МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів побудови рекомендаційної системи для онлайн-магазину електронних ігор
(тема)

Виконав:
Студент 2 курсу, групи ІПЗм-19-1
Байдак В. Є.
(прізвище, ініціали)

Спеціальність 121-Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Керівник доц. Мазурова О. О.
(посада, прізвище)

Допускається до захисту

Зав. кафедри _____ З.В. Дудар
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет *комп'ютерних наук*
Кафедра *програмної інженерії*
Рівень вищої освіти *другий (магістерський)*
Спеціальність *121 – Інженерія програмного забезпечення*
Тип програми *Освітньо-наукова*
Освітня програма *Інженерія програмного забезпечення*

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
_____

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента *Байдака Вадима Євгеновича*

1. Тема роботи Дослідження методів побудови рекомендаційної системи для онлайн-магазину електронних ігор
затверджена наказом по університету від «26» березня 2021 р. № 385 Ст
2. Термін подання роботи до екзаменаційної комісії «14» травня 2021 р.
3. Вихідні дані до роботи *електронні ресурси за обраною тематикою, мінімальні вимоги до функціональності програми, загальні вимоги до архітектури системи*
4. Перелік питань, що потрібно опрацювати в роботі *вступ, аналіз проблемної області і постановка задачі, перелік вимог до програмної системи, опис дослідження, об'єктних моделей, дослідження методів та алгоритмів, опис розробленої програмної реалізації, аналіз можливих застосувань та експлуатації системи.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів) *25 слайдів презентації*

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім’я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	доц. Мазурова О.О.		08.05.21

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
	Аналіз проблемної області дослідження	21.01.2021–30.01.2021	виконано
	Аналіз аналогів	31.01.2021–02.02.2021	виконано
	Розробка постановки задачі	02.02.2021–04.02.2021	виконано
	Дослідження існуючих методів	04.02.2021–14.02.2021	виконано
	Вдосконалення обраних методів	15.02.2021–28.02.2021	виконано
	Планування експериментального	01.03.2021–03.03.2021	виконано
	Моделювання (концептуальне, UML)	04.03.2021–11.03.2021	виконано
	Розробка схеми БД	12.03.2021–19.03.2021	виконано
	Проектування архітектури системи	20.03.2021–27.03.2021	виконано
	Створення коду програми	28.03.2021–20.04.2021	виконано
	Проведення експериментів	10.04.2021–17.04.2021	виконано
	Тестування програми	15.04.2021 –22.04.2021	виконано
	Підготовка пояснювальної записки	23.04.2021–05.05.2021	виконано
	Підготовка презентації та доповіді	03.05.2021–11.05.2021	виконано
	Попередній захист	12.05.2021	виконано
	Нормоконтроль, рецензування	14.05.2021	виконано
	Занесення диплома в електронний архів	15.05.2021	виконано
	Допуск до захисту у зав. кафедри	16.05.2021	

Дата видачі завдання __21_ _____січня_____ 2021р.

Студент _____ Байдак В. Є.

(підпис)

Керівник роботи _____ доц. Мазурова О.О.

(підпис)

РЕФЕРАТ / ABSTRACT

Кваліфікаційна робота магістра містить: 117 с., 32 рис., 2 табл., 22 джер.

ЕЛЕКТРОННА ГРА, МАШИННЕ НАВЧАННЯ, ОНЛАЙН-МАГАЗИН, РЕКОМЕНДАЦІЙНА СИСТЕМА, ASP.NET MVC 5, SQL SERVER, ML.NET, UML.

Об’єктом дослідження є методи побудови рекомендаційної системи в контексті онлайн-магазину електронних ігор.

Метою даної роботи є дослідження методів побудови рекомендаційної системи для онлайн-магазину електронних ігор.

Методи розробки базуються на концептуальному та UML-моделюванні предметної області, використанні фреймворку ASP.NET MVC 5 та бібліотеки машинного навчання ML.NET, мові програмування C#, базах даних SQL Server, мові розмітки HTML 5 та CSS3.

У результаті роботи проведено дослідження методів побудови рекомендаційної системи, спроектована схема бази даних та архітектура додатку, розроблена і апробована рекомендаційна система для онлайн-магазину електронних ігор.

ASP.NET MVC 5, ELECTRONIC GAME, MACHINE LEARNING, ML.NET, ONLINE STORE, RECOMMENDATION SYSTEM, SQL SERVER, UML.

The object of research is the methods of building recommendation systems in the context of the online store of electronic games.

The purpose of this work is to study the methods of building recommendation systems for the online store of electronic games.

Development methods are based on conceptual and UML-modeling of the subject area, using ASP.NET MVC 5 framework and ML.NET machine learning library, C# programming language, SQL Server database, HTML 5 mark-up language and CSS3.

As a result of the work the research of methods of construction of recommendation systems is carried out, the scheme of database and architecture of applications is developed, the recommendation system for online store of electronic games is developed and approved.

Я, Байдак Вадим Євгенович, студент гр. ІПЗм-19-1, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів побудови рекомендаційної системи для онлайн-магазину електронних ігор», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

В інтернеті, де кількість виборів величезна, існує потреба у фільтруванні, визначенні пріоритетів та ефективному наданні відповідної інформації, щоб полегшити проблему перевантаження інформацією, що створило потенційну проблему для багатьох користувачів інтернету. Рекомендаційні системи вирішують цю проблему шляхом пошуку у великому обсязі динамічно генерованої інформації, щоб надати користувачам персоналізований контент та послуги.

Різке зростання кількості доступної цифрової інформації та кількості відвідувачів Інтернету створило потенційну проблему перевантаження інформацією, яка заважає своєчасному доступу до предметів, що цікавлять користувачів в інтернеті. Системи пошуку інформації, такі як Google, Yahoo та Bing, частково вирішили цю проблему, але визначення пріоритетів та персоналізація (де система відображає наявний контент за інтересами та уподобаннями користувача) інформації відсутні. Це значно збільшило попит на рекомендаційні системи. Рекомендаційні системи – це системи фільтрації інформації, які вирішують проблему перевантаження інформацією, фільтруючи важливі фрагменти інформації з великої кількості динамічно генерованої інформації відповідно до уподобань користувача, інтересу або спостережуваної поведінки щодо товару. Рекомендаційна система має можливість передбачити, чи віддасть певний користувач перевагу об’єкту чи ні, на основі профілю користувача.

Рекомендаційні системи вигідні як постачальникам послуг, так і користувачам. Вони зменшують витрати на пошук та відбір предметів в середовищі інтернет-покупок. Рекомендаційні системи також покращують процес прийняття рішень. В середовищі електронної комерції рекомендаційні системи збільшують доходи постачальників послуг завдяки тому, що вони є ефективним засобом продажу більшої кількості товарів. Рекомендаційні системи підтримують користувачів, дозволяючи їм вийти за межі звичайного пошуку. Отже, необхідність використання ефективних і точних рекомендаційних методів у системі, яка

надаватиме відповідні та надійні рекомендації для користувачів, не можна недооцінювати [1].

Ігрова індустрія є одним із найважливіших та інноваційних секторів у сучасній технічній галузі. Ігри стали невід'ємною частиною життя сучасного суспільства. Це викликало потребу в онлайн магазинах продажу ігор, кількість яких сьогодні збільшується кожного дня. Існує багато різних типів відеоігор, різних жанрів, платформ, видавців тощо. Різноманіття ігор і їх властивостей може викликати перевантаження інформацією для користувачів, що призводить до проблем вибору бажаної гри. Це, у свою чергу, призводить до зменшення продажів ігор і до зменшення прибутків постачальників ігор. Усі ці фактори роблять рекомендаційні системи необхідними частинами сучасних онлайн-магазинів продажу ігор.

Метою даної роботи є дослідження методів побудови рекомендаційної системи для онлайн-магазину електронних ігор. Рекомендаційна система з використанням аналітичних профілів користувачів повинна розширювати веб-додатку продажу ігор GameStore, що забезпечить адаптованість функціоналу сайту та пропозицій на основі поведінки та запитів користувачів.

Під час виконання кваліфікаційної роботи проведено аналіз проблемної області розробки рекомендаційних систем, був проведений аналіз існуючих аналогів та методів побудови рекомендаційних систем, був запропонований і описаний вдосконалений метод побудови рекомендаційної системи та було проведене планування експериментального дослідження. Крім того, був проведений аналіз та UML-моделювання предметної області продажу ігор з використанням рекомендаційних систем, проектування бази даних, була розроблена архітектура системи і був описаний алгоритму надання рекомендацій для запропонованого методу. Було проведене експериментальне дослідження запропонованого методу і на основі цього методу була розроблена рекомендаційна система на базі онлайн-магазину електронних ігор. Також, був проведений аналіз дослідницької експлуатації та можливих застосувань запропонованого методу і розробленої системи.

Теоретичною і методологічною основою роботи є праці вітчизняних і зарубіжних вчених, в яких розглядаються практичні і теоретичні питання в області рекомендаційних систем (див. додаток А), розвитку електронної комерції та інформаційних систем та технологій, програмних засобів підтримки онлайн-систем з продажу товарів, машинного навчання.

В ході дослідження застосовувалися методи системного аналізу, порівняння, узагальнення, моделювання.

Отримані результати, програмний продукт рекомендаційна система для онлайн-магазину електронних ігор, можуть бути застосовані в системі електронної комерції для надання користувачам можливості знаходити бажані об’єкти для купівлі і для підвищення конкурентоспроможності онлайн магазину.

Робота перевірена на унікальність тексту в мережі інтернет та базі ХНУРЕ та на відповідність вимогам оформлення (див. додаток Б та Е).

За результатами кваліфікаційної роботи магістра було розроблено презентацію (див. додаток В). За результатами роботи створено тези доповіді на участь у науково-технічній конференції «Інноваційні технології-2020» та була написана стаття і подана до опублікування до науково-аналітичного журналу «Біоніка інтелекту» (див. додаток Г). Лістинг коду наведено в додатку Д.

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ

1.1 Аналіз проблемної області розробки рекомендаційних систем

На сьогоднішній день, великий потік інформації, доступний людині, мимоволі ініціює роботу з пошуку і фільтрації даних в ньому. Основним джерелом отримання інформації є пошукові системи. Однак при відсутності конкретного запиту складно знайти необхідну інформацію. Цю проблему частково вирішують рекомендаційні системи, здатні знаходити об'єкти схожі на те, що подобається або потрібно користувачеві на основі інформації про нього. Рекомендаційні системи аналізують інтереси користувачів і намагаються передбачити, що саме буде найцікавіше для конкретного користувача в даний момент часу.

Загалом, рекомендаційні системи – це алгоритми, спрямовані на те, щоб запропонувати користувачам релевантні предмети (будь то фільми для перегляду, книги для читання, товари для придбання, або щось інше, залежно від галузей).

Рекомендаційні системи є критично важливими в деяких галузях, оскільки вони можуть приносити величезний дохід, коли вони ефективні, або також можуть бути способом, який дозволяє суттєво виділитися серед конкурентів [2].

Однією зі сфер, де використовуються рекомендаційні системи, є комерційні сайти продажу товарів, зокрема онлайн-системи продажу електронних ігор. В сфері продажу ігор рекомендаційна система може виявити потреби відвідувачів онлайн-системи і в потрібний момент зробити цікаві саме їм пропозиції на сайті, збільшуючи дохід онлайн-системи за рахунок зростання конверсії, середнього чека і частоти повторних покупок [3].

За даними Європейського ринку ігор, у 2019 році понад 2,5 мільярди людей витрачали частину свого часу на електронні ігри. Величезна кількість експертів зробила індустрію електронних ігор однією з найцінніший у світі. Лише за останній рік прибуток цієї галузі зріс на 10%, досягнувши 116 млрд доларів. Однією з найбільших причин цього успіху є висока диверсифікація, яку має ця галузь протягом останніх років. Зараз можна знайти безліч ігрових платформ, величезну

кількість жанрів, ігрових категорій та навіть платформи, що забезпечують соціальну взаємодію між гравцями. Ця адаптивність породила величезну кількість об’єктів з різними атрибутами, які здатні привабити різноманітних користувачів. Прикладом адаптації є поширене створення онлайн-систем продажу електронних ігор, що займаються цифровим розповсюдженням ігор, дозволяють користувачам купувати, бачити відгуки, ділитися думками. Понад 40 мільярдів доларів у всьому світі щорічно витрачається на покупки електронних ігор в онлайн системах [4].

Однак, факт наявності різноманітності продуктів і великої кількості користувачів ускладнює вибір конкретної нової гри, яка може сподобатися користувачеві. Також, згідно звіту Steam, користувачі ніколи не грали у близько 37% придбаних ігор. Цей контекст створює потребу в рекомендаційних системах, які є системами, здатними робити релевантні персоналізовані пропозиції, попереджаючи користувачів про нові та невідомі для них ігри.

Гра у електронні ігри – це періодична активність і у цьому вона більше схожа на прослуховування музики, ніж на перегляд фільмів. В улюблену гру грають багато разів, але користувачі також хочуть відкривати для себе нові ігри. Це представляє подвійний виклик для галузі: потреба у іграх, у які користувачі захочуть повертатися, а також потреба в інструменті, який би допомагав користувачам знаходити нові ігри, які будуть споживатися так само як ті, що вже подобаються користувачу. Сучасні онлайн-системи продажу ігор дозволяють відстежувати взаємозв’язок між властивостями об’єктів та між вподобаннями користувачів. Ці відносини дозволяють стверджувати, що з другим викликом можна зіткнутися. Можливість розробити алгоритм, який вирішує вищезазначені проблеми, приносить великі переваги для індустрії електронних ігор, спільноти користувачів і навіть для розробників ігор, передбачаючи, чого хочуть користувачі найбільше, а також просуваючи нові ігри [5].

Рекомендаційна система визначається як стратегія прийняття рішень для користувачів у складних інформаційних середовищах. Крім того, рекомендаційна система була визначена, з точки зору електронної комерції, як інструмент, який допомагає користувачам шукати в записах об’єкти, пов’язані з їх інтересами та

уподобаннями. Також, рекомендаційна система визначається як засіб надання допомоги та доповнення соціального процесу, використовуючи рекомендації інших користувачів, щоб робити вибір, коли немає достатньо особистого знання або досвіду щодо альтернативних варіантів. Рекомендаційні системи вирішують проблему перевантаження інформацією, з якою зазвичай стикаються користувачі, надаючи їм персоналізовані, ексклюзивні рекомендації щодо контенту та послуг. Існує декілька підходів до побудови рекомендаційних систем, які можуть використовувати колаборативну фільтрацію, контентну фільтрацію або гібридний підхід. Колаборативна техніка фільтрування є найбільш зрілою та найпоширеніше реалізованою [6].

Колаборативна фільтрація рекомендує елементи, визначаючи інших користувачів зі схожим смаком. Контентна фільтрація співвідносять властивості контенту і характеристики користувача. Гібридний підхід поєднує в собі інструменти колаборативної і контентної фільтрації.

Крім того, для підвищення точності рекомендацій використовуються різноманітні методи кластеризації для поділення користувачів на групи – кластеризації на основі мінімума, максимума, групового середнього, відстаней між центроїдами, методу Вона.

На сьогоднішній день майже всі рекомендаційні системи для онлайн магазинів ігор базуються на веб-додатках. Веб-додаток – це комп’ютерна програма, яка використовує веб-браузери та веб-технології для виконання завдань в інтернеті.

Веб-додатки використовують комбінацію скриптів на стороні сервера для обробки та пошуку інформації та клієнтських скриптів для представлення інформації користувачам. Це дозволяє користувачам взаємодіяти з контентом за допомогою онлайн-форм, систем управління контентом, кошиків для покупок тощо. Крім того, додатки дозволяють працівникам створювати документи, ділитися інформацією, співпрацювати над проектами та працювати над загальними документами незалежно від місця розташування чи пристрою.

Веб-додатки зазвичай кодуються мовою, що підтримується браузером, такою як JavaScript та HTML. Деякі додатки є динамічними, що вимагає обробки на стороні сервера. Інші повністю статичні, без необхідності обробки на сервері.

Веб-додаток потребує веб-сервера для управління запитами від клієнта, сервера додатку для виконання запитуваних завдань і, іноді, роботи з базою даних для зберігання інформації [7].

Для коректної роботи рекомендаційної системи на базі веб-додатку продажу ігор, вона повинна оперувати такими даними як інформація про ігри (назва, жанри, платформа, популярність, рейтинг і т. д.), інформація про користувачів (логін, пароль, роль, вподобання), інформація про замовлення користувачів (дата замовлення, обрані ігри і т. д.), інформація про оцінки ігор від користувачів. Усі ці дані, потрібні для рекомендаційної системи, повинні бути систематизовані і збережені в спеціальному сховищі. Таким сховищем буде база даних, вона стане однією із найважливіших частин системи в цілому, тож треба ретельно підбирати її тип і спосіб використання.

Зараз найбільш поширеним типом баз даних є реляційні бази. Реляційна модель, яка використовується в таких базах даних, надає наступні переваги:

– простота і доступність для розуміння кінцевим користувачем. Єдиною інформаційною конструкцією є таблиця;

– при проектуванні реляційних баз даних застосовуються суворі правила, що базуються на математичному апараті;

– реляційна модель забезпечує повну незалежність даних. При зміні структури реляційної бази даних зміни, які потрібно зробити в прикладних програмах, як правило, мінімальні;

– маніпулювання даними на рівні мови системи управління базами даних (СУБД) проводиться ненавігаційно, тому для побудови запитів і написання прикладних програм немає необхідності знання конкретної організації бази даних у зовнішній пам'яті.

Крім того, при розробці рекомендаційної системи, важливим фактором є архітектура цієї системи. Архітектура програмного забезпечення – сукупність

найважливіших рішень про організацію програмної системи. Фіксація в архітектурі на початкових стадіях розробки прийнятих рішень по дизайну системи, багато в чому, визначає можливість досягнення необхідних атрибутів якості готового продукту.

Архітектурою, яка поєднує в собі простоту розробки і ефективність роботи, є багаторівнева (багатошарова) архітектура – клієнт-серверна архітектура, яка ділить додаток на логічні та фізичні рівні. Рівні є способом розділити обов'язки та керувати залежностями. Кожен рівень несе певну відповідальність. Вищий рівень може використовувати послуги нижчого рівня, але не навпаки. Традиційний трьох-рівневий додаток має рівень презентації, середній рівень і рівень бази даних. Середній рівень не є обов’язковим. Складніші програми можуть мати більше трьох рівнів.

На реальному проекті з розробки рекомендаційної системи на базі онлайн-сервісу з продажу ігор буде доцільно використовувати реляційну базу даних з СУБД, яка дозволяє зручно і просто керувати великою кількістю структурованих даних та будувати систему на основі багаторівневої архітектури, що додає розробці додатку гнучкості [7].

1.2 Аналіз існуючих аналогів

На сьогоднішній день рекомендаційні системи впроваджені в багатьох онлайн-системах із різноманітних галузей. Найкрупнішими з них є Steam, Netflix, Spotify

Netflix – американський провайдер медійних послуг та продюсерська компанія. Основною сферою бізнес-діяльності компанії є надання передплачуваних послуг із мережевої трансляції бібліотеки кінофільмів та телепередач (включно зі створеними самою «Netflix») [8].

Netflix пропонує персоналізовані рекомендації, які допомагають

користувачам знайти шоу та фільми, що їх цікавлять. Для цього Netflix створили власну, складну систему рекомендацій.

Netflix оцінює ймовірність того, що користувач перегляне певний фільм з каталогу, базуючись на ряді факторів:

– взаємодії з сервісом (наприклад, історія переглядів та оцінка інших фільмів);
– інші учасники зі схожими смаками та уподобаннями на цьому сервісі;
– інформація про фільми (жанр, категорії, актори, рік випуску тощо).

Окрім збору інформації про те, що користувач дивився, Netflix збирає наступні дані:

– час доби перегляду фільмів;
– пристрої, на яких користувач переглядає фільми;
– як довго користувач переглядає фільми.

Усі ці показники використовуються в якості вхідних даних для рекомендаційних алгоритмів. Рекомендаційна система Netflix не включає демографічну інформацію (наприклад, вік чи стать) для прийняття рішень.

Коли користувач створює обліковий запис Netflix, йому пропонується вибрати кілька фільмів, які йому подобаються. Ці фільми будуть використані для підбору початкових рекомендації. Якщо користувач не обрав улюблені фільми, йому будуть запропоновані популярні фільми в якості рекомендацій.

Як тільки користувач починає переглядати фільми в Netflix, початкові вподобання будуть замінені на нові. Нещодавно переглянуті фільми будуть переважувати ті, що були переглянуті раніше, для алгоритмів рекомендаційної системи.

Окрім вибору фільмів, які включаються в рядки на домашній сторінці Netflix, система також ранжує кожен фільм у рядку, а потім ранжує самі рядки, використовуючи рекомендаційні алгоритми.

У кожному рядку є три рівні персоналізації:

– вибір рядка (наприклад, «Продовжуйте перегляд», «Популярні зараз», «Комедії, що отримали нагороди» тощо);

– вибір фільмів, які відображаються в рядку;

– вибір позиції фільмів у рядку.

Найбільш рекомендовані рядки розміщуються вгорі. Найбільш рекомендовані фільми знаходяться зліва в кожному рядку.

Spotify – цифрова музична платформа на базі хмарних технологій, що забезпечує доступ з різних пристроїв до понад 50 мільйонів музичних композицій, а також до швидко зростаючої кількості подкастів та відео.

Рекомендаційна система Spotify надає пропозиції для користувачів на основі їхньої взаємодії із системою в минулому (прослуховування/ пропускання/ додавання до плейлисту), атрибутів пісень/виконавців, яких вони слухають, та уподобань «подібних» користувачів.

Spotify використовує глибоке навчання для автоматизації процесу та виявлення прихованих закономірностей між виконавцями, жанрами та уподобаннями користувачів.

Spotify використовує три рекомендаційні моделі:

– колаборативна фільтрація: використовує поведінку користувача та поведінку подібних йому користувачів;

– обробка природної мови (NLP): для текстів пісень, плейлистів, публікацій у блогах, коментарів;

– аудіо моделі: використовуються для необробленого аудіо.

Кожен користувач Spotify має свій власний профіль вподобань, який визначається тим, що він слухає, коли слухає, як часто тощо.

Рекомендаційна система Spotify також видає рандомізовані рекомендації для нових користувачів, щоб сформувати більше відгуків.

За допомогою обробки природної мови, Spotify виділяє нові атрибути пісень, відмінні від жанрів, які можуть також слугувати вхідними даними для рекомендаційної системи.

Аудіо моделі застосовуються для нових пісень, щоб вони потрапляли до користувачів і генерувалися дані необхідні для рекомендаційної системи. Spotify використовує той тип нейронної мережі, який використовується пошуковими

системами для розуміння вмісту зображень. Ці мережі обробляють необроблений звук і знаходять цілий ряд характеристик, включаючи тональність, темп і навіть гучність. В результаті Spotify може розмістити нові пісні у відповідних плейлистах, де вони можуть генерувати нові дані для побудови рекомендацій, коли користувачі взаємодіють із вмістом плейлиста [9].

Steam – це онлайн-платформа від розробника ігор Valve, де користувачі можуть купувати, грати, створювати та обговорювати ігри для персонального комп'ютера. Платформа розміщує тисячі ігор (а також завантажуваний контент, або DLC, та створені користувачем додатки, що називаються "модами") від великих розробників та інді дизайнерів ігор.

Рекомендаційна система Steam створює персоналізовані списки ігор, які можуть сподобатися користувачам на основі того, у що вони грали раніше.

Рекомендаційна система Steam використовує машинне навчання для побудови рекомендацій на основі колаборативної фільтрації.

Система дивиться на те, в які ігри грає користувач та в які ігри грають інші користувачі і після цього робить обґрунтовані пропозиції на основі рішень інших людей, які грають в ігри у Steam. Ідея полягає в тому, що якщо гравці з подібними до користувача ігровими звичками схильні грати в гру, яку користувач ще не пробував, тоді ця гра, швидше за все, буде доцільною рекомендацією для користувача.

На рекомендації Steam також впливають:

- мітки ігор (екшн, для одного гравця, відмінний саундтрек тощо), в які грав користувач;
- відгуки користувачів;
- популярність ігор на даний момент;
- наявність ігор у друзів;
- наявність ігор в списку лідерів продажів.

Наведені сервіси є лідерами у своїх галузях і не в останню чергу через правильно побудовані рекомендаційні системи, які допомагають користувачам знайти саме той контент, який їм цікавий.

1.3 Постановка задачі

Метою даної роботи є дослідження методів побудови рекомендаційної системи для онлайн-магазину електронних ігор. Практичною задачею є розширення веб-додатку продажу ігор GameStore за допомогою впровадження у додаток рекомендаційної системи з використанням аналітичних профілів користувачів, що забезпечить адаптованість функціоналу сайту та пропозицій на основі поведінки та запитів користувачів.

Для коректної роботи рекомендаційної системи на базі веб-додатку продажу ігор повинно бути реалізоване наступне:

- можливість залишати оцінки та відгуки для ігор;
- механізм відстеження популярності ігор;
- збір та зберігання даних замовлень користувачів;
- можливість створення власного профілю користувача з інформацією про нього;
- можливість вибору користувачем улюбленого жанру;
- механізм вибору персоналізованих рекомендацій.

У ході виконання кваліфікаційної роботи необхідно вирішити наступні завдання:

- провести аналіз методів побудови рекомендаційних систем та обрати найкращі для подальшого дослідження;
- вдосконалити обраний метод побудови рекомендаційної системи для онлайн-системи продажу ігор;
- спланувати експериментальне дослідження розробленого методу;
- провести аналіз та моделювання предметної області продажу ігор;
- спроектувати і розробити базу даних відповідно предметній області;
- спроектувати архітектуру та програмно реалізувати рекомендаційну систему для онлайн-магазину продажу ігор на базі вдосконаленого методу;

– провести експериментальне дослідження відносно ефективності розробленого методу;

– протестувати систему.

Для реалізації рекомендаційної системи на базі веб-додатку продажу ігор були обрані наступні технології. Вся система буде базуватися на фреймворці для створення веб-додатків ASP .NET MVC 5 та на бібліотеці машинного навчання ML.NET.

Для доступу до реляційних даних буде використовуватися СУБД MS SQL Server 2016, ORM Entity Framework та FluentAPI. Для доступу до нереляційних даних буде використовуватися MongoDB та C# бібліотека MongoDB.Driver.

Для реалізації ін’єкції залежностей буде використовуватися IoC-контейнер Autofac.

Для створення уявлень онлайн сервісу буде використовуватися ASP.NET Razor, HTML 5, CSS 3, Boostrap 4 та JavaScript і JQuery.

Після постановки задачі можна переходити до опису прийнятих проектних рішень і моделювання системи.

2 ОПИС ПРИЙНЯТИХ ПРОЕКТНИХ РІШЕНЬ

2.1 Дослідження існуючих методів побудови рекомендаційних систем

2.1.1 Поняття рекомендаційної системи

Рекомендаційні системи (РС) – програми, які намагаються передбачити, які об'єкти (ігри, фільми, музика, книги, новини, веб-сайти) будуть цікаві користувачеві, маючи певну інформацію про його профіль. Прогнозування може бути зроблене на підставі інформації про користувачів, об'єкти і попередньо введені користувачами оцінки. РС можуть бути використані як в комерційних цілях для пошуку цільових груп користувачів і просування товару в цих групах, так і для організації експертного оцінювання, де РС дозволяють істотно знизити навантаження на експертів.

РС працюють з наступними вихідними даними:

– $u \in U \subset$ N – ідентифікатори користувачів РС;

– $i \in I \subset$ N – ідентифікатори об'єктів предметної області РС, наприклад, ігри в РС в галузі комп’ютерних ігор;

– ρ: $U \times I \rightarrow [0,1]$ – функція оцінки близькості об'єктів; значення ρ(u, i) показує, наскільки об'єкти i і u близькі; як правило, оцінки близькості задаються самими користувачами за час роботи з РС; будемо вважати, що чим менше значення оцінки, тим об'єкти ближче; будемо говорити, що між користувачем u і об'єктом i виконується відношення близькості R, якщо $\rho(u, i) \leq \varepsilon_0 \in \varepsilon(0)$, будемо називати такі об'єкти близькими.

Як правило, РС вирішують наступні два завдання (користувач, для якого виконується рішення, називається активним і позначається символом u_a):

– завдання прогнозування: спрогнозувати невідоме значення $\rho(u_a, i_p) = \perp$ (символом ⊥ будемо позначати невідоме значення) шляхом алгоритмічного обчислення значення прогнозної функції $\bar{\rho}(u_a, i_p)$: $U \times I \rightarrow [0,1]$, де i_p –

прогнозований об'єкт; при цьому потрібно, щоб прогноз був складений точно, тобто $|\bar{\rho}(u_a, i_p) - \rho(u_a, i_p)| \leq \varepsilon_0$;

– завдання $topN$ – формування підмножини об'єктів:

$$I_{topN} = \{i: (u_a R_i) \wedge \rho(u_a, i) = \perp\} \wedge |I_{topN}| = N.$$

Так як невідомо, чи виконується відношення $u_a R_i$ в силу того, що $\rho(u_a, i) = \perp$, то виконання відношення $u_a R_i$ визначається за значенням прогнозної функції: $u_a R_i \Leftrightarrow \bar{\rho}(u_a, i) \leq \varepsilon_0$.

Рішення названих завдань проводиться РС за рахунок аналізу інформації про характеристики користувачів і об'єктів. X – безліч характеристик користувачів, наприклад, соціально-демографічні показники РС онлайн-магазину. Y – безліч характеристик об'єктів, наприклад, найменування жанрів ігор. Значенням характеристик користувачів є значення вагової функції w_U: $U \times X \rightarrow [0,1]$, об'єктів – w_I: $I \times Y \rightarrow [0,1]$. Значення вагів можуть задаватися користувачами, експертами, алгоритмічно і т. д. Структуру даних, яка представляє інформацію про користувача u і об'єкт i, назвемо контентом користувача $c_X(u)$ і контентом об'єкта $c_Y(i)$ відповідно.

Модель РС – це трійка:

$$(c_X; c_Y; П)$$

де П – правило алгоритмічного обчислення значення прогнозної функції $\bar{\rho}$.

Щоб визначити якість виконання завдання, проводиться тестування, для якого вихідна множина даних P розбивається на навчальну і тестову множини P_0 і $P_\perp$ відповідно. Якщо $\rho(u, i) \in P_0$, будемо позначати такі об'єкти i_0. Якщо $\rho(u, i) \in P_\perp$, будемо позначати такі об'єкти $i_\perp$.

Можна виділити два основних типи рекомендаційних систем:

а) рекомендаційні системи на основі контентної фільтрації (Content-based):

1) користувачеві рекомендуються об'єкти, схожі на ті, які цей користувач вже вжив;

2) схожості оцінюються за ознаками вмісту об'єктів;

3) сильна залежність від предметної області, корисність рекомендацій обмежена.

б) рекомендаційні системи на основі колаборативної фільтрації (Collaborative Filtering):

1) для рекомендації використовується історія оцінок як самого користувача, так і інших користувачів;

2) більш універсальний підхід, часто дає кращий результат;

3) є свої проблеми (наприклад, холодний старт).

2.1.2 Контентна фільтрація

Контентна фільтрація формує рекомендації, ґрунтуючись на описі рекомендованих об'єктів і відношенні користувача до характеристик об’єкту. Таким чином, на основі раніше зроблених оцінок формуються ставлення до тих чи інших характеристик, які використовуються для наступних рекомендацій. Наприклад, якщо користувач високо оцінює екшн-ігри та стратегії і ставить низькі бали симуляторам, то стає очевидно, ігри з якими характеристиками більше задовольнять користувача.

Цей підхід ґрунтується на порівнянні характеристик товару (content) з вподобаннями користувача, які формуються на основі історії оцінок або покупок користувача. Чим більше товар відповідає цим вподобанням користувача, тим вище вірогідність того, що він сподобається користувачу і тим вище рівень рекомендованості цього товару. Очевидно, що цей підхід потребує наявності характеристик всіх товарів.

Зазвичай предметами контентної фільтрації є товари, які мають неструктуровані характеристики, наприклад такі товари як фільми, ігри, музика тощо. До неструктурованих характеристик можна віднести відгуки, описи, рецензії тощо. Але для контентної фільтрації можна використовувати і звичайні числові та категоріальні характеристики – жанри, теги тощо.

Для опису неструктурованих ознак використовується типовий для тексту спосіб – вектори в просторі слів. Ці вектори складаються з елементів-ознак, що потенційно характеризує інтерес користувача. Так само продукт є вектором в тому ж просторі.

Коли користувач взаємодіє з системою (скажімо, він купує ігри), векторні описи придбаних ним товарів об'єднуються (підсумовуються і нормалізуються) в єдиний вектор. Цей вектор є вектором інтересів користувача. Далі, для надання рекомендацій, необхідно знайти товари, описи яких є найближчими до вектору інтересів користувача. Для цього використовуються алгоритми пошуку n найближчих сусідів.

В якості міри близькості двох векторів найчастіше використовується косинусну відстань:

$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} =$$

$$= \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

де A – вектор інтересів першого користувача,

B – вектор інтересів другого користувача.

2.1.3 Колаборативна фільтрація

Колаборативна фільтрація – це набір технік, в яких для надання рекомендацій відносно об’єктів, які користувач ще не оцінював, використовуються дані про інших користувачів, про їх відношення до різних об’єктів і історію обраного користувача. В основу цього підходу покладене наступне допущення – користувачі, які в минулому однаково оцінювали об’єкти, мають схожі смаки і тому будуть оцінювати об’єкти схожим чином у майбутньому.

Існують три типи колаборативної фільтрації: підхід, заснований на моделі, заснований на сусідстві та гібридний. Ці підходи відрізняються концептуально і цю різницю можна продемонструвати наступним прикладом. В якості вхідних даних візьмемо матрицю, яка містить оцінки різних об’єктів користувачами так, що рядок цієї матриці містить оцінки конкретного користувача по всім об’єктам, а стовпець містить усі оцінки для конкретно об’єкта. Підхід, заснований на моделі (model-based підхід), доповнюючи цю матрицю за допомогою алгоритмів машинного навчання, намагається передбачити оцінки користувачів. Підхід, заснований на сусідстві, обирає рекомендації, знаходячи найбільш схожі стовпці (item-based підхід) або рядки (user-based підхід). Гібридний підхід є поєднанням вищеописаних підходів. Колаборативна фільтрація на основі гібридного підходу може бути більш ефективною і видавати більш точний результат, але цей підхід є дорогим за часом виконання та складним в застосуванні і реалізації.

В класичному вигляді алгоритм використовує принцип n найближчих сусідів. Бракуюча інформація про користувача доповнюється даними по його n найближчим сусідам (користувачам найбільш схожим на поточного в термінах вподобань). В даному випадку під схожість мається на увазі відповідність абр кореляція інтересів, яка може визначатися різними способами.

До недоліку описаного алгоритму можна віднести те, що через квадратичну складність він погано застосовується на практиці. Будь-який алгоритм заснований на принципі найближчих сусідів вимагає розрахунок відстаней між усіма парами

користувачів (користувачів може бути дуже багато). Складність за пам’яттю розрахунку матриці відстаней дорівнює O(n^2m), де n – кількість користувачів, а m – кількість товарів. Якщо користувачів буде дуже багато, наприклад мільйон, то для зберігання матриці відстаней в необробленому стані необхідно мати щонайменше 4 терабайти вільної пам’яті.

Цю проблему можна частково вирішити придбанням високопродуктивної апаратури. Але більш доцільним підходом до вирішення цієї проблеми є оптимізації алгоритму:

– матриця відстаней повинна оновлюватися пакетами (раз в визначений інтервал часу), а не при кожній покупці або оцінці користувача;

– матриця відстаней повинна оновлюватися інкрементально, а не перераховуватися повністю;

– необхідно використовувати наближені та ітеративні алгоритми, наприклад ALS.

Для забезпечення ефективності алгоритму, необхідно, щоб виконувалися наступні умови:

– інтереси користувачів не повинні змінюватися з часом, або можуть змінюватися, але для всіх користувачів однаково;

– якщо інтереси користувачів збігаються, то вони повинні збігатися в усьому.

РС, які використовують колаборативну фільтрацію в якості правила П, будемо називати колаборативними РС (далі КРС). Вони діляться на два типи по множині, яка фільтрується: множини користувачів та об'єктів. Будемо називати перші суб'єктно-орієнтованими (далі СОК), а останні – об'єктно-орієнтованими (далі ООК).

Опишемо теорію, на якій ґрунтуються колаборативні П. Рішення будується по навчальній множині, а її якість визначається по тестовій. Навчальна множина виступає в ролі інформації минулого часу, тестове – в ролі інформації майбутнього часу. Правило П СОК ґрунтоване на твердженні, яке свідчить, що якщо у минулому

користувачі були близькі по перевагам, то і в майбутньому вони будуть близькі по перевагам. У введеній термінології це затвердження прийме наступний вигляд:

$$u_a R_u\ u \text{ виконується на } \mathrm{P}_0 \Rightarrow u_a R_u\ u \text{ виконується на } \mathrm{P}_\perp$$

де u_a – активний користувач,

u – користувач з множини користувачів $U \subset \mathbb{N}$,

P_0 – навчальна множина даних,

$\mathrm{P}_\perp$ – тестова множина даних,

R_u – відношення близькості користувачів.

Виконання відносини близькості R_u між користувачами встановлюється КРС на підставі значень характеристик користувачів. Характеристиками для СОК завжди виступають об'єкти, а значеннями вагів – значення $\rho(u, i) \in \mathrm{P}_0$, які були виставлені самими користувачами і характеризують переваги користувачів. Для визначення близькості по перевагах використовуються так звані заходи близькості:

$$\delta_u{:}U \times U \to [0,1]{:}\ (1 - \delta_u\ (u, v)) \le \varepsilon_0 \Leftrightarrow uR_u v.$$

Користувачі, між якими виконується відношення близькості, називаються сусідами.

Правило П СОК задається формулою:

$$u \in U, (u_a R u) \Rightarrow |\bar{\rho}(u_a, i_p) - \rho(u_a, i_p))| \le \varepsilon_0,$$
$$\rho(u_a, i_p) = f(\{\rho(u, i_p)\})$$

Правило П СОК говорить про те, що якщо користувачі u є сусідами для користувача u_a, то оцінки $\rho(u_a, i_p)$, $\rho(u, i_p)$ корелюють, тому невідоме значення

$\rho(u_a, i_p)$ можна функціонально визначити по значенням $\{\rho(u, i_p)\}$, тобто прогнозна функція є функцією від значень оцінок близькості сусідів.

Правило П ООК засноване на твердженні: якщо користувачеві подобається об'єкт i, який близький за характеристикам до об'єкта j, то користувачеві сподобається об'єкт j. З урахуванням введеної термінології дане твердження набуде вигляду:

$$(u_a R i) \wedge (i R_i j) \Rightarrow u_a R j$$

де R_i – відношення близькості об'єктів.

Відношення близькості R_i між об'єктами встановлюється РС на підставі значень мір близькості: 1 - $\delta_i(i, j) \leq \varepsilon_0 \Leftrightarrow i R_i j$, δ_i: $I \times I \rightarrow [0,1]$ – міра близькості об'єктів. Об'єкти, між якими виконується відношення близькості, називаються сусідами.

При вирішення завдання $topN$ в ООК використовується інформація тільки про ті об'єкти, для яких відомо, що $(u_a R i_0)$, $(u_a R i_\perp)$, тому будемо вважати, що $P = \{\rho(u, i)$: $uRi\}$ для завдання $topN$.

Правило П ООК задається формулою:

$$(i R_i i_0)) \Rightarrow (\bar{\rho}(u_a, i) = 0) \Rightarrow u_a R i.$$

Значення $\bar{\rho}(u_a, i)$ задаються рівними нулю, тому що тоді об'єкти i будуть близькі активному користувачеві при будь-якому пороговому значенні ε_0.

Правило виводу ООК говорить про те, що якщо існує об'єкт i, що є сусідом об'єкта i_0, то, слідуючи евристичному твердженню, $u_a R i$, так як $u_a R i_0$ за прийнятим для завдання $topN$ вигляду вихідної множини.

2.1.4 Порівняння методів побудови рекомендаційних систем

Розглянемо основні недоліки і переваги колаборативної фільтрації по відношенню до контентної. До недоліків можна віднести:

– необ'єктивні та упереджені оцінки недобросовісних користувачів можуть призвести до погіршення якості рекомендацій іншим користувачам;

– проблема «білих ворон». Під «білими воронами» маються на увазі такі користувачі, які мають унікальні переваги до об’єктів і специфічний смак, який відрізняється від загального тренду уподобань інших користувачів. Рекомендації для «білих ворон» мають знижену якість;

– проблема доступу до персональних даних користувачів (конфіденційність). Оскільки колаборативна фільтрація оперує персональними даними користувачів (наприклад, оцінки товарів), деякі користувачі можуть негативно ставитися до факту обробки їх персональної інформації;

– колаборативна фільтрація потребує відносно складних та трудомістких обчислень.

До переваг можна віднести:

– колаборативна фільтрація добре справляється з проблемою «холодного старту», яка властива всім рекомендаційним системам, за рахунок більш швидкого підвищення якості рекомендацій [10];

– колаборативна фільтрація значно швидше реагує на зміну вподобань користувачів;

– для надання рекомендації колаборативна фільтрація не потребує наявності описових характеристик предметів, що рекомендуються.

Для реалізації у проекті була обрана колаборативна фільтрація заснована на моделі оскільки вона надає найбільш точний рекомендаційний прогноз і дозволяє збільшувати швидкість підвищення якості рекомендацій.

2.2 Дослідження методів розбиття користувачів на групи

2.2.1 Використання аналітичного профіля

Найбільшою проблемою для рекомендаційних систем на основі колаборативної фільтрації є той факт, що користувачі не оцінюють усі наявні предмети, а тільки якусь їх частину. Це призводить до того, що вхідна матриця для колаборативної фільтрації, яка містить оцінки предметів користувачами, найчастіше сильно розріджена. Цей факт призводить до посилення проблем «білих ворон», «холодного старту» та до зниження якості рекомендацій і ефективності рекомендаційної системи в цілому [11, 12]. Методи контекстної фільтрації не завжди бувають ефективними при спробі виправити цю ситуацію. Користувачі не завжди здатні коректно висловити те, чого вони хочуть, найчастіше це неможливо. Наприклад, користувачі хочуть зіграти в хорошу гру, а така гра характеризується не жанрами, платформами або іншими загальними характеристиками. Неможливо виділити загальні характеристики, які допоможуть відрізнити погану гру від хорошої – ці характеристики, в багатьох випадках, індивідуальні для кожного користувача. Це можна підтвердити рейтингом найпопулярніших ігор, наприклад, з сайту Metacritic. У першій десятці представлені найрізноманітніші ігри: стратегії, гонки, шутери, симулятори. Деякі з цих жанрів навіть несумісні.

Така ситуація призводить до пошуку інших способів покращення якості рекомендацій колаборативної фільтрації. Одним з таких способів пропонується використання характеристик користувачів. Ці характеристики можуть бути виділені в так званий аналітичний профіль. До цих характеристик можна віднести: стать, вік, рівень освіти, професію, творчі інтереси, вподобання та інші. Знаходження прямих або зворотних залежностей між характеристиками користувача і його відношенням до рекомендованих предметів є дуже складною задачею, але є інше рішення. Це рішення полягає в тому, щоб розділяти користувачів на окремі групи на основі аналітичних профілів і для визначення рекомендованих предметів використовувати в якості вхідних даних для

колаборативної фільтрації інформацію про тих користувачів, які належать до групи обраного користувача.

Дане завдання можливо вирішити двома шляхами і обидва шляхи є предметом машинного навчання. Перший шлях це кластеризація. Серед безлічі аналітичних профілів користувачів за допомогою спеціального алгоритму виділяється логічна структура і на її основі користувачі поділяються на групи (кластери). Другий шлях це класифікація. Кожен користувач відноситься до заздалегідь виділених соціальних груп на основі близькості. Перший варіант представляється більш привабливим, оскільки на ранньому етапі розробки проекту неможливо передбачити які користувачі будуть використовувати дану рекомендаційну системою і яке буде їх співвідношення.

2.2.2 Дослідження методів кластеризації

Кластеризація – це завдання розбиття об'єктів на групи таким чином, щоб схожі об'єкти були віднесені до однієї групі (кластеру). Дане завдання вирішується в таких важливих сферах як пошук інформації, аналіз зображень, біоінформатика, розпізнавання образів та інші. Виділяють два основних типи кластеризації: ієрархічна кластеризація та кластеризація на основі центроїдів, розглянемо їх.

Ієрархічна кластеризація – метод кластеризації заснований на побудові ієрархії кластерів. Існує дві стратегії, які реалізують даний підхід: агломеративна і дивізійна.

Агломеративний алгоритм спочатку виділяє кожен об'єкт як окремий кластер і потім починає об'єднувати схожі пари виділених кластерів, поступово зменшуючи їх загальну кількість. На кожній ітерації подібні кластери зливаються з іншими кластерами, поки не утворюється один кластер або K кластерів.

Алгоритм агломеративної ієрархічної фільтрації виглядає наступним чином:

– обчислюється матриця близькості;

– кожна точка даних береться за окремий кластер;

– об’єднуються два найближчі кластери та оновлюється матриця близькості поки не залишиться лише один кластер (або К кластерів).

Ключова операція цього алгоритму – обчислення близькості двох кластерів.

Дивізійний алгоритм є прямою протилежністю агломеративному алгоритму і відштовхується від ідеї, що всі об'єкти спочатку знаходяться в одному кластері, який рекурсивно ділиться на підкластери. Несхожі точки даних відокремлюються від кластера і кожна відокремлена точка даних розглядається як окремий кластер. В результаті отримується n кластерів.

Обчислення близькості двох кластерів є важливою процедурою для об'єднання або розділення кластерів. Існують декілька підходів, які використовуються для обчислення близькості двох кластерів:

– мінімум;

– максимум;

– групове середнє;

– відстань між центроїдами;

– метод Варда.

Обчислення близькості на основі мінімума, також відомий як алгоритм Single-Linkage – близькість двох кластерів C1 і C2 дорівнює мінімуму близькостей між точками Pi і Pj, такими, що Pi належить C1, а Pj належить C2.

Перевага обчислення близькості на основі мінімума – може відокремлювати нееліптичні форми, доки розрив між двома кластерами не малий.

Недолік обчислення близькості на основі мінімума – не може правильно розділити кластери, якщо між ними є шум.

Обчислення близькості на основі максимума, також відомий як алгоритм Complete-Linkage – близькість двох кластерів C1 і C2 дорівнює максимуму близькостей між точками Pi і Pj, такими, що Pi належить C1, а Pj належить C2.

Перевага обчислення близькості на основі максимума – добре працює при розділенні кластерів, якщо між кластерами є шум.

Недоліки обчислення близькості на основі максимума – упереджений до глобулярних кластерів та має тенденцію розбивати великі кластери.

Обчислення близькості на основі групового середнього – бере усі пари точок з різних кластерів і обчислює їх близькості, а потім середнє значення цих близкостей.

Перевага обчислення близькості на основі групового середнього – добре працює при розділенні кластерів, якщо між кластерами є шум.

Недоліки обчислення близькості на основі групового середнього – упереджений до глобулярних кластерів.

Обчислення близькості на основі відстані між центроїдами – обчислюються центроїди двох кластерів C1 і C2 і близькість між двома центроїдами приймається за близькість між двома кластерами. Даний підхід є найменш популярним в реальному світі.

Обчислення близькості на основі методу Варда – точно таке ж, як і на основі групового середнього, за винятком того, що метод Варда обчислює суму квадратів відстаней між Pi та Pj.

Перевага обчислення близькості на основі методу Варда – добре працює при розділенні кластерів, якщо між кластерами є шум. Недоліки обчислення близькості на основі групового середнього – також; упереджений до глобулярних кластерів.

Складність за пам’яттю для ієрархічної кластеризації дуже висока, коли кількість точок даних велика, оскільки необхідно зберігати матрицю близькості в оперативній пам'яті. Складність простору порядку квадрата n:

$$\text{Складність за пам’яттю} = O(n^2),$$

де n – кількість точок даних.

Складність за часом для ієрархічної кластеризації – оскільки необхідно виконати n ітерацій і в кожній ітерації, необхідно оновити матрицю близькості та відновити матрицю, складність за часом також дуже велика. Складність за часом порядку куба n.

$$\text{Складність за часом} = O(n^3),$$

де n – кількість точок даних.

Обмеження ієрархічної кластеризації:

- для ієрархічної кластеризації не існує математичної мети;
- усі підходи до розрахунку близькості між кластерами мають свої недоліки;
- висока складність за пам’яттю та часом.

Кластеризація на основі центроїдів. В даному методі завдання ставиться таким чином: вводяться центри кластерів, і об'єкт присвоюється кластеру з найближчим центром.

Найпопулярнішим методом кластеризація на основі центроїдів є метод k середніх (k-means). K-means – це некерований алгоритм кластеризації, який розподіляє точки даних у кластери на основі подібності. Алгоритм полягає в простому розбитті даного набору даних на певну кількість кластерів (припускається k кластерів). Основна ідея полягає у визначенні k центрів, по одному для кожного кластера.

Для початку роботи з k-means алгоритмом, спершу потрібно довільно ініціалізувати k точок, які називаються кластерними центроїдами. K-means – це ітераційний алгоритм, який виконується у два етапи:

- призначення кластерів;
- переміщення центроїда.

Призначення кластерів – алгоритм проходить через кожну з точок даних і залежно від того, який центроїд знаходиться ближче, він призначає точки даних одному з кластерів. Близькість базується на Евклідовій відстані.

Переміщення центроїда – k-means переміщає центроїди до середнього значення точок в кластері. Іншими словами, алгоритм обчислює середнє значення всіх точок кластера і переміщує центроїд у це середнє розташування.

Цей ітеративний процес повторюється до тих пір, поки в кластерах не буде змін (або поки не буде дотримано якусь іншу умову зупинки). k вибирається

випадковим чином або шляхом надання конкретних початкових точок від користувача.

Алгоритм розбиває дані на k кластерів, навіть якщо k не є потрібною кількістю кластерів для використання. Тому, використовуючи k-means, користувачам потрібен певний спосіб визначити, чи використовують вони правильну кількість кластерів.

Одним із методів перевірки кількості кластерів є ліктьовий метод. Ідея ліктьового методу полягає у запуску кластеризації k-means на наборі даних для діапазону значень k і для кожного значення k обчислюється сума квадратних помилок SSE.

Потім будується лінійна діаграма SSE для кожного значення k. Якщо лінійна діаграма виглядає як рука, тоді «лікоть» на руці – це значення k, яке є найкращим. Ідея полягає в тому, що ми хочемо невеликий SSE, але SSE має тенденцію до зменшення до 0 при збільшенні k (SSE дорівнює 0, коли k дорівнює кількості точок даних у наборі даних, оскільки тоді кожна точка даних є кластером, і між ним та центром кластера немає помилок) [13].

Крім того, алгоритм k-means може завершити роботу з досить поганими результатами, якщо початкові розташування центроїдів не були оптимальними. Тож процес k-means повторюється кілька разів і приймається результат із найменшими кумулятивними варіаціями між кластерами [14].

Переваги k-means алгоритму – легко зрозуміти та реалізувати, легко масштабується.

Недоліки k-means алгоритму – складно передбачити значення k та початкові позиції центроїдів [15].

Методи кластеризації можуть бути застосовані для розділення користувачів на групи на основі аналітичних профілей для поліпшення роботи рекомендаційної системи.

2.3 Вдосконалення методу побудови рекомендаційної системи на основі колаборативної фільтрації

2.3.1 Загальна модель вдосконаленої рекомендаційної системи

Для онлайн-магазину продажу ігор буде побудована рекомендаційна система, яка буде поєднувати у собі алгоритми колаборативної фільтрації для надання рекомендацій і k-means кластеризацію на основі аналітичних профілів користувачів для поліпшення рекомендацій.

Вхідними даними для кластеризації будуть характеристики користувачів нормовані до шкали від 1 до h, на основі яких будуть створюватися кластери «схожих» користувачів за допомогою алгоритму k-means:

$$\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & \cdots & \cdots & x_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ x_{m1} & \cdots & \cdots & x_{mn} \end{pmatrix} \xrightarrow{k-means} \begin{pmatrix} c_1 \\ c_2 \\ \cdots \\ c_k \end{pmatrix}$$

де x_{ij} – значення j-ої характеристики для і-го користувача,

n – кількість характеристик,

m – кількість користувачів,

c_i – кластер отриманий в результаті роботи алгоритму k-means,

k – кількість кластерів.

Після побудови кластерів «схожих» користувачів, для кожного кластера застосовується алгоритм машинного навчання на основі колаборативної фільтрації (факторизація матриці з градієнтним спуском [16]) для побудови моделей, за допомогою яких будуть визначатися прогнозовані оцінки ігор користувачами:

$$\begin{pmatrix} c_1 \\ c_2 \\ \cdots \\ c_k \end{pmatrix} \xrightarrow{collaboration\ filtering} \begin{pmatrix} m_1 \\ m_2 \\ \cdots \\ m_k \end{pmatrix}$$

де m_i – модель для побудови рекомендації для і-го кластеру.

Для вибору рекомендованих для користувача ігор з вхідного набору ігор буде виконуватися наступне:

– вибір моделі відповідно до кластеру користувача;

– застосування моделі до користувача і кожної гри з вхідного набору – отримання прогнозованої оцінки гри користувачем;

– вибір для рекомендації тих ігор, прогнозована оцінка яких перевищує певний поріг (зазвичай береться 3,5).

$$\begin{pmatrix} g_1 \\ g_2 \\ \dots \\ g_n \end{pmatrix} + u_{c_i} \overset{m_i}{\Rightarrow} \begin{pmatrix} r_1 \\ r_2 \\ \dots \\ r_n \end{pmatrix} \overset{r_j > r_{threshold}}{\Longrightarrow} \begin{pmatrix} g_f \\ \dots \\ \dots \\ g_l \end{pmatrix}, \begin{pmatrix} g_f \\ \dots \\ \dots \\ g_l \end{pmatrix} \in \begin{pmatrix} g_1 \\ g_2 \\ \dots \\ g_n \end{pmatrix}$$

де $\begin{pmatrix} g_1 \\ g_2 \\ \dots \\ g_n \end{pmatrix}$ – вхідний набір ігор для рекомендацій,

n – кількість ігор у вхідному наборі даних,

u_{c_i} – користувач з кластеру c_i, для якого будуються рекомендації,

m_i – рекомендаційна модель для кластеру c_i,

$\begin{pmatrix} r_1 \\ r_2 \\ \dots \\ r_n \end{pmatrix}$ – отримані прогнозовані значення оцінок ігор користувачем,

$r_{threshold}$ – поріг рекомендації (зазвичай береться 3,5),

$\begin{pmatrix} g_f \\ \dots \\ \dots \\ g_l \end{pmatrix}$ – ігри, обрані для рекомендації користувачу.

Кожен новий користувач повинен бути віднесений до існуючих кластерів, або кластери можуть бути перебудовані. Для нових користувачів рекомендації можуть будуватися на основі найпопулярніших ігор в рамках його кластеру.

2.3.2 Моделювання колаборативної фільтрації для онлайн-магазину ігор

Представимо вхідний набір даних для колаборативної фільтрації у вигляді матриці C. Рядки матриці C відповідають за ігри, а стовпці за користувачів. Оцінки можуть приймати значення від 1 до 5, якщо користувач не оцінював гру, то відповідна комірка ініціалізується як 0. Приклад матриці C наведений на рисунку 3.1.

	користувачі					
ігри	5	0	3	...	0	3
	0	4	0	...	2	4
	0	0	0	...	0	0
	⋮	⋮	⋮	⋱	⋮	⋮
	0	0	0	...	0	3
	5	0	3	...	0	0

Рисунок 2.1 – Матриця оцінок ігор користувачами

Нехай наступний набір векторів описує h характеристик кожної гри:

$$g^{(1)}, g^{(2)}, \dots, g^{(\text{кількість ігор})}\ g^i \in R^h$$

Інший набір векторів описує відношення користувача до кожної з характеристик:

$$u^{(1)}, u^{(2)}, \dots, u^{(\text{кількість користувачів})}\ u^i \in R^h$$

Отже, вектори $u^{(j)}$і $g^{(i)}$ мають однакову розмірність, а параметр $u_n^{(j)}$ відображає ставлення користувача j до характеристики $g_n^{(i)}$ гри i. У такому випадку оцінку даної гри цим користувачем можна розрахувати як $(u^{(i)})^T g^i$. Надалі будемо позначати кількість користувачів як n_u, а кількість ігор як n_g. Уявімо, що набір

векторів $u^{(1)}, u^{(2)}, \dots, u^{(n_u)}$ відомий. В такому випадку можна знайти значення вектору $g^{(j)}$ за допомого мінімізації наступного функціоналу:

$$F = \frac{1}{2} \sum_{j:r(i,j)} \left(\left(u^{(j)}\right)^T g^i - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{l=1}^{n} \left(g_l^i\right)^2$$

де $r(i, j) = \begin{cases} 1, \text{якщо } j \text{ користувач оцінив } i \text{ гру} \\ 0, \text{якщо не оцінів} \end{cases}$

$\frac{\lambda}{2} \sum_{l=1}^{n} (g_l^i)^2$ – регулярізація

За допомогою цієї функції розраховується квадратична помилка того, на скільки відрізняються оцінки, які були отримані за допомогою параметрів u і g, в порівнянні зі справжніми значеннями. Отже, після мінімізації функціоналу F, ми отримаємо деякі характеристики $g^{(i)}$ для i гри. Але нам потрібні характеристики для всіх ігор, в цьому випадку функціонал буде мати наступний вигляд:

$$F\left(g^{(1)}, \dots, g^{(n_g)}\right) =$$

$$= \frac{1}{2} \sum_{i=1}^{n_g} \sum_{j:r(i,j)=1} \left(\left(u^{(j)}\right)^T g^i - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_g} \sum_{l=1}^{n} \left(g_l^i\right)^2 \qquad (2.1)$$

Розглянемо зворотну ситуацію коли нам відомі характеристики ігор $g^{(1)}, g^{(2)}, \dots, g^{(n_g)}$. Тоді значення векторів $u^{(1)}, u^{(2)}, \dots, u^{(n_u)}$ можна знайти за допомогою мінімізації наступного функціоналу:

$$F\left(u^{(1)}, \dots, u^{(n_u)}\right) =$$

$$= \frac{1}{2} \sum_{j=1}^{n_g} \sum_{i:r(i,j)=1} \left(\left(u^{(j)}\right)^T g^i - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{l=1}^{n} \left(u_l^j\right)^2 \qquad (2.2)$$

Однак обидва набори векторів $u^{(1)}, u^{(2)}, \dots, u^{(n_u)}$ і $g^{(1)}, g^{(2)}, \dots, g^{(n_g)}$ нам невідомі. Дану проблему можна вирішити двома способами. Перший спосіб полягає в ініціалізації одного з наборів випадковими маленькими значеннями і обчисленні іншого набору векторів. Після цього, обчислити перший набір на основі другого набору векторів – ця операція повторюється декілька разів. Представити це можна так:

$$u \to g \to u \to g \to u \to g \dots$$

Алгоритм, який реалізує цю ідею, називається алгоритмом найменших квадратів, що чергуються (Alternating Least Squares).

Другий спосіб є більш простим. Можна обчислювати обидва набори u і g одночасно. З функціоналів (2.1) і (2.2), можна виявити, що вони обчислюють однакову суму за винятком регуляризації. Перший функціонал обчислює суму помилок для користувачів, які оцінили дану гру. Другий функціонал для кожного користувача обчислює суму квадратних відхилень для тих ігор, які були оцінені даним користувачем. Отже, обидва функціонала обчислюють суму відхилень для всіх пар користувач-гра, які мають значення 1 в таблиці r, тобто даний користувач переглянув і оцінив дану гру. Отримаємо наступний функціонал (функція вартості або функція втрат), який потрібно мінімізувати:

$$\begin{aligned} & F\left(g^{(1)}, \dots, g^{(n_g)}, u^{(1)}, \dots, u^{(n_u)}\right) = \\ & = \frac{1}{2} \sum_{(i,j):r(i,j)=1} \left(\left(u^{(j)}\right)^T g^i - y^{(i,j)}\right)^2 + \\ & + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{l=1}^{n} \left(u_l^j\right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_g} \sum_{l=1}^{n} \left(g_l^i\right)^2 \end{aligned} \qquad (2.3)$$

Вектори u і g ініціалізуються маленькими випадковими значеннями. Після того, як обчислені всі характеристики, можна отримати наступну матрицю прогнозувань:

$$\begin{bmatrix} \left(u^{(1)}\right)^T g^{(1)} & \left(u^{(2)}\right)^T g^{(1)} & \cdots & \left(u^{(n_u)}\right)^T g^{(1)} \\ \left(u^{(1)}\right)^T g^{(2)} & \left(u^{(2)}\right)^T g^{(2)} & \cdots & \left(u^{(n_u)}\right)^T g^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ \left(u^{(1)}\right)^T g^{(n_g)} & \left(u^{(2)}\right)^T g^{(n_g)} & \cdots & \left(u^{(n_u)})\right)^T g^{(n_g)}\right) \end{bmatrix} \quad (2.4)$$

У цій матриці містяться прогнозування оцінок для всіх ігор всіма користувачами. Наприклад, прогноз оцінки користувача j для гри i обчислюється як $(u^{(j)})^T g^{(i)}$.

Приведемо матрицю (2.4) до векторизованого вигляду. Для цього введемо матриці G і U:

$$G = \begin{bmatrix} \cdots & (g^{(1)})^T & \cdots \\ \cdots & (g^{(2)})^T & \cdots \\ & \vdots & \\ \cdots & (g^{(n_g)})^T & \cdots \end{bmatrix}, \quad (2.5)$$

$$U = \begin{bmatrix} \cdots & (u^{(1)})^T & \cdots \\ \cdots & (u^{(2)})^T & \cdots \\ & \vdots & \\ \cdots & (u^{(n_u)})^T & \cdots \end{bmatrix} \quad (2.6)$$

Таким чином, матрицю прогнозування можна подати таким чином:

$$C = G(U)^T \quad (2.7)$$

Вищеописаний метод називається факторизацією матриці низького рангу (Low Rank Matrix Factorization) [16]. Матриця P апроксимується за допомогою представлення її як добутку двох матриць. Метою цієї апроксимації є отримання

передбачень для ігор, які ще не були оцінені користувачем, тобто нулі в початковій матриці повинні бути замінені на прогнозні значення. Досягти цієї мети допомагає той факт, що мінімізуємий функціонал враховує помилки тільки для тих комірок матриці, для яких вже були проставлені оцінки.

2.3.3 Підхід до оптимізації градієнтного спуску

В попередньому розділі було наведені два підходи до обчислення наборів векторів $u^{(1)}, u^{(2)}, \dots, u^{(n_u)}$ і $g^{(1)}, g^{(2)}, \dots, g^{(n_g)}$. Перший – алгоритм ALS, другий – мінімізація функції втрат (3.4). Будемо використовувати другу стратегію, оскільки вона є більш простою в реалізації.

Серед алгоритмів мінімізації одним з найпопулярніших є градієнтний спуск. Суть даного алгоритму полягає в тому, що параметри функції втрат оновлюються в протилежному напрямку градієнта даної функції. Тобто, якщо при поточних параметрах функція зростає (градієнт додатний), то її мінімум знаходиться зліва, і додатний градієнт віднімається. Якщо градієнт від'ємний, то мінімум функції знаходиться праворуч і ми віднімаємо від'ємний градієнт:

$$\omega = \omega - \lambda \times \nabla_{\omega} F(\omega) \tag{2.8}$$

де ω – параметри,

$F(\omega)$ – функція втрат,

λ – коефіцієнт швидкості навчання (learning rate).

З формули (2.8) можна побачити, що параметри оновлюються на величину кратну градієнту. Коефіцієнт швидкості навчання використовується для регуляції швидкості сходження алгоритму та його стабільності. Якщо значення коефіцієнту занадто велике, то алгоритм може розходитися, а якщо занадто маленьке, то алгоритм може сходиться дуже повільно.

Градієнт для векторів $g^{(1)}, g^{(2)}, \dots, g^{(n_g)}$ при функції втрат (2.3) обчислюється наступним чином:

$$\frac{\partial Y}{\partial g_k^{(i)}} = \sum_{j:r(i,j)=1} \left(\left(\theta^{(j)}\right)^T g^i - y^{(i,j)} \right) u_k^{(j)} + \lambda g_k^{(i)} \qquad (2.9)$$

де λ – коефіцієнт регулярізації.

Для векторів $u^{(1)}, u^{(2)}, \dots, u^{(n_u)}$:

$$\frac{\partial Y}{\partial u_k^{(j)}} = \sum_{i:r(i,j)=1} \left(\left(u^{(j)}\right)^T g^i - y^{(i,j)} \right) g_k^{(i)} + \lambda u_k^{(j)} \qquad (2.10)$$

де λ – коефіцієнт регулярізації.

Алгоритм градієнтного спуску складається з трьох кроків:

а) задаються початкові значення для параметрів (для колаборативної фільтрації беруться випадкові маленькі значення) і точність розрахунку ε;

б) за формулою (2.8) обчислюються оновлені значення параметрів;

в) перевіряється умова зупинки алгоритму. Наприклад, алгоритм припиняє роботу, якщо функція втрат змінилася на величину меншу, ніж ε, інакше переходить в пункт б).

Такий алгоритм називається алгоритмом пакетного градієнту (Batch Gradient Descent). Але цей алгоритм не застосовується коли необхідно працювати з великими даними, тому що необхідно робити обчислення для всього набору даних для кожного оновлення. Однак, існує альтернативний алгоритм, який має назву алгоритм стохастичного градієнта (Stochastic Gradient Descent). В цьому алгоритмі оновлення параметрів виконується для кожної пари вхідних даних. Отже, замість формули (2.8) застосовується формула:

$$\omega = \omega - \lambda \times \nabla_\omega F\left(\omega, x^{(i)}; y^{(i)}\right) \qquad (2.11)$$

де ω – параметри,

$(x^{(i)};\ y^{(i)})$ – пара вхідних даних.

Градієнтний спуск сходиться в локальний мінімум і це є його значним недоліком. Потрапляння в локальний мінімум може істотно погіршити якість прогнозів і оскільки функція втрат (2.3) має безліч локальних мінімумів, то цей недолік є суттєвим. Спробувати вирішити цю проблему можна намагаючись запустити алгоритм декілька разів, щоб параметри були ініціалізовані в такому місці, де можливо уникнути локальні мінімуми. Але на це може піти величезна кількість спроб, враховуючи специфіку функції втрат, і не має гарантії, що алгоритм зійшовся саме в глобальний мінімум.

Однак, можливо уникнути проблеми локальних мінімумів, оптимізувавши класичний алгоритм градієнтного спуску. До найбільш ефективних і простих оптимізацій відноситься оптимізація, яка носить назву «Імпульс» (Momentum)

Оптимізація типу імпульс потребує заміну формули (2.11) на формули:

$$v_t = 0.9 \times v_{t-1} - \lambda \times \nabla_\omega F\left(\omega, x^{(i)}; y^{(i)}\right) \quad (2.12)$$

$$\omega = \omega - v_t \quad (2.13)$$

В векторі v накопичується імпульс для кожного параметра. Отже, для мінімізації функції витрат (2.3) пропонується використовувати описаний оптимізований алгоритм градієнтного спуску.

2.3.4 Моделювання кластеризації для розбиття користувачів на групи

Рекомендаційна система, що розробляється в даній роботі, для підвищення ефективності та поліпшення якості рекомендацій використовує кластеризацію користувачів. Характеристики користувача повинні бути виражені в чисельних

показниках і представлені масивом чисел від 1 до n після нормування. Для кластеризації даних подібного типу найчастіше використовують алгоритм k-means.

Даний алгоритм розділяє безліч вхідних об'єктів на задане число кластерів. Алгоритм починає свою роботу з ініціалізації центрів кластерів (центроїдів):

$$\mu_1, \mu_2, \dots, \mu_k \in R^n.$$

де k – кількість кластерів,

n – розмірність кожного об'єкта.

Кожен елемент множини відноситься до кластеру з найближчим центром. Наступні етапи повторюються до тих пір, поки алгоритм не зійдеться.

Далі заповнюється вектор c. Цей процес записується наступним чином – $for\ i = 1\ to\ l$:

$$c^{(i)} = \left\{p: \|x^{(i)} - \mu_p\|^2 \leq \|x^{(i)} - \mu_j\|^2 \forall j, 1 \leq j \leq K\right\} \quad (2.14)$$

де $x^{(i)}$ – об'єкт з набору вхідних даних

c – вектор, який містить відповідність між об'єктом $x^{(i)}$ і кластером $c^{(i)}$, до якого він належить

l – кількість об'єктів множини вхідних даних

З формули (2.14) виходить, що кожен об'єкт з набору вхідних даних відноситься до кластеру з найближчим центром.

На наступному етапі оновлюється положення центроїдів – $for\ k = 1\ to\ K$:

$$\mu_k = \frac{1}{S_k} \sum_{x^{(j)}: c^{(j)} = k} x^{(j)} \quad (2.15)$$

З формули (2.15) виходить, що значення центру мас безлічі вхідних об'єктів присвоюються кожному центру кластера k, до якого належать ці об'єкти. Якщо не

відбулося зміни положення центрів кластерів після поточної ітерації, то алгоритм вважається завершеним.

Однак у k-means алгоритму є ряд недоліків:

– число кластерів визначається заздалегідь;

– не гарантується досягнення глобального мінімуму;

– може відбуватися різне розбиття для однакових вхідних даних при різних початкових положеннях центрів кластерів.

Розглянемо підходи до мінімізації даних недоліків. Введемо функцію втрат для оцінювання якості розбиття:

$$F\left(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K\right) = \frac{1}{m}\sum_{i=1}^{m} \left\|x^i - \mu_{c^{(i)}}\right\|^2 \qquad (2.16)$$

Наведений функціонал обчислює суми квадратичних відстаней від центрів кластерів до об'єктів відповідних цим кластерам. Метою алгоритму оптимізації кластеризації є мінімізація даного функціоналу.

Розглянемо перший недолік – число кластерів визначається заздалегідь. Цей недолік можна мінімізувати, використавши метод ліктя. За цим методом розробник самостійно обирає кількість кластерів і оцінює зменшення функції втрат, при збільшенні кількості кластерів.

Другий недолік полягає у відсутності гарантії досягнення глобального мінімуму. Даний недолік можливо мінімізувати за допомогою множинної ініціалізації. Алгоритм k-means ініціалізується різними наборами центроїдів, для кожного набору оцінюється значення функції витрат – обирається той варіант, при якому функція втрат має найменше значення.

Таким чином, був запропонований і описаний вдосконалений метод побудови рекомендаційної системи (див. рис 2.2).

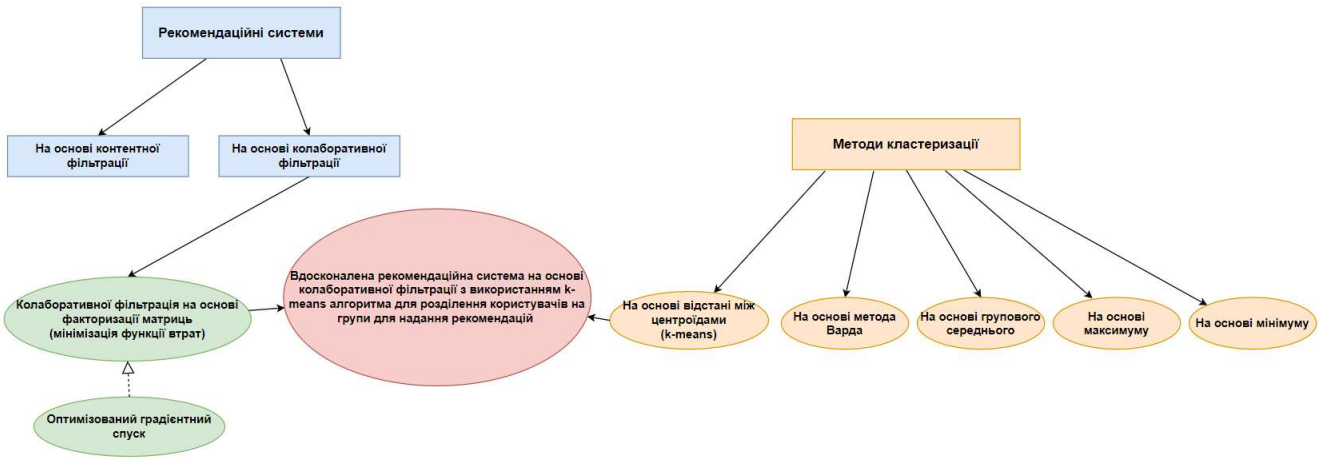

Рисунок 2.2 – Вдосконалений метод побудови рекомендаційної системи

Запропонований метод використовує k-means кластеризацію для розбиття користувачів на групи і колаборативну фільтрацію на основі факторизації матриці з використанням оптимізованого градієнтного спуску для надання рекомендацій.

2.4 Планування експериментального дослідження

Критично важливою задачею при розробці рекомендаційної системи є задача оцінки якості рекомендацій. Оцінити ефективність рекомендаційної системи і параметри якості рекомендацій можна за допомогою багатьох метрик, розглянемо основну.

Однією з найпопулярніших метрик, що вимірюють якість рекомендацій, є метрика середньоквадратичної помилки (Root Mean Square Error (RMSE)). Вона розраховується за формулою:

$$RMSE = \sqrt{\frac{1}{\mathrm{T}} \sum_{(i,j)\in \mathrm{T}} (y'_{ij} - y_{ij})^2}$$

де T – загальна кількість оцінок з тестового набору,

(i, j) – пара гра-користувач з тестового набору,

y' – прогнозована оцінка,

y – дійсна оцінка користувача.

Метрику RMSE не є ідеальною мірою якості рекомендацій і має ряд недоліків, наприклад, ті користувачі, які мають більш широкий розкид оцінок, впливатимуть більше. Крім того, можна мати ідеальну метрику RMSE при дуже поганому ранжуванні. Але дана метрика має широке застосування в галузі рекомендаційних систем, тому є найбільш переважною для використання.

2.5 Аналіз та UML-моделювання предметної області продажу ігор

За останні роки світ відеоігор значно змінився. Його диверсифікація різко збільшила кількість користувачів, які беруть участь в онлайн спільнотах цієї розважальної сфери, а отже, збільшилась і кількість доступних ігор та їх типів. Цей контекст перевантаження інформацією лежить в основі розробки рекомендаційних систем, які могли б використовувати інформацію, яку збирають платформи продажу ігор і, отже, слідувати тенденції щорічного виходу нових ігор.

На сьогоднішній день електронні ігри зайняли стабільне місце на ринку як ігрової індустрії взагалі, так і в житті багатьох людей. Люди покупають та грають в електронні ігри по величезній кількості причин, намагаючись заповнити вільний час або, навпаки, втискуючи їх в свій щільний графік, шукаючи можливості поспілкуватися з людьми і знайти нових друзів або, навпаки, приховати свої риси і уявити себе тим, ким гравець в реальності не є. Одні люди постійно витрачають гроші, купуючи нове спорядження, здібності, особливі переваги або що-небудь ще для своїх персонажів в іграх, інші – зробили заробіток за допомогою електронних ігор своєю справою, професійно «прокачувати» героїв і потім продаючи їх. Все це в сумі робить світ комп’ютерних ігор окремим і ізольованим, де відбувається

спілкування і взаємодія з своїм правилами, які часто відрізняються від правил, прийнятих в «реальному» світі взаємодій віч-на-віч.

Існує безліч класифікацій електронних ігор. В основу різних класифікацій покладено різні принципи.

За жанром комп'ютерні ігри можна розділити на:

– рольові ігри – це ігри, в яких гравець керує вигаданим персонажем (або персонажами), який виконує завдання в уявному світі;

– стратегії – ігри, які зосереджуються на вмілому мисленні та плануванні для досягнення перемоги;

– екшен/шутер від першої особи – ігри, які базуються на фізичних викликах, включаючи координацію типу «руки-очі» та перевірку реакції;

– симулятори – ігри, які намагаються скопіювати різні дії з реального життя у формі гри для різних цілей, таких як навчання, аналіз або прогнозування;

– карточні ігри-стратегії;

– квести – ігри, в яких гравець бере на себе роль головного героя в інтерактивній історії, яка керується дослідженнями або вирішенням головоломок. [7].

За платформами ігри поділяються на:

– ігри для персонального комп'ютера (ПК). ПК є одним з найдавніх платформ для запуску відеоігор. Гра на ПК має багато переваг, включаючи якісні візуальні ефекти та більшу універсальність;

– консольні ігри. Ігрові консолі дещо простіші у використанні у порівнянні з ПК. Також, в консольні ігри зазвичай грають за допомогою контролерів;

– мобільні ігри. Остання тенденція в іграх – це посилений розвиток ігор для мобільних телефонів та планшетів. Сьогодні більшість людей мають смартфон з ігровими можливостями;

– браузерні ігри – це ігри, в які можна грати просто використовуючи веб-браузер. На сьогоднішній день інтернет може бути найдоступнішим способом для людей будь-якого віку грати в ігри.

Крім того, існує безліч видавців ігор – як нових, які тільки починають привертати увагу гравців (Amazon Game Studios, 11 bit studios), так і старих, які вже зарекомендували себе в ігровій спільноті (Electronic Arts, Bethesda Sowtworks). Зазвичай видавці випускають ігри різноманітних жанрів і на різні платформи.

Дана класифікація дозволяє встановити сегменти споживачів комп'ютерних ігор. Так, наприклад, споживачами симуляторів є любителі історичної та військової техніки (в основному чоловіки); екшени і шутери, як правило, обирають підлітки чоловічої статі, рольові ігри – люди, зацікавлені в міжособистісної комунікації та взаємодії з іншими гравцями; квести – любителі вирішувати різні головоломки; стратегії (економічні, військові та адміністративні) охоплюють широкий сегмент споживачів. В ПК ігри грають люди, які цінують якість візуальних ефектів та продуктивність. В консольні ігри грають ті, кому важливий баланс між якістю і простотою використання. В мобільні ігри грають ті, хто полюбляє грати, але у кого не вистачає часу або засобів на «великі» ігри. В браузерні ігри грають ті, кому потрібна простота і доступність.

Масова кастомізація стає як ніколи популярною. Сучасні рекомендаційні системи – на базі колаборативної фільтрації та на базі контентної фільтрації – використовують різні джерела інформації для побудови рекомендацій. Контентна фільтрація дає рекомендації на основі уподобань користувачів щодо властивостей продукту. Колаборативна фільтрація імітує рекомендації від користувача до користувача. Вона передбачає вподобання користувачів як лінійну зважену комбінацію уподобань інших користувачів.

Дані, що використовуються для реалізації рекомендаційних систем, можуть бути явними, наприклад, відгуки чи оцінки, або неявними, такими як поведінка та події, такі як історія замовлень, історія пошуку, кліки тощо. Неявні дані важче обробити, оскільки важко визначити, яка інформація корисна, а яка ні. Але неявні дані легше отримати порівняно з явними даними, оскільки користувачеві не потрібно робити нічого більше, ніж користуватися веб-сайтом або додатком, як зазвичай [17].

Персоналізація в сфері продажу ігор набула великого значення завдяки великій кількості ігор та їх властивостей, доступних в інтернеті. Однією з цілей персоналізованих програм є надання відповідної інформації, яка відповідає особистим інтересам користувача, та забезпечення ефективного доступу до інформації. Таким чином, створення аналітичних профілів користувачів для довгострокових та короткострокових інтересів є вирішальним для персоналізації. Профіль користувача – це структурована інформація, яка містить уподобання та контекст користувача. Під контекстом користувача маються на увазі його властивості – такі як вік, стать, тощо. Уподобаннями користувачів в сфері продажу ігор можуть бути – улюблений жанр, улюблена платформа, улюблений видавець.

Якість рекомендацій колаборативної фільтрації можна поліпшити, працюючи з даними «схожих» користувачів. «Схожість» користувачів може визначатися на основі даних, які користувач залишив про себе (аналітичний профіль користувача) [18].

Також для колаборативної фільтрації може бути проблематичним надання рекомендацій новим користувачам, які ще не оцінювали жодної гри. В цьому випадку «схожість» користувачів може допомогти при виборі рекомендацій – можна рекомендувати найбільш популярні серед «схожих» користувачів ігри.

На основі проведеного аналізу предметної області продажу ігор з використанням рекомендаційних систем, виконаємо UML-моделювання онлайн-магазину продажу ігор з рекомендаційною системою.

Перша і одна з найбільш важливих діаграм – діаграма варіантів використання або UseCase (див. рис. 2.2 та 2.3). UseCase-діаграма – діаграма поведінки, яка відображає відносини між акторами і прецедентами (варіантами використання).

На рисунках 2.2 та 2.3 зображено п’ять акторів: гість, користувач, модератор, менеджер, адміністратор.

Гість – це відвідувач онлайн-системи, який не має власного акаунту. Він може лише переглядати каталог ігор (пошук, фільтрація, сортування), інформацію про ігри та їх коментарі і оцінки. Крім того, гість може залишити коментар про гру. Після того як гість переглянув та обрав гру для покупки, для того, щоб зробити

замовлення гість повинен стати користувачем системи зареєструвавшись в ній. Також, щоб оцінювати ігри та отримувати рекомендації, гостю потрібно зареєструватися.

Користувач, у додаток до можливостей гостя, може замовляти ігри і оплачувати їх. Користувач також може додавати і видаляти ігри до кошику в процесі пошуку ігор для купівлв. Користувач може оплатити своє замовлення одним з наступним способів – PayPal, VISA або банк. Крім того, користувач може оцінювати гру, перглядати історію своїх оцінок, заповняти профіль користувача (вік, стать, улюблені жанри) та отримувати персоналізовані рекомендації ігор.

Слід відмітити, що з впровадженням системи оцінок, користувачам та гостям стане можливим перегляд загальної оцінки гри, сортування і фільтрація ігор за оцінками.

Адміністратором онлайн-системи назначає модераторів та менеджерів.

Актор модератор має наступні можливості: модерування коментарів, які були написані гостями та користувачами, та блокування користувачів.

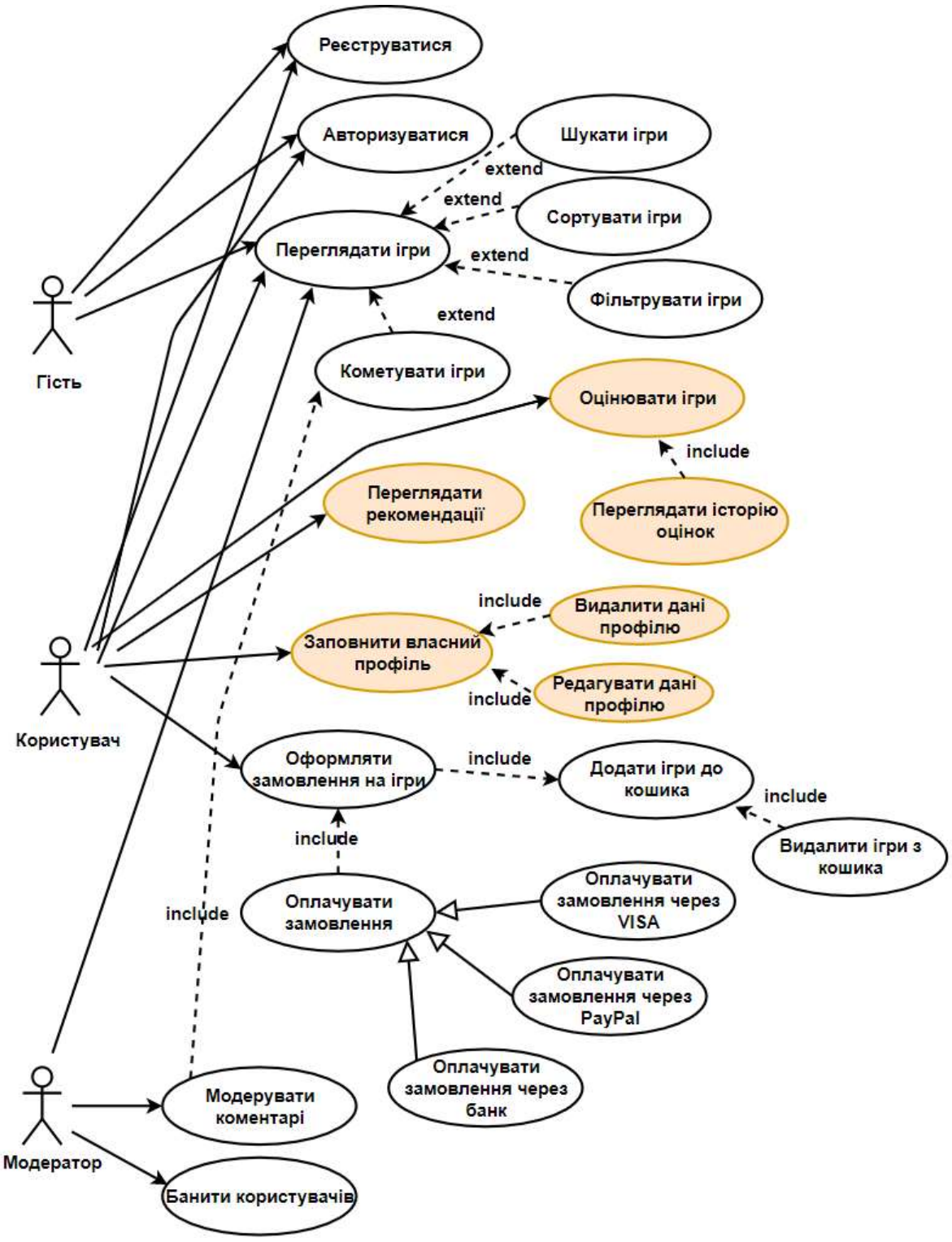

Рисунок 2.2 – UseCase-діаграма (Гість-Користувач-Модератор)

Актор менеджер керує основними сутностями системи (ігри, жанри видавці, платформи) – додає, змінює, видаляє інформацію. Також, менеджер керує статусами замовлень користувачів.

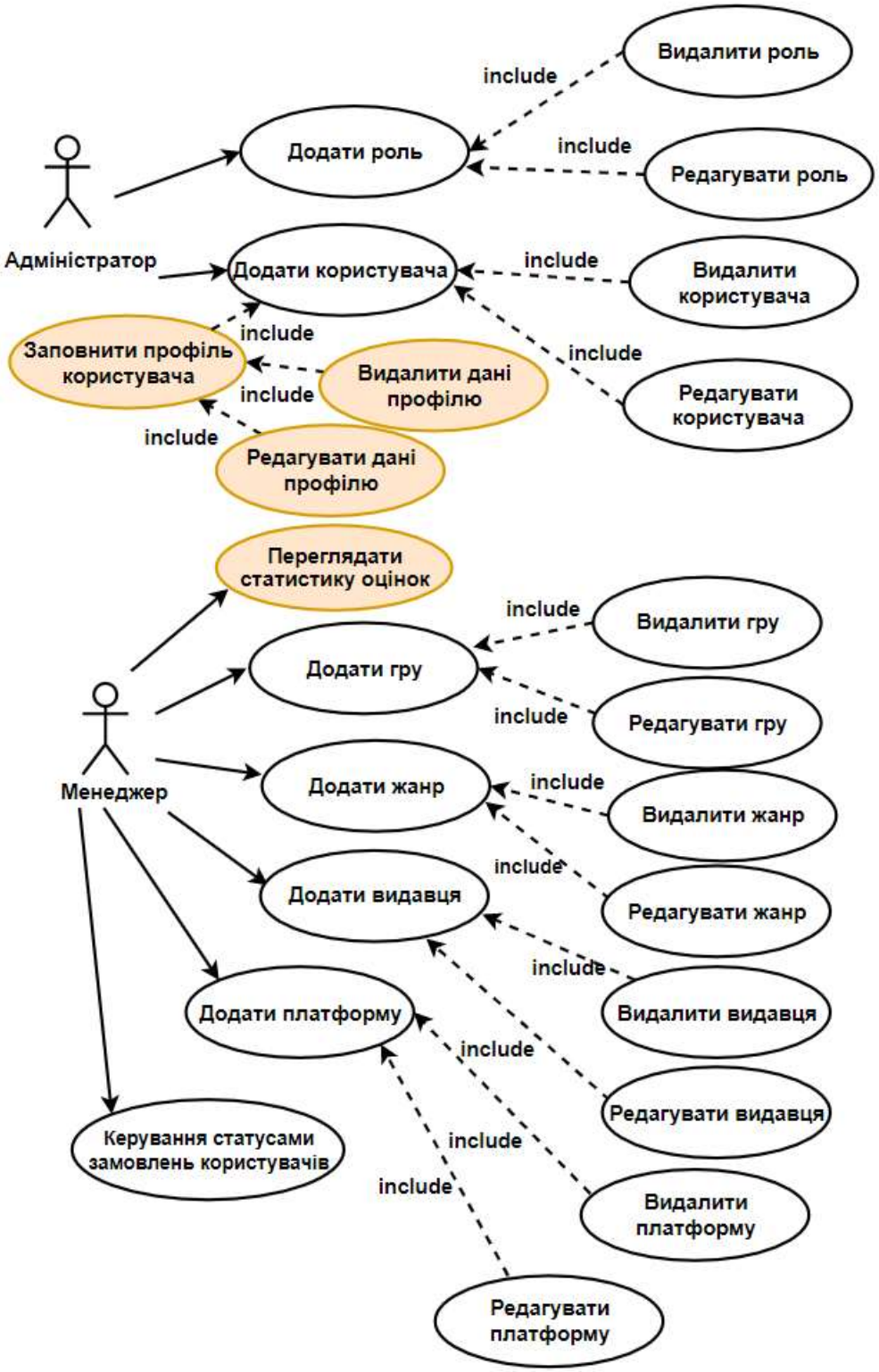

Рисунок 2.3 – UseCase-діаграма (Адміністратор-Менеджер)

Власник онлайн-системи назначає адміністратора.

Актор адміністратор керує сутностями користувачів та ролей – додає, редагує та видаляє користувачів та ролі, додавання, редагування та видалення даних профілів користувачів[7].

2.6 Проектування бази даних

Для коректної роботи рекомендаційної системи на основі онлайн-сервісу з продажу ігор необхідно вводити, оброблювати, зберігати та надавати велику кількість різноманітних даних, таких як: дані про ігри, про користувачів, про замовлення, про оцінки ігор користувачами.

Щоб мати змогу зберігати ці дані, буде використовуватися база даних, яка є невід'ємною частиною кожного сучасного додатку. База даних (БД) може зберігати велику кількість систематизованої інформації, і швидко надавати її після введення запиту користувача. Проектування бази даних є дуже важливим етапом при створенні рекомендаційної системи на основі онлайн-системи з продажу ігор.

Проектування бази даних являє собою складний трудомісткий процес відображення предметної області у внутрішню модель даних. Сам процес проектування бази даних полягає в створенні схеми бази даних і визначення необхідних обмежень цілісності інформації.

В якості моделі зберігання даних була обрана реляційна модель. Реляційна база даних – це тип бази даних, який використовує структуру, що дозволяє ідентифікувати та отримувати доступ до даних по відношенню до іншого фрагмента даних у базі даних. Часто дані в реляційній базі даних упорядковуються у таблиці. Таблиці можуть містити сотні, тисячі, іноді навіть мільйони рядків даних. Ці рядки часто називають записами. Таблиці також можуть мати багато стовпців даних. Стовпці маркуються описовою назвою і мають певний тип даних.

Реляційні бази даних та системи, що використовуються для управління ними, є дуже стабільними та мають ряд властивостей, що роблять їх дуже популярними.

Однією з таких властивостей є вбудована відповідність ACID, яка чудово підходить для забезпечення правильного та надійного завершення фінансових операцій. Для роботи та взаємодії з реляційними базами даних використовується добре визначена мова – мова структурованих запитів (SQL).

Однак ця високоструктурована природа реляційних баз даних також є їх недоліком, оскільки вони жорсткі до визначеного характеру стовпців у своїх таблицях. Отже, якщо є новий набір даних, який не відповідає параметрам таблиць, його буде важко включити до бази даних. Крім того, такі бази даних важко масштабувати, що вимагає великих інвестицій в інфраструктуру щоразу, коли бази даних збільшуються [7].

Для рекомендаційної системи на основі колаборативної фільтрації необхідно зберігати дані оцінок ігор користувачами. Оскільки кожна гра може мати багато оцінок від різних користувачів, а кожен користувач може залишити оцінки до різних ігор, необхідно виділити окрему сутність оцінки гри користувачем, яка буде містити ідентифікатор користувача, ідентифікатор гри, оцінку (див. рис. 2.4).

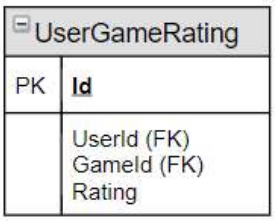

Рисунок 2.4 – Сутність «Оцінка гри користувачем»

Для поліпшення рекомендацій застосовується кластеризація на основі характеристик користувача (аналітичний профіль), тому сутність користувача повинна бути розширена додатковими властивостями, такими як стать, вік, улюблені жанри (див. рис. 2.5).

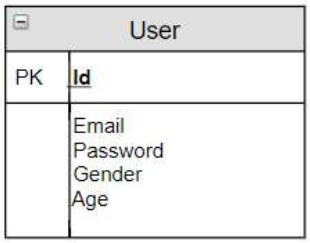

Рисунок 2.5 – Розширена сутність «Користувач»

Користувач може мати декілька улюблених жанрів. Декілька користувачів можуть мати одні і ті самі улюблені жанри.

Для коректної кластеризації користувачів з використанням такої властивості як улюблений жанр, сутність жанру повинна бути розширена власними характеристиками, такими як – рівні навичок, екшену, стратегічності, сюжету, зрілості, насилля та освіти. Розширена сутність жанру наведена на рисунку 2.6.

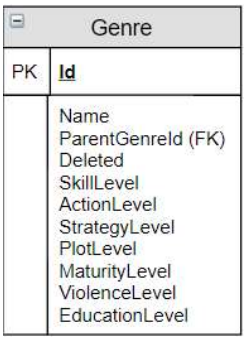

Рисунок 2.6 – Розширена сутність «Жанр»

Доповнена для реалізації рекомендаційної системи схема бази даних наведена на рисунку 2.7.

Таким чином, ми спроектували та побудували реляційну модель бази даних для рекомендаційної системи веб-додатку продажу ігор.

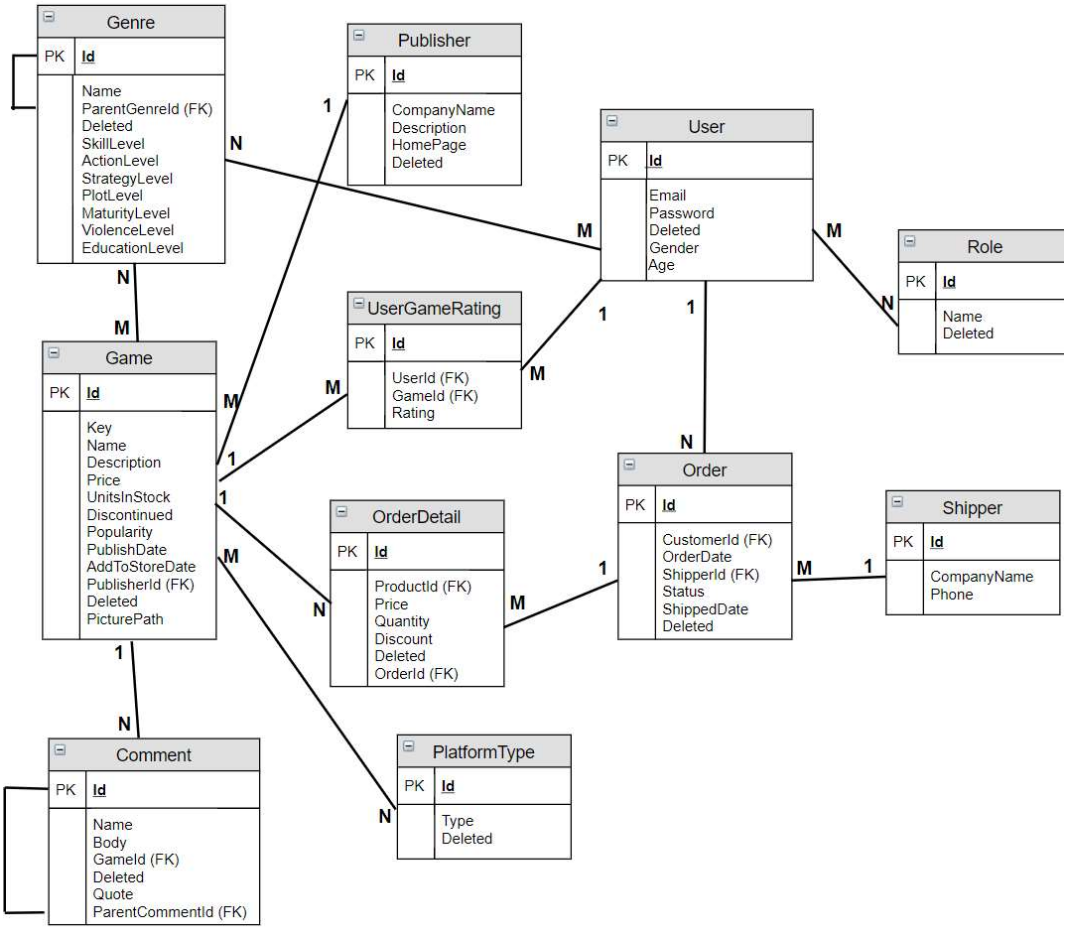

Рисунок 2.7 – Схема бази даних з урахуванням рекомендаційної системи

2.7 Розробка архітектури системи

В ході планування та моделювання рекомендаційної системи для онлайн-магазину електронних ігор було визначено, що система повинна складатися з наступних компонентів:

– сервер бази даних, на якому зберігаються дані сутностей, необхідних для рекомендаційної системи (дані про користувачів, ігри та оцінки), на основі реляційної моделі (SQL Server) та який отримує і обробляє зовнішні запити;

– веб-сервер, який відповідає за взаємодію з сервером бази даних, обробку отриманих даних, бізнес-логіку, ін’єкцію залежностей, тренування та

побудову моделей необхідних рекомендаційній системі та за підготовку і відображення користувачу веб-сторінок онлайн-системи;

– клієнт користувача, в якому за допомогою браузера користувач може взаємодіяти з рекомендаційною системою на базі онлайн-магазину електронних ігор;

Як вже було зазначено у попередніх розділах, веб-сервер був побудований на основі N-рівневої або багатошарової архітектури.

До таких основних шарів, які були запропоновані в ході бакалаврської роботи, як шар доступу до даних (GameStore.DAL), шар бізнес-логіки (GameStore.BLL), шар уявлень (GameStore.WEB), шар ін’єкції залежностей (GameStore.DI), був доданий новий шар – шар машинного навчання (GameStore.ML), який відповідає за надання рекомендацій користувачам і за тренування та тестування моделей необхідних для надання цих рекомендацій. Побудуємо діаграму розгортання для опису загальної архітектури системи.

Діаграма розгортання – це діаграма, яка показує конфігурацію вузлів системи під час виконання обробки та компоненти, які знаходяться на цих вузлах. Діаграми розгортання – це свого роду структурна діаграма, що використовується при моделюванні фізичних аспектів об'єктно-орієнтованої системи. Вони часто використовуються для статичного відображення розгортання системи (топологія апаратного забезпечення).

Призначення діаграми розгортання:

– показати структуру системи під час виконання;

– показати обладнання, яке буде використано для реалізації системи, та зв’язки між різними елементами обладнання;

– змоделювати фізичні елементи апаратного забезпечення та шляхи зв'язку між ними [19].

Діаграма розгортання системи онлайн-магазину електронних ігор наведена на рисунку 2.8.

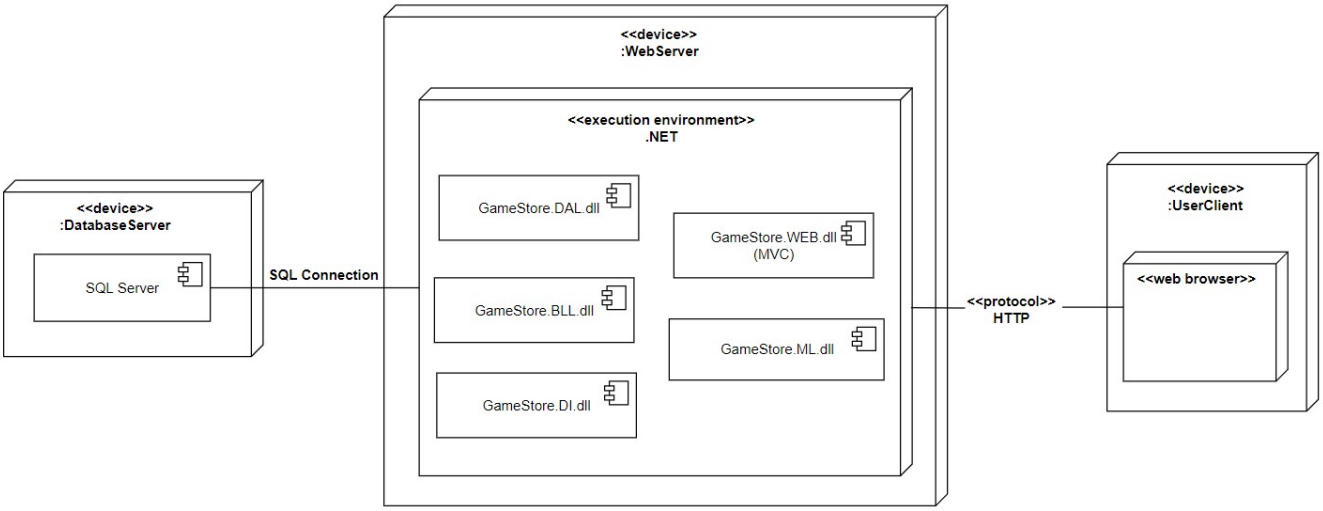

Рисунок 2.8 – Діаграма розгортання

Побудуємо діаграму компонентів системи. Призначення діаграми компонентів – показати взаємозв'язок між різними компонентами в системі.

Компонент доступу до даних складається з інтерфейсів, репозиторіїв, моделей сутностей системи і робить запити до джерела даних для взаємодії з сутностями системи (отримання інформації, додавання, оновлення, видалення даних).

Компонент бізнес-логіки складається з інтерфейсів, сервісів, транспортних моделей та звертається до компоненту доступу до даних для того, щоб виконати операції над даними, які необхідні для забезпечення бізнес процесів системи.

Компонент машинного навчання складається з модулю кластеризації, модулю колаборативної фільтрації і користується компонентом бізнес-логіки для отримання даних необхідних для тренування моделей рекомендаційної системи.

Компонент уявлень складається з контролерів, уявлень, моделей уявлень та використовує компонент бізнес-логіки та компонент машинного навчання для обробки безпосередніх запитів користувачів, реалізації бізнес-логіки та надання користувачам рекомендацій.

Для визначення конкретних реалізацій класів для використовуваних інтерфейсів, усі вищеописані компоненти використовують компонент ін’єкції залежностей, який складається з модулів зіставлення інтерфейс-клас.

Діаграма компонентів наведена на рисунку 2.9.

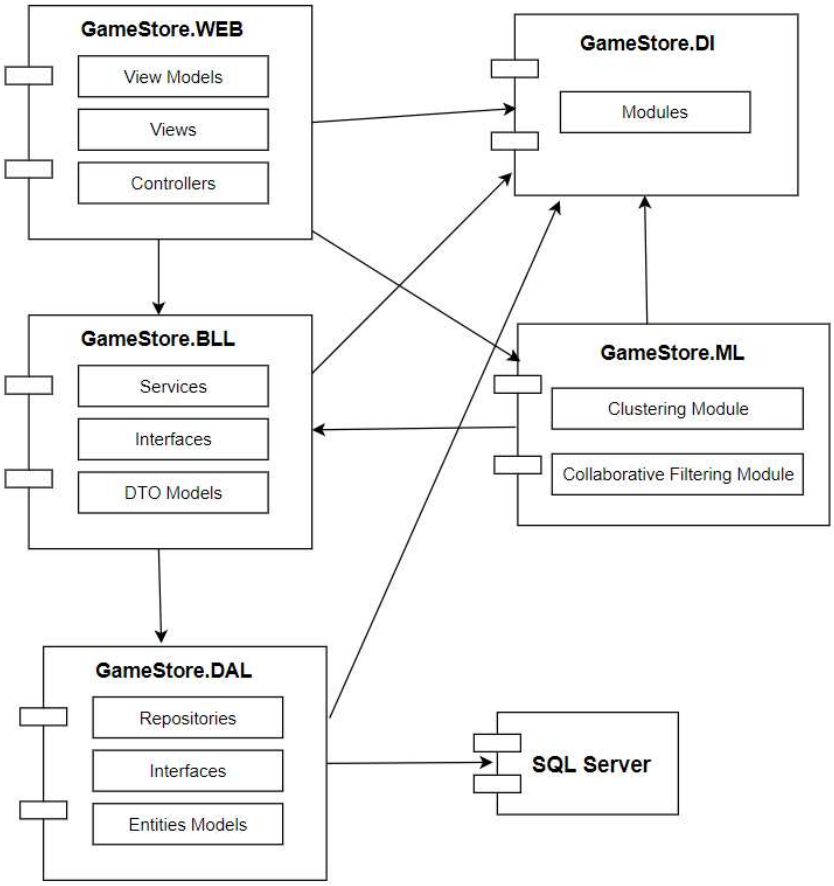

Рисунок 2.9 – Діаграма компонентів

Таким чином, ми спроектували архітектуру онлайн-системи продажу ігор з рекомендаційною системою та побудували діаграми розгортання і компонентів.

2.8 Опис алгоритму надання рекомендацій

Перш ніж переходити до реалізації рекомендаційної системи для онлайн-магазину електронних ігор, необхідно продумати та спроектувати алгоритм, який буде використовуватися для надання рекомендацій користувачам. Алгоритм надання рекомендацій включає в себе кластеризацію користувачів за аналітичним профілем та колаборативну фільтрацію для кластеру користувачів для знаходження

рекомендацій. Зобразимо алгоритм за допомогою діаграми послідовності (див. рис. 2.10). Діаграма послідовності – це діаграма взаємодії, що детально описує спосіб виконання операцій. Діаграма послідовності фокусується на часі і наочно показує порядок взаємодії, використовуючи вертикальну вісь діаграми для представлення час і показуючи які операції коли виконуються.

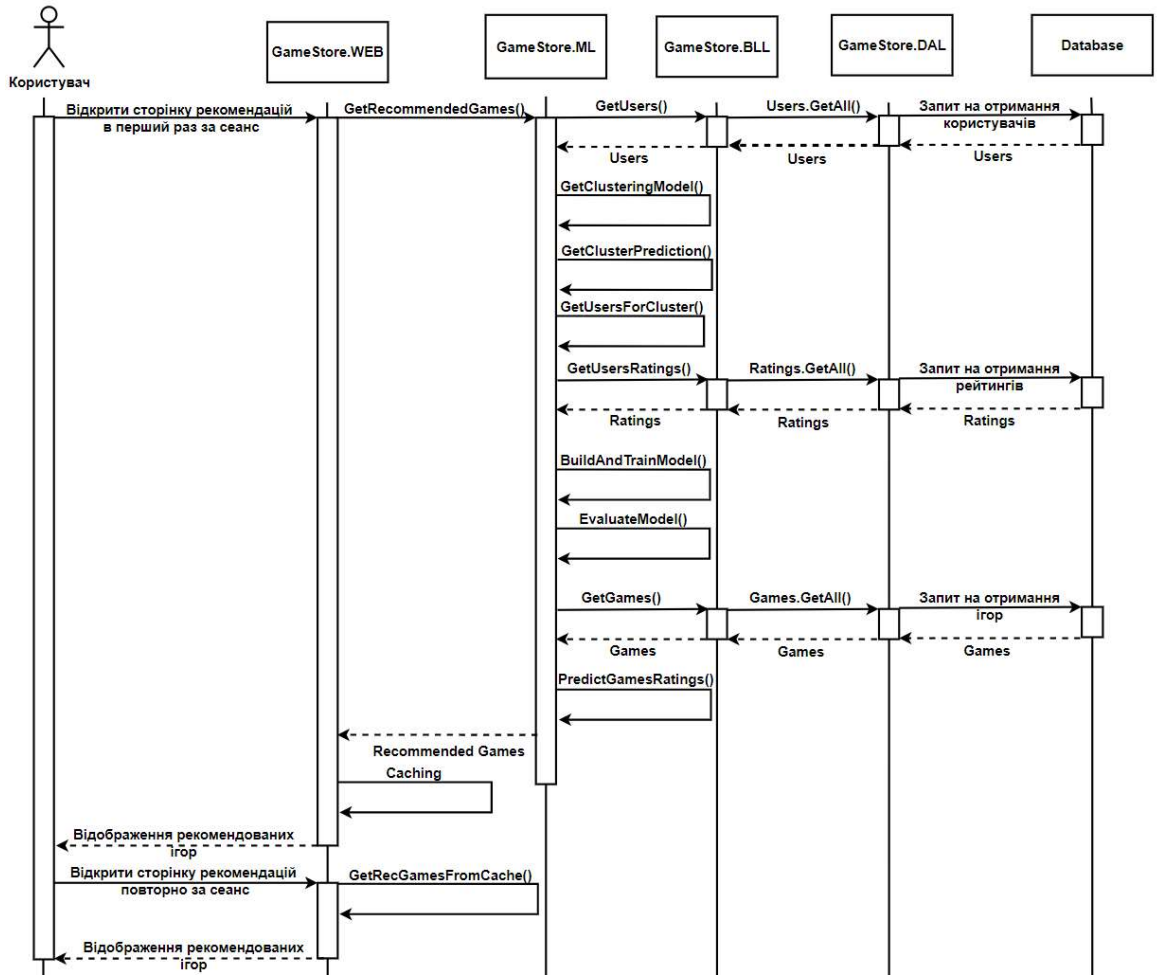

Рисунок 2.10 – Діаграма послідовності

Як можна побачити на діаграмі послідовності, основними об’єктами системи є користувач, шари додатку (шари уявлення, машинного навчання, бізнес-логіки та доступу до даних) та база даних.

Алгоритм починається з того, що користувач відкриває сторінку з рекомендаціями. При відкриті сторінки з рекомендаціями робиться виклик до екшен методу контролера на нарі уявлення, який, в свою чергу викликає метод GetRecommendedGames() шару машинного навчання для отримання рекомендацій

для поточного користувача. В методі GetRecommendedGames() викликається метод GetUser() шару бізнес-логіки для отримання усіх користувачів системи з їх аналітичними профілями. Метод GetUser(), у свою чергу, викликає відповідний метод шару доступу до даних, який робить запит у базу даних.

Після того як інформація про користувачів отримана, у методі GetRecommendedGames() викликається спочатку метод GetClusteringModel() для отримання моделі кластеризації на основі отриманих аналітичних профілів користувачів системи. А після цього викликається метод GetClusterPrediction(), який на основі отриманої моделі кластеризації знаходить кластер поточного користувача. Далі викликається метод GetUsersForCluster() для отримання усіх користувачів, які відносяться до того ж самого кластеру, що і поточний користувач.

Після того, як знайдені користувачі з кластеру поточно користувача, викликається метод GetUsersRatings() шару бізнес-логіки для отримання усіх оцінок знайдених користувачів. Метод GetUsersRatings (), у свою чергу, викликає відповідний метод шару доступу до даних, який робить запит у базу даних.

Далі, на основі отриманих оцінок будується, тренується та оцінюється модель рекомендаційної системи на основі колаборативної фільтрації (методи BuildAndTrainModel() і EvaluateModel() відповідно). Після цього викликається метод GetGames() шару бізнес-логіки для отримання усіх ігор системи, які не оцінював поточний користувач. Метод GetGames(), у свою чергу, викликає відповідний метод шару доступу до даних, який робить запит у базу даних.

Далі, в методі GetRecommendedGames() на основі отриманої рекомендаційної моделі робляться передбачення оцінок для отриманих ігор. На основі передбачень формується список рекомендованих ігор поточному користувачу. Цей список повертається з шару машинного навчання до шару уявлень, кеширується і відображається у браузері користувача на сторінці рекомендацій.

Слід відмітити, що описаний алгоритм виконується тільки при першому відкриванні сторінки рекомендацій користувачем в рамка браузерного сеансу. При повторних відкриттях сторінки рекомендацій в рамка браузерного сеансу рекомендації для відображення будуть витягатися з кешу.

3 ОПИС ЕКСПЕРИМЕНТАЛЬНОГО ДОСЛІДЖЕННЯ

3.1 Підготовка вхідних даних для дослідження

Як вже було зазначено в попередніх розділах для створення рекомендаційної моделі необхідні наступні вхідні дані:

– інформація про аналітичні профілі користувачів для кластеризації – вік, стать, інформація про улюблені жанри (рівні навичок, екшену, стратегічності, сюжету, зрілості, насилля та освіти);

– інформація про оцінки ігор користувачами для колаборативної фільтрації.

В якості базового набору даних були використані дані про ігри та користувачів отримані за допомогою API найкрупнішої онлайн-системи продажу ігор – Steam. Дані були отримані у вигляді масиву ігор та користувачів в форматі JSON. Приклад інформації про гру, отриманої з Steam API наведено на рисунку 3.1.


```
{
   "publisher":"Kotoshiro",
   "genres":["Action", "Casual", "Indie", "Simulation"],
   "app_name":"Lost Summoner Kitty",
   "title":"Lost Summoner Kitty",
   "url":"http://store.steampowered.com/app/761140/Lost_Summoner_Kitty/",
   "release_date":"2018-01-04",
   "discount_price":4.49,
   "reviews_url":"http://steamcommunity.com/app/761140/reviews/?browsefilter=mostrecent&p=1",
   "price":4.99,
   "early_access":false,
   "id":"761140",
   "developer":"Kotoshiro"
}
```


Рисунок 3.1 – Приклад інформації про гру з Steam API

Як можна побачити Steam API надає таку корисну інформацію про гру як ідентифікатор гри та жанри, до яких ця гра належить. Однак, Steam API не надає жодної додаткової інформації про жанри.

На рисунку 3.2 наведено приклад інформації про користувача, отриманої з Steam API.

```
{
    "user_id":"Riot-Punch",
    "items_count":2,
    "steam_id":"76561197963445855",
    "user_url":"http://steamcommunity.com/id/Riot-Punch",
    "items":[
        {
            "item_id":"10",
            "item_name":"Counter-Strike",
            "playtime_forever":1005,
            "playtime_2weeks":102
        },
        {
            "item_id":"20",
            "item_name":"Team Fortress Classic",
            "playtime_forever":603,
            "playtime_2weeks":10
        }
    ]
}
```


Рисунок 3.2 – Приклад інформації про користувача з Steam API

Як можна побачити Steam API надає таку корисну інформацію про користувача як ідентифікатор користувача та дані про ігри в які гра користувач (ідентифікатор гри та кількість зіграних годин). Однак, Steam API не надає інформації про оцінки ігор користувачами та про характеристики користувачів (стать, вік, улюблені жанри).

Оскільки Steam API не надає усіх необхідних даних, необхідних для побудови рекомендаційної системи, був написаний окремий додаток для визначення даних, яких бракує і для приведення вхідних даних до виду придатного для використання для побудови і тренування моделі.

Був створений консольний додаток на мові програмування C#, який зчитує дані отриманих з Steam API ігор та користувачів, створює відповідні C# об’єкти за допомогою бібліотеки Json.NET, визначає і додає бракуючу інформацію і зберігає об’єкти у форматі CSV для подальшого загруження даних у базу даних і використання їх для побудови, тренування і тестування рекомендаційної моделі.

Розглянемо основні етапи роботи даного додатку. Блок-схема алгоритму, який використовується у додатку, неведена на рисунку 3.3.

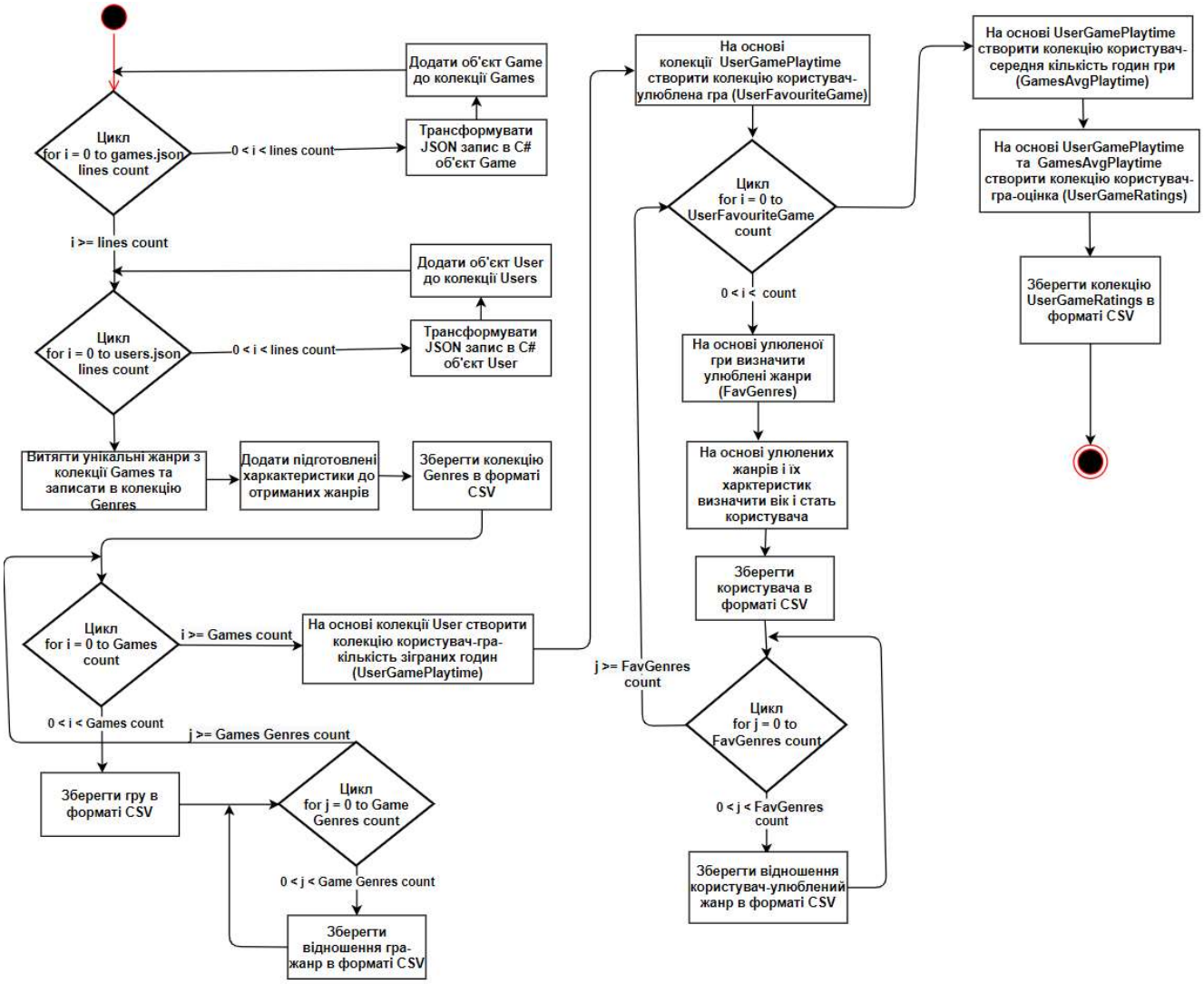

Рисунок 3.3 – Блока-схема алгоритму підготовки вхідних даних

Отже, як зазначено на блок-схемі, алгоритм підготовки даних починається з зчитування даних ігор та користувачів з JSON файлів, отриманих з Steam API, трансформування їх в C# об’єкти та додавання отриманих об’єктів до відповідних колекцій Games і Users.

Після цього витягаються унікальні жанри з колекції ігор Games та записуються в окрему колекцію Genres. До отриманих жанрів додаються підготовлені характеристики – рівні навичок, екшену, стратегічності, сюжету, зрілості, насилля та освіти [20]. На рисунку 3.4 наведений приклад об’єкта жанру з доданими характеристиками.

```
{
    Id = 19,
    Name = "Action",
    Skill = 7,
    Action = 10,
    Strategy = 5,
    Plot = 7,
    Maturity = 5,
    Violence = 6,
    Education = 4
};
```

Рисунок 3.4 – Приклад об’єкта жанру

Після того як отримана колекція жанрів з характеристиками, вона зберігається в файлі в форматі CSV.

Далі, кожна отримана гра з колекції Games зберігається в файлі у форматі CSV. Для кожної гри, у свою чергу, витягаються відповідні їй жанри і відношення типу «гра-жанр» зберігаються в файлі у форматі CSV – це необхідно для підтримки залежності між грою та жанром типу «багато до багатьох».

Після цього, на основі колекції Users визначаються відношення типу «користувач-гра-кількість зіграних годин» і записуються в колекцію UserGamePlaytime. На основі отриманої колекції для кожного користувача знаходиться грв, в яку він грав довше за інші і на основі цього створюється колекція улюблених ігор користувачів (UserFavouriteGame).

Далі, для кожного користувача на основі колекції UserFavouriteGame визначаються його улюблені жанри. Кожному жанру визначаються додаткові характеристики, які описують гравця – мінімальний та максимальний вік гравця, кількість відсотків гравців чоловічої та жіночою статі [21]. Приклад об’єкта жанру з додатковими характеристиками наведено на рисунку 3.5.

```
{
    NameEn = "Action",
    MaleChanceRatio = 60,
    FemaleChanceRatio = 40,
    AgeMin = 15,
    AgeMax = 28
};
```

Рисунок 3.5 – Приклад об’єкта жанру з додатковими характеристиками

На основі улюблених жанрів і їх додаткових характеристик визначається вік і стать користувача. Вік визначається як випадкове число в інтервалі який відповідає улюбленому жанру користувача. Стать визначається за допомогою алгоритму зваженого випадкового на основі визначаючих стать відсотків, які відповідають улюбленому жанру. Код алгоритму зваженого випадкового наведено на рисунку 3.6.

```
public static int GetWeightedRandomGenderValue(int maleRatioChannce, int femaleRatioChance)
{
    Random random = new Random();
    int x = random.Next(0, 100);

    if ((x -= maleRatioChannce) < 0)
    {
        return 1;
    }

    return 2;
}
```

Рисунок 3.6 – Алгоритму зваженого випадкового

Після того як визначені стать та вік користувачів, дані про них зберігаються в файлі у форматі CSV. Також, для кожного користувача витягаються і зберігаються в файлі у форматі CSV відношення типу «користувач-улюблений жанр» – це необхідно для підтримки залежності між користувачем та жанром типу «багато до багатьох».

Після цього, на основі колекції UserGamePlaytime визначаються відношення типу «гра-середня кількість годин скільки в неї грали» (колекція GamesAvgPlaytime). На основі колекції GamesAvgPlaytime визначаються оцінки,

які користувачі поставили б зіграним іграм. Середня кількість годин буде відповідати максимальній оцінці гри від користувача – 5. Якщо користувач грав в гру більше середньої кількості годин, то оцінка так само буде дорівнювати 5. Якщо користувач грав гру менше середньої кількості годин, то оцінка буде розраховуватися пропорційна. Було обрано використання середньої кількість годин (замість максимальної кількості годин) в якості відповідності до оцінки 5 для того, щоб зменшити розрив між мінімальною та максимальною оцінкою гри і тим самим забезпечити більшу варіативність оцінок.

Після отримання оцінок ігор користувачами, ці дані зберігаються в файлі у форматі CSV.

Таким чином, за допомогою окремого додатку для отримання вхідних даних були отримані CSV файли з такими даними:

– інформація про жанри з додатковими характеристиками;
– інформація про ігри;
– інформація про відношення типу «гра-жанр»;
– інформація про користувачів з їх характеристиками;
– інформація про відношення типу «користувач-улюблений жанр»;
– інформація про оцінки ігор користувачами.

Отримані CSV файли були використані для заповнення бази даних додатку за допомогою SQL команди BULK INSERT (див. рис. 3.7).

```
BULK INSERT games
FROM 'D:\EF_CodeFirst_Init_Commands\games.csv'
WITH (FIRSTROW = 1,
    CODEPAGE = '65001',
    FIELDTERMINATOR = '|',
    ROWTERMINATOR='\n',
    BATCHSIZE=250000,
    KEEPNULLS);
```

Рисунок 3.7 – Приклад команди BULK INSERT

Таким чином було отримано інформацію про 18 жанрів, 23857 ігор, 2987 користувачів та 180166 оцінок ігор користувачами. Додані до бази даних дані

використовуються для побудови, тренування та тестування рекомендаційної моделі.

3.2 Проведення експериментального дослідження

Як вже було наведено в попередніх розділах, для побудови, тренування та тестування рекомендаційної моделі був створений окремий шар додатку – шар мишиного навчання. Дані, які були записані в базу даних і описані в попередньому підрозділі, використовуються саме в цьому шарі. Дані про користувачів використовуються для кластеризації за аналітичним профілем. Дані про оцінки ігор користувачами використовуються для надання рекомендацій з використанням колаборативної фільтрації.

Для оцінки ефективності розробленої на базі запропонованого методу рекомендаційної системи використовувалася метрика RMSE (Root of Mean Squared Error) [1]. RMSE використовується для вимірювання різниці між передбачуваними значеннями моделі рекомендаційної системи та спостережуваними значеннями тестового набору даних. Чим цей показник нижче, тим ефективніша модель і точніші рекомендації. За тестовий набір даних зазвичай беруть 20 відсотків від загальних вхідних даних. Решта 80 відсотків використовується для навчання рекомендаційної моделі.

Для вибору кількості кластерів для розробленої рекомендаційної системи була проведена низка експериментів з замірюванням RMSE показника та показника R-квадрат. Показник R-квадрат вказує наскільки дані відповідають моделі. Значення цього показника варіюється від 0 до 1. Значення 0 означає, що дані є випадковими та не можуть відповідати моделі. Значення 1 означає, що модель точно відповідає даним. Показники RMSE та R-квадрат для різної кількості кластерів наведені у таблиці 3.1.

Таблиця 3.1 – Показники RMSE та R-квадрат для різної кількості кластерів

Показники	Кількість кластерів				
	2	3	4	5	6
RMSE	0.96	0.96	1.11	0.78	1.01
R-квадрат	0.54	0.53	0.48	0.72	0.32

При кількості кластерів більше 6, показник R-квадрат все більше зменшувався, тому в таблиці 3.1 наведені результати експериментів тільки для кластерів від 2 до 6. Для розробленої РС було обрано 5 кластерів для розбиття, оскільки при цій кількості кластерів рекомендаційна система показує найменший показник RMSE та найбільший показник R-квадрат.

Також було проведене порівняння показника RMSE рекомендаційної системи, яка використовує колаборативну фільтрацію, та розробленої рекомендаційної системи, яка використовує колаборативну фільтрацію на основі методу факторизації матриці низького рангу та k-means кластеризацію для розбиття користувачів на групи за аналітичним профілем. В таблиці 3.2 наведене порівняння показника RMSE для наведених рекомендаційних систем для користувачів з різною кількістю оцінок.

Таблиця 3.2 – Порівняння RMSE для різних рекомендаційних систем

RMSE для рекомендаційної системи	Користувачі (кількість оцінок)				
	16	42	84	112	166
з k-means кластеризацією та колаборитивною фільтрацією	0.82	0.88	0.78	0.93	0.99
з колаборитивною фільтрацією	1.53				

RMSE для рекомендаційної системи з колаборативною фільтрацією має одне і теж значення для різних користувачів оскільки використовується одна і та сама рекомендаційна модель для всіх користувачів. У випадку рекомендаційної системи з k-means кластеризацією, рекомендаційна модель відрізняється в залежності від того до якого кластеру був віднесений користувач. Як можна побачити з таблиці 3.2, показники RMSE для рекомендаційної системи з k-means кластеризацією нижчі (в деяких випадках майже в 2 рази) ніж для рекомендаційної системи, яка використовує тільки колаборативну фільтрацію. Отже, можна зробити висновок, що розроблена система з використанням k-means кластеризації будує більш ефективні рекомендаційні моделі і робить точніші рекомендації.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Реалізована рекомендаційна система створювалась на базі онлайн-магазину електронних ігор, розробленого в результаті бакалаврської роботи. Був створений новий шар додатку – шар машинного навчання, який відповідає за надання рекомендацій користувачам. Крім того, була створена додаткова логіка в усіх інших шарах додатку для коректного функціонування рекомендаційної системи.

4.1 Опис шару машинного навчання

Шар машинного навчання був створений з використанням бібліотеки ML.NET. ML.NET – це платформа для машинного навчання з відкритим кодом, яка доступна для .NET розробників. ML.NET дозволяє .NET розробникам розробляти та навчати власні моделі та вбудовувати кастомне машинне навчання у свої програми, використовуючи .NET. Ця платформа забезпечує завантаження даних із файлів та баз даних, перетворення отриманих даних та містить багато алгоритмів машинного навчання [22].

Шар машинного навчання складається з двох модулів – модулю кластеризації та модулю колаборативної фільтрації. Модуль колаборативної фільтрації використовує модуль кластризації для отримання вхідних даних для побудови рекомендаційної системи.

Модуль кластеризації містить наступні методи:

- GetClusteringModel();
- GetClusterPrediction();
- GetUsersByClusterId().

Метод GetClusteringModel() відповідає за побудову моделі кластеризації і робить наступне:

– отримує дані користувачів за допомогою методу шару бізнес-логіки;

– робить перетворення даних до виду необхідного для побудови моделі кластеризації;

– задає параметри кластеризації (кількість кластерів);

– тренує і повертає модель кластеризації.

Код методу GetClusteringModel() наведений на рисунку 4.1.

```
public ClusteringModel GetClusteringModel()
{
    var usersData = _userService.GetRegularUsers().Select(UserToUserData);

    IDataView dataView = _mlContext.Data.LoadFromEnumerable(usersData);

    var featuresColumnName = "Features";
    var pipeline = _mlContext.Transforms
        .Concatenate(featuresColumnName,
            "Age",
            "Gender",
            "SkillLevel",
            "ActionLevel",
            "StrategyLevel",
            "PlotLevel",
            "MaturityLevel",
            "ViolenceLevel",
            "EducationLevel")
        .Append(_mlContext.Clustering.Trainers.KMeans(featuresColumnName, numberOfClusters: _numberOfClusters));

    var model = pipeline.Fit(dataView);

    return model;
}
```

Рисунок 4.1 – Метод GetClusteringModel()

Метод GetClusterPrediction() відповідає за передбачення кластеру користувача і робить наступне:

– приймає на вхід ідентифікатор користувача і модель кластеризації;

– на основі моделі створює об’єкт типу PredictionEngine, який необхідний для передбачення кластеру користувача;

– робить передбачення кластеру користувача і повертає його;

Код методу GetClusterPrediction() наведений на рисунку 4.2.

```
public ClusterPrediction GetClusterPrediction(int userId, ClusteringModel model)
{
    var user = _userService.GetUserForML(userId);
    var userData = UserToUserData(user);

    var predictor = _mlContext.Model.CreatePredictionEngine<UserData, ClusterPrediction>(model);

    var prediction = predictor.Predict(userData);

    return prediction;
}
```

Рисунок 4.2 – Метод GetClusterPrediction()

Метод GetUsersByClusterId() приймає на вхід ідентифікатор кластеру і модель кластеризації та визначає і повертає користувачів, які належать до заданого кластеру.

Модуль колаборативної фільтрації містить наступні методи:

– LoadData();

– BuildAndTrainModel();

– EvaluateModel();

– PredictGameRating();

– GetRecommendedGames();

Метод LoadData() призначений для перетворення даних до виду, необхідного для побудови, тренування та тестування рекомендаційної моделі. Метод LoadData() робить наступне:

– приймає на вхід ідентифікатор користувача;

– отримує модель кластеризації за допомогою методу GetClusteringModel();

– визначає кластер користувача за допомогою методу GetClusterPrediction();

– визначає користувачів з кластеру поточного користувача за допомогою методу GetUsersByClusterId();

– отримує оцінки ігор для визначених користувачів за допомогою методу шару бізнес-логіки;

– записує 80 відсотків отриманих даних про оцінки до об’єкту для тренування моделі, а 20 відсотків отриманих даних до об’єкту для тестування моделі;

– повертає комплексний об’єкт з даними для тренування і тестування моделі.

Частина методу LoadData(), яка відповідає за запис даних для тренування і тестування наведена на рисунку 4.3.

```
var amountForTraining = (int)(userRatings.Count() * 0.8);
var amountForTesting = userRatings.Count() - amountForTraining;

var userRatingsForTraining = userRatings.Take(amountForTraining);
var userRatingsForTesting = userRatings.Skip(amountForTraining).Take(amountForTesting);

IDataView trainingDataView = _mlContext.Data.LoadFromEnumerable(userRatingsForTraining);
IDataView testDataView = _mlContext.Data.LoadFromEnumerable(userRatingsForTesting);

return (trainingDataView, testDataView);
```

Рисунок 4.3 – Частина методу LoadData()

Метод BuildAndTrainModel() призначений для побудови і тренування рекомендаційної системи і робить наступне:

– приймає на вхід набір даних для тренування;

– робить додаткові перетворення даних;

– задає параметри колаборативної фільтрації з мінімізацію на основі алгоритму градієнтного спуску;

– тренує і повертає рекомендаційну модель.

Код методу BuildAndTrainModel() наведений на рисунку 4.4.

```
public ITransformer BuildAndTrainModel(IDataView trainingDataView)
{
    IEstimator<ITransformer> estimator = _mlContext.Transforms.Conversion
        .MapValueToKey(outputColumnName: "UserIdEncoded", inputColumnName: "UserId")
        .Append(_mlContext.Transforms.Conversion.MapValueToKey(outputColumnName: "GameIdEncoded", inputColumnName: "GameId"));

    var options = new MatrixFactorizationTrainer.Options
    {
        MatrixColumnIndexColumnName = "UserIdEncoded",
        MatrixRowIndexColumnName = "GameIdEncoded",
        LabelColumnName = "Label",
        NumberOfIterations = _numberOfIterations,
        ApproximationRank = _approximationRanks
    };

    var trainerEstimator = estimator.Append(_mlContext.Recommendation().Trainers.MatrixFactorization(options));

    ITransformer model = trainerEstimator.Fit(trainingDataView);

    return model;
}
```

Рисунок 4.4 – Метод BuildAndTrainModel()

Метод EvaluateModel() використовує набір даних для тестування для оцінки якості отриманої рекомендаційної моделі з використанням метрик RMSE та R-квадрат. Код методу EvaluateModel() наведений на рисунку 4.5.

```
public RecommendationEvaluationModel EvaluateModel(IDataView testDataView, ITransformer model)
{
    var prediction = model.Transform(testDataView);

    var metrics = _mlContext.Regression.Evaluate(prediction, labelColumnName: "Label", scoreColumnName: "Score");

    var result = new RecommendationEvaluationModel
    {
        RootMeanSquaredError = metrics.RootMeanSquaredError,
        RSquared = metrics.RSquared
    };

    return result;
}
```

Рисунок 4.5 – Метод EvaluateModel()

Метод PredictGameRating() приймає на вхід ідентифікатор гри, ідентифікатор користувача, рекомендаційну модель і робить передбачення оцінки гри користувачем. Код методу PredictGameRating() наведений на рисунку 4.6.

```
public float PredictGameRating(int userId, int gameId, ITransformer model)
{
    var predictionEngine = _mlContext.Model.CreatePredictionEngine<GameRating, GameRatingPrediction>(model);

    var input = new GameRating { UserId = userId, GameId = gameId };

    var gameRatingPrediction = predictionEngine.Predict(input);

    return gameRatingPrediction.Score;
}
```

Рисунок 4.6 – Метод PredictGameRating()

Основним методом модулю колабоативної фільтрації, який використовує усі інші методи і повертає колекцію рекомендованих ігор користувачу, є метод GetRecommendedGames(). Він приймає на вхід ідентифікатор користувача.У цьому методі після загрузки даних, побудови рекомендаційної моделі і її оцінки на якість, отримуються усі ігри за допомогою методу шару бізнес логіки і для кожної гри робиться передбачення оцінки гри користувачем. Якщо передбачена оцінка більше

за 3.5, то гра додається до колекції рекомендованих ігор. Перед цим оцінка пропорційно переводиться до 100-відсоткової школи, щоб продемонструвати наскільки гра збігається з уподобаннями користувача.

Код всього модуля колаборативної фільтрації наведено у Додатку Г.

4.2 Опис додаткової логіки додатку

Для підтримки рекомендаційної системи і її коректного функціонування була створена додаткова логіка в усіх основних шарах додатку.

На шарі доступу до даних був створений новий репозиторій RatingRepository для забезпечення CRUD операцій над сутністю оцінки гри користувачем в базі даних. Для цієї сутності була створена модель доступу до даних UserGameRating (див. рис. 4.7).

```
public class UserGameRating
{
    public int Id { get; set; }

    public int UserId { get; set; }

    public virtual User User { get; set; }

    public int GameId { get; set; }

    public virtual Game Game { get; set; }

    public int Rating { get; set; }
}
```

Рисунок 4.7 – Модель UserGameRating

Також, була змінена модель доступу до даних жанру для того щоб підтримувати нові, необхідні для кластеризації, характеристики. На рисунку 4.8 наведені поля моделі Genre для цих характеристик.

```
public int? SkillLevel { get; set; }

public int? ActionLevel { get; set; }

public int? StrategyLevel { get; set; }

public int? PlotLevel { get; set; }

public int? MaturityLevel { get; set; }

public int? ViolenceLevel { get; set; }

public int? EducationLevel { get; set; }
```

Рисунок 4.8 – Нові поля моделі Genre

Крім того, модель доступу до даних користувача була розширена новими полями для додаткових характеристик користувач – стать, вік, улюблені жанри (див. рис. 4.9).

```
public int? Age { get; set; }

public int? Gender { get; set; }

public virtual ICollection<Genre> Genres { get; set; }
```

Рисунок 4.9 – Нові поля моделі User

На шарі бізнес-логіки були зроблені відповідні зміни в транспортних моделях для жанру та користувача. Був створений окремий сервіс RatingService для роботи з оцінками гри користувачем та відповідна транспортна модель. Частина коду сервісу RatingService наведена на рисунку 4.10.

```
public IEnumerable<UserGameRating> GetAllRatings()
{
    var ratings = _db.Ratings.GetAll().ToList();

    return ratings;
}

public UserGameRatingDTO GetRating(int userId, int gameId)
{
    var rating = _db.Ratings.GetRating(userId, gameId);
    var ratingDto = _mapper.Map<UserGameRating, UserGameRatingDTO>(rating);

    return ratingDto;
}
```

Рисунок 4.10 – Частина сервісу RatingService

На шарі ін’єкції залежностей була додана відповідна логіка для зіставлення нового сервісу і репозиторію для оцінок з їх інтерфейсами.

На шарі уявлення були внесені зміни у моделі уявлення, контролери і уявлення для користувачів і жанрів, щоб забезпечити підтримку нових характеристик.

На рисунку 4.11 наведено частину уявлення сторінки особистого кабінету користувача, де він може додати інформацію про свій вік, стать та улюблені жанри (характеристики необхідні для рекомендаційної системи).

На рисунку 4.12 наведено частину уявлення сторінки редагування інформації про жанр, де менеджер може додати або змінити характеристики жанру, необхідні для рекомендаційної системи.

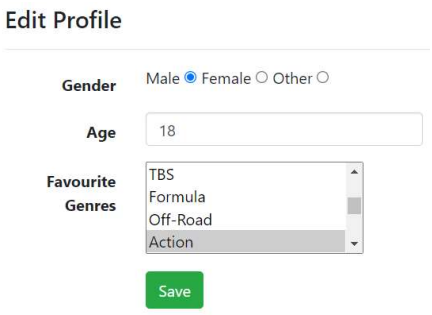

Рисунок 4.11 – Особистий кабінет користувача

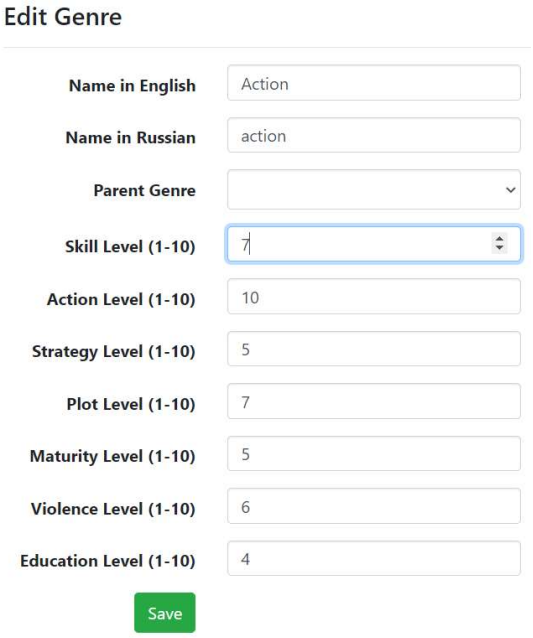

Рисунок 4.12 – Блок редагування характеристик жанру

Також, був створений новий контролер, моделі уявлення та уявлення для роботи з оцінками для ігор.

На сторінці гри був доданий блок оцінок, в якому користувач може виставити грі оцінку, а також подивитися середню оцінку гри серед усіх користувачів (див. рис. 4.13).

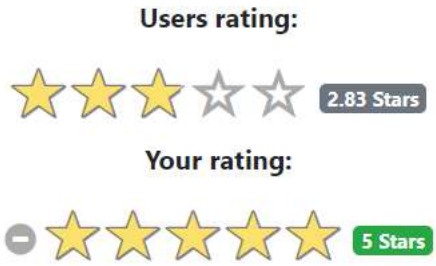

Рисунок 4.13 – Блок оцінок гри

Було додане уявлення для сторінки перегляду історії оцінок користувачем (див. рис. 4.14).

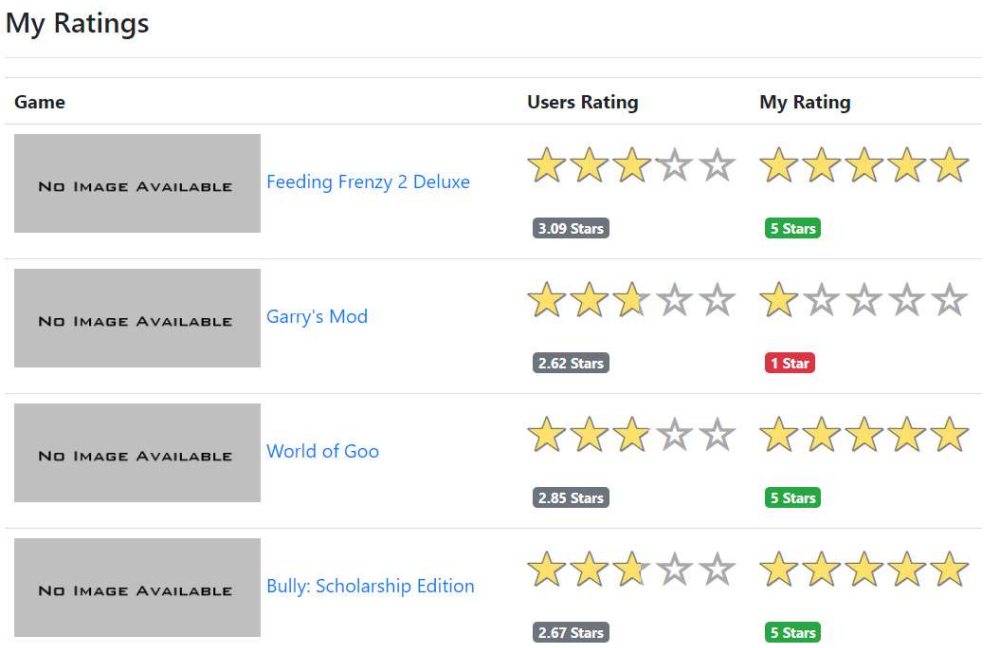

Рисунок 4.14 – Сторінка історії оцінок

Також було створене уявлення для сторінки зі списком рекомендацій для користувача (див. рис. 4.15).

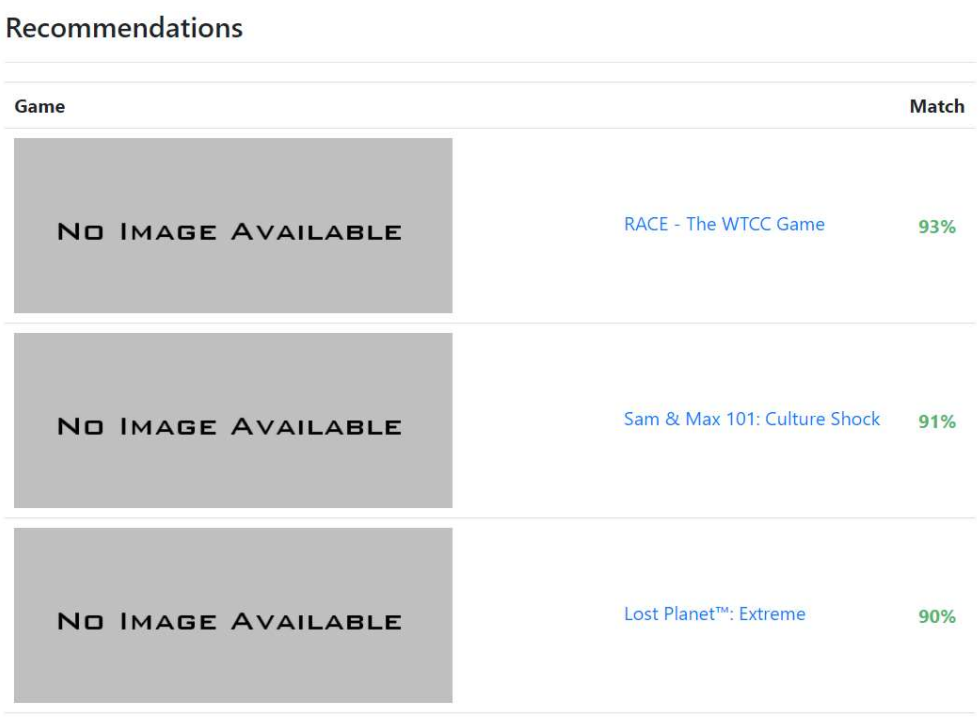

Рисунок 4.15 – Сторінка рекомендацій

На цій сторінці користувач може побачити рекомендації, які були запропоновані розробленою рекомендаційною системою, а також ступінь відповідності рекомендації вподобанням користувача у відсотках.

5 АНАЛІЗ ДОСЛІДНИЦЬКОЇ ЕКСПЛУАТАЦІЇ ТА МОЖЛИВИХ ЗАСТОСУВАНЬ

5.1 Аналіз можливих застосувань

В ході виконання магістерської кваліфікаційної роботи було досліджено методи побудови рекомендаційних систем, був запропонований оптимізований комбінований підхід та на основі цього підходу була розроблена і апробована рекомендаційна система на базі онлайн-магазину електроних ігор. Розроблена система дозволяє надавати користувачам більш якісні і точні персоналізовані рекомендації. Запропонований підхід і натреновані в ході роботи моделі можуть бути використані як для побудови нових рекомендаційних систем, так і для розширення і вдосконалення вже існуючих для будь-яких онлайн-магазину, які мають користувачів та продукти, які можна оцінювати. Програмну систему, що була розроблена, можна використовувати як готовий продукт для надання рекомендацій в сфері ігрової інтернет комерції. Крім того, розроблений інтерфейс користувач є зручним засобом для оцінювання продуктів, перегляду середніх оцінок продуктів, історії оцінок та рекомендацій.

Запропонований підхід і розроблений додаток призначений перш за все для власників бізнесу в сфері інтернет комерції, які хочуть збільшити свій прибуток та виділитися серед конкурентів за рахунок надання своїм користувачам можливості отримувати точні і якісні персоналізовані рекомендації завдяки ефективній рекомендаційній системі.

У подальшому можна додати до розробленої рекомендаційної системи на основі колаборативної фільтрації модуль контентної фільтрації, щоб робити ще точніші рекомендації за рахунок використання гібридного підходу. Крім того, можна провести дослідження і визначити, за рахунок яких додаткових характеристих аналітичного профіля користувача можна покращити якість кластеризації.

5.2 Опис тестування системи

Тестування програмного забезпечення – це спосіб перевірити, чи відповідає фактичний програмний продукт очікуваним вимогам, і забезпечити відсутність дефектів у програмному продукті. Тестування передбачає виконання програмних та системних компонентів за допомогою ручних або автоматизованих інструментів для оцінки одної або декількох функцій програмного забезпечення. Метою тестування програмного забезпечення є виявлення помилок, прогалин або відсутніх вимог у порівнянні з фактичними вимогами.

Для тестування розробленої системи було використано модульне і функціональне тестування.

Модульне тестування – це тип тестування програмного забезпечення, де тестуються окремі блоки або компоненти програми. Метою є перевірка того факту, що кожна одиниця програмного коду працює належним чином. Модульне тестування проводиться під час розробки (фази кодування) програми розробниками. Модульні тести ізолюють розділ коду та перевіряють його коректність. Одиницею може бути окрема функція, метод, процедура, модуль або об’єкт.

Для розробленої системи було проведене модульне тестування нових компонентів пов’язаних з роботою з оцінками, користувачами, жанрами та рекомендаціями.

Функціональне тестування – це тип тестування програмного забезпечення, який перевіряє програмну систему на відповідність функціональним вимогам та специфікаціям. Метою функціональних тестів є перевірка кожної функції програмного забезпечення шляхом надання відповідного вводу та перевірки результату на відповідність функціональним вимогам.

Для розробленої системи було проведене функціонально тестування на відповідність системи поставленим вимогам та специфікації.

ВИСНОВКИ

В ході магістерської кваліфікаційної роботи були досліджені методи побудови рекомендаційної системи, був розширений веб-додаток продажу ігор GameStore за допомогою впровадження у додаток розробленої рекомендаційної системи з використанням аналітичних профілів користувачів, що забезпечило адаптованість функціоналу сайту та пропозицій на основі поведінки та запитів користувачів.

Зокрема була проаналізована проблемна область розробки рекомендаційних систем. Також були проаналізовані існуючі аналоги – були виявлені переваги та недоліки рекомендаційних систем Netflix, Spotify, Steam. Крім того, були визначені вимоги до програмного продукту та призначення розробки.

Було проведено дослідження і порівняння основних підходів до побудови рекомендаційних системи та основних методів кластеризації на основі аналітичних профілів користувачів.

Був описаний вдосконалений метод побудови рекомендаційної системи на основі колаборативної фільтрації та кластеризації k-means для розділення користувачів на групи на основі аналітичного профіля.

Був проведений аналіз та UML-моделювання предметної області, спроектована база даних, розроблена архітектура система, розроблений алгоритм надання рекомендацій. На базі виділених підходів до перевірки рекомендаційних систем на точність було проведене експериментальне дослідження на предмет ефективності розробленої рекомендаційної системи.

Розроблена рекомендаційна система надає точні і релевантні персоналізовані рекомендації ігор на основі оцінок цільового користувача та інших користувачів системи з використанням кластеризації користувачів для підвищення точності рекомендацій для користувачів в рамках кластеру.

Розроблена рекомендаційна система реалізовує наступні функції:

– додавання користувачем оцінок та відгуків для ігор;

– створення профілю користувача з інформацією про нього;

– додавання користувачем інформації про улюблений жанр, платформу, видавця;

– кластеризація користувачів на основі профілів користувачів за допомогою методу k-means;

– тренування моделі колаборативної фільтрації для окремих кластерів користувачів;

Розроблену рекомендаційну систему можна використовувати в сервісах електронної комерції для надання релевантних рекомендацій користувачам для спрощення вибору об’єктів для покупки та збільшення прибутку та конкурентоспроможності постачальника послуг.

За результатами роботи були опубліковані тези на XVII науково-технічної конференції студентів, аспірантів, докторантів та молодих учених «Інноваційні Технології» 25-26 листопада 2020 року та написана та подана до опублікування до науково-аналітичного журналу «Біоніка інтелекту» стаття «Розробка комбінованого методу побудови рекомендаційної системи для онлайн-магазину електронних ігор» (див. додаток Д).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Isinkaye F.O., Folajimi Y.O., Ojokoh B.A., Recommendation systems: Principles, methods and evaluation, Egyptian Informatics Journal, Volume 16, Issue 3, 2018, Pages 261-273

2. Baptiste Rocca Introduction to recommender systems URL: https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada (дата звернення: 06.02.2021)

3. Блинков Н. Объем рынка компьютерных игр URL: http://www.it-weekly.ru/market/business/73058.html (дата звернення: 25.01.2021)

4. 2019 Essential Facts About the Computer and Video Game Industry URL: https://www.theesa.com/esa-research/2019-essential-facts-about-the-computer-and-video-game-industry/#:~:text=2018%20was%20a%20record%2Dbreaking,one%20gamer%20in%20their%20household (дата звернення: 09.02.2021)

5. Cheuque, Germán & Guzman Gomez, Jose Antonio & Parra, Denis. (2019). Recommender Systems for Online Video Game Platforms: the Case of STEAM. 763-771.

6. Jena K. C., Mishra S., Sahoo S. and Mishra B. K., Principles, techniques and evaluation of recommendation systems, International Conference on Inventive Systems and Control (ICISC), Coimbatore, India, 2017, pp. 1-6

7. Байдак В.Є. Веб-додаток продажу ігор. Бакалаврська атестаційна робота. ХНУРЕ. 2019, с. 92.

8. Netflix URL: https://uk.wikipedia.org/wiki/Netflix (дата звернення: 09.02.2021)

9. Boyd Clark How Spotify Recommends Your New Favorite Artist URL: https://clarkboyd.medium.com/ (дата звернення: 11.02.2021)

10. Chalyi S., Leshchynskyi V., Leshchynska, I. Method of forming recommendations using temporal constraints in a situation of cyclic cold start of the

recommender system Chalyi S., Leshchynskyi V., Leshchynska I. EUREKA, Physics and Engineering, 2019, 2019(4), с. 34-40

11. Chalyi S., Leshchynskyi V Temporal modeling of user preferences in recommender system Chalyi S., Leshchynskyi V. CEUR Workshop Proceedings, 2020, 2711, с. 518-528

12. Serhii Chalyi, Volodymyr Leshchynskyi. Method of constructing explanations for recommender systems based on the temporal dynamics of user preferences. «EUREKA: Physics and Engineering».-2020.- Number 3.-p.43-50.

13. Madushan Dileka Introduction to K-means Clustering URL: https://medium.com/@dilekamadushan/introduction-to-k-means-clustering-7c0ebc997e00 (дата звернення: 12.02.2021)

14. Chris I. K-Means Clustering Explained URL: https://medium.com/dataseries/k-means-clustering-explained-visually-in-5-minutes-b900cc69d175 (дата звернення: 10.02.2021)

15. Keerti Prajapati K-means Clustering explained in detailed URL:https://medium.com/@codingpilot25/k-means-clustering-explained-in-detailed-30484910e381 (дата звернення: 30.01.2021)

16. Yuan Lu, Jie Yang, Notes on Low-rank Matrix Factorization, URL: https://yangjiera.github.io/pdf/low-rank.pdf (дата звернення: 12.02.2021)

17. Doo Hyodan, Audrey Germain, Geordan Jove Recommendation System for Steam Game Store: An overview of recommender systems URL: https://audreygermain.github.io/Game-Recommendation-System/ (дата звернення: 29.01.2021)

18. Байдак В. Є., Мазурова О. О. Комбінований підхід до побудови рекомендаційної системи для онлайн-системи продажу електронних ігор //Інноваційні технології: матеріали наук.-техн. конф. студентів, аспірантів, докторантів та молодих учених / за заг. ред. П. В. Горінова, К. О. Бабікової , Л. М. Мельничук; ІНТЛ НАУ (м. Київ, 25-26 листоп. 2020 р.). Київ, 2020, с.106-110

19. What is Deployment Diagram? URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-deployment-diagram/ (дата звернення: 12.04.2021)

20. Clearwater D. What Defines Video Game Genre? Thinking about Genre Study after the Great Divide // The Journal of the Canadian Game Studies Association Vol 5(8) – 2019, – P. 29-49

21. Denton M. Game Genre & Statistics: Not All Games Are Created Equal (part 1) URL: https://www.gamify.com/gamification-blog/not-all-games-are-created-equal-pt1 (дата звернення: 14.04.2021)

22. Machine Learning for .NET URL: https://github.com/dotnet/machinelearning (дата звернення: 20.04.2021)