underweight | healthy_weight',
1265                'DiabetesPedigreeFunction': 'low | medium',
1266                'Age': 'medium | high',
1267                'Outcome': 'high'})
1268 # Czworki Glucose i BloodPressure i BMI (2) 137
1269 fuzzy.add_rule({'Pregnancies': 'low | medium',
1270                'Glucose': 'medium | medium_high | high',
1271                'BloodPressure': 'medium | medium_high | high',
1272                'SkinThickness': 'low | medium_low | medium',
1273                'Insulin': 'low | medium',
1274                'BMI': 'overweight | obese',
1275                'DiabetesPedigreeFunction': 'medium | high',

```

```

1276         'Age': 'low | medium',
1277         'Outcome': 'high'})
1278 fuzzy.add_rule({'Pregnancies': 'low | medium',
1279                'Glucose': 'medium | medium_high | high',
1280                'BloodPressure': 'medium | medium_high | high',
1281                'SkinThickness': 'low | medium_low | medium',
1282                'Insulin': 'low | medium',
1283                'BMI': 'overweight | obese',
1284                'DiabetesPedigreeFunction': 'low | medium',
1285                'Age': 'medium | high',
1286                'Outcome': 'high'})
1287 # Czworki Glucose i BloodPressure i DiabetesPedigreeFunction (1)
1288     138
1289 fuzzy.add_rule({'Pregnancies': 'low | medium',
1290                'Glucose': 'medium | medium_high | high',
1291                'BloodPressure': 'medium | medium_high | high',
1292                'SkinThickness': 'low | medium_low | medium',
1293                'Insulin': 'low | medium',
1294                'BMI': 'underweight | healthy_weight',
1295                'DiabetesPedigreeFunction': 'medium | high',
1296                'Age': 'medium | high',
1297                'Outcome': 'high'})
1298 # Czworki Glucose i SkinThickness i Insulin (3) 141
1299 fuzzy.add_rule({'Pregnancies': 'low | medium',
1300                'Glucose': 'medium | medium_high | high',
1301                'BloodPressure': 'low | medium_low | medium',
1302                'SkinThickness': 'medium | medium_high | high',
1303                'Insulin': 'medium | high',
1304                'BMI': 'overweight | obese',
1305                'DiabetesPedigreeFunction': 'low | medium',
1306                'Age': 'low | medium',
1307                'Outcome': 'high'})
1308 fuzzy.add_rule({'Pregnancies': 'low | medium',
1309                'Glucose': 'medium | medium_high | high',
1310                'BloodPressure': 'low | medium_low | medium',
1311                'SkinThickness': 'medium | medium_high | high',
1312                'Insulin': 'medium | high',
1313                'BMI': 'underweight | healthy_weight',
1314                'DiabetesPedigreeFunction': 'medium | high',
1315                'Age': 'low | medium',
1316                'Outcome': 'high'})
1317 fuzzy.add_rule({'Pregnancies': 'low | medium',
1318                'Glucose': 'medium | medium_high | high',
1319                'BloodPressure': 'low | medium_low | medium',
1320                'SkinThickness': 'medium | medium_high | high',
1321                'Insulin': 'medium | high',
1322                'BMI': 'underweight | healthy_weight',
1323                'DiabetesPedigreeFunction': 'low | medium',
1324                'Age': 'medium | high',
1325                'Outcome': 'high'})
1326 # Czworki Glucose i SkinThickness i BMI (2) 143
1327 fuzzy.add_rule({'Pregnancies': 'low | medium',
1328                'Glucose': 'medium | medium_high | high',
1329                'BloodPressure': 'low | medium_low | medium',
1330                'SkinThickness': 'medium | medium_high | high',

```



```

1330         'Insulin': 'low | medium',
1331         'BMI': 'overweight | obese',
1332         'DiabetesPedigreeFunction': 'medium | high',
1333         'Age': 'low | medium',
1334         'Outcome': 'high'})
1335 fuzzy.add_rule({'Pregnancies': 'low | medium',
1336                 'Glucose': 'medium | medium_high | high',
1337                 'BloodPressure': 'low | medium_low | medium',
1338                 'SkinThickness': 'medium | medium_high | high',
1339                 'Insulin': 'low | medium',
1340                 'BMI': 'overweight | obese',
1341                 'DiabetesPedigreeFunction': 'low | medium',
1342                 'Age': 'medium | high',
1343                 'Outcome': 'high'})
1344 # Czworki Glucose i SkinThickness i DiabetesPedigreeFunction (1)
1345     144
1346 fuzzy.add_rule({'Pregnancies': 'low | medium',
1347                 'Glucose': 'medium | medium_high | high',
1348                 'BloodPressure': 'low | medium_low | medium',
1349                 'SkinThickness': 'medium | medium_high | high',
1350                 'Insulin': 'low | medium',
1351                 'BMI': 'underweight | healthy_weight',
1352                 'DiabetesPedigreeFunction': 'medium | high',
1353                 'Age': 'medium | high',
1354                 'Outcome': 'high'})
1355 # Czworki Glucose i Insulin i BMI (2) 146
1356 fuzzy.add_rule({'Pregnancies': 'low | medium',
1357                 'Glucose': 'medium | medium_high | high',
1358                 'BloodPressure': 'low | medium_low | medium',
1359                 'SkinThickness': 'low | medium_low | medium',
1360                 'Insulin': 'medium | high',
1361                 'BMI': 'overweight | obese',
1362                 'DiabetesPedigreeFunction': 'medium | high',
1363                 'Age': 'low | medium',
1364                 'Outcome': 'high'})
1365 fuzzy.add_rule({'Pregnancies': 'low | medium',
1366                 'Glucose': 'medium | medium_high | high',
1367                 'BloodPressure': 'low | medium_low | medium',
1368                 'SkinThickness': 'low | medium_low | medium',
1369                 'Insulin': 'medium | high',
1370                 'BMI': 'overweight | obese',
1371                 'DiabetesPedigreeFunction': 'low | medium',
1372                 'Age': 'medium | high',
1373                 'Outcome': 'high'})
1374 # Czworki Glucose i Insulin i DiabetesPedigreeFunction (1) 147
1375 fuzzy.add_rule({'Pregnancies': 'low | medium',
1376                 'Glucose': 'medium | medium_high | high',
1377                 'BloodPressure': 'low | medium_low | medium',
1378                 'SkinThickness': 'low | medium_low | medium',
1379                 'Insulin': 'medium | high',
1380                 'BMI': 'underweight | healthy_weight',
1381                 'DiabetesPedigreeFunction': 'medium | high',
1382                 'Age': 'medium | high',
1383                 'Outcome': 'high'})
1384 # Czworki Glucose i BMI i DiabetesPedigreeFunction (1) 148

```



```

1384     fuzzy.add_rule({'Pregnancies': 'low | medium',
1385                    'Glucose': 'medium | medium_high | high',
1386                    'BloodPressure': 'low | medium_low | medium',
1387                    'SkinThickness': 'low | medium_low | medium',
1388                    'Insulin': 'low | medium',
1389                    'BMI': 'obese | overweight',
1390                    'DiabetesPedigreeFunction': 'medium | high',
1391                    'Age': 'medium | high',
1392                    'Outcome': 'high'})
1393
1394     # Czworki BloodPressure i SkinThickness i Insulin (3) 151
1395     fuzzy.add_rule({'Pregnancies': 'low | medium',
1396                    'Glucose': 'low | medium_low | medium',
1397                    'BloodPressure': 'medium | medium_high | high',
1398                    'SkinThickness': 'medium | medium_high | high',
1399                    'Insulin': 'medium | high',
1400                    'BMI': 'overweight | obese',
1401                    'DiabetesPedigreeFunction': 'low | medium',
1402                    'Age': 'low | medium',
1403                    'Outcome': 'high'})
1404     fuzzy.add_rule({'Pregnancies': 'low | medium',
1405                    'Glucose': 'low | medium_low | medium',
1406                    'BloodPressure': 'medium | medium_high | high',
1407                    'SkinThickness': 'medium | medium_high | high',
1408                    'Insulin': 'medium | high',
1409                    'BMI': 'underweight | healthy_weight',
1410                    'DiabetesPedigreeFunction': 'medium | high',
1411                    'Age': 'low | medium',
1412                    'Outcome': 'high'})
1413     fuzzy.add_rule({'Pregnancies': 'low | medium',
1414                    'Glucose': 'low | medium_low | medium',
1415                    'BloodPressure': 'medium | medium_high | high',
1416                    'SkinThickness': 'medium | medium_high | high',
1417                    'Insulin': 'medium | high',
1418                    'BMI': 'underweight | healthy_weight',
1419                    'DiabetesPedigreeFunction': 'low | medium',
1420                    'Age': 'medium | high',
1421                    'Outcome': 'high'})
1422     # Czworki BloodPressure i SkinThickness i BMI (2) 153
1423     fuzzy.add_rule({'Pregnancies': 'low | medium',
1424                    'Glucose': 'low | medium_low | medium',
1425                    'BloodPressure': 'medium | medium_high | high',
1426                    'SkinThickness': 'medium | medium_high | high',
1427                    'Insulin': 'low | medium',
1428                    'BMI': 'overweight | obese',
1429                    'DiabetesPedigreeFunction': 'medium | high',
1430                    'Age': 'low | medium',
1431                    'Outcome': 'high'})
1432     fuzzy.add_rule({'Pregnancies': 'low | medium',
1433                    'Glucose': 'low | medium_low | medium',
1434                    'BloodPressure': 'medium | medium_high | high',
1435                    'SkinThickness': 'medium | medium_high | high',
1436                    'Insulin': 'low | medium',
1437                    'BMI': 'overweight | obese',
1438                    'DiabetesPedigreeFunction': 'low | medium',

```

```

1439         'Age': 'medium | high',
1440         'Outcome': 'high'})
1441 # Czworkki BloodPressure i SkinThickness i
1442     DiabetesPedigreeFunction (1) 154
1443 fuzzy.add_rule({'Pregnancies': 'low | medium',
1444                 'Glucose': 'low | medium_low | medium',
1445                 'BloodPressure': 'medium | medium_high | high',
1446                 'SkinThickness': 'medium | medium_high | high',
1447                 'Insulin': 'low | medium',
1448                 'BMI': 'underweight | healthy_weight',
1449                 'DiabetesPedigreeFunction': 'medium | high',
1450                 'Age': 'medium | high',
1451                 'Outcome': 'high'})
1452 # Czworkki BloodPressure i Insulin i BMI (2) 156
1453 fuzzy.add_rule({'Pregnancies': 'low | medium',
1454                 'Glucose': 'low | medium_low | medium',
1455                 'BloodPressure': 'medium | medium_high | high',
1456                 'SkinThickness': 'low | medium_low | medium',
1457                 'Insulin': 'medium | high',
1458                 'BMI': 'overweight | obese',
1459                 'DiabetesPedigreeFunction': 'medium | high',
1460                 'Age': 'low | medium',
1461                 'Outcome': 'high'})
1462 fuzzy.add_rule({'Pregnancies': 'low | medium',
1463                 'Glucose': 'low | medium_low | medium',
1464                 'BloodPressure': 'medium | medium_high | high',
1465                 'SkinThickness': 'low | medium_low | medium',
1466                 'Insulin': 'medium | high',
1467                 'BMI': 'overweight | obese',
1468                 'DiabetesPedigreeFunction': 'low | medium',
1469                 'Age': 'medium | high',
1470                 'Outcome': 'high'})
1471 # Czworkki BloodPressure i Insulin i DiabetesPedigreeFunction (1)
1472     157
1473 fuzzy.add_rule({'Pregnancies': 'low | medium',
1474                 'Glucose': 'low | medium_low | medium',
1475                 'BloodPressure': 'medium | medium_high | high',
1476                 'SkinThickness': 'low | medium_low | medium',
1477                 'Insulin': 'medium | high',
1478                 'BMI': 'underweight | healthy_weight',
1479                 'DiabetesPedigreeFunction': 'medium | high',
1480                 'Age': 'medium | high',
1481                 'Outcome': 'high'})
1482 # Czworkki BloodPressure i BMI i DiabetesPedigreeFunction (1) 158
1483 fuzzy.add_rule({'Pregnancies': 'low | medium',
1484                 'Glucose': 'low | medium_low | medium',
1485                 'BloodPressure': 'medium | medium_high | high',
1486                 'SkinThickness': 'low | medium_low | medium',
1487                 'Insulin': 'low | medium',
1488                 'BMI': 'obese | overweight',
1489                 'DiabetesPedigreeFunction': 'medium | high',
1490                 'Age': 'medium | high',
1491                 'Outcome': 'high'})
1492 # Czworkki SkinThickness i Insulin i BMI (2) 160

```

```

1492     fuzzy.add_rule({'Pregnancies': 'low | medium',
1493                    'Glucose': 'low | medium_low | medium',
1494                    'BloodPressure': 'low | medium_low | medium',
1495                    'SkinThickness': 'medium | medium_high | high',
1496                    'Insulin': 'medium | high',
1497                    'BMI': 'overweight | obese',
1498                    'DiabetesPedigreeFunction': 'medium | high',
1499                    'Age': 'low | medium',
1500                    'Outcome': 'high'})
1501     fuzzy.add_rule({'Pregnancies': 'low | medium',
1502                    'Glucose': 'low | medium_low | medium',
1503                    'BloodPressure': 'low | medium_low | medium',
1504                    'SkinThickness': 'medium | medium_high | high',
1505                    'Insulin': 'medium | high',
1506                    'BMI': 'overweight | obese',
1507                    'DiabetesPedigreeFunction': 'low | medium',
1508                    'Age': 'medium | high',
1509                    'Outcome': 'high'})
1510     # Czworki SkinThickness i Insulin i DiabetesPedigreeFunction (1)
1511     161
1512     fuzzy.add_rule({'Pregnancies': 'low | medium',
1513                    'Glucose': 'low | medium_low | medium',
1514                    'BloodPressure': 'low | medium_low | medium',
1515                    'SkinThickness': 'medium | medium_high | high',
1516                    'Insulin': 'medium | high',
1517                    'BMI': 'underweight | healthy_weight',
1518                    'DiabetesPedigreeFunction': 'medium | high',
1519                    'Age': 'medium | high',
1520                    'Outcome': 'high'})
1521     # Czworki SkinThickness i BMI i DiabetesPedigreeFunction (1) 162
1522     fuzzy.add_rule({'Pregnancies': 'low | medium',
1523                    'Glucose': 'low | medium_low | medium',
1524                    'BloodPressure': 'low | medium_low | medium',
1525                    'SkinThickness': 'medium | medium_high | high',
1526                    'Insulin': 'low | medium',
1527                    'BMI': 'obese | overweight',
1528                    'DiabetesPedigreeFunction': 'medium | high',
1529                    'Age': 'medium | high',
1530                    'Outcome': 'high'})
1531     # Czworki Insulin i BMI i DiabetesPedigreeFunction (1) 163
1532     fuzzy.add_rule({'Pregnancies': 'medium | high',
1533                    'Glucose': 'medium | medium_high | high',
1534                    'BloodPressure': 'low | medium_low | medium',
1535                    'SkinThickness': 'low | medium_low | medium',
1536                    'Insulin': 'low | medium',
1537                    'BMI': 'overweight | obese',
1538                    'DiabetesPedigreeFunction': 'medium | high',
1539                    'Age': 'medium | high',
1540                    'Outcome': 'high'})
1541
1542
1543     # Wiekszosc niskich
1544     # Same niskie (1) 164
1545     fuzzy.add_rule({'Pregnancies': 'medium | high',

```

```

1546         'Glucose': 'medium | medium_high | high',
1547         'BloodPressure': 'medium | medium_high | high',
1548         'SkinThickness': 'medium | medium_high | high',
1549         'Insulin': 'medium | high',
1550         'BMI': 'overweight | obese',
1551         'DiabetesPedigreeFunction': 'medium | high',
1552         'Age': 'medium | high',
1553         'Outcome': 'high'})
1554
1555 # Pojedyncze niskie (8) 172
1556 fuzzy.add_rule({'Pregnancies': 'low | medium',
1557                 'Glucose': 'medium | medium_high | high',
1558                 'BloodPressure': 'medium | medium_high | high',
1559                 'SkinThickness': 'medium | medium_high | high',
1560                 'Insulin': 'medium | high',
1561                 'BMI': 'overweight | obese',
1562                 'DiabetesPedigreeFunction': 'medium | high',
1563                 'Age': 'medium | high',
1564                 'Outcome': 'high'})
1565
1566 fuzzy.add_rule({'Pregnancies': 'medium | high',
1567                 'Glucose': 'low | medium_low | medium',
1568                 'BloodPressure': 'medium | medium_high | high',
1569                 'SkinThickness': 'medium | medium_high | high',
1570                 'Insulin': 'medium | high',
1571                 'BMI': 'overweight | obese',
1572                 'DiabetesPedigreeFunction': 'medium | high',
1573                 'Age': 'medium | high',
1574                 'Outcome': 'high'})
1575
1576 fuzzy.add_rule({'Pregnancies': 'medium | high',
1577                 'Glucose': 'medium | medium_high | high',
1578                 'BloodPressure': 'low | medium_low | medium',
1579                 'SkinThickness': 'medium | medium_high | high',
1580                 'Insulin': 'medium | high',
1581                 'BMI': 'overweight | obese',
1582                 'DiabetesPedigreeFunction': 'medium | high',
1583                 'Age': 'medium | high',
1584                 'Outcome': 'high'})
1585
1586 fuzzy.add_rule({'Pregnancies': 'medium | high',
1587                 'Glucose': 'medium | medium_high | high',
1588                 'BloodPressure': 'medium | medium_high | high',
1589                 'SkinThickness': 'low | medium_low | medium',
1590                 'Insulin': 'medium | high',
1591                 'BMI': 'overweight | obese',
1592                 'DiabetesPedigreeFunction': 'medium | high',
1593                 'Age': 'medium | high',
1594                 'Outcome': 'high'})
1595
1596 fuzzy.add_rule({'Pregnancies': 'medium | high',
1597                 'Glucose': 'medium | medium_high | high',
1598                 'BloodPressure': 'medium | medium_high | high',
1599                 'SkinThickness': 'medium | medium_high | high',
1600                 'Insulin': 'low | medium',
1601                 'BMI': 'overweight | obese',
1602                 'DiabetesPedigreeFunction': 'medium | high',
1603                 'Age': 'medium | high',
1604                 'Outcome': 'high'})
1605
1606 fuzzy.add_rule({'Pregnancies': 'medium | high',

```

```

1601         'Glucose': 'medium | medium_high | high',
1602         'BloodPressure': 'medium | medium_high | high',
1603         'SkinThickness': 'medium | medium_high | high',
1604         'Insulin': 'medium | high',
1605         'BMI': 'obese | overweight',
1606         'DiabetesPedigreeFunction': 'medium | high',
1607         'Age': 'medium | high',
1608         'Outcome': 'high'})
1609 fuzzy.add_rule({'Pregnancies': 'medium | high',
1610                'Glucose': 'medium | medium_high | high',
1611                'BloodPressure': 'medium | medium_high | high',
1612                'SkinThickness': 'medium | medium_high | high',
1613                'Insulin': 'medium | high',
1614                'BMI': 'overweight | obese',
1615                'DiabetesPedigreeFunction': 'low | medium',
1616                'Age': 'medium | high',
1617                'Outcome': 'high'})
1618 fuzzy.add_rule({'Pregnancies': 'medium | high',
1619                'Glucose': 'medium | medium_high | high',
1620                'BloodPressure': 'medium | medium_high | high',
1621                'SkinThickness': 'medium | medium_high | high',
1622                'Insulin': 'medium | high',
1623                'BMI': 'overweight | obese',
1624                'DiabetesPedigreeFunction': 'medium | high',
1625                'Age': 'low | medium',
1626                'Outcome': 'high'})
1627 # Pary Pregnancies (7) 179
1628 fuzzy.add_rule({'Pregnancies': 'low | medium',
1629                'Glucose': 'low | medium_low | medium',
1630                'BloodPressure': 'medium | medium_high | high',
1631                'SkinThickness': 'medium | medium_high | high',
1632                'Insulin': 'medium | high',
1633                'BMI': 'overweight | obese',
1634                'DiabetesPedigreeFunction': 'medium | high',
1635                'Age': 'medium | high',
1636                'Outcome': 'high'})
1637 fuzzy.add_rule({'Pregnancies': 'low | medium',
1638                'Glucose': 'medium | medium_high | high',
1639                'BloodPressure': 'low | medium_low | medium',
1640                'SkinThickness': 'medium | medium_high | high',
1641                'Insulin': 'medium | high',
1642                'BMI': 'overweight | obese',
1643                'DiabetesPedigreeFunction': 'medium | high',
1644                'Age': 'medium | high',
1645                'Outcome': 'high'})
1646 fuzzy.add_rule({'Pregnancies': 'low | medium',
1647                'Glucose': 'medium | medium_high | high',
1648                'BloodPressure': 'medium | medium_high | high',
1649                'SkinThickness': 'low | medium_low | medium',
1650                'Insulin': 'medium | high',
1651                'BMI': 'overweight | obese',
1652                'DiabetesPedigreeFunction': 'medium | high',
1653                'Age': 'medium | high',
1654                'Outcome': 'high'})
1655 fuzzy.add_rule({'Pregnancies': 'low | medium',

```

```

1656         'Glucose': 'medium | medium_high | high',
1657         'BloodPressure': 'medium | medium_high | high',
1658         'SkinThickness': 'medium | medium_high | high',
1659         'Insulin': 'low | medium',
1660         'BMI': 'overweight | obese',
1661         'DiabetesPedigreeFunction': 'medium | high',
1662         'Age': 'medium | high',
1663         'Outcome': 'high'})
1664 fuzzy.add_rule({'Pregnancies': 'low | medium',
1665                 'Glucose': 'medium | medium_high | high',
1666                 'BloodPressure': 'medium | medium_high | high',
1667                 'SkinThickness': 'medium | medium_high | high',
1668                 'Insulin': 'medium | high',
1669                 'BMI': 'obese | overweight',
1670                 'DiabetesPedigreeFunction': 'medium | high',
1671                 'Age': 'medium | high',
1672                 'Outcome': 'high'})
1673 fuzzy.add_rule({'Pregnancies': 'low | medium',
1674                 'Glucose': 'medium | medium_high | high',
1675                 'BloodPressure': 'medium | medium_high | high',
1676                 'SkinThickness': 'medium | medium_high | high',
1677                 'Insulin': 'medium | high',
1678                 'BMI': 'overweight | obese',
1679                 'DiabetesPedigreeFunction': 'low | medium',
1680                 'Age': 'medium | high',
1681                 'Outcome': 'high'})
1682 fuzzy.add_rule({'Pregnancies': 'low | medium',
1683                 'Glucose': 'medium | medium_high | high',
1684                 'BloodPressure': 'medium | medium_high | high',
1685                 'SkinThickness': 'medium | medium_high | high',
1686                 'Insulin': 'medium | high',
1687                 'BMI': 'overweight | obese',
1688                 'DiabetesPedigreeFunction': 'medium | high',
1689                 'Age': 'low | medium',
1690                 'Outcome': 'high'})
1691 # Pary Glucose (6) 185
1692 fuzzy.add_rule({'Pregnancies': 'medium | high',
1693                 'Glucose': 'low | medium_low | medium',
1694                 'BloodPressure': 'low | medium_low | medium',
1695                 'SkinThickness': 'medium | medium_high | high',
1696                 'Insulin': 'medium | high',
1697                 'BMI': 'overweight | obese',
1698                 'DiabetesPedigreeFunction': 'medium | high',
1699                 'Age': 'medium | high',
1700                 'Outcome': 'high'})
1701 fuzzy.add_rule({'Pregnancies': 'medium | high',
1702                 'Glucose': 'low | medium_low | medium',
1703                 'BloodPressure': 'medium | medium_high | high',
1704                 'SkinThickness': 'low | medium_low | medium',
1705                 'Insulin': 'medium | high',
1706                 'BMI': 'overweight | obese',
1707                 'DiabetesPedigreeFunction': 'medium | high',
1708                 'Age': 'medium | high',
1709                 'Outcome': 'high'})
1710 fuzzy.add_rule({'Pregnancies': 'medium | high',

```

```

1711         'Glucose': 'low | medium_low | medium',
1712         'BloodPressure': 'medium | medium_high | high',
1713         'SkinThickness': 'medium | medium_high | high',
1714         'Insulin': 'low | medium',
1715         'BMI': 'overweight | obese',
1716         'DiabetesPedigreeFunction': 'medium | high',
1717         'Age': 'medium | high',
1718         'Outcome': 'high'})
1719 fuzzy.add_rule({'Pregnancies': 'medium | high',
1720                'Glucose': 'low | medium_low | medium',
1721                'BloodPressure': 'medium | medium_high | high',
1722                'SkinThickness': 'medium | medium_high | high',
1723                'Insulin': 'medium | high',
1724                'BMI': 'obese | overweight',
1725                'DiabetesPedigreeFunction': 'medium | high',
1726                'Age': 'medium | high',
1727                'Outcome': 'high'})
1728 fuzzy.add_rule({'Pregnancies': 'medium | high',
1729                'Glucose': 'low | medium_low | medium',
1730                'BloodPressure': 'medium | medium_high | high',
1731                'SkinThickness': 'medium | medium_high | high',
1732                'Insulin': 'medium | high',
1733                'BMI': 'overweight | obese',
1734                'DiabetesPedigreeFunction': 'low | medium',
1735                'Age': 'medium | high',
1736                'Outcome': 'high'})
1737 fuzzy.add_rule({'Pregnancies': 'medium | high',
1738                'Glucose': 'low | medium_low | medium',
1739                'BloodPressure': 'medium | medium_high | high',
1740                'SkinThickness': 'medium | medium_high | high',
1741                'Insulin': 'medium | high',
1742                'BMI': 'overweight | obese',
1743                'DiabetesPedigreeFunction': 'medium | high',
1744                'Age': 'low | medium',
1745                'Outcome': 'high'})
1746 # Pary BloodPressure (5) 190
1747 fuzzy.add_rule({'Pregnancies': 'medium | high',
1748                'Glucose': 'medium | medium_high | high',
1749                'BloodPressure': 'low | medium_low | medium',
1750                'SkinThickness': 'low | medium_low | medium',
1751                'Insulin': 'medium | high',
1752                'BMI': 'overweight | obese',
1753                'DiabetesPedigreeFunction': 'medium | high',
1754                'Age': 'medium | high',
1755                'Outcome': 'high'})
1756 fuzzy.add_rule({'Pregnancies': 'medium | high',
1757                'Glucose': 'medium | medium_high | high',
1758                'BloodPressure': 'low | medium_low | medium',
1759                'SkinThickness': 'medium | medium_high | high',
1760                'Insulin': 'low | medium',
1761                'BMI': 'overweight | obese',
1762                'DiabetesPedigreeFunction': 'medium | high',
1763                'Age': 'medium | high',
1764                'Outcome': 'high'})
1765 fuzzy.add_rule({'Pregnancies': 'medium | high',

```

```

1766         'Glucose': 'medium | medium_high | high',
1767         'BloodPressure': 'low | medium_low | medium',
1768         'SkinThickness': 'medium | medium_high | high',
1769         'Insulin': 'medium | high',
1770         'BMI': 'obese | overweight',
1771         'DiabetesPedigreeFunction': 'medium | high',
1772         'Age': 'medium | high',
1773         'Outcome': 'high'})
1774 fuzzy.add_rule({'Pregnancies': 'medium | high',
1775                 'Glucose': 'medium | medium_high | high',
1776                 'BloodPressure': 'low | medium_low | medium',
1777                 'SkinThickness': 'medium | medium_high | high',
1778                 'Insulin': 'medium | high',
1779                 'BMI': 'overweight | obese',
1780                 'DiabetesPedigreeFunction': 'low | medium',
1781                 'Age': 'medium | high',
1782                 'Outcome': 'high'})
1783 fuzzy.add_rule({'Pregnancies': 'medium | high',
1784                 'Glucose': 'medium | medium_high | high',
1785                 'BloodPressure': 'low | medium_low | medium',
1786                 'SkinThickness': 'medium | medium_high | high',
1787                 'Insulin': 'medium | high',
1788                 'BMI': 'overweight | obese',
1789                 'DiabetesPedigreeFunction': 'medium | high',
1790                 'Age': 'low | medium',
1791                 'Outcome': 'high'})
1792 # Pary SkinThickness (4) 194
1793 fuzzy.add_rule({'Pregnancies': 'medium | high',
1794                 'Glucose': 'medium | medium_high | high',
1795                 'BloodPressure': 'medium | medium_high | high',
1796                 'SkinThickness': 'low | medium_low | medium',
1797                 'Insulin': 'low | medium',
1798                 'BMI': 'overweight | obese',
1799                 'DiabetesPedigreeFunction': 'medium | high',
1800                 'Age': 'medium | high',
1801                 'Outcome': 'high'})
1802 fuzzy.add_rule({'Pregnancies': 'medium | high',
1803                 'Glucose': 'medium | medium_high | high',
1804                 'BloodPressure': 'medium | medium_high | high',
1805                 'SkinThickness': 'low | medium_low | medium',
1806                 'Insulin': 'medium | high',
1807                 'BMI': 'obese | overweight',
1808                 'DiabetesPedigreeFunction': 'medium | high',
1809                 'Age': 'medium | high',
1810                 'Outcome': 'high'})
1811 fuzzy.add_rule({'Pregnancies': 'medium | high',
1812                 'Glucose': 'medium | medium_high | high',
1813                 'BloodPressure': 'medium | medium_high | high',
1814                 'SkinThickness': 'low | medium_low | medium',
1815                 'Insulin': 'medium | high',
1816                 'BMI': 'overweight | obese',
1817                 'DiabetesPedigreeFunction': 'low | medium',
1818                 'Age': 'medium | high',
1819                 'Outcome': 'high'})
1820 fuzzy.add_rule({'Pregnancies': 'medium | high',

```



```

1821         'Glucose': 'medium | medium_high | high',
1822         'BloodPressure': 'medium | medium_high | high',
1823         'SkinThickness': 'low | medium_low | medium',
1824         'Insulin': 'medium | high',
1825         'BMI': 'overweight | obese',
1826         'DiabetesPedigreeFunction': 'medium | high',
1827         'Age': 'low | medium',
1828         'Outcome': 'high'})
1829 # Pary Insulin (3) 197
1830 fuzzy.add_rule({'Pregnancies': 'medium | high',
1831                 'Glucose': 'medium | medium_high | high',
1832                 'BloodPressure': 'medium | medium_high | high',
1833                 'SkinThickness': 'medium | medium_high | high',
1834                 'Insulin': 'low | medium',
1835                 'BMI': 'obese | overweight',
1836                 'DiabetesPedigreeFunction': 'medium | high',
1837                 'Age': 'medium | high',
1838                 'Outcome': 'high'})
1839 fuzzy.add_rule({'Pregnancies': 'medium | high',
1840                 'Glucose': 'medium | medium_high | high',
1841                 'BloodPressure': 'medium | medium_high | high',
1842                 'SkinThickness': 'medium | medium_high | high',
1843                 'Insulin': 'low | medium',
1844                 'BMI': 'overweight | obese',
1845                 'DiabetesPedigreeFunction': 'low | medium',
1846                 'Age': 'medium | high',
1847                 'Outcome': 'high'})
1848 fuzzy.add_rule({'Pregnancies': 'medium | high',
1849                 'Glucose': 'medium | medium_high | high',
1850                 'BloodPressure': 'medium | medium_high | high',
1851                 'SkinThickness': 'medium | medium_high | high',
1852                 'Insulin': 'low | medium',
1853                 'BMI': 'overweight | obese',
1854                 'DiabetesPedigreeFunction': 'medium | high',
1855                 'Age': 'low | medium',
1856                 'Outcome': 'high'})
1857 # Pary BMI (2) 199
1858 fuzzy.add_rule({'Pregnancies': 'medium | high',
1859                 'Glucose': 'medium | medium_high | high',
1860                 'BloodPressure': 'medium | medium_high | high',
1861                 'SkinThickness': 'medium | medium_high | high',
1862                 'Insulin': 'medium | high',
1863                 'BMI': 'obese | overweight',
1864                 'DiabetesPedigreeFunction': 'low | medium',
1865                 'Age': 'medium | high',
1866                 'Outcome': 'high'})
1867 fuzzy.add_rule({'Pregnancies': 'medium | high',
1868                 'Glucose': 'medium | medium_high | high',
1869                 'BloodPressure': 'medium | medium_high | high',
1870                 'SkinThickness': 'medium | medium_high | high',
1871                 'Insulin': 'medium | high',
1872                 'BMI': 'obese | overweight',
1873                 'DiabetesPedigreeFunction': 'medium | high',
1874                 'Age': 'low | medium',
1875                 'Outcome': 'high'})

```

```

1876 # Pary DiabetesPedigreeFunction (1) 200
1877 fuzzy.add_rule({'Pregnancies': 'medium | high',
1878                'Glucose': 'medium | medium_high | high',
1879                'BloodPressure': 'medium | medium_high | high',
1880                'SkinThickness': 'medium | medium_high | high',
1881                'Insulin': 'medium | high',
1882                'BMI': 'overweight | obese',
1883                'DiabetesPedigreeFunction': 'low | medium',
1884                'Age': 'low | medium',
1885                'Outcome': 'high'})
1886 # Trojki Pregnancies i Glucose (6) 206
1887 fuzzy.add_rule({'Pregnancies': 'low | medium',
1888                'Glucose': 'low | medium_low | medium',
1889                'BloodPressure': 'low | medium_low | medium',
1890                'SkinThickness': 'medium | medium_high | high',
1891                'Insulin': 'medium | high',
1892                'BMI': 'overweight | obese',
1893                'DiabetesPedigreeFunction': 'medium | high',
1894                'Age': 'medium | high',
1895                'Outcome': 'high'})
1896 fuzzy.add_rule({'Pregnancies': 'low | medium',
1897                'Glucose': 'low | medium_low | medium',
1898                'BloodPressure': 'medium | medium_high | high',
1899                'SkinThickness': 'low | medium_low | medium',
1900                'Insulin': 'medium | high',
1901                'BMI': 'overweight | obese',
1902                'DiabetesPedigreeFunction': 'medium | high',
1903                'Age': 'medium | high',
1904                'Outcome': 'high'})
1905 fuzzy.add_rule({'Pregnancies': 'low | medium',
1906                'Glucose': 'low | medium_low | medium',
1907                'BloodPressure': 'medium | medium_high | high',
1908                'SkinThickness': 'medium | medium_high | high',
1909                'Insulin': 'low | medium',
1910                'BMI': 'overweight | obese',
1911                'DiabetesPedigreeFunction': 'medium | high',
1912                'Age': 'medium | high',
1913                'Outcome': 'high'})
1914 fuzzy.add_rule({'Pregnancies': 'low | medium',
1915                'Glucose': 'low | medium_low | medium',
1916                'BloodPressure': 'medium | medium_high | high',
1917                'SkinThickness': 'medium | medium_high | high',
1918                'Insulin': 'medium | high',
1919                'BMI': 'overweight | obese',
1920                'DiabetesPedigreeFunction': 'medium | high',
1921                'Age': 'medium | high',
1922                'Outcome': 'high'})
1923 fuzzy.add_rule({'Pregnancies': 'low | medium',
1924                'Glucose': 'low | medium_low | medium',
1925                'BloodPressure': 'medium | medium_high | high',
1926                'SkinThickness': 'medium | medium_high | high',
1927                'Insulin': 'medium | high',
1928                'BMI': 'overweight | obese',
1929                'DiabetesPedigreeFunction': 'low | medium',
1930                'Age': 'medium | high',

```

```

1931         'Outcome': 'high'})
1932 fuzzy.add_rule({'Pregnancies': 'low | medium',
1933                'Glucose': 'low | medium_low | medium',
1934                'BloodPressure': 'medium | medium_high | high',
1935                'SkinThickness': 'medium | medium_high | high',
1936                'Insulin': 'medium | high',
1937                'BMI': 'overweight | obese',
1938                'DiabetesPedigreeFunction': 'medium | high',
1939                'Age': 'low | medium',
1940                'Outcome': 'high'})
1941 # Trojki Pregnancies i BloodPressure (5) 211
1942 fuzzy.add_rule({'Pregnancies': 'low | medium',
1943                'Glucose': 'medium | medium_high | high',
1944                'BloodPressure': 'low | medium_low | medium',
1945                'SkinThickness': 'low | medium_low | medium',
1946                'Insulin': 'medium | high',
1947                'BMI': 'overweight | obese',
1948                'DiabetesPedigreeFunction': 'medium | high',
1949                'Age': 'medium | high',
1950                'Outcome': 'high'})
1951 fuzzy.add_rule({'Pregnancies': 'low | medium',
1952                'Glucose': 'medium | medium_high | high',
1953                'BloodPressure': 'low | medium_low | medium',
1954                'SkinThickness': 'medium | medium_high | high',
1955                'Insulin': 'low | medium',
1956                'BMI': 'overweight | obese',
1957                'DiabetesPedigreeFunction': 'medium | high',
1958                'Age': 'medium | high',
1959                'Outcome': 'high'})
1960 fuzzy.add_rule({'Pregnancies': 'low | medium',
1961                'Glucose': 'medium | medium_high | high',
1962                'BloodPressure': 'low | medium_low | medium',
1963                'SkinThickness': 'medium | medium_high | high',
1964                'Insulin': 'medium | high',
1965                'BMI': 'overweight | obese',
1966                'DiabetesPedigreeFunction': 'medium | high',
1967                'Age': 'medium | high',
1968                'Outcome': 'high'})
1969 fuzzy.add_rule({'Pregnancies': 'low | medium',
1970                'Glucose': 'medium | medium_high | high',
1971                'BloodPressure': 'low | medium_low | medium',
1972                'SkinThickness': 'medium | medium_high | high',
1973                'Insulin': 'medium | high',
1974                'BMI': 'overweight | obese',
1975                'DiabetesPedigreeFunction': 'low | medium',
1976                'Age': 'medium | high',
1977                'Outcome': 'high'})
1978 fuzzy.add_rule({'Pregnancies': 'low | medium',
1979                'Glucose': 'medium | medium_high | high',
1980                'BloodPressure': 'low | medium_low | medium',
1981                'SkinThickness': 'medium | medium_high | high',
1982                'Insulin': 'medium | high',
1983                'BMI': 'overweight | obese',
1984                'DiabetesPedigreeFunction': 'medium | high',
1985                'Age': 'low | medium',

```

```

1986         'Outcome': 'high'})
1987 # Trojki Pregnancies i SkinThickness (4) 215
1988 fuzzy.add_rule({'Pregnancies': 'low | medium',
1989                'Glucose': 'medium | medium_high | high',
1990                'BloodPressure': 'medium | medium_high | high',
1991                'SkinThickness': 'low | medium_low | medium',
1992                'Insulin': 'low | medium',
1993                'BMI': 'overweight | obese',
1994                'DiabetesPedigreeFunction': 'medium | high',
1995                'Age': 'medium | high',
1996                'Outcome': 'high'})
1997 fuzzy.add_rule({'Pregnancies': 'low | medium',
1998                'Glucose': 'medium | medium_high | high',
1999                'BloodPressure': 'medium | medium_high | high',
2000                'SkinThickness': 'low | medium_low | medium',
2001                'Insulin': 'medium | high',
2002                'BMI': 'overweight | obese',
2003                'DiabetesPedigreeFunction': 'medium | high',
2004                'Age': 'medium | high',
2005                'Outcome': 'high'})
2006 fuzzy.add_rule({'Pregnancies': 'low | medium',
2007                'Glucose': 'medium | medium_high | high',
2008                'BloodPressure': 'medium | medium_high | high',
2009                'SkinThickness': 'low | medium_low | medium',
2010                'Insulin': 'medium | high',
2011                'BMI': 'overweight | obese',
2012                'DiabetesPedigreeFunction': 'low | medium',
2013                'Age': 'medium | high',
2014                'Outcome': 'high'})
2015 fuzzy.add_rule({'Pregnancies': 'low | medium',
2016                'Glucose': 'medium | medium_high | high',
2017                'BloodPressure': 'medium | medium_high | high',
2018                'SkinThickness': 'low | medium_low | medium',
2019                'Insulin': 'medium | high',
2020                'BMI': 'overweight | obese',
2021                'DiabetesPedigreeFunction': 'medium | high',
2022                'Age': 'low | medium',
2023                'Outcome': 'high'})
2024 # Trojki Pregnancies i Insulin (3) 218
2025 fuzzy.add_rule({'Pregnancies': 'low | medium',
2026                'Glucose': 'medium | medium_high | high',
2027                'BloodPressure': 'medium | medium_high | high',
2028                'SkinThickness': 'medium | medium_high | high',
2029                'Insulin': 'low | medium',
2030                'BMI': 'overweight | obese',
2031                'DiabetesPedigreeFunction': 'medium | high',
2032                'Age': 'medium | high',
2033                'Outcome': 'high'})
2034 fuzzy.add_rule({'Pregnancies': 'low | medium',
2035                'Glucose': 'medium | medium_high | high',
2036                'BloodPressure': 'medium | medium_high | high',
2037                'SkinThickness': 'medium | medium_high | high',
2038                'Insulin': 'low | medium',
2039                'BMI': 'overweight | obese',
2040                'DiabetesPedigreeFunction': 'low | medium',

```

```

2041         'Age': 'medium | high',
2042         'Outcome': 'high'})
2043 fuzzy.add_rule({'Pregnancies': 'low | medium',
2044                 'Glucose': 'medium | medium_high | high',
2045                 'BloodPressure': 'medium | medium_high | high',
2046                 'SkinThickness': 'medium | medium_high | high',
2047                 'Insulin': 'low | medium',
2048                 'BMI': 'overweight | obese',
2049                 'DiabetesPedigreeFunction': 'medium | high',
2050                 'Age': 'low | medium',
2051                 'Outcome': 'high'})
2052 # Trojki Pregnancies i BMI (2) 220
2053 fuzzy.add_rule({'Pregnancies': 'low | medium',
2054                 'Glucose': 'medium | medium_high | high',
2055                 'BloodPressure': 'medium | medium_high | high',
2056                 'SkinThickness': 'medium | medium_high | high',
2057                 'Insulin': 'medium | high',
2058                 'BMI': 'obese | overweight',
2059                 'DiabetesPedigreeFunction': 'low | medium',
2060                 'Age': 'medium | high',
2061                 'Outcome': 'high'})
2062 fuzzy.add_rule({'Pregnancies': 'low | medium',
2063                 'Glucose': 'medium | medium_high | high',
2064                 'BloodPressure': 'medium | medium_high | high',
2065                 'SkinThickness': 'medium | medium_high | high',
2066                 'Insulin': 'medium | high',
2067                 'BMI': 'obese | overweight',
2068                 'DiabetesPedigreeFunction': 'medium | high',
2069                 'Age': 'low | medium',
2070                 'Outcome': 'high'})
2071 # Trojki Pregnancies i DiabetesPedigreeFunction (1) 221
2072 fuzzy.add_rule({'Pregnancies': 'low | medium',
2073                 'Glucose': 'medium | medium_high | high',
2074                 'BloodPressure': 'medium | medium_high | high',
2075                 'SkinThickness': 'medium | medium_high | high',
2076                 'Insulin': 'medium | high',
2077                 'BMI': 'overweight | obese',
2078                 'DiabetesPedigreeFunction': 'low | medium',
2079                 'Age': 'low | medium',
2080                 'Outcome': 'high'})
2081 # Trojki Glucose i BloodPressure (5) 226
2082 fuzzy.add_rule({'Pregnancies': 'medium | high',
2083                 'Glucose': 'low | medium_low | medium',
2084                 'BloodPressure': 'low | medium_low | medium',
2085                 'SkinThickness': 'low | medium_low | medium',
2086                 'Insulin': 'medium | high',
2087                 'BMI': 'overweight | obese',
2088                 'DiabetesPedigreeFunction': 'medium | high',
2089                 'Age': 'medium | high',
2090                 'Outcome': 'high'})
2091 fuzzy.add_rule({'Pregnancies': 'medium | high',
2092                 'Glucose': 'low | medium_low | medium',
2093                 'BloodPressure': 'low | medium_low | medium',
2094                 'SkinThickness': 'medium | medium_high | high',
2095                 'Insulin': 'low | medium',

```

```

2096         'BMI': 'overweight | obese',
2097         'DiabetesPedigreeFunction': 'medium | high',
2098         'Age': 'medium | high',
2099         'Outcome': 'high'})
2100 fuzzy.add_rule({'Pregnancies': 'medium | high',
2101                'Glucose': 'low | medium_low | medium',
2102                'BloodPressure': 'low | medium_low | medium',
2103                'SkinThickness': 'medium | medium_high | high',
2104                'Insulin': 'medium | high',
2105                'BMI': 'overweight | obese',
2106                'DiabetesPedigreeFunction': 'medium | high',
2107                'Age': 'medium | high',
2108                'Outcome': 'high'})
2109 fuzzy.add_rule({'Pregnancies': 'medium | high',
2110                'Glucose': 'low | medium_low | medium',
2111                'BloodPressure': 'low | medium_low | medium',
2112                'SkinThickness': 'medium | medium_high | high',
2113                'Insulin': 'medium | high',
2114                'BMI': 'overweight | obese',
2115                'DiabetesPedigreeFunction': 'low | medium',
2116                'Age': 'medium | high',
2117                'Outcome': 'high'})
2118 fuzzy.add_rule({'Pregnancies': 'medium | high',
2119                'Glucose': 'low | medium_low | medium',
2120                'BloodPressure': 'low | medium_low | medium',
2121                'SkinThickness': 'medium | medium_high | high',
2122                'Insulin': 'medium | high',
2123                'BMI': 'overweight | obese',
2124                'DiabetesPedigreeFunction': 'medium | high',
2125                'Age': 'low | medium',
2126                'Outcome': 'high'})
2127 # Trojki Glucose i SkinThickness (4) 230
2128 fuzzy.add_rule({'Pregnancies': 'medium | high',
2129                'Glucose': 'low | medium_low | medium',
2130                'BloodPressure': 'medium | medium_high | high',
2131                'SkinThickness': 'low | medium_low | medium',
2132                'Insulin': 'low | medium',
2133                'BMI': 'overweight | obese',
2134                'DiabetesPedigreeFunction': 'medium | high',
2135                'Age': 'medium | high',
2136                'Outcome': 'high'})
2137 fuzzy.add_rule({'Pregnancies': 'medium | high',
2138                'Glucose': 'low | medium_low | medium',
2139                'BloodPressure': 'medium | medium_high | high',
2140                'SkinThickness': 'low | medium_low | medium',
2141                'Insulin': 'medium | high',
2142                'BMI': 'overweight | obese',
2143                'DiabetesPedigreeFunction': 'medium | high',
2144                'Age': 'medium | high',
2145                'Outcome': 'high'})
2146 fuzzy.add_rule({'Pregnancies': 'medium | high',
2147                'Glucose': 'low | medium_low | medium',
2148                'BloodPressure': 'medium | medium_high | high',
2149                'SkinThickness': 'low | medium_low | medium',
2150                'Insulin': 'medium | high',

```

```

2151         'BMI': 'overweight | obese',
2152         'DiabetesPedigreeFunction': 'low | medium',
2153         'Age': 'medium | high',
2154         'Outcome': 'high'})
2155 fuzzy.add_rule({'Pregnancies': 'medium | high',
2156                'Glucose': 'low | medium_low | medium',
2157                'BloodPressure': 'medium | medium_high | high',
2158                'SkinThickness': 'low | medium_low | medium',
2159                'Insulin': 'medium | high',
2160                'BMI': 'overweight | obese',
2161                'DiabetesPedigreeFunction': 'medium | high',
2162                'Age': 'low | medium',
2163                'Outcome': 'high'})
2164 # Trojki Glucose i Insulin (3) 233
2165 fuzzy.add_rule({'Pregnancies': 'medium | high',
2166                'Glucose': 'low | medium_low | medium',
2167                'BloodPressure': 'medium | medium_high | high',
2168                'SkinThickness': 'medium | medium_high | high',
2169                'Insulin': 'low | medium',
2170                'BMI': 'overweight | obese',
2171                'DiabetesPedigreeFunction': 'medium | high',
2172                'Age': 'medium | high',
2173                'Outcome': 'high'})
2174 fuzzy.add_rule({'Pregnancies': 'medium | high',
2175                'Glucose': 'low | medium_low | medium',
2176                'BloodPressure': 'medium | medium_high | high',
2177                'SkinThickness': 'medium | medium_high | high',
2178                'Insulin': 'low | medium',
2179                'BMI': 'overweight | obese',
2180                'DiabetesPedigreeFunction': 'low | medium',
2181                'Age': 'medium | high',
2182                'Outcome': 'high'})
2183 fuzzy.add_rule({'Pregnancies': 'medium | high',
2184                'Glucose': 'low | medium_low | medium',
2185                'BloodPressure': 'medium | medium_high | high',
2186                'SkinThickness': 'medium | medium_high | high',
2187                'Insulin': 'low | medium',
2188                'BMI': 'overweight | obese',
2189                'DiabetesPedigreeFunction': 'medium | high',
2190                'Age': 'low | medium',
2191                'Outcome': 'high'})
2192 # Trojki Glucose i BMI (2) 235
2193 fuzzy.add_rule({'Pregnancies': 'medium | high',
2194                'Glucose': 'low | medium_low | medium',
2195                'BloodPressure': 'medium | medium_high | high',
2196                'SkinThickness': 'medium | medium_high | high',
2197                'Insulin': 'medium | high',
2198                'BMI': 'obese | overweight',
2199                'DiabetesPedigreeFunction': 'low | medium',
2200                'Age': 'medium | high',
2201                'Outcome': 'high'})
2202 fuzzy.add_rule({'Pregnancies': 'medium | high',
2203                'Glucose': 'low | medium_low | medium',
2204                'BloodPressure': 'medium | medium_high | high',
2205                'SkinThickness': 'medium | medium_high | high',

```



```

2206         'Insulin': 'medium | high',
2207         'BMI': 'obese | overweight',
2208         'DiabetesPedigreeFunction': 'medium | high',
2209         'Age': 'low | medium',
2210         'Outcome': 'high'})
2211 # Trojki Glucose i DiabetesPedigreeFunction (1) 236
2212 fuzzy.add_rule({'Pregnancies': 'medium | high',
2213                'Glucose': 'low | medium_low | medium',
2214                'BloodPressure': 'medium | medium_high | high',
2215                'SkinThickness': 'medium | medium_high | high',
2216                'Insulin': 'medium | high',
2217                'BMI': 'overweight | obese',
2218                'DiabetesPedigreeFunction': 'low | medium',
2219                'Age': 'low | medium',
2220                'Outcome': 'high'})
2221 # Trojki BloodPressure i SkinThickness (4) 240
2222 fuzzy.add_rule({'Pregnancies': 'medium | high',
2223                'Glucose': 'medium | medium_high | high',
2224                'BloodPressure': 'low | medium_low | medium',
2225                'SkinThickness': 'low | medium_low | medium',
2226                'Insulin': 'low | medium',
2227                'BMI': 'overweight | obese',
2228                'DiabetesPedigreeFunction': 'medium | high',
2229                'Age': 'medium | high',
2230                'Outcome': 'high'})
2231 fuzzy.add_rule({'Pregnancies': 'medium | high',
2232                'Glucose': 'medium | medium_high | high',
2233                'BloodPressure': 'low | medium_low | medium',
2234                'SkinThickness': 'low | medium_low | medium',
2235                'Insulin': 'medium | high',
2236                'BMI': 'overweight | obese',
2237                'DiabetesPedigreeFunction': 'medium | high',
2238                'Age': 'medium | high',
2239                'Outcome': 'high'})
2240 fuzzy.add_rule({'Pregnancies': 'medium | high',
2241                'Glucose': 'medium | medium_high | high',
2242                'BloodPressure': 'low | medium_low | medium',
2243                'SkinThickness': 'low | medium_low | medium',
2244                'Insulin': 'medium | high',
2245                'BMI': 'overweight | obese',
2246                'DiabetesPedigreeFunction': 'low | medium',
2247                'Age': 'medium | high',
2248                'Outcome': 'high'})
2249 fuzzy.add_rule({'Pregnancies': 'medium | high',
2250                'Glucose': 'medium | medium_high | high',
2251                'BloodPressure': 'low | medium_low | medium',
2252                'SkinThickness': 'low | medium_low | medium',
2253                'Insulin': 'medium | high',
2254                'BMI': 'overweight | obese',
2255                'DiabetesPedigreeFunction': 'medium | high',
2256                'Age': 'low | medium',
2257                'Outcome': 'high'})
2258 # Trojki BloodPressure i Insulin (3) 243
2259 fuzzy.add_rule({'Pregnancies': 'medium | high',
2260                'Glucose': 'medium | medium_high | high',

```



```

2261         'BloodPressure': 'low | medium_low | medium',
2262         'SkinThickness': 'medium | medium_high | high',
2263         'Insulin': 'low | medium',
2264         'BMI': 'overweight | obese',
2265         'DiabetesPedigreeFunction': 'medium | high',
2266         'Age': 'medium | high',
2267         'Outcome': 'high'})
2268 fuzzy.add_rule({'Pregnancies': 'medium | high',
2269               'Glucose': 'medium | medium_high | high',
2270               'BloodPressure': 'low | medium_low | medium',
2271               'SkinThickness': 'medium | medium_high | high',
2272               'Insulin': 'low | medium',
2273               'BMI': 'overweight | obese',
2274               'DiabetesPedigreeFunction': 'low | medium',
2275               'Age': 'medium | high',
2276               'Outcome': 'high'})
2277 fuzzy.add_rule({'Pregnancies': 'medium | high',
2278               'Glucose': 'medium | medium_high | high',
2279               'BloodPressure': 'low | medium_low | medium',
2280               'SkinThickness': 'medium | medium_high | high',
2281               'Insulin': 'low | medium',
2282               'BMI': 'overweight | obese',
2283               'DiabetesPedigreeFunction': 'medium | high',
2284               'Age': 'low | medium',
2285               'Outcome': 'high'})
2286 # Trojki BloodPressure i BMI (2) 245
2287 fuzzy.add_rule({'Pregnancies': 'medium | high',
2288               'Glucose': 'medium | medium_high | high',
2289               'BloodPressure': 'low | medium_low | medium',
2290               'SkinThickness': 'medium | medium_high | high',
2291               'Insulin': 'medium | high',
2292               'BMI': 'obese | overweight',
2293               'DiabetesPedigreeFunction': 'low | medium',
2294               'Age': 'medium | high',
2295               'Outcome': 'high'})
2296 fuzzy.add_rule({'Pregnancies': 'medium | high',
2297               'Glucose': 'medium | medium_high | high',
2298               'BloodPressure': 'low | medium_low | medium',
2299               'SkinThickness': 'medium | medium_high | high',
2300               'Insulin': 'medium | high',
2301               'BMI': 'obese | overweight',
2302               'DiabetesPedigreeFunction': 'medium | high',
2303               'Age': 'low | medium',
2304               'Outcome': 'high'})
2305 # Trojki BloodPressure i DiabetesPedigreeFunction (1) 246
2306 fuzzy.add_rule({'Pregnancies': 'medium | high',
2307               'Glucose': 'medium | medium_high | high',
2308               'BloodPressure': 'low | medium_low | medium',
2309               'SkinThickness': 'medium | medium_high | high',
2310               'Insulin': 'medium | high',
2311               'BMI': 'overweight | obese',
2312               'DiabetesPedigreeFunction': 'low | medium',
2313               'Age': 'low | medium',
2314               'Outcome': 'high'})
2315 # Trojki SkinThickness i Insulin (3) 249

```

```

2316     fuzzy.add_rule({'Pregnancies': 'medium | high',
2317                    'Glucose': 'medium | medium_high | high',
2318                    'BloodPressure': 'medium | medium_high | high',
2319                    'SkinThickness': 'low | medium_low | medium',
2320                    'Insulin': 'low | medium',
2321                    'BMI': 'overweight | obese',
2322                    'DiabetesPedigreeFunction': 'medium | high',
2323                    'Age': 'medium | high',
2324                    'Outcome': 'high'})
2325     fuzzy.add_rule({'Pregnancies': 'medium | high',
2326                    'Glucose': 'medium | medium_high | high',
2327                    'BloodPressure': 'medium | medium_high | high',
2328                    'SkinThickness': 'low | medium_low | medium',
2329                    'Insulin': 'low | medium',
2330                    'BMI': 'overweight | obese',
2331                    'DiabetesPedigreeFunction': 'low | medium',
2332                    'Age': 'medium | high',
2333                    'Outcome': 'high'})
2334     fuzzy.add_rule({'Pregnancies': 'medium | high',
2335                    'Glucose': 'medium | medium_high | high',
2336                    'BloodPressure': 'medium | medium_high | high',
2337                    'SkinThickness': 'low | medium_low | medium',
2338                    'Insulin': 'low | medium',
2339                    'BMI': 'overweight | obese',
2340                    'DiabetesPedigreeFunction': 'medium | high',
2341                    'Age': 'low | medium',
2342                    'Outcome': 'high'})
2343     # Trojki SkinThickness i BMI (2) 251
2344     fuzzy.add_rule({'Pregnancies': 'medium | high',
2345                    'Glucose': 'medium | medium_high | high',
2346                    'BloodPressure': 'medium | medium_high | high',
2347                    'SkinThickness': 'low | medium_low | medium',
2348                    'Insulin': 'medium | high',
2349                    'BMI': 'obese | overweight',
2350                    'DiabetesPedigreeFunction': 'low | medium',
2351                    'Age': 'medium | high',
2352                    'Outcome': 'high'})
2353     fuzzy.add_rule({'Pregnancies': 'medium | high',
2354                    'Glucose': 'medium | medium_high | high',
2355                    'BloodPressure': 'medium | medium_high | high',
2356                    'SkinThickness': 'low | medium_low | medium',
2357                    'Insulin': 'medium | high',
2358                    'BMI': 'obese | overweight',
2359                    'DiabetesPedigreeFunction': 'medium | high',
2360                    'Age': 'low | medium',
2361                    'Outcome': 'high'})
2362     # Trojki SkinThickness i DiabetesPedigreeFunction (1) 252
2363     fuzzy.add_rule({'Pregnancies': 'medium | high',
2364                    'Glucose': 'medium | medium_high | high',
2365                    'BloodPressure': 'medium | medium_high | high',
2366                    'SkinThickness': 'low | medium_low | medium',
2367                    'Insulin': 'medium | high',
2368                    'BMI': 'overweight | obese',
2369                    'DiabetesPedigreeFunction': 'low | medium',
2370                    'Age': 'low | medium',

```

```

2371         'Outcome': 'high'})
2372 # Trojki Insulin i BMI (2) 254
2373 fuzzy.add_rule({'Pregnancies': 'medium | high',
2374                'Glucose': 'medium | medium_high | high',
2375                'BloodPressure': 'medium | medium_high | high',
2376                'SkinThickness': 'medium | medium_high | high',
2377                'Insulin': 'low | medium',
2378                'BMI': 'obese | overweight',
2379                'DiabetesPedigreeFunction': 'low | medium',
2380                'Age': 'medium | high',
2381                'Outcome': 'high'})
2382 fuzzy.add_rule({'Pregnancies': 'medium | high',
2383                'Glucose': 'medium | medium_high | high',
2384                'BloodPressure': 'medium | medium_high | high',
2385                'SkinThickness': 'medium | medium_high | high',
2386                'Insulin': 'low | medium',
2387                'BMI': 'obese | overweight',
2388                'DiabetesPedigreeFunction': 'medium | high',
2389                'Age': 'low | medium',
2390                'Outcome': 'high'})
2391 # Trojki Insulin i DiabetesPedigreeFunction (1) 255
2392 fuzzy.add_rule({'Pregnancies': 'medium | high',
2393                'Glucose': 'medium | medium_high | high',
2394                'BloodPressure': 'medium | medium_high | high',
2395                'SkinThickness': 'medium | medium_high | high',
2396                'Insulin': 'low | medium',
2397                'BMI': 'overweight | obese',
2398                'DiabetesPedigreeFunction': 'low | medium',
2399                'Age': 'medium | high',
2400                'Outcome': 'high'})
2401 # Trojki BMI i DiabetesPedigreeFunction (1) 256
2402 fuzzy.add_rule({'Pregnancies': 'medium | high',
2403                'Glucose': 'medium | medium_high | high',
2404                'BloodPressure': 'medium | medium_high | high',
2405                'SkinThickness': 'medium | medium_high | high',
2406                'Insulin': 'medium | high',
2407                'BMI': 'obese | overweight',
2408                'DiabetesPedigreeFunction': 'low | medium',
2409                'Age': 'low | medium',
2410                'Outcome': 'high'})

```

3.9 Testowanie klasyfiaktora zbioru rozmytego

```

1     # Test dzialania Klasyfikatora rozmytego:
2     for _ in range(3):
3         shuffled_data = ProcessingData.shuffle(cleared_data)
4         _, validation_data = ProcessingData.split(shuffled_data, 0.7)
5
6         # true negative
7         tn = 0
8         # false positive
9         fp = 0
10        # false negative

```

```

11     fn = 0
12     # true positive
13     tp = 0
14
15     # sprawdzenie dla kazdej probki w zbiorze walidacyjnym
16     for _, value in validation_data.iterrows():
17         # zapisanie wyniku i jego wartosci liczbowej do zmiennych
18         result = fuzzy.compute(value)
19         outcome = value['Outcome']
20
21         # zliczenie przypadkow
22         if outcome == 0:
23             if result[0] == 'low':
24                 tn += 1
25             elif result[0] == 'high':
26                 fp += 1
27         else:
28             if result[0] == 'high':
29                 tp += 1
30             elif result[0] == 'low':
31                 fn += 1
32
33     # wyswietlenie dokladnosci
34     print(f'Accuracy for all: {(tn + tp)/(tn + fp + fn + tp)}; {(tn +
35         tp) / (tn + fp + fn + tp) * 100:0.2f}%')
36
37     # Tworzenie tablicy pomylek
38     cf_matrix = np.array([[tn, fp],
39                           [fn, tp]])
40     group_names = ['True Neg', 'False Pos', 'False Neg', 'True Pos']
41     group_counts = [f'{value:0.0f}' for value in cf_matrix.flatten()]
42     group_percentages = [f'{value:.2f}' for value in cf_matrix.
43                           flatten() / np.sum(cf_matrix)]
44
45     labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in zip(group_names,
46                                                           group_counts, group_percentages)]
47
48     labels = np.asarray(labels).reshape(2,2)
49
50     plt.figure(figsize=(16, 12))
51     sns.set(font_scale=1.4)
52     plot = sns.heatmap(cf_matrix, annot=labels, annot_kws={'size':
53         16}, fmt='', cmap='Blues')
54     plot.set_title('Tablica pomylek')
55     plot.set_xlabel('Przewidziane etykiety')
56     plot.set_ylabel('Prawdziwe etykiety')
57     plt.show()

```
