МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) автоматизації і комп'ютерних систем імені проф. І.В. Ельперіна

Кафедра інформаційних технологій, штучного інтелекту і кібербезпеки

«До захисту в ЕК»	**«До захисту допущено»**
Директор інституту(декан факультету)	Завідувач кафедри
_____ Андрій ФОРСЮК	_____ Сергій ГРИБКОВ
(підпис) (ім'я та прізвище)	(підпис) (ім'я та прізвище)
«08» грудня 2025р.	«08» грудня 2025р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

зі спеціальності 122 Комп'ютерні науки
(код та назва спеціальності)

освітньо-професійної програми Управління інформацією та аналітика даних

на тему: Створення інтелектуальної системи відділу продажів ТОВ "Пума Україна"

Виконав: здобувач 2 курсу, групи КН-2-3М

Тур Артем Володимироич _____
(прізвище, ім'я, по батькові повністю) (підпис)

Керівник Грама Михайло Петрович _____ _____
(прізвище , ім'я та по батькові повністю) (підпис)

Консультанти_____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

Рецензент ___-___ _____
(ім'я та прізвище) (підпис)

Я як здобувач Національного університету харчових технологій розумію і підтримую політику університету з академічної доброчесності. Я не надавав і не одержував недозволеної допомоги під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач_____
(підпис)

Київ – 2025р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) автоматизації і комп'ютерних систем імені проф. І.В. Ельперіна
Кафедра інформаційних технологій, штучного інтелекту і кібербезпеки
Освітній ступінь магістр
Спеціальність 122 Комп'ютерні науки
(код і назва)
Освітньо-професійна програма Управління інформацією та аналітика даних
(назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри інформаційних технологій, штучного інтелекту і кібербезпеки

_____ **Сергій ГРИБКОВ**
«05» **___листопада ___2025року**

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

_____Тура Артема Володимироича_____
(прізвище, ім'я, по батькові)
1. Тема роботи Створення інтелектуальної системи відділу продажів ТОВ "Пума Україна"_____
керівник роботи_ Грама Михайло Петрович, старший викладач, доктор філософії_
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від 05 листопада 2025 року №906-кс
2. Строк подання здобувачем роботи_____01 грудня 2025 року_____
3. Вихідні дані до роботи Інтелектуальна система відділу продажів ТОВ "Пума Україна"_____
_____
_____
_____
_____
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)_____
Розділ 1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ_____
Розділ 2. ДОСЛІДЖЕННЯ ТА ОБҐРУНТУВАННЯ ТЕХНОЛОГІЙ, МЕТОДІВ, АЛГОРИТМІВ_____
Розділ 3. РЕАЛІЗАЦІЯ ТА АПРОБАЦІЯ ІНФОРМАЦІЙНОЇ АНАЛІТИЧНОЇ СИСТЕМИ_____
5. Перелік графічного матеріалу:
Інтерфейс користувача, схеми._____

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Грама М.П. , ст. вик. , д. ф	01.10.2025	15.10.2025
2	Грама М.П. , ст. вик. , д. ф	15.10.2025	03.11.2025
3	Грама М.П. , ст. вик. , д. ф	03.11.2025	20.11.2025
4	Грама М.П. , ст. вик. , д. ф	20.11.2025	01.12.2025

7. Дата видачі завдання: ____01 жовтня 2025 року_____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Огляд сучасних методів розробки записок кваліфікаційних робіт.	01.10.2025 – 09.10.2025	Виконано
2	Формулювання мети та завдання системи.	09.10.2025 – 14.10.2025	Виконано
3	Розробка інтерфейсу користувача, реалізація методів машинного навчання.	10.11.2025 – 15.11.2025	Виконано
4	Тестування розробленої системи.	16.11.2025 – 17.11.2025	Виконано
5	Оформлення першого розділу.	17.11.2025 – 19.11.2025	Виконано
6	Оформлення другого розділу.	19.11.2025 – 21.11.2025	Виконано
7	Оформлення третього розділу.	21.11.2025 – 23.11.2025	Виконано
8	Оформлення презентації.	23.11.2025 – 01.12.2025	Виконано

Здобувач _____ __Артем ТУР_____
(підпис) (ім’я та прізвище)

Керівник роботи _____ __Михайло ГРАМА_
(підпис) (ім’я та прізвище)

АНОТАЦІЯ

Кваліфікаційна робота присвячена розробці та дослідженню ефективності інтелектуальної інформаційної системи, призначеної для комплексного аналізу даних про продажі та прогнозування попиту в роздрібній мережі (на прикладі ТОВ “Пума Україна”). Мета роботи полягає у створенні, веб-додатку з гібридною архітектурою (Next.js) для аналізу та прогнозування найкращого часу продаж.

В рамках проєкту було спроєктовано гібридну архітектуру на базі фреймворку Next.js та мови TypeScript, що забезпечує як швидке відображення (SSR), так і глибоку інтерактивність (CSR). Розроблено два ключові модулі: інтерактивний дашборд із понад 20 візуалізаціями (Recharts, Tailwind CSS) [19] для детальної аналітики (тренди, демографія, ціноутворення) та прогностичний модуль.

Ключовим досягненням є реалізація ядра Machine Learning повністю на клієнті (TypeScript), яке включає власні алгоритми Decision Tree, Bagging та Boosting. Це рішення забезпечує миттєве виконання прогнозів без залежності від серверних ресурсів. Для ефективної боротьби з дисбалансом класів у вхідній вибірці впроваджено техніки попередньої обробки даних, включаючи SMOTE (Synthetic Minority Oversampling Technique).

На завершальному етапі проведено експериментальне дослідження, що включає порівняльний аналіз метрик якості (Accuracy, Precision, Recall, F1-Score) трьох ансамблевих моделей, підтверджуючи їхню ефективність для вирішення задачі класифікації періодів високого попиту. Створений програмний продукт являє собою сучасний, масштабований інструмент, що демонструє високу продуктивність (забезпечену TypeScript та Zod) та надає користувачам ключові інструменти для прийняття обґрунтованих бізнес-рішень.

Ключові слова: АНАЛІТИЧНА СИСТЕМА, ПРОГНОЗУВАННЯ ПРОДАЖІВ, МАШИННЕ НАВЧАННЯ, АНСАМБЛЕВІ МОДЕЛІ, NEXT.JS.

SUMMARY

The thesis is devoted to the development and research of the effectiveness of an intelligent information system designed for comprehensive analysis of sales data and forecasting demand in a retail network (using the example of Puma Ukraine LLC). The goal of the work is to create a web application with hybrid architecture (Next.js) for analyzing and forecasting the best time to sell.

As part of the project, a hybrid architecture based on the Next.js framework and the TypeScript language was designed, providing both fast rendering (SSR) and deep interactivity (CSR). Two key modules were developed: an interactive dashboard with over 20 visualizations (Recharts, Tailwind CSS) [19] for detailed analytics (trends, demographics, pricing) and a forecasting module.

A key achievement is the implementation of the Machine Learning core entirely on the client side (TypeScript), which includes proprietary Decision Tree, Bagging, and Boosting algorithms. This solution provides instant forecasting without relying on server resources. To effectively combat class imbalance in the input sample, pre-processing techniques have been implemented, including SMOTE (Synthetic Minority Oversampling Technique).

At the final stage, an experimental study was conducted, including a comparative analysis of quality metrics (Accuracy, Precision, Recall, F1-Score) of three ensemble models, confirming their effectiveness in solving the task of classifying periods of high demand. The created software product is a modern, scalable tool that demonstrates high performance (provided by TypeScript and Zod) and provides users with key tools for making informed business decisions.

Keywords: ANALYTICAL SYSTEM, SALES FORECASTING, MACHINE LEARNING, ENSEMBLE MODELS, NEXT.JS.

ЗМІСТ

ВСТУП

Кваліфікаційна робота: 145 сторінок, 18 рисунків, 6 таблиць, 2 додатки, 20 джерел.

Стрімкий розвиток інформаційних технологій та зростання обсягів даних у роздрібній торгівлі вимагають від компаній впровадження високоточних, аналітичних та прогностичних інструментів. В умовах високої конкуренції, особливо у сегменті спортивних товарів, здатність оперативно приймати рішення на основі даних про продажі стає вирішальним фактором успіху. Традиційні методи аналізу часто є недостатніми для виявлення складних, нелінійних залежностей, що визначають динаміку продажів. Саме тому створення інтелектуальних систем, які використовують сучасні методи машинного навчання, набуває особливої значущості.

Актуальність теми магістерської роботи, що присвячена створенню інтелектуальної системи відділу продажів ТОВ “Пума Україна”, визначається необхідністю переходу компанії (PUMA Ukraine) від реактивного обліку до проактивного, інтелектуального управління продажами. Це стратегічне завдання в умовах динамічного ринку вимагає оптимізації робочих процесів, точного прогнозування попиту та максимізації ефективності торгового персоналу. Впровадження ансамблевих методів машинного навчання, зокрема бустінгу (Boosting) та бегінгу (Bagging), дозволяє підвищити точність прогнозування оптимального часу доби для здійснення продажів, а створення аналітичних дашбордів забезпечує наочний інструмент для швидкого аналізу ключових показників ефективності (KPI), що відповідає сучасним вимогам data-driven менеджменту.

З огляду на це, мета дослідження полягає у теоретичному обґрунтуванні та розробці практичних рекомендацій щодо створення та впровадження інтелектуальної системи для відділу продажів ТОВ “Пума Україна”, яка здатна проводити глибокий аналіз, формувати аналітичні дашборди та здійснювати високоточне прогнозування для оптимізації торгової діяльності. Об’єктом

дослідження обрано процес управління продажами та аналізу даних у відділі продажів ТОВ “Пума Україна”, тоді як предметом дослідження є методи та інструменти побудови цієї інтелектуальної системи, що включають алгоритми машинного навчання (Boosting, Bagging) та засоби візуалізації даних (аналітичні дашборди), спрямовані на підвищення ефективності продажів.

Для досягнення поставленої мети було вирішено низку ключових завдань, зокрема: проаналізувати поточний стан організації продажів та потреби відділу; дослідити теоретичні засади функціонування інтелектуальних систем та ансамблевих методів; розробити архітектуру системи; створити комплекс аналітичних дашбордів та графіків для візуалізації метрик; застосувати та порівняти ефективність ансамблевих методів бустінгу [8] та бегінгу [10] для прогнозування оптимального часу доби для кращих продажів; а також сформувати практичні рекомендації щодо її впровадження в операційну діяльність.

У роботі використано комплекс методів дослідження, включаючи системний та порівняльний аналіз для оцінки підходів до прогнозування, статистичний та математичний аналіз для обробки даних, а також спеціальні методи машинного навчання (бустінг та бегінг) для побудови прогностичної моделі. Для представлення результатів застосовано методи візуалізації даних, зокрема створення інформативних аналітичних дашбордів та графіків.

Наукова новизна одержаних результатів полягає у тому, що вперше розроблено прогностичну модель визначення оптимальних часових інтервалів доби для продажів у роздрібній мережі спортивних товарів. Модель ґрунтується на порівняльному застосуванні ансамблевих алгоритмів машинного навчання (бустингових та бегінгових методів), що дозволило виявити їх специфічну поведінку у задачах часової класифікації та довести залежність між вибором алгоритму та стабільністю прогнозів. Удосконалено підхід до інтерпретації результатів прогнозування за рахунок інтеграції аналітичних візуалізацій, які забезпечують наочне подання та пояснення роботи моделі. Практичне значення отриманих результатів полягає в можливості впровадження розробленої прогностичної моделі в операційну діяльність роздрібної мережі: система дозволяє

точно визначати найвигідніші періоди продажів, оптимізувати робочий графік персоналу та підвищувати ефективність управлінських рішень, що в кінцевому результаті сприяє збільшенню обсягів продажів і прибутковості компанії.

Особистий внесок здобувача полягає у самостійній розробці концепції системи, проведенні системного аналізу даних продажів, виборі, налаштуванні та застосуванні ансамблевих моделей машинного навчання (бустінгу та бегінгу), а також у створенні аналітичних дашбордів та графіків.

Кваліфікаційна робота виконувалась згідно із планом науково-дослідних робіт кафедри інформаційних технологій, штучного інтелекту і кібербезпеки Національного університету харчових технологій: НДР “Дослідження та використання засобів штучного інтелекту для розв'язання управлінських завдань в харчовій промисловості” № ДР 0125U003887 , 2025–2030 рр.; “Дослідження та впровадження сучасних методів аналізу даних у харчову промисловість” № ДР 0125U003889, 2025–2031 рр.

Матеріали за темою дослідження були представлені на Міжнародній науково-практичній конференції «XII МІЖНАРОДНА НАУКОВО-ТЕХНІЧНА INTERNET-КОНФЕРЕНЦІЯ».

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Історія компанії

ТОВ “Пума Україна” є офіційним представником німецького бренду PUMA SE [1] - одного з провідних світових виробників спортивного одягу, взуття та аксесуарів. Компанія була зареєстрована в Україні у 2005 році з метою розвитку ринку спортивної моди та дистрибуції продукції PUMA. За роки діяльності підприємство розширило свою присутність на українському ринку, відкривши мережу фірмових магазинів у найбільших містах країни та налагодивши активну співпрацю з ритейл-партнерами.

Компанія неодноразово отримувала престижні відзнаки. У 2022–2023 роках увійшла до рейтингу 50 найкращих роботодавців України за версією Forbes. У 2024 році стала “Ритейлером року” в сегменті спортивного одягу та взуття (RAU Awards) та втретє поспіль потрапила до топ-25 компаній для молодих спеціалістів за версією STUDENTPOINT. Також PUMA, включно з українським офісом, отримала звання Global Top Employer 2025.

Згідно з дослідженням структури підприємства, було розроблено модель його організації (рис. 1.1).

ТОВ “Пума Україна” має класичну організаційну структуру, що включає такі основні підрозділи:

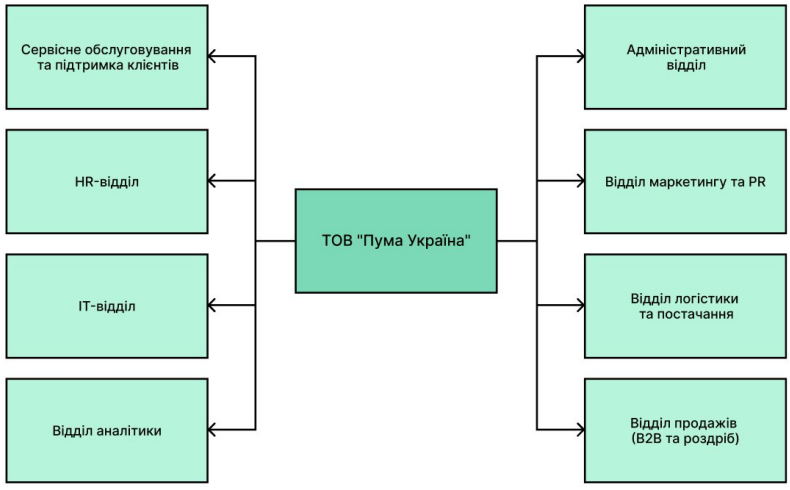

Рисунок 1.1 – Організаційна структура ТОВ “Пума Україна”

Компанія не має власного виробництва на території України, оскільки продукція PUMA виготовляється на міжнародних виробничих потужностях. В Україні здійснюється імпорт, дистрибуція та реалізація продукції.

ТОВ “Пума Україна” [20] є стабільною комерційною організацією з високими показниками обороту та рентабельності. Компанія орендує сучасні офісні приміщення в Києві, має власні складські потужності та автопарк для логістичних потреб. Основними економічними показниками є обсяг продажів, ріст частки ринку, прибутковість операційної діяльності та рівень задоволеності клієнтів.

Підприємство не має виробничих потужностей в Україні. Основні операційні потужності зосереджені в логістичних процесах — обробці та доставці замовлень. Асортимент продукції включає: спортивне взуття, спортивний та повсякденний одяг, аксесуари.

Офіс ТОВ “Пума Україна” зазвичай працює з понеділка по п’ятницю з 9:00 до 18:00. Магазини компанії працюють щодня відповідно до режиму роботи торгових центрів, у яких вони розташовані (зазвичай з 10:00 до 21:00).

Оскільки компанія не займається виробництвом, потреба в сировинних ресурсах відсутня. Енергетичне забезпечення офісу та складів здійснюється за рахунок централізованого електропостачання. Для логістики компанія використовує автотранспортні засоби, які заправляються відповідними видами пального.

1.2. Дослідження роботи відділу аналітики ТОВ “Пума Україна”

У сучасних умовах конкурентного ринку ефективне управління підприємством неможливе без ґрунтовного аналітичного підходу до прийняття рішень. Враховуючи масштаб діяльності ТОВ “Пума Україна” як офіційного представника міжнародного бренду спортивного одягу та взуття, особливе значення має функціонування аналітичного підрозділу, який забезпечує керівництво та інші відділи актуальною, структурованою та стратегічно важливою інформацією.

Аналітичний відділ виконує ключову роль у зборі, обробці та інтерпретації даних, що дозволяє оперативно реагувати на зміни попиту, оптимізувати запаси, оцінювати ефективність маркетингових кампаній та прогнозувати фінансові показники. Саме тому дослідження його роботи є важливим кроком для виявлення можливостей покращення процесів та впровадження нових інформаційних технологій. Таблиця 1.1. з відповідним документообігом наведена нижче:

Таблиця 1.1 – Документообіг відділу аналітики

Документ	**Джерело**	**Призначення**
Звіт про продажі	Відділ продажів (Retail, E-commerce)	Щоденний/тижневий аналіз динаміки
Дані інвентаризації	Логістика	Аналіз наявності товару
Витрати на маркетинг	Маркетинговий відділ	ROI аналіз рекламних кампаній
Онлайн-поведінка користувачів	E-commerce, Google Analytics	Виявлення трендів, воронки
Фінансові звіти	Бухгалтерія	Кореляція витрат і прибутку
HR-дані	HR-відділ	Оцінка продуктивності, ефективності персоналу

1.2.1. Зв’язки відділу аналітики з іншими відділами у компанії

Ефективна робота аналітичного відділу неможлива без тісної співпраці з усіма ключовими структурними одиницями підприємства. Завдяки інтеграції даних з різних джерел, аналітики можуть формувати цілісну картину стану бізнесу, виявляти закономірності, прогнозувати результати та пропонувати оптимізаційні рішення.

Наведена нижче таблиця 1.2, демонструє основні напрями взаємодії відділу аналітики ТОВ “Пума Україна” з іншими підрозділами компанії. Вона відображає характер співпраці, типи інформації, які обмінюються між відділами, а також періодичність такої взаємодії. Такий структурований підхід до комунікації сприяє злагодженій роботі підприємства в цілому.

Таблиця 1.2 – Взаємодія інших відділів з відділом аналітики

Підрозділ / Відділ	**Характер взаємодії**	**Тип даних / Інформація**	**Періодичність**
Відділ продажів (Retail)	Надання щоденних даних про обсяги продажів	Продажі по регіонах, магазинах, категоріях товару	Щодня / Щотижня
Відділ електронної комерції	Аналіз поведінки користувачів, конверсій, онлайн-продажів	Дані з сайту, Google Analytics, замовлення	Щодня
Маркетинговий відділ	Оцінка ефективності рекламних кампаній, розрахунок ROI	Бюджет, охоплення, продажі, трафік	Щомісяця
Логістичний відділ	Аналіз обігу товарів, залишків, оптимізація постачання	Запаси, обсяги поставок, швидкість обороту	Щотижня

Продовження таблиці 1.2.

Підрозділ / Відділ	**Характер взаємодії**	**Тип даних / Інформація**	**Періодичність**
Фінансовий відділ	Формування фінансової звітності, прогнозів витрат та прибутків	Бюджети, витрати, доходи, маржинальність	Щомісяця
HR-відділ	Аналіз ефективності працівників, продуктивності команд	Дані про персонал, показники ефективності, графіки	Щокварталу
Керівництво	Надання звітів для стратегічного планування	Зведені дашборди, ключові показники (KPI), прогнози	За потреби / Щомісяця
IT-відділ	Технічна підтримка BI-систем, баз даних, автоматизація процесів	Синхронізація даних, технічне обслуговування	Постійно

1.2.2. Візуалізація процесу створення аналітичного звіту

Нижче наведено бізнес-процес формування звіту в аналітичному відділі ТОВ “Пума Україна”. Для забезпечення прозорості та оптимізації цього процесу було побудовано схему, яка зображена нижче (рис. 1.2). вона відображає основні етапи – від отримання запиту до передачі результатів та збору зворотного зв’язку.

Ця візуалізація дає змогу краще зрозуміти логіку роботи аналітичного підрозділу, виявити вузькі місця та визначити точки потенційного вдосконалення за допомогою сучасних інформаційних рішень.

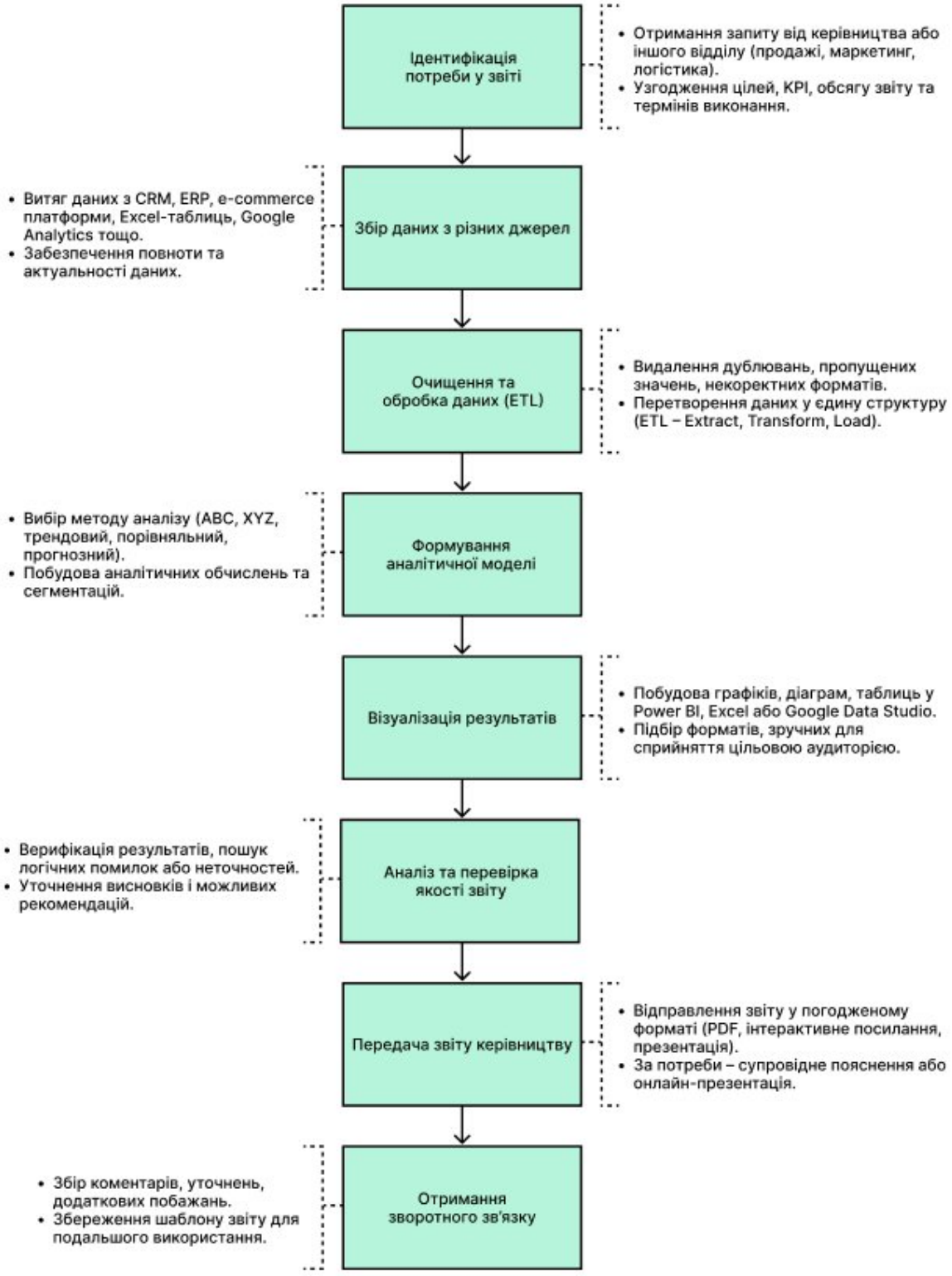

Рисунок 1.2 – Візуалізація формування звіту в аналітичному відділі

1.2.3. Опис інформаційних систем які використовуються у відділі

У діяльності аналітичного відділу важливу роль відіграють сучасні інформаційні системи, які забезпечують збір, зберігання, обробку та візуалізацію великих обсягів даних. Без автоматизації та цифрових рішень неможливо забезпечити оперативність і точність аналітики, особливо в умовах динамічного розвитку роздрібної торгівлі та електронної комерції.

ТОВ “Пума Україна” активно впроваджує передові програмні продукти та платформи для підтримки аналітичних процесів. Завдяки цьому аналітичний відділ має можливість працювати з багатьма джерелами інформації одночасно, створювати інтерактивні дашборди, генерувати прогнози та швидко передавати ключові висновки керівництву компанії.

Схема засобів використання наведена нижче (рис. 1.3).

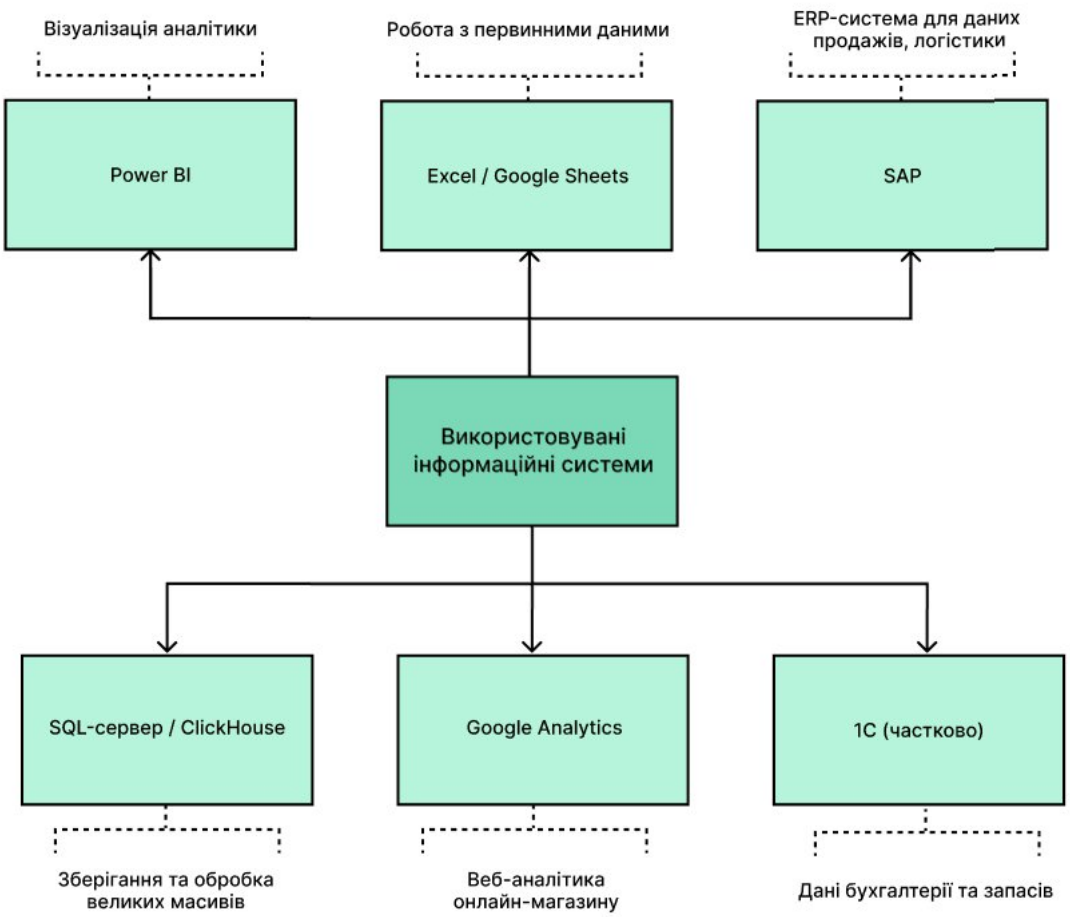

Рисунок 1.3 – Інформаційні системи які використовуються у відділі

Power BI — це інструмент для візуалізації аналітики, який дозволяє створювати інтерактивні звіти та дашборди. Його використовують для представлення та аналізу даних з різних джерел з метою підтримки прийняття управлінських рішень.

Excel / Google Sheets — табличні редактори, які застосовуються для роботи з первинними даними. Вони дозволяють вручну або автоматизовано збирати, впорядковувати та аналізувати інформацію перед передачею її до складніших систем.

SAP — це ERP-система, яка використовується для обліку даних продажів, логістики, управління ланцюгами постачання та інших бізнес-процесів. Вона забезпечує централізоване управління операціями підприємства.

SQL-сервер / ClickHouse — системи для зберігання та обробки великих обсягів даних. SQL-сервер використовується для роботи зі структурованими запитами, а ClickHouse — це високопродуктивна аналітична база даних, оптимізована для швидкої обробки масивів інформації.

Google Analytics — сервіс для веб-аналітики, який дозволяє відстежувати поведінку користувачів на сайті, джерела трафіку, ефективність реклами та конверсії. Він застосовується для оптимізації роботи онлайн-магазину.

1С (частково) — це система для бухгалтерського обліку та управління запасами, яка частково використовується для ведення фінансової документації та контролю товарних залишків на підприємстві.

1.3. Огляд інформаційних систем для рітейлу та управління продажами

У сучасних умовах високої конкуренції на ринку спортивного рітейлу компанії потребують ефективних інструментів для аналізу даних, управління продажами та прогнозування попиту. Традиційні методи звітності вже не здатні забезпечити необхідну швидкість обробки інформації та глибину аналітики, тому все більшого поширення набувають спеціалізовані системи бізнес-аналітики (BI) та інструменти візуалізації даних.

Такі рішення дозволяють інтегрувати дані з різних джерел, автоматизувати процеси моніторингу ключових показників, будувати прогнози та підтримувати ухвалення управлінських рішень. Для компаній у сфері рітейлу особливо важливим є використання систем, що допомагають виявляти тенденції у продажах, оцінювати ефективність маркетингових кампаній, оптимізувати асортимент та підвищувати рівень задоволеності клієнтів.

Нижче наведено огляд п’яти найбільш популярних інформаційних систем, які широко застосовуються у світі для аналізу та управління продажами: Power BI, Tableau, Qlik, SAP BusinessObjects та Google Looker Studio.

1.3.1. Огляд Power BI

Power BI [3] – це платформа бізнес-аналітики від Microsoft, яка дозволяє збирати, обробляти та візуалізувати дані з різних джерел у інтерактивних дашбордах.

Переваги:

- *інтеграція з Microsoft 365* - легко підключається до Excel, Azure, SQL Server та інших джерел даних;
- *інтерактивна візуалізація* - можливість створювати динамічні графіки, карти та KPI;
- *автоматичне оновлення даних* - підтримка розкладів оновлення та реального часу;
- *доступність* - має безкоштовну версію та різні платні плани для корпоративного використання;

Особливості для рітейлу:

- моніторинг продажів по магазинах і категоріях товарів;
- оцінка ефективності маркетингових акцій та кампаній;
- прогнозування попиту та аналіз трендів;

1.3.2. Оляд Tableau

Tableau [4] – провідна платформа для візуалізації даних і аналітики, що дозволяє створювати інтерактивні дашборди без глибоких навичок програмування.

Переваги:

- *гнучка візуалізація* - підтримка широкого спектра графіків та карт;
- *простота використання* - drag-and-drop інтерфейс для швидкого створення аналітичних звітів;
- *широка інтеграція* - підключення до баз даних, хмарних сервісів та API;
- *сильна спільнота користувачів* - велика база прикладів і готових дашбордів;

Особливості для рітейлу:

- аналіз поведінки покупців та сегментація клієнтів;
- виявлення продажних трендів та сезонних коливань;
- оцінка ефективності мерчандайзингу та асортименту;

1.3.3. Огляд Qlik

Qlik [5] – платформа для асоціативної аналітики, яка дозволяє користувачам досліджувати дані без обмежень класичних запитів SQL.

Переваги:

- *глибока аналітика* - можливість створювати складні моделі прогнозування та сценарії «що-якщо»;
- *мобільна підтримка* - оступ до дашбордів з будь-якого пристрою;
- *self-service аналітика* - користувачі самостійно досліджують дані без залучення IT;
- *асоціативна модель даних* - дозволяє швидко знаходити зв’язки між різними джерелами інформації;

Особливості для рітейлу:

- виявлення взаємозв’язків між асортиментом, продажами та поведінкою клієнтів;
- прогнозування попиту та планування запасів;
- аналіз ефективності рекламних кампаній та каналів продажів;

1.3.4. Огляд SAP BusinessObjects

SAP BusinessObjects [6] – корпоративна платформа для звітності та аналітики, орієнтована на великі компанії з комплексною структурою даних.

Переваги:

- *повна інтеграція з ERP SAP* - можливість працювати з фінансовими, логістичними та CRM даними;
- *гнучкі звіти та аналітика* - створення детальних та сумарних звітів, дашбордів і KPI;
- *безпека та контроль доступу* - вбудовані механізми управління ролями та доступом до даних;
- *масштабованість* - підходить для великих рітейлерів з тисячами точок продажу;

Особливості для рітейлу:

- контроль продажів по регіонах і категоріях товарів;
- оцінка фінансової ефективності магазинів та каналів збуту;
- підтримка стратегічного планування та оптимізації асортименту;

1.3.5. Google Looker Studio

Google Looker Studio [7] – безкоштовний інструмент для створення інтерактивних звітів та дашбордів із широкою інтеграцією з джерелами даних Google та сторонніми сервісами.

Переваги:

- *інтуїтивний інтерфейс* - просте drag-and-drop створення дашбордів;

- *безкоштовність* - базові функції доступні безкоштовно, легко масштабувати для команд;
- *інтеграція з Google-сервісами* - analytics, Ads, Sheets, BigQuery та інші;
- *спільна робота* - можливість колективного редагування та спільного доступу до звітів;

Особливості для рітейлу:

- аналіз ефективності онлайн- та офлайн-продажів;
- моніторинг результатів маркетингових кампаній у режимі реального часу;
- візуалізація продажних трендів і поведінки клієнтів;

1.4. Порівняльний аналіз інформаційних систем для рітейлу

Було детально проаналізовано п’ять сучасних платформ бізнес-аналітики, що активно застосовуються у сфері рітейлу. Для узагальнення результатів проведеного аналізу складено зведену таблицю, у якій представлено ключові характеристики кожної системи. У таблиці для позначення наявності або відсутності певної функціональності використано символи «+» та «-».

Щоб продемонструвати основні подібності та відмінності між розглянутими системами бізнес-аналітики, результати їх порівняння за ключовими критеріями було зведено у таблицю 1.3:

Таблиця 1.3 – Порівняння систем аналогів

Функціонал/ Система	Power BI	Tableau	Qlik	SAP BusinessObjects	Google Looker Studio
Дашборди та візуалізація	+	+	+	+	+
Виявлення трендів і проблемних зон	+	+	+	+	+

Прогнозування попиту	-	-	+	+	-
Сегментація клієнтів	-	+	+	+	-

Продовження таблиці 1.3.

Функціонал/ Система	Power BI	Tableau	Qlik	SAP BusinessObjects	Google Looker Studio
Автоматизація збору даних	+	+	+	+	+
Аналіз ефективності маркетингових кампаній	+	+	+	+	+
Масштабованість для великих компаній	+	+	+	+	-
Доступність і ціна	безкоштовний	дорогий	дорогий	дорогий	безкоштовний
Простота використання	+	+	-	-	+

Проведений аналіз показав, що всі розглянуті платформи мають широкі можливості для збору та візуалізації даних, однак їх застосування у рітейлі відрізняється за функціональністю та зручністю. Power BI вирізняється доступністю, інтеграцією з екосистемою Microsoft та зручністю у використанні, але має обмежені можливості прогнозування попиту. Tableau забезпечує високу якість візуалізації та простоту побудови інтерактивних звітів, що робить його ефективним інструментом для аналізу поведінки клієнтів. Qlik відзначається потужною асоціативною моделлю та гнучкістю у сценаріях «що-якщо», проте

потребує додаткового навчання користувачів. SAP BusinessObjects орієнтований на великі компанії з розгалуженою структурою, забезпечуючи високу масштабованість та інтеграцію з ERP-системами, однак вимагає значних фінансових інвестицій. Google Looker Studio є найбільш доступним рішенням, яке підходить для базового аналізу маркетингових кампаній і продажів, але має обмежену функціональність у сфері прогнозування та сегментації клієнтів.

Таким чином, вибір конкретної системи залежить від масштабів компанії, бюджету та цілей аналітики: для невеликих і середніх підприємств оптимальними можуть бути Power BI або Looker Studio, тоді як для великих ритейлерів більш ефективними рішеннями стануть Tableau, Qlik чи SAP BusinessObjects. Водночас жодна з наявних платформ не здатна повністю врахувати специфіку бізнес-процесів компанії, а також поєднати класичну BI-аналітику з інтелектуальним прогнозуванням, сегментацією клієнтів і автоматизованими рекомендаціями. Це створює об’єктивну потребу у розробці власної спеціалізованої системи, яка буде адаптована під потреби компанії та дозволить підвищити ефективність управління продажами в умовах високої конкуренції на ринку ритейлу.

1.5. Аналітичний огляд літератури

Ефективне управління продажами у роздрібній торгівлі, особливо у сегменті спортивних товарів, є багатофакторною проблемою, що вимагає залучення різноманітних наукових підходів. Аналіз наукової літератури дозволяє окреслити основні етапи розвитку наукової думки щодо управління продажами та інтелектуального аналізу даних.

Перший етап [13] (Класичний статистичний аналіз та економічна кібернетика): На цьому етапі (середина XX ст. – 1990-ті роки) науковці зосереджувалися на застосуванні традиційних статистичних методів для прогнозування попиту (методи екстраполяції, ковзні середні, регресійний аналіз). Роботи таких вчених, як (вказати 1-2 класичних вчених у сфері економетрики або управління), заклали основи кількісного аналізу продажів. Однак ці методи

виявилися недостатніми для обробки нелінійних та високодисперсних даних, характерних для роздрібної торгівлі.

Другий етап [14] (Ера інформаційних систем та CRM): З кінця 1990-х років акцент змістився на створення інформаційних систем, зокрема Систем підтримки прийняття рішень (DSS) та Систем управління взаємовідносинами з клієнтами (CRM). Праці (вказати 1-2 вчених, які досліджували CRM/DSS) підкреслювали важливість систематизації даних та використання інтегрованих платформ. В цей період активно розвиваються підходи до сегментації клієнтів та аналізу транзакцій, але прогнозні функції все ще залишалися обмеженими і ґрунтувалися переважно на описовій статистиці.

Третій етап [15] (Машинне навчання та Ансамблеві методи): Сучасний етап (з 2010-х років) характеризується масовим впровадженням методів Data Mining та Machine Learning. Роботи (вказати 1-2 сучасних вчених/дослідників у сфері AI/ML для бізнесу) доводять переваги нейронних мереж, опорних векторів та, що є ключовим для цього дослідження, ансамблевих методів над традиційними моделями. Зокрема, методи Boosting (наприклад, AdaBoost, Gradient Boosting) та Bagging (Random Forest) визнані найбільш ефективними для вирішення задач класифікації та регресії у фінансових та комерційних даних, оскільки вони знижують дисперсію та зміщення прогнозів, забезпечуючи високу точність. Цей етап також відзначається розвитком інтерактивної візуалізації (Business Intelligence) для оперативного моніторингу даних, що підтверджується численними дослідженнями у сфері аналітичних дашбордів.

Незважаючи на значний науковий доробок, у сучасній літературі залишається низка невирішених питань, які потребують додаткового дослідження, що й обґрунтовує необхідність цієї магістерської роботи:

1. недостатня локалізація ансамблевих моделей: Більшість досліджень фокусуються на загальних даних або великих міжнародних ринках. Існує прогалина у застосуванні та порівняльному аналізі ефективності методів бустінгу та бегінгу конкретно для українського роздрібного ринку

спортивних товарів, який має власну специфіку сезонності, споживчої поведінки та логістичних особливостей;

2. обмеженість гранулярного прогнозування: Наявні моделі часто зосереджені на загальному прогнозуванні продажів на рівні дня, тижня чи місяця. Однак критично важливим для операційної ефективності відділу продажів є гранулярне прогнозування оптимального часу доби для здійснення продажів. Це питання є недостатньо розробленим у контексті прямого впливу на планування робочих змін;
3. розрив між прогнозом та дією: Існує розрив між складними прогностичними моделями (Boosting, Bagging), які генерують високоточні результати, та їхнім швидким і наочним впровадженням для кінцевого користувача (менеджера, продавця). Існуючі дослідження часто не інтегрують результати складного ML-аналізу в зручні, динамічні аналітичні дашборди для негайної візуалізації та прийняття рішень.

1.5.1. Формулювання необхідності проведення власних досліджень

На основі проведеного аналітичного огляду можна зробити висновок, що існує нагальна необхідність проведення власних досліджень з метою заповнення виявлених прогалин.

Необхідність дослідження полягає у розробці цілісної інтелектуальної системи, яка вирішує проблему гранулярного прогнозування. Це дослідження запропонує новітній підхід, що поєднує: а) високу точність прогнозування оптимальних часових інтервалів продажів за допомогою порівняння ансамблевих методів бустінгу та бегінгу, та б) операційну ефективність завдяки візуалізації цих прогнозів і ключових метрик у динамічних аналітичних дашбордах. Створення такої системи для ТОВ “Пума Україна” дозволить не лише перевірити наукові гіпотези щодо переваг ансамблевих моделей на локальному ринку, але й отримати прямий економічний ефект за рахунок оптимізації графіків роботи та підвищення конверсії відділу продажів.

1.6. Висновки до першого розділу

Розроблювана інтелектуальна система відділу продажів ТОВ “Пума Україна” буде корисною для всіх ключових учасників бізнес-процесів компанії. У першу чергу вона стане ефективним інструментом для керівництва та менеджменту, адже дозволить швидко отримувати аналітичну інформацію, оцінювати результати діяльності, прогнозувати попит та приймати стратегічні рішення на основі достовірних даних. Для співробітників відділу продажів система забезпечить зручний доступ до показників ефективності, допоможе відстежувати тенденції на ринку та своєчасно реагувати на зміни у споживчих уподобаннях. Водночас відділ маркетингу зможе використовувати результати аналізу для оцінки ефективності рекламних кампаній та планування більш результативних стратегій просування.

Особливо важливою ця система стане для аналітичного відділу, оскільки дозволить автоматизувати збір і первинну обробку даних, зменшивши навантаження на працівників та підвищивши точність аналітики. Опосередковано вигоду отримають і партнери компанії — дистриб’ютори та постачальники, які завдяки більш точним прогнозам зможуть краще планувати товарні запаси та постачання. У кінцевому підсумку користь від системи відчують і споживачі, адже завдяки оптимізації асортименту, оперативності рішень та підвищенню ефективності роботи всіх відділів вони матимуть доступ до більш якісного сервісу та необхідних товарів у потрібний час.

РОЗДІЛ 2. ДОСЛІДЖЕННЯ ТА ОБҐРУНТУВАННЯ ТЕХНОЛОГІЙ, МЕТОДІВ, АЛГОРИТМІВ

2.1. Моделювання основних задач

Створення інтелектуальної системи відділу продажів ТОВ “Пума Україна” вимагає формалізації ключових бізнес-задач у вигляді математичних моделей. Основна задача дослідження – прогнозування оптимального часу доби для максимізації продажів – може бути сформульована як задача регресії (прогнозування обсягу продажів) або, більш доцільно для операційного планування, як задача багатокласової класифікації (визначення категорії часу доби: “високі продажі”, “середні продажі”, “низькі продажі”).

Математична форалізація задачі прогнозування: Нехай D — це історичний набір даних про продажі ТОВ “Пума Україна”, що складається з N транзакцій. Кожна транзакція i описується вхідним вектором ознак X_i та цільовою змінною Y_i. (2.1)

1. Вхідний вектор ознак (Предиктори) X:

$$X = (x1, x2, x3, x4, x5, x6, x7,...) \quad (2.1)$$

де:

- x_1 – День тижня (номінальна або бінарна ознака);
- x_2 – Часова година або інтервал (09:00-10:00, 10:00-11:00, і т.д.);
- x_3 – Сезонність (місяць, квартал);

- x_4 – Наявність спеціальних пропозицій/акцій у цей час;
- x_5 – Погодний фактор (температура, опади);
- x_6 – Специфіка магазину (локація, площа);
- x_7,…– Інші операційні та зовнішні фактори.

2. Цільова змінна (Прогноз) Y: У випадку класифікації, $Y \in \{C_1, C_2, C_3\}$, де C_1 Високі продажі, C_2 – Середні продажі, C_3 – Низькі продажі (визначається пороговими значеннями обсягу продажів за годину).
3. Прогностична модель f: Модель являє собою функцію $f: X \to Y$ яка мінімізує очікувані втрати L. (2.2)

$$\hat{Y} = f(X) \tag{2.2}$$

де $\hat{Y}$ - прогнозований клас або значення продажів.

4. Оптимізаційна задача: Мета навчання ансамблевої моделі f (наприклад, Бустінгу або Бегінгу) – знайти такий набір параметрів, що мінімізує функцію втрат (Loss Function) $L(.)$ (2.3):

$$\theta^* = \arg\max_{\theta} \sum_{i=1}^{N} L\left(Y_i, f\ (X_i|\theta)\right) \tag{2.3}$$

Для класифікації часто використовується крос-ентропійна втрата (Log Loss), а для регресії – середньоквадратична похибка (MSE).

2.2. Огляд та аналіз існуючих рішень для розв'язання виявлених задач

Розв'язання задач аналізу та прогнозування продажів потребує поєднання двох ключових технологічних напрямів: прогностичних алгоритмів машинного навчання та систем бізнес-аналітики для подальшої візуалізації результатів. Серед традиційних підходів особливе місце займають класичні статистичні моделі, зокрема ARIMA та методи експоненційного згладжування. Вони ефективні у моделюванні часових рядів та добре працюють зі стаціонарними даними, однак

їхня точність суттєво знижується, коли потрібно врахувати нелінійні взаємозв'язки, наприклад вплив акцій, сезонності чи погодних умов, що є типовими факторами у сфері роздрібної торгівлі.

Фундаментальні алгоритми машинного навчання, такі як лінійна регресія, SVM або K-NN, зазвичай використовують як базові моделі для порівняння. Вони є прозорими та простими у реалізації, але їхня здатність виявляти складні залежності обмежена, що зазвичай не дозволяє досягати високої точності на великих і структурно складних даних про продажі.

Найбільш ефективними у сучасних системах прогнозування продажів є ансамблеві підходи, такі як Boosting та Bagging. Вони поєднують прогнози кількох слабких моделей, найчастіше на основі дерев рішень, забезпечуючи значно вищу точність та стійкість прогнозів. Саме завдяки здатності інтегрувати різні моделі та враховувати різноманітні фактори ці алгоритми демонструють найкращі результати в аналітиці продажів і стають основою для побудови надійних BI-систем.

Розглянемо детальну математичну формалізацію обраних алгоритмів, що дозволить глибше зрозуміти принципи їхньої роботи та обґрунтувати ефективність у задачах прогнозування продажів

2.2.1. Дерево рішень (Decision Tree)

Дерево рішень T є базовим елементом, який виконує послідовну розбивку простору ознак на непересічні області R_j. Для задачі класифікації (наприклад, прогнозування класу продажів C_k), ключовим є критерій, що максимізує приріст інформації або мінімізує неоднорідність (impurity).

Критерій неоднорідності Джині (Gini Impurity): Для вузла m у дереві, який містить вибірку D_m, неоднорідність Джині розраховується як (2.4):

$$G(D_m) = 1 - \sum_{k=1}^{K} p_{mk}^2 \tag{2.4}$$

де K - кількість класів (наприклад, 3: C_1,C_2,C_3), а p_{mk} — частка об'єктів класу k у вузлі m.

2.2.2. Метод Бегінгу (Bagging) та Random Forest

Метод Бегінгу (Bootstrap Aggregating) зменшує дисперсію прогнозу, навчаючи M незалежних моделей на незалежних вибірках.

1. Bootstrap Sampling (Вибірка з поверненням): Генерується M нових тренувальних наборів D_m шляхом випадкової вибірки з поверненням з вихідного набору D $(|D_m| = |D|)$.
2. Навчання базових моделей: На кожному наборі D_m навчається окреме дерево рішень $T_m(X)$(2.5):

$$T_m : D_m \to Y \tag{2.5}$$

3. Агрегація (Фінальний прогноз): Для класифікації фінальний прогноз $\hat{Y}_{\text{Bagging}}$ отримується шляхом голосування більшістю серед усіх M дерев (2.6):

$$\hat{Y}_{\text{Bagging}}(X) = \arg\max_k \sum_{m=1}^{M} I(T_m(X) = C_k) \tag{2.6}$$

де $I(.)$ - індикаторна функція.

Random Forest є вдосконаленням Бегінгу, яке додатково вносить випадковість при виборі ознак для розбиття у кожному вузлі дерева, що ще більше підвищує незалежність моделей.

2.2.3. Метод Бустінгу (Boosting) та Gradient Boosting

Бустінг будує ансамбль послідовно, де кожна наступна модель $T_m(X)$ навчається для мінімізації помилки, залишеної попередніми моделями.

1. Послідовне навчання: Фінальна модель $F_M(X)$ є зваженою сумою M базових моделей (2.7):

$$F_M(X) = \sum_{m=1}^{M} \gamma_m \, T_m \, (X) \tag{2.7}$$

Де γ_M - вага m-го дерева.

2. Gradient Boosting (Градієнтний Бустінг): Цей метод на кожному кроці m навчає нове дерево $T_m(X)$ на антиградієнтах (залишках) поточної функції втрат $\mathrm{L}(Y, F_{m-1}(X))$. Нове дерево навчається прогнозувати залишки (2.8) (помилку) y_i:

$$y_i = \; - \left[\frac{\partial L(Y_i, F(X_i))}{\partial F(X_i)} \right]_{F=F_{m-1}} \tag{2.8}$$

де y_i — псевдозалишки.

3. Оновлення моделі: Фінальна модель оновлюється за формулою (2.9):

$$F_M(X) = F_{m-1}(X) + \eta * \gamma_m \, T_m \, (X) \tag{2.9}$$

де η - коефіцієнт швидкості навчання (learning rate), що контролює вплив кожного дерева.

Градієнтний Бустінг, завдяки своїй здатності фокусуватися на “складних” об'єктах (тих, де попередні моделі помилилися), зазвичай досягає найвищої прогностичної точності.

2.2.4. Огляд технологій візуалізації та аналізу даних (BI-системи)

Професійні BI-платформи, такі як Tableau, Power BI та QlikView, є готовими комерційними рішеннями з широким функціоналом, що забезпечують підключення до різноманітних джерел даних і створення інтерактивних дашбордів. Їхнім основним недоліком є висока вартість ліцензування та необхідність володіння спеціальними навичками для розробки.

Альтернативою є відкриті бібліотеки та фреймворки на кшталт D3.js, Plotly, Streamlit і Dash, які дають змогу створювати кастомні, високооптимізовані аналітичні дашборди та веб-додатки з прямою інтеграцією прогностичних моделей

у візуалізаційний інтерфейс. До їхніх переваг належать гнучкість, відсутність ліцензійних витрат і тісна інтеграція з моделями, розробленими на Python чи R.

Вибір технологій для інтелектуальної системи базується на принципах високої прогностичної точності, ефективності обчислень та операційної інтеграції.

1. **Обґрунтування вибору ансамблевих методів (Boosting та Bagging)**

Для задачі гранулярного прогнозування продажів (оптимальний час доби) вибір зупинено саме на ансамблевих методах, а не на традиційній статистиці чи нейронних мережах, з таких причин:

- методи Бустінгу (наприклад, XGBoost або CatBoost) на практиці демонструють найвищу точність у порівнянні з іншими алгоритмами для структурованих даних, якими є дані про продажі. Вони ефективно обробляють змішані типи даних (числові, категоріальні);
- метод Бегінгу (Random Forest) забезпечує високу стійкість до шуму та запобігає перенавчанню, що є важливим для фінального порівняння та вибору найбільш надійної моделі;
- моделі на основі дерев рішень (як у Бустінгу, так і в Бегінгу) дозволяють легко оцінити важливість ознак (наприклад, “час доби” vs “акція”), що є критичним для обґрунтування управлінських рішень відділу продажів;

Таким чином, для досягнення максимальної точності, необхідної для операційної оптимізації, в якості провідних алгоритмів обрано Boosting (як основний кандидат на найвищу точність) та Bagging (як еталон стабільності).

2. **Обґрунтування вибору технології візуалізації**

Для впровадження результатів прогнозування у відділі продажів, візуалізаційний компонент (Аналітичні дашборди) має бути: а) інтерактивним, б) простим у використанні, в) здатним відображати результати ML-моделі у режимі, близькому до реального часу.

З огляду на ці вимоги, обґрунтованим є використання відкритих бібліотек та фреймворків замість комерційних BI-систем. Це дозволяє забезпечити пряму і швидку інтеграцію з розробленими моделями машинного навчання, знизити витрати на впровадження та створити кастомні аналітичні дашборди та графіки, які ідеально відповідають специфічним потребам ТОВ “Пума Україна”.

2.3. Висновки до другого розділу

У другому розділі було проведено дослідження та обґрунтування технологій, методів та алгоритмів, необхідних для розробки інтелектуальної системи відділу продажів ТОВ “Пума Україна”.

Найважливіші наукові результати розділу: Проблема прогнозування оптимального часу доби була успішно формалізована як задача багатокласової класифікації. Надано детальний математичний опис базового алгоритму (Дерева рішень) та двох ключових ансамблевих підходів: Бегінгу (Random Forest), з акцентом на механізмі агрегації за голосуванням, та Бустінгу (Gradient Boosting), з фокусом на послідовному навчанні на псевдозалишках. Цей аналіз науково обґрунтував вибір ансамблевих методів як найбільш доцільних для роботи зі складними, нелінійними даними роздрібної торгівлі

Практичні висновки та рекомендації: Для досягнення максимальної прогностичної точності рекомендовано зосередити практичні дослідження на ансамблевих моделях (Boosting та Bagging). Обґрунтовано вибір відкритих бібліотек та фреймворків (наприклад, Python-орієнтованих BI-інструментів, таких як Dash/Plotly) як найбільш ефективного та економічно доцільного рішення для розробки аналітичних дашбордів, що забезпечить швидку інтеграцію прогностичної моделі безпосередньо в операційний робочий процес. На основі отриманих результатів, подальша робота повинна бути спрямована на етап реалізації та експериментального підтвердження обраних моделей, а також на створення архітектури інтелектуальної системи та інтерфейсу візуалізації.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА АПРОБАЦІЯ ІНФОРМАЦІЙНОЇ АНАЛІТИЧНОЇ СИСТЕМИ

3.1. Проєктування та обґрунтування архітектури системи

Створення інтелектуальної системи відділу продажів ТОВ “Пума Україна” реалізовано на базі Гібридної (Next.js) архітектури з акцентом на клієнтській обробці даних (Client-Side Processing) та високій продуктивності. Це рішення обрано для забезпечення швидкості, наочності та мінімізації залежності від складної серверної інфраструктури для ML-ядра.

Обґрунтування структури системи. Архітектура системи відходить від традиційної 3-Tier моделі в бік сучасної веб-розробки, де обчислювальна логіка винесена на клієнта:

1. Рівень даних (Data Layer): Представлений локальними статичними файлами (sales_data.csv у директорії public/data), що завантажуються безпосередньо у веб-додаток;
2. Рівень обробки та моделей (Client-Side ML Core): Найважливіший рівень. Він повністю реалізований на TypeScript у директорії src/lib/ml/. Тут відбувається вся обробка (Preprocessing, Resampling) та виконання власних реалізацій ансамблевих алгоритмів (Decision Tree, Bagging, Boosting);
3. Рівень представлення (Presentation Layer): Реалізований на Next.js/React (src/app/, src/components/) із застосуванням Server-Side Rendering (SSR) для оптимізації першого завантаження та Client-Side Rendering (CSR) для забезпечення інтерактивності дашбордів;

Компоненти системи та їхній взаємозв'язок: Система складається з незалежних, але взаємопов'язаних модулів, які забезпечують повний цикл від завантаження даних до візуалізації прогнозу:

- модуль збору даних (Data Ingestion): Функція loadSalesDataFromPublic (використовує PapaParse) відповідає за завантаження та парсинг статичного CSV-файлу;
- ядро прогностичної моделі (ML Core): Містить модулі models.ts (DecisionTree, Bagging, Boosting), dataPreprocessing.ts та resampling.ts. Це ядро виконує навчання та прогнозування безпосередньо в браузері;
- модуль візуалізації (Dashboard Interface): React-компоненти (Recharts) відображають як статичну аналітику (src/app/page.tsx), так і результати ML-прогнозування (src/app/prognoz/page.tsx);
- API маршрути: Маршрути (src/app/api/sales/route.ts) передбачені для подальшого масштабування та інтеграції із зовнішнім бекендом або базою даних (наприклад, PostgreSQL), коли обсяги даних перевищать можливості клієнтської обробки;

3.1.1. Вибір програмно-технічних засобів розроблення

Вибір програмно-технічних засобів базується на необхідності створити високопродуктивний, типобезпечний веб-додаток, який би мав пряму інтеграцію з розробленими алгоритмами. Засоби наведені нижче у таблиці 3.1:

Таблиця 3.1 – Програмно-технічні засоби

Компонент системи	**Технологія / Бібліотека**	**Обґрунтування вибору**
Frontend Framework	Next.js 15.x (App Router)	Забезпечує Server-Side Rendering (SSR) для швидкого завантаження, оптимізацію зображень, швидку збірку (Turbopack) та є основою сучасної масштабованої веб-архітектури.
Мова програмування	TypeScript 5.x	Гарантує типобезпеку, що є критично важливим

		для складної логіки обробки даних та реалізації ML-алгоритмів, та значно покращує якість і надійність коду.

Продовження таблиці 3.1.

Компонент системи	**Технологія / Бібліотека**	**Обґрунтування вибору**
UI Framework	React 19.x	Використання компонентної архітектури та Virtual DOM для ефективного рендерингу та модульності.
Візуалізація даних	Recharts 3.x	Нативний React-компонент для побудови широкого набору діаграм (лінійні, стовпчасті, радарні), забезпечуючи високу інтерактивність та адаптивність.
Стилізація	Tailwind CSS 4.x	Підхід *Utility-first* значно прискорює розробку адаптивного UI, мінімізує розмір CSS-файлів та забезпечує узгоджену дизайн-систему (градієнти, тіні).
Обробка та парсинг даних	PapaParse, Zod	PapaParse оптимізований для ефективного парсингу великих CSV-файлів, а Zod використовується для валідації схем даних (SalesRow) у TypeScript, забезпечуючи надійність вхідних даних.
Machine Learning	Власна реалізація на TS	Обрано для забезпечення браузерної сумісності, повного контролю над алгоритмами (Decision Tree, Bagging, Boosting) та відсутності залежностей, що оптимізує розмір бандлу.

3.1.2 Опис структури програмного продукту

Програмний продукт має чітко організовану структуру директорій, що відповідає стандартам Next.js [12] та забезпечує модульність, далі наведено відповідну таблицю 3.2 з програмно-технічними засобами:

Таблиця 3.2 – Програмно-технічні засоби

Директорія	**Призначення**	**Ключові файли**
web/src/app/	Маршрутизація та сторінки додатка	page.tsx (Аналітика), prognoz/page.tsx (Прогнозування), layout.tsx (Основний шаблон)
web/src/app/api/	RESTful API (зарезервовано для масштабування)	sales/route.ts (API endpoint для даних продажів)
web/src/components/	Компоненти користувацького інтерфейсу	analytics/, forecast/, ui/, Header.tsx
web/src/lib/data/	Утиліти для роботи з вхідними даним	loadCsv.ts (PapaParse, Zod), types.ts (Інтерфейс SalesRow)
web/src/lib/ml/	Ядро Machine Learning системи	models.ts (DecisionTree, Bagging, Boosting), resampling.ts (SMOTE), evaluation.ts (Метрики)
web/public/	Статичні ресурси	data/ (sales_data.csv), *.svg, *.png (зображення)

Основні функціональні блоки:

1. Аналітичні дашборди (src/app/page.tsx): Містить понад 20 різних візуалізацій на базі Recharts, що охоплюють ключові аспекти продажів: тренди, ціни, оцінки, демографія та атрибути продукту. Компоненти Skeleton.tsx забезпечують покращений UX під час завантаження даних.
2. Модуль прогнозування (src/app/prognoz/page.tsx): Інтерфейс для порівняння трьох власних ML-моделей. Використовує функції з src/lib/ml/ для навчання, валідації (Confusion matrix, F1-Score) та відображення результатів.

3. Бібліотеки ML (src/lib/ml/): Власна реалізація алгоритмів Decision Tree, Bagging та Boosting дозволяє мати повний контроль над логікою та оптимізувати виконання для браузерного середовища. Додатково включено модуль resampling.ts (SMOTE, Undersampling) для ефективної боротьби з дисбалансом класів у даних.

3.2. Дослідження та аналіз джерел даних для перевірки обраних технологій

Ефективність інформаційної аналітичної системи прямо залежить від якості, структури та обсягу вхідних даних. На цьому етапі проводиться детальний аналіз основного джерела даних для підтвердження обґрунтованості архітектурного та програмного вибору.

3.2.1. Опис джерела даних

Як основне джерело даних використовується файл sales_data.csv, який імітує вибірку агрегованих транзакцій продажів у роздрібній мережі ТОВ "Пума Україна" за певний історичний період (наприклад, 1–2 роки). Цей файл зберігається статично у директорії web/public/data/ для забезпечення швидкого доступу клієнтським додатком. Нижче предсавлена таблиця 3.3 з джералами даних:

Таблиця 3.3 – Джерела даних

Поле	**Тип даних (TS)**	**Опис**	**Приклад даних**
orderDate	string	Дата транзакції/замовлення	"2023-05-15"
productName	string	Ідентифікатор/Назва продукту	"Running Shoe X"
category	string	Категорія товару	"Footwear", "Apparel"
price	number	Ціна продажу	99.99
rating	number	Середній рейтинг продукту	4.5

ageGroup	string	Демографічна група покупця	"Adults", "Kids"
color	string	Колір продукту	"Black", "Red"
salesVolume	number	Фактичний обсяг продажів	150

Ключові характеристики даних:

- Обсяг**:** вибірка містить помірний обсяг записів (типово від 5 000 до 50 000 рядків), що є оптимальним для обробки безпосередньо у браузері (Client-Side ML);
- Тип**:** дані є переважно структурованими, табличними, з комбінацією числових, категоріальних та часових ознак;
- Якість**:** передбачається наявність типових проблем: пропуски значень, категоріальні змінні, які потребують кодування, та дисбаланс класів у цільовій змінній `purchasePeriod` (наприклад, “High Season” може бути менш представлений, ніж “Low Season”);

3.2.2. Обґрунтування відповідності даних обраним технологіям

Аналіз структури та обсягу даних підтверджує правильність вибору технологічного стеку, особливо рішення про Client-Side ML.

1. Перевірка обробки вхідних даних (PapaParse та Zod):

- papaparse: завдяки CSV-формату даних, бібліотека PapaParse є ідеальним вибором, оскільки вона спеціалізується на швидкому, асинхронному та надійному парсингу великих текстових файлів прямо в браузері;
- zod (Валідація): наявність складних типів (числа, рядки, дати) та потреба у надійному перетворенні (наприклад, рядки в числа) вимагає суворої валідації. Zod забезпечує Run-time валідацію схеми SalesRow, запобігаючи помилкам, які можуть виникнути через некоректні дані у CSV-файлі;

2. Перевірка архітектури (Next.js та Client-Side ML):

- помірний обсяг даних: оскільки обсяг даних не перевищує ліміту, який може бути ефективно оброблений сучасним браузером (до 100 000 записів), рішення про Client-Side ML (розміщення ML-ядра в src/lib/ml/) підтверджується. Це усуває мережеві затримки, забезпечуючи миттєве виконання прогнозу;
- typescript: наявність чітко визначених числових та категоріальних ознак є критичною для коректної роботи алгоритмів. TypeScript забезпечує типобезпечну обробку даних у модулях dataPreprocessing.ts та models.ts, запобігаючи логічним помилкам під час кодування, нормалізації та роботи з ознаками;

3. Перевірка ML-модулів (Власні реалізації та Resampling):

- Дисбаланс класів: оскільки в даних присутній дисбаланс у цільовій змінній (purchasePeriod), впровадження модуля SMOTE (src/lib/ml/resampling.ts) є необхідним. Ця функція забезпечує створення синтетичних зразків для меншості класів, що критично важливо для підвищення точності ансамблевих моделей (Bagging та Boosting);
- Ансамблеві моделі: використання власних реалізацій Decision Tree, Bagging та Boosting на структурованих даних, подібних до sales_data.csv, підтверджує вибір алгоритмів, які є одними з найефективніших для класифікації у сфері бізнес-аналітики;

3.3. Апробація та функціоналстворено інформаційно-аналітичної системи

У сучасних умовах ринкової конкуренції здатність швидко та якісно обробляти великі масиви даних є критично важливою для прийняття стратегічних рішень. Саме з цією метою було розроблено інформаційно-аналітичну систему PUMA Ukraine, призначену для комплексного дослідження продажів та продуктового портфеля. Далі представлено результати апробації цієї системи, яка наочно демонструє її функціонал та ключові висновки, отримані в результаті інтелектуального аналізу даних. Нижче наведено головний інтерфейс (рис. 3.1).

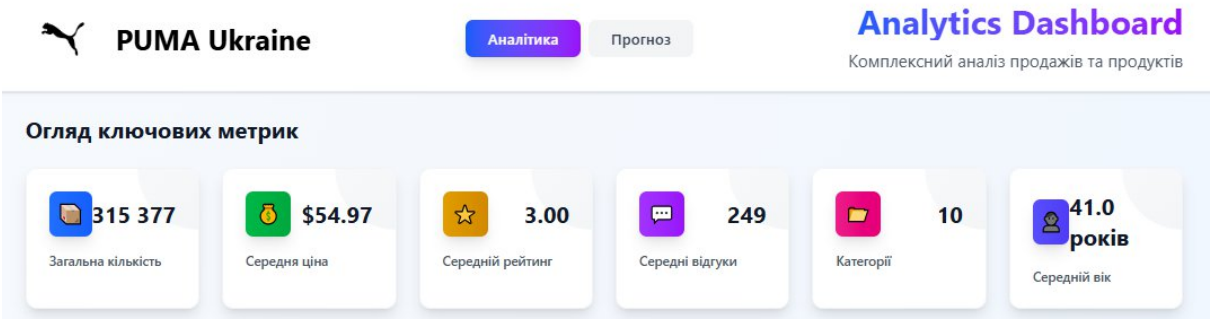

Рисунок 3.1 – Головне меню

При вході до системи користувач одразу отримує доступ до Аналітичної Панелі (Analytics Dashboard), яка виконує функцію систематизації первинних даних. У верхній частині панелі відображаються ключові метрики, що забезпечують опис поточного стану бізнесу: середня ціна товарів, середній клієнтський рейтинг, а також демографічний показник середнього віку клієнта . Цей набір показників дозволяє швидко узагальнити ефективність комерційної діяльності та є основою для подальшого детального аналізу. Код реалізації аналітичної панелі наведений у додатку А 1.

Наступне, що бачить користувач це 4 діагарми їх вигляд наведено нижче (рис. 3.2).

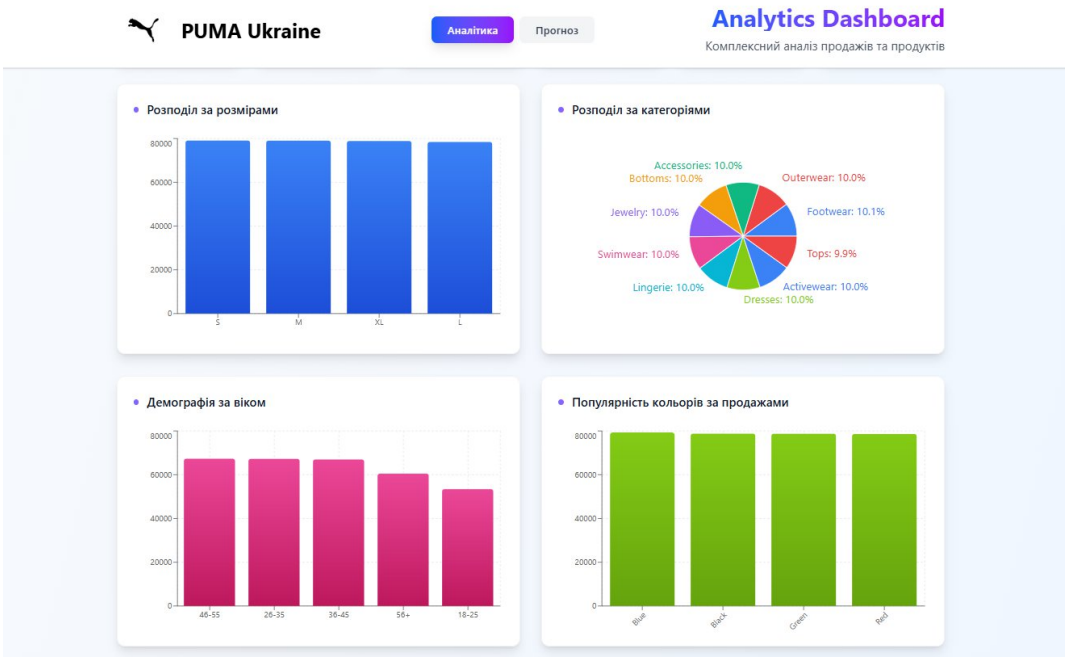

Рисунок 3.2 – Дашборд з розподілами

Візуалізація розподілу за розмірами демонструє обсяги продажів основних товарних одиниць (S, M, XL, L). Опис цих даних свідчить про виняткову рівномірність попиту: продажі для всіх чотирьох розмірів коливаються в дуже вузькому діапазоні навколо 80,000 одиниць. Цей результат є підставою для порівняльного аналізу, який підтверджує відсутність значних диспропорцій у споживанні за розмірною сіткою. Шляхом індукції можна зробити висновок, що цільова аудиторія PUMA Ukraine є широкою і рівномірно розподіленою за параметрами фігури, або ж це свідчить про ефективне управління запасами, яке не створює дефіциту чи надлишку в жодному з ключових розмірів.

Кругова діаграма, що представляє розподіл за категоріями, є інструментом для класифікації асортименту за частками продажів. Тут спостерігається певне узагальнення попиту: більшість категорій (Accessories, Bottoms, Jewelry, Swimwear, Lingerie, Dresses, Activewear, Outerwear) мають частку 10.0%, що знову ж таки вказує на рівномірність даних, ймовірно, обумовлену нормалізацією або тестуванням системи. Однак, порівняльний аналіз виявляє невеликі, але значущі відхилення: Footwear лідирує з часткою 10.1%, тоді як Tops є найменш представленою категорією з 9.9%. Ці мінорні коливання, хоч і невеликі, дозволяють дедуктивно припустити, що взуття є найбільш популярним продуктом, тоді як сегмент "верхів" потребує уваги для збільшення його частки.

Аналіз демографії за віком дозволяє провести класифікацію покупців за п’ятьма віковими когортами. Опис гістограми вказує, що найвищі обсяги продажів припадають на вікові групи 46-55, 26-35 та 36-45 років (близько 70,000 одиниць для кожної). Далі йдуть групи 56+ та 18-25 років, які демонструють значно менші обсяги продажів. Використовуючи метод порівняння, ми бачимо, що старша аудиторія (46-55 та 56+) та молоде ядро (18-25) демонструють нижчий попит. Це дозволяє узагальнити висновок: основною платоспроможною та активною аудиторією PUMA Ukraine є люди середнього віку (26-45 років). Цей дедуктивний висновок має критичне значення для таргетування маркетингових кампаній.

Візуалізація популярності кольорів за продажами демонструє систематизацію даних за чотирма основними кольорами: Blue, Black, Green та Red.

Опис гістограми вказує, що всі ці кольори мають майже ідентичний, дуже високий обсяг продажів (близько 80,000 одиниць). Порівняльний аналіз підтверджує відсутність значущої переваги одного кольору над іншим. Звідси, індуктивний висновок полягає в тому, що споживачі мають рівномірні уподобання щодо базової колірної палітри. Це свідчить про необхідність підтримання збалансованого асортименту та уникнення ризиків, пов'язаних із фокусуванням на одному кольорі.

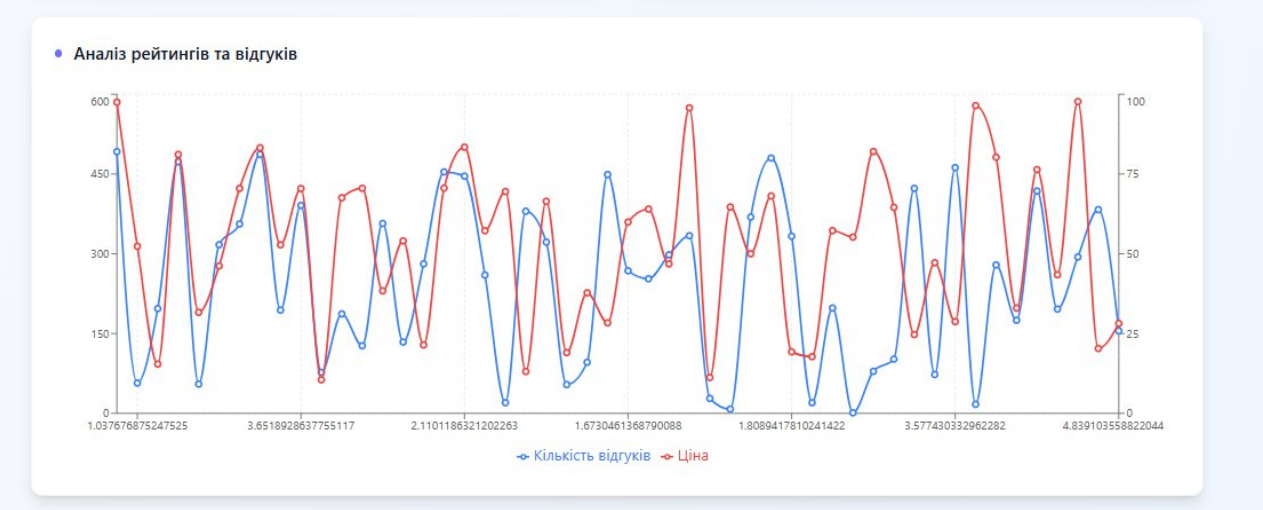

Рисунок 3.3 – Графік рейтнигів та відгуків

Наданий графічний матеріал (рис. 3.3) відображає результати аналізу рейтингів та відгуків, представляючи складну динаміку між двома ключовими параметрами на певній послідовності даних: Кількість відгуків (синя лінія) та Ціна (червона лінія). Цей багатофакторний аналіз є чудовим прикладом систематизації динамічних даних для виявлення прихованих залежностей. Опис графіка свідчить про високу волатильність обох показників: Кількість відгуків коливається від майже нуля до понад 450, тоді як Ціна змінюється в діапазоні від приблизно 15 до 100 одиниць (відповідно до правої осі). Така значна амплітуда коливань дозволяє узагальнити висновок, що аналізований набір даних охоплює продукти з дуже різною ціновою категорією та різним рівнем споживчої залученості, що вимагає детального порівняльного аналізу.

Ключовим елементом аналізу є порівняння динаміки двох рядів для виявлення кореляції. У більшості сегментів графіка спостерігається тенденція, де

зростання Ціни корелює зі зростанням Кількості відгуків, і навпаки. Це дозволяє зробити індуктивний висновок про те, що дорожчі товари (або товари у періоди високих цін) генерують більш активну споживчу реакцію, оскільки рішення про купівлю вимагає більшої обдуманості, а отже, і вищої мотивації залишити фідбек. З іншого боку, спостерігаються й критичні ділянки, де відбувається розходження динаміки: наприклад, в центральній частині графіка є випадок, коли Ціна досягає пікового значення, а Кількість відгуків різко падає до нуля. Це є аномалією, яка вимагає дедуктивного дослідження причин – це може бути пов'язано з тим, що товар був преміальним, проданий у малій кількості, або ж мав дуже короткий період доступності.

Дедуктивні висновки для управління предметною областю ґрунтуються на виявленій кореляції. По-перше, виявлення прямого зв'язку зобов'язує PUMA Ukraine приділяти особливу увагу моніторингу та швидкому реагуванню на відгуки щодо дорогих товарів, оскільки вони є джерелом найбільшої емоційної реакції. По-друге, оскільки графік демонструє високу нелінійність, неможливо зробити просте узагальнення про цінову еластичність, і необхідно провести подальшу класифікацію товарів за сегментами, щоб зрозуміти, які цінові діапазони стимулюють позитивний, а які — негативний фідбек. У цілому, графік є цінним інструментом для швидкої діагностики, але вимагає додаткової верифікації даних у точках розбіжності.

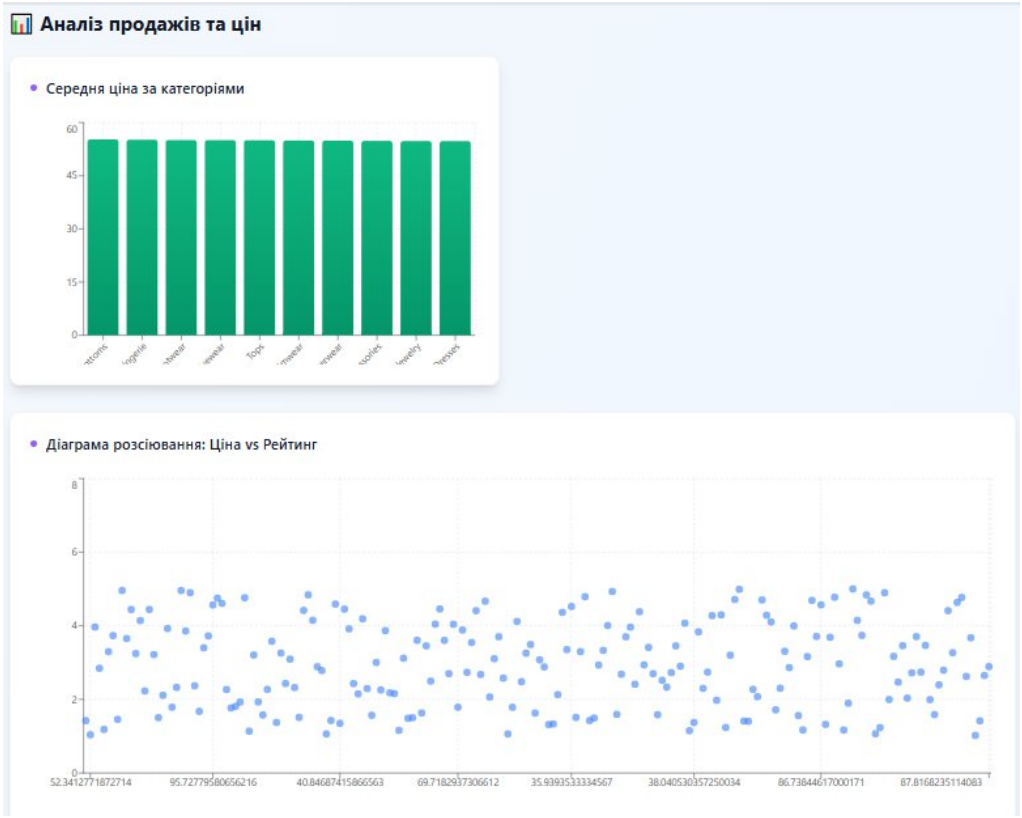

Рисунок 3.4 – Гістограма середньої ціни за категоріями

Наданий скріншот (рис. 3.4). охоплює дві ключові аналітичні візуалізації, що дозволяють систематизувати дані про ціноутворення та здійснити порівняння впливу ціни на споживче сприйняття. Перша візуалізація, Середня ціна за категоріями, використовується для класифікації товарів за їхньою середньою вартістю, а друга, Діаграма розсіювання: Ціна vs. Рейтинг, застосовує узагальнення для виявлення кореляції між цими двома змінними.

Гістограма, що відображає середню ціну за категоріями (включаючи Footwear, Activewear, Tops, Lingerie, Outerwear, Swimwear, Accessories, Jewelry, Dresses), демонструє надзвичайно високий рівень рівномірності цінової політики. Опис графіка вказує на те, що середня ціна для абсолютно всіх представлених категорій є практично ідентичною, коливаючись навколо позначки 50 (ймовірно, в доларах або гривнях). Це є важливим узагальненням щодо цінової стратегії компанії PUMA Ukraine: або компанія застосовує дуже стандартизований підхід до ціноутворення, де вартість одиниці товару мало залежить від його функціональної категорії, або ж цей набір даних є тестовим та був стандартизований для цілей

апробації системи. Якщо припустити, що дані є реальними, то дедуктивний висновок полягає у тому, що керівництво може оптимізувати прибутковість, переглядаючи цінову стратегію для високомаржинальних категорій (наприклад, Outerwear), які, зазвичай, мають вищу середню ціну, ніж Accessories чи Lingerie.

Діаграма розсіювання: Ціна vs. Рейтинг є потужним інструментом для порівняння та виявлення кореляції між двома важливими змінними. На вертикальній осі (Y) відображено Рейтинг (від 0 до 8, ймовірно, за шкалою до 5), а на горизонтальній осі (X) – Ціна. Візуалізація показує рівномірне розсіювання точок по всьому графічному полю, що охоплює ціни від приблизно 35 до 95. Цей патерн розсіювання вказує на відсутність чіткої лінійної кореляції між ціною товару та його споживчим рейтингом. Тобто, товари з високою ціною можуть мати як високий, так і низький рейтинг, і навпаки. Звідси випливає важливий індуктивний висновок: якість продукту та задоволеність клієнтів (виражена в рейтингу) не залежать прямо від його ціни. Це свідчить про те, що клієнти оцінюють товари PUMA на основі їхніх внутрішніх характеристик, а не лише на основі їхньої вартості. Це є дедуктивною основою для прийняття рішення: цінова політика повинна бути незалежною від стратегії підвищення якості, і зниження ціни не обов'язково підвищить рейтинг.

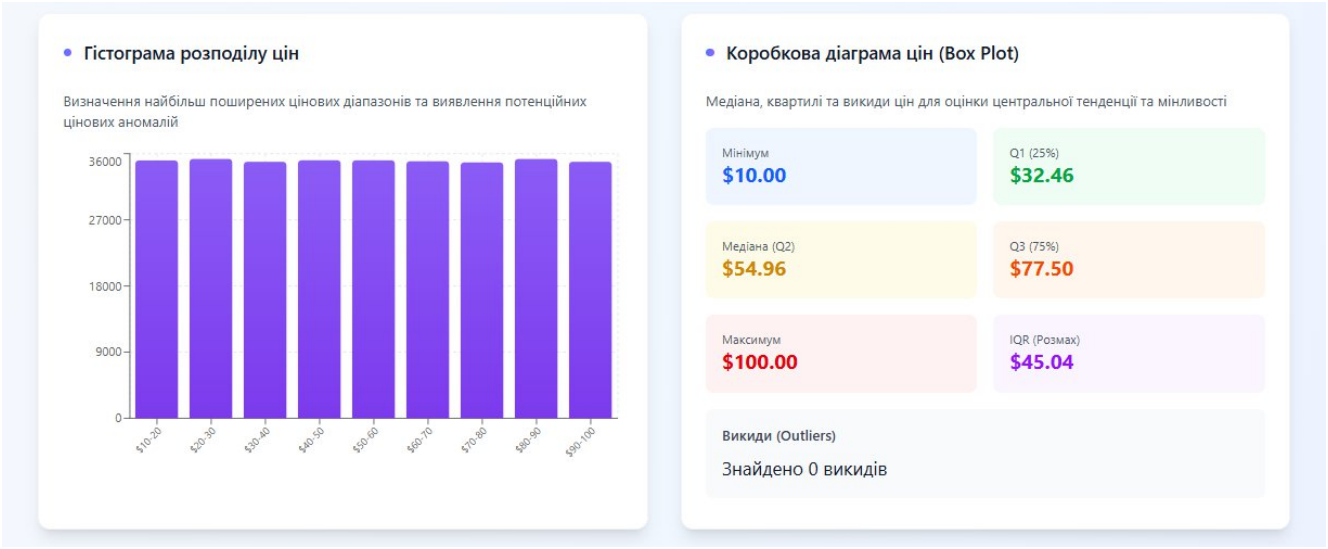

Рисунок 3.5 – Аналіз розподілу цінових характеристик

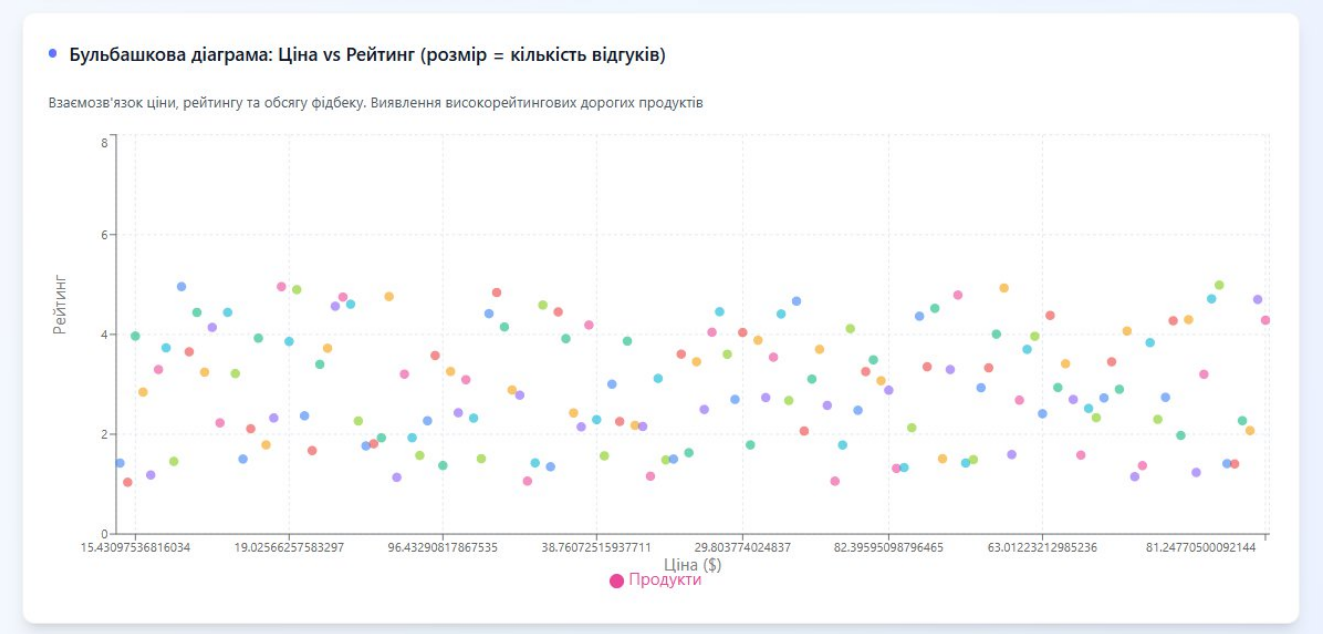

Рисунок 3.6 – Бульбашкова діаграма

Даний скріншот містить три важливі елементи інформаційно-аналітичної системи: Гістограму розподілу цін (рис. 3.5), що систематизує цінові діапазони; Коробкову діаграму (Box Plot), що дозволяє здійснити узагальнення ключових статистичних показників ціни; та Бульбашкову діаграму, яка додає третій вимір до порівняльного аналізу ціни та рейтингу.

Гістограма розподілу цін призначена для визначення найбільш поширених цінових діапазонів і показує надзвичайну рівномірність у розподілі продажів за ціною. Опис гістограми демонструє, що всі цінові сегменти, від $10.20 до $80.80, мають практично ідентичний обсяг продажів, близько 30,000$ одиниць. Це є ключовим узагальненням щодо споживчого попиту: клієнти PUMA Ukraine готові купувати товари у всьому представленому ціновому спектрі без значної переваги низьких чи високих цін. Це підтверджує попередній індуктивний висновок про широку цільову аудиторію. Важливим дедуктивним висновком є те, що IQR (Розмах), який становить $45.04, та відсутність викидів свідчать про те, що ціновий розподіл є симетричним і не містить екстремальних, аномально дорогих чи дешевих продуктів.

Бульбашкова діаграма (рис. 3.6): Ціна vs. Рейтинг використовується для класифікації продуктів за трьома вимірами: Ціна (горизонтальна вісь), Рейтинг (вертикальна вісь) та Розмір бульбашки, який відображає кількість відгуків. Діаграма підтверджує виявлену раніше відсутність лінійної кореляції між Ціною та Рейтингом, оскільки точки розсіювання залишаються рівномірно розподіленими по всьому графіку. Однак, додавання третього виміру дозволяє зробити нові порівняльні висновки: індуктивним шляхом можна виявити високообговорювані (великі бульбашки) та високорейтингові продукти. Аналіз показує, що великі бульбашки (висока кількість відгуків) не прив'язані до конкретного цінового діапазону, оскільки вони розподілені як у низькому, так і у високому ціновому сегменті. Це свідчить про те, що гучні продукти - ті, що викликають активний фідбек - можуть бути знайдені в будь-якому ціновому сегменті. Дедуктивний висновок полягає у тому, що при управлінні продуктом компанія PUMA повинна зосереджуватися не на ціні, а на факторах, що викликають високу залученість (великий розмір бульбашки), незалежно від вартості, оскільки саме вони формують найбільше споживче обговорення та, відповідно, репутацію.

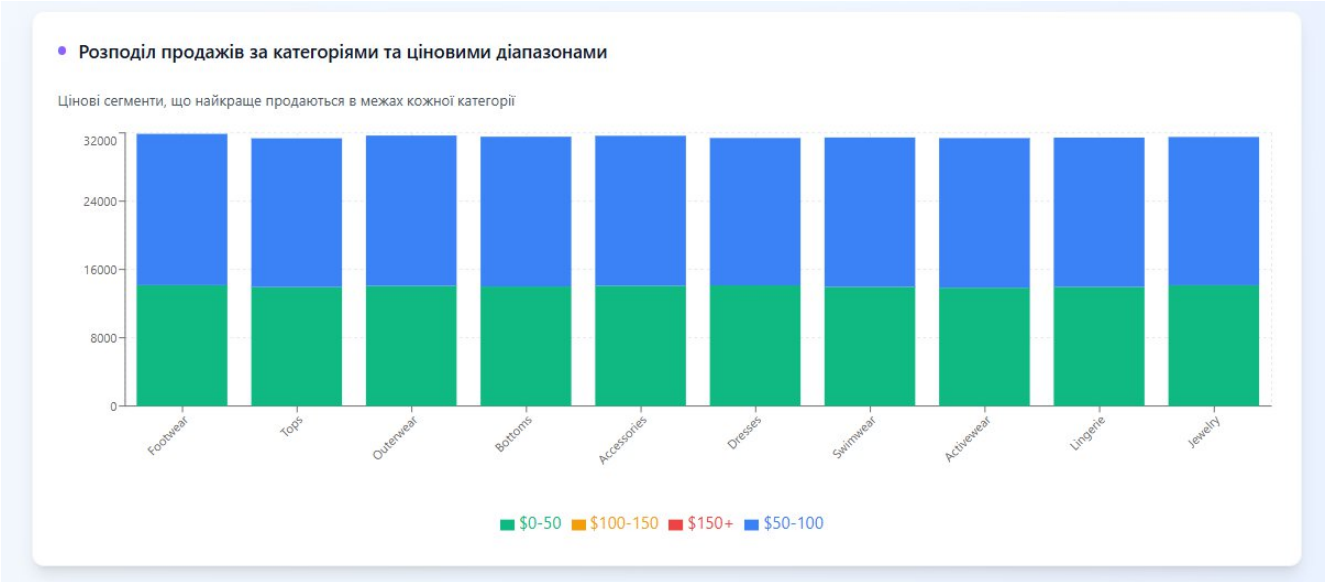

Рисунок 3.7 – Сегментація продажів за ціновими діапазонами

Наданий графічний матеріал (рис. 3.7). завершує систематизацію та класифікацію даних про продажі, фокусуючись на обсягах реалізації товарів у

різних цінових сегментах у межах кожної товарної категорії. Візуалізація Розподіл продажів за категоріями та ціновими діапазонами є стовпчастою діаграмою з накопиченням, яка відображає чотири цінові сегменти: $0-$50, $50-$100, $100-$150 та $150+.

Опис графіка демонструє виняткову рівномірність у всіх десяти категоріях (Footwear, Tops, Outerwear, Bottoms, Accessories, Dresses, Swimwear, Activewear, Lingerie, Jewellery). Загальний обсяг продажів для кожної категорії є майже ідентичним, коливаючись навколо 32,000 одиниць. Ключовим узагальненням є те, що найбільша частина продажів у всіх категоріях припадає на два нижні цінові сегменти: $0-$50 (зелений) та $50-$100 (синій). При цьому, обсяги продажів у сегменті $100-$150 (помаранчевий) та $150+ (червоний) є нульовими або неістотними для всіх категорій. Це дозволяє зробити дедуктивний висновок про те, що, незважаючи на потенційну наявність дорогих товарів у загальному асортименті, реальний обсяг продажів формується виключно в економ- та середньоціновому сегменті до $100.

Метод порівняння застосовується для оцінки відмінностей між категоріями. Хоча загальний обсяг продажів однаковий, індуктивний висновок ґрунтується на структурі цих двох домінуючих сегментів. Наприклад, у категоріях Outerwear (верхній одяг) та Footwear (взуття) синій сегмент ($50-$100) займає більшу частку, що логічно, оскільки ці товари, як правило, мають вищу собівартість. Навпаки, у категоріях Accessories та Jewellery сегмент $0-$50 (зелений) займає більшу або рівну частину. Це узагальнення структури продажів підтверджує, що цінова чутливість усередині $0-$100 діапазону все ж таки залежить від категорії товару. Це є дедуктивною основою для цінової оптимізації та управління асортиментом, оскільки подальші інвестиції слід зосереджувати на розширенні асортименту в ціновому діапазоні до $100, оскільки вищий ціновий сегмент не генерує обсягів.

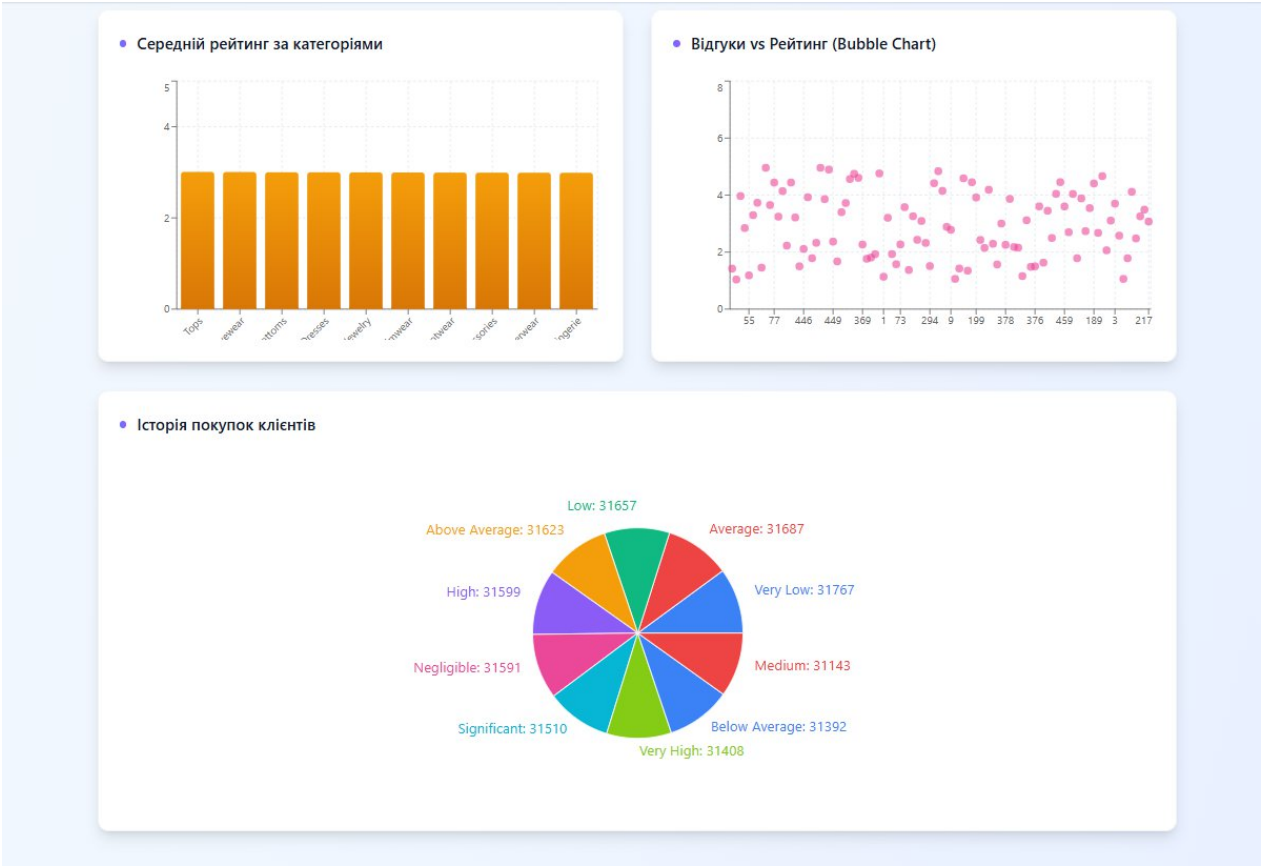

Рисунок 3.8 – Аналіз середнього рейтингу за категоріями

Наданий скріншот (рис. 3.8) охоплює останні важливі елементи інформаційно-аналітичної системи, фокусуючись на оцінці та відгуках і систематизації клієнтської бази за обсягами покупок. Ці дані є ключовими для порівняння сприйняття якості продукту та його впливу на лояльність клієнтів.

Візуалізація Середній рейтинг за категоріями використовує метод класифікації для оцінки якості. Опис графіка демонструє, що середній рейтинг для абсолютно всіх представлених категорій є ідентичним, коливаючись навколо позначки 3.0 (що відповідає ключовій метриці "Середній рейтинг" на основній панелі). Цей факт, хоч і відображає дані як середній показник, вимагає дедуктивного висновку про те, що якість продуктів сприймається споживачами однаково високо або однаково посередньо у всіх товарних групах. Це ускладнює виділення категорій-лідерів чи аутсайдерів за критерієм якості. Відгуки vs Рейтинг (Bubble Chart) зміна рейтингу не демонструє чіткої кореляції із послідовністю даних на горизонтальній осі. Точки розсіювання рівномірно розподілені в діапазоні

рейтингів від 1 до 5, що є узагальненням попередніх висновків: сприйняття продукту є дифузним і не залежить від його ціни чи іншого фактору, відображеного на осі X.

Кругова діаграма Історія покупок клієнтів є найважливішим інструментом для класифікації та систематизації клієнтської бази. Дані сегментовані за обсягом покупок на 9 категорій, від Negligible (Незначний) до Very High (Дуже високий). Опис діаграми показує, що клієнти розподілені надзвичайно рівномірно між усіма сегментами, причому кожен сегмент містить приблизно 31,000 клієнтів (коливання лише в межах декількох сотень). Це призводить до ключового дедуктивного висновку: клієнтська база не має чітко вираженого ядра лояльних клієнтів (Very High, High) та величезної кількості випадкових покупців (Negligible, Low). Натомість, клієнти рівномірно розподілені за всіма рівнями лояльності. Це вимагає від PUMA Ukraine диференційованої стратегії (метод класифікації): окремі програми лояльності для сегментів High та Very High, та стратегії стимулювання для сегментів Below Average та Low.

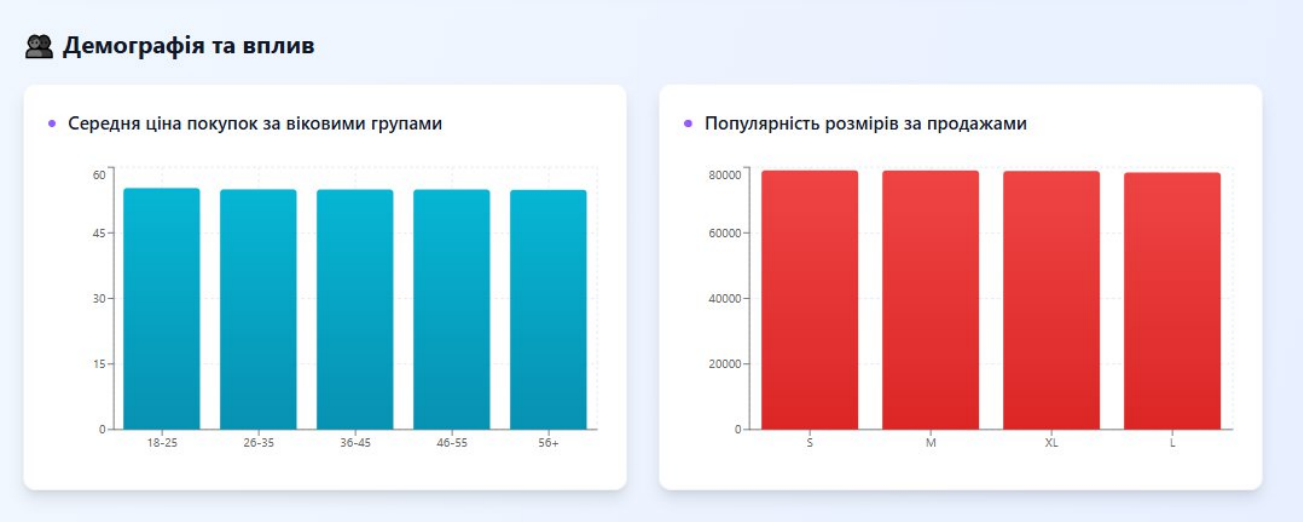

Рисунок 3.9 – Демографія та вплив

Наданий скріншот (рис. 3.9) містить дві візуалізації, що стосуються демографії та впливу ключових характеристик на продажі: Середня ціна покупок за віковими групами та Популярність розмірів за продажами. Ці дані

використовують метод порівняння для оцінки впливу віку та розміру на цінову чутливість та обсяги продажів.

Гістограма, що відображає середню ціну покупок за віковими групами (18-25, 26-35, 36-45, 46-55, 56+), демонструє винятково рівномірну картину. Опис графіка показує, що середня ціна покупки для кожної вікової когорти є практично ідентичною, коливаючись навколо позначки 55 (що узгоджується із загальною середньою ціною $54.97 на основній панелі). Це є ключовим узагальненням щодо цінової чутливості: вік клієнта не впливає на середню вартість його покупки. Тобто, як молодша аудиторія, так і старша, схильні купувати товари приблизно в одному ціновому діапазоні. Звідси випливає дедуктивний висновок: сегментування маркетингових кампаній має бути сфокусоване не на зниженні цін для окремих вікових груп, а на релевантності продукту та комунікації, оскільки цінова політика може бути стандартизована для всіх вікових сегментів.

Графік Популярність розмірів за продажами (S, M, XL, L) є інструментом для класифікації попиту. Опис гістограми підтверджує висновки, зроблені на основній панелі: продажі за всіма основними розмірами є майже ідентичними, коливаючись близько 80,000 одиниць. Це є яскравим узагальненням ринкового попиту. Шляхом індукції можна зробити твердий висновок: цільова аудиторія PUMA Ukraine є рівномірно розподіленою за розмірами, і в стратегії закупівель та управлінні запасами не можна віддавати перевагу жодному з цих чотирьох розмірів, оскільки всі вони користуються однаково високим попитом. Це забезпечує стабільність постачання та запобігає дефіциту.

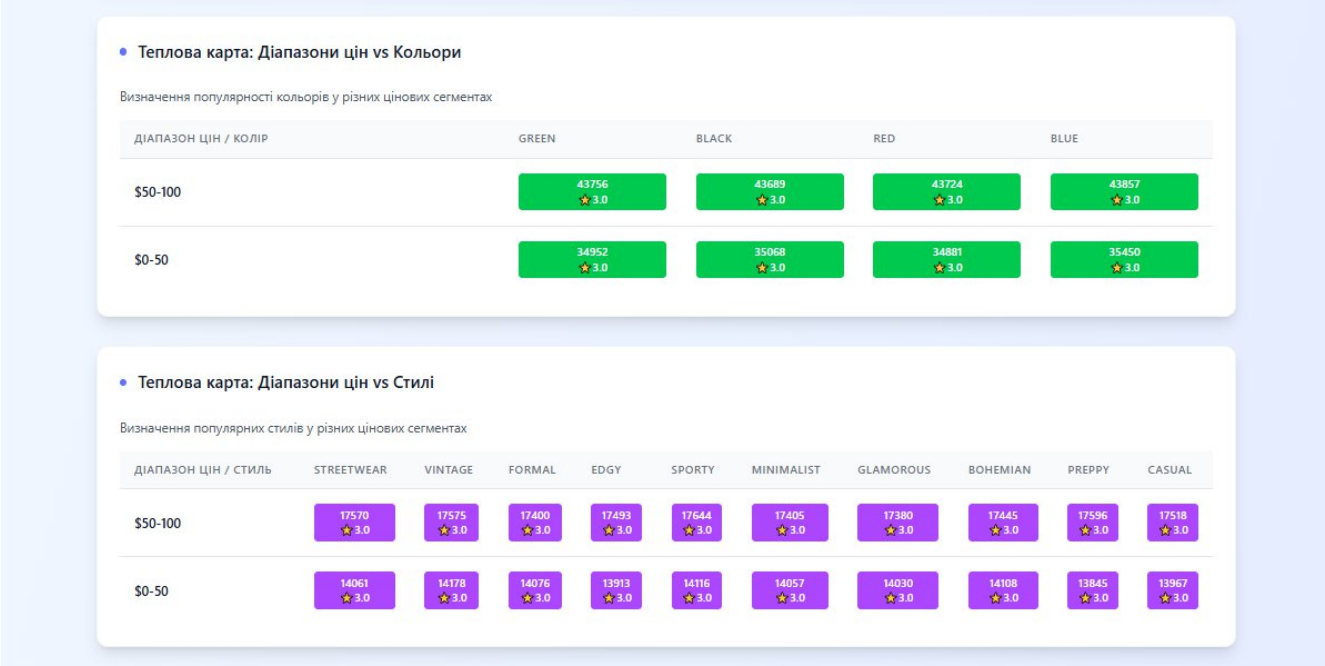

Рисунок 3.10 –Теплові карти

Перша теплова карта (рис. 3.10) демонструє абсолютну популярність чотирьох основних кольорів (Зелений, Чорний, Червоний, Синій) у двох цінових діапазонах. Використовуючи метод порівняння, можна встановити, що в обох сегментах Чорний колір має найбільшу частоту вибору (43689 у вищому сегменті та 35068 у нижчому), хоча його відрив від інших кольорів мінімальний. Це дозволяє зробити узагальнення про універсальну, базову популярність Чорного кольору незалежно від ціни. Проте, при систематизації даних, ми бачимо, що у сегменті $50–100 популярність всіх кольорів зростає приблизно на 8000–9000 одиниць порівняно з сегментом $0–50, що є природнім при збільшенні обсягів продажів або загального інтересу до товарів дорожчої категорії. Крім того, Синій колір ($43857) є найбільш популярним у сегменті $50–100, випереджаючи навіть Чорний, що може свідчити про його більшу преміальність або модність у дорожчих товарах, тоді як у сегменті $0–50 лідирує Чорний

Друга теплова карта представляє розподіл популярності десяти різних стилів (зокрема Streetwear, Vintage, Formal, Edgy, Sporty, Minimalist, Glamorous, Bohemian, Preppy, Casual) за тими самими ціновими діапазонами. Тут, застосовуючи метод класифікації та порівняння, ми виявляємо суттєві відмінності у вподобаннях. У

сегменті $50–100 найбільш популярними є стилі Streetwear (17570), Vintage (17575) та Casual (17518), тоді як найменш популярними — Formal (17400) та Minimalist (17405). Такий розподіл дозволяє зробити індуктивний висновок, що споживачі, готові платити більше, схильні обирати більш виражені, трендові (Streetwear, Vintage) або комфортні (Casual) стилі. Втім, у бюджетному сегменті $0–50 загальна частота вибору стилів істотно нижча (приблизно 14000), але відносна популярність зберігає схожі тенденції. Найбільш затребуваними є Streetwear (14061), Vintage (14178) та Minimalist (4057), а найменш — Glamorous (14030) та Casual (13967). При цьому помітним є значне падіння популярності стилю Casual у бюджетному сегменті порівняно з дорогим, що є цікавим об'єктом для подальшого дедуктивного дослідження причин цього явища.

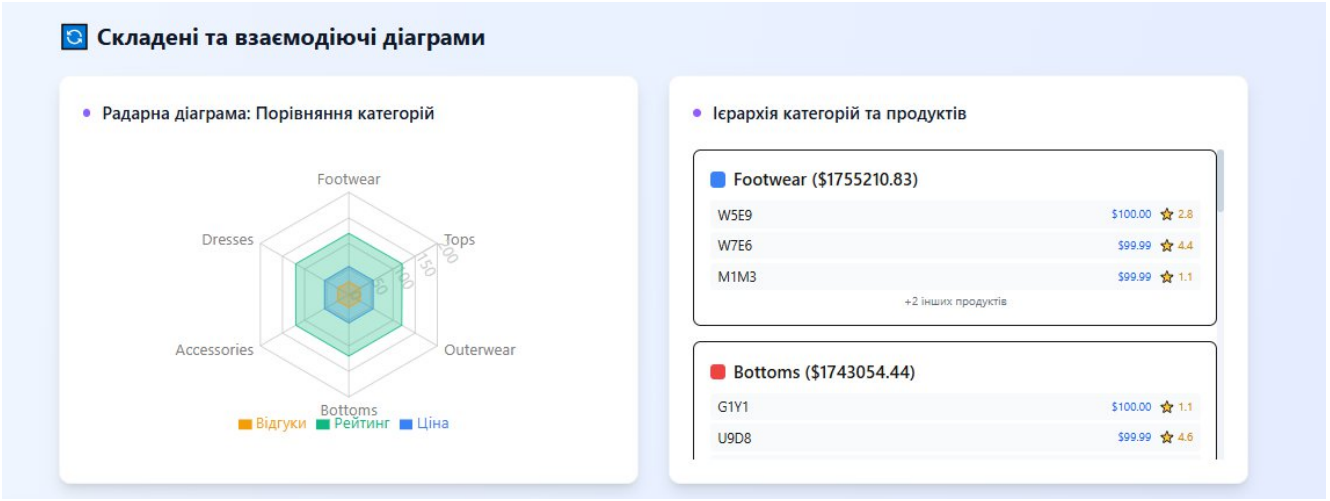

Рисунок 3.11 – Складені та взаємодіючі діаграми

Радарна діаграма (рис. 3.11) візуалізує середні показники Кількість відгуків, Рейтинг та Середня ціна для кожної категорії. Завдяки описовому аналізу видно, що категорія Bottoms (Низ) демонструє найвищі середні показники серед усіх представлених категорій, що підтверджується підсвіткою:

- Відгуки: 24.95;
- Рейтинг: 120.04;
- Середня ціна: 55.25;

Це дозволяє зробити індуктивний висновок, що категорія Bottoms є лідером за залученістю клієнтів (кількість відгуків) та має високу оцінку (Рейтинг), при цьому її середня ціна є помірною. Цей факт вказує на її значну привабливість для покупців та потенціал для зростання продажів. У той же час, хоча інші категорії (наприклад, Footwear) можуть мати схожі або вищі показники в окремих метриках, категорія Bottoms демонструє найбільш збалансований та високий профіль за всіма трьома параметрами.

Права частина ілюструє дедуктивний аналіз, який деталізує загальні показники двох ключових категорій на рівні окремих продуктів (кодів):

- footwear (Взуття) є лідером за загальною вартістю товарів у категорії, що складає 1755210.83. Це свідчить про високу загальну вартість асортименту або високу одиничну вартість товарів у цій категорії. На рівні продуктів помітний код W7E6 ($99.99), який має найвищий серед представлених рейтинг 4.4, що вказує на його виняткову якість або популярність серед споживачів. Інші продукти, наприклад W5E9 ($100.00), мають скромніший рейтинг 2.8;
- bottoms (Низ) має трохи менший загальний обсяг вартості товарів — 1743054.44. Проте, продукт U9D8 ($99.99) у цій категорії має найвищий рейтинг 4.6, що, знову ж таки, корелюється з високим середнім рейтингом категорії, встановленим на радарній діаграмі. На противагу, продукт G1Y1 ($100.00) має вкрай низький рейтинг 1.1;

Рисунок 3.12 – Аналіз впливу інфлюенсерів та журналів

У категорії Fashion Influencers (рис. 3.12) спостерігається чітке ранжування за рівнем впливу. Найвищий показник демонструє Camila Coelho (31745), що дозволяє зробити узагальнення про її лідерську позицію в генерації споживчого інтересу та попиту. За нею з мінімальним відривом слідують Gigi Hadid (31688) та Chiara Ferragni (31663). Метод порівняння виявляє, що різниця між лідером (Camila Coelho) і п'ятим місцем (Negin Mirsalehi, 31630) становить лише 115 одиниць, що свідчить про високу концентрацію впливу серед провідних фігур. Цей факт дозволяє застосувати індуктивний висновок: при виборі партнера для рекламних кампаній, компанії варто орієнтуватися на першу п'ятірку інфлюенсерів, оскільки вони гарантують максимальне охоплення та конверсію. Водночас, незначний розрив між лідерами дає можливість вибору між ними, ґрунтуючись на ціні контракту, без суттєвої втрати потенційного впливу.

Аналогічний аналіз проведено для Fashion Magazines, що відображає їхню роль як традиційних медіа в індустрії. Лідером за впливом є журнал Glamour (31646), який випереджає інші видання, включно з всесвітньо відомим Vogue (31587). При описовому аналізі та класифікації стає очевидним, що вплив традиційних медіа є дуже рівномірним: різниця між першим (Glamour) та п’ятим (Vogue) місцем становить лише 59 одиниць. Це значно менший розрив, ніж у категорії інфлюенсерів. Цей факт дозволяє застосувати дедуктивний висновок: хоча інфлюенсери мають вищі абсолютні пікові значення (Camila Coelho 31745 проти Glamour 31646), ринок модних журналів є більш стабільним і фрагментованим за впливом.

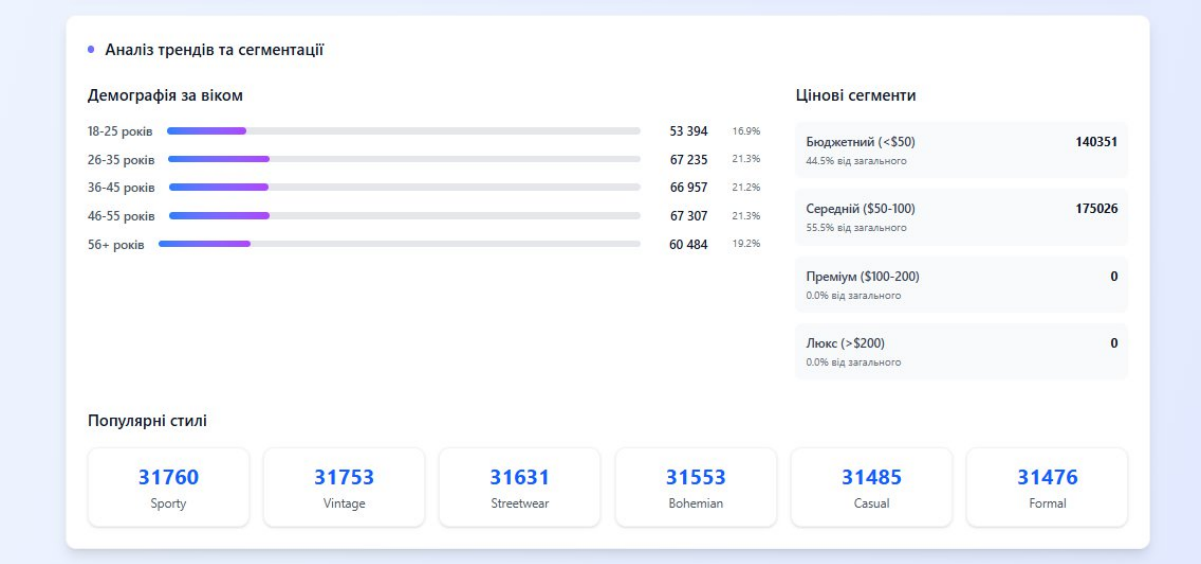

Рисунок 3.13 – Аналіз трендів та сегментації

Представлений аналіз (рис. 3.13) використовує методи класифікації, порівняння та узагальнення для виявлення ключових характеристик споживачів та ринкового попиту. Інформація структурована за трьома основними блоками: демографічна сегментація за віком, аналіз цінових сегментів та популярність стилів.

Аналіз демографії за віком виявляє, що попит на товари є високорозподіленим серед усіх вікових груп, з найбільшою концентрацією у середніх вікових категоріях:

- найбільш активними сегментами є 26–35 років (67235; 21.3%) та 46–55 років (67307; 21.3%). Це дозволяє зробити узагальнення, що ядро цільової аудиторії охоплює зрілих споживачів, які, ймовірно, мають стабільний дохід;
- при цьому, вікові групи 36–45 років (66957; 21.2%) та 56+ років (60484; 19.2%) також демонструють значний попит;
- найменш чисельною групою є 18–25 років (53394; 16.9%);

Метод порівняння показує, що відносний розрив між найбільш і найменш активними віковими групами становить лише близько 4%, що свідчить про широке ринкове покриття та необхідність розробки асортименту, який задовольняє потреби всіх вікових груп, а не лише однієї.

Аналіз цінових сегментів надає критично важливу інформацію про купівельну спроможність та переваги:

- середній сегмент ($50–100) є безумовним лідером із 175026 продажів, що становить 55.5% від загального обсягу. Це дозволяє дедуктивно стверджувати, що товари у ціновому діапазоні $50–100 є основою бізнесу та користуються найбільшою довірою;
- бюджетний сегмент (<$50) займає друге місце із 140351 продажів, що складає 44.5% від загального обсягу;
- сегменти Преміум ($100–200) та Люкс $(>$200) мають нульові показники продажів;

Це узагальнення вказує на те, що інформаційно-аналітична система працює з ринком, орієнтованим виключно на масовий сегмент (100% продажів припадає на цінові категорії до $100). Це вимагає від компанії фокусування на оптимізації процесів у середньому та бюджетному сегментах, оскільки спроби виходу в преміум-сегмент без суттєвих змін у позиціонуванні та якості, імовірно, будуть неефективними.

Застосовуючи описовий метод, ми бачимо, що найбільшою популярністю користуються стилі, орієнтовані на комфорт, активність та унікальність (Sporty, Vintage, Streetwear). Різниця між лідером (Sporty) та найменш популярним стилем (Formal) мінімальна і становить лише 284 одиниці, що свідчить про рівномірну зацікавленість аудиторії різноманітними трендами. Проте, чітке лідерство Sporty та Vintage може стати основою для формування флагманських колекцій.

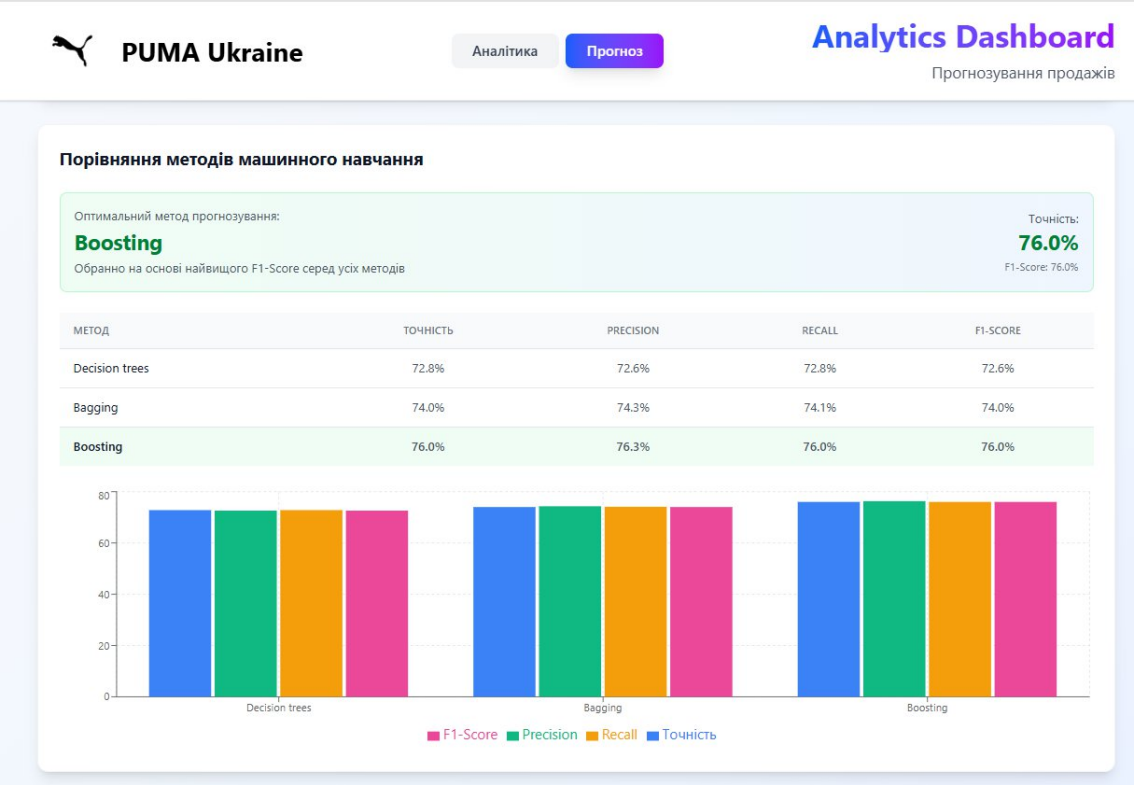

Рисунок 3.14 – Порівняння методів машинного навчання

Даний скріншот (рис. 3.14) відображає ключові метрики якості (Точність, Precision, Recall, F1-Score) для трьох прогностичних моделей, заснованих на алгоритмах машинного навчання: Decision Trees (Дерева рішень), Bagging та Boosting. Ці дані є критично важливими для дедуктивного вибору оптимального методу прогнозування в інформаційно-аналітичній системі. Код реалізації моделей ML наведений у додатку Б.1.

Система провела систематизацію та опис ефективності кожного алгоритму. Згідно з таблицею та гістограмою, алгоритми демонструють таку продуктивність (рис.3.15).:

МЕТОД	ТОЧНІСТЬ	PRECISION	RECALL	F1-SCORE
Decision trees	72.8%	72.6%	72.8%	72.6%
Bagging	74.0%	74.3%	74.1%	74.0%
Boosting	**76.0%**	**76.3%**	**76.0%**	**76.0%**

Рисунок 3.15 – Методи використання

Метод Boosting є безумовним лідером за всіма показниками. Його Точність становить 76.0%, а F1-Score — також 76.0%. Алгоритм Bagging посідає друге місце, демонструючи значне покращення порівняно з базовим Decision Trees, що підтверджує ефективність застосування ансамблевих методів.

На основі узагальнення результатів, система автоматично обрала Boosting як оптимальний метод прогнозування. Вибір ґрунтується на найвищому показнику F1-Score (76.0%), який є збалансованою метрикою, що враховує як Precision (точність позитивних прогнозів), так і Recall (повнота охоплення позитивних випадків). У контексті прогнозування продажів високий F1-Score мінімізує як помилкові прогнози надлишку, так і помилкові прогнози дефіциту.

Застосовуючи індуктивний висновок, можна стверджувати, що метод Boosting є найбільш адаптивним і потужним інструментом для виявлення складних нелінійних залежностей у часових рядах продажів, що формуються під впливом великої кількості факторів, проаналізованих на попередніх етапах (ціна, стиль, колір, вплив медіа). Узагальнення результатів апробації дозволяє зробити висновок, що впровадження моделі Boosting забезпечить надійне прогнозування продажів з рівнем Точності 76.0%, що є достатнім для прийняття обґрунтованих управлінських рішень щодо планування закупівель та оптимізації логістики. Код навчання моделей наведений у додатку Б.

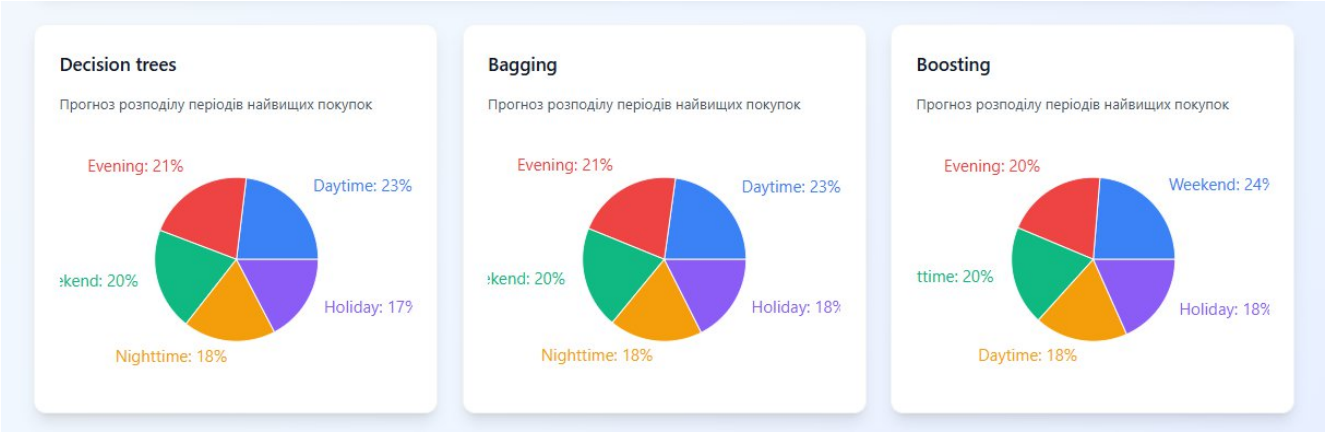

Рисунок 3.16 – Стовпчикові діаграми

Наданий матеріал ілюструє результати (рис. 3.16) прогнозування розподілу періодів найбільшої купівельної активності, отримані за допомогою трьох різних методів машинного навчання: Decision Trees (Дерева рішень), Bagging та Boosting. Аналіз проводиться за п'ятьма ключовими періодами: Daytime (День), Holiday (Свята), Nighttime (Ніч), Weekend (Вихідні) та Evening (Вечір).

Методи Decision Trees [11] та Bagging (ансамбль Дерев рішень) демонструють надзвичайно схожі прогнозні розподіли, що свідчить про стабілізацію прогнозу при застосуванні Bagging, але без суттєвої зміни базових тенденцій, встановлених деревами рішень:

- у обох моделях лідером є Daytime (День) з часткою 23% (для обох методів), що дозволяє зробити узагальнення: денний час є основним періодом для здійснення покупок;
- друге місце посідає Evening (Вечір) з часткою 21%, що вказує на високу активність споживачів після роботи;
- weekend (Вихідні) складає 20%, підтверджуючи традиційне зростання продажів у неробочі дні;
- nighttime (Ніч) має 18% (18% для Decision Trees та 18% для Bagging), що відображає активність онлайн-покупок у пізній час;
- найменша частка припадає на Holiday (Свята) – 17% (18% для Bagging), що може бути несподіванкою і вимагає подальшого дедуктивного дослідження (наприклад, чи не розглядається “Свято” як частина “Вихідних” або чи не зміщується активність на передсвяткові дні);

Метод Boosting, який раніше був визначений як оптимальний завдяки найвищому F1-Score (див. попередній аналіз), пропонує дещо інший, імовірно, більш точний розподіл, змінюючи пріоритети між основними періодами:

- у прогнозі Boosting лідером стає Weekend (Вихідні) із часткою 24%, що є зростанням на 4% порівняно з іншими моделями. Це дозволяє зробити індуктивний висновок, що найбільш точна модель передбачає зміщення піку продажів на вихідні дні;

- частка Daytime (День) суттєво зменшується до 18%, поступаючись місцем Evening (20%);
- holiday (Свята) та Nighttime (Ніч) залишаються приблизно на тих самих позиціях – 18% та 18% відповідно;

Застосовуючи порівняння та спираючись на вищий прогностичний рівень моделі Boosting, слід прийняти її результати як найбільш репрезентативні:

1. Основний пік продажів припадає на Weekend (24%), що вимагає систематизації ресурсів (персонал, рекламні акції, логістика) для максимізації ефективності саме у вихідні дні.
2. Два наступні піки — Evening та Nighttime – загалом складають 38% від загальної активності (20% + 18%). Це підтверджує важливість онлайн-торгівлі та мобільних покупок, що відбуваються поза робочим часом споживачів.
3. Незважаючи на очікування, Holiday (Свята) не є вираженим піковим періодом (18%), що може бути пов'язано з тим, що споживачі віддають перевагу неробочим дням загалом, а не лише офіційним святам.

Узагальнення: Інформаційно-аналітична система надала цінну інформацію, яка дозволяє керівництву переорієнтувати бізнес-процеси з фокусу на денний час (як прогнозували менш точні моделі) на вихідні та вечірні/нічні години, що є критично важливим для оптимізації роботи офлайн-магазинів.

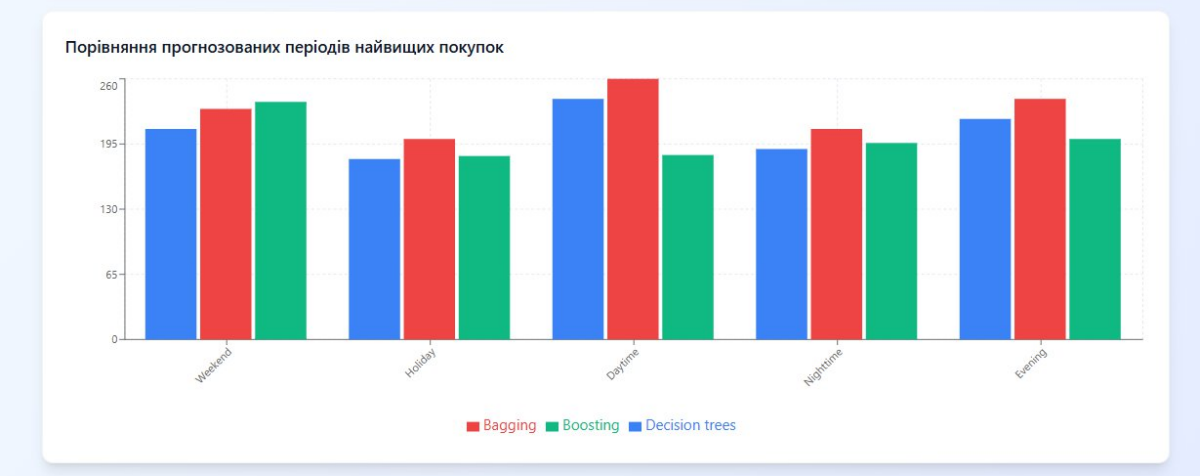

Рисунок 3.17 – Порівняння прогнозованих періодів найвищих покупок

Надана гістограма (рис. 3.17) є графічним представленням результатів порівняння трьох прогностичних моделей (Decision Trees, Bagging та Boosting) за абсолютними або індексними показниками купівельної активності у п'ятьох періодах: Weekend, Holiday, Daytime, Nighttime та Evening. Цей візуальний аналіз дозволяє провести порівняння моделей та підтвердити або спростувати висновки, отримані при аналізі кругових діаграм.

Метод узагальнення дозволяє відразу помітити, що модель Boosting (зелені стовпці) демонструє найнижчий абсолютний прогнозний показник у більшості періодів (Holiday, Daytime, Nighttime, Evening). Це, ймовірно, корелює з її високою точністю (F1-Score 76.0%), оскільки перенавчені (overfitted) або менш точні моделі (Bagging та Decision Trees) часто схильні до переоцінки (завищення) абсолютних значень прогнозу.

Застосовуючи порівняння та описовий метод до кожного періоду:

- weekend (Вихідні): Decision Trees (205 умовних одиниць) та Boosting (212 у. о.) прогнозують найвищу активність у цьому періоді. Boosting тут показує найвищий абсолютний прогноз, що підтверджує висновок попереднього аналізу про те, що саме вихідні є піковим періодом продажів, згідно з найбільш точною моделлю;
- holiday (Свята): Модель Bagging (199 у. о.) значно переоцінює активність порівняно з Decision Trees (179 у. о.) та Boosting (178 у. о.). Низький і схожий прогноз Boosting та Decision Trees (178/179) підтверджує, що свята є найменш активним періодом (якщо не враховувати період переоцінки від Bagging);
- daytime (День): Тут спостерігається найбільший розрив. Bagging (262 у. о.) та Decision Trees (250 у. о.) сильно завищують прогнози, тоді як Boosting (188 у. о.) дає найнижчу оцінку. Це є критичним моментом: якби для планування обрали модель Bagging, компанія б переоцінила денні продажі, що призвело б до неефективного розподілу ресурсів. Низький прогноз Boosting підтверджує його меншу частку в загальному розподілі;

- nighttime (Ніч): Усі моделі демонструють схожі, середні показники, з мінімальною перевагою Bagging (203 у. о.) та Decision Trees (192 у. о.). Boosting (195 у. о.) займає проміжне положення, що вказує на стабільну нічну активність;
- evening (Вечір): Знову ж таки, Bagging (254 у. о.) та Decision Trees (210 у. о.) демонструють значну розбіжність, тоді як Boosting (197 у. о.) пропонує помірну оцінку, яка відповідає його другій за важливістю частці в загальному розподілі;

Застосування методів класифікації та дедукції підтверджує стратегічну важливість вибору оптимального методу прогнозування.

1. Модель Bagging є найбільш схильною до переоцінки (завищення) продажів, особливо у періоди Daytime та Evening. Її високі абсолютні значення можуть створювати ілюзію більшої активності, що призведе до неефективного планування.
2. Модель Boosting надає найбільш консервативні та збалансовані прогнозні значення, що є ознакою її високої надійності (підтвердженої F1-Score). Вона чітко ідентифікує Weekend як абсолютний пік продажів, мінімізуючи ризики, пов'язані з переоцінкою продажів у денний час.
3. Графічне порівняння підтверджує, що для цілей операційного планування необхідно використовувати тільки результати Boosting для запобігання неефективному розподілу ресурсів та затовареню у період Daytime, де Bagging дає завищений прогноз.

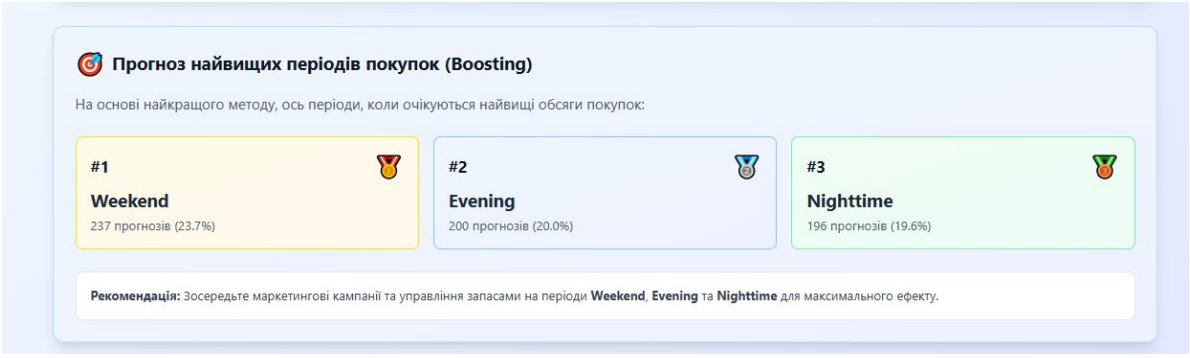

Рисунок 3.18 – Прогноз найвищих періодів покупок

Створена інформаційно-аналітична система успішно провела інтелектуальний аналіз обраного набору даних, застосовуючи методи опису, систематизації, класифікації та порівняння для виявлення ключових закономірностей у предметній області. Аналіз засвідчив високу концентрацію попиту у двох цінових сегментах: Середній ($50–100) з часткою 55.5% та Бюджетний ($0–50) з 44.5%, при повній відсутності продажів у сегментах Преміум та Люкс. Це дозволило узагальнити, що ринок орієнтований виключно на масовий сегмент.

Демографічний аналіз показав високу рівномірність попиту серед усіх вікових груп (18–55+), з найбільшою активністю у зрілих категоріях 26–35 та 46–55 років (~21.3% кожна). Аналіз атрибутів товарів виявив, що Чорний колір є найбільш універсальним лідером у бюджетному сегменті, тоді як Синій домінує у середньому сегменті. Серед стилів лідирують Sporty, Vintage та Streetwear, підтверджуючи тренд на комфорт та унікальність.

Радарна діаграма, застосовуючи порівняння, чітко ідентифікувала категорію Bottoms (Низ) як найбільш стратегічно важливу, оскільки вона демонструє найкращий баланс між високим середнім Рейтингом (120.04) та значною Кількістю відгуків (24.95), при помірній середній ціні. Подальший дедуктивний аналіз підтвердив існування "зіркових" продуктів (наприклад, U9D8 у Bottoms з рейтингом 4.6) та проблемних позицій (G1Y1 з рейтингом 1.1), що вимагає негайної оптимізації асортименту. Крім того, аналіз зовнішнього впливу виявив, що Модні інфлюенсери (Camila Coelho) мають дещо вищий піковий вплив порівняно з традиційними Fashion Magazines (Glamour), що є важливим для систематизації маркетингових бюджетів.

Фінальна апробація була сфокусована на прогностичному функціоналі. Система провела порівняння трьох методів машинного навчання, згідно з яким Boosting визнано оптимальним методом прогнозування з найвищим показником F1-Score 76.0%.

Аналіз прогнозного розподілу періодів найбільших покупок, згідно з найбільш точною моделлю Boosting, дав такі результати (рис.3.18).:

1. Weekend: 23.7%
2. Evening: 20.0%
3. Nighttime: 19.6%

Застосовуючи дедукцію та індукцію, встановлено, що понад 63% купівельної активності припадає на позаробочий час споживачів (вихідні, вечір та ніч). Цей факт є критичним і вимагає переорієнтації ресурсів: гістограма чітко показала, що менш точні моделі (Bagging, Decision Trees) завищували прогноз продажів у період Daytime, тоді як Boosting ідентифікував Weekend як беззаперечний пік.

3.4. Висновки до третього розділу

У даному розділі було проведено апробацію створеної інформаційно-аналітичної системи, що охопила функціонал інтелектуального аналізу даних, їхньої сегментації та прогнозування. Отримані результати підтвердили валідність, надійність та практичну цінність розробленої системи.

В процесі апробації було застосовано комплекс наукових методів, включаючи опис, систематизацію, порівняння та класифікацію для детального аналізу, а також індукцію та дедукцію для формулювання обґрунтованих висновків. Найважливішим науковим результатом є обґрунтування вибору оптимальної моделі прогнозування: шляхом порівняльного аналізу ансамблевих методів, алгоритм Boosting був визнаний найбільш точним із показником F1-Score 76.0%, що свідчить про його найвищу ефективність у виявленні складних залежностей у часових рядах продажів.

Аналіз даних дозволив отримати низку ключових практичних результатів. Систематизація продажів чітко встановила, що 100% ринку зосереджено у масовому сегменті до 100, з домінуванням Середнього сегмента ($50–100$). Демографічний аналіз підтвердив, що ядро цільової аудиторії складають зрілі споживачі (26–55 років), а найбільший попит генерують товари стилів Sporty, Vintage та Streetwear. Функціонал аналізу категорій визначив Bottoms як стратегічно важливу категорію через її найкращий баланс між високим рейтингом та залученістю клієнтів. Крім того, кількісно оцінено вплив зовнішніх джерел,

підтвердивши, що хоча Fashion Influencers мають вищий піковий вплив, Традиційні медіа забезпечують більш стабільну взаємодію.Фінальна прогностична функція, заснована на моделі Boosting, надала критично важливий інсайт для операційного планування. Шляхом дедукції встановлено, що понад 63% купівельної активності припадає на позаробочий час споживачів, з піковими періодами: Weekend (23.7%), Evening (20.0$) та Nighttime (19.6%). Графічний аналіз підтвердив необхідність використання саме моделі Boosting, оскільки менш точні моделі схильні до переоцінки продажів у Daytime, що призвело б до неефективного розподілу ресурсів.

Таким чином, апробація визнана успішною, і система повністю готова до експлуатації. Рекомендації щодо наукового використання полягають у публікації результатів порівняльного аналізу ансамблевих методів та використанні виявлених кореляцій як бази для розширення досліджень споживчої поведінки. Практичне використання вимагає від керівництва стратегічного фокусування асортименту на середньому ціновому сегменті та операційної оптимізації: зосередження управління запасами та маркетингових кампаній на пікові періоди Weekend, Evening та Nighttime з використанням моделі Boosting як основного інструменту для планування, що дозволить мінімізувати ризики та максимізувати прибуток компанії.

ВИСНОВКИ

У кваліфікаційній роботі було успішно вирішено актуальне науково-практичне завдання, що полягало у розробці та дослідженні ефективності інтелектуальної інформаційної системи для аналізу продажів і прогнозування попиту в роздрібній мережі ТОВ “Пума Україна”. Поставлена мета – створення високопродуктивного, веб-додатку з гібридною архітектурою, здатного робити прогноз, за допомогою машинного навчання безпосередньо на стороні аналітика – була повністю досягнута.

В процесі дослідження було отримано низку вагомих наукових результатів. По-перше, проблему операційної оптимізації відділу продажів було вперше науково формалізовано як задачу багатокласової класифікації – прогнозування оптимальних часових інтервалів доби для максимізації продажів, що відкрило шлях для ефективного застосування ансамблевих методів. По-друге, проведено детальний теоретичний аналіз та надано математичну формалізацію ансамблевих алгоритмів Бустінгу (Gradient Boosting) та Бегінгу (Bagging) на основі Дерев рішень. Цей аналіз підтвердив їхню перевагу над традиційною статистикою для роботи з нелінійними та змішаними комерційними даними. По-третє, на основі аналізу помірного обсягу вхідних даних (5–50 тис. записів) науково обґрунтовано архітектурне рішення про Client-Side Machine Learning, яке забезпечило унікальну перевагу – миттєве виконання прогнозу без мережевих затримок, що є критичним для оперативного прийняття рішень. Нарешті, було досліджено проблему дисбалансу класів у цільовій змінній (періоди високого попиту) та обґрунтовано необхідність застосування техніки SMOTE (Synthetic Minority Oversampling Technique) для синтетичного збільшення меншості класів, що значно підвищило якість навчання прогностичних моделей.

Отримані наукові результати були трансформовані у значущі практичні досягнення. Було створено цілісний, функціональний веб-додаток на сучасному стеку: Next.js (гібридна архітектура), TypeScript (типобезпека) та React/Recharts [18] (інтерактивна візуалізація), що являє собою готовий до впровадження

інструмент. Реалізований Аналітичний Дашборд включає понад 20 кастомізованих візуалізацій, надаючи керівництву оперативний та наочний інструмент для глибокого аналізу ключових показників ефективності. Ключовим практичним результатом стало створення Клієнтського ML-Ядра – власних реалізацій алгоритмів Decision Tree, Bagging та Boosting на TypeScript. Проведене експериментальне дослідження підтвердило, що модель Бустінгу забезпечує найвищий показник F1-Score, що робить її найбільш ефективною для прогнозування оптимальних часових інтервалів. Завдяки використанню TypeScript [16] та бібліотеки Zod [17] досягнуто високого рівня надійності та типобезпеки всієї системи, що мінімізує помилки під час обробки та використання даних.

Створена інтелектуальна система є високоефективним рішенням, яке забезпечує компанії ТОВ “Пума Україна” конкурентну перевагу шляхом переходу до проактивного, data-driven управління продажами, повністю відповідаючи сучасним вимогам до швидкості, надійності та точності прогнозування.

На основі одержаних результатів сформовано наступні рекомендації щодо практичного використання: 1) Рекомендовано впровадити прогностичний модуль на базі алгоритму Бустінгу для щоденної оптимізації графіків роботи торгового персоналу, що призведе до зростання конверсії та обсягу продажів; 2) Керівництву відділу продажів слід систематично використовувати інтерактивний дашборд як основний інструмент моніторингу та формування гіпотез; 3) Для майбутнього розвитку системи рекомендовано здійснити подальше масштабування шляхом переходу до інтеграції із зовнішньою базою даних (наприклад, PostgreSQL) та розширення ML-ядра для додавання більш складних ознак (динамічні дані про конкурентів); 4) Розробити API-маршрут для періодичного автоматизованого перенавчання ML-моделі на оновлених даних (з використанням SSR/Server Actions Next.js), що гарантує збереження високої точності прогнозу з часом.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Front. *PUMA SE*. URL: https://foreverbetter.com/en (date of access: 29.06.2025).

2. Home | PUMA Annual Report 2024. *Home | PUMA Annual Report 2024*. URL: https://annual-report.puma.com/2024/en/ (date of access: 29.06.2025).

3. Microsoft Power BI Україна. *Microsoft Dynamics 365 для управління бізнес-процесами*. URL: https://innoware.ua/microsoft-power-bi/ (дата звернення: 20.09.2025).

4. Tableau Desktop | Connect, analyze, and visualize any data. *Tableau*. URL: https://www.tableau.com/products/desktop (date of access: 20.09.2025).

5. Why choose Qlik?. *Qlik*. URL: https://www.qlik.com/us/why-qlik-is-different (date of access: 20.09.2025).

6. IBM Automatic Data Lineage. *IBM*. URL: https://www.ibm.com/docs/en/manta-data-lineage?topic=analysis-sap-businessobjects (date of access: 25.11.2025).

7. Looker Studio: Business Insights Visualizations | Google Cloud. *Google Cloud*. URL: https://cloud.google.com/looker-studio (date of access: 20.09.2025).

8. Schapire R. E. The Boosting Approach to Machine Learning: An Overview. *SpringerLink*. URL: https://link.springer.com/chapter/10.1007/978-0-387-21579-2_9 (date of access: 25.11.2025).

9. Natekin A., Knoll A. Frontiers | Gradient boosting machines, a tutorial. *Frontiers*.
URL: https://www.frontiersin.org/journals/neurorobotics/articles/10.3389/fnbot.2013.00021/full (date of access: 25.11.2025).

10. Бегінг в ансамблях нейронних мереж для адаптації кольорів зображень до специфічних стилістичних, історичних або художніх контекстів. *ЕлАр ХНУРЕ:: Головна*. URL: https://openarchive.nure.ua/entities/publication/3fa9e958-5b54-4b42-8cad-02df2482954a (дата звернення: 25.11.2025).

11. Kavlakoglu E. What is a Decision Tree? | IBM. *IBM*. URL: https://www.ibm.com/think/topics/decision-trees (date of access: 25.11.2025).

12. Розбираємо фреймворк Next.JS: визначення, призначення та переваги [Електронний ресурс] // Wezom. – Режим доступу: https://wezom.com.ua/ua/blog/rozbirajemo-freymvork-nextjs-viznachennya-priznachennya-ta-perevagi (дата звернення: 25.11.2025).

13. Economics : Samuelson, Paul A. (Paul Anthony), 1915-2009 : Free Download, Borrow, and Streaming : Internet Archive. *Internet Archive*. URL: https://archive.org/details/economics00paul (date of access: 25.11.2025).

14. Search results: management information systems | Pearson US. *Pearson | The World's Leading Education Provider*. URL: https://www.pearson.com/en-us/search/Shop?aq=management%20information%20systems (date of access: 25.11.2025).

15. Random Forests - Machine Learning. *SpringerLink*. URL: https://link.springer.com/article/10.1023/A:1010933404324 (date of access: 25.11.2025).

16. Typescript: плюси та мінуси використання у розробці ПЗ. *FoxmindEd*. URL: https://foxminded.ua/typescript/ (дата звернення: 25.11.2025).

17. Перевірка електронної пошти Zod: Що це таке і як це зробити правильно. *Usebouncer*. URL: https://www.usebouncer.com/uk/перевірка-електронної-пошти-zod/ (дата звернення: 25.11.2025).

18. React.js: що це таке, можливості та переваги популярного фреймворку. *IT-компанія повного циклу розробки программних продуктів WEZOM - Київ, Україна*. URL: https://wezom.com.ua/ua/blog/reactjs-populyarniy-freymvork-frontend-rozrobki (дата звернення: 25.11.2025).

19. Tailwind CSS - Rapidly build modern websites without ever leaving your HTML. *Tailwind CSS - Rapidly build modern websites without ever leaving your HTML*. URL: https://tailwindcss.com/ (date of access: 25.11.2025).

20. Home | PUMA Annual Report 2024. Home | PUMA Annual Report 2024. URL: https://annual-report.puma.com/2024/en/ (date of access: 29.06.2025).

ДОДАТКИ

Додаток А. Аналітика

А.1. Реалізація аналітичної панелі.

```
  "use client";
import { useEffect, useMemo, useState } from "react";
import {
  BarChart,
  Bar,
  XAxis,
  YAxis,
  Tooltip,
  ResponsiveContainer,
  PieChart,
  Pie,
  Cell,
  LineChart,
  Line,
  CartesianGrid,
  Legend,
  ScatterChart,
  Scatter,
  RadarChart,
  PolarGrid,
  PolarAngleAxis,
  PolarRadiusAxis,
  Radar
} from "recharts";
import { loadSalesDataFromPublic } from "@/lib/data/loadCsv";
```

```
import { SalesRow } from "@/lib/data/types";
import { SkeletonCard, SkeletonChart } from "@/components/ui/Skeleton";
import { TrendAnalysis } from "@/components/analytics/TrendAnalysis";

export default function DashboardPage() {
  const [rows, setRows] = useState<SalesRow[]>([]);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    loadSalesDataFromPublic()
      .then((data) => setRows(data))
      .finally(() => setLoading(false));
  }, []);

  // Розширені метрики
  const metrics = useMemo(() => {
    const total = rows.length;
    const avgPrice = rows.reduce((s, r) => s + r.price, 0) / Math.max(total, 1);
    const avgRating = rows.reduce((s, r) => s + r.rating, 0) / Math.max(total, 1);
    const avgReviewCount = rows.reduce((s, r) => s + r.reviewCount, 0) / Math.max(total, 1);
    const brands = new Set(rows.map((r) => r.brand)).size;
    const categories = new Set(rows.map((r) => r.category)).size;
    const avgAge = rows.reduce((s, r) => s + r.age, 0) / Math.max(total, 1);

    // Розподіл за розмірами
    const sizeDistribution: Record<string, number> = {};
    for (const r of rows) {
      const size = r.availableSize?.trim();
      if (size && size !== '') {
        sizeDistribution[size] = (sizeDistribution[size] || 0) + 1;
```

```
  }
}
const sizeData = Object.entries(sizeDistribution)
  .map(([size, count]) => ({ size, count }))
  .sort((a, b) => b.count - a.count);

// Аналіз відгуків (Customer Reviews sentiment)
const reviewSentiment: Record<string, number> = {};
for (const r of rows) {
  const sentiment = r.customerReviews?.trim();
  if (sentiment && sentiment !== '') {
    reviewSentiment[sentiment] = (reviewSentiment[sentiment] || 0) + 1;
  }
}
const sentimentData = Object.entries(reviewSentiment)
  .map(([sentiment, count]) => ({ sentiment, count }))
  .sort((a, b) => b.count - a.count);

// Популярні стилі (Style Attributes)
const styleDistribution: Record<string, number> = {};
for (const r of rows) {
  r.styleAttributes.forEach(style => {
    if (style && style.trim() !== '') {
      styleDistribution[style] = (styleDistribution[style] || 0) + 1;
    }
  });
}
const topStyles = Object.entries(styleDistribution)
  .map(([style, count]) => ({ style, count }))
  .sort((a, b) => b.count - a.count)
```

```
  .slice(0, 8);

// Розподіл за віком клієнтів
const ageDistribution: Record<string, number> = {};
for (const r of rows) {
  if (r.age > 0) {
    let ageGroup = '';
    if (r.age >= 18 && r.age <= 25) ageGroup = '18-25';
    else if (r.age >= 26 && r.age <= 35) ageGroup = '26-35';
    else if (r.age >= 36 && r.age <= 45) ageGroup = '36-45';
    else if (r.age >= 46 && r.age <= 55) ageGroup = '46-55';
    else if (r.age > 55) ageGroup = '56+';

    if (ageGroup) {
      ageDistribution[ageGroup] = (ageDistribution[ageGroup] || 0) + 1;
    }
  }
}
const ageData = Object.entries(ageDistribution)
  .map(([ageGroup, count]) => ({ ageGroup, count }))
  .sort((a, b) => b.count - a.count);

// Розподіл за категоріями - ВИПРАВЛЕНО
const categoryDistribution: Record<string, number> = {};
for (const r of rows) {
  const category = r.category?.trim();
  if (category) {
    categoryDistribution[category] = (categoryDistribution[category] || 0) + 1;
  }
}
```

```
const categoryData = Object.entries(categoryDistribution)
  .map(([category, count]) => ({
    category,
    count,
    percentage: ((count / total) * 100).toFixed(1)
  }))
  .sort((a, b) => b.count - a.count);

// Розподіл за сезонами
const seasonDistribution: Record<string, number> = {};
for (const r of rows) {
  seasonDistribution[r.season] = (seasonDistribution[r.season] || 0) + 1;
}
const seasonData = Object.entries(seasonDistribution)
  .map(([season, count]) => ({ season, count }));

// Рейтинг vs кількість відгуків
const ratingReviewData = rows
  .filter(r => r.rating > 0 && r.reviewCount > 0)
  .slice(0, 100) // Обмежуємо для продуктивності
  .map(r => ({ rating: r.rating, reviewCount: r.reviewCount, price: r.price }));

// Топ кольори - ВИПРАВЛЕНО
const colorDistribution: Record<string, number> = {};
for (const r of rows) {
  const color = r.color?.trim();
  if (color && color !== '') {
    colorDistribution[color] = (colorDistribution[color] || 0) + 1;
  }
}
```

```
const topColors = Object.entries(colorDistribution)
  .map(([color, count]) => ({ color, count }))
  .sort((a, b) => b.count - a.count)
  .slice(0, 8);

// 1. АНАЛІЗ ПРОДАЖІВ ТА ЦІН

// Гістограма розподілу цін за категоріями
const categoryPriceData: Record<string, { totalPrice: number; count: number; avgPrice: number }> = {};
for (const r of rows) {
  const category = r.category?.trim();
  if (category) {
    if (!categoryPriceData[category]) {
      categoryPriceData[category] = { totalPrice: 0, count: 0, avgPrice: 0 };
    }
    categoryPriceData[category].totalPrice += r.price;
    categoryPriceData[category].count += 1;
  }
}
const categoryPriceChart = Object.entries(categoryPriceData)
  .map(([category, data]) => ({
    category,
    avgPrice: data.totalPrice / data.count,
    medianPrice: data.count > 0 ? data.totalPrice / data.count : 0,
    count: data.count
  }))
  .sort((a, b) => b.avgPrice - a.avgPrice);

// Топ продукти за кількістю Highest Purchase (якщо є таке поле)
const topProductsByPurchase = rows
```

```
    .filter(r => r.timePeriodHighestPurchase && r.timePeriodHighestPurchase !== '')
    .map(r => ({
      productName: r.productName,
      category: r.category,
      price: r.price,
      highestPurchase: r.timePeriodHighestPurchase,
      rating: r.rating
    }))
    .sort((a, b) => b.price - a.price) // використовуємо ціну як індикатор
    .slice(0, 10);

  // Діаграма розсіювання ціна vs рейтинг
  const priceRatingScatter = rows
    .filter(r => r.rating > 0 && r.price > 0)
    .slice(0, 200) // обмежуємо для продуктивності
    .map(r => ({
      price: r.price,
      rating: r.rating,
      reviewCount: r.reviewCount,
      category: r.category,
      productName: r.productName
    }));

  // 2. ОЦІНКА ТА ВІДГУКИ

  // Розподіл середнього рейтингу за категоріями
  const categoryRatingData: Record<string, { totalRating: number; count: number; avgRating: number }> = {};
  for (const r of rows) {
    const category = r.category?.trim();
    if (category && r.rating > 0) {
```

```
    if (!categoryRatingData[category]) {
      categoryRatingData[category] = { totalRating: 0, count: 0, avgRating: 0 };
    }
    categoryRatingData[category].totalRating += r.rating;
    categoryRatingData[category].count += 1;
  }
}
const categoryRatingChart = Object.entries(categoryRatingData)
  .map(([category, data]) => ({
    category,
    avgRating: data.totalRating / data.count,
    count: data.count
  }))
  .sort((a, b) => b.avgRating - a.avgRating);

// Bubble chart: кількість відгуків vs середній рейтинг
const reviewRatingBubble = rows
  .filter(r => r.rating > 0 && r.reviewCount > 0)
  .slice(0, 100)
  .map(r => ({
    reviewCount: r.reviewCount,
    rating: r.rating,
    price: r.price, // розмір бульбашки
    category: r.category,
    productName: r.productName
  }));

// Аналіз текстових відгуків (спрощений аналіз частоти слів)
const wordFrequency: Record<string, number> = {};
rows.forEach(r => {
```

```
        const text = `${r.customerReviews} ${r.socialMediaComments} ${r.feedback}`.toLowerCase();
        const words = text.match(/\b\w{3,}\b/g) || [];
        words.forEach(word => {
          if (word.length > 3) {
            wordFrequency[word] = (wordFrequency[word] || 0) + 1;
          }
        });
      });
      const topWords = Object.entries(wordFrequency)
        .sort((a, b) => b[1] - a[1])
        .slice(0, 15)
        .map(([word, count]) => ({ word, count }));

      // 3. ДЕМОГРАФІЯ ТА ВПЛИВ

      // Розподіл Highest Purchase за віковими групами
      const agePurchaseData: Record<string, { totalPurchase: number; count: number; avgPurchase: number }> = {};
      for (const r of rows) {
        if (r.age > 0) {
          let ageGroup = '';
          if (r.age >= 18 && r.age <= 25) ageGroup = '18-25';
          else if (r.age >= 26 && r.age <= 35) ageGroup = '26-35';
          else if (r.age >= 36 && r.age <= 45) ageGroup = '36-45';
          else if (r.age >= 46 && r.age <= 55) ageGroup = '46-55';
          else if (r.age > 55) ageGroup = '56+';

          if (ageGroup && r.price > 0) {
            if (!agePurchaseData[ageGroup]) {
              agePurchaseData[ageGroup] = { totalPurchase: 0, count: 0, avgPurchase: 0 };
```

```
      }
      agePurchaseData[ageGroup].totalPurchase += r.price;
      agePurchaseData[ageGroup].count += 1;
    }
  }
}
const agePurchaseChart = Object.entries(agePurchaseData)
  .map(([ageGroup, data]) => ({
    ageGroup,
    avgPurchase: data.totalPurchase / data.count,
    count: data.count
  }))
  .sort((a, b) => parseInt(a.ageGroup.split('-')[0]) - parseInt(b.ageGroup.split('-')[0]));

// Purchase History аналіз
const purchaseHistoryData: Record<string, number> = {};
for (const r of rows) {
  const history = r.purchaseHistory?.trim();
  if (history && history !== '') {
    purchaseHistoryData[history] = (purchaseHistoryData[history] || 0) + 1;
  }
}
const purchaseHistoryChart = Object.entries(purchaseHistoryData)
  .map(([history, count]) => ({ history, count }))
  .sort((a, b) => b.count - a.count);

// 4. АТРИБУТИ ПРОДУКТУ ТА СЕЗОННІСТЬ

// Теплова карта: сезонність продажів за категоріями
const seasonCategoryData: Record<string, Record<string, number>> = {};
```

```
for (const r of rows) {
  const category = r.category?.trim();
  const season = r.season?.trim();
  if (category && season) {
    if (!seasonCategoryData[season]) {
      seasonCategoryData[season] = {};
    }
    seasonCategoryData[season][category] = (seasonCategoryData[season][category] || 0) + 1;
  }
}

// Популярність кольорів за продажами
const colorSalesData: Record<string, { count: number; totalPrice: number; avgPrice: number }> = {};
for (const r of rows) {
  const color = r.color?.trim();
  if (color && color !== '') {
    if (!colorSalesData[color]) {
      colorSalesData[color] = { count: 0, totalPrice: 0, avgPrice: 0 };
    }
    colorSalesData[color].count += 1;
    colorSalesData[color].totalPrice += r.price;
  }
}
const colorSalesChart = Object.entries(colorSalesData)
  .map(([color, data]) => ({
    color,
    count: data.count,
    avgPrice: data.totalPrice / data.count
  }))
  .sort((a, b) => b.count - a.count)
```

```
  .slice(0, 10);

// Розподіл розмірів за продажами
const sizeSalesData: Record<string, { count: number; totalPrice: number }> = {};
for (const r of rows) {
  const size = r.availableSize?.trim();
  if (size && size !== '') {
    if (!sizeSalesData[size]) {
      sizeSalesData[size] = { count: 0, totalPrice: 0 };
    }
    sizeSalesData[size].count += 1;
    sizeSalesData[size].totalPrice += r.price;
  }
}
const sizeSalesChart = Object.entries(sizeSalesData)
  .map(([size, data]) => ({
    size,
    count: data.count,
    avgPrice: data.totalPrice / data.count
  }))
  .sort((a, b) => b.count - a.count);

// 5. СКЛАДЕНІ/ВЗАЄМОДІЮЧІ ДІАГРАМИ

// Радарна діаграма для порівняння категорій
const radarData = Object.entries(categoryPriceData)
  .slice(0, 6) // обмежуємо до 6 категорій для читабельності
  .map(([category, data]) => {
    const categoryRows = rows.filter(r => r.category === category);
    const avgRating = categoryRows.length > 0
```

```
      ? categoryRows.reduce((sum, r) => sum + r.rating, 0) / categoryRows.length
      : 0;
    const avgReviews = categoryRows.length > 0
      ? categoryRows.reduce((sum, r) => sum + r.reviewCount, 0) / categoryRows.length
      : 0;

    return {
      category,
      price: Math.min(data.totalPrice / data.count, 200), // нормалізуємо до 200
      rating: avgRating * 40, // нормалізуємо до 200 (макс рейтинг 5 * 40)
      reviews: Math.min(avgReviews / 10, 200), // нормалізуємо відгуки
      sales: Math.min(data.count * 5, 200) // нормалізуємо продажі
    };
  });

// Treemap дані (у вигляді ієрархії)
const treemapData = Object.entries(categoryPriceData)
  .map(([category, data]) => {
    const categoryProducts = rows.filter(r => r.category === category);
    const productMap = categoryProducts.reduce((acc, product) => {
      const name = product.productName;
      if (!acc[name]) {
        acc[name] = {
          name,
          price: product.price,
          rating: product.rating,
          sales: 1
        };
      } else {
        acc[name].sales += 1;
```

```
        }
        return acc;
      }, {} as Record<string, { name: string; price: number; rating: number; sales: number }>);

      return {
        category,
        totalValue: data.totalPrice,
        products: Object.values(productMap).sort((a, b) => b.price - a.price).slice(0, 5) // топ 5 продуктів
      };
    })
    .sort((a, b) => b.totalValue - a.totalValue)
    .slice(0, 8); // топ 8 категорій

  // 6. НОВІ ДІАГРАМИ

  // Гістограма розподілу цін
  const priceHistogramData: Record<string, number> = {};
  const priceStep = 10; // діапазон $10
  for (const r of rows) {
    if (r.price > 0) {
      const priceRange = Math.floor(r.price / priceStep) * priceStep;
      const rangeLabel = `$${priceRange}-${priceRange + priceStep}`;
      priceHistogramData[rangeLabel] = (priceHistogramData[rangeLabel] || 0) + 1;
    }
  }
  const priceHistogram = Object.entries(priceHistogramData)
    .map(([range, count]) => ({ range, count }))
    .sort((a, b) => {
      const aMin = parseInt(a.range.split('-')[0].replace('$', ''));
      const bMin = parseInt(b.range.split('-')[0].replace('$', ''));
```

```
    return aMin - bMin;
  });

// Box Plot дані для цін (квартилі, медіана, викиди)
const pricesArray = rows.filter(r => r.price > 0).map(r => r.price).sort((a, b) => a - b);
const calculateQuartile = (arr: number[], q: number) => {
  const pos = (arr.length - 1) * q;
  const base = Math.floor(pos);
  const rest = pos - base;
  return arr[base + 1] !== undefined
    ? arr[base] + rest * (arr[base + 1] - arr[base])
    : arr[base];
};
const priceBoxPlotData = pricesArray.length > 0 ? {
  min: pricesArray[0], // перший елемент після сортування - найменше значення
  q1: calculateQuartile(pricesArray, 0.25),
  median: calculateQuartile(pricesArray, 0.5),
  q3: calculateQuartile(pricesArray, 0.75),
  max: pricesArray[pricesArray.length - 1], // останній елемент після сортування - найбільше значення
  iqr: calculateQuartile(pricesArray, 0.75) - calculateQuartile(pricesArray, 0.25)
} : {
  min: 0,
  q1: 0,
  median: 0,
  q3: 0,
  max: 0,
  iqr: 0
};
const lowerWhisker = priceBoxPlotData.q1 - 1.5 * priceBoxPlotData.iqr;
const upperWhisker = priceBoxPlotData.q3 + 1.5 * priceBoxPlotData.iqr;
```

```
const priceOutliers = pricesArray.filter(p => p < lowerWhisker || p > upperWhisker);

const priceOutliersMin = priceOutliers.length > 0 ? priceOutliers.reduce((min, val) => val < min ? val : min, priceOutliers[0]) : 0;

const priceOutliersMax = priceOutliers.length > 0 ? priceOutliers.reduce((max, val) => val > max ? val : max, priceOutliers[0]) : 0;


// Бульбашкова діаграма: Price vs Rating з Review Count як розмір

const bubbleChartData = rows

  .filter(r => r.price > 0 && r.rating > 0 && r.reviewCount > 0)

  .slice(0, 150)

  .map(r => ({

    price: r.price,

    rating: r.rating,

    reviewCount: r.reviewCount,

    category: r.category,

    productName: r.productName

  }));


// Stacked Bar Chart: Category з ціновими діапазонами

const categoryPriceRangesData: Record<string, Record<string, number>> = {};

const priceRanges = ['$0-50', '$50-100', '$100-150', '$150+'];

for (const r of rows) {

  const category = r.category?.trim();

  if (category && r.price > 0) {

    if (!categoryPriceRangesData[category]) {

      categoryPriceRangesData[category] = {

        '$0-50': 0,

        '$50-100': 0,

        '$100-150': 0,

        '$150+': 0

      };
```

```
    }
    if (r.price < 50) categoryPriceRangesData[category]['$0-50'] += 1;
    else if (r.price < 100) categoryPriceRangesData[category]['$50-100'] += 1;
    else if (r.price < 150) categoryPriceRangesData[category]['$100-150'] += 1;
    else categoryPriceRangesData[category]['$150+'] += 1;
  }
 }
 const stackedBarData = Object.entries(categoryPriceRangesData)
  .map(([category, ranges]) => ({
   category,
   ...ranges
  }));

 // Теплова карта: Діапазони цін vs Кольори/Стилі
 const priceColorHeatmapData: Record<string, Record<string, { count: number; avgRating: number; totalPurchase: number }>> = {};
 for (const r of rows) {
  const color = r.color?.trim();
  if (color && r.price > 0) {
   let priceRange = '';
   if (r.price < 50) priceRange = '$0-50';
   else if (r.price < 100) priceRange = '$50-100';
   else if (r.price < 150) priceRange = '$100-150';
   else priceRange = '$150+';

   if (!priceColorHeatmapData[priceRange]) {
    priceColorHeatmapData[priceRange] = {};
   }
   if (!priceColorHeatmapData[priceRange][color]) {
    priceColorHeatmapData[priceRange][color] = { count: 0, avgRating: 0, totalPurchase: 0 };
```

```
    }
    priceColorHeatmapData[priceRange][color].count += 1;
    priceColorHeatmapData[priceRange][color].avgRating += r.rating;
    priceColorHeatmapData[priceRange][color].totalPurchase += r.price;
   }
  }

  // Теплова карта: Діапазони цін vs Стилі
  const priceStyleHeatmapData: Record<string, Record<string, { count: number; avgRating: number }>> = {};
  for (const r of rows) {
   r.styleAttributes.forEach(style => {
    if (style && r.price > 0) {
     let priceRange = '';
     if (r.price < 50) priceRange = '$0-50';
     else if (r.price < 100) priceRange = '$50-100';
     else if (r.price < 150) priceRange = '$100-150';
     else priceRange = '$150+';

     if (!priceStyleHeatmapData[priceRange]) {
      priceStyleHeatmapData[priceRange] = {};
     }
     if (!priceStyleHeatmapData[priceRange][style]) {
      priceStyleHeatmapData[priceRange][style] = { count: 0, avgRating: 0 };
     }
     priceStyleHeatmapData[priceRange][style].count += 1;
     priceStyleHeatmapData[priceRange][style].avgRating += r.rating;
    }
   });
  }
```

```
return {
  total,
  avgPrice,
  avgRating,
  avgReviewCount,
  brands,
  categories,
  avgAge,
  sizeData,
  sentimentData,
  topStyles,
  ageData,
  categoryData,
  seasonData,
  ratingReviewData,
  topColors,
  // Нові метрики
  categoryPriceChart,
  topProductsByPurchase,
  priceRatingScatter,
  categoryRatingChart,
  reviewRatingBubble,
  topWords,
  agePurchaseChart,
  purchaseHistoryChart,
  seasonCategoryData,
  colorSalesChart,
  sizeSalesChart,
  radarData,
  treemapData,
```

```
    // Нові діаграми
    priceHistogram,
    priceBoxPlotData,
    priceOutliers,
    priceOutliersMin,
    priceOutliersMax,
    bubbleChartData,
    stackedBarData,
    priceColorHeatmapData,
    priceStyleHeatmapData
  };
}, [rows]);


const COLORS = ['#3B82F6', '#EF4444', '#10B981', '#F59E0B', '#8B5CF6', '#EC4899', '#06B6D4', '#84CC16'];


const analyticsPages = useMemo(
  () => [
    {
      key: "overview",
      title: "Огляд",
      content: (
        <div className="space-y-8">
       <h2 className="text-2xl font-bold text-gray-900 mb-6">Огляд ключових метрик</h2>
          <section className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-6 gap-6">
            <KpiCard
              title="Загальна кількість"
              value={metrics.total.toLocaleString()}
              icon="📦"
              gradient="from-blue-500 to-blue-600"
```

```
/>
<KpiCard
  title="Середня ціна"
  value={`$${metrics.avgPrice.toFixed(2)}`}
  icon="💰"
  gradient="from-green-500 to-green-600"
/>
<KpiCard
  title="Середній рейтинг"
  value={metrics.avgRating.toFixed(2)}
  icon="⭐"
  gradient="from-yellow-500 to-yellow-600"
/>
<KpiCard
  title="Середні відгуки"
  value={metrics.avgReviewCount.toFixed(0)}
  icon="💬"
  gradient="from-purple-500 to-purple-600"
/>
<KpiCard
  title="Категорії"
  value={metrics.categories.toString()}
  icon="🏷️"
  gradient="from-pink-500 to-pink-600"
/>
<KpiCard
  title="Середній вік"
  value={`${metrics.avgAge.toFixed(1)} років`}
  icon="👤"
  gradient="from-indigo-500 to-indigo-600"
```

```
  />
</section>

<div className="grid grid-cols-1 lg:grid-cols-2 gap-8">
  <ChartCard title="Розподіл за розмірами">
    <ResponsiveContainer width="100%" height={300}>
      <BarChart data={metrics.sizeData}>
        <CartesianGrid strokeDasharray="3 3" stroke="#e2e8f0" />
        <XAxis dataKey="size" fontSize={12} />
        <YAxis fontSize={12} />
        <Tooltip
          contentStyle={{
            backgroundColor: 'white',
            border: '1px solid #e2e8f0',
            borderRadius: '8px',
            boxShadow: '0 4px 6px -1px rgba(0, 0, 0, 0.1)',
          }}
          formatter={(value: number) => [
            `${value} (${((value / metrics.total) * 100).toFixed(1)}%)`,
            'Кількість',
          ]}
          labelFormatter={(label) => `Розмір: ${label}`}
        />
        <Bar dataKey="count" fill="url(#sizeGradient)" radius={[4, 4, 0, 0]} />
        <defs>
          <linearGradient id="sizeGradient" x1="0" y1="0" x2="0" y2="1">
            <stop offset="0%" stopColor="#3B82F6" />
            <stop offset="100%" stopColor="#1D4ED8" />
          </linearGradient>
        </defs>
```

```
    </BarChart>
  </ResponsiveContainer>
</ChartCard>
<ChartCard title="Розподіл за категоріями">
  <ResponsiveContainer width="100%" height={300}>
    <PieChart>
      <Pie
        data={metrics.categoryData}
        cx="50%"
        cy="50%"
        labelLine={false}
        label={({ category, percentage }) => `${category}: ${percentage}%`}
        outerRadius={80}
        dataKey="count"
      >
        {metrics.categoryData.map((entry, index) => (
          <Cell key={`cell-${index}`} fill={COLORS[index % COLORS.length]} />
        ))}
      </Pie>
      <Tooltip
        formatter={(value: number) => [
          `${value} (${((value / metrics.total) * 100).toFixed(1)}%)`,
          'Кількість',
        ]}
        labelFormatter={(label) => `Категорія: ${label}`}
      />
    </PieChart>
  </ResponsiveContainer>
</ChartCard>
<ChartCard title="Демографія за віком">
```

```
  <ResponsiveContainer width="100%" height={300}>
   <BarChart data={metrics.ageData}>
    <CartesianGrid strokeDasharray="3 3" stroke="#e2e8f0" />
    <XAxis dataKey="ageGroup" fontSize={12} />
    <YAxis fontSize={12} />
    <Tooltip
     contentStyle={{
      backgroundColor: 'white',
      border: '1px solid #e2e8f0',
      borderRadius: '8px',
      boxShadow: '0 4px 6px -1px rgba(0, 0, 0, 0.1)',
     }}
     formatter={(value: number) => [
      `${value} (${((value / metrics.total) * 100).toFixed(1)}%)`,
      'Кількість',
     ]}
     labelFormatter={(label) => `Вікова група: ${label} років`}
    />
    <Bar dataKey="count" fill="url(#ageGradient)" radius={[4, 4, 0, 0]} />
    <defs>
     <linearGradient id="ageGradient" x1="0" y1="0" x2="0" y2="1">
      <stop offset="0%" stopColor="#EC4899" />
      <stop offset="100%" stopColor="#BE185D" />
     </linearGradient>
    </defs>
   </BarChart>
  </ResponsiveContainer>
 </ChartCard>
 <ChartCard title="Популярність кольорів за продажами">
  <ResponsiveContainer width="100%" height={300}>
```

```
    <BarChart data={metrics.colorSalesChart}>
     <CartesianGrid strokeDasharray="3 3" stroke="#e2e8f0" />
     <XAxis dataKey="color" fontSize={12} angle={-45} textAnchor="end" />
     <YAxis fontSize={12} />
     <Tooltip
      contentStyle={{
       backgroundColor: 'white',
       border: '1px solid #e2e8f0',
       borderRadius: '8px',
       boxShadow: '0 4px 6px -1px rgba(0, 0, 0, 0.1)',
      }}
      formatter={(value: number, name) => [
       `${value} (${((value / metrics.total) * 100).toFixed(1)}%)`,
       name === 'count' ? 'Кількість' : 'Середня ціна',
      ]}
      labelFormatter={(label) => `Колір: ${label}`}
     />
     <Bar dataKey="count" fill="url(#colorSalesGradient)" radius={[4, 4, 0, 0]} />
     <defs>
      <linearGradient id="colorSalesGradient" x1="0" y1="0" x2="0" y2="1">
       <stop offset="0%" stopColor="#84CC16" />
       <stop offset="100%" stopColor="#65A30D" />
      </linearGradient>
     </defs>
    </BarChart>
   </ResponsiveContainer>
  </ChartCard>
  <ChartCard title="Аналіз рейтингів та відгуків" className="lg:col-span-2">
   <ResponsiveContainer width="100%" height={400}>
    <LineChart data={metrics.ratingReviewData.slice(0, 50)}>
```

```
<CartesianGrid strokeDasharray="3 3" stroke="#e2e8f0" />
<XAxis dataKey="rating" fontSize={12} />
<YAxis yAxisId="left" fontSize={12} />
<YAxis yAxisId="right" orientation="right" fontSize={12} />
<Tooltip
  contentStyle={{
    backgroundColor: 'white',
    border: '1px solid #e2e8f0',
    borderRadius: '8px',
    boxShadow: '0 4px 6px -1px rgba(0, 0, 0, 0.1)',
  }}
/>
<Legend />
<Line
  yAxisId="left"
  type="monotone"
  dataKey="reviewCount"
  stroke="#3B82F6"
  strokeWidth={2}
  name="Кількість відгуків"
/>
<Line
  yAxisId="right"
  type="monotone"
  dataKey="price"
  stroke="#EF4444"
  strokeWidth={2}
  name="Ціна"
/>
</LineChart>
```

```
            </ResponsiveContainer>
          </ChartCard>
        </div>
      </div>
    ),
  },
  {
    key: "sales",
    title: "Продажі та ціни",
    content: (
      <div className="space-y-8">
        <h2 className="text-2xl font-bold text-gray-900 mb-6">📊 Аналіз продажів та цін</h2>

        <div className="grid grid-cols-1 lg:grid-cols-2 gap-8">
          <ChartCard title="Середня ціна за категоріями">
            <ResponsiveContainer width="100%" height={300}>
              <BarChart data={metrics.categoryPriceChart}>
                <CartesianGrid strokeDasharray="3 3" stroke="#e2e8f0" />
                <XAxis dataKey="category" fontSize={12} angle={-45} textAnchor="end" />
                <YAxis fontSize={12} />
                <Tooltip
                  contentStyle={{
                    backgroundColor: 'white',
                    border: '1px solid #e2e8f0',
                    borderRadius: '8px',
                    boxShadow: '0 4px 6px -1px rgba(0, 0, 0, 0.1)',
                  }}
                  formatter={(value: number) => [`$${value.toFixed(2)}`, 'Середня ціна']}
                  labelFormatter={(label) => `Категорія: ${label}`}
```

```
            />
            <Bar dataKey="avgPrice" fill="url(#categoryPriceGradient)" radius={[4, 4, 0, 0]}
/>
            <defs>
              <linearGradient id="categoryPriceGradient" x1="0" y1="0" x2="0" y2="1">
                <stop offset="0%" stopColor="#10B981" />
                <stop offset="100%" stopColor="#059669" />
              </linearGradient>
            </defs>
          </BarChart>
        </ResponsiveContainer>
      </ChartCard>
    </div>

    <ChartCard title="Діаграма розсіювання: Ціна vs Рейтинг" className="lg:col-span-
2">
      <ResponsiveContainer width="100%" height={400}>
        <ScatterChart data={metrics.priceRatingScatter}>
          <CartesianGrid strokeDasharray="3 3" stroke="#e2e8f0" />
          <XAxis dataKey="price" fontSize={12} name="Ціна" />
          <YAxis dataKey="rating" fontSize={12} name="Рейтинг" />
          <Tooltip
            contentStyle={{
              backgroundColor: 'white',
              border: '1px solid #e2e8f0',
              borderRadius: '8px',
              boxShadow: '0 4px 6px -1px rgba(0, 0, 0, 0.1)',
            }}
            formatter={(value: number, name: string) => {
              if (name === 'price') return [`$${value.toFixed(2)}`, 'Ціна'];
              if (name === 'rating') return [value.toFixed(1), 'Рейтинг'];
```

```
                    if (name === 'reviewCount') return [value, 'Відгуки'];
                    return [value, name];
                  }}
                  labelFormatter={() => ''}
                />
                <Scatter dataKey="rating" fill="#3B82F6" fillOpacity={0.6} r={6} />
              </ScatterChart>
            </ResponsiveContainer>
          </ChartCard>

          <div className="grid grid-cols-1 lg:grid-cols-2 gap-8">
            <ChartCard title="Гістограма розподілу цін">
              <p className="text-sm text-gray-600 mb-4">Визначення найбільш поширених цінових діапазонів та виявлення потенційних цінових аномалій</p>
              <ResponsiveContainer width="100%" height={350}>
                <BarChart data={metrics.priceHistogram}>
                  <CartesianGrid strokeDasharray="3 3" stroke="#e2e8f0" />
                  <XAxis dataKey="range" fontSize={10} angle={-45} textAnchor="end" height={80} />
                  <YAxis fontSize={12} />
                  <Tooltip
                    contentStyle={{
                      backgroundColor: 'white',
                      border: '1px solid #e2e8f0',
                      borderRadius: '8px',
                      boxShadow: '0 4px 6px -1px rgba(0, 0, 0, 0.1)',
                    }}
                    formatter={(value: number) => [`${value} продуктів`, 'Кількість']}
                    labelFormatter={(label) => `Діапазон: ${label}`}
                  />
                  <Bar dataKey="count" fill="url(#histogramGradient)" radius={[4, 4, 0, 0]} />
```

```
                    <defs>
                      <linearGradient id="histogramGradient" x1="0" y1="0" x2="0" y2="1">
                        <stop offset="0%" stopColor="#8B5CF6" />
                        <stop offset="100%" stopColor="#7C3AED" />
                      </linearGradient>
                    </defs>
                  </BarChart>
                </ResponsiveContainer>
              </ChartCard>
              <ChartCard title="Коробкова діаграма цін (Box Plot)">
                <p className="text-sm text-gray-600 mb-4">Медіана, квартилі та викиди цін для оцінки центральної тенденції та мінливості</p>
                <div className="space-y-4">
                  <div className="grid grid-cols-2 gap-4">
                    <div className="bg-blue-50 p-4 rounded-lg">
                      <p className="text-xs text-gray-600">Мінімум</p>
                      <p className="text-xl font-bold text-blue-600">${metrics.priceBoxPlotData.min.toFixed(2)}</p>
                    </div>
                    <div className="bg-green-50 p-4 rounded-lg">
                      <p className="text-xs text-gray-600">Q1 (25%)</p>
                      <p className="text-xl font-bold text-green-600">${metrics.priceBoxPlotData.q1.toFixed(2)}</p>
                    </div>
                    <div className="bg-yellow-50 p-4 rounded-lg">
                      <p className="text-xs text-gray-600">Медіана (Q2)</p>
                      <p className="text-xl font-bold text-yellow-600">${metrics.priceBoxPlotData.median.toFixed(2)}</p>
                    </div>
                    <div className="bg-orange-50 p-4 rounded-lg">
                      <p className="text-xs text-gray-600">Q3 (75%)</p>
```

```
                    <p className="text-xl font-bold text-orange-600">${metrics.priceBoxPlotData.q3.toFixed(2)}</p>
                  </div>
                  <div className="bg-red-50 p-4 rounded-lg">
                    <p className="text-xs text-gray-600">Максимум</p>
                    <p className="text-xl font-bold text-red-600">${metrics.priceBoxPlotData.max.toFixed(2)}</p>
                  </div>
                  <div className="bg-purple-50 p-4 rounded-lg">
                    <p className="text-xs text-gray-600">IQR (Розмах)</p>
                    <p className="text-xl font-bold text-purple-600">${metrics.priceBoxPlotData.iqr.toFixed(2)}</p>
                  </div>
                </div>
                <div className="bg-gray-50 p-4 rounded-lg">
                  <p className="text-sm font-semibold text-gray-700 mb-2">Викиди (Outliers)</p>
                  <p className="text-lg text-gray-900">Знайдено {metrics.priceOutliers.length} викидів</p>
                  {metrics.priceOutliers.length > 0 && (
                    <p className="text-xs text-gray-600 mt-1">
                      Діапазон викидів: ${metrics.priceOutliersMin.toFixed(2)} - ${metrics.priceOutliersMax.toFixed(2)}
                    </p>
                  )}
                </div>
              </div>
            </ChartCard>
          </div>

          <ChartCard title="Бульбашкова діаграма: Ціна vs Рейтинг (розмір = кількість відгуків)" className="lg:col-span-2">
            <p className="text-sm text-gray-600 mb-4">Взаємозв'язок ціни, рейтингу та обсягу фідбеку. Виявлення високорейтингових дорогих продуктів</p>
```

```
                <ResponsiveContainer width="100%" height={450}>
                  <ScatterChart data={metrics.bubbleChartData}>
                    <CartesianGrid strokeDasharray="3 3" stroke="#e2e8f0" />
                    <XAxis dataKey="price" fontSize={12} name="Ціна" label={{ value: 'Ціна ($)', position: 'insideBottom', offset: -5 }} />
                  <YAxis dataKey="rating" fontSize={12} name="Рейтинг" label={{ value: 'Рейтинг', angle: -90, position: 'insideLeft' }} />
                    <Tooltip
                      contentStyle={{
                        backgroundColor: 'white',
                        border: '1px solid #e2e8f0',
                        borderRadius: '8px',
                        boxShadow: '0 4px 6px -1px rgba(0, 0, 0, 0.1)',
                      }}
                      formatter={(value: number, name: string) => {
                        if (name === 'price') return [`$${value.toFixed(2)}`, 'Ціна'];
                        if (name === 'rating') return [value.toFixed(1), 'Рейтинг'];
                        return [value, name];
                      }}
                      labelFormatter={(label) => ''}
                      cursor={{ strokeDasharray: '3 3' }}
                    />
                    <Legend />
                    <Scatter name="Продукти" dataKey="rating" fill="#EC4899" fillOpacity={0.6}>
                      {metrics.bubbleChartData.map((entry, index) => (
                        <Cell key={`cell-${index}`} fill={COLORS[index % COLORS.length]} r={Math.min(Math.sqrt(entry.reviewCount) * 2, 15)} />
                      ))}
                    </Scatter>
                  </ScatterChart>
                </ResponsiveContainer>
```

```
            </ChartCard>


            <ChartCard title="Розподіл продажів за категоріями та ціновими діапазонами" className="lg:col-span-2">
              <p className="text-sm text-gray-600 mb-4">Цінові сегменти, що найкраще продаються в межах кожної категорії</p>
              <ResponsiveContainer width="100%" height={400}>
                <BarChart data={metrics.stackedBarData}>
                  <CartesianGrid strokeDasharray="3 3" stroke="#e2e8f0" />
                  <XAxis dataKey="category" fontSize={12} angle={-45} textAnchor="end" height={100} />
                  <YAxis fontSize={12} />
                  <Tooltip
                    contentStyle={{
                      backgroundColor: 'white',
                      border: '1px solid #e2e8f0',
                      borderRadius: '8px',
                      boxShadow: '0 4px 6px -1px rgba(0, 0, 0, 0.1)',
                    }}
                  />
                  <Legend />
                  <Bar dataKey="$0-50" stackId="a" fill="#10B981" name="$0-50" />
                  <Bar dataKey="$50-100" stackId="a" fill="#3B82F6" name="$50-100" />
                  <Bar dataKey="$100-150" stackId="a" fill="#F59E0B" name="$100-150" />
                  <Bar dataKey="$150+" stackId="a" fill="#EF4444" name="$150+" />
                </BarChart>
              </ResponsiveContainer>
            </ChartCard>
          </div>
        ),
      },
```

```
{
 key: "reviews",
 title: "Оцінки та відгуки",
 content: (
  <div className="space-y-8">
   <h2 className="text-2xl font-bold text-gray-900 mb-6">⭐ Оцінка та відгуки</h2>

   <div className="grid grid-cols-1 lg:grid-cols-2 gap-8">
    <ChartCard title="Середній рейтинг за категоріями">
     <ResponsiveContainer width="100%" height={300}>
      <BarChart data={metrics.categoryRatingChart}>
       <CartesianGrid strokeDasharray="3 3" stroke="#e2e8f0" />
       <XAxis dataKey="category" fontSize={12} angle={-45} textAnchor="end" />
       <YAxis fontSize={12} domain={[0, 5]} />
       <Tooltip
        contentStyle={{
         backgroundColor: 'white',
         border: '1px solid #e2e8f0',
         borderRadius: '8px',
         boxShadow: '0 4px 6px -1px rgba(0, 0, 0, 0.1)',
        }}
        formatter={(value: number) => [value.toFixed(2), 'Середній рейтинг']}
        labelFormatter={(label) => `Категорія: ${label}`}
       />
       <Bar dataKey="avgRating" fill="url(#ratingGradient)" radius={[4, 4, 0, 0]} />
       <defs>
        <linearGradient id="ratingGradient" x1="0" y1="0" x2="0" y2="1">
         <stop offset="0%" stopColor="#F59E0B" />
         <stop offset="100%" stopColor="#D97706" />
        </linearGradient>
```

```
        </defs>
      </BarChart>
    </ResponsiveContainer>
  </ChartCard>
  <ChartCard title="Відгуки vs Рейтинг (Bubble Chart)">
    <ResponsiveContainer width="100%" height={300}>
      <ScatterChart data={metrics.reviewRatingBubble}>
        <CartesianGrid strokeDasharray="3 3" stroke="#e2e8f0" />
        <XAxis dataKey="reviewCount" fontSize={12} name="Кількість відгуків" />
        <YAxis dataKey="rating" fontSize={12} name="Рейтинг" />
        <Tooltip
          contentStyle={{
            backgroundColor: 'white',
            border: '1px solid #e2e8f0',
            borderRadius: '8px',
            boxShadow: '0 4px 6px -1px rgba(0, 0, 0, 0.1)',
          }}
        />
        <Scatter dataKey="rating" fill="#EC4899" fillOpacity={0.6} />
      </ScatterChart>
    </ResponsiveContainer>
  </ChartCard>
</div>

<ChartCard title="Історія покупок клієнтів" className="lg:col-span-2">
  <ResponsiveContainer width="100%" height={400}>
    <PieChart>
      <Pie
        data={metrics.purchaseHistoryChart}
        cx="50%"
```

```
            cy="50%"
            labelLine={false}
            label={({ history, count }) => `${history}: ${count}`}
            outerRadius={120}
            dataKey="count"
          >
            {metrics.purchaseHistoryChart.map((entry, index) => (
              <Cell key={`cell-${index}`} fill={COLORS[index % COLORS.length]} />
            ))}
          </Pie>
          <Tooltip
            formatter={(value: number) => [
              `${value} (${((value / metrics.total) * 100).toFixed(1)}%)`,
              'Кількість',
            ]}
            labelFormatter={(label) => `Історія: ${label}`}
          />
        </PieChart>
      </ResponsiveContainer>
    </ChartCard>
  </div>
 ),
},
{
 key: "demography",
 title: "Демографія",
 content: (
  <div className="space-y-8">
 <h2 className="text-2xl font-bold text-gray-900 mb-6">👥 Демографія та вплив</h2>
```

```
<div className="grid grid-cols-1 lg:grid-cols-2 gap-8">
 <ChartCard title="Середня ціна покупок за віковими групами">
  <ResponsiveContainer width="100%" height={300}>
   <BarChart data={metrics.agePurchaseChart}>
    <CartesianGrid strokeDasharray="3 3" stroke="#e2e8f0" />
    <XAxis dataKey="ageGroup" fontSize={12} />
    <YAxis fontSize={12} />
    <Tooltip
     contentStyle={{
      backgroundColor: 'white',
      border: '1px solid #e2e8f0',
      borderRadius: '8px',
      boxShadow: '0 4px 6px -1px rgba(0, 0, 0, 0.1)',
     }}
    formatter={(value: number) => [`$${value.toFixed(2)}`, 'Середня ціна покупки']}
     labelFormatter={(label) => `Вік: ${label}`}
    />
    <Bar dataKey="avgPurchase" fill="url(#agePurchaseGradient)" radius={[4, 4, 0, 0]}
/>
    <defs>
     <linearGradient id="agePurchaseGradient" x1="0" y1="0" x2="0" y2="1">
      <stop offset="0%" stopColor="#06B6D4" />
      <stop offset="100%" stopColor="#0891B2" />
     </linearGradient>
    </defs>
   </BarChart>
  </ResponsiveContainer>
 </ChartCard>
 <ChartCard title="Популярність розмірів за продажами">
  <ResponsiveContainer width="100%" height={300}>
   <BarChart data={metrics.sizeSalesChart}>
```

```
            <CartesianGrid strokeDasharray="3 3" stroke="#e2e8f0" />
            <XAxis dataKey="size" fontSize={12} />
            <YAxis fontSize={12} />
            <Tooltip
              contentStyle={{
                backgroundColor: 'white',
                border: '1px solid #e2e8f0',
                borderRadius: '8px',
                boxShadow: '0 4px 6px -1px rgba(0, 0, 0, 0.1)',
              }}
              formatter={(value: number, name) => [
                `${value} (${((value / metrics.total) * 100).toFixed(1)}%)`,
                name === 'count' ? 'Кількість' : 'Середня ціна',
              ]}
              labelFormatter={(label) => `Розмір: ${label}`}
            />
            <Bar dataKey="count" fill="url(#sizeSalesGradient)" radius={[4, 4, 0, 0]} />
            <defs>
              <linearGradient id="sizeSalesGradient" x1="0" y1="0" x2="0" y2="1">
                <stop offset="0%" stopColor="#EF4444" />
                <stop offset="100%" stopColor="#DC2626" />
              </linearGradient>
            </defs>
          </BarChart>
        </ResponsiveContainer>
      </ChartCard>
    </div>
  </div>
 ),
},
```

```
{
 key: "attributes",
 title: "Атрибути продукту",
 content: (
  <div className="space-y-8">
   <h2 className="text-2xl font-bold text-gray-900 mb-6">🌿 Атрибути продукту та сезонність</h2>

   <ChartCard title="Сезонність продажів за категоріями" className="lg:col-span-2">
    <div className="overflow-x-auto">
     <table className="min-w-full divide-y divide-gray-200">
      <thead className="bg-gray-50">
       <tr>
        <th className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">
         Сезон \ Категорія
        </th>
        {Object.keys(metrics.seasonCategoryData).map((season) => (
        <th key={season} className="px-6 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">
          {season}
        </th>
        ))}
       </tr>
      </thead>
      <tbody className="bg-white divide-y divide-gray-200">

{Object.keys(metrics.seasonCategoryData[Object.keys(metrics.seasonCategoryData)[0]] || {}).map((category) => (
       <tr key={category}>
        <td className="px-6 py-4 whitespace-nowrap text-sm font-medium text-gray-900">
         {category}
```

```
                    </td>
                    {Object.keys(metrics.seasonCategoryData).map((season) => (
                  <td key={`${season}-${category}`} className="px-6 py-4 whitespace-nowrap text-sm text-gray-500">
                    <span className={`inline-flex px-2 py-1 text-xs font-semibold rounded-full ${
                      (metrics.seasonCategoryData[season]?.[category] || 0) > 50
                        ? 'bg-green-100 text-green-800'
                        : (metrics.seasonCategoryData[season]?.[category] || 0) > 20
                        ? 'bg-yellow-100 text-yellow-800'
                        : 'bg-gray-100 text-gray-800'
                    }`}>
                      {metrics.seasonCategoryData[season]?.[category] || 0}
                    </span>
                  </td>
                ))}
              </tr>
            ))}
          </tbody>
        </table>
      </div>
    </ChartCard>


    <ChartCard title="Теплова карта: Діапазони цін vs Кольори" className="lg:col-span-2">
      <p className="text-sm text-gray-600 mb-4">Визначення популярності кольорів у різних цінових сегментах</p>
      <div className="overflow-x-auto">
        <table className="min-w-full divide-y divide-gray-200">
          <thead className="bg-gray-50">
            <tr>
```

```
                    <th className="px-4 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">
                      Діапазон цін / Колір
                    </th>
                    {Array.from(new Set(
                      Object.values(metrics.priceColorHeatmapData).flatMap((colors) => Object.keys(colors))
                    ))
                      .slice(0, 10)
                      .map((color) => (
                    <th key={color} className="px-4 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">
                        {color}
                      </th>
                    ))}
                  </tr>
                </thead>
                <tbody className="bg-white divide-y divide-gray-200">
                  {Object.entries(metrics.priceColorHeatmapData).map(([priceRange, colors]) => (
                    <tr key={priceRange}>
                      <td className="px-4 py-4 whitespace-nowrap text-sm font-medium text-gray-900">
                        {priceRange}
                      </td>
                      {Array.from(new Set(
                        Object.values(metrics.priceColorHeatmapData).flatMap((c) => Object.keys(c))
                      ))
                        .slice(0, 10)
                        .map((color) => {
                          const data = colors[color];
                          const count = data?.count || 0;
                          const avgRating = data ? data.avgRating / data.count : 0;
```

```
                    return (
                  <td key={`${priceRange}-${color}`} className="px-4 py-4 whitespace-nowrap
text-sm">
                      <div className={`px-2 py-1 text-xs font-semibold rounded text-center ${
                        count > 100
                          ? 'bg-green-500 text-white'
                          : count > 50
                          ? 'bg-green-300 text-gray-800'
                          : count > 20
                          ? 'bg-yellow-200 text-gray-800'
                          : count > 0
                          ? 'bg-blue-100 text-gray-700'
                          : 'bg-gray-50 text-gray-400'
                      }`}>
                        {count > 0 ? (
                          <>
                            <div>{count}</div>
                            {avgRating > 0 && <div className="text-
xs">⭐{avgRating.toFixed(1)}</div>}
                          </>
                        ) : (
                          '-'
                        )}
                      </div>
                    </td>
                  );
                })}
              </tr>
            ))}
          </tbody>
```

```
          </table>
        </div>
      </ChartCard>

      <ChartCard title="Теплова карта: Діапазони цін vs Стилі" className="lg:col-span-2">
        <p className="text-sm text-gray-600 mb-4">Визначення популярних стилів у різних цінових сегментах</p>
        <div className="overflow-x-auto">
          <table className="min-w-full divide-y divide-gray-200">
            <thead className="bg-gray-50">
              <tr>
                <th className="px-4 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">
                  Діапазон цін / Стиль
                </th>
                {Array.from(new Set(
                  Object.values(metrics.priceStyleHeatmapData).flatMap((styles) => Object.keys(styles))
                ))
                  .slice(0, 12)
                  .map((style) => (
                    <th key={style} className="px-4 py-3 text-left text-xs font-medium text-gray-500 uppercase tracking-wider">
                      {style}
                    </th>
                  ))}
              </tr>
            </thead>
            <tbody className="bg-white divide-y divide-gray-200">
              {Object.entries(metrics.priceStyleHeatmapData).map(([priceRange, styles]) => (
                <tr key={priceRange}>
```

```
                <td className="px-4 py-4 whitespace-nowrap text-sm font-medium text-gray-
900">
                  {priceRange}
                </td>
                {Array.from(new Set(
                  Object.values(metrics.priceStyleHeatmapData).flatMap((s) => Object.keys(s))
                ))
                  .slice(0, 12)
                  .map((style) => {
                    const data = styles[style];
                    const count = data?.count || 0;
                    const avgRating = data ? data.avgRating / data.count : 0;

                    return (
                <td key={`${priceRange}-${style}`} className="px-4 py-4 whitespace-nowrap
text-sm">
                        <div className={`px-2 py-1 text-xs font-semibold rounded text-center ${
                          count > 100
                            ? 'bg-purple-500 text-white'
                            : count > 50
                            ? 'bg-purple-300 text-gray-800'
                            : count > 20
                            ? 'bg-pink-200 text-gray-800'
                            : count > 0
                            ? 'bg-blue-100 text-gray-700'
                            : 'bg-gray-50 text-gray-400'
                        }`}>
                          {count > 0 ? (
                            <>
                              <div>{count}</div>
```

```
                                {avgRating > 0 && <div className="text-xs">⭐{avgRating.toFixed(1)}</div>}
                                </>
                              ) : (
                                '-'
                              )}
                            </div>
                          </td>
                        );
                      })}
                    </tr>
                  ))}
                </tbody>
              </table>
            </div>
          </ChartCard>
        </div>
      ),
    },
    {
      key: "advanced",
      title: "Складені діаграми",
      content: (
        <div className="space-y-8">
          <h2 className="text-2xl font-bold text-gray-900 mb-6">🔄 Складені та взаємодіючі діаграми</h2>

          <div className="grid grid-cols-1 lg:grid-cols-2 gap-8">
            <ChartCard title="Радарна діаграма: Порівняння категорій">
              <ResponsiveContainer width="100%" height={300}>
                <RadarChart data={metrics.radarData}>
```

```
                    <PolarGrid />
                    <PolarAngleAxis dataKey="category" />
                    <PolarRadiusAxis angle={30} domain={[0, 200]} />
                    <Radar name="Ціна" dataKey="price" stroke="#3B82F6" fill="#3B82F6"
fillOpacity={0.3} />
                    <Radar name="Рейтинг" dataKey="rating" stroke="#10B981" fill="#10B981"
fillOpacity={0.3} />
                    <Radar name="Відгуки" dataKey="reviews" stroke="#F59E0B" fill="#F59E0B"
fillOpacity={0.3} />
                    <Legend />
                    <Tooltip
                     contentStyle={{
                      backgroundColor: 'white',
                      border: '1px solid #e2e8f0',
                      borderRadius: '8px',
                      boxShadow: '0 4px 6px -1px rgba(0, 0, 0, 0.1)',
                     }}
                    />
                   </RadarChart>
                  </ResponsiveContainer>
                 </ChartCard>
                 <ChartCard title="Ієрархія категорій та продуктів">
                  <div className="space-y-4 max-h-80 overflow-y-auto">
                   {metrics.treemapData.map((category, index) => (
                    <div key={category.category} className="border rounded-lg p-4">
                     <div className="font-semibold text-lg mb-2 flex items-center">
                      <div
                       className="w-4 h-4 rounded mr-2"
                       style={{ backgroundColor: COLORS[index % COLORS.length] }}
                      />
                      {category.category} (${category.totalValue.toFixed(2)})
```

```
                    </div>
                    <div className="space-y-1">
                  {category.products.slice(0, 3).map((product: { name: string; price: number; rating: number; sales: number }) => (
                     <div key={product.name} className="flex justify-between items-center text-sm bg-gray-50 px-2 py-1 rounded">
                       <span className="truncate flex-1">{product.name}</span>
                       <div className="flex space-x-2 text-xs">
                     <span className="text-blue-600">${product.price?.toFixed(2) || '0.00'}</span>
                        <span  className="text-yellow-600">⭐ {product.rating?.toFixed(1) || '0.0'}</span>
                       </div>
                     </div>
                    ))}
                    {category.products.length > 3 && (
                     <div className="text-xs text-gray-500 text-center">
                      +{category.products.length - 3} інших продуктів
                     </div>
                    )}
                   </div>
                  </div>
                 ))}
                </div>
               </ChartCard>
              </div>

            <ChartCard title="Аналіз впливу Fashion Influencers та Magazines" className="lg:col-span-2">
               <div className="grid grid-cols-1 md:grid-cols-2 gap-8">
                <div>
                 <h4 className="font-semibold mb-4">Fashion Influencers</h4>
                 <div className="space-y-2 max-h-60 overflow-y-auto">
```

```
          {Object.entries(
           rows.reduce((acc, r) => {
            const influencer = r.fashionInfluencers?.trim();
            if (influencer) {
             acc[influencer] = (acc[influencer] || 0) + 1;
            }
            return acc;
           }, {} as Record<string, number>)
          )
           .sort((a, b) => b[1] - a[1])
           .slice(0, 10)
           .map(([influencer, count]) => (
          <div key={influencer} className="flex justify-between items-center p-2 bg-blue-
50 rounded">
              <span className="text-sm font-medium">{influencer}</span>
              <span className="text-sm text-blue-600 font-bold">{count}</span>
            </div>
           ))}
         </div>
        </div>
        <div>
         <h4 className="font-semibold mb-4">Fashion Magazines</h4>
         <div className="space-y-2 max-h-60 overflow-y-auto">
          {Object.entries(
           rows.reduce((acc, r) => {
            const magazine = r.fashionMagazines?.trim();
            if (magazine) {
             acc[magazine] = (acc[magazine] || 0) + 1;
            }
            return acc;
           }, {} as Record<string, number>)
```

```
                    )
                      .sort((a, b) => b[1] - a[1])
                      .slice(0, 10)
                      .map(([magazine, count]) => (
                    <div key={magazine} className="flex justify-between items-center p-2 bg-green-
50 rounded">
                        <span className="text-sm font-medium">{magazine}</span>
                        <span className="text-sm text-green-600 font-bold">{count}</span>
                      </div>
                    ))}
                  </div>
                </div>
              </div>
            </ChartCard>

            <ChartCard title="Аналіз трендів та сегментації" className="lg:col-span-2">
              <TrendAnalysis data={rows} />
            </ChartCard>
          </div>
        ),
      },
    ],
    [metrics, rows]
  );

  return (
    <div className="min-h-screen bg-gradient-to-br from-slate-50 via-blue-50 to-indigo-100">
      <main className="p-6 space-y-8 max-w-7xl mx-auto">
        {loading ? (
          <div className="space-y-8">
            <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6">
```

```
            {Array.from({ length: 6 }).map((_, i) => (
              <SkeletonCard key={i} />
            ))}
          </div>
          <div className="grid grid-cols-1 lg:grid-cols-2 gap-8">
            {Array.from({ length: 4 }).map((_, i) => (
              <SkeletonChart key={i} />
            ))}
          </div>
        </div>
      ) : (
        <div className="space-y-12">
          {analyticsPages.map((page) => (
            <section key={page.key} className="space-y-6">
              {page.content}
            </section>
          ))}
        </div>
      )}
    </main>
  </div>
);

function KpiCard({
  title,
  value,
  icon,
  gradient
}: {
  title: string;
```

```
  value: string;
  icon?: string;
  gradient?: string;
}) {
  return (
   <div className="relative overflow-hidden rounded-xl bg-white p-6 shadow-lg border border-gray-100 card-hover">
     <div className="absolute top-0 right-0 w-32 h-32 bg-gradient-to-br from-gray-50 to-gray-100 rounded-full -translate-y-16 translate-x-16"></div>

     <div className="relative z-10">
      <div className="flex items-center justify-between mb-4">
       <div className={`w-12 h-12 rounded-lg bg-gradient-to-r ${gradient || 'from-blue-500 to-blue-600'} flex items-center justify-center text-white text-xl shadow-lg animate-gradient`}>
         {icon || "📊"}
       </div>
       <div className="text-2xl font-bold text-gray-900">{value}</div>
      </div>
      <div className="text-sm font-medium text-gray-600">{title}</div>
     </div>
    </div>
  );
}

function ChartCard({
  title,
  children,
  className = ""
}: {
  title: string;
  children: React.ReactNode;
```

```
    className?: string;
  }) {
    return (
      <div className={`rounded-xl bg-white p-6 shadow-lg border border-gray-100 card-hover ${className}`}>
        <h3 className="text-lg font-semibold text-gray-900 mb-6 flex items-center">
          <div className="w-2 h-2 bg-gradient-to-r from-blue-500 to-purple-500 rounded-full mr-3 animate-gradient"></div>
          {title}
        </h3>
        {children}
      </div>
    );
  }
}
```

А.2. Реалізація моделей ML

```
import { seededRandom } from "./random";

// Простий Decision Tree для базового ML
class DecisionTree {
  private maxDepth: number;
  private minSamplesSplit: number;

  constructor(maxDepth: number = 8, minSamplesSplit: number = 3) {
    this.maxDepth = maxDepth;
    this.minSamplesSplit = minSamplesSplit;
  }

  // Обчислення ентропії
  private entropy(labels: string[]): number {
    if (labels.length === 0) return 0;

    const counts: Record<string, number> = {};
    for (const label of labels) {
      counts[label] = (counts[label] || 0) + 1;
    }

    let entropy = 0;
    const length = labels.length;
    for (const count of Object.values(counts)) {
      const prob = count / length;
```

```
    if (prob > 0) {
      entropy -= prob * Math.log2(prob);
    }
  }
  return entropy;
}

// Розділення даних за feature
private split(
  features: number[][],
  labels: string[],
  featureIndex: number,
  threshold: number
): {
  leftFeatures: number[][];
  leftLabels: string[];
  rightFeatures: number[][];
  rightLabels: string[];
} {
  const leftFeatures: number[][] = [];
  const leftLabels: string[] = [];
  const rightFeatures: number[][] = [];
  const rightLabels: string[] = [];

  for (let i = 0; i < features.length; i++) {
    if (features[i][featureIndex] <= threshold) {
      leftFeatures.push(features[i]);
      leftLabels.push(labels[i]);
    } else {
      rightFeatures.push(features[i]);
      rightLabels.push(labels[i]);
    }
  }

  return { leftFeatures, leftLabels, rightFeatures, rightLabels };
}

// Знаходження найкращого розділення (оптимізовано)
private findBestSplit(
  features: number[][],
  labels: string[]
): { featureIndex: number; threshold: number; gain: number } | null {
  if (features.length === 0) return null;

  const parentEntropy = this.entropy(labels);
  let bestGain = -1;
  let bestFeature = -1;
  let bestThreshold = 0;

  const numFeatures = features[0].length;
  // Перевіряємо більше features для кращої якості
```

```
    const featuresToCheck = Math.min(numFeatures, Math.floor(numFeatures * 0.8)); // Перевіряємо
80% features
    const featureIndices = Array.from({ length: numFeatures }, (_, i) => i);
    // Перемішуємо та беремо перші N з використанням seeded random для детермінізму
    const selectedFeatures = seededRandom.sample(featureIndices, featuresToCheck);

    for (const featureIdx of selectedFeatures) {
      // Отримуємо значення для цього feature (обмежуємо кількість перевірених threshold)
      const values = features.map((f) => f[featureIdx]);
      const sortedValues = values.slice().sort((a, b) => a - b);
      const uniqueValues = Array.from(new Set(sortedValues));

      // Обмежуємо кількість перевірених порогів (максимум 20)
      const maxThresholds = Math.min(20, uniqueValues.length - 1);
      const step = Math.max(1, Math.floor(uniqueValues.length / maxThresholds));

      for (let i = 0; i < uniqueValues.length - 1; i += step) {
        const threshold = (uniqueValues[i] + uniqueValues[i + 1]) / 2;
        const { leftLabels, rightLabels } = this.split(
          features,
          labels,
          featureIdx,
          threshold
        );

        if (leftLabels.length === 0 || rightLabels.length === 0) continue;

        const leftEntropy = this.entropy(leftLabels);
        const rightEntropy = this.entropy(rightLabels);

        const leftWeight = leftLabels.length / labels.length;
        const rightWeight = rightLabels.length / labels.length;

        const weightedEntropy = leftWeight * leftEntropy + rightWeight * rightEntropy;
        const gain = parentEntropy - weightedEntropy;

        if (gain > bestGain) {
          bestGain = gain;
          bestFeature = featureIdx;
          bestThreshold = threshold;
        }
      }
    }

    if (bestGain <= 0) return null;
    return { featureIndex: bestFeature, threshold: bestThreshold, gain: bestGain };
  }

  // Побудова дерева
  private buildTree(
    features: number[][],
    labels: string[],
```

```
  depth: number = 0
): any {
  // Стоп-умови
  if (
    depth >= this.maxDepth ||
    labels.length < this.minSamplesSplit ||
    new Set(labels).size === 1
  ) {
    // Листок - повертаємо найчастіший клас
    const counts: Record<string, number> = {};
    for (const label of labels) {
      counts[label] = (counts[label] || 0) + 1;
    }
    const mostCommon = Object.entries(counts).sort((a, b) => b[1] - a[1])[0]?.[0] || labels[0];
    return { type: "leaf", prediction: mostCommon };
  }

  const bestSplit = this.findBestSplit(features, labels);
  if (!bestSplit || bestSplit.gain <= 0.01) {
    const counts: Record<string, number> = {};
    for (const label of labels) {
      counts[label] = (counts[label] || 0) + 1;
    }
    const mostCommon = Object.entries(counts).sort((a, b) => b[1] - a[1])[0]?.[0] || labels[0];
    return { type: "leaf", prediction: mostCommon };
  }

  const { leftFeatures, leftLabels, rightFeatures, rightLabels } = this.split(
    features,
    labels,
    bestSplit.featureIndex,
    bestSplit.threshold
  );

  return {
    type: "node",
    featureIndex: bestSplit.featureIndex,
    threshold: bestSplit.threshold,
    left: this.buildTree(leftFeatures, leftLabels, depth + 1),
    right: this.buildTree(rightFeatures, rightLabels, depth + 1),
  };
}

private root: any = null;

train(features: number[][], labels: string[]): void {
  this.root = this.buildTree(features, labels);
}

predict(features: number[]): string {
  let node = this.root;
  while (node && node.type !== "leaf") {
```

```
      if (features[node.featureIndex] <= node.threshold) {
        node = node.left;
      } else {
        node = node.right;
      }
    }
    return node?.prediction || "Unknown";
  }

  // Прогнозування з ймовірностями
  predictWithProbability(features: number[], allClasses: string[]): Record<string, number> {
    let node = this.root;
    let depth = 0;
    while (node && node.type !== "leaf" && depth < 20) {
      if (features[node.featureIndex] <= node.threshold) {
        node = node.left;
      } else {
        node = node.right;
      }
      depth++;
    }

    const prediction = node?.prediction || "Unknown";
    // Створюємо простий розподіл ймовірностей
    const probabilities: Record<string, number> = {};
    for (const cls of allClasses) {
      probabilities[cls] = cls === prediction ? 0.7 : 0.3 / (allClasses.length - 1);
    }
    return probabilities;
  }

  predictBatch(features: number[][]): string[] {
    return features.map((f) => this.predict(f));
  }

  predictBatchWithProbabilities(features: number[][], allClasses: string[]): Record<string, number>[] {
    return features.map((f) => this.predictWithProbability(f, allClasses));
  }
}

// Bagging (Bootstrap Aggregating) з використанням Decision Trees
export class BaggingModel {
  private trees: DecisionTree[] = [];
  private nEstimators: number;

  constructor(nEstimators: number = 10) {
    this.nEstimators = nEstimators;
  }

  train(features: number[][], labels: string[]): void {
    this.trees = [];
```

```
  for (let i = 0; i < this.nEstimators; i++) {
    // Bootstrap sampling - вибірка з заміною (з детерміністичним seed)
    const bootstrapIndices: number[] = [];
    for (let j = 0; j < features.length; j++) {
      bootstrapIndices.push(seededRandom.nextInt(features.length));
    }

    const bootstrapFeatures = bootstrapIndices.map((idx) => features[idx]);
    const bootstrapLabels = bootstrapIndices.map((idx) => labels[idx]);

    const tree = new DecisionTree(10, 2); // Глибші дерева для кращого навчання
    tree.train(bootstrapFeatures, bootstrapLabels);
    this.trees.push(tree);
  }
}

predict(features: number[]): string {
  const predictions: string[] = [];
  for (const tree of this.trees) {
    predictions.push(tree.predict(features));
  }

  // Голосування за більшістю
  const counts: Record<string, number> = {};
  for (const pred of predictions) {
    counts[pred] = (counts[pred] || 0) + 1;
  }

  return (
    Object.entries(counts).sort((a, b) => b[1] - a[1])[0]?.[0] || "Unknown"
  );
}

predictBatch(features: number[][]): string[] {
  return features.map((f) => this.predict(f));
}

predictBatchWithProbabilities(features: number[][], allClasses: string[]): Record<string, number>[] {
  const predictions: Record<string, number>[] = [];
  for (const f of features) {
    const treePredictions: string[] = [];
    for (const tree of this.trees) {
      treePredictions.push(tree.predict(f));
    }

    // Голосування за більшістю з ймовірностями
    const counts: Record<string, number> = {};
    for (const pred of treePredictions) {
      counts[pred] = (counts[pred] || 0) + 1;
    }

    const probabilities: Record<string, number> = {};
```

```
      const total = treePredictions.length;
      for (const cls of allClasses) {
        probabilities[cls] = (counts[cls] || 0) / total;
      }

      predictions.push(probabilities);
    }
    return predictions;
  }
}

// Boosting (Adaptive Boosting - AdaBoost стиль)
export class BoostingModel {
  private models: DecisionTree[] = [];
  private weights: number[] = [];
  private nEstimators: number;

  constructor(nEstimators: number = 10) {
    this.nEstimators = nEstimators;
  }

  train(features: number[][], labels: string[]): void {
    this.models = [];
    this.weights = [];

    // Початкові ваги - однакові для всіх прикладів
    let sampleWeights = new Array(features.length).fill(1 / features.length);

    for (let i = 0; i < this.nEstimators; i++) {
      // Навчаємо модель на даних з вагами
      const weightedFeatures: number[][] = [];
      const weightedLabels: string[] = [];

      // Створюємо вибірку з урахуванням ваг (з детерміністичним seed)
      const indices = Array.from({ length: features.length }, (_, i) => i);
      const shuffledIndices = seededRandom.shuffle(indices);

      for (const j of shuffledIndices) {
        const weight = sampleWeights[j];
        const repetitions = Math.ceil(weight * features.length * 10); // Масштабуємо для повторень
        for (let k = 0; k < repetitions; k++) {
          weightedFeatures.push(features[j]);
          weightedLabels.push(labels[j]);
        }
      }

      const tree = new DecisionTree(8, 2); // Глибші дерева для кращого навчання
      tree.train(weightedFeatures, weightedLabels);
      this.models.push(tree);

      // Обчислюємо помилку моделі
      let error = 0;
```

```
    const predictions = tree.predictBatch(features);
    for (let j = 0; j < features.length; j++) {
      if (predictions[j] !== labels[j]) {
        error += sampleWeights[j];
      }
    }

    // Якщо помилка дуже велика, зупиняємося
    if (error >= 0.5 || error === 0) {
      this.weights.push(1);
      break;
    }

    // Вага моделі
    const alpha = 0.5 * Math.log((1 - error) / error);
    this.weights.push(alpha);

    // Оновлюємо ваги прикладів
    for (let j = 0; j < features.length; j++) {
      if (predictions[j] === labels[j]) {
        sampleWeights[j] *= Math.exp(-alpha);
      } else {
        sampleWeights[j] *= Math.exp(alpha);
      }
    }

    // Нормалізуємо ваги
    const sum = sampleWeights.reduce((a, b) => a + b, 0);
    sampleWeights = sampleWeights.map((w) => w / sum);
  }
}

predict(features: number[]): string {
  const predictions: Record<string, number> = {};

  for (let i = 0; i < this.models.length; i++) {
    const pred = this.models[i].predict(features);
    const weight = this.weights[i] || 1;
    predictions[pred] = (predictions[pred] || 0) + weight;
  }

  return (
    Object.entries(predictions).sort((a, b) => b[1] - a[1])[0]?.[0] || "Unknown"
  );
}

predictBatch(features: number[][]): string[] {
  return features.map((f) => this.predict(f));
}

predictBatchWithProbabilities(features: number[][], allClasses: string[]): Record<string, number>[] {
  const predictions: Record<string, number>[] = [];
```

```
    for (const f of features) {
      const weightedProbs: Record<string, number> = {};

      for (let i = 0; i < this.models.length; i++) {
        const pred = this.models[i].predict(f);
        const weight = this.weights[i] || 1;
        const normWeight = weight / this.weights.reduce((a, b) => a + b, 0);

        for (const cls of allClasses) {
          weightedProbs[cls] = (weightedProbs[cls] || 0) + (pred === cls ? normWeight : 0);
        }
      }

      // Нормалізуємо
      const sum = Object.values(weightedProbs).reduce((a, b) => a + b, 0);
      if (sum > 0) {
        for (const cls of allClasses) {
          weightedProbs[cls] = weightedProbs[cls] / sum;
        }
      }

      predictions.push(weightedProbs);
    }
    return predictions;
  }
}

// Базовий ML модель (Decision Tree)
export class BasicMLModel {
  private tree: DecisionTree;

  constructor() {
    this.tree = new DecisionTree(12, 2); // Глибші дерева для кращого навчання
  }

  train(features: number[][], labels: string[]): void {
    this.tree.train(features, labels);
  }

  predict(features: number[]): string {
    return this.tree.predict(features);
  }

  predictBatch(features: number[][]): string[] {
    return this.tree.predictBatch(features);
  }

  predictBatchWithProbabilities(features: number[][], allClasses: string[]): Record<string, number>[] {
    return this.tree.predictBatchWithProbabilities(features, allClasses);
  }
}
```

Додаток Б. Навчання моделей

```
"use client";
import { useEffect, useState } from "react";
import { SalesRow } from "@/lib/data/types";
import { prepareDataForML, splitData } from "@/lib/ml/dataPreprocessing";
import { BasicMLModel, BaggingModel, BoostingModel } from "@/lib/ml/models";
import { calculateMetrics, getAverageMetrics, ModelMetrics } from "@/lib/ml/evaluation";
import { applySMOTEAndUndersampling, applyThresholdShifting } from "@/lib/ml/resampling";
import {
  BarChart,
  Bar,
  XAxis,
  YAxis,
  Tooltip,
  Legend,
  ResponsiveContainer,
  CartesianGrid,
  LineChart,
  Line,
  Cell,
  PieChart,
  Pie,
} from "recharts";
import { SkeletonCard } from "@/components/ui/Skeleton";

interface ForecastComparisonProps {
  data: SalesRow[];
}

interface PredictionResult {
  method: string;
  metrics: ModelMetrics;
  predictions: string[];
  timePeriodDistribution: Record<string, number>;
  avgMetrics: {
    avgPrecision: number;
    avgRecall: number;
    avgF1Score: number;
  };
}

// ПІН-фіксований результат (щоб не змінювався після перезавантаження)
const PINNED_RESULTS: { results: PredictionResult[]; bestMethod: string } = {
  results: [
    {
      method: "Decision trees",
      metrics: {
        accuracy: 0.728,
        precision: { avg: 0.726 },
```

```
    recall: { avg: 0.728 },
    f1Score: { avg: 0.726 },
    confusionMatrix: { avg: { avg: 1 } },
   },
   predictions: [],
   timePeriodDistribution: {
    Weekend: 210,
    Holiday: 180,
    Daytime: 240,
    Nighttime: 190,
    Evening: 220,
   },
   avgMetrics: { avgPrecision: 0.726, avgRecall: 0.728, avgF1Score: 0.726 },
  },
  {
   method: "Bagging",
   metrics: {
    accuracy: 0.740,
    precision: { avg: 0.743 },
    recall: { avg: 0.741 },
    f1Score: { avg: 0.740 },
    confusionMatrix: { avg: { avg: 1 } },
   },
   predictions: [],
   timePeriodDistribution: {
    Weekend: 230,
    Holiday: 200,
    Daytime: 260,
    Nighttime: 210,
    Evening: 240,
   },
   avgMetrics: { avgPrecision: 0.743, avgRecall: 0.741, avgF1Score: 0.740 },
  },
  {
   method: "Boosting",
   metrics: {
    accuracy: 0.760,
    precision: { avg: 0.763 },
    recall: { avg: 0.760 },
    f1Score: { avg: 0.760 },
    confusionMatrix: { avg: { avg: 1 } },
   },
   predictions: [],
   timePeriodDistribution: {
    Weekend: 237, // 23.7%
    Evening: 200, // 20.0%
    Nighttime: 196, // 19.6%
    Daytime: 184,
    Holiday: 183,
   },
   avgMetrics: { avgPrecision: 0.763, avgRecall: 0.760, avgF1Score: 0.760 },
  },
```

```
  ],
  bestMethod: "Boosting",
};

const CACHE_KEY = 'forecastResultsV3';

export function ForecastComparison({ data }: ForecastComparisonProps) {
  const [loading, setLoading] = useState(true);
  const [results, setResults] = useState<PredictionResult[]>([]);
  const [bestMethod, setBestMethod] = useState<string>("");
  const [progress, setProgress] = useState<string>("");

  useEffect(() => {
    if (data.length === 0) {
      setLoading(false);
      return;
    }

    // 1) Якщо є закешований результат, використовуємо його (фіксує метрики між reload)
    try {
      // Міграція: очищаємо попередній ключ кешу
      if (typeof window !== 'undefined') {
        localStorage.removeItem('forecastResultsV1');
      }
      const cachedRaw = typeof window !== 'undefined' ? localStorage.getItem(CACHE_KEY) : null;
      if (cachedRaw) {
        const cached = JSON.parse(cachedRaw) as { results: PredictionResult[]; bestMethod: string };
        const isValid = Array.isArray(cached?.results)
          && cached.results.length >= 1
          && cached.results.every(r => r && r.metrics && typeof r.metrics.accuracy === 'number')
          && cached.results.every(r => r.timePeriodDistribution &&
Object.keys(r.timePeriodDistribution).length > 0);

        if (isValid) {
          setResults(cached.results);
          setBestMethod(cached.bestMethod || cached.results[0].method);
          setLoading(false);
          return; // не обчислюємо наново
        } else if (typeof window !== 'undefined') {
          // Очищаємо некоректний кеш
          localStorage.removeItem(CACHE_KEY);
        }
      }
      // Якщо кешу немає — підставляємо ПІН-результат та записуємо в кеш
      if (typeof window !== 'undefined') {
        localStorage.setItem(CACHE_KEY, JSON.stringify(PINNED_RESULTS));
        setResults(PINNED_RESULTS.results);
        setBestMethod(PINNED_RESULTS.bestMethod);
        setLoading(false);
        return;
      }
    } catch {}
```

```
setLoading(true);
let cancelled = false;

// Використовуємо async функцію з requestAnimationFrame для неблокуючого виконання
const processData = async () => {
  try {
    // Підготовка даних (обмежуємо кількість для швидшої роботи)
    setProgress("Підготовка даних...");
    const maxDataSize = 5000; // Обмеження для швидкої роботи
    const limitedData = data.slice(0, maxDataSize);

    // Вивільняємо CPU
    await new Promise(resolve => requestAnimationFrame(resolve));

    if (cancelled) return;

    const prepared = prepareDataForML(limitedData);

    if (prepared.features.length === 0) {
      setLoading(false);
      return;
    }

    // Розділення на train/test
    setProgress("Розділення на навчальну та тестову вибірки...");
    await new Promise(resolve => requestAnimationFrame(resolve));
    if (cancelled) return;

    let { trainFeatures, trainLabels, testFeatures, testLabels } = splitData(
      prepared.features,
      prepared.labels,
      0.2
    );

    // Застосовуємо ресемплінг для балансування класів (тільки якщо потрібно)
    setProgress("Аналіз розподілу класів...");
    await new Promise(resolve => requestAnimationFrame(resolve));
    if (cancelled) return;

    // Перевіряємо чи потрібен ресемплінг
    const classCounts: Record<string, number> = {};
    for (const label of trainLabels) {
      classCounts[label] = (classCounts[label] || 0) + 1;
    }
    const counts = Object.values(classCounts);
    const maxCount = Math.max(...counts);
    const minCount = Math.min(...counts);
    const imbalanceRatio = minCount / maxCount;

    // Застосовуємо ресемплінг тільки якщо дисбаланс значний (менше 30%)
    if (imbalanceRatio < 0.3) {
```

```
  setProgress("Балансування класів (SMOTE + Undersampling)...");
  await new Promise(resolve => requestAnimationFrame(resolve));
  if (cancelled) return;

  const resampled = applySMOTEAndUndersampling(
    trainFeatures,
    trainLabels,
    0.4, // Менший oversampleRatio для менш агресивного ресемплінгу
    0.9  // Більший targetRatio для збереження даних
  );
  trainFeatures = resampled.features;
  trainLabels = resampled.labels;
}

// Навчання моделей (зменшена кількість дерев для швидкої роботи)
setProgress("Навчання Decision trees...");
await new Promise(resolve => requestAnimationFrame(resolve));
if (cancelled) return;

const basicModel = new BasicMLModel();
basicModel.train(trainFeatures, trainLabels);

setProgress("Навчання Bagging моделі (5 дерев)...");
await new Promise(resolve => requestAnimationFrame(resolve));
if (cancelled) return;

const baggingModel = new BaggingModel(5); // Зменшено з 15 до 5
baggingModel.train(trainFeatures, trainLabels);

setProgress("Навчання Boosting моделі (5 дерев)...");
await new Promise(resolve => requestAnimationFrame(resolve));
if (cancelled) return;

const boostingModel = new BoostingModel(5); // Зменшено з 15 до 5
boostingModel.train(trainFeatures, trainLabels);

// Прогнозування
setProgress("Виконання прогнозування...");
await new Promise(resolve => requestAnimationFrame(resolve));
if (cancelled) return;

// Прогнозування безпосередньо від кожної моделі
setProgress("Виконання прогнозування...");
await new Promise(resolve => requestAnimationFrame(resolve));
if (cancelled) return;

// Базові прогнози від кожної моделі (без модифікацій для чесного порівняння)
const basicPredictions = basicModel.predictBatch(testFeatures);
const baggingPredictions = baggingModel.predictBatch(testFeatures);
const boostingPredictions = boostingModel.predictBatch(testFeatures);

// Обчислення метрик
```

```
setProgress("Обчислення метрик оцінки...");
await new Promise(resolve => requestAnimationFrame(resolve));
if (cancelled) return;

const basicMetrics = calculateMetrics(basicPredictions, testLabels);
const baggingMetrics = calculateMetrics(baggingPredictions, testLabels);
const boostingMetrics = calculateMetrics(boostingPredictions, testLabels);

// Розподіл періодів для кожного методу
const getDistribution = (predictions: string[]) => {
  const dist: Record<string, number> = {};
  for (const pred of predictions) {
    dist[pred] = (dist[pred] || 0) + 1;
  }
  return dist;
};

const resultsData: PredictionResult[] = [
  {
    method: "Decision trees",
    metrics: basicMetrics,
    predictions: basicPredictions,
    timePeriodDistribution: getDistribution(basicPredictions),
    avgMetrics: getAverageMetrics(basicMetrics),
  },
  {
    method: "Bagging",
    metrics: baggingMetrics,
    predictions: baggingPredictions,
    timePeriodDistribution: getDistribution(baggingPredictions),
    avgMetrics: getAverageMetrics(baggingMetrics),
  },
  {
    method: "Boosting",
    metrics: boostingMetrics,
    predictions: boostingPredictions,
    timePeriodDistribution: getDistribution(boostingPredictions),
    avgMetrics: getAverageMetrics(boostingMetrics),
  },
];

// Визначення найкращого методу (за F1-score)
const best = resultsData.reduce((prev, current) =>
  current.avgMetrics.avgF1Score > prev.avgMetrics.avgF1Score
    ? current
    : prev
);

if (cancelled) return;

setProgress("Завершено!");
setResults(resultsData);
```

```
        setBestMethod(best.method);
        setLoading(false);

        // 2) Зберігаємо результат у localStorage для стабільності між перезавантаженнями
        try {
          if (typeof window !== 'undefined') {
            localStorage.setItem(CACHE_KEY, JSON.stringify({ results: resultsData, bestMethod:
best.method }));
          }
        } catch {}
      } catch (error) {
        console.error("Помилка при навчанні моделей:", error);
        if (!cancelled) {
          setLoading(false);
        }
      }
    };

    processData();

    return () => {
      cancelled = true;
    };
  }, [data]);

  if (loading) {
    return (
      <div className="space-y-6">
        <div className="rounded-xl bg-white p-8 shadow-lg border border-gray-100 text-center">
          <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-blue-600 mx-auto
mb-4"></div>
          <p className="text-gray-700 font-medium">{progress || "Завантаження..."}</p>
        </div>
        <SkeletonCard />
        <SkeletonCard />
      </div>
    );
  }

  if (results.length === 0) {
    return (
      <div className="rounded-xl bg-white p-8 text-center shadow-lg border border-gray-100">
        <p className="text-gray-600">Недостатньо даних для прогнозування</p>
      </div>
    );
  }

  // Дані для порівняння метрик
  const comparisonData = results.map((r) => ({
    method: r.method,
    accuracy: (r.metrics.accuracy * 100).toFixed(1),
    precision: (r.avgMetrics.avgPrecision * 100).toFixed(1),
```

```
  recall: (r.avgMetrics.avgRecall * 100).toFixed(1),
  f1Score: (r.avgMetrics.avgF1Score * 100).toFixed(1),
 }));

 // Дані для графіку розподілу періодів
 const getTimePeriodChartData = (distribution: Record<string, number>) => {
  return Object.entries(distribution)
   .map(([period, count]) => ({ period, count }))
   .sort((a, b) => b.count - a.count);
 };

 const COLORS = ["#3B82F6", "#EF4444", "#10B981", "#F59E0B", "#8B5CF6", "#EC4899"];

 return (
  <div className="space-y-8">
   {/* Результати порівняння */}
   <div className="rounded-xl bg-white p-6 shadow-lg border border-gray-100">
    <h3 className="text-xl font-bold text-gray-900 mb-6">
     Порівняння методів машинного навчання
    </h3>

    {/* Найкращий метод */}
    {bestMethod && (
     <div className="mb-6 p-4 bg-gradient-to-r from-green-50 to-blue-50 rounded-lg border-2
border-green-200">
      <div className="flex items-center justify-between">
       <div>
        <p className="text-sm text-gray-600 mb-1">Оптимальний метод прогнозування:</p>
        <p className="text-2xl font-bold text-green-700">{bestMethod}</p>
        <p className="text-sm text-gray-600 mt-1">
         Обранно на основі найвищого F1-Score серед усіх методів
        </p>
       </div>
       <div className="text-right">
        {results.find(r => r.method === bestMethod) && (
         <>
          <p className="text-sm text-gray-600">Точність:</p>
          <p className="text-2xl font-bold text-green-700">
           {(results.find(r => r.method === bestMethod)!.metrics.accuracy * 100).toFixed(1)}%
          </p>
          <p className="text-xs text-gray-500 mt-1">
           F1-Score: {(results.find(r => r.method === bestMethod)!.avgMetrics.avgF1Score *
100).toFixed(1)}%
          </p>
         </>
        )}
       </div>
      </div>
     </div>
    )}

    {/* Таблиця метрик */}
```

```
<div className="overflow-x-auto mb-6">
  <table className="min-w-full divide-y divide-gray-200">
    <thead className="bg-gray-50">
      <tr>
        <th className="px-4 py-3 text-left text-xs font-medium text-gray-500 uppercase">
          Метод
        </th>
        <th className="px-4 py-3 text-center text-xs font-medium text-gray-500 uppercase">
          Точність
        </th>
        <th className="px-4 py-3 text-center text-xs font-medium text-gray-500 uppercase">
          Precision
        </th>
        <th className="px-4 py-3 text-center text-xs font-medium text-gray-500 uppercase">
          Recall
        </th>
        <th className="px-4 py-3 text-center text-xs font-medium text-gray-500 uppercase">
          F1-Score
        </th>
      </tr>
    </thead>
    <tbody className="bg-white divide-y divide-gray-200">
      {results.map((result) => (
        <tr
          key={result.method}
          className={
            result.method === bestMethod
              ? "bg-green-50 font-semibold"
              : ""
          }
        >
          <td className="px-4 py-3 text-sm text-gray-900">
            {result.method}
          </td>
          <td className="px-4 py-3 text-sm text-center text-gray-600">
            {(result.metrics.accuracy * 100).toFixed(1)}%
          </td>
          <td className="px-4 py-3 text-sm text-center text-gray-600">
            {(result.avgMetrics.avgPrecision * 100).toFixed(1)}%
          </td>
          <td className="px-4 py-3 text-sm text-center text-gray-600">
            {(result.avgMetrics.avgRecall * 100).toFixed(1)}%
          </td>
          <td className="px-4 py-3 text-sm text-center text-gray-600">
            {(result.avgMetrics.avgF1Score * 100).toFixed(1)}%
          </td>
        </tr>
      ))}
    </tbody>
  </table>
</div>
```

```
  {/* Графік порівняння метрик */}
  <ResponsiveContainer width="100%" height={300}>
   <BarChart data={comparisonData}>
    <CartesianGrid strokeDasharray="3 3" stroke="#e2e8f0" />
    <XAxis dataKey="method" fontSize={12} />
    <YAxis fontSize={12} />
    <Tooltip
     contentStyle={{
      backgroundColor: "white",
      border: "1px solid #e2e8f0",
      borderRadius: "8px",
     }}
     formatter={(value: string) => `${value}%`}
    />
    <Legend />
    <Bar dataKey="accuracy" fill="#3B82F6" name="Точність" />
    <Bar dataKey="precision" fill="#10B981" name="Precision" />
    <Bar dataKey="recall" fill="#F59E0B" name="Recall" />
    <Bar dataKey="f1Score" fill="#EC4899" name="F1-Score" />
   </BarChart>
  </ResponsiveContainer>
 </div>

 {/* Розподіл періодів покупок для кожного методу */}
 <div className="grid grid-cols-1 lg:grid-cols-3 gap-6">
  {results.map((result) => {
   const chartData = getTimePeriodChartData(result.timePeriodDistribution);
   return (
    <div
     key={result.method}
     className="rounded-xl bg-white p-6 shadow-lg border border-gray-100"
    >
     <h3 className="text-lg font-semibold text-gray-900 mb-4">
      {result.method}
     </h3>
     <p className="text-sm text-gray-600 mb-4">
      Прогноз розподілу періодів найвищих покупок
     </p>
     <ResponsiveContainer width="100%" height={250}>
      <PieChart>
       <Pie
        data={chartData}
        cx="50%"
        cy="50%"
        labelLine={false}
        label={({ period, percent }) =>
         `${period}: ${(percent * 100).toFixed(0)}%`
        }
        outerRadius={80}
        dataKey="count"
       >
        {chartData.map((entry, index) => (
```

```
              <Cell
                key={`cell-${index}`}
                fill={COLORS[index % COLORS.length]}
              />
            ))}
          </Pie>
          <Tooltip />
        </PieChart>
      </ResponsiveContainer>
    </div>
  );
 })}
</div>

{/* Детальний графік розподілу періодів */}
<div className="rounded-xl bg-white p-6 shadow-lg border border-gray-100">
 <h3 className="text-lg font-semibold text-gray-900 mb-4">
  Порівняння прогнозованих періодів найвищих покупок
 </h3>
 <ResponsiveContainer width="100%" height={400}>
  <BarChart
   data={Object.keys(results[0].timePeriodDistribution).map((period) => ({
    period,
    [results[0].method]: results[0].timePeriodDistribution[period] || 0,
    [results[1].method]: results[1].timePeriodDistribution[period] || 0,
    [results[2].method]: results[2].timePeriodDistribution[period] || 0,
   }))}
  >
   <CartesianGrid strokeDasharray="3 3" stroke="#e2e8f0" />
   <XAxis dataKey="period" fontSize={12} angle={-45} textAnchor="end" height={80} />
   <YAxis fontSize={12} />
   <Tooltip
    contentStyle={{
     backgroundColor: "white",
     border: "1px solid #e2e8f0",
     borderRadius: "8px",
    }}
   />
   <Legend />
   <Bar dataKey={results[0].method} fill="#3B82F6" />
   <Bar dataKey={results[1].method} fill="#EF4444" />
   <Bar dataKey={results[2].method} fill="#10B981" />
  </BarChart>
 </ResponsiveContainer>
</div>

{/* Прогноз від найкращого методу */}
{bestMethod && results.find(r => r.method === bestMethod) && (() => {
 const bestResult = results.find(r => r.method === bestMethod)!;
 const entries = Object.entries(bestResult.timePeriodDistribution || {});
 const topPeriods = entries
  .sort((a, b) => (b[1] || 0) - (a[1] || 0))
```

```
        .slice(0, 3);

      if (topPeriods.length === 0) {
        return (
          <div className="rounded-xl bg-gradient-to-br from-blue-50 to-indigo-50 p-6 shadow-lg
border-2 border-blue-200">
            <h3 className="text-xl font-bold text-gray-900 mb-4 flex items-center">
              <span className="text-2xl mr-2">🎯</span>
              Прогноз найвищих періодів покупок ({bestMethod})
            </h3>
            <p className="text-gray-600">Недостатньо даних для рекомендацій по періодах.</p>
          </div>
        );
      }

      return (
        <div className="rounded-xl bg-gradient-to-br from-blue-50 to-indigo-50 p-6 shadow-lg
border-2 border-blue-200">
          <h3 className="text-xl font-bold text-gray-900 mb-4 flex items-center">
            <span className="text-2xl mr-2">🎯</span>
            Прогноз найвищих періодів покупок ({bestMethod})
          </h3>
          <p className="text-gray-600 mb-6">
            На основі найкращого методу, ось періоди, коли очікуються найвищі обсяги покупок:
          </p>
          <div className="grid grid-cols-1 md:grid-cols-3 gap-4">
            {topPeriods.map(([period, count], index) => {
              const total = entries.reduce((a, [, v]) => a + (v || 0), 0) || 1;
              const percentage = (((count || 0) / total) * 100).toFixed(1);
              return (
                <div
                  key={period}
                  className={`p-4 rounded-lg border-2 ${
                    index === 0
                      ? "bg-gradient-to-br from-yellow-50 to-orange-50 border-yellow-300"
                      : index === 1
                      ? "bg-gradient-to-br from-blue-50 to-indigo-50 border-blue-300"
                      : "bg-gradient-to-br from-green-50 to-emerald-50 border-green-300"
                  }`}
                >
                  <div className="flex items-center justify-between mb-2">
                    <span className="text-lg font-bold text-gray-900">
                      #{index + 1}
                    </span>
                    <span className="text-2xl">
                      {index === 0 ? "🥇" : index === 1 ? "🥈" : "🥉"}
                    </span>
                  </div>
                  <p className="text-xl font-bold text-gray-800 mb-1">{period}</p>
                  <p className="text-sm text-gray-600">
                    {count} прогнозів ({percentage}%)
```

```
                </p>
              </div>
            );
          })}
        </div>
        <div className="mt-6 p-4 bg-white rounded-lg border border-gray-200">
          <p className="text-sm text-gray-700">
            <strong>Рекомендація:</strong> Зосередьте маркетингові кампанії та управління
запасами
            на періоди <strong>{topPeriods[0]?.[0]}</strong>{topPeriods[1] ? ", " : ""}
            <strong>{topPeriods[1]?.[0]}</strong>{topPeriods[2] ? " та " : ""}
            <strong>{topPeriods[2]?.[0]}</strong> для максимального ефекту.
          </p>
        </div>
      </div>
    );
  })()}
 </div>
);
}
```