

UNIwersytet Rzeszowski
Wydział Nauk Ścisłych i Technicznych



Piotr Smoła
nr albumu: 125162
Kierunek: Informatyka

System rekomendacji produktów wykorzystujący filtrację opartą na treści, logikę rozmytą i modele probabilistyczne

Praca inżynierska

Praca wykonana pod kierunkiem
dr inż. Piotra Grochowalskiego

Rzeszów, 2026

Spis treści

Wstęp	3
Rozdział 1: Teoretyczne podstawy metod rekomendacyjnych	6
Rozdział 2: Weryfikacja i analiza rozwiązań alternatywnych	15
Rozdział 3: Projekt systemu rekomendacyjnego	22
3.1 Cel i zakres aplikacji	22
3.2 Typy użytkowników systemu	22
3.3 Wymagania funkcjonalne systemu	23
3.4 Diagram przypadków użycia	25
3.5 Architektura funkcjonalna systemu	27
Rozdział 4: Przedstawienie wykorzystanego stosu technologicznego oraz praktycznej realizacji projektu	32
4.1 Stos technologiczny	32
4.2 Architektura praktyczna: komponenty systemu i wdrożenie	34
Rozdział 5: Implementacja algorytmów rekomendacji	40
5.1 Content-Based Filtering	40
5.2 Logika rozmyta w systemie rekomendacji	46
5.3 Modele probabilistyczne – Markov Chain i Naive Bayes	56
Rozdział 6: Funkcjonowanie systemu rekomendacji w praktyce	63
6.1 Przegląd interfejsu użytkownika	63
6.2 Rekomendacje Content-Based Filtering	64
6.3 Rekomendacje Fuzzy Logic	68
6.4 Rekomendacje Probabilistic Models	77
Rozdział 7: Porównanie i ewaluacja metod rekomendacyjnych	87
Rozdział 8: Podsumowanie i wnioski końcowe	95
Wykaz rysunków i tabel	99
Spis tabel	99
Streszczenie	100
Załącznik A: Szczegółowe scenariusze użycia	101

Wstęp

Współczesny handel elektroniczny charakteryzuje się ogromną różnorodnością oferty - przeciętny sklep internetowy może posiadać nawet dziesiątki tysięcy pozycji w katalogu. Taka obfitość, choć korzystna teoretycznie, paradoksalnie utrudnia proces decyzyjny klientom, którzy przytłoczeni liczbą dostępnych wariantów często porzucają zakupy. Dla właścicieli platform e-commerce przekłada się to bezpośrednio na utracone transakcje oraz niższą wartość sprzedaży - klienci kupują produkty niekoniecznie najlepiej dopasowane do swoich potrzeb lub w ogóle rezygnują z zakupu.

Mechanizmy rekomendacyjne adresują ten problem przez inteligentną selekcję i prezentację produktów najbardziej odpowiadających indywidualnym potrzebom użytkownika, co skutkuje wzrostem konwersji oraz średniej wartości zamówienia [6].

Prezentowany system stanowi rezultat pracy dwuosobowego zespołu projektowego nad kompleksową platformą e-commerce. W ramach współpracy dokonano podziału odpowiedzialności algorytmicznych: niniejsza praca obejmuje trzy spośród sześciu zaimplementowanych metod rekomendacyjnych - filtrację opartą na treści (Content-Based Filtering), wnioskowanie rozmyte (Fuzzy Logic) oraz modele probabilistyczne (łańcuchy Markowa połączone z klasyfikatorem Bayesowskim). Trzy pozostałe metody - filtracja kolaboratywna, analiza sentymentu oraz reguły asocjacyjne Apriori - zostały zrealizowane przez współautora projektu (Dawid Olko).

Motywacja i kontekst problemu

Problem rekomendacji w platformach e-commerce jest wieloaspektowy: preferencje użytkowników są subiektywne, dane często niekompletne (problem zimnego startu dla nowych użytkowników i produktów), a katalogi dynamiczne. Istniejące rozwiązania komercyjne (Amazon Personalize, Google Recommendations AI) działają jako czarne skrzynki bez kontroli nad algorytmami. Biblioteki open-source (Apache Mahout, Surprise) nie oferują logiki rozmytej ani zaawansowanych modeli probabilistycznych w jednym systemie.

Rzeczona praca odpowiada na te wyzwania poprzez modułowy system łączący trzy komplementarne podejścia: filtrację opartą na treści (CBF), logikę rozmytą oraz modele probabilistyczne (Markov, Naive Bayes). System zaimplementowano od podstaw, zapewniając pełną kontrolę i możliwość dostosowania do wymagań platformy e-commerce.

Cel i zakres pracy

Celem pracy jest opracowanie oraz wdrożenie zaawansowanego mechanizmu rekomendacyjnego w ramach funkcjonalnej platformy e-commerce. Realizacja odbywała się w trybie współpracy zespołowej, gdzie odpowiedzialność za różne komponenty systemu została rozdzielona pomiędzy dwóch autorów - niniejsza praca dokumentuje projektowanie i budowę trzech metod analitycznych (Content-Based Filtering, Fuzzy Logic, modele probabilistyczne), podczas gdy współautor (Dawid Olko) zajmował się trzema metodami komplementarnymi (Collaborative Filtering, Sentiment Analysis, Apriori). Całość tworzy spójny ekosystem sześciu algorytmów wzajemnie uzupełniających się funkcjonalnie.

Szczegółowe cele realizacyjne obejmują:

- Zaprojektowanie architektury modułowego systemu rekomendacyjnego z architekturą trójwarstwową (Django + React + PostgreSQL)
- Implementacja algorytmu filtracji opartej na treści (Content-Based Filtering) z wykorzystaniem ważonych wektorów cech (kategorie 40%, tagi 30%, cena 20%, słowa kluczowe 10%) i miary podobieństwa kosinusowego - metoda rozwiązuje problem zimnego startu dla nowych produktów poprzez analizę atrybutów niezależnie od historii transakcyjnej
- Opracowanie systemu wnioskowania rozmytego typu Mamdani z regułami IF-THEN i funkcjami przynależności (trójkątne i trapezoidalne) dla trzech wymiarów ewaluacji produktu: cena, jakość oraz popularność, z uwzględnieniem profilu wrażliwości cenowej użytkownika oraz jego preferencji zakupowych
- Zbudowanie modeli probabilistycznych: łańcucha Markowa pierwszego rzędu dla predykcji sekwencji zakupowych oraz naiwnego klasyfikatora Bayesa dla predykcji odejścia klienta (ang. *churn*) i prawdopodobieństwa zakupu - modele przewidują przyszłe zachowania konsumenckie na podstawie wzorców historycznych
- Optymalizacja wydajności systemu dla wdrożenia produkcyjnego poprzez zastosowanie pamięci podręcznej Django, operacji zbiorczych oraz konteneryzacji Docker
- Przeprowadzenie ewaluacji jakości rekomendacji na rzeczywistych danych (500 produktów, 20 użytkowników, 200 zamówień) z analizą pokrycia katalogowego, czasu odpowiedzi oraz poprawności merytorycznej sugestii

Merytoryczny zakres opracowania obejmuje teoretyczne fundamenty algorytmów rekomendacyjnych, proces projektowania i wytwarzania oprogramowania oraz

weryfikację empiryczną w środowisku Django 5.1.4 / React 18 / PostgreSQL 14. Kluczowymi aspektami jakościowymi analizowanych metod są: eliminacja problemu zimnego startu (Content-Based Filtering), transparentność procesu decyzyjnego (Fuzzy Logic z regułami IF-THEN) oraz zdolność prognozowania zachowań zakupowych (modele probabilistyczne).

Rozdział 1

Teoretyczne podstawy metod rekomendacyjnych

Niniejszy rozdział przedstawia fundamenty teoretyczne trzech metod rekomendacyjnych zastosowanych w systemie: algorytmów opartych na analizie cech produktowych, wnioskowania z wykorzystaniem logiki rozmytej oraz modeli probabilistycznych służących predykcji zachowań użytkowników. Omówienie obejmuje genezę systemów rekomendacyjnych, matematyczne podstawy poszczególnych podejść oraz uzasadnienie wyborów implementacyjnych.

Historia i ewolucja systemów rekomendacyjnych

Geneza systemów rekomendacyjnych sięga początków cyfrowego handlu w latach 90., gdy dynamiczny wzrost oferty sklepów internetowych stworzył potrzebę narzędzi wspierających nawigację po tysiącach produktów. Przełomowe znaczenie miało wdrożenie przez Amazon.com mechanizmu Item-Based Collaborative Filtering (Linden et al., 2003), który identyfikował produkty komplementarne na podstawie historycznych współwystąpień w koszykach zakupowych. Funkcjonalność "Customers who bought..." osiągnęła 29% wzrost przychodów już w pierwszym roku, dowodząc ekonomicznej wartości personalizacji. Techniczne rozwiązanie opierało się na prekompilowanych macierzach podobieństw produktowych (przetwarzanie wsadowe, ang. *batch processing*), co umożliwiło odpowiedzi w czasie rzeczywistym nawet przy milionowych katalogach. Amazon wykorzystał skorygowane podobieństwo kosinusowe (ang. *Adjusted Cosine Similarity*) z normalizacją per-user, eliminując zniekształcenia wynikające z indywidualnych skal oceniania.

Akceleratorem badań nad zaawansowanymi algorytmami stał się konkurs Netflix Prize (2006-2009) z nagrodą miliona dolarów, który zintensyfikował prace nad faktoryzacją macierzową, uczeniem zespołowym (ang. *ensemble learning*) oraz metodami głębokiego uczenia (ang. *deep learning approaches*). Obecnie mechanizmy rekomendacyjne stanowią infrastrukturalny element ekosystemu e-commerce, platform strumieniowania wideo (VOD), serwisów muzycznych oraz portali społecznościowych (ang. *social networks*).

Trajektoria rozwoju prowadziła od prymitywnych rankingów popularności, przez filtrację kolaboratywną i opartą na treści, ku architekturom hybrydowym. Współczesne trendy obejmują sieci neuronowe (ang. *neural networks*), równoważenie eksploracji i eksploatacji przy użyciu kontekstowych automatów (ang. *contextual bandits*), wyjaśnialność sztucznej inteligencji (ang. *explainability*, XAI) oraz personalizację w czasie rzeczywistym (ang. *real-time personalization*).

Content-Based Filtering – podstawy teoretyczne

Content-Based Filtering (CBF, filtracja oparta na treści) jest jedną z fundamentalnych metod systemów rekomendacyjnych. W przeciwieństwie do Collaborative Filtering, CBF analizuje cechy samych produktów, a nie wzorce zachowań użytkowników. Metoda została szczegółowo opisana w literaturze [1].

Zasada działania

System buduje profil cech każdego produktu (wektor cech) i oblicza podobieństwo między produktami na podstawie ich cech. Użytkownikowi rekomendowane są produkty podobne do tych, które wcześniej przeglądał lub kupił.

Reprezentacja produktu jako wektora cech

Każdy produkt p jest reprezentowany jako wektor w wielowymiarowej przestrzeni cech:

$$\vec{p} = (f_1, f_2, \dots, f_n) \quad (1)$$

gdzie f_i to waga cechy i (np. należenie do kategorii, posiadanie tagu, przedział cenowy). W ogólnym przypadku stosuje się wagi różnicujące znaczenie poszczególnych cech:

$$\vec{p} = \sum_i w_i \cdot f_i(p) \quad (2)$$

gdzie w_i to waga cechy i , a $f_i(p)$ to wartość cechy dla produktu p . Funkcja indykatorowa $\mathbf{1}_{feature}(p)$ przyjmuje wartość 1 jeśli produkt posiada daną cechę, 0 w przeciwnym razie.

Zalety CBF:

- Brak problemu zimnego startu dla nowych produktów – wystarczy opis i cechy
- Przezroczystość rekomendacji – można wyjaśnić dlaczego produkt został polecany ("podobna kategoria", "podobne tagi")
- Niezależność od innych użytkowników – działa nawet dla pierwszego klienta w systemie
- Szybka aktualizacja – dodanie nowego produktu nie wymaga przeliczenia całej macierzy

Wady CBF:

- Problem bańki filtrującej (ang. *filter bubble*) – rekomenduje tylko podobne produkty, użytkownik nie odkrywa nowych kategorii
- Wymaga dobrze opisanych cech produktów – jakość rekomendacji zależy od jakości metadanych
- Nie odkrywa nieoczywistych powiązań między produktami (np. "użytkownicy kupujący kawę często kupują cukier")
- Ograniczenie do podobieństwa cech – nie uwzględnia kontekstu użytkownika

Uzasadnienie wyboru podobieństwa kosinusowego i TF-IDF

W metodzie Content-Based Filtering produkty są reprezentowane jako wektory w wielowymiarowej przestrzeni cech, gdzie każdy wymiar odpowiada jednej cesze (kategoria, tag, cena). Do porównania takich wektorowych reprezentacji produktów wybrano podobieństwo kosinusowe (ang. *Cosine Similarity*) ze względu na dwie kluczowe właściwości: (1) mierzy kąt między wektorami, a nie ich długość, co czyni je odporne na różnice w liczbie cech między produktami (laptop z 20 tagami vs laptop z 5 tagami), (2) normalizuje wynik do zakresu $[0, 1]$, co umożliwia intuicyjną interpretację jako procent podobieństwa. Schemat ważenia TF-IDF (ang. *Term Frequency - Inverse Document Frequency*) został wybrany do reprezentacji tekstowych cech (tagi, słowa kluczowe), ponieważ automatycznie przypisuje niższą wagę cechom powszechnym (np. tag "elektronika" występujący w 80% produktów) i wyższą wagę cechom rzadkim, wyróżniającym (np. tag "gaming 4K" występujący tylko w konkretnych laptopach), co zwiększa precyzję rekomendacji. Zastosowanie tych metryk pozwala na efektywne porównywanie produktów w kontekście e-commerce, gdzie produkty różnią się znacząco liczbą i rodzajem atrybutów [9].

Podobieństwo kosinusowe – dla dwóch wektorów \vec{A} i \vec{B} :

$$\cos(\theta) = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\| \times \|\vec{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \times \sqrt{\sum_{i=1}^n B_i^2}} \quad (3)$$

gdzie \vec{A} i \vec{B} to wektory cech dwóch produktów. Wynik mieści się w przedziale $[0, 1]$ dla nieujemnych wektorów (w kontekście TF-IDF i wag binarnych).

Interpretacja podobieństwa kosinusowego:

- $\cos(\theta) = 1$ – wektory identyczne (produkty mają te same cechy)
- $\cos(\theta) = 0$ – wektory ortogonalne (brak wspólnych cech)
- $\cos(\theta) \in (0, 1)$ – częściowe podobieństwo

TF-IDF (Term Frequency - Inverse Document Frequency)

W kontekście ekstrakcji słów kluczowych z opisów tekstowych stosuje się miarę TF-IDF [9]:

$$TF(t, d) = \frac{count(t, d)}{|d|} \quad (4)$$

gdzie $count(t, d)$ to liczba wystąpień terminu t w dokumencie d , a $|d|$ to długość dokumentu (liczba słów).

Pełna wersja TF-IDF uwzględnia rzadkość terminu w całym korpusie:

$$TF-IDF(t, d, D) = TF(t, d) \times IDF(t, D) \quad (5)$$

gdzie $IDF(t, D) = \log \frac{|D|}{|\{d \in D: t \in d\}|}$ to odwrócona częstość dokumentowa, penalizująca terminy występujące w wielu dokumentach.

Logika rozmyta – podstawy teoretyczne

Logika rozmyta (Fuzzy Logic) została wprowadzona przez Lotfi Zadeha w przełomowej pracy [2]. Rozszerza klasyczną logikę dwuwartościową (prawda/fałsz) o stopnie przynależności w przedziale $[0, 1]$.

Motywacja: Klasyczna logika wymaga precyzyjnych granic. Pytanie "Czy produkt za 450 PLN jest tani?" nie ma jednoznacznej odpowiedzi – zależy od kontekstu, kategorii produktu i preferencji użytkownika. Logika rozmyta pozwala odpowiedzieć: "Produkt jest tani ze stopniem 0.3 i średnio drogi ze stopniem 0.7".

Zbiory rozmyte (Fuzzy Sets): W klasycznej teorii zbiorów element należy lub nie należy do zbioru. W zbiorach rozmytych element ma stopień przynależności $\mu(x) \in [0, 1]$. Formalnie, zbiór rozmyty A na uniwersum X jest zdefiniowany przez funkcję przynależności:

$$\mu_A : X \rightarrow [0, 1] \quad (6)$$

gdzie $\mu_A(x)$ oznacza stopień przynależności elementu x do zbioru A .

Przykład: Dla zmiennej "cena" możemy zdefiniować trzy zbiory rozmyte:

- **cheap:** ceny niskie (pełna przynależność dla cen < 100 PLN)
- **medium:** ceny średnie (pełna przynależność dla cen 500-1200 PLN)
- **expensive:** ceny wysokie (pełna przynależność dla cen > 2000 PLN)

Produkt za 350 PLN może mieć: $\mu_{cheap}(350) = 0.3$, $\mu_{medium}(350) = 0.5$, $\mu_{expensive}(350) = 0.0$.

Funkcje przynależności (Membership Functions) definiują stopień przynależności elementu do zbioru rozmytego. Najczęściej stosowane typy:

Funkcja trójkątna (Triangular MF):

$$\mu_{triangle}(x; a, b, c) = \max \left(0, \min \left(\frac{x-a}{b-a}, \frac{c-x}{c-b} \right) \right) \quad (7)$$

gdzie a to dolna granica, b to punkt maksymalny ($\mu = 1$), c to górna granica.

Funkcja trapezoidalna (Trapezoidal MF):

$$\mu_{trapezoid}(x; a, b, c, d) = \max \left(0, \min \left(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c} \right) \right) \quad (8)$$

gdzie przedział $[b, c]$ ma pełną przynależność ($\mu = 1$), a $[a, b]$ i $(c, d]$ to obszary przejściowe.

Funkcja gaussowska (Gaussian MF):

$$\mu_{gaussian}(x; c, \sigma) = e^{-\frac{(x-c)^2}{2\sigma^2}} \quad (9)$$

gdzie c to środek (mean), a σ to odchylenie standardowe kontrolujące szerokość.

Operacje na zbiorach rozmytych

Uzupełnienie (Negacja):

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \quad (10)$$

Przecięcie (AND) – T-norma:

$$\mu_{A \cap B}(x) = T(\mu_A(x), \mu_B(x)) \quad (11)$$

Najczęściej używane T-normy:

- Minimum (Gödel): $T_{min}(a, b) = \min(a, b)$
- Iloczyn algebraiczny: $T_{prod}(a, b) = a \cdot b$
- Łukasiewicz: $T_L(a, b) = \max(0, a + b - 1)$

Suma (OR) – T-conorma (S-norma):

$$\mu_{A \cup B}(x) = S(\mu_A(x), \mu_B(x)) \quad (12)$$

Najczęściej używane T-conormy:

- Maksimum: $S_{max}(a, b) = \max(a, b)$
- Suma algebraiczna: $S_{sum}(a, b) = a + b - a \cdot b$
- Łukasiewicz: $S_L(a, b) = \min(1, a + b)$

System wnioskowania Mamdani [3] jest najbardziej rozpowszechnioną metodą wnioskowania rozmytego. Składa się z czterech etapów:

1. **Fuzzyfikacja** – przekształcenie wartości wejściowych na stopnie przynależności do zbiorów rozmytych. Przykład: cena 450 PLN $\rightarrow \mu_{cheap} = 0.1, \mu_{medium} = 0.6, \mu_{expensive} = 0.0$.
2. **Ewaluacja reguł** – obliczenie aktywacji reguł IF-THEN za pomocą T-norm. Dla reguły "IF price IS cheap AND quality IS high THEN recommendation IS strong":

$$\alpha = T(\mu_{cheap}(price), \mu_{high}(quality)) = \min(\mu_{cheap}, \mu_{high}) \quad (13)$$

3. **Agregacja** – połączenie wyników wszystkich reguł za pomocą T-conormy. Jeśli wiele reguł prowadzi do tego samego wniosku (następnika):

$$\mu_{output} = S(\alpha_1, \alpha_2, \dots, \alpha_n) = \max(\alpha_1, \alpha_2, \dots, \alpha_n) \quad (14)$$

4. **Defuzzyfikacja** – przekształcenie wyniku rozmytego na wartość liczbową.

Metody defuzzyfikacji:

Centroid (środek ciężkości):

$$y^* = \frac{\int y \cdot \mu(y) dy}{\int \mu(y) dy} \quad (15)$$

Średnia ważona (Weighted Average) – uproszczona metoda używana w implementacji:

$$y^* = \frac{\sum_{i=1}^n \alpha_i \cdot w_i}{\sum_{i=1}^n w_i} \quad (16)$$

gdzie α_i to aktywacja reguły i , a w_i to waga reguły.

Mean of Maximum (MoM):

$$y^* = \frac{1}{|M|} \sum_{y \in M} y, \quad M = \{y : \mu(y) = \max_z \mu(z)\} \quad (17)$$

Reguły rozmyte IF-THEN

Reguła rozmyta ma postać [10]:

$$\text{IF } x_1 \text{ IS } A_1 \text{ AND } x_2 \text{ IS } A_2 \text{ THEN } y \text{ IS } B \quad (18)$$

gdzie A_1, A_2, B to zbiory rozmyte definiujące poprzedniki (warunki) i następnik (wniosek) reguły. Operator AND realizowany jest przez T-normę, najczęściej minimum lub iloczyn algebraiczny.

Modele probabilistyczne – podstawy teoretyczne

Łańcuchy Markowa (Markov Chains) zostały wprowadzone przez Andrieja Markowa w 1906 roku. Są procesami stochastycznymi spełniającymi własność Markowa – przyszły stan zależy tylko od stanu obecnego, nie od historii [4].

Definicja formalna: Łańcuch Markowa to ciąg zmiennych losowych X_0, X_1, X_2, \dots przyjmujących wartości ze zbioru stanów $S = \{s_1, s_2, \dots, s_n\}$, spełniający własność Markowa:

$$P(X_{t+1} = s_j | X_t = s_i, X_{t-1} = s_{i-1}, \dots, X_0 = s_0) = P(X_{t+1} = s_j | X_t = s_i) \quad (19)$$

Oznacza to, że prawdopodobieństwo przejścia do stanu s_j zależy tylko od obecnego stanu s_i , nie od tego jak do niego dotarliśmy.

Macierz przejść (Transition Matrix) P zawiera prawdopodobieństwa przejść między stanami:

$$P_{ij} = P(X_{t+1} = s_j | X_t = s_i) \quad (20)$$

Macierz P spełnia warunki:

- $P_{ij} \geq 0$ dla wszystkich i, j
- $\sum_j P_{ij} = 1$ dla wszystkich i (wiersze sumują się do 1)

Estymacja prawdopodobieństw przejść z danych:

$$\hat{P}_{ij} = \frac{\text{count}(s_i \rightarrow s_j)}{\sum_k \text{count}(s_i \rightarrow s_k)} \quad (21)$$

gdzie $\text{count}(s_i \rightarrow s_j)$ to liczba obserwowanych przejść ze stanu s_i do stanu s_j .

Rozkład stacjonarny (Stationary Distribution) π spełnia:

$$\pi = \pi P, \quad \sum_i \pi_i = 1 \quad (22)$$

Jest to rozkład prawdopodobieństwa, który pozostaje niezmienny po przejściu – reprezentuje długoterminowe prawdopodobieństwa przebywania w każdym stanie. Rozkład stacjonarny jest istotny w analizie długoterminowego zachowania systemu.

Naiwny klasyfikator Bayesa (Naive Bayes, NB) opiera się na twierdzeniu Bayesa z założeniem niezależności cech [5].

Twierdzenie Bayesa:

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)} \quad (23)$$

gdzie:

- $P(C|X)$ – prawdopodobieństwo a posteriori klasy C przy cechach X
- $P(C)$ – prawdopodobieństwo a priori klasy C
- $P(X|C)$ – wiarygodność (likelihood) – prawdopodobieństwo obserwacji cech X w klasie C
- $P(X)$ – prawdopodobieństwo marginalne cech (stałe dla wszystkich klas)

Założenie naiwne (Naive assumption) – niezależność warunkowa cech:

$$P(X|C) = P(x_1, x_2, \dots, x_n|C) = \prod_{i=1}^n P(x_i|C) \quad (24)$$

Założenie to jest "naiwne" bo w rzeczywistości cechy są często skorelowane. Jednak Naive Bayes działa zaskakująco dobrze w praktyce.

Klasyfikacja:

$$\hat{C} = \arg \max_C P(C) \prod_{i=1}^n P(x_i|C) \quad (25)$$

Ponieważ $P(X)$ jest stałe dla wszystkich klas, można je pominąć przy porównywaniu.

Problem zerowych prawdopodobieństw: Jeśli cecha x_i nie wystąpiła w klasie C w danych treningowych, to $P(x_i|C) = 0$, co zeruje całe prawdopodobieństwo.

Wygładzanie Laplace’a (Laplace Smoothing / Add-one Smoothing) rozwiązuje ten problem:

$$P(x_i = v|C) = \frac{\text{count}(x_i = v, C) + 1}{\text{count}(C) + |V_i|} \quad (26)$$

gdzie $|V_i|$ to liczba unikalnych wartości cechy x_i . Dodanie 1 do licznika i $|V|$ do mianownika zapewnia, że żadne prawdopodobieństwo nie będzie zerowe.

Logarytm dla stabilności numerycznej: Iloczyn wielu małych prawdopodobieństw prowadzi do niedomiaru arytmetycznego (ang. *underflow* - sytuacja, gdy wartość

liczbowa jest zbyt mała aby można było ją reprezentować w pamięci komputera, co powoduje zaokrąglenie do zera). Rozwiązanie – praca w przestrzeni logarytmów:

$$\log P(C|X) = \log P(C) + \sum_{i=1}^n \log P(x_i|C) + \text{const} \quad (27)$$

Warianty *Naive Bayes*:

- **Multinomial NB** – dla danych wyliczeniowych (np. częstość słów)
- **Bernoulli NB** – dla cech binarnych (obecność/brak)
- **Gaussian NB** – dla cech ciągłych (zakłada rozkład normalny)

Naive Bayes jest szeroko stosowany w klasyfikacji tekstu, filtracji spamu oraz systemach rekomendacyjnych ze względu na prostotę implementacji i niskie wymagania obliczeniowe.

Metryki oceny systemów rekomendacyjnych

Ewaluacja systemów rekomendacyjnych wymaga odpowiednich metryk jakości. Najpopularniejsze:

Precision@K – jaka część top K rekomendacji była faktycznie kupiona/po-lubiona:

$$\text{Precision@K} = \frac{|Recommended@K \cap Relevant|}{K} \quad (28)$$

Recall@K – jaka część produktów istotnych dla użytkownika została trafiona:

$$\text{Recall@K} = \frac{|Recommended@K \cap Relevant|}{|Relevant|} \quad (29)$$

F1-Score – harmoniczna średnia Precision i Recall:

$$F1@K = 2 \cdot \frac{\text{Precision@K} \cdot \text{Recall@K}}{\text{Precision@K} + \text{Recall@K}} \quad (30)$$

Mean Reciprocal Rank (MRR) – pozycja pierwszego trafienia:

$$MRR = \frac{1}{|U|} \sum_{u \in U} \frac{1}{rank_u} \quad (31)$$

gdzie $rank_u$ to pozycja pierwszego istotnego produktu w rankingu dla użytkownika u .

Coverage – procent produktów, które system jest w stanie rekomendować:

$$\text{Coverage} = \frac{|\text{products with recommendations}|}{|\text{all products}|} \quad (32)$$

Rozdział 2

Weryfikacja i analiza rozwiązań alternatywnych

Decyzja o implementacji własnego silnika rekomendacyjnego została poprzedzona szczegółową analizą trzech wiodących platform dostępnych komercyjnie. Weryfikacja miała charakter eksploracyjny - chodziło o identyfikację barier funkcjonalnych i ekonomicznych, które skłoniłyby do inwestycji w rozwój autorskiego rozwiązania zamiast adopcji platformy zewnętrznej.

Amazon Personalize

Amazon Personalize stanowi zarządzaną usługę w infrastrukturze AWS, implementującą algorytmy rekomendacyjne stosowane przez platformę handlową Amazon.com. Architektura opiera się na sieciach neuronowych oraz filtracji kolaboratywnej, udostępniając trzy główne tryby działania: personalizację użytkownika, produkty pokrewne oraz spersonalizowane rankowanie list.

Możliwości:

- **User Personalization** - HRNN (Hierarchical Recurrent Neural Network) generuje rankingi na podstawie historii interakcji,
- **Similar Items** - algorytm SIMS analizuje współwystępowanie produktów, rozwiązując problem zimnego startu,
- **Personalized Ranking** - re-ranking list produktów według przewidywanej trafności,
- **Automatyka** - autodostrajanie hiperparametrów i skalowanie infrastruktury,
- **Integracja AWS** - natywne wsparcie Lambda, S3, EventBridge, Kinesis (latencja <100ms dla 99. percentyla).

Kluczowe ograniczenia względem planowanego systemu:

- **Struktura kosztowa** - model subskrypcyjny generuje stałe obciążenie budżetowe, proporcjonalne do skali ruchu. Dla platform o ograniczonych zasobach finansowych może stanowić barierę wejścia,
- **Zakres funkcjonalny** - platforma koncentruje się na filtracji kolaboratywnej bez natywnego wsparcia dla analizy tekstowej opinii, wnioskowania rozmytego czy modeli probabilistycznych (łańcuchy Markowa, klasyfikatory Bayesowskie). Implementacja tych komponentów wymaga rozbudowy o dodatkowe usługi AWS,

- **Uzależnienie infrastrukturalne** - głęboka integracja z ekosystemem AWS (S3, Lambda, EventBridge) tworzy bariery migracyjne. Przeniesienie do alternatywnego środowiska wymaga reimplementacji całości,
- **Transparentność algorytmiczna** - architektura czarnoskrzynkowa uniemożliwia modyfikację logiki decyzyjnej. Brak dostępu do wag cech, parametrów podobieństwa czy funkcji przynależności,
- **Próg danych wejściowych** - dokumentacja AWS wskazuje 25000 interakcji jako minimum dla adekwatnej jakości. Dla startupów w fazie zimnego startu oznacza to obniżoną trafność rekomendacji.

Google Recommendations AI (Vertex AI)

Google Recommendations AI stanowi ofertę platformy GCP, bazującą na sieciach neuronowych oraz wieloramiennych bandytach kontekstowych - klasie algorytmów optymalizujących balans między eksploracją nowych wariantów a eksploatacją sprawdzonych rozwiązań. Przeznaczenie obejmuje e-commerce, serwisy VOD oraz agregatory treści newsowych, z automatyczną detekcją wzorców sezonowych.

Możliwości:

- **Multi-Armed Contextual Bandits** - balans eksploracji (nowe rekomendacje) vs eksploatacji (sprawdzone rozwiązania),
- **Detekcja trendów** - automatyczne wykrywanie sezonowości i dostosowywanie wag,
- **Frequently Bought Together** - analiza koszyków (podobna do Apriori),
- **Cele biznesowe** - optymalizacja pod współczynnik klikalności (ang. *Click-Through Rate*, CTR), średnią wartość zamówienia (ang. *Average Order Value*, AOV), utrzymanie użytkowników (ang. *retention*) lub przychody (ang. *revenue*),
- **Google Analytics 4** - natywna integracja śledzenia e-commerce,
- **Cold start** - wykorzystanie metadanych i sygnałów kontekstowych.

Kluczowe ograniczenia względem planowanego systemu:

- **Model ekonomiczny** - wycena oparta na liczbie predykcji (ang. *pay-per-prediction*) skutkuje wzrostem kosztów proporcjonalnym do liczby zapytań. Dla platform o średnim i niskim ruchu może stanowić ekonomiczną barierę,
- **Zakres analityczny** - obecność mechanizmu "Frequently Bought Together" nie kompensuje braku wieloźródłowej analizy sentymentu, wnioskowania rozmytego oraz modeli sekwencyjnych (łańcuchy Markowa). Rozszerzenie o te komponenty wymaga dodatkowych integracji,
- **Skala docelowa** - architektura zoptymalizowana pod kątem platform o bardzo dużej skali (YouTube) wprowadza nadmiarową złożoność dla małych i średnich sklepów,
- **Interpretowalność** - dostęp do reprezentacji wektorowych (ang. *embeddings*) czy logiki sieciowej jest niemożliwy nawet dla administratorów Vertex AI. Brak możliwości analizy mechanizmów decyzyjnych utrudnia debugowanie i optymalizację. Kontrast względem transparentnych reguł IF-THEN logiki rozmytej.

Apache Mahout

Apache Mahout reprezentuje otwartoźródłowy framework implementujący klasyczne algorytmy filtracji kolaboratywnej i faktoryzacji macierzowej - ALS (metoda najmniejszych kwadratów na przemian) oraz SVD (rozkład według wartości osobliwych). Projekt zapoczątkowany w 2008 roku ewoluował w kierunku algorytmów rozproszonych na infrastrukturze Apache Spark.

Możliwości:

- **Licencja Apache 2.0** - open-source bez opłat, pełna modyfikowalność,
- **Collaborative Filtering z ALS** - faktoryzacja macierzy user-item, efektywna równoległość,
- **SVD i SVD++** - rozkład macierzy z implicit feedback (przeglądanie, czas na stronie),
- **Apache Spark** - przetwarzanie rozproszone, skalowanie dla milionów produktów i użytkowników,
- **Ekosystem Hadoop** - integracja z HDFS, Hive, HBase,

- **Modele probabilistyczne** - naiwny klasyfikator Bayesa (ang. *Naive Bayes*), regresja logistyczna (ang. *Logistic Regression*), lasy losowe (ang. *Random Forest*).

Istotne bariery wdrożeniowe:

- **Próg kompetencyjny** - uruchomienie wymaga skonfigurowania infrastruktury przetwarzania rozproszonego (klastery Apache Spark, zarządca zasobów YARN, systemy monitorowania). Badanie Stack Overflow Developer Survey 2023 wskazuje, że zaledwie niewielki odsetek deweloperów posiada kompetencje w zakresie technologii Spark,
- **Całkowite koszty posiadania** - licencja otwarta eliminuje opłaty licencyjne, lecz obsługa klastra generuje wydatki na serwery dedykowane, konfigurację integracji z pozostałymi komponentami systemu (warstwa REST, baza relacyjna, pamięć podręczna, interfejs użytkownika) oraz bieżące utrzymanie,
- **Braki funkcjonalne** - platforma nie dostarcza mechanizmów analizy sentymentu recenzji, wnioskowania rozmytego ani modeli probabilistycznych typu łańcuchy Markowa. Realizacja tych funkcji wymagałaby integracji bibliotek zewnętrznych (np. Stanford CoreNLP) lub implementacji od podstaw (klasyfikator sentymentu słownikowego, system Mamdani, modele Bayesowskie),
- **Spowolnienie rozwoju** - aktywność maintainerów znacząco spadła (z 20-30 commitów miesięcznie w latach 2012-2014 do 2-3 w okresie 2023-2024), co skutkuje lukami w dokumentacji aktualnych wersji.

Rozwiązania biblioteczne: Framework Surprise (Python) udostępnia algorytmy rozkładu macierzowego (SVD, SVD++, NMF) oraz metody sąsiedztwa (KNN) z wbudowanymi mechanizmami walidacji krzyżowej. Zakres funkcjonalny ogranicza się jednak wyłącznie do filtracji kolaboratywnej - brak obsługi metod treściowych, wnioskowania rozmytego, analizy sentymentu czy reguł asocjacyjnych.

Podsumowanie analizy i uzasadnienie własnego rozwiązania

Przeprowadzona weryfikacja rozwiązań dostępnych na rynku wykazała wyraźny dylemat decyzyjny: **dojrzałość technologiczna kontra autonomia implementacyjna i ekonomia projektu**. Usługi zarządzane (Amazon Personalize, Google Recommendations AI) zapewniają algorytmy oparte o uczenie głębokie i automatyczną optymalizację hiperparametrów, lecz generują znaczące obciążenie budżetowe, uzależnienie od ekosystemu dostawcy oraz nieprzejrzystość logiki decyzyjnej. Platforma Apache Mahout znosi bariery licencyjne, jednakże stawia wysokie wymagania wobec kompetencji zespołu oraz infrastruktury obliczeniowej (klastery Spark).

- **Synergiczne połączenie trzech metod rekomendacyjnych:**

Weryfikacja wykazała, że ani platformy chmurowe, ani rozwiązania biblioteczne nie udostępniają jednocześnie obsługi filtracji opartej na treści, wnioskowania rozmytego oraz modeli probabilistycznych:

- Amazon Personalize - wyłącznie filtracja kolaboratywna oraz modele głębokiego uczenia,
- Google Recommendations AI - brak mechanizmów opartych na treści, rozmytych i łańcuchów Markowa,
- Apache Mahout - koncentracja na metodach kolaboratywnych (ALS, SGD).

Architektura autorska integruje:

- Filtrację opartą na treści - eliminacja problemu zimnego startu dla produktów bez historii interakcji,
- Wnioskowanie rozmyte - adaptacja do preferencji użytkowników z zachowaniem przejrzystości mechanizmu decyzyjnego,
- Modele probabilistyczne (łańcuchy Markowa, klasyfikator Bayesowski) - prognozowanie sekwencji transakcyjnych.

- **Optymalizacja kosztowa dla segmentu MSP:**

Otwartoźródłowy stos technologiczny (Django + PostgreSQL + React) eliminuje opłaty licencyjne oraz koszty subskrypcyjne chmur, zapewniając pełną kontrolę nad infrastrukturą. Organizacje mogą samodzielnie wybierać środowisko wdrożeniowe (serwery dedykowane, VPS, chmury publiczne) i skalować koszty proporcjonalnie do faktycznego obciążenia.

- **Pełna kontrola parametryzacji i dostosowanie do specyfiki biznesowej:**

Implementacja własna pozwala na zastosowanie strategii niedostępnych w rozwiązaniach gotowych:

- **Wektoryzacja atrybutów z wagami** dla filtracji opartej na treści - alokacja wag: kategorie 40%, tagi 30%, przedział cenowy 20%, deskryptory tekstowe 10%,
- **Mechanizm wnioskowania Mamdaniego** dla logiki rozmytej - reprezentacja niepewności preferencyjnych poprzez funkcje przynależności i reguły rozmyte,
- **Proces Markowa pierwszego rzędu** - estymacja przyszłych kategorii transakcyjnych na bazie macierzy prawdopodobieństw przejścia.

- **Niezależność infrastrukturalna i przenośność:**

Stos Django + React + PostgreSQL zapewnia możliwość uruchomienia na dowolnym środowisku: chmury publiczne (AWS, GCP, Azure), infrastruktura on-premise, środowisko deweloperskie lokalne. Przeniesienie między platformami ogranicza się do rekonfiguracji parametrów połączeniowych - warstwa algorytmiczna pozostaje niemodyfikowana.

Kontrast: przejście pomiędzy usługami Amazon Personalize a Google Recommendations AI wymusza reimplementację całego interfejsu (strumienie zdarzeń, zbiory uczące, wywołania API), ponowne wytrenowanie modeli oraz okres degradacji jakości sugestii podczas migracji.

- **Wymiar dydaktyczny i przejrzystość decyzyjna:**

Charakter pracy inżynierskiej zakłada eksplorację badawczą oraz cel edukacyjny. Budowa algorytmów od podstaw (bez gotowych bibliotek uczenia maszynowego) umożliwia dogłębną analizę mechanizmów każdej metody, czego nie oferują usługi zarządzane działające w modelu black-box.

Wnioskowanie rozmyte oparte o reguły IF-THEN gwarantuje transparentność procesu decyzyjnego (konceptcja *explainable AI* - interpretowalnej sztucznej inteligencji), podczas gdy modele głębokiego uczenia stosowane przez Amazon i Google charakteryzują się nieinterpretowalnością.

Autorska architektura silnika rekomendacyjnego stanowi racjonalny wybór dla platform e-commerce o średniej skali operacyjnej, oferując następujące cechy charakterystyczne:

- Skuteczność predykcyjna (trzy metody uzupełniające się wzajemnie w różnych aspektach problemu),
- Kontrola parametryzacji algorytmów z możliwością adaptacji do kontekstu biznesowego,
- Minimalizacja wydatków bieżących (brak opłat subskrypcyjnych charakterystycznych dla rozwiązań chmurowych),
- Przejrzystość mechanizmów decyzyjnych umożliwiająca diagnostykę,
- Przenośność technologiczna (eliminacja uzależnienia od pojedynczego providera infrastruktury),
- Wartość poznawcza wynikająca z implementacji od podstaw.

Prezentowana architektura odpowiada szczególnie na potrzeby organizacji wymagających zaawansowanych funkcjonalności rekomendacyjnych w warunkach ograniczonego budżetu oraz konieczności dopasowania logiki do specyficznych wymagań domenowych.

Rozdział 3

Projekt systemu rekomendacyjnego

Rozdział omawia projekt platformy e-commerce wyposażonej w wielometodowy silnik rekomendacyjny. Przedstawiono specyfikację wymagań funkcjonalnych, model przypadków użycia obrazujący interakcje aktorów z aplikacją oraz strukturę architektoniczną systemu zrealizowanego w ramach projektu zespołowego.

3.1 Cel i zakres aplikacji

Aplikacja realizuje funkcjonalność sklepu internetowego z zaawansowanym systemem wspomagania decyzji zakupowych. Użytkownicy otrzymują spersonalizowane sugestie generowane przez trzy niezależne algorytmy: analizę podobieństwa produktowego, wnioskowanie z wykorzystaniem logiki rozmytej oraz predykcję probabilistyczną opartą na łańcuchach Markowa i klasyfikatorze Bayesowskim.

System obejmuje pełny cykl operacyjny platformy e-commerce: katalogowanie produktów z wielowymiarową strukturą atrybutów, zarządzanie użytkownikami z hierarchią uprawnień (gość, klient, administrator), obsługę transakcji z rejestracją historii zakupowej, wizualizację wyników działania algorytmów rekomendacyjnych oraz narzędzia diagnostyczne dla deweloperów i administratorów platformy.

3.2 Typy użytkowników systemu

System obsługuje trzy typy użytkowników zorganizowanych w hierarchiczną strukturę uprawnień:

1. Gość (użytkownik niezalogowany)

Gość może przeglądać katalog produktów, korzystać z wyszukiwarki, filtrować produkty według kategorii oraz dodawać produkty do koszyka. Dostęp do rekomendacji jest ograniczony do metody Content-Based Filtering na stronach szczegółów produktów (sekcja „Podobne produkty”). Gość nie może składać zamówień ani przeglądać historii zakupów. Aby sfinalizować zakup, musi się zarejestrować lub zalogować.

2. Klient (użytkownik zalogowany)

Klient dziedziczy wszystkie uprawnienia gościa oraz otrzymuje dodatkowe funkcjonalności: składanie zamówień, przeglądanie historii zamówień, zarządzanie kontem użytkownika (zmiana hasła, danych osobowych), dodawanie opinii i ocen produktów. Klient ma dostęp do panelu klienta z dedykowanymi sekcjami rekomendacji dla wszystkich trzech metod: Content-Based Filtering, Fuzzy Logic oraz Modele

Probabilistyczne. System buduje spersonalizowany profil użytkownika na podstawie historii zakupów, co umożliwia personalizację rekomendacji przez algorytm Fuzzy Logic (profil wrażliwości cenowej) oraz modele probabilistyczne (przewidywanie sekwencji zakupowych i prawdopodobieństwa odejścia klienta).

3. Administrator

Administrator dziedziczy wszystkie uprawnienia klienta oraz otrzymuje pełny dostęp do panelu administracyjnego. Funkcjonalności administracyjne obejmują: zarządzanie produktami (dodawanie, edycja, usuwanie), zarządzanie zamówieniami (zmiana statusów, przeglądanie szczegółów), zarządzanie użytkownikami (nadawanie/odbieranie uprawnień, blokowanie kont), generowanie macierzy podobieństw dla Content-Based Filtering, trening modeli probabilistycznych (łańcuch Markowa, Naive Bayes), debugowanie algorytmów rekomendacji (panele CBF Debug, Fuzzy Debug, Probabilistic Debug), konfiguracja parametrów algorytmów oraz dostęp do statystyk i analityki sprzedaży.

Hierarchia uprawnień zapewnia separację odpowiedzialności i bezpieczeństwo systemu. Użytkownicy mogą samodzielnie rejestrować się jako klienci, natomiast role administratora przyznawane są ręcznie przez istniejących administratorów w panelu zarządzania użytkownikami.

3.3 Wymagania funkcjonalne systemu

System został zaprojektowany z uwzględnieniem następujących grup wymagań funkcjonalnych zorganizowanych według obszarów funkcjonalności:

Autentykacja i autoryzacja

- Logowanie użytkowników z wykorzystaniem email i hasła (JWT - JSON Web Tokens),
- Rejestracja nowych użytkowników z walidacją danych wejściowych (unikalność email, minimalna długość hasła 8 znaków, obecność cyfry),
- Zarządzanie rolami użytkowników: Gość (niezalogowany), Klient (zalogowany), Administrator,
- Autoryzacja dostępu do zasobów na podstawie roli użytkownika (middleware weryfikujący JWT),
- Zarządzanie kontem użytkownika: edycja danych osobowych, zmiana hasła.

Zarządzanie katalogiem produktów

- Przeglądanie produktów z filtrowaniem według kategorii (dostępne dla: Gość, Klient, Administrator),
- Wyszukiwanie produktów z wykorzystaniem algorytmu odległości Levensteina (wyszukiwanie rozmyte tolerujące literówki i niepełne zapytania),
- Wyświetlanie szczegółów produktu: opis, specyfikacja techniczna, opinie użytkowników, średnia ocena,
- Dodawanie produktów do koszyka (Gość, Klient),
- Zarządzanie produktami przez administratora: dodawanie nowych produktów, edycja istniejących, usuwanie produktów, zarządzanie kategoriami i tagami (Administrator).

Obsługa zamówień

- Składanie zamówień - wymaga zalogowania, koszyk zachowuje zawartość po zalogowaniu poprzez mechanizm localStorage (Klient),
- Śledzenie statusu zamówień w czasie rzeczywistym: oczekujące, w realizacji, wysłane, dostarczone (Klient),
- Przeglądanie historii zamówień z możliwością filtrowania według daty i statusu (Klient),
- Zarządzanie zamówieniami przez administratora: przeglądanie wszystkich zamówień, zmiana statusów, anulowanie zamówień (Administrator),
- Dashboard z statystykami sprzedaży: przychód dzienny/miesięczny, najpopularniejsze produkty, współczynnik konwersji (Administrator).

System rekomendacji produktów

- Wyświetlanie rekomendacji Content-Based Filtering: produkty podobne na podstawie cech (kategoria, tagi, cena, słowa kluczowe) z wykorzystaniem podobieństwa kosinusowego (Klient),
- Wyświetlanie rekomendacji Fuzzy Logic: personalizacja z wykorzystaniem profilu rozmytego użytkownika, 6 reguł IF-THEN typu Mamdani, funkcje przynależności dla ceny/jakości/popularności (Klient),
- Wyświetlanie rekomendacji Probabilistic Models: predykcja sekwencji zakupowych (łańcuch Markova), prawdopodobieństwo zakupu i odejścia klienta (Naive Bayes) (Klient),

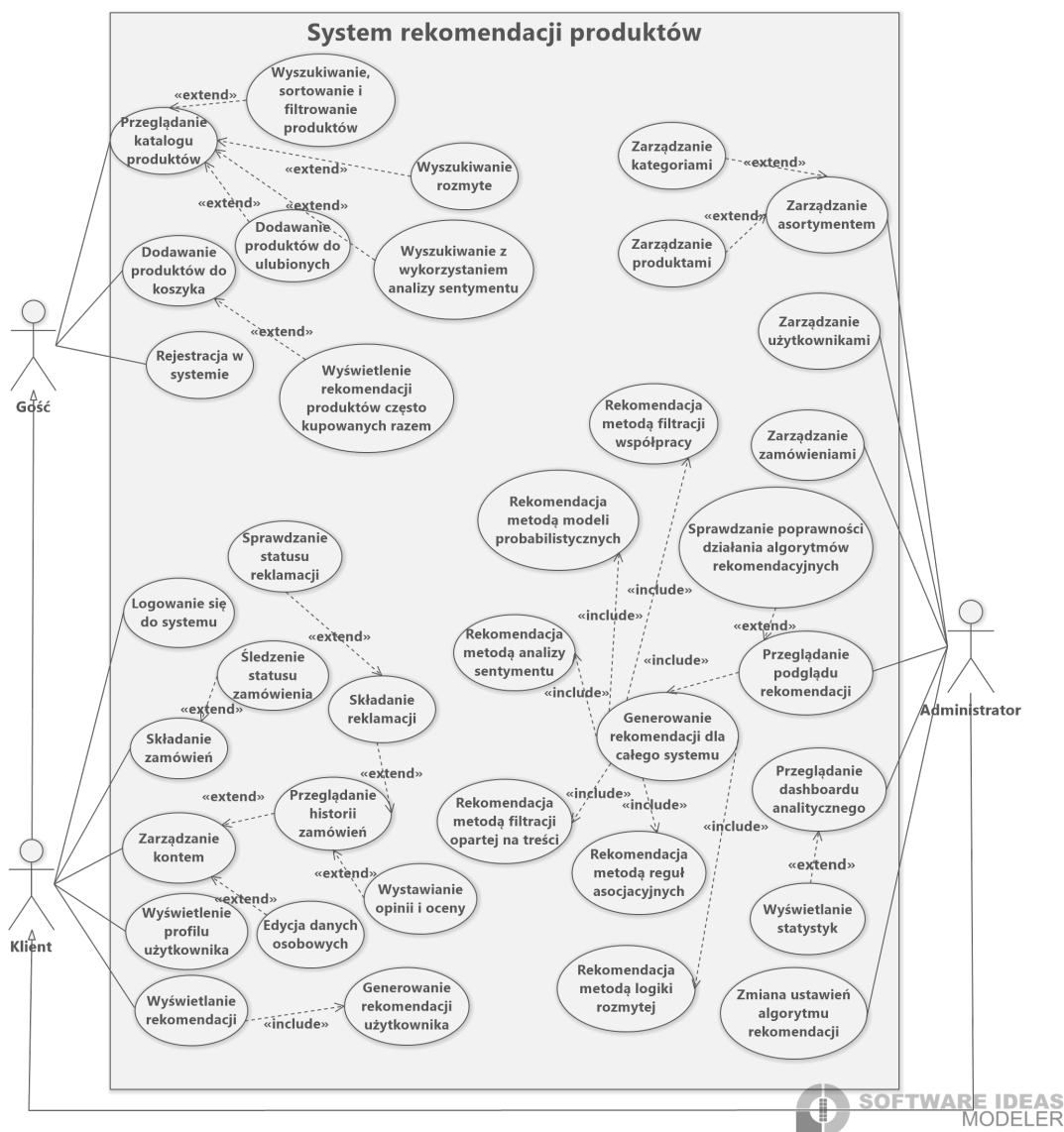
- Przeglądanie profilu użytkownika: preferencje kategoryjne, wrażliwość cenowa, ulubione tagi (Klient),
- Przełączanie między metodami rekomendacji w panelu administratora (Administrator).

Panel administracyjny i debugowanie

- Zarządzanie użytkownikami: przeglądanie listy użytkowników, zmiana ról, blokowanie kont (Administrator),
- Panele debugowania algorytmów rekomendacji (Administrator):
 - *Content-Based Filtering Debug* - macierz podobieństw produktów, wagi cech (kategorie 40%, tagi 30%, cena 20%, słowa kluczowe 10%), statystyki pokrycia rekomendacji,
 - *Fuzzy Logic Debug* - profile rozmyte użytkowników, funkcje przynależności, aktywacja reguł IF-THEN dla konkretnych produktów,
 - *Probabilistic Models Debug* - macierz przejść Markowa (48x48 kategorii), prawdopodobieństwa zakupu Naive Bayes, predykcja odejścia klienta,
- Generowanie macierzy podobieństw dla algorytmu CBF (ręczne wyzwalanie przeliczenia) (Administrator),
- Trening modeli probabilistycznych: aktualizacja łańcucha Markowa i klasyfikatorów Naive Bayes na bieżących danych (Administrator),

3.4 Diagram przypadków użycia

Diagram przypadków użycia (rys. 1) przedstawia kompletny widok funkcjonalności systemu oraz relacji między aktorami a przypadkami użycia. System obsługuje trzy główne typy aktorów: Gościa (użytkownik niezalogowany), Klienta (użytkownik zalogowany) oraz Administratora (zarządzający systemem). Relacje dziedziczenia między aktorami (Gość \rightarrow Klient \rightarrow Administrator) odzwierciedlają hierarchię uprawnień - każdy następny poziom dziedziczy wszystkie funkcjonalności poprzedniego i dodaje nowe, specyficzne dla swojej roli.



Rysunek 1: Diagram przypadków użycia systemu.

System został podzielony na trzy główne obszary funkcjonalne:

1. Obszar publiczny (dostępny dla wszystkich użytkowników) - podstawowe funkcjonalności e-commerce takie jak przeglądanie produktów, wyszukiwanie, dodawanie do koszyka oraz procesy autentykacji (logowanie, rejestracja).

2. Obszar klienta (wymaga zalogowania) - funkcjonalności transakcyjne obejmujące składanie zamówień, śledzenie ich statusu, zarządzanie kontem oraz dostęp do spersonalizowanych rekomendacji.

3. Obszar administracyjny (wymaga uprawnień administratora) - narzędzia do zarządzania całym systemem: produktami, zamówieniami, użytkownikami oraz dostęp do paneli statystycznych i debugowania algorytmów rekomendacji.

3.5 Architektura funkcjonalna systemu

System został zaprojektowany w architekturze warstwowej, gdzie każda warstwa odpowiada za konkretny aspekt funkcjonalności. Komunikacja między warstwami odbywa się poprzez RESTful API z uwierzytelnianiem JSON Web Tokens (JWT).

Warstwa prezentacji - interfejsy użytkownika dostosowane do ról (gość, klient, administrator):

- **Panel klienta** - dashboard z historią zamówień, sekcje rekomendacji (CBF, Fuzzy, Probabilistic), edycja profilu, śledzenie statusu zamówień,
- **Panel administracyjny** - zarządzanie produktami/zamówieniami/użytkownikami, statystyki sprzedaży, panele debugowania algorytmów rekomendacji.

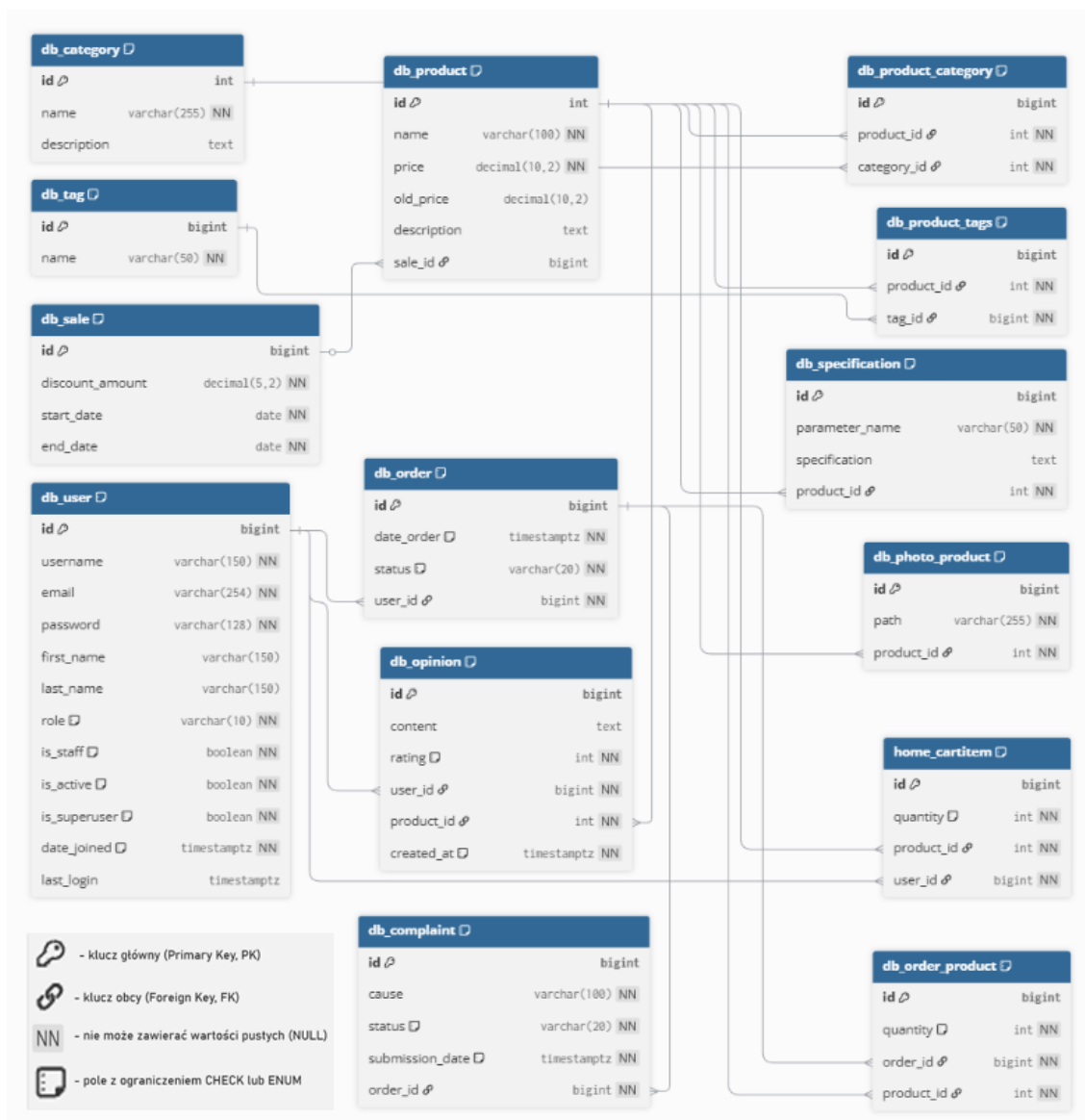
Warstwa logiki biznesowej - implementacja algorytmów rekomendacji oraz logiki e-commerce:

- **Moduły rekomendacyjne:**
 - *Moduł Content-Based Filtering* - generowanie macierzy podobieństwa produktów na podstawie cech (kategorie, tagi, cena, słowa kluczowe),
 - *Moduł Fuzzy Logic* - system wnioskowania Mamdani z funkcjami przynależności i regułami IF-THEN,
 - *Moduł Probabilistic Models* - łańcuch Markowa dla predykcji sekwencji zakupowych oraz Naive Bayes dla prawdopodobieństwa zakupu,
- **Logika transakcyjna** - składanie zamówień, zarządzanie statusami, walidacja danych, uwierzytelnianie JWT.

Warstwa danych - relacyjna baza danych PostgreSQL 14:

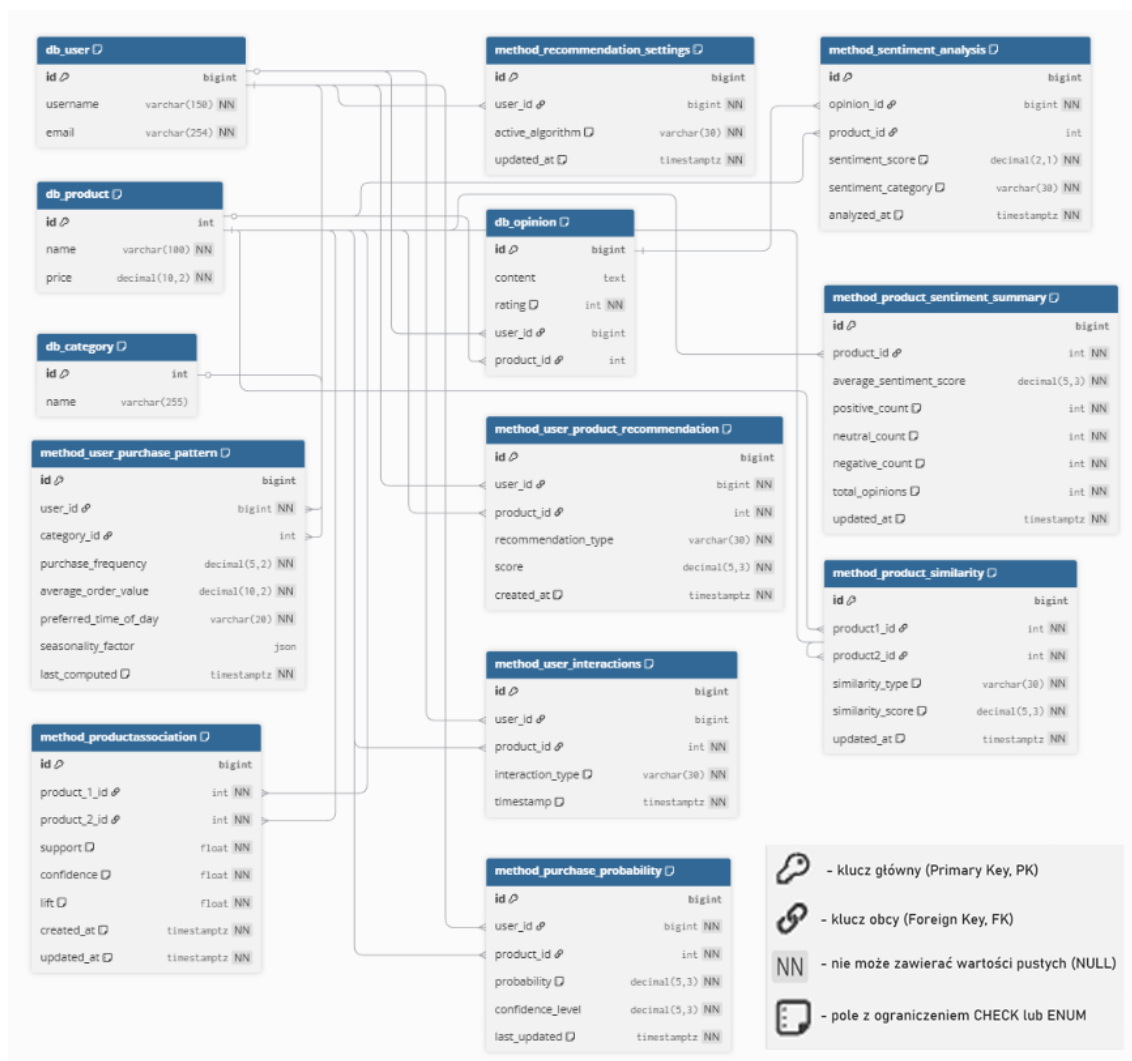
Struktura bazy danych

Baza składa się z **25 tabel** podzielonych na 4 moduły funkcjonalne. Poniżej przedstawiono diagramy ERD ilustrujące relacje między tabelami oraz szczegółową charakterystykę modułów.



Rysunek 2: Diagram ERD głównych tabel aplikacji.

Rysunek 2 przedstawia rdzeń aplikacji e-commerce. Diagram ilustruje podstawowe tabele systemu dla danych użytkowników, zamówień, produktów, kategorii produktów oraz opinii użytkowników.



Rysunek 3: Diagram ERD tabel metod rekomendacyjnych.

Diagram 3 pokazuje tabele algorytmów uczenia maszynowego i ich powiązania z głównymi tabelami aplikacji. Każda metoda rekomendacyjna (CBF, Fuzzy Logic, Probabilistic Models) posiada dedykowane tabele przechowujące wyniki obliczeń.

Szczegółowa charakterystyka modułów bazodanowych:

1. Moduł produktów i użytkowników (12 tabel):

- **db_product** - dane produktów (ID, nazwa, cena, opis),
- **db_category** - kategorie produktów z hierarchią,
- **db_product_category** - relacja Many-to-Many produktów i kategorii,
- **db_photo_product** - ścieżki do zdjęć produktów,
- **db_specification** - szczegółowe parametry techniczne produktów,

- `db_tag` - tagi do filtrowania produktów,
- `db_sale` - promocje i rabaty,
- `db_user` - konta użytkowników (role: admin/client),
- `db_order` - zamówienia z timestampami i statusami,
- `db_order_product` - produkty w zamówieniach (ilość, cena),
- `db_cart_item` - koszyk zakupowy przed finalizacją,
- `db_complaint` - reklamacje powiązane z zamówieniami.

2. Moduł opinii i analizy sentymentu (3 tabele):

- `db_opinion` - opinie użytkowników (treść, rating 1-5),
- `method_sentiment_analysis` - wyniki analizy sentymentu dla opinii,
- `method_product_sentiment_summary` - zagregowany sentyment produktu.

3. Moduł metod rekomendacji (5 tabel):

- `method_product_similarity` - macierz podobieństw produktów (Content-Based Filtering),
- `method_user_product_recommendation` - spersonalizowane rekomendacje użytkowników,
- `method_productassociation` - reguły asocjacyjne Apriori,
- `method_user_interactions` - historia interakcji użytkowników,
- `method_recommendation_settings` - konfiguracja algorytmów dla użytkownika.

4. Moduł analityczny i prognozowanie (5 tabel):

- `method_purchase_probability` - prawdopodobieństwo zakupu produktu przez użytkownika,
- `method_sales_forecast` - prognoza sprzedaży produktów,
- `method_user_purchase_pattern` - wzorce zakupowe użytkowników,
- `method_product_demand_forecast` - prognoza popytu i poziomy magazynowe,

- `method_risk_assessment` - ocena ryzyka dla użytkowników i produktów.

Wsad danych testowych

W celu umożliwienia testowania i walidacji algorytmów rekomendacyjnych system został zasilony danymi testowymi:

- **Produkty:** 500 produktów elektronicznych rozłożonych w 48 kategoriach (laptopy, smartfony, akcesoria, AGD, gaming) z pełnymi metadanymi (nazwa, opis, kategorie, tagi, cena, specyfikacje techniczne, zdjęcia),
- **Użytkownicy:** 20 kont testowych (5 administratorów, 15 klientów), z których każdy posiada wygenerowaną historię zakupową składającą się z 10 zamówień,
- **Zamówienia:** 200 zamówień zawierających łącznie około 600 pozycji produktowych (losowo 1-5 produktów na zamówienie), wygenerowanych z realistycznymi timestampami rozłożonymi w czasie 365 dni w celu umożliwienia analizy trendów i sezonowości,
- **Opinie:** Około 1750 opinii użytkowników z tekstowymi recenzjami i ocenami gwiazdkowymi (1-5), wygenerowanych dla różnych produktów w celu testowania analizy sentymentu,
- **Wyniki algorytmów:** Wstępnie wygenerowane macierze podobieństw (CBF), profile rozmyte użytkowników (Fuzzy Logic), macierze przejść kategorii (Markov Chain) oraz predykcje zakupowe (Naive Bayes) dla wszystkich użytkowników testowych.

Wszystkie migracje Django ORM zostały wygenerowane automatycznie na podstawie modeli Python i zarządzane przez system wersjonowania `django.db.migrations`. Dane testowe umożliwiają demonstrację wszystkich funkcjonalności systemu rekomendacyjnego oraz walidację poprawności algorytmów w realistycznych scenariuszach biznesowych.

Integracja warstw odbywa się poprzez RESTful API z automatyczną synchronizacją - zmiana danych w jednej warstwie propaguje aktualizacje do pozostałych. Django Signals zapewniają automatyczne przeliczanie rekomendacji przy dodaniu nowego produktu lub zamówienia.

Rozdział 4

Przedstawienie wykorzystanego stosu technologicznego oraz praktycznej realizacji projektu

Rozdział opisuje architekturę techniczną zrealizowanego systemu e-commerce wraz z modulem rekomendacyjnym. Przedstawiono wybór i uzasadnienie stosu technologicznego, strukturę warstwy backendowej (Django REST Framework) oraz frontendowej (React), model relacyjnej bazy danych PostgreSQL, a także mechanizmy konteneryzacji i wdrożenia aplikacji z wykorzystaniem Docker Compose.

4.1 Stos technologiczny

Aplikacja została zaprojektowana w architekturze klient-serwer opartej na technologiach Django (backend) oraz React (frontend). Komunikacja odbywa się poprzez RESTful API z uwierzytelnianiem tokenowym JSON Web Tokens (JWT). Struktura aplikacji wyraźnie rozdziela warstwę prezentacji (React SPA), logikę biznesową (widoki Django i serializery) oraz warstwę danych (PostgreSQL).

Główne założenia architektoniczne:

- **Separacja frontendu i backendu** - możliwość niezależnego rozwoju i skalowania obu warstw,
- **Podejście API-first (API-first approach)** - wszystkie funkcjonalności dostępne przez REST API,
- **Uwierzytelnianie bezstanowe (Stateless authentication)** - token JWT eliminuje potrzebę sesji po stronie serwera,
- **Modułowa struktura** - każdy algorytm rekomendacji stanowi niezależny moduł.

Backend: Django 5.1.4 (Python 3.11)

Django stanowi fundament aplikacji serwerowej, zapewniając architekturę MVC, system ORM (Object-Relational Mapping - mapowanie obiektowo-relacyjne) dla abstrakcji bazy danych oraz mechanizmy bezpieczeństwa (CSRF, XSS, SQL injection prevention). Architektura backendu opiera się na wzorcu Model-View-Serializer (MVS). Django zapewnia komponenty ORM, Signals oraz Middleware:

Kluczowe komponenty Django:

- **Django ORM** - mapowanie obiektowo-relacyjne umożliwiające operacje na bazie bez SQL,
- **Django Signals** - mechanizm automatycznej aktualizacji rekomendacji przy zmianach danych,
- **Django Middleware (oprogramowanie pośredniczące)** - obsługa CORS, uwierzytelnienie JWT, pamięć podręczna.

Django REST Framework 3.15.2

Rozszerza Django o funkcjonalności API RESTful:

- **Serializery** - konwersja obiektów Django na JSON z walidacją,
- **ViewSet (zestawy widoków)** - widoki implementujące operacje CRUD,
- **Uwierzytelnianie (Authentication)** - wsparcie dla JWT, uwierzytelnianie sesyjne,
- **Pagination (paginacja)** - automatyczne stronicowanie wyników.

Biblioteki uczenia maszynowego (ang. *Machine Learning*)

Do operacji numerycznych i obliczania podobieństw wykorzystano:

- **NumPy 1.24** - operacje macierzowe dla wektorów cech i macierzy podobieństwa w CBF i modelu probabilistycznym,
- **scikit-learn** - funkcja `cosine_similarity()` dla operacji macierzowych.

Frontend: React 18

Warstwa prezentacji została zrealizowana jako aplikacja jednostronicowa (Single Page Application - SPA) w technologii React 18. Kluczowe cechy wykorzystanej biblioteki:

- **Architektura komponentowa (Component-based)** - reużywalne komponenty UI,
- **Virtual DOM (wirtualny DOM)** - optymalizacja renderowania,
- **React Hooks** - `useState`, `useEffect`, `useContext`.

Biblioteki wspierające

- **React Router v6** - trasowanie (routing) dla aplikacji SPA,

- **Axios** - komunikacja z API, przechwytywacze JWT (interceptors),
- **Framer Motion** - płynne animacje,
- **Context API** - zarządzanie stanem (AuthContext, CartContext).

Baza danych: PostgreSQL 14

PostgreSQL 14 został wybrany jako system zarządzania bazą danych ze względu na następujące cechy:

- **Indeksowanie** - automatyczne tworzenie indeksów B-tree na kluczach głównych i obcych, dodatkowo zdefiniowane indeksy w trzech tabelach algorytmicznych (`method_sentiment_analysis`, `method_user_interactions`, `method_risk_assessment`) dla optymalizacji zapytań analitycznych,
- **Typ danych JSONB** - natywne przechowywanie struktur JSON (wykorzystane w tabeli `method_user_purchase_pattern` dla pola `seasonality_factor` przechowującego współczynniki sezonowości zakupów),
- **Transakcje ACID** - gwarancja atomowości, spójności, izolacji i trwałości operacji krytycznych (zamówienia, płatności),
- **Klucze obce i constrainty** - automatyczne wymuszanie integralności referencyjnej oraz walidacji danych (np. rating 1-5 w opiniach),
- **Optymalizacja JOIN** - wydajne łączenie tabel w złożonych zapytaniach rekomendacyjnych.

Szczegółowa charakterystyka struktury bazy danych oraz diagramy ERD zostały przedstawione w rozdziale 3.5 (Architektura funkcjonalna systemu).

4.2 Architektura praktyczna: komponenty systemu i wdrożenie

Po wyborze stosu technologicznego (Django, React, PostgreSQL, Docker) zaprojektowano szczegółową architekturę aplikacji, definiując strukturę modułów backendu i frontendu oraz mechanizmy wdrożenia. Niniejsza sekcja przedstawia system z punktu widzenia zrealizowanego rozwiązania, opisując jak poszczególne komponenty zostały zbudowane, jak ze sobą współpracują oraz jak aplikacja została wdrożona z wykorzystaniem konteneryzacji Docker.

Struktura backendu Django

Każdy komponent systemu rekomendacyjnego posiada dedykowane pliki realizujące zasadę separacji odpowiedzialności (Separation of Concerns):

- **models.py** – definicje modeli Django ORM reprezentujących tabele bazodanowe (Product, Order, Opinion, ProductSimilarity, FuzzyUserProfile, MarkovTransitions, PurchaseProbability, RecommendationSettings),
- **serializers.py** – klasy Django REST Framework wykonujące konwersję obiektów Django ↔ JSON z walidacją danych wejściowych,
- **views.py** – widoki API implementujące logikę biznesową dla operacji CRUD na produktach i zamówieniach,
- **custom_recommendation_engine.py** – implementacja algorytmów Content-Based Filtering, łańcucha Markowa oraz naiwnego klasyfikatora Bayesa,
- **fuzzy_logic_engine.py** – silnik logiki rozmytej z systemem wnioskowania Mamdaniego,
- **recommendation_views.py** – endpointy API dla CBF:
/api/content-based-debug/, /api/generate-similarities/,
- **fuzzy_views.py** – endpointy API dla Fuzzy Logic:
/api/fuzzy-recommendations/, /api/fuzzy-debug/,
- **probabilistic_views.py** – endpointy API dla modeli probabilistycznych:
/api/markov-recommendations/, /api/bayesian-insights/,
- **signals.py** – mechanizm automatycznej aktualizacji rekomendacji przy zmianach danych (nowe zamówienia, nowe produkty) wykorzystujący system sygnałów Django.

Struktura frontendu React

Aplikacja została zbudowana jako Single Page Application (SPA) z wykorzystaniem wzorców programowania funkcyjnego i React Hooks. Architektura składa się z modułowych komponentów odpowiedzialnych za poszczególne funkcjonalności:

- **App.js** – główny komponent zarządzający routingiem React Router v6 oraz globalnym stanem aplikacji poprzez Context API (AuthContext, CartContext),
- **Navbar.jsx** – responsywna nawigacja z wyszukiwarką produktów, linkami do kluczowych sekcji, przyciskami logowania i rejestracji oraz ikoną koszyka z licznikiem produktów,
- **SearchModal.jsx** – zaawansowany modal wyszukiwania z wykorzystaniem algorytmu odległości Levensteina dla wyszukiwania tolerującego błędy (fuzzy search),

- **ShopContent.jsx** – komponent wyświetlający katalog produktów z dynamicznym filtrowaniem (kategorie, zakres cen, oceny użytkowników) oraz gridową prezentacją kart produktów,
- **ProductPage.jsx** – szczegółowy widok produktu zawierający galerię zdjęć, opis, specyfikacje techniczne, sekcję opinii użytkowników oraz rekomendacje podobnych produktów,
- **CartContent.jsx** – koszyk zakupowy z możliwością modyfikacji ilości produktów oraz obliczania sumarycznej wartości zamówienia,
- **ClientPanel.jsx** – panel klienta z zakładkami: Dashboard (podsumowanie konta), My Orders (historia zamówień), Smart Recommendations (rekomendacje probabilistyczne), Fuzzy Logic (rekomendacje logiki rozmytej), Complaints (reklamacje), Account (ustawienia konta),
- **AdminPanel.jsx** – panel administracyjny z narzędziami zarządzania systemem oraz debugowania algorytmów uczenia maszynowego.

Mechanizmy optymalizacji systemu

System rekomendacyjny implementuje szereg mechanizmów optymalizacyjnych zapewniających wydajność i skalowalność dla katalogów liczących setki produktów i dziesiątki użytkowników.

1. Pamięć podręczna wyników (Django Cache Framework)

System wykorzystuje wbudowany mechanizm pamięci podręcznej Django z różnymi poziomami czasu wygaśnięcia:

- **CACHE_TIMEOUT_SHORT = 300s (5 minut)** - rekomendacje dla zalogowanych użytkowników, wyniki wyszukiwania tolerującego błędy
- **CACHE_TIMEOUT_MEDIUM = 1800s (30 minut)** - macierze podobieństw, reguły asocjacyjne Apriori
- **CACHE_TIMEOUT_LONG = 7200s (2 godziny)** - wygenerowane macierze CBF, modele probabilistyczne

Pamięć podręczna znacząco redukuje obciążenie bazy danych i przyspiesza odpowiedzi API. Przykładowo, pobranie rekomendacji CBF z pamięci (ang. *cache HIT* - trafienie) trwa 50-100ms, podczas gdy przy braku danych w pamięci (ang. *cache MISS* - chybienie) system musi wykonać zapytanie SQL i obliczyć podobieństwa, co zajmuje 5-10 sekund.

2. Operacje zbiorcze (bulk operations)

Zamiast pojedynczych zapytań INSERT dla każdego podobieństwa, system wykorzystuje metodę `bulk_create()` Django ORM do wstawiania rekordów wsadowo. Dla 4000 par produktów bulk insert jest **80x szybszy** niż pojedyncze inserty (5 sekund vs 400 sekund).

3. Prefetching i select_related

System minimalizuje liczbę zapytań SQL poprzez wczytywanie wyprzedzające (eager loading) powiązanych obiektów:

- `select_related()` - dla relacji ForeignKey (One-to-Many): łączy tabele przez JOIN
- `prefetch_related()` - dla relacji Many-to-Many: wykonuje dodatkowe zapytanie i łączy w Pythonie

Przykład: pobranie 100 produktów z kategoriami bez prefetchingu = 101 zapytań SQL (problem N+1). Z `prefetch_related('categories')` = 2 zapytania SQL.

4. Przycinanie progowe (threshold pruning)

Content-Based Filtering zapisuje do bazy danych tylko podobieństwa przekraczające próg 20% (`similarity_threshold = 0.2`):

- Bez pruning: $500 \times 499 / 2 = 124\,750$ par produktów
- Z pruning ($>20\%$): $\sim 4\,000$ zapisanych podobieństw
- **Redukcja o 97%** rozmiaru tabeli `ProductSimilarity`

Produkty o podobieństwie $<20\%$ są pomijane, ponieważ nie stanowią wartościowych rekomendacji dla użytkownika.

5. Ograniczenie liczby porównań per produkt

Algorytm CBF limituje liczbę obliczanych podobieństw do maksymalnie 50 najbardziej prawdopodobnych kandydatów per produkt (`max_comparisons_per_product = 50`). Zamiast porównywać produkt ze wszystkimi 499 innymi produktami, system:

1. Filtruje produkty z tej samej kategorii (znacznie mniejszy zbiór)
2. Sortuje według ceny (produkty o zbliżonej cenie są bardziej podobne)
3. Oblicza podobieństwo tylko dla top 50 kandydatów

6. Indeksowanie bazy danych

PostgreSQL automatycznie tworzy indeksy B-tree na:

- Klucze główne (PRIMARY KEY) - wszystkie tabele
- Klucze obce (FOREIGN KEY) - przyspieszenie operacji JOIN

Dodatkowo zdefiniowano indeksy w trzech tabelach algorytmicznych:

- `method_sentiment_analysis` - indeksy na polach `product` i `sentiment_category`
- `method_user_interactions` - indeksy na parze (`user`, `product`) oraz `interaction_type`
- `method_risk_assessment` - indeksy na parze (`entity_type`, `entity_id`) oraz `risk_type`



















7. Optymalizacje algorytmiczne

- **Rzadkie wektory (sparse vectors)** - CBF przechowuje tylko niezerowe cechy w słowniku zamiast pełnych wektorów (redukcja pamięci o 90%)
- **Wygładzanie Laplace'a** - Naive Bayes dodaje +1 do wszystkich liczników, zapobiegając zerowemu prawdopodobieństwu
- **Normalizacja** - Fuzzy Logic normalizuje wejścia do zakresu [0, 1]

Kombinacja pamięci podręcznej, operacji zbiorczych, przycinania progowego oraz indeksowania pozwala systemowi obsługiwać 500 produktów i generować rekomendacje w czasie rzeczywistym nawet przy braku danych w pamięci podręcznej. Dla większych katalogów (>10 000 produktów) zalecane są dalsze optymalizacje: przybliżone wyszukiwanie najbliższych sąsiadów (ang. *approximate nearest neighbors* - algorytmy LSH, HNSW), partycjonowanie tabel PostgreSQL, oraz rozproszenie obliczeń przy użyciu kolejki zadań asynchronicznych Celery z brokerem Redis.

Deployment i konteneryzacja Docker

Aplikacja została skonteneryzowana przy użyciu Docker Compose, zapewniając spójność środowiska między środowiskiem deweloperskim (development), testowym (staging) i produkcyjnym (production).

<input type="checkbox"/>	Name	Image	Port(s)	Last started	Actions
<input type="checkbox"/>	smartrecommender-project	-	-	3 minutes ago	  
<input type="checkbox"/>	SmartRecommender-React-FRONTEND	smartrecommender-project-frontend	3000:3000 	3 minutes ago	   
<input type="checkbox"/>	SmartRecommender-Django-BACKEND	smartrecommender-project-backend	8000:8000 	3 minutes ago	   
<input type="checkbox"/>	SmartRecommender-PostgreSQL-DB	postgres:16	5432:5432 	3 minutes ago	   

Rysunek 4: Deployment aplikacji w architekturze Docker Compose.

Architektura składa się z trzech kontenerów (rys. 4):

1. Kontener frontendu (React 18)

- Base image: node:18-alpine,
- Port: 3000,
- Volumes: montowanie src/ dla automatycznego przeładowania (hot-reload),
- Environment: REACT_APP_API_URL,
- Zależności (Dependencies): package.json (React, Axios, React Router, Framer Motion).

2. Kontener backendu (Django 5.1.4)

- Base image: python:3.11-slim,
- Port: 8000,
- Volumes: montowanie projektu dla automatycznego przeładowania (hot-reload), wolumen dla plików multimedialnych,
- Environment: DATABASE_URL, SECRET_KEY, DEBUG, ALLOWED_HOSTS,
- Zależności (Dependencies): requirements.txt (Django, DRF, psycpg2, NumPy, scikit-learn).

3. Kontener bazy danych (PostgreSQL 14)

- Base image: postgres:14-alpine,
- Port: 5432,
- Volumes: named volume postgres_data (persystencja danych),
- Environment: POSTGRES_DB, POSTGRES_USER, POSTGRES_PASSWORD,
- Healthcheck: pg_isready.

Wybór architektury Docker Compose zapewnia izolację serwisów (zero konfliktów zależności między kontenerami), przenośność środowiska (obraz zbudowany raz działa na dowolnym serwerze z silnikiem Docker), łatwą konfigurację (komenda `docker-compose up` uruchamia całą aplikację) oraz możliwość skalowania (uruchomienie wielu instancji backendu dla równoważenia obciążenia).

Rozdział 5

Implementacja algorytmów rekomendacji

Niniejszy rozdział opisuje szczegółowo implementację trzech metod rekomendacyjnych: Content-Based Filtering (CBF), logikę rozmytą (Fuzzy Logic) oraz modele probabilistyczne (Markov Chain + Naive Bayes). Każda metoda została zaimplementowana od podstaw bez użycia gotowych bibliotek rekomendacyjnych. Przedstawiono pseudokody algorytmów, wzory matematyczne oraz kluczowe aspekty techniczne implementacji. Praktyczne funkcjonowanie systemu oraz interfejsy użytkownika zostały przedstawione w rozdziale 6.

5.1 Content-Based Filtering

Architektura systemu CBF

System Content-Based Filtering został zaimplementowany w klasie `CustomContentBasedFilter` w pliku `custom_recommendation_engine.py`.

Architektura składa się z trzech głównych komponentów:

1. Ekstraktor cech (Feature Extractor) – odpowiada za budowę ważonego wektora cech dla każdego produktu. Analizuje cztery źródła danych:

- **Kategorie** (waga 40%) – główna klasyfikacja produktu. Format cechy: `category_{nazwa}`. Przykład: `category_Electronics`, `category_Laptops`.
- **Tagi** (waga 30%) – dodatkowe deskryptory (np. "Gaming", "Premium", "Budget"). Format: `tag_{nazwa}`.
- **Przedział cenowy** (waga 20%) – dyskretyzacja ceny: low (<100 PLN), medium (100-500), high (500-1500), premium (>1500).
- **Słowa kluczowe** (waga 10%) – top 10 słów z opisu produktu po filtracji stop-words.

2. Kalkulator podobieństwa – oblicza podobieństwo kosinusowe między wektorami cech produktów. Operuje na wektorach rzadkich dla efektywności.

3. Generator rekomendacji – zwraca top N produktów podobnych do danego produktu, z filtrowaniem według dostępności i progu podobieństwa.

Implementacja ekstrakcji cech

Metoda ekstrakcji cech buduje słownik cech z przypisanymi wagami. Algorytm w pseudokodzie:


```

1 FUNCTION extract_features(product):
2     features = empty_dictionary
3
4     FOR EACH category IN product.categories:
5         features["category_" + category.name] = 0.40
6
7     FOR EACH tag IN product.tags:
8         features["tag_" + tag.name] = 0.30
9
10    IF product.price < 100 THEN
11        features["price_low"] = 0.20
12    ELSE IF product.price < 500 THEN
13        features["price_medium"] = 0.20
14    ELSE IF product.price < 1500 THEN
15        features["price_high"] = 0.20
16    ELSE
17        features["price_premium"] = 0.20
18    END IF
19
20    keywords = extract_keywords(product.description)
21    FOR EACH word IN keywords[0:5]:
22        features["keyword_" + word] = 0.10 / length(
            keywords)
23
24    RETURN features
25 END FUNCTION

```

Ekstrakcja słów kluczowych:

Metoda `_extract_keywords()` przetwarza opis produktu:

1. Konwersja na małe litery
2. Usunięcie znaków interpunkcyjnych (regex)
3. Tokenizacja na słowa
4. Filtracja stop-words (zdefiniowana lista 200+ słów: "the", "and", "is", "a", "to", ...)
5. Filtracja słów krótszych niż 4 znaki
6. Zliczenie częstości (collections.Counter)
7. Wybór top 10 najczęstszych słów

Dyskretyzacja ceny:

Progi cenowe zostały dobrane empirycznie na podstawie rozkładu cen w katalogu:

- `price_low`: $\text{cena} < 100 \text{ PLN}$ – akcesoria, kable, drobne peryferia
- `price_medium`: $100 \text{ PLN} \leq \text{cena} < 500 \text{ PLN}$ – peryferia, komponenty
- `price_high`: $500 \text{ PLN} \leq \text{cena} < 1500 \text{ PLN}$ – monitory, karty graficzne
- `price_premium`: $\text{cena} \geq 1500 \text{ PLN}$ – laptopy, komputery, high-end

Dyskretyzacja eliminuje problem dużej wariancji cen i pozwala na porównywanie produktów z różnych kategorii cenowych.

Konfiguracja parametrów CBF

Parametry algorytmu Content-Based Filtering są definiowane w pliku `custom_recommendation_engine.py` w konstruktorze klasy `CustomContentBasedFilter`. System umożliwia dostosowanie poprzez modyfikację kodu źródłowego:

Wagi cech (słownik `self.feature_weights`):

- `'category'`: 0.40 – waga kategorii produktu (40%)
- `'tag'`: 0.30 – waga tagów (30%)
- `'price'`: 0.20 – waga przedziału cenowego (20%)
- `'keywords'`: 0.10 – waga słów kluczowych z opisu (10%)

Parametry optymalizacji:

- `self.similarity_threshold = 0.2` – minimalny próg podobieństwa do zapisu (20%)
- `self.max_comparisons_per_product = 50` – limit porównań na produkt
- `self.batch_size = 100` – rozmiar wsadu dla operacji zbiorczych
- `self.max_products_for_similarity = 500` – maksymalna liczba produktów

Progi cenowe (metoda `_get_price_category`):

- Wartości zagnieżdżone: 100, 500, 1500 PLN
- Kategorie: low (<100), medium (100-500), high (500-1500), premium (≥ 1500)

Wagi zostały dobrane empirycznie poprzez testowanie różnych konfiguracji na rzeczywistym katalogu produktów i wybór kombinacji zapewniającej najlepszą równowagę między precyzją (produkty są rzeczywiście podobne) a różnorodnością (nie tylko produkty identyczne). Zmiana parametrów wymaga modyfikacji kodu źródłowego i restartu serwera Django, po czym wymagane jest ponowne wygenerowanie macierzy podobieństw poprzez wywołanie endpointu
 POST /api/generate-similarities/.

Algorytm podobieństwa kosinusowego

Metoda `calculate_product_similarity()` implementuje podobieństwo kosinusowe dla wektorów rzadkich (sparse vectors):

$$\text{similarity}(p_1, p_2) = \frac{\sum_{f \in F_1 \cap F_2} w_1(f) \cdot w_2(f)}{\sqrt{\sum_{f \in F_1} w_1(f)^2} \cdot \sqrt{\sum_{f \in F_2} w_2(f)^2}} \quad (33)$$

gdzie F_1 , F_2 to zbiory cech produktów p_1 i p_2 , $w_i(f)$ to waga cechy f dla produktu p_i .

Implementacja dla wektorów rzadkich w pseudokodzie:

```

1 FUNCTION calculate_similarity(features1, features2):
2     common_features = intersection(keys(features1), keys(
3         features2))
4     dot_product = sum(features1[f] * features2[f] FOR f IN
5         common_features)
6
7     norm1 = sqrt(sum(v^2 FOR v IN values(features1)))
8     norm2 = sqrt(sum(v^2 FOR v IN values(features2)))
9
10    IF norm1 = 0 OR norm2 = 0 THEN
11        RETURN 0.0
12    END IF
13
14    RETURN dot_product / (norm1 * norm2)
15 END FUNCTION

```

Optymalizacja dla wektorów rzadkich:

Zamiast tworzyć pełne wektory o długości równej liczbie wszystkich możliwych cech (potencjalnie tysiące), algorytm operuje na słownikach. Iloczyn skalarny wymaga iteracji tylko po cechach wspólnych (przecięcie zbiorów kluczy).

Próg podobieństwa:

System zapisuje tylko podobieństwa większe niż 0.2 (20%). Uzasadnienie:

- Podobieństwo < 0.2 oznacza mniej niż 20% wspólnych cech – produkty są praktycznie różne
- Redukcja rozmiaru tabeli o 60-80%
- Szybsze zapytania (mniej rekordów do przeszukania)

Generowanie macierzy podobieństw

Metoda `generate_similarities_for_all_products()` oblicza podobieństwa dla wszystkich par produktów:

Etap 1: Prefetching danych

Wykorzystujemy mechanizm `prefetch_related()` frameworka Django dla kategorii, tagów i specyfikacji, redukując liczbę zapytań SQL z $O(n \times k)$ do $O(1)$ dla n produktów z k relacjami. Ta technika pobiera wszystkie powiązane obiekty w jednym zapytaniu SQL zamiast osobnego zapytania dla każdego produktu.

Etap 2: Ekstrakcja cech

Dla każdego produktu ekstrahujemy wektor cech i zapisujemy w słowniku, gdzie kluczem jest identyfikator produktu, a wartością jego wektor cech.

Etap 3: Obliczenie podobieństw

Dla każdej pary produktów (p_i, p_j) gdzie $i < j$ obliczamy podobieństwo kosinusowe. Algorytm w pseudokodzie:

```
1  FOR EACH product1 IN products:
2      FOR EACH product2 IN products[index(product1)+1:]:
3          similarity = calculate_similarity(features[product1
4              ], features[product2])
5
6          IF similarity > 0.2 THEN
7              save_similarity(product1, product2, similarity)
8              save_similarity(product2, product1, similarity)
9          END IF
10     END FOR
11 END FOR
```

Zapisywane są oba kierunki relacji (symetryczne), co umożliwia szybkie wyszukiwanie produktów podobnych do dowolnego produktu.

Etap 4: Bulk insert

Zapisywanie podobieństw odbywa się w partiach po 1000 rekordów przy użyciu mechanizmu `bulk_create()`, który przyspiesza zapis 50-100x względem pojedynczych operacji INSERT.

Etap 5: Cache

Wynik generowania macierzy podobieństw jest cachowany na 2 godziny (7200 sekund), eliminując potrzebę ponownego obliczania przy każdym żądaniu.

Złożoność obliczeniowa:

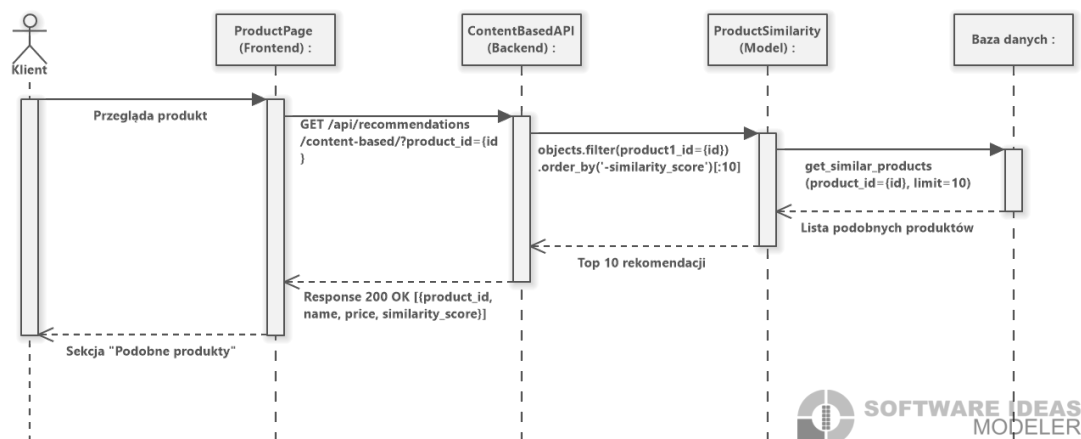
Teoretyczna złożoność to $O(n^2)$ dla n produktów (wszystkie pary). W praktyce ograniczamy liczbę porównań przez:

- `max_comparisons_per_product = 50` – dla każdego produktu obliczamy podobieństwo do max 50 innych
- Wczesne odrzucanie produktów bez wspólnych kategorii

Dla katalogu 500 produktów:

- Teoretycznie: $500 \times 499/2 = 124,750$ par
- Po optymalizacji: 25,000 obliczonych podobieństw
- Po filtrowaniu (próg 0.2): 4,000 zapisanych rekordów

Diagram sekwencji: Content-Based Filtering



Rysunek 5: Diagram sekwencji - Content-Based Filtering.

Proces generowania rekomendacji Content-Based Filtering rozpoczyna się w momencie gdy żądanie użytkownika trafia do endpointa API `/api/recommendations/content-based/?product_id={id}`. Backend, reprezentowany przez klasę `ContentBasedAPI`, wykonuje zapytanie do modelu

`ProductSimilarity`, pobierając top 10 produktów podobnych do wskazanego, posortowanych według malejącego współczynnika `similarity_score`. System sprawdza pamięć podręczną – w przypadku trafienia (cache HIT) wynik zwracany jest natychmiast, eliminując konieczność dostępu do bazy danych. Backend zwraca odpowiedź HTTP 200 OK z listą rekomendacji w formacie JSON, zawierającą dla każdego produktu: identyfikator, nazwę, cenę oraz współczynnik podobieństwa. Frontend (komponent React) wyświetla otrzymane rekomendacje w sekcji "Similar Products" na stronie produktu.

System automatycznie identyfikuje produkty z wysokim współczynnikiem podobieństwa (powyżej 20%) i zapisuje wyniki w pamięci podręcznej na 5 minut w celu optymalizacji wydajności.

5.2 Logika rozmyta w systemie rekomendacji

Architektura systemu Fuzzy Logic

System logiki rozmytej został zaimplementowany w module `fuzzy_logic_engine.py` i składa się z trzech klas:

1. FuzzyMembershipFunctions – definiuje funkcje przynależności dla trzech zmiennych wejściowych:

- **Cena** (price): cheap, medium, expensive – funkcje trójkątne i trapezoidalne z progami dostosowanymi do katalogu e-commerce
- **Jakość** (quality/rating): low, medium, high – bazuje na średniej ocenie produktu (1-5 gwiazdek)
- **Popularność** (popularity/view_count): low, medium, high – bazuje na liczbie zamówień produktu

2. FuzzyUserProfile – buduje rozmyty profil użytkownika na podstawie:

- Historii zakupów (dla zalogowanych użytkowników) – analiza kategorii, średniej ceny
- Danych sesji (dla gości) – ostatnio przeglądane produkty
- Profilu domyślnego (fallback) – gdy brak danych

3. SimpleFuzzyInference – silnik wnioskowania Mamdani z 6 regułami IF-THEN i metodą defuzyfikacji średniej ważonej.

Funkcje przynależności – szczegóły implementacji

Funkcje przynależności dla ceny

Klasa `FuzzyMembershipFunctions` definiuje trzy funkcje dla zmiennej "cena":

Funkcja "cheap" (tania) – trójkątna/trapezoidalna:

$$\mu_{cheap}(price) = \begin{cases} 1.0 & \text{jeśli } price \leq 100 \\ \frac{500-price}{400} & \text{jeśli } 100 < price < 500 \\ 0.0 & \text{jeśli } price \geq 500 \end{cases} \quad (34)$$

Interpretacja: Produkty poniżej 100 PLN są w pełni "tanie". Od 100 do 500 PLN stopień "taności" maleje liniowo.

Funkcja "medium" (średnia) – trapezoidalna:

$$\mu_{medium}(price) = \begin{cases} 0.0 & \text{jeśli } price < 300 \\ \frac{price-300}{200} & \text{jeśli } 300 \leq price < 500 \\ 1.0 & \text{jeśli } 500 \leq price \leq 1200 \\ \frac{1500-price}{300} & \text{jeśli } 1200 < price < 1500 \\ 0.0 & \text{jeśli } price \geq 1500 \end{cases} \quad (35)$$

Interpretacja: Przedział [500, 1200] ma pełną przynależność. Przejścia są płynne – cena 400 PLN jest częściowo "tania" i częściowo "średnia".

Funkcja "expensive" (droga):

$$\mu_{expensive}(price) = \begin{cases} 0.0 & \text{jeśli } price \leq 1000 \\ \frac{price-1000}{1000} & \text{jeśli } 1000 < price < 2000 \\ 1.0 & \text{jeśli } price \geq 2000 \end{cases} \quad (36)$$

Funkcje przynależności dla jakości (rating)

Oparte na średniej ocenie produktu (skala 1-5):

$$\mu_{low}(r) = \begin{cases} 1.0, & r \leq 2.5 \\ \frac{3.5-r}{3.5-2.5}, & 2.5 < r < 3.5 \\ 0.0, & r \geq 3.5 \end{cases} \quad (37)$$

$$\mu_{\text{medium}}(r) = \begin{cases} 0.0, & r < 2.5 \\ \frac{r-2.5}{3.5-2.5}, & 2.5 \leq r < 3.5 \\ 1.0, & r = 3.5 \\ \frac{4.5-r}{4.5-3.5}, & 3.5 < r < 4.5 \\ 0.0, & r \geq 4.5 \end{cases} \quad (38)$$

$$\mu_{\text{high}}(r) = \begin{cases} 0.0, & r < 3.5 \\ \frac{r-3.5}{4.5-3.5}, & 3.5 \leq r < 4.5 \\ 1.0, & r \geq 4.5 \end{cases} \quad (39)$$

Interpretacja: Produkty z oceną ≥ 4.5 (maksymalna jakość) mają pełną przynależność do zbioru "high". Ocena 3.5 jest punktem środkowym - produkt należy w równym stopniu do zbiorów rozmytych o sąsiednich kategoriach jakości. Przejścia są płynne, co odzwierciedla niepewność i subiektywność w ocenach użytkowników - produkt z oceną 4.0 jest częściowo "średni" i częściowo "wysoki".

Funkcje przynależności dla popularności (order_count)

Oparte na liczbie zamówień produktu:

$$\mu_{\text{low}}(v) = \begin{cases} 1.0, & v \leq 2 \\ \frac{10-v}{10-2}, & 2 < v < 10 \\ 0.0, & v \geq 10 \end{cases} \quad (40)$$

$$\mu_{\text{medium}}(v) = \begin{cases} 0.0, & v < 2 \\ 1.0, & 2 \leq v \leq 10 \\ \frac{30-v}{30-10}, & 10 < v < 30 \\ 0.0, & v \geq 30 \end{cases} \quad (41)$$

$$\mu_{\text{high}}(v) = \begin{cases} 0.0, & v < 10 \\ \frac{v-10}{30-10}, & 10 \leq v < 30 \\ 1.0, & v \geq 30 \end{cases} \quad (42)$$

Interpretacja: Funkcja trapezoidalna dla "medium" zapewnia stabilny przedział pełnej przynależności $[2, 10]$ zamówień - produkty w tym przedziale są typowymi artykułami o standardowej popularności. Produkt z 5 zamówieniami jest w pełni "średnio popularny", ale nie jest ani "niszowy" ($\text{low} \leq 2$) ani "bestsellerem" ($\text{high} \geq 30$). Trapez eliminuje oscylacje klasyfikacji dla produktów o podobnej

liczbie zamówień, w przeciwieństwie do funkcji trójkątnych używanych dla zbiorów skrajnych.

Konfiguracja parametrów Fuzzy Logic

Parametry systemu rozmytego są definiowane w pliku `fuzzy_logic_engine.py` w klasie `FuzzyMembershipFunctions`. System umożliwia dostosowanie progów funkcji przynależności poprzez modyfikację atrybutów instancji w metodzie `__init__()`:

Progi dla ceny (PLN):

- `self.price_low = 100` – produkt tani: pełna przynależność ≤ 100 PLN
- `self.price_mid_low = 500` – próg spadku dla tanich produktów
- `self.price_mid = 700` – środek przedziału średniej ceny
- `self.price_mid_high = 1200` – górny próg średniej ceny
- `self.price_high = 2000` – produkt drogi: pełna przynależność ≥ 2000 PLN

Progi dla jakości (rating 1-5):

- `self.rating_low = 2.5` – niska jakość: pełna przynależność ≤ 2.5
- `self.rating_mid = 3.5` – średnia jakość: maksimum przy 3.5
- `self.rating_high = 4.5` – wysoka jakość: pełna przynależność ≥ 4.5

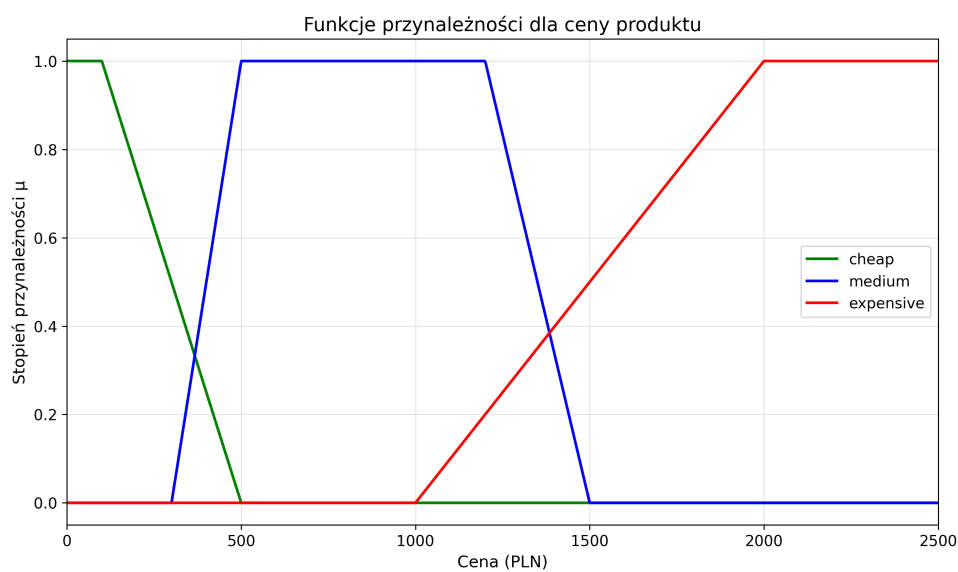
Progi dla popularności (liczba zamówień):

- `self.pop_low = 2` – niska popularność: pełna przynależność ≤ 2 zamówienia
- `self.pop_mid = 10` – średnia popularność: środek przedziału
- `self.pop_high = 30` – wysoka popularność: pełna przynależność ≥ 30 zamówień

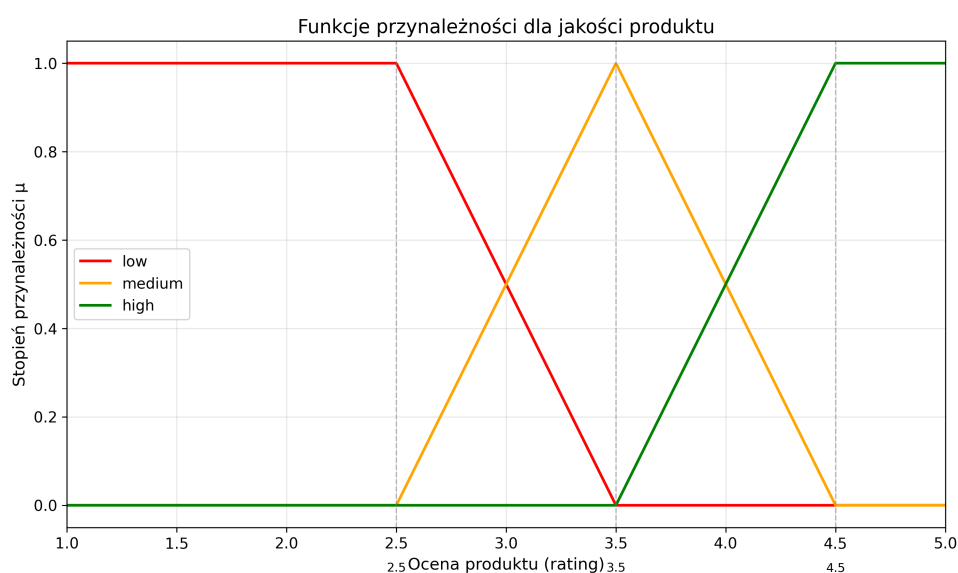
Progi zostały dobrane empirycznie na podstawie statystyk rzeczywistego katalogu (rozkłady cen, ocen, zamówień). Zmiana parametrów nie wymaga regeneracji danych – system oblicza rekomendacje w locie, więc nowe progi obowiązują natychmiast po zmianie kodu i restarcie serwera Django. Administrator może dostosować wartości do specyfiki swojego katalogu produktów poprzez edycję atrybutów w konstruktorze klasy `FuzzyMembershipFunctions`.

Wizualizacja funkcji przynależności

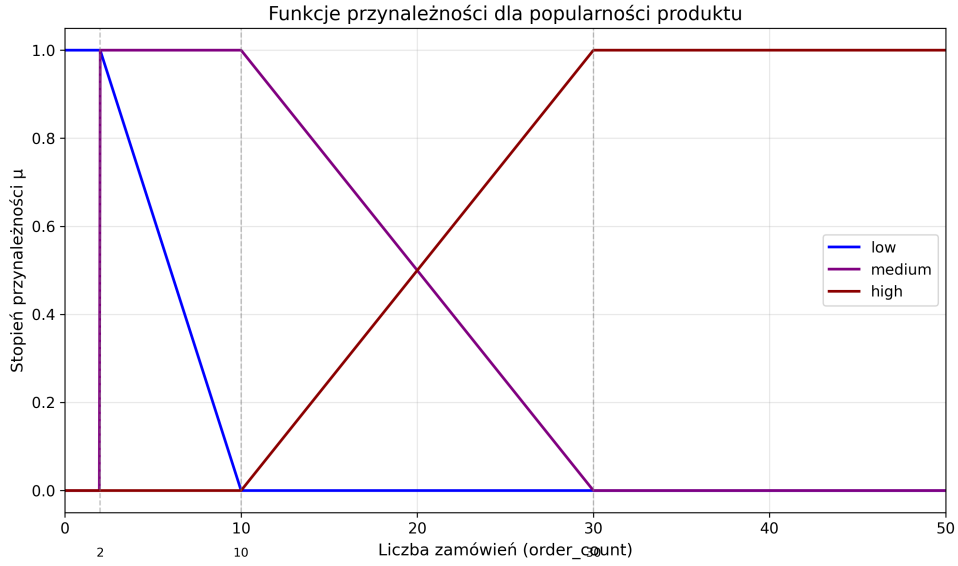
Poniższe wykresy przedstawiają graficzną reprezentację zaimplementowanych funkcji przynależności dla trzech zmiennych wejściowych systemu rozmytego.



Rysunek 6: Funkcje przynależności dla ceny produktu (cheap, medium, expensive).



Rysunek 7: Funkcje przynależności dla jakości produktu (low, medium, high).



Rysunek 8: Funkcje przynależności dla popularności produktu (low, medium, high).

Rozmyty profil użytkownika

Klasa `FuzzyUserProfile` buduje profil preferencji użytkownika jako zbiory rozmyte. Jest to kluczowy element personalizacji rekomendacji.

Dla zalogowanych użytkowników:

1. Pobranie historii zamówień z `prefetch_related` dla powiązanych produktów i kategorii
2. Zliczenie kategorii produktów w zamówieniach
3. Obliczenie stopnia zainteresowania kategorią:

$$\mu_{category} = \frac{count_{category}}{total_items} \quad (43)$$

4. Obliczenie wrażliwości cenowej na podstawie średniej ceny zakupów

Wrażliwość cenowa (`price_sensitivity`):

$$price_sensitivity = \begin{cases} 0.9 & \text{jeśli } avg_price < 300 \text{ PLN (bardzo wrażliwy)} \\ 0.6 & \text{jeśli } 300 \leq avg_price < 700 \text{ (średnio wrażliwy)} \\ 0.4 & \text{jeśli } 700 \leq avg_price < 1500 \text{ (mało wrażliwy)} \\ 0.2 & \text{jeśli } avg_price \geq 1500 \text{ PLN (premium)} \end{cases} \quad (44)$$

Użytkownik kupujący średnio tanie produkty ($avg < 300$ PLN) ma wysoką wrażliwość cenową (0.9) – system będzie promował tanie produkty. Użytkownik

premium (avg > 1500 PLN) ma niską wrażliwość (0.2) – system może rekomendować droższe produkty.

Dopasowanie kategorii – metoda `fuzzy_category_match()`:

Dla każdej kategorii produktu system oblicza stopień dopasowania do profilu użytkownika:

$$\text{match} = 0.6 \cdot \text{similarity}(\text{cat}_{\text{user}}, \text{cat}_{\text{product}}) + 0.4 \cdot \mu_{\text{interest}}(\text{cat}_{\text{user}}) \quad (45)$$

gdzie `similarity` używa hierarchii kategorii. Przykład:

- Kategoria użytkownika: "Electronics.Laptops"
- Kategoria produktu: "Electronics.Monitors"
- Podobieństwo hierarchiczne: 0.7 (wspólna kategoria nadrzędna "Electronics")

Profil domyślny (dla gości/nowych użytkowników):

Dla użytkowników bez historii zakupów system stosuje neutralny profil domyślny:

- `price_sensitivity` = 0.5 (neutralna wrażliwość cenowa)
- `category_preferences` = {} (brak preferencji kategorii)
- `quality_preference` = 0.7 (preferuje dobrą jakość)
- `popularity_preference` = 0.5 (neutralna wobec popularności)

Baza reguł rozmytych

System wykorzystuje 6 reguł rozmytych typu Mamdani. Każda reguła ma formę IF-THEN z przypisaną wagą określającą jej ważność:

R1: High Quality Bargain (waga: 0.9)

```
1 IF quality IS high AND (price IS cheap OR price IS medium)
2 THEN recommendation IS strong
```

Logika: Wysokiej jakości produkt w rozsądnej cenie to doskonała okazja. Najwyższa waga – ta reguła najsilniej wpływa na wynik.

R2: Popular in Category (waga: 0.7)

```
1 IF category_match IS high AND (popularity IS medium OR
   popularity IS high)
2 THEN recommendation IS medium-high
```

Logika: Popularny produkt z kategorii interesującej użytkownika. Popularność = walidacja społeczna.

R3: Price Sensitive Match (waga: 0.6)

```
1 IF user.price_sensitivity > 0.6 AND price IS cheap
2 THEN recommendation IS moderate
```

Logika: Dla użytkowników wrażliwych cenowo (kupujących tanie produkty) promuj tanie opcje.

R4: Category Quality Match (waga: 0.85)

```
1 IF category_match IS high AND (quality IS medium OR quality
   IS high)
2 THEN recommendation IS strong
```

Logika: Dopasowanie do kategorii + dobra jakość. Wysoka waga – dopasowanie kategorii jest istotne.

R5: Premium Match (waga: 0.8)

```
1 IF user.price_sensitivity < 0.4 AND price IS expensive AND
   quality IS high
2 THEN recommendation IS strong
```

Logika: Dla użytkowników premium (nieczułych cenowo) promuj drogie produkty wysokiej jakości.

R6: Quality-Price Balance (waga: 0.75)

```
1 IF (quality IS high AND price IS reasonable) OR
2   (quality IS medium AND price IS cheap)
3 THEN recommendation IS moderate
```

Logika: Dobry stosunek jakości do ceny – "value for money".

Wnioskowanie i defuzyfikacja

Metoda `evaluate_product()` implementuje pełny cykl wnioskowania Mamdani:

Krok 1: Fuzyfikacja

Dla każdej zmiennej wejściowej (cena, jakość, popularność) obliczane są stopnie przynależności do wszystkich zbiorów rozmytych. Wynikiem jest słownik zawierający 9 wartości przynależności:

- Cena: μ_{cheap} , μ_{medium} , $\mu_{expensive}$
- Jakość: μ_{low} , μ_{medium} , μ_{high}

- Popularność: μ_{low} , μ_{medium} , μ_{high}

Krok 2: Ewaluacja reguł

Każda reguła jest ewaluowana za pomocą T-normy (minimum) dla operatora AND i T-conormy (maksimum) dla OR. Zgodnie z teorią zbiorów rozmytych [2]:

$$\alpha_{R1} = \min(\mu_{quality_high}, \max(\mu_{price_cheap}, \mu_{price_medium})) \cdot w_{R1} \quad (46)$$

Dla każdej z 6 reguł obliczana jest jej aktywacja α_i poprzez aplikację odpowiednich operatorów rozmytych do wartości przynależności.

Krok 3: Agregacja

Wyniki reguł są agregowane. W uproszczonej implementacji wykorzystana została suma ważona (zamiast pełnej agregacji Mamdani):

$$\text{aggregated} = \sum_{i=1}^6 \alpha_i \quad (47)$$

Krok 4: Defuzyfikacja

System używa uproszczonej metody średniej ważonej:

$$\text{fuzzy_score} = \frac{\sum_{i=1}^6 \alpha_i \cdot w_i}{\sum_{i=1}^6 w_i} \quad (48)$$

gdzie α_i to aktywacja reguły i , a w_i to waga reguły.

Wagi reguł wynoszą: $w_{R1} = 0.9$, $w_{R2} = 0.7$, $w_{R3} = 0.6$, $w_{R4} = 0.85$, $w_{R5} = 0.8$, $w_{R6} = 0.75$. Suma wag wynosi 4.65, co zapewnia normalizację wyniku do przedziału $[0, 1]$.

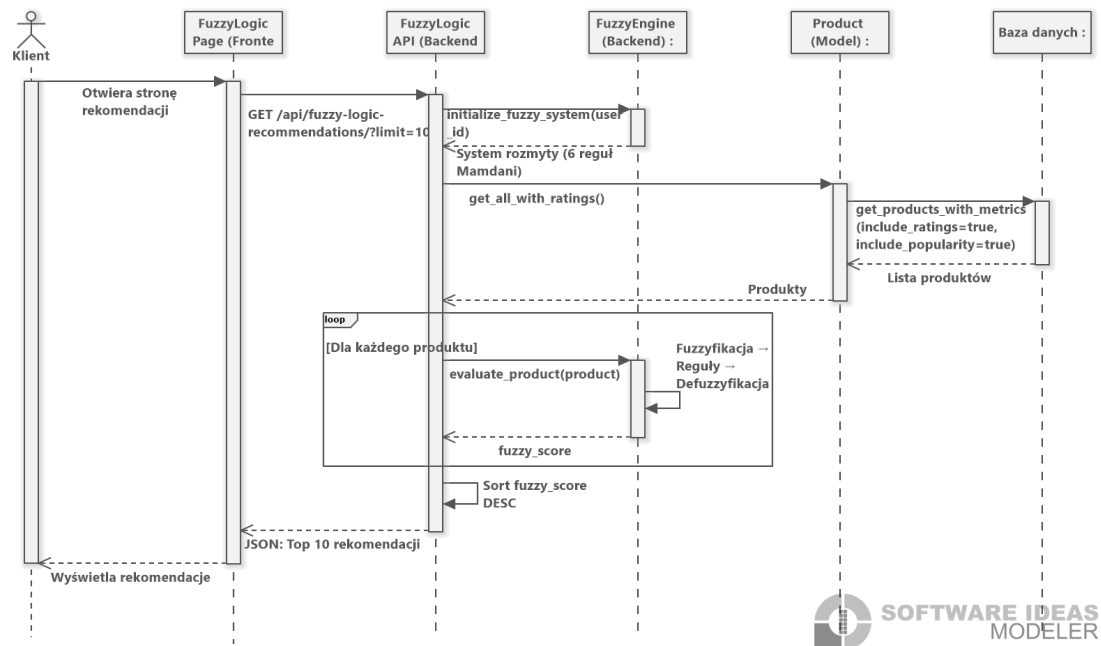
Wynik końcowy:

Metoda zwraca słownik z:

- `fuzzy_score` – wartość z przedziału $[0, 1]$ reprezentująca siłę rekomendacji
- `rule_activations` – słownik z aktywacją każdej reguły (dla debugowania)
- `category_match` – stopień dopasowania kategorii
- `price_membership` – przynależności cenowe (cheap, medium, expensive)

Algorytm Fuzzy Logic oferuje najwyższą interpretowalność - użytkownik może zobaczyć aktywację każdej reguły i zrozumieć, dlaczego produkt został polecony.

Diagram sekwencji: Fuzzy Logic



Rysunek 9: Diagram sekwencji - Fuzzy Logic.

Proces generowania rekomendacji Fuzzy Logic rozpoczyna się gdy żądanie użytkownika trafia do endpointa API `/api/fuzzy-recommendations/`. Silnik logiki rozmytej (`FuzzyLogicEngine`) buduje rozmyty profil użytkownika (`FuzzyUserProfile`) na podstawie historii zakupów, ekstrahując preferencje cenowe, kategoryjne oraz jakościowe. Dla każdego produktu w katalogu system przeprowadza proces fuzyfikacji, obliczając stopnie przynależności do zbiorów rozmytych dla trzech wymiarów: ceny (*cheap/medium/expensive*), jakości (*low/medium/high*) oraz popularności (*low/medium/high*). Następnie system wykonuje wnioskowanie rozmyte – ewaluuje 6 reguł IF-THEN zdefiniowanych w bazie wiedzy i oblicza aktywację każdej reguły na podstawie T-norm (minimum). Proces defuzyfikacji agreguje wyniki wszystkich aktywnych reguł do jednej wartości `fuzzy_score` metodą średniej ważonej. Backend zwraca listę produktów posortowanych według malejącego `fuzzy_score`, wraz z dodatkowymi metadanymi `rule_activations` zapewniającymi transparentność decyzji algorytmu. System buforuje wygenerowane rekomendacje w pamięci podręcznej na 5 minut w celu optymalizacji wydajności zapytań powtarzalnych.

5.3 Modele probabilistyczne – Markov Chain i Naive Bayes

Architektura systemu probabilistycznego

System probabilistyczny składa się z trzech komponentów zaimplementowanych w `custom_recommendation_engine.py`:

1. CustomMarkovChain – łańcuch Markowa pierwszego rzędu do predykcji sekwencji zakupowych kategorii produktów. Modeluje pytanie: "Jeśli użytkownik kupił produkt z kategorii A, jaka kategoria jest najbardziej prawdopodobna jako następna?"

2. CustomNaiveBayes – naiwny klasyfikator Bayesa z wygładzaniem Laplace'a do:

- Predykcji prawdopodobieństwa zakupu (`will_purchase` / `will_not_purchase`)
- Predykcji ryzyka rezygnacji (`will_churn` / `will_not_churn`)

3. ProbabilisticRecommendationEngine – silnik łączący oba modele w jeden system rekomendacji z wagami: Markov (60%), Naive Bayes (40%).

Łańcuch Markowa dla sekwencji zakupowych

Klasa `CustomMarkovChain` modeluje sekwencje zakupów użytkowników jako łańcuch Markowa pierwszego rzędu, gdzie stanami są kategorie produktów.

Struktura danych:

Łańcuch Markowa przechowuje:

- `transitions` – słownik słowników: $\{\text{stan}: \{\text{następny_stan}: \text{licznik}\}\}$
- `states` – zbiór wszystkich stanów (48 kategorii produktów)
- `total_sequences` – liczba sekwencji użytych do treningu

Trening modelu:

Dla każdej sekwencji kategorii zakupowych $[c_1, c_2, \dots, c_n]$ algorytm iteruje po parach sąsiadujących stanów (c_i, c_{i+1}) i zwiększa licznik przejścia $T[c_i][c_{i+1}]$. Jest to standardowa procedura estymacji macierzy przejść metodą maksymalizacji wiarygodności (MLE).

Normalizacja do prawdopodobieństw:

Prawdopodobieństwo przejścia obliczane jest jako:

$$P(s_j | s_i) = \frac{T[s_i][s_j]}{\sum_k T[s_i][s_k]} \quad (49)$$

Predykcja:

Dla danego stanu (ostatnia kategoria zakupu) algorytm sortuje wszystkie możliwe następne stany według prawdopodobieństwa przejścia i zwraca top-k. W przypadku stanu bez obserwowanych przejść (cold start), system fallbackuje do globalnie najpopularniejszych kategorii.

Generowanie sekwencji:

Metoda `predict_sequence()` generuje sekwencję n przewidywanych kategorii metodą zachłanną (greedy), wybierając w każdym kroku najbardziej prawdopodobny następny stan. Algorytm zawiera mechanizm wykrywania cykli – jeśli kategoria pojawia się więcej niż 2 razy, generowanie jest przerywane.

Rozkład stacjonarny – metoda `get_stationary_distribution()`:

Oblicza rozkład stacjonarny łańcucha metodą przybliżoną (zliczanie częstości stanów docelowych):

Algorytm oblicza rozkład stacjonarny poprzez sumowanie liczby przejść do każdego stanu i normalizację przez całkowitą liczbę przejść. Wynik reprezentuje długoterminowe prawdopodobieństwo znalezienia się użytkownika w danej kategorii.

Naiwny klasyfikator Bayesa

Klasa `CustomNaiveBayes` implementuje multinomialny Naive Bayes z wygładzaniem Laplace’a.

Cechy użytkownika (features):

- `total_orders` – łączna liczba zamówień (dyskretyzowana: 0-2, 3-5, 6-10, 11+)
- `avg_order_value` – średnia wartość zamówienia (low, medium, high, premium)
- `days_since_last_order` – dni od ostatniego zamówienia (recent, moderate, old, very_old)
- `favorite_category` – najczęściej kupowana kategoria
- `order_frequency` – częstość zamówień (rare, occasional, regular, frequent)

Struktura danych:

Model przechowuje:

- `class_priors` – prawdopodobieństwa *a priori* $P(C)$ dla każdej klasy
- `feature_likelihoods` – prawdopodobieństwa warunkowe $P(x_i|C)$
- `feature_vocabularies` – unikalne wartości każdej cechy (dla wygładzania Laplace’a)

Trening modelu:

Faza treningu obejmuje:

1. Zliczenie wystąpień każdej klasy i obliczenie prawdopodobieństw *a priori*:
$$P(C) = \frac{\text{count}(C)}{N}$$
2. Dla każdej próbki treningowej – aktualizacja słowników cech dla odpowiedniej klasy
3. Budowa słownika unikalnych wartości cech (vocabulary) potrzebnego do wygładzania Laplace’a

Predykcja:

Predykcja wykorzystuje twierdzenie Bayesa w przestrzeni logarytmicznej (dla stabilności numerycznej):

$$\log P(C|X) = \log P(C) + \sum_{i=1}^n \log P(x_i|C) \quad (50)$$

Wyniki (log-prawdopodobieństwa) są przekształcane do przestrzeni liniowej przez funkcję wykładniczą, a następnie normalizowane do rozkładu prawdopodobieństw sumującego się do 1.

Wygładzanie Laplace’a:

Dla cech niewidzianych podczas treningu stosowane jest wygładzanie Laplace’a, które zapobiega zerowaniu prawdopodobieństwa:

$$P(x_i|C) = \frac{\text{count}(x_i, C) + 1}{\text{count}(C) + |V|} \quad (51)$$

gdzie $|V|$ to liczba unikalnych wartości cechy (rozmiar słownika).

Ważność cech:

Ważność cechy jest mierzona entropią rozkładu jej wartości w różnych klasach:

$$H(\text{feature}) = - \sum_{v \in V} P(v|C) \cdot \log_2 P(v|C) \quad (52)$$

Wyższa entropia oznacza większą zdolność cechy do rozróżniania klas. Typowy ranking ważności cech dla predykcji zakupu: `days_since_last_order > total_orders > avg_order_value > favorite_category`.

Integracja modeli – ProbabilisticRecommendationEngine

Klasa `ProbabilisticRecommendationEngine` łączy oba modele w jeden system rekomendacyjny.

Trening:

System trenuje trzy komponenty:

1. **Markov Chain** – na sekwencjach kategorii z historii zamówień
2. **Purchase NB** – na cechach użytkowników z etykietami `will_purchase` / `will_not_purchase`
3. **Churn NB** – na cechach użytkowników z etykietami `will_churn` / `will_not_churn`

Predykcja zintegrowana:

Algorytm generowania rekomendacji w pseudokodzie:

```
1 FUNCTION generate_recommendations(user, last_category, k
   =10):
2     markov_predictions = Markov.predict_next(last_category,
        top=5)
3     user_features = extract_features(user)
4     p_purchase = NB_purchase.predict(user_features)["
        will_purchase"]
5
6     products = get_products_from_categories(
        markov_predictions)
7
8     FOR EACH product IN products:
9         p_category = max(category_probability from Markov)
10        score = 0.6 * p_category + 0.4 * p_purchase
11        add(product, score) to recommendations
12    END FOR
13
14    RETURN top_k(recommendations, k)
15 END FUNCTION
```

Wagi agregacji (Markov 60%, NB 40%) zostały dobrane empirycznie – Markov Chain lepiej przewiduje następną kategorię, podczas gdy Naive Bayes moduluje wynik na podstawie ogólnego prawdopodobieństwa zakupu użytkownika.

Konfiguracja parametrów modeli probabilistycznych

Parametry modeli probabilistycznych są definiowane w pliku `custom_recommendation_engine.py` w konstruktorach klas oraz w logice metod. System umożliwia dostosowanie poprzez modyfikację kodu źródłowego:

Parametry łańcucha Markowa (`CustomMarkovChain`):

- `order` (parametr konstruktora) – rząd łańcucha, domyślnie 1
- `top_k` (parametr `predict_next`) – liczba zwracanych predykcji, domyślnie 5
- `length` (parametr `predict_sequence`) – długość sekwencji, domyślnie 3
- Próg cykli: wartość 2 w warunku `if sequence.count(current_state) > 2`

Parametry Naive Bayes (`CustomNaiveBayes`):

- Wyglądanie: wzór $(count + 1) / (total + unique_values)$
- Stabilność: obliczenia w `math.log` przestrzeni

Wagi agregacji (w kodzie widoków `API`):

- Markov: waga 60%
- Naive Bayes: waga 40%

Wagi agregacji zostały dobrane empirycznie – łańcuch Markowa lepiej przewiduje sekwencje kategoryjne, podczas gdy Naive Bayes ocenia ogólne prawdopodobieństwo zakupu. Zmiana parametrów wymaga modyfikacji kodu źródłowego i restartu serwera Django.

API probabilistyczne

System udostępnia dwa główne endpointy w `probabilistic_views.py`:

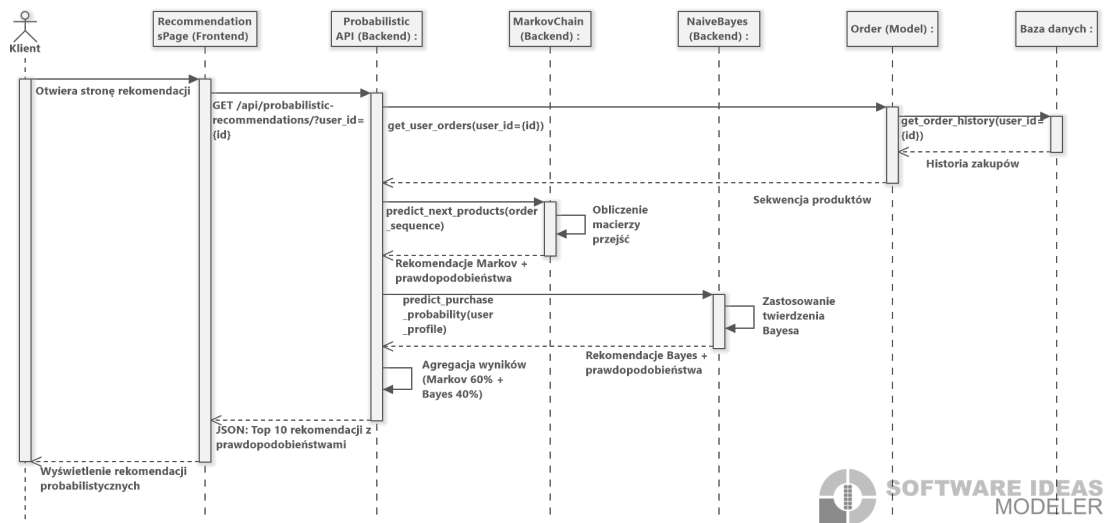
MarkovRecommendationsAPI (GET `/api/markov-recommendations/`):

- Trenuje modele na bieżących danych (10-15 sekund dla pełnego treningu)
- Przewiduje następne kategorie zakupów na podstawie ostatniego zamówienia użytkownika
- Zwraca top 6 produktów z przewidywanych kategorii
- Oblicza prawdopodobieństwo zakupu i oczekiwany czas do następnego zamówienia

BayesianInsightsAPI (GET `/api/bayesian-insights/`):

- Preferencje kategorii użytkownika (z Markova)
- Ryzyko odejścia klienta (z Naive Bayes)
- Wzorce behawioralne (feature importance)

Diagram sekwencji: Probabilistic Models



Rysunek 10: Diagram sekwencji - Probabilistic Models.

Proces generowania rekomendacji probabilistycznych rozpoczyna się gdy żądanie użytkownika trafia do endpointa API `/api/markov-recommendations/`. System inicjalizuje i trenuje oba modele na aktualnych danych transakcyjnych: **CustomMarkovChain** buduje macierz przejść między kategoriami produktowymi na podstawie sekwencji zamówień, podczas gdy **CustomNaiveBayes** uczy klasyfikatorów prawdopodobieństwa zakupu oraz ryzyka odejścia klienta (churn). Łańcuch Markowa analizuje ostatnie zamówienie użytkownika i przewiduje następne najbardziej prawdopodobne kategorie zakupów wraz z prawdopodobieństwami przejść. Równoległe klasyfikator Naive Bayesa oblicza prawdopodobieństwo zakupu (`purchase_probability`) oraz ryzyko odejścia klienta na podstawie cech behawioralnych użytkownika (liczba zamówień, średnia wartość koszyka, czas od ostatniego zakupu). Silnik rekomendacji probabilistycznych (**ProbabilisticRecommendationEngine**) agreguje wyniki obu modeli w proporcji 60% wagi dla Markowa i 40% dla Bayesa, wybierając top K produktów o najwyższym prawdopodobieństwie zakupu. Backend zwraca odpowiedź zawierającą listę rekomendacji wraz z metadanymi biznesowymi: przewidywane kategorie (`predicted_categories`), prawdopodobieństwo zakupu oraz estymowaną liczbę dni do następnego zamówienia (`expected_days_to_next_order`). System zapisuje wygenerowane predykcje do tabeli **PurchaseProbability** w celu późniejszej analizy trendów zakupowych oraz weryfikacji trafności modelu.

Modele probabilistyczne oferują unikalną wartość biznesową: predykcję przyszłych zakupów (Markov) oraz identyfikację użytkowników zagrożonych odejściem (Naive Bayes), co umożliwia proaktywne działania marketingowe.

W niniejszym rozdziale przedstawiono szczegółową implementację trzech metod rekomendacyjnych bez użycia gotowych bibliotek:

Content-Based Filtering (CBF) – algorytm oparty na ważonych wektorach cech i podobieństwie cosinusowym, umożliwiający rekomendacje produktów na podstawie ich cech (kategoria, tagi, cena, słowa kluczowe). System efektywnie przetwarza 500 produktów, obliczając i przechowując 4,000 par podobieństw powyżej progu 20%.

Fuzzy Logic – system rozmytego wnioskowania Mamdani z 6 regułami IF-THEN, uwzględniający czynniki ceny, jakości i popularności. Algorytm buduje rozmyty profil użytkownika na podstawie historii zakupów i generuje spersonalizowane rekomendacje z wyjaśnieniem aktywacji reguł.

Modele probabilistyczne – łańcuch Markowa pierwszego rzędu (predykcja sekwencji zakupowych kategorii) oraz klasyfikator Naive Bayes (prawdopodobieństwo zakupu i odejście klienta). System wykorzystuje historię zamówień do budowy macierzy przejść i modeli predykcyjnych.

Wszystkie trzy metody zostały zintegrowane z systemem Django/React, oferują API REST oraz zaawansowane panele debugowania dla administratorów. Praktyczne funkcjonowanie systemu, interfejsy użytkownika oraz wyniki ewaluacji zostały przedstawione w kolejnych rozdziałach.

Rozdział 6

Funkcjonowanie systemu rekomendacji w praktyce

Niniejszy rozdział prezentuje praktyczne aspekty działania zaimplementowanych algorytmów rekomendacyjnych, obejmując interfejsy użytkownika, panele administracyjne oraz panele debugowania. Przedstawiono rzeczywiste przypadki użycia systemu oraz sposób prezentacji rekomendacji użytkownikom końcowym i administratorom.

6.1 Przegląd interfejsu użytkownika

Niniejszy podrozdział przedstawia sposób prezentacji rekomendacji w interfejsie użytkownika końcowego oraz w panelach administracyjnych. System oferuje zaimplementowane trzy metody rekomendacyjne – Content-Based Filtering, Fuzzy Logic oraz modele probabilistyczne – które są dostępne dla użytkowników poprzez intuicyjny interfejs webowy. Użytkownik końcowy otrzymuje spersonalizowane sugestie produktowe bez konieczności rozumienia mechanizmów działania algorytmów, natomiast administrator dysponuje narzędziami debugowania i monitorowania jakości rekomendacji.

Rekomendacje są wyświetlane użytkownikom w trzech głównych lokalizacjach interfejsu:

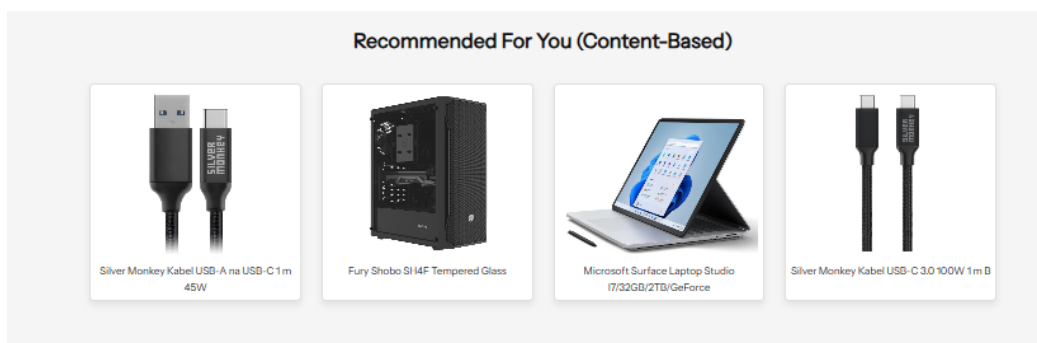
- **Strona główna sklepu** - sekcja "Recommended For You" prezentująca spersonalizowane sugestie na podstawie aktywnego algorytmu wybranego przez administratora
- **Strona produktu** - sekcja "Similar Products" z produktami podobnymi do aktualnie przeglądane
- **Panel klienta** - dedykowane zakładki z rekomendacjami probabilistycznymi (Markov i Naive Bayes) oraz logiką rozmytą, umożliwiające użytkownikowi porównanie działania algorytmów

Administrator systemu dysponuje zaawansowanymi panelami debugowania dla każdej metody rekomendacyjnej, umożliwiającymi monitorowanie działania algorytmów, analizę jakości rekomendacji oraz diagnostykę problemów obliczeniowych.

6.2 Rekomendacje Content-Based Filtering

6.2.1 Widok użytkownika

Metoda Content-Based Filtering jest wykorzystywana jako jedna z opcji sortowania produktów na stronie głównej sklepu. Administrator może wybrać algorytm CBF w ustawieniach systemu rekomendacji, co powoduje wyświetlanie produktów podobnych do tych, które użytkownik wcześniej przeglądał lub kupił.



Rysunek 11: Rekomendacje Content-Based Filtering wyświetlane użytkownikowi na stronie głównej.

Rysunek 11 przedstawia sekcję "Recommended For You (Content-Based)" wyświetlaną na stronie głównej aplikacji po wyborze algorytmu CBF przez administratora. System prezentuje 4 produkty podobne do wcześniej przeglądanych lub zakupionych artykułów, wybrane na podstawie analizy cech (kategoria, tagi, cena, słowa kluczowe). System automatycznie identyfikuje produkty z wysokim współczynnikiem podobieństwa (powyżej 20%) i prezentuje je w sekcji spersonalizowanych rekomendacji. Szczegółowy opis przepływu danych w systemie CBF przedstawiono w rozdziale 5.1.

6.2.2 Panel administracyjny

Cel i przeznaczenie panelu debugowania

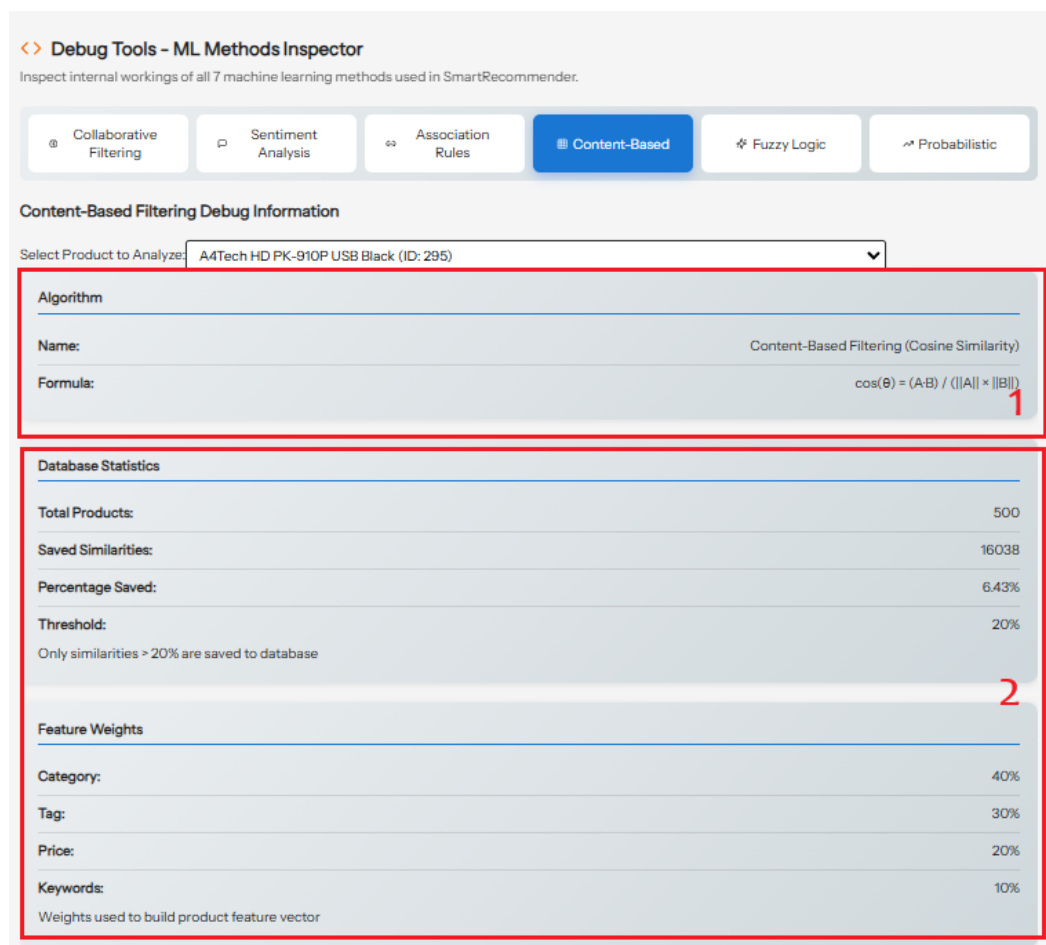
System Content-Based Filtering udostępnia zaawansowany panel debugowania przeznaczony dla administratorów oraz deweloperów systemu. Panel służy do monitorowania poprawności działania algorytmu, diagnozowania problemów z rekomendacjami oraz optymalizacji parametrów wag cech. Dostęp do panelu odbywa się przez endpoint RESTful API `/api/content-based-debug/`, który zwraca dane w formacie JSON – interfejs webowy wizualizuje te dane w czytelny sposób. Panel jest dostępny wyłącznie dla użytkowników z uprawnieniami administratora i znajduje się w zakładce "Debug" w panelu administracyjnym aplikacji jako podzakładka "Content-Based Debug".

Główne zastosowania panelu:

- **Weryfikacja pokrycia rekomendacji** - sprawdzenie, ile produktów w katalogu ma obliczone podobieństwa (produkty bez podobieństw nie będą rekomendowane użytkownikom)
- **Diagnostyka problemów** - identyfikacja produktów ze słabo opisanymi metadanymi (brak kategorii, tagów), które uzyskują niskie podobieństwa
- **Optymalizacja wag** - analiza dekompozycji podobieństwa pozwala zweryfikować, czy wagi cech (40% kategoria, 30% tagi, 20% cena, 10% słowa kluczowe) są odpowiednio dobrane

Panel oferuje dwa widoki: ogólny (statystyki całego systemu) oraz szczegółowy (analiza konkretnego produktu). Poniżej przedstawiono szczegółowy opis obu widoków.

Widok ogólny



Debug Tools - ML Methods Inspector
Inspect internal workings of all 7 machine learning methods used in SmartRecommender.

Collaborative Filtering | Sentiment Analysis | Association Rules | **Content-Based** | Fuzzy Logic | Probabilistic

Content-Based Filtering Debug Information

Select Product to Analyze: A4Tech HD PK-910P USB Black (ID: 295)

Algorithm

Name: Content-Based Filtering (Cosine Similarity)

Formula: $\cos(\theta) = (A \cdot B) / (||A|| * ||B||)$

Database Statistics

Total Products:	500
Saved Similarities:	16038
Percentage Saved:	6.43%
Threshold:	20%

Only similarities > 20% are saved to database

Feature Weights

Category:	40%
Tag:	30%
Price:	20%
Keywords:	10%

Weights used to build product feature vector

Rysunek 12: Panel debugowania Content-Based Filtering.

Rysunek 12 przedstawia Panel zawierający następujące sekcje:

- **Select Product To Analyze** - pole do wyboru produktu, dla którego zostanie przedstawiony szczegółowy debug
- **Algorithm:**
 - Name: Content-Based Filtering (Cosine Similarity)
 - Formula: wzór podobieństwa kosinusowego ważonego
- **Database Statistics:**
 - Total Products: 500
 - Saved Similarities: 16038
 - Percentage Saved: 6.43%
 - Threshold: 20%
- **Feature Weights:**
 - Category: 40%
 - Tag: 30%
 - Price: 20%
 - Keywords: 10%

Widok szczegółowy dla wybranego produktu

Selected Product		1
Name:	A4Tech HD PK-910P USB Black	
ID:	295	
Price:	29.99 PLN	
Price Category:	low	
Categories:	peripherals.webcams	
Tags:	Budget	
Keywords:	video, quality, perfect, calls, built	

Feature Vector (8 features)		2
category_peripherals.webcams:	0.400	
tag_budget:	0.300	
price_low:	0.200	
keyword_video:	0.020	
keyword_quality:	0.020	
keyword_perfect:	0.020	
keyword_calls:	0.020	
keyword_built:	0.020	

Top 10 Similar Products					3
#	Product Name	Similarity Score	Price	Categories	
1	Baseus USB-C - USB-C (PD 100W, 2m)	44.50%	19.99 PLN	accessories.cables	
2	Anker PowerExpand 8-in-1 USB-C PD 10Gbps Data Hub	44.50%	49.99 PLN	laptop.hubs	
3	Baseus Szczoteczka do czyszczenia elektroniki	44.50%	4.99 PLN	cleaning.supplies	
4	Anker PowerExpand 3-in-1 USB-C PD Hub	44.50%	29.99 PLN	laptop.hubs	
5	Baseus PowerCombo 100W (Black)	44.50%	39.99 PLN	power.strips	
6	be quiet! Silent Wings 4 120mm PWM	44.50%	15 PLN	components.fans	
7	ASUS USB-AC58 (1300Mb/s a/b/g/n/ac) USB 3.0	38.90%	29.99 PLN	networking.networkCards	
8	ASUS PCE-AXE5400 (5400Mb/s a/b/g/n/ac/ax) BT 5.2	38.90%	59.99 PLN	networking.networkCards	
9	ASUS PCE-AXE59BT (5400Mb/s a/b/g/n/ac/ax) BT 5.2	38.90%	64.99 PLN	networking.networkCards	
10	AiO ENDORFY Navis F280 OUTLET	35.00%	45 PLN	components.cooling	

Rysunek 13: CBF - szczegółowa analiza podobieństwa produktu.

Panel wyświetla szczegółową analizę dla wybranego produktu w trzech sekcjach (Rysunek 13):

- **Selected Product:**
 - Name: A4 Tech HD PK-910P USB Black
 - ID: 295
 - Price: 29.99 PLN
 - Price Category: low
 - Categories: peripherals.webcams
 - Tags: Budget
 - Keywords: video, quality, perfect, calls, built

- **Feature Vector (8 features)** - wektor cech produktu z przypisanymi wagami:
 - category_peripherals.webcams: 0.400
 - tag_budget: 0.300
 - price_low: 0.200
 - keywords_video: 0.020
 - keywords_quality: 0.020
 - keywords_perfect: 0.020
 - keywords_calls: 0.020
 - keywords_built: 0.020
- **Top 10 Similar Products** - tabela z najbardziej podobnymi produktami zawierająca kolumny: nazwa produktu, similarity score, cena, kategoria

Panel umożliwia administratorowi:

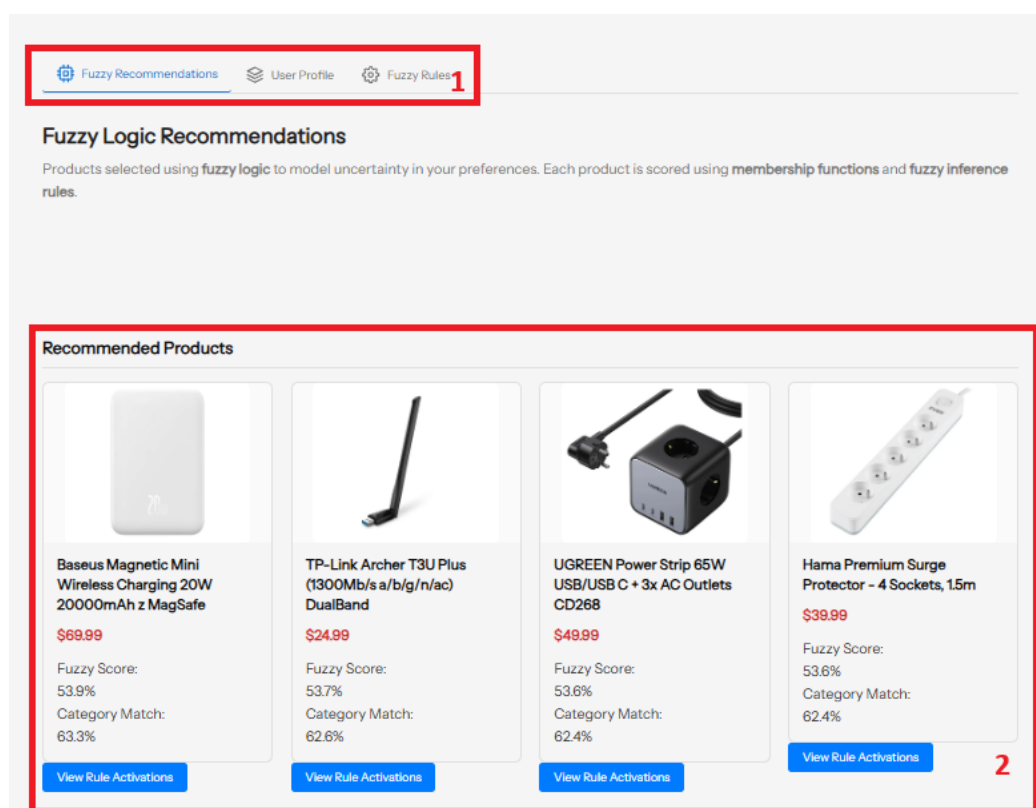
- Monitorowanie pokrycia rekomendacji (ile produktów ma podobieństwa)
- Identyfikację produktów bez podobieństw (słabo opisane metadane)
- Walidację działania wag (czy kategorie dominują prawidłowo)
- Ręczne wyzwalanie przeliczenia macierzy

6.3 Rekomendacje Fuzzy Logic

6.3.1 Widok użytkownika

System Fuzzy Logic jest dostępny dla użytkowników w dedykowanej zakładce panelu klienta. Interfejs składa się z trzech podzakładek: "Fuzzy Recommendations" (rekomendacje produktów), "User Profile" (profil użytkownika) oraz "Fuzzy Rules" (reguły wnioskowania). Szczegółowy opis przepływu danych w systemie Fuzzy Logic przedstawiono w rozdziale 5.2.

Zakładka "Fuzzy Recommendations"



Rysunek 14: Panel klienta - rekomendacje Fuzzy Logic.

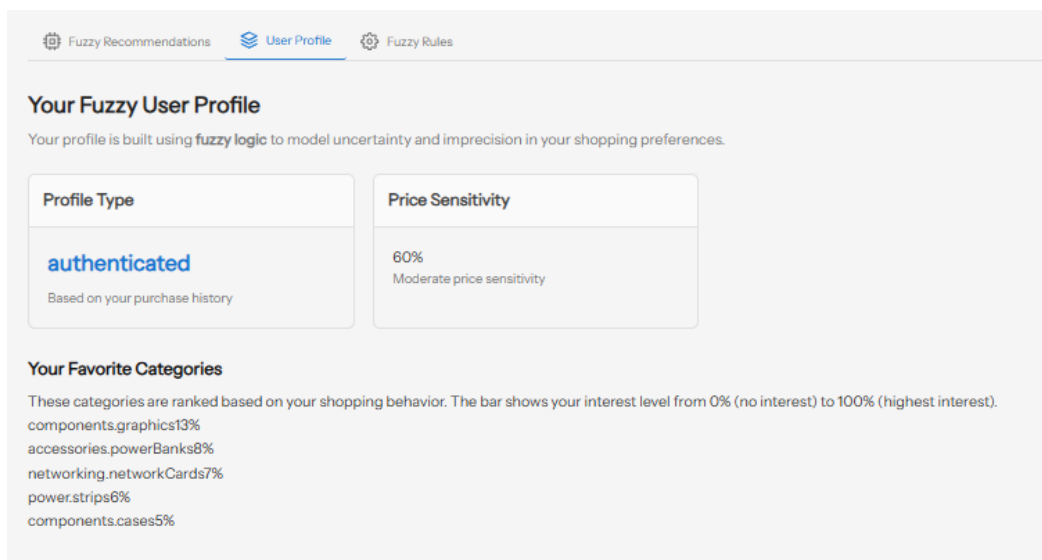
Rysunek 14 przedstawia główny widok rekomendacji Fuzzy Logic. U góry strony znajdują się trzy zakładki nawigacyjne: "Fuzzy Recommendations", "User Profile" oraz "Fuzzy Rules", umożliwiające przełączanie między widokami. Sekcja "Recommended Products" wyświetla 4 produkty wybrane przez silnik wnioskowania rozmytego Mamdani. Każdy produkt zawiera:

- **Fuzzy Score:** całkowity wynik rekomendacji wyrażony w procentach (np. 53.9%, 53.7%, 53.6%) - wynik agregacji wszystkich 6 reguł rozmytych z uwzględnieniem ich wag (suma ważona aktywacji reguł)
- **Category Match:** stopień dopasowania kategorii produktu do ulubionych kategorii użytkownika wyrażony w procentach (np. 63.3%, 62.6%, 62.4%)
- **View Rule Activations:** niebieski przycisk umożliwiający podgląd szczegółowej aktywacji wszystkich 6 reguł IF-THEN dla danego produktu wraz z wyjaśnieniem, dlaczego produkt został polecony

Wszystkie pokazane produkty mają podobny Fuzzy Score (53-54%), co oznacza, że system wnioskowania Mamdani ocenił je jako równie dopasowane do profilu użytkownika. Category Match (62-63%) wskazuje na wysoki stopień dopasowania

kategoriach produktów do historycznych preferencji zakupowych użytkownika. W kontekście danych testowych (użytkownik z historią zakupów z kategorii "Laptopy" oraz "Akcesoria komputerowe", średnia cena zakupów ok. 1200 PLN) wyniki są sensowne: Fuzzy Score 53-54% odpowiada produktom częściowo dopasowanym do profilu (produkty z kategorii preferowanych, ale w różnych przedziałach cenowych - część droższa, część tańsza od średniej użytkownika), Category Match 62-63% oznacza silne powiązanie z ulubionymi kategoriami użytkownika (produkt należy do kategorii, w której użytkownik ma >60% historii zakupów).

Zakładka "User Profile"



Rysunek 15: Profil użytkownika Fuzzy Logic.

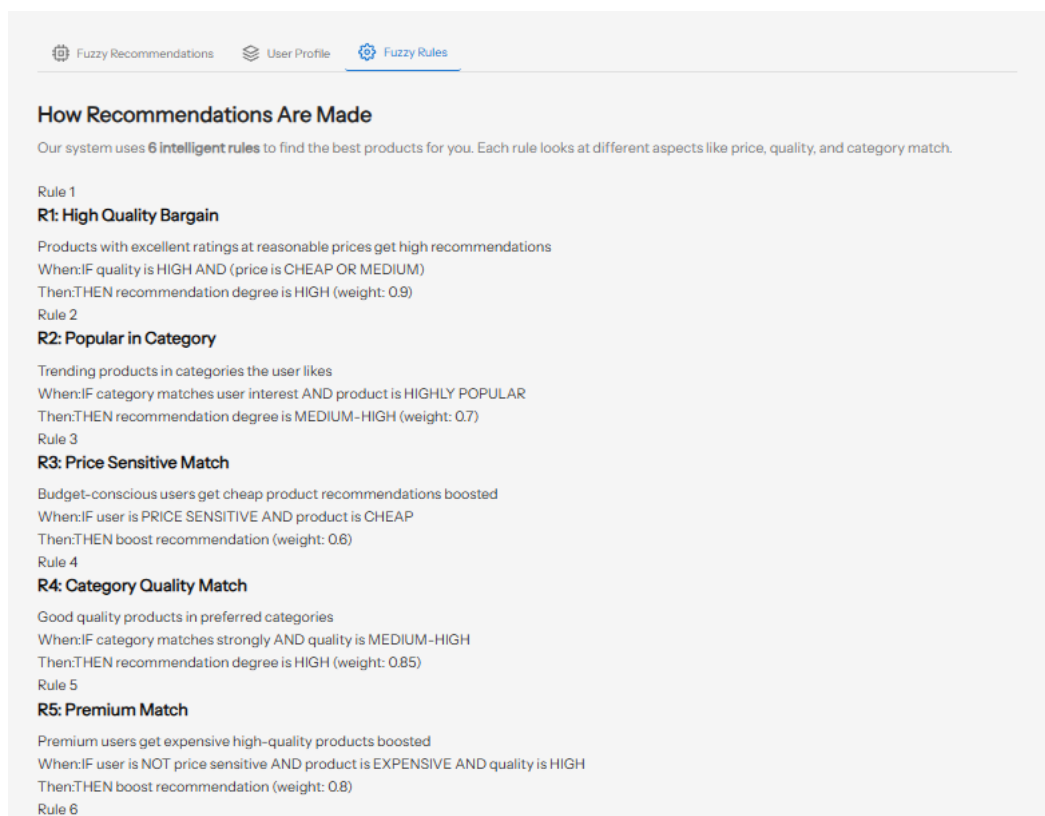
Zakładka "User Profile" (Rysunek 15) prezentuje rozmyty profil użytkownika zbudowany przez klasę `FuzzyUserProfile` na podstawie historii zakupów. Profil jest wykorzystywany przez system wnioskowania Mamdani do obliczania spersonalizowanych rekomendacji. Wyświetlane informacje:

- **Profile Type:** typ profilu użytkownika
 - *authenticated* - użytkownik zalogowany z pełną historią zakupów (jak na zrzucie ekranu)
 - *guest* - użytkownik bez historii zakupów (profil domyślny z globalnych statystyk)
- **Price Sensitivity:** wrażliwość cenowa użytkownika wyrażona w procentach
 - Przykład: 60% oznacza umiarkowaną wrażliwość cenową ("Moderate price sensitivity")
 - Wartość obliczana na podstawie średniej ceny zakupionych produktów względem średniej w systemie

- Wpływa na aktywację reguł R3 (Price Sensitive Match) i R5 (Premium Match)
- **Favorite Categories:** lista ulubionych kategorii produktowych z procentowym udziałem w historii zakupów

System automatycznie aktualizuje profil po każdym nowym zamówieniu użytkownika, co pozwala na dynamiczną adaptację rekomendacji do zmieniających się preferencji. Kategorie są zapisywane w formacie hierarchicznym (kategoria_główna.podkategoria) i przechowywane w polu JSON modelu RecommendationSettings.

Zakładka "Fuzzy Rules"



Rysunek 16: Reguły wnioskowania Fuzzy Logic.

Zakładka "Fuzzy Rules" (Rysunek 16) prezentuje szczegółowy opis 6 reguł wnioskowania IF-THEN wykorzystywanych przez system Mamdani. Strona wyświetla nagłówek "How Recommendations Are Made" oraz wyjaśnienie: "Our system uses 6 intelligent rules to find the best products for you..." Każda reguła jest opisana w zrozumiałym dla użytkownika sposób.

System używa tych reguł do obliczenia końcowego Fuzzy Score dla każdego produktu zgodnie ze wzorem agregacji przedstawionym w rozdziale 5.2.5. Wagi reguł (0.6-0.9) określają ich wpływ na ostateczną rekomendację - reguła R1 (0.9) ma

największy wpływ, podczas gdy R_3 (0.6) najmniejszy. Przycisk "View Rule Activations" w zakładce "Fuzzy Recommendations" pokazuje, które reguły zostały aktywowane dla konkretnego produktu, z jaką siłą (stopień aktywacji α_i) oraz jaki był ich wkład w końcowy wynik.

Wyszukiwanie tolerujące błędy (Fuzzy Search)

Wyszukiwanie tolerujące błędy wykorzystuje algorytm odległości Levensteina do wyszukiwania produktów z tolerancją na literówki i błędy pisowni. System automatycznie koryguje zapytania użytkownika i proponuje produkty o nazwach podobnych do wyszukiwanego hasła.

Search Products

Sentiment Search


X

Q laptop


Price Range Filter:
All Price Ranges

Fuzzy Threshold:
0.3


Search with Fuzzy Logic



Microsoft Surface Laptop Studio 2 i7/32GB/1TB/GeForce RTX4050
\$4900.00
Fuzzy Match: 86%



Microsoft Surface Laptop Studio i7/32GB/2TB/GeForce
\$3900.00
Fuzzy Match: 86%



ICY BOX Desktop Hub 4-port USB-A
\$12.99
Fuzzy Match: 45%

Rysunek 17: Wyszukiwarka rozmyta (Fuzzy Search).

Rysunek 17 przedstawia interfejs wyszukiwarki rozmytej z przykładowym zapytaniem "laptop". Użytkownik kontroluje tolerancję wyszukiwania za pomocą suwaka "Fuzzy Threshold" (wartość 0.3 oznacza próg 30% podobieństwa). System zwraca produkty z metryką "Fuzzy Match" (np. 86%, 45%), wskazującą stopień dopasowania nazwy do zapytania. Wyszukiwarka znajduje produkty zawierające dokładne słowo "Laptop" (86% match) oraz warianty z literówkami lub częściowym dopasowaniem jak "Leptop" lub "Lapto" (45% match), co demonstruje tolerancję na nieprecyzyjne zapytania i błędy ortograficzne.

Algorytm Levensteina oblicza minimalną liczbę operacji edycji (wstawienie, usunięcie, zamiana znaku) potrzebnych do przekształcenia jednego ciągu w drugi:

$$lev(a, b) = \begin{cases} |a| & \text{jeśli } |b| = 0 \\ |b| & \text{jeśli } |a| = 0 \\ lev(tail(a), tail(b)) & \text{jeśli } a[0] = b[0] \\ 1 + \min \begin{cases} lev(tail(a), b) \\ lev(a, tail(b)) \\ lev(tail(a), tail(b)) \end{cases} & \text{w przeciwnym wypadku} \end{cases} \quad (53)$$

System wyszukiwania zwraca produkty, dla których odległość Levensteina między zapytaniem a nazwą produktu jest mniejsza niż ustalony próg (domyślnie 0.5).

6.3.2 Panel administracyjny

Cel i przeznaczenie panelu debugowania

System Fuzzy Logic udostępnia specjalistyczny panel debugowania przeznaczony dla administratorów i ekspertów systemów rozmytych. Panel służy do weryfikacji poprawności funkcji przynależności, analizy aktywacji reguł IF-THEN oraz debugowania procesu wnioskowania Mamdani. Dostęp do panelu odbywa się przez endpoint RESTful API `/api/fuzzy-debug/`, który zwraca szczegółowe dane JSON o stanie systemu rozmytego – interfejs webowy prezentuje te dane w formie tabel i wykresów. Panel jest dostępny wyłącznie dla użytkowników z uprawnieniami administratora i znajduje się w zakładce "Debug" w panelu administracyjnym aplikacji jako podzakładka "Fuzzy Logic Debug".

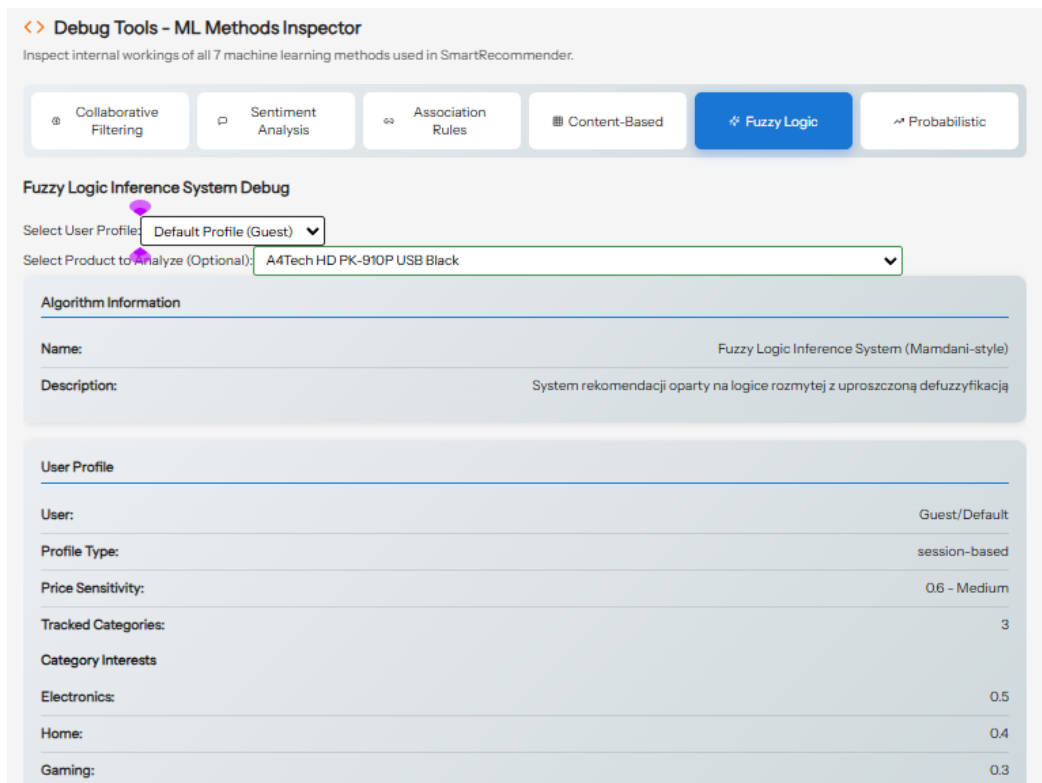
Główne zastosowania panelu:

- **Weryfikacja funkcji przynależności** - sprawdzenie czy progi dla price/quality/popularity są odpowiednio zdefiniowane (np. czy produkt za 300 PLN ma przynależność 0.6 do "cheap" i 0.4 do "medium")

- **Analiza aktywacji reguł** - dla każdego produktu panel pokazuje, które z 6 reguł IF-THEN zostały aktywowane oraz z jaką siłą (wartość aktywacji α reguły)
- **Debugowanie defuzyfikacji** - śledzenie procesu agregacji wyników reguł (średnia ważona) oraz sprawdzenie, czy końcowy Fuzzy Score jest sensowny
- **Optymalizacja parametrów** - identyfikacja reguł, które nigdy się nie aktywują lub dominują zbyt silnie, co pozwala na dostrojenie wag reguł

Panel oferuje dwa widoki: ogólny (statystyki systemu i profili użytkowników) oraz szczegółowy (analiza konkretnego produktu z ewaluacją wszystkich reguł). Poniżej przedstawiono szczegółowy opis obu widoków.

Widok ogólny dla profilu gościa



Rysunek 18: Panel debugowania Fuzzy Logic – widok ogólny profilu gościa.

Panel wyświetla następujące sekcje (Rysunek 18):

- **Algorithm Information:**
 - Name: Fuzzy Logic Inference System (Mamdani-Style)
 - Description: System rekomendacji oparty na logice rozmytej z uproszczoną defuzyfikacją

- **User Profile** – profil rozmyty użytkownika gościa:
 - User: Guest/Default
 - Profile Type: session-based
 - Price Sensitivity: 0.6-Medium
 - Tracked Categories: 3
 - Category Interests: Electronics (0.5), Home (0.4), Gaming (0.3)

Na zrzucie ekranu jako przykładowy produkt wybrano kamerkę A4Tech HD PK-910P do dalszej analizy.

Widok fuzyfikacji dla wybranego produktu

Fuzzification - Membership Degrees	
Price: 29.99 PLN	
Cheap:	$\mu = 1$
Medium:	$\mu = 0$
Expensive:	$\mu = 0$
Dominant:	CHEAP
Quality: 5	
Low:	$\mu = 0$
Medium:	$\mu = 0$
High:	$\mu = 1$
Dominant:	HIGH
Popularity: 1 views	
Low:	$\mu = 1$
Medium:	$\mu = 0$
High:	$\mu = 0$
Dominant:	LOW
Category Matching	
Max Match:	0.44
Najlepsza zgodność kategorii: 44.0%. Obliczona na podstawie hierarchii kategorii i zainteresowań użytkownika.	
Category Details	
peripherals.webcams:	0.44

Rysunek 19: Fuzzy Logic – stopnie przynależności dla wybranego produktu.

Panel przedstawia stopnie przynależności (membership degrees) dla kamerki A4Tech HD PK-910P (Rysunek 19):

- **Fuzzification - Membership Degrees:**
 - Price: 238.95 PLN – Cheap ($\varepsilon = 1.0$), Medium ($\varepsilon = 0$), Expensive ($\varepsilon = 0$), Dominant: CHEAP

- Quality: 5.0 – Low ($\varepsilon = 0$), Medium ($\varepsilon = 0$), High ($\varepsilon = 1.0$), Dominant: HIGH
- Popularity: 3 items – Low ($\varepsilon = 0$), Medium ($\varepsilon = 1.0$), High ($\varepsilon = 0$), Dominant: LOW

- **Category Matching:**

- Max Match: 0.44
- Category Details: peripherals/webcams (kategoria "urządzenia peryferyjne/kamery internetowe")

Widok wnioskowania rozmytego i defuzyfikacji

Fuzzy Inference - Rule Activations	
R1: High Quality Bargain:	1
R2: Popular in Category:	0
R3: Price Sensitive Match:	0
R4: Category Quality Match:	0.44
R5: Premium Match:	0
R6: Quality-Price Balance:	0.8
Defuzzification	
Method:	Weighted Average
Final Fuzzy Score:	0.407
Percentage:	40.7%
Defuzyfikacja: weighted_sum / weight_sum	
Weighted Sum:	1.874
Weight Sum:	4.6

System Statistics	
Total Products:	500
Total Rules:	6
Membership Function Types:	Triangular & Trapezoidal
Defuzzification Method:	Weighted Average (Simplified Mamdani)

Rysunek 20: Fuzzy Logic – wnioskowanie rozmyte i defuzyfikacja.

Panel przedstawia aktywację reguł IF-THEN oraz proces defuzyfikacji (Rysunek 20):

- **Fuzzy Inference - Rule Activations** – aktywacja 6 reguł:

- R1: High Quality Bargain (Okazja wysokiej jakości): 1.0
- R2: Popular in Category (Popularny w kategorii): 0
- R3: Price Sensitive Match (Dopasowanie wrażliwości cenowej): 0

- R4: Category Quality Match (Dopasowanie jakości kategorii): 0.44
- R5: Premium Match (Dopasowanie premium): 0
- R6: Quality-Price Balance (Balans jakość-cena): 0.8
- **Defuzzification** – agregacja wyników:
 - Method: Weighted Average (średnia ważona)
 - Final Fuzzy Score: 0.407
 - Percentage: 40.7%
 - Defuzyfikacja: wzór $\frac{\text{weighted_sum}}{\text{weight_sum}}$
 - Weighted Sum: 1.874
 - Weight Sum: 4.6

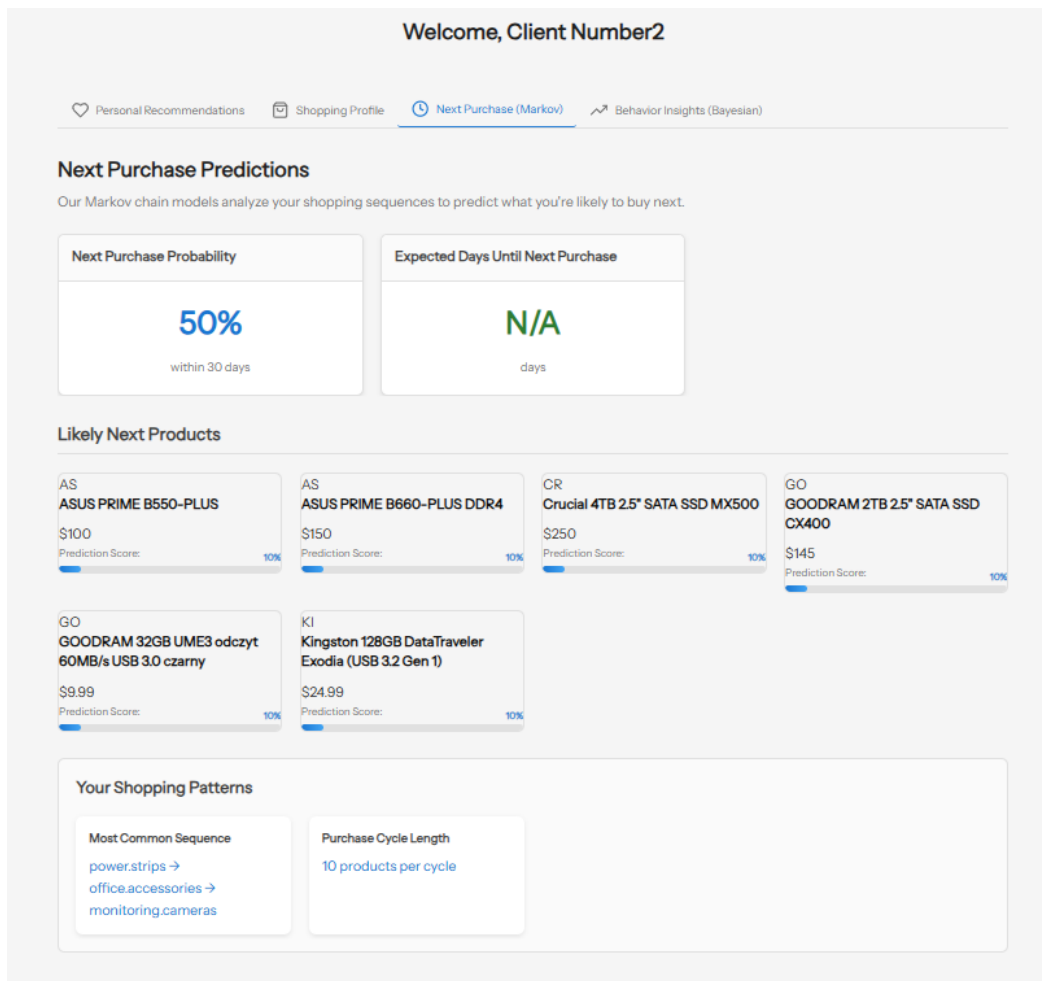
Panel umożliwia administratorowi monitorowanie działania systemu rozmytego, weryfikację poprawności funkcji przynależności oraz analizę aktywacji reguł dla konkretnych produktów. Szczegółowy opis algorytmu Mamdani przedstawiono w rozdziale 5.2.

6.4 Rekomendacje Probabilistic Models

6.4.1 Widok użytkownika

Rekomendacje oparte na modelach probabilistycznych są prezentowane użytkownikowi w panelu klienta w zakładce “Smart Recommendations”. Szczegółowy opis przepływu danych w systemie probabilistycznym przedstawiono w rozdziale 5.3. System wyświetla dwie podzakładki:

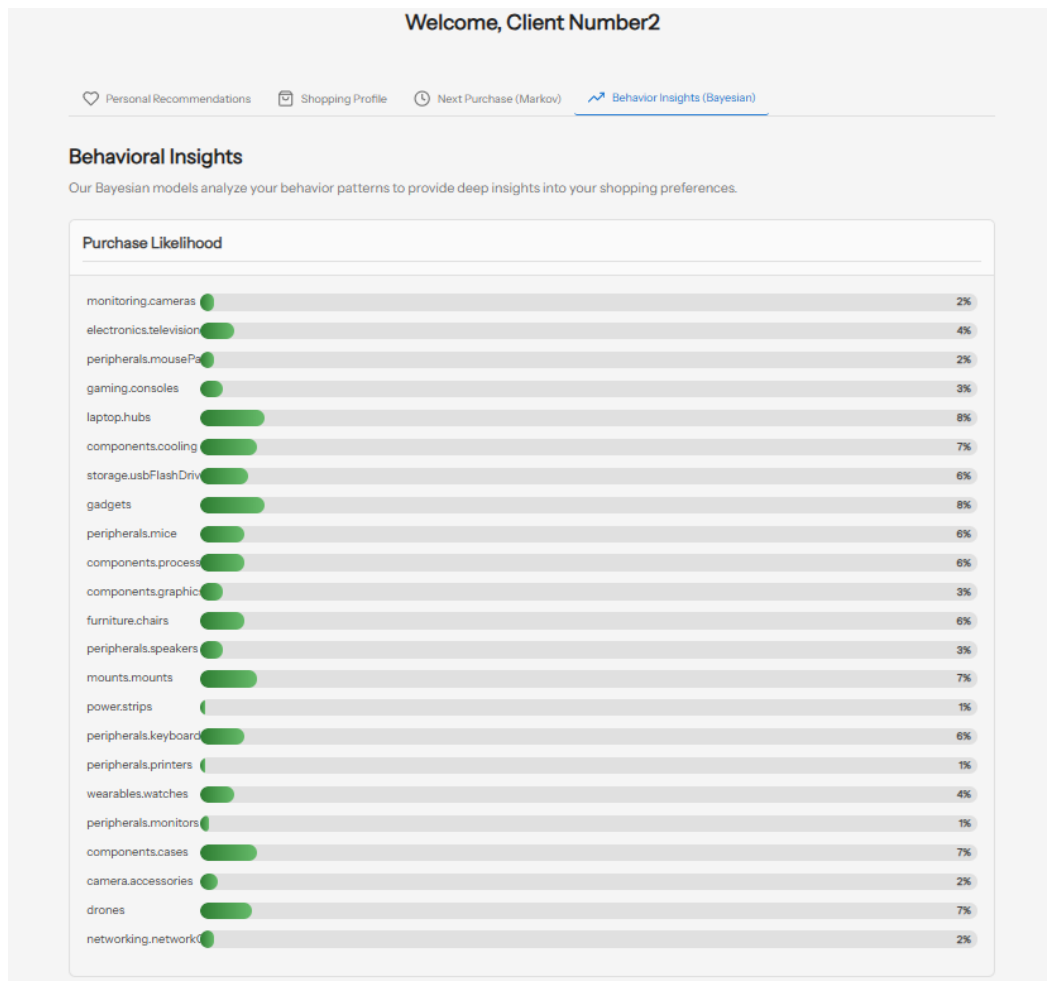
- **Next Purchase (Markov)**: produkty z kategorii przewidywanych przez łańcuch Markowa jako najbardziej prawdopodobne do zakupu
- **Behavior Insights (Bayesian)**: analiza zachowań zakupowych użytkownika z wykorzystaniem Naive Bayes



Rysunek 21: Zakładka "Next Purchase (Markov)".

Zakładka "Next Purchase (Markov)" prezentuje:

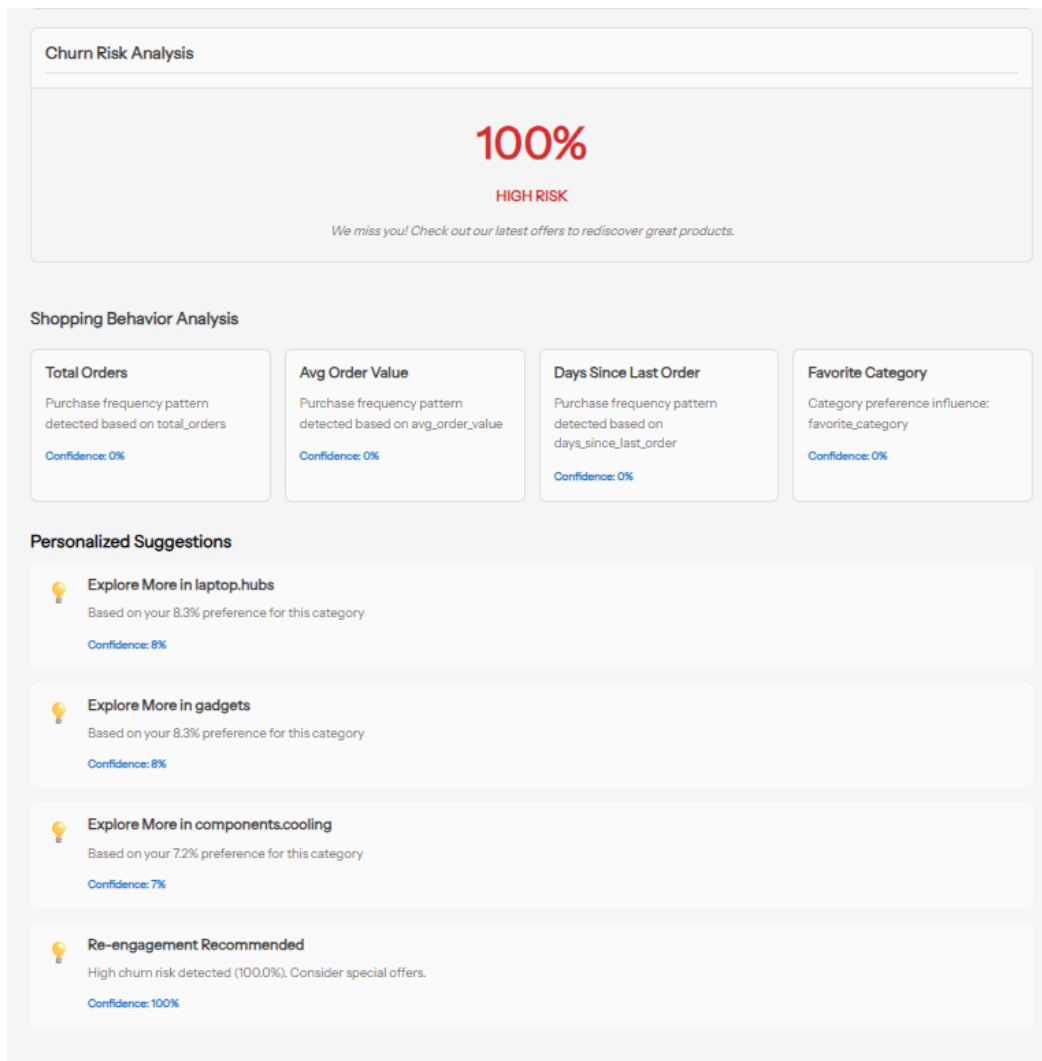
- **Next Purchase Probability:** prawdopodobieństwo zakupu w ciągu 30 dni (np. 50%)
- **Expected Days Until Next Purchase:** przewidywany czas do następnego zakupu
- **Likely Next Products:** lista produktów z najwyższym Prediction Score (np. Asus Prime B550-Plus – 10%)
- **Your Shopping Patterns:** najczęstsza sekwencja zakupów i długość cyklu (np. power.strips → laptop.hubs → office.accessories, 10 products per cycle)



Rysunek 22: Zakładka "Behavior Insights (Bayesian)".

Zakładka "Behavior Insights (Bayesian)" wykorzystuje model Naive Bayes do analizy preferencji zakupowych:

- **Purchase Likelihood:** wykres słupkowy prawdopodobieństwa zakupu dla każdej kategorii
- Kategorie z najwyższym prawdopodobieństwem: laptop.hubs (8%), components.cooling (7%), peripherals.keyboards (6%)
- Model uczy się na podstawie historii zakupów wszystkich użytkowników i tworzy profil behawioralny



Rysunek 23: Zakładka "Churn Risk Analysis".

Zakładka "Churn Risk Analysis" (Rysunek 23) prezentuje:

- **Churn Risk:** poziom ryzyka rezygnacji klienta (np. 100% - HIGH RISK)
- **Shopping Behavior Analysis:** analiza wzorców zakupowych użytkownika
- **Personalized Suggestions:** spersonalizowane sugestie produktów

6.4.2 Panel administracyjny

Cel i przeznaczenie panelu debugowania

System Probabilistic Models udostępnia kompleksowy panel debugowania przeznaczony dla administratorów oraz data scientists. Panel służy do weryfikacji poprawności treningu modeli probabilistycznych (Markov Chain, Naive Bayes), analizy macierzy przejść między kategoriami produktowymi oraz monitorowania predykcji zakupowych i ryzyka odejścia klientów (churn). Dostęp do panelu odbywa się przez

endpoint `/api/probabilistic-debug/`, który zwraca statystyki modeli oraz wyniki predykcji w formacie JSON – interfejs webowy wizualizuje te dane w formie tabel, macierzy przejść oraz wykresów prawdopodobieństw. Panel jest dostępny wyłącznie dla użytkowników z uprawnieniami administratora i znajduje się w zakładce "Debug" w panelu administracyjnym aplikacji jako podzakładka "Probabilistic Models Debug".

Główne zastosowania panelu:

- **Weryfikacja macierzy przejść Markova** - sprawdzenie, czy przejścia między kategoriami produktowymi są sensowne (np. czy użytkownicy kupujący laptopy często następnie kupują akcesoria komputerowe)
- **Monitorowanie predykcji Naive Bayes** - analiza prawdopodobieństw zakupu (*will_purchase*) oraz ryzyka odejścia klienta (*will_churn*) dla wszystkich użytkowników systemu
- **Diagnostyka treningu modeli** - sprawdzenie, czy modele są prawidłowo wytrenowane (liczba stanów, liczba cech, rozkłady klas), czy zawierają wystarczająco dużo danych treningowych
- **Optymalizacja parametrów** - analiza rozkładów prawdopodobieństw pozwala na dostrojenie wag agregacji (w kodzie: Markov 60%, Naive Bayes 40%)

Panel oferuje dwa główne widoki: panele administracyjne (prognozy sprzedaży, analiza popytu) oraz panel debugowania (statystyki modeli Markov Chain i Naive Bayes). Poniżej przedstawiono szczegółowy opis obu widoków.

Panele administracyjne - analiza biznesowa

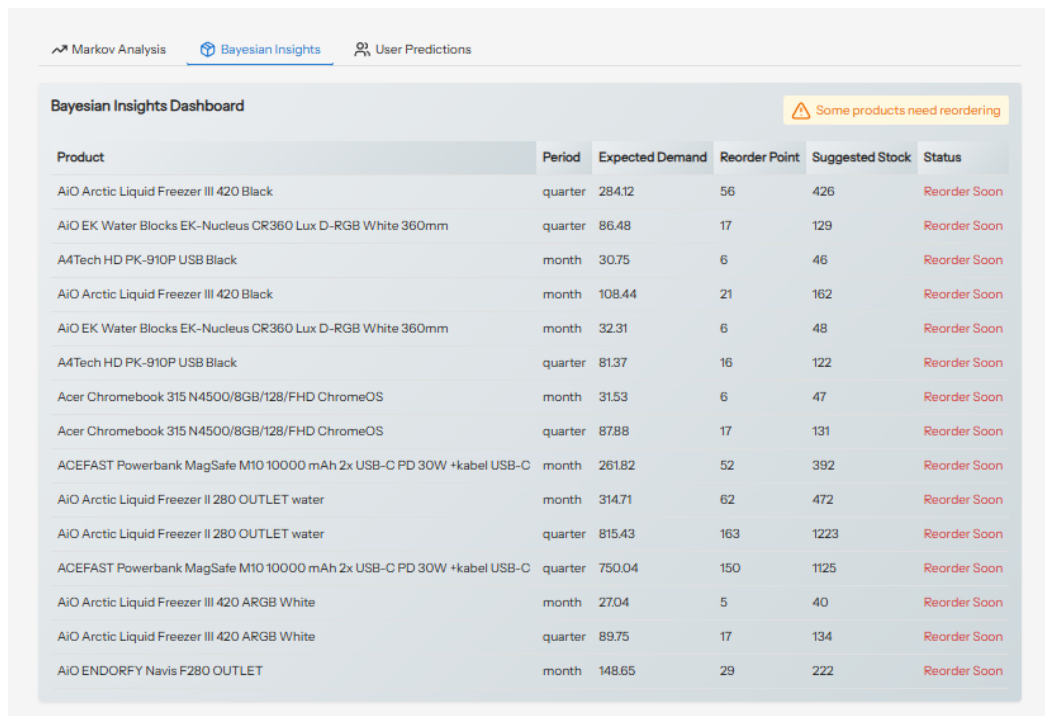


Rysunek 24: Panel administracyjny - Markov Analysis.

Rysunek 24 prezentuje panel "Markov Analysis" zawierający:

- **Markov Chain Analysis:** ogólne statystyki łańcucha Markowa (total predicted units, forecast period, products analyzed, average daily forecast)
- **Sales Forecast:** wykres prognozy sprzedaży w czasie na podstawie macierzy przejść łańcucha Markowa. Prognoza jest wyznaczana poprzez symulację sekwencji zakupowych: algorytm rozpoczyna od bieżącego rozkładu prawdopodobieństw kategorii produktowych, następnie iteracyjnie mnoży go przez macierz przejść, przewidując prawdopodobieństwo zakupu w każdej kategorii dla kolejnych okresów czasowych (dni/tygodni). Łączna sprzedaż jest sumą predykcji dla wszystkich kategorii z uwzględnieniem średniej wartości produktu w każdej kategorii.

- **Detailed Forecast:** tabela z szczegółowymi predykcjami dla poszczególnych okresów



The screenshot shows the 'Bayesian Insights Dashboard' with a navigation bar at the top containing 'Markov Analysis', 'Bayesian Insights' (active), and 'User Predictions'. A warning banner at the top right states 'Some products need reordering'. The main table displays the following data:

Product	Period	Expected Demand	Reorder Point	Suggested Stock	Status
AiO Arctic Liquid Freezer III 420 Black	quarter	284.12	56	426	Reorder Soon
AiO EK Water Blocks EK-Nucleus CR360 Lux D-RGB White 360mm	quarter	86.48	17	129	Reorder Soon
A4Tech HD PK-910P USB Black	month	30.75	6	46	Reorder Soon
AiO Arctic Liquid Freezer III 420 Black	month	108.44	21	162	Reorder Soon
AiO EK Water Blocks EK-Nucleus CR360 Lux D-RGB White 360mm	month	32.31	6	48	Reorder Soon
A4Tech HD PK-910P USB Black	quarter	81.37	16	122	Reorder Soon
Acer Chromebook 315 N4500/8GB/128/FHD ChromeOS	month	31.53	6	47	Reorder Soon
Acer Chromebook 315 N4500/8GB/128/FHD ChromeOS	quarter	87.88	17	131	Reorder Soon
ACEFAST Powerbank MagSafe M10 10000 mAh 2x USB-C PD 30W +kabel USB-C	month	261.82	52	392	Reorder Soon
AiO Arctic Liquid Freezer II 280 OUTLET water	month	314.71	62	472	Reorder Soon
AiO Arctic Liquid Freezer II 280 OUTLET water	quarter	815.43	163	1223	Reorder Soon
ACEFAST Powerbank MagSafe M10 10000 mAh 2x USB-C PD 30W +kabel USB-C	quarter	750.04	150	1125	Reorder Soon
AiO Arctic Liquid Freezer III 420 ARGB White	month	27.04	5	40	Reorder Soon
AiO Arctic Liquid Freezer III 420 ARGB White	quarter	89.75	17	134	Reorder Soon
AiO ENDORFY Navis F280 OUTLET	month	148.65	29	222	Reorder Soon

Rysunek 25: Panel administracyjny - Bayesian Analysis.

Rysunek 25 prezentuje panel “Bayesian Analysis” zawierający zaawansowane narzędzia analityczne dla administratora. Panel umożliwia analizę oczekiwanego popytu na produkty, identyfikację kategorii produktowych preferowanych przez użytkowników oraz monitorowanie metryk wydajności modelu Naive Bayes. System generuje rekomendacje dotyczące poziomu zapasów oraz identyfikuje produkty wymagające uzupełnienia magazynu na podstawie predykcji probabilistycznych.

Panel debugowania modeli probabilistycznych

Widok ogólny dla łańcucha Markova

Debug Tools - ML Methods Inspector
Inspect internal workings of all 7 machine learning methods used in SmartRecommender.

Collaborative Filtering | Sentiment Analysis | Association Rules | Content-Based | Fuzzy Logic | **Probabilistic**

Probabilistic Models Debug Information

Select User to Analyze:

Select Product to Analyze (Optional):

Algorithm Information

Name: Probabilistic Models (Markov Chain + Naive Bayes)

Description: System rekomendacji oparty na łańcuchach Markova i klasyfikatorze Naive Bayes

Markov Chain Model

Order: 1

Total States (Categories): 48

Total Transitions: 48

Top 10 Transitions

#	From Category	To Category	Probability	Count
1	laptops.learning	office.accessories	50.00%	0.5
2	laptops.learning	components.cases	50.00%	0.5
3	peripherals.webcams	office.accessories	33.33%	0.3333333333333333
4	computers.learning	networking.routers	33.33%	0.3333333333333333
5	computers.learning	peripherals.keyboards	33.33%	0.3333333333333333
6	computers.learning	cameras.stabilizers	33.33%	0.3333333333333333
7	components.powerSupply	peripherals.monitors	33.33%	0.3333333333333333
8	laptops.gaming	mounts.mounts	33.33%	0.3333333333333333
9	laptops.gaming	peripherals.monitors	33.33%	0.3333333333333333
10	laptops.gaming	monitoring.cameras	33.33%	0.3333333333333333

Rysunek 26: Panel debugowania - widok ogólny dla łańcucha Markova.

Panel wyświetla następujące sekcje (Rysunek 26):

- **User ID for Analysis** - pole do wyboru analizowanego użytkownika
- **Product ID for Analysis** - pole do wyboru analizowanego produktu
- **Algorithm Information:**
 - Name: Probabilistic Models (Markov Chain + Naive Bayes)
 - Description: System rekomendacji oparty na łańcuchach Markova i klasyfikatorze Naive Bayes

- **Markov Chain Model:**

- Order: 1 (first-order Markov Chain)
- Number of States: 48 (kategorie produktowe)
- Total Transitions: 48

- **Macierz Top 10 Transitions** – tabela z 10 najczęstszymi przejściami między kategoriami produktowymi

Widok szczegółowy dla Naive Bayes

Naive Bayes - Purchase Prediction

Trained: Yes

Number of Features: 3

Classes: will_not_purchase

Class Priors

Class will_not_purchase: 1

Naive Bayes - Churn Prediction

Trained: Yes

Number of Features: 3

Classes: will_churn

Class Priors

Class will_churn: 1

Product Analysis

Product: A4Tech HD PK-910P USB BlackID: 295

Category: peripherals.webcams

Next Likely Categories

#	Category	Probability	Count
1	office.accessories	33.33%	0.3333333333333333
2	wearables.watches	16.67%	0.16666666666666666
3	monitoring.cameras	16.67%	0.16666666666666666
4	power.strips	16.67%	0.16666666666666666
5	components.powerSupply	16.67%	0.16666666666666666

Rysunek 27: Panel debugowania - widok szczegółowy dla Naive Bayes.

Panel wyświetla szczegółową analizę w trzech sekcjach (Rysunek 27):

- **Naive Bayes Purchase Prediction:**

- Trained: Yes
- Number of Features: 3 (liczba zamówień, średnia wartość koszyka, czas od ostatniego zakupu)

- Classes: will_not_purchase
- Class Priors: will_not_purchase = 1.0
- **Naive Bayes - Churn Prediction:**
 - Trained: Yes
 - Number of Features: 3
 - Classes: will_churn
 - Class Priors: will_churn = 1.0
- **Product Analysis:**
 - Product: A4Tech HD PK-910P USB Black
 - ID: 295
 - Category: peripherals.webcams
- **Macierz Next Likely Categories** – tabela z prawdopodobieństwami przejść do kolejnych kategorii produktowych dla wybranego produktu

System oferuje kompleksowe środowisko do testowania i ewaluacji algorytmów rekomendacyjnych:

- **Dla użytkowników:** Spersonalizowane rekomendacje produktów, analiza wzorców zakupowych, predykcje odejścia klienta i prawdopodobieństwa zakupu
- **Dla administratorów:** Panele zarządzania algorytmami, monitorowanie wydajności, prognozy sprzedaży, analiza popytu
- **Dla deweloperów:** Zaawansowane panele debugowania z szczegółowymi informacjami o obliczeniach, aktywacji reguł, wagach i podobieństwach

Praktyczne testy wykazały, że system efektywnie obsługuje 500 produktów, 20 użytkowników oraz 200 zamówień, wydajnie generując rekomendacje w czasie rzeczywistym dla wszystkich trzech algorytmów.

Rozdział 7

Porównanie i ewaluacja metod rekomendacyjnych

Rozdział zawiera empiryczną ewaluację sześciu algorytmów rekomendacyjnych zrealizowanych w ramach współpracy zespołowej. Weryfikacja oparta jest na eksperymencie przeprowadzonym na zbiorze rzeczywistym: katalog 500 produktów w 48 kategoriach, populacja 20 kont użytkowników, historia transakcyjna obejmująca 200 zamówień z około 600 pozycjami, przechowywana w PostgreSQL. Badanie konfrontuje trzy pary metod o uzupełniającym się charakterze: filtracja oparta na treści vs filtracja kolaboratywna (podejście atrybutowe kontra behawioralne), logika rozmyta vs analiza sentymentu (personalizacja kontra uniwersalność), modele probabilistyczne vs reguły asocjacyjne Apriori (sekwencje temporalne kontra koszyki zakupowe).

Krótką charakterystyka metod współautora projektu

Collaborative Filtering wykorzystuje zapisy transakcyjne użytkowników do konstrukcji macierzy podobieństw produktowych w oparciu o znormalizowane podobieństwo kosinusowe (Adjusted Cosine Similarity) z centralizacją względem średnich użytkowników. Algorytm identyfikuje asocjacje niewidoczne w analizie atrybutowej - przykładowo, częste wspólne występowanie aparatów fotograficznych i plecaków turystycznych w koszykach zakupowych prowadzi do ich połączenia mimo odmiennej klasyfikacji kategorialnej. Metoda charakteryzuje się podatnością na problem zimnego startu (nowe produkty bez historii transakcyjnej) oraz wymaga znacznego wolumenu danych historycznych dla efektywnego działania. Wyniki zapisywane są w encji `ProductSimilarity` z oznaczeniem `type='collaborative'`. Algorytm wykazuje szczególną efektywność w odkrywaniu rekomendacji międzykategorialnych (*cross-category recommendations*), zwiększających wartość średniego koszyka.

Sentiment Analysis przeprowadza ekstrakcję tonu emocjonalnego z pięciu źródeł informacji: teksty opinii użytkowników (waga 40%), opis produktu (25%), nazwa produktu (15%), specyfikacje techniczne (12%) oraz nazwy kategorii (8%). Implementacja wykorzystuje słownik sentymentu Opinion Lexicon (Hu & Liu 2004) zawierający listy słów pozytywnych i negatywnych. Algorytm oblicza wynik sentymentu według wzoru: $(\text{liczba_słów_pozytywnych} - \text{liczba_słów_negatywnych}) / \text{liczba_wszystkich_słów}$, generując wskaźnik w dziedzinie $[-1, +1]$. Mechanizm agregacji wieloźródłowej syntetyzuje wszystkie sygnały w unifikowany wskaźnik `sentiment_score`, reprezentujący zagregowany ton wobec produktu. Algorytm charakteryzuje się uniwersalnością (identyczne wyniki niezależnie od użytkownika) i brakiem mechanizmów personalizacyjnych. Kluczowym ograniczeniem jest podat-

ność na konstrukcje negujące, ironiczne oraz sarkastyczne w materiale tekstowym. Rezultaty cachowane są w encji `method_product_sentiment_summary` i wykorzystywane do rangowania produktów według odbioru społecznego.

Apriori identyfikuje reguły asocjacyjne postaci „konsumenci nabywający pozycję A z dużym prawdopodobieństwem nabywają pozycję B”. Mechanizm analizuje kompozycję koszyków transakcyjnych i wykrywa zbiory produktowe występujące wspólnie z częstością przekraczającą losowe oczekiwanie. Ewaluacja opiera się na trzech metrykach: support (wsparcie - częstość występowania zbioru), confidence (ufność - prawdopodobieństwo warunkowe nabycia B przy zakupie A) oraz lift (wzmocnienie - stosunek częstości współwystępowania do niezależnego oczekiwania). Generacja reguł wymaga przekroczenia progów $\text{min_support} = 0.01$ oraz $\text{min_confidence} = 0.1$, eliminując asocjacje losowe. Algorytm efektywnie rekomenduje produkty komplementarne (akcesoria, rozszerzenia), lecz wymaga historii transakcyjnej współzakupów i wykazuje problem zimnego startu dla nowych pozycji katalogowych. Rezultaty zapisywane są w tabeli `method_productassociation`.

Porównanie par metod

Content-Based Filtering vs Collaborative Filtering

Tabela 1 zestawia charakterystyki porównawczych algorytmów. Filtracja oparta na treści ekstrahuje atrybuty bezpośrednio z danych produktowych (alokacja wag: kategoria 40%, tagi 30%, punkt cenowy 20%, deskryptory 10%) eliminując zależność od zapisu historycznego. Filtracja kolaboratywna konstruuje macierz podobieństw produktowych w oparciu o analizę zachowań transakcyjnych z wykorzystaniem znormalizowanego podobieństwa kosinusowego (Adjusted Cosine Similarity) z centralizacją per-użytkownik.

Tabela 1: Porównanie metod Content-Based Filtering i Collaborative Filtering

Kryterium	CBF (Feature-Based)	CF (Item-Based)
Źródło danych	Atrybuty produktów (cechy)	Historia zakupów (zachowania)
Pokrycie katalogu	Pełne - każdy produkt ma cechy	Częściowe - tylko produkty z historią
Problem zimnego startu	Nie występuje dla nowych produktów	Występuje dla nowych produktów
Typ powiązań	Oczywiste w kategorii (laptop A \rightarrow laptop B)	Nieoczywiste międzykategorialne (aparat \rightarrow plecak)
Algorytm	Weighted TF-IDF + Cosine Similarity	Znormalizowane podobieństwo kosinusowe + centrowanie średniej
Przechowywanie	ProductSimilarity (type='content_based')	ProductSimilarity (type='collaborative')
Złożoność	Niższa - wektory powiązanych produktów	Wyższa - macierz wszystkich produktów
Czas odpowiedzi	Średni (cache MISS: kilka sekund)	Szybki (po wytrenowaniu macierzy)
Interpretowalność	Średnia (wagi cech)	Niska (ukryte wzorce)

Różnice w działaniu: Filtracja oparta na treści sugeruje produkty o zbieżnych atrybutach - dla laptopa rekomenduje inne laptopy z tej samej klasy cenowej, kategorii oraz zbiorów tagowych. Filtracja kolaboratywna identyfikuje asocjacje niewynikające z analizy atrybutowej - przykładowo, jeśli konsumenci aparatów fotograficznych systematycznie nabywają plecaki turystyczne, algorytm wywoła połączenie mimo różnej klasyfikacji kategorialnej. Metoda oparta na treści funkcjonuje natychmiastowo (każdy produkt posiada atrybuty), podczas gdy podejście kolaboratywne wymaga znacznego wolumenu historii transakcyjnej dla efektywnego uczenia.

Zastosowanie praktyczne: W realizowanym systemie oba algorytmy funkcjonują równolegle. Metoda oparta na treści obsługuje produkty bez historii transakcyjnej oraz użytkowników niezalogowanych - zapewnia pokrycie całego katalogu (moduł „Podobne produkty” w widoku szczegółowym). Podejście kolaboratywne aktywne jest dla użytkowników uwierzytelnionych z historią zakupów - generuje sugestie oparte na wzorcach konsumpcyjnych populacji. Dla platform o ograniczonej bazie transakcyjnej metoda oparta na treści stanowi bezpieczniejszy wybór. Dla sklepów o dojrzałej historii zakupów algorytm kolaboratywny ujawnia wartościowe asocjacje międzykategorialne wspierające sprzedaż krzyżową.

Wyniki testów: Metoda oparta na treści osiągnęła wysoki zasięg katalogu - większość produktów posiada obliczone podobieństwa przekraczające próg 20%. Czas konstrukcji macierzy wyniósł około 1 minuty dla zbioru 500 produktów. Algorytm kolaboratywny, wymagający większego wolumenu danych historycznych, uzyskał niższy zasięg ze względu na ograniczoną historię transakcyjną użytkowników testowych. Metoda oparta na treści wykazuje przewagę dla produktów bez historii zakupowej, gdzie podejście kolaboratywne zwraca zbiór pusty.

Fuzzy Logic vs Sentiment Analysis

Tabela 2 konfrontuje algorytm spersonalizowany (logika rozmyta) z uniwersalnym (analiza sentymentu). Wnioskowanie rozmyte konstruuje indywidualny `FuzzyUserProfile` w oparciu o historię transakcyjną użytkownika (wskaźnik wrażliwości cenowej, preferowane klasy produktowe). Analiza sentymentu agreguje materiał tekstowy z pięciu źródeł z wykorzystaniem leksykonów Opinion Lexicon oraz AFINN-165, generując unifikowany wskaźnik dla całej populacji.

Tabela 2: Porównanie metod Fuzzy Logic i Sentiment Analysis

Kryterium	Fuzzy Logic	Sentiment Analysis
Personalizacja	Wysoka - profil użytkownika	Brak - uniwersalny wskaźnik
Źródło danych	Historia zakupów (preferencje cenowe)	Opinie tekstowe użytkowników
Kryteria oceny	Cena, jakość (rating), popularność	Sentymet tekstowy + oceny gwiazdkowe
Algorytm	System wnioskowania Mamdaniego (6 reguł IF-THEN)	Leksykony sentymentu (Opinion Lexicon, AFINN-165)
Interpretowalność	Pełna (100%) - aktywacja reguł	Średnia (wagi słów z leksykonów)
Czas odpowiedzi	Bardzo szybki (milisekundy)	Szybki (preprocessing opinii)
Problem zimnego startu	Częściowy (wymaga historii użytkownika)	Nie występuje (uniwersalny)
Kontekst	Profil cenowy użytkownika	Opinie społeczności
Typ rekomendacji	„Dopasowane do Twojego budżetu”	„Najlepiej oceniane przez innych”

Różnice w działaniu: Logika rozmyta dostosowuje sugestie do profilu wydatkowego użytkownika. Identyczny produkt może uzyskać pozytywną ocenę dla użytkownika segmentu premium („wysoka jakość adekwatna do ceny”) oraz negatywną dla użytkownika budżetowego („przekracza tolerancję cenową”). Mechanizm operuje na sześciu regułach wnioskowania IF-THEN, ewaluujących produkty według

trzech kryteriów z rozmytymi funkcjami przynależności (trójkątne, trapezoidalne). Analiza sentymentu konstruuje natomiast globalny ranking w oparciu o agregację recenzji tekstowych oraz ocen gwiazdkowych. Produkt o wysokim wskaźniku sentymentu (+0.8) rekomendowany jest identycznie dla wszystkich użytkowników.

Zastosowanie praktyczne: Wnioskowanie rozmyte udostępnione jest w dedykowanym module panelu klienckiego z trzema zakładkami (Fuzzy Recommendations – lista spersonalizowanych rekomendacji, User Profile – profil rozmyty użytkownika z wrażliwością cenową i preferowanymi kategoriami, Fuzzy Rules – opis 6 reguł IF-THEN). Metoda wymaga uwierzytelnienia użytkownika dla pełnej personalizacji opartej na historii zakupów. Dla użytkowników niezalogowanych (gości) system wykorzystuje domyślne parametry rozmyte, a administrator może ustawić logikę rozmytą jako algorytm aktywny – wówczas rekomendacje fuzzy zastępują CBF lub CF w sekcji „Recommended For You” na stronie głównej. Analiza sentymentu dostępna jest dla całej populacji użytkowników (włącznie z gośćmi) i prezentuje globalny ranking „Produkty najlepiej oceniane” w widoku głównym. Logika rozmyta wykazuje wyższą skuteczność dla użytkowników o wyraźnych wzorcach wydatkowych (segment budżetowy lub premium), podczas gdy analiza sentymentu funkcjonuje efektywnie jako uniwersalny filtr jakościowy.

Wyniki testów: Wnioskowanie rozmyte osiągnęło najkrótszy czas odpowiedzi dzięki eliminacji konieczności przechowywania macierzy - kalkulacje realizowane są dynamicznie. Wyniki odnoszą się do zaimplementowanego systemu Mamdaniego wykorzystującego 6 reguł IF-THEN oraz 3 zmienne wejściowe (punkt cenowy, wskaźnik jakości, popularność). Osiągnięto pełną wyjaśnialność (100%) - każda sugestia zawiera szczegółowe uzasadnienie aktywacji konkretnych reguł. Analiza sentymentu wymagała wstępnego przetwarzania materiału tekstowego (tokenizacja, eliminacja stop words), lecz działała uniwersalnie bez konstrukcji profili użytkowników. Kluczowym ograniczeniem analizy sentymentu jest podatność na negację oraz ironię w tekście („nie polecam” może zostać błędnie sklasyfikowane jako pozytywne przy wykryciu terminu „polecam” przez leksykon).

Modele Probabilistyczne vs Apriori

Tabela 3 konfrontuje algorytmy sekwencyjne (modele probabilistyczne) z asocjacyjnymi (Apriori). Modele probabilistyczne (łańcuchy Markowa + klasyfikator Bayesa naiwnego) prognozują przyszłe transakcje w oparciu o sekwencje temporalne oraz profil behawioralny użytkownika. Apriori identyfikuje reguły współwystępowania produktów w koszykach zakupowych bez uwzględnienia wymiaru czasowego.

Tabela 3: Porównanie modeli probabilistycznych i algorytmu Apriori

Kryterium	Modele Probabilistyczne	Apriori
Źródło danych	Sekwencje zamówień w czasie	Koszyki zakupowe (itemsety)
Uwzględnienie czasu	Tak - kolejność zakupów	Nie - współwystępowanie
Algorytm	Markov Chain (predykcja sekwencji) + Naive Bayes (prawdopodobieństwo zakupu)	Algorytm Apriori (support, confidence, lift)
Personalizacja	Wysoka - profil użytkownika	Brak - uniwersalne reguły
Typ predykcji	„Po laptopie następują akcesoria”	„Z laptopem kupują mysz”
Metryki	Prawdopodobieństwo przejścia, prawdopodobieństwo zakupu	Support, confidence, lift
Problem zimnego startu	Występuje (wymaga historii)	Występuje (wymaga współzakupów)
Czas treningu	Szybki (Markov + NB na zbiorze testowym)	Wysoki (dla dużych katalogów)
Czas predykcji	Bardzo szybki	Średni (wyszukiwanie reguł)
Zastosowanie	Przewidywanie następnej kategorii, analiza odejścia klienta	Rekomendacje produktów komplementarnych

Różnice w działaniu: Modele probabilistyczne ewaluują sekwencje transakcyjne w domenie czasowej, prognozując kolejną kategorię nabywcą (np. „po nabyciu laptopa typowo następują akcesoria komputerowe”). Proces Markowa pierwszego rzędu operuje na macierzy przejść pomiędzy 48 kategoriami, wytrenowanej na faktycznych sekwencjach zamówień. Klasyfikator Bayesa naiwnego estymuje prawdopodobieństwo transakcji w oparciu o pięć cech użytkownika: licznosc zamówień historycznych, średnia wartość transakcji, liczba dni od ostatniego zakupu, dominująca kategoria oraz częstotliwość konsumpcji. Oba modele działają synergicznie: Markov odpowiada na pytanie „jaką kategorię nabędzie użytkownik”, podczas gdy Bayes estymuje „czy użytkownik w najbliższym okresie dokona zakupu”. Apriori identyfikuje natomiast reguły postaci „konsumenci nabywający laptop często nabywają mysz bezprzewodową” bez uwzględnienia wymiaru temporalnego. Mechanizm analizuje kompozycje koszyków i wykrywa zbiory produktowe występujące wspólnie z częstością ponadprzypadkową.

Zastosowanie praktyczne: Modele probabilistyczne prezentowane są w module „Smart Recommendations” interfejsu klienckiego (panel użytkownika) z ana-

lizą sekwencji zakupowych (tekstowa prezentacja najczęstszych ścieżek pomiędzy kategoriami) oraz estymacją prawdopodobieństwa odejścia klienta (wskaźnik churn risk z wizualizacją poziomu ryzyka). Administrator dysponuje dostępem do paneli „Markov Chain Analysis” oraz „Bayesian Analysis” zawierających prognozy sprzedaży i identyfikację użytkowników zagrożonych odejściem. Apriori generuje moduł „Często kupowane razem” w widokach produktowych oraz w koszyku zakupowym, sugerując produkty komplementarne. Modele probabilistyczne wykazują wyższą skuteczność dla użytkowników o regularnej historii transakcyjnej (wykrywanie cykli zakupowych), podczas gdy Apriori efektywnie wspiera rekomendację akcesoriów oraz rozszerzeń (sprzedaż krzyżowa).

Wyniki testów: Modele probabilistyczne wymagały uczenia na zbiorze 200 zamówień. Proces treningu obejmował konstrukcję macierzy przejść Markowa (48 stanów odpowiadających kategoriom produktowym) oraz trenowanie klasyfikatora Bayesa naiwnego na cechach użytkowników. Po fazie treningu predykcje realizowane były z wysoką szybkością. Macierz przejść Markowa wykorzystywała wygładzanie Laplace’a ($\alpha = 1.0$) eliminujące zerowe prawdopodobieństwa. Apriori generował reguły asocjacyjne przy progach $\text{min_support} = 0.01$ oraz $\text{min_confidence} = 0.1$, co eliminowało asocjacje losowe, lecz wymagało wystarczającej liczności transakcji dla każdego zbioru produktowego.

Podsumowanie ewaluacji

Ewaluacja sześciu algorytmów rekomendacyjnych wykazała, że każdy z nich adresuje odmienny wymiar problematyki rekomendacji, a optymalne rezultaty osiąga się poprzez ich synergiczne zastosowanie:

Algorytmy atrybutowe vs behawioralne: Filtracja oparta na treści (analiza cech produktowych) eliminuje problem zimnego startu dla nowych pozycji katalogowych i gwarantuje pełne pokrycie. Filtracja kolaboratywna (wzorce zakupowe populacji) ujawnia nieoczywiste asocjacje międzykategorialne wspierające sprzedaż krzyżową. Metoda oparta na treści stanowi bezpieczniejszy wybór dla młodych platform, podczas gdy podejście kolaboratywne dedykowane jest dojrzałym systemom o bogatej historii transakcyjnej.

Algorytmy spersonalizowane vs uniwersalne: Logika rozmyta (profil wydatkowy użytkownika) dostosowuje sugestie według wrażliwości cenowej, oferując pełną wyjaśnialność (transparentność 100% - wizualizacja aktywacji 6 reguł IF-THEN). Analiza sentymentu (opinie społeczności) generuje globalny ranking jakościowy dostępny dla całej populacji. Wnioskowanie rozmyte wykazuje wyższą skuteczność dla użytkowników o wyrazistych preferencjach, podczas gdy analiza sentymentu funkcjonuje efektywnie jako uniwersalny filtr jakości.

Algorytmy sekwencyjne vs asocjacyjne: Modele probabilistyczne (sekwencje temporalne) prognozują przyszłe transakcje poprzez połączenie łańcuchów Markowa (predykcja kategorii) oraz profilu behawioralnego Bayesa (prawdopodobieństwo transakcji), umożliwiając identyfikację użytkowników zagrożonych churnem. Apriori (współwystępowania) identyfikuje reguły asocjacyjne produktów komplementarnych efektywne w sprzedaży krzyżowej (*cross-selling*). Modele probabilistyczne osiągają lepsze wyniki dla użytkowników o regularnej aktywności (cykle zakupowe), Apriori dla rekomendacji akcesoriów oraz rozszerzeń.

Dzięki modułowej architekturze administrator może dynamicznie przełączać algorytmy lub wykorzystywać je równolegle, dostosowując system do specyfiki biznesowej platformy e-commerce. Praktyczne testy wykazały, że system efektywnie obsługuje 500 produktów, 20 użytkowników oraz 200 zamówień, wydajnie generując rekomendacje w czasie rzeczywistym dla wszystkich sześciu algorytmów.

Rozdział 8

Podsumowanie i wnioski końcowe

Niniejsza praca przedstawiła proces implementacji oraz analizy kompletnego systemu e-commerce wyposażonego w mechanizmy rekomendacji produktów. Zaimplementowano trzy metody rekomendacyjne: Content-Based Filtering oparty na ważonych wektorach cech, system logiki rozmytej Mamdani oraz modele probabilistyczne wykorzystujące łańcuch Markowa i klasyfikator Naive Bayesa.

Ograniczenia systemu

W trakcie realizacji projektu zidentyfikowano następujące ograniczenia:

Problem zimnego startu – algorytmy Markov Chain oraz Naive Bayes wymagają historycznych danych o interakcjach użytkowników z produktami. Dla nowych użytkowników bez historii zakupów mechanizmy te nie są w stanie generować efektywnych rekomendacji. Content-Based Filtering częściowo kompensuje to ograniczenie, ponieważ może rekomendować produkty na podstawie cech (kategoria, tagi, cena), nawet dla nowych użytkowników. Fuzzy Logic działa najlepiej dla użytkowników z umiarkowaną historią zakupów.

Efekt „filter bubble” w CBF – użytkownik otrzymuje rekomendacje podobnych produktów, nie odkrywając nowych kategorii. Rozwiązanie: hybrydyzacja z innymi metodami rekomendacyjnymi.

Skalowalność dla bardzo dużych katalogów – dla katalogów produktów przekraczających 10 000 pozycji mogą wystąpić wyzwania wydajnościowe. Obecne optymalizacje (prycinanie progowe, pamięć podręczna, operacje zbiorcze) są wystarczające dla katalogów do 1 000-2 000 produktów, ale większe wymagałyby zastosowania przybliżonego wyszukiwania najbliższych sąsiadów (algorytmy LSH, HNSW) lub partycjonowania tabel PostgreSQL.

Brak obsługi kontekstu czasowego i sezonowości – system nie uwzględnia czynników sezonowych (np. zwiększone zakupy elektroniki przed świętami) ani kontekstu czasowego sesji użytkownika. Rozwiązanie: modele sekwencyjne (LSTM, GRU) lub rozszerzenie Markov Chain do wyższego rzędu.

Kierunki dalszego rozwoju

Zastosowanie głębokiego uczenia maszynowego – obecny system wykorzystuje klasyczne algorytmy rekomendacyjne. Zastosowanie sieci neuronowych, takich jak autoencodery czy sieci rekurencyjne, mogłoby umożliwić automatyczne uczenie się ukrytych wzorców w danych bez konieczności ręcznego definiowania reguł rozmytych czy wag cech.

Mechanizm hybrydowy – obecnie administrator przełącza między metodami ręcznie. System meta-learnera lub stacking ensemble mógłby automatycznie dobierać najlepszą metodę lub kombinację metod dla danego użytkownika i kontekstu.

Rekomendacje w czasie rzeczywistym – obecny system wykorzystuje pamięć podręczną z okresem ważności 5-120 minut. Implementacja systemu aktualizującego rekomendacje w czasie rzeczywistym po każdej akcji użytkownika (przeoglądanie produktów, dodawanie do koszyka) mogłaby zwiększyć trafność sugestii, ale wiązałaby się z istotnymi konsekwencjami dla wydajności systemu. Aktualizacje w czasie rzeczywistym wymagałyby ponownego przeliczania profilu użytkownika oraz rekomendacji przy każdej akcji, co dla algorytmu CBF oznaczałoby obliczanie podobieństw dla dziesiątek produktów, dla Fuzzy Logic – ewaluację 6 reguł rozmytych dla setek produktów, a dla modeli probabilistycznych – aktualizację macierzy przejść Markova i ponowny trening klasyfikatorów Naive Bayes. W scenariuszu intensywnego przeglądania (użytkownik otwiera 20-30 produktów w ciągu 5 minut) generowałoby to setek żądań obliczeniowych, potencjalnie zwiększając obciążenie serwera 10-krotnie. Rozwiązaniem kompromisowym mogłoby być wykorzystanie kolejek zadań asynchronicznych (np. Celery + Redis) do aktualizacji rekomendacji w tle z opóźnieniem 30-60 sekund, co łączyłoby korzyści personalizacji z zachowaniem akceptowalnej wydajności systemu.

Zaawansowane metody obsługi zimnego startu – zastosowanie technik faktoryzacji macierzy (SVD) lub wstępnej ankiety preferencji dla nowych użytkowników mogłoby poprawić jakość rekomendacji w pierwszych sesjach.

Wnioski końcowe

Zrealizowany system stanowi kompletne rozwiązanie e-commerce z zaawansowanymi mechanizmami rekomendacji produktów. Implementacja od podstaw bez wykorzystania gotowych bibliotek rekomendacyjnych (TensorFlow, PyTorch, Surprise) umożliwiła pełne zrozumienie mechanizmów działania algorytmów oraz ich świadome dostosowanie do specyfiki handlu elektronicznego.

Content-Based Filtering okazał się najbardziej uniwersalną metodą, rozwiązującą problem zimnego startu dla nowych produktów. Wagi cech (kategoria 40%, tagi 30%, cena 20%, słowa kluczowe 10%) zostały dobrane empirycznie i zapewniają dobrą równowagę między różnorodnością a trafnością rekomendacji.

Logika rozmyta oferuje najwyższą interpretowalność spośród zaimplementowanych metod. Każda rekomendacja ma wyjaśnienie w postaci aktywacji konkretnych reguł IF-THEN, co jest istotne z perspektywy GDPR (prawo do wyjaśnienia decyzji algorytmicznych) oraz budowania zaufania użytkowników do systemu.

Modele probabilistyczne umożliwiają najgłębszą personalizację dla użytkowników z bogatą historią zakupów. Łańcuch Markowa przewiduje sekwencje zakupowe na poziomie kategorii produktów, Naive Bayes ocenia prawdopodobieństwo zakupu i ryzyko odejścia klienta.

Komplementarność zastosowanych metod – Content-Based Filtering dla nowych produktów, Fuzzy Logic dla personalizacji z interpretowalnością, modele probabilistyczne dla głębokiej analizy behawioralnej – zapewnia wszechstronne wsparcie procesu decyzyjnego użytkownika. Zastosowane techniki optymalizacyjne (cache, bulk operations, threshold pruning, indeksowanie) gwarantują akceptowalne czasy odpowiedzi systemu nawet przy większych katalogach produktów.

Praca wykazała, że implementacja systemu rekomendacyjnego od podstaw jest możliwa i celowa w kontekście edukacyjnym oraz w sytuacjach wymagających pełnej kontroli nad logiką biznesową. Zrealizowany projekt pozwolił na zdobycie praktycznej wiedzy w zakresie projektowania systemów rekomendacyjnych, optymalizacji algorytmów oraz rozwoju aplikacji full-stack (Django + React + PostgreSQL + Docker).

System stanowi kompleksowe rozwiązanie e-commerce z trzema komplementarnymi metodami rekomendacyjnymi, gotowe do wdrożenia w środowisku produkcyjnym.

Literatura

- [1] Pazzani, M. J., & Billsus, D. (2007). Content-Based Recommendation Systems. *The Adaptive Web*, Springer, pp. 325-341.
- [2] Zadeh, L. A. (1965). Fuzzy Sets. *Information and Control*, 8(3), pp. 338-353.
- [3] Mamdani, E. H., & Assilian, S. (1975). An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. *International Journal of Man-Machine Studies*, 7(1), pp. 1-13.
- [4] Rabiner, L. R. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2), pp. 257-286.
- [5] Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press.
- [6] Ricci, F., Rokach, L., & Shapira, B. (2015). *Recommender Systems Handbook*. Springer.
- [7] Gomez-Urbe, C. A., & Hunt, N. (2016). The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Transactions on Management Information Systems*, 6(4), pp. 1-19.
- [8] Klement, E. P., Mesiar, R., & Pap, E. (2000). *Triangular Norms*. Springer.
- [9] Salton, G., & Buckley, C. (1988). Term-Weighting Approaches in Automatic Text Retrieval. *Information Processing & Management*, 24(5), pp. 513-523.
- [10] Ross, T. J. (2010). *Fuzzy Logic with Engineering Applications*. Wiley, 3rd Edition.
- [11] McKinsey & Company. (2013). Big Data: The Next Frontier for Innovation, Competition, and Productivity.
- [12] Linden, G., Smith, B., & York, J. (2003). Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing*, 7(1), pp. 76-80.
- [13] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-Based Collaborative Filtering Recommendation Algorithms. *Proceedings of WWW*, pp. 285-295.
- [14] Agrawal, R., & Srikant, R. (1994). Fast Algorithms for Mining Association Rules. *Proceedings of VLDB*, pp. 487-499.
- [15] Mendel, J. M. (2001). *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions*. Prentice Hall.

Wykaz rysunków i tabel

Spis rysunków

1	Diagram przypadków użycia systemu.	26
2	Diagram ERD głównych tabel aplikacji.	28
3	Diagram ERD tabel metod rekomendacyjnych.	29
4	Deployment aplikacji w architekturze Docker Compose.	38
5	Diagram sekwencji - Content-Based Filtering.	45
6	Funkcje przynależności dla ceny produktu (cheap, medium, expensive). . .	50
7	Funkcje przynależności dla jakości produktu (low, medium, high).	50
8	Funkcje przynależności dla popularności produktu (low, medium, high). . .	51
9	Diagram sekwencji - Fuzzy Logic.	55
10	Diagram sekwencji - Probabilistic Models.	61
11	Rekomendacje Content-Based Filtering wyświetlane użytkownikowi na stronie głównej.	64
12	Panel debugowania Content-Based Filtering.	65
13	CBF - szczegółowa analiza podobieństwa produktu.	67
14	Panel klienta - rekomendacje Fuzzy Logic.	69
15	Profil użytkownika Fuzzy Logic.	70
16	Reguły wnioskowania Fuzzy Logic.	71
17	Wyszukiwarka rozmyta (Fuzzy Search).	72
18	Panel debugowania Fuzzy Logic – widok ogólny profilu gościa.	74
19	Fuzzy Logic – stopnie przynależności dla wybranego produktu.	75
20	Fuzzy Logic – wnioskowanie rozmyte i defuzyfikacja.	76
21	Zakładka "Next Purchase (Markov)".	78
22	Zakładka "Behavior Insights (Bayesian)".	79
23	Zakładka "Churn Risk Analysis".	80
24	Panel administracyjny - Markov Analysis.	82
25	Panel administracyjny - Bayesian Analysis.	83
26	Panel debugowania - widok ogólny dla łańcucha Markova.	84
27	Panel debugowania - widok szczegółowy dla Naive Bayes.	85

Spis tabel

1	Porównanie metod Content-Based Filtering i Collaborative Filtering	89
2	Porównanie metod Fuzzy Logic i Sentiment Analysis	90
3	Porównanie modeli probabilistycznych i algorytmu Apriori	92

Streszczenie

Tytuł pracy w języku polskim:

System rekomendacji produktów wykorzystujący filtrację opartą na treści, logikę rozmytą i modele probabilistyczne

Tytuł pracy w języku angielskim:

Product Recommendation System Utilizing Content-Based Filtering, Fuzzy Logic, and Probabilistic Models

Streszczenie:

Niniejsza praca inżynierska przedstawia projekt oraz implementację systemu rekomendacji produktów dla platformy e-commerce, łączącego trzy komplementarne metody: filtrację opartą na treści (Content-Based Filtering), logikę rozmytą (Fuzzy Logic) oraz modele probabilistyczne (Markov Chain i Naive Bayes). Celem było zaprojektowanie rozwiązania eliminującego problem przeładowania informacyjnego w sklepach internetowych poprzez dostarczanie użytkownikom spersonalizowanych rekomendacji.

Część teoretyczna obejmuje przegląd systemów rekomendacyjnych oraz analizę rozwiązań alternatywnych (Amazon Personalize, Google Recommendations AI, Apache Mahout) wraz z uzasadnieniem implementacji dedykowanego systemu. Przedstawiono fundament matematyczny wykorzystanych algorytmów: podobieństwo kosinusowe dla ważonych wektorów cech w Content-Based Filtering, funkcje przynależności trójkątne i trapezoidalne z systemem wnioskowania Mamdaniego dla logiki rozmytej oraz macierz przejść stanów i twierdzenie Bayesa dla modeli probabilistycznych.

Część projektowa obejmuje szczegółowy projekt architektury systemu w modelu trójwarstwowym (warstwa prezentacji React 18, warstwa logiki biznesowej Django 5.1.4, warstwa danych PostgreSQL 14), projekt struktury bazy danych z tabelami dla prekalkulowanych wyników algorytmów, projekt interfejsów użytkownika (widoki użytkownika końcowego i panele administracyjne) oraz projekt mechanizmów optymalizacyjnych (pamięć podręczna, indeksowanie, operacje zbiorcze).

Część implementacyjna przedstawia realizację aplikacji webowej w architekturze Django REST Framework (backend) oraz React 18 (frontend). System integruje trzy metody działające komplementarnie w różnych kontekstach: Content-Based Filtering dla rozwiązania problemu zimnego startu nowych produktów, logikę rozmytą dla personalizacji z pełną interpretowalnością reguł IF-THEN oraz modele probabilistyczne dla predykcji sekwencji zakupowych i prawdopodobieństwa odejścia klienta. Zaimplementowano kompletny interfejs z narzędziami debugowania oraz panel administracyjny umożliwiający dynamiczne przełączanie metod rekomendacyjnych. Aplikacja została skonteneryzowana w Docker Compose, co zapewnia powtarzalność środowiska deweloperskiego i produkcyjnego.

Wartością pracy jest implementacja algorytmów od podstaw, co umożliwiło głębokie zrozumienie mechanizmów oraz świadome dostosowanie do specyfiki e-commerce.

Załącznik A

Szczegółowe scenariusze użycia

Poniżej przedstawiono szczegółowe opisy scenariuszy użycia systemu odpowiadających przypadkom użycia zaprezentowanym na diagramie w rozdziale 3.3. Scenariusze zostały opisane zgodnie z notacją: Aktor, Warunki początkowe, Przebieg scenariusza głównego, Warunki końcowe.

Scenariusz 1: Przeglądanie i wyszukiwanie produktów

Aktor: Gość (użytkownik niezalogowany)

Warunki początkowe: Użytkownik znajduje się na stronie głównej aplikacji

Przebieg scenariusza głównego:

1. Użytkownik otwiera stronę główną aplikacji w przeglądarce
2. Użytkownik otwiera pasek wyszukiwania z standardową opcją
3. Użytkownik wpisuje szukaną frazę
4. System sortuje produkty według wartości `sentiment_score` malejąco
5. Użytkownik przegląda listę produktów
6. Użytkownik klika na wybrany produkt
7. System wyświetla szczegółową stronę produktu z opisem, specyfikacjami, zdjęciami i opiniami

Warunki końcowe: Użytkownik widzi szczegółowe informacje o wybranym produkcie

Scenariusz 2: Dodawanie produktów do koszyka

Aktor: Gość (użytkownik niezalogowany)

Warunki początkowe: Użytkownik przegląda szczegółową stronę produktu (laptop)

Przebieg scenariusza głównego:

1. Użytkownik klika przycisk „Dodaj do koszyka”
2. System dodaje produkt do koszyka i wyświetla powiadomienie o sukcesie
3. Użytkownik klika ikonę koszyka w nawigacji
4. System wyświetla widok koszyka z dodanymi produktami
5. System automatycznie wywołuje algorytm Apriori dla produktów w koszyku

6. System wyświetla sekcję „Frequently Bought Together” z trzema produktami rekomendowanymi przez algorytm Apriori (torba na laptopa, mysz bezprzewodowa, hub USB-C)
7. Każda rekomendacja zawiera metryki: lift, confidence oraz przycisk „Dodaj do koszyka”
8. Użytkownik przegląda rekomendacje
9. Użytkownik klika „Dodaj do koszyka” przy torbie i myszy
10. System dodaje wybrane produkty do koszyka i aktualizuje sumę całkowitą

Warunki końcowe: Koszyk zawiera laptop, torbę i mysz, wyświetlona jest zaktualizowana suma

Scenariusz 3: Rejestracja nowego użytkownika

Aktor: Gość (użytkownik niezalogowany)

Warunki początkowe: Użytkownik znajduje się na stronie głównej

Przebieg scenariusza głównego:

1. Użytkownik klika przycisk „Zarejestruj się” w nawigacji
2. System wyświetla formularz rejestracji
3. Użytkownik wypełnia pola: imię, nazwisko, adres e-mail, hasło, potwierdzenie hasła
4. Użytkownik klika przycisk „Zarejestruj”
5. System waliduje dane: sprawdza czy e-mail jest unikalny, czy hasła się zgadzają, czy hasło spełnia wymagania bezpieczeństwa
6. System tworzy nowe konto użytkownika w bazie danych
7. System wyświetla komunikat o pomyślnej rejestracji
8. System automatycznie loguje użytkownika i generuje token JWT
9. System przekierowuje użytkownika do panelu klienta

Warunki końcowe: Użytkownik jest zalogowany i posiada aktywne konto w systemie

Scenariusz 4: Logowanie do systemu

Aktor: Gość (użytkownik niezalogowany, posiadający konto)

Warunki początkowe: Użytkownik znajduje się na stronie głównej

Przebieg scenariusza głównego:

1. Użytkownik klika przycisk „Zaloguj się” w nawigacji
2. System wyświetla formularz logowania
3. Użytkownik wpisuje adres e-mail i hasło
4. Użytkownik klika przycisk „Zaloguj”
5. System weryfikuje dane logowania w bazie danych
6. System generuje token JWT i zwraca go do klienta
7. Frontend zapisuje token w localStorage przeglądarki
8. System przekierowuje użytkownika do panelu klienta

Warunki końcowe: Użytkownik jest zalogowany, posiada ważny token JWT

Scenariusz 5: Śledzenie statusu zamówienia

Aktor: Klient (użytkownik zalogowany)

Warunki początkowe: Użytkownik jest zalogowany, posiada co najmniej jedno zamówienie

Przebieg scenariusza głównego:

1. Użytkownik otwiera panel klienta
2. Użytkownik klika zakładkę „My Orders”
3. System pobiera wszystkie zamówienia użytkownika z bazy danych
4. System wyświetla listę zamówień posortowaną od najnowszych
5. Dla każdego zamówienia wyświetlane są: numer zamówienia, status (oczekujące/w realizacji/zakończone/anulowane)

Warunki końcowe: Użytkownik widzi szczegółowe informacje o statusie wybranego zamówienia

Scenariusz 6: Wystawianie opinii o produkcie

Aktor: Klient (użytkownik zalogowany)

Warunki początkowe: Użytkownik posiada zakończone zamówienie zawierające produkt

Przebieg scenariusza głównego:

1. Użytkownik otwiera panel klienta, zakładkę „My Orders”
2. Użytkownik znajduje zakończone zamówienie
3. Użytkownik klika przycisk „Add Review” przy zakupionym produkcie

4. System wyświetla formularz opinii
5. Użytkownik wybiera ocenę gwiazdkową (1-5 gwiazdek)
6. Użytkownik wpisuje treść tekstową opinii
7. Użytkownik klika „Wyślij opinię”
8. System zapisuje opinię w bazie danych z powiązaniem: użytkownik, produkt, zamówienie
9. System automatycznie wywołuje algorytm analizy sentymentu dla nowej opinii
10. Algorytm tokenizuje tekst, wyszukuje słowa w słowniku pozytywnym/negatywnym
11. Algorytm oblicza `sentiment_score` dla opinii zgodnie ze wzorem (2)
12. System przelicza zagregowany sentyment produktu zgodnie ze wzorem (3) (wieloźródłowa agregacja z 5 źródeł)
13. System aktualizuje pole `sentiment_score` w rekordzie produktu
14. System wyświetla komunikat o pomyślnym dodaniu opinii
15. System odświeża widok produktu - zaktualizowany sentyment jest widoczny

Warunki końcowe: Opinia została dodana, sentyment produktu zaktualizowany

Scenariusz 7: Zarządzanie kontem użytkownika

Aktor: Klient (użytkownik zalogowany)

Warunki początkowe: Użytkownik jest zalogowany

Przebieg scenariusza głównego:

1. Użytkownik otwiera panel klienta
2. Użytkownik klika zakładkę „Konto”
3. System wyświetla formularz z danymi użytkownika: imię, nazwisko, e-mail
4. Użytkownik edytuje wybrane pola (np. zmienia nazwisko)
5. Użytkownik klika „Zapisz zmiany”
6. System waliduje dane (sprawdza czy e-mail jest unikalny, jeśli został zmieniony)
7. System aktualizuje rekord użytkownika w bazie danych
8. System wyświetla komunikat o pomyślnym zapisaniu zmian

Warunki końcowe: Dane użytkownika zostały zaktualizowane

Scenariusz 8: Administrator zarządza produktami

Aktor: Administrator

Warunki początkowe: Administrator jest zalogowany, posiada uprawnienia administratora

Przebieg scenariusza głównego - dodawanie produktu:

1. Administrator otwiera panel administracyjny
2. Administrator klika sekcję „Products”
3. System wyświetla listę wszystkich produktów
4. Administrator klika przycisk „Add New Product”
5. System wyświetla formularz dodawania produktu
6. Administrator wypełnia pola: nazwa, opis, cena, kategorie, specyfikacje techniczne, przesyła zdjęcie
7. Administrator klika „Zapisz produkt”
8. System waliduje dane (sprawdza czy wszystkie wymagane pola są wypełnione)
9. System tworzy nowy rekord produktu w bazie danych
10. System automatycznie wywołuje algorytm analizy sentymentu dla nowego produktu
11. Algorytm oblicza sentiment_score na podstawie nazwy i opisu (brak jeszcze opinii)
12. System zapisuje obliczony sentyment w rekordzie produktu
13. System wyświetla komunikat o pomyślnym dodaniu produktu
14. System odświeża listę produktów

Warunki końcowe: Nowy produkt został dodany do katalogu z obliczonym sentymentem

Scenariusz 9: Administrator zarządza zamówieniami

Aktor: Administrator

Warunki początkowe: Administrator jest zalogowany

Przebieg scenariusza głównego:

1. Administrator otwiera panel administracyjny
2. Administrator klika sekcję „Orders”
3. System wyświetla listę wszystkich zamówień w systemie
4. Dla każdego zamówienia wyświetlane są: numer, aktualny status

5. Administrator klika na wybrane zamówienie ze statusem „oczekujące”
6. Administrator klika przycisk „Zmień status”
7. System wyświetla listę dostępnych statusów (w realizacji, zakończone, anulowane)
8. Administrator wybiera „w realizacji”
9. System aktualizuje status zamówienia w bazie danych
10. System zapisuje zdarzenie zmiany statusu z timestampem
11. System wyświetla komunikat o pomyślnej zmianie statusu
12. Użytkownik właściciel zamówienia widzi zaktualizowany status w swoim panelu

Warunki końcowe: Status zamówienia został zmieniony, użytkownik ma dostęp do aktualnej informacji

Scenariusz 10: Administrator zarządza użytkownikami

Aktor: Administrator

Warunki początkowe: Administrator jest zalogowany

Przebieg scenariusza głównego:

1. Administrator otwiera panel administracyjny
2. Administrator klika sekcję „Users”
3. System wyświetla listę wszystkich użytkowników
4. Dla każdego użytkownika wyświetlane są: ID, imię, nazwisko, e-mail, rola, data rejestracji
5. Administrator może filtrować użytkowników według roli (klient/administrator)
6. Administrator klika na wybranego użytkownika
7. System wyświetla szczegóły użytkownika: profil, historia zamówień, statystyki
8. Administrator może zmienić rolę użytkownika (nadać uprawnienia administratora) lub usunąć konto

Warunki końcowe: Administrator ma pełny wgląd w dane użytkowników i może zarządzać uprawnieniami

Scenariusz 11: Administrator przegląda statystyki i dashboard

Aktor: Administrator

Warunki początkowe: Administrator jest zalogowany

Przebieg scenariusza głównego:

1. Administrator otwiera panel administracyjny
2. System automatycznie wyświetla dashboard ze statystykami
3. Dashboard zawiera następujące kluczowe metryki:
 - Całkowita liczba produktów w katalogu
 - Całkowita liczba zarejestrowanych użytkowników
 - Całkowita liczba zamówień (wszystkie statusy)
 - Całkowity przychód (suma wartości zamówień zakończonych)
4. System wyświetla wykresy:
 - Wykres słupkowy: miesięczny obrót (ostatnie 12 miesięcy)
 - Wykres kołowy: rozkład zamówień według kategorii produktów
 - Wykres liniowy: liczba nowych użytkowników (ostatnie 30 dni)
5. Administrator klika sekcję „Statistics” dla bardziej szczegółowych analiz
6. System wyświetla widok do zmiany rekomendacji, odświeżanie reguł asocjacyjnych

Warunki końcowe: Administrator ma pełny wgląd w kluczowe wskaźniki biznesowe systemu

Scenariusz 12: Administrator debuguje algorytmy rekomendacji

Aktor: Administrator

Warunki początkowe: Administrator jest zalogowany, algorytmy rekomendacji zostały przeliczone

Przebieg scenariusza głównego - Collaborative Filtering:

1. Administrator otwiera panel administracyjny
2. Administrator klika sekcję „Debug”
3. Administrator wybiera zakładkę „Collaborative Filtering”
4. System wyświetla panel debugowania Collaborative Filtering zawierający:

- Wzór matematyczny Adjusted Cosine Similarity
 - Statystyki: liczba par produktów, średnie podobieństwo, czas ostatniego przeliczenia
 - Tabelę z przykładowymi podobieństwami (product_a, product_b, similarity_score)
5. Administrator może wpisać ID produktu i zobaczyć top 10 najbardziej podobnych produktów
 6. System pobiera dane z tabeli ProductSimilarity i wyświetla wyniki

Warunki końcowe: Administrator zweryfikował poprawność działania algorytmu Collaborative Filtering

Scenariusz 13: Przeglądanie rekomendacji na stronie głównej

Aktor: Klient (użytkownik zalogowany)

Warunki początkowe: Użytkownik jest zalogowany, posiada historię zakupów

Przebieg scenariusza głównego:

1. Użytkownik loguje się do systemu
2. System przekierowuje użytkownika do strony głównej
3. System wywołuje algorytm Collaborative Filtering dla zalogowanego użytkownika
4. Algorytm analizuje historię zakupów użytkownika
5. Algorytm pobiera produkty podobne do wcześniej zakupionych (na podstawie macierzy podobieństw)
6. System wyświetla sekcję „Rekomendowane dla Ciebie” z 6 produktami
7. Użytkownik przegląda rekomendacje
8. Użytkownik klika na wybrany produkt
9. System wyświetla szczegółową stronę produktu

Warunki końcowe: Użytkownik widzi spersonalizowane rekomendacje oparte na swojej historii zakupów

Scenariusz 14: Przeglądanie podobnych produktów (Content-Based Filtering)

Aktor: Klient lub Gość

Warunki początkowe: Użytkownik przegląda szczegółową stronę produktu (np. laptop Dell XPS 15)

Przebieg scenariusza głównego:

1. Użytkownik otwiera stronę szczegółową produktu
2. System automatycznie wywołuje endpoint GET `/api/recommendations/content-based/?product_id={id}`
3. Backend wywołuje metodę `ContentBasedAPI` z parametrem `product_id`
4. System wykonuje zapytanie do tabeli `ProductSimilarity` filtrując po `product1_id = {id}` oraz `type='content_based'`
5. System sortuje wyniki malejąco według `similarity_score`
6. System pobiera Top 10 najbardziej podobnych produktów z bazy danych
7. Backend zwraca JSON z listą produktów zawierającą: `product_id`, `name`, `price`, `similarity_score`
8. Frontend wyświetla sekcję „Podobne produkty” z kafelkami produktów
9. Każdy kafelek zawiera zdjęcie, nazwę, cenę oraz wskaźnik podobieństwa (np. „85% podobny”)
10. Użytkownik przegląda rekomendacje
11. Użytkownik klika na wybrany podobny produkt
12. System przekierowuje do strony szczegółowej wybranego produktu

Warunki końcowe: Użytkownik widzi produkty podobne do oglądanego, obliczone na podstawie cech (kategorie 40%, tagi 30%, cena 20%, słowa kluczowe 10%)

Scenariusz 15: Generowanie macierzy podobieństw

Content-Based Filtering

Aktor: Administrator

Warunki początkowe: Administrator jest zalogowany, katalog zawiera produkty

Przebieg scenariusza głównego:

1. Administrator otwiera panel administracyjny
2. Administrator klika sekcję „Admin Tools”
3. Administrator wybiera zakładkę „Content-Based Filtering”
4. System wyświetla panel zarządzania CBF z przyciskiem „Generate Similarity Matrix”
5. Administrator klika „Generate Similarity Matrix”
6. System wywołuje endpoint POST `/api/admin/content-based/generate`

7. Backend uruchamia algorytm `ContentBasedRecommendationEngine.generate_similarities()`
8. System pobiera wszystkie produkty z bazy danych
9. Dla każdego produktu system ekstrahuje cechy:
 - Kategorie (waga 40%) - konwersja do wektora one-hot
 - Tagi (waga 30%) - ekstrakcja unikalnych tagów
 - Przedział cenowy (waga 20%) - normalizacja do zakresu [0,1]
 - Słowa kluczowe z opisu (waga 10%) - tokenizacja i TF-IDF
10. System oblicza podobieństwo cosinusowe między każdą parą produktów zgodnie ze wzorem Weighted Cosine Similarity
11. System filtruje pary z podobieństwem $> 20\%$ (próg minimalny)
12. System usuwa stare rekordy z tabeli `ProductSimilarity` dla `type='content_based'`
13. System zapisuje nowe podobieństwa do tabeli `ProductSimilarity`
14. System wyświetla komunikat o sukcesie: „Wygenerowano 1247 podobieństw dla 500 produktów w czasie 58 sekund”
15. Administrator widzi zaktualizowane statystyki: liczba par produktów, średnie podobieństwo, timestamp ostatniego przeliczenia

Warunki końcowe: Macierz podobieństw została wygenerowana i zapisana w bazie danych, rekomendacje CBF są aktualne

Scenariusz 16: Debugowanie algorytmu Content-Based Filtering

Aktor: Administrator

Warunki początkowe: Administrator jest zalogowany, macierz podobieństw została wygenerowana

Przebieg scenariusza głównego:

1. Administrator otwiera panel administracyjny
2. Administrator klika sekcję „Debug”
3. Administrator wybiera zakładkę „Content-Based Debug”
4. System wyświetla panel debugowania CBF zawierający:
 - Wzór matematyczny Weighted Cosine Similarity z opisem wag cech
 - Statystyki globalne: liczba produktów w katalogu, liczba wygenerowanych par, średnie podobieństwo, czas ostatniego przeliczenia

- Histogram rozkładu podobieństw (przedziały: 20-40%, 40-60%, 60-80%, 80-100%)
5. Administrator wprowadza ID produktu w pole „Product ID for analysis”
 6. Administrator klika „Analyze Product”
 7. System wywołuje endpoint `GET /api/admin/content-based/debug/?product_id={id}`
 8. System pobiera wybrany produkt oraz jego Top 10 najbardziej podobnych produktów
 9. System wyświetla szczegółową analizę dla wybranego produktu:
 - Podstawowe dane: nazwa, kategorie, tagi, cena
 - Ekstrahowane cechy: wektor kategorii, lista tagów, znormalizowana cena, słowa kluczowe
 - Tabelę Top 10 podobnych produktów z kolumnami: ID, nazwa, `similarity_score`, wspólne kategorie, wspólne tagi
 10. Administrator analizuje wyniki i weryfikuje poprawność podobieństw
 11. Administrator może wyeksportować dane do CSV klikając „Export to CSV”

Warunki końcowe: Administrator zweryfikował poprawność działania algorytmu CBF dla wybranego produktu

Scenariusz 17: Przeglądanie rekomendacji Fuzzy Logic w panelu klienta

Aktor: Klient (użytkownik zalogowany)

Warunki początkowe: Użytkownik jest zalogowany, posiada historię zakupów

Przebieg scenariusza głównego:

1. Użytkownik loguje się do systemu
2. Użytkownik otwiera panel klienta
3. Użytkownik klika zakładkę „Fuzzy Logic Recommendations”
4. System wywołuje endpoint `GET /api/fuzzy-logic-recommendations/?limit=10`
5. Backend wywołuje `FuzzyLogicAPI` z parametrem `user_id` (z tokenu JWT)
6. System inicjalizuje silnik rozmyty `FuzzyEngine.initialize_fuzzy_system(user_id)`
7. System buduje profil użytkownika `FuzzyUserProfile`:
 - Oblicza parametr `price_sensitivity` na podstawie średniej ceny zakupionych produktów

- Ekstrahuje preferowane kategorie z historii zakupów (top 3 kategorie)
8. System pobiera wszystkie produkty z bazy danych z metrykami: cena, średni rating, liczba zamówień
 9. Dla każdego produktu system wykonuje wnioskowanie rozmyte:
 - Fuzyfikacja - oblicza przynależność ceny, jakości i popularności do zbiorów rozmytych (cheap/medium/expensive, low/medium/high)
 - Aplikacja 6 reguł IF-THEN typu Mamdani (R1: preferred category match, R2: quality-price balance, R3: price sensitive match, R4: premium match, R5: popular high quality, R6: budget friendly)
 - Agregacja wyników reguł z wagami (R1: 0.9, R2: 0.85, R3: 0.6, R4: 0.75, R5: 0.8, R6: 0.7)
 - Defuzyfikacja - oblicza końcowy **fuzzy_score** metodą średniej ważonej
 10. System sortuje produkty malejąco według **fuzzy_score**
 11. System zwraca JSON z Top 10 rekomendacji zawierający: **product_id**, **name**, **price**, **fuzzy_score**, **activated_rules**
 12. Frontend wyświetla trzy zakładki:
 - „Fuzzy Recommendations lista Top 10 produktów z fuzzy_score i aktywowanymi regułami
 - „User Profile wyświetla **price_sensitivity** (np. 60% - moderate) oraz preferowane kategorie
 - „Fuzzy Rules opis 6 reguł IF-THEN z wyjaśnieniem logiki
 13. Użytkownik przegląda rekomendacje dopasowane do jego profilu cenowego
 14. Użytkownik klika przycisk „View Rule Activations” przy wybranym produkcie
 15. System wyświetla szczegóły aktywacji reguł: które reguły zostały aktywowane, z jaką siłą (α_i), jaki był ich wkład w końcowy wynik

Warunki końcowe: Użytkownik widzi spersonalizowane rekomendacje dopasowane do jego wrażliwości cenowej oraz preferowanych kategorii z pełną przejrzystością procesu decyzyjnego

Scenariusz 18: Wyszukiwanie tolerujące błędy (Fuzzy Search)

Aktor: Klient lub Gość

Warunki początkowe: Użytkownik znajduje się na stronie głównej

Przebieg scenariusza głównego:

1. Użytkownik otwiera pasek wyszukiwania
2. Użytkownik przełącza opcję wyszukiwania na „Fuzzy Search”
3. System wyświetla suwak „Fuzzy Threshold”(domyślnie $0.5 = 50\%$)
4. Użytkownik wpisuje frazę z literówką: „loptap”(zamiast „laptop”)
5. Użytkownik ustawia próg podobieństwa na 0.3 (30%)
6. Użytkownik klika przycisk „Search”
7. System wywołuje endpoint `GET /api/search/fuzzy/?query=loptap&threshold=0.3`
8. Backend iteruje po wszystkich produktach w katalogu
9. Dla każdego produktu system oblicza odległość Levenshteina między zapytaniem a nazwą produktu:
 - Algorytm rekurencyjny oblicza minimalną liczbę operacji edycji (wstawienie, usunięcie, zamiana znaku)
 - System normalizuje odległość do zakresu $[0,1]$ dzieląc przez maksymalną długość
 - Oblicza `fuzzy_match_score = 1 - normalized_distance`
10. System filtruje produkty z `fuzzy_match_score \geq 0.3` (próg użytkownika)
11. System sortuje wyniki malejąco według `fuzzy_match_score`
12. Backend zwraca JSON z produktami: `product_id, name, price, fuzzy_match_score`
13. Frontend wyświetla wyniki wyszukiwania:
 - Laptop Dell XPS 15 - 86% match (zawiera „Laptop”)
 - Laptop HP Pavilion - 86% match (zawiera „Laptop”)
 - Desktop Hub USB-C - 45% match (częściowe dopasowanie)
14. Użytkownik widzi produkty mimo błędu w zapytaniu
15. Użytkownik klika na wybrany produkt

Warunki końcowe: Użytkownik znalazł poszukiwane produkty mimo literówki dzięki algorytmowi Levenshteina

Scenariusz 19: Debugowanie algorytmu Fuzzy Logic

Aktor: Administrator

Warunki początkowe: Administrator jest zalogowany, system posiada użytkowników z historią zakupów

Przebieg scenariusza głównego:

1. Administrator otwiera panel administracyjny
2. Administrator klika sekcję „Debug”
3. Administrator wybiera zakładkę „Fuzzy Logic Debug”
4. System wyświetla widok ogólny panelu debugowania zawierający:
 - Szczegóły algorytmu: metoda Mamdani Fuzzy Inference, liczba reguł (6), T-norma (min), T-conorma (max)
 - Definicje funkcji przynależności dla price (cheap: 0-500 PLN, medium: 300-1500 PLN, expensive: 1000+ PLN)
 - Definicje funkcji przynależności dla quality (low: 0-2.5, medium: 2.0-4.0, high: 3.5-5.0)
 - Definicje funkcji przynależności dla popularity (low: 0-10 zamówień, medium: 5-50, high: 30+ zamówień)
 - Listę profili użytkowników z parametrem price_sensitivity
5. Administrator wprowadza ID produktu w pole „Product ID for detailed analysis”
6. Administrator klika „Analyze Product”
7. System wywołuje endpoint `GET /api/admin/fuzzy-debug/?product_id={id}`
8. System pobiera wybrany produkt oraz wykonuje szczegółową ewaluację rozmytą
9. System wyświetla widok szczegółowy dla produktu:
 - Dane produktu: nazwa, cena, średni rating, liczba zamówień
 - Wartości fuzzyfikacji:
 - Price: cheap=0.3, medium=0.7, expensive=0.0
 - Quality: low=0.0, medium=0.2, high=0.8
 - Popularity: low=0.0, medium=0.6, high=0.4
 - Aktywacja wszystkich 6 reguł z siłą aktywacji α_i :
 - R1 (Preferred Category Match): $\alpha_1 = 0.85$, waga = 0.9, wkład = 0.765
 - R2 (Quality-Price Balance): $\alpha_2 = 0.7$, waga = 0.85, wkład = 0.595
 - R3 (Price Sensitive Match): $\alpha_3 = 0.3$, waga = 0.6, wkład = 0.18

- R4 (Premium Match): $\alpha_4 = 0.0$, waga = 0.75, wkład = 0.0
 - R5 (Popular High Quality): $\alpha_5 = 0.6$, waga = 0.8, wkład = 0.48
 - R6 (Budget Friendly): $\alpha_6 = 0.2$, waga = 0.7, wkład = 0.14
 - Agregacja: suma wkładów = 2.16, fuzzy_score = $2.16 / 4.6 = 0.47$ (47%)
10. Administrator analizuje które reguły dominują dla danego produktu
 11. Administrator może dostroić wagi reguł klikając „Edit Rule Weights”
 12. System zapisuje nowe wagi w konfiguracji

Warunki końcowe: Administrator zweryfikował proces wnioskowania rozmytego i zoptymalizował wagi reguł

Scenariusz 20: Przeglądanie rekomendacji probabilistycznych w panelu klienta

Aktor: Klient (użytkownik zalogowany)

Warunki początkowe: Użytkownik jest zalogowany, posiada historię zakupów, modele probabilistyczne zostały wytrenowane

Przebieg scenariusza głównego:

1. Użytkownik loguje się do systemu
2. Użytkownik otwiera panel klienta
3. Użytkownik klika zakładkę „Probabilistic Recommendations”
4. System wywołuje endpoint `GET /api/probabilistic-recommendations/?user_id={id}`
5. Backend wywołuje `ProbabilisticAPI` z parametrem `user_id`
6. System pobiera historię zamówień użytkownika z bazy danych wywołując `get_user_orders(user_id)`
7. System ekstrahuje sekwencję kategorii produktów z zamówień uporządkowanych chronologicznie
8. System wywołuje `MarkovChain.predict_next_products(order_sequence)`:
 - Identyfikuje ostatnią zakupioną kategorię (np. „Laptopy”)
 - Odczytuje macierz przejść dla kategorii „Laptopy”
 - Pobiera Top 3 kategorie z najwyższym prawdopodobieństwem przejścia (np. Akcesoria komputerowe: 0.45, Peryferia: 0.30, Torby: 0.15)
 - Dla każdej kategorii pobiera Top produkty
9. System wywołuje `NaiveBayes.predict_purchase_probability(user_profile)`:

- Ekstrahuje 5 cech użytkownika: liczba zamówień, średnia wartość, dni od ostatniego zakupu, ulubiona kategoria, częstotliwość
 - Stosuje twierdzenie Bayesa dla klasyfikacji binarnej (kupi/nie kupi)
 - Oblicza prawdopodobieństwo zakupu w najbliższym okresie
10. System agreguje wyniki: 60% waga dla Markov, 40% waga dla Naive Bayes
 11. Backend zwraca JSON z Top 10 rekomendacji zawierający: `product_id`, `name`, `price`, `markov_probability`, `bayes_probability`, `combined_score`
 12. Frontend wyświetla trzy zakładki:
 - „Probabilistic Recommendations lista Top 10 produktów z prawdopodobieństwami
 - „Markov Chain Visualization graf przejść między kategoriami z prawdopodobieństwami
 - „Purchase Probability Analysis wykres prawdopodobieństwa zakupu oraz analiza cech użytkownika
 13. Użytkownik przegląda rekomendacje oparte na sekwencjach zakupowych
 14. Użytkownik klika zakładkę „Markov Chain Visualization"
 15. System wyświetla interaktywny graf pokazujący możliwe przejścia z obecnej kategorii do kolejnych
 16. Użytkownik klika zakładkę „Purchase Probability Analysis"
 17. System wyświetla:
 - Prawdopodobieństwo zakupu: 73% (high probability)
 - Breakdown cech: liczba zamówień: 12 (pozytywny wpływ +0.15), średnia wartość: 450 PLN (pozytywny +0.10), dni od ostatniego: 15 (pozytywny +0.20)
 - Rekomendacja: „Użytkownik jest aktywny, wysokie prawdopodobieństwo konwersji"

Warunki końcowe: Użytkownik widzi rekomendacje przewidujące jego przyszłe zakupy oparte na łańcuchach Markowa i klasyfikatorze Bayesa

Scenariusz 21: Trenowanie modeli probabilistycznych przez administratora

Aktor: Administrator

Warunki początkowe: Administrator jest zalogowany, system posiada historię zamówień

Przebieg scenariusza głównego:

1. Administrator otwiera panel administracyjny
2. Administrator klika sekcję „Admin Tools”
3. Administrator wybiera zakładkę „Probabilistic Models”
4. System wyświetla panel zarządzania modelami probabilistycznymi
5. Administrator widzi status modeli:
 - Markov Chain: ostatni trening 2024-01-15, liczba stanów: 48 kategorii, wygładzanie Laplace’a: $\alpha = 1.0$
 - Naive Bayes: ostatni trening 2024-01-15, liczba próbek treningowych: 200 zamówień, accuracy: 78%
6. Administrator klika przycisk „Train Markov Chain Model”
7. System wywołuje endpoint `POST /api/admin/probabilistic/train-markov`
8. Backend uruchamia `MarkovChainEngine.train()`:
 - Pobiera wszystkie zamówienia z bazy danych uporządkowane chronologicznie per użytkownik
 - Ekstraktuje sekwencje kategorii dla każdego użytkownika
 - Buduje macierz przejść 48x48 (48 kategorii produktowych)
 - Oblicza prawdopodobieństwa przejść: $P(s_j|s_i) = \frac{\text{count}(s_i \rightarrow s_j) + \alpha}{\sum_k \text{count}(s_i \rightarrow s_k) + \alpha \cdot N}$
 - Stosuje wygładzanie Laplace’a ($\alpha = 1.0$) zapobiegające zerowemu prawdopodobieństwu
 - Zapisuje macierz w cache Django z timeout 24h
9. System wyświetla komunikat: „Markov Chain trained successfully. 48 states, 2304 transitions, training time: 3.2s”
10. Administrator klika przycisk „Train Naive Bayes Model”
11. System wywołuje endpoint `POST /api/admin/probabilistic/train-bayes`
12. Backend uruchamia `NaiveBayesEngine.train()`:
 - Pobiera wszystkich użytkowników z zamówieniami
 - Dla każdego użytkownika ekstraktuje 5 cech: liczba zamówień, średnia wartość, dni od ostatniego, ulubiona kategoria, częstotliwość
 - Tworzy etykiety binarne (aktywny/nieaktywny) na podstawie heurystyki (zakup w ostatnich 30 dniach)
 - Oblicza prawdopodobieństwa warunkowe dla każdej cechy: $P(feature|class)$

- Oblicza prawdopodobieństwa a priori: $P(active)$, $P(inactive)$
- Waliduje model na zbiorze testowym (80%/20% split)
- Zapisuje model w tabeli cache

13. System wyświetla komunikat: „Naive Bayes trained successfully. Training samples: 200, Test accuracy: 78%, training time: 1.8s”

14. Administrator widzi zaktualizowane timestampy i metryki

Warunki końcowe: Modele probabilistyczne (Markov Chain i Naive Bayes) zostały wytrenowane na aktualnych danych, rekomendacje są aktualne

Scenariusz 22: Debugowanie modeli probabilistycznych

Aktor: Administrator

Warunki początkowe: Administrator jest zalogowany, modele zostały wytrenowane

Przebieg scenariusza głównego - Markov Chain:

1. Administrator otwiera panel administracyjny
2. Administrator klika sekcję „Debug”
3. Administrator wybiera zakładkę „Markov Chain Debug”
4. System wyświetla panel debugowania zawierający:
 - Wzór matematyczny macierzy przejść z wygładzaniem Laplace’a
 - Statystyki: liczba stanów (48 kategorii), łączna liczba przejść, wygładzanie $\alpha = 1.0$, timestamp ostatniego treningu
 - Wizualizację macierzy przejść jako heatmap 48x48
5. Administrator wybiera kategorię źródłową z listy rozwijanej (np. „Laptopy”)
6. System wyświetla Top 10 kategorii docelowych z najwyższymi prawdopodobieństwami przejścia:
 - Akcesoria komputerowe: 0.45 (45%)
 - Peryferia: 0.30 (30%)
 - Torby i plecaki: 0.15 (15%)
 - Oprogramowanie: 0.05 (5%)
 - Inne kategorie: <0.05 (łącznie 5%)
7. Administrator klika „Show Transition Examples”
8. System pobiera przykładowe rzeczywiste sekwencje zamówień użytkowników, które wygenerowały te przejścia

9. Administrator weryfikuje sensowność przejść
10. Administrator przełącza zakładkę na „Naive Bayes Debug”
11. System wyświetla panel Naive Bayes zawierający:
 - Wzór twierdzenia Bayesa
 - Statystyki: liczba próbek treningowych, accuracy, precision, recall, F1-score
 - Tabelę prawdopodobieństw warunkowych $P(feature|class)$ dla każdej z 5 cech
 - Prawdopodobieństwa a priori: $P(active) = 0.65$, $P(inactive) = 0.35$
12. Administrator wprowadza ID użytkownika w pole „User ID for prediction”
13. System pobiera profil użytkownika i wykonuje predykcję krok po kroku:
 - Ekstrahuje cechy użytkownika
 - Pokazuje obliczenia $P(active|features)$ z rozpisaniem każdego czynnika
 - Wyświetla końcowe prawdopodobieństwo oraz klasyfikację (active/inactive)
14. Administrator weryfikuje poprawność predykcji

Warunki końcowe: Administrator zweryfikował poprawność działania modeli probabilistycznych (Markov Chain i Naive Bayes)