МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВЕТЕРИНАРНОЇ МЕДИЦИНИ ТА БІОТЕХНОЛОГІЙ ІМ. С.З. ҐЖИЦЬКОГО
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

першого (бакалаврського) рівня вищої освіти

на тему: **«Розробка системи рекомендацій на основі даних користувачів для сезонного підбору одягу»**

Виконав: студент 4 курсу групи Іт-41

Спеціальності 126 «Інформаційні системи та технології»
(шифр і назва)

Кухар Ростислав Ярославович
(Прізвище та ініціали)

Керівник: к.т.н., доцент Падюка Р.І.
(Прізвище та ініціали)

ДУБЛЯНИ-2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА
ІНФОРМАЦІЙНИХ ТЕХНЕОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Перший (бакалаврський) рівень вищої освіти
Спеціальність 126 «Інформаційні системи та технології»

«ЗАТВЕРДЖУЮ»

Завідувач кафедри_____

д.т.н., проф. А. М. Тригуба
«____» _____2025 р.

ЗАВДАННЯ

на кваліфікаційну роботу студенту

Кухару Ростиславу Ярославовичу

1. Тема роботи: «Розробка системи рекомендацій на основі даних користувачів для сезонного підбору одягу»

Керівник роботи Падюка Роман Іванович, доцент
затверджені наказом по університету від 25.02.2025 року № 123/к-с.

2. Строк подання студентом роботи 10.06.2025 р.
3. Вихідні дані до роботи: *вимоги до проектування інформаційних систем; методика проектування інформаційних систем; моніторингу функціонування комп'ютерної мережі закладу освіти.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)
Вступ.
1. Аналіз предметної області.
2.Постановка задачі
3.Проектування системи рекомендацій для сезонного підбору одягу
4. Охорона праці та безпека в надзвичайних ситуаціях
Висновки та пропозиції.
Список використаної літератури.
5. Перелік ілюстраційного матеріалу (з точним зазначенням обов'язкових креслень): *Актуальність теми, мета роботи, аналіз предметної області, Методи надання рекомендацій, використані технології, моделювання системи рекомендацій, архітектура та реалізація, проблеми та можливі рішення.*

6. Консультанти з розділів:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3	*Падюка Р.І., доцент кафедри ІТ*		
4	*Городецький І.М., доцент кафедри інженерної механіки*		

7. Дата видачі завдання 26 лютого 2025 р.

Календарний план

№ з/п	Назва етапів дипломного проекту	Терміни виконання етапів роботи	Примітка
1	*Написання першого розділу*	*26.02.25-20.03.25*	
2	*Виконання другого розділу та аркушів ілюстраційного матеріалу до нього*	*21.03-15.04.25*	
3.	*Виконання третього розділу та аркушів ілюстраційного матеріалу до нього*	*16.04-30.04.25*	
4.	*Написання розділу «Охорона праці»*	*01-15.05.25*	
5.	*Завершення оформлення розрахунково-пояснювальної записки та аркушів ілюстраційного матеріалу*	*15-30.05.25*	
6.	*Завершення роботи вцілому*	*01 - 10.06.25*	

Студент _____ Кухар Р.Я.
(підпис)

Керівник роботи _____Падюка Р.І.
(підпис)

УДК 004.8:658.8:687

Розробка системи рекомендацій на основі даних користувачів для сезонного підбору одягу. Кухар Р.Я. Кафедра ІТ – Дубляни, Львівський НУВМБ ім. С.З. Гжицького, 2025.

Кваліфікаційна робота: 63 с. текст. част., 7 рис., 5 табл., 10 арк. ілюстраційного матеріалу, 12 джерел.

Здійснено комплексний аналіз систем рекомендацій, підкреслено їх актуальність і вплив на бізнес і залучення користувачів. Означено методи формування рейтингів інтересів користувачів, висвітлюючи явні та неявні фактори, що впливають на переваги. Розглянуто різні існуючі рекомендаційні алгоритми, включаючи спільну фільтрацію, контентні та гібридні підходи, з детальним описом методів на основі пам’яті та моделей, таких як кластеризація та матрична факторізація. Означено загальні проблеми, такі як розрідженість даних, холодний запуск, масштабованість і надмірна спеціалізація, пропонуючи такі рішення, як профілювання користувачів і продуктів, моделі машинного навчання та різноманітність рекомендацій.

Розроблено персоналізовану систему рекомендацій щодо одягу з використанням Python, деталізовано модулі обробки даних, евристична фільтрація кандидатів і архітектура системи, розроблена для оптимізації якості та ефективності рекомендацій.

Ключові слова: рекомендаційна система, одяг, користувачі, фільтрація, машинне навчання

Key words: recommendation system, clothing, users, filtering, machine learning.

ЗМІСТ

ВСТУП

Ми живемо в еру великих даних, тому не дивно, що сфера інтелектуального аналізу даних і машинного навчання стрімко розвивається. Стрімкий розвиток цієї галузі дозволяє елегантно вирішувати велику кількість прикладних задач, які донедавна здавалися нерозв'язними або важко вирішуваними на практиці.

Це створює як великі можливості, так і деякі загрози для компаній. Адже за досить короткий термін, використовуючи сучасні технології, можна суттєво вдосконалити свій продукт і зробити його більш конкурентоспроможним. Однак це працює лише для компаній, які можуть швидко адаптуватися до нових реалій сьогодення. Інертність і консервативність рідко виживають у конкуренції. Тому ми спостерігаємо великий інтерес до цієї сфери серед сучасних успішних компаній. Це виражається у фінансуванні досліджень, масштабній підготовці експертів, організації конференцій, семінарів і відкритих конкурсів з аналізу даних.

На мій погляд, особливо цікавих змін зазнає сфера маркетингу. Зміни відображаються не лише на технічному, а й на концептуальному та теоретичному рівні. Це пояснюється тим, що розвиток галузі аналізу даних також сприяв популяризації таких дисциплін, як прикладна математика та статистика. У результаті маркетингові методи з хорошою теоретичною основою тепер можуть швидко розвиватися. У свою чергу компанії різними способами сприяли цьому, адже добре відомо, що хороший маркетинг є важливою складовою успішних продуктів.

Метою цієї роботи є створення персоналізованої системи рекомендацій, яка здатна використовувати та аналізувати дані користувачів і клієнтів і рекомендувати їм конкретні продукти на основі цього. Щоб створити цю систему ми будемо використовувати машинне навчання та інтелектуальні методи аналізу даних.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Аналіз актуальності та важливості задачі рекомендацій

Завдання системи персоналізованих рекомендацій полягає в тому, щоб вибрати з набору продуктів ті предмети, які, швидше за все, будуть використані користувачем для виконання дії, що цікавить бізнес. Приклади продуктів і дій:

□ Продукти або пропозиції, які користувач хоче придбати;

□ Відео або фільми, які користувач хоче переглянути;

□ Статті чи книги, які користувач прочитає від початку до кінця;

□ Музика чи подкасти, які сподобаються користувачеві;

□ Контент соціальної мережі, який цікавить користувача;

Бізнес-моделі систем рекомендацій також різноманітні. Системи рекомендацій можуть бути основою продукту (наприклад, Tripadvisor), або їх можна використовувати для покращення взаємодії з користувачем (наприклад, Netflix, Spotify), або для додаткового просування продукту (наприклад, Amazon). Останній пункт є особливою важливим сьогодні з огляду на велику популярність хмарних технологій. Велика кількість наукових публікацій вивчала вплив систем рекомендацій на різні категорії продуктів. Ці дослідження показують дуже цікаву статистику. Наприклад, ще в 2010 році команді YouTube вдалося підвищити рейтинг кліків нішевих відео на 200% шляхом модернізації системи рекомендацій[2]. Нещодавнє дослідження краудсорсингової платформи Amazon Mechanical Turk показало, що впровадження систем рекомендацій підвищило кінцевий коефіцієнт конверсії користувачів на 20% [3]. Дослідження McKinsey показало, що 35% покупок Amazon пов’язані з рекомендованими продуктами[4].

Тому бізнес-вигоди систем рекомендацій очевидні. У свою чергу, користувачі не тільки люблять персоналізовані рекомендації, вони навіть вимагають їх. Недавній звіт Smart Insights показує, що 80% користувачів онлайн-магазину сказали, що вони не будуть використовувати платформу, яка не надає жодних персоналізованих послуг[5].

Можна помітити, що відсутність персоналізованих послуг у сучасних продуктах не тільки призводить до втрати додаткового прибутку, але також може призвести до відтоку користувачів. Тому для багатьох сучасних програмних продуктів наявність ефективного персоналізованого алгоритму рекомендацій є дуже важливою та актуальною проблемою.

1.2. Формування рейтингів зацікавленості користувачів

Як встановлювати рейтинги, це незалежна тема дослідження, але зазвичай рейтинги створюються на основі відгуків користувачів. Наприклад, для рекомендацій музики, подкастів, книг або відео рейтингом може бути кількість і/або тривалість взаємодії з відповідним типом продуктів. У соціальних мережах також можна використовувати спеціальні засоби взаємодії з контентом (лайки, репости, коментарі тощо). Для інтернет-магазинів інтерес зазвичай включає покупки та перегляд товарів. Крім того, ваги покупок мають бути вищими, ніж для перегляду.

Фактори, що впливають на рейтинги, можна умовно розділити на дві категорії: явні фактори та неявні фактори. Явні фактори – це зазвичай ті, які інтерпретуються та чітко відображають емоції та інтереси користувача. Наприклад, лайки і репости в соціальних мережах, або покупки в інтернет-магазинах. Неявними факторами можуть бути будь-які відгуки, інтерпретація яких вимагає певних припущень щодо поведінки користувача. Наприклад, час взаємодії з контентом або перегляду товару в магазині може не відповідати дійсному інтересу користувача до продукту: можливо, у користувача просто поганий настрій і він не підходить для перегляду певного типу контенту, або товар з’явився випадково, але користувачеві це нецікаво. Однак цей зворотній зв’язок також можна використати, зробивши певні припущення щодо моделі поведінки користувача (користувач завжди взаємодіє з цікавим вмістом; переглядає лише цікаві продукти). Це особливо корисно у випадках, коли є дуже обмежена інформація про явні фактори взаємодії. Зазвичай різні фактори

оцінюються евристично та включаються до остаточної оцінки процентного рейтингу у вигляді зваженої суми. Крім того, вага неявних факторів значно нижча, ніж явних, щоб уникнути того, що помилки, які неминуче виникають під час прийняття припущень, домінують у формуванні остаточного рішення щодо рекомендації конкретного продукту.

1.3. Огляд існуючих на ринку систем рекомендацій.

Навіть найпримітивніші інтернет-магазини можуть використовувати алгоритми рекомендацій, але чим конкретніші дані, чим чіткіші критерії рекомендацій, а бажання надати максимально точні рекомендації буде значно ускладнювати цю систему рекомендацій. Великі компанії розширюють існуючі алгоритми або створюють власні алгоритми системи рекомендацій. Наприклад, такі компанії, як Netflix, Amazon, Google, LinkedIn, Facebook, Instagram, Twitter тощо.

Система рекомендацій GroupLens.

GroupLens, створена дослідницькою лабораторією GroupLens, є одним із піонерів систем рекомендацій. Система орієнтована на рекомендації новин. GroupLens збирає рейтинги від читачів Usenet і використовує ці рейтинги, щоб передбачити, чи сподобається стаття іншим читачам, перш ніж прочитати її. Деякі з найперших алгоритмів спільної фільтрації були розроблені в рамках системи GroupLens. Загальні ідеї, запропоновані дослідницькою групою, пізніше були застосовані до інших типів контенту, таких як книги (BookLens) і фільми (MovieLens) [2]. Окрім досліджень у сфері рекомендаційних систем, дослідницька група також опублікувала кілька наборів даних, у тому числі 3 набори даних із системи MovieLens.

Система рекомендацій Amazon

Система рекомендацій Amazon також є одним із піонерів у сфері рекомендацій, зосереджуючись на комерційному секторі. Бізнес електронних книг усвідомив переваги використання рекомендацій і розвивався з часом, і

сьогодні майже всі типи існуючих продуктів можна знайти на веб-сайті Amazon. Рекомендації тут базуються на явних оцінках продукту, наданих користувачами, а також на неявному зборі даних (поведінка користувачів при покупці та відгуки користувачів). Оцінки користувачів надаються за шкалою від 1 до 5. Відгуки користувачів та інформація про оцінки збираються, коли користувач входить до системи за допомогою механізму автентифікації.

Система рекомендацій Facebook. У соціальних мережах люди часто рекомендують потенційних друзів або контент, який може зацікавити. Мета таких рекомендацій дещо відрізняється від рекомендацій продукту. Якщо рекомендації продуктів і продажі допомагають компаніям збільшити прибутки, рекомендація друзів і вмісту можуть покращити досвід користувачів у певній мережі, по-перше, утримуючи користувачів у мережі довше, а по-друге, залучаючи до мережі більше користувачів. Чим більше людей переглядає мережу, тим вище дохід від реклами.

Такі рекомендації базуються не на рейтингах, а на зв’язках між користувачами, між користувачами та групами тощо. Тому ці системи використовують інші алгоритми, засновані на обробці з’єднань.

Система рекомендацій Spotify. Щоб краще зрозуміти, як насправді працюють алгоритми та їх комбінації, розглянемо приклад одного з найпопулярніших додатків сучасності – потокового сервісу Spotify.

Spotify було засновано в 2006 році і спочатку планувалося використовувати як музичну бібліотеку, але в міру розвитку платформи розробники зрозуміли, що персоналізація матиме позитивний вплив на додаток. Тому до системи почали застосовувати алгоритми для аналізу вподобань користувачів і надання рекомендацій на основі прослуханих треків. З часом алгоритми вдосконалювалися, а рекомендації, які надавала система, ставали все кращими.

Для надання рекомендацій система використовує внутрішні та зовнішні дані:

• Внутрішні дані включають дані композиції (виконавець, зображення,

назва, опис, жанр, тексти пісень, файл композиції) і дані користувача (історія прослуховування пісень, тривалість прослуховування, кількість повторних прослуховувань, пропущених треків, збережених пісень, створених списків відтворення, взаємодії з іншими користувачами).

• Зовнішні дані - аналіз статей, публікацій в блогах та інших даних в Інтернеті, пов'язаних з треками, жанрами та виконавцями.

Загалом дані, які використовує Spotify у своєму алгоритмі, можна розділити на три категорії:

• Аналіз поточної поведінки користувача - збережені списки відтворення, пропущені треки, улюблені жанри, треки та виконавці, спільні списки відтворення, частота прослуховування певного списку відтворення чи пісні та залежність користувача від прослуховування вмісту в певний час доби.

• Аналіз інших користувачів (користувачів і виконавців) - списки відтворення, які сподобалися іншим користувачам, поточні тенденції популярності, що люблять слухати користувачі, схожі на поточного користувача, нові треки виконавців, які сподобалися поточному користувачеві.

• Аналіз треку - аналіз спектрограми, тривалості, гучності, темпу, такту та інших характеристик треку.

Spotify використовує систему BaRT («Бандитський алгоритм для рекомендацій») для рекомендацій. Система має два режими: експлуатація та розвідка[3].

Оперативний означає режим, у якому система використовує зібрану інформацію про користувачів. Це стандартний режим, у якому зазвичай працює система. На основі спільної фільтрації. Цей режим ідеальний, якщо в системі достатньо інформації про користувача. Але однією проблемою, з якою стикаються такі системи, є відсутність даних про користувачів або елементи. Наприклад, в системі з'явився новий користувач, або трек ще не прослуховувався або прослуховувався нечасто. У цьому випадку система може не надавати відповідні рекомендації. Режим дослідження може уникнути таких проблем.

Режим дослідження — це коли система рекомендує контент, для якого

недостатньо даних, щоб судити, чи сподобається трек користувачеві. Вміст рекомендовано для збору додаткової інформації. Перемикання в цей режим гарантує, що навіть якщо композицію щойно додано, вона все ще матиме шанс бути рекомендованою. Якщо користувач слухає рекомендований трек більше 30 секунд, рекомендація вважається успішною.

Завдання системи BaRT – відібрати максимально актуальні рекомендації. Система вчиться на своїх помилках і продовжує вдосконалюватися з кожним відгуком, який отримує. Однак команда Spotify не зупинилася на досягнутому й додала контекст до алгоритму. Тепер завданням алгоритму є не лише надання рекомендацій, але й врахування інших факторів для рекомендації треків (наприклад, на основі дня тижня, нових релізів, попередньої історії прослуховування користувача, популярності треку тощо).

Однак якщо до системи додається новий виконавець, виникає так звана проблема холодного запуску. У цьому випадку для аналізу аудіофайлів потрібні методи машинного навчання. Доріжки перетворюються на спектрограми (частотно-часові графіки звуків), а потім подаються на вхід нейронної мережі, яка починає їх аналізувати і намагається зрозуміти, як вухо користувача сприймає мелодію.

Окрім аналізу спектрограм, Spotify використовує методи обробки природної мови (NLP) для аналізу вмісту десятків мільйонів веб-сайтів, включаючи форуми, блоги та статті. Ця інформація обробляється, щоб зрозуміти ставлення користувачів до певних пісень, проаналізувати поточні тенденції популярності та іншу інформацію, яка допомагає рекомендувати певні треки.

Кожен метод рекомендації має свої недоліки, але шляхом їх поєднання, збільшення можливості дослідження та забезпечення збору явних і неявних даних, які допомагають генерувати рекомендації, система забезпечує високу якість наданих рекомендацій, що підтверджується кількістю користувачів Spotify.

2. ПОСТАНОВКА ЗАДАЧІ

2.1 Дослідження існуючих методів надання рекомендацій

У цьому розділі представлено основні методи та алгоритми колаборативної фільтрації. У цьому документі розглядатимуться алгоритми на основі пам’яті та моделі.

Основна ідея цього методу вибору рекомендацій базується на схожості між користувачами та/або елементами. Слово «колаборація» означає, що деякі спільні дії різних користувачів повинні бути інтегровані в систему, а інформація про ці дії використовується для фільтрації великої кількості інформації та створення рекомендацій. Наприклад, припустимо, що є два користувачі A і B, які мали однакові оцінки щодо деяких елементів у минулому. На основі цієї інформації можна вважати, що ці два користувачі схожі, тому історія їхніх уподобань матиме більший вплив на рекомендації, які вони отримують, ніж попередні уподобання інших користувачів, які не мають із ними нічого спільного.

Щоб відобразити ступінь симпатії користувача до елемента, можна використовувати різні системи рейтингу користувачів. Наприклад, числовий рейтинг, двійковий рейтинг і уніграмний рейтинг. Числовий рейтинг - оцінити щось за шкалою від низького до високого (наприклад, рейтинг фільму або книги). Бінарний рейтинг – користувачі можуть вибирати лише між «подобається» і «не подобається» (наприклад, подобається чи не подобається відео на YouTube). Рейтинг Unigram - користувачі можуть відзначати лише той контент, який їм подобається (наприклад, лайкнути пост в Instagram). Ця оцінка відноситься до явного методу збору інформації про вподобання користувача.

Перед розглядом алгоритму необхідно ввести поняття матриці взаємодії користувач-елемент (скорочено U-елемент):

U представляє набір користувачів системи (U = {u1, u2, …, un}), а E представляє набір елементів (E = {e1, e2, …, em}). R представляє набір оцінок, а

rue представляє рейтинг користувача «u» до елемента «e». Якщо ми представимо користувачів як рядки матриці, елементи як стовпці матриці, а перетин рядків і стовпців представляє оцінку відповідного користувача для відповідного продукту, ми отримаємо матрицю взаємодії.

Слід зазначити, що якщо система містить велику кількість користувачів і продуктів, матриця буде дуже розрідженою, оскільки кожен користувач не буде оцінювати кожен продукт.

Ця матриця є основною основою алгоритмів спільної фільтрації.

Алгоритми спільної фільтрації можна розділити на:

• Алгоритми на основі пам’яті – алгоритми, які прості у використанні, але займають велику пам’ять.

• Алгоритми на основі моделі - використовуйте алгоритми машинного навчання, які можуть значно зменшити використання пам'яті.

Алгоритми на основі пам'яті

Алгоритми на основі пам’яті намагаються спрогнозувати оцінку користувача продукту шляхом обчислення середньозваженої оцінки продукту користувачами, схожими на даного користувача. Алгоритми на основі пам’яті завантажують всю матрицю взаємодії в системну пам’ять і роблять прогнози на основі даних, що містяться в пам’яті.

Алгоритми на основі пам’яті далі поділяються на фільтрацію на основі користувача та фільтрацію на основі елементів. Ці два методи дуже схожі, але результати відрізняються. Результат фільтрації на основі користувачів можна описати так: «Користувачам, схожим на вас, також подобається...». У той час як фільтрація на основі елементів, швидше за все, рекомендуватиме продукти, які відповідають наступному твердженню: «Користувачам, яким подобається цей продукт, також подобається...». Обидва методи покладаються на матрицю взаємодії у своїх алгоритмах.

Фільтрація на основі користувача

Фільтрування на основі користувача включає наступні кроки:

1. Нормалізація даних.
2. Розрахунок подібності.
3. Вибір сусіда.
4. Оцінка товару.
5. Вибір товару.

Тепер обговоримо кожен етап докладніше.

Поширеною проблемою є те, що різні користувачі по-різному оцінюють продукти, тому користувачі зі схожими інтересами можуть давати різні оцінки різним продуктам. Якщо ми маємо справу з рейтинговою системою, то потрібно враховувати людський фактор, тобто одні люди з доброти можуть, в принципі, всьому давати високі оцінки, а інші — низькі. Тобто оцінка 8/10, яку поставила одна особа, може бути такою ж, як оцінка 4/10, яку поставила інша особа. Щоб запобігти втручанню людського фактора в роботу алгоритму, першим кроком алгоритму є нормалізація оцінок.

Найпоширенішими методами нормалізації рейтингу є нормалізація за Гауссом, MinMax і нормалізація без зв’язку.

Нормалізація Гауса: цей метод враховує два фактори, які можуть спричинити відхилення в оцінках користувачів зі схожими інтересами:

1. Упередження середнього рейтингу. Проблема полягає в тому, що деякі користувачі можуть бути більш лояльними до продукту і тому давати вищі оцінки, ніж більш вибагливі користувачі. Таким чином, середня оцінка більш лояльних користувачів буде вищою, ніж у більш вибагливих. Щоб виключити вплив цього фактора на розрахунок рекомендацій, необхідно відняти рейтинг кожного користувача від середнього рейтингу відповідного користувача.

2. Дисперсія різних шкал оцінювання. Проблема полягає в тому, що більш консервативні користувачі мають менший діапазон оцінок, тоді як більш

ліберальні користувачі мають більший діапазон оцінок. Щоб врахувати цей фактор, необхідно рейтинг кожного користувача розділити на дисперсію його рейтингу [4]. (Дисперсія - це міра ступеня дисперсії значення випадкової величини відносно середнього значення розподілу [5].)

Поєднуючи ці ідеї, ми отримуємо формулу (2.1) $R_y(x)_{avg}$ для розрахунку нормалізованого рейтингу користувача y для продукту x.

$$R_y(x)_{avg} = \frac{R_y(x) - \bar{R}_y}{\sqrt{\Sigma_x\left(R_y(x) - \bar{R}_y\right)^2}} \tag{2.1}$$

У цій формулі $R_y(x)$ – це оцінка користувача x для продукту y, $\bar{R_y}$ – середня оцінка користувача (тобто сума всіх оцінок, наданих певним користувачем, поділена на кількість оцінок, наданих конкретним користувачем).

Тобто у знаменнику формули виключений перший штучний фактор – більш лояльний і менш лояльний рейтинги. У чисельнику дисперсія оцінок користувачів обчислюється за допомогою формули дискретності для дискретних випадкових величин.

Як альтернатива, для вирішення цієї проблеми можна використати нормалізацію MinMax, значення якої коливається від 0 до 1. У цьому випадку формула 2.2 для розрахунку нормалізованої оцінки виглядає наступним чином:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2.2}$$

Де x_{norm} – нормалізований бал, x_{min} – найнижчий бал користувача, x_{max} – найвищий бал користувача.

Однак під час фактичної роботи та впровадження нормалізації я виявив, що застосування цих формул до балів усіх користувачів із однаковими балами призведе до ділення на нуль. Щоб вирішити цю проблему, під час фактичної роботи, коли різниця в балах дорівнює нулю, ми вирішили встановити середню оцінку користувача як середню з дозволених оцінок. Тобто $xnorm=rmax/2$, де $rmax$ це максимальний бал, дозволений після нормалізації.

Іншим методом нормалізації є нормалізація без зв’язку. Цей метод нормалізації перетворює оцінку продукту користувачем у ймовірність того, що продукт сподобається користувачеві. Він використовується в багатокритеріальній спільній фільтрації [6]. Ідея цієї спільної фільтрації полягає в більш детальному моделюванні вподобань користувачів на основі оцінок різних аспектів продукту чи послуги. Слід додати, що для цього типу системи рекомендацій замість двовимірної матриці використовується тривимірна матриця взаємодії: матриця користувач×елемент×критерій (Un×m×k). Щоб обчислити ймовірність того, що продукт сподобається користувачеві на основі оцінки користувача певних аспектів продукту, ми використовуємо такі два припущення:

1) Коли велика кількість продуктів отримує найвищу оцінку користувача за параметром R, користувачеві, ймовірно, сподобається продукт із найвищою оцінкою за параметром R.

2) Коли більшість продуктів мають рейтинг R, користувачеві навряд чи сподобаються продукти категорії R.

Пояснення: припустімо, що у нас є продукт «взуття» з 3 категоріями рейтингу: зручність, довговічність і естетичність.

Давайте подивимося, яке взуття купив користувач. Ми бачимо, що найвища оцінка користувача в категорії «Комфорт», тому можна припустити, що користувач віддає перевагу зручному взуттю та ігнорує естетику та довговічність — ця частина пояснення відображає ідею першої гіпотези. Припустимо, що користувач все-таки вибирає комфорт, але після аналізу товарів ми виявили, що більшість взуття на сайті є зручним, тобто на сайті взагалі не продається незручне взуття. Тому, оскільки все взуття зручне, комфорт не може бути головним критерієм для користувача, але довговічність є найважливішою для нього, і високий комфорт є загальною рисою цього типу продукції.

Базуючись на наведених вище припущеннях, ми будуємо формулу (2.3) для перетворення рейтингу товару в імовірність того, що продукт сподобається користувачеві.

$$p_y(A) = p_y(Rating \leq R) - \frac{p_y(Rating = R)}{2} \quad (2.3)$$

Ця формула розраховує ймовірність події A, де A означає, що користувачеві подобається продукт, який отримав найвищу оцінку за параметром R. $py(Rating<R)$ представляє відсоток елементів із оцінкою не вищою за R, і $py(Rating=R)$ представляє відсоток елементів з тим же балом, що й R.

Після виконання необхідної нормалізації можна обчислити подібність. Розрахунок подібності є одним із ключових факторів у спільній фільтрації, оскільки список отриманих рекомендацій залежить від того, які користувачі схожі на даного користувача.

Найпримітивніший спосіб визначити схожість користувачів полягає в тому, що їм подобається один і той же продукт. Набори налаштувань збігаються, тому користувачі мають щось спільне. Однак на основі такого підходу може бути згенеровано багато нерелевантних рекомендацій, оскільки недостатньо просто ідентифікувати схожих користувачів; вам потрібно вибрати користувачів, які максимально схожі на даного користувача. Термін «міра подібності» вводиться тут, а метод її обчислення буде обговорено пізніше. Наприклад, щоб обчислити міру подібності між двома користувачами (а також між продуктами), користувачі представлені у формі векторів оцінки продукту, а потім для двох векторів користувачів подібність обчислюється за допомогою формули для міри подібності.

У системах рекомендацій можна використовувати наступні заходи подібності:

- Міра косинуса
- Коефіцієнт кореляції Пірсона
- Евклідова відстань
- Манхеттенська відстань тощо.

Значення кожного показника подібності залежить від певних особливостей, тому вибір того, який показник подібності використовувати в

рекомендаційній системі, можна визначити шляхом експериментування та перевірки кореляції отриманих значень. Крім того, при виборі слід враховувати формат даних.

Виберіть формулу для розрахунку подібності між користувачем, для якого мають бути створені рекомендації, та кожним користувачем у системі.

Після обчислення подібності можна приступати до генерації рекомендацій, але варто звернути увагу на важливу деталь: чи потрібно враховувати думку кожного користувача при створенні для нього рекомендацій? Може здатися, що немає ніякої шкоди в розгляді оцінок усіх користувачів для підвищення точності, але якщо користувач X менш схожий на даного користувача, то рейтинг користувача X, швидше за все, матиме незначний вплив на даного користувача. Чим більше даних потрібно обчислити, тим довше потрібно отримати зворотний зв'язок від системи. Ось чому перед прогнозуванням рейтингу додається ще один етап: вибір сусідів. Вибір сусідів означає рішення, чиї рейтинги враховувати під час створення рекомендації. Варто зазначити, що цей етап необов'язковий і його можна пропустити, якщо кількість користувачів і продуктів у системі невелика або достатньо ресурсів для виконання необхідних розрахунків. Ідея полягає в тому, що якщо користувачі абсолютно різні, то немає сенсу враховувати їх при розрахунку рекомендації.

Традиційно існує чотири підходи до відбору користувачів, які будуть розглядатися для рейтингу [7]:

• Усі користувачі – цей підхід придатний лише тоді, коли вибірка даних невелика і не надто навантажує систему.

• Порогове значення – для отриманого показника схожості встановлюється порогове значення, і враховуються лише оцінки користувачів, чия схожість з користувачем досягає порогового значення.

• K найближчих сусідів – відсортуйте користувачів за подібністю та виберіть перших k користувачів.

• Виконайте кластеризацію користувачів і враховуйте тільки оцінки користувачів у тому самому кластері (використання цього підходу перетворює

алгоритм на гібридний алгоритм, оскільки кластеризація є алгоритмом фільтрації на основі моделі, який буде розглянуто у відповідному розділі).

На основі одного із способів виберіть набір користувачів, які будуть враховуватися при формуванні прогнозованого рейтингу.

Коли у вас будуть схожі користувачі, ви можете почати оцінювати ймовірність того, що продукт сподобається цьому користувачеві. Перше, що варто відзначити, недоцільно рекомендувати користувачеві продукти, які він вже оцінив. Крім того, зменшення кількості оцінюваних товарів може знизити навантаження на систему та прискорити генерацію рекомендацій, оскільки зменшує обсяг необхідних обчислень.

Тому першим кроком в оцінці товарів є виключення вже оцінених продуктів з оцінки. Далі розраховується початковий рейтинг за такою формулою:

$$R_{u,i} = k \sum_{u' \in U} sim(u, u')\, R_{u',i} \tag{2.4}$$

Де Ru,i – прогнозована оцінка користувача для продукту i, $sim(u,u')$ - схожість між користувачем u та користувачем u', Ru',i - оцінка користувачем u' продукту i, k – нормалізований коефіцієнт.

З формули зрозуміло, що схожість користувачів є ваговим фактором для їхніх оцінок.

Нормалізований коефіцієнт обчислюється за формулою:

$$k = \frac{1}{\sum_{u' \in U} |sim(u, u')|} \tag{2.5}$$

Застосувавши ці формули, ми отримаємо прогнозований рейтинг цього користувача для кожного елемента, який раніше не оцінювався. Залишилося відсортувати товари в порядку спадання за рейтингом і подати користувачеві

перші n позицій, де n – потрібне число. Якщо рейтинг занадто низький, користувачеві також можна рекомендувати лише товари з рейтингом вище певного порогу.

Фільтрування на основі елементів

Цей алгоритм рекомендацій подібний до алгоритму на основі користувача, але порівняння базується не на векторах користувача, а на векторах елементів. Оскільки методи подібні, здається, що ймовірність отримання схожих результатів висока, але це не так. Можна сказати, що метод, заснований на елементах, є більш загальним і менш персоналізованим, оскільки кількість оцінок користувачів для окремого товару зазвичай перевищує кількість усіх оцінок для одного користувача. Цей метод враховує кореляцію між послугами.

Щоб краще зрозуміти різницю фільтрації на основі елементів, розглянемо кроки алгоритму, згадані в методі на основі користувача. Отже:

1. Нормалізація рейтингу - кроки залишаються тими самими.

2. Розрахунок подібності - обчислення подібності полягає не в схожості між користувачем та іншими користувачами, а в подібності між елементами, які сподобалися користувачеві, та іншими елементами.

3. Вибір сусідів - виберіть товари, які найбільше схожі на товари, які сподобалися користувачеві.

4. Оцінка продукту - оцінюйте продукти на основі їх схожості та попередніх оцінок користувача.

5. Вибір товару - кроки ті ж.

Методи на основі елементів також мають багато переваг. Уподобання користувачів можуть змінюватися з часом, але описи продуктів змінюються рідко, якщо взагалі змінюються. Оцінка схожості продуктів точніша, ніж оцінка схожості користувачів, оскільки ми маємо більше інформації для прийняття рішень.

Основною проблемою алгоритмів на основі пам’яті є великий обсяг пам’яті, необхідний для завантаження повної матриці взаємодії, а також для виконання та збереження значень додаткових обчислень, виконаних на цій матриці. Продуктивність таких систем погіршується, коли кількість продуктів і користувачів є великою. Для вирішення таких проблем з’явилися методи генерації рекомендацій на основі моделі. Вони використовують п’ять загальних підходів: кластеризація, класифікація, латентні моделі, марковські процеси прийняття рішень і матрична факторізація. [9] Давайте обговоримо деякі з цих методів більш детально:

Кластеризація

Метод заснований на припущенні, що користувачі з однієї групи мають однакові інтереси і тому дають однакові оцінки продуктам. Користувачі поділяються на різні групи (підгрупи, тобто найбільш схожі користувачі).

У процесі поділу кожен користувач представляється у вигляді вектора, що містить усі оцінки продукту. Несхожість між двома користувачами вимірює відстань між ними. Ця відстань обчислюється за формулами для евклідової відстані, відстані Манхеттена або відстані Мінковського. Чим менша відстань, тим вище схожість між користувачами. За допомогою цього методу можна зменшити кількість обчислень, оскільки пошук рекомендацій для користувача X використовує лише інших користувачів із тієї ж групи, що й користувач X.

Перш ніж описувати алгоритм, введемо поняття «центроїд». Центроїд - це вектор, елементами якого є середні значення кожної змінної, обчислені по всіх записах групи [8].

Найпопулярнішим алгоритмом кластеризації є алгоритм K-середніх, який складається з наступних кроків:

1. Випадково виберіть k користувачів, кожен з яких представляє середнє значення (центроїд) групи.

2. Для кожного користувача обчисліть відстань між ним і кожним кластером і призначте користувача кластеру з центроїдом із найменшою відстанню до нього.

3. Обчисліть нове середнє (центроїд) кожного кластера та повторіть крок 2, щоб призначити користувачів. Повторюйте кроки 2 і 3, доки система не стане стабільною. Якщо центроїди кластера не змінюються після повторного обчислення або зсув кожного центроїда досить малий (попередньо встановлене порогове значення), система стабільна.

Основною проблемою кластеризації є розрідженість матриці взаємодії користувача, що призводить до неправильних значень при розрахунку кластерів. Щоб вирішити цю проблему, пропонується алгоритм, який спочатку групує подібні продукти в групи, а потім використовує групи продуктів для створення груп користувачів на основі цих груп продуктів.

Розкладання матриці

Декомпозиція матриці є одним з найпопулярніших методів генерації рекомендацій. Ідея цього методу полягає в розкладанні матриці взаємодії користувача на добуток матриць низької розмірності.

Ідея цього методу полягає в тому, що певні фактори можна використовувати для унікального опису продуктів системи. Ці фактори також можна використовувати для опису інтересів користувачів. Використовуючи машинне навчання, система самостійно визначає ці фактори в процесі навчання. Системи, які використовують цей метод, зосереджені на пошуку зв’язків, які можна виразити математично. Щоб більш чітко описати ці фактори, розглянемо приклад: користувач високо оцінив книги A, B і C, але ми знаємо, що ці книги

належать до категорії наукової фантастики, тому ми визначаємо фактори, за якими ці книги подобаються користувачам. Це досить загальний приклад, і система, яка використовує цей метод, може знайти більш конкретні зв'язки між оцінками користувачів і продуктами. Деякі зв'язки можуть бути неочікуваними, але вони мають математичні основи. Можна сказати, що алгоритм усуває шум і зберігає корисні дані.

nxm-вимірна матриця взаємодії користувача (R) (n – кількість користувачів, а k – кількість продуктів) представляється як добуток двох матриць. U являє собою nxk-вимірну матрицю, що відображає приховані фактори користувачів; I являє собою kxm-мірну матрицю, що відображає приховані фактори продуктів.

Алгоритм пошуку матриць U і I наступний:

• Ініціалізувати задані матриці випадковими значеннями.

• Перемножте їх і порівняйте з матрицею R. Обчисліть похибку кожної одиниці та загальну похибку.

• Використовуйте методи мінімізації помилок (включаючи найменші квадрати, градієнтний спуск тощо), щоб зробити добуток U*I якомога ближчим до матриці R.

Рейтинг користувача буде розраховуватися за такою формулою:

$$\tilde{r}_{ui} = \sum_{f=0}^{k} U_{u,f} I_{f,i}$$

Обчислення прогнозованого рейтингу на основі матриці, отриманої під час процесу декомпозиції, значно зменшить кількість обчислень, оскільки розмірність матриці менша, тому можна виконати менше операцій.

2.2 Типові проблеми при побудові системи рекомендацій

Будь-яка система рекомендацій зіткнеться з багатьма проблемами при розрахунку прогнозованого рейтингу. Це пов’язано з недостатньою кількістю даних, обмеженнями ресурсів і програмного забезпечення. У цьому розділі розглядаються типові проблеми та їх вирішення.

Колекція оцінок відомих користувачів

Ця проблема викликана людським фактором. Користувачі системи рекомендацій не завжди оцінюють товари, які купують. Припустимо, користувач А купує товар Б, але не оцінює його. Таким чином, система не може отримати інформацію про те, чи подобається продукт користувачеві (або популярність продукту, якщо це рейтингова система), що може вплинути на формування наступних рекомендацій.

Існує два шляхи вирішення цієї проблеми:

- Відвертий: після того, як користувач придбає продукт, попросіть його оцінити його.

- Неявно: виходячи з попередньої інформації про вподобання користувача, спробуйте передбачити оцінку користувача після покупки продукту.

Крім того, повертаючись до попереднього розділу, ми можемо звернутися до системного підходу Spotify. Хоча він не використовує рейтинги, він може судити про те, чи подобається трек користувачеві, на основі часу прослуховування та кількості відтворень.

Холодний старт

На відміну від попередньої проблеми, холодний запуск - це проблема самої системи, а не проблема, пов'язана з діями користувача.

Є три варіанти цієї проблеми:

• У систему входить новий користувач. Система не має інформації про нього, тому не може визначити схожих користувачів і сформувати рекомендації.

• До системи додано новий продукт. Як і в попередньому пункті, товар ніхто не оцінює, тому він нікому не рекомендований. Тоді продукт стає непопулярним.

• Система запускається вперше. Оскільки з ним ще ніхто не взаємодіяв, матриця взаємодії в системі абсолютно порожня.

Для вирішення цієї проблеми потрібно:

• Коли новий користувач приєднується до системи, ви можете явно попросити його оцінити певні продукти/категорії, щоб зібрати інформацію про вподобання користувача, що допоможе вам знайти схожих користувачів і надати рекомендації. Зазвичай під час реєстрації на деяких веб-сайтах ви побачите, що система просить вас ввести жанр фільмів, пісень чи книг, улюблених авторів тощо. Усі ці дані дають системі можливість зібрати інформацію про користувача перед тим, як оцінювати його. Ви також можете використовувати інформацію про геолокацію користувача або напряму рекомендувати найпопулярніші продукти на даний момент.

• Додаючи нові продукти, ви повинні застосувати підхід до рекомендацій на основі вмісту. Нові продукти оцінюються на основі певних атрибутів, важливих у системі рекомендацій, і отримують свої гіпотетичні оцінки.

• Що стосується останнього варіанту, то тут можливим рішенням є використання методів машинного навчання. Знову повертаючись до прикладу Spotify, ми все одно можемо оцінити популярність певних продуктів на основі інформації в Інтернеті. Що стосується користувачів, то для збору інформації про них підходять вищевказані методи.

Масштабованість

Чим більше користувачів в системі, тим більше обчислень потрібно виконати. Важливо розуміти, що система, яка забезпечує швидкі та релевантні результати при використанні 1000 користувачів, може погіршитися, якщо кількість користувачів зросте до сотень тисяч або навіть мільйонів.

Цю проблему можна вирішити одночасно з двох напрямків:

- Поліпшити обладнання (використовувати більш потужні сервери, процесори, збільшити кількість серверів і т.д.)

- Програмне забезпечення (вдосконалити використовувані алгоритми, перейти до методів, які краще працюють з великою кількістю користувачів і з меншим навантаженням на систему)

Надспеціалізація

Ця проблема виникає, коли рекомендовані продукти занадто схожі один на одного. Наприклад, користувачі часто купують продукти на певному сайті.

Після цього йому рекомендують тільки цукор (різні марки, різні види, але тільки одну категорію товарів) з урахуванням товарів, які він купує найчастіше.

Можливим вирішенням цієї проблеми є рекомендація товарів, які можуть зацікавити користувача, але не схожі на товари, які він зазвичай купує (диверсифікація рекомендацій).

2.3 Вибір засобів розробки рекомендаційної системи

Для виконання поставленої задачі – розробки системи рекомендацій щодо вибору одягу – використовувалася мова програмування Python [8].

Python — це інтерпретована, об’єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією та автоматичним керуванням пам’яттю. Мова зосереджена на підвищенні продуктивності розробника, читабельності та якості коду, а також на забезпеченні переносимості програм, написаних на ньому.

Мова дозволяє швидко будувати прототипи і тестувати різні способи вирішення поставленого завдання. Таким чином, це значно скорочує час, який потрібен програмістам для досягнення кінцевого результату.

На додаток до вищезазначених функцій, Python підтримує не лише об’єктно-орієнтоване програмування, але й кілька стилів програмування, таких як імперативний, процедурний, структурований і функціональний. Крім того,

мова має дуже потужні засоби метапрограмування, які особливо корисні для розробки інтелектуальних систем і проведення різноманітних експериментів.

Завдяки цьому Python став однією з найпопулярніших мов у сфері інтелектуального аналізу даних і машинного навчання, а також має найбільшу спільноту вчених і розробників у цій галузі. В результаті він має велику кількість зручних інструментів для вирішення типових завдань.

Однак Python також має деякі недоліки. Найбільшим недоліком є його низька продуктивність порівняно з іншими мовами, особливо компільованими мовами програмування. Тому питання продуктивності та оптимізації обчислень особливо важливі під час розробки системи.

Щоб вирішити цю проблему, ми використали такі бібліотеки та фреймворки Python: Numpy, Scipy, Pandas, Matplotlib, Seaborn, Tqdm, Implicit, Sklearn і Lightgbm. Далі наведено детальний вступ до кожної бібліотеки та фреймворку.

Numpy — це бібліотека, яка додає підтримку великих багатовимірних масивів і матриць. Він також має велику кількість математичних функцій високого рівня для обробки цих масивів. Програмування за допомогою Numpy часто називають векторно-орієнтованим програмуванням, оскільки бібліотека зазвичай намагається мінімізувати кількість імперативного коду та спрощує всі обчислення до певних високорівневих операцій над масивами. Бібліотека дуже швидка, оскільки написана мовами C і Fortran і використовує Python як оболонку для низькорівневих API.

Scipy — це бібліотека, спеціально розроблена для наукових та інженерних обчислень. Він має велику кількість пов'язаних функцій, які допомагають з необхідними високоточними обчисленнями. Scipy містить модулі для оптимізації, інтеграції, спеціальних функцій, обробки сигналів, обробки зображень, генетичних алгоритмів, вирішення звичайних диференціальних рівнянь та інших задач.На наукові та інженерні розробки.

Pandas — це бібліотека, що спеціалізується на обробці та аналізі табличних і часових даних. Оскільки вхідними даними для цього завдання є

структурована таблиця, бібліотека дуже зручна. Оскільки ключові частини коду написані на мовах програмування C і Cython, бібліотека має хорошу оптимізацію та продуктивність.

Matplotlib — це бібліотека для візуалізації даних у вигляді двовимірної графіки. Він дозволяє швидко та зручно створювати різні ілюстрації та графіки в тому ж середовищі програмування, що й операції з даними.

Seaborn також є бібліотекою для візуалізації даних. Однак seaborn є високорівневою обгорткою Matplotlib, спеціалізованою для статистичної візуалізації.

Tqdm — це невеликий інструмент для візуалізації виконання складних і трудомістких числових обчислень.

Implicit — це бібліотека, яка реалізує деякі популярні алгоритми для побудови систем рекомендацій із неявними даними функцій на основі методів спільної фільтрації.

Sklearn — це платформа машинного навчання, яка надає функції для створення, навчання та перевірки різних алгоритмів. Бібліотека працює спільно з бібліотеками Numpy і Scipy. Sklearn — одна з найпопулярніших бібліотек машинного навчання.

Lightgbm — це фреймворк, який реалізує один із найпотужніших методів машинного навчання — посилення градієнта. Реалізація Lightgbm дуже швидка та оптимізована серед інших доступних альтернатив. Підвищення градієнта може вирішити такі проблеми, як регресія, класифікація, сортування тощо. Бібліотека також підтримує обчислення GPU, що допомагає значно скоротити час, необхідний для створення моделі.

РОЗДІЛ 3.
ПРОЕКТУВАННЯ СИСТЕМИ РЕКОМЕНДАЦІЙ ДЛЯ СЕЗОННОГО ПІДБОРУ ОДЯГУ

3.1. Моделювання системи рекомендацій.

Для моделювання архітектури та бізнес-процесів розробленої інформаційної системи ми застосували підхід до створення діаграми BPMN.

У бізнес-процесі використання рекомендаційної системи можна виділити наступні аспекти:

• Обробка даних про продукти, клієнтів та їх взаємодію

• Збір та аналіз даних

• Формування рекомендацій

На рисунку 3.1 представлена схема розглянутого бізнес-процесу.

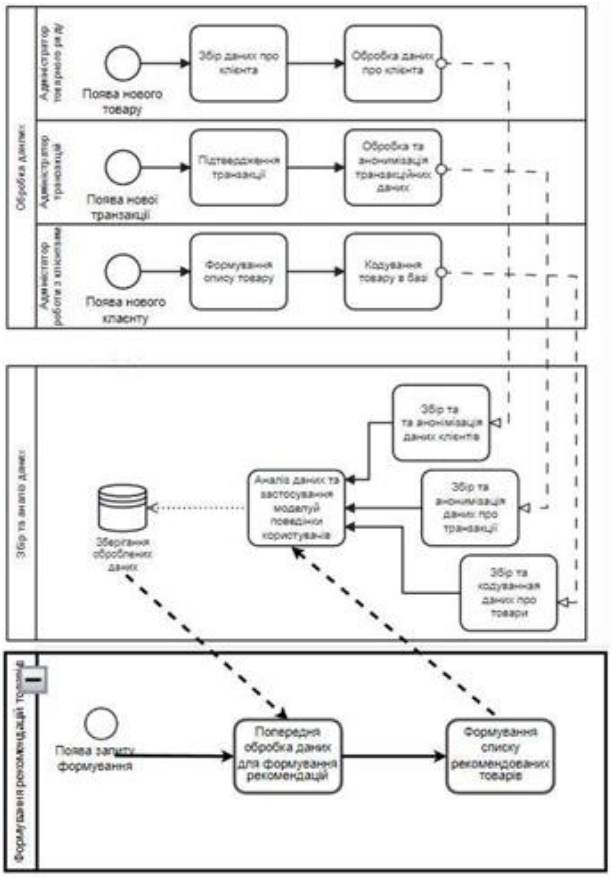


Рисунок 3.1 — BPMN Діаграма бізнес процесів інформаційної системи рекомендацій.

Щоб побудувати та навчити систему рекомендацій, ми використали персоналізований набір даних рекомендацій щодо моди H&M [ ], який містить серію зображень продуктів і такі файли: articles.csv, customers.csv і transactions_tran.csv. Далі розберемо їх детально.

articles.csv – цей файл містить докладні відомості про всі доступні продукти. «Таблиця 3.1» надає приклад даних у цьому файлі. Файл містить 105 000 записів, кожна з яких має 25 полів. Давайте подивимося на вміст деяких полів цього файлу:

□ article_id – унікальний ідентифікатор для кожного запису;

□ product_code, prod_name – унікальний ідентифікатор для кожного продукту та його назва (не обов’язково унікальна);

□ product_type, product_type_name – код групи товарів та її назва;

□ graphic_appearance_no, graphic_appearance_name – код групи графічного дизайну товару та його назва;

□ colour_group_code, colour_group_name – код групи кольорів товару та його назва;

□ perceived_colour_value_id, perceived_colour_value_name, perceived_colour_master_id, perceived_colour_master_name – додаткова інформація про колір товару;

□ department_no, department_name – унікальний ідентифікатор товарного відділу та його назва;

□ index_code, index_name – індекс товару в групі товарів та його назва;

□ section_no, section_name – унікальний ідентифікатор товарного відділу та його назва;

□ outfit_group_no, garment_group_name – індекс типу одягу та його назва;

□ outfit_group_no, garment_group_name – текстовий опис товару;

Таблиця 3.1 – Приклад даних із файлу Articles.csv

	article	product_ code	prod_na me	product_ type_no	product_ type_name	…
1	2	3	4	5	6	7

Продовження табл. 3.1

1	2	3	4	5	6	7
0	10877	108775	Strap top	253	Vest top	…
1	10877	108775	Strap top	253	Vest top	…
2	10877	108775	OP T-	253	Bra	…
…	…	…	…	…	…	

customer.csv – цей файл містить усю доступну інформацію про користувачів. У таблиці «Таблиця 3.2» наведено приклад даних із цього файлу. У файлі є 1,5 мільйона клієнтів, описаних у таких полях:

□ customer_id – унікальний ідентифікатор для кожного користувача;

□ FN – двійкове значення, яке визначає, чи отримує користувач розсилку;

□ Active (Активний) – двійкове значення, яке визначає, чи був користувач нещодавно активним;

□ club_member_status – двійкове значення, яке визначає, чи є користувач учасником H&M Club (для додаткових знижок і бонусних програм)

□ fashion_news_frequency – як часто користувач отримує новини;

□ age – вік користувача;

□ postal_code – хеш поштового індексу користувача;

Таблиця 3.2 – Приклад даних з файлу customers.csv

	customer_id	FN	Active	…	age	postal_code
0	bacae5abe1	0	0	…	49	52043ee2
1	3b00ade914	0	0	…	25	2973abc5
2	2d5b43e64	1	1	…	24	64f17e6a
…	…	…	…	…	…	…

transactions_tran.csv – Цей файл містить усі доступні транзакції (покупки товарів користувачами). Таблиця «Таблиця 3.3» надає приклад даних у цьому файлі. Він містить 31 мільйон транзакцій. Давайте подивимося на вміст кожного поля цього файлу:

□ t_dat – дата операції;

□ customer_id – унікальний ідентифікатор користувача (з файлу клієнти.csv);

□ article_id – унікальний ідентифікатор запису (з файлу articles.csv);

□ price – стандартизована ціна товару;

□ sales_channel_id – покупка онлайн або офлайн;

Таблиця 3.3 – Приклад даних з файлу transactions_tran.csv

	t_dat	customer_id	article_id	price	sales_channel_id
0	2025-09-20	bacae5abe1	108775015	0.05081	2
1	2025-09-20	bacae5abe1	118473015	0.03049	2
2	2025-09-20	2d5b43e64	685687003	0.01523	2
…	…	…	…	…	…

Images – колекція зображень продуктів. Загалом є 105 тисяч зображень, по одному для кожного продукту. Зображення сортуються за категоріями (всього 86 категорій). Назви зображень такі: category_index/product_index. Усі зображення кольорові та мають розмір 1750 x 1166 пікселів.

Як вихід системи організаторам потрібен файл csv з 12 рекомендованими товарами для кожного покупця. Ці рекомендації повинні передбачити, що кожен покупець купить наступного тижня.

Розвідувальний аналіз даних

Для кращого розуміння даних та їх характеристик перед вирішенням завдання зазвичай виконується дослідницький аналіз даних, який являє собою швидкий аналіз даних шляхом їх перетворення та представлення в зручному вигляді.

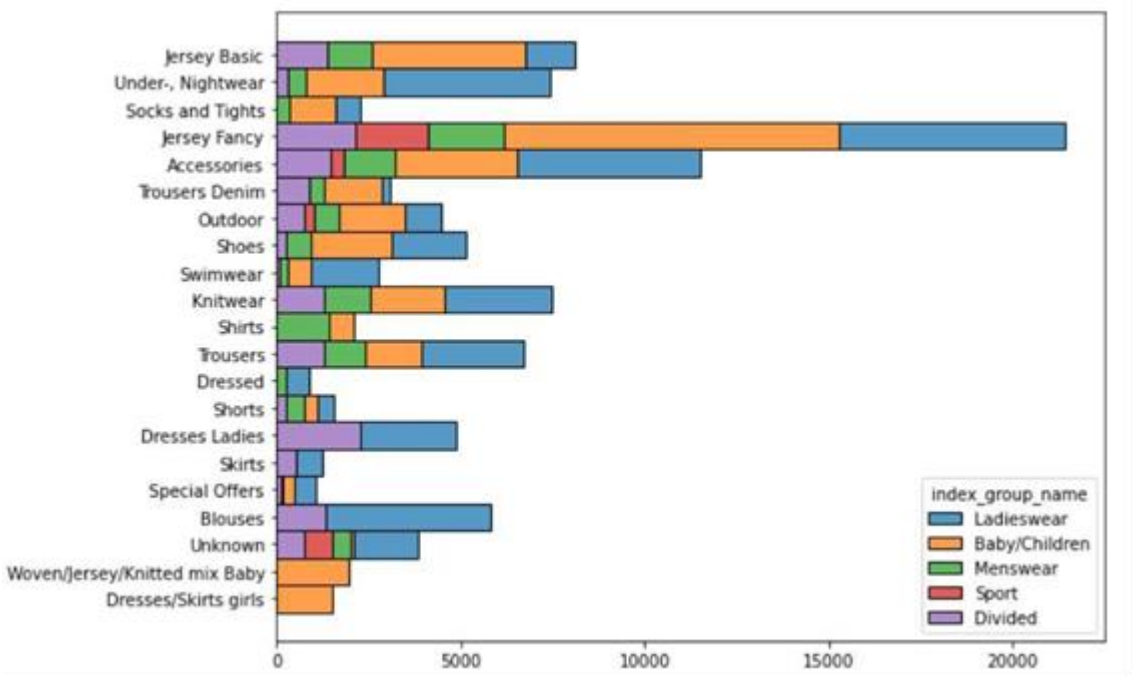


Рисунок 3.2 – Стовпчикова діаграма розміру груп товарів

Давайте спочатку перевіримо файл articles.csv. Давайте подивимося на кількість різних типів продуктів у кожній групі: на малюнку 2.2 показано стовпчасту діаграму кількості різних типів продуктів і розподіл кожної групи в цих продуктах. Як бачимо, групи та типи товарів розподілені нерівномірно. Найбільше товарів у групі жіночого одягу, найменше – у групі спортивного одягу.

Перевіримо структуру групи індексів товарів. З п'яти існуючих груп дві містять підгрупи: група дитячого одягу поділяється на групу аксесуарів, яка далі поділяється на три групи за висотою; група жіночого одягу поділяється на звичайний одяг, аксесуари та нижню білизну.

Давайте також перевіримо кількість унікальних значень у кожному стовпці:

- prod_name: 45875
- product_type_name: 131
- product_group_name: 19
- graphical_appearance_name: 30
- colour_group_name: 50

- perceived_colour_value_name: 8
- perceived_colour_master_name: 20
- department_name: 250
- index_name: 10
- index_group_name: 5
- section_name: 56
- garment_group_name: 21
- detail_desc: 43404

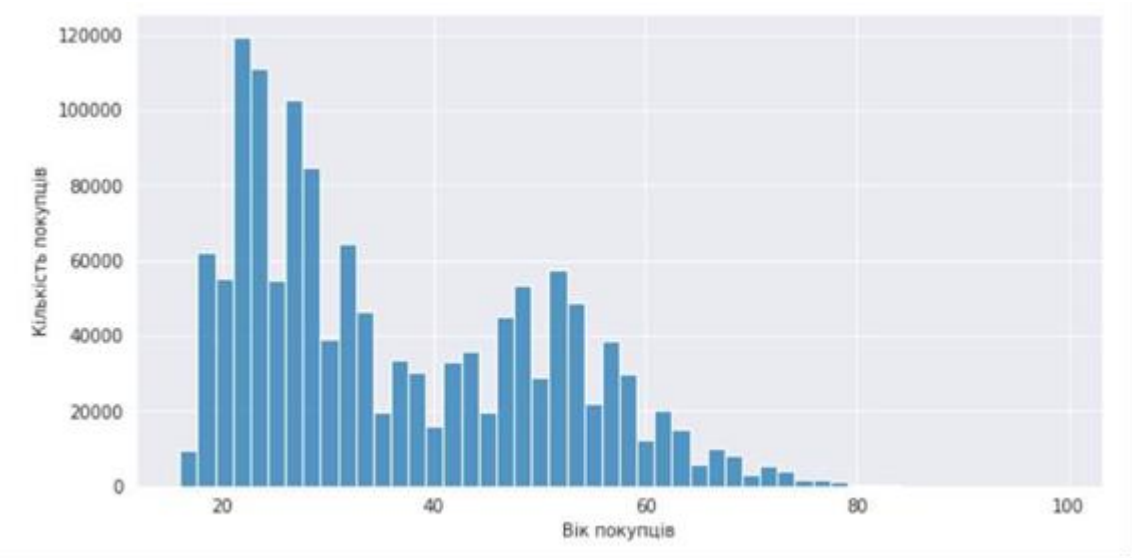


Рисунок 3.3 – Гістограма розподілу віку користувачів

Далі розглянемо файл клієнта. Особливої уваги заслуговує стовпець postal_code. Там ви можете спостерігати хеші поштового індексу клієнта. Цікаво, що одне зі значень з'являється дуже часто - більше 12 000 разів. Можна припустити, що це значення кодує невідомий поштовий індекс або код якогось великого розподільного центру.

Розглянемо розподіл користувачів за віком. На малюнку 2.3 показано гістограму розподілу користувачів за віком. Ми бачимо, що вік користувача є бімодальним, з режимами клієнтів 21 і 54.

Ми також розглядаємо змінну, яка визначає статус користувача в клубі магазину. Ми бачимо, що 93% користувачів є активними членами клубу, майже

7% знаходяться на етапі реєстрації, і лише 0,3% призупинили своє членство. Крім того, з цих 0,3% майже 99,5% нещодавно були неактивними. Враховуючи, що в середньому 66% клієнтів неактивні, можна припустити, що більшість користувачів, які вийшли з клубу, не повернуться в магазин. Тому цих клієнтів слід ігнорувати в конкурсі, оскільки вони навряд чи зроблять покупку наступного тижня.

Візьмемо приклад клієнтів, які підписуються на розсилку. Видно, що лише 35% клієнтів отримують розсилку. Решта взагалі не хочуть отримувати розсилку. Цікаво, що майже всі (97%) клієнти, які отримували розсилку, були активними протягом останнього тижня. Враховуючи, що загальний відсоток активних користувачів становить 33%. Тому можна вважати, що в конкурсі найцінніші користувачі, які отримали розсилку. Адже вони, швидше за все, залишаться активними протягом наступного тижня, тому для них слід будувати рекомендації. Тому, щоб отримати більш високі показники в конкурсі, варто звернути особливу увагу на користувачів, які підписалися на розсилку.

Нарешті, давайте подивимося на файл транзакцій - transactions.csv. Дивлячись на дати транзакцій, ми бачимо, що дані надані з вересня 2018 року по вересень 2020 року. В середньому в день здійснювалося 43 тисячі транзакцій.

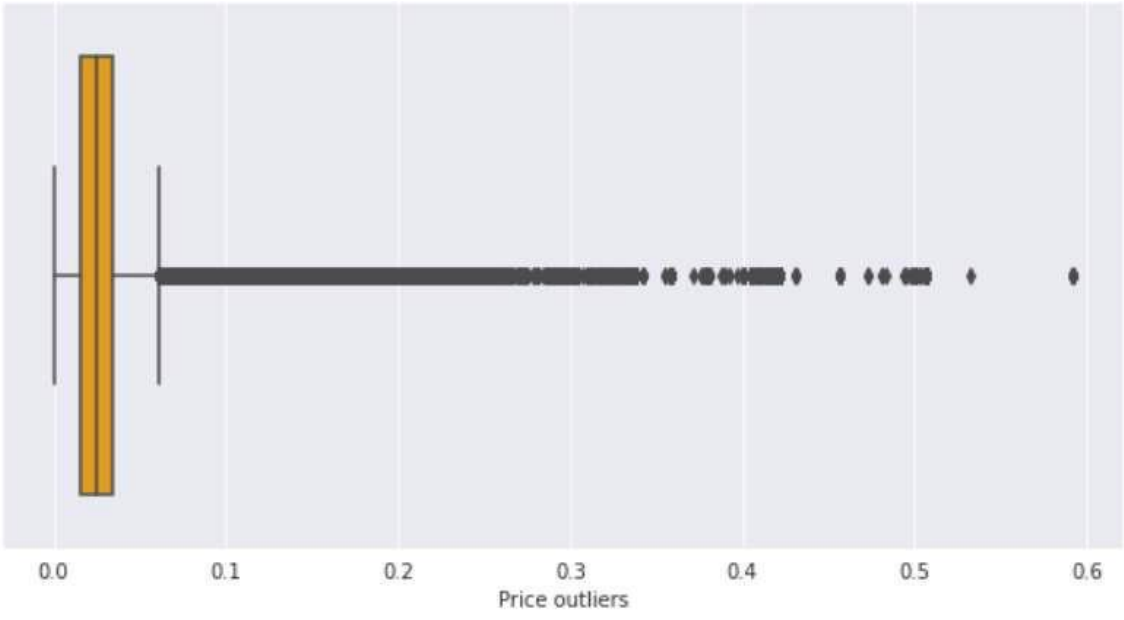


Рисунок 3.4 – Графік «ящик з вусами» ціни товару

Перевіримо значення цін на продукцію. Це нормалізовані значення реальних номінальних цін. Середнє значення дорівнює 0,0254, 25-й і 75-й процентиль - 0,158 і 0,0339, а максимальне і мінімальне значення - 0 і 0,5915. Розподіл цінових значень можна побачити на графіку «коробки з вусами» на малюнку 2.4. Крім того, є цікаве спостереження: користувачі, які робили покупки за цінами вище 75-го процентиля, в середньому робили значно менше покупок і були менш активними останнім часом. Тому вони не дуже цінні індикативні користувачі з точки зору результатів матчів. Максимальну увагу краще зосередити на покупцях, які зробили покупки в звичайному блакитному сегменті. Крім того, є невелика кількість покупців (<10 000), які зробили велику кількість покупок у магазині (>200 записів). Розумно припустити, що такі покупці потребують додаткової уваги, оскільки вони з більшою ймовірністю, ніж інші, зроблять наступну покупку протягом наступного тижня.

Виходячи з особистого досвіду, рекомендується перевірити гіпотезу про сезонність обсягу транзакцій і сезонність покупок окремих товарів. На рисунку 2.5 показано залежність регулярних одиниць об’єму від часу. Насправді можна побачити, що з березня по серпень існує регулярний сезон, протягом якого закупівлі значно вищі порівняно з рештою року. Крім того, різні види товарів поводяться дуже схожим чином. На мою думку, таку поведінку користувачів можна пояснити кількома факторами. По-перше, в цей період в магазинах проводяться різні акції та знижки. По-друге, у користувачів в гардеробі зазвичай більше літнього одягу, ніж зимового. Таким чином, загалом легкий одяг має вищі продажі та більшу варіативність, як показано на малюнку 3.5.

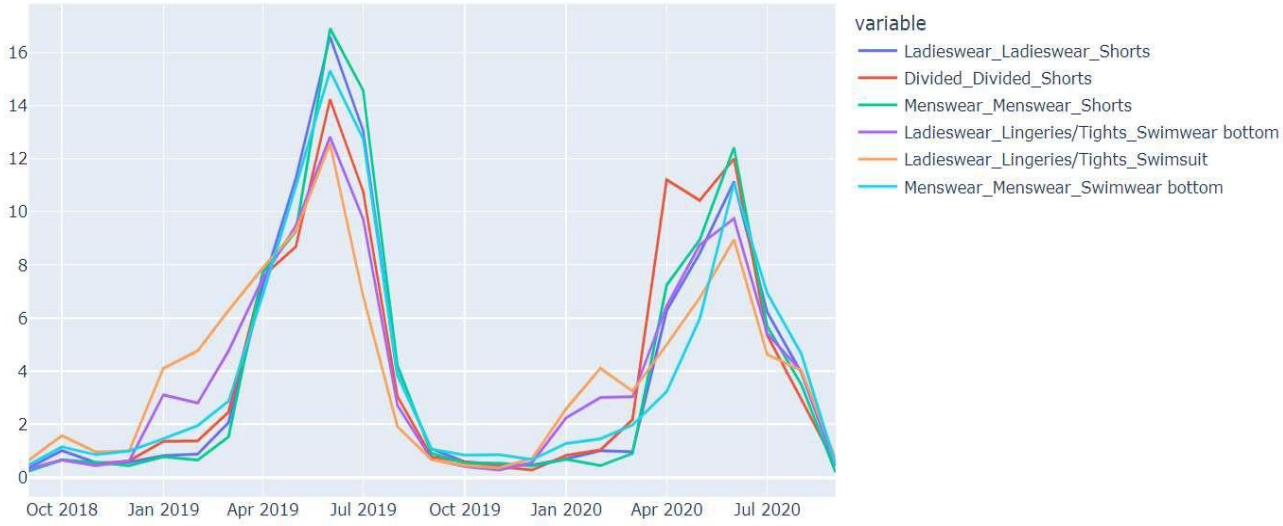


Рисунок 3.5 – Графік сезонності об’єму транзакцій

З вищенаведених міркувань можна зробити висновок, що вигідно розрізняти сезон, до якого належить той чи інший продукт. Адже в один сезон одні товари продаються добре, а в іншій — погано. Залежно від конкуренції є сенс скласти товарні рекомендації на останню неділю вересня. Визначити сезонність товару можна шляхом нормування кількості куплених товарів за вказаним вище сезоном. Таким чином, значення 1 означає продукт, який було придбано лише з березня по серпень, а значення 0 означає продукт, який ніколи не купувався в певний час.

Ми також вивчимо, як сезонність товару впливає на його глобальну структуру в просторі ознак. Для візуалізації ми будемо використовувати метод PCA — спроектувати всі характеристики продукту на двовимірну площину та розфарбувати продукти відповідно до їхньої сезонної оцінки, як показано вище. Результат показано на рисунку 3.6. Осі відповідають компонентам розподілу PCA. Видно, що сезонність продукту впливає на глобальну структуру простору ознак, тобто формує певні товарні кластери. Тому хорошим рішенням є додавання сезонності як окремої функції до остаточного алгоритму генерації рекомендацій.

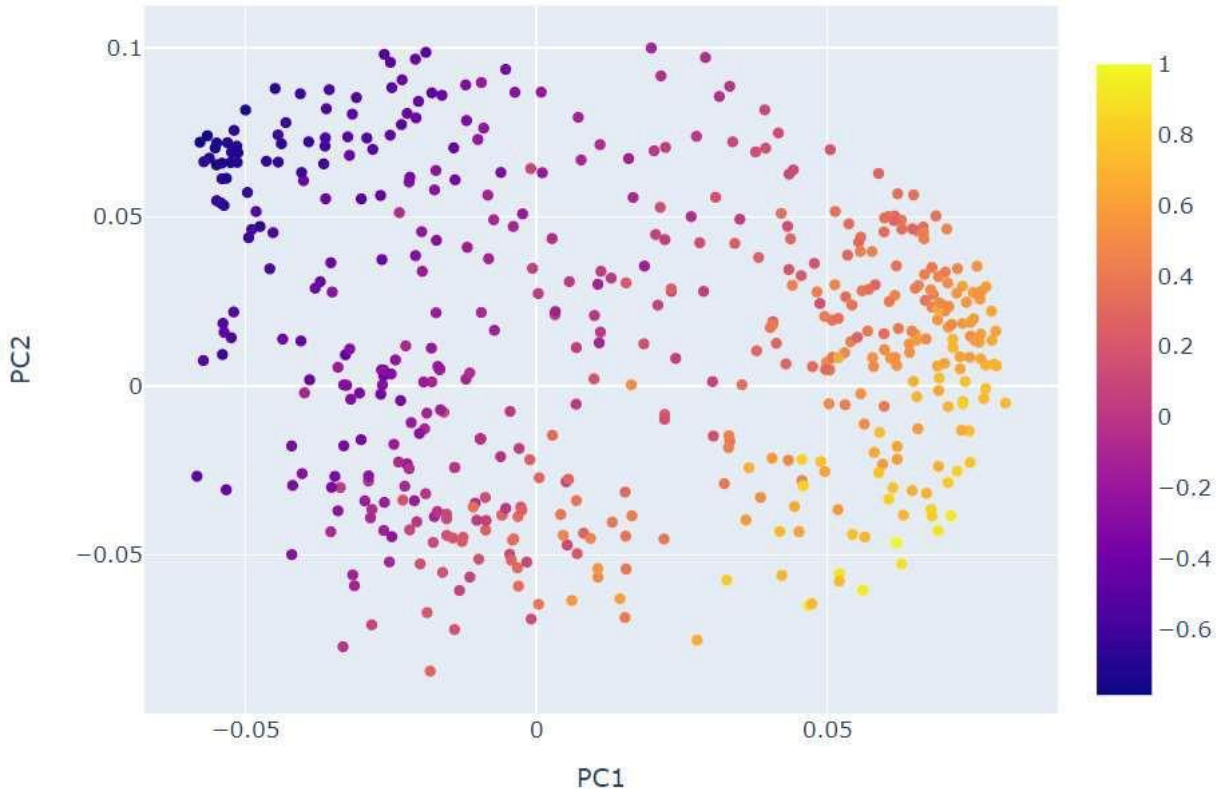


Рисунок 3.6 – Візуалізація структури простору ознак та сезонності товарів

Як бачите, інформація про трансакції суттєво допомагає звузити коло потенційних кандидатів для рекомендованих товарів. Однак існують інші подібні методи оптимізації, крім перерахованих вище. Наприклад, ви можете знайти продукти, які більше не виробляються. Якщо ви виберете вересень 2019 року як порогове значення умови, 20% усіх існуючих товарів не було придбано після цієї дати. Тому рекомендувати такі предмети немає сенсу.

3.2. Опис реалізації обраного методу побудови рекомендаційної системи

Озираючись на проблему, очевидно, що було б помилкою обмежуватися класичною спільною фільтрацією або моделями вмісту.

Найбільш правильним вибором є двофакторна система: на першому етапі відбираються кандидати, які можуть стати хорошими рекомендаціями; на другому кроці відібрані кандидати ранжуються.

На етапі проектування ми також вирішили відмовитися від аналізу існуючих зображень. Адже це потребує чималих обчислювальних ресурсів і, як показав досвід інших конкурсантів, суттєво не покращує якість кінцевої моделі.

Також важливо правильно спроектувати збір і попередню обробку даних, що вводяться в систему, і визначити, як сформувати набір рекомендацій на основі отриманих результатів моделі. Ці питання будуть детально розглянуті нижче.

Якщо кількість продуктів відносно невелика, може бути дуже важко створити модель для остаточного списку рекомендацій. Модель підсилення градієнта довела свою відмінну продуктивність у цьому рейтинговому конкурсі завдань. Тому ми рекомендуємо використовувати цю модель у цьому дослідженні.

Ця модель ранжирує продукти певним чином, розміщуючи продукти, які, на думку моделі, найбільше цікавлять користувача, у верхній частині. Отже, якщо у нас є кандидати з попередньою рекомендацією, після їх ранжування ми можемо вибрати перші 12 позицій як остаточну рекомендацію користувачеві.

3.3. Реалізація модулів обробки даних в рекомендаційній системі

Модуль збору та попередньої обробки даних.

Як зазначалося вище, щоб вирішити цю проблему, необхідно обробити відносно великий обсяг даних. Щоб полегшити експерименти та прискорити систему рекомендацій, необхідно розробити модуль, який може автоматично обробляти необроблені дані та оптимізувати, обробляти, зберігати та повторно використовувати оброблені дані.

У цьому випадку можна застосувати кілька ефективних методів оптимізації. По-перше, велика кількість рядкових індексів може бути замінена числовими індексами. Крім того, розріджені матриці можна використовувати при побудові необхідних матриць. Причому, значення категоріальних ознак можна закодувати у вигляді двійкових стовпців.

Рекомендується зберігати оброблені дані у форматі Parquet. Це дуже ефективний формат, який може значно зменшити кінцевий обсяг даних (порівняно з вхідним форматом csv), а також дозволяє швидко читати, записувати та змінювати дані.

Модуль формування кандидатів для рекомендацій.

Як зазначалося вище, аналіз зображень не використовуватиметься в цьому дослідженні. Однак ми не будемо відмовлятися від контент-аналізу повністю. Як показують результати EDA, аналізуючи певні закономірності в даних, ми можемо як знайти потенційно хороших кандидатів у рекомендації, так і відфільтрувати велику кількість продуктів, на які не варто звертати увагу.

Ми пропонуємо розробити набір евристичних правил для фільтрації потенційних рекомендаційних продуктів. Прикладом такого правила є додавання товарів, які користувач уже придбав, до рекомендованих кандидатів. Його також можна зробити складнішим, наприклад, додавши певні вагові коефіцієнти, такі як коефіцієнт повторної покупки або коефіцієнт часу з моменту останньої покупки.

Крім того, як показують результати EDA, продукти, які часто купують парами, також можуть бути чудовими кандидатами для рекомендацій. Таким чином, аналізуючи частоту взаємних покупок товарів, можна знайти хороші рекомендації щодо певних товарів.

Особливий інтерес представляє імпліцитна інформація про взаємодію покупця з товаром. Для цього випадку хорошим рішенням є використання трохи вдосконаленої моделі латентного фактора – iALS. Ця модель співпраці враховує неявні фактори, щоб знайти продукти, які, швидше за все, будуть рекомендовані користувачеві. Великою перевагою цієї моделі є те, що її можна застосовувати в різних сценаріях. Ви можете використовувати підхід на основі елементів для створення рекомендацій для одного користувача або підхід на основі користувачів для створення рекомендацій для групи користувачів. Останнє

виглядає особливо перспективним, оскільки, як показали результати EDA, інтереси різних груп користувачів можуть істотно відрізнятися.

Одним із найважливіших правил під час вибору товарів-кандидатів для рекомендації є врахування короткострокових тенденцій покупок (наприклад, випуски нового одягу, знижки, сезонні зміни). Таким чином, якщо протягом останнього тижня намітилася певна чітка тенденція, можна припустити, що ця тенденція збережеться.

Крім правил відбору товарів-кандидатів для рекомендації, також рекомендується розробити правила фільтрації товарів. Наприклад, як показав аналіз розвідки, сезонність купівлі деяких товарів дуже очевидна. Тому влітку не варто рекомендувати такі вироби, як зимовий одяг. Пропонується розрахувати середню сезонність товарів за минулий період часу та рекомендувати продукти з приблизно однаковими значеннями.

3.4. Опис архітектури розробленої рекомендаційної системи

Опис компонентів рекомендаційної системи

У процесі розробки програмного забезпечення ми прийняли структуру проекту Cookiecutter. Відповідно до цієї структури ми розробили чотири основні програмні компоненти: DataLoader, FeaturesExtractor, CandidatesRetrieval і Model. У таблиці 3.4 наведено опис функціонування модулів розробленої системи рекомендацій:

Таблиця 3.4 Опис функціонування модулів розробленої системи рекомендацій

DataLoader	Читає необроблені дані, попередньо обробляє, змінює, перетворює, групує та далі зберігає їх.

FeaturesExtractor	Виконує додаткову обробку даних і підготовку до спеціальних функцій, яких немає у вхідних даних, щоб покращити продуктивність системи рекомендацій.
CandidatesRetrieval	Цей модуль виконує попередній відбір продуктів-кандидатів у продукті на основі заданих правил для подальшого детального розгляду основною моделлю.
Model	Модель ранжує попередньо відібрані продукти-кандидати для формування остаточного списку рекомендацій. Цей компонент також містить функції, необхідні для перевірки якості моделі.

Графічний вигляд діаграми компонентів наведено на рисунку 3.7.

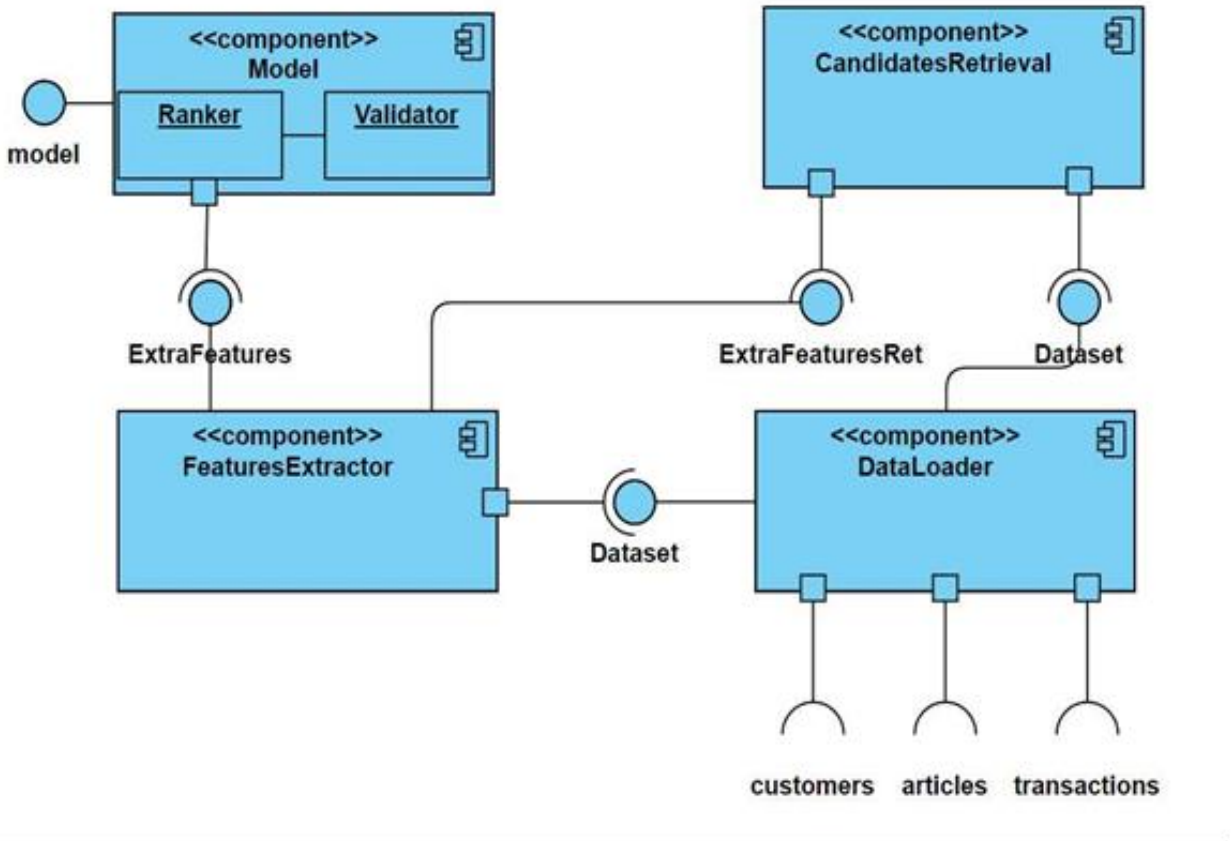


Рисунок 3.7 Діаграма компонентів інформаційної системи рекомендацій.

Таблиця 3.5. Опис основних класів рекомендаційної системи

Клас	Метод	Опис
DataLoader	_load_raw	Завантаження необроблених даних
	_ids_to_idxs	Кодування id користувачів та товарів у вигляді цілого числа
	_get_base_features	Отримати базові характеристики з даних
	_transform_cols	Закодувати мітки та змінити тип даних
	set_params	Встановити параметри
	save	Зберігти оброблені дані у форматі .pqt
	load	Завантажити оброблені дані з формата .pqt
	preprocess	Попередня обробка даних: закодувати індекси, закодувати категоріальні характеристики та очистити дані.
	train_val_split	Розділити дані на тренувальну та валідаційну вибірку
Validator	_ap_at_k	Значення метрики AP@K
	map_at_k	Значення метрики MAP@K
	recall_at_k	Значення метрики Recall@K
	hr_at_k	Значення метрики HitRate@K
FeaturesExtractor	cum_sales	Підрахунок акумулятивних продажів товару

Продовження табл. 3.2.

	week_sales	Підрахунок щотижневих продаж товару
	period_sale	Підрахунок продаж за останні n днів
	repurchase_frac	Підрахунок частки повторних купівель товару
	popularity	Підрахунок коефіцієнту популярності товару
RuleRetrieval	collect	Зібрати претендентів на рекомендації.
	_set_params	Встановити параметри відбору претендентів
	retrieve	Обирає претендентами на рекомендацію товари, що купляють схожі користувачі
	_get_freq_pair	Підрахувати частоти купівель разом
ALS(UserRule)	_to_sparce	Перевести дані у вигляд зрідженої матриці.
	_train	Тренування моделі
	retrieve	Передбачити кандидатів на основі моделі ALS
GroupALS(UserRule)	_to_sparce	Перевести дані у вигляд зрідженої матриці.
	_train	Тренування моделі

Продовження табл. 3.2.

	Retrieve	Передбачити кандидатів для групи користувачыв на основі моделі ALS
	_predict	Передбачити значення зацікавленості
	retrieve	Передбачити кандидатів за допомогою метода колаборатвиної фільтрації на основі товарів
UserSaleTrend(UserRule)	retrieve	Передбачити кандидатів на основі тренду для користувача
SaleTrend(ItemRule)	retrieve	Передбачити кандидатів серед популярних товарів під час тренду.
NoLongerAvaliable(FilterRule)	get_unavaliabel_items	Знайти товари, що вийшли з обороту
	retrieve	Відфільтрувати з рекомендацій недоступні товари.
Model	get_recomendations	Отримати список персональних рекомендації

Щоб запустити програмне забезпечення, необхідно встановити інтерпретатор Python версії >=3.7. Ви повинні встановити всі необхідні залежності, указані у файлі requirements.txt.

Усі бібліотеки та розширення можна встановити окремо, але для спрощення розгортання рекомендується встановити дистрибутив Python Anaconda. Більшість необхідних бібліотек і розширень вже включено в Anaconda.

Розроблена рекомендаційна система має різні варіанти використання. Щоб змінити варіант використання, вам потрібно внести деякі зміни у файл налаштувань settings.py. Давайте розглянемо роботу цього файлу докладніше:

Параметр FORCE_PREPROCESS відповідає за початок попередньої обробки даних. Якщо програмне забезпечення запускається вперше, цей параметр ігнорується та обробляється. Однак для повторного використання, якщо дані не змінилися, вони не можуть бути оброблені знову, але завантажуються результати попереднього запуску.

Параметр FORCE_RETRIEVAL має схоже значення з попереднім, але відповідає за попередній вибір рекомендованих кандидатів. Якщо стратегія відбору не змінилася, можна використовувати результати попереднього запуску. Змінна WEEK_NUM також належить до групи параметрів вибору. Він відповідає за те, наскільки глибоко здійснюється пошук тенденцій у даних під час процесу відбору кандидатів. Значення за замовчуванням — 6. Вищі значення можуть збільшити обсяг оперативної пам’яті програми, але продуктивність буде кращою. Останнім параметром вибору є змінна RETRIEVAL_TEST. Він відповідає за початок відбору кандидатів для тестування системи.

Останній параметр VALIDATE включає або вимикає остаточний тест продуктивності системи.

Гнучкі налаштування дозволяють виконувати мінімально необхідні операції, таким чином скорочуючи час виконання кінцевої програми та повторно використовуючи результати модуля з останнього запуску.

4. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Структурно-функціональний аналіз виробничого процесу та розроблення моделі травмонебезпечних ситуацій

У зображеннях процесів формування, виникнення аварій та виробничих травм усі випадкові події, що утворюють конкретну аварійну ситуацію, пов'язані між собою причинно-наслідковими зв'язками.

Метод логічного моделювання потенційних аварій, травм та катастроф відкриває можливість розробити досконалу систему управління ОП виробництва, яка базується на оперативному пошуку виробничих небезпек, їх глибокому аналізі й терміновому прийнятті заходів для усунення потенційних небезпек ще до виникнення травмонебезпечних та катастрофічних ситуацій. Деякі небезпечні ситуації в табл. 4.1.

Працівники, що обслуговують електрообладнання вениляційної системи, зобов'язані знати Правила безпечної експлуатації електроустановок споживачів відповідно до займаної посади або роботи, як вони виконують, і мати відповідну групу з електробезпеки [11,12].

Працівники, що порушили вимоги Правил безпечної експлуатації електроустановок, усуваються від роботи і несуть відповідальність (дисциплінарну, адміністративну, кримінальну) згідно з чинним законодавством. Такі працівники не допускаються до робіт в електроустановках без позачергової перевірки знань вимог правил безпечної експлуатації електроустановок.

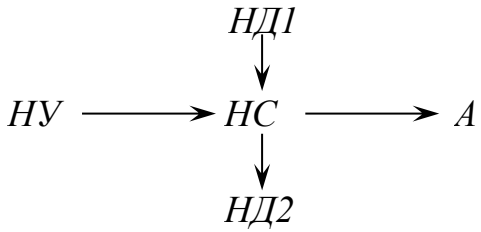
Забороняється допускати до роботи в електроустановках осіб, які не пройшли навчання і перевірку знань Правил безпечної експлуатації електроустановок.

Працівнику, який пройшов перевірку знань Правил безпечної експлуатації електроустановок, видається посвідчення встановленої форми.

Таблиця 4.1. Моделювання процесів формування та виникнення травмонебезпечних і аварійних ситуацій

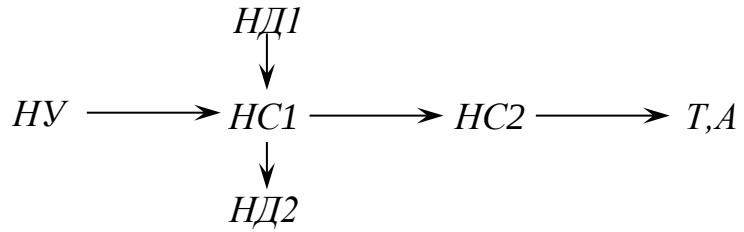
Вид робіт	Виробнича небезпека			Можливі наслідки	Заходи запобігання небезпечним ситуаціям
	Небезпечна умова (НУ)	Небезпечна дія (НД)	Небезпечна ситуація (НС)		
Використання механічної вентиляції	Оператор не перевірив обладнання НУ	Пошкоджений трубопровід мережі НД1 Закупорений трубопровід шланга НД2	Відмова вентиляційної системи (двигуна) НС	Аварія	Розвісити плакати, провести інструктажі із експлуатації обладнання системи

Модель процесу:



Використання електронних пристроїв регулювання	Пошкоджена ізоляція провідників з’єднання НУ	Пробій на корпус НД1 Коротке замикання НД2	Ураження людини електричним струмом НС1 Виведення обладнання із ладу НС2	Травма Аварія	Заміна провідників, установлення захисного обладнання (запобіжників, захист від ураження людини струмом) тощо

Модель процесу:



Посвідчення про перевірку знань працівника є документом, який засвідчує право на самостійну роботу в електроустановках на зазначеній посаді за фахом.

4.2. Вимоги техніки безпеки під час роботи обладнання та протипожежні заходи

Вимоги правил техніки безпеки перед початком роботи. Для початку роботи пов’язаної з вентиляцією вимикають рубильники або автоматичні вимикачі щита низької напруги, запирають шафу і вивішують попереджувальні плакати. Також повинні бути основні захисні засоби до яких належать такі, ізоляція яких надійно захищає від робочої напруги мережі і за допомогою яких можна дотикатися до струмопровідних частин, що перебувають під напругою, без небезпеки ураження електричним струмом (інструмент з ізольованими ручками, ізолюючі струмовимірювальні кліщі, діелектричні рукавиці).

Вимоги правил техніки безпеки під час роботи. Виконавши ці операції, надівають діелектричні рукавиці і за допомогою покажчика напруги перевіряють відсутність напруги на всіх фазах. Потім, приєднавши один кінець переносного заземлення до заземлюючого пристрою, накладають його на струмоведучі частини. Після цього остаточно приступають до роботи.

Вимоги правил техніки після закінчення роботи. Після закінчення роботи системи перед її вимиканням необхідно виконати такі технічні операції: перевірити надійність кріплення, зняти переносні тимчасові заземлення, відімкнути щит низької напруги і зняти плакати з техніки безпеки; якщо тимчасове переносне заземлення встановлене на лінії, його також треба зняти тощо [12].

Протипожежні заходи на об’єкті. Для запобігання пожеж на об’єкті розроблено організаційні, експлуатаційні, технічні режимного характеру, пожежно-евакуаційні, профілактичні заходи. До організаційних заходів

відносяться правила розміщення машин, що обслуговують приміщення, обладнання, матеріалів з дотримання певних проходів, не допускається захаращення приміщень, проходів і т.д.

4.3. Розрахунок штучного заземлення

Вибір штучного заземлення проводиться в залежності від характеру ґрунту і способу забивання стержнів [21]. Розраховуємо заземлюючий контур підстанції напругою 10/0,4 кВ з глухозаземленою нейтраллю. Характер ґрунту – чорнозем з $\rho=2\cdot10^4$ Ом·см. Кліматична зона – IV (K_c – 1,2, K_n – 1,5). Струм замикання на землю в мережі становить 50 А.

В відповідності з діючими правилами, опір заземлюючого пристрою повинен становити

$$R = \frac{125}{I_з} = \frac{125}{50} = 2,5 Ом , \qquad (4.1)$$

де $I_з$ – струм замикання на землю, А.

Приймаємо 3 Ом. Контур заземлення розміщуємо в ряд з а = 5 м, l = 2,5 м. В якості стержневого заземлювача приймаємо кутникові сталь 50х50х5 мм, а протяжного – пластинчасту сталь 40х4 мм.

Опір одиночного стержня становить:

$$R_o = 0.00318\rho \cdot K_c, Ом \qquad (4.2)$$

де K_c – коефіцієнт сезонності для стержневого заземлювача (K_c – 1,2).

$$R_o = 0.00318 \cdot 2 \cdot 10^4 \cdot 1.2 = 76.32 Ом .$$

Число стержнів приймаємо 15. При цьому коефіцієнт використання стержневих заземлювачів становить η_c = 0,7. Опір всіх стержнів розтікання струму становить:

$$R_c = \frac{R_o}{n \cdot \eta_c}, Ом , \qquad (4.3)$$

де *n* – число стержнів, шт.

$$Rc = \frac{76.32}{15 \cdot 0.7} = 7.3Ом .$$

Довжина протяжного заземлювача становить *l* = 35 м (3500 см); приймаємо *t* = 50 см, *b* = 0,4 см. Опір протяжного заземлювача становить:

$$R_{пр} = \frac{0,366}{l} \cdot \rho \cdot 2 \cdot \lg \frac{2 \cdot l^2}{t \cdot b}, Ом \qquad (4.4)$$

$$R_{пр} = \frac{0,366}{3500} \cdot 1,2 \cdot 10^4 \cdot 2 \cdot \lg \frac{2 \cdot 3500^2}{0,4 \cdot 50} = 3,2Ом$$

Коефіцієнт використання протяжного заземлювача η_n = 0,71. Дійсний опір протяжного заземлення становить:

$$R_n = \frac{R_{пр}}{\eta_n} = \frac{3,2}{0,71} = 4,5Ом \qquad (4.5)$$

Опір всього заземлюю чого пристрою становить:

$$R_u = \frac{R_c \cdot R_n}{R_c + R_n} = \frac{4.5 \cdot 7.3}{4.5 + 7.3} 2.78 < 3Ом \qquad (4.6)$$

Отже, число стержнів вибрано вірно.

4.4. Захист цивільного населення

Забезпечення захисту населення і території у разі загрози та виникнення надзвичайних ситуацій є одним з найважливіших завдань не лише підприємства, але й цілої держави. Актуальність проблеми забезпечення природо-техногенної безпеки населення і території зумовлена тенденціями зростання втрат людей і шкоди територіям, що спричиняються небезпечними природними явищами, промисловими аваріями і катастрофами.

Інженерний захист проводиться з метою виконання вимог ІТЗ із питань забудови міст, розміщення ПНО, будівлі будинків, інженерних споруд та інше.

Медичний захист проводиться для зменшення ступеня ураження людей, своєчасного надання допомоги постраждалим та їх лікування, забезпечення епідеміологічного благополуччя в районах надзвичайних ситуацій.

Біологічний захист включає своєчасне виявлення чинників біологічного зараження, їх характеру і масштабів, проведення комплексу адміністративно-господарських, режимно-обмежувальних і спеціальних протиепідемічних та медичних заходів.

Радіаційний і хімічний захист включає заходи щодо виявлення і оцінки радіаційної та хімічної обстановки, організацію і здійснення дозиметричного та хімічного контролю, розроблення типових режимів радіаційного захисту, забезпечення засобами індивідуального захисту, організацію і проведення спеціальної обробки.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

Швидкий розвиток інтелектуального аналізу даних і машинного навчання суттєво змінив різні сфери, включно з маркетингом, завдяки більш ефективним і теоретично обґрунтованим методам. Системи персоналізованих рекомендацій відіграють вирішальну роль у сучасних бізнес-моделях, підвищуючи залучення користувачів і збільшуючи дохід за допомогою адаптованого контенту, наприклад продуктів, відео, музики та взаємодії в соціальних мережах. Ці системи покладаються на явні та неявні відгуки користувачів для створення рейтингів інтересів, а передові алгоритми, такі як спільна фільтрація та нейронні мережі, вирішують такі проблеми, як проблема холодного запуску. Провідні платформи, такі як Netflix, Amazon і Spotify, є прикладом різноманітних підходів і постійного вдосконалення технологій рекомендацій, які життєво важливі для підтримки конкурентоспроможності та задоволення потреб користувачів у сучасному середовищі, керованому даними.

Здійснено грунтовний огляд методів спільної фільтрації, що використовуються в системах рекомендацій, розрізняючи алгоритми на основі пам’яті та на основі моделі. Підходи, засновані на пам’яті, покладаються на обчислення подібності користувачів або елементів із такими методами, як фільтрація на основі користувачів, яка нормалізує оцінки та вимірює подібність за допомогою різних показників, і фільтрація на основі функцій, яка порівнює вектори ознак. Алгоритми, засновані на моделях, усувають обмеження методів, що потребують інтенсивної пам’яті, використовуючи такі методи, як кластеризація, матрична декомпозиція та моделі латентних факторів для покращення масштабованості та ефективності.

У рамках проведеної роботи означено загальні виклики створення систем рекомендацій, такі як розрідженість даних, холодний запуск, масштабованість і надмірна спеціалізація, а також стратегії пом’якшення цих проблем. Крім того, у ньому наголошується на використанні Python та його бібліотек, таких як Numpy,

Pandas, Scikit-learn і LightGBM, для розробки систем рекомендацій, підкреслюючи їхню роль в обробці даних, візуалізації та машинному навчанні.

Здійснено розробку та реалізацію системи рекомендацій для вибору сезонного одягу з використанням діаграми BPMN для моделювання бізнес-процесів, таких як обробка даних, аналіз і формування рекомендацій. Він використовує повний набір даних H&M, що включає деталі продукту, інформацію про клієнта та записи транзакцій, а також зображення продуктів, відсортовані за категоріями.

Система використовує двоетапний підхід, поєднуючи фільтрацію кандидатів на основі евристики та моделей неявної взаємодії, як-от iALS, із моделлю рейтингу, що підвищує градієнт, для створення персоналізованих пропозицій щодо продукту, оптимізованих за допомогою модульної обробки даних і гнучкої архітектури програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. H&M Personalized Fashion Recommendations [Електронний ресурс]. – 2022. Режим доступу до ресурсу: https://www.kaggle.com/competitions/h-and-m-personalized-fashion- recommendations/overview
2. J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, and D. Sampath. Fourth ACM Conference on Recommender Systems, RecSys, 2010, С.295–296.
3. Dokyun Lee , Kartik Hosanagar. How Do Product Attributes and Reviews Moderate the Impact of Recommender Systems Through Purchase Stages? Management Science, 2021. Vol. 67, No. 1.
4. Ian MacKenzie, Chris Meyer, and Steve Noble. How retailers can keep up with consumers [Електронний ресурс]. – 2012. – Режим доступу до ресурсу: https://www.mckinsey.com/industries/retail/our- insights/how-retailers-can-keep-up-with-consumers
5. Gabrielle Wright. Personalized marketing in a competitive environment for brands and e-commerce retailers [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.smartinsights.com/ecommerce/consumers-personalized-marketing-engagement/
6. Nicolas Gillis, François Glineur. Accelerated Multiplicative Updates and Hierarchical ALS Algorithms for Nonnegative Matrix Factorization. Neural Computation, 2012. 24(4), pp. 1085-1105.
7. Jianfeng Gao, Hamid Palangi. Learning Deep Structured Semantic Models for Web Search using Clickthrough Data [Електронний ресурс]. – 2013. – Режим доступу до ресурсу: https://www.microsoft.com/en-us/research/publication/learning-deep- structured-semantic-models-for-web-search-using-clickthrough-data/
8. Mark Lutz's. Learning Python, 5th Edition. O'Reilly Media, Inc. 2013.
9. Matevž Kunaver, Tomaž Požrl. Diversity in recommender systems – A survey. Elsevier, 2017. С 154-162.

10. Dominik Kowald, Emanuel Lacic. Popularity Bias in Collaborative Filtering-Based Multimedia Recommender Systems. BIAS Workshop at ECIR. 2022.
11. Пістун І. П., Тимочко В.О., Городецький І. М.. Березовецький А. П. Охорона праці (гігієна праці та виробнича санітарія): навч. посібн. / за ред. І.П. Пістуна. Ч. І. Львів: Тріада плюс, 2011. 224 с.
12. Пістун І. П., Тимочко В.О., Городецький І. М.. Березовецький А. П. Охорона праці (гігієна праці та виробнича санітарія): навч. посібн. / за ред. І.П. Пістуна. Ч. ІІ. Львів: Тріада плюс, 2011. 224 с.

ДОДАТКИ

Фрагмент коду програми

```
class DataLoader():

    def __init__(self, src_dir, raw_dir):

        """
        Params
        ----------
        src_dir : str
            Base directory.
        raw_dir : str
           Raw data directory.
        """

        self._base = Path(src_dir)
        self._raw_dir = self._base / raw_dir


    def save(self, data, code):
        """
        Save data to parquet format

        Params
        ----------
        data : dict

        code : str
        """

        path = self._base / "preprocessed" / code

        if not os.path.exists(path):
            os.makedirs(path)

        data["users"].to_parquet(path / "users.pqt")
        data["items"].to_parquet(path / "items.pqt")
        data["transactions"].to_parquet(path / "transactions.pqt")

    def load(self, code):
        """
        Load data from parquet format.
        """

        path = self._base / "preprocessed" / code
        if not os.path.exists(path):
            raise OSError(f"{path} does not exist.")


        data = {}
        data["users"] = pd.read_parquet(path / "users.pqt")
        data["items"] = pd.read_parquet(path / "items.pqt")
        data["transactions"] = pd.read_parquet(path / "transactions.pqt")

        return data

    def _load_raw(self):
        """
        Load raw data

        Returns
        -------
        dict
            transformed data
        """

        ARTICLES = "articles.csv"
        CUSTOMERS = "customers.csv"
        TRANSACTIONS = "transactions_train.csv"


        articles = pd.read_csv(self._raw_dir / ARTICLES)
        customers = pd.read_csv(self._raw_dir / CUSTOMERS)
        transactions = pd.read_csv(self._raw_dir / TRANSACTIONS)
```

```
    return {
    "items": articles,
    "users": customers,
    "transactions": transactions
    }

def _ids_to_idxs(self, data, id_dir):
    """
    Convert ids to integers

    Params
    ----------
    data : dict
        Raw data dict
    id_dir : str
        Directory to store mappings

    Returns
    -------
    dict
        transformed data
    """

    items = data["items"]
    users = data["users"]
    transactions = data["transactions"]

    user_id_map = self._base / id_dir / "user_id_map.pkl"
    user_idx_map = self._base / id_dir / "user_idx_map.pkl"

    item_id_map = self._base / id_dir / "item_id_map.pkl"
    item_idx_map = self._base / id_dir / "item_idx_map.pkl"

    if not os.path.isdir(self._base / id_dir):
        os.makedirs(self._base / id_dir)

    id_maps = []
    for _dict, field, path_id in [
        (users, 'customer_id', user_id_map),
        (items, 'article_id', item_id_map)
    ]:

        if not os.path.exists(path_id):
            mapping = dict(zip(_dict[field], _dict.index + 1))
            pickle.dump(mapping, open(path_id, "wb"))
        else:
            mapping = pickle.load(open(path_id, "rb"))
        id_maps.append(mapping)


    idx_maps = []
    for _dict, field, path_idx in [
        (users, 'customer_id', user_idx_map),
        (items, 'article_id', item_idx_map)
    ]:

        if not os.path.exists(path_idx):
            mapping = dict(zip(_dict.index + 1, _dict[field]))
            pickle.dump(mapping, open(path_idx, "wb"))
        else:
            mapping = pickle.load(open(path_idx, "rb"))
        idx_maps.append(mapping)

    self.id_maps = id_maps
    self.idx_maps = idx_maps
```

```
        transactions["customer_id"] = transactions["customer_id"].map(id_maps[0])
        transactions["article_id"] = transactions["article_id"].map(id_maps[1])
        users["customer_id"] = users["customer_id"].map(id_maps[0])
        items["article_id"] = items["article_id"].map(id_maps[1])

        return {
        "users": users,
        "items": items,
        "transactions": transactions
        }

    def _get_base_features(self, data):

        """
        Base features engineering

        Params
        ----------
        data : dict

        Returns
        -------
        dict
            transformed data
        """

        items = data["items"]
        users = data["users"]
        transactions = data["transactions"]

        items = items.apply(self.gender_map, axis=1)
        items["season"] = 0
        items = items.apply(self.winter_season_map, axis=1)
        items = items.apply(self.summer_season_map, axis=1)

        transactions = pd.merge(
            transactions,
            items[["article_id", "gender", "product_type_name"]],
            on="article_id",
            how="left",
        )

        transactions_counts = transactions\
            .groupby(["customer_id"])\
            .size()\
            .reset_index(name="trans_count")

        sale_gender = (
            transactions.groupby(["customer_id", "gender"])
            .size()
            .reset_index(name="count")
        )

        sale_gender = sale_gender.merge(transactions_counts, on=["customer_id"], how="left")
        sale_gender["ratio"] = sale_gender["count"] / sale_gender["trans_count"]
        sale_gender = pd.pivot_table(
            sale_gender, values="ratio", index="customer_id", columns=["gender"]
        )

        sale_gender = sale_gender.reset_index()


        sale_gender["user_gender"] = 0
        sale_gender.loc[sale_gender[1] >= 0.8, "user_gender"] = 1  # * male
        sale_gender.loc[sale_gender[2] >= 0.8, "user_gender"] = 2  # * female

        users = users.merge(
            sale_gender[["customer_id", "user_gender"]], on="customer_id", how="left"
        )
        users["user_gender"] = users["user_gender"].fillna(0)
```

```python
@staticmethod
def gender_map(item):
    item['gender'] = 0

    if item["index_group_name"] == "Menswear":
        item['gender'] = 1
    elif item["index_group_name"] == "Ladieswear":
        item['gender'] = 2
    else:
        department_name = item["department_name"].lower()

        if "men" in department_name or "boy" in department_name:
            item['gender'] = 1

        elif "ladies" in department_name or "girl" in department_name:
            item['gender'] = 2

        elif item["product_type_name"] in articles.LADIES_TYPES:
            item['gender'] = 2

    return item

@staticmethod
def winter_season_map(item):
    if item["product_type_name"] in articles.SUMMER:
        item["season"] = 1

    return item

@staticmethod
def summer_season_map(item):
    if item["product_type_name"] in articles.WINTER:
        item["season"] = 2

    return item


def _transform_cols(self, data):

    """
    Transform types

    Params
    ----------
    data : dict

    Returns
    -------
    dict
        transformed data
    """

    users = data["users"]
    items = data["items"]
    transactions = data["transactions"]

    transactions["price"] = transactions["price"].astype("float32")
    transactions["sales_channel_id"] = transactions["sales_channel_id"].astype("int8")

    user_sparse_repr = [x for x in users.columns if x not in ["age", "user_gender"]]
    for repr in tqdm(
        [x for x in user_sparse_repr if x != "customer_id"], "Sparse Users encoding",
    ):
        label_encoder = LabelEncoder()
        users[repr] = label_encoder.fit_transform(users[repr].astype(str)) + 1
        users[repr] = users[repr].astype("int32")

    users["age"] = users["age"].fillna(0)
    users = users.fillna(-1)
    users["age"] = users["age"].astype("int8")
```

```python
class RuleCollector:
    """Collect retrieval candidates by rules"""

    def collect(
        self,
        week_num: int,
        trans_df: pd.DataFrame,
        customer_list: np.ndarray,
        rules: List,
        filters: List = [],
        min_pos_rate: float = 0.01,
        item_id: str = "article_id",
        norm: bool = True,
        norm_type: str = "quantile",
        compress=True,
    ) -> pd.DataFrame:

        customer_list = np.array(customer_list)

        # * prepare valid data to calculate positive rate of retrieved items
        start_date, end_date = get_start_end_date(week_num)
        mask = (start_date <= trans_df["t_dat"]) & (trans_df["t_dat"] < end_date)
        label = trans_df.loc[mask, ["customer_id", "article_id"]]
        label = label.drop_duplicates(["customer_id", "article_id"])
        label["label"] = 1

        # * Get items to be removed.
        rm_items = []
        for filter in filters:
            rm_items += filter.retrieve()

        # * Retrieve items for each user.
        pred_df = None
        print()
        for rule in tqdm(rules, "Retrieve items by rules", desc='Retrieving items by rules'):
            items = rule.retrieve()
            if norm:
                if norm_type == "quantile":
                    scaler = QuantileTransformer(output_distribution="normal")
                elif norm_type == "minmax":
                    scaler = MinMaxScaler()
                items["score"] = scaler.fit_transform(
                    items["score"].values.reshape(-1, 1)
                )
            items = items.loc[~items[item_id].isin(rm_items)]

            if label.shape[0] != 0:
                # * not test set, calculate the positive rate
                label_customers = label["customer_id"].unique()
                tmp_items = items[items["customer_id"].isin(label_customers)]
                tmp_items = tmp_items.merge(
                    label, on=["customer_id", "article_id"], how="left"
                )
                tmp_items["label"] = tmp_items["label"].fillna(0)
                pos_rate = tmp_items["label"].mean()

                # * if the positive rate is too low, we need to find the `n` that
                # * has the highest positive rate.
                if pos_rate < min_pos_rate:
                    tmp_items = tmp_items.sort_values(by="score", ascending=False)
                    tmp_items["rank"] = tmp_items.groupby("customer_id")["score"].rank(
                        ascending=False
                    )

                    rank = tmp_items["rank"].max()
                    best_pos_rate = pos_rate
                    best_rank = rank
                    while rank > 0:
                        tmp_pos_rate = tmp_items.loc[
                            tmp_items["rank"] <= rank, "label"
                        ].mean()
```

```python
def _ap_at_k(actual, predicted, k=10):
    if len(predicted) > k:
        predicted = predicted[:k]

    score = 0.0
    num_hits = 0.0

    for i, p in enumerate(predicted):
        if p in actual and p not in predicted[:i]:
            num_hits += 1.0
            score += num_hits / (i + 1.0)

    if actual is None:
        return 0.0

    return score / min(len(actual), k)


def _rk(actual, predicted, k=10):
    if len(predicted) > k:
        predicted = predicted[:k]

    score = sum([1 for r in actual if r in predicted]) / len(actual)

    return score


def map_at_k(actual: Iterable, predicted: Iterable, k: int = 12) -> float:

    return np.mean(
        [_ap_at_k(a, p, k) for a, p in zip(actual, predicted) if a is not None]
    )
```

```python
def train_rank_model(
    train,
    valid,
    train_group,
    valid_group,
    params,
    feats,
    cat_features,
    model_dir
):

    train_set = lgb.Dataset(
        data=train[feats],
        label=train["label"],
        group=train_group,
        feature_name=feats,
        categorical_feature=cat_features,
        params=params,
    )

    valid_set = lgb.Dataset(
        data=valid[feats],
        label=valid["label"],
        group=valid_group,
        feature_name=feats,
        categorical_feature=cat_features,
        params=params,
    )
```