МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему:

«Рекомендаційна система підбору фільмів»

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп'ютерні науки»**

Освітня програма **«Аналітика даних»**

Освітній рівень: бакалавр

Виконав: студент 4 курсу, групи АнД-41

Лаврій С.П.
(прізвище та ініціали)

Керівник: Мінаєва Ю.І.
(прізвище та ініціали)

Кандидат технічних наук, доцент
(науковий ступінь, звання)

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри *інтелектуальних технологій*
Протокол № 11 від 06.06.2022 р.
зав. кафедри_____ доц. Іларіонов О.Є.

Київ - 2022

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
інтелектуальних технологій
Іларіонов О.Є.

_____

"___"_____2022 р.

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Лаврій Софія Петрівна

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи)

Рекомендаційна система підбору фільмів

затверджена протоколом засідання кафедри від «23» грудня 2021 р. № 4

2. Термін здачі студентом закінченого проекту (роботи) 29 травня 2022 року

3. Вихідні дані до проекту (роботи)

Система, яка формує та видає персоналізовані рекомендації користувачу

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

1) Аналітичний оглял існуючих типів рекомендаційних систем та їх застосувань у сфері фільмів
2) Інструменти та методи реалізації рекомендаційної системи
3) Функціональний аналіз розробленої системи

5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій)

Актуальність теми (1 слайд), підходи на основі колаборативної фільтрації та на основі вмісту (2-3 слайди), алгоритм знаходження приблизних найближчих сусідів (1-2 слайди), метрики для оцінювання (2 слайди), опис даних та інтерфейсу (2-3 слайди), висновки (1 слайд)

6. Консультанти з випускної кваліфікаційної роботи із зазначенням розділів випускної кваліфікаційної роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1			
2			
3			

7. Дата видачі завдання 15 лютого 2022 року

Керівник _____ / Мінаєва Ю.І. /
(підпис) (ПІБ)

Завдання прийняв до виконання _____ / Лаврій С.П. /
(підпис) (ПІБ)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Примітка
1	Обговорення з керівником постановки завдання та змісту пояснювальної записки	15.02.2022 - 22.02.2022	
2	Аналіз постановки задачі, формалізація задачі, аналіз літературних джерел та написання 1 розділу роботи	23.02.2022 - 01.03.2022	
3	Проектно-технологічна реалізація рекомендаційної системи фільмів	01.03.22 – 10.04.22	
4	Розробка та тестування веб-сторінки для рекомендаційної системи	11.04.22 – 13.05.22	
5	Оформлення пояснювальної записки, підготовка презентації	14.05.2022 - 29.05.2022	

Студент-дипломник _____ / Лаврій С.П. /
(підпис) (ПІБ)

Керівник випускної кваліфікаційної роботи _____ / Мінаєва Ю.І. /
(підпис) (ПІБ)

Анотація

Лаврій Софія Петрівна виконала випускну кваліфікаційну роботу на тему «Рекомендаційна система підбору фільмів» за спеціальністю 122 – «Комп’ютерні науки», освітня програма «Аналітика даних»

У випускній кваліфікаційній роботі проведено аналіз сучасних підходів до формування рекомендацій, розглянуто метрики оцінювання якості наданих рекомендацій, розроблено програмне забезпечення, що надає персональні рекомендації фільмів, на основі попередніх оцінок користувачів та інтерфейс, який транслює отримані рекомендації користувачеві.

Ключові слова: рекомендаційна система, колаборативна фільтарція, контент, оцінювання.

Summary

The degree project: «Movies recommendation system» has completed by Sofiia Lavriiy specialty 122 – «Computer Scienses», educational programm «Data Analysis».

In the graduation work, the modern approaches to recommendations formation are analyzed, metrics of the estimation of quality of the given recommendations are considered. The software, which is providing personalized recommendations of films based on preliminary ratings, is developed and the interface transmits the received recommendations to the user is designed.

Keywords: recommendation system, colaborative filtering, content, evaluation.

Зміст

ВСТУП

Кількість інформації в Інтернеті настільки велика, що людина просто не здатна знайти те, що їй дійсно потрібно. Це пов’язано з тим, що останні декілька років існує тенденція переїзду великого та малого бізнесу в Інтернет. Більшості з них навіть немає вже офлайн, адже онлайн бізнес потенційно може охопити ширше коло користувачів, допомагає скоротити витрати на оренду приміщень, оплату комунальних послуг та інше. Все більше з’являється онлайн магазинів, в яких можна купити що завгодно: продукти харчування, одяг, книжки, товари для дому та безліч інших одиниць продукції. Як і звичайний бізнес такі платформи постійно ростуть та розвиваються, розширюють свій асортимент, покращують якість товарів і обслуговування. Всю цю інформацію можна і потрібно обробляти. У зв’язку з цим виникає необхідність з одного боку допомогти користувачеві підібрати товари, які його можуть зацікавити, а з іншого боку – підвищити рівень доходу сервісу. Цю функцію виконують рекомендаційні системи.

Це інформаційні системи, завдання яких, запропонувати користувачеві дії, послуги або товари на основі попередніх переваг його або користувачів, схожих на нього за купленим товарами / послугами. У зв’язку з поширеністю рекомендаційних систем різних типів, тема є актуальною.

Для такої предметної області як фільми рекомендаційні системи є надзвичайно важливими, адже дуже часто користувачі заходять на такі платформи не знаючи конкретно, що саме хочуть подивитись. І задача рекомендаційних систем запропонувати такі підбірки фільмів, аби користувач залишився задоволеним. З боку бізнесу це принесе дохід сервісу, адже такі платформи зазвчай мають місячну або річну підписку, яка і є

основним джерелом їх доходу. Відповідно, від якості рекомендацій буде залежати чи продовжить користувач користуватись даним сервісом, чи ні.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ТИПІВ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

Рекомендаційні системи є одними з найуспішніших і найпоширеніших застосувань технологій машинного навчання в бізнесі. Вони застосовуються у випадках, коли багато користувачів взаємодіють з багатьма елементами.[1]

Система рекомендацій – це система, які розраховує та прогнозує наближену до реальності оцінку, яку користувач може дати певному продукту. Ці прогнози потім формуються в підбірки та пропонуються користувачеві.

Системи рекомендацій стали ключовим компонентом багатьох онлайн-сервісів, таких як електронна комерція, соціальні мережі, служба новин або онлайн відео трансляції. Вони часто використовуються різними великими компаніями, такими як Google, Instagram, Spotify, Amazon, Reddit, Netflix тощо, щоб збільшити взаємодію з користувачами та платформою. Наприклад, Spotify рекомендує пісні, подібні до тих, які неодноразово слухав користувач. щоб максимально зберегти аудиторію для продовження користування їхньою платформою для прослуховування музики.[11] Amazon використовує рекомендації, щоб пропонувати продукти різним користувачам на основі даних, які вони зібрали для цього користувача.[3]

1.1 Типи рекомендаційних систем

Традиційно підходи до систем рекомендацій можна розділити на такі широкі категорії: колаборативна фільтрація, фільтрація вмісту та гібридні системи рекомендацій. Зовсім недавно були запропоновані деякі варіанти, які враховують положення користувача (рекомендація з урахуванням

географії), послідовність взаємодій користувача (послідовна рекомендація) та взаємодії поточного сеансу користувача для передбачення наступного кліку (рекомендація на основі сеансу).

1.1.1 Системи колаборативної фільтрації

Колаборативна фільтрація (*eng.* collaborative filtering) – це метод прогнозування інтересів користувача шляхом визначення ступеня схожості користувачів та їх уподобань. Основна ідея, що лежить в основі колаборативної фільтрації, полягає в тому, що якщо користувачі A і B мають схожі смаки в одному продукті, то вони, швидше за все, мають схожі смаки і в інших продуктах.

Існують два поширені типи підходів у спільній фільтрації: на основі пам’яті та на основі моделі.

Основна ідея підходів, що засновані на пам’яті, які також називаються спільною фільтрацією сусідства, тому що рейтинги прогнозуються на основі найближчих користувачів/елементів. Цей підхід можна ще розділити на фільтрацію на основі користувачів і елементів. Перше означає, що рекомендації підбираються відповідно до інтересів користувачів-однодумців, а друге рекомендує елементи на основі їх подібності між собою.

Підходи на основі моделі – це прогнозні моделі, що використовують машинне навчання. На вхід до такої моделі подається матриця оцінок, яка передається в оригінальному або зміненому вигляді. Далі методи оптимізації такі, як дерева рішень, матрична факторизація[5] та її модифікації, або підходи, що засновані на побудові асоціативних правил видають прогнозовані оцінки користувачів на той чи інший елемент.

Яскравим прикладом роботи системи на основі колаборативної фільтрації є Coursera: алгоритм підбирає рекомендовані для користувача курси на основі інших осіб, які закінчили такі ж курси.

Основною перевагою використання моделей спільної фільтрації є їх простота впровадження та широке охоплення елементів, яке вони забезпечують. Окрім того, такі системи не вимагають розуміння змісту об’єкта: будь то фільм, готель, книга чи товар в магазині.

Головним недоліком цієї моделі є те, що вона не є зручною для рекомендацій нових елементів або користувачів, які з’являються в системі, адже з ними не було ще ніякої взаємодії. Це називається проблемою холодного старту. Також, відомо, що алгоритми на основі пам’яті погано працюють на дуже розріджених наборах даних.

1.1.2 Системи на основі контенту

Даний тип систем генерує рекомендації на основі уподобань та профілю користувачів, намагаючись підібрати подібні елементи до тих, які їм подобалися раніше. Рівень схожості між елементами зазвичай встановлюється на основі їх атрибутів. На відміну від більшості моделей спільної фільтрації, які використовують оцінки між цільовим користувачем та іншими користувачами, моделі на основі вмісту зосереджуються на оцінках, наданих тільки цільовим користувачем. По суті, підхід на основі вмісту використовує різні джерела даних для створення рекомендацій.

Найпростіші форми систем на основі контенту вимагають метаданих об’єкту та явних або неявних реакцій користувача на елемент.

Метадані об’єкту – це атрибути елементу. Для фільмів це може бути жанр, режисер, країна виробник і т.д. Чим більше є інформації про елемент, тим краще буде передбачення системи.

Зворотній зв’язок від користувача може бути явним: оцінка, лайк або дизлайк, або неявним: перегляд трейлеру до фільма, прочитання опису фільму та інше. Чим більше користувач взаємодіє з платформою, тим легше зрозуміти його вподобання.

Прикладом такої системи слугує Amazon вони рекомендують продукти, подібні до тих, які ви користувач купував раніше. Ще одним яскравим прикладом є Reddit. Сайт використовує алгоритмічні підходи, щоб рекомендувати користувачам нові публікації на своїй платформі. Їхні рекомендації обмежені часом публікації, кількістю лайків та дизлайків, коментарів тощо. Це враховується у формулу для отримання оцінки публікації.

Моделі, засновані на вмісті, є найбільш вигідними для рекомендацій елементів, коли недостатньо доступних даних про оцінку. Це пов’язано з тим, що інші елементи з подібними атрибутами могли бути оцінені користувачем. Таким чином, модель повинна мати можливість використовувати оцінки разом із атрибутами елемента, щоб генерувати рекомендації, навіть якщо даних не так багато.

Існує два основних недоліки систем на основі контенту.

Перший – надані рекомендації є «очевидними» , бо базуються на основі елементів / вмісту, які вже споживав користувач. Це є недоліком, оскільки якщо користувач ніколи не взаємодіяв з певним типом елементу, він ніколи не буде рекомендований користувачеві. Це зменшує різноманітність рекомендацій, що є негативним результатом для багатьох підприємств.

Другий – вони неефективні для надання рекомендацій для нових користувачів. Під час створення моделі потрібна історія явних/неявних даних рівня користувача для елементів. Як правило, важливо мати великий набір даних рейтингів, щоб робити надійні прогнози.

1.1.3 Системи засновані на послідовності

Такі системи використовують послідовність взаємодій користувача з елементами в рамках сеансу в процесі формування рекомендацій. Це може бути передбачення наступного товару в кошику для покупок в Інтернеті, наступного відео для перегляду або наступного пункту призначення мандрівника і т.д.

Один з алгоритмів Netflix працює, базуючись саме на такому підході: використовується послідовність дій користувача, а також поточний контекст, щоб передбачити ймовірність наступної дії. Вони навчили модель передбачати, що користувач ймовірно захоче дивитися далі надавши певну інформацію про користувача: країну, пристрій, дату й час, коли він дивився фільм.[2]

1.1.4 Гібридна рекомендаційна система

Різні методи рекомендаційних систем мають свої переваги і недоліки. Часто багато з цих методів можуть здаватися обмежуючими, якщо вони використовуються ізольовано, особливо якщо для вирішення проблеми доступно кілька джерел даних. Гібридні рекомендаційні системи – це системи, розроблені для використання різних доступних джерел даних для створення більш надійних та точних прогнозів.

Гібридні рекомендаційні системи мають дві переважні конструкції – паралельну та послідовну. Паралельний дизайн забезпечує вхідні дані для кількох рекомендаційних систем, кожна з цих рекомендацій об’єднується для створення одного виходу. Послідовна конструкція надає вхідні параметри єдиному механізму рекомендацій, а вихідні дані послідовно передаються

наступному рекомендаційному. На рисунку 1.1 наведено візуальне представлення обох конструкцій.

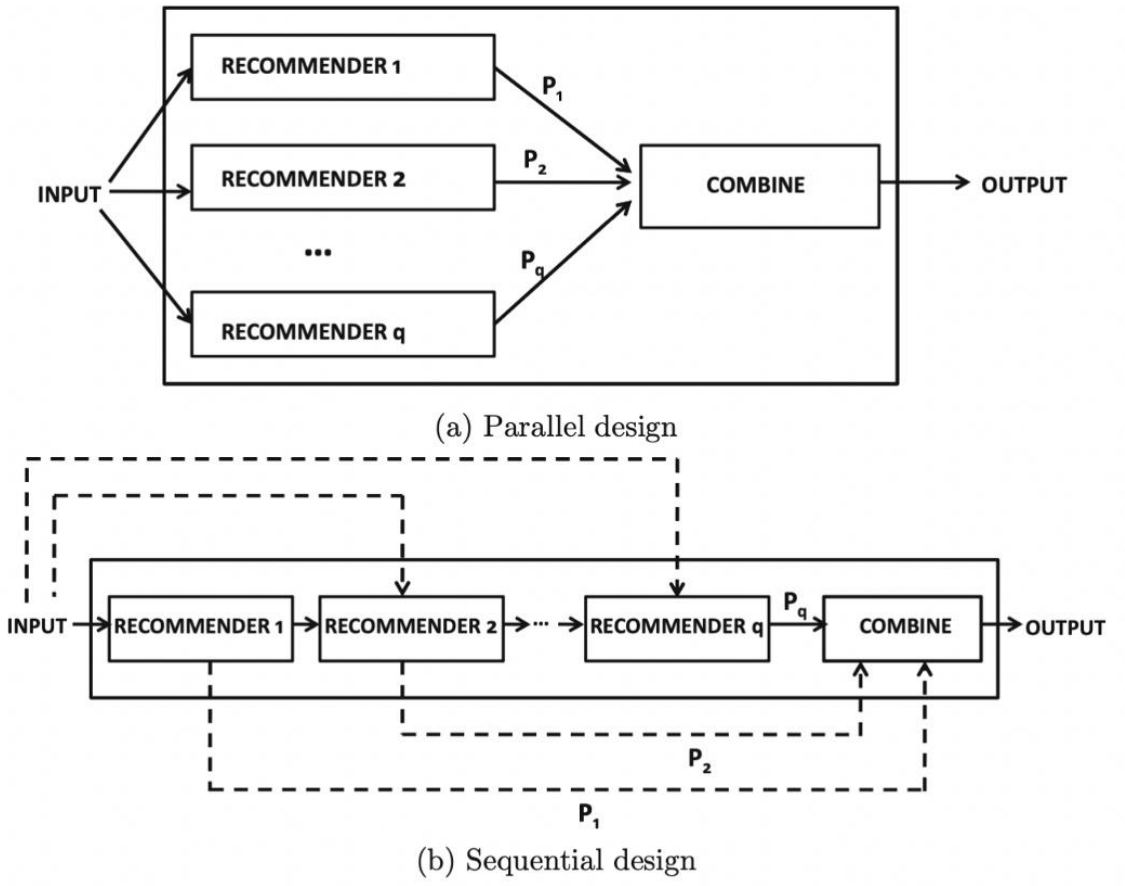

Рисунок 1.1 Схема різних видів гібридної рекомендаційної системи. a) Паралельна робота алгоритмів; b) Послідовна робота алгоритмів

Гібридні системи поєднують різні моделі для боротьби з недоліками однієї моделі з іншою. Це дає більш надійні та персоналізовані рекомендації для користувачів. Проте, ці типи моделей, як правило, мають високу обчислювальну складність і вимагають великої бази даних рейтингів та інших атрибутів для оновлення. Без актуальних показників (залучення користувачів, оцінки тощо) це ускладнює перенавчання та надання нових

рекомендацій з оновленими елементами та оцінками від різних користувачів.[3]

Як узагальнення всіх типів рекомендаційних та їх підвидів наведено рисунок 1.2:

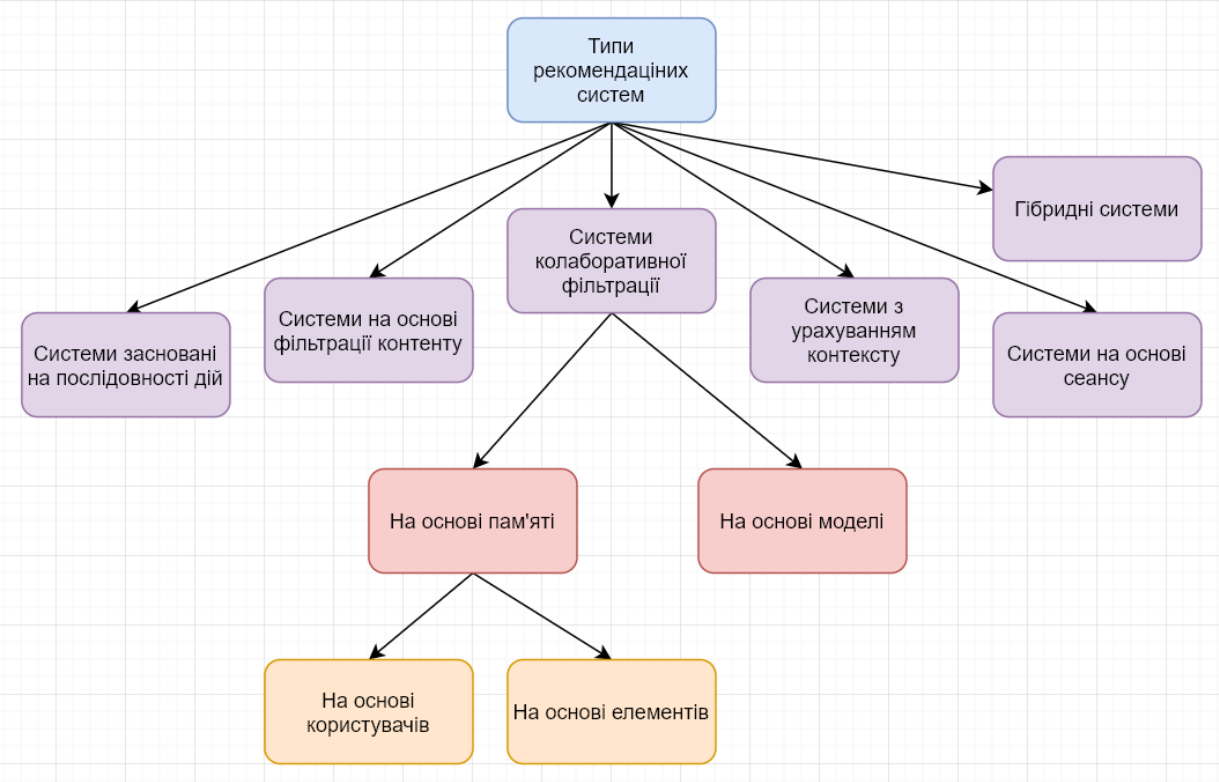

Рисунок 1.2 Схема розгалуження типів рекомендаційних систем

1.2 Проблеми рекомендаційних систем та способи їх усунення

Система веб-рекомендацій страждає від багатьох проблем, таких як зміна інтересів користувача, шахрайство та інші. Найпоширеніші з них – це холодний старт, конфіденційність, надмірна спеціалізація, масштабованість, розрідженість та передбачуваність.

Основними проблемами, які впливають на якість прогнозу ймовірної оцінки та відповідно рекомендацій є:

- Холодний старт

Проблему холодного запуску можна розділити на дві категорії: холодний запуск нових елементів і нових користувачів. Проблема з холодним старту для елемента виникає, коли недостатньо попередніх оцінок, пов’язаних із цим елементом. Аналогічно важко рекомендувати товари новим користувачам, оскільки система не має жодної інформації, пов’язаної з його попередніми покупками, оцінками, відкугами.

Щоб обмежити проблему холодного запуску, можна використовувати демографічну інформацію користувача, тобто використовувати спільну фільтрацію, щоб формувати рекомендації.

- Масштабованість

Оскільки кількість користувачів і елементів зростає, системі потрібно більше ресурсів, щоб дати користувачам найточніші рекомендації. Більшість ресурсів використовується з метою визначення користувачів зі схожими смаками та предметів зі схожими атрибутами. Відповідно зростання масштабів галузевих наборів даних і використання складніших моделей підняло планку обчислювальних ресурсів, необхідних для рекомендаційних систем.

- Розрідженість

Однією з головних проблем при побудові рекомендаційних систем є саме розрідженість даних. Дане явище виникає, тому що зазвичай на платформах багато користувачів і велика кількість представлених об’єктів. В результаті їх взаємодії формується матриця, проте більшість її елементів рівні нулю, тому що не зафіксовано ніякого фідбеку від споживача.[10]

- Конфіденційність

Конфіденційність також є великою проблемою в контексті демографічних рекомендаційних систем. Щоб дати найбільш точні рекомендації, система повинна отримати відповідну інформацію про користувача, включаючи персональні дані та дані про місцезнаходження конкретного користувача, що може порушити конфіденційність користувача.

- Надмірна спеціалізація

Хороша система рекомендацій повинна пропонувати різноманітні елементи, чого, наприклад, система на основі вмісту зробити не може. Як результат це заважає користувачам відкривати щось нове та відмінне, а натомість рекомендують предмети, з якими вони вже знайомі.

Дану проблему можна подолати за допомогою гібридної системи рекомендацій, де буде використовуватись колаборативна фільтрація. Вона якраз забезпечить різноманітність вибору так як використовується не тільки взаємодія користувача, а й інших для формування рекомендацій.

- Передбачуваність

Ще одна проблема, з якою зазвичай стикаються системи рекомендацій у наші дні – це передбачуваність. Навіть якщо елементи, рекомендовані користувачеві, різноманітні, вони можуть бути знайомі користувачеві. Наприклад, система рекомендує лише бестселери. Рекомендації в цьому випадку дійсно різноманітні, але користувач, можливо, вже знайомий ними.[9]

1.3 Приклади застосування рекомендаційних систем

Netflix – найвідоміша та найпоширеніша платформа перегляду фільмів. Основне завдання рекомендаційної системи в Netflix полягає в тому, щоб допомогти нашим учасникам знайти контент, який вони будуть дивитися та

насолоджуватися, щоб максимально довгостроково залишатися задоволеними. Це складна проблема з багатьох причин, у тому числі через те, що кожна людина унікальна, має безліч інтересів, які можуть змінюватися в різних контекстах, і потребують системи рекомендацій, коли вони не впевнені, що вони хочуть дивитися. Оскільки місячна підписка є послугою, задоволеність учасників тісно пов’язана з ймовірністю того, що людина продовжить користування послугою, що безпосередньо впливає на дохід компанії. Таким чином, цінність системи рекомендацій може бути виміряна збільшенням утримання членів.

У Netflix немає єдиної моделі, яка б керувала всіма рекомендаціями, а скоріше набір методів, де всі узгоджені з метою підвищення рівня задоволеності користувачів. Експериментуючи з різними видами рекомендаційних алгоритмів, Netflix виявили, що не існує універсального варіанту. Найефективніший метод залежить від конкретної рекомендаційної задачі, яку потрібно вирішити, а також від наявних даних. З цієї причини для створення персоналізованих рекомендацій для різних частин домашньої сторінки Netflix використовуються різні моделі машинного навчання.

Основний спосіб взаємодії учасника з рекомендаціями – це домашня сторінка, яку вони бачать, коли заходять на Netflix на будь-якому пристрої. Основна функція домашньої сторінки – допомогти кожному споживачу легко знайти те, що йому сподобається. Проблема полягає в тому, що каталог фільмів Netflix містить набагато більше відео, ніж можна відобразити на одній сторінці, і кожен учасник має свій унікальний набір інтересів. Таким чином, загальним алгоритмічним завданням стає те, як найкращим чином налаштувати домашню сторінку кожного учасника, щоб зробити її актуальною, охопити їхні інтереси та наміри, і при цьому дозволити вивчати наш каталог.[7]

Рисунок 1.3 Вигляд домашньої сторінки Netflix

На малюнку вище зображено домашню сторінку Netflix, де перераховано різні рекомендаційні завдання, кожне з яких працює на основі окремого алгоритму. Наприклад, існує спеціальний алгоритм (1) для вибору першого відео, яке відображатиметься на видному місці у верхній частині домашньої сторінки. Окремий спосіб формування рекомендацій використовується для ранжування вже переглянутих відео, які користувач може захотіти продовжити переглядати (7), ще інший – щоб допомогти споживачам відкривати нові відео (5).

Вихід результат кожного з цих алгоритмів можна представити як різні рядки рекомендованих відео на домашній сторінці. У службі Netflix є ще кілька завдань персоналізації. Наприклад, існує алгоритм (4, 6), який

вибирає, які рядки відображати персоналізованим способом, щоб створити структуру домашньої сторінки[7]. Крім того, повідомлення та сповіщення (3), надіслані користувачам платформи, також персоналізовані. Більш того, Netflix використовує методи рекомендацій як частину пошукової (2) системи[6].

Зараз домашня сторінка Netflix на більшості пристроїв структурована з відео (фільми та телешоу), організованих у тематично узгоджені рядки, представлені у двовимірному макеті. Учасники можуть прокручувати або горизонтально в рядку, щоб побачити більше відео в цій підбірці, або вертикально. Такий макет навігації дає можливість учасникам легко пропускати цілі групи вмісту, які можуть не відповідати їх поточним намірам. Це дозволяє учаснику мати як релевантність, так і різноманітність.

Крім того, при формуванні домашньої сторінки також важливо враховувати, як учасники переміщуються сторінкою і на які позиції на сторінці вони, ймовірно, звернуть увагу. Розміщення найрелевантніших відео в позиціях, які найімовірніше будуть побачені, як правило, у верхньому лівому куті, має скоротити час пошуку для користувача. [7]

Крім рекомендаційного завдання, доступні дані та їх властивості мають вирішальний вплив на те, який алгоритм рекомендацій добре працює. Перша і найважливіша відмінність полягає в тому, чи містять дані лише взаємодію користувача з елементом, чи містять додаткову інформацію: атрибути користувача та елемента, контекстна інформація щодо взаємодії споживача з з платформою. Було з’ясовано, що більш складні моделі, включаючи алгоритми глибокого навчання, краще працюють, коли дані підкріплені такою додатковою інформацією.[4]

1. 4 Постановка задачі

Метою дипломної роботи є розробка рекомендаційної системи фільмів для полегшення вибору фільмів та збільшення рівня задоволеності користувача.

Завдання дипломної роботи – обробка оцінок, знаходження користувачів зі схожими інтересами та формування персоналізованої підбірки рекомендацій та розробка інтерфейсу для взаємодії учасників з платформою.

Об’єкт дослідження – рекомендаційні системи фільмів на основі колаборативної фільтрації та вмісту.

Предмет дослідження – оцінки користувачів на фільми та зв’язки між ними.

Для користувача рекомендаційна система має вигляд «чорного ящика», так як він бачить лише її результат – власне рекомендації. На рисунку 2.4 зображено діаграму IDEF0. Вхідними даними, як вже неодноразово було описано вище, є оцінки користувачів, їх взаємодія з платформою. Також, передається інформація про споживача та метадані фільмів. Управління такою системою виконують алгоритми формування рекомендацій та користувач, адже його дії безпосередньо впливають на результат. Механізмами є вимоги платформи для перегляду фільмів. Так як користувач використовує свої персональні дані для створення профілю, який потім використовується для формування рекомендацій, ще двома механізмами виступають закон про конфіденційність персональних даних та право сервісу на їх зберігання та обробку. Результатом є персональна підбірка рекомендацій фільмів.

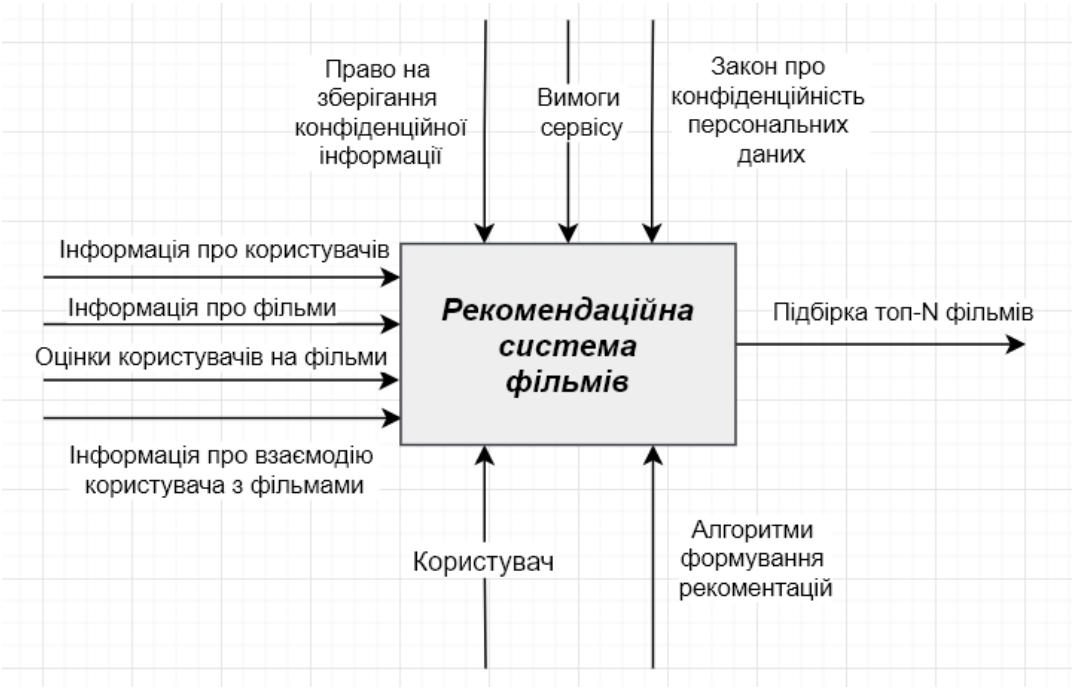

Рисунок 1.4 Схема IDEF0 “Рекомендаційна система фільмів”

На рисунку 1.5 наведено дерево цілей рекомендаційної системи фільмів. Основними функціями є збір даних, їх обробка та власне формування рекомендацій. Збір даних можна деталізувати як формування таблиць з метаданими фільмів, даними користувачів та їх оцінками на фільми. Обробка даних поділяється на такі підпроцеси як формування матриці оцінок, її нормалізація та підготовка до подання на вхід в нейронну мережу. Останнім етапом є формування рекомендацій, яке складається з 3 підфункцій: подання даних на вхід, застосування визначених алгоритмів машинного або ж глибинного навчання та формування підбірки топ-N персоналізованих рекомендацій.

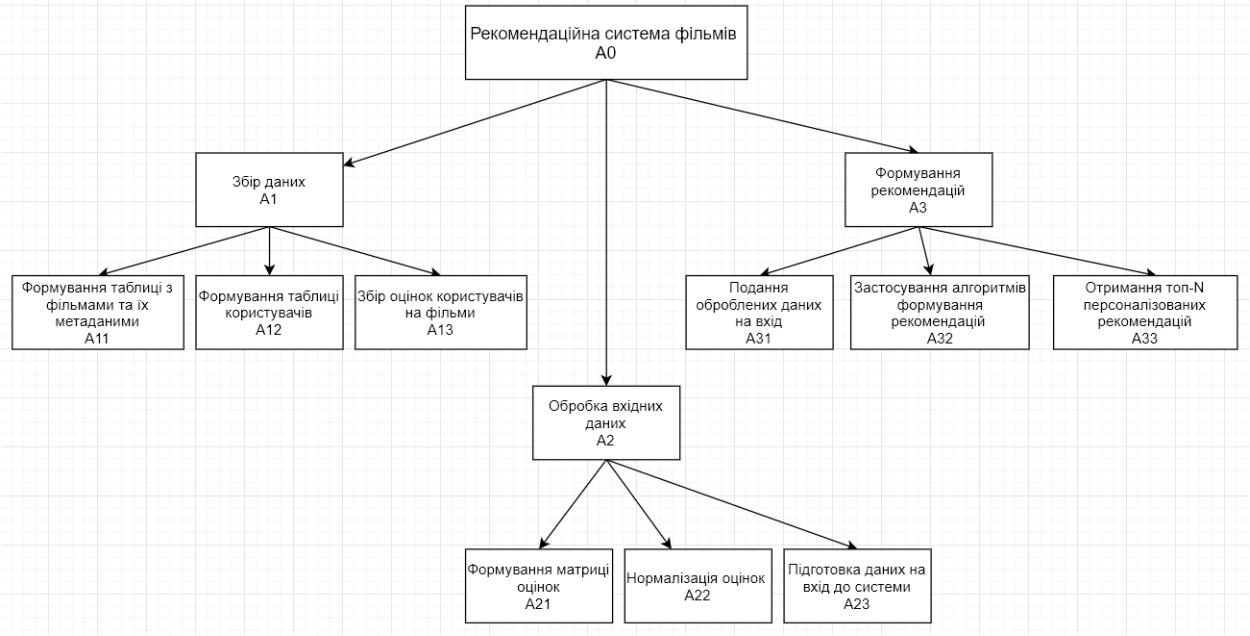

Рисунок 1.5 Дерево функцій рекомендаційної системи фільмів

1.5 Опис вимог

1.5.1 Функціональні вимоги

Функціональні вимоги до рекомендаційної системи:

- зручний та зрозумілий інтерфейс, який буде реалізовано англійською мовою
- надання топ-N найпопулярніших фільмів
- надання топ-N фільмів з найвищою середньою оцінкою
- надання топ-N персоналізованих рекомендацій
- можливість додати фільм у список “До перегляду” та видалити його звідти
- вхід в особистий кабінет або реєстрація, якщо ще немає профілю
- можливість залишити оцінку на фільм
- пошук фільмів за ключовим словом
- можливість перейти на сайт IMDB для перегляду фільму

- Можливість переглянути список фільмів, які користувач вже бачив та оцінив

1.5.2 Нефункціональні вимоги

До нефункціональних вимог відноситься наступні:

- Надійність: якщо користувач робить некоректні дії, то система буде реагувати на це без збоїв.
- Швидкодія: рекомендації будуть надаватись в термін до 30 секунд.
- Коректна відповідь системи на будь-які дії користувачів.
- Система буде в робочому стані 24 години на добу та матиме зв’язок з базою даних.

1.6 Висновки

Отже, у даному розділі було детально розглянуто тему рекомендаційних систем. Описано різні типи, їх особливості, переваги та недоліки. Проаналізовано, які проблеми виникають та які можливі способи їх усунення. Також наведено приклади їх застосування в різних сферах: комерція, книги, подорожі, ресторани, музика, навчальні платформи та багато іншого. Осболиву увагу було приділено фільмам: в роботі було проведено аналітичний огляд найвідомішої стрімінгової платформи Netflix. Розглянуто їх підходи до формування різних частин домашньої сторінки: окремі алгоритми для пошукових запитів користувачів, для розсилки електронних листів, для виведення відео, яке буде на весь екран та для інших елементів сторінки.

У розробці точної рекомендаційної системи заінтересованими сторонами є як користувач, так і стрімінгові платформи. Перші отримують якісні рекомендації, які задовольняюсь їх інтереси. А другі – отримують аудиторію, а відповідно й заробіток.

Окрім того, в даному розділі були визначені постановка задачі, предмет та об’єкт дослідження. Описані функціональні та нефункціональні вимоги до розроблюваної системи та інтерфейсу. Також, побудовано схеми IDEF0 та дерево цілей. “Чорна скринька” описує, які дані потрібні для системи, хто та що впливає на формування рекомендацій та якими механізмами керується система. Друга схема деталізує процес збору даних, їх обробку та формування рекомендацій.

РОЗДІЛ 2. ІНСТРУМЕНТИ ТА МЕТОДИ РЕАЛІЗАЦІЇ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ

2.1 Опис інструментів для реалізації програмної частини

2.1.1 Python

Для розробки рекомендаційної системи було обрано мову програмування Python. Це мова високого рівня з відкритим вихідним кодом, є інтерпретованою, і забезпечує чудовий підхід до об'єктно-орієнтованого програмування, що є важливим, адже методи формування рекомендацій в даній роботі побудовані саме на класах. Окрім того, Python надає чудові функціональні можливості для роботи з математикою, статистикою та науковими функціями. Це найпоширеніша мова для реалізації проектів у сфері наук про дані, оскільки він надає велику колекцію бібліотек, які допомагають вирішувати складні проблеми, будувати потужні системи і програми.

Python – це мова програмування, яка дуже швидко масштабується. Це означає, що Python має все більше і більше можливостей. Однією з таким можливостей є широкий вибір бібіліотек та фреймворків.

Бібліотеки, які було використано для написання програмної частини:

1) Numpy — це бібліотека, в якій реалізовано математичні функції для обробки масивів великих розмірів. NumPy розшифровується як числовий (numerical) Python. Він був створений у 2005 році Тревісом Оліфантом. Дана бібліотека пропонує багато корисних функцій для операцій з n-масивами та

матрицями типу NumPy, які мають високу продуктивність та швидке виконання. [30]

2) Pandas є однією з найпопулярніших бібліотек Python для маніпуляцій з даними та їх аналізу. У 2008 році в AQR Capital Management почалася розробка цього модулю. Основі функції Pandas:

- швидкий та ефективний об'єкт DataFrame з інтегрованим індексуванням;
- інструменти для читання та запису даних між структурами даних у пам’яті та різними форматами: CSV та текстові файли, Microsoft Excel, бази даних SQL та швидкий формат HDF5;
- гнучка зміна розмірності та типів даних;
- агрегування або перетворення даних: групування, об’єднання, злиття т.д.;
- вбудована візуалізація даних.[29]

3) Sklearn — це бібліотека Python для машинного навчання. Даний проект було розпочато ще в 2007 році як частина Google Summer of Code, а перший публічний випуск був зроблений на початку 2010 року. scikit-learn — це пакет машинного навчання Python з відкритим вихідним кодом, який пропонує функціональні можливості для підтримки навчання моделей. Крім того, він надає інструменти для розробки, вибору та оцінки моделі, а також багато інших утиліт, включаючи функціональні можливості попередньої обробки даних. Більш конкретно, основна функціональність scikit-learn включає класифікацію, регресію, кластеризацію, зменшення розмірності, вибір моделі та попередню обробку. Бібліотека дуже проста у використанні та, головне, ефективна, оскільки вона побудована на NumPy , SciPy та matplotlib.[31]

2.1.2 Flask

Ще однією перевагою мови Python є можливітсь веб-дизайну. Існує два основних фреймворки: Django і Flask. Для створення інтерфейсу було обрано мікрофреймворк Flask. Він був представлений Арміном Ронахером у 2011 році як пробний метод об’єднання двох рішень, тобто Werkzeug (фреймворк сервера) та Jinja2 (бібліотека шаблонів).

Flask класифікується як мікрофреймворк, оскільки він не залежить від зовнішніх бібліотек для виконання певних завдань. Він має свої інструменти, технології та бібліотеки для підтримки функцій розробки веб-додатків. Оскільки цей фреймворк більш незалежний і гнучкий, вважається, що краще починати з Flask. Саме тому його було і обрано для розробки інтерфейсу рекомендаційної системи.

2.1.3 Tableau

Tableau було засновано в 2003 році в результаті проекту з інформатики в Стенфорді, який мав на меті покращити потік аналізу та зробити дані більш доступними для людей за допомогою візуалізації. Співзасновники Кріс Столте, Пет Ханрахан і Крістіан Шабот розробили та запатентували основну технологію Tableau, VizQL, яка візуально виражає дані шляхом перекладу дій перетягування в запити даних через інтуїтивно зрозумілий інтерфейс. Додаток має підтримку широкого кола СУБД та різних типів файлів для імпорту даних. Також Tableau пропонує великий вибір діаграм та широкий функціонал інструментів. [28]

2.1.4 PostgreSQL

PostgreSQL — це розширена реляційна база даних корпоративного класу з відкритим кодом, яка підтримує запити як SQL (реляційні), так і JSON

(нереляційні). Це високостабільна система керування базами даних, що підтримується більш ніж 20-річним розвитком спільноти, що сприяло її високому рівню стійкості, цілісності та коректності. PostgreSQL використовується як основне сховище даних або сховище даних для багатьох веб-, мобільних, геопросторових та аналітичних додатків.

PostgreSQL підтримує більшість провідних мов програмування та протоколів, у тому числі і Python.

2.2 Опис методів для знаходження найближчих сусідів

Алгоритми, які реалізовані в роботі основані на пошуку користувачів схожих за інтересами або фільми, які подібні між собою. Дані, які описуються користувача або фільм кодуються у вектори. Для пошуку найближчих сусідів існує декілька способів. Найпростіший - це “виснажливий” пошук: для заданого вектора обчислювати відстань до всіх векторів та відбирати n найближчих. Проте більшість сучасних програм мають масивні набори даних з високою розмірністю (сотні чи тисячі), тому лінійне сканування займе деякий час та споживатиме багато пам’ті. Це єдиний точний спосіб, але доволі ресурсозатратний.

Одним з вирішенням цієї проблеми є використання методу знаходження приблизного сусіда. Такий пошук може бути на порядки швидшим, але варто усвідомлювати, що можуть бути певні втрати точності. Задача в тому, аби побудувати структуру даних, яка дозволить знайти найближчі точки до будь-якої точки запиту за сублінійний час.

Методи пошуку приблизного найближчого сусіда (Approximate Nearest Neighbors) прискорюють пошук шляхом попередньої обробки даних у ефективний індекс. Алгоритми на основі дерева є однією з найпоширеніших

стратегій в роботі з ANN. Вони створюють ліси (колекції дерев) як свою структуру даних, розбиваючи набір даних на підмножини.

Одним з найвідоміших рішень є Annoy [25], бібліотека, ка використовується, щоб підібрати музичні рекомендації в Spotify.

Для побудови індексу в Annoy кожне дерево формується так:

- Випадковим чином обираються дві точки
- Гіперплощиною, яка є рівновіддаленою до обраних точок, простір розбивається на два менших
- Рекурсивно продовжується поділ на підпростори, поки кількість точок в кожному підпросторі не буде менша або рівна заданому числу точок.[24]

Детальніше розглянемо приклад побудови індексу на двовимірному прикладі. Обирається 2 випадкові точки та проводиться гіперплощина рівновіддалена до цих точок. Внаслідок поділу утворюється 2 менших простори (рис. 2.1а). Для кожного з утворених просторів продяться аналогічні дії.

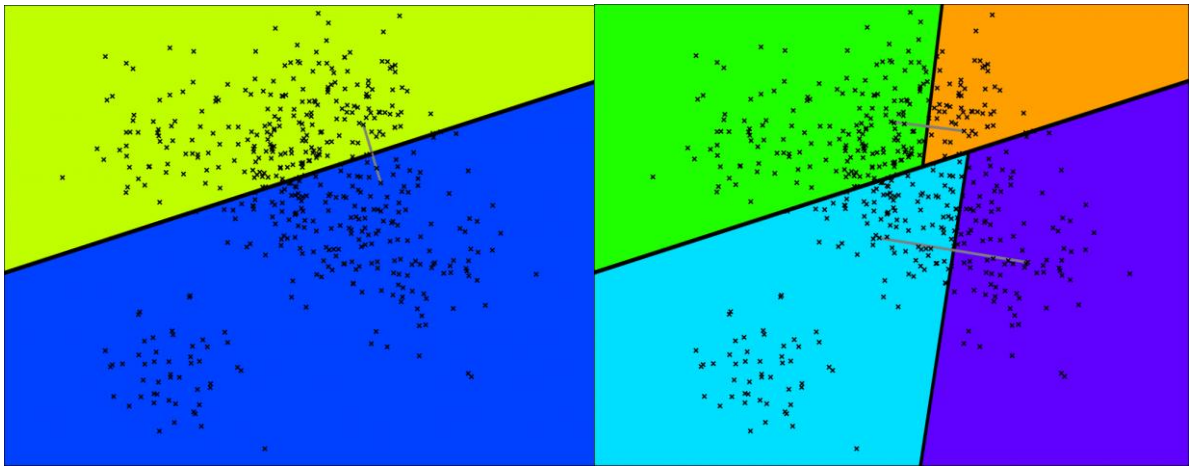

Рисунок 2.1 а) перший поділ простору; б) 2 ітерація поділу

Маючи такий поділ простору (рис. 2.1б) бінарне дерево пошуку виглядає так:

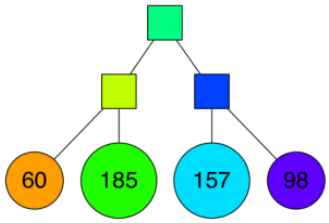

Рисунок 2.2 Бінарне дерево пошуку після 2 ітерацій алгоритму Annoy

Поділ продовжується доки в кожному вузлі не залишиться щонайбільше K елементів. Для K=10 простір має вигляд:

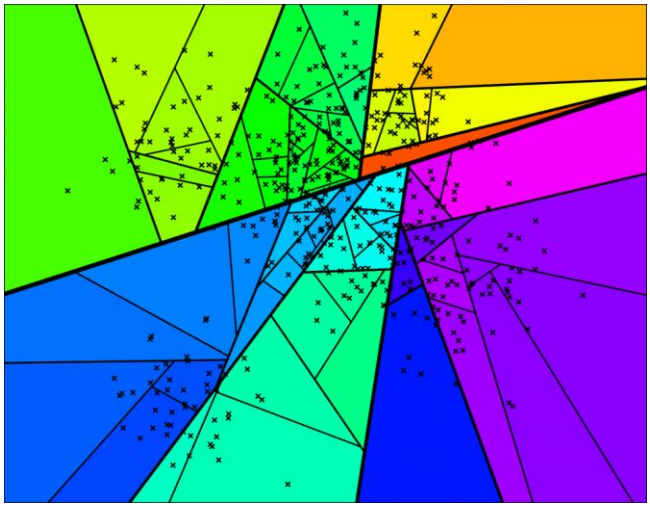

Рисунок 2.3 Результуючий простір

З відповідним бінарним деревом:

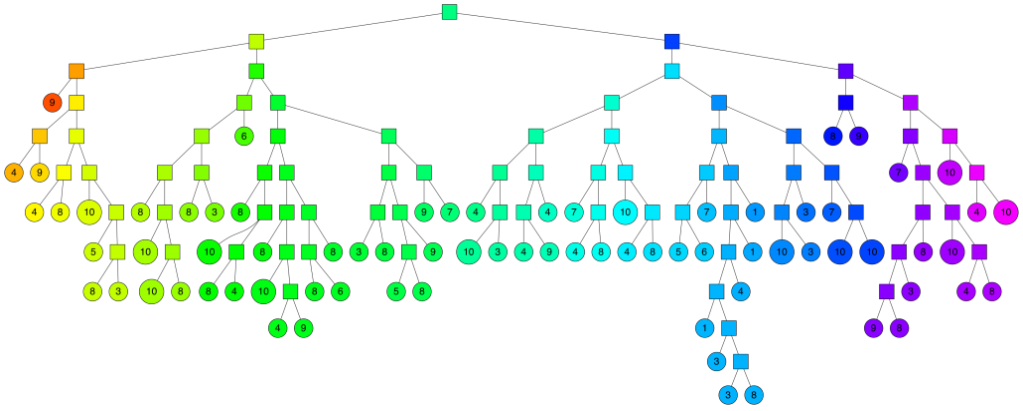

Рисунок 2.4 Результуюче бінарне дерево

Отримавши такий результат можна легко знайти будь-яку точку: достатньо обійти отримане двійкове дерево від кореня. Кожен проміжний вузол (маленькі квадратики) визначає гіперплощину і таким чином можна з’ясувати, по якій стороні гіперплощини потрібно йти. Пошук точки можна виконати за логарифмічний час, оскільки це висота дерева.

Для того, аби розширити коло сусідів використовується 2 основних прийоми: пріоритетна черга та ліс дерев.

Суть використання пріоритетної черги полягає в тому, щоб опуститися по обидві сторони гіперплощини, якщо точки в сусідньому просторі доволі близькі. Тому замість того, щоб просто йти по одному шляху бінарного дерева, алгоритм в такому випадку спускається ще кількома, як показано на рис.2.5 :

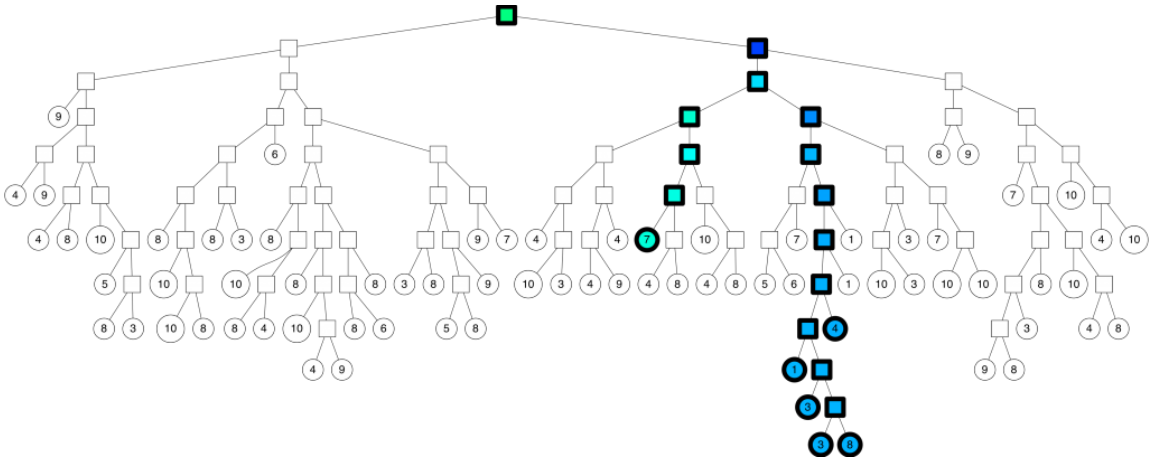

Рисунок 2.5 Шляхи пошуку сусідів заданої точки з пріоритетною чергою

Другий підхід для розширення кола сусідів - побудувати багато дерев, або ліс. Додаючи більше дерев Annoy має більше шансів знайти необхідні точки. Кожне дерево будується за допомогою випадкового набору паралельних розщеплень. Відповідно шукана точка потрапить в різні

підпростори, які потім об’єднаються в один, як на рис. 2.6а. Звузивши коло точок для вибору до n сусідів зостовується виснажливий пошук - знаходження відстаней до кожної з запропонованих точок. Після цього точки ранжуються по мірі близькості до цільової точки та відбирається k найближчих сусідів.

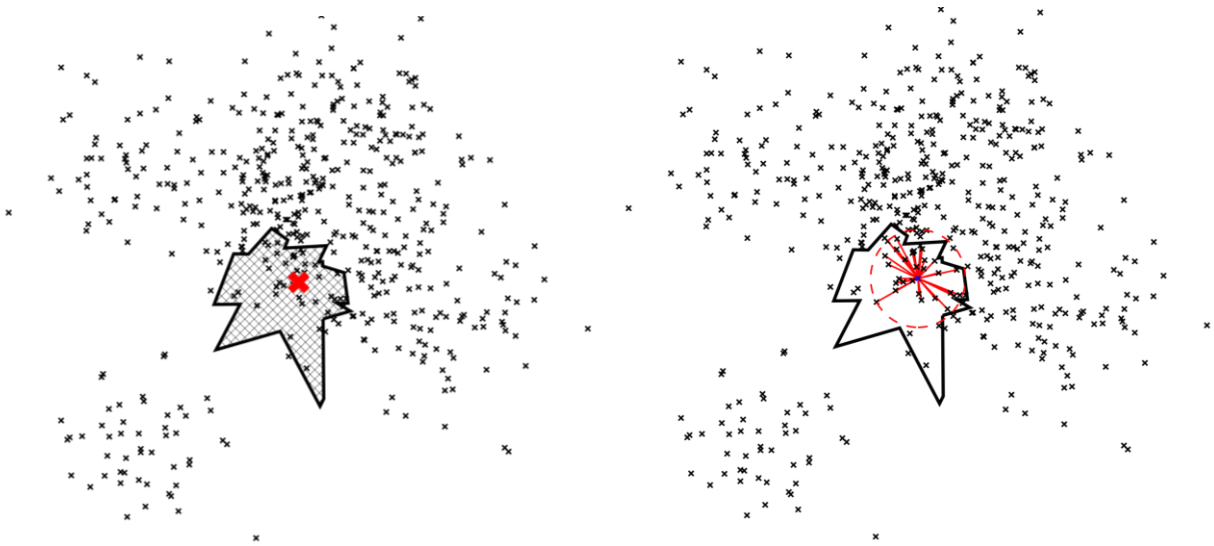

Рисунок 2.6 а) Область точок, які є потенційними сусідами; б) Знаходження відстаней від цільової точки до всіх точок в області та вибір кращих

Як видно на рис. 2.6б алгоритм знаходить майже всіх найближчих сусідів в околі точки, проте не всіх. Але A в Annoy означає “приблизний”, і упущення кількох сусідів допустимо, адже ідея приблизних алгоритмів полягає в тому, що пожертвування невеликою точністю може дати великий приріст продуктивності. [26]

2.3 Метрики вимірювання якості та точності рекомендацій

З появою рекомендаційних систем виникла потреба в оцінювання наданих рекомендацій, адже для бізнесу важливо, щоб товари, які пропонує система користувачу були релевантні та в результаті приносили додатковий прибуток. Нижче наведено різні аспекти для оцінювання рекомендаційних систем.[15]

2.3.1 Точність прогнозного рейтингу

Показники точності прогнозного рейтингу обчислюють похибку між прогнозованою алгоритмом оцінкою та фактичним рейтингом користувача. Вони є найважливішими показниками оцінки рекомендаційної системи. Середня абсолютна похибка (MAE) – це показник, який обчислює абсолютну похибку між фактичним рейтингом користувача та прогнозованим рейтингом. Формульний вигляд наведено нижче:

$$\square\square\square = \frac{\Sigma_{\square,\square}^{\square} |\square_{\square\square} - \square\grave{}_{\square\square}|}{|\square|}, \quad (2.1)$$

де $\square_{\square\square}$ – це фактична оцінка користувача $\square$ продукту $\square$;

$\square\grave{}_{\square\square}$ – прогнозована оцінка, надана рекомендаційним алгоритмом;

$\square(\square)$ список поведінки користувача в тестовому наборі.

Крім того, середня квадратична помилка (MSE), корінь від середньої квадратичної помилки (RMSE) і нормалізована середня абсолютна помилка (NMAE) [16] є всіма показниками, подібними до MAE.

$$\square\square\square = \frac{\sum_{\square,\square} (\square_{\square\square} - \square_{\square\square}^{`})^2}{|\square|}, \quad (2.2)$$

$$\square\square\square\square = \sqrt{\frac{\sum_{\square,\square} (\square_{\square\square} - \square_{\square\square}^{`})^2}{|\square|}}, \quad (2.3)$$

$$\square\square\square\square = \frac{\square\square\square}{\square_{\square\square\square} - \square_{\square\square\square}}, \quad (2.4)$$

де $\square_{\square\square\square}$ і $\square_{\square\square\square}$ є максимальним і мінімальним значенням інтервалу оцінки користувачів.

MSE і RMSE посилюють штрафи: чим більша похибка в прогнозі, тим суворіше покарання. Усі діапазони MAE, MSE та RMSE є [0, +∞). NMAE нормалізує MAE всередині інтервалу оцінювання, щоб можна було порівняти ефективність одного і того ж алгоритму рекомендацій на різних наборах даних.

2.3.2. Релевантність рекомендацій

Для визначення релевантності рекомендацій найчастіше використовуються □□□□□□□□□□, □□□□□□ та □*1*.

Білсус Д. та ін. вперше ввели поняття □□□□□□□□□□ і □□□□□□ в систему рекомендацій і оцінили їх. □□□□□□□□□□ обчислює співвідношення продуктів, які подобаються користувачеві серед рекомендованих, до всіх в рекомендованому списку. Показано точність рекомендацій користувача □ у формулі (2.5):

$$\square(\square_{\square}) = \frac{\square_{\square} \cap \square_{\square}}{\square_{\square}}, \quad (2.5)$$

де □□ – це список рекомендацій користувача □;

□□ – це реальний набір даних про користувача □.

□□□□□□ розраховує співвідношення продуктів, які подобаються користувачам у рекомендованому списку, до всіх продуктів, які подобаються користувачам у системі. Рекомендований □□□□□□ користувача □ показано у формулі (2.6):

$$□(□_□) = \frac{□_□ ∩ □_□}{□_□} \quad (2.6)$$

□□□□□□□□□□ та □□□□□□ важко покращити одночасно. Обидва вони часто негативно корелюють і залежать від довжини рекомендованого списку. Тому для обчислення зазвичай використовується □*1* – середнє гармонійне з двох. Чим більше значення □*1*, тим кращий ефект прогнозу. Розрахунок показано у формулі (2.7):

$$□1 = \frac{2*□□}{□+□}, \quad (2.7)$$

де □ – □□□□□□□□□, а □ – □□□□□□.

2.3.3. Точність сортування рекомендацій

Показники точності рейтингування вимірюють ступінь однаковості між впорядкованим списком рекомендацій і рейтингом товарів, які виставив користувач. Ці метрики підходять для оцінки рекомендаційних систем, які видають користувачам рейтинговий список.

Чжоу Тао та ін. [18] запропонував показник середнього рейтингу для обчислення точності рекомендаційної системи. Ранг продукту □, виставлений користувачем □ показано у формулі (2.8):

$$\square\square\square = \frac{\square_{\square}}{\square}, \tag{2.8}$$

де □ – кількість продуктів, не обраних користувачем у навчальному наборі;

$\square_{\square}$ – положення продукту □, яке слід передбачити в тестовому наборі рекомендованого списку.

Чим менша оцінка рейтингу, тим більше система прагне ранжувати продукти, які подобаються користувачам, на початку.

Усереднена середня точність (MAP) [19] означає отримані результати пошуку, як показано у формулі (2.9).

$$\square\square\square = \frac{\sum_{\square}^{\square} \square\square_{\square}}{|\square|}, \tag{2.9}$$

$$\square\square_{\square} = \frac{1}{\square_{\square}} \sum_{\square}^{\square_{\square}} \frac{\sum_{\square}^{\square_{\square}} \square(\square_{\square\square} < \square_{\square\square}) + 1}{\square_{\square\square}}, \tag{2.10}$$

де $\square_{\square}$ представляє результат пошуку;

$\square_{\square\square}$ відображає позицію продукту □ у списку рекомендацій;

$\square_{\square\square} > \square_{\square\square}$ означає, що продукт j ранжується перед продуктом □ у список рекомендацій користувача □;

□ встановлено користувачем.

2.3.4 Інші метрики оцінювання якості рекомендацій

Щоб задовольнити широкі інтереси користувачів, список рекомендацій повинен охоплювати користувачів різні сфери інтересів, тобто рекомендація має бути якомога різноманітнішою. Окрім того, рекомендації мають відкривати користувачу якісь нові продукти або у даному випадку фільми.

Індивідуальне різноманіття вимірює скільки різних сфер охоплює рекомендаційна пібріка. Якщо припустити, що $□(□,□) ∈ [0,1]$ визначає подібність між продуктами □ та j, тоді різноманітність списку рекомендацій $□_□$ користувача □ показано у формулі (2.11).

$$□□□□□□□□□(□_□) = 1 - \frac{\sum_{□,□∈□_□,□≠□} □(□,□)}{0.5 * |□_□| * (|□_□| - 1)}. \quad (2.11)$$

Рекомендація новинок – це рекомендація продуктів, про які користувачі раніше не чули. Найпростіший спосіб оцінити дану метрику – розрахувати середню популярність продуктів у рекомендованому списку за формулю (2.12).

$$□□□□□□□(□_□) = \frac{\sum_{□∈□_□} □(□)}{|□_□|}, \quad (2.12)$$

де □(□) - це популярність продукту □;

$□_□$ – список рекомендацій користувачів □.

2.4 Опис роботи системи

Нижче зображено VAD діаграму, яка демонструє реакцію системи на дії користувача. Для авторизації користувача система перевіряє чи такий ID в

БД, потім користувача переадресовує на головну сторінку, а система в цей час робить запит до БД та виводить топ-10 найпопулярніших та найвище оцінених фільмів. Коли користувач переходить на вкладку з персональними рекомендаціями, то система запускає модель, яка формує топ-10 рекомендацій основаних, на вмісті та топ-10 на основі колаборативної фільтрації. Для кожного з підходів описано короткий алгоритм (дії системи), який повертає персоналізовані рекомендації. Якщо користувач хоче подивитись фільм, то система переадресовує його на сайт IMDB. Передбачається, що після перегляду фільму користувач повернеться в систему та оцінить фільм, який щойно переглянув, тоді система запише рейтинг в базу даних.

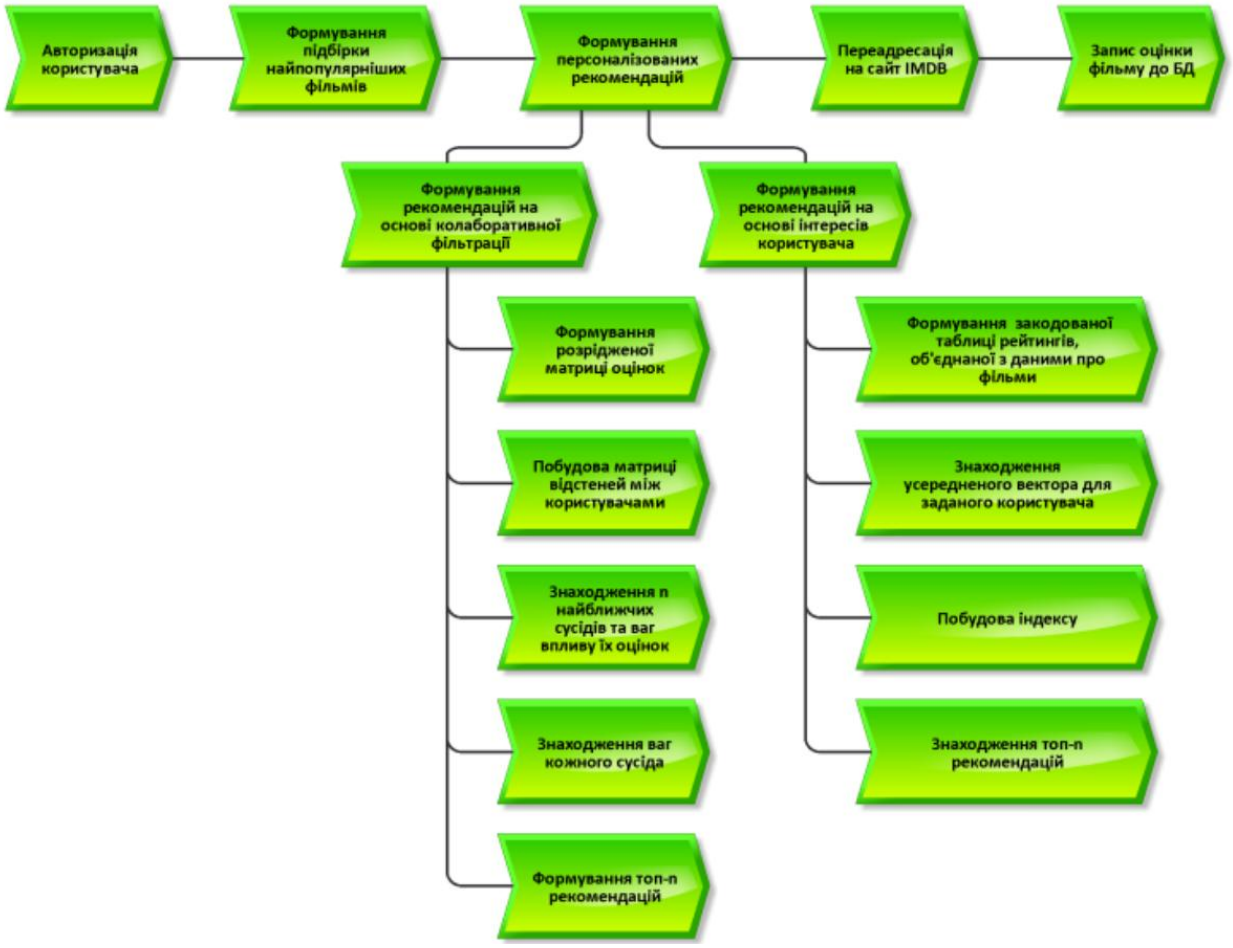

Рисунок 2.7 VAD діаграма роботи системи

2.5 Практична цінність

Практична цінність даної роботи полягає в тому, що користувач може зберігати список переглянутих фільмів в одному місці. Окрім того, дана розробка передбачає формування списку бажань - список, куди можна додавати фільми для подальшого перегляду. Але, найголовніше, що ця система персоналізовано підбирає рекомендації фільмів на основі його інтересів. Вона може пропонувати фільми як схожі на ті, що вже були переглянуті, так і інших жанрів, які потенційно можуть сподобатись користувачу. Така система допомагає користувачу організовувати свою історію фільмів та спрощує вибір фільму перед переглядом.

2.6 Висновки

Отже, у даному розділі детально розглянуто всі інструменти, які були використані в процесі реалізції рекомендаційної системи: мова Python та його бібліотеки, які найкраще підходять для задач машинного навчання; мікрофреймворк Flask для розробки інтерфейсу; PostgreSQL як СУБД для підтримки розробленої системи.

Окрім того, було описано метод для знаходження приблизних найближчих сусідів та вказано його переваги над виснажливим пошуком.

Також, у цьому розділі систематично представлені показники оцінки рекомендаційної системи та розглянуто його з багатьох точок зору: точність прогнозованої оцінки, релевантність рекомендації, якість сортування,

новизна та інші. На практиці часто вибирають різні показники оцінки відповідно до завдань та вимог до рекомендаційної системи. Але в будь-якому випадку кінцевою метою є підвищення рівня задоволеності користувачів, адже бізнес-вигоди збільшуються зі збільшенням задоволеності користувачів.

РОЗДІЛ 3. ФУНКЦІОНАЛЬНИЙ АНАЛІЗ РОЗРОБЛЕНОЇ СИСТЕМИ

3.1 Опис даних

Датасет movielens100k - набір даних, який сформувала рекомендаційна система MovieLens та надала для відкритого доступу всім бажаючим. Дані зберігаються в 3 csv файлах: таблиця користувачів, фільмів та оцінок.

Таблиця користувачів (рис. 3.1) містить такі поля:

- user_id: унікальне ID, яке інкрементно присвоюється кожному новому користувачу;
- age: вік користувача;
- gender: стать;
- occupation: професія;
- zip_code: код міста, де проживає користувач.

	user_id bigint	age bigint	gender text	occupation text	zip_code text
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344
8	8	36	M	administrator	05201
9	9	29	M	student	01002

Рисунок 3.1 Таблиця користувачів

Таблиця фільмів (рис. 3.2) складається з ID фільму, назви, дати випуску, посилання на сайт IMDB для перегляду та 19 колонок типу boolean з назвами жанрів - якщо фільм відноситься до певного жанру то в цих колонках значення 1, інакше - 0.

	item_id bigint	title text	release_date text	imdb_url text	unknown boolean	Action boolean	Adventure boolean	Animation boolean	Children's boolean	Comedy boolean	Crime boolean	Documentary boolean
1	1	Toy Sto...	01-Jan-1995	http://us.imd...	false	false	false	true	true	true	false	false
2	2	Golden...	01-Jan-1995	http://us.imd...	false	true	true	false	false	false	false	false
3	3	Four R...	01-Jan-1995	http://us.imd...	false	false	false	false	false	false	false	false
4	4	Get Sh...	01-Jan-1995	http://us.imd...	false	true	false	false	false	true	false	false
5	5	Copyca...	01-Jan-1995	http://us.imd...	false	false	false	false	false	false	true	false
6	6	Shang...	01-Jan-1995	http://us.imd...	false	false	false	false	false	false	false	false
7	7	Twelve...	01-Jan-1995	http://us.imd...	false	false	false	false	false	false	false	false
8	8	Babe (...	01-Jan-1995	http://us.imd...	false	false	false	false	true	true	false	false
9	9	Dead ...	01-Jan-1995	http://us.imd...	false	false	false	false	false	false	false	false

Рисунок 3.2 Таблиця фільмів

Найважливішою є таблиця оцінок (рис. 3.3), в якій покрово записуються дії: який користувач (user_id) якому фільму (item_id) яку оцінку (rating) поставив та коли (timestamp).

	user_id bigint	item_id bigint	rating bigint	timestamp bigint
1	259	255	4	874724710
2	259	286	4	874724727
3	259	298	4	874724754
4	259	185	4	874724781
5	259	173	4	874724843
6	259	108	4	874724882
7	259	772	4	874724882
8	259	288	3	874724905
9	259	928	4	874724937
10	259	117	4	874724988

Рисунок 3.3 Таблиця оцінок

Однією з можливостей системи є змога додати фільм у список бажань. Для цього створено та додано до БД ще одну таблицю (рис. 3.4), яка по структурі схожа на таблицю оцінок. Колонками в таблиці є:

- ID користувача;
- ID фільму, який він хоче подивитись;
- час, коли фільм було додано в список юажань;
- колонка типу boolean яка показує звідки було додано фільм - якщо 0, то користувач в пошуку знайшов фільм і додав його в список бажань, а якщо 1, то фільм додано з рекомендованих системою.

	user_id bigint	item_id bigint	datetime date	from_recommendations boolean
1	325	100	2022-05-08	true
2	764	100	2022-05-08	true
3	64	100	2022-05-08	true
4	653	100	2022-05-08	true
5	432	100	2022-05-08	true
6	56	100	2022-05-08	true
7	56	163	2022-05-08	true
8	653	181	2022-05-08	true
9	653	12	2022-05-08	true
10	654	1	2022-05-08	true

Рисунок 3.4 Таблиця списку бажань користувачів

Дані таблиці мають відношення 1 до багатьох: 1 користувач ставить оцінки багатьом фільмам та додає багато фільмів в список бажань. Аналогічно з фільмами: 1 фільм може бути оцінений багатьма користувачами та доданий в список бажань не одного користувача.

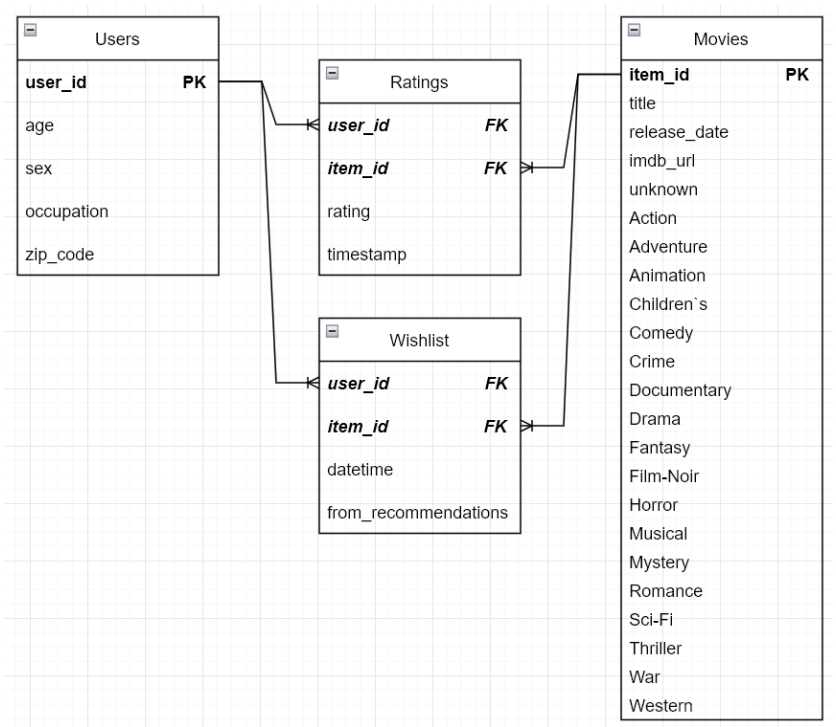

Рисунок 3.5 Даталогічна модель бази даних

Для кращого розуміння даних, які є в наборі візуалізуємо їх. Як бачимо на рис. 3.6 основна частка користувачів віком від 20 до 50 років. У вибірці представлено 71% чоловіків та 29% жінок. Найчастіша професія - студент, на другому місці - представники навчальної сфери, і на третьому - адміністратори. Більшість користувачів є жителями США та переважно східних штатів.

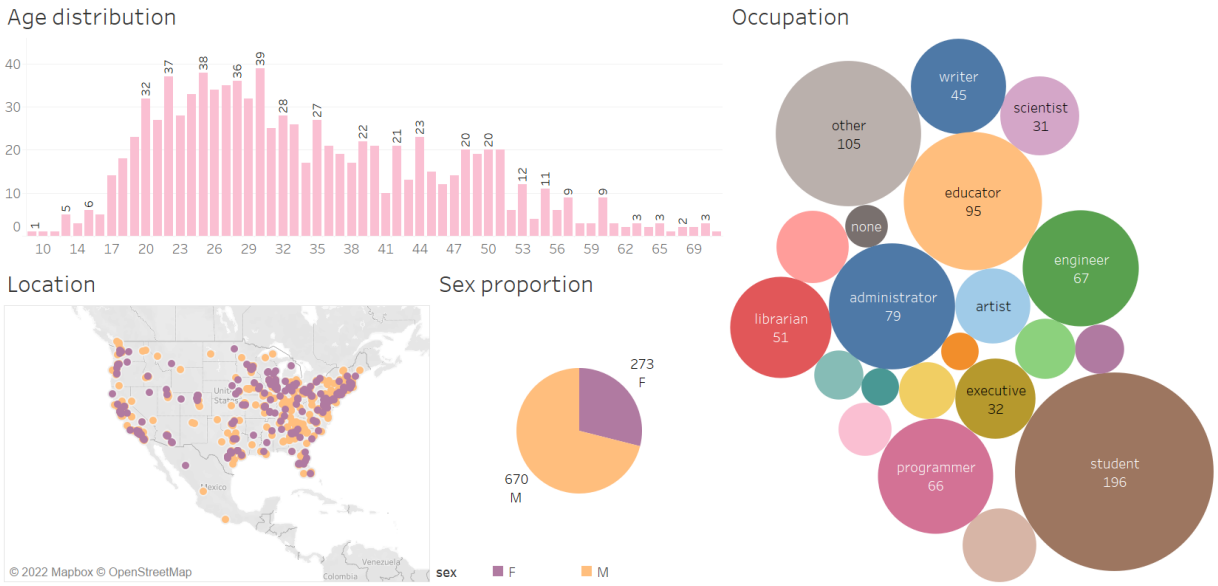

Рисунок 3.6 Опис користувачів

Також, на рис. 3.7 візуалізовано розподіл оцінок: найпопулярнішою оцінкою є 4, за нею 3 і 5.

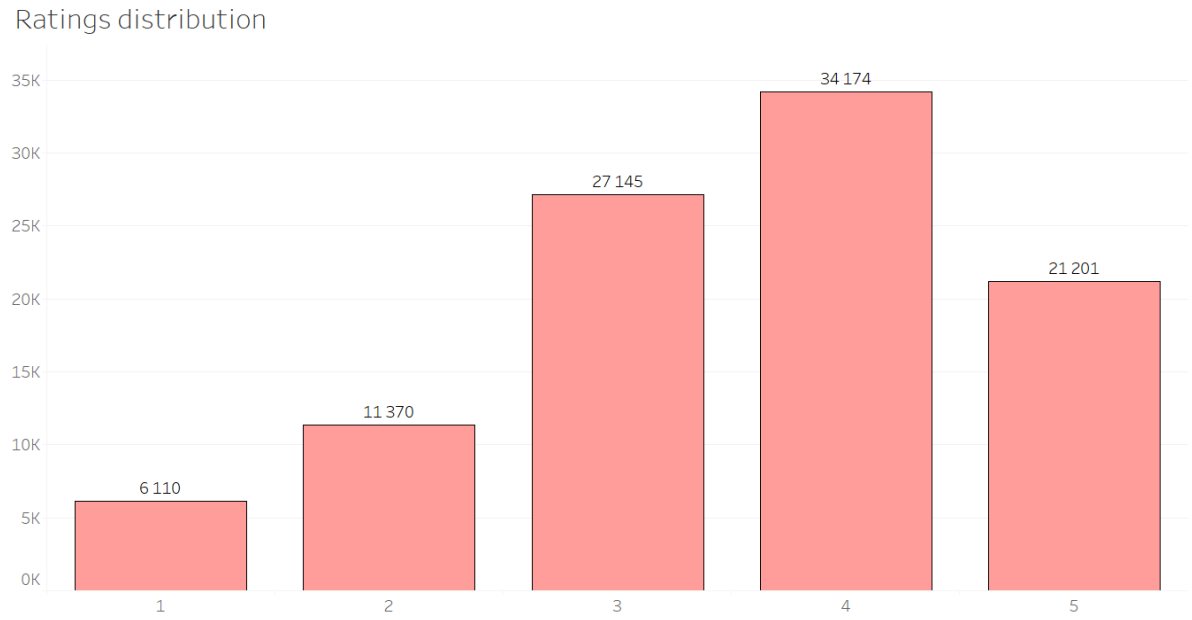

Рисунок 3.7 Розподіл оцінок в датасеті

3.2 Опис інтерфейсу

3.2.1 Опис основних сторінок

Сторінка, на яку потрапляють користувачі вперше - це головна сторінка (Рисунок 3.8а), на якій є топ-10 найбільш популярних фільмів та топ-10 фільмів з найвищою оцінкою. Користувач може перейти за посиланням на сайт IMDB, що подивитись фільм, який йому сподобався. Також, він може зробити пошук по фільмах, які є в системі, тоді його переадресує на сторінку з результатами пошуку (Рисунок 3.8 б).

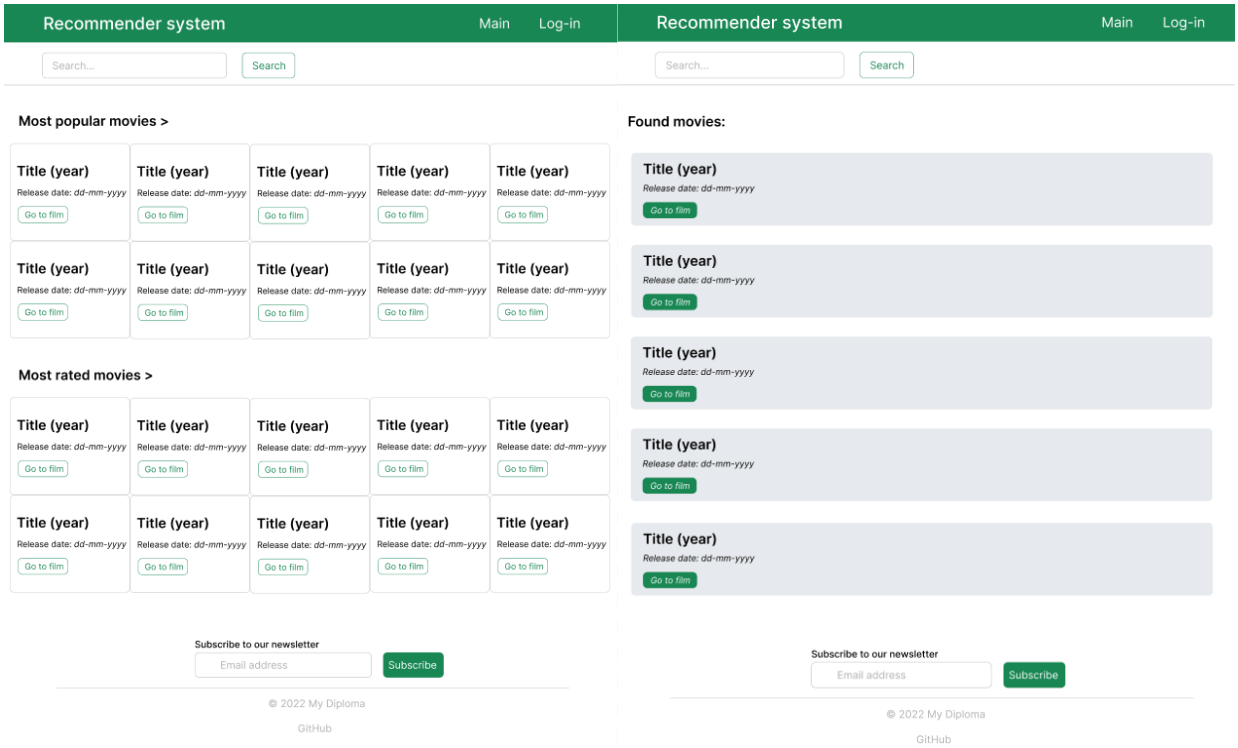

Рисунок 3.8 а) головна сторінка для незалогіненого користувача; б) сторінка з результатами пошуку фільмів

Далі користувач може увійтив систему, для того аби мати можливість користуватись ширшим функціоналом сайту. Форма для входу (Рисунок 3.9) проста: користувач вводить свій ID, після цього система перевіряє чи є такий

унікальний код в базі даних та авторизовує користувача, якщо такий є, і передадресовує на сторінку реєстрації (Рисунок 3.10), якщо - немає.

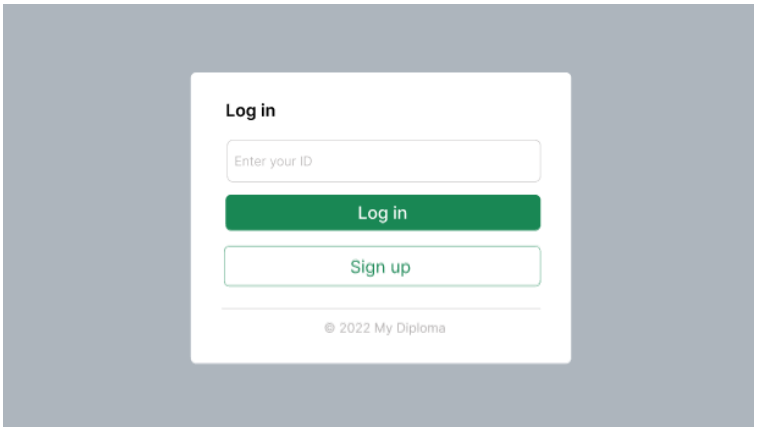

Рисунок 3.9 Сторінка для входу користувача в систему

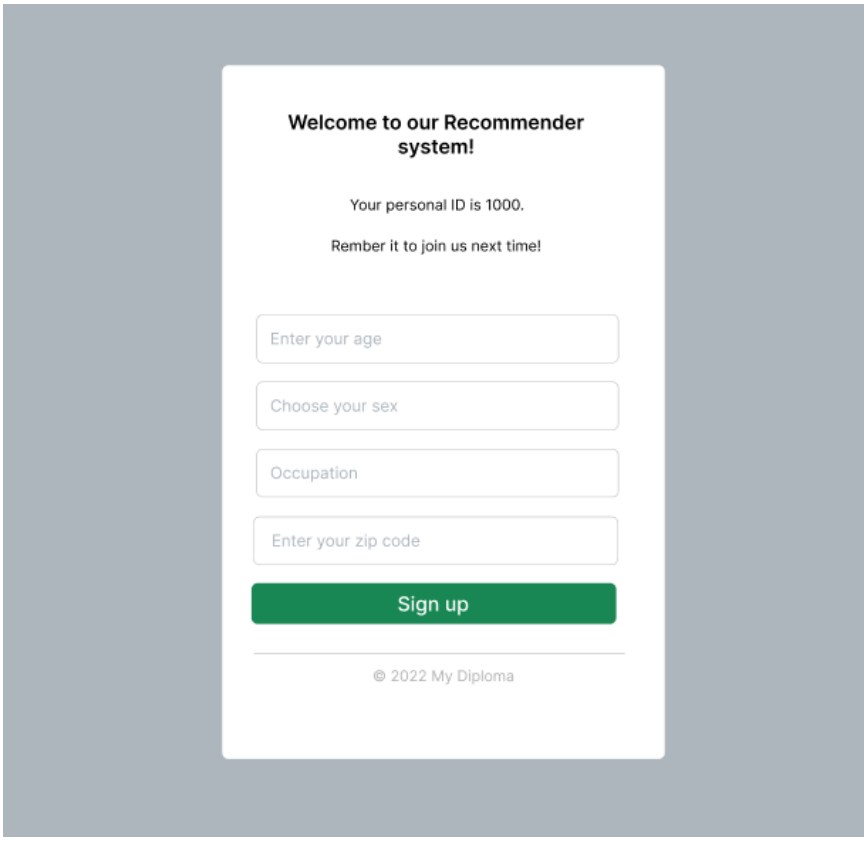

Рисунок 3.10 Сторінка з формою для реєстрації нового користувача

Як видно на рис. 3.10 для нового користувача автоматично присвоюється ID та є форма для заповнення персональних даних.

Для авторизованого користувача головна сторінка має повніший функційонал: фільм можна додати до списку бажань або оцінити, якщо користувач його вже бачив. Аналогічний вигляд має сторінка з персональними рекомендаціями (рис. 3.11) за винятком підбірки фільмів.

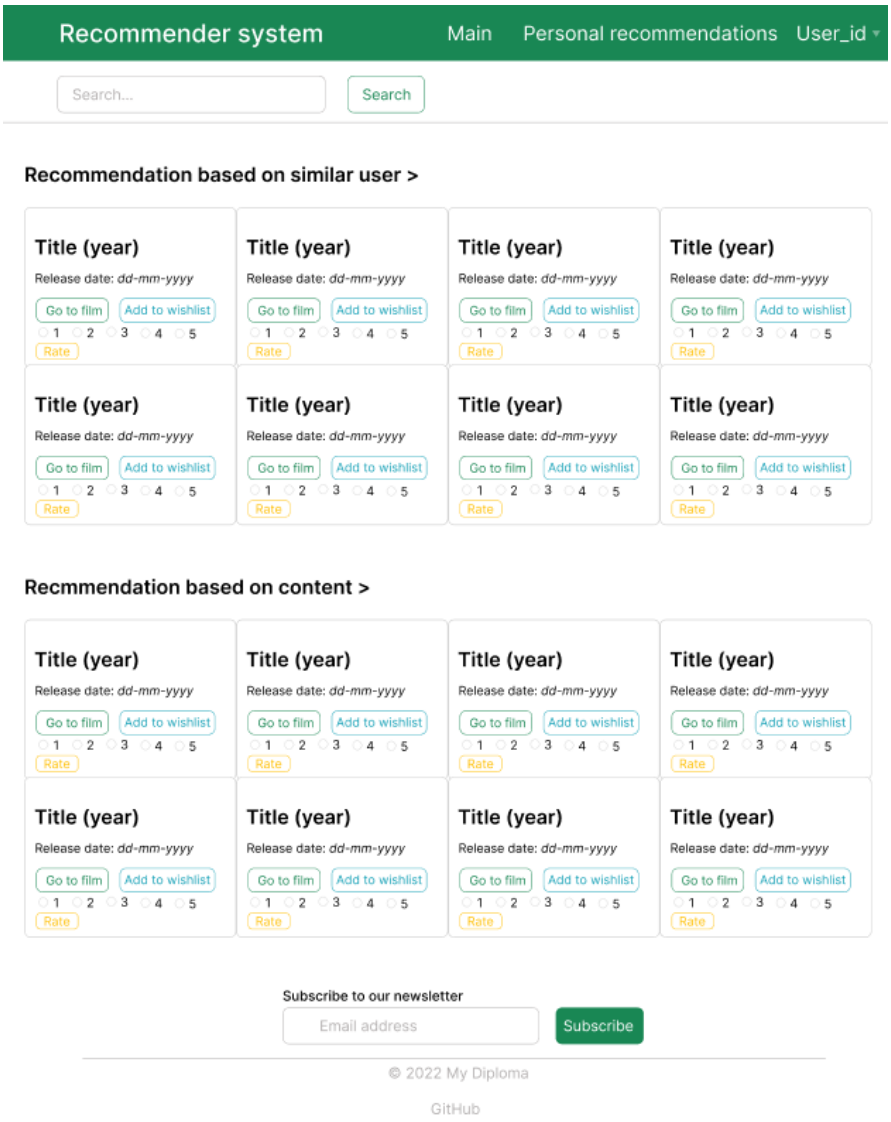

Рисунок 3.11 Сторінка з персональними рекомендаціями

Ще однією сторінкою, яку можна віднести до основних є сторінка зі списком бажань користувача (рис. 3.12). Якщо фільм, який вже є в цьому спику став для користувача неактуальним чи нецікавим, то він може видалити його зі списку бажань. Окрім того, є можливість оцінити фільм, якщо користувач його вже подивився. У такому випадку оцінка запишеться в базу даних, а фільм автоматично видалиться зі списку бажань та переміститься у список переглянутих.

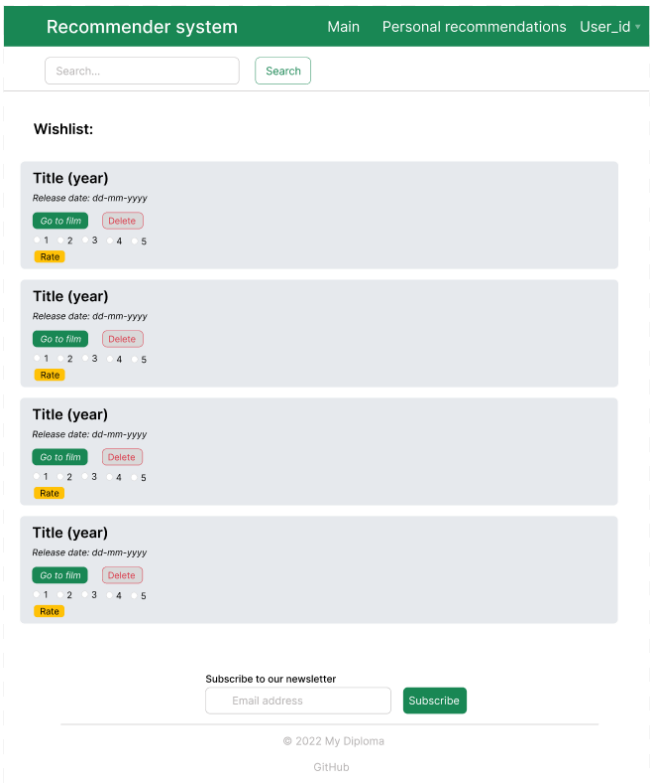

Рисунок 3.12 Сторінка зі списком бажань користувача

3.2.2 Опис переходів між сторінками

На зображенні нижче показано моєливі переходи між сторінками:

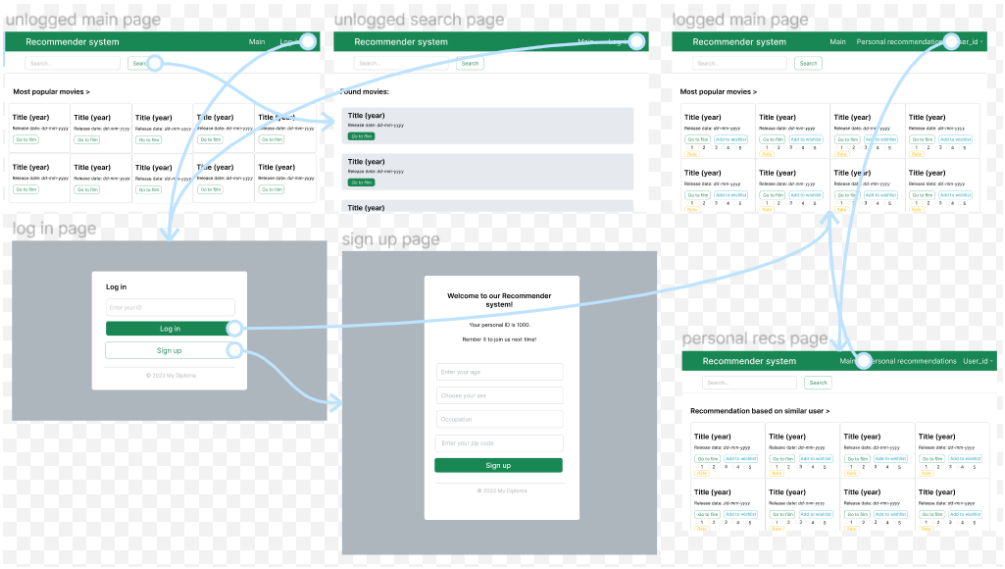

Рисунок 3.13 Макет переходів між сторінками в Figma

Як бачимо, з головної сторінки, куди заходять всі користувачі можна перейти або на сторінку з результатами пошуку фільмів або на сторінку входу в систему. Якщо користувач вже зареєстрований в системі, то він може ввести свій ID та перейти на головну сторінку для авторизованих користувачів, яка має ширший функціонал. Якщо користувач новий, то він може зареєструватись: йому автоматично присвоюється ID та висвічується форма для заповнення, після чого він потрапляє теж на головну сторінку. Далі користувач може перейти на:

- Сторінку з персональними рекомендаціями
- Сторінку зі списком бажань
- Сторінку з переглянутими та оціненими фільмами
- Сторінку з результатами пошуку по фільмам

Схему переходів можна побачити на діаграмі нижче:

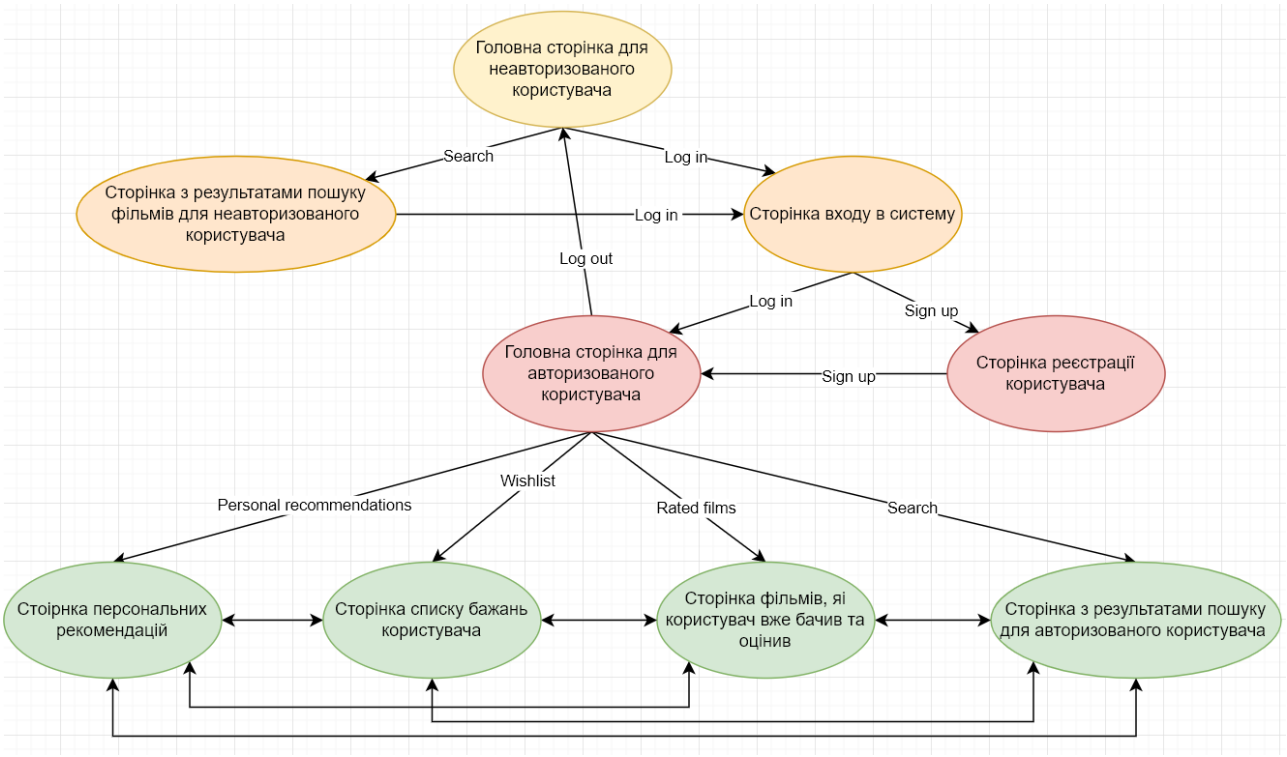

Рисунок 3.14 Схема з’єднань сторінок: які кнопки на яку сторінку ведуть (у вузлах знаходяться сторінки, стрілки - кнопки, які ведуть на ту сторінку)

3.2.3 Опис тест-кейсів

Однією з нефункційональних вимог до системи є її коректність на будь-які дії користувача. Якщо користувач введе в пошуковий рядок набір симовів, які не відповідають назві жодного фільму в системі, то його переадресує на сторінку, як на рис.

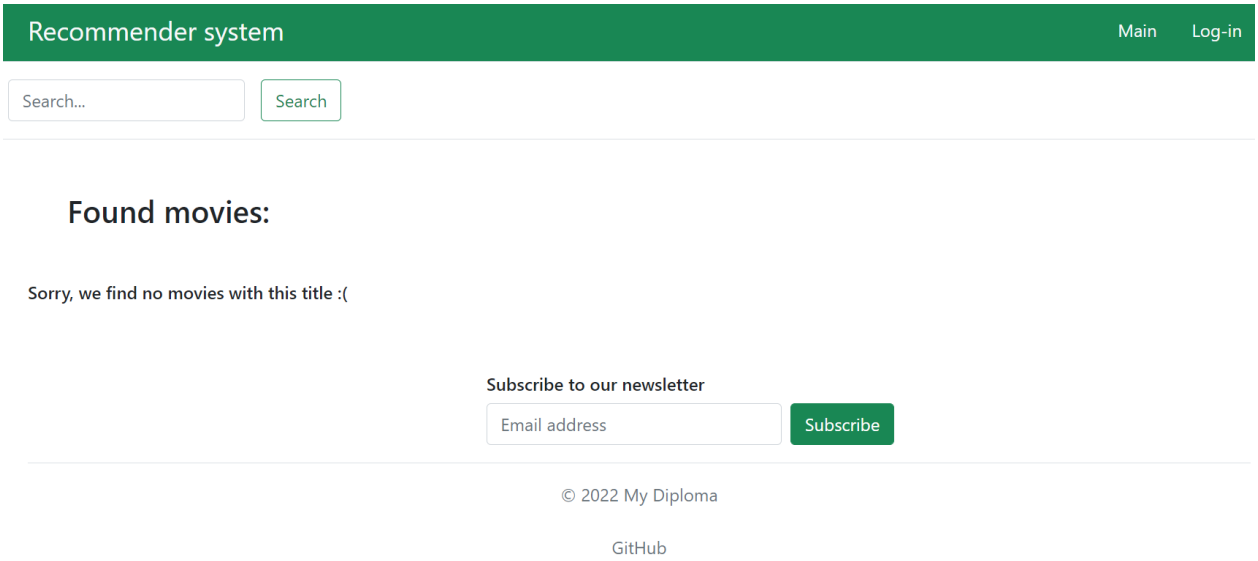

Рисунок 3.15 Сторінка з результатами пошуку, якщо в системі немає відповідних фільмів

Якщо користувач ще немає фільмів у списку бажань, але переходить на сторінку “wishlist”, то він отримає повідомлення як на рис. 3.16.

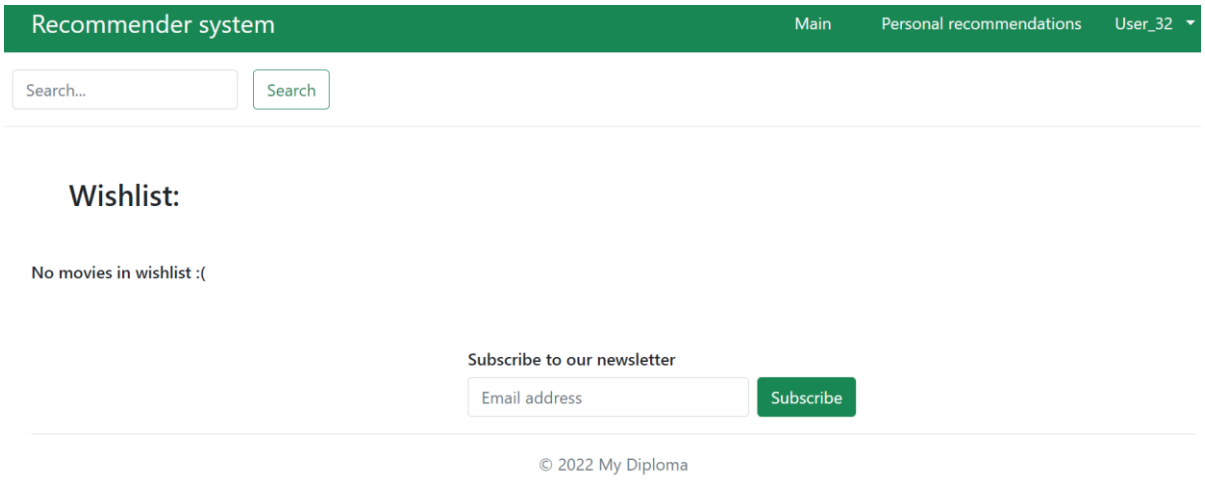

Рисунок 3.16 Повідомлення користувачу, що в нього немає фільмів у списку бажань

Якщо користувач при вході вводить в поле ID неіснуюче значення, то система переадресовує його на сторінку реєстрації (рис. 3.17).

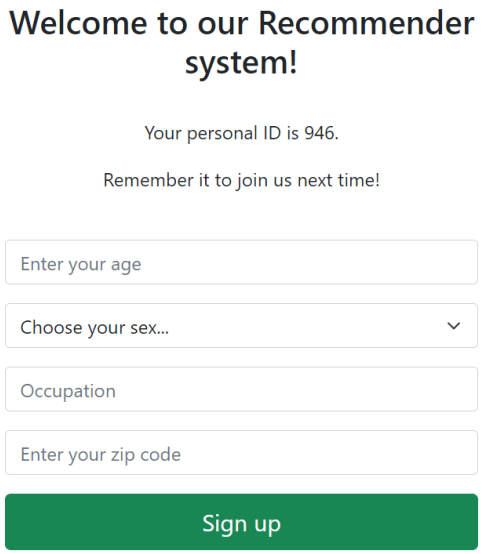

Рисунок 3.17 Сторінка реєстрації нового користувача

Коли новий користувач переходить на сторінку з персональними рекомендаціями, то отримує повідомлення про те, що в системи немає жодної інформації про його вподобання і пропонує оцінити ті фільми, які він вже бачив для того, аби сформувати персоналізований топ рекомендацій (рис.3.18).

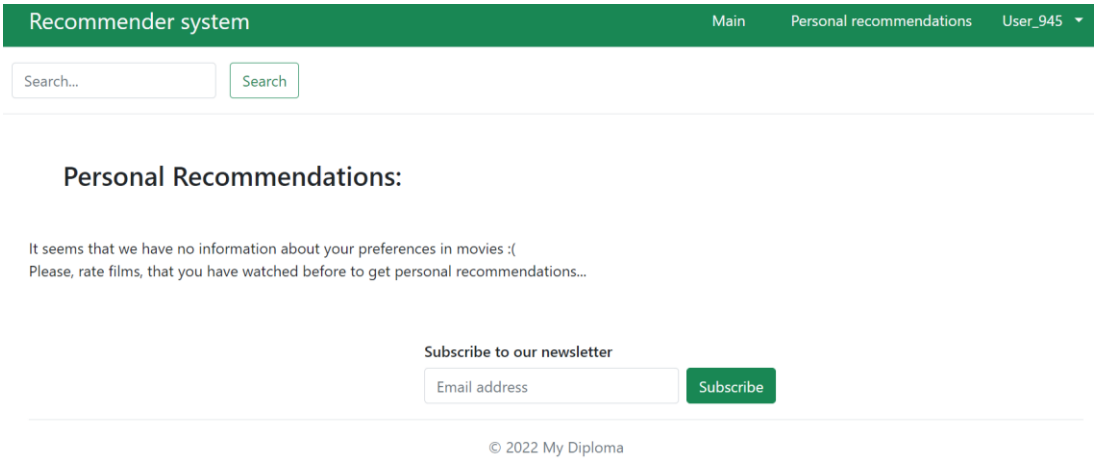

Рисунок 3.18 Сторінка персональних рекомендацій для нового користувача

При спробі нового користувача перейти на сторінку “rated films” висвічується повідомлення (рис. 3.19), про те, що він ще не оцінив жодного фільма.

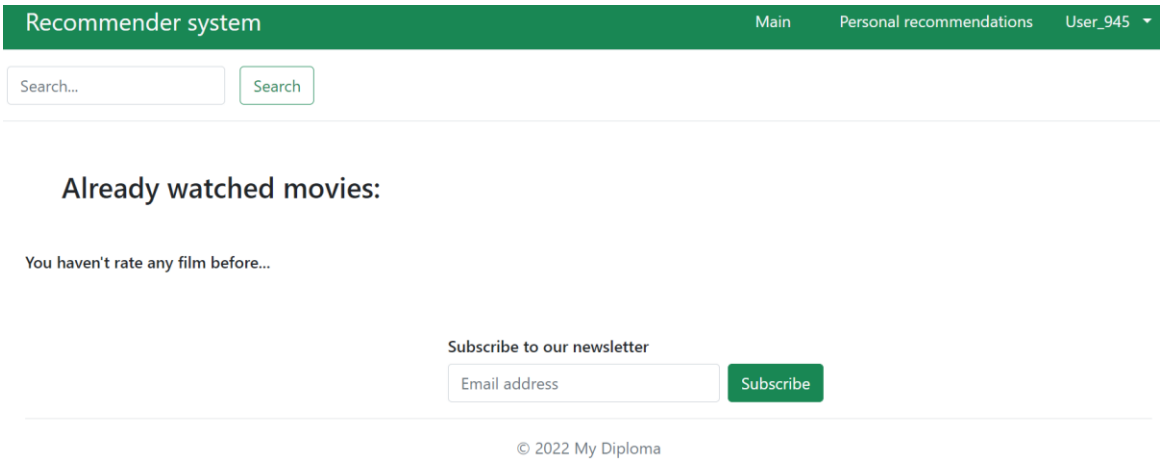

Рисунок 3.19 Повідомлення про те, що кристувач ще не оцінив жодного фільму

3.3 Висновки

Отже, у даному розділі було детально розглянуто розробку інтерфейсу: які сторінки було розроблено, який функціонал доступний на кожній з сторінок. Також наведено схеми переходів між сторінками та якими кнопками вони пов’язані між собою. Більш того, проведено тестування системи на коректність різних дій користувача.

Також було описано дані, які використовуються для тренування рекомендаційних моделей та представлення на сайті. Проведено аналіз користувачів: їх вікової категорії, розміщення, сфери професійної діяльності та статі.

ВИСНОВКИ

У ході виконання дипломної роботи було розроблено рекомендаційну систему методом колаборативної фільтрації та на основі змісту. Обрано було 2 найпопулярніших підходи для того аби порівняти, які результати вони дають та максимально урізноманітнити рекомендовані фільми. Перший спосіб підбирає фільми, які можуть сподобатись користувачеві на основі схожих до нього користувачів, а інший знаходить фільми, які найбільш схожі на ті, що йому вже сподобались.

Також у роботі було розглянуто та детально описано методи оцінювання рекомендацій, адже важливо, щоб користувач був задоволеним підбіркою. Основними метриками оцінювання є точність прогнозування оцінки. Ця метрика є важливою, адже саме на основі прогнозів оцінок формується кінцева підбірка фільмів. Ще однією метрикою, яка є не менш важливою - це релевантність наданих рекомендацій. Система може точно прогнозувати оцінку, але чи сподобаються фільми, які потрапили в підбірку користувачеві? Тому необхідно оцінювати скільки фільмів користувачу сподобалось з підбірки, проте ця оцінка можлива лише, коли система є діючою та можна відслідкувати вподобання користувача.

Для того, аби презентувати персоналізовані рекомендації користувачу, було розроблено сайт, на якому є сторінка з топ-10 найпопулярніших фільмів та топ-10 фільмів з найвищою оцінкою. Є сторінка з підбіркою рекомендацій на основі колаборативної фільтрації та на основі вмісту. Також користувач має змогу переглянути вже оцінені фільми, та фільми зі списку бажань. Користувач, який є новий для системи не отримає персональних рекомендацій, поки не проставить оцінки на фільми, які вже бачив.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. https://medium.com/recombee-blog/machine-learning-for-recommender-systems-part-1-algorithms-evaluation-and-cold-start-6f696683d0ed
2. https://developer.nvidia.com/blog/how-to-build-a-winning-recommendation-system-part-1/
3. https://towardsdatascience.com/recommendation-systems-explained-a42fc60591ed#:~:text=Recommendation%20engines%20are%20a%20subclass,returned%20back%20to%20the%20user.
4. Deep learning for recommender systems: A Netflix case study – Harald Steck, Linas Baltrunas, Ehtsham Elahi, Dawen Liang, Yves Raimond, Justin Basilico.
5. https://towardsdatascience.com/recommendation-system-matrix-factorization-d61978660b4b
6. Challenges in Search on Streaming Services: Netflix Case Study, https://arxiv.org/abs/1903.04638
7. Learning a Personalized Homepage, https://netflixtechblog.com/learning-a-personalized-homepage-aa8ec670359a
8. The Netflix Recommender System: Algorithms, Business Value and Innovation – CARLOS A. GOMEZ-URIBE, NEIL HUNT, https://dl.acm.org/doi/pdf/10.1145/2843948
9. Trends, problems and solutions of recommender system, https://www.researchgate.net/publication/307862659_Trends_problems_and_solutions_of_recommender_system
10. Reducing Data Sparsity in Recommender Systems, https://www.iasj.net/iasj/download/ca46a3f16fea9999
11. https://prjctrmag.com/ai-in-spotify
12. Recommender Systems Handbook / R. Francesco, R. Lior, S. Bracha, K. B. Paul. – Dordrecht: Springer, 2015. – 1009 p.
13. Matrix and Tensor Factorization Techniques for Recommender Systems / Panagiotis Symeonidis, Andreas Zioupos – Dordrecht: Springer
14. https://www.academia.edu/40417916/Software_Requirements_Specification_For_Netflix_Movies_and_TV_Streaming_School_of_Computer_Science_and_Engineering_LOVELY_PROFESSIONAL_UNIVERSITY

15. Evaluation Metrics for Personalized Recommendation Systems. Shuhao Jiang and Jinlin Song 2021 J. Phys.: Conf. Ser. 1920 012109

16. Balabanović, M., Shoham, Y. (1997) Fab: content-based, collaborative recommendation. Communications of the ACM, 40(3): 66-72.

17. Billsus, D., Pazzani, M. J. (1998). Learning collaborative information filters. In: Icml. Madison. pp. 46-54.

18. Zhou, T., Jiang, L. L., Su, R. Q., Zhang, Y. C. (2008). Effect of initial configuration on network-based recommendation. EPL (Europhysics Letters), 81(5): 58004.

19. Yu, D., Cheng, T., Yuan, X. (2020) Software Crowdsourcing Task Recommendation Algorithm. Based on Learning to Rank. Computer Science, 47(12): 106-113.

20. Liu, J., Zhou, T., Guo, Q., Wang, B. (2009) Overview of the Evaluated Algorithms for the Personal Recommendation Systems. Complex Systems and Complexity Science, 6(03): 1-10.

21. Yao, Y. Y. (1995) Measuring retrieval effectiveness based on user preference of documents. Journal of the American Society for Information science, 46(2): 133-145.

22. https://machinelearninginterview.com/topics/machine-learning/ndcg-evaluation-metric-for-recommender-systems/

23. https://towardsdatascience.com/comprehensive-guide-to-approximate-nearest-neighbors-algorithms-8b94f057d6b6

24. https://github.com/spotify/annoy

25. https://erikbern.com/2015/10/01/nearest-neighbors-and-vector-models-part-2-how-to-search-in-high-dimensional-spaces.html

26. https://www.geeksforgeeks.org/python-for-data-science/

27. https://www.tableau.com/

28. https://pandas.pydata.org/

29. https://numpy.org/

30. https://scikit-learn.org/stable/

ДОДАТКИ

Додаток А

Лістинг програми, де реалізовано колаборативну фільтрацію.

```
    class CollaborativeFilteringModel(BaseModel):
        def __init__(self):
            self.model_name = 'collaborative_filtering'
            self.rating_matrix = None
            self.mean_users_rating = None
            self.preference_matrix = None
            self.cosine_matrix = None
            self.users_distances = None

        def fit(self, train_df):
            self.rating_matrix =
pd.pivot_table(train_df, values='rating',
index='user_id', columns=['item_id'])
            self.mean_users_rating =
self.rating_matrix.mean(axis=1)
            self.preference_matrix =
self.rating_matrix.subtract(self.mean_users_rating,
axis=0).fillna(0)
            self.cosine_matrix =
cosine_similarity(self.preference_matrix)
            self.users_distances =
pd.DataFrame(self.cosine_matrix,

index=self.rating_matrix.index,

columns=self.rating_matrix.index)

        def predict_one(self, user, film, neighbours=10,
threshold=0.15):
```

```
            if user in self.rating_matrix.index and film
in self.rating_matrix.columns:
                users_who_saw_film =
self.rating_matrix.loc[:, film].notnull()
                user_distances =
self.users_distances.loc[users_who_saw_film].loc[:,
user]
                filtered_distances =
user_distances[user_distances >
threshold].sort_values(ascending=False).head(
                    neighbours)
                users_weights = filtered_distances /
sum(filtered_distances)
                users_preferences =
self.preference_matrix.loc[users_weights.index, film]
                user_delta =
users_weights.dot(users_preferences)
                rating_prediction =
self.mean_users_rating.loc[user] + user_delta
                # print('all info known')
            elif film in self.rating_matrix.columns:
                rating_prediction =
self.rating_matrix.loc[:, film].mean()
                # print('user unknown')
            else:
                rating_prediction =
self.mean_users_rating.loc[user]
                # print('film unknown')

            return rating_prediction

        def predict(self, test_df):
            prediction_list = []
            for user, film in zip(test_df['user_id'],
test_df['item_id']):
```

```
prediction_list.append(self.predict_one(user, film))

            test_df_copy = deepcopy(test_df)
            test_df_copy['predicted_rating'] =
prediction_list
            return test_df_copy

        def top_n(self, user, n=10):
            watched_movies_id =
self.rating_matrix.loc[user,
self.rating_matrix.loc[user, :].notnull()].index
            df = pd.DataFrame({'user_id': [user] *
(len(self.rating_matrix.columns)-
len(watched_movies_id)),
                               'item_id': [movie for
movie in self.rating_matrix.columns if
                                           movie not in
watched_movies_id]})
            prediction = self.predict(df) \
                .sort_values('predicted_rating',
ascending=False) \
                .drop(['user_id'], axis=1) \
                .head(n)
            return prediction
```

Додаток Б

Лістинг програми, де реалізовано підхід на основі контенту.

```
    class ContentBasedModel(BaseModel):
        def __init__(self):
            self.model_name = 'content_based'
            self.rating_df = None
            self.encoded_movies = None
            self.index = None

        def fit(self, movies_df, rating_df):
            self.rating_df = rating_df
            encoded_movies_df =
pd.concat([movies_df.iloc[:, 0],
pd.to_datetime(movies_df.iloc[:, 2], format='%d-%b-
%Y'),

movies_df.iloc[:, 4:].astype(int)], axis=1)
            encoded_movies_df['release_year'] =
pd.DatetimeIndex(encoded_movies_df['release_date']).yea
r.to_list()
            encoded_movies_df.drop(['release_date'],
axis=1, inplace=True)

            scaler = MinMaxScaler()
            encoded_movies_df['release_year'] =
scaler.fit_transform(

np.array(encoded_movies_df['release_year']).reshape(-1,
1))

            encoded_movies_df =
encoded_movies_df.set_index('item_id')

            self.encoded_movies = encoded_movies_df
```

```
            f = 20  # Length of item vector that will be
indexed

            self.index = AnnoyIndex(f, 'angular')
            for i in encoded_movies_df.index:
                v = encoded_movies_df.loc[i].tolist()
                self.index.add_item(i, v)

            self.index.build(10)  # 10 trees

        def top_n(self, user_id, n=10):
            watched_movies = self.encoded_movies.merge(
                self.rating_df[self.rating_df.user_id ==
user_id][['item_id', 'rating']],
                how='inner', on='item_id')
            print(watched_movies.head())
            number_watched_movies = len(watched_movies)
            weights = watched_movies['rating'] /
watched_movies.rating.sum()
            watched_movies.drop(['rating', 'item_id'],
axis=1, inplace=True)
            mean_user_vector =
watched_movies.mul(pd.Series(weights),
axis=0).mean().to_list()
            ttl_recs = zip(

self.index.get_nns_by_vector(mean_user_vector,
number_watched_movies + n, include_distances=True)[0],

self.index.get_nns_by_vector(mean_user_vector,
number_watched_movies + n, include_distances=True)[1])
            recs_matrix = [[movie, distance] for movie,
distance in ttl_recs if
```

```
                            movie not in
self.rating_df[self.rating_df.user_id ==
user_id]['item_id']][:n]
            top_n = pd.DataFrame(recs_matrix,
columns=['item_id', 'distance'])
            return top_n
```

Додаток В

Лістинг програми, де реалізовано функцію для оцінювання точності рекомендацій.

```
    def evaluation(prediction_df):
        prediction_df_copy = deepcopy(prediction_df)
        prediction_df_copy['abs_delta'] =
abs(prediction_df['rating'] -
prediction_df['predicted_rating'])
        prediction_df_copy['squared_delta'] =
(prediction_df['rating'] -
prediction_df['predicted_rating']) ** 2
        mae =
prediction_df_copy.groupby('user_id')['abs_delta'].mean
().mean()
        mse =
prediction_df_copy.groupby('user_id')['squared_delta'].
mean().mean()
        rmse = np.sqrt(mse)
        return mae, rmse
```

Додаток Г

Лістинг програми, де реалізовано поділ вибірки на тестову та навчальну.

```
    def train_test_split(df, datetime_column,
train_part):
        df =
df.sort_values(datetime_column).reset_index(drop=True)
        datetime_dict =
df[datetime_column].value_counts().sort_index()
        split_datetime = df.loc[0, datetime_column]

        values_sum = 0
        for key, value in datetime_dict.items():
            if values_sum < len(df) * train_part:
                split_datetime = key
                values_sum += value

        train_df = df[df[datetime_column] <
split_datetime].reset_index(drop=True)
        test_df = df[df[datetime_column] >=
split_datetime].reset_index(drop=True)
        return train_df, test_df
```

Додаток Д

Event-driven process chain діаграма руху користувача по сайту: