Міністерство освіти і науки України
Національний університет «Києво-Могилянська академія»
Факультет інформатики
Кафедра математики

Магістерська робота

освітній ступінь – магістр

на тему: **«ПРОГНОЗНА ЦІНОУТВОРЮЮЧА МОДЕЛЬ ДЛЯ ТОВАРІВ З ХОЛОДНИМ СТАРТОМ»**

Виконав: студент 2-го року навчання,
Спеціальності
113 Прикладна математика

Величко Ростислав Анатолійович

Керівник: Дрінь С. С.
кандидат фіз.-мат. наук, ст. викладач

Рецензент: _____

Магістерська робота захищена
з оцінкою _____

Секретар ЕК_____
(підпис)

«_____» _____ 20___р.

Київ - 2024

Графік підготовки магістерської роботи до захисту

№ з/п	**ПЕРЕЛІК РОБІТ**	Термін виконання	Дата ознайомлення наукового керівника	Підпис наукового керівника	Примітки
1.	Вибір теми, затвердження її на засіданні кафедри та закріплення наукового керівника. Узгодження календарного графіка підготовки магістерської роботи. Ознайомлення студента з критеріями оцінювання магістерської роботи.	жовтень	12.10.2023		
2.	Вивчення джерел, літератури, періодичних видань, наукових публікацій, збір та узагальнення фактів, даних.	жовтень - листопад	16.11.2023		
3.	Складання плану магістерської роботи та узгодження із науковим керівником.	листопад	28.11.2023		
4.	Постановка експерименту, аналіз отриманих результатів наукового дослідження.	листопад - березень	12.03.2024		
5.	Проміжний контроль виконання роботи.	лютий	15.02.2024		
6.	Написаня кваліфікаційної роботи в цілому, ознайомлення зї першим варіантом наукового керівника.	січень - березень	28.03.2024		
	Розділ 1 (постановка проблеми, теоретичні основи, огляд літературних джерел).	квітень	06.04.2024		
	Розділ 2 (аналітично-дослідницька частина).	квітень	15.04.2024		
	Розділ 3 (проєктно-рекомендаційна частина).	квітень	25.04.2024		
7.	Повне завершення написання кваліфікаційної роботи, оформлення її згідно з вимогами й подання на відгук науковому керівнику.	квітень - початок травня	10.05.2024		
8.	Подання магістерської роботи для перевірки письмових робіт студентів НаУКМА на відповідність вимогам академічної доброчесності.	кінець травня	30.05.2024		
9.	Подання на зовнішню рецензію.	кінець травня	25.05.2024		
10.	Підготовка до захисту магістерської роботи на засіданні кафедри: написання доповіді та виготовлення ілюстративного матеріалу.	**до 17 травня**	16.05.2024		
11.	Підготовка до захисту магістерської роботи на засіданні кафедри: написання доповіді та виготовлення ілюстративного матеріалу.	**до 17 травня**	16.05.2024		
12.	Подання магістерської роботи на кафедру з усіма супроводжуючими документами.	**до 29 травня**	29.05.2024		
13.	Публічний захист перед екзаменаційною комісією	згідно з розкладом роботи ЕК	05.06.2024		

Графік узгоджено «_____» _____ 2024 р.

Науковий керівник _____
(ПІБ)

Виконавець магістерської роботи _____
(ПІБ)

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра математики

Освітній ступінь магістра

Спеціальність 113 Прикладна математика

Освітньо-наукова програма “Прикладна математика”

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

_____

“____”_____ 20__ року

ЗАВДАННЯ

ДЛЯ МАГІСТЕРСЬКОЇ РОБОТИ СТУДЕНТУ

Величко Ростиславу Анатолійовичу

1. Тема роботи: «Прогнозна ціноутворююча модель для товарів з холодним стартом»
керівник роботи: Дрінь Світлана Сергіївна, кандидат фіз.-мат. наук, ст. викладач
затверджені наказом вищого навчального закладу
від “___” _____ 20__ року № ___
2. Строк подання студентом роботи:
3. План роботи:
    1. Вступ і ознайомлення з існуючими методами
    2. Розробка гібридної моделі на базі методів GBDT та KNN
    3. Програмна реалізація
    4. Висновки
    5. Список літератури

Зміст

Перелік умовних позначень

GBDT – Gradient Boosting Decision Tree

HNSW – Hierarchical Navigable Small World

KNN – K-Nearest Neighbor

LSH – Locality-Sensitive Hashing

MAE — Mean Absolute Error

MSE – Mean Squared Error

SKU – Stock Keeping Unit

SVD – Singular Value Decomposition

Анотація

Метою кваліфікаційної роботи є прогнозування ціни на нові товари, тобто без історії продажів на основі цін сусідніх схожих товарів.

У роботі була розроблена ціноутворююча гібридна модель для товарів з холодним стартом за допомогою градієнтного бустингу (Gradient Boosting Decision Trees, GBDT). Були знайдені сусідні товари до заданого SKU методом K-найближчих сусідів (K-Nearest Neighbor, KNN). Також були знайдені значення середньоквадратичної помилки та середньої абсолютної помилки для оцінки точності моделі.

1 Вступ

1.1 Актуальність

Прогнозна ціноутворююча модель для товарів з холодним стартом є надзвичайно актуальною темою в сучасних умовах розвитку електронної комерції та роздрібної торгівлі. Задача прогнозування цін на нові товари, для яких відсутня історія продажів та цінових даних, є важливою проблемою для багатьох компаній. Ефективне рішення цієї задачі дозволяє оптимізувати процес ціноутворення, збільшити прибутковість та знизити ризики.

Проблема прогнозного ціноутворення для нових товарів виникла разом з розвитком електронної комерції. У 2000-х роках з'явилися перші спроби використання алгоритмів машинного навчання для прогнозування цін. У монографії Hastie et al. (2009) були узагальнені результати досліджень у цій сфері, що значно просунули розуміння методів прогнозного моделювання. У подальших дослідженнях, таких як робота Harrington (2012), були описані нові підходи та алгоритми, що дозволили покращити точність прогнозів.

У нашій роботі ми зосереджуємося на створенні моделі прогнозного ціноутворення для товарів з холодним стартом. Ми використовуємо градієнтний бустинг як основний метод для побудови моделі, оскільки він дозволяє ефективно обробляти складні залежності між ознаками товарів та їх цінами. Наша мета знайти групу схожих товарів та побудувати модель, яка зможе точно прогнозувати ціну нових товарів на основі обмеженого набору ознак.

Тема цієї роботи є надзвичайно актуальною. Використання машинного навчання для вирішення завдань ціноутворення відкриває нові можливості для бізнесу та допомагає знижувати ризики, пов'язані з випуском нових товарів на ринок. Крім того, наш підхід може бути адаптований для вирішення інших завдань у сфері прогнозування та аналітики даних, що робить його універсальним інструментом у сучасному світі бізнесу.

1.2 Мета, завдання дослідження

Метою кваліфікаційної роботи є розгляд декількох альтернативних підходів до задачі прогнозування ціни на новий товар з певної групи товарів викори-

стовуючи методи GBDT та KNN. Завданням є побудувати гібридну модель та оцінити точність цієї моделі.
Це передбачає розгляд таких задач:

- Огляд існуючих методів та моделей.
- Побудова першого етапу гібридної моделі а саме: пошук групи схожих товарів, які ми розглядаємо використовуючи метод KNN (K-Nearest Neighbor).
- Побудова другого етапу гібридної моделі а саме: прогнозування ціни на товар без передісторії (з холодним стартом) використовуючи модель GBDT.
- Оцінка моделі.
- Практична реалізація:
  - Обробка та підготовка даних.
  - Прогнозування ціни на основі створеної гібридної моделі.

1.3 Огляд літератури

Модель багаторукого бандита (Multi-armed bandit)
Модель багаторукого бандита - це математична модель, яка використовується для прийняття рішень у ситуаціях, де потрібно збалансувати експлорацію (дослідження нових варіантів) та експлуатацію (використання вже відомих та ефективних стратегій).

Відповідно до книги Lattimore and Szepesvári (2020) розглянемо, як працює дана модель. Уявімо, що маємо кілька різних цінових стратегій, і ми хочемо визначити, яка з них найефективніша для нового товару, який не має історії продажів (холодний старт). Модель багаторукого бандита дозволяє системі ефективно експериментувати з різними ціновими стратегіями для товару. Кожна рука моделі відповідає одній ціновій стратегії, і сама модель

визначає, яку стратегію використовувати для максимізації очікуваного прибутку. Моделі багаторукого бандита ефективно збирає дані про відгуки на новий товар, експериментуючи з різними цінами і швидше знаходячи оптимальну цінову стратегію для максимізації прибутку в умовах холодного старту. Такий підхід сприяє швидшій адаптації до ринкових умов та споживацьких уподобань.

Дерева рішень та Випадковий ліс

Дерево рішень - це алгоритм машинного навчання, який використовується як для класифікації, так і для регресії. Описаний даний алгоритм у роботі Breiman (2017).

Основна ідея полягає у створенні дерева, де кожен вузол представляє певний тест на ознаку даних, а кожне ребро відображає можливий результат цього тесту. У результаті, шлях від кореневого вузла до листового вузла представляє собою послідовність рішень, які призводять до класифікації об'єкта чи прогнозування його значення. Дерева рішень легко інтерпретуються, оскільки їх можна відобразити графічно. Вони також володіють природною здатністю до управління перенавчанням та адаптацією до даних. Для задачі пошуку найближчих товарів до заданого, дерева рішень використовуються для класифікації товарів за їх ознаками та структуруванням.

Випадковий ліс є розширенням дерев рішень і є одним з найпопулярніших алгоритмів машинного навчання. Вперше він був представлений у роботі Ho (1995).

Основна ідея полягає в тому, що випадковий ліс побудований на основі декількох дерев рішень, кожен з яких навчається на випадковій підмножині даних і випадковій підмножині ознак. Коли потрібно зробити прогноз, кожне дерево надає свій внесок, а результат обчислюється у середньому або шляхом голосування. Випадкові ліси відомі своєю високою точністю та володіють здатністю уникати перенавчання. Вони широко використовуються в задачах класифікації, регресії та кластеризації.

Нейронні мережі

Ідея використання нейронних мереж для задачі прогнознозування ціни на

товар з холодним стартом виникла в середині 20-го століття, коли почали активно досліджувати можливості використання машинного навчання в економіці та бізнесі. Однак, використання нейронних мереж для цільового ціноутворення стало популярним з поширенням обчислювальних технологій та збільшенням доступності великих обсягів даних. Алгоритм роботи полягає в тому, що на основі зібраних даних про сусідні схожі товари, створюється нейронна мережа (одношарова чи багатошарова, з різними типами шарів). Ваги нейронів оновлюються в процесі навчання таким чином, щоб мінімізувати помилку між прогнозованими і фактичними результатами. Одним із прикладів використання нейронних мереж для задачі прогнозування ціни на товар є робота Qian et al. (2019), де використовуються Графові нейронні мережі. Серед переваг нейронних мереж можна виокремити те, що вони можуть працювати з великими обсягами даних та мають високу точність прогнозів. Якщо говорити про недоліки - то це те, що нейронні мережі зазвичай потребують велику кількість даних, та тей факт, що у багатьох випадках вони є доволі складними для інтерпретації

Існуючі методи пошуку найближчих сусідів

Далі ми розглянемо відомі методи, які застосовуються для вирішенння проблеми пошуку найближчих сусідів.

1. Методи матричного розкладу

Згідно з дослідженням Koren et al. (2009), методи матричного розкладу, такі як Singular Value Decomposition (SVD), дозволяють ефективно робити рекомендації на основі історії покупок користувачів та їх взаємодії з продуктами. Алгоритм матричного розкладання, запропонований Koren et al. (2009), полягає у розкладанні матриці рейтингів користувачів та товарів на два більш простих матриці: матрицю профілів користувачів та матрицю профілів товарів. Для пошуку найближчих сусідніх товарів до заданого цей процес зазвичай відбувається в два етапи:

- Створення простору ознак: Спочатку товари представляються у вигляді векторів у просторі ознак. Ці вектори можна отримати шляхом матричного розкладання, де кожна товарна позиція буде мати свій векторний

представник на основі інформації про її характеристики та рейтинги, які надають їй користувачі.

- Обчислення подібності: Після того, як товари будуть представлені у вигляді векторів у просторі ознак, можна обчислити схожість між ними. Це може бути зроблено, наприклад, за допомогою косинусної схожості або евклідової відстані між векторами. Товари з найбільшими показниками до заданого будуть найближчими товарами.

2. Пошук за допомогою просторових хеш-таблиць (Locality-Sensitive Hashing, LSH)

Locality-Sensitive Hashing (LSH) – це метод для пошуку схожих об'єктів у великих наборах даних, особливо у високовимірних просторах. Ідея полягає в тому, щоб хешувати схожі об'єкти у ті ж самі "відра"з високою ймовірністю, тим самим скорочуючи обсяг даних, який потрібно переглядати для пошуку найближчих сусідів. LSH був вперше запропонований в роботі Indyk and Motwani (1998).

Основна ідея алгориму LSH полягає в тому, що використовується сімейство хеш-функцій, які мають властивість "просторової чутливості": схожі об'єкти з високою ймовірністю мають однакові хеші.

Для евклідових просторів часто використовуються хеш-функції, засновані на випадкових проекціях:

$$h_{\mathbf{a},b}(\mathbf{v}) = \left\lfloor \frac{\mathbf{a} \cdot \mathbf{v} + b}{w} \right\rfloor,$$

де $\mathbf{a}$ – випадковий вектор, b – випадкове зміщення, а w – параметр ширини «відра» (визначає розмір відрізку простору, на який дані розбиваються при хешуванні).

До переваг даного алгоритму можна віднести швидкість та масштабованість, а до недоліків - те, що він дає не точні, а приблизні результати.

3. Графові методи (Hierarchical Navigable Small World, HNSW)

Графові методи для пошуку найближчих сусідів використовують графи для організації даних, що дозволяє ефективно знаходити найближчі точки. Один

з найбільш популярних алгоритмів у цій категорії — Hierarchical Navigable Small World (HNSW).

Відповідно до роботи Aumüller et al. (2020), основна ідея полягає в тому, що HNSW будує граф, де вершини представляють точки даних, а ребра з'єднують близькі точки. Граф організований в декілька рівнів, де верхні рівні мають менше точок і служать для швидкої навігації, а нижні — більш детальні та точні. Переваги алгоритму це швидкість, точність, а недоліки - складність реалізації, час побудови, використання великих обсягів пам'яті.

1.4 Опис структури кваліфікаційної роботи

Дана кваліфікаційна робота складається з наступних розділів.
В розділі «Перелік умовних позначень» відповідно містяться всі умовні позначення разом із поясненнями, які зустрічаються в роботі.

Розділ «1 Вступ» складається з 4 підрозділів: «1.1 Актуальність» - де описується чому дане дослідження взагалі є актуальним; «1.2 Мета, завдання дослідження» - в даному підрозділі ми чітко формуємо мету роботи, та конкретні завдання, які потрібно виконати; «1.3 Огляд літератури» - в даному підрозділі зазначені та коротко описані відомі методи, які використовуються для вирішення задачі ціноутворення на товар з холодним стартом; «1.4 Опис структури кваліфікаційної роботи» - це даний розділ, де ми описуємо з чого складається кожний розділ, та про що в ньому йде мова.

Розділ «2 Теоретична частина» складається з 3 підрозділів: «2.1 Пошук найближчих сусідів до досліджуваного SKU» - в даному підрозділі зазначаються відомі методи пошуку найближчих сусідів та описується метод KNN, який далі використовується у роботі; «2.2 Метод градієнтного бустингу (Gradient Boosting Decision Tree, GBDT) у задачі ціноутворення для товарів з холодним стартом» - в даному розділі повністю описується теоретична частина від дерев рішень до GBDT; «2.3 Оцінки моделі» - тут зазначаються та описуються

різні методи оцінки, які використовуються у роботі для побудованої моделі.

Розділ «3 Програмна частина» складається з 3 підрозділів: «3.1 Датасет» - в даному підрозділі описується датасет та його попередня обробка; «2.2 Програмна реалізація методу KNN» - опис програмної частини використання методу KNN; «3.3 Програмна реалізація прогнозування ціни на товар з використанням GBDT» - побудова та налаштування моделей, візуалізація та інтерпретація отриманих результатів.

В розділі «Висновки» формуються результати даної кваліфікаційної роботи та перспективи подальшого дослідження.

В розділі «Список літератури» зазначені список всіх джерел, які використовувались у ході даної роботи.

2 Теоретична частина

В даній роботі ми прогнозуємо ціну товару на основі двох різних груп товарів (знайдених найближчих сусідів до заданого товару). До першої групи найближчих сусідів входять всі товари з датасету, які були знайдені за допомогою методу KNN. До другої ж групи найближчих сусідів входять лише товари з тієї категорії, кількість входжень до якої є найбільшою серед всіх знайдених найближчих сусідів. Саме в цьому і полягає наукова новизна нашої роботи. Це є дуже важливою складовою в сучасних умовах швидкозростаючого ринку, оскільки разом зі зростом ринку, збільшуються й датасети з якими власне і працює побудована модель. Через цей зріст для менеджера магазину буде дуже важливо отримувати прогноз ціни на новий товар, як опираючись лише на товари з конкретної категорії, так і взагалому на всі товари, які є сусідніми до заданого. В даній роботі ми порівнюємо отримані результат для різних груп товарів, та робимо відповідні висновки.

2.1 Пошук найближчих сусідів до досліджуваного SKU

Stock Keeping Unit (SKU) — це ідентифікатор товарної позиції, одиниця обліку запасів, складський номер, який використовується в торгівлі для відстеження статистики по реалізованих товарах, або послугах.

Пошук найближчого сусіда є важливим інструментом у багатьох галузях інформатики, такі як розпізнавання зображень, машинне навчання та комп'ютерна лінгвістика. Наприклад, можна використовувати пошук найближчого сусіда на дескрипторах зображень, щоб розпізнати рукописний текст цифр, або можна знайти семантично подібні фрази до даної фрази. До того ж, пошук найближчих сусідів використовується для вирішення таких завдань, як рекомендаційні системи та аналіз великих об'ємів даних. Він дозволяє здійснювати класифікацію нових об'єктів на основі подібності із вже існуючими, а також знаходити структурні аналогії у наборах даних.

Метод K-найближчих сусідів (K-Nearest Neighbor, KNN)

Метод K-найближчих сусідів (KNN) є одним із найфундаментальніших

і найпростіших методів класифікації та був одним із перших варіантів для класифікаційного дослідження, коли немає або немає попередніх знань про розподіл даних. Метод KNN був розроблений з необхідності виконувати дискримінаційний аналіз, коли надійні параметричні оцінки щільності ймовірності невідомі або їх важко визначити. У неопублікованому звіті Fix (1985) представили непараметричний метод класифікації шаблонів, який з тих пір став відомим як метод K-найближчих сусідів. Пізніше в 1967 році було розроблено деякі формальні властивості методу KNN, наприклад, в роботі Cover and Hart (1967) було показано, що для $k = 1$ та $n \to \infty$ похибка класифікації k-найближчого сусіда обмежена вище подвоєною частотою помилок Байєса. Після встановлення таких формальних властивостей методу, почалася довга лінія досліджень, включаючи нові підходи до відхилення (Hellman (1970)), уточнення щодо частоти помилок Байєса (Fukunaga and Hostetler (1975)), підходи, зважені на відстані (Dudani (1976); Bailey and Jain (1978)), методи м'якого обчислення (Bermejo and Cabestany (2000)) і нечіткі методи (Jóźwik (1983); Keller et al. (1985)).

Міжвибіркова геометрична відстань

Метод KNN зазвичай базується на евклідовій відстані, або косинусній подібності між тестовим зразком і вказаними навчальними зразками. Нехай x_i - це вектор з p ознаками, n - загальна кількість векторів (n) і p - загальна кількість ознак. Евклідова відстань між зразком i та j визначається як:

$$d(x_i, x_j) = (x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \ldots + (x_{ip} - x_{jp})^2.$$

Коли ми використовуємо косинусну подібність, відстань між двома векторами визначається як кут між ними, а не як різниця їх значень. Ось як це виглядатиме для косинусної подібності:

$$\text{cosine\_similarity}(x_i, x_j) = \frac{x_i \cdot x_j}{\|x_i\| \cdot \|x_j\|}, \tag{1}$$

де $x_i \cdot x_j$ - це скалярний добуток між векторами x_i і x_j, а $\|x_i\|$ і $\|x_j\|$ - їхні довжини відповідно.

Правило прийняття рішення та матриця невідповідностей.

Класифікація зазвичай передбачає розподіл зразків на тренувальні та тестові категорії. Нехай x_i - тренувальний зразок, x_j - тестовий зразок, w_i - справжній клас тренувального зразка, w_j - передбачений клас тестового зразка. Тут W - загальна кількість класів. Під час процесу тренування ми використовуємо лише справжній клас w_i кожного тренувального зразка для навчання класифікатора, тоді як під час тестування ми передбачаємо клас w_j кожного тестового зразка.

За правилом 1-найближчого сусіда передбачений клас тестового зразка x_j встановлюється рівним справжньому класу w_i його найближчого сусіда, де x_m є найближчим сусідом для x_j. Для K-найближчих сусідів передбачений клас тестового зразка x_j встановлюється рівним найбільш часто зустрічаються справжнім класом серед найближчих тренувальних зразків. Це формує правило прийняття рішення.

Матриця невідповідностей, що використовується для табулювання передбачень класу тестових зразків під час тестування, позначається як C і має розміри $W \times W$. Під час тестування, якщо передбачений клас тестового зразка x_j є вірним (тобто $w_i = w_j$), тоді діагональний елемент матриці невідповідностей збільшується на 1. Однак, якщо передбачений клас невірний (тобто $w_i \neq w_j$), тоді позадіагональний елемент матриці збільшується на 1.

Після класифікації всіх тестових зразків точність класифікації базується на відношенні кількості правильно класифікованих зразків до загальної кількості класифікованих зразків, поданій у формі

$$Acc = \frac{\sum_w c_{ww}}{n_{\text{total}}},$$

де c_{ww} - діагональний елемент C, а n_{total} - загальна кількість класифікованих зразків(Peterson (2009)).

Алгоритм методу KNN:

1. Нехай маємо набір даних $(x_i, y_i) = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, де x_i - це вектор ознак і-го товару, а y_i - відповідна мітка класу або значення.

2. Для класифікації нового об'єкта x_{new} спочатку виміряємо відстань між

x_{new} і всіма іншими об'єктами в наборі даних. Зазвичай використовується косинусна подібність, формула (1).

3. Обираємо k найближчих сусідів x_i, які мають найменші відстані до x_{new}.

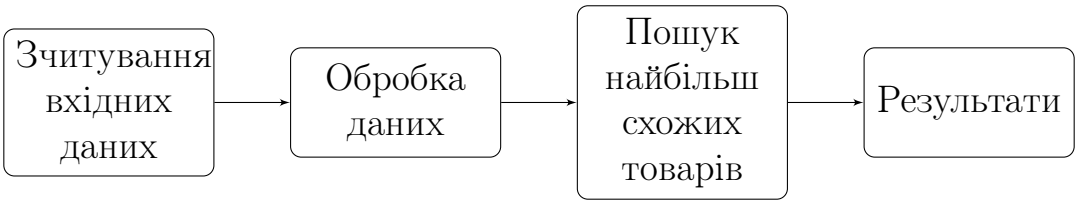

Рис. 1: Блок-схема процесу роботи методу KNN

Ми обираємо саме метод KNN для пошук найближчих сусідів саме через те, що він є ефективним інструментом у задачах, де важлива точність прогнозування, а також він дозволяє швидко аналізувати та обробляти великі обсяги даних і знаходити в них закономірності, що допомагає у вирішенні задачі ціноутворення.

2.2 Метод градієнтного бустінгу (Gradient Boosting Decision Trees, GBDT) у задачі ціноутворення для товарів з холодним стартом

Дерева рішень

Дерева рішень це модель машинного навчання, яка є одним із найбільш популярних методів розв'язання задач класифікації та прогнозування. Якщо залежна, тобто цільова змінна приймає неперервні значення, то дерево рішень установлює залежність цієї змінної від незалежних змінних, тобто розв'язує задачу чисельного прогнозування. Найпростіший варіант вигляду дерева рішень - це певний спосіб показати набір правил в ієрархічній, послідовній структурі (відповіді «Так» чи «Ні» на переліз питань). Кожне дерево рішень має кореневий вузол, гілки, внутрішні вузли та листові вузли.

Для дерева рішень важливим поняттям є критерій розщеплення, що називають його критерієм розбивки. Даний критерій вимірює нечистоту вузла рішення в дереві, для того щоб відсортувати атрибути для подальшого поділу на 2 секції для регресії. Найбільш відомі критерії розбивки - це індекс Gini та ентропія.

Індекс Gini - за його допомогою, атрибут обирається за допомогою відстаней між розподілами класів, обчислюється за формулою:

$$\text{gini}(T) = 1 - \sum_{j=1}^{n} p_j^2,$$

де T - набір даних, p_j - ймовірність класу j у вузлі T, а n - кількість класів.

Ентропія - міра інформативності підпросторів атрибутів, обчислюється за формулою:

$$\text{Entropy}(x_i) = -\sum_{c \in C} d(c) \log_2(d(c)),$$

де x_i - розглядуваний набір даних, c - класи в наборі даних x_i, $d(c)$ - частка точок даних, що належать до класу C відносно загальної кількості точок даних у наборі даних x_i. Значення ентропії коливаються від 0 до 1. Ентропія, рівна 0, вказує на те, що всі випадки в наборі даних належать до одного класу, тоді як ентропія, рівна 1, вказує на максимальне різноманіття.
Діапазон значень ентропії - від 0 до 1, де 0 - свідчить про те, що всі входження до набору даних схожі, а 1 - свідчить про те, що всі входження до набору максимально різні.

Бустинг

Бустинг - це метод, що використовується в машинному навчанні для зменшення помилок у прогнозному аналізі даних. Тобто потрібно натренувати моделі машинного навчання, на відомих даних, щоб робити припущення про невідомі дані. Одна модель машинного навчання може робити помилки у прогнозах залежно від точності навчального набору даних. Наприклад, якщо модель була навчена на недостатньому наборі даних, вона може час від часу давати невірні результати. Boosting намагається подолати цю проблему шляхом послідовного тренування декількох моделей для підвищення точності всієї системи.
Тобто, бустинг покращує точність прогнозування та продуктивність моделей машинного навчання, перетворюючи декілька слабких учнів на одну сильну навчальну модель. Слабкі учні (Weak Learners) - це моделі, які самостійно

можуть робити прогнози, тобто вони мають низьку точність прогнозування, схожу на випадкове вгадування, схильні до перенавчання, тобто вони не можуть класифікувати дані, які сильно відрізняються від їхнього початкового набору даних. Сильні учні (Strong Learners) - це ансамблі слабких учнів, які об'єднані таким чином, що результуюча модель має високу точність прогнозів. Бустинг перетворює систему слабких учнів на єдину сильну навчальну систему

Бустинг створює ансамбльну модель, комбінуючи кілька слабких дерев рішень послідовно. Воно присвоює ваги виходу кожного окремого дерева. Потім воно надає неправильним класифікаціям від першого дерева рішень вищу вагу та ввід до наступного дерева. Після численних циклів метод бустинг об'єднує ці слабкі правила в одне потужне правило прогнозу.

Бустинг і Беггінг - це два загальні методи ансамблей, які покращують точність прогнозування. Основна відмінність між цими методами навчання - це метод навчання. Беггінг покращує точність слабких учнів, навчаючи кілька з них одночасно на кількох наборах даних. Натомість, у бустингу слабкі учні навчаються один за одним.

Алгоритм виконує наступні загальні кроки для навчання моделі бустингу:

1. Спочатку кожному зразку даних (x_i, y_i) присвоюються початкові рівні ваги w_i. Дані потім передаються до першого базового алгоритму, який називається базовим алгоритмом $h_1(x)$. Базовий алгоритм робить прогнози для кожного зразка даних:

$$h_1(x_i).$$

2. Далі оцінюються прогнози моделі та вага кожного зразка оновлюється згідно з помилкою прогнозу. Також, присвоюється вага в залежності від продуктивності моделі. Модель, яка дає кращі прогнози, матиме значний вплив на кінцеве рішення. Для кожного зразка даних визначається коефіцієнт ρ_j, який відображає вагу базового алгоритму $h_j(x)$ в кінцевому прогнозі.

3. Зважені дані передаються до наступного базового алгоритму $h_{j+1}(x)$.

4. Кроки 2 і 3 повторюються доти, поки помилки на навчальних даних не

будуть менші за певний поріг.
З точки зору математики, отримуємо:

$$h(x) = \sum_{j=1}^{m} \rho_j h_j(x),$$

де ρ - ваги j-го класифікатора.

Градієнтний спуск

Метод градієнтного спуску ґрунтується на принципі, що якщо функція багатьох змінних $F(x)$ визначена та має похідні в околиці точки a, то $F(x)$ найшвидше зменшується при русі від точки a у напрямку негативного градієнта F в точці a, позначеного як $-\nabla F(a)$ (де ∇ позначає градієнт, вектор часткових похідних функції). Це означає, що якщо

$$a_{n+1} = a_n - \gamma \nabla F(a_n)$$

при достатньо малому кроці (або швидкості навчання) $\gamma \in \mathbb{R}^+$, то

$$F(a_n) \geq F(a_{n+1}).$$

Інакше кажучи, $\gamma \nabla F(a)$ віднімається від a, оскільки ми прагнемо рухатися проти градієнта до локального мінімуму. З цього розуміння, ми починаємо з x_0 як початкового припущення для локального мінімуму F, і розглядаємо послідовність $x_0, x_1, x_2, \ldots$ таку, що

$$x_{n+1} = x_n - \gamma_n \nabla F(x_n)$$

для $n \geq 0$.
Як результат, ми маємо

$$F(x_0) \geq F(x_1) \geq F(x_2) \geq \ldots,$$

і очікуємо, що послідовність x_n наближатиметься до локального мінімуму. Варто зазначити, що розмір кроку γ може змінюватися на кожній ітерації.

Метод градієнтного бустінгу GBDT

Метод градієнтного бустінгу (Gradient Boosting Decision Trees, GBDT) — це ансамблевий метод машинного навчання, який використовується для вирішення завдань регресії, класифікації та інших задач. Даний метод вперше був запропонований Джеромом Фрідманом в 1999 році Friedman (1999).

Ми маємо навчальний набір даних, який складається з n спостережень. Кожне спостереження містить інформацію про ціну товару y_i та його ознаки (назву та категорію товару). Також, нам дано певний SKU, для якого ми хочемо здійснити прогноз ціни.

Позначимо навчальний набір як (x_i, y_i), де x_i - вектор ознак i-го товару, y_i - його ціна. Наша мета - побудувати модель $F(x)$, яка прогнозує ціну для нового SKU на основі ознак товару x.

Модель градієнтного бустінгу буде апроксимувати функцію $F(x)$ шляхом додавання слабких моделей у вигляді ансамблю. Кожна наступна модель намагатиметься покращити попередні прогнози шляхом апроксимації залишкової функції. Апроксимація

$$F(x) \approx F_0(x) + \sum_{m=1}^{M} h_m(x),$$

де $F_0(x)$ - це початкове базове значення, а $h_m(x)$ - вклад m-тої моделі.

Позначимо $F_m(x)$ - прогноз, зроблений моделлю після m ітерацій. При добавленні m-тої моделі, ми намагаємося мінімізувати наступну функцію втрат

$$L(y, F_{m-1}(x) + h(x)) = L(y - (F_{m-1}(x) + h(x)))^2,$$

де L - функція втрат (квадратична втрата), y - цільова змінна, $F_{m-1}(x)$ - прогноз, зроблений попередніми моделями, а $h(x)$ - нова модель, яку ми намагаємося навчити на цій ітерації.

На кожному кроці m ми шукаємо таку модель $h_m(x)$, яка мінімізує цю функцію втрат шляхом градієнтного спуску.

Градієнт буде мати вигляд

$$\frac{\partial L}{\partial(F_{m-1}(x)+h(x))} = -2(y-(F_{m-1}(x)+h(x))).$$

Отже, після M ітерацій ми отримаємо модель у вигляді ансамблю

$$F_M(x) = F_0(x) + \sum_{m=1}^{M} h_m(x),$$

де $F_0(x)$ - початкове базове значення (середнє значення ціни в навчальному наборі), а $h_m(x)$ - вклад m-тої моделі.

Отже, для прогнозу нової ціни для заданого SKU за допомогою цін його найближчих сусідів за допомогою GBDT, ми будемо використовувати цю модель $F_M(x)$, де x - ознаки нового товару.

2.3 Оцінки моделі

Для оцінки нашої моделі використовуються середньоквадратична помилка (Mean Squared Error, MSE) та середня абсолютна помилка (Mean Absolute Error, MAE).

Середньоквадратична помилка (MSE) вимірює середньоквадратичну різницю між фактичними (y_i) та прогнозованими ($\hat{y}_i$) значеннями ціни товару для всіх n спостережень. Формула MSE

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2. \tag{2}$$

Середня абсолютна помилка (MAE) вимірює середню абсолютну різницю між фактичними (y_i) та прогнозованими ($\hat{y}_i$) значеннями ціни товару для всіх n спостережень. Формула MAE

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|. \tag{3}$$

3 Програмна частина

3.1 Датасет

Опис датасету

Датасет, з яким ми працюємо в даній кваліфікаційній роботі, це датасет української компанії, який містить дані про розподіл товарів за категорією, та їх оборот і кількість продажів за період з січня 2019 по червень 2023 року. В дасатесті мається 3616 різних SKU. Сам датасет складається з трьох файлів. Перший файл - category.xlsx, він містить 3616 SKU (рядків) розподілу товарів за категорією, групою, підгрупою 1 та 2 відповідно, та 4 ознаки (стовпці). Список ознак:

– Категорія: Категорія відповідного товару (наприклад - 1. Домашній затишок та комфорт)

– Група: Група відповідного товару (наприклад - 1.1 текстиль домашній)

– Підгрупа 1: Підгрупа 1 відповідного товару (наприклад - 1.1.1 подушки)

– Підгрупа 2: Конкретний товар (1.1.1.1 подушки 70х70)

Наступний файл - oborot.xlsx, він містить 4202 SKU (рядків) обороту конкретного товару кожного місяця за період з січня 2019 по червень 2023 року включно та 55 ознак (стовпців). Список ознак:

– Назва категорії: Назва конкретного товару (наприклад - 1.1.1.1 подушки 70х70)

– 54 ознаки кожного місяця за період з січня 2019 по червень 2023 року включно: оборот конкретного товару в цьому місяці, дані зазначені в гривнях (наприклад - 194,58)

Останній файл - sales.xlsx, він містить 4202 SKU (рядків) значення кількості продажів конкретного товару кожного місяця за період з січня 2019 по червень 2023 року включно та 55 ознак (стовпців). Список ознак:

– Назва категорії: Назва конкретного товару (наприклад - 1.1.1.1 подушки 70х70)

– 54 ознаки кожного місяця за період з січня 2019 по червень 2023 року включно: кількість продажів конкретного товару в цьому місяці (наприклад - 3)

Варто зазначити, що в файлі oborot.xlsx та sales.xlsx кількість записів більше, ніж у файлі з розподілом товару за категоріями, а також в ньому існують деякі записи, для яких значення всіх ознак, окрім його назви є нульовими (відсутні), тому він точно потребує попередньої обробки. Інформація про попередню обробку датасету буде докладно описана далі.

Попередня обробка датасету

1. Дані завантажуються з Excel-файлу (`category.xlsx`) за допомогою бібліотеки `pandas`.
2. У стовпці Підгрупа 2 усі значення `NaN` замінюються на пустий рядок, це нам потрібно для того, щоб потім коректно використовувати метод KNN.
3. Ініціалізується функція `remove_characters`, яка видаляє всі символи перед першою літерою в тексті. Далі ми її застосовуємо лише для стовпця «Підгрупа 2» з файлу category.xlsx, для того щоб ми отримали лише самі назви товарів, не включаючи цифри та знаки, які є перед назвою та свічать про розподілення товару до відповідної категорії, групи та підгрупи 1. Це нам потрібно, оскільки для пошуку найближчих сусідів за допомогою методу KNN, нам важлива лише сама назва товару, а всі інші данні будуть нам заважати.
4. Текстові дані у стовпці «Підгрупа 2» векторизуються за допомогою `TfidfVectorizer` з бібліотеки `sklearn`.
5. Дані завантажуються з двох Excel файлів (`oborot.xlsx` та `sales.xlsx`) за допомогою функції `pd.read_excel()`. Перші два рядки в кожному з файлів видаляються `iloc[2:]`, оскільки вони не містять корисної інформації для аналізу (текстові описи стовпців).
6. В рядках даних містяться дублікати в колонці Назва категорії. Ці дублікати видаляються за допомогою функції `drop_duplicates()`.
7. Створюється функція `convert_to_numeric()`, яка замінює коми на крапки у всіх колонках (окрім першої, оскільки вона містить назву товарів, а інші - числові характеристики) та перетворює їх у числовий формат. Функція застосовується до обох датафреймів.
8. Два датафрейми об'єднуються на основі колонки Назва категорії за допомогою функції `merge()`. Для уникнення конфліктів у назвах колонок додаються суфікси `_oborot` та `_sales`.

9. Створюється список всіх місяців, які присутні в даних, щоб використовувати їх пізніше для обчислення ціни товарів.
10. Створюється функція `calculate_price()`, яка обчислює ціну товару для кожного рядка даних. Ціна обчислюється як відношення об'єму обороту до кількості продажів за останній місяць, де є ненульові значення. Ця функція застосовується до об'єднаного датафрейму з використанням методу `apply()`.
11. Видаляються рядки, у яких значення в колонці «Ціна» є порожніми `NaN`.
12. У колонці «Назва категорії» видаляються всі символи до першої літери за допомогою регулярного виразу.
13. Результати зберігаються в новий Excel файл `prices.xlsx`, де залишаються тільки колонки «Назва категорії» та «Ціна».

3.2 Програмна реалізація методу KNN

Після попередньої обробки датасету, ми вже маємо векторизовані назви товарів з датасету. Далі створюється модель `NearestNeighbors` з бібліотеки `sklearn.neighbors`. В якості метрики подібності обирається косинусна подібність, та вказується кількість сусідів, яку ми хочемо знайти, в нашому випадку ця кількість - 100 товарів. Наступним кроком ініціалізується функція `process_item_name`, яка очищає назву введеного товару (SKU, до якого ми і будемо шукати найближчих сусідів), аналогічно до того, як було зроблено для датасету. Потім створюється функція `find_nearest_items`, яка обчислює вектор для введеного товару і знаходить найближчі сусіди за допомогою моделі KNN. Тут варто зупинитися, та поговорити про косинусну подібність.

Косинусна подібність вже була описана у підрозділі «2.1 Пошук найближчих сусідів до досліджуваного SKU» , вона рахується за формулою (1). Область значень косинусної подібності лежить в межах від -1 до 1. Для нашої задачі вона має наступний вигляд

$$\text{cosine similarity}(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|},$$

де:

A - вектор досліджуваного SKU , B - вектор товару з датасету.

$A \cdot B$ — скалярний (внутрішній) добуток векторів A та B.

$\|A\|$ та $\|B\|$ — норми (довжини) векторів A та B.

θ — кут між векторами A та B.

Граничні значення косинусної подібності:
– Якщо косинусна подібність дорівнює 1, це означає, що кут між двома векторами дорівнює 0 градусів і вектори співнаправлені. Іншими словами, обидва вектори мають однаковий напрямок і є максимально подібними.
– Якщо косинусна подібність дорівнює 0, це означає, що кут між двома векторами дорівнює 90 градусів і вектори ортогональні (перпендикулярні) один до одного. Це свідчить про те, що між векторами немає подібності.
– Якщо косинусна подібність дорівнює -1, це означає, що кут між двома векторами дорівнює 180 градусів і вектори протилежно направлені. Це свідчить про максимальну антиподібність між векторами.

Отже функція `find_nearest_items` знаходить 100 найближчих сусідів, а далі ми встановлюємо порогове значення косинусної подібності 0.4 та відфільтровуємо реузльтати на його основі. Значення 0.4 було обране після тестування, воно дає змогу вірно виокремити найбільш схожі товари. Для всіх проведених тестів, якщо значення косинусної подібності було менше за 0.4, то у всіх випадках це значеннння було максимально найближене до 0, що свідчить про те, що ці товари неварто розглядати, як найближчих сусідів.
Також після отримання результатів, а саме знайдених найближчих сусідів, ми їх аналізуємо. Шукається інформація про підгрупу та групу для кожного знайденого товару, та рахується кількість входжень знайдених товарів до кожної групи та підгрупи. Це ми робимо для того, щоб також окремо визначити лише ті товари з групи та підгрупи серед найближчих сусідів, кількість яких найбільш. Навіщо це робиться було докладно описано у розділі 2 - Теоретична частина.

В результаті ми отримуємо список знайдених найближчих сусідів до за-

даного товару, кількість входжень знайдених товарів до груп та підгруп, де вони містяться, та список найближчих сусідів до заданого товару, кількість входжень яких, до пігруп та груп відповідно, найбільша.

Приклад роботи:

Заданий SKU - «50х90 кухонні рушники».

Результат роботи першої частини програми:

Найближчі сусіди до товару '50х90 кухонні рушники':

- рушники кухонні мікрофібра (косинусна подібність: 0.7694)
- рушники кухонні махрові (косинусна подібність: 0.7417)
- рушники кухонні лляні (косинусна подібність: 0.7110)
- рушники кухонні бавовняні (косинусна подібність: 0.6936)
- рушники для обличчя мікрофібра (косинусна подібність: 0.4403)
- рушники паперові одношарові (косинусна подібність: 0.4365)
- рушники пляжні мікрофібра (косинусна подібність: 0.4268)
- рушники для обличчя махрові (косинусна подібність: 0.4260)
- рушники декоративні Великодні (косинусна подібність: 0.4221)
- рушники паперові Z-складання (косинусна подібність: 0.4160)

Найближчі сусіди до товару '50х90 кухонні рушники' з найбільшим вхожденням до груп та підгруп:

- рушники кухонні мікрофібра (similarity: 0.7694)
- рушники кухонні махрові (similarity: 0.7417)
- рушники кухонні лляні (similarity: 0.7110)
- рушники кухонні бавовняні (similarity: 0.6936)
- рушники декоративні Великодні (similarity: 0.4221)

Кількість входжень знайдених товарів для кожної Підгрупи 1:

- 1.2.1 рушники, серветки кухонні, побутові: 5
- 1.1.4 рушники для особистої гігієни: 3
- 28.5.1 рушники паперові в рулонах: 1
- 28.5.2 рушники паперові листові: 1

Кількість входжень знайдених товарів для кожної Групи:

- 1.2 текстиль кухонний: 5
- 1.1 текстиль домашній: 3
- 28.5 рушники паперові: 2

Тобто ми знайшли 10 найближчих сусідів, значення косинусної подібності вектора яких, по відношенню до вектору заданого SKU більше за 0.4. Також в дужках біля кожного знайденого сусіда виводиться саме значення косинусної подібності. Далі ми порахували кількість входжень знайдених товарів до різних груп, та підгруп. Останнє - це ми виокремили групу знайдених товарів, з найбільшою кількістю входжень до кокретної групи та підгрупи, і для цього варіанту отримали 5 найближчих сусідів.

3.3 Програмна реалізація прогнозування ціни на товар з використаннням GBDT

Використані бібліотеки:

- **numpy** (імпортовано як **np**): Використовується для обробки числових даних та векторизації.
- **pandas** (імпортовано як **pd**): Використовується для роботи з даними у вигляді таблиць.
- **DecisionTreeRegressor** з бібліотеки **sklearn.tree**: Використовується для побудови дерев рішень для задачі регресії.

- **TfidfVectorizer** з **sklearn.feature_extraction.text**: Використовується для перетворення текстових даних у числові вектори на основі частотності з якою слова з'являються у документі.

- **train_test_split** з **sklearn.model_selection**: Використовується для розділення даних на навчальний та тестовий набори.

- **mean_squared_error** і **mean_absolute_error** з **sklearn.metrics**: Використовуються для обчислення середньоквадратичної помилки (MSE) та середньої абсолютної помилки (MAE) відповідно.

Першим реалізовано випадок для відфільтрованих найближчих сусідів (з найбільшою кількістю входжень до підгруп).
– Спочатку імпортуємо бібліотеки вище, далі завантажуємо дані про ціни з файлу `prices.xlsx` та зберігаємо їх у змінній `df_prices`.
– Далі отримуємо список найближчих сусідів `nearest_item_names` із змінної `filtered_nearest_items` (яка містить вже відфільтрованих найближчих сусідів).
– Наступним кроком ми підготовляємо дані для моделювання, а саме назви категорій товарів та їх ціни витягуються з `filtered_nearest_items_df` і зберігаються в `X_filtered` і `y_filtered` відповідно; дані розділяються на навчальний та тестовий набори у пропорції 4:1 за допомогою `train_test_split`.
– На даному кроці ми створюємо клас `GradientBoostingRegressor`, який реалізує алгоритм градієнтного бустингу для регресії. Він має 3 основних методи: `__init__`, `fit`, `predict`. В `__init__` ми встановлюємо початкові значення параметрів моделі, таких як кількість дерев (`n_estimators` = 100), швидкість навчання (`learning_rate` = 0.1) і максимальна глибина кожного дерева (`max_depth` = 3). Метод `fit` призначений для навчання моделі на навчальних даних, він будує ансамбль дерев рішень, починаючи з нульового передбачення та додаванням нових дерев на кожній ітерації. В нашому випадку нульове передбачення - це середнє значення ціни відфільтрованих товарів, а ітеративне навчання дерев реалізовано таким чином, що обчислюються залишки (residuals), які є негативними градієнтами MSE, далі відбувається навчання нового дерева на залишках та додавання нового дерева до ансамблю. Метод `predict` використовує навчену модель для здійснення пе-

редбачень на нових даних, обчислюючи передбачені значення за допомогою ансамблю дерев з врахуванням швидкості навчання.

– Модель навчається на навчальних даних з використанням методу `fit`.

– Прогнози моделі робляться на тестовому наборі за допомогою методу `predict`. Оцінюється точність моделі за допомогою середньоквадратичної помилки (MSE), формула (2) та середньої абсолютної помилки (MAE), (3).

– Далі відбувається прогнозування для нового товару, а саме задається SKU (50x90 кухонні рушники), він векторизується. Після цього ми використовуємо навчену модель `mode_filtered`, яка є екземпляром класу.

`GradientBoostingRegressor`, для прогнозування ціни для цього нового товару. Метод `predict` цього класу обчислює прогнозовану ціну, використовуючи вектор, що ми отримали на попередньому кроці.

– Результатом роботи є набір даних: значення Mean Squared Error (MSE) та Mean Absolute Error (MAE), прогнозована ціна для нового товару.

Для нашого прикладу маємо такі результати:

Показник	**Значення**
Mean Squared Error	8.775585168564922
Mean Absolute Error	2.962361417613476
Прогнозована ціна для нового товару	17.83200621264477

Другим реалізовано випадок для всіх знайдених найближчих сусідів без фільтрування.

Алгоритм прогнозування ціни ідентичний, тільки змінюються вхідні дані, а саме: список найближчих сусідів отримується із змінної `nearest_items` (яка містить всіх знайдених найближчих сусідів), та відповідно інше середнє значення ціни найближчих сусідів.

Для цього випадку маємо наступні результати:

Показник	**Значення**
Mean Squared Error	26.296527809503758
Mean Absolute Error	4.933243360806845
Прогнозована ціна для нового товару	24.714122562051696

Щоб краще розуміти отримані значенні похибок, переведемо їх значення

у відсотки. Для переведення значень MSE і MAE у відсотки скористаємось наступними формулами:

1. Mean Squared Error (MSE) у відсотках:

$$MSE_{\text{proc}} = \left(\frac{MSE}{\text{U}^2}\right) \times 100.$$

2. Mean Absolute Error (MAE) у відсотках:

$$MAE_{\text{proc}} = \left(\frac{MAE}{\text{U}}\right) \times 100,$$

де U - середня ціна знайдених найближчих сусідів.
В результаті отримаємо:

Model	**Mean Squared Error (%)**	**Mean Absolute Error (%)**
Filtered	2.029479	14.245980
Nearest	2.024072	13.686624

Табл. 1: Порівняння MSE та MAE у відсотках для моделей Filtered та Nearest

Додавши сюди значення прогнозованих цін, отримаємо:

Model	**Mean Squared Error (%)**	**Mean Absolute Error (%)**	**Прогнозована ціна**
Filtered	2.029479	14.245980	17.83200621264477
Nearest	2.024072	13.686624	24.714122562051696

Табл. 2: Порівняння MSE та MAE у відсотках для моделей Filtered та Nearest

Модель «Filtered» та модель «Nearest» були оцінені за допомогою двох метрик помилок: середньоквадратичної помилки (MSE) та середньої абсолютної помилки (MAE). Для обох моделей були отримані значення MSE і MAE у відсотках, що відображає їхню відносну точність відносно середнього значення цільової змінної.

Модель «Filtered» показала значення MSE та MAE на рівні приблизно 2.03% та 14.25% відповідно. Прогнозована ціна для нового товару за цією моделлю складає близько 17.83.

Модель «Nearest» показала значення MSE та MAE на рівні приблизно 2.02% та 13.69% відповідно. Прогнозована ціна для нового товару за цією моделлю складає близько 24.71.

Отже, за цими метриками можна зробити висновок, що модель «Nearest» показала кращі результати, оскільки має нижчі значення MSE та MAE, а також більш точний прогноз для нового товару. Але варто зазначити те, що в самі прогнозовані ціни відрізняються між собою.

Висновки

В результаті даної кваліфікаційної роботи було побудовано модель для прогнозування ціни на товар з холодним стартом на основі ціни товарів найближчих сусідів за допомогою методу GBDT. Пошук найближчих сусідів виконується за допомогою методу KNN. Прогноз ціни відбувається для двох різних моделей, а саме для двох різних наборів товарів (знайдених найближчих сусідів до заданого товару). До першого набору найближчих сусідів входять всі товари з датасету, які були знайдені за допомогою методу KNN. До другого набору найближчих сусідів входять лише товари з тієї категорії, кількість входжень товарів до якої є найбільшою серед всіх знайдених найближчих сусідів. Це є важливою складовою, оскільки через швидкий зріст ринку та збільшенні дати, для менеджера магазину необхідно отримувати прогноз ціни на новий товар, як опираючись лише на товари з конкретної категорії, так і взагалому на всі товари, які є сусідніми до заданого.

Результати, які ми отримали в ході даної роботи показали, що побудовані моделі забезпечують достатньо високу точність прогнозування ціни товару з холодним стартом. Також варто зазначити, що значення похибок для різних наборів найближчих сусідів схожі, хоча модель «Nearest» все-таки демонструє вищу точність, а самі значення прогнозованої ціни відрізняються. Саме це і пояснює новизну нашої роботи: для товару без історії продажів ми спрогнозували дві різні ціни, що дає умовному менеджеру магазину краще розуміння при прийнятті рішень про те, на основі яких товарів формувати ціну новогу, та дає ширший спектр можливої ціни.

До перспектив дослідження можна віднести адаптація моделей для зарубіжного ринку товарів, удосконалення моделей, використання додаткових характеристик товарів.

Список використаних джерел

Aumüller, M., Bernhardsson, E., and Faithfull, A. (2020). Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Information Systems*, 87:101374.

Bailey, T. and Jain, A. (1978). A note on distance-weighted k-nearest neighbor rules. *IEEE Transactions on Systems, Man, and Cybernetics*, 8:311–313.

Bermejo, S. and Cabestany, J. (2000). Adaptive soft k-nearest-neighbour classifiers. *Pattern Recognition*, 33(12):1999–2005.

Breiman, L. (2017). *Classification and regression trees*. Routledge.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27.

Dudani, S. A. (1976). The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, (4):325–327.

Fix, E. (1985). *Discriminatory analysis: nonparametric discrimination, consistency properties*, volume 1. USAF school of Aviation Medicine.

Friedman, J. H. (1999). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Fukunaga, K. and Hostetler, L. (1975). K-nearest-neighbor bayes-risk estimation. *IEEE Transactions on Information Theory*, 21(3):285–293.

Harrington, P. (2012). *Machine learning in action*. Simon and Schuster.

Hastie, T., Tibshirani, R., Friedman, J. H., and Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer.

Hellman, M. E. (1970). The nearest neighbor classification rule with a reject option. *IEEE Transactions on Systems Science and Cybernetics*, 6(3):179–185.

Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE.

Indyk, P. and Motwani, R. (1998). Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613.

Jóźwik, A. (1983). A learning scheme for a fuzzy k-nn rule. *Pattern Recognition Letters*, 1(5-6):287–289.

Keller, J. M., Gray, M. R., and Givens, J. A. (1985). A fuzzy k-nearest neighbor algorithm. *IEEE transactions on systems, man, and cybernetics*, (4):580–585.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Lattimore, T. and Szepesvári, C. (2020). *Bandit algorithms*. Cambridge University Press.

Peterson, L. E. (2009). K-nearest neighbor. *Scholarpedia*, 4(2):1883.

Qian, T., Liang, Y., and Li, Q. (2019). Solving cold start problem in recommendation with attribute graph neural networks. *arXiv preprint arXiv:1912.12398*.