МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

Бралатан Роман Андрійович

Допускається до захисту:
завідувач кафедри
інформаційних технологій,
д. т. н., доцент
_____ Т. В. Нескородєва
«_____» _____ 2022р.

Рекомендаційна система для користувачів інтернет-магазину на базі платформи Salesforce

Спеціальність 122 «Комп’ютерні науки»

Кваліфікаційна (магістерська) робота

Науковий керівник:
Баркалов О.О., професор
кафедри інформаційних технологій

_____
(підпис)

Оцінка: _____/_____/_____
(бали за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця – 2022

Зміст

ВСТУП

З давніх давен люди займалися й займаються торгівлею. Найголовніше для продавця це знайти підхід до клієнта, запропонувати йому такий товар, який прийдеться до вподоби. У минулому, коли спілкування проходило безпосередньо віч на віч, вся відповідальність лягала на плечі продавця, проте з плином часу і розвитком технологій виникла потреба у автоматизації.

Перша згадка про рекомендаційні системи була у 1990 році від шведського вченого Юссі Карлгрена, які він описав наступним чином “цифрова полиця для книг”. Найбільшої популярності такі системи почали набирати у 2000 роках, під час зародження та розвитку інтернет комерції. Першим великим та успішним брендом, що використав рекомендаційні системи був Amazon.

Нам усім відомі такі сервіси як Google, apple store. Spotify, YouTube, Netflix та багато інших, якими користуються міліони користувачів. На нашу думку, один з основних чинників їх популярності це їх система аналізу даних та рекомендаційна система. Адже набагато приємніше користуватися сервісом, який гарно розуміє ваші вподобання та швидко рекомендує вам саме той контент чи продукт, на який ви очікували. У час коли серед бізнесу ведеться величезна конкуренція, рекомендаційні системи все більше і більше розвиваються, адже вимоги користувачів до сервісу також постійно ростуть.

Наразі ми живемо у вік цифрових технологій, де набули популярності інтернет магазини та онлайн торгівля. Саме тому набули популярності й рекомендаційні системи, адже нам тепер не потрібні продавці та консультанти, система буде самостійно пропонувати покупцю ті товари, в яких він найбільше зацікавлений. Все що потрібно для продуктивної роботи рекомендаційної системи це обрати її алгоритм роботи та задати правильну стратегію.

Для досягнення більшого успіху у торгівлі, постійно йде робота над розвитком та удосконаленням рекомендацій. Загалом розрізняють такі основні види

рекомендаційних алгоритмів: алгоритм колаборативної фільтрації, контентної фільтрації та фільтрації на основі знань.

Кожен із перерахованих вище алгоритмів, які використовуються під час розробки рекомендаційних систем, має ряд своїх недоліків та переваг. Предметом дослідження даної роботи є аналіз уже існуючих методів та алгоритмів, для того щоб виявити їх сильні та слабкі сторони та побудувати власну систему з урахуванням отриманих та проаналізованих даних.

Метою дослідження є розробка удосконалення сучасних рекомендаційних алгоритмів для підвищення релевантності пропозицій та їх програмна реалізація з урахуванням особливостей системи Salesforse.

Для досягнення мети поставлено та вирішено такі завдання дослідження:

1) аналіз сучасного стану та тенденцій рекомендаційних систем;
2) модифікація item-item алгоритму рекомендацій;
3) створення власної системи збору та аналізу інформації про дії користувача;
4) модифікація алгоритму Thompson Sampling;
5) програмна реалізація рекомендаційної системи;

Новизна роботи полягає в тому, що запропонована рекомендаційна система об'єднує в собі декілька модернізованих алгоритмів, та власну модель збору та оцінки даних, розподілення їх по різним матрицям. Запропоновано модернізований алгоритм item-item, до якого додано операцію розбиття усього масиву на масиви по 3 елемента, з метою подальшого їх використання у рекомендаціях для збільшення їх варіативності. Також розроблено модернізовано алгоритм Tompson Sampling, додано нижню межу для коефіцієнту успішності продажів, що були введені для матриць, з метою уникнути втрачання показу рекомендацій з використанням матриць що не користуються успішністю.

Створена рекомендаційна система вирішує ряд недоліків, що присутні у її аналогів. Першим вирішеним критичним недоліком є проблема “Холодного старту”, притаманна алгоритмам колаборативної фільтрації, за рахунок створення нових матриць та створеної моделі оцінювання продуктів. Наступна проблема, що була вирішена це доцільність та варіативність рекомендацій. Її вирішення полягає у модернізації item-item та Tompson sampling алгоритмів. Також була вирішена проблема залежності рекомендацій від дій інших користувачів.

За матеріалами дослідження опубліковано тези [23] та зроблено доповідь на конференцію КТОД 2022.

Структура роботи: робота складається зі Вступу, 4 розділів, Висновків до кожного розділу. Список літературних джерел налічує 22 позиції; робота виконана на 59 сторінках, вона містить 41 рисунок та 1 таблицю.

РОЗДІЛ 1 ОГЛЯД СТАНУ ПИТАННЯ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Рекомендаційні системи як об’єкт автоматизації

Рекомендаційна система – це алгоритми, що пропонують різні товари та послуги на основі отриманих та проаналізованих даних про користувача [1].

Для інтернет магазинів рекомендаційна система це вагома складова для їх успішної роботи. Якщо ресурс містить велику кількість товарів, користувач просто може не обрати річ яка йому по смаку та не знайти те що він шукав. Якщо ж користувачеві одразу запропонують товари які користуються популярністю у інших користувачів, цікаві новинки, або запитають що саме він шукав і запропонують йому товари оглядаючись на його уподобання, то шанс що покупець знайде для себе щось цікаве значно підвищується, а отже і те що він продовжить користуватися даним сервісом (див. рис. 1.1).

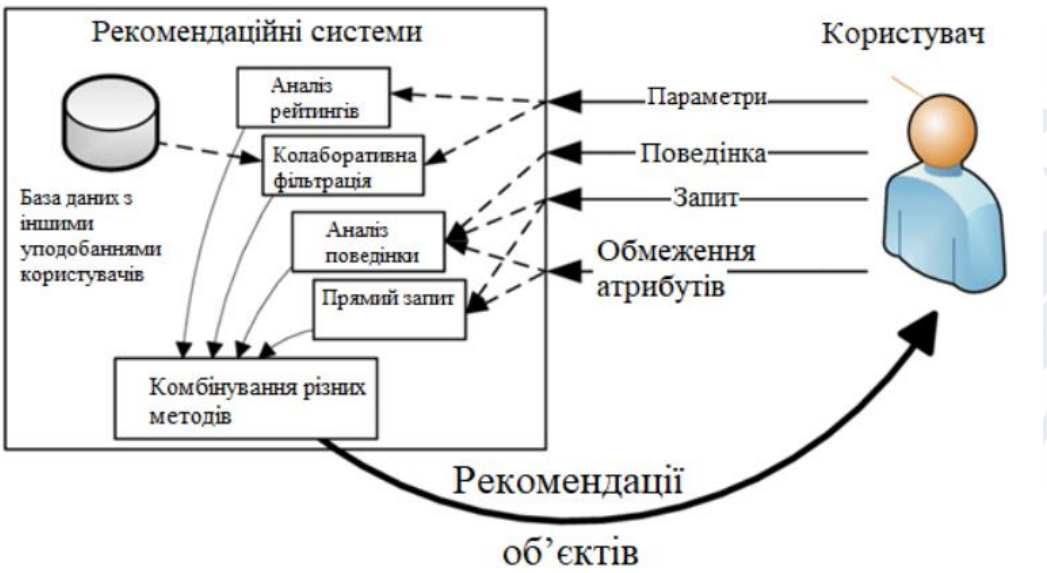


Рис. 1.1. Схема взаємодії користувача з рекомендаційною системою

Основним елементом побудови рекомендаційних систем є машинне навчання [2], яке використовується для того, щоб не обмежувати функціонування системи лише людським фактором. Машинне навчання сприяє постійному розвитку системи та постійному вдосконаленню. Перш за все при розробці таких систем потрібно чітко розуміти з якими саме даними буде працювати система та властивості цих даних. В залежності від бізнес

задачі, розробнику потрібно обрати найбільш оптимальний алгоритм за допомогою якого можна буде налаштувати найбільш ефективну взаємодію між користувачем та системою.

Для автоматизації рекомендаційних систем існує два основних методи машинного навчання (див. рис. 1.2):

1. Машинне навчання з вчителем (контрольоване).
2. Машинне навчання без вчителя (неконтрольоване).

Рис. 1.2. Схема видів машинного навчання

Контрольований підхід працює за наступною формулою: y = f(X), де (X) це вхідні значення до яких застосовується функція (f) – алгоритм машинного навчання який обрав розробник. Головне у навчанні з вчителем це те, що при використанні нових вхідних даних ми можемо передбачувати вихідні змінні. Тобто нам одразу відома правильна відповідь, яку нам має повернути алгоритм, залишилось лише навчити його її отримувати з наданої інформації.

Під час контрольованого машинного навчання використовуються наступні алгоритми, які допомагають з вирішенням такої проблеми як прогнозування релевантних рекомендацій користувачу:

- Алгоритм класифікації [3].
- Алгоритм регресії [4].

У свою чергу неконтрольований підхід машинного навчання використовується у тому випадку, коли вхідні дані відомі (X), а вихідні дані навпаки невідомі (Y). У такому випадку головною метою буде поділ усіх даних на групи за певними ознаками. Таким чином у результаті роботи машинного навчання без учителя буде знайдено залежності між даними та побудована їх структура на основі знайдених залежностей. Для покращення та підняття точності роботи неконтрольованого підходу потрібно самостійно задати тему роботи за допомогою алгоритму, самостійно зазначити ряд залежностей у вхідних даних для того об алгоритм їх використовував у подальшому для прийняття рішень. Таким чином машинне навчання без учителя працює за принципом шаблонів, порівнюючи усі дані, та виявляючи розбіжності між ними.

1.2 Формалізація задачі

Розглянувши різні моделі рекомендаційних систем, було прийняте рішення створити свою, більш оптимізовану систему, яка буде зручніша як для користувача так і для бізнесу. Буде створена наступна рекомендаційна система, для інтернет-магазинів, яка буде основана на існуючих методах та алгоритмах рекомендаційних систем та буде працювати на основі методу багаторуких бандитів [5].

Система буде інтегрована у інтернет магазин, який має свою базу продуктів та користувачів. Продукти мають різний тип, параметри, відносяться до різних категорій. Також буде діяти система знижок на продукти та їх новизна. Усі користувачі будуть поділятися на два типи:

- Зареєстровані
- Незареєстровані

Для незареєстрованих користувачів буде вестись окремий аналіз їх дій на сайті. Тобто будуть аналізуватися наступні дії користувача:

- перехід на конкретну сторінку з продуктом;
- додавання продукту до корзини;
- купівля продукту;
- рейтинг (оцінка) продукту користувачами сайту.

Дії користувачів будуть проаналізовані та записані у вигляді оцінок кожного продукту до матриці. Таким чином буде сформована загальна база продуктів посортованих за їх популярністю, знижками та новизною, які будуть відображатися усім незареєстрованим користувачам.

З іншого боку для зареєстрованих користувачів буде вестись індивідуальний збір та аналіз їх дій на сайті. Кожен користувач буде мати власну базу, у якій будуть записувати його уподобання. Категорії, на які він найчастіше переходить, ключові слова по яким він шукає продукти або категорії, на які конкретні продукти він переходить та що добавляє у корзину та врешті решт купує. Таким чином буде індивідуальний підхід до кожного користувача.

Для того щоб вирішити проблему “холодного старту” [6], коли користувач тільки зареєструвався на сайті та про нього ще не зібрано жодної інформації, планується об’єднати ці дві бази зібраних та проаналізованих знань під час показу рекомендацій. Тобто за допомогою методу багаторуких бандитів ми будемо брати дані з загального масиву знань для незареєстрованих користувачів та поміщати продукти з неї у рекомендаційну карусель на сайті, а також по мірі наповнення індивідуальної бази користувача, використовувати інформацію і звідти. Таким чином в першу чергу для щойно зареєстрованих користувачів спочатку буде відображатися загальна інформація, яка буде замінятися поступово на більш індивідуальну, в залежності від поведінки та уподобань користувача. Коефіцієнт відображення

індивідуальних або загальних рекомендацій буде напряму залежати від успішності продаж тих чи інших продуктів для даного користувача.

Для більш точного аналізу, рекомендаційний рейтинг продуктів буде складатися не лише з позитивних сценаріїв, але й з негативних. У користувачів буде можливість оцінювати продукти по шкалі від одного до п’яти, де п’ять це найвища оцінка. Ці оцінки також будуть впливати на рейтинг продукту.

Власник інтернет магазину, або його маркетингова команда буде мати можливість додати свої продукти, які вони забажають, до третього масиву даних, з метою їх подальшої рекомендації. Таким чином вирішиться питання бізнесу, коли певні продукти які потребують розкрутки за тією чи іншою причиною, опиняться у рекомендаціях. У налаштуваннях Бізнес-менеджера, який надає Salesforce [7], можна буде налаштувати частоту з якою будуть відображатися продукти з даного масиву даних.

1.3 Аналіз методів вирішення задачі

1.3.1 Алгоритм колаборативної фільтрації

Колаборативна фільтрація [8] широко використовується не в останню чергу через відносну простоту реалізації. Принцип її роботи і справді нескладний, хоча він може поділятися на два різні підходи.

Підхід, заснований на зіставленні користувачів (user-based) [9], бере до уваги схожість заданого користувача на інших користувачів, задіяних у системі. Наприклад, якщо користувач позитивно оцінив продукт з конкретної категорії, то іншому користувачу, який шукає схожий товар можна його запропонувати.

Концепція зіставлення об'єктів (item-based) [10], навпаки, аналізує самі об'єкти і виявляє їх схожість. Ті об’єкти, які користувачу колись сподобалися, або викликали більше всього інтересу. На практиці це виглядає так –

користувачу подобалися телефони конкретної марки, чому б йому не запропонувати відповідну модель телефону?

Колаборативна фільтрація дозволяє отримувати дуже точні та доречні рекомендації, засновані на аналізі та зіставленні відмінностей у користувачів зі схожою поведінкою.

Тобто алгоритми колаборативної фільтрації працюють на основі інформації зібраної при взаємодії користувачів з контентом та послугами, які надає сервіс. Такі алгоритми використовують user-item матрицю [11], комірки якої заповнюють самі користувачі у процесі користування інформаційною системою.

Недоліки:

- Користувачі не в захваті від користування системою відгуків, тому її використовують і заповнюють дуже малий відсоток відвідувачів. Через це user-item матриці містять мало інформації, що впливає на точність та доречність рекомендацій.
- Проблема “холодного старту”. Усі нові користувачі та контент доданий до системи не надають зовсім ніякої інформації про себе, тому вони зазвичай втрачаються у потоці інформації.

1.3.2 Контентна фільтрація

Контентна фільтрація [12] вибудовує внутрішні зв'язки між запропонованими товарами чи будь-яким контентом. Цей простий принцип проявляється у рекомендації користувачеві об'єктів, схожих на ті, що він вибрав раніше. Наприклад, якщо в книгарні ви купуєте посібник з гри на гітарі, вам автоматично будуть запропоновані інші популярні самовчителі або посібники того ж автора. Великий плюс рекомендаційних систем, що використовують принцип контентної фільтрації, – можливість зацікавити нового користувача пропозиціями буквально з його перших споживчих кроків. Вам не потрібно довгий час збирати дані про переваги людини, ви можете

одразу включити відвідувача в роботу з ресурсом. Також важлива перевага контентної фільтрації полягає в можливості рекомендувати користувачеві ті об'єкти, які не оцінили та обійшли стороною інші користувачі. Останній момент часто має місце при використанні колаборативного методу.

Контентна фільтрація повністю ігнорує думки користувачів щодо тих чи інших об'єктів. Вибудовуючи зв'язки суто між самими об'єктами, ми маємо можливість миттєво, без збору оцінок та додаткових персональних відомостей, запропонувати людині щось схоже на ту позицію, яка її зацікавила. Виключаючи досвід користувача з рекомендаційної системи як основну субстанцію, ми начебто вирішуємо проблему холодного старту, коли розрідженість даних про користувача заважає системі виробляти персоналізовані рекомендації. Проте зворотний бік контентної фільтрації полягає у зовсім недоречних, а часом і просто безглуздих рекомендаціях.

Інша складність, пов'язана з використанням принципу контентної фільтрації – значний обсяг роботи з розбудови звязків між усіма об'єктами, що знаходяться в системі. Але найголовніший недолік цього методу виявляється у досить низькому, а часом досить умовному попаданні в ціль. Контентна фільтрація не передбачає високого ступеня персоналізації, тому точність рекомендацій невелика.

1.3.3 Фільтрація на основі знань

Системи цього типу знаходять широке застосування в онлайн-магазинах. По суті, рекомендації, що ґрунтуються на знаннях [13], схожі з попереднім методом контентної фільтрації, проте такі алгоритми використовують глибший аналіз об'єктів, вибудовуючи зв'язки між ними не за банальними критеріями схожості, а виходячи із взаємопов'язаності тих чи інших груп товарів.

На практиці це виглядає в такий спосіб – придбавши, наприклад, смартфон, сайт пропонує вам аксесуари, які підходять для використання разом із вашим новим пристроєм. Це можуть бути чохли, навушники, карти пам'яті і

таке інше. Додатково простимулювати покупця можна за допомогою знижки на аксесуари, які можуть бути дуже доречними у зв'язку з придбанням нового девайсу.

Рекомендації, що ґрунтуються на знаннях, демонструють непогані результати, піднімаючи оборот великих мережевих торгових майданчиків на десятки відсотків. Крім того, на відміну від контентної фільтрації, цей тип рекомендацій має високу точність, пропонуючи користувачеві те, що йому дійсно може стати в нагоді.

Якщо вас цікавлять точні рекомендації, то вам безперечно варто подумати над впровадженням [14] системи у своєму ресурсі. Як і контентна фільтрація, рекомендаційна система, заснована на знаннях, вивчає та аналізує взаємозв'язки між об'єктами (товарами), але, крім того, вона враховує ряд додаткових опцій, що належать до індивідуальних властивостей конкретного користувача.

Побажання користувача. Знайома всім ситуація – сайт пропонує користувачеві вказати бажані характеристики, після чого пропонує товари, що підходять під запит.

Демографічні особливості. Власне, демографічні дані для вироблення рекомендацій використовують найбільші соціальні мережі, такі як Facebook, LinkedIn, Google та інші.

Звичайно, для реалізації такої системи потрібно добре попрацювати - вам доведеться зібрати і обробити величезний масив даних, що сильно відображається на швидкодії системи.

1.3.4 Аналіз існуючих алгоритмів

Проаналізувавши три основних алгоритми (див. рис. 3.1), що застосовуються під час розробки рекомендаційних систем, можна зробити наступні висновки:

- Кожен алгоритм має свої переваги та недоліки;
- Потрібно ставити чіткі бізнес-задачі, обирати алгоритм виходячи із його ефективності відносно поставлених задач, яі він має виконувати;
- Рекомендаційні алгоритми постійно розвиваються та вдосконалюються, разом із виникаючими новими потребами користувачів та власників бізнесу.

Було виявлено п’ять основних проблем та критеріїв, за якими буде надана оцінка для кожного розглянутого алгоритму у таблиці 1.1.

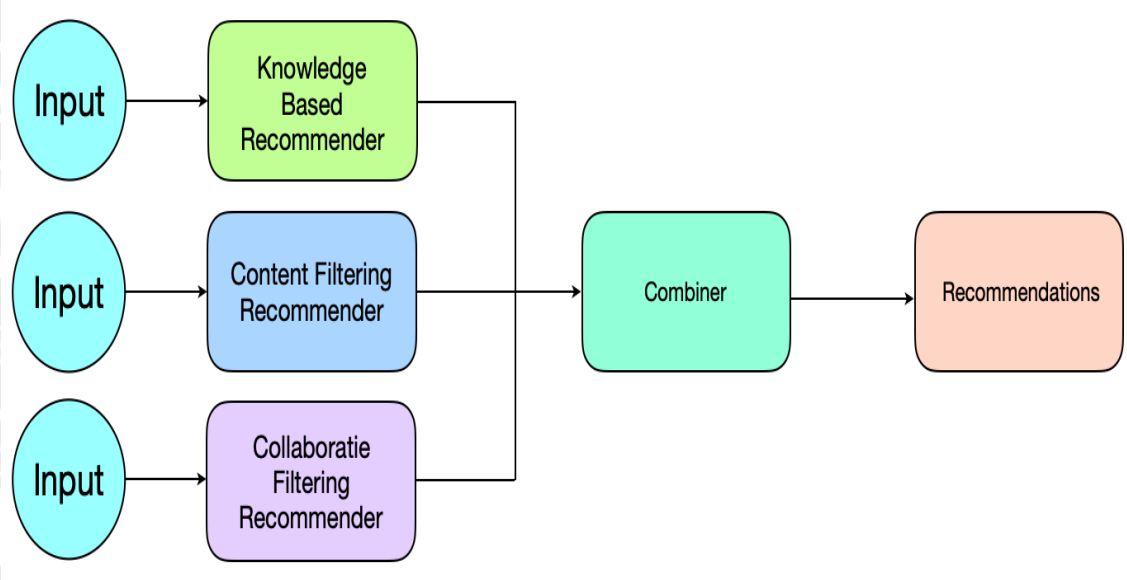


Рис. 1.3. Схема трьох основних алгоритмів рекомендаційних систем

Таблиця 1.1 – Оцінка алгоритмів надання рекомендацій

	Контентна фільтрація	Колаборативна фільтрація	Фільтрація на знаннях
Проблема холодного старту	Відсутня	Присутня	Відсутня
Складність реалізації	Низька	Низька	Середня
Точність рекомендацій	Середня	Середня	Висока
Швидкість	Висока	Середня	Середня
Залежність від інших користувачів	Відсутня	Присутня	Відсутня

1.4 Деталізація завдань дослідження

На основі проаналізованих методів створення рекомендаційних систем, виділено ряд задач, які потрібно вирішити.

Перш за все, потрібно вирішити проблему холодного старту, притаманну для алгоритмів колаборативної фільтрації. Потрібно дослідити поведінку системи для нових користувачів та запропонувати їм найбільш успішні продукти на основі отриманих даних від інших користувачів.

Забезпечити високу швидкість роботи системи завдяки розбиттю матриці з інформацією на три окремі матриці, що будуть обчислюватися паралельно в залежності від статусу (зареєстрований та незареєстрований) користувача. Таким чином вийде значно розвантажити навантаження на систему за рахунок зменшення розмірності матриць, та зробити їх обчислення паралельними, що також підвищить швидкодію.

Завдяки створенню та введенню матриці, яку можна наповнювати вручну, ми позбавимося від проблеми залежності рекомендацій від дій інших

користувачів, адже якщо уявити що у нас ще не зібрано жодної інформації на сайті, ми завжди зможемо запропонувати користувачу товари, які забажає власник інтернет-магазину, або маркетингова команда.

Наступна проблема, яка буде вирішена це точність рекомендацій. За допомогою алгоритму багаторуких бандитів буде оцінюватись якість запропонованих товарів користувачеві та успіх, яким вони користуються у конкретного користувача. Таким чином карусель рекомендацій буде постійно адаптуватись під користувача, пропонуючи йому товари в більшій мірі з тієї матриці даних, яка приносить найбільший успіх (продаж товарів).

1.5 Висновки за розділом 1

Швидкий розвиток інтернет торгівлі, та все більша її популяризація робить дослідження та розробку рекомендаційних систем актуальною задачею сьогодення. Існуючі методи та системи недосконалі, тому ми можемо виділити наступні основні проблеми, які потребують вирішення:

- Проблема холодного старту;
- Доцільність рекомендацій;
- Залежність від інших користувачів

Отримані результати дослідження підтвердили, що є сенс у розробці нових та адаптації і удосконаленню уже існуючих методів та алгоритмів, що використовуються у рекомендаційних системах.

Сформовано основні задачі, що будуть розглянуті у ході даної роботи, рішення яких дозволить підняти рівень продуктивності рекомендацій, принесе користь багатьом власникам комерційних проектів та буде містити основну

РОЗДІЛ 2 МОДЕЛІ ТА АЛГОРИТМИ

2.1 Модифікація Item-Item алгоритму

В основі роботи цього алгоритму лежить принцип використання user-item матриці. Ця матриця формується за таким принципом: у її рядках записуються користувачі системи, а у стовпцях продукти та категорії продуктів. Пересічення рядків та стовпців матриці містить оцінку, яку продукт отримав в результаті певних дій користувача. Побудувати таку матрицю можна за допомогою двох методів збору інформації:

- Явний.
- Неявний.

Явний спосіб [15] полягає у тому, що користувачеві пропонується залишити відгук, оцінку або анкету про даний продукт. Не всі користувачі у захваті від такої перспективи, тому у даній роботі було прийняте рішення використовувати неявний збір інформації [16]. Тобто система буде сама відслідковувати та аналізувати дії виконані покупцем на сайті. Таким чином буде зібрано більше інформації, що в свою чергу дасть більшу точність рекомендацій.

Після формування user-item матриці потрібно обрати одну з мір схожості, за якими будуть оцінюватися продукти та послуги. Найпоширенішими є наступні міри:

- Коефіцієнт кореляції Пірсона
- Коефіцієнт Тахімото
- Косинусна міра
- Евклідова відстань

Обирати один з запропонованих варіантів потрібно лише виходячи із формату вхідних даних, який ми будемо оцінювати, на роботу алгоритму рекомендації це ніяк не впливає.

Після того як буде проведена оцінка схожості продуктів та послуг буде обрахована вихідна оцінка. Вираховуватись вона буде за наступною формулою:

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

У даній формулі:

- $r_{u,i}$ – прогнозована оцінка.
- $r_{u',i}$ – відома оцінка схожого продукту або послуги.
- $sim(u, u')$ – ступінь подібності продукту або послуги.

У результаті роботи item-item алгоритм [17] поверне перелік продуктів та послуг посортованих за ступенем схожості (див. рис. 2.1). Для того щоб підвищити якість роботи алгоритму буде також враховуватися оцінка продуктів.

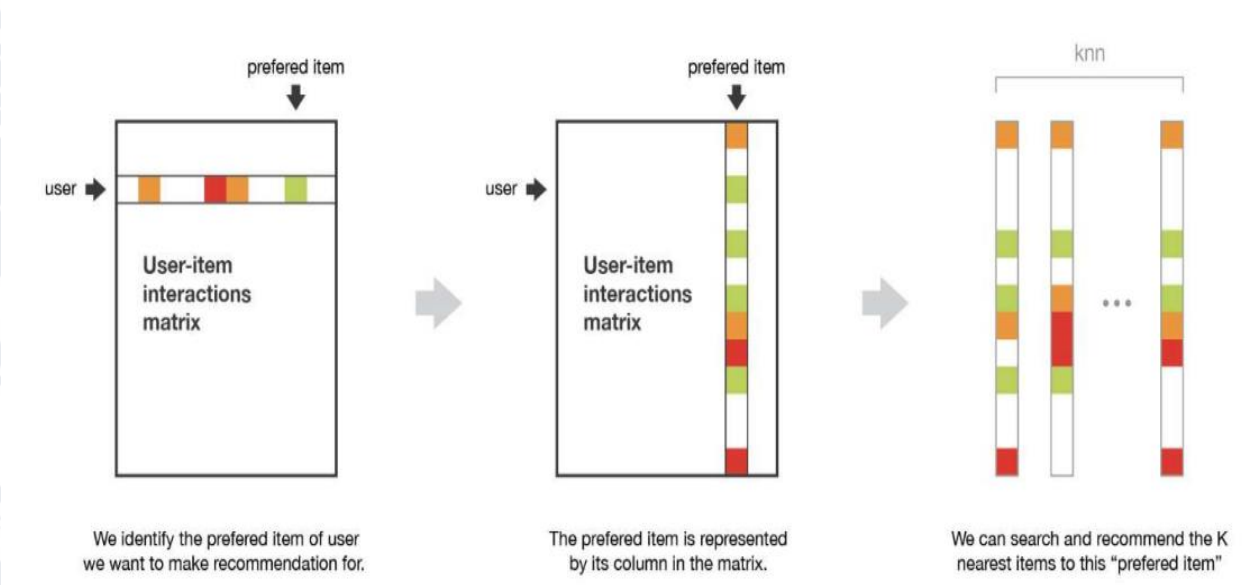


Рис. 2.1. Результат роботи item-item алгоритму

Для того, щоб уникнути повторів рекомендацій кожного разу, коли користувач перезавантажує сторінку, було прийнято рішення додати рандомізацію до уже сформованої матриці. Таким чином, кожного разу, item-item алгоритм буде розбивати масив отриманих посортованих елементів на

масиви що містять по три елементи, та обирати один випадковий продукт із кожного такого масиву, що буде відображено на рекомендаційній каруселі.

2.2 Модифікація алгоритму Thompson Sampling

Пошук буде вестись між трьома масивами даних з їх переліком продуктів. Кожен продукт буде оцінюватися алгоритмом наступним чином:

- Успішний (продався завдяки рекомендації – 1)
- Не успішний (не був проданий завдяки рекомендації – 0).

Таким чином за допомогою розподілу Бернуллі [18] ми можемо описати задачу наступним чином: нам потрібно знайти вибірку даних з трьох запропонованих вище, продукти з якої будуть мати найбільший успіх продаж.

Отже, в результаті роботи семплювання Томпсона [19] ми отримуємо модель ймовірностей винагороди. Коли доступність винагороди є бінарною (як у цьому випадку так чи ні), ідеальною моделлю для такої ймовірності є бета-розподіл [20]. Тому ми введемо два лічильники успіхів та невдач, α та β. Це будуть змінні, за допомогою яких буде вестись рахунок успіхів та невдач. У бета-розподілу є усереднене значення, яке обчислюється наступним чином:

$$\frac{\alpha}{\alpha + \beta} = \frac{\text{кількість успіхів}}{\text{кількість спроб}}$$

Нам потрібно знайти dataset з найбільшою ймовірністю успіху продажу. Спочатку ми не знаємо, яка ймовірність успіху продажу у конкретної вибірки продуктів, тому можна присвоїти α і β значення 1, що дасть нам рівномірний розподіл. Це початкове припущення називається апріорною ймовірністю: ймовірність виникнення якоїсь події до того, як ми отримали будь-які відомості про неї.

Перевіривши dataset на успішність, ми можемо скоригувати наші очікування на те, що він працює. Така нова ймовірність після отримання якихось відомостей називається апостеріорною ймовірністю [21]. Вона теж представлена бета-розподілом, але тепер ми оновили значення α і β на основі отриманої винагороди.

Якщо dataset успішний, то винагорода дорівнює 1, і альфа-лічильник успіхів – збільшується на 1. Лічильник невдач – не росте. Якщо ж ми не отримаємо винагороду, то α не зміниться, а β збільшиться на 1. Чим більше ми збираємо даних, тим сильніше бета-розподіл починає відрізнятися від прямої лінії і стає все більш точною моделлю ймовірності усередненої винагороди. Підтримуючи значення α і β, алгоритм семплювання Томпсона може описати очікувану середню винагороду та рівень його достовірності.

На відміну від жадібного алгоритму, у якому на кожному тимчасовому кроці обирають дію з найвищою очікуваною винагородою, навіть якщо достовірність очікування невисока, семплювання Томпсона бере зразки з бета-розподілу кожної дії, і вибирає дію з найвищим повернутим значенням. Оскільки дії, що рідко аналізуються, мають широкі розподіли, то у них більше діапазон можливих значень. Тому dataset, який має низьку очікувану середню винагороду, але який тестувався рідше за dataset з більш високим середнім очікуванням, може повернути більше вибіркового значення і буде обраний на наступному часовому кроці.

Кількість використання dataset збільшує достовірність його очікуваного середнього. Це відображено у звуженні розподілу ймовірності, і вибіркове значення буде братися з діапазону значень, які ближчі до реального середнього (див. зелену криву на рис 2.2). В результаті дослідження (explore) зменшується, а використання (exploit) зростає, тому що алгоритм починає частіше вибирати dataset з вищою ймовірністю отримання винагороди.

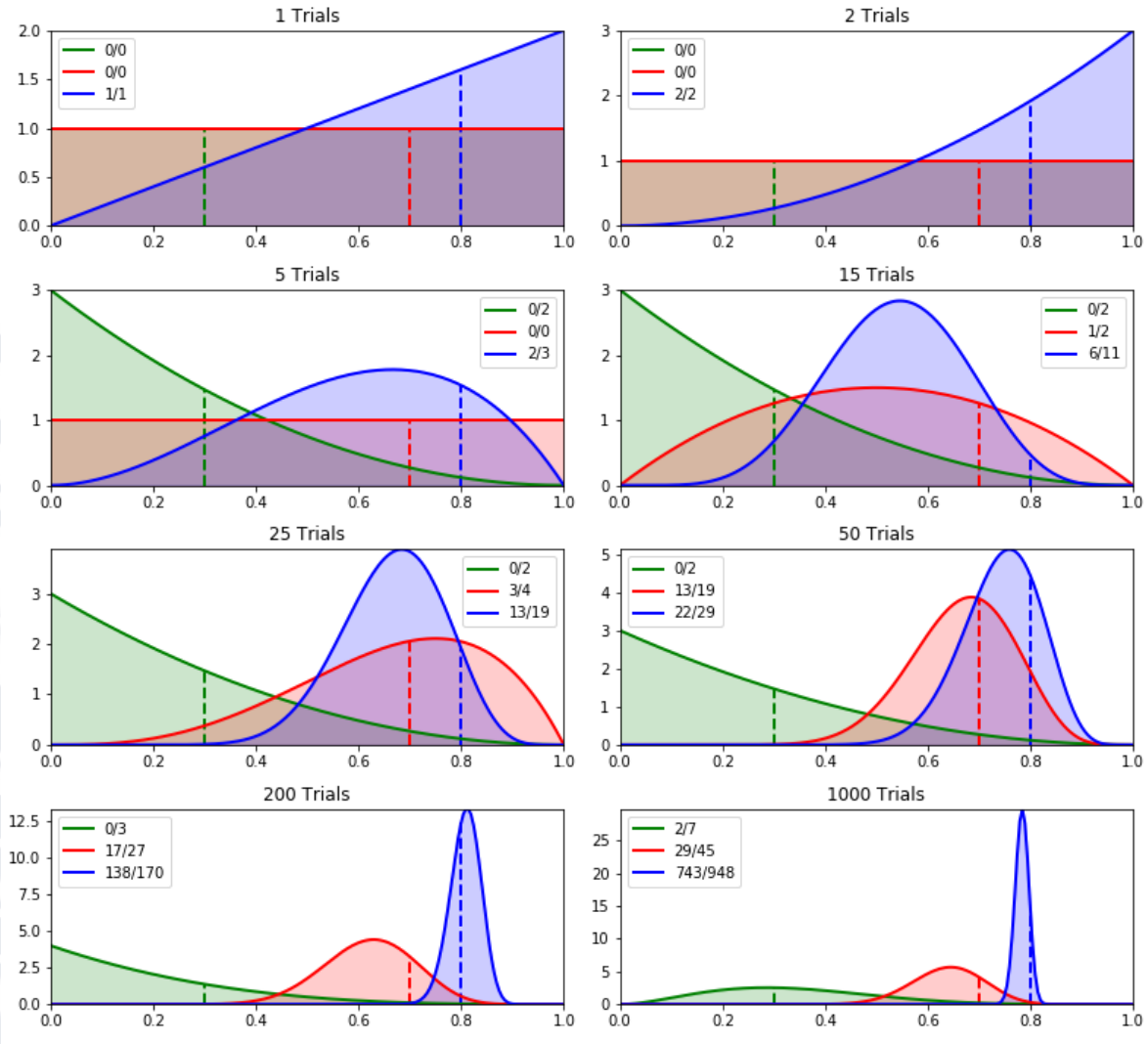


Рис. 2.2. Графіки проведених експериментів роботи з трьома масивами даних

Потрібно уникнути повного ігнорування одного з запропонованих масивів даних, для вирішення цієї задачі було прийняте рішення поставити нижню межу для коефіцієнту успішності кожної матриці, що буде рівна 0,3. Таким чином, ми залишаємо можливість залишатися продуктам з невдалої матриці на рекомендаційній каруселі, у меншій кількості, але з надією, що вона у перспективі зможе повернути свою успішність.

2.3 Модель збору та аналізу даних

Для роботи усієї рекомендаційної системи буде створено три матриці, заповнені отриманою або введеною самостійно інформацію (матриця, яку зможуть наповнювати власники та контент команда сайту).

Матриця, що стосується зареєстрованих користувачів буде заповнюватися за допомогою неявного способу отримання інформації. Для реалізації цього було прийнято рішення використовувати так звані події (actions). Таким чином можна буде класифікувати кожну дію користувача та присвоїти їй оцінку. Для вирішення конкретних задач створюваної рекомендаційної системи виділено наступні дії:

1) Відкриття сторінки конкретного товару (оцінка – 2)
2) Додавання товару до кошику (оцінка – 3)
3) Купівля товару (оцінка – 4)

Після того як користувач виконає одну із описаних дій, вона одразу буде записана до матриці з інформацією з відповідною оцінкою що відноситься до даного товару. Усі неоцінені товари будуть мати стартову оцінку рівну одиниці. Якщо товар отримав найвищу оцінку у матриці після того як його купили, то оцінка на нижчу може змінитися лише в тому випадку, якщо покупець цього товару поставить йому негативну оцінку у рейтингу. Якщо це трапиться, оцінка буде змінена на нуль.

Рис. 2.3. – Приклад відслідковування та запису дій користувача.

Матриця, що містить в собі інформацію про незареєстрованих користувачів, заповнюється в загальному. Тобто у ній буде один рядок користувач, та безліч стовпців із продуктами. Далі по аналогії з іншою матрицею за допомогою скриптів буде отримуватися інформація, надаватись оцінки. Різницею у підходах цих матриць буде те, що оцінка буде не замінятись, а додаватись або відійматись. Таким чином можна буде сформувати рейтинг популярності продуктів користувачів.

Третя матриця, буде заповнюватися власноруч. Вона буде містити не усі товари доступні на сайті, а лише ті, які вигідні бізнесу. Співробітники інтернет-магазину зможуть самостійно присвоювати продуктам у матриці оцінки. Використовувати можна буде будь які цілочислові значення.

У результаті, коли всі три матриці будуть містити інформацію і постійно змінюватися, буде проведено їх аналіз та посортовано за рейтингом. Далі у хід піде item-item алгоритм, для того щоб знайти найбільш схожі товари за атрибутами (бренд, категорія, ціна і т. д.) для їх подальшого розміщення у рекомендаційній каруселі. Щоб побудувати найбільш сприятливу для користувачів та власників інтернет-магазину стратегію, товар для розміщення у рекомендаціях буде братися з перших позицій матриць за допомогою Thompson Sampling алгоритму. Під час його використання буде сформований рейтинг успішності кожної матриці, що у подальшому буде відображатися на кількості розміщених товарів з даної матриці у рекомендаційній каруселі.

2.4 Висновки за розділом 2

Основною задачею створення рекомендаційної системи є правильний вибір алгоритмів та створення бізнес-логіки за якою вона буде працювати. Кожна подібна система має вирішувати поставлені перед нею задачі, тому дуже часто недостатньо просто використати уже готовий алгоритм, потрібно його модифікувати під потреби системи.

Правильна архітектура рекомендацій, збір та аналітика даних, на основі яких вона буде працювати це основний критерій її успіху. Тому важливо робити таку систему адаптивною, щоб вона постійно навчалась та підлаштовувалась під потреби користувачів та комерції. Для вирішення цієї задачі найкраще підходить алгоритм семплювання Томпсона, якщо його взяти за основу та оптимізувати відносно потреб системи, даних з якими він буде працювати, то в результаті отримаємо значне підвищення доцільності рекомендацій та успіху продаж.

Система має бути зручною для користувачів, тому доцільно організовувати збір інформації про їх дії таким чином, щоб це не створювало для відвідувачів незручностей. Виходячи з цього, було прийнято рішення використовувати неявний збір інформації, щоб користувачі не відволікалися на відповіді системі, заповненню анкет і т. д.

РОЗДІЛ 3 РОЗРОБКА СИСТЕМИ АВТОМАТИЗАЦІЇ

3.1 Архітектура системи

Провівши аналіз усіх доступних архітектур створення що використовуються під час розробки веб ресурсів, було виділено три основних шаблона:

- Файл-сервер
- Клієнт-сервер
- Архітектура web-додаток

Файл-серверна архітектура використовується у тому випадку, коли усі дані будуть зберігатися на спеціально виділеному для цього мережевому ресурсі, що називається “файловим сервером”. Даний підхід зазначає, що усі функціональні компоненти системи знаходяться на стороні клієнта (комп'ютер користувача), а дані знаходяться на стороні серверу.

Переваги:

- багатокористувацький режим роботи з даними;
- централізоване керування правами доступу до загальних даних;
- доступна вартість розробки;
- швидкий у розробці.

Недоліки:

- послідовний доступ до даних;
- відсутність гарантії цілісності даних;
- продуктивність залежить від продуктивності клієнта, мережі, серверу;

Клієнт-сервісна архітектура [22] являє собою мережеву інфраструктуру, де сервери постачають сервіси (послуги), а клієнтські комп’ютери їх використовують. Класичний вигляд клієнт-серверної архітектури передбачує

наявність у мережі сервера та підключених до нього клієнтів. У таких системах сервер, виконує роль постачальника послуг з використанням бази даних.

Переваги:

- підтримка багатокористувацької роботи;
- цілісність даних;
- механізми керування правами доступу до сервера;
- можливість розподілу функцій між вузлами мережі.

Недоліки:

- поломки серверу впливають на всю систему;
- потрібні високо кваліфіковані співробітники;
- надмірна вартість устаткування.

Архітектура web-додаток використовує сервіси, розміщені у мережі Internet, через спеціальний додаток. Архітектура таких сервісів схожа за концепцією з багатоланкової клієнт-серверною, однак, сервера додатків і баз даних розташовуються в мережі Internet. Можна відокремити наступні недоліки:

- складність реалізації;
- залежність від платформи;
- низька продуктивність;

Виходячи з вище перерахованих принципів роботи даних архітектур та їх недоліків, було прийняте рішення використовувати клієнт-серверну архітектуру системи так як вона забезпечує цілісність даних, підтримує багатокористувацьку роботу та надає можливість чітко розділити клієнтську та серверну частини веб-програми (див. рис. 3.1).

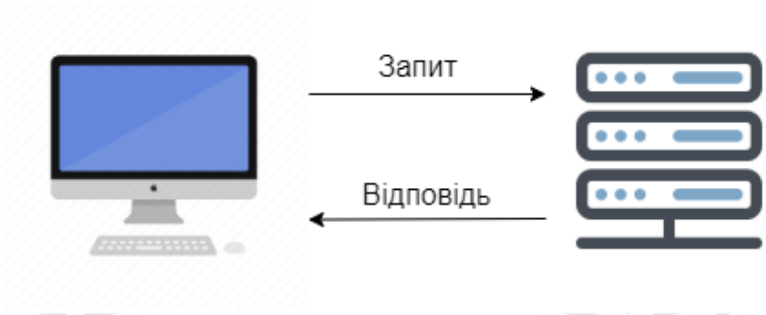


Рис. 3.1. Архітектура клієнт-сервер

Для створення інтернет магазину буде використана платформа Salesforce, яка надає хостинг, сервера та зручну базу даних в якій можна розмістити вибірку продуктів та користувачів. Використана наразі готова база даних продуктів з різними параметрами, на які буде опиратися рекомендаційна система.

Створена система поділена на наступні компоненти:

- Сервери, до яких будуть надсилатися запити, після чого вони будуть повертати потрібну інформацію та дані.
- Клієнтська частина, за допомогою якої буде відбуватися обмін даними з сервером.
- Мережа, з допомогою неї буде підтримуватися зв'язок між усіма компонентами.

Таким чином ми отримали дві логічні частини:

- Frontend – відображення даних для користувача.
- Backend – відповідає за бізнес-логіку програмного продукту.

Такий підхід дозволяє чітко розмежувати зони відповідальності для розробників, та налагодити їх паралельну роботу на даним проектом. Так як усі подібні сайти завжди продовжують розвиватися, нарощувати нові функціональності, лагодити деякі баги, вважаю за потрібне використовувати зручну структуру проекту.

У якості бази даних буде використана база даних надана платформою Salesforce. Вона є найбільш зручною для інтернет комерції, адже вона спроектована та заточена під розробку інтернет магазинів. Усі таблиці даної бази чітко заточені під інтернет магазини, мають власні атрибути, та можливість додавати кастомні при потребі.

Інформація, що буде передана до бази даних буде збиратися на Frontend частині проекту, а надсилатися за допомогою Backend інструментів, що також надає платформа Salesforce. Отримуватися інформація з бази буде за допомогою Backend засобів і далі оброблятися та надсилатися у Frontend частину сайту.

3.2 Опис реалізації серверної частини системи

Для реалізації серверної частини програми був використаний архітектурний паттерн Model-View-Controller (MVC). За допомогою моделей формуються необхідні нам дані, які надходять до контроллерів, які вже в свою чергу виконують ті чи інші команди, повертаючи нам необхідний результат. Структуру даного проекту зображено на рис. 3.2.

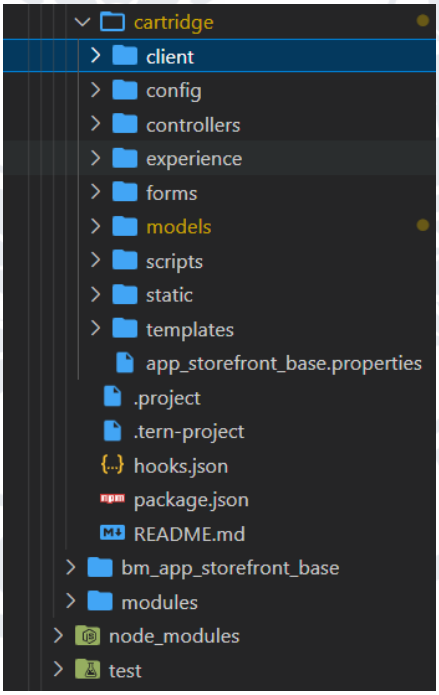
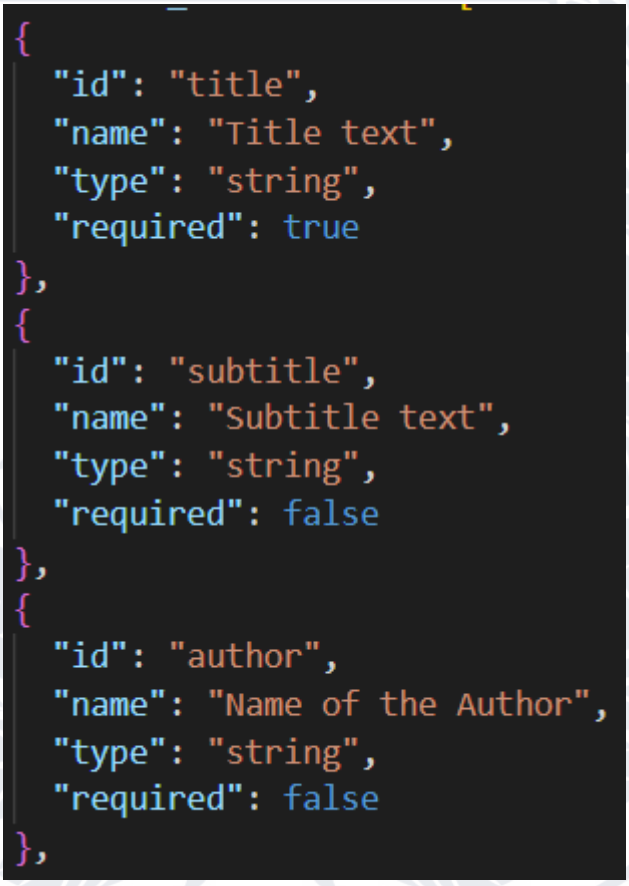


Рис 3.2. Структура проекту

Для написання backend частини проекту використовується мова програмування javascript п’ятої версії, так як він найкоректніше підтримується платформою Salesforce.

Користувач при взаємодії з різними компонентами сайту відправляє за допомогою клієнтських скриптів запити на контроллери. В залежності від типу запиту (get або post) контроллери повертають відповідь у форматі JSON (див. рис. 3.3) або HTML (сформовану нову сторінку сайту). При формуванні нових сторінок контроллери часто збирають нову інформацію, яку динамічно вставляють у HTML розмітку.



```
{
  "id": "title",
  "name": "Title text",
  "type": "string",
  "required": true
},
{
  "id": "subtitle",
  "name": "Subtitle text",
  "type": "string",
  "required": false
},
{
  "id": "author",
  "name": "Name of the Author",
  "type": "string",
  "required": false
},
```


Рис. 3.3. Приклад JSON відповіді контроллеру

Для того щоб уся інформація була структурованою у платформі Salesforce є можливість створювати нові класи та методи.

Було введено наступні класи:

- Products

- Customers
- Promotions (пропозиції)

Кожен клас має власну модель (див. рис 3.4), яка формує екземпляр даного класу, опираючись на всі вхідні дані що зберігаються у базі даних що надає платформа Salesforce.

```
/**
 * Decorate product with product tile information
 * @param {Object} product - Product Model to be decorated
 * @param {dw.catalog.Product} apiProduct - Product information returned by the script API
 * @param {string} productType - Product type information
 *
 * @returns {Object} - Decorated product model
 */
module.exports = function productTile(product, apiProduct, productType) {
    var productHelper = require('*/cartridge/scripts/helpers/productHelpers');
    var productSearchHit = productHelper.getProductSearchHit(apiProduct);
    decorators.base(product, apiProduct, productType);
    decorators.searchPrice(product, productSearchHit, promotionCache.promotions, productHelper.getProductSearchHit);
    decorators.images(product, apiProduct, { types: ['medium'], quantity: 'single' });
    decorators.ratings(product);
    if (productType === 'set') {
        decorators.setProductsCollection(product, apiProduct);
    }

    decorators.searchVariationAttributes(product, productSearchHit);

    return product;
};
```

Рис 3.4. Приклад моделі продукту.

Після отримання екземпляру класу, що потребує система у даний момент, він відправляється у контроллер (див. рис. 3.5) у потрібному форматі для подальшого його використання.

```
module.exports.render = function (context, modelIn) {
    var model = modelIn || new HashMap();
    var content = context.content;

    model = carouselBuilder.init(model, context);
    model.textHeadline = content.textHeadline ? content.textHeadline : null;
    model.displayRatings = context.content.displayRatings;
    model.swatches = true;

    model.regions = PageRenderHelper.getRegionModelRegistry(context.component);

    var recommender = content.recommender;
    model.limit = parseInt(content.count, 10) || 1;

    if (recommender) {
        model.recommender = recommender.value;
    } else {
        throw new Error(Resource.msg('pd.no.prods.error', 'error', null));
    }

    model.productLoadUrl = URLUtils.abs('recomendationCarousel-Load');

    model.id = 'carousel-' + PageRenderHelper.safeCSSClass(context.component.getID());
    return new Template('experience/components/recomendation/recomendationCarousel').render(model);
};
```

Рис. 3.5. Приклад контролерра що відповідає за рендеринг рекомендаційної каруселі.

Також для уникнення повторів коду, було створено middleware (див. рис. 3.6), що викликаються перед початком виконання контроллерів або після їх виконання. Таким чином, їх було зручно додавати у всіх потрібних для цього місцях

```
function validateLoggedInAjax(req, res, next) {
    if (!req.currentCustomer.profile) {
        if (req.querystring.args) {
            req.session.privacyCache.set('args', req.querystring.args);
        }

        var target = req.querystring.rurl || 1;

        res.setStatusCode(500);
        res.setViewData({
            loggedin: false,
            redirectUrl: URLUtils.url('Login-Show', 'rurl', target).toString()
        });
    } else {
        res.setViewData({
            loggedin: true
        });
    }
    next();
}
```

Рис. 3.6. Приклад middleware що перевіряє чи користувач залогінився.

Для збору інформації у потрібному форматі та її валідації використовуються форми у XML форматі (див. рис. 3.7). Для валідації текстових полів було використано регулярні вираження, щоб уникнути можливості маніпуляції хибною інформацією зі сторони користувачів.

```xml
<?xml version="1.0"?>
<form xmlns="http://www.demandware.com/xml/form/2008-04-19" secure="false">
    <field
        formid="newpassword"
        label="label.input.newpassword.profile"
        mandatory="true"
        min-length="8"
        max-length="255"
        range-error="error.message.8_255characters"
        value-error="ValueErrorText"
        type="string"/>

    <field
        formid="newpasswordconfirm"
        label="label.input.newpasswordconfirm.profile"
        mandatory="true"
        min-length="8"
        max-length="255"
        range-error="error.message.8_255characters"
        value-error="ValueErrorText"
        type="string"/>
</form>
```

Рис. 3.7. Приклад форми для валідації пароля користувача.

3.3 Опис реалізації клієнтської частини системи

Клієнтська частина інтернет магазину реалізована переважно за допомогою наступних компонентів:

- Jquery - JavaScript Framework.
- Javascript – мова веб-програмування.
- ISML - розширення мови на основі тегів, що відповідає стандартам SGML. Включає в себе методи, надані платформою salesforce. Їх можна використовувати разом із звичним нам HTML.
- CSS – (Cascading Style Sheets) Це мова для опису представлення веб-сторінок, надає інструменти для задання кольорів, стилів, розміщення на сторінці і т. д. різних HTML елементів сторінки.

В залежності від сторінки, на якій знаходиться користувач, відпрацьовують ті чи інші елементи розмітки. Дані, для подальшого їх відображення на сторінках інтернет-магазину отримуються із контроллерів, далі вони динамічно додаються до HTML розмітки за допомогою методів та інструментів, що надає нам ISML(див. рис. 3.8).

```
<div class="row product-grid">
    <isloop items="${slotcontent.content}" var="product" begin="0" end="${end}">
        <isobject object="${product}" view="recommendation">
            <div class="col-6 col-sm-4">
                <isinclude url="${URLUtils.url('Tile-Show', 'pid', product.ID, 'swatches', true, 'ratings', true, 'showQuickView', false)}" />
            </div>
        </isobject>
    </isloop>
</div>
```

Рис. 3.8. Приклад побудови рекомендаційної каруселі із використанням мови ISML

На кожній такій сторінці підключені CSS стилі та JS клієнтські скрипти. За допомогою стилів ми задаємо візуальну складову елементів (див. рис. 3.9), а за допомогою скриптів організовуємо обробку тих чи інших подій, які створюють користувачі, а також надсилання результати цих дій до серверної частини проекту.

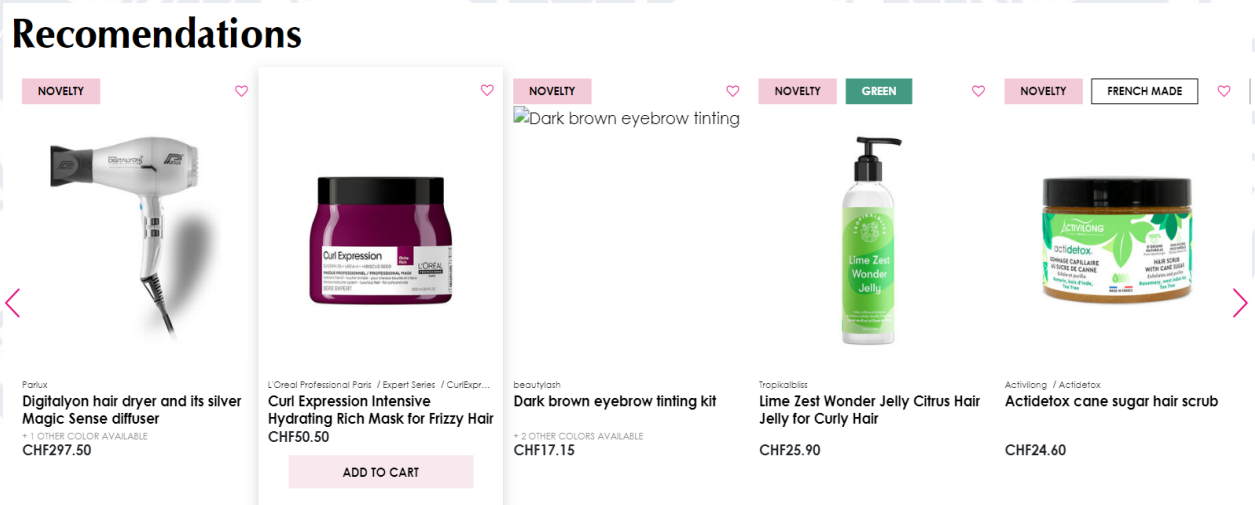


Рис. 3.9. Приклад вигляду рекомендаційної каруселі

Усі запити до серверної частини сайту виконані за допомогою ajax запитів. Типи запитів поділяються на post та get.

Одним з головних завдань клієнтської частини системи у даній роботі є збір інформації про поведінку користувачів. Кожен перехід користувача на нову сторінку, відкриття конкретних продуктів та додавання їх у корзину, купівля продукту – усе це збирається та записується за допомогою клієнтських скриптів. Приклад такого скрипта продемонстровано на рис. 3.10.

```
function initAddItemToCart() {
    if (analyticsData.isSFRA) {
        $('body').on('product:afterAddToCart', function () {
            $.ajax({
                url: EmarsysUrls.emarsysAddToCartAjax
            }).done(function (data) {
                if (data) {
                    window.ScarabQueue.push(['cart', data.cartObj]);
                    window.ScarabQueue.push(['go']);
                }
            });
        });
    }
}
```

Рис. 3.10. – скрипт що відслідковує подію додавання продукту до корзини

Клієнтські скрипти написані за допомогою javascript та jquery. Перед використанням jquery, його потрібно підключити на даній сторінці, де він буде використаний за допомогою тегу HTML тегу script (див. рис. 3.11).

```
<head>
<script type="text/javascript" src="jquery.js"></script>
</head>
```

Рис. 3.11. Приклад підключення jquery до сторінки.

Скрипти які збирають інформацію підключені на кожній сторінці інтернет-магазину для повного відслідковування та аналізу поведінки користувачів.

3.4 Опис реалізації бази даних

Базу даних надає платформа Salesforce, вона дуже зручна у використанні, та її можна заповнювати як власноруч, створюючи та конфігуруючи нові продукти, так і завантажувати в неї dataset з потрібною інформацією у правильному форматі (XML).

Для того щоб поділити масиви даних по категоріям, платформа дає змогу використовувати класи, на які будуть ділитися об’єкти що в ній зберігаються. Таким чином, для даної роботи було прийняте рішення використовувати наступні класи:

- Products
- Customers
- Promotions (пропозиції)

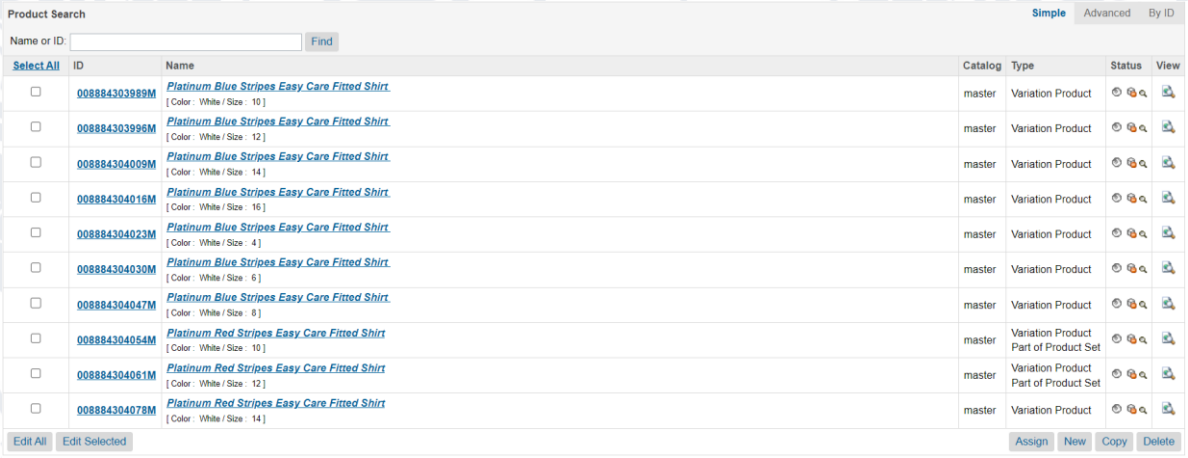


Рис. 3.12. Приклад зберігання екземплярів класу продукт

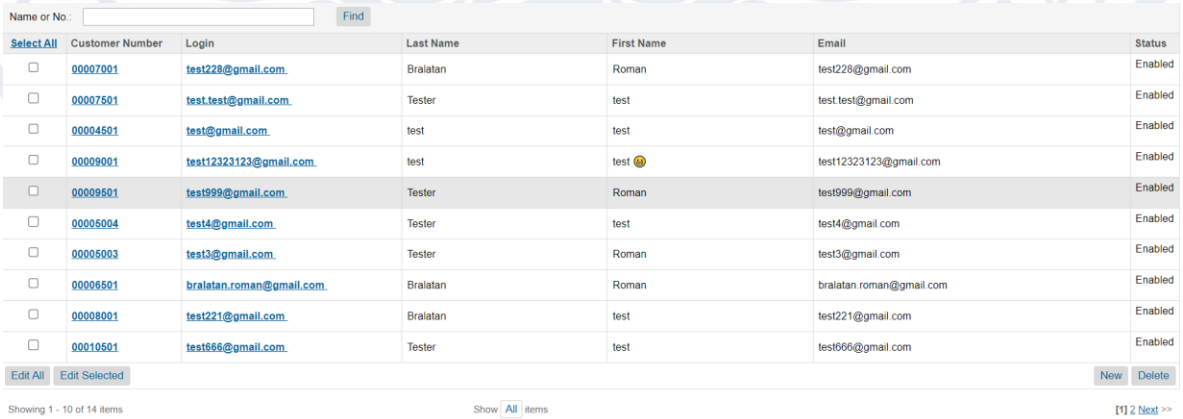


Рис. 3.13 Приклад зберігання екземплярів класу користувачів

Платформа надає клас Transaction із своїми вбудованими методами, що дозволяють вільно записувати інформацію у базу даних. Для його використання, спочатку потрібно підключити даний клас до проекту (рис. 3.14).

```
var Transaction = require('dw/system/Transaction');
```

Рис. 3.14. Підключення системного класу Transaction

Далі потрібно сформувати потрібну інформацію у правильному форматі, та використовуючи метод wrap() класу Transaction вказати у ньому шлях до атрибуту та його id, після чого присвоїти йому потрібне нам значення (див. рис. 3.15).

```
Transaction.wrap(function () {
    profile.setFirstName(formInfo.firstName);
    profile.setLastName(formInfo.lastName);
    profile.setEmail(formInfo.email);
    profile.setPhoneHome(formInfo.phone);
});
```

Рис. 3.15. Приклад запису даних

Для створених власноруч атрибутів таблиць, до шляху перед id атрибуту додається слово custom, щоб система змогла його знайти та коректно записати у нього дані (див. рис. ….).

```
customer.profile.custom.uploadedDocument
```

Рис. 3.16. Приклад розташування створених власноруч атрибутів

Для отримання інформації із бази даних потрібно використовувати ті класи, екземпляри яких ми хочемо отримати. У нашому випадку необхідно використовувати наступні:

- CustomerMgr;

- ProductMgr;
- PromotionMgr.

Кожен із перерахованих класів надає методи, за допомогою яких можна отримати як конкретний екземпляр класу по його атрибуту, так і усі існуючі екземпляри.

3.5 Інструкція розробнику

Проект потрібно розгортати використовуючи середовище програмування Visual Studio Code. Відкриваємо проект у середовищі, після чого потрібно відкрити термінал. У терміналі потрібно вказати шлях до директорії з проектом, після чого ввести команду встановлення node modules (див. рис.3.16).

Рис. 3.16. Команда для встановлення node modules

Після встановлення усіх потрібних компонентів, потрібно також встановити потрібні для нас додатки у середовищі. Для цього нам потрібно перейти до Extensions секції у Visual Studio Code, та у пошукову стрічку ввести “Prophet” (див. рис. 3.17). Prophet Debugger використовується для того щоб заливати код до платформи Salesforce для подальшого його використання, а також для того щоб можна було відлагоджувати код.

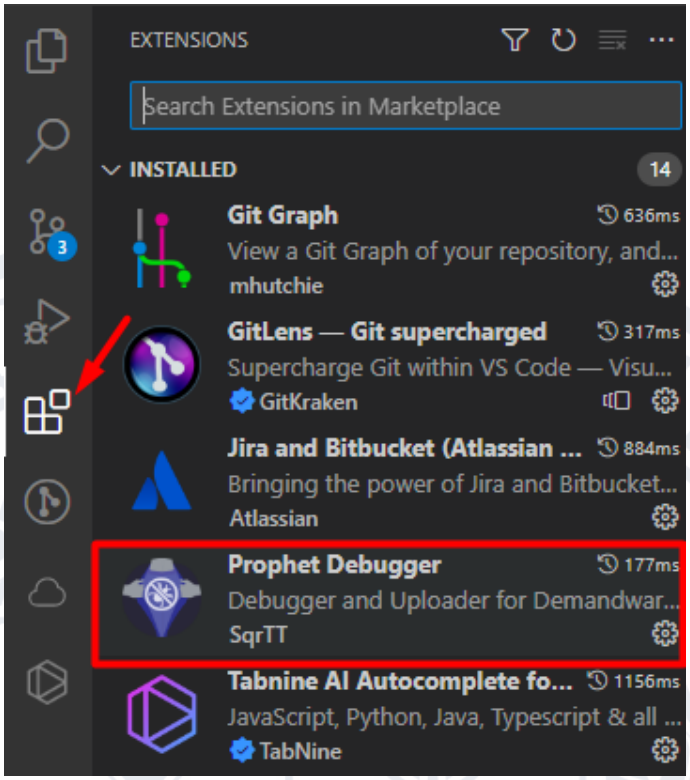


Рис. 3.17. Приклад встановлення компонентів

Наступним етапом налаштування проекту будуть наступні файли, розташовані у проекті:

- dw.json;
- launch.json;

Перед тим як перейти до їх налаштування, потрібно отримати аккаунт Salesforce із логіном та паролем. Далі потрібно створити sandbox на якому буде розгорнуто ваш сайт.

Файл dw.json містить інформацію про розробника та sandbox, він потрібен для того, щоб середовище чітко розуміло куди саме відправляти сформований у проекті код. У файлі потрібно вказати наступну інформацію у json форматі (див. рис. 3.18):

- hostname – URL розташування sandbox, наданого Salesforce;
- username – логін користувача;
- password – пароль користувача;

- code-version – назва код версії яку ви використовуєте та на яку буде відправлений ваш код;
- cartridge – назви папок проекту, код з яких ви хочете надіслати на sandbox;

```
{
    "hostname": "bjbc-011.sandbox.us01.dx.commercecloud.salesforce.com",
    "username": "r.bralatan@test.com",
    "password": "**********",
    "cartridge": [],
    "code-version": "roma"
}
```

Рис. 3.18. Приклад dw.json файлу

Наступний файл для налаштування launch.json. Цей файл використовується Prophet Debugger компонентом для відлагоджування коду. Його слід розмістити у кореневій директорії. Контент що має містити даний файл зображено на рисунку 3.19.

```
{
    // Use IntelliSense to learn about possible attributes.
    // Hover to view descriptions of existing attributes.
    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "type": "prophet",
            "request": "launch",
            "name": "Attach to Sandbox"
        }
    ]
}
```

Рис. 3.19. Приклад launch.json файлу

Останнім етапом налаштування проекту є компіляція клієнтських скриптів. Усі команди компіляції розташовані у package.json файлі (див. рис. 3.20), що також знаходиться у кореневій директорії проекту.

```
"compile:js": "sgmf-scripts --compile js",
"compile:js:bm": "sgmf-scripts --compile js --cartridgeName",
"compile:scss": "sgmf-scripts --compile css",
"compile:scss:bm": "sgmf-scripts --compile css --cartridgeName",
```

Рис. 3.20. Приклад команд для компіляції клієнтських скриптів.

Дані команди потрібно скопіювати та ввести у терміналі. Після завершення компіляції, можна відправляти код до sandbox, та працювати із сайтом. Для відправки коду потрібно натиснути F1 та обрати наступну команду “Prophet: Enable upload” (див. рис. 3.21).

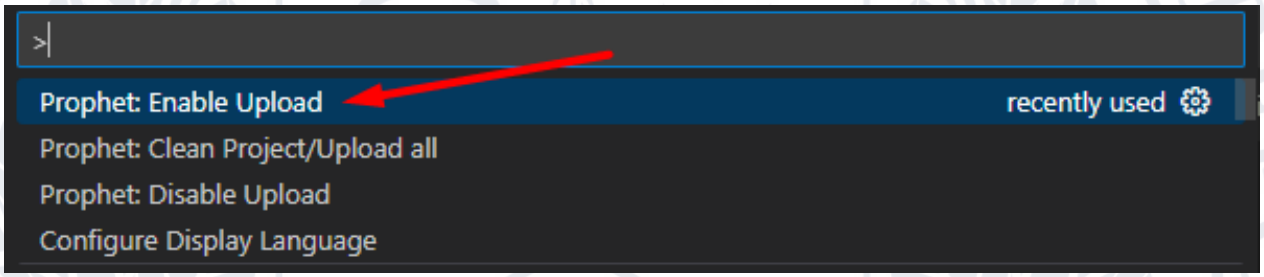


Рис. 3.21. Приклад відправки коду до sandbox.

3.6 Висновки за розділом 3

Інтернет-магазин це комерційний проект, який має приносити прибуток та відповідати усім стандартам. Важливою складовою його написання є проектування архітектури проекту, адже правильно підібрана архітектура забезпечує швидкодію системи, цілісність та захищеність даних, багатокористувацький режим роботи, зручність розробки і т. д., тому перш а все потрібно правильно сформувати архітектуру.

Існуючі архітектури, що використовуються для веб розробки, створені відповідно до задач, які вони мають виконувати. Найкращим варіантом для електронної комерції являється архітектура клієнт-сервісу, через його зручність для розробників, та ряду інших переваг.

Наступним етапом розробки будь якого програмного забезпечення є підбір інструментів для реалізації, виходячи із поставлених задач. Є безліч варіантів мов програмування та сервісів, які можна використовувати, тому слід ретельно аналізувати усі існуючі аналоги, обирати той, що є найдоречнішим.

РОЗДІЛ 4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

4.1 Методика експерименту та його план

Експерименти це найкращий спосіб перевірити на практиці роботу тих чи інших компонентів та алгоритмів системи. Було прийняте рішення, для підтвердження правильної та коректної роботи усіх створених у ході даної роботи алгоритмів та моделей, провести ряд дослідів. Потрібно впевнитися в коректній роботі рекомендаційної системи на старті її роботи із не заповненими матрицями що містять інформацію про дії користувачів, прослідкувати її реакцію на заповнення цих даних, та впевнитися в її адаптивності.

У ході експериментів очікується підтвердження адаптивності рекомендаційної системи відносно до уподобань користувача, а також відносно успішності запропонованих нею товарів. Очікується коректний збір та аналіз інформації про дії користувачів, та коректна обробка отриманих даних для подальшого використання з метою рекомендацій тих чи інших продуктів. Рекомендаційна карусель має пропонувати нові продукти кожного разу, після перезавантаження сторінки.

Проводитися усі експерименти будуть на створеному тестовому інтернет-магазині, розробленому у ході даної роботи разом із рекомендаційною системою. Інтернет-магазин наповнено різними категоріями із продуктами, що мають різні атрибути та бренди, з якими будуть взаємодіяти користувачі.

4.2 Відображення рекомендаційної каруселі для нового користувача

Перший дослід проведено у щойно створеній рекомендаційній системі, коли ще не зібрано жодних аналітичних даних з діями користувачів, їх уподобаннями та запитами. Реєструємо нового користувача, та заходимо у систему використовуючи його аккаунт (див. рис. 4.1).

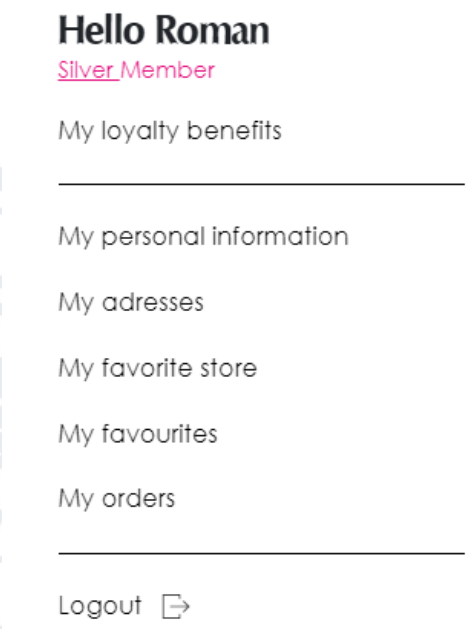


Рис. 4.1. Персональний кабінет користувача

Умовно присвоюємо кожній матриці з даними колір, для того щоб у подальшому помічати кожен продукт відповідним кольором матриці, з якої його було взято та поміщено до рекомендаційної каруселі. Матриця що містить інформацію про дії усіх користувачів інтернет-магазину буде мати червоний колір, матриця що містить дані про дії конкретного користувача, який увійшов у систему буде зображена зеленим кольором, а матриця, яка заповнюється вручну власником або маркетинговою командою буде помічена синім кольором.

У тільки створеній системі, початково усім матрицям надано однакового коефіцієнту успішності, та продукти що містяться у матрицях із зібраною інформацією про дії користувачів мають однаковий рейтинг продуктів. Таким чином на старті рекомендаційна система в більшій мірі опирається на матрицю що заповнює власник або маркетингова команда, так як у ній уже власноруч надані рейтинги продуктів, які в першу чергу мають бути відображені на рекомендаційній каруселі. Сторінку із рекомендаційною системою зображено на рис. 4.2, де відмічено потрібним кольором кожен продукт відповідно до того з якої матриці його було взято.

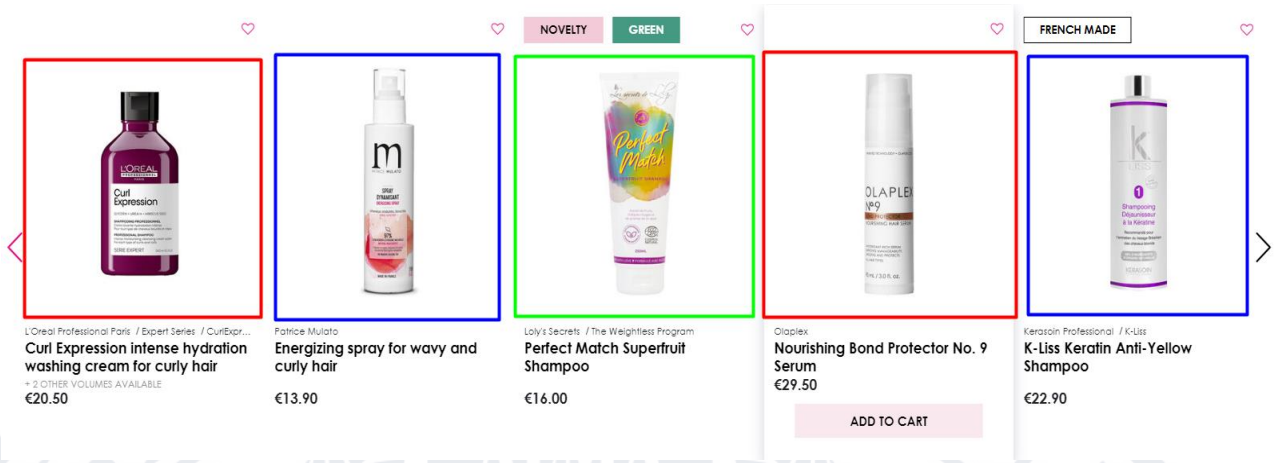


Рис. 4.2. Рекомендаційна карусель

Рекомендаційна карусель містить у собі п’ятнадцять продуктів, на сторінці відображено лише п’ять, для того щоб побачити інші потрібно натиснути на кнопку збоку каруселі, після чого відобразяться наступні продукти. На рис. 4.2 видно, що продукти взяті у наступному порядку: спочатку червоний, потім синій, потім зелений. Карусель закінчується на синьому продукті, тому якщо ми перейдемо на її наступну сторінку, вона розпочнеться із зеленого продукту (див. рис. 4.3).

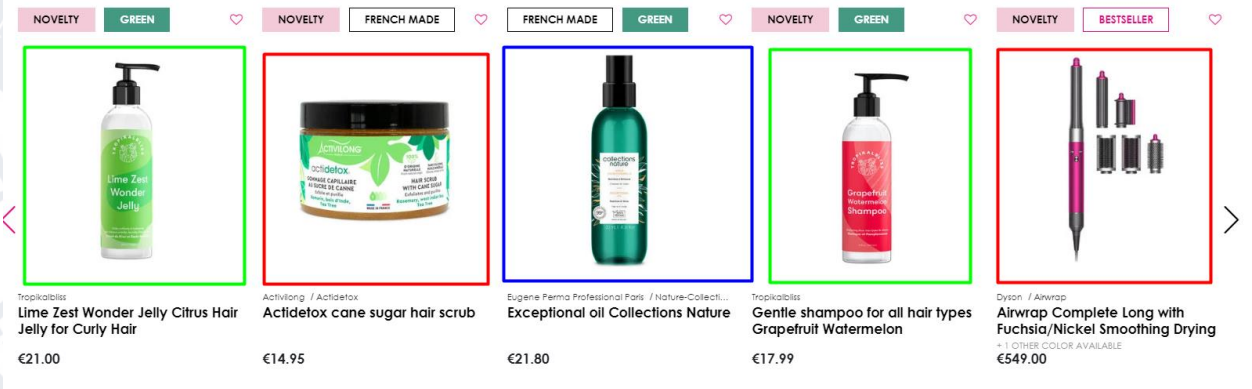


Рис. 4.3. Наступна сторінка рекомендаційної каруселі.

Виходячи з даного експерименту, можна зробити висновки, що без зібраної та проаналізованої інформації щодо успішності роботи кожної матриці, ми маємо рівномірно розподілену рекомендаційну систему, що пропонує користувачеві продукти із усіх трьох матриць в однаковій кількості.

4.3 Збір інформації та її використання

Візьмемо нового користувача, та спробуємо додати певні продукти до кошику. Перейдемо на сторінку електронних приладів, та додамо до кошику два продукти з цієї категорії (див. рис. 4.4).

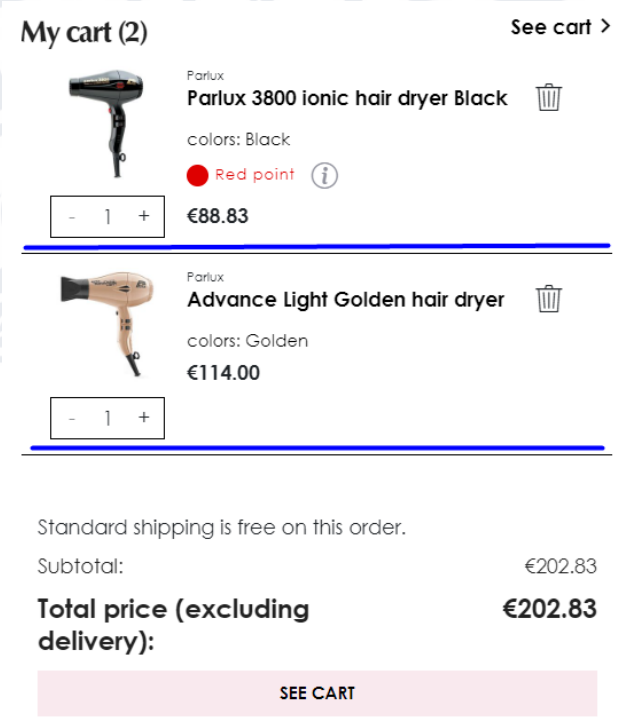


Рис. 4.4. Продукти у кошику користувача

Додавши два продукти, було створено події з певною оцінкою, які записали до матриці із інформацією про уподобання конкретного користувача, що конкретні продукти цікаві відвідувачеві та повисили їх рейтинг. Таким чином, матриця за рахунок отриманої інформації та item-item алгоритму надала більшого пріоритету усім продуктам, що лежать у цій категорії, або відносяться до того ж бренду, або містять схожі атрибути. При наступному відвідуванні рекомендаційної каруселі, видно, що товар того ж бренду та категорії (але не той що було додано до кошику) було розміщено у рекомендаційній каруселі (див. рис. 4.5).

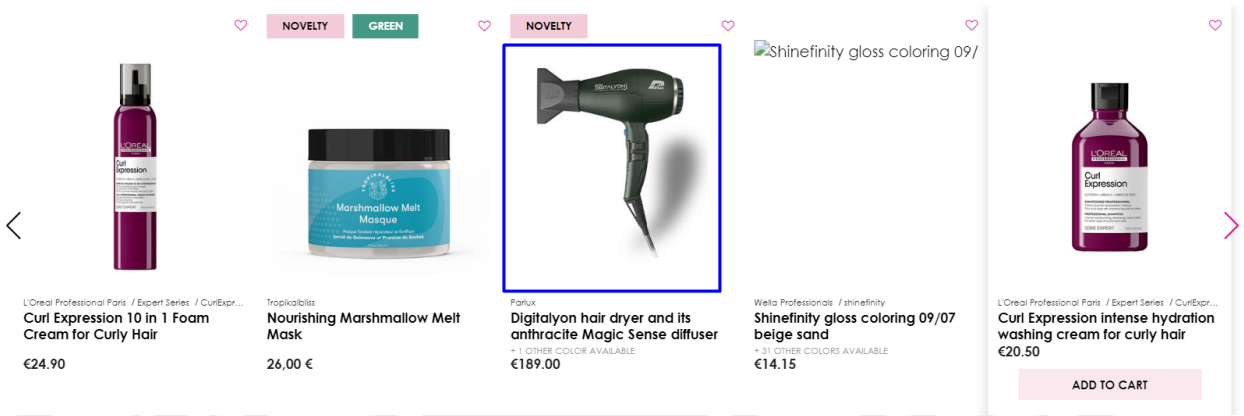


Рис 4.5. Продукт, запропонований на основі проаналізованої інформації

Перейдемо до наступної категорії, де знаходяться маски для волосся. Обираю одну з них та додаю до свого кошику (див. рис. 4.6).

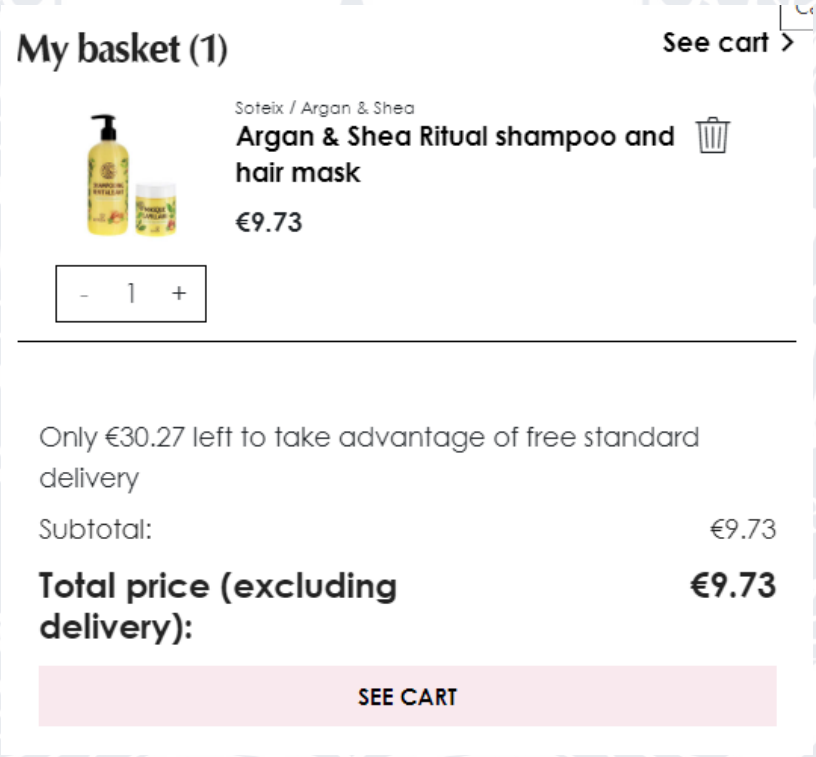


Рис. 4.6. Додавання нового експериментального продукту до карти

Після додавання маски для волосся до карти, я вирішую її придбати, тому переходжу на сторінку розрахунку, та купую товар (див. рис. 4.7).

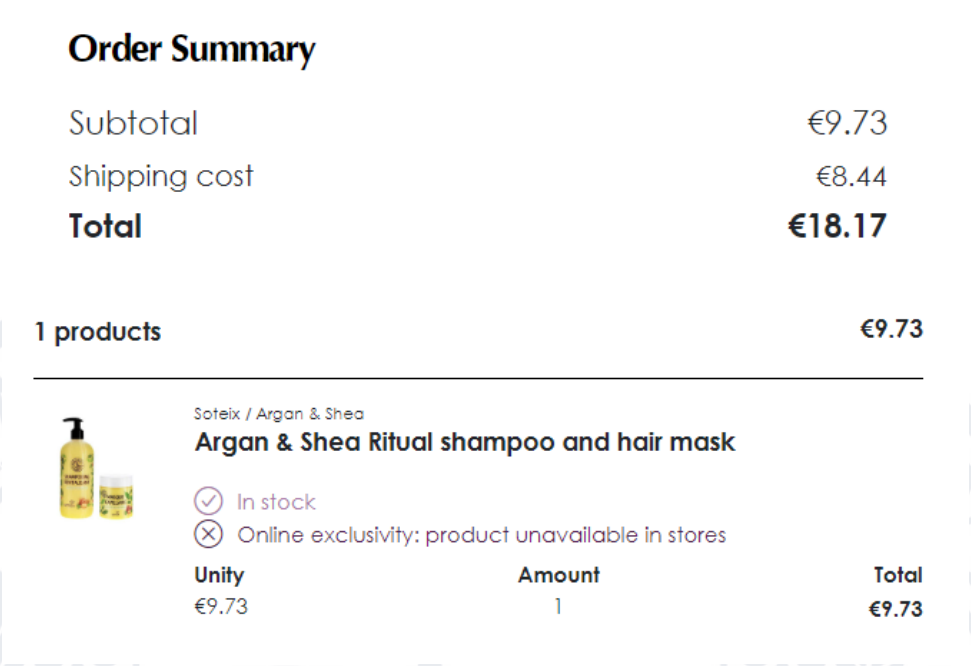


Рис. 4.7. Сторінка підтвердження купівлі товару

Після купівлі даного товару, переходжу до рекомендаційної каруселі. Подія купівлі товару має вищий пріоритет та оцінку, ніж додавання продукту до кошику, тому на рекомендаційній каруселі на початку розташований товар з категорії масок для волосся, далі ідуть товари з інших матриць, після яких на другій позиції розташовано продукт з категорії електроніки, які ми додавали до кошику, але не купували (див. рис. 4.8).

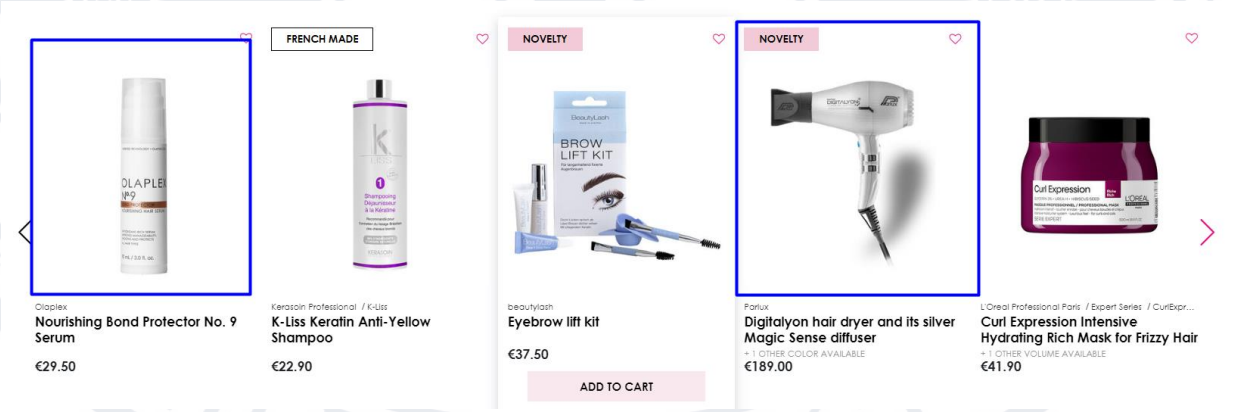


Рис. 4.8. Рекомендації після певних дій користувача

Таким чином, можна зробити висновок, що система у реальному часі відслідковує та аналізує дії користувача, надає їм оцінку, та на основі

отриманих даних додає до рекомендаційної каруселі продукти відповідно до їх рейтингу.

4.4 Адаптація системи на основі успішності матриць

Візьмемо стартову ситуацію, коли коефіцієнт успішності кожної з трьох матриць однаковий, і вони розміщується на рекомендаційній каруселі рівномірно та послідовно (див. рис. 4.9). На коефіцієнт успішності впливає кількість разів, коли користувач купив товари скориставшись рекомендаційною каруселлю та яка саме кількість успішних купівель була завдяки продуктам з конкретної матриці. Тобто коефіцієнт успіху конкретної матриці – це кількість її успішних продаж поділена на суму успіхів усіх матриць.

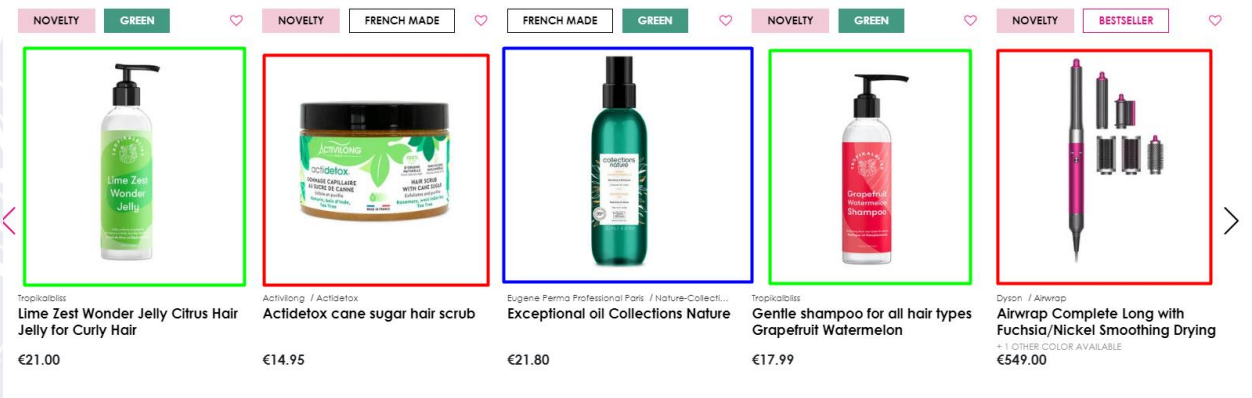


Рис. 4.9. Вигляд каруселі при однакових коефіцієнтах успішності

Для того щоб навчати систему, потрібно купувати товари із рекомендаційної каруселі. Придбаємо декілька товарів із синьої матриці та подивимось на результати на рис. 4.10.

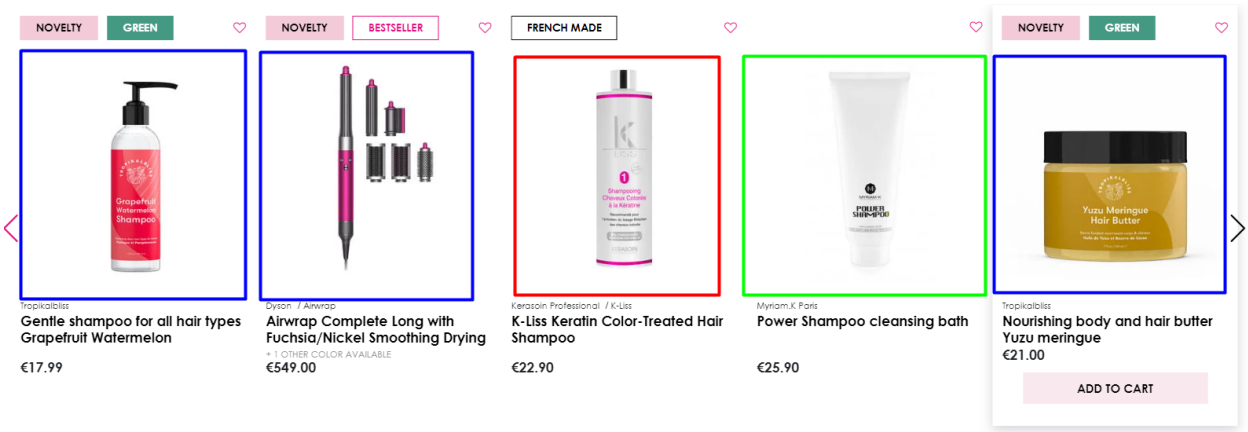


Рис. 4.10. Вигляд каруселі після збільшення успішності синьої матриці

Ми бачимо, що у результаті збільшення успішності синьої матриці, карусель взяла дві позиції з неї, та помістила на початок рекомендацій. Продовжимо купувати товари із синьої матриці та перевіримо результати (див. рис. 4.11).

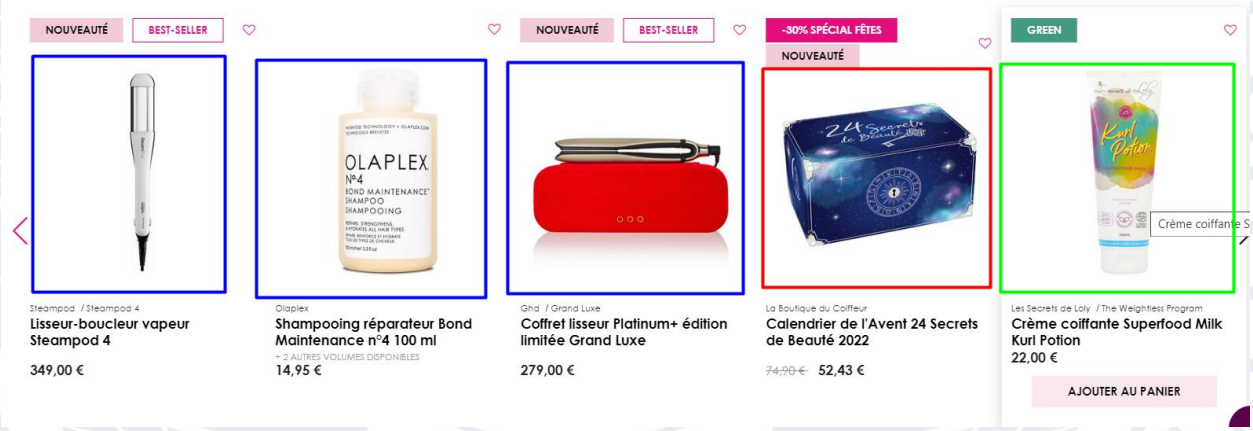


Рис. 4.11. Карусель після наступного збільшення успішності синьої матриці

Тепер на початку каруселі міститься три продукту із успішної матриці, інші матриці мають коефіцієнт 0,3, так як ми зафіксували його нижню межу, щоб не втрачати продуктів із інших матриць, в надії що товари з них також можуть принести продажі. Тому з неуспішних матриць карусель взяла лише по одному продукту. Спробуємо купити декілька продуктів із червоної матриці та перейдемо до рекомендаційної каруселі (див. рис. 4.12).

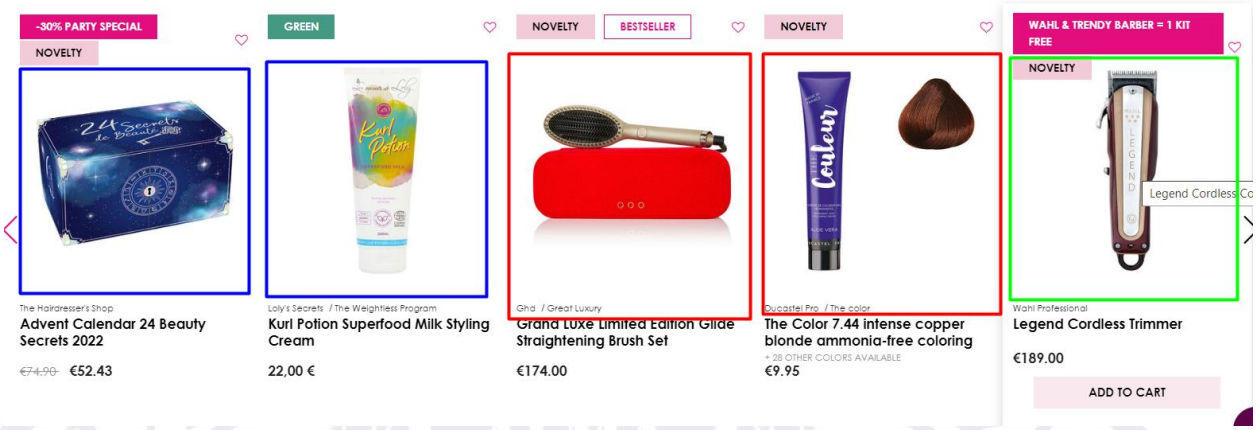


Рис. 4.12. Карусель після збільшення успішності червоної матриці

У результаті ми бачимо, що тепер успішність червоної матриці піднялась і рекомендаційна система запропонувала 2 продукти із синьої та 2 продукти з червоної матриці. Таким чином у результаті успішних рекомендацій, карусель рекомендацій адаптується відповідно до уподобань користувача та обирає найпродуктивнішу стратегію, що є вигідним як для відвідувача, так і для комерції.

4.5 Оновлення продуктів у рекомендаційній каруселі

Проведемо дослід на уже сформованій рекомендаційній каруселі, відповідно до усіх отриманих та проаналізованих даних із метою переконатися, що при перезавантаженні сторінки вона не буде рекомендувати однакові продукти кожного разу, поки користувач не виконає якоїсь дії на сайті. Для цього переходимо на сторінку із рекомендаціями та фіксуємо її вигляд на першій сторінці (див. рис. 4.13).

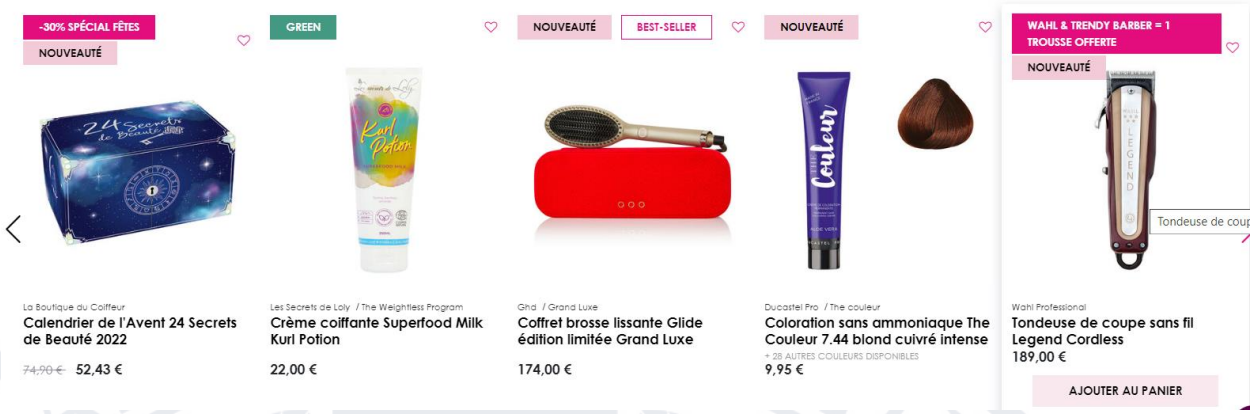


Рис. 4.13. Рекомендаційна карусель при першому завантаженні

Після того, як ми отримали перший результат, перезавантажуємо сторінку та фіксуємо наступний результат (див. рис. 4.14).

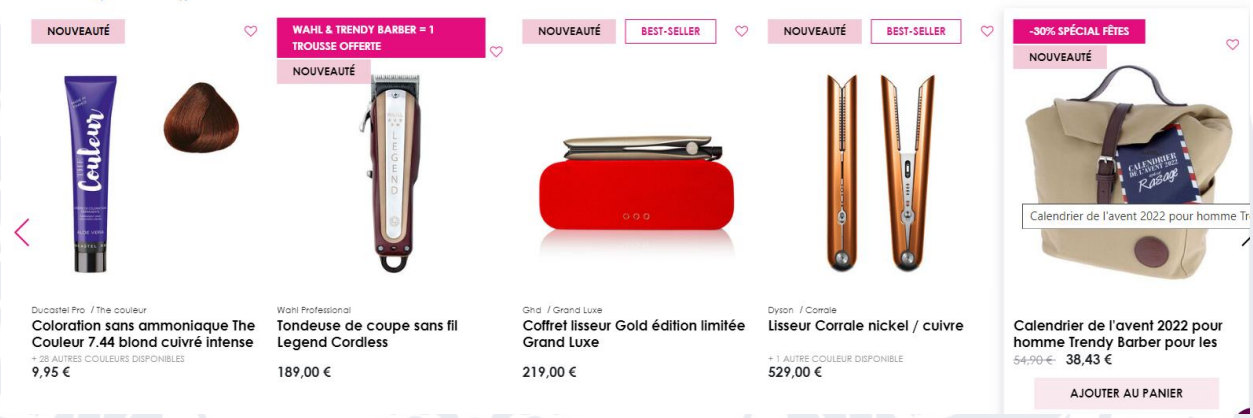


Рис. 4.14. Рекомендаційна карусель після перезавантаження сторінки

У результаті ми отримали нові продукти, що були поміщені до рекомендаційної каруселі, за рахунок рандомізації , що була додана до item-item алгоритму, що робить рекомендаційну систему більш різноманітно наповненою та динамічною.

4.6 Висновки за розділом 4

Експериментами було доведено, що створена рекомендаційна система функціонує, та відповідає усім вимогам. За рахунок модернізації item-item алгоритму ми отримали більшу варіативність рекомендацій, що є важливим елементом будь-якої подібної системи. За рахунок модернізації алгоритму Sampling Tompson ми здобули адаптивність, яка базується на успішності продажу товарів, що збільшує кількість доцільних рекомендацій

користувачеві. Це робить користування даною рекомендаційною системою зручним та вигідним як для відвідувача так і для бізнесу.

Розбиття user-item матриці на три матриці відповідно до задач які вони виконують, робить її більш незалежною від дій та уподобань інших користувачів, даючи можливість відвідувачу побачити ті товари, які цікаві саме йому. Також це вирішує проблему холодного старту, що також є важливим для рекомендаційної системи. Збір та аналіз інформації про дії користувача відбувається непомітно для нього, щоб не турбувати його зайвий раз, а аналіз зібраної інформації відбуваються у реальному часі, так що після певних дій та перезавантаження сторінки ми одразу отримаємо нові результати на рекомендаційній каруселі, відповідно до отриманих даних.

ВИСНОВКИ

Рекомендаційні системи є невід'ємною частиною будь-якого успішного електронного комерційного проекту. Успіх таких проектів напряму залежить від обраної стратегії за якою будуть працювати рекомендації, тому слід ретельно підходити до проектування таких систем, вони мають бути сформовані відповідно до бізнес задач, які ставить перед собою комерційний проект.

Процес дослідження даного питання показав, що усі архітектури побудови рекомендаційних моделей мають свої переваги і недоліки, тому розробникам завжди є простір для розвитку та удосконалення. Користувач оцінює усі аспекти сервісу яким він користується, тому в першу ергу важливо розробити саме таку систему, яка буде максимально комфортна, доцільна та зручна для відвідувача. Кожен сервіс, інтернет-магазин і т. д. конкурують за свою клієнтську базу, постійно працюючи та удосконалюючи свої алгоритми, інтерфейси й інші деталі, на які звертають увагу користувачі.

Розроблена рекомендаційна система у даній роботі була побудована із урахуванням недоліків та переваг уже існуючих, та організована таким чином, щоб задовольняти усі потреби користувачів та бізнесу. Найпоширенішими та найвагомішими проблемами є проблема "Холодного старту" та доцільність рекомендацій. Запропонована та розроблена у ході дослідження архітектура повністю вирішує ці проблеми, та додатково надає можливості бізнесу також долучатися до роботи рекомендацій, для просування потрібних йому продуктів. Система адаптивна, тому вона буде підлаштовуватися індивідуально під кожного користувача, аналізувати його уподобання у реальному часі і тому доцільність таких рекомендацій є дуже високою, що у свою чергу підвищує її продуктивність.

Результати даних досліджень мають зацікавити в першу чергу людей, що працюють у сфері інтернет торгівлі, сервісів які надають ті чи інші послуги. У подальшому дану рекомендаційну систему можна використовувати як

основу, налаштовуючи її в залежності від потреб та задач поставлених перед нею, та додаючи новий функціонал.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Recommendations systems: [Електронний ресурс] – режим доступу до ресурсу: http://infolab.stanford.edu/~ullman/mmds/ch9.pdf
2. Authors: Alex Smola and S.V.N. Vishwanathan – Machine learning - [Електронний ресурс] – режим доступу до ресурсу: https://alex.smola.org/drafts/thebook.pdf
3. Authors: N. Chandra Sekhar Reddy, K. Sai Prasad and A. Mounika - classification algorithm - [Електронний ресурс] – режим доступу до ресурсу: https://www.ripublication.com/ijcir17/ijcirv13n8_23.pdf
4. Алгоритм регресії - [Електронний ресурс] – режим доступу до ресурсу: https://cs.stanford.edu/~ermon/cs325/slides/ml_linear_reg.pdf
5. Algorithms Many-Armed Bandits - [Електронний ресурс] – режим доступу до ресурсу: https://proceedings.neurips.cc/paper/2008/file/49ae49a23f67c759bf4fc791ba842aa2-Paper.pdf
6. The cold start problem in recommender systems - [Електронний ресурс] – режим доступу до ресурсу: http://individual.utoronto.ca/_zihayatm/Papers/IST_2.pdf
7. Salesforse Basics - [Електронний ресурс] – режим доступу до ресурсу: https://resources.docs.salesforce.com/latest/latest/en-us/sfdc/pdf/basics.pdf
8. Authors: Peter Brusilovsky – Collaborative Filtering [Електронний ресурс] – режим доступу до ресурсу: https://sites.pitt.edu/~peterb/2480-152/CollaborativeFiltering.pdf
9. Authors: Dietmar Jannach, Markus Zanker – user-based algorithm [Електронний ресурс] – режим доступу до ресурсу: https://web-ainf.aau.at/pub/jannach/files/BookChapter_CollaborativeFiltering2019.pdf
10. Authors: Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl - Item-Based Collaborative Filtering Recommendation Algorithms -

[Електронний ресурс] – режим доступу до ресурсу: https://www.ra.ethz.ch/cdstore/www10/papers/pdf/p519.pdf

11. Authors: Zakris Strömqvist – User-Item matrix - [Електронний ресурс] – режим доступу до ресурсу: https://uu.diva-portal.org/smash/get/diva2:1214390/FULLTEXT01.pdf
12. Authors: Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl - Explaining Collaborative Filtering Recommendations - [Електронний ресурс] – режим доступу до ресурсу: https://grouplens.org/site-content/uploads/explain-CSCW-20001.pdf
13. Authors: Francesco Ricci - Knowledge-Based Recommender Systems - [Електронний ресурс] – режим доступу до ресурсу: http://eia.udg.es/arl/Agentsoftware/4-KnowledgeBased.pdf
14. Knowledge Based Recommender Systems - [Електронний ресурс] – режим доступу до ресурсу: https://www.aaai.org/Papers/Workshops/2000/WS-00-04/WS00-04-011.pdf Brendon Towle & Clark Quinn
15. Collection of information - [Електронний ресурс] – режим доступу до ресурсу: https://dspace.nlu.edu.ua/bitstream/123456789/6421/1/Konspekt_leksiy_95.pdf
16. Автори: І.В. Захарова та Л.Я. Філіпова – Збір та аналіз інформації - [Електронний ресурс] – режим доступу до ресурсу: https://kpdi.edu.ua/biblioteka/%D0%9E/%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D0%B8%20%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%BE-%D0%B0%D0%BD%D0%B0%D0%BB%D1%96%D1%82%D0%B8%D1%87%D0%BD%D0%BE%D1%97%20%D0%B4%D1%96%D1%8F%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%96%20%D0%97%D0%B0%D1%85%D0%B0%D1%80%D0%BE%D0%B2%D0%B0%20%D0%86.%D0%92..pdf

17. Authors: Peng Hu , Rong Du , Yao Hu and Nan Li - Hybrid Item-Item Recommendation via Semi-Parametric Embedding - [Електронний ресурс] – режим доступу до ресурсу: https://www.ijcai.org/proceedings/2019/0350.pdf
18. Розподіл Бернуллі - [Електронний ресурс] – режим доступу до ресурсу: http://people.umass.edu/~biep540w/pdf/bernoulli.pdf
19. Authors: Daniel J. Russo , Benjamin Van Roy , Abbas Kazerouni , Ian Osband and Zheng Wen - Thompson Sampling algorithm - [Електронний ресурс] – режим доступу до ресурсу: https://web.stanford.edu/~bvr/pubs/TS_Tutorial.pdf
20. Author: Chris Piech - Beta Distribution - [Електронний ресурс] – режим доступу до ресурсу: https://web.stanford.edu/class/archive/cs/cs109/cs109.1166/pdfs/22%20Beta.pdf
21. Posterior probability - [Електронний ресурс] – режим доступу до ресурсу: https://astro.uni-bonn.de/~kbasu/ObsCosmo/Slides2012/Lecture3_2012.pdf
22. Client-server Architecture - [Електронний ресурс] – режим доступу до ресурсу: https://cs.uwaterloo.ca/~m2nagapp/courses/CS446/1195/Arch_Design_Activity/ClientServer.pdf
23. Бралатан Р.А, Штовба С.Д. Модифікація рекомендаційного алгоритму колаборативної фільтрації з використанням семплування Томпсона / Тези доп. конференції КТОД-2022, Вінниця: ДонНУ імені Василя Стуса. *Прийнято до друку.*

Бралатан Роман Андрійович
Прізвище, ім'я по батькові

Інформаційних і прикладних технологій
Факультет

122 Комп'ютерні науки
Шифр і назва спеціальності

Технології обробки даних (Data Science)
Освітня програма

ДЕКЛАРАЦІЯ АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ

Усвідомлюючи свою відповідальність за надання неправдивої інформації, стверджую, що подана кваліфікаційна (магістерська) робота на тему «Рекомендаційна система для користувачів інтернет-магазину на базі платформи Salesforce» є написаною мною особисто.

Одночасно заявляю, що ця робота:

- не передавалась іншим особам і подається до захисту вперше;
- не порушує авторських та суміжних прав, закріплених статтями 21-25 Закону України «Про авторське право та суміжні права»;
- не отримувалась іншими особами, а також дані та інформація не отримувалась у недозволений спосіб.

Я усвідомлюю, що у разі порушення цього порядку моя кваліфікаційна робота буде відхилена без права її захисту, або під час захисту за неї буде поставлена оцінка «незадовільно».

_____ _____

(дата) (підпис здобувача освіти)