

# Dokumentacja Techniczna Systemu Ekspertowego BrewSense

Gabriela Bieniek, Kacper Bułaś, Michał Dyjak, Mateusz Florian

16 grudnia 2025

## Spis treści

|          |   |          |
|----------|---|----------|
| <b>1</b> | <b>Wprowadzenie</b>                                     | <b>2</b> |
| 1.1      | Cel projektu . . . . .                                  | 2        |
| 1.2      | Zakres działania systemu . . . . .                      | 2        |
| 1.3      | Zastosowanie logiki rozmytej w ocenie jakości . . . . . | 2        |
| <b>2</b> | <b>Architektura systemu</b>                             | <b>2</b> |
| 2.1      | Ogólny schemat działania . . . . .                      | 2        |
| 2.2      | Główne komponenty . . . . .                             | 2        |
| 2.3      | Przepływ danych . . . . .                               | 3        |
| <b>3</b> | <b>Model systemu rozmytego</b>                          | <b>3</b> |
| 3.1      | Zmienne wejściowe . . . . .                             | 3        |
| 3.2      | Zmienna wyjściowa . . . . .                             | 3        |
| 3.3      | Funkcje przynależności . . . . .                        | 4        |
| 3.4      | Reguły rozmyte i wnioskowanie . . . . .                 | 4        |
| 3.5      | Defuzyfikacja . . . . .                                 | 4        |
| <b>4</b> | <b>Implementacja</b>                                    | <b>4</b> |
| 4.1      | Inicjalizacja systemu . . . . .                         | 4        |
| 4.2      | Kluczowe funkcje . . . . .                              | 5        |
| 4.3      | Decyzje projektowe . . . . .                            | 5        |
| <b>5</b> | <b>Scenariusze testowe i eksperymenty</b>               | <b>5</b> |
| 5.1      | Metodologia . . . . .                                   | 5        |
| 5.2      | Przypadki testowe . . . . .                             | 5        |
| 5.3      | Wizualizacja wyników . . . . .                          | 5        |
| <b>6</b> | <b>Analiza wyników</b>                                  | <b>6</b> |
| 6.1      | Charakterystyka odpowiedzi systemu . . . . .            | 6        |
| 6.2      | Stabilność . . . . .                                    | 6        |
| 6.3      | Ograniczenia . . . . .                                  | 6        |
| <b>7</b> | <b>Możliwości rozbudowy</b>                             | <b>6</b> |
| <b>8</b> | <b>Podsumowanie</b>                                     | <b>7</b> |

# 1 Wprowadzenie

## 1.1 Cel projektu

Celem projektu BrewSense jest stworzenie zautomatyzowanego systemu wspomagania decyzji, służącego do oceny jakości naparu kawowego. System ma na celu obiektywizację procesu oceny sensorycznej poprzez zastosowanie algorytmów sztucznej inteligencji, a konkretnie logiki rozmytej (ang. *fuzzy logic*). Aplikacja pozwala użytkownikowi na zdefiniowanie parametrów fizykochemicznych naparu i otrzymanie natychmiastowej, interpretowalnej oceny jakościowej wraz z uzasadnieniem.

## 1.2 Zakres działania systemu

System przetwarza cztery kluczowe parametry wejściowe:

- Gorycz (skala sensoryczna),
- Kwasowość (skala sensoryczna),
- Aromat (skala sensoryczna),
- Temperatura naparu (stopnie Celsjusza).

Na podstawie zdefiniowanej bazy wiedzy eksperckiej, system generuje wynik punktowy w skali 0–100 oraz werbalną klasyfikację jakości kawy.

## 1.3 Zastosowanie logiki rozmytej w ocenie jakości

Ocena jakości kawy jest procesem z natury nieprecyzyjnym i subiektywnym. Pojęcia takie jak "lekko gorzka" czy "zbyt gorąca" są trudne do zdefiniowania przy użyciu klasycznej logiki binarnej (prawda/fałsz). Logika rozmyta pozwala na matematyczne modelowanie niepewności i nieprecyzyjności języka naturalnego. Dzięki zastosowaniu funkcji przynależności, system BrewSense potrafi ocenić stopień, w jakim dany parametr należy do zbioru (np. zbioru "wysoka jakość"), co pozwala na płynne przejścia między ocenami i naśladowanie ludzkiego procesu decyzyjnego.

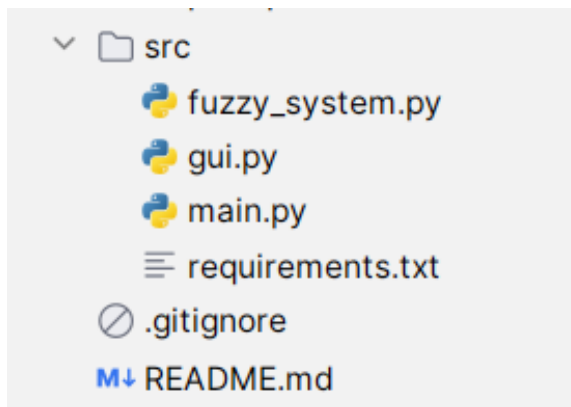
# 2 Architektura systemu

## 2.1 Ogólny schemat działania

System BrewSense oparty jest na architekturze modularnej, separującej warstwę prezentacji (GUI) od logiki biznesowej (Fuzzy Logic Engine). Program został zaimplementowany w języku Python.

## 2.2 Główne komponenty

- **Warstwa Logiki (Backend):** Oparta na bibliotece `scikit-fuzzy`. Odpowiada za definicję zmiennych lingwistycznych, funkcji przynależności, bazy reguł oraz proces wnioskowania. Plik: `fuzzy_system.py`.
- **Warstwa Prezentacji (Frontend):** Zrealizowana przy użyciu biblioteki `PyQt5`. Odpowiada za interakcję z użytkownikiem, wizualizację parametrów (w tym animację poziomu cieczy i pary) oraz prezentację wyników. Plik: `gui.py`.



Rysunek 1: Schemat blokowy architektury systemu BrewSense

- **Punkt Wejścia:** Plik `main.py` inicjalizujący aplikację i pętlę zdarzeń.

## 2.3 Przepływ danych

1. Użytkownik wprowadza parametry wejściowe za pomocą suwaków w GUI. 2. Warstwa GUI przekazuje wartości numeryczne do instancji klasy `CoffeeQualitySystem`. 3. Silnik rozmyty dokonuje "fazyfikacji" (rozmycia) danych wejściowych. 4. Następuje wnioskowanie na podstawie bazy reguł. 5. Wynik jest poddawany "defuzyfikacji" (wyostrażaniu) do konkretnej wartości liczbowej. 6. Wynik oraz wygenerowane wyjaśnienie tekstowe są zwracane do GUI i prezentowane użytkownikowi.

## 3 Model systemu rozmytego

Model decyzyjny jest sercem projektu BrewSense. Wykorzystano sterownik rozmyty typu Mamdani.

### 3.1 Zmienne wejściowe

Zdefiniowano cztery zmienne wejściowe (Antecedents):

| Zmienna             | Zakres      | Interpretacja fizyczna                            |
|---------------------|-------------|---|
| Bitterness (Gorycz) | 0.0 – 10.0  | Intensywność goryczy (0 - brak, 10 - ekstremalna) |
| Acidity (Kwasowość) | 0.0 – 10.0  | Poziom kwasowości (pożądana w średnim zakresie)   |
| Aroma (Aromat)      | 0.0 – 10.0  | Intensywność i jakość zapachu                     |
| Temperature         | 60.0 – 95.0 | Temperatura serwowania w °C                       |

Tabela 1: Zmienne wejściowe systemu.

### 3.2 Zmienna wyjściowa

Zmienna wyjściowa (Consequent): **Quality** (Jakość).

- **Zakres:** 0.0 – 100.0 pkt.
- **Zbiory lingwistyczne:** Low (Niska), Medium (Średnia), High (Wysoka).

### 3.3 Funkcje przynależności

Zastosowano trójkątne funkcje przynależności (`trimf`) ze względu na ich prostotę obliczeniową i łatwość interpretacji.

Dla typowej zmiennej (np. Gorycz), zbiory zdefiniowano następująco:

- **Low (Niska)**: Funkcja malejąca, szczyt w 0.
- **Medium (Średnia)**: Funkcja trójkątna, szczyt w 5.
- **High (Wysoka)**: Funkcja rosnąca, szczyt w 10.

Wyjątkiem jest zmienna *Temperature*, gdzie zakresy są dostosowane do fizyki parzenia kawy (np. optimum w okolicach 75-85 °C).

### 3.4 Reguły rozmyte i wnioskowanie

Baza wiedzy składa się z zestawu reguł typu IF-THEN. Przykładowa struktura logiczna:

$$R_1 : \text{IF } (Aroma \text{ is } High) \text{ AND } (Temp \text{ is } Good) \text{ THEN } (Quality \text{ is } High) \quad (1)$$

Zastosowano mechanizm **Catch-All**, czyli reguły domyślne, które aktywują się w przypadkach niejednoznacznych, zapobiegając sytuacji, w której system nie zwraca żadnego wyniku.

### 3.5 Defuzyfikacja

Do przekształcenia wynikowego zbioru rozmytego w konkretną wartość liczbową (crisp value) wykorzystano metodę środka ciężkości (Centroid):

$$z^* = \frac{\int \mu_C(z) \cdot z \, dz}{\int \mu_C(z) \, dz} \quad (2)$$

Gdzie  $\mu_C(z)$  to wypadkowa funkcja przynależności dla zmiennej wyjściowej.

## 4 Implementacja

Implementacja logiki znajduje się w klasie `CoffeeQualitySystem` w pliku `fuzzy_system.py`.

### 4.1 Inicjalizacja systemu

Poniższy listing przedstawia sposób definicji zmiennych przy użyciu biblioteki `scikit-fuzzy`:

```
1 def _create_variables(self):
2     # Zmienne wejściowe
3     self.bitterness = ctrl.Antecedent(np.arange(0, 10.1, 0.1), 'bitterness')
4     self.acidity = ctrl.Antecedent(np.arange(0, 10.1, 0.1), 'acidity')
5     self.aroma = ctrl.Antecedent(np.arange(0, 10.1, 0.1), 'aroma')
6     self.temperature = ctrl.Antecedent(np.arange(60, 95.1, 0.1), '
    temperature')
7
8     # Zmienna wyjściowa
9     self.quality = ctrl.Consequent(np.arange(0, 100.1, 0.1), 'quality')
```

Listing 1: Definicja zmiennych rozmytych

## 4.2 Kluczowe funkcje

- `_create_membership_functions()`: Automatyczne lub ręczne przypisanie kształtów funkcji (np. `automf(3)` dla standardowych trójkątów).
- `_create_rules()`: Definicja logiki biznesowej. Implementacja zawiera reguły promujące balans smaku i karzące ekstremalne odchylenia (np. zbyt wysoka temperatura "zabija" jakość).
- `evaluate(bitterness, acidity, aroma, temperature)`: Główna metoda publiczna. Przyjmuje ostre wartości wejściowe, wykonuje symulację w `ctrl.ControlSystemSimulation` i zwraca wynik.

## 4.3 Decyzje projektowe

W warstwie prezentacji (`gui.py`) zastosowano `QPropertyAnimation` do wizualizacji poziomu kawy. Dodatkowo, w zależności od zwróconej temperatury i wyniku jakości, kolor cieczy w wirtualnym kubku zmienia się dynamicznie (od "lury" po głęboki brąz), co stanowi dodatkowe sprzężenie zwrotne dla użytkownika.

# 5 Scenariusze testowe i eksperymenty

Testy przeprowadzono w celu weryfikacji poprawności reguł oraz stabilności systemu.

## 5.1 Metodologia

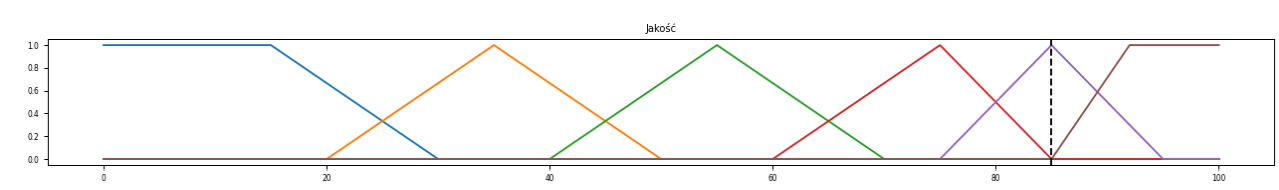
Testy polegały na wprowadzeniu wektorów danych wejściowych reprezentujących typowe profile kawy (np. Espresso, Americano) oraz przypadki skrajne.

## 5.2 Przypadki testowe

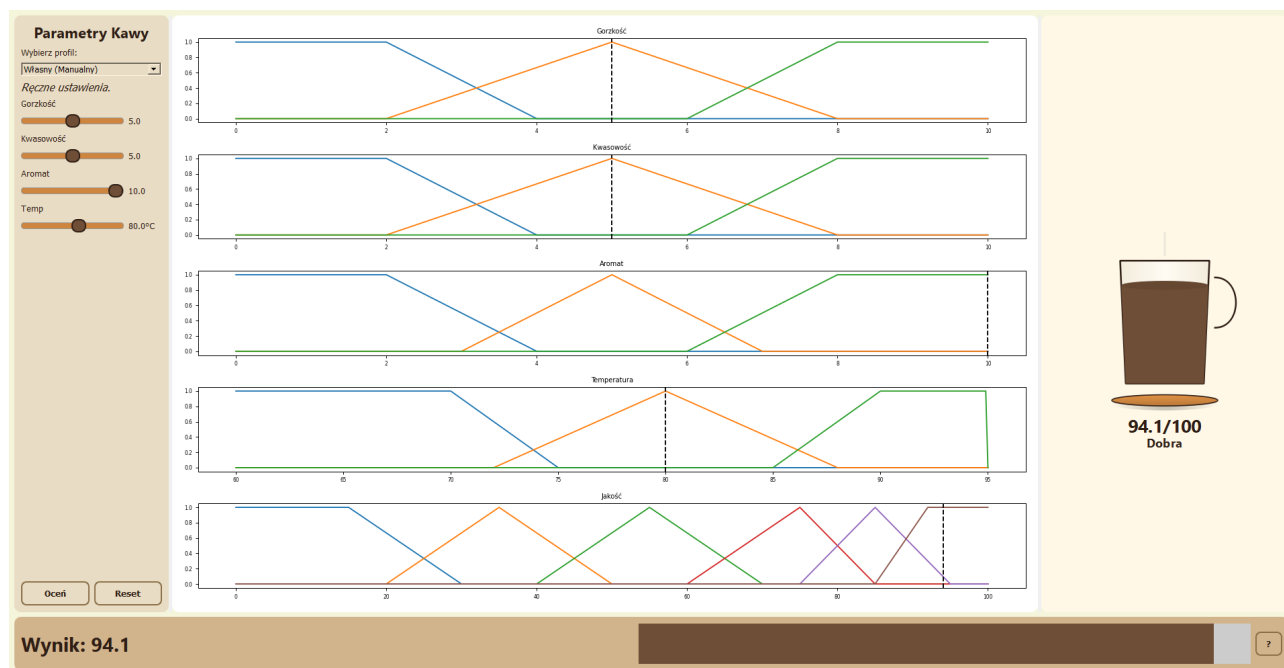
| ID | Gorycz | Kwasowość | Aromat | Temp [°C] | Wynik Oczekiwany       | Opis                    |
|----|--------|-----------|--------|-----------|------------------------|-------------------------|
| T1 | 2.0    | 5.0       | 9.0    | 80.0      | <b>Wysoki (&gt;80)</b> | Idealna, aromatyczna k  |
| T2 | 9.0    | 8.0       | 2.0    | 95.0      | <b>Niski (&lt;30)</b>  | Przepalona, gorzka, wrz |
| T3 | 5.0    | 5.0       | 5.0    | 75.0      | <b>Średni ( 50)</b>    | Przeciętna kawa         |
| T4 | 1.0    | 9.0       | 6.0    | 65.0      | <b>Niski/Śr</b>        | Kwaśna i zimna ("lura   |

Tabela 2: Zdefiniowane przypadki testowe.

## 5.3 Wizualizacja wyników



Rysunek 2: Wizualizacja przestrzeni wyjściowej systemu rozmytego.



Rysunek 3: Interfejs aplikacji BrewSense podczas analizy próbki

## 6 Analiza wyników

### 6.1 Charakterystyka odpowiedzi systemu

System wykazuje nieliniową charakterystykę odpowiedzi, co jest typowe dla sterowników rozmytych.

- **Czułość na temperaturę:** Zauważono silny spadek oceny jakości przy przekroczeniu 90°C, co jest zgodne z wiedzą baristyczną (degradacja związków smakowych).
- **Synergia parametrów:** Wysoki aromat potrafi nieznacznie podnieść ocenę nawet przy średniej goryczy, jednak ekstremalna kwasowość dominuje nad aromatem, drastycznie obniżając wynik.

### 6.2 Stabilność

Dzięki zastosowaniu reguł "catch-all" oraz ciągłych funkcji przynależności, system jest stabilny. Nie obserwuje się nagłych skoków wartości wyjściowej przy minimalnych zmianach wejścia (w przeciwieństwie do systemów opartych na sztywnych progach *if-else*).

### 6.3 Ograniczenia

Obecny model zakłada uniwersalny wzorzec kawy. Nie rozróżnia gatunków (np. Robusta vs Arabica), gdzie wysoka kwasowość może być cechą pożądaną dla Arabiki, a wadą dla Robusty.

## 7 Możliwości rozbudowy

System BrewSense posiada otwartą architekturę, co pozwala na dalszy rozwój. Rekomendowane kierunki zmian:

1. **Profile Kawowe:** Dodanie zmiennej kontekstowej "Typ ziarna", która dynamicznie modyfikowałaby bazę reguł (np. inne reguły dla espresso, inne dla kawy przelewowej).
2. **Nowe zmienne wejściowe:**
  - Czas ekstrakcji (sekundy),
  - Stopień zmielenia (skala mikrometryczna),
  - Ciśnienie (bary).
3. **Kalibracja ekspercka:** Dostrojenie kształtu funkcji przynależności (np. przejście z trójkątnych na gaussowskie) w oparciu o dane z rzeczywistych cuppingów (profesjonalnych testów kawy).

## 8 Podsumowanie

Projekt BrewSense stanowi funkcjonalną implementację systemu ekspertowego opartego na logice rozmytej. Analiza wykazała, że system poprawnie modeluje intuicyjne zasady oceny jakości kawy, oferując płynne i interpretowalne wyniki.

Główne osiągnięcia projektu to:

- Skuteczna implementacja sterownika Mamdani w środowisku Python.
- Stworzenie responsywnego interfejsu graficznego z wizualizacją parametrów.
- Opracowanie stabilnej bazy reguł pokrywającej większość scenariuszy parzenia kawy.

Rozwiązanie to może stanowić bazę dydaktyczną do nauki systemów sterowania lub prototyp dla komercyjnych ekspresów do kawy wyposażonych w inteligentne systemy diagnostyczne.