Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра математики

Кваліфікаційна робота

освітній ступінь – бакалавр

на тему: **«МОДЕЛЬ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ ПРОДАЖІВ З ПЕРЕВАНТАЖЕННЯМ»**

Виконала: студентка 4-го року навчання

освітньої програми «Прикладна математика»,

спеціальності 113 Прикладна математика

Крючкова Анастасія Сергіївна

Керівник: Дрінь С.С.,

ст. викладач, к.н.

Рецензент_____

(прізвище та ініціали)

Кваліфікаційна робота захищена

з оцінкою _____

Секретар ЕК _____

«____» _____ 2022 р.

Київ 2022

Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Кафедра математики факультету інформатики

ЗАТВЕРДЖУЮ

Зав. кафедри математики,

проф., д.ф. - м.н.

_____ Б. В. Олійник

(підпис)

“____”_____2022 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на кваліфікаційну роботу

студентці 4-го курсу факультету інформатики

Крючковій Анастасії Сергіївні

Тема: Модель рекомендаційної системи продажів з перевантаженням
Вихідні дані: Досліджено та модифіковано модель градієнтного бустингу LightGBM

Зміст ТЧ до кваліфікаційної роботи:

Індивідуальне завдання
Календарний план
Перелік прийнятих скорочень
Анотація
Вступ
Опис задачі, що потребує модель рекомендаційної системи
Модель градієнтного бустингу LightGBM, ключові метрики продуктивності
Програмна реалізація
Висновки
Використана література

Дата видачі “___” _____ 2022 р. Керівник _____

(підпис)

Завдання отримав_____

(підпис)

Тема: Модель рекомендаційної системи продажів з перевантаженням

Календарний план

№ п/п	Назва етапу кваліфікаційної роботи	Термін виконання етапу	Примітка
1.	Визначення теми кваліфікаційної роботи.	29.10.2021	
2.	Визначення літератури для роботи.	16.03.2022	
3.	Огляд і аналіз літератури за темою роботи.	20.03.2022	
4.	Оформлення теоретичної частини кваліфікаційної роботи.	31.03.2022	
5.	Виконання практичного застосування вибраного методу.	31.04.2022	
6.	Оформлення практичної частини кваліфікаційної роботи.	15.05.2022	
7.	Створення презентації кваліфікаційної роботи.	30.05.2022	
8.	Захист кваліфікаційної роботи.	20.06.2022	

Студент Крючкова Анастасія Сергіївна

Керівник Дрінь Світлана Сергіївна

“31 травня**”** 2022р.

ЗМІСТ

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ 2

Календарний план 3

ЗМІСТ 4

Перелік прийнятих скорочень 5

Анотація 6

Вступ 6

Опис задачі, що потребує модель рекомендаційної системи 8

- Фактори впливу на попит 8
- Дерева рішень 8
- Діф таргет 9
- Переривчастий попит 10
- Прогнозування продажів товарів, у яких немає історії 10

Модель градієнтного бустингу LightGBM 12

- Опис алгоритму 12
- Ключові метрики продуктивності алгоритму 25
- Додавання додаткових фіч для моделі 27

Програмна реалізація 29

- Огляд даних 29
- Валідація 31
- Тривіальне рішення (базовий підхід) 31
- Model-based базове рішення 32
- Додаткові ознаки(фічі) 33
- Meta Feature 34
- Фінальне рішення 35

Висновки 36

Використана література 37

Перелік прийнятих скорочень

Рітейл — роздрібна торгівля, процес продажу продукції або послуг кінцевим споживачам.

SKU — Stock Keeping Unit, одиниця зберігання запасів, окремий тип товару для продажу, придбання або відстеження в запасах, і всі атрибути, пов’язані з типом товару, які відрізняють його від інших типів товарів.

GBM — Gradient Boosting Machine, техніка машинного навчання, яка використовується серед інших у задачах регресії та класифікації.

LightGBM — Light Gradient Boosting Machine, інформаційна методологія навчання, яка належить до класу алгоритмів градієнтного бустингу та використовує алгоритм навчання, який базується на регресійних деревах.

DT — Decision tree, це інструмент підтримки прийняття рішень, який використовує деревоподібну модель рішень та їх можливих наслідків, включаючи випадкові результати подій, витрати ресурсів та корисність.

Діф таргет — difference target, цільовий масив, який маємо передбачити.
MAE — Mean Absolute Error, Середня абсолютна похибка.

MSE — Mean Square Error, Середня квадратична помилка.

MAPE — Mean Absolute Percentage Error, Середня абсолютна відсоткова помилка.

MAE — Mean Absolute Error, Середня абсолютна помилка.

Анотація

У даній роботі описано типову задачу рітейлу з прогнозування продажів та розглянуто власну модель рекомендаційної системи продажів з перевантаженням, а саме модифіковану модель градієнтного бустингу LightGBM.

Описано фактори впливу на попит споживачів та відповідно на прогнозування товарів. Розглянуто основні проблеми, що виникають під час прогнозування та ключові рішення, що допомагають у створенні моделей.

Описано основні кроки побудови власної моделі, ключові метрики продуктивності алгоритму та імпортування додаткових ознак, таких як ембеддінг вектори. Наведено приклад застосування моделі для щоденних продажів та цін електронної комерції по Україні, із заданими параметрами. Проведено аналіз отриманих результатів.

Вступ

Електронна комерція завойовує домінування на світовому ринку роздрібної торгівлі: цього року світові продажі електронної комерції вперше перевищать 5 трильйонів доларів, що складає більш ніж п’яту частину загальної роздрібної торгівлі[1]. Бізнес електронної комерції — це створення для клієнтів пропозицій експрес-доставки їжі або непродовольчих товарів за розкладом. Важливо вміти передбачити тенденції поведінки споживачів, оскільки вони допомагають інформувати про важливі операції, такі як планування запасів. Перевищення запасів призводить до непотрібних фінансових витрат, наприклад їх збільшення на зберігання, та порушення екологічної стійкості, тоді як недостатня кількість запасів спричиняє як втрату потенційного доходу, так і незадоволені потреби споживачів. Тому є надзвичайно важливим точне прогнозування попиту, щоб мати змогу оптимізовувати планування запасів.

Власне саме ця проблема і зумовила мету моєї роботи, яка полягає у тому, щоб створити модель, яка дозволить спрогнозувати продажі електронної комерції

з перевантаженням.

Поставлена мета зумовила наступне наукове завдання:

1. У розділі 1. описати основну задачу електронної комерції, що потребує модель рекомендаційної системи та основні проблеми, що виникають під час прогнозування продажів, а саме враховування факторів впливу на попит, переривчатий попит та прогнозування продажів товарів, у яких немає історії.
2. У розділі 1. розглянути основні підходи до моделювання і прогнозування продажів та планування запасів, а саме дерева рішень, діф таргет та ембеддінг.
3. У розділі 2. описати метод градієнтного бустингу LightGBM та ключові метрики продуктивності алгоритму.
4. Після теоретичного моделювання у розділі 3. розробити алгоритм програмного продукту, який спрогнозує продажі та ціни електронної комерції по Україні, які доступні з початку 2020 року і до середини липня 2021 року. для та зробити відповідні висновки.

1. Опис задачі, що потребує модель рекомендаційної системи

Точне прогнозування продажів має вирішальне значення для роздрібних компаній, щоб виробляти необхідну кількість у потрібний час.

Кожен окемий товар має окремий номер SKU, що є штрих-кодом товару. Маючи історію продажів для багатьох комбінацій SKU, магазин, день та кількість, нашим завданням є створити прогноз продажів для всіх товарів у кожному магазині у визначеній кількості. На прикладі даних продажів з української роздрібної корпорації, яка має 56 000 SKU - одиниць складського обліку, та більше 300-т магазинів по Україні, в день нараховує 17 млн. комбінацій продажів, тобто маємо зробити прогнози на кожен день у горизонті прогнозів.

1.1. Фактори впливу на попит

Для того, аби зробити хороший прогноз, треба думати як споживач і враховувати фактори, що впливають на його рішення купити чи не купити певний продукт. Кількість факторів, що впливають на рішення споживача, завжди залежатиме від індустрії та групи товарів, для якої розраховується оптимальна ціна. Основні: історія продажів, залишки, промо-календарі, погода, курс валют, розташування магазинів конкурентів, середня ціна полиці/категорії, крос-цінова еластичність попиту на товари у портфелі (зміна попиту на один товар, у випадку зміни ціни іншого товару), та навіть погода чи такі фактори як пандемія Covid.

1.2. Дерева рішень

Дерева рішень (DT, decision trees) – це непараметричний метод навчання з наглядом, який використовується для класифікації та регресії [10]. Мета полягає в тому, щоб створити модель, яка прогнозує значення цільової змінної, вивчаючи прості правила прийняття рішень, виведені з характеристик даних.

Дерево рішень зазвичай починається з одного вузла, на основі якого

дерево розбивається на гілки та розгалужується до можливих результатів. Кожен із цих результатів призводить до додаткових вузлів, які розгалужуються до інших можливостей. Кінець гілки, який більше не розділяється, — це рішення/лист. Це надає йому деревоподібну форму.

Дерева рішень можуть моделювати складні взаємозв'язки між різними факторами. Але дерева (ліс дерев) не здатні моделювати тренд: вони не можуть передбачити нічого, чого вони не бачили в історичних даних. Хоча якщо проста лінійна регресія з цією задачею справляється, то дерева — ні.

1.3. Діф таргет

Для величин з трендом варто використовувати діф таргет (difference target — цільовий масив, який маємо передбачити) з даних відмінною рисою якого є те, що у статистичних термінах це залежна змінна. У рітейлі часто можна спостерігати сильний тренд на продукти. Ми можемо прогнозувати звичайні продажі: скільки продасться якогось товару в день, а можемо прогнозувати приріст або різницю між продажами минулого та наступного днів.

Vanilla target представляє собою продажі певного товару в час t: $sales_t$.

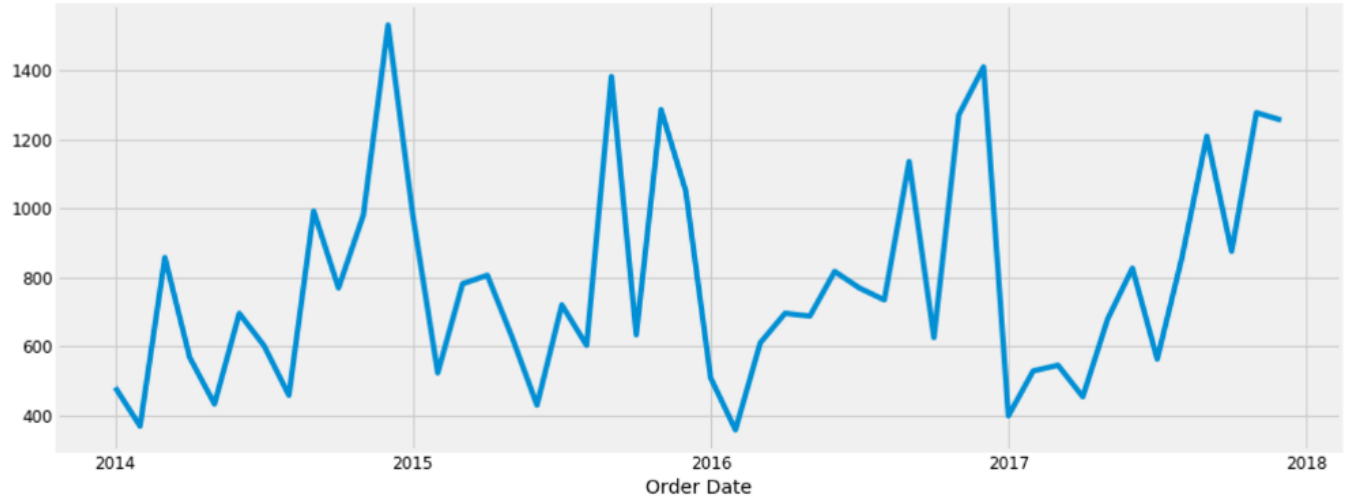

рис.1.1 Vanilla target

Difference target — приріст продажів в час t порівняно з $(t - 1)$: $sales_t - sales_{t-1}$.

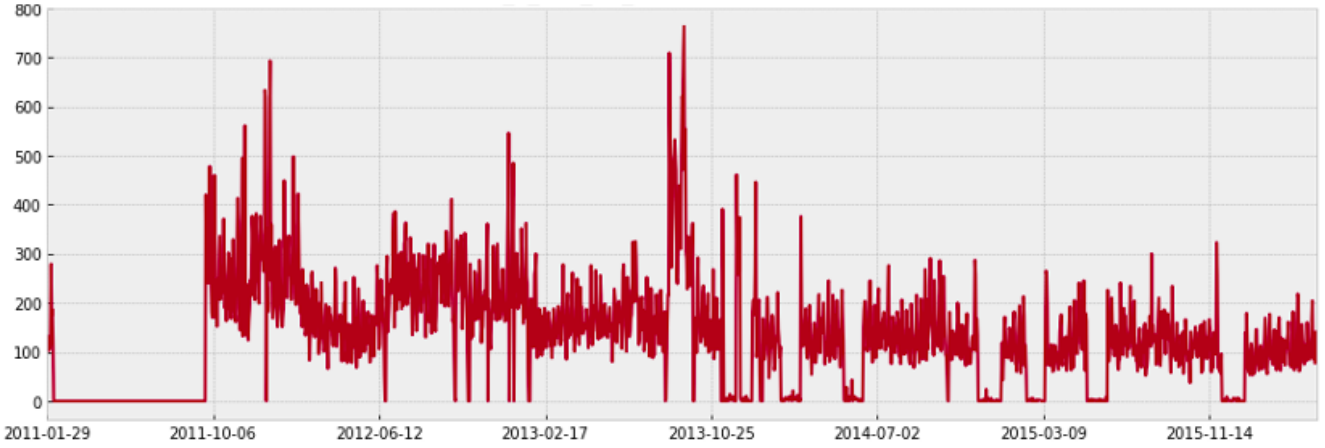

рис.1.2 Difference target

1.3.1. Переривчастий попит

Друга проблема, яка виникає в рітейлі — це переривчастий попит (intermittent demand). Періодичний попит виникає, коли продукт відчуває кілька періодів нульового попиту. Часто в таких ситуаціях, коли виникає попит, він невеликий, а іноді і дуже мінливий за розміром.

Ми не завжди знаємо походження нуля в історичних даних рітейлу. Товару не було на полиці чи він був і ніхто його не купував. Довгі серії нулів між високо амплітудними значеннями, як правило, вказують на перше.

Більшість класичних та ML моделей мають великий негативний bias (недопрогноз для днів, де продажі були):

$$forecast\ bias = (\text{факт продажу} - \text{прогноз})/\text{прогноз}\ (\%)$$

Рішенням часто є викинути довгі серії нулів з тренувальних даних.

1.4. Прогнозування продажів товарів, у яких немає історії

Якщо в магазині з’являється новий продукт, який раніше продавався у інших магазинах даної компанії, то ми тренуємо модель на даних всіх магазинів. Якщо в мережі з’являється продукт з новими властивостями, то маємо тренувати модель на даних для всіх категорій товарів, де категорією товару є група подібних товарів, що мають схожі характеристики. Варто додавати фічі (ознаки), які будуть описувати тип продукту та класифікувати його, натомість як подавати в модель такі фічі як id товару та id магазину вважається поганою практикою, оскільки їхній вплив буде переважати і відбудеться поглинання впливу, тобто нівелювання впливу, інших зазначених фіч, і модель буде гірше передбачати попит для нових товарів.

Також у випадку прогнозування товарів, що не мають історії продажів, можливе використання ембедінгу.

Ембеддінг в NLP (обробка природної мови) означає процес або, частіше, результат процесу перетворення мовної сутності – слова, речення, параграфа чи цілого тексту на набір чисел – числовий вектор. Основна мета використання ембеддінгу — відстань між векторами товарів, що мають схожу природу продажів має бути мінімальною [5].

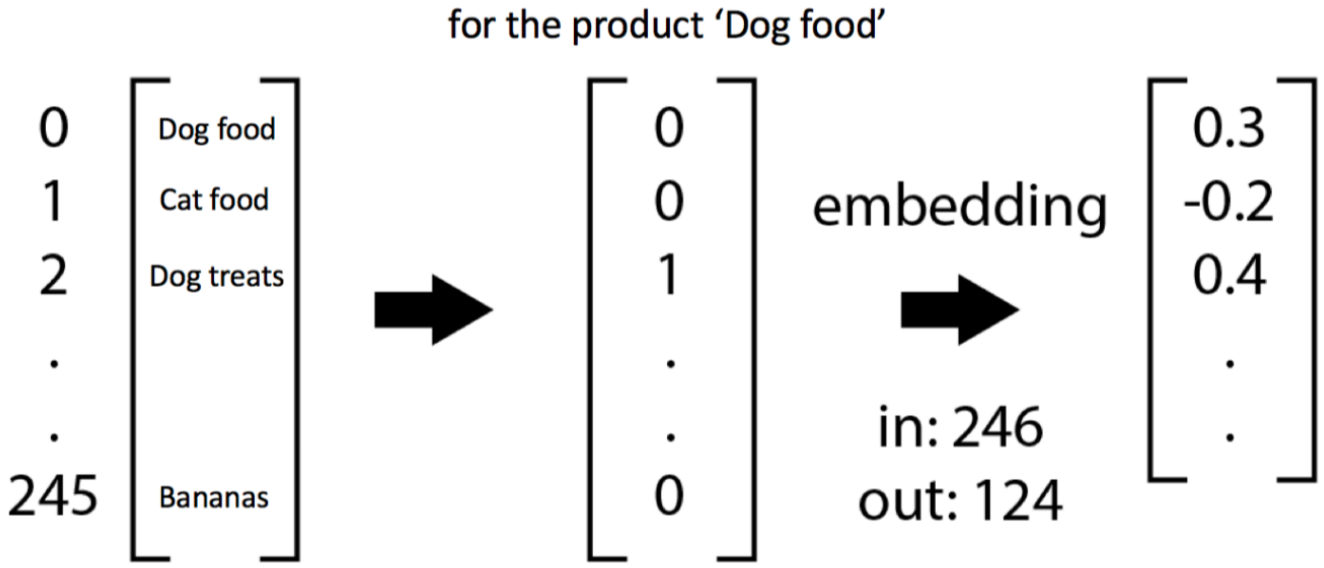

рис. 1.3. Приклад ембеддінгу

Алгоритм роботи ембедінгу наступний:

1. Подаємо на вхід загальні характеристики товару, що не залежать від історії продажу: Product#1 features, Product#2 features, Product#2 features.

2. Ембеддінг створює репрезентацію товару, яку вивчила нейронна мережа NLP, і цей вектор використовується як фіча (ознака) у моделі.

3. На виході отримуємо прогноз на продажі товару по місяцях/днях тижня: Product#1 sales, Product#2 sales, Product#2 sales.

Ембедінги допомагають знаходити паттерни. Завдяки ним можемо сказати, які продукти є більш схожими між собою. Ембеддінги подаються в моделі як фічі.

2. Модель градієнтного бустингу LightGBM

2.1. Опис алгоритму

Матеріал цього й наступного підрозділів має реферативний характер. Використовуються теоретичні положення з робіт [2], [4] та [11].

LightGBM (Light Gradient Boosting Machine) — це інформаційна методологія навчання, яка належить до класу алгоритмів градієнтного бустингу та використовує алгоритм навчання, який базується на регресійних деревах[3]. Бустинг (boosting) — це процедура послідовної побудови композиції з алгоритмів машинного навчання, у якій кожен наступний алгоритм прагне компенсувати недоліки композиції всіх попередніх алгоритмів[13].

Дерево рішень — це алгоритм машинного навчання з наглядом, який виглядає як перевернуте дерево, де кожен вузол представляє змінну-провісник (predictor variable, feature), зв'язок між вузлами представляє рішення, а кожен вузол лівої частини — результат (змінна відповіді). Ми хочемо передбачити відповідь або клас y_i (response) з вхідних даних x_i (inputs). Ми робимо це шляхом вирощування бінарного дерева. Розглянемо дані, що складаються з p вхідних даних і змінною відповіді для кожного з N спостережень: тобто набір форми $\left(x_i, y_i\right)$ для $i = 1, 2, ..., N$ з $x_i = \left(x_{i1}, x_{i2}, ..., x_{ip}\right)$. Алгоритм повинен автоматично визначити змінні розщеплення та точки розщеплення, а також топологію (форму) дерева. Побудова дерева регресії визначається наступним чином:

1. Набір значень цільової змінної відповіді y_i ділиться на M областей $R_1, R_2, ..., R_M$, відомі як кінцеві вузли або листя дерева.
2. Змінна відповіді моделюється як константа c_m у кожній області так, що: $f(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$, де I — індикаторна функція, що пробігається по значеннях в даному регіоні.

Маючи за критерій мінімізації суми квадратів $\sum\left(y_i - f\left(x_i\right)\right)^2$, можемо розрахувати оптимальне $\hat{c}_m$, яке є середнім значенням змінної відповіді y_i в області m:

$$\hat{c}_m = average\left(x_i | y_i \in R_m\right)$$

Проблема, яка виникає, полягає в тому, що використання суми квадратів для пошуку найкращих результатів робить алгоритм надзвичайно трудомістким. З цієї причини зазвичай використовується підхід, згідно з яким на кожному кроці цільова змінна поділяється на дві області через дві гілки, вибирається змінна розщеплення j та точка поділу s, що в результаті дає найбільше зменшення суми квадратів. Знайшовши змінну j та точку s, та визначивши пару півплощин:

$$R_1(j,s) = \left\{X | X_j \leq s\right\} \text{ та } R_2(j,s) = \left\{X | X_j > s\right\},$$

можемо мінімізувати наступну функцію:

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i \in R_1(j,s)}\left(y_i - c_1\right)^2 + \min_{c_2}\sum_{x_i \in R_2(j,s)}\left(y_i - c_2\right)^2\right].$$

Для будь-якого вибору значень j та s, внутрішня мінімізація вирішується за допомогою:

$$\hat{c}_1 = ave\left(y_i | x_i \in R_1(j,s)\right) \text{ та } \hat{c}_2 = ave\left(y_i | x_i \in R_2(j,s)\right).$$

Для кожної змінної розщеплення визначення точки розщеплення s може бути виконано дуже швидко, і, отже, шляхом сканування всіх вхідних даних, можна визначити найкращу пару (j, s).

Знайшовши найкращий поділ, ми розбиваємо дані на дві результуючі області та повторюємо процес поділу для кожної з двох областей. Потім процес повторюється для кожної створеної області.

Розмір дерева є параметром налаштування, який регулює складність моделі, і оптимальний розмір дерева слід адаптивно вибирати з даних. Завелике дерево може переповнювати дані, тоді як маленьке дерево може не охоплювати важливу структуру. Найкраща стратегія — вирощувати велике дерево T_0 з

попередньо визначеною кількістю вузлів, та зупинити процес розщеплення лише тоді, коли досягається деякий мінімальний розмір вузла. Потім це велике дерево обрізати, використовуючи критерій, заснований на складності дерева (за допомогою обрізки витрат і складності). Опис алгоритму наведений нижче:

Ми визначаємо піддерево $T \subset T_0$ як будь-яке дерево, яке можна отримати шляхом обрізання T_0, тобто згортання будь-якої кількості його внутрішніх (нетермінальних) вузлів. Індексуємо кінцеві вузли на m, при цьому вузол m представляє область R_m. Нехай $|T|$.позначаємо кількість кінцевих вузлів у T, тоді:

$$\hat{c}_m = \frac{1}{N_m} \sum_{x_i \in R_m} y_i$$

$$Q_m(T) = \sum_{x_i \in R_m} \left(y_i - \hat{c}_m\right)^2$$

визначаємо критерій складності вартості (cost complexity criterion):

$$Ca(T) = \sum_{m=1}^{|T|} Q_m(T) + a|T|$$

Перший параметр функції Ca вимірює, наскільки добре дерево адаптується до даних, на яких навчається (малі значення вказують на хорошу адаптацію), другий параметр вимірює складність дерева. Параметр $\alpha \geq 0$ вказує на протилежність між складністю та гарною адаптацією дерева. Для $\alpha = 0$ отримане дерево є T_0, оскільки для кожного вузла, включеного в дерево, вартість не додається. При зростанні параметра α, збільшується вартість складності дерева, тому це призводить до менших дерев, які погано адаптуються до даних, на яких вони навчаються. Чим менший параметр α, тим більше створюється дерево, що часто призводить до перенавчання на початкових даних і, отже, до поганої продуктивності для інших наборів даних.

Як зазначалося вище, LightGBM — це алгоритм градієнтного бустингу. Техніка Boosting заснована на створенні послідовних дерев. Кожне дерево навчається на основі інформації з попередніх дерев. Алгоритм працює наступним чином:

1. Для кожного спостереження в наборі навчальних даних задаються $f(x) = 0$ та $\varepsilon_i = y_i$.
2. У кожному раунді *k* тренується дерево $\hat{f}^k$ з *d* вузлів, яке має в якості змінної відповіді залишки операції, що залишилися від попереднього раунду регресії, які позначаються ε_i.
3. Додано обрізану версію нового дерева, для того, аби:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^k(x)$$

4. Відповідно:

$$\varepsilon_i \leftarrow \varepsilon_i - \lambda \hat{f}^k(x)$$

5. Повторення процесу з другого кроку *K* разів (параметр *K* визначається користувачем) остаточна форма моделі виходить:

$$\hat{f}(x) = \lambda \sum_{k=1}^{K} \hat{f}^k(x)$$

Аби техніка Boosting була ефективною, користувач повинен вказати кількість дерев, які потрібно створити: параметр λ і кількість вузлів у кожному дереві. Велика кількість дерев може легко бути надмірно адаптованою до навчальних даних, що призводить до поганої здатності до узагальнення. Параметр λ визначає, наскільки швидко модель буде навчатися. Типові значення λ від 0,001 до 0,1. Кількість вузлів контролює складність кожного дерева. Часто дерева одного поділу, також відомі як гілки, є задовільними, оскільки навчання в моделі відбувається повільно та контрольовано.

Техніка Gradient Boosting є розширенням техніки Boosting, що поєднує два методи, алгоритм Gradient Descent і техніку Boosting. Градієнтний спуск (Gradient Descent) — це метод оптимізації. Щоб знайти загальний мінімум функції за допомогою цього методу, спочатку обчислюється її похідна, а потім використовується обернений процес знаходження похідної. Похідна вимірює, наскільки зміниться значення функції $J(\theta)$, якщо змінна θ зазнає незначної зміни, що є, по суті, нахилом функції. Високі значення функції вказують на великий нахил і, отже, велика зміна значення (θ) для малих змін θ. Цей алгоритм є ітеративним, а саме він ініціалізує випадкове значення в θ, обчислює похідну функції в даній точці та модифікує θ так, щоб:

$$\theta = \theta - \rho\frac{dj}{d\theta}$$

де параметр ϱ визначає, наскільки швидко він буде рухатися у від'ємному напрямку похідної. Від'ємним напрямком похідної — протилежний напрямок для похідної за певним напрямком. Процес повторюється до тих пір, поки алгоритм не збіжиться.

У випадку підсилення градієнта алгоритм пропонує навчання дерев для похідної функції втрат у від'ємному напрямку похідної. Наприклад, взявши за функцію втрат суму квадратів залишків ε_i, поділену на 2 так, що:

$$L\left(y_i, \hat{y}_i\right) = \frac{1}{2}\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2$$

Візьмемо похідну:

$$\frac{dL\left(y_i,\hat{y}_i\right)}{d\hat{y}_i} = \hat{y}_i - y_i$$

Тобто від'ємна похідна функції втрат дорівнює залишкам ε_i. Отже, по суті, процес включає навчання дерева на основі залишків ε_i, до яких додається обрізана версія нового дерева. Таким чином, техніка Gradient Boosting додає

послідовні дерева в будь — який момент часу *t* до від'ємної похідної функції втрат, щоб:

$$\hat{y}_i^{(t)} = \sum_{t=1}^{K} f_t(x_i), f_t \in F$$

де $F = \{f(x) = w_{q(x)}\}$ та $q: R^m \to T, w \in R^T$ такі, що q представляє структуру кожного дерева, T представляє кількість листків, і кожна f_t відповідає незалежній деревоподібній структурі q з вагою листків, що позначається як w. У техніці LightGBM поєднуються дерева різної структури q, причому структура кожного дерева — це кількість створених вузлів. Функція втрат, яка мінімізується в будь — який момент *t*, визначається формулою:

$$L^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{\,t}\right) + \sum_{k=1}^{T} \Omega_{f(t)}$$

Перший параметр вимірює, наскільки добре модель адаптується до навчальних даних (малі значення вказують на хорошу адаптацію), а другий параметр вимірює складність кожного дерева, де на додаток до кількості листків (T) вводиться нова змінна, яка призводить до зменшення ваги листя:

$$\Omega_{f(t)} = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

Параметр γ вказує значення штрафу (штрафні методи — це певний клас алгоритмів для розв'язування задач оптимізації) за зростання дерева, так що великі значення параметра γ призведуть до малих дерев і малих значень γ. Збільшення значення параметра λ призводить до зменшення ваги дерев. Таким чином:

$$\hat{y}_i^{(t)} = \sum_{t=1}^{K} f_t(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)$$

Отже, проблема полягає в тому, щоб визначити, яка $f_t(x_i)$ мінімізує функцію втрат у момент часу *t*:

$$L^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t)}\right) + \sum_{k=1}^{T} \Omega_{f(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t\left(x_i\right)\right) + \sum_{k=1}^{T} \Omega_{f(t)}$$

З розширення Тейлора в степеневий ряд випливає, що:

$$f(x + \Delta x) \cong f(x) + f'(x)\Delta x + \frac{1}{2} f''(x)(\Delta x)^2$$

Отже, маємо наступне співвідношення:

$$L^{(t)} \cong \sum_{i=1}^{n} \left[l\left(y_i, \hat{y}_i^{(t-1)}\right) + g_i f_t\left(x_i\right) + \frac{1}{2} h_i f_t^2\left(x_i\right) \right] + \Omega_{f(t)}$$

де $g_i = d_{\hat{y}_i^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right)$ та $h_i = d^2_{\hat{y}_i^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right)$.

Позбуваючись констант, функція втрат набуває вигляду:

$$L'^{(t)} \cong \sum_{i=1}^{n} \left[g_i f_t\left(x_i\right) + \frac{1}{2} h_i f_t^2\left(x_i\right) \right] + \Omega_{f(t)}$$

Встановивши набір спостережень на листку j як $I_j = \left\{ i | g\left(x_i\right) = j \right\}$, вищезгадане співвідношення набуває наступного вигляду:

$$L'^{(t)} \cong \sum_{i=1}^{n} \left[g_i w_q\left(x_i\right) + \frac{1}{2} h_i w_q^2\left(x_i\right) \right] + \Omega_{f(t)} = \sum_{i=1}^{T} \left[\left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma T$$

Якщо мають місце рівності $G_j = \sum_{i \in I_j} g_i$ та $H_j = \sum_{i \in I_j} h_i$, то отримаємо наступне співвідношення:

$$L'^{(t)} = \sum_{i=1}^{T} \left[G_j w_j + \frac{1}{2} \left(H_j + \lambda \right) w_j^2 \right] + \gamma T$$

Припускаючи, що структура дерева $(q(x))$ відома, оптимальна вага кожного листка отримується шляхом мінімізації вищенаведеного співвідношення щодо w_j, так що:

$$w_j = - \frac{G_j}{H_j + \lambda}$$

Згодом, замінивши w_j, виходить наступне рівняння, яке також обчислює якість структури нового дерева:

$$L^{'(t)} = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j+\lambda} + \gamma T$$

Нарешті, алгоритм поділу базується на наступній функції:

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{\left(G_L+G_R\right)^2}{H_L+H_R+\lambda}\right] - \gamma$$

де перший аргумент — це оцінка лівої частини поділу, другий аргумент — це оцінка правої частини поділу, третій — це оцінка у випадку, якщо розподіл не відбувається, і γ вимірює вартість складності поділу.

Процес вирішення проблеми починається зі створення дерева та вирощування його до певної глибини, визначеної користувачем. Дерево обрізається з негативним коефіцієнтом підсилення Gain, а потім до моделі додається усічена версія нового дерева. Процедура повторюється K разів (K: параметр, визначений користувачем). Важливо відзначити, що алгоритм LightGBM, який характеризується своєю ефективністю, точністю та швидкістю, створює гістограми та використовує згенеровані класи замість усього діапазону значень кожної змінної, досягаючи значного скорочення часу навчання. Він також росте вертикально, а це означає, що він росте на рівні листка (рис.), тоді як інші алгоритми ростуть на глибині (рис.).

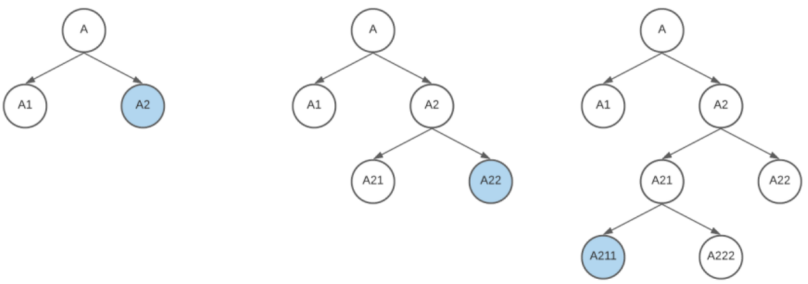

рис.2.1. Листовидний метод

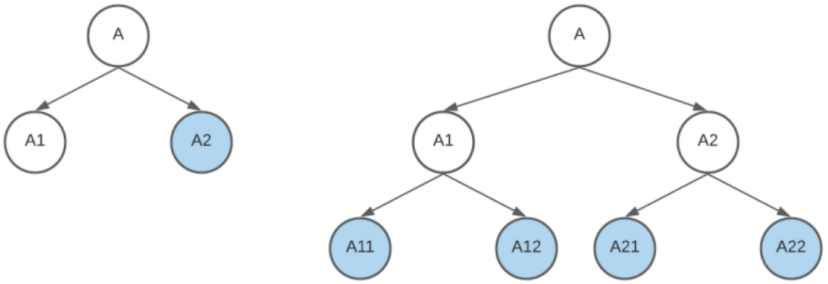

рис.2.2. Глибинний метод

Під час росту дерева по листу алгоритм стає дуже ефективним, оскільки дозволяє значно зменшити втрати, таким чином отримати точність, при цьому процеси регресії завершуються швидко.

Ще одна важлива особливість, яка робить LightGBM одним з найбільш повних і поширених алгоритмів машинного навчання, полягає в тому, що він використовує не всі навчальні дані, а їх вибірку, яка є результатом методу Gradient One Side Sampleing (GOSS). Основна ідея методології GOSS зосереджується на тому факті, що не всі спостереження однаково сприяють навчанню алгоритму, оскільки ті, у яких перша похідна функції вартості невелика, краще навчаються, ніж ті, у яких перша похідна функції вартості велика. Ігнорування спостережень з малою похідною призводить до упередженості у даних вибірки та у певній зміні в розподілі даних, що в свою чергу призводить до поділу, більшого за оптимальний, і до очевидної надмірної адаптації моделі до вибірки. Для вирішення проблеми вибираються випадкові спостереження з малою похідною функції вартості, які сортуються за абсолютним значенням їх похідної: $\alpha \times 100\%$ з найбільшою похідною і $b \times 100\%$ з решти. Для розрахунку функції втрат спостереження з малою похідною множать на $\frac{1-\alpha}{b}$, таким чином надаючи більшого значення погано навченим, без істотної диференціації розподілу даних. Навчаючи лише одну вибірку на кожній ітерації, досягається значне збільшення процесу навчання алгоритму, що призводить до його швидкої збіжності до оптимального рішення.

GBDT (gradient boosted decision trees) використовує дерева рішень для вивчення функції від вхідного простору X^s до градієнтного простору G[1]. Зокрема, для навчальної множини T з n випадків, таких що $T = \{x_1, x_2, ..., x_n\}$, де кожен x_i є вектором з розмірністю s у просторі X^s. На кожній ітерації алгоритму градієнтного бустингу, від'ємні нахили функції вартості по відношенню до результату моделі позначаються як $G = \{g_1, g_2, ..., g_n\}$. Модель дерева рішень розбиває кожен вузол за найбільш інформативним ознакою (з найбільшим приростом інформації). Для GBDT приріст інформації зазвичай вимірюється дисперсією після поділу, яка визначається, як показано нижче.

Означення. Нехай O — навчальний набір даних на фіксованому вузлі дерева рішень. Підсилення дисперсії ознаки розщеплення j в точці d для цього вузла визначається як:

$$V_{j|O}(d) = \frac{1}{n_O}\left(\frac{\left(\sum_{\{x_i \in O: x_{ij} \le d\}} g_i\right)^2}{n^j_{l|O}(d)} + \frac{\left(\sum_{\{x_i \in O: x_{ij} > d\}} g_i\right)^2}{n^j_{r|O}(d)}\right),$$

де $n_O = \sum I[x_i \in O], n^j_{l|O}(d) = \sum I[x_i \in O: x_{ij} \le d]$ та $n^j_{r|O}(d) = \sum I[x_i \in O: x_{ij} > d]$.

Для ознаки j алгоритм дерева рішень вибирає $d^*_j = argmax_d V_j(d)$ і обчислює найбільше посилення $V_j(d^*_j)$.

Потім дані поділяються відповідно до ознаки j^* в точці d_{j^*} в лівий і правий дочірній вузол.

Реалізуючи метод GOSS, випадки класифікуються за абсолютними значеннями їхніх ступенів у порядку спадання. Таким чином, створюється множина A з $\alpha \times 100\%$ випадків з більшим кутовим коефіцієнтом нахилу, множина A^c, що складається з $(1 - \alpha) \times 100\%$ випадків з меншим кутовим

коефіцієнтом нахилу, та підмножина B з розмірністю $b \times |A^c|$. Потім усі випадки класифікуються відповідно до оціненої вартості дисперсії у векторі $\tilde{V}_j(d)$ на множині $A \cup B$, на ознаці j, із заданою точкою поділу d, як показано нижче:

$$\tilde{V}_j(d) = \frac{1}{n}\left(\frac{\left(\sum_{x_i \in A_l} g_i + \frac{1-a}{b}\sum_{x_i \in B_l} g_i\right)^2}{n_l^j(d)} + \frac{\left(\sum_{x_i \in A_r} g_i + \frac{1-a}{b}\sum_{x_i \in B_r} g_i\right)^2}{n_r^j(d)} \right)$$

Де A — набір екземплярів із високими градієнтами, а B — набір екземплярів із низькими градієнтами, g_i – градієнт кожного зразка.

$$\begin{cases} A_l = \{x_i \in A : x_{ij} \le d\} \\ A_r = \{x_i \in A : x_{ij} > d\} \\ B_l = \{x_i \in B : x_{ij} \le d\} \\ B_r = \{x_i \in B : x_{ij} > d\} \\ n_l^j(d) = \sum \mathbb{I}(x_i \in (A_l \cup B_l)) \\ n_r^j(d) = \sum \mathbb{I}(x_i \in (A_r \cup B_r)) \end{cases}$$, поки коефіцієнт $\frac{1-\alpha}{b}$ використовується для нормалізації суми нахилів вище B відносно величини A^c. Це означає, що під час пошуку найкращої точки розщеплення d ознаки j нам просто потрібно знайти d, що максимізує $\tilde{V}_j(d)$.

Таким чином, в GOSS ми використовуємо оцінку $\tilde{V}_j(d)$ над меншою підмножиною екземплярів замість точного $V_j(d)$ для всіх випадків, щоб визначити точку розщеплення, і вартість обчислень може бути значною мірою зменшена.

Наступна теорема вказує, що GOSS не втратить значної точності навчання і буде перевершувати випадкову вибірку.

Теорема. Позначимо похибку апроксимації в GOSS як $E(d) = \left|\tilde{V}_j(d) - V_j(d)\right|$, та $\bar{g}_l^j(d) = \frac{\Sigma_{x_i \in (A \cup A^c)_l} |g_i|}{n_l^j(d)}$, $\bar{g}_r^j(d) = \frac{\Sigma_{x_i \in (A \cup A^c)_r} |g_i|}{n_r^j(d)}$. З імовірністю принаймні $1 - \delta$, ми маємо

$$\mathcal{E}(d) \leq C_{a,b}^{2} ln1/\delta \cdot m \left\{\frac{1}{n_l^j(d)}, \frac{1}{n_r^j(d)}\right\} + 2DC_{a,b}\sqrt{\frac{ln1/\delta}{n}}, (1)$$

де $C_{a,b} = \frac{1-a}{\sqrt{b}} max_{x_i \in A^c}\left|g_i\right|$, та $D = max\left(g^{-j}{}_l(d), g^{-j}{}_r(d)\right)$.

Згідно з теоремою, маємо такі тезиси:

1. Коефіцієнт асимптотичного наближення GOSS є $O\left(\frac{1}{n_l^j(d)} + \frac{1}{n_r^j(d)} + \frac{1}{\sqrt{n}}\right)$.

   Якщо поділ не надто незбалансований (наприклад, $n_l^j(d) \geq O(\sqrt{n})$ та $n_r^j(d) \geq O(\sqrt{n})$), у похибці апроксимації домінуватиме другий член нерівності (1), який зменшується до 0 в $O(\sqrt{n})$ з $n \to \infty$. Це означає, що коли кількість даних велика, наближення є досить точним.
2. Випадкова вибірка є окремим випадком GOSS з $a = 0$. У багатьох випадках GOSS може перевершити випадкову вибірку за умов $C_{0,\beta} > C_{a,\beta-a}$, що еквівалентно $\frac{\alpha_a}{\sqrt{\beta}} > \frac{1-a}{\sqrt{\beta-a}}$ з $\alpha_a = max_{x_i \in A \cup A^c}\left|g_i\right| / max_{x_i \in A^c}\left|g_i\right|$.

Далі ми аналізуємо ефективність узагальнення в GOSS. Розглянемо помилку узагальнення в GOSS $\mathcal{E}_{gen}^{GOSS}(d) = \left|V_j(d) - V_*(d)\right|$, який є розривом між виграшем від дисперсії, розрахованим вибірковими навчальними екземплярами в GOSS, і справжнім виграшем від дисперсії для основного розподілу. Маємо:

$$\mathcal{E}_{gen}^{GOSS}(d) \leq \left|\tilde{V}_j(d) - V_j(d)\right| + \left|V_j(d) - V_*(d)\right| \triangleq E_{GOSS}(d) + E_{gen}(d)$$

Таким чином, якщо апроксимація GOSS є точною, похибка узагальнення з GOSS буде близькою до обчисленої з використанням повних екземплярів даних. З іншого боку, вибірка збільшить різноманітність base learners, що потенційно допоможе покращити ефективність узагальнення. Base learners зазвичай генеруються з навчальних даних за допомогою алгоритму базового навчання,

який може бути деревом рішень, нейронною мережею або іншими видами алгоритмів машинного навчання.

2.2. Ключові метрики продуктивності алгоритму

Матеріал цього підрозділу має реферативний характер. Використовуються теоретичні положення з робіт [8] та [9].

Існує ряд метрик, що застосовується для визначення точності прогнозування.

Коефіцієнт детермінації (Coefficient of Determination) — R^2. Щоб виразити кореляцію між двома випадковими величинами, використовується R^2, що виражається у відсотках. Ця метрика дає швидкість мінливості значень Y, розраховану за X, і навпаки. R^2 визначається наступним чином:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2}{\sum_{i=1}^{n}\left(Y_i - Y^{-}_i\right)^2}$$

де Y_i – спостережувані значення залежної змінної, $\hat{Y}_i$ є оціночними значеннями залежної змінної, Y^{-} – середнє арифметичне спостережуваних значень, та n – кількість спостережень. R^2 досягає значень в інтервалі $[0, 1]$, з оптимальною продуктивністю, коли його значення наближається до одиниці, що вказує на те, що регресійна модель оптимально адаптується до даних.

Середня абсолютна похибка (Mean Absolute Error) — MAE. MAE є мірою, яка кількісно визначає похибку між оціненими та спостережуваними значеннями, що розраховується за наступною формулою:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|f_i - y_i\right| = \frac{1}{n}\sum_{i=1}^{n}\left|e_i\right|$$

де f_i – розрахункові значення і y_i — спостережувані дані. Середнє абсолютного значення різниці між цими величинами визначається як абсолютна похибка їх співвідношення $\left|e_i\right| = \left|f_i - y_i\right|$.

Середня квадратична помилка (Mean Square Error) — MSE. MSE є основним показником порівняння, який обчислює, наскільки добре модель наближається до кількості контрольних прикладів у процесі регресії. Він задається наступною формулою:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{Y}_l - Y_i\right)^2$$

Середня абсолютна відсоткова помилка (Mean Absolute Percentage Error) — MAPE. MAPE забезпечує об'єктивну оцінку помилки оцінки у відсотках від попиту без залежності від порядку величини попиту. Вона задається наступною формулою:

$$MAPE = 100\sum_{t=1}^{T}\frac{\left[\frac{|A_t - F_t|}{A_t}\right]}{T}$$

де A_t — фактичне значення, та F_t — прогнозоване.

Середня абсолютна помилка (Mean Absolute Error) — MAE, що вимірює середню величину помилок у наборі передбачень, не враховуючи їх напрямок. Це середнє за тестовою вибіркою абсолютних відмінностей між прогнозом і фактичним спостереженням, де всі індивідуальні відмінності мають однакову вагу. Вона задається наступною формулою:

$$MAE = \frac{\Sigma_D|y_{pred} - y_{true}|}{\Sigma_D y_{true}}$$

Де D позначає сукупність помилок/продажів за всіма SKU та місцями в певний день. Після обчислення % MAE для кожного дня в тестовому наборі ми дивимося на середній % MAE.

Загальне уявлення, що MAPE і MAE оптимізовують середнє значення помилкове, вони оптимізовують медіану. Це не має значення у випадку симетричного розподілу, але стає великою проблемою, коли 90%+ даних — нулі. MSE правильно обирає середній прогноз, але з іншого боку, воно надає більшу вагу продажам з більшою амплітудою та дуже чутливе до аутлаєрів — точок даних, які помітно відрізняються від інших. Вони представляють помилки

в вимірюванні, неправильний збір даних або просто показують змінні, які не враховуються під час збору даних.

що є гострою проблемою для переривчастого попиту (intermitted demand), бо ці серії як правило дуже волатильні. MAE більш стійкий до викидів. MSE працюють за принципом усереднення похибок, тоді як розрахунок MAE заснований на медіані помилки. Отже, треба акуратно використовувати подібні метрики для оцінки прогнозу для ланцюгів постачання.

У роботі [2] представлене порівняння продуктивності з точки зору вищезгаданих показників різних алгоритмів машинного навчання з алгоритмом LightGBM, ранжоване від найбільш ефективного до найменш ефективного методу на навчальних даних, що створювалися за допомогою нелінійного аналізу історії часу 90 3D R/C будівель з трьома різними розподілами заповнень кладкою, які зазнали 65 землетрусів.

Табл.1. Показники продуктивності порівнюваних алгоритмів для набору даних ROW_FORM_BARE

Regression Metric / Machine Learning Algorithm	R^2	MAE	MSE	RMSE	MAPE	TT (Sec)
Light Gradient Boosting Machine	**0.9076**	**0.1722**	**0.0867**	**0.2902**	**0.1899**	**0.082**
Gradient Boosting Regressor	0.8968	0.1904	0.0968	0.3068	0.2452	0.205
Random Forest Regressor	0.8883	0.1887	0.1035	0.3184	0.1752	0.823
Extra Trees Regressor	0.8840	0.1884	0.1064	0.3237	0.1706	0.657
k-Nearest Neighbors Regressor	0.8343	0.2406	0.1542	0.3875	0.2377	0.065
Linear Regression	0.8312	0.2757	0.1585	0.3939	0.5849	0.019
Bayesian Ridge	0.8312	0.2757	0.1588	0.3941	0.5835	0.018
Ridge Regression	0.8283	0.2768	0.1622	0.3981	0.5804	0.017
Decision Tree Regressor	0.7897	0.2565	0.1941	0.4378	0.2318	0.024
AdaBoost Regressor	0.7721	0.3527	0.2099	0.4578	0.9696	0.143
Elastic Net	0.7650	0.3100	0.2224	0.4675	0.3449	0.020
Lasso Regression	0.7647	0.3100	0.2227	0.4678	0.3484	0.018
Orthogonal Matching Pursuit	0.7550	0.3202	0.2318	0.4776	0.3477	0.018
Huber Regressor	0.7378	0.3438	0.2478	0.4919	0.7261	0.065
Least Angle Regression	0.5082	0.4422	0.4716	0.5778	1.6721	0.021

Табл.2. Показники продуктивності порівнюваних алгоритмів для набору даних ROW_FORM_FULLMASONRY

Regression Metric / Machine Learning Algorithm	R^2	MAE	MSE	RMSE	MAPE	TT (Sec)
Light Gradient Boosting Machine	**0.7833**	**0.1535**	**0.2979**	**0.3861**	**0.9794**	**0.078**
Bayesian Ridge	0.7385	0.2045	0.3217	0.4217	1.5333	0.016
Ridge Regression	0.7367	0.2049	0.3230	0.4228	1.4272	0.017
Linear Regression	0.7366	0.2052	0.3240	0.4230	1.4559	0.017
Least Angle Regression	0.7342	0.2082	0.3249	0.4247	1.5213	0.019
k-Nearest Neighbors Regressor	0.6760	0.1549	0.3447	0.4532	0.2621	0.063
Elastic Net	0.6423	0.2640	0.3752	0.4868	2.3327	0.017
Orthogonal Matching Pursuit	0.6378	0.2504	0.3786	0.4880	1.4332	0.016
Decision Tree Regressor	0.6352	0.2614	0.3809	0.4893	1.4516	0.017
Lasso Regression	0.6300	0.2711	0.3814	0.4941	2.2259	0.018
Huber Regressor	0.5998	0.2856	0.4016	0.5117	2.4230	0.060
Gradient Boosting Regressor	0.4942	0.1759	0.4125	0.5323	0.5681	0.182
Random Forest Regressor	0.3898	0.1538	0.4435	0.5410	0.2347	0.764
Extra Trees Regressor	0.2019	0.1567	0.5402	0.5937	0.2384	0.633
AdaBoost Regressor	0.0737	0.3369	0.6196	0.6696	4.8885	0.117

* TT (Sec)=Training Time in seconds, час тренування алгоритмів у секундах

Табл.1 та 2. Показники продуктивності порівнюваних алгоритмів для зазначених наборів даних

Таблиці 1 та 2 чітко показують перевагу алгоритму LightGBM, який є відмінним за всіма показниками, а похибка продуктивності залишається дуже

низькою порівняно з іншими алгоритмами. Зокрема, точність LightGBM в середньому перевищує другий найкращий метод майже на 3,5%. Ці особливості чітко продемонстровані дуже високими результатами продуктивності, яких він досяг, а також його здатністю узагальнювати нові невідомі ситуації та ефективно моделювати дані реального світу [2].

2.3. Додавання додаткових фіч для моделі

Однією з переваг моделей GBM є те, що вони можуть генерувати показники важливості фіч на основі якості розбиття (або отримання інформації).

Метод оцінки важливості фіч (ознак) — метод, який обчислює оцінку для всіх вхідних ознак для даної моделі, де оцінки просто представляють «важливість» кожної фічі. Вищий показник означає, що конкретна ознака матиме більший вплив на модель, яка використовується для прогнозування певної змінної.

Підключення зовнішніх джерел інформації є важливим кроком в аналізі даних, оскільки це допомагає зрозуміти зв’язок між ознаками та цільовою змінною. Під час навчання моделі, ми можемо використовувати оцінки, обчислені на основі важливості ознак, щоб зменшити розмірність самої моделі. Та імпорт фіч також корисний для інтерпретації отриманих результатів.

Важливість ознаки обчислюється, помічаючи збільшення або зменшення похибки, коли ми переставляємо значення ознаки. Якщо перестановка значень спричиняє значні зміни в помилці, це означає, що ця функція важлива для нашої моделі. Імпортування фіч перестановки заснована на алгоритмі, який працює наступним чином:

1. Обчислити середню квадратичну помилку MSE з вихідними значеннями.
2. Перемішати значення для функцій і зробити прогнози.
3. Обчислити середню квадратичну помилку з перемішаними значеннями.
4. Порівняти різницю між ними.

5. Відсортувати відмінності в порядку спадання, щоб отримати функції з найбільшою чи найменшою важливістю.

Наступний абзац підрозділу має реферативний характер. Використовуються теоретичні положення з робіт [6] та [7].

Значення Gini використовується для розрахунку домішки ноди, де нода — це обчислювальний блок, а домішка ноди є мірою однорідності міток у вузлі, який має одне або більше зважених вхідних зв'язків, передавальну функцію, що описує залежність вхідних від вихідних з'єднань. Імпортування ознак — це в основному зменшення домішки ноди, зважене на кількість зразків, які досягають цього вузла із загальної кількості зразків, тобто ймовірність ноди. Припустимо, що у нас є дерево з двома дочірніми нодами, рівняння, яке ми маємо:

$$ni_j = w_j C_j - w_{left(j)} C_{left(j)} - w_{right(j)} C_{right(j)}$$

де ni_j — значення вузла j, — зважена кількість вибірок, що досягають вузла j, C_j — величина домішки вузла j, $left\ (j)$ — дочірній вузол зліва від вузла j, $right(j)$ — дочірній вузол праворуч від вузла j.

Це рівняння дає нам важливість вузла j, який використовується для обчислення важливості ознак для кожного дерева рішень. Одна функція може використовуватися в різних гілках дерева. Таким чином, ми розраховуємо важливість ознак наступним чином:

$$fi_i = \frac{\sum_{j:node\ j\ splits\ on\ feature\ i} ni_j}{\sum_{j \in all\ nodes} ni_j}$$

Об'єкти нормалізуються щодо суми всіх значень ознак, присутніх у дереві, і після поділу їх на загальну кількість дерев у нашому випадковому лісі ми отримуємо загальну важливість ознак. Завдяки цьому ви можете краще зрозуміти важливість особливостей у випадкових лісах.

3. Програмна реалізація

3.1. Огляд даних

Розглянемо датасет з тренувальними даними про щоденні продажі та ціни електронної комерції по Україні, які доступні з початку 2020 року і до середини липня 2021 року. А також маємо набір тестувальних даних із продажами продуктів за останній тиждень. У даних представлений продаж п'ятьох основних груп товарів: сири, тропічні фрукти, мінеральна вода, йогурти та хлібобулочні вироби.

Ми маємо спрогнозувати продажі (та відсутність продажів) продуктів на наступні 2 тижні, маючи для тестування і коригування моделі дані про продажі на перший тиждень.

Завантажуємо та аналізуємо дані: маємо дерево: в основі лежать 5 commodity_group, які діляться на 208 productCategoryId, які діляться на 182 productTypeId, і в самому низу — на 1961 товарів (SKU).

Опис тренувальних даних Train.zip:

ID	Унікальний ідентифікатор рядка
geoCluster	ID місця доставки замовлення
SKU	ID продукту
date	Дата замовлення
price	Ціна SKU, що було замовлене для доставки
sales	Кількість SKU, що було замовлене для доставки

Опис даних у файлі з інформацією про локацію geo_params.csv:

geoCluster	ID місця доставки замовлення
CityID	Місто, у яке було замовлено доставку

Опис даних у файлі з інформацією про продукт sku.csv:

SKU	ID продукту
lagerUnitTypeId	Тип артикула виміру кількості на упаковку

lagerUnitType_caption	Тип артикула виміру кількості на опис упаковки (кг, мл, г)
lagerUnitQuantity	Кількість SKU в упаковці
trademark	Ідентифікатор торгової марки виробника SKU
countryOfOrigin	Ідентифікатор країни виробника
countryOfOrigin_caption	Скорочена назва країни виробника
productTypeId	Категорія SKU, яка включає всі близькі замінники ідентифікатора
productType_caption	Категорія SKU, яка включає всі близькі замінники назви
brandId	Ідентифікатор бренду виробника
commodity_group	Товарна група
commodity_group_caption	Скорочена назва товарної групи
productCategory	Категорія SKU, вищий рівень агрегації
productCategory_caption	Абревіатура категорії SKU

Опис тестувальних даних Test.zip:

файл test.csv

ID	Унікальний ідентифікатор рядка
geoCluster	ID місця доставки замовлення
SKU	ID продукту
date	Дата замовлення
price	Ціна SKU, що було замовлене для доставки
sales	Кількість SKU, що було замовлене для доставки

файл Sample_submission.csv

ID	Унікальний ідентифікатор рядка
sales	Кількість SKU, що було замовлене для доставки

Ми надаємо прогноз на 1 666 028 варіацій SKU+геокод+дата. По кожному прогнозу підсумовується помилка, ділиться на суму реальних продажів, обчислюється окремо для кожного дня, а потім усереднюється за 2 тижні.

3.2. Оціночна метрика

Орієнтуємося на цільову метрику MAE — середня абсолютна помилка, що вимірює середню величину помилок у наборі передбачень, не враховуючи їх напрямок. Детальний опис метрики у розділі 2.2. Ключові метрики продуктивності алгоритму.

Публічний та приватний розділ (Public and Private)

Дані тесту далі випадковим чином поділяються на публічні (перші 7 днів, 20 липня – 26 липня) та приватні (останні 7 днів, 27 липня – 2 серпня).

3.3. Аналіз даних

	geoCluster	SKU	date	price	price_change	sales
195875	2017	17	2021-07-21	19.99	0.0	NaN
776122	2158	419952	2021-07-24	374.99	0.0	NaN
831838	2183	540390	2021-07-20	41.29	NaN	NaN
1081854	2258	607635	2021-07-24	16.39	0.0	3.0
1615220	2984	819150	2021-08-01	51.19	0.0	2.0
491855	2061	837329	2021-07-27	25.59	0.0	NaN
643144	2117	868167	2021-08-01	24.09	0.0	NaN
1287839	2406	810184	2021-07-27	71.09	0.0	NaN
58052	1935	838007	2021-07-28	248.49	0.0	NaN
1535683	2807	559301	2021-07-29	326.59	0.0	NaN

рис. 3.1. Огляд даних

Якщо на місці price та sales пропущені значення, то це означає, що продажів в цей день не було, тому і ціну до таблиці не заносили, тож варто ці значення для продажів(sales) заповнити нулями, а для ціни (prices) заповнити останніми відомими значеннями ціни в таблиці для відповідного geoCluster та SKU.

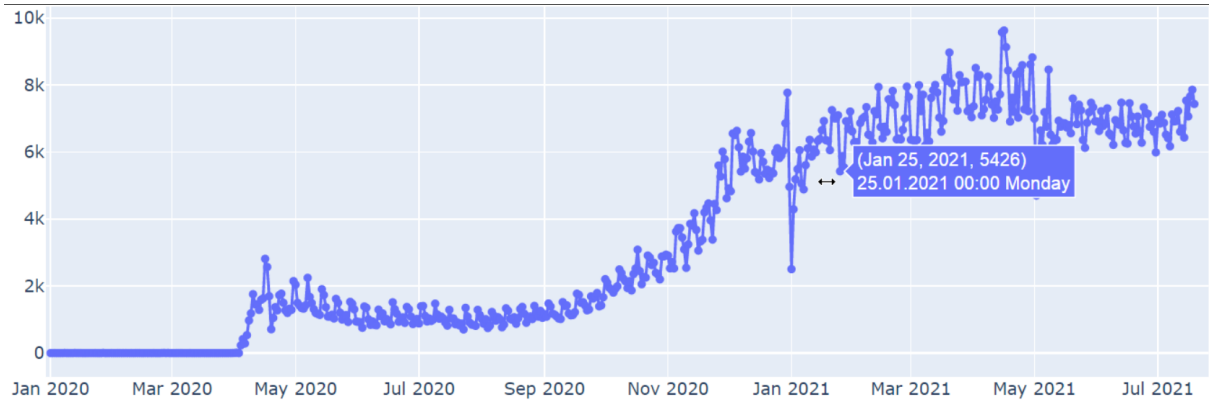

рис. 3.2. Загальний обсяг продажу товарів за день для всього діапазону даних

На рис.3.2. показано загальну кількість добових одиниць (daily units — загальний обсяг продажу товарів за день для всього діапазону даних. Можемо помітити, що щорічно в дні свят, промоакцій та чорної п'ятниці спостерігаються стрибки продажів, що відповідають великим розпродажам. Таким чином, ми можемо створити змінні індикатора для певних свят, щоб врахувати ці події в нашому моделюванні.

Ми бачимо, що до квітня 2020 року даних немає, що говорить про відсутність продажів, певно склади для рітейлу тільки почали відкривати. Починаючи з 2020-04-07 продажі перевищили значення 500 штук на день і вже не падали нижче зазначеної кількості. Тож усі дані до цього також можемо відкинути, оскільки вони не несуть ніякої цінності.

Можемо відмітити більшу волатильність (фінансовий показник, що характеризує мінливість ціни) продажів під час пандемії Covid. Зокрема, спостерігається стрибок на початку Covid, потенційно через збільшення попиту на певні вітаміни у фруктах, різке падіння й швидке відновлення у другій половині 2020 року, та відносно стабільні продажі протягом 2021 року.

Виходячи зі статистики, яка була подана у файлі продажів, були SKU, які надійшли в продаж пізніше ніж зазначений період даних, або ж були заміни SKU. Тому при прогнозуванні ми локалізуємо свою увагу на період активних продажів.

Якщо період даних великий, то маємо виділяти сезонність, у нашому ж випадку ми це опускаємо через незначну кількість даних.

Можемо помітити, що дні тижня та місяці впливають на інтенсивність продажу різних SKU. Також у продажів є кореляція, що пов'язана із запитами в Google, як-от: "доставка Сільпо". Припускаємо можливість кореляції з геокластерами з найбільшою кількістю продажів. Можемо допустити, що це Київ, тому додаткові датасети та фічі брати по Києву.

3.4. Валідація

Валідацію, тобто процес перевірки правильності типу даних, яких ми очікували, робимо time based k-fold стратегією, в якій в трейн потрапляє 1 місяць, а в валідацію 2 тижні (окремо рахуємо метрику на 1-й і 2-й тиждень, щоб змоделювати public/private розбиття на lb):

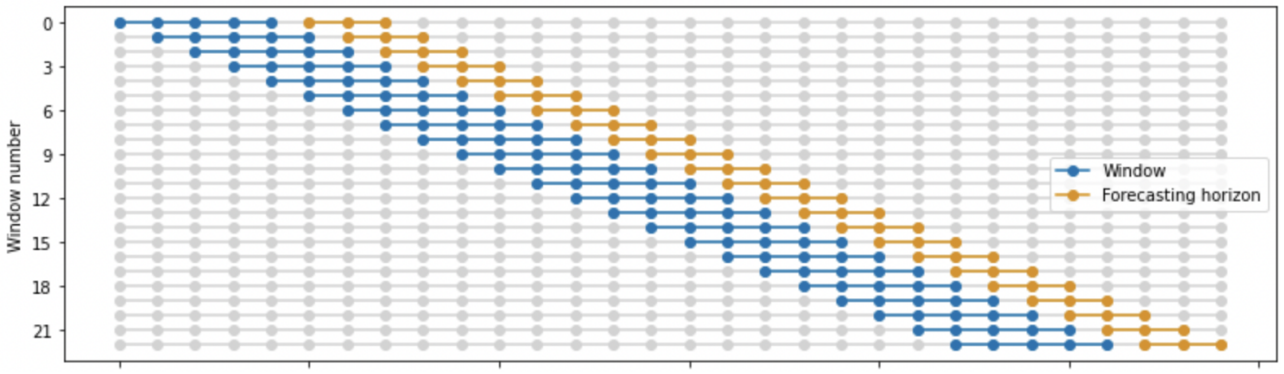

рис.3.2. Валідація даних

3.5. Тривіальне рішення (базовий підхід)

Для кожної дати і (geoCluster, SKU) пари робимо прогноз на всі 14 днів медіанним значенням продажів SKU в geoCluster за останній місяць.

MAE (1st week)	MAE (2nd week)
0.93600	0.93877

Таким чином ми встановили базовий MAE для нашого набору даних за допомогою тривіальної прогнозної моделі, а саме прогнозування середнього цільового значення з навчального набору даних. Модель, яка буде досягати нижчого значення MAE, ніж MAE для тривіальної моделі, є найбільш ефективною.

3.6. Model-based базове рішення

Ознаки:

- geo_cluster_id
- sku_id
- median_sales — (медіанне значення кількості продажів sku_id в geo_cluster_id за попередній місяць)
- median_non_zero_sales — (медіанне значення кількості продажів sku_id в geo_cluster_id за попередній місяць без урахування днів, коли продажів не відбувалось)

Модель: градієнтний бустинг (lightgbm) з MAE loss і early stopping за цільовою метрикою MAE.

3.7. Додаткові ознаки(фічі)

1. Датасет із вихідними/робочими днями, святами (коли були перенесення робочих днів), зібраний самостійно в Excel
2. Дані за погодою у Києві
3. Дані щодо забруднення повітря в Києві за 2020 — 2021
4. Дані за періодом локдауну
5. Тренди з доставки (по тижнях)
6. Дати трансляції Чемпіонату Європи з футболу

Також додали інші фічі, такі як: city_id, category_id, product_type_id, brand_id, trademark_id, origin_country_id, commodity_group_id, згруповані по city_id/ category_id/ product_type_id/ brand_id/ trademark_id/ origin_country_id/ commodity_group_id медіани продажів, weekday, week_no (week of year), month та ембеддінг вектори текстових ознак товарів.

	w2v_emb_1	w2v_emb_2	w2v_emb_3	w2v_emb_4	w2v_emb_5	w2v_emb_6	w2v_emb_7	w2v_emb_8
sku_id								
17	-0.613400	0.368053	-0.634122	0.005915	1.693699	0.005733	0.982584	-0.281478
18	-0.586724	0.400572	-0.590650	-0.046997	1.767318	-0.038265	0.922868	-0.100942
24	-0.629215	0.420973	-0.613216	0.025009	1.732937	-0.019977	0.995371	-0.262781
25	-0.586287	0.306531	-0.660516	0.026975	1.584505	-0.098628	0.995551	-0.196404
208	0.665920	0.657977	-0.096269	-1.337203	2.158585	-0.638259	0.331346	-1.494282
...	...	...	...	...	...	...	...	...
873803	-0.342109	0.669412	0.246947	-0.243078	1.171487	-0.181959	0.598105	-0.322623
873804	-0.317623	0.764013	0.438810	-0.085553	1.303396	-0.221918	0.642721	-0.156087
873805	-0.317623	0.764013	0.438810	-0.085553	1.303396	-0.221918	0.642721	-0.156087
874108	-0.317623	0.764013	0.438810	-0.085553	1.303396	-0.221918	0.642721	-0.156087
874109	-0.317623	0.764013	0.438810	-0.085553	1.303396	-0.221918	0.642721	-0.156087

рис.3.7.1 Word to vec ембеддінг

p.s. маппінг назв використаних колонок на первинні дивіться у файлі *col_mapping.json* (генерується у ноутбуці FN-eda-data-cleaning-and-no-ml-baseline-v2.ipynb)

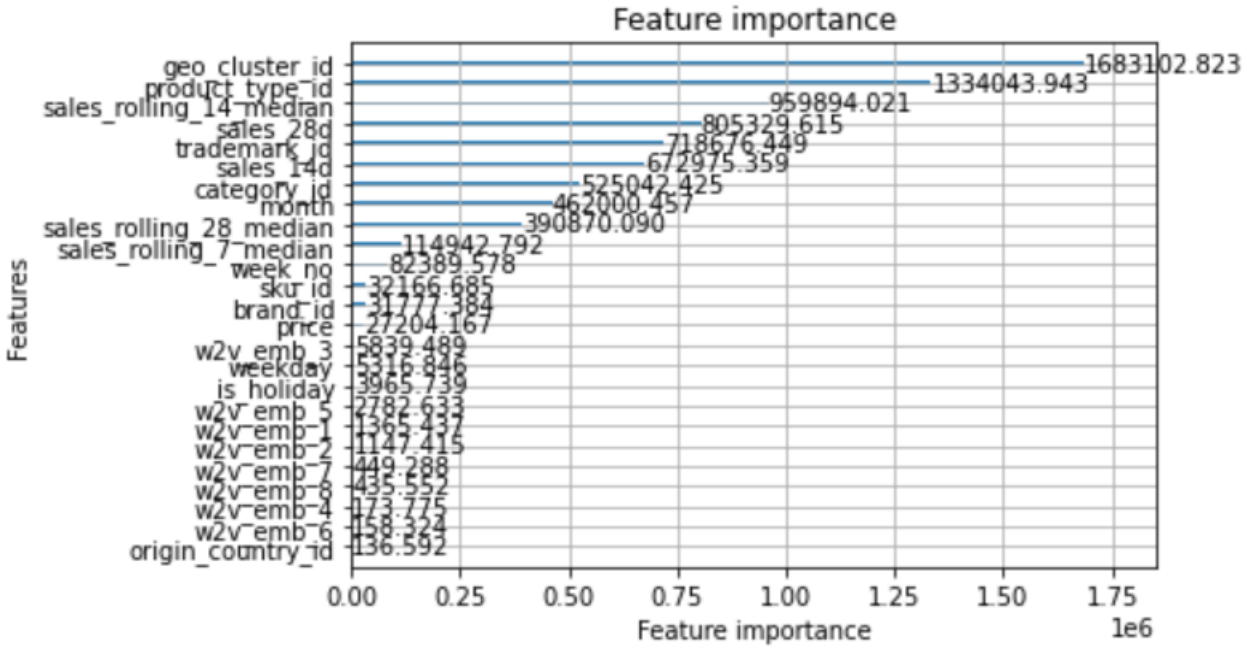

рис. 3.5. Метод оцінки важливості фіч(ознак)

3.8. Meta Feature

Оскільки в тестовому датасеті ціни заповнені всюди (а продажі, очевидно, не всюди були), то природним чином можна вирішити, що ціни в тесті дозаповнили штучним чином й це нам може допомогти. Спробуємо досягти того ж ефекту на трейні і валідації.

Очевидні варіанти як можна “дозаповнити ціну” штучно:

1. для днів, в яких продажів товару sku_id в geo_cluster_id не було, дозаповнимо їх такою ж, як в день попереднього продажу в тесті, якщо ж попереднього продажу не було, то заповнимо ціною наступного продажу (ffill, bfill)
2. для днів, в яких продажів товару sku_id в geo_cluster_id не було, дозаповнимо їх такою ж як в день попереднього продажу в тесті, якщо ж попереднього продажу не було, то заповнимо ціною останнього продажу з трейну
3. для днів, в яких продажів товару sku_id в geo_cluster_id не було, дозаповнимо їх такою ж як в день попереднього продажу в тесті, якщо ж попереднього продажу не було, то заповнимо її деякою середньою/медіаною для sku_id серед інших geo_cluster_id в той же день

Оскільки в тренувальних даних ціни на більшість sku_id в рамках кожного geo_cluster_id досить часто змінюються, то за рахунок штучного характеру дозаповнення пропущених цін можна згенерувати корисний індикатор (price_change) — “чи змінилася ціна в цей день для sku_id в geo_cluster_id порівняно з минулим днем?”, який ідейно інтерпретується так:

- ціна відносно попереднього дня змінилась, тобто в цей день точно (100%) були продажі більше 0
- ціна відносно попереднього дня не змінилась, тобто ймовірно продажів не було (але це не точно), оскільки скоріш за все це просто дублювання ціни з моменту останньої реальної продажі

Порахувавши статистики для price_change по різних періодах трейну і тесту, а також метрики на валідації і public leaderboard, доходимо до висновку, що скоріш за все було обрано схему подібну до варіанта 3 для заповнення пропущених цін в тесті.

Корисність (за метрикою MAE) price_change фічі пояснюється не величиною зміни ціни, а саме природою цієї зміни (ми знаємо ціну, тільки якщо були продажі) і по суті для прогнозу кількості продажів в певний день ми використовуємо інформацію про те чи були вони взагалі.

3.9. Фінальне рішення

Фінальним рішенням є LightGBM з MAE loss і early stopping за цільовою метрикою MAE, що використовує всі вищенаведені фічі з підрозділу 3.7. та 3.8. LightGBM:

MAE (1-ий тиждень)	MAE (2-ий тиждень)
0.7384	0.7938

Порівнявши з тривіальним рішенням, а саме моделлю прогнозу на всі 14 днів медіанним значенням продажів SKU в geoCluster за останній місяць для кожної дати і (geoCluster, SKU) пари, отримаємо слабші результати, а саме:

MAE (1st week)	MAE (2nd week)
0.93600	0.93877

Тож можемо зробити висновок про те, що дана модифікація моделі градієнтного бустингу LightGBM досягає MAE краще, тобто нижча середня абсолютна помилка, ніж MAE для тривіальної моделі, й дозволяє ефективно вирішувати проблему прогнозування продажів для електронної комерції з середньою абсолютною помилкою 0.7384 для першого тижня та 0.7938 для другого в горизонті прознозу.

Висновки

У роботі описано основну задачу електронної комерції, що потребує модель рекомендаційної системи та основні проблеми, що виникають під час прогнозування продажів. Розглянуто основні підходи до прогнозування продажів та методи вирішення проблем, що виникають під час прогнозування, а саме такі методи, як дерева рішень, діф таргет та ембеддінг. Надано визначення методу градієнтного бустингу LightGBM, ключові метрики продуктивності алгоритму та порівняння продуктивності з іншими алгоритмами. Наведено приклад застосування даного методу на даних продажів та цін електронної комерції по Україні, які доступні з початку 2020 року і до середини липня 2021 року, наведено прогноз на наступні 2 тижні та проведено аналіз ефективності даного алгоритму за допомогою ключової метрики продуктивності MAE. Проведено порівняння модифікованої моделі градієнтного бустингу LightGBM з тривіальною моделлю, а саме медіанним значенням кількості продажів товарів за попередній місяць, та зроблено висновок про значно більшу ефективність модифікованої моделі LightGBM у вирішенні проблеми прогнозування продажів для електронної комерції з нижчою середньою абсолютною помилкою MAE у горизонті прознозу.

Використана література

1. Global Ecommerce Forecast 2022, report by Ethan Cramer — Flood, Feb 2, 2022. https://www.emarketer.com/content/global — ecommerce — forecast — 2022
2. Konstantinos Demertzis, Konstantinos Kostinakis, Konstantinos Morfidis, Lazaros Iliadis (2021). "A comparative evaluation of machine learning algorithms for the prediction of R/C buildings' seismic damage".
3. LightGBM documentation https://lightgbm.readthedocs.io/en/latest/Features.html
4. Ke, Guolin; Meng, Qi; Finley, Thomas; Wang, Taifeng; Chen, Wei; Ma, Weidong; Ye, Qiwei; Liu, Tie-Yan (2017). "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". Advances in Neural Information Processing Systems.
5. Understanding customers better through neural network embeddings - Adam Hornsby, 2018 https://anhornsby.github.io/embeddings-retail/#/
6. Terence Shin (2021). Understanding Feature Importance and How to Implement it in Python https://towardsdatascience.com/understanding-feature-importance-and-how-to-implement-it-in-python-ff0287b20285#:~:text=Feature%20Importance%20refers%20to%20techniques,to%20predict%20a%20certain%20variable
7. Stacey Ronaghan (2018). The Mathematics of Decision Trees, Random Forest and Feature Importance in Scikit-learn and Spark https://towardsdatascience.com/the-mathematics-of-decision-trees-random-forest-and-feature-importance-in-scikit-learn-and-spark-f2861df67e3 https://anhornsby.github.io/embeddings-retail/#/7
8. Daniel Waller. Methods for Intermittent Demand Forecasting https://www.lancaster.ac.uk/pg/waller/pdfs/Intermittent_Demand_Forecasting.pdf

9. "2.5 Evaluating forecast accuracy | OTexts". www.otexts.org. Retrieved 2016-05-18.
10. 1.10. Decision Trees. Scikit-learn documentation https://scikit-learn.org/stable/modules/tree.html
11. RC Steorts (2017). Tree Based Methods: Regression Trees https://www.math.mcgill.ca/yyang/resources/doc/tree.pdf
12. Echo Yang, David Chen (11 Feb 2020). Tree-Math, LightGBM. https://github.com/YC-Coder-Chen/Tree-Math/blob/master/LightGBM.md
13. Leo Breiman (1996). "BIAS, VARIANCE, AND ARCING CLASSIFIERS"