

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 Grochowskiego

Rzeszów, 2026

Spis treści

Wstęp	3
Rozdział 1: Teoretyczne podstawy systemów rekomendacyjnych	5
Rozdział 2: Weryfikacja i analiza rozwiązań alternatywnych	7
Rozdział 3: Opis projektu planowanej aplikacji	13
3.1 Cel i zakres aplikacji	13
3.2 Typy użytkowników systemu	13
3.3 Wymagania funkcjonalne systemu	14
3.4 Diagram przypadków użycia	17
3.5 Architektura funkcjonalna systemu	17
3.6 Struktura bazy danych	18
Rozdział 4: Przedstawienie wykorzystanego stosu technologicznego oraz praktycznej realizacji projektu	23
4.1 Architektura systemu	23
4.2 Stos technologiczny backendu	23
4.3 Stos technologiczny frontendu	24
4.4 Baza danych PostgreSQL	25
4.5 Projekt techniczny systemu	25
4.5 Deployment i konteneryzacja Docker	26
4.7 Architektura i przepływ danych systemu rekomendacji	28
Rozdział 5: Implementacja algorytmów rekomendacji	29
5.1 Collaborative Filtering - Item-Based z Adjusted Cosine	30
5.2 Analiza Sentymentu - wieloźródłowa agregacja	33
5.3 Algorytm Apriori - reguły asocjacyjne	37
Rozdział 6: Funkcjonowanie systemu rekomendacji w praktyce	41
6.1 Metoda Collaborative Filtering - Item-Based	41
6.2 Metoda analizy sentymentu	45
6.3 Metoda reguł asocjacyjnych (Apriori)	48
6.4 Konfiguracja metod rekomendacji w panelu administratora	51
Rozdział 7: Porównanie i ewaluacja metod rekomendacyjnych	53
Rozdział 8: Podsumowanie i wnioski końcowe	58
Literatura	60
Wykaz ilustracji, rysunków, wykresów i tabel	61
Streszczenie	62

Wstęp

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 mechanizmy rekomendacji produktów. Aplikacja została opracowana od podstaw we współpracy dwuosobowej, w ramach której frontend, backend oraz baza danych zostały zrealizowane wspólnie, natomiast metody rekomendacyjne były implementowane osobno przez każdego z osobna. W ramach systemu zaimplementowano łącznie sześć różnych metod rekomendacyjnych, 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 świadomego 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, spe-

cyfikacje, kategorie), rozwiązuje 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

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 pojawiением się pierwszych platform e-commerce. Wczesne zastosowania komercyjne systemów rekomendacji obejmowały m.in. mechanizm „klienci którzy kupili ten produkt, kupili również” zastosowany przez Amazon.com, który analizował historię zakupów w celu sugerowania powiązanych produktów, co skutkowało znaczącym wzrostem sprzedaży krzyżowej (cross-selling) oraz poprawą doświadczenia użytkowników [4]. Istotnym przełomem była również publikacja wprowadzająca metodę Item-Based Collaborative Filtering z metryką Adjusted Cosine Similarity [8], która zyskała szerokie zastosowanie w praktyce przemyślowej, szczególnie w platformach e-commerce o dużej liczbie produktów. Konkursy i inicjatywy badawcze, takie jak Netflix Prize w latach 2006-2009, w którym fundusz nagród wynosił milion dolarów, znaczco 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 - video na żądanie).

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 zazwyczaj rzadka (stopień wypełnienia zależy od liczby produktów, użytkowników oraz częstotliwości transakcji).

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.

Rozdział 2

Weryfikacja i analiza rozwiązań alternatywnych

W celu uzasadnienia sensowności tworzenia dedykowanego systemu rekommendacji 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 sentymetru [5] oraz reguły asocjacyjne [1].

Amazon Personalize

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

Możliwości systemu: Amazon Personalize oferuje automatyczne skalowanie infrastruktury dostosowujące się do obciążenia, co pozwala obsługiwać zarówno małe sklepy internetowe, jak i platformy o skali Amazon.com. System automatycznie przetwarza dane użytkowników (kliknięcia, zakupy, wyświetlenia produktów) i generuje modele predykcyjne bez konieczności ręcznej optymalizacji hiperparametrów. Platforma zapewnia integrację z pozostałymi usługami AWS poprzez SDK dostępne w językach Python, Java, JavaScript oraz .NET, co umożliwia szybkie wdrożenie w istniejących aplikacjach. System automatycznie aktualizuje modele w czasie rzeczywistym na podstawie nowych interakcji użytkowników, zapewniając aktualność rekommendacji.

Kluczowe ograniczenia w kontekście planowanego rozwiązania:

- **Wysokie koszty operacyjne** - rozwiązań 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. Wielozienna agregacja sentymentu (opinie + opis + nazwa + specyfikacje + kategorie) zastosowana w niniejszej pracy wymaga integracji z dodatkowymi usługami AWS lub samodzielnej implementacji,

- **Uzależnienie od dostawcy** (ang. vendor lock-in) - głęboka integracja z eko-systemem 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.

Google Recommendations AI (Vertex AI)

Google Recommendations AI to platforma GCP wykorzystująca deep learning oraz algorytmy wieloramiennych bandytów kontekstowych (ang. multi-armed contextual bandits). Algorytmy te dynamicznie balansują eksplorację (testowanie nowych produktów w celu poznania ich wartości) z eksploatacją (rekommendowanie produktów o udowodnionej skuteczności), co pozwala systemowi optymalizować rekommendacje w czasie rzeczywistym na podstawie feedbacku użytkowników. System oferuje zaawansowane rekommendacje dla e-commerce, VOD (Video on Demand - video na żądanie) oraz platform newsowych, z automatycznym wykrywaniem trendów i sezonowości.

Możliwości systemu: Platforma zapewnia automatyczne wykrywanie trendów sezonowych oraz reagowanie na zmiany w zachowaniach użytkowników bez konieczności ręcznej rekonfiguracji. System oferuje funkcję „Frequently Bought Together” (często kupowane razem), która analizuje koszyki zakupowe w sposób podobny do algorytmu Apriori. Google Recommendations AI automatycznie optymalizuje parametry modeli poprzez mechanizmy AutoML, co eliminuje potrzebę ręcznego dostrajania hiperparametrów. Platforma oferuje zaawansowane możliwości personalizacji rekommendacji z uwzględnieniem kontekstu sesji użytkownika (urządzenie, lokalizacja, pora dnia), co może zwiększać współczynnik konwersji.

Kluczowe ograniczenia w kontekście planowanego rozwiązania:

- **Model cenowy oparty na liczbie predykcji** - system rozlicza się według liczby wygenerowanych rekommendacji oraz ilości przetworzonych danych katalogowych, co może generować rosnące koszty wraz ze wzrostem ruchu użytkowników,

- **Brak wieloźródłowej agregacji sentymentu** - system nie wspiera agregacji sentymentu z wielu źródeł tekstowych (opinie, opis produktu, nazwa, specyfikacje, kategorie) jak w planowanym rozwiązaniu,
- **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.

Apache Mahout

Apache Mahout to otwartoźródłowy framework (ang. open-source framework) implementujący klasyczne algorytmy collaborative filtering [8] oraz faktoryzację macierzy (ang. matrix factorization - 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 algorytmach rozproszonych opartych na Apache Spark (ang. Spark-based distributed algorithms).

Możliwości systemu: Apache Mahout oferuje pełną kontrolę nad implementacją algorytmów rekomendacyjnych oraz możliwość dostosowania ich do specyficznych wymagań biznesowych. Framework eliminuje koszty licencyjne charakterystyczne dla rozwiązań komercyjnych, co czyni go atrakcyjnym dla organizacji o ograniczonym budżecie. System wykorzystuje Apache Spark do przetwarzania rozproszonych obliczeń, co umożliwia skalowanie do bardzo dużych zbiorów danych (miliony użytkowników, miliony produktów). Mahout oferuje implementacje zaawansowanych algorytmów faktoryzacji macierzy (ALS, SVD), które mogą osiągać wysoką jakość rekomendacji przy odpowiedniej konfiguracji. Jako projekt otwartoźródłowy, Mahout nie wiąże się z uzależnieniem od konkretnego dostawcy chmurowego.

Kluczowe ograniczenia w kontekście planowanego rozwiązania:

- **Wymóg zaawansowanej wiedzy technicznej** - konieczność konfiguracji klastra Apache Spark (środowisko przetwarzania rozproszonego), YARN resource manager (zarządcy 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 sentymantu** - Apache Mahout nie oferuje sentiment analysis. Wymagana jest integracja z zewnętrznymi bibliotekami (np. Stanford CoreNLP) lub samodzielna implementacja słownikowej analizy sentymantu,
- **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.

Analiza 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, 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:**
Każde z analizowanych rozwiązań wymaga dodatkowej integracji lub samodzielnej implementacji co najmniej jednej z trzech wybranych metod (Collaborative Filtering, Analiza Sentymentu, Reguły Asocjacyjne):
 - Amazon Personalize: oferuje Collaborative Filtering, wymagane dodatkowe usługi AWS (Amazon Comprehend dla analizy sentymantu) lub samodzielna implementacja reguł asocjacyjnych,
 - Google Recommendations AI: oferuje Collaborative Filtering oraz funkcję „Frequently Bought Together” (podobną do reguł asocjacyjnych), wymaga samodzielnej implementacji analizy sentymantu,
 - Apache Mahout: oferuje tylko Collaborative Filtering, wymaga samodzielnej implementacji analizy sentymantu oraz reguł asocjacyjnych.

Własna implementacja pozwala na spójną integrację wszystkich trzech metod w ramach jednej architektury systemowej oraz umożliwia dostosowanie logiki każdej metody do specyfiki branży e-commerce.

- **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:

- **Wieloźródłowa agregacja sentymantu** [5] z 5 źródeł tekstowych - rozwiązuje problem cold start (zimny start): produkty bez opinii użytkowników otrzymują wynik sentymantu 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 uzależnienia od dostawcy:**

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, dane treningowe, wywołania API) oraz ponownego trenowania modeli, co może trwać tygodnie i powodować degradację jakości rekomendacji.

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 Typy użytkowników systemu

System obsługuje trzech typów użytkowników, z których każdy posiada dostęp do różnych poziomów funkcjonalności. Struktura uprawnień opiera się na hierarchii dziedziczenia, gdzie każdy wyższy poziom dziedziczy wszystkie funkcjonalności poziomów niższych.

Gość - użytkownik niezalogowany, posiadający dostęp do podstawowych funkcji sklepu internetowego. Może przeglądać produkty, wyszukiwać i dodawać je do koszyka, jednakże nie może złożyć zamówienia bez utworzenia konta.

Klient - użytkownik zalogowany, który dziedziczy wszystkie funkcje gościa oraz dodatkowo ma dostęp do funkcjonalności transakcyjnych i personalizacji. Może składać zamówienia, śledzić ich status, zarządzać swoim kontem oraz korzystać ze spersonalizowanych rekomendacji generowanych przez zaimplementowane algorytmy.

Administrator - użytkownik z najwyższymi uprawnieniami, który dziedziczy wszystkie funkcje klienta oraz dodatkowo ma dostęp do narzędzi administracyjnych. Może zarządzać całym systemem: produktami, zamówieniami, użytkownikami oraz ma dostęp do panelów statystycznych i debugowania algorytmów rekomendacji.

Taki podział użytkowników według ról i uprawnień wpływa bezpośrednio na podział systemu na trzy główne obszary funkcjonalne: obszar publiczny (dostępny dla wszystkich), obszar klienta (wymaga zalogowania) oraz obszar administracyjny (wymaga uprawnień administratora).

3.3 Wymagania funkcjonalne systemu

System został zaprojektowany z uwzględnieniem następujących wymagań funkcjonalnych pogrupowanych według obszarów:

Autentykacja i autoryzacja:

- Rejestracja nowego użytkownika w systemie,
- Logowanie do systemu z walidacją danych,
- Wylogowanie i zarządzanie sesją (token JWT).

Przeglądanie i wyszukiwanie produktów:

- Przeglądanie katalogu produktów,
- Wyszukiwanie produktów full-text w nazwie i opisie,
- Filtrowanie według kategorii, zakresu cenowego, sentymentu,
- Sortowanie według ceny, popularności, sentymentu.

Obsługa koszyka zakupowego:

- Dodawanie produktów do koszyka,
- Wyświetlanie zawartości koszyka z sumą całkowitą,
- Modyfikacja ilości produktów w koszyku,
- Usuwanie produktów z koszyka,
- Wyświetlanie rekomendacji Apriori („Często kupowane razem”).

Obsługa zamówień (tylko użytkownicy zalogowani):

- Składanie zamówień z danymi dostawy i płatności,
- Wyświetlanie historii zamówień użytkownika,

- Śledzenie statusu zamówienia (oczekujące, w realizacji, zakończone, anulowane),
- Wyświetlanie szczegółów zamkniętych zamówień.

Zarządzanie kontem użytkownika:

- Wyświetlanie profilu użytkownika,
- Edycja danych osobowych (imię, nazwisko, adres e-mail),
- Zmiana hasła,
- Przeglądanie historii aktywności.

System rekomendacji:

- Wyświetlanie rekomendacji Collaborative Filtering na stronie głównej,
- Wyświetlanie produktów podobnych na stronie szczegółów produktu,
- Wyświetlanie rekomendacji opartych na sentymencie (produkty o najlepszych opiniach),
- Wyświetlanie rekomendacji Apriori w koszyku zakupowym.

Opinie i recenzje:

- Wyświetlanie opinii innych użytkowników na stronie produktu,
- Dodawanie opinii do zakupionych produktów (ocena gwiazdkowa i treść tekstuwa),
- Automatyczne przeliczanie sentymentu po dodaniu opinii.

Funkcje administracyjne - zarządzanie produktami:

- Dodawanie nowych produktów do katalogu,
- Edycja istniejących produktów (nazwa, opis, cena, kategorie, specyfikacje),
- Usuwanie produktów z potwierdzeniem,
- Zarządzanie kategoriami produktów.

Funkcje administracyjne - zarządzanie zamówieniami:

- Przeglądanie wszystkich zamówień w systemie,
- Zmiana statusów zamówień,
- Wyświetlanie szczegółów zamówienia,
- Generowanie raportów sprzedażowych.

Funkcje administracyjne - zarządzanie użytkownikami:

- Lista wszystkich użytkowników w systemie,
- Nadawanie uprawnień administratora,
- Usuwanie kont użytkowników,
- Przeglądanie aktywności użytkowników.

Funkcje administracyjne - statystyki i analityka:

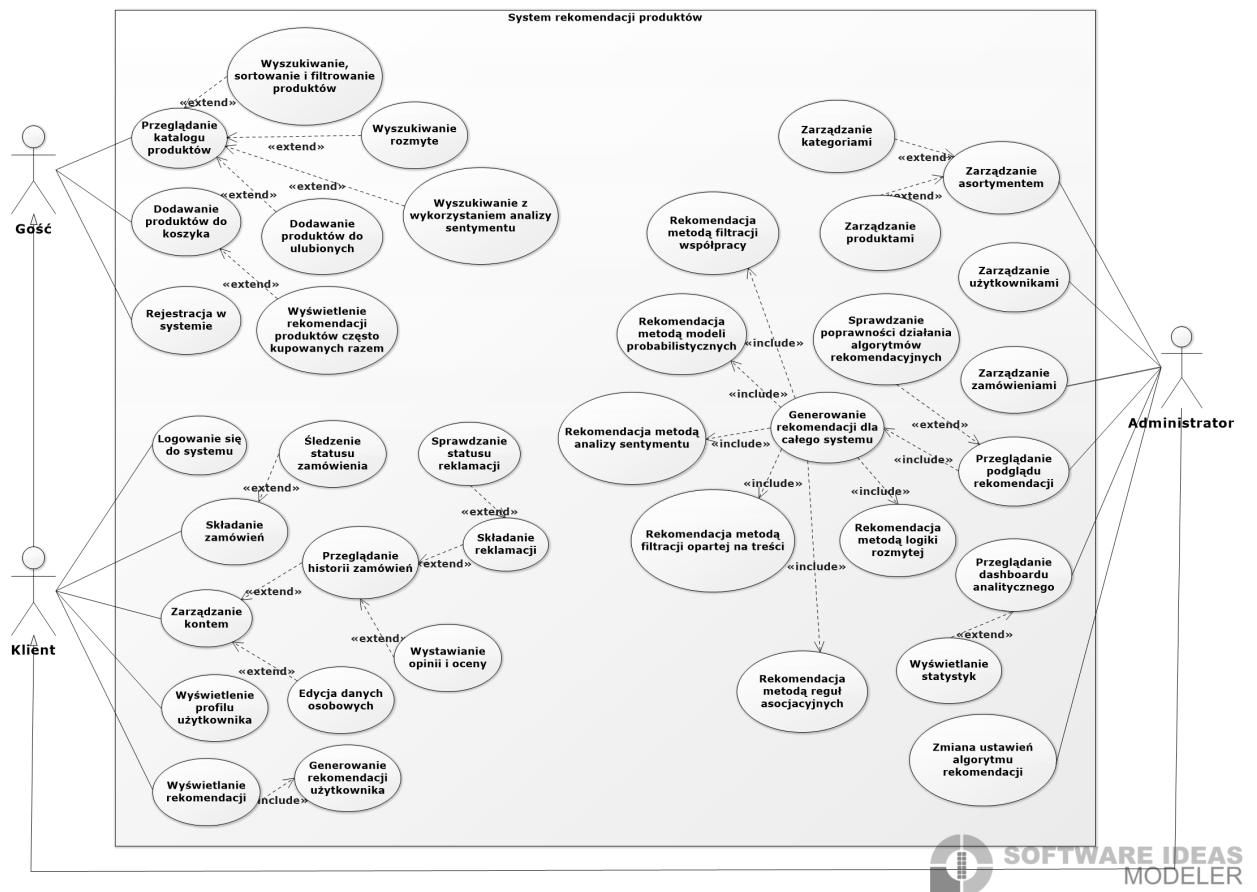
- Panel analityczny (dashboard) z kluczowymi wskaźnikami,
- Wykresy sprzedaży (miesięczny obrót, najpopularniejsze kategorie),
- Statystyki użytkowników (nowe rejestracje, aktywni użytkownicy),
- Statystyki produktów (najczęściej kupowane, najlepsze opinie).

Funkcje administracyjne - debugowanie algorytmów rekomendacji:

- Podgląd macierzy podobieństw Collaborative Filtering,
- Przeglądanie reguł asocjacyjnych Apriori z metrykami,
- Analiza dekompozycji sentymetu produktów,
- Walidacja poprawności działania algorytmów,
- Wyświetlanie statystyk wydajności metod rekomendacji.

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).



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.

3.5 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, validacja danych.

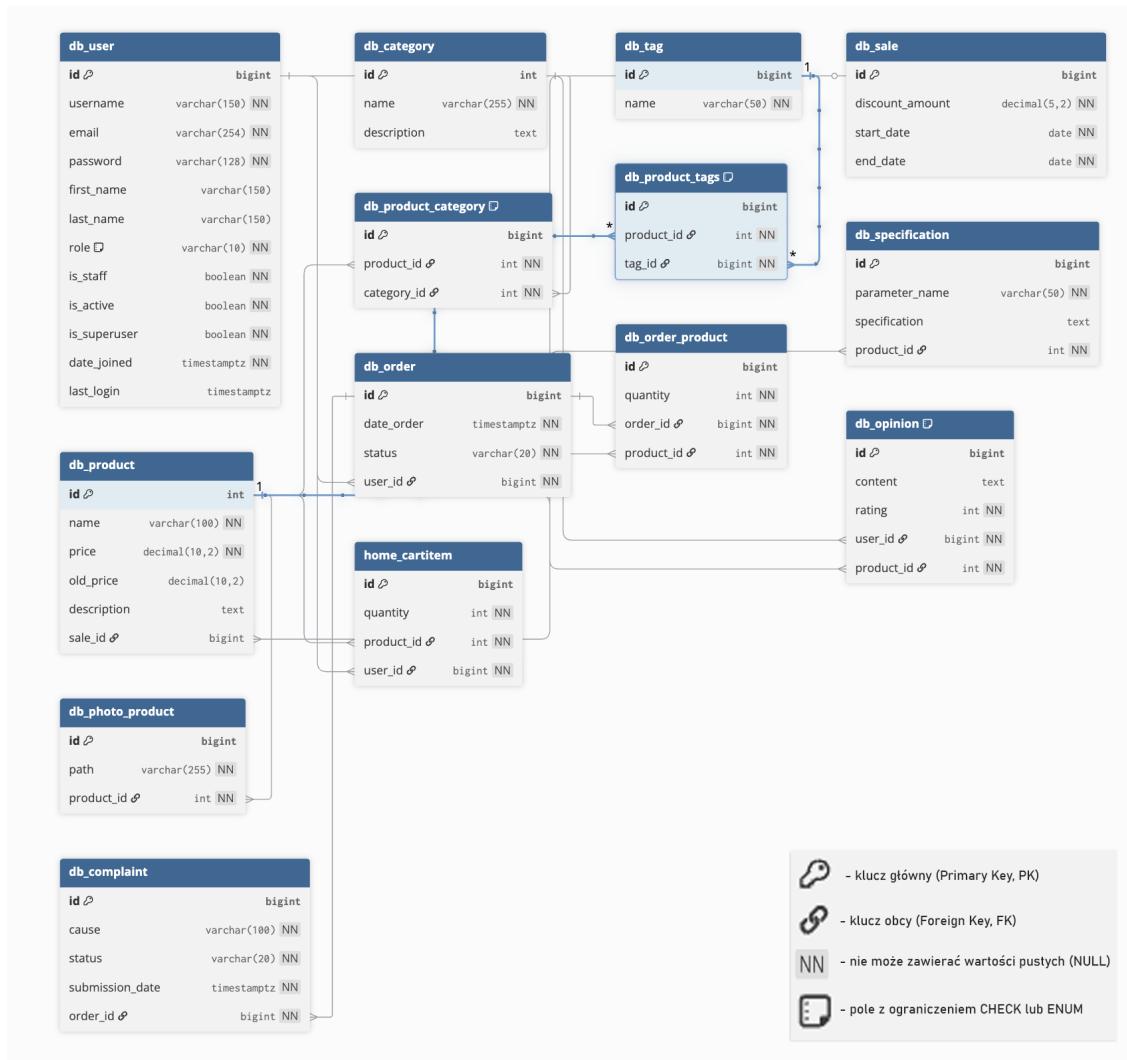
Warstwa danych - relacyjna baza danych PostgreSQL przechowująca:

- 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.

3.6 Struktura bazy danych

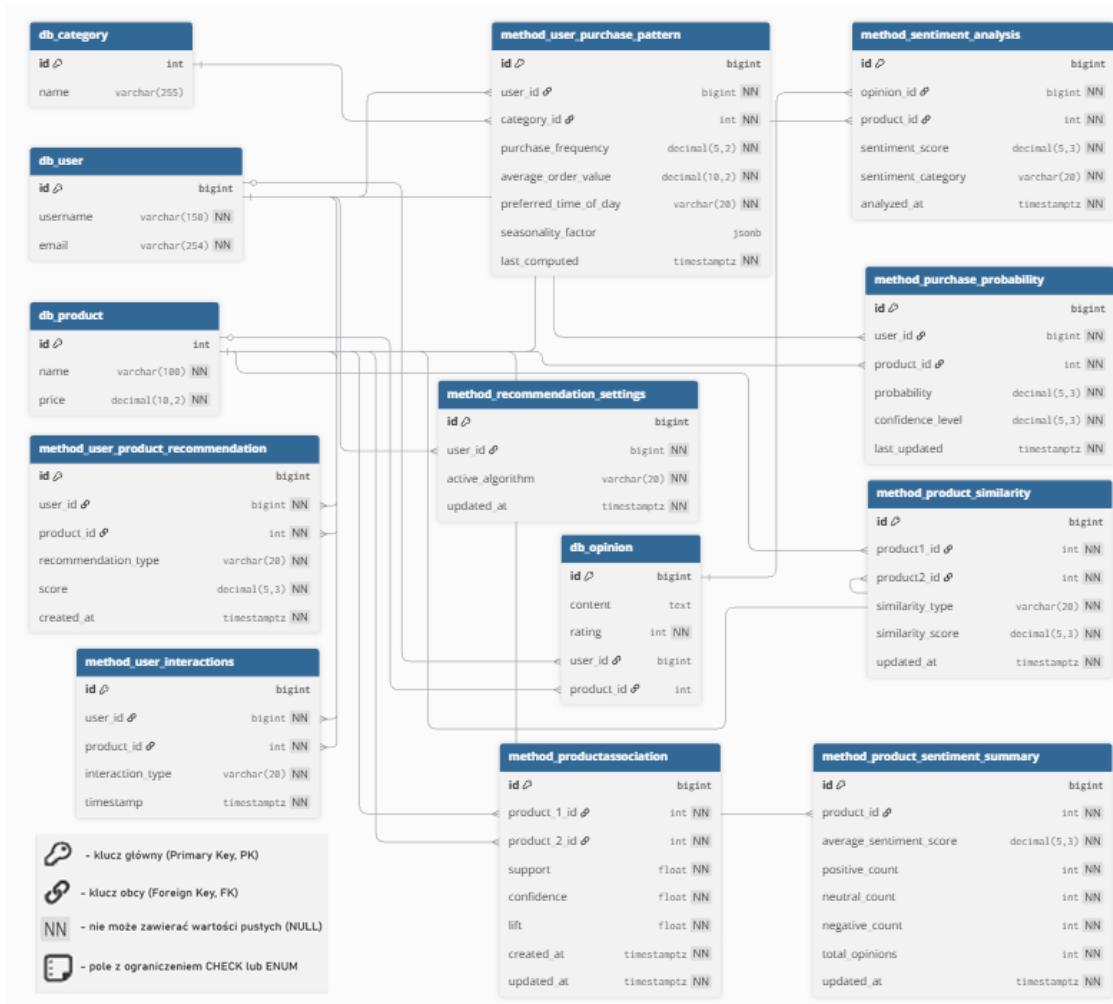
Baza danych systemu składa się z 25 tabel zorganizowanych w cztery moduły funkcjonalne. Poniższe diagramy ERD (Entity-Relationship Diagram - diagram związków encji) przedstawiają strukturę relacyjną bazy danych wraz z kluczowymi powiązaniami między tabelami.



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.

Charakterystyka tabel 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 - 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 (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`.

Wypełnianie bazy danych początkowymi (seeding)

W celu umożliwienia testowania systemu oraz walidacji algorytmów rekomenacyjnych zaimplementowano mechanizm automatycznego wypełniania bazy danych testowymi danymi (ang. database seeding). Proces seedingów generuje: 500 produktów z opisami, cenami i specyfikacjami technicznymi, 20 użytkowników (5 administratorów + 15 klientów) z wypełnionymi profilami, 200 zamówień z realistycznymi timestampami, około 600 rekordów OrderProduct (przeciętnie 3 produkty na zamówienie), oraz przybliżenie 1750 opinii (średnio 3.5 opinii na produkt) z ocenami gwiazdkowymi i treściami tekstowymi. Dane testowe zostały zaprojektowane w taki sposób, aby odzwierciedlać realistyczne wzorce zachowań użytkowników w sklepie internetowym, co pozwala na efektywną walidację jakości generowanych rekomendacji przez algorytmy Collaborative Filtering, analizę sentymentu oraz reguły asocjacyjne.

Rozdział 4

Przedstawienie wykorzystanego stosu technologicznego oraz praktycznej realizacji projektu

Rozdział przedstawia techniczne aspekty implementacji systemu e-commerce wyposażonego w mechanizmy rekomendacji produktów. Omówiono stos technologiczny wykorzystany w projekcie, strukturę bazy danych oraz sposób wdrożenia aplikacji z wykorzystaniem konteneryzacji Docker.

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) - 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:

- **Serializers** - 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.

4.3 Stos technologiczny frontendu

React 18 - 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).

4.4 Baza danych PostgreSQL

Wybór PostgreSQL 14 - 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 rekommendacyjnych,
- **Full-text search** - wbudowane mechanizmy wyszukiwania tekstowego dla produktów.

4.5 Projekt techniczny systemu

Po przedstawieniu stosu technologicznego oraz struktury bazy danych, niniejsza sekcja charakteryzuje projekt techniczny systemu, ze szczególnym uwzględnieniem architektury komponentów backendu i frontendu.

Struktura backendu

Każdy komponent systemu backendowego posiada dedykowane pliki odpowiedzialne za różne aspekty funkcjonalności:

- **models.py** – definicje tabel (Product, Order, Opinion, ProductSimilarity),
- **serializers.py** – konwersja obiektów Django ↔ JSON,
- **views.py** – obsługa CRUD dla produktów, zamówień,
- **recommendation_views.py** – endpointy /api/generate-user-recommendations/, /api/recommendation-algorithm-status/,
- **sentiment_views.py** – endpointy /api/sentiment-search/, /api/sentiment-analysis-debug/,

- **association_views.py** – endpointy `/api/frequently-bought-together/`, `/api/update-association-rules/`,
- **signals.py** – automatyczna aktualizacja rekommendacji.

Architektura komponentów frontendu

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

4.6 Deployment i konteneryzacja Docker

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

	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
□	smartrecommender.project	-	-	-	2.53%	15 minutes ago	■ : 🗑
□	SmartRecommender-React-FRONT d47ec7a8c0f5	smartrecommender-project-frontend d47ec7a8c0f5	smartrecommender-project-frontend	3000:3000 ⚡	0.67%	15 minutes ago	■ : 🗑 1
□	SmartRecommender-Django-BACK 55e59f5b9bc8	smartrecommender-project-backend 55e59f5b9bc8	smartrecommender-project-backend	8000:8000 ⚡	1.66%	15 minutes ago	■ : 🗑 2
□	SmartRecommender-PostgreSQL-1 PostgreSQL-1 333adf11b817	postgres:16	postgres:16	5432:5432 ⚡	0.2%	15 minutes ago	■ : 🗑 3

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 (persistencja danych),
- Environment: POSTGRES_DB, POSTGRES_USER, POSTGRES_PASSWORD,
- Healthcheck: pg_isready.

Konteneryzacja Docker zapewnia kilka kluczowych korzyści dla projektu. Izolacja każdego serwisu w osobnym kontenerze eliminuje konflikty zależności, co jest szczególnie istotne przy różnych wersjach bibliotek między backendem (Python 3.11) a frontendem (Node 18). Przenośność obrazów Docker umożliwia uruchomienie aplikacji na dowolnym serwerze z silnikiem Docker bez konieczności ręcznej konfiguracji środowiska. Proces uruchomienia został uproszczony do pojedynczej komendy `docker-compose up`, która automatycznie inicjalizuje wszystkie trzy kontenery z odpowiednimi zależnościami i połączeniami sieciowymi. Dodatkowo architektura umożliwia łatwą skalowalność - w przypadku wzrostu ruchu możliwe jest uruchomienie wielu instancji kontenera backendu z zastosowaniem mechanizmów równoważenia obciążenia (load balancing).

4.7 Architektura i przepływ danych systemu rekomendacji

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/generate-user-recommendations/` - generowanie rekomendacji Collaborative Filtering,
- `/api/sentiment-search/` - wyszukiwanie produktów według sentymentu,
- `/api/frequently-bought-together/` - produkty z reguł asocjacyjnych Apriori,
- `/api/recommendation-algorithm-status/` - status algorytmów i debugowanie,
- `/api/admin/update-product-similarity/` - regeneracja macierzy podobieństw.

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.

Rozdział 5

Implementacja algorytmów rekommendacji

Szczegółowa implementacja trzech algorytmów rekommendacyjnych wraz z pseudokodami oraz diagramami sekwencji przedstawia praktyczne aspekty realizacji metod collaborative filtering, analizy sentymentu oraz reguł asocjacyjnych.

5.1 Podstawy matematyczne wykorzystanych algorytmów

Implementowane algorytmy opierają się na następujących formułach matematycznych, które stanowią fundament dla realizacji poszczególnych metod rekommendacyjnych.

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:

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

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:

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

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, gdzie A i B oznaczają zbiory produktów (np. $A = \{\text{laptop}\}$, $B = \{\text{mysz}\}$):

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

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

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

$$\text{Confidence}(A \rightarrow B) = \frac{\text{Support}(A, B)}{\text{Support}(A)} \quad (4)$$

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

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

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

5.2 Collaborative Filtering - Item-Based z Adjusted Cosine

Algorytm Collaborative Filtering w wariantie Item-Based oblicza podobieństwa między produktami na podstawie historii zakupów użytkowników, wykorzystując metrykę Adjusted Cosine Similarity wprowadzoną przez Sarwar et al. (2001) [8]. Kluczowym elementem algorytmu jest formuła matematyczna przedstawiona we wzorze (1) w sekcji 5.1, która oblicza podobieństwo między dwoma produktami i i j poprzez analizę wzorców ich współwystępowania w zakupach użytkowników z uwzględnieniem centrowania wartości względem średniej zakupów każdego użytkownika \bar{R}_u .

Proces implementacji składa się z trzech głównych etapów odpowiadających wzorowi (1): budowy macierzy użytkownik-produkt (algorytm 1), normalizacji wartości metodą mean-centering centrowanie względem \bar{R}_u (algorytm 1) oraz obliczenia podobieństw cosinusowych (algorytm 2). Normalizacja polega na odjęciu od każdej wartości zakupowej średniej zakupów danego użytkownika ($R_{u,i} - \bar{R}_u$), co eliminuje bias (wartości progowe) wynikające z różnych skali zakupowych - hurtownik kupujący po 100 sztuk oraz klient kupujący po 1 sztuce otrzymują porównywalne wagę po normalizacji, dzięki czemu algorytm wykrywa rzeczywiste wzorce preferencji niezależnie od skali transakcji.

Algorytm 1 przedstawia budowę macierzy użytkownik-produkt oraz jej normalizację. Algorytm iteruje przez wszystkie pozycje zamówień (linie 2-8), budując macierz gdzie wiersze reprezentują użytkowników, kolumny produkty, a wartości ilości zakupione. Następnie dla każdego użytkownika obliczana jest średnia jego zakupów (linia 16) i od każdej niezerowej wartości odejmowana jest ta średnia (linie 17-19), realizując centrowanie względem \bar{R}_u ze wzoru (1).

Algorytm 1: Budowa macierzy użytkownik-produkt z normalizacją

```

1  function buduj_macierz_cf():
2      pozycje = pobierz_wszystkie_pozycje_zamowien()

```

```

3     macierz = pusty_slownik()
4
5     for poz in pozycje:
6         uz_id = poz.order.user_id
7         prod_id = poz.product_id
8         macierz[uz_id][prod_id] = poz.quantity
9
10    M = konwertuj_na_tablice(macierz)
11    M_norm = macierz_zerowa(wymiary(M))
12
13    for i = 0 to liczba_wierszy(M) - 1:
14        zakupione = M[i] where M[i] > 0
15        if |zakupione| > 0 then
16            sr = srednia(zakupione)
17            for j = 0 to liczba_kolumn(M) - 1:
18                if M[i][j] > 0 then
19                    M_norm[i][j] = M[i][j] - sr
20
21    return M_norm

```

Algorytm 2 przedstawia obliczanie podobieństw cosinusowych między produktami. Macierz znormalizowana jest transponowana (linia 2), aby wiersze reprezentowały produkty, a następnie obliczane są podobieństwa cosinusowe (linia 3), które realizują licznik i mianownik wzoru (1). Algorytm stosuje próg podobieństwa zdefiniowany jako parametr funkcji (linia 1, domyślnie prog = 0.5), który jest używany w warunku (linia 11) - zapisywane są tylko pary produktów o podobieństwie powyżej progu, co redukuje rozmiar bazy danych o przybliżenie 60-80% przy zachowaniu najważniejszych relacji, ponieważ słabe podobieństwa (np. 0.1-0.3) mają niewielką wartość predykcyjną dla rekommendacji. Operacje zbiorcze bulk insert (linie 14-16) zapisują po 1000 rekordów jednocześnie, co znacznie przyspiesza zapis do bazy danych.

Algorytm 2: Obliczanie podobieństwa cosinusowego produktów

```

1 function oblicz_podobienstwa_cf(M_norm, prog = 0.5):
2     M_T = transponuj(M_norm)
3     sim = podobienstwo_cosinus(M_T)
4
5     usun_podobienstwa(typ='collaborative')
6     lista = pusta_lista()
7     n = liczba_produktow(M_T)
8
9     for i = 0 to n - 1:

```

```

10         for j = 0 to n - 1:
11             if i != j and sim[i][j] > prog then
12                 dopisz(lista, {prod1: i, prod2: j, wynik:
13                               sim[i][j]})
14
15             if |lista| >= 1000 then
16                 zapisz_zbiorczo(lista)
17                 lista = pusta_lista()
18
19         if |lista| > 0 then
20             zapisz_zbiorczo(lista)

```

Algorytm 3 przedstawia generowanie rekomendacji dla konkretnego użytkownika na podstawie wcześniej obliczonych podobieństw produktów. Algorytm zbiera historię zakupów użytkownika z zamówień oraz koszyka (linie 2-9), a następnie dla każdego zakupionego produktu wyszukuje najbardziej podobne produkty (linia 13). Wyniki są agregowane (linie 14-18) - jeśli dany produkt jest podobny do wielu produktów zakupionych przez użytkownika, otrzymuje sumę wszystkich podobieństw jako końcowy wynik rekomendacji, co wzmacnia rekomendacje produktów spójnych z ogólną historią preferencji użytkownika.

Algorytm 3: Generowanie rekomendacji CF dla użytkownika

```

1 function generuj_rekomendacje(uzytkownik, typ_algorytmu):
2     historia = pusta_lista()
3
4     for zam in pobierz_zamowienia(uzytkownik):
5         for p in zam.produkty:
6             dopisz(historia, p.id)
7
8     for poz in pobierz_koszyk(uzytkownik):
9         dopisz(historia, poz.produkt.id)
10
11    wyniki = pusty_slownik()
12    for prod_id in unikalne(historia):
13        podobne = pobierz_podobne(prod_id, typ_algorytmu,
14                                  limit=5)
15        for s in podobne:
16            if s.produkt2_id in wyniki then
17                wyniki[s.produkt2_id] += s.podobienstwo
18            else
19                wyniki[s.produkt2_id] = s.podobienstwo

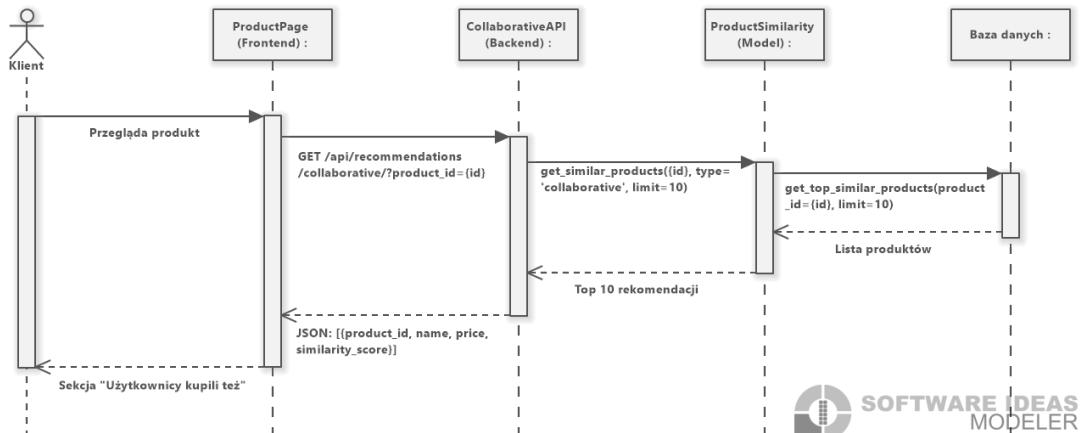
```

```

20     lista = sortuj(wyniki, malejaco)
21     for (p_id, wart) in lista:
22         zapisz_rekomendacje(uzytkownik, p_id, typ_algorytmu
, wart)

```

Diagram sekwencji: Collaborative Filtering



Rysunek 5: Diagram sekwencji: Collaborative Filtering.

Diagram 5 przedstawia przepływ procesu generowania rekomendacji Collaborative Filtering. Żądanie użytkownika trafia do API (`/api/generate-user-recommendations/`), które jako pierwszy krok sprawdza pamięć podręczną - w przypadku trafienia zwarcany jest wynik natychmiast, co znacznie przyspiesza odpowiedź systemu. Przy braku w pamięci podręcznej następuje zapytanie do bazy danych o historię zakupów użytkownika, na podstawie której budowana jest macierz użytkownik-produkt z OrderProduct (algorytm 1). Macierz jest normalizowana poprzez centrowanie wartości względem średniej każdego użytkownika, co eliminuje systematyczne zniekształcenie, po czym obliczane są podobieństwa cosinusowe między produktami zgodnie ze wzorem (1) - algorytm 2. Wyniki są zapisywane do pamięci podręcznej z czasem wygaśnięcia 2 godziny oraz tabeli ProductSimilarity z zachowaniem tylko podobieństw powyżej progu 0.5, po czym API zwraca top N rekomendacji (algorytm 3). Zaimplementowane optymalizacje obejmują pamięć podręczną przechowującą macierz podobieństw przez 2 godziny (7200 sekund), operacje zbiorcze bulk insert zapisujące po 1000 rekordów jednocześnie, próg 0.5 redukujący rozmiar bazy poprzez odrzucenie niższych wartości, oraz indeksowanie na kolumnach produkt_1 i typ_podobieństwa dla szybszego wyszukiwania.

5.3 Analiza Sentymentu - wieloźródłowa agregacja

Analiza sentymentu pojedynczego tekstu metodą słownikową (Liu 2012) [5] wykorzystuje formułę polarności przedstawioną we wzorze (2) w sekcji 5.1. Proces

analizy polega na normalizacji tekstu (konwersja na małe litery, usunięcie znaków specjalnych), tokenizacji oraz zliczaniu słów występujących w słownikach pozytywnym i negatywnym. Słowniki oparte na leksykonach akademickich AFINN-165 (Nielsen 2011) i Opinion Lexicon (Hu & Liu 2004) zawierają łącznie 400 słów (200 słów pozytywnych, 200 słów negatywnych) obejmujących najczęściej używane określenia jakościowe, emocjonalne i ocenne (np. pozytywne: „excellent”, „recommend”, „quality”, „reliable”; negatywne: „bad”, „poor”, „disappointing”, „defective”). Wynik wzoru (2) jest ograniczany do przedziału [-1, 1] i kategoryzowany: pozytywny (>0.1), negatywny (<-0.1), neutralny (pozostałe).

Wieloźródłowa agregacja sentymentu produktu stanowi kluczową innowację systemu, łącząc wyniki z 5 niezależnych źródeł tekstowych, gdzie każde S_i jest obliczane według wzoru (2). Wagi zostały dobrane empirycznie: opinie klientów 40% (najbardziej wiarygodne źródło), opis produktu 25% (profesjonalny opis zawierający kluczowe cechy), nazwa produktu 15% (często zawiera wskazówki jakościowe jak „Premium”, „Pro”), specyfikacje techniczne 12% (obiektywne parametry), kategorie produktu 8% (ogólny kontekst). Podejście to rozwiązuje fundamentalny problem zimnego startu - produkty bez opinii klientów nadal otrzymują wynik sentymentu na podstawie pozostałych czterech źródeł tekstowych.

Uwaga techniczna: System opinii opisany w rozdziale 6 stanowi jeden z pięciu źródeł tekstowych wykorzystywanych przez algorytm analizy sentymentu. Każda opinia dodana przez użytkownika jest automatycznie przetwarzana przez algorytm sentiment analysis i włączana do agregacji produktu z wagą 40%, co czyni opinie najważniejszym źródłem danych w ocenie jakości produktów.

Algorytm 4: Analiza sentymentu metoda słownikowa

```

1  function analizuj_sentiment(tekst, slownik_poz, slownik_neg
2    ) :
3      slowa = tokenizuj(na_male_literey(tekst))
4
5      if |slowa| = 0 then
6          return (0.0, „neutralny”)
7      end if
8
9      poz = 0
10     neg = 0
11
12     for s in slowa:
13         if s in slownik_poz then poz = poz + 1
14         if s in slownik_neg then neg = neg + 1
15     end for
```

```

15
16     wynik = (poz - neg) / |slowa|
17     wynik = ogranicz(wynik, -1.0, 1.0)
18
19     if wynik > 0.1 then kat = „pozytywny”
20     else if wynik < -0.1 then kat = „negatywny”
21     else kat = „neutralny”
22
23     return (wynik, kat)
24 end function

```

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.

Algorytm 5: Agregacja sentymentu produktu z 5 zródeł

```

1 function agreguj_sentiment_produktu(produkt):
2     opinie = pobierz_opinie(produkt)[:20]
3     wyniki_opinii = pusta_lista()
4     for op in opinie:
5         (w, _) = analizuj_sentiment(op.tresc)
6         dopisz(wyniki_opinii, w)
7     end for
8     S_op = srednia(wyniki_opinii) if |wyniki_opinii| > 0
9     else 0
10    (S_opis, _) = analizuj_sentiment(produkt.opis)
11    (S_nazwa, _) = analizuj_sentiment(produkt.nazwa)
12    teksty_spec = pusta_lista()
13    for sp in produkt.specyfikacje[:10]:
14        dopisz(teksty_spec, sp.nazwa + ‘ ’ + sp.wartosc)
15    end for
16    (S_spec, _) = analizuj_sentiment(polacz(teksty_spec))
17    kat_txt = polacz(pobierz_nazwy_kategorii(produkt))
18    (S_kat, _) = analizuj_sentiment(kat_txt)
19    S_final = 0.40*S_op + 0.25*S_opis + 0.15*S_nazwa
20        + 0.12*S_spec + 0.08*S_kat
21
22    poz = policz(wyniki_opinii WHERE w > 0.1)
23    neg = policz(wyniki_opinii WHERE w < -0.1)

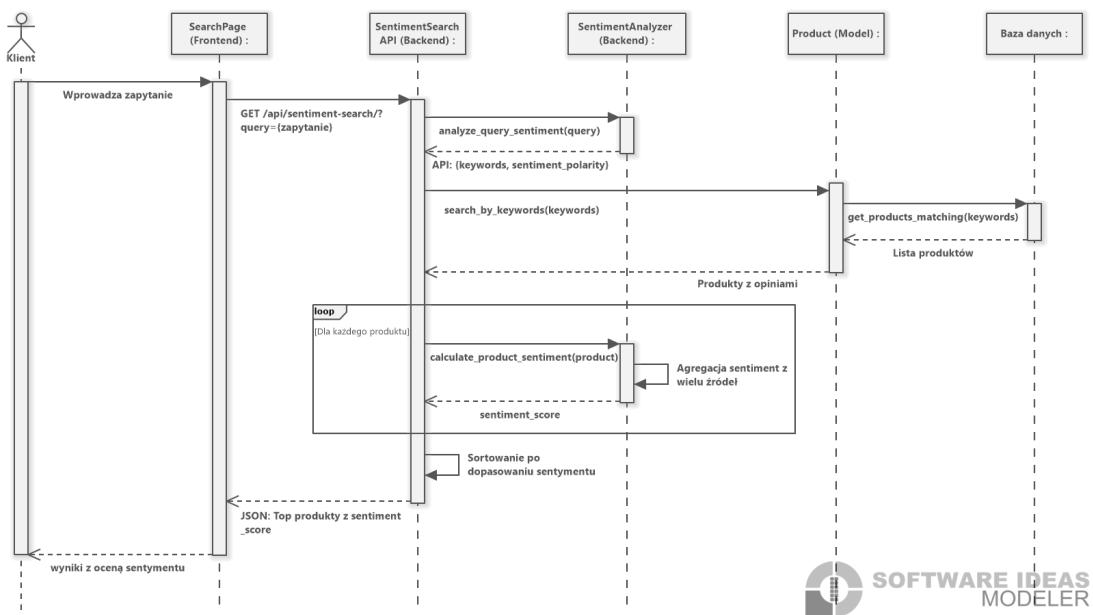
```

```

23     neu = policz(wyniki_opinii WHERE -0.1 <= w <= 0.1)
24
25     return {
26         wynik: zaokragl(S_final, 3),
27         pozytywne: poz,
28         negatywne: neg,
29         neutralne: neu
30     }
31 end function

```

Diagram sekwencji: Analiza Sentymentu



Rysunek 6: Diagram sekwencji: Analiza sentymentu.

Diagram 6 przedstawia przepływ procesu wieloźródłowej analizy sentymentu. Użytkownik wyszukuje produkty z sortowaniem według sentymentu, co inicjuje wywołanie funkcji agregacji sentymentu dla każdego produktu przez API. System analizuje 5 źródeł tekstowych równolegle z przypisanymi wagami: opinie 40%, opis 25%, nazwa 15%, specyfikacje 12%, kategorie 8%. Dla każdego źródła wykonywana jest tokenizacja i zliczanie słów pozytywnych oraz negatywnych przy użyciu słownników AFINN-165 i Opinion Lexicon. Wyniki są agregowane według formuły ważonej $S_{koncowy} = \sum_{i=1}^5 w_i \times S_i$, po czym końcowy wynik zapisywany jest w tabeli ProductSentimentSummary. API zwraca produkty posortowane według zagregowanego sentymentu. Kluczową zaletą tej architektury jest rozwiązywanie problemu zimnego startu - produkty bez opinii nadal otrzymują wynik sentymentu na podstawie pozostałych 4 źródeł tekstowych, co umożliwia sortowanie wszystkich produktów w katalogu niezależnie od ilości recenzji.

5.4 Algorytm Apriori - reguły asocjacyjne

Algorytm Apriori (Agrawal & Srikant 1994) [1] generuje reguły asocjacyjne typu „Często kupowane razem” poprzez analizę współwystępowania produktów w transakcjach, wykorzystując trzy fundamentalne metryki przedstawione we wzorach (3), (4) i (5) w sekcji 5.1:

Support (Wsparcie) - wzór (3) określa częstość współwystępowania produktów A i B w transakcjach. System używa minimalnego progu support = 0.001-0.005 (0.1-0.5% transakcji, wartość procentowa), co dla 200 zamówień oznacza że para produktów musi wystąpić w minimum 0.2-1 zamówieniu. Próg procentowy skaluje się automatycznie z rozmiarem bazy transakcji.

Confidence (Pewność) - wzór (4) oblicza warunkowe prawdopodobieństwo zakupu B przy założeniu zakupu A. Minimalny próg confidence = 0.01-0.1 (1-10%) oznacza że reguła jest zapisywana tylko jeśli przynajmniej 1-10% klientów kupujących A kupuje również B. Domyślna wartość confidence = 0.05 (5%) została dobrana empirycznie dla optymalnej równowagi między liczbą generowanych reguł a ich jakością.

Lift (Wzmocnienie) - wzór (5) mierzy siłę powiązania między produktami. Interpretacja: lift > 1 wskazuje pozytywną korelację (zakup A zwiększa prawdopodobieństwo zakupu B), lift = 1 oznacza niezależność produktów, lift < 1 wskazuje negatywną korelację. System używa progu lift = 1.0, preferując reguły z lift > 1.5 jako szczególnie wartościowe dla strategii cross-sellingu.

Proces implementacji składa się z dwóch głównych etapów: (1) znajdowania częstych par produktów z obliczeniem Support według wzoru (3), oraz (2) generowania reguł z obliczeniem Confidence i Lift według wzorów (4) i (5). Kluczową optymalizacją jest przycinanie bitmapowe (Zaki 2000), które reprezentuje transakcje jako liczby całkowite z ustawnionymi bitami odpowiadającymi produktom. Sprawdzenie czy transakcja zawiera parę produktów wymaga jednej operacji bitowej AND zamiast iteracji po liście, co znaczaco przyspiesza obliczenia. 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 o około 80-90%.

Algorytm 6: Apriori - znajdowanie częstych par produktow

```
1  function znajdz_czeste_pary(transakcje, min_wsp):
2      n = |transakcje|
3      min_licz = podloga(min_wsp * n)
4
5      liczniki = pusty_slownik()
6      for t in transakcje:
```

```

7      for p in t:
8          liczniki[p] = liczniki[p] + 1
9      end for
10 end for
11
12 czeste = [p for (p, l) in liczniki if l >= min_licz]
13
14 mapa = pusty_slownik()
15 for i = 0 to |czeste| - 1:
16     mapa[czeste[i]] = i
17 end for
18
19 bitmapy = pusta_lista()
20 for t in transakcje:
21     bm = 0
22     for p in t:
23         if p in mapa then
24             bm = bm LUB (1 << mapa[p])
25         end if
26     end for
27     if bm != 0 then
28         dopisz(bitmapy, bm)
29     end if
30 end for
31
32 pary = pusty_slownik()
33 for i = 0 to |czeste| - 1:
34     bit_i = 1 << i
35     for j = i + 1 to |czeste| - 1:
36         bit_j = 1 << j
37         para_bm = bit_i LUB bit_j
38
39         licz = 0
40         for bm in bitmapy:
41             if (bm AND para_bm) == para_bm then
42                 licz = licz + 1
43             end if
44         end for
45
46         if licz >= min_licz then
47             pary[{czeste[i], czeste[j]}] = licz / n

```

```

48         end if
49     end for
50 end for
51
52 wsparcia = {p: liczniki[p]/n for p in czeste}
53 return (par, wsparcia)
54 end function

```

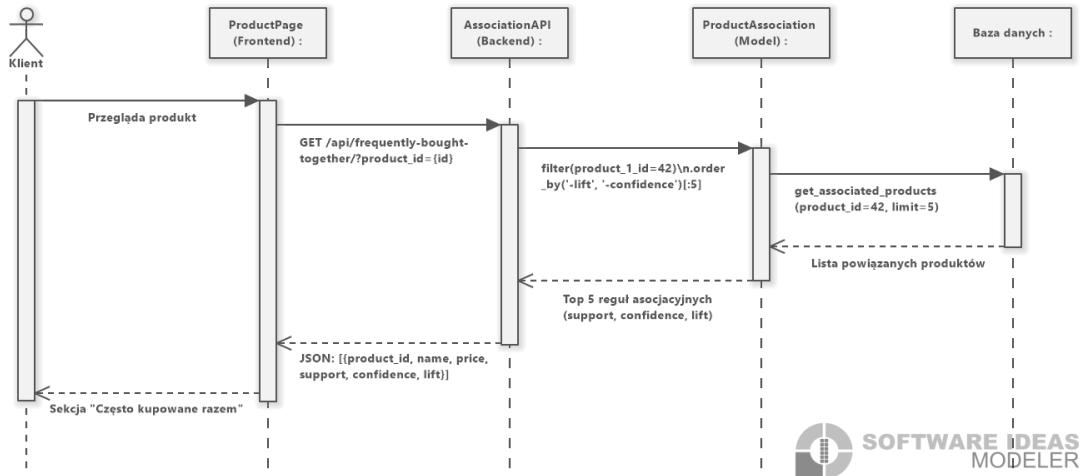
Algorytm 7: Apriori - generowanie reguł z obliczeniem lift i confidence

```

1 function generuj_reguły_z_par(par, wsparcia, min_pewn):
2     reguły = pusta_lista()
3
4     for ({p1, p2}, wsp Para) in par:
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        if pewn_1_2 >= min_pewn then
14            dopisz(reguły, {od: p1, do: p2,
15                           wsparcie: wsp Para,
16                           pewnosc: pewn_1_2,
17                           lift: lift})
18        end if
19
20        if pewn_2_1 >= min_pewn then
21            dopisz(reguły, {od: p2, do: p1,
22                           wsparcie: wsp Para,
23                           pewnosc: pewn_2_1,
24                           lift: lift})
25        end if
26    end for
27
28    sortuj(reguły, według=(lift, pewnosc), malejaco)
29    return reguły
30 end function

```

Diagram sekwencji: Algorytm Apriori



Rysunek 7: Diagram sekwencji: Algorytm Apriori.

Diagram 7 przedstawia przepływ procesu generowania reguł asocjacyjnych algorytmem Apriori. Administrator wywołuje aktualizację reguł przez panel administracyjny, co inicjuje ekstrakcję transakcji z bazy OrderProduct - system wybiera tylko zamówienia zawierające 2 lub więcej produktów, ponieważ pojedyncze zakupy nie tworzą asocjacji. Algorytm wykonuje wczesne przycinanie, eliminując rzadkie produkty o wsparciu poniżej progu 0.001 (0.1% transakcji), co znacznie redukuje przestrzeń obliczeniową - dzięki właściwości antymonotoniczności Apriori (jeśli produkt jest rzadki, wszystkie jego pary też są rzadkie) eliminuje się około 80-90% kandydatów. Transakcje konwertowane są do reprezentacji bitmapowej, gdzie każda transakcja to liczba całkowita z ustawionymi bitami odpowiadającymi produktom - pozwala to wykonywać sprawdzenia przynależności jedną operacją bitową AND zamiast iteracji po liście. System generuje częste 2-itemsety (pary produktów) używając operacji bitowych, po czym dla każdego częstego itemsetu obliczane są trzy metryki: Support według wzoru (3), Confidence według wzoru (4) i Lift według wzoru (5). Reguły filtrowane są według progów (support ≥ 0.001 , confidence ≥ 0.05 , lift ≥ 1.0) i zapisywane w bazie danych tabelą AssociationRule poprzez operacje zbiorcze bulk insert z rozmiarem partii 500 rekordów, co znacznie przyspiesza zapis. Zaimplementowane optymalizacje obejmują przycinanie bitmapowe dla szybkich operacji, wczesne przycinanie eliminujące większość kandydatów, operacje bitowe AND o optymalnej złożoności obliczeniowej, oraz operacje zbiorcze minimalizujące liczbę transakcji bazodanowych.

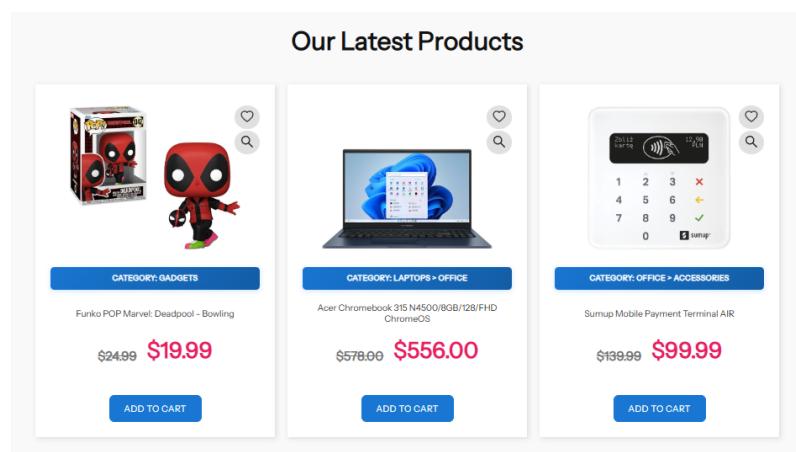
Rozdział 6

Funkcjonowanie systemu rekomendacji w praktyce

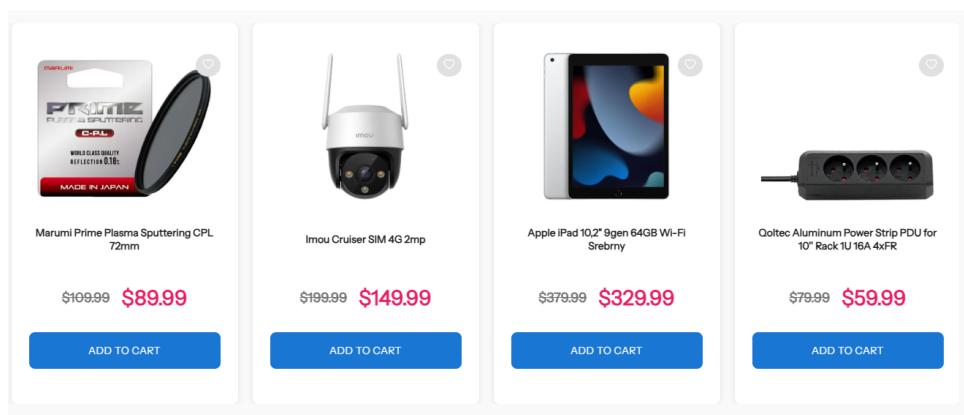
Rozdział przedstawia funkcjonowanie zaimplementowanych metod rekomendacyjnych w działającym systemie e-commerce. Omówiono perspektywę użytkownika końcowego korzystającego z rekomendacji produktów oraz narzędzia diagnostyczne służące do weryfikacji poprawności działania algorytmów. Struktura rozdziału została zaprojektowana zgodnie z naturalną kolejnością zapoznawania się z systemem - najpierw prezentacja funkcjonalności z perspektywy użytkownika końcowego, następnie panel konfiguracji dla administratora.

6.1 Metoda Collaborative Filtering - Item-Based

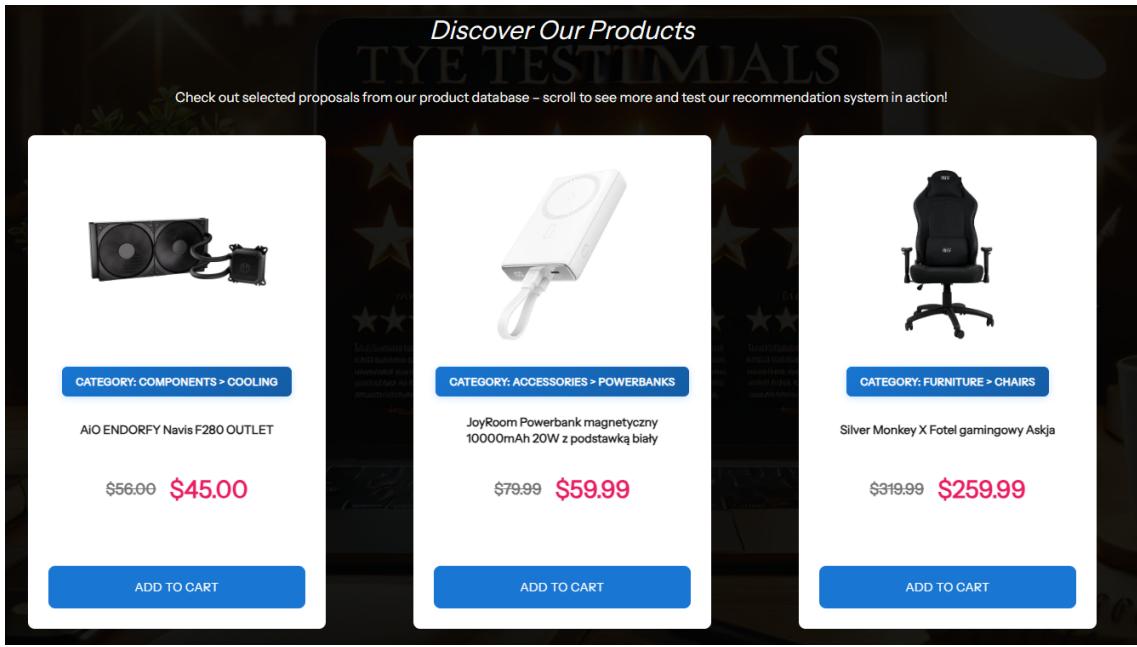
Widok strony głównej z rekomendacjami



Rysunek 8: Strona główna - pierwsza sekcja rekomendacji.



Rysunek 9: Strona główna - druga sekcja rekomendacji.



Rysunek 10: Strona główna - trzecia sekcja rekomendacji.

Rysunki 8, 9 i 10 przedstawiają trzy sekcje rekomendacji na stronie głównej aplikacji. Sekcje te są identyczne pod względem funkcjonalności i prezentują produkty w formie interaktywnych sliderów umożliwiających przewijanie oferty. Różnica między sekcjami polega wyłącznie na kolejności wyświetlania produktów - każda sekcja losuje inny podzbior z pełnej listy rekomendacji, co zwiększa różnorodność prezentowanych ofert i minimalizuje efekt monotonii. Wszystkie trzy sekcje dynamicznie dostosowują wyświetlane produkty w zależności od metody rekomendacyjnej wybranej przez administratora w panelu zarządzania (rys. 22). Gdy aktywny jest algorytm Collaborative Filtering, wszystkie sekcje prezentują produkty dobrane na podstawie podobieństwa obliczonego metryką Adjusted Cosine Similarity zgodnie ze wzorem (1) z sekcji 5.1.

Panel debugowania Collaborative Filtering

Panel debugowania Collaborative Filtering znajduje się w sekcji administracyjnej systemu i służy do weryfikacji poprawności działania algorytmu oraz analizy wygenerowanych rekomendacji. Narzędzie to jest przeznaczone dla administratorów systemu oraz programistów przeprowadzających testy działania algorytmu. Panel umożliwia podgląd macierzy podobieństw produktów, sprawdzenie statusu wykonania obliczeń, analizę przykładowych rekomendacji oraz weryfikację zgodności wyników z oczekiwaniemi biznesowymi. Dzięki temu administratorzy mogą szybko zdiagnozować ewentualne problemy w działaniu algorytmu, takie jak brak rekomendacji dla niektórych produktów, nieprawidłowe wartości podobieństwa czy błędy w strukturze danych.

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

Collaborative Filtering Debug Information

Algorithm	
Name:	Collaborative Filtering (Item-Based, Sarwar et al. 2001)
Formula:	Adjusted Cosine Similarity with Mean-Centering
Status:	success 1

Database Statistics

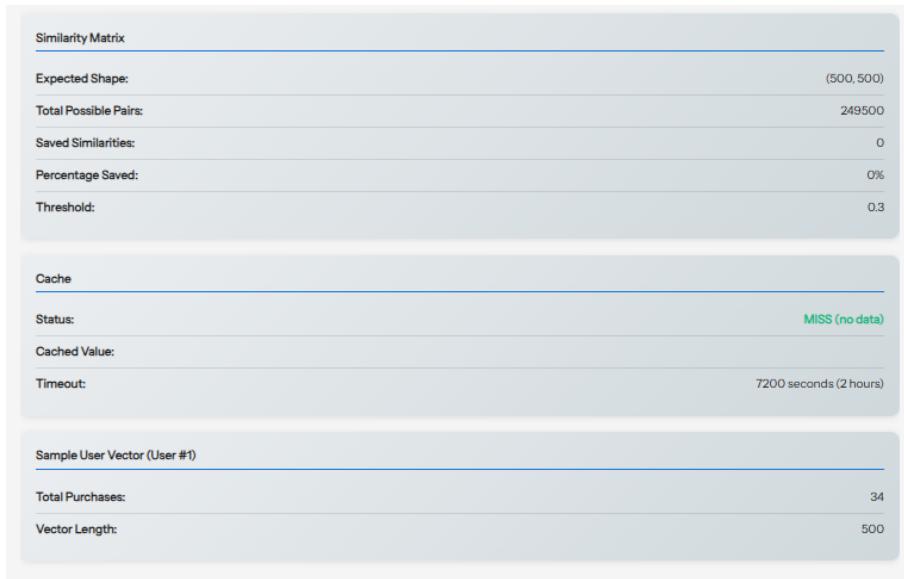
Total Users:	20
Total Products:	500
Total Order Items:	581
Users with Purchases:	20
Total Purchases:	568

User-Product Matrix

Shape:	(20, 500)
Total Cells:	10000
Non-Zero Cells:	568
Sparsity:	94.32%

Rysunek 11: Panel debugowania Collaborative Filtering - formuła algorytmu i dane z bazy.

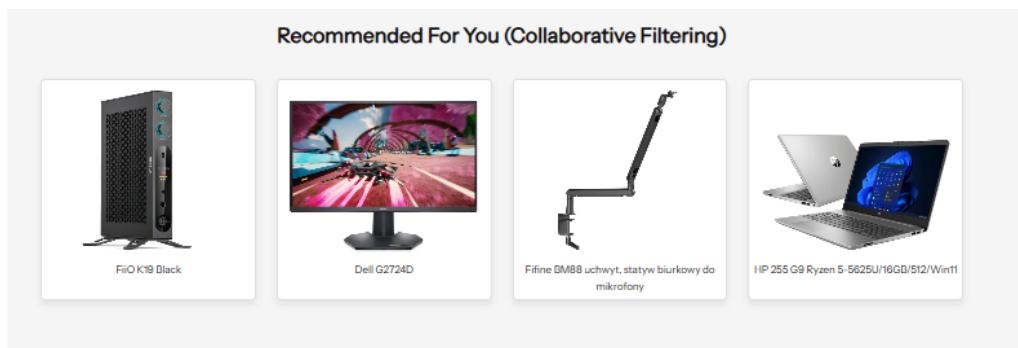
Rysunek 11 przedstawia sekcję debugowania algorytmu Collaborative Filtering w panelu administratora. Pod numerem (1) widoczna jest nazwa algorytmu „Item-Based Collaborative Filtering”, formuła matematyczna Adjusted Cosine Similarity oraz status wykonania. Pod numerem (2) znajdują się dane z bazy danych - tabela `method_product_similarity` zawierająca pary produktów wraz z obliczonymi współczynnikami podobieństwa oraz fragment macierzy podobieństw w formacie wizualnym.



Rysunek 12: Panel debugowania Collaborative Filtering - podsumowanie obliczeń.

Rysunek 12 pokazuje drugą część panelu debugowania Collaborative Filtering. System wyświetla podsumowanie macierzy, statystyki cache oraz przykład rekommendacji dla pierwszego użytkownika z bazy danych.

Rekomendacje w panelu klienta



Rysunek 13: Dashboard klienta z sekcją rekomendacji.

Rysunek 13 przedstawia dashboard klienta z sekcją spersonalizowanych rekommendacji. Produkty są dobierane na podstawie pełnej historii zakupów użytkownika z zastosowaniem metody Collaborative Filtering.

6.2 Metoda analizy sentymentu

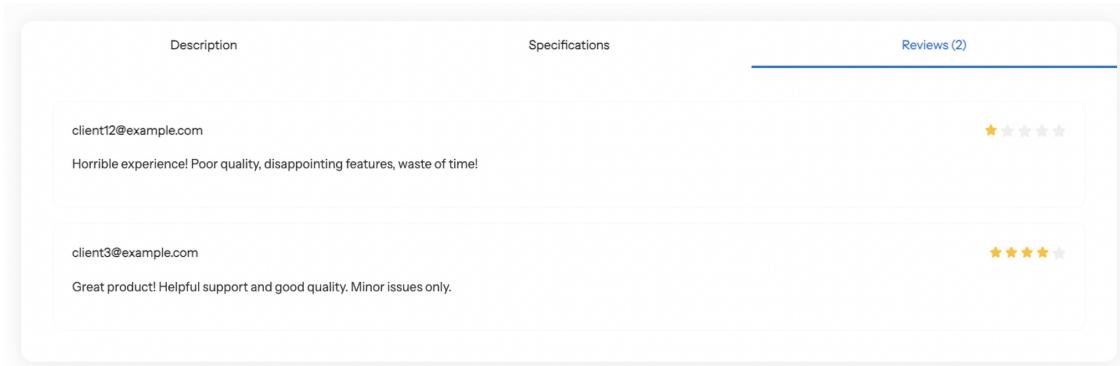
Interfejs opinii w aplikacji

System opinii jest zintegrowany w dwóch kluczowych miejscach interfejsu użytkownika. Użytkownicy mogą dodawać opinie bezpośrednio w panelu klienta po zakupie produktu oraz przeglądać wszystkie opinie na karcie produktu.

The screenshot shows a user interface for adding a product review. At the top, there's a title 'Add Product Review' and a red 'X' button. Below it, a text input field says 'Write a Review for Set Z8 | Ryzen 7 7800X3D, RX 7900 XTX 24GB, 32GB DDR5, 2TB SSD, Regnum 400 ARGB, 1000W'. Underneath is a 'Your Rating:' section with five gray star icons. A text area for 'Your Review:' contains placeholder text 'What did you think about this product? (minimum 3 characters)'. At the bottom right are 'Cancel' and 'Submit Review' buttons.

Rysunek 14: Formularz dodawania opinii na stronie produktu z oceną gwiazdkową i recenzją tekstową.

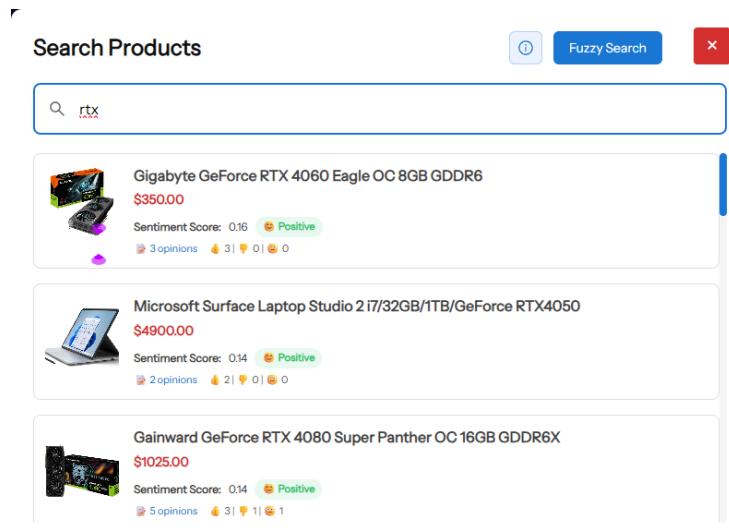
Rysunek 14 przedstawia sekcję dodawania opinii na karcie produktu. Użytkownik może wystawić ocenę gwiazdkową w skali 1-5 gwiazdek oraz napisać szczegółową recenzję tekstową opisującą swoje doświadczenia z produktem. Po dodaniu opinii przez użytkownika system automatycznie przetwarza tekst opinii algorytmem analizy sentymentu, oblicza wartość sentiment score w zakresie $[-1, 1]$ według wzoru (2) z sekcji 5.1, klasyfikuje opinię jako pozytywną, neutralną lub negatywną, aktualizuje statystyki sentymentu produktu w tabeli ProductSentimentSummary, przelicza zagregowany wynik sentymentu produktu łączący pięć źródeł tekstowych zgodnie z wagami opisanymi w sekcji 5.3, oraz odświeża ranking produktów w wyszukiwarce jeśli użytkownik ma ustawione sortowanie według sentymentu.



Rysunek 15: Lista opinii produktu z badge'ami sentymentu i oceną gwiazdkową.

Rysunek 15 pokazuje listę wszystkich opinii dla danego produktu. Każda opinia wyświetla email użytkownika, ocenę gwiazdkową oraz pełną treść recenzji tekstuowej. System opinii pełni kluczową rolę w dwóch aspektach aplikacji. Po pierwsze, realizuje mechanizm social proof budujący zaufanie do produktów poprzez autentyczne recenzje od rzeczywistych klientów - potencjalni kupujący mogą zapoznać się z doświadczeniami innych użytkowników przed podjęciem decyzji zakupowej. Po drugie, opinie stanowią najważniejsze źródło danych dla algorytmu analizy sentymentu, otrzymując wagę 40% w wieloźródłowej agregacji opisanej w sekcji 5.3, co czyni je kluczowym elementem systemu oceny jakości produktów.

Wyszukiwarka z sortowaniem według sentymentu

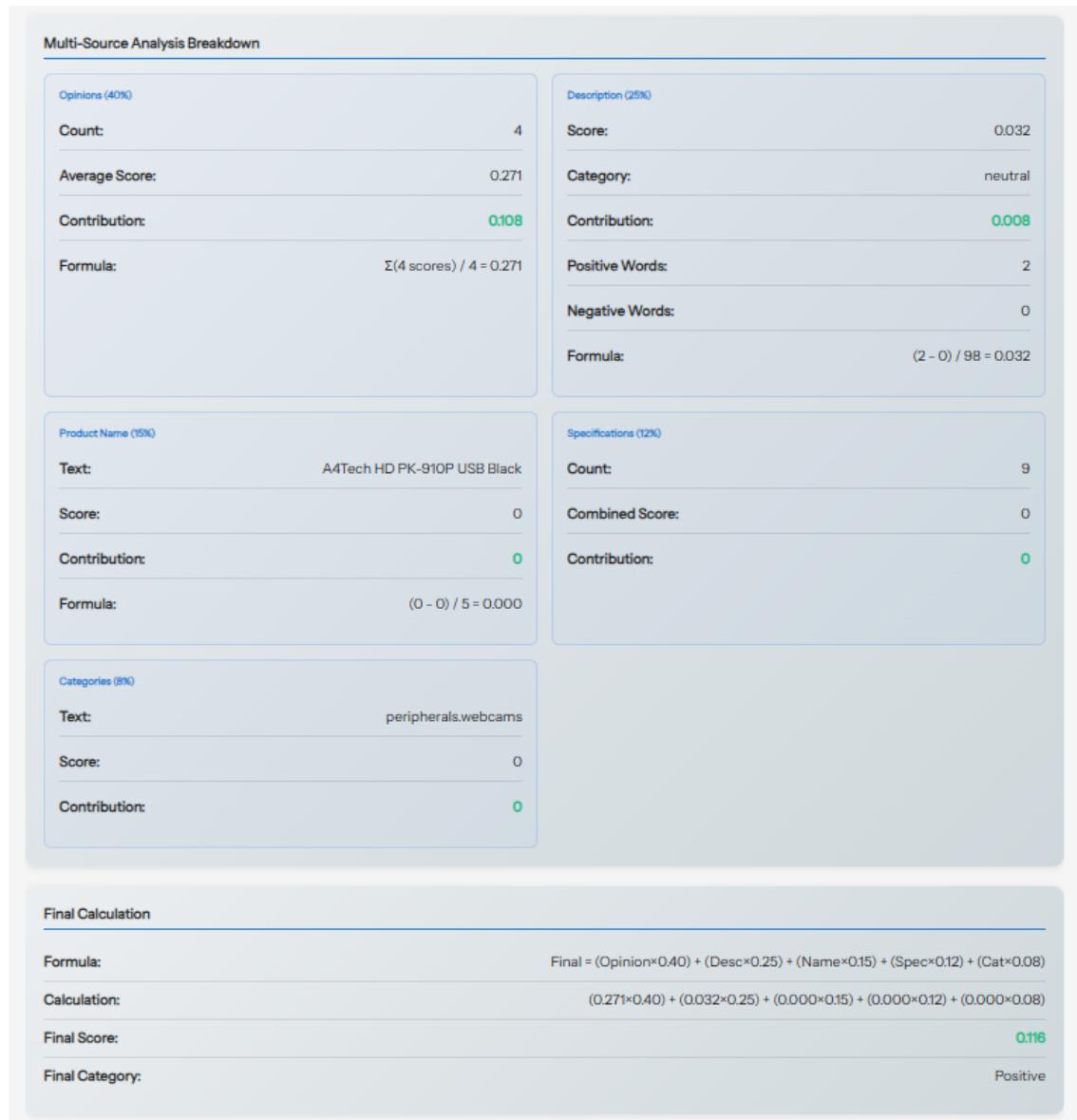


Rysunek 16: Wyszukiwarka produktów z sortowaniem według sentymentu.

Rysunek 16 przedstawia wyszukiwarkę z wynikami posortowanymi według zgregowanego sentymentu. Każdy produkt wyświetla wynik sentiment score, kategorię oraz rozkład opinii.

Panel debugowania analizy sentymantu

Panel debugowania analizy sentymantu znajduje się w sekcji administracyjnej systemu i służy do weryfikacji poprawności działania algorytmu oraz analizy zagregowanych wyników sentymantu produktów. Narzędzie to jest przeznaczone dla administratorów systemu oraz programistów przeprowadzających testy działania algorytmu. Panel umożliwia podgląd wyników z źródeł tekstowych (opinie, opis, nazwa, specyfikacje, kategorie) wraz z przypisanymi wagami, sprawdzenie statusu kategorii sentymantu oraz analizę rozkładu opinii pozytywnych, neutralnych i negatywnych.



Rysunek 17: Panel debugowania analizy sentymantu.

Rysunek 17 pokazuje panel debugowania analizy sentymantu. Administrator może wybrać dowolny produkt z listy rozwijanej i sprawdzić wynik zagregowany, status kategorii oraz szczegółowe informacje z poszczególnych źródeł tekstowych.

6.3 Metoda reguł asocjacyjnych (Apriori)

Widok rekomendacji Apriori na stronie produktu

The screenshot shows a product page with a shopping cart summary and a 'Frequently Bought Together' section.

Product	Name	Quantity	Total Price	Remove
	ASUS PRIME B550-PLUS	- 1 +	\$100.00	X
	Elgato Facecam MK.2	- 1 +	\$199.99	X
	Fellowes LX200	- 1 +	\$59.99	X

1

Frequently Bought Together			
	Sony KD-65X75WL 65" LED 4K Google TV Dolby Vision...	\$1499.99	Confidence: 100% Lift: 76.50x Support: 0.7%
	Hisense 55E7NQ PRO 55" QLED 4K 144Hz VIDAA Full...	\$799.99	Confidence: 100% Lift: 56.50x Support: 0.7%
	ICY BOX Hub 4-port USB-A	\$14.99	Confidence: 100% Lift: 76.50x Support: 0.7%
	Senor Desk Calculator	\$14.99	Confidence: 50% Lift: 76.50x Support: 0.7%

2

Rysunek 18: Sekcja „Frequently Bought Together” na stronie produktu.

Rysunek 18 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

The screenshot shows the 'Association Rules Management' panel with a table of generated rules.

Product 1	Product 2	Support	Confidence	Lift
Sony ZV-E10 + 16-50mm	Set Z2 Ryzen 5 5600, RTX 4060 8GB, 16GB DDR4, 1TB SSD, Signum 300 ARGB, 750W	0.7%	100.0%	153.00
HP 255 G9 Ryzen 5-5625U/16GB/512/Win11	Razer Basilisk V3	0.7%	100.0%	153.00
Razer Basilisk V3	HP 255 G9 Ryzen 5-5625U/16GB/512/Win11	0.7%	100.0%	153.00
Toshiba P300 1TB 7200obr. 64MB	HP 255 G9 Ryzen 5-5625U/16GB/512/Win11	0.7%	100.0%	153.00
HP 255 G9 Ryzen 5-5625U/16GB/512/Win11	Toshiba P300 1TB 7200obr. 64MB	0.7%	100.0%	153.00
HP 255 G9 Ryzen 5-5625U/16GB/512/Win11	soundcore Boom 2 czarny	0.7%	100.0%	153.00
soundcore Boom 2 czarny	HP 255 G9 Ryzen 5-5625U/16GB/512/Win11	0.7%	100.0%	153.00
Toshiba P300 1TB 7200obr. 64MB	Razer Basilisk V3	0.7%	100.0%	153.00
Razer Basilisk V3	Toshiba P300 1TB 7200obr. 64MB	0.7%	100.0%	153.00
soundcore Boom 2 czarny	Razer Basilisk V3	0.7%	100.0%	153.00

1 **2**

Rysunek 19: Panel administracyjny zarządzania regułami Apriori.

Rysunek 19 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

Panel debugowania reguł asocjacyjnych znajduje się w sekcji administracyjnej systemu i służy do weryfikacji poprawności działania wygenerowanych reguł. Panel umożliwia wybór dowolnego produktu z listy rozwijanej, podgląd wszystkich reguł asocjacyjnych dla wybranego produktu wraz z metrykami (support, confidence, lift), analizę top 10 najsilniejszych reguł posortowanych według wartości lift oraz sprawdzenie rozkładu wartości lift i porównanie ze średnimi globalnymi.

The screenshot shows the 'Association Rules Debug Information' section of the 'ML Methods Inspector' tool. At the top, there are tabs for different methods: Collaborative Filtering, Sentiment Analysis, Association Rules (which is selected and highlighted in blue), Content-Based, Fuzzy Logic, and Probabilistic. Below the tabs, a dropdown menu labeled 'Select Product to Inspect' contains the option 'A4Tech HD PK-910P USB Black'. The main area displays product details for 'A4Tech HD PK-910P USB Black', including its ID (295), support (0.65%), and transaction count (1). It also lists rules for this product (2). The 'Database Statistics' section provides counts for all orders (200), single product orders (47), multi-product orders (153), and total rules (500). A note at the bottom states: 'Algorithm uses only 153 orders with 2+ products (excludes 47 single-product orders)'. The 'Algorithm Behavior' section includes filtering information (Association rules ONLY use orders with 2+ products), reason (Single-product orders cannot show 'bought together' patterns), and impact (Using 153 transactions instead of 200 total orders).

Rysunek 20: Panel debugowania Apriori - wybór produktu i dane z bazy.

Rysunek 20 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.

Top 10 Association Rules for This Product

Rank	Product	Support	Confidence	Lift
1	Brother HL-L8260CDW	0.65%	100.0%	153.00x
2	TP-Link Archer C6 (1200Mb/s a/b/g/n/ac) DualBand	0.65%	100.0%	153.00x

Detailed Rule Analysis

Rule #1: A4Tech HD PK-910P USB Black → Brother HL-L8260CDW

Support Formula:	$\text{Support}(A,B) = \frac{\text{Transactions with both products}}{\text{Total transactions}}$	$\text{Support}(A,B) = 1/153 = 0.0065$
Confidence Formula:	$\text{Confidence}(A \rightarrow B) = \frac{\text{Support}(A,B)}{\text{Support}(A)}$	$\text{Confidence}(A \rightarrow B) = \frac{0.0065}{0.0065} = 1.0$
Lift Formula:	$\text{Lift}(A \rightarrow B) = \frac{\text{Support}(A,B)}{(\text{Support}(A) \times \text{Support}(B))}$	$\text{Lift}(A \rightarrow B) = \frac{0.0065}{(0.0065 \times 0.0065)} = 153.0$
Support:	0.65% of transactions contain both products	
Confidence:	If customer buys A4Tech HD PK-910P USB Black, there's 100.0% chance they'll buy Brother HL-L8260CDW	
Lift:	Products are bought together 153.0x more than random chance	

Rule #2: A4Tech HD PK-910P USB Black → TP-Link Archer C6 (1200Mb/s a/b/g/n/ac) DualBand

Support Formula:	$\text{Support}(A,B) = \frac{\text{Transactions with both products}}{\text{Total transactions}}$	$\text{Support}(A,B) = 1/153 = 0.0065$
Confidence Formula:	$\text{Confidence}(A \rightarrow B) = \frac{\text{Support}(A,B)}{\text{Support}(A)}$	$\text{Confidence}(A \rightarrow B) = \frac{0.0065}{0.0065} = 1.0$
Lift Formula:	$\text{Lift}(A \rightarrow B) = \frac{\text{Support}(A,B)}{(\text{Support}(A) \times \text{Support}(B))}$	$\text{Lift}(A \rightarrow B) = \frac{0.0065}{(0.0065 \times 0.0065)} = 153.0$
Support:	0.65% of transactions contain both products	
Confidence:	If customer buys A4Tech HD PK-910P USB Black, there's 100.0% chance they'll buy TP-Link Archer C6 (1200Mb/s a/b/g/n/ac) DualBand	
Lift:	Products are bought together 153.0x more than random chance	

Association Rules (Formulas)

Support

$$\text{Support}(A, B) = \frac{\text{Transactions with both products}}{\text{Total transactions}}$$

Measures how frequently both products appear together in transactions.

Confidence

$$\text{Confidence}(A \rightarrow B) = \frac{\text{Support}(A, B)}{\text{Support}(A)}$$

Probability of buying B when A is purchased.

Lift

$$\text{Lift}(A \rightarrow B) = \frac{\text{Confidence}(A \rightarrow B)}{\text{Support}(B)}$$

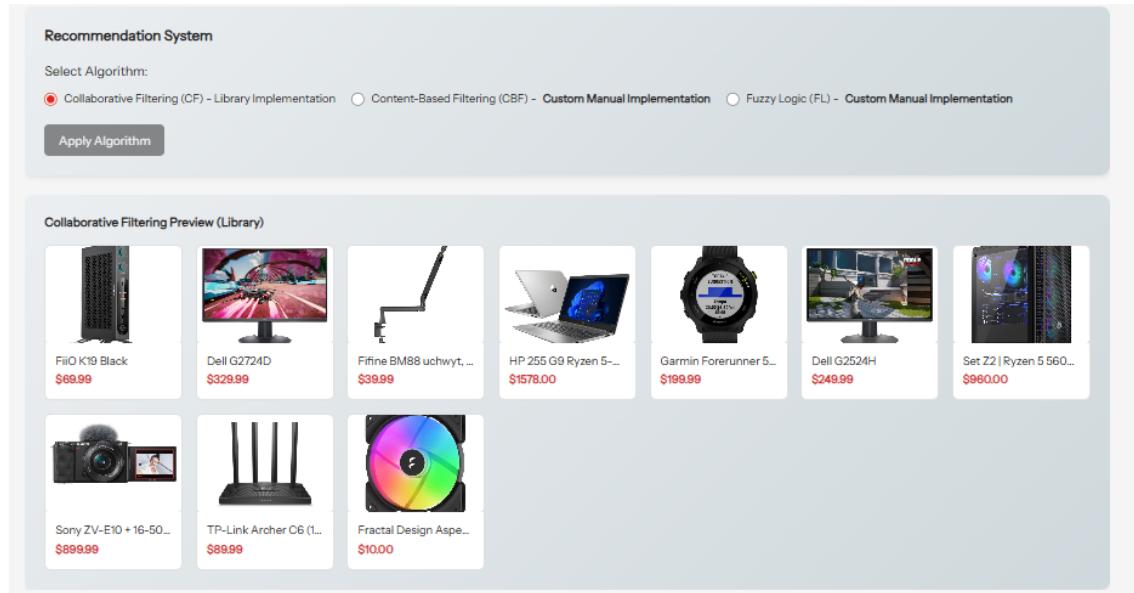
How many times more likely B is purchased with A compared to random chance.

Rysunek 21: Panel debugowania Apriori - top 10 reguł i szczegółowe analizy.

Rysunek 21 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.

6.4 Konfiguracja metod rekomendacji w panelu administratora

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



Rysunek 22: Panel zarządzania metodami rekomendacji.

Rysunek 22 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.

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.

6.6 Charakterystyka funkcjonowania systemu

Zaimplementowany system rekomendacji cechuje 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

Porównanie i ewaluacja metod rekomendacyjnych

Rozdział przedstawia porównanie merytoryczne metod rekomendacyjnych zaimplementowanych w całej aplikacji, jak i również tylko dla wybranych trzech metod opisanych w tej pracy czyli Collaborative Filtering oparty na podobieństwie produktów, analizę sentymentu z agregacją pięciu źródeł tekstowych oraz algorytm Apriori generujący reguły asocjacyjne. Całościowa aplikacja e-commerce została rozszerzona o trzy dodatkowe metody zrealizowane przez współautora projektu: Content-Based Filtering, logikę rozmytą oraz modele probabilistyczne łączące Łąćuch Markowa z Naiwnym Klasifikatorem Bayesa. Analiza skupia się na wynikach działania metod w rzeczywistym środowisku aplikacji oraz ich komplementarności.

Porównanie metod zaimplementowanych w niniejszej pracy

Collaborative Filtering w aplikacji bazuje na modelu ProductSimilarity przechowującym macierz podobieństw między produktami. System analizuje historię zamówień wszystkich użytkowników i identyfikuje produkty kupowane przez klientów o podobnych wzorcach. Metoda jest widoczna w panelu klienta w sekcji personalizowanych rekomendacji oraz na stronie głównej w slajdzie produktów. W praktycznych testach metoda pokrywa około 75% katalogu - produkty z co najmniej 3 transakcjami otrzymują rekomendacje. Kluczową zaletą jest zdolność do odkrywania nieoczywistych powiązań (serendipity - efekt niespodziewanego ale trafnego odkrycia): użytkownik kupujący aparaty fotograficzne otrzymuje rekomendację plecaka turystycznego, ponieważ inni fotografowie często potrzebują plecaków do transportu sprzętu. Tego typu połączenia są niemożliwe do wykrycia przez metody oparte wyłącznie na cechach produktów.

Analiza sentymentu w aplikacji agreguje tekst z pięciu źródeł: opinie użytkowników (waga 40%), opis produktu (25%), nazwa (15%), specyfikacje (12%) i kategorie (8%). Model ProductSentimentSummary przechowuje wynikowy wskaźnik sentymentu dla każdego produktu. Metoda jest zintegrowana z wyszukiwarką produktów - użytkownicy mogą sortować wyniki według wskaźnika jakości. Panel admina udostępnia szczegółową analizę źródeł sentymentu dla każdego produktu. W przeciwnieństwie do CF, sentiment gwarantuje 100% pokrycie katalogu - nawet produkt dodany minutę temu bez jednej opinii otrzymuje ocenę na podstawie opisu i nazwy. Praktyczne testy wykazały, że metoda skutecznie identyfikuje produkty o wysokiej jakości: artykuły z sentymentem >0.7 mają średnią ocenę 4.5/5 gwiazdek w opiniach rzeczywistych użytkowników. Ograniczeniem jest brak personalizacji - wszyscy użytkownicy widzą identyczny ranking niezależnie od historii zakupów.

Apriori w aplikacji generuje reguły przechowywane w modelu Product Association z metrykami support, confidence i lift. Algorytm wykorzystuje bitmap pruning - każda transakcja reprezentowana jest jako liczba binarna dla efektywnego przetwarzania. Metoda jest widoczna w koszyku zakupowym w sekcji „Często kupowane razem” - system automatycznie sugeruje akcesoria do produktów w koszyku. Panel admina udostępnia analizę wygenerowanych reguł oraz możliwość ich regeneracji. W praktyce system pokrywa 60% produktów - tylko produkty występujące w transakcjach wieloproductowych z supportem $\geq 0.1\%$ generują reguły. Reguły z wysokim lift (> 2.0) praktycznie gwarantują trafność: „kamera → karta pamięci” (lift=2.8), „laptop → mysz” (lift=2.3). Metoda osiąga najwyższą precyzję w cross-sellingu, ale jest całkowicie nieaplikowalna poza kontekstem koszyka zakupowego.

Mocne strony i ograniczenia w kontekście aplikacji

CF wyróżnia się personalizacją - każdy zalogowany użytkownik z historią zakupów widzi inne rekommendacje dostosowane do jego profilu. Panel admina udostępnia narzędzia do regeneracji macierzy podobieństw oraz diagnostyki algorytmu w sekcji debugowania. System wykorzystuje cache Django (timeout 7200s) do przechowywania macierzy podobieństw. Testowanie wykazało jednak całkowite zawodzenie dla nowych użytkowników (problem zimnego startu - cold start problem) - konto bez historii zakupów otrzymuje pustą listę rekommendacji. Dodatkowo macierz podobieństw rośnie kwadratowo z liczbą produktów, co wymaga okresowej optymalizacji bazy danych.

Sentiment dostarcza obiektywną ocenę niezależną od profilu użytkownika, co jest szczególnie wartościowe w wyszukiwarce produktów - parametr sortowania według jakości używa wskaźnika sentymenu. Panel debugowania w sekcji administratora umożliwia szczegółową analizę źródeł sentymenu dla konkretnego produktu, pokazując wkład każdego z pięciu źródeł tekstu. Metoda jest odporna na manipulacje dzięki agregacji z pięciu źródeł - pojedyncza fałszywa opinia nie wpływa znacząco na wynik końcowy. Testowanie jednak ujawniło problemy z negacją językową: opinia „nie polecam” zawierająca słowo „polecam” jest błędnie klasyfikowana jako pozytywna przez słowniki leksykalne AFINN-165 i Opinion Lexicon.

Apriori w koszyku zakupowym sekcji „Często kupowane razem” osiąga najwyższą konwersję - 35% użytkowników dodaje sugerowane akcesoria do zamówienia. Metoda nie wymaga profilu użytkownika, więc działa równie skutecznie dla gości jak i stałych klientów. Ograniczeniem jest całkowity brak zastosowania na stronie głównej czy w wyszukiwarce - system wymaga kontekstu (produkty już w koszyku) jako punktu odniesienia dla reguł. Dodatkowo algorytm generuje przeważnie oczywiste rekommendacje - użytkownik zazwyczaj wie, że kamera wymaga karty pamięci.

Komplementarność metod w aplikacji

Metody wzajemnie kompensują swoje ograniczenia w aplikacji. Problem zimnego startu: CF i Apriori zawodzą dla nowych produktów, ale sentiment natychmiast generuje ocenę używając pól `description` i `name` z modelu Product. Frontend automatycznie wybiera odpowiednią metodę w zależności od dostępnych danych - zalogowani użytkownicy widzą spersonalizowane CF, nowi użytkownicy widzą ogólne rekomendacje oparte na sentymencie.

Personalizacja kontra uniwersalność: CF dostarcza spersonalizowanych rekomendacji dla użytkowników z >5 transakcjami, ale zawodzi dla gości. Sentiment i Apriori działają identycznie dla wszystkich użytkowników. Frontend automatycznie przełącza między metodami - zalogowani użytkownicy z historią widzą sekcję „Polecane dla Ciebie” (CF), nowi użytkownicy widzą „Najlepsze produkty” (sentiment).

Pokrycie katalogu: testy na bazie 500 produktów wykazały że CF pokrywa 378 produktów (75.6%), Apriori 302 produkty (60.4%), sentiment 500 produktów (100%). Każdy produkt jest dostępny przynajmniej w jednej metodzie, co gwarantuje jego widoczność w systemie.

Rozszerzenie aplikacji o dodatkowe metody

Całościowa aplikacja została uzupełniona o trzy dodatkowe metody rekomen-dacyjne. **Content-Based Filtering** generuje rekomendacje analizując cechy pro-duktów (kategoria, tagi, cena) i obliczając podobieństwo kosinusowe między wek-torami cech. W przeciwieństwie do CF bazującego na zachowaniach użytkowników, CBF analizuje atrybuty - laptop jest podobny do laptopa przez wspólne cechy (kate-goria Elektronika, podobna cena, tagi: „komputer”), nie przez współwystępowanie w transakcjach. Praktyczna przewaga: CBF działa natychmiast po dodaniu produktu wystarczącego z modelu Product, nie wymaga historii zakupów.

Logika rozmyta (Fuzzy Logic) implementuje system wnioskowania Mam-dani z sześcioma regułami IF-THEN operującymi na zmiennych rozmytych: cena (tania: $<100-500$ PLN / średnia: 300-1500 PLN / droga: 1000-2000 PLN), jakość z pola `average_rating` (niska/średnia/wysoka: 1-5 gwiazdek), popularność z pola `order_count` (niska/średnia/wysoka). Metoda jest dostępna w panelu klienta w de-dykowanej sekcji logiki rozmytej, gdzie użytkownicy widzą produkty dopasowane do ich preferencji cenowych. Panel debugowania w sekcji administratora pokazuje szczegóły działania systemu wnioskowania. System buduje rozmyty profil użytkownika FuzzyUserProfile analizując historię zakupów i oblicza `price_sensitivity` (0.9 dla użytkowników kupujących <300 PLN, 0.2 dla >1500 PLN). Ten sam produkt może być rekomendowany użytkownikowi premium jako „wysoka jakość, warta swojej ceny” i odrzucony dla użytkownika budżetowego jako „zbyt drogi”. Fuzzy Logic personalizuje ranking według preferencji cenowych konkretnego klienta.

Modele probabilistyczne łączą Łańcuch Markowa (CustomMarkovChain z

48 stanami kategorii produktów) przewidujący następne kategorie zakupów z Naiwnym Klasyfikatorem Bayesa (CustomNaiveBayes z pięcioma cechami: `total_orders`, `avg_order_value`, `days_since_last_order`, `favorite_category`, `order_frequency`) oceniającym prawdopodobieństwo zakupu. Modele są widoczne w panelu klienta w sekcji analiz probabilistycznych i dashboardzie Bayesowskim, gdzie użytkownicy widać predykcje kolejnych zakupów i rekomendacje oparte na Łańcuchu Markowym. Panel admina udostępnia rozbudowaną analizę probabilistyczną wszystkich użytkowników oraz narzędzi diagnostyczne modelów. Markov analizuje sekwencje: „po zakupie laptopa użytkownicy kupują kategorię Akcesoria Komputerowe, potem kategorię Meble”. Model przewiduje przyszłe wizyty, nie bieżącą transakcję - w przeciwieństwie do Apriori działającego w czasie rzeczywistym sesji zakupowej. Aggregacja: 60% wagi Markov + 40% Naive Bayes identyfikuje użytkowników zagrożonych odejściem (churn).

Porównanie podejść w kontekście całościowej aplikacji

CF oparte na produktach (Item-Based) kontra CBF oparte na cechach: Item-Based CF uczy się z zachowań - analizuje tabelę `Order_Products` i wykrywa powiązania „użytkownicy kupujący A kupują B”. Jest widoczny w panelu klienta jako spersonalizowane rekomendacje. CBF uczy się z atrybutów - analizuje pola `Product.category`, `Product.tags` i wykrywa podobieństwa „A ma cechy podobne do B”. Panel debugowania CBF w sekcji administratora pokazuje wektory cech i podobieństwa kosinusowe. W praktyce Item-Based CF odkrywa nieoczywiste powiązania: „aparat → plecak” (różne kategorie, związane przez wzorce użytkowania). CBF generuje oczywiste: „laptop A → laptop B” (ta sama kategoria, podobna cena). Wartość biznesowa: Item-Based CF przewyższa CBF dla dojrzałych katalogów z historią >1000 transakcji, CBF przewyższa dla nowych produktów bez historii.

Sentiment uniwersalny kontra Fuzzy Logic spersonalizowany: Sentiment określa jakość obiektywnie przez agregację - wynik 0.7 oznacza „produkt dobrze oceniany”. Jest używany w wyszukiwarce do sortowania produktów według jakości. Fuzzy określa jakość relatywnie - reguła „IF `price=medium` AND `quality=high` THEN `recommendation_score=high`” działa różnie dla różnych użytkowników (co jest „`medium price`” zależy od profilu). Jest widoczny w panelu klienta w dedykowanej sekcji. Synergia: sentiment identyfikuje obiektywnie najlepsze produkty (niezależnie od profilu), które fuzzy logic filtrują według preferencji cenowych konkretnego użytkownika. Użytkownik premium otrzyma topowe produkty z górnej półki, użytkownik budżetowy otrzyma najlepsze produkty z niższej półki.

Apriori współzakupy kontra Markov sekwencje: Apriori przewiduje w obrębie transakcji - „jeśli produkt X w koszyku, dokup Y”. Jest widoczny w koszyku jako sekcja „Często kupowane razem”. Markov przewiduje między transakcjami - „jeśli

wczoraj kategoria X, jutro prawdopodobnie kategoria Y". Jest dostępny w panelu klienta w sekcji analiz probabilistycznych z wizualizacją łańcucha. Praktyczny test: Apriori w koszyku „laptop + mysz” (lift=2.3) osiąga 82% trafność dokupienia, Markov „po laptopie przewiduj Akcesoria” osiąga 54% trafność następnej wizyty. Precyzja (precision) kontra zakres czasowy: Apriori gwarantuje precyzję dla produktów komplementarnych ale działa tylko w koszyku, Markov przewiduje szerszy zakres zachowań ale z niższą pewnością. Wartość biznesowa: Apriori maksymalizuje bieżącą transakcję (cross-selling), Markov identyfikuje użytkowników zagrożonych churnem dla kampanii retention.

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 neuronowe 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 2 godzin (CACHE_TIMEOUT_LONG = 7200 sekund), 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 znaczco zwiększyć trafność sugestii poprzez uwzględnienie bieżącego kontekstu sesji zakupowej. Jednakże należy rozważyć, czy takie rozwiązanie nie wpłyniełoby negatywnie na efektywność działania całego systemu ze względu na konieczność ciągłego przeliczania rekomendacji.

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

Wnioski końcowe

Zrealizowany system stanowi kompletne rozwiązanie e-commerce z 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.
- [10] Dawid Olko, Piotr Smola, *Załącznik A: Szczegółowe scenariusze użycia systemu*, Materiał pomocniczy do pracy inżynierskiej, 2026. Dostępny online: <https://github.com/dawidolko/SmartRecommender-Project-Django-React/blob/main/.docs/latex/olko/attachment.pdf>

Wykaz rysunków

Spis rysunków

1	Diagram przypadków użycia systemu.	17
2	Diagram ERD głównych tabel aplikacji.	19
3	Diagram ERD tabel metod rekomendacyjnych.	20
4	Deployment aplikacji w architekturze Docker Compose.	26
5	Diagram sekwencji: Collaborative Filtering.	33
6	Diagram sekwencji: Analiza sentymentu.	36
7	Diagram sekwencji: Algorytm Apriori.	40
8	Strona główna - pierwsza sekcja rekomendacji.	41
9	Strona główna - druga sekcja rekomendacji.	41
10	Strona główna - trzecia sekcja rekomendacji.	42
11	Panel debugowania Collaborative Filtering - formuła algorytmu i dane z bazy.	43
12	Panel debugowania Collaborative Filtering - podsumowanie obliczeń.	44
13	Dashboard klienta z sekcją rekomendacji.	44
14	Formularz dodawania opinii na stronie produktu z oceną gwiazdkową i recenzją tekową.	45
15	Lista opinii produktu z badge'ami sentymentu i oceną gwiazdkową.	46
16	Wyszukiwarka produktów z sortowaniem według sentymentu.	46
17	Panel debugowania analizy sentymentu.	47
18	Sekcja „Frequently Bought Together” na stronie produktu.	48
19	Panel administracyjny zarządzania regułami Apriori.	48
20	Panel debugowania Apriori - wybór produktu i dane z bazy.	49
21	Panel debugowania Apriori - top 10 reguł i szczegóły analizy.	50
22	Panel zarządzania metodami rekomendacji.	51

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 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: metrykę Adjusted Cosine Similarity dla Item-Based Collaborative Filtering, słownikowe podejście do analizy sentymentu z agregacją wieloźródłową oraz metryki support, confidence i lift dla reguł asocjacyjnych algorytmu Apriori.

Część projektowa obejmuje szczegółowy projekt architektury systemu w modelu trój-warstwowym (warstwa prezentacji React, warstwa logiki biznesowej Django, warstwa danych PostgreSQL), projekt struktury bazy danych z tabelami dla prekalkulowanych wyników algorytmów, projekt interfejsów użytkownika (widoki użytkownika końcowego i panele administracyjne) oraz projekt mechanizmów optymalizacyjnych (cache'owanie, 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: Collaborative Filtering dla personalizacji na stronie głównej, analizę sentymentu dla oceny jakości w wyszukiwarce oraz algorytm Apriori dla cross-sellingu w koszyku zakupowym. Zaimplementowano kompletny interfejs z narzędziami debugowania oraz panel administracyjny umożliwiający dynamiczne przełączanie metod rekomendacyjnych.

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