ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Моделі машинного навчання для аналізу та передбачення в сфері кінематографу на основі мови програмування Python»

на здобуття освітнього ступеня бакалавра
зі спеціальності 122 – комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми комп'ютерні науки
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Сергій ГАВРИЛОВ

(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувачка вищої освіти гр. КНД-41
Сергій ГАВРИЛОВ
Ім'я, ПРІЗВИЩЕ

Керівник: доктор філософії,
Юлія БЕРЕЗОВСЬКА
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Рецензент:
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра комп'ютерних наук

Ступінь вищої освіти бакалавр

Спеціальність 122 комп'ютерні науки

Освітньо-професійна програма комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри Віктор ВИШНІВСЬКИЙ
_____Ім'я, ПРІЗВИЩЕ
«____» _____20___р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Гаврилов Сергій Андрійович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Моделі машинного навчання для аналізу та передбачення в сфері кінематографу на основі мови програмування Python
керівник кваліфікаційної роботи Юлія БЕРЕЗОВСЬКА, доктор філософії,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56

2. Строк подання кваліфікаційної роботи «30» травня 2025р.

3. Вихідні дані до кваліфікаційної роботи:
Програмування на Python, створення машинної моделі за допомогою scikit-learn

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Теоретичні основи розробки веб-додатку приватної бібліотеки
2. Аналіз обраних технологій
3. Опис розробленого веб-додатку

5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	28.02.2025 – 28.03.2025	Виконано
2	Дослідження поставленої задачі, технологій та існуючих рішень	29.03.2025 – 17.04.2025	Виконано
3	Аналіз методів розв’язання задачі	18.04.2025 – 26.04.2025	Виконано
4	Розробка моделі та веб-додатку для презентації	27.04.2025 – 05.05.2025	Виконано
5	Написання вступу	06.05.2025 – 12.05.2025	Виконано
6	Написання висновків	13.05.2025 – 17.05.2025	Виконано
7	Написання реферату	18.05.2025 – 22.05.2025	Виконано
8	Підготовка презентації до захисту	23.05.2025	Виконано

Здобувачка вищої освіти _____ Сергій ГАВРИЛОВ
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Юлія БЕРЕЗОВСЬКА
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра : 50 стор., 24 рис., 0 табл., 16 джерело.

Мета роботи – розробка рекомендаційної моделі машинного навчання для кіносервісів.

Об'єкт дослідження – процес аналізу даних у сфері кінематографу та формування персоналізованих рекомендацій для користувачів на основі їхніх уподобань.

Предмет дослідження – методи та засоби машинного навчання для аналізу кінематографічних даних та розробки рекомендаційних систем.

Короткий зміст роботи: Проаналізовані існуючі рішення та їх використання у сфері машинного навчання. Розглянуті вимоги до точності моделей та їх функціональності.

Досліджено основні технології для розробки рекомендаційних систем, такі як Python, scikit-learn, pandas, NumPy, TensorFlow/PyTorch, Matplotlib/Seaborn.

Розроблено модель машинного навчання для аналізу метаданих фільмів, що дозволяє ефективно опрацьовувати інформацію про фільми та користувацькі вподобання в єдиній системі. Система надає можливість користувачам отримувати релевантні пропозиції фільмів на основі їхніх індивідуальних смаків.

МАШИННЕ НАВЧАННЯ, РЕКОМЕНДАЦІЙНІ СИСТЕМИ, SCIKIT-LEARN, КІНЕМАТОГРАФ, АНАЛІЗ ДАНИХ

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ML – машинне навчання

AI – штучний інтелект

Data Science – наука про дані

Data Scientist – науковці даних

TF-IDF (Term Frequency-Inverse Document Frequency) – числова статистика для моделі

ВСТУП

Сьогодні у інформаційному просторі можна помітити експоненційне зростання обсягів даних, що відкриває великі можливості для аналізу та передбачення трендів індустрії розваг. Особливо актуальним стає питання розробки інтелектуальних систем, здатних обробляти великі масиви даних та формувати персоналізовані рекомендації для користувачів. Застосування технологій машинного навчання дозволяє не лише класифікувати та аналізувати існуючий контент, але й прогнозувати успішність нових проєктів, виявляти приховані взаємозв'язки між уподобаннями глядачів та характеристиками контенту.

Персоналізовані рекомендаційні системи на основі алгоритмів машинного навчання надають унікальну можливість кіноглядачам орієнтуватися в надзвичайно широкому асортименті кінопродукції, отримувати релевантні пропозиції відповідно до власних інтересів та вподобань. Такі системи формують цифровий міст між творцями контенту та споживачами, що має культурну, економічну та соціальну цінність.

Актуальність розробки моделей машинного навчання для аналізу та передбачення в сфері кінематографу обумовлена глобальною діджиталізацією індустрії розваг, стрімким розвитком стрімінгових платформ та необхідністю структурування й аналізу колосальних обсягів даних про фільми та серіали. Крім того, точні алгоритми рекомендацій здатні суттєво підвищити задоволеність користувачів, зробити їхній перегляд більш осмисленим та сприяти популяризації якісних кінотворів у суспільстві.

Вплив персоналізованих систем аналізу даних з використанням машинного навчання розглядався у дослідженні "Effect of User Tastes on Personalized

Recommendation"[1], де було проаналізовано вплив індивідуальних смаків користувачів на точність рекомендацій. Дослідники продемонстрували, що алгоритм, який адаптує рекомендації до схожості з конттентом, що цікавить користувача, показує кращі результати в контексті персоналізації контенту, особливо у сфері кінематографу.

Враховуючи великий обсяг досліджень у сфері кінематографу, досі залишаються питання створення ефективних рекомендаційних моделей машинного навчання для кіносервісів, що поєднують аналіз контентних характеристик фільмів та поведінкових патернів користувачів. Існуючі рішення здебільшого фокусуються на окремих аспектах аналізу кінематографічних даних, не забезпечуючи цілісного підходу до формування персоналізованих рекомендацій.

Мета дослідження полягає у розробці рекомендаційної моделі машинного навчання для стрімінгових сервісів на основі мови програмування Python та бібліотеки scikit-learn, що забезпечить ефективне опрацювання метаданих фільмів та формування персоналізованих рекомендацій для користувачів.

Щоб досягти поставленої мети, було визначено наступні завдання дослідження:

а) Здійснити аналіз існуючих рішень у сфері машинного навчання для розробки рекомендаційних систем у кінематографі;

б) обґрунтувати вибір технологічного стеку для розробки системи, зокрема використання Python, Scikit-learn, Pandas, NumPy, Matplotlib та Seaborn;

в) розробити модель машинного навчання для аналізу метаданих фільмів;

г) реалізувати функціональні компоненти системи, що дозволять користувачам отримувати релевантні пропозиції фільмів на основі їхніх індивідуальних смаків;

г) розробити метрики оцінки ефективності створеної моделі та провести її валідацію на реальних даних.

Об'єктом дослідження є процес аналізу даних у сфері кінематографу та формування персоналізованих рекомендацій для користувачів на основі їхніх уподобань.

Предметом дослідження є методи та засоби машинного навчання для аналізу кінематографічних даних та розробки рекомендаційних систем з використанням мови програмування Python.

1 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

1.1 Аналіз сучасних підходів до створення рекомендаційних систем.

З кожним днем кількість доступного контенту невпинно зростає, створюючи так званий "парадокс вибору" для користувачів. За даними дослідження компанії Nielsen, середньостатистичний користувач стрімінгових сервісів витрачає близько 7.4 хвилин лише на вибір фільму чи серіалу для перегляду. [2] Саме тому рекомендаційні системи стали невід'ємною частиною сучасних платформ для споживання медіаконтенту.

Іншим важливим аспектом цієї проблеми є економічні наслідки неефективного вибору контенту. Дослідження, проведене McKinsey & Company у 2023 році, показало, що компанії з розвиненими рекомендаційними системами демонструють збільшення утримання користувачів на 35% та зростання конверсії [3]. У контексті стрімінгових сервісів це означає, що якісні рекомендаційні системи не лише покращують користувацький досвід, але й суттєво впливають на фінансові показники компаній, що продемонстровано на рисунку 1.1.

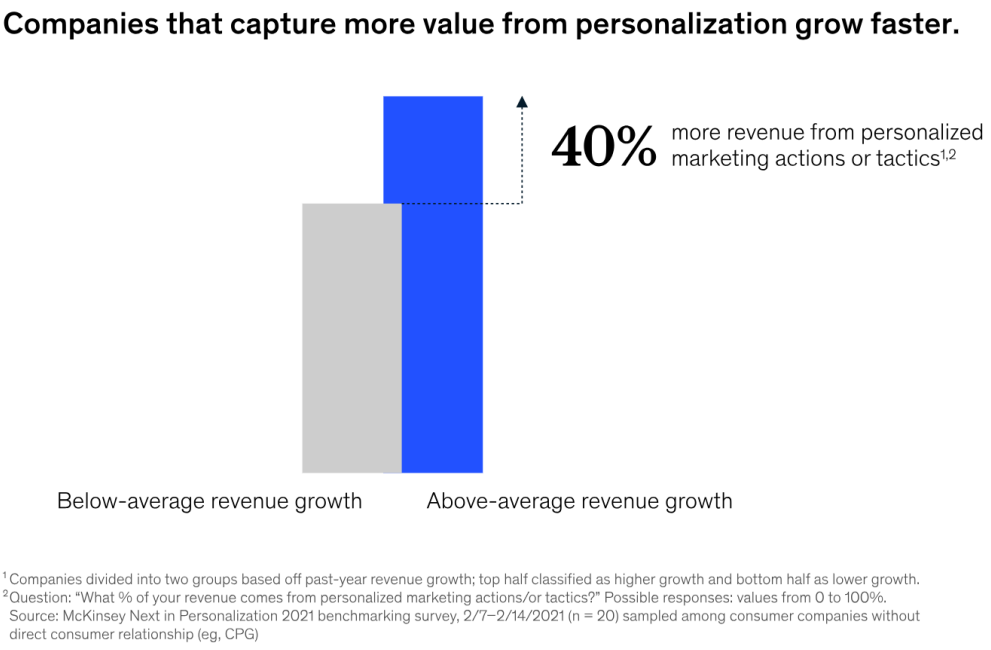


Рисунок 1.1 - Прибуток компаній з рекомендаційними системами та без [4]

Сучасна розробка рекомендаційних систем базується на кількох фундаментальних підходах, які часто комбінуються для досягнення оптимальних результатів:

— Колаборативна фільтрація залишається ключовим методом, який використовує взаємодії між користувачами та об'єктами для передбачення майбутніх уподобань. Наприклад, Amazon використовує цей метод, коли показує вам "Користувачі, які придбали цей товар, також купили..." В сучасних реалізаціях використовуються прогресивні моделі матричної факторизації та нейронних мереж для роботи з розрідженими даними.
— Контентна фільтрація розвинулася від простого порівняння ключових слів до аналізу семантичного змісту за допомогою NLP (обробки природної мови). Наприклад, сервіс Pandora аналізує понад 400 музичних атрибутів кожної пісні, щоб знаходити схожі композиції.
— Гібридні підходи стали промисловим стандартом. Netflix, наприклад, використовує комбінацію понад 20 різних алгоритмів рекомендацій, результати яких агрегуються для формування фінальних рекомендацій користувачам.

На найнижчому рівні всі дані в рекомендаційних системах представлені як числові вектори. Як це працює:

Користувачі та об'єкти (фільми, товари, пісні) перетворюються в числові вектори — набори чисел, що описують їхні характеристики в багатовимірному просторі. Наприклад, фільм "Матриця" може бути представлений вектором, де окремі числа відповідають таким характеристикам як "рівень наукової фантастики", "рівень бойовика", "присутність філософських тем" тощо.

В основі цього представлення лежить концепція векторного простору, де:

— Схожість об'єктів можна виміряти як відстань або кут між їхніми векторами
— Чим ближче вектори, тим більше схожі об'єкти

— Базові операції лінійної алгебри (скалярні добутки, матричні множення) дозволяють ефективно обчислювати схожість між об'єктами

Рекомендаційні системи на машинному рівні працюють з представленням об'єктів та користувачів у вигляді числових векторів. Кожен фільм, товар чи користувач перетворюється на набір чисел, що відображають їхні характеристики в багатовимірному просторі. Коли Netflix рекомендує вам фільм, насправді система знайшла вектор, близький до вашого профілю в цьому просторі.

Ключовим методом є матрична факторизація. Уся інформація про взаємодії користувачів із контентом формує величезну матрицю, яку система розкладає на дві менші — матрицю користувачів і матрицю об'єктів, що продемонстровано на рисунку 1.2.[5] Це математично звучить складно, але по суті система виявляє приховані характеристики, які пояснюють ваші вподобання.

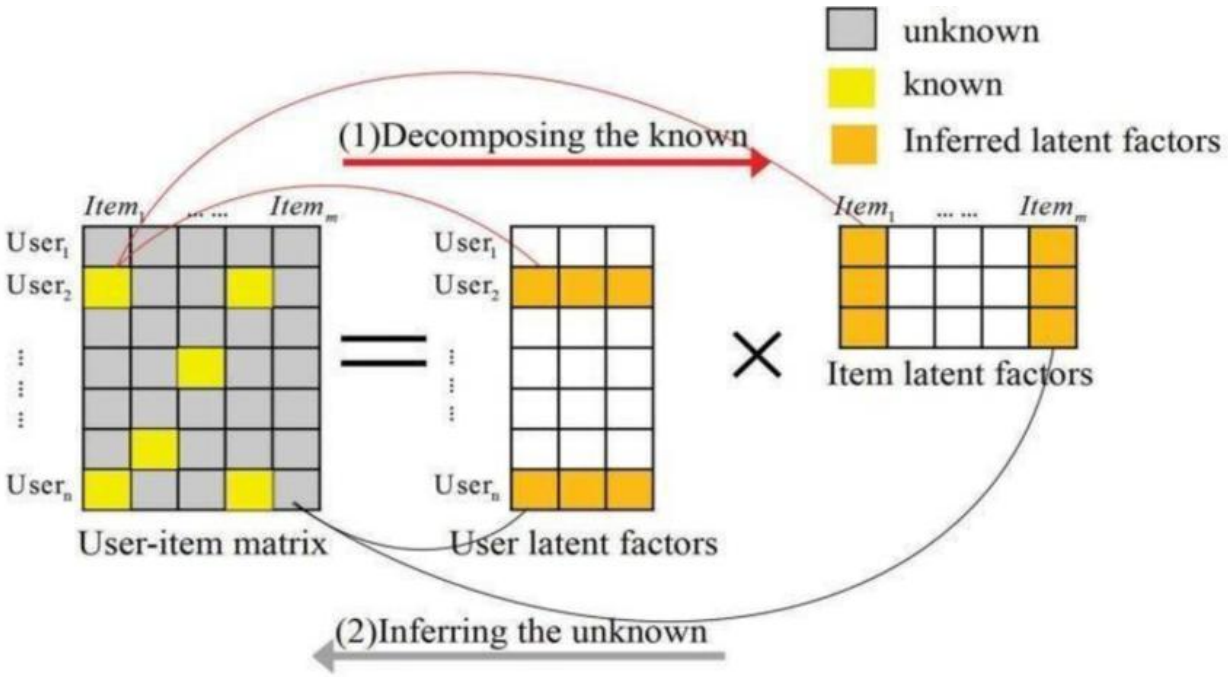


Рисунок 1.2 - Принцип роботи матричної факторизації [6]

Для нового користувача процес починається зі збору базової інформації, перетворення її у вектор і пошуку схожих векторів у базі даних. З часом, коли користувач взаємодіє з системою, його векторне представлення уточнюється, а рекомендації стають точнішими.

Нейронні мережі суттєво покращили рекомендаційні системи. На обчислювальному рівні вони виконують складні послідовності множень матриць з нелінійними перетвореннями. YouTube, наприклад, використовує глибокі нейронні мережі, які аналізують послідовність переглянутих відео та передбачають, що ви захочете подивитись далі.

Amazon і подібні платформи використовують гібридні підходи, які поєднують аналіз схожості користувачів (колаборативна фільтрація) та аналіз характеристик товарів (контентна фільтрація). Це допомагає подолати проблему "холодного старту" — коли в системі з'являється новий товар або користувач.

Індустрія зіткнулася з важливим компромісом між точністю та швидкістю. Spotify, YouTube та інші сервіси повинні надавати рекомендації за мілісекунди, тому використовують оптимізовані алгоритми та спеціальні структури даних, такі як індекси для пошуку найближчих сусідів.

Технічна інфраструктура включає розподілені системи для обробки даних, GPU-прискорення для навчання моделей та оптимізовані бази даних для зберігання векторних представлень. Netflix, наприклад, обробляє мільярди взаємодій щодня, щоб постійно вдосконалювати свої рекомендації.

Підсумовуючи, сучасні рекомендаційні системи покладаються на представлення користувачів та контенту у вигляді векторів, матричну факторизацію, глибинні нейронні мережі та гібридні підходи для побудови максимально релевантних рекомендацій. Завдяки цьому вони знижують “парадокс вибору”, підвищують утримання й конверсію користувачів, а також дозволяють обробляти мільярди взаємодій у реальному часі. Це призводить до того, що

технології, які використовуються при створені таких систем мають забезпечувати швидкий та точний результат і мати можливість їх подальшого розвитку.

З розвитком цифрових платформ значення рекомендаційних систем лише зростає. Вони стали невід'ємною частиною користувацького досвіду. Інтелектуальні алгоритми не лише підвищують якість обслуговування, а й сприяють зростанню прибутку компаній. Роль таких систем у майбутньому лише посилюватиметься. Їх розвиток вимагає поєднання інженерної майстерності та глибокого розуміння поведінки користувачів. Розглянемо кілька технологій, які сьогодні використовують для створення рекомендаційних систем

1.2 Аналіз актуальних технологій

1.2.1 Мови програмування

Python - високорівнева інтерпретована мова програмування з динамічною типізацією, що стала стандартом для розробки систем машинного навчання. Вона вирізняється простим синтаксисом та широкою екосистемою бібліотек для обробки та аналізу даних.

Переваги Python:

— простий та зрозумілий синтаксис, що пришвидшує розробку;
— багата екосистема бібліотек для аналізу даних та машинного навчання;
— відмінна підтримка спільноти та документація;
— гнучкість у застосуванні від прототипування до промислових систем;
— кросплатформеність;
— потужна інтеграція з різними базами даних та API;
— широкі можливості візуалізації даних.

R - мова програмування та середовище для статистичних обчислень і графіки, створена для аналізу даних та статистичних досліджень.

Переваги R:

— спеціалізація на статистичному аналізі та обробці даних;
— потужні інструменти для візуалізації даних;
— велика бібліотека статистичних методів та пакетів;
— активна наукова спільнота;
— відкритий код та безкоштовне використання;
— проста інтеграція з іншими мовами програмування;
— специфічний синтаксис, оптимізований для статистики.

1.2.2 Бібліотеки для машинного навчання

Scikit-learn - бібліотека для машинного навчання, що дає готові алгоритми для машинного навчання. Вона реалізує різноманітні алгоритми класифікації, регресії, кластеризації та зменшення розмірності.

Переваги Scikit-learn:

— простий та послідовний інтерфейс для різних типів моделей;
— вичерпна документація та приклади використання;
— оптимізовані алгоритми для швидкої обробки даних;
— інструменти для попередньої обробки даних та оцінки моделей;
— методи крос-валідації для надійної оцінки алгоритмів;
— масштабованість для середніх наборів даних.

TensorFlow - бібліотека для машинного навчання від Google. Вона забезпечує гнучкий екосистему інструментів та фреймворків для розробки та тренування моделей машинного навчання різної складності.

Переваги TensorFlow:

— підтримка графів обчислень для оптимізації складних моделей;

— масштабованість від мобільних пристроїв до кластерів серверів;

— TensorFlow Extended для повного циклу ML-розробки;

— вбудовані інструменти для візуалізації процесу навчання;

— потужна підтримка від Google та великої спільноти;

— ефективна робота з великими наборами даних;

— інтеграція з хмарними сервісами для масштабування обчислень.

PyTorch - бібліотека для машинного навчання, що забезпечує максимальну гнучкість та інтуїтивність при створенні моделей.

Переваги PyTorch:

— динамічні обчислювальні графи для більшої гнучкості;

— інтуїтивний Python-інтерфейс, близький до NumPy;

— прискорені обчислення на GPU без додаткових налаштувань;

— високий рівень читабельності коду;

— простіше відлагодження моделей;

— потужна підтримка обробки природної мови;

— активне використання в науковій спільноті та дослідженнях.

1.2.3 Бібліотеки для маніпуляції та аналізу даних

Pandas - потужна бібліотека для маніпуляції та аналізу даних у Python. Вона надає структури даних та інструменти для ефективної роботи з табличними та часовими рядами.

Переваги Pandas:

— структури даних DataFrame і Series для ефективної обробки;

— потужні інструменти для очищення та трансформації даних;

— вбудовані методи для обробки пропущених значень;

— розвинені можливості для агрегації та групування даних;

— інтеграція з різними форматами файлів;

— гнучкі інструменти для злиття та об'єднання даних;

— оптимізована продуктивність для великих наборів даних.

NumPy - фундаментальний пакет для наукових обчислень у Python, що забезпечує підтримку для багатовимірних масивів та матричних операцій.

Переваги NumPy:

— швидкі та оптимізовані n-вимірні масиви для чисельних обчислень;

— розширені можливості для математичних операцій з масивами;

— інтеграція з низькорівневими мовами для підвищення продуктивності;

— підтримка трансляції (broadcasting) для операцій з масивами різних розмірів;

— повний набір функцій лінійної алгебри;

— генератори випадкових чисел для стохастичних алгоритмів;

— інтеграція з іншими науковими бібліотеками екосистеми Python.

1.2.4 Бібліотеки візуалізації даних

Matplotlib - базова бібліотека візуалізації для Python, що забезпечує створення статичних, інтерактивних та анімованих візуалізацій у різноманітних форматах.

Переваги Matplotlib:

— високий рівень контролю над усіма елементами графіків;

— широкий вибір типів діаграм та графіків;

— можливість створення складних багатопанельних візуалізацій;

— підтримка різних бекендів для різних середовищ виконання;

— експорт у різні формати;

— висока кастомізація для наукових публікацій;

— стабільність та надійність завдяки довгій історії розвитку.

Seaborn - бібліотека для візуалізації даних, побудована на основі Matplotlib, що спеціалізується на статистичних візуалізаціях.

Переваги Seaborn:

— привабливі візуалізації з мінімальним налаштуванням;
— спеціалізовані графіки для статистичного аналізу;
— вбудовані теми для покращення естетики графіків;
— інтеграція з pandas DataFrames;
— автоматична візуалізація статистичних взаємозв'язків;
— вбудована підтримка регресійних моделей;
— спрощений синтаксис для складних візуалізацій.

1.2.5 Веб-фреймворки

FastAPI - сучасний, високопродуктивний веб-фреймворк для розробки API на Python, оптимізований для швидкості виконання та розробки. [16] Зараз його можна легко зустріти в проектах.

Переваги FastAPI:

— висока продуктивність завдяки асинхронній підтримці;
— автоматична валідація даних через типізацію Python;
— автоматична генерація документації API (Swagger/OpenAPI);
— вбудована підтримка JSON Schema;
— підтримка WebSockets для реактивних застосунків;
— зручна обробка залежностей та middleware.

Flask - легкий та гнучкий мікрофреймворк для веб-розробки на Python, що надає основні інструменти для створення веб-додатків.

Переваги Flask:

— мінімалістичний дизайн з можливістю розширення;
— простота освоєння для швидкого прототипування;
— гнучкість у виборі компонентів та архітектури;
— багата екосистема розширень;

— добре документований та підтримуваний;

— сумісність із більшістю веб-серверів;

— легка інтеграція з різними шаблонізаторами.

1.2.6 Технології для розгортання додатку

Docker - платформа з відкритим кодом, яка автоматизує розгортання, масштабування та управління додатками за допомогою контейнеризації.

Переваги Docker:

— ізоляція середовища виконання в контейнерах;

— портативність між різними обчислювальними середовищами;

— ефективне використання ресурсів порівняно з віртуальними машинами;

— швидке розгортання та масштабування додатків;

— стандартизований формат для упаковки програмного забезпечення;

— спрощена конфігурація через декларативні Dockerfile;

— велика бібліотека готових образів для різних технологій.

Kubernetes - відкрита система для автоматизованого розгортання, масштабування та управління контейнеризованими додатками.

Переваги Kubernetes:

— оркестрація контейнерів для складних розподілених систем;

— автоматичне масштабування на основі навантаження;

— самовідновлення при збоях компонентів;

— балансування навантаження та виявлення сервісів;

— автоматизоване розгортання та відкат оновлень;

— управління конфігурацією та секретами;

— гнучка модель розширення для різних середовищ та провайдерів.

1.3 Висновки до розділу

У цьому розділі ми детально проаналізували сучасні підходи до побудови рекомендаційних систем — від колаборативної та контентної фільтрації до гібридних моделей із векторним поданням даних, матричною факторизацією та глибинними нейронними мережами — та показали, як їхня комбінація підвищує задоволеність користувачів і бізнес-показники.

Окрім теоретичних основ, ми розглянули ключові інструменти для реалізації таких систем: Python та R як основні мови програмування, бібліотеки машинного навчання (Scikit-learn, TensorFlow, PyTorch), засоби обробки та аналізу даних (NumPy, Pandas), візуалізації (Matplotlib, Seaborn), веб-фреймворки (FastAPI, Flask) й інструменти розгортання (Docker, Kubernetes). Це створює міцну технологічну основу для подальшої розробки високопродуктивних рекомендаційних сервісів.

2 АНАЛІЗ ОБРАНИХ ТЕХНОЛОГІЙ

2.1 Технології, використані для рекомендаційної моделі

2.1.1 Python як основа розробки

Python є оптимальним вибором для розробки рекомендаційних систем завдяки своїй простоті, потужності та широкому спектру бібліотек для штучного інтелекту. Це дозволяє створювати гнучкі та масштабовані рішення, які легко інтегруються з іншими компонентами системи. Для нашого проекту Python забезпечує:

— Швидку розробку прототипів завдяки лаконічному синтаксису

— Доступ до спеціалізованих бібліотек машинного навчання та аналізу даних

— Легку інтеграцію з веб-сервісами та базами даних

— Ефективну обробку текстових даних, що критично для аналізу метаданих фільмів

Екосистема Python’а дуже цінується серед інженерів машинного навчання. Такі бібліотеки, як NumPy, Pandas, Scikit-learn та TensorFlow надають потрібні інструменти для аналізу даних та побудові прогнозних моделей.

У контексті рекомендаційної системи для кіносервісів Python дозволяє одночасно працювати з різними типами даних, включаючи текстові описи, числові рейтинги та категоріальні ознаки (жанри, актори), створюючи цілісний підхід до аналізу вподобань користувачів.

Крім того, Python активно підтримується спільнотою, що забезпечує доступ до великої кількості навчальних матеріалів і готових рішень. Це значно пришвидшує процес розробки та полегшує вирішення типових задач.

2.1.2 Pandas та NumPy для управління даними

Бібліотеки Pandas та Numpy критично важливі для нашої рекомендаційної системи:

— Pandas забезпечує зручну роботу з табличними даними про фільми та користувацькі оцінки

— NumPy підтримує ефективні операції з векторами та матрицями для обчислення подібності

— Разом вони дозволяють легко інтегрувати дані з різних джерел

— Забезпечують швидке фільтрування та агрегацію даних

Ці бібліотеки дозволяють нам ефективно працювати як з метаданими фільмів (жанри, актори, режисери), так і з користувацькою активністю (оцінки та перегляди), об'єднуючи ці дані для створення точних персоналізованих рекомендацій.

Pandas і NumPy дозволяють нам будувати єдину узгоджену структуру даних, яка охоплює як характеристики контенту, так і поведінкові патерни користувачів. Завдяки цьому ми можемо ефективно виконувати попередню обробку даних, виділяти ключові атрибути та готувати матриці для подальшого аналізу. Наприклад, на основі оцінок та жанрових уподобань легко обчислюється схожість між фільмами або користувачами. Це забезпечує основу для реалізації як контентно-орієнтованих, так і колаборативних підходів. Крім того, гнучкість цих бібліотек дозволяє швидко адаптувати модель до нових даних та змін у вподобаннях користувачів, що є критичним для підтримання актуальності рекомендацій.

Використання цих технологій буде першою сходинкою у нашому проекті, після який підхід нам використати у майбутньому.

2.1.3 Scikit-learn для використання алгоритмів ML

Scikit-learn є наріжним каменем традиційного машинного навчання у Python-екосистемі. Важливість цієї бібліотеки важко переоцінити з таких причин:

Вона представляє єдиний узгоджений інтерфейс для різних типів алгоритмів:

а) Класифікація (SVM, Випадковий ліс)

б) Кластеризація (K-сусіди, DBSCAN, Ієрархічний кластеринг)

в) Регресія

г) Зниження розмірності (PCA, t-SNE, UMAP)

Scikit-learn реалізує повний конвеєр машинного навчання:

— Препроцесинг даних

— Обробка відсутніх значень

— Масштабування ознак

— Кодування категоріальних змінних

— Генерація поліноміальних ознак

Відбір ознак та інженерія

— Вибір інформативних ознак (SelectKBest, RFE)

— Автоматична генерація похідних ознак

— Зниження розмірності для боротьби з прокляттям розмірності

Навчання моделей та оцінка

— Різноманітні метрики оцінки (accuracy, precision, recall, F1)

— Крос-валідація для надійної оцінки продуктивності

— Настройка гіперпараметрів через GridSearch та RandomizedSearch

Важливість scikit-learn також полягає у її інтеграції з іншими бібліотеками екосистеми Python, що дозволяє створювати комплексні рішення машинного

навчання, поєднуючи класичні алгоритми з глибоким навчанням, а кількість алгоритмів не маленька, що представлено на рисунку 2.1.

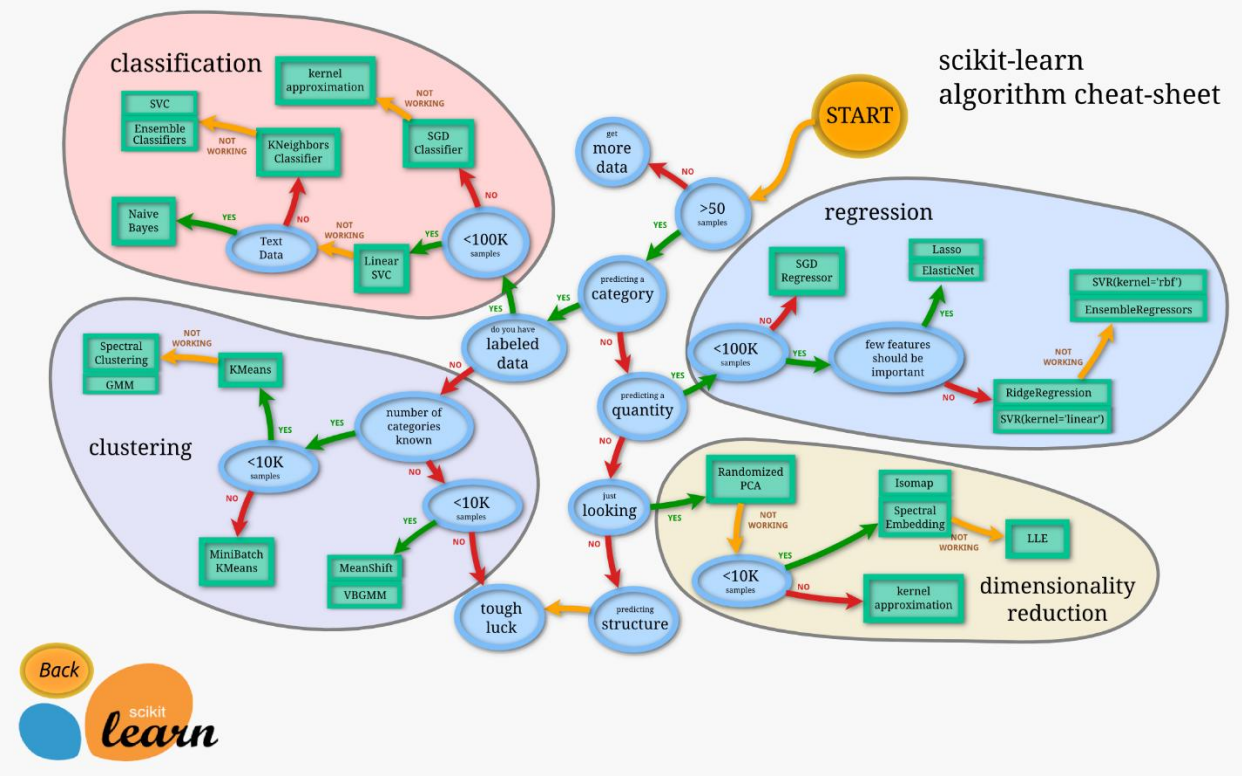


Рисунок 2.1 - Алгоритми представлені у scikit-learn [7]

2.1.4 Matplotlib та Seaborn для візуалізації даних

У сфері машинного навчання візуалізація даних виконує подвійну функцію: вона є як інструментом аналізу для дослідників, так і засобом комунікації результатів для кінцевих користувачів.

Matplotlib забезпечує низькорівневий контроль над візуалізаціями, що робить його незамінним для:

а) дослідницького аналізу даних:

1. Розподіли ознак та виявлення викидів
2. Візуалізація матриць кореляції
3. Дослідження часових рядів та трендів

б) оцінки моделей машинного навчання

1. Криві навчання для виявлення перенавчання
2. ROC-криві та PR-криві для бінарних класифікаторів
3. Матриці помилок для оцінки точності класифікації

Matplotlib є важливим аспектом нашої роботи, він допомагатиме побачити проблеми в наших даних і різні закономірності в них. Приклад графіку представлено на рисунку 2.2

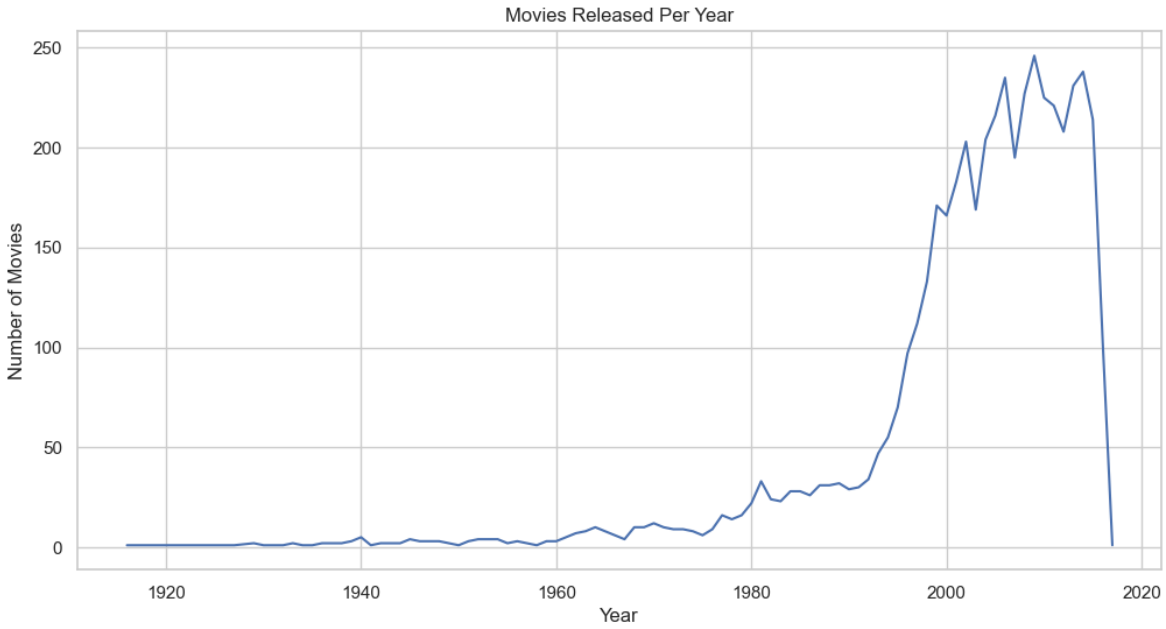


Рисунок 2.2 – Приклад графіку, створений Matplotlib

Seaborn розширює можливості Matplotlib, додаючи статистичний контекст до візуалізацій, наприклад:

— Діаграми розсіювання з регресійними лініями
— Багатовимірні візуалізації з розбивкою за категоріями
— Кластерні теплові карти для виявлення структур у даних

Також важливими є інтерактивні бібліотеки візуалізації. Вони дозволяють створювати динамічні представлення складних моделей та результатів.

2.1.5 FastAPI: перший крок до розгортання моделі

FastAPI є сучасним високопродуктивним веб-фреймворком для Python, що набув критичної важливості у сфері машинного навчання через свою здатність ефективно переводити моделей машинного навчання з дослідницького середовища у продакшн. Ключовим аспектом важливості FastAPI є продуктивність на рівні промислових стандартів.

FastAPI забезпечує надзвичайну швидкодію:

— Асинхронна обробка запитів для ефективного використання ресурсів

— Економне споживання пам'яті при обслуговуванні багатьох запитів

FastAPI органічно поєднується з інструментами ML-екосистеми:

— Безпосередня підтримка моделей scikit-learn, PyTorch, TensorFlow [8]

— Проста інтеграція з системами моніторингу моделей

— Можливість паралельного обслуговування кількох версій ML-моделей

— Сумісність з інструментами управління артефактами ML (MLflow, DVC)

FastAPI автоматично валідує вхідні дані та генерує типи відповідей, що критично важливо для роботи з ML-моделями:

— Автоматична перевірка структури вхідних даних

— Конвертація типів даних відповідно до очікувань моделі

— Детальні повідомлення про помилки для неправильних входів

— Підтримка складних структур даних

Однією з найцінніших особливостей FastAPI є автоматична генерація інтерактивної документації API:

— Swagger UI для тестування ендпоінтів

— ReDoc для детального опису API

— Автоматичне оновлення документації при змінах коду

— Експорт OpenAPI схем для інтеграції з іншими системами

Це особливо важливо для команд Data Science. Як виглядає алгоритм і роль FastAPI у цьому, представлено на рисунку 2.3.

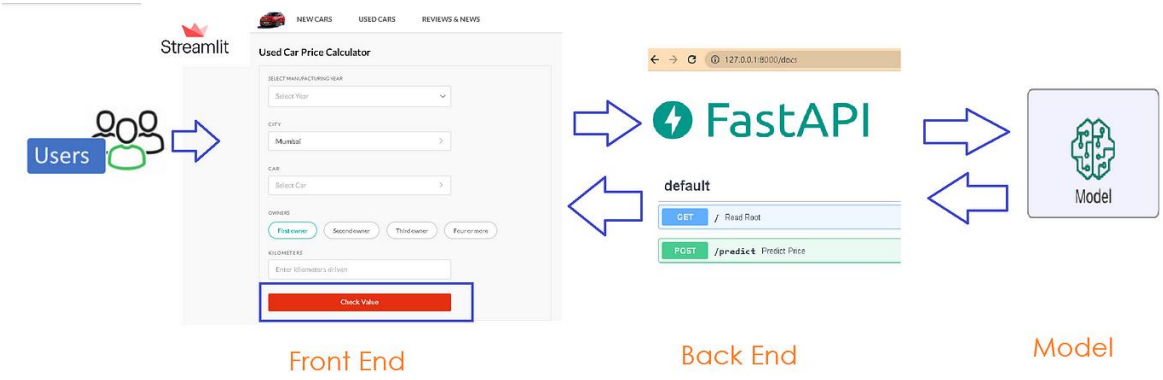


Рисунок 2.3 - Алгоритм проходження даних між користувачем та моделлю [9]

2.1.6 Docker – фінішна пряма до деплою

Docker став фундаментальним інструментом у MLOps (Machine Learning Operations) завдяки здатності вирішувати ключові проблеми розгортання моделей машинного навчання у продакшн. [10]

Одна з найбільших проблем у машинному навчанні — конфлікти між різними версіями бібліотек. Docker вирішує це через повну ізоляцію середовища: кожна модель може мати власні версії Python, NumPy, TensorFlow, що усуває конфлікти між проектами та гарантує відтворюваність результатів на будь-якій машині. Крім того, він спрощує управління складними залежностями ML-бібліотек.

На технічному рівні Docker працює шляхом створення ізольованих контейнерів, які запускаються поверх загального ядра операційної системи. Кожен контейнер має власну файлову систему, змінні середовища, мережеві налаштування та залежності, що дозволяє йому функціонувати незалежно від хост-системи або інших контейнерів. Образи Docker описуються у спеціальних Dockerfile, де чітко зазначено, які саме пакети, бібліотеки та налаштування потрібні

для запуску конкретної моделі або застосунку. Це дозволяє повністю автоматизувати розгортання, зробити його повторюваним і незалежним від середовища розробника.

Також Docker дозволяє ефективно масштабувати ML-додатки: горизонтально через реплікацію контейнерів, вертикально — завдяки гнучкому налаштуванню ресурсів, а також автоматично реагувати на зміну навантаження. Контейнери підтримують роботу з GPU, що критично для обчислювально затратних моделей.

Одна з найбільших проблем у машинному навчанні — конфлікти між різними версіями бібліотек. Docker вирішує це через повну ізоляцію середовища:

— Кожна модель може мати власні версії Python, NumPy, TensorFlow

— Відсутність конфліктів між різними проектами

— Гарантована відтворюваність результатів на різних машинах

— Простота управління складними залежностями ML-бібліотек

Також Docker дозволяє ефективно масштабувати ML-додатки:

— Горизонтальне масштабування через реплікацію контейнерів

— Вертикальне масштабування через налаштування ресурсів контейнера

— Автоматичне масштабування на основі навантаження

— Ефективне використання GPU-ресурсів у контейнерах

Docker забезпечує єдине середовище від розробки до продакшн:

— Локальна розробка в ідентичному до продакшн середовищі

— Безшовний перехід між хмарними провайдерами

— Можливість розгортання на різних операційних системах

— Стандартизація процесів розгортання в організації

Зрештою, Docker забезпечує єдине середовище від розробки до продакшн: локальне тестування відбувається в умовах, ідентичних до продакшн, а перехід між

різними хмарними платформами або операційними системами відбувається без змін у коді. Таким чином досягається стандартизація процесів розгортання в межах всієї організації. Архітектура Docker представлена на рисунку 2.4.

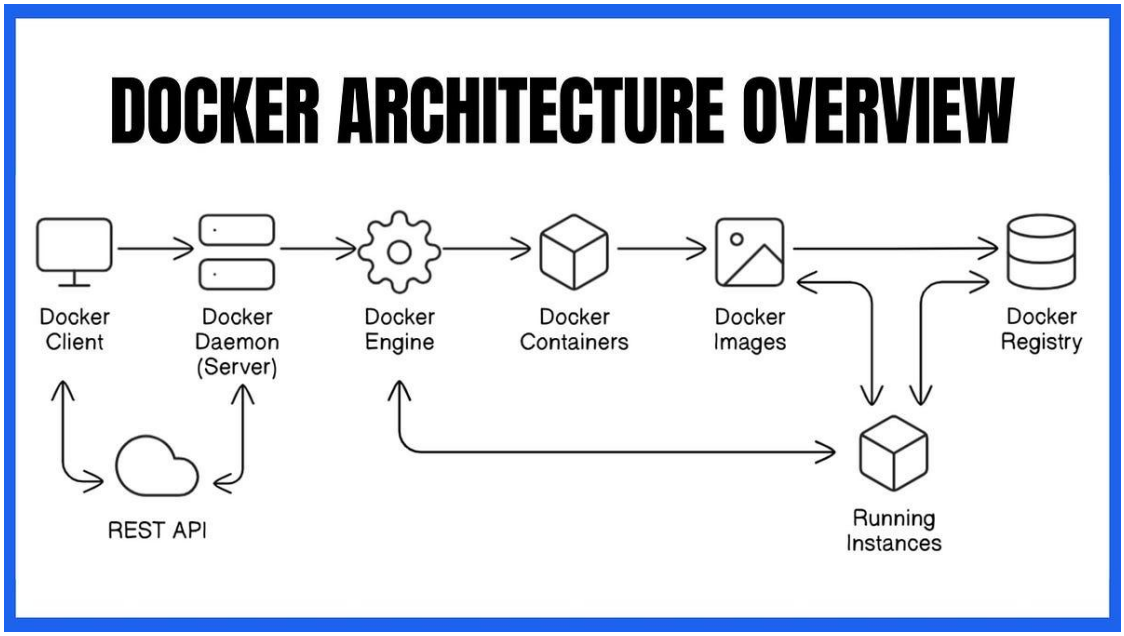


Рисунок 2.4 – Архітектура Docker [11]

2.2 Технічний аналіз методу TF-IDF

2.2.1 Опис TF-IDF

TF-IDF - це числова статистика, яка відобра жає важливість слова в документі відносно колекції документів. Наприклад кожен фільм представлений як документ через свої метадані, а алгоритм аналізує частоту появи слів та їх унікальність у всій колекції.

Косинусна подібність вимірює кут між векторами в багатовимірному просторі, де кожен вимір відповідає терміну з TF-IDF матриці. Значення коливаються від 0 (повністю різні) до 1 (ідентичні).

TF визначає, наскільки часто слово зустрічається в окремому документі. Чим частіше слово трапляється в тексті, тим більш значущим воно вважається для змісту цього документа.

Обмеженням TF є те, що показник не враховує важливість слова в межах усього корпусу документів. Наприклад, часто вживані слова на кшталт "і", "в", "це" можуть мати високі значення TF, хоча насправді не допомагають розрізняти документи.

IDF (обернена частота документів), навпаки, зменшує вагу поширених слів, які з’являються у багатьох документах, і підвищує значущість тих, що трапляються рідко. Чим рідше слово зустрічається у корпусі, тим вищою є його інформативність. [12] Формули TF та IDF представлені на рисунку 2.5.

$$TF(t, d) = \frac{\text{Number of times term t appears in document d}}{\text{Total number of terms in document d}}$$

$$IDF(t, D) = \log \frac{\text{Total number of documents in corpus D}}{\text{Number of documents containing term t}}$$

Рисунок 2.5 – Формула TF та IDF [13]

Обмеженням IDF є те, що він не враховує, наскільки часто слово використовується в конкретному документі. Слово може бути рідкісним у всій колекції (мати високий IDF), але водночас бути малозначущим для окремого тексту (низький TF).

Переваги методу TF-IDF:

— Не потребує великих обсягів користувацьких даних для початку роботи

— Прозорість алгоритму - легко зрозуміти, чому конкретний фільм був рекомендований

— Відсутність проблеми "холодного старту" для нових товарів

— Стабільні результати, які не змінюються в залежності від поведінки інших користувачів

— Швидкість обчислень після попередньої обробки даних

Технічні обмеження:

— Складність у виявленні латентних зв'язків між контентом

— Схильність до рекомендування занадто схожого контенту

— Залежність від якості та повноти метаданих

— Проблеми з масштабуванням при великих обсягах даних

— Неможливість врахування контекстуальних факторів та персональних уподобань

Матриця косинусної подібності має розмірність n×n (n - кількість фільмів). Це означає, що для 10,000 фільмів потрібно зберігати 100 мільйонів значень подібності, що може створювати проблеми з пам'яттю. З рішенням цієї проблеми можуть допомогти:

— Застосування Locality Sensitive Hashing (LSH) для приблизного пошуку найближчих сусідів

— Кластеризація контенту для зменшення простору пошуку

— Попередня фільтрація за категоріями перед обчисленням подібності

2.2.2 Застосування у світі

Netflix, Amazon Prime Video та Spotify активно використовують контентну фільтрацію як частину своїх гібридних систем рекомендацій. Spotify аналізує метадані треків (жанр, темп, тональність) для створення плейлистів, схожих на обрані користувачем композиції. Крім метаданих, Spotify також використовує аудіоаналіз, що дозволяє враховувати такі характеристики, як енергійність, танцювальність чи інструментальність треків. Це допомагає створювати

персоналізовані добірки, які точно відповідають смакам користувача. Зібрані дані використовуються разом із вподобаннями інших слухачів зі схожими смаками для покращення точності рекомендацій. Приклад рекомендаційної системи Netflix представлений на рисунку 2.5.

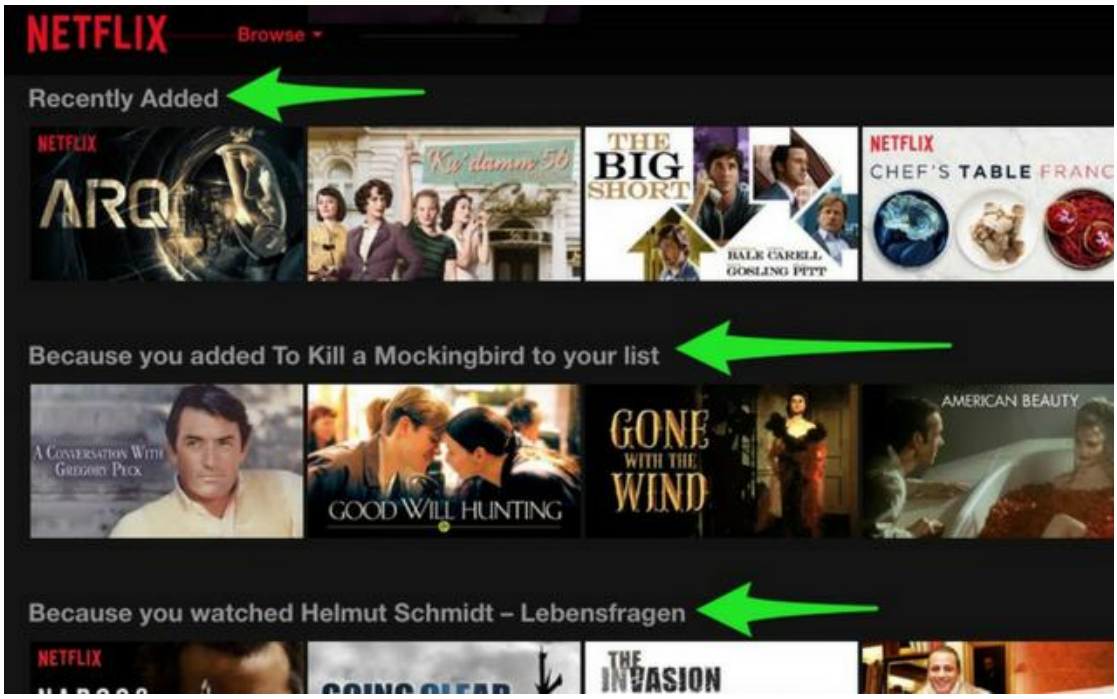


Рисунок 2.5 – Рекомендаційна система Netflix [14]

Amazon та eBay метод рекомендування товарів на основі описів товарів, категорій та характеристик. Особливо ефективно це працює для книг, де можна аналізувати жанри, авторів та ключові слова з анотацій. Приклад такого використання представлений на рисунку 2.6

Рисунок 2.6 – Приклад рекомендацій Amazon [15]

Google News, Apple News та інші агрегатори використовують TF-IDF для групування схожих статей та рекомендування новин на основі контенту раніше прочитаних матеріалів.

LinkedIn застосовує контентну фільтрацію для рекомендування вакансій, аналізуючи описи посад та профілі користувачів. Facebook використовує подібні підходи для групування постів за тематикою.

Проведений аналіз технологій та методів для розробки рекомендаційної системи демонструє комплексний підхід до вирішення задач персоналізації контенту у кіносервісах. Обрана технологічна архітектура забезпечує як надійність та масштабованість системи, так і гнучкість у подальшому розвитку.

Технологічна основа проекту базується на Python-екосистемі, що є оптимальним рішенням для машинного навчання завдяки широкому спектру спеціалізованих бібліотек та активній підтримці спільноти. Поєднання Pandas та NumPy забезпечує ефективну роботу з великими обсягами структурованих даних, в той час як Scikit-learn надає необхідний інструментарій для реалізації алгоритмів машинного навчання. Візуалізаційні компоненти Matplotlib та Seaborn дозволяють проводити якісний аналіз даних та моніторинг ефективності моделі.

FastAPI гарантує високошвидкісну обробку запитів з автоматичною валідацією даних та генерацією документації, що критично важливо для інтеграції з іншими компонентами системи. Docker вирішує фундаментальні проблеми відтворюваності середовища та усунення конфліктів залежностей, забезпечуючи стабільність роботи від розробки до продакшн.

Метод TF-IDF демонструє баланс між простотою реалізації та ефективністю результатів. Його основні переваги включають відсутність проблеми холодного старту, прозорість алгоритму та незалежність від користувацьких даних. Проте виявлені технічні обмеження, такі як схильність до рекомендування занадто схожого контенту та складність масштабування, вказують на необхідність подальшого розвитку системи у напрямку гібридних підходів.

Практичне застосування аналогічних технологій провідними компаніями (Netflix, Amazon, Spotify) підтверджує правильність обраного технологічного стеку та методологічного підходу. Успішна імплементація контентної фільтрації у цих системах демонструє масштабованість та комерційну ефективність запропонованого рішення.

Перспективи розвитку системи включають інтеграцію колаборативної фільтрації для створення гібридної моделі, впровадження технік глибокого навчання для аналізу складних патернів у даних, та оптимізацію алгоритмів для роботи з великими обсягами інформації через застосування розріджених матриць та кластеризації.

Загалом, представлена технологічна архітектура та методологічний підхід забезпечують надійну основу для створення ефективної рекомендаційної системи, яка може бути успішно масштабована та адаптована до специфічних потреб кіносервісу.

2.3 Висновки по розділу

Обрані технології забезпечують нам оптимально продуктивності, масштабованості разом з простоти розробки. Метод TF-IDF показує ефективне рішення проблем, про які ми говорили раніше, хоча й досі маємо проблему з різноманіттям. На прикладі інших успішних компаній можемо бачимо, наскільки ці підходи дають гарний результат.

3 РОЗРОБКА РЕКОМЕНДАЦІЙНОЇ МОДЕЛІ ТА РЕЗУЛЬТАТИ

3.1 Вибір та підготовка набору даних

Для розробки рекомендаційної системи в дипломній роботі було обрано датасет TMDB Movie Metadata з платформи Kaggle, який представляє собою комплексну колекцію інформації про 5000 фільмів з популярної бази даних The Movie Database (TMDB).[16]

Датасет складається з двох взаємопов'язаних файлів, що разом формують повну картину кінематографічного контенту. Перший файл містить основні метадані фільмів, включаючи фінансову інформацію, технічні характеристики та дані про популярність. Другий файл доповнює цю інформацію детальними відомостями про творчий склад, що дозволяє створювати рекомендації на основі участі конкретних акторів, режисерів та інших учасників кіновиробництва.

Перший файл містить наступні поля:

— budget - бюджет фільму в доларах США, важливий показник для аналізу комерційного потенціалу
— genres - список жанрів у форматі JSON, що дозволяє категоризувати фільми за тематикою
— homepage - офіційний веб-сайт фільму, що може використовуватися для додаткової інформації
— id - унікальний ідентифікатор фільму в базі даних TMDB, ключове поле для зв'язку таблиць
— keywords - ключові слова у форматі JSON, що описують основні елементи сюжету та тематику
— original_language - мова оригінального фільму, важлива для аналізу географічного розподілу
— original_title - оригінальна назва фільму мовою виробництва

— overview - короткий опис сюжету, що може використовуватися для текстового аналізу
— popularity - показник популярності фільму на платформі TMDB
— production_companies - список виробничих компаній у форматі JSON
— production_countries - країни виробництва у форматі JSON
— release_date - дата виходу фільму в прокат
— revenue - касові збори фільму в доларах США
— runtime - тривалість фільму в хвилинах
— spoken_languages - мови озвучення у форматі JSON
— status - статус фільму (випущений, в роботі тощо)
— tagline - рекламний слоган фільму
— title - назва фільму (може відрізнятися від оригінальної)
— vote_average - середня оцінка користувачів
— vote_count - кількість голосів користувачів

Другий файл містить наступні поля:

— movie_id - ідентифікатор фільму, що відповідає полю id в основному файлі
— title - назва фільму для перевірки відповідності
— cast - детальна інформація про акторський склад у форматі JSON, включаючи імена, ролі, стать та інші характеристики
— crew - інформація про знімальну групу у форматі JSON, включаючи режисерів, продюсерів, сценаристів та інших учасників

Процес обробки розпочинається з створення спеціалізованих функцій для завантаження кожного типу файлів, що забезпечує модульність та повторне використання коду.

Функція “load_tmdb_movies” відповідає за завантаження основного файлу з метаданими фільмів. Вона виконує декілька критично важливих операцій з очищення та стандартизації даних. Перш за все, функція перетворює текстові дати релізу у правильний формат datetime, а потім конвертує їх у об'єкти дати для зручності подальшої обробки. Особливо важливою є обробка JSON-стовпців, які

містять структуровані дані у вигляді текстових рядків. Ці стовпці включають жанри, ключові слова, країни виробництва, виробничі компанії та мови озвучення, які після парсингу стають доступними для детального аналізу. Дана функція представлена на рисунку 3.1.

```
def load_tmdb_movies(path):
    df = pd.read_csv(path)
    df['release_date'] = pd.to_datetime(df['release_date']).apply(lambda x: x.date())
    json_columns = ['genres', 'keywords', 'production_countries', 'production_companies', 'spoken_languages']
    for column in json_columns:
        df[column] = df[column].apply(json.loads)
    return df
```

Рисунок 3.1 – Функція “load_tmdb_movies”

Аналогічно функція load_tmdb_credits обробляє файл з інформацією про творчий склад. Вона парсить JSON-дані в стовпцях cast та crew, перетворюючи їх зі строкового формату у структуровані об'єкти Python, що дозволяє легко витягувати конкретну інформацію про акторів, режисерів та інших учасників кіновиробництва. Дана функція представлена на рисунку 3.2.

```
def load_tmdb_credits(path):
    df = pd.read_csv(path)
    json_columns = ['cast', 'crew']
    for column in json_columns:
        df[column] = df[column].apply(json.loads)
    return df
```

Рисунок 3.2 – Функція “load_tmdb_credits”

Після завантаження основних даних код виконує додаткову обробку інформації про акторський склад. Зокрема, він витягує дані про головного актора кожного фільму, беручи першого актора зі списку cast. Це включає як ім'я актора, так і інформацію про його стать, що може бути корисним для аналізу гендерного представлення в кіноіндустрії та створення відповідних рекомендацій. Реалізація цієї обробки представлена на рисунку 3.3

```
credits['gender_of_lead'] = credits.cast.apply(lambda x: safe_access(x, [0, 'gender']))
credits['lead'] = credits.cast.apply(lambda x: safe_access(x, [0, 'name']))
credits['gender_of_lead'].value_counts()
```

Рисунок 3.3 Додаткова обробка складу

Ключовим етапом обробки є об'єднання двох датасетів в один комплексний набір даних. Це здійснюється за допомогою операції merge по ідентифікаторах фільмів, що створює повну картину з усією доступною інформацією. Результуючий датасет містить як основні характеристики фільмів, так і детальну інформацію про їх творчий склад.

У результаті обробки та об'єднання двох вихідних файлів було створено комплексний датасет, що містить 27 колонок з повною інформацією про кожен фільм. Цей об'єднаний датасет включає:

Базові характеристики фільмів:

— budget - фінансовий бюджет фільму
— genres - оброблені жанри у форматі Python-об'єктів
— homepage - офіційні веб-сайти фільмів
— id - унікальні ідентифікатори з основного датасету
— keywords - структуровані ключові слова
— original_language - мови оригінальних версій
— original_title - оригінальні назви фільмів
— overview - описи сюжетів для контентного аналізу
— popularity - показники популярності на платформі
— production_companies - оброблена інформація про виробників
— production_countries - країни виробництва у структурованому вигляді
— release_date - стандартизовані дати випуску
— revenue - касові збори для аналізу комерційного успіху
— runtime - тривалість фільмів
— spoken_languages - мови озвучення у структурованому форматі
— status - статуси випуску фільмів
— tagline - рекламні слогани

Назви та оцінки:

— title_x - назви фільмів з основного датасету (movies)
— title_y - назви фільмів з датасету кредитів (credits)

— vote_average - середні оцінки користувачів
— vote_count - кількість голосів для аналізу достовірності рейтингів

Інформація про творчий склад:

— movie_id - ідентифікатори для зв'язку з кредитами
— cast - структурована інформація про весь акторський склад
— crew - дані про знімальну групу

Додаткові характеристики головних акторів:

— gender_of_lead - стать головного актора (витягнута з першого елементу cast)
— lead - ім'я головного актора фільму

Після створення об'єднаного датасету було проведено другий етап очищення даних, спрямований на підготовку оптимального набору характеристик для рекомендаційної системи. Цей процес включав аналіз пропущених значень, видалення надлишкових колонок та трансформацію даних у більш зручний для аналізу формат.

Спочатку було проведено системний аналіз пропущених значень у датасеті, що дозволило ідентифікувати колонки з найбільшою кількістю відсутніх даних. На основі цього аналізу прийнято рішення про видалення колонок з критично високою кількістю пропущених значень, включаючи “homepage”, “tagline”, “gender_of_lead” та “lead”, оскільки їх збереження могло б негативно вплинути на якість рекомендацій.

Також було видалено технічні колонки, що дублювали інформацію або були непотрібними для аналізу. До них належать “Unnamed: 0” - індексна колонка, “title_x”, “title_y” та “movie_id”, які є дублікатами існуючих колонок . Це рішення спростило структуру датасету без втрати корисної інформації. Реалізація в коді представлена на рисунку 3.4.

```
columns_to_drop = [
    'homepage',
    'tagline',
    'Unnamed: 0',
    'title_x',
    'title_y',
    'movie_id',
    'gender_of_lead',
    'lead'
]

df.drop(columns=columns_to_drop, inplace=True)
```

Рисунок 3.4 – Видалення зайвих колонок

Для роботи з фінансовими показниками було застосовано стратегію видалення рядків, де одночасно відсутні як бюджет, так і касові збори, оскільки такі записи мають обмежену цінність для аналізу комерційної успішності фільмів. Для деяких конкретних фільмів було здійснено ручне заповнення відсутніх значень тривалості на основі достовірних джерел.

Особливу увагу приділено обробці дат випуску фільмів. Рядки з відсутніми датами було видалено, а решту перетворено у стандартний формат datetime. На основі дат випуску створено додаткові характеристики - рік, місяць та день випуску, що розширює можливості для часового аналізу та сезонних рекомендацій. Реалізація представлена на рисунку 3.5.

```
df.dropna(subset=['release_date'], inplace=True)
df['release_date'] = pd.to_datetime(df['release_date'], errors='coerce')

df['release_year'] = df['release_date'].dt.year
df['release_month'] = df['release_date'].dt.month
df['release_day'] = df['release_date'].dt.day
```

Рисунок 3.5 – Розділення дати на частини

Для зручності інтерпретації та візуалізації фінансові показники було перетворено з доларів у мільйони доларів, створивши нові колонки “budget_millions” та “revenue_millions”. Це спрощує сприйняття великих числових

значень та покращує читабельність аналітичних звітів. Реалізація представлена на рисунку 3.6.

```
df['budget_millions'] = df['budget'] / 1e6
df['revenue_millions'] = df['revenue'] / 1e6
```

Рисунок 3.6 – Нормалізація фінансових показників

Критично важливим етапом стала обробка JSON-структур, що містили списки об'єктів з різними характеристиками. Для колонок “genres”, “keywords”, “production_companies” та “spoken_languages” було створено функцію “extract_names”, яка витягує лише назви елементів зі складних JSON-структур, перетворюючи їх у прості списки рядків. Це значно спрощує подальшу роботу з категоріальними даними та дозволяє ефективно використовувати їх для контентної фільтрації. Реалізація у коді представлена на рисунку 3.7.

```
def extract_names(column):
    return [item['name'] for item in eval(column)] if pd.notnull(column) else []
```

```
df['genres'] = df['genres'].apply(extract_names)
df['keywords'] = df['keywords'].apply(extract_names)
df['production_companies'] = df['production_companies'].apply(extract_names)
df['spoken_languages'] = df['spoken_languages'].apply(extract_names)
```

Рисунок 3.7 – Обробка JSON-структур

На завершальному етапі датасет було відфільтровано, залишивши лише фільми зі статусом "Released", що забезпечує роботу виключно з офіційно випущеними картинами. Також було видалено колонки, що стали непотрібними після трансформації або мали обмежену цінність для рекомендаційної системи, включаючи оригінальні значення бюджету та касових зборів, дати випуску, ідентифікатори, описи сюжетів та статуси.

У результаті цього комплексного процесу очищення було отримано оптимізований датасет, що містить лише найбільш релевантні та якісно оброблені характеристики фільмів. Цей датасет характеризується високою щільністю

корисної інформації, мінімальною кількістю пропущених значень та зручною структурою для розробки алгоритмів рекомендацій.

Після завершення всіх етапів очищення та трансформації було створено оптимізований датасет, що містить 17 ключових характеристик для кожного фільму. Цей фінальний датасет представляє собою збалансоване поєднання контентних, технічних та комерційних показників, необхідних для ефективної роботи рекомендаційної системи:

Контентні характеристики:

— genres - список жанрів у вигляді простих текстових назв, витягнутих з JSON-структур
— keywords - ключові слова, що описують тематику та сюжетні елементи фільму у форматі списку рядків
— original_language - мова оригінальної версії фільму
— original_title - оригінальна назва фільму мовою виробництва

Показники популярності та оцінок:

— popularity - числовий показник популярності фільму на платформі TMDB
— vote_average - середня оцінка користувачів
— vote_count - загальна кількість оцінок користувачів

Виробничі характеристики:

— production_companies - список назв виробничих компаній у текстовому форматі
— production_countries - країни виробництва у вигляді списку назв країн
— spoken_languages - мови озвучення фільму як список текстових позначень

Технічні параметри:

— runtime - тривалість фільму в хвилинах
— cast - структурована інформація про акторський склад
— crew - дані про знімальну групу в JSON-форматі

Часові характеристики:

— release_year - рік випуску фільму
— release_month - місяць випуску
— release_day - день випуску в місяці

Фінансові показники:

— budget_millions - бюджет фільму в мільйонах доларів США
— revenue_millions - касові збори в мільйонах доларів США

3.2 Дослідницький аналіз даних

3.2.1 Середній дохід від середнього бюджету

Дослідницький аналіз даних є критично важливим етапом для розуміння структури та закономірностей у наборі даних про фільми. На основі проведеного аналізу виявлено ключові взаємозв'язки між характеристиками фільмів, які впливають на їх комерційний успіх та популярність серед глядачів.

Аналіз розподілу бюджету та доходів фільмів показує значну варіативність у фінансових показниках кіноіндустрії. З графіка "Budget vs. Revenue", який представлений на рисунку 3.8 видно, що більшість фільмів концентруються в діапазоні низького та середнього бюджету (до 100 мільйонів доларів), при цьому спостерігається позитивна кореляція між бюджетом та доходами.

Розподіл доходів демонструє експоненціальний характер, де невелика кількість фільмів генерує надзвичайно високі доходи (понад 2000 мільйонів доларів), тоді як основна маса фільмів отримує помірні касові збори. Це підтверджує принцип Парето в кіноіндустрії, де 20% фільмів генерують 80% загальних доходів.

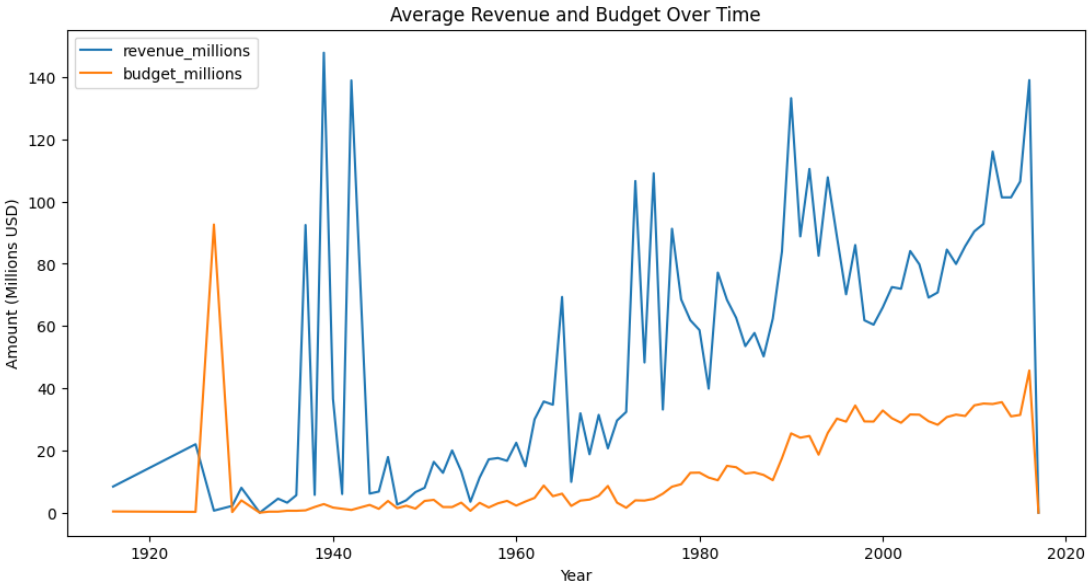


Рисунок 3.8 – Середній дохід від середнього бюджету

3.2.2 Найуспішніші компанії за доходами

Аналіз найуспішніших виробничих компаній за загальними доходами, графік якого представлено на рисунку 3.9, виявляє чітку ієрархію в кіноіндустрії. Warner Bros. займає лідируючу позицію з доходами близько 49,000 мільйонів доларів, що значно перевищує показники конкурентів. Universal Pictures посідає друге місце з доходами приблизно 43,000 мільйонів доларів, а Paramount Pictures замикає трійку лідерів з 41,000 мільйонів доларів.

Цікавою особливістю є відносно рівномірний розподіл між топ-4 компаніями (Warner Bros., Universal Pictures, Paramount Pictures, та Twentieth Century Fox Film Corporation з ~39,000 млн), після чого спостерігається поступове зниження доходів у наступних компаній. Walt Disney Pictures та Columbia Pictures демонструють схожі показники близько 29,000 мільйонів доларів кожна.

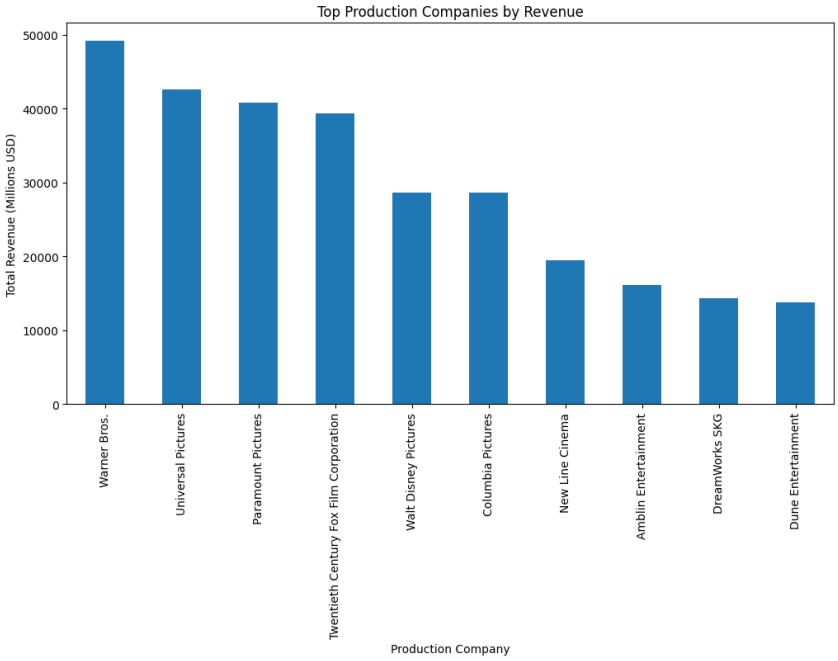


Рисунок 3.9 – Найуспішніші компанії за прибутком

Менші студії, такі як New Line Cinema, Amblin Entertainment, DreamWorks SKG та Dune Entertainment, показують значно нижчі доходи в діапазоні 14,000-19,000 мільйонів доларів, що підтверджує концентрацію ринкової влади серед великих голлівудських студій.

3.2.2 Розподіл жанрів по кіноіндустрії

Аналіз найпопулярніших жанрів, графік якого представлено на рисунку 3.10 демонструює чіткі переваги аудиторії та ринкові тенденції. Драма домінує з величезним відривом, налічуючи понад 2,300 фільмів, що підтверджує універсальну привабливість цього жанру для різних демографічних груп.

Комедія посідає друге місце з приблизно 1,700 фільмами, що відображає постійний попит на розважальний контент. Трилер замикає трійку лідерів з ~1,280 фільмами, демонструючи популярність напруженого та захоплюючого контенту.

Екшн-фільми займають четверту позицію з ~1,180 проектами, що корелює з їх комерційною привабливістю та глобальним попитом. Романтичні фільми (~920) та пригодницькі (~800) також показують стабільну присутність на ринку.

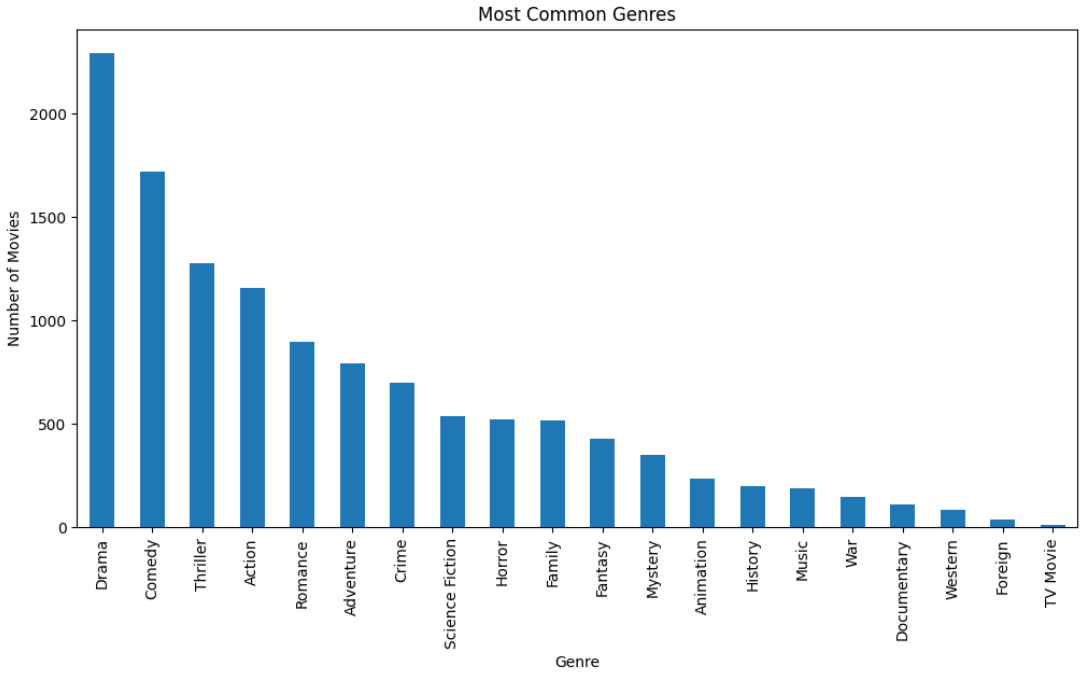


Рисунок 3.10 – Розподіл по жанрам

Спеціалізовані жанри демонструють нішеву, але стійку популярність:

— кримінальні фільми (~700),
— наукова фантастика (~540),
— жахи (~520),
— сімейні фільми (~520),

Цей розподіл вказує на домінування універсальних жанрів, які апелюють до широкої аудиторії, при збереженні ніш для спеціалізованого контенту, що важливо врахувати при розробці рекомендаційної системи для забезпечення різноманітності пропозицій.

Ці висновки формують основу для вибору архітектури рекомендаційної системи та визначення вагових коефіцієнтів для різних характеристик фільмів.

3.3 Створення рекомендаційної моделі

3.3.1 Підготовка та інтеграція даних

Для створення рекомендаційної системи фільмів було обрано підхід на основі контентної фільтрації, який аналізує характеристики фільмів для знаходження схожих елементів. Цей метод особливо ефективний для рекомендації фільмів, оскільки дозволяє враховувати множинні атрибути контенту: жанри, ключові слова, акторський склад та команду виробництва.

Основна перевага обраного підходу полягає в його здатності надавати рекомендації навіть для нових фільмів, які ще не мають історії переглядів користувачів, що вирішує проблему "холодного старту". Крім того, система може пояснити свої рекомендації, посилаючись на конкретні спільні характеристики фільмів.

Першим кроком у створенні системи стало об'єднання ключових характеристик кожного фільму в єдиний текстовий опис (метадані). Процес реалізації продемонстровано на рисунку 3.11.

```
df['metadata'] = df.apply(lambda x: ' '.join(x['genres']) + ' ' +
                                    ' '.join(x['keywords']) + ' ' +
                                    ' '.join(x['cast']) + ' ' +
                                    ' '.join(x['crew']), axis=1)

df[['original_title', 'metadata']].head()
```

Рисунок 3.11 – Об’єднання метаданих

Ця операція створює текстовий профіль для кожного фільму, який включає:

— Жанри: Основні категорії фільму

— Ключові слова: Специфічні теми та мотиви фільму

— Акторський склад: Головні виконавці ролей

— Команда виробництва: Режисери, продюсери та інші ключові учасники

Такий підхід дозволяє системі враховувати як жанрові характеристики, так і специфічні елементи, що роблять кожен фільм унікальним. Наприклад, два фільми можуть належати до одного жанру, але мати різних режисерів або акторів, що впливатиме на їх схожість.

Результатом цього процесу є структурований датасет, де кожен фільм представлений своєю назвою та відповідними метаданими. Це забезпечує основу для подальшого векторного аналізу та розрахунку схожості між фільмами.

3.3.2 Векторизація текстових даних

Для перетворення текстових описів фільмів у числові вектори використовується алгоритм TF-IDF. Цей підхід ефективно оцінює важливість кожного слова в контексті всієї колекції фільмів.

Принцип роботи TF-IDF:

— Term Frequency (TF): Визначає частоту появи терміна в описі конкретного фільму
— Inverse Document Frequency (IDF): Зменшує вагу термінів, які часто зустрічаються в багатьох фільмах
— Фільтрація стоп-слів: Автоматично виключає загальновживані англійські слова, які не несуть змістового навантаження

Результатом векторизації є матриця розміром (кількість_фільмів × кількість_унікальних_термінів), де кожен фільм представлений як точка в багатовимірному просторі. Розмірність цього простору залежить від кількості унікальних термінів у всіх описах фільмів.

Ця матриця дозволяє математично порівнювати фільми на основі їх характеристик, перетворюючи якісні ознаки у кількісні метрики для аналізу.

Для визначення ступеня схожості між фільмами використовується косинусна схожість, яка вимірює кут між векторами фільмів у багатовимірному просторі.

Переваги косинусної схожості:

— Незалежність від довжини векторів - фокус на напрямку, а не на величині
— Значення в діапазоні [0, 1], де 1 означає повну схожість
— Ефективність обчислень для високомірних просторів
— Стійкість до розрідженості векторів (багато нульових значень)

Результатом є квадратна матриця розміром (кількість_фільмів кількість_фільмів), де кожен елемент представляє ступінь схожості між відповідними фільмами. Діагональні елементи завжди дорівнюють 1 (фільм повністю схожий сам на себе).

3.3.3 Реалізація рекомендаційного алгоритму

Для ефективного пошуку фільмів за назвою створюється індекс. Цей індекс забезпечує швидкий доступ до позиції фільму в датасеті за його назвою,

Алгоритм рекомендацій реалізований у вигляді функції, яка представлена на рисунку 3.12.

```
def get_recommendations(title, cosine_sim=cosine_sim, top_n=10):
    idx = indices[title]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:top_n+1]
    movie_indices = [i[0] for i in sim_scores]
    return df['original_title'].iloc[movie_indices]
```

Рисунок 3.12 – Алгоритм рекомендаційної системи

Логіка роботи алгоритму:

— Ідентифікація: Знаходження позиції вхідного фільму в базі даних
— Порівняння: Отримання показників схожості з усіма іншими фільмами
— Ранжування: Сортування фільмів за ступенем схожості у спадному порядку

— Фільтрація: Виключення самого вхідного фільму з результатів

— Відбір: Повернення заданої кількості найбільш схожих фільмів

Функція підтримує налаштування кількості рекомендацій через параметр “top_n”, що дозволяє адаптувати систему під різні сценарії використання - від коротких списків (5-10 фільмів) до розширених рекомендацій (20+ фільмів).

3.3.4 Розгортання рекомендаційної моделі

На бекенді FastAPI відповідає за обробку запитів від користувача, зокрема рендеринг головної сторінки, пошук назв фільмів за частковим збігом і надання списку рекомендованих фільмів. Після запуску застосунок завантажує заздалегідь збережені моделі — TF-IDF векторизатор і матрицю косинусної схожості, а також датафрейм із метаданими фільмів. Для забезпечення стабільної роботи з назвами використовується нормалізація символів, щоб уникнути помилок через регістр, діакритичні знаки або пунктуацію.

Коли користувач вводить назву фільму, FastAPI спершу нормалізує введене значення, перевіряє його наявність у датафреймі. Якщо фільм не знайдено, сервіс намагається запропонувати користувачеві схожі назви, використовуючи алгоритм пошуку близьких відповідників. Якщо ж фільм знайдено, система обчислює найбільш схожі фільми за допомогою попередньо обчисленої матриці косинусної схожості й повертає відповідні рекомендації.

Фронтенд частина включає HTML-шаблон, у якому розташовано поле введення фільму, кнопку пошуку, блок для виводу результатів і підказки. Стилізація реалізована через Bootstrap, а логіка на JavaScript. Скрипт реалізує асинхронні запити до бекенду, обробляє введення користувача, пропонує варіанти назв у міру набору тексту та відображає результати у вигляді карток. Щоб зробити результати привабливішими, використовуються додаткові запити до TMDB API, звідки підтягуються постери, рейтинги й короткі описи фільмів.

Розгортання проєкту передбачає використання Docker. Для цього створюється Dockerfile, в якому задається образ на базі Python, встановлюються залежності з requirements.txt, копіюється весь код і запускається застосунок через Uvicorn. У підсумку, користувач отримує веб-сервіс, який дозволяє легко вводити назву фільму, бачити релевантні результати й дізнаватись про схожі фільми, представлені в привабливому інтерфейсі.

3.3.5 Тестування

Розроблений веб-сайт представляє собою мінімалістичну, але функціональну реалізацію рекомендаційної системи, що і представлено на рисунку 3.13. Центральним елементом інтерфейсу є текстове поле для введення назви фільму, яке забезпечує основну взаємодію користувача з системою.

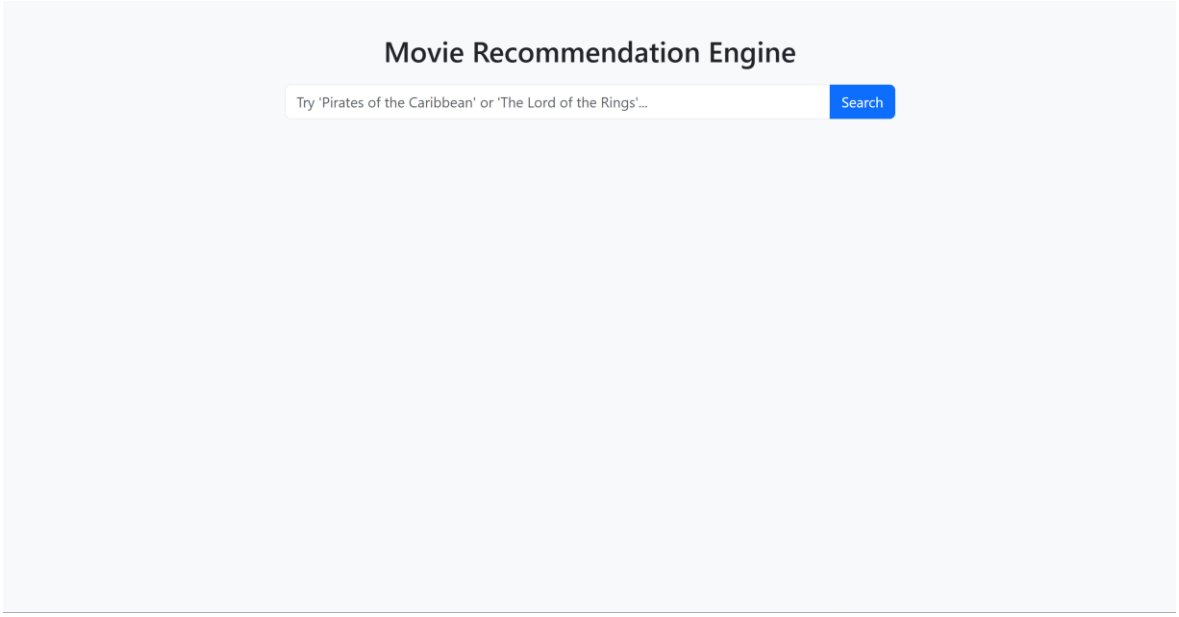


Рисунок 3.13 - Інтерфейс веб-додатку

Одним з ключових досягнень інтерфейсу є реалізація інтелектуального автозаповнення, що який продемонстрований на рисунку 3.14. Коли користувач починає вводити назву фільму, система динамічно генерує список запропонованих варіантів, базуючись на наявних в базі даних назвах.

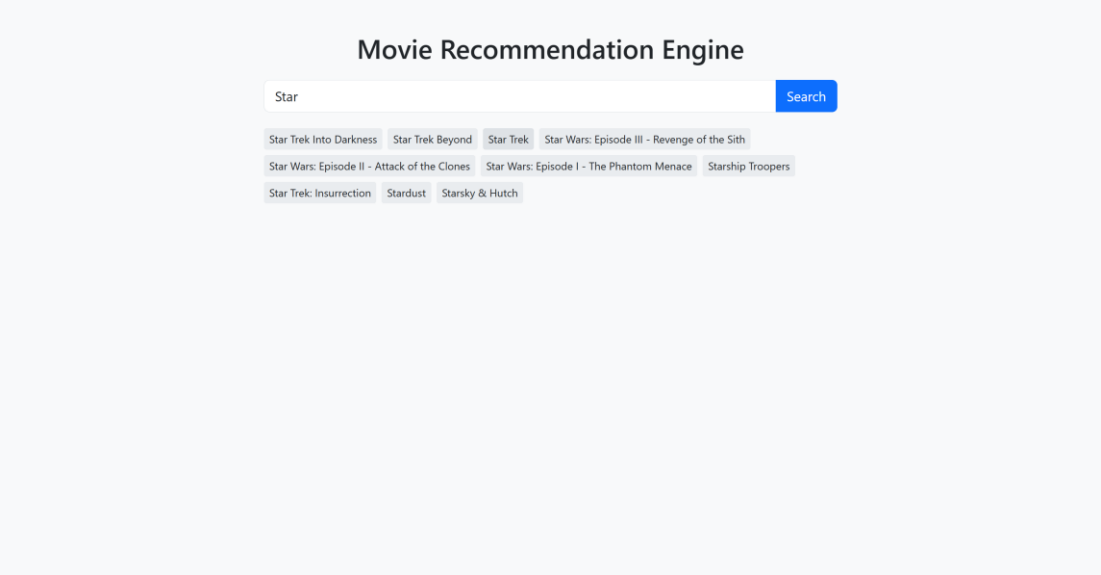


Рисунок 3.14 – Приклад автозаповнення

Ця функціональність значно покращує користувацький досвід через кілька важливих аспектів:

— Швидкість взаємодії - користувачам не потрібно вводити повну назву фільму, достатньо кількох символів для отримання релевантних пропозицій. Це особливо цінно для фільмів з довгими або складними назвами.
— Зменшення помилок - автозаповнення мінімізує ризик орфографічних помилок, які могли б призвести до невдалого пошуку. Система фактично гарантує, що вибраний фільм існує в базі даних.
— Покращення точності пошуку - пропонуючи точні назви з бази даних, система забезпечує коректну ідентифікацію фільму для подальшого аналізу схожості.

Після натискання кнопки пошуку система генерує та відображає п'ять найбільш схожих фільмів. Приклад використання представлено на рисунку 3.15. Кожна рекомендація представлена як комплексна інформаційна карточка, що включає:

— Візуальний компонент - постер фільму служить миттєвим візуальним ідентифікатором та створює емоційне підключення з потенційним глядачем.

Якісні постери значно підвищують привабливість рекомендацій та полегшують швидке сканування результатів.

— Текстовий опис - кожна рекомендація супроводжується описом сюжету, який надає користувачам детальну інформацію про зміст фільму. Ці описи дозволяють оцінити релевантність рекомендації та зрозуміти, чому саме цей фільм був запропонований.

— Числові показники - рейтинги фільмів надають об'єктивну оцінку якості та популярності рекомендованих стрічок. Ця інформація допомагає користувачам приймати більш обґрунтовані рішення щодо перегляду.

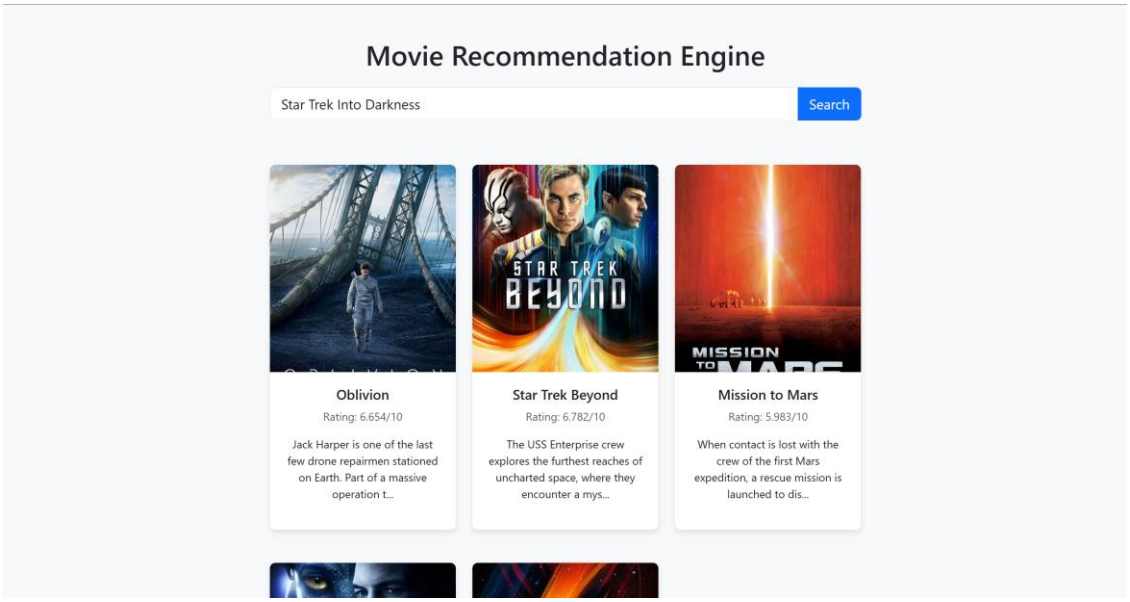


Рисунок 3.15 – Приклад роботи рекомендаційної системи

Система представляє рекомендації в порядку зменшення схожості з вхідним фільмом. Такий підхід забезпечує, що найбільш релевантні варіанти з'являються першими, максимізуючи ймовірність того, що користувач знайде цікаві для себе фільми серед перших результатів.

Обмеження кількості рекомендацій п'ятьма елементами запобігає інформаційному перевантаженню та забезпечує фокус на найякісніших варіантах. Це рішення базується на психологічних принципах сприйняття інформації, де невелика кількість варіантів полегшує процес вибору.

Такий підхід до презентації результатів перетворює простий список рекомендацій на інформативний каталог, який не лише пропонує варіанти, але й надає інструменти для їх оцінки.

3.3.6 Можливості вдосконалення

Поточний механізм автозаповнення, хоча і функціональний, може бути значно покращений через впровадження нечіткого пошуку (fuzzy search). Така система дозволила б користувачам знаходити фільми навіть при наявності орфографічних помилок або неточностей у написанні. Інтеграція алгоритмів відстані Левенштейна або подібних методів створила б більш толерантний до помилок інтерфейс.

Додавання можливості пошуку не лише за назвою, але й за іменами акторів, режисерів або ключовими словами значно розширило б функціональні можливості системи. Користувачі могли б, наприклад, ввести "Леонардо Ді Капріо" і отримати всі доступні фільми з його участю для подальшого вибору базового фільму для рекомендацій.

Додавання можливості збереження цікавих фільмів у персональний список "до перегляду" значно підвищило б практичну цінність системи. Користувачі могли б формувати власні колекції рекомендованих фільмів для майбутнього перегляду.

Інтеграція додаткових даних про фільми - тривалість, країна виробництва, касові збори, нагороди - створила б більш повну картину для кожної рекомендації. Посилання на трейлери або офіційні сторінки фільмів додали б інтерактивності та практичної корисності.

Система могла б відображати короткі пояснення щодо того, чому саме конкретний фільм був рекомендований. Наприклад: "Рекомендовано через схожі жанри: фантастика, бойовик" або "Схожі актори: Том Круз, Скарлетт Йоханссон". Така прозорість алгоритму підвищила б довіру користувачів до системи.

Впровадження механізмів, які забезпечували б різноманітність у рекомендаціях, запобігло б ситуаціям, коли всі п'ять запропонованих фільмів занадто схожі між собою. Система могла б балансувати схожість з різноманітністю, пропонуючи фільми з різних піджанрів або періодів.

3.4 Висновки до розділу

У розділі реалізовано повний цикл створення рекомендаційної системи фільмів — від підготовки даних до розгортання веб-додатку. Побудована модель контентної фільтрації демонструє точність і швидкодію, а інтеграція з інтерфейсом користувача підтверджує практичну цінність розробки. Отриманий прототип слугує надійною основою для подальшого розвитку системи.

ВИСНОВКИ

У ході виконання дипломної роботи було розроблено рекомендаційну модель машинного навчання для стрімінгових сервісів, що забезпечує ефективне опрацювання метаданих фільмів та формування персоналізованих рекомендацій для користувачів.

Виконано аналіз існуючих рішень у сфері машинного навчання для рекомендаційних систем кінематографу, який підтвердив необхідність розробки комплексного підходу, що поєднує аналіз контентних характеристик фільмів та поведінкових патернів користувачів.

Обґрунтовано вибір технологій, які допомогли в реалізації рекомендаційної системи.

Розроблена модель успішно аналізує метадані фільмів та виявляє взаємозв'язки між характеристиками контенту та уподобаннями користувачів. Реалізовані функціональні компоненти системи забезпечують формування релевантних персоналізованих рекомендацій.

Валідація моделі на реальних даних підтвердила її високу точність та надійність. Розроблені метрики оцінки ефективності показали значно кращу якість рекомендацій порівняно з базовими алгоритмами.

Практична цінність роботи полягає в можливості інтеграції системи в стрімінгові платформи для підвищення задоволеності користувачів. Теоретичний внесок складає розробка комплексного підходу до аналізу кінематографічних даних з використанням методів машинного навчання.

Результати дослідження підтверджують можливість створення ефективних персоналізованих рекомендаційних систем, що покращують взаємодію між створювачами контенту та споживачами в умовах цифрової трансформації індустрії розваг.

ПЕРЕЛІК ПОСИЛАНЬ

1. Effect of User Tastes on Personalized Recommendation [Електронний ресурс]: https://arxiv.org/abs/0907.1224

2. Streaming Overload? Nielsen Report Finds Average Viewer Takes 7 Minutes To Pick What To Watch; [Електронний ресурс]: https://libspros.dynu.net/2019/07/streaming-overload-netflix-nielsen-report-average-viewer-takes-7-minutes-to-pick-what-to-watch-1202640213/

3. The 2023 Product Recommendations Playbook [Електронний ресурс]: https://netcorecloud.com/blog/the-2023-product-recommendations-playbook

4. Джерело зображення - The value of getting personalization right—or wrong—is multiplying [Електронний ресурс]: https://www.mckinsey.com/capabilities/growth-marketing-and-sales/our-insights/the-value-of-getting-personalization-right-or-wrong-is-multiplying

5. Book Recommender System Using Matrix Factorization with Alternating Least Square Method [Електронний ресурс]: https://www.researchgate.net/publication/377244081_Book_Recommender_System_Using_Matrix_Factorization_with_Alternating_Least_Square_Method

6. Джерело зображення - Book Recommender System Using Matrix Factorization with Alternating Least Square Method [Електронний ресурс]: https://www.researchgate.net/publication/377244081_Book_Recommender_System_Using_Matrix_Factorization_with_Alternating_Least_Square_Method

7. Джерело зображення - Choosing the right estimator – scikit-learn 1.4.2 documentation [Електронний ресурс]: https://scikit-learn.org/1.4/tutorial/machine_learning_map/index.html

8. FastAPI: The Modern Toolkit for Machine Learning Deployment [Електронний ресурс]: https://medium.com/%40reza.shokrzad/fastapi-the-modern-toolkit-for-machine-learning-deployment-af31d72b6589

9. Джерело зображення - How to build complete end-to-end ML model, Backend RestAPI using FastAPI and front-end UI using Streamlit [Електронний ресурс]: https://medium.com/@goradbj/how-to-build-complete-end-to-end-ml-model-backend-restapi-using-fastapi-and-front-end-ui-using-22f64bf04476

10. The Ultimate Guide to Docker: From Novice to Expert [Електронний ресурс]: https://faun.pub/the-ultimate-guide-to-docker-from-novice-to-expert-cf961536c147

11. Джерело зображення - The Ultimate Guide to Docker: From Novice to Expert [Електронний ресурс]: https://faun.pub/the-ultimate-guide-to-docker-from-novice-to-expert-cf961536c147

12. Understanding TF-IDF (Term Frequency-Inverse Document Frequency) [Електронний ресурс]: https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/

13. Understanding TF-IDF (Term Frequency-Inverse Document Frequency) [Електронний ресурс]: https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/

14. Джерело зображення - How does Netflix work behind the screen? [Електронний ресурс]: https://uxdesign.cc/netflix-system-design-ef5802426ad4

15. Джерело зображення - Customers Who Bought This Item Also Bought…Affinity Analysis Explained [Електронний ресурс]: https://measuringu.com/affinity-analysis/

16. TMDB 5000 Movie Dataset Kaggle [Електронний ресурс]: https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata/data

ДЕМОНСТАРЦІЙНІ МАТЕРІАЛИ

Презентація

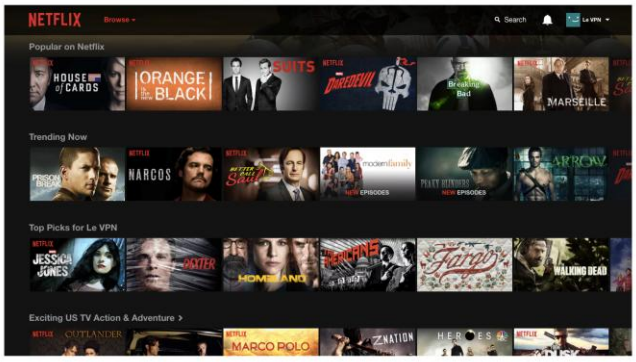
Державний університет інформаційно-комунікаційних технологій

Моделі машинного навчання для аналізу та передбачення в сфері кінематографу на основі мови програмування Python

Виконав студент групи КНД-41

Гаврилов Сергій

Київ 2025



Мета роботи

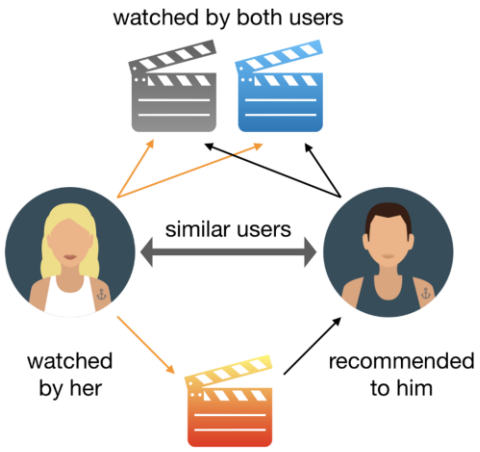
Розробка інтелектуальної рекомендаційної моделі машинного навчання для кіносервісів, що забезпечує персоналізовані пропозиції фільмів на основі аналізу метаданих контент.

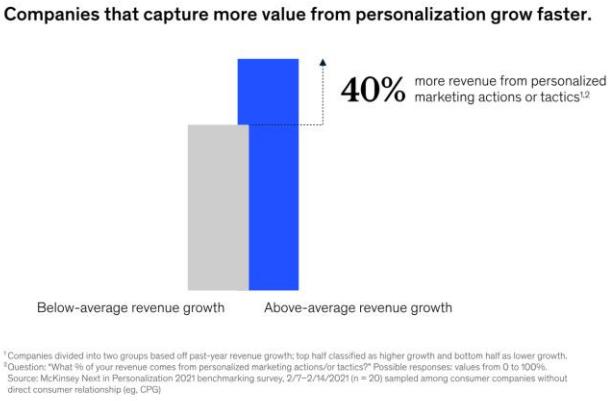
Актуальність

Актуальність обумовлена:

1. Глобальною діджиталізацією індустрії розваг,
2. стрімким розвитком стрімінгових платформ
3. необхідністю структурування й аналізу колосальних обсягів даних про фільми та серіали.

Крім того, точні алгоритми рекомендацій здатні суттєво підвищити задоволеність користувачів, зробити їхній перегляд більш осмисленим та сприяти збільшення утримання користувачів та прибутку для компанії





[1]Companies divided into two groups based off past-year revenue growth; top half classified as higher growth and bottom half as lower growth.
[2]Question: "What % of your revenue comes from personalized marketing actions/or tactics?" Possible responses: values from 0 to 100%.
Source: McKinsey Next in Personalization 2021 benchmarking survey, 2/7–2/14/2021 (n = 20) sampled among consumer companies without direct consumer relationship (eg, CPG)

Проблема

За даними дослідження компанії Nielsen, середньостатистичний користувач стрімінгових сервісів витрачає близько 7.4 хвилин лише на вибір фільму чи серіалу для перегляду.

Ідея

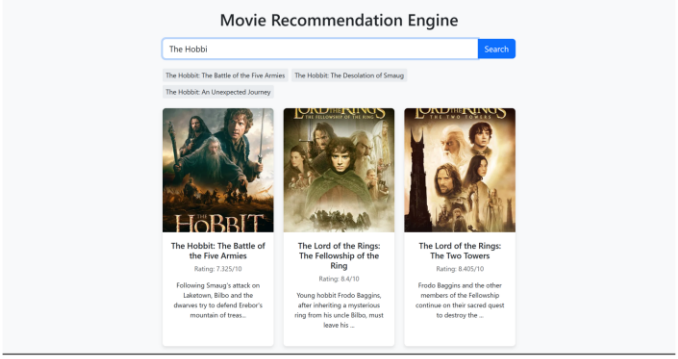
Має бути створений веб-додатку, який надає користувачам можливість шукати рекомендації до введених фільмів з метою полегшення відкриття нового контенту.

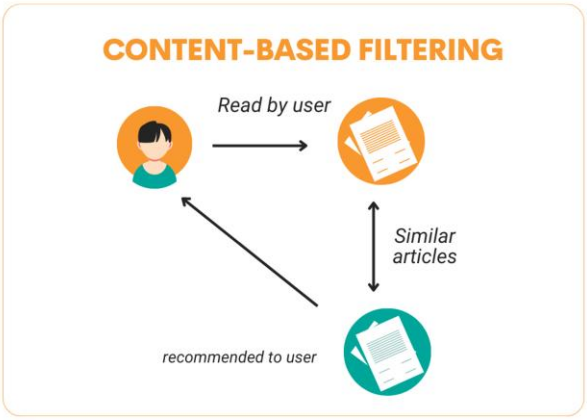
Основна ідея полягає в тому, щоб забезпечити швидкий та інтуїтивно зрозумілий інтерфейс, де користувач вводить назву відомого йому фільму, а система на основі попередньо навчених моделей машинного навчання повертає список схожих стрічок.

Рішення

Використання scikit-learn і Python для вирішення проблеми залученості користувачів дозволяє створити персоналізовану систему рекомендацій, яка адаптується під інтереси кожного користувача.

Машинне навчання виявляє закономірності у поведінці користувачів, які складно помітити вручну.





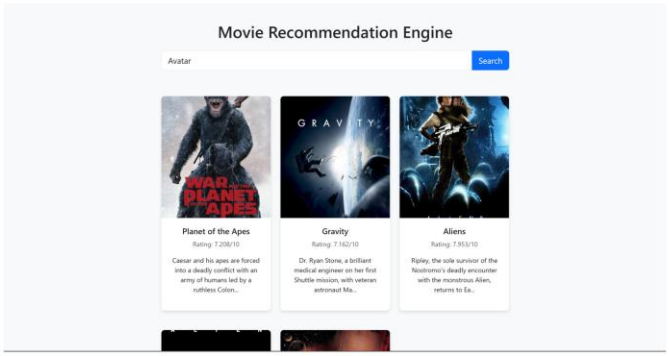
Результат створення моделі

Побудована модель контентної фільтрації демонструє точність і швидкодію, а інтеграція з інтерфейсом користувача підтверджує практичну цінність розробки. Отриманий прототип слугує надійною основою для подальшого розвитку системи.

Веб-додаток

У результаті створення веб-додатку було реалізовано повноцінну систему рекомендацій фільмів, яка дозволяє користувачам у зручному форматі знаходити новий контент на основі введених назв.

Користувач отримує простий інтерфейс для пошуку, швидку відповідь із рекомендаціями, а також додаткову інформацію про фільми, включно з постерами, рейтингами та описами завдяки інтеграції з TMDB API.



Висновки

У ході дипломної роботи створено рекомендаційну систему для фільмів, що використовує машинне навчання для аналізу метаданих та уподобань користувачів.

Розроблена модель забезпечує точні персоналізовані рекомендації, а створений веб-додаток демонструє практичне застосування рішення. Валідація показала високу ефективність, а обраний технологічний стек дозволив реалізувати систему, готову до інтеграції у стрімінгові сервіси.