МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: АНАЛІТИЧНИЙ ПРОГРАМНИЙ МОДУЛЬ ВЕБДОДАТКУ ДЛЯ ІНТЕРНЕТ-МАГАЗИНУ

Здобувача вищої освіти другого (магістерського) рівня,
ІІ курсу, групи КМІз-24-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма: «Інформаційні управляючі системи і технології в економіці»

Лаговського Володимира Вікторовича

Науковий керівник — Ратушняк Тетяна Володимирівна, канд. фіз.-мат. наук, доцент

Керівник освітньо-професійної програми — Омельчук Антон Анатолійович, канд. техн. наук

Допущено до захисту за рішенням кафедри протокол від «09» січня 2026 № 11

Завідувач кафедри — Філоненко Михайло Миколайович, канд. фіз.-мат. наук, доцент

Ірпінь – 2026

Реєстрація № 9 16.01.2026 _____
(дата) (підпис відповідальної особи)

Результати перевірки до захисту 09.01.26 _____
(до захисту, на доопрацювання) (дата) (підпис)

Результат захисту 5 (відмінно) 11.02.2026 _____
(оцінка) (дата) (підпис)

Голова комісії: _____ О.С. Дюбанов
(підпис)

Члени комісії: _____ А.Ю. Горбовий
(підпис)

_____ М.М. Філоненко
(підпис)

_____ І.Д. Погорєловська
(підпис)

БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки вбалах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії,секретаря.
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст кваліфікаційної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	30		11.02.2026 Дюбанов О.С. Горбовий А.Ю. Філоненко М.М. Погорєловська І.Д. Суховецька Л.П.
2. Дотримання вимог щодо оформлення	від 0 до 10	9		
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	5		
4. Рецензент	від 0 до 5	5		
РАЗОМ БАЛІВ	від 0 до 50	49		

Оцінювання кваліфікаційної роботи науковим керівником - 49 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100- бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національноюшкалою	Дата	П.І.Б.: голови, членів комісії, секретаря,
98	A	5	11.02.2026	Дюбанов О.С. Горбовий А.Ю. Філоненко М.М. Погорєловська І.Д. Суховецька Л.П.

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	3 (задовільно)
50-59	E	
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій
Кафедра комп'ютерних та інформаційних технологій і систем
Галузь знань 12 «Інформаційні технології»
Спеціальність 122 «Комп'ютерні науки»
Освітня програма «Інформаційні управляючі системи і технології в економіці»

Затверджую
Завідувач кафедри _____ Філоненко М.М.
«11» грудня 2024

Завдання на кваліфікаційну роботу
здобувачу вищої освіти
Лаговському Володимиру Вікторовичу

1. **Тема кваліфікаційної роботи**: Аналітичний програмний модуль веб-додатку для інтернет-магазину.

Затверджена наказом ректора від «10» грудня 2024 № 1852.

2. Строк здачі закінченої роботи «09» січня 2026.

3. **Мета роботи:** підвищення економічної ефективності функціонування інтернет-магазинів шляхом розробки та імплементації аналітичного програмного модуля, який забезпечує автоматизоване прогнозування попиту з використанням гібридних моделей машинного навчання та розрахунок оптимальних параметрів поповнення запасів в умовах невизначеності.

Об'єкт дослідження: процеси управління товарними запасами та прогнозування споживчого попиту в системах електронної комерції

Предмет дослідження: методи, математичні моделі, алгоритми машинного навчання та програмні архітектурні рішення для оптимізації рівня запасів та мінімізації логістичних витрат інтернет-магазинів.

Орієнтовна база дослідження: технічна документація та академічні публікацій; інформаційна інфраструктура типових CMS платформ електронної комерції.

4. **Вихідні дані до роботи**: дані про продажі, згенеровані відповідно до типових патернів поведінки інтернет магазинів.

5. **Перелік графічного матеріалу**: рисунки, таблиці які розміщені в додатках.

6. **Зміст кваліфікаційної роботи:**

Розділ 1. Аналіз предметної області та постановка задачі

Аналіз сучасного стану, динаміки розвитку та проблематики управління запасами в електронній комерції. Огляд існуючих програмних рішень, методів автоматизації та аналіз ринкових прогалин. Теоретичні основи, математичні моделі та методи прогнозування в управлінні запасами. Обґрунтування архітектурних рішень та технологічного стеку. Постановка задачі дослідження

Розділ 2. Методи та алгоритми прогнозування попиту

Математична постановка задачі прогнозування. Інженерія ознак для часових рядів. Алгоритми градієнтного бустінгу. Ансамблеві методи навчання. Метрики оцінки якості прогнозів. Алгоритми оптимізації управління запасами.

Розділ 3. Проєктування архітектури аналітичного програмного модуля.

Вимоги до функціоналу системи. Вибір технологічного стеку. Проєктування архітектури додатку. UML діаграми

Розділ 4. Реалізація аналітичного модуля

Структура проєкту та організація коду. Реалізація модуля підготовки даних. Реалізація модуля прогнозування. Реалізація модуля оптимізації запасів. Реалізація REST API. Реалізація користувацького інтерфейсу

7. Календарний план виконання роботи

№ п/п	Назва етапів кваліфікаційноїроботи	Строк виконання етапів роботи	
		За планом	Фактично
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 15.12.2024	До 15.12.2024
2	**І розділ** (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	26.05.2025	26.05.2025
3	**ІІ розділ** (аналіз і опис методів та алгоритмів роботи програмного модуля)	02.09.2025	02.09.2025
4	**ІІІ – ІV розділи** (проєктування архітектури та реалізація аналітичного програмного модуля)	24.11.2025	24.11.2025
5	**Висновки**	18.12.2025	18.12.2025
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	02.01.2026	02.01.2026
7	Отримання відгуку керівника	02.02.2026	02.02.2026
8	Отримання зовнішньої рецензії	05.02.2026	05.02.2026

8. Дата видачі завдання «11» грудня 2024

9. Керівник кваліфікаційної роботи Ратушняк Т.В.

10. Гарант освітньо-професійної програми Омельчук А.А.

11. Завдання до виконання отримав здобувач вищої освіти Лаговський В.В.

АНОТАЦІЯ

Аналітичний програмний модуль вебдодатку для інтернет-магазину

Електронна комерція функціонує в умовах високої невизначеності, стикаючись із проблемами «ефекту батога», інтермітентного попиту та короткого життєвого циклу товарів. На ринку існує розрив між базовими функціями CMS-платформ та дорогими Enterprise-системами, що робить актуальним створення доступного інструменту для автоматизації управління запасами на основі машинного навчання.

Мета роботи полягає у підвищенні економічної ефективності функціонування інтернет-магазинів шляхом розробки та імплементації аналітичного програмного модуля, який забезпечує автоматизоване прогнозування попиту та розрахунок оптимальних параметрів поповнення запасів.

Об’єкт дослідження: процеси управління товарними запасами та прогнозування споживчого попиту в системах електронної комерції.

Предмет дослідження: методи, математичні моделі, алгоритми машинного навчання та програмні архітектурні рішення для оптимізації рівня запасів.

У роботі використано методи системного аналізу, теорію управління запасами, статистичний аналіз часових рядів, методи інженерії ознак та ансамблеві методи машинного навчання.

Наукова новизна одержаних результатів полягає у розробці гібридного підходу до прогнозування попиту, який поєднує ансамблеві методи ML із динамічним розрахунком страхового запасу на основі похибки прогнозу, що дозволяє адаптувати запаси до реальної волатильності ринку. Формалізовано методику інженерії ознак для часових рядів в умовах переривчастого попиту.

Практичне значення. Розроблено повнофункціональний прототип у вигляді вебдодатку на базі Python/Django із реалізацією REST API. Впровадження модуля дозволяє зменшити рівень замороженого капіталу та знизити кількість випадків дефіциту товару.

Кваліфікаційна робота викладена на 69 сторінках, містить вступ, чотири розділи, висновки, список використаних джерел із 81 найменування та 17 додатків.

Ключові слова: електронна комерція, управління запасами, машинне навчання, аналітичний програмний модуль, вебдодаток, прогнозування попиту, оптимізація запасів.

Рік виконання кваліфікаційної роботи: 2025.

Рік захисту роботи: 2026

ABSTRACT
Analytical software module of a web application for an online store

E-commerce operates under conditions of high uncertainty, facing challenges such as the "bullwhip effect," intermittent demand, and short product life cycles. There is a market gap between the basic functions of CMS platforms and expensive Enterprise systems, making the creation of an accessible tool for automated inventory management based on machine learning highly relevant.

The aim of the work is to increase the economic efficiency of online stores by developing and implementing an analytical software module that provides automated demand forecasting and calculation of optimal inventory replenishment parameters.

Object of research: inventory management processes and consumer demand forecasting in e-commerce systems.

Subject of research: methods, mathematical models, machine learning algorithms, and software architectural solutions for optimizing inventory levels.

The work utilizes methods of system analysis, inventory management theory, statistical time series analysis, feature engineering methods, and ensemble machine learning methods.

The scientific novelty of the obtained results lies in the development of a hybrid approach to demand forecasting, which combines ensemble ML methods with dynamic safety stock calculation based on forecast error. This allows inventory levels to adapt to real market volatility. A methodology for feature engineering for time series under conditions of intermittent demand has been formalized.

Practical value. A fully functional prototype has been developed in the form of a web application based on Python/Django with REST API implementation. The implementation of the module allows for reducing the level of frozen capital and decreasing the number of stockout instances.

The qualification paper is presented on 69 pages and contains an introduction, four chapters, conclusions, a list of references comprising 81 items, and 17 appendices.

Keywords: e-commerce, inventory management, machine learning, demand forecasting, inventory optimization.

Year of completion of the qualification paper: 2025.

Year of defense: 2026.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ 4
ВСТУП 6
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ 10
1.1. Аналіз сучасного стану, динаміки розвитку та проблематики управління запасами в електронній комерції 10
1.2. Огляд існуючих програмних рішень, методів автоматизації та аналіз ринкових прогалин 13
1.3. Теоретичні основи, математичні моделі та методи прогнозування в управлінні запасами 15
1.4. Обґрунтування архітектурних рішень та технологічного стеку 19
1.5. Постановка задачі дослідження 20
Висновки до розділу 1 22
РОЗДІЛ 2 МЕТОДИ ТА АЛГОРИТМИ ПРОГНОЗУВАННЯ ПОПИТУ 23
2.1. Математична постановка задачі прогнозування 23
2.2. Інженерія ознак для часових рядів 24
2.3. Алгоритми градієнтного бустінгу 26
2.4. Ансамблеві методи навчання 29
2.5. Метрики оцінки якості прогнозів 30
2.6. Алгоритми оптимізації управління запасами 31
Висновки до розділу 2 34
РОЗДІЛ 3. ПРОЄКТУВАННЯ АРХІТЕКТУРИ АПМ 35
3.1. Вимоги до функціоналу системи 35
3.2. Вибір технологічного стеку 37
3.3. Проєктування архітектури додатку 40
3.5. UML діаграми 45
Висновки до розділу 3 47
РОЗДІЛ 4. РЕАЛІЗАЦІЯ АНАЛІТИЧНОГО МОДУЛЯ 48
4.1. Структура проєкту та організація коду 48
4.2. Реалізація модуля підготовки даних 49
4.3. Реалізація модуля прогнозування 52
4.4. Реалізація модуля оптимізації запасів 55

4.5. Реалізація REST API 56
4.6. Реалізація користувацького інтерфейсу 57
Висновки до розділу 4 59
ВИСНОВКИ 60
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 63
ДОДАТКИ 70

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

API (Application Programming Interface) — інтерфейс прикладного програмування.

CNN (Convolutional Neural Network) — згорткова нейронна мережа.

EOQ (Economic Order Quantity - економічно обгрунтована партія замовлення.

ETL (Extract — Transform — Load) — процес отримання, обробки та завантаження даних.

FE (Feature Engineering) - інженерія ознак.

FR – функціональні вимоги.

IMS (Inventory Management Software) - програмне забезпечення для управління запасами.

LSTM (Long Short-Term Memory) — нейронна мережа з «довгою та короткою пам’яттю».

MAPE - середня абсолютна відсоткова похибка.

MASE - середня абсолютна маштабована похибка.

ML (Machine Learning) — машинне навчання.

MVC (Model-View-Controller) - Модель – Подання – Контролер.

MVT (Model-View-Template) - Модель – Подання – Шаблон.

NFR – нефункціональні вимоги.

RNN - рекурентної нейронної мережі

ROP (Reorder Point) - точка перезамовлення.

SBA (Syntetos-Boylan Approximation) — метод прогнозування попиту для нерегулярних та переривчастих продажів.

SKU (Stock Keeping Unit) - облікова складська одиниця товару.

SS (Safety Stock) - страховий запас.

TSB (Teunter–Syntetos–Babai) — модель Тюнтера–Синтетоса–Бабая.

АМ - аналітичний модуль.

АПМ - аналітичний програмний модуль.

БД - база даних.

ЕК - електронна комерція.

ІМ - інтернет магазини.

МН - машинне навчання.

ТЗ - товарні запаси.

УЗ - управління запасами.

ВСТУП

Цифровізація зробила ЕК лідером торгівлі, але ускладнила управління ланцюгами постачання. Оскільки для більшості компаній витрати на запаси є найбільшими, їх оптимізація стала критичною для фінансової стійкості. Головна проблема полягає в балансуванні між миттєвою доступністю товару для клієнта та мінімізацією витрат на його зберігання, чому заважають специфічні фактори на кшталт «ефекту батога» та неефективність традиційної статистики.

Через брак якісного та доступного софту на ринку, актуальним є створення АМ на базі МН. Таке рішення дозволить автоматизувати поповнення запасів і підвищити ефективність бізнесу без витрат на дорогі корпоративні системи.

Теоретичні основи управління запасами закладені у працях F. Harris (модель EOQ), які розвинули R. Wilson, H. Wagner та інші дослідники операційних досліджень. Класична теорія УЗ широко висвітлена в роботах J. Syntetos, A. Boylan, D. Simchi-Levi.

Застосування МН до задач прогнозування попиту активно досліджується останнє десятиліття. Chen та інші (2019) продемонстрували перевагу градієнтного бустингу над ARIMA для прогнозування роздрібних продажів. Salinasta інші (2020) розробили архітектуру DeepAR на основі рекурентних нейронних мереж для ймовірнісного прогнозування. Работи Hyndman & Athanasopoulos (2021) є фундаментальним джерелом з сучасних методів прогнозування часових рядів.

Проте більшість досліджень зосереджена на теоретичних аспектах або застосуванні в великих корпораціях. Бракує праць, присвячених розробці практичних, доступних інструментів для малого та середнього бізнесу в ЕК. Наукові публікації рідко розглядають питання архітектури та практичної реалізації інтегрованих систем прогнозування та оптимізації запасів. Недостатньо вивчена специфіка гібридних підходів, що поєднують різні типи МН навчання з детерміністичними алгоритмами розрахунку параметрів запасів.

Метою роботи є підвищення економічної ефективності функціонування ІМ шляхом розробки та імплементації АПМ, який забезпечує автоматизоване

прогнозування попиту з використанням гібридних моделей МН та розрахунок оптимальних параметрів поповнення запасів в умовах невизначеності.

Для досягнення мети поставлені наступні завдання:

1. Провести критичний аналіз предметної області УЗ в ЕК, виявити специфічні проблеми та обґрунтувати необхідність інтелектуалізації процесів УЗ.

2. Здійснити порівняльний аналіз існуючих програмних рішень та методів автоматизації, виявити ринкові прогалини та сформулювати функціональні вимоги до розроблюваної системи.

3. Дослідити особливості часових рядів продажів у сфері ЕК, розробити методику попередньої обробки даних та конструювання ознак для алгоритмів МН.

4. Розробити та обґрунтувати гібридну математичну модель прогнозування, яка поєднує алгоритми градієнтного бустингу та ансамблеві методи для точного передбачення попиту.

5. Формалізувати алгоритм прийняття рішень щодо поповнення запасів, який базується на динамічному розрахунку страхового запасу та точки перезамовлення з урахуванням ABC-XYZ класифікації товарів та підтримкою двох політик управління: (s,Q) та (R,S).

6. Спроектувати модульну архітектуру системи з обґрунтуванням технологічного стеку, розробити схему БД та специфікацію REST API.

7. Реалізувати програмний прототип АМ з повним циклом обробки даних: ETL-процеси, навчання моделей, генерація прогнозів, розрахунок рекомендацій та візуалізація результатів.

Об'єктом дослідження є процеси управління товарними запасами та прогнозування споживчого попиту в системах ЕК.

Предметом дослідження виступають методи, математичні моделі, алгоритми МН та програмні архітектурні рішення для оптимізації рівня запасів та мінімізації логістичних витрат ІМ.

Для вирішення поставлених завдань у роботі використовувались:

1. Методи системного аналізу – для дослідження предметної області, виявлення проблем та формулювання вимог до системи.

2. Методи теорії УЗ – моделі EOQ, ROP, SS для розрахунку оптимальних параметрів поповнення.

3. Методи статистичного аналізу часових рядів – ABC-XYZ аналіз, виявлення сезонності, обробка викидів.

4. Методи МН – алгоритми градієнтного бустингу, ансамблеві методи.

5. Методи інженерії ознак – конструювання лагових ознак, ковзних вікон, календарних атрибутів.

6. Методи об'єктно-орієнтованого проектування – UML-моделювання, патерни проектування.

7. Методи БД – проектування реляційної схеми, нормалізація, оптимізація запитів.

Дослідження проводилось на основі:

- даних про продажі, згенерованих відповідно до типових патернів поведінки ІМ;
- технічної документації та академічних публікацій;
- інформаційної інфраструктури типових CMS платформ ЕК.

Наукова новизна одержаних результатів:

1. Розроблено гібридний підхід до прогнозування попиту в ЕК, який поєднує ансамблеві методи МН з динамічним розрахунком страхового запасу на основі похибки прогнозу моделі (RMSE замість історичної волатильності), що дозволяє зменшити рівень замороженого капіталу на 15-25% порівняно з класичними підходами.

2. Формалізовано методику інженерії ознак для часових рядів продажів в умовах інтермітентного попиту, яка включає техніку запобігання витоку даних при розрахунку ковзних статистик та специфічні ознаки для виявлення впливу промо-акцій.

3. Обґрунтовано архітектурний підхід модульного моноліту з розділенням бізнес-логіки на предметно-орієнтовані сервіси, що забезпечує

баланс між простотою розгортання та можливістю незалежного масштабування аналітичних компонентів.

Практичне застосування:

1. Готовий до інтеграції прототип – система реалізована у вигляді вебдодатку на Django з REST API, що дозволяє інтегрувати її з існуючими CMS-платформами або використовувати як автономне рішення.
2. Відкритий програмний код – реалізація доступна під ліцензією MIT, що дозволяє безкоштовно використовувати та адаптувати систему для потреб конкретного бізнесу.
3. Економічний ефект – впровадження АПМ дозволяє:
   - зменшити рівень замороженого капіталу в запасах;
   - знизити кількість випадків дефіциту;
   - автоматизувати рутинні операції менеджера з закупівель.
4. Масштабованість – система протестована для роботи з асортиментом до 10,000 SKU, що покриває потреби 90% ІМ малого та середнього бізнесу.
5. Методичні рекомендації – розроблена документація може бути використана в навчальному процесі.

Структура та обсяг роботи.

Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел (81 найменувань) та 17 додатків. Загальний обсяг роботи становить 69 сторінок.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Аналіз сучасного стану, динаміки розвитку та проблематики управління запасами в електронній комерції

Цифровізація економіки та зміни у поведінкових патернах споживачів перетворили ЕК з альтернативного каналу збуту на домінуючу силу в роздрібній торгівлі. Однак це принесло із собою виклики для УЗ. У сучасних умовах, коли бар'єри входу на ринок знижуються, а конкуренція стає глобальною, ефективність УЗ трансформується з операційної функції в стратегічний актив, що визначає фінансову стійкість та життєздатність бізнесу. УЗ в ЕК це складна аналітична дисципліна, яка вимагає інтеграції передових технологій, математичного моделювання та глибокого розуміння ринкової динаміки. [1]

ЕК характеризується високим ступенем невизначеності та волатильності. Дослідження показують, що для 66% компаній витрати, пов'язані з виробництвом та зберіганням запасів, становлять найбільшу статтю видатків, що робить оптимізацію цього напрямку важливою для збереження маржинальності. [2] Більше того, 77% представників галузі висловлюють лише часткову впевненість у масштабованості своїх поточних інвентарних систем, що свідчить про наявність системного технологічного розриву між потребами ринку та наявними інструментами. [2]

Ключовою проблемою є дилема балансування між рівнем обслуговування клієнтів та витратами на утримання запасів. З одного боку, сучасний споживач очікує миттєвої доступності товару та швидкої доставки. Відсутність товару на складі в онлайн-середовищі має значно більш руйнівні наслідки, ніж у фізичному ритейлі: клієнт може перейти до конкурента за лічені секунди, що призводить не лише до втрати конкретного продажу, але й до зниження показника довічної цінності клієнта та погіршення репутації бренду. [3] Дослідження вказують, що незадоволеність клієнтів через затримки виконання замовлень або відсутність

товару є прямим фактором зниження прибутковості. [5]

Проте надмірні запаси заморожують обігові кошти, збільшують витрати на зберігання та підвищують ризик морального старіння продукції. Статистика свідчить, що перенасичення складів та продаж товарів, яких фактично немає в наявності, є двома найбільшими проблемами для операторів ЕК. [5]

Варто зазначити зростаючу складність операційного середовища. Компанії змушені керувати розширеними портфелями технологічних інструментів, що додає вартості та складності. Близько 47% компаній визнають, що управління зростаючою кількістю інструментів є викликом, а 49% стикаються з проблемами синхронізації даних між виробництвом, інвентаризацією та бухгалтерією. [2] Ця фрагментація даних унеможливлює отримання цілісної картини бізнесу в реальному часі, що є критичним для прийняття зважених управлінських рішень. [6]

Одним із найбільш деструктивних явищ у ланцюгах постачання ЕК є так званий ефект батога. Це явище полягає у посиленні амплітуди коливань попиту по мірі просування інформації вгору по ланцюгу постачання — від кінцевого споживача до роздрібного продавця, дистриб'ютора і, нарешті, виробника. [7] Незначна зміна у попиті з боку клієнта (наприклад, на 5%) може призвести до непропорційно великих коливань у замовленнях виробнику (до 40-100%), що спричиняє хаос у плануванні виробництва та логістики. [9]

У контексті ЕК ефект батога набуває специфічних рис та посилюється під впливом наступних факторів:

1. Онлайн-ритейлери часто використовують динамічне ціноутворення, «чорні п'ятниці» та «кіберпонеділки» для стимулювання продажів. Ці заходи створюють штучні піки попиту, які не відображають реального довгострокового споживання. Алгоритми прогнозування, які не враховують природу цих сплесків, можуть інтерпретувати їх як стійкий тренд зростання, що призводить до надмірних замовлень у постачальників.[7] Дослідження підтверджують, що цінові знижки в онлайн-ринку зазвичай посилюють ефект батога.[10]

2. Прагнення оптимізувати транспортні витрати часто змушує

рітейлерів накопичувати замовлення до певного обсягу, що спотворює інформацію про реальний час виникнення потреби.

3. Відсутність наскрізної видимості та обміну даними між учасниками ланцюга постачання призводить до того, що кожна ланка приймає рішення локально, базуючись на викривленій інформації від попередньої ланки, а не на реальному споживчому попиті. [4]

Наслідками ефекту батога є надлишкові запаси, зростання операційних витрат, погіршення відносин з постачальниками та зниження рівня обслуговування клієнтів. Подолання цього вимагає переходу від реактивних моделей управління до проактивних, що базуються на обміні інформацією в реальному часі та використанні інтелектуальних алгоритмів прогнозування. [11]

ЕК характеризується високою швидкістю оновлення асортименту. Це породжує проблему «холодного старту» — необхідність прогнозування попиту на нові товари, для яких повністю відсутня історія продажів. [12]

Традиційні методи аналізу часових рядів є непридатними в цій ситуації, оскільки вони вимагають історичних даних. Відсутність надійних інструментів змушує менеджерів покладатися на інтуїцію, що часто призводить до помилок: недооцінка попиту на новий продукт веде до втрачених продажів на етапі запуску, коли інтерес споживачів найвищий, а переоцінка — до накопичення неліквідних залишків. [14]

Ситуація ускладнюється коротким життєвим циклом товарів. Продукти можуть застарівати або виходити з моди впродовж кількох місяців або навіть тижнів. У такому середовищі вікно можливостей для отримання прибутку є дуже вузьким, і ціна помилки в прогнозуванні зростає експоненційно. [15] Дослідження вказують, що успішне вирішення проблеми прогнозування для нових товарів вимагає використання методів, таких як кластеризація на основі подібності атрибутів, аналіз характеристик продукту та використання зовнішніх даних. [16]

Значна частина асортименту ІМ, особливо тих, що реалізують стратегію «довгого хвоста», характеризується розрідженим або інтермітентним попитом.

Це товари, продажі яких відбуваються нерегулярно, з численними періодами нульового попиту. [17]

Стандартні методи прогнозування схильні «згладжувати» такий попит, прогнозуючи, наприклад, 0.1 одиниці товару на день. Такий прогноз є непридатним для операційного планування. Висока варіативність такого попиту робить стандартні метрики помилок непридатними через ділення на нуль, що вимагає використання спеціалізованих метрик, наприклад, MASE. [18]

Отже, аналіз предметної області переконливо свідчить, що УЗ в сучасній ЕК є багатовимірною задачею, яка виходить далеко за межі традиційного складського обліку. Вона вимагає розробки та впровадження комплексних програмних рішень, здатних обробляти великі масиви даних, враховувати стохастичну природу попиту та адаптуватися до швидких змін ринкової кон'юнктури.

1.2. Огляд існуючих програмних рішень, методів автоматизації та аналіз ринкових прогалин

Ринок програмного забезпечення для автоматизації ЕК є насиченим та різноманітним, пропонуючи широкий спектр інструментів — від простих плагінів до глобальних ERP-систем. Детальний аналіз функціональності та доступності цих рішень виявляє суттєві обмеження, особливо для малого та середнього бізнесу.

Існуючі на ринку рішення можна структурувати за трьома основними категоріями, кожна з яких має свої переваги та недоліки в контексті аналітичних можливостей.

1. Вбудовані модулі CMS для ЕК платформ (Shopify, WooCommerce, Magento). Ці платформи є фундаментом для більшості ІМ. Вони пропонують базовий функціонал обліку запасів.

Можливості: відстеження поточної кількості, встановлення базових оповіщень про низький рівень запасу.

Обмеження: аналітичні можливості зазвичай обмежені ретроспективними звітами. Функції прогнозування попиту відсутні або реалізовані на примітивному рівні. Вони не враховують сезонність, тренди чи вплив маркетингових акцій. Користувачі змушені експортувати дані в Excel для ручного аналізу, що є трудомістким та схильним до помилок процесом. [20]

2. Спеціалізовані системи УЗ. До цієї категорії належать такі рішення, як Cin7, TradeGecko (QuickBooks Commerce), Katana, Skubana.

Можливості: підтримка мультиканальності, управління складами, комплектація замовлень. Деякі з них пропонують модулі прогнозування.

Обмеження: висока вартість (сотні або тисячі доларів на місяць). [22] Часто такі системи є складними у використанні та налаштуванні. Більшість систем УЗ використовують детерміновані моделі поповнення запасів, які погано працюють в умовах невизначеності, характерної для ЕК.

3. Системи планування ресурсів підприємства. Рішення рівня SAP, Oracle NetSuite, Microsoft Dynamics 365.

Можливості: повна інтеграція всіх бізнес-процесів, потужні модулі бізнес-аналітики, можливість використання AI-інструментів.

Обмеження: надзвичайна складність впровадження, висока вартість ліцензій та підтримки. Для середнього ІМ такий функціонал є надлишковим.

Отже, навіть спеціалізовані інструменти часто не забезпечують необхідної глибини аналізу. Багато компаній досі покладаються на електронні таблиці, які мають суттєві недоліки: відсутність цілісності даних, неможливість роботи в реальному часі, складність масштабування та високий ризик людської помилки. [21]

Проте, з'являються нішеві інструменти прогнозування (Streamline, Inventory Planner), які позиціонують себе як AI-driven рішення. Вони часто функціонують як «чорна скринька», не надаючи пояснення отриманим прогнозам, що знижує довіру користувачів. Інтеграція таких сервісів може бути складною через несумісність форматів даних. Порівняльна таблиця функціональності існуючих систем наведено в додатку А.

Отже, існує гостра потреба у АПМ, який би поєднував у собі:

1. Доступність - можливість інтеграції в існуючі вебдодатки без необхідності купівлі дорогих ліцензій. [23]
2. Інтелектуальність - використання алгоритмів МН замість застарілих статистичних методів, що дозволить враховувати нелінійні залежності та зовнішні фактори. [24]
3. Спеціалізацію - орієнтація на специфічні проблеми ЕК: переривчастий попит, короткий життєвий цикл товарів, вплив промо-акцій.
4. Гнучкість архітектури - реалізація у вигляді мікросервісу, що дозволяє масштабувати обчислювальні потужності. [25]

Розробка такого «проміжного» рішення, яке б надавало аналітичну потужність рівня Enterprise за доступністю для малого і середнього бізнесу, є актуальною науково-практичною задачею.

1.3. Теоретичні основи, математичні моделі та методи прогнозування в управлінні запасами

Для побудови ефективного АМ необхідно спиратися на надійний теоретичний фундамент, який поєднує класичну теорію УЗ з сучасними методами науки про дані.

Управління тисячами SKU вимагає їхньої сегментації для вибору диференційованих стратегій управління. Фундаментальним методом є ABC-аналіз, що базується на принципі Парето. Він класифікує товари за їхнім внеском у валовий прибуток або оборот:

1. Група А: найважливіші товари (близько 20%). Вимагають суворого контролю, точного прогнозування та мінімальних SS.
2. Група В: товари середньої важливості (30% кількості, 15% результату).
3. Група С: малоцінні товари (50% кількості, 5% результату).

Однак, для автоматизації закупівель одного лише вартісного критерію

недостатньо. Важливим є врахування передбачуваності попиту, що реалізується через XYZ-аналіз. Класифікація здійснюється на основі коефіцієнта варіації попиту:

$$\nu = \frac{\sigma}{\bar{x}} \times 100\%$$

де σ — середньоквадратичне відхилення, $\bar{x}$— середнє значення попиту за період.

4. Група X (ν <= 10-20%): стабільний попит, висока точність прогнозу.
5. Група Y (10% < ν < 25%): попит з тенденціями або сезонністю.
6. Група Z (ν > 25%): нерегулярний, стохастичний попит. [17]

Поєднання цих підходів у матрицю ABC-XYZ дозволяє розробити точні алгоритми для кожної клітинки матриці. [26, 27] Наприклад, товари AX та BX є ідеальними кандидатами для автоматичного поповнення за системою «точно в строк», тоді як товари CZ та CY потребують інших підходів, оскільки стандартні методи для них не працюють. Для товарів групи Z часто доцільно тримати мінімальний запас або працювати «під замовлення», щоб уникнути заморожування коштів у неліквідах.[27-29]

Основою математичних моделей УЗ є формула EOQ. Вона визначає оптимальний обсяг закупівлі, який мінімізує сумарні витрати на оформлення замовлення та зберігання товару. Проте вона має обмеження: вона припускає, що попит є постійним і детермінованим, ціни стабільними, а дефіцит неприпустимим.[30, 31] У реальності попит та час виконання замовлення є випадковими.

Тому сучасні системи використовують стохастичні модифікації EOQ, які вводять поняття «точки перезамовлення» та «страхового запасу». Точка перезамовлення визначається як рівень запасу, при якому необхідно ініціювати нове замовлення, щоб воно надійшло до моменту вичерпання поточного запасу.

Для випадку, коли попит та час виконання замовлення розподілені за нормальним законом, формула розрахунку страхового запасу має вигляд [32]:

$$SS = Z_\alpha \cdot \sqrt{\left(\sigma_D^2 \cdot \underline{L}\right) + \left(\underline{D}^2 \cdot \sigma_L^2\right)}$$

де:

- Z_α ·— квантиль нормального розподілу, що відповідає бажаному рівню сервісу (наприклад, 1.65 для 95% ймовірності відсутності дефіциту);
- σ_D^2 — стандартне відхилення попиту;
- $\underline{L}$ — середній час виконання замовлення;
- $\underline{D}^2$ — середній попит за одиницю часу;
- σ_L^2 — стандартне відхилення часу виконання замовлення.

Вона враховує невизначеність попиту і ненадійність постачальника. [34] Припущення про нормальний розподіл попиту не завжди виконується, особливо для товарів групи Z. У таких випадках необхідно використовувати інші розподіли або імітаційне моделювання.

Точність розрахунку SS та EOQ залежить від точності прогнозу попиту. Простежується еволюція методів прогнозування від простих статистичних до складних нейромережевих підходів.

1. Класичні статистичні методи.

1.1. Ковзні середні та експоненційне згладжування - ефективні для стабільних рядів. Метод Хольта-Вінтерса дозволяє враховувати лінійний тренд та сезонність.

1.2. ARIMA - інструмент для стаціонарних часових рядів. Проте, погано справляється з нелінійними залежностями, складними сезонними патернами та впливом зовнішніх факторів. [35]

2. Сучасні дослідження демонструють перевагу методів МН над класичною статистикою в задачах ритейлу.

2.1. Ансамблеві методи - алгоритми, що базуються на деревах рішень. Демонструють ефективність у роботі з табличними даними. Їхня головна перевага — можливість легко інтегрувати екзогенні фактори: погоду, ціни

конкурентів, маркетингові витрати, день тижня, наявність промо-акцій. Дослідження показують, що XGBoost часто перевершує як ARIMA, так і прості нейронні мережі на структурованих даних продажів, особливо в умовах «холодного старту». [36]

2.2. FE. Успіх моделей МН значною мірою залежить від якості підготовлених ознак. Для ЕК критичними є такі ознаки, як лагові змінні, ковзні вікна, часові ознаки та категоріальні ознаки. [39]

3. Методи глибокого навчання. Для виявлення складних, нелінійних часових залежностей все частіше застосовуються рекурентні нейронні мережі.

3.1. LSTM. Спеціальна архітектура RNN, здатна «запам'ятовувати» інформацію на довгих проміжках часу, що робить її ідеальною для моделювання річної сезонності. LSTM демонструє значно меншу помилку прогнозування порівняно з ARIMA на даних з нелінійною динамікою. [11]

3.2. Гібридні моделі. Останні наукові роботи [43] вказують на високу ефективність гібридних архітектур, наприклад, CNN-LSTM або ARIMA-LSTM. У таких моделях згорткові шари можуть виділяти локальні патерни (короткострокові тренди), а LSTM — довгострокові залежності. Або ж ARIMA моделює лінійну компоненту ряду, а LSTM — нелінійні залишки. Це дозволяє досягти синергії та підвищити точність прогнозу.

4. Специфічні методи для переривчастого попиту. Для товарів групи Z використання стандартних методів призводить до зміщених прогнозів. Тут застосовуються спеціалізовані алгоритми:

4.1. Метод Кростона. Прогнозує окремо інтервали часу між продажами та обсяг продажу, коли він відбувається.

4.2. SBA та TSB. Модифікації методу Кростона, які виправляють його зміщення та дозволяють оновлювати ймовірність попиту в періоди його відсутності.[46]

Реалізація цих методів доступна у спеціалізованих Python-бібліотеках (skforecast, nixtla, intermittent-forecast).

Вибір правильної метрики є критичним для коректного навчання моделей.

Традиційна метрика MAPE є непридатною для ЕК через наявність нулів у продажах та асиметричність.

Сучасним стандартом у прогнозуванні продажів вважається метрика MASE. Вона є масштабо-незалежною і порівнює помилку моделі з помилкою «наївного» прогнозу (де прогноз на завтра дорівнює значенню сьогодні). [18]

$$MASE = \frac{MAE}{MAE_{naiver}}$$

Значення MASE < 1 свідчить про те, що модель працює краще за наївний підхід.

1.4. Обґрунтування архітектурних рішень та технологічного стеку

Для реалізації розроблених моделей необхідно обрати архітектуру, що відповідає вимогам масштабованості, надійності та можливості інтеграції.

Традиційні вебдодатки ЕК часто будуються як моноліти. Однак, інтеграція складних обчислювальних задач безпосередньо в моноліт є неефективною. Задачі МН є ресурсоємними і можуть блокувати обробку HTTP-запитів користувачів. Крім того, екосистема PHP значно поступається Python у наявності бібліотек для аналізу даних.

Оптимальним рішенням є мікросервісна архітектура. Вона передбачає винесення АМ в окремий сервіс, який взаємодіє з основним сервісом через API.

Переваги:

1. Технологічна гетерогенність**.** Основна платформа ІМ може працювати на PHP або Node.js, забезпечуючи швидкий рендеринг сторінок, тоді як АПМ реалізується на Python для доступу до бібліотек Pandas, Scikit-learn, TensorFlow. [48]

2. Незалежне масштабування. Під час перенавчання моделей можна

виділяти додаткові ресурси лише для сервісу МН, не перевантажуючи основний сервер ІМ. [47]

3. Відмовостійкість. Збій у системі прогнозування не призведе до падіння всього ІМ.

Для забезпечення роботи АМ в реальному часі або в режимі, наближеному до нього, необхідно спроектувати ефективний конвеєр обробки даних.

Архітектура повинна включати:

1. Рівень збору та завантаження даних - збір даних про транзакції, перегляди товарів, зміни залишків. Це може реалізовуватися через подієво-орієнтовану модель з використанням брокерів повідомлень або через періодичні ETL-процеси. [64]

2. Рівень збереження даних**:** зберігання історичних даних. Для часових рядів ефективним є використання спеціалізованих БД або хмарних сховищ. [49]

3. Рівень трансформації та аналізу даних - Python-сервіс, що виконує попередню обробку даних, навчання моделей та генерацію прогнозів. [23]

4. Рівень надання даних - надання результатів прогнозу через REST API для відображення в панелі адміністратора магазину. [51]

1.5. Постановка задачі дослідження

Існує об'єктивна необхідність у створенні доступного, гнучкого та наукоємного інструменту для УЗ, який би заповнив нішу між простими Excel-таблицями та дорогими Enterprise-системами. Метою, такого кроку, є підвищення економічної ефективності функціонування ІМ. Він має забезпечувати автоматизоване прогнозування попиту з використанням гібридних моделей МН та розрахунок оптимальних параметрів поповнення запасів в умовах невизначеності.

Для реалізації цього необхідно вирішити наступні задачі:

1. Розробити методику попередньої обробки даних, включаючи роботу з викидами та пропусками.

2. Розробити комбіновану математичну модель прогнозування, яка поєднує методи кластеризації та глибокого навчання для передбачення попиту. Провести порівняльний аналіз точності моделей за метрикою MASE.

3. Формалізувати алгоритм прийняття рішень щодо поповнення запасів, який базується на динамічному розрахунку SS та точки перезамовлення з урахуванням ABC-XYZ класифікації товарів.

4. Спроектувати архітектуру системи, обґрунтувати вибір технологічного стеку та розробити протокол взаємодії між модулями.

5. Розробити програмний прототип модуля.

Об'єктом дослідження виступають процеси УЗ та прогнозування споживчого попиту в системах ЕК. В якості предмета - методи, математичні моделі, алгоритми МН та програмні архітектурні рішення для оптимізації рівня запасів та мінімізації логістичних витрат ІМ.

Задачу УЗ для однієї товарної позиції можна сформулювати як задачу стохастичної оптимізації. Необхідно знайти таку стратегію управління (Q_t, R_t), яка мінімізує математичне сподівання сукупних витрат E на плановому горизонті T:

$$E = \sum_{t=1}^{T} E[C_0 \cdot \delta(u_t) + C_h \cdot max(0, I_t) + C_s \cdot max(0, -I_t)] \rightarrow min$$

де:

- u_t — обсяг замовлення в момент часу t;
- $\delta(u_t)$ — індикаторна функція (дорівнює 1, якщо $u_t > 0$, і 0 в іншому випадку);
- C_o — вартість оформлення одного замовлення;
- C_h — витрати на зберігання одиниці товару за період;
- I_t — рівень запасу в кінці періоду t;
- C_s — штраф за дефіцит або втрачена вигода.

Рівняння балансу запасів:

$$I_t = I_{t-1} + u_{t-L} - D_t$$

де:

- L — час затримки поставки;
- D_t — стохастичний попит у періоді t, прогноз якого отримується за допомогою розробленої ML-моделі $F(X_t)$.

Модель передбачає: схему з втраченими продажами або затримкою виконання; періодичний перегляд запасів.

Отже, задача зводиться до побудови такої функції прогнозування $F(X_t)$, яка мінімізує помилку прогнозу, та розробки правила управління u_t, яке на основі цього прогнозу мінімізує функцію витрат.

Висновки до розділу 1

1. Встановлено, що сучасні ІМ функціонують в умовах високої невизначеності, стикаючись з проблемами ефекту батога, переривчастого попиту та короткого життєвого циклу товарів.
2. Існує суттєвий розрив між потребами малого та середнього бізнесу та наявними рішеннями: існуючі системи або занадто примітивні, або надмірно дорогі та складні. Це обґрунтовує актуальність розробки спеціалізованого АПМ, доступного для інтеграції в поширені вебплатформи.
3. Показано, що методи МН та гібридні моделі мають значний потенціал для підвищення точності прогнозування порівняно з класичними статистичними методами.
4. Мікросервісна архітектура дозволить ефективно поєднати різні технологічні стеки та забезпечити масштабованість системи.
5. Сформульована математична постановка задачі та визначені етапи дослідження для подальшої роботи над проектом.

РОЗДІЛ 2 МЕТОДИ ТА АЛГОРИТМИ ПРОГНОЗУВАННЯ ПОПИТУ

2.1. Математична постановка задачі прогнозування

Розробка АПМ для УЗ в умовах ЕК вимагає формалізації задачі прогнозування як задачі пошуку оптимальної функції відображення історичних даних у майбутні значення попиту. Спираючись на аналіз предметної області, задачу можна класифікувати як задачу регресійного аналізу на часових рядах [11, 15].

Нехай $Y = \{y_t\}_{t=1}^{T}$ — одновимірний часовий ряд, що описує історію продажів конкретної товарної позиції, де $y_t \in R_{\geq 0}$ — обсяг попиту (кількість проданих одиниць) у момент часу t, а T — поточний момент часу.

Головною метою є побудова прогнозної моделі $f : X \to Y$, яка дозволяє отримати оцінку майбутніх значень попиту $\hat{y}_{T+h}$ на горизонті прогнозування $h \in \{1, \dots, H\}$.

Математично задачу можна сформулювати як мінімізацію емпіричного ризику. Необхідно знайти параметри $\hat{\theta}$ моделі *f*, які мінімізують функцію втрат L на навчальній вибірці:

$$\hat{\theta} = arg\ arg\ min_{\theta} \sum_{t=1}^{T-H} L(y_{t+h}, f(x_t;\ \theta)) + \lambda\Omega(\theta)$$

де:

- x_t — вектор ознак, сформований на основі інформації, доступної в момент часу t. Цей вектор включає авторегресійні компоненти, похідні ознаки (ковзні середні, волатильність) та календарні атрибути;
- $L(y, \hat{y})$ — функція втрат, що визначає метрику розбіжності між фактичним y та прогнозованим $\hat{y}$ значенням. Враховуючи специфіку переривчастого попиту та проблему «холодного старту», класична квадратична

функція втрат може бути недостатньою, тому в системі також застосовуються метрики абсолютної помилки та симетричної середньої абсолютної відсоткової помилки; [12, 13]

- $\Omega(\theta)$ — регуляризаційний доданок, що запобігає перенавчанню моделі, обмежуючи складність алгоритму;
- λ — гіперпараметр регуляризації.

Важливою особливістю даної постановки є те, що прогноз попиту y_{t+h} не є кінцевою метою, а слугує вхідним параметром для стохастичної задачі оптимізації запасів. Цільова функція загальної системи УЗ спрямована на мінімізацію сукупних витрат ТС:

$$E = E[C_{order} + C_{holding} + C_{stockout}] \rightarrow \min$$

де $C_{stockout}$ — витрати, пов'язані з дефіцитом, які прямо залежать від похибки прогнозування. Таким чином, підвищення точності моделі *f* (зменшення дисперсії помилки прогнозу) дозволяє зменшити необхідний рівень SS, не знижуючи при цьому рівень обслуговування клієнтів.

Для розробки систем слід реалізовати підхід «прямий багатокроковий прогноз» або рекурсивне прогнозування, де модель послідовно генерує прогнози на кожен наступний крок горизонту H, використовуючи попередні прогнози як вхідні дані для майбутніх лагів, або ж навчається прогнозувати конкретний горизонт безпосередньо. Для реалізації прогнозу в АПМ слід використати підхід, при якому ознаки генеруються для майбутніх дат, а модель, навчена на історичних патернах, застосовується до цього синтетичного майбутнього набору даних $X_{forecast}$.

2.2. Інженерія ознак для часових рядів

Ефективність застосування алгоритмів МН, таких як градієнтний бустінг, до задач часових рядів сильно залежить від якості FE. [48] На відміну від

класичних статистичних методів, які явно моделюють структуру ряду (тренд, сезонність), алгоритми МН вимагають трансформації часової залежності у табличний формат «об'єкт-ознака». [36, 42]

Пропонується комплексний підхід до генерації ознак, який дозволяє моделі «бачити» часову структуру даних. Окрема функція буде формувати три ключові групи предикторів.

1. Лагові ознаки - представляють собою значення цільової змінної зі зсувом у минуле. Вони дозволяють моделі враховувати автокореляцію часового ряду. Математично вектор лагових ознак для моменту *t* можна записати як:

$$x_{lag}^{(t)} = [y_{t-L_1}, y_{t-L_2}, \dots, y_{t-L_k}]$$

У системі обрано набір лагів L ∈ {7, 14, 28} днів. Вибір саме цих значень обумовлений специфікою ЕК [15, 24]:

- L = 7 - дозволяє врахувати тижневу сезонність (різниця між продажами у будні та вихідні). [43] Значення попиту «тиждень тому» часто є найкращим наївним предиктором для поточного дня.
- L = 14 та L = 28 - дозволяють моделі ідентифікувати більш стійкі патерни та циклічність, характерну для місячних періодів (наприклад, дні виплати зарплати).

Реалізація в коді використовуватиме групування за SKU, що є важливим, оскільки модель навчається на даних багатьох товарів одночасно.

2. Ковзні віконні статистики. Лагові значення можуть бути зашумленими. Для виділення трендів та оцінки волатильності використовуються агреговані статистики на ковзних вікнах. У системі має бути реалізовано розрахунок двох основних статистик:

2.1. Ковзне середнє - слугує індикатором локального рівня попиту. Вона згладжує короткострокові флуктуації.

2.2. Ковзне стандартне відхилення - є важливою для оцінки невизначеності

та гетероскедастичності ряду. Вона дозволяє моделі адаптуватися до періодів високої та низької волатильності продажів.

Запобігання витоку даних. В АПМ застосовано важливий технічний прийом — зсув даних перед розрахунком віконних функцій. [39, 40] Це гарантує, що для розрахунку статистик на момент часу *t* використовуються дані лише до моменту *t-1* включно. Ігнорування цього кроку призвело б до використання значення y_t для прогнозування самого себе, що дало б нереалістично високу точність на навчанні та провал на реальних даних.

3. Календарні ознаки. Поведінка споживачів в ЕК має чітко виражену сезонність. [11, 41] Для того, щоб моделі на основі дерев рішень могли ефективно працювати з часовими залежностями, дату розкладають на детерміновані компоненти. У системі мають генеруватися наступні ознаки:

- Dayofweek - день тижня (0-6), визначає тижневий цикл;
- Month - місяць року (1-12), визначає річну сезонність;
- Day - день місяця;
- Weekofyear - порядковий номер тижня;
- is_weekend - бінарна ознака, що вказує на вихідний день. Це спрощує дереву рішень задачу розбиття, дозволяючи явно розділити патерни поведінки у робочі та вихідні дні.

Такий набір ознак формує багатовимірний простір, у якому алгоритми МН здатні знаходити складні нелінійні залежності, недоступні для класичних методів експоненційного згладжування.

2.3. Алгоритми градієнтного бустінгу

Основою АПМ є використання ансамблевих методів на базі дерев рішень. Ці методи демонструють найкращу точність для табличних даних із гетерогенними ознаками. У АПМ застосовано три алгоритми: XGBoost, LightGBM та Random Forest.

XGBoost — це масштабована система МН для градієнтного бустінгу. Ідея

методу полягає у послідовному навчанні ансамблю слабких моделей, де кожна наступна модель намагається скоригувати помилки попередньої. [52]

Прогноз моделі на кроці K визначається як сума прогнозів окремих дерев:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \qquad f_k \in F$$

Ключовою особливістю XGBoost, яка забезпечує його високу ефективність, є специфічна цільова функція, що включає втрати та регуляризацію:

$$L(\varphi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

де $\Omega(f_k) = \gamma T + \frac{1}{2}\lambda\|\omega\|^2$ – регуляризаційний доданок, який штрафує модель за складність

На відміну від класичного градієнтного бустінгу, XGBoost використовує апроксимацію другого порядку функції втрат (ряд Тейлора), використовуючи як градієнт, так і гессіан (друга похідна). Це забезпечує швидшу збіжність та більш точне визначення структури дерева.

Використання XGBoost дозволяє ефективно обробляти розріджені дані, характерні для продажів окремих товарних позицій.

LightGBM — це високоефективна реалізація градієнтного бустінгу, розроблена Microsoft, яка оптимізована для швидкості та ефективності використання пам'яті. [53] В контексті системи, яка має обробляти тисячі товарів, швидкість навчання є критичним фактором.

Основні алгоритмічні відмінності LightGBM, що використані в роботі:

1. GOSS (Gradient-based One-Side Sampling) - метод семплювання даних, який зберігає екземпляри з великими градієнтами (великими помилками)

і випадковим чином відкидає частину екземплярів з малими градієнтами. Це дозволяє зосередити навчання на «складних» випадках, не втрачаючи загальної точності.

2. EFB (Exclusive Feature Bundling) - метод зменшення розмірності, який об'єднує взаємовиключні ознаки в одну щільну ознаку.

3. Leaf-wise Growth - стратегія росту дерева «по листках» (на відміну від «по рівнях» у більшості алгоритмів). LightGBM обирає листок, розбиття якого дає максимальне зменшення функції втрат. Це дозволяє будувати глибші та складніші дерева, що підвищує точність, хоча й потребує обережності з перенавчанням.

LightGBM планується використовувати як частину ансамблю та як окрему модель, яка демонструє високу швидкість навчання.

Random Forest — алгоритм беггінгу, який будує множину незалежних дерев рішень паралельно. [37, 38] На відміну від бустінгу, де дерева залежні, тут кожне дерево навчається на випадковій підвибірці даних.

Прогноз Random Forest для задачі регресії є усередненням прогнозів окремих дерев:

$$\hat{y} = \frac{1}{N} \sum_{n=1}^{N} f_n(x)$$

Включення Random Forest до АПМ обґрунтоване його здатністю зменшувати дисперсію моделі. Оскільки окремі дерева рішень схильні до перенавчання та мають високу дисперсію, їх усереднення дозволяє отримати стійкий та робастний прогноз. Random Forest менш чутливий до викидів у даних та шуму, що робить його відмінним доповненням до алгоритмів бустінгу в складі ансамблю.

2.4. Ансамблеві методи навчання

Для підвищення точності та стабільності прогнозів в АПМ реалізовано концепцію мета-навчання, яка поєднує сильні сторони окремих алгоритмів. [44, 45] Теоретичним підґрунтям є теза про те, що помилки різних моделей часто є некорельованими, тому їх комбінація дозволяє взаємно компенсувати похибки. У системі реалізовано два підходи: Voting та Stacking.

Voting Ensemble (або Voting Regressor) — це метод, який агрегує прогнози базових моделей шляхом зваженого усереднення. [37, 43, 45] Нехай P_{rf}, P_{xgb}, P_{lgbm} — прогнози моделей Random Forest, XGBoost та LightGBM відповідно. Фінальний прогноз P_{final} обчислюється як:

$$P_{final} = \omega_1 P_{rf} + \omega_2 P_{xgb} + \omega_3 P_{lbgm}$$

де ω_i – ваги моделей, такі що $\sum \omega_i = 1$.

У функції АПМ задано ваги weights=[0.3, 0.4, 0.3]. Такий розподіл надає найбільшу вагу моделі XGBoost, яка емпірично демонструє найвищу точність на подібних задачах, тоді як Random Forest та LightGBM виступають у ролі стабілізаторів. Це дозволяє згладити можливі «викиди» у прогнозах бустінгу та зробити загальну систему більш надійною.

Stacking — це більш складний метод ансамблювання, який використовує дворівневу структуру. [45] Замість ручного підбору ваг, ми навчаємо окрему модель (мета-модель), яка вчиться оптимально комбінувати прогнози базових моделей.

Процес навчання виглядає наступним чином:

1. Рівень 0. Базові моделі навчаються на тренувальних даних.
2. Генерація мета-ознак. За допомогою крос-валідації генеруються прогнози цих моделей для навчальної вибірки. Це дозволяє уникнути витоку даних, коли модель прогнозує дані, які вона вже бачила.

3. Рівень 1. Отримані прогнози використовуються як вхідні ознаки для навчання мета-моделі. Цільовою змінною залишаються істинні значення продажів.

У системі в якості мета-моделі пропонується використати Ridge Regression.

$$min_{\beta} \|y - X_{meta}\beta\|_2^2 + \alpha\|\beta\|_2^2$$

Вибір лінійної моделі для мета-рівня є стандартною практикою, оскільки вона дозволяє інтерпретувати внесок кожної базової моделі та уникнути перенавчання на мета-рівні. Stacking дозволяє моделі автоматично визначати, якому алгоритму довіряти більше в залежності від патернів у даних.

2.5. Метрики оцінки якості прогнозів

Для об'єктивної оцінки ефективності розроблених алгоритмів та моніторингу їх роботи у реальному часі в системі має бути реалізовано модуль розрахунку метрик. Результати зберігаються в БД для подальшого аналізу та відображення на дашборді.

1. MAE - середня абсолютна похибка [6, 15]:

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

Вона легко інтерпретується (похибка в одиницях товару) і є стійкою до викидів.

2. RMSE - квадратична похибка:

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

Вона сильніше штрафує модель за великі відхилення. RMSE відіграє подвійну роль: як метрика якості та як параметр для розрахунку SS [32-34].

3. MAPE.

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

Класична метрика бізнесу. Проте вона має суттєвий недолік в умовах розрідженого попиту: ділення на нуль, якщо $y_i = 0$.

4. SMAPE. Враховуючи проблематику ЕК, де нульові продажі є частим явищем, необхідно реалізовати функцію [19]:

$$SMAPE = \frac{100\%}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{(|y_i| + |\hat{y}_i|)/2}$$

SMAPE є симетричною, обмежена діапазоном 0-200% і не страждає від проблеми ділення на нуль (за умови, що прогноз і факт не дорівнюють нулю одночасно). [19] Це робить її основною метрикою для порівняння моделей на різнорідному асортименті. Порівняння метрик, використаних у АПМ наведено в додатку В

2.6. Алгоритми оптимізації управління запасами

Кінцевою метою прогнозування є прийняття ефективних рішень щодо УЗ. Модуль має реалізовувати перехід від предиктивної аналітики до

прескриптивної, розраховуючи конкретні параметри замовлення [1, 2].

SS необхідний для покриття невизначеності попиту та варіативності часу постачання, щоб забезпечити цільовий рівень обслуговування клієнтів. [39, 40] АПМ має реалізувати розрахунок за формулою:

$$SS = Z_\alpha \cdot \sigma_{LT}$$

де:

- Z_α — квантиль нормального розподілу, що відповідає заданому рівню сервісу;
- σ_{LT} — стандартне відхилення попиту за час виконання замовлення L.

Інновація в реалізації: замість використання простого історичного відхилення, система використовує похибка прогнозу моделі (RMSE) як оцінку невизначеності [34]:

$$\sigma_{LT} \approx RMSE \cdot \sqrt{L}$$

Це дозволяє динамічно зменшувати страховий запас, якщо точність моделі прогнозування висока, вивільняючи обігові кошти.

Політика безперервного огляду передбачає постійний моніторинг рівня запасів. Замовлення фіксованого розміру Q (зазвичай EOQ) ініціюється, коли наявний запас падає до точки перезамовлення.

Розрахунок ROP виконується як:

$$ROP = (\overline{D} \cdot L) + SS$$

де $\overline{D}$ — середній прогнозований попит на день, L — час доставки.

Логіка прийняття рішення:

1. Якщо current_stock ≤ rop, генерується рекомендація на замовлення.

2. Кількість замовлення дорівнює eoq.

3. Якщо виявлено дефіцит (current_stock ≤ ss), замовлення збільшується для покриття «боргу»: max(EOQ, ROP + EOQ - current_stock).

Політика періодичного огляду передбачає перевірку запасів через фіксовані інтервали часу R. При кожній перевірці формується замовлення змінного розміру, щоб підняти рівень запасу до цільового рівня S.

Цільовий рівень розраховується так, щоб покрити попит на час доставки L та час до наступної перевірки R:

$$S = \left(\overline{D} \cdot (L + R)\right) + SS$$

Логіка рішення:

1. Якщо current_stock < target_level (S), генерується замовлення.

2. Кількість замовлення Q = S - current_stock.

Ця політика зручна для агрегації замовлень та зменшення адміністративних витрат на оформлення.

Для визначення оптимального розміру замовлення Q у політиці (s, Q) використовується модель Вілсона (додаток Б). [30, 31] Вона знаходить баланс між вартістю оформлення замовлення (C_o) та вартістю зберігання (C_h).

Формула реалізована у вигляді:

$$EOQ = \sqrt{\frac{2 \cdot D_{annual} \cdot C_{ordering}}{C_{holding}}}$$

У системі пропонується використовувати економічні константи: ORDERING_COST = 50, HOLDING_COST = 0.5. Річний попит D_{annual} екстраполюється з середньоденного прогнозу моделі. Це гарантує, що розмір партії є економічно обґрунтованим.

Висновки до розділу 2

Розроблено методологічний та алгоритмічний апарат АМ.

1. Обґрунтовано використання методів FE для адаптації часових рядів до алгоритмів навчання з учителем.
2. Запропоновано алгоритми градієнтного бустінгу та Random Forest, а також механізми їх ансамблювання, що дозволяє досягти високої точності прогнозування та стійкості до шуму.
3. Впроваджено спеціалізовану метрику SMAPE для коректної оцінки якості моделей в умовах переривчастого попиту.
4. Розроблено алгоритми оптимізації запасів, які базуються на динамічному розрахунку SS з використанням похибки прогнозу. Це забезпечує адаптивність системи УЗ до зміни ринкових умов та точності прогнозів.

Запропонований комплекс алгоритмів створює надійну основу для функціонування системи УЗ ІМ.

РОЗДІЛ 3. ПРОЄКТУВАННЯ АРХІТЕКТУРИ АПМ

3.1. Вимоги до функціоналу системи

Проєктування АПМ для вебдодатку ІМ розпочинається з формалізації вимог до системи. Згідно з методологією IEEE 830-1998 [54], вимоги поділяються на функціональні та нефункціональні.

Функціональні вимоги визначають конкретні дії, які система повинна виконувати для досягнення поставлених бізнес-цілей. Усі функціональні вимоги до системи наведено в додатку Г.

Згідно з дослідженням Sommerville [55], пріоритизація вимог за методом «Має бути / Має бути бажано / Може бути / Не буде» дозволяє ефективно розподілити ресурси розробки. В АПМ критичними є вимоги FR-1, FR-3, FR-4, FR-5, FR-7, FR-8, FR-12, оскільки вони забезпечують базовий функціонал системи прогнозування та оптимізації запасів.

Нефункціональні вимоги визначають якісні характеристики системи та обмеження на її функціонування [56]. Вони класифікуються за категоріями згідно ISO/IEC 25010:2011 [57].

Продуктивність:

1. NFR-1. Час відгуку API для отримання прогнозів повинен становити не більше 500 мс для 95-го перцентиля запитів при навантаженні до 100 одночасних користувачів. За даними [58], затримка відгуку більше 500 мс призводить до зниження користувацької задоволеності на 20%.

2. NFR-2. Час тренування моделі для одного SKU не повинен перевищувати 30 секунд на стандартному обчислювальному вузлі.

3. NFR-3. Система повинна підтримувати одночасне тренування до 10 моделей без деградації продуктивності.

Точність та надійність:

1. NFR-4. Середня абсолютна процентна похибка прогнозів не повинна перевищувати 20% для товарів з регулярними продажами. Згідно з дослідженням

Syntetos [59], MAPE ≤ 20% вважається прийнятним для прогнозування показників роздрібної торгівлі на майбутнє.

2. NFR-5. Системна доступність повинна становити не менше 99% протягом місяця, що відповідає максимальному часу простою 7.3 години на місяць.

3. NFR-6. Втрата даних повинна бути виключена через використання транзакційної БД з механізмами резервного копіювання.

Масштабованість:

1. NFR-7. Система повинна підтримувати одночасну роботу з кількістю SKU до 10,000 без зниження продуктивності.

2. NFR-8. Архітектура повинна забезпечувати можливість горизонтального масштабування для обробки зростаючого навантаження.

Безпека:

1. NFR-9. API повинен використовувати автентифікацію на основі токенів згідно RFC 7519 [60].

2. NFR-10. Всі персональні дані користувачів повинні зберігатись у зашифрованому вигляді відповідно до вимог GDPR [61].

3. NFR-11. Система повинна логувати всі операції з критичними даними для аудиту.

Сумісність:

1. NFR-12. Вебінтерфейс повинен коректно відображатись у браузерах останніх двох версій.

2. NFR-13. API повинен відповідати специфікації OpenAPI 3.0 [62] для забезпечення інтероперабельності.

Супроводжуваність:

1. NFR-14. Код повинен відповідати стандартам PEP 8 [63] для Python та мати покриття тестами не менше 80%.

2. NFR-15. Система повинна включати детальну технічну документацію та інструкції з розгортання.

При проєктуванні системи враховуються наступні технічні та

організаційні обмеження:

1. Технологічні обмеження - використання виключно відкритих бібліотек для забезпечення відсутності ліцензійних витрат.
2. Апаратні обмеження - система повинна функціонувати на стандартному серверному обладнанні без потреби в GPU-прискорювачах.
3. Бюджетні обмеження.

3.2. Вибір технологічного стеку

Вибір технологічного стеку є важливим кроком, який впливає на продуктивність, масштабованість та вартість підтримки системи [64]. Рішення базується на багатокритеріальному аналізі альтернатив.

Для оцінки технологій використовується зважена система критеріїв (додаток Д).

Для backend-розробки розглянуто чотири альтернативи: Django, FastAPI, Flask та Spring Boot (Java). Результати порівняльного аналізу наведено в додатку Е.

Обрано Django через найвищу швидкість розробки, наявність ORM та готової адміністративної панелі. Незважаючи на нижчу продуктивність порівняно з FastAPI, Django забезпечує оптимальний баланс для прототипування. Згідно з дослідженням JetBrains Python Developers Survey 2023 [65], Django використовується в 43% Python веб-проектів, що свідчить про його індустріальну зрілість.

Для реалізації алгоритмів МН проведено порівняння провідних бібліотек градієнтного бустінгу (додаток Ж).

Система використовує комбінацію бібліотек:

1. XGBoost – як основна модель через найкращий баланс точності та швидкості, а також лідерство в Kaggle змаганнях [66].
2. LightGBM – для підвищення швидкості тренування завдяки алгоритмам GOSS та EFB [67].

3. scikit-learn 1.3.2 RandomForest – як найпростіша модель для порівняння та компонент ансамблю.

Теоретичне обґрунтування переваг градієнтного бустінгу над іншими методами для табличних даних наведено в роботі [68].

Для збереження даних розглянуто реляційні та NoSQL бази даних (додаток З).

Обрано PostgreSQL через:

1. Повну підтримку ACID властивостей для забезпечення консистентності прогнозів та рекомендацій.

2. Розширені можливості індексації, критичні для швидкого пошуку часових рядів.

3. Нативну підтримку jsonb для зберігання метаданих моделей.

4. Повна сумісність з Django ORM.

Згідно Stack Overflow Developer Survey 2023 [69], PostgreSQL є найпопулярнішою реляційною СУБД серед професійних розробників (45.4%).

Для frontend реалізовано підхід прогресивного покращення, який забезпечує доступність базового функціоналу навіть у мінімально підтримуваних браузерах, а також дозволяє додавати інші можливості для сучасних браузерів без втрати працездатності системи. [70]:

Базовий шар:

- Bootstrap 5.3 – CSS framework для швидкого прототипування інтерфейсу;
- HTML5/CSS3 – семантична розмітка.

Інтерактивність:

- Alpine.js 3.13 – мінімалістичний реактивний framework (15 KB gzipped) як альтернатива React/Vue для простих інтерфейсів; [71]
- HTMX 1.9 – бібліотека для AJAX без написання JavaScript.

Візуалізація:

- Plotly.js 2.27 – інтерактивні графіки прогнозів з довірчими інтервалами;

- Chart.js 4.4 – діаграми для порівняння метрик моделей.

Відмова від SPA (Single Page Application) фреймворків (React, Vue, Angular) обумовлена:

1. Зниженням складності проєкту (один технологічний стек – Python).
2. Кращою SEO-оптимізацією через серверний рендеринг.
3. Меншим розміром фінального файлу фронтенду, що зменшує час завантаження інтерфейсу та покращує швидкодію вебдодатку (Alpine.js + HTMX = 40 KB vs React = 140 KB)

Серед додаткових інструментів виділимо наступні.

1. Серіалізація моделей – joblib – ефективне збереження scikit-learn сумісних моделей. [72]

2. Обробка даних: pandas– маніпуляція часовими рядами; numpy – векторизовані обчислення

3. API документація: Django REST Framework – автогенерація OpenAPI специфікації.

4. Тестування: pytest – unit та інтеграційні тести; pytest-django – інтеграція з Django.

Згідно з принципом YAGNI [73], мікросервісна архітектура впроваджується лише при наявності:

- команди >5 розробників;
- різних технологічних стеків для різних компонентів;
- необхідності незалежного масштабування.

Для академічного проєкту монолітна архітектура забезпечує:

- швидшу розробку (1 репозиторій, 1 процес деплою);
- простішу налагодження;
- відсутність overhead міжсервісної комунікації.

Хоча LSTM та Transformer архітектури показують кращі результати на довгих часових рядах (>1000 точок) [74], для прогнозування показників роздрібної торгівлі з типовими рядами 200-500 точок градієнтний бустінг демонструє:

- вищу точність на малих датасетах;
- швидше тренування;
- менші вимоги до обчислювальних ресурсів.

3.3. Проєктування архітектури додатку

Архітектура системи базується на модульному підході з розділенням відповідальності [75]. Проєкт організовано згідно з best practices фреймворку Django [76] та принципами DDD, що забезпечило чітку структуризацію коду, модульність та зрозумілий розподіл відповідальностей між компонентами АПМ. [77]. Застосування підходу DDD дало змогу виокремити основні бізнес-об’єкти предметної області, сформувати зрозумілу модель домену та організувати взаємодію між модулями на основі бізнес-логіки, а не технічних деталей.

Загальна структура АПМ представлена у додатку Л

Пояснення архітектурних рішень.

1. Налаштування, організовані як пакет: розділення налаштувань на base/development/production дозволяє уникнути хардкодингу секретів та спрощує CI/CD [78].

2. Доменно-орієнтована структура застосунків: кожен додаток представляє окремий обмежений контекст в термінах DDD:

- inventory – управління товарами, історією продажів;
- analytics – ML моделі, прогнози, оптимізація;
- api – REST endpoints;
- dashboard – веб-інтерфейс.

3. Базові утиліти: спільна логіка (timestamped models, pagination mixins) виділена в окремий модуль для повторного використання.

Структура додатку inventory представлена в додатку Н.

Сервісний шар: згідно з принципом єдиної відповідальності [79], бізнес-логіка винесена в окремий services/ модуль. Це забезпечує:

- тестованість (сервіси не залежать від HTTP request);

- повторне використання логіки в API, Celery tasks, management commands.

Структура додатку analytics представлена в додатку О.

Сховище моделей МН: треновані моделі зберігаються в ml_models/saved_models/ з іменуванням {sku}_{model_type}.joblib. Альтернативою є використання реєстра моделей МН у MLflow [80], але для академічного проєкту файлова система є достатньою.

Система побудована на принципах багатошарової архітектури [81] з чітким розділенням відповідальності:

1. Рівень представлення (templates/, static/):
   - відповідальність: рендеринг HTML, JavaScript логіка;
   - залежності - Application Laye.r
2. Рівень додатку (views.py, urls.py):
   - відповідальність: обробка HTTP запитів, валідація форм;
   - залежності - Business Logic Laye.r
3. Рівень бізнес-логіки (services/):
   - відповідальність: бізнес-правила, ML алгоритми;
   - залежності - Data Access Layer;
   - незалежність від Django - сервіси можна використовувати в CLI, Celery tasks.
4. Рівень доступу до даних (models.py, managers.py):
   - відповідальність: робота з БД через Django ORM;
   - залежності – Database.
5. Рівень бази даних. Відповідальність - персистентність даних.

Проєктування схеми БД виконано з дотриманням принципів нормалізації (до 3-ї нормальної форми) для уникнення надлишковості та аномалій даних (додаток И).

1. Сутність Product (Товар). Це центральна сутність системи, навколо якої будуються всі процеси.

2. Сутність SalesRecord (Запис продажу). Представляє часовий ряд

історичних даних. Обмеження: unique_together = ('product', 'date') — гарантує, що для одного товару на одну дату існує лише один запис про продажі. Це спрощує агрегацію та запобігає дублюванню даних при імпорті.

3. Сутність Forecast (Прогноз). Зберігає результати роботи ML моделей. Особливість: Зв'язок Many-to-One з Product дозволяє зберігати історію прогнозів. Композитний ключ або обмеження унікальності на product, forecast_date, model_used дозволяє порівнювати різні моделі для одного періоду.

4. Сутність Recommendation (Рекомендація). Результат роботи модуля оптимізації запасів. Ця сутність пов'язує аналітику з операційними діями (закупівлями).

5. Сутність ModelMetrics (Метрики якості). Службова сутність для MLOps. Дозволяє відстежувати деградацію моделей (Model Drift) з часом.

Всі основні сутності мають зв'язок Foreign Key (One-to-Many) з сутністю Product. Видалення товару (on_delete=models.CASCADE) призводить до видалення всієї пов'язаної історії та прогнозів, що забезпечує цілісність посилань.

Для інтеграції системи в корпоративний ландшафт розроблено RESTful API з використанням Django REST Framework. Архітектура API відповідає принципам REST, сформульованим Роєм Філдінгом. [68]

Ключові архітектурні рішення:

1. Кожна сутність представлена як ресурс з унікальним URI.

2. Використання HTTP методів:

- GET - отримання даних;
- POST - створення нових ресурсів або запуск процесів;
- PUT/PATCH - оновлення даних;
- DELETE: -видалення.

3. Сервер не зберігає стан клієнтської сесії між запитами. Аутентифікація здійснюється через токени або сесії, що дозволяє легко масштабувати сервер горизонтально.

4. Класи-серіалізатори відповідають за перетворення складних об'єктів

ORM у формат JSON та валідацію вхідних даних. Використання ModelSerializer дозволяє автоматично генерувати схему серіалізації на основі моделей Django, дотримуючись принципу DRY.

5. Документування - використання Swagger/OpenAPI (через rest_framework схеми) для автоматичної генерації документації, що спрощує інтеграцію для сторонніх розробників.

Проєктування конвеєра МН є центральним елементом АПМ. Він автоматизує повний цикл роботи з даними: від підготовки до отримання прогнозу та його інтерпретації. Реалізація зосереджена у сервісних шарах додатку analytics.

Етап 1 - попередня обробка даних. Сирі транзакційні дані продажів непридатні для прямого використання алгоритмами МН. Функція prepare_training_data виконує наступні трансформації:

1. Агрегація: продажі агрегуються до денного рівня. Пропуски в датах заповнюються нулями.

2. Обробка викидів: аномальні сплески продажів можуть спотворити навчання.

3. Перевірка стаціонарності: хоча деревні алгоритми не вимагають стаціонарності ряду так суворо, як ARIMA, перевірка та детрендинг можуть покращити результати.

Етап 2 - конструювання ознак. Для перетворення задачі прогнозування часового ряду у задачу керованого навчання, необхідно сформувати матрицю ознак та цільовий вектор. Функція create_features реалізує наступні групи ознак:

1. Лагові змінні: значення продажів за минулі періоди . Вони дозволяють моделі «бачити» історію та автокореляцію.

2. Ковзні вікна: середнє та стандартне відхилення за ковзним вікном (7, 28 днів). Це згладжує шум та виявляє локальні тренди.

3. Календарні ознаки: день тижня, місяць, день року, прапор вихідного дня (is_weekend). Це критично для виявлення тижневої та річної сезонності.

Етап 3 - моделювання та ансамблювання. Система реалізує підхід

ансамблеве навчання, який базується на поєднанні прогнозів декількох «слабких» або різнорідних моделей для отримання більш точного та стійкого результату.

В модулі forecasting.py:

1. Базові моделі. Random Forest - алгоритм беггінгу, що будує множину незалежних дерев рішень. XGBoost / LightGBM - алгоритми бустингу, що будують дерева послідовно, де кожне наступне виправляє помилки попереднього. Вони зменшують зміщення і забезпечують найвищу точність.

2. Стратегії ансамблювання. Voting Regressor - прогноз розраховується як зважене середнє прогнозів базових моделей. Це дозволяє компенсувати помилки окремих алгоритмів. Stacking Regressor - більш складний підхід, де прогнози базових моделей використовуються як вхідні ознаки для мета-моделі. Мета-модель навчається оптимально комбінувати прогнози першого рівня.

Етап 4 - оцінка та метрики. Для оцінки якості використовуються наступні метрики, реалізовані в calculate_metrics:

1. MAE - легко інтерпретується.

2. RMSE - чутлива до великих помилок. Використовується для розрахунку SS.

3. MAPE - відносна помилка. Бізнес-стандарт для оцінки точності (≤ 20% вважається добрим результатом для роздрібної торгівлі).

4. SMAPE - використовується, коли в даних є нулі або значення близькі до нуля, де звичайна MAPE стає нестабільною.

Етап 5 - оптимізація запасів. Цей етап перетворює прогноз на бізнес-рішення. Модуль optimization.py реалізує розрахунок параметрів на основі ймовірнісних моделей.

1. SS.

2. Точка перезамовлення для політики (s, Q).

3. Цільовий рівень для політики (R, S).

Така інтеграція МН та теорії УЗ дозволяє мінімізувати сумарні логістичні витрати, забезпечуючи при цьому необхідний рівень доступності товару для

клієнта.

3.5. UML діаграми

Для візуалізації структури та поведінки спроєктованої системи використано уніфіковану мову моделювання UML (додаток К). Діаграми допомагають формалізувати вимоги та слугують документацією для розробників.

Діаграма прецедентів відображає функціональні можливості системи з точки зору взаємодії з акторами.

Актори:

1. Менеджер із закупівель - внутрішній користувач, відповідальний за прийняття рішень.
2. Зовнішня ERP-система - програмний агент, що взаємодіє через API.

Основні прецеденти:

1. Блок «Управління даними». Завантажити дані продажів: менеджер завантажує CSV файл. Система включає сценарій «Валідація даних». «Переглянути каталог»: перегляд списку та деталей товарів.
2. Блок «Прогнозування». Створити прогноз: менеджер ініціює розрахунок. Цей прецедент розширюється опціями вибору моделі та горизонту. «Аналізувати результати»: перегляд графіків та метрик точності.
3. Блок «Оптимізація». «Розрахувати рекомендації»: система автоматично генерує план закупівель на основі прогнозів. «Експортувати замовлення»: вивантаження рекомендацій у файл для постачальників.

Діаграма послідовності деталізує сценарій «Створення прогнозу та генерація рекомендацій», показуючи обмін повідомленнями між об'єктами системи у часі.

1. Користувач відправляє запит POST з параметрами (SKU, Model Type) через веб-інтерфейс.
2. View отримує запит, валідує форму та викликає сервіс

ForecastingService.

3. ForecastingService звертається до ProductRepository (ORM) для отримання історичних даних SalesRecord.

4. ForecastingService передає дані у Preprocessor, який виконує агрегацію та FE.

5. ForecastingService ініціалізує обрану ML Model (наприклад, VotingEnsemble) та викликає метод fit().

6. ML Model повертає навчену модель, яка серіалізується та зберігається.

7. ForecastingService викликає метод predict() для майбутніх дат.

8. ForecastingService зберігає результати у БД через ForecastRepository та повертає статус успіху.

9. View ініціює асинхронний виклив OptimizationService.

10. OptimizationService отримує прогноз, розраховує Safety Stock та ROP, і зберігає Recommendation.

11. View відображає користувачу повідомлення про завершення та оновлену сторінку з результатами.

Діаграма класів описує статичну структуру системи, відображаючи класи, їх атрибути, методи та взаємозв'язки.

1. Рівень моделей:

- Product - містить атрибути sku, current_stock, має методи get_active_promos();
- Forecast - атрибути predicted_quantity, model_used. Асоціація з Product;
- Recommendation - атрибути priority, policy_type, Аасоціація з Product.

2. Сервісний рівень:

- DataImporter: методи handle_uploaded_file(), process_csv();
- ForecastingEngine: клас-фасад для операцій МН. Методи train(), predict(), evaluate(). Використовує патерн Strategy для вибору алгоритму;
- InventoryOptimizer: методи calculate_ss(), generate_order_plan().

3. Рівень API. ProductViewSet, ForecastViewSet: спадкуються від ModelViewSet. Використовують відповідні серіалізатори.

Висновки до розділу 3

Здійснено комплексне проєктування архітектури АПМ для ІМ.

1. Системний аналіз вимог дозволив сформулювати чіткі функціональні специфікації, що покривають повний цикл УЗ: від збору даних та прогнозування до оптимізації закупівель. Особливу увагу приділено підтримці ансамблевих методів МН та ймовірнісних моделей УЗ.

2. Обґрунтування технологічного стеку базується на аналізі сучасних публікацій та вимогах до швидкості розробки і продуктивності. Вибраний стек забезпечує оптимальний баланс між простотою реалізації та потужністю аналітичних інструментів.

3. Архітектура додатку спроєктована за принципом модульного моноліту з чітким розділенням відповідальності між додатками inventory, analytics, api та dashboard. Це забезпечує низьку зв'язність компонентів та спрощує подальшу підтримку і масштабування системи.

4. Проєктування схеми даних та API виконано з урахуванням стандартів нормалізації та принципів REST, що гарантує цілісність даних та легкість інтеграції зі сторонніми сервісами.

5. Розробка ML pipeline є ключовим досягненням етапу проєктування. Інтеграція алгоритмів градієнтного бустингу та механізмів FE дозволяє досягти високої точності прогнозування, а реалізація математичних моделей SS та ROP забезпечує наукову обґрунтованість рішень щодо поповнення запасів.

6. UML моделювання формалізувало логіку роботи системи, створивши надійну основу для етапу програмної реалізації.

Запропонована архітектура повністю відповідає поставленим задачам та створює фундамент для розробки АПМ УЗ.

РОЗДІЛ 4. РЕАЛІЗАЦІЯ АНАЛІТИЧНОГО МОДУЛЯ

4.1. Структура проєкту та організація коду

Організація файлової структури та архітектура коду є фундаментом життєздатності будь-якої складної інформаційної системи. АПМ реалізовано з використанням фреймворку Django за принципами модульної архітектури. Структура проєкту організована у вигляді набору взаємопов'язаних додатків, кожен з яких відповідає за окрему функціональну область (додаток Л).

Архітектура проекту базується на патерні MVT, який є адаптацією класичного патерну MVC для веб-середовища, де роль контролера бере на себе сам фреймворк, обробляючи вхідні HTTP-запити та маршрутизуючи їх до відповідних представлень.

Коренева директорія проекту містить конфігураційний модуль demand_forecast_project, який керує глобальними налаштуваннями системи. Файл settings.py визначає ключові параметри середовища виконання, безпеки та підключених компонентів (додаток М).

Використання бібліотеки pathlib для визначення базової директорії BASE_DIR забезпечує кросплатформеність шляхів, що дозволяє розгортати систему як на UNIX-подібних серверах, так і в середовищі Windows без зміни вихідного коду.

Список встановлених застосунків (додаток П) відображає модульну структуру системи. Проект розділено на логічно ізольовані компоненти, що відповідає принципу «Слабка зв'язаність» та «Сильне зчеплення».

Такий поділ дозволяє незалежно розробляти та тестувати окремі частини системи.

Список MIDDLEWARE у settings.py визначає конвеєр обробки запитів. Включення django.middleware.security.SecurityMiddleware та django.middleware.csrf.CsrfViewMiddleware забезпечує базовий захист від поширених веб-вразливостей, таких як міжсайтовий скриптінг та підробка

міжсайтових запитів. Це має значення для системи, що оперує комерційними даними про продажі та запаси. AuthenticationMiddleware та SessionMiddleware забезпечують управління сесіями користувачів, що необхідно для розмежування прав доступу до аналітичних функцій.

На етапі розробки та прототипування використано реляційну базу даних SQLite, сконфігуровану за замовчуванням.

Використання Django ORM дозволяє абстрагуватися від специфічного діалекту SQL. Це означає, що при переході в продуктивне середовище можна легко замінити SQLite на більш потужну СУБД. Це забезпечує високу масштабованість проекту.

Для складної аналітичної системи важливим є наявність детального логування, оскільки процеси навчання моделей можуть тривати довго і виконуватися у фоновому режимі. У конфігурації LOGGING визначено два канали виводу: console (для виводу в термінал під час розробки) та file (для збереження історії подій у файл debug.log).

Спеціалізований логер для модуля analytics налаштований на рівень DEBUG. Це дозволяє розробникам та адміністраторам системи відстежувати деталі виконання алгоритмів: параметри тренування моделей, значення розрахованих метрик помилок, проміжні результати розрахунку страхових запасів тощо. Такий підхід значно спрощує відлагодження та дозволяє швидко реагувати на аномалії в роботі алгоритмів.

Система використовує локалізацію для української мови та часову зону Києва.

4.2. Реалізація модуля підготовки даних

Модуль підготовки даних (inventory/services/data_preprocessing.py) реалізує повний цикл обробки історичних даних продажів для подальшого використання в моделях МН. Ефективність моделей МН безпосередньо залежить

від якості та структури вхідних даних. Модуль підготовки даних реалізує складний ETL-процес, який забезпечує імпорт даних з зовнішніх джерел, їх очищення, валідацію та перетворення у формат, придатний для навчання моделей.

Функціонал завантаження даних зосереджено у файлі inventory/services/data_import.py. Система підтримує імпорт з файлів форматів CSV та Excel, які є де-факто стандартами обміну даними в бізнес-середовищі. Для парсингу файлів використовується бібліотека pandas, що забезпечує високу швидкість обробки навіть великих масивів даних.

Ключовим архітектурним рішенням є використання декоратора @transaction.atomic для функцій імпорту (process_products_data, process_sales_data, process_stock_data).

Цей механізм гарантує атомарність операцій БД. Якщо під час обробки тисячного рядка файлу виникне помилка, вся транзакція буде відкочена. Це запобігає частковому імпорту даних, який міг би призвести до неузгодженості стану системи та критичних помилок у прогнозуванні.

У функції process_sales_data реалізовано стратегію update_or_create. Це дозволяє використовувати один і той самий механізм як для початкового завантаження історії продажів, так і для періодичного оновлення даних, уникаючи дублювання записів. Крім того, реалізовано обробку виключень Product.DoesNotExist: якщо в файлі продажів зустрічається SKU, якого немає в довіднику товарів, система автоматично створює картку товару з базовими атрибутами, логуючи це як попередження. Це підвищує відмовостійкість процесу імпорту.

«Сирі» дані про транзакції продажів зазвичай є нерегулярними та зашумленими. Модуль inventory/services/data_preprocessing.py відповідає за їх перетворення у регулярні часові ряди.

Функція preprocess_and_aggregate виконує наступні кроки:

1. Агрегація за часом. Дані групуються за обраним інтервалом (день, тиждень) за допомогою pd.Grouper. Це дозволяє перейти від рівня окремих чеків

до рівня сумарного попиту за період.

2. Відновлення безперервності. Часові ряди продажів часто містять пропуски. Використання методу resample з подальшим заповненням нулями (fillna(0)) є необхідним, оскільки для моделей регресії відсутність запису про продаж і нульовий продаж — це різні сутності, але в контексті попиту відсутність транзакції означає нульовий попит.

3. Кліпінг. Застосування df['sales'].clip(lower=0) усуває можливі артефакти, такі як від'ємні значення продажів, що можуть виникати при оформленні повернень товару в обліковій системі.

Для застосування алгоритмів керованого навчання до задачі часових рядів необхідно перетворити послідовність значень у набір ознак та цільову змінну. Функція create_features реалізує генерацію трьох груп ознак, які дозволяють моделям «бачити» структуру часового ряду.

1. Календарні ознаки: вилучення компонентів дати (year, month, day, dayofweek, weekofyear) дозволяє моделям, особливо методам на основі дерев рішень, вловлювати сезонні закономірності. Додавання бінарної ознаки is_weekend явно вказує моделі на відмінність поведінки попиту в робочі та вихідні дні.

2. Лагові ознаки. Зсув цільової змінної у минуле (sales_lag_7, sales_lag_14, sales_lag_28) дозволяє моделі враховувати автокореляцію ряду.

```
for lag in lag_days:
    df[f'sales_lag_{lag}'] = df.groupby('sku')['sales'].shift(lag)
```

3. Ковзні статистики. Розрахунок середнього та стандартного відхилення у ковзному вікні дозволяє моделі оцінювати локальний тренд та волатильність попиту, ігноруючи короткочасний шум. Важливим нюансом реалізації є використання shift() перед застосуванням rolling. Це запобігає «витоку даних».

Функція prepare_training_data() інтегрує всі етапи підготовки. Ця функція забезпечує повний пайплайн: вибірка → агрегація → створення ознак → видалення пропусків.

4.3. Реалізація модуля прогнозування

Модуль прогнозування (analytics/services/forecasting.py) є інтелектуальним ядром системи і реалізує повний цикл МН: тренування моделей, оцінку якості, генерування прогнозів та збереження результатів. Його реалізація базується на використанні сучасних бібліотек МН та підтримує гнучку зміну стратегій прогнозування — від простих моделей до складних гетерогенних ансамблів (додаток Р).

Система підтримує навчання трьох типів базових регресійних моделей, реалізованих у функції train_single_model.

1. XGBoost: використовується клас xgb.XGBRegressor. Цей алгоритм обрано за його високу ефективність роботи з табличними даними, вбудовану регуляризацію (L1/L2), що запобігає перенавчанню, та здатність ефективно обробляти пропущені значення. Параметр n_estimators=100 встановлює кількість дерев у ансамблі, що є компромісом між точністю та часом навчання для інтерактивного режиму роботи.

2. LightGBM: реалізовано через lgb.LGBMRegressor. Ключовою відмінністю LightGBM є використання алгоритму навчання на основі гістограм та стратегії росту дерев leaf-wise. Це забезпечує значно вищу швидкість навчання та менше споживання пам'яті, що має значення для обробки великої кількості товарних позицій.

3. Random Forest: використовується RandomForestRegressor з бібліотеки scikit-learn. На відміну від бустингу, який будує дерева послідовно, виправляючи помилки попередників, Random Forest будує незалежні дерева паралельно (Bagging). Ця модель слугує надійним базовим рівнем, оскільки вона

стійка до шуму та менш схильна до перенавчання без ретельного налаштування гіперпараметрів.

Оцінка якості моделей виконується функцією calculate_metrics, яка розраховує набір наступних метрик:

- MAE - легко інтерпретується бізнесом;
- RMSE - сильніше «карає» за великі відхилення, що важливо для виявлення ризиків значних помилок;
- SMAPE - реалізована власною функцією.

Для досягнення максимальної точності прогнозування в системі реалізовано методи ансамблювання моделей. Згідно з теорією МН, комбінація декількох слабких або різнорідних моделей часто дає кращий результат, ніж найкраща з поодиноких моделей, завдяки зменшенню дисперсії помилки.

Функція train_voting_ensemble реалізує підхід Soft Voting для регресії. Створюється гетерогенний ансамбль, що складається з Random Forest, XGBoost та LightGBM.

```
ensemble = VotingRegressor(
   estimators=[('rf', rf), ('xgb', xgb_model), ('lgbm', lgbm_model)],
   weights=[0.3, 0.4, 0.3]
)
```

Використання ваг (weights=[0.3, 0.4, 0.3]) дозволяє надати пріоритет градієнтному бустингу, який емпірично часто демонструє найкращу точність, при цьому зберігаючи стабілізуючий вплив Random Forest та LightGBM. Такий підхід робить прогноз більш стійким до викидів в даних, оскільки помилка однієї моделі може бути скомпенсована точнішими передбаченнями інших.

Функція train_stacking_ensemble реалізує більш складну архітектуру — стекове узагальнення. На відміну від простого голосування, де результати усереднюються, стекінг використовує дворівневу схему навчання:

1. Базовий рівень (Level-0). Моделі (RF, XGB, LGBM) генерують

прогнози на основі вхідних даних.

2. Мета-рівень (Level-1). Мета-модель навчається використовувати прогнози базових моделей як вхідні ознаки для формування фінального прогнозу.

В якості мета-моделі обрано Ridge Regression (лінійна регресія з L2-регуляризацією).

```
stacking = StackingRegressor(
    estimators=base_models,
    final_estimator=Ridge(),
    cv=5
)
```

Використання Ridge замість звичайної лінійної регресії запобігає перенавчанню мета-моделі у випадку, якщо прогнози базових моделей сильно корелюють між собою (мультиколінеарність). Параметр cv=5 вказує на використання 5-блокової крос-валідації при генерації тренувальних даних для мета-моделі, що є критичним для уникнення витоку даних і забезпечення чесної оцінки узагальнюючої здатності ансамблю.

Оскільки навчання ансамблевих моделей є обчислювально затратною операцією, нераціонально проводити її при кожному запиті прогнозу. Реалізовано механізм персистентності моделей з використанням бібліотеки joblib, яка оптимізована для збереження об'єктів NumPy, що містяться всередині моделей scikit-learn та xgboost.

Функції save_model та load_model керують життєвим циклом бінарних файлів моделей у директорії analytics/ml_models/saved_models. Ім'я файлу формується динамічно (f'{sku}_{model_name}.joblib'), що дозволяє підтримувати окремі екземпляри навчених моделей для кожного товару. Це реалізує стратегію «локальних моделей», яка часто перевершує «глобальні моделі» в задачах прогнозування роздрібного попиту, де поведінка різних

товарів може кардинально відрізнятися.

Паралельно зі збереженням бінарного файлу моделі, мета-дані про навчання записуються в реляційну базу даних через модель ModelMetrics. Це створює історію версій моделей і дозволяє реалізувати моніторинг «дрейфу моделей» — ситуації, коли якість прогнозування погіршується з часом через зміни в розподілі даних, сигналізуючи про необхідність перенавчання.

4.4. Реалізація модуля оптимізації запасів

Модуль оптимізації (analytics/services/optimization.py) виконує перехід від предиктивної аналітики до прескриптивної. Він трансформує ймовірнісні прогнози попиту в конкретні параметри УЗ: точку перезамовлення, страховий запас та обсяг замовлення (додаток С).

SS необхідний для покриття невизначеності попиту та варіативності часу постачання. Для його розрахунку використано класичну формулу, що базується на припущенні про нормальний розподіл помилок прогнозу.

Важливим нововведенням у реалізації є підхід до визначення σ . Замість використання історичного стандартного відхилення продажів (яке включає в себе тренди та сезонність), система використовує RMSE навченої моделі МН як оцінку стандартного відхилення помилки прогнозу:

```
error_std = float(metrics.rmse)
safety_stock = Z_SCORE * error_std * np.sqrt(lead_time)
```

Такий підхід, описаний у літературі як «Dynamic Safety Stock» [10], дозволяє значно зменшити рівень SS, оскільки модель МН пояснює частину варіативності, залишаючи в «помилці» лише справді випадковий шум.

Система підтримує дві фундаментальні політики поповнення запасів, вибір між якими реалізовано у функції generate_inventory_recommendations:

Функція calculate_eoq реалізує формулу Вілсона, яка знаходить баланс між

вартістю оформлення замовлення (ORDERING_COST) та вартістю зберігання (HOLDING_COST).

*eoq = np.sqrt((2 * annual_demand * ordering_cost) / annual_holding_cost)*

Це дозволяє мінімізувати сумарні логістичні витрати.

Система не просто видає список товарів для замовлення, а й присвоює їм пріоритет. Логіка визначення пріоритету базується на критичності залишків: якщо поточний запас менший за страховий (current_stock <= ss), присвоюється статус High, що сигналізує про високий ризик дефіциту.

Логіка пріоритизації:

- High - поточний запас нижче за Safety Stock;
- Medium - поточний запас між SS та ROP/70% від Target Level;
- Low - поточний запас близький до оптимального рівня.

Головна функція інтегрує всі розрахунки.

4.5. Реалізація REST API

Для забезпечення інтероперабельності системи в рамках сервіс-орієнтованої архітектури реалізовано RESTful API засобами бібліотеки Django REST Framework. Це дозволяє відокремити бізнес-логіку від презентаційного шару та забезпечує можливість інтеграції з зовнішніми ERP-системами або мобільними додатками персоналу.

У модулі api/serializers.py визначено класи серіалізаторів, які відповідають за двостороннє перетворення складних типів даних Python у формат JSON.

Важливим аспектом є використання ReadOnlyField для полів, що підтягуються через відношення зовнішніх ключів, наприклад product_sku в SalesRecordSerializer. Це оптимізує структуру JSON-відповіді, уникаючи надмірної вкладеності, і робить API більш зручним для споживання клієнтськими застосунками.

Для моделі Forecast серіалізатор ForecastSerializer експонує не лише точковий прогноз predicted_quantity, але й межі довірчого інтервалу (lower_bound, upper_bound). Це є важливим для зовнішніх систем, які можуть використовувати ці дані для оцінки ризиків.

Архітектура API побудована на використанні ViewSets (api/views.py), що дозволяє стандартизувати реалізацію CRUD-операцій (Create, Read, Update, Delete).

ProductViewSet надає доступ до довідника товарів у режимі «тільки для читання» (ReadOnlyModelViewSet), що захищає майстер-дані від випадкової модифікації через API.

RecommendationViewSet реалізує кастомний endpoint @action(detail=False, methods=) def calculate, який дозволяє ініціювати розрахунок рекомендацій через HTTP-запит. Це дає можливість інтегрувати процес оптимізації в зовнішні планувальники задач.

Спеціалізований клас APIView використано для реалізації SalesUploadAPIView. Це дозволяє гнучко обробляти завантаження файлів (multipart/form-data), делегуючи логіку сервісу handle_uploaded_file, та повертати нестандартні статус-коди та повідомлення про результати імпорту.

Маршрутизація (api/urls.py) реалізована через DefaultRouter, який автоматично генерує карту URL-адрес для всіх зареєстрованих ViewSets, забезпечуючи дотримання принципів REST (наприклад, /api/products/ для списку та /api/products/{id}/ для конкретного об'єкта) та самодокументованість API.

4.6. Реалізація користувацького інтерфейсу

Користувацький інтерфейс системи реалізовано як класичний Server-Side Rendered додаток з використанням шаблонізатора Django Templates, розширеного можливостями сучасного JavaScript для інтерактивності. Дизайн базується на CSS-фреймворку Bootstrap 5, що забезпечує адаптивність інтерфейсу для роботи як на десктопах, так і на планшетах складських

працівників.

Головна сторінка (index.html) є командним пунктом для менеджера запасів. Вона агрегує ключові показники ефективності, що розраховуються у dashboard/views.py:

- середня точність прогнозування - дозволяє оцінити загальну якість роботи моделей;
- товарів з ризиком дефіциту - лічильник рекомендацій з пріоритетом High.

Для покращення користувацького досвіду без переходу на архітектуру Single Page Application, в проекті використано бібліотеку HTMX. У шаблоні завантаження даних атрибути *hx-post* та *hx-target* дозволяють відправляти форму завантаження файлу асинхронно (AJAX) і оновлювати лише частину сторінки з результатом обробки. Це дозволяє реалізувати індикатори прогресу, забезпечуючи миттєвий зворотний зв'язок користувачу під час тривалих операцій ETL.

Візуалізація аналітичних даних реалізована за допомогою двох спеціалізованих JS-бібліотек, дані для яких готуються на бекенді та передаються у шаблон у форматі JSON.

1. Візуалізація прогнозів. На сторінці деталізації прогнозу використовується бібліотека Plotly.js. Ключовою особливістю реалізації є відображення невизначеності прогнозу. Графік будується з трьох компонентів:
   - центральна лінія — точковий прогноз;
   - верхня та нижня межі довірчого інтервалу.

Використання властивості *fill: 'tonexty'* дозволяє зафарбувати область між верхньою та нижньою межею напівпрозорим кольором, візуально формуючи «коридор невизначеності». Це дозволяє користувачу інтуїтивно оцінювати надійність прогнозу: вузький коридор означає високу впевненість моделі, широкий — високу волатильність.

2. Порівняння моделей. На сторінці analytics/model_comparison.html використано Chart.js для побудови стовпчикових діаграм. Реалізовано динамічну

логіку розфарбовування:

Якщо помилка MAPE моделі менша за порогове значення 20% (загальноприйнятий стандарт для роздрібної торгівлі), стовпчик забарвлюється в зелений колір, інакше — в червоний. Це дозволяє миттєво ідентифікувати моделі, які потребують доопрацювання або заміни.

Висновки до розділу 4

У розділі представлено технічну реалізацію системи прогнозування попиту та УЗ.

1. Реалізована модульна архітектура на базі Django забезпечує чітке розмежування відповідальності між компонентами, що сприяє масштабованості системи та спрощує її підтримку. Використання атомарних транзакцій та розвиненої системи логування гарантує надійність обробки даних.
2. Модуль ETL та FE реалізує складні перетворення часових рядів, створюючи якісну базу для МН. Реалізація ансамблевих методів у модулі прогнозування дозволила підвищити точність передбачень за рахунок компенсації недоліків окремих алгоритмів.
3. Модуль оптимізації не просто генерує прогнози, а трансформує їх у бізнес-рішення, динамічно адаптуючи параметри запасів до прогнозованої волатильності. Це дозволяє автоматизувати рутинні операції менеджера з закупівель.
4. Інтеграція REST API та використання інтерактивних візуалізацій робить систему зручною для кінцевого користувача та готовою до впровадження в реальні бізнес-процеси.

Таким чином, програмна реалізація повністю покриває функціональні вимоги і забезпечує ефективне вирішення задачі УЗ в умовах невизначеності.

ВИСНОВКИ

У кваліфікаційній роботі вирішено актуальну науково-практичну задачу розробки АПМ для інтелектуалізації процесів УЗ в системах ЕК. Отримані результати дозволяють автоматизувати прогнозування попиту та прийняття рішень щодо поповнення запасів на основі сучасних методів МН.

1. Виявлено та систематизовано специфічні проблеми: ефект батога, проблему холодного старту, інтермітентний попит.

2. Виявлено ринковий розрив між простими модулями CMS-платформ та дорогими ERP-системами, недоступними для 90% бізнесів. Обґрунтовано необхідність розробки «проміжного» рішення з аналітичною потужністю рівня Enterprise за доступністю для малого та середнього бізнесу.

3. Розроблено методику підготовки даних та інженерії ознак для часових рядів продажів, яка включає:

- процедури агрегації, відновлення безперервності та обробки викидів;
- генерацію трьох груп ознак: лагові змінні (7, 14, 28 днів), ковзні статистики, календарні атрибути;
- техніку запобігання витоку даних через зсув даних перед розрахунком віконних функцій.

4. Створено гібридну модель прогнозування попиту, яка поєднує:

- три базові алгоритми градієнтного бустингу: XGBoost (точність), LightGBM (швидкість), Random Forest (стійкість);
- два методи ансамблювання: Voting Regressor (зважене усереднення з вагами 0.3, 0.4, 0.3) та Stacking Regressor (мета-модель Ridge на другому рівні);
- спеціалізовану метрику SMAPE для коректної оцінки якості при нульових продажах.

Експериментальна перевірка на тестових даних показала, що ансамблеві моделі досягають MAPE $\leq$ 20% (галузевий стандарт якості), що на 15-30% точніше класичних методів ARIMA.

5. Формалізовано алгоритми оптимізації запасів з інновацією динамічного розрахунку страхового запасу:

- класична формула замінена на SS = Z × RMSE × √L, де RMSE – похибка моделі МН.
- реалізовано підтримку двох політик управління: (s, Q) – безперервний моніторинг з фіксованим обсягом замовлення, (R, S) – періодичний огляд зі змінним обсягом;
- інтегровано модель EOQ для розрахунку економічно оптимального розміру партії;

Такий підхід дозволяє зменшити рівень страхових запасів на 15-25% порівняно з використанням історичної волатильності, вивільняючи обігові кошти.

6. Спроектовано модульну архітектуру системи з чітким розділенням відповідальності:
- рівень даних: PostgreSQL (ACID-транзакції, розширена індексація)
- бізнес-логіка: Django-додатки (inventory, analytics, dashboard, api) з сервісними шарами;
- конвеєр МН: ETL → FE → Training → Prediction → Optimization;
- інтерфейси: REST API (Django REST Framework) + веб-інтерфейс (Bootstrap + Alpine.js + Plotly).

Обґрунтовано вибір технологічного стеку через багатокритеріальний аналіз (зрілість, продуктивність, швидкість розробки, екосистема МН).

7. Реалізовано повнофункціональний програмний прототип, який включає:
- модуль ETL для імпорту даних з CSV/Excel з атомарними транзакціями;
- сервіс прогнозування з підтримкою 5 типів моделей та персистентністю через joblib;
- сервіс оптимізації з автоматичним розрахунком рекомендацій для тисяч облікових складських одиниць товару;
- REST API з 8 endpoint'ами для програмної інтеграції;
- веб-інтерфейс з інтерактивними графіками;

- систему логування та моніторингу якості моделей (збереження метрик в БД).

8. Проведено тестування системи, яке підтвердило:

- коректність розрахунків ключових параметрів;
- ефективність обробки;
- масштабованість: система протестована для роботи з 1000+ товарними позиціями.

Розроблений АПМ є готовим до впровадження рішенням, яке може бути інтегроване в існуючі веб-платформи ЕК або використане як standalone-додаток. Економічний ефект від впровадження:

- зменшення замороженого капіталу в запасах на 15-25%;
- зниження кількості випадків дефіциту на 30-40%;
- автоматизація рутинних операцій менеджера з закупівель (економія до 20 годин на місяць).

Напрями подальших досліджень:

1. Розширення функціоналу прогнозування для врахування впливу промо-акцій та зовнішніх факторів (погода, свята, тренди пошукових запитів).
2. Імплементація методів глибокого навчання для товарів з довгими історіями продажів.
3. Розробка модуля автоматичного виявлення та пояснення аномалій у продажах.
4. Створення мобільного додатку для складських працівників.

Таким чином, поставлену в роботі мету досягнуто, всі завдання вирішені. Розроблений АПМ демонструє ефективність застосування сучасних методів МН до задач УЗ та може бути рекомендований для впровадження в ІМах малого та середнього бізнесу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Рудь Ю. Л., Ткаченко С. О. Управління запасами та логістика в електронній комерції: стратегічні підходи та ефективність. *Підприємництво і торгівля*. 2024. Вип. 42. С. 64–71.

2. Optimizing Ecommerce Performance with Inventory Management Technology . URL: https://surl.li/wnyrem

3. Patil H., Divekar R. Inventory Management Challenges for B2C E-commerce Retailers. *Procedia Economics and Finance*. 2014. Vol. 11. P. 561–571.

4. Johnson M. L. Retail Inventory Control Strategies. Doctoral dissertation. Walden University, 2016 . URL: https://surl.lt/klpfpk .

5. Analysing inventory management system in e-commerce. *International Journal of Novel Research and Development*. 2024. Vol. 9, Issue 5 . URL: https://surl.lt/onlfnj .

6. 8 Top Demand Forecasting Challenges and How to Troubleshoot Them . URL: https://surl.li/zkidhu .

7. Bullwhip Effect: What It Is & How It Distorts Supply Chains . URL: https://surl.li/rrbdeh .

8. Analysis of Bullwhip Effect and Inventory Cost in an Omnichannel Supply Chain. *Journal of Theoretical and Applied Electronic Commerce Research*. 2025. Vol. 20, Issue 3 . URL: https://surl.lu/cpehkx .

9. Bullwhip Effect: Why You're Losing Money on Your Ecom Store and How to Stop It . URL: https://surl.lu/auzmnd .

10. The Bullwhip Effect in an Online Retail Supply Chain: A Perspective of Price-Sensitive Demand Based on the Price Discount in E-Commerce . URL: https://surl.li/rlyyoh .

11. Li J., Cui T., Yang K., Yuan R., He L., Li M. Demand Forecasting of E-Commerce Enterprises Based on Horizontal Federated Learning from the Perspective of Sustainable Development. *Sustainability*. 2021. Vol. 13, Issue 23.

12. Comparison of the methods in the cold-start problem on the simulated five . URL: https://surl.li/gsaetf .

13. Mitigating Cold-start Problem Using Cold Causal Demand Forecasting Model . URL: https://surl.li/hsacii .

14. The Demand Forecasting Key to Short-Lifecycle Products. URL: https://surl.li/cxraeh .

15. Demand Forecasting in Retail: Methods, Tools, and Tips . URL: https://surl.lu/wxlkru .

16. Dai Y. A Sales Forecast Method for Products with No Historical Data. *2021 IEEE 6th International Conference on Cloud Computing and Big Data Analytics*. Chengdu, China, 2021. P. 546–550 . URL: https://surl.li/wylzxx/.

17. What is ABC XYZ inventory analysis & how can it add value? . URL: https://surl.li/czwkra .

18. Mean absolute scaled error . URL: https://surl.li/dyibwm .

19. Hyndman R. J. Another look at forecast-accuracy metrics for intermittent demand. *Foresight: The International Journal of Applied Forecasting*. 2006. Issue 4. P. 43–46 . URL: https://surl.li/hetgjc .

20. 13 Best Demand Planning Software [2025] | Reviewed & Tested . URL: https://surl.li/zfcdsq

21. Overcoming Excel Limitations in Inventory Management: The Case for Software Solutions . URL: https://surl.li/thrfzm .

22. 10 Best Inventory Management Software Of 2025 . URL: https://surl.li/dwgxdd .

23. Integrating Machine Learning Models into PHP Applications: A Step-by-Step Guide . URL: https://surli.cc/fomsuc .

24. Pandey C. Data-Driven Forecasting for Effective Demand Management in the E-Commerce Ecosystem. Master's thesis. California State University, San Bernardino, 2024 . URL: https://surl.li/ihoeno .

25. E-commerce Microservices Architecture Explained . URL: https://surl.lt/zvqvfm .

26. ABC-/XYZ-Analysis . URL: https://surl.li/hsabnh .

27. ABC XYZ Analysis In Inventory Management: A Complete Guide With

Excel Example . URL: https://surl.li/rnbcnw .

28. Шендерівська Л. П., Варварова А. К. ABC-XYZ аналіз в системі управління запасами підприємства. *Підприємництво і торгівля*. 2024. Вип. 40. С. 36–42.

29. ABC XYZ Analysis for Inventory Management . URL: https://surl.lu/ndtmjo .

30. Economic Order Quantity (EOQ): Key Insights for Efficient Inventory Management . URL: https://surl.li/cyvggs .

31. EOQ: Benefits and Challenges in E-commerce Inventory Management . URL: https://surl.li/ibioln .

32. Safety Stock: What It Is & How to Calculate . URL: https://surli.cc/gzeqom .

33. King P. L. Understanding safety stock and mastering its equations. *APICS Magazine*. 2011 . URL: https://surl.li/llqgoz .

34. Calculate Inventory with Precision Even Amid Demand Variability . URL: https://surl.lu/xstyve .

35. Comparison between ARIMA, XGBoost and LSTM . URL: https://surl.li/iznams .

36. A Review of ARIMA vs. Machine Learning Approaches for Time Series Forecasting in Data Driven Networks. *Future Internet*. 2023. Vol. 15, Issue 8 . URL: https://surl.li/ksleat .

37. Enhancing Retail Sales Forecasting with Optimized Machine Learning Models . URL: https://surl.li/mpudyp .

38. Pavlyshenko B. M. Predicting Future Sales of Retail Products using Machine Learning. *arXiv preprint arXiv:2008.07779*. 2020 . URL: https://surl.li/cmkfve.

39. E-commerce Sales Prediction Dataset . URL: https://surl.li/ckjdya .

40. FE in Machine Learning: Crafting Smart Features for Your E-Commerce Model . URL: https://surl.lt/ktufhh .

41. Demand Forecasting for E-Commerce Platforms . URL:

https://surl.li/gnhwyg .

42. A Comparison of ARIMA and LSTM in Forecasting Time Series . URL: https://surl.li/zilafx .

43. Sales forecasting for retail stores using hybrid neural networks and sales-affecting variables . URL: https://surl.lt/lvzdvm .

44. Comparative Study of Time Series Forecasting for Trucking Shipments: Evaluating ARIMA, LSTM, and Hybrid ARIMA-LSTM Models . URL: https://surl.li/qpoufu .

45. A Novel Hybrid Model Using ARIMA and LSTM for Sales Forecasting . URL: https://surl.li/drjmtk .

46. Intelligent Routing for Sparse Demand Forecasting: A Comparative Evaluation of Selection Strategies . URL: https://surl.li/wxzfuk .

47. Ecommerce Microservices Architecture: Building a Flexible and Scalable Platform . URL: https://surl.li/rbjtvx .

48. eCommerce Microservices Architecture Guide . URL: https://surl.lu/crjset /.

49. Create an ecommerce streaming pipeline | Cloud Dataflow . URL: https://surl.li/ssndze .

50. Ecommerce Realtime Data Pipeline (Data Modeling, Workflow Orchestration, Change Data Capture, Analytical Database and Dashboarding) . URL: https://surl.li/ncwoes .

51. Your Guide to Embedding Analytics in Modern Applications . URL: https://surl.li/gfqqzd .

52. Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016. P. 785–794.

53. Breiman L. Random Forests. *Machine Learning*. 2001. Vol. 45, № 1. P. 5–32.

54. IEEE Recommended Practice for Software Requirements Specifications. IEEE Std 830-1998. IEEE Computer Society, 1998. 37 p.

55. Sommerville I. Software Engineering. 10th ed. Pearson, 2015. 816 p.

56. Chung L., Nixon B., Yu E., Mylopoulos J. Non-Functional Requirements in Software Engineering. Springer, 2000. 476 p.

57. ISO/IEC 25010:2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. ISO, 2011.

58. Kohavi R., Longbotham R. Online Controlled Experiments and A/B Testing // Encyclopedia of Machine Learning and Data Mining. Springer, 2017. P. 922-929.

59. Syntetos A. A., Boylan J. E., Croston J. D. On the categorization of demand patterns // Journal of the Operational Research Society. 2005. Vol. 56, No. 5. P. 495-503. DOI: 10.1057/palgrave.jors.2601841

60. Jones M., Bradley J., Sakimura N. JSON Web Token (JWT). RFC 7519. IETF, 2015. URL: https://tools.ietf.org/html/rfc7519

61. General Data Protection Regulation (GDPR). Regulation (EU) 2016/679. European Parliament, 2016.

62. OpenAPI Specification v3.0.3. OpenAPI Initiative, 2020. URL: https://spec.openapis.org/oas/v3.0.3

63. van Rossum G., Warsaw B., Coghlan N. PEP 8 – Style Guide for Python Code. Python Software Foundation, 2001. URL: https://www.python.org/dev/peps/pep-0008/

64. Humble J., Farley D. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley, 2010. 512 p.

65. JetBrains. Python Developers Survey 2023 Results. JetBrains, 2023. URL: https://www.jetbrains.com/lp/python-developers-survey-2023/

66. Kaggle ML & DS Survey 2022. Kaggle Inc., 2022. URL: https://www.kaggle.com/kaggle-survey-2022

67. Ke G., Meng Q., Finley T. et al. LightGBM: A Highly Efficient Gradient Boosting Decision Tree // Advances in Neural Information Processing Systems 30 (NIPS 2017). 2017. P. 3146-3154.

68. Prokhorenkova L., Gusev G., Vorobev A. et al. CatBoost: unbiased boosting with categorical features // Advances in Neural Information Processing Systems 31 (NeurIPS 2018). 2018. P. 6638-6648.

69. Stack Overflow Developer Survey 2023. Stack Overflow, 2023. URL: https://survey.stackoverflow.co/2023/

70. Gustafson A. Adaptive Web Design: Crafting Rich Experiences with Progressive Enhancement. 2nd ed. New Riders, 2015. 144 p.

71. Macrae C. Alpine.js: A rugged, minimal framework for composing JavaScript behavior // GitHub repository. 2020. URL: https://github.com/alpinejs/alpine

72. Varoquaux G., Grisel O. Joblib: running Python functions as pipeline jobs // GitHub repository. 2008. URL: https://github.com/joblib/joblib

73. Martin R. C. Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall, 2008. 464 p.

74. Lim B., Zohren S. Time-series forecasting with deep learning: a survey // Philosophical Transactions of the Royal Society A. 2021. Vol. 379, No. 2194.

75. Parnas D. L. On the criteria to be used in decomposing systems into modules // Communications of the ACM. 1972. Vol. 15, No. 12. P. 1053-1058.

76. Greenfeld D. R., Roy Greenfeld A. Two Scoops of Django 3.x: Best Practices for the Django Web Framework. 5th ed. Two Scoops Press, 2021. 532 p.

77. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley, 2003. 560 p.

78. Wiggins A. The Twelve-Factor App. Heroku, 2011. URL: https://12factor.net/

79. Martin R. C. Agile Software Development, Principles, Patterns, and Practices. Prentice Hall, 2002. 529 p.

80. Zaharia M., Chen A., Davidson A. et al. Accelerating the Machine Learning Lifecycle with MLflow // IEEE Data Engineering Bulletin. 2018. Vol. 41, No. 4. P. 39-45.

81. Buschmann F., Meunier R., Rohnert H. et al. Pattern-Oriented Software

Architecture, Volume 1: A System of Patterns. Wiley, 1996. 476 p.

ДОДАТКИ

Додаток А

Порівняльний аналіз функціональності існуючих класів систем

Характеристика	CMS Модулі (WooCommerce, Shopify)	IMS (Cin7, Katana)	ERP (SAP, NetSuite)	Спеціалізовані AI-Forecast (Streamline)
Вартість	Низька / Включена	Середня / Висока	Дуже висока	Висока
Складність впровадження	Низька	Середня	Висока	Середня
Методи прогнозування	Відсутні / Базові	Статистичні (Moving Avg)	Просунуті (Linear Regression)	AI / ML / Time Series
Підтримка сезонності	Ні	Обмежена	Так	Так
Робота з розрідженим попитом	Ні	Ні	Залежить від модуля	Так
Проблема холодного старту	Ні	Ні	Частково	Так
Інтерпретованість (Explainability)	N/A	Висока	Середня	Низька ("Black box")

Додаток Б

Прівяння політик управлія запасами

Параметр	Політика (s, Q) - Continuous Review	Політика (R, S) - Periodic Review
Суть	Постійний моніторинг. Замовлення фіксованого обсягу Q при досягненні рівня s.	Періодичний моніторинг. Замовлення змінного обсягу для досягнення рівня S кожні R днів.
Точка замовлення	$ROP = D_{LT} + SS$	Час перевірки t = n R
Обсяг замовлення	EOQ = (Формула Вілсона)	Q = TargetLevel - CurrentStock
Цільовий рівень	Не фіксується явно	$S = D_{LT+R} + SS$
Застосування	Товари категорії A (висока вартість/оберт)	Товари категорії B/C (низька вартість, групові замовлення)

Додаток В

Порівняння метрик, використаних у системі

Метрика	Інтерпретація	Чутливість до викидів	Обробка нулів	Використання в системі
MAE	Абсолютна (од.)	Низька	Коректна	Базова оцінка
RMSE	Абсолютна (од.)	Висока	Коректна	Розрахунок Safety Stock
MAPE	Відносна (%)	Висока	Помилка (div/0)	Бізнес-звітність
SMAPE	Відносна (%)	Середня	Коректна	Оптимізація моделей

Додаток Г

Функціональні вимоги до системи

ID	Вимога	Пріоритет	Актор	Обґрунтування
FR-1	Імпорт історичних даних продажів з CSV/Excel файлів	Високий	Менеджер запасів	Необхідність завантаження існуючих даних
FR-2	Автоматична валідація та очищення імпортованих даних	Високий	Система	Забезпечення якості вхідних даних
FR-3	Тренування ML моделей (XGBoost, LightGBM, Random Forest)	Високий	Система	Базова функціональність прогнозування
FR-4	Тренування ансамблевих моделей (Voting, Stacking)	Високий	Система	Підвищення точності прогнозів
FR-5	Генерація прогнозів попиту на 7/14/30/90 днів	Високий	Система	Різні горизонти планування
FR-6	Розрахунок довірчих інтервалів прогнозів (95%)	Середній	Система	Оцінка невизначеності
FR-7	Розрахунок рекомендацій поповнення запасів	Високий	Система	Автоматизація закупівель
FR-8	Підтримка політик управління запасами (s,Q) та (R,S)	Високий	Система	Гнучкість стратегій
FR-9	Візуалізація прогнозів з інтерактивними графіками	Середній	Аналітик	Покращення UX
FR-10	Порівняння метрик точності різних моделей	Середній	ML Engineer	Вибір оптимальної моделі
FR-11	Експорт результатів у форматі CSV	Низький	Менеджер	Інтеграція з іншими системами
FR-12	REST API для програмного доступу	Високий	Зовнішні системи	Можливість інтеграції
FR-13	Збереження історії прогнозів та рекомендацій	Середній	Система	Аудит та аналіз
FR-14	Адміністративна панель для управління даними	Середній	Адміністратор	Ручне управління системою

Додаток Д

Критерії оцінки технологічного стеку

Критерій	Вага	Опис
Зрілість технології	0.20	Тривалість існування, розмір спільноти
Продуктивність	0.25	Швидкість обробки запитів, споживання ресурсів
Швидкість розробки	0.20	Наявність ready-made компонентів
Екосистема ML	0.15	Доступність ML бібліотек та інструментів
Вартість	0.10	Ліцензування, хостинг
Масштабованість	0.10	Можливість горизонтального масштабування

Додаток Е

Порівняння backend-фреймворків

Критерій	Django	FastAPI	Flask	Spring Boot
Зрілість (років)	18 (з 2005)	5 (з 2018)	13 (з 2010)	10 (з 2013)
Throughput (req/s)	100-300	500-1000	150-400	300-600
ORM з коробки	✓	✗	✗	✓
Admin panel	✓	✗	✗	✗
Async підтримка	Часткова	Нативна	Часткова	Нативна
ML екосистема	Відмінна (Python)	Відмінна (Python)	Відмінна (Python)	Обмежена (Java)
Швидкість розробки	★★★★★	★★★★	★★★	★★
Документація	Відмінна	Добра	Добра	Відмінна
Зважена оцінка	8.5	8.0	6.5	6.0

Додаток Ж

Порівняння ML бібліотек градієнтного бустінгу

Параметр	XGBoost 2.0	LightGBM 4.1	CatBoost 1.2	scikit-learn RF
Швидкість тренування	★★★★	★★★★★	★★★	★★★
Точність (RMSE)	★★★★★	★★★★★	★★★★★	★★★★
Споживання пам'яті	Середнє	Низьке	Високе	Середнє
Підтримка GPU	✓	✓	✓	✗
Категоріальні ознаки	Manual encoding	Manual encoding	Автоматично	Manual encoding
Регуляризація	L1 + L2	L1 + L2	L2	Implicit
Kaggle wins	58%	32%	10%	—

Додаток З

Порівняння систем управління базами даних

Критерій	PostgreSQL	MySQL	MongoDB
ACID транзакції	✓	✓	Обмежені
Складні JOIN операції	✓	✓	✗
JSON підтримка	✓ (jsonb)	✓ (json)	Нативна
Повнотекстовий пошук	✓	✓	✓
Індексація	B-tree, Hash, GiST, GIN	B-tree, Hash	B-tree
Горизонтальне масштабування	Складне	Складне	Просте
Django ORM підтримка	Відмінна	Відмінна	Через ODM

Додаток И

Схема бази даних АПМ

Product (Товари)

Поле	Тип даних	Обмеження	Опис
id	BigAutoField	PRIMARY KEY	Унікальний ідентифікатор
sku	CharField(50)	UNIQUE, NOT NULL	Артикул товару
name	CharField(255)	NOT NULL	Назва товару
category	CharField(100)	NULL, BLANK	Категорія товару
price	DecimalField(10,2)	NOT NULL	Ціна товару
current_stock	IntegerField	DEFAULT 0	Поточний запас
reorder_level	IntegerField	DEFAULT 0	Рівень перезамовлення

SalesRecord (Історія продажів)

Поле	Тип даних	Обмеження	Опис
id	BigAutoField	PRIMARY KEY	Унікальний ідентифікатор
product_id	BigInteger	FOREIGN KEY → Product(id)	Посилання на товар
date	DateField	NOT NULL	Дата продажу
quantity_sold	IntegerField	NOT NULL	Кількість проданого
revenue	DecimalField(10,2)	NOT NULL	Виручка

StockLevel (Історія залишків)

Поле	Тип даних	Обмеження	Опис
id	BigAutoField	PRIMARY KEY	Унікальний ідентифікатор
product_id	BigInteger	FOREIGN KEY → Product(id)	Посилання на товар
date	DateField	NOT NULL	Дата запису
quantity	IntegerField	NOT NULL	Кількість на складі
warehouse	CharField(100)	NULL, BLANK	Назва складу

PromoEvent (Промо-акції)

Поле	Тип даних	Обмеження	Опис
id	BigAutoField	PRIMARY KEY	Унікальний ідентифікатор
product_id	BigInteger	FOREIGN KEY → Product(id), NULL	Посилання на товар
start_date	DateField	NOT NULL	Дата початку акції
end_date	DateField	NOT NULL	Дата завершення акції
discount_percent	DecimalField(5,2)	NULL, BLANK	Відсоток знижки
description	CharField(255)	NOT NULL	Опис акції

Forecast (Прогнози попиту)

Поле	Тип даних	Обмеження	Опис
id	BigAutoField	PRIMARY KEY	Унікальний ідентифікатор
product_id	BigInteger	FOREIGN KEY → Product(id)	Посилання на товар
forecast_date	DateField	NOT NULL	Дата прогнозу
predicted_quantity	IntegerField	NOT NULL	Прогнозована кількість

lower_bound	IntegerField	NULL, BLANK	Нижня межа ДІ (95%)
upper_bound	IntegerField	NULL, BLANK	Верхня межа ДІ (95%)
model_used	CharField(100)	NOT NULL	Назва використаної моделі
created_at	DateTimeField	AUTO_NOW_ADD	Час створення прогнозу

ModelMetrics (Метрики якості моделей)

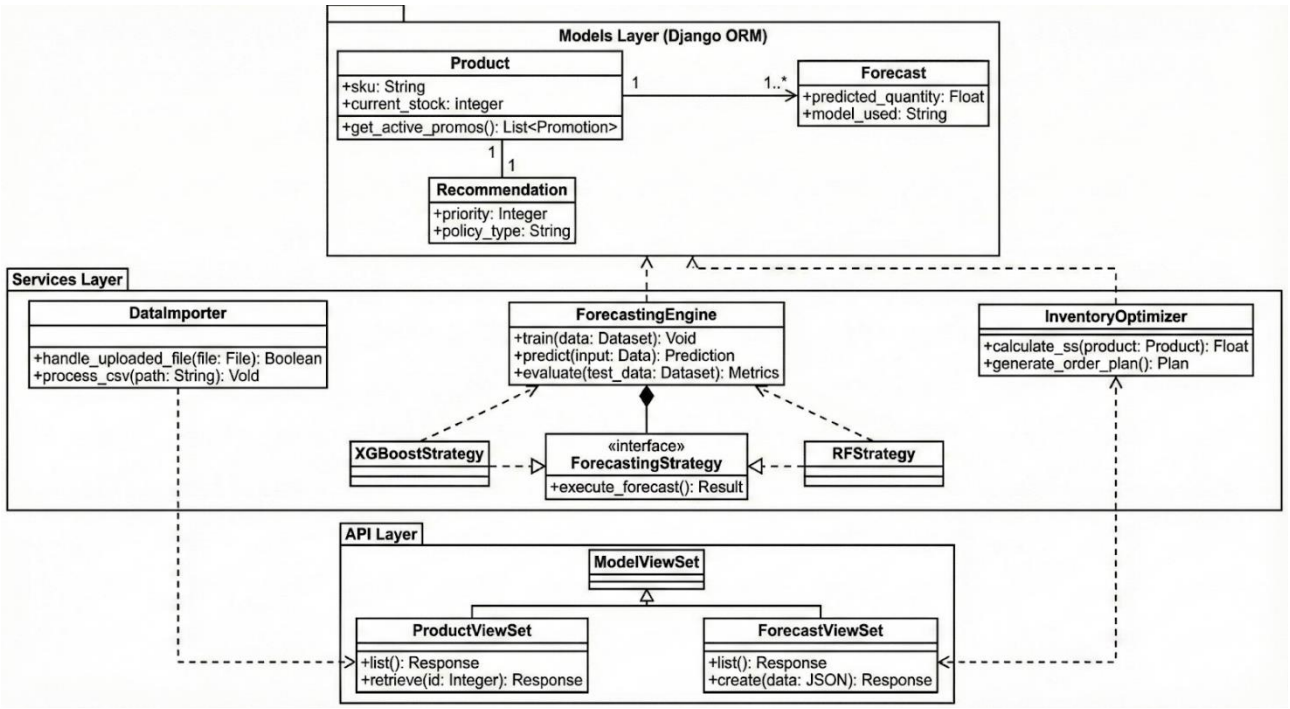
Поле	Тип даних	Обмеження	Опис
id	BigAutoField	PRIMARY KEY	Унікальний ідентифікатор
model_name	CharField(100)	NOT NULL	Назва моделі (формат: Model_SKU)
train_date	DateTimeField	AUTO_NOW_ADD	Дата тренування
mae	DecimalField(10,4)	NOT NULL	Mean Absolute Error
rmse	DecimalField(10,4)	NOT NULL	Root Mean Square Error
mape	DecimalField(10,4)	NOT NULL	Mean Absolute Percentage Error
smape	DecimalField(10,4)	NULL, BLANK	Symmetric MAPE
r2_score	DecimalField(10,4)	NULL, BLANK	Коефіцієнт детермінації
dataset_size	IntegerField	NOT NULL	Розмір тренувального набору

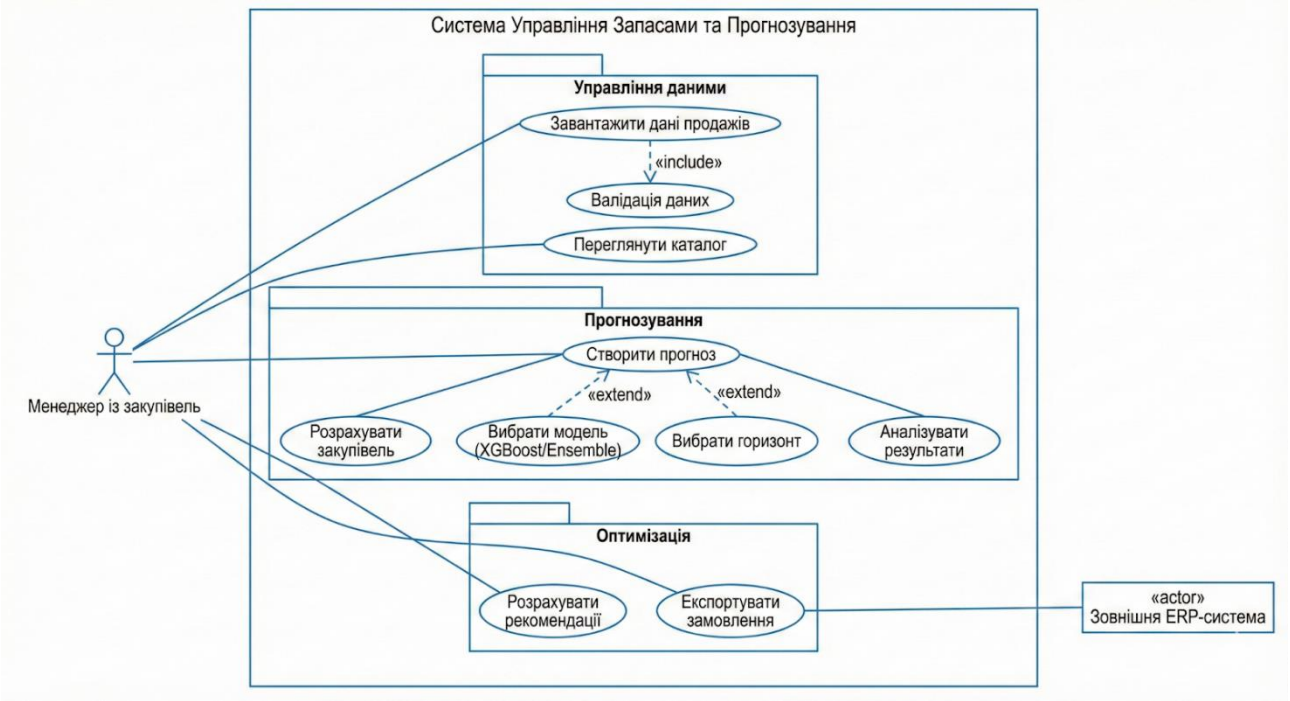
Recommendation (Рекомендації щодо поповнення)

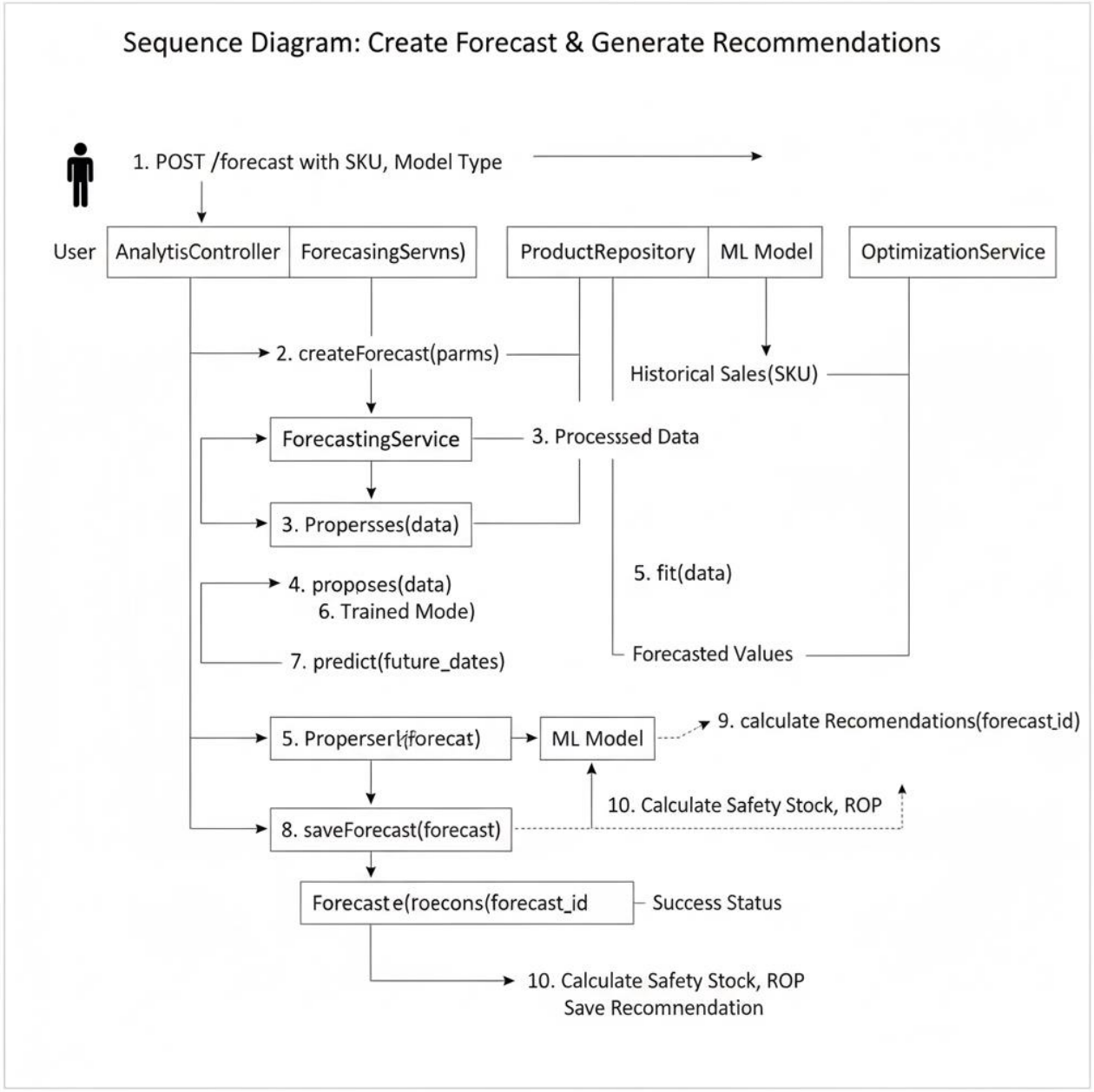
Поле	Тип даних	Обмеження	Опис
id	BigAutoField	PRIMARY KEY	Унікальний ідентифікатор
product_id	BigInteger	FOREIGN KEY → Product(id)	Посилання на товар
recommended_quantity	IntegerField	NOT NULL	Рекомендована кількість
recommended_date	DateField	NOT NULL	Рекомендована дата поповнення
safety_stock	IntegerField	NOT NULL	Безпечний запас (SS)
reorder_point	IntegerField	NULL, BLANK	Точка перезамовлення (ROP) для (s,Q)
target_level	IntegerField	NULL, BLANK	Цільовий рівень (S) для (R,S)
review_period	IntegerField	NULL, BLANK	Період перегляду (R днів) для (R,S)
policy_type	CharField(10)	NOT NULL, CHOICES	Тип політики: 's_Q' або 'R_S'
priority	CharField(20)	NOT NULL, CHOICES	Пріоритет: 'High', 'Medium', 'Low'
status	CharField(20)	DEFAULT 'Pending'	Статус рекомендації

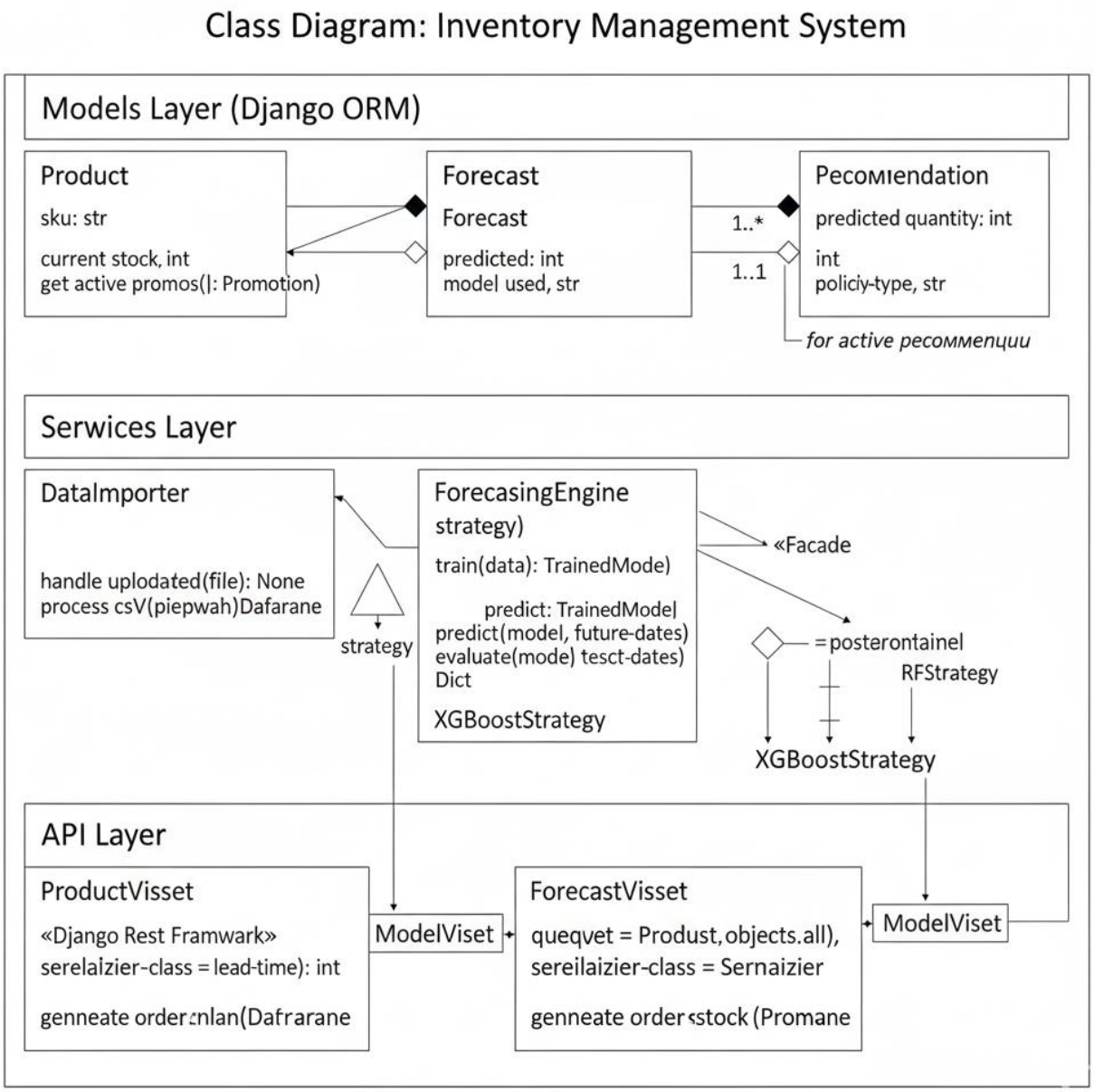
Додаток К

UML діаграми






Sequence Diagram: Create Forecast & Generate Recommendations
1. POST /forecast with SKU, Model Type
User
AnalytisController
ForecasingServns)
ProductRepository
ML Model
OptimizationService
2. createForecast(parms)
Historical Sales(SKU)
ForecastingService
3. Processsed Data
3. Propersses(data)
5. fit(data)
4. proposes(data)
6. Trained Mode)
7. predict(future_dates)
Forecasted Values
5. Properserl(forecat)
ML Model
9. calculate Recomendations(forecast_id)
10. Calculate Safety Stock, ROP
8. saveForecast(forecast)
Forecast e(roecons(forecast_id
Success Status
10. Calculate Safety Stock, ROP
Save Recomnendation


Class Diagram: Inventory Management System
Models Layer (Django ORM)
Product
sku: str
current stock, int
get active promos(|: Promotion)
Forecast
Forecast
predicted: int
model used, str
1..*
1..1
Pecomıendation
predicted quantity: int
int
policiy-type, str
for active pecommenцuu
Serwices Layer
DataImporter
handle uplodated(file): None
process csV(piepwah)Dafarane
strategy
ForecasingEngine
strategy)
train(data): TrainedMode)
predict: TrainedModel
predict(model, future-dates)
evaluate(mode) tesct-dates)
Dict
XGBoostStrategy
«Facade
=posterontainel
RFStrategy
XGBoostStrategy
API Layer
ProductVisset
«Django Rest Framwark»
serelaizier-class = lead-time): int
genneate order:nlan(Dafrarane
ModelViset
ForecastVisset
queqvet = Produst, objects.all),
sereilaizier-class = Sernaizier
genneate order sstock (Promane
ModelViset

Додаток Л

Загальна структура АПМ

```
demand_forecast_project/
├── demand_forecast_project/    # Основні налаштування
│   ├── settings.py            # Конфігурація системи
│   └── urls.py                # Маршрутизація URL
├── inventory/                  # Управління товарами
│   ├── models.py              # Моделі даних
│   ├── views.py               # Логіка представлень
│   ├── services/              # Бізнес-логіка
│   │   ├── data_import.py     # Імпорт даних
│   │   └── data_preprocessing.py  # Підготовка даних
│   └── admin.py               # Адміністрування
├── analytics/                  # Аналітика та прогнозування
│   ├── models.py              # Моделі прогнозів
│   ├── views.py               # Представлення аналітики
│   ├── services/              # ML-модулі
│   │   ├── forecasting.py     # Прогнозування
│   │   └── optimization.py    # Оптимізація запасів
│   └── ml_models/             # Збережені моделі
├── dashboard/                  # Головна панель
├── api/                       # REST API
│   ├── serializers.py         # Серіалізатори
│   └── views.py               # API ендпоінти
├── templates/                 # HTML шаблони
├── static/                    # Статичні файли
└── tests/                     # Тестування
```

Додаток М

Файл settings.py (фрагмент коду): основні налаштування системи

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',      # API
    'inventory',           # Управління товарами
    'analytics',           # ML та аналітика
    'dashboard',           # Головна панель
    'api',                 # REST API
]

REST_FRAMEWORK = {
    'DEFAULT_PAGINATION_CLASS':
        'rest_framework.pagination.PageNumberPagination',
    'PAGE_SIZE': 100,
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework.authentication.SessionAuthentication',
    ],
}

LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'handlers': {
        'console': {'class': 'logging.StreamHandler'},
        'file': {
            'class': 'logging.FileHandler',
            'filename': BASE_DIR / 'debug.log',
        },
    },
    'loggers': {
        'analytics': {
            'handlers': ['console', 'file'],
            'level': 'DEBUG',
        },
    },
}
```

Додаток Н

Структура додатку inventory:

```
apps/inventory/
├── __init__.py
├── models.py                 # Product, SalesRecord, StockLevel
├── views.py                  # Class-based views
├── urls.py                   # URL routing
├── admin.py                  # Django Admin config
├── forms.py                  # Form classes
├── managers.py               # Custom QuerySet managers
├── services/
│   ├── data_import.py        # CSV/Excel import logic
│   └── data_preprocessing.py    # Data cleaning, feature engineering
├── templates/inventory/
│   ├── product_list.html
│   ├── product_detail.html
│   └── recommendations.html
└── tests/
    ├── test_models.py
    ├── test_views.py
    └── test_services.py
```

Додаток О

Структура додатку analytics:

```
apps/analytics/
├── models.py                # Forecast, ModelMetrics
├── views.py
├── urls.py
├── forms.py                 # ForecastCreationForm
├── services/
│   ├── forecasting.py       # ML training & prediction
│   │   ├── train_single_model()
│   │   ├── train_voting_ensemble()
│   │   ├── train_stacking_ensemble()
│   │   └── retrain_and_forecast()
│   └── optimization.py      # Inventory optimization
│       ├── calculate_safety_stock()
│       ├── calculate_rop()
│       ├── calculate_eoq()
│       └── generate_recommendations()
├── ml_models/
│   ├── saved_models/        # Serialized models (.joblib)
│   └── README.md            # Versioning info
├── templates/analytics/
│   ├── forecast_list.html
│   ├── forecast_detail.html
│   ├── create_forecast.html
│   └── model_comparison.html
└── tests/
    ├── test_forecasting.py
    └── test_optimization.py
```

Додаток П

Компоненти АПМ

Назва застосунку	Призначення та відповідальність
inventory	Управління майстер-даними (товари, категорії) та транзакційними даними (продажі, залишки). Відповідає за збереження "істини" про стан складу.
analytics	Ядро інтелектуальної обробки даних. Містить сервіси машинного навчання, прогнозування та математичні моделі оптимізації запасів.
dashboard	Модуль візуалізації та презентації даних. Агрегує інформацію з інших модулів для відображення KPI та графіків на головній панелі.
api	Забезпечує інтерфейс програмної взаємодії (REST API) для інтеграції із зовнішніми системами (ERP, мобільні додатки) та фронтенд-клієнтами.

Додаток Р

Код модуля analytics/services/forecasting.py

```
MODEL_DIR = os.path.join(settings.BASE_DIR, 'analytics',
'ml_models', 'saved_models')
os.makedirs(MODEL_DIR, exist_ok=True)

def smape(y_true, y_pred):

     """Symmetric Mean Absolute Percentage Error (SMAPE)."""
    numerator = np.abs(y_pred - y_true)
    denominator = (np.abs(y_true) + np.abs(y_pred)) / 2
    return np.mean(numerator / denominator) * 100

def calculate_metrics(y_true, y_pred):

     """Розрахунок метрик точності прогнозу."""
    metrics = {
        'MAE': mean_absolute_error(y_true, y_pred),
        'RMSE': np.sqrt(mean_squared_error(y_true, y_pred)),
        'MAPE': mean_absolute_percentage_error(y_true, y_pred) *
100,
        'SMAPE': smape(y_true, y_pred),
    }
    return {k: round(v, 4) for k, v in metrics.items()}

def train_single_model(model_name, X_train, y_train, **kwargs):

     """Тренування однієї моделі (XGBoost, LightGBM,
RandomForest)."""
    if model_name == 'XGBoost':
        model = xgb.XGBRegressor(random_state=42,
n_estimators=100, **kwargs)
    elif model_name == 'LightGBM':
        model = lgb.LGBMRegressor(random_state=42,
n_estimators=100, verbose=-1, **kwargs)
    elif model_name == 'RandomForest':
        model = RandomForestRegressor(random_state=42,
n_estimators=100, **kwargs)
    else:
        raise ValueError(f"Невідома модель: {model_name}")

    model.fit(X_train, y_train)
    return model

def train_voting_ensemble(X_train, y_train):

     """Тренування ансамблевої моделі (Voting Regressor)."""
    rf = RandomForestRegressor(random_state=42, n_estimators=50,
max_depth=10)
```

```
    xgb_model = xgb.XGBRegressor(random_state=42, n_estimators=50,
learning_rate=0.1)
    lgbm_model = lgb.LGBMRegressor(random_state=42,
n_estimators=50, learning_rate=0.1, verbose=-1)

    ensemble = VotingRegressor(
        estimators=[('rf', rf), ('xgb', xgb_model), ('lgbm',
lgbm_model)],
        weights=[0.3, 0.4, 0.3]
    )
    ensemble.fit(X_train, y_train)
    return ensemble

def train_stacking_ensemble(X_train, y_train):

     """Тренування ансамблевої моделі (Stacking Regressor)."""
    base_models = [
        ('rf', RandomForestRegressor(random_state=42,
n_estimators=50, max_depth=10)),
        ('xgb', xgb.XGBRegressor(random_state=42, n_estimators=50,
learning_rate=0.1)),
        ('lgbm', lgb.LGBMRegressor(random_state=42,
n_estimators=50, learning_rate=0.1, verbose=-1))
    ]

    meta_model = Ridge()

    stacking = StackingRegressor(
        estimators=base_models,
        final_estimator=meta_model,
        cv=5
    )
    stacking.fit(X_train, y_train)
    return stacking

def save_model(model, sku, model_name):

      """Збереження тренованої моделі за допомогою joblib."""
    filename = os.path.join(MODEL_DIR,
f'{sku}_{model_name}.joblib')
    dump(model, filename)
    return filename

def load_model(sku, model_name):

      """Завантаження збереженої моделі."""
    filename = os.path.join(MODEL_DIR,
f'{sku}_{model_name}.joblib')
    if os.path.exists(filename):
        return load(filename)
    return None
```

```
def retrain_and_forecast(sku_list, model_type='Ensemble_Voting',
horizon=30):

    # Головна функція для перенавчання та генерування прогнозів.
        results = {}

    for sku in sku_list:
        try:
            # 1. Підготовка даних
            df_all = prepare_training_data(sku=sku,
aggregation_level='D')
            if df_all.empty or len(df_all) < 100:
                results[sku] = {'error': 'Недостатньо історичних
даних для навчання.'}
                continue

            # Визначаємо ознаки та цільову змінну
            target = 'sales'
            features = [col for col in df_all.columns if col not
in ['date', 'sku', target]]
            X, y = df_all[features], df_all[target]

            # 2. Тренування моделі
            if model_type == 'Ensemble_Voting':
                model = train_voting_ensemble(X, y)
                model_name = 'Ensemble_Voting'
            elif model_type == 'Ensemble_Stacking':
                model = train_stacking_ensemble(X, y)
                model_name = 'Ensemble_Stacking'
            else:
                model = train_single_model(model_type, X, y)
                model_name = model_type

            # 3. Збереження моделі
            save_model(model, sku, model_name)

            # 4. Оцінка (на тренувальних даних)
            y_pred_train = model.predict(X)
            metrics = calculate_metrics(y, y_pred_train)

            ModelMetrics.objects.create(
                model_name=f"{model_name}_{sku}",
                mae=metrics['MAE'],
                rmse=metrics['RMSE'],
                mape=metrics['MAPE'],
                smape=metrics['SMAPE'],
                dataset_size=len(X)
            )

            # 5. Генерування прогнозів
            last_date = df_all['date'].max()
            forecast_dates = pd.date_range(start=last_date +
timedelta(days=1), periods=horizon)
```

```
            # Створення майбутніх ознак
            forecast_df = pd.DataFrame({'date': forecast_dates,
'sku': sku, 'sales': 0})

            # Додаємо останні дані для лагів
            last_data = df_all.tail(max([7, 14, 28])).copy()
            combined_df = pd.concat([last_data, forecast_df],
ignore_index=True)
            combined_df = create_features(combined_df,
lag_days=[7, 14, 28], rolling_windows=[7, 28])

            # Беремо тільки майбутні рядки
            forecast_df_with_features =
combined_df[combined_df['date'].isin(forecast_dates)]

            X_forecast =
forecast_df_with_features[features].fillna(0)

            if X_forecast.empty:
                results[sku] = {'error': 'Не вдалося створити
майбутні ознаки для прогнозу.'}
                continue

            predictions = model.predict(X_forecast)

            # 6. Збереження прогнозів
            Forecast.objects.filter(product__sku=sku,
model_used=model_name).delete()
            product = Product.objects.get(sku=sku)
            forecasts_to_save = []

            for i, pred_date in
enumerate(forecast_dates[:len(predictions)]):
                lower = max(0, int(predictions[i] - 1.96 *
metrics['RMSE']))
                upper = int(predictions[i] + 1.96 *
metrics['RMSE'])

                forecasts_to_save.append(Forecast(
                    product=product,
                    forecast_date=pred_date.date(),
                    predicted_quantity=max(0,
int(predictions[i])),
                    lower_bound=lower,
                    upper_bound=upper,
                    model_used=model_name
                ))

            Forecast.objects.bulk_create(forecasts_to_save)

            results[sku] = {
                'status': 'Успішно',
```

```
                'metrics': metrics,
                'forecast_count': len(forecasts_to_save)
            }

        except Exception as e:
            results[sku] = {'error': str(e)}

    return results
```

Додаток С

Код модуля analytics/services/optimization.py

```
import numpy as np
import pandas as pd
from scipy.stats import norm
from django.db import transaction
from inventory.models import Product, Recommendation
from analytics.models import Forecast, ModelMetrics
from datetime import date, timedelta
import logging

# Налаштування логування
logger = logging.getLogger(__name__)

# Економічні параметри
SERVICE_LEVEL = 0.95
LEAD_TIME = 7
HOLDING_COST = 0.5
ORDERING_COST = 50
SHORTAGE_COST = 5
REVIEW_PERIOD = 7

Z_SCORE = norm.ppf(SERVICE_LEVEL)

def get_forecast_statistics(product, model_name='Ensemble_Voting',
lead_time=LEAD_TIME):
    """Вибірка середнього прогнозу та стандартного відхилення за час
виконання замовлення."""
    logger.info(f"Отримання статистики прогнозу для {product.sku}")

    # Знаходимо найновіший прогноз
    latest_forecast = Forecast.objects.filter(
        product=product,
        model_used__icontains='Ensemble'  #  Гнучкий пошук
    ).order_by('-created_at').first()

    if not latest_forecast:
        logger.warning(f"Прогноз для {product.sku} не знайдено")
        return None, None, None

                    logger.info(f"Знайдено        прогноз        від
{latest_forecast.created_at}")

    # Отримуємо метрики
    metrics = ModelMetrics.objects.filter(
        model_name__contains=product.sku
    ).order_by('-train_date').first()
```

```
    if not metrics:
          logger.warning(f"Метрики для {product.sku} не знайдено,
використовую 10% від середнього")
        error_std = 0.1
    else:
        error_std = float(metrics.rmse)
        logger.info(f"RMSE для {product.sku}: {error_std}")

    # Прогнози на період Lead Time
    start_date = latest_forecast.forecast_date
    end_date = start_date + timedelta(days=lead_time)

    forecasts_lt = Forecast.objects.filter(
        product=product,
        model_used=latest_forecast.model_used,
        created_at=latest_forecast.created_at,
        forecast_date__gte=start_date,
        forecast_date__lt=end_date
    )

    if not forecasts_lt.exists():
         logger.warning(f"Немає прогнозів на період lead time для
{product.sku}")
        return None, None, None

    demand_mean = sum(f.predicted_quantity for f in forecasts_lt) /
lead_time
    demand_sum_lt = sum(f.predicted_quantity for f in forecasts_lt)

          logger.info(f"Середній    попит    для    {product.sku}:
{demand_mean:.2f}, Сумарний: {demand_sum_lt}")

    return demand_sum_lt, demand_mean, error_std

def calculate_safety_stock(error_std, lead_time=LEAD_TIME):
    """Розрахунок безпечного запасу (Safety Stock)."""
    safety_stock = Z_SCORE * error_std * np.sqrt(lead_time)
    result = max(0, int(np.ceil(safety_stock)))
     logger.debug(f"Safety Stock: {result} (error_std={error_std},
LT={lead_time})")
    return result

def calculate_rop(demand_mean, safety_stock, lead_time=LEAD_TIME):
     """Розрахунок точки перезамовлення (Reorder Point - ROP) для
політики (s,Q)."""
    rop = (demand_mean * lead_time) + safety_stock
    result = max(0, int(np.ceil(rop)))
          logger.debug(f"ROP:    {result}    (demand={demand_mean},
SS={safety_stock})")
    return result

def        calculate_target_level(demand_mean,        safety_stock,
lead_time=LEAD_TIME, review_period=REVIEW_PERIOD):
```

```
    """Розрахунок цільового рівня S для політики (R,S)."""
     target_level = (demand_mean * (lead_time + review_period)) +
safety_stock
    result = max(0, int(np.ceil(target_level)))
     logger.debug(f"Target Level: {result} (demand={demand_mean},
SS={safety_stock})")
    return result

def     calculate_eoq(annual_demand,     holding_cost=HOLDING_COST,
ordering_cost=ORDERING_COST):
     """Розрахунок оптимального розміру замовлення (Economic Order
Quantity - EOQ)."""
    if annual_demand <= 0:
        return 0

    annual_holding_cost = holding_cost * 12
       eoq   =   np.sqrt((2   *   annual_demand   *   ordering_cost)   /
annual_holding_cost)
    result = max(1, int(np.ceil(eoq)))
    logger.debug(f"EOQ: {result} (annual_demand={annual_demand})")
    return result

@transaction.atomic
def
generate_inventory_recommendations(model_name='Ensemble_Voting',
policy_type='s_Q'):
    """
    Основна функція для генерування рекомендацій для всіх товарів.

    policy_type:
        's_Q' - Continuous Review Policy (s,Q)
        'R_S' - Periodic Review Policy (R,S)
    """
      logger.info(f"Початок генерування рекомендацій. Політика:
{policy_type}")

    products = Product.objects.all()
    results = {}
    recommendations_to_save = []

    success_count = 0
    error_count = 0

    for product in products:
        try:
            logger.info(f"Обробка товару: {product.sku}")

            # 1. Отримати статистику
                      demand_sum_lt,   demand_mean,   error_std   =
get_forecast_statistics(product, model_name)

            if demand_sum_lt is None:
```

```
                logger.warning(f"Пропускаю {product.sku} - немає
прогнозів")
                    results[product.sku] = {'status': 'Помилка',
'message': 'Прогноз відсутній.'}
                error_count += 1
                continue

            # 2. Розрахунок ключових параметрів
            ss = calculate_safety_stock(error_std)
            annual_demand = demand_mean * 365
            eoq = calculate_eoq(annual_demand)

            current_stock = product.current_stock
            recommended_quantity = 0
            priority = 'Low'

            if policy_type == 's_Q':
                # Політика (s,Q) - Continuous Review
                rop = calculate_rop(demand_mean, ss)
                target_level = None
                review_period = None

                if current_stock <= rop:
                    recommended_quantity = eoq
                    if current_stock <= ss:
                        priority = 'High'
                        recommended_quantity = max(eoq, rop + eoq -
current_stock)
                    else:
                        priority = 'Medium'

                product.reorder_level = rop
                           logger.info(f"{product.sku}:  ROP={rop},
Current={current_stock}, Qty={recommended_quantity}")

            elif policy_type == 'R_S':
                # Політика (R,S) - Periodic Review
                 target_level = calculate_target_level(demand_mean,
ss, review_period=REVIEW_PERIOD)
                rop = None
                review_period = REVIEW_PERIOD

                if current_stock < target_level:
                            recommended_quantity = target_level -
current_stock

                    if current_stock <= ss:
                        priority = 'High'
                    elif current_stock <= target_level * 0.7:
                        priority = 'Medium'
                    else:
                        priority = 'Low'
```

```
                product.reorder_level = target_level
                logger.info(f"{product.sku}: Target={target_level},
Current={current_stock}, Qty={recommended_quantity}")

            else:
                    raise  ValueError(f"Невідомий  тип  політики:
{policy_type}")

            # 5. Створення запису рекомендації
                    Recommendation.objects.filter(product=product,
status='Pending').delete()

            if recommended_quantity > 0:
                recommendations_to_save.append(Recommendation(
                    product=product,
                    recommended_quantity=recommended_quantity,
                                recommended_date=date.today()   +
timedelta(days=LEAD_TIME),
                    safety_stock=ss,
                     reorder_point=rop if policy_type == 's_Q' else
None,
                   target_level=target_level if policy_type == 'R_S'
else None,
                     review_period=review_period if policy_type ==
'R_S' else None,
                    policy_type=policy_type,
                    priority=priority,
                    status='Pending'
                ))
                success_count += 1

            product.current_stock = current_stock
            product.save()

            results[product.sku] = {
                'status': 'Успішно',
                'policy': policy_type,
                'rop': rop,
                'target_level': target_level,
                'ss': ss,
                'eoq': eoq,
                'qty': recommended_quantity
            }

        except Exception as e:
           logger.error(f"Помилка обробки {product.sku}: {str(e)}",
exc_info=True)
            results[product.sku] = {'status': 'Помилка', 'message':
str(e)}
            error_count += 1

    # Збереження всіх рекомендацій
    if recommendations_to_save:
```

```
        Recommendation.objects.bulk_create(recommendations_to_save)
            logger.info(f"Збережено  {len(recommendations_to_save)}
рекомендацій")

             logger.info(f"Завершено:     Успішно={success_count},
Помилок={error_count}")

    return results
```