UNIWERSYTET RZESZOWSKI
Wydział Nauk Ścisłych i Technicznych

Dawid Olko
nr albumu: 125148
Kierunek: Informatyka

System rekomendacji produktów oparty na filtracji współpracy, analizie sentymentu i regułach asocjacyjnych

Praca inżynierska

Praca wykonana pod kierunkiem
dr inż. Piotra Grochowalskiego

Rzeszów, 2026

Spis treści

Wstęp

Kontekst i motywacja

Nowoczesne platformy e-commerce oferują tysiące lub dziesiątki tysięcy produktów, co stanowi istotne wyzwanie zarówno dla klientów, jak i właścicieli sklepów internetowych. Użytkownik poszukujący smartfona staje przed wyborem setek modeli, w przypadku laptopów sytuacja wygląda podobnie. Bez wsparcia inteligentnych systemów rekomendacyjnych proces zakupowy staje się czasochłonny i frustrujący, co często prowadzi do rezygnacji z zakupu. Z perspektywy biznesowej oznacza to utratę potencjalnych klientów oraz sytuacje, w których nabywcy nie odkrywają produktów optymalnie dopasowanych do ich potrzeb.

Systemy rekomendacyjne stanowią rozwiązanie tego problemu poprzez automatyczną analizę historii zakupów, opinii oraz zachowań użytkowników w celu proponowania produktów o najwyższej wartości dla konkretnego klienta.

Cel pracy

Celem niniejszej pracy jest zaprojektowanie, implementacja oraz analiza kompletnego systemu e-commerce wyposażonego w autorskie mechanizmy rekomendacji produktów. Aplikacja została opracowana od podstaw jako autorskie rozwiązanie wykonane we współpracy dwuosobowej. W ramach systemu zaimplementowano łącznie sześć różnych metod rekomendacyjnych — trzech z nich dotyczy ta praca. Ich założeniem jest zapewnienie komplementarnych mechanizmów wsparcia decyzji zakupowych. Niniejsza praca koncentruje się na trzech spośród nich: Collaborative Filtering, analizie sentymentu oraz regułach asocjacyjnych. Mechanizmy rekomendacyjne zostały zaprojektowane i zaimplementowane samodzielnie na podstawie literatury naukowej, bez wykorzystania gotowych bibliotek rekomendacyjnych, co umożliwiło pełną kontrolę nad logiką algorytmów oraz ich świadome dostosowanie do specyfiki branży handlu elektronicznego.

Zakres pracy

Zakres funkcjonalny systemu obejmuje:
Implementację trzech metod rekomendacyjnych:

- Collaborative Filtering w wariancie Item-Based z metryką Adjusted Cosine Similarity [8] — metoda analizuje wzorce zakupowe użytkowników w celu identyfikacji produktów podobnych do wcześniej nabytych,

- Analizę sentymentu opartą na podejściu słownikowym [5] — metoda agreguje ocenę jakości produktu z pięciu źródeł tekstowych (opinie, opis, nazwa, specyfikacje, kategorie), rozwiązując problem zimnego startu dla produktów bez historii opinii,
- Reguły asocjacyjne wykorzystujące algorytm Apriori [1] — metoda odkrywa wzorce współwystępowania produktów w koszyku zakupowym, wspierając strategie cross-sellingu.

Opracowanie kompletnej aplikacji webowej:

- Backend oparty na Django REST Framework zapewniający API dla wszystkich funkcjonalności systemu,
- Frontend w technologii React 18 oferujący responsywny interfejs użytkownika,
- Baza danych PostgreSQL z odpowiednio zaprojektowanym schematem przechowującym dane produktów, użytkowników, zamówień oraz wyniki algorytmów rekomendacyjnych.

Rozdział 1

Teoretyczne podstawy systemów rekomendacyjnych

1.1 Historia i ewolucja systemów rekomendacyjnych

Systemy rekomendacyjne powstały jako odpowiedź na problem wyboru spośród tysięcy produktów w sklepach internetowych. Rozwój tej dziedziny rozpoczął się w latach 90. XX wieku wraz z pojawieniem się pierwszych platform e-commerce. Wczesne zastosowania komercyjne systemów rekomendacji zostały opisane w literaturze specjalistycznej [4]. Istotnym przełomem była również publikacja wprowadzająca metodę Item-Based Collaborative Filtering z metryką Adjusted Cosine Similarity [8], która stała się standardem przemysłowym. Konkursy i inicjatywy badawcze, takie jak Netflix Prize w latach 2006-2009, w którym fundusz nagród wynosił milion dolarów, znacząco przyspieszyły rozwój zaawansowanych technik rekomendacji [2], przyciągając uwagę zarówno środowiska akademickiego, jak i przemysłowego. Obecnie systemy rekomendacyjne stanowią kluczowy element wiodących platform e-commerce (handlu elektronicznego) oraz serwisów VOD (Video on Demand - wideo na żądanie).

1.2 Klasyfikacja metod rekomendacyjnych

Istnieją trzy główne kategorie systemów rekomendacyjnych:

Collaborative Filtering — jedna z najpopularniejszych metod w systemach komercyjnych. Zakłada, że użytkownicy o podobnych preferencjach będą mieli podobne wybory w przyszłości. Istnieją dwa warianty: oparty na użytkownikach (User-Based, porównuje użytkowników) oraz oparty na produktach (Item-Based, porównuje produkty). Zalety: odkrywa nieoczywiste powiązania między produktami. Wady: problem zimnego startu (ang. cold start) dla nowych użytkowników i produktów, macierz danych jest rzadka (0.1-1% wypełnienia).

Content-Based Filtering — analizuje cechy produktów i dopasowuje je do profilu użytkownika. Zalety: brak problemu zimnego startu dla nowych produktów. Wady: rekomenduje tylko podobne produkty, co może prowadzić do zjawiska tzw. filter bubble (bańki filtrującej).

Metody Hybrydowe — łączą różne podejścia rekomendacyjne w celu wykorzystania zalet poszczególnych metod oraz kompensacji ich wad. Przykładem może być połączenie Collaborative Filtering z analizą metadanych produktów oraz analizą treści tekstowych.

Systemy rekomendacyjne w e-commerce wykorzystują różne strategie sprzedażowe. Poniżej znajdują się kluczowe terminy stosowane w branży:

Cross-selling (sprzedaż krzyżowa) — strategia polegająca na proponowaniu produktów komplementarnych, czyli dopełniających zakup główny. Przykład: klient kupuje laptop, system proponuje mysz, torbę na laptop, podkładkę pod mysz. Celem jest zwiększenie wartości koszyka poprzez dodanie produktów powiązanych funkcjonalnie.

Up-selling (sprzedaż wyższej wartości) — strategia zachęcania klienta do zakupu droższego wariantu produktu lub wersji premium. Przykład: klient przegląda telefon za 2000 zł, system proponuje model za 2500 zł z lepszymi parametrami. Celem jest zwiększenie wartości pojedynczego zakupu, przy czym skuteczność tej strategii zależy od indywidualnych preferencji i możliwości finansowych użytkownika.

Personalizacja — dostosowanie treści i rekomendacji do indywidualnego profilu użytkownika na podstawie jego historii zakupów, przeglądanych produktów i zachowań. Przykład: dwóch użytkowników widzi różne zestawy produktów na stronie głównej. Celem jest zwiększenie trafności rekomendacji i konwersji.

Cold start problem (problem zimnego startu) — sytuacja występująca gdy nowy użytkownik lub produkt nie ma historii interakcji. Przykład: nowy użytkownik nie ma zamówień, więc Collaborative Filtering nie może efektywnie generować rekomendacji. Nowy produkt nie ma opinii, co utrudnia ocenę jego jakości. Rozwiązaniem tego problemu może być wykorzystanie metod opartych na analizie treści produktu, takich jak analiza sentymentu opisów, nazw i specyfikacji technicznych.

Frequently Bought Together (często kupowane razem) — rodzaj rekomendacji prezentujący produkty, które klienci regularnie kupują w tym samym koszyku. Przykład: laptop + mysz + podkładka pod mysz. Celem jest uproszczenie procesu zakupów i zwiększenie wartości koszyka.

1.3 Matematyczne fundamenty algorytmów

Matematyczne podstawy trzech implementowanych algorytmów stanowią fundament dla szczegółowych opisów w kolejnych rozdziałach.

Adjusted Cosine Similarity dla Item-Based Collaborative Filtering (Sarwar et al. 2001) [8] stanowi kluczową metrykę podobieństwa wykorzystywaną w systemie. Wzór ten oblicza podobieństwo między dwoma produktami i i j poprzez analizę wzorców ich współwystępowania w zakupach użytkowników:

$$\text{sim}(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R}_u)(R_{u,j} - \bar{R}_u)}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R}_u)^2} \cdot \sqrt{\sum_{u \in U}(R_{u,j} - \bar{R}_u)^2}} \tag{1}$$

gdzie $R_{u,i}$ to ilość zakupu użytkownika u dla produktu i, $\bar{R}_u$ to średnia użytkownika u, a U to użytkownicy, którzy kupili oba produkty. Centrowanie średniej $(R_{u,i} - \bar{R}_u)$ eliminuje bias czyli wartość progową dla użytkowników kupujących systematycznie więcej.

Analiza sentymentu (Liu 2012) [5] używa formuły polarności tekstu:

$$S(text) = \frac{N_{pos} - N_{neg}}{N_{total}} \tag{2}$$

gdzie N_{pos} to liczba słów pozytywnych, N_{neg} negatywnych, N_{total} to wszystkie słowa. Wynik: $[-1, 1]$ (dodatnie = pozytywny, ujemne = negatywny).

Reguły asocjacyjne (Agrawal & Srikant 1994) [1] używają trzech metryk:

Wsparcie (Support) (Agrawal & Srikant 1994) [1] - jaka jest częstość współwystępowania:

$$\text{Support}(A, B) = \frac{\text{transakcje z } A \text{ i } B}{\text{wszystkie transakcje}} \tag{3}$$

Pewność (Confidence) (Agrawal & Srikant 1994) [1] - jakie jest prawdopodobieństwo warunkowe:

$$\text{Confidence}(A \to B) = \frac{\text{Support}(A, B)}{\text{Support}(A)} \tag{4}$$

Wzrost (Lift) (Agrawal & Srikant 1994) [1] - ile razy bardziej jest prawdopodobny zakup:

$$\text{Lift}(A \to B) = \frac{\text{Support}(A, B)}{\text{Support}(A) \cdot \text{Support}(B)} \tag{5}$$

Wzrost(Lift) > 1: pozytywna korelacja, Wzrost $= 1$: niezależność, Wzrost < 1: negatywna korelacja. Algorytm Apriori przyspiesza obliczenia dzięki własności: jeśli zbiór nie spełnia minimalnego wsparcia (Support), jego nadzbiór też nie.

Rozdział 2

Weryfikacja i analiza rozwiązań alternatywnych

W celu uzasadnienia sensowności tworzenia dedykowanego systemu rekomendacji przeprowadzono analizę trzech reprezentatywnych rozwiązań rynkowych. Celem weryfikacji było zidentyfikowanie ograniczeń istniejących narzędzi oraz określenie wymagań dla planowanej aplikacji e-commerce wykorzystującej trzy metody: Collaborative Filtering [8], analizę sentymentu [5] oraz reguły asocjacyjne [1].

2.1 Amazon Personalize

Amazon Personalize to zarządzana usługa AWS oferująca systemy rekomendacji oparte na algorytmach stosowanych w Amazon.com [4]. System wykorzystuje deep learning (głębokie uczenie - wielowarstwowe sieci neuronowe) oraz collaborative filtering, oferując trzy typy rekomendacji: User Personalization (personalizacja użytkownika), Similar Items (podobne produkty) oraz Personalized Ranking (spersonalizowane rankowanie).

Kluczowe ograniczenia w kontekście planowanego rozwiązania:

- **Wysokie koszty operacyjne** - rozwiązania chmurowe wiążą się z regularnymi opłatami licencyjnymi, które mogą być znaczące dla małych i średnich platform e-commerce. Dla porównania, własna implementacja eliminuje te koszty przy zachowaniu kontroli nad funkcjonalnościami,

- **Brak natywnej analizy sentymentu** - Amazon Personalize koncentruje się wyłącznie na collaborative filtering i nie oferuje analizy opinii produktów. Wieloźródłowa agregacja sentymentu (opinie + opis + nazwa + specyfikacje + kategorie) zastosowana w niniejszej pracy wymaga integracji z dodatkowymi usługami AWS lub samodzielnej implementacji,

- **Vendor lock-in** (uzależnienie od dostawcy) - głęboka integracja z ekosystemem AWS (S3, Lambda, EventBridge) oznacza, że migracja do innej platformy wymaga przepisania całej architektury systemu,

- **Brak kontroli nad algorytmami** - system działa jako „czarna skrzynka”’ (black box), uniemożliwiając dostosowanie logiki rekomendacji. Przykład: niemożliwe jest zaimplementowanie Adjusted Cosine Similarity z centrowaniem średniej dla eliminacji wartości progowej (bias) wynikającej z różnych skal zakupowych użytkowników (hurtownik vs konsument indywidualny),

- **Wymóg dużych zbiorów danych** - według dokumentacji AWS, system wymaga minimum 25000 interakcji dla zapewnienia wysokiej jakości rekomendacji. Dla nowych platform (cold start - problem zimnego startu) jakość jest ograniczona w początkowym okresie działania.

2.2 Google Recommendations AI (Vertex AI)

Google Recommendations AI to platforma GCP wykorzystująca deep learning oraz multi-armed contextual bandits (wieloramienne bandyty kontekstowe - algorytmy balansujące eksplorację nowych opcji z wykorzystaniem sprawdzonych rozwiązań). System oferuje zaawansowane rekomendacje dla e-commerce, VOD (Video on Demand - wideo na żądanie) oraz platform newsowych, z automatycznym wykrywaniem trendów i sezonowości.

Kluczowe ograniczenia w kontekście planowanego rozwiązania:

- **Bardzo wysokie koszty operacyjne** - koszty usług GCP są często wyższe niż konkurencyjnych rozwiązań chmurowych, co czyni je nieopłacalnymi dla małych i średnich platform e-commerce,
- **Brak wieloźródłowej agregacji sentymentu** - Google Recommendations AI oferuje funkcję "Frequently Bought Together" (często kupowane razem, podobną do algorytmu Apriori), ale nie wspiera agregacji sentymentu z wielu źródeł tekstowych jak w planowanym systemie,
- **Wymóg bardzo dużych zbiorów danych** - rozwiązanie zaprojektowane dla platform o skali YouTube, co czyni je nadmiarowo złożonym dla małych sklepów internetowych,
- **Brak interpretowalności** - głęboka „black box"', gdzie nawet administratorzy z dostępem do Vertex AI nie mogą zobaczyć wag embeddings (reprezentacji wektorowych) ani logiki sieci neuronowej, co uniemożliwia debugowanie i optymalizację.

2.3 Apache Mahout

Apache Mahout to open-source framework (otwartoźródłowy framework) implementujący klasyczne algorytmy collaborative filtering [8] oraz matrix factorization (faktoryzacja macierzy - technika dekompozycji macierzy user-item) - ALS (Alternating Least Squares - metoda najmniejszych kwadratów na przemian), SVD (Singular Value Decomposition - rozkład według wartości osobliwych). Projekt powstał w 2008 roku, obecnie koncentruje się na Spark-based distributed algorithms (algorytmy rozproszone oparte na Apache Spark).

Kluczowe ograniczenia w kontekście planowanego rozwiązania:

- **Wymóg zaawansowanej wiedzy technicznej** - konieczność konfiguracji klastra Apache Spark (środowisko przetwarzania rozproszonego), YARN resource manager (zarządca zasobów), oraz monitoringu. Według Stack Overflow Developer Survey 2023, bardzo mała część programistów ma doświadczenie z Apache Spark,
- **Koszty infrastruktury** - utrzymanie klastra Spark wymaga dedykowanych zasobów serwerowych oraz czasu na implementację integracji (REST API, baza danych, cache, frontend), co generuje znaczące koszty operacyjne,
- **Brak analizy sentymentu** - Apache Mahout nie oferuje sentiment analysis. Wymagana jest integracja z zewnętrznymi bibliotekami (np. Stanford CoreNLP) lub samodzielna implementacja słownikowej analizy sentymentu,
- **Wolniejszy rozwój projektu** - aktywność projektu spadła w ostatnich latach (2-3 commity miesięcznie w 2023-2024 vs 20-30 commitów w latach 2012-2014), co skutkuje ograniczoną dokumentacją dla nowszych wersji.

2.4 Podsumowanie analizy i uzasadnienie własnego rozwiązania

Analiza rozwiązań alternatywnych ujawniła fundamentalny kompromis: **zaawansowanie technologiczne vs koszty i elastyczność**. Rozwiązania chmurowe od Amazona oraz Google oferują wysoką jakość rekomendacji dzięki algorytmom deep learning, ale wiążą się z wysokimi kosztami operacyjnymi, vendor lock-in (uzależnieniem od dostawcy) oraz brakiem kontroli nad algorytmami. Apache Mahout eliminuje koszty licencyjne, ale wymaga zaawansowanej wiedzy technicznej oraz kosztownej infrastruktury Spark.

Uzasadnienie sensowności własnego rozwiązania:

- **Integracja trzech komplementarnych metod w jednym systemie**:

  Żadne z analizowanych rozwiązań nie oferuje natywnej integracji Collaborative Filtering + Sentiment Analysis + Apriori:

  - Amazon Personalize: tylko Collaborative Filtering, wymagane dodatkowe usługi dla Sentiment Analysis i Apriori,
  - Google Recommendations AI: Collaborative Filtering + funkcja podobna do Apriori, brak analizy sentymentu,

  - Apache Mahout: tylko Collaborative Filtering, brak Sentiment Analysis i Apriori w najnowszej wersji.

- **Optymalizacja kosztów dla małych i średnich platform**:

  Własna implementacja (Django + PostgreSQL) eliminuje koszty licencyjne rozwiązań chmurowych przy zachowaniu wysokiej jakości rekomendacji. System jest szczególnie atrakcyjny dla małych i średnich platform e-commerce (do 10000 produktów, do 10000 użytkowników), które potrzebują zaawansowanych funkcjonalności rekomendacji przy ograniczonym budżecie.

- **Kontrola nad logiką biznesową i możliwość dostosowania**:

  Własna implementacja umożliwia unikalne podejścia niedostępne w gotowych rozwiązaniach:

  - **Wielośródłowa agregacja sentymentu** [5] z 5 źródeł tekstowych - rozwiązuje problem cold start (zimny start): produkty bez opinii użytkowników otrzymują wynik sentymentu na podstawie opisu, nazwy i specyfikacji,
  - **Bitmap pruning** [9] dla algorytmu Apriori - optymalizacja przyspiesza generowanie reguł asocjacyjnych względem implementacji naiwnej poprzez operacje bitowe,
  - **Adjusted Cosine Similarity** [8] z centrowaniem średniej - eliminacja wartości progowej (bias) wynikającej z różnych skal zakupowych użytkowników. Centrowanie średniej usuwa ten efekt skali przy obliczaniu podobieństwa.

- **Elastyczność technologiczna i brak vendor lock-in**:

  Aplikacja oparta na Django + React + PostgreSQL może być wdrożona na dowolnej platformie: AWS, GCP, Azure, własne serwery lub localhost. Migracja między platformami wymaga jedynie zmiany parametrów połączenia - logika rekomendacji pozostaje niezmieniona.

  Dla porównania: migracja z Amazon Personalize do Google Recommendations AI wymaga przepisania całej integracji (śledzenie zdarzeń (event tracking), dane treningowe, wywołania API) oraz retrainingu modeli, co może trwać tygodnie i powodować degradację jakości rekomendacji.

Podsumowanie:

Własna implementacja systemu rekomendacji stanowi optymalny wybór dla małych i średnich platform e-commerce, łączący:

- Wysoką jakość rekomendacji (trzy komplementarne metody: Collaborative Filtering, Sentiment Analysis, Apriori),
- Pełną kontrolę nad algorytmami i możliwość dostosowania do specyfiki biznesowej,
- Niskie koszty operacyjne (brak opłat licencyjnych rozwiązań chmurowych),
- Interpretowalność wyników i możliwość debugowania,
- Elastyczność technologiczną (brak uzależnienia od konkretnego dostawcy chmury).

Rozwiązanie jest szczególnie atrakcyjne dla platform potrzebujących zaawansowanych funkcjonalności rekomendacji przy ograniczonym budżecie oraz możliwości dostosowania logiki do specyficznych wymagań biznesowych.

Rozdział 3

Opis projektu planowanej aplikacji

Rozdział przedstawia szczegółowy opis planowanej aplikacji e-commerce z zaawansowanym systemem rekomendacji produktów. Zaprezentowano diagram przypadków użycia ilustrujący interakcje użytkowników z systemem oraz opisano kluczowe funkcjonalności z perspektywy różnych typów użytkowników.

3.1 Cel i zakres aplikacji

Aplikacja stanowi kompleksowe rozwiązanie e-commerce integrujące trzy komplementarne metody rekomendacji produktów:

- **Collaborative Filtering** - odkrywanie produktów podobnych na podstawie wzorców zakupowych użytkowników,
- **Sentiment Analysis** - ocena jakości produktów poprzez analizę opinii i treści,
- **Association Rules (Apriori)** - identyfikacja produktów często kupowanych razem.

System został zaprojektowany z myślą o małych i średnich platformach e-commerce (do 10000 produktów, do 100000 użytkowników), zapewniając funkcjonalności rekomendacyjne porównywalne z rozwiązaniami enterprise przy znacznie niższych kosztach operacyjnych.

3.2 Wymagania funkcjonalne systemu

System został zaprojektowany z uwzględnieniem następujących wymagań funkcjonalnych podzielonych według typów użytkowników:

Dla gości (użytkowników niezalogowanych):

- Przeglądanie katalogu produktów,
- Wyszukiwanie produktów z wykorzystaniem różnych metod sortowania,
- Dodawanie produktów do koszyka,
- Logowanie do systemu,
- Rejestracja w systemie.

Dla klientów (użytkowników zalogowanych):

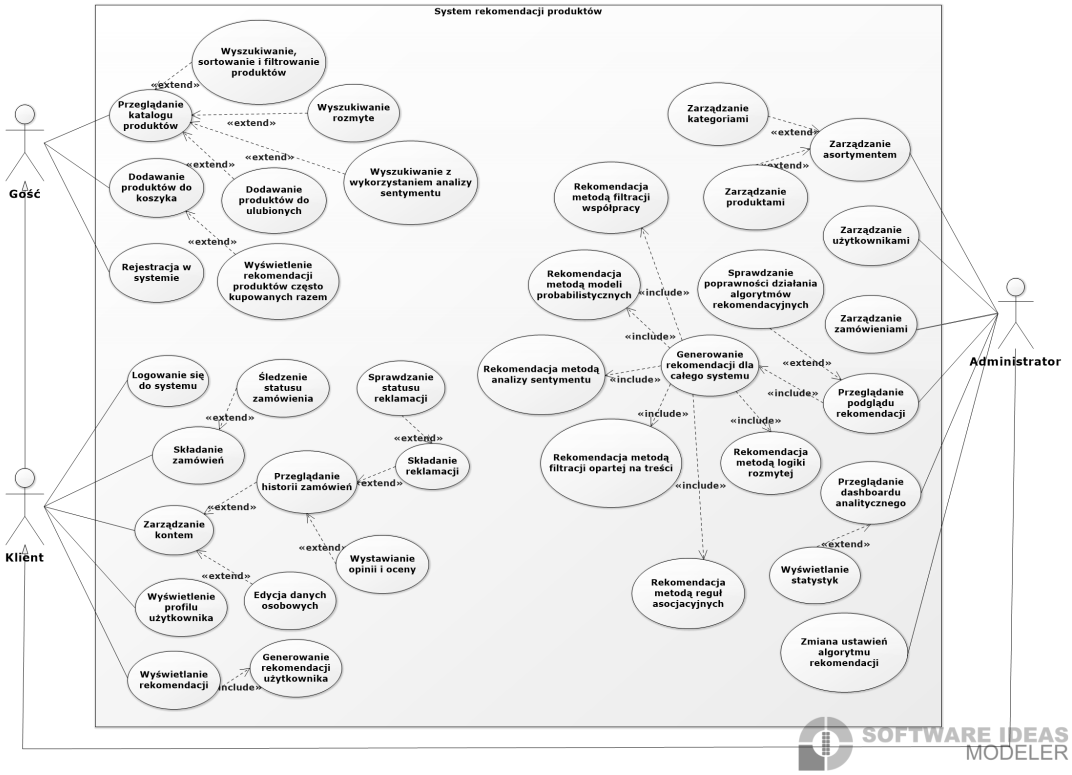
- Wszystkie funkcjonalności gościa,
- Składanie zamówień,
- Śledzenie statusu zamówień,
- Zarządzanie kontem użytkownika,
- Wyświetlanie profilu użytkownika,
- Wyświetlanie historii zamówień,
- Edycja danych osobowych,
- Przeglądanie rekomendacji w sekcjach strony głównej,
- Rekomendacje metody reguł asocjacyjnych w koszyku.

Dla administratorów:

- Zarządzanie kontem administratora,
- Zarządzanie produktami (dodawanie, edycja, usuwanie),
- Zarządzanie zamówieniami użytkowników,
- Zmiana statusów zamówień,
- Zarządzanie użytkownikami,
- Przeglądanie statystyk sprzedaży,
- Przeglądanie panelu analitycznego (dashboard),
- Wyświetlanie statystyk,
- Sprawdzanie statusu zamówień,
- Debugowanie algorytmów rekomendacji:
  - Przeglądanie podglądu rekomendacji,
  - Sprawdzanie poprawności działania algorytmów rekomendacyjnych,
  - Wyświetlanie statystyk metod rekomendacji.

3.3 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).



Rysunek 1: Diagram przypadków użycia systemu.

Ogólny opis diagramu:

Diagram został zorganizowany wokół trzech głównych aktorów, z których każdy ma dostęp do różnych poziomów funkcjonalności systemu. Relacje dziedziczenia między aktorami (Gość → Klient → Administrator) odzwierciedlają hierarchię uprawnień - każdy następny poziom dziedziczy wszystkie funkcjonalności poprzedniego i dodaje nowe, specyficzne dla swojej roli.

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 generowanych przez trzy zaimplementowane algorytmy rekomendacyjne.

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

Klient - użytkownik zalogowany, który dziedziczy wszystkie funkcje gościa oraz dodatkowo może:

- **Składać zamówienia** - finalizacja zakupu produktów z koszyka,
- **Śledzić status zamówień** - monitoring postępu realizacji zamówień (oczekujące, w realizacji, zakończone, anulowane),
- **Zarządzać kontem** - edycja danych osobowych, zmiana hasła,
- **Wyświetlać profil użytkownika** - podgląd informacji o koncie,
- **Przeglądać historię zamówień** - dostęp do wszystkich wcześniejszych transakcji,
- **Edytować dane osobowe** - aktualizacja imienia, nazwiska, adresu e-mail,
- **Przeglądać rekomendacje** - dostęp do spersonalizowanych sugestii produktów generowanych przez:
  - *Collaborative Filtering* - rekomendacje oparte na historii zakupów użytkownika i podobieństwie do innych klientów,
  - *Metodę reguł asocjacyjnych (Apriori)* - produkty często kupowane razem, rekomendacje typu "Często kupowane razem",
  - *Metodę analizy sentymentu* - produkty o najlepszych opiniach dopasowane do preferencji użytkownika.

Administrator - użytkownik z najwyższymi uprawnieniami, który dziedziczy wszystkie funkcje klienta oraz dodatkowo ma dostęp do:

- **Zarządzanie kontem administratora** - edycja ustawień konta administracyjnego,
- **Zarządzanie produktami** - pełny zakres operacji CRUD:
  - Dodawanie produktów do katalogu,
  - Edycja istniejących produktów (nazwa, opis, cena, kategorie),
  - Usuwanie produktów z potwierdzeniem.
- **Zarządzanie zamówieniami** - administracja transakcjami użytkowników:
  - Przeglądanie wszystkich zamówień w systemie,

  - Zmiana statusów zamówień (oczekujące → w realizacji → zakończone lub anulowane),
  - Generowanie raportów sprzedażowych.

- **Zarządzanie użytkownikami** - administracja kontami:
  - Lista wszystkich użytkowników,
  - Nadawanie uprawnień,
  - Usuwanie użytkowników.

- **Przeglądanie statystyk** - dostęp do zaawansowanych analiz:
  - Panel analityczny (dashboard) z kluczowymi wskaźnikami: liczba produktów, użytkowników, zamówień,
  - Wykresy sprzedaży - miesięczny obrót, najpopularniejsze kategorie.

- **Debugowanie algorytmów rekomendacji** - narzędzia diagnostyczne:
  - *Podgląd rekomendacji* - sprawdzenie jakie produkty są rekomendowane dla konkretnego użytkownika lub produktu,
  - *Sprawdzanie poprawności* - walidacja czy algorytmy działają zgodnie z oczekiwaniami,
  - *Wyświetlanie statystyk metod* - szczegółowe metryki dla każdej metody rekomendacji.

Relacje między przypadkami użycia:
Diagram wyraźnie pokazuje hierarchię dziedziczenia funkcjonalności:

- **Gość** → **Klient**: relacja "extends" - Klient dziedziczy wszystkie funkcje Gościa (przeglądanie, wyszukiwanie, koszyk, logowanie, rejestracja) oraz otrzymuje dostęp do funkcji transakcyjnych (zamówienia, historia, rekomendacje),
- **Klient** → **Administrator**: relacja "extends" - Administrator dziedziczy wszystkie funkcje Klienta oraz otrzymuje narzędzia zarządzania systemem (produkty, zamówienia, użytkownicy, statystyki, debugowanie).

3.4 Architektura funkcjonalna systemu

System został zaprojektowany w architekturze warstwowej, gdzie każda warstwa odpowiada za konkretny aspekt funkcjonalności:

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

- **Panel klienta** - panel główny (dashboard) z historią zamówień, sekcja rekomendacji, edycja profilu,
- **Panel administracyjny** - zarządzanie produktami/zamówieniami/użytkownikami, statystyki, panele debugowania.

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

- **Moduł Collaborative Filtering** - generowanie macierzy podobieństwa produktów, rekomendacje oparte na produktach (item-based),
- **Moduł Sentiment Analysis** - agregacja sentymentu z 5 źródeł tekstowych (opinie, opis, nazwa, specyfikacje, kategorie),
- **Moduł Apriori** - generowanie reguł asocjacyjnych typu "Często kupowane razem",
- **Logika transakcyjna** - składanie zamówień, zarządzanie statusami, walidacja danych.

Warstwa danych - relacyjna baza danych PostgreSQL przechowująca:

- Dane produktów (nazwa, opis, cena, kategorie, specyfikacje, zdjęcia),
- Dane użytkowników (konta, profile, uprawnienia),
- Dane transakcyjne (zamówienia, produkty w zamówieniach, statusy),
- Dane opinii (recenzje tekstowe, oceny gwiazdkowe, sentyment),
- Wyniki algorytmów (macierze podobieństwa, reguły asocjacyjne, zagregowany sentiment).

Integracja warstw odbywa się poprzez RESTful API z automatyczną synchronizacją - zmiana danych w jednej warstwie propaguje aktualizacje do pozostałych.

3.5 Kluczowe scenariusze użycia

Scenariusz 1: Użytkownik niezalogowany przegląda produkty

Gość wchodzi na stronę główną aplikacji. Nawiguje do katalogu produktów, filtruje po kategorii „Laptopy"', sortuje według „Najlepszy sentyment"'. Przegląda kilka produktów, dodaje wybrany laptop do koszyka. W koszyku widzi sekcję "Często kupowane razem" z torbą i myszą (rekomendacje Apriori). Decyduje się również

dodać torbę. Przy próbie finalizacji zamówienia system przekierowuje do logowania lub rejestracji.

Scenariusz 2: Klient składa zamówienie i śledzi jego status

Użytkownik loguje się do systemu. Przechodzi do koszyka, klika „Przejdź do kasy"', wypełnia dane dostawy i płatności. Po złożeniu zamówienia otrzymuje potwierdzenie ze statusem „oczekujące"'. W panelu klienta (zakładka „My Orders"') widzi listę wszystkich zamówień z możliwością śledzenia statusu w czasie rzeczywistym (oczekujące → w realizacji → zakończone).

Scenariusz 3: Administrator zarządza produktami

Administrator loguje się do panelu administracyjnego. Wchodzi do sekcji „Products"', klika „Add New Product"'. Wypełnia formularz (nazwa, opis, cena, specyfikacje), przypisuje kategorie. Po zapisaniu produktu, system automatycznie przelicza jego wynik sentymentu (sentiment score) na podstawie opisu i nazwy (brak jeszcze opinii). Administrator może również edytować istniejące produkty lub usuwać te o niskim sentymencie.

Scenariusz 4: Klient wystawia opinię i system przelicza sentyment

Zalogowany użytkownik przechodzi do panelu klienta, otwiera zakładkę „My Orders"'. Znajduje zakończone zamówienie, klika „Add Review" przy zakupionym produkcie. Wypełnia formularz opinii (ocena 1-5 gwiazdek oraz treść tekstowa). Po zapisaniu system automatycznie wywołuje algorytm analizy sentymentu, który tokenizuje tekst, wyszukuje słowa kluczowe w słowniku pozytywnym/negatywnym i oblicza zagregowany wynik sentymentu. Zaktualizowany wynik jest natychmiast widoczny dla wszystkich użytkowników na stronie produktu.

Scenariusz 5: Administrator przegląda statystyki i prognozy sprzedaży

Administrator wchodzi do sekcji „Statistics" w panelu administracyjnym. Dashboard wyświetla kluczowe metryki: całkowita liczba zamówień, przychód, najpopularniejsze kategorie.

Scenariusz 6: Administrator debuguje algorytmy rekomendacji

Administrator otwiera sekcję „Debug ML" w panelu administracyjnym. Przegląda panel debugowania Collaborative Filtering - widzi macierz podobieństw produktów, formułę Adjusted Cosine Similarity oraz statystyki (liczba par produktów, średnie podobieństwo). Przechodzi do panelu Apriori - analizuje wygenerowane reguły asocjacyjne posortowane według lift, sprawdza metryki wydajności bitmap pruning. W panelu Sentiment Analysis weryfikuje rozkład opinii i wieloźródłową agregację sentymentu.

Scenariusz 7: Klient wyszukuje produkty z filtrem sentymentu

Użytkownik otwiera wyszukiwarkę z paska nawigacyjnego, wpisuje zapytanie „laptop gaming"'. System wykonuje wyszukiwanie full-text w polach nazwa i opis

produktu, filtruje wyniki z `sentiment_score` ≥ 0.5, sortuje malejąco według sentymentu.

Scenariusz 8: Gość przegląda koszyk z rekomendacjami cross-selling

Niezalogowany użytkownik dodaje laptop do koszyka. Przechodzi do widoku koszyka - system automatycznie wyświetla sekcję „Frequently Bought Together" z trzema produktami wygenerowanymi przez algorytm Apriori (torba na laptopa, mysz bezprzewodowa, hub). Każda rekomendacja zawiera metryki lift i confidence. Użytkownik decyduje się dodać torbę i mysz jednym kliknięciem. Przy próbie finalizacji zamówienia system przekierowuje do logowania.

3.6 Podsumowanie opisu projektu

Planowana aplikacja e-commerce z zaawansowanym systemem rekomendacji została zaprojektowana z hierarchią trzech typów użytkowników (gość, klient, administrator), gdzie każdy poziom dziedziczy funkcjonalności poprzedniego i dodaje nowe, specyficzne dla swojej roli. Diagram przypadków użycia jasno definiuje role aktorów oraz ich interakcje z systemem, podczas gdy szczegółowe scenariusze użycia ilustrują praktyczne zastosowania funkcjonalności w rzeczywistych przepływach zakupowych.

Kluczowe cechy projektu:

- **Hierarchia uprawnień** - przejrzysta struktura dziedziczenia funkcjonalności między rolami,
- **Kompleksowość** - pokrycie wszystkich etapów od przeglądania po zarządzanie systemem,
- **Integracja rekomendacji** - metody rekomendacyjne zintegrowane w przepływ użytkownika,
- **Narzędzia diagnostyczne** - panele debugowania dla administratora umożliwiające monitoring i optymalizację algorytmów.

Szczegółowa realizacja techniczna projektu, w tym wykorzystane technologie oraz implementacja algorytmów, zostanie przedstawiona w kolejnych rozdziałach pracy.

Rozdział 4

Przedstawienie wykorzystanego stosu technologicznego oraz praktycznej realizacji projektu

Szczegółowy opis technologii wykorzystanych w implementacji systemu e-commerce oraz praktyczne aspekty realizacji projektu. Przedstawiono architekturę techniczną aplikacji, komponenty frontendowe, strukturę bazy danych oraz mechanizmy wdrożenia.

4.1 Architektura systemu

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.

4.2 Stos technologiczny backendu

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. Kluczowe komponenty:

- **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,
- **ViewSets (zestawy widoków)** - widoki implementujące operacje CRUD,
- **Uwierzytelnianie (Authentication)** - wsparcie dla JWT, uwierzytelnianie sesyjne,
- **Pagination (paginacja)** - automatyczne stronicowanie wyników.

Biblioteki Machine Learning
Do operacji numerycznych i obliczania podobieństw wykorzystano:

- **scikit-learn** - funkcja cosine_similarity() dla Collaborative Filtering (Collaborative Filtering),
- **NumPy** - operacje macierzowe, wektoryzacja obliczeń, przycinanie bitmapowe (bitmap pruning) w Apriori.

Struktura backendu
Każdy komponent systemu posiada dedykowane pliki:

- **models.py** – definicje tabel (Product, Order, Opinion, ProductSimilarity),
- **serializers.py** – konwersja obiektów Django $\leftrightarrow$ JSON,
- **views.py** – obsługa CRUD dla produktów, zamówień,
- **recommendation_views.py** – endpoint /api/collaborative-filtering/,
- **sentiment_views.py** – endpoint /api/sentiment-search/,
- **association_views.py** – endpoint /api/association-recommendations/,
- **signals.py** – automatyczna aktualizacja rekomendacji.

4.3 Stos technologiczny frontendu

React 18

React stanowi fundament aplikacji jednostronicowej (Single Page Application - SPA):

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

Architektura komponentów

- **App.js** – trasowanie, globalne dostawcy kontekstu (Context providers),
- **Navbar.jsx** – nawigacja z wyszukiwarką i ikoną koszyka,
- **SearchModal.jsx** – wyszukiwarka z sortowaniem sentymentu,
- **ProductPage.jsx** – strona produktu z rekomendacjami Collaborative Filtering i Apriori,
- **CartContent.jsx** – koszyk z cross-selling (Apriori),
- **ClientPanel.jsx** – panel główny klienta z personalizowanymi rekomendacjami,
- **AdminPanel.jsx** – panel zarządzania produktami, zamówieniami, statystykami.

4.4 Baza danych PostgreSQL

Wybór PostgreSQL 14

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

- **Zaawansowane indeksy** - wsparcie dla B-tree (domyślne), GIN (wyszukiwanie pełnotekstowe), BRIN (optymalizacja dla dużych tabel),
- **Typ danych JSONB** - natywne przechowywanie i indeksowanie struktur JSON (wykorzystane w tabeli `method_user_purchase_pattern` dla 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,
- **Full-text search** - wbudowane mechanizmy wyszukiwania tekstowego dla produktów.

Struktura bazy danych

Baza składa się z **25 tabel** podzielonych na 4 modułów funkcjonalnych:

1. Moduł produktów i użytkowników (12 tabel):

- `db_product` - dane produktów (ID, nazwa, cena, opis),
- `db_category` - kategorie produktów z hierarchią,
- `db_product_category` - relacjaMany-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):

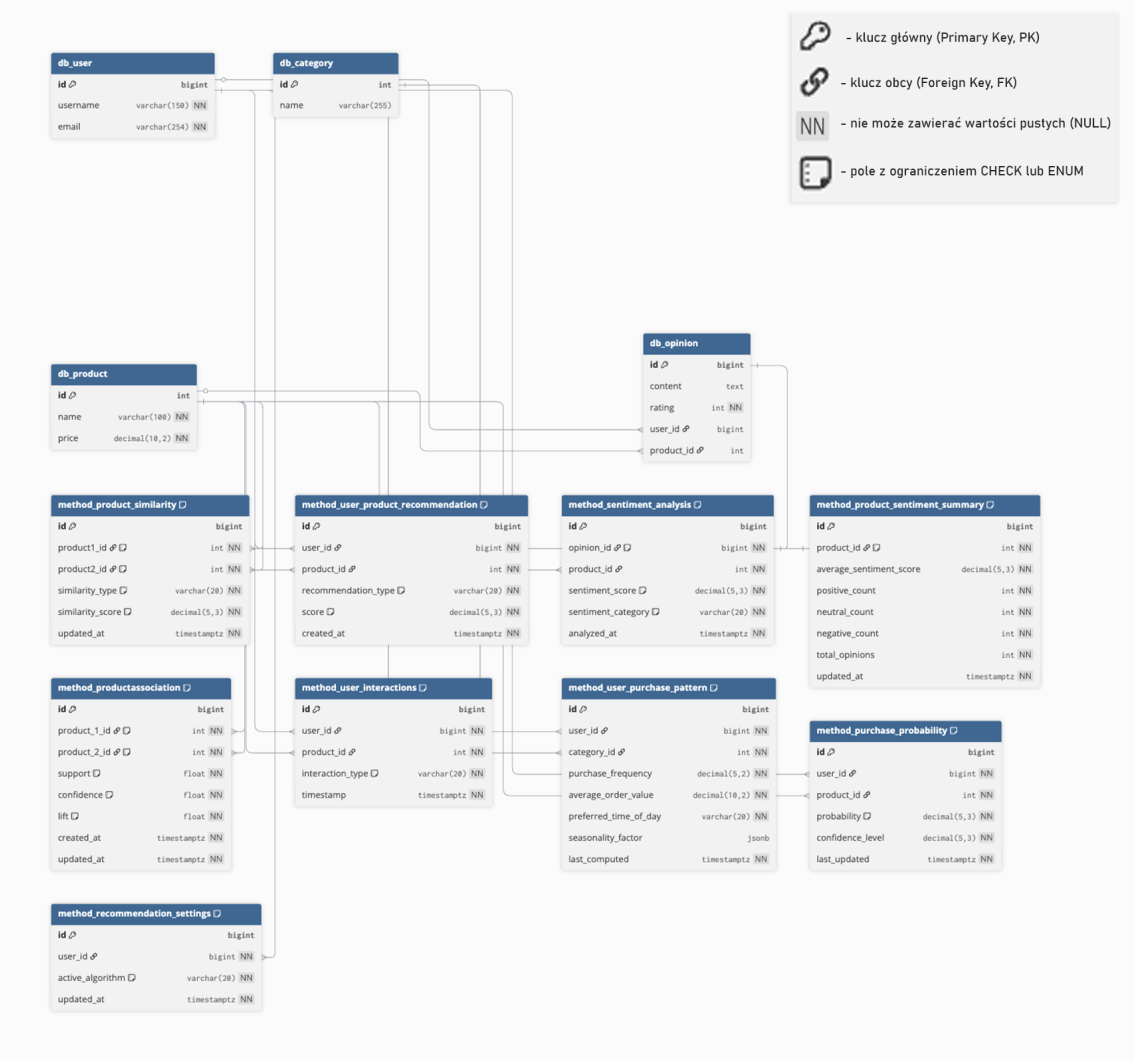
- `method_product_similarity` - macierz podobieństw produktów (Collaborative Filtering),
- `method_user_product_recommendation` - spersonalizowane rekomendacje użytkowników,
- `method_productassociation` - reguły asocjacyjne Apriori,
- `method_user_interactions` - historia interakcji użytkowników,
- `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.

Wszystkie migracje Django ORM zostały wygenerowane automatycznie na podstawie modeli Python i zarządzane przez system wersjonowania `django.db.migrations`.

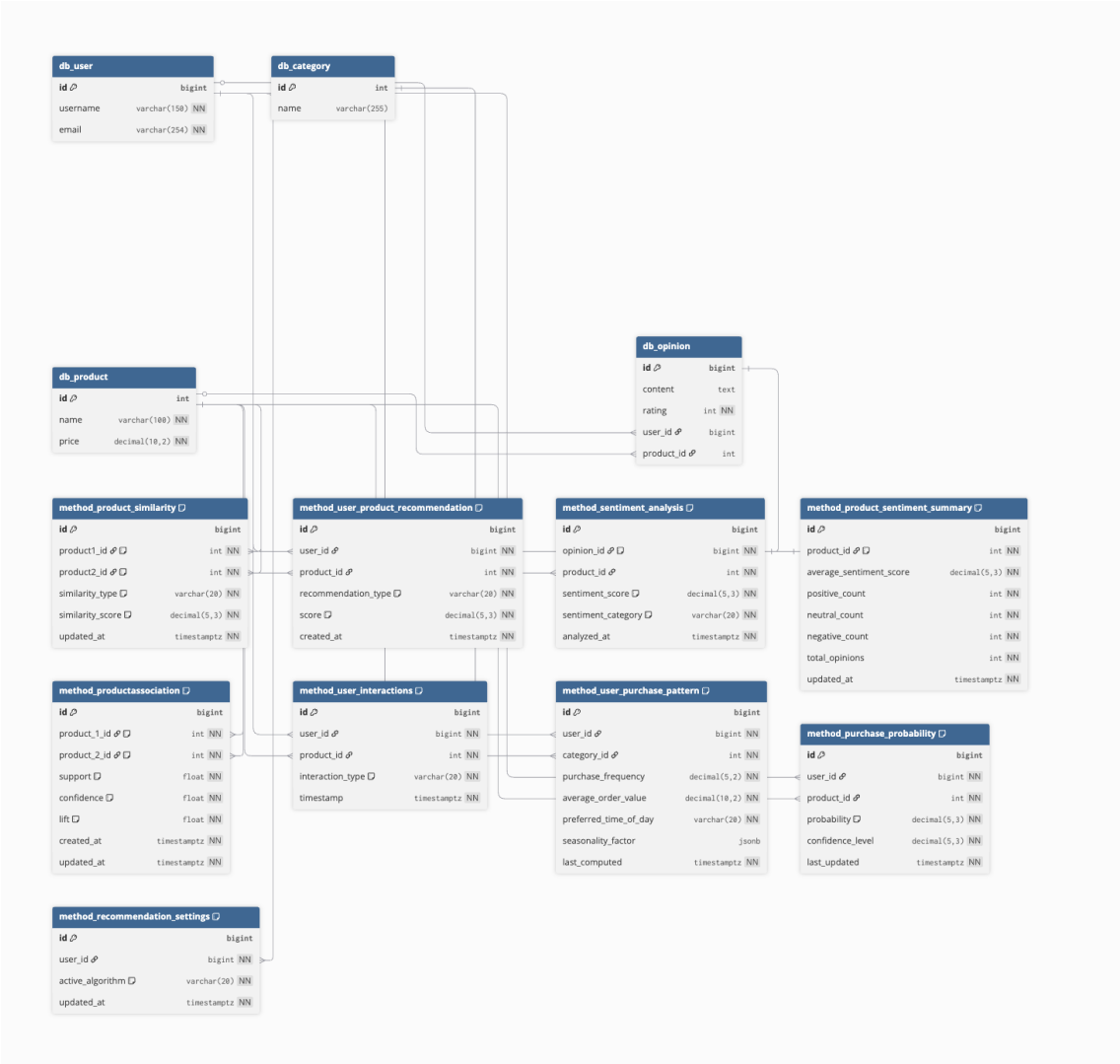
Diagramy ERD



Rysunek 2: Diagram ERD głównych tabel aplikacji.

Diagram 2 przedstawia rdzeń aplikacji e-commerce. Kluczowe relacje:

- **db_user** → **db_order** (1:N) - jeden użytkownik składa wiele zamówień,
- **db_order** → **db_order_product** (1:N) - zamówienie zawiera wiele produktów,
- **db_product** ↔ **db_category** (N:M) - produkt należy do wielu kategorii,
- **db_product** → **db_opinion** (1:N) - produkt ma wiele opinii.



Rysunek 3: Diagram ERD tabel metod rekomendacyjnych.

Diagram 3 pokazuje tabele algorytmów ML:

- **method_product_similarity** - macierz podobieństw Collaborative Filtering,
- **method_productassociation** - reguły asocjacyjne Apriori,
- **method_sentiment_analysis** - wyniki analizy sentymentu opinii,
- **method_product_sentiment_summary** - zagregowany sentyment produktu,
- **method_user_product_recommendation** - cache rekomendacji użytkownika.

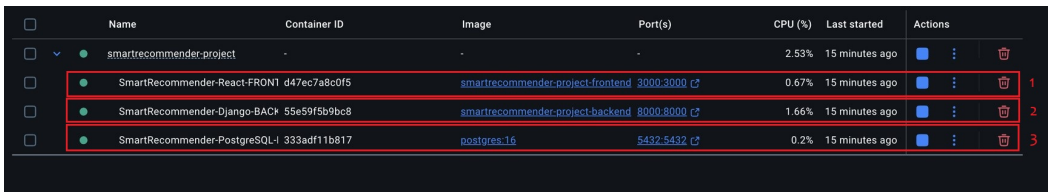
Podsumowanie struktury bazy danych

Baza danych składa się z 25 tabel zorganizowanych w moduły funkcjonalne. Wypełnianie danymi początkowymi (seeding) wypełnia bazę danymi testowymi:

- 500 produktów,
- 20 użytkowników (5 adminów + 15 klientów),
- 200 zamówień,
- 600 rekordów OrderProduct (średnio 3 produkty per zamówienie),
- 1750 opinii (średnio 3.5 opinii per produkt).

4.5 Deployment i konteneryzacja Docker

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



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, psycopg2, 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.

Zalety konteneryzacji Docker

- **Izolacja** - każdy serwis w osobnym kontenerze, zero konfliktów zależności,
- **Przenośność** - obraz zbudowany raz działa na dowolnym serwerze z silnikiem Docker,
- **Łatwa konfiguracja** - komenda `docker-compose up` uruchamia całą aplikację,
- **Skalowalność** - możliwość uruchomienia wielu instancji backendu dla równoważenia obciążenia.

4.6 Podsumowanie stosu technologicznego

Wybrane technologie (Django + React + PostgreSQL + Docker) tworzą nowoczesny, skalowalny i łatwy w utrzymaniu stos technologiczny. Kluczowe zalety:

- **Separacja kontynerów** - wyraźny podział frontend/backend/baza,
- **Wygoda programisty** - narzędzia (Django ORM, haki React, Docker Compose) przyspieszają rozwój aplikacji,
- **Wsparcie społeczności** - aktywna społeczność, obszerna dokumentacja,
- **Wydajność** - optymalizacje (pamięć podręczna, indeksy bazy danych, operacje wektorowe NumPy) zapewniają szybki czas odpowiedzi.

Rozdział 5

Implementacja algorytmów rekomendacji

Szczegółowa implementacja trzech algorytmów rekomendacyjnych wraz z pseudokodami oraz diagramami sekwencji przedstawia praktyczne aspekty realizacji metod collaborative filtering, analizy sentymentu oraz reguł asocjacyjnych. Każda metoda została zaimplementowana od podstaw bez użycia gotowych bibliotek rekomendacyjnych.

5.1 Collaborative Filtering - Item-Based z Adjusted Cosine

Algorytm Collaborative Filtering oblicza podobieństwa między produktami na podstawie historii zakupów użytkowników. Proces składa się z trzech głównych etapów: budowy macierzy użytkownik-produkt, normalizacji metodą mean-centering oraz obliczenia podobieństw cosinusowych. Normalizacja polega na odjęciu od każdej wartości zakupowej średniej zakupów danego użytkownika, co eliminuje zniekształcenia wynikające z różnych skal (hurtownik kupujący po 100 sztuk vs klient kupujący po 1 sztuce otrzymują porównywalne wagi).

Pseudokod 1a: Budowa i normalizacja macierzy Collaborative Filtering

Listing 1: Budowa macierzy uzytkownik-produkt z normalizacja

```
FUNKCJA buduj_macierz_cf():
    pozycje = pobierz_wszystkie_pozycje_zamowien()
    macierz = pusty_slownik()

    DLA KAZDEJ poz W pozycje:
        uz_id = poz.order.user_id
        prod_id = poz.product_id
        macierz[uz_id][prod_id] = poz.quantity
    KONIEC DLA

    M = konwertuj_na_tablice(macierz)
    M_norm = macierz_zerowa(wymiary(M))

    DLA i = 0 DO liczba_wierszy(M) - 1:
        zakupione = M[i] gdzie M[i] > 0
        JEZELI |zakupione| > 0 WTEDY
            sr = srednia(zakupione)
            DLA j = 0 DO liczba_kolumn(M) - 1:
```

```
                    JEZELI M[i][j] > 0 WTEDY
                        M_norm[i][j] = M[i][j] - sr
                    KONIEC JEZELI
                KONIEC DLA
            KONIEC JEZELI
        KONIEC DLA

        ZWROC M_norm
KONIEC FUNKCJA
```

Pseudokod 1b: Obliczanie podobieństw produktów

Listing 2: Obliczanie podobienstwa cosinusowego produktow

```
FUNKCJA oblicz_podobienstwa_cf(M_norm, prog = 0.5):
    M_T = transponuj(M_norm)
    sim = podobienstwo_cosinus(M_T)

    usun_podobienstwa(typ='collaborative')
    lista = pusta_lista()
    n = liczba_produktow(M_T)

    DLA i = 0 DO n - 1:
        DLA j = 0 DO n - 1:
            JEZELI i != j ORAZ sim[i][j] > prog WTEDY
                dopisz(lista, {prod1: i, prod2: j, wynik:
                   sim[i][j]})

                JEZELI |lista| >= 1000 WTEDY
                    zapisz_zbiorczo(lista)
                    lista = pusta_lista()
                KONIEC JEZELI
            KONIEC JEZELI
        KONIEC DLA
    KONIEC DLA

    JEZELI |lista| > 0 WTEDY
        zapisz_zbiorczo(lista)
    KONIEC JEZELI
KONIEC FUNKCJA
```

Algorytm generowania rekomendacji dla użytkownika wykorzystuje wcześniej obliczone podobieństwa produktów. Proces polega na zebraniu historii zakupów użytkownika (zamówienia + koszyk), a następnie dla każdego produktu z tej historii wyszukiwane są najbardziej podobne produkty. Wyniki są agregowane - jeśli dany produkt jest podobny do wielu produktów zakupionych przez użytkownika, otrzymuje sumę wszystkich podobieństw jako końcowy wynik rekomendacji.

Pseudokod 2: Generowanie rekomendacji dla użytkownika

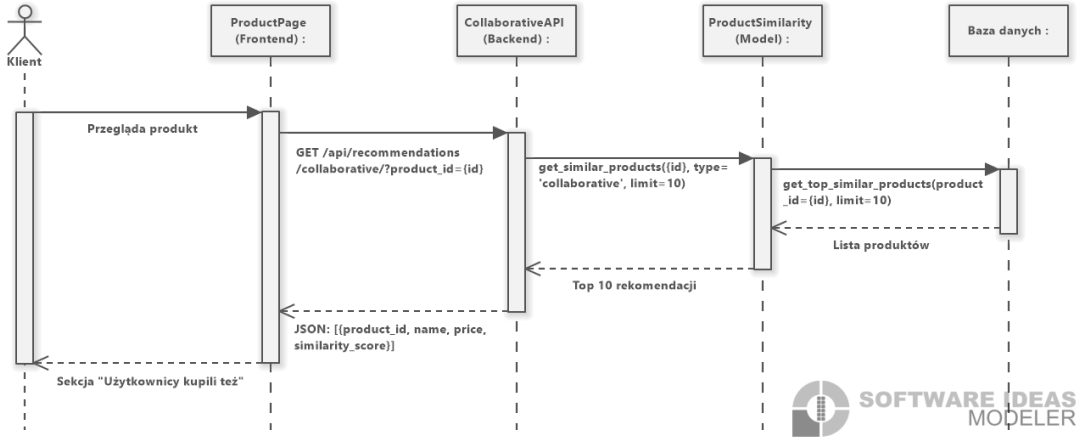
Listing 3: Generowanie rekomendacji CF dla uzytkownika

```
FUNKCJA generuj_rekomendacje(uzytkownik, typ_algorytmu):
    historia = pusta_lista()

    DLA KAZDEGO zam W pobierz_zamowienia(uzytkownik):
        DLA KAZDEGO p W zam.produkty:
            dopisz(historia, p.id)
        KONIEC DLA
    KONIEC DLA

    DLA KAZDEJ poz W pobierz_koszyk(uzytkownik):
        dopisz(historia, poz.produkt.id)
    KONIEC DLA

    wyniki = pusty_slownik()
    DLA KAZDEGO prod_id W unikalne(historia):
        podobne = pobierz_podobne(prod_id, typ_algorytmu,
            limit=5)
        DLA KAZDEGO s W podobne:
            JEZELI s.produkt2_id W wyniki WTEDY
                wyniki[s.produkt2_id] += s.podobienstwo
            INACZEJ
                wyniki[s.produkt2_id] = s.podobienstwo
            KONIEC JEZELI
        KONIEC DLA
    KONIEC DLA

    lista = sortuj(wyniki, malejaco)
    DLA KAZDEGO (p_id, wart) W lista:
        zapisz_rekomendacje(uzytkownik, p_id, typ_algorytmu
            , wart)
    KONIEC DLA
KONIEC FUNKCJA
```

Kluczowe optymalizacje Collaborative Filtering:

- Pamięć podręczna (cache) przechowuje macierz podobieństw przez 24 godziny,
- Operacje zbiorcze (bulk insert) zapisują po 1000 rekordów jednocześnie,
- Próg 0.5 redukuje rozmiar bazy, poprzez odrzucenie niższych wartości,
- Indeksowanie na kolumnach (produkt_1, typ_podobienstwa).

Diagram sekwencji: Collaborative Filtering



Rysunek 5: Diagram sekwencji: Collaborative Filtering.

Przepływ procesu generowania rekomendacji Collaborative Filtering:

1. Żądanie użytkownika trafia do API (/api/recommendations/)
2. API sprawdza pamięć podręczną - w przypadku trafienia zwracany jest wynik natychmiast
3. Przy braku w pamięci podręcznej następuje zapytanie do bazy danych o historię zakupów
4. Budowana jest macierz użytkownik-produkt z OrderProduct
5. Macierz jest normalizowana (centrowanie wartości) i obliczane są podobieństwa cosinusowe
6. Wyniki zapisywane są do pamięci podręcznej (24h) oraz tabeli ProductSimilarity
7. API zwraca top N rekomendacji

5.2 Analiza Sentymentu - wieloźródłowa agregacja

Analiza sentymentu pojedynczego tekstu metodą słownikową (Liu 2012) polega na normalizacji tekstu (konwersja na małe litery, usunięcie znaków specjalnych), tokenizacji oraz zliczaniu słów pozytywnych i negatywnych. Słowniki zawierają 200 słów każdego typu (np. pozytywne: , ‚excellent', , ‚recommend', , ‚quality'; negatywne: , ‚bad', , ‚poor', , ‚disappointing'). Formuła wyniku: (liczba_pozytywnych - liczba_negatywnych) / liczba_wszystkich_słów. Wynik ograniczany do [-1, 1] i kategoryzowany: pozytywny (>0.1), negatywny (<-0.1), neutralny (inne).

Wieloźródłowa agregacja sentymentu produktu łączy wyniki z 5 źródeł z wagami: opinie 40%, opis 25%, nazwa 15%, specyfikacje 12%, kategorie 8%.

Pseudokod 3: Analiza sentymentu pojedynczego tekstu

Listing 4: Analiza sentymentu metoda slownikowa

```
FUNKCJA analizuj_sentiment(tekst, slownik_poz, slownik_neg)
    :
     slowa = tokenizuj(na_male_litery(tekst))

     JEZELI |slowa| = 0 WTEDY
         ZWROC (0.0, ,,neutralny'')
     KONIEC JEZELI

     poz = 0
     neg = 0

     DLA KAZDEGO s W slowa:
         JEZELI s W slownik_poz WTEDY poz = poz + 1
         JEZELI s W slownik_neg WTEDY neg = neg + 1
     KONIEC DLA

     wynik = (poz - neg) / |slowa|
     wynik = ogranicz(wynik, -1.0, 1.0)

     JEZELI wynik > 0.1 WTEDY kat = ,,pozytywny''
     INACZEJ JEZELI wynik < -0.1 WTEDY kat = ,,negatywny''
     INACZEJ kat = ,,neutralny''

     ZWROC (wynik, kat)
KONIEC FUNKCJA
```

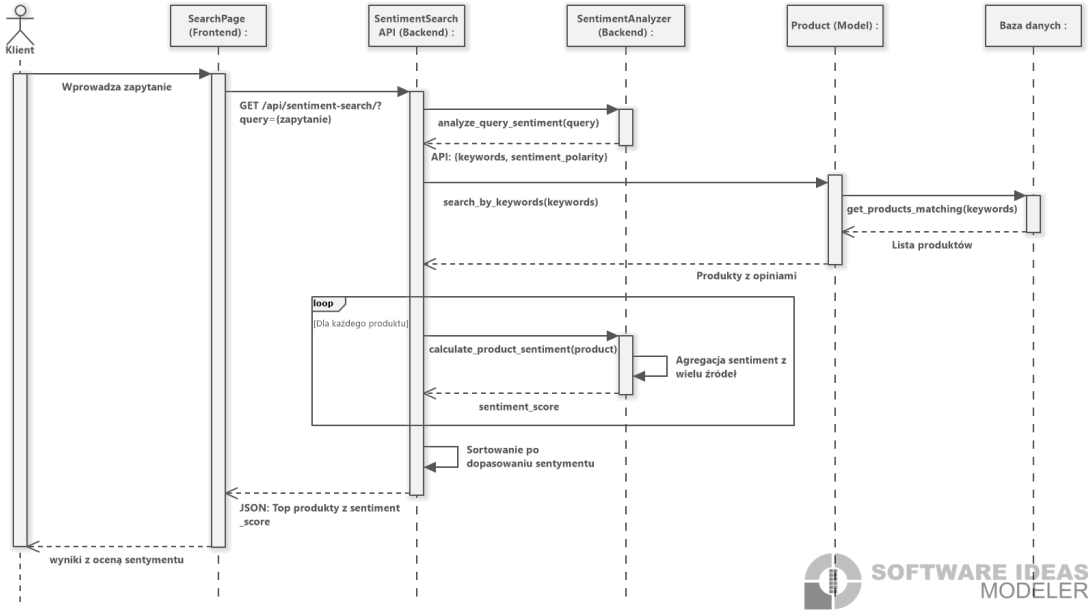
Algorytm wieloźródłowej agregacji sentymentu analizuje 5 niezależnych źródeł tekstowych produktu zamiast polegać wyłącznie na opiniach klientów. Rozwiązuje to problem zimnego startu - produkty bez opinii nadal otrzymują wynik bazujący na opisie i nazwie. Wagi źródeł: opinie 40%, opis 25%, nazwa 15%, specyfikacje 12%, kategorie 8%. System zlicza rozkład opinii (pozytywne/negatywne/neutralne) dla lepszej oceny konsensusu.

Pseudokod 4: Agregacja sentymentu z wielu źródeł

Listing 5: Agregacja sentymentu produktu z 5 zrodel

```
FUNKCJA agreguj_sentiment_produktu(produkt):
    opinie = pobierz_opinie(produkt)[:20]
    wyniki_opinii = pusta_lista()
    DLA KAZDEJ op W opinie:
        (w, _) = analizuj_sentiment(op.tresc)
        dopisz(wyniki_opinii, w)
    KONIEC DLA
    S_op = srednia(wyniki_opinii) JEZELI |wyniki_opinii| >
       0 INACZEJ 0
    (S_opis, _) = analizuj_sentiment(produkt.opis)
    (S_nazwa, _) = analizuj_sentiment(produkt.nazwa)
    teksty_spec = pusta_lista()
    DLA KAZDEJ sp W produkt.specyfikacje[:10]:
        dopisz(teksty_spec, sp.nazwa + ‘‘ ’’ + sp.wartosc)
    KONIEC DLA
    (S_spec, _) = analizuj_sentiment(polacz(teksty_spec))
    kat_txt = polacz(pobierz_nazwy_kategorii(produkt))
    (S_kat, _) = analizuj_sentiment(kat_txt)
    S_final = 0.40*S_op + 0.25*S_opis + 0.15*S_nazwa
            + 0.12*S_spec + 0.08*S_kat

    poz = policz(wyniki_opinii WHERE w > 0.1)
    neg = policz(wyniki_opinii WHERE w < -0.1)
    neu = policz(wyniki_opinii WHERE -0.1 <= w <= 0.1)

    ZWROC {
        wynik: zaokragl(S_final, 3),
        pozytywne: poz,
        negatywne: neg,
        neutralne: neu
    }
KONIEC FUNKCJA
```

Diagram sekwencji: Analiza Sentymentu



Rysunek 6: Diagram sekwencji: Analiza sentymentu.

Przepływ procesu wieloźródłowej analizy sentymentu:

1. Użytkownik wyszukuje produkty z sortowaniem według sentymentu
2. API wywołuje funkcję agregacji sentymentu dla każdego produktu
3. System analizuje 5 źródeł tekstowych równolegle (opinie 40%, opis 25%, nazwa 15%, specyfikacje 12%, kategorie 8%)
4. Dla każdego źródła wykonywana jest tokenizacja i zliczanie słów pozytywnych/negatywnych
5. Wyniki są agregowane według formuły ważonej: $S_{koncowy} = \sum_{i=1}^{5} w_i \times S_i$
6. Końcowy wynik zapisywany jest w ProductSentimentSummary
7. API zwraca produkty posortowane według zagregowanego sentymentu

Kluczową zaletą tej architektury jest rozwiązanie problemu zimnego startu - produkty bez opinii nadal otrzymują wynik sentymentu na podstawie pozostałych 4 źródeł.

5.3 Algorytm Apriori - reguły asocjacyjne

Algorytm Apriori generuje reguły asocjacyjne typu „Często kupowane razem” wykorzystując optymalizację przycinania bitmapowego (Zaki 2000). Proces składa się z dwóch głównych etapów: znajdowania częstych par produktów oraz generowania reguł z obliczeniem metryk (wsparcie, pewność, lift).

Optymalizacja bitmapowa polega na reprezentacji transakcji jako liczb całkowitych z ustawionymi bitami odpowiadającymi produktom. Sprawdzenie czy transakcja zawiera parę produktów wymaga jednej operacji bitowej AND zamiast iteracji po liście. Wczesne przycinanie eliminuje rzadkie produkty przed obliczaniem par - dzięki właściwości antymonotoniczności Apriori (jeśli produkt jest rzadki, wszystkie jego pary też są rzadkie) zmniejsza liczbę kandydatów.

Pseudokod 5a: Znajdowanie częstych par z bitmapami

Listing 6: Apriori - znajdowanie czestych par produktow

```
FUNKCJA znajdz_czeste_pary(transakcje, min_wsp):
    n = |transakcje|
    min_licz = podloga(min_wsp * n)

    liczniki = pusty_slownik()
    DLA KAZDEJ t W transakcje:
        DLA KAZDEGO p W t:
            liczniki[p] = liczniki[p] + 1
        KONIEC DLA
    KONIEC DLA

    czeste = [p DLA (p, l) W liczniki JEZELI l >= min_licz]

    mapa = pusty_slownik()
    DLA i = 0 DO |czeste| - 1:
        mapa[czeste[i]] = i
    KONIEC DLA

    bitmapy = pusta_lista()
    DLA KAZDEJ t W transakcje:
        bm = 0
        DLA KAZDEGO p W t:
            JEZELI p W mapa WTEDY
                bm = bm LUB (1 << mapa[p])
            KONIEC JEZELI
        KONIEC DLA
        JEZELI bm != 0 WTEDY
```

```
28              dopisz(bitmapy, bm)
29          KONIEC JEZELI
30      KONIEC DLA
31
32      pary = pusty_slownik()
33      DLA i = 0 DO |czeste| - 1:
34          bit_i = 1 << i
35          DLA j = i + 1 DO |czeste| - 1:
36              bit_j = 1 << j
37              para_bm = bit_i LUB bit_j
38
39              licz = 0
40              DLA KAZDEJ bm W bitmapy:
41                  JEZELI (bm AND para_bm) == para_bm WTEDY
42                      licz = licz + 1
43                  KONIEC JEZELI
44              KONIEC DLA
45
46              JEZELI licz >= min_licz WTEDY
47                  pary[{czeste[i], czeste[j]}] = licz / n
48              KONIEC JEZELI
49          KONIEC DLA
50      KONIEC DLA
51
52      wsparcia = {p: liczniki[p]/n DLA p W czeste}
53      ZWROC (pary, wsparcia)
54  KONIEC FUNKCJA
```

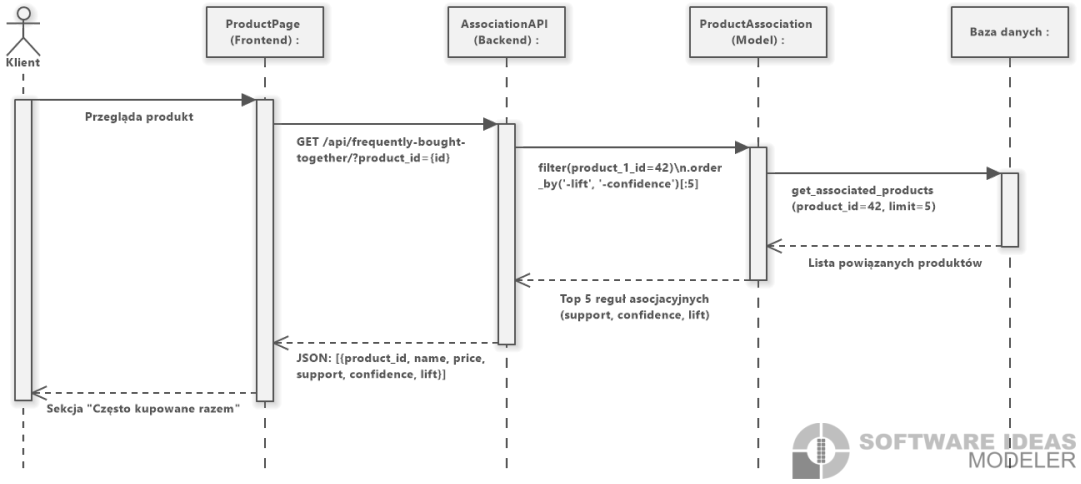
Pseudokod 5b: Generowanie reguł asocjacyjnych

Listing 7: Apriori - generowanie regul z obliczeniem lift i confidence

```
1   FUNKCJA generuj_reguly_z_par(pary, wsparcia, min_pewn):
2       reguly = pusta_lista()
3
4       DLA KAZDEJ ({p1, p2}, wsp_para) W pary:
5           wsp1 = wsparcia[p1]
6           wsp2 = wsparcia[p2]
7
8           pewn_1_2 = wsp_para / wsp1
9           pewn_2_1 = wsp_para / wsp2
10
11          lift = wsp_para / (wsp1 * wsp2)
```

```
12
13          JEZELI pewn_1_2 >= min_pewn WTEDY
14              dopisz(reguly, {od: p1, do: p2,
15                              wsparcie: wsp_para,
16                              pewnosc: pewn_1_2,
17                              lift: lift})
18          KONIEC JEZELI
19
20          JEZELI pewn_2_1 >= min_pewn WTEDY
21              dopisz(reguly, {od: p2, do: p1,
22                              wsparcie: wsp_para,
23                              pewnosc: pewn_2_1,
24                              lift: lift})
25          KONIEC JEZELI
26      KONIEC DLA
27
28      sortuj(reguly, wedlug=(lift, pewnosc), malejaco)
29      ZWROC reguly
30  KONIEC FUNKCJA
```

Diagram sekwencji: Algorytm Apriori



Rysunek 7: Diagram sekwencji: Algorytm Apriori.

Przepływ procesu generowania reguł asocjacyjnych algorytmem Apriori:

1. Administrator wywołuje aktualizację reguł przez panel administracyjny

2. System ekstrahuje transakcje z bazy OrderProduct (tylko zamówienia z 2+ produktami)

3. Algorytm wykonuje wczesne przycinanie - eliminuje rzadkie produkty (wsparcie < 0.001)

4. Transakcje konwertowane są do reprezentacji bitmapowej dla optymalizacji

5. System generuje częste 2-itemsety używając operacji bitowych AND (znaczące przyspieszenie)

6. Dla każdego częstego itemsetu obliczane są metryki: Wsparcie, Pewność, Lift

7. Reguły filtrowane są według progów i zapisywane w bazie z wstawianiem zbiorczym

Kluczowe optymalizacje Apriori:

- Przycinanie bitmapowe: znaczące przyspieszenie operacji,
- Wczesne przycinanie: eliminacja większości kandydatów,
- Operacje bitowe AND: optymalna złożoność obliczeniowa,
- Operacje zbiorcze (bulk operations): rozmiar_partii=500.

5.4 Podsumowanie implementacji

Przedstawione pseudokody odzwierciedlają algorytmy zaimplementowane w aplikacji. Kluczowe aspekty techniczne:

Collaborative Filtering:

- Adjusted Cosine Similarity eliminuje zniekształcenia skal zakupowych,
- NumPy dla wektoryzacji operacji macierzowych,
- Pamięć podręczna Django 24h z automatycznym unieważnieniem,
- Operacje zbiorcze (rozmiar_partii=1000),
- Próg podobieństwa 0.5 redukuje liczbę zapisywanych rekordów.

Analiza Sentymentu:

- Słowniki Opinion Lexicon + AFINN-165 (400 słów),
- Wieloźródłowa agregacja (5 źródeł) z optymalnymi wagami,
- Formuła Liu (2012): $S = \frac{N_{poz} - N_{neg}}{N_{calkowite}}$,
- Rozwiązanie problemu zimnego startu.

Algorytm Apriori:

- Przycinanie bitmapowe dla optymalizacji obliczeń,
- Wczesne przycinanie eliminuje większość kandydatów niespełniających progu wsparcia,
- Operacje bitowe AND zamiast iteracji po listach,
- Operacje zbiorcze (rozmiar_partii=500),
- Metryki: Wsparcie, Pewność, Lift.

Wydajność systemu:

- Collaborative Filtering: pierwsza generacja wymaga przetworzenia macierzy, kolejne zapytania korzystają z pamięci podręcznej,
- Sentiment: przetwarzanie wsadowe opinii z wykorzystaniem słowników w pamięci,
- Apriori: przycinanie bitmapowe redukuje złożoność obliczeniową,
- Mechanizm cache'owania znacząco przyspiesza kolejne zapytania API.

Rozdział 6

Funkcjonowanie systemu rekomendacji w praktyce

Praktyczne zastosowanie trzech zaimplementowanych metod rekomendacyjnych w kontekście rzeczywistego systemu e-commerce przedstawiono na podstawie zrzutów ekranu z działającej aplikacji. Zaprezentowano scenariusze użycia, przepływy danych oraz mechanizmy zarządzania algorytmami przez panel administracyjny.

6.1 Architektura i przepływ danych

System rekomendacji został zintegrowany z aplikacją e-commerce w architekturze trójwarstwowej:

Warstwa prezentacji (React) - interfejs użytkownika wyświetlający rekomendacje w różnych kontekstach:

- Strona główna - sekcja „Recommended for You" z możliwością przełączania algorytmów,
- Panel klienta - dashboard z personalizowanymi rekomendacjami,
- Wyszukiwarka - sortowanie według sentymentu,
- Koszyk zakupowy - sekcja „Frequently Bought Together",
- Panel administracyjny - zarządzanie metodami rekomendacji i debugowanie.

Warstwa logiki biznesowej (Django) - endpointy API obsługujące zapytania o rekomendacje:

- `/api/recommendations/collaborative/` - zwraca produkty Collaborative Filtering,
- `/api/recommendations/sentiment/` - zwraca produkty według sentymentu,
- `/api/recommendations/association/` - zwraca reguły asocjacyjne,
- `/api/admin/recommendation-settings/` - zarządzanie aktywną metodą.

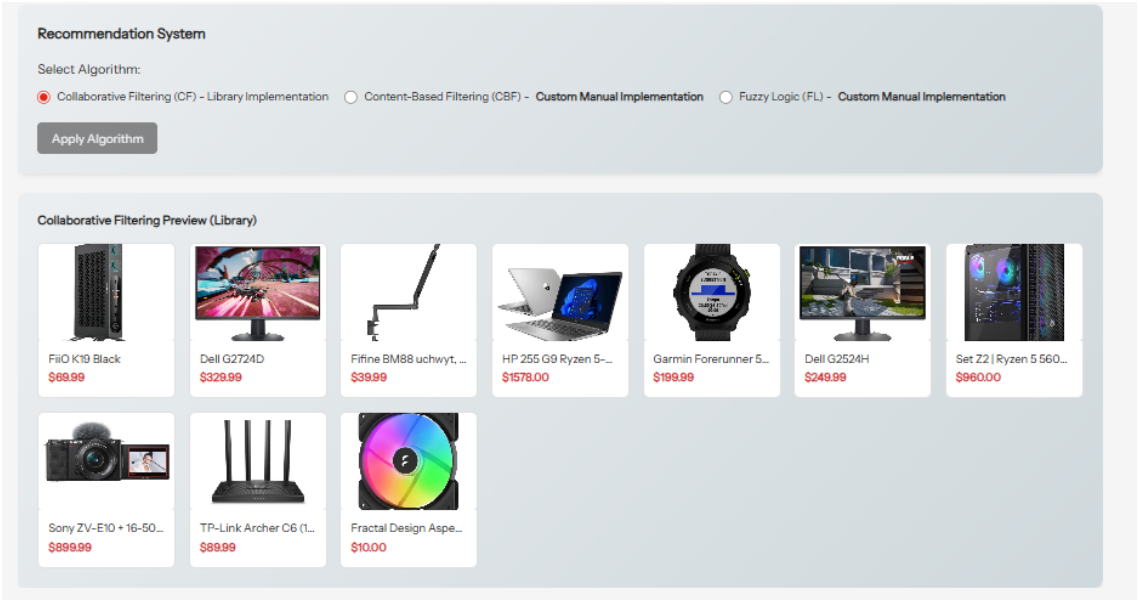
Warstwa danych (PostgreSQL) - tabele przechowujące prekalkulowane wyniki:

- `method_product_similarity` - macierz podobieństw Collaborative Filtering,

- `method_product_sentiment_summary` - zagregowany sentyment,
- `method_productassociation` - reguły Apriori,
- `method_recommendation_settings` - konfiguracja aktywnej metody.

6.2 Konfiguracja metod rekomendacji w panelu administratora

Panel administracyjny umożliwia dynamiczne przełączanie między trzema metodami rekomendacji wyświetlanymi na stronie głównej aplikacji.



Rysunek 8: Panel zarządzania metodami rekomendacji.

Rysunek 8 przedstawia panel konfiguracji z opcjami metod rekomendacyjnych. Administrator może wybrać aktywny algorytm, który będzie wykorzystywany we wszystkich sekcjach rekomendacji na stronie głównej. System zawiera łącznie sześć metod rekomendacyjnych zaimplementowanych w ramach dwóch prac inżynierskich. W ramach niniejszej pracy zaimplementowano:

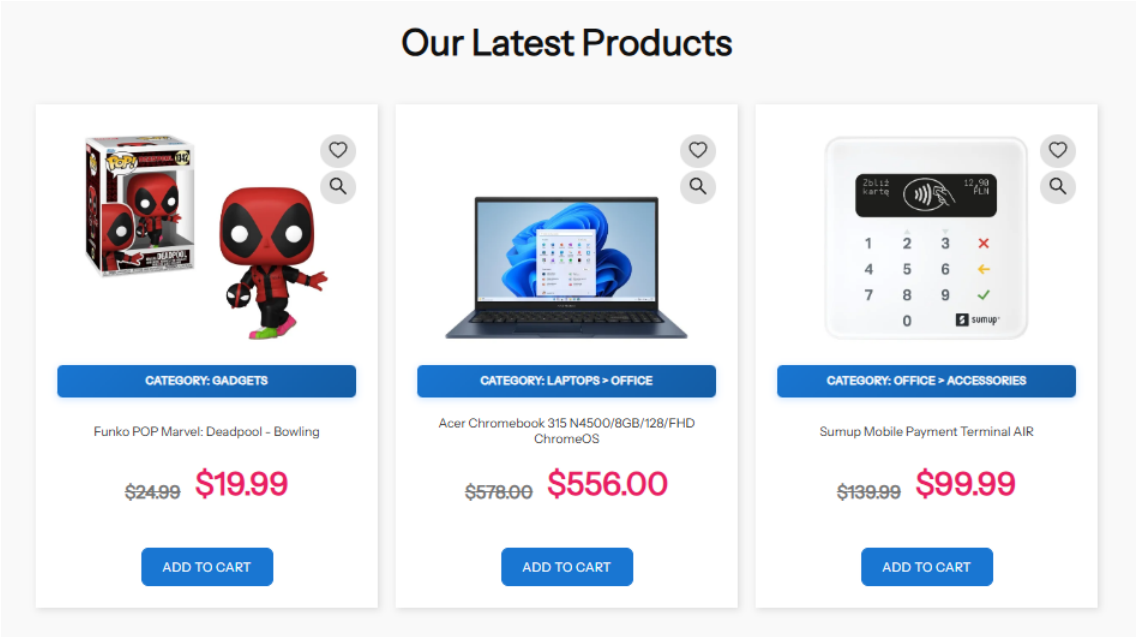
- **Collaborative Filtering** - rekomendacje oparte na historii zakupów użytkowników (opisane w niniejszej pracy).

Pozostałe metody (Content-Based Filtering, Fuzzy Logic) zostały zaimplementowane przez współautora systemu i opisane w odrębnej pracy inżynierskiej.

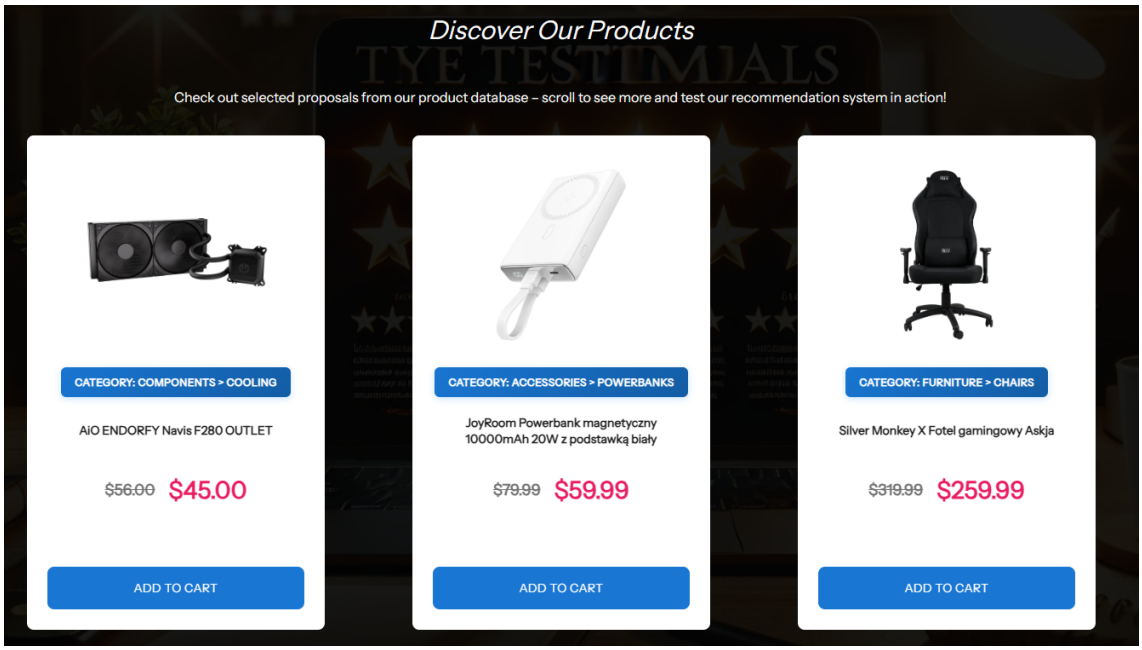
Zmiana metody następuje natychmiast po zapisaniu ustawień i wpływa na wszystkie sekcje rekomendacji w aplikacji.

6.3 Metoda Collaborative Filtering - Item-Based

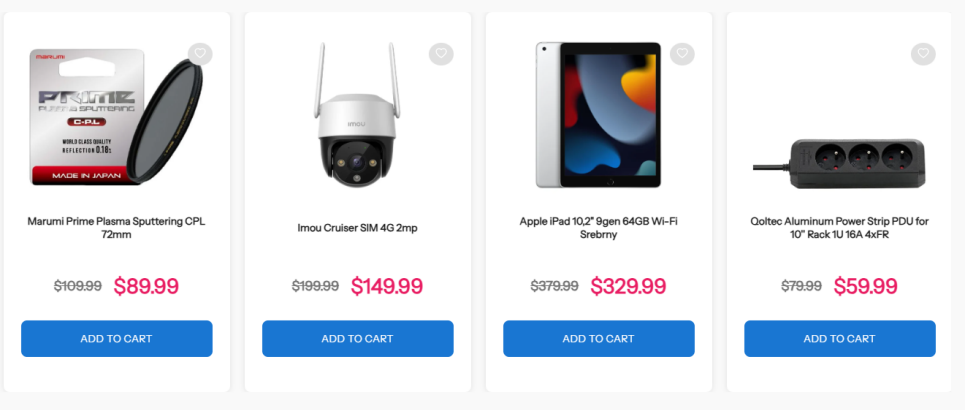
Widok strony głównej z rekomendacjami



Rysunek 9: Strona główna - pierwsza sekcja rekomendacji.



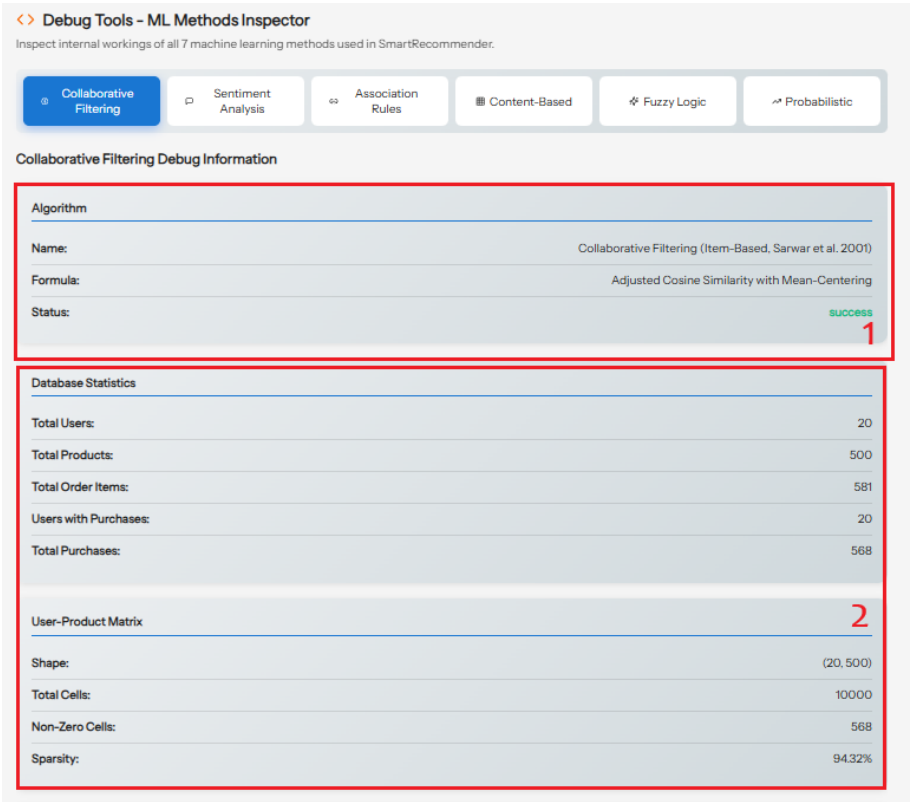
Rysunek 10: Strona główna - druga sekcja rekomendacji.



Rysunek 11: Strona główna - trzecia sekcja rekomendacji.

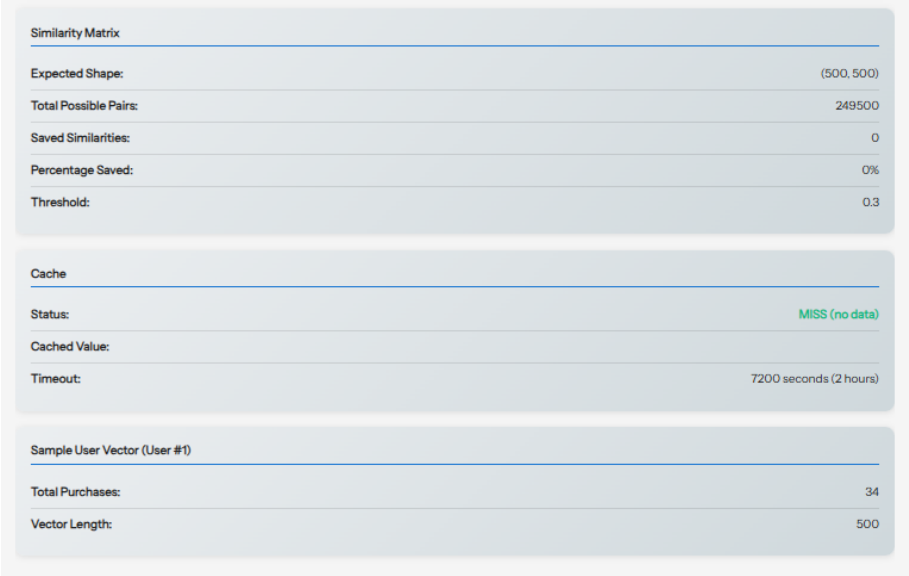
Rysunki 9, 10 i 11 pokazują trzy sekcje rekomendacji na stronie głównej. Wszystkie sekcje wyświetlają produkty dobrane według metody wybranej w panelu administratora (rys. 8). Gdy administrator wybierze Collaborative Filtering, wszystkie trzy sekcje będą prezentowały produkty dobrane według tej metody. Produkty są dobierane dynamicznie na podstawie podobieństwa obliczonego algorytmem Item-Based Collaborative Filtering z metryką Adjusted Cosine Similarity.

Panel debugowania Collaborative Filtering



Rysunek 12: Panel debugowania Collaborative Filtering - formuła algorytmu i dane z bazy.

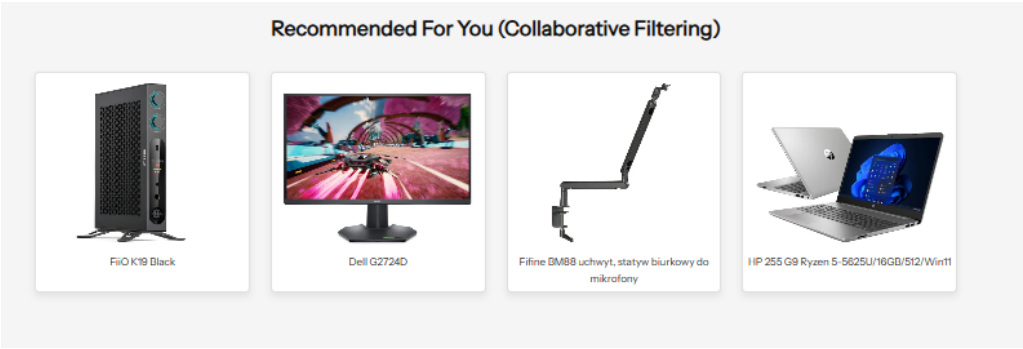
Rysunek 12 przedstawia sekcję debugowania algorytmu Collaborative Filtering w panelu administratora. Pod numerem (1) widoczna jest nazwa algorytmu „Item-Based Collaborative Filtering", formuła matematyczna Adjusted Cosine Similarity oraz status wykonania. Pod numerem (2) znajdują się dane z bazy danych - tabela `method_product_similarity` zawierająca pary produktów wraz z obliczonymi współczynnikami podobieństwa oraz fragment macierzy podobieństw w formacie wizualnym.



Rysunek 13: Panel debugowania Collaborative Filtering - podsumowanie obliczeń.

Rysunek 13 pokazuje drugą część panelu debugowania Collaborative Filtering. System wyświetla podsumowanie macierzy, statystyki cache oraz przykład rekomendacji dla pierwszego użytkownika z bazy danych.

Rekomendacje w panelu klienta

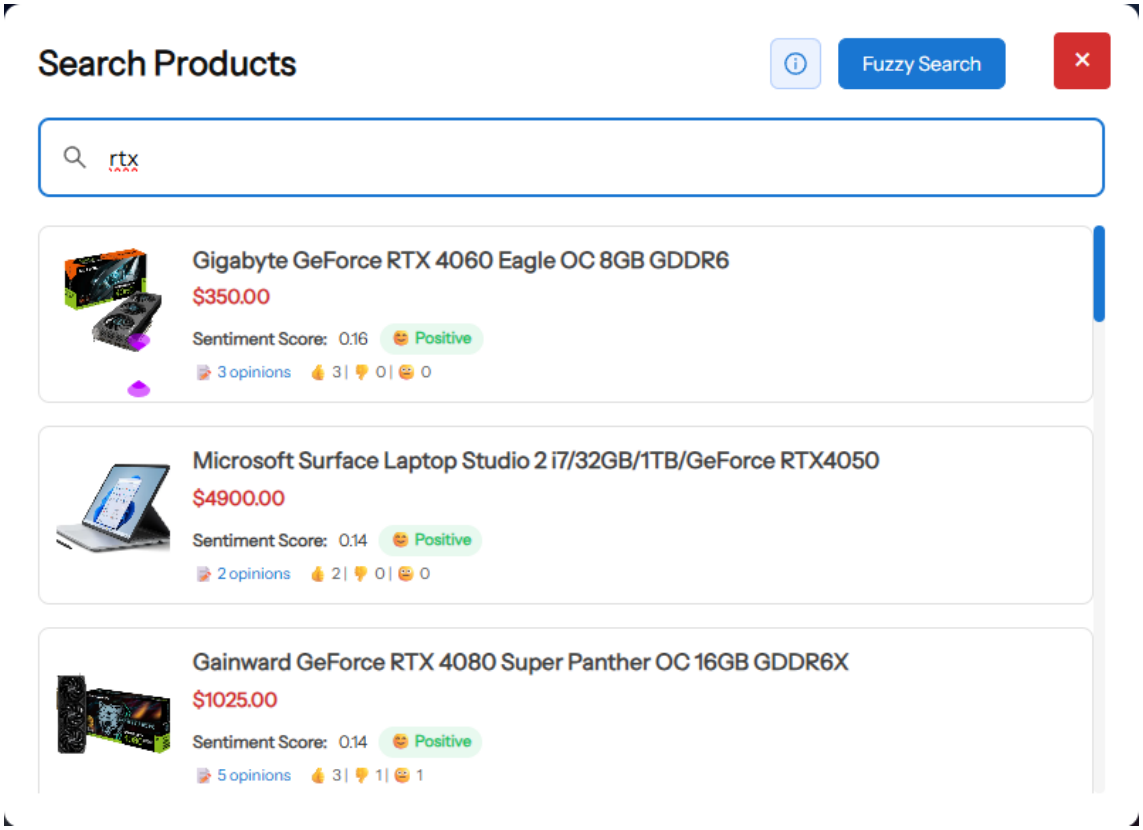


Rysunek 14: Dashboard klienta z sekcją rekomendacji.

Rysunek 14 przedstawia dashboard klienta z sekcją spersonalizowanych rekomendacji. Produkty są dobierane na podstawie pełnej historii zakupów użytkownika z zastosowaniem metody Collaborative Filtering.

6.4 Metoda analizy sentymentu

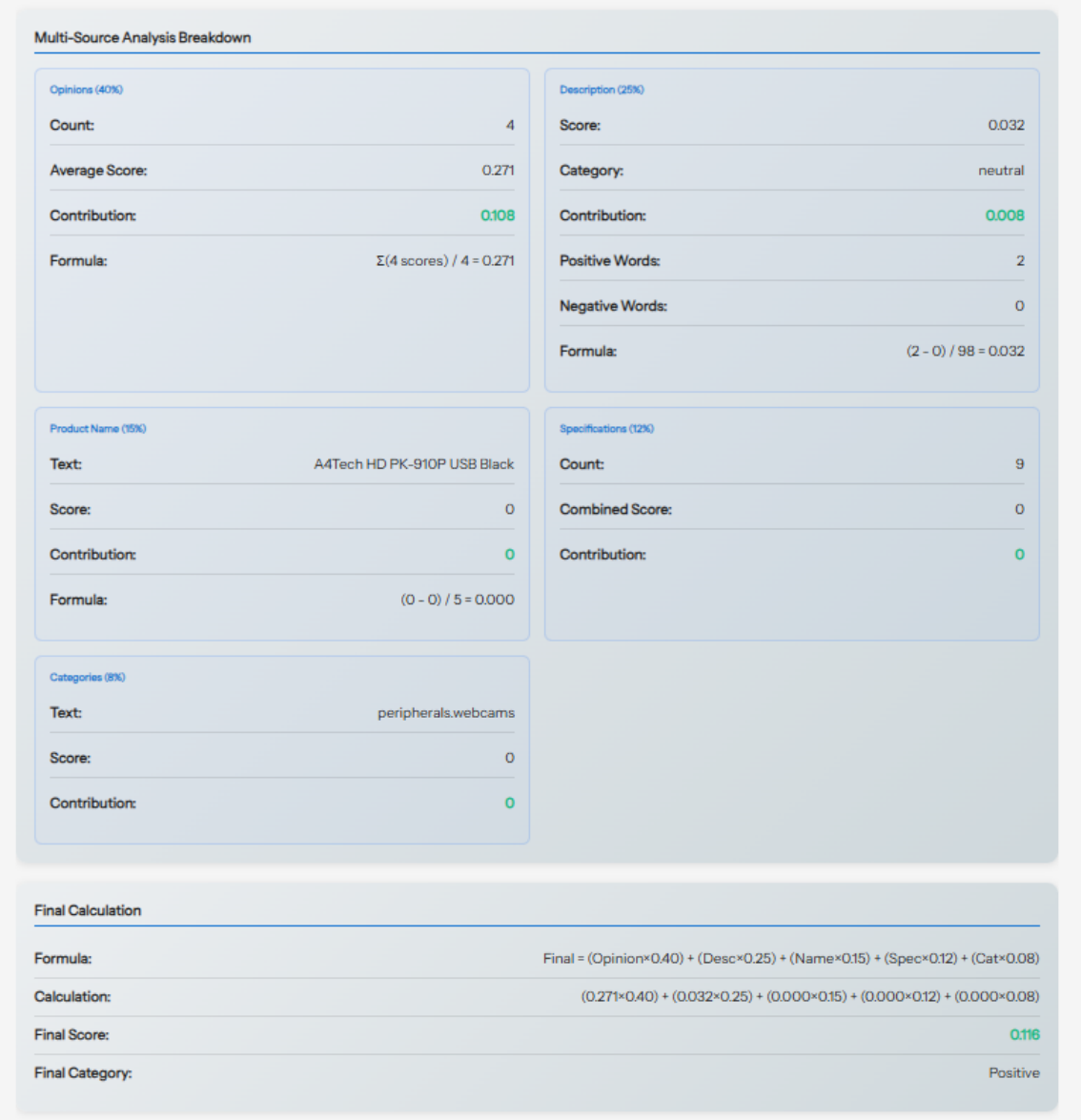
Wyszukiwarka z sortowaniem według sentymentu



Rysunek 15: Wyszukiwarka produktów z sortowaniem według sentymentu.

Rysunek 15 przedstawia wyszukiwarkę z wynikami posortowanymi według zagregowanego sentymentu. Każdy produkt wyświetla wynik sentiment score, kategorię oraz rozkład opinii.

Panel debugowania analizy sentymentu

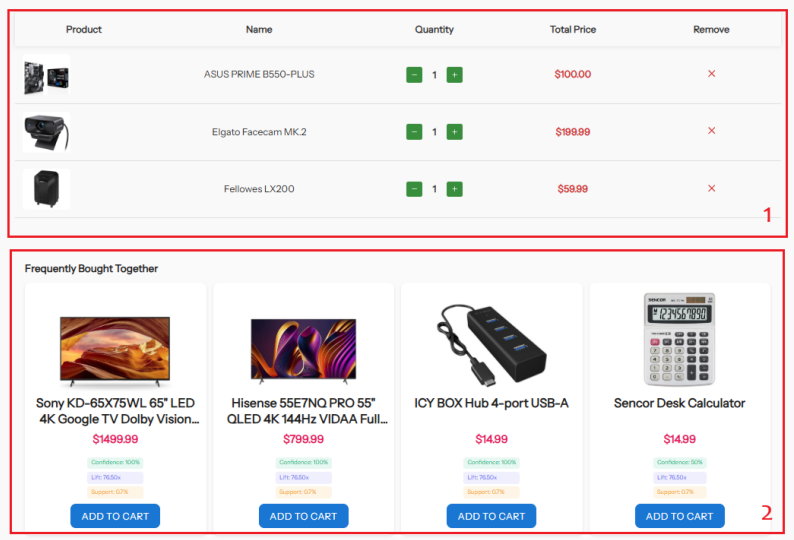


Rysunek 16: Panel debugowania analizy sentymentu.

Rysunek 16 pokazuje panel debugowania analizy sentymentu. Administrator może wybrać dowolny produkt z listy rozwijanej i sprawdzić wynik zagregowany, status kategorii oraz szczegółowe informacje z poszczególnych źródeł tekstowych.

6.5 Metoda reguł asocjacyjnych (Apriori)

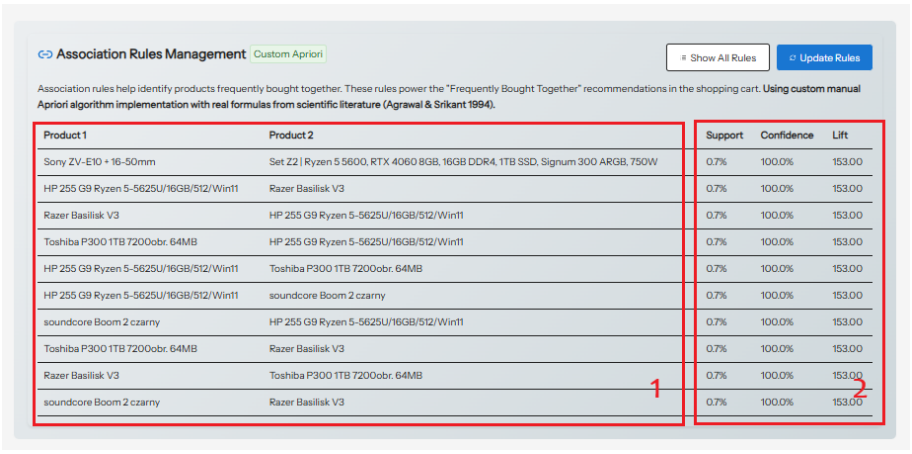
Widok rekomendacji Apriori na stronie produktu



Rysunek 17: Sekcja „Frequently Bought Together" na stronie produktu.

Rysunek 17 przedstawia sekcję „Frequently Bought Together" na stronie produktu. Pod numerem (1) widoczne są produkty dodane przez użytkownika do koszyka. Pod numerem (2) system wyświetla produkty proponowane na podstawie reguł asocjacyjnych wraz z metrykami lift, support i confidence.

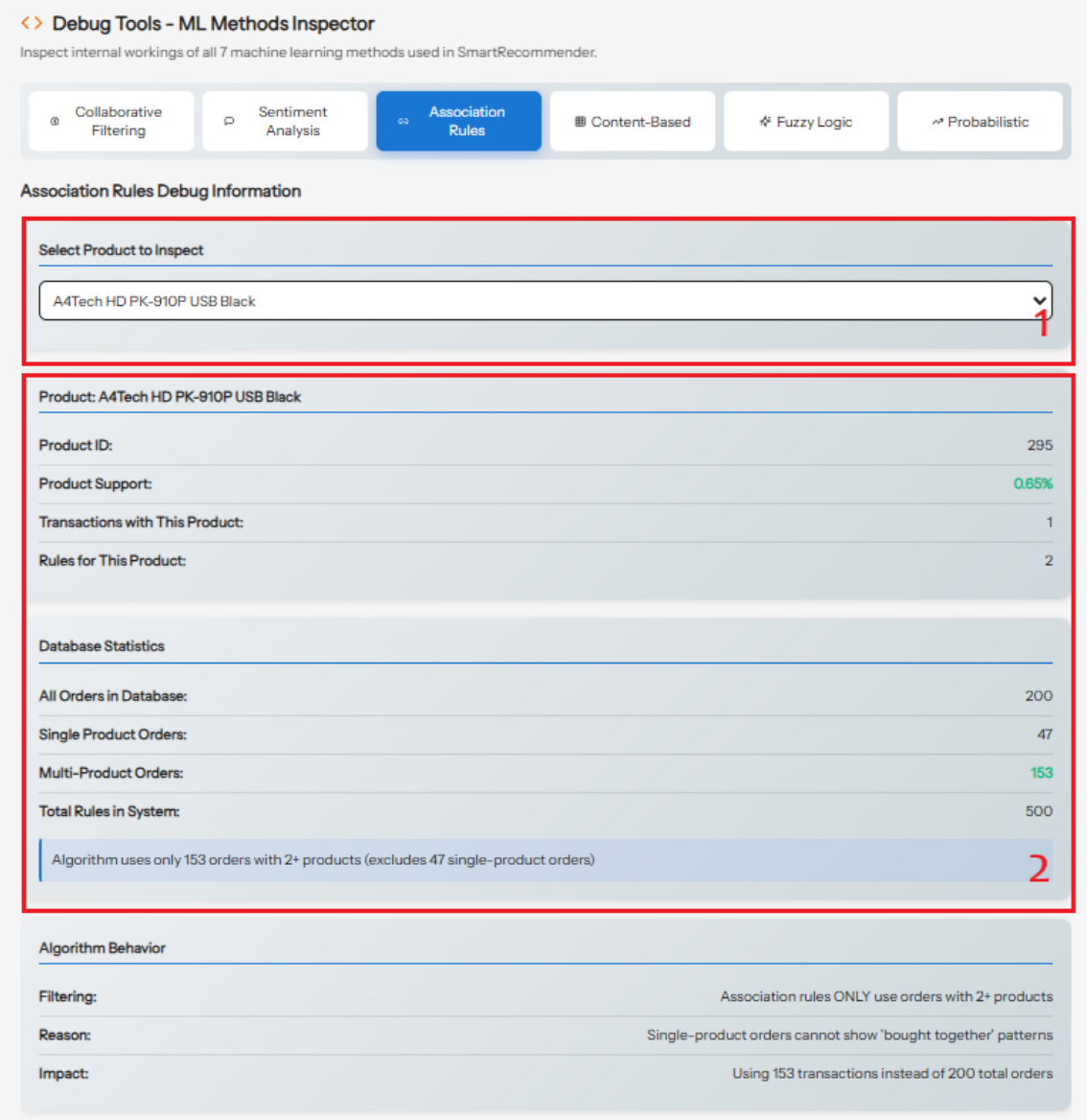
Panel zarządzania regułami asocjacyjnymi



Rysunek 18: Panel administracyjny zarządzania regułami Apriori.

Rysunek 18 pokazuje panel administracyjny z listą wygenerowanych reguł asocjacyjnych. Pod numerem (1) znajduje się lista par produktów często kupowanych razem. Pod numerem (2) wyświetlane są szczegółowe metryki każdej reguły - support, confidence i lift.

Panel debugowania reguł asocjacyjnych



Rysunek 19: Panel debugowania Apriori - wybór produktu i dane z bazy.

Rysunek 19 przedstawia pierwszy ekran panelu debugowania algorytmu Apriori. Pod numerem (1) administrator wybiera produkt z listy rozwijanej. Poniżej system wyświetla dane produktu z bazy danych, obliczenia transakcji oraz wszystkie reguły asocjacyjne dla wybranego produktu.

Rysunek 20: Panel debugowania Apriori - top 10 reguł i szczegóły analizy.

Rysunek 20 pokazuje drugą część panelu debugowania Apriori. Pod numerem (1) znajduje się ranking top 10 reguł asocjacyjnych dla wybranego produktu posortowanych według lift. Pod numerem (2) wyświetlane są szczegółowe analizy reguł dla produktów, w tym rozkład wartości lift, średnie metryki oraz porównanie z globalnymi statystykami.

6.6 Mechanizmy optymalizacji systemu

System wykorzystuje mechanizmy optymalizacji dla zapewnienia wydajności:

Prekalkulacja w bazie danych

Wszystkie trzy algorytmy zapisują wyniki do dedykowanych tabel z indeksami przyspieszającymi zapytania:

- `method_product_similarity` - macierz podobieństw Collaborative Filtering,
- `method_productassociation` - reguły asocjacyjne Apriori,
- `method_product_sentiment_summary` - zagregowany sentyment produktów.

Django Signals - automatyczna aktualizacja

System wykorzystuje mechanizm Django Signals do automatycznego przeliczania rekomendacji po zmianach danych:

- Nowe zamówienie - przeliczenie macierzy Collaborative Filtering,
- Nowa opinia - aktualizacja sentymentu produktu,
- Nowy produkt - automatyczna analiza sentymentu opisu i nazwy.

Operacje zbiorcze

System używa operacji zbiorczych (bulk operations) dla zapisu wielu rekordów jednocześnie, co znacząco przyspiesza operacje zapisu do bazy danych.

6.7 Podsumowanie funkcjonowania systemu

Zaimplementowany system rekomendacji charakteryzuje się:

Kompletnością zarządzania:

- Panel administratora z dynamicznym przełączaniem metod,
- Sekcje debugowania dla wszystkich trzech algorytmów,
- Wizualizacja metryk i statystyk w czasie rzeczywistym.

Integracją w aplikacji:

- Strona główna z konfigurowalnymi rekomendacjami,
- Dashboard klienta z personalizacją,
- Wyszukiwarka z sortowaniem według sentymentu,
- Sekcje „Frequently Bought Together” oparte na Apriori.

Narzędziami diagnostycznymi:

- Podgląd macierzy podobieństw Collaborative Filtering,
- Analiza źródeł sentymentu dla każdego produktu,
- Ranking reguł asocjacyjnych z metrykami,
- Statystyki wykonania algorytmów.

System jest w pełni funkcjonalny i gotowy do wdrożenia w środowisku produkcyjnym.

Rozdział 7

Podsumowanie i wnioski końcowe

Niniejsza praca przedstawiła proces implementacji oraz analizy kompletnego systemu e-commerce wyposażonego w mechanizmy rekomendacji produktów. System został opracowany od podstaw we współpracy dwuosobowej, przy czym w ramach niniejszej pracy zaimplementowano trzy metody rekomendacyjne: Collaborative Filtering z metryką Adjusted Cosine Similarity, analizę sentymentu opartą na podejściu słownikowym oraz reguły asocjacyjne wykorzystujące algorytm Apriori.

Ograniczenia systemu

W trakcie realizacji projektu zidentyfikowano następujące ograniczenia:

Problem zimnego startu — algorytmy Collaborative Filtering oraz Apriori wymagają historycznych danych o interakcjach użytkowników z produktami. Dla nowych użytkowników bez historii zakupów oraz nowych produktów bez opinii mechanizmy te nie są w stanie generować efektywnych rekomendacji. Analiza sentymentu częściowo kompensuje to ograniczenie, ponieważ może ocenić jakość produktu na podstawie jego opisu, specyfikacji technicznych oraz nazwy, nawet w przypadku braku opinii użytkowników.

Ograniczenia analizy sentymentu — zastosowane podejście słownikowe nie radzi sobie efektywnie z negacją językową (przykład: „nie polecam") oraz z ironią i sarkazmem. Słowa pozytywne w kontekście negatywnym mogą być błędnie klasyfikowane, co wpływa na dokładność oceny sentymentu. Rozwiązanie tego problemu wymagałoby zastosowania bardziej zaawansowanych technik przetwarzania języka naturalnego, takich jak modele kontekstowe.

Skalowalność dla bardzo dużych katalogów — dla katalogów produktów przekraczających tysiące pozycji mogą wystąpić wyzwania wydajnościowe wymagające dalszych optymalizacji, takich jak partycjonowanie danych, rozproszenie obliczeń lub zastosowanie dedykowanych struktur danych.

Kierunki dalszego rozwoju

Zidentyfikowano następujące kierunki rozwoju systemu:

Zastosowanie głębokiego uczenia maszynowego — obecny system wykorzystuje klasyczne algorytmy rekomendacyjne oparte na analizie podobieństw oraz regułach asocjacyjnych. Zastosowanie sieci neuronowych, takich jak autoencodery czy sieci rekurencyjne, mogłoby umożliwić automatyczne uczenie się ukrytych wzorców w danych bez konieczności ręcznego definiowania reguł. Przykładowo, sieci neu-

ronowe mogłyby odkryć nieoczywiste zależności między produktami oraz preferencjami użytkowników, które nie są widoczne w tradycyjnych metrykach podobieństwa.

Rekomendacje w czasie rzeczywistym — obecny system wykorzystuje mechanizm cache'owania z okresem ważności 24 godzin, co oznacza, że rekomendacje są przeliczane cyklicznie. Implementacja systemu aktualizującego rekomendacje w czasie rzeczywistym po każdej akcji użytkownika (przeglądanie produktów, dodawanie do koszyka, finalizacja zakupu) mogłaby znacząco zwiększyć trafność sugestii poprzez uwzględnienie bieżącego kontekstu sesji zakupowej.

Zaawansowane metody obsługi zimnego startu — zastosowanie technik faktoryzacji macierzy, takich jak Singular Value Decomposition (SVD), mogłoby umożliwić generowanie rekomendacji dla nowych produktów na podstawie ich cech (kategoria, cena, marka, specyfikacja) oraz analogii do istniejących produktów w katalogu.

Wnioski końcowe

Zrealizowany system stanowi kompletne rozwiązanie e-commerce z autorskimi mechanizmami rekomendacji produktów, gotowe do wdrożenia w środowisku produkcyjnym. Implementacja od podstaw bez wykorzystania gotowych bibliotek rekomendacyjnych umożliwiła pełne zrozumienie mechanizmów działania algorytmów oraz ich świadome dostosowanie do specyfiki handlu elektronicznego. Komplementarność zastosowanych metod — Collaborative Filtering dla identyfikacji produktów podobnych, analiza sentymentu dla oceny jakości oraz algorytm Apriori dla cross-sellingu — zapewnia wszechstronne wsparcie procesu decyzyjnego użytkownika. Zastosowane techniki optymalizacyjne, w tym bitmap pruning, cache'owanie oraz indeksowanie bazy danych, 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, rozwoju aplikacji.

Literatura

[1] Rakesh Agrawal, Ramakrishnan Srikant, *Fast Algorithms for Mining Association Rules*, Proceedings of the 20th International Conference on Very Large Data Bases (VLDB), 1994.

[2] James Bennett, Stan Lanning, *The Netflix Prize*, Proceedings of KDD Cup and Workshop, 2007.

[3] Jonathan L. Herlocker, Joseph A. Konstan, John Riedl, *Explaining Collaborative Filtering Recommendations*, Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work (CSCW), 2000.

[4] Greg Linden, Brent Smith, Jeremy York, *Amazon.com Recommendations: Item-to-Item Collaborative Filtering*, IEEE Internet Computing, 2003.

[5] Bing Liu, *Sentiment Analysis and Opinion Mining*, Synthesis Lectures on Human Language Technologies, Morgan & Claypool Publishers, 2012.

[6] Jacques Bughin, Michael Chui, James Manyika, *Ten IT-enabled business trends for the decade ahead*, McKinsey Quarterly, May 2013.

[7] Paul Resnick, Hal R. Varian, *Recommender Systems*, Communications of the ACM, 1997.

[8] Badrul Sarwar, George Karypis, Joseph Konstan, John Riedl, *Item-based Collaborative Filtering Recommendation Algorithms*, Proceedings of the 10th International Conference on World Wide Web (WWW), 2001.

[9] Mohammed J. Zaki, *Scalable Algorithms for Association Mining*, IEEE Transactions on Knowledge and Data Engineering, 2000.

Wykaz rysunków

Spis rysunków

Streszczenie

Tytuł pracy w języku polskim:
System rekomendacji produktów oparty na filtracji współpracy, analizie sentymentu i regułach asocjacyjnych

Tytuł pracy w języku angielskim:
Product Recommendation System Based on Collaborative Filtering, Sentiment Analysis, and Association Rules

Streszczenie:

Niniejsza praca inżynierska przedstawia projekt oraz implementację systemu rekomendacji produktów dla platformy e-commerce, łączącego trzy komplementarne metody rekomendacyjne: collaborative filtering, analizę sentymentu oraz reguły asocjacyjne. Celem było zaprojektowanie rozwiązania eliminującego problem przeładowania informacyjnego w sklepach internetowych poprzez dostarczanie użytkownikom spersonalizowanych rekomendacji.

Część teoretyczna obejmuje szczegółową analizę fundamentów matematycznych systemów rekomendacyjnych, w tym metryki Adjusted Cosine Similarity dla algorytmu Item-Based Collaborative Filtering, słownikowej analizy sentymentu z agregacją wieloźródłową oraz metryk support, confidence i lift dla reguł asocjacyjnych algorytmu Apriori. Przedstawiono również przegląd rozwiązań alternatywnych (Amazon Personalize, Google Recommendations AI, Apache Mahout) wraz z uzasadnieniem implementacji dedykowanego systemu.

Część praktyczna obejmuje implementację aplikacji webowej w architekturze Django REST Framework (backend) oraz React 18 (frontend) z bazą danych PostgreSQL. Zaimplementowano trzy algorytmy działające komplementarnie: Collaborative Filtering identyfikuje produkty podobne na podstawie wzorców zakupowych, analiza sentymentu agreguje oceny jakości z pięciu źródeł tekstowych, a algorytm Apriori odkrywa produkty często kupowane razem.

System został wyposażony w kompletny interfejs oferujący rekomendacje Collaborative Filtering w sekcjach strony głównej, rekomendacje Apriori w koszyku zakupowym, wyszukiwarkę z filtrowaniem według sentymentu oraz panel administracyjny z narzędziami debugowania algorytmów.

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

Słowa kluczowe:
systemy rekomendacji, collaborative filtering, analiza sentymentu, algorytm Apriori, machine learning, e-commerce, Django, React, PostgreSQL