Міністерство освіти і науки України

Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук

Кафедра теоретичної та прикладної системотехніки

«Затверджую»
Зав. кафедри теоретичної та
прикладної системотехніки
_____ д.т.н., проф. С. І. Шматков
«___» _____ 2023 р

Пояснювальна записка

до кваліфікаційної роботи магістра

на тему: **«Комп'ютерна модель персоналізованих рекомендацій вибору товарів на базі пошукових алгоритмів»**

Захищено на засіданні
Атестаційної комісії № 42
протокол № __ від __.12.2023 р.
Оцінка _____ / _____
Голова Атестаційної комісії
д.т.н., професор
_____СКОБ Ю.О.

Виконав:

студент групи КУ– 61
Галузь знань: 15 – Автоматизація та приладобудування
за спеціальністю 151 – Автоматизація та комп'ютерно-інтегровані технології
ЯСІНСЬКИЙ Ярослав Андрійович

Керівник:

д.т.н., с.н.с., професор кафедри теоретичної та прикладної системотехніки
ТОЛСТОЛУЗЬКА Олена Геннадіївна

Рецензент:

к.т.н., доц., виконувач обов'язків завідувача кафедри безпеки інформаційних систем і технологій
МЕЛКОЗЬОРОВА Ольга Михайлівна

Харків – 2023

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи складається із вступу, трьох розділів, висновків, списку використаних джерел та додатків. Загальний обсяг роботи складає 83 сторінки із яких 55 сторінок основної частини, 3 таблиці, 27 рисунків, 60 найменувань списку використаних джерел на 6 сторінках і 4 додатків на 15 сторінках.

Об'єкт дослідження: процес розробки комп'ютерних моделей на базі пошукових алгоритмів та алгоритмів кластеризації.

Предмет дослідження: комп'ютерні моделі персоналізованих рекомендацій вибору на базі пошукових алгоритмів та алгоритмів кластеризації.

Мета дослідження: метою є підвищення ефективності надання рекомендацій вибору товарів, за рахунок використання комп'ютерної моделі на базі пошукових алгоритмів та алгоритмів кластеризації.

Перший розділ присвячений аналізу пошукових алгоритмів, алгоритмів кластеризації та методів надання персоналізованих рекомендацій. Він охоплює використання алгоритмів кластеризації. Також досліджується аналіз існуючих механізмів рекомендацій. У другому розділі розглядаються обрані технології для серверної і Android-платформи, детально описуються їхні характеристики та відмінності, які роблять їх ефективними для даної системи. Визначаються вхідні та вихідні дані системи, що включають формування дата-сету товарів, необхідних для роботи системи. Надається огляд запропонованого рішення. Третій розділ присвячений тестуванню розробленої моделі. Включає оцінку ефективності алгоритмів кластеризації у системі персоналізованих рекомендацій, загальне тестування розробленої моделі та демонстрацію роботи програмного рішення.

Ключові слова: АЛГОРИТМИ КЛАСТЕРИЗАЦІЇ, ГІБРИДНА МОДЕЛЬ, РЕКОМЕНДАЦІЙНІ СИСТЕМИ, K-MEANS++, TF-IDF, ДАТАСЕТ, ANDROID ЗАСТОСУНОК.

ABSTRACT

The explanatory note to the qualification work consists of an introduction, three sections, conclusions, a list of used sources and appendices. The total volume of the work is 83 pages, of which 55 pages are the main part, 3 tables, 27 figures, 60 names of the list of used sources on 6 pages, and 4 appendices on 15 pages.

The object of the research: the process of developing computer models based on search algorithms and clustering algorithms.

The subject of the research: computer models of personalized recommendations of choice based on search algorithms and clustering algorithms.

The purpose of the research: the purpose is to increase the efficiency of providing recommendations for the product selection, due to the use of computer model based on search algorithms and clustering algorithms.

The first chapter is devoted to analyzing search algorithms, clustering algorithms, and methods of providing personalized recommendations. It covers the use of clustering algorithms. Analysis of existing recommendation mechanisms is also explored. The second section examines selected technologies for the server and Android platforms and describes their characteristics and differences that make them effective for this system. The input and output data of the system are determined, including the formation of a dataset of goods necessary for the operation of the system. An overview of the proposed solution is provided. The third section is devoted to testing the developed model. Includes evaluation of the effectiveness of clustering algorithms in the system of personalized recommendations, general testing of the developed model, and demonstration of the software solution.

Keywords: CLUSTERING ALGORITHMS, HYBRID MODEL, RECOMMENDATION SYSTEMS, K-MEANS++, TF-IDF, DATASET, ANDROID APPLICATION.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

KNN (англ. K-Nearest Neighbors) – це алгоритм машинного навчання, який використовується для класифікації об'єктів на основі їхніх ближніх сусідів. Основна ідея полягає в тому, щоб призначити клас новому об'єкту на основі класів його найближчих сусідів. У цьому методі "K" відноситься до кількості сусідів, які беруться до уваги під час класифікації.

HDBSCAN (англ. Hierarchical Density-Based Spatial Clustering of Applications with Noise) – це алгоритм кластеризації, який використовується для виявлення кластерів у даних з різною щільністю. Алгоритм працює шляхом побудови ієрархічного дерева кластеризації, де кожен вузол дерева представляє групу об'єктів з подібною щільністю.

TF-IDF (англ. Term frequency–inverse document frequency) – це статистичний показник, який використовується для оцінки важливості слова в контексті документа, що є частиною колекції документів або корпусу. Вага деякого слова пропорційна частоті вживання цього слова в документі і обернено пропорційна частоті вживання слова в усіх документах колекції.

CSV (англ. Comma-Separated Values) – це формат для збереження даних у вигляді таблиці, де кожен рядок відповідає рядку таблиці, а значення розділяються комами.

MVVM (англ. Model-View-ViewModel) – це архітектурний шаблон програмування, який використовується для розробки програмних додатків з інтерфейсом користувача.

REST (англ. Representational State Transfer) – це архітектурний стиль для розробки веб-сервісів, який використовує HTTP протокол, базується на ресурсах, має однорідний інтерфейс, що сприяє ефективній та масштабованій взаємодії між клієнтом і сервером.

ВСТУП

У сучасному світі, насиченому широким спектром товарів та послуг, питання надання персоналізованих рекомендацій для вибору стає актуальним завданням для багатьох сфер, зокрема електронної комерції та онлайн-платформ. Зростання обсягу інформації та розмаїття варіантів вибору споживачів ставлять під загрозу ефективність та зручність процесу вибору певного продукту. У сфері електронної комерції є важливо підтримувати цікавість та залученість користувачів для просування продукції.

Рекомендаційні системи, що працюють на основі пошукових алгоритмів та алгоритмів кластеризації, мають потенціал для значного покращення користувацького досвіду, пропонуючи релевантні та персоналізовані пропозиції товарів. Існуючі компанії та підприємства постійно шукають інноваційні способи для залучення та утримування клієнтів. Ефективна система рекомендацій може сприяти підвищенню конкурентоспроможності платформи, збільшуючи продажі завдяки персоналізованим пропозиціям.

Одним із ключових викликів у сфері персоналізації є розробка ефективних методів надання рекомендацій. У створеному дослідженні розглядаються обраний пошуковий алгоритм та різноманітні методи кластеризації, зокрема на основі центроїдів, ієрархічна кластеризація, а також методи на базі щільності та розподілу. Крім того, важливо розглянути існуючі механізми персоналізації, такі як моделі фільтрації на основі контенту та колаборативної фільтрації. Аналіз цих моделей дозволить визначити їхні переваги та обмеження, що сприятиме вибору ефективного методу для дослідження.

Аналіз проблем, пов'язаних з алгоритмами кластеризації в системах рекомендацій продуктів, сприяє прийняттю обґрунтованих рішень. Виявлення проблем і недоліків таких систем дозволяє вдосконалювати алгоритми, що призводить до більш точних прогнозів і та збільшення продажів компаній.

З розвитком технологій, зростають і можливості рекомендаційних систем. Вивчення описаних алгоритмів в контексті рекомендацій продуктів відповідає технологічному прогресу в галузі машинного навчання, гарантуючи, що рекомендації залишатимуться релевантними та ефективними.

Тема кваліфікаційної роботи, присвячена аналізу можливості покращення систем рекомендацій товарів на базі пошукових алгоритмів та кластеризації, є актуальною, оскільки вона може сприяти поліпшенню якості та швидкості надання відповідних рекомендацій заснованих на пошуку користувача.

Даній роботі буде проведено дослідження ефективності різних типів кластеризації та створено гібридну модель надання персоналізованих рекомендацій для вибору товару користувачем за допомогою поєднання алгоритму обраного алгоритму кластеризації та пошукового алгоритму на базі мобільного додатку Android.

Усе це дозволить тестувати та обґрунтувати запропоноване рішення. Дослідження такого типу важливе для розробки високоефективних систем рекомендацій, що відповідають унікальним потребам користувачів.

РОЗДІЛ 1.
АНАЛІЗ АЛГОРИТМІВ ТА МЕТОДІВ НАДАННЯ РЕКОМЕНДАЦІЙ

1.1 Використання пошукових алгоритмів для надання рекомендацій

Алгоритми пошуку – це алгоритми, що використовуються для пошуку певних елементів у датасетах. Загальні алгоритми пошуку включають двійковий пошук, інтерполяційний пошук та лінійний пошук [1].

Алгоритми пошуку дозволяють краще розуміти наміри користувачів і визначенні шаблонів у їхніх пошукових запитах, їх можна використати для надання списку рекомендацій шляхом ідентифікації елементів, схожих на ті, якими висловив інтерес користувач. Це можна зробити шляхом аналізу минулої поведінки користувача, наприклад його історії пошуку, історії покупок або звичок перегляду. Потім алгоритм пошуку використовує цю інформацію, щоб знайти інші елементи, які мають подібні характеристики. Наприклад, якщо користувач дивився багато відео про те як програмування, обраний алгоритм може порекомендувати інші відео з порадами про програмування [2].

До переваг даних алгоритмів можемо віднести декілька важливих характеристик. Вони відрізняються швидким пошуком інформації у великих наборах даних, незалежно від організації даних. Їхня точність виявляється завдяки отриманню релевантних даних навіть із неоднозначних або погано оформлених запитів користувачів. Ці алгоритми демонструють масштабованість, плавно адаптуючись до обробки наборів даних, що з часом розширюються. Крім того, їх універсальність дозволяє отримувати різноманітні типи даних, наприклад текст, зображення та відео [3-4].

Однак, цей тип алгоритмів має недоліки. Їхня складність часто створює труднощі для розуміння та реалізації, що робить обслуговування складним завданням. Крім того, ці алгоритми можуть вимагати значних обчислювальних

ресурсів, особливо під час обробки великих наборів даних. Також, їхня здатність повертати найбільш релевантну інформацію може погіршитися, особливо з неоднозначними або погано сформульованими запитами [5].

1.1.1 Алгоритми неінформованого пошуку

Алгоритми неінформованого пошуку, які також називають алгоритмами сліпого пошуку, працюють без будь-якої попередньої інформації про розташування цільового елемента. Ці алгоритми систематично переміщаються в просторі пошуку без вказівок із самих даних. Поширені неінформовані алгоритми пошуку включають пошук у ширину, який перетинає простір пошуку рівень за рівнем, починаючи з кореневого вузла та розширюючи всі його сусіди перед переходом на наступний рівень. Пошук у глибину – заглиблюється якомога глибше в кожну гілку, перш ніж повертатися до інших гілок. Інший вид – пошук за рівномірною вартістю, що розширює вузол із найменшою накопиченою вартістю, враховуючи кількість кроків, зроблених для досягнення цього вузла [6-7].

Описаний тип пошуку має мають явні переваги, наприклад простота в реалізації та гарантування знаходження рішення, якщо воно існує в скінченному просторі пошуку, однак є певні недоліки, такі як використання значного об’єму пам’яті, особливо в сценаріях із великим простором пошуку. У випадку з відсутності евристичних вказівок алгоритми неінформованого пошуку можуть досліджувати численні непотрібні стани під час роботи.

Розглянемо сценарій, коли користувач намагаєтеся знайти обраний товар у магазині. З початку, розпочинається огляд на першої полиці на першому стенді, досліджуючи маркування товарів, поки не знайдете заданий. Якщо товар відсутній на першому стенді – перехід до розгляду другого, процес повторюється до визначення заданого. Це систематичне дослідження магазину, рівень за рівнем, так працює алгоритм BFS (англ. Breadth-first search) – що є пошуком вшир, оскільки він не має попередньої інформації про

положення товару. BFS досліджує простір пошуку рівень за рівнем, починаючи з кореневого вузла та розгортаючи всі вузли на заданій глибині перед переходом до наступної [8].

1.1.2 Алгоритми інформованого пошуку

Інформовані алгоритми пошуку, які ще мають назву евристичних алгоритмів пошуку, базуються на інформації про місце розташування елемента, що шукають, завдяки чому вони ефективніше орієнтуються в порівнянні з іншим типом алгоритму. На відміну від алгоритмів неінформованого пошуку, які сліпо досліджують простір пошуку, алгоритми інформованого пошуку використовують евристики, які є оцінками відстані між вузлом і цільовим вузлом для конкретного елементу. Завдяки впровадженню евристики в процес пошуку обґрунтовані алгоритми пошуку можуть визначати пріоритети вузлів, які з більшою ймовірністю приведуть до коректних рекомендацій товарів, що сприяє більш ефективному пошуку шляху для надання рекомендацій [9].

Алгоритми інформованого пошуку мають певні обмеження. По-перше, евристика має бути допустимою, гарантуючи, що вона ніколи не переоцінює фактичну вартість досягнення мети. Якщо евристика не має інформації, вона не керуватиме процесом пошуку, зменшуючи ефективність алгоритму. По-перше, повнота обґрунтованих алгоритмів пошуку не гарантується. Коли евристика є або неприйнятною, або неінформованою, ці алгоритми можуть не знайти рішення, навіть якщо воно існує [10].

GBFS (англ. Greedy Best-First Search) — це жадібний алгоритм пошуку, який прагне знайти рішення проблеми шляхом визначення пріоритетів вузлів, які здаються ближчими до мети. Він використовує евристичну функцію *f(x)*, яка оцінює вартість або відстань від певного вузла *x* до цільового вузла. На кожному кроці GBFS розширює вузол із найнижчим значенням функції у просторі пошуку, припускаючи, що вузол, найближчий до цільового вузла,

приведе до ефективного рішення. Описаний алгоритм є ефективним у випадках, коли евристична функція є точною, однак він не гарантує, що він знайде оптимальне рішення, оскільки він може застрягти в локальних мінімумах, де знаходить неоптимальне рішення і не може дослідити альтернативні шляхи, які могли б привести до заданого елементу [11].

1.2 Використання алгоритмів кластеризації для надання рекомендацій

Алгоритми кластеризації – це група методів машинного навчання, які використовуються для поєднання схожих елементів або окремих профілів користувачів у кластери, таким чином дозволяючи розпізнати шаблони та переваги у наборах даних [12].

Вони є гарним рішенням для вирішення проблем персоналізованих рекомендацій вибору, оскільки їх можна використовувати для групування товарів та створення списків подібних елементів на основі опису уподобань, чи аналізу поведінки вибору. На далі, отриману інформацію можна використовувати для надання переліку рекомендацій кожному користувачеві на базуючись на відповідному кластері, до якого вони належать.

Одними з ключових переваг використання алгоритмів кластеризації цієї моделі є можливість прогнозувати схожість елементів в залежності від відповідності до певної характеристики, завдяки чому можливо реалізувати ефективний пошук товарів за характеристиками. Це критично важливо для систем рекомендацій, тому що це дозволяє знайти користувачів, які мають подібні потреби, або знайти товари, що мають збіг побажаннями користувача. За допомогою кластеризації користувачів рекомендації можна адаптувати до вподобань конкретної групи, до якої належить користувач.

В наслідок, чого з’являється можливість є сегментувати базу користувачів на окремі підгрупи, що можуть представляти різні сегменти ринку, групи за вподобаннями, цільову аудиторію певних товарів. Проводячи

аналіз кожного кластера, можна генерувати персоналізовані рекомендації для користувачів, що належать до цих сегментів.

Кластеризація дозволяє визначити товари, які є знаходяться поруч в обраному просторі ознак, оскільки вони мають подібний опис та характеристики, та теоретично можуть бути цікавими для користувача. у випадку сформованих кластерів, алгоритм дозволяє оновлювати рекомендації, однак для цього треба мати повний датасет, без пустих полів, щоб забезпечити точність розрахунків. При включенні нових товарів, алгоритм може призначити їх до відповідного кластера, що допоможе підтримувати рівень актуальності рекомендацій.

1.2.1 Кластеризація на основі центроїдів

Розглянемо модель з використанням центроїда, для побудови кластерів. Така модель є ґрунтовною технікою для проведення аналізу даних і неконтрольованого машинного навчання. Такий тип може використовується для проведення групування подібних точок властивостей товарів у кластери шляхом встановлення репрезентативних точок, що називаються центроїдами. Достатньо відомим алгоритмом на основі центроїда є K-means, що працює шляхом виділення подібних точок даних. Наприклад, користувачів або елементів у кластери, що представлені центроїдами. До переваг такого підходу відноситься створення інтерпретованих кластерів, що дозволяє створювати інтерпретовані фрагменти з бази даних предметів або користувачів, сприяючи персоналізованим рекомендаціям шляхом групування подібних об’єктів. Слід зазначити, що описаний алгоритм можна використовувати для підвищення ефективності систем рекомендацій. Наприклад, можливе використання K-means для попереднього фільтрування набору елементів, які розглядаються для рекомендації кожному користувачеві. Головним недоліком цього алгоритму у персоналізованих системах рекомендацій є чутливість до стартового вибору центроїдів. Якщо початкові центроїди не вибрані ретельно,

алгоритм може сходитися до локального оптимуму, що призведе до неоптимальної кластеризації [13-14].

Існує покращена версія алгоритму, що має назву K-means++ – це варіант звичайного K-means, що усуває його основний недолік, тобто чутливість до початкових умов. Головне покращення – модифікація вибору перших центроїд кластера, що зменшує ймовірність збіжності алгоритму до локального оптимуму. Для обрання стартових центроїді, алгоритм K-means++ проводить обрання першого центроїду на базі однієї точки з датасету, що обрана випадково. Наступний крок, обрання наступного центроїда з точки датасету, що є найдальшою від усіх існуючих центроїдів. Цей процес повторюється ітеративно, доки не буде обрано певну кількість заданих центроїдів. Отже, вибираючи стартові центроїди кластера описаним способом, алгоритм K-means++ буде з меншою ймовірністю зіходити до локального оптимуму, що призведе до більш точних результатів кластеризації [15].

1.2.2 Ієрархічна кластеризація

Ієрархічна кластеризація базується на підході аналізу елементів, що більш пов’язані з елементами поряд, а ніж з тими, що розташовані далі. Цей тип алгоритмів створює деревоподібні структури, з метою генерації кластерів. Основним підходом для створення кластера є підбір елементів зі структури, за відстанню, тоді створений фрагмент даних характеризується максимальною відстанню, для групування частин кластера.

Моделі подібного типу краще працюють на даних малого об’єму. Для забезпечення ефективної роботи на великих датасетах, необхідно підвищення обчислювальних ресурсів.

Ієрархічна кластеризація – це вид алгоритму неконтрольованого навчання, що поєднує елементи датасету в ієрархію, з якої формуються кластери. Ієрархію можна візуалізувати у вигляді дерева, де кожен елемент є

частиною гілки, яка в свою чергу є кластером. Кореневий кластер дерева представляє весь датасет.

Цікавим прикладом є алгоритм HDBSCAN, що ієрархічним алгоритм кластеризації, який особливо добре підходить для персоналізованих систем рекомендацій. Цей алгоритм кластеризації на базі щільності, що означає, що він групує точки датасету разом на основі їх щільності в просторі даних. Це робить HDBSCAN більш стійким до викидів і шуму, ніж інші алгоритми кластеризації [16].

Щоб використовувати HDBSCAN для персоналізованих систем рекомендацій, першим кроком є об’єднання користувачів у групи зі схожими перевагами, що можна зробити шляхом кластеризації користувачів на основі їхніх попередніх оцінок елементів, таких як фільми, книги чи продукти. Після об’єднання користувачів у кластери наступним кроком є створення персоналізованих рекомендацій для кожного користувача. Це можна зробити, порекомендувавши елементи, популярні серед користувачів у кластері користувачів.

У ієрархічній кластеризації моделі структура може бути організована у формі дерева або інших ієрархічних структур, де кореневі вузли координують дії свої відгалужені вузли. Це спрощує керування та масштабування обчислювальних ресурсів, а також дозволяє забезпечити високу доступність і надійність системи [17].

1.2.3 Кластеризація на основі щільності

Кластеризація на базі щільності відноситься до неконтрольованих методів навчання, які ідентифікують відмінні групи елементів в даних на основі ідеї, що кластер у просторі даних є безперервною областю високої щільності, відокремленої від інших таких кластерів суміжними областями низької точки щільності. Точки даних у роздільних областях із низькою

щільністю точок зазвичай вважаються шумом чи викидами. непараметричний підхід, коли групи в даних вважаються областями високої щільності.

Підходи кластеризації на основі щільності не вимагають кількості кластерів як вхідних параметрів, а також не роблять припущень щодо основної щільності чи дисперсії в межах груп, які можуть існувати в даних. Отже, кластери побудовані на даному підході, не обов'язково є групами елементів із високою внутрішньо-кластерною подібністю, що можна виміряти функцією відстані, але можуть мати нечітку форму в просторі характеристик, також вони ще мають назву «природних кластерів» [18].

Ця властивість робить цей тип гарним вибором для систем, де кластери не можуть бути добре охарактеризовані, як окремі категорії з низькою схожістю всередині кластера. Наприклад, у просторових даних, де групи точок у просторі можуть бути побудовані вздовж структур, таких як дельти рік, розгалуження доріг.

Розглянемо цей тип кластеризації на прикладі, алгоритму Mean shift. Перший крок, передбачає оцінку базової функції щільності ймовірності точок даних. Зазвичай це робиться за допомогою оцінки щільності ядра, де кожна точка даних представлена функцією ядра з центром у цій точці. Функція ядра визначає вагу, призначену кожній точці даних у процесі оцінки щільності. Наступний крок, алгоритм ітеративно переміщує точки даних до областей з вищою щільністю на основі векторів середнього зсуву, обчислених як зважене середнє значення відмінностей між кожною точкою та її сусідами. Ітерації тривають до конвергенції, що вказується векторами мінімального середнього зсуву. Кінцеве положення кожної точки даних позначає центр кластера, що дозволяє призначити найближчий центр і ідентифікувати окремі кластери в даних [19].

На відміну від добре відомого підходу кластеризації K-means, Mean shift не потребує припущень щодо кількості кластерів і форми розподілу, але його продуктивність залежить від вибору параметрів масштабу. Пропускна здатність є єдиним параметром для налаштування, тому для одновимірного

випадку це відносно проста процедура, але в багатовимірному випадку це може бути важко [20].

Процедура Mean shift складається з двох етапів:

1. Побудова щільності ймовірності в деякому просторі ознак,

2. Відображення кожної точки на найближчий до неї максимум (моду) щільності.

Кожна точка даних зсувається до середньозваженого набору даних. Алгоритм середнього зсуву намагається знайти стаціонарні точки оціненої функції щільності ймовірності.

1.2.4 Кластеризація на основі розподілу

Кластеризація на базі розподілу – це модель кластеризації, у якій підлаштовуються дані щодо ймовірності належності до того самого розподілу. Цей підхід до кластеризації припускає, що дані складаються з розподілів, таких як гаусівський, біноміальний тощо. Результати гаусівського розподілу можна помітити, коли є фіксована кількість розподілів і всі наступні дані вписуються в нього таким чином, щоб розподіл даних міг бути максимальним Моделі розподілу кластеризації пов’язані зі способом, у який набори даних створюються, що впорядковуються на основі принципів випадкової вибірки, для отримання точок даних з однієї форми розподілу. Тоді згенеровані кластери визначаються як об’єкти, що з великою ймовірністю, належать до одного розподілу [21-22].

Наприклад, алгоритм очікування-максимізації є одним з прикладів кластеризації на основі розподілу, що використовує багатовимірні розподіли, та має значну перевагу з огляду гнучкості алгоритму в порівнянні з такими методами кластеризації, як на базі центроїда та щільності. Це можливо пояснити тим, що описані алгоритми є залежними від форми кластерів, яка регулюється декількома гіперпараметрами, і якщо було некоректно визначено їх значення, вони можуть надати небажані та помилкові результати. Помітним

недоліком є неможливість використання його у випадках, коли дано інформацію про тип розподілу інформації в датасеті. Отже, прикладом реалізації цього підходу є моделі GMM (англ. Gaussian mixture models). Такі моделі є видом алгоритму машинного навчання, що можна використати для кластеризації даних за різними характеристиками на основі розподілу ймовірностей. Цей алгоритм відносно стійкий до викидів та шуму, однак, ще може давати точні результати, навіть якщо присутні певні елементи, які не підходять до жодного із кластерів. GMM має багато застосувань, таких як оцінка щільності, кластеризація та сегментація зображень. Для кластеризації можна використати результат робити алгоритму для групування точок даних, які походять з того самого розподілу Гауса [23].

1.3 Аналіз існуючих механізмів надання персоналізованих рекомендацій

Персоналізація — це процес пристосування рішення або надання переваги відповідно до вподобань, характеристик або конкретних потреб людини. Це процес налаштування та коригування вибору таким чином, щоб відображати переваги та вимоги кожної людини, сприяючи більш індивідуальному досвіду. Цей підхід має на меті підвищити задоволеність і залученість користувачів, враховуючи особисті смаки, поведінку та обставини під час представлення варіантів або надання рекомендацій. Система рекомендацій – це програмний додаток, який використовує алгоритми та дані, щоб передбачити інтереси користувачів і рекомендувати елементи, які можуть їх зацікавити [24].

З точки зору маркетингу, персоналізація використовується для створення цільових рекламних кампаній, які з більшою ймовірністю привернуть увагу окремих споживачів. Наприклад, компанія може використовувати надання персоналізованих рекомендацій, щоб показувати різні оголошення різним людям залежно від їх віку, статі чи інтересів.

Основним видом використання систем персоналізації слугують інформаційні системи, які пов’язані використанням та взаємодією з контентом користувачами. На прикладі платформи YouTube, можна побачити, що важливе значення відіграє час перегляду відео та безпосередня взаємодія з ним у формі коментування або відзнаки подобається, щоб ті більш стали рекомендованими [25].

Окрім взаємодії користувач може отримати персоналізовані рекомендації завдяки механізмам аналізу контенту. Розглянемо на прикладі додатку Spotify, який використовує багато підходів для вирішення цієї задачі. Одним з яких, є контентна фільтрація, що базується на аналізу характеристик товарів, такі як категорії, ключові слова, опис та інші. На цих метаданих формується профіль користувача для якого формуються відповідні підбірки музичних виконавців чи композицій [26].

1.3.1 Модель фільтрації на основі контенту

Фільтрація на основі контенту – це метод для надання персоналізованих рекомендацій, що враховує характеристики самого об’єкта, на відміну від колаборативної фільтрації, де використовується схожість між користувачами або предметами на основі історії взаємодії [27].

Щоб почати процес надання рекомендацій, користувач повинен здійснити пошук чи переглянути кілька товарів, після чого система зможе почати надавати відповідні рекомендації. В основі рекомендацій на основі вмісту лежить основна концепція стабільних уподобань. Ці системи працюють за припущення, що налаштування користувача залишаються відносно схожими з часом. На відміну від методів спільної фільтрації, які покладаються на взаємодію та поведінку користувачів, рекомендації на основі вмісту залежать від атрибутів – характеристик, які визначають елемент, таких як його жанр, стиль або особливості – пов’язаних із елементами. Цей підхід особливо цінний, коли історія взаємодії користувача обмежена або недоступна [28].

Атрибути предмета можна розділити на два основні типи: об’єктивні атрибути та суб’єктивні атрибути. Розглянемо, на прикладі додатку для прослуховування музики. Перше – це фактичні характеристики, властиві самому елементу, такі як жанр, автор, дата публікації, ключові слова та технічні характеристики. Друге – це більш абстрактні властивості, які відображають сприйняті характеристики предмета, такі як настрій, тон, стиль, теги, створені користувачем.

Системи рекомендації на базі контенту використовують атрибути та характеристики товарів, щоб побудувати вектор з характеристик для персоналізації. Коли користувачі виявляють свої вподобання під час взаємодії з елементами, відбувається оновлення вектору кожного користувача окремо. Для визначення переваги можна використати косинусну відстань між векторами користувача та елемента [29].

Загальна формула виглядає так:

$$P(m,n) = \frac{m \cdot n}{\|m\| \cdot \|n\|} \tag{1.1}$$

де $P(m, n)$ – подібність вектору m та вектору n;
m, n – вектори інтересів.

Розберемося на прикладі: цільовий користувач хоче обрати легковий автомобіль, йому подобається тип кузову пікап, а тип седан або позашляховик має низький пріоритет. Вектор для пікапу має додатні значення, а вектор для седану має від’ємне значення для цього конкретного користувача. Уявімо, що до системи було добавлено новий тип – пікап-позашляховик. Оскільки наш користувач віддає перевагу пікапам, косинусний кут між вектором типу кузову та вектором користувача буде мати велику додатною частку, що призведе до меншого кута, що означає, що це хороша рекомендація для обраного користувача. Якщо косинусна відстань велика, необхідно ігнорувати елемент, оскільки це погана рекомендація.

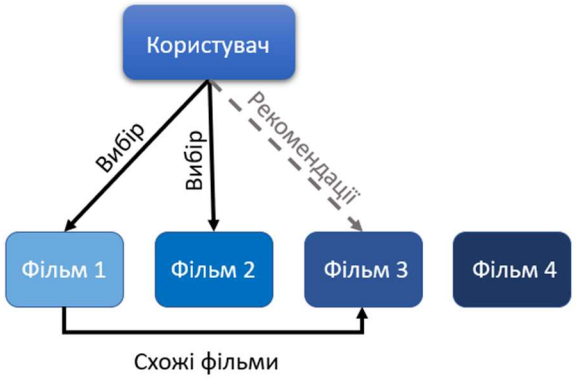

Рисунок 1.1 – Рекомендована система фільтрації на основі контенту.

На рис. 1.1 зображено приклад, де модель автоматично пропонує третій елемент, а не четвертий, оскільки він більше схожий на перші два. Цю подібність можна обчислити на основі ряду ознак, таких як актори у фільмі, тривалість фільму тощо.

Переваги:

Найновіші елементи можна запропонувати відразу після їх запуску, не чекаючи оновлення моделі. Оскільки рекомендації базуються на повсякденній діяльності користувача, усі вподобання та параметри пропозицій точно налаштовані відповідно до вибору користувача. Таку систему можливо відносно легко масштабувати для великої кількості клієнтів, оскільки потрібні тільки дані від конкретного користувача окремо, щоб створити список персоналізованих рекомендацій.

Недоліки:

Якщо користувач раніше віддавав перевагу певним типам контенту, система може продовжувати рекомендувати подібні елементи, потенційно сприяючи ізоляції переліку рекомендацій від інших елементів, знижуючи різноманітність, як результат виявити нові товари, що можуть зацікавити користувача, неможливо. Уподобання можуть змінюватися з плином часу, однак система повільно адаптується до цих змін. У випадку, якщо інтереси користувача продовжують змінюватися, чи переглядає нові елементи, система може не відразу відобразити ці зміни без тривалого накопичення взаємодій

користувача з обраними елементами. Також, якщо користувач є неактивним, рідко обирає елементи, то система може надавати менш точні рекомендації [30].

1.3.2 Модель на основі колаборативної фільтрації

Колаборативна фільтрація — це комплекс методів в персоналізованих системах рекомендацій, що використовує колективні переваги та поведінку групи користувачів, щоб передбачити та запропонувати елементи, які цікавлять окремого користувача. Цей підхід ґрунтується на ідеї, що група профілів користувачів відображає поведінку взаємодій з елементами у минулому, отже, певна група буде мати схожі рекомендації через певний проміжок часу, таким чином підвищуючи релевантність і точність наданих рекомендацій.

Загальний алгоритм роботи виглядає так:

1. Користувач виражає свої переваги, оцінюючи елементи (наприклад, книги, фільми або музичні записи) системи. Ці рейтинги можна розглядати як приблизне відображення інтересу користувача до відповідного домену.

2. Система порівнює оцінки цього користувача з оцінками інших користувачів і знаходить людей із найбільш подібними смаками.

3. Для схожих користувачів система рекомендує елементи, які схожі користувачі оцінили високо, але ще не оцінені цим користувачем (імовірно, відсутність оцінки часто вважається незнайомістю товару)

Існує декілька підходів для побудови моделей на базі колабративної фільтрації. Розглянемо варіант на базі на пам'яті, що заснований на даних рейтингу користувачів для обчислення схожості між елементами чи користувачами.

Стандартний метод спільної фільтрації, який можна використати для роботи – KNN. Моделі на основі розрізняють за двома типами на основі користувача та на основі елементів. Розглянемо модель на прикладі:

$$Q_{mn} = \frac{\sum_k sim\,(K_m, K_k) q_{kn}}{N} \quad (1.2)$$

Маємо матрицю оцінок (*i*, *j*) з користувачем K_m, *m* дорівнює від одного до *i* та товар T_j, *n* дорівнює від одного до *j*, де *N* кількість оцінок. Тепер необхідно спрогнозувати рейтинг Q_{mn}, якщо цільовий користувач *m* не взаємодіяв з товаром *n*. Процес полягає в тому, щоб обчислити схожість між цільовим користувачем *m* та всіма іншими користувачами, вибрати найкращих *k* подібних користувачів і взяти середньозважене значення оцінок цих користувачів *k* зі схожістю, як вагові коефіцієнти [31].

Хоча різні люди можуть мати різні базові показники при оцінюванні, деякі люди зазвичай ставлять високі бали, деякі є необ'єктивними, навіть якщо вони задоволені пунктами.

Щоб уникнути цього упередження, можна відняти середню оцінку кожного користувача для всіх елементів під час обчислення зваженого середнього значення та додати її для цільового користувача.

Алгоритм рекомендацій на базі користувача використовує векторну модель на основі подібності, для визначення обмеженого перелік найбільш подібних за профілем користувачів до обраного активного користувача, після того, користувацькі матриці елементів користувача поєднуються, з метою отримання нового набору елементів, що будуть рекомендовані. Отже, щоб запропонувати новий список рекомендованих товарів для обраного користувача, створюється група схожих користувачів на основі взаємодії цільового користувача.

Для пропозицій використовуються елементи, які є найпопулярнішими в цій групі, але новими для цільового користувача (рис.1.2).

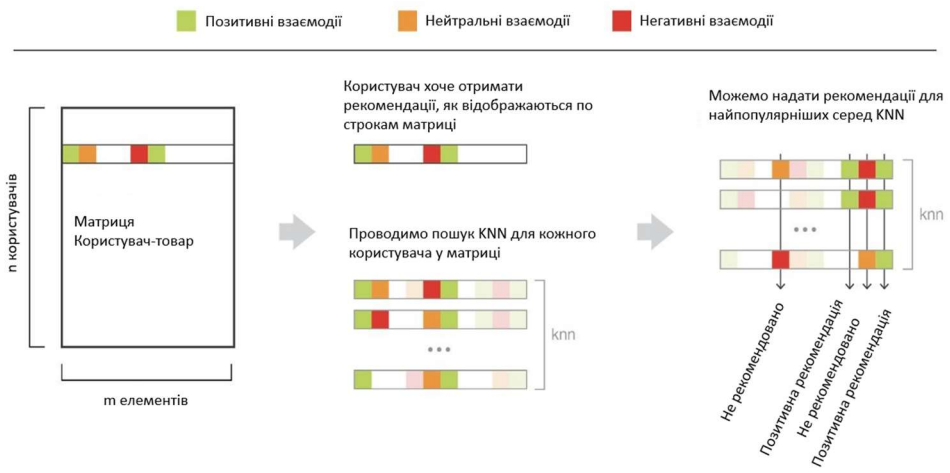

Рисунок 1.2 – Робота метода колоборативної фільтрації.

Популярні продукти будуть більш популярними в довгостроковій перспективі, і буде бракувати нових і різноманітних варіантів Цей процес розпочинається з аналізу позицій, які вже сподобалися користувачеві. Вони стають основою для подальшого пошуку схожих продуктів чи елементів. Використовуючи метод, наприклад KNN, є можливість розрахувати схожі продукти, які можуть зацікавити цільового користувача. Ця модель дозволяє створити кластери схожих елементів і пропонує нові позиції з цих кластерів користувачеві. Важливою перевагою такого підходу є можливість забезпечення довгострокової популярності певних продуктів. Чим більше елементів користувач сприймає як приємні чи корисні, тим більше схожих позицій може бути запропоновано. Це стимулює утримання популярних продуктів у списку рекомендацій, що сприяє їхній постійній привабливості для користувачів.

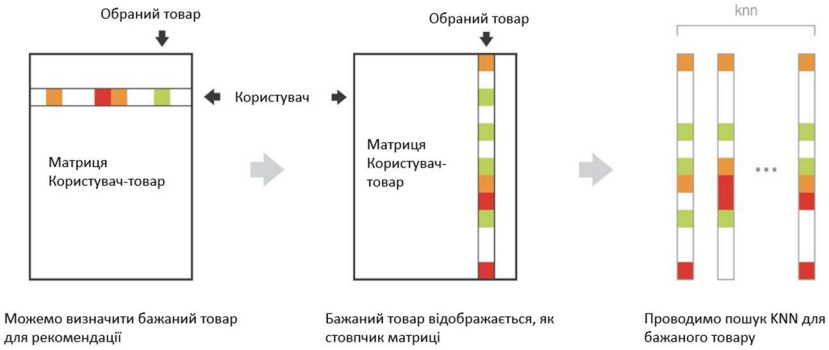

Рисунок 1.3 – Надання рекомендацій на основі матриці взаємодій.

Переваги:

Не потребує детальних характеристик і контекстних даних продуктів або елементів. Може допомогти користувачам відкрити нові інтереси, навіть якщо вони не шукають їх активно, рекомендуючи нові предмети, подібні до того, що їх цікавить.

Недоліки:

Зі зростанням бази користувачів алгоритми страждають через великий обсяг даних і відсутність масштабованості. Нестача інформації може призвести до труднощів у рекомендаціях нових продуктів або користувачів, оскільки пропозиції ґрунтуються на історії взаємодій. Відсутність різноманітності в довгостроковій перспективі. Це може здатися нелогічним, оскільки суть спільної фільтрації полягає в тому, щоб рекомендувати нові елементи користувачеві. Однак, оскільки алгоритми працюють на основі історичних рейтингів, вони не рекомендуватимуть елементи з невеликою чи обмеженою кількістю даних. Популярні продукти будуть більш популярними в довгостроковій перспективі, і буде бракувати нових і різноманітних варіантів [32].

1.3.3 Модель на основі бандитських алгоритмів

Системи рекомендацій добре працюють, у випадку наявності великої кількості даних про переваги елементів користувача. Маючи багато даних, як результат, підвищуємо високу ймовірність щодо того, що подобається користувачам. І навпаки, маючи дуже мало даних, у нас низька ймовірність визначення, Незважаючи на низьку відповідність, моделі рекомендації схильні жадібно просувати елементи, з якими було більше взаємодій у минулому. І оскільки вони впливають на те, наскільки оприлюднюється товар, потенційно релевантні елементи, які не рекомендуються, продовжують не отримувати рекомендацій, створюючи постійний цикл зворотного зв’язку. Підхід використання бандицьких алгоритмів вирішує описану проблему шляхом

дослідження та моделювання невизначеності, аналізуючи це, алгоритм визначає про актуальність невивчених предметів. Це важливо, коли набір елементів швидко змінюється, наприклад, для реклами та новин, реклами. Якщо нові елементи постійно додаються, очікування збору пакетних даних перед повторним навчанням моделі може бути надто повільним, описаний алгоритм гарно підходить для вирішення цього, оскільки може поступово опрацьовувати нові дані та адаптивно зосереджуватися на товарах із вищим рейтингом. Як результат, це зменшує проблему холодного старту, хоча може понизити точність надання рекомендацій [33].

Поширеним прикладом бандитських алгоритмів є UCB (англ. Upper Confidence Bound), розглянемо його на прикладі LinUCB Yahoo для рекомендацій новин. Ридж-регресія навчена оцінювати кількість користувачів яким подобається товар лінійно на рисах руки, що означає, що вона намагається знайти найкращі значення коефіцієнтів, які мінімізують суму квадратів помилок між фактичними і прогнозованими значеннями цікавості товару. Потім UCB виводиться шляхом підсумовування прогнозованої цікавості та стандартного відхилення регресії [34].

1.3.4 Гібридні моделі

Гібридні системи рекомендацій поєднують в собі кілька методів рекомендацій, щоб надати більш релевантні та різноманітні елементи. Ці системи можуть поєднувати різні підходи, наприклад, фільтрацію на основі контенту та спільну фільтрацію, чи інші методи. Використовуючи різноманіття стратегії, можуть подолати обмеження окремих методів та запропонувати більш точні та вичерпні рекомендації. Розглянемо типи таких систем детальніше на рис 1.4.

Рисунок 1.4 – Зважена гібридна система рекомендацій.

У системі зважених рекомендацій можливо визначити кілька моделей, які можуть добре інтерпретувати набір інформації. Система зважених рекомендацій отримує вихідні дані з кожної моделі та поєднує результат у статичні зважування, вага котрих не змінюється в поєднанні у тестовому наборі. Наприклад, можна об'єднати модель на основі вмісту та модель спільного фільтрування елементів, і кожна з них має вагу половину відсотків для формування фінального списку рекомендацій. Перевага використання зваженого гібриду полягає в тому, що відбувається інтеграція декількох моделей.

Рисунок 1.5 – Схема варіації гібридної моделі на базі перемикання.

На рис. 1.5 зображена схема варіації гібридної моделі на базі перемикання. Модель вибирає єдину систему рекомендацій на основі ситуації. З початку, необхідно встановити критерії селектора рекомендацій на основі профілю користувача або інших функцій. Підхід перемикання інтегрує додатковий рівень у рекомендаційну модель, що обирає відповідну модель для вирішення задачі. Важливо зазначити, що система рекомендацій чутлива до слабких та сильних сторін кожної з складових моделі рекомендацій.

Рисунок 1.6 – Змішана система рекомендацій.

Змішаний гібридний підхід (рис. 1.6) спочатку використовує профіль користувача та функції для створення різних наборів даних-кандидатів. Система рекомендацій відповідно вводить інший набір кандидатів у модель рекомендації та комбінує прогноз для отримання рекомендації результату. Змішана гібридна система рекомендацій здатна надавати велику кількість рекомендацій одночасно та адаптувати частковий набір даних до відповідної моделі, щоб мати кращу продуктивність.

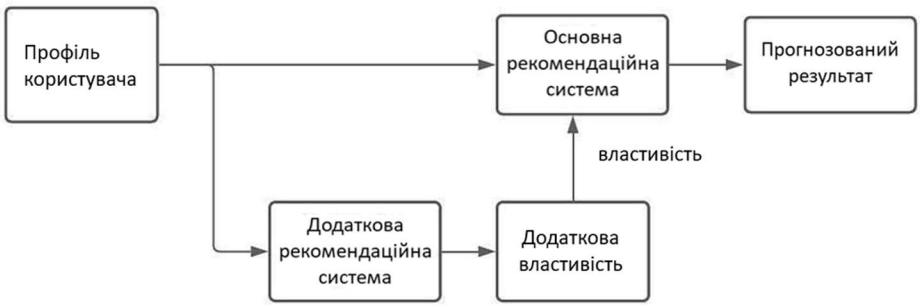

Рисунок 1.7 – Гібридна модель комбінації функцій.

У описаній моделі відбувається комбінація функцій, тобто додається до системи віртуальну модель рекомендацій, яка працює як розробка функцій щодо вихідного набору даних профілю користувача (рис. 1.7). Наприклад, можна додати функції спільної моделі рекомендацій до моделі рекомендацій на основі вмісту. Гібридна модель здатна розглядати спільні дані з підсистеми, спираючись виключно на одну модель.

Прикладом гібридної моделі може слугувати поєднання колаборaційної фільтрації та фільтрації на основі контенту, що використовує характеристики товарів та взаємодії користувачів для підвищення точності рекомендацій.

Компонент на основі вмісту аналізує функції або атрибути предмета для створення профілів користувачів на основі їхніх уподобань щодо конкретних характеристик. Компонент колаборaційної фільтрації – визначає подібність користувачів або шаблони їх взаємодії, щоб рекомендувати елементи, які сподобалися схожим користувачам, але з якими активний користувач ще не взаємодіяв. Рекомендації з обох підходів поєднуються, щоб надати більш різноманітні та точні пропозиції. Наприклад, якщо користувач любить фантастику і має такі ж уподобання, як інший користувач, який любить комедії, система може запропонувати фантастичні комедії.

До переваг відноситься зменшення впливу слабких сторін індивідуальних підходів. На основі вмісту зменшує проблему холодного запуску для нових елементів, тоді як спільна фільтрація надає випадкові рекомендації. Також, є покращена персоналізація, що поєднує вподобання користувача та характеристики предметів для більш персоналізованих пропозицій. Модель буде пропонувати різноманітні пропозиції, враховуючи як смаки користувачів, так і особливості предметів. З точки розгляду недоліків, можна віднести обмежене передбачення, тому алгоритм все ще може не мати можливості представити нові або несподівані предмети, оскільки часто рекомендує на основі наявних уподобань користувача чи функцій товарів. Також, значною мірою залежить від наявності та якості функцій предмета та даних про взаємодію з користувачем [35].

1.4 Формування задач роботи

Необхідно розробити комп’ютерну модель персоналізованих рекомендацій вибору товарів на базі мобільної платформи Android, з метою

підвищення ефективності персоналізації вибору товарів, за рахунок використання алгоритмів кластеризації.

Для досягнення вище визначеної мети були поставлені такі задачі:

1. Створення дата-сету з набором товарів.
2. Програмна реалізація моделі на платформі додатку Android.
3. Реалізація обраних алгоритмів кластеризації, а саме K-means++, HBDSCAN та Mean shift.
4. Реалізація обраного пошукового алгоритму.
5. Тестування ефективності роботи розробленої комп'ютеризованої моделі.
6. Складання звіту та надання рекомендацій щодо застосування моделі.

Висновки за розділом 1

Використання алгоритмів пошуку передбачає врахування багатьох характеристик опису елементів, щоб рекомендувати продукти, які відповідають індивідуальним смакам та потребам. Використання інформованих алгоритмів пошуку є більш відповідним до умов поставленої задачі оскільки враховує контекст запиту.

Алгоритми кластеризації надають можливість згрупувати користувачів чи товари на основі схожості їхніх характеристик. Одним із підходів є кластеризація на основі центроїдів, де об'єкти об'єднуються навколо центральних точок. Одним з найпростіших алгоритмів кластеризації на основі центроїдів є метод K-means. Ієрархічна кластеризація надає більшу деталізацію, розглядаючи відношення підгруп на різних рівнях. Кластеризація на основі щільності визначає групи на основі густини об'єктів у просторі, а кластеризація на основі розподілу використовує ймовірність приналежності об'єкта до певного кластера. Одним з популярних алгоритмів кластеризації на основі щільності є алгоритм DBSCAN, у цьому алгоритмі кластери утворюються навколо точок щільності. Алгоритми кластеризації дозволяють

обрати необхідні елементи для надання рекомендацій, сприяючи точнішому визначенню їхніх уподобань та потреб.

Розглянуто різні моделі надання персоналізованих рекомендацій, таких як фільтрація на основі контенту, колаборативна фільтрація, бандитських алгоритмів та різних типів гібридних моделей, що поєднують існуючі підходи. Вибір ефективного підходу залежить від конкретних характеристик товарів та цілей власника системи. Персоналізовані рекомендації відіграють важливу роль у розвитку електронної комерції, оскільки дозволяють підвищувати конверсію і залученість клієнтів.

Формування задач роботи у цьому контексті передбачає дослідження існуючих методів кластеризації для конкретного типу даних, реалізацію обраного пошукового механізму та розробку моделі для вирішення проблеми ефективного надання рекомендацій. Також, важливо враховувати взаємозв'язок між вибором алгоритму та конкретними характеристиками вводу користувачів чи об'єктів.

РОЗДІЛ 2.
ОГЛЯД ТЕХНОЛОГІЙ ТА РОЗРОБКА КОМП'ЮТЕРНОЇ МОДЕЛІ СИСТЕМИ

2.1 Опис обраних технологій для серверної частини моделі

У даній роботі пропонується створення клієнт-серверної архітектури для реалізації моделі. Мовою розробки було обрано Python, що найбільше підходить під контекст обраної схеми моделі та машинного навчання та штучного інтелекту, що продовжує свій розвиток, а їх використання постійно зростає через потребу в автоматизації. Штучний інтелект дозволяє створити рішення для великої кількості різнотипних завдань, таких як фільтри спаму, пошукові системи та системи рекомендацій. Python дозволяє зменшити об'єм коду, завдяки великій кількості створених бібліотек [36].

Архітектурний підхід системи виконаний на базі партерну REST. Це підхід для мережевих застосунків, що ґрунтується на принципах роботи з ресурсами через стандартні HTTP методи, представленнями ресурсів у форматах, таких як JSON або XML, що пропонує простий та ефективний спосіб взаємодії клієнтів і серверів, підвищуючи масштабованість системи [37].

Попит на інтелектуальні рішення зростає, що підкреслює потребу в подальшому розвитку систем штучного інтелекту для вирішення завдань з надання персоналізованих рекомендацій. Python є ефективним інструментом для реалізації пошукових алгоритмів та алгоритмів кластеризації, завдяки вже створеним рішенням. Python забезпечує надійні фреймворки та бібліотеки, які значно скорочують час розробки програмного забезпечення [38].

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn import metrics
from sklearn.cluster import KMeans
from sklearn.cluster import MeanShift
from sklearn.cluster import HDBSCAN
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel
from sklearn.metrics.pairwise import cosine_similarity
from memory_profiler import profile
import time
```

Рисунок 2.1 – Імпорт бібліотек.

Для головного функціоналу моделі необхідно імпортувати бібліотеки зазначені на рис 2.1.

Pandas – це пакет обробки даних на Python, що можна використати для зчитування табличних даних, тобто у вигляді рядків і стовпців, також відомих як фрейми даних. Дозволяє імпортувати набори даних з баз даних, електронних таблиць, файлів формату CSV, та інших, та використовується в робочому процесі інтелектуального аналізу даних. Також, є функція очистки датасету, що є важливими у випадку коли є відсутні значення [39].

Seaborn, широко визнана бібліотека Python для візуалізації даних, побудована на основі Matplotlib і вирізняється своїм зручним інтерфейсом для створення візуально переконливої та інформативної статистичної графіки. Відмінною особливістю Seaborn є його вміння створювати складні багатоплощинні візуалізації. Також для візуалізації використовується Matplotlib для побудови пласких графіків [40-41].

Для оцінки використання пам'яті програмою використовується профілізатор пам'яті, основне призначення якого виявлення витоків пам'яті та покращення використання пам'яті у програмах на Python. Профілізатор пам'яті аналізує ефективність використання місця в коді та характеристики використовуваних пакетів, пропонуючи ідеї для оптимізації використання пам'яті. Пакет memory_profiler перевіряє використання пам'яті

інтерпретатором на кожному рядку. Стовпчик інкрементів дозволяє виявити місця в коді, де виділяються великі обсяги пам'яті. Це особливо важливо при роботі з масивами [42].

Для створення REST-серверу використаний фреймворк Flask. Flask - це фреймворк, написаний на мові Python, спеціально створений для розробки додатків. Цей фреймворк побудовано на основі інструментарію Werkzeug WSGI, який надає основні функції веб-сервера для управління HTTP запитами та відповідями [43].

Для реалізації основних функцій моделі, використовується бібліотека Scikit-learn – це пакет, що містить вже створенні алгоритми для машинного навчання, такі як класифікація, кластеризація, та регресія. Пакет надає набір інструментів для таких завдань, як розробка комп’ютерних моделей, калькуляція метрик для оцінювання ефективності, а також різноманітні доповнення, що наприклад включають функції попередньої обробки інформації датасету [44].

2.2 Опис обраних технологій для додатку на платформі Android

Реалізація клієнта у вигляді мобільного додатку дозволяє зручним способом для взаємодії з моделлю.

Android є хорошим вибором для розробки мобільних додатків, оскільки це найпоширеніша мобільна операційна система у світі. Він має велику базу користувачів, що означає, що ви можете охопити багато людей за допомогою свого додатка. За оцінками, у 2023 році у світі налічувалося 7,33 мільярда користувачів смартфонів, 49,11%, або 3,6 мільярда, користуються смартфонами Android [45].

Додаток розробляється під версію Android 14, що є останньою версію, на момент розробки моделі. Однією з найважливіших змін у Android 14 є нова система дозволів для повідомлень. Тепер усі програми за замовчуванням не можуть надсилати сповіщення [46].

Обрана операційна система також має відкритий вихідний код, що означає, що його можна безкоштовно використовувати та змінювати. Це робить його гарним вибором для розробки моделі [47].

Мовою розробки додатку є Kotlin. Його використання для розробки Android додатків обґрунтоване, завдяки його компактності інтуїтивному синтаксису, що як результат, дозволяє писати менше коду. Також надає безпеку за типами, що підвищує надійності створених застосунків, та має можливість працювати з вже існуючим Java-кодом. Ця мова має підтримку від Google та активно рекомендується для розробки нових Android проектів [48].

Для зберігання тимчасових файлів використовується бібліотека Room, що забезпечує рівень абстракції над SQLite, з метою забезпечення зручного програмного інтрефейсу взаємодії для створення запитів SQL і перевірку сформованого запиту SQL під час компіляції [49].

Програма реалізує ін’єкції залежностей за допомогою Dagger Hilt – це бібліотека, яка спрощує реалізацію ін’єкції залежностей шляхом інтеграції з інфраструктурою Android і зменшення кількості шаблонного коду, необхідного для керування залежностями в програмі [50].

Для взаємодії серверною частиною використовується бібліотека Retrofit – бібліотека, яка дозволяє реалізувати під’єднання до сервісів API на основі HTTP. Бібліотека використовує основну функціональність бібліотеки OkHttp, усуваючи шаблонний код, який потрібно писати. Retrofit дозволяє вказати заголовки запиту, наприклад, передати заголовок авторизації через анотації або перехопити запит, застосувати різні типи конвертерів для перетворення об'єктів запиту чи відповіді до необхідного формату [51].

Архітектурним підходом для побудови структури мобільного додатку є підхід MVVM, що розділяє компоненти програми. Це розподіл гарантує, що елементи інтерфейсу користувача залишаються відокремленими від бізнес-логіки, а бізнес-логіка, в свою чергу, залишається ізольованою від операцій з базою даних, та сторонніми API. Рівень Model втілює головний функціонал роботи з даними додатку, що не може бачити користувач, охоплює як

віддалені, так і локальні джерела даних, репозиторії та класи. Рівень View містить код інтерфейсу користувача, написаний у форматі XML. Він надсилає дії користувача до ViewModel, але не отримує відповіді безпосередньо. Останній рівень виконує роль сполучної ланки між представленням та бізнес-логікою. ViewModel не має доступу до методів окремого екрану, з яким відбувається взаємодія [52].

Для написання асинхронного коду використовується підхід Kotlin Coroutines, Звичайне програмування може блокувати виконання коду, коли очікується результат виклику деякої операції, наприклад, мережевого запиту чи завантаження файлу. Даний підхід дозволяє уникнути цієї проблеми шляхом використання легковагових потокоподібних структур [53].

2.3 Вхідні та вихідні дані

Для вхідними даними для цієї моделі слугує окремий товар, характеристики, якого вводить користувач. Як результат роботи моделі користувач повинен отримати список рекомендованих товарів з відсотком відповідності її запиту. Припустимо, дано набір характеристик товару, кожна характеристика описується, як x_n, де n – це кількість характеристик. Тоді k – кількість рекомендованих товарів, r_k – це рекомендований моделлю товар, p_k – це оцінка відповідності у відсотках. Нижче (на рис.2.2) зображена схема, що описує вхідні та вихідні дані.

Рисунок 2.2 – Схема опису вхідних та вихідних даних.

На прикладі мікрохвильової печі, характеристиками товару, що необхідні для роботи моделі є ціна, колір, об'єм, довжина, ширина, назва

моделі, та виробник. Для перевірки роботи моделі датасет обов'язково повинен містити категоріальні дані. Для описаного прикладу вище, це може бути колір, виробник та назва моделі.

2.4 Створення дата-сету з набором товарів

Для роботи моделі необхідний датасет з набором товарів які будуть використані для розрахунків. Було створено датасет з даними про мікрохвильові печі з українських відкритих джерел, з переліком характеристик цифрових та категоріальних. Для тестування надання персоналізованих рекомендацій було обрано об'єм датасету розміром 100 елементів. Дані для датасету про існуючі товари з отримані з відкритих джерел, а саме з агрегаторів товарів.

Створений датасет, містить 9 типів характеристик, що описують товар, окрім id_product, що є ідентифікаційним номером, та не використовується в обчисленнях. Категоріальними даними є колір, виробник та назва виробу. Характеристики ширини, глибини та висоти, містять значення з плаваючою точкою (рис. 2.3). Ціна, потужність, та максимальна споживана потужність – цілі числа.

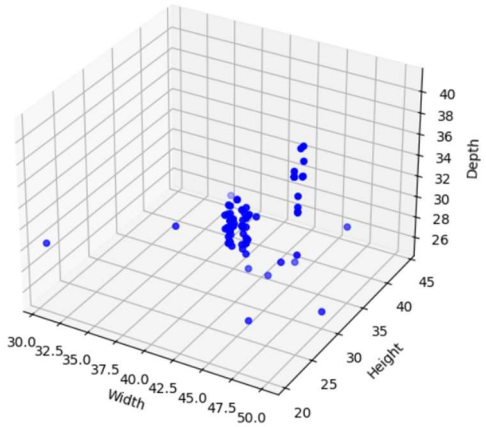

Рисунок 2.3 – Тривимірна точкова діаграма ширини, висоти та глибини.

Загальний тренд розподілу характеристик розмірів товару – розміщення у трьох стовбцях, з варіаціями у глибині. Перший тип, з високим показником

висоти, більше 45 см. Другий, має подібні значення ширини та висоти. Третій, має схожі значення з другим, з малою відмінністю у ширині, що складає приблизно 2 см. На діаграмі присутні товари унікальних розмірів, які можуть бути оцінені, як аномалії.

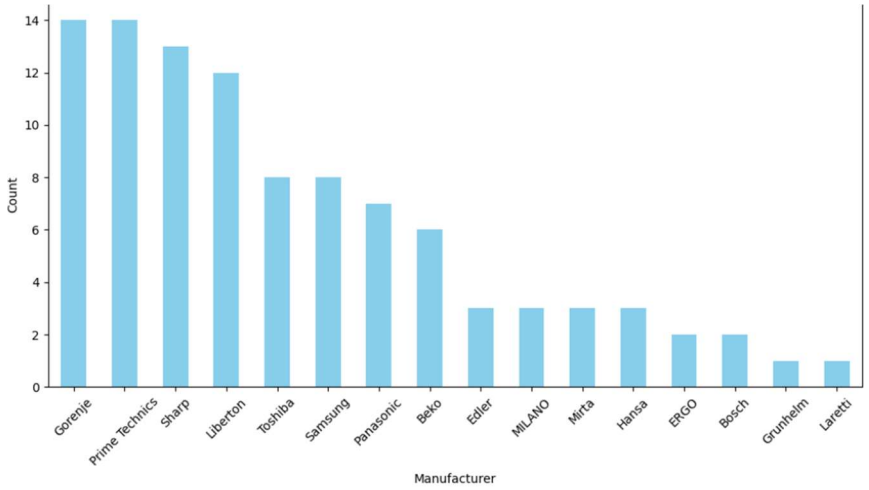

Рисунок 2.4 – Гістограма кількості товарів за виробником.

З гістограми на рис. 2.4, можна зробити висновок, що найбільша кількість товару має категорія виробника Prime Technics та Gorenje, що складає 14 елементів. Найменше значення, мають категорії – Grunhelm та Laretti, 1 елемент.

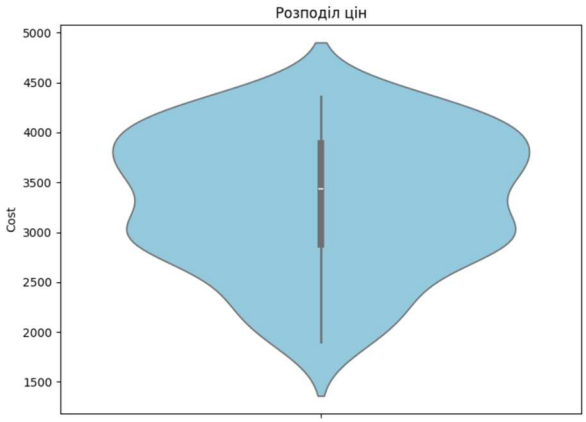

Рисунок 2.5 – Розподіл цін.

На даній діаграмі (рис 2.5) можна побачити що найбільша кількість товару має ціну 3000 та 4000 відповідно до пікових значень. Мінімальне значення ціни 1900. Максимальне значення у датасеті – 4363.

2.5 Запропоноване рішення

У даній роботі пропонується створення гібридної моделі з використанням алгоритмів кластеризації та рекомендацій на основі TF-IDF векторів.

Перша частина моделі надає рекомендації будуть на основі схожості кластерів. Товари групуються в кластери на основі їхніх характеристик. Потім, щоб надати персоналізовану рекомендацію користувачу, вибирається кластер, який найбільш схожий на кластер, в якому вже знаходиться товар. Після того, як кластер визначений, можна надати рекомендацію користувачу, вибираючи об'єкти, які є найбільш схожими на об'єкти, які вже знаходяться в кластері. За допомогою розрахунку евклідової відстані визначається відповідність вимогам користувача.

Евклідова відстань є відображенням найкоротшої міри довжини між двома векторами. Це квадратний корінь із суми квадратів різниць між відповідними елементами. Метрика евклідової відстані відповідає L2-нормі різниці між векторами та векторними просторами [54].

Припустімо дві точки, такі як (m_1, n_1) і (m_2, n_2) у двовимірній координатній площині. Отже, формула евклідової відстані визначається як:

$$E = \sqrt{(m_2 - m_1)^2 + (n_2 - n_1)^2} \quad (2.1)$$

де, E – евклідова відстань;

m_1, n_1 – координата першої точки;

m_2, n_2 – координата другої точки.

Евклідова відстань використовується в багатьох алгоритмах, як стандартна метрика відстані для вимірювання схожості між двома записаними спостереженнями. Однак спостереження, які потрібно порівняти, мають включати безперервні характеристики та мати числові змінні, такі як вага,

зріст, зарплата тощо. Користувачі здебільшого обирають його для обчислення відстані між двома рядками даних, які складаються з числових значень.

Друга частина моделі використовує категоріальні параметри для надання персоналізованих рекомендацій, завдяки інформованого алгоритму пошуку TF-IDF – це метод представлення текстових документів у вигляді числових векторів. Це дозволяє використовувати математичні методи для аналізу та порівняння даних, аналізу тексту та машинного навчання.

TF – позначення частоти термінів, описує кількість разів, коли слово з'являється в документі. Приклад – характеристика, яка присутня багато разів у кількох товарів, отримає вищий ранг, оскільки вона може вказувати на контекст [55].

$$TF(s,k) = \frac{f_1(s,k)}{f_2(s)} \tag{2.2}$$

де s – кількість слів;

k – датасет;

$f_1(s,k)$ – співвідношення s до k;

$f_2(s)$ – загальне s у датасеті k.

За допомогою цього описаної формули можемо розрахувати частоту появи певного слова в елементі. Наприклад, якщо обраному переліку елементів є 8 категоріальних ознак і містить три слова «black», коефіцієнт TF буде дорівнювати (3/8).

IDF – позначення зворотної частоти документу, описує міру того, наскільки рідкісним або поширеним є слово у датасеті. Слова, які зустрічаються в багатьох документах, матимуть низький показник IDF, тоді як слова, які зустрічаються лише в кількох документах, матимуть високий показник IDF [56].

$$IDF(h,k) = \ln\left(\frac{f_1(k)}{f_2(h)}\right) \quad (2.3)$$

де h — характеристика елемента;

k – датасет;

$f_1(k)$ – це загальна кількість елементів у датасеті;

$f_2(h)$ – кількість елементів, що містять обрану характеристику h.

Загальна формула розрахунку описує, що TF-IDF є добутком частоти терміну та зворотної частоти елемента. Це надає більшого значення характеристиці, яка рідко зустрічається у датасеті. Надалі, розрахований показник використовується для обчислення косинусної подібності між обраним елементом та кожним зі списку рекомендованих товарів.

Наступним кроком є поєднання результатів роботи двох підходів. Якщо датасет має змішані дані – у цифровому форматі та категоріальному, тобто у вигляді опису таких характеристик, як колір, назва, виробник, та в не враховує їх у кластеризації, модель може обробити ці дані окремо, таким чином підвищивши точність наданих рекомендацій. Або, якщо датасет має більшість даних у категоріальному, то є можливість провести розрахунки кластеризації та TF-IDF окремо.

Останнім кроком, є порівняння двох отриманих списків з рекомендаціями, за допомогою розрахунку відстані між позиціями товарів у списках. У випадках однакової відстані між парами товарів A, B та B, A, за пріоритетним товар слід вважати, той який має на кращий показник точності.

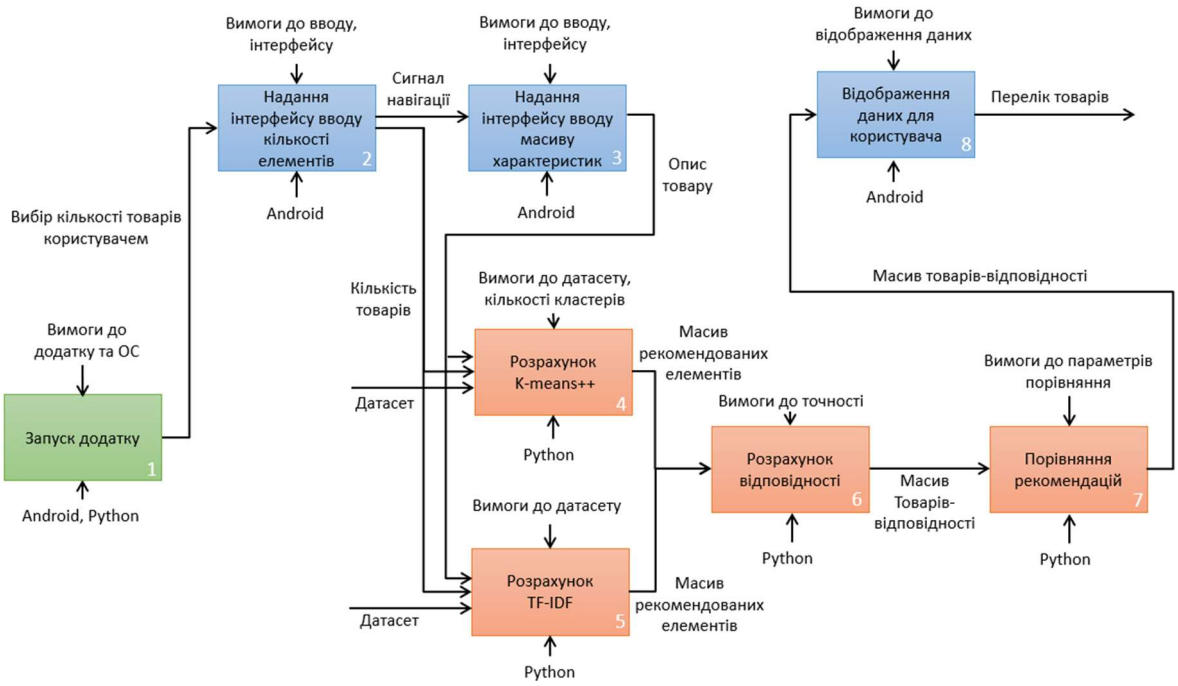

Рисунок 2.6 – Розроблена модель надання рекомендацій в нотації IDEF0.

Розглянемо схему моделі створеної моделі по блоках (рис. 2.6). Де помаранчевий колір відповідає за функціональні блоки, такі що не видимими для користувача. Зелений колір відповідає за блок ініціалізації. Синій колір – це видимі блоки, з якими користувач має можливість взаємодіяти.

Блок 1 “Запуск додатку”: Це ініціалізація додатку та системи, є стартовою точкою, де відбувається ініціалізація змінних, завантаження моделей та інтерфейсу. Тут перевіряється з’єднання з сервером. Головною вимогою додатку є підтримка мобільними пристроями на базі платформи Android.

Блок 2 “Надання інтерфейсу вводу кількості елементів”: Для роботи цього блоку необхідно отримати ціле число кількості товарів, що буде відображена. Вимогою до цього блоку відображення помилок вводу. Вихідними даними є число кількості рекомендацій.

Блок 3 “Надання інтерфейсу вводу масиву характеристик”: Відображається повний перелік полів форми, для визначення характеристик, що відносяться до бажаного товару. Для старту роботи, цей блок повинен

отримати сигнал керування від блоку 2, тобто сигнал навігації. Вихідними даними є опис товару.

Блок 4 “Розрахунок K-means++”: Це етап, де дані групуються в кластери на базі отриманих загального датасету товарів та вихідних даних з блоків 2 та 3. Тут використовуються алгоритм кластеризації для створення сегменту товарів для користувача. Важливою вимогою є розрахунок правильної кількості кластерів. Спільною вимогою для цього та наступного блоку є відповідність датасету. У випадку, якщо датасет містить неповні дані, стовпці з пустими значеннями будуть проігноровані, що знизить якість рекомендацій. Результатом роботи є масив товарів, що можуть бути запропоновані.

Блок 5 “Розрахунок TF-IDF”: Цей блок є функціональним. Відповідає виконання пошукового алгоритму для надання переліку товарів на виході, де текстові дані перетворюються на числові вектори за допомогою методу TF-IDF.

Блок 6 “Розрахунок відповідності”: Тут обчислюється подібність між елементами чи товарами, за допомогою евклідової відстані для числових характеристик та косинусної схожості для текстових характеристик. На вхід отримує два масиви товарів, з блоку 4 та 5. Вимогою є коректність розрахунків відповідності, тобто, якщо датасет має товар, що користувач шукає, відстань до нього повинна дорівнювати нулю.

Блок 7 “Порівняння рекомендацій”: Цей етап включає порівняння рекомендацій, отриманих з різних методів, за допомогою обчислення відстані між позиціями товарів у списках, та формування рейтингу товарів. Для порівняння головною вимогою є розрахунок відсотків на базі відстані. Розмір масиву зменшується до введеної кількості елементів.

Блок 8 “Відображення даних для користувача”: Це показ результатів користувачеві через інтерфейс додатку – це фінальний список рекомендацій. Тут вимогою є коректне відображення даних, тобто елементи відображатися послідовно.

Ці кроки можуть бути впроваджені в коді програми як окремі функції чи методи для виконання певних завдань.

Висновки за розділом 2

Було розглянуто обрані технології для створення комп'ютеризованої моделі системи. Почавши з обраної мови програмування, Python виявився відмінним вибором для реалізації серверної частини моделі, оскільки містить багато бібліотек для роботи з кластеризацією та машинним навчанням. Алгоритми кластеризації взяті зі стандартних реалізацій у пакеті scikit-learn.

Додатково, використання архітектурного підходу REST забезпечує ефективну взаємодію між клієнтами та серверами, що є важливим для масштабованих систем.

Реалізація мобільного додатку вимагає уважного вибору технологій. Використання Android 14 надає можливість використовувати нові функції операційної системи. Мова програмування Kotlin виявилася оптимальною завдяки своїй компактності та безпеці за типами. Для зберігання даних обрано Room, а для взаємодії з сервером – Retrofit. Використання підходу MVVM дозволяє структурувати код. Kotlin Coroutines допомагають уникнути блокування програми при асинхронних операціях. Одним із ключових етапів було створення дата-сету з набором товарів. Для цього були використані дані про мікрохвильові печі, зібрані з українських відкритих джерел.

У цій роботі запропоновано гібридну модель, поєднуючи алгоритми кластеризації та рекомендацій з використанням TF-IDF векторів. Модель надає рекомендації на основі схожості кластерів товарів та персоналізовані рекомендації для користувачів. Вона використовує евклідову відстань для визначення схожості та TF-IDF для категоріальних параметрів товарів, що дозволяє враховувати текстову інформацію. Цей підхід дозволяє отримати більш точні та персоналізовані рекомендації для користувачів, враховуючи різноманітні характеристики товарів.

РОЗДІЛ 3.
ТЕСТУВАННЯ РОЗРОБЛЕНОЇ МОДЕЛІ ТА ОСОБЛИВОСТІ РОБОТИ

3.1 Оцінка ефективності алгоритмів кластеризації у системі персоналізованих рекомендацій

Для реалізації системи персоналізованих рекомендацій на базі кластеризації, необхідно дослідити ефективність роботи моделі на різних алгоритмах кластеризації. Для оцінки ефективності надання рекомендацій використовуються три параметри, відповідність запиту користувача у відсотковому відношенні, використання оперативної пам’яті, та час виконання запиту.

Розрахунки проводяться на стороні серверу, на базі процесору Intel(R) Core(TM) i7-7700HQ CPU, з частотою 2,80 ГГц. Встановлена оперативною пам'ять - 16,0 ГБ.

Перший кроком, проводиться кластеризація, на основі введеного масиву характеристик. Як результат, отримуємо рекомендації на основі відповідного запиту кластеру. Після цього відбувається перевірка вимогам користувача за допомогою розрахунку евклідової відстані для обраного товару та кожного елементу у цьому списку.

Останнім кроком є розрахунок відповідності у відсотках, для цього обирається найбільша евклідова відстань з наявної у списку рекомендованих елементів. Це відбувається за формулою [57]:

$$S = 1 - \frac{x_i}{x_{max}} \tag{3.1}$$

де S – схожість у відсотках;

x_i – евклідова відстань і-го елемента зі списку;

X_{max} – максимальна значення відстані у списку.

Для дослідження були обрані три алгоритмі, такі як K-means++, HDBSCAN та Mean Shift, кожен з яких відноситься різних типів кластеризації. Щоб підтвердити точність роботи моделі, нам необхідно ввести характеристики товару з датасету, та перевірити з яким відсотком надаються рекомендації. Якщо, введений товар збігається на 100 відсотків, то модель працює правильно. Для порівняння, результатів різних алгоритмів необхідно порівняти відсотки схожих елементів, що відрізняються за одною чи декількома характеристиками. До прикладу, оберемо мікрохвильову піч з характеристиками зазначеними на рис. 3.1.

```
microwave = pd.DataFrame({
    'manufacturer': ['Gorenje'],
    'color': ['black'],
    'name': ['MO17E1B'],
    'cost': [2689],
    'width': [45.5],
    'height': [26.1],
    'depth': [35.3],
    'power': [700],
    'max_consumption': [1150],
})
```

Рисунок 3.1 – Опис характеристик мікрохвильової печі.

Першим було протестовано K-means++. Для коректної роботи цього алгоритму нам необхідно визначити кількість кластерів. Для цього можна використати метод ліктя, що визначає оптимальну кількість кластерів у наборі даних шляхом побудови суми квадратів у середині кластера проти кількості кластерів.

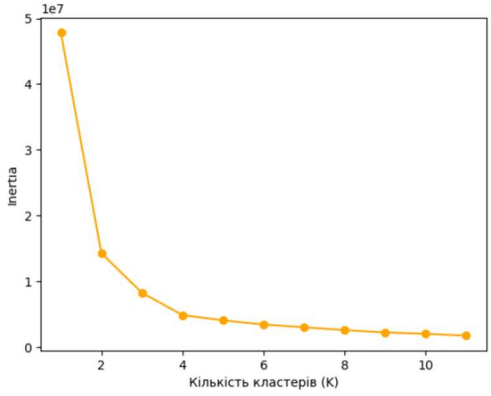

Рисунок 3.2 – Метод ліктя.

Вісь у описує інерцію кластерів, наскільки якісно відбувся процес кластеризації для методу K-means++. Цей параметр обчислюється шляхом вимірювання відстані між кожною точкою даних і її центроїдом, зведення цієї відстані в квадрат і підсумовування цих квадратів по одному кластеру. «Точка ліктя» розташована там, де швидкість зниження різко змінюється, та вказує на оптимальну кількість кластерів для використання в алгоритмі кластеризації, у даному випадку – визначена кількість кластерів дорівнює 3 (рис. 3.8).

3.2 Тестування роботи моделі на базі кластеризації

Для реалізації системи персоналізованих рекомендацій на базі кластеризації, необхідно дослідити ефективність роботи моделі на різних алгоритмах кластеризації, щоб визначити, який більше підходить для вирішення задачі (табл. 3.1, 3.2, 3.3). Отже, було обрано три різні типи алгоритми кластеризації. Важливо зауважити, що введений елемент не був врахований в розрахунках оскільки його розрахунок схожості дорівнює нулю.

Перший, на основі центроїдів – K-means++, що працює шляхом випадкового обрання першого центроїда з датасету. Наступним кроком є вибору послідовний вибір центроїдів на базі розподілу ймовірностей. Цей метод допомагає досягти кращої ініціалізації, порівняно з випадковим вибором центроїдів у класичному підході K-means.

	Product	Similarity
0	M017E1B	0.00
18	M017E1W	10.07
19	M020E1WH	130.34
22	M017E1S	152.00
11	MW-4001BR	199.42
20	LMW-2074M	211.43
28	M017E1BH	251.16
29	LMW-2079M	284.72
31	R200BKW	322.38
14	PMW 20711 KB	331.45

Рисунок 3.3 – Рекомендації K-means++.

Після запуску моделі на основі алгоритму K-means++, модель може визначити який елемент був введений через те, що модель виставила 100 відсотків збігу введеному елементу. На додаток до цього, отримано список десяти рекомендацій товарів (рис. 3.3).

Другий, Mean Shift – це непараметричний алгоритм кластеризації, який використовується для пошуку кластерів у наборі даних. Він працює шляхом ітеративного зміщення елементів у датасеті у бік максимальної щільності кластеру товарів. Цей процес триває до певного моменту, коли елементи осідають у найщільніших областях, утворюючи кластери.

	Product	Similarity
0	M017E1B	0.00
18	M017E1W	10.07
13	PMW 20757 HB	100.35
17	PMW 20711 KW	126.33
19	M020E1WH	130.34
11	MW-4001BR	199.42
14	PMW 20711 KB	331.45
1	Y35MW	379.32
15	PMW 20715 KB	379.42
12	PMW 20757 HW	382.12

Рисунок 3.4 – Рекомендації Mean Shift.

Третій, HDBSCAN – це алгоритм кластеризації на основі щільності, не вимагає заздалегідь вказувати кількість кластерів і може ідентифікувати кластери різної форми. Він використовує ієрархічний підхід для побудови дерева кластерів, дендрограму, якого можна побачити на рис 3.5. Вісь *y* позначає відстань, а вісь *x* позначає товари. Відстань у дендрограмі HDBSCAN відноситься до міри відмінності або подібності між кластерами, коли вони зливаються або розщеплюються під час процесу ієрархічної кластеризації.

Менші висоти вказують на більшу подібність, тоді як більші висоти вказують на більшу відмінність між кластерами. Це ієрархічне представлення дозволяє візуально зрозуміти, як кластери пов’язані один з одним на різних

рівнях деталізації, допомагаючи у визначенні значущих кластерів у наборі даних.

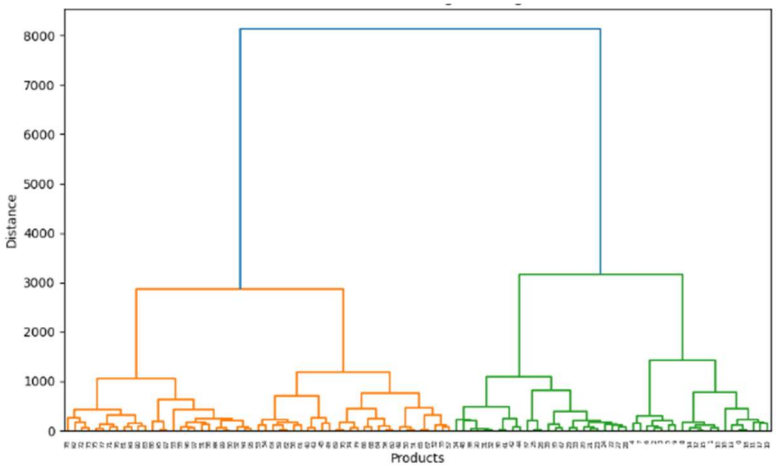

Рисунок 3.5 – Ієрархічна діаграма кластеризації.

Під час розподілу товарів за цим методом, спочатку кожен товар розглядається як окремий кластер, а потім послідовно об'єднується з подібними до нього за характеристиками. На початку кожен товар утворює свій власний кластер.

Після цього проводиться пошук найближчого за схожістю товару або кластера, і вони об'єднуються у більший кластер. Цей процес триває, поки всі товари не об'єднаються в один великий кластер або досягне певної визначеної точки зупинки.

Ці маленькі групи товарів становлять основу для рекомендацій, оскільки товари всередині кожного такого класу схожі за певними параметрами.

Рекомендації можуть будуватися на базі порівняння подібності між товарами у кожному малому кластері, що дозволяє надавати рекомендації для користувачів на основі їхніх виборів і смаків, які відображаються в цих малих групах товарів.

	Product	Similarity
0	MO17E1B	0.00
18	MO17E1W	10.07
17	PMW 20711 KW	126.33
19	MO20E1WH	130.34
11	MW-4001BR	199.42
24	MW-MM-20P(WH)	220.47
34	AMG20M70GSVH	398.25
8	AMGF17M2BH	429.67
45	MW-4010B	612.05
7	Elegance MW-2510W	796.89

Рисунок 3.6 – Рекомендації HDBSCAN.

З першого огляду результати ієрархічної кластеризації гірші аніж двох методів перед ним. Найбільша відстань складає 796.86.

Для визначення кращого методу для надання персоналізованих рекомендацій було обрано збігаються елементи з кількох списків, заснованих на різних алгоритмах кластеризації.

Для порівняння ефективності надання рекомендацій запропонованих методів було вирішено порахувати параметри, такі як: середню суму відстаней, максимальну відстань та підсумкову суму відстаней.

Результати розрахунків задані у табл. 3.1:

Таблиця 3.1

Порівняння результатів надання рекомендацій

Евклід. відстань	K-means++	Mean Shift	HDBSCAN
Максимальна	331.45	382.12	796.89
Середня	210.33	237.5	324.83
Сума	1892.97	2137.57	2923.49

Для тестування часу, було вирішено провести п'ять ітерацій запуску моделі, щоб переконатися, що результат не був випадковим та можна було середній час виконання операцій.

Ці параметри задані у табл. 3.2:

Таблиця 3.2

Час роботи моделі (сек.)

Спроба	K-means++	Mean Shift	HDBSCAN
1	0.86	1.83	0.32
2	0.73	1.79	0.29
3	0.74	1.79	0.33
4	0.73	1.80	0.32
5	0.79	1.81	0.31
Сер. значення	0.77	1.804	0.314

Після декількох запусків програми можна зробити висновок, що найшвидшим алгоритмом є ієрархічний підхід, що може пояснюватися тим, що даний алгоритм працює з маленьким датасетом. Mean Shift може бути менш ефективним, оскільки він та може виявляти складність у визначенні параметрів. K-means++ середні показники в порівнянні з двома попередніми підходами. Отже, результати між декількома запусками одного алгоритму відрізняються на незначну частину, однак найбільша різниця між ітераціями складає 0.09 сек, що може бути пов'язано з першим запуском моделі.

Таблиця 3.3.

Використання оперативної пам'яті (MiB)

Спроба	K-means++	Mean Shift	HDBSCAN
1	1.4	1.5	0.8
2	1.4	1.5	0.9
3	1.5	1.6	0.8
4	1.4	1.6	0.9
5	1.4	1.5	0.8
Сер. значення	1.42	1.54	0.84

МіВ означає мебібайт. Один мебібайт дорівнює 1 048 576 байтам, тоді як мегабайт зазвичай вважається 1 000 000 байт [58].

За результатами експериментів проведених за допомогою Memory profiler за вимірами обсягу пам'яті можна сказати, що найменше споживає пам'яті ієрархічна кластеризація. Кластеризація на основі щільності показує найбільші результати по споживанню пам'яті, що може пояснюватися тим, що даний підхід, який зазвичай використовується для розпізнавання зображень. Кластеризація на основі центроїдів вимагає трохи менше пам'яті, проте все одно більше ніж HDBSCAN.

3.3 Тестування роботи розробленої моделі

Розроблена гібридна модель, поєднує два методи для надання рекомендацій: алгоритм кластеризації K-means++ та пошуковий алгоритм TF-IDF.

Почнемо з алгоритму K-means++. Він дозволяє групувати схожі елементи у кластери за допомогою ітеративного процесу. Кожен кластер має свій центр, а елементи призначаються до найближчого за відстанню центру кластера. Це дозволяє згрупувати схожі об'єкти разом, що є важливим для подальшої рекомендації товарів для вибору.

Друга частина, алгоритм TF-IDF, який враховує важливість термів у тексті. TF-IDF оцінює, наскільки слово важливе для певного документа у колекції документів. Це дозволяє сортувати слова за їхньою значущістю для конкретного елемента та отримувати більш точні рекомендації на основі схожості тексту чи характеристик об'єктів [59].

Для створення гібридної моделі необхідно протестувати роботу модуля з алгоритмом TF-IDF для надання рекомендацій. Відповідно протестуємо роботу цього модуля на тому ж наборі даних. Для цього візьмемо той самий елемент з датасету, тобто мікрохвильову піч.

	Product	Similarity
0	M017E1B	0.00
18	M017E1W	10.07
19	M020E1WH	130.34
22	M017E1S	152.00
46	M020A3W	208.05
20	LMW-2074M	211.43
27	ED-2053B	241.00
28	M017E1BH	251.16
29	LMW-2079M	284.72
30	YC-MS01E-W	305.04

Рисунок 3.7 – Рекомендації TF-IDF.

Для роботи даного методу необхідно конвертувати всі значення в рядковий тип, оскільки даний метод працює з рядковими значеннями. Як результат, отримаємо список рекомендацій, заснованих на схожості з введеним. Варто відзначити, що даний підхід реалізований із зазначенням імені товару, яка за задумом моделі надалі може бути опціональною.

Наступним кроком, нам необхідно об'єднати рекомендації двох підходів, для цього два списки отриманих. Гібридна модель поєднує алгоритм кластеризації K-means++ та алгоритм TF-IDF, після чого вибираються ті елементи, у яких найменша евклідова відстань. Фінальний результат роботи гібридної моделі зображено нижче (рис. 3.8).

	Product	Similarity	Similarity
0	M017E1B	0.00	100.00
18	M017E1W	10.07	98.65
19	M020E1WH	130.34	82.51
22	M017E1S	152.00	79.60
46	M020A3W	208.05	72.08
20	LMW-2074M	211.43	71.63
27	ED-2053B	241.00	67.66
28	M017E1BH	251.16	66.30
29	LMW-2079M	284.72	61.80

Рисунок 3.8 – Рекомендації гібридної моделі.

Проведемо розрахунок показників ефективності. Сума відстаней для 10 елементів дорівнює 1688.19. Середнє арифметичне значення відстані дорівнює

187.57, що є нижчим ніж у використання просто алгоритму кластеризації без підходу з використанням гібридної моделі, тобто це означає, що рекомендації надаються більш ефективно. Такий результат є вищим за результати моделі на базі алгоритмів кластеризації без використання гібридної моделі.

```
microwave = pd.DataFrame({
    'cost': [6000],
    'manufacturer': ['Samsung'],
    'color': ['red'],
    'width': [50],
    'height': [50],
    'depth': [50],
    'power': [1000],
    'max_consumption': [1500],
})
```

Рисунок 3.9 – Рекомендації гібридної моделі.

Перевіримо роботу моделі на прикладі не нормованого вводу, тобто більшого за присутні у датасеті (рис. 3.9). Введені дані перевищують максимальне допустиме значення цін, ширини, висоти, глибини, та інших характеристик, або взагалі відсутні – товару червоного кольору немає. Після запуску моделі отримуємо рекомендації, які повинні мати низький відсоток відповідності.

	Product	Euclidean	Similarity
99	NN-GT261WZPE	1686.17	34.45
98	MGC20130SB	1729.14	32.78
97	R242BK	1738.76	32.41
96	ME81KRW-1	1745.39	32.15
95	NN-E20JWMEPG	1777.22	30.91
94	YC-MS02E-B	1804.75	29.84
91	MO20S4BC	1825.41	29.04
92	LMW-2381MG	1829.22	28.89
90	ME81KRW-2	1842.83	28.36
89	MW2-MM23PF(BK)	1845.72	28.25

Рисунок 3.10 – Рекомендації гібридної моделі.

Отже, найбільший відсоток складає 34.45, що є поганою рекомендацію. З метою надання рекомендацій з високою відповідністю, необхідно обмежити ввід у межах максимальних значень, які наявні у переліку (рис. 3.10).

Поєднуючи ці два методи, створено гібридну систему, яка використовує переваги обох підходів, наприклад покращує обробку категоріальних даних, оскільки модель рекомендації на базі кластеризації використовує кодування всіх не цифрових характеристик, що може знизити ефективність роботи системи.

Кластеризація дозволяє групувати схожі об'єкти, а TF-IDF допомагає врахувати важливість та контекст кожного об'єкта для надання більш точних та персоналізованих рекомендацій.

3.4 Запуск програмного рішення та приклади використання

Розглянемо роботу програмного рішення, у вигляді мобільного застосунку на платформі з операційною системою Android.

Після першого запуску користувач повинен побачить екран вибором товару. У поточний реалізації програма підтримує тільки вибір мікрохвильової печі, тому після завантаження, цей екран буде схований.

Розроблене програмне рішення підтримує можливість зміни кількості елементів рекомендацій, які будуть відображені у разі якщо в заданому діапазоні кількість рекомендацій є недостатньою. За замовчуванням, модель працює з 10 елементами та виводить 10 елементів рекомендацій. максимальну кількість елементів рекомендації.

Рекомендується обмежити кількість відображення, так як елементи сортуються в порядку зменшення, і чим більше буде рекомендацій тим сумарно їх точність буде меншою через неналежні рекомендації. Для протестованого датасету рекомендована максимальна кількість з елементів складає 15 елементів.

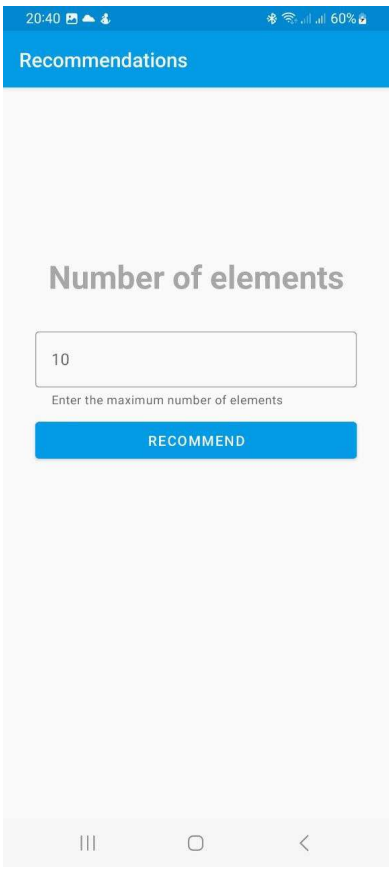

Рисунок 3.11 – Екран вводу кількості елементів.

Додаток використовує дозвіл на роботу з мережею, що є прихованим від користувача, що є стандартною практикою в Android [60].

Для справної роботи, необхідно мати доступ до мережі, щоб модель змогла отримати запит користувача.

Загальний алгоритм для отримання персоналізованих рекомендацій, щодо вибору товару складається з 3 кроків:

1. Вибір кількості товарів
2. Ввід характеристик товару.
3. Вибір товару з переліку рекомендованих.

Наступним, що побачить користувач, буде екран вибору товару, у якому в залежності від кількості характеристик елементу у датасеті, буде відображено пусті поля для вводу.

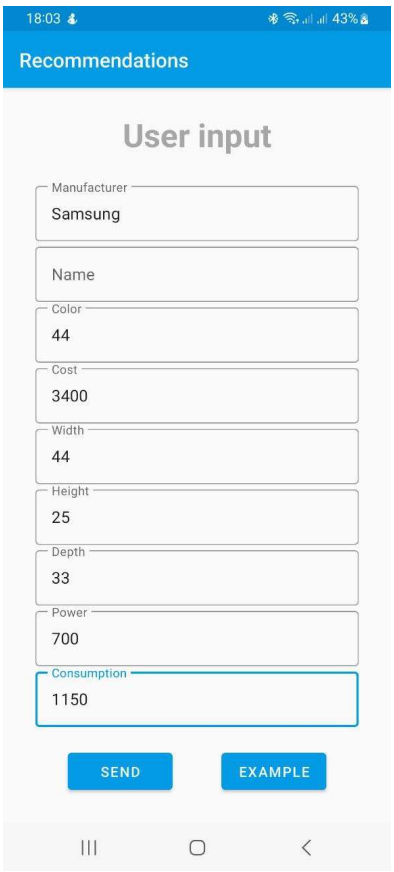

Рисунок 3.12 – Екран вводу запиту.

Користувачу необхідно ввести всі характеристики товару, окрім назви, яка є опціонально, оскільки користувач може не пам'ятати пряму специфікацію моделі товару. Для того, щоб надіслати запит користувачу треба натиснути кнопку Send, після спрацьовування якої буде надіслано GET-запит на сервер, для проведення розрахунків та отримання списку товарів.

Також для зручності, була створена кнопка Example, після натискання якої, поля для вводу будуть автоматично заповненні прикладом (рис. 3.12). Це зроблено для швидкої демонстрації роботи та прикладу формату вводу. Програма оснащена строковим валідатором, який перевіряє відповідність характеристики, яку ввів користувач. Поля повинні бути заповненні за таким форматом: Manufacturer (виробник), Name(назва моделі товару) та Color (колір) приймають тільки строкові значення; Cost (ціна), Power (потужність), Consumption (максимальна споживана потужність) приймають тільки значення цілих чисел; Width(ширина), Height (висота) та Depth (глибина)

можуть приймати, як цілі числа, так і числа з плаваючою точкою, до 3 знаку після точки, тому що інші символи будуть проігноровані, оскільки датасет не містить значень такого типу.

Рисунок 3.13 – Екран рекомендацій.

Після виконання запиту користувача додаток, завантажить список персоналізованих рекомендацій з 10 товарів, за допомогою розробленої гібридної моделі. Список містить назву моделі товару, на округлений відсоток відповідності. Важливо, користувач повинен бачити наскільки відрізняється обраний ним елемент, від того що хоче отримати, з цією метою, модель відображає тільки відсотки відповідності. Перший елемент списку має відповідність 86,5 відсотків, що означає, що датасет не містить введеного елементу. Для отримання детальної інформації про елемент списку, користувач може натиснути на нього, після чого буде завантажено інформацію про опис обраного товару. Додаток підтримує відображення у темному режимі, змінюючи колір екрану з білого на темно-сірий.

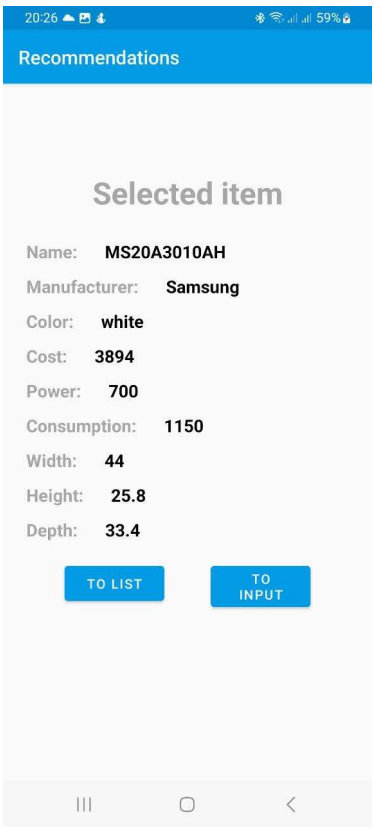

Рисунок 3.14 – Екран з обраним товаром.

Після вибору товару з переліку, користувач побачить екран з характеристиками товару. Також на екрані присутні кнопки для навігації: щоб повернутися назад до списку, щоб повернутися до екрану зі вводом товару

Отже, створена модель запропонувала вибрати покупцю товар, який зображено на рис. 3.14, оскільки ймовірність збігу з введеним елементом становить 86%. Однак, програма надає й інші елементи для того, щоб у користувача була можливість зробити вибір іншого товару, переглянувши його характеристики.

Висновки за розділом 3

Оцінка ефективності алгоритмів кластеризації для системи персоналізованих рекомендацій включає розгляд трьох ключових параметрів: відповідність рекомендацій користувачеві, використання пам'яті та час

виконання запиту. Кроки алгоритму включають кластеризацію, роботу пошукового алгоритму, перевірку вимог користувача та розрахунок відповідності за евклідовою відстанню. Важливим етапом є порівняння рекомендацій різних алгоритмів на основі подібних елементів та їх характеристик. Базуючись на результатах дослідження ефективності алгоритмів кластеризації, для створення гібридної моделі було обрано алгоритм K-Means++. Використання методу ліктя дозволяє визначити кількість кластерів, для ефективної роботи обраного алгоритму. Гібридна модель, поєднуючи алгоритм кластеризації K-means++ та інформований алгоритм пошуку TF-IDF, створює рекомендації на основі схожості об'єктів та їх характеристик. K-means++ дозволяє групувати схожі об'єкти, а TF-IDF враховує важливість термінів у тексті для точніших рекомендацій. Тестування моделі на мікрохвильовій печі показало, що гібридна модель працює ефективніше за алгоритми окремо. Порівняння показників відстаней та відсотків схожості демонструє покращення ефективності рекомендацій гібридної моделі. Однак важливо обмежувати ввідний діапазон для отримання більш точних рекомендацій. У програмному рішенні для мобільного застосунку на платформі Android запропоновано простий та ефективний алгоритм вибору товару, що включає кроки введення характеристик, вибору товару з рекомендаційного списку та надання детальної інформації про обраний товар. Додаток вимагає доступ до мережі для роботи моделі та забезпечення персоналізованих рекомендацій. Його інтерфейс простий та зрозумілий, з автоматичним заповненням полів для швидкого ознайомлення з форматом вводу. Програма також підтримує темний режим для зручності користувача, дозволяючи змінити колір екрану. Такий підхід сприяє зручності користувача та ефективному вибору товару за його характеристиками. Як результат, розроблена модель пропонує користувачеві список персоналізованих рекомендацій для вибору товару ґрунтуючись на введених характеристиках мікрохвильової печі.

ВИСНОВКИ

Під час проведення досліджень щодо створення комп'ютерної моделі персоналізованих рекомендацій товарів на базі пошукового алгоритму та алгоритмів кластеризації, було проведено детальний аналіз характеристик кожного з підходів.

Сукупність результатів, що були отримані у кваліфікаційній роботі дозволила вирішити науково-технічне завдання, що було спрямоване на розроблення гібридної моделі надання персоналізованих рекомендацій, для підвищення ефективності роботи моделі. В результаті проведеного тестування моделі та досліджень було отримано практичні та наукові результати, а саме найкращим підходом для забезпечення кластеризації виявився K-means++, хоча у порівняні з ієрархічною кластеризацію на невеликому об’ємі даних працює повільніше. Для оцінки ефективності надання рекомендацій використовувалось порівняння переліку елементів до введеного масиву характеристик за допомогою розрахунку евклідової відстані. Алгоритм кластеризації на основі центроїд виявися найкращим для цієї задачі, у порівнянні з кластеризацією на основі щільності та ієрархічної кластеризації. Також було протестовано ефективність надання рекомендацій заснованих на інформованому алгоритму пошуку TF-IDF, що виявилися кращими за результати використання тільки моделі на кластеризації.

Підсумовуючи, запропонована гібридна модель дозволяє підвищити ефективність роботи системи завдяки поєднанню їх переваг окремих методів. Підхід кластеризації дозволяє обрати з великого масиву даних кластер на базі якого будуть надаватись рекомендації, та підхід TF-IDF дозволяє краще працювати з категоріальними даними. У перспективі запропонований підхід може бути розширений за допомогою інших алгоритмів надання рекомендацій або різних архітектур гібридних систем і поширений для використання у комерційних цілях у вигляді додатку, чи моделі для уже існуючої системи з метою надання рекомендацій.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Vector Similarity Search Algorithms: [Електронний ресурс], URL: https://medium.com/@serkan_ozal/vector-similarity-search-53ed42b951d9 (дата звернення: 22.03.2023).
2. How Search Engine Personalization Affects Rankings: [Електронний ресурс], URL: https://marketbrew.ai/how-search-engine-personalization-affects-rankings (дата звернення: 22.03.2023).
3. Understanding The Different Types of Search Algorithms: [Електронний ресурс], URL: https://www.luigisbox.com/blog/types-of-search-algorithms/ (дата звернення: 22.03.2023).
4. Improved Search Algorithms: [Електронний ресурс], URL: https://aicontentfy.com/en/blog/impact-of-ai-on-content-consumption-patterns (дата звернення: 22.03.2023).
5. A Guide To Understanding Search Engine Algorithms: [Електронний ресурс], URL: https://seobase.com/what-is-a-search-algorithm/ (дата звернення: 22.03.2023).
6. Uninformed & Informed Search Algorithms: [Електронний ресурс], URL: https://databasetown.com/search-algorithms-in-artificial-intelligence-ai/ (дата звернення: 22.03.2023).
7. Introduction to Artificial Intelligence: [Електронний ресурс], URL: https://www.cs.ucdavis.edu/~vemuri/classes/ecs170/blindsearches_files/blind_searches.htm (дата звернення: 22.03.2023).
8. What is Uninformed Search Algorithm in AI?: [Електронний ресурс], URL: https://www.educative.io/answers/what-is-uninformed-search-algorithm-in-ai (дата звернення: 23.03.2023).
9. Informed Search Techniques (Part — 6): [Електронний ресурс], URL: https://medium.com/@nidhiworah02/informed-search-techniques-part-6-25632f867529 (дата звернення: 22.03.2023).

10. Difference between Informed and Uninformed Search in AI: [Електронний ресурс], URL: https://www.geeksforgeeks.org/difference-between-informed-and-uninformed-search-in-ai/ (дата звернення: 22.03.2023).
11. Greedy Best-First Search: [Електронний ресурс], URL: https://www.codecademy.com/resources/docs/ai/search-algorithms/greedy-best-first-search (дата звернення: 22.03.2023).
12. Building Blocks for AI Part 2: Clustering and Classification: [Електронний ресурс], URL: https://medium.com/@kamalmeet/building-blocks-for-ai-part-2-clustering-and-classification-ce537056b7f9 (дата звернення: 25.03.2023).
13. K-Means Advantages and Disadvantage: [Електронний ресурс], URL: https://developers.google.com/machine-learning/clustering/ (дата звернення: 26.03.2023).
14. Diego, U. Unsupervised Learning: K-Means Clustering. Towards Data Science: [Електронний ресурс], URL: https://towardsdatascience.com/unsupervised-learning-k-means-clustering-27416b95af27 (дата звернення: 26.03.2023).
15. Arthur, D.; Vassilvitskii, S. "k-means++: the advantages of careful seeding". Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms.: книга: Society for Industrial and Applied Mathematics Philadelphia, PA, USA., 2007. С. 1027–1035.
16. Hierarchical Clustering and Density-Based Spatial Clustering of Applications with Noise (DBSCAN): [Електронний ресурс], URL: https://medium.com/mlearning-ai/hierarchical-clustering-and-density-based-spatial-clustering-of-applications-with-noise-dbscan-b8d903095532 (дата звернення: 26.03.2023).
17. The hdbscan Clustering Library: [Електронний ресурс], URL: https://hdbscan.readthedocs.io/en/latest/ (дата звернення: 30.04.2023).
18. Mean-Shift Clustering: A Powerful Technique for Data Analysis with Python: [Електронний ресурс], URL: https://medium.com/@shruti.dhumne/mean-

shift-clustering-a-powerful-technique-for-data-analysis-with-python-f0c26bfb808a (дата звернення: 30.04.2023).

19. Sander, J. Density-Based Clustering. In: Sammut, C., Webb, G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA. 2011. DOI: https://doi.org/10.1007/978-0-387-30164-8_211
20. Damir Demirović, An Implementation of the Mean Shift Algorithm, Image Processing On Line, 9. 2019. С.251–268. DOI: https://doi.org/10.5201/ipol.2019.255
21. Clustering Algorithms: [Електронний ресурс], URL: https://developers.google.com/machine-learning/clustering/clustering-algorithms (дата звернення: 30.04.2023).
22. Gaussian mixture models: [Електронний ресурс], URL: https://scikit-learn.org/stable/modules/mixture.html (дата звернення: 30.04.2023).
23. Gaussian Mixture Models: What are they & when to use?: [Електронний ресурс], URL: https://vitalflux.com/gaussian-mixture-models-what-are-they-when-to-use/ (дата звернення: 30.04.2023).
24. Personalization Glossary, Personalization Engines: [Електронний ресурс], URL: https://www.dynamicyield.com/glossary/personalization-engines/ (дата звернення: 30.04.2023).
25. How YouTube Recommends Videos: [Електронний ресурс], URL: https://www.searchenginejournal.com/how-youtube-recommends-videos/420288/ (дата звернення: 05.05.2023).
26. Inside Spotify's Recommender System: A Complete Guide to Spotify Recommendation Algorithms: [Електронний ресурс], URL: https://www.music-tomorrow.com/blog/how-spotify-recommendation-system-works-a-complete-guide-2022 (дата звернення: 05.05.2023).
27. What Content-Based Filtering is and Why You Should Use It: [Електронний ресурс], URL: https://www.upwork.com/resources/what-is-content-based-filtering (дата звернення: 05.05.2023).

28. Deep Dive into Content-Based Recommender Systems: Unveiling the Power of Attribute-Based Recommendations: [Електронний ресурс], URL: https://neemz.medium.com/deep-dive-into-content-based-recommender-systems-unveiling-the-power-of-attribute-based-48d27fc98812 (дата звернення: 06.06.2023).
29. Understanding Cosine Similarity and Its Applications: [Електронний ресурс], URL: https://builtin.com/machine-learning/cosine-similarity (дата звернення: 06.06.2023).
30. A Guide to Content-Based Filtering In Recommender Systems: [Електронний ресурс], URL: https://www.turing.com/kb/content-based-filtering-in-recommender-systems (дата звернення: 06.06.2023).
31. Introduction to Recommender System: [Електронний ресурс], URL: https://towardsdatascience.com/intro-to-recommender-system-collaborative-filtering-64a238194a26 (дата звернення: 06.06.2023).
32. Collaborative filtering in recommender system: [Електронний ресурс], URL: https://www.turing.com/kb/collaborative-filtering-in-recommender-system#user-based-collaborative-filtering (дата звернення: 07.06.2023).
33. Emerging Concepts in Recommender Systems system: [Електронний ресурс], URL: https://medium.com/@aman.jain.004/emerging-concepts-in-recommender-systems-a169e3504926 (дата звернення: 07.06.2023).
34. Li, L., Chu, W., Langford, J., & Schapire, R. E. (2010). A Contextual-Bandit Approach to Personalized News Article Recommendation. 2010. DOI: https://doi.org/10.1145/1772690.1772758
35. 7 Types of Hybrid Recommendation System: [Електронний ресурс], URL: https://medium.com/analytics-vidhya/7-types-of-hybrid-recommendation-system-3e4f78266ad8 (дата звернення: 14.06.2023).
36. Python (in Machine Learning): [Електронний ресурс], URL: https://corporatefinanceinstitute.com/resources/data-science/python-in-machine-learning/ (дата звернення: 28.06.2023).

37. REST API Tutorial: What is REST?: [Електронний ресурс], URL: https://restfulapi.net/ (дата звернення: 28.06.2023).
38. Python for AI and Machine Learning: [Електронний ресурс], URL: https://skillet.com.my/python-for-ai-and-machine-learning/ (дата звернення: 28.06.2023).
39. Pandas documentation: [Електронний ресурс], URL: https://pandas.pydata.org/ (дата звернення: 28.06.2023).
40. Seaborn: statistical data visualization: [Електронний ресурс], URL: https://seaborn.pydata.org/ (дата звернення: 28.06.2023).
41. Matplotlib: Visualization with Python: [Електронний ресурс], URL: https://matplotlib.org/stable/users/index (дата звернення: 28.06.2023).
42. Memory-profiler: [Електронний ресурс], URL: https://pypi.org/project/memory-profiler/ (дата звернення: 07.08.2023).
43. Flask's documentation: [Електронний ресурс], URL: https://flask.palletsprojects.com/en/3.0.x/ (дата звернення: 07.08.2023).
44. Scikit-learn User Guide: [Електронний ресурс], URL: https://scikit-learn.org/stable/user_guide.html (дата звернення: 07.08.2023).
45. How Many Android Users Are There?: [Електронний ресурс], URL: https://www.bankmycell.com/blog/how-many-android-users-are-there (дата звернення: 01.09.2023).
46. Android 14: [Електронний ресурс], URL: https://developer.android.com/about/versions/14/ (дата звернення: 01.09.2023).
47. Android Open Source Project: [Електронний ресурс], URL: https://source.android.com/ (дата звернення: 01.09.2023).
48. Android's Kotlin-first approach: [Електронний ресурс], URL: https://developer.android.com/kotlin/first (дата звернення: 01.09.2023).
49. Save data in a local database using Room: [Електронний ресурс], URL: https://developer.android.com/training/data-storage/room (дата звернення: 01.09.2023).

50. Hilt Design Overview: [Електронний ресурс], URL: https://dagger.dev/hilt/ (дата звернення: 01.09.2023).
51. Retrofit, a type-safe HTTP client: [Електронний ресурс], URL: https://square.github.io/retrofit/ (дата звернення: 02.09.2023).
52. ViewModel overview: [Електронний ресурс], URL: https://developer.android.com/topic/libraries/architecture/viewmodel (дата звернення: 04.09.2023).
53. Coroutines documentation: [Електронний ресурс], URL: https://kotlinlang.org/docs/coroutines-overview.html (дата звернення: 04.09.2023).
54. Ciarlet, Philippe G., Linear and Nonlinear Functional Analysis with Applications, Society for Industrial and Applied Mathematics, 2013. С. 173, ISBN 978-1-61197-258-0.
55. TF-IDF from scratch in python: [Електронний ресурс], URL: https://towardsdatascience.com/tf-term-frequency-idf-inverse-document-frequency-from-scratch-in-python-6c2b61b78558 (дата звернення: 04.09.2023).
56. Understanding TF-IDF for Machine Learning: [Електронний ресурс], URL: https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/ (дата звернення: 04.09.2023).
57. Euclidean distance score and similarity: [Електронний ресурс], URL: https://stats.stackexchange.com/questions/53068/euclidean-distance-score-and-similarity (дата звернення: 04.09.2023).
58. MiB to MB Conversion: [Електронний ресурс], URL: https://www.gbmb.org/mib-to-mb (дата звернення: 04.09.2023).
59. TF-IDF – Term Frequency-Inverse Document Frequency: [Електронний ресурс], URL: https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/ (дата звернення: 10.09.2023).
60. Connect to the network: [Електронний ресурс], URL: https://developer.android.com/develop/connectivity/network-ops/connecting (дата звернення: 10.09.2023).

ДОДАТКИ

ДОДАТОК А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) **Магістр**
Галузь знань: **15 – Автоматизація та приладобудування**
Спеціальність: **151 – «Автоматизація та комп'ютерно-інтегровані технології»**

ЗАТВЕРДЖУЮ
Завідувач кафедри теоретичної та прикладної системотехніки
_____д.т.н., проф. Шматков С. І.

«08 » грудня 2022 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЕКТ)

Ясінського Ярослава Андрійовича
(прізвище, ім'я, по батькові студента)

1. Тема роботи: «Комп'ютерна модель персоналізованих рекомендацій вибору товарів на базі пошукових алгоритмів»

керівник роботи Толстолузька Олена Геннадіївна, д. т. н., с.н.с.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «10» листопада 2023 року № 4101-5/3197

2. Строк подання студентом роботи 28 .11.2023

3. Перелік питань, які потрібно розробити:
1. Дослідження методів персоналізації на базі алгоритмів кластеризації для вирішення задачі надання персоналізованого вибору товарів.
2. Аналіз проблем систем рекомендації виробу на базі алгоритмів кластеризації.
3. Створення дата-сету з набором товарів.
4. Програмна реалізація моделі на платформі додатку Android.
5. Тестування та дослідження ефективності роботи комп'ютеризованої моделі.
6. Складання звіту та надання рекомендацій.

4. План роботи:

№ з/п	Назви етапів роботи	Термін виконання виконання етапів роботи
1	Аналіз предметної області та постановка задачі	8.12.2022 – 24.01.2023
2	Дослідження алгоритмів кластеризації для вирішення задач надання персоналізованих рекомендацій	24.01.2023 – 30.04.2023
3	Аналіз проблем систем рекомендації виробу на базі алгоритмів кластеризації	30.04.2023 – 23.05.2023
4	Аналіз можливостей удосконалення алгоритму кластеризації та механізмів надання персоналізованих рекомендацій вибору	23.05.2023 – 20.06.2023
5	Побудова тестового дата-сету товарів	20.06.2023 – 04.07.2023
6	Розробка моделі на платформі Android з реалізацію вибору товару	04.07.2023 – 05.09.2023
7	Порівняння результатів роботи моделі	05.09.2023 – 18.09.2023
8	Доопрацювання та тестування розробленої моделі	18.09.2023 – 02.10.2023
9	Розробка рекомендацій щодо застосування комп'ютеризованої моделі	02.10.2023 – 24.10.2023
10	Підготовка до попереднього захисту кваліфікаційної роботи	24.10.2023 – 14.11.2023
11	Оформлення пояснювальної записки	14.11.2023 – 28.11.2023

5. Дата видачі завдання 08.12.2022

Студент _____ **Ясінський Я. А.**
підпис ініціали, прізвище

Керівник роботи _____ **Толстолузька О. Г.**
підпис ініціали, прізвище

ДОДАТОК Б

ІНДИВІДУАЛЬНЕ ТЕХНІЧНЕ ЗАВДАННЯ

Технічне завдання

на розробку програмного виробу

«Комп'ютерна модель персоналізованих рекомендацій вибору товарів на базі пошукових алгоритмів»

Назва розділу	Назва і зміст підрозділу
1. Введення	1.1. Назва програмного виробу: Комп'ютерна модель персоналізованих рекомендацій вибору товарів на базі пошукових алгоритмів. 1.2. Галузь застосування: комп'ютерні технології.
2. Підстава для розробки	2.1. Навчальний план за спеціальністю 151 - «Автоматизація та комп'ютерно-інтегровані технології» 2.2. Завдання на кваліфікаційну роботу магістра затверджено наказом ректора №4101-5/3197 від «10» листопада 2023 року.
3.Призначення розробки	3.1. Мета розробки програмного виробу: підвищення ефективності надання рекомендацій вибору товарів, за рахунок використання комп'ютерної моделі на базі пошукових алгоритмів та алгоритмів кластеризації. Комп'ютерна модель буде використовувати гібридний підхід для надання рекомендацій вибору. 3.2. Призначення програмного виробу: надання персоналізованих рекомендацій вибору товарів. 3.3. Вихідні дані для розробки: підходи для побудови гібридних моделей надання персоналізованих рекомендацій.

<table>
<tr><td>4. Технічні вимоги до програмного виробу</td><td colspan="2">4.1. Вимоги до функціональних характеристик: можливість надавати рекомендації товарів у мобільному додатку.
4.2. Вимоги до надійності: забезпечити користувачам безпечне під’єднання зі стороннього API.
4.3. Вимоги до умов експлуатації: встановлення додатку на систему Android.
4.4. Вимоги до складу і параметрів технічних засобів: Мобільний пристрій Android з виходом у мережу.
4.5. Вимоги до інформаційної та програмної сумісності: Android OS 5.0 – мінімум.</td></tr>
<tr><td>5. Вимоги до програмної документації.</td><td colspan="2">Програмною документацією до виробу «Комп’ютерна модель персоналізованих рекомендацій вибору товарів на базі пошукових алгоритмів» вважати:
1) Справжнє Технічне завдання на розробку програмного виробу (представити у вигляді Додатку Б до пояснювальної записки до кваліфікаційної роботи).
2) Програму і методику випробувань розробленого програмного виробу (представити у вигляді Додатку В до пояснювальної записки до кваліфікаційної роботи).
3) Опис програмного виробу (представити в розділах 2, 3 пояснювальної записки до кваліфікаційної роботи).
4) Текст програми (представити в Додатку Г до пояснювальної записки до кваліфікаційної роботи).</td></tr>
<tr><td>6. Техніко економічні показники</td><td colspan="2">Орієнтовна оцінка термінів розробки комп’ютерної моделі складає 5 місяців. Витрати грошових коштів відсутні.</td></tr>
<tr><td rowspan="2">7. Стадії і етапи розробки</td><td>Дата</td><td>Назва етапу</td></tr>
<tr><td>08.12.2022 - 05.03.2023

05.03.2023 - 01.04.2023</td><td>1. Аналіз предметної області та постановка задачі.
2. Дослідження алгоритмів</td></tr>
</table>

	01.04.2023 - 29.07.2023 17.07.2023 - 23.08.2023 24.08.2022 - 28.11.2023	персоналізованих рекомендацій 3. Аналіз проблем систем рекомендації виробу на базі пошукових алгоритмів та алгоритмів кластеризації. 4. Побудова, та дослідження тестового дата-сету товарів. 5. Розробка моделі. Порівняння результатів роботи. моделі Розробка рекомендацій щодо застосування комп'ютерної моделі.
8. Порядок контролю і приймання	В даному розділі повинні бути вказані загальні вимоги до приймання розробленого програмного виробу наприклад: 1) Перевірка ходу розробки програмного виробу. Керівнику робіт виконувати 1 раз в 3 тижні. 2) Випробування програмного виробу відповідно до Програми і методики випробувань провести на базі комп'ютерного класу або приватного приміщення. 3) Захист розробленого програмного виробу провести на засіданні атестаційної комісії. 4) Пояснювальну записку надати на паперових носіях в одному примірнику, в електронному вигляді.	

Виконавець:
студент групи КУ-61
Ясінський Я. А.

Замовник:
д.т.н., професор
Толстолузька О. Г.

ДОДАТОК В

Програма і методика випробувань програмного виробу

«Комп’ютерна модель персоналізованих рекомендацій вибору товарів на базі пошукових алгоритмів»

1. Об'єкт випробувань

1. Назва програмного виробу: «Комп’ютерна модель персоналізованих рекомендацій вибору товарів на базі пошукових алгоритмів».
2. Галузь застосування: комп’ютерні технології.
3. Перераховані відомості запозичуються з відповідних розділів Технічного завдання.

2. Мета випробувань

Перевірка відповідності функціональності програмної реалізації комп’ютерної моделі заявленим функціональним можливостям в технічному завданні (Додаток Б до пояснювальної записки до кваліфікаційної роботи).

3. Загальні положення

3.1. Підстави для проведення випробувань

Підставою для проведення випробувань є наказ про призначення атестаційної комісії.

3.2. Місце і тривалість випробувань

Приймальні (приймально-здавальні) випробування проводяться на базі комп'ютерного класу кафедри або приватного приміщення в період роботи атестаційної комісії.

3.3. Обсяг випробувань

Приймальні випробування програмного вироби проводяться в обсязі відповідному цієї програми і методики випробувань.

3.4. Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

4. Вимоги до програми або програмного виробу

Модель повинна задовольняти наступним вимогам:

- працювати на операційній системі Android;
- мінімальна версія підтримуваної операційної системи Android 5.0;
- версія операційної системи під яку розробляться модель Android 14;
- вимоги до надійності;
- передбачити захист від некоректних дій користувача;
- бути легко розширюваною;
- доступ у мережу інтернет;
- елементи програми повинні бути ізольовані одне від одного для зменшення їх впливу на роботи програми під час редагування програмного коду;
- вимоги до складу і параметрів технічних засобів;
- вимоги до маркування та упаковки (не висуваються);
- вимоги до транспортування і зберігання (не висуваються) ;
- Спеціальні вимоги (не пред’являються).

Вимоги до програмної документації

Програмною документацією до виробу «Комп’ютерна модель персоналізованих рекомендацій вибору товарів на базі пошукових алгоритмів» вважати:

- справжнє технічне завдання на розробку програми (представити як Додаток Б до пояснювальної записки до кваліфікаційної роботи);

– програму і методику випробувань розробленої програми (представити як Додаток В до пояснювальної записки до кваліфікаційної роботи);

– рекомендацій щодо застосування створеної програмної стандартизації у проектах (представити в Розділі 3 пояснювальної записки до кваліфікаційної роботи);

– текст програми (представити як Додаток Г до пояснювальної записки до кваліфікаційної роботи).

6. Засоби і порядок випробувань

6.1. Засоби випробувань

Для проведення випробувань необхідний проект для розробки програмної моделі за допомогою мови програмування Python, з використання бібліотеки scikit-learn, та мови програмування Kotlin.

6.2. Порядок проведення випробувань

Як правило, випробування проводяться в два етапи:

- ознайомчий (1-й етап);

- випробування програмного виробу (2-й етап).

Перелік перевірок, що проводяться на 1 етапі випробувань, включає в себе:

1. Перевірку комплектності програмної документації.

2. Перевірка комплектності складу програмної документації здійснюється за критерієм наявності зазначеної в ТЗ документації.

3. Перевірку комплектності складу технічних і програмних засобів.

4. Методику проведення перевірок на 1 етапі випробувань.

5. Якість програмної документації перевіряється на відповідність вимогам стандартів ЕСПД.

Перелік перевірок, що проводяться на 2 етапі випробувань, включає в себе:

1. Перевірку відповідності технічних характеристик програми вимогам технічного завдання.

2. Перевірку ступеня виконання функціональних вимог до програми.

3. Методику проведення перевірок, що входять до переліку по 2 етапу випробувань.

1. Програма працює відповідно до умов експлуатації операційних систем Android.

2. Для роботи необхідна мова програмування Python, версії не нижче 3.9.

3. Для роботи необхідна мова програмування Kotlin, версії не нижче 1.9.21.

4. Порядок проведення випробувань:

4.1. Спочатку потрібно запустити PyCharm, і запустити код на Python, натиснувши кнопку запуску.

4.2. Тестування моделі необхідно задати характеристики товару.

4.3. Провести випробування обраного алгоритму кластеризації.

4.4. Провести тестування роботи валідатора за допомогою Android Studio.

4.5. Провести тестування роботи гібридної моделі.

Для проведення випробувань пропонується провести тест 1, тест 2 та тест 3.

Тест 1

1. Задати в програмі опис характеристик товару.

2. Натискання на кнопку запуску.

3. Отримання результатів у вигляді списку товарів з розрахунком евклідової відстані до заданого елементу.

	Product	Similarity
0	M017E1B	0.00
18	M017E1W	10.07
19	M020E1WH	130.34
22	M017E1S	152.00
11	MW-4001BR	199.42
20	LMW-2074M	211.43
28	M017E1BH	251.16
29	LMW-2079M	284.72
31	R200BKW	322.38
14	PMW 20711 KB	331.45

Рисунок В.1 – Тест 1

Тест 2

1. Запуск додатку на операційній системі Android.

2. Ввід невалідної інформації до поля з номером елементів.

3. Натискання на кнопку Recommend.

3. Отримання результату у вигляді помилки валідації.

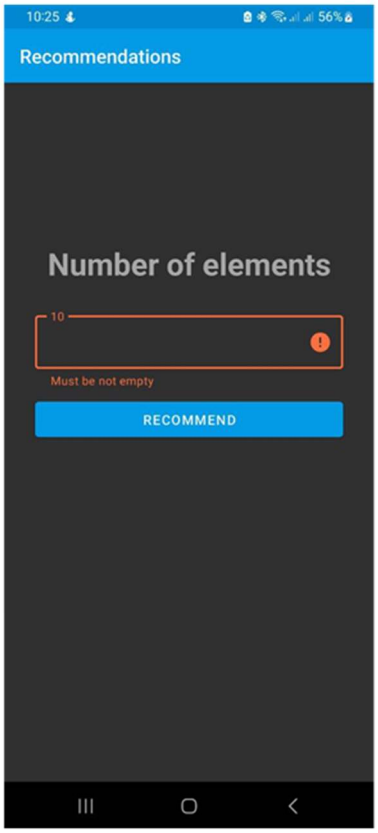

Рисунок В.2 – Тест 2

Тест 3

1. Ввід розміру масиву рекомендацій, який дорівнює 10.

2. Запуск алгоритму надання рекомендацій.

3. Отримання результатів виконання алгоритму на екрані з відображенням відсотку відповідності рекомендації.

Рисунок В.3 – Тест 3

Тест вважається пройденим, якщо відбуваються вказані операції і їх відображення у програмному продукті.

Висновки: випробування пройшло успішно, оскільки кожен з тестів показав очікуванні результати.

Виконавець:

студент групи КУ-61, Ясінський Я. А.

ДОДАТОК Г

Лістинг програмного коду

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn import metrics
from sklearn.cluster import KMeans, AgglomerativeClustering, MeanShift
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
from sklearn.cluster._hdbscan import hdbscan
from sklearn.metrics import davies_bouldin_score
from sklearn.metrics.pairwise import cosine_similarity
from memory_profiler import profile
import time


microwave_data = pd.read_csv('dataset/Book1.csv')

microwave = pd.DataFrame({
    'cost': [2689],
    'width': [45.5],
    'height': [26.1],
    'depth': [35.3],
    'power': [700],
    'max_consumption': [1150]
})

# Вибір числових колонок для порівняння
numeric_cols = [
    'cost',
    'width',
    'height',
    'depth',
    'power',
    'max_consumption',
]


class Stopwatch:
    def __init__(self):
        # Ініціалізація початкового часу
        self.start_time = None

    def start(self):
        # Запуск лічильника часу при старті
        self.start_time = time.time()

    def stop(self):
        # Перевірка, чи був запущений лічильник
        if self.start_time is None:
            # Викидання помилки, якщо лічильник не був запущений
            raise ValueError("Stopwatch has not been started.")
        # Обчислення пройденого часу
        elapsed_time = time.time() - self.start_time
        # Скидання значення початкового часу
        self.start_time = None

        # Повернення пройденого часу
        return elapsed_time
```

```
# Метод ліктя для вибору найкращої кількості кластерів
def find_best_clusters(maximum_k):
    # Зберігання центрів кластерів
    clusters_centers = []
    # Зберігання значень k (кількість кластерів)
    k_values = []  # Зберігання значень k (кількість кластерів)

    # Перебір можливих значень k
    for k in range(1, maximum_k):
        # Створення моделі KMeans з поточним значенням k
        kmeans_model = KMeans(n_clusters=k)
        # Навчання моделі до даних
        kmeans_model.fit(microwave_data[numeric_cols])

        # Збереження внутрішньокластерної суми квадратів відстаней
        clusters_centers.append(kmeans_model.inertia_)
        # Додавання поточного значення k
        k_values.append(k)

    # Виведення внутрішньокластерних сум квадратів відстаней
    print(clusters_centers)
    # Виведення значень k
    print(k_values)
    # Виклик функції для побудови графіка методу ліктя
    generate_elbow_plot(clusters_centers, k_values)
    # Повернення списку внутрішньокластерних сум та списку k
    return clusters_centers, k_values


@profile
# Kmeans++
def kmeans_recomendation_engine(user_input, num_clusters):
    # Ініціалізація та навчання моделі K-Means
    # kmeans = KMeans(n_clusters=num_clusters)

    # Ініціалізація та навчання моделі K-Means++
    kmeans = KMeans(num_clusters, init='k-means++', random_state=42)
    kmeans.fit(microwave_data[numeric_cols])  # Вибір параметрів для
кластеризації

    labels = kmeans.labels_  # Призначення товарів до кластерів
    user_cluster = kmeans.predict(user_input)  # Визначення кластера
користувача

    # Знаходження інших товарів у тому ж кластері
    return microwave_data.iloc[labels == user_cluster[0]]



@profile
# Mean Shift
def mean_shift_recomendation_engine(user_input):

    # Ініціалізація та навчання моделі кластеризації Mean Shift
    meanshift = MeanShift()
    # Вибір параметрів для кластеризації
    meanshift.fit(microwave_data[numeric_cols])

    # Призначення продуктів кластерам
    labels = meanshift.labels_

    # Приклад введення користувача та отримання рекомендацій
    # Визначення кластера користувача
    user_cluster = meanshift.predict(user_input)
```

```
# Функція для обчислення відсотка подібності за евклідовою відстанню
def calculate_euclidean_distance_percentage(user_input, similar_items):
    # Створення масиву з числових значень користувацького вводу
    user_vector = np.array(user_input[numeric_cols])

    # Створення масиву числових значень для подібних об'єктів
    similar_vectors = np.array(similar_items[numeric_cols])

    # Обчислення евклідових відстаней між векторами
    euclidean_distances = np.linalg.norm(similar_vectors - user_vector,
axis=1)

    # Знаходження максимальної відстані
    max_distance = np.max(euclidean_distances)

    # Обчислення відсотка подібності на основі відстаней
    similarity_percentages = 1 - (euclidean_distances / max_distance)

    # Переведення відсотка подібності в масштаб від 0 до 100
    similarity_percentages *= 100

    # Повернення відсотків подібності
    return similarity_percentages


def get_recommendations(microwave, data):
    # Об'єднання відповідних атрибутів в єдиний опис для TF-IDF
    microwave['description'] = microwave['manufacturer'] + ' ' +
microwave['name'] + ' ' + microwave['color']

    # Об'єднання наявних атрибутів даних в опис для TF-IDF
    data['description'] = data['manufacturer'] + ' ' + data['name'] + ' ' +
data['color']

    # Створення векторизатору TF-IDF
    tfidf = TfidfVectorizer(stop_words='english')

    # Навчання та трансформування даних опису
    tfidf_matrix = tfidf.fit_transform(data['description'])

    # Обчислити косинусну подібність між кожним описом продукту
    cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)

    # Представлення TF-IDF для нових мікрохвильових даних
    new_tfidf_matrix = tfidf.transform(microwave['description'])

    # Обчислити косинусну подібність між новою мікрохвильовою піччю та
існуючими продуктами
    similarity_scores = linear_kernel(new_tfidf_matrix, tfidf_matrix)

    # індекси продуктів з найвищими показниками подібності
    similar_indices = similarity_scores.argsort().flatten()[::-1]

    # Рекомендації на основі балів подібності
    # Кількість рекомендацій для надання
    top_n = 10
    recommended_products = data.iloc[similar_indices][:top_n]

    return recommended_products
```

```
def calculate_metrics(X, labels):
    metrics.silhouette_score(X, labels)
    metrics.calinski_harabasz_score(X, labels)
    davies_bouldin_score(X, labels)

    print(metrics.silhouette_score(X, labels))
    print(metrics.calinski_harabasz_score(X, labels))
    print(davies_bouldin_score(X, labels))


def print_similarities(similarities, recomendation_list):
    # Знаходження максимальної відстані серед подібностей
    max_distance = np.max(similarities)

    # Обчислення відсотків подібності на основі відстаней
    similarity_percentages = 1 - (similarities / max_distance)
    similarity_percentages *= 100

    # Округлення значень подібностей і відсотків подібності до 2 знаків після
коми
    similarities = similarities.round(2)
    similarity_percentages = similarity_percentages.round(2)

    # Створення DataFrame для виведення результатів
    similarities_df = pd.DataFrame({
        'Product': recomendation_list['name'],  # Назви продуктів
        'Euclidean': similarities,  # Значення евклідової відстані
        'Similarity': similarity_percentages,  # Відсоток подібності
    })

    # Сортування результатів за спаданням відсотку подібності
    similarities_df = similarities_df.sort_values(by='Similarity',
ascending=False)

    # Виведення результатів
    print(similarities_df)


def plot_hierarchical_dendrogram():
    # Видобуття числових стовпців з даних про мікрохвильові печі
    user_input = microwave_data[numeric_cols]

    # Ієрархічна кластеризація
    # Обчислення матриці зв'язків
    linkage_matrix = linkage(user_input, method='ward')

    # Побудова дендрограми
    # Задання розмірів фігури для дендрограми
    plt.figure(figsize=(10, 6))
    # Побудова дендрограми з використанням матриці зв'язків
    dendrogram(linkage_matrix)
    # Заголовок дендрограми
    plt.title('Hierarchical Clustering Dendrogram')
    # Підпис вісі x
    plt.xlabel('Products')
    # Підпис вісі y
    plt.ylabel('Distance')
```

```
def plot_3d():
    # Створення нової фігури для графіку
    fig = plt.figure(figsize=(8, 6))

    # Додавання тривимірних координат до графіку
    ax = fig.add_subplot(111, projection='3d')

    # Розміщення точок у тривимірному просторі на основі даних про ширину,
висоту та глибину
    ax.scatter(microwave_data['width'], microwave_data['height'],
microwave_data['depth'], c='blue', marker='o')

    # Підписи осей
    ax.set_xlabel('Width')
    ax.set_ylabel('Height')
    ax.set_zlabel('Depth')

    # Заголовок графіку
    plt.title('Тривимірна точкова діаграма ширини, висоти та глибини')

    # Відображення графіку
    plt.show()


# Гістограма розподілу вартості
def plot_cost_distribution():
    # Створення нової фігури для графіку
    plt.figure(figsize=(8, 6))

    # Побудова гістограми на основі даних про вартість з використанням 10 бінів
    plt.hist(microwave_data['cost'], bins=10, color='blue', alpha=0.7)

    # Підписи осей
    # Вісь x - вартість
    plt.xlabel('Cost')
    # Вісь y - частота
    plt.ylabel('Frequency')
    # Заголовок графіку - Розподіл цін
    plt.title('Розподіл цін')

    # Відображення гістограми
    plt.show()


# Створення скрипкового графіку за допомогою бібліотеки seaborn
def cost_violin_plot():
    # Створення нової фігури для графіку
    plt.figure(figsize=(8, 6))

    # Побудова скрипкового графіку для візуалізації розподілу вартості продуктів
по вертикалі
    sns.violinplot(y=microwave_data['cost'], color='skyblue')

    # Встановлення заголовка графіку
    plt.title('Розподіл цін')

    # Підпис вісі y - вартість
    plt.ylabel('Cost')
```