Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра математики факультету інформатики

Кваліфікаційна робота

За спеціальністю 113 Прикладна математика

освітній ступінь – бакалавр

на тему: **«МОДЕЛЬ ТОРГІВ В APPLE SEARCH ADS**

З АУКЦІОНОМ ДРУГОГО БІДА»

Виконала: студент 4-го року навчання,

Освітньої програми «Прикладна математика», 113

Журавльова Анастасія Дмитрівна

Керівник Дрінь С. С,_____

кандидат фіз.-мат. наук, доцент, Старший викладач

Рецензент _____

Кваліфікаційна робота захищена

з оцінкою _____

Секретар ЕК _____

«____» _____ 20____ р.

Київ – 2023

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Зав.кафедри математики,
проф., д.ф.-м.н.
_____Р. К. Чорней
(підпис)
«___»_____2023 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на кваліфікаційну роботу

студентці Журавльовій Анастасії Дмитрівні факультету інформатики 4 курсу

ТЕМА Модель торгів в Apple Search Ads з аукціоном другого біта
Зміст ТЧ до курсової роботи:

Індивідуальне завдання
Зміст
Анотація
Вступ
1 Загальна структура Apple Search Ads
2 Модель аукціону другої ціни в Apple Search Ads
3 Рекомендаційні системи та типи фільтрацій
4 Алгоритми класифікацій
5 Практична реалізація
Висновки
Список літератури
Додатки
Дата видачі „___” _____ 2022р. Керівник _____
(підпис)
Завдання отримав _____
(підпис)

ЗМІСТ

Тема: Дослідження моделі торгів в Apple Search Ads

Календарний план виконання роботи:

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на кваліфікаційну роботу	Жовтень	
2.	Обговорення деталей теми на розробка плану виконання	Жовтень-Листопад	
3.	Огляд технічної літератури за темою роботи та опрацювання матеріалів	Жовтень-Січень	
4.	Написання теоретичної частини роботи	Березень-Квітень	
5.	Розробка практичної частини та фіналізація практичної частини роботи	Квітень	
6.	Попередній захист кваліфікаційної роботи	Травень	
7.	Захист кваліфікаційної роботи	Травень-Червень	

Студент: Журавльова Анастасія Дмитрівна

Керівник: Дрінь Світлана Сергіївна

“__” _____ 2023

АНОТАЦІЯ

У цій кваліфікаційній роботі розглядається загальна структура та основні показники системи Apple Search Ads. Дану тему було замовлено українською IT-компанією для проведення дослідженння моделі торгів другої ціни, яка реалізована в цій пошуковій системі та пошуку можливості реалізації рекомендаційної системи на основі даної моделі. В роботі розглядається принцип призначення ставок другої ціни та те, як він використовується в системі Apple Search Ads. Далі, проведено дослідження систем рекомендацій. Вивчається загальна структура цих систем і порівнюються різні вимірювання подібності. У наступному розділі розглядаються алгоритми фільтрації, зокрема алгоритм k-найближчих сусідів та алгоритм випадкового лісу. В останньому розділі приведено практичну реалізацію моделі торгів другого біда на основі вище зазначених методів.

Результати дослідження дозволяють отримати глибше розуміння процесу рекомендацій та моделі аукціону другої ціни в системі Apple Search Ads. Робота може бути використана для покращення ефективності рекламних кампаній та оптимізації використання рекомендаційних систем у контексті пошукової реклами.

Ключові слова:

Apple Search Ads, аукціон другої ціни, рекомендаційні системи, спільна фільтрація, метод k-найближчих сусідів, метод випадкового лісу, Python

ВСТУП

В сучасному світі різні моделі торгів активно впроваджуються в різних сервісах, наприклад, в пошукових системах, різноманітних аукціонах. Моделі торгів тісно пов’язанні з рекомендаційними системами. Дані системи широко впровадженні в великих всесвітніх компаніях, як Google, Amazon, Netflix, YouTube, Meta, так і в українських компаніях, як Rozetka, MakeUp, Megogo тощо. Відома українська IT-компанія «Genesis» також зацікавилась можливостями використання рекомендаційних систем в їх бізнесі. Вони замовили тему цієї кваліфікаційної роботи для дослідження. Алгоритми рекомендаційних системи постійно розвиваються і вдосконалюються. На сьогоднішній день дослідження і впровадження нових систем це великий і важливий розділ машинного навчання. Багато продуктових IT компаній мають на меті першочергово побудувати свій бренд, своє впізнання серед людей, свій імідж. Одним з шляхів для досягання цієї мети для продуктових компаній зі своїм застосунком – рекламування та платне просування на різних платформах. Найефективніше це робити зокрема в App Store, Google Play Market тощо, де користувачі безпосередньо шукають підходящий додаток. І починати дослідження цього ринку треба починати з системи розміщення. В подальшій своїй роботі я досліджу використання рекомендаційних систем в рамках Apple Search Ads. Тема моєї кваліфікаційної роботи була замовлена українською IT компанією. Вони дали завдання: дослідити і порівняти різні моделі торгів на вхідному датасеті. Отже, метою моєї роботи є порівняння різних видів торгів для Apple Search Ads та провести дослідження чи підходять наявні методи побудов інших рекомендаційних систем для моделі торгів в Apple Search Ads. Дана тема описує використання вже відомих алгоритмів фільтрації та класифікації в нових сферах, таких як маркетинг, економіка і бізнес для збору інформації, аналізу даних, моделювання та прогнозування, що піходить для спеціальності «Прикладна математика».

РОЗДІЛ 1. ЗАГАЛЬНА СТРУКТУРА Apple Search Ads

Apple Search Ads це відносно нова рекламна платформа. Вона була запущена в 2016 році, для того щоб розробники могли легко та ефективно просувати свої застосунки. Завдяки цій системі вони можуть зробити так, щоб їхній додаток мав найвищу позицію в пошуку. Загальний потенціал охоплення в App Store нараховує зараз 650 млн користувачів за даними Apple Inc [1].

1.1 Загальний опис Apple Search Ads

Існує декілька варіантів розміщень рекламних оголошень в Apple Search Ads [1]:

- **Вкладка «Сьогодні»** - реклама на цій вкладці розміщується у вигляді великого банеру, що допомагає досягти максимального рівня помітності вашого застосунку.

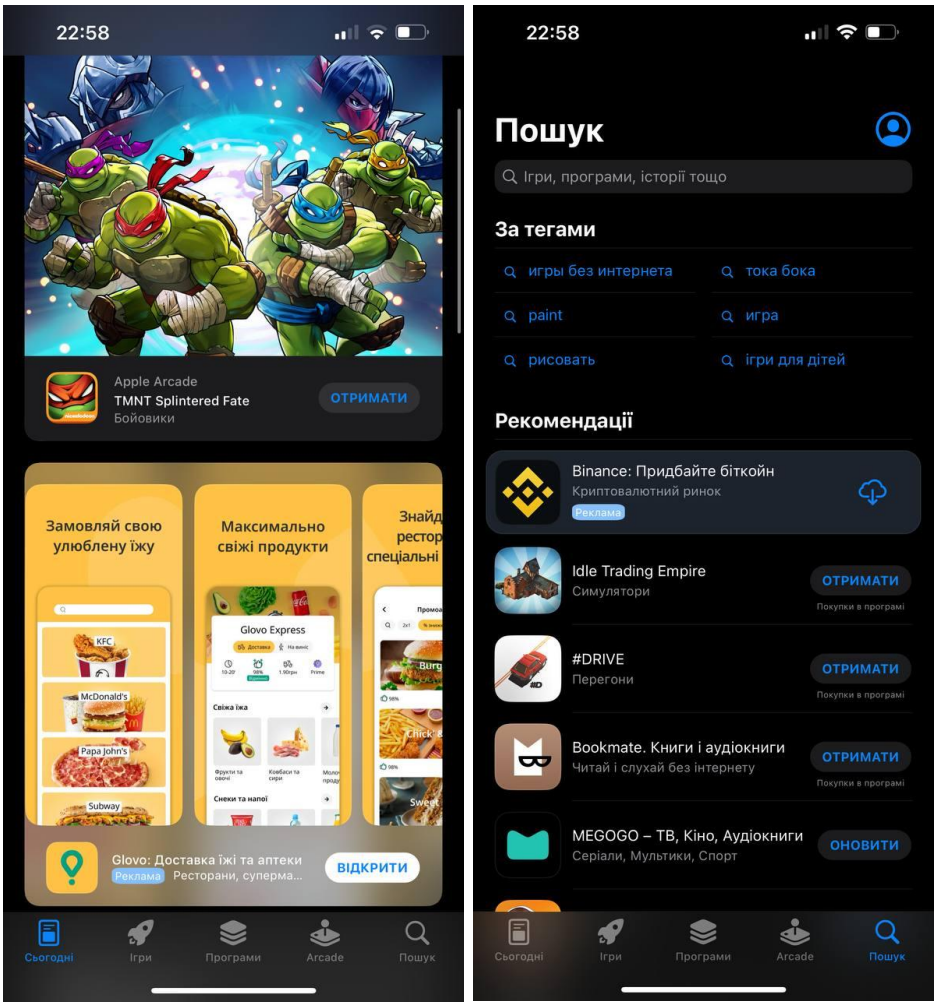


Рисунок 1.1 Вкладка «Сьогодні» і Вкладка «Пошук»

- **Вкладка «Пошук»** - це основна вкладка в App Store. Саме нею користуються 70% користувачів, щоб завантажити певний додаток. Це гарне місце для розміщення реклами на видному місці, перед безпосередньо потенційним завантаженням додатка користувачем

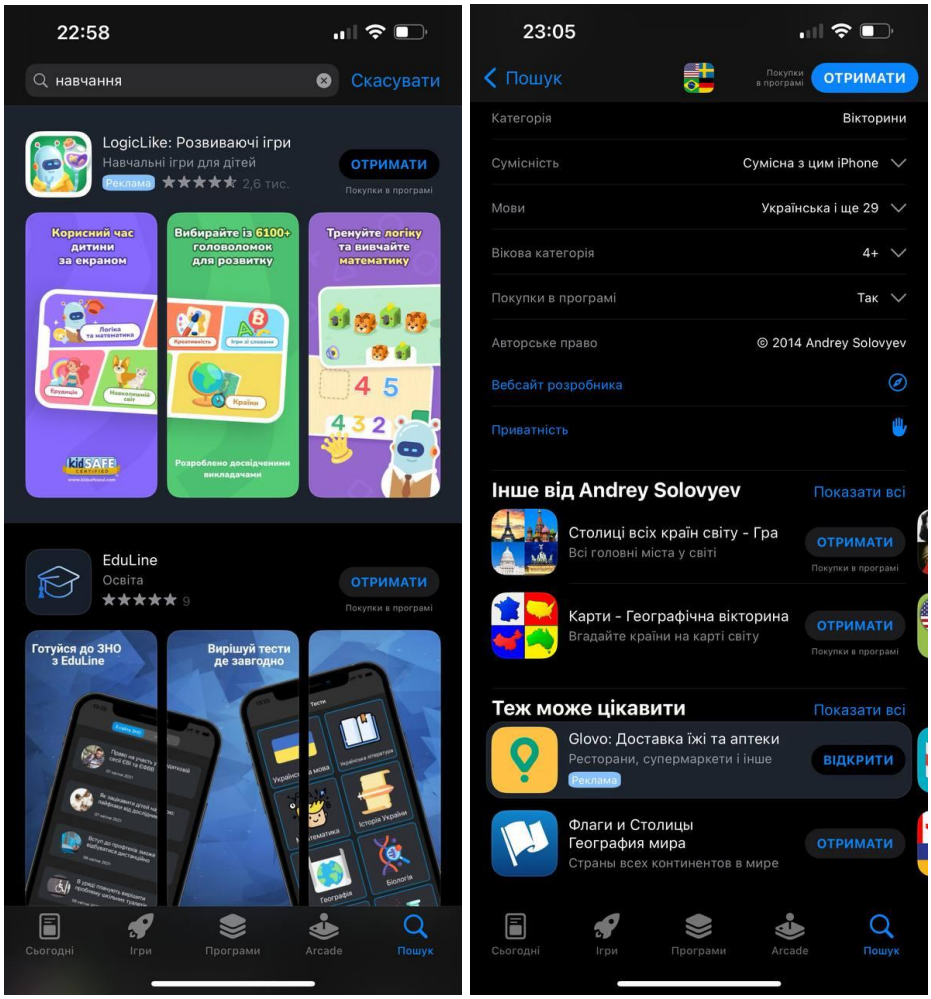


Рисунок 1.2 Вкладка «Пошук» і результати пошуку

- **Результати пошуку** – місце, звідки 65% користувачів завантажують застосунки. Розміщуючись на першій позиції пошуку, ви зможете привернути увагу потенційних, а головне релевантних клієнтів, які вже мають намір обрати, щось конкретне. Просування в такому розміщенні відбувається за допомогою ключових слів, за якими користувачі будуть шукати додаток.
- **Продуктова сторінка** – це безпосередньо опис самого застосунку. Коли користувач, прочитавши опис і зрозумівши, що це не зовсім те, що

хотілось, він побачить рекламу вашого застосунку як потенційного схожого додатку.

Структура рекламного кабінету Apple Search Ads виглядає наступним чином:

1. Група кампаній, які створюються для просування застосунку.
2. Рекламна кампанія – на цьому рівні можна виставляти таргетинг на певну географію, а також регулювати статус (активна, неактивна рекламна кампанія).
3. Група оголошень – можна додати додаткові таргетинги, а саме на пристрої, на вік та стать, спеціальний тип аудиторії та багато інших.
4. Ключові слова – останній і найважливіший етап цієї, адже за допомогою них буде з’являтись рекламне оголошення

Існує чотири види ключових слів [2]:

1. Точні ключові слова – це слова, які повністю відповідають пошуковому запиту користувача. Також вони включають форму множини і граматичні помилки в пошукових запитах. Через відсутність варіантів інших інтерпретацій, ці слова зазвичай мають меншу кількість показів, проте доля цільових дій, а саме завантажень, є великою відносно інших видів ключових слів
2. Широкі ключові слова – вони допускають використання синонімів, припущення граматичних помилок, вживання близьких за змістом пошукових запитів. Такий тип ключових слів/словосполучень дає можливість знаходити нові потенційні ключові слова серед пошукових запитів користувачів. Найкращі ключові слова слід переводити в точний типу ключових слів
3. Пошукові ключові слова (Автоматичні) – це автоматична можливість від Apple Search Ads, щоб розміщати ваші рекламні пости, використовуючи теги, метатеги, які завчасно вже є в описах застосунків, у якості ключових слів.
4. Негативні слова (мінус-слова) – це слова, на які не буде показуватись реклама застосунку. Сюди правильно включати нерелевантні слова.

Типи стратегій кампаній [2]:

- Брендова кампанія – стратегія, яка зосереджена на ключових словах з назвою бренду
- Категоріальна або загальна кампанія - в рамках цієї стратегії можливе таргетування на певну категорію або вид додатків.
- Конкурентна кампанія – стратегія, яка використовую ключові слова конкурентів
- Пошукова кампанія – стратегія, що використовує автоматичні або широкі ключові слова для пошуку нових релевантних пошукових запитів

1.2 Головні показники:

CPA – Cost per Acquisition, ціна, яку ви платите за завантаження

CPT – Cost per Tap, середня ціна за клік на рекламу

CR – Conversion Rate, це відношення кількості завантажень до кількості кліків

TTR – Tap-Through-Rate, це відношення кількості натискань до кількості показів

Spend – кількість витрат

Impressions – кількість показів

Taps – кількість натискань

Conversions – кількість завантажень після натискань на рекламне оголошення

РОЗДІЛ 2. МОДЕЛЬ АУКЦІОНУ ДРУГОЇ ЦІНИ В APPLE SEARCH ADS

2.1 Загальний принцип моделі торгів другої ціни

Існує багато різних моделей торгів. В Apple Search Ads використовується типічна для пошукових систем модель аукціону другого біда (другої ціни). Вивчення та впровадження цього аукціону почалось з праці англійського економіста та професора Колумбійського університету Вільяма Вікрі, який запропонував новий вид аукціону – «аукціон другої ціни» у 1961 році [3]. На аукціоні Вікрі учасники подають запечатані ставки, не знаючи про ставки інших учасників аукціону. Той, хто запропонує найвищу ставку, виграє товар і сплачує суму ставки. Цей тип аукціону стратегічно еквівалентний аукціону другої ціни та надає учасникам стимул для участі в торгах за їх справжню вартість. Аукціон Вікрі був зроблений на основі подібного аукціону, який був запропонований голландським економістом Яном Тінбергеном у 1951 році. Аукціони Тінбергена також є аукціонами з таємними ставками, але переможець сплачує середнє значення ставок усіх інших учасників. Аукціони Вікрі мають багато переваг перед іншими видами аукціонів. По-перше, вони ефективні, тобто продукт продаватиметься тому учаснику, який цінує його найбільше. По-друге, вони справедливі, тобто всі учасники торгів мають рівні шанси на перемогу в аукціоні. Концепт аукціону полягає в наступному. Нехай задано, що:

n – кількість учасників аукціону

b_i – ставка і-го учасника

v_i – ціна товару

Виграш для і-го учасника буде записуватись як:

$$\begin{cases} v_i - \max_{j \neq i} b_j \text{ , якщо } b_i > \max_{j \neq i} b_j \\ 0 \text{ , якщо } b_i < \max_{j \neq i} b_j \end{cases}$$

Пізніше концепт даного аукціону розвивався і вдосконалювався. Один з видів аукціону другої ціни широко використовується в різноманітних пошукових системах, як Google, Yahoo!, і в App Store.

2.2 Принцип загальної моделі торгів другої ціни

Головну увагу приділимо узагальненому аукціону другої ціни, який розорблений на основі аукціону Вікрі спеціально для унікального середовища ринку онлайн-реклами.

Розглянемо аукціон з n учасників і k позицій, де $n \geq k$. Кожна позиція пов'язана з публічно відомим коефіцієнтом клікабельності CTR_i. Кожен учасник має власну лінійну приватну оцінку клікабельності, рівну v_i. Кожен учасник подає публічно відому ставку b_i. Задано вектор ставок b розміром n-на-1, учасники аукціону впорядковані за розміром ставки

$$b_1 \geq b_2 \geq \ldots \geq b_i$$

В результаті найвищу позицію отримує учасник з найвищою ставкою, другу позицію - учасник з другою найвищою ставкою і так далі. Після завершення аукціону кожен учасник платить

$$b_{i-1}$$

за кожен клік (тобто ставку наступного свого конкурента), а прибуток кожного учасника визначається як

$$(v_i - b_{i-1}) \times CTR_i$$

В будь-який момент часу учасник може змінити свою ставку. Аукціон закінчується, коли жоден учасник не бажає більше змінювати свою ставку[4]. Важливо наголосити, що для переміщення на одну позицію гравець повинен підняти ставку вище, ніж ставка гравця, що перебуває на одну позицію вище нього, але для переміщення на одну позицію вниз гравець повинен знизити ставку нижче ставки гравця, що перебуває на одну позицію нижче нього, і вище ставки гравця, що перебуває на дві позиції нижче нього[5]

2.2 Застосування моделі торгів другої ціни в Apple Search Ads

Apple Search Ads використовує модель узагальненого аукціону другої ціни, щоправда трохи змінений. Після завершення аукціону кожен учасник платить

$$b_{i-1} + 0.01$$

за кожен клік (тобто ставку наступного свого конкурента), а прибуток кожного учасника визначається як

$$(v_i - (b_{i-1} + 0.01) \times CTR_i$$

На практиці в Apple Search Ads модель виглядає наступним чином. Розглянемо три застосунки. Кожен рекламодавець дає певну ставку для свого рекламного оголошення, наприклад, 3.00$, 2.00$ і 1.00$ відповідно ставки. В аукціоні виграє рекламодавець, який поставив найбільшу ставку – 3.00$. Але кінцева ціна, яку сплачує переможець буде наступна найбільша ставка + 0.01$ – 2.01$.

РОЗДІЛ 3. РЕКОМЕНДАЦІЙНІ СИСТЕМИ

3.1 Загальна структура

Рекомендаційна система – це набір програмних засобів і технік, за допомогою яких на основі аналізу даних рекомендують найбільш відповідні предмети користувачам даної системи. На сьогоднішній момент рекомендаційні системи є важливим розділом машинного навчання, адже кожен бізнес намагається привернути увагу користувача і найефективніше це можна зробити за допомогою вдалої рекомендаційної системи. Перший, хто зробив великий вклад в дослідження даної теми був Amazon, і тепер його система рекомендацій працює майже ідеально. Розглянемо загальну структуру будь-якої рекомендаційної системи. Всього є 3 етапи її побудови [6]:

1. Відбір кандидатів

На цьому першому етапі система починає з великої сукупності і генерує набагато менше підмножини кандидатів. Наприклад, генератор кандидатів в App Store скорочує мільйони застосунків до сотень чи тисяч. Модель повинна швидко оцінювати запити з огляду на величезний розмір сукупності. Дана модель може надавати кілька генераторів кандидатів, кожен із яких призначає свої підмножини кандидатів.

2. Підрахунок балів

Потім інша модель оцінює та ранжує кандидатів, щоб вибрати набір елементів для відображення користувачу. Оскільки ця модель оцінює відносно невелику кількість елементів, система може використовувати більш точну модель, засновану на додаткових запитах.

3. Переоцінка

На останньому етапі система повинна враховувати додаткові обмеження остаточного ранжування. Наприклад, система видаляє елементи, які не сподобалися користувачеві, або підвищує оцінку свіжого контенту. Повторне

ранжування також може допомогти забезпечити різноманітність, свіжість та справедливість даних.

Якщо описувати роботу рекомендаційної системи більш технічно, її можна подати наступним чином [9].

1) На вхід подаються певні дані
2) Підготовка даних:
   a) Вимірювання відстанні
   b) Вибіркове дослідження
   c) Зменшення розмірності (Principal Component Analysis, наприклад)
3) Аналіз даних:
   a) Фільтрація даних
   b) Прогнозування методом класифікації
      i) Використання методів kNN, Decision Tree, Random Forest тощо
   c) Опис
      i) Асоціативний аналіз
      ii) Кластерізація такими методами, як k-means, Density-based тощо
4) Інтерпретація даних

3.2 Типи фільтрацій

Одним з головних етапів є фільтрування даних. Цьому етапу ми приділимо основну увагу, а також оберемо оптимальний для нашої модель тип фільтрації.

3.2.1 Контентна фільтрація [8]

У рекомендаційних системах, що використовують контент-орієнтоване фільтрування, застосовуються алгоритми машинного навчання для прогнозування та рекомендації нових, але схожих елементів користувачеві. Рекомендаційна система зберігає попередні дані користувача, такі як кліки та рейтинги для створення профілю користувача. Вона працює шляхом аналізу властивостей товару (таких як жанр, сюжет або ключові слова) і рекомендації товарів з подібними властивостями користувачу. Цей тип системи рекомендацій фокусується на особливостях товару та вподобаннях користувача.

З вхідного датасету, на жаль, не можливо було провести гарну фільтрацію на основі контенту через нестачу даних про користувачів. Тому я зосередилась на більш детальному описі методів спільної фільтрації

3.2.2 Спільна фільтрація (Collaborative Filtering)[8]

Спільна фільтрація - це тип системи рекомендацій, який використовує колективну поведінку користувачів для рекомендації товарів. Він працює, аналізуючи поведінку та вподобання користувача, а потім рекомендує товари, які також сподобалися іншим користувачам з подібною поведінкою та вподобаннями. Система використовує історичні дані взаємодії користувачів з товарами та виявляє подібності між користувачами.

Переваги спільного фільтрування

Персоналізовані рекомендації: Спільне фільтрування може надавати користувачам персоналізовані рекомендації на основі їх поведінки та вподобань. Це допомагає користувачам знайти товари, які ймовірно їм сподобаються.

Масштабованість: Спільне фільтрування може бути застосоване до широкого спектру товарів, що робить його масштабованим для великих наборів даних.

Неочікуваність: Спільне фільтрування може познайомити користувачів з новими товарами, які вони, можливо, не відкрили б самостійно, рекомендуючи популярні товари серед подібних користувачів.

Недоліки спільного фільтрування

Проблема холодного старту: Для ефективної роботи спільного фільтрування потрібна значна кількість даних користувача. Без цих даних може бути неможливо надати точні рекомендації, особливо новим користувачам.

Обмежена різноманітність: Спільне фільтрування може призводити до відсутності різноманітності в рекомендаціях, оскільки воно рекомендує ли ше товари, які є популярними серед подібних користувачів. Це може призвести до того, що користувачі будуть викладені обмеженому набору товарів.

3.2.3 Гібридна фільтрація [8]

Гібридна система рекомендацій має переваги, оскільки вона вирішує деякі недоліки як спільного фільтрування, так і фільтрування на основі вмісту. Вона може надавати більш різноманітні рекомендації, ніж спільне фільтрування, оскільки не обмежується рекомендаціями товарів, що є популярними серед користувачів з подібними смаками. І вона може надавати більш персоналізовані рекомендації, ніж фільтрування на основі вмісту, оскільки враховує вподобання та поведінку користувача.

Одним з головних завдань побудови правильної рекомендаційної системи з гарною фільтрацією є знаходження правильного підходу для вимірювання відстань (релятивізації об'єктів). Розглянемо даний аспект детальніше.

3.3 Міра подібності [7]

Будь-який метод фільтрації зіставляють кожен елемент і кожен запит (або контекст) із вкладеним вектором у загальному вкладеному просторі $E = \mathbb{R}^d$. Як правило, вкладений простір є низькорозмірним (тобто набагато менше розміру сукупності) та фіксує деяку приховану структуру елемента або набору запитів. Схожі елементи, такі як застосунки в App Store, які зазвичай переглядає той самий користувач, виявляються близько один до одного в просторі для вкладення. Поняття «близькість» визначається мірою подібності.

Мірою подібності є функція $s : E \times E \to \mathbb{R}$, яка приймає пару вкладень і повертає скаляр, що вимірює їхню подібність. Вкладення можна використовувати для створення кандидатів наступним чином: за запитом, що вбудовує $q \in E$, система шукає вкладення елементів $x \in E$, які близькі до q, тобто вкладення з високою схожістю $s(q, x)$.

Для визначення ступеня подібності більшість рекомендаційних систем покладаються однією чи кілька з таких факторів:

- Косинус кута між двома векторами $s(q, x) = cos(q, x)$

- скалярний добуток двох векторів, які обчислюються за формулою $s(q,x) = \langle q,x \rangle = \sum_{i=1}^{d} q_i x_i$. Також він обраховується за формулою $s(q,x) = \|x\|\|q\|cos(q,x)$, тобто косинус кута помножений на добуток норм. Отже, якщо вкладення нормовані, то скалярний добуток і косинус рівні.
- Євклідова відстань – відстань у евклідовому просторі $s(q,x) = \|q - x\| = \left[\sum_{i=1}^{d}(q_i - x_i)^2\right]^{\frac{1}{2}}$. Менша відстань означає більшу подібність. При нормованості вкладень квадрат евклідової відстані збігається зі скалярним добутком і косинусом з точністю до константи, оскільки в цьому випадку $\frac{1}{2}\|q - x\|^2 = 1 - \langle q,x \rangle$

3.3.1 Порівняння мір подібності [7]

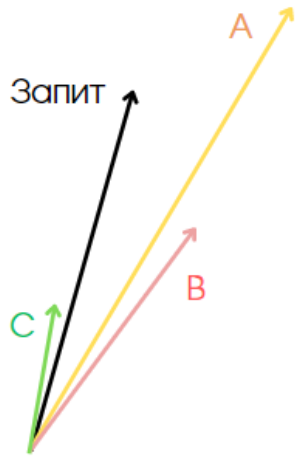


Рисунок 3.1 Порівняння мір подібності

Розглянемо приклад малюнку. Чорний вектор ілюструє вкладення запиту. Інші три вектори вкладення (елемент A, елемент B, елемент C) є елементами-кандидатами. Залежно від використовуваної подібності заходи ранжування елементів може бути різним.

Елемент A має найбільшу норму та має більш високий рейтинг відповідно до скалярного добутку. Елемент C має найменший кут із запитом і, таким чином, посідає перше місце відповідно до косинусної подібності. Елемент B фізично найближче до запиту, тому йому сприяє евклідова відстань.

3.1.3 Яку міру подібності вибрати? [7]

У порівнянні з косинусом подібність скалярного добутку чутлива до норми вкладення. Тобто чим більше норма вкладення, тим вища подібність (для виробів з гострим кутом) і тим більша ймовірність того, що застосунок буде рекомендований. Це може вплинути на рекомендації таким чином:

- Елементи, які дуже часто з'являються в навчальній вибірці (наприклад, популярні додатки в App Store), зазвичай мають вкладення з великими нормами. Якщо бажано фіксувати інформацію про популярність, то вам слід віддати перевагу скалярному добутку. Однак, якщо ви не обережні, популярні предмети можуть переважати в рекомендаціях. . На практиці можна використовувати інші варіанти мір подібності, які менше акцентують увагу на нормі об’єкта. Наприклад, визначте $s(q,x) = \|x\|^{\alpha}\|q\|^{\alpha}cos(q,x)$ для деяких $\alpha \in (0,1)$
- Елементи, які з’являються дуже рідко, можуть не часто оновлюватися під час навчання. Отже, якщо вони ініціалізовані з великою нормою, система може рекомендувати рідкісні елементи замість більш відповідних елементів. Щоб уникнути цієї проблеми, треба бути обережним щодо вбудовування ініціалізації та використовуйте відповідну регуляризацію.

Висновки:

Побудова рекомендаційної системи це важкий та трудомісткий процес, який включає багато нюансів. В заявленому датасеті немає даних про попередню поведінку користувачів. Проаналізувавши різні типи фільтрацій, я дійшла висновку, що для виконання завдання моєї кваліфікаційної роботи надалі буде використовуватись саме метод спільного фільтрування.

РОЗДІЛ 4. МЕТОДИ КЛАСИФІКАЦІЇ

4.1 Алгоритм k-найближчих сусідів (kNN method) [9]

Алгоритм k-найближчих сусідів є одним з найпоширеніших видів спільної фільтрації. Заданій певній точці для класифікації, класифікатор kNN знаходить k найближчих точок (найближчих сусідів) з навчальних записів. Потім він присвоює мітку класу відповідно до міток класу його найближчих сусідів. Основна ідея полягає в тому, що якщо запис потрапляє в певний район, де мітка класу переважає, це означає, що ймовірно запис належить до цього самого класу.

Отже, нехай задано точку q для якої ми хочемо знати її клас l, і навчальний набір

$$X = \{\{x_1, l_1\} \ldots \{x_n\}\}$$

де x_j – це j -тий елемент та l_j його відповідна мітка класу. Алгоритм k-найближчих сусідів знаходить таку підмножину

$$Y = \{\{y_1, l_1\} \ldots \{y_k\}\}$$

таку, що $Y \in X$ та крім того, виконується, що сума

$$\sum_{1}^{k} d(q, y_k)$$

є мінімальною. Y містить k точок з X, які найближчі до запиту q. Тоді, мітка класу q є такою:

$$l = f(\{l_1 \ldots l_k\})$$

Найважливішою проблемою в алгоритмі kNN є вибір значення k. Якщо k занадто мале, класифікатор буде чутливим до шумових точок. Але якщо k занадто велике, сусідство може включати занадто багато точок з інших класів.

Цей метод класифікації - як і більшість класифікаторів і технік кластеризації - дуже залежить від визначення відповідної міри схожості або відстані.

Найпростішим і найпоширенішим прикладом міри відстані є Евклідова відстань:

$$d(x,y) = \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2}$$

де n - кількість вимірів (атрибутів), а x_k та y_k – k-й атрибут (компонента) об'єктів даних x та y відповідно. Відстань Мінковського є узагальненням евклідової відстані, який найчастіше використовується для побудови рекомендаційних систем:

$$d(x,y) = \left(\sum_{k=1}^{n}|x_k - y_k|^r\right)^{\frac{1}{r}}$$

де r - ступінь відстані. Залежно від значення r, узагальнена відстань Мінковського відома під конкретними назвами:

- при $r = 1$ - містобудівна (Мангеттенська, таксі, або L1 норма) відстань;
- при $r = 2$ - евклідова відстань;
- при $r \to \infty$ - супремум (L_{max} норма або L_∞ норма) відстань, що відповідає обчисленню максимальної різниці між будь-яким виміром об'єктів даних.

Махаланобісівська відстань визначається як:

$$d(x,y) = \sqrt{(x-y)\sigma^{-1}(x-y)^T}$$

де σ - коваріаційна матриця даних.

Іншим дуже поширеним підходом є розглядати елементи як вектори у n-вимірному просторі та обчислювати їх схожість як косинус кута, який вони утворюють:

$$cos(x,y) = \frac{(x \bullet y)}{\|x\|\|y\|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} {x_i}^2}\sqrt{\sum_{i=1}^{n} {y_i}^2}}$$

де • позначає скалярний добуток векторів, а $\|x\|$ - норма вектора x. Ця подібність відома як косинусна схожість або L2 норма.

Подібність між елементами також можна визначити їх кореляцією, яка вимірює лінійний зв'язок між об'єктами. Хоча існує кілька коефіцієнтів кореляції, які можуть бути застосовані, найчастіше використовується коефіцієнт Пірсона. Нехай задано коваріацію точок даних x і y Σ та їх стандартне відхилення σ, ми обчислюємо коефіцієнт кореляції Пірсона за допомогою такого виразу:

$$cor(x,y) = \frac{cov(x,y)}{\sigma_x \sigma_y} = \frac{\sum(x-\bar{x})\,(y-\bar{y})}{\sigma_x \sigma_y}$$

де $\bar{x}$ і $\bar{y}$ - середні значення x і y, а $\sigma_x \; i \;\; \sigma_y$ - їх стандартні відхилення.

4.2 Алгоритм Random Forest

Метод випадкового лісу є важливим розділом під час вивчення машинного навчання. Він широко використовується в різних типах задач.

Випадковий ліс - це метод навчання з учителем, який використовує ансамбль дерев для класифікації та регресії. Випадковий ліс є методом bagging. Дерева випадкового лісу ростуть паралельно, що означає, що немає взаємодії між ними протягом процесу росту [10]. Метод випадкового лісу описується наступним чином [12]:

Задано ансамбль класифікаторів $h_1(x), h_2(x), \ldots, h_{\mathrm{k}}\,(x)$, і тренувальний набір вибирається випадковим чином з розподілу випадкового вектора Y, X. Визначимо функцію межі (margin function) як

$$mg(X,Y) \;=\; av_k I(h_k(X) = Y) - \max_{j \neq Y} av_k I(h_k(X) = j)$$

де I(·) - функція-індикатор. Функція межі вимірює рівень, на який середня кількість голосів за клас X, Y перевищує середній голос за будь-який інший клас. Чим більша межа, тим більшу впевненість маємо в класифікації. Загальна помилка узагальнення визначається як

$$PE^* = \; P_{\mathrm{X,Y}}\,(mg(X,Y) \;<\; 0)$$

де X, Y вказують, що йдеться про імовірність у просторі X, Y.

У випадкових лісах $h_k(X) = h(X, \Theta k)$. Для великої кількості дерев випливає з Сильного Закону Великих Чисел і структури дерев, що при зростанні кількості дерев, майже напевно для всіх послідовностей $\Theta 1, \ldots$ PE* збігається до

$$P_{\mathrm{X,Y}}\left(P_{\Theta}(\mathrm{h}(\mathrm{X},\Theta)=\mathrm{Y}) - \max_{j\neq Y} P_{\Theta}(\mathrm{h}(\mathrm{X},\Theta)=\mathrm{j}) < 0\right)$$

Цей результат пояснює, чому випадкові ліси не перенавчаються при додаванні більше дерев, але дають обмежене значення загальної помилки узагальнення. Для випадкових лісів може бути отримана верхня межа для загальної помилки узагальнення на основі двох параметрів, які вимірюють точність окремих класифікаторів та залежність між ними. Взаємодія між цими двома факторами лежить в основі розуміння роботи випадкових лісів. Функція межі для випадкового лісу (random forest) визначається наступним чином:

$$\mathrm{mr}(\mathrm{X},\mathrm{Y}) = P_{\Theta}(\mathrm{h}(\mathrm{X},\Theta)=\mathrm{Y}) - \max_{j\neq Y} P_{\Theta}(\mathrm{h}(\mathrm{X},\Theta)=\mathrm{j})$$

і сила набору класифікаторів {h(x, k)} визначається як:

$$\mathrm{s} = \mathrm{E}_{\mathrm{X,Y}}\mathrm{mr}(\mathrm{X},\mathrm{Y})$$

Випадковий ліс має наступний алгоритм [13]

1. Для b=1 до B:

A) Створення bootstrap-вибірки Z* розміром N з тренувальних даних.

B) Виростити дерево випадкового лісу T_b на bootstrap-вибірці даних, повторюючи наступні кроки рекурсивно для кожного кінцевого вузла дерева, до досягнення вузла мінімального розміру $n_{\min}$

i) Випадково виберіть m змінних з p змінних.

ii) Оберіть найкращу змінну/точку розбиття серед m змінних.

iii) Розділіть вузол на два дочірні вузли.

2. Виведення ансамблю дерев $\{T_b\}_1^{\mathrm{B}}$

Для прогнозування нової точки x:

Регресія задається формулою: $\hat{f}_{rf}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$

Класифікація виражається наступним чином: Позначимо $\hat{C}_b(x)$ прогнозом класу b-го дерева випадкового лісу. Тоді: $\hat{C}_b(x) =$ majority vote $\{\hat{C}_b(x)\}_1^B$

РОЗДІЛ 5. ПРАКТИЧНА РЕАЛІЗАЦІЯ

5.1 Підготовка датасету

Вхідний датасет сет був наданий як файл у форматі .csv. Для швидкої і легкої підготовки необхідних даних засобами Power BI, я почала початкове опрацювання данних. Всього було 253 840 рядків та 19 стовпців в датасеті.

За допомогою Power BI я трохи скоротила початковий датасет. Він включав Автоматичні ключові слова, які не можливо спрогнозувати нашими засобами. Я не прибирала зайві стовпці для модель, бо вони можуть знадобиться для подальшого аналізу даних. Тож я залишила дані.

Стовпці-фактори:

campaign_name – назва рекламної кампанії

country_or_region – країни чи регіони

ad_group_name – назва груп оголошень

keyword – ключове слово

cpt_bid – ставка на ціну за клік на аукціоні

keyword_match_type – тип ключового слова

spend – витрати

taps – кліки

installs – завантаження

impressions – покази

ttr – коефіцієнт клікабельності, розраховується як :

кількість людей,що зробили кліккількість людей,що побачили рекламу

cr – коефіцієнт конверсій

avg_cpa – средня ціна за дію

avg_cpt – середня ціна за клік

new_downloads – нові завантаження

redownloads – повторні завантаження

lat_on_installs – скриті завантаження

lat_off_installs – відкриті завантаження

5.2 Основний хід роботи

Продовження роботи буде проходити в середовищі Collab Research за допомогою мови Python. Спочатку пересемо оброблений датасет в середовище файлів і імпортуємо його і необхідні бібліотеки.

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
import pandas as pd

df = pd.read_excel('asa.xlsx')
```

Пишемо метод, який описує модель аукціону другого біда

```
pd.options.mode.chained_assignment = None

def prediction_value(df, column_name):
    df[column_name] = df[column_name].sort_values(ascending=False).reset_index(drop=True)
    for i in range(0, len(df)):
        value = max(df[column_name].iloc[i] - 0.99, 0)
        df[column_name].iloc[i] = value
    return df[column_name]
```

Обираємо показники, за допомогою яких будемо будувати модель. А саме: факторами моделі є витрати на рекламу певного ключового слова, кількість

натискань, завантажень і показів. Змінною, що прогнозується буде ставка ключового слова.

```
X = df[["spend", "taps", "installs", "impressions"]]
y = prediction_value(df, 'cpt_bid')
```

Розбиваємо дані на тренувальний і тестовий датасети

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Застосовуємо нашу першу модель – k найближчих сусідів. Для цього використовуємо функцію KNeighborsRegressor (неперервні дані) з бібліотеки sklearn []

Моделюємо роботу при $k = 5$

```
knn_regressor = KNeighborsRegressor()
knn_regressor.fit(X_train, y_train)

knn_pred = knn_regressor.predict(X_test)

knn_mse = mean_squared_error(y_test, knn_pred)
print("Mean Squared Error (k-Nearest Neighbors):", knn_mse)
```

А також будуємо модель використовуючи метод Random Forest за допомогою функції RandomForestRegressor (неперервні дані) з бібліотеки sklearn []

```
rf_regressor = RandomForestRegressor()
rf_regressor.fit(X_train, y_train)

rf_pred = rf_regressor.predict(X_test)

rf_mse = mean_squared_error(y_test, rf_pred)
print("Mean Squared Error (Random Forest):", rf_mse)
```

Результат роботи:

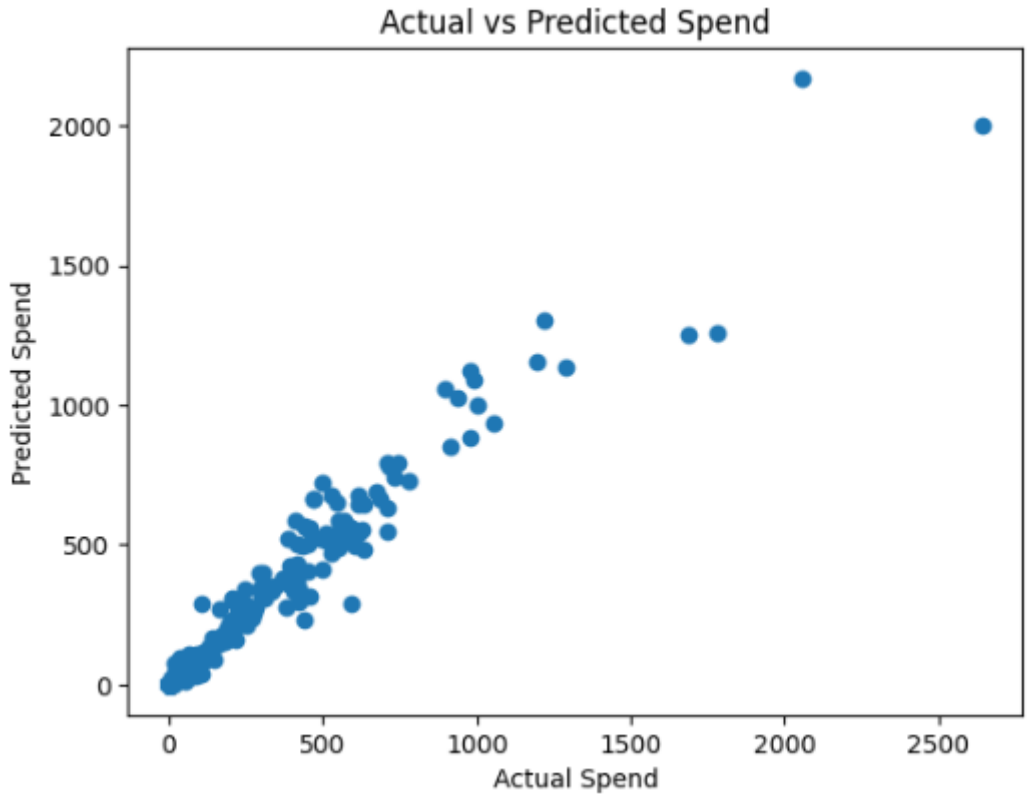
```
Mean Squared Error (Random Forest): 0.3087189601410156
Mean Squared Error (k-Nearest Neighbors): 0.3157940295201343
```

Показники доволі схожі, про MSE для випадкового лісу має трохи кращий показник. Отже, модель, що використовує метод RandomForest краще підходить

для реалізації аукціону другого біда. Подивимось на інші варіанти моделей прогнозування витрат на рекламу

Якщо за допомогою цих двох методів змоделювати можливі витрати, виходить доволі гарні результати:

```
Mean Squared Error (Random Forest): 39.35843458258109
R-squared (Random Forest): 0.9652908125845242
Mean Squared Error (k-Nearest Neighbors): 98.848419094935
R-squared (k-Nearest Neighbors): 0.9128281309844565
```



По результатам цього моделювання можна дійти висновку, що метод RandomForest значно краще працює для прогнозування витрат на рекламу, так як

$$MSE\_RandomForest < MSE\_KNeighborsRegressor$$

а також

$$R - squared\_RandomForest > R - squared\_KNeighborsRegressor$$

ВИСНОВКИ

В ході дослідження були розглянуті основні аспекти структури та моделі аукціону другої ціни в системі Apple Search Ads. Було встановлено, що ця модель торгів дозволяє ефективно визначати ціни на рекламні оголошення та забезпечувати оптимальну розстановку пріоритетів між рекламними пропозиціями.

Для покращення ефективності системи були досліджені рекомендаційні системи, які використовуються для персоналізації рекламного контенту. Зокрема, були проаналізовані різні типи фільтрацій, включаючи контентну фільтрацію, спільну фільтрацію та гібридну фільтрацію. В результаті порівняння, виявлено, що спільна фільтрація є найбільш підходящим методом для досягнення поставленої мети дипломної роботи.

Було проведено порівняльний аналіз двох алгоритмів - k-найближчих сусідів та Random Forest. За результатами порівняння, було виявлено, що модель, що використовує алгоритм Random Forest, показує кращі результати прогнозування витрат на рекламу. величина середньоквадратичної помилки (MSE) для Random Forest була нижча, а також коефіцієнт детермінації (R-squared) був вищим порівняно з алгоритмом k-найближчих сусідів.

Отже, на основі проведених досліджень можна зробити висновок, що модель, що використовує метод Random Forest, є більш ефективною для прогнозування ставка та витрат на рекламу в контексті системи Apple Search Ads. Дані висновки можуть бути використані для подальшої оптимізації рекламних кампаній та підвищення ефективності використання рекомендаційних систем у пошуковій рекламі.

СПИСОК ЛІТЕРАТУРИ

[1] Apple Search Ads довідка [Електронний ресурс]. URL: https://searchads.apple.com/advanced

[2] Сертифікаційна програма від Apple Search Ads [Електронний ресурс]. URL: https://bit.ly/3N0SnIm

[3] Стаття Вікіпедії - Vickrey auction: https://en.wikipedia.org/wiki/Vickrey_auction

[4] Varian, Hal R. (2007): “Position Auctions,” International Journal of Industrial Organization 25(6)

[5] Kevin Bryan (2008): «Generalized Second Price Auctions with Hierarchical Bidding». Virginia Commonwealth University. URL: https://scholarscompass.vcu.edu/cgi/viewcontent.cgi?referer=&httpsredir=1&article=2607&context=etd

[6] Machine Learning Certification Program by Google. URL: https://developers.google.com/machine-learning/recommendation/overview/types

[7] Machine Learning Certification Program by Google. URL: https://developers.google.com/machine-learning/recommendation/overview/candidate-generation

[8] Diko Sakti Prabowo, «Hybrid Recommendation System using LightFM» [Електронний ресурс]. URL: https://medium.com/@dikosaktiprabowo/hybrid-recommendation-system-using-lightfm-e10dd6b42923

[9] Francesco Ricci, Lior Rokach, Bracha Shapira, Paul B. Kantor (2011) “Recommender Systems Handbook”. URL: https://www.cse.iitk.ac.in/users/nsrivast/HCC/Recommender_systems_handbook.pdf

[10] Yancy Dennis. Random Forest Regression Model. URL: https://python.plainenglish.io/random-forest-regression-model-d060706a5e7f

[11] Will Koehrsen. An Implementation and Explanation of the Random Forest in Python. URL: https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76

[12] LEO BREIMAN (2001). “Random Forests”

[13] [Електроний ресурс]. URL: https://www.math.mcgill.ca/yyang/resources/doc/randomforest.pdf

Додатково опрацьовані джерела:

1 Machine Learning Certification Program by Google. URL: https://developers.google.com/machine-learning/recommendation/content-based/basics
2 A Guide to Content-Based Filtering In Recommender Systems https://www.turing.com/kb/content-based-filtering-in-recommender-systems
3 Anna Ivashko“Optimal Strategy Modelling in an Online Auction for the Rent of Computing Resources” https://ceur-ws.org/Vol-2792/paper4.pdf
4 Shubik, Martin (1983): “Auctions, Bidding, and Markets: A Historical Sketch,” in Auctions, Bidding and Contracting: Uses and Theory, NYU Press, New York
5 Shadi AlZu’bi, Amjed Zraiqat, and Samar Hendawi (01.10.2022) “Sustainable Development: A Semantics-aware Trends for Movies Recommendation System using Modern NLP”
6 Amit, Y. & Geman, D. (1997). “Shape quantization and recognition with randomized trees”

ДОДАТКИ

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
import pandas as pd

df = pd.read_excel('asa.xlsx')

pd.options.mode.chained_assignment = None

def prediction_value(df, column_name):
    df[column_name] =
df[column_name].sort_values(ascending=False).reset_index(drop=True)
    for i in range(0, len(df)):
        value = max(df[column_name].iloc[i] - 0.99, 0)
        df[column_name].iloc[i] = value
    return df[column_name]


X = df[["spend", "taps", "installs", "impressions"]]
y = prediction_value(df, 'cpt_bid')

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

rf_regressor = RandomForestRegressor()
rf_regressor.fit(X_train, y_train)

rf_pred = rf_regressor.predict(X_test)

rf_mse = mean_squared_error(y_test, rf_pred)
print("Mean Squared Error (Random Forest):", rf_mse)

knn_regressor = KNeighborsRegressor()
knn_regressor.fit(X_train, y_train)

knn_pred = knn_regressor.predict(X_test)

knn_mse = mean_squared_error(y_test, knn_pred)
print("Mean Squared Error (k-Nearest Neighbors):", knn_mse)
```

```
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

X = df[["cpt_bid","taps","installs","impressions"]]
y = df['spend']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

rf_reg = RandomForestRegressor(n_estimators=100)
rf_reg.fit(X_train, y_train)

y_pred = rf_reg.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error (Random Forest):", mse)
print("R-squared (Random Forest):", r2)

knn_regressor = KNeighborsRegressor()
knn_regressor.fit(X_train, y_train)

knn_pred = knn_regressor.predict(X_test)

knn_mse = mean_squared_error(y_test, knn_pred)
knn_r2 = r2_score(y_test, knn_pred)
print("Mean Squared Error (k-Nearest Neighbors):", knn_mse)
print("R-squared (k-Nearest Neighbors):", knn_r2)
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import mean_squared_error, r2_score

print("Metrics (Random Forest):")
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print("Mean Squared Error:", mse)
print("R-squared:", r2)

plt.scatter(y_test, y_pred)
plt.xlabel("Actual Spend")
plt.ylabel("Predicted Spend")
plt.title("Actual vs Predicted Spend")
plt.show()
```

```
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

df = pd.read_excel('asa.xlsx')

X = df[["cpt_bid","spend","taps","installs","impressions",]]
y = df['keyword_status']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

knn_pred = knn.predict(X_test)

knn_accuracy = accuracy_score(y_test, knn_pred)
print("Accuracy (k-Nearest Neighbors):", knn_accuracy)

rf = RandomForestClassifier(n_estimators=100)
rf.fit(X_train, y_train)

rf_pred = rf.predict(X_test)

rf_accuracy = accuracy_score(y_test, rf_pred)
print("Accuracy (Random Forest):", rf_accuracy)
```