МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

ІНТЕЛЕКТУАЛЬНА СИСТЕМА КЛАСТЕРИЗАЦІЇ У КОМП'ЮТЕРНИХ ІГРАХ ДЛЯ ПЕРСОНАЛІЗОВАНОГО КОНТЕНТУ

Спеціальність 122 «Комп'ютерні науки»

122 – КРМ – 601.21810215

Виконав студент 6-го курсу, групи 601
_____Д. Ю. Коверзнєв
«19» лютого 2024 р.

Керівник: канд. техн. наук, доцент
_____І. О. Калініна
«19» лютого 2024 р.

Миколаїв – 2024

Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Освітньо-кваліфікаційний рівень **магістр**
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____2024р.

ЗАВДАННЯ
На кваліфікаційну роботу магістра

Коверзнєв Дмитро Юрійович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «Інтелектуальна система кластеризації у комп'ютерних іграх для персоналізованого контенту».

Керівник роботи Калініна Ірина Олександрівна, кандидат технічних наук, доцент.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «01» лютого 2024 р. № 20

2. Строк подання студентом роботи «19» лютого 2024 р.

3. Вхідні (початкові) дані до роботи: набір ігор з платформи Steam з основними характеристиками.

Очікуваний результат роботи: повнофункціональна рекомендаційна система комп'ютерних ігор виконана за допомогою кластеризації.

4. Зміст пояснювальної записки (перелік питань, які потрібно розглянути):

– дослідження сучасних методів та алгоритмів кластеризації для рекомендаційних систем;

– розробка та реалізація алгоритму побудови рекомендаційної системи на основі кластеризації для персоналізованого контенту у комп'ютерних іграх;

– аналіз ефективності та точності розробленої рекомендаційної системи;

– вивчення впливу різних параметрів та підходів до кластеризації на якість рекомендацій та їхню релевантність для користувачів;

– проведення експериментального дослідження з метою оцінки швидкості та масштабованості розробленої рекомендаційної системи.

5. Перелік графічного матеріалу: презентація.

6. Завдання до спеціальної частини: Правила безпеки роботи з обладнанням _____

_____

_____.

7. Консультанти:

Розділ	**Прізвище, ініціали та посада консультанта**	**Підпис**
Спеціальна частина з охорони праці	**д-р біол. наук, професор Л. І. Григор'єва**	
Методична частина	**канд. техн. наук, доцент І. О. Калініна**	

Керівник роботи ___канд._технічних наук, доцент Калініна І. О._____

(наук. ступінь, вчене звання, прізвище та ініціали)

_____

(підпис)

Завдання прийнято до виконання ___Коверзнєв Д. Ю._____

(прізвище та ініціали)

_____

(підпис)

Дата видачі завдання « 31 » ___жовтня_____ 2023 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи магістра

Тема: Інтелектуальна система кластеризації у комп'ютерних іграх для персоналізованого контенту.

№	Найменування роботи	Початок	Закінчення	Примітки
1	Визначення керівника і теми КРМ. Подання заяви на затвердження теми КРМ	01.09.2023	10.10.2023	Виконано
2	Отримання завдання на виконання КРМ	11.10.2023	01.11.2023	Виконано
3	Складання календарного плану на період виконання КРМ	02.11.2023	10.11.2023	Виконано
4	Огляд літератури за темою дослідження	11.11.2023	26.11.2023	Виконано
5	Проходження передатестаційної практики, збір та аналіз матеріалів до КРМ	27.11.2023	23.12.2023	Виконано
6	Аналіз предметної області та розробка технічного завдання. Моделювання результатів	25.12.2023	12.01.2024	Виконано
7	Опис фахової частини КРМ, зокрема дослідження публікацій щодо систем рекомендацій, огляд існуючих систем, реалізація системи рекомендації з аналізом отриманих результатів	13.01.2024	25.01.2024	Виконано
8	Розробка спеціальної частини з охорони праці та методичної частини	26.01.2024	02.02.2024	Виконано
9	Перший попередній захист КРМ на засіданні комісії кафедри	29.01.2024	29.01.2024	Виконано
10	Корегування роботи за результатами попереднього захисту	30.01.2024	05.02.2024	Виконано
11	Доробка та остаточне оформлення КРМ	06.02.2024	11.02.2024	Виконано
12	Другий попередній захист КРМ на засіданні комісії кафедри	12.02.2024	12.02.2024	Виконано
13	Подання КРМ, її електронної копії та інших документів (відгуку, рецензії) до захисту	19.02.2024	20.02.2024	Виконано
14	Захист КРМ перед екзаменаційною комісією (ЕК)	26.02.2024	27.02.2024	Виконано

Розробив студент _____ Коверзнєв Д. Ю._____ _____
(прізвище та ініціали) (*підпис*)

Керівник роботи __ канд. технічних наук, доцент Калініна І. О. _____
(*наук. ступінь, вчене звання, прізвище та ініціали)* *(підпис)*

«09» листопада 2023р.

АНОТАЦІЯ

до кваліфікаційної роботи магістра
студента групи 601 ЧНУ ім. Петра Могили

Коверзнєва Дмитра Юрійовича

на тему: **“ІНТЕЛЕКТУАЛЬНА СИСТЕМА КЛАСТЕРИЗАЦІЇ У КОМП'ЮТЕРНИХ ІГРАХ ДЛЯ ПЕРСОНАЛІЗОВАНОГО КОНТЕНТУ”**

Кваліфікаційна робота присвячена розробці та налагодженню інтелектуальної рекомендаційної системи, що ґрунтується на використанні методів кластеризації. Дослідження спрямоване на вдосконалення процесу надання рекомендацій, використовуючи різні методи валідації для визначення оптимальних параметрів кластеризації.

Об'єкт – процес надання рекомендацій за допомогою кластеризації.

Предмет – методи та алгоритми кластеризації для рекомендаційних задач.

Мета кваліфікаційної роботи - дослідження методів та ефективності інтелектуальних систем кластеризації у комп'ютерних іграх для персоналізованого контенту.

У роботі розглядаються теоретичні аспекти функціонування нейронних мереж та алгоритмів у контексті надання рекомендацій. Проводиться практичне застосування розробленої системи на реальних наборах даних. Результати експериментів порівнюються з існуючими методами побудови рекомендаційних систем, що дозволяє визначити ефективність та конкурентоспроможність запропонованого підходу.

Кваліфікаційна робота містить 116 сторінок, 19 рисунків, 6 таблиці та 1 додаток. В роботі використано 49 джерела.

Ключові слова: рекомендаційна система, кластеризація, k-means, ієрархічна кластеризація, аналіз даних, Tkinter, sklearn, matplotlib.

ABSTRACT

to the master's qualification work by the student of the group 601 of Petro Mohyla Black Sea National University

Koverzniev Dmitriy

"INTELLIGENT CLUSTERING SYSTEM IN COMPUTER GAMES FOR PERSONALIZED CONTENT"

The qualification work is devoted to the development and adjustment of an intelligent recommender system based on the use of clustering methods. The study is aimed at improving the recommendation process using various validation methods to determine the optimal clustering parameters.

The object is the process of providing recommendations using clustering.

The subject is clustering methods and algorithms for recommendation problems.

The purpose of the qualification work is to study the methods and effectiveness of intelligent clustering systems in computer games for personalized content.

The paper considers the theoretical aspects of the functioning of neural networks and algorithms in the context of providing recommendations. The practical application of the developed system is carried out on real data sets. The results of the experiments are compared with the existing methods of building recommendation systems, which allows us to determine the effectiveness and competitiveness of the proposed approach.

The qualification work contains 116 pages, 19 figures, 6 tables and 1 appendix. 49 sources were used in the work.

Keywords: recommender system, clustering, k-means, hierarchical clustering, data analysis, Tkinter, sklearn, matplotlib.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

КРС – колаборативна рекомендаційна система

ООК – об'єктно-орієнтовані РС

РС – рекомендаційні системи

СОК – суб'єктно-орієнтовані РС

GUI – graphical user interface

SSE – sum of squared errors

UI – user interface

ВСТУП

В онлайн-середовищі, де варіації вибору є величезними, виникає потреба в ефективному фільтруванні, встановленні пріоритетів та збалансованому представленні відповідної інформації для подолання проблеми інформаційного перенавантаження. Ця проблема стає потенційним завданням для багатьох інтернет-користувачів. Рекомендаційні системи вирішують це завдання, просіюючи величезний потік динамічно генерованої інформації для надання персоналізованого контенту та послуг користувачам.

Рекомендаційні системи є вигідними як для постачальників послуг, так і для користувачів. Вони зменшують витрати на пошук та відбір предметів у сфері інтернет-покупок, сприяючи оптимізації цього процесу. Також вони поліпшують прийняття рішень та сприяють збільшенню доходів постачальників послуг в інтернет-торгівлі, як ефективний інструмент для збільшення обсягу продажів.

Рекомендаційні системи підтримують користувачів, виходячи за межі стандартного пошуку і дозволяючи їм знаходити більше різноманітний контент та послуги. Таким чином, важливість використання точних та ефективних методів рекомендацій у системі, яка забезпечить високоякісні та надійні рекомендації для користувачів, надто важлива для недооцінки [1].

Об’єктом кваліфікаційної роботи є процес надання рекомендацій за допомогою кластеризації.

Предметом кваліфікаційної роботи є методи та алгоритми кластеризації для рекомендаційних задач.

Метою кваліфікаційної роботи є дослідження методів та ефективності інтелектуальних систем кластеризації у комп'ютерних іграх для персоналізованого контенту.

Завданням кваліфікаційної роботи є:

1) дослідження сучасних методів та алгоритмів кластеризації для рекомендаційних систем;

2) розробка та реалізація алгоритму побудови рекомендаційної системи на основі кластеризації для персоналізованого контенту у комп'ютерних іграх;

3) аналіз ефективності та точності розробленої рекомендаційної системи;

4) вивчення впливу різних параметрів та підходів до кластеризації на якість рекомендацій та їхню релевантність для користувачів;

5) проведення експериментального дослідження з метою оцінки швидкості та масштабованості розробленої рекомендаційної системи.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

У даному розділі проведено дослідження, що спрямоване на розгляд актуальних аспектів інтелектуальних систем кластеризації у галузі комп'ютерних ігор для оптимізації індивідуального ігрового досвіду. Дослідження базується на аналізі літературних джерел, де розглядаються різні аспекти застосування інтелектуальних систем у сфері геймінгу.

1.1 Аналіз проблемної області розробки рекомендаційних систем

В наш час інтенсивного потоку інформації, доступного для людини, автоматизація пошуку та фільтрації даних стає надзвичайно важливою. Одним з основних джерел отримання інформації є пошукові системи, проте без конкретного запиту важко відшукати потрібну інформацію. Цю проблему вирішують рекомендаційні системи, які здатні автоматично знаходити об'єкти, схожі на те, що цікавить чи потрібно користувачеві, базуючись на його індивідуальних інтересах. Аналізуючи інтереси користувачів, рекомендаційні системи прагнуть передбачити, що саме може бути найбільш цікавим для конкретного користувача у певний момент часу.

У загальному розумінні, рекомендаційні системи представляють собою алгоритми, спрямовані на надання користувачам релевантних рекомендацій у різних галузях, таких як фільми, книги, товари тощо. Їхня важливість визнається у різних сферах, оскільки ефективні рекомендаційні системи можуть призводити до значного збільшення прибутку, а також дозволяють підприємствам виділятися серед конкурентів [1].

Однією з таких сфер, де рекомендаційні системи грають важливу роль, є онлайн-системи продажу товарів, зокрема, електронних ігор. У контексті продажу ігор рекомендаційна система може ефективно визначати потреби відвідувачів і відповідно пропонувати їм цікаві пропозиції, що сприяє зростанню прибутку за рахунок підвищення конверсії, середнього чека та частоти повторних покупок [2].

За даними Європейського ринку ігор, у 2019 році понад 2,5 мільярди людей витрачали частину свого часу на електронні ігри. Індустрія електронних ігор стала однією з найцінніших у світі, зазначивши ріст прибутку на 10% за останній рік, досягнувши 116 млрд доларів. Високу цінність галузі забезпечує її значна диверсифікація, що включає різні платформи, жанри та ігрові категорії, а також платформи для соціальної взаємодії між гравцями. Ця адаптивність породила величезну кількість об'єктів з різними атрибутами, спроможних привертати різноманітних користувачів.

Онлайн-системи продажу електронних ігор, які здійснюють цифрове поширення ігор, відіграли ключову роль у цьому розвитку, дозволяючи користувачам купувати, рецензувати та обмінюватися враженнями про ігри. Щорічно понад 40 мільярдів доларів витрачається на електронні ігри в онлайн-системах [3].

Незважаючи на широкий вибір продуктів і велику аудиторію, вибір нової гри може бути викликом для користувача. Звіт Steam підтверджує, що приблизно 37% придбаних ігор користувачі ніколи не грали. Це створює необхідність у рекомендаційних системах, які здатні надавати персоналізовані пропозиції, допомагаючи користувачам відкривати нові ігри, що відповідають їхнім вподобанням.

Гра в електронні ігри, як і прослуховування музики, є періодичною активністю, де користувачі хочуть як повертатися до улюблених ігор, так і відкривати нові. Сучасні онлайн-системи продажу ігор вміють відстежувати взаємозв'язки між властивостями об'єктів та вподобаннями користувачів, вирішуючи подвійний виклик в усвідомленні потреби гравців у постійно оновлюваних і вражаючих іграх, а також у потребі в ефективному інструменті для відкриття нових ігрових експериенцій.

Рекомендаційна система визначається як стратегія прийняття рішень для користувачів у складних інформаційних середовищах. У контексті електронної комерції, вона є інструментом, який допомагає користувачам знаходити об'єкти, що

відповідають їхнім інтересам та уподобанням. Рекомендаційні системи також виступають як засіб допомоги та доповнення соціального процесу, використовуючи рекомендації інших користувачів для прийняття рішень у випадках, коли особистого досвіду або знань про альтернативи недостатньо. Вони ефективно вирішують проблему перевантаження інформацією, надаючи персоналізовані рекомендації щодо контенту та послуг [4].

Існує кілька підходів до побудови рекомендаційних систем, включаючи колаборативну фільтрацію, контентну фільтрацію та гібридний підхід. Колаборативна фільтрація рекомендує елементи, визначаючи інших користувачів зі схожими смаками, контентна фільтрація враховує властивості контенту та характеристики користувача, а гібридний підхід комбінує обидва методи [5].

Кластеризація — це неконтрольована задача машинного навчання, яка включає в себе групування схожих об'єктів у кластери або категорії на основі спільних характеристик чи властивостей.

Кластеризація може мати декілька аспектів:

– групування Гравців: використання кластеризації для групування гравців за їхньою грою, стилем гри, попередніми виборами або іншими характеристиками. Це дозволить створити групи користувачів зі схожими інтересами та стилів гри;

– рекомендації в Грі: кластеризація може використовуватися для розподілу ігрового контенту (завдань, завдань, рівнів тощо) на кластери, щоб персоналізовано рекомендувати гравцям подібний контент, який іншим користувачам із схожими вподобаннями дуже подобається;

– генерація Завдань: використання кластеризації для створення груп завдань або викликів, які відповідають стилю гри конкретного кластера гравців. Наприклад, створення викликів для любителів стратегічних ігор відрізняється від тих, які подобаються шутерам;

– персоналізація Графічних Налаштувань: групування гравців за їхніми відповідями на графічні налаштування (якості графіки, роздільної здатності тощо) за допомогою кластеризації. Це може використовуватися для рекомендацій та

автоматичного налаштування графіки;

– динамічне підлаштування гри: кластеризація може динамічно адаптувати гру до зміни вподобань гравців. Наприклад, якщо виявлено, що гравець змінив стиль гри, система може автоматично пристосовувати рекомендації та ігровий контент.

Використання кластеризації в іграх дозволяє створити більш індивідуалізований та захоплюючий досвід для гравців, а також покращити роботу систем рекомендацій у сфері геймінгу.

Кластерний аналіз представляє собою завдання розділення заданої вибірки об'єктів на групи або кластери так, щоб об'єкти всередині кожного кластера були схожі між собою, а об'єкти різних кластерів суттєво відрізнялися. Це завдання входить в область статистичної обробки та представляє собою вид некерованого навчання.

Для підвищення точності рекомендацій використовуються різноманітні методи кластеризації, що допомагають поділити користувачів на групи з використанням різних методів, таких як кластеризація на основі мінімума, максимума, групового середнього та інших методів.

Кластеризація k-середніх є одним з найвідоміших і широко використовуваних методів часткової кластеризації. Алгоритм k-середніх приймає вхідний параметр k і розбиває набір n об'єктів на неперекриваючіся k кластерів, де k<n. Основна мета цього методу - мінімізувати суму квадратів відстані між об'єктом і центроїдом, який називається сумою квадратів помилок.

Алгоритм k-середніх працює наступним чином. По-перше, випадковим чином вибирають k центроїдів з набору даних. Центроїд представляє середнє значення всіх об'єктів у кластері. Наступним кроком є призначення залишених об'єктів до найближчого кластера на основі відстані між об'єктом і середнім значенням кластера, яке є центроїдом, і потім розрахунок нового середнього значення для кожного кластера. Цей процес повторюється до тих пір, поки не

відбувається зміна значень центроїдів. Іншими словами, до тих пір, поки функція критерію не збігається [6].

У цьому методі мінімізується сума квадратів відстаней між кожним спостереженням та центром його кластера. Функціонал мінімізації виглядає як сума квадратів відстаней та обирається таким чином, щоб кожне спостереження було призначено до кластера, чий центр є найближчим до нього за метрикою.

Головні переваги методу k-середніх полягають у його простоті та швидкості виконання. Цей метод виявляється більш зручним для кластеризації великої кількості спостережень порівняно з методом ієрархічного кластерного аналізу, оскільки у випадку останнього дендрограми можуть стати перевантаженими і втратити наочність.

Хоча метод k-середніх має свої переваги, існують значні недоліки, які призводять до розробки різних модифікацій та нечітких аналогів. Одним з основних недоліків простого методу є порушення умови зв'язності елементів у межах одного кластера. У зв'язку з цим розробляються модифікації, а також нечіткі варіанти (fuzzy k-means methods), які дозволяють елементам належати декільком кластерам з різним ступенем приналежності на першій стадії алгоритму [7].

Незважаючи на очевидні переваги методу, існують суттєві недоліки:

1) результат класифікації сильно залежить від початкових позицій кластерних центрів;

2) алгоритм чутливий до викидів, що може викривляти середнє значення;

3) кількість кластерів повинна бути визначена заздалегідь дослідником.

Незважаючи на ці недоліки, метод k-середніх залишається популярним і успішно використовується в різних галузях, таких як маркетингові сегментації, геостатистика, астрономія, сільське господарство і т. д.

Ієрархічна кластеризація використовує принцип "дерева". Кожна точка спочатку вважається окремим кластером, і на кожному етапі об'єднуються найбільш схожі кластери, створюючи новий вузол у дереві. Результат представляє собою дендрограму, і кількість кластерів може визначатися на основі її аналізу [7].

Одина з переваг - відсутність необхідності передбачення кількості кластерів перед запуском алгоритму.

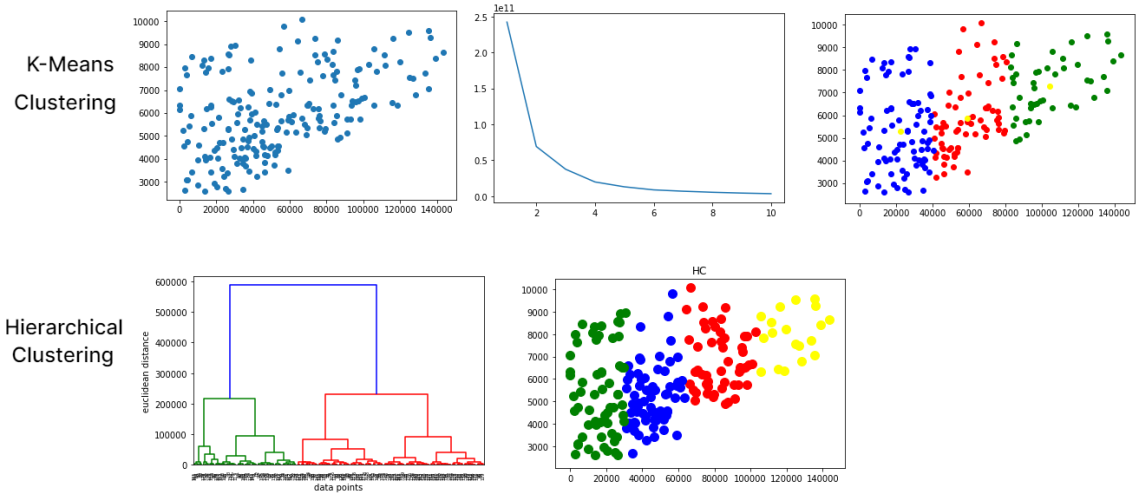


Рисунок 1.1 – Порівняння k-means та ієрархічної кластеризації

K-Means та ієрархічна кластеризації мають ряд відмінностей, які наведені у таблиці 1. Обидві методики мають свої сильні та слабкі сторони, що робить їх придатними для різних сценаріїв.

Таблиця 1.1 – Ключові відмінності між ієрархічною та K-Means кластеризацією

Особливість	**Ієрархічна кластеризація**	**Кластеризація K-Means**
Тип кластеризації	Агломераційний (знизу вгору) або розділовий (зверху вниз)	Частковий (на основі центроїда)
Кількість кластерів	Може бути визначений за дендрограмою або обраний користувачем	Повинен бути визначений користувачем

Продовження таблиці 1.1

Особливість	**Ієрархічна кластеризація**	**Кластеризація K-Means**
Форма грона	Може працювати з неопуклими формами та змінними розмірами кластерів	Припускає сферичні кластери однакового розміру
Метрика відстані	Можна використовувати різні міри відстані, наприклад евклідову, манхеттенську чи косинусну	Потрібно використовувати евклідову відстань
Масштабованість	Може бути обчислювально дорогим для великих наборів даних або багатьох кластерів	Може ефективно обробляти великі набори даних і багато кластерів
Інтерпретованість	Надає ієрархічну структуру та дендрограму, які можуть допомогти в інтерпретації результатів кластеризації	Забезпечує центри кластерів і призначення, але не має ієрархічної структури
Стійкість до викидів	Може обробляти викиди та шум, але може об’єднувати їх у існуючі кластери	Чутливий до викидів і шуму, які можуть впливати на центри кластерів

Вибір між використанням ієрархічної кластеризації та кластеризації K-Means може становити викликове завдання [7].

Ось кілька аспектів, які варто врахувати:

– розмір даних: Ієрархічна кластеризація вимагає великих обчислень і не є ефективною для обробки великих обсягів даних. З іншого боку, кластеризація K-Means працює швидше і може застосовуватися до обширних наборів даних;

– структура даних: Ієрархічна кластеризація підходить для структурованих даних, тоді як кластеризація K-Means може застосовуватися як до структурованих, так і до неструктурованих даних;

– кількість кластерів: Якщо ви вже знаєте, скільки кластерів ви хочете сформувати, кластеризація K-Means може бути оптимальним варіантом. У випадку невизначеності стосовно кількості кластерів, ієрархічна кластеризація може бути більш передпочтенною.

Узагальнюючи, ієрархічна кластеризація та кластеризація K-Means представляють собою потужні алгоритми, які можуть бути застосовані в різноманітних сценаріях.

Показники внутрішньої валідності кластерів, такі як індекси Калінскі–Харабаса, Данна чи Девіса–Болдіна, часто використовуються для визначення оптимальної кількості сегментів для поділу набору даних [8].

При створені рекомендаційних систем використовуються спеціальні метрики для оцінки кластеризації:

– Silhouette Score: вимірює, наскільки об'єкт в своєму кластері схожий на інші об'єкти в цьому ж кластері порівняно з іншими кластерами. Визначається для кожного об'єкта шляхом порівняння середньої відстані до інших об'єктів у тому ж кластері з середньою відстанню до об'єктів у найближчому сусідньому кластері. Високий Silhouette Score (близько 1) вказує на хорошу кластеризацію, коли об'єкти в кластерах схожі між собою, а відстань між кластерами достатньо велика;

– Davies-Bouldin Index: вимірює середню відстань між кожним кластером та кластером, який найбільше схожий на нього. Обчислюється як середнє відношення внутрішніх індексів подібності між кластерами до максимального міжкластерного індексу подібності. Менший індекс вказує на кращу кластеризацію, коли кожен кластер схожий на свій найбільш схожий кластер, і вони

віддалені один від одного;

– Calinski-Harabasz Index: вимірює відношення дисперсії всередині кластерів до дисперсії між кластерами. Розраховується як відношення сліду матриці розкиду всередині кластерів до сліду матриці розкиду між кластерами. Велике значення вказує на кращу кластеризацію, коли внутрішні кластери компактні, а вони розташовані далеко один від одного.

Сучасні рекомендаційні системи переважно базуються на веб-додатках та окремих програмах. Веб-додатки використовують веб-браузери та веб-технології для надання користувачам можливості взаємодіяти з контентом, шукаючи і купуючи ігри, обмінюючись враженнями та взаємодіючи з іншими користувачами. Це ефективно допомагає вирішувати завдання управління контентом, обробки інформації та покупок, а також сприяє спільній роботі та обміну інформацією.

У веб-розробці, зазвичай, використовуються мови програмування, які підтримуються браузерами, такі як JavaScript та HTML. Програми такі як Steam написані на мовах програмування Java, C++, Objective-C. Деякі додатки є динамічними і вимагають обробки на стороні сервера, тоді як інші є повністю статичними і не потребують обробки на сервері. Веб-додаток для правильної роботи потребує веб-сервера для обробки клієнтських запитів, сервера додатків для виконання завдань та, іноді, взаємодії з базою даних для зберігання інформації.

Для рекомендаційної системи, що базується на веб-додатку для продажу ігор, важливо опрацьовувати дані, такі як інформація про ігри (назва, жанри, платформи, популярність, рейтинг і т. д.), інформація про користувачів (логін, пароль, роль, вподобання), інформація про замовлення користувачів (дата замовлення, обрані ігри і т. д.), а також інформація про оцінки ігор від користувачів. Усі ці дані, необхідні для рекомендаційної системи, повинні бути структуровані і збережені в спеціальному сховищі, яким є база даних.

Python – дуже потужна мова програмування з великим обсягом бібліотек в тому числі й для машинного навчання Для розробки рекомендаційної системи на Python використовують різноманітні бібліотеки.

Python вирізняється численними перевагами, серед яких можна виділити наступні:

– чистий синтаксис: використання відступів для виділення блоків полегшує роботу з кодом і підвищує читабельність;

– переносність програм: Python є переносним на більшість платформ, що сприяє розповсюдженню програм між різними операційними системами;

– багатий стандартний дистрибутив: мова має велику кількість корисних модулів, включаючи засоби для розробки графічного інтерфейсу;

– інтерактивний режим: можливість використовувати Python в діалоговому режимі дозволяє швидко експериментувати та розв'язувати прості задачі;

– середовище розробки IDLE: стандартний дистрибутив містить просте, але потужне середовище розробки IDLE, написане мовою Python;

– математичні можливості: зручність для розв'язання математичних проблем, включаючи роботу з комплексними числами та потужність у діалоговому режимі;

– відкритий код: можливість редагування мови користувачами сприяє розвитку та спільноті;

– ефективні структури даних: Python надає ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування;

– широкий спектр застосувань: його синтаксис, динамічна обробка типів і інтерпретована природа роблять його ідеальним для написання скриптів та швидкої розробки прикладних програм у різних галузях;

– розширюваність: можливість розширення функціоналу та типів даних за допомогою C чи C++ робить Python зручним для налагодження та використання у прикладних програмах.

Python також відомий своєю активною спільнотою, великою кількістю доступних бібліотек та постійним оновленням.

Tkinter є стандартною бібліотекою для створення графічного інтерфейсу користувача в мові програмування Python. Вона надає засоби для розробки вікон, кнопок, полів введення та інших елементів GUI, що спрощує створення зручного та інтуїтивно зрозумілого інтерфейсу для користувачів.

Scikit-learn надає широкий спектр інструментів для роботи з методами машинного навчання, включаючи алгоритми кластеризації, такі як K-Means та Ієрархічна кластеризація, які використовуються для групування схожих об'єктів у кластери. Ці алгоритми допомагають створювати структуру кластерів, що використовується для персоналізації рекомендацій та покращення користувацького досвіду.

1.2 Аналіз існуючих аналогів

На сучасному етапі розвитку комп'ютерних ігор рекомендаційні системи вже успішно впроваджені в онлайн-платформах з різних галузей. Прикладами таких систем є відомі платформи, такі як Steam, Netflix та Spotify, кожна з яких використовує власні підходи та методи для створення персоналізованих рекомендацій.

Netflix, американський провайдер медійних послуг, в основному спеціалізується на передплачуваних послугах мережевої трансляції кінофільмів та телепередач. Їхні персоналізовані рекомендації базуються на взаємодії користувачів із сервісом, а також іншими користувачами зі схожими вподобаннями. Враховуються інформація про фільми, час доби перегляду, тип пристрою та інші фактори, що формують індивідуальний профіль вподобань [9].

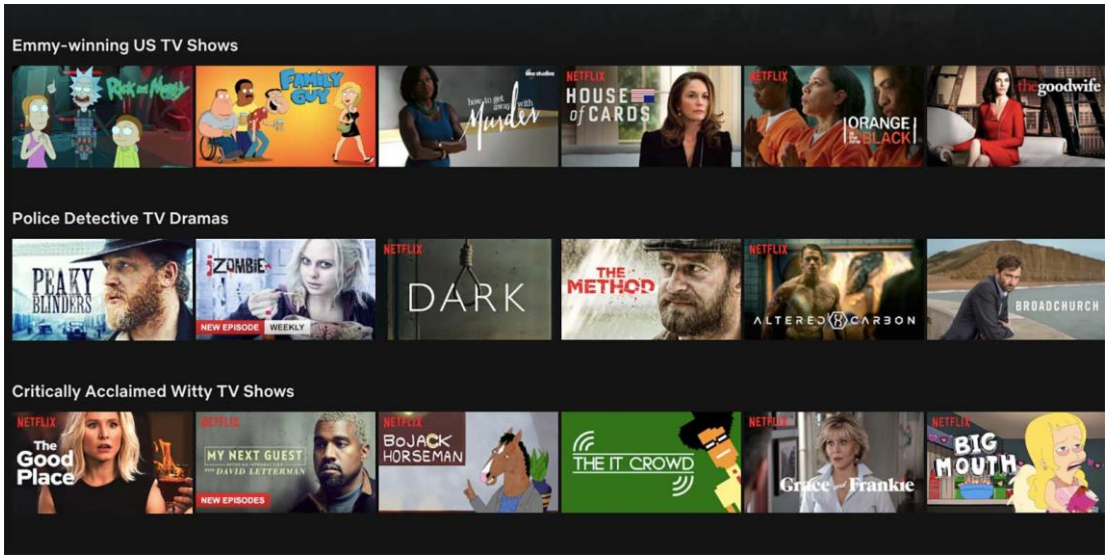


Рисунок 1.2 – Сторінка рекомендацій платформи Netflix

Netflix, завдяки своїй складній системі рекомендацій, враховує історію перегляду, взаємодію з сервісом та інші фактори для пропозицій, які найбільше відповідають інтересам користувача.

Spotify використовує глибоке навчання для аналізу взаємодії з музикою та іншим контентом, створюючи персоналізовані плейлисти та рекомендації. У той час як Steam, спеціалізуючись на іграх, використовує колаборативну фільтрацію для врахування ігрових вподобань користувачів та їхньої історії гри [10].

Spotify, цифрова музична платформа, також застосовує рекомендаційні системи на основі користувальницької взаємодії. Вони використовують глибоке навчання для виявлення закономірностей між виконавцями, жанрами та уподобаннями користувачів, враховуючи колаборативну фільтрацію, обробку природної мови та аудіо моделі [11].

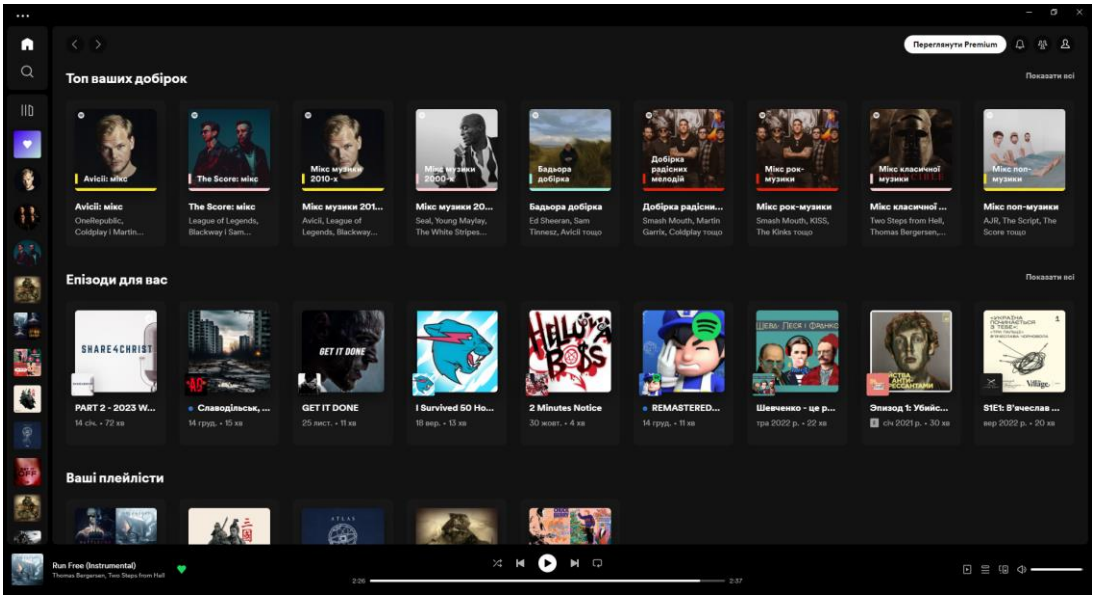


Рисунок 1.3 – Головна сторінка Spotify з рекомендаціями

Steam, як онлайн-платформа для гравців, також використовує машинне навчання та колаборативну фільтрацію для рекомендацій. Алгоритми враховують ігрові звички користувачів, їхній вибір ігор, відгуки та інші фактори, що формують персоналізовані рекомендації. Ці приклади розкривають, як лідери галузі використовують інтелектуальні системи для побудови персоналізованих рекомендацій, щоб задовольнити унікальні потреби та інтереси кожного користувача [12].

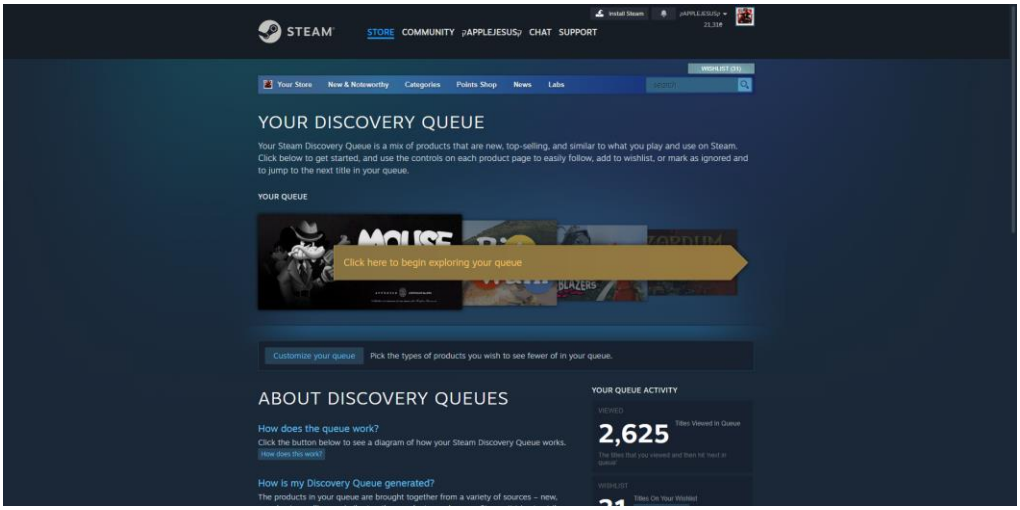


Рисунок 1.4 – Сторінка списку рекомендацій платформи Steam

Проаналізувавши вищевказані приклади використання рекомендаційних систем відомих онлайн-платформ, можна визначити, що такі інтелектуальні системи є ключовим елементом сучасної галузі комп'ютерних ігор. Їх впровадження стає необхідним етапом розвитку, оскільки комп'ютерні ігри вже давно перетворилися на не лише засіб розваг, але й активний інтерактивний відпочинок, який впливає на різні сфери життя гравців.

Розглядаючи практику таких гігантів як Steam, Netflix та Spotify, можна визначити, що персоналізовані рекомендації стають ключовим елементом взаємодії із користувачами. Кожна з цих платформ використовує свої власні методи та підходи, але загальним для них є фокус на індивідуальних потребах та вподобаннях кожного гравця.

Amazon використовує рекомендаційні системи для поліпшення користувацького досвіду та стимулювання продажів. Основні принципи та елементи їхньої рекомендаційної системи включають:

– колаборативна фільтрація: Amazon використовує колаборативну фільтрацію для порівняння поведінки користувачів та визначення схожості між ними. Наприклад, якщо користувач А і користувач Б мають схожі історії покупок, то товари, які сподобалися користувачеві А, можуть бути рекомендовані користувачеві Б;

– контент-базована фільтрація: окрім порівняння поведінки користувачів, Amazon враховує також характеристики товарів. Наприклад, якщо користувач переглядав електроніку, система може рекомендувати подібні товари в тій же категорії;

– особистий кабінет користувача: Amazon веде деталізований особистий кабінет для кожного користувача. Це включає історію покупок, оцінки, відгуки та збережені елементи. Ці дані використовуються для аналізу вподобань та створення персоналізованих рекомендацій;

– рекомендації на сторінках товарів: Amazon виводить рекомендації на сторінках товарів, щоб спонукати користувачів до додаткових покупок. Це може

включати "Спільно купували разом", "Схожі товари" та інші пропозиції;

– пошукові рекомендації: Amazon використовує рекомендації під час пошуку. Наприклад, під час введення запиту система може пропонувати варіанти для полегшення процесу пошуку;

– машинне навчання: Amazon використовує технології машинного навчання, такі як алгоритми глибинного навчання, для аналізу великої кількості даних та прогнозування вподобань користувачів.

Результатом такого підходу є ефективна система рекомендацій, яка сприяє збільшенню продажів та поліпшенню задоволення користувачів від покупок на Amazon.

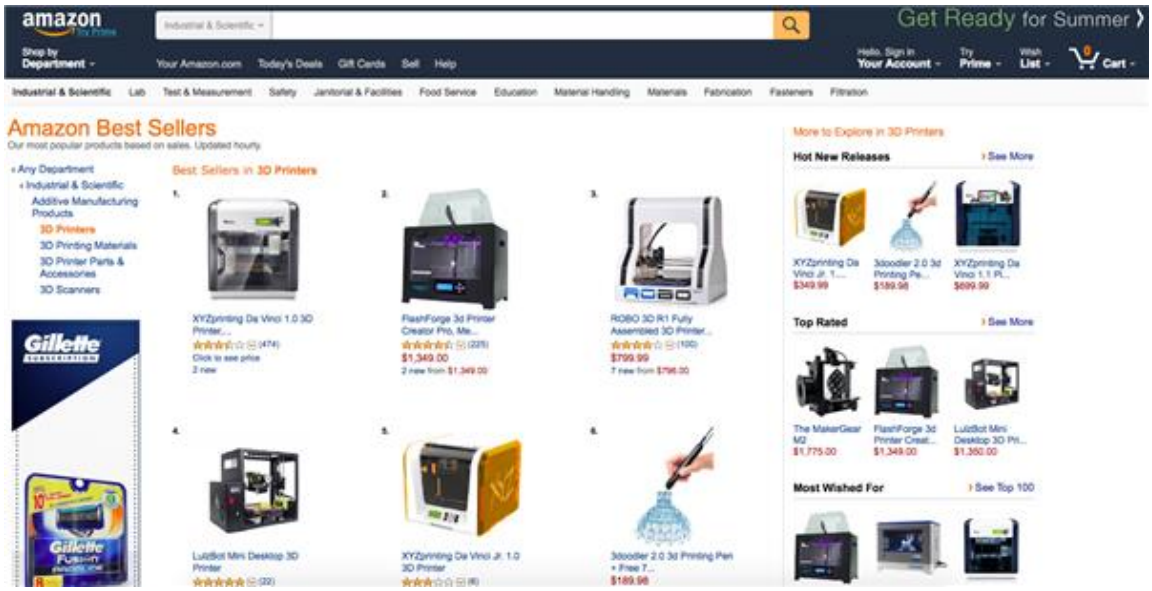


Рисунок 1.5 – Головна сторінка Amazon

Визначивши успішний досвід цих платформ, можна визнати актуальність розробки інтелектуальних систем кластеризації у комп'ютерних іграх. Такі системи можуть значно покращити індивідуальний ігровий досвід, надаючи гравцям контент, який відповідає їхнім унікальним вподобанням та стилю гри. У нашій роботі ми проведемо аналіз предметної області, розглянемо використані інструменти та аналізуватимемо попередні дослідження та публікації для

визначення оптимальних шляхів впровадження інтелектуальних систем кластеризації для персоналізованого рекомендаційного контенту в комп'ютерних іграх.

1.3 Постановка задачі

Мета даної роботи полягає в аналізі предметної області, що стосується інтелектуальних систем кластеризації у контексті комп'ютерних ігор та їхнього впливу на створення персоналізованого рекомендаційного контенту. Також робота передбачає розробку інтелектуальної системи кластеризації, яка сприятиме персоналізації рекомендаційного контенту у світі комп'ютерних ігор, покращуючи тим самим ігровий досвід та задоволення гравців.

Для коректної роботи рекомендаційної системи необхідно:

– якісні дані: забезпечити наявність достатнього обсягу якісних та репрезентативних даних щодо користувачів та їх взаємодії з продуктами;

– алгоритми рекомендацій: вибрати чи розробити відповідні алгоритми рекомендацій, такі як колаборативна фільтрація, контент-фільтрація, глибинне навчання тощо;

– персоналізація: врахувати потреби користувачів у персоналізованих рекомендаціях, враховуючи їхні вподобання та попередні дії;

– оптимізація: забезпечити ефективну оптимізацію алгоритмів та інфраструктури для швидкості та високої продуктивності системи;

– взаємодія та зворотній зв'язок: розробити механізми взаємодії з користувачами та збору зворотного зв'язку для постійного вдосконалення системи;

– захист даних: забезпечити безпеку та конфіденційність особистих даних користувачів;

– інтеграція: інтегрувати рекомендаційну систему в існуючі платформи та середовища для зручності користувачів;

– тестування та валідація: провести обширне тестування та валідацію

системи для переконання в її надійності та ефективності.

Рекомендаційна система повинна мати можливість:

- аналіз даних:
  - завантаження даних із файлу;
  - створення нового стовпця з рейтингом у відсотках;
- інтерфейс:
  - візуальний інтерфейс з урахуванням бібліотеки Tkinter;
  - вибір гри з списку;
- кластеризація:
  - вибір параметра для кластеризації зі списку;
  - сортування даних за вибраним параметром;
  - визначення кластерів за допомогою методів K-Means або Ієрархічної кластеризації;
- рекомендації:
  - пошук схожих ігор із використанням кластеризації;
  - виведення рекомендацій у вигляді спливаючого повідомлення;
- оновлення графіка:
  - вибір методу кластеризації та вказівку кількості кластерів;
  - можливість збереження графіки у зображення.

Для реалізації рекомендаційної системи були обрані наступні технології. Вся система буде базуватися на платформі Tkinter для створення інтерфейсу користувача, а також на бібліотеці машинного навчання sklearn для проведення кластеризації та надання персоналізованих рекомендацій користувачам.

Програма написана на мові програмування Python у середовищі Visual Studio Code

У програмі використовуються наступні бібліотеки:

- tkinter - бібліотека для створення графічного інтерфейсу користувача (GUI) в програмах на мові програмування Python;

– ttk - модуль ttk (themed Tkinter) розширює звичайну бібліотеку tkinter та надає додатковий функціонал, такий як нові стилі для віджетів;

– messagebox - частина бібліотеки tkinter, використовується для виведення повідомлень та сповіщень користувачу;

– filedialog - частина бібліотеки tkinter, дозволяє вибирати файли та теки через діалогове вікно;

– pandas - бібліотека для роботи з аналізу та обробки даних;

– matplotlib - бібліотека для візуалізації даних та створення графіків;

– FigureCanvasTkAgg - клас, який забезпечує інтеграцію графіків matplotlib в tkinter;

– mplcursors - бібліотека для відображення виринаючих підписів на графіках matplotlib при наведенні;

– sklearn - бібліотека для машинного навчання, використовується для кластеризації (KMeans, AgglomerativeClustering) та обчислення метрик.

Ці бібліотеки дозволяють вам створювати графічний інтерфейс, обробляти та візуалізувати дані, використовувати методи машинного навчання для кластеризації та отримання рекомендацій.

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

Висновки до розділу 1

У ході аналізу проблемної області розробки рекомендаційних систем було виявлено, що ці системи стали необхідним інструментом в різних галузях, включаючи онлайн-торгівлю, медіа-індустрію та ігрову сферу. Вони відіграють важливу роль у підвищенні задоволеності користувачів, збільшенні продажів і покращенні залучення аудиторії.

Проведений аналіз існуючих аналогів рекомендаційних систем показав різноманітність підходів до рекомендацій, включаючи контентну фільтрацію,

колаборативну фільтрацію та гібридні методи. Це дозволяє розробникам обирати найбільш ефективний підхід для конкретної задачі.

В рамках постановки задачі було визначено, що основною метою проекту є створення інтелектуальної системи рекомендацій у галузі комп'ютерних ігор для надання персоналізованого контенту користувачам.

2 ДОСЛІДЖЕННЯ ІСНУЮЧИХ МЕТОДІВ КЛАСТЕРИЗАЦІЇ

2.1 Огляд Методів Кластеризації

2.1.1 Дослідження методів кластеризації

Кластеризація є завданням групування об'єктів так, щоб подібні елементи були об'єднані в одній групі або кластері. Ця задача виявляється ключовою в областях, таких як пошук інформації, аналіз зображень, біоінформатика і розпізнавання образів. Існують два основних підходи до вирішення завдання кластеризації: ієрархічний та на основі центроїдів, які розглядаються нижче.

Ієрархічна кластеризація – це метод, що ґрунтується на створенні ієрархії кластерів і реалізується за допомогою агломеративної та дивізійної стратегій. Агломеративний алгоритм спочатку розглядає кожен об'єкт як окремий кластер, а потім починає об'єднувати схожі пари кластерів, зменшуючи їх кількість на кожній ітерації. Цей процес триває, поки не утвориться один кластер або K кластерів.

Щодо алгоритму агломеративної ієрархічної фільтрації, його виконання включає обчислення матриці близькості, розгляд кожної точки даних як окремого кластера, об'єднання двох найближчих кластерів та оновлення матриці близькості. Ці кроки повторюються до утворення одного кластера або K кластерів, де ключовою операцією є обчислення близькості між кластерами.

Алгоритм агломеративної ієрархічної кластеризації має такий хід виконання.

Крок 1. Обчислюється матриця близькості.

Крок 2. Кожна точка даних розглядається як окремий кластер.

Крок 3. Два найближчі кластери об'єднуються, і матриця близькості оновлюється.

Крок 4. Крок 3 повторюється до тих пір, поки не залишиться лише один кластер або K кластерів.

Ключовою операцією в цьому алгоритмі є обчислення близькості між двома кластерами.

У дивізійному алгоритмі, який протистоїть агломеративному, всі об'єкти спочатку знаходяться в єдиному кластері. Після цього цей кластер рекурсивно розбивається на підкластери, відокремлюючи неподібні точки даних у власні кластери. Цей процес призводить до формування n кластерів.

Для визначення близькості між кластерами застосовуються різні методи:

– мінімум (алгоритм Single-Linkage): визначення близькості двох кластерів полягає у визначенні мінімуму відстаней між їхніми точками;

– максимум (алгоритм Complete-Linkage): близькість визначається максимумом відстаней між точками різних кластерів;

– групове середнє: обчислює середнє значення відстаней між парами точок з різних кластерів;

– відстань між центроїдами: використовує відстань між центроїдами кластерів для визначення їхньої близькості;

– метод Варда: обчислює суму квадратів відстаней між точками кластера.

Ці підходи мають свої переваги та недоліки, і вибір конкретного методу може залежати від властивостей даних та мети проведення кластеризації. Обчислення близькості на основі мінімума, відомого також як алгоритм Single-Linkage, визначає близькість двох кластерів C1 і C2 як мінімум відстаней між точками Pi і Pj, де Pi належить C1, а Pj належить C2.

Перевага підходу з обчисленням близькості на основі мінімума полягає в здатності ефективно виділяти нееліптичні форми, особливо при невеликому розриві між двома кластерами.

З іншого боку, недоліком цього методу є неспроможність правильно розділити кластери в разі наявності шуму.

Щодо обчислення близькості на основі максимума, відомого як алгоритм Complete-Linkage, його перевагою є ефективність при розділенні кластерів, особливо якщо між ними присутні шумові елементи. Проте недоліками цього методу є властивість бути упередженим до глобулярних кластерів та тенденція розбивати великі кластери.

У випадку обчислення близькості на основі групового середнього беруться всі можливі пари точок з різних кластерів, обчислюються їх близькості, а потім визначається середнє значення цих близкостей.

Перевагою цього методу є його ефективність при розділенні кластерів в умовах наявності шуму між ними. Однак недоліками є упередженість до глобулярних кластерів.

Обчислення близькості на основі відстані між центроїдами включає в себе розрахунок центроїдів кластерів C1 і C2, приймаючи відстань між їхніми центроїдами за міру близькості між кластерами. Цей метод є менш популярним в реальному світі. Обчислення близькості на основі методу Варда ідентичне обчисленню на основі групового середнього, за винятком того, що метод Варда використовує суму квадратів відстаней між точками Pi та Pj.

Перевага обчислення близькості на основі методу Варда полягає в його ефективності при розділенні кластерів, особливо в умовах наявності шуму між ними.

Недоліком є аналогічна упередженість до глобулярних кластерів, яка спостерігається і у випадку обчислення на основі групового середнього.

Складність за пам'яттю для ієрархічної кластеризації значно зростає при великій кількості точок даних, оскільки потрібно зберігати матрицю близькості в оперативній пам'яті.

Складність простору порядку квадрата n:

Складність за пам’яттю = $O(n^2)$,

де n – кількість точок даних.

Складність за часом для ієрархічної кластеризації значно зростає через необхідність виконання n ітерацій. На кожній ітерації обов'язково потрібно оновити матрицю близькості та відновити структуру кластерів, що також призводить до великої складності за часом. Оцінювана складність за часом становить порядок куба n, що вказує на значний обчислювальний обсяг при обробці великої кількості точок даних.

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.

Метод кластеризації на основі центроїдів визначає завдання через введення центрів кластерів, і кожен об'єкт присвоюється кластеру з найближчим центром.

Одним із найбільш визнаних методів цього підходу є метод k середніх (k-means), що є некерованим алгоритмом кластеризації. Процес алгоритму полягає у поділі набору даних на певну кількість кластерів (позначених як k кластерів).

Для початку роботи з алгоритмом k-means потрібно ініціалізувати k кластерних центроїдів. Сам алгоритм є ітераційним і включає два основних етапи: призначення кластерів і переміщення центроїдів.

На етапі призначення кластерів алгоритм обходить кожну точку даних і визначає, до якого кластера вона належить, вибираючи той, що має найближчий центроїд. Використовується Евклідова відстань для визначення близькості.

Етап переміщення центроїдів включає переміщення кожного центроїда в середнє розташування точок у відповідному кластері. Іншими словами, алгоритм обчислює середнє значення всіх точок у кластері і переміщує центроїд у це середнє розташування.

Цей ітеративний процес продовжується, доки кластери залишаються незмінними або доки не виконується яка-небудь інша умова зупинки. Кількість кластерів (k) може бути вибрана випадковим чином або шляхом надання конкретних початкових точок від користувача.

Алгоритм розбиває дані на k кластерів, навіть якщо k не є оптимальною кількістю кластерів для використання. Таким чином, користувачам потрібно

визначити, чи вони використовують правильну кількість кластерів. Одним із методів для цього є ліктьовий метод.

Ліктьовий метод включає в себе запуск кластеризації k-means на наборі даних для різних значень k. Для кожного значення k обчислюється сума квадратних помилок (SSE). Потім будується лінійна діаграма SSE для кожного значення k. Якщо діаграма має форму "ліктя", то "лікоть" на діаграмі вказує на оптимальне значення k. Мета полягає в обранні значення k, при якому SSE має невелике значення, і при цьому не зменшується із збільшенням k, оскільки SSE тенденційно зменшується до нуля при збільшенні k.

Крім того, алгоритм k-means може давати непокладені результати, якщо початкове розташування центроїдів не є оптимальним. Тому процес k-means повторюється кілька разів, і вибирається результат із найменшими кумулятивними варіаціями між кластерами.

Переваги алгоритму k-means включають його легкість у розумінні та реалізації, а також простоту масштабування.

Недоліки алгоритму k-means включають складність передбачення значення k та вибір оптимальних початкових позицій центроїдів.

Методи кластеризації можуть бути використані для розділення користувачів на групи на основі аналітичних профілів для поліпшення функціонування рекомендаційної системи.

2.1.2 Аргументація вибору базового методу кластеризації даних

Після ретельного аналізу існуючих методів кластеризації, які можна використовувати для вирішення визначеної задачі, можна виділити їх у три основні категорії:

– графові алгоритми кластеризації;
– статистичні алгоритми кластеризації;
– ієрархічні алгоритми кластеризації.

Залежно від алгоритмічної основи, можна виділити такі підкатегорії:

– адаптивні та неадаптивні;
– глобальні та децентралізовані;
– статичні та динамічні.

Також існують базові вимоги до вибору алгоритму, серед яких:

– стабільність;
– оптимальність;
– точність;
– простота;
– надійність;
– продуктивність.

Ці характеристики визначають фактори, які можуть впливати на подальший розвиток та вдосконалення нового алгоритму. При розробці та підтримці режиму швидкого розгортання рекомендовано базуватися на динамічних алгоритмах.

Динамічні алгоритми забезпечують сумісність із змінними мережами та адаптуються до змін стану мережі в залежності від конфігурацій та зовнішніх впливів. Також важливо вибрати алгоритм, який має найвищу продуктивність та одночасно враховує інші характеристики, такі як стабільність та точність. При цьому важливо, щоб ці характеристики не конфліктували між собою, забезпечуючи ефективну роботу алгоритму. До алгоритмів, які відповідають цим критеріям, входять такі, як k-means та алгоритми на основі теорії графів. Вони схожі за характеристиками, але відрізняються за своєю природою.

2.2 Дослідження існуючих методів побудови рекомендаційних систем

2.2.1 Поняття рекомендаційної системи

Рекомендаційні системи (РС) – це програми, які ставлять за мету передбачити, які об'єкти (такі як ігри, фільми, музика, книги, новини або веб-сайти) будуть цікаві користувачеві, використовуючи певну інформацію про його профіль.

Прогнозування може ґрунтуватися на даних про користувачів, об'єкти та раніше надані оцінки користувачами.

РС мають широке застосування як у комерційних цілях для визначення цільових груп користувачів та рекламних стратегій, так і в організації експертних оцінок, де вони допомагають суттєво зменшити навантаження на експертів.

РС працюють з наступними вихідними даними:

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

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

– ρ: $U \times I \to [0,1]$ – функція оцінки близькості об'єктів; значення $\rho(u, i)$ 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 $\rho(\mathrm{u}, \mathrm{i}) \leq \varepsilon_0 \in \varepsilon(0)$, 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 $\rho(\mathrm{u}_a, i_p) = \perp$ (символом $\perp$ будемо позначати невідоме значення) шляхом алгоритмічного обчислення значення прогнозної функції:

$$\bar{\rho}(u_a, i_p): U \times I \to [0,1], \qquad (2.1)$$

де i_p – прогнозований об'єкт; при цьому потрібно, щоб прогноз був складений точно, тобто $(|\rho)^{-}(u\_a, i\_p) - \rho(u\_a, i\_p)| \leq \varepsilon_0$;

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

$$I_{topN} = \{i: (u_a R_i) \wedge \rho(u_a, i) = \perp\} \wedge \left|I_{topN}\right| = N \qquad (2.2)$$

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:

$$u_a R_i \Leftrightarrow \bar{\rho}(u_a, i) \leq \varepsilon_0. \quad (2.3)$$

В рамках вирішення вказаних завдань рекомендаційні системи (РС) використовують аналіз інформації про характеристики користувачів і об'єктів. Позначимо безліч характеристик користувачів як X, прикладом можуть бути соціально-демографічні показники онлайн-магазину РС. Аналогічно, Y визначає безліч характеристик об'єктів, таких як назви жанрів ігор.

Значення характеристик користувачів визначаються через вагову функцію $w_U: U \times X \to [0,1]$, де U - множина користувачів. Аналогічно, для об'єктів використовується вагова функція $w_I: I \times Y \to [0,1]$, де I - множина об'єктів. Значення цих вагових функцій можуть бути визначені користувачами, експертами, алгоритмічно тощо.

Структуру даних, що містить інформацію про користувача u, і об'єкт i, називається контентом користувача $c_X(u)$, і контентом об'єкта $c_Y(i)$ відповідно.

Модель РС складається з:

$$(c_X; c_Y; \Pi) \quad (2.4)$$

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 $\rho(u, i) \in P_0$, 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).

2.2.2 Контентна фільтрація

Контентна фільтрація генерує рекомендації, базуючись на характеристиках рекомендованих об'єктів і відношенні користувача до цих характеристик. Таким чином, на основі попередніх оцінок формуються вподобання стосовно певних характеристик, які використовуються для майбутніх рекомендацій. Наприклад, якщо користувач високо оцінює екшн-ігри та стратегії, а низько - симулятори, то на основі цих вподобань можна зробити висновок, які ігри з якими характеристиками ймовірніше задовольнять користувача.

Цей підхід ґрунтується на порівнянні характеристик товару (контенту) з вподобаннями користувача, які формуються на основі його історії оцінок або покупок. Чим більше товар відповідає цим вподобанням, тим вище ймовірність того, що він сподобається користувачу і тим вищий рівень рекомендованості цього товару. Очевидно, що цей підхід передбачає наявність характеристик для всіх товарів.

Контентна фільтрація використовується для рекомендацій зазвичай у випадках, коли предметами є товари з неструктурованими характеристиками, такими як фільми, ігри, музика і т. д. Ці характеристики можуть включати в себе відгуки, описи, рецензії тощо. Однак, крім неструктурованих, для контентної фільтрації можна використовувати і числові та категоріальні характеристики, такі як жанри, теги тощо.

Для опису неструктурованих ознак часто використовують вектори в просторі слів, які складаються з елементів-ознак, що потенційно характеризують інтерес користувача. Також, продукти можуть бути представлені векторами в тому ж просторі. Коли користувач взаємодіє з системою (наприклад, купує ігри), векторні описи придбаних товарів об'єднуються в єдиний вектор, який є вектором інтересів користувача.

Для надання рекомендацій використовують алгоритми пошуку n найближчих сусідів, де вектор інтересів користувача порівнюється з векторами товарів. Зазвичай в якості міри близькості використовується косинусна відстань:

$$sim(A,B) = \cos(\theta) = \frac{A*B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i * B_i}{\sqrt{\sum_{i=1}^{n} (A_i{}^2)} * \sqrt{\sum_{i=1}^{n} (B_i{}^2)}} \qquad (2.5)$$

де A – вектор інтересів першого користувача, B – вектор інтересів другого користувача.

2.2.3 Колаборативна фільтрація

Колаборативна фільтрація використовує дані про інших користувачів та їх відношення до об'єктів для надання рекомендацій щодо неоцінених об'єктів. Цей підхід базується на припущенні, що користувачі з схожими історіями оцінок мають подібні смаки і схоже оцінюватимуть об'єкти в майбутньому.

Існують три типи колаборативної фільтрації: модельний підхід, підхід на основі сусідства та гібридний. Модельний підхід використовує алгоритми машинного навчання для передбачення оцінок користувачів на основі матриці оцінок. Підхід на основі сусідства визначає рекомендації, знаходячи схожі стовпці або рядки в матриці оцінок. Гібридний підхід поєднує обидва підходи.

Хоч гібридна колаборативна фільтрація може бути ефективною та точною, вона вимагає більше часу та складнощів у виконанні та реалізації.

Алгоритм колаборативної фільтрації у класичному вигляді використовує принцип n найближчих сусідів, де бракуюча інформація про користувача доповнюється даними його n найближчих сусідів – інших користувачів з схожими вподобаннями. Схожість визначається на основі кореляції інтересів.

Недоліком цього підходу є його квадратична складність, оскільки потрібно розраховувати відстані між усіма парами користувачів, що може бути неефективним при великій кількості користувачів. Складність за пам'яттю для зберігання матриці відстаней також зростає квадратично, що робить його важко застосовуваним на практиці при великих обсягах даних.

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.

Рекомендаційні системи (РС), що використовують колаборативну фільтрацію як основне правило (означене як П), поділяються на два типи в залежності від множини, яку вони фільтрують: множину користувачів та множину об'єктів. Перші називаються суб'єктно-орієнтованими РС (СОК), а другі – об'єктно-орієнтованими РС (ООК).

Теорія, на якій ґрунтуються колаборативні системи, передбачає побудову рішень на основі навчальної множини та оцінку якості цих рішень за допомогою тестової множини. Правило П СОК базується на припущенні, що, якщо користувачі були схожі за вподобаннями в минулому, то вони залишатимуться схожими в майбутньому. У термінології це припущення виражається наступним чином:

$u_a R_u$u виконується на $P_0 \Rightarrow u_a R_u u$ виконується на $P_\perp$ (2.6)

де u_a – активний користувач,

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

P_0 – навчальна множина даних,

$P_\perp$ – тестова множина даних,

R_u – відношення близькості користувачів.

Для встановлення відносини близькості R_u між користувачами в колаборативних рекомендаційних системах (КРС), основаних на суб'єктно-орієнтованому підході (СОК), використовуються значення характеристик користувачів. У випадку СОК об'єкти завжди виступають характеристиками, а вагові значення $\rho(u,i) \in P_0$, що визначаються самими користувачами і відображають їхні вподобання, використовуються для визначення близькості. Для цього застосовуються так звані міри близькості:

$$\delta_u: U \times U \rightarrow [0,1]: \left(1 - \delta_u(u,v)\right) \leq \varepsilon_0 \Leftrightarrow uR_u v \qquad (2.7)$$

Користувачі, між якими виконується відношення близькості, називаються сусідами.

Правило П СОК задається формулою:

$$u \in U, (u_a R u) \Rightarrow \left|\bar{\rho}(u_a, i_p) - \rho(u_a, i_p)\right) \leq \varepsilon_0,$$
$$\rho(u_a, i_p) = f(\{\rho(u, i_p)\}) \qquad (2.8)$$

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 28 $\rho(u_a, i_p)$ можна функціонально визначити по значенням $\{\rho(u, i_p)\}$, тобто прогнозна функція є функцією від значень оцінок близькості сусідів.

Правило П для об'єктно-орієнтованих колаборативних рекомендаційних систем (ООК) ґрунтується на такому твердженні: якщо користувач оцінює позитивно об'єкт i, який схожий за характеристиками на об'єкт j, то користувачеві також сподобається об'єкт j. У введеній термінології це твердження можна виразити так:

$$(u_a R i) \wedge (i R_i j) \Rightarrow u_a R j \qquad (2.9)$$

де R_i– відношення близькості об'єктів.

Відношення близькості R_i між об'єктами встановлюється РС на підставі значень мір близькості: $1 - \delta_i(i,j) \leq \varepsilon_0 \Leftrightarrow iR_i j, \delta_i : I \times I \rightarrow [0,1]$ – 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 $(u_a Ri_0)$, $(u_a Ri_\perp)$, тому будемо вважати, що $P = \{\rho(u,i): uRi\}$ для завдання $topN$.

Правило П ООК задається формулою:

$$(iR_i\ i_0\ )) \Rightarrow (\bar{\rho}(u_a, i) = 0) \Rightarrow u_a\ Ri. \qquad (2.10)$$

Значення $\bar{\rho}(u_a, i)$ встановлюються на нуль, щоб забезпечити те, що об'єкти i вважатимуться близькими для активного користувача при будь-якому значенні порогу ε_0.

Правило виводу для об'єктно-орієнтованих колаборативних рекомендацій говорить про те, що якщо існує об'єкт i, який є сусідом об'єкта i_0, то, відповідно до евристичного твердження, $u_a Ri$, оскільки $u_a Ri_0$ відповідає прийнятому для задачі виду вихідної множини.

2.2.4 Порівняння методів побудови рекомендаційних систем

Враховуючи обговорені недоліки та переваги колаборативної фільтрації порівняно з контентною, можна визначити такі аспекти:

Недоліки колаборативної фільтрації:

1) необ'єктивні та упереджені оцінки: оцінки, надані недобросовісними користувачами, можуть спричинити несправедливі рекомендації та погіршити якість обслуговування для інших користувачів;

2) проблема «білих ворон»: унікальні користувачі зі специфічними уподобаннями можуть отримувати менш якісні рекомендації через недостатню репрезентацію їх уподобань у великій масі користувачів;

3) конфіденційність: збір та обробка особистих даних для колаборативної фільтрації може порушити приватність користувачів, що може викликати негативну реакцію;

4) складні обчислення: реалізація колаборативної фільтрації може вимагати значних обчислювальних ресурсів та часу, особливо при збільшенні кількості користувачів та об'єктів.

Переваги колаборативної фільтрації:

1) реакція на «холодний старт»: колаборативна фільтрація добре працює при виникненні нових об'єктів або користувачів, допомагаючи швидко покращувати якість рекомендацій;

2) швидка адаптація до змін вподобань: система колаборативної фільтрації швидко адаптується до зміни вподобань користувачів, забезпечуючи актуальні рекомендації;

3) не потребує описових характеристик об'єктів: для генерації рекомендацій колаборативна фільтрація не вимагає детальних описів характеристик об'єктів, опираючись на взаємодію користувачів.

У зв'язку з вищезазначеним, вибір колаборативної фільтрації на основі моделі для проекту обгрунтований її високою точністю та здатністю швидко адаптуватися до змін в уподобаннях користувачів.

Таблиця 2.1 – Порівняння контентної та колаборативної фільтрації

Аспект	Контентна фільтрція	Колаборативна фільтрація
Об'єктивність та упередженість	Точність рекомендацій залежить від якості описових характеристик об'єктів.	Уразливі до необ'єктивних або недобросовісних оцінок користувачів.
Проблема «білих ворон»	Можливість точнішого врахування специфічних уподобань користувачів через детальний опис об'єктів.	Унікальні користувачі з унікальними уподобаннями можуть отримувати менш якісні рекомендації.

Продовження таблиці 2.1

Аспект	Контентна фільтрція	Колаборативна фільтрація
Конфіденційність	Не потребує збору особистих даних користувачів для генерації рекомендацій.	Може порушити приватність користувачів через збір та обробку їх особистих даних.
Складність обчислень	Легко виконується без значного обчислювального навантаження.	Може вимагати значних обчислювальних ресурсів та часу, особливо зі збільшенням кількості користувачів та об'єктів.
Реакція на «холодний старт»	Може виявити ефективність при виникненні нових об'єктів або користувачів.	Вимагає великої кількості користувачів та об'єктів для точних рекомендацій.
Адаптація до змін вподобань	Залежить від оновлення даних та перегенерації моделі зміни уподобань користувачів.	Швидко адаптується до зміни уподобань користувачів без необхідності перегенерації моделі.
Вимога описових характеристик	Вимагає детальних описів характеристик об'єктів для генерації рекомендацій.	Не потребує детальних описів об'єктів, опирається на взаємодію користувачів.

У таблиці 2 видно, що контентна фільтрація має переваги у великій кількості аспектів, зокрема у об'єктивності, конфіденційності та складності обчислень. Вона не потребує оцінок користувачів та не порушує їх приватність, що робить її більш

привабливою для проекту з обмеженим доступом до особистих даних користувачів.

2.3 Моделювання кластеризації для розбиття ігр на групи

Рекомендаційна система, яку ми розробляємо у цьому проекті, використовує кластеризацію ігор для підвищення ефективності та покращення якості рекомендацій. Характеристики ігор повинні бути виражені числовими показниками, представленими масивом чисел від 1 до n після нормування. Для кластеризації таких даних найчастіше використовується алгоритм k-means.

Вказаний алгоритм розподіляє безліч вхідних об'єктів на визначену кількість кластерів. Початкова робота алгоритму включає ініціалізацію центрів кластерів (центроїдів):

$$\mu_1 , \mu_2 , \dots , \mu_k \in R^n . \qquad (2.14)$$

де k – кількість кластерів,

n – розмірність кожного об'єкта.

Кожен елемент множини призначається кластеру з найближчим центром. Далі цей процес повторюється до тих пір, поки алгоритм не досягне конвергенції.

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

$$c^{(1)} = \{p: \ \| x^{(i)} - \mu_p \|^2 \leq \| x^{(i)} - \mu_j \|^2 \ \ \forall j, 1 \leq j \leq K\} \qquad (2.15)$$

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 (2.15) 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:

$$\mu_k = \frac{1}{s_k} \sum_{x(j):C(j)=k} x^{(j)} \tag{2.16}$$

Згідно з формулою (2.16), кожному центру кластера k, до якого належать вхідні об'єкти, присвоюються значення центру мас безлічі вхідних об'єктів. Якщо положення центрів кластерів не змінюється після поточної ітерації, алгоритм вважається завершеним.

Проте алгоритм k-means має кілька недоліків:

– кількість кластерів повинна бути визначена заздалегідь;

– немає гарантії досягнення глобального мінімуму;

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

Для подолання цих недоліків розглянемо підходи до мінімізації. Введемо функцію втрат для оцінювання якості розбиття.

$$F\left(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_k \right) = \frac{1}{m} \sum_{i=1}^{m} \| x^{i} - \mu_{c^{(i)}} \|^2 \tag{2.17}$$

Наведений функціонал обчислює суми квадратичних відстаней від центрів кластерів до об'єктів, що відповідають цим кластерам. Метою оптимізації кластеризації є мінімізація цього функціоналу.

Розглянемо перший недолік - число кластерів визначається заздалегідь. Метод ліктя може бути використаний для мінімізації цього недоліку. Розробник

обирає кількість кластерів і оцінює зменшення функції втрат при збільшенні кількості кластерів.

Другий недолік - відсутність гарантії досягнення глобального мінімуму. Множинна ініціалізація може бути використана для мінімізації цього недоліку. Алгоритм k-means ініціалізується різними наборами центроїдів, і для кожного набору оцінюється значення функції втрат. Обирається той варіант, при якому функція втрат має найменше значення.

Отже, запропонований і описаний метод побудови рекомендаційної системи використовує k-means кластеризацію для розбиття ігри на групи та контентну фільтрацію для надання рекомендацій.

2.4 Планування експериментального дослідження

Критично важливою задачею при розробці рекомендаційної системи є оцінка якості рекомендацій. Для вимірювання ефективності рекомендаційної системи та параметрів її якості використовуються різноманітні метрики. Однак у випадку кластеризації ігор, де завдання полягає в групуванні подібних ігор разом, більш відповідною метрикою є Silhouette Score.

Silhouette Score враховує ступінь схожості кожного об'єкта зі своїм власним кластером порівняно з іншими кластерами. Високий Silhouette Score свідчить про те, що об'єкти добре відокремлені від сусідніх кластерів, що є бажаним результатом для кластеризації ігор.

Ця метрика розраховується за допомогою наступної формули:

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \quad (2.18)$$

де *S(i)* – Silhouette Score для об’єкта *i*,

a(i) – середня відстань від об’єкта *i* до інших об’єктів у тому ж кластері,

$b(i)$ – середня відстань від об’єкта i до об’єктів у сусідньому кластері.

Метрика Silhouette Score, хоча і корисна для вимірювання ступеня схожості об'єктів у кластері та їх віддаленості від сусідніх кластерів, не завжди є ідеальним показником для оцінки якості рекомендацій.

Несприятливі умови, такі як великий розкид даних, можуть призвести до неправильних висновків при використанні Silhouette Score. Варто враховувати, що ідеальне значення Silhouette Score не завжди відображає повну якість ранжування та персоналізованих рекомендацій.

Незважаючи на обмеження. метрика Silhouette Score залишається широко використовуваною в галузі рекомендаційних систем і залишається однією з основних для оцінки ефективності роботи таких систем.

Davies-Bouldin Index є іншою метрикою, яка враховує якість кластеризації і може доповнювати Silhouette Score у вимірюванні ефективності рекомендаційної системи для ігор.

Davies-Bouldin Index оцінює середню відстань між кожним кластером та кластером, який найбільше схожий на нього.

Високе значення Davies-Bouldin Index вказує на те, що об'єкти добре відокремлені від сусідніх кластерів, що є бажаним результатом для кластеризації ігор. Визначається ця метрика наступним чином:

$$DBI = \frac{1}{k}\sum_{i=1}^{k} max_{j \neq i}\left(\frac{S_i + S_j}{M_{ij}}\right) \quad (2.19)$$

де:

k - кількість кластерів,

S_i- внутрішній критерій схожості (середня відстань від об'єкта до інших об'єктів у тому ж кластері),

M_{ij}- міра схожості між кластерами i та j.

Davies-Bouldin Index приймає значення від 0 до нескінченності, де менше значення вказує на кращу кластеризацію.

Також, подібно до Silhouette Score, важливо враховувати, що ця метрика має свої обмеження та не завжди є ідеальним показником для оцінки якості рекомендацій.

Обмеження, такі як великий розкид даних, можуть впливати на вірогідність неправильних висновків при використанні Davies-Bouldin Index.

Calinski-Harabasz Index - метрика, яка використовується для оцінки якості кластеризації. Вона вимірює відношення між дисперсією всередині кластерів та дисперсією між кластерами, допомагаючи визначити оптимальну кількість та якість кластерів.

Індекс розраховується за наступною формулою:

$$CHI = \frac{B(k)}{W(k)} \times \frac{N-k}{k-1} \quad (2.20)$$

де:

– *B(k)* - середня відстань між центроїдами кластерів,

– *W(k)* - загальна внутрішньокластерна дисперсія,

– *N* - загальна кількість об'єктів у вибірці,

– *k* - кількість кластерів.

Високе значення Calinski-Harabasz Index вказує на кращу кластеризацію, коли внутрішні кластери компактні, а вони розташовані далеко один від одного.

Метрика Calinski-Harabasz Index, подібно до попередніх, враховує якість розташування об'єктів у кластерах і їх віддаленість від інших кластерів.

Однак, також як і інші метрики, вона має свої обмеження та вразливість до особливостей даних, що може вплинути на точність її використання в оцінці рекомендаційних систем для ігор.

Таблиця 2.2 – Метрики кластеризації

Метрика	Формула
Silhouette Score	$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$
Davies-Bouldin Index	$DBI = \frac{1}{k}\sum_{i=1}^{k} max_{j \neq i}(\frac{S_i + S_j}{M_{ij}})$
Calinski-Harabasz Index	$CHI = \frac{B(k)}{W(k)} \times \frac{N-k}{k-1}$

2.5 Характеристика набору даних

В останні роки відзначається раптова диверсифікація світу відеоігор, що призвело до значного зростання кількості учасників онлайн спільнот та різноманітності ігор. Ця збільшена доступність і різноманіття створює виклики для рекомендаційних систем, які використовують інформацію від платформ продажу ігор, маючи на меті врахувати тенденції випуску нових ігор щороку.

На сучасному етапі електронні ігри займають важливе місце в галузі розваг, привертаючи увагу багатьох користувачів. Причини для придбання та гри в ігри різноманітні, починаючи від заповнення вільного часу до спроб взаємодії з іншими гравцями, знаходження нових друзів чи втілення фантазійних образів.

За останні роки електронні ігри стали не тільки формою розваг, але й засобом заробітку для деяких гравців, які вдосконалюють та продавати свої ігрові досягнення.

Цей світ відділяється від реальності, маючи свої власні правила та можливості взаємодії.

Існує безліч класифікацій електронних ігор, і різні принципи можуть лежати в основі цих класифікацій. За жанром комп’ютерні ігри можна розділити на:

– рольові ігри (RPG): гравець керує вигаданим персонажем або персонажами, які виконують завдання в уявному світі;

– стратегії: ці ігри зосереджуються на вмілому мисленні та плануванні для

досягнення перемоги;

– екшен/шутер від першої особи: це ігри, які базуються на фізичних викликах і вимагають від гравця координації "руки-очі" та швидкої реакції;

– симулятори: ігри, які намагаються імітувати різні дії з реального життя для різних цілей, таких як навчання, аналіз або прогнозування;

– карточні ігри-стратегії: ігри, які поєднують в собі елементи стратегії та використання гральних карт;

– квести: гравець бере на себе роль головного героя в інтерактивній історії, яка керується дослідженнями або вирішенням головоломок.

За платформами ігри поділяються на:

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

– консольні ігри: ігрові консолі є менш складними у використанні порівняно з ПК. Гравці зазвичай використовують контролери для гри в консольні ігри;

– мобільні ігри: остання тенденція в іграх - це посилений розвиток ігор для мобільних телефонів та планшетів. Зараз більшість людей мають смартфони з ігровими можливостями;

– браузерні ігри: це ігри, які можна грати просто використовуючи веб-браузер.

На сучасний момент Інтернет став легкодоступним для осіб будь-якого віку, що призвело до загального попиту на браузерні ігри. Окрім цього, існує велика кількість видавців ігор - як нових, які лише починають здобувати увагу гравців (наприклад, Amazon Game Studios, 11 bit studios), так і вже визнаних, що мають своє місце в ігровій спільноті (такі як Electronic Arts, Bethesda Softworks). Зазвичай ці видавці представляють ігри різних жанрів та для різних геймінгових платформ.

Ця класифікація дозволяє ідентифікувати різні сегменти споживачів комп'ютерних ігор. Наприклад, любителі історичної та військової техніки, в

основному чоловіки, обирають симулятори; підлітки чоловічої статі часто віддають перевагу екшенам і шутерам; рольові ігри привертають увагу тих, хто цікавиться міжособистісною комунікацією та взаємодією з іншими гравцями; любителі головоломок обирають квести, а широкий спектр споживачів охоплює стратегічні ігри, включаючи економічні, військові та адміністративні.

Сегментація за типами гри також визначається платформою. Гравці на ПК цінують якість візуальних ефектів та продуктивність. Консольні гравці оцінюють баланс між якістю та простотою використання. Мобільні ігри відповідають тим, хто любить грати, але має обмежений час або ресурси для "великих" ігор. Браузерні ігри привертають тих, хто шукає простоту та доступність.

Зараз масова кастомізація набуває надзвичайної популярності. Сучасні рекомендаційні системи, які базуються на колаборативній фільтрації та контентній фільтрації, використовують різноманітні джерела інформації для створення персоналізованих рекомендацій.

Контентна фільтрація забезпечує рекомендації на основі особистих уподобань користувачів стосовно властивостей продукту. В цьому випадку система аналізує характеристики товарів чи контенту, які вже сподобалися користувачеві, і надає рекомендації на основі подібних властивостей.

Колаборативна фільтрація, навпаки, імітує рекомендації від користувача до користувача. Вона враховує вподобання користувачів як лінійну зважену комбінацію уподобань інших користувачів. Це означає, що система аналізує, які товари чи контент сподобалися іншим користувачам зі схожими вподобаннями і рекомендує подібні елементи користувачеві.

Обидві ці стратегії дозволяють створювати персоналізовані рекомендації, але базуються на різних принципах аналізу інформації про користувачів.

Набір даних містить дані ігор з платформи Steam. Кожен рядок має унікальний AppID і зазвичай є окремим випуском, за винятком деяких перевипусків і оновлених версій.

Набір даних містить дані зі 27075 ігор. Основна мета цього набору даних - аналіз та кластеризація ігор на основі різних факторів, щоб надавати рекомендації гравцям.

Набір даних містить наступні стовпці:

– appid - унікальний ідентифікатор для кожного заголовка;
– name - назва програми (гри);
– release_date -дата випуск;
– english - підтримка мов: 1 якщо англійська;
– developer - ім'я розробника;
– publisher – ім’я видавця;
– platforms - список підтримуваних платформ, розділених крапками з комами. Щонайбільше включає: windows;mac;linux;
– required_age -мінімальний необхідний вік відповідно до стандартів PEGI UK. Багато з 0 не мають рейтингу;
– categories - відокремлений крапкою з комою список категорій ігор;
– genres - перелік ігрових жанрів;
– steamspy_tags - список найпопулярніших тегів Steamspy, схожих на жанри;
– achievements - кількість досягнень у грі;
– positive_ratings - кількість позитивних оцінок;
– negative_ratings - кількість негативних оцінок;
– average_playtime - середній час гри користувача;
– median_playtime - медіанний час гри користувача;
– owners - орієнтовна кількість власників;
– price - поточна повна ціна власності у фунтах стерлінгів.

За допомогою інтелектуальної системи кластеризації, такої як K-Means, ці ігри можуть бути об'єднані в кластери або групи на основі схожості в їх характеристиках.

Це дозволить розробити персоналізовану систему рекомендацій, яка пропонує гравцям ігри, схожі на ті, які їм сподобалися раніше. Для коректної роботи програми та надання рекомендацій необхідно, щоб набір даних містив інформацію про комп'ютерні ігри, а також характеристики цих ігор.

Таблиця 3.1 – Опис вимог до набору даних

Вимоги до набору даних	Опис
Інформація про ігри	Назва гри, жанр, рік випуску, розробник, видавець та інші пов'язані атрибути.
Відгуки користувачів	Кількість позитивних та негативних відгуків, загальна оцінка гри, рейтинг ігор та інші параметри, що відображають популярність та якість ігор.
Характеристики ігор	Ціна, кількість гравців, тип гри (одиночна гра, мережева гра, онлайн-гра тощо), доступна мова та інші характеристики.
Теги та категорії	Теги або категорії, які відображають основні особливості або тематику ігор.
Унікальні ідентифікатори	Унікальні ідентифікатори для кожної гри, що дозволяють ідентифікувати окремі ігри та їх відповідність до користувачів.

2.6 Опис алгоритму надання рекомендацій

Перед тим як приступати до створення рекомендаційної системи, важливо ретельно розглянути та розробити алгоритм, який буде використовуватися для генерації рекомендацій для користувачів. Алгоритм надання рекомендацій включає в себе кроки, які наведені нижче.

Крок 1. Очищення та оптимізація даних: перш за все, проводимо аналіз та очистку набору даних:

Крок 2. Видаляємо ігри з малою кількістю відгуків, щоб мати надійні дані для кластеризації.

Крок 3. Перевіряємо та оптимізуємо формати дат та числових значень для подальшої зручності обробки.

Крок 4. Визначення оптимальної кількості кластерів: використовуємо методи кластерного аналізу, такі як метод ліктя, для визначення оптимальної кількості кластерів.

Крок 5. Кластеризація ігор: застосовуємо алгоритм кластеризації, такі як K-Means та ієрархічну кластеризацію до наших даних. Кожна гра буде відзначена певним кластером.

Крок 6. Вибір гри: користувач обирає конкретну гру. Система знаходить цю гру в кластері і далі аналізує інші ігри у цьому кластері.

Крок 7. Рекомендації: система надає рекомендації, базуючись на грі, обраній користувачем, і інших іграх у тому ж кластері. Рекомендації можуть бути надані наступними способами:

– схожість за тегами: Враховуючи теги гри, рекомендуємо ігри з аналогічними тегами у тому ж кластері;

– популярність: Рекомендації можуть базуватися на популярності ігор у вибраному кластері;

– загальні критерії: Враховуючи різні характеристики гри, такі як жанр, розробник, видавець тощо, система може рекомендувати ігри, які спільні за цими критеріями.

Крок 8. Візуалізація результатів: для зручності буде створена графічна візуалізація кластерів та рекомендацій, щоб користувач міг легко розуміти, які ігри входять до одного кластеру та чому вони рекомендовані.

Цей підхід може допомогти користувачам знаходити нові ігри, які мають спільні риси з тими, що їм вже сподобалися, і робити більш персоналізований вибір.

На рисунку 2.1 наведено блок-схему роботи програми. За допомогою цієї блок-схеми можна краще зрозуміти основні операції програмної реалізації.

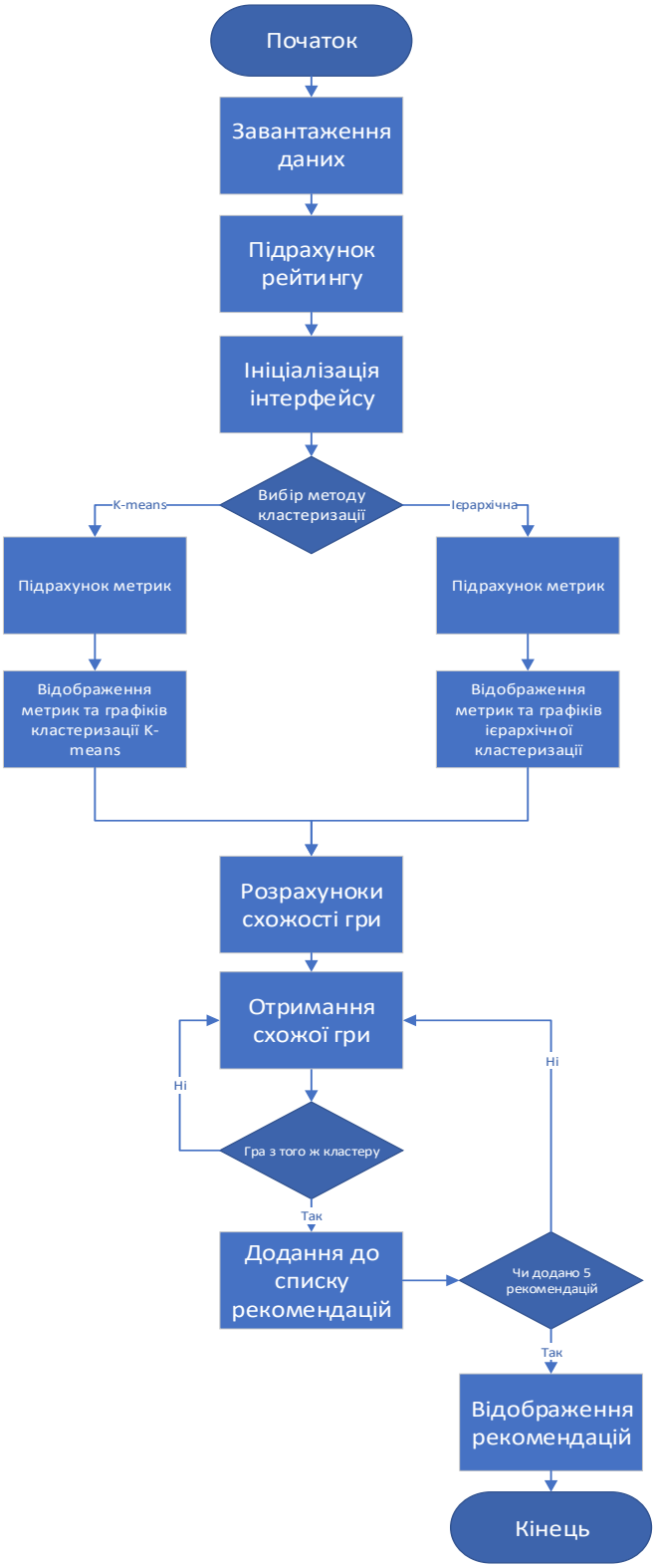


Рисунок 2.1 – Блок-схема роботи програми

Висновки до розділу 2

У даному розділі було проведено дослідження існуючих методів кластеризації та побудови рекомендаційних систем з метою підготовки до реалізації власної інтелектуальної системи кластеризації у комп'ютерних іграх для персоналізованого контенту.

У розділі було розглянуто різноманітні методи кластеризації, такі як K-means та ієрархічна кластеризація, а також методи побудови рекомендаційних систем, зокрема контентна і колаборативна фільтрація. Проведено аналіз переваг та недоліків кожного з методів, їхній придатності для конкретного завдання побудови рекомендаційних систем у галузі комп'ютерних ігор.

На основі проведеного дослідження було обрано базовий метод кластеризації даних, який враховує особливості даних та відповідає поставленій меті системи. Також було розроблено модель вдосконаленої рекомендаційної системи, яка використовує контентну фільтрацію та враховує результати кластеризації для покращення рекомендацій.

Для подальшого дослідження та реалізації був сформований план експериментального дослідження, а також описано базу даних та алгоритм надання рекомендацій у контексті інтелектуальної системи кластеризації у комп'ютерних іграх.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Під час реалізації програми використовується мова програмування Python та бібліотеки для аналізу даних, такі як pandas, scikit-learn, та matplotlib. Програма пропонує користувачу вибрати метод кластеризації (K-means або ієрархічну кластеризацію) та надає можливість аналізу метрик якості кластеризації.

3.1 Опис користувацького інтерфейсу та підготовка даних

Для реалізації системи рекомендацій використовується користувацький інтерфейс, що дозволяє вибирати метод кластеризації, відображати метрики та графіки для аналізу результатів кластеризації, а також шукати схожі ігри на основі обраної гри.

Основні кроки підготовки даних наведені нижче.

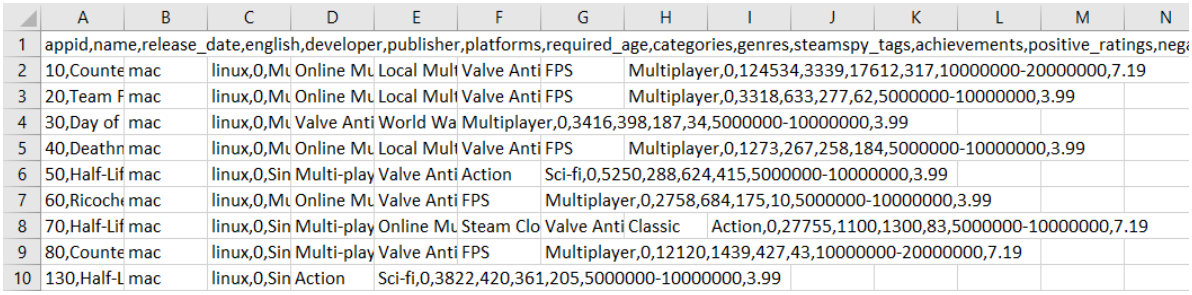
Крок 1. Дані завантажуються з CSV-файлу «steam.csv».

Крок 2. Відбираються тільки ті дані, у яких кількість позитивних оцінок перевищує 1000.

Крок 3. Обчислюється рейтинг для кожної гри на основі кількості позитивних та негативних оцінок.

Крок 4. Визначаються ознаки («features») для подальшої кластеризації, що включають ціну та рейтинг гри.

На рисунку 3.1 можна побачити фрагмент набору даних, який було використано для кластеризації даних та надання рекомендацій.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	appid,name,release_date,english,developer,publisher,platforms,required_age,categories,genres,steamspy_tags,achievements,positive_ratings,nega													
2	10,Counte	mac	linux,0,Mu	Online Mu	Local Mult	Valve Anti	FPS	Multiplayer,0,124534,3339,17612,317,10000000-20000000,7.19						
3	20,Team F	mac	linux,0,Mu	Online Mu	Local Mult	Valve Anti	FPS	Multiplayer,0,3318,633,277,62,5000000-10000000,3.99						
4	30,Day of	mac	linux,0,Mu	Valve Anti	World Wa	Multiplayer,0,3416,398,187,34,5000000-10000000,3.99								
5	40,Deathn	mac	linux,0,Mu	Online Mu	Local Mult	Valve Anti	FPS	Multiplayer,0,1273,267,258,184,5000000-10000000,3.99						
6	50,Half-Lif	mac	linux,0,Sin	Multi-play	Valve Anti	Action	Sci-fi,0,5250,288,624,415,5000000-10000000,3.99							
7	60,Ricoch	mac	linux,0,Mu	Online Mu	Valve Anti	FPS	Multiplayer,0,2758,684,175,10,5000000-10000000,3.99							
8	70,Half-Lif	mac	linux,0,Sin	Multi-play	Online Mu	Steam Clo	Valve Anti	Classic	Action,0,27755,1100,1300,83,5000000-10000000,7.19					
9	80,Counte	mac	linux,0,Sin	Multi-play	Valve Anti	FPS	Multiplayer,0,12120,1439,427,43,10000000-20000000,7.19							
10	130,Half-L	mac	linux,0,Sin	Action	Sci-fi,0,3822,420,361,205,5000000-10000000,3.99									

Рисунок 3.1 – Фрагмент набору даних

Користувацький інтерфейс (UI) наведений на рисунку 3.2 був створений за допомогою бібліотеки Tkinter, яка є стандартною бібліотекою для роботи з графічним інтерфейсом у Python. Tkinter надає простий спосіб створення віконних додатків з різноманітними елементами інтерфейсу, такими як кнопки, мітки, випадаючі списки тощо.

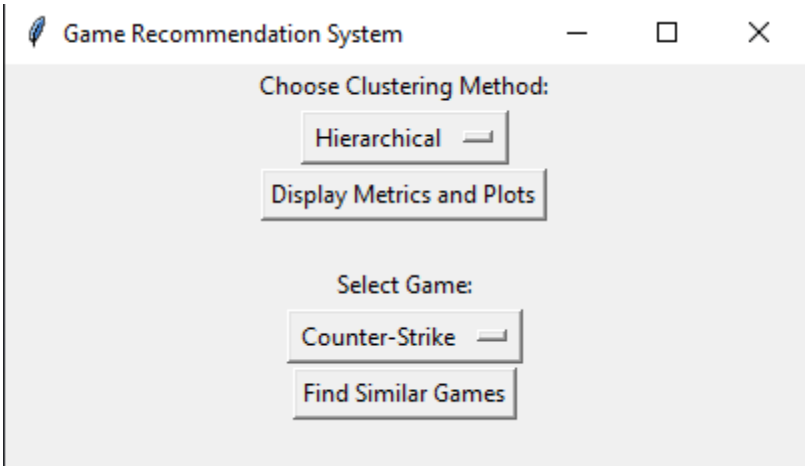


Рисунок 3.2 – Користувацький інтерфейс застосунку

UI містить наступні елементи:

– вікно програми з заголовком "Game Recommendation System";

– вибір методу кластеризації з випадаючим списком;

– для відображення метрик та графіків є кнопка "Display Metrics and Plots";

– користувач може обрати гру з випадаючого списку, використовуючи надпис "Select Game";

– щоб знайти схожі ігри, користувач натискає кнопку "Find Similar Games".

Якщо кластеризація ще не проводилася у застосунку, елементи UI "Select Game", та "Find Similar Games" не показуються, щоб уникнути появи помилок.

3.2 Опис логіки надання рекомендацій

Рекомендаційна система була створена за допомогою бібліотеки scikit-learn. Бібліотека scikit-learn (Sklearn) - це відкрита бібліотека машинного навчання для мови програмування Python. Вона надає широкий спектр інструментів для роботи з класичними і сучасними алгоритмами машинного навчання, включаючи класифікацію, регресію, кластеризацію, а також валідацію моделей, обробку даних і підготовку їх до аналізу.

Перед кластеризацією необхідно визначитися з ознаками для кластеризації. Для прикладу було обрано такі ознаки як рейтинг та ціна. Рейтинг у даному випадку – відсоток позитивних відгуків від загальної кількості. Для підрахунку рейтингу кожної гри було створено окремий метод calculate_rating(row). Логіка методу наведена на рисунку 3.3.

У даному методі, параметр «row» - це кожен окремий рядок або запис у наборі даних, який передається функції як аргумент при її виклику. Коли функція `apply` використовується разом із набором даних, вона застосовується до кожного рядка окремо. У цьому випадку «row» - це кожний рядок у наборі даних, який містить дані про ігри, такі як кількість позитивних і негативних відгуків. Метод «calculate_rating» обчислює рейтинг гри, використовуючи кількість позитивних та негативних відгуків у кожному рядку.

```
def calculate_rating(row):
    return row['positive_ratings'] / (row['positive_ratings'] + row['negative_ratings'])
```

Рисунок 3.3 – Метод calculate_rating(row)

Логіка рекомендаційної системи складається з блоку кластеризації та блоку надання рекомендацій. Логіка блоку кластеризації міститься у методі Clustering(), та наведена на рисунку 3.4.

```
def Clustering():
    plt.figure(figsize=(15, 5))

    if clustering_method.get() == 'K-means':
        # K-means
        kmeans = KMeans(n_clusters=4)
        df['kmeans_cluster'] = kmeans.fit_predict(df[features])

        # Metrics
        silhouette = silhouette_score(df[features], kmeans.labels_)
        db_index = davies_bouldin_score(df[features], kmeans.labels_)
        ch_index = calinski_harabasz_score(df[features], kmeans.labels_)
        metrics_info = f"Silhouette Score: {silhouette:.4f}\nDavies-Bouldin Index: {db_index:.4f}\nCalinski-Harabasz Index: {ch_index:.4f}"

        # Silhouette Score
        plt.subplot(1, 3, 1)
        silhouette_scores = []
        for k in range(2, 11):
            kmeans = KMeans(n_clusters=k)
            kmeans.fit(df[features])
            silhouette_scores.append(silhouette_score(df[features], kmeans.labels_))
        plt.plot(range(2, 11), silhouette_scores, marker='o')
        plt.xlabel('Number of clusters')
        plt.ylabel('Silhouette Score')
        plt.title('Silhouette Score for KMeans')
```

```
        # Davies-Bouldin Index
        plt.subplot(1, 3, 2)
        db_scores = []
        for k in range(2, 11):
            kmeans = KMeans(n_clusters=k)
            kmeans.fit(df[features])
            db_scores.append(davies_bouldin_score(df[features], kmeans.labels_))
        plt.plot(range(2, 11), db_scores, marker='o')
        plt.xlabel('Number of clusters')
        plt.ylabel('Davies-Bouldin Index')
        plt.title('Davies-Bouldin Index for KMeans')

        # Calinski-Harabasz Index
        plt.subplot(1, 3, 3)
        ch_scores = []
        for k in range(2, 11):
            kmeans = KMeans(n_clusters=k)
            kmeans.fit(df[features])
            ch_scores.append(calinski_harabasz_score(df[features], kmeans.labels_))
        plt.plot(range(2, 11), ch_scores, marker='o')
        plt.xlabel('Number of clusters')
        plt.ylabel('Calinski-Harabasz Index')
        plt.title('Calinski-Harabasz Index for KMeans')

        plt.tight_layout()
        plt.show()

        # K-means plot
        sns.scatterplot(x='price', y='rating', hue='kmeans_cluster', data=df, palette='viridis')
        plt.title('K-means Clustering')

    elif clustering_method.get() == 'Hierarchical':
        # Hierarchical
        agg_clustering = AgglomerativeClustering(n_clusters=10)
        df['hierarchical_cluster'] = agg_clustering.fit_predict(df[features])

        # Silhouette Score
        plt.subplot(1, 3, 1)
        silhouette_scores = []
        for k in range(2, 11):
            agg_clustering = AgglomerativeClustering(n_clusters=k)
            agg_clustering.fit(df[features])
            silhouette_scores.append(silhouette_score(df[features], agg_clustering.labels_))
        plt.plot(range(2, 11), silhouette_scores, marker='o')
        plt.xlabel('Number of clusters')
        plt.ylabel('Silhouette Score')
        plt.title('Silhouette Score for Hierarchical Clustering')
```

```
    # Davies-Bouldin Index
    plt.subplot(1, 3, 2)
    db_scores = []
    for k in range(2, 11):
        agg_clustering = AgglomerativeClustering(n_clusters=k)
        agg_clustering.fit(df[features])
        db_scores.append(davies_bouldin_score(df[features], agg_clustering.labels_))
    plt.plot(range(2, 11), db_scores, marker='o')
    plt.xlabel('Number of clusters')
    plt.ylabel('Davies-Bouldin Index')
    plt.title('Davies-Bouldin Index for Hierarchical Clustering')

    # Calinski-Harabasz Index
    plt.subplot(1, 3, 3)
    ch_scores = []
    for k in range(2, 11):
        agg_clustering = AgglomerativeClustering(n_clusters=k)
        agg_clustering.fit(df[features])
        ch_scores.append(calinski_harabasz_score(df[features], agg_clustering.labels_))
    plt.plot(range(2, 11), ch_scores, marker='o')
    plt.xlabel('Number of clusters')
    plt.ylabel('Calinski-Harabasz Index')
    plt.title('Calinski-Harabasz Index for Hierarchical Clustering')

    plt.tight_layout()
    plt.show()

    # Metrics
    silhouette = silhouette_score(df[features], agg_clustering.labels_)
    db_index = davies_bouldin_score(df[features], agg_clustering.labels_)
    ch_index = calinski_harabasz_score(df[features], agg_clustering.labels_)
    metrics_info = f"Silhouette Score: {silhouette:.4f}\nDavies-Bouldin Index: {db_index:.4f}\nCalinski-Harabasz Index: {ch_index:.4f}"

    # Hierarchical
    linked = linkage(df[features], 'ward')
    dendrogram(linked, orientation='top', distance_sort='descending', show_leaf_counts=True)
    plt.title('Hierarchical Clustering')


# Metrics MessageBox
messagebox.showinfo("Clustering Metrics", metrics_info)

show_game_selection()

plt.tight_layout()
plt.show()
```

Рисунок 3.4 – Метод Clustering()

Метод Clustering() виконує кластеризацію за допомогою двох методів: k-means та ієрархічної кластеризації. В залежності від обраного методу відображаються відповідні графіки та метрики кластеризації.

Якщо обрано метод "K-means", виконується кластеризація методом k-means з чотирма кластерами. Далі обчислюються та відображаються на графіку метрики, такі як Silhouette Score, Davies-Bouldin Index та Calinski-Harabasz Index. На рисунку 3.5 видно, що метрики відображаються для різної кількості кластерів. Після цього відображається графік кластеризації за допомогою методу k-means.

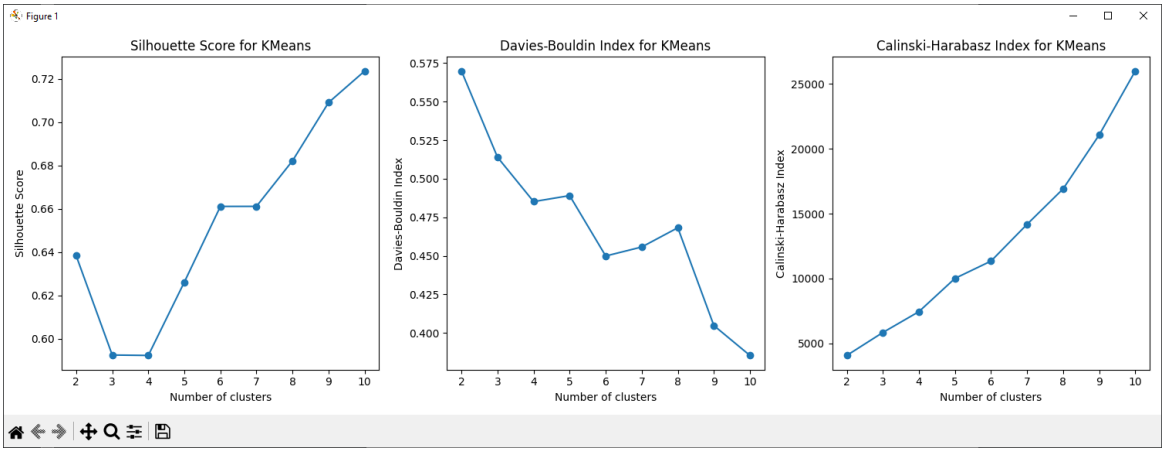


Рисунок 3.5 – Графіки метрик

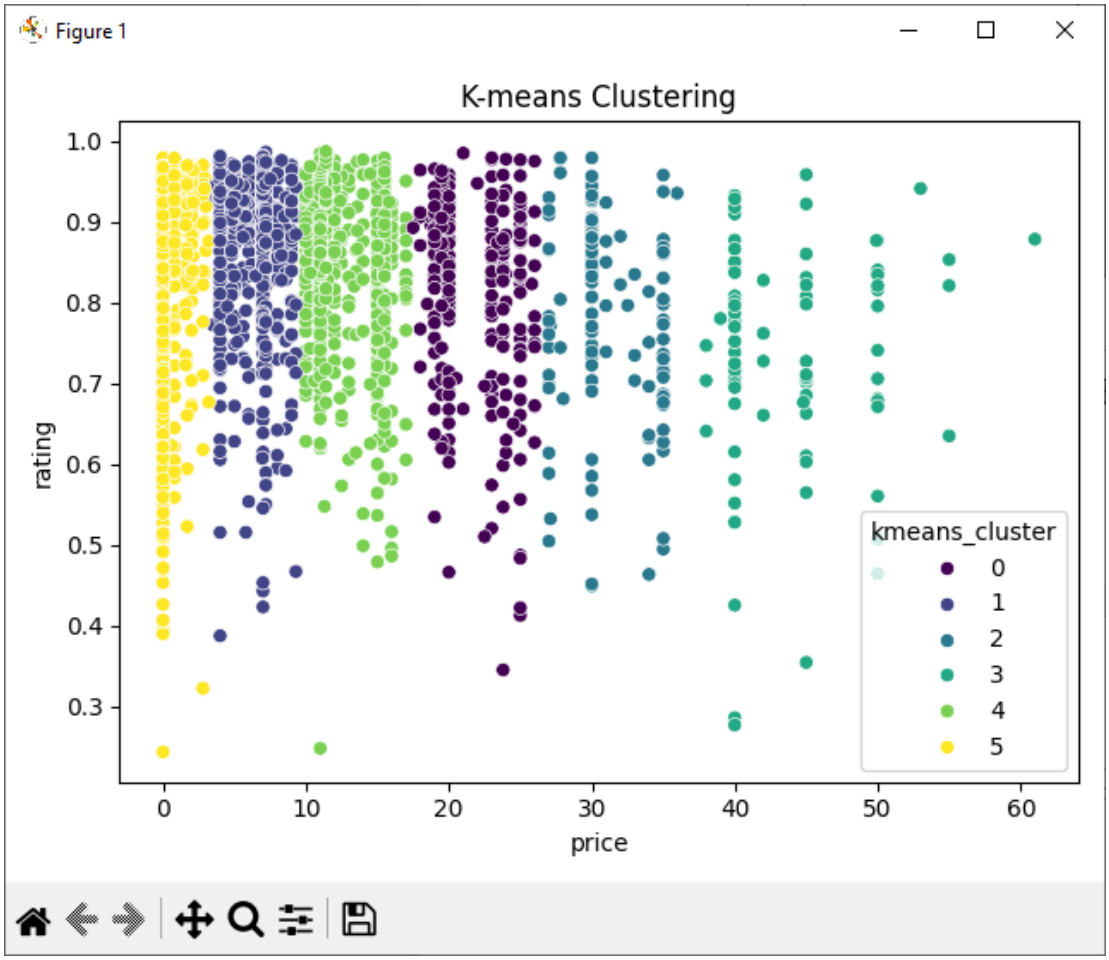


Рисунок 3.6 – Графік кластеризація методом k-means

Якщо обрано метод "Hierarchical", виконується ієрархічна кластеризація. Після цього також обчислюються та відображаються метрики, а саме Silhouette Score, Davies-Bouldin Index та Calinski-Harabasz Index для різної кількості

кластерів. На рисунку 3.8 по вісі x наведені app_id ігор у дата сеті. На рисунку 3.7 та на рисунку 3.8 можна побачити дендрограму ієрархічної кластеризації.

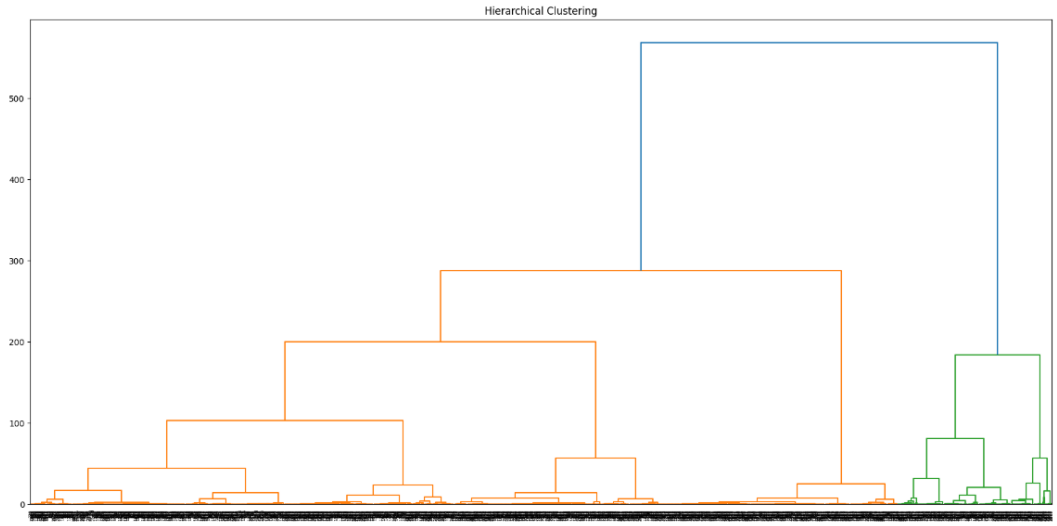


Рисунок 3.7 – Графік методом ієрархічної кластеризації

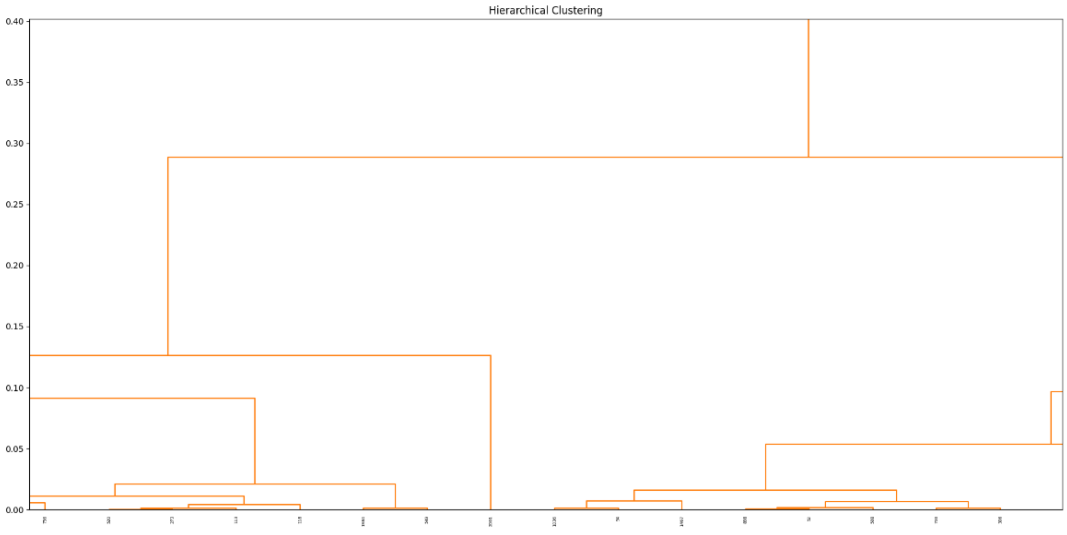


Рисунок 3.8 – Збільшений фрагмент графіку методом ієрархічної кластеризації

Після метрик, відображається вікно вибору ігри, де користувач може обрати гру зі списку. Список містить ті самі ігри, які представлені у наборі даних. Логіка пошуку рекомендацій наведена на рисунку 3.9.

```
def find_similar_games():
    title = selected_game.get()
    selected_game_data = df[df['name'] == title]

    if not selected_game_data.empty:
        app_id = selected_game_data.index[0]

        # Cosine similarity calculation
        cm = CountVectorizer().fit_transform(df['steamspy_tags'] + ' ' + df['name'])
        cs = cosine_similarity(cm)

        scores = list(enumerate(cs[app_id]))

        sorted_scores = sorted(scores, key=lambda x: x[1], reverse=True)
        sorted_scores = sorted_scores[1:]

        recommended_games = []
        for item in sorted_scores:
            try:
                game_title = df.iloc[item[0]]['name']
                if clustering_method.get() == 'K-means':
                    if df.iloc[item[0]]['kmeans_cluster'] == df.iloc[app_id]['kmeans_cluster']:
                        recommended_games.append(game_title)
                elif clustering_method.get() == 'Hierarchical':
                    if df.iloc[item[0]]['hierarchical_cluster'] == df.iloc[app_id]['hierarchical_cluster']:
                        recommended_games.append(game_title)
            except KeyError as e:
                print(f'Ignoring Error: {e}, item[0]: {item[0]}, app_id: {app_id}')

            if len(recommended_games) == 5:
                break

        if recommended_games:
            messagebox.showinfo("Top 5 Recommended Games", "\n".join(recommended_games))
        else:
            messagebox.showinfo("No Recommendations", f"No recommendations for {title} in the same cluster.")
```

Рисунок 3.9 – Метод find_similar_games()

У даному методі Find_similar_games() виконується пошук схожих ігор для вибраної користувачем гри. Кроки, які відбуваються в цьому методі:

1) отримання назви вибраної користувачем гри зі змінної «selected_game»;

2) пошук даних про вибрану гру у наборі даних;

3) якщо гра знайдена:

– отримання індексу гри з наборі даних;

– обчислення схожості косинусу між вибраною грою та всіма іншими грами. Це виконується шляхом використання CountVectorizer для перетворення

текстових даних гри на вектори та використання cosine_similarity для обчислення схожості косинусу між цими векторами;

– створення списку, що містить пари (індекс гри, схожість) та сортування цього списку за схожістю у зворотньому порядку;

– вибірка та збереження 5 найбільш схожих ігор, які також належать до того ж самого кластера, що й обрана гра;

– виведення інформаційного вікна з переліком рекомендованих ігор, якщо вони були знайдені. Інакше виведення повідомлення про те, що рекомендації для вибраної гри не знайдені в тому ж кластері;

4) Якщо гра не знайдена в наборі даних, виводиться відповідне повідомлення.

3.3 Аналіз результатів

В залежності від обраного метода кластеризації, обраної гри, та ознак кластеризації данні будуть змінюватися.

Для аналізу результатів кластеризації, обраної гри та ознак кластеризації, необхідно звернутися до конкретних властивостей кластерів, які утворилися під час роботи алгоритму. У моєму випадку, якщо обрано метод кластеризації "K-means" та обрана гра "Counter-Strike", то результатом може бути список ігор, які належать до того ж кластера, що й обрана гра "Counter-Strike". Це може бути пояснено тим, що ігри в одному кластері можуть мати подібні характеристики або мають схожий контекст або жанр.

На рисунку 3.10 наведено результат рекомендаційної системи через метод кластеризації k-means:

– Counter-Strike: Source;
– Counter-Strike: Condition Zero;
– Counter-Strike: Global Offensive;
– Ricochet;
– BRINK.

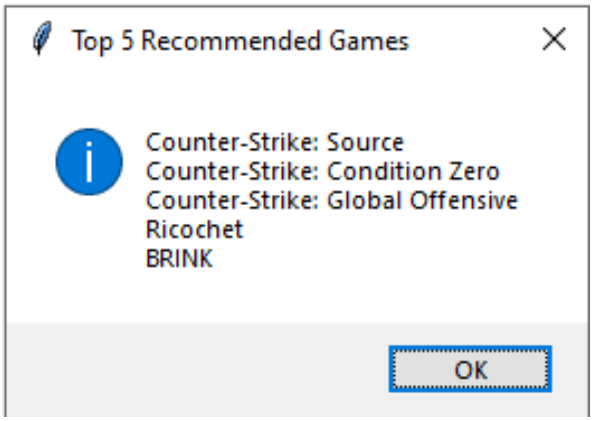


Рисунок 3.10 – Топ 5 рекомендованих ігор за допомогою методу k-means

З іншого боку, якщо обрано ієрархічний метод кластеризації, можуть виникнути інші результати, оскільки сам процес кластеризації може відрізнятися. Так, для того ж прикладу з грою "Counter-Strike" можуть бути інші ігри, які також належать до того ж кластера:

– Counter-Strike: Source;
– Counter-Strike: Condition Zero;
– Ricochet;
– Blockstorm;
– Deathmatch Classic.

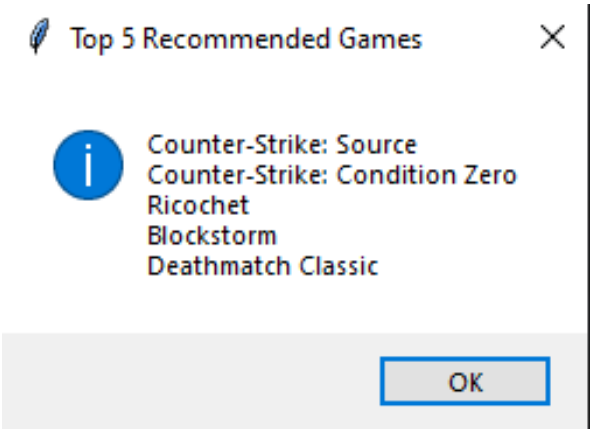


Рисунок 3.11 – Топ 5 рекомендованих ігор за допомогою ієрархічного методу кластеризації

Ці результати вказують на те, що обрані ігри мають схожі характеристики або можуть бути взаємопов'язані з певним контекстом або жанром у галузі комп'ютерних ігор. Такий аналіз може допомогти розуміти, які ігри можуть бути цікавими для користувача в залежності від його вибору та вподобань.

Протестуємо різні параметри, наприклад якщо кластерів буде 10, ми отримаємо результати, які представлені на рисунку 3.12 та на рисунку 3.13

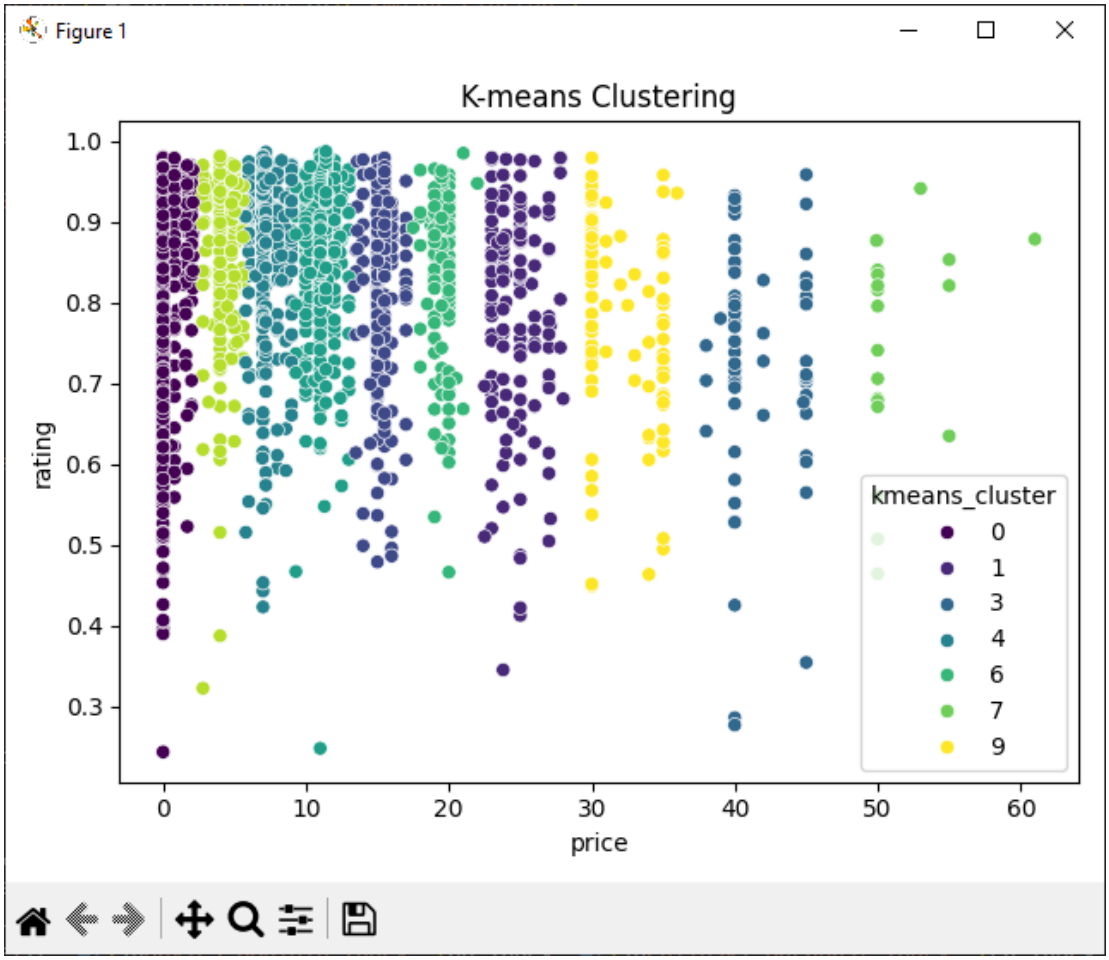


Рисунок 3.12 – Графік кластеризація методом k-means

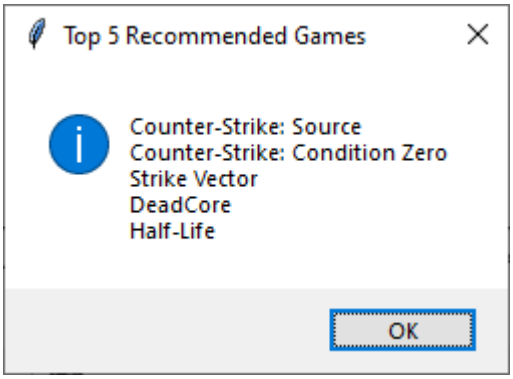


Рисунок 3.13 – Топ 5 рекомендованих ігор за допомогою кластеризації методом k-means

Через збільшення кількості кластерів деякі ігри, які пропонувалися раніше, опинилися у іншому кластері. Але завелика кількість кластерів призводить до менш точних результатів. Результати роботи занесені у таблицю 3.1.

Таблиця 3.2 – Результати надання рекомендації до гри Counter-Strike

Метод	Кількість кластерів	Гра 1	Гра 2	Гра 3	Гра 4	Гра 5
k-means	6	Counter-Strike: Source	Counter-Strike: Condition Zero	Counter-Strike: Global Offensive	Ricochet	BRINK
Ієрархічний	3	Counter-Strike: Source	Counter-Strike: Condition Zero	Ricochet	Blockstorm	Deathmatch Classic
k-means	10	Counter-Strike: Source	Counter-Strike: Condition Zero	Strike Vector	DeadCore	Half-life

Якщо взяти іншу гру для пошуку рекомендацій, наприклад «Day of Defeat», поділити ігри на 3 кластери, як вказано на рисунку 3.14, то результатом будуть ігри, які найбільше схожі на обрану гру. На рисунку 3.15 можна побачити результат роботи програми, який було отримано за допомогою методу k-means.

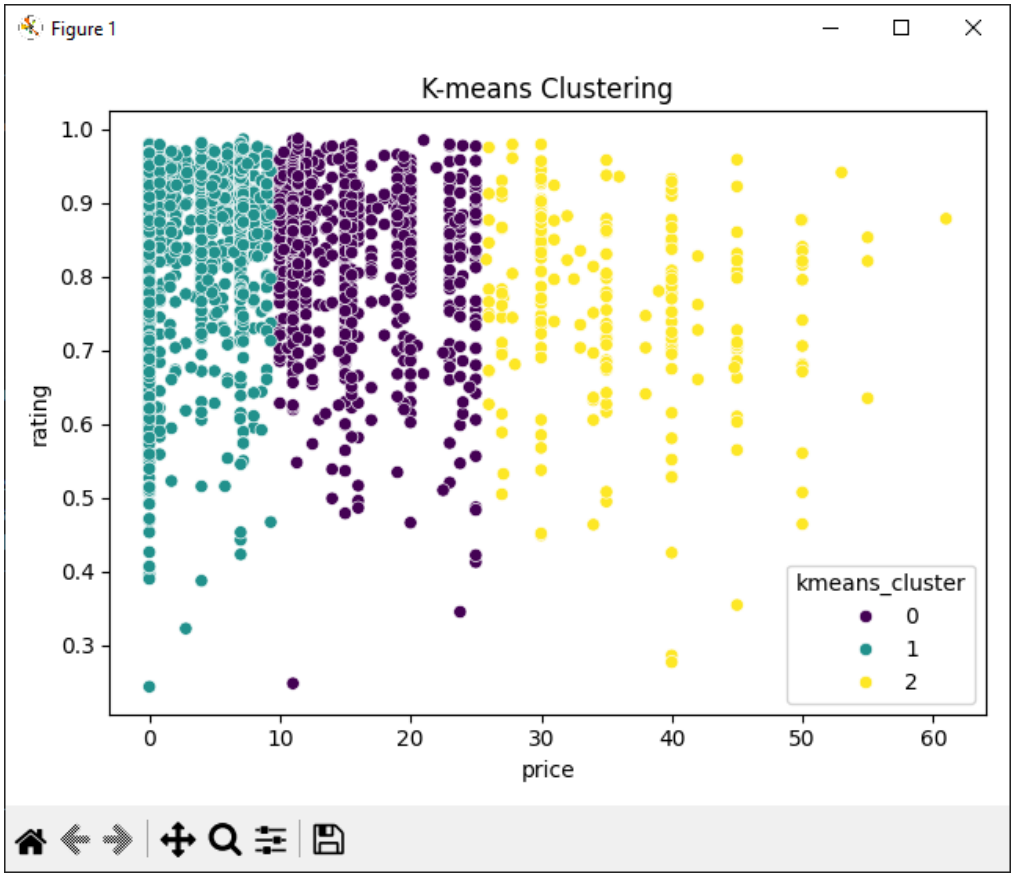


Рисунок 3.14 – Ділення ігор на 3 кластери методом k-means

Рисунок 3.15 – Топ 5 рекомендованих ігор за допомогою кластеризації методом k-means

На рисунку 3.16 та на рисунку 3.17 можна побачити, що при збільшені кількості кластерів до 7, зі списку прибралися безкоштовні ігри «War Thunder» та «World of Warships», так як обрана гра «Day of Defeat» згідно з набором даних коштує 3.99 фунтів.

Усі ігри, які були рекомендовані, є найбільш схожими на обрану гру. Вони відповідають темі, тегам, та ціні, в залежності від обраної кількості кластерів.

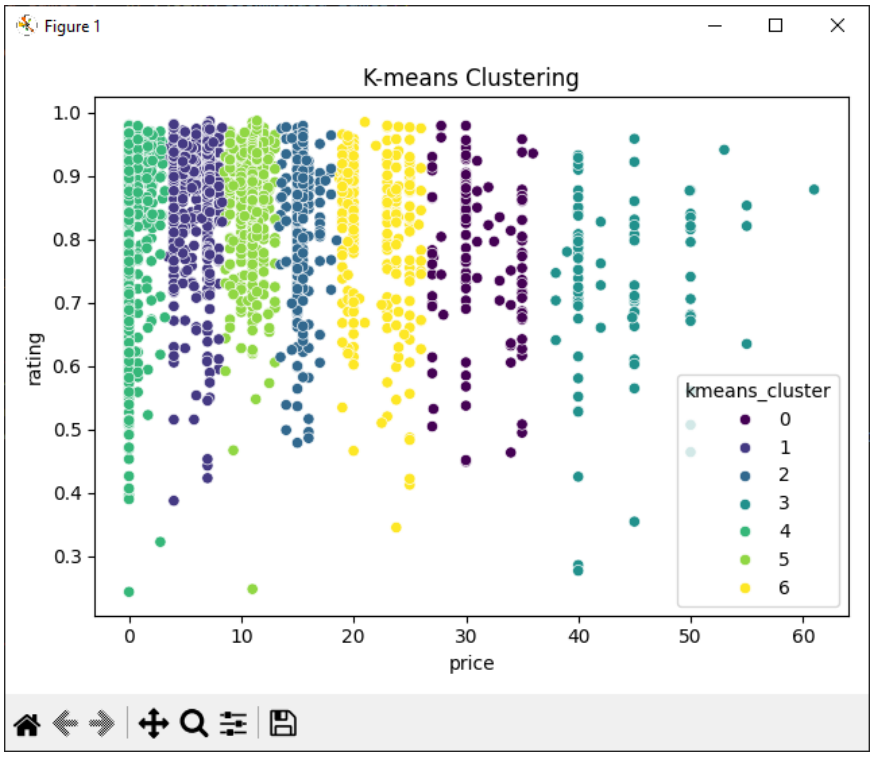


Рисунок 3.16 – Ділення ігор на 7 кластерів методом k-means

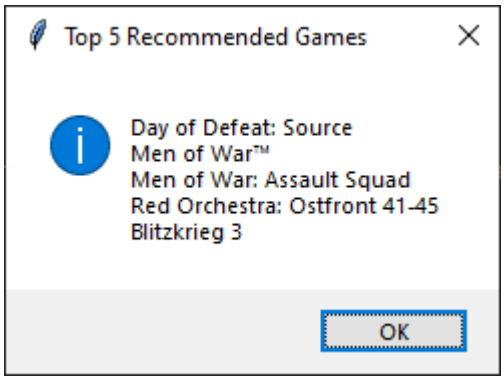


Рисунок 3.17 – Топ 5 рекомендованих ігор за допомогою кластеризації методом k-means

Таблиця 3.3 – Результати надання рекомендації до гри Day of Defeat

Метод	Кількість кластерів	Гра 1	Гра 2	Гра 3	Гра 4	Гра 5
k-means	3	Day of Defeat: Source	Men of War	War Thunder	Men of War: Assault Squad	World of Warships
k-means	7	Day of Defeat: Source	Men of War	Men of War: Assault Squad	Red Orchestra: Ostfront 41-45	Blitzkrieg 3

Висновки до розділу 3

У даному розділі була проведена програмна реалізація інтелектуальної системи кластеризації комп'ютерних ігор для персоналізованої рекомендації контенту. За допомогою мови програмування Python та використання відповідних бібліотек, таких як pandas, scikit-learn, та matplotlib, було створено програму, яка може аналізувати та кластеризувати дані про ігри на основі їх характеристик.

Програма надає користувачу можливість обрати метод кластеризації, включаючи K-means та ієрархічну кластеризацію. Після обрання методу користувач має можливість переглянути графіки та метрики якості кластеризації, такі як Silhouette Score, Davies-Bouldin Index, та Calinski-Harabasz Index. Крім того, система може рекомендувати користувачеві схожі ігри залежно від обраного методу кластеризації та вибраної гри.

Розроблена програмна система відкриває перспективи для подальшого дослідження у сфері інтелектуальних систем рекомендацій в галузі комп'ютерних ігор.

ВИСНОВКИ

У рамках кваліфікаційної роботи магістра було проведено комплексне дослідження та розроблено рекомендаційну систему для підвищення ефективності сервісів електронної комерції, зокрема магазинів комп'ютерних ігор.

Були проаналізовані проблеми та переваги існуючих аналогів рекомендаційних систем, визначені вимоги до програмного продукту та поставлено завдання. На основі цього аналізу було розроблено вдосконалену рекомендаційну систему, яка поєднує в собі контентну фільтрацію та кластеризацію для підвищення точності рекомендацій.

Було проведено аналіз та порівняння основних підходів до побудови рекомендаційних систем, а також методів кластеризації. Розроблений метод кластеризації ігор дозволяє поділити їх на групи зі схожими характеристиками, що допомагає покращити якість та адаптивність системи.

Розроблена рекомендаційна система реалізовує наступні функції:

– кластеризація ігор методами k-means та ієрархічним методом;

– виведення метрик кластеризації;

– надання рекомендацій у вигляді 5 ігор які найбільше схожі на обрану користувачем гру, за умови, що ці ігри лежать у одному й той самому кластері.

Розроблена рекомендаційна система надає точні та персоналізовані рекомендації на основі обраної користувачем гри. Це відкриває нові можливості для підвищення задоволення користувачів, збільшення продажів та підвищення конкурентоспроможності платформ електронної комерції.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Baptiste Rocca Introduction to recommender systems URL: https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada (дата звернення: 05.12.2023)
2. Блинков Н. Об’єм ринку комп’ютерних ігор URL: http://www.itweekly.ru/market/business/73058.html (дата звернення: 05.12.2023)
3. 2019 Essential Facts About the Computer and Video Game Industry URL: https://www.theesa.com/esa-research/2019-essential-facts-about-the-computer-andvideo-gameindustry/#:~:text=2018%20was%20a%20record%2Dbreaking,one%20gamer%20in%20 their%20household (дата звернення: 05.12.2023)
4. Cheuque, Germán & Guzman Gomez, Jose Antonio & Parra, Denis. (2019). Recommender Systems for Online Video Game Platforms: the Case of STEAM. 763-771. (дата звернення: 16.12.2023)
5. 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 (дата звернення: 16.12.2023)
6. Food Recommendation System Using Clustering Analysis for Diabetic Patients, Maiyaporn Phanich, Phathrajarin Pholkul, and Suphakant Phimoltares, Advanced Virtual and Intelligent Computing (AVIC), Research Center Department of Mathematics, Faculty of Science, Chulalongkorn University Pathumwan, Bangkok, Thailand, 2010. (дата звернення: 16.12.2023)
7. Dr. Manju Kaushik, Mrs. Bhawana Mathur, Comparative Study of K-Means and Hierarchical Clustering Techniques, ijournals, International Journal of Software & Hardware Research in Engineering. ISSN No:2347-4890, 2014, с. 93-98 (дата звернення: 16.12.2023)
8. Marek_Gagolewski, Maciej Bartoszuk, Anna Cena. Are cluster validity measures (in)

valid?. URL: https://www.sciencedirect.com/science/article/abs/pii/S0020025521010082 (дата звернення: 16.12.2023)

9. Netflix, URL: https://uk.wikipedia.org/wiki/Netflix (дата звернення: 05.12.2023) (дата звернення: 16.12.2023)
10. How Netflix's Recommendations System Works, URL: https://help.netflix.com/en/node/100639 (дата звернення: 13.12.2023)
11. Boyd Clark How Spotify Recommends Your New Favorite Artist URL: https://clarkboyd.medium.com/ (дата звернення: 05.12.2023)
12. Chalyi S., Leshchynskyi V., Leshchynska, I. Method of forming recommendations using temporal constraints in a situation of cyclic cold start of the 88 recommender system Chalyi S., Leshchynskyi V., Leshchynska I. EUREKA, Physics and Engineering, 2019, 2019(4), c. 34-40 (дата звернення: 05.12.2023)
13. Inside Spotify's Recommender System: A Complete Guide to Spotify Recommendation Algorithms, URL: https://www.music-tomorrow.com/blog/how-spotify-recommendation-system-works-a-complete-guide-2022 (дата звернення: 05.12.2023)
14. Introducing The Steam Interactive Recommender. URL: https://store.steampowered.com/news/app/593110/view/1716373422378712840 (дата звернення: 05.12.2023)
15. S.Pandya, J.Shah, N.Joshi, H.Ghayvat, S.C.Mukhopadhyay, M.H.Yap, A Novel Hybrid based Recommendation System based on Clustering and Association Mining, Tenth International Conference on Sensing Technology, Massey University, Palmerston North, 2016 (дата звернення: 16.12.2023)
16. Rishabh Ahuja, Arun Solanki, Anand Nayyar, Movie Recommender System Using K-Means Clustering AND K-Nearest Neighbor, 2019 (дата звернення: 16.12.2023)
17. Chris I. K-Means Clustering Explained URL: https://medium.com/dataseries/k-means-clustering-explained-visually-in-5-minutesb900cc69d175 (дата звернення: 16.12.2023)

18. Keerti Prajapati K-means Clustering explained in detailed URL:https://medium.com/@codingpilot25/k-means-clustering-explained-in-detailed30484910e381 (дата звернення: 16.12.2023)
19. Yuan Lu, Jie Yang, Notes on Low-rank Matrix Factorization, URL: https://yangjiera.github.io/pdf/low-rank.pdf (дата звернення: 16.12.2023)
20. 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: 16.12.2023)
21. 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 89
22. What is Deployment Diagram? URL: https://www.visualparadigm.com/guide/uml-unified-modeling-language/what-is-deployment-diagram/ (дата звернення: 16.12.2023)
23. 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
24. 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-equalpt1 (дата звернення: 16.12.2023)
25. Bao J., Zheng Y. Location-Based Recommendation Systems. B: Shekhar S., Xiong H., Zhou X. (ред.) Encyclopedia of GIS. Springer, Cham, 2017.
26. Jerold Angelus Grundy Newbrain // Duct Publishing – 2012 – с. 120.
27. D. Goldberg Using Collaborative Filtering to Weave an Information Tapestry / D. Goldberg, D. Nichols, B.M. Oki, D. Terry // Comm. ACM. – 1992. – Vol. 35, No12.

– с. 61-70.

28. Gleb Beliakov, Tomasa Calvo and Simon James. Aggregation of preferences in recommender systems / Gleb Beliakov and Simon James // School of Information Technology, Deakin University – 2008 – с. 705-735.
29. J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In Proceedings of the 21th International Conference on Machine Learning, 2004 – с. 9-16
30. C.C. Aggarwal Horting Hatches an Egg: A New Graph-Theoretic Approach to Collaborative Filtering / C.C. Aggarwal, J.L. Wolf, K-L. Wu, P.S. Yu // Proc. Fifth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining. – 1999 – с. 201-212.
31. P. Resnick GroupLens: An Open Architecture for Collaborative Filtering of Netnews / P. Resnick, N. Iakovou, M. Sushak, P. Bergstrom, J. Riedl // Proc. 1994 Computer Supported Cooperative Work Conf. – 1994 – с. 175-186. 85
32. U. Shardanand Social Information Filtering: Algorithms for Automating "Word of Mouth" / U. Shardanand, P. Maes // Proc. Conf. Human Factors in Computing Systems. – 1995 – с. 210-217.
33. J.S. Breese Empirical Analysis of Predictive Algorithms for Collaborative Filtering / J.S. Breese, D. Heckerman, C. Kadie // Proc. 14th Conf. Uncertainty in Artificial Intelligence. – 1998 – с. 43-52.
34. B. Sarwar Item-Based Collaborative Filtering Recommendation Algorithms / B. Sarwar, G. Karypis, J. Konstan, J. Riedl // Proc. 10th Int'l WWW Conf. – 2001 – с. 285- 295.
35. J. Delgado Memory-Based Weighted-Majority Prediction for Recommender Systems / J. Delgado, N. Ishii // Proc. ACM SIGIR '99 Workshop Recommender Systems: Algorithms and Evaluation. – 1999 – с. 186-198.
36. A. Nakamura Collaborative Filtering Using Weighted Majority Prediction Algorithms / A. Nakamura, N. Abe //Proc. 15th Int'l Conf. Machine Learning. – 1998 – с. 395-403.

37. I. Soboroff Combining Content and Collaborationin Text Filtering / I. Soboroff, C. Nicholas // Proc. Int'l Joint Conf. Artificial Intelligence Workshop: Machine Learning for Information Filtering. – 1999 – с. 86-91.
38. M. Claypool Combining Content-Based and Collaborative Filters in an Online Newspaper / M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, M. Sartin // Proc. ACM SIGIR '99 Workshop Recommender Systems: Algorithms and Evaluation, Aug. – 1999.
39. D. Billsus User Modeling for Adaptive News Access / D. Billsus, M. Pazzani // User Modeling and User-Adapted Interaction. – 2000. – Vol. 10, No2, No3. – с. 147-180.
40. A.I. Schein Methods and Metrics for Cold-Start Recommendations / A.I. Schein, A. Popescul, L.H. Ungar, D.M. Pennock // Proc. 25th Ann. Int'l ACM SIGIR Conf. – 2002 – с. 253-260.
41. А. В. Заболеева-Зотова Латентный семантический анализ: новые решения в Internet / А. В. Заболеева-Зотова, А. Ю. Пастухов, П. В. Сердюков, Н. А. Козлова, С. А. Чернов // Информационные технологии. – 2001 – с. 67-82. 86
42. M. Pazzani A Framework for Collaborative, Content-Based, and Demographic Filtering // Artificial Intelligence Rev. – 1999. – с. 393-408.
43. M. Balabanovic Fab: Content-Based, Collaborative Recommendation / M. Balabanovic, Y. Shoham // Comm. ACM. – 1997. – Vol. 40, No3. – с.66-72.
44. P. Melville Content-Boosted Collaborative Filtering for Improved Recommendations / P. Melville, R.J. Mooney, R. Nagarajan // Proc. 18th Nat'l Conf. Artificial Intelligence. – 2002 – с. 187–192.
45. E. A¨ımeur, G. Brassard, J. M. Fernandez, and F. S. M. Onana, Privacy-preserving demographic filtering, in Proceedings of the ACM symposium on Applied computing. New York, NY, USA: ACM, 2006 – с. 872–878.
46. Powers, David M. W. Evaluatio n: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation // Journal of Machine Learning Technologies, 2011 – с. 37–63.
47. Когулько О.С. Використання методів колаборативної фільтрації для роботи

рекомендаційної системи / Міжнародна науково-практична конференція «Математичне та імітаційне моделювання систем» (МОДС-2018) – м. Київ., 25-29 червня 2018 р. – с. 83-86.

48. Когулько О. С., Попенко В.Д. Надання рекомендацій елементів на основі гібридної фільтраціЇ/ О.С. Когулько, В.Д. Попенко / Всеукраїнська науковопрактична конференція молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2018) – м. Київ.: НТУУ «КПІ ім. Ігоря Сікорського», 29-30 листопада 2018 р – с. 61-65.

ДОДАТОК А

Код програмної реалізації

```
import pandas as pd
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.cluster import KMeans, AgglomerativeClustering
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.metrics import silhouette_score, davies_bouldin_score,
calinski_harabasz_score
import tkinter as tk
from tkinter import messagebox

def calculate_rating(row):
    return row['positive_ratings'] / (row['positive_ratings'] +
row['negative_ratings'])

def find_similar_games():
    title = selected_game.get()
    selected_game_data = df[df['name'] == title]

    if not selected_game_data.empty:
        app_id = selected_game_data.index[0]

        # Cosine similarity calculation
        cm = CountVectorizer().fit_transform(df['steamspy_tags'] + ' ' + df['name'])
        cs = cosine_similarity(cm)

        scores = list(enumerate(cs[app_id]))

        sorted_scores = sorted(scores, key=lambda x: x[1], reverse=True)
        sorted_scores = sorted_scores[1:]

        recommended_games = []
        for item in sorted_scores:
            try:
                game_title = df.iloc[item[0]]['name']
                if clustering_method.get() == 'K-means':
                    if df.iloc[item[0]]['kmeans_cluster'] ==
df.iloc[app_id]['kmeans_cluster']:
                        recommended_games.append(game_title)
                elif clustering_method.get() == 'Hierarchical':
                    if df.iloc[item[0]]['hierarchical_cluster'] ==
df.iloc[app_id]['hierarchical_cluster']:
                        recommended_games.append(game_title)
```

```
            except KeyError as e:
                print(f'Ignoring Error: {e}, item[0]: {item[0]}, app_id: {app_id}')

            if len(recommended_games) == 5:
                break

        if recommended_games:
            messagebox.showinfo("Top 5 Recommended Games",
"\n".join(recommended_games))
        else:
            messagebox.showinfo("No Recommendations", f"No recommendations for
{title} in the same cluster.")


def Clustering():
    plt.figure(figsize=(15, 5))

    if clustering_method.get() == 'K-means':
        # K-means
        kmeans = KMeans(n_clusters=7)
        df['kmeans_cluster'] = kmeans.fit_predict(df[features])

        # Metrics
        silhouette = silhouette_score(df[features], kmeans.labels_)
        db_index = davies_bouldin_score(df[features], kmeans.labels_)
        ch_index = calinski_harabasz_score(df[features], kmeans.labels_)
        metrics_info = f"Silhouette Score: {silhouette:.4f}\nDavies-Bouldin Index:
{db_index:.4f}\nCalinski-Harabasz Index: {ch_index:.4f}"

        # Silhouette Score
        plt.subplot(1, 3, 1)
        silhouette_scores = []
        for k in range(2, 11):
            kmeans = KMeans(n_clusters=k)
            kmeans.fit(df[features])
            silhouette_scores.append(silhouette_score(df[features], kmeans.labels_))
        plt.plot(range(2, 11), silhouette_scores, marker='o')
        plt.xlabel('Number of clusters')
        plt.ylabel('Silhouette Score')
        plt.title('Silhouette Score for KMeans')

        # Davies-Bouldin Index
        plt.subplot(1, 3, 2)
        db_scores = []
        for k in range(2, 11):
            kmeans = KMeans(n_clusters=k)
            kmeans.fit(df[features])
            db_scores.append(davies_bouldin_score(df[features], kmeans.labels_))
        plt.plot(range(2, 11), db_scores, marker='o')
```

```
        plt.xlabel('Number of clusters')
        plt.ylabel('Davies-Bouldin Index')
        plt.title('Davies-Bouldin Index for KMeans')

        # Calinski-Harabasz Index
        plt.subplot(1, 3, 3)
        ch_scores = []
        for k in range(2, 11):
            kmeans = KMeans(n_clusters=k)
            kmeans.fit(df[features])
            ch_scores.append(calinski_harabasz_score(df[features], kmeans.labels_))
        plt.plot(range(2, 11), ch_scores, marker='o')
        plt.xlabel('Number of clusters')
        plt.ylabel('Calinski-Harabasz Index')
        plt.title('Calinski-Harabasz Index for KMeans')

        plt.tight_layout()
        plt.show()

        # K-means plot
        sns.scatterplot(x='price', y='rating', hue='kmeans_cluster', data=df,
palette='viridis')
        plt.title('K-means Clustering')

    elif clustering_method.get() == 'Hierarchical':
        # Hierarchical
        agg_clustering = AgglomerativeClustering(n_clusters=10)
        df['hierarchical_cluster'] = agg_clustering.fit_predict(df[features])

        # Silhouette Score
        plt.subplot(1, 3, 1)
        silhouette_scores = []
        for k in range(2, 11):
            agg_clustering = AgglomerativeClustering(n_clusters=k)
            agg_clustering.fit(df[features])
            silhouette_scores.append(silhouette_score(df[features],
agg_clustering.labels_))
        plt.plot(range(2, 11), silhouette_scores, marker='o')
        plt.xlabel('Number of clusters')
        plt.ylabel('Silhouette Score')
        plt.title('Silhouette Score for Hierarchical Clustering')

        # Davies-Bouldin Index
        plt.subplot(1, 3, 2)
        db_scores = []
        for k in range(2, 11):
            agg_clustering = AgglomerativeClustering(n_clusters=k)
            agg_clustering.fit(df[features])
```

```
            db_scores.append(davies_bouldin_score(df[features],
agg_clustering.labels_))
        plt.plot(range(2, 11), db_scores, marker='o')
        plt.xlabel('Number of clusters')
        plt.ylabel('Davies-Bouldin Index')
        plt.title('Davies-Bouldin Index for Hierarchical Clustering')

        # Calinski-Harabasz Index
        plt.subplot(1, 3, 3)
        ch_scores = []
        for k in range(2, 11):
            agg_clustering = AgglomerativeClustering(n_clusters=k)
            agg_clustering.fit(df[features])
            ch_scores.append(calinski_harabasz_score(df[features],
agg_clustering.labels_))
        plt.plot(range(2, 11), ch_scores, marker='o')
        plt.xlabel('Number of clusters')
        plt.ylabel('Calinski-Harabasz Index')
        plt.title('Calinski-Harabasz Index for Hierarchical Clustering')

        plt.tight_layout()
        plt.show()

        # Metrics
        silhouette = silhouette_score(df[features], agg_clustering.labels_)
        db_index = davies_bouldin_score(df[features], agg_clustering.labels_)
        ch_index = calinski_harabasz_score(df[features], agg_clustering.labels_)
        metrics_info = f"Silhouette Score: {silhouette:.4f}\nDavies-Bouldin Index:
{db_index:.4f}\nCalinski-Harabasz Index: {ch_index:.4f}"

        # Hierarchical
        linked = linkage(df[features], 'ward')
        dendrogram(linked, orientation='top', distance_sort='descending',
show_leaf_counts=True)
        plt.title('Hierarchical Clustering')



    # Metrics MessageBox
    messagebox.showinfo("Clustering Metrics", metrics_info)

    show_game_selection()

    plt.tight_layout()
    plt.show()

def show_game_selection():
    game_label.pack()
    games_dropdown.pack()
```

```
    search_button.pack()
# Data Loading
df = pd.read_csv('steam.csv')
df = df[df['positive_ratings'] > 1000]
df['rating'] = df.apply(calculate_rating, axis=1)
features = ['price', 'rating']
# Init UI
root = tk.Tk()
root.title("Game Recommendation System")
root.geometry("400x200")
clustering_method = tk.StringVar(root)
clustering_method.set("K-means")
# Cluster method selecor
method_label = tk.Label(root, text="Choose Clustering Method:")
method_label.pack()
method_option_menu = tk.OptionMenu(root, clustering_method, "K-means",
"Hierarchical")
method_option_menu.pack()
metrics_button = tk.Button(root, text="Display Metrics and Plots",
command=Clustering)
metrics_button.pack()
space_label = tk.Label(root, text="", height=1)
space_label.pack()
# Game list selector
game_label = tk.Label(root, text="Select Game:")
game_label.pack()
selected_game = tk.StringVar(root)
games_dropdown = tk.OptionMenu(root, selected_game, *df['name'])
games_dropdown.pack()
search_button = tk.Button(root, text="Find Similar Games",
command=find_similar_games)
search_button.pack()
game_label.pack_forget()
games_dropdown.pack_forget()
search_button.pack_forget()
root.mainloop()
```