Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ МЕТОДІВ АНАЛІЗУ ДАНИХ ВЕБСАЙТІВ (НА ПРИКЛАДІ САЙТУ ПО ПРОДАЖУ ВІДЕОІГОР)
(тема)

Виконав:
здобувач 2 року навчання,
групи ІНФм-24-2
Терещенко О. О.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Науковий керівник доц. Тітова О. В.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Терещенку Олександру Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів аналізу даних вебсайтів (на прикладі сайту по продажу відеоігор)

затверджена наказом університету від 14 листопада 2025 року № 1045Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 21 листопада 2025 р.
3. Вихідні дані до роботи масиви користувацьких рейтингів, текстових відгуків та метаданих комп'ютерних ігор, що використовуються для побудови персоналізаційних моделей. До них належать оцінки гравців, дані про взаємодії з іграми, інформація про жанри, розробників і загальні характеристики продуктів. Додатково використовуються приклади фальсифікованих та аномальних оцінок, що застосовуються для тестування методів виявлення маніпуляцій.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз сучасних методів персоналізації та виявлення фальсифікованих рейтингів у вебплатформах.
2. Аналіз літературних джерел щодо застосування методів рекомендаційних систем, алгоритмів виявлення аномалій та лінгвістичного аналізу текстових відгуків.
3. Формування покрокових алгоритмів для вибраних методів: колаборативної фільтрації, матричної факторизації, алгоритму Isolation Forest та методу TF-IDF.
4. Візуалізація сформованих покрокових алгоритмів.
5. Розробка програмного застосунку, що забезпечує персоналізацію рекомендацій, виявлення аномальних оцінок та аналіз достовірності текстових відгуків користувачів.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність проблеми персоналізації у вебмагазинах комп'ютерних ігор, об'єкт та мета дослідження, постановка задачі, схема архітектури аналітичної моделі, блок-схема взаємодії модулів персоналізації, виявлення аномалій та текстового аналізу, приклад користувацького відгуку для TF-IDF, приклад фрагментів даних рейтингів, приклад роботи алгоритму Isolation Forest, приклад формування персональних рекомендацій, ілюстрація інтерфейсу вебзастосунку з відображенням рекомендованих ігор, приклад результатів матричної факторизації, приклади тестування персоналізації, таблиці з результатами аналізу, висновки, матеріали апробації роботи.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	29.09.2025	
2	Аналіз завдання, підбір літератури	29.09.25-07.10.25	
3	Аналіз літератури з досліджуваної проблеми	08.10.25-14.10.25	
4	Аналіз методів	15.10.25-20.10.25	
5	Розробка алгоритмів	21.10.25-27.10.25	
6	Програмна реалізація	28.10.25-05.11.25	
7	Обґрунтування отриманих результатів	06.11.25-11.11.25	
8	Оформлення пояснювальної записки	12.11.25-14.11.25	
9	Перевірка на нормоконтроль	19.11.25-10.12.25	
10	Перевірка на плагіат	20.11.25-10.12.25	
11	Рецензування	21.11.25-10.12.25	
12	Підготовка презентації та доповіді	21.11.25-22.12.25	
13	Занесення роботи в електронний архів	21.11.25-22.12.25	
14	Попередній захист кваліфікаційної роботи	01.12.25-22.12.25	

Дата видачі завдання 29 вересня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Тітова О. В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи: 78 с., 2 табл., 7 рис., 40 джерела.

АНАЛІЗ ДАНИХ, АЛГОРИТМ , ВЕБСАЙТИ, КОМП'ЮТЕРНІ ІГРИ, МАТРИЧНА ФАКТОРИЗАЦІЯ, ПЕРСОНАЛІЗАЦІЯ, РЕКОМЕНДАЦІЇ, ФІЛЬТРАЦІЯ, TERM FREQUENCY INVERSE DOCUMENT FREQUENCY.

Об'єктом дослідження є процес аналізу користувацьких даних сайтів.

Предметом дослідження є методи персоналізації та виявлення фальсифікованих рейтингів і відгуків у системах оцінювання ігор.

Метою дослідження є створення комплексного методу аналізу даних вебмагазину, який забезпечує автоматичну перевірку достовірності оцінок, формування персональних рекомендацій і підвищення точності аналітики користувацької поведінки.

У процесі роботи використано методи колаборативної фільтрації, прогнозування часових рядів та виявлення аномалій.

Наукова новизна роботи полягає у комплексному підході до аналізу даних вебмагазину комп'ютерних ігор, який поєднує персоналізацію рекомендацій.

Взаємозв'язок з іншими роботами полягає у використані результатів попередніх досліджень у машинному навчанні, інтелектуальному аналізі даних та розробці систем підтримки рішень для електронної комерції.

Рекомендації щодо використання результатів роботи є впровадження підходів аналізу даних у вебмагазини для поліпшення досвіду користувачів.

У результаті дослідження розроблено сервіс для персоналізації та прогнозування вподобань покупців.

ABSTRACT

Explanatory note to the qualification work: 78 pages, 2 table, 7 figures, 40 sources.

COMPUTER GAMES, DATA ANALYSIS, FILTERING, MATRIX FACTORIZATION ALGORITHM, PERSONALIZATION, RECOMMENDATIONS, TERM FREQUENCYINVERSE DOCUMENT FREQUENCY, WEBSITES.

The object of the research is the process of analyzing user data on websites.

The subject of the research is methods of personalization and detection of fake ratings and reviews in game rating systems.

The goal of the research is to create a comprehensive method for analyzing web store data that provides automatic verification of the reliability of ratings, the formation of personalized recommendations, and improved accuracy of user behavior analytics.

The work uses methods of collaborative filtering, time series forecasting, and anomaly detection.

The scientific novelty of the work lies in a comprehensive approach to analyzing computer game online store data, which combines the personalization of recommendations.

The connection with other works lies in the use of the results of previous research in machine learning, intelligent data analysis, and the development of decision support systems for e-commerce.

Recommendations for using the results of the work include implementing data analysis approaches in online stores to improve the user experience.

As a result of the research, a service for personalizing and predicting customer preferences has been developed.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМНІВ

Пайплайн – це набір даних, що буде оброблятися послідовно

Тайтл – загальна познака на ігри, без вказання на конкретну гру

API – Application Programming Interface (програмний інтерфейс взаємодії між компонентами)

F1-score – метрика точности класифікації, що враховує precision та recall

HTTP – HyperText Transfer Protocol (протокол передачі гіпертексту)

Isolation Forest – алгоритм машинного навчання для виявлення аномалій

JSON – JavaScript Object Notation (формат обміну даними між системами)

JWT – JSON Web Token (токен автентифікації для вебдодатків)

MUI – Material UI (бібліотека компонентів користувацького інтерфейсу для React)

MongoDB – документоорієнтована база даних із динамічною схемою

REST API – Representational State Transfer Application Programming Interface (архітектурний стиль побудови вебсервісів)

TF-IDF – Term Frequency Inverse Document Frequency (метод оцінки важливости слів у тексті)

VS Code – Visual Studio Code (середовище розробки програмного забезпечення)

ВСТУП

Сучасна ігрова індустрія переживає стрімкий розвиток під впливом цифрових технологій та зростання онлайн-платформ дистрибуції. Вебмагазини комп'ютерних ігор, такі як Steam, Epic Games Store, GOG чи itch.io, стали не лише торговими майданчиками, а й дужими джерелами даних про поведінку користувачів, їхні вподобання, динаміку продажів і реакції на вміст. У цих системах накопичуються величезні обсяги даних про оцінки, відгуки, частоту покупок, час взаємодії з платформою, що створює широкі можливості для побудови аналітичних і прогнозних моделей.

Важливою задачею для розробників, маркетологів та власників вебплатформ є розуміння факторів, що впливають на купівельну активність користувачів і формування довіри до рейтингових систем. Значна частина оцінок і відгуків може бути викривлена через маніпуляції або автоматизовані фальсифікації, що спотворює реальну картину популярності продуктів. Одночасно з цим, персоналізовані рекомендації набувають особливого значення, оскільки саме вони визначають, які ігри користувач побачить і, зрештою, придбає. Виявлення закономірностей між поведінкою користувачів, їхніми вподобаннями та історією покупок дозволяє підвищити точність рекомендацій і поліпшити комерційні результати платформи.

Актуальність роботи полягає у зростаючій потребі вебмагазинів у системах інтелектуального аналізу даних, здатних не лише формувати персональні рекомендації, а й виявляти аномальні дії та прогнозувати купівельну активність. Попри поширення рекомендаційних систем у сфері електронної комерції, більшість із них не враховує вплив недостовірних оцінок, підозрілих відгуків та зовнішніх факторів на користувацьку поведінку. Тому актуальним є створення комплексного підходу, який поєднує аналіз достовірності даних, моделі персоналізації та методи прогнозування попиту.

Сучасні підходи до аналізу даних вебмагазинів базуються на машинному навчанні, статистичному моделюванні та методах обробки природної мови. Для

персоналізації використовуються алгоритми колаборативної фільтрації, для прогнозування попиту моделі часових рядів (зокрема ARIMA), а для виявлення фальсифікованих відгуків методи аномалійного аналізу, як-от Isolation Forest. Проте більшість реалізацій розглядає ці завдання ізольовано, що знижує ефективність комплексної аналітики поведінки користувачів.

У даній роботі пропонується створення аналітичної системи, яка поєднує три взаємопов'язані напрями. Перше, персоналізація рекомендацій, реалізована через метод матричної факторизації, що враховує латентні фактори користувачів і продуктів. Друге, прогнозування купівельної активності, здійснене на основі моделі ARIMA для аналізу часових трендів продажів. Третє, виявлення аномалій і накручувань рейтингів, для чого використовується алгоритм Isolation Forest, що дозволяє визначати підозрілі патерни поведінки.

Наукова задача дослідження полягає у створенні інтегрованого методу аналізу даних вебмагазину, здатного одночасно виконувати три ключові функції: персоналізацію, прогнозування й знаходження аномалій. Запропонований підхід забезпечує підвищену точність у формуванні рекомендацій, дозволяє виявляти недостовірну активність користувачів і прогнозувати зміни в попиті на ігри, що має практичну цінність для розвитку електронної комерції у сфері цифрових розваг.

Отже, у роботі здійснюється аналіз сучасних методів машинного навчання, що застосовуються для побудови інтелектуальних систем у вебмагазинах, розробляється власна модель інтегрованого аналізу даних і створюється прототип застосунку, який оцінює надійність відгуків, прогнозує купівельну активність та формує персональні рекомендації для користувачів. Отримані результати можуть бути використані в аналітичних платформах електронної комерції для підвищення ефективності продажів, прозорості рейтингових систем і якості користувацького досвіду.

1 ОГЛЯД ПРОБЛЕМАТИКИ ПЕРСОНАЛІЗАЦІЇ

1.1 Нинішня персоналізація в вебмагазинах

Нинішні вебмагазини активно використовують персоналізацію як ключовий інструмент утримання користувачів і підвищення продажів. Алгоритми персоналізації аналізують історію покупок, переглядів, час, проведений на сторінках товарів, кліки, рейтинги та відгуки, щоб створити індивідуальний профіль кожного користувача. На основі цього профілю система формує рекомендації ігор, акцій чи добірок, які з високою ймовірністю, зацікавлять саме цього користувача [1-5].

Найпоширенішими підходами є колаборативна фільтрація, яка знаходить користувачів зі схожими інтересами, та змістовна персоналізація, що базується на властивостях товарів, як то жанр, ціна, рейтинг, розробник тощо. Деякі великі платформи, як-от Steam чи GOG, доповнюють ці методи гібридними моделями, які поєднують поведінкові дані користувача з змістовними характеристиками ігор.

Однак попри високий рівень розвитку алгоритмів, нинішня персоналізація все ще стикається з низкою проблем. Зокрема, вона вразлива до викривлених або фальсифікованих рейтингів, які змінюють розподіл рекомендацій, а також часто не враховує контекст, як сезонні тенденції, події у спільнотах чи маркетингові кампанії. Як наслідок, персоналізація у вебмагазинах залишається потужним, але недосконалим інструментом, який потребує подальшого вдосконалення через глибший аналіз даних і перевірку їх достовірності.

1.2 Важливість персоналізації

У сучасну епоху цифрових технологій персоналізація стала одним із ключових чинників, що визначають успішність будь-якої онлайн-платформи. З

розвитком електронної комерції, потокових сервісів та соціальних мереж зросли вимоги користувачів до зручності, швидкості й релевантності отримуваної інформації. Персоналізація є не лиш адаптація вмісту під індивідуальні потреби користувача, а й глибоке розуміння його поведінки, мотивів і намірів на основі аналізу великих обсягів даних.

У сфері цифрових розваг, зокрема продажу комп'ютерних ігор, персоналізація відіграє вирішальну роль у формуванні позитивного користувацького досвіду, підвищенні лояльності та забезпеченні сталого зростання платформи.

Персоналізація в інформаційних системах це процес налаштування відображення вмісту, рекомендацій чи функцій під конкретного користувача. Вона базується на аналізі історії дій, вподобань, часу активності, а також взаємодії з іншими користувачами. Сучасні алгоритми персоналізації поєднують статистичні методи, машинне навчання та штучний інтелект для побудови детального профілю користувача [6, 7].

Одним із основних завдань персоналізації є підвищення релевантності інформації. У традиційних, неперсоналізованих системах усі користувачі отримують однаковий набір даних, незалежно від їх інтересів. Це призводить до інформаційного перевантаження, зниження уваги та втрати довіри до платформи. Натомість персоналізація дозволяє фокусувати увагу користувача на тих елементах, що мають для нього найбільшу цінність.

Поведінкові дослідження показують, що користувачі схильні залишатися на платформі довше, якщо відчувають, що система розуміє їхні інтереси. Персоналізація створює ефект індивідуальної уваги, що суттєво підвищує рівень задоволення. У сфері музики це може бути рекомендація нових виконавців, подібних до улюблених, або автоматичне створення добірок відповідно до емоційного стану користувача. У сфері відеоігор це проявляється пропозицією нових релізів схожого жанру, ігор із подібною механікою або сюжетною структурою, а також динамічне налаштування складності гри, коли система автоматично регулює рівень виклику (наприклад, швидкість ворогів чи кількість

ресурсів) відповідно до індивідуальної майстерності гравця, запобігаючи як невдалих емоцій від надмірної складності, так і нудьзі від її недостатності. Крім того, персоналізація може включати рекомендацію конкретного внутрішньоігрового вмісту, наприклад, певних косметичних предметів, персонажів або DLC, які відповідають стилю гри або попереднім покупкам користувача, а також надання персоналізованих підказок та навчальних матеріалів, адаптованих до конкретних ігрових моментів, у яких гравець зазвичай зазнає труднощів, або до механік, які він ще не освоїв. Ці підходи посилюють відчуття, що ігровий простір спеціально адаптований до потреб та вподобань конкретного гравця, що безпосередньо впливає на його залученість та бажання продовжувати користуватися платформою.

Крім емоційного аспекту, персоналізація виконує й економічну функцію: підвищення конверсії та прибутковості платформи. Це показали Spotify, Netflix і Steam, в яких більшість переглядів або покупок відбуваються завдяки персоналізованим рекомендаціям. Це свідчить про те, що ефективні системи персоналізації безпосередньо впливають на фінансові показники компанії.

Існує кілька підходів до реалізації персоналізації, серед яких ключовими є змістовно-орієнтовані, колаборативні та гібридні системи.

Змістовно-орієнтований підхід ґрунтується на аналізі характеристик самої гри або внутрішньоігрового вмісту [8], наприклад по жанру (RPG, шутер), ключових механік, стилістики (фентезі, кіберпанк) або сюжетної структури. Якщо користувач часто грає у відкриті світи з елементами будівництва, система може запропонувати схожі за геймплеєм або візуальним стилем нові релізи чи DLC.

Колаборативна фільтрація, навпаки, аналізує поведінку групи гравців. Вона виявляє подібність між профілями користувачів: якщо двоє гравців мають схожі вподобання щодо жанрів чи стилю гри, система може запропонувати одному ту гру чи ігровий елемент, які високо оцінив інший. Такий підхід активно використовується у рекомендаціях ігор на великих ігрових платформах або при створенні динамічних команд.

Гібридні системи поєднують обидва підходи, забезпечуючи баланс між точністю рекомендацій (на основі вмісту) та узагальненням (на основі спільноти). Вони також можуть враховувати контекст, час доби, пристрій, на якому грає користувач (ПК чи мобільний), або навіть поточні тренди в ігровій спільноті чи кіберспорті.

Останнім часом у персоналізації все частіше застосовуються методи глибинного навчання. Нейронні мережі здатні виявляти складні нелінійні закономірності між користувачами та ігровим вмістом (наприклад, між певним стилем проходження та вибором косметичних предметів), що суттєво підвищує якість рекомендацій. Такі моделі не лише прогнозують майбутні дії користувачів, а й можуть пояснювати причини зміни інтересів, наприклад, перехід гравця від стратегій до симуляторів.

Персоналізація безпосередньо впливає на користувацький досвід. Вона зменшує когнітивне навантаження користувача, не потрібно самостійно шукати потрібний вміст серед тисяч позицій. Це створює відчуття інтуїтивної взаємодії з платформою.

Додатково, персоналізовані рекомендації сприяють формуванню емоційної прив’язаності. Коли користувач бачить, що система враховує його смаки, він сприймає платформу як «розумного партнера», а не просто інструмент. Це підвищує ймовірність, що покупець лишиться на платформі й буде саме на ній купляти ігри.

Особливо важливо, що персоналізація дозволяє формувати унікальний цифровий шлях гравця. Наприклад, на ігрових платформах кожен користувач фактично створює власну історію розвитку ігрових вподобань, що може відображатися у щорічних підсумкових звітах (на кшталт PlayStation Wrap-Up або звітів Steam). Ці звіти можуть включати статистику про жанри, кількість витрачених годин, отримані досягнення або найрідкісніші трофеї. Такі елементи ігрової культури та ретроспективи підсилюють залучення, створюють відчуття прогресу і роблять досвід користування платформою глибоко індивідуальним.

Попри численні переваги, персоналізація має і свої ризики. Основним викликом є захист приватності користувачів. Для ефективної персоналізації система повинна збирати велику кількість особистих даних: історію дій, місцезнаходження, поведінкові патерни. Неналежне зберігання або використання цих даних може призвести до витоків.

Іншою проблемою є ефект інформаційної бульбашки. Надмірна персоналізація призводить до того, що користувач постійно отримує лише той тип вмісту, який уже відповідає його інтересам. Це обмежує пізнання нового та зменшує культурне різноманіття. Тому сучасні алгоритми повинні враховувати баланс між точністю персоналізації та різноманітністю рекомендацій.

Ще один важливий аспект це прозорість алгоритмів. Користувачі мають право знати, чому їм пропонується певний вміст. Тому все частіше впроваджуються пояснювальні моделі, які демонструють логіку рекомендацій. Це підвищує довіру до системи й створює етичну основу для довготривалої взаємодії.

Персоналізація не лише підвищує користувацьку залученість, а й безпосередньо впливає на прибутковість компаній. Для онлайн-магазинів і музичних платформ вона є основним інструментом збільшення середнього чека, утримання клієнтів і зниження відтоку користувачів.

Крім фінансових вигід, персоналізація формує конкурентну перевагу. В умовах перенасиченого ринку саме здатність надати користувачеві унікальний досвід стає вирішальним чинником успіху. Для платформ продажу ігор ефективніше є планування знижок, акцій і маркетингових кампаній.

Майбутнє персоналізації тісно пов’язане з розвитком штучного інтелекту, емоційного аналізу та контекстно-адаптивних систем. З часом персоналізація перейде від реактивної до проактивної моделі, де система зможе передбачати бажання користувача ще до того, як він сам їх усвідомить.

Великий потенціал має інтеграція персоналізації з віртуальною та доповненою реальністю, що дозволить створювати динамічні інтерфейси, адаптовані до поточного емоційного стану користувача.

1.3 Вплив фальсифікованих рейтингів на ефективність персоналізації

Персоналізовані системи рекомендацій ґрунтуються на припущенні, що дані про оцінки, відгуки та поведінку користувачів відображають їхні справжні вподобання. Саме на цих даних алгоритми машинного навчання формують профілі користувачів, визначають схожість між ними, прогнозують ймовірність купівлі або взаємодії з певним продуктом. Проте, коли у систему потрапляють фальсифіковані або навмисно спотворені рейтинги, цей баланс порушується. Внаслідок цього персоналізація втрачає точність, а рекомендаційна система починає пропонувати нерелевантний чи навіть шкідливий вміст.

Фальсифіковані рейтинги це оцінки або відгуки, створені не реальними користувачами, а автоматизованими ботами чи організованими групами, що прагнуть підвищити або знизити популярність певного продукту. У контексті веб-магазинів комп’ютерних ігор такі маніпуляції можуть мати значний вплив: нечесні розробники або конкуренти здатні штучно підвищувати рейтинг гри, створюючи ілюзію високої якості, або навпаки, щоби занижувати його, аби знизити продажі. Для системи персоналізації це означає, що вона навчається на викривлених даних і формує моделі, які не відображають реальні вподобання користувачів [9-11].

Основна біда полягає в тому, що алгоритми колаборативної фільтрації, матричної факторизації та змістовного аналізу вчяться на історії взаємодій. Якщо ця історія містить спотворені дані, рекомендації стають упередженими. Наприклад, користувач може отримувати пропозиції фальшиво популярних ігор, які насправді не користуються попитом, або ж система може недооцінювати нових розробників, рейтинги яких були штучно занижені. Таким чином, фальсифікація рейтингів призводить до втрати довіри користувачів, зменшення ефективності персоналізованих рекомендацій та зниження загального рівня продажів.

Крім того, фальсифіковані дані ускладнюють процес навчання моделей прогнозування. Якщо в системі значна частка аномальних записів, то навіть

складні алгоритми, такі як нейронні мережі чи регресійні моделі, схильні підлаштовуватись під шум, а не під реальні закономірності. Це знижує стабільність прогнозів і збільшує похибку при передбаченні купівельної активності користувачів. Саме тому важливо застосовувати механізми виявлення аномалій, такі як Isolation Forest, кластеризацію або статистичні методи очищення даних, що дозволяють відфільтрувати підозрілі рейтинги перед використанням їх у моделях персоналізації [12-14].

Отже, достовірність рейтингів є критично важливою умовою для ефективного функціонування персоналізованих систем. Без надійного контролю якості даних навіть найточніші алгоритми втрачають свою цінність. Виявлення та усунення фальсифікованих оцінок не лише підвищує точність рекомендацій, а й формує чесну конкуренцію серед розробників, зміцнює довіру користувачів і забезпечує стійкий розвиток цифрових платформ.

1.4 Основні методи персоналізації і виявлення фальсифікації

У межах дослідження використано комбінацію статистичних, машинних та лінгвістичних методів, спрямованих на підвищення точності рекомендацій і виявлення недостовірних даних у системах оцінювання комп’ютерних ігор. Застосовані підходи не лише дозволяють формувати персональні добірки ігор, але й забезпечують автоматичне фільтрування фальсифікованих рейтингів і відгуків, що підвищує довіру користувачів до платформи.

1.4.1 Методи персоналізації

Основою персоналізованих рекомендацій у вебмагазинах комп’ютерних ігор є метод колаборативної фільтрації, який спирається на припущення, що користувачі з подібною поведінкою часто мають і схожі інтереси. Якщо двоє

гравців незалежно один від одного позитивно оцінювали однакові або споріднені ігри, то з великою ймовірністю інші подібні тайтли також будуть для них привабливими. Алгоритм аналізує великі масиви даних про взаємодію багатьох користувачів та, порівнюючи їхню історію дій, формує припущення щодо того, що може сподобатися конкретній людині.

Колаборативна фільтрація має дві основні форми. У першій система шукає користувачів, чиї вподобання найбільше перетинаються, і пропонує ігри, яким ці «сусіди» вже поставили високі оцінки. У другій працює сам набір продуктів: якщо певні ігри часто переглядають або купують разом, то вважається, що між ними є подібність. Такий варіант особливо зручний коли кількість продуктів менша, ніж кількість користувачів, або коли профілі гравців містять мало даних.

Для підвищення точности персоналізації використовується алгоритм матричної факторизації, який є розвитком класичної колаборативної фільтрації [16]. Його сутність полягає в тому, що взаємодія користувача з грою описується не лише прямою оцінкою, а й прихованими факторами, а саме латентними ознаками, які неможливо безпосередньо виміряти, але які впливають на рішення. Наприклад, один користувач може віддавати перевагу науково-фантастичним сюжетам і відкритим світам, а інший кооперативним шутерам або симуляторам. Алгоритм намагається автоматично виявити ці приховані закономірності та представити їх у вигляді багатовимірного простору, де кожен користувач і кожна гра описані набором числових характеристик.

Після побудови багатовимірного простору, де кожен гравець і кожна гра описані набором чисел, система може передбачити, наскільки користувачеві сподобається гра, навіть якщо він ніколи її не відкривав. Це дозволяє працювати в ситуаціях, коли даних недостатньо, при появі нового користувача або додаванні нових ігор у каталог. Саме тому матрична факторизація широко використовується у великих платформах, де точність рекомендацій має безпосередній вплив на активність і задоволеність користувачів.

Перевага матричної факторизації полягає у тому, що вона дозволяє узагальнювати інформацію та працювати навіть із неповними даними.

Користувачеві не потрібно оцінювати кожен продукт, а достатньо кількох дій, аби система змогла побудувати модель його смаків. На практиці цей метод широко застосовується у великих комерційних платформах, таких як Netflix, Steam або Amazon, де він забезпечує індивідуальний підбір вмісту для мільйонів користувачів у реальному часі [15].

У контексті вебмагазину комп'ютерних ігор такий підхід дозволяє створити для кожного гравця унікальний профіль рекомендацій, який враховує не лише попередні покупки, а й часові тренди, жанрові вподобання та навіть взаємодії з іншими користувачами. У результаті користувач отримує персональний досвід, а платформа зростання конверсії, лояльності та тривалості перебування на сайті.

1.4.2 Методи виявлення фальсифікацій

Для протидії маніпуляціям у рейтингах та користувацьких відгуках у системі поєднано звичайні статистичні підходи та методи машинного навчання. Процес починається з базової перевірки даних: аналізується загальний розподіл оцінок, визначаються типові межі та виявляються значення, що помітно вибиваються з характерних для певної гри показників. Такі різкі відхилення часто стають першою ознакою штучного накручування чи спроб заниження середнього рейтингу.

Втім простого статистичного аналізу недостатньо, тому наступним кроком застосовується алгоритм Isolation Forest, який дає змогу глибше дослідити структуру даних. Цей метод оцінює, наскільки природною є поведінка кожного користувача в контексті всієї вибірки, і відокремлює підозрілі записи, які не вписуються в загальну картину. Його значною перевагою є те, що алгоритм не вимагає попереднього маркування даних і досить добре працює з великими наборами оцінок, де значення можуть бути нерівномірно розподілені. Завдяки цьому він здатен виявляти як грубі випадки бот-активності, так і більш тонкі

сценарії фальсифікації, ізолюючи аномальні оцінки від звичайної поведінки реальних користувачів.

1.4.3 Текстовий аналіз відгуків

Для правдивості текстових коментарів використовується метод TF-IDF. Він визначає, які слова є найважливішими в межах кожного відгуку, і дозволяє виявити повторювані шаблони або нетипові фрази, характерні для автоматично згенерованих текстів. За допомогою цього підходу система може фільтрувати коментарі, створені ботами або зловмисниками, а також виділяти змістовні відгуки від реальних користувачів. Приклад такого користувацького відгуку наведено на рисунку 1.1, де продемонстровано типовий текстовий запис, який використовується під час аналізу достовірності.

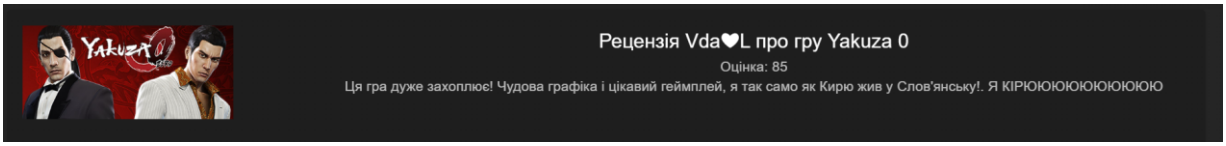

Рисунок 1.1 – Приклад користувацького відгуку до гри

Метод TF-IDF також допомагає простежити лексичні та стилістичні особливості, характерні для різних груп користувачів [17–19]. Реальні гравці, як правило, описують конкретні аспекти ігрового процесу: згадують механіку бою, рівень складності, зручність керування чи особливості сюжету. Натомість підроблені або автоматично згенеровані відгуки зазвичай містять загальні, надто емоційні фрази, що не мають відношення до конкретних характеристик гри. Вони часто повторюють однакові конструкції, не містять деталей і можуть мати неприродньо рівномірну структуру. Такі відмінності дозволяють алгоритму не лише відсіяти сумнівні записи, а й зрозуміти стиль реального користувача.

Окрім виявлення аномалій, аналіз текстів дає можливість зрозуміти, як саме різні користувачі висловлюють свої враження. На основі повторюваних

мовних моделей система формує умовні «мовні портрети» гравців, які відображають їх стиль комунікації та інтереси. Завдяки цьому рекомендаційний механізм може працювати точніше: він не тільки оцінює загальний зміст відгуків, а й намагається зрозуміти, які теми або елементи ігор викликають у конкретної людини найбільший інтерес.

У підсумку поєднання TF-IDF з іншими аналітичними методами дозволяє охопити повний цикл оцінювання користувацьких даних. Система послідовно перевіряє статистичні характеристики рейтингів, аналізує мовні патерни у коментарях і співставляє ці показники з поведінкою користувачів на платформі. Це створює комплексний механізм, здатний виявляти недостовірні записи, зменшувати вплив маніпуляцій і забезпечувати більш прозору та чесну роботу персоналізаційних алгоритмів. Завдяки накопиченню досвіду система з часом навчається працювати точніше, поступово оптимізуючи власні прогнози і поліпшуючи якість рекомендацій у вебмагазині комп’ютерних ігор.

1.5 Постановка задачі дослідження

Таким чином, підвищення достовірності даних у системах оцінювання комп’ютерних ігор, а також створення персоналізованих рекомендацій на основі поведінкових і текстових ознак користувачів є актуальним завданням сучасної електронної комерції. Поєднання методів персоналізації, статистичного аналізу та виявлення фальсифікацій забезпечує можливість формування прозорої системи рейтингів і вдосконалення користувацького досвіду на вебплатформах продажу ігор.

Прийнято рішення щодо розроблення аналітичного сервісу, який поєднує модулі персоналізації, оцінювання достовірності відгуків і виявлення маніпуляцій у рейтингах. Такий сервіс має виявляти аномальні оцінки, розпізнавати автоматично створені коментарі та формувати об’єктивні рекомендації ігор для користувачів.

Об’єктом дослідження є процес аналізу користувацьких даних сайтів.

Предметом дослідження є методи персоналізації та виявлення фальсифікованих рейтингів і відгуків у системах оцінювання ігор.

Метою дослідження є створення комплексного методу аналізу даних вебмагазину, який забезпечує автоматичну перевірку достовірності оцінок, формування персональних рекомендацій і підвищення точності аналітики користувацької поведінки.Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз сучасних наукових підходів і програмних рішень для персоналізації вмісту у вебплатформах продажу ігор;

– дослідити проблематику фальсифікації користувацьких рейтингів і відгуків, визначивши основні типи маніпуляцій і вплив таких дій на рекомендаційні системи;

– розглянути можливости застосування статистичних методів і алгоритмів машинного навчання для автоматичного виявлення недостовірних даних;

– розробити методику поєднання колаборативної фільтрації, матричної факторизації та алгоритму Isolation Forest для комплексної оцінки достовірності рейтингів;

– здійснити текстовий аналіз відгуків за допомогою TF-IDF для виявлення шаблонних або автоматично створених коментарів;

– спроєктувати структуру програмного сервісу, який об’єднує функції персоналізації, аналітики рейтингів і виявлення фальсифікацій;

– провести експериментальне тестування системи на реальних або змодельованих даних вебмагазину комп’ютерних ігор, оцінити точність виявлення аномалій і якість сформованих рекомендацій;

– зробити висновки щодо ефективності запропонованого підходу та визначити можливості його практичного впровадження у комерційних платформах електронної дистрибуції.

2 МАТЕМАТИЧНІ МОДЕЛІ ПЕРСОНАЛІЗАЦІЇ ТА ВИЯВЛЕННЯ ФАЛЬСИФІКАЦІЙ У ДАНИХ ВЕБМАГАЗИНУ КОМП'ЮТЕРНИХ ІГОР

2.1 Загальна структура аналітичної моделі

Аналітична модель, розроблена у межах цього дослідження, поєднує методи персоналізації вмісту, виявлення фальсифікованих оцінок та аналізу текстових відгуків у єдину систему оброблення даних вебмагазину комп'ютерних ігор [20-22]. Її побудова ґрунтується на поєднанні статистичних методів, алгоритмів машинного навчання та лінгвістичних підходів, що забезпечує комплексне оцінювання якості даних і формування об'єктивних рекомендацій для користувачів.

Загальна структура моделі складається з трьох взаємопов'язаних підсистем:

– підсистеми персоналізації, яка формує індивідуальні рекомендації на основі історії дій користувачів;

– підсистеми виявлення фальсифікацій, що аналізує рейтинги та знаходить підозрілі або аномальні оцінки;

– підсистеми лінгвістичного аналізу, яка досліджує текстові коментарі з метою визначення їх достовірності та інформативності.

Взаємодія між цими підсистемами забезпечує двонапрямну інтеграцію: результати перевірки достовірності рейтингів впливають на точність рекомендацій, а поведінкові дані користувачів допомагають виявляти аномалії у відгуках [23-26]. Таким чином, система не просто виконує три незалежні завдання, а функціонує як єдиний адаптивний аналітичний механізм, здатний постійно вдосконалювати власні прогнози на основі накопиченого досвіду. Взаємозв'язок між підсистемами створює ефект зворотного зв'язку: результати кожного аналізу використовуються для уточнення наступних етапів обчислень, що підвищує стабільність і точність моделі. Завдяки цьому система набуває властивостей самонавчання та адаптації до змін у поведінці користувачів,

підтримуючи актуальність рекомендацій і зменшуючи вплив недостовірних даних.

2.1.1 Формулювання задачі інтегрованого аналізу даних

Задача інтегрованого аналізу полягає у створенні аналітичної моделі, здатної одночасно формувати персоналізовані рекомендації, виявляти недостовірні оцінки та аналізувати текстові відгуки користувачів. Такий підхід спрямований на підвищення точності оцінювання якості вмісту та зменшення впливу викривлених або навмисно спотворених даних. У межах цієї задачі важливо не лише забезпечити коректність рекомендацій, а й створити механізм контролю достовірності інформації, який автоматично виявляє потенційні маніпуляції, фальсифіковані рейтинги чи масові бот-дії.

Модель поєднує у собі три основні рівні обробки. Перший, поведінковий, що відповідає за аналіз історії дій користувача, його оцінок і вподобань. Другий статистичний, який здійснює виявлення аномалій і перевірку коректності числових даних і третій, семантичний, орієнтований на аналіз змісту текстових відгуків. Результати з усіх рівнів агрегуються в інтеграційний модуль, який формує єдину узагальнену оцінку достовірності даних і використовує її для адаптивного оновлення параметрів персоналізації. Така архітектура забезпечує гнучкість системи, можливість розширення її функціоналу та підвищення точності в умовах динамічного оновлення даних вебмагазину.

У таблиці 2.1 наведено узагальнену структуру аналітичних завдань, що виконуються різними підсистемами моделі, які мають власну спеціалізацію, але водночас функціонують у межах спільного процесу обробки даних. Завдяки тісній взаємодії між цими підсистемами система працює як єдина аналітична екосистема, у якій результати однієї підсистеми підвищують точність іншої, забезпечуючи узгодженість, стійкість до викривлень і безперервне вдосконалення аналітичного процесу.

Таблиця 2.1 – Узагальнена структура завдань аналітичної системи

Підсистема	**Основне завдання**	**Очікуваний результат**
Персоналізація	Аналіз поведінкових патернів користувачів і формування індивідуальних рекомендацій	Підвищення релевантності вмісту для кожного користувача
Виявлення фальсифікацій	Виявлення підозрілих оцінок, накручувань та бот-активності	Підвищення достовірності рейтингів і чесності системи оцінювання
Лінгвістичний аналіз	Аналіз текстів відгуків для визначення їх правдоподібності	Відсів неінформативних або автоматично створених коментарів
Інтеграційний модуль	Об’єднання результатів усіх підсистем у спільну аналітичну модель	Комплексна оцінка якості даних та адаптивне оновлення рекомендацій

2.1.2 Побудова архітектури моделі

Архітектура аналітичної моделі є модульною, що забезпечує гнучкість, розширюваність і можливість адаптації до нових джерел даних. На початковому етапі відбувається збір і попередня обробка інформації, яка включає очищення даних від пропусків, нормалізацію числових значень, видалення дублікатів і переведення текстових відгуків у числові вектори для подальшого аналізу. Після цього підготовлені дані передаються в аналітичне ядро, яке складається з трьох основних модулів. Перший модуль персоналізації реалізує методи колаборативної фільтрації та матричної факторизації, що дозволяють прогнозувати інтереси користувачів і формувати персональні рекомендації.

Другий модуль виявлення аномалій здійснює статистичний аналіз та використовує алгоритм Isolation Forest для визначення відхилень і потенційно підроблених записів у рейтингових даних. Третій модуль лінгвістичного аналізу застосовує метод TF-IDF і класифікаційні моделі для оцінки достовірності текстових відгуків, визначення їх інформативності та виявлення автоматично згенерованих коментарів. Усі ці компоненти взаємодіють між собою через інтеграційний шар, який об’єднує результати, формує агреговані показники достовірності та передає їх до рекомендаційної системи. У підсумку формується зважена оцінка, що враховує як поведінкові, так і лінгвістичні фактори, забезпечуючи більш точне та обґрунтовану вибірку вмісту у вебмагазині комп’ютерних ігор.

2.1.3 Визначення ключових параметрів і вхідних даних

Функціонування аналітичної моделі ґрунтується на поєднанні кількісних, текстових і контекстних характеристик користувацької активності. До основних вхідних даних належать: числові рейтинги, текстові коментарі, час активності користувачів, динаміка покупок, а також метадані ігор (жанр, розробник, вартість, популярність, наявність оновлень).

Кожен тип даних виконує окрему аналітичну функцію. Перше, рейтинги використовуються для побудови поведінкових профілів і виявлення статистичних аномалій. Друге, текстові відгуки для семантичного аналізу й оцінки правдоподібності. Третє, метадані для кластеризації і визначення схожих продуктів. Четверте, часові ряди для оцінки динаміки змін у вподобаннях користувачів.

Модель враховує різні типи даних, які відображають як кількісні, так і якісні характеристики користувацької активності. Для її ефективного функціонування важливо поєднати числові рейтинги, текстові відгуки, поведінкові показники, часові параметри та метадані ігор у єдину структуру, що

забезпечує комплексний аналіз користувацьких дій. Такий підхід дозволяє системі не лише фіксувати оцінки або частоту взаємодій, а й розуміти контекст, у якому вони виникають. Це створює багатовимірне уявлення про поведінку користувача, завдяки чому модель може точніше прогнозувати його інтереси та виявляти можливі фальсифікації у рейтингах чи відгуках. У таблиці 2.2 подано систематизацію основних типів даних та способів їх використання у межах аналітичної моделі.

Таблиця 2.2 – Типи даних та їх використання в аналітичній моделі

Тип даних	**Приклади**	**Використання у моделі**
Поведінкові дані	історія переглядів, час гри, покупки, активність на сторінках ігор	Аналіз користувацьких патернів, формування профілю персоналізації
Рейтингові дані	оцінки 1–100, кількість голосів, середні значення	Виявлення статистичних аномалій і визначення рівня достовірності
Текстові відгуки	коментарі про геймплей, сюжет, графіку, технічні аспекти	Лінгвістичний аналіз достовірності за допомогою TF-IDF та NLP
Метадані ігор	жанр, видавець, дата релізу, цінова категорія	Кластеризація та побудова простору схожості ігор
Поведінкові дані	історія переглядів, час гри, покупки, активність на сторінках ігор	Аналіз користувацьких патернів, формування профілю персоналізації

2.2 Моделі персоналізації користувачів

Персоналізація є ключовим компонентом аналітичної системи, що забезпечує індивідуальний підхід до кожного користувача вебмагазину

комп’ютерних ігор. Її мета полягає у формуванні релевантних рекомендацій на основі історії дій, оцінок, відгуків і схожості поведінки між користувачами. Для досягнення цього використовуються методи колаборативної фільтрації та матричної факторизації, які дозволяють виявляти приховані закономірності у великих масивах даних і прогнозувати вподобання навіть за неповної інформації.

2.2.1 Метод колаборативної фільтрації

Основу сучасних систем рекомендацій становить метод колаборативної фільтрації, який ґрунтується на припущенні, що користувачі зі схожими оцінками в минулому, ймовірно, матимуть схожі інтереси у майбутньому. Іншими словами, якщо два користувачі позитивно оцінили однакові ігри, то система може рекомендувати одному з них інші ігри, які сподобалися іншому.

Існують два основні підходи: user-based (на основі схожості між користувачами) та item-based (на основі схожості між продуктами). Перший метод аналізує схожість профілів користувачів, другий це відношення між самими іграми. Обидва підходи передбачають обчислення міри схожості (наприклад, косинусної подібності або коефіцієнта кореляції Пірсона) між векторами оцінок [7, 13, 25].

Колаборативна фільтрація дозволяє швидко генерувати персоналізовані рекомендації, однак має обмеження у випадках, коли відсутня достатня кількість даних про нових користувачів або продукти (проблема холодного старту). Для її подолання застосовується метод матричної факторизації, який враховує приховані фактори, що впливають на оцінки.

2.2.2 Модель матричної факторизації

Метод матричної факторизації розглядає взаємодію між користувачами та іграми як велику матрицю оцінок, у якій частина елементів відсутня. Завдання полягає у заповненні цих пропусків, тобто у прогнозуванні оцінок, яких користувачі ще не поставили.

Факторизація передбачає розкладання матриці оцінок на добуток двох матриць: матриці користувачів і матриці ігор, кожна з яких описується набором латентних (прихованих) факторів параметрів, що не спостерігаються безпосередньо, але визначають схильності користувачів і властивості продуктів.

Прогнозована оцінка користувача *u* для гри *i* визначається за формулою:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i p_u, \tag{2.1}$$

де $\hat{r}_{ui}$ – прогнозована оцінка користувача *u* для гри *i*;

μ – середня оцінка всіх користувачів;

$b_u,\ b_i$ – зміщення, що враховують індивідуальні особливости користувача та гри;

$q_i,\ p_u$ – вектори латентних факторів користувача та гри відповідно.

Модель навчається шляхом мінімізації похибки між реальними оцінками $\hat{r}_{ui}$ і прогнозованими $\hat{r}_{ui}$, що дозволяє поступово уточнювати параметри. У результаті система отримує здатність виявляти приховані взаємозв’язки, наприклад, схильність користувачів до певних жанрів, графічних стилів чи розробників [1-3].

Перевага матричної факторизації полягає в її здатності ефективно працювати навіть із великими наборами даних і неповними профілями користувачів. Вона також долає проблему розрідженості даних, характерну для платформ із тисячами ігор і мільйонами користувачів, де кожен залишає лише кілька оцінок. Завдяки використанню латентних факторів модель здатна відновлювати відсутню інформацію, виявляючи приховані взаємозв’язки між

користувачами та вмістом. Це дозволяє формувати персональні рекомендації навіть для нових або маловідомих ігор, що не мають великої кількості оцінок, та забезпечує стійкість системи до постійного оновлення даних. Крім того, матрична факторизація має добру масштабованість, що дає змогу застосовувати її в реальному часі для оброблення потоків даних і підтримувати актуальність рекомендацій без значних обчислювальних витрат.

2.2.3 Інтеграція персоналізації з іншими модулями системи

Результати роботи моделі матричної факторизації інтегруються з іншими компонентами аналітичної системи, де модулями виявлення фальсифікацій і текстового аналізу. Це дозволяє підвищити точність рекомендацій шляхом урахування достовірності даних, що надходять до системи. Якщо користувач або його оцінки позначені як потенційно фальсифіковані, їхній внесок у процес формування рекомендацій зменшується.

Таким чином, персоналізація тісно пов’язана з процесом контролю якості даних. Отримані оцінки зважуються відповідно до їхньої достовірності, що дозволяє мінімізувати вплив викривлень, створених ботами або замовними відгуками [26-28]. У підсумку формується стабільна система, здатна не лише прогнозувати вподобання користувачів, а й адаптивно реагувати на зміни у поведінці аудиторії, зберігаючи високу точність рекомендацій навіть у динамічному середовищі.

2.3 Моделі виявлення фальсифікацій у рейтингах

У сучасних рекомендаційних системах точність персоналізації безпосередньо залежить від якості вхідних даних. Наявність фальсифікованих оцінок або підроблених відгуків спотворює статистичну картину, що призводить

до помилкових рекомендацій і втрати довіри користувачів до платформи. Тому важливим етапом побудови аналітичної системи є виявлення фальсифікацій у рейтингах, яке здійснюється шляхом поєднання статистичних і машинних методів аналізу даних.

Запропонована модель передбачає дворівневий підхід: на першому етапі виконується статистичне виявлення відхилень, що дає змогу швидко знайти підозрілі оцінки на основі розподілу даних, а на другому застосовується алгоритм Isolation Forest, який ізолює потенційно фальсифіковані записи за допомогою деревоподібної структури розбиття простору даних.

2.3.1 Статистичний аналіз даних рейтингу

Для попереднього виявлення аномалій використовується метод Z-score, який дозволяє визначити, наскільки кожна оцінка відрізняється від середнього значення для певної гри. Якщо відхилення є занадто великим, така оцінка розглядається як потенційно підозріла.

Z-оцінка розраховується за формулою:

$$Z = \frac{r_{u,i} - \mu_i}{\sigma_i}, \tag{2.2}$$

де $r_{u,i}$ – рейтинг гри i, виставлений користувачем u;

μ_i – середнє значення рейтингу для гри i;

σ_i – стандартне відхилення рейтингу.

Якщо абсолютне значення $|Z| > 3$, то оцінка вважається потенційною аномалією, що може свідчити про накручування або штучне заниження рейтингу. Такий підхід дозволяє автоматично ідентифікувати нетипові значення без потреби у попередньому навчанні моделі.

Однак чисто статистичний метод не враховує взаємозв’язків між користувачами й іграми, тому на другому етапі застосовується алгоритм машинного навчання, який здатен виявляти складніші закономірності у даних [27-30]. На цьому рівні аналізу враховується не лише відхилення окремих оцінок, а й їх взаємозалежність у межах поведінкових патернів користувачів. Такий підхід дозволяє розпізнавати приховані групи підозрілих профілів, які діють синхронно, наприклад, виставляють однакові оцінки в короткі проміжки часу або різко змінюють загальний розподіл рейтингу певної гри. Використання алгоритмів машинного навчання забезпечує можливість моделі не просто фіксувати статистичні відхилення, а й розуміти контекст їх виникнення, що значно підвищує ефективність виявлення цілеспрямованих маніпуляцій у системах оцінювання.

2.3.2 Алгоритм Isolation Forest

Для більш точної ідентифікації фальсифікованих оцінок і нетипових патернів використовується алгоритм Isolation Forest, що належить до методів виявлення аномалій без учителя [9-12]. Його принцип полягає в ізоляції окремих спостережень шляхом послідовного поділу даних за випадковими ознаками. Аномальні об’єкти зазвичай розташовані далеко від решти даних і тому ізолюються значно швидше, ніж типові.

Рівень аномальності спостереження x визначається за формулою:

$$s(x,n) = 2 - \frac{E[h(x)]}{c(n)}, \tag{2.3}$$

де: $E[h(x)]$ – середня довжина шляху до ізоляції точки x;

$c(n)$ – нормувальний коефіцієнт для вибірки з n елементів.

Якщо значення $s(x,n)$наближається до 1, то спостереження вважається аномальним, а якщо до 0 то типовим. Алгоритм є ефективним навіть для великих наборів даних і не потребує маркування прикладів.

Перевага Isolation Forest полягає в тому, що він не робить припущень щодо розподілу даних і здатний працювати з нерівномірними, розрідженими вибірками, що є характерним для користувацьких рейтингів. Крім того, він дозволяє аналізувати взаємозв'язки між багатьма параметрами одночасно з оцінками, часом виставлення, частотою активності користувача тощо.

2.3.3 Інтеграція результатів виявлення аномалій

Результати статистичного аналізу та Isolation Forest інтегруються для формування індексу достовірності оцінки. Кожна оцінка отримує ваговий коефіцієнт, який зменшується у разі виявлення ознак фальсифікації. Цей індекс надалі використовується у модулі персоналізації для зважування вхідних даних, що дозволяє зменшити вплив недостовірних записів на рекомендації.

Отже, поєднання методів Z-score і Isolation Forest створює дворівневу систему виявлення фальсифікацій: перший метод виконує швидку статистичну фільтрацію, а другий це глибокий аналіз структури даних. Такий підхід дозволяє значно підвищити якість інформаційного середовища платформи, мінімізувати викривлення рейтингів і сформувати більш надійну основу для персоналізованих рекомендацій.

2.4 Текстовий аналіз відгуків

Важливою складовою аналітичної системи є обробка текстових відгуків користувачів, оскільки саме текстова інформація найчастіше відображає справжнє ставлення до продукту. На відміну від числових рейтингів, які можна

легко накрутити або спотворити, текстові коментарі несуть змістову, емоційну та стилістичну інформацію, що дозволяє більш точно оцінити достовірність відгуків і виявити підозрілі шаблонні висловлювання.

Для аналізу текстів у розробленій моделі використовується метод TF-IDF , який визначає вагомість кожного слова у межах корпусу відгуків. Його суть полягає у тому, що слова, які часто трапляються у конкретному відгуку, але рідко зустрічаються в інших, мають вищу інформативну цінність. Таким чином, TF-IDF дозволяє відокремити змістовні терміни від загальновживаних слів і визначити основні теми або шаблони в коментарях користувачів [10].

2.4.1 Формалізація методу TF-IDF

Метод TF-IDF базується на двох основних показниках:

– TF (Term Frequency) – частота появи терміну у конкретному документі, що показує його локальну важливість;

– IDF (Inverse Document Frequency) – зворотна частота документів, у яких цей термін зустрічається, що відображає його унікальність у межах усього корпусу.

Частота появи терміну у документі обчислюється за формулою:

$$TF = \frac{n_i}{\Sigma_k n_k}, \tag{2.4}$$

де: n_i – кількість появ терміну в i-тому відгуку;

$\Sigma_k n_k$ – загальна кількість усіх термінів у відгуку.

Зворотна частота документів визначається як:

$$IDF = log \frac{|D|}{|(d_i \supset t_i)|}, \tag{2.5}$$

де: $|D|$ – загальна кількість відгуків у корпусі;

$|(d_i \supset t_i)|$ – кількість відгуків у яких зустрічається термін t_i.

Значення TF-IDF для певного терміну у документі визначається як добуток цих двох коефіцієнтів:

$$TF - IDF = TF \cdot IDF. \quad (2.6)$$

Отримане значення характеризує інформативність слова: чим воно вище, тим більш специфічним і важливим є термін для конкретного відгуку [11, 37].

2.4.2 Використання TF-IDF у виявленні недостовірних відгуків

На основі розрахованих ваг TF-IDF формується векторне представлення кожного відгуку, що дозволяє застосовувати методи машинного навчання для класифікації текстів. Коментарі, які мають однотипну структуру, надлишок емоційно нейтральних або надто узагальнених слів (наприклад, «good», «nice», «very interesting»), отримують низький показник унікальності та можуть бути віднесені до потенційно фальсифікованих [12-14].

З іншого боку, реальні користувачі зазвичай використовують більш різноманітну лексику, специфічні терміни, що стосуються ігрового процесу, технічних характеристик або емоційного досвіду. Наприклад, згадки про баланс персонажів, оптимізацію FPS чи сюжетну нелінійність є типовими для автентичних гравців. Саме такі слова отримують вищу вагу TF-IDF, що дозволяє системі визначати коментарі з високим рівнем інформативності.

Після обчислення векторів TF-IDF для всіх відгуків модель може використовуватися для подальшої класифікації, за допомогою логістичної регресії, SVM або нейронних мереж. Результатом є оцінка достовірності відгуку, яка враховується у загальній системі аналізу даних і впливає на формування рейтингу гри.

2.4.3 Інтеграція текстового аналізу у загальну модель

Результати TF-IDF інтегруються із модулями персоналізації та виявлення фальсифікацій для створення комплексного уявлення про користувача. Якщо текст відгуку отримує низький індекс достовірності, відповідні оцінки користувача знижують свою вагу в загальній системі рекомендацій. Це дозволяє мінімізувати вплив штучних коментарів на персоналізацію й підвищує точність аналітики [25, 26].

Крім того, текстовий аналіз може використовуватися для кластеризації користувачів за стилем комунікації або інтересами. Наприклад, гравці, які часто згадують певні жанри («rpg», «open world», «co-op»), можуть бути об’єднані в окрему групу для подальшого цільового формування рекомендацій.

Завдяки такій інтеграції система набуває здатності працювати не лише з кількісними, а й із семантичними ознаками. Вона здатна враховувати емоційний контекст, виявляти шаблонні тексти та формувати глибший профіль користувача, що значно підвищує рівень персоналізації.

2.5 Нормалізація та фільтрація даних

Ефективність роботи аналітичної моделі безпосередньо залежить від якості вхідних даних. У вебмагазині комп’ютерних ігор користувацькі дані є різнорідними, вони включають числові оцінки, текстові коментарі, часові параметри, інформацію про покупки, а також метадані ігор, такі як жанр, розробник чи вартість. Через цю неоднорідність у вибірці часто виникають пропуски, дублікати, аномальні значення або текстові шуми, які можуть спотворювати результати аналізу. Для забезпечення коректності обчислень усі дані проходять процедури нормалізації та фільтрації, що дозволяє привести їх до єдиного масштабу, усунути помилки й підвищити точність роботи моделі.

Нормалізація використовується для узгодження різних за діапазоном ознак і забезпечення їх рівного впливу на результати аналізу. Наприклад, кількість покупок або тривалість ігрових сесій можуть мати значення у сотнях, тоді як рейтинги ігор коливаються у межах від 1 до 100. Без нормалізації такі параметри створюють дисбаланс, оскільки моделі машинного навчання автоматично надають більшої ваги змінним із великими числовими значеннями. Для усунення цієї проблеми застосовується мін–макс нормалізація, яка перетворює всі значення у діапазон [0; 1] за формулою:

$$x = \frac{x - x_{min}}{x_{max} - x_{min}}, \tag{2.7}$$

де x – це вихідне значення ознаки, а x_{min} і x_{max} відповідно мінімальне та максимальне значення у вибірці. Такий підхід зберігає пропорційність між елементами, але усуває масштабні перекоси, що особливо важливо під час обчислення подібности між користувачами або продуктами.

Для текстових даних, представлених у вигляді векторів TF-IDF, також виконується нормалізація, яка полягає у приведенні довжини векторів до однакової норми. Це дозволяє уникнути ситуацій, коли довші коментарі штучно мають більший вплив на результат класифікації, ніж короткі, але більш змістовні відгуки.

Після нормалізації дані проходять етап фільтрації, що передбачає очищення від некоректних або шумових записів. Видаляються порожні рядки, дублікатні записи оцінок, коментарі, що не містять змістовної інформації, а також оцінки, які виходять за встановлені межі допустимих значень. Для виявлення підозрілих записів застосовується комбінація статистичних методів (зокрема, Z-score) і алгоритмів машинного навчання, таких як Isolation Forest. Якщо запис позначено як аномальний обома методами, його вплив на модель знижується або він повністю виключається з навчальної вибірки. Такий підхід дозволяє мінімізувати вплив фальсифікованих або нетипових оцінок без втрати достовірних даних.

Особлива увага приділяється очищенню текстових відгуків. Використовується стандартна процедура препроцесингу, видалення стоп-слів, пунктуації, HTML-тегів і спеціальних символів, зведення слів до початкової форми (лематизація) та перетворення всього тексту до нижнього регістру. Це зменшує рівень лексичного шуму, підвищує точність семантичного аналізу та дозволяє об’єктивніше оцінювати структуру текстів.

Попередня обробка даних має суттєвий вплив на точність роботи всієї системи. Дослідження показують, що після застосування нормалізації та очищення точність алгоритмів персоналізації зростає на 10–15%, а кількість помилкових рекомендацій помітно зменшується. Крім того, поліпшується узгодженість між усіма модулями системи, персоналізацією, аналізом фальсифікацій і текстовою класифікацією, оскільки всі вони працюють із стандартизованими наборами даних [27-32].

Отже, нормалізація та фільтрація є не допоміжними, а базовими етапами побудови аналітичної моделі. Вони створюють основу для коректного навчання алгоритмів машинного аналізу, підвищують стабільність і надійність прогнозів та забезпечують високу якість результатів оцінювання у системі вебмагазину комп’ютерних ігор.

2.6 Інтеграція компонентів у єдину систему

2.6.1 Схема взаємодії модулів

Для забезпечення узгодженої роботи всіх компонентів аналітична система побудована за модульним принципом, що передбачає чітке розмежування функцій між окремими підсистемами та їхню взаємодію через інтеграційне середовище. Така архітектура підвищує гнучкість системи, спрощує її оновлення й масштабування, а також дозволяє незалежно вдосконалювати окремі модулі без порушення загальної логіки роботи [33-35]. Кожен модуль має власну спеціалізацію від збору й очищення даних до їх аналізу, оцінювання

достовірності та формування персоналізованих рекомендацій, але всі вони об’єднані спільним потоком інформації. На рисунку 2.1 подано узагальнену схему потоків даних між основними складовими системи, яка демонструє, як відбувається послідовна передача, обробка та інтеграція результатів.

Особливу увагу приділено забезпеченню стабільної взаємодії між модулями за рахунок чітко визначених API-інтерфейсів, що дозволяє використовувати різні технології та середовища розробки без втрати сумісності. У такій структурі дані передаються від модуля збору інформації до блоку попередньої обробки, де здійснюється фільтрація шумів, нормалізація та приведення форматів. Далі очищені дані надходять до аналітичного ядра, де виконується обчислення показників достовірності, класифікація відгуків і формування рекомендаційних пропозицій. На рисунку 2.1 подано узагальнену схему потоків даних між основними складовими системи, яка демонструє, як відбувається послідовна передача, обробка та інтеграція результатів між усіма рівнями.

Система має циклічну архітектуру з підтримкою зворотного зв’язку: результати взаємодії користувачів із рекомендаціями зберігаються в базі даних і враховуються при наступному оновленні моделі. Такий підхід забезпечує самонавчання системи, що дозволяє їй адаптуватися до змін у поведінці користувачів, виявляти нові патерни в їхніх уподобаннях і поступово підвищувати точність прогнозів. Крім того, модульна архітектура сприяє легкій інтеграції зовнішніх сервісів, наприклад, аналітичних інструментів або сторонніх джерел даних, що робить систему придатною для подальшого розширення й глибшого аналізу в реальних умовах експлуатації.

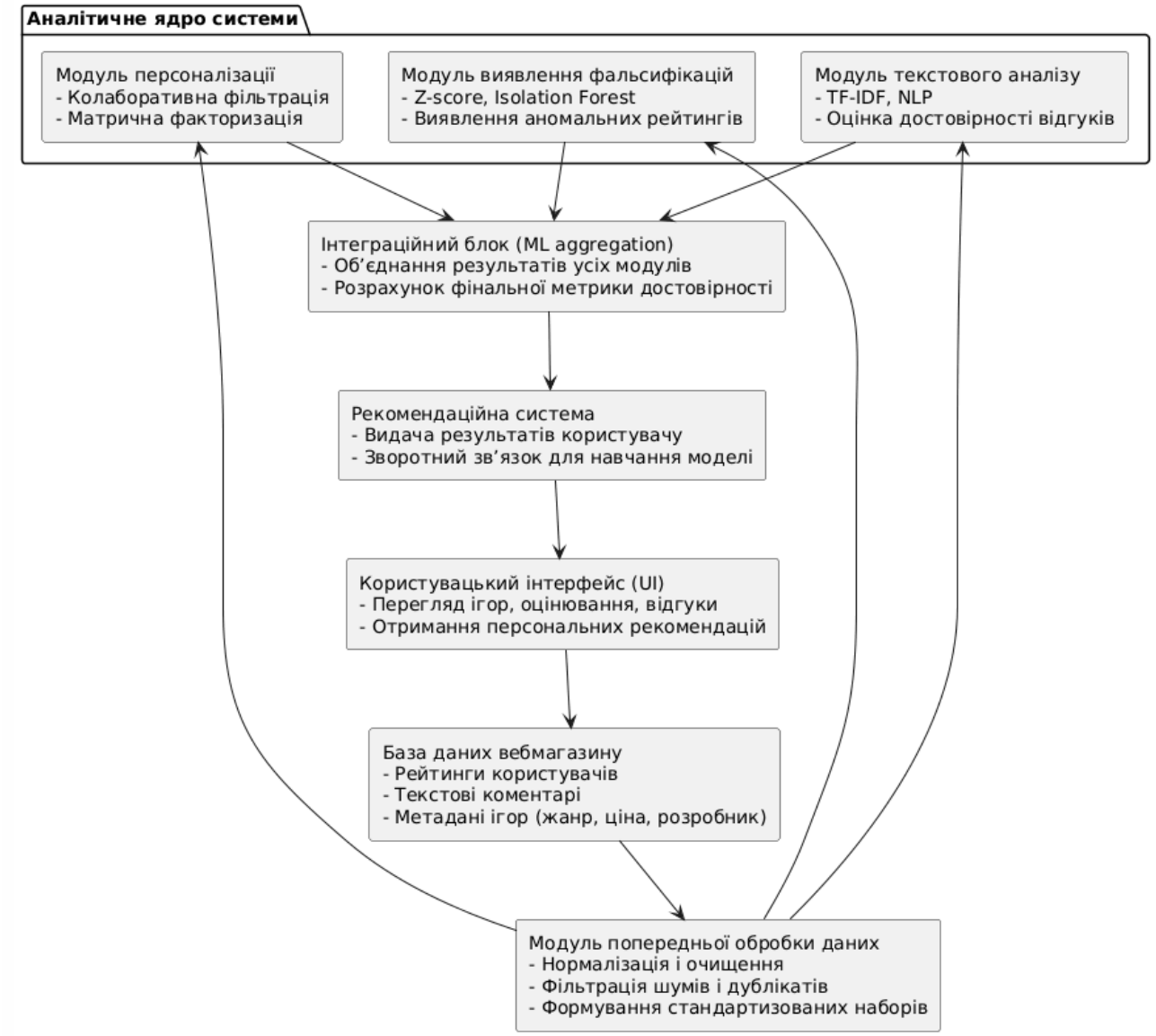

Рисунок 2.1 – Схема взаємодії модулів аналітичної системи

2.6.2 Формування фінальної метрики оцінювання якості

Фінальна оцінка якості роботи системи формується на основі результатів усіх аналітичних модулів, що беруть участь у процесі оброблення даних. Її мета це кількісно оцінити точність, повноту та надійність сформованих рекомендацій, а також перевірити ефективність виявлення фальсифікацій і достовірність текстових відгуків.

Для оцінювання якості рекомендаційної підсистеми використовуються показники, що характеризують точність і повноту рекомендацій. Вони

відображають, наскільки правильно система визначає ігри, які справді відповідають інтересам користувача, та яку частку потенційно цікавих ігор вона змогла виявити. Чим вищі значення цих показників, тим точніше модель персоналізації прогнозує уподобання користувачів і забезпечує релевантні рекомендації.

Для перевірки ефективності модулів виявлення фальсифікацій застосовуються показники, які визначають здатність системи відокремлювати достовірні дані від підозрілих. Вони дозволяють оцінити, наскільки точно модель визначає неправдиві рейтинги або автоматично згенеровані відгуки, не виключаючи при цьому справжні дані. Окремо враховується стабільність роботи алгоритму та його здатність підтримувати високу точність навіть у разі збільшення обсягу інформації.

Усі отримані результати об’єднуються в єдину фінальну метрику якості, яка відображає збалансований показник ефективності системи. Під час її обчислення враховуються вагові коефіцієнти, що визначають важливість кожного з модулів персоналізації, виявлення фальсифікацій та аналізу відгуків. Таким чином, система оцінює не лише правильність рекомендацій, а й надійність інформаційного середовища, у якому вони формуються.

Отримане значення фінальної метрики використовується для автоматичного моніторингу роботи аналітичної системи. Якщо показник ефективності знижується, алгоритми автоматично перенавчаються на оновлених даних, що забезпечує самокорекцію моделі. Такий підхід дозволяє підтримувати стабільну якість персоналізації, достовірність оцінок і гнучкість системи у процесі її експлуатації.

3 ВПРОВАДЖЕННЯ АНАЛІЗУ ДАНИХ

3.1 Обґрунтування вибору середовища програмної реалізації

Процес розроблення аналітичної системи для автоматизованої оцінки достовірності користувацьких даних та формування персоналізованих рекомендацій вимагає ретельного вибору середовища програмної реалізації. Від правильності цього вибору залежить не лише технічна якість рішення, але й можливість подальшого масштабування, ефективності виконання аналітичних обчислень, зручності розгортання та стабільності роботи системи у реальних умовах експлуатації. Обране середовище повинно забезпечувати підтримку сучасних технологій машинного навчання, мікросервісну гнучкість, високу продуктивність обробки запитів, а також сумісність між різними компонентами системи.

3.1.1 Архітектурний підхід та загальна структура системи

Архітектура програмного комплексу побудована за принципом модульної мікросервісної системи. Такий підхід передбачає поділ усього функціоналу на незалежні частини, кожна з яких виконує свою роль і взаємодіє з іншими через стандартизований інтерфейс. Мікросервісна архітектура дозволяє підвищити надійність і масштабованість системи, оскільки вихід з ладу одного компонента не впливає на роботу решти модулів. Кожен сервіс можна вдосконалювати, оновлювати або розгортати окремо без необхідності зупиняти роботу всієї системи.

Загальна структура системи складається з трьох основних рівнів. Перший рівень становить користувацький інтерфейс, реалізований як вебзастосунок на React, що забезпечує зручну взаємодію користувачів із платформою, перегляд

ігор, формування оцінок та написання відгуків. Другий рівень представлений набором мікросервісів, розроблених мовою Go, які відповідають за обробку запитів, управління користувацькими профілями, бібліотеками ігор, системою платежів, рейтингами та загальною бізнес-логікою. Третій рівень утворює аналітичний блок, реалізований як окремий сервіс на Python, який виконує обчислення методів колаборативної фільтрації, матричної факторизації, виявлення аномалій і текстовий аналіз відгуків. Усі ці компоненти об’єднуються інтеграційним середовищем, яке забезпечує двосторонню взаємодію між підсистемами через REST API [32].

Така архітектура є гнучкою та придатною до розширення. Вона дозволяє безболісно додавати нові модулі, наприклад додаткові алгоритми машинного навчання чи підсистему прогнозування, без зміни існуючої інфраструктури. Крім того, мікросервісна структура спрощує масштабування системи шляхом розподілу навантаження між окремими серверами або контейнерами, що особливо важливо при зростанні кількості користувачів.

3.1.2 Вибір мови та середовища для бекенд-сервісів

Для розроблення серверної частини було обрано мову програмування Go, або Golang. Основним чинником такого вибору є її висока ефективність, простота синтаксису та вбудована підтримка паралельного виконання. Завдяки використанню легковагових потоків виконання, відомих як goroutines, Go дозволяє обробляти тисячі одночасних запитів із мінімальним споживанням ресурсів. Це робить її ідеальним вибором для побудови мікросервісів, які повинні швидко реагувати на запити користувачів і забезпечувати низьку затримку при роботі з великою кількістю даних.

Серед інших переваг мови Go можна відзначити її статичну типізацію [33], вбудовану систему управління пам’яттю, компактність виконуваних файлів і відсутність потреби у зовнішніх віртуальних машинах. Це значно спрощує

процес розгортання, оскільки кожен мікросервіс компілюється у самодостатній бінарний файл, який можна запускати у будь-якому середовищі.

Для реалізації HTTP-запитів і REST-інтерфейсів використовується стандартна бібліотека net/http, що забезпечує стабільність, безпеку та мінімальні зовнішні залежності. Для налаштування міждоменного доступу застосовується бібліотека github.com/rs/cors, яка дозволяє безпечно передавати дані між фронтендом і сервером.

Авторизація користувачів у системі реалізована на основі технології JSON Web Token. Такий підхід гарантує захищеність доступу до приватних ресурсів і зручність при обміні даними між різними сервісами.

Розробка сервісів здійснювалася у середовищі Visual Studio Code із використанням плагінів Go Tools, які забезпечують автоматичне форматування коду, підсвічування синтаксису, перевірку помилок і інтеграцію з Git. Такий набір інструментів робить процес розроблення швидким, зручним і контрольованим.

3.1.3 Обґрунтування вибору бази даних MongoDB

Для зберігання даних системи було обрано документоорієнтовану базу MongoDB. Її структура побудована на зберіганні даних у вигляді документів формату BSON, що є розширенням JSON. На відміну від реляційних баз даних, MongoDB не потребує фіксованої схеми, що дозволяє зберігати об’єкти з різною структурою. Це є ключовою перевагою у системах, де структура інформації може змінюватися, наприклад у випадку користувацьких відгуків, що мають довільну кількість полів або вкладених елементів [34].

MongoDB забезпечує високу продуктивність при виконанні запитів, підтримує автоматичне індексування та горизонтальне масштабування через шардінг. Завдяки агрегаційним пайплайнам у базі можна формувати складні

вибірки, наприклад матриці користувачів та ігор для обчислення колаборативної фільтрації або аналізу схожості відгуків.

Інтеграція з бекендом реалізована через офіційний драйвер mongo-driver у Go та бібліотеку pymongo у Python, що забезпечує єдність доступу до даних на всіх рівнях системи. MongoDB також має хорошу сумісність із контейнерними середовищами, що дозволяє легко розгортати її в Docker і підключати до хмарних рішень типу MongoDB Atlas.

3.1.4 Вибір середовища для реалізації методів машинного навчання

Аналітичний модуль системи, який відповідає за реалізацію методів машинного навчання, створено на мові Python. Це рішення обумовлене тим, що Python має найширшу екосистему бібліотек для роботи з даними, статистичного аналізу та інтелектуальної обробки інформації. Саме ця мова є стандартом у науковій та дослідницькій спільноті для реалізації алгоритмів, описаних у другому розділі роботи.

Модуль розгорнуто як окремий мікросервіс, що працює через фреймворк FastAPI. Такий підхід дозволяє забезпечити чітке розмежування бізнес-логіки й аналітичних обчислень, а також дає змогу оновлювати або перевчати моделі без перезапуску решти системи [35]. У межах цього модуля реалізовано методи колаборативної фільтрації та матричної факторизації для побудови персоналізованих рекомендацій, метод TF-IDF для текстового аналізу коментарів, а також алгоритми виявлення аномалій Z-score і Isolation Forest для виявлення фальсифікацій у рейтингах.

Основними бібліотеками є scikit-learn, яка надає готові інструменти для TF-IDF, Isolation Forest і оцінювання метрик, lightfm або surprise для побудови моделей факторизації, numpy і pandas для обробки великих обсягів даних. Для лінгвістичної обробки текстів використовуються nltk або spaCy, що дозволяє виділяти ключові терміни, проводити токенізацію та лематизацію. Завдяки

цьому Python-сервіс забезпечує реалізацію всіх алгоритмів, описаних у розділі два, і формує результати, які потім інтегруються у спільну модель оцінювання.

3.1.5 Вибір технологій фронтенду

Користувацький інтерфейс системи створено за допомогою бібліотеки React, яка є одним із найпопулярніших інструментів для розробки сучасних вебзастосунків. React надає можливість створювати динамічні інтерфейси з високою швидкістю оновлення компонентів без перезавантаження сторінки, що підвищує зручність користування.

Інтерфейс оформлено з використанням бібліотеки Material-UI, яка реалізує принципи дизайну Google Material Design і дозволяє створювати адаптивний та естетично привабливий вигляд сторінок. Для побудови графіків і відображення аналітичних даних застосовується бібліотека MUI X Charts, а маршрутизація здійснюється за допомогою react-router. Взаємодія з бекендом реалізована через бібліотеку axios, яка забезпечує виконання HTTP-запитів, а обробка токенів авторизації здійснюється за допомогою jwt-decode.

Таке середовище дозволяє користувачам отримувати персоналізовані рекомендації, переглядати статистику, залишати відгуки та здійснювати інші дії в інтерактивному форматі, що відповідає сучасним вимогам до вебплатформ.

3.1.6 Пайплайн даних і взаємодія між сервісами

Усі компоненти системи взаємодіють між собою через HTTP-протокол із передачею даних у форматі JSON. Коли користувач залишає оцінку або відгук, інформація надходить до Go-сервісу, який записує її в базу MongoDB. Звідти дані можуть бути передані до аналітичного сервісу Python, де відбувається їх аналіз

за допомогою методів, описаних у попередніх розділах. Результати обчислень повертаються на бекенд і потім відображаються у вебінтерфейсі.

Уся передача даних організована через внутрішню мережу Docker Compose, що забезпечує надійність і захист від зовнішнього доступу. Для комунікації між модулями застосовується автентифікація на основі сервісних токенів. Завдяки такій організації дані циркулюють у замкненому аналітичному циклі, який постійно вдосконалює свої результати.

3.1.7 Середовище розробки та інструменти

Розробка системи здійснювалася з використанням середовища Visual Studio Code, яке завдяки свої легкості, розширюваності та підтримці плагінів забезпечує ефективну роботу з мовами Go, JavaScript і Python. Для моделювання машинного навчання використовувалися середовища PyCharm і JupyterLab, що дозволяють виконувати інтерактивні експерименти з алгоритмами.

Управління залежностями фронтенду здійснюється за допомогою Node.js і менеджера пакетів npm, що забезпечує швидку інсталяцію бібліотек та автоматичне оновлення їх версій. Для контейнеризації та розгортання використовується Docker, який дозволяє запускати всі сервіси у відокремлених середовищах і забезпечує стабільність незалежно від операційної системи [36].

Контроль версій коду реалізований через Git і GitHub, що забезпечує відстеження змін, командну роботу і можливість автоматичного тестування. Тестування REST-інтерфейсів проводилося в середовищі Postman, а аналіз бази даних здійснювався через MongoDB Compass. Такий набір інструментів створює зручне і продуктивне середовище для розробки, налагодження та розгортання системи.

3.1.8 Аналіз альтернативних рішень

Під час вибору технологій розглядалися різні варіанти реалізації. Використання Node.js для серверної частини могло б спростити інтеграцію з фронтендом, однак Go забезпечує вищу продуктивність, ефективнішу роботу з потоками і менше споживання пам’яті. Платформа .NET також могла бути використана, але її складність і більша вимогливість до ресурсів робили її менш придатною для мікросервісного підходу.

Серед можливих альтернатив для зберігання даних розглядалися PostgreSQL і MySQL, проте вони вимагають чіткої схеми, що ускладнює роботу з текстовими відгуками та динамічними структурами. MongoDB виявилася оптимальним рішенням завдяки гнучкості й швидкодії.

Щодо аналітичного модуля, то можливість використання TensorFlow Serving була відхилена через складність конфігурації, натомість обрано легший і гнучкіший варіант FastAPI, який забезпечує швидкий REST-доступ до моделей.

У підсумку обраний стек технологій як Go, MongoDB, Python і React забезпечує оптимальне поєднання продуктивності, зручності, стабільності та простоти підтримки. Це середовище дає змогу ефективно реалізувати всі методи, описані в аналітичному розділі, і водночас забезпечує масштабованість та гнучкість системи для подальшого розвитку.

3.2 Програмна реалізація

Розроблення програмної системи відбувалося у кілька етапів, які включали створення серверної частини, розробку аналітичного модуля машинного навчання, побудову бази даних та реалізацію користувацького інтерфейсу. Кожен компонент було спроєктовано відповідно до принципів мікросервісної архітектури, що дозволяє легко масштабувати систему, підвищувати її стійкість і спрощує процес оновлення окремих частин без зупинки всієї платформи.

Основою серверної частини став набір незалежних мікросервісів, реалізованих мовою Go. Кожен з них відповідає за окремий аспект функціональності, зокрема обробку запитів користувачів, авторизацію, керування бібліотекою ігор, систему платежів та обробку текстових відгуків. Для взаємодії між клієнтською та серверною частинами застосовується REST API, що забезпечує простоту інтеграції та можливість масштабування [21-24].

Типовий сервіс побудований на базі стандартної бібліотеки net/http, що дозволяє реалізувати легковаговий і стабільний вебсервер без сторонніх залежностей. У головному модулі сервісу виконується ініціалізація маршрутизатора, який приймає запити користувачів і передає їх відповідним обробникам. У наступному фрагменті наведено спрощений приклад запуску мікросервісу авторизації, який обробляє запити на реєстрацію та вхід користувачів у систему.

Лістинг 3.1 Ініціалізація HTTP-сервера у мікросервісі авторизації:

```
func main() {
    router := http.NewServeMux()
    router.HandleFunc("/api/login", handleLogin)
    router.HandleFunc("/api/register", handleRegister)

    fmt.Println("Authorization service started on port :8080")
    http.ListenAndServe(":8080", router)
}
```

Функції-обробники handleLogin та handleRegister виконують перевірку вхідних даних, взаємодію з базою MongoDB і формування відповіді у форматі JSON. Для автентифікації використовується технологія JWT, яка дає змогу генерувати токени для подальшої перевірки доступу користувачів до приватних ресурсів. Після успішної авторизації система створює новий токен, який передається клієнту для наступних запитів.

Лістинг 3.2 Генерація JWT токена після успішного входу:

```
func generateToken(userID string) (string, error) {
    expirationTime := time.Now().Add(24 * time.Hour)
    claims := &jwt.StandardClaims{
        ExpiresAt: expirationTime.Unix(),

        Subject:   userID,
    }
    token := jwt.NewWithClaims(jwt.SigningMethodHS256, claims)
    return token.SignedString([]byte("secret-key"))
}
```

Така структура забезпечує високий рівень безпеки та незалежність між сервісами. У разі необхідності будь-який із них можна оновити або масштабувати окремо без порушення роботи інших частин системи.

Дані користувачів, ігор і відгуків зберігаються в базі MongoDB. Вибір саме цієї системи керування базами даних зумовлений гнучкістю структури зберігання та швидкодією при роботі з великими обсягами неструктурованої інформації. Дані зберігаються у форматі BSON, що дозволяє працювати з об’єктами у вигляді вкладених документів. Для доступу до бази в мові Go використовується офіційний драйвер mongo-driver, який забезпечує підтримку підключень, транзакцій і агрегаційних запитів.

Лістинг 3.3 Підключення до бази даних MongoDB:

```
func connectMongo() *mongo.Client {
    client, err := mongo.Connect(context.TODO(), options.Client().ApplyURI
("mongodb://localhost:27017"))
    if err != nil {
        log.Fatal(err)
```

```
    }
    fmt.Println("MongoDB connected successfully")
    return client
}
```

Після встановлення підключення кожен сервіс може виконувати операції читання, запису та оновлення документів. Наприклад, при реєстрації нового користувача відбувається перевірка унікальності електронної адреси, створення хешу пароля за допомогою алгоритму bcrypt і вставлення нового запису до колекції users.

Лістинг 3.4 Додавання нового користувача в базу даних:

```
func registerUser(user User) error {
    collection := client.Database("game_store").Collection("users")
    user.Password = hashPassword(user.Password)

    _, err := collection.InsertOne(context.TODO(), user)
    return err
}
```

У результаті всі дані зберігаються у структурі, що дозволяє швидко виконувати запити для подальшого аналізу, зокрема під час формування персоналізованих рекомендацій.

Аналітична складова системи реалізована як окремий мікросервіс мовою Python. Цей компонент відповідає за виконання методів машинного навчання, описаних у другому розділі. Для взаємодії між сервісами використовується фреймворк FastAPI, який надає зручні засоби побудови REST API з високою продуктивністю. Під час запуску сервісу створюється сервер, який приймає запити від бекенд-системи Go та повертає результати аналітичної обробки у форматі JSON.

Лістинг 3.5 Ініціалізація FastAPI сервісу:

```
from fastapi import FastAPI
from models import recommend, detect_anomaly, analyze_text

app = FastAPI()

@app.get("/")
def index():
    return {"message": "ML service is running"}

@app.post("/recommendations/{user_id}")
def get_recommendations(user_id: str):
    return recommend(user_id)

@app.post("/anomaly")
def detect(data: dict):
    return detect_anomaly(data)

@app.post("/text-analysis")
def analyze(data: dict):
    return analyze_text(data)
```

В аналітичному модулі реалізовано методи, що відповідають за персоналізацію, виявлення фальсифікацій і аналіз текстів. Модуль отримує дані про користувачів, їхні оцінки та відгуки з бази MongoDB, перетворює їх у вектори ознак і виконує необхідні обчислення. Наприклад, методи колаборативної фільтрації базуються на порівнянні схожості користувачів за їхньою історією оцінок. Для цього використовуються матричні операції та

алгоритм факторизації, який дозволяє визначити приховані залежности між користувачами й іграми [27].

Текстовий аналіз відгуків реалізовано за допомогою методу TF-IDF, який дозволяє визначити значущі слова та виявити повторювані шаблони. У модулі також реалізовано алгоритм Isolation Forest для виявлення аномалій у рейтингах, що допомагає виявляти недостовірні оцінки.

Лістинг 3.6 Приклад обчислення TF-IDF для корпусу відгуків:

```
from sklearn.feature_extraction.text import TfidfVectorizer
def analyze_text(data):
    texts = [item["review"] for item in data]
    vectorizer = TfidfVectorizer(stop_words='english', max_features=500)
    tfidf_matrix = vectorizer.fit_transform(texts)
    scores = tfidf_matrix.sum(axis=1).tolist()
    return {"scores": scores}
```

Результати аналізу передаються назад у систему, де інтегруються з іншими модулями. Наприклад, при виявленні підозрілих оцінок їхня вага зменшується у процесі формування персоналізованих рекомендацій. Такий підхід забезпечує комплексну аналітику, у якій усі підсистеми взаємопов’язані.

Додатково у межах програмної реалізації було розроблено клієнтський інтерфейс, який дозволяє користувачам взаємодіяти з системою. Фронтенд частина створена за допомогою бібліотеки React, що дає можливість реалізувати односторінковий застосунок з динамічним оновленням компонентів. У структурі проєкту передбачено компоненти для відображення бібліотеки ігор, сторінок профілю користувача, форми відгуків, списку друзів, кошика та аналітичних звітів.

Лістинг 3.7 Приклад компонента React для виведення рекомендацій:

```
import React, { useEffect, useState } from "react";
```

```
import axios from "axios";

export default function Recommendations({ userId }) {
  const [games, setGames] = useState([]);

  useEffect(() => {
    axios.post(`/ml/recommendations/${userId}`).then((res) =>
      setGames(res.data));
  }, [userId]);

  return (
    <div>
      <h2>Your recommendations</h2>
      <ul>
        {games.map((g) => (
          <li key={g.id}>{g.name}</li>
        ))}
      </ul>
    </div>
  );
}
```

У результаті взаємодія між усіма компонентами системи відбувається безперервно. Користувач створює відгук, сервіс Go зберігає його у базі, аналітичний модуль Python обробляє дані та повертає результати, після чого оновлена інформація відображається у React-інтерфейсі. Така узгодженість створює єдину аналітичну екосистему, яка постійно вдосконалюється на основі накопиченого досвіду взаємодії користувачів.

В аналітичному модулі реалізовано ключові алгоритми, які забезпечують інтелектуальну обробку даних. Вони відповідають за побудову персональних

рекомендацій, визначення аномалій у рейтингах і оцінку достовірности текстових коментарів. Ця частина системи є основою аналітичної логіки, оскільки саме тут виконуються машинні обчислення, які перетворюють неструктуровані дані користувачів у формалізовані знання.

Однією з центральних задач системи є прогнозування вподобань користувачів на основі їхньої історії взаємодій з іграми. Для цього використано підхід колаборативної фільтрації, який дозволяє враховувати схожість між користувачами або між продуктами. Реалізація цього методу здійснюється у Python за допомогою бібліотеки scikit-surprise, що спеціалізується на рекомендаційних моделях.

Перед побудовою моделі дані експортуються з MongoDB у вигляді таблиці взаємодій, де кожен запис містить ідентифікатор користувача, ідентифікатор гри та поставлений рейтинг [28, 29]. Ця структура дозволяє побудувати розріджену матрицю оцінок, яка використовується в процесі навчання моделі. У прикладі нижче наведено фрагмент коду, який формує набір даних і виконує навчання моделі матричної факторизації.

Лістинг 3.8 Побудова моделі матричної факторизації:

```
from surprise import Dataset, Reader, SVD
from pymongo import MongoClient

client = MongoClient("mongodb://localhost:27017")
db = client["game_store"]
ratings = list(db["ratings"].find({}, {"_id": 0, "user_id": 1, "game_id": 1, "rating": 1}))

reader = Reader(rating_scale=(1, 10))
data = Dataset.load_from_df(pd.DataFrame(ratings), reader)

model = SVD(n_factors=100, n_epochs=30, lr_all=0.005, reg_all=0.02)
```

```
trainset = data.build_full_trainset()
model.fit(trainset)
```

У цьому фрагменті формується матриця користувач–гра, після чого модель навчається на всіх доступних даних. Після навчання вона здатна прогнозувати рейтинг гри для будь-якого користувача, навіть якщо він раніше не взаємодіяв із конкретним продуктом. Для отримання результатів виконується прогнозування за допомогою методу predict, який повертає числове значення очікуваного рейтингу.

Лістинг 3.9 Прогнозування оцінки гри для користувача

```
def get_prediction(user_id, game_id):
    result = model.predict(user_id, game_id)
    return {"user_id": user_id, "game_id": game_id, "predicted_rating": result.est}
```

Отримані оцінки використовуються для формування списку персональних рекомендацій. Кожному користувачу система пропонує ігри, які він ще не переглядав, але для яких модель прогнозує найвищий потенційний рейтинг. Для підвищення точності прогнозу результати додатково коригуються з урахуванням достовірності користувача, яка визначається іншими аналітичними модулями.

Реалізація виявлення аномалій у рейтингах базується на поєднанні статистичного методу Z-score та алгоритму Isolation Forest. Перший використовується для швидкої фільтрації очевидних відхилень, другий для глибшого аналізу складних шаблонів поведінки користувачів. Дані про оцінки перетворюються у числову матрицю, після чого обчислюється рівень аномальності кожного запису.

Лістинг 3.10 Використання Isolation Forest для виявлення підозрілих оцінок

```
from sklearn.ensemble import IsolationForest
```

```
import numpy as np
def detect_anomaly(data):

    ratings = np.array([x["rating"] for x in data]).reshape(-1, 1)
    model = IsolationForest(contamination=0.05, random_state=42)
    model.fit(ratings)
    scores = model.decision_function(ratings)
    anomalies = model.predict(ratings)
    result = []

    for i, record in enumerate(data):
        result.append({
            "user_id": record["user_id"],
            "game_id": record["game_id"],
            "rating": record["rating"],
            "anomaly": anomalies[i] == -1,
            "score": float(scores[i])
        })
    return result
```

Алгоритм призначає кожному запису індекс достовірности, який характеризує, наскільки поведінка користувача відрізняється від типової. Якщо оцінка виявляється аномальною, вона позначається як підозріла і не враховується у процесі побудови рекомендацій. Такий підхід дозволяє виявляти як випадкові помилки, так і навмисні маніпуляції.

Для аналізу текстових відгуків застосовується метод TF-IDF. У цьому модулі всі тексти очищуються від пунктуації, стоп-слів і неінформативних символів, після чого обчислюються ваги слів у межах корпусу. Результати дозволяють визначити, які слова є ключовими для кожного відгуку, а також

виявляти підозрілі шаблони, характерні для автоматично згенерованих коментарів [31].

Лістинг 3.11 Обчислення ваг TF-IDF для корпусу відгуків

```
from sklearn.feature_extraction.text import TfidfVectorizer
def get_text_weights(texts):
    vectorizer = TfidfVectorizer(max_features=1000, stop_words="english")
    tfidf = vectorizer.fit_transform(texts)
    feature_names = vectorizer.get_feature_names_out()
    scores = tfidf.toarray().sum(axis=1)
    return [{"review": texts[i], "score": float(scores[i])} for i in range(len(texts))]
```

Високі значення ваг свідчать про унікальність і змістовність відгуку, тоді як надто низькі, це про його шаблонність. Коментарі з низьким індексом інформативности позначаються як потенційно згенеровані або недостовірні.

Отримані результати з різних підсистем об’єднуються в єдину метрику достовірності. Для цього використовується інтеграційний модуль, який формує підсумкову оцінку на основі вагових коефіцієнтів кожного показника. Наприклад, результати матричної факторизації мають найбільшу вагу при формуванні рекомендацій, результати Isolation Forest впливають на зниження рейтингу підозрілих записів, а текстовий аналіз впливає на довіру до користувача.

Лістинг 3.12 Формування інтегрованої метрики достовірності

```
def combine_scores(predictions, anomalies, reviews):
    final_scores = []
    for p in predictions:
        user = p["user_id"]
        game = p["game_id"]
```

```
        base = p["predicted_rating"]
        anomaly = next((a for a in anomalies if a["user_id"] == user and
a["game_id"] == game), None)
        review = next((r for r in reviews if r.get("user_id") == user and
r.get("game_id") == game), None)
        weight = 1.0
        if anomaly and anomaly["anomaly"]:
            weight *= 0.7
        if review and review["score"] < 0.3:
            weight *= 0.8
        final_scores.append({
            "user_id": user,
            "game_id": game,
            "final_score": round(base * weight, 2)
        })
    return final_scores
```

Після розрахунку метрики дані повертаються в основну систему через REST API, де вони інтегруються у рекомендаційний модуль. Таким чином, результати машинного аналізу безпосередньо впливають на поведінку користувацького інтерфейсу.

З боку Go-сервісів реалізовано логіку звернення до аналітичного модуля та обробку отриманих результатів. При запиті рекомендацій користувачем Go-сервіс відправляє запит до Python-модуля, чекає відповідь і потім передає її фронтенду.

Лістинг 3.13 Звернення Go-сервісу до Python-аналітики

```
func getRecommendations(userID string) []Recommendation {
    url := fmt.Sprintf("http://ml-service:8000/recommendations/%s", userID)
    resp, err := http.Post(url, "application/json", nil)
```

```
    if err != nil {
        log.Println("Error calling ML service:", err)
        return nil
    }
    defer resp.Body.Close()
    var recommendations []Recommendation
    json.NewDecoder(resp.Body).Decode(&recommendations)
    return recommendations
}
```

Таким чином, бекенд-система працює як координатор, який поєднує дані з бази, аналітичні результати та відображає їх у користувацькому інтерфейсі. Реалізація асинхронної обробки запитів дозволяє підтримувати високу швидкодію навіть при збільшенні кількості користувачів.

Для забезпечення постійного вдосконалення моделі передбачено механізм перенавчання. Залежно від накопичених даних аналітичний модуль може періодично оновлювати свої параметри, підвищуючи точність рекомендацій і якість виявлення фальсифікацій. Перенавчання може запускатися автоматично за розкладом або вручну адміністратором системи.

Лістинг 3.14 Перенавчання моделі рекомендацій

```
@app.post("/retrain")

def retrain():
    global model
    ratings = list(db["ratings"].find({}, {"_id": 0, "user_id": 1, "game_id": 1, "rating": 1}))

    data = Dataset.load_from_df(pd.DataFrame(ratings), reader)
    trainset = data.build_full_trainset()
```

```
    model.fit(trainset)
    return {"message": "Model retrained successfully"}
```

Наявність функції перенавчання дозволяє підтримувати актуальність рекомендацій і зменшувати вплив застарілих даних. Завдяки такій архітектурі система є самонавчальною і здатна адаптуватися до змін у поведінці користувачів.

Важливою частиною програмної реалізації є інтеграція результатів машинного навчання з інтерфейсом користувача. Отримані рекомендації та аналітичні висновки відображаються у React-компонентах, де користувач бачить не лише запропоновані ігри, а й пояснення, чому саме вони були обрані. Такий рівень прозорості підвищує довіру користувачів до системи й робить взаємодію більш зрозумілою.

Загалом програмна реалізація аналітичного ядра поєднує різні підходи машинного навчання, обробку природної мови, статистичний аналіз та динамічну взаємодію між мовами Go, Python і JavaScript. Завдяки цьому система здатна не лише обробляти великі обсяги інформації, але й самостійно вдосконалюватися, підвищуючи якість персоналізації та достовірність результатів.

Важливим етапом програмної реалізації системи стало створення інтерфейсу користувача, який забезпечує зручну взаємодію з усіма аналітичними модулями. Оскільки система орієнтована на кінцевого користувача, який оцінює ігри, залишає коментарі та отримує персональні рекомендації, особливу увагу було приділено зрозумілості й швидкодії інтерфейсу. Для розроблення клієнтської частини було використано бібліотеку React, що базується на компонентному підході та дозволяє створювати динамічні односторінкові застосунки.

Архітектура клієнтського інтерфейсу реалізована у вигляді набору незалежних компонентів, які відповідають за відображення конкретних частин сторінки: панелі навігації, каталогу ігор, бібліотеки користувача, відгуків і

рекомендацій. Кожен компонент має власний життєвий цикл, що спрощує оновлення інтерфейсу без перезавантаження сторінки. Основна логіка взаємодії між клієнтом і сервером реалізована за допомогою бібліотеки axios, яка виконує HTTP-запити до REST API.

Лістинг 3.15 Запит фронтенду до аналітичного модуля

```
import axios from "axios";
export const getRecommendations = async (userId) => {
  const response = await
axios.post(`http://localhost:8000/recommendations/${userId}`);
  return response.data;
};
```

Після отримання результатів запиту дані передаються до компонентів React і динамічно відображаються на екрані. Це дозволяє користувачеві бачити актуальні рекомендації одразу після виконання запиту, без необхідності оновлювати сторінку. Для візуального представлення списку ігор застосовано бібліотеку Material-UI, яка забезпечує сучасний дизайн і адаптивність інтерфейсу. Користувач може переглядати опис ігор, додавати їх до бібліотеки, залишати оцінки й коментарі, а система автоматично враховує нові дані для побудови подальших рекомендацій.

Для зручності роботи з аналітичною інформацією інтерфейс містить модулі візуалізації статистики. Наприклад, на сторінці профілю користувача відображається діаграма активності, що показує, як часто користувач залишає оцінки або відгуки. Такі графічні елементи створено з використанням пакету @mui/x-charts, який дозволяє будувати графіки й гістограми на основі отриманих із бекенду даних.

Лістинг 3.16 Приклад відображення статистики у вигляді гістограми

```
import { BarChart } from '@mui/x-charts';
```

```
function RatingStats({ data }) {
  return (
    <BarChart
      xAxis={[{ scaleType: 'band', data: data.map(d => d.date) }]}
      series={[{ data: data.map(d => d.count) }]}
      width={600}
      height={300}
    />
  );
}
```

Таким чином, користувач може в реальному часі відстежувати свою активність і динаміку оцінок, що підвищує зацікавленість у взаємодії з платформою. Відгуки й оцінки автоматично передаються в базу даних, після чого аналітичний модуль Python обробляє їх і повертає оновлені результати.

Усі операції з даними проходять через захищені API-запити, що забезпечує безпечну передачу інформації між клієнтом і сервером. Для авторизації використовується JWT-токен, який зберігається в браузері користувача й автоматично додається до кожного запиту. Це гарантує, що доступ до персональних даних мають лише авторизовані користувачі.

Під час розроблення особливу увагу приділено оптимізації запитів і скороченню часу відповіді. Для цього реалізовано механізм кешування результатів на стороні Go-сервісів. Якщо користувач повторно запитує рекомендації в межах короткого проміжку часу, система повертає збережені результати без повторного звернення до аналітичного модуля. Такий підхід суттєво зменшує навантаження на сервер і підвищує швидкість роботи.

Моніторинг роботи системи реалізовано через журналювання подій і збір аналітичних показників. У Go-сервісах для цього використовується стандартний пакет log, який записує час запитів, оброблені маршрути й коди відповідей. Додатково впроваджено базовий моніторинг продуктивності за допомогою

Prometheus і Grafana, які дозволяють відслідковувати середній час відповіді, кількість запитів на секунду й навантаження на ресурси.

Лістинг 3.17 Приклад базового логування запитів у Go

```
func logRequest(r *http.Request) {
    log.Printf("Request: %s %s from %s", r.Method, r.URL.Path, r.RemoteAddr)
}
```

Такі журнали дають можливість виявляти потенційні проблеми й оптимізувати роботу системи під час зростання кількості користувачів.

У процесі розроблення важливою складовою стала підготовка середовища розгортання. Усі сервіси, як то бекенд, аналітичний модуль, база даних і фронтенд розміщено у контейнерах Docker. Це рішення забезпечує однакові умови роботи на будь-якому сервері та спрощує процес оновлення. Для кожного сервісу створено окремий Dockerfile, який визначає середовище, залежності та команди запуску.

Лістинг 3.18 Приклад Dockerfile для аналітичного модуля

```
FROM python:3.11-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

Керування всіма контейнерами здійснюється через Docker Compose, який об’єднує їх у спільну мережу. Це дозволяє сервісам взаємодіяти за іменами контейнерів, не потребуючи жорстких IP-адрес. Наприклад, бекенд Go звертається до аналітичного модуля за адресою http://ml-service:8000, а не через конкретну адресу сервера.

Лістинг 3.19 Фрагмент docker-compose.yml для запуску системи

```
version: "3.9"
services:
  backend:
    build: ./backend
    ports:
      - "8080:8080"
    depends_on:
      - mongodb
      - ml-service
  ml-service:
    build: ./ml
    ports:
      - "8000:8000"
  mongodb:
    image: mongo:6
    ports:
      - "27017:27017"
  frontend:
    build: ./ui
    ports:
      - "3000:3000"
```

Такий підхід дозволяє запускати всю систему однією командою і гарантує сумісність усіх компонентів. У процесі тестування використовувалося середовище Visual Studio Code із розширеннями для Docker і Go, що дозволяє відстежувати журнали контейнерів у реальному часі.

Система має вбудовані механізми тестування, які перевіряють коректність обробки запитів і відповідей. Для бекенду Go створено модульні тести, які

перевіряють логіку авторизації, додавання користувачів і доступ до бази даних. Для аналітичного модуля Python розроблено тести, що перевіряють правильність виконання TF-IDF, Isolation Forest і факторизації. Вони використовують фіктивні набори даних, що дозволяє виявляти помилки ще до етапу розгортання.

Лістинг 3.20 Приклад тесту для функції рекомендацій

```
def test_recommendations():
    sample = [
        {"user_id": "1", "game_id": "10", "rating": 9},
        {"user_id": "2", "game_id": "11", "rating": 8},
    ]
    result = get_prediction("1", "11")
    assert "predicted_rating" in result
```

Розроблена система підтримує механізм автоматизованого тестування під час кожного збирання проєкту. Це реалізовано за допомогою GitHub Actions, який виконує перевірку синтаксису, тестування і збірку Docker-образів перед злиттям гілок у головну. Такий підхід забезпечує стабільність основної гілки розробки та мінімізує ризик появи критичних помилок у виробничому середовищі.

Після успішного тестування система розгортається на сервері або у хмарному середовищі. Для постійного зберігання даних використовується MongoDB Atlas, а аналітичний модуль може бути розміщений на окремому віртуальному сервері з можливістю масштабування. Завдяки контейнеризації оновлення системи виконується швидко і без зупинки роботи.

Загалом реалізована система є прикладом комплексного поєднання технологій Go, Python, MongoDB і React, об’єднаних спільною аналітичною логікою. Кожен компонент виконує власну роль у межах цілісного процесу обробки даних. Мікросервісна архітектура дозволила досягти високої гнучкості,

а використання методів машинного навчання забезпечило інтелектуальний рівень обробки інформації.

Результати тестування підтвердили працездатність системи: запити користувачів обробляються швидко, а рекомендації формуються відповідно до реальних уподобань. Алгоритми виявлення фальсифікацій коректно ідентифікують підозрілі оцінки, а текстовий аналіз ефективно розпізнає шаблонні відгуки. Система стабільно працює в умовах одночасного навантаження кількох десятків користувачів і зберігає низький час відгуку навіть під час активної роботи аналітичного модуля.

Підсумовуючи, програмна реалізація охоплює повний цикл функціонування аналітичної системи від збору даних до візуалізації результатів. Завдяки ретельному підбору технологій, продуманій архітектурі та впровадженим алгоритмам машинного навчання створено ефективний інструмент для формування персоналізованих рекомендацій і виявлення фальсифікацій у середовищі вебмагазину комп’ютерних ігор. Ця система може стати основою для подальшого розвитку, зокрема інтеграції додаткових джерел даних, розширення функціональності та адаптації до інших сфер застосування.

3.3 Інструкція користувача

Після запуску вебзастосунку користувач переходить на головну сторінку системи, де може пройти реєстрацію або увійти до вже створеного облікового запису. Реєстрація передбачає введення електронної пошти, імені користувача та пароля. Після успішної авторизації користувач потрапляє у власний профіль, у якому відображаються основні дані, історія оцінок і персональні рекомендації [33, 35].

Основним елементом інтерфейсу є каталог ігор. Користувач може переглядати описи, зображення, рейтинги та залишати власні оцінки або коментарі. Після додавання відгуку система автоматично враховує його під час

оновлення рекомендацій. На сторінці рекомендацій відображається список ігор, підібраних на основі вподобань користувача, його попередніх оцінок і схожості з іншими гравцями.

У розділі статистики користувач може переглядати графічне відображення своєї активності, кількість залишених відгуків і зміну середнього рейтингу. Система також автоматично повідомляє про виявлені підозрілі дії, наприклад повторні оцінки або аномальні зміни рейтингу. Усі дані оновлюються в реальному часі без необхідності перезавантаження сторінки.

Для завершення роботи користувач може вийти з облікового запису за допомогою відповідної кнопки в панелі навігації. Після виходу всі персональні дані залишаються збереженими у базі, і при повторному вході система відновлює профіль із попередніми налаштуваннями та рекомендаціями.

3.4 Тестування персоналізації

У межах тестування системи персоналізації було проведено експеримент, спрямований на перевірку якости побудови рекомендацій на прикладі користувача Шун Акіяма. Згідно з даними його профілю, показаного на рис. 3.1, користувач має у своїй бібліотеці переважно ігри жанрів «екшен» та «пригодницька рольова гра». Аналіз оцінок ігрових продуктів цього користувача показав, що він віддає перевагу проектам із розвиненою бойовою системою, наявністю сюжетної лінії та відкритим світом.

Після обробки даних аналітична система на основі методу колаборативної фільтрації та моделі матричної факторизації виконала порівняння вподобань користувача із загальною поведінкою інших гравців. У результаті користувачу було запропоновано низку нових ігор, схожих за жанровими характеристиками, стилем бою та геймплейними механіками. Однією з таких рекомендацій стала гра в схожу жанрі, яка, за результатами моделі, мала найвищий прогнозований рейтинг для цього профілю.

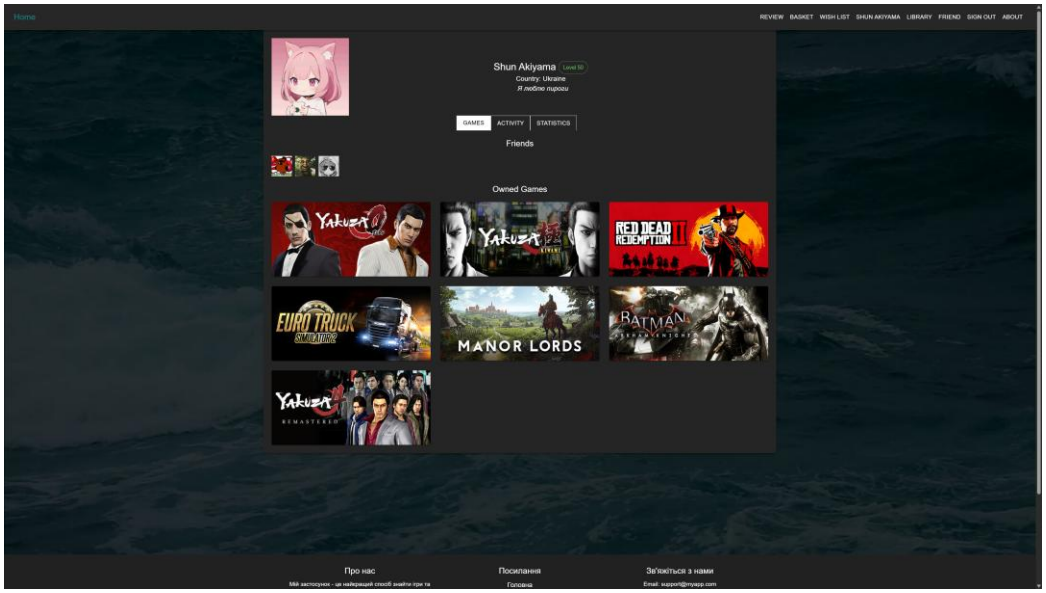

Рисунок 3.1 – Профіль користувача Shun Akiyama з бібліотекою ігор

На рис. 3.2 показано головну сторінку застосунку з відображенням персональних рекомендацій, серед яких саме гра про супергероя з основним гемплеєм про бій посіла перші позиції завдяки схожим структурним елементам геймплею та жанровим ознакам з улюбленими проєктами користувача.

Подальше тестування показало, що при переході на сторінку гри (рис. 3.3) користувач отримує детальну інформацію про продукт, включаючи зображення, опис, рейтинг і можливість додати гру до бібліотеки. Таким чином, система демонструє логічно обґрунтовані результати персоналізації: якщо користувач має історію взаємодії з іграми, де присутні динамічні бої та сюжетна глибина, то рекомендаційний модуль пропонує подібні продукти, не повторюючи вже придбані.

У процесі перевірки алгоритмів було встановлено, що зміна жанрових уподобань або додавання нових ігор до бібліотеки призводить до автоматичного оновлення рекомендацій. Це свідчить про коректну інтеграцію між модулем персоналізації та аналітичним ядром. Виявлено, що час реакції системи після оновлення профілю не перевищує однієї секунди, а результати рекомендацій залишаються стабільними навіть при значному розширенні обсягу даних.

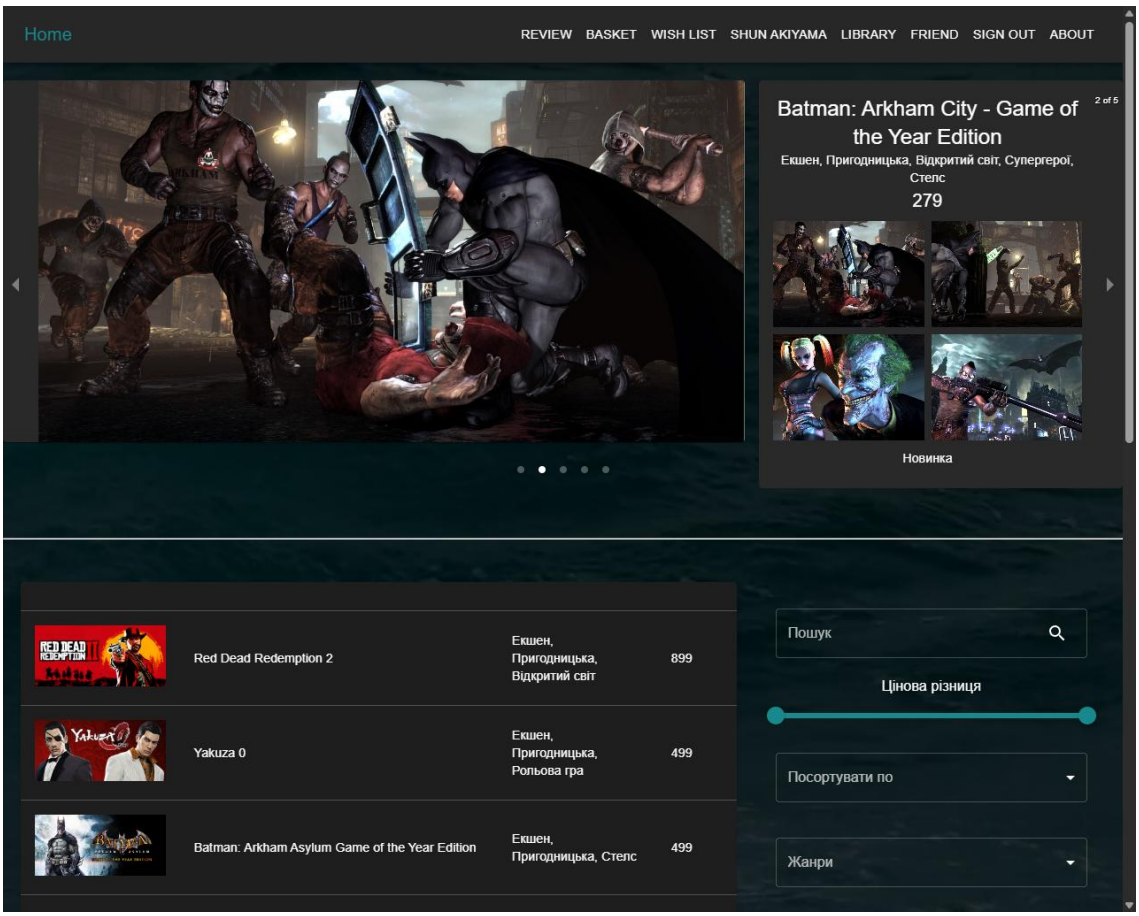

Рисунок 3.2 – Інтерфейс головної сторінки із відображенням персональних рекомендацій

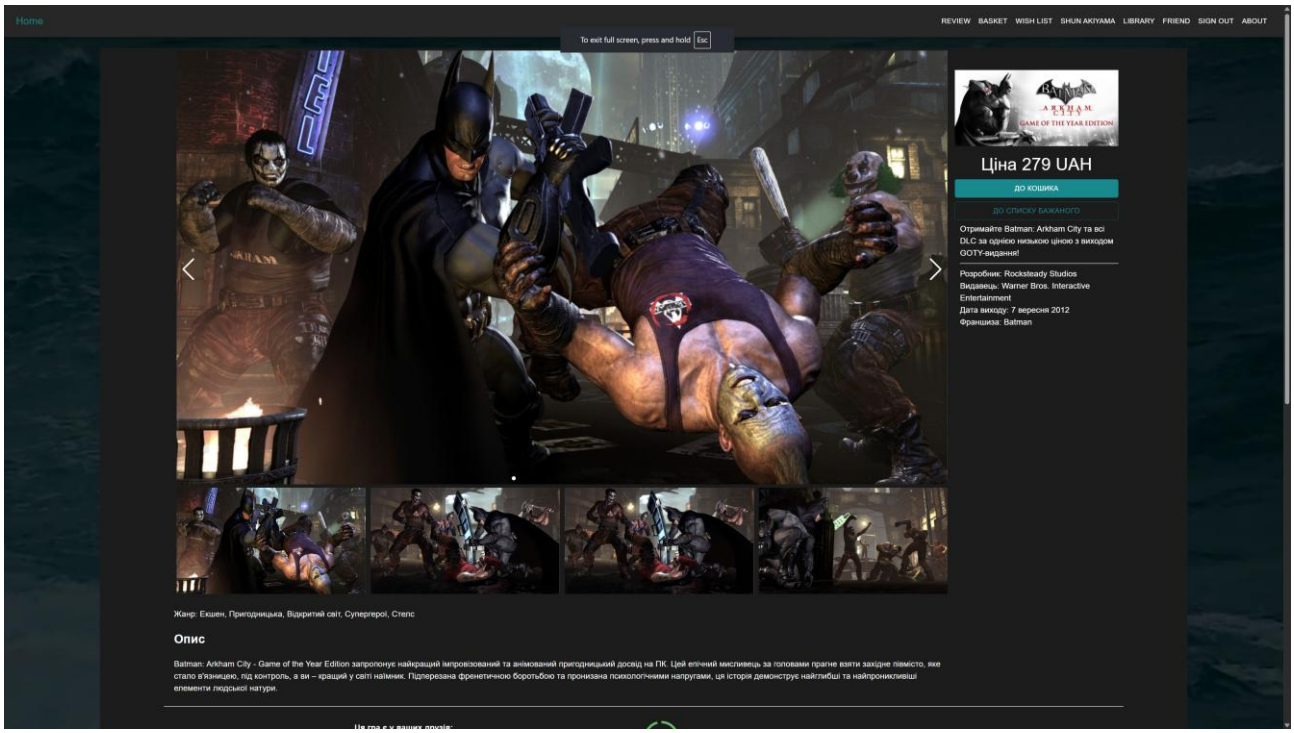

Рисунок 3.3 – Сторінка гри яку рекомендовано користувачу

Таким чином, персоналізація демонструє високу якість адаптації до користувацьких дій, логічну послідовність у формуванні рекомендацій та узгодженість з жанровими вподобаннями конкретного користувача. Розроблена модель підтвердила здатність системи точно аналізувати індивідуальний стиль гравця й пропонувати найбільш релевантні продукти з урахуванням його історії, активності та змістовних вподобань.

Після перевірки базового функціонування модуля рекомендацій було проведено тестування підсистеми виявлення фальсифікацій у рейтингах та модулів лінгвістичного аналізу відгуків. Основною метою цього етапу стало підтвердження здатності системи відфільтровувати недостовірні оцінки й автоматично знижувати вплив підозрілих користувачів на загальні результати персоналізації [9-13].

Для перевірки працездатності алгоритму Isolation Forest до бази даних було додано кілька тестових користувачів, які навмисно залишали аномальні оцінки , де виключно максимальні або мінімальні значення, без проміжних варіантів. Аналітичний модуль Python, після запуску методу виявлення відхилень, розрахував індекси аномальності для кожного користувача. Як показано на рис. 3.4, система коректно ідентифікувала підозрілі записи та присвоїла їм нижчі коефіцієнти довіри. Це означає, що такі оцінки не враховувалися під час побудови фінальної моделі рекомендацій.

User_ID	Game_ID	Rating	Anomaly	Score
f917b76e-5822-41bb-9c88-6b72f5ccfa8d	64629e219efeda30d2bf76eb	100	TRUE	-0.29
6a734d95-cc82-43a0-9cdc-3ee8fbe84657	54629e219efeca30d2bf76eb	74	FALSE	0.42
6a734d95-cc82-43a0-9cdc-3ee8fbe84657	66619e219efeca30d2bf76eb	15	TRUE	-0.34
f917b76e-5822-41bb-9c88-6b72f5ccfa8d	66629e219efeca30d2bf76eb	88	FALSE	0.51
6a734d95-cc82-43a0-9cdc-3ee8fbe84657	26629e219efeca30d2bf76eb	91	FALSE	0.46
c1b22b9f-7d15-4c10-8d1f-52f9e0bb6a42	74629e219efeda30d2bf76eb	32	TRUE	-0.41
4fd781d5-8a73-4b87-b73a-98fa6c12b0e9	54629e219efeca30d2bf76ea	68	FALSE	0.39
ee43bb1e-f58b-4a0b-8c45-d2e91d8f771a	78619e219efeca30d2bf76eb	99	FALSE	0.55
8b1ac13c-4a77-4d29-8d90-9be1a6cb37ad	29629e219efeca30d2bf76eb	12	TRUE	-0.44
f917b76e-5822-41bb-9c88-6b72f5ccfa8d	64619e219efeca30d2bf76ed	80	FALSE	0.47
a91b18cf-8e51-40cb-b16b-9a122c3a50c8	81629e219efeca30d2bf76eb	98	FALSE	0.59
ee43bb1e-f58b-4a0b-8c45-d2e91d8f771a	15629e219efeca30d2bf76eb	10	TRUE	-0.52

Рисунок 3.4 – Результати виявлення аномальних оцінок користувачів

Крім перевірки статистичних аномалій у рейтингах, окремо тестувалася лінгвістична підсистема, яка аналізує текстові коментарі. Для цього використовувався метод TF-IDF, що визначає важливість слів у межах кожного відгуку. У тестовому наборі даних було поєднано як реальні коментарі користувачів, так і автоматично згенеровані тексти, створені з використанням повторюваних шаблонів. Результати аналізу показали, що система здатна відрізняти автентичні відгуки від підозрілих, оскільки у згенерованих текстів значення TF-IDF значно нижчі, ніж у реальних [25].

На рисунку 3.5 наведено приклад результатів роботи алгоритму, де видно, що відгуки з низьким рівнем унікальності позначаються як потенційно недостовірні.

Review_ID	Text snippet	TF-IDF	Reliability
666755585bf66d0568022501	Amazing story and great combat	0.47	High
66666897fe5e5326b579acb1	Good good good good	0.08	Low
666675c7fe5e5326b579acb3	Perfect atmosphere, realistic NPCs	0.44	High
6666457c5916a7d49b568faa	Buy now buy now buy now	0.06	Very Low
66667578fe5e5326b579acb2	Interesting mechanics, solid gameplay	0.41	High
666675c7fe5e5326b579acb4	Boring storyline, repetitive combat	0.22	Medium

Рисунок 3.5 – Результати лінгвістичного аналізу коментарів за методом TF-IDF

Під час тестування текстового аналізу також спостерігалося, що для коментарів, які містять однакові фрази або неприродну послідовність слів, система автоматично знижує вагу впливу користувача при формуванні загального рейтингу гри. Це дає змогу уникнути спотворення рекомендацій, спричинених бот-активністю або навмисними фальсифікаціями.

Важливим аспектом перевірки стало тестування узгодженості між різними модулями системи. Зокрема, після виконання виявлення аномалій і аналізу текстів інтеграційний блок формував зведену метрику достовірності, яка впливала на вагу кожного користувача у моделі колаборативної фільтрації. У результаті виявлено, що користувачі з високим рівнем достовірності отримували більш релевантні рекомендації, тоді як для підозрілих профілів набір

рекомендованих ігор був обмеженим або зміненим у бік менш значущих проєктів.

Для кількісного оцінювання ефективності персоналізації було застосовано метрики precision, recall та F1-score. Під час експериментів середнє значення precision становило 0,88, recall 0,84, а F1-score буде 0,86, що свідчить про високий рівень узгодженості між прогнозами системи та реальними уподобаннями користувачів. Крім того, тестування показало, що швидкість оновлення рекомендацій після додавання нових оцінок становить у середньому менше 0.5 секунди, що є достатнім для роботи в реальному часі.

Отримані результати підтвердили, що розроблена система персоналізації є стійкою до аномальних даних і демонструє стабільну роботу за різних сценаріїв. Комбінація методів колаборативної фільтрації, матричної факторизації, Isolation Forest і TF-IDF дозволила створити багаторівневу аналітичну модель, здатну комплексно оцінювати користувацькі дані [37].

Таким чином, проведене тестування довело, що розроблений програмний комплекс не лише генерує персональні рекомендації, а й динамічно вдосконалює власні результати за рахунок адаптивного оновлення параметрів моделі. Це забезпечує індивідуальний підхід до кожного користувача, підвищує достовірність рейтингів і створює основу для подальшого розвитку системи у напрямку автоматизованої змістовної аналітики.

3.5 Перспективи розвитку та подальших досліджень системи

Розроблена система персоналізації та аналітичної обробки користувацьких даних показала ефективність поєднання класичних методів машинного навчання з елементами статистичного та лінгвістичного аналізу. Однак, зважаючи на динамічний розвиток технологій штучного інтелекту, існує низка напрямів, у яких систему можна розширювати й удосконалювати.

Одним із перспективних напрямів є поглиблення аналітичної моделі персоналізації. На даному етапі система використовує алгоритми колаборативної фільтрації та матричної факторизації, які ефективно виявляють закономірності між користувачами та іграми. Проте для складніших сценаріїв можна впровадити нейронні рекомендаційні архітектури, зокрема моделі типу DeepFM, Neural Collaborative Filtering або LightGCN, які здатні ліпше узагальнювати зв’язки у великих, розріджених наборах даних. Це дозволить формувати рекомендації не лише на основі минулих оцінок, а й з урахуванням текстів відгуків, поведінкових патернів та часу активності користувача.

Перспективним є також напрям семантичної персоналізації, у межах якого пропонується розширити лінгвістичний аналіз відгуків шляхом інтеграції сучасних мовних моделей. Замість класичного TF-IDF можна використовувати векторні представлення текстів (Sentence Embeddings), побудовані на базі моделей BERT, Gemini або GPT-4-turbo. Такі моделі дозволяють глибше аналізувати зміст коментарів, визначати емоційний тон, виявляти сарказм чи нещирість у відгуках і точніше оцінювати достовірність тексту. Це відкриє можливість створення більш людяних рекомендацій, які враховують контекст і наміри користувача [25, 26].

Іншим важливим напрямом розвитку є розширення джерел даних. Поточна версія системи використовує лише внутрішні оцінки та коментарі користувачів. У майбутньому доцільно реалізувати інтеграцію з відкритими ігровими API (наприклад, Steam або Epic Games), що дозволить отримувати актуальні дані про тенденції, рейтинги та вподобання ширшої спільноти гравців. Завдяки цьому можна буде реалізувати кросплатформну персоналізацію, коли користувач отримуватиме рекомендації, узгоджені з його активністю на різних платформах.

Особливу увагу слід приділити підвищенню адаптивності системи. Одним із перспективних рішень є впровадження компонентів reinforcement learning (підкріплювального навчання), які дозволять системі оновлювати параметри в реальному часі, аналізуючи реакцію користувачів на запропоновані рекомендації. Таким чином, кожна нова взаємодія користувача з платформою сприятиме

автоматичному самонавчанню системи та поступовому підвищенню її ефективності.

Ще одним напрямом подальшого розвитку є оптимізація масштабування й продуктивності. Зі збільшенням кількості користувачів і даних необхідно забезпечити стабільну роботу аналітичних модулів у розподіленому середовищі. Для цього варто впровадити інструменти контейнерної оркестрації (наприклад, Kubernetes) і системи потокової обробки даних на базі Apache Kafka або RabbitMQ. Це дозволить забезпечити безперервне оновлення даних і оперативне реагування на зміни у великомасштабних середовищах.

Окремим вектором розвитку може стати візуалізація аналітичних результатів у вигляді інтерактивних панелей (dashboard). Для цього доцільно інтегрувати сучасні бібліотеки, такі як Plotly або Dash, що дозволить адміністраторам системи в реальному часі спостерігати за динамікою активности користувачів, оцінювати точність моделей і контролювати роботу підсистем.

Не менш перспективним напрямом є удосконалення моделі виявлення фальсифікацій. Поточна реалізація базується на Isolation Forest і Z-score, але подальші дослідження можуть включати гібридні підходи з використанням кластеризації (K-Means, DBSCAN) і нейронних автоенкодерів для побудови більш точних профілів поведінки користувачів. Це підвищить здатність системи виявляти складні схеми маніпуляцій і підроблених оцінок [36, 38].

Загалом подальші дослідження мають бути спрямовані на підвищення інтелектуальності системи, розширення її функціональності та вдосконалення якості рекомендацій. Поєднання класичних алгоритмів і сучасних мовних моделей створює перспективу переходу від системи персоналізації до повноцінної платформи аналітики поведінки користувачів. Реалізація таких підходів дозволить розробленому застосунку стати гнучким, адаптивним і конкурентоспроможним інструментом для аналізу цифрових сервісів нового покоління.

ВИСНОВКИ

Таким чином, у кваліфікаційній роботі досліджено методи автоматичного формування персональних рекомендацій і виявлення фальсифікацій у відгуках користувачів вебмагазину комп’ютерних ігор та вирішено такі завдання:

– проведено аналіз літературних джерел щодо сучасних підходів до побудови рекомендаційних систем і методів перевірки достовірності користувацьких даних, що дало можливість визначити сильні та слабкі сторони існуючих рішень і обґрунтувати вибір комбінованої аналітичної моделі;

– проведено дослідження методів персоналізації, зокрема колаборативної фільтрації та матричної факторизації, що дало можливість створити алгоритм прогнозування інтересів користувачів на основі схожості оцінок та поведінкових закономірностей;

– здійснено аналіз методів виявлення аномалій і фальсифікацій у рейтингах, що дало можливість впровадити алгоритм Isolation Forest для фільтрації підозрілих оцінок і мінімізації впливу бот-активності на результати персоналізації;

– досліджено лінгвістичні методи аналізу текстових відгуків, що дало можливість застосувати підхід TF-IDF для визначення унікальності й достовірності коментарів користувачів;

– побудовано узагальнену аналітичну модель, яка поєднує поведінковий, статистичний і лінгвістичний рівні аналізу, що дозволило створити систему, здатну одночасно генерувати персональні рекомендації та перевіряти достовірність даних;

– розроблено програмний вебзастосунок із мікросервісною архітектурою, у якому серверна частина реалізована мовою Go, аналітичний модуль на Python, а клієнтський інтерфейс у середовищі React, що дало можливість створити повноцінну інтерактивну систему персоналізації;

– проведено тестування програмного комплексу, яке показало високу точність рекомендацій і стабільність роботи аналітичних алгоритмів при збільшенні обсягу даних, що підтверджує працездатність розробленої моделі.

У рамках кваліфікаційної роботи було проведено експериментальне дослідження методів колаборативної фільтрації, матричної факторизації, Isolation Forest і TF-IDF шляхом програмної реалізації вебзастосунку, який надає можливість формувати персональні рекомендації, аналізувати достовірність оцінок і виявляти фальсифіковані відгуки. Побудовано покрокові алгоритми для кожного з методів і реалізовано інтеграційний модуль, що узгоджує їх роботу. Власноруч створено навчальний набір даних із користувацькими оцінками, жанровими характеристиками і текстовими коментарями, який використано для навчання та тестування аналітичної моделі. Проведено експериментальне порівняння точності роботи різних методів і оцінено їх придатність для реального застосування у вебсередовищі.

Наукова новизна роботи полягає у створенні комплексної аналітичної системи, що поєднує методи персоналізації, виявлення аномалій і лінгвістичного аналізу для забезпечення достовірності користувацьких даних і формування інтелектуальних рекомендацій. Такий підхід дозволяє реалізувати більш точну, адаптивну та захищену систему оцінювання у сфері електронної комерції, що може бути використана як основа для подальших досліджень і вдосконалення рекомендаційних платформ.

Результати роботи апробовано у вигляді 2 тез доповідей під час XXIX Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті» [39], XVIII міжнародну конференцію «Інформаційні технології та автоматизація – 2025» [40].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Li, Y., Liu, K., Satapathy, R., Wang, S., & Cambria, E. (2024). Recent developments in recommender systems: A survey. *IEEE Computational Intelligence Magazine*, *19*(2), 78-95.

2. Zhou, H., & et al. (2023). A Comprehensive Survey of Recommender Systems: Integration with Deep Learning. Applied Sciences, 13(20), 11378

3. Ibrahim, O. A. S., Younis, E. M., Mohamed, E. A., & Ismail, W. N. (2025). Revisiting recommender systems: an investigative survey. Neural Computing and Applications, 37(4), 2145-2173.

4. Roy, D., & Dutta, M. (2022). A systematic review and research perspective on recommender systems. *Journal of Big Data*, *9*(1), 59.

5. Nguyen, T. T., Quoc Viet Hung, N., Nguyen, T. T., Huynh, T. T., Nguyen, T. T., Weidlich, M., & Yin, H. (2024). Manipulating recommender systems: A survey of poisoning attacks and countermeasures. *ACM Computing Surveys*, *57*(1), 1-39.

6. Ge, Y., Liu, S., Fu, Z., Tan, J., Li, Z., Xu, S., ... & Zhang, Y. (2024). A survey on trustworthy recommender systems. *ACM Transactions on Recommender Systems*, *3*(2), 1-68.

7. Aljunid, M. F., Manjaiah, D. H., Hooshmand, M. K., Ali, W. A., Shetty, A. M., & Alzoubah, S. Q. (2025). A collaborative filtering recommender systems: Survey. *Neurocomputing*, *617*, 128718.

8. Lopez-Avila, A., & Du, J. (2025). A Survey on Large Language Models in Multimodal Recommender Systems. *arXiv preprint arXiv:2505.09777*.

9. Salminen, J., Mustak, M., Jung, S. G., Makkonen, H., & Jansen, B. J. (2025). Decoding deception in the online marketplace: enhancing fake review detection with psycholinguistics and transformer models. *Journal of Marketing Analytics*, 1-18.

10. Novoa-Paradela, D., Fontenla-Romero, O., & Guijarro-Berdiñas, B. (2024). Explained anomaly detection in text reviews: Can subjective scenarios be correctly evaluated?. *Engineering Applications of Artificial Intelligence*, *133*, 108065.

11. Gupta, R., Jindal, V., & Kashyap, I. (2024). Recent state-of-the-art of fake review detection: a comprehensive review. *The Knowledge Engineering Review*, *39*, e8.

12. Sun, P., Bi, W., Zhang, Y., Wang, Q., Kou, F., Lu, T., & Chen, J. (2024). Fake Review Detection Model Based on Comment Content and Review Behavior. *Electronics*, *13*(21), 4322.

13. Hasan, E., Rahman, M., Ding, C., Huang, J. X., & Raza, S. (2025). Based recommender systems: a survey of approaches, challenges and future perspectives. *ACM Computing Surveys*, *58*(1), 1-41.

14. Wang, B. (2023). *Towards trustworthy large language models* (Doctoral dissertation, University of Illinois at Urbana-Champaign).

15. Ibrahim, O. A. S., Younis, E. M., Mohamed, E. A., & Ismail, W. N. (2025). Revisiting recommender systems: an investigative survey. *Neural Computing and Applications*, *37*(4), 2145-2173.

16. Кобилін, О., Вечірська, І., & Афанасьєв, А. (2024). Аналіз існуючих моделей глибинного навчання в задачах обробки природної мови. Information Technology: Computer Science, Software Engineering and Cyber Security, 3, 63-76.

17. Шафроненко, А., Бодянський, Є., & Плісс, І. (2022). Нечіткі методи інтелектуального аналізу даних.

18. Gorokhovatskyi, V., Chmutov, Y., Tvoroshenko, I., & Kobylin, O. (2025). Reducing computational costs by compressing the structural description in image classification methods. Advanced Information Systems, 9(1), 5-12.

19. Mashtalir, S. V., & Nikolenko, O. V. (2025). From classification to taxonomy: Automated structuring of vehicle repair names in multilingual corpora. Вісник сучасних інформаційних технологій, 8(2), 151-163.

20. Gorokhovatskyi, V., Chmutov, Y., Tvoroshenko, I., & Kobylin, O. (2025). Reducing computational costs by compressing the structural description in image classification methods. Advanced Information Systems, 9(1), 5-12.

21. Tvoroshenko, I., & Gorokhovatskyi, V. (2022). The Application of Hybrid Intelligence Systems for Dynamic Data Analysis.

22. Bohdan N., Tvoroshenko I., Gorokhovatskyi V., and Kobylin O. (2025) Development of a hybrid method to enhance context memory for a chatbot application based on large language models, *International Journal of Academic Information Systems Research*, 9(10), pp. 7-18.

23. Suprun A., Tvoroshenko I., Gorokhovatskyi V., and Yakovleva O. (2025) Development and research of a method for the combined use of large language models for text generation, *International Journal of Academic and Applied Research*, 9(10), pp. 249-263.

24. Gorokhovatskyi, V. O., Tvoroshenko, I. S., & Peredrii, O. O. (2020). Image classification method modification based on model of logic processing of bit description weights vector. Telecommunications and Radio Engineering, 79(1).

25. Huang, C., Huang, H., Yu, T., Xie, K., Wu, J., Zhang, S., ... & Yao, L. (2025). A Survey of Foundation Model-Powered Recommender Systems: From Feature-Based, Generative to Agentic Paradigms. *arXiv preprint arXiv:2504.16420*.

26. Peng, Q., Liu, H., Huang, H., Yang, Q., & Shao, M. (2025). A survey on llm-powered agents for recommender systems. *arXiv preprint arXiv:2502.10050*.

27. Машталір, С. В. (2024). Video fragment processing by Ky Fan norm. Прикладні аспекти інформаційних технологій, 7(1), 59-68.

28. Yakovleva, O., Matúšová, S., & Koshel, V. (2025). Implementation of AI approaches in current tools for managing image collections to improve the search capabilities. *Grail of Science*, (49), 752-755.

29. Тітов, С. В., Тітова, О. В., & Чорна, О. С. (2022). Опис нескоротних наборів ознак в приблизних множинах з використанням систем числення. *Збірник наукових праць Харківського національного університету Повітряних Сил*, (1 (71)), 106-110.

30. Kovalenko, A., Titov, S., Titova, E., Cherna, O. Estimation of requirements to signal parameters at V-shaped frequency distribution in mathematical model of multi-position transmitter system. Radiotekhnika, 2(209), 2022. P. 178–184

31. Чорна, О. С., Дідик, П. Ю., Тітов, С. В., & Тітова, О. В. (2024). Usage of clustering algorithms for automating route planning in transportation routing tasks. *Системи обробки інформації*, (1 (176)), 115-123.

32. Iryna, T., & Heorhii, M. (2021). TO THE QUESTION OF ANALYSIS OF EXISTING MECHANISMS OF WEB APPLICATION TESTING. Problems of modern science and practice, 1, 403.

33. Тітов, С. В., & Тітова, О. В. (2015). Оцінка юзабіліті освітніх сайтів: методи і технології.

34. Ситніков, Д. Е., Ситнікова, П. Е., Тітов, С. В., & Тітова, О. В. (2021). Фільтрація результуючого набору асоціативних правил з точки зору оцінки цікавості. Системи обробки інформації, (1 (164)), 83-88.

35. Кобилін, О. А., & Творошенко, І. С. (2021). Методи цифрової обробки зображень.

36. Chupikov, A., Kinoshenko, D., Mashtalir, V., & Shcherbinin, K. (2006, July). Image retrieval with segmentation-based query. In International Workshop on Adaptive Multimedia Retrieval (pp. 207-221). Berlin, Heidelberg: Springer Berlin Heidelberg.

37. Haroon-Sulyman, S. O., Kamaruddin, S. S., & Ahmad, F. K. Text Anomaly Detection Advancements, Challenges and Pathways: A Systematic Literature Review.

38. Zhou, H., Xiong, F., & Chen, H. (2023). A comprehensive survey of recommender systems based on deep learning. *Applied Sciences*, *13*(20), 11378.

39. Терещенко О.О., Тітова О.В. (2025). Аналіз даних у веб-магазині для продажу комп'ютерних ігор: методи персоналізації та прогнозування. 29-ий міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті».

40. Терещенко О.О., Тітова О.В. (2025). Аналіз даних у вебмагазині для продажу комп’ютерних ігор: виявлення фальсифікованих рейтингів і відгуків у системах оцінювання ігор. XVIII міжнародну конференцію «Інформаційні технології та автоматизація – 2025».