Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра математики

Курсова робота

освітній ступінь – магістр

на тему: **«Моделювання прогнозу попиту з використання рекомендаційної системи на базі методу LightGBM»**

Виконав: студент 1-го року навчання,
Спеціальності
113 Прикладна математика
Кольчик Микита Олегович
Керівник Дрінь Світлана
Сергіївна,
кандидат фіз.-мат. наук
«____»_____ 20___ р.

Київ – 2023

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра математики

Освітній ступінь магістр

Спеціальність 113 Прикладна математика

Освітня програма магістр

ЗАТВЕРДЖУЮ

Завідувач кафедри математики

Олійник Б.В.

«____»_____ 20___ року

ЗАВДАННЯ

ДЛЯ КУРСОВОЇ РОБОТИ СТУДЕНТУ

Студента Кольчика Микити Олеговича

1. Тема роботи «**Моделювання прогнозу попиту з використання рекомендаційної системи на базі методу LightGBM**», керівник роботи Дрінь Світлана Сергіївна, кандидат фіз.-мат. наук.
2. Строк подання студентом роботи: 17 травня 2023
3. План роботи

   Вступ

   Розділ 1. Алгоритм LightGBM

   Розділ 2. Моделювання попиту

   Розділ 3. Прогнозування попиту в датасеті з продажу напоїв

   Висновки

   Список використаних джерел

Зміст

Вступ

Сучасна економіка кожного підприємства є дуже складною. В кожного бізнесу є багато постачальників, багато точок продажів, і багато брендів. Для того, щоб не заплутатися в такому різноманітті, необхідні системи обліку, за допомогою яких можна контролювати прибутковість того чи іншого елементу. Також, існує велика кількість додаткових змінних, які впливають на рентабельність бізнесу: кількість складів, кількість запасів на складах, швидкість доставки, вартість доставки тощо.

Для того, щоб залишатися бізнесу прибутковим, необхідно завжди оптимізовувати управління запасами таким чином, щоб кількість запасів відповідала потребам покупців і, в той же час, не витрачати багато грошей на утримання або доставку товарів. Останнім часом, для таких задач оптимізації використовуються алгоритми машинного навчання. За допомогою таких алгоритмів можна прогнозувати майбутній попит на товари для того, щоб контролювати кількість запасів.

Отже, **метою** нашої роботи буде спрогнозувати попит на товари за допомогою алгоритму LightGBM і створити алгоритм, за допомогою якого буде можливо змоделювати прогнози на роботу магазинів.

Робота складається з трьох розділів. Перший розділ присвячено огляду алгоритму LightGBM, його структурі, порівнянні з іншими алгоритмами, перевагам і недолікам. У другому розділі розглядається теорія оптимізації запасів і метрикам, за допомогою яких можна контролювати прибутковість, маючи навіть обмежені дані щодо продажів. Третій розділ присвячено розробці програмного коду, який прогнозуватиме дані продажів і рахуватиме метрики запасів для ланцюга «Магазин-Товар-Тип пакування товару».

1. Алгоритм LightGBM

1.1 Ансамблеві методи

Ансамблеві методи машинного навчання широко використовуються у машинному навчанні і мають більшу точність, ніж звичайні методи, оскільки вони тренуються не на одній моделі, а на багатьох, таких як лінійна регресія або дерева рішень. Серед ансамблевих методів виділяють три:

- Stacking – навчаються декілька слабких моделей, які подаються на вхід до мета- моделі.
- Bagging – базові моделі навчаються паралельно, а результатом буде їх усереднення.
- Boosting – базові моделі навчаються послідовно, виправляючи помилки попередніх моделей.

Бустінг може відбуватися двома методами: адаптивним і градієнтним. Алгоритми градієнтного бустінгу нас цікавлять найбільше, бо на одному із них будуватиметься LightGBM. Ідея градієнтного бустінгу полягає в тому, що кожна наступна модель намагається скоригувати помилки попередньої моделі. Тобто наступна модель навчається на помилках попередньої.

Алгоритмом, на базі якого побудований LightGBM є GBDT (Gradient-Boosted Decision Trees).

Класичний алгоритм градієнтного бустінгу виглядає таким чином [14].

1. На вхід подаються дані $\{(x_i, y_i)\}_{i=1}^{n}$ і диференційована функція втрат $L(y_i, F(x))$;

2. Ініціалізується модель прогнозів константою

$$F_0(x) = argmin_\gamma \sum_{i=1}^{n} L(y_i, \gamma)$$

Ця константа на першому кроці є середнім значенням y_i.

Далі, запускається цикл із M дерев, які вчитимуться один за одним на помилках попереднього дерева, де $m \epsilon [1, M]$.

Рахуються залишки, які називаються псевдо-залишками:

$$r_{im} = -\left[\frac{\partial L\left(L(y_i, F(x_i))\right)}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{для } i = 1, \dots n$$

Будується дерево рішень для r_{im} з листами R_{jm}

Рахуються вихідні значення для кожного листка, або гамма-значення

$$\gamma_{jm} = argmin_{\gamma} \sum_{x_{i \epsilon R_{ij}}} L(y_i, F_{m-1}(x_i) + \gamma)$$

На останньому кроці циклу оновлюється початкова модель передбачень $F_m(x)$:

$$F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \epsilon R_{jm})$$

Фінальним результатом буде прогноз $F_m(x)$ після m циклів

Зазвичай, кількість дерев для ітерації обирають $M = 100$, але оптимальна кількість може бути визначена функціями перебору, наприклад GridSearch, бо кількість дерев є гіперпараметром.

1.2 LightGBM – алгоритм для задач регресії

LightGBM- відкрита бібліотека градієнтного бустінгу, створена у 2017 році, яка є покращеною версією алгоритму GBDT яка використовується для завдань регресії і класифікації. Автори, дослідники з Microsoft, доводять, що цей алгоритм має кращу оптимізацію у порівнянні з подібними алгоритмами градієнтного бустінгу, адже вони аналізують всі дані для того, щоб оцінити

приріст інформації, в той час як у LightGBM пропонуються новітні методи GOSS (Gradient-based One-Side Sampling) і EFB (Exclusive Feature Bundling) для пришвидшення оцінки приросту інформації.

Gradient-based One-Side Sampling – алгоритм для зменшення кількості спостережень у вибірці. Його ідея полягає в тому, що ми беремо всі спостереження з великою помилкою і деяку кількість спостережень із маленькою помилкою. Це означає, що кількість спостережень, які не вносять великий вклад у градієнти, може бути зменшена, одночасно зберігаючи важливу інформацію. По-друге, це означає, що швидкість роботи алгоритму значно збільшується. Оскільки такий розподіл спостережень буде некоректним, використовується коефіцієнт *K*, який компенсуватиме цю неточність. [5]

$$K = \frac{(1 - A)}{B}, \text{де}$$

$$A - \text{частина всіх елементів з великою помилкою,}$$

$$B - \text{частина елементів з маленькою помилкою, які беруть участь у розрахунках}$$

В деревах рішень є декілька методів, щоб знайти розбиття. Одне із них – за допомогою ентропії і приросту дисперсії. Ентропія – міра невизначеності, яка, іншими словами, визначає непередбачуваність системи. Ентропія рахується за наступною формулою:

$$E(d) = -\sum_{i=1}^{n} P_i * log_2 P_i$$

Приріст дисперсії (Variance gain) – обернене значення для ентропії, воно позначає чистоту вузла. Тобто:

$$V(d) = 1 - E(d)$$

В той же час, приріст дисперсії в алгоритмі LightGBM знаходиться за формулою:

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

де $A_l = \{x_i \in A: x_{ij} \leq d\}, A_r = \{x_i \in A: x_{ij} > d\}, B_l = \{x_i \in B: x_{ij} \leq d\},$

$B_r = \{x_i \in B: x_{ij} \geq d\}, g_i -$ градієнти функції втрат.

Тобто, знаходиться приріст дисперсії $\tilde{V}$, для параметра j датасету у вузлі d. Сума в дужках складається з двох доданків. Перший стосується елементів $x_{ij} \leq d$, другий- $x_{ij} \geq d$. Для балансу вибірок A і B, сума градієнтів елементів з множини B, використовується множник $\frac{1-a}{b}$. Таким чином, замість того, щоб рахувати приріст дисперсії для всіх елементів певного параметру, приріст дисперсії рахується на меншій кількості елементів, що значно зменшує час її обчислення.

Exclusive Feature Bundling – алгоритм для зменшення розмірності датасету. У великому датасеті з великою кількістю змінних багато ознак можуть бути взаємовиключними, тому такі ознаки можуть бути об'єднані без втрати інформації в одну функцію, яка називається «ексклюзивний набір функцій». [5]

Це означає, що розмірність датасету зменшиться з O(#data × #feature) до O(#data × #bundle) [цитата].

1.3 Порівняння LightGBM з іншими алгоритмами

1.3.1 LightGBM vs XGBoost

Обидва алгоритми є асиметричними деревами, але у їх побудові є деякі відмінності. По-перше, LightGBM здійснює листове зростання дерева (leaf-wise tree growth), в той час як XGBoost – зростання по рівнях (level-wise tree growth). Різниця між ними в тому, що листове зростання ділить дерево на основі внеску до глобальної функції втрат, а не функції втрат вздовж певної гілки.

По-друге, є відмінність у роботі з категоріальними змінними. XGBoost за замовчанням обробляє їх як числові, хоча це не зовсім коректним, тому перед роботою з алгоритмом необхідно провести з такими даними деякі маніпуляції, наприклад перетворити їх у dummy variables. В той же час, LightGBM має влаштовані алгоритми кодування категоріальних ознак, які потім об'єднує в групи.

1.3.2 LightGBM vs CatBoost

CatBoost створює збалансовані дерева на кожному кроці, використовуючи так звані «oblivious trees», що призводить до того, що на кожному розділенні дерева розділяються за однаковими умовами.

Також у CatBoost використовується Впорядкований Бустінг. Це алгоритм, який перетворює категоріальні ознаки за певним принципом, не схожим на інші. Він враховує порядок, за яким надходять категоріальні ознаки і будує розподіл імовірностей для кожної категоріальної ознаки.

1.4 LightGBM для часових рядів

Зазвичай, для прогнозування часових рядів використовується ряд класичних алгоритмів, які в якості вхідних даних беруть деяке вікно у наявному часовому ряді. Наприклад, у ряді довжиною $N=[a_1,a_2...]$, береться вікно довжиною n, яке подається на вхід алгоритму, а наступне значення ряду виступатиме у ролі залежної змінної. Алгоритмами, які використовують такий метод є ARIMA, Rolling Means, Recurrent Neural Networks. Але, якщо в наших даних ми маємо не тільки часовий ряд, але й деякі інші параметри, як категоріальні або числові змінні, можна використати інший метод. Наряді з іншими ознаками, ми можемо розкласти кожну часову відмітку (timestamp) на декілька інших параметрів, таких як рік, місяць, день, година тощо. Потім, на

цих даних тренується модель машинного навчання, яка вивчає не тільки числові і категоріальні ознаки, але й часові параметри, які сприймає також як числові. Це означає, що ми перетворюємо задачу прогнозування часових рядів у задачу навчання з вчителем (supervised learning).

1.5 Метрика RMSE для порівняння роботи алгоритмів.

Для порівняння алгоритмів, що виконують задачу регресії, використовується багато метрик, серед яких є MAE (Mean Absolute Error), MAPE (Mean Absolute Percentage Error) і найпопулярніша – RMSE (Root Mean Squared Error). Сенс кожної з них полягає в тому, що усереднюється сума різниць значень прогнозу і значень тестової вибірки. Зокрема, формула для знаходження RMSE виглядає таким чином:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$

За допомогою цієї метрики ми будемо порівнювати якість прогнозу алгоритму LightGBM з іншими алгоритмами.

1.6 Висновки до розділу 1

LightGBM є відносно новим алгоритмом градієнтного бустінгу для машинного навчання, який можна використовувати не тільки у задачах класифікації, але й також у прогнозуванні часових рядів, якщо у вхідних даних наявні інші змінні. Також, LightGBM є швидким і точним, адже у ньому є такі функції, які зменшують розмірність у вибірках для того, щоб прискорити його роботу і зменшити кількість обчислень, при цьому не погіршуючи якість вхідних даних. Таким чином, LightGBM буде використовуватись у задачі з прогнозування попиту і моделювання запасів.

2. Моделювання попиту

2.1 Інвентаризація запасів

В сучасному світі, коли у бізнесу багато постачальників, розгалужена мережа складів, товарів і магазинів, необхідно проводити оптимізацію запасів для того, щоб тримати баланс між ціною витрат на утримання товару і ціною нестачі товарів у магазинах. За допомогою оптимізації запасів, підприємства можуть зменшити витрати, які пов'язані з надлишковими запасами і, з іншого боку, можуть гарантовано задовольняти попит клієнтів для максимізації свого прибутку.

2.2 Страхові Запаси

Страхові запаси – це інвентар, який необхідний для того, щоб запобігти проблемі вичерпання товару. Обрахунок кількості страхових запасів є важливим для будь-якого бізнесу, бо це, насамперед, є гарантією того, що власник бізнесу не отримає збитків у період високого попиту.

Попередньо, важливо ввести ще декілька понять для розуміння підходів, які використовуються для обрахунку страхових запасів.

- Рівень сервісу (Service Level) – імовірність того, що запаси, які ми отримали на початку циклу, будуть більші або дорівнювати попиту протягом наступного циклу. Позначається символом α. Зазвичай:

$$\alpha = 0.95$$

- Стохастичний попит (Stochastic Demand) – У реальному світі неможливо передбачити реальний попит на товар, тому у роботі з даними, ми відштовхуємося від того, що попит стохастичний, але належить деякому розподілу.
- Розподіл попиту (Demand Distribution) – оскільки ми працюємо із стохастичним попитом, ми повинні розуміти який розподіл попит має. Неправильне розуміння розподілу може призвести до неправильних

розрахунків і, врешті, до втрати прибутку. Зазвичай для опису попиту використовуються два розподіли- нормальний і гамма розподіли.

- Функція щільності імовірності (Probability Density Function) – інтуїтивно її можна уявити, що чим вища щільність навколо деякого значення x, тим вища імовірність того, що випадковий розподіл буде близьким до цього значення. Для нормального розподілу $N(0,1)$ функція щільності задається формулою

$$f_N(x, 0,1) = \frac{1}{\sqrt{2\pi}} * e^{-\frac{x^2}{2}}$$

- Функція розподілу (Cumulative Distribution Function) – показує, наскільки імовірно, що випадкове значення x цього розподілу є нижчим за деяке порогове значення z.

$$F_X(z) = \int_{-\infty}^{z} f_X(x)dx = \alpha$$

- Функція квантилів (Inverse Cumulative Distribution Function) – за допомогою цієї функції ми можемо отримати поріг z, який досягне імовірності α того, що розподіл X буде нижче цього порогу.

$$F^{-1}(\alpha) = z \leftrightarrow F_X(z) = \alpha$$

2.3 Нормальний розподіл попиту

Функція розподілу є корисною моделювання попиту, бо вона фактично означає імовірність, що попит буде нижче деякого порогу. Тоді ми можемо порахувати необхідний запас для деякого рівня обслуговування. Отримуємо:

$$\iota = F_N^{-1}(\alpha, \mu, \sigma)$$

Тоді страхові запаси для нормально розподіленого попиту виглядають так [6]:

$S_s = z_\alpha * \sigma * \sqrt{\tau}$, де

$z_\alpha -$ коефіцієнт рівня обслуговування,

$\sigma -$ стандартне відхилення попиту,

$$\tau - \text{час доставки}$$

Для випадків, коли поповнення товару відбувається негайно, тобто $\tau = 1$, страхові запаси обчислюються таким чином [6]:

$$S_s = z_\alpha * \sigma$$

2.4 Гамма розподіл попиту

Нажаль, на практиці нормальні розподіли трапляються дуже нечасто, через те, що дані не розподілені рівномірно і середнє не збігається з медіаною і модою. Побудувавши гістограму попиту, можна зрозуміти, з яким розподілом ми працюємо, хоча є й інші методи, але про них пізніше.

Один із найчастіших розподілів у попиті – гамма розподіл. Він характеризується двома параметрами: параметр форми k і параметр масштабу θ. Причому обидва параметри залежать від середнього і стандартного відхилення.

$$k = \frac{\mu^2}{\sigma^2} \qquad \theta = \frac{\sigma^2}{\mu}$$

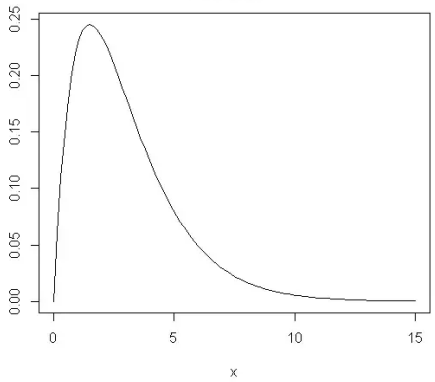

Рисунок 1- Гамма розподіл

Для того, щоб дізнатися наші основні метрики, ми можемо скористатися схожими формулами, але будуть деякі відмінності, адже ми працюємо з нерівномірним розподілом.

Запаси рахуються схожим чином [6]:

$$\iota = F_\Gamma^{-1}(\alpha, k, \theta), \text{де}$$

$$F_\Gamma^{-1} - \text{функція квантилів гамма розподілу}$$

Тоді страхові запаси обчислюються так:

$$S_s = \iota - \mu_x$$

2.5 Зміщення попиту

У гамма розподілі імовірність отримати значення близькі до нуля є низькі, тому для того, щоб покращити розподіл і наблизити його якомога краще, варто змістити його на мінімальне значення розподілу. Тоді розподіл почнеться з нашого мінімального значення і, отже, буде більш наближеним до реального. Маємо нове середнє:

$$\mu'_x = \mu_x - x_{min}$$

І нова формула для обрахунку загальних запасів також додаватиме мінімальне значення [6]:

$$\iota = F_\Gamma^{-1}(\alpha, k', \theta') + x_{min}$$
$$S_s = \iota - \mu_x$$

2.6 Визначення розподілу

Перед тим, як використовувати формули з оптимізації запасів, нам потрібно визначити, який розподіл використовувати. Цей тест називається *good-of-fit test.* Існує багато видів таких тестів, наприклад тест Колмогорова-Смірнова. Також, одним із найпростіших методів з визначення розподілу є обрахунок суми квадратів залишків (*Residual Sum of Squares, RSS*) із функцією щільності класичних розподілів і щільністю вхідного набору даних. Для того, щоб облегшити вибір розподілу, можна обрати до п’яти відомих розподілів і обрати той, у якого сума квадратів залишків буде найменшою. Суть цього тесту полягає в тому, що ми «підганятимемо» класичні розподіли до того, який є у нас і, таким чином, зможемо знайти розподіл, який підходить найбільше нашому набору даних.

Отже, для побудови тесту RSS, необхідно обчислити щільність вхідних даних, функцію щільності обраних розподілів і обрахувати суму квадратів їх різниць.

$$RSS = \sum_{i=1}^{n} (pdf_i(input) - pdf_i(classical))^2$$

Тест на визначення підходящого розподілу можна зробити автоматично, використовуючи бібліотеку *distfit* у python. Все, що вимагається – це вказати вхідні дані у функцію distfit і розподіли, які ми бажаємо протестувати. В результаті ми отримаємо список із розподілами і відповідний RSS до них. Також, за допомогою функції plot_summary можна вивести графік з результатами.

2.7 Модель симуляції продажів

Маючи часовий ряд попиту і числа необхідного і страхового запасів, ми можемо провести симуляцію продажів. Симуляція продажів є інструментом, який дозволяє протестувати прибутковість тієї чи іншої бізнес моделі. Для симуляції використовується кілька політик, які визначаються початково бізнес моделлю.

- (s, Q) – політика з постійною перевіркою і точкою повторного замовлення. Згідно з нею, на кожному кроці симуляції перевіряються наявні запаси і у випадку, якщо вони перетинають деяку межу s, виконується повторне замовлення фіксованої кількості Q. Приклад цієї симуляції зображено на рисунку.

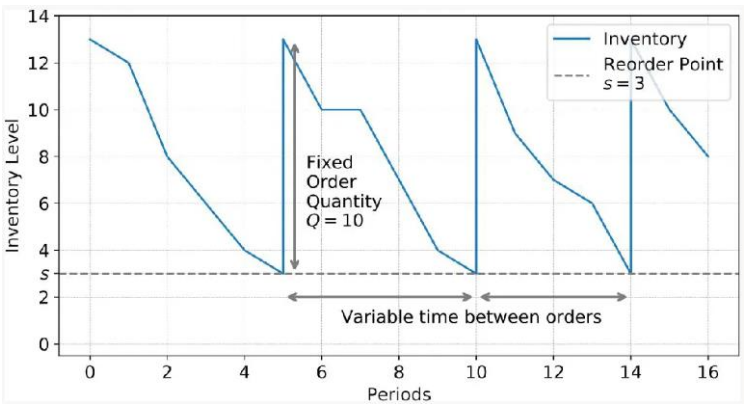

Рисунок 2- політика симуляції (s, Q) *[6]*

- (R, S) – політика з періодичною перевіркою і замовленням до необхідного рівня. Згідно з цією політикою, огляд кількості запасів, що лишилися, відбувається з фіксованого графіку, а поповнення запасів виконується до фіксованого рівня, але кожного разу замовлення відбуватиметься на різну кількість товарів. R – фіксований період перевірки, S – необхідна кількість запасів, до якої відбувається поповнення. Приклад цієї симуляції зображено на рисунку.

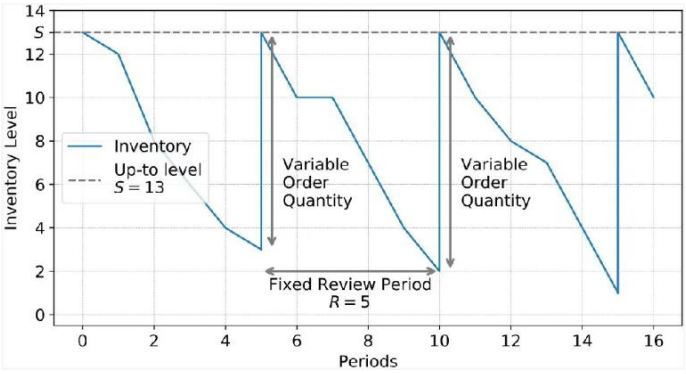

Рисунок 3- політика симуляції (R, S) *[6]*

- (R, s, Q) – політика з періодичним оглядом, точкою повторного замовлення і фіксованою кількістю замовлення. Ця політика об'єднує попередні дві. Тобто, ми замовляємо фіксовану кількість товарів кількості Q, якщо запаси сягають межі s.

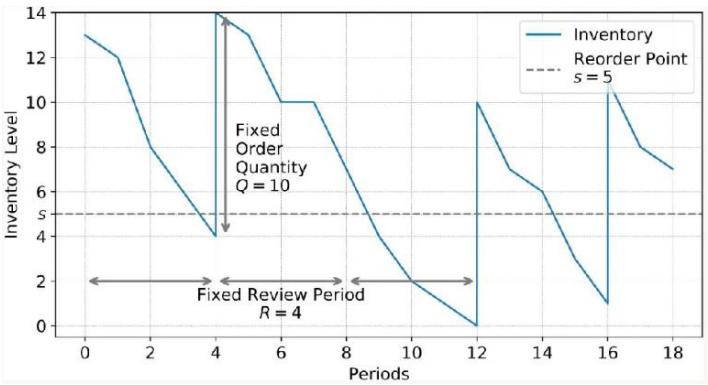

Рисунок 4- політика симуляції (R, s, Q) *[6]*

2.8 Висновки до розділу 2

Цей розділ було присвячено теорії оптимізації запасів. Було визначено такі метрики як необхідні запаси і страхові запаси, які будуть використовуватися для симуляції продажів і запасів. Також, були визначені політики для симуляції, за допомогою яких можна буде зробити оптимальний вибір між необхідними і страховими запасами.

3. Прогнозування попиту в датасеті з продажу напоїв

3.1 Бібліотеки

Для нас мова Python буде найбільш зручною, оскільки в неї достатньо функціоналу для виконання всіх задач, оскільки з Python легко застосувати алгоритми прогнозування, аналізувати дані і візуалізувати результати. Також, на цій мові програмування вже створена велика кількість бібліотек, яка полегшить обрахунки. Ось декілька бібліотек, які знадобляться для виконання задачі:

- Pandas – бібліотека для роботи з датасетом, за допомогою якої можна легко обробляти і перетворювати дані.
- Numpy – бібліотека для роботи з числовими масивами.
- Scipy – бібліотека, яка містить велику кількість статистичних методів.
- Scikit-learn – бібліотека, в якій імплементовані алгоритми машинного навчання.
- LightGBM – бібліотека з алгоритмом LightGBM, який ми використовуватимемо для прогнозування попиту.

3.2 Опис датасету

Маємо датасет із продажів різних напоїв у шести магазинах в Греції. Датасет був взятий з сайту: Kaggle [9]. Завантажуємо його:

```
gr_sales = pd.read_csv("train.csv")
gr_test = pd.read_csv("test.csv")
```

Рисунок 5- завантаження датасету

Датасет має такі змінні (колонки):

- Дата
- Місто

- Довгота
- Широта
- Населення
- ID Магазину
- Бренд
- Упаковка
- Ємність
- Ціна
- Кількість проданих товарів.

Перші п’ять елементів виглядають так:

date	city	lat	long	pop	shop	brand	container	capacity	price	quantity
31/01/12	Athens	37.97945	23.71622	672130.0	shop_1	kinder-cola	glass	500ml	0.96	13280.0
31/01/12	Athens	37.97945	23.71622	672130.0	shop_1	kinder-cola	plastic	1.5lt	2.86	6727.0
31/01/12	Athens	37.97945	23.71622	672130.0	shop_1	kinder-cola	can	330ml	0.87	9848.0
31/01/12	Athens	37.97945	23.71622	672130.0	shop_1	adult-cola	glass	500ml	1.00	20050.0
31/01/12	Athens	37.97945	23.71622	672130.0	shop_1	adult-cola	can	330ml	0.39	25696.0

Рисунок 6- Перші п’ять елементів датасету

Датасет, на якому відбуватиметься тренування, містить 7560 елементів. Для кожної трійки (*ID* магазину, Бренд, Упаковка) період даних починається 2012-01, а закінчується 2017-12, тобто цей період – 60 місяців.

Для тестового набору даних період починається 2018-01, а закінчується 2018-12. Тобто період для тестової вибірки- 12 місяців. Довжина тестового набору- 1010 елементів.

3.3 Постановка задачі

Тобто, маємо історичні дані продажів у 6 магазинів, 5 брендів і 3 види упаковок. Нашою задачею буде спрогнозувати попит з продажів для кожного магазину, бренду і ємності, користуючись історичними даними; порахувати

ключові метрики для оптимізації запасів і змоделювати прибутки, які ми отримаємо, застосувавши ці метрики.

3.4 Обробка датасету

По- перше, необхідно прибрати значення Nan. Прибравши їх, тренувальна вибірка зменшиться до 6376 елементів.

Змінна «Дата» - формату DateTime, і для того, щоб використовувати їх у навчанні, ми можемо зробити невелике перетворення- зробити зі змінної формату DateTime числових змінних «Місяць» і «Рік». Це допоможе алгоритму у врахуванні сезонності, якщо вона є.

Змінні Довгота, Широта, Населення, Ціна і Кількість- числові змінні. Їх перетворювати не треба.

Змінні ID Магазину, Бренд, Упаковка, Ємність – категоріальні. Їх необхідно перетворити.

Також, проаналізуємо змінну Кількість. За допомогою Pandas є можливість її описати. Буде виведена інформація про середнє значення, середньоквадратичне відхилення, мінімальне і максимальне значення і квантилі:

```
gr_sales['quantity'].describe()
```

Рисунок 7- Команда для опису залежної змінної

```
count      6376.000000
mean      29416.483061
std       17891.264214
min        2953.000000
25%       16439.750000
50%       25180.000000
75%       37804.000000
max      145287.000000
```

Рисунок 8- Опис змінної “quantity”

3.5 Підготовка датасету

Для того, щоб застосувати алгоритм LightGBM, ми повинні перетворити датасет у такий, щоб алгоритм зміг прочитати всі змінні. Тож,

- Змінну «Дата» формату «День/Місяць/Рік» перетворюємо у дві змінні «Місяць» і «Рік». «День» не враховуємо, бо всі продажі відбуваються помісячно в різні дні.
- Змінні «Місто», «ID Магазину», «Бренд», «Упаковка», «Ємність» перетворюємо у категоріальні змінні за допомогою LabelEncoder.
- Виділяємо змінну «Кількість проданих пляшок» у залежну змінну y.
- Виділяємо всі інші змінні як незалежні – масив X

Перші п’ять елементів тренувальної вибірки виглядають так:

	city	lat	long	pop	shop	brand	container	capacity	price	year	month
0	0	37.97945	23.71622	672130.0	0	2	1	2	0.96	2012.0	1.0
1	0	37.97945	23.71622	672130.0	0	2	2	0	2.86	2012.0	1.0
2	0	37.97945	23.71622	672130.0	0	2	0	1	0.87	2012.0	1.0
3	0	37.97945	23.71622	672130.0	0	0	1	2	1.00	2012.0	1.0
4	0	37.97945	23.71622	672130.0	0	0	0	1	0.39	2012.0	1.0

Рисунок 9- Перші 5 елементів готового до тренування датасету

3.6 Застосування LightGBM

Після того, як ми підготували датасет до роботи, можемо застосувати LightGBM. Оскільки ми вирішуємо задачу регресії, з бібліотеки lightgbm завантажуємо LGBMRegressor. Далі застосовуємо його до датасету і отримуємо прогнозовані дані- y_{pred}.

Також для порівняння застосуємо алгоритм випадкового лісу і порівняємо метрику *RMSE,* яка показує, наскільки коректним є навчання алгоритму.

Для візуалізації роботи алгоритму, а саме передбачень, можна зрозуміти, що алгоритм навчився добре і прогнози на тестовий період майже співпадають з тестовою вибіркою:

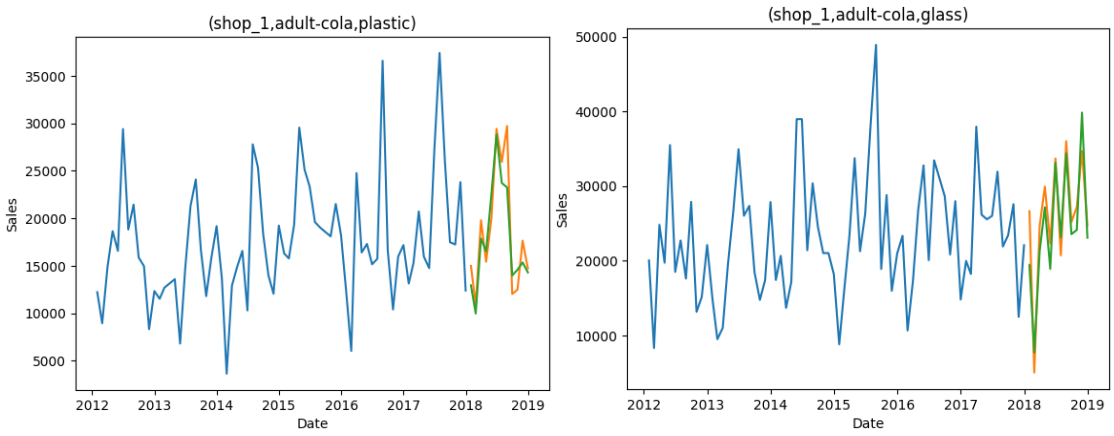

Рисунок 10,11- Візуалізація прогнозу для кількох груп з LightGBM (зеленим), візуалізація тестової вибірки (жовтим), візуалізація тренувальної вибірки (синім)

Метрика середньоквадратичного відхилення для всієї вибірки для всіх магазинів також показує, зо алгоритм добре навчився:

$$RMSE_{lightGBM} = 14609177$$

$$RMSE_{Random\ Forest} = 22745292$$

Порівнявши роботу Random Forest і LightGBM, можна зробити висновки, що LightGBM навчився краще.

3.7 Оптимізація запасів

Перед тим, як рахувати оптимізаційні метрики, необхідно зрозуміти, з яким розподілом продажів ми працюємо.

Зробимо дві гістограми: змінної всіх продажів і продажів, розбитих по магазинам. А також визначимо розподіл за допомогою команди:

```
sns.displot(gr_sales['quantity'], kde = True, color = 'g')
```

Рисунок 12- Команда для гістограми набору даних

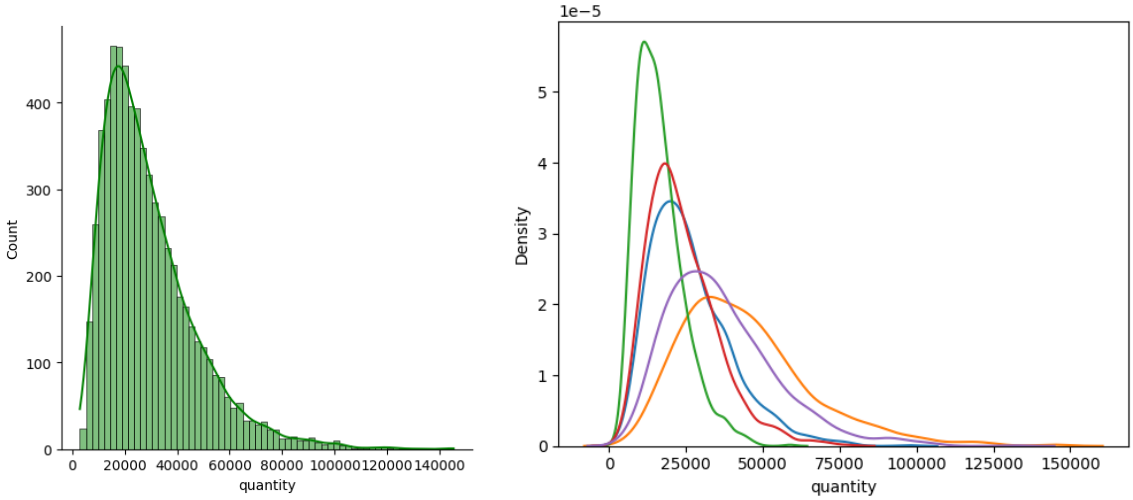

Рисунок 13,14- Розподіл змінної "quantity"

За допомогою бібліотеки *distfit* проведемо тест на розподіл, щоб зрозуміти, до формул оптимізації якого розподілу застосувати вхідні дані.

Для прикладу, візьмемо 4 класичних розподіли- Нормальний, Експоненційний, Т і Лог-Гамма. І проведемо тест:

```
dfit = distfit(method='parametric', todf=True,distr=['norm','gamma','t','loggamma'])
dfit.fit_transform(y_train)
# Print model results
dfit.model
```

Рисунок 15- команда для тесту на розподіл

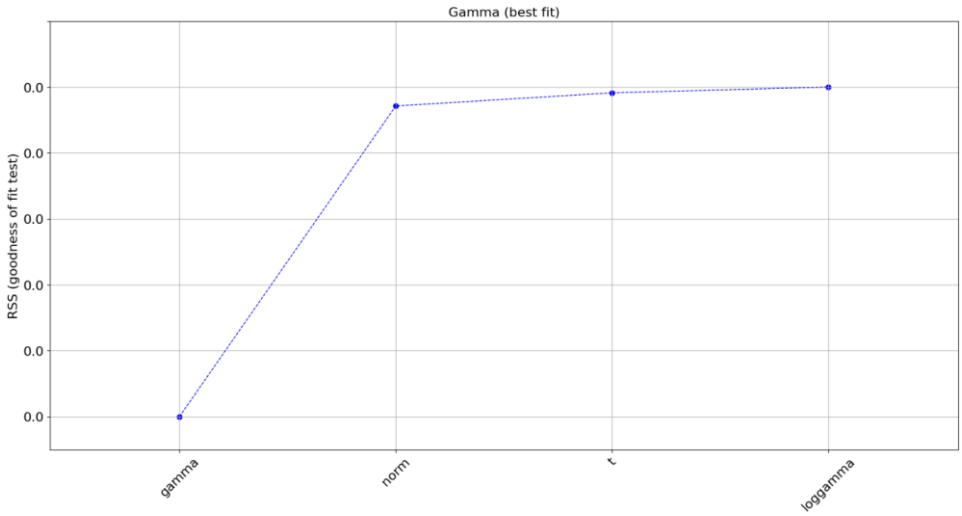

Рисунок 16- Результат тесту на розподіл з distfit

Як бачимо, оскільки гамма розподіл має найменший RSS. Тому ми маємо справу з гамма розподілом. Тобто, нам буде потрібно рахувати всі метрики відповідно до гамма розподілу.

Оскільки наш датасет має мінімум $\min = 2953$, необхідно виконати усі розрахунки, використовуючи формули зі зміщеним середнім.

Для кожної трійки (ID магазину, Бренд, Упаковка) рахуємо параметри гамма розподілу і відповідні параметри необхідних запасів і страхових запасів. Всю інформацію зібрано в новий датасет *group_info*:

shop_name	brand_name	container	Inventory	Safety Stocks
shop_1	adult-cola	can	150571.056898	21961.563567
shop_1	adult-cola	glass	114225.605027	15135.046746
shop_1	adult-cola	plastic	87197.113214	11602.774827
shop_1	gazoza	can	226097.928287	31134.116941
shop_1	gazoza	glass	200884.662984	26799.385314
...	...	...	...	...
shop_6	lemon-boost	glass	169165.639329	21604.171443
shop_6	lemon-boost	plastic	123283.869835	15652.651447
shop_6	orange-power	can	311882.338196	41576.103342
shop_6	orange-power	glass	250910.464141	34191.035666
shop_6	orange-power	plastic	196162.851061	25746.853132

Рисунок 17- Результат обрахунку необхідних і страхових запасів

Наприклад, для $(shop\_1,\ adult\_cola,\ plastic)$ страхові запаси дорівнюють $S_s = 11602$, а необхідні запаси $\iota = 87197$.

3.8 Симуляція продажів

Отримавши результати прогнозування попиту за допомогою LightGBM з одного боку і метрики необхідних запасів і страхових запасів з іншого боку, ми можемо провести симуляцію продажів для кожної трійки (ID магазину, Бренд, Упаковка) для того, щоб візуалізувати увесь процес.

Для симуляції обрано політику (s, Q) через те, що все ж таки, датасет має обмежену інформацію і ми не маємо даних про собівартість товарів і швидкість їх доставки.

Як відбуватиметься експеримент: ми порахували необхідні запаси, від них на кожній ітерації часу відніматимуться прогнозовані продажі. Коли кількість інвентаря досягнуть кількості страхових продажів, ми поповнюємо запаси на рівень необхідних. Нехай день поповнення запасів буде тим самим днем, коли запаси закінчилися. Це означає, що кількість днів доставки дорівнює нулю, це також називається негайним поповненням (immediate replenishment).

Приклад симуляції для трійки (ID магазину, Бренд, Упаковка):

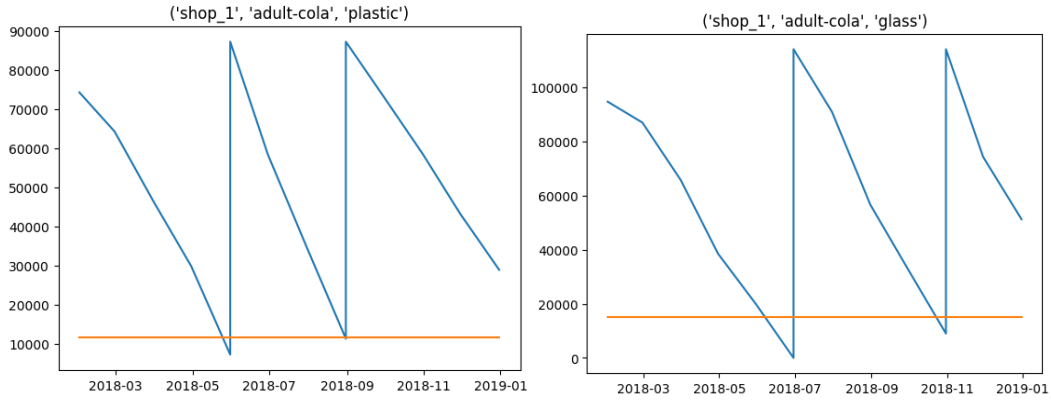

Рисунок 18,19- Результат симуляції продажів

Можемо побачити, що коли наявні запаси перетинають межу страхових запасів або дорівнюють нулю, вони автоматично поповнюються до рівня необхідних, які були пораховані для відповідної трійки.

Для повторного замовлення використовувались деяка група правил, яка може відрізнятися в залежності від бізнес задач.

- Фіксоване значення запасів, перетинаючи яке відбувається повторне замовлення
- Фіксована кількість повторного замовлення. Це число дорівнює кількості необхідного замовлення
- Період повторного замовлення є невідомим

Ці правила описують політику, що ми обрали. Також варто зауважити, що період повторного замовлення не є постійним, тому він відрізнятиметься від одного циклу до іншого.

Оскільки запаси не можуть бути менше за нуль, а прогнозований попит може вивести значення запасів менше нуля, ми можемо позначити цю подію як прогнозовані збитки.

Провівши ще одну симуляцію, ми можемо порахувати потенціальні доходи і збитки для кожного магазину по бренду і визначити, які з них будуть найбільш прибутковими, а які найбільш збитковими.

Для цього, у циклі ми зробимо два масиви: доходів і збитків. Якщо прогнозовані продажі виводять запаси у від'ємне значення, ми позначаємо це як збитки, якщо запаси залишаються більші за нуль, це прибутки. Також, у датасеті маємо ціну кожного товару, тому, перемноживши кількість на ціну, можна отримати і доходи і збитки.

Чисті прибутки отримаємо, якщо від доходів віднімемо втрати.

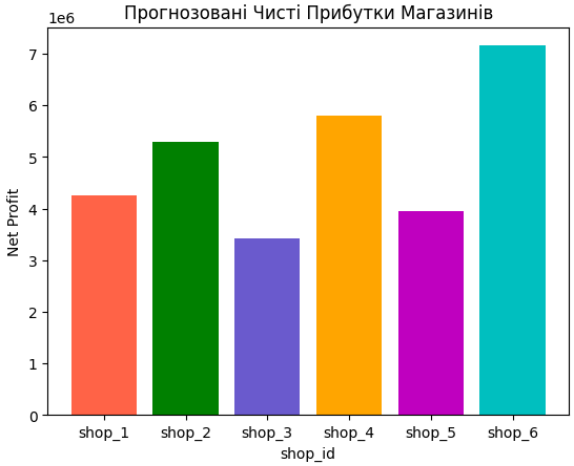

Рисунок 20- Прогнозовані чисті прибутки магазинів

3.9 Валідація

Останній крок- порахуємо як співставляються прогнозовані доходи з втратами. Для цього поділимо витрати на втрати і отримаємо відсоткове відношення, яке покаже, наскільки втрати відрізняються від доходів.

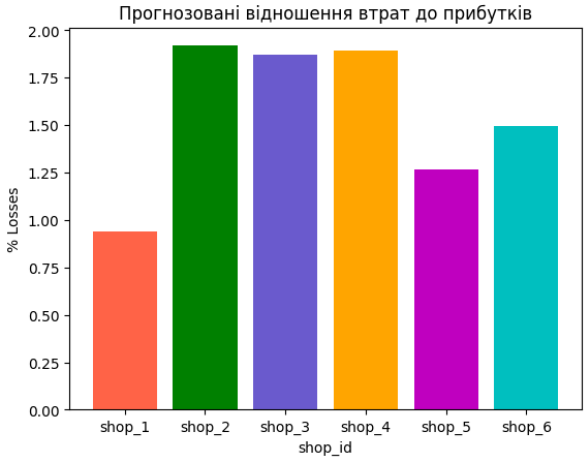

Рисунок 21- Відношення втрат до доходів у %

Як можемо побачити, відношення втрат до доходів є меншим за 2%, тому, оскільки це дуже незначні відсотки, можна вважати модель прогнозування і симуляцію валідною.

Висновки

У даній роботі було розглянуто алгоритм LightGBM, який є потужним інструментом для прогнозування даних. Також, були досліджені елементи з теорії оптимізації запасів, які допомогли в обрахунках необхідних і страхових запасів магазинів.

Завдання в цій роботі було розділено на дві частини. Перша- знайти прогнози для часового ряду продажів кожного набору даних (ID магазину, Бренд, Упаковка). Для цього був використаний алгоритм LightGBM. Так ми отримали майбутні прогнозовані помісячні продажі на наступний рік.

Друга частина полягала в тому, щоб зробити симуляцію продажів, маючи прогнозовані дані, необхідні запаси і страхові запаси, використовуючи політику з постійною перевіркою і точкою повторного замовлення.

Результатом роботи є запропоновані симуляційні моделі, які допомогли описати динаміку продажів для усіх магазинів. Це дало змогу зрозуміти, як будуть витрачатися запаси і чи вистачить їх для того, щоб, використовуючи такі моделі, бути прибутковими. Оскільки відношення втрат до доходів не перевищувало 2%, можна зробити висновки, що такі моделі є коректними і корисними.

Список використаних джерел

1. Mirzaee, Ashkan. (2017). Alternative methods for calculating optimal safety stock levels.

2. Müller, S. (2020). Data-driven inventory optimization [Dissertation].

3. Peng, Tu & Chen, Xiaoya et al. (2020). The Prediction of Hepatitis E through Ensemble Learning.

4. Re, Matteo & Valentini, Giorgio. (2012). Ensemble methods: A review.

5. Ke G, Meng Q, Finley T, et al. (2020). Lightgbm: A highly efficient gradient boosting decision tree.

6. Vandeput, Nicolas. Inventory Optimization: Models and Simulations. 10.1515/9783110673944.

7. Towardsdatascience, MAE і RMSE [Електронний ресурс] / Режим доступу до ресурсу:

https://towardsdatascience.com/what-are-rmse-and-mae-e405ce230383

8. Towardsdatascience, Симуляції запасів [Електронний ресурс] / Режим доступу до ресурсу:

https://towardsdatascience.com/make-your-inventory-simulation-in-python-9cb950da8cf3

9. Kaggle [Електронний ресурс] / Режим доступу до ресурсу:

https://www.kaggle.com/datasets/veeralakrishna/predict-demand

10. Izza, Yacine & Ignatiev, Alexey & Marques-Silva, Joao. (2020). On Explaining Decision Trees.

11. Vandeput, Nicolas. (2021). Data Science for Supply Chain Forecasting.

12. Twill, Supply chain forecasting [Електронний ресурс] / Режим доступу до ресурсу:

twill.net/knowledge-hub/logistics-know-how/supply-chain-forecasting

13. Thecleverprogrammer, Demand prediction [Електронний ресурс] / Режим доступу до ресурсу:

thecleverprogrammer.com/2021/11/22/product-demand-prediction-with-machine-learning

14. Tao, Hai & Salih, Sinan & Oudah, Atheer & Abba, Sani & Ameen, Ameen & Awadh, Salih & A. Alawi, Omer & Mostafa, Reham & Udayar Pillai, Surendran & Yaseen, Zaher. (2022). Development of new computational machine learning models for longitudinal dispersion coefficient determination: case study of natural streams, United States. Environmental Science and Pollution Research. 29. 10.1007/s11356-022-1855