ПЗВО «ПЕРШИЙ ЄВРОПЕЙСЬКИЙ УНІВЕРСИТЕТ»
Навчально-науковий інститут «Європейська школа бізнесу»
Кафедра інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
інформаційних технологій
ННІ «Європейська школа
бізнесу», професор

О. В. Нестеренко

«___ » _____ 2026 р.

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

зі спеціальності F2 «Інженерія програмного забезпечення»

ТЕМА:

«Розробка інтелектуальної системи прогнозування попиту на товари для автоматизації процесів ритейлу»

Виконавець: Чесніший Денис Юрійович

Науковий керівник: Нестеренко О. В.,
доктор технічних наук, професор

КИЇВ – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ПЗВО «ПЕРШИЙ ЄВРОПЕЙСЬКИЙ УНІВЕРСИТЕТ»
Навчально-науковий інститут «Європейська школа бізнесу»
Кафедра інформаційних технологій

Ступінь вищої освіти – перший (бакалаврський) рівень
Спеціальність: F2 «Інженерія програмного забезпечення»
Освітньо-професійна програма: «Інженерія програмного забезпечення»
Освітньо-кваліфікаційний рівень: бакалавр

ЗАТВЕРДЖУЮ
Завідувач кафедри інформаційних технологій
ННІ «Європейська школа бізнесу», професор
_____ О. В. Нестеренко
«____» _____ 2026 р.

З А В Д А Н Н Я
НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ БАКАЛАВРСЬКОЇ РОБОТИ

Студент: Чесніший Денис Юрійович, група ІПЗ-22-401

1. Тема: «Розробка інтелектуальної системи прогнозування попиту на товари для автоматизації процесів ритейлу», затверджена наказом Ректора від «09» лютого 2026 р. №25-с.

2. Термін виконання роботи: з «26» травня 2026 р. по «22» червня 2026 р.

3. Дата подання роботи на випускну кафедру: «12» червня 2026 р.

4. Вихідні дані роботи: Система прогнозування попиту на товари для автоматизації процесів ритейлу призначена для аналітичної підтримки та оптимізації роздрібних операцій, управління запасами, ціноутворення, маркетингу й логістики на основі рекомендацій, сформованих із застосуванням штучного інтелекту. Необхідно розробити модель прогнозування, визначити структури даних і алгоритми їх обробки, спроєктувати структуру бази даних, передбачити захист інформації та санкціонований доступ, а також розглянути основні питання програмної інженерії.

5. Зміст пояснювальної записки:

– Вступ - актуальність теми, мета, завдання, об’єкт, предмет і практичне значення роботи.

– Розділ 1 - аналіз предметної області та формування концепції розробки.

– Розділ 2 - питання програмної інженерії: аналіз вимог, проєктування архітектури, структури даних і прототипу.

– Розділ 3 - технічна реалізація: середовище розробки, мови програмування, інструментарій, реалізація й тестування.

– Список використаних джерел - не менше 30 найменувань.

– Додатки - схеми, скріншоти та фрагменти програмного коду за необхідності.

– Загальний обсяг пояснювальної записки без додатків - 80–90 сторінок; шрифт 14 пт; міжрядковий інтервал 1,5.

6. Перелік обов’язкового графічного матеріалу:

– загальна схема роботи системи Retail Demand Sense;
– ER-діаграма бази даних SQLite;
– структура вхідного CSV-файлу;
– скріншоти web-інтерфейсу;
– графіки прогнозування;
– результати оцінювання якості моделі;
– фрагменти програмного коду;
– результати тестування.

Графічний матеріал для презентації роботи оформлюється у вигляді комп'ютерної презентації обсягом до 15 слайдів.

7. Календарний план-графік виконання КБР

з/п	Завдання	Термін виконання	Відмітка про виконання
	Складання і затвердження індивідуальних завдань на виконання кваліфікаційної бакалаврської роботи (КБР)	27.02.26	
	Підготовка вступу і розділу 1 КБР	31.03.26	
	Підготовка розділу 2 КБР	30.04.26	
	Підготовка розділу 3 КБР, висновків і переліку використаних джерел	29.05.26	
	Подання студентом завершеної КБР науковому керівнику для перевірки на плагіат та оформлення відгуку	05.06.26	
	Попередній розгляд КБР на комісії від кафедри	11–12.06.26	
	Доопрацювання роботи, прийняття кафедрою рішення про допуск роботи до захисту в ЕК, оформлення та зовнішнє рецензування	15–19.06.26	
	Захист кваліфікаційної бакалаврської роботи в ЕК і присвоєння випускникам кваліфікації	23–24.06.26	

Студент _____ Чесніший Д. Ю.
Керівник _____ Нестеренко О. В.

РЕФЕРАТ

Чесніший Д. Ю. Розробка інтелектуальної системи прогнозування попиту на товари для автоматизації процесів ритейлу. Кваліфікаційна бакалаврська робота зі спеціальності F2 «Інженерія програмного забезпечення», ОПП «Інженерія програмного забезпечення», ПЗВО «Перший європейський університет», 2026 р.

Загальний обсяг роботи - 86 сторінок. Робота складається зі вступу, трьох розділів, висновків, списку використаних джерел і додатка; містить 12 рисунків, 1 нумеровану таблицю, 40 джерел та 1 додаток.

Ключові слова: ПРОГНОЗУВАННЯ ПОПИТУ, РИТЕЙЛ, МАШИННЕ НАВЧАННЯ, RETAIL DEMAND SENSE, FLASK, SQLITE, CSV, MAE, RMSE, MAPE.

Актуальність теми зумовлена необхідністю автоматизації аналітичних процесів у сфері ритейлу. Ручний аналіз історичних продажів не завжди дає змогу своєчасно врахувати сезонність, календарні закономірності, цінові зміни, акційні періоди та попередню динаміку попиту. Застосування методів машинного навчання дає можливість опрацьовувати сукупність факторів, виявляти приховані залежності та отримувати кількісну оцінку майбутнього попиту для підтримки управлінських рішень.

Метою роботи є розробка інтелектуальної системи, яка на основі історичних даних продажів формує прогноз попиту на товари ритейлу, оцінює якість прогнозування та надає користувачеві зрозумілий web-інтерфейс для аналізу результатів.

Для досягнення мети проаналізовано предметну область ритейлу; визначено структуру вхідних CSV-даних; спроєктовано базу даних SQLite; реалізовано імпорт, перевірку та попередню обробку даних; сформовано ознаки для машинного навчання; навчено й порівняно регресійні моделі; реалізовано прогнозування попиту та оцінювання якості за метриками MAE,

RMSE і MAPE; розроблено Flask web-інтерфейс; проведено тестування основних функцій системи.

Об’єктом дослідження є процес прогнозування попиту на товари у сфері роздрібної торгівлі. Предметом дослідження є методи машинного навчання, структури даних, алгоритми обробки інформації та програмні засоби реалізації системи прогнозування попиту.

Практичне значення одержаних результатів полягає у створенні працездатного програмного прототипу Retail Demand Sense, який охоплює повний цикл від імпорту історичних продажів до побудови прогнозу, оцінювання моделі та візуалізації результатів. Система може використовуватися як навчально-прикладний інструмент і засіб аналітичної підтримки менеджерів ритейлу.

ABSTRACT

Chesnishyi D. Yu. Development of an Intelligent Product Demand Forecasting System for Retail Process Automation. Bachelor's qualification thesis in specialty F2 "Software Engineering", Educational and Professional Program "Software Engineering", First European University, 2026.

The total volume of the thesis is 86 pages. It consists of an introduction, three chapters, conclusions, a list of references, and one appendix; it contains 12 figures, 1 numbered table, 40 references, and 1 appendix.

Keywords: DEMAND FORECASTING, RETAIL, MACHINE LEARNING, RETAIL DEMAND SENSE, FLASK, SQLITE, CSV, MAE, RMSE, MAPE.

The relevance of the research is determined by the need to automate analytical processes in retail. Manual analysis of historical sales does not always allow seasonal patterns, calendar effects, price changes, promotional periods, and previous demand dynamics to be considered in a timely manner. Machine learning methods make it possible to process multiple factors, identify hidden dependencies, and obtain a quantitative estimate of future demand to support managerial decision-making.

The purpose of the thesis is to develop an intelligent system that forecasts retail product demand from historical sales data, evaluates forecasting quality, and provides a clear web interface for analyzing the results.

To achieve this purpose, the retail domain was analyzed; the structure of input CSV data was defined; a SQLite database was designed; data import, validation, and preprocessing were implemented; machine learning features were generated; regression models were trained and compared; demand forecasting and quality evaluation using MAE, RMSE, and MAPE were implemented; a Flask web interface was developed; and the core system functions were tested.

The object of the research is the process of forecasting product demand in retail. The subject of the research comprises machine learning methods, data

structures, information-processing algorithms, and software tools for implementing a demand forecasting system.

The practical significance of the results lies in the development of a functional Retail Demand Sense prototype covering the complete workflow from importing historical sales data to generating forecasts, evaluating the model, and visualizing results. The system can be used as an educational and applied tool and as analytical decision support for retail managers.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Позначення	Розшифрування
API	Application Programming Interface - інтерфейс програмування застосунків
CSS	Cascading Style Sheets - каскадні таблиці стилів
CSV	Comma-Separated Values - текстовий формат табличних даних із розділювачами
ERP	Enterprise Resource Planning - система планування ресурсів підприємства
Flask	мікрофреймворк Python для створення web-застосунків
HTML	HyperText Markup Language - мова розмітки гіпертексту
JavaScript	мова програмування для реалізації інтерактивності web-інтерфейсу
MAE	Mean Absolute Error - середня абсолютна помилка прогнозування
MAPE	Mean Absolute Percentage Error - середня абсолютна відсоткова помилка
ML	Machine Learning - машинне навчання
POS	Point of Sale - система обліку продажів у точці реалізації
Python	мова програмування, використана для серверної логіки й машинного навчання
RMSE	Root Mean Squared Error - корінь із середньоквадратичної помилки
SQLite	вбудована реляційна система керування базами даних
UI	User Interface - користувацький інтерфейс

ЗМІСТ

ВСТУП

Розроблення виконано з урахуванням загальних вимог до структури звітів та бібліографічного оформлення [1; 2; 40].

Роздрібна торгівля є однією з найбільш динамічних сфер економічної діяльності, оскільки вона безпосередньо пов’язана з поведінкою споживачів, зміною купівельної спроможності, сезонністю, конкуренцією, ціноутворенням, маркетинговими активностями та організацією постачання товарів. У сучасних умовах ритейл працює з великими обсягами даних: щоденними продажами, товарними залишками, цінами, категоріями товарів, акційними періодами, календарними подіями та логістичними характеристиками. Такі дані можуть містити цінну інформацію для прийняття управлінських рішень, однак їх ручний аналіз часто є складним, трудомістким і недостатньо точним.

Одним із важливих завдань у ритейлі є прогнозування попиту на товари. Попит визначає, яку кількість товару покупці можуть придбати в певний період часу. Для підприємств роздрібної торгівлі правильне розуміння майбутнього попиту має практичне значення, оскільки воно впливає на планування запасів, оцінку сезонності, аналіз ефективності акцій, формування цінової політики, роботу з постачальниками та загальну організацію роздрібних операцій. Якщо попит оцінено неточно, це може призвести до надлишку товарів, дефіциту позицій, нераціонального використання складських ресурсів або втрати потенційного продажу. Водночас прогноз не повинен розглядатися як автоматичне управлінське рішення. Він є інструментом підтримки аналізу, який допомагає менеджеру краще оцінити ситуацію та прийняти обґрунтоване рішення.

Актуальність теми кваліфікаційної роботи зумовлена необхідністю автоматизації аналітичних процесів у сфері ритейлу. Традиційний підхід, за якого менеджер аналізує продажі вручну або орієнтується переважно на досвід, не завжди дозволяє врахувати велику кількість факторів. Попит може змінюватися залежно від дня тижня, місяця, сезону, ціни, наявності акції,

категорії товару, попередньої динаміки продажів та інших обставин. У звичайних табличних звітах такі закономірності не завжди очевидні. Тому виникає потреба в інформаційній системі, яка здатна збирати історичні дані, структурувати їх, формувати ознаки для подальшої обробки, будувати прогноз та подавати результат у зрозумілому для користувача вигляді.

Використання методів машинного навчання є доцільним для задач прогнозування попиту, оскільки такі методи дозволяють знаходити залежності між попередніми продажами та факторами, які можуть впливати на майбутні значення. На відміну від простого усереднення або ручного аналізу, регресійна модель може враховувати набір ознак одночасно. До таких ознак можуть належати попередні продажі, ковзні середні, день тижня, місяць, акційний період, ціна товару, категорія та інші характеристики. Це не означає, що модель гарантує абсолютно точний результат, однак вона дає можливість отримати кількісну оцінку очікуваного попиту і перевірити якість прогнозування за допомогою метрик. У роботі враховано підходи до прогнозування попиту та ритейл-аналітики, описані у працях з forecasting і retail forecasting [35; 36].

Тема роботи - «Розробка інтелектуальної системи прогнозування попиту на товари для автоматизації процесів ритейлу» - поєднує два напрями: прикладну задачу ритейлу та програмну реалізацію інтелектуальної системи. З одного боку, досліджується проблема прогнозування попиту на товари, яка має безпосереднє практичне значення для роздрібної торгівлі. З іншого боку, робота передбачає створення програмного прототипу, у якому реалізовано обробку даних, збереження в базі даних, навчання моделей, генерацію прогнозу, оцінку точності та інтерфейс для взаємодії з користувачем.

У межах роботи розробляється система Retail Demand Sense. Вона призначена для аналізу даних продажів товарів і побудови прогнозу майбутнього попиту. Система працює з CSV-файлом, який містить записи продажів, товари, категорії, ціни, кількість проданих одиниць та допоміжні ознаки. Після імпорту дані очищуються, нормалізуються та можуть

зберігатися у SQLite. Для побудови прогнозу використовується ML-підхід: на основі підготовлених даних формуються ознаки, навчаються регресійні моделі, порівнюються їхні результати, після чого обрана модель застосовується для прогнозування попиту на майбутній період.

Розроблена система не належить до промислових ERP-, POS-або складських платформ. Її призначення полягає в демонстрації програмної реалізації підходу до прогнозування попиту та аналітичної підтримки ритейлу. Система не здійснює автоматичне замовлення товарів, не змінює ціни, не запускає маркетингові кампанії та не оптимізує логістичні маршрути. Водночас результати прогнозування можуть бути використані як інформаційна основа для таких процесів. Наприклад, менеджер може переглянути очікуваний попит, оцінити динаміку продажів, звернути увагу на сезонні коливання та використати отриману інформацію під час планування роздрібних операцій.

Метою кваліфікаційної роботи є розробка інтелектуальної системи, яка на основі даних продажів формує прогноз попиту на товари ритейлу, оцінює якість прогнозування та надає користувачу зрозумілий інтерфейс для аналізу результатів.

Для досягнення поставленої мети необхідно виконати такі завдання:

1. Проаналізувати предметну область ритейлу та визначити особливості попиту на товари.

2. Дослідити проблему прогнозування попиту та фактори, які впливають на динаміку продажів.

3. Визначити структуру вхідних даних, необхідних для навчання моделі прогнозування.

4. Розробити структуру бази даних для збереження товарів, історичних продажів, результатів імпорту, прогнозів і метрик.

5. Реалізувати імпорт CSV-файлу з історичними даними продажів.

6. Реалізувати попередню обробку даних і формування ознак для машинного навчання.

7. Навчити та порівняти кілька моделей машинного навчання для задачі прогнозування попиту.

8. Обрати модель з найкращими показниками якості на тестовій вибірці.

9. Реалізувати побудову прогнозу попиту для вибраного товару та горизонту прогнозування.

10. Оцінити якість прогнозування за метриками MAE, RMSE і MAPE. Застосування інформаційних систем у підприємствах та аналізі маркетингових даних також розглядається у працях викладачів кафедри [37; 38].

11. Розробити web-інтерфейс для перегляду набору даних, прогнозу, метрик моделі та опису системи.

12. Провести тестування основних функцій системи та перевірити її працездатність.

Об’єктом дослідження є процес прогнозування попиту на товари у сфері роздрібної торгівлі.

Предметом дослідження є методи машинного навчання, структури даних, алгоритми обробки інформації та програмна реалізація системи прогнозування попиту.

У роботі використовуються такі методи дослідження: аналіз предметної області ритейлу; аналіз і структурування вхідних даних; методи попередньої обробки табличних даних; feature engineering для формування ознак; методи машинного навчання для регресійного прогнозування; порівняльний аналіз моделей; методи оцінки якості прогнозу за допомогою MAE, RMSE і MAPE; методи програмної інженерії для проєктування, реалізації та тестування web-застосунку.

Практична цінність роботи полягає у створенні програмного прототипу, який демонструє повний цикл прогнозування попиту: від завантаження даних продажів до побудови прогнозу, оцінки якості моделі та представлення результатів у користувацькому інтерфейсі. Такий підхід може бути корисним для аналізу продажів у ритейлі, попереднього планування запасів, оцінки сезонності, аналізу впливу ціни та акцій, а також обґрунтування управлінських

рішень. Практична цінність системи полягає не в автоматичному заміщенні менеджера, а в наданні йому структурованої аналітичної інформації.

Програмна реалізація виконана з використанням Python, Flask, SQLite, pandas, NumPy, scikit-learn, HTML, CSS і JavaScript. Backend забезпечує обробку запитів, маршрути застосунку, API, імпорт CSV і взаємодію з модулями прогнозування. SQLite використовується як локальна база даних для збереження структурованої інформації. Модулі обробки даних і ML реалізують підготовку даних, формування ознак, навчання моделей, вибір моделі та генерацію прогнозу. Інтерфейс дає змогу переглядати огляд системи, структуру набору даних, прогноз попиту, метрики моделі та опис системи.

Пояснювальна записка складається зі вступу, трьох розділів, висновків, списку використаних джерел і додатка. У вступі обґрунтовується актуальність теми, визначаються мета, завдання, об’єкт, предмет, методи дослідження та практична цінність роботи. У першому розділі розглядається предметна область ритейлу, проблема прогнозування попиту, фактори впливу на продажі, підходи до прогнозування та концепція розробленої системи. У другому розділі описуються питання програмної інженерії: вимоги до системи, структура даних, база даних, архітектура та алгоритми роботи. У третьому розділі подається технічна реалізація системи, описуються CSV-імпорт, preprocessing, feature engineering, навчання моделей, побудова прогнозу, метрики та тестування. У висновках узагальнюються результати роботи, а в додатку наведено фрагменти програмного коду, які підтверджують основні технічні рішення системи.

Таким чином, робота спрямована на розв’язання прикладної задачі прогнозування попиту в ритейлі засобами програмної інженерії та методів ML. Розроблена система дозволяє показати, як дані продажів можуть бути перетворені на аналітичний прогноз, придатний для використання в управлінському аналізі.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА КОНЦЕПЦІЯ СИСТЕМИ

1.1. Предметна область ритейлу та попиту на товари

Теоретичний опис предметної області спирається на джерела з управління ритейлом, ланцюгами постачання та операційним менеджментом [28; 29; 31; 33].

Ритейл, або роздрібна торгівля, є сферою економічної діяльності, у якій товари реалізуються кінцевим споживачам для особистого використання. На відміну від оптової торгівлі, ритейл працює з великою кількістю окремих покупців, широким асортиментом товарів, постійною зміною купівельної поведінки та необхідністю швидко реагувати на попит. До ритейлу можуть належати продуктові магазини, супермаркети, магазини побутових товарів, аптеки, інтернет-магазини та інші формати роздрібної торгівлі.

Особливістю ритейлу є те, що ефективність роботи значною мірою залежить від правильного розуміння потреб покупців. Підприємство має забезпечити наявність товарів, які користуються попитом, у потрібний час і в потрібній кількості. Якщо товару недостатньо, покупець може не здійснити покупку або обрати конкурента. Якщо товару надто багато, виникають додаткові витрати на зберігання, ризик псування або морального старіння продукції, а також неефективне використання оборотних коштів. Тому прогнозування попиту є важливим елементом управління роздрібними операціями.

Попит на товари можна розглядати як кількість продукції, яку споживачі готові придбати за певний період часу за певних умов. У практичному сенсі для ритейлу попит часто оцінюється через фактичні продажі, оскільки саме історія продажів є доступним джерелом даних про поведінку покупців. Однак фактичні продажі не завжди повністю дорівнюють реальному попиту. Наприклад, якщо товар був відсутній на полиці або його залишок був обмежений, продажі могли бути нижчими не через слабкий інтерес покупців,

а через недостатню доступність товару. Саме тому під час аналізу попиту важливо враховувати не лише кількість проданих одиниць, а й інші характеристики товару та умов продажу.

Попит у ритейлі не є сталим. Він змінюється під впливом багатьох факторів, частина з яких має регулярний характер, а частина може виникати випадково. Одним із базових факторів є сезонність. Наприклад, попит на напої може зростати в теплий період, а попит на певні харчові продукти може залежати від свят або календарних подій. Іншим фактором є день тижня: у вихідні покупці можуть частіше здійснювати великі покупки, тоді як у будні попит може бути більш рівномірним або нижчим.

Важливу роль відіграє ціна товару. Зміна ціни може впливати на рішення покупця, особливо якщо товар має аналоги або належить до категорії з високою чутливістю до ціни, що узгоджується з підходами до аналізу маркетингових факторів у ритейлі [30]. Зниження ціни або проведення акції може тимчасово підвищити попит, тоді як підвищення ціни може зменшити кількість покупок. Саме тому в системах прогнозування попиту доцільно враховувати ціну як одну з ознак, що характеризує умови продажу.

Маркетингові активності також впливають на динаміку попиту. Акції, знижки, спеціальні пропозиції, святкові кампанії або розміщення товару в більш помітній зоні магазину можуть змінити поведінку покупців. У даних такі фактори можуть бути представлені як ознака акційного періоду або святкового дня. Хоча сама по собі наявність акції не гарантує зростання продажів для кожного товару, вона є важливим сигналом для моделі прогнозування.

На попит впливає і категорія товару. Різні категорії мають різну частоту покупки, термін зберігання, сезонність і чутливість до ціни. Наприклад, товари щоденного споживання можуть мати стабільніший попит, тоді як напої, кондитерські вироби або сезонні товари можуть демонструвати помітніші коливання. Категорія товару допомагає моделі враховувати загальні властивості групи товарів.

Окремо варто зазначити вплив логістичних і складських факторів, які розглядаються в працях з логістики, управління запасами та ланцюгами постачання [32; 34]. Якщо постачання затримується або товар має обмежений залишок, фактичні продажі можуть спотворювати уявлення про реальний попит. У такій ситуації історичні продажі слід аналізувати обережно, оскільки низький обсяг реалізації може бути наслідком не слабкого попиту, а операційних обмежень. У межах розробленої системи такі характеристики не використовуються для автоматичного логістичного планування, однак вони можуть бути частиною вхідних даних і допоміжного аналізу.

Для ритейлу прогнозування попиту важливе з кількох причин. По-перше, воно допомагає оцінити очікувану динаміку продажів і краще розуміти, як може змінюватися інтерес покупців до конкретного товару. По-друге, прогноз може бути основою для попереднього планування запасів, оскільки менеджер отримує орієнтовне значення майбутнього попиту. По-третє, аналіз попиту дає змогу оцінювати вплив сезонності, ціни, акцій і календарних факторів. По-четверте, прогнозування сприяє більш обґрунтованому прийняттю управлінських рішень, оскільки рішення спирається не лише на інтуїцію, а й на дані.

У сучасних умовах ритейл усе більше залежить від інформаційних систем, які здатні обробляти великі обсяги даних і подавати результати у зрозумілому вигляді. Важливо, щоб така система не тільки виконувала розрахунок, а й пояснювала його результат користувачу. Для менеджера недостатньо побачити одне число прогнозу; потрібно також розуміти, який товар аналізується, за який період побудовано прогноз, якою є середня очікувана кількість продажів, яка динаміка попиту та які фактори могли вплинути на результат. Саме тому в розробленій системі важливе місце займає інтерфейс, що поєднує графік, таблицю прогнозу, метрики моделі та текстову інтерпретацію.

Таким чином, предметна область ритейлу характеризується великою кількістю змінних факторів, які впливають на продажі. Попит на товари

залежить від історичної динаміки, ціни, сезонності, акцій, календарних ознак, категорії товару та операційних умов. Прогнозування попиту дозволяє перетворити історичні продажі на аналітичну інформацію, яка може використовуватися для підтримки рішень у роздрібній торгівлі.

1.2. Проблемна ситуація прогнозування попиту

Проблемна ситуація прогнозування попиту полягає в тому, що ритейл працює з великою кількістю товарів, кожен з яких має власну динаміку продажів. Навіть у межах одного магазину або одного набору даних товари можуть суттєво відрізнятися за частотою покупки, сезонністю, реакцією на акції, рівнем ціни та стабільністю попиту. Для менеджера ручний аналіз такої інформації стає складним, особливо якщо історія продажів охоплює тривалий період і містить багато щоденних записів.

Ручний аналіз продажів зазвичай передбачає перегляд таблиць, порівняння значень за періодами, пошук зростання або зниження продажів, оцінку середніх значень і спробу зробити висновок про майбутній попит. Такий підхід може бути прийнятним для невеликої кількості товарів, але він швидко втрачає ефективність зі збільшенням асортименту. Людині складно одночасно враховувати сезонність, день тижня, зміну ціни, акції, попередні продажі та поведінку окремих категорій товарів. Крім того, ручний аналіз залежить від досвіду конкретного фахівця і може бути суб’єктивним.

Помилки прогнозування попиту створюють практичні проблеми для роздрібної торгівлі. Якщо очікуваний попит завищено, підприємство може сформувати надлишкові запаси, що призводить до додаткових витрат і неефективного використання ресурсів. Якщо попит занижено, може виникнути нестача товару, що впливає на якість обслуговування покупців і потенційні продажі. У випадку товарів з обмеженим терміном придатності неточний прогноз може бути особливо небажаним, оскільки надлишок продукції може призвести до втрат. У випадку товарів з високою сезонністю

запізніла реакція на зміну попиту може зменшити ефективність управлінських рішень.

Важливо також, що попит не завжди змінюється лінійно. Для деяких товарів характерна стабільна динаміка, для інших - сезонні коливання, періодичні піки або поступове зростання чи спадання. Якщо аналізувати тільки загальне середнє значення продажів, можна втратити важливі закономірності. Наприклад, середній попит може виглядати стабільним, але в окремі дні тижня продажі можуть суттєво відрізнятися. Або навпаки: товар може мати кілька різких піків через акції, але в інші періоди продаватися рівномірно. Такі особливості потребують більш структурованого підходу до аналізу.

Історичні продажі є основним джерелом інформації для оцінки майбутнього попиту. Якщо товар продавався стабільно протягом тривалого часу, це може свідчити про відносно передбачувану поведінку покупців. Якщо останні продажі відрізняються від попереднього рівня, це може бути сигналом зміни динаміки. Якщо продажі мають повторювані піки, це може вказувати на сезонність або календарну залежність. Проте самі по собі історичні дані не дають готової відповіді. Їх потрібно очистити, структурувати, перетворити на ознаки та використати в алгоритмі прогнозування.

Для цього потрібна інформаційна система, яка автоматизує ключові етапи аналізу. Така система повинна приймати структуровані вхідні дані, виконувати їхню перевірку й підготовку, формувати ознаки, застосовувати модель прогнозування, оцінювати якість результату та подавати інформацію користувачу у зручній формі. Важливо, щоб система не була лише набором технічних скриптів, а мала зрозумілий інтерфейс, через який користувач може вибрати товар, задати горизонт прогнозування, переглянути графік і прочитати коротку інтерпретацію.

Розроблена система Retail Demand Sense спрямована саме на підтримку такого процесу. Вона дозволяє працювати з CSV-файлом історичних продажів, переглядати структуру набору даних, будувати прогноз для

вибраного товару та аналізувати якість моделі. На сторінці прогнозу фактичні продажі відображаються окремо від майбутнього прогнозу: історичні значення показані синьою лінією, прогнозні - зеленою пунктирною лінією після останньої історичної дати. Це важливо з точки зору коректного сприйняття результату, оскільки користувач чітко бачить межу між фактом і прогнозом.

Проблема прогнозування попиту також пов’язана з необхідністю оцінювати якість моделі. Якщо система будує прогноз, але не показує його точність, користувач не може зрозуміти, наскільки надійним є підхід. Тому в розробленій системі передбачено сторінку оцінки моделі, де показано MAE, RMSE і MAPE, а також порівняння кількох алгоритмів машинного навчання. Це дозволяє пояснити, чому обрано конкретну модель, і показати, що прогнозування не є випадковим розрахунком, а базується на перевіреному підході.

Потреба в інформаційній системі для підтримки аналізу пояснюється ще й тим, що результати прогнозування мають бути доступними не лише розробнику, а й користувачу, який оцінює попит. Якщо прогноз існує тільки як результат виконання Python-скрипту, його складно використовувати в управлінському процесі. Web-інтерфейс робить результат більш зрозумілим: він показує ключові показники, графік, фактори впливу, таблицю прогнозу та опис якості моделі. Такий підхід відповідає практичному характеру кваліфікаційної роботи з інженерії програмного забезпечення.

Отже, проблемна ситуація полягає у складності ручного аналізу продажів, багатофакторності попиту та ризику помилкових управлінських оцінок. Дані про попередні продажі можуть бути основою для прогнозування, але для цього потрібні обробка даних, формування ознак, регресійна модель та зрозуміле представлення результату. Саме цю задачу вирішує розроблена система, яка поєднує аналіз даних, прогнозування та інтерфейс для прийняття рішень у ритейлі.

1.3. Огляд підходів до прогнозування попиту

Для опису підходів до прогнозування використано праці з часових рядів, статистичного прогнозування та retail forecasting [35; 36].

Прогнозування попиту може виконуватися різними методами, вибір яких залежить від характеру даних, доступної історії продажів, стабільності попиту, кількості товарів, наявності додаткових факторів і вимог до точності. У загальному вигляді під прогнозуванням розуміють побудову оцінки майбутнього значення певного показника на основі попередніх спостережень і відомих умов. У ритейлі таким показником найчастіше є кількість проданих одиниць товару за певний період.

Класичні підходи до прогнозування попиту часто базуються на статистичному аналізі історичних даних. Найпростішим способом є використання середніх значень за попередні періоди. Наприклад, якщо товар протягом останніх двох тижнів продавався приблизно однаково, можна використати середній денний продаж як орієнтир для наступних днів. Такий підхід простий для розуміння і може бути корисним для товарів зі стабільним попитом. Однак він має суттєве обмеження: середнє значення згладжує коливання і не завжди враховує сезонність, акції, зміну ціни або відмінності між днями тижня.

Іншим класичним підходом є аналіз тренду. Тренд показує загальний напрям зміни показника: зростання, спадання або відносну стабільність. Для ритейлу це може бути важливо, оскільки попит на окремі товари може поступово збільшуватися або зменшуватися. Наприклад, якщо продажі товару протягом останніх тижнів зростають, прогноз на майбутній період може враховувати цю тенденцію. Водночас тренд не завжди є стійким. Короткочасне зростання може бути пов’язане з акцією або сезонним фактором, а не з довгостроковою зміною попиту.

Поширеним напрямом є аналіз часових рядів. Часовий ряд є послідовністю значень, упорядкованих у часі. У задачах ритейлу часовим рядом може бути щоденна кількість продажів конкретного товару. Аналіз

часових рядів дозволяє враховувати послідовність спостережень, сезонність, циклічність і залежність майбутніх значень від попередніх. До відомих статистичних моделей часових рядів належать, наприклад, ARIMA та її модифікації. Такі методи часто використовуються для даних, у яких добре виражені часові закономірності. У сучасній практиці також застосовуються інструменти на зразок Prophet або нейронні мережі, зокрема LSTM, однак у межах даної програмної реалізації ці методи не використовуються; у роботі вони згадуються лише як можливий напрям подальшого розвитку відповідно до сучасних підходів машинного навчання [25; 26]. Їх доцільно розглядати як альтернативні підходи, які можуть бути предметом подальшого розвитку системи.

Для задач ритейлу важливо, що попит залежить не тільки від попередніх значень продажів, а й від додаткових ознак. Наприклад, два однакові значення продажів можуть мати різне пояснення залежно від ціни, дня тижня, акції або категорії товару. Саме тому для табличних даних ритейлу доцільним є модельний підхід, у якому алгоритм отримує не лише минулі значення продажів, а й набір пояснювальних ознак. Такий підхід дозволяє розглядати прогнозування попиту як задачу регресії, де потрібно передбачити числове значення майбутніх продажів.

ML-підхід має кілька переваг у контексті табличних даних ритейлу. По-перше, він дозволяє одночасно враховувати різні типи ознак: числові, календарні, категоріальні та похідні від попередніх продажів. По-друге, такі моделі можуть виявляти складніші залежності, ніж просте середнє або лінійний тренд. По-третє, якість моделей можна порівнювати за формальними метриками, що робить вибір моделі більш обґрунтованим. По-четверте, після навчання модель може використовуватися для швидкого отримання прогнозу для різних товарів і періодів.

У реалізованій системі Retail Demand Sense використано саме regression-підхід. Це означає, що цільовим значенням є кількість продажів товару, а модель навчається встановлювати залежність між цією кількістю та набором

ознак. До таких ознак належать календарні характеристики, попередні продажі, ковзні середні, ціна, категорія товару та ознака акційного періоду. Такий підхід відповідає структурі наявних даних, оскільки система працює з табличним CSV-набором продажів.

Водночас будь-яке прогнозування має обмеження. Модель не може точно передбачити всі майбутні події, особливо якщо вони не представлені в історичних даних. Різкі зміни ринку, несподівані перебої постачання, зовнішні економічні чинники або зміни поведінки покупців можуть вплинути на фактичний попит і зменшити точність прогнозу. Крім того, якість прогнозування залежить від якості вхідних даних: якщо дані неповні, містять помилки або не охоплюють достатній період, модель може давати менш надійні результати.

Тому прогноз у системі слід розглядати не як гарантоване значення майбутніх продажів, а як аналітичну оцінку очікуваного попиту. Він допомагає користувачу побачити можливу динаміку, порівняти товари, оцінити вплив факторів і прийняти більш обґрунтоване управлінське рішення. Такий підхід є коректним для дипломної системи прогнозування попиту, оскільки поєднує прикладну задачу ритейлу з методами аналізу даних та програмною реалізацією.

1.4. Машинне навчання для задач прогнозування попиту

Під час опису ML-підходу враховано джерела щодо регресійних моделей, ансамблевих алгоритмів і оцінювання якості моделей [13; 21; 22; 24; 27].

Задача прогнозування попиту в межах розробленої системи розглядається як задача регресії. Регресія є типом задачі машинного навчання, у якій потрібно передбачити числове значення на основі набору вхідних ознак. У даному випадку таким числовим значенням є кількість проданих одиниць товару за певну дату або прогнозований обсяг продажів у майбутньому

періоді. На відміну від класифікації, де результатом є належність до певного класу, регресія дає кількісну оцінку.

Для використання машинного навчання історичні продажі потрібно перетворити на навчальний набір даних. Початковий CSV-файл містить записи про продажі: дату, товар, категорію, кількість проданих одиниць, ціну, ознаки акції, святкового дня, залишку та інші допоміжні характеристики. Однак модель машинного навчання не працює безпосередньо з неструктурованим описом ситуації. Їй потрібно надати набір ознак, які чисельно або категоріально описують умови продажу. Саме для цього використовується етап feature engineering.

Feature engineering є процесом створення нових ознак на основі початкових даних. У задачі прогнозування попиту цей етап має особливе значення, оскільки майбутні продажі часто залежать від попередньої динаміки. Наприклад, якщо товар активно продавався протягом останнього тижня, це може бути важливим сигналом для прогнозу. Якщо середні продажі за останні 14 днів вищі за попередній рівень, це може свідчити про зміну попиту. Якщо певний день є вихідним або належить до місяця з підвищеною сезонністю, це також може вплинути на очікувану кількість продажів.

У реальному проєкті формування ознак реалізовано в модулі src/feature_engineering.py. Система створює календарні ознаки, ознаки категорії, лаги продажів, ковзні середні та інші характеристики, потрібні для навчання моделі. Серед ознак, які можуть використовуватися в задачі прогнозування, є попередні продажі, середні продажі за 7 і 14 днів, день тижня, місяць, категорія товару, ціна та акційний період. Такі ознаки дозволяють моделі враховувати не лише поточний запис, а й контекст попередньої поведінки товару.

Попередні продажі використовуються у вигляді лагових ознак. Лаг означає значення показника за попередній період. Наприклад, продажі за попередній тиждень можуть допомогти оцінити, чи зберігається попит на товар. Ковзні середні за 7 і 14 днів згладжують випадкові коливання та дають

моделі стабільнішу оцінку короткострокового рівня попиту. День тижня та місяць допомагають враховувати календарні закономірності. Категорія товару дає моделі інформацію про тип товару, а ціна і акційний період описують умови продажу.

Після формування ознак дані поділяються на навчальну та тестову вибірки. Навчальна вибірка використовується для побудови моделі, тобто для того, щоб алгоритм знайшов залежності між ознаками та кількістю продажів. Тестова вибірка використовується для перевірки якості моделі на даних, які не використовувалися під час навчання. Такий поділ потрібний для того, щоб оцінити не лише здатність моделі запам’ятати історичні дані, а й її здатність узагальнювати закономірності для нових спостережень.

У проєкті навчання моделей реалізовано в src/train_model.py. У цьому модулі порівнюються кілька алгоритмів машинного навчання: RandomForestRegressor, GradientBoostingRegressor та ExtraTreesRegressor, теоретичні основи яких пов’язані з ансамблевими методами, зокрема випадковими лісами, градієнтним бустингом та екстремально рандомізованими деревами [22; 23; 24]. Вибір кількох моделей є важливим, оскільки різні алгоритми можуть по-різному працювати з однаковими даними. Одна модель може краще враховувати нелінійні залежності, інша - давати стабільніші результати на певних типах ознак. Тому вибір моделі не повинен базуватися лише на припущенні або популярності алгоритму. Він має спиратися на результати вимірювання якості.

Для оцінки якості прогнозування в системі використовуються метрики MAE, RMSE і MAPE. MAE показує середню абсолютну помилку в одиницях товару. Якщо MAE дорівнює певному значенню, це означає, що в середньому прогноз відхиляється від фактичних продажів приблизно на таку кількість одиниць. RMSE також вимірює помилку прогнозу, але сильніше реагує на великі відхилення. Це корисно, коли потрібно враховувати не лише середню помилку, а й ризик значних промахів. MAPE показує середню відносну

похибку у відсотках, тому її зручно використовувати для порівняння якості моделей.

У файлі models/metrics.json зберігаються результати навчання, назва обраної моделі, значення метрик, кількість навчальних і тестових записів, а також важливість ознак. Ці дані використовуються на сторінці «Модель» web-інтерфейсу, де користувач може побачити, яка модель обрана, які значення MAE, RMSE і MAPE отримано та які фактори найбільше впливають на прогноз. Такий підхід підвищує прозорість системи: користувач бачить не лише результат прогнозування, а й оцінку якості моделі.

У поточній реалізації основною моделлю обрано ExtraTreesRegressor, оскільки під час порівняння вона показала найнижче значення MAPE серед протестованих алгоритмів. Це означає, що саме ця модель дала найменшу середню відносну похибку на тестовій вибірці. Важливо, що вибір моделі здійснюється не довільно, а на основі вимірюваних метрик. Для задач прогнозування попиту це є принциповим, оскільки користувачу потрібно мати підставу довіряти підходу, який застосовується для оцінки майбутніх продажів.

Отже, машинне навчання в задачі прогнозування попиту дозволяє перейти від простого перегляду історичних продажів до побудови кількісного прогнозу. Історичні дані перетворюються на ознаки, моделі навчаються на цих ознаках, результати порівнюються за метриками, а обрана модель використовується для прогнозування. Такий підхід відповідає меті роботи, оскільки забезпечує програмну реалізацію інтелектуальної системи для аналізу попиту в ритейлі.

1.5. Концепція розробленої системи Retail Demand Sense

Концепція розробленої системи Retail Demand Sense полягає у створенні web-застосунку, який поєднує обробку історичних продажів, машинне навчання та зрозуміле представлення результатів прогнозування. Система призначена для того, щоб користувач міг завантажити або використати набір

даних про продажі, переглянути його структуру, побудувати прогноз попиту для вибраного товару, оцінити якість моделі та отримати коротке пояснення результату.

Загальна ідея системи полягає в тому, що історичні продажі розглядаються як джерело інформації про поведінку попиту. Спочатку дані потрапляють у систему у вигляді CSV-файлу. Далі вони проходять перевірку, очищення та нормалізацію. Після цього на основі початкових даних формуються ознаки для машинного навчання. Навчена модель використовує ці ознаки для побудови прогнозу майбутнього попиту. Результат прогнозування передається у web-інтерфейс, де відображається у вигляді графіка, таблиці та текстової інтерпретації.

Повний шлях даних у системі можна описати так: CSV → очищення → нормалізація → формування ознак → ML-модель → прогноз → web-інтерфейс. На першому етапі користувач працює з CSV-файлом, який містить історичні записи продажів. На другому етапі система перевіряє коректність і структуру даних. На третьому етапі дані приводяться до формату, придатного для аналізу. На четвертому етапі створюються ознаки, що описують календарний контекст, попередні продажі, середні значення, ціну, категорію та акційні періоди. На п’ятому етапі модель машинного навчання використовується для оцінки майбутнього попиту. На завершальному етапі результат подається користувачу через web-інтерфейс.

Система має п’ять основних сторінок: «Огляд», «Дані», «Прогноз», «Модель» і «Про систему». Така структура дозволяє послідовно показати основні частини роботи: загальну ідею, вхідні дані, прогнозування, якість моделі та опис архітектури. Вона також відповідає логіці пояснення системи під час захисту кваліфікаційної роботи.

Сторінка «Огляд» виконує роль головної аналітичної панелі. На ній користувач бачить назву системи, коротке пояснення її призначення, ключові показники набору даних, практичне використання прогнозу, демонстраційний графік і коротку інтерпретацію прогнозу. Ця сторінка дає загальне уявлення

про те, що система аналізує історичні продажі, будує прогноз і допомагає оцінити майбутній попит.

Сторінка «Дані» відповідає за представлення структури вхідного набору даних. На ній показано історичний період, кількість товарів, кількість категорій, кількість записів продажів і середній денний попит. Також на сторінці є блок структури даних, імпорт CSV і таблиця товарів із базовою статистикою. Таблиця містить товар, категорію, середню ціну, середній денний попит, мінімальні та максимальні продажі, кількість записів, історичний період і кнопку для переходу до прогнозу.

Сторінка «Прогноз» є основною сторінкою для демонстрації результату машинного навчання. На ній користувач обирає товар і горизонт прогнозування: 7, 14 або 30 днів. Графік показує історичні продажі та майбутній прогноз. Важливо, що прогнозні значення починаються після останньої історичної дати, тобто не накладаються на фактичні продажі. Крім графіка, сторінка містить короткий висновок, інтерпретацію динаміки, фактори впливу, пояснення графіка та таблицю прогнозу по днях.

Сторінка «Модель» призначена для оцінки якості прогнозування. На ній показано, як оцінювалась модель, чому обрано ExtraTreesRegressor, що означають метрики MAE, RMSE і MAPE, які значення метрик отримано, які моделі порівнювались і які фактори найбільше впливають на прогноз. Ця сторінка важлива для обґрунтування того, що система не просто виводить випадкове число, а використовує модель, якість якої перевірено на тестовій вибірці.

Сторінка «Про систему» має пояснювальний характер. Вона містить мету роботи, об’єкт і предмет дослідження, опис вхідних даних, алгоритм роботи, архітектуру, відповідність кваліфікаційній роботі, використані технології, опис SQLite, питання санкціонованого доступу, практичну цінність і автора. Саме ця сторінка допомагає пов’язати програмну реалізацію з вимогами кваліфікаційної роботи.

Реально система виконує такі функції: приймає CSV-дані з історичними продажами, показує структуру набору даних, формує ознаки для моделі, використовує навчену ML-модель для прогнозування попиту, показує графік історії та прогнозу, надає текстову інтерпретацію, відображає метрики якості та дозволяє переглянути опис системи. Також у проєкті реалізовано API endpoints і автоматичні тести, що підтверджує працездатність основних частин застосунку.

Водночас важливо чітко визначити межі системи. Retail Demand Sense не є ERP-системою, POS-системою або повноцінною системою управління складом. Вона не керує автоматично закупівлями, не змінює ціни, не запускає маркетингові кампанії та не оптимізує логістичні маршрути. У системі можуть бути дані, пов’язані з ціною, акціями, залишками або затримками постачальника, але вони використовуються як аналітичні ознаки або контекст для прогнозування, а не як механізм автоматичного управління бізнес-процесами.

Таким чином, прогноз у системі використовується як інструмент підтримки управлінських рішень. Він допомагає оцінити майбутній попит, побачити зміну динаміки продажів, проаналізувати фактори впливу та отримати кількісну основу для подальших дій менеджера. Саме така постановка відповідає темі роботи: система є інтелектуальним програмним прототипом для прогнозування попиту та автоматизації аналітичної частини процесів ритейлу.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ПРОГНОЗУВАННЯ ПОПИТУ

2.1. Загальні вимоги до системи

Вимоги до системи сформовано з урахуванням підходів програмної інженерії та пріоритезації вимог [3; 4; 5; 39].

Проєктування інтелектуальної системи прогнозування попиту починається з визначення її призначення, меж використання та вимог до основних функцій. Розроблена система Retail Demand Sense призначена для аналізу історичних продажів товарів ритейлу, підготовки даних до машинного навчання, побудови прогнозу майбутнього попиту та відображення результатів у web-інтерфейсі. Основна ідея системи полягає не в заміні управлінських рішень, а в наданні користувачу структурованої аналітичної інформації, яка може бути використана під час планування роздрібних операцій.

Користувачем системи є особа, яка аналізує продажі товарів і потребує зрозумілого інструмента для оцінки майбутнього попиту. У межах дипломного проєкту таким користувачем можна вважати менеджера ритейлу, аналітика, студента або викладача, який перевіряє роботу системи під час захисту. Користувач не повинен взаємодіяти з Python-скриптами безпосередньо. Для нього важливо мати web-інтерфейс, у якому можна переглянути набір даних, вибрати товар, побудувати прогноз, оцінити графік і зрозуміти, які фактори вплинули на результат.

Система має виконувати кілька основних задач. По-перше, вона повинна приймати CSV-файл з історичними записами продажів. По-друге, вона має перевіряти структуру файлу, очищати та нормалізувати дані, щоб вони були придатними для подальшого аналізу. По-третє, система повинна формувати ознаки для машинного навчання, зокрема календарні характеристики, попередні продажі та ковзні середні. По-четверте, вона має використовувати навчену модель для побудови прогнозу попиту на вибраний період. По-п’яте,

результати прогнозу мають бути представлені у зрозумілій формі: графік, короткий висновок, фактори впливу та таблиця прогнозних значень.

До функціональних вимог системи належить можливість імпорту CSV-файлу з історичними продажами. Система повинна перевіряти наявність обов'язкових колонок, приводити числові поля до коректного формату, обробляти дату та відкидати записи, які не можуть бути використані для аналізу. Після завантаження даних користувач повинен мати можливість переглянути структуру набору даних: товари, категорії, середні ціни, середній денний попит, мінімальні та максимальні продажі, кількість записів і історичний період. Ця функція важлива, оскільки прогнозування не може бути відокремлене від якості та структури вхідних даних.

Окремою функціональною вимогою є побудова прогнозу для вибраного товару. Користувач повинен мати можливість вибрати товар зі списку і задати горизонт прогнозування. У системі передбачено горизонти 7, 14 і 30 днів, що дозволяє переглядати короткостроковий прогноз у кількох варіантах. Після вибору параметрів система формує прогнозні значення на майбутні дати, які починаються після останньої історичної дати продажів. Це є важливим з погляду коректної візуалізації, оскільки прогноз не повинен накладатися на фактичні дані.

Система також повинна відображати графік фактичних і прогнозних значень. Фактичні продажі мають бути відокремлені від прогнозу візуально, щоб користувач міг швидко зрозуміти, де закінчується історія і де починається прогноз. Крім графіка, користувач повинен бачити короткий висновок: сумарний прогнозований попит, середній прогнозований попит на день, тип динаміки та період прогнозу. Така інтерпретація потрібна для того, щоб результат не залишався лише набором чисел.

Ще однією функціональною вимогою є перегляд метрик якості моделі. Система повинна показувати MAE, RMSE і MAPE, оскільки ці метрики дозволяють оцінити, наскільки прогноз відхиляється від фактичних значень на тестовій вибірці. Також користувач повинен бачити інформацію про

порівняння моделей і про те, чому обрано конкретний алгоритм. У поточній реалізації основною моделлю є ExtraTreesRegressor, оскільки вона показала найнижче значення MAPE серед протестованих алгоритмів.

До функціональних вимог належить і наявність сторінки опису системи. Вона потрібна для пояснення мети роботи, об'єкта і предмета дослідження, вхідних даних, архітектури, бази даних, практичної цінності та меж використання. Для дипломного проєкту така сторінка має особливе значення, оскільки вона пов'язує програмну реалізацію з темою кваліфікаційної роботи та допомагає пояснити логіку системи під час захисту.

Нефункціональні вимоги визначають якість роботи системи та зручність її використання. Насамперед web-інтерфейс має бути зрозумілим і не перевантаженим зайвими бізнес-функціями. Оскільки головна тема роботи - прогнозування попиту, інтерфейс не повинен виглядати як складська CRM або система закупівель. Основний акцент має бути на історичних даних, прогнозі, метриках моделі та поясненні факторів впливу. Тому в системі використовується стриманий аналітичний стиль, окремі сторінки для даних, прогнозу та моделі, а також короткі пояснення біля основних результатів.

Система повинна запускатися локально, оскільки це спрощує демонстрацію під час захисту та не потребує окремої серверної інфраструктури. Для локального збереження даних використовується SQLite. Такий вибір відповідає масштабу навчального прототипу: база даних не потребує окремого сервера, легко створюється разом із застосунком і достатня для збереження товарів, продажів, історії імпортів, прогнозів і метрик. Стабільність основних маршрутів також є важливою вимогою, оскільки користувач повинен мати можливість відкрити сторінки «Огляд», «Дані», «Прогноз», «Модель», «Про систему» та службовий маршрут перевірки стану.

Вимоги до зручності інтерфейсу полягають у тому, що користувач повинен швидко зрозуміти призначення системи. На головній сторінці мають бути показані основні KPI, приклад прогнозу та коротке пояснення практичного використання результату. На сторінці даних потрібно чітко

показати структуру CSV і базову статистику товарів. На сторінці прогнозу потрібно забезпечити зрозумілий вибір товару та горизонту прогнозування. На сторінці моделі потрібно пояснити метрики простою мовою, без надмірної технічної деталізації в основному інтерфейсі.

Вимоги до надійності пов’язані з перевіркою структури даних, обробкою помилок і тестуванням. Якщо CSV-файл не містить обов’язкових колонок, система повинна повідомити про помилку, а не працювати з некоректним набором даних. Якщо модель не навчена або даних недостатньо, користувач повинен отримати зрозуміле повідомлення. У проєкті також передбачено автоматичні тести, які перевіряють працездатність основних частин системи. Для технічної перевірки існує маршрут /health, який повертає інформацію про стан застосунку, наявність моделі, метрик, даних і бази.

Система має певні обмеження. Вона не є промисловою ERP-, POS-або складською системою. Вона не виконує автоматичного замовлення товарів, не змінює ціни, не запускає маркетингові кампанії та не оптимізує логістику. Дані про залишки, акції, свята або затримку постачальника використовуються як допоміжний контекст для аналізу і формування ознак, а не як основа автоматичного управління бізнес-процесами. Таке обмеження є важливим, оскільки система позиціонується саме як інтелектуальний прототип прогнозування попиту та підтримки прийняття рішень.

Загальну схему роботи системи Retail Demand Sense наведено на рис. 2.1. Система послідовно виконує імпорт CSV-даних, їх перевірку та очищення, збереження у SQLite, формування ознак, навчання моделі, побудову прогнозу, оцінювання якості та відображення результатів у web-інтерфейсі.

Рис. 2.1. Загальна схема роботи системи Retail Demand Sense

2.2. Структура вхідних даних

Вхідні дані є основою роботи системи прогнозування попиту. У розробленому проєкті дані подаються у вигляді CSV-файлу з історичними записами продажів. Такий формат є зручним для дипломного прототипу, оскільки CSV легко створювати, переглядати, експортувати з табличних редакторів і обробляти за допомогою Python-бібліотек. У системі CSV-файл використовується як джерело інформації для імпорту, попередньої обробки, формування ознак і подальшого прогнозування.

Структура CSV визначена таким чином, щоб один рядок описував продаж певного товару за конкретну дату. Основними полями є date, product_id, product_name, category, sales_quantity, price, stock_quantity, promo, holiday і supplier_delay_days. Також у наборі можуть використовуватися допоміжні поля product_icon, base_demand і seasonality_type. Частина цих полів є обов’язковою для роботи системи, а частина доповнює дані

контекстом, який може бути корисним для аналізу або відображення в інтерфейсі.

Поле date містить дату продажу. Воно потрібне для впорядкування записів у часі, побудови історії продажів, визначення останньої доступної дати та формування майбутніх дат прогнозу. Без дати неможливо коректно побудувати часову послідовність, розрахувати лагові ознаки або визначити календарні характеристики. У процесі обробки система перетворює це поле у формат дати, після чого може отримати день тижня, місяць, день місяця, квартал і ознаку вихідного дня.

Поля product_id і product_name ідентифікують товар. Ідентифікатор потрібний для однозначного групування записів, оскільки назва товару теоретично може змінюватися або містити варіації написання. Саме за product_id система групує історичні продажі, розраховує лаги та ковзні середні для конкретного товару. Назва товару потрібна для відображення в інтерфейсі, щоб користувач працював не з технічним номером, а зі зрозумілою товарною позицією.

Поле category описує категорію товару. Категорія важлива для прогнозування, оскільки різні групи товарів можуть мати різну поведінку попиту. Наприклад, напої, молочні продукти, бакалія або кондитерські вироби можуть відрізнятися за сезонністю, середнім рівнем продажів і реакцією на календарні фактори. У системі категорія перетворюється на числову ознаку, що дозволяє моделі враховувати належність товару до певної групи.

Поле sales_quantity містить кількість проданих одиниць товару. Саме це поле є цільовою змінною для задачі машинного навчання. Під час навчання модель отримує ознаки, сформовані на основі історичних даних, і навчається передбачати значення sales_quantity. Під час прогнозування система оцінює майбутні значення цього показника для вибраного товару та горизонту прогнозу. Тому якість і повнота цього поля мають найбільше значення для результату.

Поле price містить ціну товару. Ціна може впливати на попит, оскільки зміна вартості часто змінює поведінку покупців. У межах розробленої системи ціна використовується як одна з ознак, що описує умови продажу. Також на основі історії цін може формуватися допоміжна ознака зміни ціни. Водночас система не виконує автоматичного ціноутворення і не рекомендує зміну ціни. Вона лише враховує ціну як один із факторів, який може бути пов'язаний із динамікою продажів.

Поле stock_quantity описує залишок товару. У контексті цієї роботи воно не використовується як механізм автоматичного управління складом. Його доцільно розглядати як допоміжну характеристику, що може пояснювати певні особливості продажів. Наприклад, низький залишок може обмежити фактичні продажі, а високий залишок може свідчити про доступність товару. У системі це поле може використовуватися як додаткова ознака, однак прогноз залишається інструментом аналізу попиту, а не автоматичного управління запасами.

Поля promo і holiday описують акційний період і святковий день. Вони важливі, оскільки попит у ритейлі часто змінюється під впливом промоакцій, знижок, свят або календарних подій. Ознака promo дозволяє врахувати, що в певні дні продажі могли бути вищими через акційні умови. Ознака holiday допомагає відобразити можливий вплив святкового періоду. У моделі ці поля використовуються як пояснювальні ознаки, а не як окремий модуль маркетингового планування.

Поле supplier_delay_days описує затримку постачальника в днях. У межах системи це поле також має допоміжний характер. Воно може бути корисним для аналізу контексту продажів, але система не оптимізує логістику та не керує постачальниками. Наявність такого поля дозволяє показати, що вхідний набір даних може містити не лише чисті продажі, а й додаткові характеристики, які потенційно впливають на доступність товару або інтерпретацію попиту.

Поле product_icon використовується переважно для візуального представлення товару або категорії в інтерфейсі. Воно не є ключовим для машинного навчання і не визначає прогноз. Поле base_demand описує базовий рівень попиту для товару або категорії. Воно може бути корисним як додаткова ознака, що задає орієнтовний рівень продажів. Поле seasonality_type описує тип сезонності або характер попиту, наприклад стабільний, літній, зимовий або святковий. Воно допомагає інтерпретувати динаміку, однак не повинно сприйматися як абсолютне правило майбутніх продажів.

Під час завантаження CSV система перевіряє обов'язкові колонки, приводить числові поля до числового формату, перетворює дату, заповнює відсутні допоміжні поля стандартними значеннями та сортує записи за товаром і датою. Некоректні записи, які не можуть бути використані для аналізу, відкидаються. Така підготовка потрібна для того, щоб модель отримувала узгоджений набір даних, а користувач міг переглядати в інтерфейсі лише очищену та структуровану інформацію.

Отже, структура вхідних даних побудована так, щоб підтримати основні етапи прогнозування попиту. Вона містить як фактичні продажі, так і фактори, які можуть впливати на попит. При цьому допоміжні поля не перетворюють систему на складську, логістичну або маркетингову платформу. Вони використовуються для аналітичного контексту та формування ознак, що відповідає меті кваліфікаційної роботи.

Фрагмент структури CSV-файлу, який використовується для імпорту історичних продажів, наведено на рис. 2.2.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	date	product_id	product_name	category	sale	price	stock	prom	holid	supp	proi	bas	season	shelf	supplier_name	region
2	01.05.2024	1	Рис довгозернистий 1 кг	Бакалія	23	68.41	251	0	0	0		24	stable	365	AgroTrade LLC	Київ
3	02.05.2024	1	Рис довгозернистий 1 кг	Бакалія	26	68.71	225	0	0	1		24	stable	365	AgroTrade LLC	Київ
4	03.05.2024	1	Рис довгозернистий 1 кг	Бакалія	22	68.51	203	0	0	0		24	stable	365	AgroTrade LLC	Київ
5	04.05.2024	1	Рис довгозернистий 1 кг	Бакалія	22	69.4	181	0	0	1		24	stable	365	AgroTrade LLC	Київ
6	05.05.2024	1	Рис довгозернистий 1 кг	Бакалія	24	68.53	157	0	0	1		24	stable	365	AgroTrade LLC	Київ
7	06.05.2024	1	Рис довгозернистий 1 кг	Бакалія	25	68.89	132	0	0	0		24	stable	365	AgroTrade LLC	Київ
8	07.05.2024	1	Рис довгозернистий 1 кг	Бакалія	22	68.61	330	0	0	0		24	stable	365	AgroTrade LLC	Київ
9	08.05.2024	1	Рис довгозернистий 1 кг	Бакалія	21	68.1	309	0	0	1		24	stable	365	AgroTrade LLC	Київ
10	09.05.2024	1	Рис довгозернистий 1 кг	Бакалія	23	68.22	286	0	0	0		24	stable	365	AgroTrade LLC	Київ
11	10.05.2024	1	Рис довгозернистий 1 кг	Бакалія	22	69.1	264	0	0	0		24	stable	365	AgroTrade LLC	Київ
12	11.05.2024	1	Рис довгозернистий 1 кг	Бакалія	25	69.41	239	0	0	0		24	stable	365	AgroTrade LLC	Київ
13	12.05.2024	1	Рис довгозернистий 1 кг	Бакалія	28	68.13	211	0	0	0		24	stable	365	AgroTrade LLC	Київ
14	13.05.2024	1	Рис довгозернистий 1 кг	Бакалія	25	68.11	186	0	0	1		24	stable	365	AgroTrade LLC	Київ
15	14.05.2024	1	Рис довгозернистий 1 кг	Бакалія	25	69.32	351	0	0	0		24	stable	365	AgroTrade LLC	Київ
16	15.05.2024	1	Рис довгозернистий 1 кг	Бакалія	23	68.85	328	0	0	1		24	stable	365	AgroTrade LLC	Київ
17	16.05.2024	1	Рис довгозернистий 1 кг	Бакалія	28	68.28	300	0	0	0		24	stable	365	AgroTrade LLC	Київ
18	17.05.2024	1	Рис довгозернистий 1 кг	Бакалія	22	69.27	278	0	0	1		24	stable	365	AgroTrade LLC	Київ
19	18.05.2024	1	Рис довгозернистий 1 кг	Бакалія	25	69.02	253	0	0	0		24	stable	365	AgroTrade LLC	Київ
20	19.05.2024	1	Рис довгозернистий 1 кг	Бакалія	21	68.86	232	0	0	0		24	stable	365	AgroTrade LLC	Київ
21	20.05.2024	1	Рис довгозернистий 1 кг	Бакалія	22	68.58	210	0	0	0		24	stable	365	AgroTrade LLC	Київ
22	21.05.2024	1	Рис довгозернистий 1 кг	Бакалія	23	69.22	413	0	0	0		24	stable	365	AgroTrade LLC	Київ
23	22.05.2024	1	Рис довгозернистий 1 кг	Бакалія	21	68.39	392	0	0	1		24	stable	365	AgroTrade LLC	Київ
24	23.05.2024	1	Рис довгозернистий 1 кг	Бакалія	21	68.93	371	0	0	1		24	stable	365	AgroTrade LLC	Київ
25	24.05.2024	1	Рис довгозернистий 1 кг	Бакалія	19	68.39	352	0	0	0		24	stable	365	AgroTrade LLC	Київ
26	25.05.2024	1	Рис довгозернистий 1 кг	Бакалія	23	69.35	329	0	0	0		24	stable	365	AgroTrade LLC	Київ
27	26.05.2024	1	Рис довгозернистий 1 кг	Бакалія	25	68.27	304	0	0	0		24	stable	365	AgroTrade LLC	Київ
28	27.05.2024	1	Рис довгозернистий 1 кг	Бакалія	25	68.86	279	0	0	0		24	stable	365	AgroTrade LLC	Київ

Рис. 2.2. Приклад структури вхідного CSV-файлу

2.3. Проєктування бази даних SQLite

Використання SQLite як локальної бази даних відповідає завданню навчального прототипу та підтверджується документацією SQLite [7; 17].

Для локального збереження даних у системі Retail Demand Sense використовується SQLite. Вибір SQLite є доцільним для навчального прототипу, оскільки ця база даних не потребує окремого серверного процесу, зберігає дані в одному файлі, легко запускається локально разом із Flask-застосунком і достатня для демонстрації структури даних, імпорту, прогнозів і метрик. У межах кваліфікаційної роботи важливо показати не промислову масштабовану інфраструктуру, а коректну організацію збереження даних і відокремлення бази від інтерфейсу та ML-логіки.

У реальному коді робота з базою даних реалізована в модулі src/database.py. У цьому файлі визначено шлях до бази retail_demand.db, SQL-схему та функції для створення таблиць, імпорту даних, збереження прогнозів і метрик. Під час запуску застосунку Flask викликає ініціалізацію бази даних, унаслідок чого створюються необхідні таблиці, якщо їх ще немає. Такий підхід спрощує локальний запуск і робить систему придатною для демонстрації без ручного налаштування СУБД.

У базі даних передбачено таблицю products, яка зберігає інформацію про товари. Вона містить ідентифікатор товару, назву, категорію, ціну, залишок, базовий попит, тип сезонності та допоміжні характеристики. Роль цієї таблиці полягає в тому, щоб зберігати каталог товарів, отриманий із CSV-набору. Водночас каталог у системі не використовується як складський модуль. Він потрібний для ідентифікації товарів, відображення їх в інтерфейсі та зв’язку з історичними продажами.

Таблиця sales зберігає історичні записи продажів. Для кожного запису фіксуються товар, дата, кількість продажів, ціна, залишок, ознака акції, ознака святкового дня та затримка постачальника. Саме ця таблиця є основним джерелом історії, на якій базується аналіз попиту. Продажі пов’язані з товарами через зовнішній ключ product_id, що дозволяє зберігати дані у структурованому вигляді та уникати дублювання опису товару в кожному записі.

Таблиця imports використовується для журналювання імпортів CSV-файлів. У ній зберігається назва файлу, кількість рядків, кількість товарів, дата імпорту, статус і повідомлення про помилку, якщо імпорт не був успішним. Така таблиця потрібна для відстеження того, які дані були завантажені в систему. Для дипломного прототипу це також демонструє, що імпорт є окремою операцією, результат якої можна перевірити.

Таблиця forecasts призначена для збереження прогнозних значень. Вона містить товар, дату прогнозу, прогнозовану кількість і час створення запису. У поточному web-інтерфейсі прогноз може будуватися без обов’язкового

збереження кожного перегляду, щоб сторінки відкривалися швидко і не накопичували зайві записи. Однак наявність таблиці прогнозів показує, що структура бази передбачає можливість збереження результатів прогнозування, якщо це потрібно для подальшого аналізу.

Таблиця model_metrics зберігає інформацію про якість моделі: назву моделі, MAE, RMSE, MAPE, дату створення запису та сирі метрики у JSON-форматі. Ця таблиця потрібна для фіксації результатів навчання моделі. Вона пов’язує ML-частину системи з базою даних і дозволяє показати, що результати навчання не залишаються лише в пам’яті програми. Окрім цього, метрики зберігаються у файлі models/metrics.json, який використовується для швидкого відображення даних на сторінці «Модель».

У схемі також є таблиця users, яка демонструє можливість подальшого впровадження санкціонованого доступу. У поточній демонстраційній версії основний інтерфейс відкритий для зручності захисту, однак на рівні структури даних і коду передбачено можливість зберігати користувачів, хеші паролів і ролі. Це не означає, що авторизація є центральною функцією поточної версії, але показує напрям розширення системи відповідно до вимог захищеного доступу.

У схемі також присутня таблиця recommendations, пов’язана з історичними етапами розробки. У межах поточного позиціонування системи закупівлі не є головним фокусом, а інтерфейс зосереджено на прогнозуванні попиту. Тому при описі дипломної роботи доцільно акцентувати увагу на таблицях товарів, продажів, імпортів, прогнозів і метрик, не подаючи систему як повноцінний модуль закупівель.

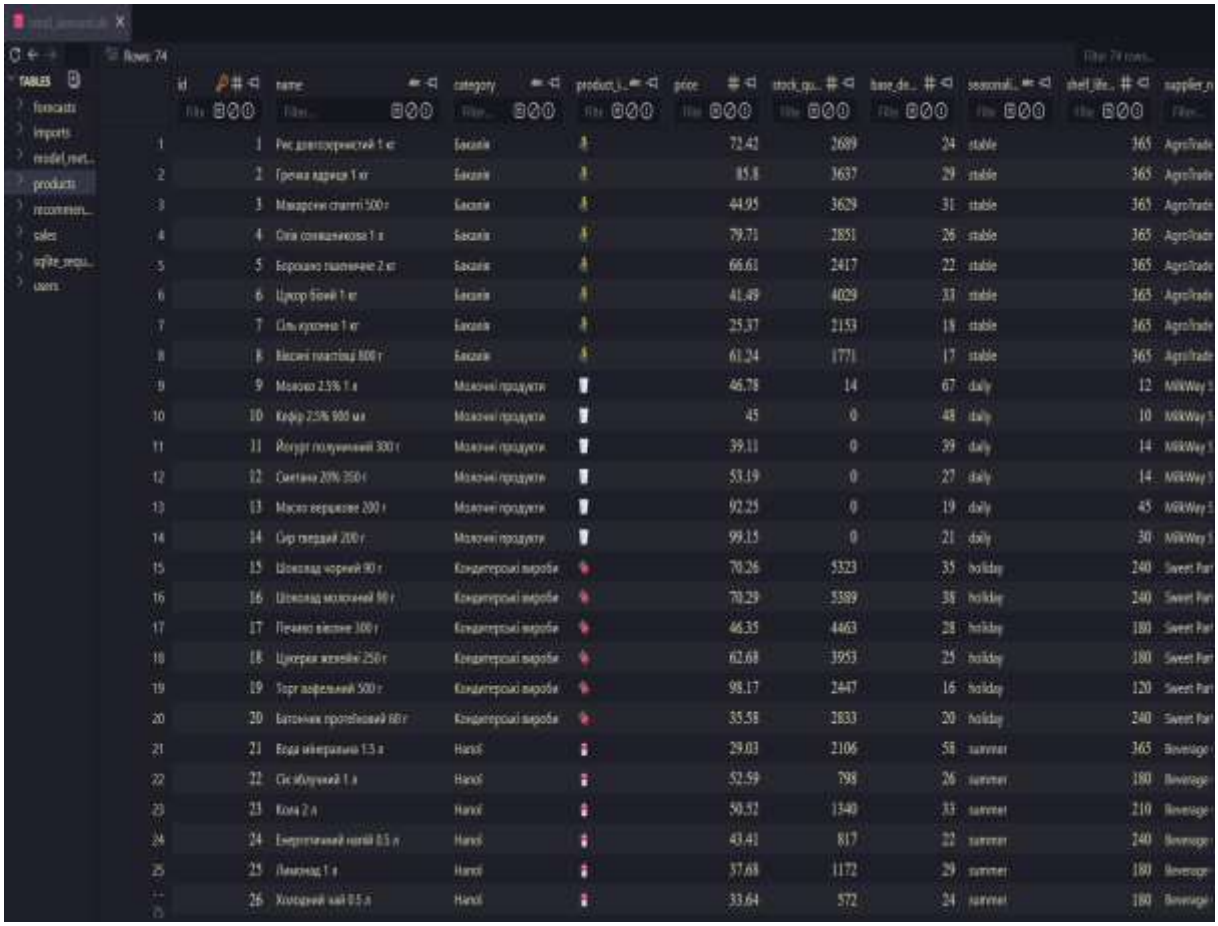
База даних допомагає відокремити зберігання даних від інших частин системи. Користувацький інтерфейс не повинен самостійно працювати з CSV як із єдиним джерелом інформації, а ML-логіка не повинна відповідати за структуру web-сторінок. SQLite виконує роль локального сховища, у якому дані можуть бути структуровані, пов’язані між собою та використані різними модулями. Такий підхід відповідає принципам програмної інженерії, оскільки

розділяє відповідальність між рівнем даних, рівнем обробки, ML-модулем і рівнем представлення.

ER-діаграму бази даних SQLite системи Retail Demand Sense наведено на рис. 2.3. Центральною таблицею є products, з якою пов’язані таблиці sales, forecasts і recommendations. Окремі службові таблиці imports, users і model_metrics використовуються для журналювання імпорту, можливого санкціонованого доступу та збереження результатів оцінювання моделей.

Рис. 2.3. ER-діаграма бази даних SQLite системи Retail Demand Sense

Приклад таблиці products з базовими характеристиками товарів наведено на рис. 2.4.



id	name	category	product_t...	price	stock_qu...	base_de...	seasonali...	shelf_life...	supplier_n
1	1	Рис довгозернистий 1 кг	Бакалія		72.42	2689	24	stable	365	AgroTrade
2	2	Гречка ядриця 1 кг	Бакалія		85.8	3637	29	stable	365	AgroTrade
3	3	Макаронні спагеті 500 г	Бакалія		44.95	3629	31	stable	365	AgroTrade
4	4	Олія соняшникова 1 л	Бакалія		79.71	2851	26	stable	365	AgroTrade
5	5	Борошно пшеничне 2 кг	Бакалія		66.61	2417	22	stable	365	AgroTrade
6	6	Цукор білий 1 кг	Бакалія		41.49	4029	33	stable	365	AgroTrade
7	7	Сіль кухонна 1 кг	Бакалія		25.37	2153	18	stable	365	AgroTrade
8	8	Вівсяні пластівці 800 г	Бакалія		61.24	1771	17	stable	365	AgroTrade
9	9	Молоко 2.5% 1 л	Молочні продукти		46.78	14	67	daily	12	MilkWay
10	10	Кефір 2.5% 900 мл	Молочні продукти		45	0	48	daily	10	MilkWay
11	11	Йогурт полуничний 300 г	Молочні продукти		39.11	0	39	daily	14	MilkWay
12	12	Сметана 20% 350 г	Молочні продукти		53.19	0	27	daily	14	MilkWay
13	13	Масло вершкове 200 г	Молочні продукти		92.25	0	19	daily	45	MilkWay
14	14	Сир твердий 200 г	Молочні продукти		99.15	0	21	daily	30	MilkWay
15	15	Шоколад чорний 90 г	Кондитерські вироби		70.26	5323	35	holiday	240	Sweet Par
16	16	Шоколад молочний 90 г	Кондитерські вироби		70.29	5389	38	holiday	240	Sweet Par
17	17	Печиво вівсяне 300 г	Кондитерські вироби		46.35	4463	28	holiday	180	Sweet Par
18	18	Цукерки асорті 250 г	Кондитерські вироби		62.68	3953	25	holiday	180	Sweet Par
19	19	Торт вафельний 500 г	Кондитерські вироби		98.17	2447	16	holiday	120	Sweet Par
20	20	Батончик протеїновий 60 г	Кондитерські вироби		35.58	2833	20	holiday	240	Sweet Par
21	21	Вода мінеральна 1.5 л	Напої		29.03	2106	58	summer	365	Beverage
22	22	Сік яблучний 1 л	Напої		52.59	798	26	summer	180	Beverage
23	23	Кола 2 л	Напої		50.52	1340	33	summer	210	Beverage
24	24	Енергетичний напій 0.5 л	Напої		43.41	817	22	summer	240	Beverage
25	25	Лимонад 1 л	Напої		57.68	1172	29	summer	180	Beverage
26	26	Холодний чай 0.5 л	Напої		33.64	572	24	summer	180	Beverage

Рис. 2.4. Таблиця products з базовими характеристиками товарів

Приклад таблиці sales з історичними записами продажів наведено на рис. 2.5.

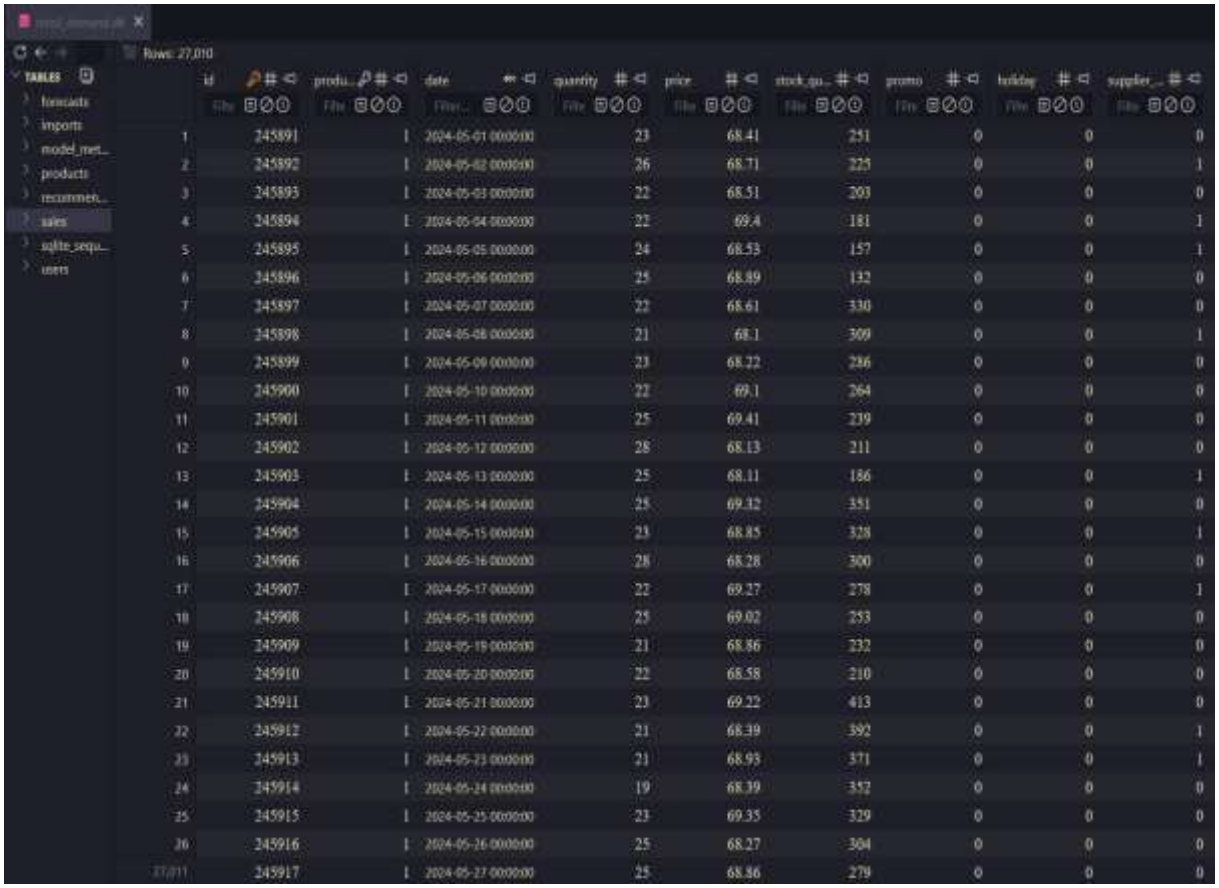


Рис. 2.5. Таблиця sales з історичними записами продажів

2.4. Архітектура системи

Архітектура системи Retail Demand Sense побудована як кілька логічних рівнів, кожен із яких виконує окрему роль. Такий підхід робить систему зрозумілішою, спрощує супровід і дозволяє пояснити роботу проєкту під час захисту. Основними рівнями є рівень даних, рівень обробки даних, рівень машинного навчання, рівень API та рівень web-інтерфейсу.

Рівень даних містить CSV-файл, SQLite-базу даних, файл навченої моделі та файл метрик. CSV-файл у папці data є вхідним джерелом історичних продажів. SQLite-база retail_demand.db використовується для збереження товарів, продажів, імпортів, прогнозів і метрик. Файл models/demand_model.joblib зберігає навчений артефакт моделі, а models/metrics.json містить метрики якості, порівняння моделей і важливість

ознак. Такий розподіл дозволяє зберігати вихідні дані, структуровану базу і результати навчання окремо.

Рівень обробки даних реалізовано насамперед у модулі src/data_processing.py. Він відповідає за завантаження CSV, перевірку обов'язкових колонок, нормалізацію застарілих назв полів, перетворення дати, приведення числових колонок до потрібного типу, заповнення допоміжних полів стандартними значеннями та сортування записів. Цей рівень є проміжним між сирими даними та моделлю машинного навчання. Його завдання полягає в тому, щоб не допустити передачу некоректних або неповних даних у наступні етапи.

Рівень формування ознак реалізовано в src/feature_engineering.py. Він створює набір ознак, потрібних для навчання та прогнозування. До них належать календарні ознаки, категорія товару в числовому вигляді, лагові ознаки продажів, ковзні середні, стандартне відхилення за короткий період, зміна ціни, ознаки акційного періоду, співвідношення залишку до базового попиту та короткостроковий тренд. Саме цей рівень перетворює історичні записи продажів на навчальний набір для регресійної моделі.

Рівень машинного навчання складається з модулів src/train_model.py і src/predict.py. Модуль навчання відповідає за підготовку навчальної та тестової вибірок, навчання кількох моделей, обчислення MAE, RMSE і MAPE, вибір найкращої моделі за MAPE, збереження артефакту моделі та метрик. Модуль прогнозування завантажує навчений артефакт, отримує історію вибраного товару, формує майбутні рядки ознак і покроково прогнозує значення попиту на заданий період.

Рівень API і маршрутизації реалізовано у Flask-застосунку app/app.py. Він створює web-застосунок, ініціалізує базу даних, обробляє маршрути сторінок, приймає CSV-файли, викликає модулі імпорту, навчання та прогнозування, а також повертає JSON-відповіді для API. Серед службових маршрутів є /health, який дозволяє перевірити стан застосунку, наявність

моделі, метрик, даних і бази. API-маршрути надають дані про товари, прогноз, метрики, важливість ознак, історію імпортів і дані для графіків.

Рівень web-інтерфейсу складається з Jinja2-шаблонів у app/templates, CSS-файлу app/static/css/styles.css і JavaScript-файлу app/static/js/app.js. Шаблони формують сторінки «Огляд», «Дані», «Прогноз», «Модель» і «Про систему». CSS відповідає за візуальне оформлення, адаптивність, відступи, таблиці, картки та навігацію. JavaScript використовується для побудови графіків на Canvas і взаємодії з API. Важливо, що frontend не виконує машинне навчання самостійно, а лише відображає підготовлені backend-дані.

Повний шлях даних у системі можна описати так: CSV-дані надходять у застосунок через файл або вбудований приклад, проходять обробку і перевірку, після чого з них формуються ознаки. На основі цих ознак модель навчається або використовується для прогнозу. Результат прогнозування передається через Flask-маршрути та API до HTML/CSS/JS-рівня представлення, де користувач бачить графік, висновок, фактори впливу і таблицю прогнозу. Такий шлях відповідає логіці інтелектуальної системи: дані перетворюються на ознаки, ознаки використовуються моделлю, а результат подається користувачу в зрозумілому вигляді.

Архітектура системи є достатньо простою для локальної кваліфікаційної роботи, але водночас демонструє базові принципи програмної інженерії. Логіка роботи з даними винесена в окремі модулі src, web-маршрути зосереджені в Flask-застосунку, шаблони відповідають за представлення, а модель і метрики зберігаються окремо. Такий підхід полегшує тестування, пояснення структури проєкту та можливе майбутнє розширення.

2.5. Алгоритм роботи системи

Алгоритм роботи системи Retail Demand Sense описує послідовність дій від запуску застосунку до отримання прогнозу та перегляду метрик. На першому етапі запускається Flask-застосунок командою ру app/app.ру. Під час запуску створюється об’єкт застосунку, ініціалізується база даних SQLite,

перевіряється наявність CSV-файлу та моделі. Якщо приклад набору даних або модель відсутні, код передбачає їх створення або навчання на старті застосунку. Після цього сервер відкривається локально і стає доступним через браузер.

На другому етапі користувач відкриває web-інтерфейс. Головна сторінка «Огляд» показує призначення системи, ключові показники набору даних, практичне використання прогнозу, демонстраційний графік і коротку інтерпретацію. Це дозволяє швидко зрозуміти, що система працює з історичними продажами, будує прогноз попиту і надає користувачу аналітичний результат. Головна сторінка не вимагає обов'язкової авторизації, що спрощує демонстрацію під час захисту.

На третьому етапі користувач може перейти на сторінку «Дані» і переглянути структуру вхідного набору. Якщо потрібно, він може завантажити CSV-файл з історичними продажами або використати демонстраційний приклад. Після завантаження система перевіряє, чи містить файл обов'язкові поля, зокрема дату, ідентифікатор товару, назву, категорію, кількість продажів, ціну, залишок, ознаки акції, святкового дня та затримки постачальника. Якщо структура файлу неправильна, система повертає повідомлення про помилку.

На четвертому етапі виконується очищення і нормалізація даних. Цей процес включає перетворення дати у коректний формат, приведення числових колонок до числових типів, обробку відсутніх допоміжних полів, обмеження некоректних від'ємних значень і сортування записів за товаром та датою. Такі дії потрібні для того, щоб подальші етапи працювали з узгодженим набором даних. Після підготовки дані можуть бути імпортовані в SQLite, де зберігаються товари, продажі та інформація про імпорт.

На п'ятому етапі формується набір ознак для машинного навчання. Історичні продажі не передаються в модель лише як сирі рядки CSV. На їх основі створюються календарні ознаки, кодується категорія товару, розраховуються лаги продажів, ковзні середні за 7 і 14 днів, короткостроковий

тренд, зміна ціни та інші характеристики. Цей етап є одним із ключових, оскільки від якості ознак залежить здатність моделі знаходити закономірності в даних.

На шостому етапі виконується навчання і порівняння моделей. Підготовлений набір даних поділяється на навчальну та тестову вибірки. Навчальна вибірка використовується для побудови моделей, а тестова - для перевірки якості прогнозування. У системі порівнюються RandomForestRegressor, GradientBoostingRegressor і ExtraTreesRegressor. Для кожної моделі обчислюються MAE, RMSE і MAPE. Це дозволяє не обирати алгоритм довільно, а спиратися на кількісні результати.

На сьомому етапі система обирає модель за значенням MAPE. MAPE показує середню відносну похибку прогнозу у відсотках, тому є зручною метрикою для порівняння моделей. У поточній реалізації найкраще значення MAPE показала модель ExtraTreesRegressor, тому вона зберігається як основна. Разом із моделлю зберігаються список ознак, мапінг категорій і метрики. Ці результати використовуються під час побудови прогнозу та на сторінці оцінки якості моделі.

На восьмому етапі користувач будує прогноз для вибраного товару. На сторінці «Прогноз» він обирає товар і горизонт прогнозування. Система бере останні історичні продажі цього товару, визначає останню доступну дату, формує майбутні дати та для кожної з них створює рядок ознак. Після цього навчена модель прогнозує очікувану кількість продажів. Прогноз будується покроково: нові прогнозні значення можуть використовуватися як частина історії для наступних майбутніх дат, що дозволяє сформувати послідовність прогнозу на весь обраний період.

На дев’ятому етапі результат відображається у web-інтерфейсі. Користувач бачить графік, на якому історичні продажі та прогноз розділені вертикальною межею початку прогнозу. Історичні продажі показані до останньої фактичної дати, а прогнозні значення починаються після неї. Додатково система показує короткий висновок, тип динаміки попиту, фактори

впливу та таблицю прогнозу по днях. Така форма подання допомагає інтерпретувати результат, а не лише отримати числовий масив.

На десятому етапі користувач може переглянути сторінку «Модель» і сторінку «Про систему». Сторінка «Модель» пояснює, як оцінювалась якість прогнозування, які моделі порівнювались, чому обрано ExtraTreesRegressor і що означають MAE, RMSE та MAPE. Сторінка «Про систему» описує мету, об’єкт, предмет, вхідні дані, алгоритм роботи, архітектуру, SQLite, санкціонований доступ і практичну цінність. Разом ці сторінки завершують логіку системи: користувач бачить не тільки прогноз, а й обґрунтування того, як він був отриманий.

Таким чином, алгоритм роботи системи охоплює повний цикл: запуск web-застосунку, завантаження або використання CSV-даних, очищення, нормалізацію, формування ознак, навчання та вибір моделі, побудову прогнозу і представлення результату. Цей цикл відповідає темі кваліфікаційної роботи, оскільки демонструє розробку інтелектуальної системи прогнозування попиту на товари для автоматизації аналітичної частини процесів ритейлу.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1. Загальна структура програмного проєкту

Програмна реалізація системи Retail Demand Sense побудована як Flask/Python-застосунок із окремими модулями для web-інтерфейсу, роботи з даними, бази даних, машинного навчання, прогнозування, тестування та документації. Такий поділ дозволяє розділити відповідальність між частинами проєкту: Flask відповідає за взаємодію з користувачем і маршрути, модулі src реалізують обробку даних і машинне навчання, папка models зберігає результати навчання, а папка tests містить автоматичні перевірки.

Центральним файлом web-застосунку є app/app.py. У ньому створюється Flask-застосунок, визначаються маршрути сторінок, API endpoints, логіка завантаження CSV-файлу, виклики модулів прогнозування та передавання даних у HTML-шаблони. Саме цей файл поєднує web-рівень із рівнем даних і машинного навчання. Він не містить усієї ML-логіки безпосередньо, а використовує окремі модулі, що є правильним з погляду організації коду.

Папка app/templates/ містить HTML-шаблони інтерфейсу. До основних шаблонів належать dashboard.html, products.html, forecast.html, metrics.html і about.html. Також у проєкті є base.html, який задає спільну структуру сторінок, навігацію та підключення статичних ресурсів. Такий підхід дозволяє не дублювати спільну розмітку на кожній сторінці та підтримувати єдиний стиль інтерфейсу. Шаблони використовуються Flask для формування готових HTML-сторінок на основі даних, переданих із backend-частини.

Файл app/static/css/styles.css відповідає за візуальне оформлення системи. У ньому описані стилі sidebar-навігації, карток, таблиць, кнопок, форм, графічних блоків та адаптивної поведінки інтерфейсу. Оформлення має стриманий аналітичний характер, що відповідає призначенню системи як дипломного прототипу ML-системи прогнозування попиту. CSS не змінює

логіку роботи, але робить результати зрозумілішими для користувача та комісії під час захисту.

Файл app/static/js/app.js реалізує клієнтську логіку, пов’язану з відображенням графіків і допоміжною взаємодією з інтерфейсом. У ньому є функції для відображення стану кнопок, показу назви вибраного CSV-файлу, побудови графіка попиту та графіка важливості ознак. Графік попиту будується на основі даних, отриманих через API, і спеціально відокремлює фактичні продажі від прогнозу: історичні значення заповнюють синю лінію, а прогнозні значення починаються після історичного періоду та відображаються зеленою пунктирною лінією. Це важливо для коректного пояснення результатів прогнозування.

Модуль src/data_processing.py відповідає за завантаження і підготовку CSV-даних. У ньому визначено обов’язкові та допоміжні колонки, перевірку структури файлу, нормалізацію старих назв полів, перетворення дат, приведення числових значень до коректних типів, очищення некоректних записів і підготовку каталогу товарів. Цей модуль виконує роль першого технічного фільтра між сирими даними та подальшими етапами системи.

Модуль src/feature_engineering.py реалізує формування ознак для машинного навчання. У ньому визначено список ознак FEATURE_COLUMNS, які використовуються моделями. До них входять ідентифікатор товару, категорія в числовому вигляді, базовий попит, ціна, залишок, затримка постачальника, акційний період, святковий день, календарні ознаки, лаги продажів, ковзні середні, зміна ціни та короткостроковий тренд. Саме цей модуль перетворює історичні продажі на набір ознак, придатний для регресійного прогнозування.

Модуль src/train_model.py відповідає за навчання моделей машинного навчання. У ньому завантажуються підготовлені дані, формується навчальна і тестова вибірка, навчаються кілька регресійних моделей, обчислюються метрики якості та обирається найкраща модель. Результатом роботи цього модуля є збережений артефакт моделі та файл метрик. Такий підхід дозволяє

відокремити етап навчання від web-інтерфейсу і не виконувати важкі обчислення під час кожного відкриття сторінки.

Модуль src/predict.py реалізує побудову прогнозу. Він завантажує навчений артефакт моделі, отримує історію вибраного товару, формує майбутні рядки ознак і генерує прогнозні значення на 7, 14 або 30 днів. Також у ньому формується коротке пояснення прогнозу, визначається тип динаміки попиту та виділяються фактори, які могли вплинути на результат. Саме цей модуль забезпечує практичну частину інтелектуальної системи - оцінку майбутнього попиту.

Модуль src/database.py відповідає за роботу з SQLite. У ньому описана SQL-схема, функції створення бази, імпорту продажів, збереження прогнозів, збереження метрик, журналювання імпортів і отримання оглядової інформації. Завдяки цьому база даних відокремлена від шаблонів і ML-модулів. Така структура є зручною для підтримки коду та відповідає принципу розділення відповідальності.

Файл models/metrics.json містить результати оцінки моделі. У ньому зберігаються MAE, RMSE, MAPE, назва обраної моделі, кількість навчальних і тестових записів, дата навчання, важливість ознак і порівняння моделей. Цей файл використовується для швидкого відображення інформації на сторінці «Модель» і для пояснення, чому саме ExtraTreesRegressor обрано як основний алгоритм. Окремо в папці models також зберігається навчений артефакт моделі, який використовується під час прогнозування.

Папка tests/ містить автоматичні тести, які перевіряють основні частини системи. У проєкті є тести API, імпорту даних, feature engineering, навчання моделі та інших допоміжних частин. Вони не замінюють повну промислову систему тестування, але підтверджують, що ключові компоненти працюють очікувано. Папка docs/ містить документацію для дипломної роботи: технічний опис, опис бази даних, інструкцію користувача, план скріншотів, питання комісії, матеріали для захисту та чернетку пояснювальної записки.

Таким чином, структура проєкту є логічно розділеною. Web-рівень знаходиться в папці app, обробка даних і ML-логіка - у src, результати навчання - у models, документація - у docs, а перевірки - у tests. Така організація робить проєкт придатним для пояснення в межах кваліфікаційної роботи з інженерії програмного забезпечення.

3.2. Реалізація backend-частини на Flask

Реалізація backend-частини спирається на можливості Python, Flask і Jinja2 [18; 19; 20].

Backend-частина системи реалізована з використанням Flask. Flask у цьому проєкті виконує роль центрального шару, який поєднує web-інтерфейс, SQLite-базу даних і модулі прогнозування. Він приймає HTTP-запити від користувача, викликає потрібні функції обробки даних або прогнозування, формує контекст для шаблонів і повертає HTML-сторінки або JSON-відповіді. Такий підхід добре підходить для дипломного прототипу, оскільки Flask є достатньо легким, зрозумілим і не потребує складної серверної інфраструктури.

Основний Flask-застосунок створюється у файлі app/app.py. Під час створення застосунку задається секретний ключ, ініціалізується база даних і налаштовується логування. Також у цьому файлі визначено допоміжні функції для очищення кешів, повідомлень користувачу, обробки помилок і формування контексту головної сторінки. Це дозволяє зібрати основну логіку маршрутизації в одному місці, але не змішувати її з низькорівневою обробкою даних.

Головний маршрут / відкриває сторінку «Огляд». Він отримує обраний товар або використовує демонстраційний товар, після чого формує прогноз на стандартний період і передає дані у шаблон dashboard.html. Ця сторінка показує загальну ідею системи, ключові показники набору даних, приклад графіка та коротку інтерпретацію. Вона є першою точкою взаємодії користувача з системою.

Маршрути /products і /dataset відкривають сторінку «Дані». У Flask-коді ці маршрути обробляються однією функцією, яка завантажує історичні продажі, формує summary набору даних, групує записи за товарами та розраховує базову статистику. На цій сторінці користувач бачить кількість товарів, категорій, записів продажів, середній денний попит, а також таблицю товарів із середньою ціною, мінімальними та максимальними продажами, кількістю записів і історичним періодом.

Маршрут /forecast відповідає за сторінку прогнозування. Він отримує список товарів, читає параметр горизонту прогнозування, визначає вибраний товар і викликає функцію forecast_product. Після цього Flask додає до прогнозних рядків метадані дня, наприклад робочий або вихідний день, і передає все у шаблон forecast.html. Ця сторінка є ключовою для демонстрації ML-частини системи, оскільки саме тут користувач бачить фактичні продажі, прогноз, фактори впливу та таблицю прогнозу.

Маршрут /metrics відкриває сторінку «Модель». Він завантажує метрики з models/metrics.json, перетворює технічні назви ознак на зрозумілі підписи та передає дані у шаблон metrics.html. На цій сторінці пояснюється, як оцінювалась модель, які алгоритми порівнювались, які значення MAE, RMSE і MAPE отримано та які фактори мають найбільший вплив на прогноз. Маршрут /about відкриває сторінку «Про систему», де подано опис мети, архітектури, бази даних, доступу і практичної цінності.

Окремим службовим маршрутом є /health. Він повертає JSON-відповідь із інформацією про стан застосунку, наявність моделі, метрик, даних і бази даних. Такий маршрут корисний для швидкої технічної перевірки працездатності системи. Він не є сторінкою для користувача, але важливий для тестування і демонстрації того, що backend може повідомити про стан ключових компонентів.

У системі також реалізовано API endpoints. Наприклад, /api/products повертає список товарів, /api/forecast/<product_id> повертає прогноз для конкретного товару, /api/metrics повертає метрики моделі, а /api/chart-

data/<product_id> повертає історичні та прогнозні значення для побудови графіка. Саме API графіка використовується JavaScript-кодом для відображення фактичних продажів і прогнозу на Canvas. Такий поділ дозволяє не вбудовувати всі дані безпосередньо в HTML, а отримувати їх у структурованому JSON-форматі.

Передавання даних у HTML-шаблони відбувається через функцію render_template. Backend формує словник із потрібними даними: списком товарів, обраним товаром, прогнозом, метриками, оглядовими показниками, історією імпортів і поясненнями. Шаблон отримує ці дані та відображає їх у вигляді карток, таблиць, форм і текстових блоків. Це дозволяє відокремити логіку отримання даних від логіки їх візуального представлення.

Локальний запуск застосунку виконується командою py app/app.py. Якщо файл даних або модель відсутні, код передбачає створення демонстраційного набору або навчання моделі. Після запуску сервер працює на локальному хості й доступний через браузер. Для дипломного прототипу такий підхід є зручним, оскільки не потребує складного деплою, дозволяє швидко продемонструвати систему на комп'ютері й одночасно зберігає реальну структуру web-застосунку.

Flask-застосунок є центральною частиною системи, оскільки саме він координує всі інші компоненти. Він не виконує самостійно всі обчислення, але викликає відповідні модулі: data_processing.py для роботи з CSV, database.py для SQLite, train_model.py для навчання, predict.py для прогнозу. Завдяки цьому backend є не просто набором HTML-сторінок, а інтеграційним рівнем інтелектуальної системи прогнозування попиту.

3.3. Реалізація імпорту та попередньої обробки CSV-даних

Для обробки табличних даних використано підходи, які підтримуються бібліотеками pandas і NumPy [9; 15; 16].

Імпорт CSV-даних є початковим етапом роботи системи, оскільки прогнозування попиту базується на попередніх продажах. На сторінці «Дані»

користувач може завантажити CSV-файл. Backend отримує файл через Flask-запит, читає його за допомогою pandas, перевіряє структуру та зберігає підготовлений набір як основний файл даних. Після цього дані можуть бути імпортовані в SQLite і використані для навчання або прогнозування.

У системі визначено обов'язкові колонки CSV-файлу. До них належать date, product_id, product_name, category, sales_quantity, price, stock_quantity, promo, holiday і supplier_delay_days. Якщо хоча б одна з цих колонок відсутня, система повертає помилку про неправильну структуру CSV. Така перевірка важлива, оскільки модель не може коректно працювати з неповним набором даних. Наприклад, без sales_quantity немає цільової змінної, без date неможливо сформувати часову послідовність, а без product_id неможливо згрупувати продажі за товарами.

За перевірку і підготовку CSV відповідає модуль src/data_processing.py. У ньому реалізовано нормалізацію застарілих назв колонок, наприклад sales може бути перейменовано в sales_quantity, а stock - у stock_quantity. Це робить систему трохи стійкішою до простих відмінностей у назвах полів, але не скасовує вимоги до наявності основної структури. Такий підхід допомагає уникнути помилок у разі використання різних варіантів CSV-файлів.

Після перевірки структури виконується обробка дат. Поле date перетворюється у формат дати за допомогою pandas. Це потрібно для сортування записів, розрахунку календарних ознак і визначення останньої історичної дати. Якщо дата не може бути коректно розпізнана, такий запис не може бути надійно використаний для прогнозування. Тому система очищає дані від рядків, у яких критичні поля не вдалося привести до потрібного формату.

Числові поля також приводяться до числового типу. До них належать ідентифікатор товару, кількість продажів, ціна, залишок, ознаки акції та святкового дня, затримка постачальника, базовий попит і термін придатності, якщо такі поля є в наборі. Перетворення числових значень потрібне для того, щоб модель машинного навчання отримувала коректні числові ознаки, а не

текстові рядки. Некоректні значення, які неможливо перетворити, обробляються як відсутні та можуть бути відкинуті, якщо належать до критичних колонок.

Очищення даних включає видалення рядків із відсутніми значеннями в основних полях, обмеження від'ємних значень продажів і залишків, приведення ознак promo і holiday до діапазону 0 або 1, а також заповнення допоміжних полів стандартними значеннями. Наприклад, якщо в CSV немає поля seasonality_type, система може використати стандартне значення stable. Якщо відсутній product_icon, задається стандартне значення для відображення. Такі дії дозволяють зберегти працездатність системи без зайвого ускладнення CSV-формату.

Нормалізація даних у цьому проєкті означає приведення полів до узгоджених типів і значень. Це не є складною статистичною стандартизацією всіх ознак, а практичним етапом підготовки даних до роботи системи. У результаті записи сортуються за product_id і датою, що важливо для подальшого формування лагів і ковзних середніх. Якщо дані не були б упорядковані в часі, розрахунок попередніх продажів міг би бути некоректним.

Після підготовки дані можуть бути збережені у CSV-файл з кодуванням utf-8-sig, що зручно для відкриття в Excel, а також імпортовані в SQLite через функції з src/database.py. Під час імпорту формується каталог товарів і таблиця історичних продажів. Також створюється запис у журналі імпортів, де зберігається назва файлу, кількість рядків, кількість товарів і статус операції. Це дозволяє відстежувати, що дані були завантажені успішно.

Важливо підкреслити, що імпорт і preprocessing не перетворюють систему на складську або закупівельну платформу. Поля, пов'язані із залишками або затримкою постачальника, використовуються як допоміжні характеристики для аналізу. Система не приймає автоматичних рішень про закупівлі, не керує складом і не змінює бізнес-процеси. Її задача на цьому етапі - підготувати структуровані дані для прогнозування попиту.

Таким чином, реалізація імпорту CSV і попередньої обробки даних забезпечує надійну основу для ML-частини системи. Без цього етапу модель могла б отримати некоректні значення, неповні записи або неправильно впорядковану історію продажів. Саме тому preprocessing є одним із ключових технічних компонентів розробленої системи.

Сторінка перегляду набору даних і базової статистики товарів наведена на рис. 3.1.

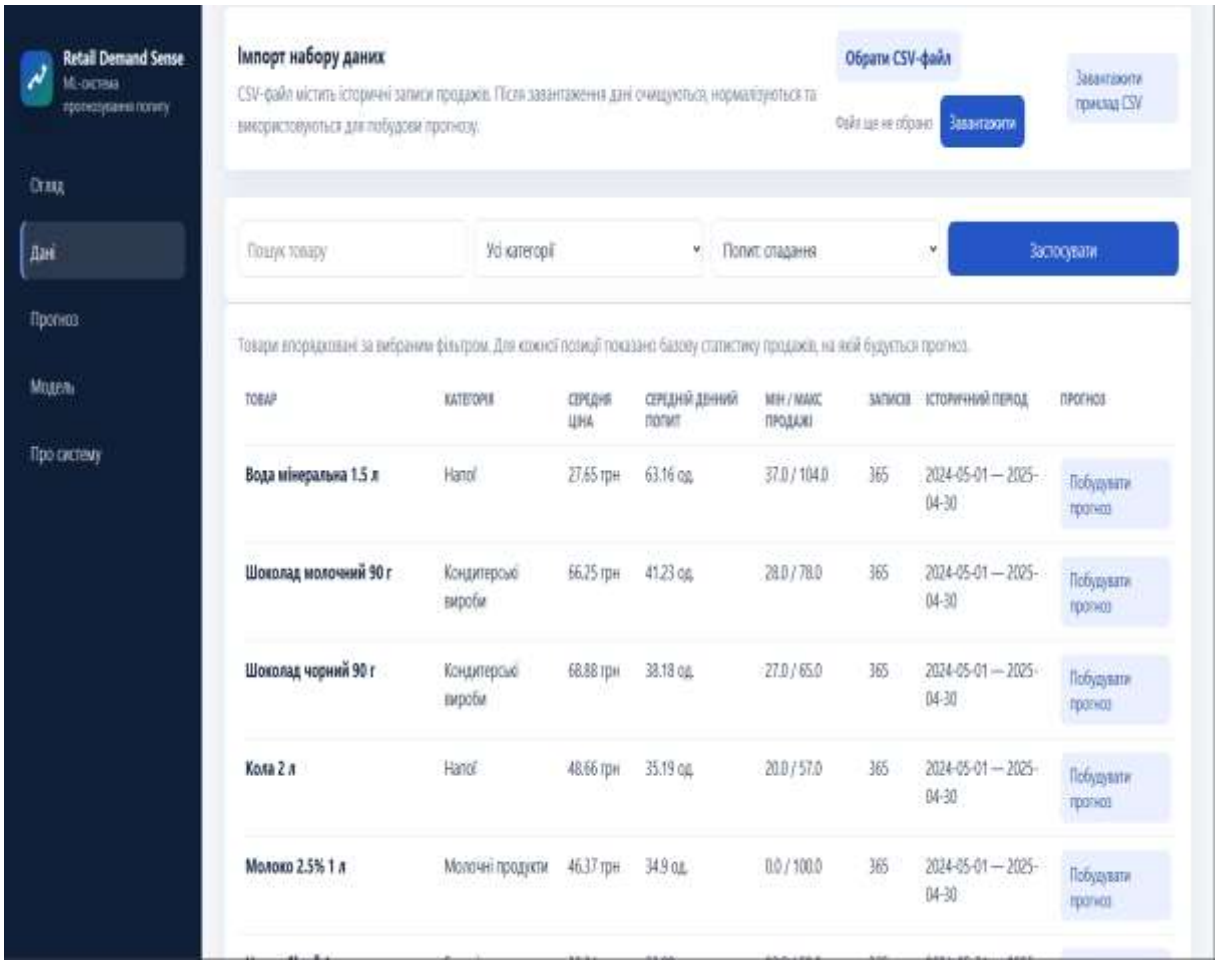


Рис. 3.1. Сторінка набору даних для прогнозування

Приклад завантаження CSV-файлу через web-інтерфейс наведено на рис. 3.2.

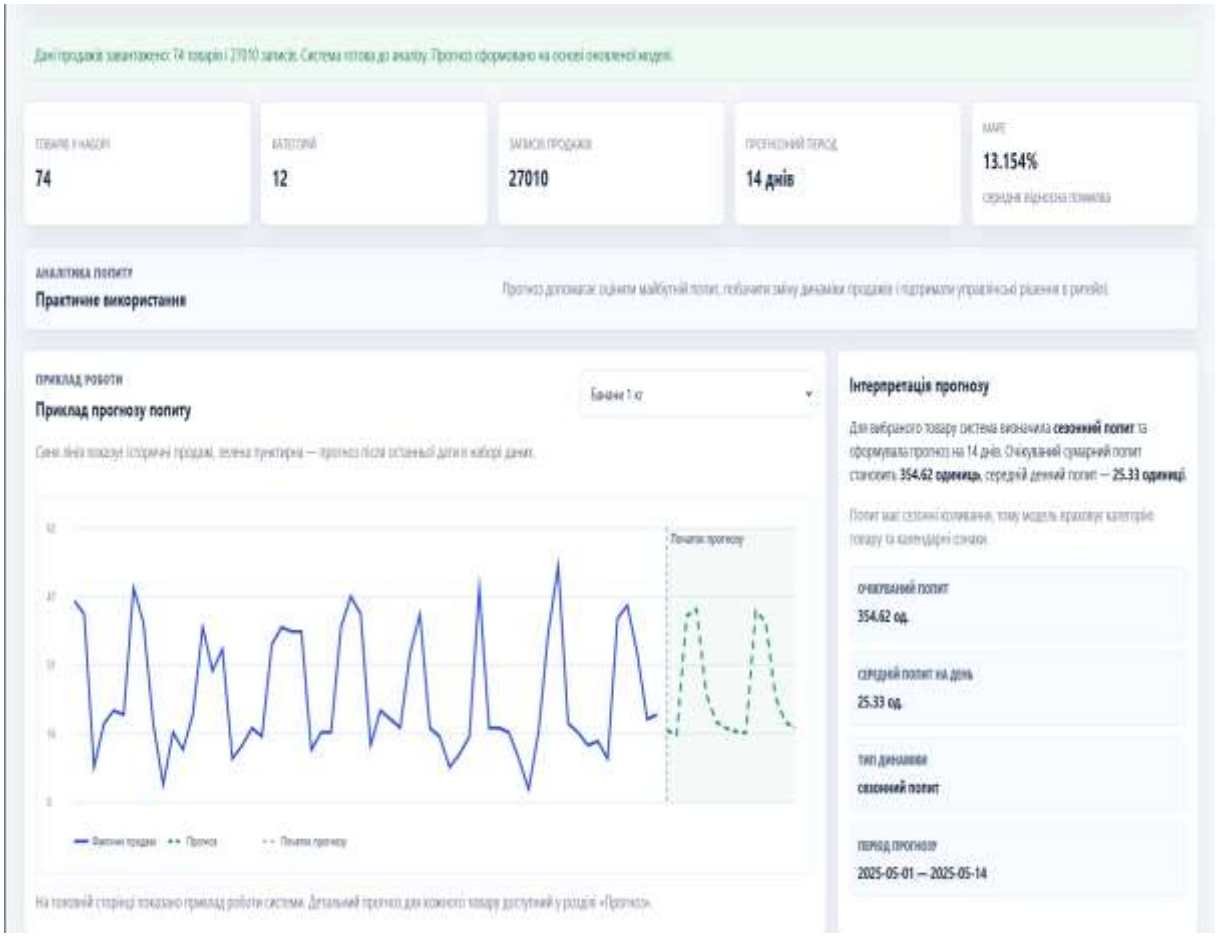


Рис. 3.2. Завантаження CSV-файлу з історичними записами продажів

3.4. Формування ознак для машинного навчання

Формування ознак, або feature engineering, є етапом, на якому початкові історичні записи продажів перетворюються на набір вхідних характеристик для моделі машинного навчання. У задачі прогнозування попиту цей етап є особливо важливим, оскільки модель повинна отримати не лише кількість продажів у минулому, а й контекст, який може пояснювати зміну попиту. У проєкті цей етап реалізовано в модулі src/feature_engineering.py.

Однією з груп ознак є календарні характеристики. Система отримує з дати день тижня, місяць, день місяця, квартал і ознаку вихідного дня. Такі ознаки корисні для ритейлу, оскільки купівельна поведінка може змінюватися залежно від календаря. Наприклад, у вихідні дні попит на частину товарів може бути вищим або нижчим, ніж у робочі дні. Місяць і квартал можуть

відображати сезонні зміни, які важко побачити лише за окремими значеннями продажів.

День тижня є важливою ознакою, оскільки попит у роздрібній торгівлі часто має тижневу циклічність. Для деяких товарів покупці можуть частіше здійснювати покупки наприкінці тижня, для інших - у будні. Ознака вихідного дня доповнює день тижня і дає моделі простіший сигнал про зміну поведінки покупців у суботу та неділю. Такі ознаки не гарантують точного прогнозу самі по собі, але допомагають моделі враховувати регулярні календарні відмінності.

До найбільш важливих ознак належать лагові значення продажів. Лагова ознака показує, якими були продажі товару в попередні періоди. У системі використовуються lag_1, lag_7 і lag_14, тобто продажі за попередній день, попередній тиждень і попередні 14 днів. Такі ознаки корисні, тому що майбутній попит часто залежить від нещодавньої динаміки. Якщо товар стабільно продавався в останні дні, модель отримує сигнал про підтримання певного рівня попиту.

Крім лагів, система використовує ковзні середні за 7 і 14 днів. Ковзне середнє згладжує випадкові коливання і дозволяє оцінити загальний рівень продажів за короткий період. Наприклад, якщо в один день продажі випадково були дуже високими або низькими, середнє за тиждень допомагає не переоцінити цей одиничний стрибок. Середнє за 14 днів дає ширший контекст, а порівняння середнього за 7 і 14 днів дозволяє оцінити короткостроковий тренд.

У модулі feature engineering також формується стандартне відхилення продажів за 7 днів. Воно показує, наскільки нестабільними були нещодавні продажі. Для товарів зі стабільним попитом коливання будуть меншими, а для товарів із піками або сезонними змінами - більшими. Така ознака може допомогти моделі відрізнити стабільний товар від товару, продажі якого змінюються значно сильніше.

Категорія товару також використовується як ознака. Оскільки моделі машинного навчання працюють із числовими значеннями, категорії перетворюються на числовий код. Це дозволяє враховувати, що товари з різних категорій можуть мати різні закономірності попиту. Наприклад, напої можуть мати більш виражену сезонність, а базові продукти харчування - стабільніший попит. Категорія не визначає прогноз повністю, але додає моделі інформацію про тип товару.

Ціна товару використовується як числова ознака, оскільки вона може впливати на кількість продажів. Якщо ціна змінюється, попит також може змінитися. У системі також розраховується ознака зміни ціни, що дозволяє моделі враховувати не лише поточну ціну, а й її відносну зміну. Водночас система не виконує автоматичне ціноутворення. Ціна використовується лише як фактор для аналізу та прогнозування попиту.

Акційний період представлений ознакою promo. Акції можуть створювати піки продажів, тому їх важливо враховувати під час навчання. Крім поточної ознаки акції, у системі розраховується кількість акційних днів за попередній тиждень. Це дозволяє врахувати, що вплив акції може проявлятися не лише в конкретний день, а й у нещодавньому контексті продажів. Ознака святкового дня також використовується як додатковий календарний фактор.

До набору ознак входять також base_demand, stock_quantity, supplier_delay_days, stock_ratio і trend_7. Вони дають моделі додатковий контекст про базовий рівень попиту, доступність товару, затримку постачальника, співвідношення залишку до базового попиту та різницю між коротшим і довшим ковзним середнім. При цьому ці ознаки не означають, що система автоматично керує запасами або логістикою. Вони використовуються як числові характеристики, які можуть бути пов’язані з фактичними продажами.

Функція підготовки навчальних даних формує ознаки, видаляє записи з відсутніми значеннями в критичних колонках і повертає підготовлений набір

разом із мапінгом категорій. Такий мапінг потрібний для того, щоб під час прогнозування майбутніх дат категорія товару кодувалася так само, як під час навчання. Узгодженість ознак під час навчання і прогнозування є обов'язковою умовою правильної роботи моделі.

Отже, feature engineering у системі Retail Demand Sense забезпечує перехід від сирих історичних продажів до навчального набору даних. Саме цей етап дає моделі можливість враховувати календар, попередню динаміку, середній попит, категорію, ціну та акційні умови. Без формування таких ознак прогнозування зводилося б до простого використання історичних значень і було б менш гнучким.

3.5. Навчання моделей і вибір найкращого алгоритму

Порівняння моделей і використання ExtraTreesRegressor узгоджується з підходами scikit-learn та описом ансамблевих методів [13; 14; 22; 24].

Навчання моделей машинного навчання реалізовано в модулі src/train_model.py. Цей модуль відповідає за повний цикл підготовки моделі: завантаження даних, формування ознак, поділ на навчальну та тестову вибірки, навчання кількох алгоритмів, обчислення метрик, вибір найкращої моделі та збереження результатів. Такий поділ дозволяє не виконувати навчання під час кожного відкриття сторінки, а використовувати вже збережену модель для швидкого прогнозування.

Перед навчанням система завантажує CSV-набір даних і передає його до функції підготовки навчальних даних. Після feature engineering формується таблиця, у якій кожен рядок містить набір ознак і цільову змінну sales_quantity. Ознаки беруться зі списку FEATURE_COLUMNS, що гарантує однакову структуру вхідних даних для всіх моделей. Цільова змінна показує фактичну кількість продажів, яку модель повинна навчитися прогнозувати.

Дані поділяються на навчальну і тестову вибірки за часовим принципом. У коді використовується дата, що відповідає приблизно 80% історичного періоду. Записи до цієї межі потрапляють у навчальну вибірку, а пізніші

записи - у тестову. Такий підхід є логічним для прогнозування, оскільки модель навчається на минулому і перевіряється на пізніших даних. Це ближче до реальної ситуації прогнозування, ніж випадкове перемішування всіх записів.

У системі порівнюються три регресійні алгоритми: RandomForestRegressor, GradientBoostingRegressor та ExtraTreesRegressor. Усі вони можуть працювати з табличними даними та нелінійними залежностями. RandomForestRegressor використовує ансамбль дерев рішень, GradientBoostingRegressor будує послідовність моделей, які поступово виправляють помилки, а ExtraTreesRegressor також використовує ансамбль дерев, але з додатковою випадковістю під час побудови розбиттів. Для задачі кваліфікаційної роботи ці моделі є доречними, оскільки добре працюють із числовими та категоріально закодованими ознаками.

Кожна модель навчається на однакових навчальних даних і перевіряється на однаковій тестовій вибірці. Після прогнозування тестових значень система обчислює три метрики: MAE, RMSE і MAPE. MAE показує середню абсолютну помилку в одиницях товару. RMSE сильніше реагує на великі помилки, тому допомагає оцінити ризик значних відхилень. MAPE показує середню відносну похибку у відсотках і є зручною для порівняння моделей.

Для формального оцінювання застосовано такі розрахункові вирази:

$$MAE = (1/n) \cdot \Sigma |y_i - \hat{y}_i|,$$

$$RMSE = \sqrt{((1/n) \cdot \Sigma (y_i - \hat{y}_i)^2)},$$

$$MAPE = (100\%/n) \cdot \Sigma |(y_i - \hat{y}_i) / y_i|.$$

У наведених формулах n означає кількість тестових спостережень, y_i - фактичне значення продажів, а $\hat{y}_i$ - прогнозоване значення. Менше значення кожної метрики означає точніший прогноз.

Вибір найкращої моделі здійснюється за значенням MAPE. Це важливо, оскільки модель не обирається довільно або на основі припущення, що певний алгоритм «кращий». У коді порівнюються отримані значення MAPE для всіх

кандидатів, і як основна зберігається модель із найменшою середньою відносною похибкою. Такий підхід є обґрунтованим для пояснення під час захисту: якість алгоритму визначається результатами на тестовій вибірці.

У поточних метриках проєкту основною моделлю є ExtraTreesRegressor. Вона показала MAPE 13.154%, що є найнижчим значенням серед протестованих алгоритмів у файлі models/metrics.json. Для навчального прототипу такий результат можна вважати прийнятним, оскільки дані містять реалістичні коливання попиту, а модель не має гарантувати ідеальне передбачення. Важливо, що система показує користувачу не лише прогноз, а й числову оцінку його якості.

Після вибору моделі система зберігає артефакт у папці models, а також записує метрики у models/metrics.json. У цьому файлі міститься назва найкращої моделі, MAE, RMSE, MAPE, кількість навчальних і тестових записів, дата навчання, порівняння моделей і важливість ознак. Ці дані використовуються на сторінці «Модель». Таким чином, файл metrics.json виконує роль містка між ML-частиною і web-інтерфейсом.

Окремо варто зазначити важливість порівняння кількох моделей. Якби система використовувала лише один алгоритм, було б складніше обґрунтувати його вибір. Порівняння RandomForestRegressor, GradientBoostingRegressor і ExtraTreesRegressor показує, що розробник перевірив кілька варіантів і обрав той, який дав кращий результат за формальною метрикою. Це відповідає вимогам кваліфікаційної роботи щодо розробки моделі прогнозування та оцінки результатів розрахунків.

Інтерфейс сторінки оцінки якості моделі наведено на рис. 3.3.

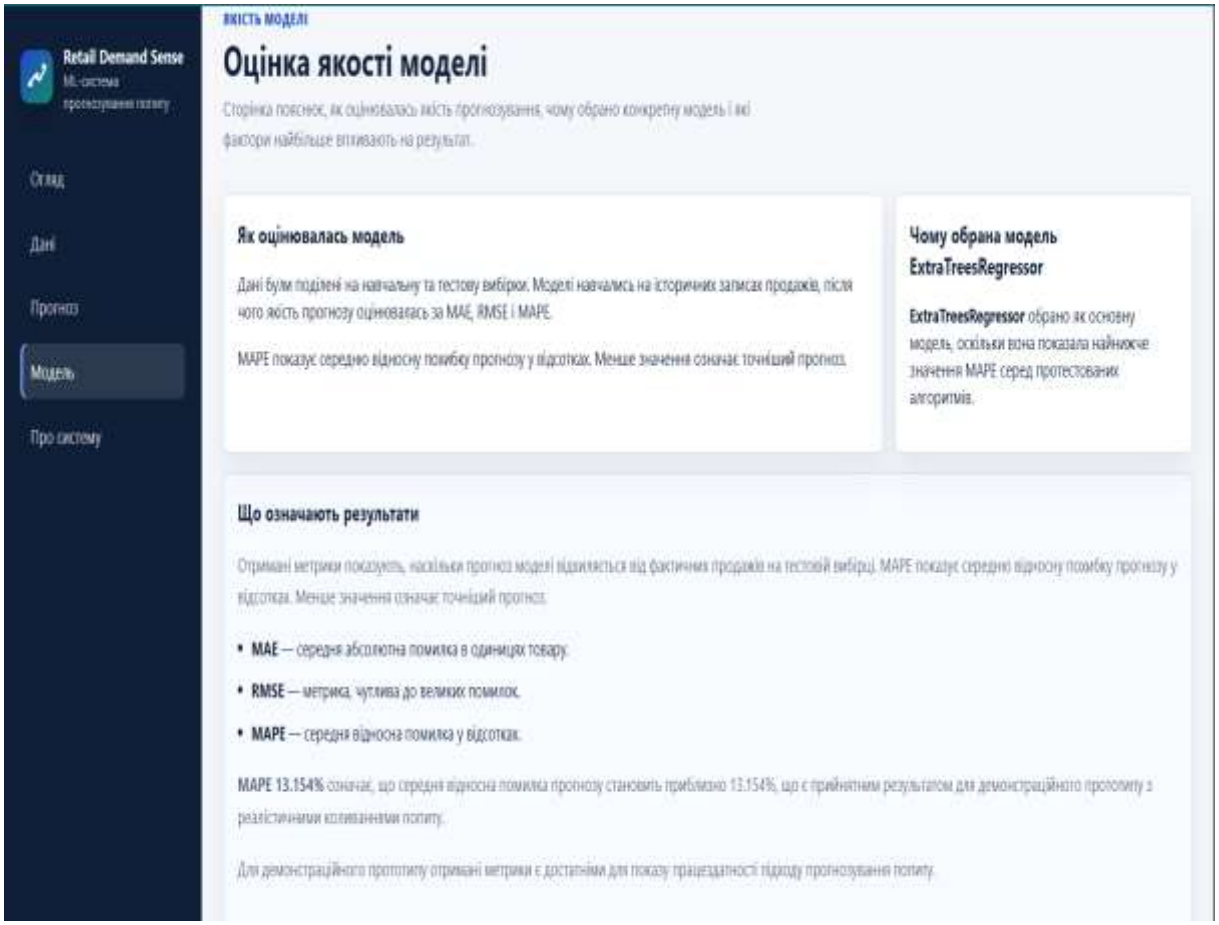


Рис. 3.3. Сторінка оцінки якості моделі

3.6. Реалізація прогнозування попиту

Реалізація прогнозування попиту зосереджена в модулі src/predict.py. Цей модуль відповідає за завантаження навченої моделі, отримання історичних даних конкретного товару, формування ознак для майбутніх дат, генерацію прогнозних значень і підготовку результату для web-інтерфейсу. Прогноз у системі розглядається як очікуване значення попиту, а не як автоматична команда до закупівлі або іншої бізнес-дії.

Користувач починає роботу з прогнозом на сторінці «Прогноз». Він обирає товар зі списку та задає горизонт прогнозування. У системі підтримуються періоди 7, 14 і 30 днів. Такий вибір дозволяє аналізувати короткострокову динаміку попиту без ускладнення інтерфейсу. За замовчуванням часто використовується горизонт 14 днів, оскільки він

достатньо короткий для демонстрації, але водночас дозволяє побачити послідовність майбутніх значень.

Після вибору товару backend викликає функцію forecast_product. Вона завантажує артефакт моделі з models/demand_model.joblib, отримує мапінг категорій і список ознак, а також завантажує історичні продажі з основного CSV-набору. Для вибраного товару беруться останні історичні записи. Якщо товар не знайдено або історії недостатньо, система повертає помилку. Це потрібно для того, щоб прогноз не будувався на випадкових або неповних даних.

Система визначає останню історичну дату продажів і формує майбутні дати прогнозу. Для кожного майбутнього дня створюється рядок ознак за допомогою функції build_future_feature_row з модуля feature engineering. У цьому рядку враховуються дата прогнозу, день тижня, місяць, вихідний день, категорія товару, остання ціна, попередні продажі, ковзні середні, базовий попит та інші ознаки, які узгоджені з тими, що використовувалися під час навчання.

Прогноз формується покроково. Для першої майбутньої дати модель використовує останні фактичні історичні значення. Після отримання прогнозу це значення додається до історії, яка використовується для наступного кроку. Такий підхід дозволяє формувати послідовність прогнозів на кілька днів вперед. При цьому прогнозні значення обмежуються знизу нулем, оскільки кількість продажів не може бути від’ємною.

Результатом роботи модуля прогнозування є словник із даними про товар, категорію, горизонт прогнозування, останню історичну дату, дату початку прогнозу, дату завершення прогнозу, список прогнозних значень, сумарний прогнозований попит, мінімальне, максимальне та середнє прогнозне значення. Також система визначає тип динаміки попиту: стабільний, зростаючий, спадний або сезонний. Це допомагає подати прогноз не лише як числа, а як зрозумілий висновок.

Модуль src/predict.py також формує фактори впливу. До них належать середні продажі за попередній тиждень, середні продажі за 14 днів, порівняння останнього тижня із ширшим періодом, базовий попит категорії, календарні ознаки та акційні періоди, якщо вони присутні в історії. Такі фактори не є повним поясненням внутрішньої роботи моделі, але вони допомагають користувачу зрозуміти, які дані враховувались під час прогнозування.

Відображення фактичної історії окремо від прогнозу реалізовано на рівні API і JavaScript-графіка. Маршрут /api/chart-data/<product_id> повертає історичні продажі та прогнозні значення окремими масивами. У app/static/js/app.js фактичні значення заповнюють лише історичну частину графіка, а прогнозні значення починаються після завершення історії. Між ними відображається вертикальна межа «Початок прогнозу». Це усуває типову помилку, коли прогноз накладається на вже відомі фактичні дані.

Короткий висновок системи показує сумарний прогнозований попит, середній прогнозований попит на день і тип динаміки. Наприклад, система може повідомити, що модель прогнозує певну кількість одиниць на наступні 14 днів, середній прогнозований попит становить певне значення на день, а динаміка оцінюється як спадна, зростаюча, стабільна або сезонна. Такий висновок робить результат доступним для користувача, який не аналізує всі значення таблиці вручну.

Отже, прогнозування в системі реалізовано як послідовний процес: вибір товару, вибір періоду, отримання історії, формування майбутніх ознак, застосування навченої моделі, підготовка результату та візуалізація. Прогноз слід інтерпретувати як аналітичну оцінку очікуваного попиту. Він може підтримати управлінські рішення в ритейлі, але не є автоматичною дією з управління закупівлями, цінами або логістикою.

Приклад сторінки прогнозування попиту для вибраного товару наведено на рис. 3.4.

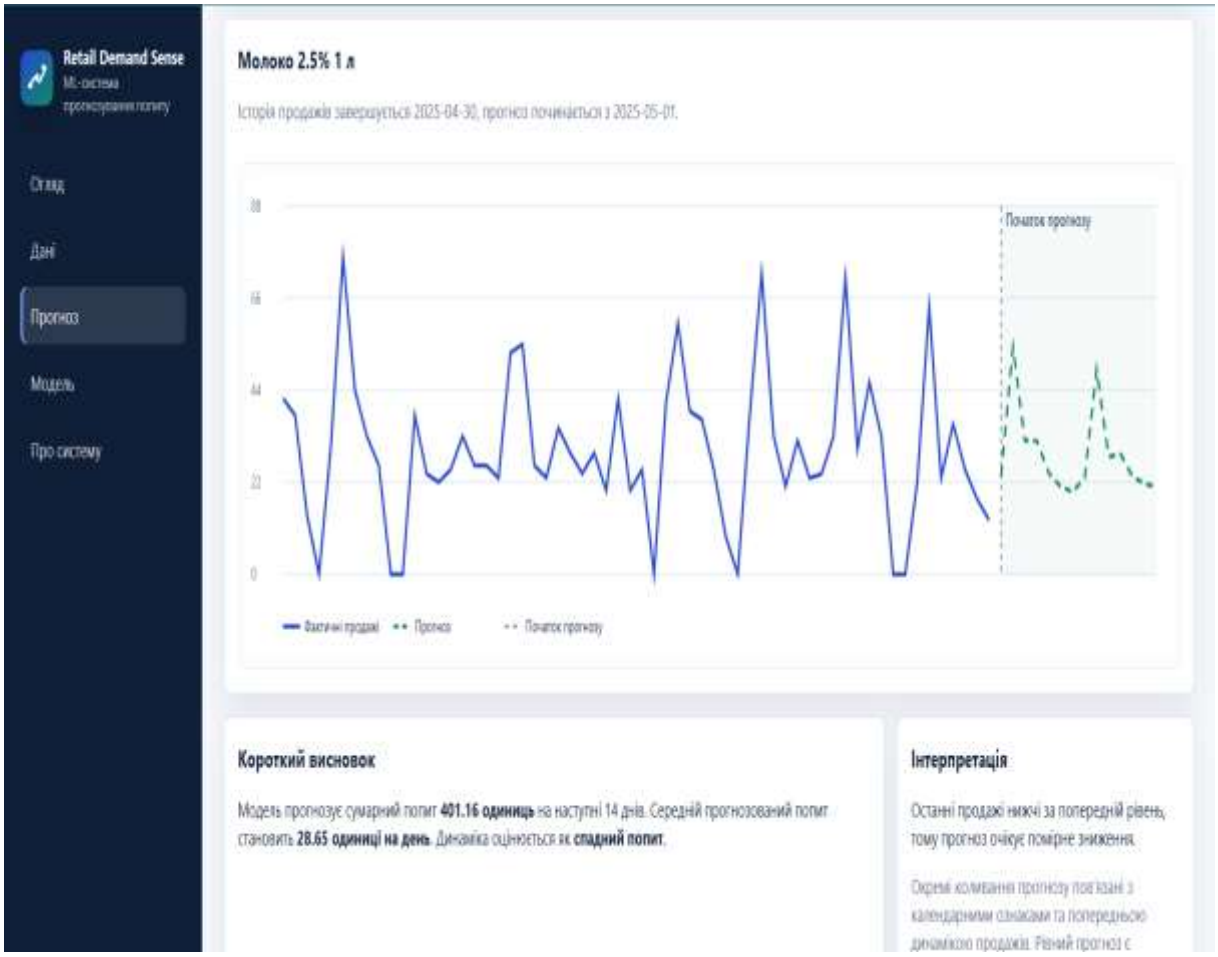


Рис. 3.4. Сторінка побудови прогнозу попиту

3.7. Реалізація web-інтерфейсу

Web-інтерфейс системи реалізовано за допомогою HTML-шаблонів Jinja2, CSS і JavaScript з урахуванням документації Flask, Jinja та стандартних web-технологій [8; 10; 11; 12; 18; 19]. Його головне завдання полягає в тому, щоб зробити результати прогнозування зрозумілими для користувача. Оскільки система створюється як дипломний прототип ML-системи, інтерфейс має не лише показувати дані, а й пояснювати логіку роботи: які дані аналізуються, що прогнозує модель, які метрики використовуються та як інтерпретувати результат.

Базова структура інтерфейсу задається шаблоном base.html. Він містить спільну навігацію, назву системи, підключення CSS і JavaScript, а також основний контейнер для сторінок. Завдяки цьому всі сторінки мають єдиний

стиль і однакову логіку навігації. Sidebar містить основні розділи: «Огляд», «Дані», «Прогноз», «Модель» і «Про систему». Це відповідає структурі системи та не перевантажує користувача зайвими пунктами.

Сторінка «Огляд» реалізована в dashboard.html. Вона виконує роль стартової аналітичної панелі. На ній користувач бачить заголовок системи, короткий опис, KPI набору даних, блок практичного використання, приклад прогнозу попиту та інтерпретацію. Ця сторінка має бути зрозумілою за кілька секунд, тому на ній не показується надмірна технічна інформація. Її задача - швидко пояснити, що система аналізує історичні продажі й прогнозує майбутній попит.

Сторінка «Дані» реалізована в products.html, хоча в інтерфейсі вона позиціонується саме як сторінка набору даних, а не складський каталог. На ній відображаються summary-показники, блок імпорту CSV, структура даних, фільтри та таблиця товарів із базовою статистикою. Користувач може побачити, який історичний період охоплює набір, скільки товарів і категорій у ньому є, який середній денний попит та які товари доступні для прогнозування. Така сторінка підтверджує, що система працює зі структурованими даними.

Сторінка «Прогноз» реалізована в forecast.html. Вона є основною для демонстрації ML-результату. Користувач обирає товар і горизонт прогнозування, після чого бачить графік історичних продажів і прогнозу, короткий висновок, інтерпретацію, фактори впливу, пояснення графіка і таблицю прогнозу по днях. Саме ця сторінка найбільш прямо відповідає темі роботи, оскільки демонструє прогнозування попиту на майбутній період.

Сторінка «Модель» реалізована в metrics.html. Вона пояснює якість прогнозування і використані алгоритми. На ній подано опис оцінювання моделі, причину вибору ExtraTreesRegressor, значення MAE, RMSE і MAPE, порівняння моделей і топ факторів прогнозу. Це важливо для захисту, оскільки комісія має бачити не лише графік, а й обґрунтування якості моделі. Інтерфейс

подає метрики простою мовою, щоб їх можна було пояснити без надмірної математичної деталізації.

Сторінка «Про систему» реалізована в about.html. Вона має пояснювальний характер і пов’язує програмну реалізацію з кваліфікаційною роботою. На ній описано мету роботи, об’єкт, предмет, вхідні дані, алгоритм роботи, архітектуру, базу SQLite, питання санкціонованого доступу, практичну цінність і автора. Ця сторінка корисна як під час демонстрації, так і під час підготовки пояснювальної записки.

CSS-оформлення у styles.css забезпечує стриманий аналітичний стиль. У системі використовуються картки, таблиці, блоки інтерпретації, кнопки, форми, sidebar-навігація та адаптивна розмітка. Візуальний стиль не є маркетинговим і не перевантажений яскравими кольорами, оскільки система повинна виглядати як серйозний навчально-прикладний прототип. Основна увага приділена читабельності, зрозумілій ієрархії та компактному поданню інформації.

JavaScript у app/static/js/app.js використовується насамперед для графіків. Він отримує дані з /api/chart-data/<product_id>, будує графік на Canvas, малює історичну лінію, прогнозну пунктирну лінію, легенду і вертикальну межу початку прогнозу. Також JavaScript оновлює підпис вибраного CSV-файлу і стан кнопок під час дії користувача. У проєкті не використовуються важкі frontend-фреймворки, що відповідає вимозі зберегти систему простою і швидкою для локального запуску.

Навігація побудована так, щоб користувач міг пройти логічний шлях: спочатку ознайомитися з оглядом, потім переглянути дані, побудувати прогноз, оцінити модель і прочитати опис системи. Така структура відповідає сценарію захисту дипломної роботи. Інтерфейс не робить акцент на закупівлях або складських операціях, а зосереджується на прогнозуванні попиту, поясненні факторів і якості моделі.

Таким чином, web-інтерфейс є не лише оболонкою для результатів Python-коду, а важливою частиною системи. Він забезпечує зрозуміле

представлення даних, прогнозів, метрик і архітектури. Саме завдяки інтерфейсу результат роботи моделі стає доступним для користувача, який може оцінити динаміку попиту без прямої роботи з кодом.

Головну сторінку інтерфейсу системи наведено на рис. 3.5.

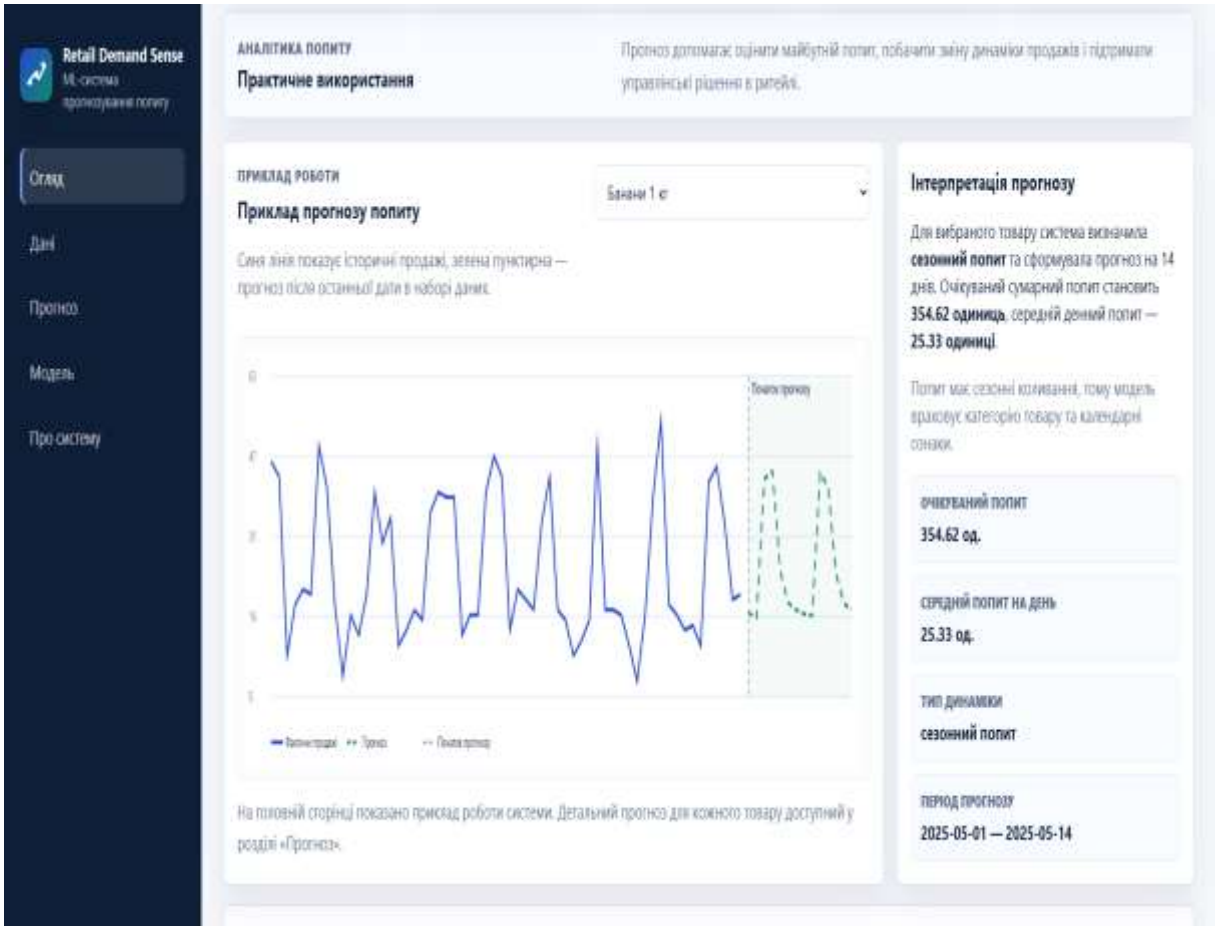


Рис. 3.5. Головна сторінка системи Retail Demand Sense

Сторінка опису системи, яка пояснює мету, об’єкт, предмет, архітектуру й практичну цінність, наведена на рис. 3.6.

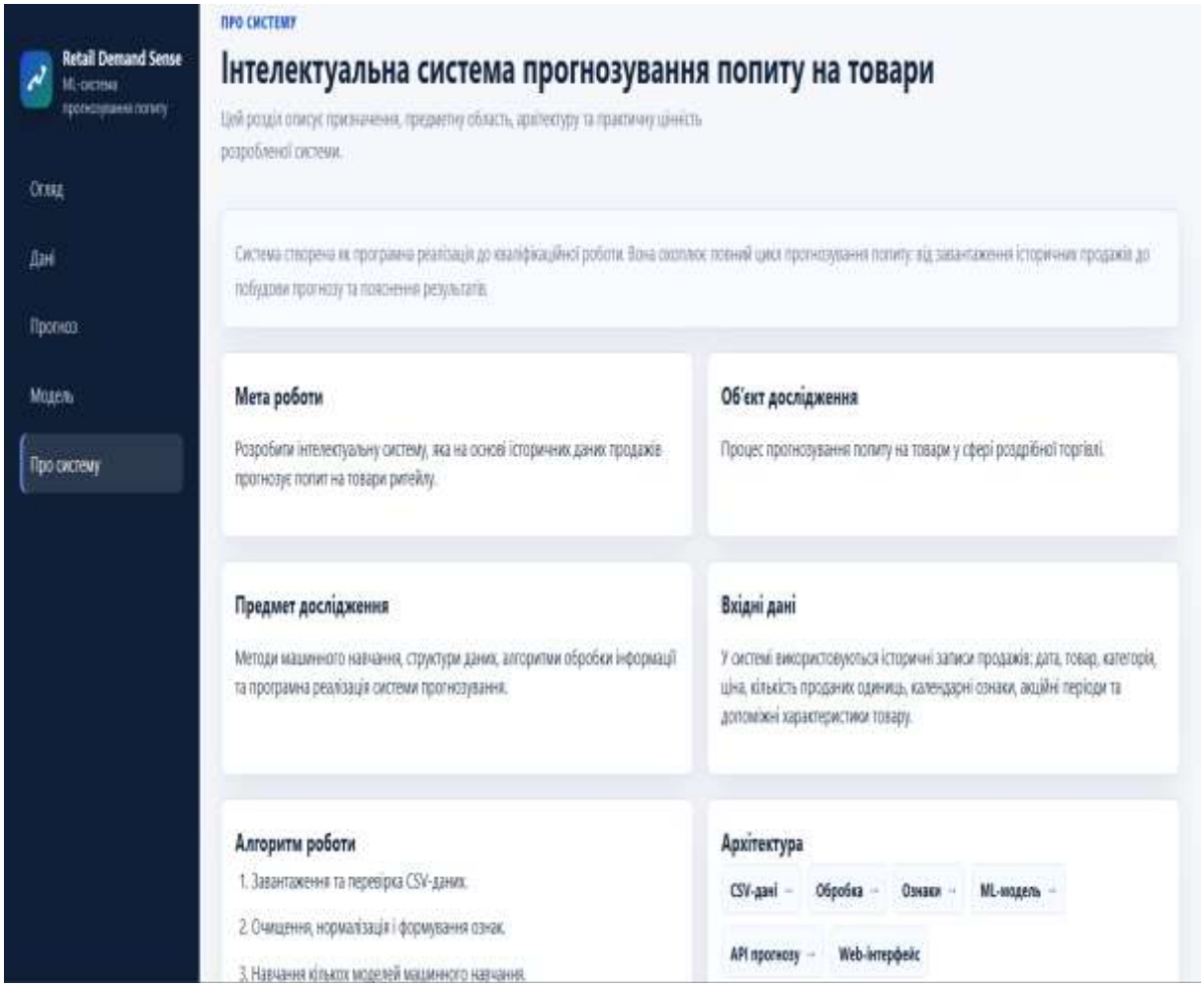


Рис. 3.6. Сторінка опису системи

3.8. Тестування та перевірка працездатності

Перевірка працездатності виконувалась з урахуванням загальних принципів тестування програмного забезпечення [5; 6].

Тестування системи потрібне для підтвердження того, що основні частини програмного проєкту працюють коректно. У межах дипломного прототипу тестування охоплює перевірку запуску, компіляції Python-модулів, працездатності маршрутів, API, імпорту CSV, feature engineering, навчання моделі та побудови прогнозу. Таке тестування не є повним промисловим QA-процесом, але воно достатнє для підтвердження працездатності основних функцій системи.

Першою технічною перевіркою є команда py -m compileall app src. Вона компілює Python-файли в папках app і src та дозволяє виявити синтаксичні помилки. Якщо ця команда проходить успішно, це означає, що основні Python-модулі можуть бути інтерпретовані без синтаксичних проблем. Така перевірка є простою, але корисною перед запуском застосунку або перед захистом.

Другою перевіркою є запуск автоматичних тестів командою py -m pytest. У проєкті тести розташовані в папці tests/. Вони перевіряють API endpoints, структуру прогнозу, сторінки web-інтерфейсу, імпорт даних, наявність обов'язкових колонок, формування ознак і навчання моделі. Якщо тести проходять успішно, це підтверджує, що ключові частини системи не зламані після змін.

У файлі tests/test_api.py перевіряються основні API endpoints і сторінки. Тести відкривають /health, /api/products, /api/metrics, /api/feature-importance та інші маршрути, а також перевіряють форму відповіді прогнозу для /api/forecast/1?days=7. Також перевіряється, що основні сторінки відкриваються без серверних помилок. Це важливо, оскільки web-інтерфейс є основним способом демонстрації системи.

У файлі tests/test_data_import.py перевіряється імпорт даних. Тест бере частину набору продажів, імпортує її через функцію import_sales_dataframe і перевіряє, що кількість імпортованих рядків та товарів визначається коректно. Також перевіряється, що в списку обов'язкових колонок є ключові поля date і sales_quantity. Це підтверджує, що базова логіка CSV-структури присутня в коді.

У файлі tests/test_model.py перевіряється feature engineering і навчання моделі. Тест формування ознак перевіряє, що всі колонки зі списку FEATURE_COLUMNS присутні у підготовлених даних і що кількість підготовлених записів достатня. Тест навчання перевіряє, що функція train_model повертає назву однієї з очікуваних моделей, містить порівняння трьох алгоритмів і має непорожній список важливості ознак. Це підтверджує, що ML-частина системи працює як цілісний процес.

Практична перевірка також включає відкриття маршрутів /, /products, /forecast, /metrics, /about і /health у браузері. Ці маршрути відповідають основним сторінкам системи та службовій перевірці стану. Якщо сторінки відкриваються без помилок, користувач може пройти повний сценарій демонстрації: побачити огляд, переглянути дані, побудувати прогноз, оцінити модель і прочитати опис системи.

Окремо перевіряється API графіка /api/chart-data/<product_id>?days=14. Цей endpoint повертає історичні продажі, прогноз, дату початку прогнозу та останню історичну дату. Саме він використовується JavaScript-кодом для побудови графіка. Перевірка цього API важлива, оскільки графік є центральним елементом сторінок «Огляд» і «Прогноз». Якщо API повертає коректну структуру, frontend може відобразити фактичні продажі та прогноз окремо.

Також перевіряється, що прогноз справді будується для вибраного товару. Для цього можна відкрити сторінку «Прогноз», вибрати товар і період, після чого система повинна показати графік, короткий висновок і таблицю прогнозу. Метрики перевіряються через сторінку «Модель» і API /api/metrics. На цій сторінці повинні відображатися MAE, RMSE, MAPE, обрана модель і порівняння алгоритмів.

Роль тестів у системі полягає в тому, щоб підтвердити працездатність основних компонентів і зменшити ризик випадкових помилок. Вони не доводять, що прогноз завжди буде точним, але показують, що дані завантажуються, ознаки формуються, модель навчається, API відповідає, а сторінки відкриваються. Для дипломного проєкту це є важливим доказом того, що система є не лише макетом інтерфейсу, а реально працюючим програмним прототипом.

Результат автоматичної перевірки проєкту наведено на рис. 3.7.

Рис. 3.7. Результат автоматичної перевірки проєкту

Результати ручної перевірки основних маршрутів web-застосунку наведено в табл. 3.1.

Таблиця 3.1 – Результати ручної перевірки маршрутів

Маршрут	Що перевірено	Результат
/	Відкриття головної сторінки «Огляд»	Успішно, HTTP 200
/products	Відображення набору даних і статистики товарів	Успішно, HTTP 200
/forecast	Відкриття сторінки прогнозування та форми вибору параметрів	Успішно, HTTP 200
/metrics	Відображення метрик моделі MAE, RMSE і MAPE	Успішно, HTTP 200
/about	Відображення опису мети, архітектури та обмежень системи	Успішно, HTTP 200
/health	Отримання службової відповіді про стан застосунку	Успішно, статус ok

3.9. Результати роботи системи

У результаті виконання кваліфікаційної роботи розроблено локальний Flask/Python-застосунок Retail Demand Sense, який реалізує основні етапи прогнозування попиту на товари ритейлу. Система запускається локально командою py app/app.py і відкривається у браузері. Користувач може перейти між сторінками «Огляд», «Дані», «Прогноз», «Модель» і «Про систему», не працюючи безпосередньо з Python-кодом або CSV-файлами вручну.

Система дозволяє переглядати набір історичних продажів. У поточному демонстраційному наборі міститься 74 товари, 12 категорій і 27010 записів продажів. Ці дані використовуються для побудови статистики товарів, формування ознак і прогнозування. На сторінці «Дані» користувач бачить

історичний період, кількість товарів, кількість категорій, кількість записів, середній денний попит і таблицю товарів із базовими показниками.

Основним результатом роботи є можливість побудови прогнозу попиту для вибраного товару. Користувач може вибрати горизонт 7, 14 або 30 днів. У стандартному демонстраційному сценарії використовується прогнозний період 14 днів. Система будує майбутні дати після останньої історичної дати, формує відповідні ознаки, застосовує навчену модель і показує прогнозні значення. Графік відображає фактичні продажі окремо від прогнозу, що дозволяє чітко відрізняти відомі дані від оцінки майбутнього попиту.

На сторінці «Прогноз» користувач бачить не лише лінію прогнозу, а й короткий висновок. У ньому зазначається сумарний прогнозований попит, середній прогнозований попит на день і тип динаміки. Також система показує фактори впливу, зокрема продажі за попередній тиждень, середні продажі за 14 днів, базовий попит категорії, календарні ознаки та акційні періоди, якщо вони є релевантними. Такий підхід робить результат зрозумілішим і придатним для пояснення під час захисту.

Сторінка «Модель» показує якість прогнозування. У поточному файлі models/metrics.json обраною моделлю є ExtraTreesRegressor. Значення MAPE становить 13.154%, MAE - 2.065, RMSE - 3.291. MAPE використовується як основна метрика порівняння, оскільки показує середню відносну похибку у відсотках. Для демонстраційного прототипу результат з MAPE близько 13% можна вважати прийнятним, оскільки система працює з даними, які мають реалістичні коливання попиту. Водночас це не означає, що модель ідеально прогнозує всі товари або всі можливі ринкові ситуації.

Порівняння моделей показує, що ExtraTreesRegressor обрано не випадково. У файлі метрик також наведено результати RandomForestRegressor і GradientBoostingRegressor. ExtraTreesRegressor має найнижче значення MAPE серед протестованих алгоритмів, тому система використовує його як основний. Це дозволяє обґрунтувати вибір моделі на основі кількісної оцінки, а не суб’єктивного припущення.

Результати роботи системи також включають зрозумілий web-інтерфейс. Головна сторінка дає загальне уявлення про систему, сторінка «Дані» показує структуру набору, сторінка «Прогноз» демонструє ML-результат, сторінка «Модель» пояснює якість прогнозування, а сторінка «Про систему» описує мету, архітектуру та практичну цінність. Така структура дозволяє використовувати систему під час захисту як демонстраційний матеріал: від вхідних даних до прогнозу і метрик.

Окремим результатом є наявність документації та тестів. У папці docs підготовлено матеріали для пояснювальної записки, технічного опису, інструкції користувача, питань комісії та плану скріншотів. У папці tests є автоматичні тести, які перевіряють API, імпорт даних, feature engineering і навчання моделі. Це підсилює проєкт із погляду програмної інженерії, оскільки показує не лише роботу інтерфейсу, а й наявність перевірок.

Розроблена система має обмеження, які важливо правильно сформулювати. Вона не є промисловою ERP-, POS-або складською системою та не приймає автоматичних рішень про закупівлі, ціни, маркетингові кампанії чи логістику. Її результатом є прогноз очікуваного попиту та пояснення факторів, які можуть бути використані в управлінському аналізі. Саме така постановка відповідає темі кваліфікаційної роботи і не перебільшує можливості реалізованого прототипу.

Таким чином, програмна реалізація досягла поставленої мети: створено інтелектуальну систему, яка працює з історичними продажами, формує ознаки, навчає і порівнює ML-моделі, будує прогноз попиту та подає результат у зрозумілому web-інтерфейсі. Для демонстраційного прототипу отримані метрики та функціональність є достатніми, щоб показати працездатність підходу прогнозування попиту на товари ритейлу.

ВИСНОВКИ

У кваліфікаційній роботі було розглянуто задачу розробки інтелектуальної системи прогнозування попиту на товари для автоматизації аналітичних процесів ритейлу. Метою роботи була розробка програмного прототипу, який на основі даних продажів формує прогноз майбутнього попиту, оцінює якість прогнозування та надає користувачу зрозумілий інтерфейс для перегляду результатів. У процесі виконання роботи поставлену мету досягнуто: створено систему Retail Demand Sense, яка реалізує повний цикл від завантаження даних до побудови прогнозу та пояснення отриманих результатів.

На першому етапі було проаналізовано предметну область ритейлу та визначено роль попиту на товари в роздрібній торгівлі. Було показано, що попит залежить від багатьох факторів: попередніх продажів, календарних закономірностей, сезонності, ціни, акційних періодів, категорії товару та інших характеристик. Також було досліджено проблемну ситуацію прогнозування попиту. Ручний аналіз продажів є трудомістким і не завжди дозволяє врахувати значну кількість факторів, тому для аналітичної підтримки рішень доцільно використовувати інформаційну систему з елементами ML.

У роботі було розглянуто підходи до прогнозування попиту. Класичні методи, такі як використання середніх значень, аналіз трендів і часових рядів, можуть бути корисними для певних задач, однак вони не завжди достатньо гнучкі для табличних даних ритейлу з різними ознаками. Тому в розробленій системі використано regression-підхід машинного навчання, за якого модель прогнозує числове значення майбутніх продажів на основі сформованих ознак.

Було визначено структуру вхідних CSV-даних. Набір містить дату продажу, ідентифікатор товару, назву товару, категорію, кількість проданих одиниць, ціну, залишок, ознаки акції та святкового дня, затримку постачальника, а також допоміжні поля для опису базового попиту та сезонності. При цьому такі поля, як залишок або затримка постачальника,

розглядаються лише як допоміжні характеристики для аналізу, а не як основа автоматичного управління закупівлями чи логістикою.

У межах проєктування було спроєктовано локальну базу даних SQLite. Вона використовується для збереження товарів, записів продажів, журналу імпортів, прогнозів, метрик моделі та службової інформації. SQLite обрано як доцільне рішення для локального прототипу, оскільки вона не потребує окремого серверного налаштування, легко запускається та дозволяє показати структуру даних у межах дипломної системи.

Програмну реалізацію виконано у вигляді Flask web-застосунку. Flask-застосунок поєднує інтерфейс, базу даних, CSV-імпорт і модулі прогнозування. У системі реалізовано сторінки «Огляд», «Дані», «Прогноз», «Модель» і «Про систему». Сторінка «Огляд» подає загальну інформацію про систему та графік прогнозу. Сторінка «Дані» показує структуру набору даних і базову статистику товарів. Сторінка «Прогноз» дозволяє вибрати товар і горизонт прогнозування. Сторінка «Модель» пояснює якість прогнозу та вибір алгоритму. Сторінка «Про систему» містить опис мети, архітектури та практичної цінності.

Окремо було реалізовано імпорт і попередню обробку CSV-файлів. Система перевіряє наявність обов’язкових колонок, перетворює дату, приводить числові поля до коректного формату, очищає некоректні записи, заповнює допоміжні поля стандартними значеннями та готує дані до подальшого аналізу. Це забезпечує узгодженість вхідного набору і зменшує ризик помилок під час навчання моделі або побудови прогнозу.

Для машинного навчання було реалізовано формування ознак. Система створює календарні ознаки, день тижня, місяць, ознаку вихідного дня, лагові значення продажів, ковзні середні за 7 і 14 днів, ознаки категорії, ціни, акційного періоду та інші допоміжні характеристики. Такі ознаки дозволяють моделі враховувати не лише окремі значення продажів, а й контекст попередньої динаміки попиту.

У системі навчено та порівняно кілька регресійних моделей: RandomForestRegressor, GradientBoostingRegressor та ExtraTreesRegressor. Для оцінки якості використовувалися метрики MAE, RMSE і MAPE. Основною моделлю обрано ExtraTreesRegressor, оскільки вона показала найнижче значення MAPE серед протестованих алгоритмів. Це означає, що вибір моделі був зроблений не довільно, а на основі кількісного порівняння результатів на тестовій вибірці.

Реалізовано прогнозування попиту для вибраного товару на 7, 14 або 30 днів. У прикладі роботи використовується горизонт 14 днів. Прогнозні значення формуються після останньої історичної дати, тому вони не накладаються на фактичні продажі. В інтерфейсі користувач бачить графік історії та прогнозу, короткий висновок, тип динаміки, фактори впливу та таблицю прогнозу по днях. Така форма подання дозволяє пояснити результат не лише як набір чисел, а як аналітичну оцінку очікуваного попиту.

У поточному наборі даних система працює з 74 товарами, 12 категоріями і 27010 записами продажів. За результатами навчання у файлі метрик зафіксовано MAPE 13.154% для моделі ExtraTreesRegressor. Для демонстраційної версії такий результат є прийнятним, оскільки система працює з даними, що мають реалістичні коливання попиту. Водночас отримані метрики не слід трактувати як гарантію ідеального прогнозування. Вони показують працездатність підходу та дають кількісну оцінку якості моделі на тестовій вибірці.

Практична цінність розробленої системи полягає в тому, що вона допомагає аналізувати майбутній попит, бачити динаміку продажів, оцінювати сезонність і використовувати результати прогнозування в аналітичній роботі ритейлу. Система може бути корисною як навчально-прикладний інструмент для аналізу даних продажів і пояснення застосування ML у роздрібній торгівлі. Вона показує, як дані можуть бути перетворені на прогноз і як цей прогноз може бути представлений користувачу у зрозумілій формі.

Водночас межі системи сформульовано свідомо. Retail Demand Sense не є повноцінною ERP-, POS-або складською системою та не керує закупівлями, цінами, маркетинговими кампаніями чи логістикою автоматично. Прогноз попиту не є командою до дії, а є аналітичною оцінкою, яку менеджер або аналітик може використати під час прийняття рішень. Такий підхід дозволяє коректно позиціонувати систему як дипломний програмний прототип, а не як готову промислову платформу.

Під час роботи також було проведено тестування. Перевірено основні маршрути web-інтерфейсу, API, імпорт даних, формування ознак, навчання моделі та доступність метрик. Для технічної перевірки можуть використовуватися команди py -m compileall app src і py -m pytest, а також ручне відкриття сторінок /, /products, /forecast, /metrics, /about і /health. Наявність тестів і документації підсилює проєкт із погляду програмної інженерії.

Можливими напрямами подальшого розвитку системи є підключення до реальної POS-або ERP-системи, використання більшого набору даних, додавання нових моделей, розширення авторизації користувачів і розмежування ролей, поглиблення аналітики товарних залишків, а також експорт прогнозів у Excel або PDF. Отже, розроблена система досягає поставленої мети в межах навчального прототипу та створює основу для подальшого розвитку інтелектуальних інструментів аналізу попиту в ритейлі.

Загалом виконана робота демонструє практичне застосування методів програмної інженерії та ML для прикладної задачі роздрібної торгівлі. У межах одного проєкту було поєднано структурування даних, локальне збереження в SQLite, обробку CSV, побудову ознак, навчання моделей, оцінку точності та web-візуалізацію результатів. Це підтверджує, що поставлена тема може бути реалізована не лише теоретично, а й у вигляді працездатної системи, яку можна запускати локально, демонструвати користувачу та використовувати як основу для подальшого вдосконалення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ 3008:2015. Звіти у сфері науки і техніки. Структура та правила оформлювання. – Київ : ДП «УкрНДНЦ», 2016. – 31 с.
2. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. – Київ : ДП «УкрНДНЦ», 2016. – 16 с.
3. ISO/IEC/IEEE 12207:2017. Systems and software engineering – Software life cycle processes. – Geneva : ISO, 2017.
4. ISO/IEC/IEEE 29148:2018. Systems and software engineering – Life cycle processes – Requirements engineering. – Geneva : ISO, 2018.
5. Pressman R. S. Software Engineering: A Practitioner’s Approach / R. S. Pressman, B. R. Maxim. – 9th ed. – New York : McGraw-Hill Education, 2019. – 672 p.
6. Sommerville I. Software Engineering / I. Sommerville. – 10th ed. – Boston : Pearson, 2016. – 816 p.
7. Owens M. The Definitive Guide to SQLite / M. Owens, G. Allen. – 2nd ed. – Berkeley : Apress, 2010. – 368 p.
8. Grinberg M. Flask Web Development: Developing Web Applications with Python / M. Grinberg. – 2nd ed. – Sebastopol : O’Reilly Media, 2018. – 316 p.
9. McKinney W. Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter / W. McKinney. – 3rd ed. – Sebastopol : O’Reilly Media, 2022. – 579 p.
10. JavaScript [Електронний ресурс] / MDN Web Docs. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 01.06.2026).
11. CSS: Cascading Style Sheets [Електронний ресурс] / MDN Web Docs. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 01.06.2026).

12. HTML: HyperText Markup Language [Електронний ресурс] / MDN Web Docs. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 01.06.2026).
13. Model evaluation: quantifying the quality of predictions [Електронний ресурс] / scikit-learn Developers. – Режим доступу: https://scikit-learn.org/stable/modules/model_evaluation.html (дата звернення: 01.06.2026).
14. scikit-learn User Guide [Електронний ресурс] / scikit-learn Developers. – Режим доступу: https://scikit-learn.org/stable/user_guide.html (дата звернення: 01.06.2026).
15. NumPy Documentation [Електронний ресурс] / NumPy Developers. – Режим доступу: https://numpy.org/doc/ (дата звернення: 01.06.2026).
16. pandas Documentation [Електронний ресурс] / The pandas development team. – Режим доступу: https://pandas.pydata.org/docs/ (дата звернення: 01.06.2026).
17. SQLite Documentation [Електронний ресурс] / SQLite Consortium. – Режим доступу: https://sqlite.org/docs.html (дата звернення: 01.06.2026).
18. Flask Documentation [Електронний ресурс] / Pallets Projects. – Режим доступу: https://flask.palletsprojects.com/ (дата звернення: 01.06.2026).
19. Jinja Documentation [Електронний ресурс] / Pallets Projects. – Режим доступу: https://jinja.palletsprojects.com/ (дата звернення: 01.06.2026).
20. Python 3 Documentation [Електронний ресурс] / Python Software Foundation. – Режим доступу: https://docs.python.org/3/ (дата звернення: 01.06.2026).
21. Pedregosa F. Scikit-learn: Machine Learning in Python / F. Pedregosa et al. // Journal of Machine Learning Research. – 2011. – Vol. 12. – P. 2825–2830.
22. Geurts P. Extremely randomized trees / P. Geurts, D. Ernst, L. Wehenkel // Machine Learning. – 2006. – Vol. 63. – P. 3–42.

23. Friedman J. H. Greedy Function Approximation: A Gradient Boosting Machine / J. H. Friedman // The Annals of Statistics. – 2001. – Vol. 29, No. 5. – P. 1189–1232.
24. Breiman L. Random Forests / L. Breiman // Machine Learning. – 2001. – Vol. 45. – P. 5–32.
25. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow / A. Géron. – 3rd ed. – Sebastopol : O'Reilly Media, 2022. – 861 p.
26. James G. An Introduction to Statistical Learning: With Applications in Python / G. James, D. Witten, T. Hastie, R. Tibshirani, J. Taylor. – Cham : Springer, 2023. – 607 p.
27. Hastie T. The Elements of Statistical Learning: Data Mining, Inference, and Prediction / T. Hastie, R. Tibshirani, J. Friedman. – 2nd ed. – New York : Springer, 2009. – 745 p.
28. Cachon G. Matching Supply with Demand: An Introduction to Operations Management / G. Cachon, C. Terwiesch. – 4th ed. – New York : McGraw-Hill Education, 2019. – 528 p.
29. Simchi-Levi D. Designing and Managing the Supply Chain / D. Simchi-Levi, P. Kaminsky, E. Simchi-Levi. – 3rd ed. – New York : McGraw-Hill, 2008. – 498 p.
30. Kotler P. Marketing Management / P. Kotler, K. L. Keller. – 15th ed. – Boston : Pearson, 2016. – 832 p.
31. Levy M. Retailing Management / M. Levy, B. A. Weitz, D. Grewal. – 10th ed. – New York : McGraw-Hill Education, 2019. – 640 p.
32. Christopher M. Logistics & Supply Chain Management / M. Christopher. – 5th ed. – Harlow : Pearson, 2016. – 328 p.
33. Chopra S. Supply Chain Management: Strategy, Planning, and Operation / S. Chopra, P. Meindl. – 7th ed. – Boston : Pearson, 2019. – 528 p.

34. Silver E. A. Inventory and Production Management in Supply Chains / E. A. Silver, D. F. Pyke, D. J. Thomas. – 4th ed. – Boca Raton : CRC Press, 2017. – 782 p.
35. Hyndman R. J. Forecasting: Principles and Practice [Електронний ресурс] / R. J. Hyndman, G. Athanasopoulos. – 3rd ed. – Melbourne : OTexts, 2021. – Режим доступу: https://otexts.com/fpp3/ (дата звернення: 01.06.2026).
36. Fildes R. Retail forecasting: research and practice / R. Fildes, S. Ma, S. Kolassa // International Journal of Forecasting. – 2019. – Vol. 35, No. 1. – P. 1–8.
37. Нестеренко О. В. Інформаційні системи управління підприємствами : навч. посіб. – Київ : УкрНЦ РІТ, 2019. – 134 с.
38. Єрмошенко М. М., Нестеренко О. В., Штулер І. Ю. Інформаційні технології аналізу даних у маркетингу : навч. посіб. Київ : Національна академія управління, 2021. – 141 с.
39. Нестеренко О. В. Метод пріоритезації вимог в програмній інженерії. Проблеми програмування. – 2024. – № 2–3. – С. 132–139.
40. Нестеренко О. В., Ремига Ю. С. Методичні рекомендації до виконання та захисту кваліфікаційних бакалаврських робіт зі спеціальності 121 «Інженерія програмного забезпечення». Київ : МЄУ, 2024. – 48 с.

ДОДАТОК А. Фрагменти програмного коду

У додатку наведено ключові фрагменти програмної реалізації, які підтверджують роботу основних частин системи: маршрути Flask, перевірку CSV-даних, формування ознак, вибір моделі та побудову прогнозу.

Лістинг А.1 – Маршрути web-застосунку Flask

```
@app.route("/")
def dashboard():
    return render_template("dashboard.html", summary=get_summary())


@app.route("/products")
def products():
    return render_template("products.html", products=get_products())


@app.route("/forecast")
def forecast():
    product_id = request.args.get("product_id", default=1, type=int)
    days = request.args.get("days", default=14, type=int)
    result = build_forecast(product_id=product_id, horizon_days=days)
    return render_template("forecast.html", result=result)


@app.route("/metrics")
def metrics():
    return render_template("metrics.html", metrics=load_metrics())
```

Лістинг А.2 – Перевірка структури CSV-файлу

```
REQUIRED_COLUMNS = {
    "date", "product_id", "product_name", "category",
    "sales_quantity", "price", "stock_quantity",
    "promo", "holiday", "supplier_delay_days"
}


def validate_sales_dataframe(df):
    missing = REQUIRED_COLUMNS -set(df.columns)
    if missing:
        raise ValueError(f"CSV file has no required columns: {missing}")
    df["date"] = pd.to_datetime(df["date"])
    df["sales_quantity"] = pd.to_numeric(df["sales_quantity"])
    df["price"] = pd.to_numeric(df["price"])
    return df.dropna(subset=["date", "product_id", "sales_quantity"])
```

Лістинг А.3 – Формування ознак для машинного навчання

```
def build_features(df):
    df = df.sort_values(["product_id", "date"]).copy()
    df["day_of_week"] = df["date"].dt.dayofweek
    df["month"] = df["date"].dt.month
    df["lag_1"] = df.groupby("product_id")["sales_quantity"].shift(1)
    df["lag_7"] = df.groupby("product_id")["sales_quantity"].shift(7)
    df["rolling_7"] = df.groupby("product_id")["sales_quantity"].transform(
        lambda s: s.shift(1).rolling(7).mean()
    )
    df["rolling_14"] =
df.groupby("product_id")["sales_quantity"].transform(
        lambda s: s.shift(1).rolling(14).mean()
    )
    return df.dropna()
```

Лістинг А.4 – Побудова прогнозу попиту

```
def build_forecast(product_id, horizon_days=14):
    history = load_product_history(product_id)
    model = joblib.load(MODEL_PATH)
    future = build_future_features(history, horizon_days)
    values = model.predict(future[FEATURE_COLUMNS])
    future["forecast"] = np.maximum(values, 0)
    return future[["date", "forecast"]]
```

Лістинг А.5 – Перевірка доступності основних маршрутів

```
def test_main_routes(client):
    routes = ["/", "/products", "/forecast", "/metrics", "/about",
"/health"]
    for route in routes:
        response = client.get(route)
        assert response.status_code == 200
```