ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення

Пояснювальна записка

до магістерської роботи
на ступінь вищої освіти магістр

на тему: **«РОЗРОБКА АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ ДЛЯ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ВИБОРУ МУЗИЧНИХ КОМПОЗИЦІЙ»**

Виконав: студент 6 курсу, групи ПДМ-61
спеціальності 121 Інженерія программного забезпечення
(шифр і назва спеціальності)
Демидов Д.Д.
(прізвище та ініціали)

Керівник Щербина І.С.
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Київ – 2020

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти - «Магістр»
Спеціальність - 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____О.В. Негоденко

"____"____20___року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Демидову Данилу Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка алгоритмів машинного навчання для рекомендаційної системи вибору музичних композицій»

Керівник роботи к.т.н., доцент Щербина І.С. ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "13" жовтня року №230.

2. Строк подання студентом роботи 24.12.2020.
3. Вихідні дані до роботи:
   3.1. Положення побудови рекомендаційних систем;
   3.2. Методи побудови рекомендаційних систем;
   3.3. Розробка моделі рекомендаційної системи;
   3.4. Науково-технічна література.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
   4.1. Загальні положення побудови рекомендаційних систем;
   4.2. Аналіз технології і методів побудови рекомендаційних систем;
   4.3. Побудова музичної рекомендаційної системи;
   4.4. Висновки
5. Перелік графічного матеріалу.
   5.1. Основні характеристики роботи;
   5.2. Актуальність задачі;
   5.3. User-based системи;
   5.4. Item-based системи;

5.5. Недоліки існуючих систем;
5.6. Методи машинного начання;
5.7. Використання машинного навчання;
5.8. Метод градієнтного бустінгу;
5.9. Реалізація алгоритмів рекомендаційних систем;
5.10. Порівняння якостей отриманих результатів;
5.11. Порівняння швидкості формування рекомендацій для одного користувача;
5.12. Архітектура рекомендаційної системи;
5.13. Висновки.

6. Дата видачі завдання 02.11.2020

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	02.11.20	Виконано
2	Дослідження положення побудови рекомендаційних систем	04.11.20	Виконано
3	Аналіз методів побудови рекомендаційних систем	11.11.20	Виконано
4	Розробка моделі рекомендаційної системи	15.11.20	Виконано
5	Висновки, оформлення роботи	29.11.20	Виконано
6	Розробка демонстраційних матеріалів	10.12.20	Виконано
7	Здача роботи	24.12.20	Виконано

Студент _____ Демидов Д.Д.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Щербина І.С.
підпис) (прізвище та ініціали

РЕФЕРАТ

Текстова частина магістерської роботи 59 стр., 32 рис., 13 табл., 30 джерел

РЕКОМЕНДАЦІЙНА СИСТЕМА, МАШИННЕ НАВЧАННЯ, ГРАДІЄНТНИЙ БУСТІНГ, АЛГОРИТМ, СИСТЕМА, МЕТОД, МАТРИЦЯ, ВИБІРКА.

Об'єктом дослідження - є рекомендаційні системи та алгоритми їх побудови.

Мета роботи – є формування списку музикальних композицій, заснованих на вподобаннях користувача та доречності використання обраного об'єкту.

Предмет дослідження – є існуючі алгоритми та структури даних рекомендаційних систем, їх ефективність в умовах неповної або відсутньої інформації.

Методи дослідження. У науковій роботі було використані загальнонаукові та спеціальні методи дослідження, а саме: методи аналізу та синтезу, індукції та порівняння під час дослідження досвіду впровадження рекомендаційних систем та побудови списку рекомендацій, метод єдності історичного та логічного – для відображення етапів впровадження рекомендаційних систем та еволюційних процесів покращення методів побудови рекомендацій, гіпотетико-дедуктивний метод при формуванні гіпотез та їх спрощення чи підтвердження, методи та алгоритми побудови рекомендаційних списків на основі даних про дії користувачів, методи машинного навчання - для розробки моделей прогнозування списку музичних композицій, а також графічні методи – для представлення результатів дослідження у вигляді схем, таблиць та рисунків тощо.

ЗМІСТ

ВСТУП

Актуальність теми. З розвитком інформаційних технологій та їх впровадженням у суспільне життя, виникає потреба пошуку акцентованої інформації в умовах невизначеності. Для вирішення таких задач в останній час створюються інтелектуальні рекомендаційні системи [1]. Популярність рекомендаційних систем зростає в кожному сегменті товарів і послуг, зокрема музичних. З соціально-економічної точки зору, такі системи є основним інструментом поширення нових композицій в сфері музики, сприяє просуванню цих композиції відповідно вподобань цільової аудиторії і стимулює користувачів набувати нові музичні треки. Крім цього такі системи значно скорочують час і полегшують пошук відповідних музичних композицій в умовах невизначеності.

Незважаючи на те, що в світі інформаційних технологій існує величезна кількість різних алгоритмів, оптимізація рекомендацій і новий підхід до формування рекомендаційних об'єктів, поліпшення якості рекомендацій, обґрунтування рекомендацій. Реалізація даного проекту є зрілою та усвідомленою ідеєю, що ґрунтується на меті дослідження сфери рекомендаційних систем та їх впливу на життя людей.

Дамо визначення поняттю рекомендаційна система. Рекомендаційні системи – це спеціальні програми, головна мета яких полягає у формуванні рекомендацій різних продуктів або сервісів для користувачів на основі їх переваг [1].

Джерела дослідження: Інформаційну та теоретичну базу роботи склали підручники та посібники з теорії побудови рекомендаційних систем, машинного навчання та розробки програмних продуктів за допомогою мови Python, СУБД PostgreSQL, лістинги програмного коду та інтернет-ресурси з тематики програмування та аналізу даних, матеріали конференцій та наукових статей присвячених обговоренню проблематики формування рекомендаційних списків, різноманітні інформаційні та статистичні ресурси.

Метою магістерської роботи є формування списку музикальних композицій, заснованих на вподобаннях користувача та доречності використання обраного об’єкту.

Завдання магістерської роботи:

- дослідити особливості побудови рекомендаційних систем, зокрема алгоритмів та методів формування списку рекомендацій;
- розкрити актуальність використання рекомендаційних систем та впровадження їх на підприємствах та додатках;
- проаналізувати принцип роботи існуючих рекомендаційних систем, зроби;
- виявити недоліки існуючих систем та запропонувати заходи для їх усунення та покращення рекомендаційних систем;
- дослідити основні методології оцінки моделі та ефективності роботи рекомендаційних систем;
- довести доцільність впровадження нових методів побудови рекомендацій;
- побудувати модель на основі популярності (наївний метод);
- побудувати модель в рекомендаційній системі на основі колаборативної фільтрації;
- побудувати модель на основі методу градієнтного бустингу;
- проаналізувати результати та запропонувати можливості покращення.
- спроектувати архітектуру рекомендаційної системи;
- розробити демонстраційний приклад застосування рекомендаційних систем;

Об’єктом дослідження є рекомендаційні системи та алгоритми їх побудови.

Предметом дослідження є існуючі алгоритми та структури даних рекомендаційних систем, їх ефективність в умовах неповної або відсутньої інформації.

Методи дослідження. У науковій роботі було використані загальнонаукові та спеціальні методи дослідження, а саме: методи аналізу та синтезу, індукції та порівняння під час дослідження досвіду впровадження рекомендаційних систем та побудови списку рекомендацій, метод єдності історичного та логічного – для відображення етапів впровадження рекомендаційних систем та еволюційних процесів покращення методів побудови рекомендацій, гіпотетико-дедуктивний метод при формуванні гіпотез та їх спрощення чи підтвердження, методи та алгоритми побудови рекомендаційних списків на основі даних про дії користувачів, методи машинного навчання - для розробки моделей прогнозування списку музичних композицій, а також графічні методи – для представлення результатів дослідження у вигляді схем, таблиць та рисунків тощо.

Практичне значення одержаних результатів. Практичне значення визначається узагальненням підходів до розуміння сутності побудови рекомендаційних систем як методів побудови списку об’єктів, що пропонуються користувачу для задоволення своїх потреб, причому корисність запропонованих об’єктів для користувача повинна відповідати очікуванням.

Публікації. За матеріалами роботи опубліковано одну статтю у науковому журналі.

1. ЗАГАЛЬНІ ПОЛОЖЕННЯ ПОБУДОВИ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

1.1 Пріоритетні напрями розвитку рекомендаційних систем

Рекомендаційні системи використовуються для прогнозування списку об’єктів, що відповідають вподобанням користувача. В даний час рекомендаційні системи для реалізацій різних рішень у сфері електронною комерції. Основною задачею таких систем є збільшення конверсії, тобто перетворення відвідувачів у реальних користувачів. Рекомендаційні системи дозволяють спросити пошук об’єктів, що входять у зону інтересів споживачів, а також дозволяють пропонувати конкретним користувачам спеціалізований контент.

Зараз використовуються різні підходи для реалізації алгоритмів формування списку рекомендацій. Найпростішим алгоритмом є використання статистичних даних для виявлення популярних, дешевих чи дорогих об’єктів, без наявності певних вхідних параметрів користувача. Більш складні алгоритми використовують конкретні риси користувачів для формування персоналізованих образів користувача, аналізуючи його поведінку та вподобання при вибору товарів чи послуг. До характеристик користувача можна віднести наступні: швидка зміна даних у часі та великий обсяг даних поведінки.

Рекомендаційні системи працюють на двох "рівнях":

І рівень – глобальне оцінювання особливостей і переваг певного контенту, що змінюється повільно. Залежність від характерних для користувача рис, наприклад: вік, стать, місце проживання, інтереси, рід діяльності.

ІІ рівень – аналіз короткочасних змін у інтересах користувачів та пропонування актуального контенту.

Існують різні підходи до розробки рекомендаційних систем, які можуть застосовуватися в залежності від:

- доступних даних про користувачів і рекомендованих сутності;
- видів явною і неявній зворотного зв'язку від користувачів;
- предметної області [5].

Ряд дослідників, які застосували свої рекомендаційні системи на практиці відзначають, що найбільш ефективно працює система, яка використовує для побудови рекомендацій дані про поточного користувача, про поведінку всіх користувачів в цілому, про властивості рекомендованих продуктів і про контексті поточного інтересу користувачах [3, 5].

Розглянемо варіанти застосування контексту на прикладі музичних сервісів:

1. Формування списку музичних композицій в залежності від часу доби – «сон», «пробудження», «фонова музика».

2. Враховувати географічне положення користувача – «Дома», «Подорож», «В морі», «На дачі».

3. Використання інформації щодо приладів, в залежності скільки користувачів прослуховують композиції. Наприклад, якщо користувач один (навушники) або музику прослуховують в компанії (колонки).

4. За видом активності користувача, використання даних з акселерометрів, GPS або смарт-годинників. За допомогою цієї інформації система пропонує різний контент, пропонування динамічної або, навпаки, спокійної музики.

Рекомендаційні системи інтегруються в онлайн-системи різного призначення. Такі рішення дозволяють покращати сервіси та задовольняти клієнтів, саме тому є вимога створення рекомендаційних систем на основі нових алгоритмів для покращення швидкості і результатів. Основними галузями використання рекомендаційних систем є музика, відео, продуктовий рітейл,

туризм, соціальні мережі, новини, онлайн-навчання та харчування. Розглянемо кожну галузь більш конкретно.

1. Музична сфера – окрім використання звичайних алгоритмів формування рекомендацій, розвиваються нові напрямки. Одним із перспективних алгоритмів є аналіз музичних спектрограм користувачів. Система аналізує залежності між спектрограмами пісень та намагається прогнозувати та пропонувати клієнту схожі композиції за показниками децибел, амплітуди звука за часом. Також використовуються аналіз емоційного стану користувача та динамічне формування списку композицій на основі почуттів людини [6].

2. Відео сфера – у цій сфері почали вперше з’являтись рекомендаційні системи. Такі світові сервіси як Netflix, Amazon-Prime навіть зараз покращають свої алгоритми формування списку рекомендацій. Система рекомендацій для таких послуг розвивається як потужний бізнес-інструмент для надання допомоги клієнтам. Він також допомагає постачальникам послуг максимізувати прибуток, створюючи безперервні ланцюги нових відео для користувачів [7]. Система рекомендацій, реалізована на веб-сайтах, що переглядають відео, таких як YouTube, динамічно масштабується, оскільки мільйони користувачів завантажують та переглядають відео кожну секунду. Рекомендації такого веб-додатку мають складний механізм масштабування і релевантность пропонувань. Також, емоційний стан людини може впливати на рекомендації, виявлені емоції використовуються для раціонального вибору відео для користувачів [22].

3. Продуктова сфера – рекомендації щодо продуктів у веб-сайтах електронної комерції підвищують продажі, перетворюючи веб-переглядачі на покупців, пропонуючи кросс-продажі додаткових продуктів. Система рекомендацій, яка використовується в веб-сайтах електронної комерції, націлена на певний сегмент ринку, використовуючи їхню низку унікальних функцій, а знання про особливості продуктової лінії надають рекомендації, що дозволяють швидко виходити на ринок, гнучко реагувати та адаптувати персоналізацію

контенту. У електронної комерції дані про покупця, такі як стать, вік, сімейний стан, кількість дітей і грошовий стан, визначають покупку продуктів. Система неявно збирає інформацію про продукти, додані до кошика, частоту придбання продукту та ціну придбаного продукту, щоб визначити відповідні продукти для рекомендації користувачеві [23].

4. Сфера туризму – успіхи в інформаційно-комунікаційних технологіях сприяють розвитку туристичної індустрії, надаючи туристам доступ до недорогих, точних, достовірних відомостей про ресторани, готелі та туристичні пам'ятки. Передбачення шляху туриста та його поведінки використовується для створення персоналізованої моделі для рекомендацій на основі поведінкових моделей. Географічна відстань, з якою людина буде подорожувати з "місця перебування", однакова, і тому, де б людина не подорожувала, рекомендовані "місця для відвідування" повинні знаходитися в межах географічної відстані. Розташування відіграє важливу роль у поведінці користувачів у визначеній точці. Формування рекомендацій, заснованих на розташуванні, надає активну класифікацію користувачів для рекомендації відповідних точок інтересу. Географічний вплив на місцеположення (тобто близькість між точками інтересу) використовується для надання рекомендацій для зацікавлених користувачів [24].

5. Соціальні мережі – традиційно в реальному житті на друзів впливає географічна відстань. У цьому віртуальному віці люди зв'язуються з іншими, спираючись на стиль життя, соціальний статус, моральні цінності, особисте ставлення та існуючі суспільні відносини. Топологічна структура соціальної мережі користувача виводить існуючі соціальні відносини для рекомендації друга. Віртуальна соціальна мережа допомагає у розробці ефективних реальних взаємодій. Подібність у відвіданих місцях та географічна близькість допомагають знайти однодумців та брати участь у соціальних подіях [25].

6. Сфера новин – у цю цифрову епоху прокрутка планшета набагато простіше, ніж читання газети. Кожна нова подія має цілодобовий цикл новин, і

вона повинна бути доставлена в переконливій манері, щоб залишатися попереду конкурентів. Рекомендації щодо новин повинні фільтрувати новини з урахуванням 24-годинного циклу. Обсяг онлайн-новин з кожним роком подвоюється щороку разом з користувачами [26]. Прогнозування наступної статті, що користувач читає, є важливим критерієм популярності джерел новин. Контекстна інформація, така як час доби, рік статті та час, проведений за статтею, допомагає вирішувати динамічний характер рекомендацій новин. Своєчасність концепції потребують регулярних та надійних рекомендаційних алгоритмів для оновлення списку рекомендацій та пропонування актуальних та популярних статей [27].

7. Онлайн-навчання – адаптивне середовище електронного навчання визначає «що» та «коли» треба студенту для навчання. Стиль навчання та рівень знань визначають переваги студента щодо навчальних об'єктів (документи, відео-лекції). Послідовна схема навчання, отримана внаслідок взаємодії з системою, допомагає будувати план навчання . Розумна навчальна система налаштовує шлях навчання учнів. Навігації та побудова шляху навчання підвищує навчальні досягнення учня. Точність і правильність пропонованого матеріалу спонукає учнів систематично розвивати свої особисті знання. Стандартизовані рекомендації розташовуються на спеціалізованих онлайн веб-платформі навчання. Вивчення та запам’ятання матеріалу є унікальним для індивідуума, тому рекомендації формуються індивідуально для кожного учня, на основі його досягнень та способу навчання [28].

Сформуємо таблицю, що відображає перспективи та принципи розвитку рекомендаційних систем у різних сферах:

Таблиця 1.1 – Основні характеристики рекомендаційних систем у різних сферах

Сфера	Рекомендаційний метод	Характеристики
1	2	3
Музика	Content Based, Collaborative, Context Based	Повторення треків, аналіз емоцій слухача, використання часу, місця та події
Відео	Content Based, Collaborative, Context Based, Hybrid	Масштабування та релевантність відео, таргетування на інтереси користувача, використання часу, місця та події
Рітейл	Collaborative, Hybrid	Швидкозростаючий ринок, кількість продуктів та користувачів, комплементарні продукти, популярні продукти
Туризм	Content Based, Collaborative, Context Based	Користувачі подорожують разом, зв’язок з геопозицією користувача, регулярність шляхів користувача
Соціальні мережі	Content Based, Collaborative, Context Based	смак, спосіб життя, географічна близькість, релевантність та популярність
Новини	Collaborative, Context Based, Hybrid	Волатильність, короткий термін життя, час та позиція читача, рік статті та характеристики статті
Онлайн-навчання	Content Based, Context Based, Hybrid	Рівень знання читача та особисті звички, динамічна подача матеріалу, пристосування до стилю навчання
Ресторани	Context Based, Hybrid	Релевантність, пристосування до особистих потреб, враховути час та погодні умови

Джерело: складено автором на основі [6, 7, 22, 23, 24, 25, 26, 27, 28]

Отже, зазначені характеристики відображають можливості покращення роботи рекомендаційних та впровадження їх у різні сфери діяльності.

1.2 Роль і місце рекомендаційних систем в інформаційних системах

Перший цифровий відеомагнітофон, який намагався врахувати переваги користувачів, щоб автоматично записувати відповідні їм передачі, з'явився в кінці 1990-х. Пристрій називалося TiVo і продавалося кінцевим користувачам такими брендами, як Sony і Філіпс.

Однією з перших рекомендаційних систем були система Інтернет-магазину Amazon і система компанії Google. У 1992 р в якості основного алгоритму для рекомендаційних систем був запропонований метод колаборативной фільтрації. .Він заснований на використанні в рекомендаційної системі інформації про доступні треках всіх користувачів. Цей метод дозволив вирішити завдання досить ефективно, виявився вельми досконалим і в даний час поліпшення показника якості рекомендаційної системи на 10% оцінюється в конкурсі NetflixPrize в 1 мільйон доларів. Чемпіоном застосування стала компанія Netflix, яка для свого бізнесу надання DVD-дисків в оренду привернула команду розробників, які налаштовують параметри рекомендаційного движка колаборативной фільтрації, і домагалася рік від року все кращих результатів.

В процесі розвитку нових підходів до вирішення завдання на гребені хвилі виявилися нові компанії, зокрема Jinni і Aprico, які зрозуміли, що якісна інформація про рекомендований контенті могла б служити серйозною основою для тонкого налаштування під інтереси глядачів. Ця нова хвиля досліджень в основному використовувала

Байєсова класифікацію, яка дозволяє зрозуміти і оцінити, чому користувач віддає перевагу той чи інший контент. Наприклад, якщо користувач дивиться фільми типу «Джеймса Бонда», але пропускає «Одинадцять друзів Оушена », то система вважає, що коли мова йде про фільми, цей користувач любить бойовики і пригоди, але не особливо любить триллери [2].

Рекомендаційні системи присутні на багатьох веб-сайтах, які ви використовуєте щодня, включаючи такі відомі сайти. Яскравими прикладами компаній, що використовують даний підхід, є Amazon, eBay, iTunes та інші. Інша важлива застосування рекомендаційних систем - це допомога користувачам у виборі книг, музики і фільмів. Наприклад, сервіси Pandora, GoodReads, and IMDb використовують рекомендаційні системи для цих цілей. Давайте розглянемо їх докладніше на прикладах.

LinkedIn – сайт бізнес-орієнтованої соціальної мережі – пропонує користувачеві рекомендації щодо людей, яких він, можливо, знає, робочі місця, які могли б його залучити, групи, в які він міг би захотіти вступити, компанії, якими він міг би зацікавитися. Спеціалізована система коллаборативной фільтрації LinkedIn заснована на технології ApacheHadoop.

Amazon – популярний сайт електронної комерції – використовує рекомендації на основі контенту. Коли відвідувач вибирає для покупки будь-якої товар, Amazon на основі цього вихідного товару рекомендує відвідувачеві інші товари, придбані іншими користувачами (за допомогою матриці покупки наступного товару на основі його схожості з попередньою покупкою). Компанія Amazon запатентувала цей підхід під назвою item-to-itemcollaborativefiltering (коллаборативна фільтрація від елемента до елементу).

Hulu – веб-сайт потокового відео – використовує рекомендаційний механізм для розпізнавання контенту, який міг би становити інтерес для користувачів. Він також використовує (в оффлайновом режимі) механізм Hadoop для масштабованої обробки величезних обсягів даних при виконанні "Коллаборативной фільтрації на основі елементів" (item-based collaborative filtering).

Netflix – постачальник відеоконтенту на умовах оренди і у вигляді потокового сервісу – є широковідомим прикладом у цій сфері. У 2006 році компанія Netflix оголосила конкурс на вдосконалення своєї рекомендаційної

системи під назвою Cinematch. У 2009 р три групи розробників об'єднаними зусиллями створили "ансамбль" з 107 рекомендаційних алгоритмів, яка формує єдиний прогноз. Цей ансамбль відіграв провідну роль в підвищенні точності прогнозування, в результаті чого саме ця об'єднана група і виграла приз Netflix.

Крім того, рекомендаційні механізми використовуються на таких сайтах, як Facebook, Twitter, Google, MySpace, Last.fm, Del.icio.us, Pandora, Goodreads, а також на вашому улюбленому сайті онлайнових новин. Використання того чи іншого рекомендаційного механізму стає стандартним елементом сучасного веб-представництва [3].

Підсумовуючи вище наведені дані, зробимо припущення про те, як використання рекомендаційних систем є важливим в сучасному світі, особливо звернемо увагу на статистичні показники використання інтернету та рекомендаційних систем.

1.3 Статистичні дані та показники активності застосування рекомендаційних систем в інтернет-мережі

За даними InternetWorldStats [4] на кінець 2018 року було нараховано понад 4,2 млрд. користувачів інтернету. Приріст нових користувачів інтернету складає 10%, також збільшилась активність в соціальних мережах (10%) та мобільних користувачів соціальних мереж (17%). Розглянемо розподіл користувачів інтернету у розрізі країн світу (рис.1.1).

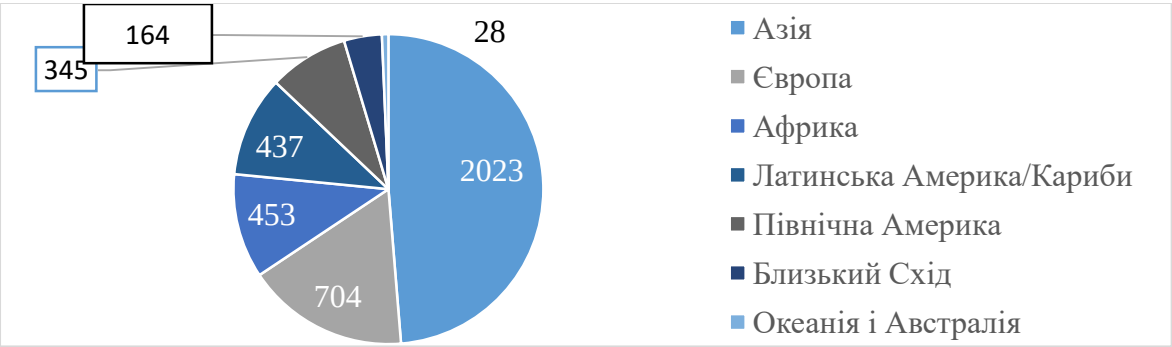

Рисунок – 1.1 Діаграма розподілу користувачів інтернету серед країн

Бачимо з рис. 1.1, що найбільша частка користувачів інтернету припадає на Азію. Найменша частка – Океанія і Австралія.

Також важливим показником є факти та статистика з приводу корисності відгуків про різні види продукції (рис.1.2).

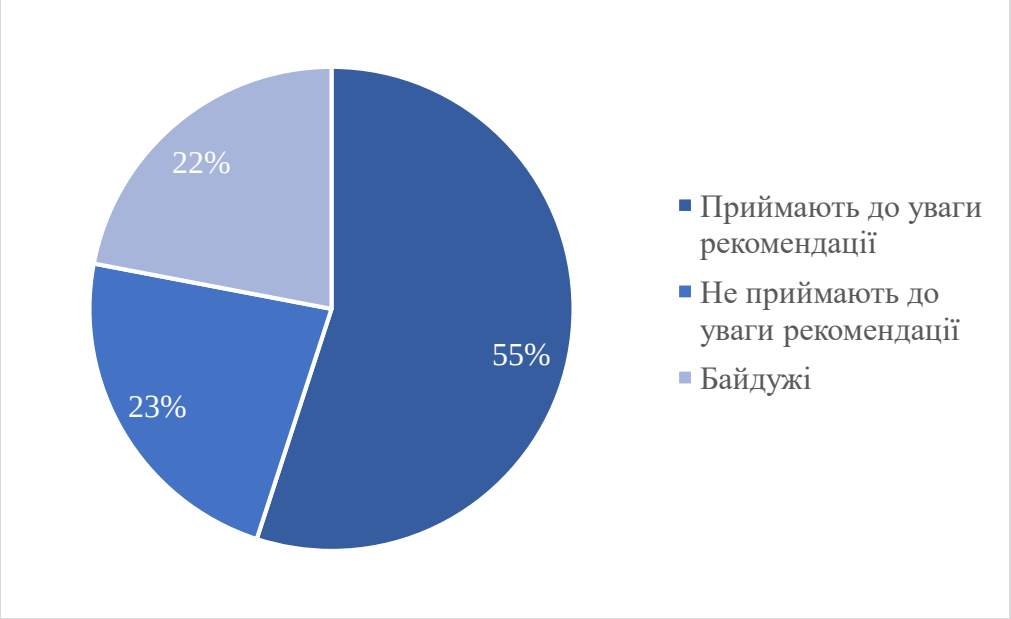

Рисунок 1.2 – Діаграма важливості рекомендацій для покупців

Виходячи з вище наведених даних зробимо висновок, що більше 50% користувачів інтернету, що здійснюють комерційну активність, користуються рекомендаціями інших покупців.

Також, розглянемо ринок музичних онлайн-сервісів (рис. 1.3). Користувачі намагаються використовувати сервіси, що пропонують їм рекомендації і формують плейліст із треків наперед.

Сервісом YouTube користується найбільша частка користувачів (82,9%), тому він займає домінуюче положення на ринку музикальних сервісів, не звертаючи увагу на те, що в першу чергу – це відеохостинг але багато людей використовують його для прослуховування музикальних треків.

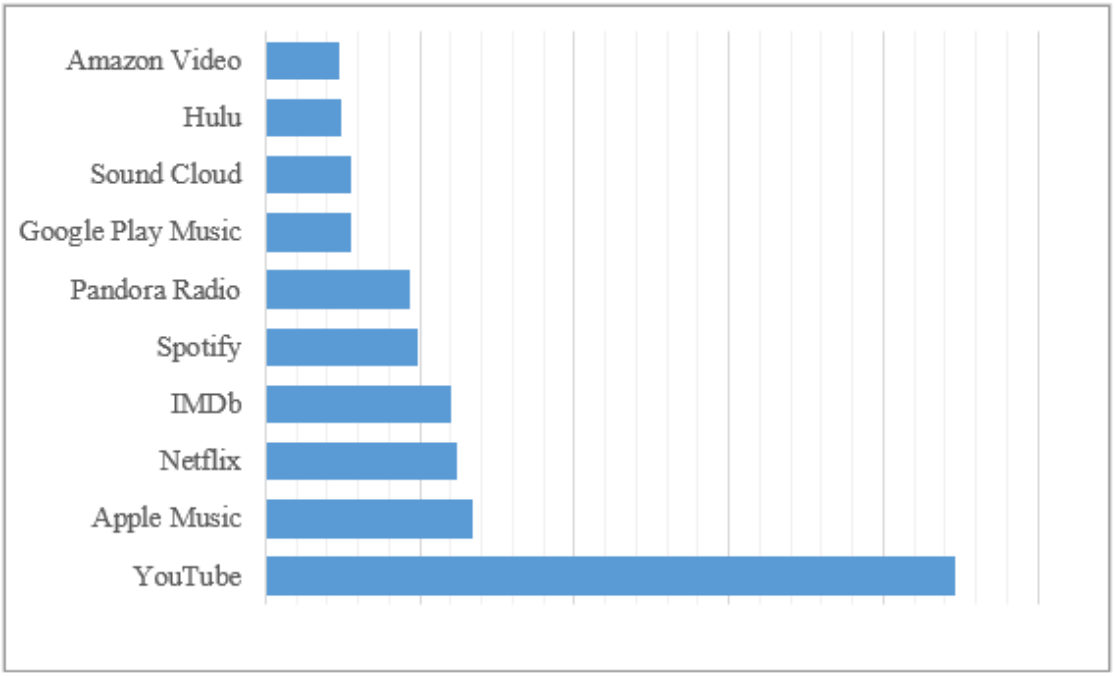

Рисунок 1.3 – Найбільш популярні музикальні та відео сервіси за часткою користувачів

Всі наведені вище ресурси використовують рекомендаційні системи, для збереження часу користувачів та надання цілеспрямованого об’єкту пошуку. В контексті статистичних даних інтернет-мережі тенденції до використання нових технологій змінюються, тому що є потреба в оптимізації користувацьких сервісів та надання послуг нової якості, особливо, рекомендацій. Для поглиблення в дослідження рекомендаційних систем з’ясуємо їх види та основні поняття.

2. АНАЛІЗ ТЕХНОЛОГІЙ І МЕТОДІВ ПОБУДОВИ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

2.1 Основні поняття та види рекомендаційних систем

Для розглядання алгоритмів, введемо наступні поняття:

Об’єкт – це те, що споживають користувачі системи рекомендації. Прикладами є пісня, фільм, товар, навіть інший користувач. Іншими словами: це що треба рекомендувати.

Користувач – це особа, що зареєстрована в системі, може створювати певну активність (купувати, слухати, коментувати, оцінювати) до об’єктів та користуватися сервісом.

Рекомендація – це один або декілька об’єктів, що система пропонує користувачу.

Основна задача рекомендаційної системи: аналізуючи інформацію про активність користувачів, надати один або декілька об’єктів, що є найбільш оптимальним для конкретного користувача [8].

У сфері рекомендацій створюються новітні алгоритми, що допомагають користувачам обирати товари. Різні підходи використовуються для задоволення специфічних потреб людей.

1. Асоціативні правила (Association Rules) – Асоціативні правила зазвичай використовуються при аналізі продуктових кореляцій (Market Basket Analysis) і виглядають приблизно так «якщо в чеку клієнта є молоко, то в 80% випадків там буде і хліб». Тобто якщо ми бачимо, що молоко в кошик клієнт вже поклав, саме час нагадати про хліб. Це не те ж саме, що аналіз рознесені в часі покупок, але якщо ми будемо вважати всю історію однієї великим кошиком, то цілком можемо

застосувати цей принцип і тут. Це може бути виправдано, коли ми, наприклад, продаємо дорогі разові товари (кредит, політ).

2. Обмежені машини Больцмана (restricted Bolzman Machines) – відносно старий підхід, заснований на стохастичних рекурентних нейронних мережах. Він являє собою модель з латентними змінними і в цьому схожий на SVD-розкладання. Тут також шукається найбільш компактний опис користувацьких переваг, яке кодується за допомогою латентних змінних. Метод не був розроблений для пошуку рекомендацій, але він успішно використовувався в топових рішеннях Netflix Prize і до сих пір застосовується в деяких завданнях.

3. Автоенкодери (autoencoders) – в основі лежить все той же принцип спектрального розкладання, тому такі мережі ще називають denoising auto-encoders. Мережа спочатку згортає відомі їй дані про користувача в деякий компактне уявлення, намагаючись залишити тільки значиму інформацію, а потім відновлює дані у вихідній розмірності. У підсумку виходить якийсь усереднений, очищений від шуму шаблон, за яким можна оцінити інтерес до будь-якого продукту.

4. DSSM (deep semantic similiarity models) – один з нових підходів. Все той же принцип, але в ролі латентних змінних тут внутрішні тензорні опису вхідних даних (embeddings). Спочатку модель створювалася для матчінга запиту з документами (як і content-based рекомендації), але вона легко трансформується в завдання матчінга користувачів і товарів.

На практиці рідко використовується тільки один підхід. Як правило кілька алгоритмів комбінуються в один, щоб досягти максимального ефекту.

Дві основні переваги об'єднання моделей – це збільшення точності і можливість більш гнучкого налаштування на різні групи клієнтів. Недоліки – менша интерпретируемость і більша складність реалізації і підтримки.

Кілька стратегій об'єднання:

1. Weighting – вважати середньозважений прогноз по кільком оцінками.

2. Stacking – передбачення окремих моделей є входами іншого (мета) класифікатора, який навчається правильно зважувати проміжні оцінки.

3. Switching – для різних продуктів / користувачів застосовувати різні алгоритми.

4. Mixing – обчислюються рекомендації по різних алгоритмах, а потім просто об'єднуються в один список.

Зазначимо основні види рекомендаційних систем [9]:

Колаборативна фільтрація (Collaborative Filtering) – рекомендації засновані на історії оцінок як самого користувача, так і інших користувачів. Цей підхід має теоретично високою точністю, але при цьому має одну важливу проблему – високий поріг входу. Існує два підходи до колаборативной фільтрації: засновані на користувачів (user-based) і на предметах (item-based).

Засновані на контенті (content-based) – рекомендації засновані на даних, зібраних про кожен конкретний товар. Користувачеві рекомендуються об'єкти, схожі на ті, якими він раніше цікавився або вже набував. Схожість оцінюється виходячи з вмісту об'єктів. Великий плюс – можливість зацікавити нового користувача пропозиціями з перших споживчих кроків. Для цього не потрібно довго збирати дані про переваги, а можна відразу включити споживача в роботу з ресурсом. Можливо рекомендувати навіть ті об'єкти, які не отримали оцінку інших користувачів. Основні недоліки – сильна залежність від предметної області, зниження точності і обмеженість корисності рекомендацій.

Засновані на знаннях (knowledge-based) – рекомендації засновані на знаннях про предметну область (а не про кожен товар). Такий тип рекомендацій має високу точність, пропонуючи користувачеві те, що йому потрібно. Крім цього, система вивчає і аналізує взаємозв'язки між об'єктами, враховує ряд додаткових опцій, що відносяться до індивідуальних властивостей конкретного користувача. До таких властивостей відносяться призначені для користувача побажання (наприклад, їх використовує Яндекс.Маркет) і демографічні

особливості (вихідні дані, які використовують найбільші соціальні мережі, такі як Facebook, LinkedIn, Вконтакте та інші). Наприклад, система може знати, про те що до плеєра має сенс запропонувати навушники, хоча це два абсолютно різні товари. Їх головна перевага - вони виключають часті ситуації інших систем: "ви купили машину, ми вам пропонуємо ще 5". Основний мінус – складність розробки та збору даних.

Гібридні (hybrid) – рекомендації засновані на комбінуванні колаборативного і тематичних підходів, що дозволяє уникнути більшості недоліків, властивих кожній системі.

Алгоритми для колаборативної фільтрації так само діляться на дві великі групи: анамнестичні (memory-based) і модельні (modelbased).

Анамнестичні алгоритми роблять передбачення про оцінки клієнта, виходячи з усіх попередніх оцінок, зроблених даним клієнтом. Значення оцінки для клієнта і товару вираховується як сукупність оцінок, даних товару іншими користувачами, схожих з даними. Модельні методи будують прогноз про оцінку товару, виходячи не з припущень, а з моделі поведінки, створеної за допомогою різних статистичних аналізів отриманої інформації.

Наведемо простий приклад колаборативної фільтрації:

Таблиця 2.1 – Приклад побудови колаборативної фільтрації

Виконавці	User_1	User_2	User_3	User_4
Nirvana	15	2	8	-
Michael Jackson	8	-	-	1
The Beatles	4	3	7	-
Queen	-	10	-	10
Frank Sinatra	-	11	2	7

Продовження таблиці 2.1 - Приклад побудови колаборативної фільтрації

	Кількість пісень, що були прослухані користувачами			
Кластер	1	2	1	2

Джерело: складено автором на основі [9]

З табл. 2.1 ідентифікуємо відмінності кожного кластера та сформуємо рекомендації для користувачів. До першого кластера відноситься User_1(перший користувач) та User_3 (третій користувач), перший користувач прослухав 8 різних пісень виконавця «Michael Jackson», а третій користувач не слухав виконавця взагалі; третій користувач прослухав 2 пісні «Frank Sinatra», а перший – жодної. Ро бимо висновок, що доцільно порекомендувати третьому користувачу прослухати композиції «Michael Jackson», але для першого неможливо сформулювати рекомендацій, так як невелика різниця між ними стосовно «Frank Sinatra».

Спираючись на цей приклад побудуємо ситуацію, де використовується контента фільтрація:

Таблиця 2.2 – Приклад побудови контентної фільтрації

Виконавці	User_3	Подібний контент
Nirvana	8	Nirvana
Michael Jackson	-	Michael Jackson
The Beatles	7	The Beatles
Queen	-	Проранжовані виконавці
Frank Sinatra	2	

Джерело: складено автором на основі [9]

З табл. 2.2 припустимо, що використовується ранжування виконавців на основі даних про користувачів, які прослухавши «Nirvana», слухають «Michael

Jackson» та «The Beatles». У цьому випадку рекомендуємо третьому користувачу прослухати композиції виконавця «Michael Jackson».

У багатьох рекомендаційних системах, присутня проблема «холодного старту». Користувачеві необхідно оцінити досить велику кількість різних товарів, перш ніж система буде спроможна правильно зрозуміти його переваги і дати йому відповідні рекомендації. Тому система не зможе давати точні рекомендації новому споживачеві, яка провела дуже мало оцінок.

2.2 Алгоритми та методи побудови рекомендаційних систем

Зробимо опис алгоритмів, що використовуються на практиці для реалізації різних підходів.

Алгоритм застосування коллаборативної фільтрації

Нехай в деякій системі існують користувачі та об'єкти (у нашому випадку, музикальні треки). Множина $U_{u_1,u_2,\ldots u_k}$ – користувачі, $M_{m_1,m_2,\ldots m_n}$ – музичні треки, $r_{i,j}$ – оцінка користувача u_i стосовно треку m_j. Приведемо матрицю оцінок згідно наведених даних:

Таблиця 2.3 – Матриця оцінок

user\tracks	m_1	…	m_j	…	m_n
u_1	$r_{1,1}$	…	$r_{1,j}$	…	$r_{1,n}$
u_i	$r_{i,1}$	…	$r_{i,j}$	…	$r_{i,n}$
u_k	$r_{k,1}$	…	$r_{k,j}$	…	$r_{k,n}$

Основна задача алгоритма – передбачити, яку оцінку отримає трек m від конкретного користувача a $(a \in U)$. Алгоритм має наступну структуру:

Для кожного користувача u розраховується кореляція інтересів з користувачем a.

Обираємо найбільш близькі результати до користувача a. Передбачимо оцінку на основі оцінок треку m «сусідями» з минулого шагу.

Для цього алгоритма розраховується міра блізькості, для знаходження найближчих сусідів. Існує декілька способів розрахувати міру блізкості [10, 11].

Наведемо основні з них:

$$sim(u,a) = \frac{\Sigma_{i=1}^{m} r_{a,i} \cdot r_{u,i}}{\sqrt{\Sigma_{i=1}^{m} r_{a,i}} \cdot \sqrt{\Sigma_{i=1}^{m} r_{u,i}}} \quad (2.1)$$

Зазначимо, що $sim(u,a)$ – міра близькості користувачів a та u. $r_{u,i}$ – оцінка треку на матриці оцінок. Якщо користувач не вказав оцінку для певного треку, таке значення відповідає 0.

Існує інша міра близькості між векторами – коефіцієнт кореляції Пірсона:

$$sim(u,a) = \frac{\Sigma_{l}(r_{a,i} - \bar{r}_{a,i})(r_{u.i} - \bar{r}_{u})}{\sqrt{\Sigma_{l \subset I}(r_{a,i} - \bar{r}_{a})^2 \cdot \Sigma_{l \subset I}(r_{u,i} - \bar{r}_{u})^2}} \quad (2.2)$$

Де I – множина об’єктів, що були оцінені як користувачем a, так і користувачем u.

R_a та R_u – це середні оцінки користувачів a та u.

На другому кроці головною метою є вибір користувачів, що мають схожі показники з користувачем a. Не є доцільним вибір усіх користувачів, так як існують користувачі, що не мають нічого спільного з користувачем а та можуть негативно вплинути на результати оцінки. Також слід зауважити, що кількість користувачів обраних на цьому кроці має вплив на обсяг розрахунку в наступному кроці.

Одним з доцільних рішень є встановлення порогу входу, що обчислені на першому кроці. Користувачі, що перевищують поріг, увійдуть до виборки, а інші – ні. Також існує спосіб відбору на основі константі k (k є N).Користувачі

сортуються по спаданню міри близькості та обирається k-користувачів, що пройшли відбір.

Результатом другого кроку отримаємо множину користувачів K, що пройшли відбір і будуть враховані на наступному кроці.

На третьому кроці розраховуємо оцінку, яка буде відповідати треку і користувача a. Розраховуємо її за формулою:

$$p_{a,i} = \bar{r}_a + \frac{\Sigma_{u \subset K}(r_{u,i} - \bar{r}_u) \times sim(a,u)}{\Sigma_{u \subset K} |sim(a,u)|} \quad (2.3)$$

$p_{a,i}$– це передбачувана оцінка користувача a стосовно об'єкту i. Основною є середня оцінка $\bar{r}_a$, а потім додаємо середнє відхилення оцінки інших користувачів із множини K для об'єкту i. Чим ближче користувач u до користувача a, тим сильніше мирі впливу. Маємо зважений коефіцієнт, що відповідає близькості інших користувачів до конкретного користувача a.

Сингулярний розклад матриці (англ. Singular-value decomposition, SVD) – один з важливих методів розкладу (чи діагоналізації) матриці, що застосовується в лінійній алгебрі для обчислення псевдоінверсії, наближення матриці, обчислення рангу матриці та інше.

Визначення: нехай матриця M порядку складається з елементів з поля K, де K – або поле дійсних чисел , або поле комплексних чисел.

Невід'ємне дійсне число σ називається сингулярним числом матриці M тоді і тільки тоді, коли існують два вектора одиничної довжини $u \in K^m$ і $v \in K^n$ такі, що:

$$Mv = \sigma u$$

$$M * u = \sigma v$$

такі вектори uu і v називаються, відповідно, лівим сингулярним вектором і правим сингулярним вектором, відповідним сингулярного числа σ.

Сингулярним розкладанням матриці M порядку $m \times n \epsilon$ розкладання такого вигляду:

$$M = \mathrm{U}\Sigma V^* \tag{2.4}$$

де Σ – матриця розміру m×n з невід'ємними елементами, у якій елементи, що лежать на головній діагоналі – це сингулярні числа (а всі елементи, що не лежать на головній діагоналі, є нульовими), а матриці U (близько m) і V (близько n) – це дві унітарні матриці, що складаються з лівих і правих сингулярних векторів відповідно (а V^* – це поєднане-транспонована матриця до V).

Сингулярне розкладання можна переформулювати в геометричних термінах. Лінійний оператор, що відображає елементи простору $\mathbb{R}^n$ в себе представимо у вигляді послідовно виконуваних лінійних операторів обертання і розтягування. Нехай матриці А поставлений у відповідність лінійний оператор. Тому компоненти сингулярного розкладання наочно показують геометричні зміни при відображенні лінійним оператором А безлічі векторів з векторного простору в себе або в векторний простір іншої розмірності.

Алгоритм градієнтного бустингу – це техніка машинного навчання для задач класифікації і регресії, яка будує модель передбачення в формі набору (ансамблю) моделей, зазвичай дерев рішень. Цей алгоритм будемо використовувати для розрахунку оцінок в рекомендаційній системі, дамо визначення оцінки.

Оцінкою в рекомендаційній системі є ступінь впливу кожного чинника музичного треку на вибір користувача. При цьому машинне навчання використовується для пошуку векторів оцінок. Оцінка знаходиться як скалярний добутку векторів властивостей користувачів і музичних треків:

$$r(\theta) = p_u^T q_i,$$
$$\theta = \{p_u, q_i \mid u \in U,\ i \in I\}, \tag{2.5}$$

де p_u– вектор властивостей користувача, q_i – вектор властивостей музичного треку, θ – скалярний добуток векторів.

Основною задачею є мінімізація похибки:

$$E(\hat{r}(\theta) - r) \to min, \tag{2.6}$$

де r – оцінка музичної композиції, $r \in R$.

Тобто задача полягає у пошуку такого параметра θ, щоб похибка (3) була найменшою. Для підбору параметра θ використовується метод градієнтного спуску. Метод градієнтного бустингу базується на методі градієнтного спуску, тому з'ясуємо основні характеристики цього метода.

Метод градієнтного спуску – це ітеративний алгоритм, який розраховує параметри у зворотньому напрямку градієнта:

$$\theta_{t+1} = \theta_t - \mu\nabla(\theta), \tag{2.7}$$

де $\nabla(\theta)$ – градієнт напрямку, μ – задає швидкість градієнтного спуску.

Найбільш простий в реалізації з усіх методів локальної оптимізації. Має досить слабкі умови збіжності, але при цьому швидкість збіжності достатньо мала (лінійна).

Основним недоліком цього методу є недостатня швидкість обробки даних в онлайн-системах. З огляду на це пропонується об'єднати метод SVD із методом градієнтного бустінгу [14]. Це дозволить скоротити час та збільшити швидкість обробки даних.

Опишемо алгоритм градієнтного бустингу, що буде використовуватись в рекомендаційній системі.

Згідно алгоритму градієнтного бустингу наближене значення функції, що описує вибір користувача в музичній рекомендаційній системі, знаходиться як:

$$y \approx \hat{f}(x),$$
$$\hat{f}(x) = argmin\, L(y, f(x)), \tag{2.8}$$

де $\hat{f}(x)$ – наближена функція, $L(y, f(x))$ – функція втрат.

Слід зауважити, оскільки існує нескінченна кількість функцій $f(x)$, необхідно обмежити множину функцій $f(x, \theta), \theta \in R^n$, де R^n – простір дійсних векторів.

Тоді кінцева функція вподобань користувача за допомогою градієнтного бустінгу розраховується наступним чином:

$$F(x) = \sum_{i=0}^{M} \hat{f}_i(x), \quad (2.9)$$

де M – кількість ітерацій підбору наближеної функції вподобань користувача.

Зауважимо, що наближення $\hat{f}_i(x)$ будується не за допомогою моделі з великою кількістю параметрів, а у вигляді суми функцій, де кожна ітерація покращує наближення, зменшує втрати. Для розв'язку задач, зазначимо обмеження функцій у вигляді $\hat{f}_i(x) = h(x, \theta)$. На кожному кроці для функцій нам знадобиться підбирати оптимальний коефіцієнт $\rho \in R$.

Ця функція можете бути використана для побудови списку музичних треків користувача на основі його вподобань.

Побудуємо алгоритм градієнтного бустингу, що був запропонований Джеромом Фридменом [15]. Зазначимо наступні складові:

1) набір даних $\{(x_i, y_i)\}_{i=1,\dots,n}$.

2) число ітерацій M.

3) обираємо функцію витрат $L(y, f)$.

4) обираємо множину функцій базового алгоритму $h(x, \theta)$.

5) додаткові гіперпараметри $h(x, \theta)$, наприклад, глибина дерева у дерев рішень.

Єдине, що залишилось з'ясувати – початкове наближення. В загальних випадках використовують константне значення. Це значення та оптимальний коефіцієнт знаходять бінарним пошуком, або іншим line search алгоритмом щодо вихідної функції втрат.

Сформуємо основні кроки алгоритму градієнтного бустингу (*GBM – gradient boosting machine)*:

1. Початкова ініціалізація градієнтного бустингу із константним значенням.

$$\hat{f}(x) = \hat{f}_0,\ \hat{f}_0 = \gamma\ ,\ \gamma \in R\ ,$$

$$\hat{f}_0 = \arg\min \sum_{i=1}^{n} L(y_i, \gamma)$$

2. Порахувати псевдо-залишки.

3. Побудувати новий базовий алгоритм як регресію на псевдо-залишках.

4. Знайти оптимальний коефіцієнт при відносно початкової функції втрат.

5. Оновити поточний наближення.

6. Скомпонувати підсумкову GBM модель.

Було зазначено основні кроки алгоритму градієнтного бустингу. Усі наведені алгоритми дозволяють створити певну модель, за допомогою якої користувач отримує рекомендації. Для оцінки якості моделі використовуються різні метрики та підходи. Перейдемо до методів тестування та оцінки моделі.

2.3 Методи тестування та оцінки якості результатів алгоритмів

Тестування рекомендаційної системи – процес непростий і завжди викликає безліч питань, головним чином через неоднозначність самого поняття «якість».

Взагалі, в задачах машинного навчання є два основні підходи до тестування:

- offline тестування моделі на історичних даних за допомогою ретро-тестів;
- тестування готової моделі за допомогою A / B тестування (запускаємо кілька варіантів, дивимося який дає кращий результат).

Обидва ці підходи активно застосовуються при розробці рекомендаційних систем. Почнемо з offline тестування.

Основне обмеження, з яким доводиться стикатися – оцінити точність прогнозу ми можемо тільки на ті товари, які користувач уже оцінив.

Стандартний підхід – це крос-валідація методами leave-one-out і leave-p-out. Багаторазове повторення тесту з усередненням результатів дозволяє отримати більш стійку оцінку якості.

1. leave-one-out – модель навчається на всіх оцінених користувачем об'єктах, крім одного, а тестується на цьому одному об'єкті. Так робиться для всіх n об'єктів, і серед отриманих n оцінок якості обчислюється середнє.

2. leave-p-out – те ж саме, але на кожному кроці виключається p точок.

Всі метрики якості можна умовно розділити на три категорії:

- Prediction Accuracy – оцінюють точність пророкує рейтингу;
- Decision Support – оцінюють релевантність рекомендацій;
- Rank Accuracy метрики – оцінюють якість ранжирування видаються рекомендацій.

На жаль, не існує єдиної рекомендованої метрики на всі випадки життя і кожен, хто займається тестуванням рекомендаційної системи, підбирає її під свої цілі.

Якщо ми візьмемо рекомендаційні системи в онлайн бізнесі, то вони, як правило, переслідує дві (іноді суперечливі) цілі:

1) проінформувати користувача про цікавий товар.

2) спонукати зробити покупку (шляхом розсилки, індивідуальні пропозиції і т.д.).

Lift – показник того, у скільки разів точність моделі перевершує якийсь baseline алгоритм. У нашому випадку baseline алгоритмом може бути просто відсутність рекомендацій. Дана метрика добре відловлює частку інкрементальних покупок і це дозволяє ефективно порівнювати різні моделі.

Кожен алгоритм має свої переваги та недоліки. Для порівняння алгоритмів використовуються різні методи, одним з найбільш відомих є метод точність-повнота (*precision-recall*).

За словами Марселя Карасіоло [9], Precision (точність) – це частка найкращих результатів, що є актуальними та стосовно досліджуваної сфери. Приведемо наступну формулу:

$$precision = \frac{\left|\left\{\begin{matrix} relevant \\ tracks \end{matrix}\right\} \cap \left\{\begin{matrix} retrieved \\ tracks \end{matrix}\right\}\right|}{\left|\left\{\begin{matrix} retrieved \\ tracks \end{matrix}\right\}\right|} \quad (2.10)$$

$$recall = \frac{\left|\left\{\begin{matrix} relevant \\ tracks \end{matrix}\right\} \cap \left\{\begin{matrix} retrieved \\ tracks \end{matrix}\right\}\right|}{\left|\left\{\begin{matrix} relevant \\ tracks \end{matrix}\right\}\right|} \quad (2.11)$$

Застосуємо цей метод та отримаємо множину релевантних та рекомендованих музичних треків. На основі неведених формул, побудуємо діаграму Вієен, яка візуально відображає метод точності і повноти:

Рисунок 2.1 – Множина релевантних та рекомендованих музичних треків.

Зрозуміло, що чим вище точність і повнота, тим краще. Але в реальному житті максимальна точність і повнота недосяжні одночасно і доводиться шукати

якийсь баланс. Тому, хотілося б мати якусь метрику яка об'єднувала б у собі інформацію про точність та повноту нашого алгоритму. Саме такою метрикою є F-міра.

F-міра є гармонійне середнє між точністю і повнотою. Вона прагне до нуля, якщо точність або повнота прагне до нуля.

$$F = 2\frac{Precision*Recall}{Precision+Recall} \quad (2.12)$$

Дана формула надає однакову вагу точності і повноти, тому F-міра буде падати однаково при зменшенні і точності і повноти. Можливо розрахувати F-міру надавши різну вагу точності і повноти, якщо ви свідомо віддаєте пріоритет однієї з цих метрик при розробці алгоритму.

$$F = (\beta^2 + 1)\frac{Precision*Recall}{\beta^2 Precision+Recall} \quad (2.13)$$

де β приймає значення в діапазоні $0 < \beta < 1$ якщо ви хочете віддати пріоритет точності, а при $\beta > 1$ пріоритет віддається повноті. При $\beta = 1$ формула зводиться до попередньої і ви отримуєте збалансовану F-міру [1].

Ми розглянули основні методи оцінки роботи алгоритмів рекомендаційних систем. Для більш якіснішого дослідження використаємо їх на практиці для порівняння результатів алгоритмів. Для подальшої роботи зазначимо основні програмні продукти та бібліотеки, що будуть використовуватись.

Опис програмних продуктів та бібліотек

На ринку програмних продуктів існує безліч рішень у сфера аналізу та дослідження даних. Найбільш відомими мови програмування, що використовуються для аналізу даних є R та Python. В роботі взято за основу мову Python.

Представимо характеристику мови Python. У 1991 році Гвідо ван Россум представив мову програмування Python. З тих часів мова стала надзвичайно популярною мовою программування загального призначення і широко

використовується в співтоваристві фахівців з аналізу даних. Основними версіями є Python 3,6 і Python 2,7 [16].

Python – інтерпретована об'єктно-орієнтована мова програмування високого рівня з строгою динамічною типізацією [17]. Зараз Python набуває популярності у таких сферах як: Data Science, Machine Learning, Artifficial Intelligence та інші. Зазначимо основні переваги:

Python – це дуже популярний, широко використовувана мова програмування загального призначення. Він має великий набір спеціально розроблених модулів і широко використовується розробниками. Багато онлайн-сервіси надають API для Python.

Python дуже простий у вивченні. Низький поріг входження робить його ідеальним першою мовою для тих, хто займається програмуванням.

Такі програмні пакети як pandas, scikit-learn і Tensorflow, роблять Python надійним варіантом для сучасних додатків в області машинного навчання.

Python має безліч інструментів, що допомогають розробнику розробити необхідний продукт, користуючись безліччю різноманітних бібліотек у різних сферах. Але також Python має певний ряд недоліків:

Безпека типів. Python – це динамічно типізований мову, а це значить, що повинно бути обережними при роботі з ним. Помилки невідповідності типів (наприклад, передача рядка (string) в якості аргументу методу, який очікує ціле число (integer) можуть час від часу траплятися.

Наприклад, в разі якщо є конкретні цілі статистичного аналізу і аналізу даних, то великий набір пакетів мови R дає йому перевагу перед Python. Крім того, існують більш швидкі та безпечні альтернативи Python серед мов програмування.

Python є хорошим варіантом в сфері науки про дані (data science), і це твердження справедливе як для початкового, так і для просунутого рівнів роботи в даній області. Велика частина науки про дані зосереджена навколо процесу ETL

(витяг-перетворення-завантаження). Ця особливість робить Python ідеально відповідним для таких цілей мовою програмування. Бібліотеки, такі як Tensorflow від Google, роблять Python дуже цікавою мовою для роботи в області машинного навчання.

Для розв’язку задачі побудови рекомендаційної системи, мова надає спеціалізовані бібліотеки, що активно використовуються в оточенні розробників. Такі бібліотеки дозволяють швидко та якісно розробляти програмне забезпечення та мають всередині частку реалізованих методів, саме тому розробник не потрібно програмувати низькорівневий код. Зазначимо основні бібліотеки та їх характеристику, що використовувались в процесі побудови рекомендаційної системи:

NumPy – це фундаментальний пакет для наукових обчислень з Python. Вона містить, серед іншого: потужний об'єкт N-розмірного масиву, складні (трансляція) функції, інструменти для інтеграції C / C + + і Fortran коду, методи для роботи з лінійною алгеброю, перетворення Фур'є і можливості використання випадкових чисел. Крім очевидних наукових застосувань, NumPy також може бути використаний як ефективний багатовимірний контейнер загальних даних. Можна визначити довільні типи даних [18].

Pandas – це відкрите open-source бібліотека, ліцензована BSD, яка забезпечує високопродуктивні, прості у використанні структури даних та інструменти аналізу даних для мови програмування Python. Python вже давно відмінно підходить для обробки та підготовки даних, але менше для аналізу та моделювання даних. панди допомагає заповнити цю прогалину, надаючи вам змогу виконувати весь процес аналізу даних у Python без необхідності перемикання на більш конкретну мову, подібну до P [19].

Scikit-learn – це безкоштовна бібліотека для вивчення машинного програмного забезпечення для мови програмування Python. Вона має різні алгоритми класифікації, регресії та кластеризації, включаючи векторні машини

підтримки, градієнтні підсилювачі, k-means та DBSCAN, і призначені для взаємодії з чисельними та науковими бібліотеками Python NumPy та SciPy [20].

Для напису програм використаємо спеціалізоване програмне забезпечення *PyCharm* – це професійна середовище розробки ПЗ на мові Python, що підтримує велику кількість бібіліотек, надає інструменті для відладки коду та рефакторінгу. Також використано *Jupyter Notebook* для візуалізація процесу виконання програми та надає можливість компілювати код блоками, побачити результат поступово. Для побудови графік в програмному середовищі використано бібліотеку *matplotlib* та інші базові бібліотеки.

Усі зазначені інструменти дозволяють швидко маніпулювати даними, очищувати та трансформувати у необхідний формат. Середовище розробки спрощує написання коду та допомагає дотримуватись конвенцій написання коду (інтегрований протокол *PEP8)* та своєчасно знаходити вразливості та недоліки написаної програми.

Для зберігання та внесення даних скористаємось базою даних. Було обрано використовувати PostgreSQL.

PostgreSQL – уніфікований сервер баз даних, що має єдиний движок – storage engine, використовує клієнт-серверну модель [30]. PostgreSQL – це об'єктно-реляційна база даних, кожна таблиця в ній представляє клас, між таблицями реалізовано успадкування. Для кожного клієнта на сервері створюється новий процес. Для роботи з такими клієнтськими процесами сервер використовує семафори. Клієнтський запит проходить наступні стадії:

1) коннект.

2) парсинг: перевіряється коректність запиту і створюється дерево запиту (*query tree*).

3) перезапис: береться дерево запитів і перевіряється наявність в ньому правил (*rules*), які лежать в системних каталогах.

4) оптимізатор: на кожен запит створюється план запиту - query plan, який передається виконавцю - *executor*. Сенс плану в тому, що в ньому перебираються всі можливі варіанти отримання результату і вибирається найшвидший варіант.

5) виконання запиту: виконавець рекурсивно проходить по дереву і отримує результат.

СУБД PostgreSQL повинна обслуговувати запити sql і повертати результати клієнтським застосуванням. Для цього механізм СУБД оптимізатор запитів приймає рішення про оптимальне використання ресурсів і будує план запиту. При цьому він може спертися на використовуваних механізми прискорення роботи (індексів, кешей в пам'яті, ресурсів процесора, дані статистики і т.д.). Результати запитів повертаються клієнтам [30].

Зазначимо основні характеристики СУБД PostgreSQL (табл. 2.5).

Таблиця 2.5 – Розміри і параметри бази даних

Максимальний розмір БД	Необмежений
Максимальний розмір таблиці	32 TB
Максимальний розмір запису (рядка) в таблиці	1.6 TB
Максимальний розмір поля в запису (рядку)	1 GB
Максимальна кількість записів (рядків) в таблиці	Не обмежено
Максимальна кількість полів (колонок) в таблиці	250 - 1600 залежно від типу даних в колонці
Максимальна кількість індексів на таблицю	Не обмежено

За допомогою БД формується процес аналізу даних. Кожен процес аналізу даних має певних набір етапів алгоритму, що має наступну структуру:

1. Постановка завдання - на цьому етапі затверджуються основні цілі ІА, яких результатів потрібно досягти, вибір об'єкта аналізу та дослідження його стану.

2. Підготовка даних - збір інформації з первинних і вторинних джерел, які в рамках поставленого завдання актуальні і значущі. Такими даними можуть бути файли різних форматів (.txt., .doc, .bak, .mdf і інші), а також сторонні бази даних і сховища, html-сторінки.

3. Вивчення та очищення даних - перетворення даних в формат, який відповідає постановленої моделі аналізу. Трансформація даних і подальша завантаження в середу аналізу або підготовлені платформи для аналізу.

4. Побудова моделей - створення структури аналізу даних, в яку вносяться підготовлена інформація. Такі структури містять алгоритм інтерпретації даних і використовують графічний інтерфейс для відображення результату.

5. Перевірка моделей - оцінка точності моделі, тестування результатів на адекватність і значущість.

6. Розгортання і оновлення моделі - створення і впровадження моделі на реальних даних, можливість оновлювати модель і регулювати налаштування моделі, а також використання моделі для аналітики.

Головний об'єкт інтелектуального аналізу даних - це інформація. Чим більше даних отримає модель, тим точніше буде проведені результати. Основні джерела даних:

- офіційні джерела на сайті компанії – звітності, новини та інформація про емітента;
- неофіційні джерела(електронні ресурси) – це новинні сайти, блоги про страхову діяльність у сфері життя;
- інформація з інсайдерських джерел компанії – це інформація, що є внутрішньою та використовується тільки робітниками компанії або довіреними лицями компанії.

3. ПОБУДОВА МУЗИЧНОЇ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ

3.1 Завантаження бібліотек, вибір та збір даних для проведення аналізу

Для побудови рекомендаційної системи музичних треків було проведено аналіз існуючих наборів даних (датасетів) для подальшого аналізу. В інтернеті існує безліч ресурсів, які надають дані для завантаження в різних форматах: csv, xml, json, txt. За допомогою посилань або API розробник здатні завантажити та обробити ці дані з метою дослідження.

В даній роботі було обрано використання даних Turi machine learning platform [21] – інтернет-ресурс, що є сховищем для великої кількості датасетів для різних типів аналізу. Було обрано два файли музичної тематики: 10000.txt та song_data.csv.

Перший файл відображає активність користувачів: відношення користувача та прослуханої пісні. Складається з наступник полів:

- user_id (guid) – унікальний ідентифікатор користувача;
- song_id (guid) – унікальний ідентифікатор пісні;
- listen_count (int) – цілочисельне значення, що відображає скількі разів користувач прослухав одну пісню.

Другий файл –це словник пісень, що відображає наявні пісні у першому файлі та інформацію (метадані) про них:

- song_id (guid) – уникальний ідентифікатор пісні;
- title (string) – назва пісні;
- release_by (string) – назва компанії, що випустила пісню;
- artist_name (string) – назва виконавця пісні.

Ці файли пов’язані між собою та мають необхідний розріз даних для побудови рекомендаційної системи.

Першим чином, завантажимо необхідні бібліотеки до проекту:

```
import pandas
from sklearn.cross_validation import train_test_split
import numpy as np
import time
from sklearn.externals import joblib
import Recommenders as Recommenders
import Evaluation as Evaluation
```

Рисунок 3.1 – Імпортування бібліотек, необхідних для дослідження

Можна побачити два класи, що не входять до списку стандартних бібліотек, вони приведені в додатках: *Recommenders.py* (Додаток 1) та *Evaluation.py* (Додаток 2), вони використовуються для основної логіки програми: будуть рекомендації стосовно обраних даних (перший файл) та проводять оцінку результатів (другий файл).

Наступним кроком завантажимо пісні:

```
#Читування userid-songid-listen_count triplets
#Завантажуємо список пісень. Цей крок може зайняти деякий час.
triplets_file = 'https://static.turi.com/datasets/millionsong/10000.txt'
songs_metadata_file = 'https://static.turi.com/datasets/millionsong/song_data.csv'

song_df_1 = pandas.read_table(triplets_file,header=None)
song_df_1.columns = ['user_id', 'song_id', 'listen_count']

#Читування метадати пісень
song_df_2 =  pandas.read_csv(songs_metadata_file)

#З'єднання обох датафреймів вище для створення повного датафрейму для рекомендаційн
ої системи
song_df = pandas.merge(song_df_1, song_df_2.drop_duplicates(['song_id']), on="song_
id", how="left")
```

Рисунок 3.2 – Завантаження набору даних пісень.

На цьому етапі завантажуємо дані за допомогою бібліотеки pandas. Також отримаємо song_df, який на основі злиття отримав дані обох файлів. Викликом команди song_df.head() отримаємо перші дані для прикладу.

На табл. 3.1 відображено структуру отриманого спільного документу, що використовується для подальшого аналізу. Довжина масиву складає 2 млн. записів про діяльність користувачів.

Таблиця 3.1 – Завантажений набір даних пісень

	user_id	song_id	listen_ count	title	release	artist_ name	year
0	b80344d0 63b5ccb3 212f7653 8f3d9e43d 87dca9e	SOAKIMP12 A8C130995	1	The Cove	Thicker Than Water	Jack Johnson	0
1	b80344d0 63b5ccb3 212f7653 8f3d9e43d 87dca9e	SOBBMDR1 2A8C13253B	2	Entre Dos Aguas	Flamenco Para Niños	Paco De Lucia	1976
2	b80344d0 63b5ccb3 212f7653 8f3d9e43d 87dca9e	SOBXHDL12 A81C204C0	1	Stronger	Graduatio n	Kanye West	2007
3	b80344d0 63b5ccb3 212f7653 8f3d9e43d 87dca9e	SOBYHAJ12 A6701BF1D	1	Constell ations	In Between Dreams	Jack Johnson	2005
4	b80344d0 63b5ccb3 212f7653 8f3d9e43d 87dca9e	SODACBL12 A8C13C273	1	Learn To Fly	There Is Nothing Left To Lose	Foo Fighters	1999

Джерело: розрахунки автора на основі [21]

Для тестової вибірки виберемо 20 тис. записів. З якою в подальшому будемо працювати.

```
song_df = song_df.head(20000)

#Об'єднаємо назву пісні та артисту для зменшення об'єму вибірки
song_df['song'] = song_df['title'].map(str) + " - " + song_df['artist_name']
```

Рисунок 3.3 – Вибір даних та трансформація стовпців

Завантажені дані готові для подальшого аналізу та застосування на практиці.

3.2 Моделювання алгоритму, заснованого на популярності (наївний метод)

Перший метод, який буде застосований – це метод заснований на популярності пісень. Такий метод є наївним, він використовує дані про кількість прослуховувань пісень відносно до загальної кількості. Популярними є ті пісні, що користувачі обирають найчастіше. Приведемо лістинг коду (рис. 3.4).

```
#Групуємо дані відповідно до пісень
song_grouped = song_df.groupby(['song']).agg({'listen_count': 'count'}).reset_index
()
#Рахуємо суму всіх прослуховувань
grouped_sum = song_grouped['listen_count'].sum()
#Рахуємо частку кожної пісні
song_grouped['percentage']  = song_grouped['listen_count'].div(grouped_sum)*100
song_grouped.sort_values(['listen_count', 'song'], ascending = [0,1])
```

Рисунок 3.3 – Створення списку найпопулярніших пісень у датасеті

Виведемо десять найбільш популярних пісень для формування рейтингу:

Таблиця 3.2 – Список десяти найбільш популярних пісень із набору даних

	song	listen_count	percentage
5183	Sehr kosmisch - Harmonia	79	0.395
1514	Dog Days Are Over (Radio Edit) - Florence + Th...	69	0.345
4918	Revelry - Kings Of Leon	65	0.325
6603	Undo – Björk	65	0.325
5175	Secrets - OneRepublic	64	0.320
7185	You're The One - Dwight Yoakam	60	0.300
2631	Horn Concerto No. 4 in E flat K495: II.Romanc...	53	0.265
1972	Fireflies - Charttraxx Karaoke	52	0.260
6199	The Scientist - Coldplay	50	0.250
2547	Hey_ Soul Sister - Train	45	0.22

Джерело: власні розрахунки автора на основі [21]

В наборі даних знайдено 7242 унікальні композиції, кількість унікальних користувачів складає 756 осіб. Використаємо метод *train_test_split()* для поділу даних на два масиви: для тренування (train_data) та тестування (test_data):

```
train_data, test_data = train_test_split(song_df, test_size = 0.20, random_state=0)
```

Рисунок 3.5 – Розподіл вибірки на масиви для тренування та тестування

За допомогою класу *Recommenders.py* створимо екземпляр рекомендаційної системи на основі популярності. Використаємо систему для опрацювання результатів для окремих користувачів.

```
pm = Recommenders.popularity_recommender_py()
pm.create(train_data, 'user_id', 'song')
user_id = users[5]
pm.recommend(user_id)
```

Рисунок 3.6 – Створення системи на основі популярності

Отримаємо наступну таблицю:

Таблиця 3.3 – Список рекомендацій на основі наївного методу

user_id	song	score	rank
1	2	3	4
4692	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Sehr kosmisch - Harmonia	66
1338	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Dog Days Are Over (Radio Edit) - Florence + Th...	56
4445	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Revelry - Kings Of Leon	56
6520	4bd88bfb25263a75bbdd467e74018f4ae570e5df	You're The One - Dwight Yoakam	51
4684	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Secrets - OneRepublic	47
5988	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Undo - Björk	47
1753	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Fireflies - Charttraxx Karaoke	44
2355	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Horn Concerto No. 4 in E flat K495: II. Romanc...	40
5615	4bd88bfb25263a75bbdd467e74018f4ae570e5df	The Scientist - Coldplay	40

Продовження таблиці 3.3 – Список рекомендацій на основі наївного методу

5824	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Tive Sim - Cartola	37

Джерело: власні розрахунки автора на основі [21]

Зробимо висновок, що таке система не враховує особистих потреб користувачів, а тільки будує рекомендаційний список на основі думки більшості користувачів сервісу та формування рейтингу для різних користувачів генерує ідентичні вибірки даних. Перейдемо до методів коллаборативної фільтрації.

3.3 Моделювання алгоритму, заснованого на моделі колаборативної фільтрації

Для побудови колаборативної моделі застосуємо методи оцінки схожості елементів в класі *Recommenders.py*.

```
is_model = Recommenders.item_similarity_recommender_py()
is_model.create(train_data, 'user_id', 'song')
```

Рисунок 3.7 – Створення моделі на основі колаборативної фільтрації

Для формування колаборативної фільтрації, використаємо метод, заснований на предметах (item-based). На рис. 3.8 зображено застосування методу, заснованого на предметах.

```
#пісні для користувача, взяті з датасету
user_id = users[5]
user_items = is_model.get_user_items(user_id)

print("------------------------------------------------------------------------------------
--------")
print("Training data songs for the user userid: %s:" % user_id)
print("------------------------------------------------------------------------------------
--------")

for user_item in user_items:
    print(user_item)

print("----------------------------------------------------------------------")
print("Recommendation process going on:")
print("----------------------------------------------------------------------")

#рекомендації щодо пісень на основі персональної моделі
is_model.recommend(user_id)
```

Рисунок 3.8 – Застосування методу заснованого на предметах

На цьому етапі обираємо конкрентного користувача (користувач з ідентифікатором 5) та отримаємо список пісень, які користувач вже прослухав:

```
----------------------------------------------------------------------
Push It - Salt-N-Pepa
Speechless - Lady GaGa
Just Lose It - Eminem
Without Me - Eminem
Ghosts 'n' Stuff (Original Instrumental Mix) - Deadmau5
16 Candles - The Crests
Missing You - John Waite
Say My Name - Destiny's Child
My Dad's Gone Crazy - Eminem / Hailie Jade
The Real Slim Shady - Eminem
Somebody To Love - Justin Bieber
Forgive Me - Leona Lewis
Ya Nada Queda - Kudai
----------------------------------------------------------------------
```

Рисунок 3.9 – Список пісень, що були прослухані користувачем

На основі цих пісень скористаємось алгоритмом схожості та побудуємо список пісень, що є оптимальними для наступних рекомендацій:

Таблиця 3.4 – Список рекомендацій на основі колаборативної фільтрації

user_id	song	score	rank
1	2	3	4
4bd88bfb25263a75bbdd467e74018f4ae570e5df	Mockingbird - Eminem	0.052260	1
4bd88bfb25263a75bbdd467e74018f4ae570e5df	Heartbreak Warfare - John Mayer	0.050858	2
4bd88bfb25263a75bbdd467e74018f4ae570e5df	Superman - Eminem / Dina Rae	0.050789	3
4bd88bfb25263a75bbdd467e74018f4ae570e5df	One Time - Justin Bieber	0.050047	4
4bd88bfb25263a75bbdd467e74018f4ae570e5df	U Smile - Justin Bieber	0.046741	5
4bd88bfb25263a75bbdd467e74018f4ae570e5df	Te Amo - Rihanna	0.044500	6
4bd88bfb25263a75bbdd467e74018f4ae570e5df	Eenie Meenie - Sean Kingston and Justin Bieber	0.044160	7
4bd88bfb25263a75bbdd467e74018f4ae570e5df	PapePlanes - M.I.A.	0.043189	8
4bd88bfb25263a75bbdd467e74018f4ae570e5df	Terre Promise - O'Rosko Raricim	0.042895	9
4bd88bfb25263a75bbdd467e74018f4ae570e5df	Brave The Elements - Colossal	0.042287	10

Джерело: власні розрахунки автора на основі [21]

В даному випадку отримали список рекомендацій для конкретного користувача. На основі цієї моделі реалізуємо персональну рекомендаційну модель для отримання схожих пісень у наборі даних.

```
song = 'Yellow - Coldplay'
is_model.get_similar_items([song])
```

Рисунок 3.10 – Реалізація рекомендаційної системи для отримання схожих наборів даних

Отримаємо наступний список схожих пісень:

Таблиця 3.5 – Застосування моделі персональних рекомендацій

	song	score	rank
1	2	3	4
0	Clocks – Coldplay	0.300000	1
1	Speed Of Sound – Coldplay	0.200000	2
2	What Goes Around...Comes Around - Justin Timbe...	0.184211	3
3	Halo – Beyoncé	0.177778	4
4	Hotel California – Eagles	0.166667	5
5	Fix You – Coldplay	0.166667	6
6	In My Place – Coldplay	0.157895	7
7	Seven Nation Army - The White Stripes	0.153846	8
8	I Kissed A Girl - Katy Perry	0.153846	9
9	Paper Planes - M.I.A.	0.150000	10

Джерело: власні розрахунки автора на основі [21]

3.4 Моделювання алгоритму, заснованого на методі градієнтного бустингу

Для побудови списку пісень на основі градієнтного бустингу є потреба розрахувати SVD-розклад матриці, де рядки є користувачами, а стовпці – музикальною композицією. Значенням виступає показник listen_count, тобто скільки разів користувач прослухав кожну пісню. Імпортуємо необхідні бібліотеки:

```
import math as mt
import csv
from sparsesvd import sparsesvd #використовується для факторизації матриць
import numpy as np
from scipy.sparse import csc_matrix
from scipy.sparse.linalg import * #використовується для множення матриць
```

Рисунок 3.11 – Імпортування необхідних бібліотек для побудови сингулярного розкладу матриці

Для впровадження обчислення SVD-розкладу виконаємо наступний код:

```
#використовуємо SVD обчислення (кількість латентних факторів)
K=2

urm = csc_matrix(song_df, dtype=np.float32)

#обчислюємо SVD матриці оцінок
U, S, Vt = computeSVD(urm, K)

#отримання очикуваної оцінки для користувача
print("Predictied ratings:")
matrix_items = computeEstimatedRatings(urm, U, S, Vt, uTest, K, True)
print(matrix_items)
```

Рисунок 3.12 – Обчислення SVD-розкладу матриці

Для розрахунку були використані методи розрахунку матриць SVD *computeSVD* та *computeEstimatedRatings* (Додаток 3). Результати SVD дають 3 матриці на виході, S and Vt. Матриця U показує вектори користувачів, а матриця

Vt - вектори пісень. Для наглядного прикладу наведемо зображення значень матриці у семантичному просторі 5-ти користувачів на основі 5-ти пісень.

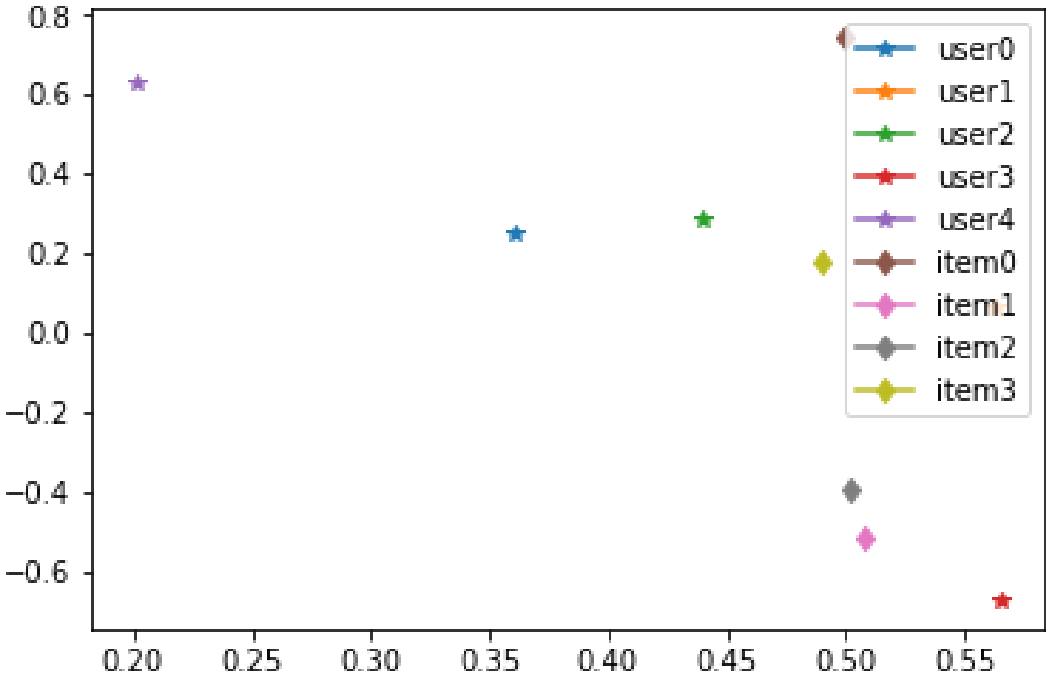

Рисунок 3.13 – Зображення вектору користувачів у семантичному просторі

Матриця U показує користувачів як 2D точки в латентному просторовому векторі, а Vt - показує пісні як 2D точки у тому самому векторі. Далі ми будуємо матриці U, S та Vt та інтерпретуємо їх. Для прикладу програма знизу зображує графіки усіх векторів користувачів з матриці U в 2D просторі. Також побудовано на тому графіку вектори пісень, які отримані з матриці Vt.

Розраховані матриці використаємо для побудови моделі на основі градієнтного бустингу. Для впровадження моделі завантажимо наступні бібліотеки:

```
import xgboost as xgb
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib as mpl
from xgboost import XGBClassifier
```

Рисунок 3.14 – Завантаження бібліотек для методу градієнтного бустингу

Побудуємо модель градієнтного бустингу з наступними константами learning_rate=0.1, max_depth=10, min_child_weight=10, n_estimators=250. Значення констант задані на основі проведеного дослідження та застосуванням досвіду інших дослідників [13, 14, 15]. Побудуємо модель градієнтного бустинга.

```
gradient_boosting = XGBClassifier(learning_rate=0.1, max_depth=10, min_child_weight
=10, n_estimators=250)
gradient_boosting.fit(test_data, train_data)
model.score(test_data, train_data)#точність навчання
```

Рисунок 3.15 – Побудова моделі градієнтного бустингу на основі введених даних

На вхід моделі подається тестова та навчальна вибірка (test_data, train_data), розмір тестової вибірки складає 20% набору даних. Для оцінки точності навчання було виконано метод gradient_boosting.score() та отримано значення 0.73. Отже, модель є адекватною та значущою. За допомогою даної моделі побудуємо список рекомендацій пісень для користувача з user_id = 3.

```
predictions = np.zeros(shape=[len(song_df)])
predictions+= gradient_boosting.predict(song_df[3].drop(['id'],axis=1))
```

Рисунок 3.16 – Формування списку композицій за допомогою моделі

Отже, отримаємо список рекомендацій із використанням виводу моделі, які наведені у таблиці 3.6.

Таблиця 3.6 – Список рекомендацій із використанням методу градієнтного бустингу

	song	score	rank
1	2	3	4
0	Creep (Explicit) - Radiohead	0.458234	1
1	In The End - Linkin Park	0.410043	2
2	Wasted - Kasabian	0.375499	3
3	Rolling Stone - Hurts	0.356532	4
4	Meds - Placebo	0.349084	5
5	Heart-Shaped Box - Nirvana	0.340062	6
6	Sonne - Rammstein	0.336521	7
7	Counting Stars - OneRepublic	0.335633	8
8	Youth Against Fascism - Sonic Youth	0.334195	9
9	Bulletproof - La Roux	0.332097	10

Джерело: власні розрахунки автора на основі [21]

На основі введених даних отримали список пісень, що відповідають вподобанням користувача з user_id = 3, що є наочним прикладом використання моделі.

Для подальшого порівняння застосуємо методи оцінки якості сформульованих моделей. Перейдемо до оцінки наведених моделей.

3.5 Порівняння якостей результатів моделей

Для порівняння результатів двох рекомендаційних систем, скористаємось бібліотекою *Evaluation.py* та методом *precision_recall_calculator* – використовується для розрахунку показника точності та повноти.

```
# Визначення кількості користувачів для обчислення точності та повноти
 user_sample = 0.05

# Створюємо екземпляр класу
 pr = Evaluation.precision_recall_calculator(test_data, train_data, pm, is_model)

# Виклик метода для обчислення показника точності-повноти
 (pm_avg_precision_list, pm_avg_recall_list, ism_avg_precision_list, ism_avg_recall
_list) = pr.calculate_measures(user_sample)
```

Рисунок 3.17 – Розрахунок показника точності і повноти

На основі отриманих даних, скористаємось бібліотекою *pylab* для візуалізації отриманих результатів. Сформулюємо метод для відображення кривої точності та повноти.

```
import pylab as pl

#Метод для генерації кривої точності та повноти
def plot_precision_recall(m1_precision_list, m1_recall_list, m1_label, m2_precision
_list, m2_recall_list, m2_label):
    pl.clf()
    pl.plot(m1_recall_list, m1_precision_list, label=m1_label)
    pl.plot(m2_recall_list, m2_precision_list, label=m2_label)
    pl.xlabel('Recall')
    pl.ylabel('Precision')
    pl.ylim([0.0, 0.20])
    pl.xlim([0.0, 0.20])
    pl.title('Precision-Recall curve')
    #pl.legend(loc="upper right")
    pl.legend(loc=9, bbox_to_anchor=(0.5, -0.2))
    pl.show()

#Виклик метода для відображення графіку
plot_precision_recall(pm_avg_precision_list, pm_avg_recall_list, "popularity_model"
, ism_avg_precision_list, ism_avg_recall_list, "item_similarity_model")
```

Рисунок 3.18 – Побудова кривої точності і повноти

Отримаємо графік, наведений на рис. 3.19.

В результаті виконання методу *plot_precision_recall()* отримали графік, що показує оцінені значення моделей. Як можна побачити, на цьому графіку

значення повноти моделі на основі колаборативної фільтрації (0.125 < *Recall* < 0.18) значно перевищує першу модель, де значення знаходяться в діапазоні 0.015 < *Recall* < 0.025. Однак значення показника точності знаходяться у однаковому проміжку 0.02 < *Precision* < 0.125.

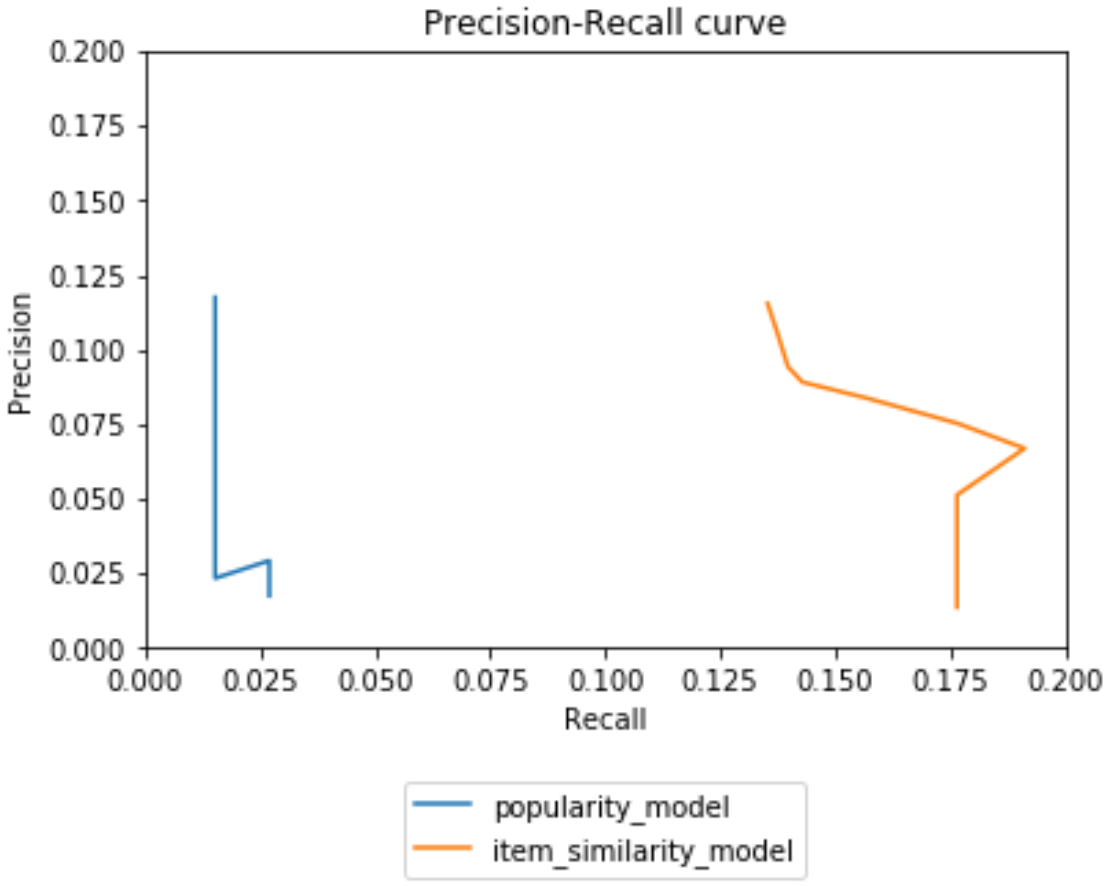

Рисунок 3.19 – Результат оцінки точності-повноти

Проведемо експеримент оцінки на більшій кількості даних (рис. 3.20). Криві точності вилучення графіків для більшої підмножини даних (100 000 рядків) (вибірка користувача = 0,005).

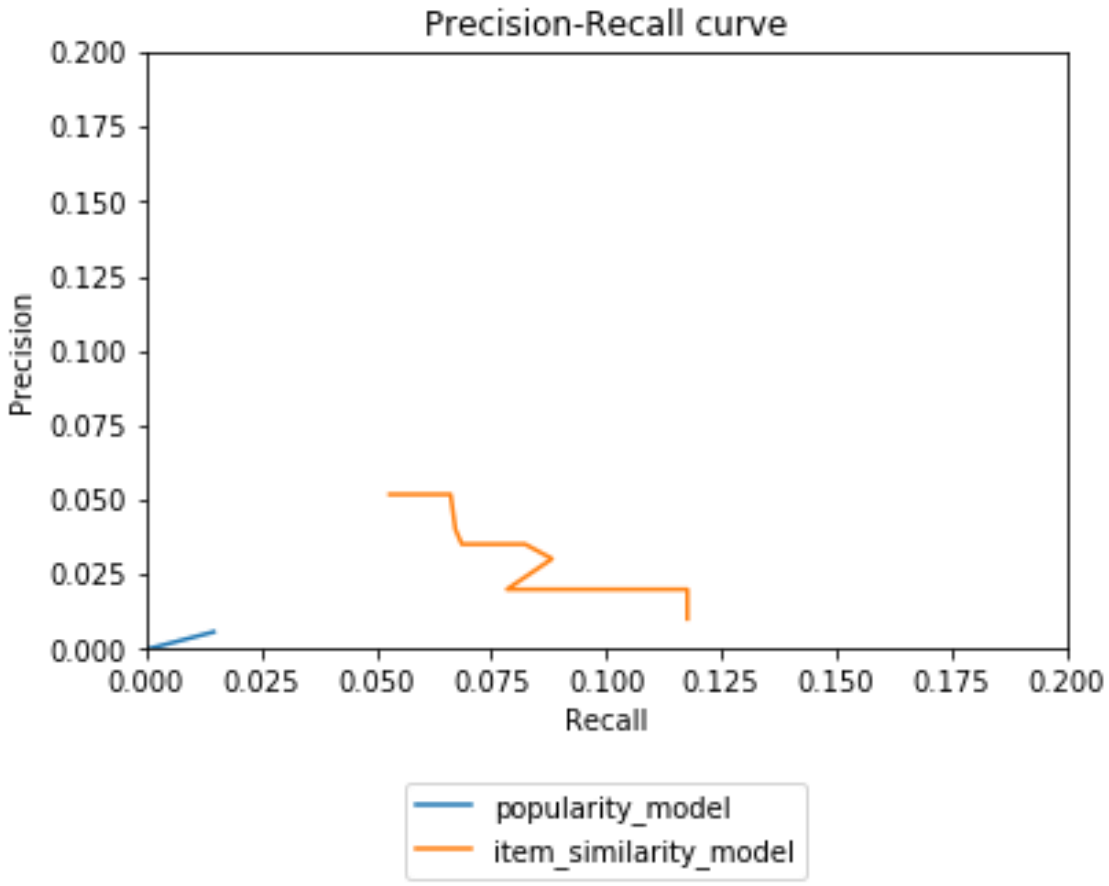

Рисунок 3.20 – Результат оцінки точності-повноти на більшому наборі даних

Після побудови графіку на більшому наборі даних отримали обидві криві і можемо оцінити результат моделей.

Показники моделі, побудованої на популярності є низькими: максимальна точність складає < 0.012, а повнота приблизно < 0.015, що порівняно з другою моделлю значно менший показник. На другій моделі бачимо, що максимально точність складає приблизно 0.05 (у 4 рази більше, ніж у першої моделі), а повнота – 0.12.

Отже, зробимо висновок, що друга модель є значно кращою як на невеликій кількості даних, так і на більшому наборі даних. Перша модель побудована тільки на припущення на загальну популярність пісні та не враховує особисті

вподобання користувачів. Її результат також є релевантним, але має значно менші показники, ніж друга модель.

У другій моделі враховано особисті вподобання користувачів, тобто побудовано модель колаборативної фільтрації на предметах (на піснях). Така модель дає значно кращі результати і може бути використана для впровадження на реальних проектах.

Для перевірки швидкості обробки запитів моделі на основі градієнтного бустингу, використаємо веб-ресурс, що дозволяє завантажити рекомендаційну модель та перевірити її швидкість – Kaggle [29]. Ця платформа використовується спеціалістами з аналізу даних для набори з набори даних та вирішення проблем у сфері аналізу даних. Завантажимо модель у систему та отримаємо результат:

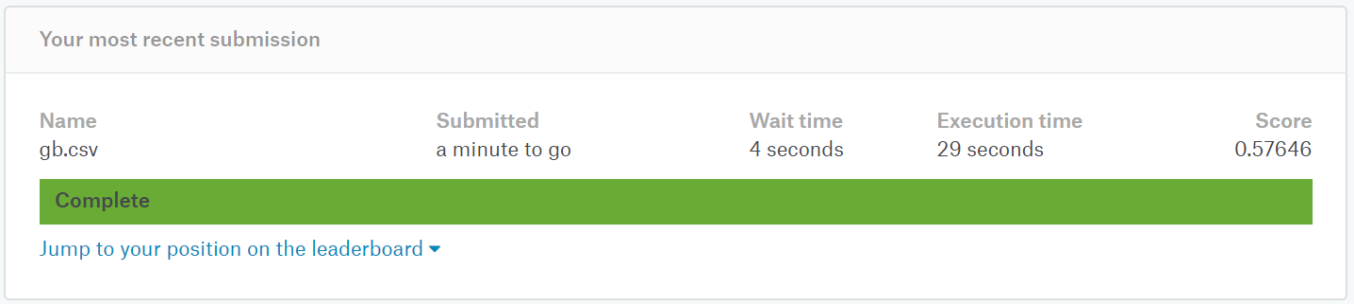

Рисунок 3.21 – Розрахунок швидкості моделі

Як можна побачити з рис. 3.21. швидкість обробки запитів на основі методів градієнтного бустингу із використанням набору даних з 10 тис. користувачів склала 29 секунд. Проведемо подібні розрахунки для інших методів на отримаємо наступні результаті:

Таблиця 3.7 – Розрахунок швидкості роботи методів при наборі даних 1 тис. користувачів

	Середній час, сек	Середня точність, %	Середня повнота, %
Naïve method	14	46.12	41.67
Collaboration Filtering	10	58.22	69.04
Gradient Boosting	9	61.17	64.59

Джерело: власні розрахунки автора за даними [29]

Було отримано розрахунок швидкості роботи методів формування рекомендаційних систем із використання набору даних 1 тис. користувачів. Значення наведені в табл. .

Таким чином, на невеликих даних градієнтних бустинг працює в середньому швидше на 1 секунду, враховуючи, що показники точності (49.12 %) та повноти (63.33 %) є помірними. Проведемо розрахунку на більшому наборі даних.

Таблиця 3.8 - Розрахунок швидкості роботи методів при наборі даних 1 тис. користувачів

	Середній час, сек	Середня точність, %	Середня повнота, %
Naïve method	68	39.08	45.52

Продовження таблиці 3.8 – Розрахунок швидкості роботи методів при наборі даних 1 тис. користувачів

Collaboration Filtering	42	49.55	58.41
Gradient Boosting	29	49.12	63.33

Джерело: власні розрахунки автора за даними [29]

Користуючись отриманими результатами табл. можемо зробити висновок, що нова модель швидше ніж модель класичної колаборативної фільтрації на 13 секунд, наївного методу – 39 секунд.

Отже, перехід до використання методу градієнтного бустингу виявився доволі ефективним кроком для оптимізації роботи алгоритмів створення рекомендаційних списків. За допомогою проведених досліджень було виявлено ряд недоліків існуючих алгоритмів, серед яких: низька швидкість обробки запитів, ефективність та значущість сформованих списків. Для усунення цих недоліків використано метод градієнтного бустингу.

На основі проведених досліджень перейдемо до реалізації демонстраційного прикладу із використанням зазначених методів.

3.6 Побудова архітектури системи та демонстраційного прикладу

На основі наведених даних було реалізовано методи побудови рекомендацій. Для впровадження методів було створено систему, що включає наступні компоненти:

- Front-end система – веб-інтерфейс користувача, що реагує на дії клієнта та надсилає запити на сервер;
- Web Application (Back-end) система – серверна частина, що обробляє запити клієнта, побудована на моделі MVC (*Model-View-Controller*), база даних моделей розташована у сховищі *PostgreSQL*, цей компонент отримує відповідь компонента *Analysis Service* та обробляє його для *Front-end* системи;
- Analysis Service (Модуль навчання моделі) – модуль роботи з моделями та методами рекомендаційних систем, використовує бібліотеки *numpy* та *sckit-learn* для їх реалізації, формує список рекомендацій;
- База даних – це організована структура, призначена для зберігання інформації про користувачів, їх дії та вподобання, побудована за допомогою *PostgreSQL*.

Наведемо схему архітектури отриманої системи:

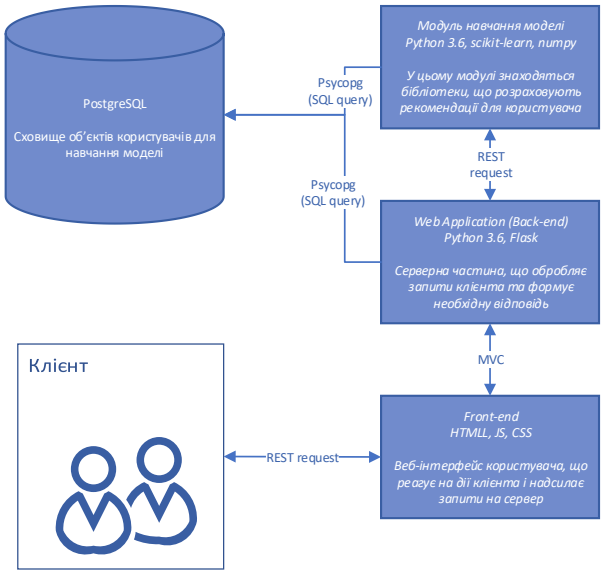

Рисунок 3.22 – Схема рекомендаційної системи

За допомогою цієї системи клієнти отримують бажані результати, що представлені у вигляді списку музичних композицій. Опишемо алгоритм роботи системи:

1. Клієнт відправляє запит на отримання списку рекомендацій, *Front-end* система отримує його у вигляді запита (*REST request*).

2. *Front-end* інтерфейс отримує та обробляє запит, надсилаючи його на *Back-end* систему, чекає відповідь зі списком музичних композицій.

3. Далі запит поступає до модулю *Analysis Service*, який розраховує та формує список музичних композицій для користувача, використовуючи реалізацію вище наведних методів.

4. Back-end система отримує список та готує відповідь для веб-інтерфейсу на основі розрахованих даних *Analysis Service*.

5. У веб-інтерфейсі відображається список рекомендацій, що було сформовано із вхідних даних користувача.

Користуючись схемою рекомендаційної системи та зазначеним алгоритмом роботи системи було зроблено демонстраційний приклад роботи системи у вигляді сайту, на якому клієнт може отримати список рекомендацій.

Система складається з наступних розділів:

1. Опис проекту – початкова сторінка, що описує загальні положення програми та мету створення, також є посилання на «Демо-зону», де користувач може отримати рекомендації;

2. «Демо-зона» – набір графічних елементів взаємодії з рекомендаційною системою, відправлення запитів на сервер *Web Application*.

3. Сторінка розробника – посилання на сторінку автора проекту, його особисті дані та реалізовані проекти.

Перейдемо до розділу «Демо-зона» та більш детально розглянемо функції, що запропоновані клієнту.

На рис. 3.23 зображено інтерфейс для взаємодії із рекомендаційною системою та знаходження схожих виконавців на основі введеного виконавці.

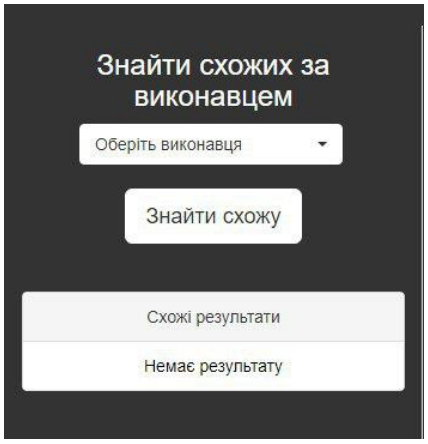

Рисунок 3.23 – Інтерфейс знаходження схожих виконавців

Розглянемо процес вибору вхідних даних на рис. 3.24.

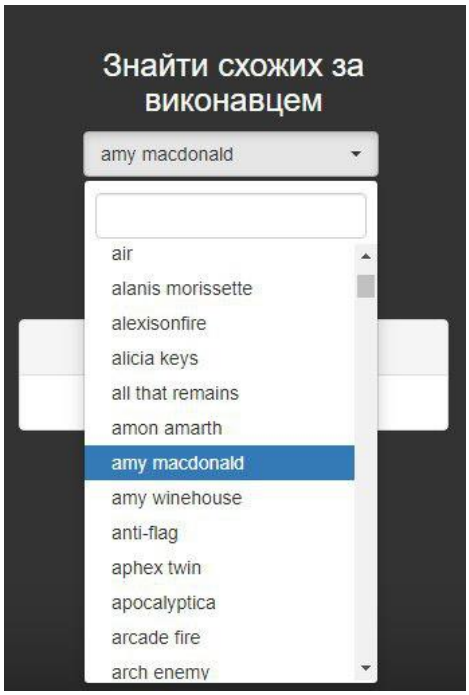

Рисунок 3.24 – Поле вибору вхідних даних для користувача

На рис. 3.24 зрозуміло, що клієнт обирає виконавця та на основі цього вибору рекомендаційні система формує список композиції, що пропонується клієнту. Клієнт ввів значення «amy macdonald» та отримав наступний результат.

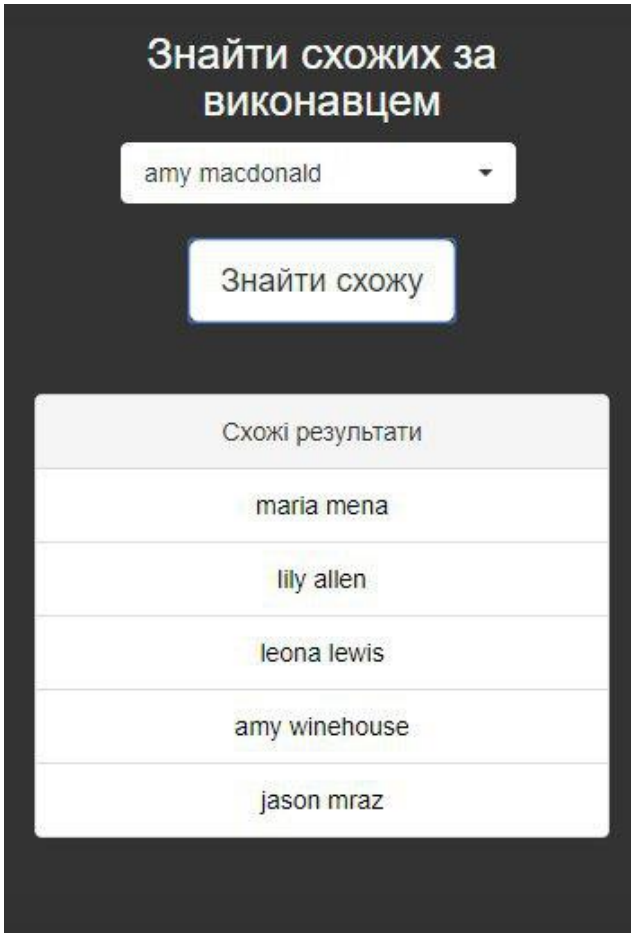

Рисунок 3.25 – Відображення результатів відбору композицій

Отже, в результаті було отримано список композицій із наступних композицій: «maria mena», «lily allen», «leona lewis», «amy winehouse», «jason mraz». Всі результати ґрунтуються на даних активності користувачей, що були зібрані у базі даних. За допомогою даного сервісу, клієнт може купити композиції, що базуються на схожих вподобаннях. Користуючись даною системою, клієнт економить час та отримує нові емоції від прослуховування композицій.

Перейдемо до наступного інструменту, що дозволяє нам отримати рекомендації на сайті (рис. 3.26).

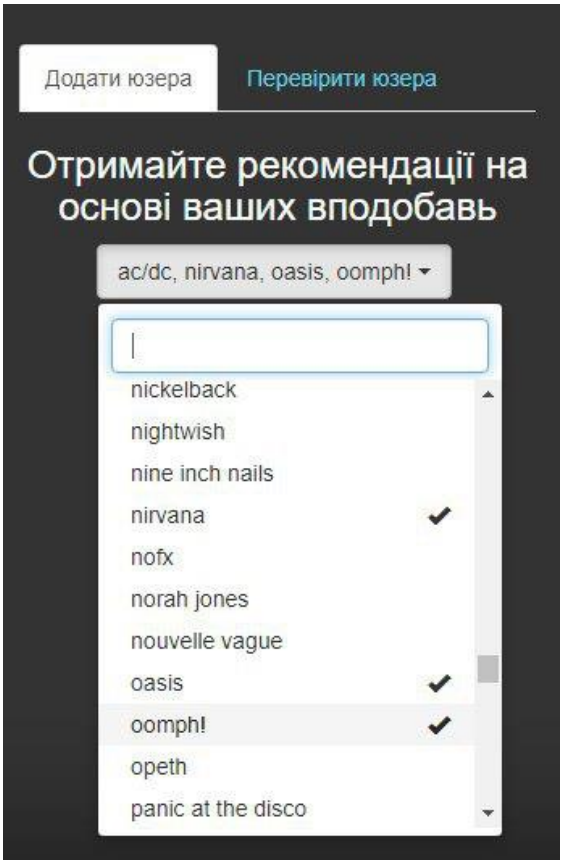

Рисунок 3.26 – Інтерфейс інструменту додавання нового користувача

Завдяки використання базі даних та збереження інформації про користувачів, є можливість запам’ятати рекомендаційний список. Клієнт може обрати декілька виконавців та отримати рекомендації, що відповідають вподобанням клієнта.

На рис. 3.26 бачимо, що клієнт відмітив масив наступних виконавців: «ac/dc», «nirvana», «oasis», «oomph!», на основі цих даних буде сформовано запит на генерацію рекомендацій. Усі дані зберігаються у сховищі, щоб була можливість отримати список у будь-який час.

На рис. 3.27 бачимо, що було додано нового юзера та отримано новий ідентифікатор, що є ключовим показником для отримання рекомендацій (у нашому випадку *userId = 1269*).

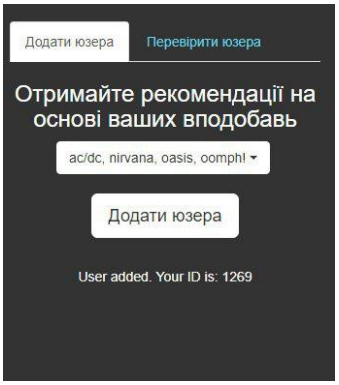

Рисунок 3.27 – Додавання користувача у базу даних

Перейдемо на наступну вкладку «Перевірити юзера» та побудуємо рекомендаційний список із використанням ідентифікатора.

На рис. 3.28 зображено, що користувач вводить свій ідентифікатор за значенням 1269, система знаходить вподобання клієнта у базі даних та їх основі формує список рекомендацій.

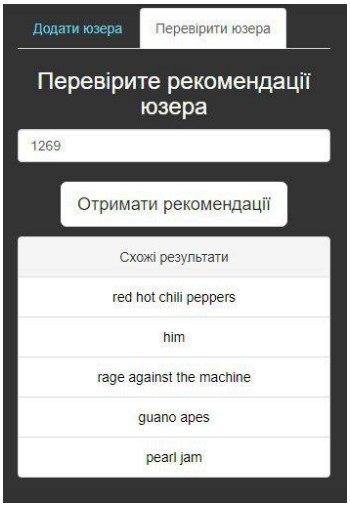

Рисунок 3.28 – Результат обробки ідентифікатора користувача та список рекомендацій

рЗауважимо, що розрахунок за допомогою декількох виконавців має певну затримку (від 5 до 60 секунд), тому що використовуються дані про всіх користувачів системи. У результаті було отримано список наступних

виконавців: «red hot chili peppers», «him», «rage against the machine», «guano apes», «pearl jam». Наразі необхідності, клієнт може звернутись та отримати новий список рекомендацій.

Треба зазначити, що система враховує відповіді клієнтам та намагається надавити новий список пісень, тобто складати списки з актуальними для користувача рекомендаціями.

Отже, проведене дослідження та розробка рекомендацій довели актуальність систем, що пропонують користувачу певний товар чи послугу. З’являються нові сфери, де застосування вище наведених алгоритмів та методів може бути використано на практиці. Серед досліджуваних методів було виявлено та побудовано найбільш продуктивний, порівняно із відомими реалізаціями.

Наведені спостереження дали підстави для побудови демонстраційного прикладу, який базується на методах рекомендаційних систем. Отримана модель дозволила відзначити суттєвий вплив часу на формування рекомендацій та необхідність поліпшення швидкості роботи алгоритмів.

Також з результатів моделювання слідує, що в залежності від алгоритма побудови формуються різні списки композицій, що відрізняються показниками релевантності та повноти. Клієнти по різному реагують на результати рекомендацій і саме тому списки слід формувати не тільки швидко, а також якісно.

Доведення ефективності рекомендацій як засобу задоволення потреб людини, дозволило визначити необхідність подальшого застосування у практиці рекомендаційних систем і зробити висновки щодо реалізації такого програмного забезпечення.

ВИСНОВКИ

У даній роботі було застосовано алгоритми рекомендаційних систем на основі реальних даних про музичні композиції. Було побудовано моделі трьох видів: модель на основі популярності пісень (наївний метод), модель на основі колаборативної фільтрації та градієнтного бустингу. Також був створений порівняльний аналіз показників моделей на основі метода точності-повноти, розраховано швидкість обробки запитів користувачів.

В першому розділі було досліджено історичні дані та хронологію розвитку рекомендаційних систем та їх алгоритмів. Наведено статистичні дані щодо діяльності користувачів в інтернеті для підтвердження актуальності дослідження теми рекомендаційних систем.

У другому розділі висвітлено основні терміни та види рекомендаційних систем. Досліджено основні засади та алгоритми побудови колаборативної фільтрації, що впроваджується в рекомендаційні системи. Обрано та застосовано інструментарій для написання програмного забезпечення, перелічено основні бібліотеки, що використовувались.

У третьому розділі застосовано необхідний інструментарій та знання на практиці для побудови моделі. Було обрано та оброблено дані про музикальні композиції з відкритих ресурсів. Реалізовано моделі на основні алгоритмів рекомендаційних систем та порівняно результати їх оцінки.

Формування рекомендаційних списків є однією з основних функцій сайтів, що пропонують клієнтам музику, фільми, товари та інше. Задля її ефективної реалізації використовуються методи та алгоритми побудови рекомендацій, статистичні дані, математичні моделі та інструменти машинного навчання, останні з набувають популярності завдяки ефективності рекомендацій та швидкості обробки. Рекомендації, сформовані за допомогою методів машинного

навчання, розраховуються з урахуванням персональних вподобань, актуальності та точності.

Система рекомендаційних списків повинна бути результативною та ефективною. Це виражається у тому наскільки її практична реалізація відповідає ключовій меті: надання списку композицій для задоволення потреб користувача.

Такій системі характерні: доступність списку рекомендацій у будь-який час, підвищення якості рекомендацій та актуальності пропонованих пісень, наявність інтуїтивного інтерфейсу та його зручність користування, здатність реагувати на дії клієнта та зберігати у пам’яті обрані композиції та вподобання, можливість формування нового списку рекомендацій при змінах вподобань та обраних музикальних композицій.

Ключовими умовами ефективного функціонування цієї системи є якісність та прозорість для користувача, надання усіх належних функцій рекомендаційної системи.

Для формування персоніфікованих рекомендацій є належним використовувати методи машинного навчання. Це пояснюється показниками точності, повноти та швидкості роботи системи. Основними технічними параметрами машинного навчання у рекомендаційних системах є: швидкість навчання (learning rate), розмір набору даних, навчальної та тестової вибірки, глибина навчання, точність рекомендацій та їх значущість.

Порівнюючи з альтернативними системами, що не використовують методи машинного навчання, тобто на основі наївного методу та класичної колаборативної фільтрації, варто визнати наступні переваги: більш ефективне використання знань про вподобання інших користувачів, які використовуються для формування рекомендації конкретного користувача, поліпшення прогнозованості та гнучкості системи, підвищення швидкості обробки запитів, визначення стратегічного орієнтиру на довгострокову перспективу.

Основними недоліками цієї системи є складнощі вибору показників, на основі яких будуються рекомендації, вибір розміру набору даних, адже не завжди є доцільно використовувати весь набір історичних даних, тому що загальні вподобання також змінюються та мають певні тенденції. Основними бар'єрами впровадження рекомендаційних систем на основі методів машинного навчання є необхідність високого рівня кваліфікації фахівців та обізнаність у предметній області об'єкту рекомендаційної системи.

Практичний досвід впровадження та застосування рекомендаційних систем свідчить, що основним стимулом використання інтелектуальних рекомендацій є негативний досвід клієнтів при користуванні сервісами, необхідність залучати нових користувачів та утримувати існуючих. Також більшість сервісів, які впроваджували інтелектуальні системи рекомендацій змогли досягти великих темпів приросту клієнтів.

З огляду на цей досвід, можна виділити такі закономірні наслідки застосування рекомендаційних систем: зменшення негативного досвіду, позитивне зростання кількості користувачів, можливість для сервісів впевнено реагувати на зміну тенденцій у сфері музики.

Згідно з результатами проведеного дослідження, варто зазначити, що використання методів машинного навчання, а саме методів градієнтного бустингу підвищили продуктивність системи, водночас не втративши актуальність та якість рекомендацій.

Отже, побудована модель довела, що машинне навчання стає невід'ємною частиною сфери послуг, зокрема рекомендаційних систем. В умовах великих масивів даних неструктурованої або слабоструктурованої інформації ці методи дозволяють отримати та використовувати неявні залежності та є наступним кроком в аналітиці. Успішне впровадження рекомендаційної системи дозволяє підвищити кількість клієнтів та збільшити задоволення від використання сервісів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Prem Melvilleand, Vikas Sindhwani Recommender Systems, Encyclopedia of Machine Learning, Claude Sammutand Geoffrey Webb (Eds), Springer, 2010.
2. Пройдл А. История систем рекомендаций [Електронний ресурс] / Адольф Пройдл. – 2015. – Режим доступу: http://old.telesputnik.ru/archive/pdf/233/48.pdf.
3. Джонс Т. Рекомендательные системы: Часть 1. Введение в подходы и алгоритмы [Електронний ресурс] – Режим доступу: https://www.ibm.com/developerworks/ru/library/os-recommender1/index.html
4. InternetWorldStats [Електронний ресурс] – Режим доступу: https://www.internetworldstats.com/stats.htm.
5. Михайлівський Н. Анатомія рекомендаційних сервісів. [Електронний ресурс] – Режим доступу: http://centrobit.ru/blog/
6. Byeong-jun Han, Seungmin Rho, Sanghoon Jun and Eenjun Hwang, 2010, “ Music emotion classification and context-based music recommendation”, Elsevier J. Multimedia Tools Applications, 47(3), pp. 433-460.
7. Amos Azaria, Avinatan Hassidim, Sarit Kraus, Adi Eshkol and Ofer Weintraub, 2013, “Movie recommender system for profit maximization,” In Proceedings of ACM conf. on Recommender Systems (RecSys '13), pp. 121-128.
8. Гомзин А. Системы рекомендаций: обзор современных подходов / А. Гомзин, А. Коршунов. – Москва: Труды Института системного программирования РАН, 2012. – 20 с.
9. Джонс Т. Рекомендательные системы: Часть 1. Введение в подходы и алгоритмы [Електронний ресурс] – Режим доступу: https://www.ibm.com/developerworks/ru/library/os-recommender1/index.html

10. S. Choi, S. Cha, C.C. Tappert; A Survey of Binary Similarity and Distance Measures; Journal of Systemics, Cybernetics and Informatics, Vol 8 No 1 2010, pa[illegible] 43-48

11. W.P. Jones, G.W. Furnas; Pictures of Relevance: A Geometric Analysis of Similarity Measures; Journal of the American society for information science. 38(6): 420-442, 1987; pages 420-442.

12. Classifier performance evaluation [Електронний ресурс] – Режим доступу: http://cmp.felk.cvut.cz/~hlavac/TeachPresEn/31PattRecog/13ClassifierPerformance.pdf

13. Evaluating Recommender Systems [Електронний ресурс] – Режим доступу: http://aimotion.blogspot.com/2011/05/evaluating-recommender-systems.html

14. Scott Hartshorn, Machine Learning With Boosting: A Beginner's Guide, Springer, 2017. – 227 с.

15. J.H. Friedman, Greedy Function Approximation: a Gradient Boosting Machine. Techinal report, Dept. of Statistics, Stanford University

16. ДоусонУлу М. Программируем на Python / М. Доусон. – СПб: Питер, 2014. – 416 с.

17. Guido van Rossum, Python Reference Manual, release 2.4.4, 18 October 2006.

18. NumPy [Електронний ресурс] – Режим доступу: http://www.numpy.org/

19. Python Data Analysis Library [Електронний ресурс] – Режим доступу: https://pandas.pydata.org/

20. Machine Learning in Python [Електронний ресурс] – Режим доступу: http://scikit-learn.org/stable/

21. Turi Machine Learning Platform User Guide [Електронний ресурс] – Режим доступу: https://turi.com/learn/userguide/index.html

22. Amos Azaria, Avinatan Hassidim, Sarit Kraus, Adi Eshkol and Ofer Weintraub, 2013, “Movie recommender system for profit maximization,” In Proceedings of ACM conf. on Recommender Systems (RecSys '13), pp. 121-128.

23. Ben Schafer J., Joseph A. Konstan and John Riedl, 2001, “ E-Commerce recommendation applications,” Springer J. Data Mining and Knowledge Discovery. 5(1-2), pp. 115-153.

24. Jia-Dong Zhang, Chi-Yin Chow and Yanhua Li, 2015, “iGeoRec: A personalized and efficient geographic allocation recommendation framework,” IEEE Trans. Services Computing, 8(5), pp. 701-714.

25. Xiao Yu, Ang Pan, Lu-An Tang, Zhenhui Li and Jiawei Han, 2011, “Geo-Friends Recommendation in GPS-Based Cyber-Physical Social Network.,” In Proceedings of IEEE Intl. Conference on Advances in Social Networks Analysis and Mining, pp. 361- 368.

26. Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon and John Riedl, 1997, “GroupLens: applying collaborative filtering to Usenet news,” Communications of the ACM, 40(3), pp. 77-87.

27. Lei Li, Dingding Wang, Tao Li, Daniel Knox and Balaji Padmanabhan, 2011, “SCENE: A scalable twostage personalized news recommendation system,” In Proceedings of ACM SIGIR’11 Conference, pp. 125-134.

28. Shu-Lin Wanga and Chun-Yi Wub, 2011, “Application of context-aware and personalized recommendation to implement an adaptive ubiquitous learning system.,” Elsevier J. on Expert Systems with Applications, 38(9), pp. 10831-10838.

29. Kaggle: Your Home For Data Science [Електронний ресурс] – Режим доступу: https://www.kaggle.com/

30. PostgreSQL: The World's Most Advanced Open Source Relational Database [Електронний ресурс] – Режим доступу: https://www.postgresql.org/

Додаток

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка алгоритмів машинного навчання для рекомендаційної системи вибору музичних композицій

Студент: Демидов Данило Дмитрович, ПДМ-61
Науковий керівник: к.т.н., доц., Щербина Ірина Сергіївна

Київ-2020

ОСНОВНІ ХАРАКТЕРИСТИКИ РОБОТИ

- **Метою** даної роботи є дослідження та розробка рекомендаційної системи із застосування ефективних алгоритмів, що використовує історичні дані та активність користувачів для пропонування наступних музикальних композицій, заснованих на вподобаннях користувача та релевантності обраного об’єкту.
- **Об’єктом дослідження** роботи є рекомендаційні системи та алгоритми їх побудови.
- **Предметом дослідження** даної роботи є алгоритми та структури даних рекомендаційних систем, їх ефективність в умовах неповної або відсутньої інформації.

АКТУАЛЬНІСТЬ ЗАДАЧІ

З розвитком інформаційних технологій та їх впровадженням у суспільне життя, виникає потреба пошуку акцентованої інформації в умовах невизначеності. Для вирішення таких задач в останній час створюються інтелектуальні ***рекомендаційні системи.***

Рекомендаційні системи - це спеціальні програми, головна мета яких полягає у формуванні рекомендацій різних продуктів або сервісів для користувачів на основі їх переваг.

У теперішній час існують наступні типи рекомендаційних систем:
- засновані на користувачах (User-based) – для рекомендації використовується історія оцінок як самого користувача, так і інших користувачів;
- засновані на предметах (Item-based) – рекомендації на основі використаних предметів;

Рис. 1. Приклад рекомендацій фільмів

Рис. 2. Приклад музичних рекомендацій

5

User-based системи

1). Засновані на користувачах (User-based) – для рекомендації використовується історія оцінок як самого користувача, так і інших користувачів.

Музикальні виконавці	User_1	User_2	User_3	User_4
Nirvana	15	2	8	-
Michael Jackson	8	-	-	1
The Beatles	4	3	7	-
Queen	-	10	-	10
Frank Sinatra	-	11	2	7
	Кількість пісень, що були прослухані користувачами			
Кластер	1	2	1	2

7

Item-based системи

2). Засновані на предметах (Item-based) – рекомендації на основі використаних предметів або об'єктів системи (фільми, треки, товари та інше)

Музикальні виконавці	User_3		Подібний контент
Nirvana	8		Nirvana
Michael Jackson	-		Michael Jackson
The Beatles	7		The Beatles
Queen	-		Проранжовані
Frank Sinatra	2		виконавці

9

НЕДОЛІКИ ІСНУЮЧИХ СИСТЕМ

Незважаючи на те, що *user-based* та *item-based* системи є найбільш поширеними, в процесі їх функціонуванні виникає ряд проблем:

- наявність даних про користувачів, що не піддаються моделюванню;
- Низька швидкість (більше 60-120 секунд на обробку запита за одним користувачем) формування рекомендацій для користувачів на великих масивах даних;
- формування рекомендацій для нових користувачів, за умовою недостатньої інформації про їх вподобання.

Саме тому існує необхідність оптимізації рекомендацій і пошук нових підходів до формування рекомендаційних об'єктів, поліпшення якості рекомендацій, обгрунтування рекомендацій, також існує проблема аналізу додаткової інформації (як розпізнати не тільки явну (explicit), але і неявну рекомендацію користувача, як враховувати зв'язку між користувачами і об'єктами, аналіз ознак між ними).

11

МЕТОДИ МАШИННОГО НАВЧАННЯ

Для усунення зазначених проблем в останній час ефективно використовуються методи машинного навчання [3]:

- метод k-найближчих сусідів;
- алгоритм Байеса;
- метод сингулярного розкладання матриці (SVD).

Серед цих методів метод SVD знаходить найбільш широке застосування на практиці.

Сингулярний розклад матриці будується наступним чином:

$$R \approx USV^t, \quad (1)$$

де U – ліва сингулярна матриця, що представляє взаємозв'язок між користувачами і прихованими факторами. S – діагональна матриця, що описує вплив кожного прихованого фактора, а V^t – права сингулярна матриця, яка вказує подібність між елементами і прихованими факторами, R – матриця вихідних оцінок.

13

ВИКОРИСТАННЯ МАШИННОГО НАВЧАННЯ

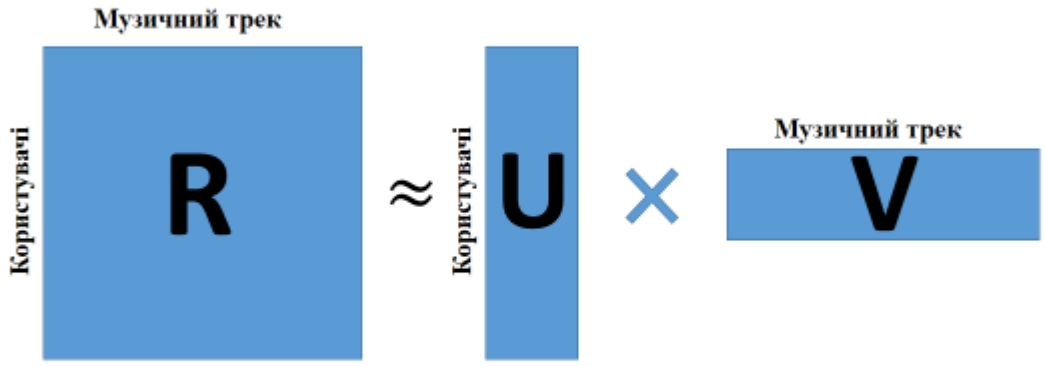

Рис. 4. Візуалізація сингулярного розкладу матриці на основі відношення користувач-музичний трексс

15

МЕТОД ГРАДІЄНТНОГО БУСТІНГУ

Основною проблемою методу SVD є повільна швидкість обробки даних в онлайн-системах. З огляду на це пропонується удосконалити застосування методу SVD шляхом поєднання його з **методом градієнтного бустінгу** [4].

Це дозволить скоротити час та збільшити швидкість обробки даних.

Згідно алгоритму градієнтного бустингу наближене значення функції, що описує вибір користувача в музичній рекомендаційній системі, знаходиться як:

$$y \approx \hat{f}(x),$$
$$\hat{f}(x) = argmin\, L(y, f(x)), \qquad (2.8)$$

де $\hat{f}(x)$ – наближена функція, $L(y, f(x))$ – функція втрат.

Слід зауважити, оскільки існує нескінченна кількість функцій $f(x)$, необхідно обмежити множину функцій $f(x, \theta), \theta \in R^n$, де R^n – простір дійсних векторів.

Тоді кінцева функція вподобань користувача за допомогою градієнтного бустінгу розраховується наступним чином:

$$F(x) = \sum_{i=0}^{M} \hat{f}_i(x), \qquad (2.9)$$

де M – кількість ітерацій підбору наближеної функції вподобань користувача.

17

РЕАЛІЗАЦІЯ АЛГОРИТМІВ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

Наївна модель

```
pm = Recommenders.popularity_recommender_py()
pm.create(train_data, 'user_id', 'song')
user_id = users[5]
pm.recommend(user_id)
```

Отримаємо список пісень для конкрентного користовача ***(user_id = 5)***, який є рекомендаційним списком елементів.

У цьому методі використовуються загальні дані про пісні та їх популярність.

Персональні характеристики не враховуються.

user_id	song	score	Rank	
4032	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Sehr kosmisch - Harmonia	86	1.0
1338	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Dog Days Are Over (Radio Edit) - Florence + Th...	56	2.0
4445	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Revelry - Kings Of Leon	56	3.0
6520	4bd88bfb25263a75bbdd467e74018f4ae570e5df	You're The One - Dwight Yoakam	51	4.0
4684	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Secrets - OneRepublic	47	5.0
5988	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Undo - Björk	47	6.0
1753	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Fireflies - Charttraxx Karaoke	44	7.0
2355	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Horn Concerto No. 4 in E flat K495: II. Romanc...	40	8.0
5035	4bd88bfb25263a75bbdd467e74018f4ae570e5df	The Scientist - Coldplay	40	9.0
5824	4bd88bfb25263a75bbdd467e74018f4ae570e5df	Tive Sim - Cartola	37	10.0

19

РЕАЛІЗАЦІЯ АЛГОРИТМІВ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

Модель коллаборативної фільтрації

```
#пісні для користувача, взяті з датасету
user_id = users[5]
user_items = is_model.get_user_items(user_id)
```

```
#рекомендації щодо пісень на основі персональної моделі
is_model = Recommenders.item_similarity_recommender_py()
is_model.create(train_data, 'user_id', 'song')
is_model.recommend(user_id)
```

На цьому етапі обираємо конкрентного користувача (користувач з ідентифікатором 5) та отримаємо список пісень, які користувач вже прослухав:

Push It - Salt-N-Pepa
Speechless - Lady GaGa
Just Lose It - Eminem
Without Me - Eminem
Ghosts 'n' Stuff (Original Instrumental Mix) - Deadmau5
16 Candles - The Crests
Missing You - John Waite
Say My Name - Destiny's Child
My Dad's Gone Crazy - Eminem / Hailie Jade
The Real Slim Shady - Eminem
Somebody To Love - Justin Bieber
Forgive Me - Leona Lewis
Ya Nada Queda - Kudai

Сгенерований список рекомендацій пісень:

user_id	song	score	rank
4be888dfb252631a75bbdd467e74003f4ae570e5df	Mockingbird - Eminem	0.062260	1
4be888dfb252631a75bbdd467e74003f4ae570e5df	Heartbreak Warfare - John Mayer	0.050855	2
4be888dfb252631a75bbdd467e74003f4ae570e5df	Superman - Eminem / Dina Rae	0.050789	3
4be888dfb252631a75bbdd467e74003f4ae570e5df	One Time - Justin Bieber	0.050047	4
4be888dfb252631a75bbdd467e74003f4ae570e5df	U Smile - Justin Bieber	0.046741	5
4be888dfb252631a75bbdd467e74003f4ae570e5df	Te Amo - Rihanna	0.044930	6
4be888dfb252631a75bbdd467e74003f4ae570e5df	Eenie Meenie - Sean Kingston and Justin Bieber	0.044160	7
4be888dfb252631a75bbdd467e74003f4ae570e5df	PapaFlames - M.I.A.	0.043189	8
4be888dfb252631a75bbdd467e74003f4ae570e5df	Terra Promise - O'Rasko Rarisin	0.042895	9
4be888dfb252631a75bbdd467e74003f4ae570e5df	Brave The Elements - Colossal	0.042287	10

РЕАЛІЗАЦІЯ АЛГОРИТМІВ РЕКОМЕНДАЦІЙНИХ СИСТЕМ

Порівняння якостей отриманих результатів

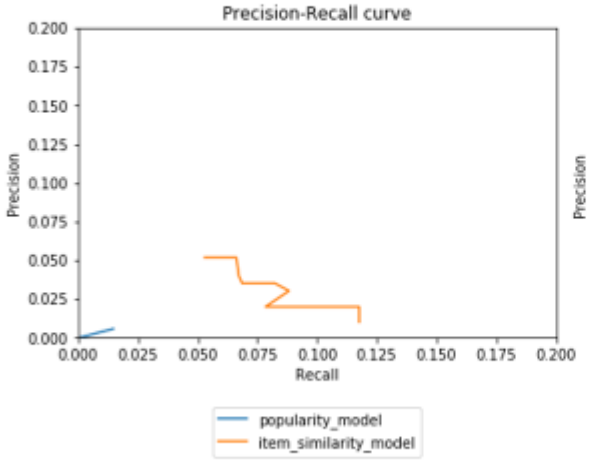

Рис.5. Результат оцінки точності-повноти

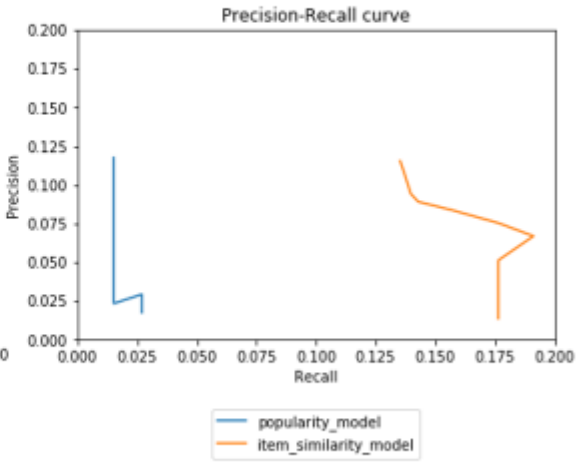

Рис. 6. Результат оцінки точності-повноти на більшому наборі даних

23

Порівняння швидкості формування рекомендацій для одного користувача

Результати розрахунків швидкості алгоритми			
	Средняя время, сек	Средняя точность, %	Средняя полнота, %
Naive method	7.12	49.07	50.52
Collaboration filtering	5.68	45.55	53.41
Gradient Boosting	4.23	49.12	63.33

Швидкість обробки при наборі даних зі 1000 користувачів

Результати розрахунків швидкості алгоритми			
	Среднее время, сек	Средняя точность, %	Средняя полнота, %
Naive method	23.62	19.42	20.56
Collaboration filtering	18.37	58.27	61.22
Gradient Boosting	10.78	55.61	62.10

Швидкість обробки при наборі даних зі 10000 користувачів

25

АРХІТЕКТУРА РЕКОМЕНДАЦІЙНИЙ СИСТЕМІ

В архітектурі системи використано наступні компоненти:

- Front-end система;
- Web Application (Back-end) система;
- Analysis Service (Модуль навчання моделі);
- База даних.

Опишемо алгоритм роботи системи:

1. Клієнт відправляє запит на отримання списку рекомендацій (*REST request*).
2. *Front-end* інтерфейс отримує та обробляє запит, надсилаючи його на *Back-end* систему.
3. Далі запит поступає до модулю *Analysis Service*, який розраховує та формує список музичних композицій для користувача.
4. Back-end система отримує список та готує відповідь для веб-інтерфейсу.
5. Клієнту відображається список рекомендацій.

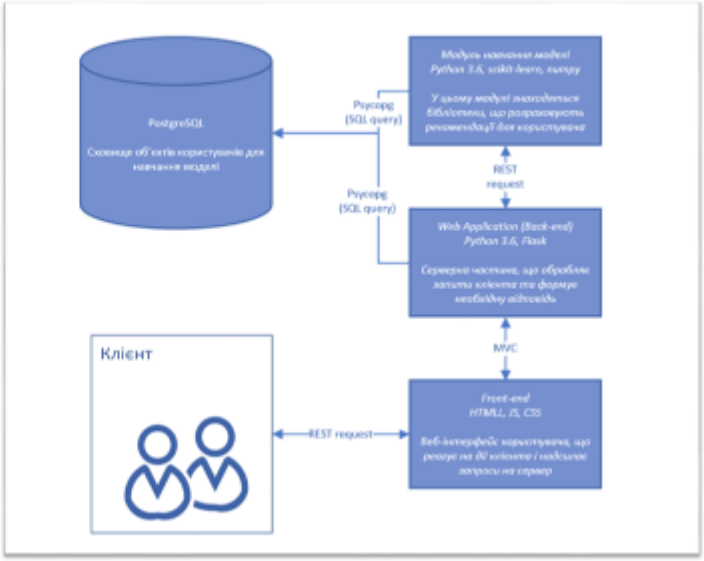

Рис. 7. Схема архитектури рекомендаційної системи

27

Висновки

- У даній роботі було застосувано алгоритми рекомендаційних систем на основі реальних даних про музичні композиції.
- Було побудовано моделі видів: модель на основі популярності пісень (наївний метод), модель на основі колаборативної фільтрації та градієнтного бустингу. Також був створений порівняльний аналіз показників моделей на основі метода точності-повноти.
- В подальшому дослідженні можливі покращення реалізації рекомендаційної системи, наприклад, застосування гібридних алгоритмів, дослідити можливості дослідження схожих об'єктів, враховувати ознаки та зв'язки між об'єктами та контекст. Також враховувати інформацію з інших джерел та предметних областей для покращення моделі — поліпшення адаптивності до нових змін в середовище та якості створених рекомендацій.

29

Апробація

Демидов Д.Д. «Розробка алгоритмів машинного навчання для рекомендаційної системи вибору музичних композицій». // Загально-університетська, науково-технічна конференція «Проблеми комп’ютерної інженерії»;

Та стаття:

Демидов Д.Д. «Розробка алгоритмів машинного навчання для рекомендаційної системи вибору музичних композицій», Журнал «Зв’язок», №6, 2020, Київ.

31