

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)
на тему:

**«Розробка інформаційної системи аналізу і
прогнозування продажів в електронній комерції»**

Виконав студент групи ІПЗм-24-2
ТУРЧИН Микола Борисович

Керівник роботи:
ЧИЖМОТРА Олена Геннадіївна

Рецензент:
ЛОКТИКОВА Тамара Миколаївна

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

зав. кафедри інженерії
програмного забезпечення

Тетяна ВАКАЛЮК

«18» вересня 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ТУРЧИН Микола Борисович**

Керівник роботи: **ЧИЖМОТРИЯ Олена Геннадіївна**

Тема роботи: **«Розробка інформаційної системи аналізу і прогнозування продажів в електронній комерції»**, затверджена Наказом закладу вищої освіти від **«18» вересня 2025 р. № 491/с**

Вихідні дані для роботи: **набір історичних даних транзакцій інтернет-магазину; методи статистичного аналізу та машинного навчання; програмні засоби розробки Python, Streamlit;**

Консультанти з магістерської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О.Г.	22.09.2025р.	22.09.2025р.
2	Чижмотря О.Г.	22.09.2025р.	22.09.2025р.
3	Чижмотря О.Г.	22.09.2025р.	22.09.2025р.

РЕФЕРАТ

Кваліфікаційна робота освітнього ступеня «магістр» присвячена розробці інформаційної системи аналізу та прогнозування продажів в електронній комерції з метою підвищення ефективності управління товарними запасами. Пояснювальна записка складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. Пояснювальна записка до кваліфікаційної магістерської роботи містить 93 сторінки, 10 ілюстрацій, 4 таблиці, 4 додатки та 32 посилання на зовнішні джерела.

Метою роботи є підвищення ефективності управління запасами інтернет-магазину шляхом розробки інформаційної системи прогнозування продажів, яка використовує гібридний підхід на основі методів машинного навчання та спеціалізованих статистичних алгоритмів.

У роботі розглянуто особливості процесів управління продажами та товарними запасами в електронній комерції. Проаналізовано методи аналізу й прогнозування часових рядів попиту в умовах стохастичної невизначеності. Реалізовано інформаційну систему з адаптивним вибором методу прогнозування залежно від характеру попиту. Запропоновано використання двоетапної моделі Zero-Inflated XGBoost, а також адитивних регресійних моделей та спеціалізованих методів прогнозування переривчастого попиту. Проведено експериментальне тестування розробленої системи та оцінено точність прогнозів.

Ключові слова: ПРОГНОЗУВАННЯ ПРОДАЖІВ, ЕЛЕКТРОННА КОМЕРЦІЯ, МАШИННЕ НАВЧАННЯ, XGBOOST, PROPHET, SBA, STREAMLIT, PYTHON, ЧАСОВІ РЯДИ.

ABSTRACT

The qualification work for the educational degree "Master" is devoted to the development of an information system for analyzing and forecasting sales in e-commerce in order to increase the efficiency of inventory management. The explanatory note consists of an introduction, three sections, conclusions, a list of sources used and appendices. The explanatory note to the qualification master's thesis contains 93 pages, 10 illustrations, 4 tables, 4 appendices, and 32 links to external sources.

The purpose of the work is to increase the efficiency of inventory management of an online store by developing an information system for forecasting sales, which uses a hybrid approach based on machine learning methods and specialized statistical algorithms.

The work considers the features of sales and inventory management processes in e-commerce. Methods for analyzing and forecasting demand time series under conditions of stochastic uncertainty are analyzed. An information system with adaptive selection of the forecasting method depending on the nature of demand is implemented. The use of the two-stage Zero-Inflated XGBoost model, as well as additive regression models and specialized methods for forecasting intermittent demand is proposed. Experimental testing of the developed system was carried out and the accuracy of the forecasts was assessed.

Keywords: SALES FORECASTING, E-COMMERCE, MACHINE LEARNING, XGBOOST, PROPHET, SBA, STREAMLIT, PYTHON, TIME SERIES.

ЗМІСТ

РЕФЕРАТ.....	3
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	7
ВСТУП.....	8
РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ ПРОДАЖІВ В ЕЛЕКТРОННІЙ КОМЕРЦІЇ.....	10
1.1 Особливості процесів продажів в електронній комерції та задачі їх аналізу.....	10
1.2 Аналіз методів та моделей прогнозування часових рядів.....	12
1.3 Порівняльний аналіз існуючих інформаційних систем та інструментів прогнозування продажів.....	15
1.4 Постановка задачі дослідження.....	19
Висновки до розділу 1.....	22
РОЗДІЛ 2. МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ СИСТЕМИ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ ПРОДАЖІВ.....	24
2.1 Математична постановка задачі прогнозування та критерії оптимізації.. 24	
2.2 Розробка моделей прогнозування для гетерогенного асортименту.....	27
2.3 Алгоритмічне забезпечення формування простору ознак та попередньої обробки даних.....	31
2.4 Проєктування архітектури та програмна реалізація системи.....	34
Висновки до розділу 2.....	41
РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ ПРОГНОЗУВАННЯ.....	43
3.1. Вимоги до технічного забезпечення та умови експлуатації.....	43
3.2. Реалізація алгоритмічного забезпечення та налаштування моделей....	46
3.3 Опис інтерфейсу користувача та сценарій роботи системи.....	53

3.4. Результати тестування системи та оцінка точності прогнозів.....	63
Висновки до розділу 3.....	66
ВИСНОВКИ.....	68
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	70
ДОДАТКИ.....	75

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CSV – (англ. Comma-Separated Values) – формат файлів із розділенням значень комами

EDA – (англ. Exploratory Data Analysis) – розвідувальний аналіз даних

ERP – (англ. Enterprise Resource Planning) – система планування ресурсів підприємства

MAE – (англ. Mean Absolute Error) – середня абсолютна похибка

MAPE – (англ. Mean Absolute Percentage Error) – середня абсолютна відсоткова похибка

ML – (англ. Machine Learning) – машинне навчання

RMSE – (англ. Root Mean Square Error) – корінь середньоквадратичної похибки

SBA – (англ. Syntetos-Boylan Approximation) – апроксимація Сінтетоса–Бойлана

SKU – (англ. Stock Keeping Unit) – складська одиниця обліку товару

WAPE – (англ. Weighted Absolute Percentage Error) – зважена абсолютна відсоткова похибка

XGBoost – (англ. eXtreme Gradient Boosting) – алгоритм градієнтного бустингу над деревами рішень

ВСТУП

Сучасний розвиток електронної комерції характеризується стрімким зростанням обсягів даних, розширенням товарного асортименту та високою конкуренцією. У цих умовах ефективне управління товарними запасами стає критичним фактором успіху бізнесу. Специфікою онлайн-рітейлу є наявність так званого «довгого хвоста» (Long Tail) — великої кількості товарів з переривчастим або рідкісним попитом, для яких класичні методи прогнозування (середнє, ковзне середнє) виявляються неефективними. Неточні прогнози призводять до двох основних проблем: дефіциту товару (втрата прибутку та клієнтів) або надлишку запасів (заморожування оборотних коштів). Це зумовлює необхідність розробки спеціалізованих інформаційних систем, здатних застосовувати сучасні методи машинного навчання для високоточного прогнозування в умовах невизначеності.

Мета і завдання дослідження. Метою роботи є підвищення ефективності процесів управління запасами в електронній комерції шляхом розробки інформаційної системи аналізу та прогнозування продажів. Для досягнення мети поставлено такі завдання:

1. Проаналізувати особливості процесів продажів в e-commerce та існуючі методи прогнозування часових рядів.
2. Обґрунтувати вибір математичних моделей для роботи з регулярним та переривчастим попитом (Prophet, SBA, XGBoost).
3. Спроекувати архітектуру інформаційної системи та розробити алгоритми попередньої обробки даних.
4. Здійснити програмну реалізацію системи з використанням мови Python та фреймворку Streamlit.
5. Провести експериментальне тестування моделей на реальних даних та оцінити їх точність.

Об'єктом дослідження є процеси управління продажами та запасами на підприємствах електронної комерції.

Предметом дослідження є методи, моделі та програмні засоби аналізу і прогнозування стохастичного попиту.

Наукова новизна одержаних результатів полягає у вдосконаленні підходу до прогнозування продажів гетерогенного асортименту шляхом комбінування двоетапної моделі машинного навчання (Zero-Inflated XGBoost) для активних товарів та методу апроксимації Сінтетоса-Бойлана (SBA) для товарів з переривчастим попитом, що забезпечило підвищення точності прогнозу порівняно з базовими методами.

Практичне значення одержаних результатів полягає у розробленому програмному комплексі, який є готовим інструментом для менеджерів із закупівель, що дозволяє автоматизувати розрахунок страхових запасів, візуалізувати тренди продажів та експортувати прогнози для подальшого використання в ERP-системах.

Апробація результатів. За темою кваліфікаційної магістерської роботи було опубліковано 3 тези конференцій [1, 2, 3].

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ ПРОДАЖІВ В ЕЛЕКТРОННІЙ КОМЕРЦІЇ

1.1 Особливості процесів продажів в електронній комерції та задачі їх аналізу

Сучасний етап розвитку світової економіки характеризується інтенсивною цифровізацією торгівельних процесів та зростанням ролі електронної комерції як одного з ключових каналів реалізації товарів і послуг. Електронна комерція упродовж останніх десятиліть еволюціонувала від альтернативної форми збуту до повноцінної, а в багатьох сегментах – домінуючої моделі взаємодії між виробниками, постачальниками та кінцевими споживачами. Така трансформація супроводжується накопиченням значних обсягів даних, які відображають як безпосередні транзакції, так і поведінкові характеристики користувачів, що створює передумови для застосування методів аналітики та інтелектуального аналізу даних [4].

Процеси продажів в електронній комерції суттєво відрізняються від традиційного офлайн-рітейлу, що зумовлює підвищену складність їх аналізу та прогнозування. Однією з ключових особливостей є значна ширина асортименту, адже сучасні інтернет-магазини часто оперують тисячами або десятками тисяч товарних позицій, кожна з яких має власну динаміку попиту. Відповідно до концепції «довгого хвоста», більшість таких товарів характеризується низькою та нерегулярною частотою продажів. У результаті часові ряди попиту для окремих товарів містять значну кількість нульових значень, що призводить до проблеми розрідженості даних та ускладнює використання класичних статистичних підходів до аналізу [4].

Додатковим фактором що ускладнює є висока нестаціонарність продажів в електронній комерції. Обсяги реалізації товарів значною мірою залежать від зовнішніх впливів, серед яких особливу роль відіграють сезонні коливання, календарні події та маркетингові активності. Пікові періоди, пов'язані з розпродажами, святами або спеціальними акціями, здатні

формувати різкі та короткотривалі сплески попиту, які складно передбачити виключно на основі історичних даних [5]. Крім того, характерною рисою електронної комерції є швидка зміна асортименту, що проявляється у частій появі нових товарів та виведенні з обігу застарілих позицій [4]. Це зумовлює проблему холодного старту, за якої прогнозування необхідно здійснювати за умов обмеженої або відсутньої історії спостережень.

Доцільно доповнити аналіз особливостей продажів поняттям життєвого циклу товару, який у контексті електронної комерції має скорочений та нестабільний характер. Для різних стадій життєвого циклу (виведення на ринок, зростання, зрілість, спад) характерні принципово різні патерни попиту, що унеможлиблює застосування єдиного підходу до прогнозування [6]. Особливо складними для моделювання є початкові етапи, коли історія продажів є обмеженою, а вплив маркетингових факторів — домінуючим.

У зв'язку з цим дедалі більшого значення набуває автоматизований аналіз продажів, який поєднує статистичні методи з алгоритмами машинного навчання та дозволяє адаптувати стратегію прогнозування до поточного стану товару. Використання інформаційних систем прогнозування забезпечує не лише зменшення похибки оцінки попиту, але й створює основу для підтримки управлінських рішень у режимі, близькому до реального часу.

У таких умовах особливої актуальності набуває задача ефективного управління товарними запасами. Помилки у прогнозуванні попиту можуть призводити до суттєвих економічних втрат, природа яких в електронній комерції є дуалістичною [7]. Дефіцит товарів на складі спричиняє не лише пряме недоотримання прибутку від втрачених транзакцій, але й несе загрозу довгострокових репутаційних ризиків. В умовах високої конкуренції та низької вартості перемикання користувач, не знайшовши потрібний товар, миттєво переходить на сайт конкурента, що часто призводить до втрати клієнта назавжди [5]. Більше того, алгоритми ранжування пошукових систем та маркетплейсів автоматично песимізують картки товарів з нульовими

залишками, знижуючи їх видимість у видачі, що вимагатиме додаткових маркетингових бюджетів для відновлення позицій після поповнення складу.

Задачі аналізу даних про продажі охоплюють декілька рівнів аналітики, починаючи з узагальнення історичної інформації та виявлення загальних тенденцій і завершуючи побудовою прогнозних моделей для оцінки майбутнього попиту. Особливе місце серед них займає предиктивний аналіз, який передбачає використання математичних моделей і алгоритмів машинного навчання для виявлення прихованих закономірностей у часових рядах [7]. Саме цей напрямок є найбільш наукомісткий, оскільки потребує врахування стохастичної природи попиту, впливу численних факторів та здатності моделей адаптуватися до зміни ринкових умов.

Таким чином, розробка інформаційної системи аналізу і прогнозування продажів в електронній комерції потребує комплексного підходу, який поєднує методи статистичного аналізу, машинного навчання та спеціалізовані моделі для роботи з переривчастим попитом. Актуальним науковим завданням є створення алгоритмів, здатних забезпечити достатню точність прогнозування для широкого спектра товарних позицій за умов нестабільності та нерівномірності попиту, що визначає подальший напрям дослідження у межах даної роботи.

1.2 Аналіз методів та моделей прогнозування часових рядів

Проблема прогнозування обсягів продажів у сучасній науковій літературі традиційно розглядається в контексті аналізу часових рядів, що являють собою послідовність статистичних даних, зібраних у хронологічному порядку. Методологічний апарат розв'язання цієї задачі пройшов тривалий еволюційний шлях від найпростіших евристик до складних ансамблевих моделей машинного навчання. Вибір конкретного математичного інструментарію безпосередньо залежить від статистичних властивостей досліджуваних процесів, таких як стаціонарність, наявність сезонних компонент та характер розподілу попиту [8].

Фундаментальну основу прогнозування в економічних системах складають класичні статистичні методи. Історично першими набули поширення методи експоненційного згладжування, зокрема моделі Хольта-Вінтерса, які дозволяють декомпонувати часовий ряд на рівень, тренд та сезонність. Подальшим розвитком цього напрямку стало створення методології Бокса-Дженкінса та авторегресійних моделей інтегрованого ковзного середнього ARIMA. Ці підходи демонструють високу ефективність при роботі з регулярними, стабільними даними, де минулі значення є надійним предиктором майбутніх [9]. Проте в умовах електронної комерції їх застосування нашоується на обмеження: жорстка вимога до стаціонарності ряду та нездатність гнучко моделювати складну сезонність, пов'язану з рухомими святами або маркетинговими активностями. Частковим вирішенням цієї проблеми є використання узагальнених адитивних моделей GAM, які лежать в основі сучасних бібліотек на кшталт Prophet [9], дозволяючи моделювати нелінійні тренди та ефекти свят як окремі компоненти регресійного рівняння.

Окремий і вкрай важливий клас задач виникає при аналізі товарів з переривчастим попитом, що складають значну частину асортименту інтернет-магазинів. Для таких часових рядів, насичених нульовими значеннями, використання стандартних метрик мінімізації середньоквадратичної помилки призводить до зміщених прогнозів, що прагнуть до нуля або константи. Для вирішення цієї проблеми у науковій практиці використовуються спеціалізовані методи, що розділяють прогноз на дві складові: прогнозування інтервалу між покупками та обсягу покупки. Найбільш відомим є метод Кростона, однак більш точні результати на практиці демонструє апроксимація Сінтетоса-Бойлана. Цей метод вводить коригувальний коефіцієнт, що нівелює зміщення оригінального алгоритму Кростона, і на сьогодні вважається одним із найбільш поширених підходів у практиці прогнозування для прогнозування попиту на товари групи C та Z [10].

Сучасний етап розвитку предиктивної аналітики нерозривно пов'язаний із методами машинного навчання, серед яких особливе місце займає градієнтний бустинг над вирішальними деревами. На відміну від класичних моделей, які розглядають кожен часовий ряд ізольовано, алгоритми бустингу, такі як XGBoost, дозволяють будувати глобальні моделі, що навчаються на даних усього асортименту одночасно [8]. Головною перевагою цього підходу є можливість включення у модель необмеженої кількості екзогенних факторів: цін, категорійних ознак, інформації про промо-акції та календарних фічів. Більше того, гнучкість архітектури бустингу дозволяє реалізовувати складні стратегії прогнозування, наприклад, двоступеневі моделі Zero-Inflated, де на першому етапі класифікатор визначає ймовірність здійснення продажу, а на другому – регресор оцінює його обсяг [9].

Окремої уваги в задачах прогнозування продажів потребує питання вибору метрик оцінювання якості моделей. Традиційні показники, такі як MSE або MAPE, не завжди адекватно відображають практичну цінність прогнозу в умовах розріджених часових рядів і наявності нульових значень [8]. У контексті управління запасами більш інформативними є агреговані метрики, орієнтовані на сумарну помилку прогнозу, які безпосередньо пов'язані з фінансовими втратами підприємства. Це обумовлює необхідність використання спеціалізованих критеріїв якості, адаптованих до специфіки електронної комерції.

Разом із підвищенням точності прогнозування зростає і складність моделей, що породжує проблему інтерпретованості результатів. У прикладних задачах управління запасами важливим є не лише числове значення прогнозу, але й розуміння причин його формування, зокрема впливу сезонності, цінових змін або маркетингових подій. Класичні статистичні та адитивні регресійні моделі забезпечують прозору структуру та дозволяють аналітику інтерпретувати окремі компоненти прогнозу, тоді як складні ансамблеві алгоритми машинного навчання часто функціонують як «чорна

скринька» [10]. Це обумовлює необхідність пошуку балансу між точністю прогнозу та можливістю його аналітичного пояснення.

Додатковим ускладнювальним чинником є масштабованість прогнозних рішень в умовах великого та різноманітного асортименту електронної комерції. Побудова окремої моделі для кожного товару є обчислювально затратною та важко підтримуваною, тоді як використання єдиної глобальної моделі не завжди дозволяє врахувати індивідуальні особливості окремих часових рядів. У цьому контексті особливої актуальності набувають адаптивні підходи, що поєднують глобальне навчання з локальною специфікою попиту та автоматизують вибір моделі залежно від статистичних характеристик конкретного товару.

Таким чином, аналіз предметної області свідчить про те, що універсального алгоритму, здатного однаково ефективно прогнозувати попит для всіх товарних категорій, не існує. Оптимальним вбачається застосування гібридного підходу, що поєднує інтерпретованість адитивних моделей для виявлення сезонності, робастність методу SBA для переривчастого попиту та потужність градієнтного бустингу для врахування складних нелінійних залежностей.

1.3 Порівняльний аналіз існуючих інформаційних систем та інструментів прогнозування продажів

Стрімкий розвиток електронної комерції зумовив появу значної кількості програмних рішень, орієнтованих на аналіз та прогнозування продажів. Такі системи активно застосовуються у сфері управління ланцюгами постачання, планування запасів та підтримки прийняття управлінських рішень. Водночас більшість наявних інструментів реалізують обмежений набір методів прогнозування або орієнтовані на вузький клас задач, що не завжди відповідає специфіці інтернет-торгівлі з її гетерогенним асортиментом та нестабільною динамікою попиту.

Характерною особливістю корпоративних систем класу ERP та SCM є орієнтація на використання інтерактивних аналітичних панелей, які слугують основним інструментом взаємодії користувача з результатами прогнозування. Такі дашборди забезпечують агреговане представлення великих обсягів даних, поєднуючи часові графіки, показники точності та сценарні порівняння, що дозволяє менеджерам оперативно оцінювати стан попиту та приймати обґрунтовані управлінські рішення без безпосереднього втручання в алгоритмічну частину системи [11].

Дашборд SAP Integrated Business Planning (рис. 1.1) забезпечує візуалізацію результатів прогнозування попиту з можливістю порівняння альтернативних сценаріїв управління запасами. Представлені графіки та агреговані показники дозволяють оцінити відхилення між сценаріями, аналізувати вплив змін цінової або маркетингової стратегії та підтримувати прийняття рішень у межах інтегрованого корпоративного планування.

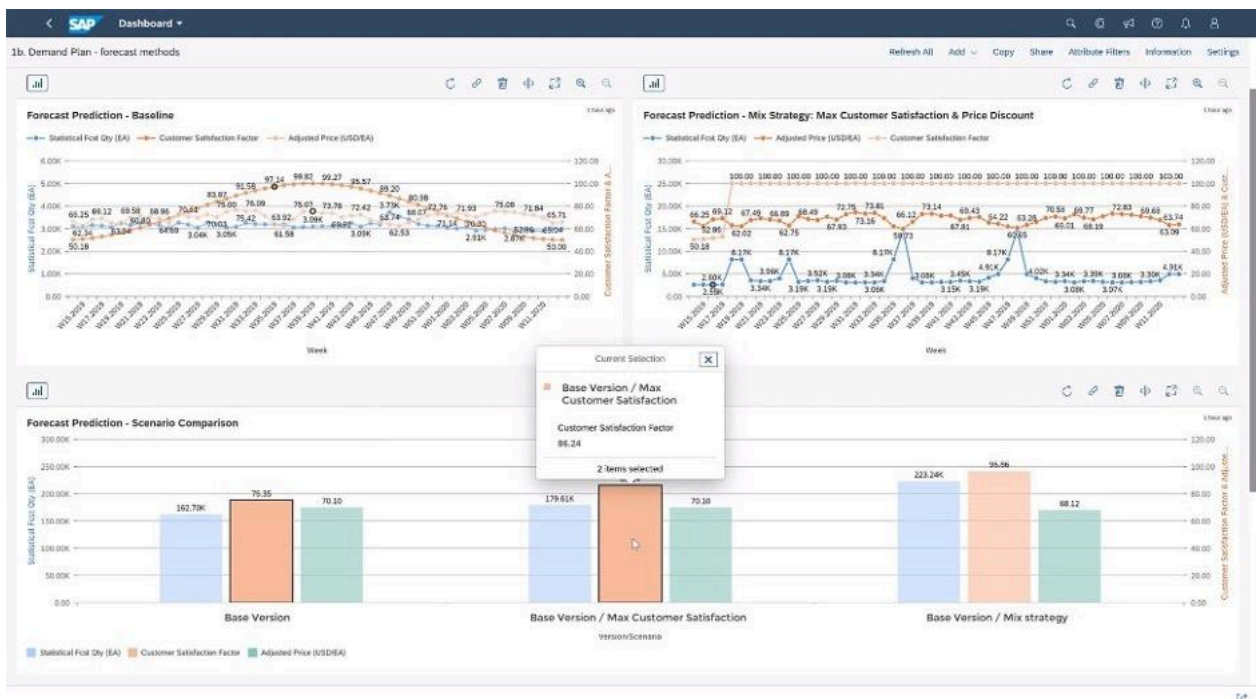


Рис 1.1. Дашборд SAP Integrated Business Planning

На дашборді Oracle Demand Management Cloud (рис. 1.2) представлено інструменти контролю точності прогнозів та аналізу попиту за сегментами. Візуалізація включає показники прогнозної точності, розподіл товарів за

категоріями та порівняння планових і фактичних значень, що сприяє оперативному виявленню проблемних зон у процесах управління попитом.

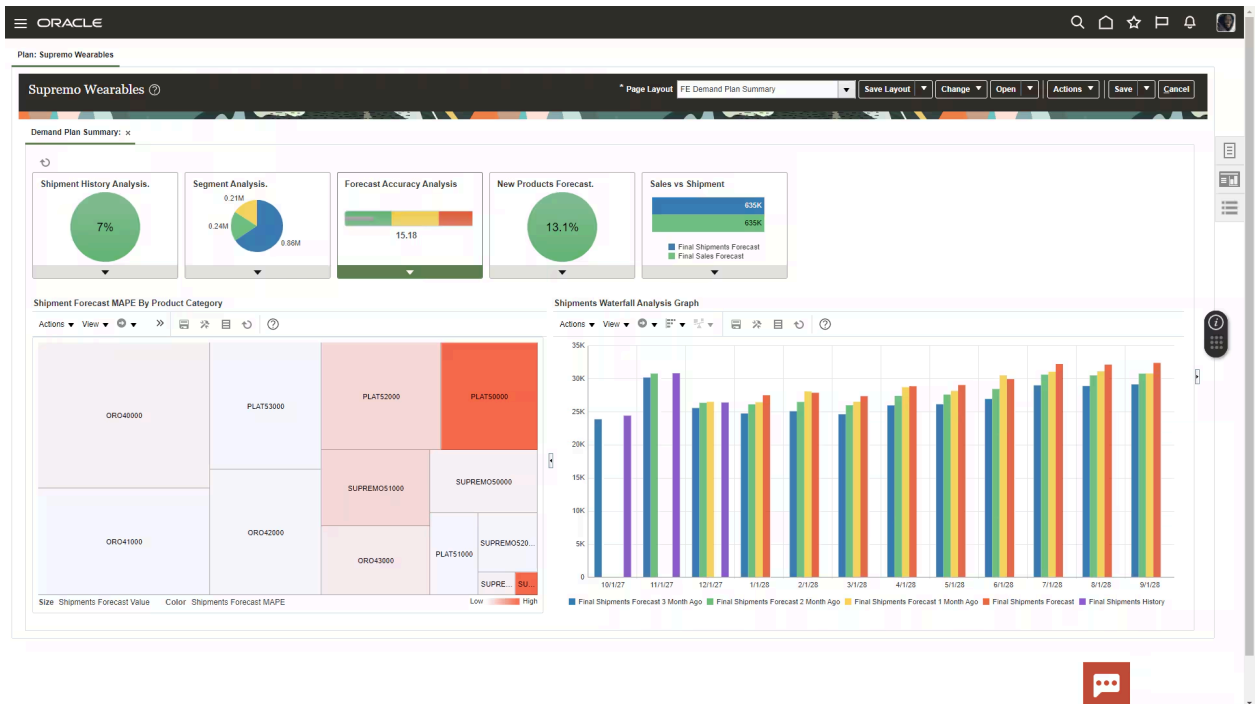


Рис 1.2. Дашборд Oracle Demand Management Cloud

Незважаючи на розвинені можливості візуалізації та інтеграції з корпоративними даними, функціональність дашбордів у складі ERP- та SCM-рішень значною мірою орієнтована на підтримку стандартних процесів планування [12]. Користувач, як правило, працює з уже сформованими прогнозами та агрегованими показниками, тоді як механізми побудови моделей і вибору алгоритмів залишаються прихованими на рівні системи. Це обмежує можливості гнучкого налаштування прогнозування відповідно до специфіки окремих товарів або нестандартних бізнес-сценаріїв.

Разом із тим корпоративні ERP-рішення мають низку суттєвих обмежень у контексті електронної комерції. Зокрема, використовувані моделі прогнозування характеризуються обмеженою гнучкістю та слабкою адаптивністю до нестабільного й переривчастого попиту, притаманного значній частині товарних позицій онлайн-магазинів. Крім того, впровадження та супровід таких систем пов'язані з високою вартістю, складністю

налаштування та необхідністю залучення спеціалізованого персоналу, що обмежує їх застосування для малих і середніх підприємств [12].

Окрему нішу займають спеціалізовані комерційні системи управління запасами, зокрема Netstock (рис. 1.3), Lokad, Forecastly та подібні SaaS-сервіси. У таких продуктах значна увага приділяється прогнозуванню переривчастого попиту (intermittent demand). Для цього використовуються алгоритми Кростона та його модифікації, а також евристичні правила класифікації товарів за стабільністю продажів. Застосування подібних підходів дозволяє зменшити ризики дефіциту або надлишку запасів для товарів із низькою частотою продажів та підвищити рівень сервісу.

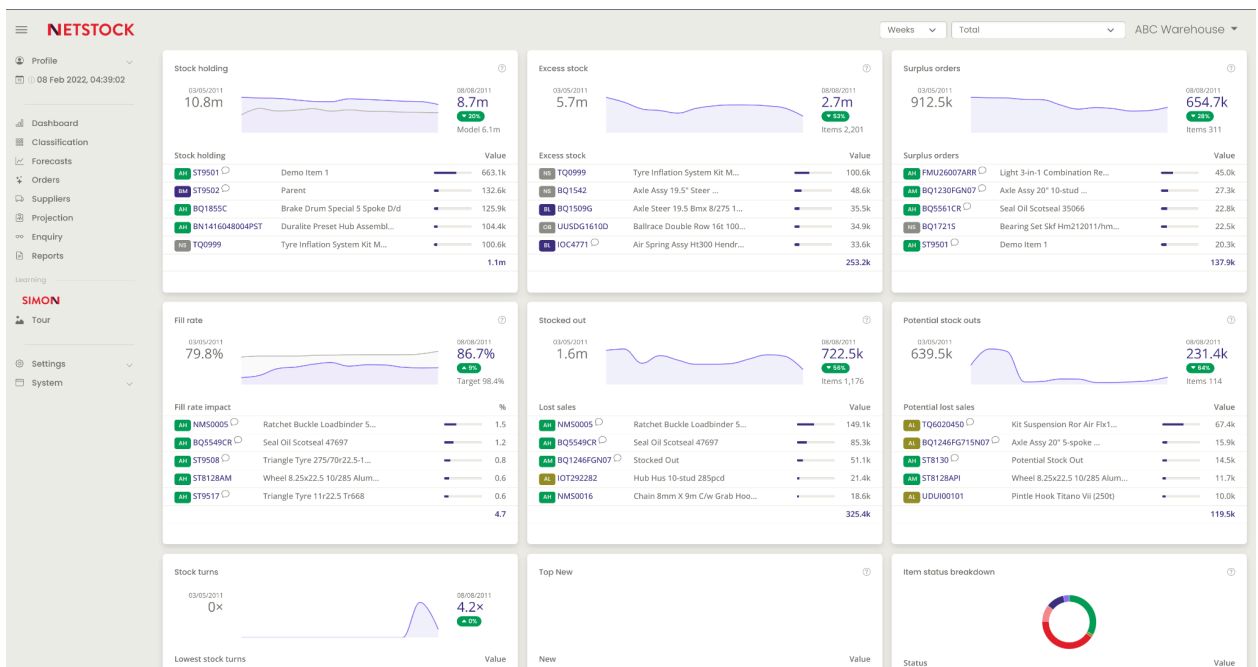


Рис 1.3. Дашборд Netstock

Водночас більшість спеціалізованих систем прогнозування запасів мають закриту архітектуру та обмежену прозорість алгоритмів. Користувач отримує готові результати без можливості детального аналізу моделей, комбінування різних методів прогнозування або адаптації алгоритмів до специфічних бізнес-умов. Крім того, такі рішення часто орієнтовані на стандартні сценарії управління запасами та не забезпечують гнучкої інтеграції з власними аналітичними модулями підприємства [13].

Аналіз існуючих інформаційних систем прогнозування свідчить, що більшість рішень реалізують або єдиний універсальний підхід для всіх товарів, або покладаються на ручний вибір моделей аналітиком. Такий підхід не враховує гетерогенність асортименту електронної комерції, де одночасно присутні як товари зі стабільним регулярним попитом, так і позиції з вираженою переривчастою динамікою продажів.

Порівняння корпоративних ERP-рішень і спеціалізованих SaaS-систем свідчить про наявність концептуального компромісу між масштабованістю, гнучкістю та прозорістю прогнозних моделей [12]. ERP-платформи забезпечують інтеграцію та централізоване планування, однак є малочутливими до особливостей окремих часових рядів, тоді як спеціалізовані сервіси ефективно працюють з переривчастим попитом, але обмежують можливості адаптації та аналітичного контролю. Це створює передумови для розробки гібридних інформаційних систем, які поєднують автоматизований вибір моделей із відкритою архітектурою та орієнтацією на специфіку електронної комерції [14].

Таким чином, проведений порівняльний аналіз дозволяє зробити висновок про доцільність розробки гнучкої інформаційної системи прогнозування продажів, здатної поєднувати різні методи обробки часових рядів та автоматично обирати адекватний алгоритм залежно від характеристик попиту конкретного товару. Такий підхід забезпечує баланс між інтерпретованістю прогнозів, точністю оцінок та адаптивністю до змінних умов електронної комерції, що і зумовлює актуальність подальших досліджень у цьому напрямку.

1.4 Постановка задачі дослідження

Аналіз підходів та програмних рішень у сфері прогнозування продажів дозволяє сформулювати низку ключових проблем, що залишаються недостатньо вирішеними. Основною з них є відсутність універсальної моделі прогнозування, здатної забезпечувати стабільно високу точність для всіх

товарних позицій асортименту, які суттєво відрізняються за характером попиту, сезонністю та частотою продажів. У зв'язку з цим актуальним є завдання розробки інформаційної системи, що поєднує різні підходи до аналізу часових рядів та забезпечує адаптивний вибір моделі прогнозування залежно від властивостей конкретного товару.

Об'єктом дослідження в даній роботі є процеси формування та зміни попиту на товари в системах електронної комерції. Предметом дослідження є методи аналізу та прогнозування часових рядів продажів, а також алгоритми обробки переривчастого та нерегулярного попиту з метою підтримки управління запасами.

Метою дослідження є розробка та реалізація інформаційної системи аналізу і прогнозування продажів в електронній комерції, яка забезпечує підвищення точності прогнозів за рахунок використання гібридного підходу, що поєднує статистичні методи прогнозування переривчастого попиту, адитивні моделі сезонності та алгоритми машинного навчання.

У межах дослідження задача прогнозування розглядається як задача аналізу множини часових рядів, що відрізняються за рівнем регулярності, сезонності та інтенсивності попиту. Кожен часовий ряд відповідає окремій товарній позиції та характеризується індивідуальними статистичними властивостями, що унеможливорює застосування єдиного універсального алгоритму прогнозування. Таким чином, предметом дослідження є не лише побудова окремих моделей прогнозування, але й розробка механізму їх адаптивного вибору залежно від характеристик попиту.

Важливою складовою постановки задачі є попередня класифікація часових рядів попиту з метою визначення доцільного підходу до прогнозування. У межах дослідження передбачається виділення груп товарів зі стабільним регулярним попитом, сезонною динамікою та переривчастим характером продажів. Така класифікація слугує основою для вибору відповідної моделі прогнозування та дозволяє підвищити загальну точність системи за рахунок врахування специфіки кожного класу товарів.

Для досягнення поставленої мети в роботі необхідно вирішити такі задачі дослідження. По-перше, провести аналіз статистичних властивостей часових рядів продажів, зокрема оцінити рівень переривчастості попиту, наявність сезонних компонент та ступінь варіативності обсягів продажів для різних товарних позицій. По-друге, дослідити ефективність застосування спеціалізованих методів прогнозування переривчастого попиту, зокрема методу Сінтетоса–Бойлана, для товарів з низькою частотою продажів. По-третє, реалізувати моделі прогнозування на основі узагальнених адитивних моделей, представлених бібліотекою Prophet, з метою виявлення та моделювання сезонних і календарних закономірностей попиту. По-четверте, розробити модель прогнозування на основі градієнтного бустингу над вирішальними деревами XGBoost, здатну враховувати нелінійні залежності та реалізовувати двоступеневу стратегію прогнозування для нуль-інфльованих часових рядів.

Інформаційна система, що розробляється в межах дослідження, включає підготовку часових рядів продажів, побудову прогнозних моделей, оцінювання якості прогнозів та вибір оптимальної моделі для кожної товарної позиції. Формалізовано вхідними даними системи є історичні дані продажів у вигляді денних часових рядів для окремих SKU, а також додаткові календарні та агреговані статистичні ознаки. Вихідними даними є прогнозовані значення обсягів продажів на заданий горизонт часу та інтегральні показники якості прогнозування.

Оцінювання ефективності розробленої системи здійснюється з використанням метрик, орієнтованих на практичні потреби управління запасами, зокрема середньої абсолютної похибки MAE, кореня середньоквадратичної похибки RMSE, зваженої абсолютної відносної похибки WAPE, а також похибки прогнозування сумарного попиту на фіксованому часовому горизонті. Використання таких метрик дозволяє оцінити не лише точність щоденних прогнозів, але й адекватність прогнозів з точки зору прийняття управлінських рішень.

З огляду на прикладний характер дослідження, до розроблюваної інформаційної системи висуваються вимоги не лише щодо точності прогнозування, але й щодо інтерпретованості результатів, масштабованості та можливості практичного використання у бізнес-середовищі. Система повинна забезпечувати автоматизовану підготовку даних, оцінювання якості прогнозів та формування рекомендацій для управління запасами без необхідності ручного втручання з боку аналітика.

Результатом виконання дослідження має стати програмний комплекс, що реалізує гібридний підхід до прогнозування продажів в електронній комерції та забезпечує можливість практичного використання отриманих прогнозів для оптимізації управління товарними запасами.

Висновки до розділу 1

У першому розділі магістерської роботи було здійснено комплексний теоретичний аналіз проблеми прогнозування продажів у сфері електронної комерції, досліджено специфіку предметної області та узагальнено сучасні наукові підходи до розв'язання задачі управління запасами.

У ході дослідження встановлено, що процеси продажів в e-commerce мають складну стохастичну природу, яка суттєво відрізняється від традиційного рітейлу. Ключовими факторами, що ускладнюють прогнозування, визначено: значну гетерогенність асортименту (наявність товарів з різними життєвими циклами), високий рівень переривчастості попиту (intermittent demand) для товарів «довгого хвоста», а також суттєвий вплив зовнішніх збурень — сезонності, календарних подій та маркетингових активностей. Доведено, що ігнорування цих особливостей при використанні класичних методів усереднення призводить до значних похибок та фінансових втрат.

Проведений критичний огляд існуючого програмного забезпечення, зокрема модулів прогнозування у складі ERP-систем та хмарних SaaS-платформ, виявив низку обмежень. Встановлено, що більшість готових

рішень або базуються на застарілих статистичних моделях (ARIMA, експоненційне згладжування), які є неефективними для розріджених даних, або мають високу вартість впровадження та закриту архітектуру, що унеможлиблює їх адаптацію під специфічні бізнес-процеси. Це обґрунтувало необхідність розробки власної спеціалізованої інформаційної системи.

У результаті порівняльного аналізу математичного апарату було обґрунтовано доцільність використання гібридного підходу, який поєднує переваги різних класів алгоритмів. Визначено, що для товарів зі стабільним попитом та вираженою сезонністю найбільш ефективними є адитивні регресійні моделі (типу Prophet) та методи градієнтного бустингу (XGBoost), здатні враховувати нелінійні залежності. Водночас для товарів категорії С з нерегулярними продажами доцільним є застосування спеціалізованих ймовірнісних методів, зокрема апроксимації Сінтетоса-Бойлана (SBA).

На основі проведеного аналізу сформульовано постановку задачі дослідження та визначено вимоги до проєктованої системи. Вона повинна забезпечувати автоматизовану класифікацію часових рядів, попередню обробку даних (ETL), генерацію ознак та адаптивний вибір оптимальної моделі прогнозування. Отримані в першому розділі теоретичні положення та методологічні висновки слугують фундаментом для проєктування архітектури та програмної реалізації інформаційної системи, опис яких буде наведено в наступних розділах роботи.

РОЗДІЛ 2. МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ СИСТЕМИ АНАЛІЗУ ТА ПРОГНОЗУВАННЯ ПРОДАЖІВ

2.1 Математична постановка задачі прогнозування та критерії оптимізації

Задача прогнозування продажів в електронній комерції формалізується як задача аналізу дискретного стохастичного процесу, що описує зміну попиту на окремі товарні позиції в часі (формула 2.1). Для кожного товару s з множини всіх товарів S задано часовий ряд продажів:

$$Y_s = y_{s,1}, y_{s,2}, \dots, y_{s,T}, \quad (2.1)$$

де $y_{s,T}$ – кількість проданих одиниць товару s у момент часу;

T – довжина історичного періоду спостережень.

Метою прогнозування є визначення майбутніх значень попиту на горизонті h шляхом побудови прогнозової функції (формула 2.2):

$$\hat{y}_{s,t+h} = F_s(X_{s,t}), \quad (2.2)$$

де $\hat{y}_{s,t+h}$ – прогнозований обсяг продажів товару s у момент часу $t + h$;

$X_{s,t}$ – вектор ознак, сформований на основі історичних продажів та зовнішніх факторів;

$F_s(\cdot)$ – модель прогнозування, параметри якої визначаються в процесі навчання.

З формальної точки зору задача прогнозування продажів у електронній комерції належить до класу задач стохастичного прогнозування багатовимірних часових рядів. На відміну від класичних задач аналізу одного часового ряду, у даному випадку розглядається сукупність взаємно незалежних, але статистично неоднорідних часових рядів, кожен з яких відповідає окремій товарній позиції. Така постановка задачі наближує проблему прогнозування продажів до задач аналізу панельних даних (panel data), де необхідно одночасно враховувати як індивідуальні властивості

окремих рядів, так і загальні закономірності попиту в межах усього асортименту [15].

Особливістю задачі прогнозування продажів в електронній комерції є гетерогенність асортименту. Різні товарні позиції демонструють істотно відмінні статистичні властивості часових рядів. Частина товарів характеризується стабільним регулярним попитом, тоді як для значної кількості SKU типовими є тривалі періоди нульових продажів. Така структура даних зумовлює необхідність використання різних моделей прогнозування для різних груп товарів.

У межах дослідження розглядається множина прогнозних моделей (формула 2.3) :

$$F = \{ F_1, \dots, F_k \}, \quad (2.3)$$

де F_i – модель прогнозування з розглянутого набору;

k – загальна кількість моделей-кандидатів, що використовуються в дослідженні;

Для кожного товару обирається модель, що забезпечує мінімальну похибку прогнозу відповідно до заданого критерію оптимальності.

Для оцінки якості прогнозних моделей у роботі використовується зважена абсолютна відносна похибка (Weighted Absolute Percentage Error, WAPE), яка визначається як (формула 2.4):

$$WAPE = (\sum |y_t - \hat{y}_t|) / (\sum |y_t|), \quad (2.4)$$

де y_t – фактичне значення попиту у момент часу t ;

$\hat{y}_t$ – прогнозоване значення попиту у момент часу t .

На відміну від класичних метрик, таких як MAPE або MSE, показник WAPE є стійким до нульових значень у часовому ряді та коректно відображає сумарну помилку прогнозу з точки зору бізнес-показників. Для додаткового аналізу локальних відхилень прогнозу від фактичних значень також використовується корінь середньоквадратичної помилки, що дозволяє оцінити точність прогнозу на щоденному рівні.

Вибір критерію оптимізації у задачі прогнозування має принципове значення, оскільки безпосередньо визначає характер помилок, які мінімізує модель. У прикладних задачах управління запасами помилка прогнозування трансформується у фінансові ризики дефіциту або надлишку товарів, тому мінімізація середньоквадратичної похибки не завжди є оптимальною. Застосування зважених агрегованих метрик, зокрема WAPE, дозволяє оцінювати сумарну похибку прогнозу з урахуванням реальних обсягів попиту та забезпечує узгодженість між математичною постановкою задачі та практичними цілями управління запасами [16].

Прогнозовані значення попиту використовуються для підтримки прийняття рішень щодо управління складськими запасами. Сумарний прогноз попиту на горизонті H визначається як (формула 2.5):

$$\hat{D}_{s,H} = \sum_{h=1}^H \hat{y}_{s,t+h}, \quad (2.5)$$

де $\hat{y}_{s,t+h}$ – прогнозоване значення попиту на товар s у момент часу $t + h$;

H – горизонт прогнозування;

$\hat{D}_{s,H}$ – сумарний прогноз попиту на товар s на горизонті H .

З метою компенсації невизначеності прогнозу вводиться страховий запас, який обчислюється за формулою (формула 2.6):

$$SS_s = Z \cdot \sigma_s \cdot \sqrt{H}, \quad (2.6)$$

де σ_s – стандартне відхилення попиту товару s ;

Z – коефіцієнт рівня сервісу;

H – довжина прогнозного горизонту;

SS_s – страховий запас товару s .

З урахуванням сумарного прогнозу попиту та величини страхового запасу рекомендований обсяг замовлення визначається за формулою (2.7):

$$Q_s = \hat{D}_{s,H} + SS_s, \quad (2.7)$$

де $\hat{D}_{s,H}$ – сумарний прогноз попиту на товар s на горизонті H ;

SS_s – страховий запас товару s ;

Q_s – рекомендований обсяг замовлення.

Таким чином, задача прогнозування у межах даної роботи розглядається не ізольовано, а як частина комплексної інформаційної системи аналізу та підтримки прийняття рішень у сфері управління запасами електронної комерції.

2.2 Розробка моделей прогнозування для гетерогенного асортименту

Вибір моделей прогнозування у межах даного дослідження зумовлений необхідністю охоплення різних типів попиту, характерних для асортименту електронної комерції. Застосування лише одного класу моделей призводить до систематичних похибок для окремих груп товарів, тому доцільним є формування обмеженого, але комплементарного набору алгоритмів [17]. До цього набору включено адитивні регресійні моделі для регулярного сезонного попиту, спеціалізовані методи для переривчастих часових рядів та алгоритми машинного навчання для складних нелінійних залежностей.

Практика електронної комерції свідчить, що асортимент інтернет-магазину є статистично неоднорідним: різні товарні позиції демонструють істотно відмінні характеристики попиту. Частина товарів характеризується стабільною динамікою реалізації з чітко вираженими сезонними коливаннями, тоді як значна кількість SKU має переривчастий попит, для якого типові тривалі інтервали відсутності продажів, що чергуються з поодинокими транзакціями. За таких умов побудова єдиної універсальної прогнозової моделі для всього асортименту є неефективною та призводить до зростання похибки прогнозу для окремих груп товарів [5].

З огляду на зазначене, у межах даного дослідження реалізовано комбінований підхід до прогнозування, який передбачає застосування різних

математичних моделей залежно від статистичних властивостей відповідних часових рядів. Алгоритмічне ядро розробленої системи складається з трьох класів моделей, кожна з яких орієнтована на окремий тип попиту та виконує специфічну роль у загальній архітектурі прогнозування.

Адитивна регресійна модель Prophet

Для товарів зі стабільною динамікою продажів та наявною сезонною структурою використовується адитивна регресійна модель на основі узагальнених адитивних моделей, реалізована в бібліотеці Prophet. Даний підхід ґрунтується на концепції узагальнених адитивних моделей, відповідно до якої часовий ряд попиту подається у вигляді суми незалежних компонент (формула 2.8) [18]:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon t, \quad (2.8)$$

де $g(t)$ описує довгостроковий тренд;

$s(t)$ – сезонну складову;

$h(t)$ – вплив календарних подій і свят;

εt – випадкову похибку.

Сезонна компонента апроксимується рядом Фур'є (формула 2.9), що дозволяє гнучко моделювати періодичні коливання різної природи, зокрема тижневі та річні цикли:

$$s(t) = \sum_{n=1}^N (a_n \cos(\frac{2\pi n t}{P}) + b_n \sin(\frac{2\pi n t}{P})), \quad (2.9)$$

де $s(t)$ – сезонна складова часового ряду попиту;

t – дискретний момент часу;

P – період сезонності;

N – кількість гармонік ряду Фур'є, що визначає ступінь деталізації сезонної компоненти;

a_n, b_n – коефіцієнти ряду Фур'є для n -тої гармоніки, які оцінюються в процесі навчання моделі;

Перевагою моделі Prophet є відсутність жорстких припущень щодо стаціонарності часового ряду, а також можливість явного врахування ефектів свят і спеціальних подій [18]. Завдяки цьому дана модель є доцільною для прогнозування попиту на товари з регулярним обігом та вираженою сезонністю.

Модель апроксимації Сінтетоса–Бойлана

Для товарних позицій із переривчастим попитом застосовується метод апроксимації Сінтетоса–Бойлана (Syntetos–Boylan Approximation, SBA), який є вдосконаленням класичного алгоритму Кростона. Основна ідея методу полягає у роздільному експоненційному згладжуванні двох стохастичних процесів: середнього обсягу продажу за умови його настання та середнього інтервалу між послідовними продажами [19].

У межах алгоритму оцінюються згладжені значення z_t – очікуваний обсяг попиту, та p_t – очікувана тривалість інтервалу між продажами. Прогноз середнього попиту на одиницю часу обчислюється за формулою (формула 2.10) [19]:

$$\hat{y}_{t+h} = (1 - \frac{\beta}{2}) \frac{z_t}{p_t}, \quad (2.10)$$

де $\hat{y}_{t+h}$ – прогноз середнього попиту на одиницю часу на горизонті h ;

z_t – згладжене значення обсягу попиту (очікуваний розмір продажу);

p_t – згладжена оцінка тривалості інтервалу між продажами;

$1 - \frac{\beta}{2}$ – параметр експоненційного згладжування інтервалів між продажами;

t – поточний момент часу;

h – горизонт прогнозування.

Завдяки своїй простоті, робастності та коректній роботі з розрідженими даними, метод SBA використовується в межах системи як базовий інструмент прогнозування інструмент прогнозування для товарів з низькою частотою продажів.

Незважаючи на ефективність спеціалізованих статистичних методів для переривчастого попиту, їх застосування обмежується випадками, коли динаміка продажів не залежить від значної кількості зовнішніх факторів. Для товарів із складною поведінкою попиту, що формується під впливом сезонності, маркетингових активностей та інших екзогенних змінних, доцільним є використання моделей машинного навчання, здатних враховувати нелінійні взаємозв'язки між ознаками [20]. Це обґрунтовує перехід до застосування градієнтного бустингу над деревами рішень.

Двоступенева модель градієнтного бустингу

Для товарів зі складною нелінійною динамікою попиту та залежністю від множини внутрішніх і зовнішніх факторів розроблено модель на основі алгоритму градієнтного бустингу над деревами рішень XGBoost. З урахуванням високої частки нульових значень у часових рядах реалізовано підхід Zero-Inflated, який передбачає декомпозицію задачі прогнозування на два послідовні етапи [20].

На першому етапі розв'язується задача бінарної класифікації, у межах якої оцінюється ймовірність настання факту продажу в заданий момент часу. На другому етапі регресійна модель, навчена виключно на ненульових спостереженнях, прогнозує очікувану інтенсивність попиту. Для стабілізації дисперсії та запобігання появи від'ємних прогнозів використовується логарифмічне перетворення цільової змінної.

Фінальний прогноз формується як добуток ймовірності продажу та відновленого значення обсягу попиту (формула 2.11):

$$\hat{y}_t = P(sale_t) \cdot (\exp(M_t) - 1), \quad (2.11)$$

$\hat{y}_t$ – фінальний прогноз попиту у момент часу t ;

$P(sale_t)$ – ймовірність настання продажу у момент часу t ;

M_t – прогнозоване значення логарифмованого обсягу попиту, отримане в межах моделі;

$\exp(M_t)$ – експоненціальна функція;

$(\exp(M_t) - 1)$ – використовується для повернення прогнозу з логарифмічного простору до початкової шкали значень.

Такий підхід дозволяє поєднати оцінку ймовірності виникнення продажу та умовний прогноз його обсягу.

Важливою особливістю реалізації є застосування прямої стратегії багатокрокового прогнозування, за якої для кожного горизонту планування навчається окрема пара моделей «класифікатор–регресор» [20]. Такий підхід дозволяє уникнути накопичення похибки, характерного для рекурсивних стратегій, і підвищує точність прогнозів на віддалених часових інтервалах.

Таким чином, поєднання адитивних регресійних моделей, спеціалізованих методів для переривчастого попиту та двоступеневих моделей машинного навчання формує адаптивну систему прогнозування, здатну ефективно працювати з гетерогенним асортиментом товарів електронної комерції.

2.3 Алгоритмічне забезпечення формування простору ознак та попередньої обробки даних

Ефективність застосування методів машинного навчання, зокрема алгоритм градієнтного бустингу, істотно залежить від якості вхідних даних. Безпосередньо історичні часові ряди продажів Y_s не містять у явному вигляді інформації про сезонні цикли, календарні ефекти або поведінкові патерни споживачів. У зв'язку з цим важливим етапом розробки системи є процедура формування простору ознак (Feature Engineering), яка полягає у перетворенні часових рядів у структуровану матрицю ознак X , що подається на вхід прогнозних моделей [21].

У розробленій системі вектор ознак X_t формується з трьох основних груп предикторів: календарних, статистичних та подієвих.

Кодування календарних та циклічних ознак

Продажі в електронній комерції характеризуються чітко вираженою тижневою та річною сезонністю, тому коректне представлення календарної дати є необхідною умовою для побудови адекватних прогнозних моделей [6]. Використання порядкових числових значень (наприклад, номер місяця) є некоректним для метричних алгоритмів, оскільки створює штучні розриви між кінцем та початком календарного циклу.

Для усунення цієї проблеми у системі застосовано тригонометричне циклічне кодування часових ознак. Кожен періодичний атрибут перетворюється на пару координат на одиничному колі за формулами (формула 2.11):

$$x_{sin} = \sin\left(\frac{2\pi k}{T}\right), x_{cos} = \cos\left(\frac{2\pi k}{T}\right), \quad (2.12)$$

де t – поточне значення часової змінної, а T – довжина відповідного циклу. Такий підхід забезпечує безперервність простору ознак та дозволяє моделі коректно інтерпретувати близькість дат.

Статистичні ознаки на ковзних вікнах

Для врахування локальних трендів та рівня волатильності попиту у системі використовуються статистичні ознаки, розраховані на основі ковзних вікон (Rolling Windows) [22]. Для кожного моменту часу обчислюються ковзні середні μ та стандартні відхилення σ на фіксованих інтервалах (формула 2.13).

$$\mu_{t,w} = \frac{1}{w} \sum_{i=1}^w y_{t-i}, \quad \sigma_{t,w} = \sqrt{\frac{1}{w} \sum_{i=1}^w (y_{t-i} - \mu_{t,w})^2}, \quad (2.13)$$

де $\mu_{t,w}$ – ковзне середнє значення попиту у момент часу t , обчислене на вікні довжини w ;

$\sigma_{t,w}$ – ковзне стандартне відхилення попиту у момент часу t , обчислене на вікні довжини w ;

y_{t-i} – фактичне значення попиту у момент часу $t - i$;

w — розмір ковзного вікна;

t — поточний момент часу.

Це дозволяє згладити випадкові флуктуації та виділити короткострокові й середньострокові тенденції.

Окремою важливою характеристикою для моделювання переривчастого попиту є частка ненульових продажів на ковзному вікні, яка слугує індикатором регулярності попиту та використовується як вхідний сигнал для класифікаційного етапу двоступеневої Zero-Inflated моделі.

Подієві ознаки та врахування зовнішніх факторів

Однією з характерних особливостей процесів продажів в електронній комерції є значний вплив зовнішніх календарних та маркетингових подій, які не проявляються безпосередньо у значеннях історичного попиту. Святкові періоди, розпродажі та підготовчі інтервали перед ними формують нетипові патерни поведінки споживачів, що супроводжуються різкими та короткочасними змінами обсягів продажів [4]. Ігнорування таких факторів призводить до систематичних похибок прогнозу, особливо для товарів із сезонною або акційною чутливістю.

Для врахування зазначених ефектів у розробленій системі реалізовано механізм формування подієвих ознак (event-based features), які інтегруються до загального вектору ознак X_t . Основою для побудови таких ознак є календарні дані та інформація про національні свята цільового ринку. У рамках реалізації використовується автоматизоване визначення святкових дат із застосуванням спеціалізованих довідників (бібліотека holidays), що дозволяє уникнути ручного кодування та забезпечує масштабованість системи.

Найпростішим типом подієвих ознак є бінарні індикатори, які набувають значення 1 у разі належності дати до святкового періоду та 0 – в

іншому випадку. Такі змінні дозволяють моделі явно фіксувати наявність аномального попиту та коригувати прогноз.

Разом з тим, практичний досвід свідчить, що зростання попиту часто починається заздалегідь і досягає максимуму не в день самої події, а в підготовчий період. Для моделювання цього ефекту введено безперервні ознаки, що відображають часову дистанцію до ключових подій. Наприклад, ознака `DaysToChristmas` визначається як різниця між датою Різдва та поточною датою спостереження. Таке представлення дозволяє алгоритмам машинного навчання виявляти поступове зростання попиту та формувати більш адекватні прогнози у передсвяткові періоди.

Подієві ознаки відіграють особливо важливу роль у моделях градієнтного бустингу, оскільки дозволяють враховувати нелінійні взаємодії між календарем, історичними лагами та статистичними характеристиками. У поєднанні з лаговими та ковзними ознаками вони забезпечують підвищення чутливості моделі до короткострокових змін ринкової ситуації без втрати стабільності прогнозу.

2.4 Проєктування архітектури та програмна реалізація системи

Обґрунтування математичних методів та алгоритмів прогнозування, виконане у попередньому підрозділі, формує теоретичний базис для створення системи. Наступним етапом є розробка програмної архітектури, яка дозволить інтегрувати обрані моделі у єдиний автоматизований комплекс. Нижче розглянуто компонентну структуру системи, схеми руху потоків даних та обґрунтування технологічних засобів реалізації.

Компонентна структура системи

Розроблена інформаційна система спроектована за модульним принципом із чітким розмежуванням відповідальностей між компонентами. Такий підхід забезпечує масштабованість, повторне використання коду та узгодженість результатів на етапах навчання й експлуатації моделей.

Компонентну структуру програмного комплексу представлено на рисунку 2.1 у вигляді UML-діаграми компонентів.

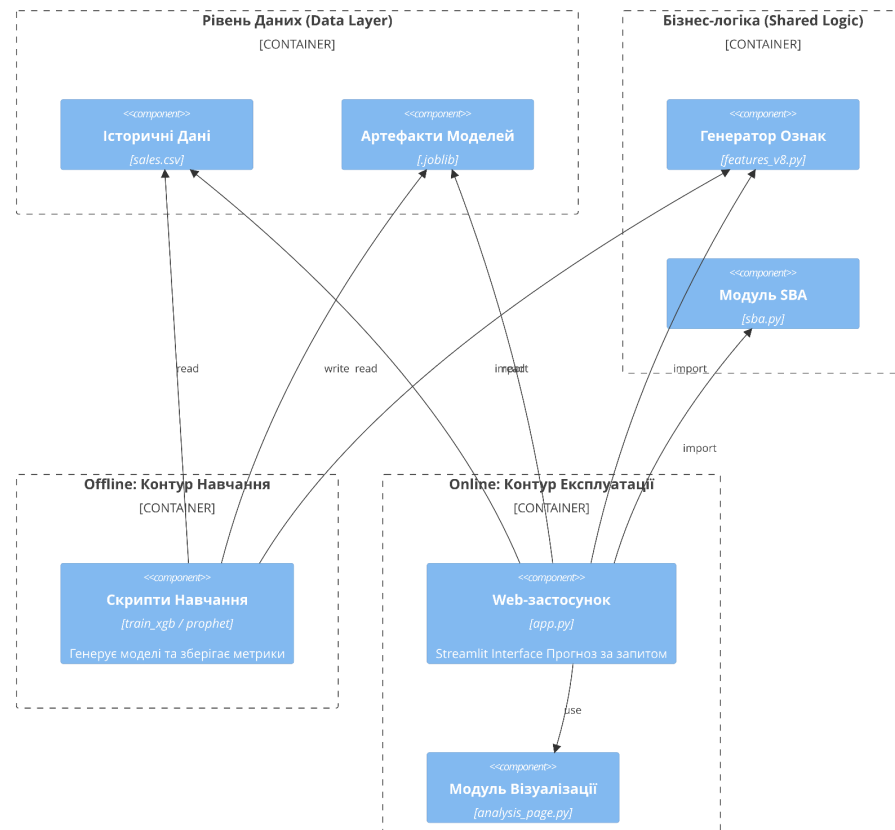


Рис 2.1. Компонентна діаграма архітектури програмної системи

Архітектура системи логічно поділена на декілька рівнів: рівень даних, рівень спільної бізнес-логіки, контур offline-навчання моделей та контур online-експлуатації.

До рівня даних належать компоненти, що відповідають за зберігання та надання інформації. Основним джерелом вхідних даних є компонент «Історичні дані», який містить щоденні обсяги продажів у вигляді CSV-файлів або таблиць бази даних. Результатом роботи контурів навчання є компонент «Артефакти моделей», у якому зберігаються навчені моделі та супровідні метадані у форматі `.joblib`. Такий підхід забезпечує відокремлення даних від логіки їх обробки та спрощує повторне використання моделей.

Рівень спільної бізнес-логіки представлений компонентами, що використовуються як у процесі навчання, так і під час прогнозування. Ключовим елементом цього рівня є генератор ознак `features_v8.py`, який реалізує алгоритми формування календарних, статистичних та подієвих

предикторів. Винесення генерації ознак в окремий компонент є принциповим проектним рішенням, оскільки гарантує ідентичність простору ознак на етапах offline-навчання та online-інференсу. До цього ж рівня належить модуль SBA, що інкапсулює логіку прогнозування переривчастого попиту та може використовуватися незалежно від інших моделей.

Контур offline-навчання моделей охоплює компоненти, відповідальні за побудову та оцінювання прогнозних моделей. До нього входять скрипти навчання моделей Prophet, XGBoost (пряма стратегія прогнозування) та базових моделей для переривчастого попиту. Ці компоненти зчитують історичні дані, використовують генератор ознак та формують навчені моделі разом із метриками якості, які зберігаються у вигляді артефактів. Offline-контур ізольований від користувацького інтерфейсу, що дозволяє виконувати обчислювально складні процедури навчання без впливу на швидкодію прикладного застосунку.

Контур онлайн-експлуатації реалізований у вигляді веб-застосунку на базі Streamlit. Даний компонент завантажує збережені модельні артефакти, виконує прогнозування за запитом користувача та взаємодіє з модулем візуалізації. Через інтерфейс користувач може обрати товарну позицію, горизонт прогнозування та рівень сервісу, після чого система формує прогноз, оцінює його точність у режимі backtest і надає рекомендації щодо управління запасами.

Таким чином, компонентна структура системи забезпечує чіткий поділ між зберіганням даних, обчислювальною логікою та рівнем представлення, що відповідає сучасним підходам до проєктування інформаційних систем аналітики та підтримки прийняття рішень в електронній комерції.

Потоки даних та сценарії виконання

Функціонування розробленої інформаційної системи ґрунтується на послідовному конвеєрі обробки даних, який поєднує процеси підготовки даних, прогнозування попиту та підтримки управлінських рішень. Динаміку

взаємодії між компонентами системи доцільно розглядати у вигляді потоків даних, що відображають порядок виконання основних обчислювальних етапів.

Архітектурно в системі виокремлюються два взаємопов'язані, але логічно незалежні контури: offline-контур підготовки моделей та online-контур експлуатації і прогнозування.

У межах офлайн-контуру виконується підготовка прогнозних моделей та оцінювання їх якості. Вхідними даними для цього етапу є історичні часові ряди продажів, представлені у вигляді щоденних агрегованих значень по кожній товарній позиції. На першому кроці ці дані передаються до модуля формування ознак, який перетворює вихідні часові ряди у матрицю предикторів із календарними, статистичними та подієвими характеристиками. Сформований простір ознак використовується для навчання моделей різних класів: адитивних регресійних моделей Prophet, моделей для переривчастого попиту SBA та двоступеневих моделей градієнтного бустингу Zero-Inflated XGBoost.

Результатом роботи offline-контуру є набір навчених моделей разом із метаданими, що включають перелік використаних ознак, статистики по товарних позиціях та показники точності. Для забезпечення відтворюваності експериментів і повторного використання моделей усі артефакти серіалізуються та зберігаються у централізованому сховищі.

Окремим етапом є порівняльний аналіз моделей, у межах якого на основі заданих критеріїв якості формується рейтинг та визначається оптимальний алгоритм для кожного SKU.

Online-контур призначений для експлуатації навчених моделей та реалізує сценарії прогнозування «за запитом». На цьому етапі система ініціалізується шляхом завантаження збережених модельних артефактів та історичних даних. Після задання користувацьких параметрів (товарна позиція, горизонт прогнозування, рівень сервісу) відповідні фрагменти даних передаються до модуля генерації ознак для формування вектору предикторів.

На етапі прогнозування реалізовано адаптивну логіку вибору алгоритму: система аналізує розрідженість історії продажів і автоматично спрямовує потік даних або до ML-ансамблю для регулярного попиту, або до методу SBA для переривчастого попиту. У випадку аналізу історичного періоду додатково активується режим backtest, у якому прогнозні значення зіставляються з фактичними даними, а точність оцінюється за допомогою метрик WAPE та RMSE.

Заключним етапом є робота підсистеми підтримки прийняття рішень. На основі сумарного прогнозу попиту та статистичних характеристик його варіації система обчислює величину страхового запасу відповідно до заданого рівня сервісу. Схему сценарію подано на рисунку 2.2.

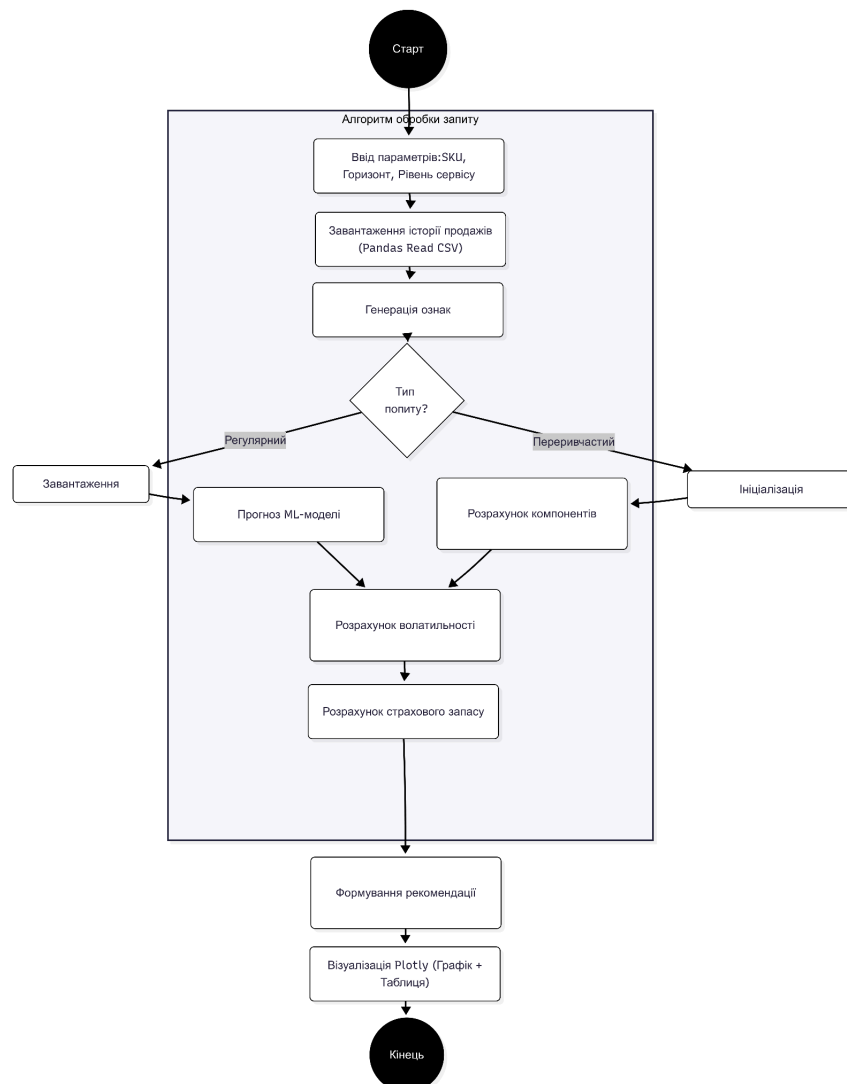


Рис 2.2. Схема сценарію виконання запиту в системі

У результаті користувач отримує не лише часовий прогноз продажів, але й кількісну рекомендацію щодо обсягу замовлення, що безпосередньо пов'язує прогнозування з прикладними бізнес-задачами. Таким чином, реалізована схема потоків даних забезпечує чітке розмежування між етапами підготовки моделей та їх експлуатації, інтегруючи прогнозні алгоритми в єдиний процес підтримки управлінських рішень.

Обґрунтування вибору засобів розробки та технологічна реалізація

Розробка інформаційної системи потребує використання інструментів, здатних ефективно працювати з великими масивами часових даних, реалізовувати сучасні алгоритми машинного навчання та забезпечувати інтерактивну взаємодію з користувачем. З огляду на це, для реалізації програмного комплексу було обрано екосистему Python, яка на сьогодні є де-факто стандартом у галузі аналітики даних і прикладного машинного навчання. Такий вибір зумовлений передусім наявністю розвиненої інфраструктури спеціалізованих бібліотек для глибокого аналізу часових рядів, а також повноцінною підтримкою інноваційних методів статистичного моделювання. Крім того, притаманні цій мові висока читабельність коду та значна швидкість прототипування виступають критично важливими факторами для ефективного виконання дослідницьких робіт і оперативної перевірки наукових гіпотез [23].

Для обробки та трансформації даних використовуються бібліотеки NumPy та Pandas, які забезпечують векторизовані обчислення, ефективну роботу з табличними структурами та підтримку часових індексів. Саме на їх основі реалізовано підготовку історичних рядів продажів, агрегацію даних та формування матриці ознак.

Для реалізації прогнозних моделей у роботі використано перевірені наукові та промислові бібліотеки, що широко застосовуються в задачах аналізу та прогнозування часових рядів. Зокрема, для моделювання сезонних і трендових компонент попиту використано бібліотеку Prophet. Її вибір

обґрунтований можливістю адитивної декомпозиції часового ряду, автоматичним виявленням точок зміни тренду, а також підтримкою календарних ефектів без жорстких вимог до стаціонарності вхідних даних, що є критично важливим для задач електронної комерції [18].

Для моделювання складних нелінійних залежностей між ознаками та обсягами продажів у роботі застосовано алгоритм XGBoost як основний інструмент машинного навчання. У межах проєкту реалізовано двоступеневу Zero-Inflated стратегію, яка поєднує задачу класифікації факту продажу з подальшим регресійним прогнозуванням його обсягу. Використання XGBoost забезпечує високу точність прогнозів, стійкість до шуму в даних, а також ефективну роботу з великим та гетерогенним простором ознак [19].

Для товарних позицій із переривчастим попитом у системі реалізовано метод Syntetos–Boylan Approximation у вигляді окремого програмного модуля. Застосування цього підходу є обґрунтованим, оскільки він є загальновизнаним у науковій літературі та практиці управління запасами для роботи з розрідженими часовими рядами [18]. Метод забезпечує незміщені оцінки середнього попиту за умов високої частки нульових спостережень і використовується як базовий інструмент прогнозування для відповідної групи товарів.

Результати offline-навчання моделей зберігаються у вигляді серіалізованих артефактів формату joblib, який оптимізований для роботи з великими об'єктами машинного навчання та широко застосовується у промислових ML-рішеннях [24].

Для побудови прикладного застосунку використано фреймворк Streamlit, що дозволяє оперативно створювати інтерактивні веб-інтерфейси для аналітичних систем без необхідності розробки окремого frontend-шару. Даний інструментарій забезпечує зручну інтеграцію з Python-моделями та дає змогу реалізувати сценарій прогнозування «за запитом» [23].

Візуалізація аналітичних даних та результатів моделювання здійснюється за допомогою бібліотеки Plotly, яка підтримує інтерактивні

часові графіки, функції масштабування та накладання декількох прогнозних моделей для їх порівняльного аналізу.

Обраний набір засобів розробки гарантує високу масштабованість програмного комплексу, можливість безперешкодного розширення системи новими алгоритмами, а також забезпечує повну відтворюваність експериментів. Важливою перевагою такого підходу є готовність системи до подальшої інтеграції з корпоративними інформаційними середовищами, зокрема системами управління ресурсами та складом. Таким чином, технологічна реалізація повністю відповідає вимогам до сучасних інформаційних технологій аналізу та прогнозування продажів в електронній комерції, створюючи надійну основу для практичного впровадження результатів дослідження.

Висновки до розділу 2

У другому розділі виконано математичне моделювання та проєктування інформаційної системи аналізу і прогнозування продажів в електронній комерції. Сформульовано постановку задачі прогнозування попиту як задачі аналізу часових рядів для гетерогенного асортименту та обґрунтовано критерії оптимізації якості прогнозу. Для оцінювання точності використано метрику WAPE як стійку до нульових значень і придатну для бізнес-інтерпретації, а також RMSE для аналізу локальних відхилень. Показано практичну інтеграцію прогнозу в задачу управління запасами через розрахунок страхового запасу з урахуванням рівня сервісу.

Розроблено комбінований підхід до прогнозування, що передбачає використання різних моделей залежно від статистичних властивостей попиту: Prophet для рядів з сезонністю, SBA для переривчастого попиту та двоступеневу Zero-Inflated модель на основі XGBoost для сценаріїв із нелінійною динамікою й високою розрідженістю даних. Запропоновано алгоритми формування простору ознак, які включають календарні, статистичні та подієві предиктори (циклічне кодування дат, лагові та ковзні

характеристики), що забезпечує узгодженість даних на етапах навчання та експлуатації.

Спроектовано модульну архітектуру програмного комплексу з розподілом на offline-контур навчання і online-контур інференсу та обґрунтовано вибір технологічного стеку. Отримані результати формують основу для практичної реалізації, демонстрації інтерфейсу та експериментального тестування системи у розділі 3.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ ПРОГНОЗУВАННЯ

3.1. Вимоги до технічного забезпечення та умови експлуатації

Коректне функціонування розробленого програмного комплексу, забезпечення відтворюваності експериментів та швидкодії обчислень безпосередньо залежать від дотримання вимог до апаратного та програмного середовища. Оскільки архітектура системи передбачає розподіл на серверну частину де виконуються обчислення та навчання моделей та клієнтську частину веб-інтерфейс, вимоги до них розглядаються окремо.

Процес навчання ансамблевих моделей машинного навчання зокрема градієнтного бустингу та декомпозиція часових рядів є ресурсомісткими операціями, що створюють значне навантаження на центральний процесор та оперативну пам'ят [25]. Для забезпечення прийняттого часу навчання моделей та швидкої генерації прогнозів у режимі реального часу, апаратне забезпечення сервера повинно відповідати характеристикам, наведеним у таблиці 3.1.

Таблиця 3.1

Вимоги до апаратного забезпечення сервера

Компонент	Мінімальні вимоги	Рекомендовані вимоги	Обґрунтування
Центральний процесор (CPU)	4 ядра, 2.0 GHz	6-8 ядер, 3.0 GHz+ (Intel Core i5/i7 або AMD Ryzen 5/7)	Забезпечення ефективної паралелізації обчислень при побудові дерев рішень у XGBoost.

Таблиця 3.1 Продовження таблиці

Оперативна пам'ять (RAM)	8 GB	16 GB і вище	Необхідна для завантаження повного історичного датасету та зберігання проміжних матриць ознак під час навчання.
Дискова підсистема	HDD 500 GB	SSD (NVMe) 256 GB+	Висока швидкість читання/запису (I/O) критична для операцій серіалізації моделей та читання великих CSV-файлів.

Завдяки реалізації системи у вигляді вебзастосунку, вимоги до робочого місця кінцевого користувача, аналітика або менеджера із закупівель, є мінімальними, оскільки робота із сервісом не потребує встановлення додаткового спеціалізованого програмного забезпечення. Для повноцінного функціонування необхідна лише наявність персонального комп'ютера, ноутбука або планшета зі стійким підключенням до локальної мережі чи мережі Інтернет. При цьому доступ до інтерфейсу здійснюється через сучасний веббраузер із підтримкою JavaScript, як-от Google Chrome, Mozilla Firefox, Microsoft Edge або Safari.

Система є повністю кросплатформною і може функціонувати під управлінням операційних систем Windows 10/11, Linux (Ubuntu 20.04+, CentOS) або macOS. Основним середовищем виконання на стороні сервера є інтерпретатор мови програмування Python версії 3.9 або вище. Для розгортання програмного комплексу необхідно встановити набір

залежностей, зафіксованих у файлі README.md, а детальний перелік ключових пакетів та їх версій представлений у таблиці 3.2.

Таблиця 3.2

Програмні залежності системи

Бібліотека	Версія	Функціональне призначення
streamlit	1.25.0+	Забезпечення роботи веб-сервера та рендеринг GUI.
pandas	2.0.0+	Маніпуляції з табличними даними, Time Series Indexing.
numpy	1.24.0+	Векторні обчислення, реалізація логіки SBA.
xgboost	1.7.5+	Реалізація регресорів та класифікаторів градієнтного бустингу.
prophet	1.1.4+	Моделювання сезонних компонент та трендів.
scikit-learn	1.3.0+	Попередня обробка даних (Label Encoding), розрахунок метрик.
joblib	1.3.0+	Збереження та завантаження навчених моделей.
plotly	5.15.0+	Візуалізація даних та побудова інтерактивних графіків.

Інформаційна система спроектована для роботи зі структурованими історичними даними продажів, завантаження яких здійснюється через інтерфейс користувача у форматі CSV з кодуванням UTF-8, що забезпечує універсальність та сумісність із більшістю сучасних систем обліку. Вхідний файл повинен містити суворо визначений набір обов'язкових атрибутів, а саме дату здійснення транзакції (Date), унікальний ідентифікатор товарної

позиції (SKU) та кількісний показник проданих одиниць (Quantity). Окрім базових полів, архітектура системи дозволяє опціонально обробляти додаткові ознаки, такі як цінові показники або маркетингові маркери, що сприяє глибшому аналізу впливу зовнішніх факторів на динаміку продажів. Такий підхід до вхідної інформації мінімізує ризики викривлення результатів моделювання та гарантує високу точність подальших прогнозних значень.

3.2. Реалізація алгоритмічного забезпечення та налаштування моделей

Логіку підготовки даних, генерації ознак та навчання моделей розділено на незалежні компоненти, що забезпечує гнучкість налаштування та можливість масштабування. Для прогнозування реалізовано стратегію ансамблювання спеціалізованих моделей.

Підготовка даних та конструювання ознак

Якість роботи алгоритмів машинного навчання критично залежить від інформативності вхідних даних та коректного представлення часових рядів у вигляді векторного простору ознак. У межах даного дослідження в модулі `features_v8.py` реалізовано розширений конвеєр генерації ознак, призначений для підготовки даних до навчання прогнозних моделей та врахування специфіки попиту в електронній комерції.

З метою врахування автокореляційної структури часових рядів у простір ознак включено лагові змінні, що відповідають значенням продажів у попередні моменти часу зі зсувами на один, два, три, сім, чотирнадцять, двадцять один, двадцять вісім та тридцять днів. Використання таких лагів дозволяє моделям захоплювати як короткострокові залежності, так і тижневі та місячні цикли, характерні для поведінки споживачів в онлайн-ритейлі.

Для опису локального тренду та рівня волатильності попиту застосовано статистичні ознаки, розраховані на основі ковзних вікон. Зокрема, для інтервалів тривалістю 7 та 28 днів обчислюються середні

значення та стандартні відхилення продажів. Такі характеристики дозволяють згладити випадкові флуктуації даних і надати моделі інформацію про поточний стан динаміки попиту.

З урахуванням вираженої сезонності продажів у електронній комерції до простору ознак включено календарні змінні. Для коректного моделювання циклічних закономірностей використано тригонометричні перетворення дня року у вигляді синусоїдальних та косинусоїдальних компонент. Додатково введено бінарні індикатори, що відображають належність дати до вихідних днів та початку календарного місяця, які часто супроводжуються зміною споживчої активності.

Окрему групу складають маркери подій, релевантні для електронної комерції. До них належать ознаки, що характеризують відстань у днях до різдвяного періоду, наявність періоду масових розпродажів наприкінці листопада, а також державні святкові дні. Включення таких змінних дозволяє моделям враховувати вплив зовнішніх факторів, які призводять до різких і короткотривалих змін попиту.

Для коректної роботи з переривчастими часовими рядами сформовано спеціалізовані ознаки, що описують ступінь розрідженості попиту. Зокрема, враховується кількість днів, що минули з моменту останнього продажу, а також частка днів із ненульовими продажами за останній місяць. Такі характеристики є важливими для відокремлення товарів зі стабільною динамікою від позицій із нерегулярним попитом та підвищують ефективність прогнозування в умовах високої частки нульових спостережень.

Налаштування двоетапної моделі XGBoost

Для товарів основного асортименту реалізовано підхід Zero-Inflated XGBoost, який ефективно вирішує проблему надмірної кількості нулів у даних шляхом декомпозиції загальної задачі на два послідовні етапи. На першому етапі застосовується алгоритм класифікації, що визначає ймовірність наявності продажу як такого, а на другому – регресійна модель,

яка прогнозує логарифм кількості товару за умови підтвердження факту продажу. Таке поєднання методів дозволяє значно підвищити точність моделювання для товарних позицій з нерегулярним попитом, оскільки система спочатку оцінює шанс виникнення транзакції, а вже потім обчислює її потенційний обсяг. Фінальний прогноз формується як добуток ймовірності та відновленого обсягу (формула 3.1):

$$\hat{y} = P_{clf} \cdot \exp(\hat{y}_{reg}), \quad (3.1)$$

де $\hat{y}$ – фінальний прогноз обсягу попиту;

P_{clf} – прогнозована імовірність настання продажу, отримана за допомогою класифікаційної моделі;

$\hat{y}_{reg}$ – прогнозоване значення логарифмованого обсягу продажу, отримане регресійною моделлю за умови наявності продажу;

$\exp(\cdot)$ – експоненціальна функція, що використовується для повернення прогнозу з логарифмічної шкали до початкового простору значень.

Такий підхід дозволяє окремо моделювати ймовірність виникнення транзакції та очікуваний обсяг продажу, що є особливо ефективним для товарів з нерегулярним попитом.

Параметри моделей підібрано емпіричним шляхом для забезпечення стабільності.

```
@dataclass
class TrainParams:
    clf_n_estimators: int = 1200
    clf_lr: float = 0.05
    clf_max_depth: int = 6
    clf_subsample: float = 0.85
    clf_colsample: float = 0.85
    clf_min_child_weight: float = 3.0
    clf_reg_lambda: float = 1.0
    clf_gamma: float = 0.0
    reg_n_estimators: int = 1500
    reg_lr: float = 0.05
    reg_max_depth: int = 6
    reg_subsample: float = 0.85
    reg_colsample: float = 0.85
    reg_min_child_weight: float = 3.0
    reg_reg_lambda: float = 1.0
    reg_gamma: float = 0.0
    early_stopping_rounds: int = 80
    n_jobs: int = -1
    random_state: int = 42
```

```
tree_method: str = "hist"
```

Ініціалізація моделей з урахуванням підібраних гіперпараметрів реалізована наступним чином.

```
clf = xgb.XGBClassifier(  
    n_estimators=params.clf_n_estimators,  
    learning_rate=params.clf_lr,  
    max_depth=params.clf_max_depth,  
    subsample=params.clf_subsample,  
    colsample_bytree=params.clf_colsample,  
    min_child_weight=params.clf_min_child_weight,  
    reg_lambda=params.clf_reg_lambda,  
    gamma=params.clf_gamma,  
    n_jobs=params.n_jobs,  
    random_state=params.random_state,  
    tree_method=params.tree_method,  
    objective="binary:logistic",  
    eval_metric="logloss",  
    scale_pos_weight=scale_pos_weight,  
)  
  
reg = xgb.XGBRegressor(  
    n_estimators=params.reg_n_estimators,  
    learning_rate=params.reg_lr,  
    max_depth=params.reg_max_depth,  
    subsample=params.reg_subsample,  
    colsample_bytree=params.reg_colsample,  
    min_child_weight=params.reg_min_child_weight,  
    reg_lambda=params.reg_reg_lambda,  
    gamma=params.reg_gamma,  
    n_jobs=params.n_jobs,  
    random_state=params.random_state,  
    tree_method=params.tree_method,  
    objective="reg:squarederror",  
    eval_metric="rmse",  
)
```

Для компенсації дисбалансу класів у класифікаторі застосовано динамічне зважування (`scale_pos_weight`), а для регресора використано зважування прикладів $\text{sample_weight} = 1 + \log(1 + y)$, що фокусує модель на значущих продажах.

Адаптивна конфігурація моделі Prophet

Використання бібліотеки Prophet у розробленій системі спрямоване на моделювання товарних позицій з вираженою сезонністю та трендовими компонентами попиту. Разом з тим, застосування однакових стандартних гіперпараметрів для всього асортименту є неефективним, оскільки часові

ряди продажів у електронній комерції суттєво відрізняються за щільністю спостережень і рівнем переривчастості [26].

З метою підвищення точності прогнозування у роботі реалізовано алгоритм адаптивного налаштування гіперпараметрів. Його ключова ідея полягає в автоматичному аналізі статистичних властивостей часових рядів та виборі режиму конфігурації моделі залежно від характеру попиту конкретного товару.

Як критерій адаптації використовується показник щільності попиту, який визначається як частка днів із ненульовими продажами у межах історичного періоду спостережень, що дозволяє системі автоматично обирати найбільш ефективний режим роботи моделі. Якщо частка днів із продажами становить менше 40%, активується режим переривчастого попиту (Intermittent), у якому модель налаштовується консервативно зі зниженим параметром гнучкості тренду `changepoint_prior_scale = 0.03`. Таке обмеження дозволяє запобігти перенаванчання на випадкових одиничних сплесках, характерних для товарів із рідкісними транзакціями. Натомість для товарів із регулярним попитом, де кількість ненульових значень перевищує 40%, застосовується режим щільного попиту (Dense). У цьому випадку гнучкість моделі підвищується до `changepoint_prior_scale = 0.06`, а сезонна компонента моделюється у мультиплікативному режимі, що забезпечує швидку адаптацію прогнозу до змін ринкової кон'юнктури та реального масштабу коливань попиту.

Фрагмент програмної реалізації алгоритму адаптивної ініціалізації моделі Prophet.

```
def pick_params(train_df: pd.DataFrame, pp: ProphetParams) -> Tuple[str, float, float, float]:
    y = train_df["y"].values.astype(float)
    nonzero_share = float(np.mean(y > 0))
    if nonzero_share >= pp.nonzero_dense_threshold:
        seasonality_mode = "multiplicative"
        cps, sps = pp.cps_dense, pp.sps_dense
    elif nonzero_share < pp.nonzero_intermittent_threshold:
        seasonality_mode = "additive"
        cps, sps = pp.cps_intermittent, pp.sps_intermittent
    else:
        seasonality_mode = "additive"
```

```

        cps, sps = 0.05, 8.0
        return seasonality_mode, float(cps), float(sps), nonzero_share
    -----
m_final = Prophet(
    yearly_seasonality="auto",
    weekly_seasonality="auto",
    daily_seasonality=False,
    seasonality_mode=seasonality_mode,
    changepoint_prior_scale=cps,
    seasonality_prior_scale=sps,
)

```

Окрему увагу в реалізації приділено врахуванню зовнішніх календарних факторів. Додавання регресора `is_weekend` та календаря державних свят дозволяє моделі коректно розрізняти регулярні зниження попиту у вихідні дні від реальних змін тренду. Це зменшує систематичні похибки прогнозування та підвищує стабільність моделі в умовах сезонних коливань, характерних для електронної комерції [27].

Запропонований підхід до адаптивної конфігурації Prophet дозволяє використовувати модель не як універсальний інструмент «за замовчуванням», а як гнучкий компонент гібридної системи прогнозування, що автоматично підлаштовується під властивості конкретного товарного ряду.

Програмна реалізація методу SBA

Для прогнозування попиту на товари з високим рівнем переривчастості, де стандартні методи машинного навчання демонструють низьку ефективність через розрідженість даних, реалізовано спеціалізований модуль `sba.py`. Оскільки готова реалізація методу SBA відсутня в популярних бібліотеках, алгоритм було розроблено програмно з використанням бібліотеки NumPy для забезпечення високої швидкодії [28]. Програмна логіка базується на класі даних `SBAState`, який інкапсулює стан моделі. Він інкапсулює поточні згладжені значення розміру замовлення (z) та інтервалу (p), а також індивідуальні коефіцієнти згладжування (α , β) для кожного товару. Це дозволяє зберігати розраховані компоненти попиту та використовувати їх для побудови прогнозу без необхідності повторного перерахунку історичного ряду.

Ключовою відмінністю методу SBA від класичного методу Кростона є введення коригуючого множника, який усуває зміщення прогнозу. Реалізація розрахункової формули наведена нижче [29].

```
def forecast_sba(state: SBASState, horizon: int) -> np.ndarray:
    horizon = int(horizon)
    if horizon <= 0:
        return np.array([], dtype=float)

    if not state.fitted or state.z == 0:
        return np.zeros(horizon, dtype=float)

    correction = 1.0 - (state.beta / 2.0)

    p_safe = max(state.p, 0.001)

    f = correction * (state.z / p_safe)
    f = max(0.0, f)

    return np.full(horizon, f, dtype=float)
```

Використання конструкції `np.full` обумовлено природою методу SBA – це стаціонарний прогноз, який показує "середню інтенсивність" попиту на майбутній період, що є достатнім для розрахунку страхових запасів рідкісних товарів [28].

Для підвищення точності моделювання відмовлено від використання фіксованих коефіцієнтів згладжування наприклад, стандартних $\alpha=0.1$, $\beta=0.1$). Натомість реалізовано функцію, яка виконує перебір комбінацій параметрів на валідаційній вибірці, наведено нижче.

```
def grid_search_sba(
    y_train: np.ndarray,
    y_val: np.ndarray,
    alphas=(0.05, 0.1, 0.2, 0.3),
    betas=(0.05, 0.1, 0.2),
    metric="wape_sum",
) -> tuple[SBASState, dict]:
    y_train = np.asarray(y_train, dtype=float)
    y_val = np.asarray(y_val, dtype=float)

    best = None
    best_score = float("inf")
    best_meta = {}
    denom = float(np.sum(np.abs(y_val)))
    denom = denom if denom > 1e-9 else 1.0

    for a in alphas:
        for b in betas:
            st = fit_sba(y_train, alpha=float(a), beta=float(b))
            fc = forecast_sba(st, horizon=len(y_val))
```

```

        if metric == "mae":
            score = float(np.mean(np.abs(y_val - fc)))
        else:
            score = float(np.abs(np.sum(y_val) - np.sum(fc)) / denom)

        if score < best_score:
            best_score = score
            best = st
            best_meta = {"alpha": float(a), "beta": float(b), "score":
float(score)}

    return best, best_meta

```

Такий підхід дозволяє системі автоматично адаптуватися до динаміки конкретного товару: для більш стабільних рядів обираються менші коефіцієнти згладжування, а для рядів, що швидко змінюються – більші.

3.3 Опис інтерфейсу користувача та сценарій роботи системи

Для забезпечення практичного використання розробленої інформаційної системи прогнозування продажів реалізовано інтерактивний користувацький інтерфейс у вигляді веб-застосунку. Інтерфейс призначений для фахівців з планування запасів, аналітиків або менеджерів електронної комерції та забезпечує доступ до основних функцій системи без необхідності взаємодії з програмним кодом.

Користувацький інтерфейс реалізовано з використанням фреймворку Streamlit, що дозволяє швидко створювати інтерактивні аналітичні панелі на базі Python [28]. Візуалізація результатів прогнозування здійснюється за допомогою бібліотеки Plotly, яка підтримує інтерактивні графіки та масштабування часових рядів.

Структурна організація та налаштування параметрів

Робочий простір системи (рис. 3.1) розділено на дві функціональні зони: бічну панель конфігурації та основну область аналітики.



Рис 3.1. Головне вікно застосунку

Така організація інтерфейсу відповідає принципам побудови аналітичних систем і дозволяє користувачу послідовно переходити від завантаження даних до аналізу результатів.

При запуску застосунку система у фоновому режимі виконує зчитування історичних даних з CSV файлів. Виконується попередня обробка та кешування даних для прискорення подальшої роботи. Користувач одразу отримує доступ до готового інтерфейсу без необхідності ручного імпорту файлів.

У бічній панелі інтерфейсу, зображеній на рис. 3.2, зосереджено основні елементи керування вхідними параметрами прогнозування, що забезпечують гнучке налаштування сценарію аналізу та зручну взаємодію користувача із системою. Центральним елементом панелі є селектор товарної позиції (SKU), у якому реалізовано механізм інтелектуального пошуку. Завдяки цьому користувач може оперативно знаходити необхідний товар як за кодом, так і за його назвою, що є критично важливим за умов великого асортименту електронної комерції.

Після вибору товарної позиції система автоматично відображає інформаційний блок з основними метаданими, зокрема назвою товару, його належністю до відповідного класу ABC та поточною ціною. Виведення цих

даних надає менеджеру необхідний аналітичний контекст та дозволяє приймати рішення з урахуванням комерційної значущості обраного товару.

Окрему увагу приділено керуванню часовим діапазоном прогнозування. Окрім можливості ручного вибору початкової та кінцевої дат, у системі реалізовано спеціалізовані пресети для типових сценаріїв використання. Режим «Наступні 30 днів» орієнтований на задачі оперативного планування майбутніх поставок і дозволяє швидко отримати прогноз на стандартному горизонті планування. Водночас режим «Останні 30 днів (Backtest)» забезпечує виконання ретроспективної перевірки точності алгоритмів шляхом порівняння згенерованих прогнозних значень з відомими історичними даними, що є важливим елементом валідації моделей.

Для підвищення аналітичної гнучкості користувачу надано можливість вибору ансамблю моделей, результати яких відображаються на графіку. Це дозволяє порівнювати прогнози, отримані за допомогою різних алгоритмів, аналізувати їх відмінності та обирати найбільш доцільний підхід залежно від характеру попиту та поставлених управлінських задач.

The screenshot shows a 'Forecast Configuration' panel with the following sections:

- Налаштування прогнозу**
 - Вибір товару:
 - SKU: 15036 — ASSORTED COLOURS SILK FAN
 - Інформація про товар
 - Назва: ASSORTED COLOURS SILK FAN
 - Клас: A
 - Ціна: 0.83 £
- Період прогнозу**
 - Швидкий вибір:
 - ☐ Без пресету
 - ☐ Наступні 30 днів
 - ☒ Останні 30 днів (Backtest)
 - Діапазон: 2025/11/16 – 2025/12/15
- Моделі на графіку**
 - ☒ XGBoost
 - ☒ Prophet
 - ☒ SBA

Рис 3.2. Панель налаштування параметрів прогнозування

Після визначення всіх вхідних умов взаємодія користувача переходить до етапу генерації даних. Для цього в інтерфейсі реалізовано кнопку «Розрахувати прогноз». Після натискання кнопки система відображає індикатор прогресу, виконує інференс обраних математичних моделей і заповнює основну робочу область аналітичними даними, опис яких наведено у наступному підрозділі.

Візуалізація результатів моделювання та сценарний аналіз

Після ініціалізації розрахунку в основній робочій області програмного застосунку формується візуальне представлення результатів прогнозування. Для побудови графіків використано бібліотеку Plotly, яка забезпечує інтерактивність, високу деталізацію та зручність аналітичного аналізу результатів (рис. 3.3).

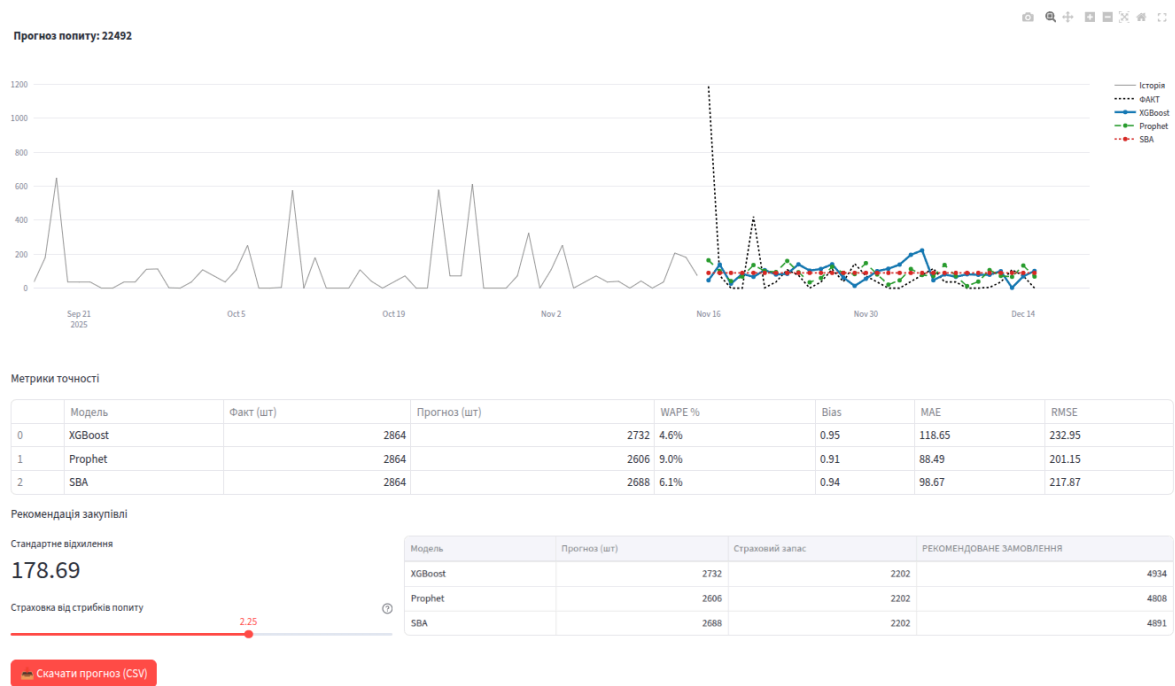


Рис 3.3. Інтерактивна візуалізація динаміки попиту та результатів прогнозування

Графічний інтерфейс реалізує концепцію накладання інформаційних шарів, за якої на одній часовій осі поєднуються історичні дані, прогнозні значення та, у режимі тестування, фактичні спостереження. Такий підхід

дозволяє користувачу одночасно оцінити попередню динаміку продажів і результати роботи різних моделей прогнозування.

На графіку відображається історичний контекст у вигляді фактичних продажів за фіксований період до моменту прогнозу, що забезпечує візуальне розуміння характеру попиту наявність трендів, сезонних коливань або поодиноких сплесків. Прогнозні траєкторії, отримані за допомогою різних моделей, візуалізуються окремими лініями з різними стилями відображення, що дає змогу легко їх порівнювати. У режимі “backtesting” додатково відображаються фактичні значення продажів на прогнозному інтервалі, що дозволяє оперативно оцінити точність і стабільність прогнозів.

Важливою особливістю інтерфейсу є підтримка режиму “unified hover” (рис. 3.4), при наведенні курсору на будь-яку дату часової шкали відображаються числові значення прогнозів усіх активних моделей для цього моменту часу.

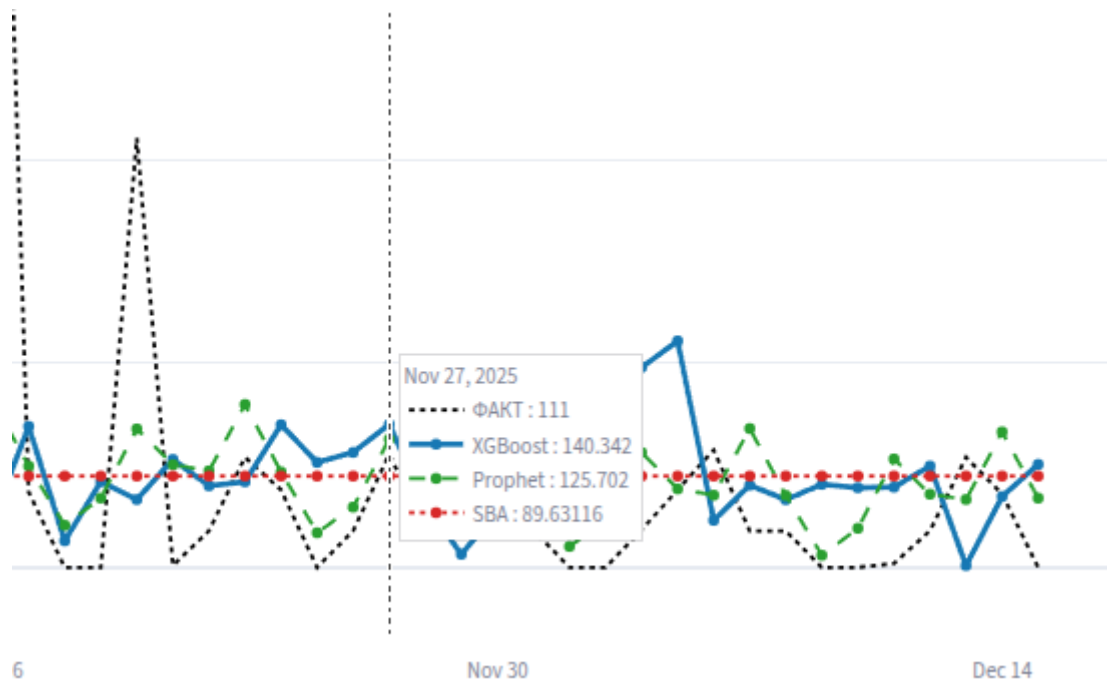


Рис 3.4. Режим unified hover

Це дозволяє аналітику виконувати сценарний аналіз і порівнювати альтернативні прогнозні підходи без необхідності звернення до табличних звітів. Крім того, реалізовано функції масштабування та панорамування

графіка, що є критично важливим для аналізу періодів пікових навантажень, зокрема під час сезонних розпродажів або маркетингових кампаній наприклад, Black Friday, коли значення має деталізація до окремого дня.

Таким чином, візуалізаційний модуль системи виконує не лише демонстраційну, але й аналітичну функцію, забезпечуючи підтримку сценарного аналізу та підвищуючи обґрунтованість управлінських рішень у сфері планування запасів.

Система підтримки прийняття рішень та управління ризиками

Важливим етапом роботи інформаційної системи є перевірка адекватності прогнозних моделей та забезпечення можливості подальшого використання отриманих результатів у процесах планування. З цією метою у програмному комплексі реалізовано механізми валідації моделей та експорту результатів прогнозування.

Валідація моделей здійснюється у режимі тестування, за якого частина історичних даних навмисно виключається з навчальної вибірки та використовується як тестовий інтервал. Для кожного товару система формує прогноз на обраному горизонті, після чого порівнює його з фактичними значеннями продажів. Дані відображаються у вигляді таблиці метрик (рис. 3.5). Такий підхід дозволяє оцінити якість моделей в умовах, максимально наближених до реальної експлуатації.

Метрики точності

	Модель	Факт (шт)	Прогноз (шт)	WAPE %	Bias	MAE	RMSE
0	XGBoost	2864	2732	4.6%	0.95	118.65	232.95
1	Prophet	2864	2606	9.0%	0.91	88.49	201.15
2	SBA	2864	2688	6.1%	0.94	98.67	217.87

Рис 3.5. Таблиця результатів

Одним із центральних функціональних блоків розробленого програмного комплексу є підсистема підтримки прийняття рішень щодо управління запасами, орієнтована на зниження ризиків дефіциту та надлишку товарів. Даний блок інтегрує результати прогнозування попиту з класичними

методами управління запасами та надає користувачеві інтерпретовані рекомендації у зручній табличній формі (рис. 3.6).

Ключовим елементом підсистеми є розрахунок страхового запасу, який дозволяє компенсувати невизначеність прогнозу та випадкові коливання попиту. Для цього система використовує оцінку волатильності попиту, представлену у вигляді стандартного відхилення, що відображається у лівій частині інтерфейсу. Дане значення розраховується на основі історичних даних або похибок прогнозної моделі та є кількісною характеристикою ризику.

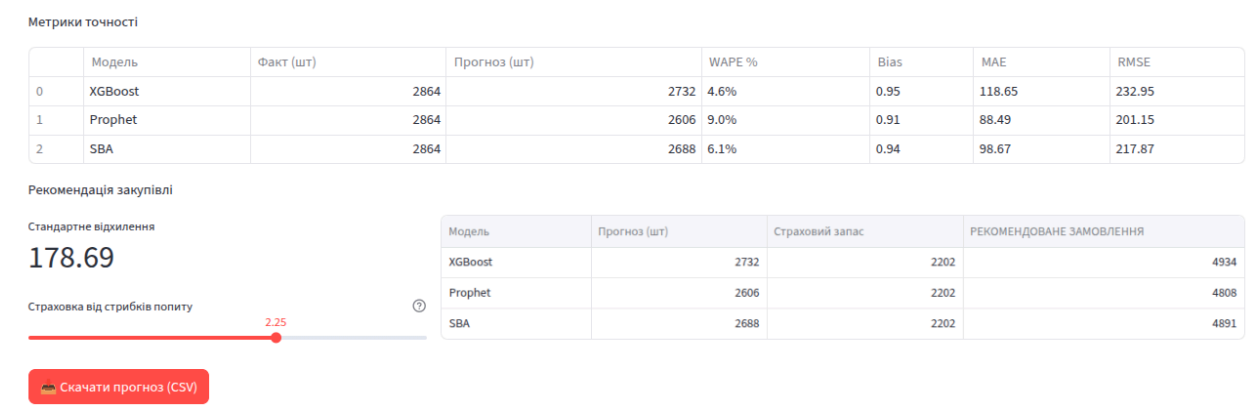


Рис 3.6. Блок рекомендацій закупівлі та управління ризиками

Користувачеві надається можливість інтерактивно керувати рівнем захисту від стрибків попиту за допомогою повзунка. Збільшення коефіцієнту призводить до зростання страхового запасу та, відповідно, до зменшення ймовірності дефіциту, але одночасно підвищує ризик надлишкових запасів. Таким чином, система дозволяє реалізувати сценарний аналіз та знайти компроміс між витратами на зберігання та рівнем обслуговування клієнтів.

Для практичного використання результатів у бізнес-процесах система підтримує експорт (рис. 3.7) прогнозних даних та розрахункових показників у табличному форматі CSV. До експортованого набору входять прогнозні значення попиту на кожен день горизонту планування, сумарний прогноз, оцінка страхового запасу та рекомендований обсяг замовлення.

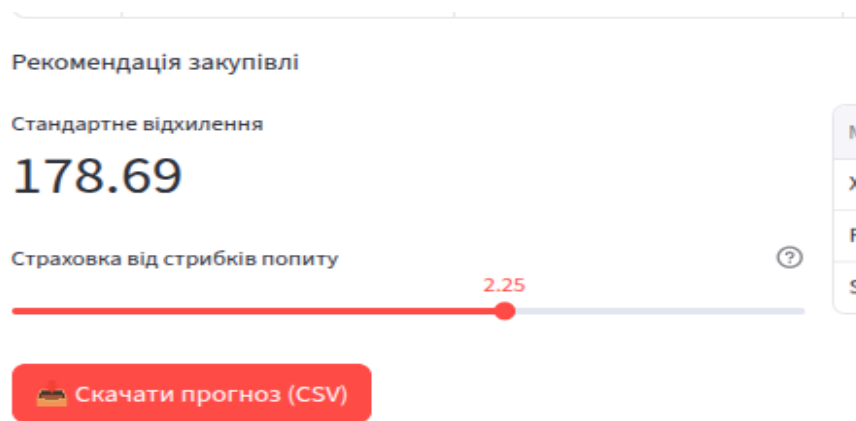


Рис 3.7. Кнопка завантаження результатів

Такий формат забезпечує сумісність з електронними таблицями, ERP-системами та іншими аналітичними інструментами.

Модуль розвідувального аналізу даних (EDA)

Для забезпечення комплексності дослідження, окрім прогнозного блоку, у програмному комплексі реалізовано модуль розвідувального аналізу даних (Exploratory Data Analysis, EDA). Його призначення полягає у попередньому аудиті вхідних даних та формуванні уявлення про ретроспективну структуру продажів перед прийняттям управлінських рішень [31]. Інтерфейс модуля (рис. 3.8) складається з чотирьох функціональних аналітичних блоків.

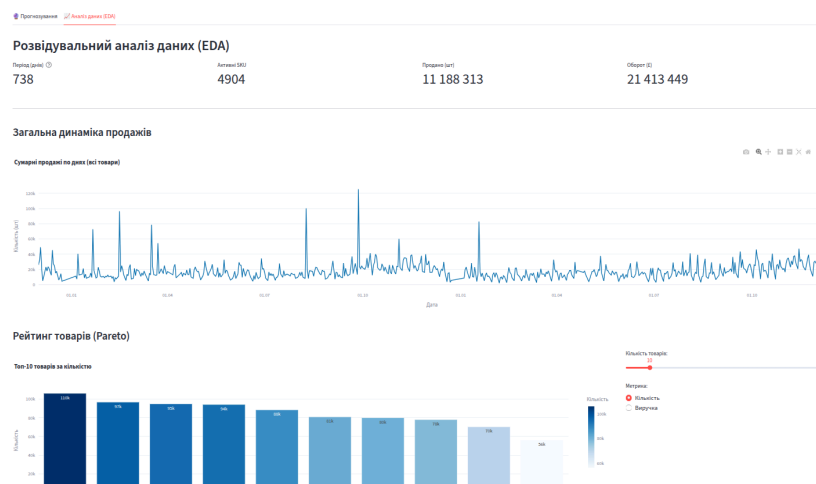


Рис 3.8. Інтерфейс модуля розвідувального аналізу даних

У верхній частині робочої області відображаються зведені метрики, що характеризують масштаб бізнесу за обраний історичний період (рис. 3.9). Система динамічно обчислює: кількість активних товарних позицій (SKU), тривалість аналізованого періоду у днях, сумарний обсяг продажів у натуральному вираженні, загальний оборот у грошовому еквіваленті.

Розвідувальний аналіз даних (EDA)

Період (днів) ⓘ	Активні SKU	Продано (шт)	Оборот (€)
738	4904	11 188 313	21 413 449

Рис 3.9. Панель агрегованих показників

Наведені показники дозволяють оперативно оцінити репрезентативність вибірки даних та масштаб подальшого прогнозування.

Графік «Загальна динаміка продажів» відображає сумарний попит по всьому асортименту у денному розрізі (рис. 3.10).

Сумарні продажі по днях (всі товари)

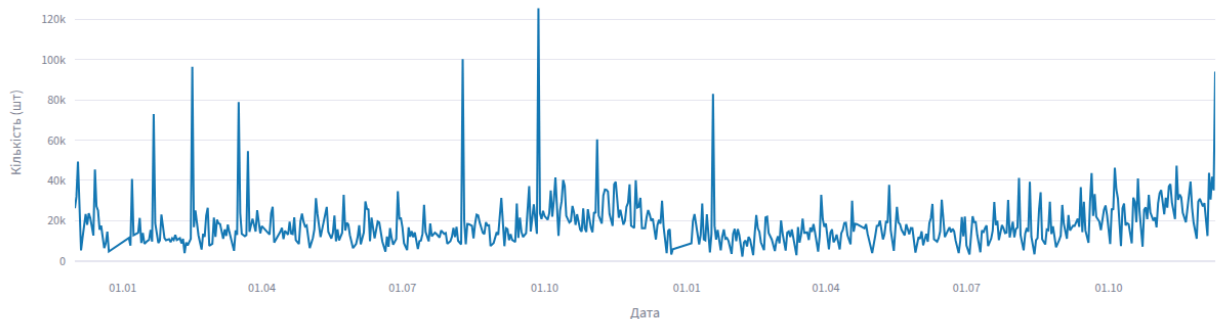


Рис 3.10. Графік динаміки продажів

Така візуалізація дає змогу ідентифікувати глобальні тренди, сезонні коливання та аномальні значення, зокрема пікові продажі напередодні свят або маркетингових кампаній. Виявлені особливості враховуються при налаштуванні параметрів прогнозних моделей.

Для визначення пріоритетних товарних позицій у системі реалізовано інтерактивну рейтингову діаграму на основі принципу Парето (рис. 3.11). Функціонал інструменту дозволяє користувачеві самостійно налаштовувати

глибину рейтингу, обираючи кількість товарів (top-N) у діапазоні від 5 до 50, а також визначати ключовий критерій ранжування – за кількістю проданих одиниць або за величиною отриманого обороту. Така гнучкість у виборі метрики забезпечує можливість проведення багатовимірної сегментації асортименту, оскільки товари, що лідирують за фізичним обсягом продажів, часто не збігаються з позиціями, що приносять найбільший прибуток, наприклад, через наявність дорогих маржинальних товарів.

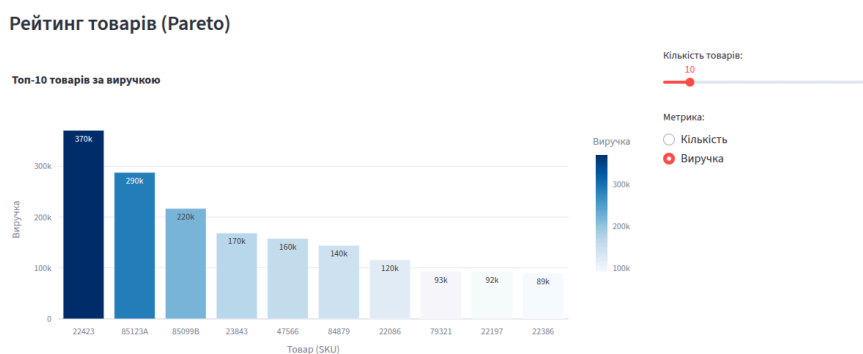


Рис 3.11. Рейтинговий аналіз за принципом Парето

Такий підхід дозволяє виділити товари категорії А, які формують основну частку продажів або доходу та мають ключове значення для управління запасами.

Для виявлення прихованих часових патернів використовується теплова карта, побудована шляхом подвійної агрегації даних за днями тижня та місяцями року (рис. 3.12).

Патерни попиту (Heatmap)



Рис 3.12. Рейтинговий аналіз за принципом Парето

Візуалізація дозволяє визначити періоди пікового попиту та повторювані поведінкові шаблони споживачів. Колірна шкала інтуїтивно підсвічує зони високої та низької інтенсивності продажів, що є корисним при формуванні календарних ознак у моделях прогнозування.

Таким чином, модуль EDA перетворює первинні транзакційні дані на аналітичні знання про поведінку ринку та структуру попиту. Його використання підвищує прозорість прогнозних рішень і є необхідною передумовою для побудови надійних моделей прогнозування продажів в електронній комерції [31].

3.4. Результати тестування системи та оцінка точності прогнозів

Заключним етапом розробки програмного комплексу стало проведення експериментального дослідження ефективності реалізованих моделей. Метою тестування було не лише підтвердження працездатності системи, а й визначення оптимального алгоритму прогнозування для подальшого використання у модулі підтримки прийняття рішень.

Валідація моделей здійснювалася методом ретроспективного тестування на репрезентативній вибірці з 20 найбільш популярних товарів, які формують основну частку товарообігу досліджуваного підприємства. Процедура передбачала вилучення даних за останні 30 днів з навчальної вибірки та генерацію прогнозів для цього періоду за п'ятьма різними алгоритмами (рис. 3.13).

```
(.venv) ~/PycharmProjects/e-commerce-forecast git:[main]
python -m src.models.compare_and_select

Generating reports...
Table saved: reports/models_pivot_comparison.csv
```

	Mean WAPE	Median WAPE	Std Dev	Mean MAE	Mean RMSE	Bias Dev
Model						
Seasonal Naive	116.89%	101.87%	0.454	87.492917	179.003209	NaN
XGBoost	74.31%	38.54%	1.288	NaN	NaN	0.26002
Prophet	60.09%	42.25%	0.449	81.823348	174.447632	NaN
SBA	69.14%	51.97%	0.588	90.769831	163.245843	NaN
Moving Avg (4w)	142.47%	96.65%	1.099	87.843690	162.073476	NaN

Рис 3.13. Виконання валідації моделей

Якість прогнозування оцінювалася за допомогою метрик WAPE та Bias. Ключовою особливістю аналізу стало використання медіанного значення WAPE (Median WAPE) як основного критерію ранжування. На відміну від середнього арифметичного, медіана є стійкою до аномальних викидів (outliers), які неминуче виникають при прогнозуванні товарів з нерегулярним попитом. Це дозволяє об'єктивно оцінити «типову» точність моделі у щоденній експлуатації, ігноруючи поодинокі критичні помилки, спричинені зовнішніми форс-мажорними факторами [32].

У рамках дослідження було проведено порівняння ефективності трьох класів алгоритмів: базових евристик (Moving Average, Seasonal Naive), спеціалізованого статистичного методу SBA та алгоритмів машинного навчання (Prophet, XGBoost). Зведені результати розрахунків наведено у таблиці 3.3.

Таблиця 3.3

Показники моделей прогнозування

Модель	Median WAPE (%)	Mean WAPE (%)	Bias Dev	Оцінка ефективності
XGBoost	38.54%	74.31%	0.26	Найвища точність
Prophet	42.25%	60.09%	—	Висока точність
SBA	51.97%	69.14%	—	Середня точність
Moving Avg (4w)	96.65%	142.47%	—	Низька точність
Seasonal Naive	101.87%	116.89%	—	Незадовільна

Аналіз отриманих даних яскраво демонструє перевагу методів машинного навчання над традиційними підходами. Безумовним лідером експерименту стала модель XGBoost, яка забезпечила найменшу медіанну помилку прогнозування на рівні 38.54%. Це у 2.6 рази ефективніше, ніж використання класичного методу ковзного середнього, де помилка склала 96.65%. Висока точність XGBoost пояснюється здатністю градієнтного бустингу враховувати складні нелінійні залежності та використовувати згенеровані лагові ознаки для адаптації до динаміки попиту. Показник зміщення ($\text{Bias Dev} = 0.26$) свідчить про помірну схильність моделі до відхилення від факту, що є допустимим в умовах високої волатильності.

Друге місце посіла модель Facebook Prophet із медіанною похибкою 42.25%, що лише на 3.7 відсоткових пункти поступається лідеру. Варто зазначити цікаву особливість: за показником середньої помилки (Mean WAPE) Prophet випередив XGBoost (60.09% проти 74.31%). Це свідчить про те, що Prophet є більш «консервативним» алгоритмом: він рідше припускається критично великих помилок на аномальних даних, хоча в типових сценаріях дещо програє бустингу в точності.

Окремої уваги заслуговує результат методу SBA (Syntetos-Boylan Approximation), який показав медіанну помилку 51.97%. Хоча цей результат поступається ML-моделям, він є суттєво кращим за наївні методи. SBA вдалося знизити помилку майже вдвічі порівняно з сезонним наївним методом (101.87%). Це підтверджує гіпотезу про те, що для товарів з нерегулярним попитом використання спеціалізованих статистичних наближень є значно ефективнішим, ніж просте усереднення. У розробленій системі SBA виконує роль надійного «страхового» алгоритму (robust baseline) для випадків, коли історичних даних недостатньо для повноцінного навчання ML-моделей.

Натомість базові методи – Moving Average та Seasonal Naive – продемонстрували незадовільні результати з похибкою, що наближається до 100%. Це доводить їхню непридатність для управління запасами в умовах

сучасного e-commerce, де попит характеризується високою стохастичністю та переривчастістю.

Висновки до розділу 3

У третьому розділі магістерської роботи було детально розглянуто програмну реалізацію розробленої інформаційної системи прогнозування продажів для електронної комерції та проведено її експериментальне тестування. Описано архітектуру програмного продукту, ключові алгоритмічні рішення та практичні аспекти експлуатації системи.

У ході роботи визначено вимоги до технічного забезпечення та умов використання системи, що підтвердило можливість її розгортання на стандартних обчислювальних ресурсах без потреби у спеціалізованому обладнанні. Реалізація алгоритмічного забезпечення базується на модульному підході, що дозволяє незалежно розвивати компоненти прогнозування, оптимізації запасів та візуалізації результатів. Особливу увагу приділено адаптивному налаштуванню моделей прогнозування, зокрема автоматичному підбору параметрів Prophet залежно від щільності попиту та використанню методу SBA для товарів із переривчастими продажами. Це дозволило підвищити стабільність прогнозів і зменшити ризик перенавчання моделей на нерепрезентативних даних.

Розроблений користувацький інтерфейс на базі Streamlit забезпечує зручну взаємодію з системою та підтримує як прогнозування, так і розвідувальний аналіз даних (EDA). Візуалізація результатів у вигляді інтерактивних графіків, рейтингових діаграм та теплових карт підвищує прозорість роботи алгоритмів і спрощує інтерпретацію результатів для менеджерів.

Експериментальне тестування на реальних історичних даних підтвердило ефективність запропонованого підходу. Порівняльний аналіз показав, що моделі машинного навчання та сезонні моделі забезпечують істотно кращу точність прогнозування порівняно з базовими евристичними

методами, тоді як спеціалізовані алгоритми для переривчастого попиту залишаються доцільними для окремих груп товарів. Отримані результати свідчать про практичну придатність системи для підтримки прийняття рішень у сфері управління запасами.

Таким чином, у третьому розділі доведено, що розроблений програмний комплекс є працездатним і може бути використаний як інструмент аналітичної підтримки процесів планування продажів та управління ризиками в електронній комерції.

ВИСНОВКИ

У магістерській роботі вирішено актуальне науково-практичне завдання підвищення ефективності управління товарними запасами підприємств електронної комерції шляхом розробки та впровадження спеціалізованої інформаційної системи прогнозування продажів. Проведений аналіз предметної області засвідчив, що сучасний онлайн-рітейл характеризується високим рівнем невизначеності, широким асортиментом та наявністю значної частки товарів з нерегулярним попитом, що робить класичні методи екстраполяції неефективними. Обґрунтовано, що для забезпечення точності планування необхідним є перехід від простих евристик до використання адаптивних моделей машинного навчання.

У ході дослідження було розроблено гібридну методологію прогнозування, яка базується на сегментації товарного асортименту та диференційованому застосуванні математичних алгоритмів. Для товарів зі стабільним попитом та вираженою сезонністю реалізовано використання ансамблевих методів, зокрема градієнтного бустингу (XGBoost) та адитивних регресійних моделей (Prophet), тоді як для позицій з переривчастим попитом застосовано метод апроксимації Сінтетоса-Бойлана (SBA). Такий підхід дозволив врахувати стохастичну природу продажів та мінімізувати вплив аномальних викидів на якість прогнозу.

Практичним результатом роботи стала програмна реалізація інформаційної системи на базі мови програмування Python та фреймворку Streamlit, яка автоматизує повний цикл роботи з даними: від попередньої обробки та генерації ознак до розрахунку страхових запасів та візуалізації результатів.

Експериментальна перевірка розробленого програмного комплексу на реальних даних підтвердила його працездатність та ефективність. Результати тестування засвідчили перевагу алгоритмів машинного навчання над базовими методами: модель Zero-Inflated XGBoost продемонструвала найвищу точність із медіанною похибкою WAPE на рівні 38,54%, що у 2,6

рази перевищує показники методу ковзного середнього. Впровадження розробленої системи дозволяє суттєво знизити ризики виникнення дефіциту товарів (Out-of-Stock) та уникнути надмірного заморожування обігових коштів у неліквідних запасах, що створює підґрунтя для прийняття обґрунтованих управлінських рішень та підвищення рентабельності бізнесу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Mykola Turchyn, Olena Chyzhmotria The importance of forecasting in e-commerce. Комп'ютерні технології: інновації, проблеми, рішення: тези VII Всеукраїнської науково-технічної конференції, 02-03 грудня 2024 року. Житомир: "Житомирська політехніка", 2024. С. 7-8. URL: <https://conf.ztu.edu.ua/wp-content/uploads/2025/01/7.pdf>.
2. Mykola Turchyn, Olena Chyzhmotria, Iryna Dmytrenko. Seaborn as a tool for effective work with categorical data. Комп'ютерні технології: інновації, проблеми, рішення: Тези VIII Всеукраїнської науково-технічної конференції, 02-03 грудня 2025 року. Житомир: "Житомирська політехніка", 2025. (подано до друку)
3. Mykola Turchyn, Olena Chyzhmotria, Iryna Dmytrenko. Seaborn as a tool for effective work with numerical data. Комп'ютерні технології: інновації, проблеми, рішення: Тези VIII Всеукраїнської науково-технічної конференції, 02-03 грудня 2025 року. Житомир: "Житомирська політехніка", 2025. (подано до друку)
4. Садчикова І., Тарасенко А., Дубина М. Теоретичне обґрунтування сутності поняття «електронна комерція». Економіка та суспільство. 2023. № 53. DOI: 10.32782/2524-0072/2023-53-36.
5. Рубаненко І., Шепель І. Електронна комерція в ЄС та Україні: тренди, виклики та перспективи. Трансформаційна економіка. 2025. № 10. С. 84–89. DOI: 10.32782/2786-8141/2025-10-13.
6. Simintiras A. C., Reynolds N., Johnston J. The impact of change in the sales process on sales performance: A multi-level analysis. 2015. DOI: 10.1007/978-3-319-10864-3_100.
7. Kim S.-Y., Ko J.-M., Shin J.-G., Yang S. Intelligent power sales forecasting technology for company strategy and operation planning. Journal of Electrical Engineering & Technology. 2025. Vol. 20. P. 1–13. DOI: 10.1007/s42835-025-02390-4.

8. Іващенко Г., Тимошенко Д., Близнюк О., Кононенко О. Моделі глибокого навчання для прогнозування часових рядів. Системи управління, навігації та зв'язку. Збірник наукових праць. 2024. № 1. С. 82–87. DOI: 10.26906/SUNZ.2024.1.082.
9. Кучанський О. Ю., Ніколенко В. В. Прогнозування часових рядів методом зіставлення зі зразком. *Управління розвитком складних систем*. 2015. № 22. С. 101–106. URL: <https://repository.knuba.edu.ua/server/api/core/bitstreams/1ac9171f-a7db-4360-ac04-6ddcd685098c/content>.
10. Андрусенко Ю. О. Аналіз основних моделей прогнозування часових рядів. Збірник наукових праць Харківського національного університету Повітряних Сил. 2020. № 65. С. 91–96. DOI: 10.30748/zhups.2020.65.14.
11. Su Y.-F., Yang C. A structural equation model for analyzing the impact of ERP on SCM. *Expert Systems with Applications*. 2010. Vol. 37. P. 456–469. DOI: 10.1016/j.eswa.2009.05.061.
12. Yang C., Su Y.-F. The relationship between benefits of ERP systems implementation and its impacts on firm performance of SCM. *Journal of Enterprise Information Management*. 2009. Vol. 22. P. 722–752. DOI: 10.1108/17410390910999602.
13. Benlian A., Koufaris M., Hess T. The role of SaaS service quality for continued SaaS use: Empirical insights from SaaS using firms. *Proceedings of the Thirty First International Conference on Information Systems (ICIS 2010)*. 2010. Paper 26. URL: https://www.researchgate.net/publication/221599026_The_role_of_SAAS_service_quality_for_continued_SAAS_use_Empirical_insights_from_SAAS_using_firms.
14. Onibokun T., Ejibenam A., Ekeocha P., Onayemi H., Halliday N. The use of AI to improve CX in SaaS environment. *International Journal of*

- Multidisciplinary Research and Growth Evaluation. 2022. Vol. 3. P. 672–679. DOI: 10.54660/IJMRGE.2022.3.3.672-679.
15. My L. N. T., Nguyen V., Vo T. A rich-spatial and multiscaled transformer-based approach for long-term multivariate time-series forecasting problem. *Journal of Forecasting*. 2025. DOI: 10.1002/for.70066.
 16. Makridakis S., Spiliotis E., Assimakopoulos V. Statistical and machine learning forecasting methods: Concerns and ways forward. *PLoS ONE*. 2018. Vol. 13. DOI: 10.1371/journal.pone.0194889.
 17. Мулеса О., Снитюк В. Розробка еволюційного методу для прогнозування часових рядів. *Automation of Technological and Business Processes*. 2020. Vol. 12. С. 4–9. DOI: 10.15673/atbp.v12i3.1854.
 18. Ning Y., Kazemi H., Tahmasebi P. A comparative machine learning study for time series oil production forecasting: ARIMA, LSTM, and Prophet. *Computers & Geosciences*. 2022. Vol. 164. Art. 105126. DOI: 10.1016/j.cageo.2022.105126.
 19. Tunggadewa K., Widodo E. Forecasting consumable part for aircraft maintenance using time series method. *Jurnal Locus Penelitian dan Pengabdian*. 2025. Vol. 4. DOI: 10.58344/locus.v4i6.4076.
 20. Tian X., Wang H., Erjiang E. Forecasting intermittent demand for inventory management by retailers: A new approach. *Journal of Retailing and Consumer Services*. 2021. Vol. 62. Art. 102662. DOI: 10.1016/j.jretconser.2021.102662.
 21. Білобородова Т. Дослідження методів аналізу нелінійної динаміки часових рядів. *Вісник Східноукраїнського національного університету імені Володимира Даля*. 2022. № 1 (271). С. 5–10. DOI: 10.33216/1998-7927-2022-271-1-5-10.
 22. Yin H., Wen D., Li J., Wei Z., Zhang X., Huang Z., Li F. Optimal matrix sketching over sliding windows. *Proceedings of the VLDB Endowment*. 2024. Vol. 17. P. 2149–2161. DOI: 10.14778/3665844.3665847.

23. Bonam V. S. M., Vangoor V. K. R., Shaik M., Sadhu A. K. R. From data to decisions: Leveraging AI for accurate sales forecasting in CRM. *Journal of Computational Analysis and Applications*. 2024. Vol. 33. P. 1949–1967. URL: https://www.researchgate.net/publication/389582346_From_Data_to_Decisions_Leveraging_AI_for_Accurate_Sales_Forecasting_in_CRM.
24. Limbu S., Rumita R., Khadka R., Chi Y. Forecasting global monthly cotton prices: The superiority of NNAR models over traditional models. *Frontiers in Artificial Intelligence*. 2025. Vol. 8. Art. 1628744. DOI: 10.3389/frai.2025.1628744.
25. Піцун О., Рудик В., Батько Ю. Класифікація вебсайтів за критерієм їх продуктивності на підставі ансамблів. *Scientific Bulletin of UNFU*. 2025. Vol. 35. С. 180–185. DOI: 10.36930/40350421.
26. Kuang Y. Time series data forecasting: Alcohol sales prediction based on Prophet. *Highlights in Science, Engineering and Technology*. 2024. Vol. 85. P. 597–601. DOI: 10.54097/x446dk79.
27. Tsoumakas G. A survey of machine learning techniques for food sales prediction. *Artificial Intelligence Review*. 2019. Vol. 52. DOI: 10.1007/s10462-018-9637-z.
28. Sezer M., Cebi F. Comparative analysis of probabilistic models for intermittent demand forecasting. 2024. DOI: 10.1007/978-3-031-70935-7_12.
29. Bai Z.-H., Chu P.-Y. TSB-HB: A hierarchical Bayesian extension of the TSB model for intermittent demand forecasting. 2025. DOI: 10.48550/arXiv.2511.12749.
30. Parker A., Heflin A., Jones L. Analyzing University of Virginia Health publications using open data, Python, and Streamlit. *Journal of the Medical Library Association*. 2021. Vol. 109. DOI: 10.5195/jmla.2021.1360.
31. Alshammari A. Implementation of exploratory data analysis (EDA) in Python. *International Journal of Computer Applications*. 2025. Vol. 187. P. 34–42. DOI: 10.5120/ijca2025925577.

32. Alzara M., Gihad N., Abdou H., Soltan A., Hilal A., Ehab A. Macroeconomic-aware forecasting of construction costs in developing countries: Using gated recurrent unit and long short-term memory deep learning framework. *PLOS ONE*. 2025. Vol. 20. Art. e0333189. DOI: 10.1371/journal.pone.0333189.

ДОДАТКИ

Алгоритми генерації ознак

```

# src/features_v8.py
from __future__ import annotations

import numpy as np
import pandas as pd

try:
    import holidays
    _UK_HOLIDAYS = holidays.country_holidays("GB")
except Exception:
    _UK_HOLIDAYS = None

def _is_holiday_uk(d: pd.Series) -> pd.Series:
    if _UK_HOLIDAYS is None:
        return pd.Series(np.zeros(len(d), dtype=int), index=d.index)
    dd = pd.to_datetime(d).dt.date
    return dd.apply(lambda x: 1 if x in _UK_HOLIDAYS else 0).astype(int)

def _days_to_christmas(ts: pd.Series) -> pd.Series:
    dt = pd.to_datetime(ts)
    year = dt.dt.year
    xmas_this = pd.to_datetime(year.astype(str) + "-12-25")
    xmas_next = pd.to_datetime((year + 1).astype(str) + "-12-25")
    target = np.where(dt <= xmas_this, xmas_this, xmas_next)
    target = pd.to_datetime(target)
    return (target - dt).dt.days.astype(int)

def _calendar_features(date_series: pd.Series, prefix: str = "") -> pd.DataFrame:
    d = pd.to_datetime(date_series)
    out = pd.DataFrame(index=d.index)

    out[f"{prefix}day_of_week"] = d.dt.dayofweek.astype(int)
    out[f"{prefix}month"] = d.dt.month.astype(int)
    out[f"{prefix}day_of_year"] = d.dt.dayofyear.astype(int)
    out[f"{prefix}is_weekend"] = (d.dt.dayofweek >= 5).astype(int)
    out[f"{prefix}is_month_start"] = d.dt.is_month_start.astype(int)
    out[f"{prefix}is_month_end"] = d.dt.is_month_end.astype(int)

    doy = out[f"{prefix}day_of_year"].astype(float)
    out[f"{prefix}sin_doy"] = np.sin(2 * np.pi * doy / 365.25)
    out[f"{prefix}cos_doy"] = np.cos(2 * np.pi * doy / 365.25)

    out[f"{prefix}is_holiday"] = _is_holiday_uk(d)

    dtc = _days_to_christmas(d)
    out[f"{prefix}days_to_christmas"] = dtc
    out[f"{prefix}is_pre_christmas"] = ((dtc >= 1) & (dtc <= 30)).astype(int)

    out[f"{prefix}is_black_friday_period"] = ((d.dt.month == 11) & (d.dt.day >= 20) &
        (d.dt.day <= 30)).astype(int)

    return out

def add_target_calendar_features_df(df: pd.DataFrame, horizon: int) -> pd.DataFrame:
    out = df.copy()
    out["Date"] = pd.to_datetime(out["Date"])

```

```

target_date = out["Date"] + pd.to_timedelta(int(horizon), unit="D")

cal = _calendar_features(target_date, prefix="target_")

rename = {
    "target_day_of_week": "target_day_of_week",
    "target_month": "target_month",
    "target_is_weekend": "target_is_weekend",
    "target_day_of_year": "target_day_of_year",
    "target_sin_doy": "target_sin_doy",
    "target_cos_doy": "target_cos_doy",
    "target_is_month_start": "target_is_month_start",
    "target_is_month_end": "target_is_month_end",
    "target_days_to_christmas": "target_days_to_christmas",
    "target_is_pre_christmas": "target_is_pre_christmas",
    "target_is_black_friday_period": "target_is_black_friday_period",
    "target_is_holiday": "target_is_holiday",
}
cal = cal.rename(columns=rename)

for c in cal.columns:
    out[c] = cal[c].values

return out

def add_target_calendar_features_row(base_row: pd.DataFrame, anchor_date, horizon:
    int) -> pd.DataFrame:
    out = base_row.copy()
    out["Date"] = pd.to_datetime(anchor_date)
    out = add_target_calendar_features_df(out, horizon=int(horizon))
    return out

def add_features_v8(
    df: pd.DataFrame,
    sku_mean_dict: dict,
    sku_std_dict: dict,
    sku_cv_dict: dict,
    sku_means_dict: dict,
    spike_window: int = 30,
    spike_sigma: float = 2.5,
) -> pd.DataFrame:
    x = df.copy()
    x["SKU"] = x["SKU"].astype(str)
    x["Date"] = pd.to_datetime(x["Date"])
    x["Quantity"] = pd.to_numeric(x["Quantity"],
        errors="coerce").fillna(0.0).clip(lower=0)

    x = x.sort_values(["SKU", "Date"]).reset_index(drop=True)

    cal = _calendar_features(x["Date"], prefix="")
    for c in cal.columns:
        x[c] = cal[c].values

    x["sku_mean_global"] = x["SKU"].map(sku_mean_dict).fillna(0.0).astype(float)
    x["sku_std_global"] = x["SKU"].map(sku_std_dict).fillna(0.0).astype(float)
    x["sku_cv_global"] = x["SKU"].map(sku_cv_dict).fillna(0.0).astype(float)

    x["sku_mean_alt"] =
        x["SKU"].map(sku_means_dict).fillna(x["sku_mean_global"]).astype(float)

    g = x.groupby("SKU", sort=False)

```

```

lag_list = [1, 2, 3, 7, 14, 21, 28, 30]
for l in lag_list:
    x[f"lag_{l}"] = g["Quantity"].shift(l)

q1 = g["Quantity"].shift(1)
x["rolling_mean_7"] = q1.groupby(x["SKU"]).rolling(7,
    min_periods=3).mean().reset_index(level=0, drop=True)
x["rolling_std_7"] = q1.groupby(x["SKU"]).rolling(7,
    min_periods=3).std().reset_index(level=0, drop=True)

x["rolling_mean_28"] = q1.groupby(x["SKU"]).rolling(28,
    min_periods=7).mean().reset_index(level=0, drop=True)
x["rolling_std_28"] = q1.groupby(x["SKU"]).rolling(28,
    min_periods=7).std().reset_index(level=0, drop=True)

nonzero = (x["Quantity"] > 0).astype(int)
nonzero_shift = g[nonzero.name].shift(1) if nonzero.name in x.columns else
    g[nonzero].shift(1)
x["_nonzero"] = (x["Quantity"] > 0).astype(int)
x["nonzero_share_30"] = g["_nonzero"].shift(1).rolling(30,
    min_periods=10).mean().reset_index(level=0, drop=True)

dsls = np.zeros(len(x), dtype=float)
for sku, idx in g.indices.items():
    ii = np.asarray(idx, dtype=int)
    qty = x.loc[ii, "Quantity"].to_numpy()
    dates = x.loc[ii, "Date"].to_numpy()

    last_sale_date = None
    for k in range(len(ii)):
        if last_sale_date is None:
            dsls[ii[k]] = np.nan
        else:
            dsls[ii[k]] = (pd.Timestamp(dates[k]) -
                pd.Timestamp(last_sale_date)).days

        if qty[k] > 0:
            last_sale_date = dates[k]

x["days_since_last_sale"] = dsls

x["spike_flag"] = 0
mu = x["rolling_mean_28"]
sd = x["rolling_std_28"].fillna(0.0)
z = (x["Quantity"] - mu) / (sd + 1e-6)
x.loc[z > spike_sigma, "spike_flag"] = 1

x.drop(columns=["_nonzero"], inplace=True, errors="ignore")

return x

```

Реалізація методу SBA

```

# src/models/intermittent/sba.py
from __future__ import annotations

from dataclasses import dataclass
import numpy as np

@dataclass
class SBASState:
    alpha: float
    beta: float
    z: float
    p: float
    q: int
    fitted: bool = True

def _clip01(x: float) -> float:
    return float(np.clip(x, 0.001, 1.0))

def fit_sba(y: np.ndarray, alpha: float = 0.1, beta: float = 0.1) -> SBASState:

    y = np.asarray(y, dtype=float)
    y = np.nan_to_num(y, nan=0.0)

    alpha = _clip01(alpha)
    beta = _clip01(beta)

    if y.size == 0:
        return SBASState(alpha, beta, 0.0, 1.0, 0, fitted=False)

    pos_indices = np.where(y > 0)[0]

    if len(pos_indices) == 0:
        return SBASState(alpha, beta, 0.0, 1.0, len(y), fitted=True)

    if len(pos_indices) == 1:
        first_idx = pos_indices[0]
        z = float(y[first_idx])
        p = float(first_idx + 1)
        q = len(y) - first_idx - 1
        return SBASState(alpha, beta, z, p, q, fitted=True)

    first_idx = pos_indices[0]
    z = float(y[first_idx])
    p = float(first_idx + 1)
    q = 0

    for i in range(first_idx + 1, len(y)):
        val = y[i]
        q += 1

        if val > 0:
            z = z + alpha * (val - z)
            p = p + beta * (q - p)
            q = 0

    return SBASState(alpha, beta, z, p, q, fitted=True)

def forecast_sba(state: SBASState, horizon: int) -> np.ndarray:

```

```

horizon = int(horizon)
if horizon <= 0:
    return np.array([], dtype=float)

if not state.fitted or state.z == 0:
    return np.zeros(horizon, dtype=float)

correction = 1.0 - (state.beta / 2.0)

p_safe = max(state.p, 0.001)

f = correction * (state.z / p_safe)
f = max(0.0, f)

return np.full(horizon, f, dtype=float)

def fit_forecast_sba(y: np.ndarray, horizon: int, alpha: float = 0.1, beta: float =
    0.1) -> np.ndarray:
    state = fit_sba(y, alpha, beta)
    return forecast_sba(state, horizon)

def grid_search_sba(
    y_train: np.ndarray,
    y_val: np.ndarray,
    alphas=(0.05, 0.1, 0.2, 0.3),
    betas=(0.05, 0.1, 0.2),
    metric="wape_sum",
) -> tuple[SBAState, dict]:
    y_train = np.asarray(y_train, dtype=float)
    y_val = np.asarray(y_val, dtype=float)

    best = None
    best_score = float("inf")
    best_meta = {}

    denom = float(np.sum(np.abs(y_val)))
    denom = denom if denom > 1e-9 else 1.0

    for a in alphas:
        for b in betas:
            st = fit_sba(y_train, alpha=float(a), beta=float(b))
            fc = forecast_sba(st, horizon=len(y_val))

            if metric == "mae":
                score = float(np.mean(np.abs(y_val - fc)))
            else:
                score = float(np.abs(np.sum(y_val) - np.sum(fc)) / denom)

            if score < best_score:
                best_score = score
                best = st
                best_meta = {"alpha": float(a), "beta": float(b), "score":
                    float(score)}

    return best, best_meta

```

Налаштування та ініціалізація моделі XGBoost

```

# src/models/train_xgb_direct.py
from __future__ import annotations

import sys
from pathlib import Path

FILE = Path(__file__).resolve()
PROJECT_ROOT = FILE.parents[2]
if str(PROJECT_ROOT) not in sys.path:
    sys.path.insert(0, str(PROJECT_ROOT))

import os
import time
import warnings
import joblib
import numpy as np
import pandas as pd
import xgboost as xgb
from dataclasses import dataclass
from typing import Dict, Any, List, Tuple
from tqdm import tqdm
from src.features_v8 import add_features_v8, add_target_calendar_features_df

warnings.filterwarnings("ignore")
HISTORY_PATH = "data/history_for_app.csv"
OUT_MODEL_PATH = "data/xgb_models_direct.joblib"
OUT_METRICS_PATH = "data/xgb_direct_metrics.csv"

TEST_DAYS = 120
TOP_N_SKUS = 20
HORIZONS = list(range(1, 31))

MIN_TRAIN_POINTS_PER_SKU = 200
MIN_SAMPLES_PER_HORIZON = 200
CONTEXT_TAIL_DAYS = 365 * 2
EPS = 1e-6

def waape_sum30(actual_sum: np.ndarray, pred_sum: np.ndarray) -> float:
    actual_sum = np.asarray(actual_sum, dtype=float)
    pred_sum = np.asarray(pred_sum, dtype=float)
    denom = float(np.sum(np.abs(actual_sum)))
    if denom <= 0:
        return 0.0
    return float(np.sum(np.abs(actual_sum - pred_sum)) / (denom + EPS))

def preprocess_history(df_long: pd.DataFrame) -> pd.DataFrame:
    df = df_long.copy()
    df["Date"] = pd.to_datetime(df["Date"])
    df["SKU"] = df["SKU"].astype(str)
    df["Quantity"] = pd.to_numeric(df["Quantity"], errors="coerce").fillna(0.0)
    df["Quantity"] = df["Quantity"].clip(lower=0)
    df = df.sort_values(["SKU", "Date"]).reset_index(drop=True)
    return df

def select_top_skus(df_long: pd.DataFrame, top_n: int) -> List[str]:
    totals = df_long.groupby("SKU")["Quantity"].sum().sort_values(ascending=False)

```

```

    return totals.head(top_n).index.tolist()

def build_global_stats(
    train_long: pd.DataFrame,
) -> Tuple[Dict[str, float], Dict[str, float], Dict[str, float], Dict[str, float]]:
    sku_means = train_long.groupby("SKU")["Quantity"].mean().to_dict()
    gs = train_long.groupby("SKU")["Quantity"].agg(["mean", "std"])
    sku_mean_dict = gs["mean"].to_dict()
    sku_std_dict = gs["std"].fillna(0.0).to_dict()
    sku_cv_dict = (gs["std"] / (gs["mean"] + EPS)).fillna(0.0).to_dict()
    return sku_means, sku_mean_dict, sku_std_dict, sku_cv_dict

def infer_feature_columns(df_feat: pd.DataFrame) -> Tuple[List[str], List[str]]:
    exclude_cols = {"Date", "SKU", "Quantity"}
    base_feature_cols = [
        c
        for c in df_feat.columns
        if c not in exclude_cols
        and not c.startswith("target_")
        and np.issubdtype(df_feat[c].dtype, np.number)
    ]

    target_feature_cols = [
        "target_day_of_week",
        "target_month",
        "target_is_weekend",
        "target_day_of_year",
        "target_sin_doy",
        "target_cos_doy",
        "target_is_month_start",
        "target_is_month_end",
        "target_days_to_christmas",
        "target_is_pre_christmas",
        "target_is_black_friday_period",
        "target_is_holiday",
    ]

    feature_cols = base_feature_cols + target_feature_cols
    return base_feature_cols, feature_cols

def safe_get_row(df: pd.DataFrame, date: pd.Timestamp) -> pd.DataFrame | None:
    tmp = df[df["Date"] == date]
    if tmp.empty:
        return None
    return tmp.iloc[[-1]].copy()

def fit_with_early_stopping(
    model,
    X_train,
    y_train,
    X_val,
    y_val,
    early_stopping_rounds: int,
    verbose: bool = False,
    sample_weight=None,
):
    base_kwargs = {"eval_set": [(X_val, y_val)], "verbose": verbose}
    if sample_weight is not None:
        base_kwargs["sample_weight"] = sample_weight

```

```

try:
    model.fit(
        X_train,
        y_train,
        early_stopping_rounds=int(early_stopping_rounds),
        **base_kwargs,
    )
    return model
except TypeError:
    pass

try:
    callbacks = [
        xgb.callback.EarlyStopping(
            rounds=int(early_stopping_rounds),
            save_best=True,
            maximize=False,
        )
    ]
    model.fit(X_train, y_train, callbacks=callbacks, **base_kwargs)
    return model
except TypeError:
    pass

try:
    model.fit(X_train, y_train, **base_kwargs)
    return model
except TypeError:
    fallback_kwargs = {"verbose": verbose}
    if sample_weight is not None:
        try:
            model.fit(X_train, y_train, sample_weight=sample_weight,
                **fallback_kwargs)
            return model
        except TypeError:
            pass
    model.fit(X_train, y_train, **fallback_kwargs)
    return model

```

@dataclass

```

class TrainParams:
    clf_n_estimators: int = 1200
    clf_lr: float = 0.05
    clf_max_depth: int = 6
    clf_subsample: float = 0.85
    clf_colsample: float = 0.85
    clf_min_child_weight: float = 3.0
    clf_reg_lambda: float = 1.0
    clf_gamma: float = 0.0

    # regressor
    reg_n_estimators: int = 1500
    reg_lr: float = 0.05
    reg_max_depth: int = 6
    reg_subsample: float = 0.85
    reg_colsample: float = 0.85
    reg_min_child_weight: float = 3.0
    reg_reg_lambda: float = 1.0
    reg_gamma: float = 0.0

    early_stopping_rounds: int = 80

    n_jobs: int = -1

```

```

random_state: int = 42
tree_method: str = "hist"

def main():
    os.makedirs("data", exist_ok=True)

    print(f"Loading history: {HISTORY_PATH}")
    if not os.path.exists(HISTORY_PATH):
        raise FileNotFoundError(f"History not found: {HISTORY_PATH}")

    df_long = pd.read_csv(HISTORY_PATH, parse_dates=["Date"], low_memory=False)
    df_long = preprocess_history(df_long)

    cutoff_date = df_long["Date"].max() - pd.Timedelta(days=TEST_DAYS)

    top_skus = select_top_skus(df_long, TOP_N_SKUS)
    df_long = df_long[df_long["SKU"].isin(top_skus)].copy()

    print(f"Evaluating SKUs: {len(top_skus)} (top {TOP_N_SKUS} by total units)")
    print(f"Global date range: {df_long['Date'].min().date()} .. {df_long['Date'].max().date()}")
    print(f"Test window: last {TEST_DAYS} days, cutoff={cutoff_date.date()}")

    train_long = df_long[df_long["Date"] < cutoff_date].copy()
    sku_means, sku_mean_dict, sku_std_dict, sku_cv_dict =
        build_global_stats(train_long)

    print("Building features (v8) ...")
    t0 = time.time()
    df_feat = add_features_v8(
        df_long,
        sku_mean_dict=sku_mean_dict,
        sku_std_dict=sku_std_dict,
        sku_cv_dict=sku_cv_dict,
        sku_means_dict=sku_means,
    ).copy()
    df_feat = df_feat.dropna().reset_index(drop=True)
    print(f"Features built: rows={len(df_feat):,} in {time.time() - t0:.1f}s")

    base_feature_cols, feature_cols = infer_feature_columns(df_feat)
    params = TrainParams()

    all_models_pack: Dict[str, Dict[str, Dict[int, Any]]] = {}
    sku_bias_30: Dict[str, float] = {}
    results: List[Dict[str, Any]] = []

    unique_skus = df_feat["SKU"].unique().tolist()

    print(f"Training XGBoost Direct ZI models for {len(unique_skus)} SKU ...")
    print(f"Horizons: {HORIZONS[0]}..{HORIZONS[-1]}")

    sku_pbar = tqdm(unique_skus, desc="SKU", unit="sku", dynamic_ncols=True)

    for sku in sku_pbar:
        sku_pbar.set_postfix_str(f"sku={sku}")

        sku_base = df_feat[df_feat["SKU"] == sku].copy()
        if len(sku_base) < MIN_TRAIN_POINTS_PER_SKU:
            continue

        last_date = sku_base["Date"].max()
        sku_base = sku_base[sku_base["Date"] >= (last_date -
            pd.Timedelta(days=CONTEXT_TAIL_DAYS))].copy()

```

```

models_clf: Dict[int, Any] = {}
models_reg: Dict[int, Any] = {}

h_pbar = tqdm(HORIZONS, desc=f"H({sku})", unit="h", leave=False,
               dynamic_ncols=True)

for h in h_pbar:
    h = int(h)

    sku_h = add_target_calendar_features_df(sku_base, horizon=h)
    y_qty = sku_h["Quantity"].shift(-h)
    sku_h = sku_h.assign(_y_qty=y_qty)

    train_full = sku_h.dropna(subset=["_y_qty"]).copy()
    if len(train_full) < MIN_SAMPLES_PER_HORIZON:
        continue

    train_mask = train_full["Date"] < cutoff_date
    val_mask = ~train_mask

    if int(train_mask.sum()) < 80 or int(val_mask.sum()) < 20:
        continue

    X = train_full[feature_cols]
    y_bin = (train_full["_y_qty"].values > 0).astype(int)

    X_train, X_val = X[train_mask], X[val_mask]
    y_train_bin, y_val_bin = y_bin[train_mask], y_bin[val_mask]

    pos = max(int(y_train_bin.sum()), 1)
    neg = max(int((y_train_bin == 0).sum()), 1)
    scale_pos_weight = float(neg / pos)

    clf = xgb.XGBClassifier(
        n_estimators=params.clf_n_estimators,
        learning_rate=params.clf_lr,
        max_depth=params.clf_max_depth,
        subsample=params.clf_subsample,
        colsample_bytree=params.clf_colsample,
        min_child_weight=params.clf_min_child_weight,
        reg_lambda=params.clf_reg_lambda,
        gamma=params.clf_gamma,
        n_jobs=params.n_jobs,
        random_state=params.random_state,
        tree_method=params.tree_method,
        objective="binary:logistic",
        eval_metric="logloss",
        scale_pos_weight=scale_pos_weight,
    )

    fit_with_early_stopping(
        clf,
        X_train,
        y_train_bin,
        X_val,
        y_val_bin,
        early_stopping_rounds=params.early_stopping_rounds,
        verbose=False,
    )
    models_clf[h] = clf

    y_qty_arr = train_full["_y_qty"].values
    is_pos = y_qty_arr > 0

```

```

if int(is_pos.sum()) < 80:
    continue

y_reg = np.loglp(y_qty_arr[is_pos])
X_pos = X.loc[is_pos].copy()

train_mask_pos = train_full.loc[is_pos, "Date"] < cutoff_date
val_mask_pos = ~train_mask_pos

if int(train_mask_pos.sum()) < 60 or int(val_mask_pos.sum()) < 20:
    continue

X_train_pos, X_val_pos = X_pos[train_mask_pos], X_pos[val_mask_pos]
y_train_pos, y_val_pos = y_reg[train_mask_pos], y_reg[val_mask_pos]

y_real_train = np.expml(y_train_pos)
sample_w = 1.0 + np.loglp(y_real_train)

reg = xgb.XGBRegressor(
    n_estimators=params.reg_n_estimators,
    learning_rate=params.reg_lr,
    max_depth=params.reg_max_depth,
    subsample=params.reg_subsample,
    colsample_bytree=params.reg_colsample,
    min_child_weight=params.reg_min_child_weight,
    reg_lambda=params.reg_reg_lambda,
    gamma=params.reg_gamma,
    n_jobs=params.n_jobs,
    random_state=params.random_state,
    tree_method=params.tree_method,
    objective="reg:squarederror",
    eval_metric="rmse",
)

fit_with_early_stopping(
    reg,
    X_train_pos,
    y_train_pos,
    X_val_pos,
    y_val_pos,
    early_stopping_rounds=params.early_stopping_rounds,
    verbose=False,
    sample_weight=sample_w,
)
models_reg[h] = reg

if not models_clf:
    continue

all_models_pack[sku] = {"clf": models_clf, "reg": models_reg}
sku_series = df_long[df_long["SKU"] ==
    sku].sort_values("Date").reset_index(drop=True)
if sku_series.empty:
    continue

max_date = sku_series["Date"].max()
anchors = sku_series[
    (sku_series["Date"] >= cutoff_date)
    & (sku_series["Date"] <= (max_date - pd.Timedelta(days=max(HORIZONS))))
].copy()

if len(anchors) < 10:
    continue

```

```

sku_base_idx = sku_base.copy()
mean_global = float(sku_mean_dict.get(sku, 0.0))
fallback_log = float(np.log1p(max(mean_global, 0.0)))
actual_sums = []
pred_sums = []

cal_pbar = tqdm(anchors["Date"].values, desc=f"Cal({sku})", unit="a",
                leave=False, dynamic_ncols=True)

for anchor_date in cal_pbar:
    anchor_date = pd.Timestamp(anchor_date)
    base_row = safe_get_row(sku_base_idx, anchor_date)
    if base_row is None:
        continue

    base_row = base_row[["Date", "SKU"] + base_feature_cols].copy()

    pred_sum = 0.0
    act_sum = 0.0

    for h in HORIZONS:
        target_date = anchor_date + pd.Timedelta(days=h)

        act = sku_series.loc[sku_series["Date"] == target_date, "Quantity"]
        act_sum += float(act.iloc[0]) if len(act) else 0.0

        tmp = add_target_calendar_features_df(base_row.copy(),
        horizon=int(h))
        X_row = tmp[feature_cols]

        clf = models_clf.get(int(h))
        if clf is not None:
            try:
                p = float(clf.predict_proba(X_row)[0, 1])
            except Exception:
                p = 0.0
        else:
            p = 0.0

        reg = models_reg.get(int(h))
        if reg is not None:
            try:
                pred_log = float(reg.predict(X_row)[0])
            except Exception:
                pred_log = fallback_log
        else:
            pred_log = fallback_log

        pred_qty = float(max(np.expml(pred_log), 0.0) * np.clip(p, 0.0, 1.0))
        pred_sum += pred_qty

    actual_sums.append(act_sum)
    pred_sums.append(pred_sum)

if len(actual_sums) >= 5 and float(np.sum(actual_sums)) > 0:
    wape30 = wape_sum30(np.array(actual_sums), np.array(pred_sums))
    bias_raw = float(np.sum(actual_sums) / (np.sum(pred_sums) + EPS))
    bias = float(np.clip(bias_raw, 0.7, 1.3))
    sku_bias_30[sku] = bias

results.append(
    {
        "SKU": sku,
        "models_clf": len(models_clf),

```

```

        "models_reg": len(models_reg),
        "anchors": len(actual_sums),
        "WAPE_sum30": wape30,
        "bias_raw": bias_raw,
        "bias_30": bias,
    }
)

sku_pbar.set_postfix(
    {
        "clf": len(models_clf),
        "reg": len(models_reg),
        "bias": f"{sku_bias_30.get(sku, 1.0):.2f}",
    }
)

pack = {
    "version": "xgb_direct_zi_v1",
    "models": all_models_pack,
    "features": feature_cols,
    "horizons": HORIZONS,
    "sku_means": sku_means,
    "global_stats": {"mean": sku_mean_dict, "std": sku_std_dict, "cv":
        sku_cv_dict},
    "sku_bias_30": sku_bias_30,
    "meta": {
        "test_days": TEST_DAYS,
        "top_n_skus": TOP_N_SKUS,
        "min_train_points_per_sku": MIN_TRAIN_POINTS_PER_SKU,
        "min_samples_per_horizon": MIN_SAMPLES_PER_HORIZON,
        "context_tail_days": CONTEXT_TAIL_DAYS,
    },
    "xgb_params": params.__dict__,
}

print(f"\nSaving models: {OUT_MODEL_PATH}")
joblib.dump(pack, OUT_MODEL_PATH)

if results:
    dfm = pd.DataFrame(results).sort_values("WAPE_sum30")
    dfm.to_csv(OUT_METRICS_PATH, index=False)
    print(f"Saved metrics: {OUT_METRICS_PATH}")
    print("\nTop-10 by WAPE_sum30:")
    print(dfm.head(10).to_string(index=False))

print("Done.")

if __name__ == "__main__":
    main()

```

Налаштування та ініціалізація моделі Prophet

```

# src/models/train_prophet.py
from __future__ import annotations

import os
import warnings
import logging
from dataclasses import dataclass
from typing import Dict, Any, List, Tuple
import joblib
import numpy as np
import pandas as pd
from prophet import Prophet
from sklearn.metrics import mean_absolute_error, mean_squared_error

warnings.filterwarnings("ignore")
logging.getLogger("cmdstanpy").setLevel(logging.WARNING)

DATA_DIR = "data"

WIDE_PATH = os.path.join(DATA_DIR, "top_products_daily_sales.csv")
XGB_MODEL_PATH = os.path.join(DATA_DIR, "xgb_models_direct.joblib")

OUT_MODEL_PATH = os.path.join(DATA_DIR, "prophet_advanced_v4_models.joblib")
OUT_METRICS_PATH = os.path.join(DATA_DIR, "prophet_advanced_v4_metrics.csv")

TEST_DAYS = 120
MIN_TRAIN_DAYS = 240
MIN_NONZERO_POINTS = 20
FREQ = "D"
EPS = 1e-6

def wape(y_true: np.ndarray, y_pred: np.ndarray) -> float:
    y_true = np.asarray(y_true, dtype=float)
    y_pred = np.asarray(y_pred, dtype=float)
    denom = float(np.sum(np.abs(y_true)))
    if denom <= 0:
        return 0.0
    return float(np.sum(np.abs(y_true - y_pred)) / (denom + EPS))

def wape_sum30_from_daily(y_true_daily: np.ndarray, y_pred_daily: np.ndarray) -> float:
    y_true_daily = np.asarray(y_true_daily, dtype=float)
    y_pred_daily = np.asarray(y_pred_daily, dtype=float)

    if len(y_true_daily) < 30:
        return np.nan

    true_s = pd.Series(y_true_daily).rolling(30).sum().dropna().values
    pred_s = pd.Series(y_pred_daily).rolling(30).sum().dropna().values
    if len(true_s) == 0:
        return np.nan
    return wape(true_s, pred_s)

def bias_sum30(y_true_daily: np.ndarray, y_pred_daily: np.ndarray) -> float:
    y_true_daily = np.asarray(y_true_daily, dtype=float)
    y_pred_daily = np.asarray(y_pred_daily, dtype=float)

```

```

if len(y_true_daily) < 30:
    return np.nan

true_s = pd.Series(y_true_daily).rolling(30).sum().dropna().values
pred_s = pd.Series(y_pred_daily).rolling(30).sum().dropna().values
if len(true_s) == 0:
    return np.nan

denom = float(np.sum(pred_s))
if denom <= 0:
    return np.nan
return float(np.sum(true_s) / (denom + EPS))

@dataclass
class ProphetParams:
    cps_intermittent: float = 0.03
    sps_intermittent: float = 5.0

    cps_dense: float = 0.06
    sps_dense: float = 10.0

    nonzero_dense_threshold: float = 0.60
    nonzero_intermittent_threshold: float = 0.40

def load_wide_matrix(path: str) -> pd.DataFrame:
    if not os.path.exists(path):
        raise FileNotFoundError(
            f"Wide matrix not found: {path}. Run: python -m
            src.data.prepare_online_retail_ii"
        )
    wide = pd.read_csv(path, parse_dates=["Date"], low_memory=False)
    if "Date" not in wide.columns:
        raise ValueError(f"'Date' column missing in {path}")
    wide["Date"] = pd.to_datetime(wide["Date"], errors="coerce").dt.normalize()
    wide = wide.dropna(subset=["Date"]).sort_values("Date").reset_index(drop=True)

    for c in wide.columns:
        if c == "Date":
            continue
        wide[c] = pd.to_numeric(wide[c], errors="coerce").fillna(0.0).clip(lower=0.0)

    return wide

def get_xgb_trained_skus(xgb_path: str) -> List[str]:
    if not os.path.exists(xgb_path):
        raise FileNotFoundError(f"XGB pack not found: {xgb_path}. Run: python -m
            src.models.train_xgb_direct")
    pack = joblib.load(xgb_path)
    models = pack.get("models", {}) or {}
    return sorted([str(s) for s in models.keys()])

def build_daily_series_from_wide(wide: pd.DataFrame, sku: str) -> pd.DataFrame:
    if sku not in wide.columns:
        return pd.DataFrame(columns=["ds", "y"])

    tmp = wide[["Date", sku]].copy()
    tmp.rename(columns={"Date": "ds", sku: "y"}, inplace=True)
    tmp["ds"] = pd.to_datetime(tmp["ds"], errors="coerce").dt.normalize()
    tmp["y"] = pd.to_numeric(tmp["y"], errors="coerce").fillna(0.0).clip(lower=0.0)

```

```

if tmp["ds"].isna().all():
    return pd.DataFrame(columns=["ds", "y"])

start = tmp["ds"].min()
end = tmp["ds"].max()
full = pd.DataFrame({"ds": pd.date_range(start, end, freq=FREQ)})
tmp = full.merge(tmp, on="ds", how="left")
tmp["y"] = tmp["y"].fillna(0.0).clip(lower=0.0)

# Regressor
tmp["is_weekend"] = (tmp["ds"].dt.dayofweek >= 5).astype(int)

return tmp

def pick_params(train_df: pd.DataFrame, pp: ProphetParams) -> Tuple[str, float,
    float, float]:
    y = train_df["y"].values.astype(float)
    nonzero_share = float(np.mean(y > 0))
    if nonzero_share >= pp.nonzero_dense_threshold:
        seasonality_mode = "multiplicative"
        cps, sps = pp.cps_dense, pp.sps_dense
    elif nonzero_share < pp.nonzero_intermittent_threshold:
        seasonality_mode = "additive"
        cps, sps = pp.cps_intermittent, pp.sps_intermittent
    else:
        seasonality_mode = "additive"
        cps, sps = 0.05, 8.0
    return seasonality_mode, float(cps), float(sps), nonzero_share

def main():
    os.makedirs(DATA_DIR, exist_ok=True)

    print("Loading wide matrix...")
    wide = load_wide_matrix(WIDE_PATH)

    print("Loading XGB pack (to get trained SKU list)...")
    xgb_skus = get_xgb_trained_skus(XGB_MODEL_PATH)

    available_skus = set(map(str, wide.columns)) - {"Date"}
    target_skus = [s for s in xgb_skus if s in available_skus]

    print(f"Prophet will train ONLY XGB-trained SKUs: {len(target_skus)}")
    if not target_skus:
        raise RuntimeError("No overlap between XGB-trained SKUs and wide matrix
            columns.")

    pp = ProphetParams()

    models: Dict[str, Any] = {}
    rows: List[Dict[str, Any]] = []

    for i, sku in enumerate(target_skus, start=1):
        print(f"\n--- [{i}/{len(target_skus)}] SKU: {sku} ---")
        df = build_daily_series_from_wide(wide, sku)

        if df.empty:
            print("Empty series, skip.")
            continue

        if len(df) < (TEST_DAYS + MIN_TRAIN_DAYS):
            print(f"Too short series: len={len(df)}. Skip.")
            continue

        train = df.iloc[:-TEST_DAYS].copy()

```

```

test = df.iloc[-TEST_DAYS:].copy()

positives = int((train["y"] > 0).sum())
if positives < MIN_NONZERO_POINTS:
    print(f"Too intermittent for Prophet (positives={positives}). Skip Prophet.")
    continue

seasonality_mode, cps, sps, nonzero_share = pick_params(train, pp)

m = Prophet(
    yearly_seasonality="auto",
    weekly_seasonality="auto",
    daily_seasonality=False,
    seasonality_mode=seasonality_mode,
    changepoint_prior_scale=cps,
    seasonality_prior_scale=sps,
)

try:
    m.add_country_holidays(country_name="UK")
except Exception:
    pass
m.add_regressor("is_weekend", standardize=False)

try:
    m.fit(train[["ds", "y", "is_weekend"]])

    df_future = test[["ds", "is_weekend"]].copy()
    fc = m.predict(df_future)
    yhat = pd.to_numeric(fc["yhat"],
        errors="coerce").fillna(0.0).clip(lower=0.0).values

    y_true = test["y"].values.astype(float)
    y_pred = yhat.astype(float)

    mae = float(mean_absolute_error(y_true, y_pred))
    rmse = float(np.sqrt(mean_squared_error(y_true, y_pred)))
    wape_daily = float(wape(y_true, y_pred))
    wape30 = float(wape_sum30_from_daily(y_true, y_pred))
    bias30_raw = float(bias_sum30(y_true, y_pred))
    bias30 = float(np.clip(bias30_raw, 0.7, 1.3)) if np.isfinite(bias30_raw)
    else np.nan

    print(
        f" nonzero_share={nonzero_share:.2f} | mode={seasonality_mode} | "
        f"cps={cps:.3f} | sps={sps:.1f}"
    )
    print(
        f" MAE={mae:.2f} | RMSE={rmse:.2f} | WAPE_daily={wape_daily:.2%} | "
        f"WAPE_sum30={wape30:.2%} | bias30={bias30:.3f}"
    )

    rows.append(
        {
            "SKU": sku,
            "Model": f"Prophet v4 ({seasonality_mode})",
            "nonzero_share": nonzero_share,
            "positives": positives,
            "cps": cps,
            "sps": sps,
            "MAE": mae,
            "RMSE": rmse,
            "WAPE_daily": wape_daily,

```

```

        "WAPE_sum30": wape30,
        "bias30_raw": bias30_raw,
        "bias30": bias30,
    }
)

m_final = Prophet(
    yearly_seasonality="auto",
    weekly_seasonality="auto",
    daily_seasonality=False,
    seasonality_mode=seasonality_mode,
    changepoint_prior_scale=cps,
    seasonality_prior_scale=sps,
)
try:
    m_final.add_country_holidays(country_name="UK")
except Exception:
    pass
m_final.add_regressor("is_weekend", standardize=False)
m_final.fit(df[["ds", "y", "is_weekend"]])

models[str(sku)] = m_final

except Exception as e:
    print(f" Prophet failed on {sku}: {e}")
    continue

metrics_df = pd.DataFrame(rows)
metrics_df.to_csv(OUT_METRICS_PATH, index=False)
print(f"\nSaved Prophet metrics: {OUT_METRICS_PATH}")

pack = {
    "version": "prophet_v4_xgb_skus",
    "models": models,
    "meta": {
        "test_days": TEST_DAYS,
        "train_only_xgb_skus": True,
        "xgb_model_path": XGB_MODEL_PATH,
        "wide_path": WIDE_PATH,
        "min_train_days": MIN_TRAIN_DAYS,
        "min_nonzero_points": MIN_NONZERO_POINTS,
        "freq": FREQ,
    },
}

joblib.dump(pack, OUT_MODEL_PATH)
print(f"Saved Prophet pack: {OUT_MODEL_PATH}")
print(f"Done. Trained Prophet models: {len(models)} / {len(target_skus)}")

if __name__ == "__main__":
    main()

```