МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ:

Програмний модуль підбору музичних композицій

Галузь знань **12 «Інформаційні технології»**
Спеціальність **122 «Комп'ютерні науки»**
Освітня програма **«Аналітика даних»**
Освітній рівень: бакалавр

Виконав: студент 4 курсу, групи АнД- 41

_____Майданик А.О.__ _____
(прізвище та ініціали)

Керівник __Мінаєва Ю.І._____ _____
(прізвище та ініціали)
__к.т.н., доцент_____
(науковий ступінь, звання)

Випускна кваліфікаційна робота бакалавра допущена до захисту рішенням кафедри *інтелектуальних технологій*
Протокол № _13_ від __05.06.2023 р.
Завідувач кафедри _____ доц. Іларіонов О.Є.

Київ – 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій
Кафедра інтелектуальних технологій
Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
інтелектуальних технологій
Іларіонов О.Є.
_____
"___"_____2023 р.

ЗАВДАННЯ
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Майданик Андрій Олександровичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи)
   Програмний модуль підбору музичних композицій
   затверджена протоколом засідання кафедри від «_11__» листопада 2022 р. № _4__
2. Термін здачі студентом закінченого проекту (роботи) __29___ травня 2023 року
3. Вихідні дані до проекту (роботи) Список рекомендацій пісень
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
   Дослідження та аналіз предметної області, що пов'язана із методами розробки модуля підбору музичних композицій, програмна реалізація застосунку, тестування розробленого застосунку
5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій)
   Мета дослідження дипломного проекту (1 слайд), опис предметної області (1 слайди), постановка задачі (1 слайд), опис навчального набору даних (1 слайд), алгоритм вирішення задачі (1 слайд), схема архітектури програмного модулю та опис інструментів програмної реалізації (1 слайд), оцінка результатів та приклади роботи застосунку (2 слайди), висновки (1 слайд).

6. Консультанти з випускної кваліфікаційної роботи із зазначенням розділів випускної кваліфікаційної роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 15 лютого 2023 року

Керівник _____ / Мінаєва Ю.І. /
(підпис) (ПІБ)

Завдання прийняв до виконання _____ / Майданик А.О. /
(підпис) (ПІБ)

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Примітка
1.	Обговорення з керівником постановки завдання, підбір літератури, огляд існуючих рішень	15.02.2023 – 01.03.2023	
2.	Аналіз постановки задачі, формалізація задачі, аналіз літературних джерел, написання розділу 1	02.03.2023 – 20.03.2023	
3.	Розробка програмного модуля підбору музичних композицій, написання розділу 2	21.03.2023 – 11.04.2023	
4.	Програмна реалізація програмного модуля підбору музичних композицій т, написання розділу 3	12.04.2023 – 15.05.2023	
5.	Робота над оформленням пояснювальної записки та презентаційних матеріалів	16.05.2023 – 29.05.2023	

Студент-дипломник _____ / Майданик А.О. /
(підпис) (ПІБ)

Керівник випускної кваліфікаційної роботи _____ / Мінаєва Ю.І. /
(підпис) (ПІБ)

Анотація

Майданик Андрій Олександрович виконав випускну кваліфікаційну роботу на тему «Програмний модуль підбору музичних композицій» за спеціальністю 122 – «Комп’ютерні науки», освітня програма «Аналітика даних»

У випускній кваліфікаційній роботі проведено аналіз сучасних підходів до формування рекомендацій, було розглянуто метрики оцінювання якості наданих рекомендацій, розроблено програмне забезпечення, що надає персональні рекомендації музичних композицій, на основі подібності пісень між собою та інтерфейс, який транслює отримані рекомендації користувачеві.

Ключові слова: програмний модуль, колаборативна фільтарція, ,рекомендаційні системи, метод K-Means, метод KNN

Summary

Andrii Oleksandrovych Maydanyk performed the final qualification work on the topic "Software module for selecting musical compositions" in the specialty 122 - "Computer Science", educational program "Data Analytics".

The final qualification work analyzes modern approaches to the formation of recommendations, considers metrics for assessing the quality of the recommendations provided, develops software that provides personalized recommendations of musical compositions based on the similarity of songs among themselves and an interface that transmits the received recommendations to the user.

Keywords: software module, collaborative filtering, recommender systems, K-Means method, KNN method

Зміст

Вступ

Рекомендаційна система — підклас системи фільтрації інформації, яка будує рейтинговий перелік об'єктів (фільми, музика, книги, новини, вебсайти), яким користувач може надати перевагу. Для цього використовується інформація з профілю користувача.

Мета алгоритму рекомендацій полягає в тому, щоб рекомендувати або прогнозувати елементи, які можуть сподобатися користувачеві, на основі його даних або на основі всієї бази даних користувачів.

Рекомендаційні системи широко використовуються в різних галузях, і їх застосування має велике значення для підтримки та залучення клієнтів, а також створення клієнтоорієнтованих продуктів. В епоху інформаційного перевантаження та швидкого розвитку технологій, рекомендаційні системи допомагають спростити процес пошуку та вибору, надаючи користувачам персоналізовані рекомендації на основі їхніх інтересів, поведінки та інших факторів. У сфері електронної комерції, рекомендаційні системи допомагають покупцям знайти товари, які найбільш відповідають їхнім потребам та вподобанням. Це сприяє збільшенню конверсії та задоволеності клієнтів, а також збільшенню продажів. В сфері медіа та розваг, рекомендаційні системи допомагають користувачам знаходити цікаві фільми, музику, книги або новини, що робить їх досвід більш насиченим та персоналізованим. Також рекомендаційні системи застосовуються в сферах соціальних мереж, подорожей, готелів, ресторанів, музеїв та багатьох інших. Вони допомагають користувачам знайти нові контакти, отримати персоналізовані пропозиції та рекомендації щодо подорожей та відпочинку, покращуючи загальний досвід та задоволення. Загалом, рекомендаційні системи стали невід'ємною частиною сучасного цифрового світу, що допомагає компаніям покращувати взаємодію з клієнтами, з абезпечувати персоналізований підхід та сприяти зростанню бізнесу. Використання рекомендаційних систем стає все більш важливим для успіху підприємств у сучасному конкурентному середовищі

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ ПРОГРАМНОГО МОДУЛЯ ПІДБОРУ МУЗИЧНИХ КОМПОЗИЦІЙ

1.1 Огляд рекомендаційних систем

Рекомендаційні системи - це набір технік інформаційного фільтрування, які пропонують користувачам потенційно корисні для них товари, фільми, музику і т.д.

Існує декілька широких категорій традиційних підходів до систем рекомендацій, а саме: колаборативна фільтрація, фільтрація вмісту та гібридні системи рекомендацій. Однак недавно були запропоновані нові варіанти, які враховують додаткові аспекти, такі як положення користувача (географічна рекомендація), послідовність взаємодій користувача (послідовна рекомендація) і взаємодії в поточному сеансі користувача для передбачення наступного кліку (рекомендація на основі сеансу).

Види РС:

1.Колоборативна фільтрація

2.На основі контенту

3. Системи засновані на послідовності

4. Гібридні

Контентна фільтрація - це метод фільтрування вмісту, який ґрунтується на математичній моделі об'єкту, для якого прогнозуються оцінки. Кожен об'єкт має власну модель, яка враховує специфічні характеристики товару (параметри моделі). Рекомендації засновані на порівнянні характеристик поточного товару з характеристиками користувача, що містяться в його профілі. Ці системи використовують експертні знання або бази даних для формування рекомендацій. Вони враховують правила, обмеження та доменні знання, щоб створити персоналізовані рекомендації.Контентна фільтрація - це метод фільтрування вмісту, що ґрунтується на моделі об'єкту, для якого будуть прогнозуватися оцінки. Кожен об'єкт має свою математичну модель, яка

враховує конкретні характеристики товару (параметри моделі). Рекомендації ґрунтуються на порівнянні характеристик поточного товару з характеристиками користувача (інформація, яка міститься в профілі користувача). Ці системи використовують експертні знання або бази знань для створення рекомендацій. Вони враховують правила, обмеження та доменні знання, щоб зробити персоналізовані рекомендації.

Рекомендаційні системи на основі послідовності можуть аналізувати послідовність дій або подій, таких як перегляди веб-сторінок, покупки, переключення телевізійних каналів, прослуховування музики тощо. За допомогою методів аналізу послідовності, система може виявляти зв'язки, патерни та тенденції у поведінці користувача.

Гібридні системи: Ці системи комбінують різні підходи, такі як контентний, колаборативний та знаннями заснований, для отримання більш точних та різноманітних рекомендацій. Вони використовують переваги кожного підходу для створення кращих рекомендацій.

На рисунку 1 схематично зображено,які типи рекомендаційних систем існують.

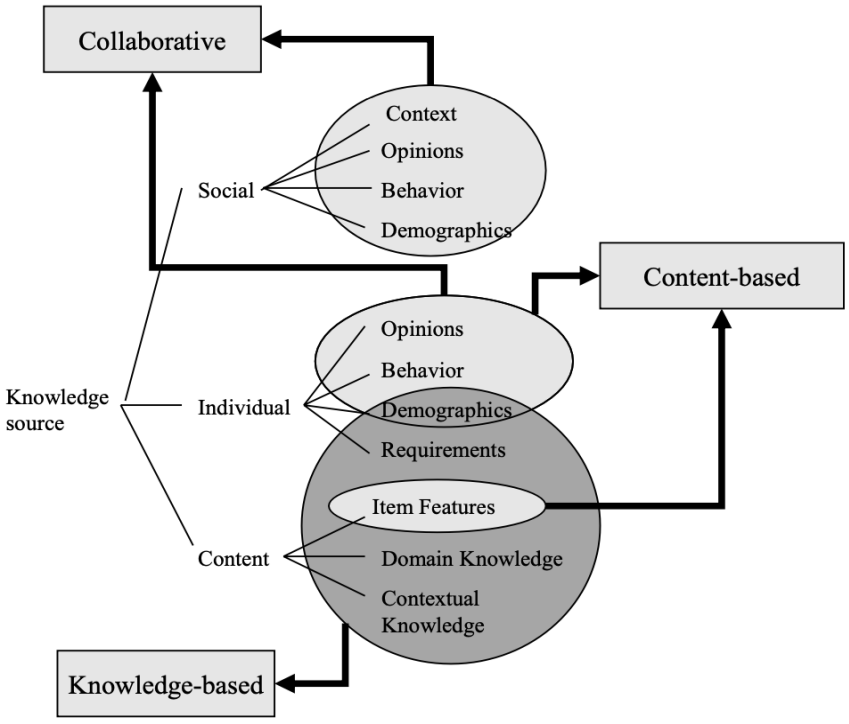

Рисунок 1.1 – Джерела знать та типи рекомендацій

1.2.1 Системи колаборативної фільтрації

Колаборативна фільтрація - це метод прогнозування інтересів користувачів шляхом визначення ступеня подібності між користувачами та їх вподобаннями. Основна концепція колаборативної фільтрації полягає у тому, що якщо користувачі A і B мають схожі смаки щодо одного продукту, ймовірно, що їх вподобання також будуть подібні щодо інших продуктів. [1].

Існують два поширені типи підходів у спільній фільтрації: на основі пам’яті та на основі моделі.

Основна ідея підходів, відомих також як методи колективної фільтрації на основі сусідства або пам'яті, полягає в тому, що прогнозування рейтингів здійснюється на основі найближчих користувачів або елементів. Цей підхід можна розділити на два типи: фільтрацію на основі користувачів і фільтрацію на основі елементів. Фільтрація на основі користувачів означає, що рекомендації формуються залежно від інтересів схожих користувачів, тоді як фільтрація на основі елементів рекомендує елементи на основі їх взаємодії.

Підходи на основі моделі використовують прогнозні моделі машинного навчання. Ці моделі отримують на вхід матрицю оцінок, яка може бути використана в її оригінальному або зміненому вигляді. Далі методи оптимізації, такі як дерева рішень, матрична факторизація та її модифікації або асоціативні правила, використовуються для прогнозування оцінок, які користувачі можуть дати певному елементу. Як приклад системи, заснованої на колаборативній фільтрації, можна згадати Reddit: алгоритм вибирає рекомендовані для користувача облікові записи та статті на основі інших осіб, які мають спільні інтереси.

Один з основних переваг використання колаборативної фільтрації полягає в її простоті впровадження та здатності рекомендувати різноманітні елементи. Крім того, такі системи не потребують аналізу вмісту об'єкта, незалежно від того, чи це музична композиція, готель, книга або товар у магазині.

Одним з основних недоліків цієї моделі є її незручність для рекомендацій нових елементів або користувачів, які тільки починають використовувати систему, оскільки з ними ще не відбулося жодної взаємодії. Цю проблему називають "холодним стартом". Крім того, відомо, що алгоритми на основі пам'яті показують гірші результати на наборах даних, які мають розріджений характер..

1.2.2 Системи засновані на послідовності

Рекомендація, заснована на знаннях, є більш універсальною категорією, в якій особа, що рекомендує, застосовує будь-які знання з предметної області, що є більш суттєвими, ніж характеристики товару. Такі системи використовують послідовність взаємодій користувача з елементами в рамках сеансу в процесі формування рекомендацій. Наприклад, це може бути передбачення наступного товару в кошику для покупок в інтернеті або наступного аудіо для прослуховування.

Один з таких алгоритмів у YouTube Music ці системи використовують послідовність дій користувача і поточний контекст, щоб передбачити ймовірність наступної дії. Вони навчили модель передбачати, які пісні користувач ймовірно захоче слухати, враховуючи різну інформацію про нього, таку як країна, пристрій, дата й час, а також попередні пісні, які він слухав.

1.2.3 Системи на основі контенту

Рекомендації на основі контенту - це індивідуально орієнтовані рекомендації, які використовують особливості товару та думки користувачів для навчання класифікатора, який може передбачити вподобання користувачів щодо нових товарів.Найпростіші форми систем на основі контенту вимагають метаданих об’єкту та явних або неявних реакцій користувача на елемент.

Метадані об'єкту представляють собою характеристики або атрибути елемента. Для музики це можуть бути такі параметри, як жанр, автор, країна виробництва та інші. Чим більше інформації про елемент мається, тим краще

система може передбачити інтереси користувача. Зворотній зв'язок від користувача може бути явним, наприклад, оцінка, лайк або дизлайк, або неявним, наприклад, перегляд сторінки виконавця. Чим більше користувач взаємодіє з платформою, тим більш точно можна зрозуміти його вподобання.

Прикладом такої системи є Amazon, який рекомендує користувачам подібні продукти до тих, які вони раніше купували. Ще одним прикладом є Reddit, який використовує алгоритмічні підходи для рекомендацій нових публікацій на платформі. Рекомендації Reddit обмежені часом публікації, кількістю лайків, дизлайків, коментарів і т.д. У формулу для розрахунку оцінки публікації враховуються ці фактори.

Основним недоліком цих систем є те, що рекомендації стають "очевидними", оскільки базуються на елементах або вмісті, які користувач вже споживав. Це може бути проблемою, оскільки якщо користувач ніколи не взаємодіяв з певним типом елементу, він ніколи не буде рекомендований таким елементом. Це зменшує різноманітність рекомендацій, що може негативно вплинути на деякі підприємства.

1.2.4 Гібридна рекомендаційна система

Ці рекомендацін системи базуються на комбінації вищезазначених методів. Гібридна система, що поєднує методи А та Б, намагається використати переваги методу А для виправлення недоліків методу Б. Наприклад, методи CF страждають від проблем з новими товарами, тобто вони не можуть рекомендувати товари, які не мають рейтингів. Це не обмежує контент-орієнтовані підходи, оскільки прогноз для нових елементів базується на їх описі (характеристиках), які, як правило, є легкодоступними. Маючи дві (або більше) базові методики РС, було запропоновано декілька способів їх комбінування для створення нової гібридної системи. Як вже було згадано, контекст користувача, коли він шукає рекомендацію, може бути використаний для кращої персоналізації результатів роботи системи. Наприклад, у часовому контексті рекомендації щодо відпочинку взимку повинні сильно відрізнятися

від тих, що надаються влітку. Або рекомендація ресторану для суботнього вечора з друзями повинен відрізнятися від того, що пропонується для обіду з колегами в робочий день.Обговорюючи можливості об'єднання декількох методів надання рекомендацій з урахуванням контексту в єдиний уніфікований підхід, автори також наводять приклад одного з таких комбінованих підходів. Обговорюються три різні алгоритмічні парадигми включення контекстної інформації в процес формування рекомендацій: редукція (попередня фільтрація), текстова пост-фільтрація та контекстне моделювання. У методах, заснованих на редукції (попередній фільтрації), для розрахунку рекомендацій використовується тільки та інформація, яка відповідає поточному контексту використання, наприклад, рейтинги елементів, що оцінюються в тому ж контексті. При контекстній пост-фільтрації алгоритм рекомендацій ігнорує формування контексту. Вихідні дані алгоритму фільтруються/коригуються для включення лише тих рекомендацій, які є релевантними в цільовому контексті. У контекстному моделюванні, більш складному з трьох підходів, дані контексту явно використовуються в моделі прогнозування.

1.3 Алгоритми рекомендаційних систем

1.3.1 USER-USER

Алгоритми з цієї категорії використовують матрицю користувач-елемент для визначення ступеня подібності інтересів користувачів та надання рекомендацій, базуючись на цій подібності. Матриця формується наступним чином: рядки матриці представляють користувачів інформаційної системи, стовпці матриці представляють елементи або послуги інформаційної системи, а значення в окремій комірці матриці відображають відгук користувача про певний елемент або послугу у визначеному форматі.

1.3.2 ITEM-ITEM

Алгоритми надання рекомендацій на основі подібності послуг (алгоритми item-item) подібні до алгоритмів користувач-користувач, за

винятком того, що вони порівнюють не користувачів, а самі послуги. Спочатку обирається певна послуга, яку користувач оцінив найкраще. Потім здійснюється пошук подібних послуг, порівнюючи їх за даними з матриці користувач-послуг. В результаті роботи алгоритм повертає впорядкований список послуг за ступенем подібності, або k найближчих послуг до тієї, яку користувач оцінив найкраще. Для покращення якості рекомендацій можна враховувати не тільки найкраще оцінені користувачем послуги, але й декілька з них.

1.3.3 Метод матричної факторазції (Matrix Factorization)

Описаний метод є одним з найпопулярніших у колаборативній фільтрації і використовується для розбиття матриці, яка відображає взаємодію між користувачами та об'єктами, на менші матриці. Цей підхід набув широкого визнання після перемоги в конкурсі рекомендаційних алгоритмів Netflix Prize. В основі цього методу лежать два припущення: 1) існують фактори, що унікально описують об'єкти, надані інформаційною системою; 2) ті самі фактори можуть використовуватися для опису інтересів користувачів. Під час роботи така система виявляє складні або незрозумілі зв'язки, які виражені математично, але можуть бути неочевидними з інтуїтивної точки зору. Проте також можна використовувати критерії, які зрозумілі для користувачів. На виході результату факторизації представлено у вигляді розкладу розрідженої матриці користувач-об'єкт на менші матриці: користувач-фактор та об'єкт-фактор. Ці матриці відображають представлення користувачів та об'єктів відповідно, і користувачі з схожими категоріями та схожими об'єктами розташовані ближче один до одного.

1.4 Аналіз бізнес процесів предметної області

Рекомендаційні системи відіграють важливу роль для таких відомих компаній, як Amazon.com, YouTube, Youtube Music, Netflix, Spotify, LinkedIn, Facebook, IMDb, Google, Apple Music та Apple TV. Крім того, багато медіа-компаній в даний час розробляють свої власні рекомендаційні системи як частину своїх сервісів для користувачів. Наприклад, компанія Netflix,

провайдер онлайн-відео, нагородила команду з призом у мільйон доларів за значні покращення їх рекомендаційної моделі. Згідно з дослідженнями компанії McKinsey, понад 75% контенту, який споживачі дивилися на платформі Netflix, був рекомендований їхньою системою рекомендацій. Ці системи допомагають користувачам знайти цікавий контент, який вони хочуть переглянути, а також допомагають компаніям зекономити на маркетингових витратах.

Торговий гігант Amazon повідомив, що 35% свого доходу він отримує завдяки своїм рекомендаційним системам. Вони показують споживачам товари, які часто придбуються разом з товарами, які вже знаходяться в їхньому кошику. Також вони рекомендують товари, які подібні до тих, які користувачі щойно переглянули, або нові версії товарів, які вони вже придбали. Ці рекомендаційні системи використовуються не тільки на веб-сайті, але й у електронних листових розсилках, що призводить до значного зростання продажів.

Оцінки вподобань користувачів до товарів можуть бути представлені числовими значеннями, такими як рейтинги на шкалі від 1 до 5 зірочок, або бінарними значеннями, такими як "сподобалось" або "не сподобалось". Крім того, відгуки користувачів можуть бути явними (explicit), коли вони свідомо вводять їх у систему, або неявними (implicit), отриманими з історії покупок, переглядів тощо. На рисунку 1.2 зображено Матрицю Манделоу

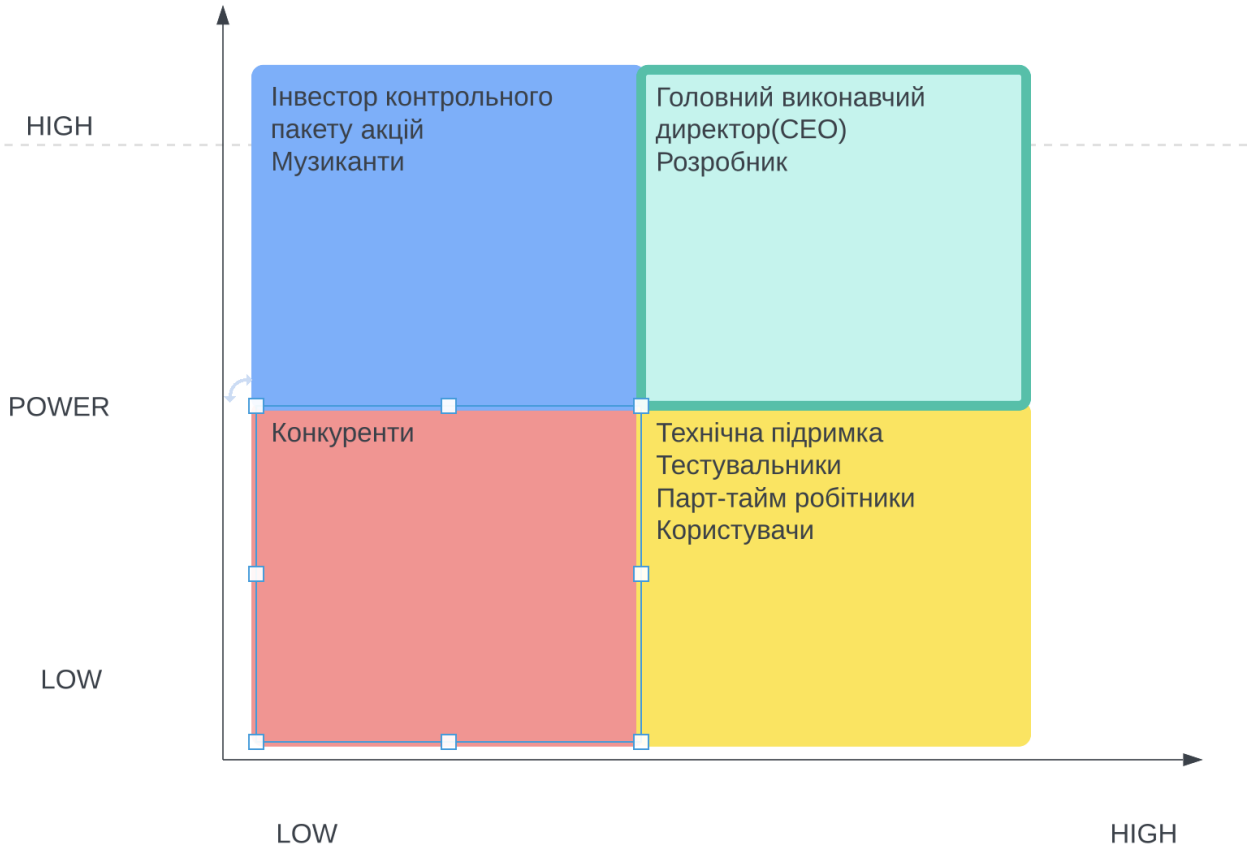

Рисунок 1.2 – Матриця Манделоу

На діаграмі можна побачити, яку владу мають різні сторони на програмний модуль та який інтерес проявляють до цього модуля.

Внутрішні сторони: Користувачі, парт-тайм робітники, розробники, Головний виконавчий директор(CEO), технічна підтримка, тестувальники.

Зовнішні сторони: Фрілансери, конкуренти, музиканти, інвестори.

1.5 Порівняльний аналіз вже існуючих програмних рішень

На сьогодні існує безліч рекомендаційних систем. Їх можна певним чином згрупувати за контентом, щодо якого надаються рекомендації. А саме:

- - аудіо контент(музика,аудіо-книги);
- - відео контент(фільми,серіали,реклмані ролики,розважальні відео);
- - товари;

- - послуги;
- - новини і т.д.

Кожна рекомендаційна система має свої особливості, оскільки кожна з них спрямована на конкретний тип контенту. Розглянемо найбільш відомих представників у кожній категорії:

1. Spotify: Це найпопулярніший сервіс для відтворення цифрових аудіо-потоків, який має найбільшу аудіо-бібліотеку. Він дозволяє користувачам легально і безкоштовно прослуховувати музичні композиції. Spotify був першим сервісом стримінгу музики, який дозволив слухати музику онлайн. Він доступний в США, більшості країн Європи, окремих країнах Азії, Австралії та Новій Зеландії. У нього потужна рекомендаційна модель.

2. Google News: Це безкоштовний агрегатор новин, який створений і управляється компанією Google Inc. Сервіс був запущений у 2002 році. Рекомендації, надані в Google News, базуються на користувацьких даних, таких як вік, стать, національність, а також на попередніх пошукових запитах у системі Google.

Кожна з цих рекомендаційних систем має свої особливості і надає рекомендації на основі конкретних даних та контексту.Netflix - це сервіс трансляцій на основі підписки, який дає користувачам змогу переглядати серіали й фільми без реклами на пристроях, підключених до Інтернету.

YouTube - відеохостинговий сайт, що надає користувачам послуги зберігання, доставки та показу відео. Користувачі можуть завантажувати, переглядати, оцінювати, коментувати, додавати до вибраного та ділитися темою або іншими відеозаписами. Завдяки простоті та зручності використання YouTube став найпопулярнішим відеохостингом і другим сайтом у світі за кількістю відвідувачів. Надає своїм користувачам рекомендації 2х типів: 1) світові тренди, тобто відео, що є найпопулярнішими в даний момент; 2) особисті рекомендації – відео які пропонуються на основі минулих переглядів конкретного користувача.Виходячи з вищезазначеного можна зрозуміти, що зараз ніша рекомендаційної дуже затребувана та корисна для бізнесу. Саме

тому розробка рекомендаційної системи є дуже доцільною, особливо якщо врахувати розвиток RS за останні декілька років.

1.6 Переваги та недоліки рекомендаційної системи

У рекомендаційних системах існує низка проблем, таких як зміна інтересів користувача, шахрайство та інші. Серед найбільш поширених проблем можна виділити холодний старт, конфіденційність, надмірну спеціалізацію, масштабованість, розрідженість та передбачуваність. Холодний старт можна розділити на дві категорії: холодний старт для нових елементів і нових користувачів. Проблема холодного старту для елемента виникає, коли недостатньо попередніх оцінок, пов'язаних з цим елементом. Так само, рекомендація товарів новим користувачам стає складнішою, оскільки система не має жодної інформації про їх попередні покупки, оцінки або вподобання. Для зменшення проблеми холодного старту можна використовувати демографічну інформацію користувача, зокрема застосовувати спільну фільтрацію, яка сприяє формуванню рекомендацій.

Розрідженість

Однією з основних проблем при створенні рекомендаційних систем є розрідженість даних. Це явище виникає через велику кількість користувачів і об'єктів на платформі. У результаті взаємодії між ними формується матриця, але більшість її елементів мають значення нуль, оскільки не було отримано жодного зворотного зв'язку від споживача. [17].

Масштабованість

З огляду на зростання кількості користувачів і елементів, рекомендаційній системі потрібно використовувати більше ресурсів для надання найточніших рекомендацій користувачам. Більшість ресурсів приділяється для визначення користувачів зі схожими смаками та предметів зі схожими атрибутами. У зв'язку зі зростанням масштабу галузевих наборів даних та використанням складніших моделей, збільшилась потреба у обчислювальних ресурсах для рекомендаційних систем.

Конфіденційність

У демографічних рекомендаційних системах виникає значна проблема зі збереженням конфіденційності. Для досягнення найвищої точності рекомендацій, система потребує доступу до особистої інформації користувача, включаючи його персональні дані та дані про місцезнаходження. Однак це може викликати проблеми з конфіденційністю самого користувача.

Холодний старт

Проблему холодного старту можна розділити на дві категорії: холодний старт нових елементів і холодний старт нових користувачів. Холодний старт елемента виникає, коли немає достатньо попередніх оцінок, пов'язаних з цим елементом. Так само, важко рекомендувати товари новим користувачам, оскільки система не має жодної інформації про їх попередні покупки, оцінки або відгуки.

Для обмеження проблеми холодного старту можна використовувати демографічну інформацію користувача, зокрема шляхом використання спільної фільтрації, що дозволить формувати рекомендації.

1.7 Постановка задачі

Метою дипломної роботи є розробка програмного модуля підбору музичних композицій для полегшення вибору музики та веб-додатку

Завдання дипломної роботи – Розробити програмний модуль підбору музичних композицій для користувача , який надає назву своїх улюблених пісень і потім отримує список пісень.

Об’єкт дослідження – методи підбору музичних композицій з використанням кластерного аналізу та методом колоборативної фільтрації

Предмет дослідження – оцінки користувачів на музику та зв’язки між ними.

Для користувача програмний модуль має вигляд «чорного ящика», так як він бачить лише її результат. Вхідними даними є улюблені пісні користувачів. Також, передається інформація про споживача. Управління

такою системою виконують алгоритми формування рекомендацій.. Так як користувач вказує свої улюблені пісні, які потім використовуються для формування рекомендацій.

Основними компонентами рекомендаційної системи є збір даних, їх обробка і формування рекомендацій. Збір даних включає нормалізацію та обробку отриманих даних. Обробка даних включає формування матриці оцінок, її нормалізацію і підготовку до використання в системі. Останнім етапом є формування рекомендацій, що включає введення даних, застосування алгоритмів кластерного аналізу або колаборативної фільтрації і створення персоналізованого списку рекомендацій топ-N музичних композицій.

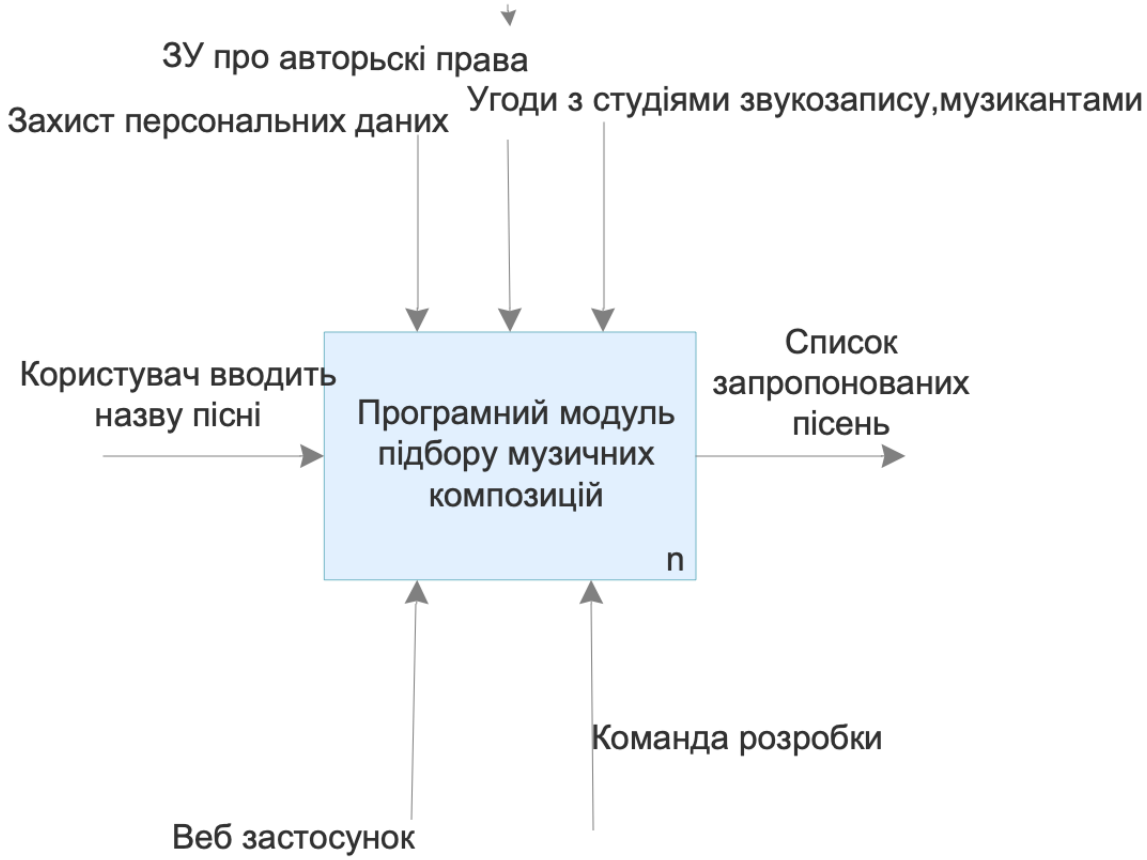

Рисунок1.2 - Опис системи в нотації IDEF0

1.8 Опис функціональних та нефункціональних вимог

1.8.1 Функціональні вимоги

Функціональні вимоги до програмного модуля:

- зручний та зрозумілий україномовний інтерфейс
- надання топ-N найпопулярніших музичних композицій
- надання топ-N персоналізованих рекомендацій
- Можливість обрання кількості надання рекомендацій

1.8.2 Нефункціональні вимоги

До нефункціональних вимог відноситься наступні:

- Надійність: якщо користувач робить некоректні дії, то система буде реагувати на це та виправляти.
- Швидкодія: рекомендації будуть надаватись в термін до 1 хвилини.
- Коректна відповідь системи на будь-які дії користувачів

Висновки за розділом 1

У першому розділі дипломної роботи було проведено аналіз предметної області. Було проведено огляд існуючих моделей та методів рекомендаційних систем, описано недоліки та переваги таких систем. Особлива увага була приділена увага сферам застосуванням рекомендаційних систем у сучасному бізнесі. Була поставлена задача на розробку та визначені функціональні та нефункціональні вимоги програмного модуля.

РОЗДІЛ 2. РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМНОГО МОДУЛЯ ПІДБОРУ МУЗИЧНИХ КОМПОЗИЦІЙ

2.1 Архітектура модульного підбору музичних композицій

Для детального розгляду архітектури модулю підбору музичних композицій та усвідомлення всіх тонкощів реалізації у даному розділі, буде приділена увага, її особливостям, перевагам та недолікам. Також будуть описані обрані методи реалізації,. Буде розглянуто архітектуру та специфіку програмного модуля підбору музичних композицій.

Інтелектуальний модуль для підбору музики бере на вхід дані, які користувач сам ввів на веб-сайті.

Програмний модуль підбору музичних композицій буде включати в себе дві основні функції:

Отримання данних:

- Користувач вводить назву улюбленої пісні
- Перевірка на наявність пісні у датасеті
- Нормалізація даних

Функція побудови моделі програмного модуля підбору музики буде включати у себе наступні підфункції:

- Користувач обирає метод
- Реалізація підбору за допомогою методу K-Means
- Реалізація підбору за допомогою методу колоборативної фільтрації

Формування результатів програмного модуля:

- Отримання та перевірка даних
- Формування n-кількості рекомендацій
- Надання списку рекомендацій

Із описаного вище функціонального аналізу побудуємо дерево функцій, яке зображене на рисунку 2.3

Вибір методу пошуку пісень замість побудова РС Розгалуження від обрання методу

Підбір пісень за допомогою методу

Виведення результатів користувачу

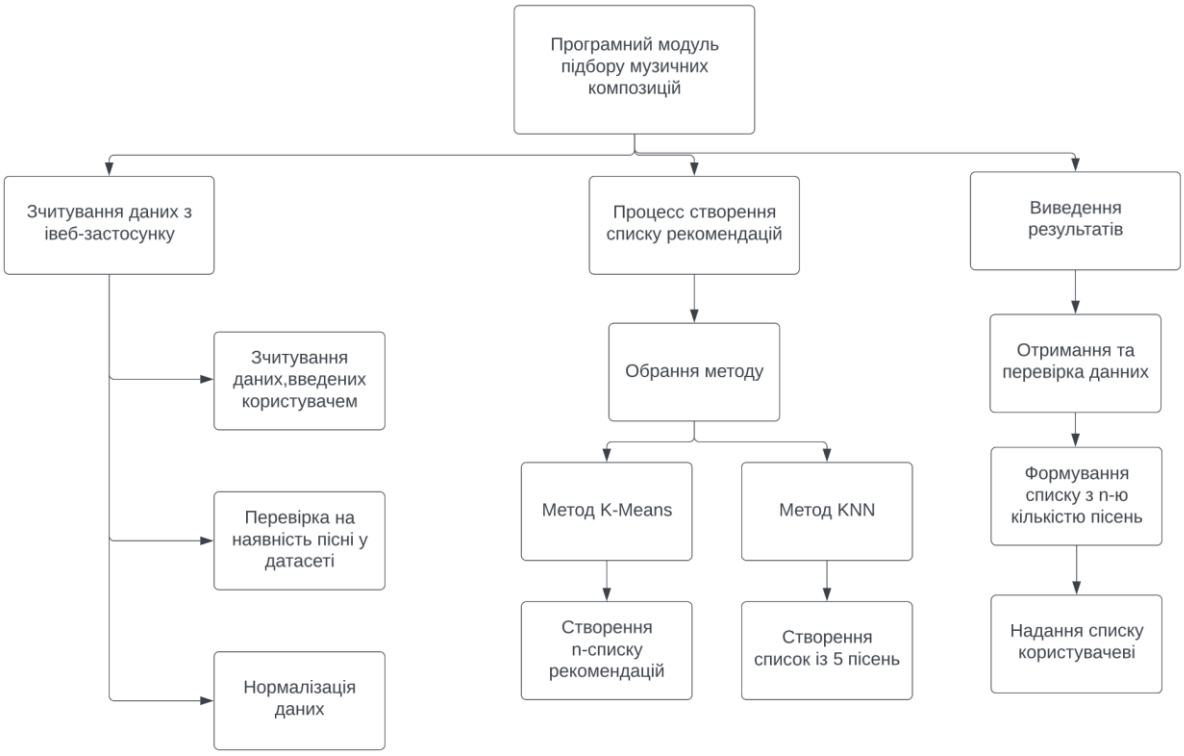

Рисунок 2.1- Дерево функцій системи

Можна легко описати карту процесів. Побудова моделі рекомендацій музики складається із чітко визначених послідовних функцій що включає: очистку даних, нормалізації, усунення дисбалансу, розподілення даних на кластери , пошук із кластерних груп найближчі значеня, щоб обрати надання рекомендацій .

Нижче зображено VAD діаграму, яка показує реакцію програмного модуля на дії користувача. Користувач заходить на веб-застосунок і потрапляє на головну сторінку. Після натискання на кнопку старт, користувача перекидає на сторінку, де йому надають обрати, яку кількість пісень він хоче ввести, щоб йому надало рекомендацію. Після обрання, його переводить на сторінку де користувач має змогу ввести пісню або пісні. Після натискання кнопки надати рекомендацію запускається програмний модуль , яка формує n кількість пісень

для надання користувачеві. Для кожного з підходів описано короткий алгоритм , який повертає рекомендації

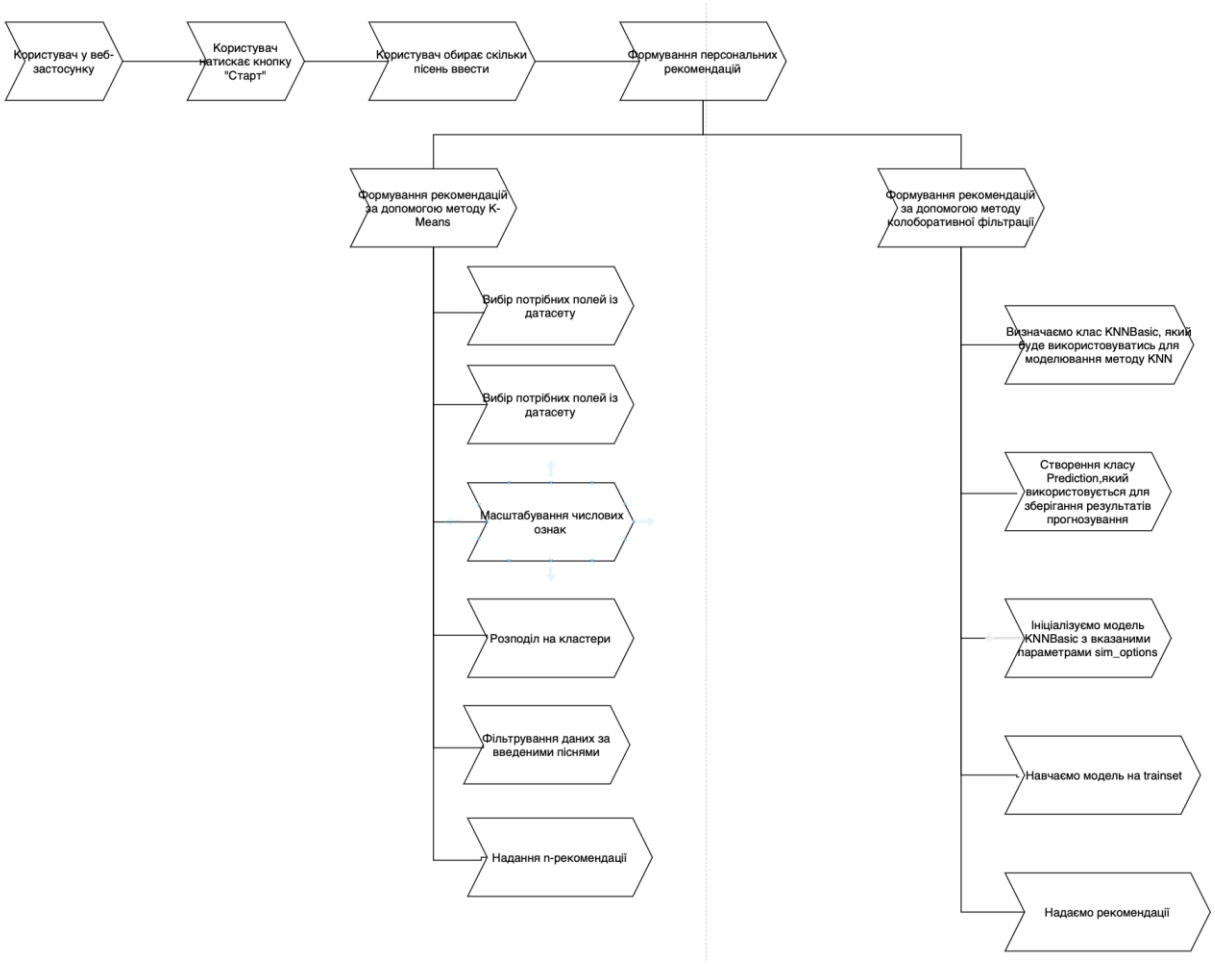

Рисунок 2.2 - VAD діаграма роботи програмного модуля

Застосунок буде мати клієнт-серверну архітектуру; відповідні вузли представлені на діаграмі як «Client» та «Server».

Далі було побудовано узагальнену архітектуру використовуючи StarUML. Для визначення компонентів програмного модуля побудуємо компонентну діаграму (рисунок 2.3).

Клієнтський вузол буде мати наступні компоненти:

- збір даних користувача (Data Collection);
- клієнтський інтерфейс (User Interface).

Серверна частина буде мати наступні компоненти:

- підготовка даних (Data Preprocessing);
- навчання (Training);

− прогнозування (Prediction).

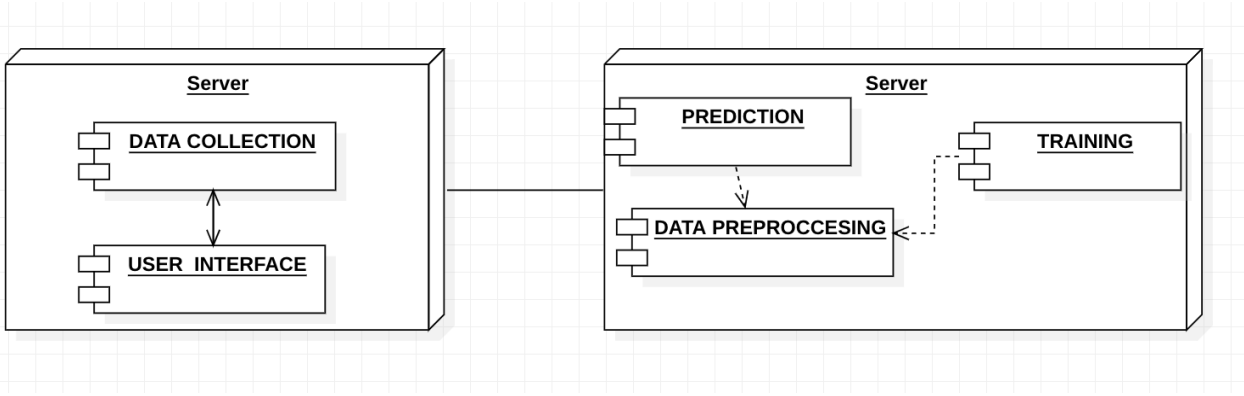

Рисунок 2.3 - Компонентна архітектура застосунку

У веб-застосунку рекомендаційна система складатиметься з двох компонентів: клієнтської частини та серверної частини. Клієнтська частина буде відповідати за відображення та візуалізацію даних системи, а також за надання основного функціоналу користувачам. Користувачі зможуть переглядати локально інформацію про дані та здійснювати вибірки даних. Обрані дані будуть надсилатись на сервер для подальшої обробки. Серверна частина відповідатиме за обробку даних системи та виконання рекомендацій з використанням обраних методів. Він оброблятиме запити від клієнта і автоматично надсилатиме результати на клієнтську частину. Крім того, сервер буде відповідальний за формування списків пісень та їх відображення . На веб-сайті користувачі матимуть можливість вибирати необхідну інформацію, отримувати результати для кожного з методів рекомендаційної системи. Таким чином, програмний модуль буде включати клієнтську та серверну частини, які співпрацюватимуть для надання рекомендаційного функціоналу та обробки даних вашої рекомендаційної системи.

Даний модуль можна описати за допомогою діаграми прецедентів (рисунок 2.4), яка буде описувати можливості користувача та системи. Для користувача буде доступним ввести необхідну інформацію,щоб отримати вдалу рекомендацію

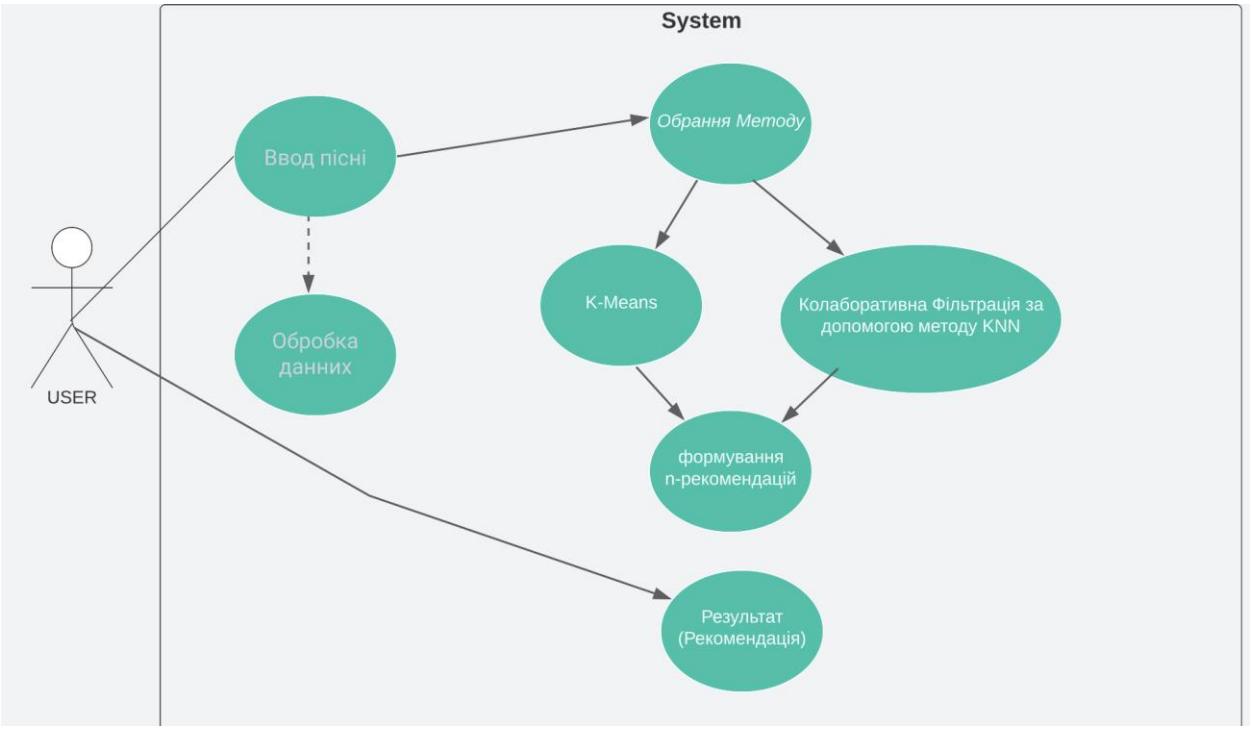

Рисунок 2.4 – UML діаграма прецедентів

Також для наглядної демонстрації процесів, які включає програмний модуль підбору музичних композицій модуль, була підготовлена діаграма діяльності, яка представлена на рисунку 2.5 Ця діаграма відображає послідовність дій, що відбуваються в програмі, і показує взаємозв’язок між ними. Після того, як користувач зайшов у веб-застосунок і обрав метод завдяки якому він хоче отримати рекомендацію, він вводіть свою улюблену пісню , відбувається блок обробки, включаючи структуризацію, очищення даних, нормалізацію та усунення дисбалансу. Потім відбувається процес навчання, класифікації та оцінки результатів. Нарешті, програма завершує свою роботу і надає рекомендацію користувачу .Цей опис відображає послідовний характер дій, які відбуваються в програмному модулі, і показує, що кожен крок залежить від попереднього.

Рисунок 2.5 – UML діаграма діяльності

2.2 Опис інструментів для реалізації програмної частини

2.2.1 Python

Для реалізації практичних задач, які поставленні у дипломній роботі ,було використано мову програмування Python. Python- це мова програмування, яка дозволяє працювати швидше та ефективніше інтегрувати ваші системи. Python широко використовується в наукових та числових обчисленнях. Він має дуже багато бібліотек, які мені потрібні, щоб будувати графіки, застосовувати методи кластерного аналізу, математичні методи та дії.

Мова програмування Python останнім часом все частіше використовується для аналізу даних, як в науці, так і комерційній сфері. Цьому сприяє простота мови, а також велика різноманітність відкритих бібліотек

Бібліотеки, які було використано для написання програмної частини:

Pandas є однією з найпопулярніших бібліотек Python для маніпуляцій з даними та їх аналізу. Основі функції Pandas[11]:

- швидкий та ефективний об’єкт DataFrame з інтегрованим індексуванням;
- інструменти для читання та запису даних між структурами даних у пам’яті та різними форматами: CSV та текстові файли, Microsoft Excel, бази даних SQL;
- гнучка зміна розмірності та типів даних;
- агрегування або перетворення даних: групування, об’єднання, злиття т.д.;
- вбудована візуалізація даних.

NumPy - це бібліотека, що надає математичні функції для роботи з великими масивами даних. Вона отримала свою назву як скорочення від "Numerical Python" (числовий Python) і була розроблена Тревісом Оліфантом у 2005 році. Ця бібліотека надає широкий набір корисних функцій для роботи з n-вимірними масивами та матрицями типу NumPy. Вона відрізняється високою продуктивністю та швидкістю виконання операцій. [12].

Matplotlib є бібліотекою на мові програмування Python, призначеною для візуалізації двовимірних (а також тривимірних) графіків даних. Вона надає можливість створювати зображення, які можуть бути використані в публікаціях або як ілюстрації. Matplotlib розроблена та підтримується в основному Джоном Хантером і поширюється під ліцензією, схожою на BSD-ліцензію. Зображення, створені за допомогою Matplotlib, можуть бути збережені в різних форматах і використовуватись для інтерактивної графіки, наукових публікацій, графічного інтерфейсу користувача, веб-додатках та будь-яких ситуаціях, де необхідно створювати діаграми. [13]

Scikit-learn (також відома як sklearn або scikits.learn) — це безкоштовна програмна бібліотека машинного навчання для мови програмування Python, яка надає функціональність для створення та тренування різноманітних алгоритмів класифікації, регресії та кластеризації, таких як лінійна регресія,

random forest, градієнтний бустинг, і працює у зв’язці з бібліотеками NumPy та SciPy. Scikit-learn є однією з найбільш популярних бібліотек машинного навчання[14].

Spotipy – це легка бібліотека на мові Python для веб-аплікатора Spotify. За допомогою Spotipy ви отримуєте повний доступ до всіх музичних даних, що надаються платформою Spotify.

2.2.2 FLASK

Flask – це легкий фреймворк веб-розробки, написаний на мові програмування Python. Він надає простий інтерфейс для створення веб-додатків, що дозволяє швидко розробляти та масштабувати веб-сайти і веб-додатки. Flask добре підходить для маленьких і середніх проектів, а також для початківців у веб-розробці. Основні особливості Flask:

Основні особливості Flask:

1. Мінімалістичний підхід: Flask пропонує мінімалістичну основу, що дозволяє розробнику самостійно обирати необхідні компоненти та бібліотеки для свого проекту. Це дає велику свободу в розробці та реалізації власних ідей.

2. Маршрутизація: Flask надає можливість визначати URL-шляхи (маршрути) та пов’язані з ними функції-обробники, які виконуються при отриманні запиту на цей URL. Це дозволяє створювати різні сторінки та функціонал веб-додатка.

3. Шаблонізація: Flask має вбудовану систему шаблонів, яка дозволяє розділити логіку додатка від представлення. Шаблонізація дозволяє легко створювати HTML-сторінки з динамічним вмістом, вставляти дані з сервера в шаблони та контролювати вигляд веб-сторінок.

4. Розширюваність: Flask має велику кількість розширень (розширювальних модулів), які допомагають розширити функціонал фреймворка. Ці розширення дозволяють додавати підтримку баз даних, форм, аутентифікації, обробки файлів та багато іншого.

5. Локальний сервер: Flask має вбудований локальний сервер, що дозволяє запускати веб-додаток локально для тестування та розробки без необхідності налаштування окремого веб-сервера.

Процес роботи з Flask зазвичай включає наступні кроки:

1. Встановлення Flask: Встановіть Flask, використовуючи менеджер пакетів Python, такий як pip.

2. Створення додатка: Створіть новий файл Python, в якому імпортується бібліотека Flask та створюється екземпляр класу Flask. Цей екземпляр буде представляти ваш веб-додаток.

3. Визначення маршрутів: Визначте URL-шляхи (маршрути) додатка та пов’язані з ними функції-обробники, які будуть виконуватися при отриманні запиту на цей URL.

4. Створення шаблонів: Створіть HTML-шаблони, які будуть використовуватися для відображення сторінок вашого веб-додатка. Вставте необхідні дані з сервера в шаблони.

5. Запуск сервера: Запустіть локальний сервер Flask, щоб перевірити роботу вашого веб-додатка. Він буде слухати запити на вказаному вами порту.

6. Тестування та вдосконалення: Перевірте роботу вашого веб-додатка, виконавши різні дії, і вдоскональте його за необхідності.

Flask є потужним і гнучким фреймворком, який допомагає розробникам швидко створювати веб-додатки з використанням мови програмування Python. Він надає зручні інструменти для обробки запитів, відображення сторінок, роботи з базами даних та іншими аспектами веб-розробки.

2.3 Методи вимірювання якості та точності рекомендацій

2.3.1 Релевантність рекомендацій

Для визначення релевантності рекомендацій найчастіше використовуються prescision, recall та F1.

- Prescision – точність
- Recall – повнота

- F – міра

Prescision обчислює співвідношення продуктів, які подобаються користувачеві серед рекомендованих, до всіх в рекомендованому списку. Показано точність рекомендацій користувача u у формулі (2.1):

$$P(L_u) = \frac{L_u \cap B_u}{L_u} \quad (2.1)$$

де Lu – це список рекомендацій користувача u;

Bu – це реальний набір даних про користувача u.

Recall розраховує співвідношення продуктів, які подобаються користувачам у рекомендованому списку, до всіх продуктів, які подобаються користувачам у системі. Рекомендований recall користувача u показано у формулі (2,2)[3]:

$$R(L_u) = \frac{L_u \cap B_u}{L_u} \quad (2.2)$$

Prescision та recall важко покращити одночасно. Обидва вони часто негативно корелюють і залежать від довжини рекомендованого списку. Тому для обчислення зазвичай використовується F1 – середнє гармонійне з двох. Чим більше значення F1, тим кращий ефект прогнозу. Розрахунок показано у формулі (2.3):

$$F1 = \frac{2*PR}{P+R} \quad (2.3)$$

2.3.2 Точність прогнозного рейтингу

Показники точності прогнозного рейтингу визначаються шляхом вимірювання відхилення між прогнозованими алгоритмом оцінками і фактичними рейтингами користувачів. Ці показники є основними метриками для оцінки ефективності рекомендаційної системи. Один з таких показників -

середня абсолютна похибка (MAE) - обчислює абсолютну величину відхилення між фактичним рейтингом користувача і прогнозованим рейтингом. Формула для обчислення MAE наведена нижче:

$$MAE = \frac{\sum_{u,i}^{T} |r_{ui} - r_{ui}|}{|T|}, \quad (2.4)$$

де rui – це фактична оцінка користувача u продукту i;

rui` – прогнозована оцінка, надана рекомендаційним алгоритмом;

T(u) список поведінки користувача в тестовому наборі.

Крім того, середня квадратична помилка (MSE), корінь від середньої квадратичної помилки (RMSE) і нормалізована середня абсолютна помилка (NMAE) [15] є всіма показниками, подібними до MAE.

$$MSE = \frac{\sum_{u,i}^{T} (r_{ui} - r_{ui})^2}{|T|}, \quad (2.5)$$

$$RMSE = \sqrt{\frac{\sum_{u,i}^{T} (r_{ui} - r_{ui})^2}{|T|}} \quad (2.6)$$

$$NMAE = \frac{MAE}{r_{max} - r_{min}} \quad (2.7)$$

де rmax і rmin є максимальним і мінімальним значенням інтервалу оцінки користувачів.

MSE і RMSE застосовують суворі штрафи: чим більше відхилення у прогнозі, тим більше покарання. Масштаби MAE, MSE і RMSE лежать в діапазоні [0, +∞). NMAE нормалізує значення MAE в межах оцінюваного інтервалу, що дозволяє порівняти ефективність одного і того ж рекомендаційного алгоритму на різних наборах даних.[20]

2.4 Огляд методів кластерного аналізу

Основна ідея кластерного аналізу - виділення серед множини даних груп, всередині яких елементи в певній мірі схожі. В рамках такого аналізу, по суті, відбувається певна класифікація досліджуваних даних за рахунок

розподілу їх по групах. Групи впорядковані ієрархічно.Структуру, отриману після аналізу кластерів можна представити у вигляді дерева - дендрограми.

Існує безліч практичних застосувань кластеризації як в інформатиці, так і в інших областях. Ось кілька прикладів застосування кластеризації :

— аналіз даних, спрощення роботи з інформацією, візуалізація даних;

— витяг і пошук інформації;

— групування і розпізнавання об'єктів.

Кластерний аналіз можна представити у вигляді такої послідовності дій :

— вибір множини об'єктів;

— визначення можини змінних, для оцінки об'єктів і складання векторів характеристик;

— нормування векторів характеристик одним з доступних методів; — визначити подібність між об'єктами за заданою метрикою;

— застосування обраного методу кластерного аналізу для розбиття множини об'єктів на кластери зі ступеня схожості;

— представлення результатів аналізу.

Достатньо багато досліджень ставлять за мету організацію отриманих даних у наглядні структури. Так, в бізнес-аналітиці часто використовують кластерний аналіз,для сегментації споживачів та знаходження груп з пов’язаними ознаками . Фактично, кластерний аналіз є набором різноманітних алгоритмів класифікації. Загалом, коли необхідно класифікувати великі масиви інформації на групи, які придатні для подальшого аналізу – кластерний аналіз є незамінним інструментом[21]

Кластерний аналіз відрізняється від звичайних статистичних методів тим, що в більшості випадків не використовується процес перевірки статистичної значущості. Його особливість полягає в тому, що він надає найбільш значимі рішення. Тому його часто використовують, коли дослідник має набір даних, але не має жодної попередньої гіпотези про класифікацію цих даних.[21]

Дендрограма –деревоподібна діаграма, що містить n рівнів, кожний з яких відповідає одному із кроків процесу послідовного укрупнення кластерів

У багатьох методах пошуку подібностей використовують лінійні міри, що може призвести до значного збільшення кількості порівнянь і вплинути на швидкодію системи. Однак існують методи кластеризації, такі як k-means або LSH, які допомагають вирішити цю проблему. Ці методи дозволяють розбити множину об'єктів на окремі підмножини (кластери), де елементи знаходяться близько один до одного. Після такого розбиття, запити до системи можна здійснювати лише в межах відповідного кластеру, оскільки елементи кластеру є найбільш подібними один до одного. Це дозволяє забезпечити більш оптимальні рекомендації. Наприклад, метод LSH використовує спеціальні хеш-функції, які максимізують кількість колізій. Однак в даному випадку зосередимося на алгоритмі k-means, який є ширше використовуваним. [13].

2.5 Застосування методу K-середніх(K-Means) для підбору музичних композицій

Для реалізації рекомендаційної системи за методом k-means спочатку було підготовлено дані. Першим кроком було завантаження даних з файлу CSV за допомогою функції `open_csv()`. Завантажені дані включали різні атрибути про музичні треки, такі як виконавці, тривалість, танцювальність, назва, дата виходу, темп та енергія. Далі, з вибраних атрибутів було створено набір даних `music_data`, який містив лише потрібні поля. У випадку наявності пропущених значень, вони були оброблені шляхом видалення відповідних записів з набору даних. Для числових атрибутів, таких як тривалість, танцювальність, темп та енергія, було проведено масштабування. Значення цих атрибутів було віднято від середнього значення та поділено на стандартне відхилення. Це дозволяє зрівняти значення в межах діапазону. Наступним кроком було визначення кількості кластерів для алгоритму k-means. У даному випадку було обрано 10 кластерів. Кількість кластерів була обрано за допомогою методу ліктя(Elbow method) Цей метод полягає в оцінці суми

квадратів відстаней внутрішньокластерних точок до центроїдів кластерів для різних кількостей кластерів. Зазвичай, графік залежності суми квадратів відстаней від кількості кластерів має форму "ліктя". Оптимальна кількість кластерів може бути визначена як точка згину на цьому графіку. Це продемонстровано на рисунку 2.1

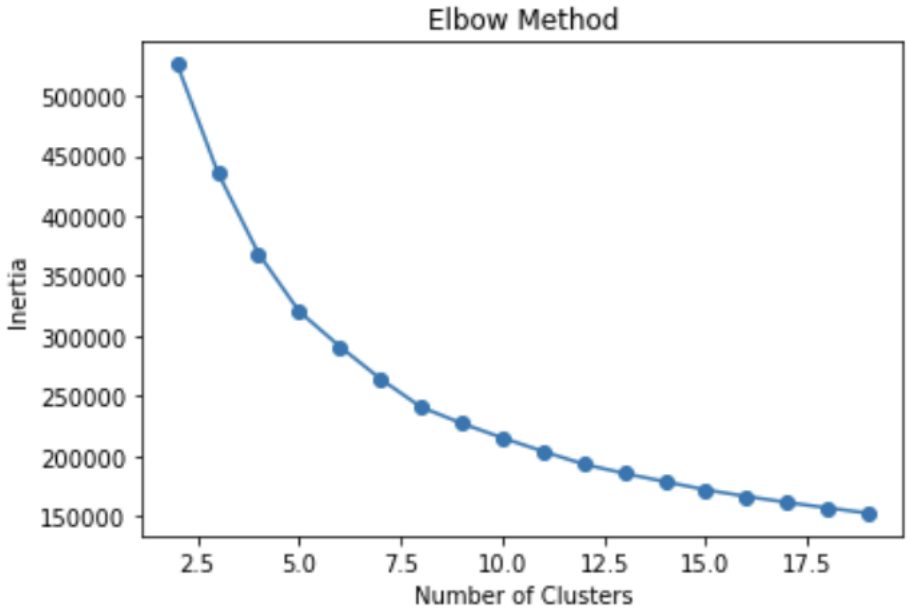

Рисунок 2.1 - Реалізація методу “Ліктя”

Для застосування алгоритму K-Means було використано бібліотеку `sklearn` із використанням об'єкту `KMeans`. Після навчання алгоритму на наборі даних, було додано мітки кластерів до даних. Функція `results(music_data, num_recommendations, input_songs)` використовує попередньо оброблені дані та вхідні пісні користувача для отримання рекомендаційних пісень. Спочатку вона фільтрує дані за введеними піснями, після чого отримує унікальні мітки кластерів, в яких знаходяться введені пісні. Далі, створюється порожній DataFrame `recommended_songs` для збереження рекомендаційних пісень. Процес вибору пісень полягає у виборі певної кількості пісень з кожного кластера, в яких знаходяться введені пісні. Цей процес повторюється для кожного унікального кластера, що дозволяє отримати рекомендації з різних кластерів. Отримані рекомендаційні пісні виводяться як результат функції `results()`.

2.6 Застосування колаборативної фільтрації з використанням K-найближчих сусідів для підбору музичних композицій

K-найближчих сусідів (KNN) є одним із найпростіших і широко використовуваних методів машинного навчання у сфері класифікації та регресії. Він відноситься до навчання з учителем та заснований на ідеї схожості об'єктів. У методі KNN, коли ми отримуємо новий приклад, ми шукаємо K найближчих до нього прикладів з навчального набору даних. Результат класифікації (у випадку класифікації) або прогнозоване значення (у випадку регресії) нового прикладу визначається більшістю класів або середнім значенням відповідних значень з його K найближчих сусідів. Основні кроки методу KNN:

1. Визначення значення K - кількості найближчих сусідів, які будуть використовуватись для класифікації або прогнозу.
2. Вимірювання схожості між новим прикладом та всіма прикладами з навчального набору даних.
3. Це може використовувати різні метрики схожості, такі як Евклідова відстань, косинусна схожість або інші. Вибір K найближчих сусідів на основі обчисленої схожості.
4. Визначення прогнозованого значення для нового прикладу на основі класів (у випадку класифікації) або значень (у випадку(регресії) його K найближчих сусідів.[18]

Згідно з автором[19] - Параметр k у методі kNN часто вибирається на підставі досвіду чи знань про розв'язувану задачу класифікації. Бажано, щоб параметр k був непарним, щоб зменшити імовірність «нічиєї». Найчастіше вибираються значення k = 3 і k = 5, але використовуються і великі значення, між 50 і 100. Як альтернативу параметр k можна вибрати так, щоб він гарантував найкращі результати на відкладеній частині навчального множини. Можна також приписати ваги “голосам” k найближчих сусідів, використовуючи їх косинусну міру

$$\text{ранг}(\text{c}, d) = \sum_{d \in S} I_c(d') \cos(\vec{v}(d'), \vec{v}(d')). \quad (2.8)$$

Метод KNN ґрунтується на ідеї, що якщо користувач А має схожі вподобання з користувачем В, то пісні, які подобаються користувачеві В, також можуть сподобатися користувачеві А. Алгоритм знаходить k найближчих сусідів для кожної пісні і використовує їхні вподобання для формування рекомендацій.

У цьому коді алгоритм KNN застосовують для формування рекомендацій пісень на основі схожості між ними.

Одним з важливих аспектів методу KNN є вибір оптимального значення K. Великі значення K можуть призвести до втрати локальних залежностей, тоді як невеликі значення K можуть бути більш схильними до шуму. Оптимальне значення K зазвичай визначається шляхом перехресної перевірки або за допомогою інших методів валідації[18].

Висновки за розділом 2

Розроблено архітектуру програмного модулю та детально розглянуто всі інструменти, які були використані в процесі реалізції рекомендаційної системи: мова Python та його бібліотеки, модуль flask для розробки інтерфейсу;

Окрім того, було описано метод K-Means та колаборативна фільтрація за допомогою методу KNN

Також, у цьому розділі представлені показники оцінки рекомендаційної системи та розглянуто методи визначення точності та релевантності рекомендаційної системи.Зазвичай оберають різні показники, але результат всі очікують однаковий – підвищення рівня задоволеності користувачів додатку та їх зацікавленості.

Основна відмінність між цими методами полягає в тому, що K-Means працює з властивостями об'єктів, тоді як використовує відносини між користувачами і об'єктами. K-Means групує об'єкти зі схожими

характеристиками, незалежно від того, хто їх споживає.В той час, як метод колаборативної фільтрації за допомогою KNN зазвичай прогнозує інтерес користувача на основі схожості з іншими піснями.

РОЗДІЛ 3. ФУНКЦІОНАЛЬНИЙ АНАЛІЗ ТА ТЕСТУВАННЯ ПРОГРАМНОГО МОДУЛЯ ПІДБОРУ МУЗИЧНИХ КОМПОЗИЦІЙ

3.1 Графічно-розвідувальний аналіз датасету spotify

У сучасному світі музика є повсюдною - майже всі люблять слухати музику. З появою потокових платформ кількість доступної музики значно зросла.Датасет від spotify надає нам великй об’єм даних.. Цей набір даних базується на підмножині користувачів з набору даних #nowplaying, які публікують свої твіти #nowplaying через Spotify. В принципі, набір даних містить користувачів, їхні плейлисти та треки, що містяться в цих плейлистах.

Файл `data.csv` зі Spotify датасету містить інформацію про пісні, зібрану з різних джерел та аналітичних даних. Датасет складається з наступних стовпців:

1. `id`: Унікальний ідентифікатор пісні.

2. `name`: Назва пісні.

3. `artists`: Виконавець або виконавці, які виконують пісню. Якщо є кілька виконавців, вони перераховані через кому.

4. `danceability`: Характеристика, що відображає наскільки пісня придатна для танцю. Значення знаходяться в діапазоні від 0 до 1, де 1 вказує на високу придатність для танцю.

5. `energy`: Енергійність пісні. Значення знаходяться в діапазоні від 0 до 1, де 1 вказує на високу енергію.

6. `key`: Тональність пісні. Цей параметр представлений числовим значенням, що відповідає за тоновість пісні.

7. `loudness`: Гучність пісні в децибелах (dB). Вона представлена в числовому форматі.

8. `mode`: Режим пісні (0 або 1), де 1 вказує на мажорний режим, а 0 - на мінорний режим.

9. `speechiness`: Показник, що відображає присутність голосу або мовлення в пісні. Значення знаходяться в діапазоні від 0 до 1, де 1 вказує на високу наявність мовлення.

10. `acousticness`: Акустичність пісні. Значення знаходяться в діапазоні від 0 до 1, де 1 вказує на високу акустичність.

11. `instrumentalness`: Інструментальність пісні. Значення знаходяться в діапазоні від 0 до 1, де 1 вказує на високу інструментальність.

12. `liveness`: Показник, що відображає наявність живого виконання в пісні. Значення знаходяться в діапазоні від 0 до 1, де 1 вказує на високу живість.

13. `valence`: Валентність пісні. Значення знаходяться в діапазоні від 0 до 1, де 1 вказує на позитивні емоції.

14. `tempo`: Темп пісні в ударах на хвилину (BPM). Представлено числовим значенням.

15. `duration_ms`: Тривалість пісні в мілісекундах.

16. `explicit`: Показник, що вказує на наявність експліцитного контенту в пісні (0 або 1).

17. `popularity`: Рейтинг популярності пісні. Значення знаходяться в діапазоні від 0 до 100.

Цей датасет надає інформацію про різні аспекти пісень, такі як танцювальність, енергія, настрій, акустичність та інші. Кожен рядок датасету представляє одну пісню та її характеристики. Ця інформація може бути використана для аналізу музичних вподобань, побудови рекомендаційних систем або в інших музичних дослідженнях.

Для кращого розуміння даних, які є в наборі візуалізуємо їх.Можемо побачити на рисунку 3.1 динаміку зміни аудіохарактеристик пісень протягом років та зробити висновок, що останні декілька років, люди полюбляють слухати ту музика, яка є більш енергій та танцювальною.

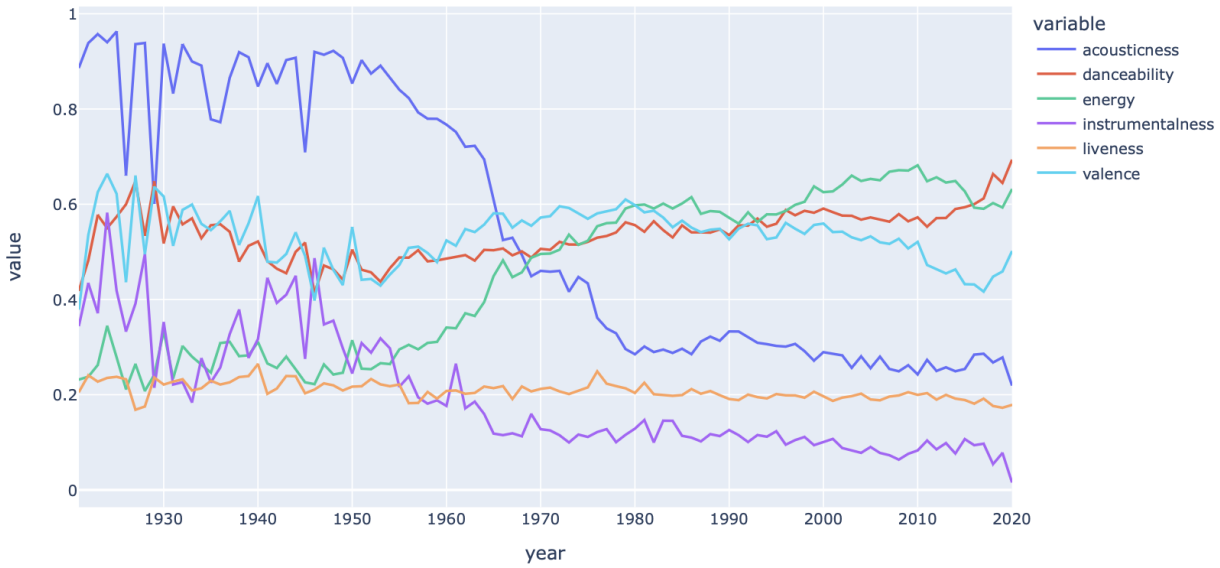

Рисунок 3.1 - Динаміка змін аудіохарактеристик

Також, на рисунку 3.2 візуалізовано топ 10 найпопулярніших жанрів у музиці та їх музикальні ознаки, завдяки цьому рисунку можна зрозуміти на скільки важливі ці дані при реалізації різних методів рекомендацій.

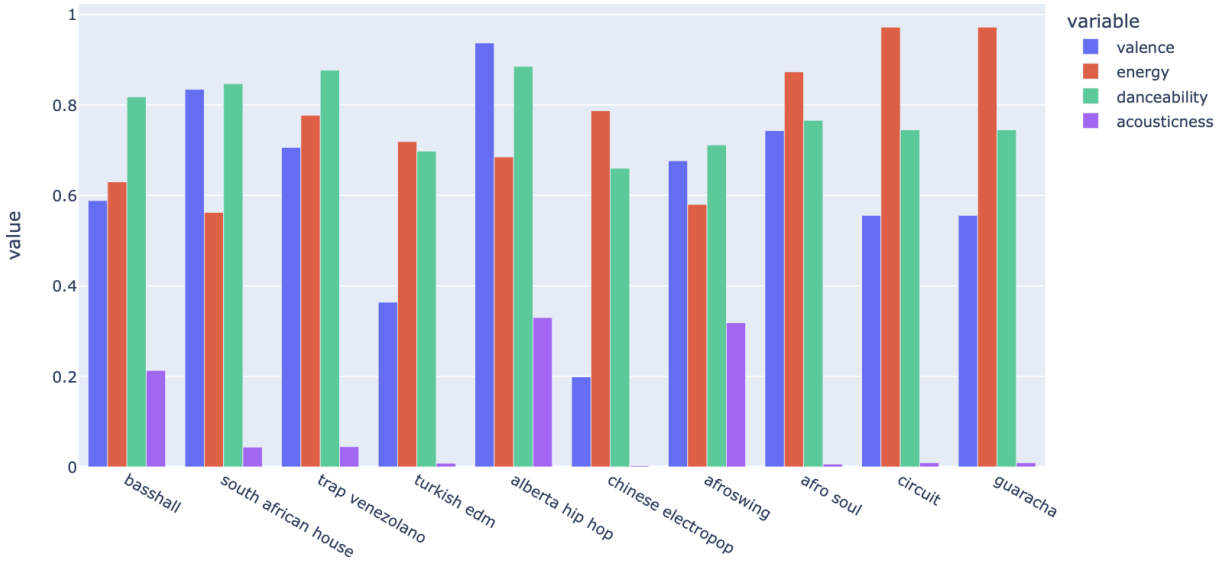

Рисунок 3.2 – ТОП 10 жанрів

Було проведено графічний аналіз для кращого розуміння, що є в датасеті, та які дані потрібні для реалізація програмного модуля та які дані

краще впливають характеристику кожної музичної композиції. Також було розглянуто , як змінювалась аудіо характеристика композицій за всі роки.

3.2 Структура програмного модуля

Застосунок складається з наступних об’єктів(рис. 3.3)

- Папки
  - Templates – містить у собі файли у форматі html. В даній папці знаходяться усі сторінки застосунку
  - Static – містить у собі дві папки “CSS” та “Image”. У даних папках містяться фото, які використовуються у веб-застосунку та файли формату css, які відповідають за дизайн веб-сторінок
- Python-скрипт
  - Main.py - Цей файл містить основну логіку веб-додатка. Він використовує фреймворк Flask для створення веб-сервера. У файлі визначено різні маршрути (`@app.route`), які пов'язують URL-адреси з функціями-обробниками. Наприклад, маршрут `/` відображає головну сторінку, маршрут `/methods` відображає сторінку з методами, і так далі. Файл також імпортує функції з інших модулів, як-от `open_csv` і `results` з `kmeans.py`, `get_recommendations` і `data_music` з `colab_fil.py`, і `get_popularity_recommendations` з `top_charts.py`. Функції-обробники маршрутів виконують певні дії, обробляють запити користувачів і відображають відповідні HTML-шаблони.
  - Colab_fil.py - У цьому файлі визначено класи та функції для реалізації методу колаборативної фільтрації. Основний клас `KNNBasic` являє собою модель, засновану на методі найближчих сусідів (KNN). Він має методи `fit` для навчання моделі та `predict` для передбачення оцінок між двома піснями. Також у файлі визначено клас `Prediction` для зберігання інформації про передбачення. Інший клас `DataMusic`

використовується для представлення даних про музику і включає методи для роботи з цими даними. Функція `get_recommendations` використовує модель `KNNBasic` для передбачення рекомендацій пісень на основі введеної назви пісні.

- Kmeans.py - У цьому файлі містяться функції для роботи з алгоритмом кластеризації K-Means. Функція `open_csv` відкриває файл із даними про музику у форматі CSV, обробляє та масштабує дані, застосовує алгоритм K-Means для кластеризації пісень за їхніми характеристиками та повертає дані з доданими мітками кластерів. Функція `results` використовується для фільтрації та вибору рекомендацій пісень на основі введених користувачем пісень та їхніх кластерів.

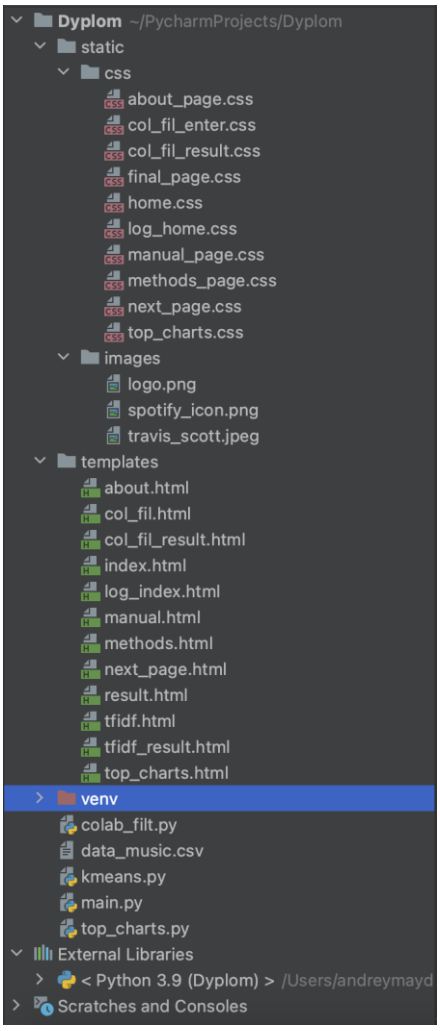

Рисунок 3.3 - структура застосунку

Ці файли разом утворюють веб-додаток, який дає змогу користувачам отримувати рекомендації пісень на основі різних методів і алгоритмів, таких як колаборативна фільтрація і кластеризація.

3.3 Опис переходів між сторінками програмного модуля

На зображенні нижче показано моєливі переходи між сторінками:

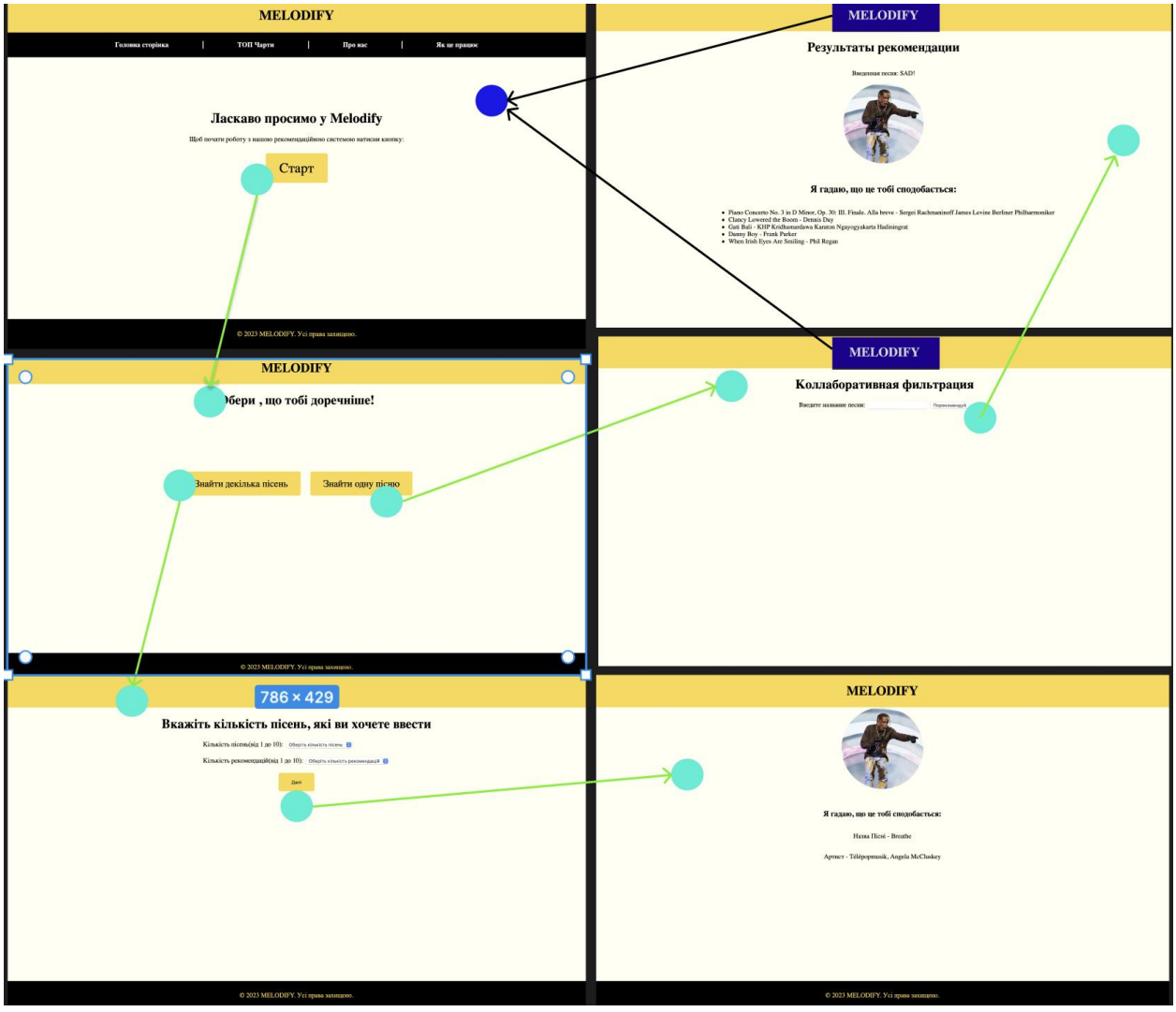

Рисунок 3.4 Макет переходів між сторінками в Figma

Можна побачити, що з головної сторінки користувач може попасти на сторінку, де обирає метод, завдяки якому хоче отримати рекомендації. Щоб потрапити на цю сторінку, він має натиснути на кнопку “Старт”. Далі, коли

користувач на сторінці де має змогу обрати два метода, він має змогу натиснути на одну з кнопок. Якщо користувач натиснув на кнопку “Знайти декільна пісень”, то він потрапляє на нову сторінку де він має змогу обрати бажану кількість пісень , яку він хоче ввести та бажану кількість рекомендацій. Далі, коли користувач ввів усі дані у нього з’является можливість натиснути кнопку “Надати рекомендацію”. Після натискання на кнопку, його переадресовує на нову сторінку де віно може побачити n-кількість рекомендацій

Якщо користувач натиснув на кнопку “Ввести 1 пісню”, то його перенаправить на сторінку де він має змогу ввести лише 1 пісню. Після того, як користувач ввів назву пісні, з’являється змога натиснути кнопку “Порекомендуй”. Коли він натисне на цю кнопку, його перенаправить на нову сторінку , де він побачить 5 рекомендацій у вигляді Пісня – Автор

Також на кожній сторінці є змога натиснути на логотип – MELODIFY, який перенаправить користувача на головну сторінку.Всю схему переходів можна побачити на рисунку 3.4, де зображено граф переходів.

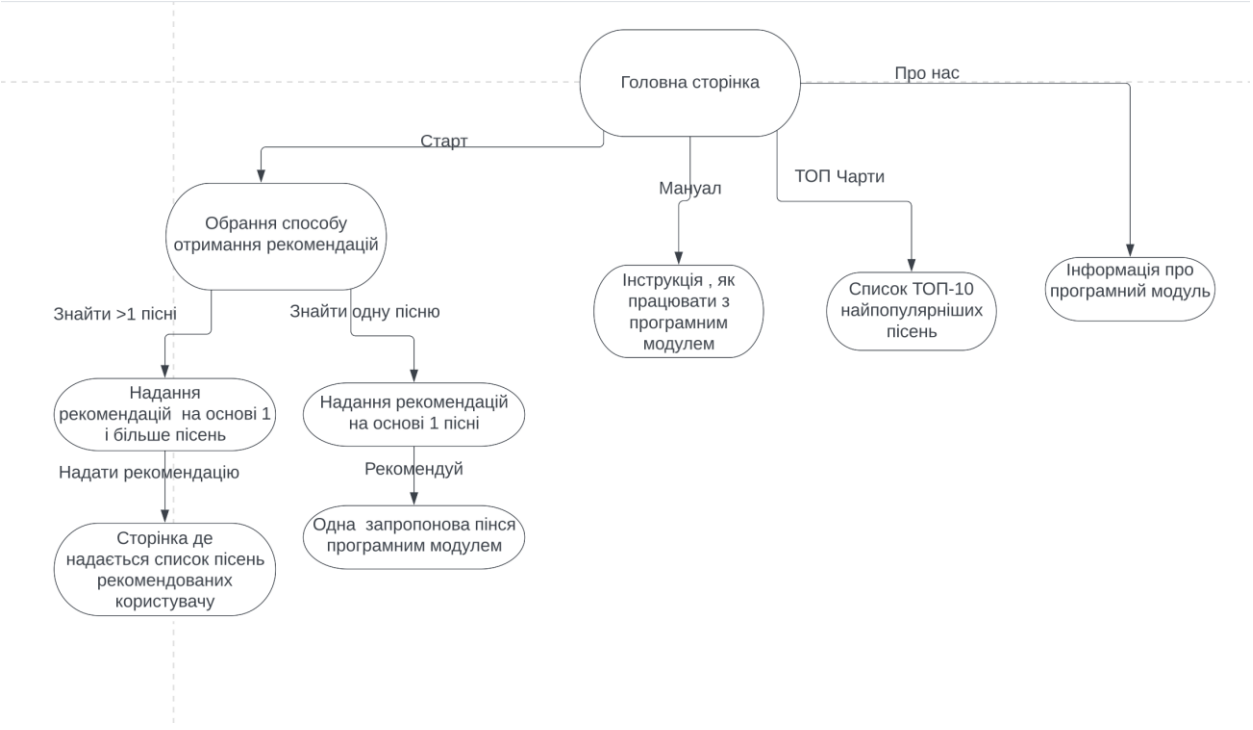

Рисунок 3.5 – Граф переходів застосунку

3.4 Опис інтерфейсу програмного модуля з підбору музичних композицій

3.3.1 Опис основних сторінок

Сторінка, на яку потрапляють користувачі вперше - це головна сторінка (Рисунок 3.5), на якій є вверхня панель та кнопка Старт, щоб як найшвидше перейти до роботи з рекомендаційною системою.

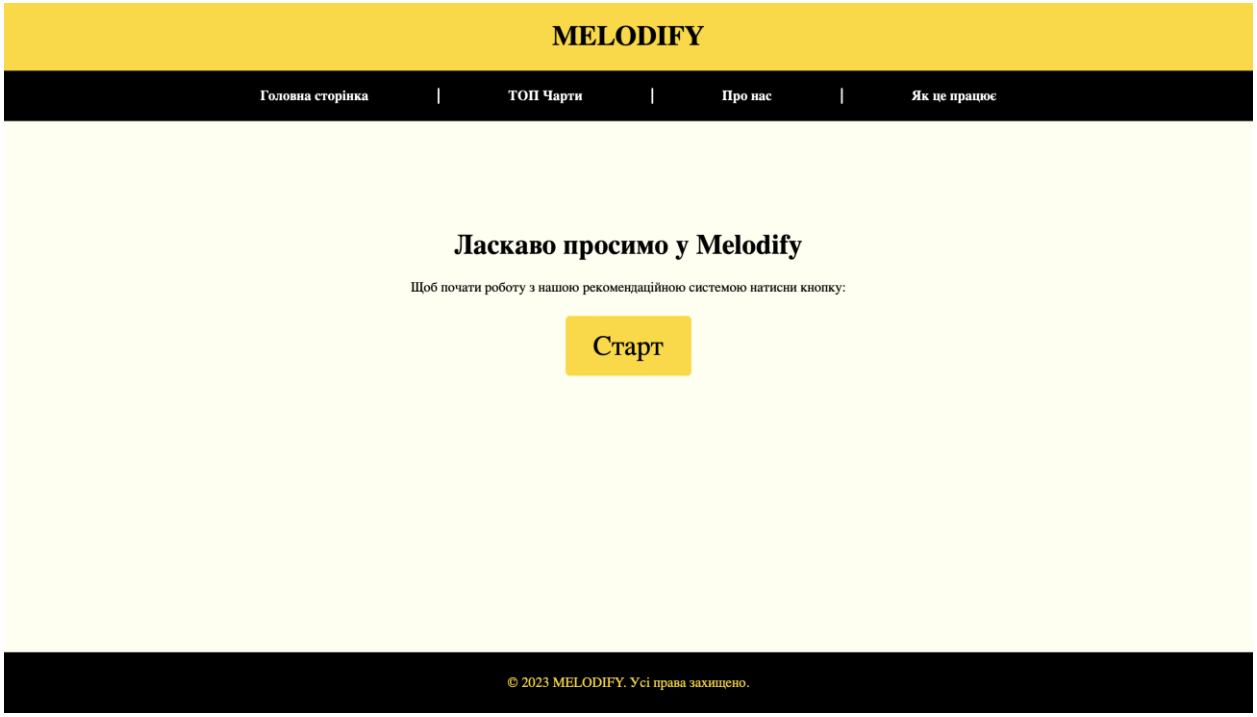

Рисунок 3.6 - Головна сторінка

Далі користувач натискає кнопку “Старт” та переходить на сторінку(рис 3.6) де він має змогу обрати метод завдяки якому він хоче отримати рекомендацію.

Рисунок 3.7 - Сторінка для обрання методу

Якщо користувач обирає метод “Знайти декільна пісень”, то він переходить на нову сторінку(рис 3.7) де він має змогу обрати кількість пісень, яку він хоче ввести та кількість рекомендацій, яку він хоче отримати.

Рисунок 3.8 - Сторінка для вводу значень

Коли користувач ввів дані в усі поля, він натискає кнопку “Надати рекомендації та переходить на нову сорінку(рис 3.8) де його чекає нові пісні, які на 99% йому сподобаються.

Рисунок 3.9 - Сторінка з наданням рекомендацій

Якщо користувач натиснув на кнопку “Знайти ТОП-5 пісень”,то він потрапляє на наступну сторінку, яку можна побачити на рисунку 3.9

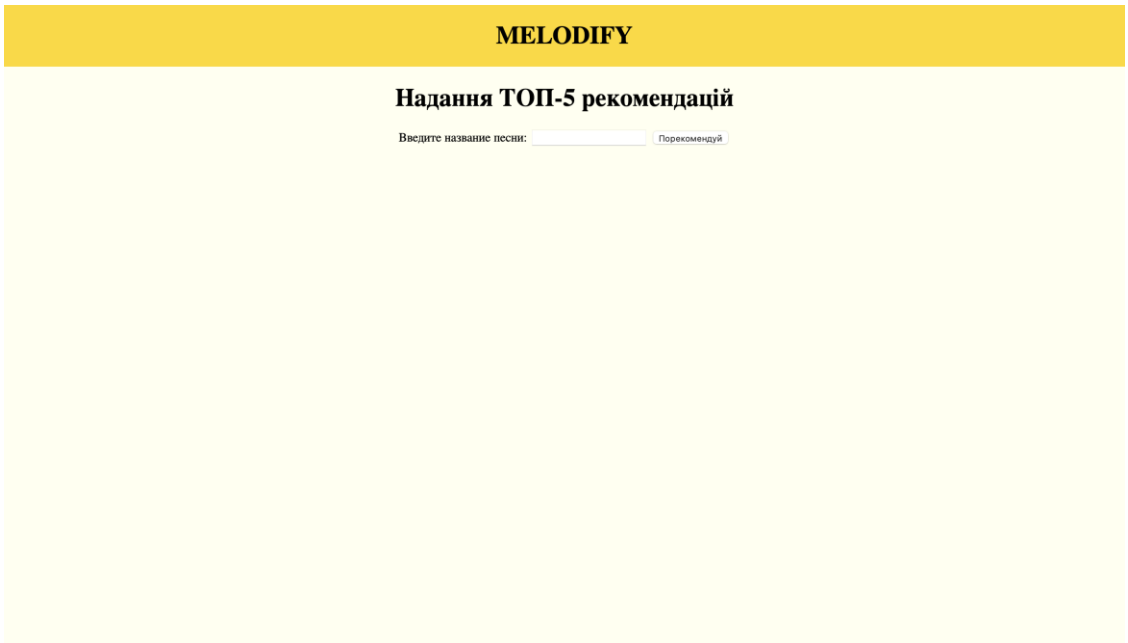

Рисунок 3.10 – Сторінка, де користувач повинен вписати 1 пісню

Коли коруставач ввів свою улюблену пісню, він повинен натиснути на кнопку “Порекомендуй” й далі його перенесе на наступнку сторінку, де йому буде надано рекомендацію у вигляді списку із 5 пісень. На рисунку 3.10 зображено сторінку, де надається список із 5 пісень

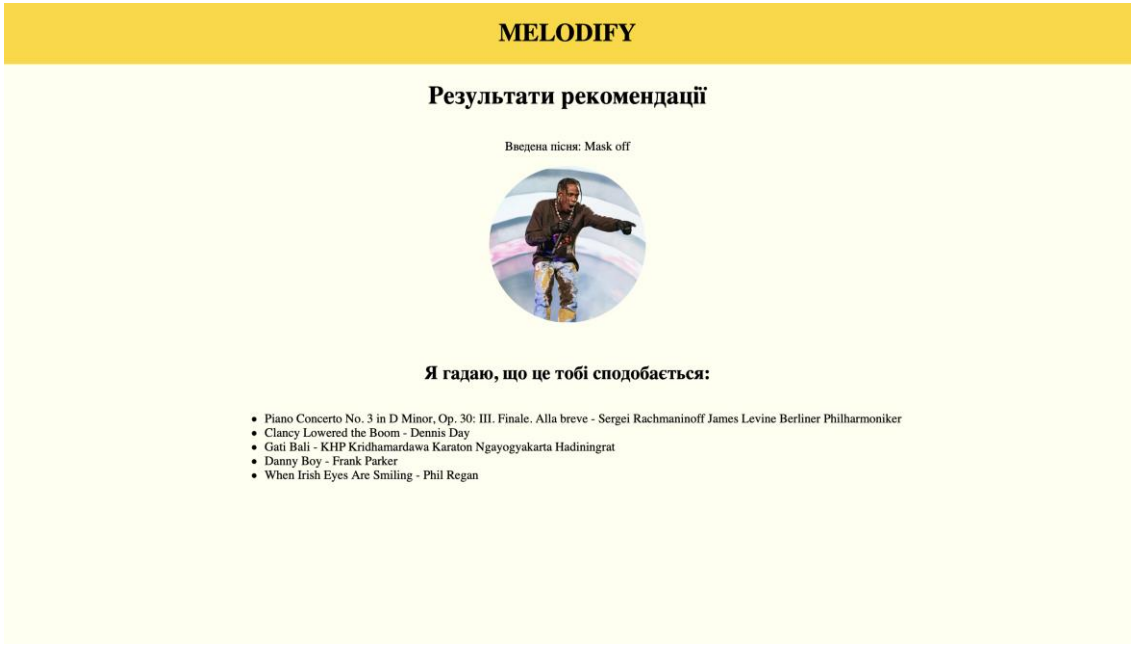

Рисунок 3.11 - Сторінка, де надається список пісень для рекомендацій

Таким чином, було розглянуто повний алгоритм , що має зробити користувач, щоб отримати рекомендації від розробленого програмного модуля

3.5 Тестування та аналіз результату роботи програмного модуля підбору музичних композицій

Як вже було зазначено, для даного дипломного проекту були обрані два метода розробрки рекоендаційної системи , такі як: метод k-means , метод колаборитивної фільтрації. У ході роботи в наборі даних були виявлені певні недоліки, усунувши які, можемо прослідкувати. Подившись на рисунку 3.8, можна зрозуміти, що метод K-means справлятися не так точно, але набагато швидше.

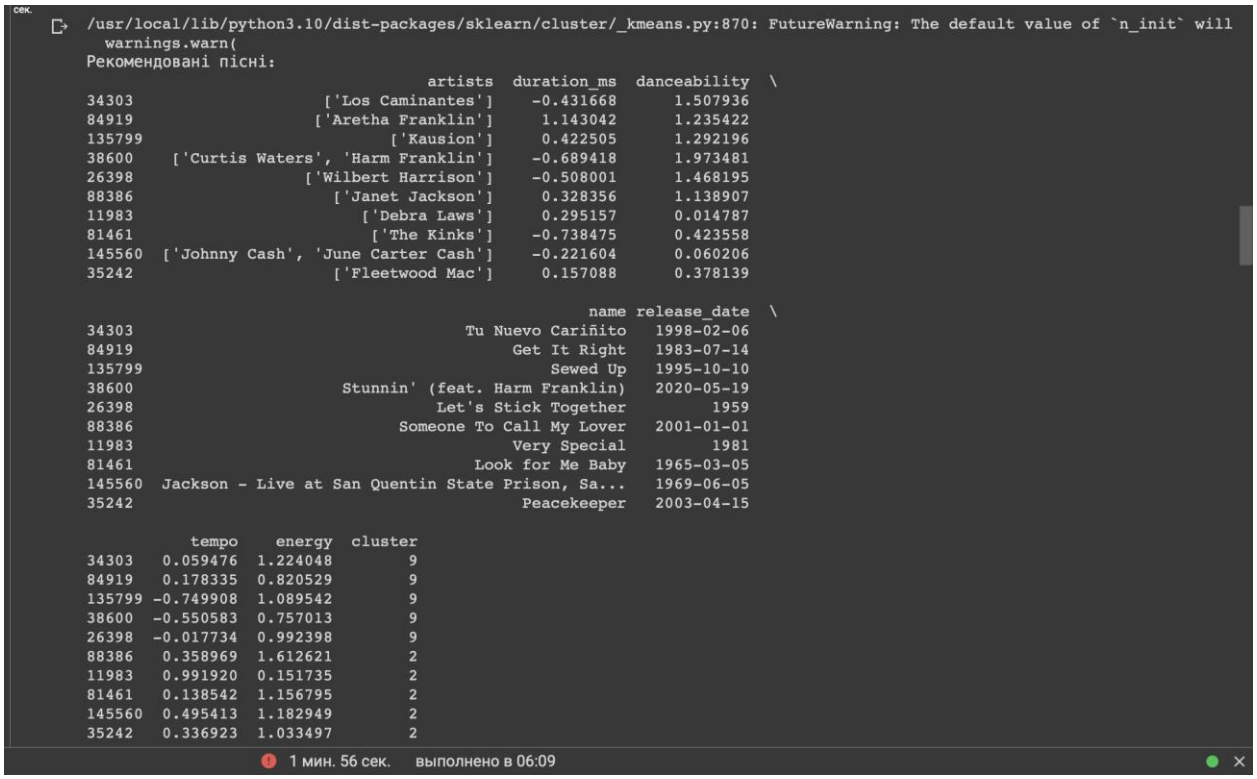

Рисунок 3.11 – Порівняння роботи двох алгоритмів

Вимірявши показники швидкості розрахунків кожного методу, можна зробити висновок, що найшвидшим виявився метод k-means. А також, аналізуючи показники точності прогнозування, найбільш точним виявився метод колаборативної фільтрації

Висновки розіл 3

У третьому розділі було описано програмну реалізацію програмного модулю.Було наведено опис інтерфейсу, які сторінки було розроблено, який функціонал доступний на кожній з сторінок.Проведено тестування системи на коректність різних дій користувача.

Також було описано дані, які використовуються для розробки програмного модуля та порівняння роботи двох методів, а саме на точність рекомендацій , швидкість надання користувачеві ТОП-N пісень, які йому можуть сподобатись.

ВИСНОВКИ

У ході виконання дипломної роботи була розроблений програмний модуль підбору музичних композицій, що базується на методах K-Means і KNN. Для забезпечення зручності та доступності користувачам, був реалізований веб-застосунок з використанням фреймворку Flask.

Перший метод, K-Means, використовує кластеризацію для групування музичних композицій за схожістю. Цей підхід дозволяє пропонувати користувачам композиції, які мають подібні характеристики та стиль, відповідно до їхніх вподобань.

Другий метод колаборативної фільтрації за допомогою KNN зазвичай використовується для основаної на спільних інтересах рекомендації. У даному програмному модулі використовується модель KNNBasic, яка реалізує просту версію KNN для рекомендацій. За допомогою цього методу виконується прогнозування оцінок користувачів для певної пісні шляхом знаходження найближчих сусідів за схожістю між піснями.Використовується "user_based=False", що означає, що прогнози робляться на основі схожості між піснями, а не між користувачами.

Для оцінювання ефективності розробленого програмного модуля були розглянуті та використані метрики точності прогнозування оцінки та релевантності наданих рекомендацій. Точність прогнозування оцінки визначається здатністю системи передбачати оцінку, що є важливим критерієм для формування підбірки композицій. Релевантність рекомендацій вимірюється у співвідношенні композицій, які користувач справді зацікавлений у підбірці.

Список використаних джерел

1. Снитюк В. Є. Прогнозування. Моделі. Методи. Алгоритми: Навчальний посібник / В. Є. Снитюк. — Київ: Маклаут, 2008. — 364 с.

2. Recommender Systems Handbook Francesco Ricci · Lior Rokach Bracha Shapira ·Paul B. Kanto

3. Trends, problems and solutions of recommender system, https://www.researchgate.net/publication/307862659_Trends_problems_and_solutions_of_recommender_system

4. Evaluation Metrics for Personalized Recommendation Systems. Shuhao Jiang and Jinlin Song 2021 J. Phys.: Conf. Ser. 1920 012109

5. https://www.python.org/doc/

6. https://www.kdnuggets.com

7. Yao, Y. Y. (1995) Measuring retrieval effectiveness based on user preference of documents. Journal of the American Society for Information science, 46(2): 133-145.

8. https://prjctrmag.com/ai-in-spotify

9. Reducing Data Sparsity in Recommender Systems, https://www.iasj.net/iasj/download/ca46a3f16fea9999

10. Deep learning for recommender systems: A Netflix case study – Harald Steck, Linas Baltrunas, Ehtsham Elahi, Dawen Liang, Yves Raimond, Justin Basilico.

11. https://pandas.pydata.org/

12. https://numpy.org/

13. https://scikit-learn.org/stable/

14. https://medium.com/recombee-blog/machine-learning-for-recommender-systems-part-1-algorithms-evaluation-and-cold-start-6f696683d0ed

15. Yu, D., Cheng, T., Yuan, X. (2020) Software Crowdsourcing Task Recommendation Algorithm. Based on Learning to Rank. Computer Science, 47(12): 106-113.

16. Jure Leskovec, Anand Rajaraman,Jeffrey D"Mining of Massive Datasets" (Видобування масивних наборів даних)

17. Reducing Data Sparsity in Recommender Systems, https://www.iasj.net/iasj/download/ca46a3f16fea9999

18. K- Nearest Neighbors (KNN) Algorithm: An Implementation Guide in Python, Medium: https://medium.com/machine-learning-with-python/k-nearest-neighbour-knn-implementation-in-python-498daa39c16e

19. http://om.univ.kiev.ua/users_upload/15/upload/file/pr_lecture_03.pdf

20. Yao, Y. Y. (1995) Measuring retrieval effectiveness based on user preference of documents. Journal of the American Society for Information science, 46(2): 133-145.

21. Benjamin S. Duran, Patrick L. Odell – Cluster Analysis

Додатки

Додаток А

Main.py

```
from flask import Flask, render_template, redirect, url_for, request

from kmeans import open_csv,results
#from TD_IDF import get_content_based_recommendations
from colab_filt import get_recommendations, data_music

import pandas as pd
from scipy.sparse import csr_matrix
from implicit.als import AlternatingLeastSquares

app = Flask(__name__)


@app.route('/')
def index():
    return render_template('index.html')

@app.route('/methods')
def methods():
    return render_template('methods.html')

@app.route('/next_page', methods=['GET', 'POST'])
def next_page():
    num_input_songs = 0
    num_recommendations = 0

    if request.method == 'POST':
```

```
            if 'song_count' in request.form and 'rec_count' in request.form:
                num_input_songs = int(request.form['song_count'])
                num_recommendations = int(request.form['rec_count'])
            return                                    render_template('next_page.html',
num_input_songs=num_input_songs,num_recommendations=num_recommendati
ons)
        return render_template('next_page.html')



    @app.route('/enter_song/<int:num_input_songs>&<int:num_recommendatio
ns>',methods=['GET','POST'] )
    def rec(num_input_songs,num_recommendations):
        input_songs = []
        if request.method == 'POST':
            for i in range(num_input_songs):
                song_name = request.form.get(f'song{i}')
                if song_name:
                    input_songs.append(song_name)
                # input_songs.append(request.form[f'song{i}'])
            return
render_template('next_page.html',num_input_songs=num_input_songs,input_song
s=input_songs,num_recommendations=num_recommendations)
        return render_template('next_page.html')



    @app.route('/result/<int:num_recommendations>&<path:input_songs>',
methods=['GET','POST'])
    def result( num_recommendations, input_songs):
        music_data = open_csv()
```

```
        input_songs_list = input_songs.split(',')  # Разделение строки на элементы массива по запятой
        recommended_songs = results(music_data,num_recommendations,input_songs_list)
        recommended_songs_list = recommended_songs.to_dict('records')

        return render_template('result.html',recommended_songs=recommended_songs_list)

    @app.route('/about')
    def about():
        return render_template('about.html')

    @app.route('/manual')
    def manual():
        return render_template('manual.html')

    ### Метод Колабортавної фільтрації
    @app.route('/col_fil', methods=['GET', 'POST'])
    def col_fil():
        if request.method == 'POST':
            song_name = request.form.get('song_name')
            return redirect(url_for('col_fil_result', song_name=song_name))
        return render_template('col_fil.html')

    @app.route('/col_fil_result', methods=['GET', 'POST'])
    def col_fil_result():
        if request.method == 'POST':
            song_name = request.form.get('song_name')
```

```
            recommendations = get_recommendations(song_name, num_recommendations=5)
            if recommendations:
                return render_template('col_fil_result.html', song_name=song_name, recommendations=recommendations)
            else:
                return render_template('col_fil_result.html', song_name=song_name, recommendations=[])
        else:
            return redirect(url_for('col_fil'))




    if __name__ == '__main__':
        app.run(debug=True)
```

K-Means.py

```
import pandas as pd
from sklearn.cluster import KMeans


def open_csv():
    # Завантаження даних із файлу CSV
    data = pd.read_csv('data_music.csv')

    # Вибір потрібних полів
    selected_features = ['artists', 'duration_ms', 'danceability', 'name', 'release_date', 'tempo', 'energy']
```

```
    music_data = data[selected_features]

    # Обробка пропущених значень (якщо необхідно)
    music_data = music_data.dropna()

    # Масштабування числових ознак (duration_ms, danceability, tempo, energy)
    music_data[['duration_ms', 'danceability', 'tempo', 'energy']] = \
        (music_data[['duration_ms', 'danceability', 'tempo', 'energy']] - music_data[
            ['duration_ms', 'danceability', 'tempo', 'energy']].mean()) / \
        music_data[['duration_ms', 'danceability', 'tempo', 'energy']].std()

    # Кількість кластерів
    num_clusters = 10

    # Застосування алгоритму K-Means
    kmeans = KMeans(n_clusters=num_clusters, random_state=42)
    kmeans.fit(music_data[['duration_ms', 'danceability', 'tempo', 'energy']])

    # Додавання міток кластерів у дані
    music_data['cluster'] = kmeans.labels_
    return music_data


def results(music_data,num_recommendations,input_songs):
    # Фільтрація даних за введеними піснями
    filtered_data = music_data[music_data['name'].isin(input_songs)]

    # Отримання унікальних міток кластерів для введених пісень
    user_clusters = filtered_data['cluster'].unique()
```

```
    # Створення порожнього DataFrame для рекомендацій
    recommended_songs = pd.DataFrame()

    # Вибір пісень із кластерів, у яких знаходяться введені пісні
    for cluster in user_clusters:
        cluster_songs = music_data[music_data['cluster'] == cluster].sample(num_recommendations)
        recommended_songs = pd.concat([recommended_songs, cluster_songs])

    return recommended_songs
```

colab_filt.py

```
import pandas as pd


data_music = pd.read_csv("data_music.csv", usecols=["artists", "duration_ms", "danceability", "name", "release_date"])

# Видаляємо дублікати записів за назвою пісні та переіндексовуємо
data_music = data_music.drop_duplicates(subset="name").reset_index(drop=True)

# Видаляємо зайві пробіли з назв пісень
data_music['name'] = data_music['name'].str.strip()

# Визначаємо клас KNNBasic, який буде використовуватись для моделювання методу KNN
class KNNBasic:
    def __init__(self, sim_options):
        self.sim_options = sim_options
```

```
    def fit(self, trainset):
        pass

    def predict(self, song_index1, song_index2):
        return Prediction(song_index2, 0, 0, 0)

# Клас Prediction використовується для зберігання результатів прогнозування
class Prediction:
    def __init__(self, iid, uid, est, details):
        self.iid = iid
        self.uid = uid
        self.est = est
        self.details = details

# Ініціалізуємо модель KNNBasic з вказаними параметрами sim_options
model = KNNBasic(sim_options={'user_based': False})

# Клас DataMusic використовується для зберігання даних про музику
class DataMusic:
    def __init__(self, data_music):
        self.data = data_music

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

    def raw_ratings(self):
        return [(i, i, 0) for i in range(len(self.data))]

    def to_inner_iid(self, song_index):
        return song_index
```

```
    def to_raw_iid(self, song_index):
        return song_index

# Створюємо об'єкт dataset класу DataMusic з використанням даних музики
dataset = DataMusic(data_music)

# Встановлюємо trainset як dataset
trainset = dataset

# Навчаємо модель на trainset
model.fit(trainset)

# Функція get_recommendations отримує назву пісні та кількість рекомендацій
def get_recommendations(song_name, num_recommendations):
    # Отримуємо індекс пісні в data_music, де назва пісні відповідає заданій назві
    song_index = data_music[data_music['name'].str.lower() == song_name.lower()].index[0]
    predictions = []
    for i in range(len(data_music)):
        # Ігноруємо пісню з тим самим індексом, що і задана пісня
        if i != song_index:
            # Виконуємо прогнозування за допомогою моделі для заданої пісні та інших пісень в data_music
            prediction = model.predict(song_index, i)
            # Отримуємо інформацію про пісню (назва та виконавці)
            song_info = (data_music.iloc[i]['name'], data_music.iloc[i]['artists'])
            # Додаємо результат прогнозування та інформацію про пісню до списку прогнозів
```

```
            predictions.append((song_info, prediction.est))
    # Сортуємо прогнози за оцінкою у зворотному порядку
    predictions.sort(key=lambda x: x[1], reverse=True)
    # Вибираємо топ-рекомендації та форматуємо відповідний вихідний формат
    top_recommendations = [(song_info[0], ' '.join(song_info[1].split(', ')).replace('[', '').replace(']', '').replace('"', '')) for song_info, _ in predictions[:num_recommendations]]
    return top_recommendations
```

top_charts.py:

```
import pandas as pd

# Загрузка данных из CSV-файла
data = pd.read_csv('data_music.csv')



# Функция для получения рекомендаций на основе популярности
def get_popularity_recommendations(num_recommendations):
    # Сортировка данных по популярности
    sorted_data = data.sort_values(by='popularity', ascending=False)

    # Получение рекомендаций
    recommendations = sorted_data[['name', 'artists', 'id']].head(num_recommendations)
    recommendations['spotify_link'] = 'https://open.spotify.com/search/' + recommendations['id'].astype(str)

    # Возвращение рекомендаций в виде списка словарей
    return recommendations.to_dict('records')
```

Додаток В

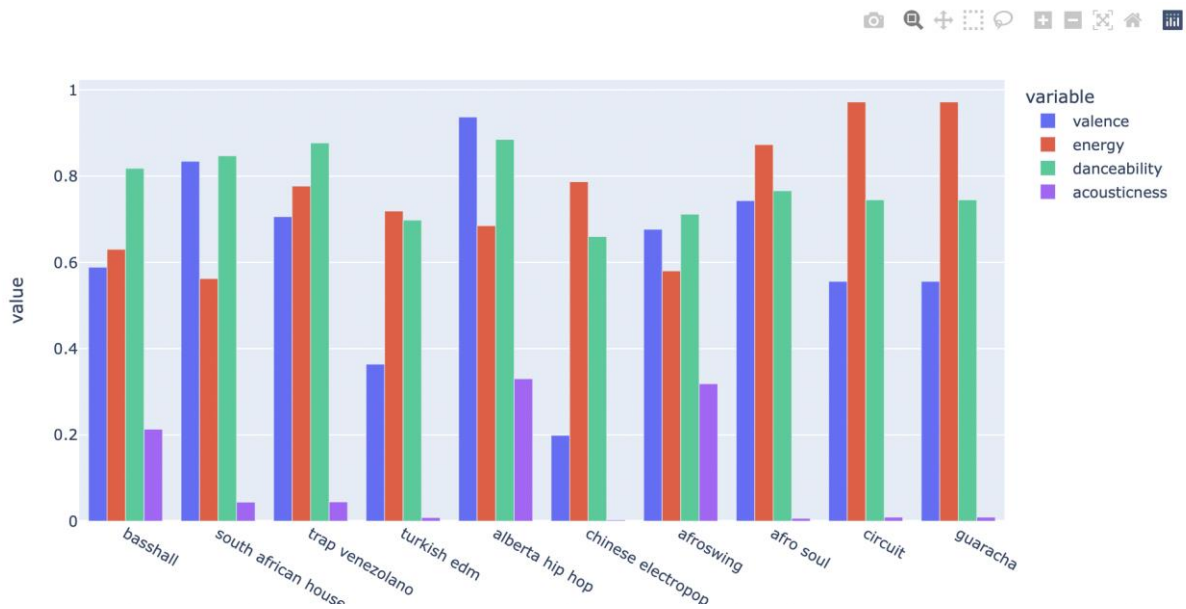

Рисунок 3 – Характеристика різних жанрів

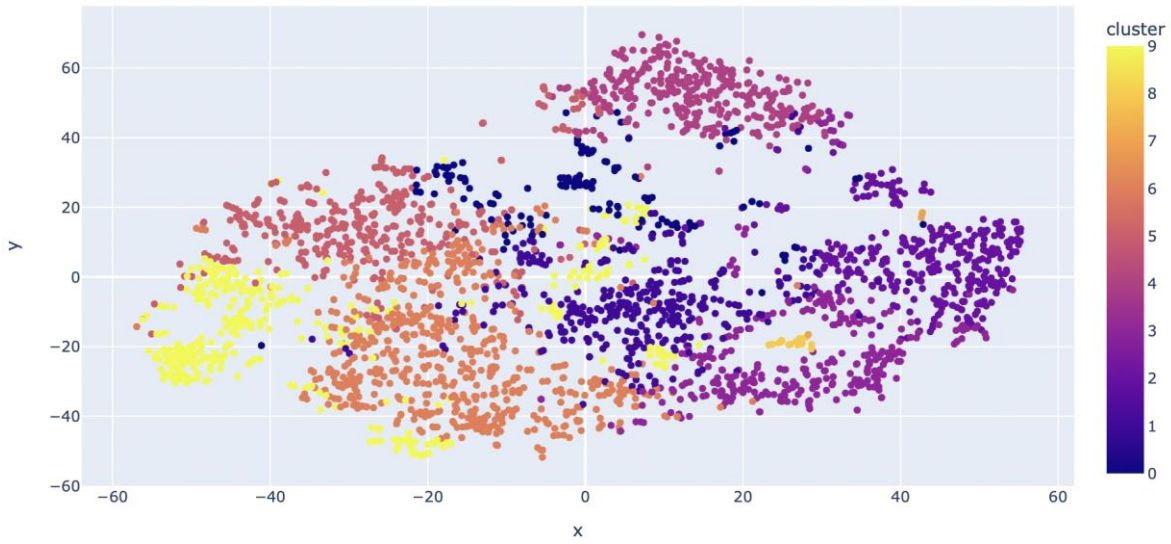

Рисунок 4 – Кластери K-Means

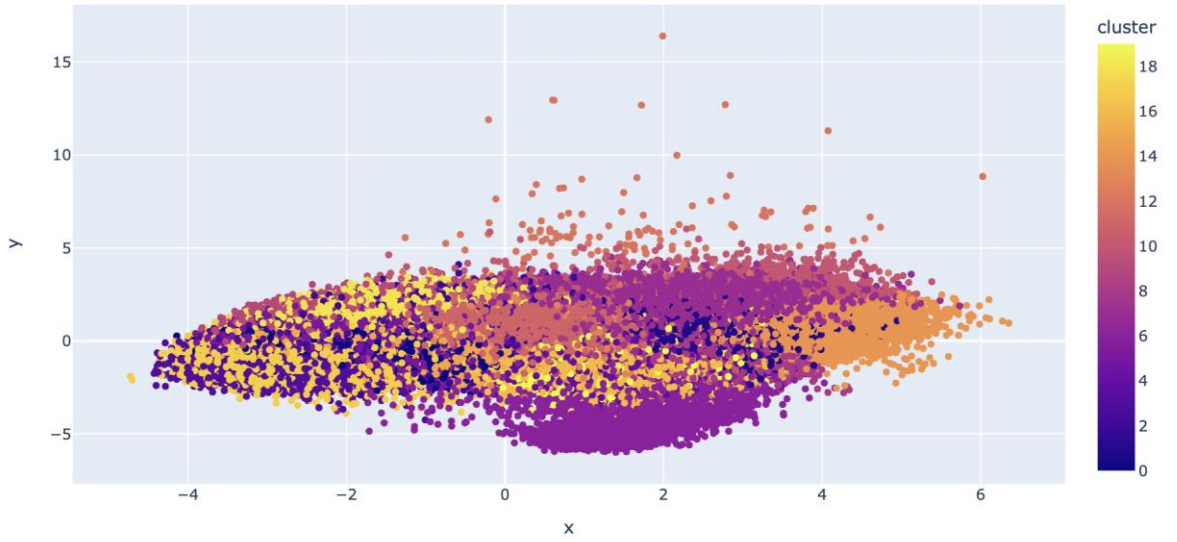

Рисунок 5 – Кластери пісень

```
print(data.isnull().sum())
```

```
valence             0
year                0
acousticness        0
artists             0
danceability        0
duration_ms         0
energy              0
explicit            0
id                  0
instrumentalness    0
key                 0
liveness            0
loudness            0
mode                0
name                0
popularity          0
release_date        0
speechiness         0
tempo               0
dtype: int64
```

Рисунок 6 – Перевірка першого датасету на пусті значення

```
:  1 print(year_data.isnull().sum())
   2
```

```
mode                0
year                0
acousticness        0
danceability        0
duration_ms         0
energy              0
instrumentalness    0
liveness            0
loudness            0
speechiness         0
tempo               0
valence             0
popularity          0
key                 0
dtype: int64
```

Рисунок 7 – Перевірка другого датасету на пусті значення

```
:  1 print(genre_data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2973 entries, 0 to 2972
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   mode              2973 non-null   int64  
 1   genres            2973 non-null   object 
 2   acousticness      2973 non-null   float64
 3   danceability      2973 non-null   float64
 4   duration_ms       2973 non-null   float64
 5   energy            2973 non-null   float64
 6   instrumentalness  2973 non-null   float64
 7   liveness          2973 non-null   float64
 8   loudness          2973 non-null   float64
 9   speechiness       2973 non-null   float64
 10  tempo             2973 non-null   float64
 11  valence           2973 non-null   float64
 12  popularity        2973 non-null   float64
 13  key               2973 non-null   int64  
dtypes: float64(11), int64(2), object(1)
memory usage: 325.3+ KB
None
```

Рисунок 8 – Перевірка датасету,вся інформація про колонки