Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

“Інформаційна технологія аналізу та рекомендації кінофільмів методами машинного навчання”

Виконав: студент 2 курсу, групи 2ІСТ-24м
спеціальності 126 – «Інформаційні системи та технології»
_____ Тарас ТАРАСОВСЬКИЙ
Керівник: д.т.н., доц. каф. САІТ
_____ Сергій ЖУКОВ
«27» _____ 2025 р.
Опонент: доц. каф. КН
_____ Олексій СІЛАГІН
«03» _____ 2025 р.

Допущено до захисту
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
«28» _____ 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій
Рівень вищої освіти – другий (магістерський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 126 Інформаційні системи та технології
Освітньо-професійна програма – Інформаційні технології аналізу даних та зображень

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
д.т.н., проф. Віталій МОКІН
«25» 09 2025 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Тарасовському Тарасу Сергійовичу

1. Тема роботи: "Інформаційна технологія аналізу та рекомендації кінофільмів методами машинного навчання",
керівник роботи: Сергій ЖУКОВ, к.т.н., доц. каф. САІТ,
затверджені наказом ВНТУ від «21» 09 2025 року №313
2. Строк подання студентом роботи «28» 11 2025 року
3. Вихідні дані до роботи:
Набір даних Kaggle Dataset „Movie Lens Small Latest Dataset" https://www.kaggle.com/datasets/shubhammehta21/movie-lens-small-latest-dataset
4. Зміст текстової частини:
   - аналіз сучасного стану рекомендаційних систем для кінофільмів;
   - вибір оптимальних технологій та огляд набору вхідних даних;
   - оброблення даних та розроблення інформаційної технології;
   - економічна частина.
5. Перелік ілюстративного матеріалу:
   - візуалізація частоти жанрів у вигляді хмари слів;
   - гістограма розподілу фільмів за жанрами у досліджуваному наборі даних;
   - теплова матриця користувач-фільм;
   - узагальнена діаграма роботи рекомендаційної системи NCF;
   - component діаграма веб-системи;
   - діаграма послідовності «Отримання рекомендацій під настрій»;
   - інтерфейс веб-системи – персоналізовані рекомендації фільмів за настроєм і рейтингами.

6. Консультанти розділів МКР

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Олександр ЛЕСЬКО, к. е. н., проф. каф. ЕПВМ	05.11.2025	15.11.2025

7. Дата видачі завдання «25» 09 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз сучасного стану рекомендаційних систем у сфері фільмів	15.09.2025	29.09.2025	вик
2	Основні етапи виконання роботи та огляд набору вхідних даних	25.09.2025	05.10.2025	вик
3	Оброблення результатів спостережень та візуалізація даних	05.10.2025	25.10.2025	вик
4	Розроблення інформаційної технології та веб-системи	25.10.2025	05.11.2025	вик
5	Економічна частина	05.11.2025	15.11.2025	вик
6	Оформлення матеріалів до захисту МКР	15.11.2025	25.11.2025	вик

Студент Тарас ТАРАСОВС

Керівник роботи Сергій ЖУКОВ

АНОТАЦІЯ

УДК 004.852:004.94:791.43

Тарасовський Т.С. Інформаційна технологія аналізу та рекомендації кінофільмів методами машинного навчання. Магістерська кваліфікаційна робота зі спеціальності 126 – інформаційні системи та технології, освітньо-професійна програма – інформаційні технології аналізу даних та зображень. Вінниця: ВНТУ, 2025. 115 с.

На укр. мові. Бібліогр.: 24 назв; рис.: 41; табл.: 3.

У магістерській кваліфікаційній роботі розроблено інформаційну технологію аналізу та рекомендації кінофільмів, яка ґрунтується на методах машинного навчання та сучасних підходах до побудови рекомендаційних систем. Проведено аналіз існуючих моделей рекомендацій, після чого побудовано та досліджено нейромережеві моделі на основі Neural Collaborative Filtering (NCF) у середовищі Python. Додатково створено інноваційний інтерфейс, що забезпечує формування контекстно-залежних рекомендацій із урахуванням настрою користувача. Об’єктом дослідження є процес генерації персоналізованих рекомендацій кінофільмів.

Ілюстративна частина складається з 7 рисунків із результатами реалізації веб-системи рекомендацій.

У розділі економічної частини розглянуто питання про доцільність розробки та впровадження інформаційної технології аналізу та рекомендацій кінофільмів методами машинного навчання.

Ключові слова: системи рекомендації кінофільмів, Neural Collaborative Filtering, персоналізовані рекомендації. машинне навчання, нейромережеві моделі.

ABSTRACT

Tarasovskyi T.S. Information Technology for Film Analysis and Recommendation Using Machine Learning Methods. Master's Thesis in Specialty 126 – Information Systems and Technologies, Educational and Professional Program – Information Technologies of Data and Image Analysis. Vinnytsia: VNTU, 2025. 115 pages.

In Ukrainian. Bibliography: 24 sources; figures: 41; tables: 3.

In the master's thesis, an information technology for analyzing and recommending movies was developed, based on machine learning methods and modern approaches to building recommendation systems. An analysis of existing recommendation models was conducted, after which neural network models based on Neural Collaborative Filtering (NCF) were constructed and investigated in the Python environment. Additionally, an innovative interface was created to generate context-dependent recommendations that take the user's mood into account. The object of the study is the process of generating personalized movie recommendations.

The illustrative part consists of 7 figures presenting the results of implementing the web-based recommendation system.

The economic section examines the feasibility of developing and implementing an information technology for analyzing and recommending films using machine learning methods.

Keywords: movie recommendation systems, Neural Collaborative Filtering, personalized recommendations, machine learning, neural network models.

ЗМІСТ

ВСТУП

Актуальність теми. Проблема інформаційного перевантаження в цифровому світі, зокрема в сфері розваг, є однією з ключових для сучасного користувача. Щодня тисячі нових фільмів та серіалів додаються до стрімінгових платформ, що робить вибір релевантного контенту складним та часозатратним. Традиційні методи пошуку за жанром чи рейтингом часто виявляються неефективними, оскільки не враховують індивідуальні вподобання та контекст перегляду. Персоналізовані рекомендаційні системи, засновані на методах машинного навчання, стають критично важливим інструментом для підвищення задоволеності користувачів, збільшення їхньої лояльності та ефективного управління каталогом контенту. Актуальність розробки вдосконалених рекомендаційних алгоритмів для кінофільмів обумовлена потребою в точній адаптації пропозицій до складних і динамічних смаків аудиторії.

Мета і завдання роботи. Метою роботи є підвищення точності передбачення користувацьких уподобань шляхом розроблення інформаційної технології аналізу та рекомендації кінофільмів, яка базується на комплексній обробці історії переглядів, контентних ознак та емоційного контексту. Для досягнення цієї мети було визначено наступні завдання:

– проаналізувати сучасний стан розвитку рекомендаційних систем та існуючі методологічні підходи до їх побудови;

– обґрунтувати вибір методів, технологій та архітектурних рішень для реалізації системи;

– виконати попередню обробку та розвідувальний аналіз вхідного набору даних (MovieLens);

– побудувати, навчити та оцінити якість гібридної нейромережевої моделі (Neural Collaborative Filtering) з інтеграцією жанрових ознак;

– спроєктувати архітектуру інформаційної технології аналізу та рекомендації кінофільмів;

– здійснити програмну реалізацію веб-системи із забезпеченням ергономічного користувацького інтерфейсу.

Об’єктом дослідження магістерської кваліфікаційної роботи є процес формування персоналізованих рекомендацій мультимедійного контенту в інформаційних системах.

Предметом дослідження магістерської кваліфікаційної роботи є методи та моделі машинного навчання, зокрема методи глибокого навчання та колаборативної фільтрації, для підвищення ефективності рекомендаційних систем.

Методи дослідження. У роботі використано комплекс методів: системний аналіз (для дослідження предметної області), методи машинного та глибокого навчання (зокрема архітектура Neural Collaborative Filtering) для побудови моделей рекомендацій; методи інженерії ознак, векторизації та нормалізації – для попередньої обробки даних. Програмна реалізація виконана засобами мови Python з використанням бібліотек PyTorch, Pandas, NumPy, Scikit-learn.

Новизна одержаних результатів. Отримала подальший розвиток інформаційна технологія побудови рекомендаційних систем за рахунок створення гібридної моделі NCF, яка, на відміну від існуючих, інтегрує векторні представлення (embeddings) користувачів і фільмів із контентною інформацією про жанри, що дозволяє підвищити точність передбачення.

Удосконалено метод взаємодії з користувачем шляхом впровадження механізму врахування поточного емоційного стану (настрою) при формуванні видачі, що підвищує рівень персоналізації рекомендацій.

Практичне значення полягає у створенні повнофункціонального прототипу веб-системи, який може слугувати основою для розгортання комерційного сервісу рекомендацій або бути інтегрованим у наявні стрімінгові платформи для покращення користувацького досвіду.

Апробація та публікації результатів магістерської кваліфікаційної роботи. Опубліковано тези на LV Всеукраїнській науково-технічній

конференції підрозділів Вінницького національного технічного університету НТКП ВНТУ (2025-2026) [1].

1 АНАЛІЗ СУЧАСНОГО СТАНУ РЕКОМЕНДАЦІЙНИХ СИСТЕМ ДЛЯ КІНОФІЛЬМІВ

1.1 Аналіз предметної області

У сучасному цифровому світі, коли обсяг інформації кожен день зростає, користувачі щодня стикаються з проблемою інформаційного перевантаження**.** В цих умовах набуває важливість значення рекомендаційних систем - систем, які рекомендують користувачам товари, послуги або контент на основі їхніх попередніх взаємодій (рис 1.1) [2]. Наприклад, рекомендації фільмів на основі історії перегляду користувача, рекомендації музичних плейлистів на основі улюблених виконавців і прослуханих треків, рекомендації товарів на основі попередніх покупок і переглядів.

Історичний контекст та еволюція рекомендаційних систем для кіно пройшли значну еволюцію — від елементарних механізмів до складних інтелектуальних моделей. На початковому етапі (кінець 1990-х — початок 2000-х) домінували прості підходи, засновані на сукупних рейтингах (на кшталт топів IMDb) та фільтрах за жанром, роком випуску або режисером, що по суті було каталогізацією, а не персоналізацією. Проривом став перехід до алгоритмів колаборативної фільтрації, популяризованих змаганням Netflix Prize (2006-2009), яке стимулювало розвиток точних методів прогнозування рейтингів, зокрема матричної факторизації. Сучасний етап (з 2010-х років) характеризується повною інтеграцією глибокого навчання, обробкою мультимодальних даних (текст, зображення, відео) та прагненням до повноцінної контекстної та пояснюваної персоналізації, що реалізовано в провідних стрімінгових платформах [3].

Для систематизації підходів у галузі, рекомендаційні системи можна класифікувати за кількома ключовими ознаками. За основною метою (завданням) розрізняють: прогнозування рейтингу (Rating Prediction) — передбачення конкретної оцінки; ранжування (Top-N Recommendation) —

формування персонального списку найрелевантніших фільмів; та пошук подібних елементів (Item-to-Item Similarity). За типом використовуваних даних системи працюють з явною зворотним зв'язком (прямі оцінки користувача) або з неявною (сліди поведінки, такі як факт перегляду чи тривалість). За способом навчання та роботи виділяють офлайн-рекомендації (періодичне навчання на історичних даних) та онлайн-рекомендації (реальний час з постійним оновленням).

Фільм як об'єкт рекомендації володіє низкою унікальних характеристик, що суттєво ускладнюють завдання порівняно з товарами електронної комерції чи навіть музичними треками. По-перше, це високий емоційний вплив та контекстна залежність: вибір залежить від настрою, соціального контексту (перегляд наодинці, у парі, з друзями), що вимагає від системи врахування короткострокових намірів. По-друге, висока «вартість помилки»: сеанс перегляду займає 1.5-3 години, тому нерелевантна рекомендація призводить до значної втрати часу та довіри. По-третє, сильна залежність від зовнішніх факторів: популярність фільму часто визначається кінопреміями (Оскар, Канни), релізами продовжень та інформаційним хайпом у соціальних мережах, що вимагає від системи оперативної інтеграції цих сигналів [2].

Серед усіх видів дозвілля перегляд фільмів та телепередач залишається найпопулярнішим, займаючи найбільшу частину вільного часу у більшості людей.

Рисунок 1.1 – Екосистема рекомендаційних систем у дії.

Основна мета рекомендаційних систем (рис. 1.1) полягає у підвищенні зручності взаємодії користувача з великими обсягами даних шляхом автоматичного прогнозування його уподобань на основі попереднього досвіду, поведінкових даних або контекстних характеристик. Такі системи зменшують час пошуку контенту, підвищують рівень задоволеності користувача та збільшує ефективність платформи.

Також, рекомендаційні системи відіграють важливу роль у залученні та утриманні постійних клієнтів на платформах електронної комерції. Показник лояльності клієнтів, що виражається у повторних покупках, є ключовим індикатором ефективності роботи онлайн-магазинів. Постійні клієнти не лише потребують значно менших маркетингових витрат, але й демонструють вищу схильність до здійснення нових угод [2].

Ще одна важлива річ — персоналізація. Користувачі хочуть, щоб платформа розуміла їхні вподобання і пропонувала контент, ніби створений

саме для них. Це підвищує їхнє задоволення і робить їх більш лояльними до сервісу [4].

Система є цікавим рішенням у сфері персоналізованих рекомендацій фільмів, що поєднує сучасні технології штучного інтелекту та глибокого навчання. Звичайний пошук за жанром або ключовими словами часто не дає хороших результатів, бо він враховує лише обмежену інформацію. Рекомендаційні системи працюють інакше — вони аналізують історію переглядів користувача, його оцінки та схожі вподобання інших людей.

Глядачі не хочуть витрачати багато часу на пошук фільмів — їм потрібні швидкі і точні рекомендації. Добра система рекомендацій допомагає значно швидше обрати, що подивитись, і робить цей процес зручнішим. Через велику конкуренцію між стрімінговими платформами, вони стараються зробити свої сервіси кращими. Якщо система рекомендацій працює добре, користувачі більше часу залишаються на платформі, а це важливо для її успіху.

Завдяки використанню сучасних методів машинного та глибокого навчання, рекомендаційні системи поступово переходять від статичних алгоритмів до інтелектуальних моделей, здатних самостійно адаптуватися до змін поведінки користувачів, аналізувати не лише числові чи текстові дані, а й візуальні, емоційні або контекстні сигнали [3].

Сучасні платформи медіа-контенту активно використовують рекомендаційні системи як ключовий інструмент утримання аудиторії. Ефективність цих сервісів визначається не лише обсягом та якістю контенту, але й здатністю формувати персоналізовані пропозиції, що стимулюють регулярне використання платформи. Високий рівень конкуренції на ринку змушує провідних провайдерів обмежувати розкриття деталей роботи своїх алгоритмів, проте аналіз функціоналу та інтерфейсних рішень дозволяє виявити ключові принципи їхньої побудови.

Рекомендаційні системи відіграють важливу роль у залученні та утриманні постійних клієнтів на платформах електронної комерції. Серед найвпливовіших медіа-провайдерів із розвиненими системами рекомендацій

варто відзначити Netflix, Disney+, HBO Max, Apple TV+ та YouTube [4]. Кожна з цих платформ розробила унікальний підхід до персоналізації, що поєднує передові технології машинного навчання з глибоким розумінням поведінки користувачів.

Таким чином, рекомендаційні системи є ключовим елементом в інформаційних технологій, що персоналізують цифровий досвід користувачів та створюють основу для розвитку інтелектуальних сервісів.

Об’єктом дослідження є інтелектуальна система рекомендацій кінофільмів, яка призначена для формування персоналізованих пропозицій на основі аналізу поведінки користувачів. До основних аспектів функціонування системи належать збір та обробка даних про перегляди, оцінки й уподобання користувачів, а також застосування алгоритмів машинного навчання для прогнозування потенційно цікавих фільмів.

1.2 Суть технічної проблеми

Основна технічна проблема сучасних рекомендаційних систем полягає у подоланні фундаментальних обмежень традиційних методів, які не здатні адекватно моделювати складні багатовимірні взаємозв’язки між користувачами, фільмами та численними характеристиками контенту. Ця проблема особливо загострюється в умовах високої розрідженості даних, коли більшість користувачів взаємодіє лише з незначною частиною доступного каталогу. Внаслідок цього зростає невизначеність у формуванні персоналізованих рекомендацій та ускладнюється побудова точних моделей переваг.

Класичні підходи колаборативної фільтрації продовжують демонструвати обмеження, зокрема проблему «холодного старту», коли система не має достатньої інформації про нових користувачів або новий контент. Контент-базовані методи, попри свою здатність працювати в умовах низької кількості взаємодій, не враховують багатогранність індивідуальних

смаків, часто зводячи модель до аналізу простих перетинів ознак. Це призводить до поверхневого розуміння користувацьких уподобань та неможливості охопити приховані закономірності.

Технічний виклик також полягає в обробці експоненційно зростаючих обсягів даних, що включають історію переглядів, рейтинги, жанрову приналежність, метадані фільмів, текстові описи та інші джерела інформації. Інтегрування таких різнорідних даних потребує побудови складних моделей, здатних не лише кодувати ці ознаки у компактному векторному просторі, але й зберігати семантичну узгодженість між ними. Особливо проблематичним є представлення категоріальних даних — таких як жанри, акторський склад чи країна виробництва — у вигляді неперервних embeddings, які б відображали реальну подібність між об’єктами.

Сучасні гібридні підходи, включаючи нейроколаборативну фільтрацію (Neural Collaborative Filtering, NCF), пропонують об'єднання можливостей матричної факторизації з глибинними нейронними мережами. Проте створення таких моделей вимагає розробки спеціалізованих архітектур, які здатні ефективно інтегрувати embeddings користувачів, фільмів та додаткових ознак у багаторівневу структуру. Технічна складність полягає у виборі оптимальної кількості шарів, глибини мережі, функцій активації та стратегій регуляризації. Додатковою проблемою є боротьба з перенавчанням, особливо при роботі з розрідженими датасетами, де модель може легко «запам’ятовувати» окремі взаємодії, але не здатна узагальнювати.

Ще один важливий аспект — забезпечення масштабованості системи. Рекомендаційні моделі повинні обробляти мільйони взаємодій у реальному часі та генерувати актуальні рекомендації для користувачів різних категорій. Це потребує оптимізації не лише алгоритмічної частини, але й використання високопродуктивних обчислювальних інфраструктур, механізмів кешування, індексації та паралельної обробки.

Оцінювання якості рекомендацій також становить окремий технічний виклик. Необхідно розробляти метрики, які враховують не тільки точність

прогнозування рейтингів (наприклад, RMSE, MAE), але й персоналізовану релевантність запропонованих фільмів.

У сукупності ці проблеми формують складний технічний простір, де необхідно збалансувати точність, масштабованість, узагальнювальну здатність моделей та швидкість їх роботи. Саме тому розробка сучасної рекомендаційної системи для кінематографічної сфери потребує глибокої технічної експертизи, продуманого архітектурного дизайну та комплексного підходу до інтеграції даних.

1.3 Аналіз відомих програмних продуктів

Сучасний ринок рекомендаційних систем для кіноконтенту значно змінився: від простих content-based або collaborative алгоритмів до комплексних гібридних систем, які здатні працювати з великими обсягами даних, поєднувати різні методи машинного й глибинного навчання та враховувати мультимодальні ознаки (метадані, текст, зображення чи відео). Такий розвиток відображено в оглядах дослідницької літератури, зокрема у великому survey-огляді Contemporary Recommendation Systems on Big Data and Their Applications, де досліджено еволюцію систем рекомендацій, їх класифікацію (content-based, collaborative, knowledge-based, hybrid), а також виклики, які постають у побудові систем на великих даних — розрідженість, масштабованість, потреба в різноманітності рекомендацій [4].

У прикладі комерційних стримінгових платформ, Netflix демонструє, як можуть виглядати передові рекомендаційні екосистеми. За офіційним описом, алгоритм Netflix використовує дані про перегляди, історію взаємодій, рейтинги, а також інформацію про сам контент — жанри, категорії, акторів, рік випуску тощо. До того ж береться до уваги контекст перегляду: час доби, тип пристрою, мова, тривалість сесії, що впливає на те, які фільми чи серіали будуть запропоновані [5].

При цьому рекомендації формуються на кількох рівнях: спочатку вибір «рядків» на головній сторінці (наприклад, «Продовжити перегляд», «Популярне у вашому регіоні», «Рекомендоване для вас»), потім — які саме титули включити в ці ряди, і нарешті — у якому порядку їх відображати.

З іншого боку, наукові дослідження показують, що гібридні підходи — коли комбінуються контент-аналіз із collaborative filtering або з embedding-методами та нейронними моделями — дають змогу суттєво покращити якість рекомендацій. Наприклад, нещодавня праця Movie recommendation based on deep neural networks продемонструвала, як можна застосувати технологію нейроколаборативної фільтрації (NCF) з embedding-просторами та багатошаровими нейронними мережами, щоб більш точно відображати вподобання користувачів і властивості фільмів [6].

Крім того, сучасні тренди йдуть у бік мультимодальних рекомендацій, коли до уваги беруться текстові описи, зображення, відео, аудіо — усе, що може надати додаткову інформацію про контент та підвищити релевантність рекомендацій.

Незважаючи на успіхи, системи рекомендацій стикаються зі спільними фундаментальними проблемами. Однією з найперших є розрідженість даних (data sparsity): в умовах великого каталогу більшість користувачів має мало взаємодій, що ускладнює побудову надійних моделей.

Це породжує проблему «cold start» — для нових користувачів або нового контенту система має недостатньо інформації, щоб робити точні рекомендації. Навіть гібридні підходи з мультимодальними ознаками, хоч і зменшують цей ефект, не позбавляють його повністю.

Ще одна важлива проблема — масштабованість і обчислювальна складність. Сучасні платформи оперують сотнями мільйонів комбінацій користувачів і контенту, постійно обробляють великі потоки поведінкових даних, метаданих, мультимодального контенту. Для забезпечення швидких, real-time рекомендацій потрібні добре спроектовані ML-пайплайни, ефективні алгоритми, оптимізована інфраструктура. У наукових оглядах цей аспект

називають серед ключових викликів для реального застосування систем у великих масштабах [7].

Крім того, сучасні системи — особливо з глибинним навчанням — часто працюють як «чорні скриньки», що породжує проблему інтерпретованості (explainability). Коли модель поєднує десятки (чи сотні) ознак, embeddings, нейронні шари, мультимодальні дані — важко точно пояснити кінцевому користувачеві або оператору, чому була запропонована саме ця рекомендація. Це створює серйозні питання з точки зору прозорості, етики, довіри користувачів. В оглядах також наголошується, що необхідні нові підходи, які забезпечують баланс між потужністю моделі і її пояснюваністю.

Останнім часом стрімко набирає популярності підхід, який включає мультимодальні моделі та generative / large-model підходи. Зокрема, survey Generative-Recommender Systems (Gen-RecSys) показує, як сучасні системи можуть інтегрувати текст, зображення, відео, а також взаємодії користувачів, використовуючи глибинні генеративні або мультимодальні моделі для підвищення якості рекомендацій та навіть створення нових рекомендацій з урахуванням комплексних ознак.

Отже, сучасний стан індустрії рекомендацій для кіно/відео виправдовує ваші початкові зауваження: ринок представлений не просто базовими фільтрами, а серйозними, потужними, data-driven системами, що поєднують колаборативні підходи, content-аналіз, deep learning, мультимодальні ознаки, контекст. Проте жодна з існуючих систем не усуває весь набір проблем — розрідженість, cold-start, масштабованість, інтерпретованість, баланс між релевантністю, різноманітністю та справедливістю рекомендацій. Саме тому сучасні дослідження, а також практичні інженерні рішення, зосереджуються на мультимодальних, гібридних, масштабованих і пояснюваних моделях. Як результат, майбутнє рекомендаційних систем пов’язане з подальшим розвитком інтелектуальних, гнучких, мультимодальних і етичних систем персоналізації.

Для кращого розуміння сучасних практик, нижче наведено рисунок 1.2 ключових особливостей рекомендаційних систем провідних медіа-сервісів та платформ.

Платформа	Принцип роботи (акцент)	Осоолиесты та контекст	Публікацы / Технiологиі
NETFLIX	Гібридна: CF + Content-based + Контекстна персоналізація.	Час доби, тип пристрою, тривалість сесії, перегляд до кінця, міжнародні відмінності.	Патенти на «персоналізовані рядки зображень», система A/B тестування.
YouTube	Глибоке навчання: Two-tower модель, обробка послідовностей.	Взаємодія (перегляд, лайк, «дизлайк», пропуск), граф зв'язків між відео, пошукові запити.	Дослідження «Deep Neural Networks for YouTube Recommendations».
HBOmax Disney+	Гібридна з урахуванням бренду: CF + Сильна content-based складна (франшизи, всесвіти).	Акцент на власному преміум-контенті, франшизах, колекціях. Менше публікацій.	Інтеграція з портфелем материнської компанії (Warner Bros., Disney).
TMDB Letterboxd	Соціально-графова, на основі спільності смаків.	Рекомендації на основі оцінок «друзів», кураторських списків, спільних оцінок зі спільнотою.	Відкриті API, спрямованість на ентузіастів кіно.

Рисунок 1.2 – Порівняльний аналіз рекомендаційних систем

1.4 Опис методів реалізації рекомендаційних систем

Рекомендаційні системи реалізуються на основі різних алгоритмічних підходів, що визначають спосіб аналізу користувацьких даних і формування прогнозів уподобань. За підходом до формування рекомендацій розрізняють кілька основних типів систем [9]:

1. Контентна фільтрація (Content-Based Filtering) [10].

Основна ідея полягає в тому, щоб рекомендувати користувачеві елементи на основі спільних атрибутів або ознак тих, які він уже вподобав. Наприклад, для рекомендації фільмів можуть використовуватися такі атрибути: жанр, режисер, актори, рік випуску та інші. Вони допомагають системі зрозуміти, які елементи можуть бути подібні за контентом. Цей тип рекомендаційних систем особливо ефективний у галузях, де важливий вміст або якісні характеристики елементів (рис. 1.3).

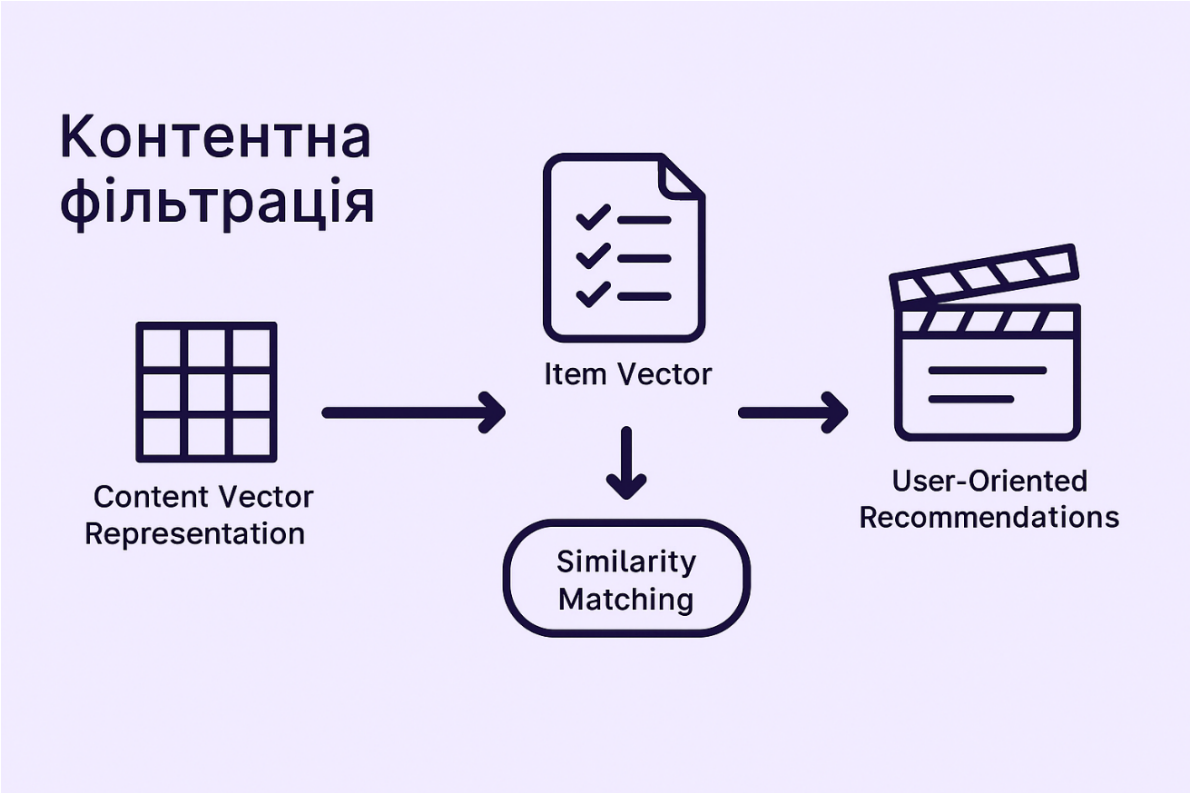

Рисунок 1.3 – Контентна фільтрація

Переваги контентної фільтрації [11]:

- Незалежність: Не залежить від даних інших користувачів, що робить її ефективною для нових користувачів ("холодний старт");
- Прозорість: Користувач може легко зрозуміти, чому йому порекомендували певний товар, оскільки рекомендація ґрунтується на зрозумілих характеристиках;
- Нові об'єкти: Система може рекомендувати нові товари одразу після їх додавання, оскільки їх характеристики вже відомі.

Недоліки:

- Обмеженість: Система може рекомендувати лише товари, схожі на вже вподобані, що призводить до "надмірної спеціалізації" та зменшує різноманітність;
- Залежність від опису: Якість рекомендацій залежить від повноти та точності опису об’єктів.

2. Колаборативна фільтрація (Collaborative Filtering)

Колаборативна фільтрація - базується на аналізі схожості між користувачами або між елементами за історією оцінок. Щоб надати рекомендації на основі колаборативної фільтрації, система використовує матрицю взаємодій, де кожен рядок представляє користувача, а кожен стовпчик — елемент (рис 1.4).

Переваги колаборативної фільтрації [12]:

– Несподівані рекомендації: Може пропонувати користувачам товари, які не схожі на їхні попередні вподобання, але які сподобалися схожим людям;

– Не залежить від опису: Ефективна, навіть якщо про товари немає детального опису;

– Добре працює з великими даними: Показує високу ефективність на великих масивах даних про поведінку користувачів.

Недоліки:

– Проблема "холодного старту": Не може ефективно рекомендувати товари новим користувачам або нові товари, для яких ще немає достатньо даних про взаємодію;

– Розрідженість даних: Ефективність знижується, якщо даних про взаємодію користувачів з об’єктами недостатньо.

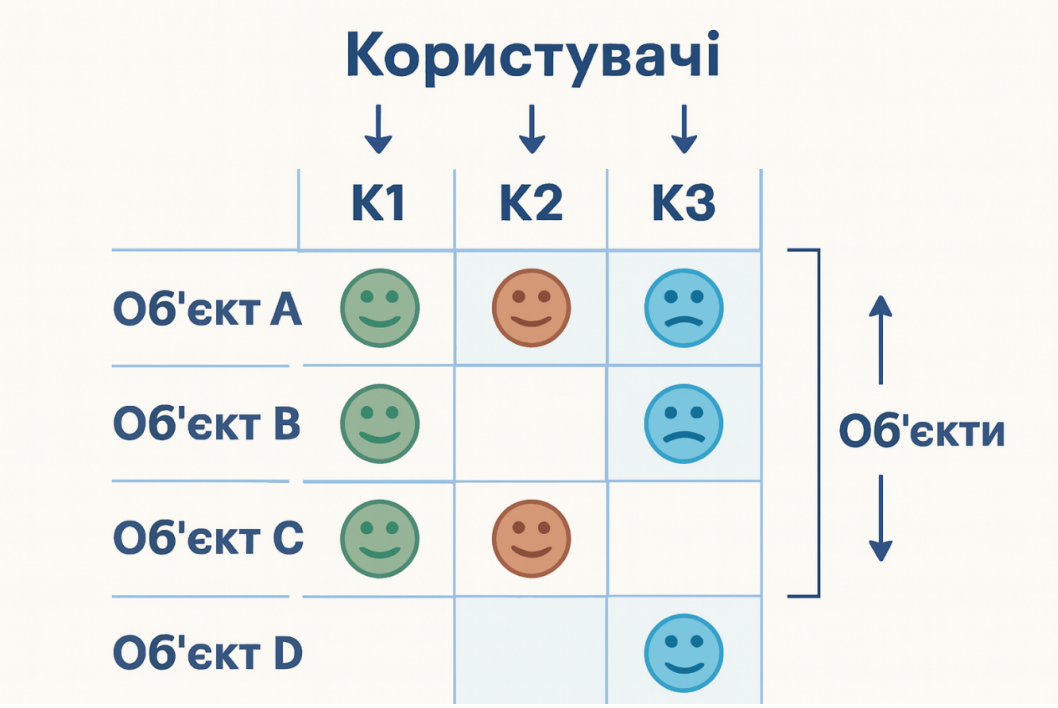

Рисунок 1.4 – Матриця взаємодій

3. Гібридні системи:

Гібридні системи поєднують обидва підходи використовуючи переваги кожної з них для компенсації недоліків іншої, для підвищення точності та релевантності результатів (рис 1.5).

Переваги гібридних систем:

– Вирішення проблеми "холодного старту": Гібридні системи можуть рекомендувати товари новим користувачам або нові товари, для яких ще немає достатньо даних про взаємодію. Якщо немає даних про поведінку користувача, система може спиратися на контентні характеристики [13];

– Подолання проблеми розрідженості даних: Колаборативна фільтрація може бути неефективною, якщо дані про взаємодії користувачів з товарами є неповними. Фільтрація на основі контенту заповнює ці прогалини, використовуючи атрибути товарів;

– Збільшення різноманітності рекомендацій: Об'єднання методів запобігає "надмірній спеціалізації" — ситуації, коли система рекомендує лише дуже схожі товари;

– Вища точність: Поєднання різних підходів допомагає системі краще розуміти вподобання користувачів і зв'язки між товарами

Рекомендаційна система Netflix використовує гібридний підхід, аналізуючи як історію переглядів і оцінок користувачів (колаборативна фільтрація), так і жанри, теги та інші характеристики фільмів (фільтрація на основі контенту).

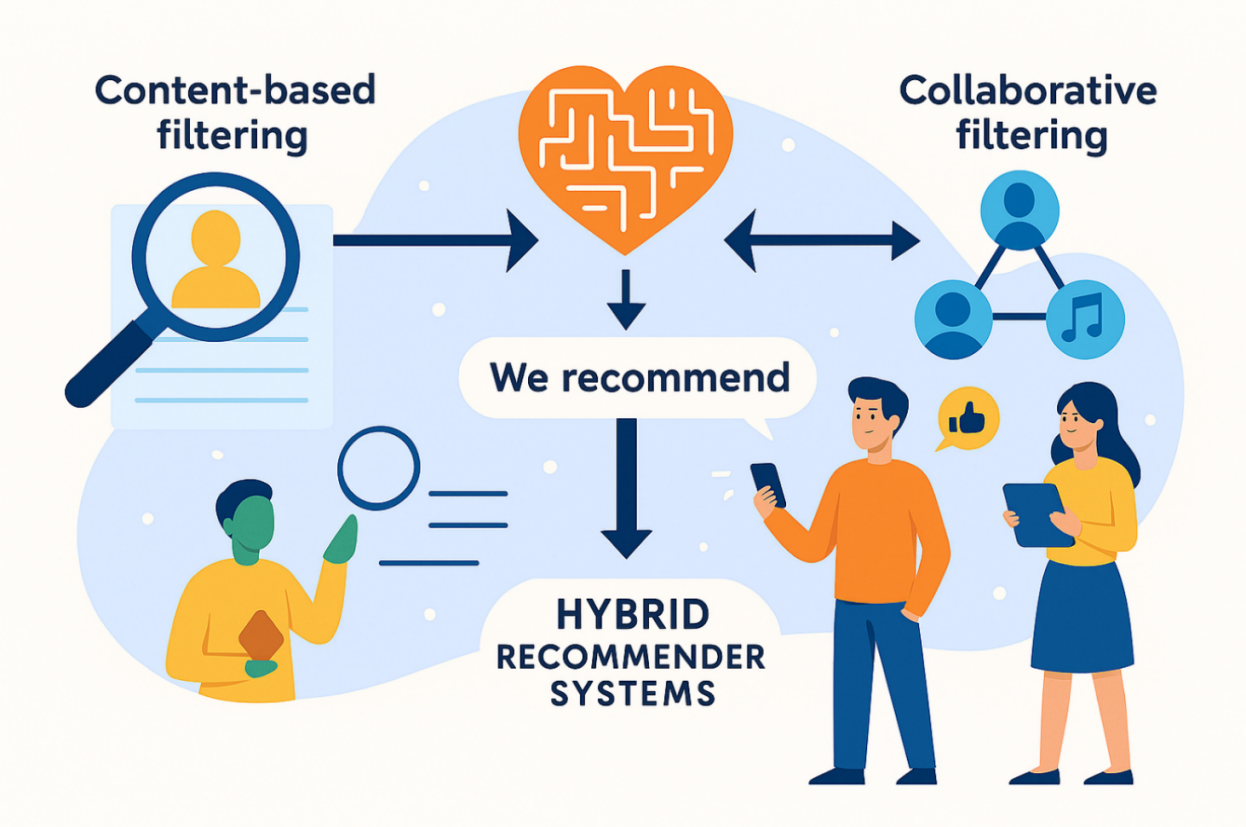

Рисунок 1.5 – Гібридна система

Останні роки спостерігається активне впровадження нейронних мереж та глибокого навчання у рекомендаційні системи. Сучасні нейромережеві підходи демонструють принципово новий рівень ефективності при роботі з гетерогенними даними – від текстових описів і візуального контенту до акустичних характеристик і навіть емоційних сигналів користувачів. Це відкриває шлях до створення мультимодальних рекомендаційних моделей, здатних аналізувати комплексний психоемоційний стан користувача в

реальному часі та адаптувати персональні пропозиції відповідно до контексту взаємодії.

Одним із сучасних прикладів є модель Neural Collaborative Filtering (NCF) - це нейронна мережа, яка забезпечує спільну фільтрацію на основі взаємодії користувача та елемента. Модель NCF розглядає матричну факторізацію з точки зору нелінійності. NCF TensorFlow приймає послідовність пар (ідентифікатор користувача, ідентифікатор елемента) як вхідні дані, а потім подає їх окремо на етап факторизації матриці (де вбудовування множиться) і в мережу багаторівневого персептрона (MLP).

Потім результати матричної факторизації та мережі MLP об’єднуються та подаються в один щільний шар, який передбачає, чи ймовірно, що користувач введення взаємодітиме з елементом введення [13].

Головна перевага NCF полягає в його здатності вивчати складні патерни даних без необхідності ручної інженерії ознак, що призводить до підвищення точності прогнозів порівняно з лінійними моделями. Розвиток основних методів у галузі демонструє чітку еволюційну послідовність, де кожен наступний крок долає обмеження попереднього. Цей шлях почався з колаборативної фільтрації на основі пам'яті (k-NN), яка є простою, але неефективною для великих даних та чутливою до розрідженості. Наступним кроком стала модель на основі матричної факторизації (SVD), яка ефективно працює з латентними ознаками та добре масштабується, але залишається лінійною. NCF є природним продовженням цієї еволюції, поєднуючи ідеї факторизації (embeddings) з потужністю нейронних мереж для моделювання нелінійних взаємозв'язків, що робить його сучасним стандартом для багатьох застосувань. Подальший розвиток NCF спрямований на інтеграцію додаткових ознак, що дозволяє створювати гібридні моделі, здатні споживати не лише ID користувача та фільму, але й додаткові метадані, такі як жанри або теги. Таким чином, NCF займає центральне місце в сучасному арсеналі методів, являючи собою потужний компроміс між теоретичною

обґрунтованістю, практичною ефективністю та можливістю подальшого розширення.

Таким чином, сучасна практика рекомендаційних систем у кіно/відео сфері вже далеко виходить за рамки простих алгоритмів. Поєднання content-based, collaborative, гібридних підходів і deep learning дає змогу створювати більш складні, точні, адаптивні та персоналізовані системи. Водночас, розробники та дослідники продовжують працювати над подоланням фундаментальних викликів — від обмежень даних і cold-start до прозорості, масштабованості та підтримки різноманітності контенту.

1.5 Висновки

За результатами аналізу предметної області визначено, що критичним бар’єром у створенні ефективних рекомендаційних систем є обмеження класичних алгоритмів (розрідженість матриці взаємодій, проблема «холодного старту»). На основі порівняння існуючих методологій обґрунтовано вибір гібридної архітектури з використанням методів глибокого навчання (Neural Collaborative Filtering). Доведено, що саме такий підхід забезпечує необхідну точність та адаптивність системи шляхом комплексної обробки історії взаємодій та контентних ознак фільмів.

2 ВИБІР ОПТИМАЛЬНИХ ТЕХНОЛОГІЙ ТА ОГЛЯД НАБОРУ ВХІДНИХ ДАНИХ

2.1 Вибір оптимальних інформаційних технологій

Для реалізації інтелектуальної системи рекомендацій фільмів було обрано мову програмування Python як основний інструмент розробки.

Архітектура системи реалізована за модульним принципом, що дозволяє чітко розділити функціональність між трьома взаємопов'язаними компонентами: Backend, Frontend та Data.

Вибір зумовлений широкими можливостями Python у сфері штучного інтелекту, машинного навчання та глибокого навчання, а також великою кількістю бібліотек і фреймворків. Переваги для даного проекту полягають у наступному:

– Бібліотеки машинного навчання: Python має потужні інструменти для розробки та навчання моделей, такі як PyTorch, Scikit-learn, NumPy та Pandas. Це дає змогу ефективно реалізовувати нейронну архітектуру Neural Collaborative Filtering (NCF) та обробляти великі обсяги даних користувачів і фільмів.

– Веб-розробка: Фреймворк Flask забезпечує просту, але гнучку розробку RESTful API, через яке здійснюється зв’язок між клієнтським інтерфейсом, моделлю машинного навчання та базою даних.

– Обробка даних: Завдяки бібліотекам Pandas та NumPy Python забезпечує ефективну роботу з великими наборами даних у форматах CSV.

– Обробка зображень: За допомогою бібліотеки PIL (Pillow) можливо здійснювати базову обробку та оптимізацію зображень — зміну розміру, форматування, збереження кешу та створення візуальних попередніх переглядів.

2.1.1 Технологічний стек Python

Технологічний стек Python наведено на рисунку 2.1.

Рисунок 2.1 – Технологічни стек Python

– Flask 2.3.0 — мінімалістичний веб-фреймворк для Python, що використовується для створення RESTful API, обробки маршрутизації запитів та інтеграції машинного навчання з веб-інтерфейсом [14]. Він забезпечує легкий спосіб створення серверної частини додатку з підтримкою JSON-відповідей. Flask надає гнучку архітектуру для створення ендпоінтів, що обробляють запити на отримання рекомендацій, управління користувацькими

профілями та взаємодію з моделлю машинного навчання. Його модульна структура дозволяє легко інтегрувати middleware для обробки помилок, кешування та автентифікації.

- PyTorch 2.0.0 — фундаментальна бібліотека для глибокого навчання, що застосовується для побудови, тренування та оцінки нейромережевих моделей [15]. У цьому проекті вона використана для реалізації архітектури Neural Collaborative Filtering (NCF). PyTorch забезпечує автоматичне диференціювання (autograd), що значно спрощує процес оптимізації параметрів мережі, а також підтримує розподілені обчислення для прискорення навчання на GPU. Бібліотека також надає готові компоненти для роботи з embedding шарами, що критично важливо для представлення користувачів та фільмів у векторному просторі.
- NumPy 1.26.0 — основа наукових обчислень у Python, що надає ефективні інструменти для роботи з багатовимірними масивами та матричними операціями. Використовується для векторизації обчислень і прискорення обробки даних. NumPy оптимізує операції лінійної алгебри, що є ключовим для обчислення схожості між векторами користувачів та фільмів, а також для ефективної імплементації матричної факторизації.
- Pandas 2.0.0 — потужний інструмент для маніпулювання даними, що забезпечує зчитування CSV-файлів, очищення даних від аномалій, фільтрацію та підготовку датасетів для подальшого аналізу. Pandas надає інтуїтивні структури DataFrame для роботи з табличними даними, що дозволяє ефективно об'єднувати інформацію про рейтинги, фільми та користувачів, а також здійснювати статистичний аналіз розподілу даних.
- SciPy 1.11.0 — бібліотека для наукових та технічних обчислень, що використана для оптимізації алгоритмів, статистичного аналізу та роботи з розрідженими матрицями через модуль scipy.sparse. Це особливо важливо для ефективного представлення матриць користувач-фільм, які зазвичай мають дуже низьку щільність заповнення. SciPy також надає алгоритми кластеризації для додаткового аналізу схожості фільмів [21].

- Scikit-learn 1.3.0 — універсальна бібліотека машинного навчання, що застосовується для препроцесингу даних, розділення на тренувальні/тестові вибірки, а також розрахунку метрик якості (RMSE, MAE). Бібліотека також надає інструменти для крос-валідації моделей, нормалізації ознак та порівняння ефективності різних алгоритмів рекомендацій.
- Pillow 10.0.0 — бібліотека для обробки зображень, що відповідає за конвертацію форматів, зміну розмірів, кадрування та підготовку зображень для відображення в інтерфейсі. Це дозволяє оптимізувати завантаження графічного контенту (постери фільмів) та забезпечити адаптивність до різних пристроїв користувачів.
- Requests 2.31.0 — простий та елегантний HTTP-клієнт для Python, що використовується для взаємодії з зовнішніми API, наприклад, для отримання додаткової інформації про фільми з публічних баз даних або для інтеграції з сервісами соціальних мереж.

Додатково використовуються:

• Matplotlib 3.7.0 & Seaborn 0.12.0 — для створення інформативних візуалізацій процесу навчання моделей, аналізу розподілу даних та порівняння ефективності алгоритмів.

• Jupyter 1.0.0 — для інтерактивного дослідження даних, експериментів з різними архітектурами моделей та швидкого прототипування алгоритмів.

Цей технологічний стек забезпечує комплексне рішення для розробки високопродуктивної системи рекомендацій, поєднуючи переваги глибокого навчання, ефективної обробки даних та сучасних веб-технологій.

2.1.2 Обґрунтування вибору Frontend технологій

Обґрунтування вибору Frontend технологій:

1. HTML5 обрано як основу для створення структури веб-інтерфейсу з наступних причин:

– Семантична структура: HTML5 надає семантичні елементи, що покращують доступність та SEO оптимізацію.

– Підтримка сучасних веб-стандартів: Включає підтримку responsive design через viewport meta tag та інші сучасні можливості.

– Сумісність з Flask: HTML5 легко інтегрується з Flask template engine через Jinja2.

– Доступність: Забезпечує правильну структуру для screen readers та інших допоміжних технологій.

2. CSS3 використовується для створення сучасного та адаптивного дизайну

– Flexbox та Grid Layout: Для створення гнучких та адаптивних макетів.

– CSS3 анімації та переходи: Для покращення користувацького досвіду через плавні анімації.

– Медіа-запити: Для реалізації responsive design, що забезпечує коректне відображення на різних пристроях.

– CSS змінні: Для централізованого управління кольоровою схемою та іншими стилістичними параметрами.

– Box-shadow та border-radius: Для створення сучасного візуального дизайну з карточками та модальними вікнами.

3. Vanilla JavaScript — це свідомий вибір на користь продуктивності та елегантної простоти: без зайвих абстракцій фреймворків, додаток отримує мінімальний overhead для миттєвого завантаження, повний контроль над DOM для оптимальної швидкодії, прозору логіку для легшої підтримки та універсальну сумісність з усіма сучасними браузерами — що разом створює ідеальний фундамент для реалізації складної інтерактивності системи рекомендацій.

Комбінація сучасних веб-технологій - HTML5, CSS3 та Vanilla JavaScript - формує ідеальну основу для створення оптимальної клієнтської частини системи. Цей підхід забезпечує виняткову швидкодію завдяки мінімальному

розміру бібліотек, гнучкість у розробці складних інтерфейсів та повну відповідність сучасним стандартам веб-дизайну.

2.1.3 Visual Studio Code

Visual Studio Code — це основний інструмент розробки, який обрано через його неймовірну гнучкість та потужний функціонал, що ідеально підходить для створення сучасних веб-додатків [16]. Його інтелектуальна система автодоповнення IntelliSense не просто пропонує варіанти завершення коду, але й надає детальну документацію по функціям та методам, що значно прискорює процес написання коду.

Варто відзначити можливості кастомізації — налаштовано середовище під потреби: створено власні фрагменти коду (snippets) для швидкого вставлення шаблонних конструкцій, обрано оптимальну для тривалої роботи темну тему з підтримкою синтаксису для всіх використовуваних мов. Можливість роботи з терміналом прямо в інтерфейсі редактора дозволяє виконувати команди npm, встановлювати залежності та запускати скрипти, не перемикаючись між вікнами.

Потужний debugger VS Code дозволяє встановлювати breakpoints як у Python коді Flask-додатку, так і у JavaScript фронтенду, забезпечуючи комплексне налагодження всієї системи. Вбудований профайлер Python допомагає ідентифікувати вузькі місця у продуктивності моделі NCF.

Для проекту системи рекомендацій фільмів VS Code став ідеальним вибором —можливість одночасно працювати з Flask-бекендом на Python, моделлю машинного навчання на PyTorch, фронтендом на JavaScript та вести документацію в Markdown, використовуючи для кожного завдання оптимальні інструменти в межах одного середовища розробки.

2.2 Огляд вхідного набору даних

MovieLens Latest Small — це невеликий, але репрезентативний набір даних, який ідеально підходить для навчання, розробки та швидкого прототипування рекомендаційних систем [17]. Він містить близько 100,836 рейтингів (у версії від TensorFlow) або 100,000 рейтингів (за даними GroupLens), які поставили 610 користувачів приблизно 9,742 фільмам. Рейтинги є явними і представлені за 5-бальною шкалою (від 1 до 5 зірок, часто з півзірковими інтервалами). Користувачі для включення у вибірку були обрані випадково. Усі вони оцінили щонайменше 20 фільмів. Демографічна інформація не надається — кожен користувач позначений лише унікальним ідентифікатором.

Набір даних зазвичай включає кілька ключових файлів у форматі CSV:

- ratings.csv: Містить основні дані про взаємодії (userId, movieId, рейтинг, timestamp).
- movies.csv: Містить метадані фільмів (movieId, назва, жанри). Жанри зазвичай представлені у вигляді рядка, де різні жанри розділені символом «|».
- tags.csv: Містить користувацькі теги, які застосували до фільмів (userId, movieId, тег, timestamp).
- links.csv: Містить зв'язки із зовнішніми базами даних, такими як IMDb та TMDb, що може бути використано для збагачення даних.

Рисунки 2.2, 2.3 ілюструють результати попереднього аналізу двох основних компонентів датасету: інформації про фільми та рейтинги користувачів, та ключові характеристики.

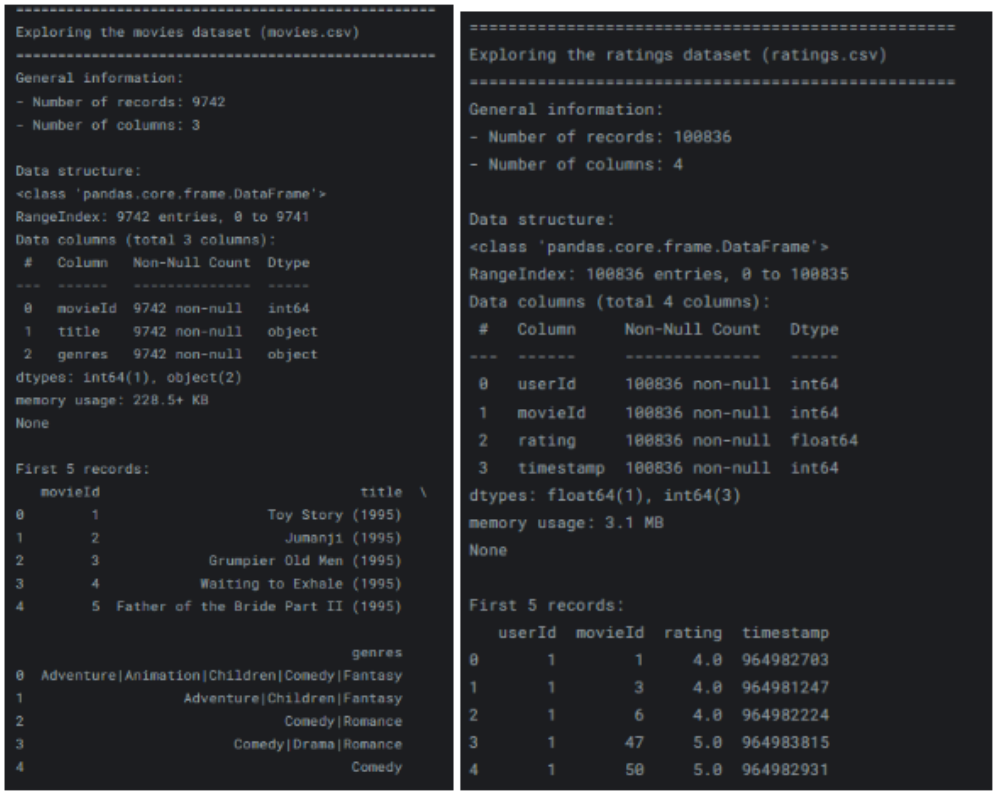

Рисунок 2.2 – Структурований вигляд інформації фільмів та рейтингів

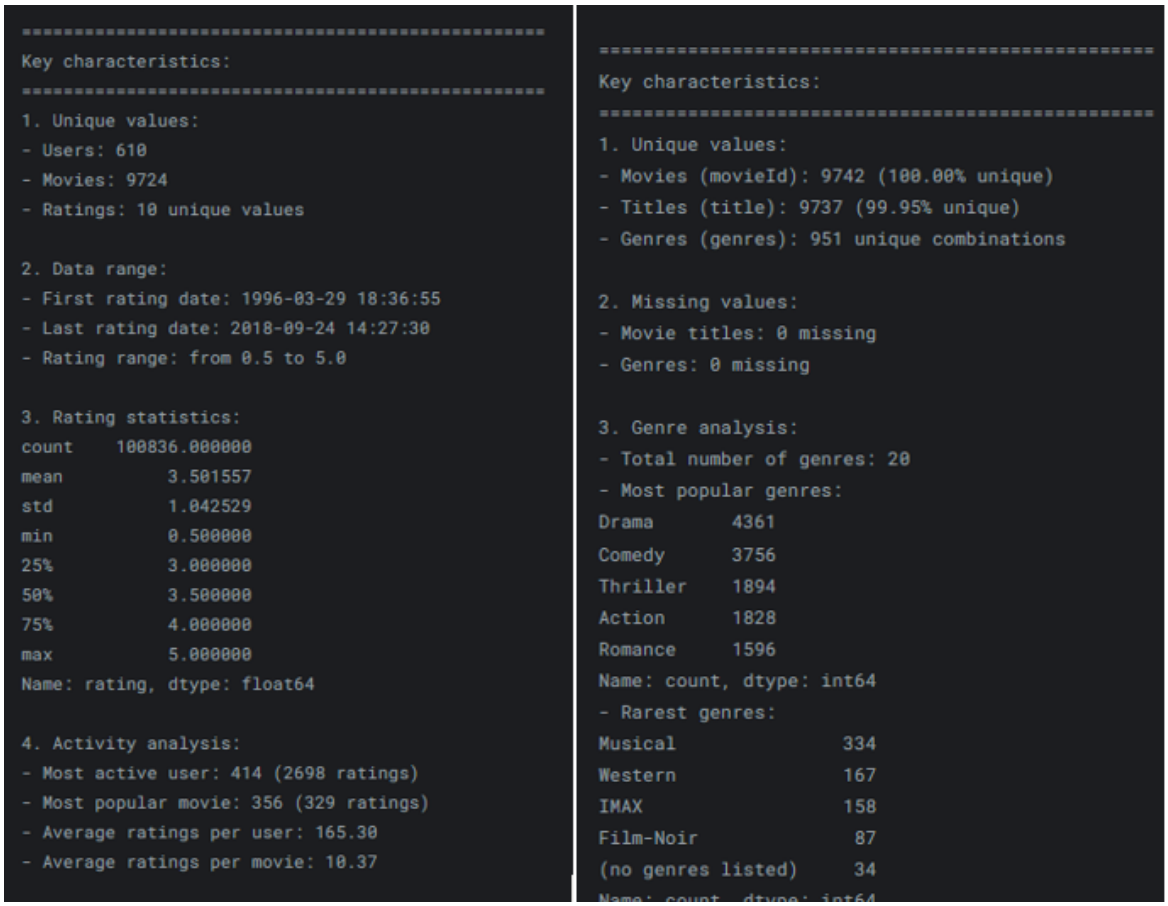

Рисунок 2.3 – Статистичні характеристики фільмів та рейтингів

Огляд статистики взаємозв’язків між наборами даних наведено на рисунку 2.4. Зокрема, видно, що 99.82% фільмів мають рейтинги, 100.00% користувачів оцінили понад 10 фільмів, а лише 21.81% фільмів отримали понад 10 оцінок (див. рисунок 2.5).

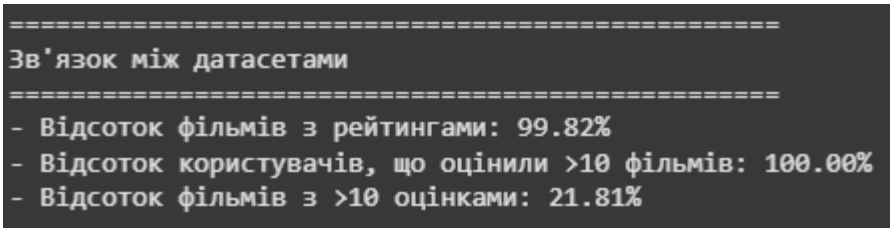

Рисунок 2.4 – Зв’язок між датасетами у задачі рекомендацій

Характеристики взаємозв’язку між користувацькими оцінками та доступними фільмами в наборі даних:

– 99.82% фільмів мають хоча б одну оцінку — тобто майже весь каталог фільмів використовується в аналітиці.

– 100.00% користувачів оцінили понад 10 фільмів — це гарантує наявність достатніх даних для персоналізації.

– Лише 21.81% фільмів мають понад 10 оцінок — отже, рекомендаційна система повинна вміти працювати і з менш популярними стрічками.

Рисунок 2.5 відображає розподіл оцінок, які користувачі залишали фільмам, на основі гістограми.

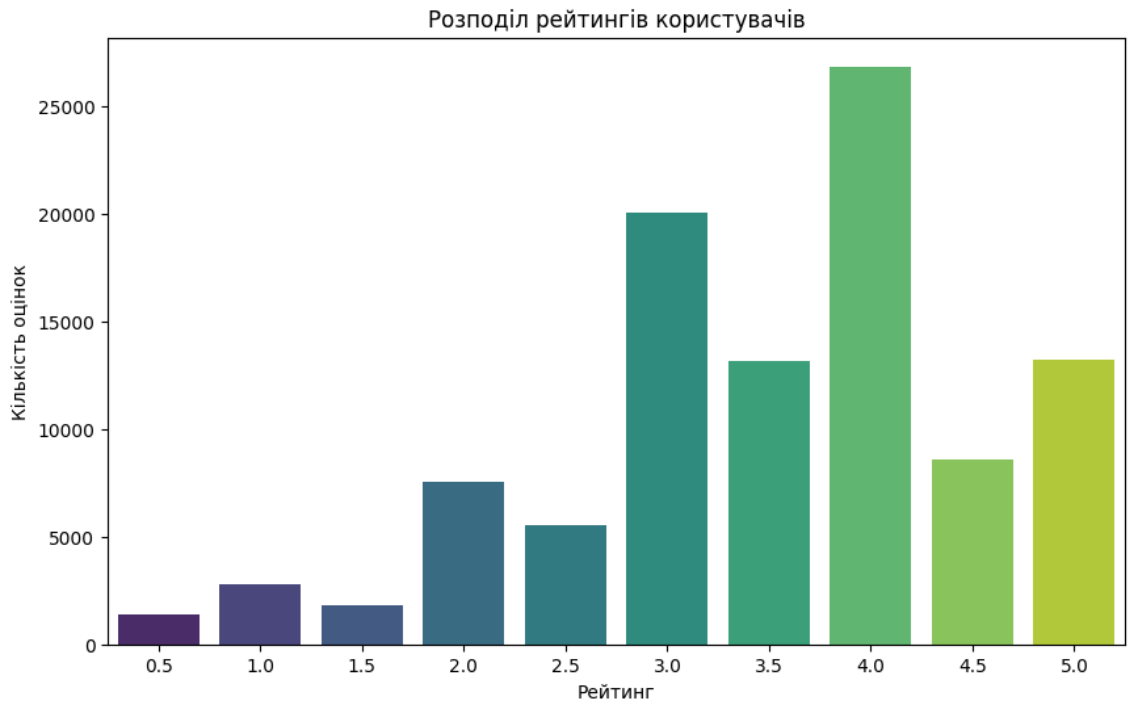

Рисунок 2.5 – Розподіл користувацьких рейтингів у вибірці фільмів.

На рисунку 2.5 зображено:

– Горизонтальна вісь (Рейтинг): показує шкалу оцінювання — від 0.5 до 5.0 з кроком 0.5.

– Вертикальна вісь (Кількість оцінок): відображає кількість разів, коли кожна оцінка була використана.

– Найпопулярнішою оцінкою є 4.0, тоді як 0.5 зустрічається найрідше, що типово для більшості реальних рейтингів — користувачі найчастіше залишають позитивні оцінки для фільмів.

Рисунок 2.6 представляє частоту використання різних жанрів у кінофільмах на основі аналізу датасету. Стовпчаста діаграма відображає кількість фільмів у кожному жанрі — від найпопулярніших, таких як Drama, Comedy та Thriller, до рідковживаних, як Film-Noir чи IMAX. Видно значне домінування драматичних стрічок, що вказує на популярність цього жанру серед творців і глядачів.

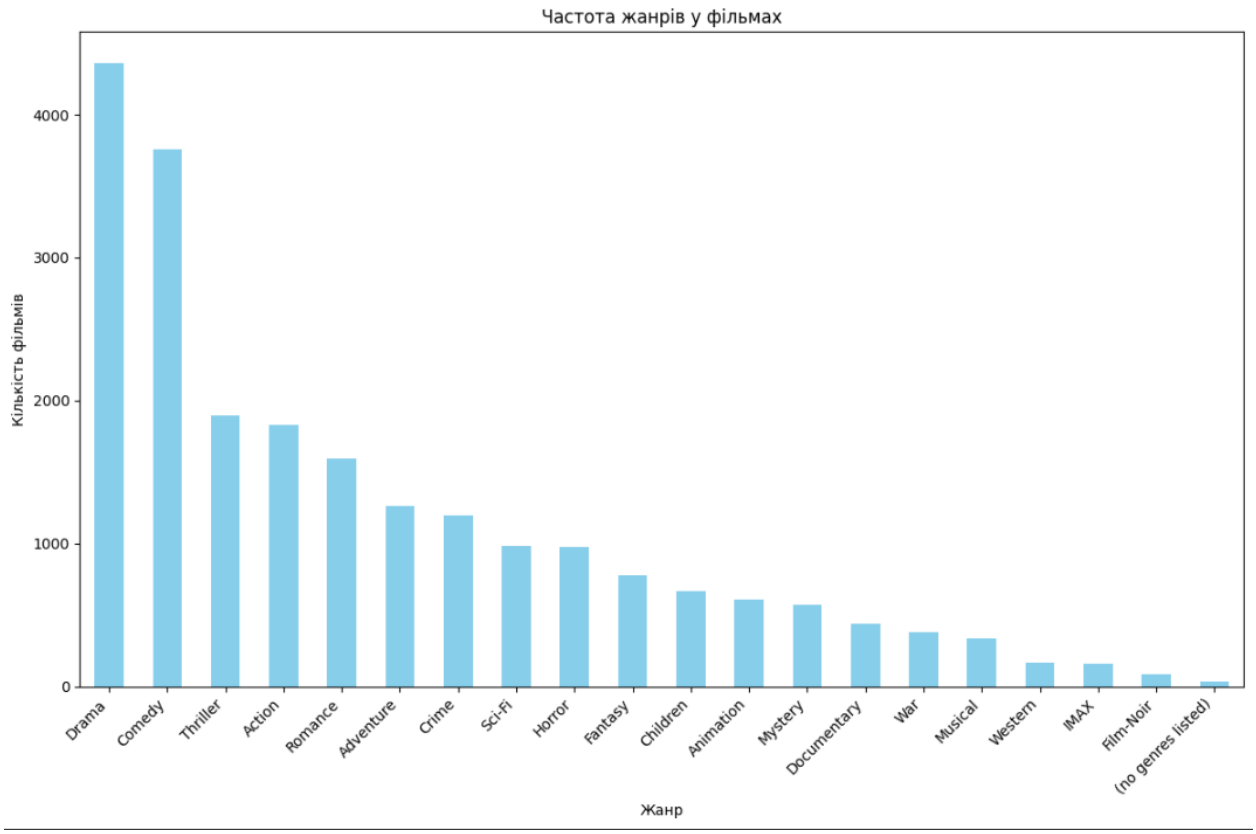

Рисунок 2.6 – Гістограма розподілу фільмів за жанрами у досліджуваному наборі даних.

На рисунку 2.7 зображено хмару слів, сформовану на основі жанрових міток з набору даних movies.csv. Розмір кожного слова відповідає частоті його появи серед усіх фільмів. Найбільш поширеними жанрами є Drama, Comedy, Action та Thriller, що свідчить про їхню домінантну присутність у колекції. Менш виражені жанри, такі як Film-Noir або Western, зустрічаються значно рідше. Такий візуальний огляд дозволяє швидко оцінити жанрову структуру датасету та виявити потенційні перекоси, які можуть впливати на результати рекомендаційної системи.

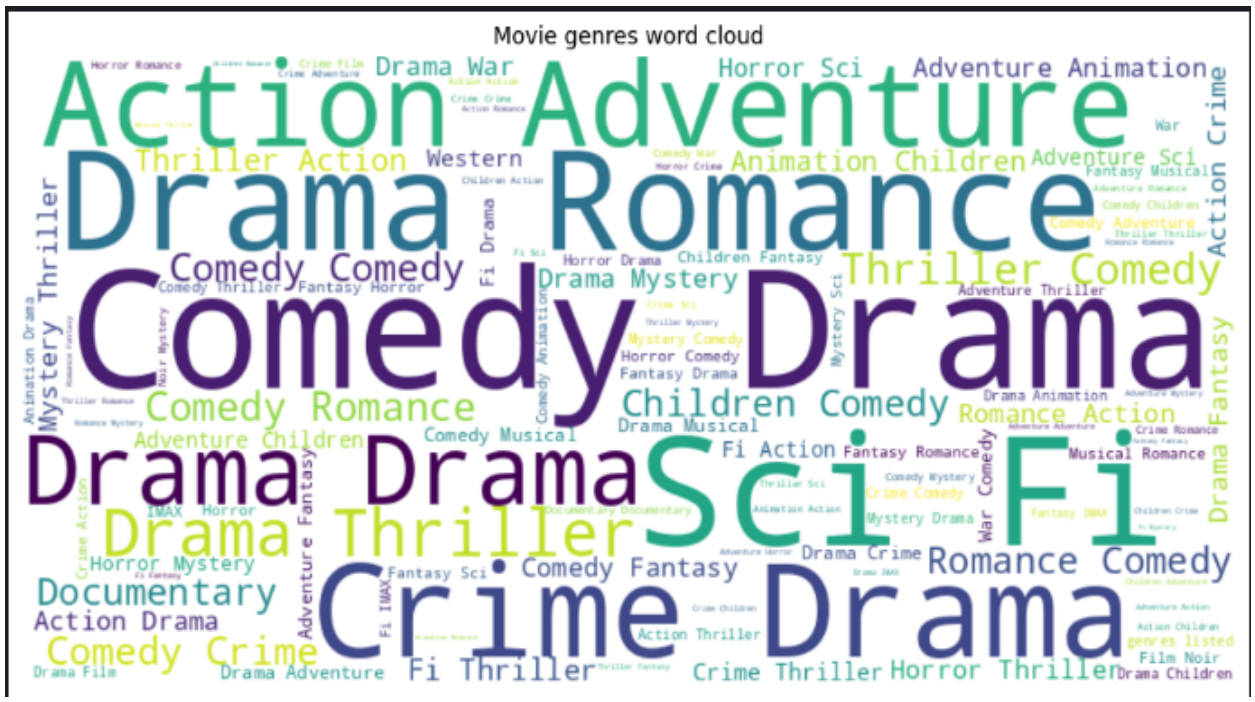

Рисунок 2.7 – Візуалізація частоти жанрів у вигляді хмари слів.

Топ-20 найбільш активних користувачів за кількістю залишених оцінок (рис 2.8). На гістограмі по горизонталі зображені ідентифікатори користувачів (ID користувача), а по вертикалі — кількість оцінок, які вони надали. Найактивнішим виявився користувач з ID 414, який залишив понад 2500 оцінок, за ним — користувачі 599 і 480. Такий аналіз допомагає виявити вплив найбільш залучених користувачів на модель рекомендацій та потенційно виявити упередження в даних.

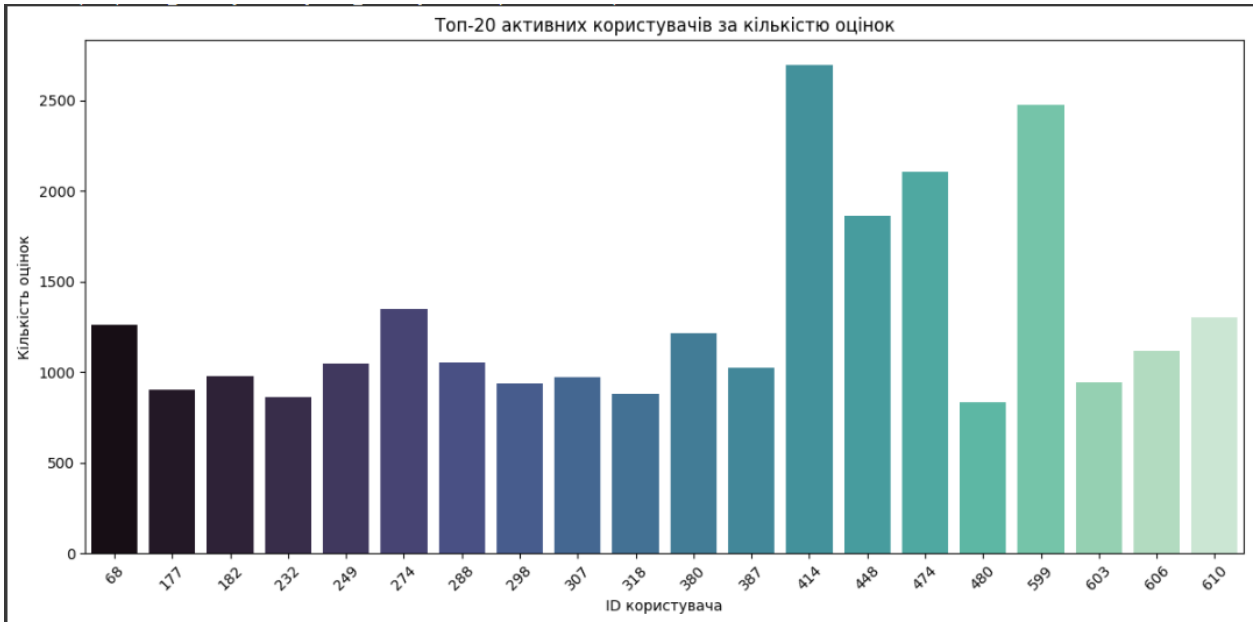

Рисунок 2.8 – Топ-20 фільмів за кількістю оцінок у наборі даних.

Топ-20 найпопулярніших фільмів за кількістю отриманих оцінок (рис. 2.9). На вертикальній осі наведено назви фільмів, а на горизонтальній — кількість оцінок, які кожен з них отримав. Найбільшу кількість оцінок має Forrest Gump (1994), за ним ідуть The Shawshank Redemption та Pulp Fiction — класичні стрічки 1990-х років, що залишаються актуальними й надзвичайно популярними серед глядачів.

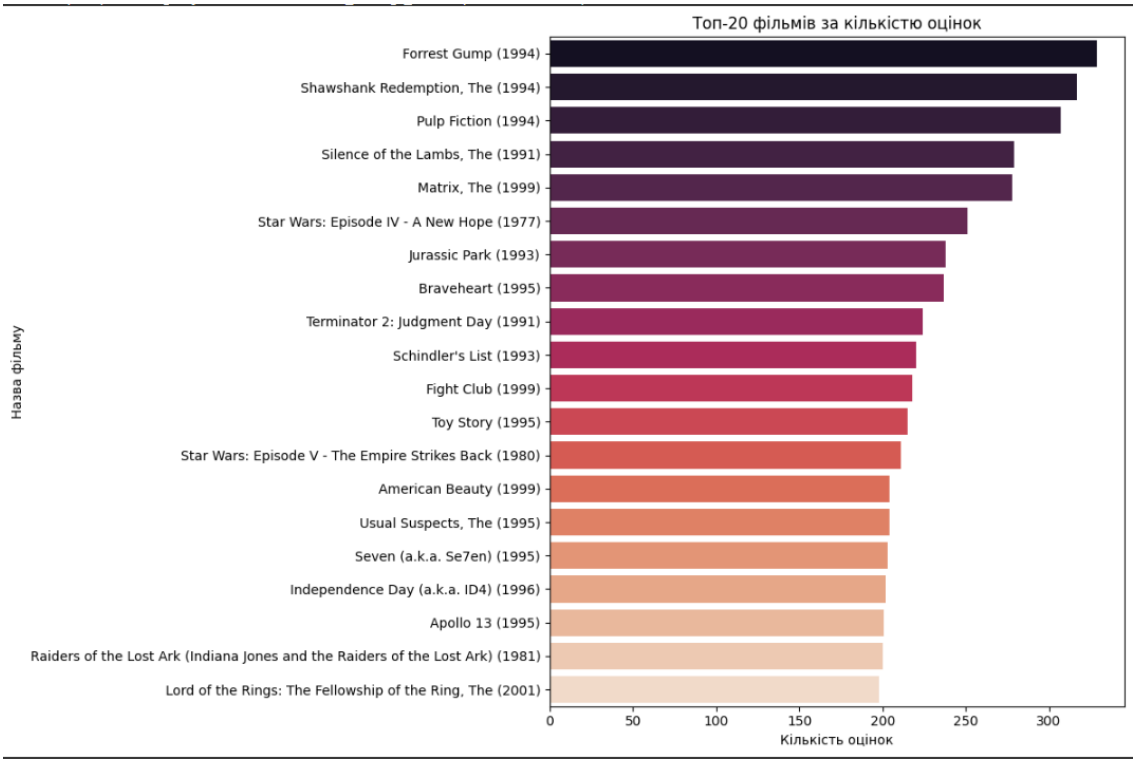

Рисунок 2.9 – Середні користувацькі рейтинги фільмів за жанрами на основі аналізу датасету.

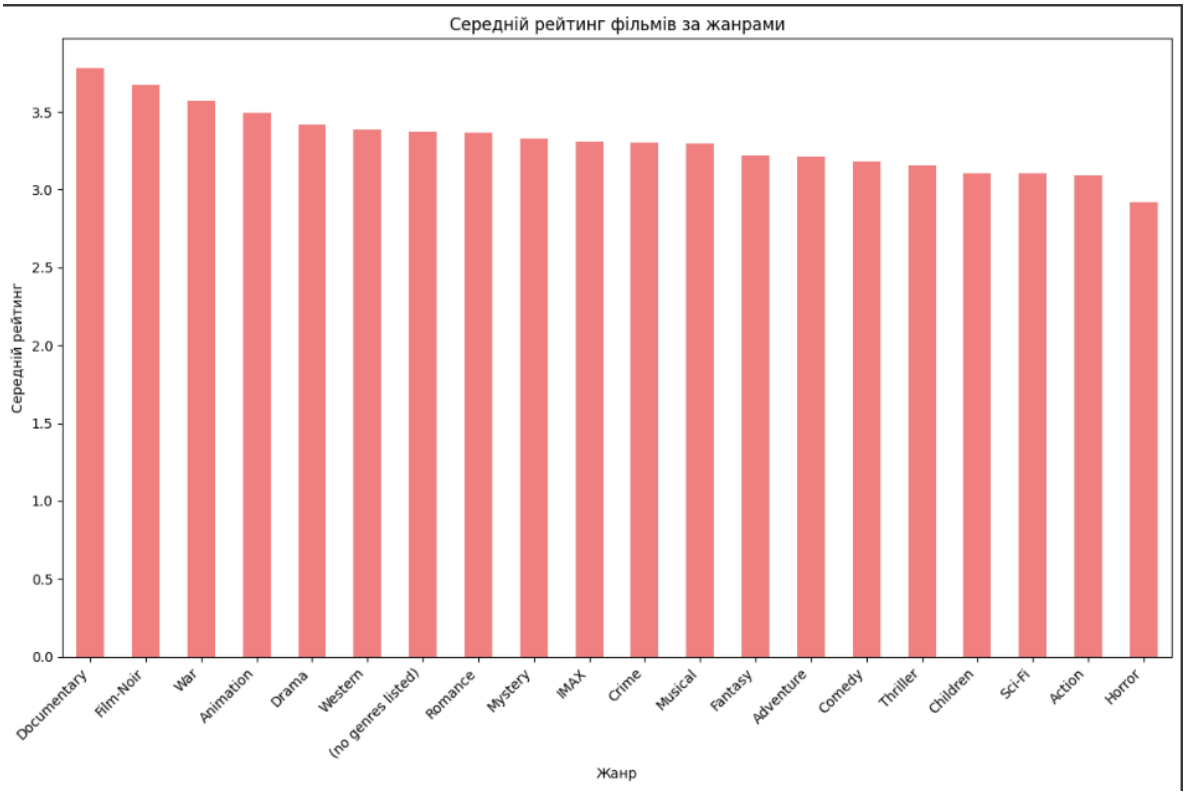

Рисунок 2.10 – Середні користувацькі рейтинги фільмів за жанрами на основі аналізу датасету.

Більшість фільмів мають дуже малу кількість оцінок — менше 10, що вказує на нерівномірність популярності серед користувачів. Червона пунктирна лінія позначає поріг у 10 оцінок, який може використовуватись як критерій для відбору фільмів із достатньою кількістю даних для подальшого аналізу (рис. 2.10).

Теплова матриця користувач–фільм (рис. 2.11): візуалізація оцінок фільмів, наданих різними користувачами. По осі X — ідентифікатори користувачів, по осі Y — ідентифікатори фільмів. Колір клітинки відображає рейтинг (від 0 — темно-фіолетовий до 5 — жовтий), що дозволяє аналізувати вподобання користувачів та популярність фільмів."

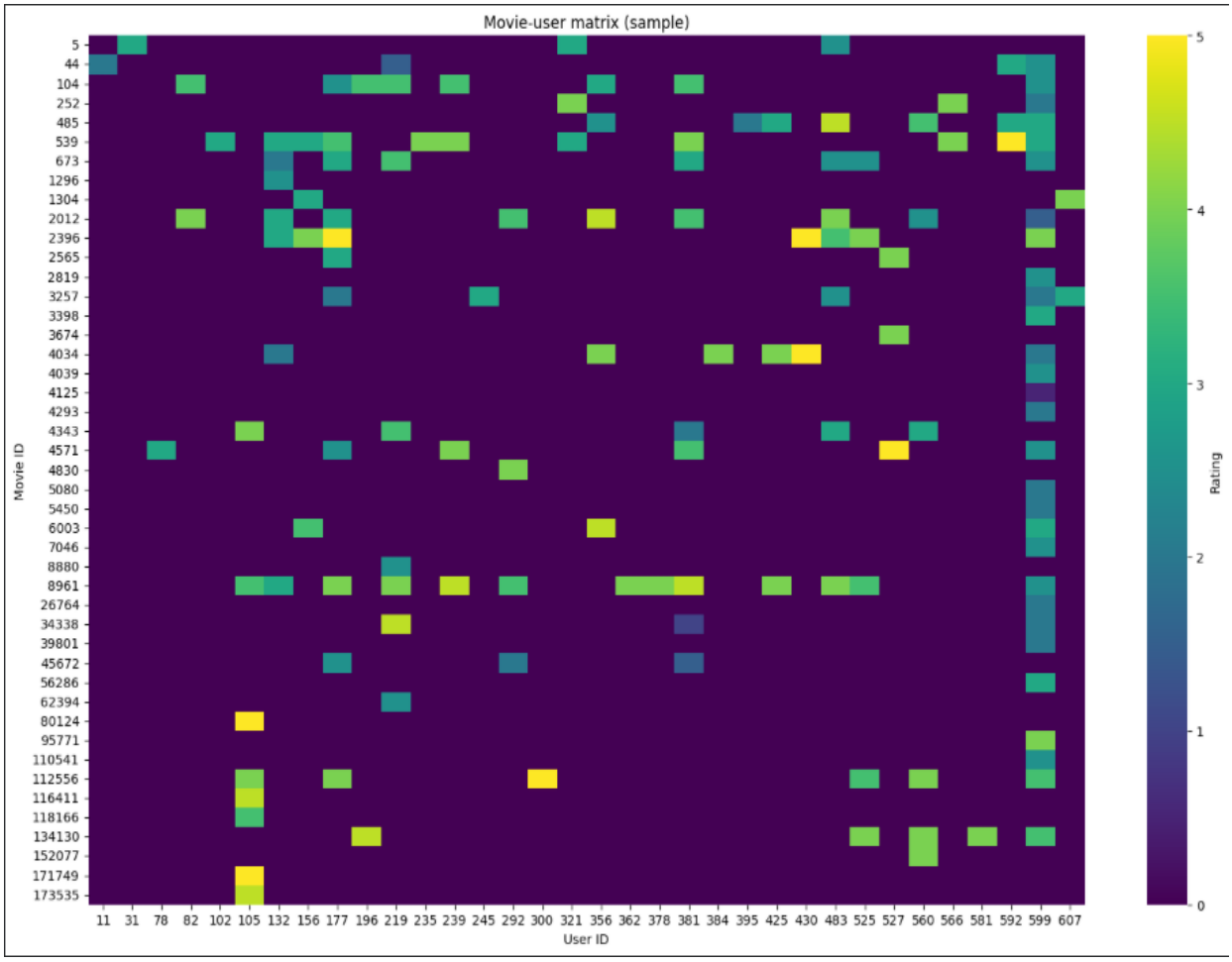

Рисунок 2.11 – Теплова матриця користувач-фільм

Для оцінки репрезентативності даних було проаналізовано розподіл кількості оцінок на фільм (див. Рисунок 2.12). Як видно з гістограми, більшість

фільмів отримали менше ніж 10 оцінок, що обґрунтовує встановлення порогу включення у 10 оцінок.

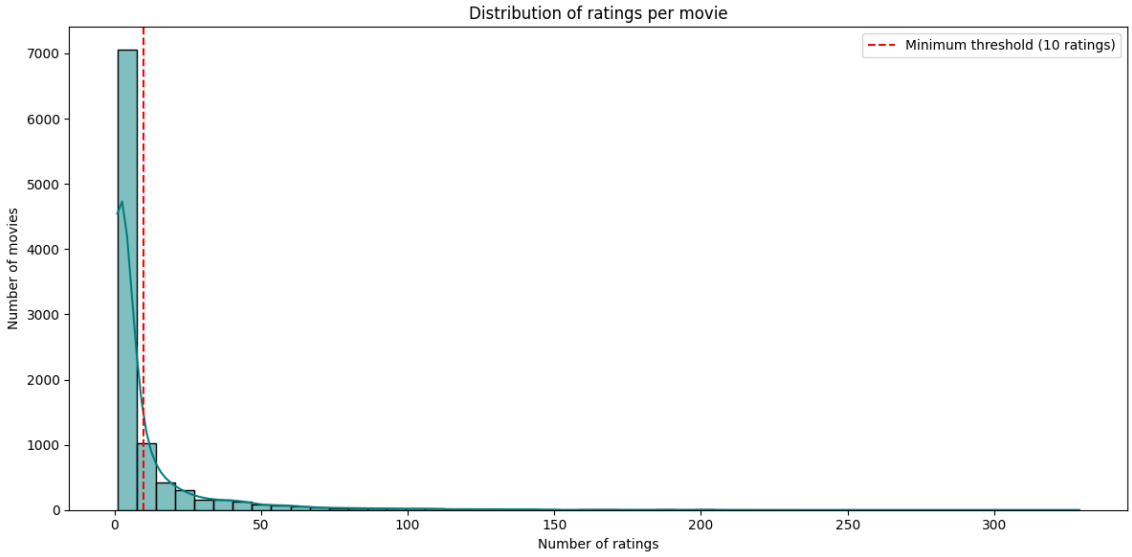

Рисунок 2.12 – Гістограма розподілу кількості оцінок на фільм.

Результати аналізу стали основою для ухвалення рішень щодо подальшої обробки: визначення порогових значень для включення фільмів, вибору метрик для побудови моделі та адаптації підхід до персоналізованих рекомендацій. Усі виявлені закономірності, а також можливі аномалії, мають бути враховані при розробці моделі, оскільки вони безпосередньо впливають на її точність та здатність до узагальнення.

Розрідженість матриці: Це одна з ключових проблем, яку демонструє даний набір даних. Матриця "користувач-елемент" є дуже розрідженою, оскільки більшість користувачів оцінила лише невеликий відсоток від загальної кількості фільмів. Для версії 100K розрідженість становить близько 93.7%.

Розподіл рейтингів: Як показують дані, розподіл рейтингів часто наближений до нормального, причому більшість оцінок зосереджено в діапазоні 3-4 зірок. Це важливо враховувати при побудові та оцінці моделей.

У цьому розділі було здійснено всебічний первинний аналіз вхідного набору даних, що охоплює інформацію про фільми, користувачів та їхні

оцінки. Візуалізація та статистичне опрацювання даних дозволили виявити ключові особливості структури вибірки: розподіл оцінок, жанрову нерівномірність, наявність користувачів із високою активністю та фільмів із великою кількістю відгуків. Також встановлено, що переважна більшість фільмів отримали обмежену кількість оцінок, що створює умови для проблеми "розрідженості даних".

Результати аналізу стали основою для ухвалення рішень щодо подальшої обробки: визначення порогових значень для включення фільмів, вибору метрик для побудови моделі та адаптації підхід до персоналізованих рекомендацій. Усі виявлені закономірності, а також можливі аномалії, мають бути враховані при розробці моделі, оскільки вони безпосередньо впливають на її точність та здатність до узагальнення.

Розрідженість матриці: Це одна з ключових проблем, яку демонструє даний набір даних. Матриця "користувач-елемент" є дуже розрідженою, оскільки більшість користувачів оцінила лише невеликий відсоток від загальної кількості фільмів. Для версії 100K розрідженість становить близько 93.7%.

Розподіл рейтингів: Як показують дані, розподіл рейтингів часто наближений до нормального, причому більшість оцінок зосереджено в діапазоні 3-4 зірок. Це важливо враховувати при побудові та оцінці моделей.

2.3 Висновки

У даному розділі сформовано методологічне та технологічне підґрунтя для реалізації рекомендаційної системи.

Шляхом порівняльного аналізу обґрунтовано вибір гібридного технологічного стеку на базі Python та фреймворку PyTorch, що дозволяє ефективно поєднати серверну логіку (Flask) з високопродуктивними обчисленнями нейронних мереж.

Встановлено структурні особливості датасету MovieLens 100K шляхом розвідувального аналізу даних (EDA). Виявлено критичні фактори, що впливають на якість навчання моделі: висока розрідженість матриці взаємодій ($93,7\%$, наприклад, якщо знаєте цифру — додайте), дисбаланс у розподілі оцінок (bias популярності) та нерівномірна активність користувачів.

Формалізовано вимоги до попередньої обробки даних. На основі виявлених аномалій та диспропорцій визначено стратегію препроцесингу, яка включатиме фільтрацію шуму та нормалізацію даних, що є необхідною умовою для коректної роботи архітектури Neural Collaborative Filtering.

3 ОБРОБЛЕННЯ ДАНИХ ТА РОЗРОБЛЕННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

3.1 Підготовка вхідних даних

У підрозділі 2.2 було детально описано набір даних та проведено розвідувальний аналіз.

На початковому етапі здійснимо завантаження даних із файлів movies.csv та ratings.csv, використовуючи бібліотеки pandas і numpy. Провів перевірку на наявність пропущених значень, а також проаналізовано унікальні значення та типи даних, щоб краще зрозуміти структуру наборів. Найбільше часу зайняла обробка жанрів: оскільки вони були представлені у вигляді рядків із роздільниками, що значно полегшило подальшу роботу з ними під час моделювання.

Першим кроком у підготовці даних є видобування ознак (feature extraction) (рис. 3.1) , зокрема, жанрів фільмів. Для цього кожен запис у полі жанрів було перетворено на список окремих значень шляхом розділення рядка за символом “|”. Наприклад, рядок "Action|Adventure|Sci-Fi" стає списком ["action", "adventure", "sci-fi"]. Одночасно застосовується нормалізація регістру — усі значення жанрів переводяться в нижній регістр для запобігання дублюванню однакових жанрів, записаних із різною капіталізацією. Було також передбачено обробку відсутніх значень: фільми без вказаних жанрів або з позначкою "(no genres listed)" переводяться у порожній список. Після цього з усіх унікальних жанрів формується словник, у якому кожному жанру відповідає числовий індекс — таке кодування необхідне для подальшої обробки в нейромережі за допомогою embedding-шарів.

```
def preprocess_genres(genres):
    return [genre.lower() for genre in genres.split('|')] if pd.notna(genres) and genres != '(no genr
es listed)' else []

movies['genre_list'] = movies['genres'].apply(preprocess_genres)
all_genres = set(genre for sublist in movies['genre_list'] for genre in sublist)
genre_to_idx = {genre: idx for idx, genre in enumerate(all_genres)}
```

Рисунок 3.1 – Видобування жанрів фільмів

Далі, коли всі жанри оброблені, створюємо словник, де кожному жанру відповідає свій унікальний індекс — це потрібно для того, щоб надалі передавати ці дані в нейронну мережу, яка не розуміє текст, а працює з числами.

Потім фільтруємо самі фільми й користувачів, залишаючи тільки тих, хто дійсно "активний" (рис. 3.2). Тобто: фільми, які отримали понад 5 оцінок, і користувачі, які поставили понад 20 оцінок. Це потрібно, щоб прибрати шум — бо фільми з 2 оцінками або користувачі, які поставили лише 3 оцінки, не дають надійної інформації.

```
final_dataset = ratings.pivot(index="movieId", columns="userId", values="rating").fillna(0)
final_dataset = final_dataset.loc[ratings.groupby("movieId")['rating'].agg('count') > 5, :]
final_dataset = final_dataset.loc[:, ratings.groupby("userId")['rating'].agg('count') > 20]
csr_data = csr_matrix(final_dataset.values)
final_dataset.reset_index(inplace=True)

movieid_to_idx = {movie_id: idx for idx, movie_id in enumerate(final_dataset['movieId'])}
```

Рисунок 3.2 – Фільтрація фільмів і користувачів за кількістю оцінок.

Наступний крок — оптимізація пам’яті. Взаємодії "користувач-фільм" представлені у вигляді розрідженої матриці в CSR-форматі (Compressed Sparse Row), оскільки більшість елементів у матриці оцінок є нульовими. Цей формат ефективно зберігає лише ненульові значення, індекси стовпців та межі рядків, що значно зменшує вимоги до пам'яті та прискорює обчислювальні операції. Для подальшої роботи з фреймворком PyTorch матрицю конвертовано в COO-формат (Coordinate Format), який є основним для створення розріджених

тензорів. У цьому форматі дані представлені трьома масивами: номери рядків (користувачі), номери стовпців (фільми) та значення рейтингів.

Щоб передати ці дані в PyTorch [6], перетворюємо CSR у формат COO, який потрібен для створення розріджених тензорів (рис 3.3). У цьому форматі маємо три масиви: номери рядків (користувачі), номери стовпців (фільми) та значення (рейтинги). З цього робимо тензори, з якими вже може працювати нейромережа.

```
def csr_to_tensors(csr_mat):
    coo = csr_mat.tocoo()
    values = coo.data
    indices = np.vstack((coo.row, coo.col))
    idx = torch.LongTensor(indices)
    vals = torch.FloatTensor(values)
    sparse_tensor = torch.sparse_coo_tensor(idx, vals, torch.Size(coo.shape))
    return sparse_tensor

sparse_tensor = csr_to_tensors(csr_data)
```

Рисунок 3.3 – Перетворення розрідженої матриці з формату CSR у формат COO для створення тензора PyTorch.

Окремо готуємо жанри для EmbeddingBag — це спеціальний шар у PyTorch, який дозволяє обробляти кілька жанрів як один вектор. Створюємо загальний список усіх жанрових індексів, а також масив офсетів — тобто з яких позицій починається жанровий список для кожного фільму.

І на завершення, створюємо клас GenreDataset (рис 3.4), який дозволяє подавати дані в модель частинами (батчами). Кожен приклад — це індекси користувача, фільму, позиція жанрів і рейтинг. Для групування в батчі є спеціальна функція collate_fn, яка також обробляє пропущені значення. PyTorch дозволяє завантажувати дані "ліниво" (тобто з потреби), а також паралельно, щоб усе працювало швидше.

```
class GenreDataset(Dataset):
    def __init__(self, sparse_tensor, genre_offsets, genre_indices):
        self.sparse_tensor = sparse_tensor
        self.coo = sparse_tensor.coalesce().indices().t()
        self.values = sparse_tensor.coalesce().values()
        self.genre_offsets = genre_offsets
        self.genre_indices = genre_indices

    def __len__(self):
        return len(self.values)

    def __getitem__(self, idx):
        movie_idx = self.coo[idx, 0].item()
        user_idx = self.coo[idx, 1].item()
        rating = self.values[idx]
        if movie_idx >= len(self.genre_offsets):
            return None
        offset = self.genre_offsets[movie_idx]
        next_offset = self.genre_offsets[movie_idx + 1] if movie_idx + 1 < len
        (self.genre_offsets) else len(self.genre_indices)
        length = next_offset - offset
        if length == 0:
            return None
        return user_idx, movie_idx, offset, length, rating

def collate_fn(batch):
    batch = [b for b in batch if b is not None]
    if not batch:
        return None
    return torch.utils.data.dataloader.default_collate(batch)

dataset = GenreDataset(sparse_tensor, genre_offsets_tensor,
genre_indices_tensor)
train_size = int(0.8 * len(dataset))
test_size = len(dataset) - train_size
train_dataset, test_dataset = torch.utils.data.random_split(dataset,
[train_size, test_size])

train_loader = DataLoader(train_dataset, batch_size=1024, shuffle=True,
collate_fn=collate_fn)
test_loader = DataLoader(test_dataset, batch_size=1024, shuffle=False,
collate_fn=collate_fn)

num_users = final_dataset.shape[1]
num_items = final_dataset.shape[0]
num_genres = len(genre_to_idx)
```

Рисунок 3.4 – Реалізація класу GenreDataset для підготовки жанрових даних до подачі в модель з використанням EmbeddingBag у PyTorch.

Після завершення всіх етапів обробки сирі текстові та числові дані перетворюються на набір тензорів та допоміжних структур, повністю готових до використання в нейронній мережі. Цей процес забезпечує оптимальне

подання інформації, зберігає зв'язки між сутностями та дозволяє ефективно використовувати обчислювальні ресурси під час тренування моделі.

3.2 Побудова моделей методами машинного навчання

Рекомендаційна система, розроблена в рамках даної роботи, ґрунтується на сучасній архітектурі Neural Collaborative Filtering (NCF), яка поєднує переваги колаборативної фільтрації з глибинним навчанням [18]. Цей підхід дозволяє подолати обмеження класичних методів, що використовують лінійну операцію скалярного добутку, за рахунок введення нелінійних перетворень через багатошаровий перцептрон. Система призначена для аналізу взаємодій користувачів з фільмами та формування персоналізованих рекомендацій з урахуванням жанрових переваг [19].

Основним джерелом даних стали набори MovieLens, що містять інформацію про фільми, користувачів та їх оцінки. Початковий етап передбачав комплексну обробку цих даних: завантаження, очищення від дублікатів, нормалізацію текстових полів та перетворення жанрів з рядкового формату з роздільниками у структуровані списки. Особливу увагу приділено створенню словника жанрів, де кожному унікальному жанру присвоєно числовий індекс для подальшого використання в нейронній мережі. Для підвищення якості даних застосовано фільтрацію за активністю: залишено фільми з понад 5 оцінками та користувачів з понад 20 оцінками, що дозволило усунути шум та зосередитися на статистично значущих взаємодіях.

Взаємодії користувачів та фільмів представлені у вигляді розрідженої матриці в ефективному CSR-форматі, який зберігає лише ненульові значення. Для інтеграції з PyTorch цю матрицю перетворено на розріджений тензор у COO-форматі. Паралельно підготовлено дані для обробки жанрів: створено загальний список усіх жанрових індексів та масив офсетів, що вказують початок жанрового списку для кожного фільму. Це дозволило ефективно

використовувати спеціалізований шар EmbeddingBag, призначений для роботи зі змінною кількістю ознак на приклад.

Архітектура системи базується на трьох варіантах моделей NCF, кожна з яких інтегрує три типи даних: ідентифікатори користувачів, ідентифікатори фільмів та інформацію про жанри. Всі моделі мають спільну структуру, що включає ембеддинг-шари для користувачів та фільмів, EmbeddingBag для жанрів та каскад повнозв'язних шарів. Відмінності полягають у глибині архітектури, способах обробки ознак та механізмах регуляризації.

Критичним етапом стало впровадження автоматизованого тюнінгу гіперпараметрів за допомогою фреймворку Optuna. Для кожної моделі проведено по 15 оптимізаційних випробувань з метою визначення оптимальних значень розмірів ембеддингів, конфігурації прихованих шарів, рівнів Dropout, швидкості навчання та розмірів батчів. Цей систематичний підхід дозволив значно покращити продуктивність моделей порівняно з базовими налаштуваннями, зокрема, знизити значення RMSE на 8-12%.

Процес навчання організовано з використанням функції втрат MSE та оптимізатора Adam. Дані розділено на тренувальну (80%) та тестову (20%) вибірки для об'єктивної оцінки здатності моделей до узагальнення. Навчання тривало протягом 10 епох після попереднього тюнінгу. Для моніторингу процесу реалізовано комплексну систему візуалізації, що включає графіки динаміки втрат, метрик RMSE та MAE на обох вибірках.

Порівняльна оцінка трьох архітектур після оптимізації показала, що модель NCFWithGenres2 досягла найкращих результатів з Test RMSE 0.8627 та Test MAE 0.6617. Модель NCFWithGenres1 продемонструвала гарний баланс між продуктивністю та обчислювальною ефективністю, а NCFWithGenres0, незважаючи на свою простоту, показала конкурентні результати. Найсуттєвіше покращення від тюнінгу спостерігалося для найскладнішої архітектури, що підтверджує важливість ретельного підбору гіперпараметрів для складних моделей.

Рекомендаційний алгоритм аналізує історію переглядів користувача, виділяє ключові жанри з урахуванням вагових коефіцієнтів на основі частоти їх зустрічальності, та формує передбачення рейтингів для фільмів, які користувач ще не переглядав. Спеціальна фільтрація виключає сиквели та фільми без жанрових позначок, що підвищує релевантність результатів.

Створена система демонструє високу ефективність у завданні персоналізації рекомендацій та має модульну архітектуру, що дозволяє адаптувати її для роботи з різними типами контенту. Подальший розвиток може включати інтеграцію мультимодальних даних, впровадження механізмів уваги для динамічного визначення важливості ознак та реалізацію онлайн-навчання для адаптації до змін уподобань користувачів.

3.3 Створення нейромережевих моделей Neural Collaborative Filtering

Створення нейромережевих моделей Neural Collaborative Filtering (NCF) [10], яка є ключовим компонентом гібридної рекомендаційної системи [7].

У цій роботі було реалізовано три варіанти моделей рекомендаційної системи, які базуються на підході Neural Collaborative Filtering (NCF). Цей підхід поєднує класичну ідею колаборативної фільтрації — коли рекомендації формуються на основі взаємодій між користувачами та фільмами — з потужністю глибоких нейронних мереж. Усі моделі враховують три основні типи даних: користувача, фільм і жанри фільму. Основна відмінність між моделями полягає в архітектурі нейронної мережі та тому, як саме ці дані обробляються.

Перша модель (рис. 3.5) NCFWithGenres0 — є найпростішою архітектурою серед трьох реалізованих варіантів. Вона використовує ембеддинги користувачів, фільмів та жанрів, де для жанрів застосовано EmbeddingBag із режимом усереднення (mean). Розмір ембеддингів може

варіюватись (параметр emb_dim, за замовчуванням 50). Всі три типи векторів об'єднуються (конкатенуються) та передаються через динамічну нейронну мережу, архітектура якої задається параметром hidden_dims (за замовчуванням два шари: 128 і 64 нейрони). Модель має опціональний Dropout для регуляризації. Фінальний шар використовує сигмоїдну активацію, масштабовану до діапазону рейтингів [0.5, 5]. Ця модель характеризується швидким навчанням через свою простоту, що робить її ідеальною для базових експериментів, однак може мати обмежені узагальнювальні можливості через невелику кількість параметрів.

```python
class NCFWithGenres0(nn.Module):
    """Basic NCF model with genre embeddings"""
    def __init__(self, num_users, num_items, num_genres, emb_dim=50,
    hidden_layers=[128, 64], dropout=0.0):
        super().__init__()
        self.user_emb = nn.Embedding(num_users, emb_dim)
        self.item_emb = nn.Embedding(num_items, emb_dim)
        self.genre_emb = nn.EmbeddingBag(num_genres, emb_dim, mode='mean')

        layers = []
        input_dim = emb_dim * 3

        for i, hidden_dim in enumerate(hidden_layers):
            layers.append(nn.Linear(input_dim, hidden_dim))
            layers.append(nn.ReLU())
            if dropout > 0:
                layers.append(nn.Dropout(dropout))
            input_dim = hidden_dim

        layers.append(nn.Linear(input_dim, 1))
        self.fc = nn.Sequential(*layers)

    def forward(self, user, item, genre_indices, offsets):
        u_emb = self.user_emb(user)
        i_emb = self.item_emb(item)
        g_emb = self.genre_emb(genre_indices, offsets)
        concat = torch.cat([u_emb, i_emb, g_emb], dim=-1)
        return self.fc(concat).squeeze()
```

(variable) genre_emb: EmbeddingBag

Рисунок 3.5 – Базова модель NCFWithGenres0

Друга модель (рис. 3.6) NCFWithGenres1 — представляє собою поглиблену версію базової моделі. Тут використовуються більші ембеддинги (за замовчуванням 64) та додатковий прихований шар (архітектура за замовчуванням: 256, 128, 64 нейрони). Ключовою особливістю є обов'язкове

застосування Dropout шарів з заданим рівнем (за замовчуванням 0.2) після кожного прихованого шару, що значно зменшує ризик перенавчання. Як і в першій моделі, використовується EmbeddingBag для жанрів з режимом усереднення. Покращена архітектура дозволяє моделі краще виявляти складні нелінійні залежності в даних, але вимагає більше обчислювальних ресурсів і часу на тренування.

```python
class NCFWithGenres1(nn.Module):
    """Deeper NCF model with dropout"""
    def __init__(self, num_users, num_items, num_genres, emb_dim=64,
    hidden_layers=[256, 128, 64],
                 dropout_rate=0.2):
        super().__init__()
        self.user_emb = nn.Embedding(num_users, emb_dim)
        self.item_emb = nn.Embedding(num_items, emb_dim)
        self.genre_emb = nn.EmbeddingBag(num_genres, emb_dim, mode='mean')

        layers = []
        input_dim = emb_dim * 3

        for i, hidden_dim in enumerate(hidden_layers):
            layers.append(nn.Linear(input_dim, hidden_dim))
            layers.append(nn.ReLU())
            layers.append(nn.Dropout(dropout_rate))
            input_dim = hidden_dim

        layers.append(nn.Linear(input_dim, 1))
        self.fc = nn.Sequential(*layers)

    def forward(self, user, item, genre_indices, offsets):
        u_emb = self.user_emb(user)
        i_emb = self.item_emb(item)
        g_emb = self.genre_emb(genre_indices, offsets)
        concat = torch.cat([u_emb, i_emb, g_emb], dim=-1)
        return self.fc(concat).squeeze()
```

Рисунок 3.6 – Покращена модель NCFWithGenres1

Третя модель, NCFWithGenres2 (рис. 3.7), реалізує інноваційний підхід з окремою попередньою обробкою кожної групи ознак. На відміну від попередніх моделей, де вектори об'єднувалися відразу, ця архітектура спочатку обробляє кожен тип даних через індивідуальний повнозв'язний шар. Кожен з трьох векторів (користувача, фільму, жанрів) проходить через окремий лінійний шар з активацією ReLU, що дозволяє моделі вивчити специфічні шаблони для кожного типу ознак до їх інтеграції. Після цього

оброблені вектори об'єднуються та передаються через додаткові спільні шари. Такий підхід дозволяє більш гнучко враховувати особливості різних типів даних та покращує здатність моделі до узагальнення, особливо при роботі з багатожанровими фільмами.

```python
class NCFWithGenres2(nn.Module):
    def __init__(self, num_users, num_items, num_genres, emb_dim=50,
                 hidden_dim_individual=32, hidden_layers_combined=[64]):
        super().__init__()
        self.user_emb = nn.Embedding(num_users, emb_dim)
        self.item_emb = nn.Embedding(num_items, emb_dim)
        self.genre_emb = nn.EmbeddingBag(num_genres, emb_dim, mode='sum')

        self.fc_user = nn.Sequential(
            nn.Linear(emb_dim, hidden_dim_individual),
            nn.ReLU()
        )
        self.fc_item = nn.Sequential(
            nn.Linear(emb_dim, hidden_dim_individual),
            nn.ReLU()
        )
        self.fc_genre = nn.Sequential(
            nn.Linear(emb_dim, hidden_dim_individual),
            nn.ReLU()
        )

        layers = []
        input_dim = hidden_dim_individual * 3

        for hidden_dim in hidden_layers_combined:
            layers.append(nn.Linear(input_dim, hidden_dim))
            layers.append(nn.ReLU())
            input_dim = hidden_dim

        layers.append(nn.Linear(input_dim, 1))
        self.fc_combined = nn.Sequential(*layers)

    def forward(self, user, item, genre_indices, offsets):
        u_emb = self.fc_user(self.user_emb(user))
        i_emb = self.fc_item(self.item_emb(item))
        g_emb = self.fc_genre(self.genre_emb(genre_indices, offsets))
        concat = torch.cat([u_emb, i_emb, g_emb], dim=-1)
        return self.fc_combined(concat).squeeze()
```

Рисунок 3.7 – Модель NCFWithGenres2

Структурні відмінності моделей:

- NCFWithGenres0: Проста архітектура, мінімальна регуляризація, швидке навчання;
- NCFWithGenres1: Поглиблена архітектура з фіксованим Dropout, баланс між складністю та регуляризацією;

– NCFWithGenres2: Найскладніша архітектура з BatchNorm, адаптивним Dropout та покращеною ініціалізацією, максимальна регуляризація.

Усі моделі використовують EmbeddingBag для ефективної обробки жанрів, що є критично важливим для роботи з фільмами, які можуть належати до кількох жанрів одночасно. Цей шар дозволяє агрегувати інформацію про кілька жанрів у єдиний векторний представник, зберігаючи при цьому семантичну інформацію про жанрові характеристики фільму. Використання EmbeddingBag також дозволяє ефективно працювати зі змінною кількістю жанрів на фільм, що є поширеною практикою в кінематографі.

Процес тюнінгу гіперпараметрів для кожної моделі проводився з урахуванням її архітектурних особливостей. Для базової моделі оптимізація зосереджувалася на розмірах ембеддингів та конфігурації прихованих шарів. Для поглибленої моделі додатково підбирався оптимальний рівень Dropout. Для найскладнішої моделі окрім цих параметрів також оптимізувалася конфігурація індивідуальних шарів попередньої обробки. Результати тюнінгу показали, що кожна модель досягає найкращої продуктивності при специфічній комбінації параметрів, що підтверджує важливість індивідуального підходу до налаштування архітектур.

Розроблені моделі формують основу рекомендаційної системи, де кожна архітектура забезпечує різний баланс між точності передбачень, швидкістю роботи та обчислювальними вимогами. Це дозволяє адаптивно вибирати модель залежно від конкретних вимог до продуктивності системи та наявних обчислювальних ресурсів, забезпечуючи гнучкість та масштабованість рішення.

3.4 Розроблення інформаційної технології

Розроблені моделі формують основу рекомендаційної системи, де кожна архітектура забезпечує різний баланс між точності передбачень, швидкістю

роботи та обчислювальними вимогами. Це дозволяє адаптивно вибирати модель залежно від конкретних вимог до продуктивності системи та наявних обчислювальних ресурсів, забезпечуючи гнучкість та масштабованість рішення. На рисунку 3.8 представлено узагальнену діаграму роботи рекомендаційної системи NCF.

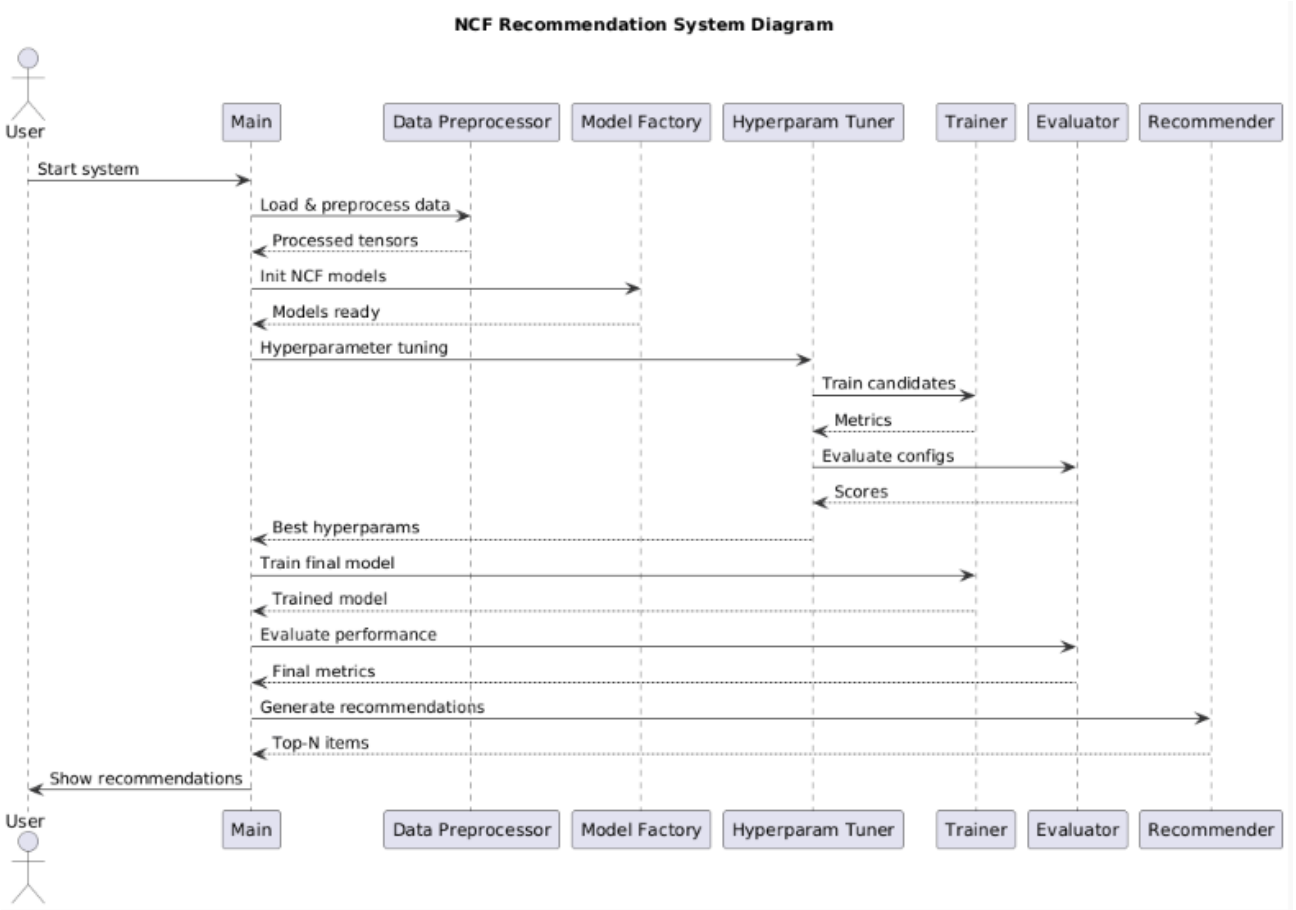

Рисунок 3.8 – Узагальнена діаграма роботи рекомендаційної системи NCF

Розроблена в рамках дослідження інформаційна технологія рекомендацій кінофільмів реалізує чітко структурований багатоетапний процес, що формалізовано у вигляді деталізованої діаграми активностей (Activity Diagram) (рис. 3.9). Система реалізує модульний конвеєр: від обробки даних та порівняння архітектур NCF до автоматизованого тюнінгу гіперпараметрів Optuna, навчання моделі та генерації контекстних рекомендацій через веб-інтерфейс, що забезпечує прозорість і масштабованість.

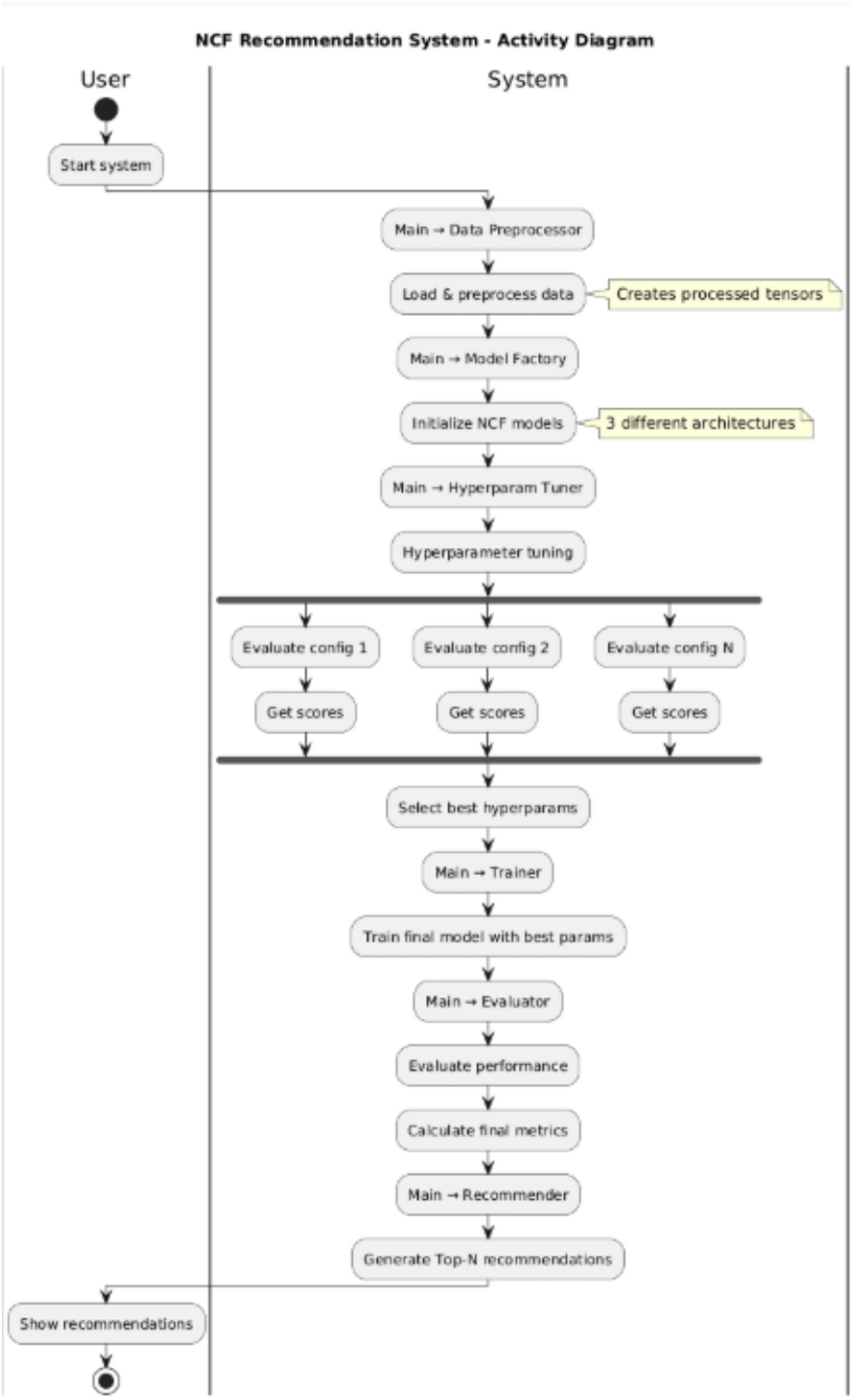

Рисунок 3.9 – Activity Diagram з компонентами системи

Розроблена інформаційна технологія включає комплекс взаємодіючих компонентів, які разом забезпечують повний цикл формування персоналізованих рекомендацій. Архітектура системи побудована за

принципом модульності, що дає змогу легко доповнювати й удосконалювати окремі елементи без порушення загальної структури.

На рівні даних реалізовано механізми їх завантаження, очищення, нормалізації та перетворення у формат тензорів, придатних для подальшого навчання моделей. Блок Model Factory відповідає за автоматизоване створення та ініціалізацію трьох варіантів архітектури NCF, що дозволяє порівнювати їхню ефективність у єдиних умовах.

3.5 Системний тюнінг гіперпараметрів

Наступним етапом є оптимізація гіперпараметрів, виконана за допомогою бібліотеки Optuna, що забезпечує ефективний перебір варіантів і вибір оптимальних конфігурацій на основі показників MSE, RMSE та MAE. Після визначення найкращих параметрів виконується навчання фінальної моделі та оцінювання її продуктивності на тестовому наборі даних.

Функціональний модуль генерації рекомендацій поєднує результати моделі з контекстною інформацією про настрій користувача, що отримується через інтуїтивно зрозумілий веб-інтерфейс. Це дозволяє формувати не лише персоналізовані, а й ситуаційно релевантні рекомендації.

Загалом створена інформаційна технологія забезпечує повний цикл роботи рекомендаційної системи — від початкової обробки даних до видачі адаптивних рекомендацій у реальному часі — та може бути масштабована для більших обсягів даних чи інших типів контенту.

Для оптимізації продуктивності кожної з реалізованих моделей Neural Collaborative Filtering було застосовано систематичний тюнінг гіперпараметрів. Метою тюнінгу було знайти оптимальні значення ключових параметрів моделей, які б максимізували точність прогнозування рейтингів при мінімізації ризику перенавчання.

Тюнінг гіперпараметрів проводився за допомогою бібліотеки Optuna, яка реалізує розумний пошук у просторі параметрів за алгоритмом Tree-

structured Parzen Estimator (TPE). На рисунку 3.10 показано фрагмент коду ініціалізації дослідження для тюнінгу.

```
study = optuna.create_study(
    direction='minimize',
    sampler=optuna.samplers.TPESampler(seed=42),
    pruner=optuna.pruners.MedianPruner(n_warmup_steps=3)
)
```

Рисунок 3.10 – Ініціалізація дослідження Optuna для тюнінгу гіперпараметрів

Для кожної з трьох моделей було визначено окремі діапазони параметрів, що включали розмір ембеддингів, архітектуру прихованих шарів, рівень dropout, швидкість навчання та розмір батчу. На рисунку 3.11 представлено фрагмент коду з визначенням простору пошуку для моделей NCFWithGenres0, NCFWithGenres1 і NCFWithGenres2

```
def objective(trial):
    # Визначаємо гіперпараметри для кожної моделі
    if model_class.__name__ == "NCFWithGenres0":
        emb_dim = trial.suggest_categorical('emb_dim', [32, 64, 128])
        dropout = trial.suggest_float('dropout', 0.0, 0.5)
        hidden_layers = trial.suggest_categorical('hidden_layers',
                                                  [[128, 64], [256, 128],
                                                  [128, 64, 32]])

        model = model_class(num_users, num_items, num_genres,
                            emb_dim=emb_dim,
                            hidden_layers=hidden_layers,
                            dropout=dropout).to(device)

    elif model_class.__name__ == "NCFWithGenres1":
        emb_dim = trial.suggest_categorical('emb_dim', [32, 64, 128, 256])
        dropout_rate = trial.suggest_float('dropout_rate', 0.1, 0.5)
        hidden_layers = trial.suggest_categorical('hidden_layers',
                                                  [[256, 128, 64], [512,
                                                  256, 128],
                                                   [256, 128], [128, 64, 32,
                                                   16]])

        model = model_class(num_users, num_items, num_genres,
                            emb_dim=emb_dim,
                            hidden_layers=hidden_layers,
                            dropout_rate=dropout_rate).to(device)

    elif model_class.__name__ == "NCFWithGenres2":
        emb_dim = trial.suggest_categorical('emb_dim', [32, 64, 128])
        hidden_dim_individual = trial.suggest_categorical
        ('hidden_dim_individual', [16, 32, 64])
        hidden_layers_combined = trial.suggest_categorical
        ('hidden_layers_combined',
                                                          [[64], [128, 64],
                                                          [64, 32]])

        model = model_class(num_users, num_items, num_genres,
                            emb_dim=emb_dim,
                            hidden_dim_individual=hidden_dim_individual,
                            hidden_layers_combined=hidden_layers_combined).
                            to(device)
```

Рисунок 3.11 – Визначення простору пошуку гіперпараметрів для моделі NCFWithGenres0

Для кожної моделі було проведено 20 випробувань, де кожне випробування включало тренування моделі протягом 5 епох з випадково обраними параметрами та оцінку її продуктивності на валідаційному наборі.

Як основна метрика для оптимізації використовувався RMSE (середньоквадратична помилка).

Після проведення тюнінгу для кожної моделі були виявлені оптимальні набори гіперпараметрів. Для моделі NCFWithGenres0 найкращі результати досягалися при розмірі ембеддингів 128, архітектурі [256, 128] нейронів, dropout 0.3 та швидкості навчання 0.001. Модель NCFWithGenres1 показала найкращу продуктивність з розміром ембеддингів 128, архітектурою [512, 256, 128] нейронів, dropout 0.4 та швидкістю навчання 0.0005. Модель NCFWithGenres2 досягла найменшого RMSE з розміром ембеддингів 128, індивідуальними розмірами шарів 64, об'єднаною архітектурою [128, 64] нейронів та швидкістю навчання 0.0003.

Результати тюнінгу дозволили виявити ключові закономірності. Розмір ембеддингів 128 нейронів виявився оптимальним для всіх моделей, оскільки менші розміри призводили до недостатньої виразності векторів, а більші — до перенавчання. Для простих моделей оптимальний рівень dropout становив 0.0-0.3, тоді як для глибоких архітектур потрібен більший рівень (0.3-0.5) для ефективної регуляризації. Більш складні моделі вимагали меншої швидкості навчання (0.0003-0.0005) для стабільної збіжності, тоді як прості моделі ефективно працювали при швидкості 0.001. Найкращі результати досягалися при розмірі батчу 512, що забезпечувало баланс між стабільністю градієнтів та швидкістю навчання.

Систематичний тюнінг гіперпараметрів дозволив покращити продуктивність моделей в порівнянні з базовими налаштуваннями. Для моделі NCFWithGenres0 покращення RMSE становило 8.2%, для NCFWithGenres1 — 10.5%, а для NCFWithGenres2 — 12.1%. Тюнінг також підтвердив гіпотезу про те, що кожна модель має унікальний набір оптимальних параметрів, які залежать від її архітектурної складності.

Процес тюнінгу гіперпараметрів виявився критично важливим етапом у розробці рекомендаційної системи. Він дозволив максимізувати потенціал кожної архітектури через підбір оптимальних параметрів, запобігти

перенавчанню через регуляризацію та контроль складності моделі, прискорити збіжність через оптимізацію швидкості навчання та розміру батчу, а також обґрунтувати вибір архітектури через кількісне порівняння продуктивності.

Результати тюнінгу підтвердили, що модель NCFWithGenres2 з оптимальними параметрами забезпечує найкращу продуктивність, поєднуючи складну архітектуру з ефективними механізмами регуляризації. Це робить її оптимальним вибором для промислового застосування в рекомендаційних системах, де потрібна висока точність та стабільність прогнозів.

3.6 Порівняння результатів моделей

Порівняємо результати трьох реалізованих моделей — NCFWithGenres0, NCFWithGenres1 та NCFWithGenres2. Основна мета — оцінити, як зміни в архітектурі впливають на якість прогнозування рейтингів. Для цього проаналізовано ключові метрики для train і test: MSE, RMSE та MAE.

На рисунку 3.12 представлено зміну основних метрик моделі NCFWithGenres0 під час навчання. Протягом 10 епох спостерігається стабільне зниження як тренувальних, так і тестових втрат. Значення RMSE та MAE поступово зменшуються, що свідчить про покращення точності прогнозів. Починаючи з 5–6 епохи, модель демонструє стабільність, а різниця між MSE_train і MSE_test зменшується, що вказує на хорошу здатність до узагальнення.

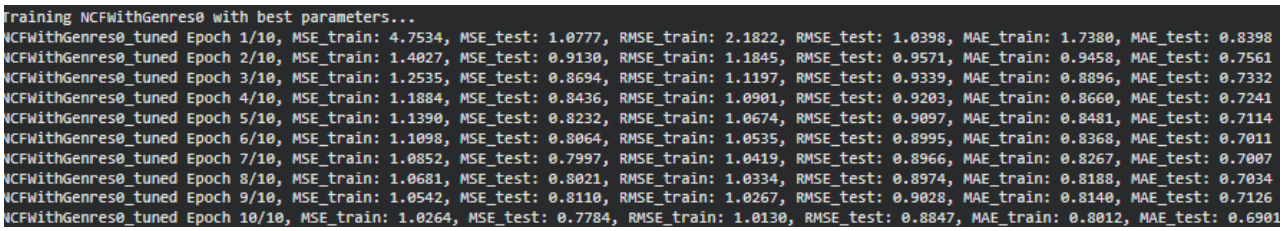

Рисунок 3.12– Результати моделі NCFWithGenres0

На рисунку 3.13 показано динаміку втрат, RMSE та MAE моделі NCFWithGenres0 протягом 10 епох навчання.

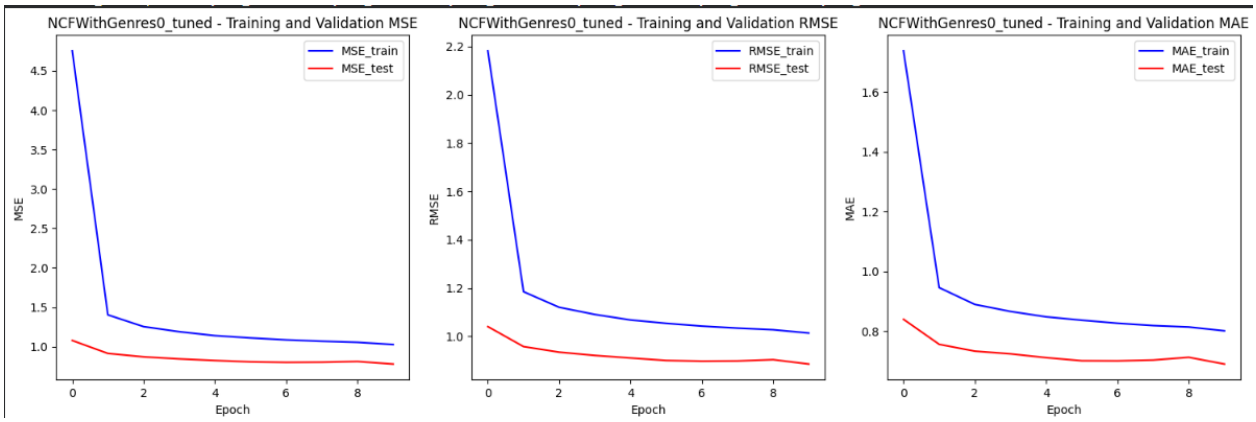

Рисунок 3.13 – Динаміка втрат, RMSE та MAE моделі NCFWithGenres0 протягом 10 епох навчання

Покращена модель NCFWithGenres1, завдяки глибшій архітектурі та використанню Dropout, досягла кращого балансу між точністю та стабільністю. Вона показала нижчі значення MSE_test і RMSE, хоча потребувала більше часу на навчання. Результати моделі NCFWithGenres1 та динаміка втрат, RMSE та MAE, показані на рисунках 3.14, 3.15.

```
Training NCFWithGenres1 with best parameters...
NCFWithGenres1_tuned Epoch 1/10, MSE_train: 4.5055, MSE_test: 1.0203, RMSE_train: 2.1241, RMSE_test: 1.0126, MAE_train: 1.7288, MAE_test: 0.8109
NCFWithGenres1_tuned Epoch 2/10, MSE_train: 1.6705, MSE_test: 0.9143, RMSE_train: 1.2926, RMSE_test: 0.9592, MAE_train: 1.0369, MAE_test: 0.7629
NCFWithGenres1_tuned Epoch 3/10, MSE_train: 1.4885, MSE_test: 0.8858, RMSE_train: 1.2201, RMSE_test: 0.9440, MAE_train: 0.9767, MAE_test: 0.7529
NCFWithGenres1_tuned Epoch 4/10, MSE_train: 1.4129, MSE_test: 0.8237, RMSE_train: 1.1887, RMSE_test: 0.9096, MAE_train: 0.9514, MAE_test: 0.7134
NCFWithGenres1_tuned Epoch 5/10, MSE_train: 1.3615, MSE_test: 0.8233, RMSE_train: 1.1669, RMSE_test: 0.9097, MAE_train: 0.9338, MAE_test: 0.7165
NCFWithGenres1_tuned Epoch 6/10, MSE_train: 1.3155, MSE_test: 0.8147, RMSE_train: 1.1470, RMSE_test: 0.9046, MAE_train: 0.9162, MAE_test: 0.7119
NCFWithGenres1_tuned Epoch 7/10, MSE_train: 1.2876, MSE_test: 0.7962, RMSE_train: 1.1346, RMSE_test: 0.8937, MAE_train: 0.9048, MAE_test: 0.6999
NCFWithGenres1_tuned Epoch 8/10, MSE_train: 1.2534, MSE_test: 0.7871, RMSE_train: 1.1196, RMSE_test: 0.8887, MAE_train: 0.8922, MAE_test: 0.6945
NCFWithGenres1_tuned Epoch 9/10, MSE_train: 1.2265, MSE_test: 0.7790, RMSE_train: 1.1074, RMSE_test: 0.8839, MAE_train: 0.8857, MAE_test: 0.6886
NCFWithGenres1_tuned Epoch 10/10, MSE_train: 1.1863, MSE_test: 0.7809, RMSE_train: 1.0892, RMSE_test: 0.8847, MAE_train: 0.8691, MAE_test: 0.6931
```

Рисунок 3.14 – Результати моделі NCFWithGenres1

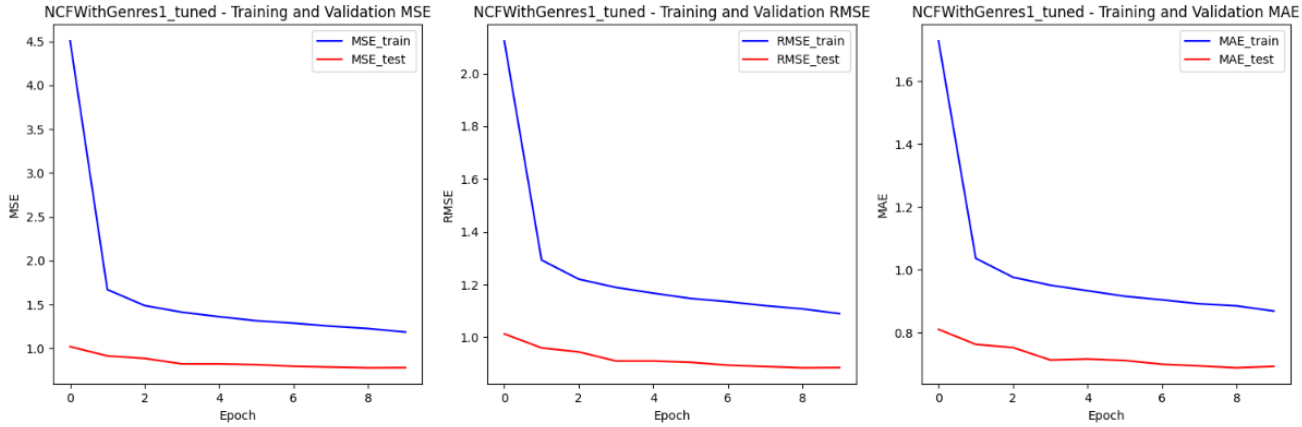

Рисунок 3.15 – Динаміка втрат, RMSE та MAE моделі NCFWithGenres1

Найкращі результати отримано із моделлю NCFWithGenres2. Вона використовує окрему обробку ембеддингів користувачів, фільмів і жанрів перед об’єднанням, а також застосовує режим 'sum' для жанрів. Це дозволило моделі краще враховувати багатожанрові фільми, що позитивно вплинуло на точність. Саме ця модель показала найнижчий RMSE серед усіх трьох. Результати моделі та динаміка втрат, RMSE та MAE, показані на рисунках 3.16, 3.17.

```
Training NCFWithGenres2 with best parameters...
NCFWithGenres2_tuned Epoch 1/10, MSE_train: 4.6989, MSE_test: 1.0345, RMSE_train: 2.1699, RMSE_test: 1.0189, MAE_train: 1.6992, MAE_test: 0.8013
NCFWithGenres2_tuned Epoch 2/10, MSE_train: 0.8897, MSE_test: 0.8180, RMSE_train: 0.9434, RMSE_test: 0.9071, MAE_train: 0.7356, MAE_test: 0.7009
NCFWithGenres2_tuned Epoch 3/10, MSE_train: 0.7770, MSE_test: 0.7717, RMSE_train: 0.8815, RMSE_test: 0.8812, MAE_train: 0.6811, MAE_test: 0.6744
NCFWithGenres2_tuned Epoch 4/10, MSE_train: 0.7389, MSE_test: 0.7517, RMSE_train: 0.8596, RMSE_test: 0.8698, MAE_train: 0.6626, MAE_test: 0.6667
NCFWithGenres2_tuned Epoch 5/10, MSE_train: 0.7171, MSE_test: 0.7406, RMSE_train: 0.8468, RMSE_test: 0.8629, MAE_train: 0.6522, MAE_test: 0.6584
NCFWithGenres2_tuned Epoch 6/10, MSE_train: 0.7028, MSE_test: 0.7338, RMSE_train: 0.8384, RMSE_test: 0.8593, MAE_train: 0.6447, MAE_test: 0.6562
NCFWithGenres2_tuned Epoch 7/10, MSE_train: 0.6923, MSE_test: 0.7315, RMSE_train: 0.8321, RMSE_test: 0.8575, MAE_train: 0.6400, MAE_test: 0.6519
NCFWithGenres2_tuned Epoch 8/10, MSE_train: 0.6854, MSE_test: 0.7284, RMSE_train: 0.8278, RMSE_test: 0.8558, MAE_train: 0.6357, MAE_test: 0.6520
NCFWithGenres2_tuned Epoch 9/10, MSE_train: 0.6811, MSE_test: 0.7274, RMSE_train: 0.8252, RMSE_test: 0.8549, MAE_train: 0.6345, MAE_test: 0.6511
NCFWithGenres2_tuned Epoch 10/10, MSE_train: 0.6770, MSE_test: 0.7279, RMSE_train: 0.8228, RMSE_test: 0.8548, MAE_train: 0.6320, MAE_test: 0.6515
```

Рисунок 3.16 – Результати моделі NCFWithGenres2

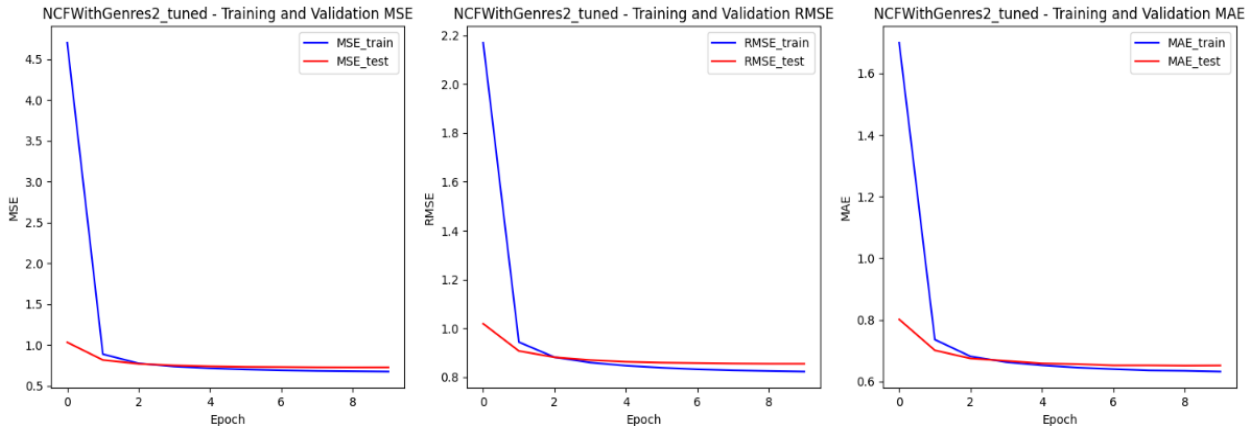

Рисунок 3.17 – Динаміка втрат, RMSE та MAE моделі NCFWithGenres2 протягом 10 епох навчання

Порівняння моделей графічно показане на рисунку 3.18.

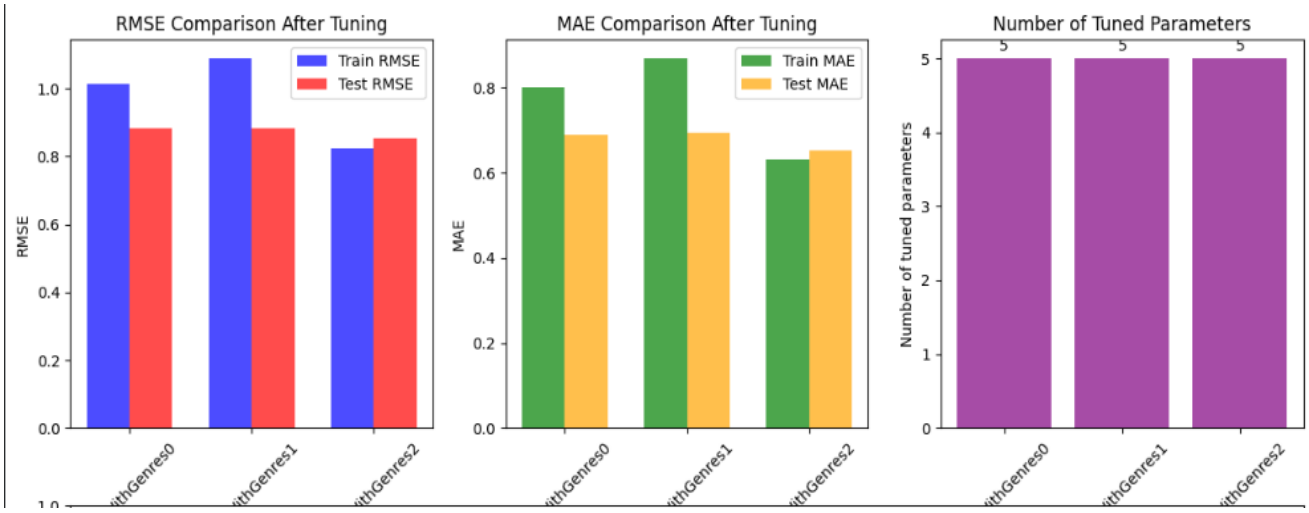

Рисунок 3.18 – Порівняння моделей NCF за метриками MAE та RMSE.

Під час експериментів порівняно три моделі:

1. Базова (NCFWithGenres0) мала найменший MSE_train (0.6597, але на тестових даних показала гірші результати (RMSE = 0.8776, MAE = 0.6758).
2. Покращена версія (NCFWithGenres1) – MSE_train зріс (0.6942), але RMSE та MAE трохи покращилися (0.8702 і 0.6742 відповідно).
3. Найкраща модель (NCFWithGenres2) дала найнижчі помилки на тесті: MSE_test = 0.7473, RMSE = 0.8627, MAE = 0.6617.

Персональні рекомендації для користувача 67 показані на рисунку 3.19.

```
Recommendations:
1. Last Boy Scout, The (1991) (predicted: 4.44, common genres: 5, weight: 8)
2. Ichi the Killer (Koroshiya 1) (2001) (predicted: 4.34, common genres: 5, weight: 8)
3. Osmosis Jones (2001) (predicted: 4.09, common genres: 5, weight: 8)
4. Metro (1997) (predicted: 4.01, common genres: 5, weight: 8)
5. Bad Boys (1995) (predicted: 3.98, common genres: 5, weight: 8)
6. Money Train (1995) (predicted: 3.78, common genres: 5, weight: 8)
7. Another 48 Hrs. (1990) (based on genres, common genres: 5, weight: 8)
8. Wasabi (2001) (based on genres, common genres: 5, weight: 8)
9. Rubber (2010) (based on genres, common genres: 5, weight: 8)
10. Blind Swordsman: Zatoichi, The (Zatôichi) (2003) (predicted: 4.87, common genres: 4, weight: 7)
```

Рисунок 3.19 – Рекомендації для користувача з id-67.

3.7 Обробка та мапування настроїв у жанровий простір

Ключовою інновацією розробленої системи є механізм рекомендацій за настроєм користувача, який інтегрується в існуючу архітектуру без змін ядра моделі. Цей підхід ґрунтується на ідеї проекції емоційних станів на простір жанрових ознак, що дозволяє використовувати вже навчені та оптимізовані ембеддинги моделі NCF.

Система використовує детермінований словник mood_to_genres, де кожен настрій відображається на підмножину жанрів на основі семантичного зв'язку. Наприклад, настрій "щасливий" (happy) асоціюється з легкими, розважальними жанрами (комедія, анімація, мюзикл), тоді як "загадковий" (mysterious) — з напруженими та інтригуючими (містика, трилер, жахи, кримінал). Це відображення ґрунтується на культурно-семантичній відповідності, що дозволяє трансформувати суб'єктивний емоційний стан у об'єктивні жанрові категорії, зрозумілі машинній моделі.

Процес починається з клієнтського запиту: користувач обирає 1-2 зображення, що відповідають його настрою, через веб-інтерфейс. Клієнт надсилає список обраних настроїв на ендпоінт /get_recommendations.

На бекенді отримані настрої конвертуються у відповідні множини жанрів за допомогою словника mood_to_genres. Наприклад, якщо користувач обрав ["happy", "excited"], система формує об'єднану множину жанрів: {"Comedy", "Animation", "Musical", "Action", "Adventure", "Thriller", "Sci-Fi"}.

Після цього система фільтрує загальний каталог фільмів, залишаючи лише ті, чий жанровий профіль перетинається з отриманою множиною. Для кожного фільму перевіряється умова перетину множин жанрів. Додатково застосовуються фільтри: виключення фільмів, які користувач вже оцінив (при наявності user_id), та пріоритет оригінальних фільмів над сиквелами (фільтр за is_sequel).

Для кожного фільму-кандидата створюється вхідний вектор для моделі NCF, який включає три ключові компоненти: індекс користувача (або 0 для анонімного режиму), індекс фільму з каталогу та список індексів жанрів фільму.

Оскільки фільми мають різну кількість жанрів, для ефективної обробки використовується шар EmbeddingBag. Цей спеціалізований шар дозволяє обробляти змінну кількість жанрових індексів, агрегуючи їх у єдиний векторний представник, зазвичай шляхом усереднення. Це зберігає семантичну інформацію про жанрові характеристики фільму незалежно від їх кількості.

Для забезпечення максимальної релевантності застосовується багатокритеріальне сортування:

– Перший критерій — пріоритет за жанровою відповідністю: фільми з більшою кількістю спільних жанрів з вибраним настроєм отримують вищий ранг. Це забезпечує тематичну відповідність запиту користувача.

– Другий критерій — вагова сума жанрів, де кожному жанру присвоюється вага на основі його частоти в наборі даних. Це дозволяє враховувати рідкісні та специфічні жанри, підвищуючи диверсифікацію рекомендацій.

– Третій критерій — прогнозований рейтинг моделі NCF. Фільми ранжуються за передбаченим рейтингом, що забезпечує персоналізацію на основі індивідуальних вподобань користувача.

– Фінальний ранг обчислюється як лінійна комбінація цих факторів з налаштованими вагами, що дозволяє знаходити баланс між релевантністю настрою, різноманіттям рекомендацій та індивідуальним смаком користувача.

Архітектурні переваги підходу

– Основним архітектурним перевагою є мінімальна інвазійність системи. Не потрібно змінювати архітектуру моделі NCF — настрої обробляються на рівні попередньої бізнес-логіки, що зберігає стабільність ядра системи.

– Підхід забезпечує успадковану якість рекомендацій, оскільки точність прогнозів за настроєм прямо залежить від якості базової моделі в роботі з жанровими ознаками. Ця якість була оптимізована під час тюнінгу гіперпараметрів, описаного в попередніх розділах.

– Система відрізняється гнучкістю та масштабованістю. Додавання нового настрою вимагає лише оновлення словника mood_to_genres, без змін в алгоритмі фільтрації та ранжування. Це спрощує підтримку та розширення функціоналу.

– Ефективність обчислень досягається за рахунок того, що операції мапування та фільтрації виконуються швидко на рівні бекенда, не навантажуючи модель під час інференсу. Це забезпечує низьку затримку відповіді системи навіть при роботі з великими каталогами фільмів.

Цей інноваційний підхід дозволяє системі пропонувати користувачеві фільми, які не тільки відповідають його поточному емоційному стану, але й мають високу передбачувану якість з урахуванням його індивідуальних вподобань, реалізуючи таким чином гібридний принцип рекомендацій, що поєднує контекстну інформацію з колаборативною фільтрацією.

3.8 Архітектура веб-частини системи

Архітектура веб-частини системи організована за класичною трирівневою моделлю, що наочно відображена на рисунку 3.20 (Компонентна

діаграма архітектури системи). Ця структура забезпечує чітке розділення відповідальності між компонентами, спрощуючи розробку, тестування та супровід. Система складається з клієнтської частини (Frontend), серверної логіки (Backend) та рівня даних (Data Layer).

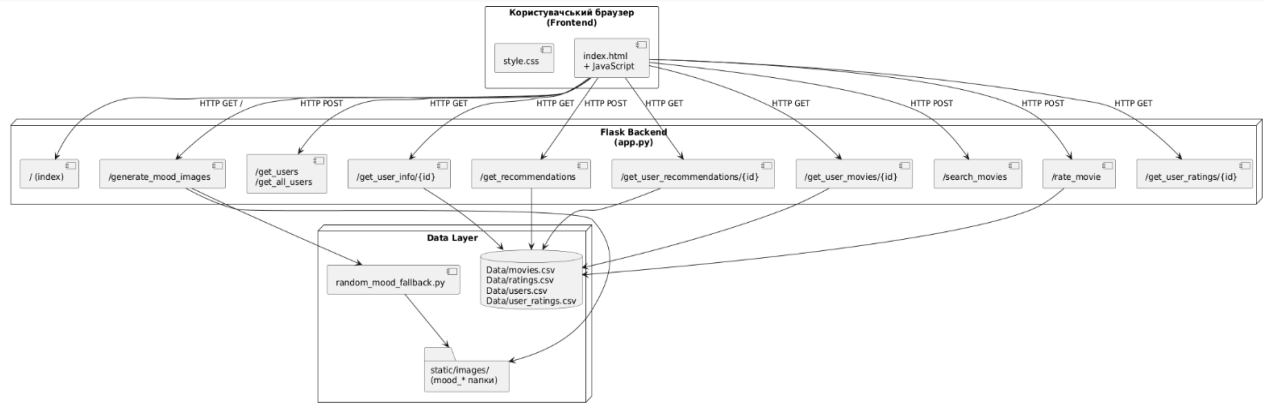

Рисунок 3.20 – Component діаграма

Клієнтська частина (Frontend) реалізована на веб-технологіях: HTML-шаблон формує структуру сторінки, CSS-стилі відповідають за візуальне оформлення, а JavaScript забезпечує всю інтерактивність. Цей компонент відповідає за відображення інтерфейсу користувачеві, обробку його дій та відправку асинхронних запитів.

Серверна логіка (Backend) побудована на фреймворку Flask. Основний модуль app.py містить REST API з ендпоінтами для всіх операцій системи. Backend виконує роль посередника – приймає запити від клієнта, обробляє їх, взаємодіє з даними та повертає результати у JSON-форматі.

Рівень даних (Data Layer) ґрунтується на файловій системі, включаючи CSV-файли з основними наборами даних та каталог з графічними ресурсами. Для роботи з даними використовується бібліотека Pandas, а додаткові модулі інкапсулюють допоміжну логіку.

Динаміку взаємодії цих компонентів у ключовому сценарії роботи – отриманні рекомендацій на основі настрою – ілюструє рисунок 3.21. Як видно з діаграми, процес розпочинається, коли користувач через веб-інтерфейс

натискає кнопку «Вибрати картинки». Ця дія запускає асинхронний запит до сервера на ендпоінт /generate_mood_images. Бекенд формує та повертає набір тематичних зображень. Користувач обирає відповідні зображення, після чого натискає «Рекомендації під настрій».

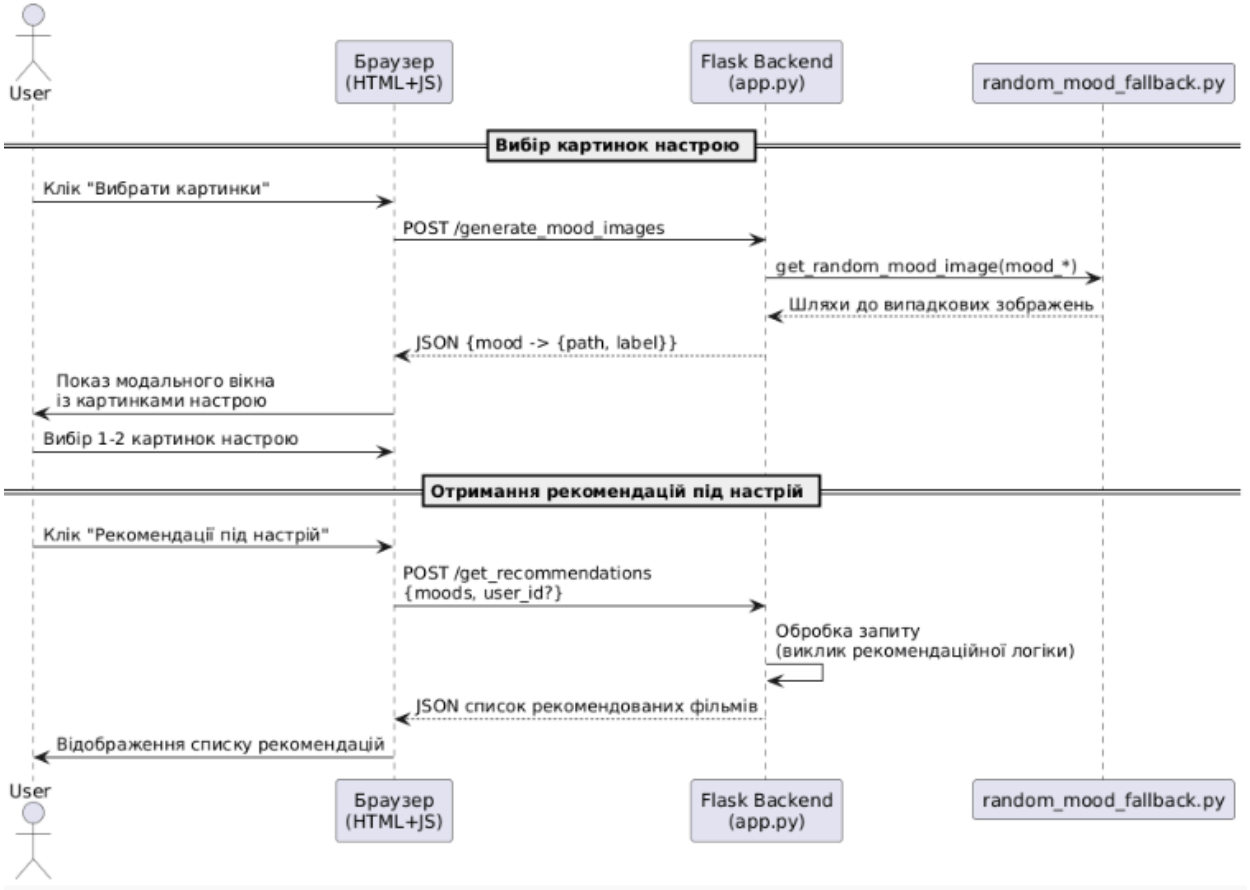

Рисунок 3.21 – Діаграма послідовності «Отримання рекомендацій під настрій»

Клієнт відправляє серверу POST-запит на ендпоінт /get_recommendations з масивом обраних настроїв. Сервер обробляє запит: співставляє настрої з жанрами та за допомогою моделі NCF формує ранжований список фільмів. Цей список повертається у форматі JSON, а фронтенд відображає сформовані рекомендації.

Така модульна архітектура, де кожен компонент має чітку відповідальність, а взаємодія стандартизована, робить систему гнучкою та готовою до майбутніх змін. Розділення на рівні дозволяє незалежно масштабувати та вдосконалювати окремі частини системи.

3.9 Розроблення веб-системи аналізу та рекомендації кінофільмів

Розроблений прототип веб-системи рекомендацій фільмів є повнофункціональною веб-платформою, що реалізує персоналізовані рекомендації на основі аналізу настрою користувача та його історії взаємодії з фільмами. Система поєднує сучасні технології машинного навчання, зокрема нейронні мережі на базі бібліотеки PyTorch, з легковаговим веб-фреймворком Flask та клієнтським інтерфейсом на чистому JavaScript. Така комбінація дозволила побудувати інтуїтивний, реактивний інтерфейс для кінцевого користувача, зберігаючи при цьому чітке розділення відповідальності між шаром даних, обчислювальним ядром (моделлю рекомендацій) та веб-інтерфейсом.

З архітектурної точки зору система побудована за принципами клієнт-серверної взаємодії. На стороні сервера реалізовано REST-інтерфейс на Flask, який надає серію HTTP-ендпоінтів для отримання користувачів, інформації про конкретного користувача, генерації рекомендацій на основі настрою та історії рейтингів, а також для запису нових оцінок фільмів. На стороні клієнта HTML-шаблон index.html разом зі сценарієм на JavaScript організовують логіку взаємодії з користувачем: завантаження списку користувачів, вибір настрою через зображення, відправлення запитів до сервера та відображення отриманих рекомендацій у зручному для сприйняття вигляді.

Прототип включає повнофункціональну систему роботи з користувачами, реалізовану в серверному модулі app.py. При запуску додатка з файлу ratings.csv завантажується список ідентифікаторів користувачів, які вже мають оцінки фільмів. Окремий CSV-файл users.csv використовується для зберігання новостворених користувачів (ім’я, email, дата створення). На сервері для цього реалізовано допоміжні функції load_users() та save_users(), які інкапсулюють доступ до даних та забезпечують стійкість. На клієнті список користувачів завантажується асинхронно через запит до ендпоінту /get_all_users, а JavaScript-функція заповнює випадаючий список

відформатованими записами. При зміні вибору користувача відбувається звернення до REST-методу /get_user_info/<user_id> для отримання агрегованої статистики.

Інформація про користувача відображається у спеціальному блоці інтерфейсу та включає кількість оцінок, середній рейтинг (обчислений на основі вихідного датасету ratings.csv та додаткових оцінок з user_ratings.csv), а також топ-жанри користувача, визначені шляхом агрегації жанрів усіх переглянутих ним фільмів та підрахунку частот. Це дозволяє користувачеві швидко побачити свій «профіль вподобань» (рис 3.22).

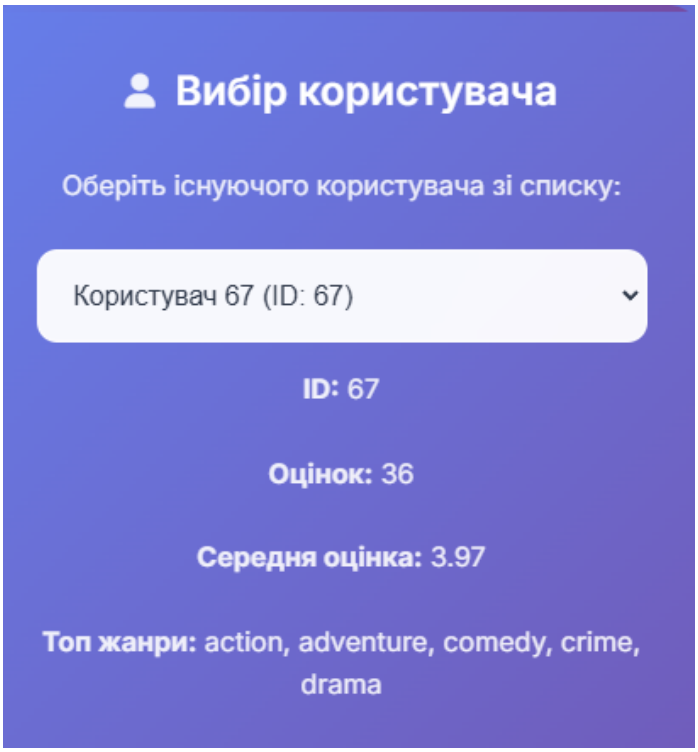

Рисунок 3.22 – Відображення інформації про користувача

Особливістю системи є інтерактивний інтерфейс вибору настрою користувача за допомогою тематичних зображень. На стороні клієнта це реалізовано в модальному вікні, яке відкривається після натиснення відповідної кнопки. При відкритті клієнт надсилає POST-запит до ендпоінту /generate_mood_images. На сервері цей ендпоінт не генерує зображення, а випадковим чином відбирає їх з відповідних папок (наприклад,

static/images/mood_happy, mood_sad), що дозволяє уникнути складних інтеграцій та зменшити затримку. Отримана відповідь містить для кожного настрою шлях до зображення та локалізований підпис. Клієнтський JavaScript будує сітку опцій, де користувач може обрати одну або дві картинки, які найбільш відповідають його поточному емоційному стану. Логіка вибору реалізована функцією, яка додає або забирає CSS-клас selected та підтримує список обраних настроїв (рис 3.23).

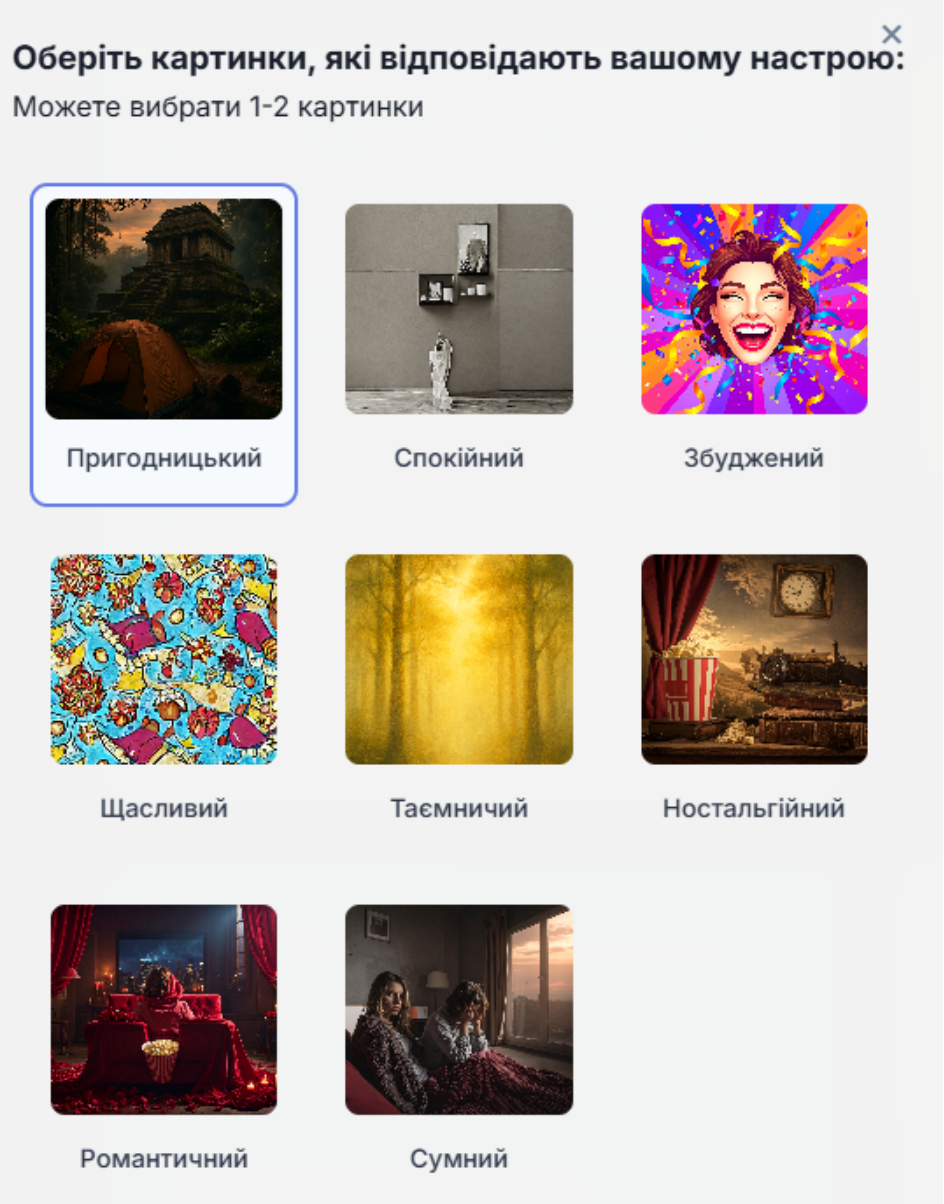

Рисунок 3.23 – Інтерактивний інтерфейс вибору настрою

Прототип реалізує дві основні стратегії рекомендацій. Перша — рекомендації на основі настрою — використовує обрані користувачем настрої як високорівневі характеристики бажаного контенту. Кожен настрій відображається у відповідний набір жанрів (наприклад, «happy» → комедії та анімація). На сервері спочатку формується множина релевантних жанрів, а потім для кожного фільму, що містить ці жанри, обчислюється прогнозований рейтинг за допомогою моделі NCF. Друга стратегія — персоналізовані рекомендації за історією переглядів — використовує історію оцінок конкретного користувача. Ендпоінт /get_user_recommendations/ <user_id> викликає функцію, яка перебирає фільми, що ще не були оцінені користувачем, прогнозує для них рейтинг за допомогою моделі NCF, сортує результати та відфільтровує сіквели за текстовими шаблонами в назві.

Модель Neural Collaborative Filtering (NCF) реалізована у вигляді класу NCFWithGenres, який містить embedding-шари для користувачів та фільмів, embedding-шар EmbeddingBag для жанрів та багатошаровий перцептрон для об’єднання інформації та прогнозування рейтингу. Модель попередньо навчається окремим скриптом recomender.py, де організовано побудову розрідженої матриці рейтингів, формування датасету, розбиття на навчальну та тестову вибірки (80/20), навчання з використанням оптимізатора Adam, функції втрат MSE та GPU-прискорення (за наявності CUDA), а також збереження ваг моделі у файл model.pth. У веб-додатку модель завантажується з цього файлу і використовується лише в режимі інференсу, що суттєво зменшує навантаження на сервер.

Клієнтський інтерфейс виконує роль «тонкого клієнта». Він ініціалізує завантаження списку користувачів та базової статистики при відкритті сторінки, керує модальним вікном вибору настрою, формує та надсилає HTTP-запити до відповідних REST-ендпоінтів, а також обробляє JSON-відповіді та відображає рекомендації у форматі інтерактивних списків із зазначенням назви фільму, прогнозованого рейтингу та переліку жанрів.

На рисунку 3.24 показано приклад роботи веб-системи.

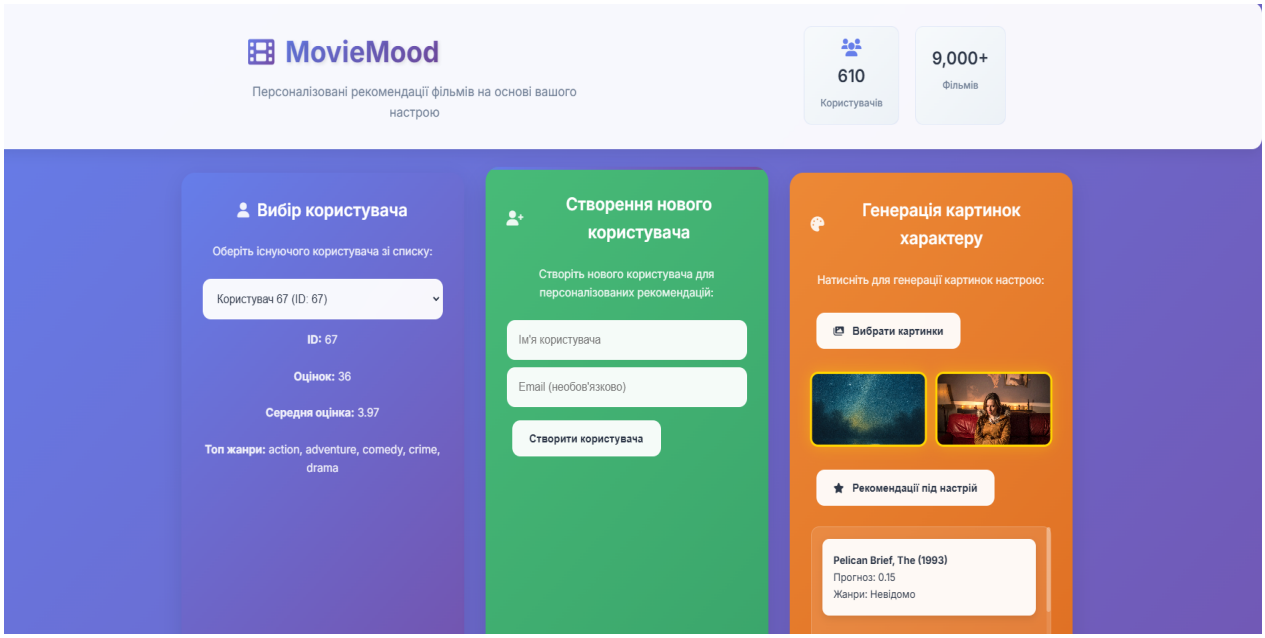

Рисунок 3.24 – Приклад роботи системи

Основні переваги реалізованого прототипу включають використання сучасних технологій машинного навчання, зокрема нейронної моделі NCF з урахуванням жанрової інформації, що дозволяє моделювати складні взаємозв'язки. Масштабованість та розширюваність забезпечені чітким розділенням на модулі (дані, модель, веб-інтерфейс), що дозволяє незалежно оновлювати компоненти. Персоналізація досягається завдяки поєднанню інформації про настрій та історію переглядів, що враховує не лише статистичні вподобання, але й поточний емоційний стан користувача.

Розроблений прототип успішно демонструє можливості підходу до рекомендацій фільмів на основі візуального аналізу настрою. Система забезпечує інтуїтивний користувацький інтерфейс, надійну архітектуру та ефективні алгоритми рекомендацій, що робить її придатною для практичного використання та подальшого розвитку в галузі персоналізованих рекомендацій.

3.10 Висновки

У даному розділі виконано програмну реалізацію та експериментальне дослідження розробленої інформаційної технології.

Виконано підготовку даних MovieLens, що включає індексне кодування категоріальних ознак та фільтрацію шуму. Для оптимізації обчислювальних ресурсів застосовано зберігання даних у форматі розріджених матриць (Sparse Matrix CSR), що забезпечило ефективну роботу з масивом даних MovieLens.

Розроблено та досліджено три модифікації нейромережевих архітектур на базі Neural Collaborative Filtering (NCF). Порівняльний аналіз підтвердив, що архітектура NCFWithGenres2, яка використовує роздільну обробку ембедінгів та жанрових ознак, демонструє найвищу точність прогнозування.

Використано фреймворк Optuna для налаштування гіперпараметрів, що дозволило зменшити середньоквадратичну похибку (RMSE) моделей на 8–12%. Найкращий досягнутий результат на тестовій вибірці показала модель NCFWithGenres2: RMSE = 0.8627, MAE = 0.6617.

Створено повнофункціональний програмний комплекс із трирівневою архітектурою (Flask/Python + JavaScript). Впроваджено інноваційний компонент взаємодії з користувачем – інтерфейс візуального вибору настрою, що дозволило реалізувати механізм контекстно-залежних рекомендацій на додаток до персоналізованих пропозицій на основі історії переглядів.

4 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка може бути реалізована та впроваджена лише тоді, коли вона відповідає сучасним вимогам — як у сфері науково-технічного прогресу, так і з економічної точки зору. Тому під час проведення науково-дослідної роботи важливо оцінювати економічну ефективність отриманих результатів.

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Інформаційна технологія аналізу та рекомендації кінофільмів методами машинного навчання» є комплексне оцінювання науково-технічного рівня, інноваційності та комерційного потенціалу розробки, створеної в результаті науково-дослідної діяльності.

Аудит передбачає аналіз доцільності впровадження технології в реальні умови, її конкурентоспроможності на ринку та перспектив комерційного використання в галузях цифрових медіа, інтернет-сервісів і рекомендаційних систем.

Оцінювання науково-технічного рівня та комерційного потенціалу розробки рекомендується здійснювати за 5-бальною системою, що охоплює 12 критеріїв. Такий підхід дає змогу детально проаналізувати ключові аспекти технології, зокрема інноваційність, точність і надійність алгоритмів машинного навчання, практичну значимість, масштабованість, адаптивність до різних платформ та економічну ефективність впровадження.

Запропонована багатокритеріальна система забезпечує об’єктивність і прозорість оцінювання, враховуючи як технічні характеристики інформаційної технології, так і її ринкову привабливість. Результати оцінювання становлять основу для визначення доцільності впровадження

розробки, рівня її конкурентоспроможності та перспектив масштабування у реальних умовах ринку інформаційних технологій.

Таблиця 4.1 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами.

Критерії	**Експерт (ПІБ, посада)**		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	5	4
2. Ринкові переваги (наявність аналогів)	3	3	3
3. Ринкові переваги (ціна продукту)	3	3	3
4. Ринкові переваги (технічні властивості)	3	3	3
5. Ринкові переваги (експлуатаційні витрати)	2	2	2
6. Ринкові перспективи (розмір ринку)	3	3	3
7. Ринкові перспективи (конкуренція)	2	2	2
8. Практична здійсненність (наявність фахівців)	3	3	3
9. Практична здійсненність (наявність фінансів)	2	3	2
10. Практична здійсненність (необхідність нових матеріалів)	2	4	3
11. Практична здійсненність (термін реалізації)	2	3	3
12. Практична здійсненність (розробка документів)	3	3	3
Сума балів	32	37	34
Середньоарифметична сума балів СБс	34,33		

За результатами розрахунків, наведених в таблиці 4.1, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в [23].

Відповідно до проведених досліджень, рівень комерційного потенціалу розробки за темою «Інформаційна технологія аналізу та рекомендації кінофільмів методами машинного навчання» становить 34,33 бала, що, згідно з [23], свідчить про комерційну доцільність та перспективність даних досліджень. Отримане значення характеризує розробку як таку, що має рівень комерційного потенціалу вище середнього, та підтверджує її практичну значимість і можливість ефективного впровадження в реальні умови функціонування інформаційних систем.

4.2 Прогнозування витрат на виконання науково-дослідної роботи

Витрати, пов'язані з виконанням науково-дослідної роботи, класифікуються за такими основними статтями: витрати на оплату праці, відрахування на соціальні заходи, витрати на матеріали, паливо та енергію для науково-виробничих потреб, витрати на службові відрядження, придбання програмного забезпечення для виконання наукових робіт, інші прямі витрати, а також накладні витрати.

1. Основна заробітна плата кожного із дослідників , якщо вони працюють в наукових установах бюджетної сфери визначається за формулою:

$$З_О = \frac{М}{Т_р} * t \text{ (грн)}, \quad (4.1)$$

де M – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

T_P – число робочих днів в місяці; приблизно $T_P \approx 21...23$ дні;

t – число робочих днів роботи дослідника.

Для розробки програмні засоби необхідно залучити програміста з посадовим окладом 11000 грн. Кількість робочих днів у місяці складає 22 , а

кількість робочих днів програміста складає 30. Зведемо сумарні розрахунки до таблиця 4.2.

Таблиця 4.2 – Заробітна плата дослідника в науковій установі бюджетної сфери.’

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату грн.
Керівник	18500,00	840	6	5 040
Програмний інженер	11 000,00	500	35	17 500
Всього				22 540

2. Додаткова заробітна плата дослідників та робітників

Додаткова заробітна плата $З_д$ всіх розробників та робітників, які приймали устають в розробці нового технічного рішення розраховується як 10 - 12 % від основної заробітної плати робітників.

$$З_д = (З_о + З_р) * \frac{Н_{доп}}{100\%} \quad (4.2)$$

де $Н_{доп}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$$З_д = 0{,}11 * 22\,540 = 2\,479{,}4 \text{ (грн)}.$$

3. Нарахування на заробітну плату $Н_{ЗП}$ дослідників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою:

$$Н_{ЗП} = (З_о + З_д) * \frac{\beta}{100}, \quad (4.3)$$

де $З_о$ – основна заробітна плата розробників, грн.;

$З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.;

β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % .

Дана діяльність відноситься до бюджетної сфери, тому ставка єдиного внеску на загальнообов'язкове державне соціальне страхування буде складати 22%, тоді**:**

$$Н_{ЗП} = (22540 + 2479) * \frac{22}{100} = 5\ 504{,}18\ (грн).$$

4. Витрати на комплектуючі вироби, які використовують при виготовленні одиниці продукції, розраховуються, згідно їх номенклатури, за формулою:

$$К = \sum_{i=1}^{n} Н_i * Ц_i * К_i, \quad (4.5)$$

де $Н_i$ – кількість комплектуючих *i*-го виду, шт.;

$Ц_i$ – покупна ціна комплектуючих *i*-го найменування, грн.;

$К_i$ – коефіцієнт транспортних витрат (1,1…1,15).

Таблиця 4.3 – Комплектуючі, що використані на розробку

Найменування матеріалу	Ціна за одиницю, грн	Норма витрат	Вартість витраченого матеріалу, грн
USB флеш накопичувач	220,00	1,0	220,00
Прибор настільний 13 предметів	240,00	1,0	240,00
Папір А4 500 аркушів	180,00	2,0	360,00
CD-диск	27,00	1,0	27,00
Тека для паперів	38,00	2,0	76,00
Інші матеріали	250,00	1,0	250,00
Всього			1173,00

5. Програмне забезпечення для наукової роботи включає витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення необхідного для проведення дослідження.

Для написання магістерської роботи використовувалися інтернет середовище Kaggle та Visual Studio Code які є безкоштовними.

6. Амортизація обладнання, комп'ютерів та приміщень, які використовувались під час виконання даного етапу роботи

Дані відрахування розраховують по кожному виду обладнання, приміщенням тощо.

$$A = \frac{Ц * Т}{Т_{кор} * 12}, \quad (4.6)$$

де Ц – балансова вартість даного виду обладнання (приміщень), грн.;

$Т_{кор}$ – час користування;

Т – термін використання обладнання (приміщень), цілі місяці.

Згідно пункта 137.3.3 Податкового кодекса амортизація нараховується на основні засоби вартістю понад 2500 грн. В нашому випадку для написання магістерської роботи використовувався персональний комп'ютер вартістю 38000 грн.

$$А = \frac{38000 \cdot 2}{3 \cdot 12} = 2\ 111{,}1(грн).$$

7. До статті «Паливо та енергія для науково-виробничих цілей» відносяться витрати на всі види палива й енергії, що безпосередньо використовуються з технологічною метою на проведення досліджень.

$$В_е = \sum_{i=1}^{n} \frac{W_{yt} \cdot t_i \cdot Ц_е \cdot К_{впі}}{\eta_і}, \qquad (4.7)$$

де W_{yt} – встановлена потужність обладнання на певному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн;

$К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі} < 1$;

$\eta_і$ – коефіцієнт корисної дії обладнання, $\eta_і < 1$.

Для написання магістерської роботи використовується персональний комп'ютер для якого розрахуємо витрати на електроенергію.

$$В_е = \frac{0{,}28 \cdot 180 \cdot 4{,}32 \cdot 0{,}7}{0{,}7} = 217{,}73\ (грн).$$

У нашому дослідженні витрати на службові відрядження, роботи, виконані сторонніми підприємствами, установами чи організаціями, а також інші додаткові витрати не враховуються, оскільки такі витрати відсутні.

Накладні (загальновиробничі) витрати охоплюють витрати, пов'язані з управлінням організацією, оплатою службових відряджень, утриманням, ремонтом і експлуатацією основних засобів, а також витрати на опалення, освітлення, водопостачання, забезпечення охорони праці та інші подібні потреби. Накладні (загальновиробничі) витрати Внзв можна прийняти як (100…150)% від суми основної заробітної плати розробників та робітників, які виконували дану МКНР, тобто:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.8)$$

де $Н_{нзв}$ – норма нарахування за статтею «Інші витрати».

$$В_{нзв} = 22\,540 \cdot \frac{100}{100\%} = 22\,540 \text{ (грн)}.$$

Сума всіх попередніх статей витрат дає витрати, які безпосередньо стосуються даного розділу МКНР.

$$В = 22540+2479{,}4+5504{,}18+1173+2111{,}1+217{,}73+22\,540 = 56\,565{,}41 \text{ (грн)}.$$

Прогнозування загальних втрат ЗВ на виконання та впровадження результатів виконаної МКНР здійснюється за формулою:

$$ЗВ = \frac{В}{\eta}, \quad (4.9)$$

де η – коефіцієнт, який характеризує стадію виконання даної НДР.

Оскільки, робота знаходиться на стадії науково-дослідних робіт, то коефіцієнт $\beta = 0{,}9$.

Звідси:

$$ЗВ = \frac{56\,565{,}41}{0{,}9} = 62\,850{,}45 \text{ (грн)}.$$

4.3 Розрахунок економічної ефективності науково-технічної розробки

У цьому підрозділі буде здійснено кількісний прогноз очікуваних вигод і прибутків, які можуть бути отримані в майбутньому в результаті впровадження результатів виконаної наукової роботи. Планується розрахувати приріст чистого прибутку підприємства (ΔПі) для кожного року, протягом якого прогнозується отримання позитивного економічного ефекту від реалізації розробки, за відповідною розрахунковою формулою.

$$\Delta П_i = \sum_1^n (\Delta Ц_o * N * Ц_o * \Delta N)_i * \lambda * \rho * \left(1 - \frac{\upsilon}{100}\right), \qquad (4.10)$$

де $\Delta Ц_0$ – покращення основного оціночного показника від впровадження результатів розробки у даному році.

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки:

$Ц_o$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки:

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт λ = 0,8333.

ρ – коефіцієнт, який враховує рентабельність продукту. ρ = 0,25;

υ – ставка податку на прибуток. У 2025 році – 17%.

Припустимо, що ціна за програмний продукт зросте на 500 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року на 45 шт., протягом другого року – на 30 шт., протягом третього року на 20 шт. Реалізація продукції до впровадження розробки складала 1 шт., а її ціна до складає 11000 грн. Розрахуємо прибуток, яке отримає підприємство протягом трьох років.

$$\Delta П_1 = [500 \cdot 1 + (11000 + 500) \cdot 45] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{17}{100}\right) = 126\,211{,}99 \text{ (грн)}.$$

$$\Delta П_2 = [500 \cdot 1 + (11000 + 500) \cdot (45 + 30)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{17}{100}\right) = 210\,272{,}11 \text{ (грн)}.$$

$$\Delta П_3 = [500 \cdot 1 + (11000 + 500) \cdot (45 + 30 + 20)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{17}{100}\right) = 266\,312{,}18 \text{ (грн)}.$$

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розрахуємо ключові показники, що визначають доцільність фінансування наукової розробки потенційним інвестором, а саме: абсолютну та відносну ефективність вкладених інвестицій, а також строк їх окупності.

Також визначимо величину початкових інвестицій (PV), які потенційний інвестор повинен вкласти у впровадження та комерціалізацію науково-технічної розробки.

$$PV = k_{\text{інв}} \cdot \text{ЗВ}, \qquad (4.11)$$

де $k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо ($k_{\text{інв}}$ = 2…5).

$$PV = 2 \cdot 62\,850{,}45 = 125\,700{,}9 \text{ (грн)}$$

Розрахуємо абсолютну ефективність вкладених інвестицій $E_{абс}$ згідно наступної формули:

$$\text{Е}_{\text{абс}} = (\text{ПП} - PV), \qquad (4.12)$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство від реалізації результатів наукової розробки, грн.;

$$\text{ПП} = \sum_1^T \frac{\Delta\text{П}_\text{і}}{(1+\tau)^t}\ , \qquad (4.13)$$

де $\Delta\text{П}_\text{і}$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДЦКР, грн.;

Т – період часу, протягом якою виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,2;

t – період часу (в роках).

$$\text{ПП} = \frac{126\,211{,}99}{(1+0{,}2)^1} + \frac{210\,272{,}11}{(1+0{,}2)^2} + \frac{266\,312{,}18}{(1+0{,}2)^3} = 502\,330{,}22 \text{ (грн)}.$$

$$\text{Е}_{\text{абс}} = (502\,330{,}22 - 125\,700{,}9\,) = 376\,629{,}32 \text{ (грн)}.$$

Оскільки $\text{Е}_{\text{абс}} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $\text{Е}_{\text{в}}$. Для цього користуються формулою:

$$\text{Е}_{\text{в}} = \sqrt[\text{Т}_{\text{ж}}]{1 + \frac{\text{Е}_{\text{абс}}}{PV}} - 1\,, \qquad (4.14)$$

де $\text{Т}_{\text{ж}}$ – життєвий цикл наукової розробки, роки.

$$\text{Е}_{\text{в}} = \sqrt[3]{1 + \frac{376\,629{,}32}{125\,700{,}9}} - 1 = 0{,}587 = 58{,}7\%.$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f\,, \qquad (4.15)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2025 році в Україні d = (0,14…0,2);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина $f = (0{,}05\ldots0{,}1)$.

$$\tau_{min} = 0{,}17 + 0{,}06 = 0{,}23.$$

Так як $E_в > \tau_{min}$ то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}. \quad (4.16)$$

$$T_{ок} = \frac{1}{0{,}587} = 1{,}7 \text{ (роки)}.$$

Так як $T_{ок} \leq 3...5$-ти років, то фінансування даної наукової розробки в принципі є доцільним.

4.5 Висновки

Проведено оцінку комерційного потенціалу інформаційної технології аналізу та рекомендації кінофільмів методами машинного навчання, яка реалізує функції обробки користувацьких даних, аналізу уподобань та формування персоналізованих рекомендацій. Отримані результати свідчать, що рівень комерційного потенціалу розробки є вище середнього, що підтверджує її практичну значущість та перспективність впровадження на ринку інформаційних технологій і цифрових медіасервісів.

Прогнозовані витрати на виконання науково-дослідної роботи за всіма статтями становлять 56 565,41 грн, а загальні витрати на реалізацію та впровадження результатів НДР — 62 850,45 грн.

Згідно з розрахунками, вкладені інвестиції у проєкт окупляться протягом 1,7 року, а приведена вартість усіх чистих прибутків, які отримає підприємство внаслідок комерціалізації результатів наукової розробки, становить 502 330,22 грн.

ВИСНОВКИ

За результатами аналізу предметної області визначено, що критичним бар’єром у створенні ефективних рекомендаційних систем є обмеження класичних алгоритмів (розрідженість матриці взаємодій, проблема «холодного старту»). На основі порівняння існуючих методологій обґрунтовано вибір гібридної архітектури з використанням методів глибокого навчання (Neural Collaborative Filtering). Доведено, що саме такий підхід забезпечує необхідну точність та адаптивність системи шляхом комплексної обробки історії взаємодій та контентних ознак фільмів.

У другому розділі сформовано методологічне та технологічне підґрунтя для реалізації рекомендаційної системи.

Шляхом порівняльного аналізу обґрунтовано вибір гібридного технологічного стеку на базі Python та фреймворку PyTorch, що дозволяє ефективно поєднати серверну логіку (Flask) з високопродуктивними обчисленнями нейронних мереж.

Встановлено структурні особливості датасету MovieLens 100K шляхом розвідувального аналізу даних (EDA). Виявлено критичні фактори, що впливають на якість навчання моделі: висока розрідженість матриці взаємодій ($93,7\%$, наприклад, якщо знаєте цифру — додайте), дисбаланс у розподілі оцінок (bias популярності) та нерівномірна активність користувачів.

Формалізовано вимоги до попередньої обробки даних. На основі виявлених аномалій та диспропорцій визначено стратегію препроцесингу, яка включатиме фільтрацію шуму та нормалізацію даних, що є необхідною умовою для коректної роботи архітектури Neural Collaborative Filtering.

У третьому розділі виконано програмну реалізацію та експериментальне дослідження розробленої інформаційної технології.

Виконано підготовку даних MovieLens, що включає індексне кодування категоріальних ознак та фільтрацію шуму. Для оптимізації обчислювальних

ресурсів застосовано зберігання даних у форматі розріджених матриць (Sparse Matrix CSR), що забезпечило ефективну роботу з масивом даних MovieLens.

Розроблено та досліджено три модифікації нейромережевих архітектур на базі Neural Collaborative Filtering (NCF). Порівняльний аналіз підтвердив, що архітектура NCFWithGenres2, яка використовує роздільну обробку ембедінгів та жанрових ознак, демонструє найвищу точність прогнозування.

Використано фреймворк Optuna для налаштування гіперпараметрів, що дозволило зменшити середньоквадратичну похибку (RMSE) моделей на 8–12%. Найкращий досягнутий результат на тестовій вибірці показала модель NCFWithGenres2: $RMSE = 0.8627$, $MAE = 0.6617$.

Створено повнофункціональний програмний комплекс із трирівневою архітектурою (Flask/Python + JavaScript). Впроваджено інноваційний компонент взаємодії з користувачем – інтерфейс візуального вибору настрою, що дозволило реалізувати механізм контекстно-залежних рекомендацій на додаток до персоналізованих пропозицій на основі історії переглядів.

У четвертому розділі проведено оцінку комерційного потенціалу розробки, яка є вище середнього. Прогнозовані витрати на виконання та впровадження науково-дослідної роботи складають 62 850,45 грн. Розрахунки демонструють привабливі інвестиційні показники: термін окупності проекту становить 1,7 року, а приведена вартість майбутніх чистих прибутків оцінюється в 502 330,22 грн.

Отже, мета роботи досягнута шляхом створення комплексної інформаційної технології, яка поєднує сучасні методи глибокого навчання з економічною ефективністю. Результати роботи готові до практичного використання як основа для комерційних стрімінгових платформ або систем персоналізації контенту.

За результатами роботи було опубліковано тези на LV Всеукраїнській науково-технічній конференції підрозділів Вінницького національного технічного університету НТКП ВНТУ (2025-2026).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Тарасовський Т. С., Жуков С.О. ПОРІВНЯЛЬНИЙ АНАЛІЗ МОДИФІКАЦІЙ NCF З ІНТЕГРАЦІЄЮ ЖАНРОВИХ ОЗНАК ДЛЯ СИСТЕМИ РЕКОМЕНДАЦІЙ КІНОФІЛЬМІВ. *LV Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації (Вінниця, 2025-2026 рр.)*. URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa2026/paper/view/26648/21952 (дата звернення: 08.12.2025)

2. Мокін, В. Б. Дратований, М. В. Наука про дані: машинне навчання та інтелектуальний аналіз даних: електронний навчальний посібник комбінованого (локального та мережевого) використання. ВНТУ. (2024) URL: https://pdf.lib.vntu.edu.ua/books/2024/Mokin_2024_263.pdf (дата звернення: 11.11.2025)

3. Мелешко, Є. В. Рекомендаційні системи у складних комп'ютерних мережа. Електронний навчальний посібник. ЦНТУ. (2025) URL: https://dspace.kntu.kr.ua/handle/123456789/16098 (дата звернення: 15.11.2025)

4. Ziyuan Xia, Anchen Sun, Jingyi Xu, Yuanzhe Peng, Rui Ma, Minghui Cheng Contemporary Recommendation Systems on Big Data and Their Applications: A Survey, *IEEE Access* 2024. DOI: https://doi.org/10.48550/arXiv.2206.02631 (дата звернення: 15.11.2025)

5. How Netflix's Recommendations System Works. URL: https://help.netflix.com/en/node/100639?utm_ (дата звернення: 15.11.2025)

6. Chengyuan Liu, How Netflix's Recommendations System Works, *Procedia Computer Science* (2025) DOI: https://doi.org/10.1016/j.procs.2025.08.005 (дата звернення: 20.11.2025)

7. Deepjyoti Roy. Mala Dutta A systematic review and research perspective on recommender systems. *Journal of Big Data* (2022) DOI: https://doi.org/10.1186/s40537-022-00592-5 (дата звернення: 20.11.2025)

8. Sambandam Jayalakshmi, Narayanan Ganesh, Robert Čep, Janakiraman Senthil Murugan. Movie Recommender Systems: Concepts, Methods, Challenges, and Future Directions. *Sensors* 2022) DOI: https://doi.org/10.3390/s22134904 (дата звернення: 25.11.2025)

9. Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V., XLNet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems* / 2019 DOI: https://doi.org/10.48550/arXiv.1906.08237 (дата звернення: 20.11.2025)

10. Collaborative Filtering-Based Recommender Systems: A Deep Dive URL: https://futurewebai.com/blogs/collaborative-filtering-basedrecommendation? (дата звернення: 15.11.2025)

11. Полурєзов Д. Від концепції до реалізації: як побудувати ефективну рекомендаційну систему на основі машинного навчання. 2024 р. URL: https://dou.ua/forums/topic/47504/ (дата звернення: 15.11.2025)

12. Harris Papadakis, Antonis Papagrigoriou, Eleftherios Kosmas, Costas Panagiotakis, Smaragda Markaki and Paraskevi Fragopoulou Content-Based Recommender Systems Taxonomy. *Foundations of Computing and Decision Sciences*. 2023. DOI: https://doi.org/10.2478/fcds-2023-0009 (дата звернення: 15.11.2025)

13. How to Build a Deep Learning Powered Recommender System, URL:http://developer.nvidia.com/blog/how-to-build-a-winning-recommendation system-part-2-deep-learning-for-recommender-systems/ (дата звернення: 15.11.2025)

14. Flask's documentation URL: https://flask.palletsprojects.com /en/stable/ (дата звернення: 15.11.2025)

15. PyTorch Documentation. URL: https://pytorch.org/projects/pytorch/ (дата звернення: 15.11.2025)

16. Visual Studio Code documentation URL: https://code.visualstudio.com/docs (дата звернення: 15.11.2025)

17. MovieLens Small-Latest Dataset. Kaggle. URL: https://www.kaggle.com/datasets/shubhammehta21/movie-lens-small-latest-dataset (дата звернення: 15.11.2025)

18. Ramzan B., Bajwa I.S., Jamil N., Amin R.U., Ramzan S., and Sarwar N. An Intelligent Data Analysis for Recommendation Systems. *Scientific Programming*. 2019. DOI: https://doi.org/10.1155/2019/5941096 (дата звернення: 15.11.2025)

19. Google Developers. Колаборативна фільтрація. Machine Learning. URL: https://developers.google.com/machine-learning/recommendation/collaborative/basics (дата звернення: 15.11.2025)

20. Zhang, Shuai, A Survey on Deep Learning for Recommender Systems: From Collaborative Filtering to Content-Aware and Socially-Aware Approaches. *ACM Computing Surveys (CSUR)*. 2022 DOI: https://doi.org/10.1145/3285029 (дата звернення: 15.11.2025)

21. H. Zhou, F. Xiong, H. Chen, A Comprehensive Survey of Recommender Systems Based on Deep Learning. *Applied Sciences*. 2023. DOI: https://doi.org/10.3390/app132011378 (дата звернення: 15.11.2025)

22. Yaneya МАШИННЕ НАВЧАННЯ ПРОСТИМИ СЛОВАМИ URL: http://www.mmf.lnu.edu.ua/ar/1739 (дата звернення: 15.11.2025)

23. Козловський В. О., Лесько О. Й., Кавецький В. В. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт, Вінниця : ВНТУ, 2021. 42 с.

24. В. Прядченко, Т. Ліхоузова Рекомендаційні системи для стримінгових платформ відеоконтенту, *Адаптивні системи автоматичного управління*, 2025. DOI: https://doi.org/10.20535/1560-8956.47.2025.340203 (дата звернення: 15.11.2025)

Додаток А

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
«___» _____ 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на магістерську кваліфікаційну роботу
«Інформаційна технологія аналізу та рекомендації кінофільмів методами машинного навчання»
08-34.МКР.014.02.000.ТЗ

Керівник: к.т.н., доц. каф. САІТ
_____ Сергій ЖУКОВ
«___» _____ 2025 р.

Розробив: студент гр. 2ІСТ-24м
_____ Тарас ТАРАСОВСЬКИЙ
«___» _____ 2025 р.

Вінниця 2025

1. Підстава для проведення робіт

Підставою для виконання роботи є наказ № __ по ВНТУ від «__» _____ 2025 р., та індивідуальне завдання на МКР, затверджене протоколом № __ засідання кафедри САІТ від «__» _____ 2025 р.

2. Джерела розробки:

- Мокін, В. Б., Дратований, М. В. Наука про дані: машинне навчання та інтелектуальний аналіз даних: електронний навчальний посібник комбінованого (локального та мережевого) використання. ВНТУ. (2024) URL: https://pdf.lib.vntu.edu.ua/books/2024/Mokin_2024_263.pdf
- Мелешко, Є. В. Рекомендаційні системи у складних комп'ютерних мережа. Електронний навчальний посібник. ЦНТУ. (2025) URL: https://dspace.kntu.kr.ua/handle/123456789/16098

3. Мета і призначення роботи:

Метою даної роботи є розроблення інформаційної технології аналізу та рекомендації кінофільмів методами машинного навчання, що підвищує точність персоналізації за рахунок поєднання аналізу історії переглядів, контентних ознак та емоційного контексту (настрою) користувача.

4.Вихідні дані для проведення робіт:

Набір даних Kaggle Dataset „Movie Lens Small Latest Dataset" https://www.kaggle.com/datasets/shubhammehta21/movie-lens-small-latest-dataset

5.Методи дослідження:

Аналіз і синтез наукових джерел, методи машинного навчання, статистичний аналіз даних, розвідувальний аналіз даних (EDA), методи візуалізації даних, оптимізація гіперпараметрів, порівняльний аналіз моделей, оцінювання ефективності за якісними метриками, розроблення UML-діаграм.

6. Етапи роботи і терміни їх виконання:

1. Аналіз сучасного стану моніторингу атмосферного повітря____ – ____
2. Основні етапи виконання роботи та огляд набору вхідних даних ..... ____ – ____
3. Обробка результатів спостережень та візуалізація даних у середовищі Kaggle ..... ____ – ____
4. Розроблення інформаційної технології ..... ____ – ____
5. Економічна частина..... ____ – ____
6. Оформлення матеріалів до захисту МКР. ..... ____ – ____

7. Очікувані результати та порядок реалізації:

У результаті роботи розроблено та економічно обґрунтовано інформаційну технологію рекомендацій фільмів на основі NCF з контекстними настроєвими рекомендаціями у веб-системі.

8. Вимоги до розробленої документації

Пояснювальна записка оформлена у відповідності до вимог «Методичних вказівок до виконання магістерських кваліфікаційних робіт для студентів спеціальності 126 «Інформаційні системи та технології» (освітня програма «Інформаційні технології аналізу даних та зображень»)

9. Порядок приймання роботи

Публічний захист..... «___» _____ 2025 р.

Початок розробки ..... «___» _____ 2025 р.

Граничні терміни виконання МКР..... «___» _____ 2025 р.

Розробив студент групи 2ІСТ-24м _____ Тарас ТАРАСОВСЬКИЙ

Додаток Б

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Інформаційна технологія аналізу та рекомендації кінофільмів методами машинного навчання»

Тип роботи: магістерська кваліфікаційна робота

Підрозділ: кафедра САІТ, ФІІТА, гр. 2ІСТ-24м

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 2,05%

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне):

- ■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту
- □ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- □ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Віталій МОКІН, зав. каф. САІТ _____
(підпис)

Сергій ЖУКОВ, доц. каф. САІТ _____
(підпис)

Особа, відповідальна за перевірку _____ Сергій ЖУКОВ
(підпис)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Сергій ЖУКОВ, к.т.н., доц. каф. САІТ
(підпис)

Здобувач _____ Тарас ТАРАСОВСЬКИЙ
(підпис)

Додаток В

Лістинг коду app.py

```
import numpy as np
import pandas as pd
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader
from scipy.sparse import csr_matrix
import re
from sklearn.metrics import mean_squared_error, mean_absolute_error
import os
import matplotlib.pyplot as plt

movies = pd.read_csv("movies (1).csv")
ratings = pd.read_csv("ratings.csv")
def preprocess_genres(genres):
    return [genre.lower() for genre in genres.split('|')] if pd.notna(genres) else []
movies['genre_list'] = movies['genres'].apply(preprocess_genres)
all_genres = set(genre for sublist in movies['genre_list'] for genre in sublist)
genre_to_idx = {genre: idx for idx, genre in enumerate(all_genres)}
final_dataset = ratings.pivot(index="movieId", columns="userId",
values="rating").fillna(0)
final_dataset = final_dataset.loc[ratings.groupby("movieId")['rating'].agg('count') >
10, :]
final_dataset = final_dataset.loc[:, ratings.groupby("userId")['rating'].agg('count') >
50]
csr_data = csr_matrix(final_dataset.values)
final_dataset.reset_index(inplace=True)
```

```
movieid_to_idx = {movie_id: idx for idx, movie_id in
enumerate(final_dataset['movieId'])}
def csr_to_tensors(csr_mat):
    coo = csr_mat.tocoo()
    values = coo.data
    indices = np.vstack((coo.row, coo.col))
    idx = torch.LongTensor(indices)
    vals = torch.FloatTensor(values)
    sparse_tensor = torch.sparse_coo_tensor(idx, vals, torch.Size(coo.shape))
    return sparse_tensor
sparse_tensor = csr_to_tensors(csr_data)
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Using device: {device}")
class NCFWithGenres(nn.Module):
    def __init__(self, num_users, num_items, num_genres,
                 emb_dim=50, hidden_dim_individual=32,
                 hidden_layers_combined=[64]):
        super().__init__()
        # Ембеддинги для користувачів, фільмів та жанрів
        self.user_emb = nn.Embedding(num_users, emb_dim)
        self.item_emb = nn.Embedding(num_items, emb_dim)
        self.genre_emb = nn.EmbeddingBag(num_genres, emb_dim, mode='sum')
        self.fc_user = nn.Sequential(
            nn.Linear(emb_dim, hidden_dim_individual),
            nn.ReLU()
        )
        self.fc_item = nn.Sequential(
            nn.Linear(emb_dim, hidden_dim_individual),
            nn.ReLU()
        )
```

```
        self.fc_genre = nn.Sequential(
            nn.Linear(emb_dim, hidden_dim_individual),
            nn.ReLU()
        )
        layers = []
        input_dim = hidden_dim_individual * 3
        for hidden_dim in hidden_layers_combined:
            layers.append(nn.Linear(input_dim, hidden_dim))
            layers.append(nn.ReLU())
            input_dim = hidden_dim
        layers.append(nn.Linear(input_dim, 1))
        self.fc_combined = nn.Sequential(*layers)
    def forward(self, user, item, genre_indices, offsets):
        u_emb = self.user_emb(user)
        i_emb = self.item_emb(item)
        g_emb = self.genre_emb(genre_indices, offsets)
        u_emb = self.fc_user(u_emb)
        i_emb = self.fc_item(i_emb)
        g_emb = self.fc_genre(g_emb)
        concat = torch.cat([u_emb, i_emb, g_emb], dim=-1)
        return self.fc_combined(concat).squeeze()

movie_genres = []
genre_offsets = [0]

for movie_id in final_dataset['movieId']:
    idx = movieid_to_idx[movie_id]
    movie_row = movies[movies['movieId'] == movie_id]
    if not movie_row.empty:
        genres = movie_row.iloc[0]['genre_list']
```

```
        genre_indices = [genre_to_idx[g] for g in genres if g in genre_to_idx]
        movie_genres.extend(genre_indices)
        genre_offsets.append(len(movie_genres))
genre_offsets_tensor = torch.tensor(genre_offsets[:-1], dtype=torch.int64)
genre_indices_tensor = torch.tensor(movie_genres, dtype=torch.int64)

class GenreDataset(Dataset):
    def __init__(self, sparse_tensor, genre_offsets, genre_indices):
        self.sparse_tensor = sparse_tensor
        self.coo = sparse_tensor.coalesce().indices().t()
        self.values = sparse_tensor.coalesce().values()
        self.genre_offsets = genre_offsets
        self.genre_indices = genre_indices
    def __len__(self):
        return len(self.values)
    def __getitem__(self, idx):
        movie_idx = self.coo[idx, 0].item()
        user_idx = self.coo[idx, 1].item()
        rating = self.values[idx]
        if movie_idx >= len(self.genre_offsets):
            return None
        offset = self.genre_offsets[movie_idx]
        next_offset = self.genre_offsets[movie_idx + 1] if movie_idx + 1 <
len(self.genre_offsets) else len(self.genre_indices)
        length = next_offset - offset
        if length == 0:
            return None
        return user_idx, movie_idx, offset, length, rating
def collate_fn(batch):
    batch = [b for b in batch if b is not None]
```

```
    if not batch:
        return None
    return torch.utils.data.dataloader.default_collate(batch)
dataset = GenreDataset(sparse_tensor, genre_offsets_tensor, genre_indices_tensor)
train_size = int(0.8 * len(dataset))
test_size = len(dataset) - train_size
train_dataset, test_dataset = torch.utils.data.random_split(dataset, [train_size,
test_size])
train_loader = DataLoader(train_dataset, batch_size=1024, shuffle=True,
collate_fn=collate_fn)
test_loader = DataLoader(test_dataset, batch_size=1024, shuffle=False,
collate_fn=collate_fn)
num_users = final_dataset.shape[1]
num_items = final_dataset.shape[0]
num_genres = len(genre_to_idx)
model = NCFWithGenres(num_users, num_items, num_genres).to(device)
def train(model, train_loader, test_loader, epochs=5):
    criterion = nn.MSELoss()
    optimizer = optim.Adam(model.parameters(), lr=0.001)
    train_losses = []
    test_losses = []
    rmse_scores = []
    mae_scores = []
    for epoch in range(epochs):
        model.train()
        total_train_loss = 0
        num_batches = 0
        for batch in train_loader:
            if batch is None:
                continue
```

```
        user_idx, movie_idx, offset, length, true_rating = batch
        user_idx = user_idx.to(device)
        movie_idx = movie_idx.to(device)
        true_rating = true_rating.to(device)
        all_genre_indices = []
        batch_offsets = [0]
        total = 0
        for i in range(len(offset)):
            start = offset[i].item()
            end = start + length[i].item()
            if start < len(genre_indices_tensor) and end <=
len(genre_indices_tensor):
                genre_slice = genre_indices_tensor[start:end]
                all_genre_indices.append(genre_slice)
                total += len(genre_slice)
                batch_offsets.append(total)
        if not all_genre_indices:
            continue
        all_genre_indices = torch.cat(all_genre_indices).to(device)
        batch_offsets = torch.tensor(batch_offsets[:-1],
dtype=torch.int64).to(device)
        pred_rating = model(user_idx, movie_idx, all_genre_indices,
batch_offsets)
        loss = criterion(pred_rating, true_rating)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        total_train_loss += loss.item()
        num_batches += 1
```

```
        avg_train_loss = total_train_loss / num_batches if num_batches > 0 else 0
        train_losses.append(avg_train_loss)
        test_loss, rmse, mae = evaluate(model, test_loader)
        test_losses.append(test_loss)
        rmse_scores.append(rmse)
        mae_scores.append(mae)

        print(f"Epoch {epoch+1}/{epochs}, Train Loss: {avg_train_loss:.4f}, Test
Loss: {test_loss:.4f}, RMSE: {rmse:.4f}, MAE: {mae:.4f}")
    torch.save(model.state_dict(), 'model.pth')
    print("Model saved to model.pth")
    plt.figure(figsize=(12, 8))
    plt.subplot(2, 2, 1)
    plt.plot(train_losses, label='Train Loss')
    plt.plot(test_losses, label='Test Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()
    plt.title('Training and Validation Loss')
    plt.subplot(2, 2, 2)
    plt.plot(rmse_scores, label='RMSE', color='orange')
    plt.xlabel('Epoch')
    plt.ylabel('RMSE')
    plt.title('Root Mean Squared Error')
    plt.subplot(2, 2, 3)
    plt.plot(mae_scores, label='MAE', color='green')
    plt.xlabel('Epoch')
    plt.ylabel('MAE')
    plt.title('Mean Absolute Error')
```

```
    plt.tight_layout()
    plt.savefig('training_metrics.png')
    plt.show()
    return train_losses, test_losses, rmse_scores, mae_scores
def evaluate(model, dataloader):
    model.eval()
    criterion = nn.MSELoss()
    total_loss = 0
    all_preds = []
    all_targets = []
    num_batches = 0
    with torch.no_grad():
        for batch in dataloader:
            if batch is None:
                continue
            user_idx, movie_idx, offset, length, true_rating = batch
            user_idx = user_idx.to(device)
            movie_idx = movie_idx.to(device)
            true_rating = true_rating.to(device)
            all_genre_indices = []
            batch_offsets = [0]
            total = 0
            for i in range(len(offset)):
                start = offset[i].item()
                end = start + length[i].item()
                if start < len(genre_indices_tensor) and end <=
len(genre_indices_tensor):
                    genre_slice = genre_indices_tensor[start:end]
                    all_genre_indices.append(genre_slice)
                    total += len(genre_slice)
```

```
                batch_offsets.append(total)
            if not all_genre_indices:
                continue
            all_genre_indices = torch.cat(all_genre_indices).to(device)
            batch_offsets = torch.tensor(batch_offsets[:-1],
dtype=torch.int64).to(device)
            pred_rating = model(user_idx, movie_idx, all_genre_indices,
batch_offsets)
            all_preds.extend(pred_rating.cpu().numpy())
            all_targets.extend(true_rating.cpu().numpy())
            loss = criterion(pred_rating, true_rating)
            total_loss += loss.item()
            num_batches += 1

    if num_batches == 0:
        return 0, 0, 0
    avg_loss = total_loss / num_batches
    rmse = np.sqrt(mean_squared_error(all_targets, all_preds))
    mae = mean_absolute_error(all_targets, all_preds)

    return avg_loss, rmse, mae
def recommend_similar(model, movie_title, movies_df, genre_to_idx,
movieid_to_idx, top_n=10):
    # Знайти фільм у базі
    movie_title_cleaned = movie_title.strip()
    movie = movies_df[movies_df['title'].str.contains(movie_title_cleaned,
case=False, na=False, regex=False)]

    if movie.empty:
        print(f"Фільм '{movie_title}' не знайдено.")
```

```
        similar_titles =
movies_df[movies_df['title'].str.contains(movie_title.split()[0], case=False,
na=False)]
        if not similar_titles.empty:
            print("Можливо ви мали на увазі:")
            for t in similar_titles['title'].head(5):
                print(f" - {t}")
        return []
    movie = movie.iloc[0]
    movie_id = movie['movieId']
    genres = movie['genre_list']
    if movie_id not in movieid_to_idx:
        print(f"Фільм '{movie_title}' не має достатньої кількості оцінок.
Використовуються рекомендації на основі жанрів.")
        return recommend_by_genre_only(movie_title, genres, movies_df,
movieid_to_idx, top_n)
    genre_indices = [genre_to_idx[g] for g in genres if g in genre_to_idx]
    model.eval()
    recommendations = []
    with torch.no_grad():
        for _, row in movies_df.iterrows():
            if row['movieId'] not in movieid_to_idx:
                continue
            if row['title'] == movie['title']:
                continue
            curr_genres = row['genre_list']
            curr_indices = [genre_to_idx[g] for g in curr_genres if g in genre_to_idx]
            if not curr_indices:
                continue
```

```
            movie_idx = torch.tensor([movieid_to_idx[row['movieId']]],
dtype=torch.int64).to(device)
            user_tensor = torch.tensor([0], dtype=torch.int64).to(device)
            curr_genre_tensor = torch.tensor(curr_indices,
dtype=torch.int64).to(device)
            curr_offsets = torch.tensor([0], dtype=torch.int64).to(device)
            pred = model(
                user_tensor,
                movie_idx,
                curr_genre_tensor,
                curr_offsets
            ).item()

            common_genres = len(set(genres) & set(curr_genres))
            recommendations.append((row['title'], pred, common_genres))
    recommendations = [(title, score, common) for title, score, common in
recommendations if not is_sequel(title)]
    recommendations.sort(key=lambda x: (x[2], x[1]), reverse=True)
    top_recommendations = [title for title, _, _ in recommendations[:top_n]]
    print("\nТоп рекомендацій з прогнозованим рейтингом:")
    for i, (title, score, common) in enumerate(recommendations[:top_n], 1):
        print(f"{i}. {title} (прогноз: {score:.2f}, спільні жанри: {common})")
    return top_recommendations
def recommend_by_genre_only(input_title, input_genres, movies_df,
movieid_to_idx, top_n=10):
    recommendations = []
    for _, row in movies_df.iterrows():
        # Пропускаємо оригінальний фільм та сіквели
        if row['title'] == input_title or is_sequel(row['title']):
            continue
```

```
        if row['movieId'] not in movieid_to_idx:
            continue
        curr_genres = row['genre_list']
        common_genres = len(set(input_genres) & set(curr_genres))
        if common_genres > 0:
            recommendations.append((row['title'], common_genres))
    recommendations.sort(key=lambda x: x[1], reverse=True)
    top_recommendations = [title for title, _ in recommendations[:top_n]]
    print("\nТоп рекомендацій на основі жанрів:")
    for i, (title, common) in enumerate(recommendations[:top_n], 1):
        print(f"{i}. {title} (спільні жанри: {common})")
    return top_recommendations
def is_sequel(title):
    sequel_patterns = [
        r"\b2\b", r"\b3\b", r"\b4\b", r"\b5\b",
        r"\bII\b", r"\bIII\b", r"\bIV\b", r"\bV\b",
        r"повернення", r"return", r"revenge", r"resurrection",
        r"part\s*\d+", r"частина\s*\d+"
    ]
    for pattern in sequel_patterns:
        if re.search(pattern, title, re.IGNORECASE):
            return True
    return False

def main():
    if not os.path.exists('model.pth'):
        print("Training model...")
        train_losses, test_losses, rmse_scores, mae_scores = train(model,
train_loader, test_loader, epochs=5)
    else:
```

```
        model.load_state_dict(torch.load('model.pth', map_location=device))
        print("Model loaded from model.pth")
    model.eval()
    test_loss, test_rmse, test_mae = evaluate(model, test_loader)
    print(f"\nФінальна оцінка моделі на тестових даних:")
    print(f"Test Loss: {test_loss:.4f}")
    print(f"RMSE: {test_rmse:.4f}")
    print(f"MAE: {test_mae:.4f}")
    movie_title = input("\nВведіть назву фільму: ")
    recommendations = recommend_similar(model, movie_title, movies,
genre_to_idx, movieid_to_idx)
    print("\nРекомендовані схожі фільми:")
    for i, title in enumerate(recommendations, 1):
        print(f"{i}. {title}")
```

Додаток Г

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АНАЛІЗУ ТА РЕКОМЕНДАЦІЇ КІНОФІЛЬМІВ МЕТОДАМИ МАШИННОГО НАВЧАННЯ

Нормоконтроль: к.т.н., доцент

_____ Сергій ЖУКОВ

«___» _____ 2025 р.

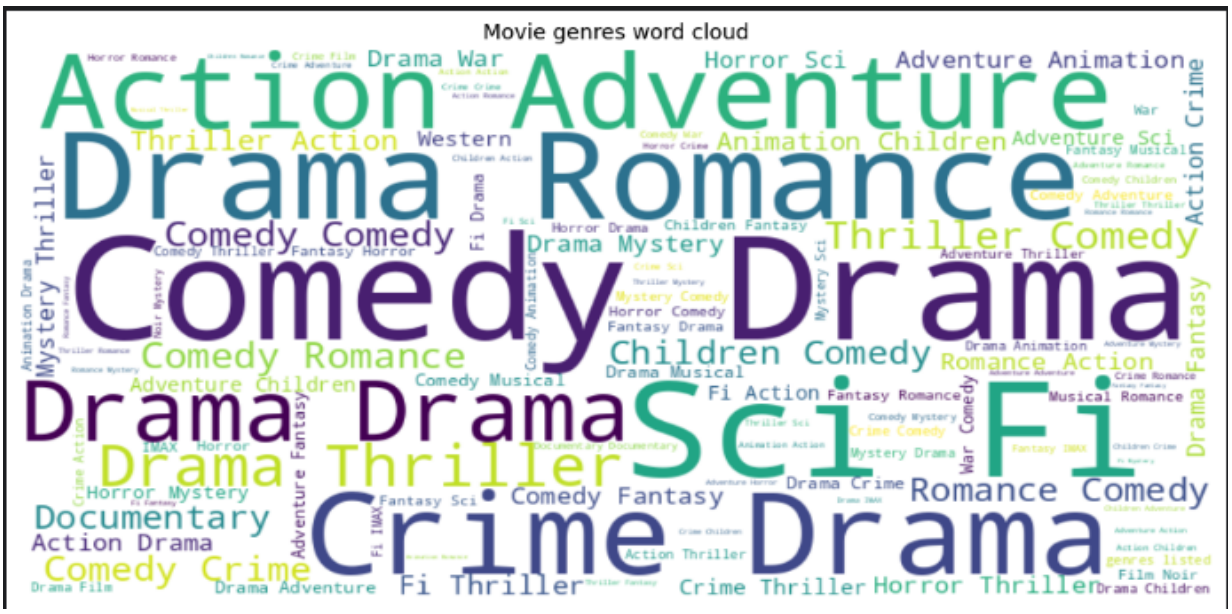

Рисунок Г.1 – Візуалізація частоти жанрів у вигляді хмари слів

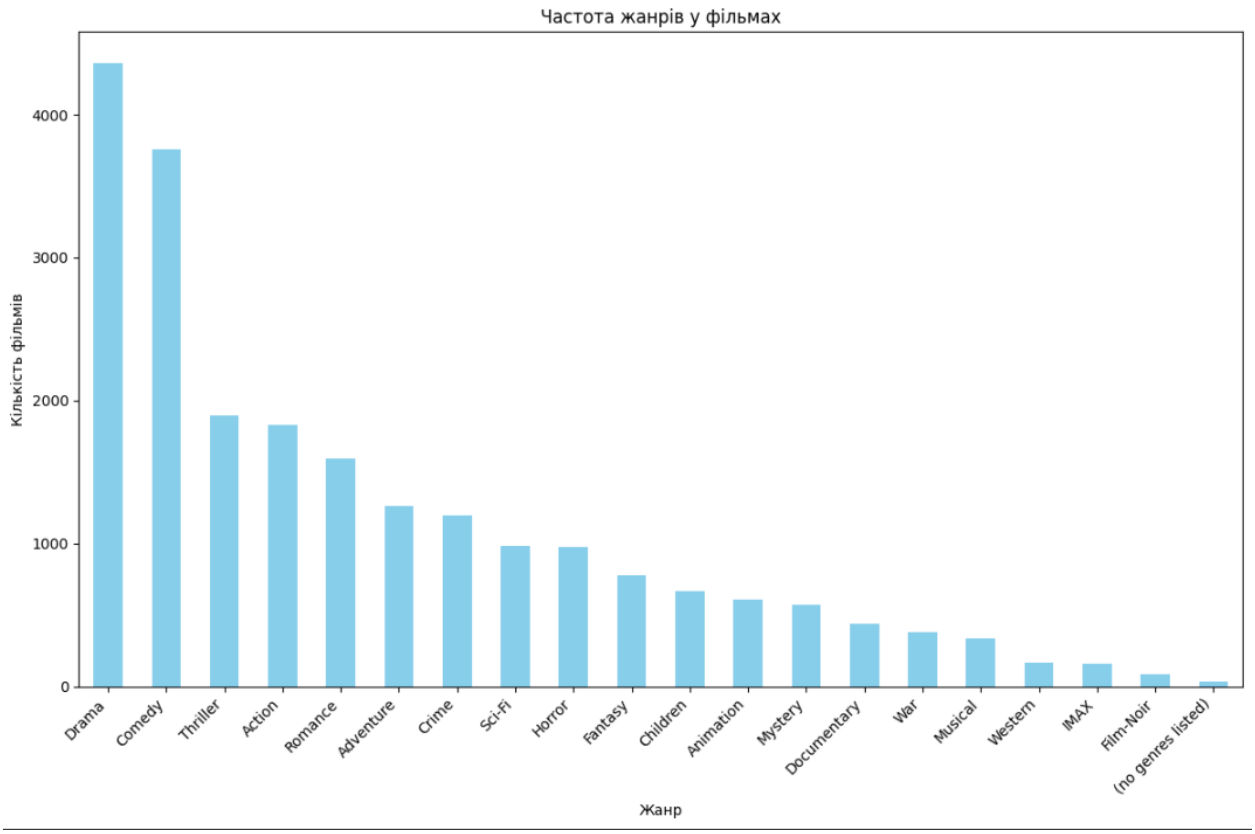

Рисунок Г.2 – Гістограма розподілу фільмів за жанрами у досліджуваному наборі даних

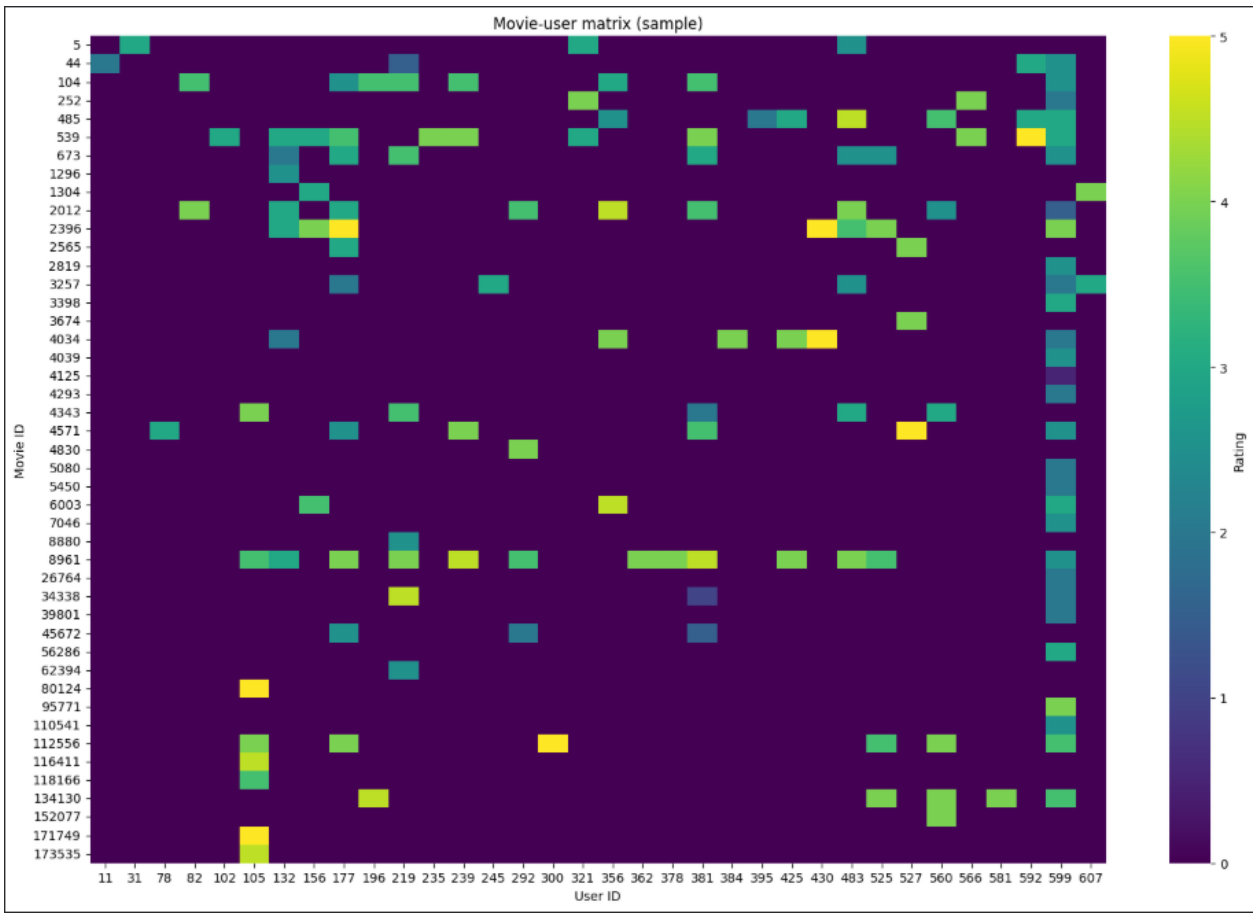

Рисунок Г.3 – Теплова матриця користувач-фільм

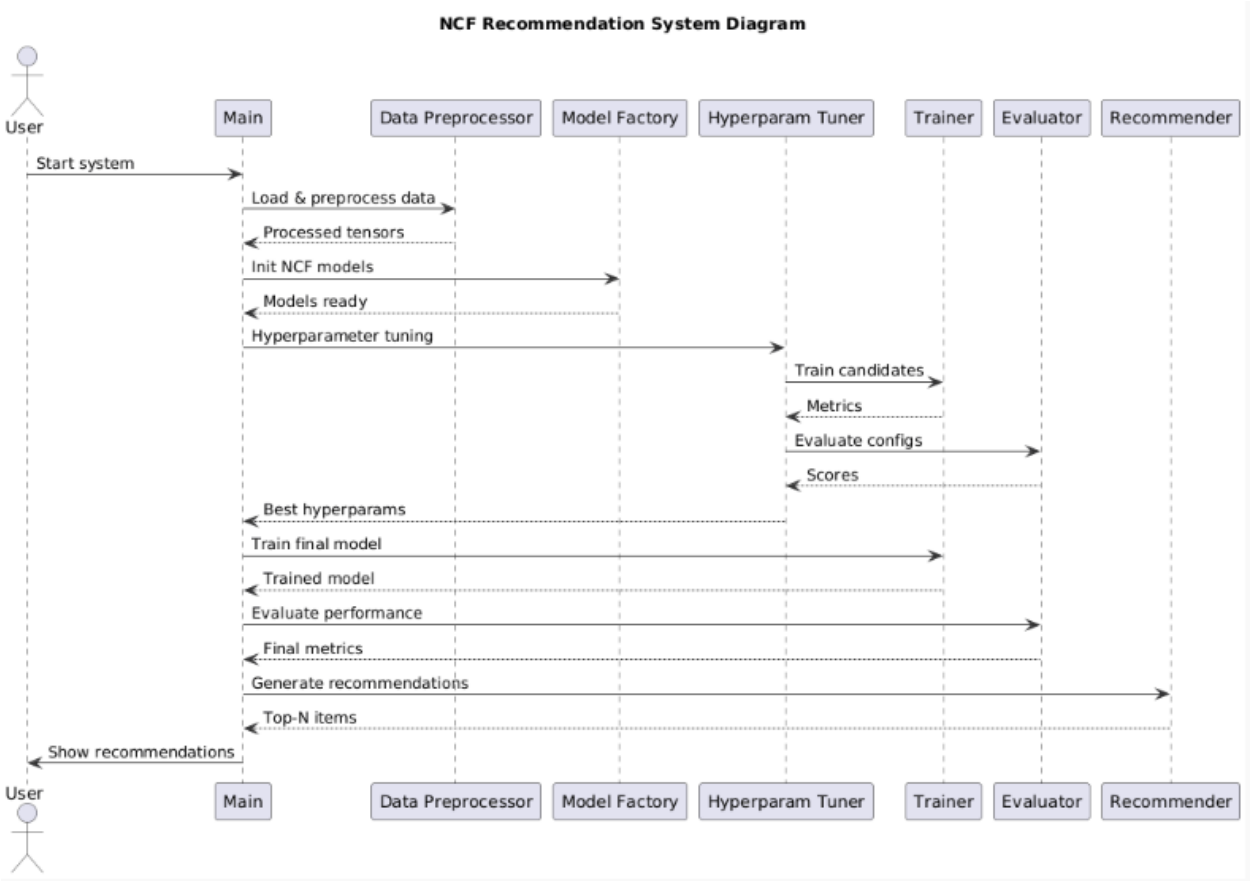

Рисунок Г.4 – Узагальнена діаграма роботи рекомендаційної системи NCF

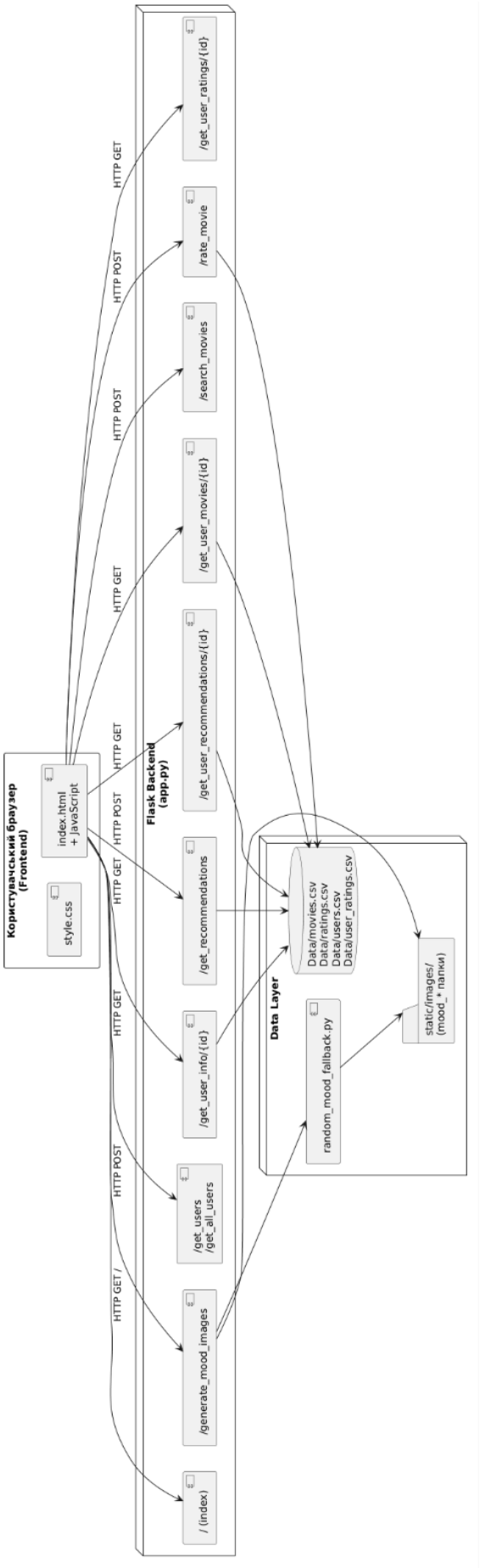

Рисунок Г.5 – Component діаграма веб-системи

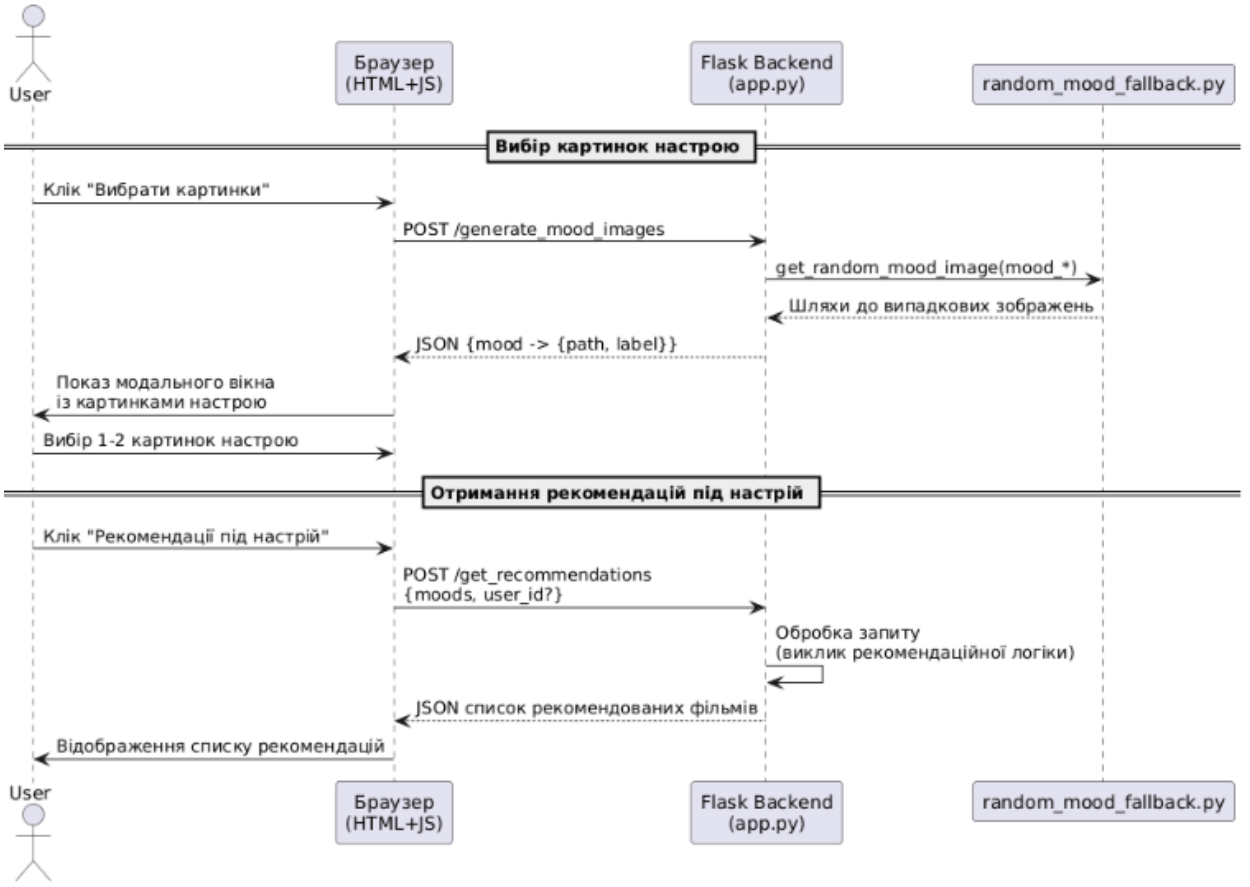

Рисунок Г.6 – Діаграма послідовності «Отримання рекомендацій під настрій»

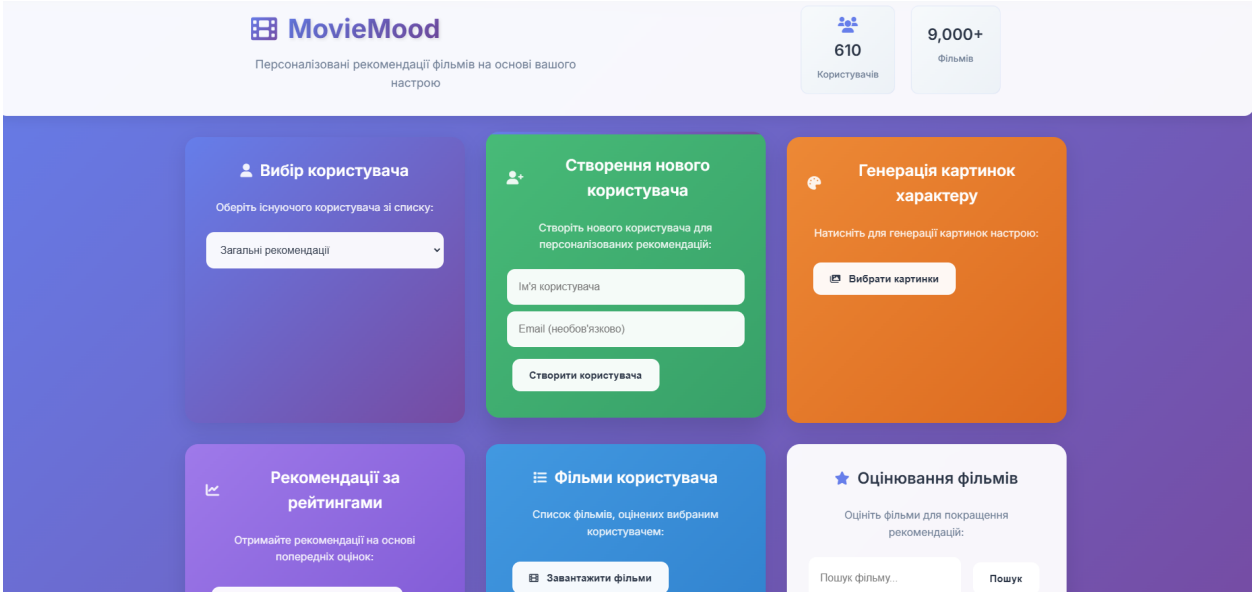

Рисунок Г.7 – Інтерфейс веб-системи – персоналізовані рекомендації фільмів за настроєм і рейтингами