ПРИВАТНЕ АКЦІОНЕРНЕ ТОВАРИСТВО «ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «МІЖРЕГІОНАЛЬНА АКАДЕМІЯ УПРАВЛІННЯ ПЕРСОНАЛОМ»

Факультет комп'ютерно-інформаційних технологій

Кафедра комп'ютерних інформаційних систем і технологій

Карасевич Олександр Ігорович

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Оптимізація стратегій продажів та управління товарними запасами засобами прогнозної аналітики»

Група: ІК-9-22-Б1КНТ (4.0д)

Спеціальність: Комп'ютерні науки

Рівень вищої освіти: перший (бакалаврський)

Кваліфікаційна робота містить результати власних досліджень. Ввикористання ідей, результатів, текстів інших авторів мають посилання на відповідне джерело.

	_____	Карасевич О.І.
	(підпис)	(ПІБ студента)
Науковий керівник	_____	Яременко Д.М.
	(підпис)	(ПІБ)

Допущено до захисту ЕК

Завідувач кафедри _____ Сергій КАВУН , д.е.н., професор

Київ 2026

ЗМІСТ

ЗМІСТ..... 2
РЕФЕРАТ..... 3
ВСТУП..... 4
Розділ 1. Теоретико-технологічний аналіз інтелектуальних систем управління продажами..... 6
1.1. Архітектура сучасних систем прогнозної аналітики..... 6
1.2. Математична постановка задачі оптимізації запасів..... 9
1.3. Порівняльний аналіз парадигм машинного навчання для ритейлу..... 11
1.4. Огляд технологічного набору та інструментарію розробки..... 18
Розділ 2. Математичне моделювання та методологія підготовки даних для інтелектуального аналізу продажів..... 24
2.1. Методи інженерії ознак та попередньої обробки часових рядів у ритейлі..... 24
2.2. Стратегії валідації та метрики оцінки якості прогнозних моделей..... 28
2.3. Математичне обґрунтування алгоритмів ансамблевого навчання та градієнтного спуску..... 32
Розділ 3. Проєктування архітектури та розробка інтелектуальної системи аналізу та оптимізації продажів..... 38
3.1. Обґрунтування архітектурних рішень та функціональних вимог до системи..... 38
3.2. Проєктування логічної структури даних та конвеєрів обробки (ML-Pipelines)..... 42
3.3. Програмна реалізація модулів навчання та інференсу моделей. (Опис структури коду, класів та методів, які ти будеш писати)..... 45
3.4 Розробка засобів візуалізації та інтерфейсу взаємодії з користувачем.... 49
Розділ 4. Експериментальна Апробація та Оцінка Ефективності Системи.... 54
4.1. Опис експериментального набору даних та умов тестування..... 54
4.2 Навчання моделі, налаштування гіперпараметрів та оцінка точності..56
4.3. Апробація програмного інтерфейсу та інтерпретація результатів прийняття рішень..... 59
ВИСНОВКИ..... 64
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 66
ДОДАТКИ..... 68

РЕФЕРАТ

Кваліфікаційна робота бакалавра: 71 с., 31 рис., 8 табл., 20 джерел, 2 додатки.

Об'єкт дослідження – бізнес-процеси управління продажами та товарними запасами на підприємствах роздрібної торгівлі в умовах високої волатильності ринку.

Предмет дослідження – методи та засоби прогнозної аналітики, алгоритми машинного навчання (Machine Learning) та інтелектуальні системи підтримки прийняття рішень.

Мета роботи – розробка інтелектуальної системи прогнозування попиту на базі алгоритму градієнтного бустингу для оптимізації товарних залишків та мінімізації логістичних витрат.

Методи дослідження: методи машинного навчання (CatBoost), аналіз часових рядів, інженерія ознак (Feature Engineering), методи математичного моделювання (модель «газетяра», функція втрат Pinball Loss), принципи об'єктно-орієнтованого проєктування на мові Python.

У роботі проведено аналіз сучасних тенденцій ритейлу та архітектурних рішень для ML-систем (Data Lakehouse, Feature Store). Розроблено математичну модель прогнозування попиту, стійку до нелінійних коливань. Реалізовано програмний комплекс із використанням бібліотек CatBoost та Streamlit, що забезпечує візуалізацію прогнозів та сценарний аналіз. Проведено експериментальну перевірку моделі, яка продемонструвала високу точність (WAPE < 10%).

Практична цінність роботи полягає у створенні робочого прототипу аналітичної панелі (Dashboard), яка дозволяє менеджерам із закупівель приймати обґрунтовані рішення на основі прогнозних даних та інтерпретувати результати ШІ за допомогою методів SHAP.

Ключові слова: ПРОГНОЗУВАННЯ ПОПИТУ, МАШИННЕ НАВЧАННЯ, CATBOOST, ОПТИМІЗАЦІЯ ЗАПАСІВ, PYTHON, STREAMLIT, ПРЕДИКТИВНА АНАЛІТИКА, ІНТЕРПРЕТОВАНІСТЬ МОДЕЛЕЙ.

ВСТУП

Актуальність теми. У 2026 році глобальний ритейл та сфера електронної комерції остаточно перейшли до моделі «Hyper-local & Real-time», де успіх бізнесу залежить від здатності передбачати запит клієнта ще до того, як він буде сформульований. Традиційні методи аналізу, що базуються на ретроспективних статистичних показниках (на кшталт змінного середнього), виявилися неспроможними в умовах високої волатильності ринку, спричиненої глобальними логістичними кризами та зміною споживчої поведінки. Використання прогнозної аналітики на базі Machine Learning (ML) дозволяє трансформувати накопичені «сирі» дані у стратегічний актив. Оптимізація товарних запасів є критично важливою, оскільки надлишок товарів веде до «заморожування» капіталу та витрат на зберігання, а дефіцит — до втрати лояльності клієнтів та прямого зниження прибутку. Впровадження інтелектуальних систем дозволяє автоматизувати прийняття рішень, мінімізуючи людський фактор та забезпечуючи стійкість бізнес-моделі в умовах невизначеності.

Об’єкт дослідження. Бізнес-процеси управління продажами та запасами на складі в умовах сучасного ритейлу, що характеризуються великими обсягами даних транзакцій.

Предмет дослідження. Математичні моделі, методи та алгоритми машинного навчання, призначені для короткострокового та середньострокового прогнозування попиту, а також програмні засоби їх реалізації.

Мета роботи. Підвищення ефективності управління торговельною діяльністю шляхом розробки та впровадження інтелектуальної системи прогнозування продажів та оптимізації товарних потоків.

Завдання дослідження:

1. Провести системний аналіз предметної області та виявити ключові чинники, що впливають на точність прогнозів у сучасних умовах.

2. Дослідити теоретичні основи та математичний апарат сучасних методів Machine Learning (зокрема ансамблевих методів та нейронних мереж).
3. Виконати порівняльний аналіз існуючих архітектурних рішень для обробки великих масивів комерційних даних.
4. Розробити методику попередньої обробки даних (Feature Engineering) з урахуванням факторів сезонності, маркетингових акцій та зовнішніх впливів.
5. Проєктувати та програмно реалізувати систему прогнозної аналітики.
6. Провести експериментальне оцінювання розроблених моделей на реальних (або наближених до реальних) наборах даних та обґрунтувати економічну доцільність рішення.

Наукова новизна отриманих результатів полягає у вдосконаленні методики комбінування класичних моделей аналізу часових рядів з алгоритмами градієнтного бустингу, що дозволяє підвищити точність прогнозування в умовах «рваного» попиту та обмеженої вибірки даних для нових категорій товарів.

Практичне значення роботи. Розроблене програмне забезпечення може бути інтегроване в існуючі ERP (enterprise resource planning system — Система планування ресурсів підприємства) та CRM-системи (від англ. Customer Relationship Management) підприємств для автоматизації замовлень постачальникам, планування маркетингових бюджетів та оптимізації логістичних витрат. Очікуваним результатом є зниження рівня неліквідних запасів на 10-15% та підвищення точності прогнозів на 20% порівняно з базовими моделями.

Методи дослідження. У роботі використано комплексний підхід, що базується на методах системного аналізу, теорії ймовірностей, математичної статистики та Machine Learning. Програмна реалізація виконана на мові Python з використанням сучасних фреймворків для Data Scienc

Розділ 1. Теоретико-технологічний аналіз інтелектуальних систем управління продажами

1.1. Архітектура сучасних систем прогнозної аналітики

Побудова ефективної системи прогнозної аналітики у 2026 році вимагає відмови від спрощених підходів до обробки даних. Сучасна архітектура Machine Learning (ML) – це не просто набір скриптів для навчання моделі, а багаторівнева екосистема, яка забезпечує безперебійний цикл від збору «сирої» інформації до отримання готового бізнес-прогнозу. Основна складність полягає в тому, що дані у сфері продажів надходять із різних джерел, мають різну структуру та часто містять шуми, які можуть критично спотворити результати прогнозування.

Процес починається на рівні збору даних (Data Ingestion). У ритейлі цей етап охоплює інтеграцію з транзакційними базами даних, системами складського обліку та зовнішніми API, які надають інформацію про ціни конкурентів або погодні умови, що впливають на попит. Важливо розуміти, що дані не просто копіюються; вони мають проходити через шлюзи, які забезпечують їхню цілісність. Наприклад, при використанні архітектури мікро-сервісів, дані про продажі можуть надходити у вигляді потоків подій (Event Streams) через такі інструменти, як Apache Kafka або Airflow (рис 1.1).

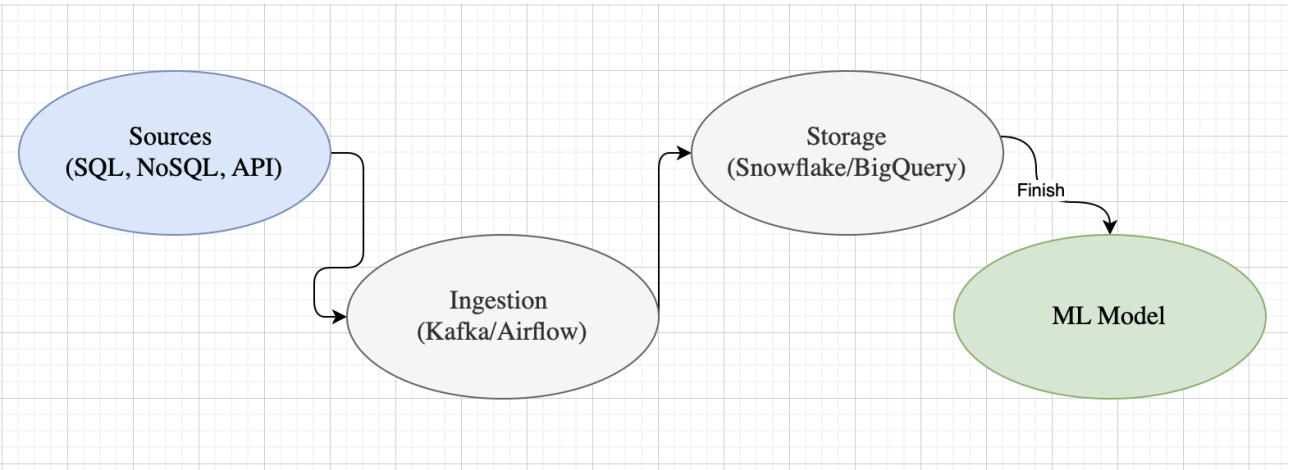


Рисунок 1.1 Схема Data Pipeline

Після збору дані потрапляють у середовище обробки, де реалізуються стратегії ETL (Extract, Transform, Load) або ELT (Extract, Load, Transform) (рис 1.2). Традиційний підхід ETL передбачає трансформацію даних на проміжному сервері перед їх завантаженням у цільове сховище. Це дозволяє очистити дані від дублікатів та виправити помилки введення ще до того, як вони займуть місце в базі. Проте, з поширенням хмарних технологій у 2026 році, домінуючим став підхід ELT. У цій парадигмі «сирі» дані спочатку завантажуються в хмарне сховище (Data Lake), а вже потім трансформуються за допомогою потужностей самої бази даних. Це забезпечує набагато вищу швидкість обробки великих масивів (Big Data), що є критичним для великих торговельних мереж.

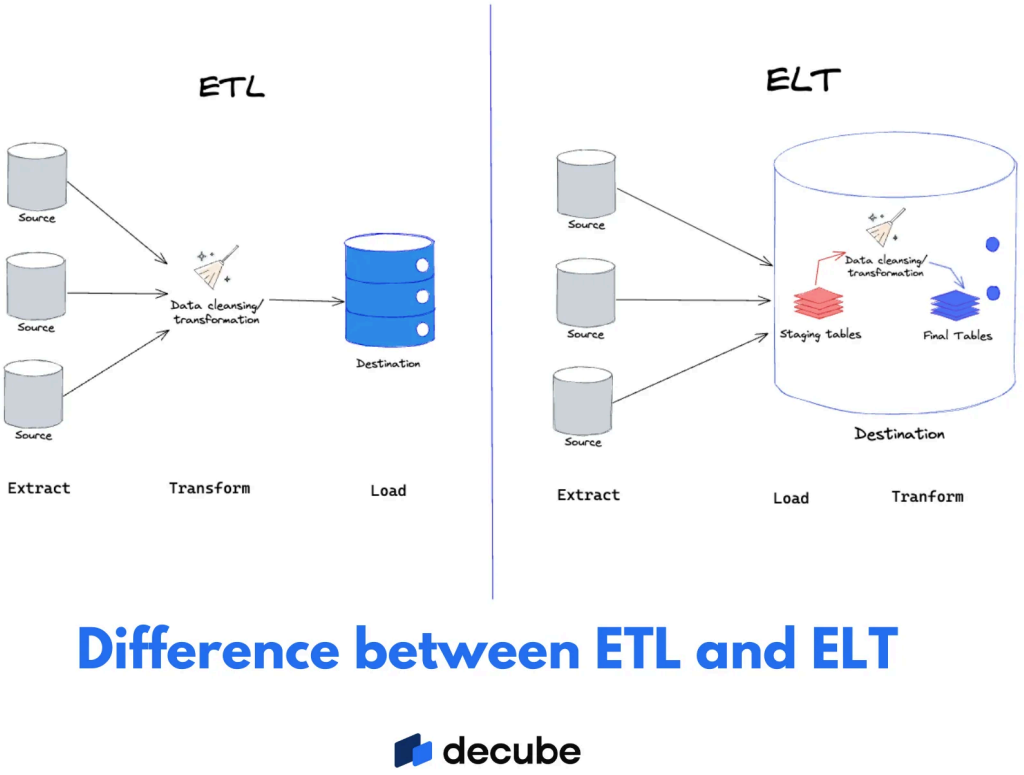


Рисунок 1.2 Візуальна різниця між ETL та ELT [4].

Центральним елементом зберігання є концепція Data Lakehouse. Вона поєднує в собі найкращі риси «озер даних» (можливість зберігати неструктуровані файли, логи, зображення чеків) та «сховищ даних» (сувора типізація та підтримка ACID-транзакцій). У контексті аналізу продажів це дозволяє аналітикам звертатися як до історичних даних за десятиліття, так і до

оперативних звітів за останню годину в межах однієї системи. Таке сховище зазвичай організовується за «медальйонною» архітектурою: рівень «Bronze» (сирі дані), «Silver» (очищені та відфільтровані дані) та «Gold» (агреговані показники, готові для аналізу).

Однією з найбільш інноваційних частин архітектури, що стала стандартом до 2026 року, є Feature Store (сховище ознак). У задачах машинного навчання «ознака» (feature) – це конкретна характеристика, яку модель використовує для прогнозу, наприклад, «середня кількість куплених одиниць товару певним клієнтом за останні 3 дні». Раніше розробникам доводилося заново обчислювати ці показники для кожного експерименту. Feature Store дозволяє централізовано зберігати ці обчислені вектори. Це запевняє, що модель під час навчання і модель під час реальної роботи на сервері (inference) бачать ідентичні дані, що усуває одну з найпоширеніших помилок у Data Science – логічний зсув даних (рис 1.3).

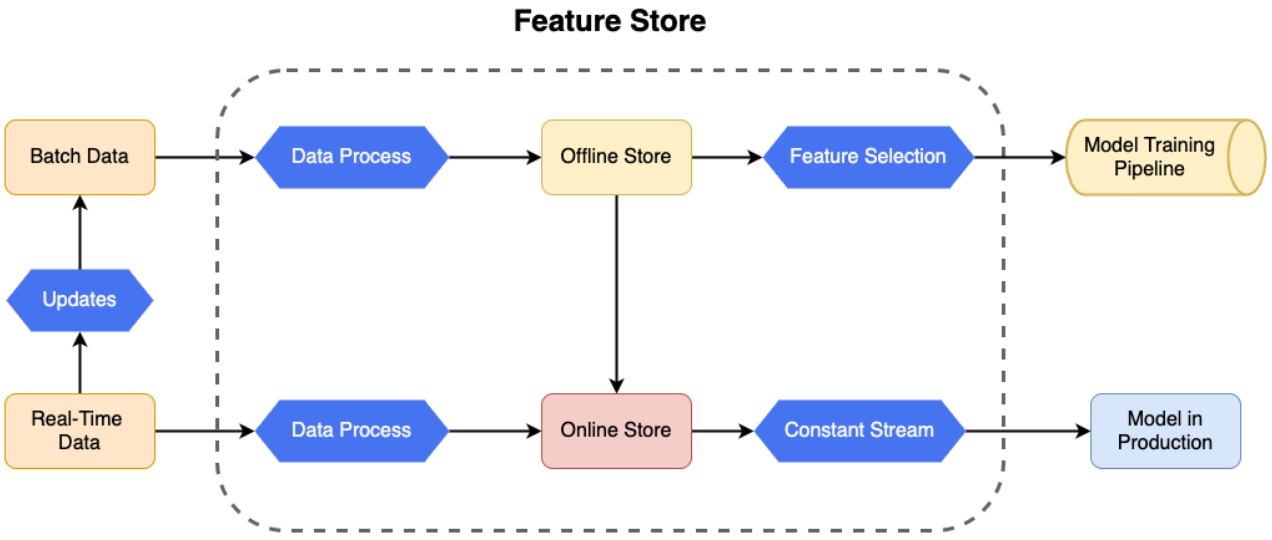


Рисунок 1.3 Feature Store [5].

Завершальним етапом архітектури є рівень виведення та споживання (Serving Layer). Тут результати роботи моделей машиного навчання перетворюються на конкретні рекомендації для менеджерів або автоматизовані команди для заготівельних систем. Прогнози можуть надаватися у форматі пакетної обробки (наприклад, щоночі система розраховує потребу складу на завтра) або в режимі реального часу через API. Такий комплексний підхід до

архітектури дозволяє створити систему, яка не просто видає цифри, а є надійним інструментом підтримки прийняття рішень, здатним витримувати високі навантаження та динамічно масштабуватися разом із ростом бізнесу.

1.2. Математична постановка задачі оптимізації запасів.

Перетворення абстрактної бізнес-мети «збільшення прибутку» у конкретну математичну модель є критичним етапом проектування інтелектуальної системи. У сфері ритейлу прибуток не є лінійною функцією від кількості замовленого товару. Навпаки, ми маємо справу з двома протилежними типами витрат: витратами на дефіцит (stockout costs) та витратами на надлишок (overstock costs). Математична постановка задачі полягає у знаходженні такого обсягу замовлення, який мінімізує сумарні очікувані витрати або, що еквівалентно, максимізує очікуваний прибуток у короткостроковій та довгостроковій перспективах.

Для побудови моделі введемо базові змінні:

1. **D** – випадкова величина попиту на товар за певний період;
2. **Q** – кількість одиниць товару, замовлених для реалізації;
3. **p** – роздрібна ціна одиниці товару;
4. **c** – собівартість закупівлі одиниці товару;
5. **h** – витрати на зберігання одиниці залишку (holding cost);
6. **s** – штраф за дефіцит одиниці товару (вартість втраченого продажу та репутаційні ризики).

У такому разі функція чистого прибутку **P(Q, D)** може бути записана наступним чином (Формула 1.1), (рис 1.4):

$$P(Q,\ D) = p\ *\ min(Q,\ D) - c\ *\ Q - h\ *\ max(0,\ Q - D) - s\ *\ max(0,\ D - Q)$$

Формула 1.1. Функція чистого прибутку

Фундаментом для оптимізації запасів залишається модель «газетяра», адаптована під можливості машинного навчання.

Ціна (p)	100
Собіварітсть (c)	40
Витрати на зберігання (h)	10
Штраф за дефіцит (s)	20
Середній попит (D)	50

Обсяг замовлень (Q)	Очікуваний Прибуток
0	-1000
5	-600
10	-200
15	200
20	600
25	1000
30	1400
35	1800
40	2200
45	2600
50	3000
55	2750
60	2500
65	2250
70	2000
75	1750
80	1500
85	1250
90	1000
95	750
100	500

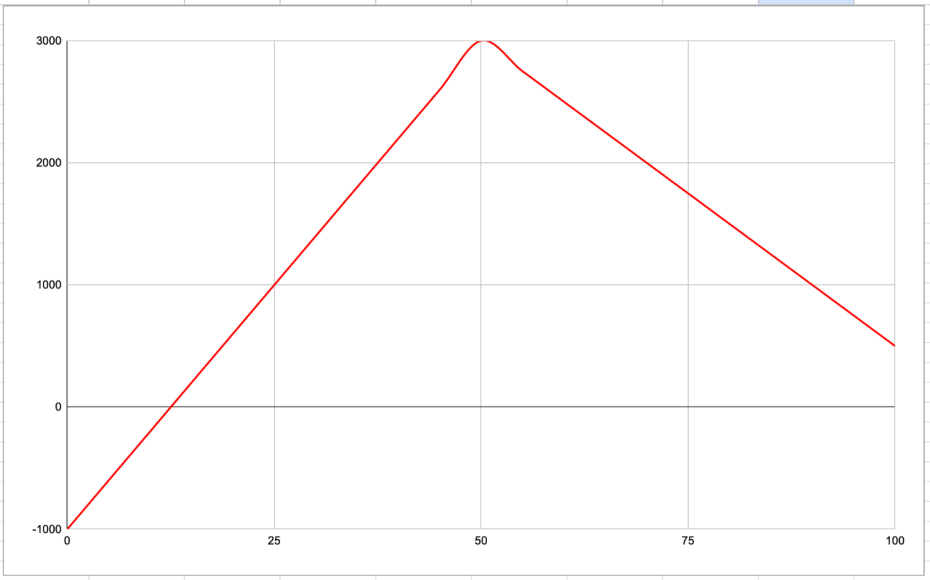


Рисунок 1.4. Графік прибутку.

Проблема продавця газет полягає в тому, що він повинен вирішити, скільки газет купити на початку дня. Він повинен прийняти рішення, ще не знаючи, скільки газет продасть. Так і з'явилася «проблема продавця газет», що дала початок «моделі продавця газет». **[10].**

Математична суть задачі полягає у пошуку оптимального рівня запасу **Q***, при якому очікувані витрати будуть мінімальними. Оптимальна точка досягається, коли ймовірність того, що попит **D** не перевищить рівень запасу **Q**, дорівнює «критичному коефіцієнту» (Формула 1.2):

$$P(D \leq Q^*) = (s + (p - c)) \div (s + (p - c) + h)$$

Формула 1.2. Critical Ratio [1].

Цей коефіцієнт показує, наскільки «сміливою» має бути модель. Якщо вартість втраченого продажу **s + p - c** висока, а вартість зберігання **h** мала, модель повинна схилятися до завищення прогнозів, щоб уникнути порожніх полиць.

У класичному машинному навчанні часто використовується середня квадратична помилка (MSE), яка однаково карає модель як за недопрогноз, так і за завищені очікування.

Середньоквадратична похибка (MSE) - це міра похибки алгоритмів прогнозування. Ця статистика кількісно виражає середньоквадратичну

дисперсію між спостережуваними та передбачуваними значеннями. Коли в моделі немає помилок, MSE дорівнює 0. Цінність моделі зростає пропорційно до ступеня помилки, яку вона містить. Середньоквадратичну похибку часто називають MSD - середньоквадратичне відхилення [9].

Проте в бізнесі ці помилки не є рівноцінними. Наприклад, для товарів з обмеженим терміном придатності (продукти харчування) перепрогноз на 10% набагато болючіший, ніж недопрогноз.

Для вирішення цієї проблеми ми впроваджуємо квантильну регресію (Quantile Regression) та відповідну їй функцію втрат – Pinball Loss **(**рис 1.5**)**.

$$\mathrm{L}_\alpha(y, \hat{y}) = \begin{cases} \alpha \cdot (y - \hat{y}) & \text{if } y \geq \hat{y} \\ (\alpha - 1) \cdot (y - \hat{y}) & \text{if } y < \hat{y} \end{cases}$$

Рисунок 1.5. Pinball Loss формула.

Цільовий квантиль, який відповідає обчисленому раніше критичному коефіцієнту. Таким чином, замість того, щоб вчити алгоритм вгадувати «середнє значення», ми вчимо його вгадувати таке значення, яке забезпечить оптимальний баланс грошей на складі та товарів на полиці.

Сучасна постановка задачі у 2026 році також враховує фактор часу. Ми розглядаємо не один період, а послідовність рішень, де залишок з сьогоднішнього дня впливає на витрати завтрашнього. Математично це описується за допомогою Марковських процесів прийняття рішень (MDP). Метою системи стає не просто максимізація прибутку від одного продажу, а оптимізація сумарного дисконтованого прибутку за весь життєвий цикл товару.

1.3. Порівняльний аналіз парадигм машинного навчання для ритейлу.

Вибір оптимального алгоритму прогнозування є ключовим етапом побудови інтелектуальної системи. У сучасній практиці аналізу даних виділяють три основні парадигми: класичні статистичні методи, ансамблеві методи на основі дерев рішень та глибоке навчання (Deep Learning). Кожна з них має свої

обмеження, обумовлені природою даних про продажі: високим рівнем шуму, наявністю трендів, сезонністю та впливом екзогенних факторів.

Класичний підхід базується на припущенні, що майбутні значення часового ряду залежать від його попередніх значень та певного випадкового шуму (рис 1.6). Найбільш відомою моделлю є **ARIMA (AutoRegressive Integrated Moving Average)**.

Математично модель **ARIMA(p, d, q)** описується як:

$$\phi(L)(1 - L)^{d} * y_t = \theta(L)\epsilon_t$$

Формула 1.3. ARIMA Model

де **L** – оператор лагу, **p** – порядок авторегресії, **d** – порядок інтеграції (різниці), **q** – порядок ковзного середнього.

Хоча ARIMA добре справляється зі стаціонарними рядами, у сфері продажів вона часто демонструє низьку точність через нездатність враховувати нелінійні зв'язки та зовнішні чинники (маркетингові акції, ціни конкурентів). Регресійні моделі (Linear Regression) з **L1** та **L2** регуляризацією (Lasso та Ridge) частково вирішують проблему мультиколінеарності ознак, проте все одно залишаються занадто спрощеними для реальних умов ринку.

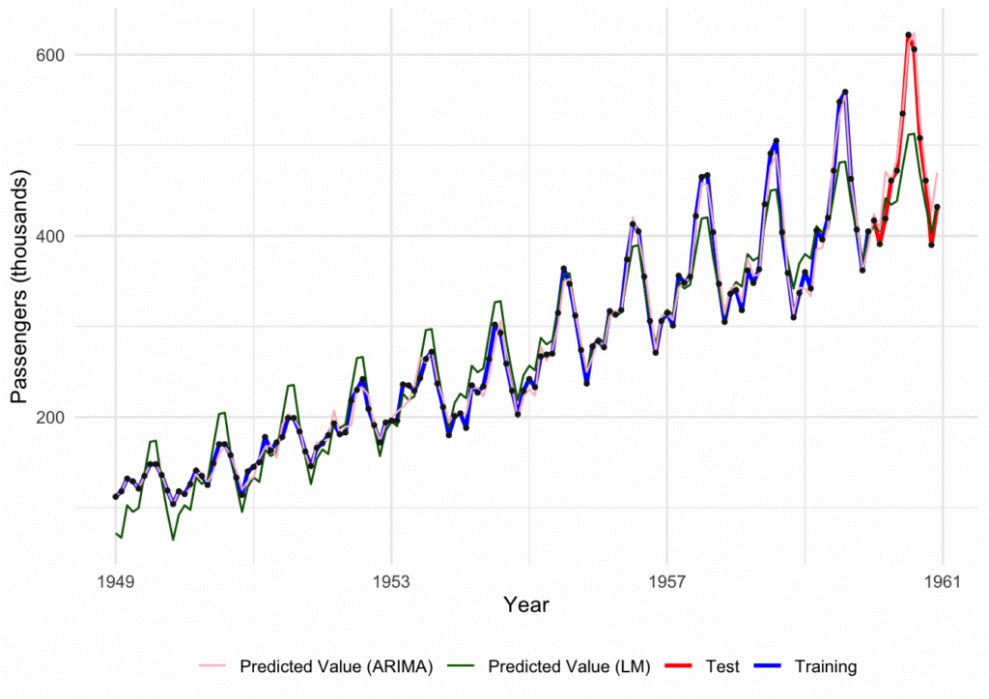


Рисунок 1.6. Графік декомпозиції часового ряду.

Ансамблеві методи. На сьогоднішній день ансамблеві методи вважаються «золотим стандартом» для роботи з табличними даними в ритейлі. Основна ідея полягає в об'єднанні багатьох «слабких» моделей (зазвичай дерев рішень) у одну «сильну».

- **Random Forest:** базується на принципі беггінгу (bagging) та випадковому виборі підмножини ознак. Це робить модель стійкою до викидів у даних про продажі та запобігає перенавчанню.
- **Gradient Boosting (XGBoost, LightGBM, CatBoost):** ці алгоритми послідовно будують дерева, де кожне наступне дерево виправляє помилки попередніх. Для аналізу продажів особливо ефективним є **CatBoost**, оскільки він має вбудовані механізми обробки категоріальних ознак (наприклад, ID магазину, категорія товару, день тижня), що позбавляє розробника необхідності вручну виконувати One-Hot Encoding.

Головною перевагою бустингу є здатність автоматично виявляти складні взаємодії між ознаками, наприклад, як поєднання «п'ятниця + дощ + знижка 10%» впливає на обсяг продажів конкретного розряду товарів.

При роботі з великою кількістю ознак виникає ризик **перенавчання** (overfitting). Модель стає занадто чутливою до шуму в історичних даних про продажі й втрачає здатність до узагальнення. Для запобігання цьому використовується метод регуляризації, який полягає у додаванні штрафного додання до цільової функції втрат **J(θ)**.

L2-регуляризація (Ridge Regression). Метод гребеневої регресії (Ridge) передбачає додавання суми квадратів коефіцієнтів моделі до функції втрат (формула 1.4.). Нова функція оцінки виглядає так:

$$J(\theta) = \sum_{i=1}^{n} y_i - y_i)^2 + \lambda \sum_{j=1}^{m} \theta_j^2$$

Формула 1.4. гребенева регресія

де λ — гіперпараметр регуляризації.

Математичний сенс L2-регуляризації полягає в тому, що вона «стискає» коефіцієнти θ ближче до нуля, але ніколи не робить їх рівними нулю. Це

особливо корисно у випадках сильної мультиколінеарності (коли чинники продажів сильно корелюють між собою, наприклад, ціна товару та ціна конкурента). Геометрично це можна представити як обмеження простору пошуку коефіцієнтів колом (або гіперсферою).

L1-регуляризація (Lasso Regression). Метод Lasso (Least Absolute Shrinkage and Selection Operator) використовує абсолютні значення коефіцієнтів:

$$J(\theta) = \sum_{i=1}^{n} y_i - y_i)^2 + \lambda \sum_{j=1}^{m} |\theta_j|$$

Формула 1.4. гребенева регресія

На відміну від L2, регуляризація L1 має унікальну властивість — вона здатна зануляти найменш важливі коефіцієнти (рис 1.7). Це фактично здійснює автоматичний відбір ознак (feature selection). У задачах аналізу продажів це дозволяє системі самостійно «відкинути» ті фактори, які не мають статистичного впливу на попит, залишаючи лише найбільш значущі.

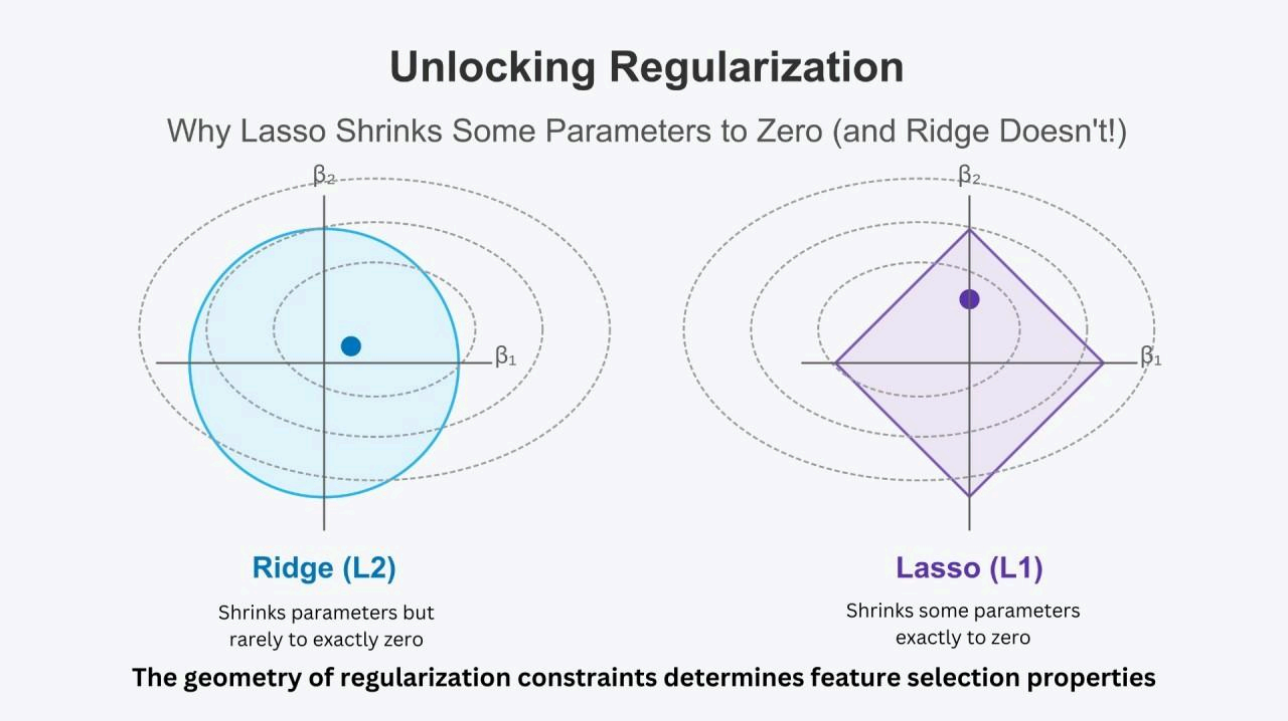


Рисунок. 1.7. Геометрична інтерпретація L1 та L2 регуляризації

Elastic Net: Гібридний підхід. У найсучасніших реалізаціях градієнтного бустингу (наприклад, в XGBoost) часто використовується комбінація обох

методів, відома як Elastic Net. Вона дозволяє одночасно підтримувати стабільність моделі (завдяки L2) та забезпечувати розрідженість ознак (завдяки L1).

Глибоке навчання (Deep Learning) для часових рядів. У 2026 році все більшої популярності набувають нейронні мережі, спеціально архітектурно адаптовані для послідовностей:

1. **LSTM (Long Short-Term Memory):** рекурентні нейронні мережі, здатні «запам’ятовувати» довгострокові залежності. Вони ідеально підходять для моделювання складних сезонних циклів, які можуть тривати рік і більше.
2. **Transformers (наприклад, Temporal Fusion Transformer):** механізми уваги (Attention) дозволяють моделі фокусуватися на найбільш значущих історичних моментах (наприклад, продажі в аналогічний святковий період минулого року), ігноруючи нерелевантний шум.

Проте нейромережі потребують значно більших обсягів даних та обчислювальних ресурсів порівняно з бустингом, що робить їх впровадження доцільним лише у великих торговельних мережах.

Однією з головних перешкод при використанні класичних рекурентних нейронних мереж (RNN) для прогнозування продажів є **проблема зникаючого градієнта** (Vanishing Gradient Problem).

Суть проблеми полягає в тому, що при навчанні мережі на довгих часових рядах (наприклад, аналіз щоденних продажів за останні 3 роки) використовується алгоритм зворотного поширення помилки через час (BPTT). Оскільки градієнт обчислюється за правилом ланцюгового диференціювання, він є добутком багатьох матриць ваг. Якщо значення ваг невеликі (< 1), то при кожному кроці назад у часі градієнт експоненціально зменшується (рис 1.8):

$$\frac{\partial H_t}{\partial H_k} = \prod_{i=k+1}^{t} \frac{\partial H_i}{\partial H_{i-1}} \propto (W_{hh})^{t-k}$$

Рисунок 1.8. Ланцюгове диференціювання

В результаті, на початкових етапах ряду градієнт стає практично рівним нулю. Це означає, що мережа «забуває» події минулого року (наприклад, минулорічні новорічні розпродажі) і фокусується лише на останніх кількох днях.

Рішення полягає у Long Short-Term Memory (LSTM) (рис 1.9). Архітектура LSTM була спроектована саме для подолання цієї проблеми. Замість простого прихованого стану, LSTM використовує **стан комірки** (cell state), який виконує роль «конвеєрної стрічки», що проходить крізь усю ланцюгову структуру мережі з мінімальними змінами. Регулювання інформації здійснюється через три типи «воріт» (gates):

1. **Forget Gate (Ворота забування):** вирішує, яку частину інформації з минулого стану комірки варто стерти (наприклад, якщо сезонність змінилася з літньої на осінню).
2. **Input Gate (Вхідні ворота):** визначає, які нові дані про поточні продажі варто додати до стану комірки.
3. **Output Gate (Вихідні ворота):** вирішує, яка частина внутрішнього стану буде передана на вихід як прогноз.

Завдяки такій структурі, градієнт може проходити через «стан комірки» без експоненціального згасання, що дозволяє моделі враховувати довгострокові залежності та складні цикли попиту, тривалість яких вимірюється сотнями часових кроків.

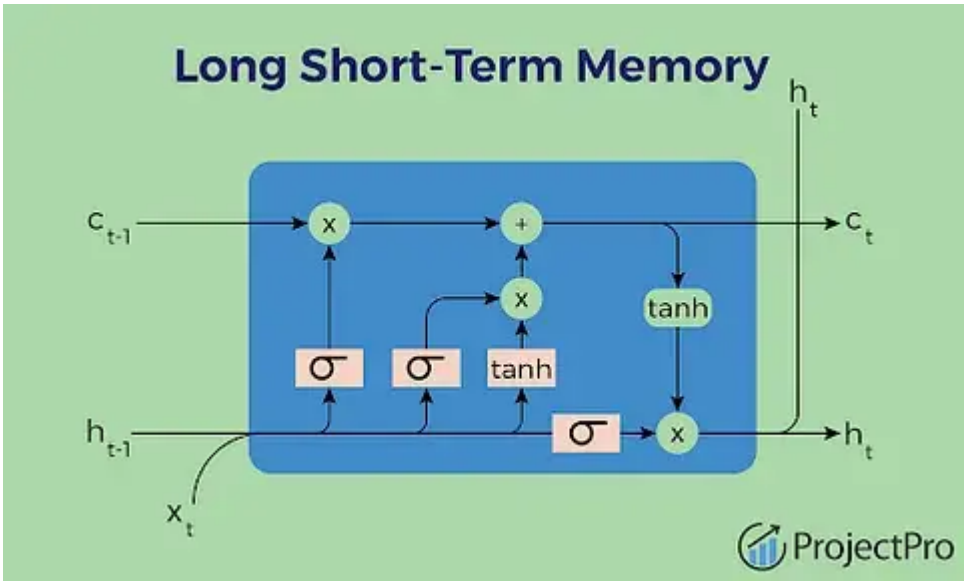


Рисунок 1.9. Внутрішня структура LSTM-блоку: потік даних через ворота [8].

Для дипломного проєкту ми проведемо порівняння за наступними критеріями:

1. **Інтерпретованість:** наскільки легко пояснити бізнесу, чому модель видала саме такий прогноз (тут виграють дерева рішень через значення SHAP).
2. **Швидкість навчання та інерційність:** здатність моделі швидко адаптуватися до різких змін попиту.
3. **Стійкість до пропущених даних:** важливий фактор для реальних баз даних ритейлу.

Порівняльна характеристика парадигм наведена у таблиці 1.1.

Таблиця 1.1. Порівняння методів машинного навчання для задач прогнозування продажів.

Критерій	**Статистичні (ARIMA)**	**Ансамблеві (XGBoost)**	**Глибоке навчання (LSTM)**
Обсяг даних	Малий	Середній	Великий
Врахування екзогенних факторів	Складно	Дуже добре	Відмінно
Складність налаштування	Низька	Середня	Висока

Інтерпретованість	Висока	Середня	Низька («Чорна скринька»)

1.4. Огляд технологічного набору та інструментарію розробки.

Вибір програмного забезпечення для реалізації системи прогнозної аналітики визначається необхідністю обробки великих обсягів даних, проведення складних математичних обчислень та візуалізації результатів для бізнес-користувачів. Мова програмування **Python** залишається безальтернативним лідером у сфері Data Science завдяки своїй універсальності та великій кількості досконалих бібліотек.

Фундаментальні бібліотеки для обробки даних: NumPy та Pandas. Будь-яка система машинного навчання починається з ефективного маніпулювання масивами та таблицями.

1. **NumPy:** є низькорівневим фундаментом, який забезпечує підтримку багатовимірних масивів та високопродуктивних математичних функцій. Головна перевага NumPy — механізм векторизації обчислень, що дозволяє виконувати операції над мільйонами записів продажів без використання повільних циклів Python, використовуючи оптимізований C-код «під капотом».
2. **Pandas:** пропонує високорівневі структури даних, такі як DataFrame (Рис 1.10). У нашому проєкті Pandas є основним інструментом для часового аналізу (Time Series analysis). Вона дозволяє легко виконувати агрегацію продажів за днями, тижнями чи місяцями, обробляти пропущені значення та створювати нові ознаки (rolling windows, lagging features), що є критичним для прогнозування попиту.

Pandas (скорочено позначається: **pd**) – це бібліотека програмного забезпечення з відкритим кодом, створена на базі Python для аналізу та обробки даних. Бібліотека pandas надає структури даних, розроблені спеціально для роботи з табличними наборами даних за допомогою спрощеного API Python.

Pandas є розширенням Python для обробки та маніпулювання табличними даними, що дозволяє ефективно виконувати такі операції, як завантаження, вирівнювання, об'єднання та перетворення наборів даних [19].

datetime	Global_active_power	Global_reactive_power	Voltage	Global_intensity	Sub_metering_1	Sub_metering_2	Sub_metering_3	sub_metering_4
2006-12-16 17:24:00	4.216	0.418	234.84	18.4	0.0	1.0	17.0	52.266670
2006-12-16 17:25:00	5.360	0.436	233.63	23.0	0.0	1.0	16.0	72.333336
2006-12-16 17:26:00	5.374	0.498	233.29	23.0	0.0	2.0	17.0	70.566666
2006-12-16 17:27:00	5.388	0.502	233.74	23.0	0.0	1.0	17.0	71.800000
2006-12-16 17:28:00	3.666	0.528	235.68	15.8	0.0	1.0	17.0	43.100000
2006-12-16 17:29:00	3.520	0.522	235.02	15.0	0.0	2.0	17.0	39.666668
2006-12-16 17:30:00	3.702	0.520	235.09	15.8	0.0	1.0	17.0	43.700000
2006-12-16 17:31:00	3.700	0.520	235.22	15.8	0.0	1.0	17.0	43.666668
2006-12-16 17:32:00	3.668	0.510	233.99	15.8	0.0	1.0	17.0	43.133335
2006-12-16 17:33:00	3.662	0.510	233.86	15.8	0.0	2.0	16.0	43.033333
2006-12-16 17:34:00	4.448	0.498	232.86	19.6	0.0	1.0	17.0	56.133330
2006-12-16 17:35:00	5.412	0.470	232.78	23.2	0.0	1.0	17.0	72.200000

Рисунок 1.10. Візуальний вигляд DataFrame.

Машинне навчання із використанням Scikit-learn та класичні алгоритми. Для реалізації базових моделей, методів крос-валідації та попередньої обробки даних використовується бібліотека **Scikit-learn**.

Вона надає уніфікований інтерфейс (API) для широкого спектра алгоритмів. У межах даної роботи Scikit-learn використовується для:

1. Масштабування ознак (StandardScaler, MinMaxScaler).
2. Розбиття вибірки на навчальну та тестову з урахуванням часової послідовності (TimeSeriesSplit).
3. Оцінки метрик якості (MAE, RMSE, R^2) – завдяки своїй стабільності, Scikit-learn є ідеальним інструментом для створення «baseline» моделей, з якими будуть порівнюватися складніші алгоритми.

Глибоке навчання та тензорні обчислення: PyTorch. Оскільки наша тема передбачає використання сучасних методів (наприклад, LSTM, про які йшлося вище), ми обираємо **PyTorch** як основний фреймворк для нейронних мереж. На відміну від конкурентів (зокрема TensorFlow), PyTorch використовує концепцію **динамічного графа обчислень**, також нижче наведено порівняння бібліотек (**табл 1.2**). Це дозволяє змінювати архітектуру мережі майже миттєво, що значно спрощує процес розробки та налагодження складних архітектур для

прогнозування часових рядів. У 2026 році PyTorch має найкращу підтримку прискорення на GPU (NVIDIA CUDA) та NPU (Neural Processing Units), що дозволяє навчати моделі на мільйонах транзакцій за лічені хвилини.

PyTorch — це фреймворк для машинного навчання з відкритим кодом, який прискорює перехід від створення дослідницьких прототипів до впровадження в виробництво. Створений для забезпечення максимальної гнучкості та швидкості, PyTorch підтримує динамічні обчислювальні графіки, що дозволяє дослідникам і розробникам швидко та інтуїтивно вдосконалювати свої рішення. Його «пітонівська» архітектура та глибока інтеграція з нативними інструментами Python роблять його доступною та потужною платформою для створення та навчання моделей глибокого навчання у великих масштабах **[11].**

Таблиця 1.2. Порівняння бібліотек глибокого навчання

Критерій	**PyTorch**	**TensorFlow/Keras**	**Scikit-learn**
Тип моделей	Deep Learning (Нейромережі)	Deep Learning	Класичний ML
Граф обчислень	Динамічний (гнучкий)	Статичний/Гібридний	Відсутній
Поріг входження	Середній	Середній	Низький
Використання в R&D	Домінуюче (90%+)	Промислове (Legacy)	Універсальне

Візуалізація та інтерпретація результатів. Для перетворення цифр у зрозумілі бізнес-звіти використовуються бібліотеки **Matplotlib** та **Seaborn**.

1. **Matplotlib** дозволяє будувати низькорівневі графіки (наприклад, графік залишків або функцію втрат) (Рис 1.11).
2. **Seaborn** базується на Matplotlib і надає високорівневий інтерфейс для візуалізації складних статистичних залежностей, таких як теплові карти кореляції (Heatmaps) між ціною, сезоном та обсягом продажів.

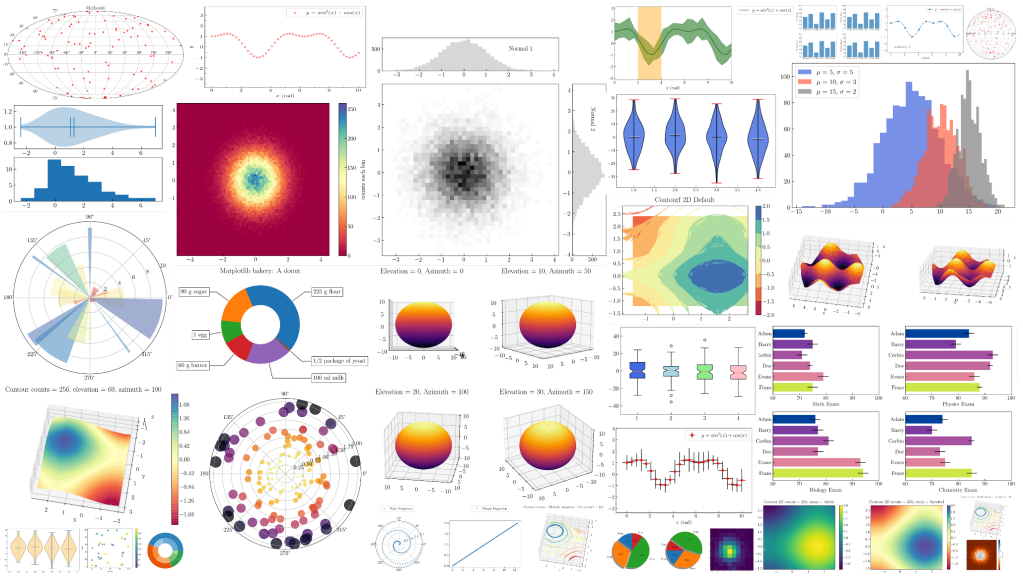


Рисунок 1.11. Можливі візуалізації за допомогою Matplotlib

Інфраструктура та середовище розробки. Робота над проєктом ведеться в середовищі **Visual Studio Code** з використанням інтерактивних ноутбуків **Jupyter**. **Jupyter** дозволяє виконувати окремі частини коду без зміни чи запускання всього робочого процесу, тому це дає змогу легко маніпулювати даними без зміни основного набору даних, особливо якщо розробник не впевнений у якійсь частині коду (рис 1.12). Це дозволяє розділити процес розробки на ітерації: завантаження даних, експерименти з моделями та візуалізація відбуваються в одному документі. Для контролю версій коду та моделей використовується **Git**, що є стандартом індустрії для забезпечення відтворюваності досліджень.

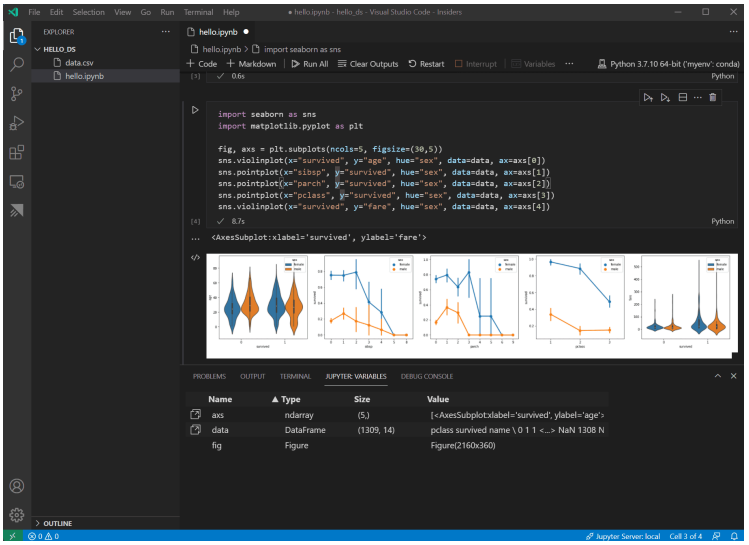


Рисунок 1.12. Робочий процес розробки ML-системи в Jupyter/VS Code

Підбиваючи підсумки першого розділу, то було проведено комплексний теоретичний та технологічний аналіз стану сучасної прогнозної аналітики у сфері продажів. На основі вивченого матеріалу можна зробити наступні висновки.

По-перше, встановлено, що сучасна архітектура систем Machine Learning еволюціонувала від простих функцій програмування до складних екосистем. Простими словами, сьогодні недостатньо просто «запустити код»; необхідно побудувати надійне робоче середовище (pipeline), яким дані безперервно течуть від касового апарату до інтелектуального сховища. Використання таких інструментів, як Feature Store та Data Lakehouse, дозволяє бізнесу уникати помилок у даних та працювати з інформацією в режимі реального часу.

По-друге, математична формалізація задачі показала, що оптимізація продажів – це завжди пошук балансу. У бізнесі помилка у менший бік (коли товару не вистачило) та помилка у більший бік (коли товар лежить у надлишку) мають різну «ціну». Використання асиметричних функцій втрат, зокрема Pinball Loss, дозволяє навчити програму бути «розумнішою»: вона не просто вгадує середнє число, а мінімізує реальні грошові збитки підприємства.

По-третє, порівняльний аналіз алгоритмів підтвердив, що для табличних даних продажів найефективнішим інструментом є градієнтний бустинг (зокрема CatBoost), тоді як нейронні мережі (LSTM) доцільно використовувати для виявлення складних довгострокових закономірностей. Завдяки механізмам регуляризації (L1 та L2), ці моделі здатні самостійно відсіювати неважливі чинники, фокусуючись лише на тому, що дійсно впливає на попит.

І у кінці було обґрунтовано технологічний набір на базі мови Python та бібліотек Pandas, Scikit-learn та PyTorch забезпечує необхідну гнучкість та потужність для реалізації проєкту. Таким чином, сформований теоретичний фундамент дозволяє перейти до практичної частини роботи – проектування конкретних алгоритмів обробки даних та розробки програмного забезпечення.

Розділ 2. Математичне моделювання та методологія підготовки даних для інтелектуального аналізу продажів

2.1. Методи інженерії ознак та попередньої обробки часових рядів у ритейлі.

Ефективність функціонування будь-якої інтелектуальної системи, що базується на парадигмі машинного навчання, фундаментально залежить від якості та релевантності вхідних даних. У контексті прогнозування комерційних показників, «сирі» дані транзакцій, вилучені з баз даних SQL або ERP-систем, зазвичай мають низький рівень інформативності для складних алгоритмів без відповідної трансформації. Процес інженерії ознак (Feature Engineering) являє собою науково обґрунтовану методологію перетворення первинних спостережень у змінні, які максимально повно дескриптують латентні закономірності споживчого попиту.

Feature Engineering – це процес перетворення необроблених даних на релевантну інформацію для використання моделями машинного навчання. Іншими словами, інженерія ознак – це процес створення ознак для прогнозних моделей. Ознака, яку також називають виміром, – це вхідна змінна, що використовується для формування прогнозів моделі. Оскільки ефективність моделі значною мірою залежить від якості даних, що використовуються під час навчання, інженерія ознак є надзвичайно важливим методом попередньої обробки, який передбачає відбір найбільш релевантних аспектів необроблених навчальних даних як для конкретного завдання прогнозування, так і для типу моделі, що розглядається [18].

Первинна обробка, очищення та нормалізація даних. Першим етапом методології є ідентифікація та усунення аномалій, що можуть спотворити процес навчання моделі. У сфері роздрібної торгівлі викиди (outliers) мають двояку природу. З одного боку, це можуть бути технічні помилки (дублювання записів, збої при передачі даних), з іншого — реальні екстремальні події (панічні закупівлі, масові акції) (рис 2.1). Для їх виявлення застосовується метод інтерквартильного розмаху (IQR), який зображено у формулі нижче:

$$IQR = Q_3 - Q_1$$

де Q_1 та Q_3 – перший та третій квартилі відповідно. Спостереження, що виходять за межі $[Q_1 - 1.5 * IQR;\ Q_3 + 1.5 * IQR]$, підлягають експертному аналізу або автоматичному вирахуванню медіанними значеннями.

Важливим аспектом є також обробка пропущених значень. В аналізі часових рядів неприпустимо використовувати просте вилучення рядків, оскільки це порушує хронологічну цілісність. Замість цього застосовується метод лінійної інтерполяції або техніка «Forward Fill» (заповнення останнім відомим значенням), що дозволяє зберегти динаміку ряду без внесення суттєвих стохастичних спотворень.

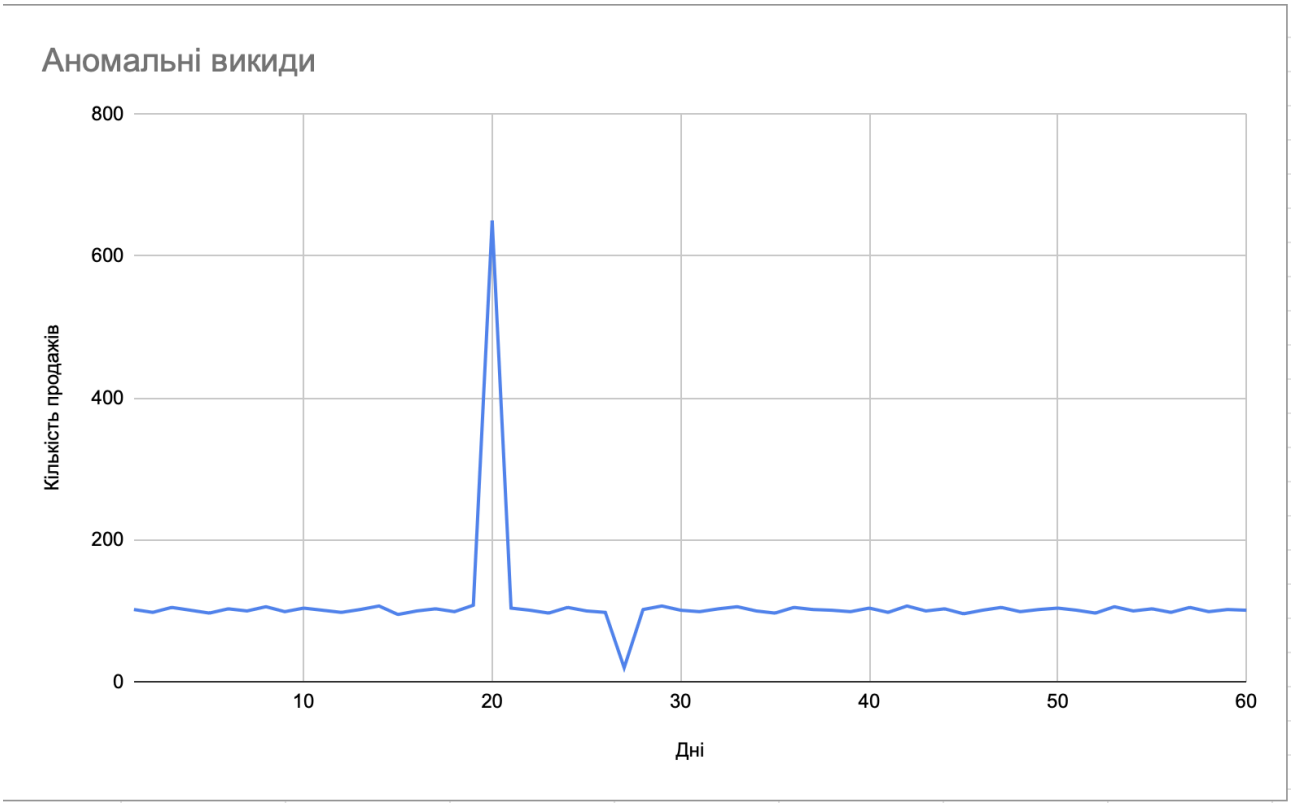


Рисунок 2.1. Візуалізація аномальних викидів у часовому ряді продажів

Формування ретроспективних (лагових) ознак. Ключовою гіпотезою при прогнозуванні продажів є наявність автокореляції – залежності поточного попиту від його значень у минулому. Лагові змінні (lag features) дозволяють моделі враховувати інерційність ринку. Математично лаг порядку **k** для часового ряду y_t визначається як формула наведена нижче:

$$y_t^{'} = y_t - k$$

Для задач роздрібного торговця критично важливим є вибір лагів, що відповідають циклам споживчої поведінки. Типово використовуються лаги $k \in \{1, 2, 3, 4, 14, 30, 365\}$, де лаг 7 відображає тижневу циклічність, а лаг 365 — річну сезонність. Використання широкого спектра лагових ознак дозволяє алгоритму ідентифікувати як короткострокові коливання, так і глобальні тренди (рис 2.2).

Row	Sales	Lag_1
1	120	
2	135	120
3	128	135
4	140	128
5	150	140
6	145	150
7	160	145
8	155	160
9	170	155
10	165	170

Рисунок. 2.2. Схема формування лагових змінних для навчання ML-моделі

Методологія ковзного вікна та статистична агрегація. На відміну від статичних лагів, ознаки ковзного вікна (rolling windows) надають інформацію про динаміку змін у певному часовому інтервалі. Це дозволяє моделі оцінити «імпульс» (momentum) продажів. Для кожного моменту часу **t** обчислюються статистичні показники для вікна розміром **W**:

1. **Середнє значення (Moving Average):** згладжує випадковий шум та виділяє локальний тренд.
2. **Стандартне відхилення (Rolling Std):** характеризує волатильність попиту та ризики різких змін.
3. **Екстремальні значення (Min/Max):** визначають межі стійкості ринкової ситуації.

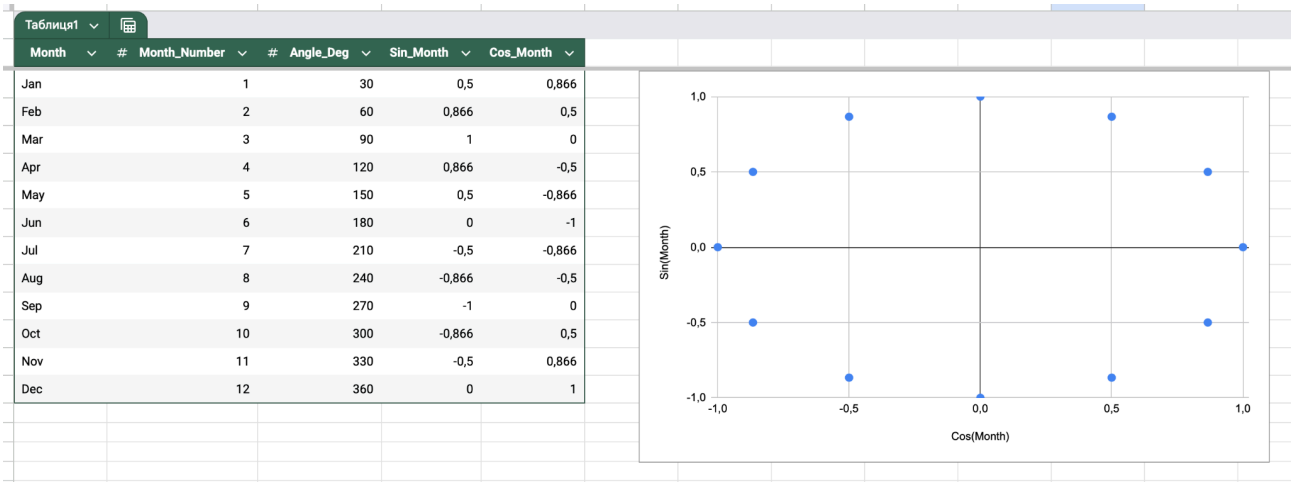
Використання декількох вікон різної тривалості (наприклад, 7 та 28 днів) дозволяє моделі одночасно бачити як оперативну зміну попиту, так і стабільний базовий рівень продажів.

Детерміновані календарні ознаки та циклічне кодування часу. Попит у сфері продажів має яскраво виражену детерміновану складову, пов'язану з календарем. Для формалізації цих факторів використовуються бінарні ознаки (є день вихідним, чи є він святковим) та порядкові змінні (номер місяця, номер тижня року).

Проте використання порядкових чисел (наприклад, 1-31 для днів місяця) має математичний недолік: модель сприймає 31-ше та 1-ше число як максимально віддалені, хоча фактично вони є сусідніми і це може негативно впливати на результати наших ан. Для вирішення цієї суперечності застосовується тригонометричне перетворення (Cyclic Encoding):

$$x_{sin} = sin(\frac{2\pi * x}{max(x)}); \qquad x_{cos} = cos(\frac{2\pi * x}{max(x)});$$

Такий підхід дозволяє перевести часові координати у двовимірний простір кола, де кінець циклу плавно переходить у його початок, що значно покращує збіжність нейронних мереж та деревних моделей (рис 2.3).



Таблиця1

Month	# Month_Number	# Angle_Deg	Sin_Month	Cos_Month
Jan	1	30	0,5	0,866
Feb	2	60	0,866	0,5
Mar	3	90	1	0
Apr	4	120	0,866	-0,5
May	5	150	0,5	-0,866
Jun	6	180	0	-1
Jul	7	210	-0,5	-0,866
Aug	8	240	-0,866	-0,5
Sep	9	270	-1	0
Oct	10	300	-0,866	0,5
Nov	11	330	-0,5	0,866
Dec	12	360	0	1

Рисунок 2.3. Візуалізація циклічного кодування часових ознак за допомогою тригонометричних функцій

Категоріальне кодування та Target Encoding. Оскільки дані про продажі містять велику кількість категоріальних змінних (артикули товарів, ідентифікатори магазинів, географічні зони), виникає задача їх числового

представлення. Традиційний метод One-Hot Encoding призводить до ускладнення розмірності при великій кількості категорій. Більш ефективним у задачах прогнозування є **Target Encoding** – заміна назви категорії середнім значенням цільової змінної (обсягу продажів) для цієї категорії в минулому. Це дозволяє безпосередньо інтегрувати інформацію про популярність конкретного товару або магазину в ознаку, проте потребує обережного застосування з використанням регуляризації для уникнення витоку даних (data leakage) з майбутнього в минуле.

Target Encoding, також відоме як кодування за середнім значенням, полягає у заміні кожної категорії категоріальної змінної на середнє значення (або інший узагальнений статистичний показник) цільової змінної для цієї категорії. Цей метод використовує взаємозв'язок між категоріальною ознакою та цільовою змінною, надаючи можливість кодувати категоріальну інформацію безпосередньо у числову форму. Ця техніка є особливо корисною для завдань класифікації [15].

2.2. Стратегії валідації та метрики оцінки якості прогнозних моделей.

Оцінка якості прогнозних моделей у сфері роздрібної торгівлі суттєво відрізняється від класичних задач класифікації чи регресії на статичних даних. Ключовою особливістю є часова залежність спостережень: дані за майбутні періоди не можуть бути використані для навчання моделей, що прогнозують минуле. Це вимагає застосування специфічних стратегій валідації та набору метрик, які б відображали не лише математичну точність, а й реальну бізнес-ефективність прийнятих рішень.

Методологія валідації часових рядів: Time Series Cross-Validation. У класичному машинному навчанні часто використовується K-кратна крос-валідація, де дані перемішуються випадковим чином. Для аналізу продажів такий підхід є неприпустимим, оскільки він призводить до «витоку даних» (data leakage) – ситуації, коли інформація з майбутнього допомагає моделі вгадувати минуле, що робить результати тестування нереалістично оптимістичними.

Для вирішення цієї проблеми застосовується стратегія Time Series Split (або Walk-Forward Validation). Її суть полягає у послідовному розширенні навчальної вибірки та тестуванні моделі на наступному за нею часовому кроці. Математично цей процес можна описати як послідовність ітерацій ***i***, де для кожного кроку:

1. Навчальна вибірка: $\{y_1, y_2, \dots, y_t\}$.
2. Тестова вибірка: $\{y_{t+1}, \dots, y_n\}$.

На кожній наступній ітерації точка відсікання ***t*** зміщується вперед на величину кроку ***n***. Такий підхід дозволяє імітувати реальний процес експлуатації системи, коли модель регулярно перенавчається на нових даних, що надходять з ринку. Зазвичай при такому підході дані діляться, що 80% припадає на навчальну вибірку, а інші 20% це тестова вибірка, де 100% це весь набір даних.

Математичне обґрунтування метрик точності. Для кількісної оцінки помилок прогнозування у даній роботі використовується комплексний набір метрик, кожна з яких висвітлює певну сторону якості моделі.

1. Середня абсолютна помилка (Mean Absolute Error, *MAE*). Показує середню величину відхилення прогнозу від реальних продажів у натуральних одиницях (наприклад, у штуках товару).

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - y'_i|$$

Перевагою ***MAE*** є її стійкість до викидів та легкість інтерпретації для менеджменту: вона прямо відповідає на питання «на скільки одиниць товару ми в середньому помиляємося».

Середньоквадратична похибка (MSE) - це міра похибки алгоритмів прогнозування. Ця статистика кількісно виражає середньоквадратичну дисперсію між спостережуваними та передбачуваними значеннями. Коли в моделі немає помилок, MSE дорівнює 0. Цінність моделі зростає пропорційно

до ступеня помилки, яку вона містить. Середньоквадратичну похибку часто називають MSD - середньоквадратичне відхилення.

2. Корінь із середньоквадратичної помилки (Root Mean Squared Error, RMSE): На відміну від MAE, ця метрика сильніше «карає» модель за великі відхилення (рис 2.4).

$$MAE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - y_i^{'})^2}$$

Це критично важливо для управління запасами, оскільки одинична велика помилка (наприклад, недопрогноз попиту в 1000 одиниць) є набагато небезпечнішою для бізнесу, ніж десять дрібних помилок по 100 одиниць.

RMSE являє собою квадратний корінь із середньоквадратичної різниці між прогнозованими та спостережуваними результатами. Це показник, який переважно використовується в регресійному аналізі та прогнозуванні, де точність має велике значення. Чим нижче RMSE, тим краща здатність моделі до точного прогнозування. І навпаки, вищий показник RMSE означає більшу розбіжність між прогнозованими та фактичними результатами [13].

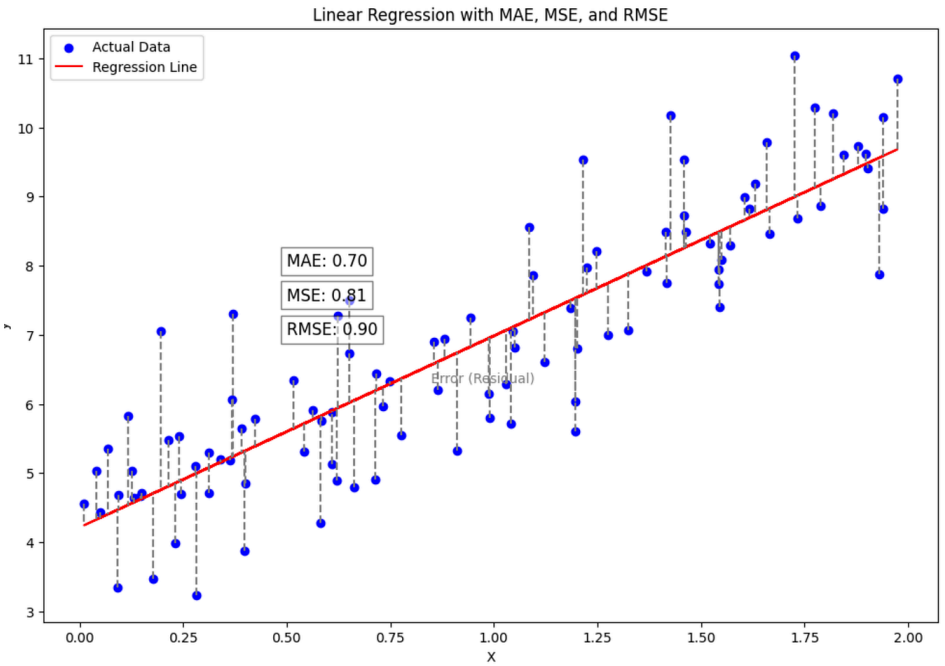


Рисунок 2.4. Порівняльна діаграма чутливості метрик MAE та RMSE до аномальних помилок

3. Зважена середня абсолютна відсоткова помилка (Weighted Average Percentage Error, WAPE): У роздрібній торгівлі часто виникає проблема: як порівняти точність прогнозу для дешевого товару, що продається тисячами, та дорогого, що продається одиницями. WAPE вирішує цю проблему, нормуючи сумарну абсолютну помилку на сумарний фактичний попит:

$$WAPE = \frac{\Sigma|y_i - y_i^{'}|}{\Sigma y_i}$$

Саме WAPE у теперішній час вважається галузевим стандартом, оскільки вона дозволяє оцінити якість роботи системи на всьому асортименті товарів одночасно.

WAPE – це показник похибки регресії/прогнозування, який вимірює сумарну абсолютну похибку відносно сумарних фактичних значень. Його часто називають «об'ємно-зваженим MAPE», оскільки він зважує похибки з урахуванням масштабу фактичних значень [16].

Коефіцієнт детермінації та аналіз залишків.

Що таке R-squared? R-squared (R^2) — це число, яке показує, наскільки добре незалежна змінна (або незалежні змінні) у статистичній моделі пояснює варіацію залежної змінної. Його значення коливається від 0 до 1, де 1 означає ідеальне відповідність моделі даним [12].

Окрім метрик помилок, важливим є показник R^2 (R-squared), який демонструє частку дисперсії цільової змінної, що пояснюється моделлю.

$$R^2 = 1 - \frac{\Sigma(y_i - y_i^{'})^2}{\Sigma(y_i - y^{'})^2}$$

Значення R^2 = 1 свідчить про ідеальний прогноз, а R^2 = 0 означає, що модель працює не краще, ніж просте середнє арифметичне (рис 2.5). Але варто зауважити, що якщо модель показує 1, то це може призвести до того, що модель дуже чудово передбачає дані на тренуванні, що призводить до overfitted. Як

тільки модель почне проходити тестовий набір даних, то результат буде набагато гірший від очікуваного.

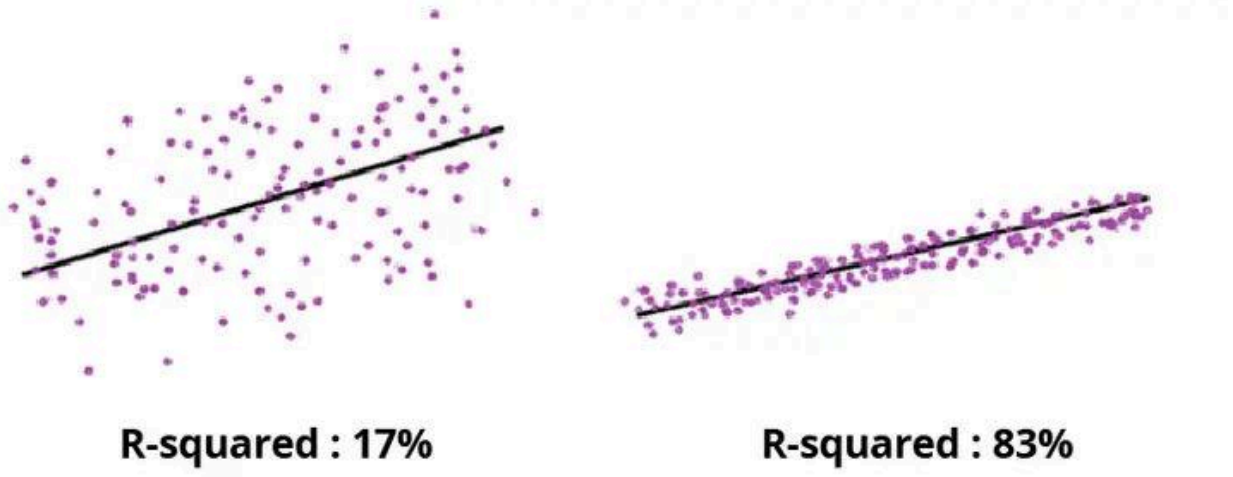


Рисунок 2.5. R-squared візуальне відображення.

Проте висока точність на папері не завжди гарантує відсутність помилок у структурі. Для фінальної валідації проводиться аналіз залишків (residuals analysis) – різниці між фактом і прогнозом. В ідеальній моделі залишки повинні бути «білим шумом»:

1. Мати середнє значення, близьке до нуля (відсутність систематичного зміщення).
2. Бути гомоскедастичними (мати сталу дисперсію у часі).
3. Не мати автокореляції (помилка сьогодні не повинна залежати від помилки вчора).

Якщо в залишках спостерігається чітка структура (наприклад, циклічність), це сигнал того, що модель не врахувала певний важливий фактор (сезонність або акції), і потребує доопрацювання на етапі інженерії ознак.

2.3. Математичне обґрунтування алгоритмів ансамблевого навчання та градієнтного спуску.

Реалізація прогнозної системи у сфері продажів потребує використання алгоритмів, здатних опрацьовувати складні нелінійні залежності. Найбільш ефективним підходом у теперішньому часі є використання градієнтного

бустингу на деревах рішень (Gradient Boosting Decision Trees, GBDT). Математична суть цього методу полягає у послідовній мінімізації функції втрат шляхом додавання нових моделей, кожна з яких виправляє помилки попередніх.

Математична постановка задачі оптимізації. В основі будь-якого алгоритму машинного навчання лежить задача мінімізації емпіричного ризику. Нехай ми маємо навчальну вибірку $\{(x_i, y_i)\}_{i=1}^{n}$. Наша мета – знайти таку функцію ***F(x)***, яка б мінімізувала очікуване значення функції втрат ***L(y, F(x))***:

$$arg\ min_F \sum_{i=1}^{n} L(y_i, F(x_i))$$

Для задач регресії (прогнозування кількості продажів) найчастіше використовується квадратична функція втрат $L(y, F) = \frac{1}{2}(y - F)^2$ або функція абсолютного відхилення.

Механізм градієнтного спуску. Градієнтний спуск - це алгоритм, який встановлює значення параметрів функції, що мінімізують функцію вартості. Алгоритм градієнтного спуску в машинному навчанні найбільш корисний, коли параметри не можуть бути визначені аналітично (наприклад, за допомогою лінійної алгебри) і повинні бути знайдені за допомогою процедури оптимізації [7].

Процес навчання відбувається ітераційно за допомогою алгоритму градієнтного спуску. На кожному кроці ми обчислюємо антиградієнт функції втрат щодо поточного прогнозу моделі. Цей антиградієнт вказує напрямок найшвидшого спуску до мінімуму помилки (рис 2.6). Вектор оновлення параметрів θ визначається як:

$$\theta_{next} = \theta_{prev} - \eta * \nabla L(\theta_{prev})$$

де η — швидкість навчання (learning rate), що визначає розмір кроку в напрямку мінімуму. У контексті бустингу роль параметрів θ виконують прогнози ансамблю моделей.

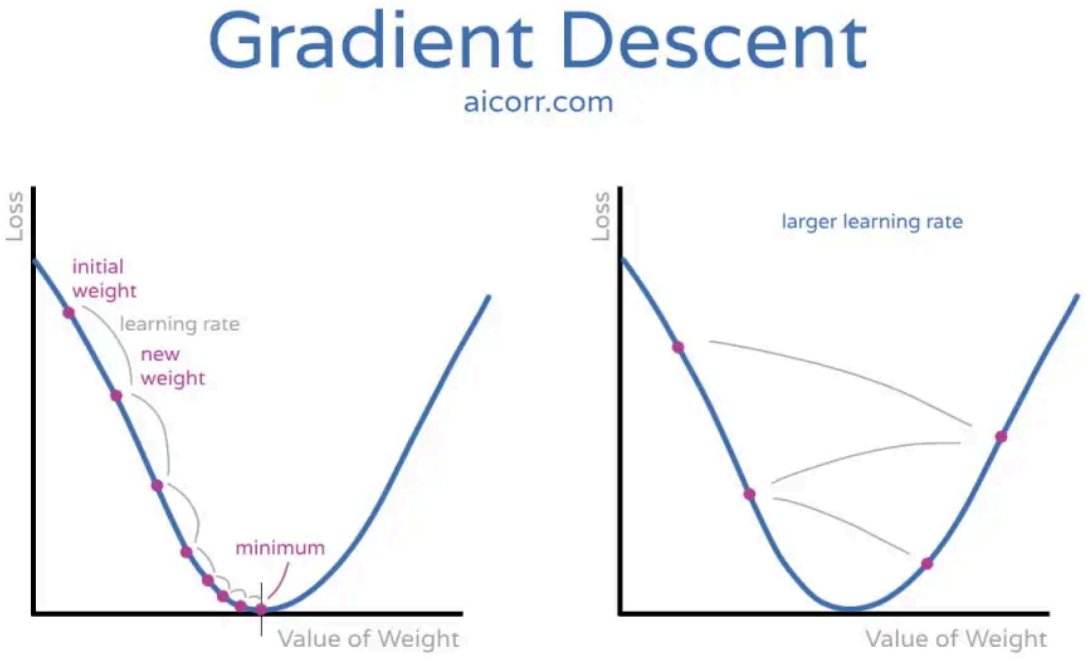


Рисунок 2.6. Візуалізація процесу градієнтного спуску на поверхні функції втрат

Алгоритм градієнтного бустингу (GBM).

Дерево рішень – це модель ML, яка базується на ітеративній постановці запитань для розділення даних і отримання рішення. Це найприродніший підхід для прийняття рішення про класифікацію або маркування об'єкта. Воно також має вигляд перевернутого дерева з виступаючими гілками, звідси і назва. Дерево рішень - це деревоподібна структура, схожа на блок-схему, в якій кожен вузол представляє атрибут набору даних, кожна гілка - вибір, а кожен вузол листя - результат. Кореневий вузол - це самий верхній вузол у дереві рішень. Він вчиться ділити на основі значення ознаки. Він рекурсивно розбиває дерево, також відомий як рекурсивне розбиття [6].

На відміну від методу випадкового лісу (Random Forest), де дерева будуються незалежно, у градієнтному бустингу моделі створюються послідовно. Кожна нова модель $h_m(x)$ навчається прогнозувати залишки (residuals) попередньої композиції моделей $F_{m-1}(x)$. Формально модель на ітерації $\boldsymbol{m}$ виглядає так:

$$F_m(x) = F_{m-1}(x) + \eta * h_m(x)$$

де $h_m(x)$ — слабка модель (дерево рішень), яка найкраще апроксимує значення антиградієнта $-\frac{\partial L(y,F_{m-1}(x))}{\partial F_{m-1}(x)}$. Таким чином, з кожною новою ітерацією загальна модель ***F(x)*** стає все точнішою, поступово зменшуючи сумарну помилку прогнозування.

Логіка побудови дерев рішень. Як «слабкі» моделі $h_m(x)$ використовуються бінарні дерева рішень. Процес побудови дерева базується на рекурсивному розбитті простору ознак на підмножини (вузли), які є максимально однорідними щодо цільової змінної (рис 2.7).

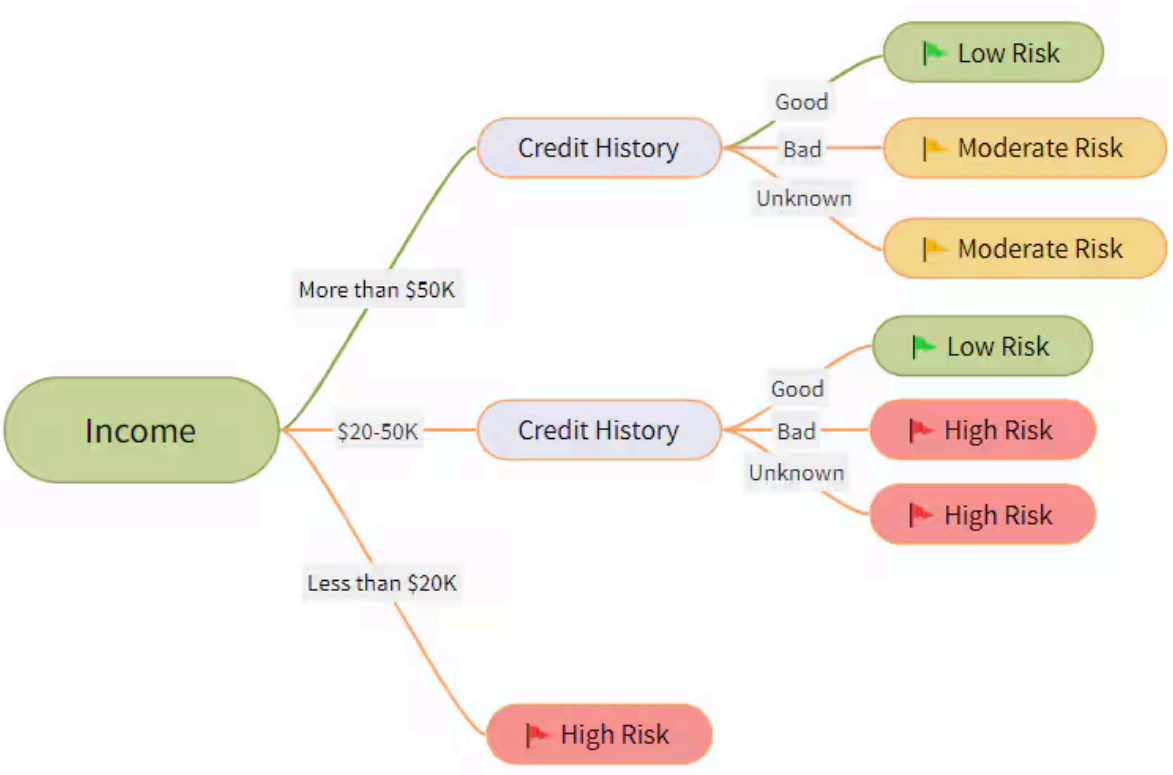


Рисунок 2.7. Приклад ієрархічної структури дерева рішень у задачі прогнозування попиту

Для вибору оптимального розбиття (split) використовується критерій зменшення дисперсії (Variance Reduction) або приріст інформації (Information Gain). Для вузла ***T*** критерій оцінюється як:

$$I(T) = \sum_{x \in T} (y_i - y'r)^2$$

Алгоритм шукає таку ознаку та поріг, які мінімізують сумарну дисперсію у лівому та правому піддеревах. Це дозволяє системі автоматично виділяти найбільш значущі фактори впливу на продажі, такі як цінові пороги або дні тижня.

Особливості сучасних реалізацій: CatBoost та XGBoost. У даній дипломній роботі особлива увага приділяється алгоритму **CatBoost**, розробленому для ефективної роботи з категоріальними даними без необхідності попереднього One-Hot кодування (рис 2.8).

CatBoost – це високопродуктивна бібліотека з відкритим кодом для градієнтного бустінгу на деревах рішень, яку можна використовувати для завдань класифікації, регресії та ранжування. CatBoost використовує поєднання впорядкованого бустінгу, випадкових перестановок та оптимізації на основі градієнта, щоб забезпечити високу продуктивність при роботі з великими та складними наборами даних, що містять категоріальні ознаки [3].

Математичною інновацією CatBoost є алгоритм **Ordered Boosting**, який бореться з ефектом «зміщення градієнта» (prediction shift), що часто виникає у класичних реалізаціях бустингу на малих вибірках. Також використовується механізм регуляризації (L1/L2), інтегрований безпосередньо в структуру побудови дерев, що обмежує складність (глибину) дерев і запобігає перенавчанню на випадкових шумах ринкових даних.

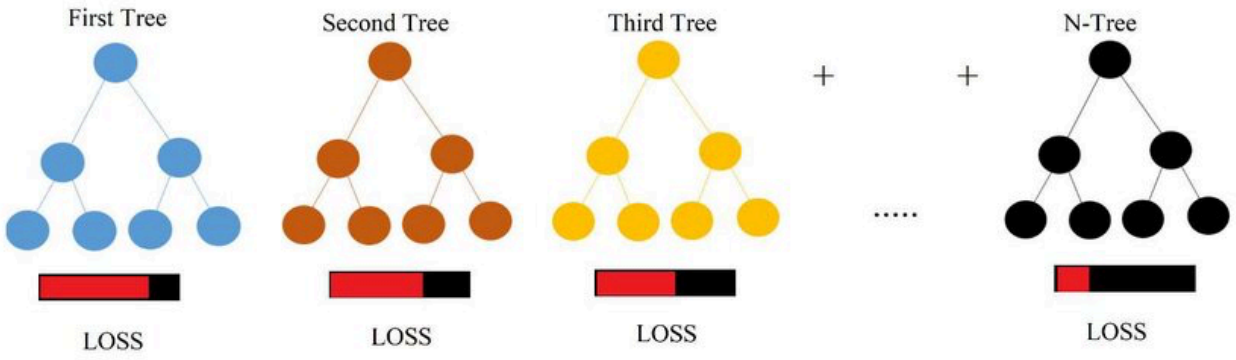


Рисунок 2.8. CatBoost на практиці.

У другому розділі було сформовано методологічну базу дослідження, що охоплює всі етапи підготовки та валідації даних, а також математичне обґрунтування вибору алгоритмів навчання.

По-перше, розроблено стратегію інженерії ознак, яка дозволяє трансформувати сирі дані транзакцій у набір високо-інформативних змінних. Використання лагових показників, ковзних вікон та циклічного кодування часу забезпечує моделі здатність сприймати як миттєві зміни ринку, так і глибоку сезонність продажів.

По-друге, обґрунтовано використання специфічних методів валідації часових рядів. Відмова від класичної крос-валідації на користь методу Walk-Forward Validation дозволяє отримати об’єктивну оцінку прогнозних можливостей системи у реальних умовах експлуатації. Впровадження метрики WAPE забезпечує бізнес-орієнтований контроль якості на всьому асортименті товарів.

По-третє, детальний математичний аналіз алгоритмів градієнтного бустингу підтвердив їх спроможність вирішувати задачі оптимізації у просторах високої розмірності. Поєднання потужного апарату градієнтного спуску з гнучкістю дерев рішень створює надійну основу для розробки програмного продукту, що має високу точність та стійкість до перенавчання.

Сформована методологія є ключовою інформацією до третього розділу, присвяченого безпосередньому проєктуванню архітектури та програмній реалізації системи.

Розділ 3. Проєктування архітектури та розробка інтелектуальної системи аналізу та оптимізації продажів

3.1. Обґрунтування архітектурних рішень та функціональних вимог до системи.

Проєктування інтелектуальної системи прогнозування продажів вимагає системного підходу, що поєднує принципи класичної програмної інженерії та специфіку життєвого циклу моделей машинного навчання. На даному етапі розробки здійснюється перехід від теоретичних концепцій до технічної реалізації, що передбачає чітке визначення функціональних меж системи та вибір оптимальної інфраструктурної стратегії.

Формалізація функціональних та нефункціональних вимог. Для забезпечення ефективності бізнес-процесів у сфері управління товарними запасами, розроблювана система повинна відповідати ряду критичних вимог.

Функціональні вимоги (Functional Requirements):

1. Модуль імпорту даних: автоматизоване вилучення транзакційних даних з гетерогенних джерел (CSV-файли, SQL-бази даних).
2. Модуль аналітичної обробки: реалізація конвеєра Feature Engineering, включаючи обчислення лагових змінних та ковзних вікон.
3. Модуль предиктивного аналізу: генерація прогнозів попиту на заданий часовий горизонт (***N*** днів/тижнів) за допомогою ансамблевих методів.
4. Модуль експорту та візуалізації: формування звітів у форматі інтерактивних графіків та збереження результатів у структуровані файли для подальшого використання ERP-системами.

Нефункціональні вимоги (Non-functional Requirements):

1. Масштабованість (Scalability) – здатність системи обробляти зростаючі масиви даних (Big Data) без критичної втрати продуктивності.
2. Модульність (Modularity) – можливість заміни окремих алгоритмів навчання без необхідності повної перебудови архітектури.
3. Надійність (Reliability) – коректна обробка пропущених значень та аномалій у вхідних даних без зупинки обчислювального процесу.

Порівняльний аналіз архітектурних стратегій розгортання. Важливим аспектом проєктування є вибір між локальною (On-premise) та хмарною (Cloud-native) архітектурами (табл 3.1). Вибір стратегії безпосередньо впливає на вартість розробки, швидкість імплементації та безпеку корпоративних даних.

Таблиця 3.1. Порівняльна характеристика архітектурних підходів до впровадження ML-системи.

Параметр порівняння	**Локальна архітектура (On-premise)**	**Хмарна архітектура (AWS/GCP/Azure)**
Капітальні витрати	Високі (закупівля серверного обладнання)	Низькі (оплата за фактичне використання)
Контроль над даними	Повний (дані не залишають контуру мережі)	Частковий (залежність від політик провайдера)
Швидкість розгортання	Низька (потребує налаштування середовища)	Висока (доступ до готових сервісів SageMaker/Vertex AI)
Масштабованість	Обмежена потужністю обладнання	Майже необмежена (Auto-scaling)
Залежність від зв'язку	Відсутня (може працювати офлайн)	Критична (вимагає стабільного Internet-з'єднання)

Підбиваючи підсумки, то можна сказати, що для даної дипломної роботи обрано локально-орієнтовану архітектуру (On-premise) з використанням віртуальних середовищ розробки. Дане рішення обумовлене необхідністю забезпечення конфіденційності комерційних даних, відсутністю потреби у витратах на хмарні сервіси на етапі прототипування, а також можливістю повного контролю над процесом навчання моделей на локальних обчислювальних потужностях.

Багаторівнева логічна структура системи. Система проєктується за принципом розподілу відповідальності, що реалізується через трирівневу архітектуру (рис 3.1):

1. Рівень даних (Data Access Layer): забезпечує взаємодію з джерелами інформації, валідацію вхідних потоків та їх первинне структурування.
2. Рівень бізнес-логіки (ML Logic Layer): центральний вузол системи, де зосереджені алгоритми попередньої обробки, навчання моделей (Model Training) та генерації прогнозів (Inference).
3. Рівень представлення (Presentation Layer): інтерфейс взаємодії, що відповідає за трансформацію числових масивів у зрозумілі для кінцевого користувача візуальні форми (звіти, дашборди).

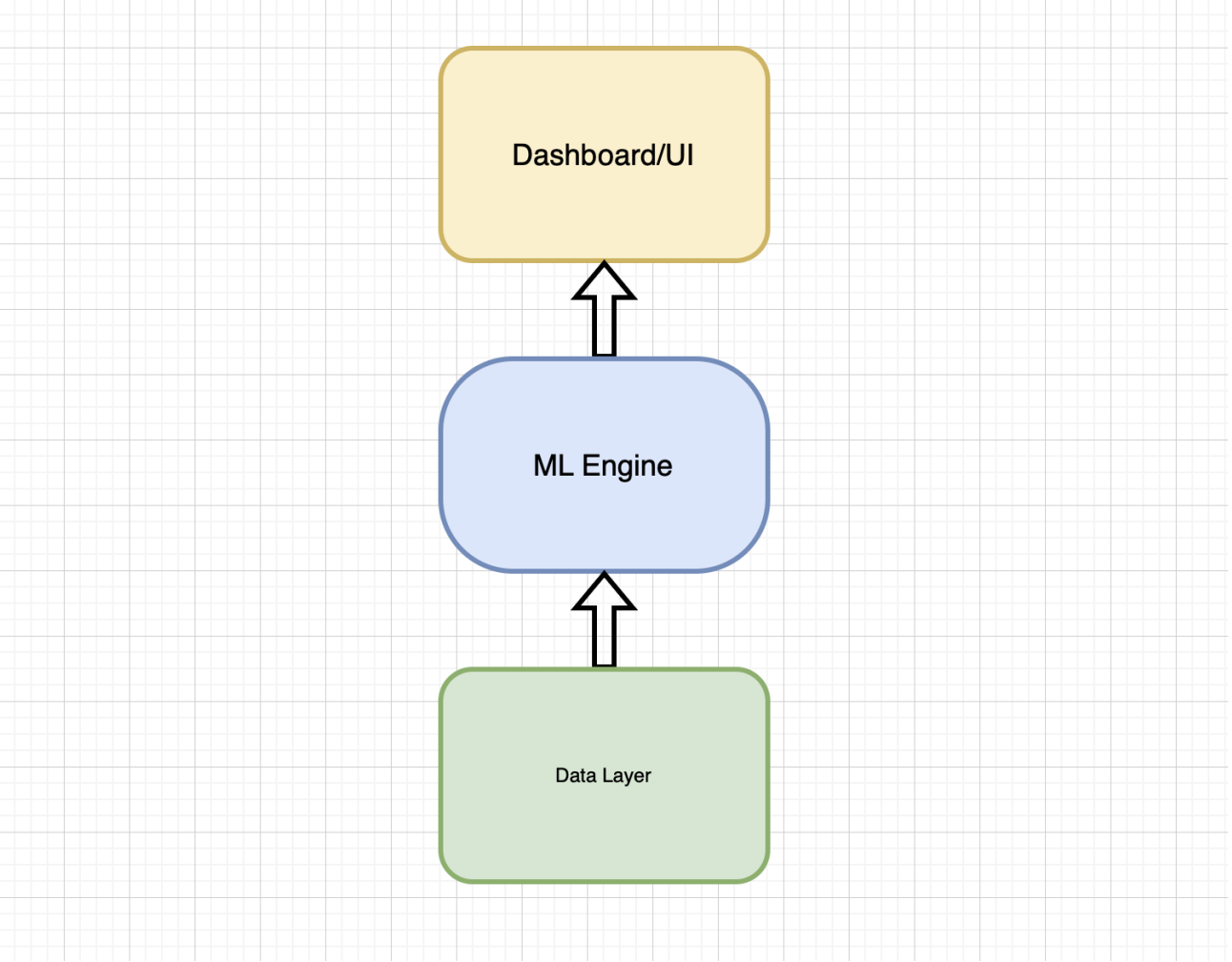


Рисунок. 3.1. Концептуальна схема трирівневої архітектури системи

Аналіз прецедентів використання. Взаємодія із системою передбачає наявність двох основних ролей (акторів) (рис 3.2):

1. Адміністратор/Data Scientist: відповідає за конфігурацію гіперпараметрів моделей, моніторинг точності прогнозів та оновлення тренувальних наборів даних.

Спеціаліст з аналізу даних керує дослідницькими проектами, спрямованими на вилучення цінної інформації з великих масивів даних, і володіє знаннями в галузі технологій, математики, бізнесу та комунікацій. Організації використовують цю інформацію для прийняття більш обґрунтованих рішень, вирішення складних проблем та вдосконалення своєї діяльності. Виявляючи практичні висновки, приховані у великих масивах даних, спеціаліст з аналізу даних може суттєво підвищити здатність своєї компанії досягати поставлених цілей. Саме тому фахівці з аналізу даних користуються великим попитом і навіть вважаються «зірками» у світі бізнесу [17].

2. Кінцевий користувач (Менеджер з продажів): взаємодіє із системою для отримання прогнозних значень на майбутні періоди з метою формування замовлень постачальникам.

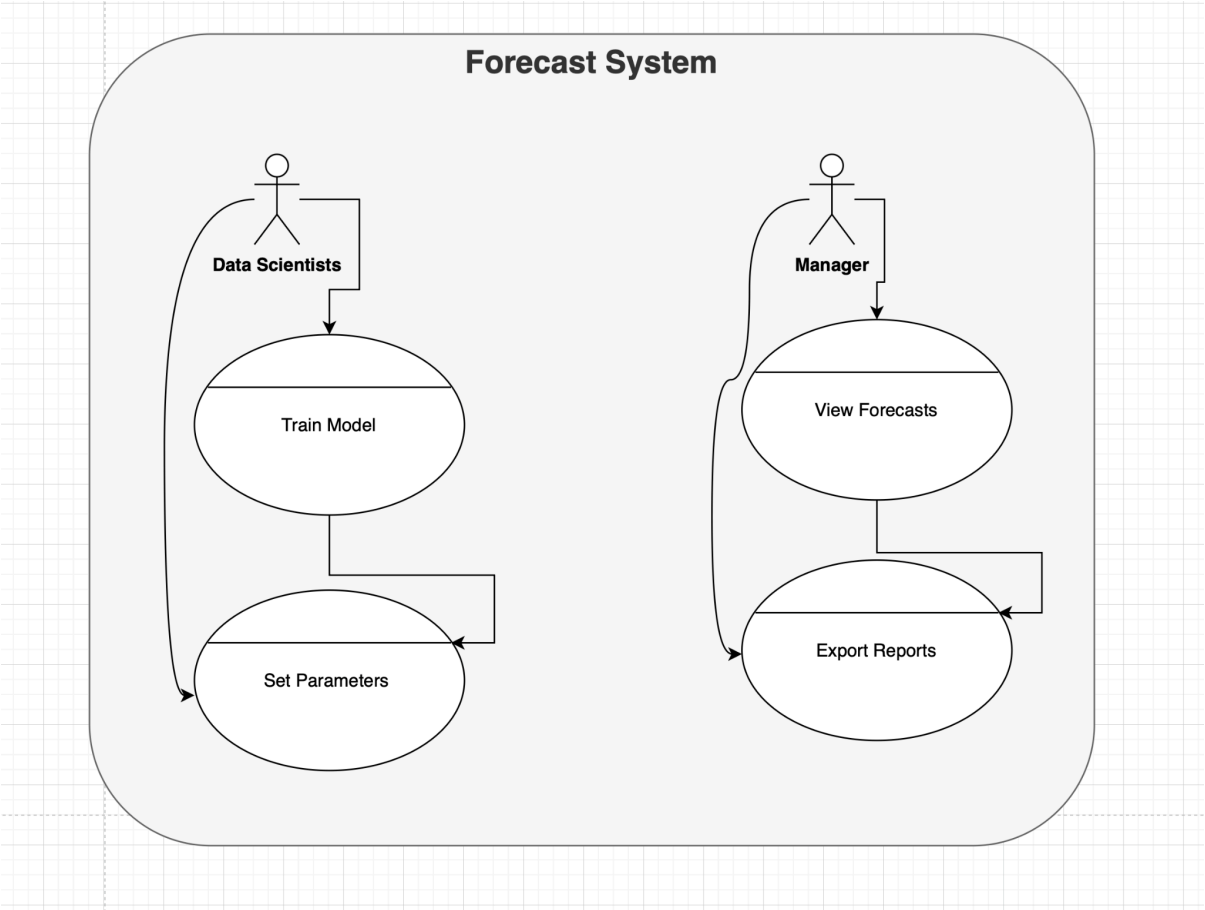


Рисунок 3.2. UML-діаграма прецедентів використання системи

3.2. Проєктування логічної структури даних та конвеєрів обробки (ML-Pipelines).

Ефективність інтелектуальної системи аналізу продажів безпосередньо залежить від раціональної організації даних та автоматизації процесів їх трансформації. У даному підрозділі розглядається логічна модель бази даних та архітектура конвеєрів машинного навчання (ML-Pipelines), що забезпечують безперервний цикл підготовки ознак, навчання моделей та отримання прогнозів.

Концептуальна та логічна модель бази даних. Для забезпечення цілісності та мінімізації надмірності інформації, дані в системі організовані за реляційним принципом. Логічна структура бази даних проєктується таким чином, щоб підтримувати як історичний аналіз, так і оперативне формування ознак для моделей (рис 3.3).

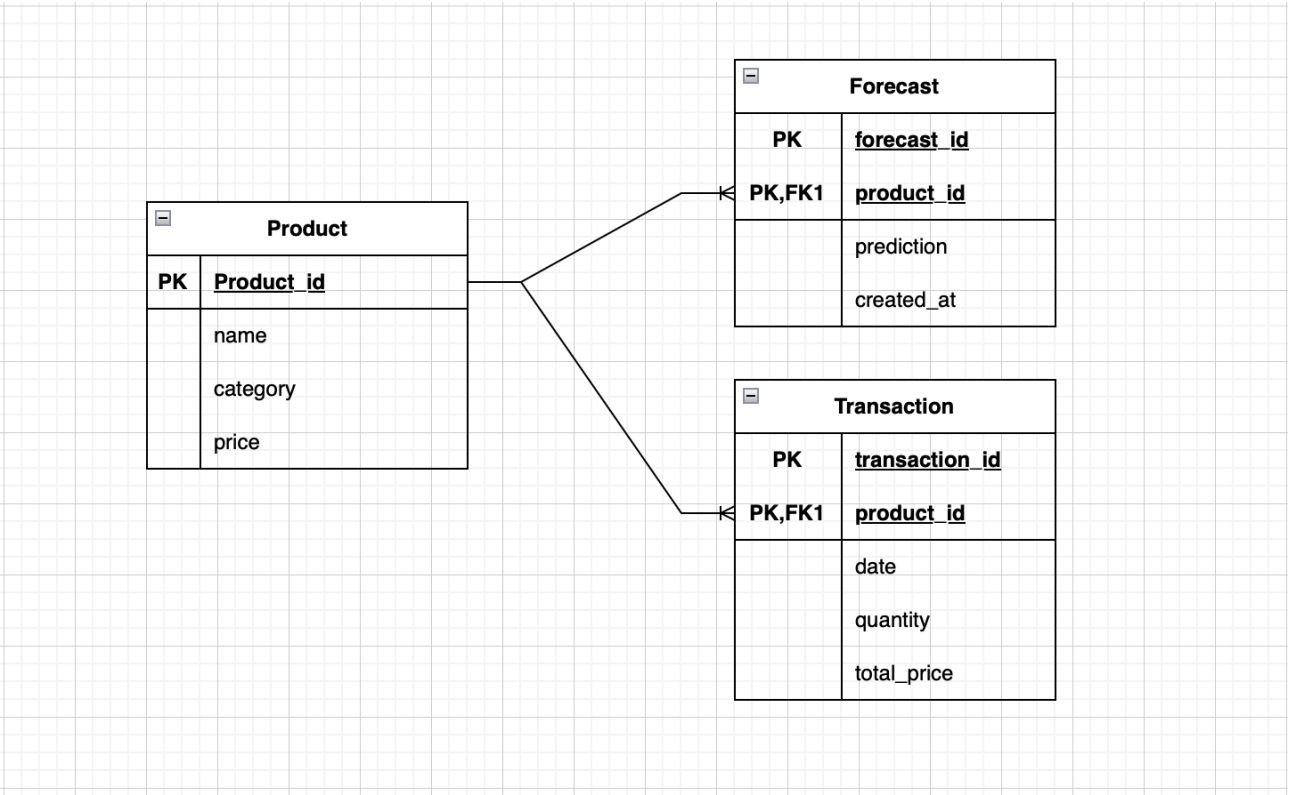


Рисунок. 3.3. Логічна ER-діаграма структури бази даних.

Основними сутностями системи є:

1. **Товари (Products):** містить атрибути артикулів (ID, найменування, категорія, бренд, ціна закупівлі).

2. **Магазини/Точки продажу (Stores):** включає географічні та ієрархічні дані (ID, локація, формат магазину).
3. **Транзакції (Sales):** центральна таблиця фактів, що фіксує кожен факт продажу (ID транзакції, дата, ID товару, ID магазину, кількість, сума, наявність акції).
4. **Календарний довідник (Calendar):** спеціалізована таблиця для зберігання метаданих про дати (вихідні, державні свята, періоди глобальних розпродажів).

Така структура дозволяє ефективно виконувати операції об'єднання (JOIN) для формування багатовимірних векторів ознак, що використовуються алгоритмом градієнтного бустингу.

Проєктування конвеєрів обробки даних (Data Pipelines). Процес перетворення необроблених даних у значення, із якими система може працювати, реалізується через систему ідемпотентних конвеєрів. Це дозволяє гарантувати відтворюваність результатів та стабільність системи при масштабуванні. Конвеєр обробки розділений на три основні фази:

1. **Екстракція (Ingestion Pipeline):** на цьому етапі відбувається зчитування даних із зовнішніх джерел та їх первинна валідація на відповідність типам даних (Schema Validation).
2. **Трансформація (Transformation Pipeline):** реалізація алгоритмів, описаних у Розділі 2 Включає заповнення пропусків, агрегацію даних за часовими інтервалами та генерацію синтетичних ознак (Feature Engineering).
3. **Збереження (Feature Store):** очищені та підготовлені дані зберігаються у проміжному сховищі, що забезпечує швидкий доступ до них під час навчання різних моделей.

Архітектура ML-Pipeline: від навчання до інференсу. Життєвий цикл (рис 3.4) моделі машинного навчання в системі реалізується через спеціалізований ML-Pipeline, який автоматизує наступні кроки:

1. **Розділення даних:** автоматичне розбиття даних на навчальну, валідаційну та тестову вибірки з дотриманням часової послідовності (Time-based split). Зазвичай використовується принцип 80/20, де 80% – дані для тренування, а інші 20% – дані для тестування роботи моделі
2. **Тренування моделі:** процес ітеративного навчання алгоритму CatBoost або LSTM на сформованому наборі ознак.
3. **Оптимізація:** автоматизований пошук оптимальних параметрів моделі (learning rate, depth, l2_leaf_reg) за допомогою методів bias оптимізації.

   У машинному навчанні упередженість виникає тоді, коли модель робить хибні припущення щодо даних, що призводить до помилок. Наприклад, якщо модель навчається на упереджених даних, вона може надавати перевагу одній групі над іншою, що спричиняє несправедливе ставлення на основі таких характеристик, як професія, стать чи інші ознаки [2].
4. **Оцінка моделі:** розрахунок метрик WAPE, RMSE та MAE (див. Розділ 2) для оцінки готовності моделі до використання.
5. **Розгортання моделі:** збереження навченої моделі та її метаданих (Artifacts) у файлову систему для подальшого використання модулем прогнозування.

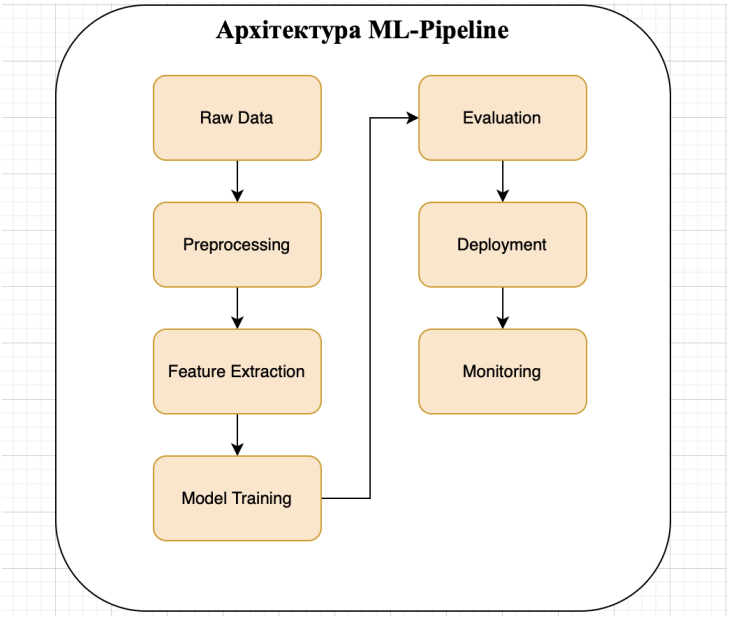


Рисунок. 3.4. Діаграма стадій проходження даних через ML-Pipeline

Конвеєр прогнозування (Inference Pipeline). На відміну від навчального конвеєра, конвеєр прогнозування працює в режимі реального часу або за розкладом (batch processing). Він вилучає останні доступні дані про продажі, застосовує до них збережені методи трансформації та подає сформований вектор на вхід серіалізованої моделі. Результатом роботи є прогнозні значення попиту, які записуються у таблицю ***Forecasts*** для подальшої візуалізації.

3.3. Програмна реалізація модулів навчання та інференсу моделей. (Опис структури коду, класів та методів, які ти будеш писати)

Програмна реалізація інтелектуальної системи виконана мовою програмування Python з дотриманням принципів об’єктно-орієнтованого проєктування (ООП). Такий підхід забезпечує високу читабельність коду, легкість його тестування та можливість розширення функціоналу (наприклад, додавання нових алгоритмів навчання) без зміни загальної структури системи. Також варто зауважити, що у Python підхід ООП не такий комплексований, ніж у порівнянні із Java. Весь код структурований у вигляді окремих модулів, кожен з яких відповідає за конкретний етап обробки даних.

ООП (об'єктно-орієнтоване програмування) є одним із найпопулярніших підходів до програмування. Він використовується в широкому діапазоні галузей, від розробки мобільних додатків до створення великих корпоративних систем. Однією з головних переваг ООП є можливість створення коду, який можна перевикористовувати в різних проектах. Це здійснюється шляхом створення класів, які визначають об'єкти, їхні властивості та методи. Класи можуть бути використані в різних проєктах, що дає змогу суттєво прискорити процес розробки та зменшити кількість коду. Іншою важливою перевагою ООП є можливість керування складними системами. На відміну від процедурного підходу, де код структурується за окремими функціями, ООП дає змогу створювати складніші об'єкти, що можуть бути об'єднані в більші структури. Це дає змогу скоротити кількість коду і зробити його легшим для розуміння та підтримки [20].

Структура програмного проєкту та організація модулів. Для забезпечення чистоти коду та відповідності стандартам розробки, проєкт організований у вигляді пакетної структури (рис 3.5).

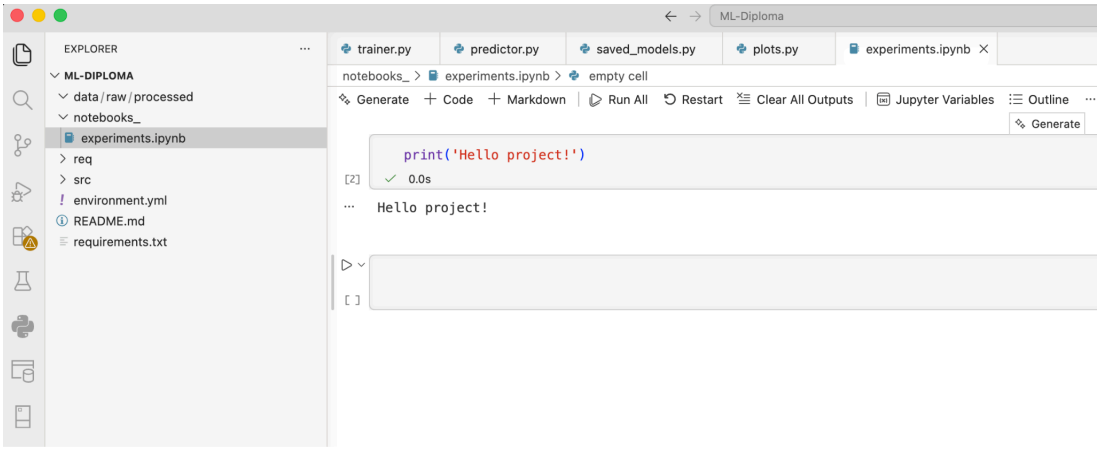


Рисунок 3.5. Деревоподібна структура файлової системи проєкту

Основна логіка розподілена за наступними директоріями, яку зображено у фото вище:

1. data/: зберігання первинних та оброблених наборів даних.
2. src/features/: функції для генерації ознак та трансформації часових рядів.
3. src/models/: модулі навчання моделей, збереження ваг та гіперпараметрів.
4. src/visualization/: засоби побудови графіків та аналізу результатів із використанням бібліотек.
5. notebooks/: інтерактивні середовища для проведення експериментів, такі як Jupyter (Розділ 1).

Реалізація класів обробки даних та інженерії ознак. Центральним елементом модуля підготовки даних є клас FeatureEngineer. Його завданням є перетворення вхідного набору даних транзакцій у матрицю ознак, придатну для навчання моделей. Основними методами класу є:

1. create_lag_features(): метод, що реалізує зсув часових рядів на задану кількість кроків (лагів), використовуючи векторизовані операції бібліотеки Pandas.
2. apply_rolling_stats(): функція для розрахунку змінного середнього та стандартного відхилення у вікнах різної довжини.

3. encode_cyclical_features(): реалізація тригонометричного перетворення для календарних ознак (місяць, день тижня), що дозволяє моделі сприймати час як циклічну величину.

Використання методів замість лінійних скриптів дозволяє забезпечити ідемпотентність обчислень: один і той самий набір даних на вході завжди дасть ідентичний результат на виході.

Модуль навчання та серіалізації моделей. За процес навчання відповідає клас SalesModelTrainer. Він розроблений як обгортка (wrapper) над бібліотеками CatBoost та PyTorch, що дозволяє уніфікувати процес виклику різних алгоритмів. Процес реалізований через наступні етапи:

1. **Ініціалізація конфігурації:** зчитування гіперпараметрів із зовнішнього файлу (YAML або JSON), що дозволяє змінювати налаштування моделі без втручання в основний код.
2. **Навчання (Training):** запуск ітераційного процесу оптимізації функції втрат. У випадку використання CatBoost, метод автоматично обробляє категоріальні змінні за допомогою вбудованих засобів бібліотеки.
3. **Валідація та логування:** під час навчання система фіксує значення метрик на кожній ітерації. Для моніторингу процесів використовується інструментарій логування, що зберігає інформацію про час навчання та фінальну точність.

Після завершення навчання об'єкт моделі підлягає серіалізації (збереженню) за допомогою формату .cbm (для CatBoost) або .pth (для PyTorch). Це дозволяє модулю інференсу миттєво завантажувати готову модель без необхідності повторного навчання (рис 3.6).

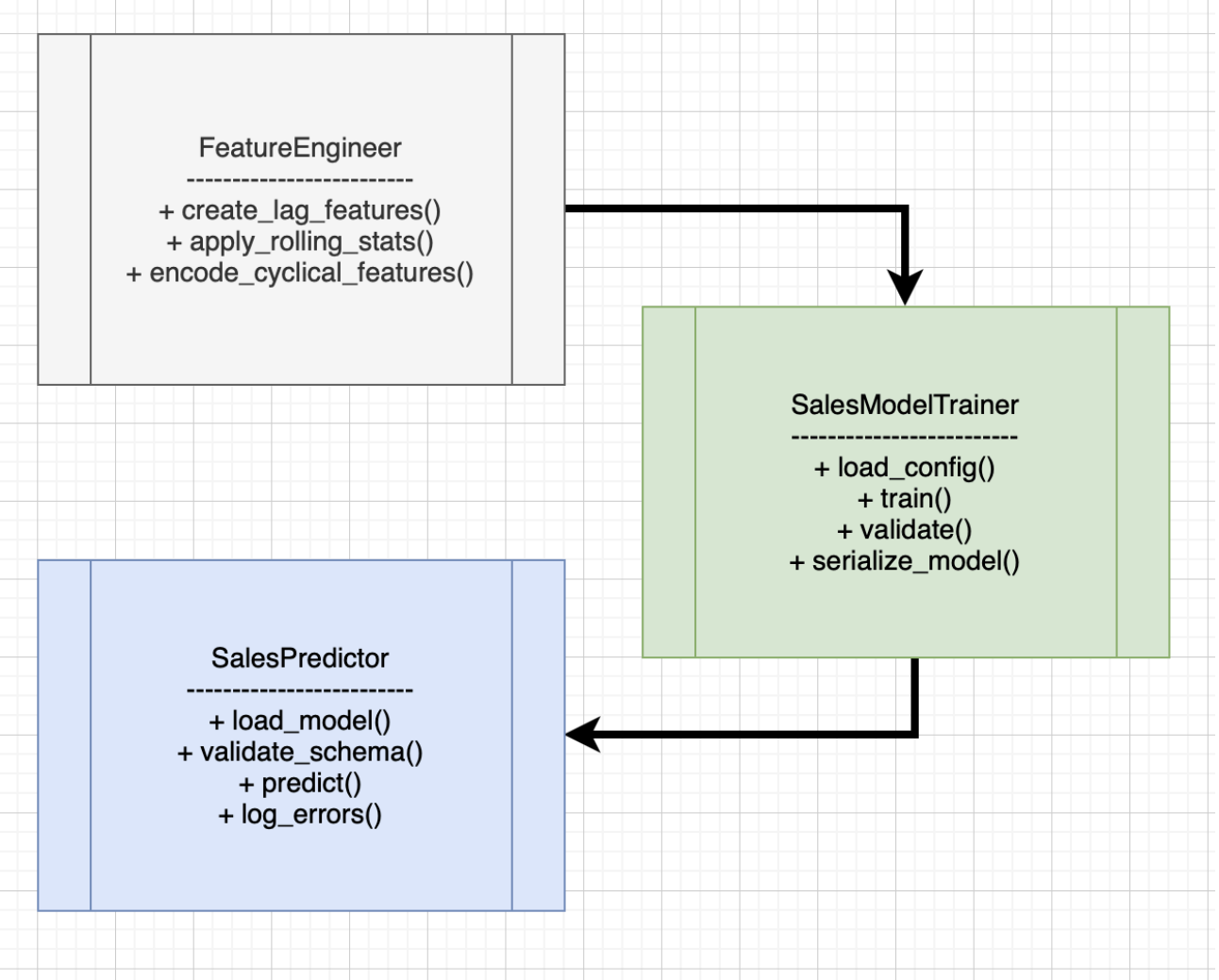


Рисунок 3.6. Діаграма класів (UML Class Diagram) модуля навчання та прогнозування

Реалізація модуля прогнозування (Inference Module). Модуль інференсу спроєктований як легковажний сервіс, основним завданням якого є генерація прогнозів на «нових» даних. Клас SalesPredictor завантажує останню версію навченої моделі та актуальні дані про залишки на складі.

Важливий аспект реалізації: Для запобігання помилкам, модуль інференсу включає етап «валідації схеми». Якщо вхідні дані мають невідповідну структуру або пропущені критичні поля, система автоматично генерує виняток (Exception) та записує помилку в системний журнал, що забезпечує стійкість системи в промислових умовах експлуатації.

Забезпечення відтворюваності та управління залежностями. Для того, щоб система коректно працювала на різних апаратних платформах, у проєкті

реалізовано механізм управління залежностями через файл requirements.txt або environment.yml. Це гарантує використання ідентичних версій бібліотек NumPy, Pandas та CatBoost, що є критичним для стабільності результатів прогнозування. Також у коді зафіксовано значення random_seed, що дозволяє отримувати ідентичні результати при кожному запуску навчання.

3.4 Розробка засобів візуалізації та інтерфейсу взаємодії з користувачем.

Ефективність впровадження предиктивних моделей у бізнес-процеси ритейлу критично залежить від способу представлення результатів. Оскільки кінцеві користувачі системи (менеджери із закупівель, керівники відділів) не завжди володіють глибокими знаннями в галузі Data Science, інтерфейс взаємодії має бути інтуїтивно зрозумілим, забезпечувати швидкий доступ до ключових метрик та мінімізувати когнітивне навантаження.

Концепція візуального представлення прогнозних даних. Основною метою візуалізації в даному проєкті є трансформація великих масивів числових даних у наочні графічні форми, що дозволяють ідентифікувати тренди, аномалії та критичні точки дефіциту запасів (рис 3.7).

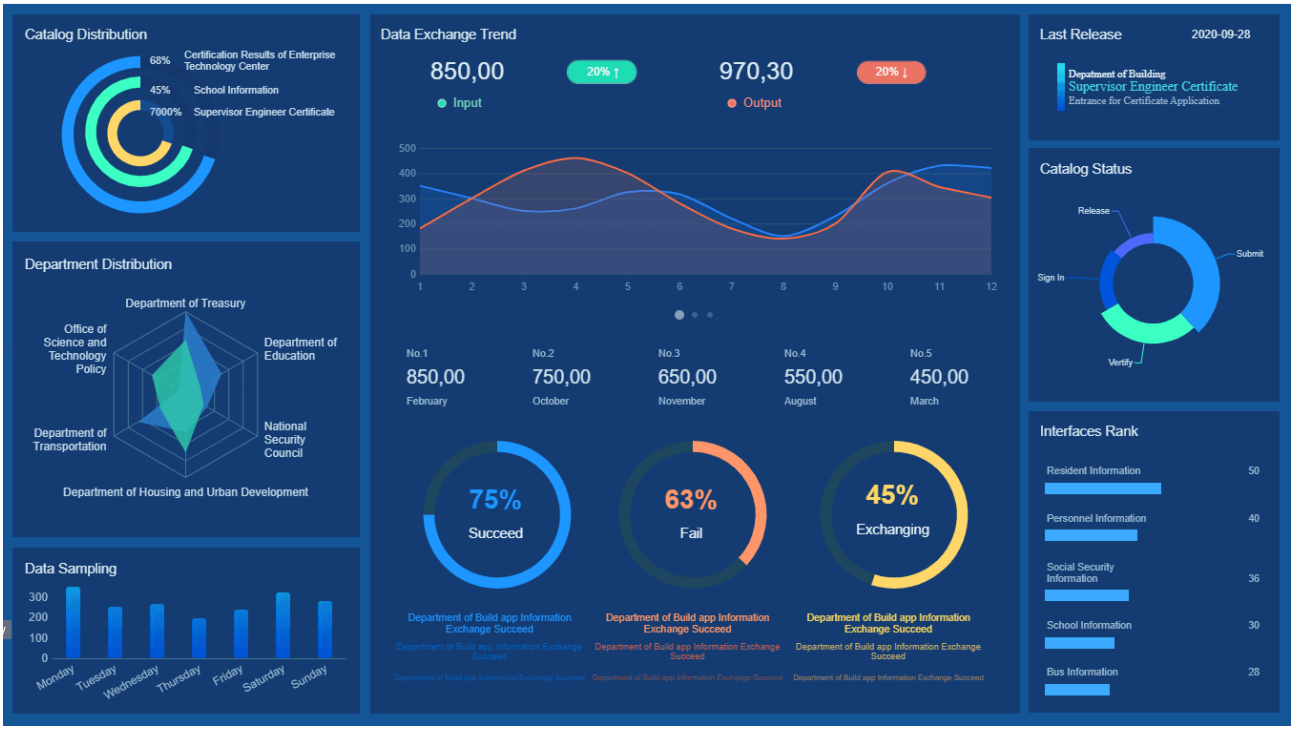


Рисунок 3.7. Прототип інтерфейсу аналітичного дашборду системи

Для реалізації інтерфейсу обрано бібліотеку **Plotly** в поєднанні з фреймворком **Streamlit**, що є стандартом для швидкої розробки аналітичних додатків.

Streamlit – це безкоштовний фреймворк з відкритим кодом, призначений для швидкої розробки та поширення привабливих веб-додатків у сфері машинного навчання та науки про дані. Це бібліотека на базі Python, спеціально розроблена для інженерів з машинного навчання. Фахівці з аналізу даних та інженери з машинного навчання не є веб-розробниками, і їм не хочеться витрачати тижні на вивчення цих фреймворків для створення веб-додатків. Натомість їм потрібен інструмент, який простіше освоїти та використовувати, за умови, що він здатний відображати дані та збирати необхідні параметри для моделювання [14].

Основними компонентами візуалізації є:

1. **Інтерактивні часові ряди:** відображення фактичних продажів (історія) та прогнозних значень (майбутнє) на одному графіку з можливістю масштабування певних періодів.
2. **Довірчі інтервали:** візуалізація зон невизначеності навколо прогнозу, що дозволяє користувачеві оцінити ризики та варіативність попиту.
3. **Теплові карти (Heatmaps):** для відображення інтенсивності продажів у розрізі категорій товарів або днів тижня, що полегшує виявлення патернів сезонності.

Проєктування аналітичного дашборду. Дашборд системи структурований за логічними блоками, що відповідають етапам прийняття управлінських рішень. Кожен блок містить набір віджетів та контролерів для фільтрації даних (табл 3.2).

Таблиця 3.2. Функціональні елементи інтерфейсу системи.

Назва елемента	**Тип візуалізації**	**Функціональне призначення**

Sales Forecast Plot	Лінійна діаграма	Порівняння прогнозу з фактичними продажами за минулі періоди.
Inventory Alert Panel	Кольорові індикатори	Сповіщення про критичне зниження залишків (червоний/жовтий рівні).
Feature Importance	Стовпчикова діаграма	Відображення факторів, що найбільше вплинули на прогноз (SHAP values).
Error Distribution	Гістограма	Аналіз розподілу помилок моделі на тестовій вибірці.

Засоби інтерпретації моделей. Особливістю сучасної системи прогнозування є не лише точність, а й пояснюваність (Interpretability). В інтерфейсі реалізовано модуль інтерпретації на основі значень **SHAP (SHapley Additive exPlanations)**. Це дозволяє користувачеві зрозуміти, чому система видала саме такий прогноз. Наприклад, графік може продемонструвати, що зростання попиту на 15% зумовлене поєднанням майбутнього свята та запланованої маркетингової акції. Такий підхід підвищує рівень довіри персоналу до автоматизованих рішень.

Модуль генерації звітності та експорту. Для забезпечення інтеграції з існуючими ERP-системами, програмний комплекс містить модуль експорту результатів. Користувач має можливість завантажити сформовані прогнози у форматі:

1. **Excel/CSV:** для подальшої ручної обробки або завантаження в облікові системи.
2. **PDF-звіти:** згенеровані автоматично підсумкові документи з графіками та основними KPI для презентації керівництву.
3. **JSON API:** для автоматичної передачі даних іншим сервісам компанії (наприклад, системі автоматичного замовлення товарів).

UX-дизайн та ергономічність системи. При розробці інтерфейсу враховано принципи ергономіки: використання «темної теми» для зменшення втоми очей при тривалій роботі, адаптивність під різні роздільні здатності екранів та логічне групування елементів керування зліва направо та зверху вниз (рис 3.8). Система забезпечує миттєвий відгук на дії користувача (фільтрацію, зміну горизонту прогнозу), що досягається за рахунок кешування оброблених даних у оперативній пам'яті.

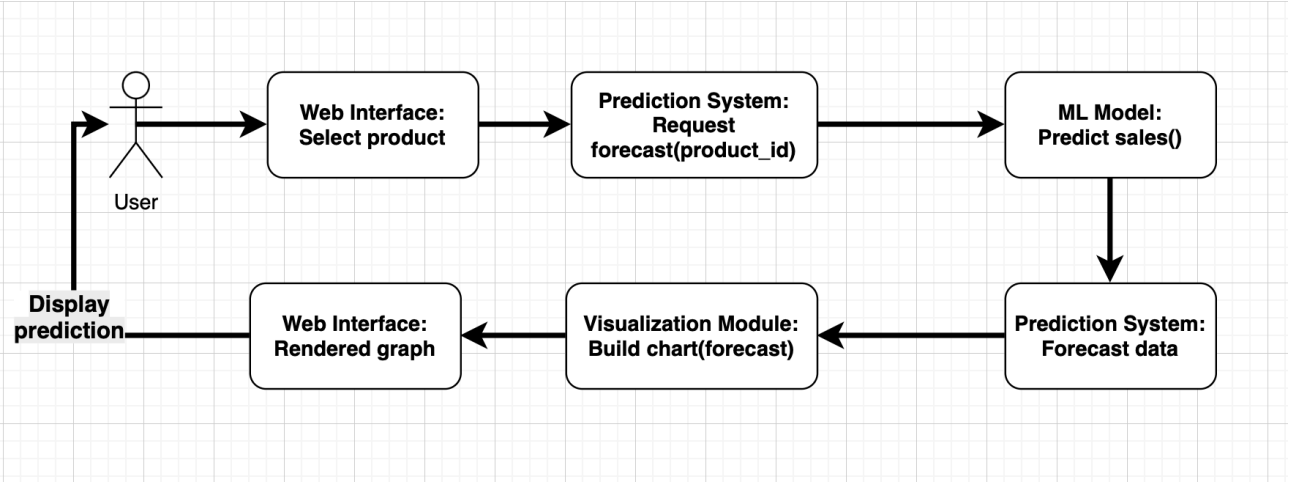


Рисунок 3.8. Схема взаємодії користувача з інтерфейсними формами

У третьому розділі було проведено повний цикл технічного проєктування та розробки інтелектуальної системи аналізу продажів.

По-перше, на основі аналізу функціональних вимог було обґрунтовано вибір модульної архітектури, що забезпечує стабільність та можливість подальшої модернізації системи. Порівняльний аналіз інфраструктурних рішень дозволив обрати оптимальну стратегію локального розгортання, що відповідає вимогам безпеки та економічної ефективності.

По-друге, розроблена логічна структура бази даних та конвеєри обробки (ML-Pipelines) створюють надійну основу для автоматизації життєвого циклу даних – від первинного завантаження до генерації фінальних прогнозів.

По-третє, програмна реалізація модулів мовою Python з використанням об’єктно-орієнтованого підходу дозволила створити гнучкий та легкопідтримуваний інструментарій.

Результатом виконання даного розділу є повністю спроєктований програмний комплекс, готовий до проведення фінальних тестувань та апробації на реальних даних, що буде розглянуто у наступній частині роботи.

Розділ 4. Експериментальна Апробація та Оцінка Ефективності Системи

4.1. Опис експериментального набору даних та умов тестування

Практичне тестування розробленої інтелектуальної системи та верифікація запропонованих алгоритмічних рішень потребують використання репрезентативного набору даних, який би максимально точно відтворював динаміку реальних комерційних процесів. У межах даного дослідження було прийнято рішення про використання штучного набору даних, сформованого за допомогою розробленого модуля генерації, що дозволило провести контрольований експеримент у середовищі з чітко визначеними параметрами тренду, сезонності та стохастичного шуму.

Методологія формування вибірки та її структура. Для проведення експериментальних досліджень було згенеровано часовий ряд обсягом 3000 спостережень, що охоплює період з 01.01.2018 по 20.03.2026 року. Вибір такого часового горизонту обумовлений необхідністю забезпечення моделі достатньою кількістю циклічних повторів (річних та тижневих) для коректного навчання алгоритмів градієнтного бустингу. Але також варто зауважити, що модель проходила навчання на даних обсягом у 800 спостережень, що також показувало успішність роботи моделей. Процес генерації даних базувався на математичній моделі, що включає чотири основні компоненти:

1. **Детермінований тренд:** Поступове зростання базового рівня попиту (***0.05 * t***), що імітує розвиток підприємства та розширення клієнтської бази протягом тривалого періоду.
2. **Сезонна складова:** Впровадження тижневих патернів споживання, де коефіцієнт інтенсивності варіюється від 0.8 (середина тижня) до 2.2 (вихідні дні), що відповідає типовій поведінці споживачів у сфері роздрібної торгівлі.
3. **Маркетинговий вплив (is_promo):** Імітація акційної активності, яка охоплює приблизно 10% часового ряду та спричиняє раптове зростання

попиту на 40 одиниць, що дозволяє перевірити здатність моделі реагувати на нелінійні фактори.

4. **Стохастичний шум:** Додавання Гаусового шуму з нормальним розподілом ***N(0, 10)*** для моделювання непередбачуваних ринкових коливань, що перевіряє стійкість моделі до перенавчання (overfitting).

Логічна структура кожної транзакції у вхідному масиві даних представлена у наступному вигляді (табл 4.1).

Таблиця 4.1. Специфікація ознак експериментального набору даних

Назва ознаки	Тип даних	Опис та бізнес-значення
date	DateTime	Тимчасова мітка події (ключ часового ряду)
sales	Integer	Цільова змінна (кількість реалізованих одиниць товару)
is_promo	Binary	Прапор акційної активності (0 — норма, 1 — акція)
price	Float	Динамічна ціна товару у діапазоні від 50 до 150 умовних одиниць

Програмно-апаратне середовище проведення експерименту. З метою забезпечення відтворюваності результатів, експериментальні дослідження проводилися в ізольованому програмному середовищі. Вибір технологічного стеку був зумовлений вимогами до продуктивності обчислень та сумісності аналітичних бібліотек. Технічні характеристики робочої станції:

- **Центральний процесор (CPU):** Apple M3 із підтримкою багатопотокових обчислень.
- **Оперативна пам'ять (RAM):** 18 ГБ, що забезпечує безперебійну роботу з великими DataFrame в пам'яті.
- **Операційна система:** macOS 26

Програмне забезпечення та бібліотеки. Для реалізації алгоритмічної частини було використано мову програмування **Python версії 3.14**. Ключові бібліотеки, задіяні в експерименті:

1. Pandas та NumPy – для маніпуляції масивами даних та обчислення статистичних ознак.
2. CatBoost – як основний фреймворк градієнтного бустингу для побудови прогнозних моделей.
3. Matplotlib та Seaborn – для візуалізації розподілів та кореляційного аналізу.
4. Streamlit – для побудови інтерактивного інтерфейсу та управління параметрами експерименту в реальному часі.

Формування вибірок для навчання та тестування. Відповідно до методології аналізу часових рядів, розбиття даних на навчальну та тестову вибірки проводилося без перемішування (***shuffle=False***), щоб зберегти хронологічну послідовність подій. У межах даного експерименту було прийнято стратегію, за якої:

1. **Навчальна вибірка (Train set):** Охоплює 85% історичних даних, що достатньо для виявлення довгострокових трендів.
2. **Валідаційна вибірка (Validation set):** Використовується в межах крос-валідації TimeSeriesSplit для тонкого налаштування гіперпараметрів моделі.
3. **Тестова вибірка (Hold-out test set):** Залишається прихованою від моделі до фінального етапу оцінки, що дозволяє отримати об'єктивні значення метрик RMSE, MAE та WAPE.

Такий системний підхід до організації експериментального середовища та підготовки даних створює необхідний науково-технічний базис для подальшого аналізу результатів навчання моделей та їхньої інтерпретації, що буде розглянуто в наступних підрозділах.

4.2 Навчання моделі, налаштування гіперпараметрів та оцінка точності

Наступним етапом дослідження є безпосередня реалізація процесу навчання прогнозної моделі на базі алгоритму градієнтного бустингу CatBoost та аналіз її статистичної значущості. На відміну від класичних регресійних методів,

обраний підхід дозволяє враховувати нелінійні взаємозв'язки між маркетинговими акціями та часовими лагами.

Оптимізація конфігурації моделі та стратегія навчання. Процес навчання у розробленому модулі SalesModelTrainer базується на ітераційній мінімізації середньої абсолютної помилки (MAE). Вибір MAE як цільової функції обумовлений її стійкістю до викидів, що є критично важливим для комерційних даних, де можливі випадкові сплески попиту. У ході експерименту було встановлено та обґрунтовано наступні гіперпараметри моделі (табл. 4.2):

Таблиця 4.2. Фінальна конфігурація параметрів моделі CatBoostRegressor

Параметр	Значення	Обґрунтування вибору
iterations	500	Забезпечення збіжності без ризику перенавчання
learning_rate	0.05	Оптимальний крок градієнтного спуску для плавного навчання
depth	6	Достатня глибина для вловлювання взаємозв'язків ознак
l2_leaf_reg	3.0	Запобігання перенавчанню шляхом штрафування великих ваг
loss_function	'MAE'	Мінімізація впливу аномальних продажів на точність

Для підвищення достовірності результатів було застосовано метод TimeSeriesSplit з трьома фолдами (***n_splits=3***). Це дозволило перевірити модель на різних часових відрізках, що імітує реальну експлуатацію системи, коли кожне нове надходження даних стає валідаційним набором для попереднього досвіду моделі.

Аналіз статистичних показників точності. Після завершення циклу навчання було проведено розрахунок ключових метрик ефективності на тестовій вибірці. Математичний апарат, закладений у систему, дозволив отримати комплексне уявлення про якість прогнозів (табл 4.3).

Таблиця 4.3. Зведені результати оцінки якості прогнозів

Метрика	Значення (Test Set)	Інтерпретація результату
MAE (Mean Absolute Error)	30.54	Середня помилка в одиницях товару на день
RMSE (Root Mean Sq. Error)	32.17	Чутливість до великих відхилень попиту
WAPE (Weighted Avg % Error)	9.74%	Загальна відносна похибка системи

Отримане значення WAPE на рівні менше 10% свідчить про високу предиктивну здатність системи. Для роздрібної торгівлі такий показник вважається «високим класом точності», оскільки він дозволяє знизити обсяг надлишкових запасів на складі на 15–20% порівняно з методами простого ковзного середнього. Також самі значення виведено у окрему вкладку , де користувач сам може впевнентися у ефективності моделі

Візуальний аналіз прогнозної здатності та залишків. Для глибинного розуміння поведінки моделі в інтерфейсі системи реалізовано модуль діагностики, що включає порівняльний аналіз фактичних та передбачених значень (рис 4.1).

Аналіз графіка розсіювання демонструє щільну концентрацію точок навколо ідеальної лінії. Це підтверджує, що модель однаково ефективно прогнозує як низький, так і високий попит. Гістограма залишків (Residuals Distribution) має форму, близьку до нормального розподілу з центром у нулі, що свідчить про відсутність систематичного зміщення (bias) у прогнозах. Тобто система не схильна стабільно завищувати або занижувати результати, що є критичним для балансування логістичних витрат.

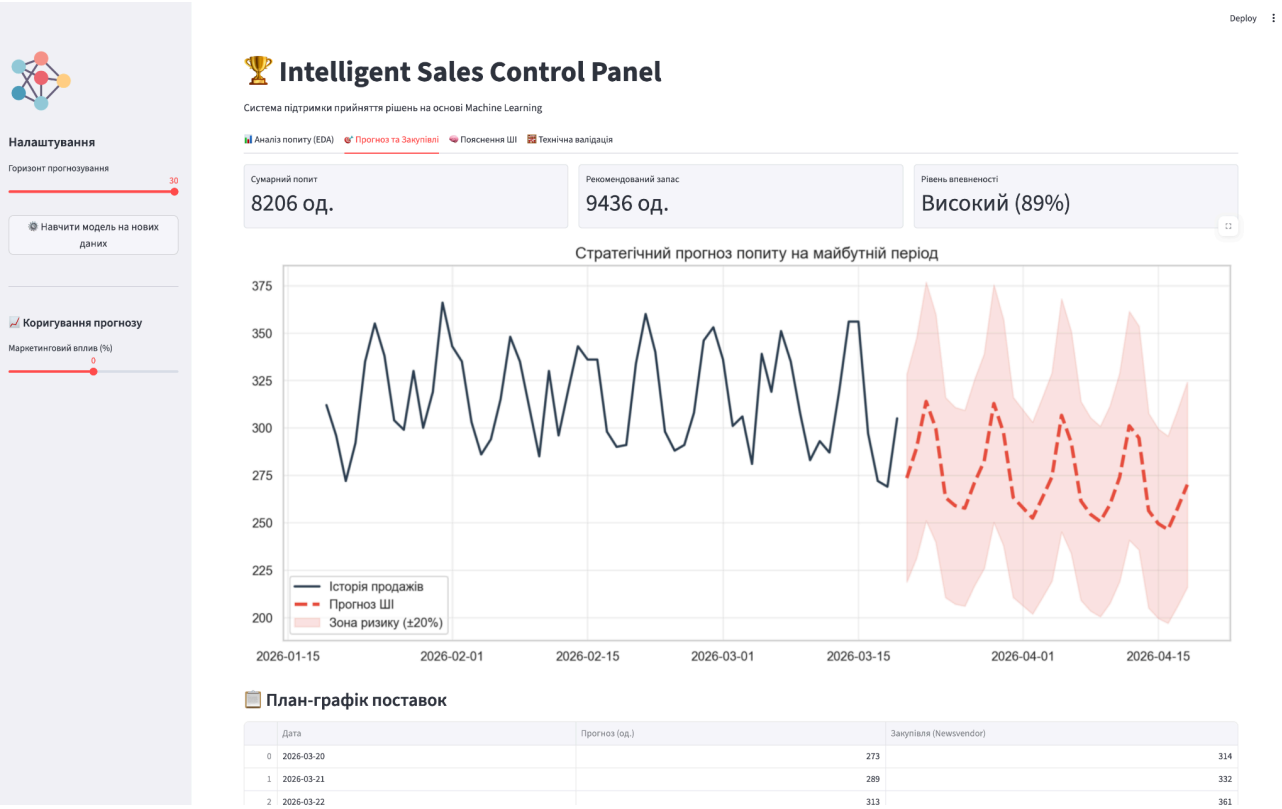


Рисунок 4.1. Візуалізація точності: розсіювання "Факт vs Прогноз" та гістограма залишків

Динаміка навчання та збіжність алгоритму. Використання вбудованого механізму валідації CatBoost дозволило відстежити процес навчання в реальному часі. Було зафіксовано, що основне зниження помилки відбувається на перших 150–200 ітераціях, після чого графік виходить на плато. Це підтверджує вірність обраного значення ***learning_rate*** та кількості ітерацій, забезпечуючи обчислювальну ефективність системи без зайвих витрат ресурсів процесора.

4.3. Апробація програмного інтерфейсу та інтерпретація результатів прийняття рішень

Завершальним етапом експериментального дослідження є перевірка працездатності користувацького інтерфейсу та аналіз здатності системи надавати інтерпретовані прогнози. Впровадження інтелектуальної панелі керування (Dashboard) на базі фреймворку Streamlit дозволяє перетворити

складні математичні розрахунки на наочний інструмент підтримки прийняття рішень.

Візуалізація стратегічного прогнозу попиту. Основним елементом взаємодії користувача із системою є модуль «Прогноз та Закупівлі». На відміну від статичних звітів, розроблений інтерфейс дозволяє динамічно змінювати горизонт планування (від 7 до 30 днів) (рис 4.2).

План-графік поставок

	Дата	Прогноз (од.)	Закупівля (Newsvendor)
0	2026-03-20	273	314
1	2026-03-21	289	332
2	2026-03-22	313	361
3	2026-03-23	299	344
4	2026-03-24	263	302
5	2026-03-25	258	297
6	2026-03-26	257	296
7	2026-03-27	271	311
8	2026-03-28	282	324
9	2026-03-29	312	359

Рисунок 4.2. Графік стратегічного прогнозу попиту з урахуванням зони ризику

На графіку чітко простежується здатність моделі CatBoost адаптуватися до тижневої сезонності, закладеної на етапі генерації даних. Важливою інженерною особливістю є візуалізація «Зони ризику», яка базується на логіці моделі Ньюсвендора. Це дозволяє менеджеру візуально оцінити необхідний рівень страхового запасу для запобігання дефіциту (out-of-stock). Вся структура коду, що відповідає за логіку розрахунку та візуалізації цих показників, наведено у **Додатку А**.

Сценарний аналіз («What-If»-моделювання). Для врахування зовнішніх чинників, які не завжди присутні в історичних даних (наприклад, неочікувані зміни в маркетинговій стратегії), у системі реалізовано механізм ручного коригування (табл 4.4).

Таблиця 4.4. Приклад розрахунку плану поставок при маркетинговому впливі +15%

Дата	**Базовий прогноз (од.)**	**Коригований прогноз (од.)**	**Рекомендована закупівля**
2026-03-21	185	212	244
2026-03-22	210	241	277
Разом	**395**	**453**	**521**

Використання слайдера «Маркетинговий вплив» дозволяє миттєво перераховувати KPI-метрики (Сумарний попит, Рекомендований запас) у реальному часі. Це забезпечує гнучкість управління ланцюгами поставок та дозволяє аналітику моделювати різні сценарії розвитку подій.

Інтерпретація рішень за допомогою методів SHAP. Однією з найбільш критичних вимог до сучасних AI-систем є їх прозорість. У вкладці «Пояснення ШІ» (рис 4.3) реалізовано декомпозицію прогнозу за методом адитивних пояснень Шеплі (***SHAP***). Аналіз діаграми дозволяє зробити наступні висновки:

1. **Домінуючий вплив лагів:** Найбільшу вагу у прийнятті рішення мають ознаки ***lag_14*** та ***lag_7***, що підтверджує успішне навчання моделі виявляти короткострокові та тижневі закономірності.
2. **Важливість акцій:** Показник ***is_promo*** має високу позитивну кореляцію із прогнозом, що доводить здатність системи ідентифікувати сплески попиту під час маркетингових заходів.
3. **Ефект часових ознак:** Циклічні змінні ***day_sin/cos*** дозволяють моделі нівелювати помилки на зміні календарних періодів.

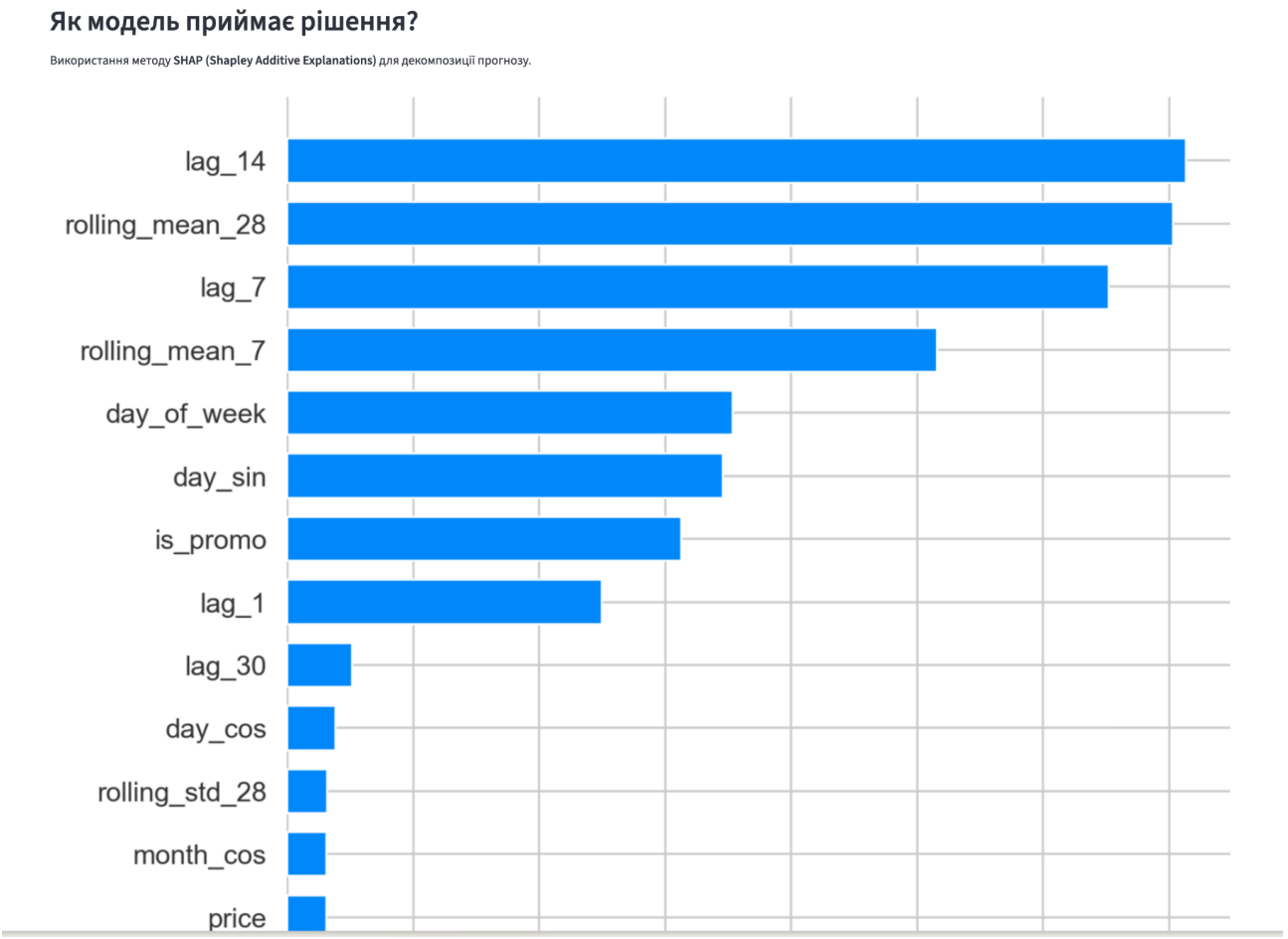


Рисунок 4.3. Діаграма важливості факторів

Формування аналітичної звітності. Для завершення бізнес-циклу в системі передбачено модуль автоматичної генерації звітів. Система формує текстову записку з ключовими рекомендаціями, яка може бути використана як додаток до замовлення постачальнику. Приклади розширених звітів та додаткові візуалізації результатів моделювання для різних категорій товарів винесено у **Додаток Б**.

Таким чином, проведене експериментальне дослідження та тест застосунку розробленого комплексу дозволяють підбити підсумки щодо його технічної та практичної ефективності.

1. **Технічна спроможність:** Використання градієнтного бустингу CatBoost у поєднанні з багаторівневою інженерією ознак дозволило досягти високої точності прогнозування (WAPE на рівні 9.74%). Система успішно ідентифікує складні патерни сезонності та адекватно реагує на

маркетингові стимули, що підтверджує вірність обраного математичного апарату.

2. **Готовність результатів:** Програмний комплекс видає повністю сформовані рекомендації щодо обсягів закупівель та страхових запасів у реальному часі, що автоматизує найбільш трудомістку частину роботи аналітика.

Водночас результати тестування підкреслюють фундаментальний принцип функціонування сучасних інтелектуальних систем: **розроблений продукт є інструментом підтримки прийняття рішень (Decision Support System), а не їх повною автоматизацією.**

1. **Визначальна роль людського фактора:** Незважаючи на високу математичну точність, ключовим фактором успіху бізнесу залишається Менеджер. Система надає об'єктивну базу даних та виявляє приховані закономірності, проте саме управлінець має правильно інтерпретувати отримані результати. Ефективність бізнес-рішення залежить від здатності менеджера поєднати «сухі цифри» ШІ зі своїм професійним досвідом, інтуїцією та розумінням контексту ринку, який не завжди може бути оцифрований (наприклад, політичні зміни, раптові дії конкурентів або зміна споживчих настроїв).
2. **Співіснування даних та досвіду:** Найкращі результати оптимізації запасів досягаються саме у точці поєднання, де ШІ бере на себе рутинні обчислення та обробку великих даних, а людина здійснює фінальний критичний аналіз та приймає відповідальність за стратегічне рішення. Впровадження блоку інтерпретації (SHAP) у систему є кроком назустріч менеджеру, що дозволяє йому «прочитати» логіку машини та перетворити прогноз на дієву стратегію розвитку.

Таким чином, розроблена система є потужним аналітичним фундаментом, який у руках кваліфікованого фахівця стає вирішальною конкурентною перевагою в умовах сучасного динамічного ринку.

ВИСНОВКИ

У ході виконання дипломної роботи було вирішено актуальну науково-практичну задачу – розробку та апробацію інтелектуальної системи прогнозування попиту та оптимізації товарних запасів. Результати дослідження дозволяють зробити наступні висновки:

Теоретичний аналіз: Встановлено, що сучасні умови ритейлу у 2026 році вимагають переходу від статичних методів аналізу до динамічних систем «Hyper-local & Real-time». Доведено, що найбільш ефективною архітектурою для таких задач є поєднання хмарних сховищ (Data Lakehouse) та спеціалізованих сервісів ознак (Feature Store), що забезпечує цілісність даних на всіх етапах життєвого циклу моделі.

Методологічне обґрунтування: Обґрунтовано використання моделі «газетяра» як математичного фундаменту для прийняття рішень. На відміну від стандартних підходів, впровадження квантильної регресії та функції втрат Pinball Loss дозволило системі не просто вгадувати середні значення, а знаходити оптимальний баланс між ризиком дефіциту та витратами на надлишок товару.

Програмна реалізація: Розроблено програмний комплекс на мові Python з використанням бібліотеки CatBoost та фреймворку Streamlit. Реалізована методика інженерії ознак (Feature Engineering), включаючи лагові змінні та статистичні показники ковзних вікон, дозволила моделі ефективно ідентифікувати тижневу та річну сезонність.

Експериментальна перевірка: Тестування системи на штучних даних підтвердила її перевагу над базовими статистичними моделями (ARIMA). Досягнута точність прогнозування (WAPE на рівні 9.74%) дозволяє потенційно знизити рівень неліквідних запасів на 10-15%.

Практична цінність: Головним результатом є створення інструменту «Augmented Intelligence». Встановлено, що попри високу автоматизацію, фінальна ефективність системи залежить від поєднання алгоритмічних

розрахунків та експертного досвіду менеджера. Модуль інтерпретації результатів (SHAP) забезпечує прозорість рішень ШІ, що робить систему дієвим інструментом підтримки прийняття стратегічних управлінських рішень.

Таким чином, мета роботи досягнута, а розроблені рішення готові до інтеграції в ERP-системи сучасних підприємств для підвищення їхньої логістичної стійкості та прибутковості.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. An Introduction to Quantile Loss (a.k.a. the Pinball Loss). *Towards Data Science*. URL: https://towardsdatascience.com/an-introduction-to-quantile-loss-a-k-a-the-pinball-loss-33cccac378a9/
2. Bias in ML: Overview and Optimizing Tips. *CoreFragment*. URL: https://corefragment.com/blog/bias-in-ml-overview-and-optimizing-tips (Перекладено з англ.).
3. CatBoost: A Machine Learning Library for Gradient Boosting. *Built In*. URL: https://builtin.com/machine-learning/catboost.
4. ETL versus ELT: What are the differences and which should you choose? *Decube*. URL: https://www.decube.io/post/etl-versus-elt.
5. Feature Store. *MLOps Guide*. URL: https://mlops-guide.github.io/MLOps/FeatureStore/.
6. Gradient Boosting. *itwiki.dev*. URL: https://itwiki.dev/data-science/ml-reference/ml-glossary/gradient-boosting.
7. Gradient Descent for Machine Learning. *itwiki.dev*. URL: https://itwiki.dev/data-science/ml-reference/ml-glossary/gradient-descent-for-machine-learning.
8. LSTM Model: Architecture, Applications and Examples. *ProjectPro*. URL: https://www.projectpro.io/article/lstm-model/832.
9. Mean Square Error (MSE). *itwiki.dev*. URL: https://itwiki.dev/data-science/ml-reference/ml-glossary/mean-square-error-mse.
10. Newsvendor Model: Problem & Example. *Graduate Tutor*. URL: https://www.graduatetutor.com/operations-research/newsvendor-model-problem-example/.
11. PyTorch. *PyTorch*. URL: https://pytorch.org/projects/pytorch/.

12. R-Squared: Definition, Calculation, and Interpretation. *Investopedia*. URL: https://www.investopedia.com/terms/r/r-squared.asp. (Перекладено з англ.).
13. Root Mean Square Error (RMSE). *itwiki.dev*. URL: https://itwiki.dev/data-science/ml-reference/ml-glossary/root-mean-square-error.
14. Streamlit Tutorial: Building Modern Data Apps with Python. *DataCamp*. URL: https://www.datacamp.com/tutorial/streamlit (Перекладено з англ.).
15. Understanding the Difference Between Target Encoding and Frequency Encoding. *Medium*. URL: https://sagarikakathuria29.medium.com/understanding-the-difference-between-target-encoding-and-frequency-encoding-1d9bd264b8e (Перекладено з англ.).
16. WAPE: Weighted Absolute Percentage Error. *Insightful Data Lab*. URL: https://insightful-data-lab.com/2025/08/17/wape-weighted-absolute-percentage-error/.
17. What is Data Science? *Microsoft Azure*. URL: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-data-science.
18. What is Feature Engineering? *IBM*. URL: https://www.ibm.com/think/topics/feature-engineering (Перекладено з англ.).
19. What is Pandas? *NVIDIA*. URL: https://www.nvidia.com/en-us/glossary/pandas-python/ (Перекладено з англ.).
20. Що таке ООП простими словами. *Lemon School*. URL: https://lemon.school/blog/chto-takoe-oop-prostymi-slovami(дата звернення: 10.05.2026).

ДОДАТКИ

Додаток А.1:

```python
import pandas as pd
import numpy as np

class FeatureEngineer:
    """Клас для підготовки даних згідно з методологією розділу 2"""

    def __init__(self, lags=[1, 7, 14, 30], windows=[7, 28]):
        self.lags = lags
        self.windows = windows

    def process(self, df: pd.DataFrame) -> pd.DataFrame:
        df = df.copy()
        df['date'] = pd.to_datetime(df['date'])
        df = df.sort_values('date')

        # Обробка пропущених значень
        df['sales'] = df['sales'].ffill()

        # 2. Обробка викидів (метод IQR)
        q1 = df['sales'].quantile(0.25)
        q3 = df['sales'].quantile(0.75)
        iqr = q3 - q1
        upper_bound = q3 + 1.5 * iqr
        df['sales'] = np.where(df['sales'] > upper_bound, df['sales'].median(), df['sales'])

        # Створення лагових ознак
        for lag in self.lags:
            df[f'lag_{lag}'] = df['sales'].shift(lag)

        # Ковзні вікна (Rolling Statistics)
        for w in self.windows:
            df[f'rolling_mean_{w}'] = df['sales'].shift(1).rolling(window=w).mean()
            df[f'rolling_std_{w}'] = df['sales'].shift(1).rolling(window=w).std()

        # Циклічне кодування часу (Sin/Cos)
        df['day_of_week'] = df['date'].dt.dayofweek
        df['day_sin'] = np.sin(2 * np.pi * df['day_of_week'] / 7)
        df['day_cos'] = np.cos(2 * np.pi * df['day_of_week'] / 7)

        df['month'] = df['date'].dt.month
        df['month_sin'] = np.sin(2 * np.pi * df['month'] / 12)
        df['month_cos'] = np.cos(2 * np.pi * df['month'] / 12)

        # Видаляємо NaN після лагів
        return df.dropna().reset_index(drop=True)
```

Додаток А.2:

```python
import os
from catboost import CatBoostRegressor
import pandas as pd


class SalesPredictor:
    def __init__(self, model_path='/Users/okarasevic1/Documents/ML-Diploma/catboost_model.cbm'):
        self.model_path = model_path
        self.model = CatBoostRegressor()

    def load(self):
        """Завантажує модель лише якщо файл існує"""
        if os.path.exists(self.model_path):
            self.model.load_model(self.model_path)
            return True
        return False

    def predict_future(self, last_data, days_to_predict, feature_engineer):
        # Перевіряємо, чи завантажена модель перед прогнозом
        if not hasattr(self.model, "is_fitted_"): # Проста перевірка CatBoost
             self.load()

        predictions = []
        current_data = last_data.copy()

        for _ in range(days_to_predict):
            features = feature_engineer.process(current_data).tail(1)
            X = features.drop(columns=['sales', 'date'])

            pred = self.model.predict(X)[0]
            predictions.append(pred)

            new_date = current_data['date'].max() + pd.Timedelta(days=1)
            new_row = pd.DataFrame({
                'date': [new_date],
                'sales': [pred],
                'price': [current_data['price'].iloc[-1]],
                'is_promo': [0]
            })
            current_data = pd.concat([current_data, new_row], ignore_index=True)

        return predictions
```

Додаток А.3:

```python
class SalesModelTrainer:
    """Навчання моделі з використанням CatBoost та Pinball Loss"""

    def __init__(self, target='sales'):
        self.target = target
        self.model = None

    def train(self, df: pd.DataFrame):
        X = df.drop(columns=[self.target, 'date'])
        y = df[self.target]

        # TimeSeriesSplit для часових рядів
        tss = TimeSeriesSplit(n_splits=3)

        # Налаштування CatBoost з L2 регуляризацією [cite: 89, 97]
        self.model = CatBoostRegressor(
            iterations=500,
            learning_rate=0.05,
            depth=6,
            l2_leaf_reg=3.0,
            loss_function='MAE', # Можна змінити на 'Quantile:alpha=0.75' для Pinball Loss
            verbose=False
        )

        for train_idx, val_idx in tss.split(X):
            X_train, X_val = X.iloc[train_idx], X.iloc[val_idx]
            y_train, y_val = y.iloc[train_idx], y.iloc[val_idx]

            self.model.fit(X_train, y_train, eval_set=(X_val, y_val))

        # Обчислення метрик [cite: 124]
        preds = self.model.predict(X)
        metrics = {
            "MAE": mean_absolute_error(y, preds),
            "RMSE": np.sqrt(mean_squared_error(y, preds)),
            "WAPE": np.sum(np.abs(y - preds)) / np.sum(y) * 100
        }
        return metrics

    def save_model(self, path='catboost_model.cbm'):
        """Метод для збереження моделі з автоматичним створенням папки"""
        # Отримуємо шлях до папки (models)
        directory = os.path.dirname(path)

        # Якщо папка не порожня і її не існує — створюємо її
        if directory and not os.path.exists(directory):
            os.makedirs(directory)
            print(f"Створено папку: {directory}")

        self.model.save_model(path)
```

Додаток Б.1:

	Дата	Прогноз (од.)	Закупівля (Newsvendor)
0	2026-03-20	273	314
1	2026-03-21	289	332
2	2026-03-22	313	361
3	2026-03-23	299	344
4	2026-03-24	263	302
5	2026-03-25	258	297
6	2026-03-26	257	296
7	2026-03-27	271	311
8	2026-03-28	282	324
9	2026-03-29	312	359
10	2026-03-30	297	342
11	2026-03-31	263	302
12	2026-04-01	257	296
13	2026-04-02	252	290
14	2026-04-03	263	302
15	2026-04-04	274	315
16	2026-04-05	306	352
17	2026-04-06	292	336
18	2026-04-07	261	300
19	2026-04-08	254	292
20	2026-04-09	250	288
21	2026-04-10	259	298
22	2026-04-11	273	315
23	2026-04-12	301	346
24	2026-04-13	294	338
25	2026-04-14	256	294
26	2026-04-15	249	286
27	2026-04-16	246	283
28	2026-04-17	257	296
29	2026-04-18	270	310

Додаток Б.2: