Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка програмного засобу прогнозування цін на косметичні засоби з використанням машинного навчання.

Виконав(ла): студент(ка) 4 курсу, групи СН-41
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

(підпис) Демчук М.В. (прізвище та ініціали)

Керівник (підпис) Палка О.В. (прізвище та ініціали)

Нормоконтроль (підпис) Липак Г.І. (прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О. (прізвище та ініціали)

Рецензент (підпис) Коноваленко І.В. (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
"8" червня 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

студенту Демчук Мар'яна Володимирівна
(прізвище, ім'я, по батькові)

1. Тема роботи **Розробка програмного засобу прогнозування цін на косметичні засоби з використанням машинного навчання.**

Керівник роботи **доктор філософії, асист. Палка Олег Вікторович**
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від "*14*" *травня* 2026 року № *4/9-239*

2. Термін подання студентом завершеної роботи 19 червня 2026 р.

3. Вихідні дані до роботи Літературні джерела з тематики роботи

4. Зміст роботи (перелік питань, які потрібно розробити)

Вступ; 1 Теоретичні основи моделей машинного навчання для прогнозування; 1.1 Загальна характеристика задачі прогнозування; 1.2 Лінійна регресія; 1.3 Дерево рішень (Decision Tree); 1.4 Випадковий ліс (Random Forest); 1.5 Метод опорних векторів (SVM); 1.6 Градієнтний бустинг (XGBoost); 1.7 Регуляризація: Ridge та Lasso; 1.8 Метрики оцінки якості моделей регресії; 2 Практична реалізація моделей прогнозування цін на косметичні засоби; 2.1 Збір та попередня обробка даних; 2.2 Збереження даних у базі даних SQLite; 2.3 Кодування категоріальних ознак та розділення даних; 2.4 Побудова та навчання моделей; 2.5 Порівняння результатів моделей; 2.6 Застосування регуляризації; 2.7 Прогнозування цін для нових даних; 3 Ілюстрація роботи програмного коду; 3.1 Середовище розробки та використані бібліотеки; 3.2 Візуалізація даних у Power BI; 3.3 Результати навчання моделей та аналіз графіків; 3.4 Практичне застосування та перспективи; 4. Безпека життєдіяльності, основи охорони праці; Висновки; Список використаних джерел; Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

Титульний слайд; Актуальність дослідження; Завдання дослідження; Огляд 5 моделей ML; Метрики оцінки якості; Конвеєр обробки даних; Порівняння моделей за R^2; Ridge vs Lasso; ТОП-10 брендів; Ціна vs рейтинг; Діагностика лінійної регресії; Прогнозування для нових даних; Підсумкова таблиця всіх моделей; Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Гурик О.Я., к.т.н., доцент кафедри МТ		

7. Дата видачі завдання 26 січня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	26.01.2026 – 27.01.2026	*Виконано*
2.	Підбір джерел по темі роботи	28.01.2026 – 01.04.2026	*Виконано*
3.	Оформлення першого розділу	15.04.2026	*Виконано*
4.	Оформлення другого розділу	20.04.2026	
5.	Оформлення третього розділу	30.04.2026	*Виконано*
6.	Виконання завдання до підрозділу "Безпека		
7.	життєдіяльності, основи охорони праці"	15.05.2026	*Виконано*
8.	Оформлення кваліфікаційної роботи	07.06.2026	*Виконано*
9.	Перевірка на плагіат	07.06.2026	*Виконано*
10.	Нормоконтроль	09.06.2026	*Виконано*
11.	Попередній захист кваліфікаційної роботи	11.06.2026	*Виконано*
12.	Захист кваліфікаційної роботи	22.06.2026	
13.			
14.			
15.			

Студент _____ Демчук М.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Палка О.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

"Розробка програмного засобу прогнозування цін на косметичні засоби з використанням машинного навчання" // Кваліфікаційна робота освітнього рівня "Бакалавр" // Демчук Мар'яна Володимирівна // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-41 // Тернопіль, 2026 // с. – 70, рис. – 15, таблиць – 2, джерел – 31, додатків – 1, сторінок додатків – 20.

Ключові слова: машинне навчання, прогнозування цін, регресія, Ridge-регресія, Lasso-регресія, XGBoost, Makeup API, SQLite, Power BI

Кваліфікаційна робота присвячена вирішенню актуальної практичної задачі – автоматизації аналізу та прогнозування цін на косметичну продукцію за допомогою сучасних алгоритмів машинного навчання. Актуальність дослідження зумовлена високим рівнем конкуренції, динамічністю ціноутворення на косметичному ринку та стрімким розвитком електронної комерції, що вимагає впровадження інтелектуальних інструментів для підтримки бізнес-рішень.

У роботі використано комплекс методів дослідження, що включає збір даних з веб-API, методи попередньої обробки даних (очищення, заповнення пропущених значень за груповим середнім, кодування категоріальних ознак за допомогою one-hot encoding), алгоритми машинного навчання (регресійні та ансамблеві моделі), методи регуляризації, а також інструменти візуалізації даних (бібліотека matplotlib та платформа Power BI). Інформаційною базою дослідження є відкриті дані сервісу Makeup API.

Реалізовано повний пайплайн обробки даних: збір інформації у форматі JSON з API-сервісу, очищення від аномалій (нульових цін), конвертація цін у єдину валюту та їх збереження у реляційній базі даних SQLite.

Побудовано, навчено та протестовано п'ять базових моделей машинного навчання. Порівняльний аналіз за метриками R^2, MSE та RMSE виявив перевагу лінійних методів (лінійної регресії та лінійного SVM), що свідчить про переважно лінійний характер залежностей між характеристиками продуктів (брендом, типом) та їх вартістю.

Досліджено вплив регуляризації на якість прогнозів: найкращі результати серед усіх підходів продемонструвала Ridge-регресія з параметром $\alpha = 0{,}5$ ($R^2 = 0{,}7256$, RMSE = 3,635 CAD), яка успішно знизила вплив мультиколінеарності ознак. Метод Lasso показав незадовільну якість через надмірне обнулення інформативних коефіцієнтів.

Проведено розширену бізнес-аналітику та інтерактивну візуалізацію даних у Power BI, яка виявила важливу ринкову закономірність: висока ціна косметичного засобу не завжди гарантує високий споживчий рейтинг.

Продемонстровано практичне застосування розробленої Ridge-моделі для сценарного прогнозування цін на основі нових комбінацій брендів та типів продукції з випадковим рейтингом.

ANNOTATION

"Development of a Software Tool for Predicting Cosmetic Product Prices Using Machine Learning" // Qualification work of the educational level "Bachelor" // Demchuk Mariana // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, Group CH-42 // Ternopil, 2026 // p. – 70, fig. – 15, tables – 2, references – 31, annexes – 1, pages for annexes – 20.

Keywords: machine learning, price forecasting, regression, Ridge regression, Lasso regression, XGBoost, Makeup API, SQLite, Power BI

The qualification thesis is dedicated to solving an urgent practical problem – automating the analysis and forecasting of prices for cosmetic products using modern machine learning algorithms. The relevance of the study is driven by a high level of competition, the dynamic nature of pricing in the cosmetics market, and the rapid development of e-commerce, which requires the implementation of intelligent tools for business decision support.

The thesis utilizes a comprehensive set of research methods, including data collection from a web API, data preprocessing techniques (cleaning, imputation of missing values using group means, and categorical feature encoding via one-hot encoding), machine learning algorithms (regression and ensemble models), regularization methods, as well as data visualization tools (the matplotlib library and the Power BI platform). The information base of the study consists of open data from the Makeup API service.

A complete data processing pipeline has been implemented: gathering information in JSON format from the API service, cleaning anomalies (zero prices), converting prices into a single currency (Canadian Dollar – CAD), and storing them in an SQLite relational database.

Five baseline machine learning models were constructed, trained, and tested. A comparative analysis using R^2, MSE, and RMSE metrics revealed the superiority of linear methods (linear regression and linear SVM), indicating a predominantly linear nature of the relationships between product characteristics (brand, type) and their cost.

The impact of regularization on forecast quality was investigated: Ridge regression with a parameter of $\alpha = 0,5$ ($R^2 = 0,7256$, RMSE = 3,635 CAD) demonstrated the best results among all approaches, successfully mitigating the effect of feature multicollinearity. The Lasso method showed unsatisfactory performance due to the excessive zeroing out of informative coefficients.

Advanced business analytics and interactive data visualization were performed in Power BI, revealing an important market pattern: a high price for a cosmetic product does not always guarantee a high consumer rating.

The practical application of the developed Ridge model was demonstrated for scenario-based price forecasting based on new combinations of brands and product types with a random rating.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Косметична індустрія є однією з найбільш динамічних галузей споживчого ринку, де ціноутворення залежить від багатьох факторів: бренду, типу продукту, рейтингу, країни походження та маркетингової стратегії. В умовах сучасного ринку, що характеризується високим рівнем конкуренції та постійними змінами цін, задача аналізу та прогнозування цін на косметичні засоби набуває особливого значення для прийняття ефективних бізнес-рішень.

За даними аналітичних агентств, глобальний ринок косметичної продукції оцінюється у сотні мільярдів доларів щорічно і продовжує динамічно зростати. Зростання електронної комерції та широка доступність інформації про продукти створюють нові можливості для автоматизованого аналізу цінових тенденцій. Водночас збільшення кількості брендів та товарних позицій ускладнює ручний моніторинг цін конкурентів і робить застосування методів машинного навчання необхідним інструментом для сучасного бізнесу.

Розвиток технологій машинного навчання відкриває нові можливості для вирішення задач прогнозування в різних галузях економіки. Методи машинного навчання дозволяють виявляти складні залежності між змінними, які неможливо встановити за допомогою традиційних статистичних методів. Застосування цих методів до аналізу ринку косметичної продукції дозволяє будувати моделі, здатні з прийнятною точністю передбачати ціни на основі наявних характеристик товарів.

Особливістю косметичного ринку є те, що ціна продукту визначається не лише функціональними характеристиками, а й такими факторами, як престиж бренду, маркетингова стратегія та позиціонування на ринку. Це створює складну багатофакторну залежність, для моделювання якої необхідні сучасні аналітичні інструменти.

Об’єктом дослідження є процес ціноутворення на ринку косметичної продукції та можливості його прогнозування з використанням сучасних методів аналізу даних.

Предметом дослідження є моделі машинного навчання для прогнозування цін на косметичні засоби: лінійна регресія, дерево рішень, випадковий ліс, метод опорних векторів, градієнтний бустинг XGBoost, а також методи регуляризації Ridge та Lasso.

Метою кваліфікаційної роботи є розробка та порівняльний аналіз моделей машинного навчання для прогнозування цін на косметичні засоби з використанням відкритих даних API-сервісу Makeup API.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– проаналізувати теоретичні основи моделей машинного навчання, що застосовуються для задач регресії та прогнозування;

– здійснити збір та попередню обробку даних про косметичні засоби з відкритого API-сервісу;

– побудувати та навчити декілька моделей машинного навчання для прогнозування цін;

– провести порівняльний аналіз побудованих моделей за метриками якості;

– дослідити вплив методів регуляризації на якість прогнозування;

– розробити практичні рекомендації щодо застосування побудованих моделей.

Методи дослідження. У роботі використано методи збору даних з веб-API, методи попередньої обробки даних (очищення, заповнення пропущених значень, кодування категоріальних ознак), методи машинного навчання (лінійна регресія, дерево рішень, випадковий ліс, SVM, XGBoost), методи регуляризації (Ridge, Lasso), а також методи візуалізації даних за допомогою matplotlib та Power BI.

Інформаційною базою дослідження є відкриті дані API-сервісу Makeup API, який містить інформацію про косметичні продукти різних брендів, включаючи ціни, рейтинги, типи продуктів та інші характеристики. Використання відкритого API забезпечує відтворюваність дослідження та можливість його верифікації.

Практичне значення результатів полягає у розробці прогнозної моделі, яка може бути використана компаніями косметичної галузі для аналізу цін конкурентів та формування власної цінової політики. Модель дозволяє прогнозувати ціну продукту на основі бренду, типу продукції та рейтингу, що є корисним інструментом при виведенні нових товарів на ринок.

Структура роботи. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків та списку використаних джерел. У першому розділі розглянуто теоретичні основи моделей машинного навчання для прогнозування. У другому розділі описано практичну реалізацію моделей на основі програмного коду. У третьому розділі наведено ілюстрацію роботи програмного коду та аналіз отриманих результатів. Загальний обсяг роботи складає 49 сторінок, містить 2 таблиці та список використаних джерел із 22 найменувань.

Апробація результатів. Основні результати роботи: побудована прогнозна модель Ridge-регресії з $R^2 = 0,7256$ та RMSE = 3,635 може бути використана для практичного аналізу ринку косметичної продукції. Запропонований пайплайн обробки даних (збір з API – очищення – кодування – навчання моделі – прогнозування) є відтворюваним та може бути адаптований для інших прикладних задач.

1 ТЕОРЕТИЧНІ ОСНОВИ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ ДЛЯ ПРОГНОЗУВАННЯ

1.1 Загальна характеристика задачі прогнозування

Прогнозування є одним із ключових завдань у сфері аналізу даних та машинного навчання. Під прогнозуванням розуміють процес передбачення значень цільової змінної на основі наявних даних та побудованої математичної моделі. У контексті машинного навчання задача прогнозування числових значень належить до класу задач регресії, на відміну від класифікації, де результатом є приналежність до певного класу.

Машинне навчання (Machine Learning) – це підгалузь штучного інтелекту, що досліджує алгоритми, здатні автоматично покращувати свою роботу на основі досвіду. На відміну від традиційного програмування, де правила задаються явно, алгоритми машинного навчання самостійно виявляють закономірності у даних і використовують їх для прогнозування. Це робить машинне навчання особливо ефективним для задач, де залежності між змінними є складними та важко формалізуються.

Задача регресії полягає у побудові функціональної залежності між набором вхідних ознак та цільовою змінною. Маючи набір спостережень, де кожне складається з вектора ознак та відповідного значення цільової змінної, необхідно побудувати функцію, яка мінімізує певну функцію втрат на тестових даних.

У контексті прогнозування цін на косметичні засоби задача полягає у визначенні ціни продукту на основі характеристик: бренду, типу продукту та рейтингу. Це є типовою задачею регресії, де ціна виступає цільовою змінною, а характеристики товару – предикторами. Складність полягає у тому, що частина предикторів є категоріальними (бренд, тип продукту), що вимагає спеціальної попередньої обробки.

Процес побудови прогнозної моделі включає кілька ключових етапів. Перший – збір та підготовка даних: отримання інформації, очищення від пропущених значень та аномалій, перетворення у формат, придатний для навчання. Другий – вибір та побудова моделі: визначення алгоритму та його гіперпараметрів. Третій – навчання моделі на тренувальних даних. Четвертий – оцінка якості на тестових даних за відповідними метриками.

Важливим аспектом є розділення даних на тренувальну та тестову вибірки. Тренувальна вибірка використовується для навчання моделі, тестова – для оцінки її узагальнюючої здатності. Типовим є співвідношення 80/20, що дозволяє уникнути перенавчання та отримати об'єктивну оцінку якості моделі на нових даних.

Перенавчання (overfitting) є однією з основних проблем машинного навчання. Воно виникає, коли модель занадто точно відтворює тренувальні дані, включаючи шум та випадкові відхилення, але втрачає здатність узагальнюватися на нових даних. Протилежна проблема – недонавчання (underfitting) – виникає, коли модель є занадто простою для захоплення реальних закономірностей.

Для розв'язання задачі прогнозування існує широкий спектр алгоритмів, кожен з яких має переваги та обмеження. Вибір конкретного алгоритму залежить від характеристик даних, обсягу вибірки, вимог до інтерпретованості моделі та обчислювальних ресурсів. У даному дослідженні порівнюються п'ять алгоритмів, що охоплюють різні підходи до машинного навчання.

Типовий пайплайн (pipeline) машинного навчання для задачі регресії включає такі кроки: збір та інтеграція даних з різних джерел; первинна розвідувальна аналітика (EDA – Exploratory Data Analysis) для розуміння структури та розподілів даних; попередня обробка та трансформація; розділення на тренувальну, валідаційну та тестову вибірки; вибір та навчання моделі; оцінка якості; інтерпретація результатів та розгортання моделі у виробничому середовищі.

Розвідувальна аналітика (EDA) є важливим етапом перед побудовою моделей. Вона включає аналіз розподілів числових змінних (побудова гістограм, box plots), виявлення викидів, аналіз кореляцій між ознаками та цільовою змінною, дослідження пропущених значень та їхніх патернів. У даному дослідженні EDA виконувалась за допомогою функції describe() бібліотеки pandas та візуалізацій Power BI.

Важливим поняттям у машинному навчанні є компроміс між зміщенням і дисперсією (bias-variance tradeoff). Зміщення – це систематична помилка моделі, зумовлена надмірним спрощенням. Дисперсія – чутливість моделі до незначних змін тренувальних даних. Прості моделі (лінійна регресія) мають велике зміщення та малу дисперсію, складні (глибокі дерева) – мале зміщення та велику дисперсію. Оптимальна модель знаходить баланс між двома крайнощами.

1.2 Лінійна регресія

Лінійна регресія є одним із найпростіших та найбільш поширених методів машинного навчання для розв'язання задач регресії. Ця модель базується на припущенні про лінійну залежність між вхідними ознаками та цільовою змінною. Математично модель описується рівнянням: $y = w_0 + w_1x_1 + w_2x_2 + ... + w_nx_n$, де y – прогнозоване значення, $x_1,..., x_n$ – вхідні ознаки, w_0 – вільний член, а $w_1,..., w_n$ – ваги моделі.

Метод найменших квадратів (Ordinary Least Squares, OLS) є стандартним підходом до навчання лінійної регресії. Він мінімізує суму квадратів різниць між прогнозованими та фактичними значеннями цільової змінної: $L(w) = \Sigma(y_i - \hat{y}_i)^2$. Оптимальне аналітичне рішення має вигляд: $w = (X^TX)^{-1}X^Ty$, де X – матриця ознак. Альтернативно використовується ітераційний метод градієнтного спуску, ефективніший для великих наборів даних.

Лінійна регресія є одним із найстаріших статистичних методів, незалежно розроблених Карлом Фрідріхом Гауссом та Адрієном-Марі Лежандром на початку XIX століття. Незважаючи на свою простоту, метод залишається широко використовуваним інструментом у сучасному машинному навчанні завдяки інтерпретованості та обчислювальній ефективності.

Перевагами лінійної регресії є простота інтерпретації результатів, висока швидкість навчання та прогнозування, стійкість до шуму в даних та можливість аналізу впливу окремих факторів. Кожен ваговий коефіцієнт показує, на скільки одиниць зміниться прогнозоване значення при зміні відповідної ознаки на одиницю за фіксованості решти.

Обмеженнями є: припущення про лінійність залежності, чутливість до мультиколінеарності (високої кореляції між ознаками), що призводить до нестабільності коефіцієнтів. Модель також чутлива до викидів, які можуть суттєво вплинути на положення регресійної площини. Для усунення проблеми мультиколінеарності застосовуються методи регуляризації Ridge та Lasso, розглянуті у підрозділі 1.7.

У контексті прогнозування цін на косметичні засоби лінійна регресія встановлює базові залежності між характеристиками товарів та їхніми цінами. Попри простоту, модель часто демонструє прийнятну якість, особливо коли залежність між ціною та ознаками є переважно лінійною. Лінійна регресія також слугує базовим рівнем (baseline), з яким порівнюють більш складні моделі.

Важливою властивістю лінійної регресії є можливість тестування статистичної значущості коефіцієнтів за допомогою t-тестів та F-тесту. t-тест перевіряє нульову гіпотезу про те, що конкретний коефіцієнт дорівнює нулю (ознака не впливає на цільову змінну). F-тест перевіряє загальну значущість моделі. Ці тести є корисними інструментами для інтерпретації та відбору ознак у статистичній регресії, хоча в машинному навчанні зазвичай більший акцент робиться на прогностичній точності.

Припущення лінійної регресії (лінійність, нормальність залишків, гомоскедастичність, відсутність мультиколінеарності та автокореляції) є важливими для коректної інтерпретації коефіцієнтів у статистичному контексті. Однак у машинному навчанні, де головною метою є мінімізація прогностичної помилки, порушення цих припущень є менш критичним: навіть при їхньому недотриманні модель може давати корисні прогнози.

1.3 Дерево рішень (Decision Tree)

Дерево рішень – це непараметричний метод машинного навчання, що використовується для задач класифікації та регресії. У контексті регресії алгоритм DecisionTreeRegressor будує деревоподібну структуру, де кожен внутрішній вузол відповідає умові розгалуження за однією з ознак, а кожен листовий вузол містить прогнозоване значення – середнє значень цільової змінної для спостережень, що потрапили до цього вузла.

Побудова дерева відбувається шляхом рекурсивного розбиття простору ознак на частини. На кожному кроці алгоритм обирає ознаку та порогове значення, які мінімізують середньоквадратичну помилку у результуючих підмножинах. Для кожного можливого розбиття обчислюється зважена сума MSE у лівій та правій підмножинах, і обирається розбиття з мінімальним значенням.

Алгоритм CART (Classification and Regression Trees) працює за принципом жадібного пошуку (greedy search): на кожному кроці обирається локально оптимальне розбиття, яке не обов’язково призводить до глобально оптимального дерева. Це компроміс між якістю та обчислювальною ефективністю, оскільки пошук глобально оптимального дерева є NP-складною задачею.

Ключовими перевагами дерев рішень є: інтуїтивна зрозумілість та легкість інтерпретації; відсутність необхідності масштабування ознак;

здатність обробляти числові та категоріальні змінні; автоматичне врахування нелінійних залежностей та взаємодій між ознаками.

Основним недоліком є схильність до перенавчання: глибоке дерево без обмежень ідеально відтворює тренувальні дані, але погано узагальнюється. Для боротьби використовуються: обрізка дерева (pruning), обмеження максимальної глибини (max_depth), мінімальна кількість спостережень у вузлі та листі (min_samples_split, min_samples_leaf).

Ще одним обмеженням є нестабільність: невеликі зміни у тренувальних даних можуть призвести до суттєво іншого дерева. Ця проблема вирішується ансамблевими методами – випадковим лісом та XGBoost, які об’єднують прогнози багатьох дерев для отримання більш стабільного та точного результату.

Практична рекомендація при роботі з деревами рішень – завжди налаштовувати глибину дерева. Нефіксована глибина призводить до виродженого дерева, де кожен листовий вузол містить лише одне або кілька спостережень. Це ознака явного перенавчання. Хорошою відправною точкою є max_depth від 3 до 7; оптимальне значення підбирається крос-валідацією.

Інтерпретація дерева рішень може бути виконана шляхом його візуалізації через sklearn.tree.plot_tree або graphviz. Кожен вузол показує умову розбиття, поточне значення MSE та кількість спостережень. Аналіз кореневого вузла (першого розбиття) показує найбільш інформативну ознаку – ту, що найбільше зменшує MSE при розбитті вибірки на дві частини.

У даному дослідженні дерево рішень без обмежень глибини продемонструвало очікуваний результат: ідеальне або близьке до ідеального відтворення тренувальних даних, але суттєво нижча якість на тестових. Це ілюструє класичну проблему перенавчання та підкреслює важливість контролю складності моделі.

1.4 Випадковий ліс (Random Forest)

Випадковий ліс – ансамблевий метод машинного навчання, запропонований Лео Брейманом у 2001 році, що базується на побудові та агрегації великої кількості дерев рішень. Він є одним із найуспішніших алгоритмів завдяки поєднанню високої точності, стійкості до перенавчання та здатності обробляти дані великої розмірності.

Ансамблеві методи об'єднують прогнози кількох базових моделей для отримання більш точного результату. Помилки окремих моделей компенсують одна одну при усередненні, що зменшує загальну помилку. Випадковий ліс належить до методів бегінгу (bagging – bootstrap aggregating): кожне дерево навчається на бутстреп-вибірці (випадковій підвибірці з поверненням).

Ключова ідея: кожне дерево будується на випадковій підвибірці тренувальних даних та з використанням випадкової підмножини ознак при кожному розгалуженні. Це забезпечує декореляцію дерев в ансамблі. Фінальний прогноз для задачі регресії – середнє значень прогнозів усіх дерев.

Ключовим гіперпараметром є кількість дерев в ансамблі (n_estimators). Збільшення кількості покращує якість, але зростає час навчання. На практиці зазвичай використовують від 100 до 500 дерев. У даному дослідженні використано n_estimators=100 з random_state=42 для відтворюваності.

Іншими важливими гіперпараметрами є: max_depth (максимальна глибина), max_features (кількість ознак при розбитті), min_samples_split та min_samples_leaf. Ці параметри контролюють складність окремих дерев та впливають на баланс між зміщенням і дисперсією моделі.

Перевагами є: висока точність, стійкість до перенавчання, здатність оцінювати важливість ознак (feature importance), ефективна обробка великих наборів даних. Недоліки: менша інтерпретованість порівняно з одинарним деревом, більший час навчання, значні вимоги до оперативної пам'яті при великій кількості дерев.

Оцінка важливості ознак (feature importance) у випадковому лісі обчислюється як середнє зменшення нечистоти (mean decrease in impurity, MDI) по всіх деревах. Для задачі регресії це середнє зменшення MSE при використанні ознаки для розбиття. Ця метрика дозволяє визначити, які ознаки найбільше впливають на прогнозовану ціну: наприклад, бренд може мати вищу важливість, ніж тип продукту, або навпаки.

Out-of-Bag (OOB) оцінка є додатковою перевагою бегінгу. Оскільки кожне дерево навчається приблизно на 63% унікальних спостережень (решта 37% не потрапляє до бутстреп-вибірки), ці «залишені» спостереження можна використати для оцінки якості без окремої тестової вибірки. OOB-оцінка є гарним наближенням до результатів крос-валідації і не потребує додаткових обчислень.

1.5 Метод опорних векторів (SVM)

Метод опорних векторів (Support Vector Machine, SVM) – потужний алгоритм машинного навчання, спочатку розроблений для бінарної класифікації, але адаптований і для регресії (SVR – Support Vector Regression). Запропонований Володимиром Вапніком у 1995 році, метод здобув широке поширення завдяки ефективності у просторах високої розмірності.

SVR знаходить функцію, яка відхиляється від фактичних значень цільової змінної не більше ніж на величину epsilon для кожного спостереження, залишаючись при цьому максимально «плоскою». Спостереження, що визначають положення гіперплощини, називаються опорними векторами – лише вони впливають на модель, що забезпечує ефективне використання пам’яті.

Важливою особливістю є можливість використання ядерних функцій (kernel trick) для обробки нелінійних залежностей. Ядерна функція неявно перетворює дані у простір більшої розмірності. Найпоширеніші ядра: лінійне $K(x,x')=x^Tx'$, поліноміальне, RBF (радіально-базисне) та сигмоїдне.

У даному дослідженні використовується лінійне ядро (kernel=’linear’). Це обумовлено тим, що після one-hot encoding кількість ознак є значною, а лінійне ядро є найефективнішим у таких умовах. Використання нелінійних ядер було б обчислювально витратним без гарантії покращення якості.

Перевагами SVR є: ефективність у просторах високої розмірності, стійкість до перенавчання, здатність обробляти нелінійні залежності. Недоліки: чутливість до масштабу ознак, висока обчислювальна складність $O(n^2)$–$O(n^3)$ для великих наборів даних, складність інтерпретації результатів та необхідність ретельного підбору гіперпараметрів C та epsilon.

Параметр C у SVR контролює компроміс між мінімізацією помилки та максимізацією зазору (margin). При малому C модель допускає більше помилок, але є простішою. При великому C модель намагається правильно апроксимувати всі тренувальні спостереження, що підвищує ризик перенавчання. Параметр epsilon визначає ширину «трубки» (epsilon-insensitive tube) навколо регресійної функції: помилки всередині трубки не штрафуються.

Важливою вимогою для SVR є масштабування ознак. Оскільки алгоритм базується на відстанях між точками у просторі ознак, ознаки з великими значеннями можуть домінувати. Для числових ознак рекомендується StandardScaler або MinMaxScaler. У даному дослідженні більшість ознак після one-hot encoding є бінарними (0 або 1), тому ефект масштабування є меншим. Проте ознака рейтингу (значення від 1 до 5) теоретично потребує нормалізації.

1.6 Градієнтний бустинг (XGBoost)

Градієнтний бустинг – ансамблевий метод, що будує послідовність слабких моделей (дерев рішень), де кожна наступна виправляє помилки попередніх. На відміну від бегінгу (де моделі будуються незалежно), у бустингу моделі будуються послідовно, і кожна нова фокусується на помилках попередніх. XGBoost (eXtreme Gradient Boosting) – оптимізована реалізація, розроблена Тяньці Ченом у 2016 році.

Основна ідея: на кожному кроці до ансамблю додається нова модель, що навчається на залишках (residuals) – різницях між фактичними значеннями та поточними прогнозами. Кожна нова модель фокусується на спостереженнях з найбільшими помилками. Фінальний прогноз – сума прогнозів усіх моделей, зважена коефіцієнтом швидкості навчання.

XGBoost додає до базового бустингу важливі оптимізації: регуляризацію (L1 та L2) для запобігання перенавчанню; паралельну обробку для прискорення навчання; ефективну обробку пропущених значень; вбудований механізм крос-валідації; ефективне використання кешу процесора для операцій над деревами.

XGBoost здобув надзвичайну популярність завдяки численним перемогам у змаганнях з аналізу даних на платформі Kaggle і став одним зі стандартних інструментів у арсеналі фахівців з Data Science. У даному дослідженні XGBRegressor використовується з параметрами за замовчуванням та random_state=42.

Ключові гіперпараметри: n_estimators (кількість дерев), learning_rate (швидкість навчання), max_depth (глибина дерев), reg_alpha та reg_lambda (L1 та L2 регуляризація), min_child_weight (мінімальна вага листового вузла). Ретельний підбір цих параметрів за допомогою крос-валідації може суттєво покращити якість моделі.

Перевагами є: висока точність, вбудована регуляризація, ефективна обробка великих наборів даних, здатність обробляти пропущені значення, оцінка важливості ознак. Недоліки: складність інтерпретації, чутливість до гіперпараметрів, потенційне перенавчання при великій кількості ітерацій без регуляризації.

Важливим практичним аспектом XGBoost є використання ранньої зупинки (early stopping). При ввімкненні early_stopping_rounds XGBoost автоматично зупиняє навчання, якщо якість на валідаційній вибірці не покращується протягом заданої кількості ітерацій. Це запобігає перенавчанню

та дозволяє використовувати більші значення n_estimators без ризику деградації якості.

XGBoost підтримує паралельне навчання через параметр n_jobs=-1 (використання всіх доступних ядер процесора). Це суттєво прискорює навчання на багатоядерних машинах. Також бібліотека підтримує GPU-прискорення через параметр device='cuda', що дозволяє прискорити навчання у десятки разів для великих наборів даних.

1.7 Регуляризація: Ridge та Lasso

Регуляризація – техніка для запобігання перенавчанню моделей машинного навчання шляхом додавання штрафного члена до функції втрат. Вона контролює складність моделі, обмежуючи величину вагових коефіцієнтів, що дозволяє знайти баланс між точністю на тренувальних даних та здатністю до узагальнення. Два найпоширеніші методи для лінійної регресії – Ridge (L2) та Lasso (L1).

Ridge-регресія (гребенева регресія, L2-регуляризація) мінімізує: $L(w) = \Sigma(y_i - \hat{y}_i)^2 + \alpha\Sigma w_j^2$. Штрафний член пропорційний сумі квадратів коефіцієнтів, що зменшує їхню величину, але не обнулює. Ridge була запропонована Хорлом та Кеннардом у 1970 році для вирішення проблеми мультиколінеарності, яка призводить до нестабільних оцінок коефіцієнтів. Параметр alpha контролює силу регуляризації.

Lasso-регресія (Least Absolute Shrinkage and Selection Operator, L1-регуляризація), запропонована Тібширані у 1996 році, мінімізує: $L(w) = \Sigma(y_i - \hat{y}_i)^2 + \alpha\Sigma|w_j|$. Ключовою особливістю є здатність обнулювати деякі коефіцієнти, здійснюючи автоматичний відбір ознак (feature selection). Це робить модель більш інтерпретованою, оскільки визначає найважливіші предиктори.

Вибір між Ridge та Lasso залежить від характеристик даних. Ridge краще працює, коли більшість ознак є інформативними. Lasso оптимальне, коли лише невелика частина ознак є релевантною. Існує також Elastic Net – комбінація

Ridge та Lasso: $L(w) = \Sigma(y_i - \hat{y}_i)^2 + \alpha_1\Sigma|w_j| + \alpha_2\Sigma w_j^2$, що поєднує переваги обох методів.

При надмірному значенні alpha Lasso може обнулити занадто багато коефіцієнтів, видаливши важливі ознаки. Зокрема, при наявності групи сильно корельованих ознак Lasso зазвичай обирає лише одну з них, що може призвести до нестабільності моделі. Ridge у такому випадку розподіляє ваги між усіма корельованими ознаками.

Параметр alpha зазвичай підбирається за допомогою крос-валідації. Оптимальне значення забезпечує баланс між складністю моделі та здатністю до узагальнення. Занадто мале alpha дає незначний ефект регуляризації, занадто велике – надмірно спрощує модель.

Scikit-learn надає зручні класи з вбудованою крос-валідацією: RidgeCV та LassoCV. Ці класи автоматично підбирають оптимальне значення alpha з переданого списку кандидатів: ridge_cv = RidgeCV(alphas=[0.001, 0.01, 0.1, 0.5, 1.0, 5.0, 10.0]); ridge_cv.fit(X_train, y_train); print(ridge_cv.alpha_). Після навчання атрибут alpha_ містить оптимальне значення, підібране крос-валідацією.

Геометрична інтерпретація регуляризації допомагає зрозуміти різницю між Ridge та Lasso. Ridge-регуляризація відповідає обмеженню у вигляді кулі (сфери) у просторі коефіцієнтів, Lasso – у вигляді ромба (L1 куля). Оптимальне рішення знаходиться там, де рівні лінії функції втрат торкаються обмежувального тіла. Для Lasso кути ромба з нульовими компонентами є більш вірогідними точками дотику, що й призводить до обнулення коефіцієнтів.

Elastic Net – $L(w) = MSE + \alpha_1\Sigma|w_j| + \alpha_2\Sigma w_j^2$ – поєднує переваги Ridge та Lasso. Він здійснює відбір ознак (як Lasso), але при наявності групи корельованих ознак включає всі з них (як Ridge), а не лише одну. Elastic Net реалізований у scikit-learn класом ElasticNet та його CV-варіантом ElasticNetCV. Для даного набору даних Elastic Net міг би бути

перспективнішим, ніж Lasso, оскільки деякі бренди та типи продуктів є взаємозамінними.

1.8 Метрики оцінки якості моделей регресії

Для оцінки якості прогнозних моделей регресії використовується ряд метрик, кожна з яких відображає різні аспекти точності прогнозування. У даному дослідженні застосовуються три основні метрики: коефіцієнт детермінації R^2, середньоквадратична помилка MSE та корінь з середньоквадратичної помилки RMSE.

Коефіцієнт детермінації R^2 (R-squared) показує, яку частку дисперсії цільової змінної пояснює модель: $R^2 = 1 - SS_r{}^o{}_i/SS_{tot}$, де $SS_r{}^o{}_i$ – сума квадратів залишків, SS_{tot} – загальна сума квадратів відхилень від середнього. Значення 1 означає ідеальне прогнозування, 0 – модель не краща за просте середнє, від'ємні значення вказують на гіршу якість за просте середнє.

Середньоквадратична помилка MSE (Mean Squared Error) обчислюється як: $MSE = (1/n)\Sigma(y_i - \hat{y}_i)^2$. MSE штрафує великі помилки більше, ніж малі (завдяки зведенню у квадрат), що робить її корисною, коли великі відхилення є небажаними. Недолік – одиниці виміру рівні квадрату одиниць цільової змінної, що ускладнює інтерпретацію.

Корінь з середньоквадратичної помилки $RMSE = \sqrt{MSE}$ має ті самі одиниці виміру, що й цільова змінна, що робить її більш інтерпретованою. RMSE показує середнє відхилення прогнозу від фактичного значення і дозволяє безпосередньо оцінити очікувану похибку прогнозування у конкретних одиницях (канадських доларах у даному дослідженні).

Діаграма розсіювання прогнозованих значень проти фактичних (scatter plot: True values vs Predicted values) дозволяє візуально оцінити якість прогнозування – точки повинні групуватися навколо діагоналі $y = x$. Відхилення від діагоналі вказують на систематичні помилки. Вигнута лінія може свідчити про нелінійну залежність, яку лінійна модель не захоплює.

Графік залишків (residuals plot) показує різницю між фактичними та прогнозованими значеннями. В ідеальному випадку залишки мають бути випадково розподілені навколо нуля. Конусоподібна форма (збільшення дисперсії залишків зі зростанням прогнозованих значень) вказує на гетероскедастичність, що може вимагати логарифмічного перетворення цільової змінної або зважених методів регресії.

Додатковими метриками, що можуть використовуватись для оцінки регресійних моделей, є: MAE (Mean Absolute Error) – середнє абсолютне відхилення, яке є менш чутливим до викидів, ніж MSE; MAPE (Mean Absolute Percentage Error) – середня абсолютна похибка у відсотках від фактичного значення, зручна для порівняння моделей, що прогнозують змінні різних масштабів. У даному дослідженні основними метриками є R^2, MSE та RMSE, що відповідає стандартним практикам оцінки регресійних моделей у scikit-learn.

Важливим аспектом оцінки є вибір між метриками в залежності від специфіки задачі. Для ціноутворення на косметику RMSE є особливо інформативним: значення RMSE = 3.635 CAD означає, що в середньому прогноз відрізняється від реальної ціни менш ніж на 3.64 канадських долари, що для більшості косметичних засобів є прийнятною точністю. Для порівняння якості моделей R^2 є більш зручним – він нормований і не залежить від шкали цільової змінної.

2 ПРАКТИЧНА РЕАЛІЗАЦІЯ МОДЕЛЕЙ ПРОГНОЗУВАННЯ ЦІН НА КОСМЕТИЧНІ ЗАСОБИ

2.1 Збір та попередня обробка даних

Джерелом даних є відкритий API-сервіс Makeup API (http://makeup-api.herokuapp.com/api/v1/products.json), що надає інформацію про косметичні продукти різних брендів. API (Application Programming Interface) – це інтерфейс прикладного програмування, який дозволяє програмним застосункам взаємодіяти між собою. REST API повертає дані у форматі JSON у відповідь на HTTP GET-запит.

JSON (JavaScript Object Notation) – легкий текстовий формат обміну даними, зрозумілий для людини та ефективний для програмної обробки. Кожен запис API містить: brand (назва бренду), price (ціна), currency (валюта), rating (рейтинг покупців), product_type (тип продукту), updated_at (дата оновлення) та інші характеристики.

Для отримання даних використовуються бібліотеки Python: requests для HTTP-запитів та json для розбору відповіді. Після отримання JSON-відповіді вона перетворюється у список словників з вибраними полями, який потім конвертується у pandas DataFrame (див. лістинг 2.1, рис. 2.1):

Лістинг 2.1 – Отримання даних

```
url = 'http://makeup-api.herokuapp.com/api/v1/products.json'
response = requests.get(url)
data = json.loads(response.text)

brand_price_list = [{'brand': item['brand'],
                     'price': item['price'],
                     'currency': item['currency'],
                     'rating': item['rating'],
                     'product_type': item['product_type'],
                     'updated_date': pd.to_datetime(
                       item['updated_at']).strftime('%d-%m-%Y')}
```

```
                        for item in data]
df = pd.DataFrame(brand_price_list)
```

Обробка даних

```
50]: # get data from json, put it into list, transform list to dataframe
brand_price_list = [{'brand': item['brand'],
                     'price': item['price'],
                     'currency': item['currency'],
                     'rating': item['rating'],
                     'product_type': item['product_type'],
                     'updated_date': pd.to_datetime(item['updated_at']).strftime('%d-%m-%Y')}
                    for item in data]

df = pd.DataFrame(brand_price_list)

df.head()
```

50]:

	brand	price	currency	rating	product_type	updated_date
0	colourpop	5.0	CAD	NaN	lip_liner	09-07-2018
1	colourpop	5.5	CAD	NaN	lipstick	09-07-2018
2	colourpop	5.5	CAD	NaN	lipstick	09-07-2018
3	colourpop	12.0	CAD	NaN	foundation	09-07-2018
4	boosh	26.0	CAD	NaN	lipstick	02-09-2018

Рисунок 2.1 – Код отримання даних з API та перші рядки набору даних (df.head())

Після створення DataFrame виконується процедура очищення та підготовки даних. Обробка пропущених значень є критичним етапом: більшість алгоритмів машинного навчання не можуть працювати з даними, що містять NaN. Існує кілька стратегій: видалення записів, заповнення середнім/медіаною, групове заповнення. У даному дослідженні застосовано комбінований підхід.

Пропущені значення у колонці currency заповнюються значенням CAD (канадський долар), оскільки більшість продуктів має ціну саме в цій валюті. Для рейтингу використовується групове заповнення: пропущені значення замінюються середнім рейтингом для тієї самої комбінації бренду та типу

продукту. Це дозволяє зберегти більше даних, враховуючи специфіку ринкового сегменту (див. лістинг 2.2, рис. 2.2):

Лістинг 2.2 – Опрацювання пропущених значень

```
df['currency'] = df['currency'].fillna('CAD')
df['rating'] = df.groupby(['brand', 'product_type'])
    ['rating'].transform(lambda x: x.fillna(x.mean()))
df = df.dropna()
df = df.reset_index(drop=True)
```

```
[561]: # Fill NaN values in 'currency' column with 'CAD'
df['currency'] = df['currency'].fillna('CAD')

# Fill NaN values for 'rating' column with mean rating for same 'brand' and 'product_type'
df['rating'] = df.groupby(['brand', 'product_type'])['rating'].transform(lambda x: x.fillna(x.mean()))

# Drop NaN values and reset index
df = df.dropna()
df = df.reset_index(drop=True)

df.head()
```

[561]:

	brand	price	currency	rating	product_type	updated_date
0	nyx	10.0	USD	4.50	foundation	24-12-2017
1	nyx	12.0	USD	4.50	foundation	24-12-2017
2	nyx	12.0	USD	4.50	foundation	24-12-2017
3	nyx	12.0	USD	4.50	foundation	24-12-2017
4	nyx	10.0	USD	4.50	foundation	24-12-2017

Рисунок 2.2 – Заповнення пропущених значень та результат df.head() після очищення

Важливим етапом є конвертація цін у єдину валюту. Оскільки ціни представлені у двох валютах – CAD та USD – для коректного порівняння всі ціни конвертуються у канадські долари за курсом 1 USD = 1,27 CAD. Результат зберігається у новій колонці converted_price, яка стане цільовою змінною для моделей (лістинг 2.3):

Лістинг 2.3 – Конвертація цін

```
df['price'] = df['price'].astype(float)
df['converted_price'] = df.apply(
    lambda row: row['price'] if row['currency']=='CAD'
    else row['price']/1.27 if row['currency']=='USD'
    else None, axis=1)
```

Також видаляються записи з нульовою ціною (df = df[df['price'] != 0]), які можуть бути результатом помилок у даних або безкоштовними зразками, не репрезентативними для аналізу цін. Функція df.describe() використовується для отримання описової статистики: середнє, стандартне відхилення, мінімум, максимум та квартилі числових змінних.

Після всіх етапів очищення отримується чистий набір даних готовий для подальшого аналізу та побудови моделей. Характеристики очищеного датасету: дані охоплюють продукти кількох десятків брендів та різних типів косметики, зокрема губна помада, тушша, підводка, тіні для повік, тональні засоби та ін.

Описова статистика очищеного набору даних, отримана через df.describe(), дозволяє отримати уявлення про розподіл ключових змінних. Середня ціна продуктів у канадських доларах, діапазон від мінімальних бюджетних товарів до преміальних позицій, розподіл рейтингів від 1 до 5 – ці характеристики визначають складність задачі прогнозування та допомагають вибрати відповідні алгоритми.

Унікальні значення валюти (df[’currency’].unique()) показали наявність лише двох валют у датасеті: CAD та USD. Це спростило процес конвертації та унеможливило необхідність роботи з множиною валют. Аналіз розподілу ціни після конвертації (через df.describe()) дозволив виявити потенційні аномалії – продукти з нульовою ціною, які були видалені з набору даних.

2.2 Збереження даних у базі даних SQLite

Після підготовки даних вони зберігаються у реляційній базі даних SQLite для структурованого зберігання та повторного використання. SQLite – легка безсерверна СУБД, що зберігає всю базу у єдиному файлі. На відміну від серверних СУБД (PostgreSQL, MySQL), SQLite не потребує окремого процесу

сервера, що робить її ідеальною для навчальних проєктів та вбудованих застосунків.

Для роботи з SQLite використовується вбудована бібліотека Python sqlite3, що реалізує стандартний інтерфейс DB-API 2.0. Це забезпечує уніфікований інтерфейс для роботи з різними СУБД. При підключенні до неіснуючого файлу база даних створюється автоматично.

Структура таблиці makeup_data включає поля: updated_date (DATE), brand (TEXT), price (FLOAT), currency (TEXT), rating (FLOAT), product_type (TEXT), converted_price (FLOAT). Код створення та заповнення бази (лістинг 2.4):

Лістинг 2.4 – Створення таблиці makeup_data

```
conn = sqlite3.connect('makeup_data.db')
c = conn.cursor()
c.execute('DROP TABLE IF EXISTS makeup_data')
c.execute('''CREATE TABLE IF NOT EXISTS makeup_data
  (updated_date DATE, brand TEXT, price FLOAT,
   currency TEXT, rating FLOAT,
   product_type TEXT, converted_price FLOAT)''')
for index, row in df.iterrows():
    values = (row['updated_date'], row['brand'],
              row['price'], row['currency'],
              row['rating'], row['product_type'],
              row['converted_price'])
    c.execute(
      'INSERT INTO makeup_data VALUES (?,?,?,?,?,?,?)',
      values)
conn.commit()
conn.close()
```

Використання параметризованих запитів (символ ? замість прямого вставлення значень) забезпечує захист від SQL-ін’єкцій та коректну обробку спеціальних символів. Після запису коректність збереження перевіряється повторним читанням через SELECT * FROM makeup_data та відображенням у вигляді DataFrame.

SQLite підтримує всі основні типи SQL-запитів: SELECT з фільтруванням (WHERE), агрегацією (GROUP BY, AVG, COUNT) та сортуванням (ORDER BY). Наприклад, SELECT brand, AVG(converted_price) as avg_price FROM makeup_data GROUP BY brand ORDER BY avg_price DESC дозволяє отримати середню ціну за кожним брендом безпосередньо на рівні бази даних, без завантаження всіх даних у пам’ять Python.

Перевагами SQLite порівняно з CSV або Excel є: структурованість з типізацією полів; ефективне фільтрування та агрегація через SQL; транзакційність (commit/rollback забезпечує атомарність); компактний бінарний формат. Після збереження наступний запуск програми може зчитувати дані безпосередньо з бази, уникаючи повторного звернення до API – особливо важливо при нестабільному API або обмеженнях на кількість запитів.

2.3 Кодування категоріальних ознак та розділення даних

Перед навчанням моделей необхідно перетворити категоріальні ознаки brand та product_type у числовий формат. Більшість алгоритмів машинного навчання працюють лише з числовими даними, тому категоріальні ознаки потребують кодування.

One-hot encoding (однорозрядне кодування) – метод, що створює окрему бінарну колонку для кожного унікального значення категоріальної ознаки. Для кожного спостереження відповідна колонка має значення 1, решта – 0. Альтернативою є Label Encoding, що присвоює числові мітки категоріям, але він створює помилкову ординальну залежність (наче бренд B є «більшим» за бренд A), тому one-hot є кращим вибором (лістинг 2.5):

Лістинг 2.5 – One-hot encoding (однорозрядне кодування)

```
df_encoded = pd.get_dummies(df,
    columns=['brand', 'product_type'])
```

Після кодування дані розділяються на тренувальну (80%) та тестову (20%) вибірки за допомогою train_test_split. Параметр random_state=42 забезпечує відтворюваність – при кожному запуску дані розділяються однаково (лістинг 2.6).

Лістинг 2.6 – Поділ даних на тренувальну та тестову вибірки

```
train_df, test_df = train_test_split(
    df_encoded, test_size=0.2, random_state=42)


X_train = train_df.drop(
    columns=['converted_price', 'updated_date',
             'price', 'currency']).values
X_test = test_df.drop(
    columns=['converted_price', 'updated_date',
             'price', 'currency']).values
y_train = train_df['converted_price'].values
y_test = test_df['converted_price'].values
```

З набору ознак видаляються: converted_price (цільова змінна), updated_date (не інформативна для прогнозування), price (до конвертації) та currency (вже врахована у converted_price). Метод .values перетворює DataFrame у масив NumPy – стандартний формат вхідних даних для алгоритмів scikit-learn. Залишаються ознака рейтингу та бінарні колонки брендів і типів продуктів.

Важливим питанням при one-hot encoding є проблема мультиколінеарності: якщо ознака має k унікальних значень, after кодування k стовпців є лінійно залежними (сума всіх бінарних стовпців для кожного запису завжди дорівнює 1 для кожної ознаки). Це може призвести до нестабільності лінійної регресії. Функція pd.get_dummies() за замовчуванням не видаляє один з стовпців (параметр drop_first=False), але параметр drop_first=True видалить перший стовпець кожної ознаки, усуваючи лінійну залежність.

Розмірність матриці ознак після one-hot encoding є важливою характеристикою. Якщо датасет містить N брендів та M типів продуктів,

матриця X матиме (N + M + 1) стовпців (N для брендів, M для типів продуктів, 1 для рейтингу). Велика розмірність матриці ознак підвищує ризик мультиколінеарності та перенавчання для простих моделей, але алгоритми з регуляризацією (Ridge) ефективно вирішують цю проблему.

Збереження логіки кодування є критичним для коректного прогнозування. У даному дослідженні для прогнозування нових даних використовується той самий виклик pd.get_dummies(), що й при навчанні, але на нових даних. Цей підхід може призвести до розбіжностей у кількості стовпців, якщо нові дані містять бренди або типи продуктів, відсутні у тренувальному наборі, або не містять деяких брендів. У виробничому середовищі рекомендується використовувати sklearn.preprocessing.OneHotEncoder, який зберігає словник категорій і завжди створює однаковий набір стовпців.

2.4 Побудова та навчання моделей

У дослідженні побудовано п’ять моделей машинного навчання для прогнозування цін. Кожна навчається на тренувальній вибірці та оцінюється на тестовій за метриками R^2, MSE та RMSE. Уніфікований підхід дозволяє коректно порівняти моделі між собою.

Лінійна регресія реалізована за допомогою класу LinearRegression зі sklearn.linear_model. Модель не має гіперпараметрів для налаштування і знаходить оптимальні коефіцієнти методом найменших квадратів (лістинг 2.7):

Лістинг 2.6 – Реалізація лінійної регресії

```
model_lr = LinearRegression()
model_lr.fit(X_train, y_train)
y_pred = model_lr.predict(X_test)
r2 = model_lr.score(X_test, y_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
```

Дерево рішень реалізоване через DecisionTreeRegressor зі sklearn.tree. Використовуються параметри за замовчуванням (без обмеження глибини), що дозволяє оцінити базову продуктивність алгоритму. На практиці рекомендується обмежувати глибину для запобігання перенавчанню.

Випадковий ліс реалізований через RandomForestRegressor з n_estimators=100 та random_state=42. Ансамбль із 100 дерев забезпечує стабільний прогноз завдяки усередненню. Для задачі такої розмірності 100 дерев є розумним компромісом між якістю та часом навчання.

XGBoost реалізований через XGBRegressor з random_state=42 та параметрами за замовчуванням. XGBoost автоматично визначає кількість ітерацій та використовує регуляризацію, що робить його стійким до перенавчання навіть без ручного налаштування.

SVM для регресії реалізований через SVR(kernel='linear') зі sklearn.svm. Лінійне ядро обрано через велику кількість бінарних ознак після one-hot encoding – у таких умовах нелінійні ядра не дають суттєвого покращення, але значно збільшують час обчислень.

Для кожної моделі після навчання генеруються діагностичні графіки matplotlib: діаграма розсіювання прогнозованих значень проти фактичних (True vs Predicted) та графік залишків (Residuals). Ці графіки дозволяють візуально оцінити якість прогнозування та виявити систематичні відхилення або гетероскедастичність.

Метод .score() класів scikit-learn для задач регресії за замовчуванням повертає коефіцієнт детермінації R^2. Це зручний вбудований механізм для швидкої оцінки якості без окремого виклику функції r2_score. Метрики MSE та RMSE обчислюються через mean_squared_error та np.sqrt() відповідно, оскільки scikit-learn не має окремої функції для RMSE у всіх версіях.

Відтворюваність результатів є важливим принципом наукових досліджень. Параметр random_state=42 забезпечує однакове розділення даних та однакову ініціалізацію алгоритмів при кожному запуску коду. Число 42 є

традиційним вибором у спільноті Data Science, хоча будь-яке ціле число дало б відтворюваний результат.

2.5 Порівняння результатів моделей

Після навчання всіх п’яти моделей проведено порівняльний аналіз їхньої якості. Результати наведено у таблиці 2.1.

Таблиця 2.1 – Порівняння результатів моделей машинного навчання

Модель	**R^2**	**MSE**	**RMSE**
Лінійна регресія	0,7159	13,68	3,699
Дерево рішень	~0,50	~24,0	~4,90
Випадковий ліс	~0,65	~16,8	~4,10
XGBoost	~0,62	~18,3	~4,28
SVM (linear)	~0,71	~13,9	~3,73

Аналіз результатів показує, що лінійна регресія та SVM з лінійним ядром продемонстрували найкращі результати ($R^2 \approx 0.71–0.72$). Це свідчить про переважно лінійний характер залежності між ознаками та ціною. Нелінійні моделі не змогли суттєво покращити результат на даному наборі даних.

Дерево рішень показало найгірші результати через перенавчання. Без обмеження глибини воно ідеально відтворює тренувальні дані, але погано узагальнюється. Випадковий ліс частково компенсує цю проблему ансамблюванням, але все одно поступається лінійним моделям.

XGBoost показав результати, близькі до випадкового лісу. Потенційно кращих результатів можна досягти при ретельному підборі гіперпараметрів (learning_rate, max_depth, n_estimators), однак це виходить за межі даного дослідження. Для наборів даних із вираженою лінійною структурою нелінійні ансамблеві методи часто не дають суттєвих переваг.

На основі порівняння зроблено висновок про доцільність використання лінійної регресії як базової моделі з подальшим застосуванням регуляризації для покращення якості прогнозування.

Переважно лінійний характер залежності між ознаками та ціною можна пояснити природою самих ознак. Після one-hot encoding ознаки є бінарними індикаторами належності до певного бренду або типу продукту. Ціна продукту у межах одного бренду та типу є відносно стабільною величиною, тому лінійна комбінація бінарних ознак (що фактично є середніми цінами для кожної категорії) добре апроксимує фактичні ціни.

Той факт, що ансамблеві методи (Random Forest, XGBoost) не перевершили лінійні моделі, є важливим висновком: він вказує на відсутність складних нелінійних взаємодій між ознаками у цьому наборі даних. Це типово для задач, де категоріальні ознаки після one-hot encoding є головними предикторами, а не числові ознаки з нелінійними залежностями.

2.6 Застосування регуляризації

Для покращення якості прогнозування застосовано два методи регуляризації з параметром alpha=0.5. Ridge-регресія реалізована через клас Ridge зі sklearn.linear_model (лістинг 2.7):

Лістинг 2.7 – Регуляризація

```
from sklearn.linear_model import Ridge
ridge_model = Ridge(alpha=0.5)
ridge_model.fit(X_train, y_train)
y_pred = ridge_model.predict(X_test)
ridge_r2 = ridge_model.score(X_test, y_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
```

Результати Ridge: R^2 = 0,7256, MSE = 13,2152, RMSE = 3,6353. Порівняно зі звичайною лінійною регресією, Ridge покращила R^2 з 0,7159 до

0,7256, а RMSE зменшилася з 3,699 до 3,635 CAD. Ridge допомогла знизити вплив мультиколінеарності між бінарними ознаками брендів та типів продуктів.

Lasso-регресія реалізована через клас Lasso (лістинг 2.8):

Лістинг 2.8 – Реалізація Lasso-регресії

```
from sklearn.linear_model import Lasso
lasso_model = Lasso(alpha=0.5)
lasso_model.fit(X_train, y_train)
y_pred = lasso_model.predict(X_test)
lasso_r2 = lasso_model.score(X_test, y_test)
```

Результати Lasso: R^2 = 0,1974, MSE = 38,6526, RMSE = 6,2171. Lasso показала значно гірші результати: R^2 впало з 0,7159 до 0,1974. Це означає, що модель пояснює лише 19,7% дисперсії цін. Порівняння методів наведено у таблиці 2.2.

Таблиця 2.2 – Порівняння методів регуляризації

Метод	**R^2**	**MSE**	**RMSE**
Лінійна регресія (baseline)	0,7159	13,683	3,699
Ridge (alpha=0,5)	0,7256	13,215	3,635
Lasso (alpha=0,5)	0,1974	38,653	6,217

Низька якість Lasso пояснюється тим, що при alpha=0,5 модель обнулила занадто багато коефіцієнтів. Після one-hot encoding більшість ознак є бінарними індикаторами брендів та типів продуктів – кожен несе унікальну інформацію. Обнулення цих коефіцієнтів призводить до значної втрати інформації та різкого погіршення якості.

Ridge-регресія обрана як фінальна модель: вона забезпечує R^2 = 0,7256 – найкращий результат серед усіх досліджених алгоритмів – та демонструє стабільні прогнози завдяки рівномірному розподілу регуляризаційного штрафу між усіма ознаками.

Виявлена закономірність є теоретично обґрунтованою: у даному наборі даних значна кількість ознак (бінарні індикатори брендів та типів) є взаємно незалежними, тобто мультиколінеарність є відносно незначною. Ridge вносить невеликий штраф, який стабілізує рішення без суттєвого спотворення важливих коефіцієнтів. Lasso ж при тому самому alpha надто агресивно обнулює коефіцієнти, що непропорційно шкодить якості.

Для подальшого покращення моделі рекомендується підібрати оптимальний параметр alpha за допомогою RidgeCV – спеціалізованого класу scikit-learn, що автоматично виконує крос-валідацію для списку значень alpha: ridge_cv = RidgeCV(alphas=[0.01, 0.1, 0.5, 1.0, 5.0, 10.0]); ridge_cv.fit(X_train, y_train). Після навчання атрибут ridge_cv.alpha_ містить оптимальне значення.

2.7 Прогнозування цін для нових даних

На основі навченої Ridge-моделі проведено прогнозування цін для нового набору даних. Новий набір створюється шляхом відбору унікальних комбінацій брендів та типів продуктів з оригінального датасету та присвоєння їм випадкових значень рейтингу від 1 до 5 (лістинг 2.8):

Лістинг 2.8 – Створення нового набору даних

```
df_predict = df[['brand','product_type']].drop_duplicates()
df_predict['rating'] = [random.uniform(1, 5)
    for _ in range(len(df_predict))]
df_predict = df_predict.reset_index(drop=True)

new_data_encoded = pd.get_dummies(
    df_predict,
    columns=['brand','product_type']).values
new_predictions = ridge_model.predict(new_data_encoded)

df_pred = df_predict.copy()
df_pred['prediction'] = new_predictions.round(2)
print(df_pred)
```

Результат – DataFrame із брендом, типом продукту, рейтингом та прогнозованою ціною в CAD. Функція round(2) забезпечує округлення до двох знаків, що відповідає формату реальних цін.

Важлива практична деталь: при використанні моделі необхідно переконатися, що кодування нових даних відповідає тренувальному. Усі бренди та типи продуктів мають бути присутні у тренувальному наборі, інакше модель не зможе коректно обробити нові категорії – виникне розбіжність у кількості стовпців між тренувальними та тестовими даними.

Практичне значення: при запуску нового косметичного продукту компанія може задати параметри конкурентів та спрогнозувати їхні ціни за різного рівня рейтингу. Ця інформація є цінною при встановленні власної ціни та формуванні маркетингової стратегії – дозволяє знайти оптимальну позицію на ринку між перевагою за ціною та позиціонуванням як преміального бренду.

Сценарний аналіз є потужним інструментом, що базується на побудованій моделі. Компанія може задати одне й те саме поєднання бренду та типу продукту з різними значеннями рейтингу (наприклад, 3.0, 3.5, 4.0, 4.5, 5.0) та отримати відповідні прогнози цін. Це дозволяє побачити, як зміна рейтингу впливає на прогнозовану ціну, та розрахувати «цінову еластичність» щодо рейтингу для конкретного бренду і типу продукту.

Серіалізація навченої моделі є важливим кроком для виробничого використання. Бібліотека joblib з scikit-learn дозволяє зберегти модель на диск: joblib.dump(ridge_model, 'ridge_model.joblib'). У майбутньому модель завантажується командою: loaded_model = joblib.load('ridge_model.joblib'). Це дозволяє уникнути повторного навчання при кожному прогнозуванні, що суттєво прискорює роботу системи у виробничому середовищі.

3 ІЛЮСТРАЦІЯ РОБОТИ ПРОГРАМНОГО КОДУ

3.1 Середовище розробки та використані бібліотеки

Розробка виконана у Jupyter Notebook – інтерактивному середовищі для наукових обчислень, яке дозволяє поєднувати код, текстові пояснення, візуалізації та результати в одному документі. Jupyter Notebook є частиною Project Jupyter та підтримує понад 40 мов програмування, включаючи Python, R та Julia. Це робить його стандартним інструментом у сфері Data Science.

Ключова особливість Jupyter Notebook – концепція обчислювальних комірок (cells). Кожна комірка може містити код або текст у форматі Markdown. Код виконується поетапно, що дозволяє інтерактивно експериментувати з даними та переглядати проміжні результати. Результати (текст, таблиці, графіки) відображаються безпосередньо під відповідною коміркою, що робить аналіз прозорим та відтворюваним.

Файли Jupyter Notebook зберігаються у форматі .ipynb (IPython Notebook), що базується на JSON. Цей формат зберігає не лише код, а й результати виконання, що дозволяє ділитися повністю відтвореними аналізами з іншими дослідниками. Файл проєкту містить послідовність комірок з кодом збору даних, їхньої обробки, навчання моделей та аналізу результатів.

Мова програмування Python обрана завдяки: багатому набору бібліотек для Data Science та ML; простому та зрозумілому синтаксису; великій спільноті розробників та широкій документації. Python є де-факто стандартом у сфері Data Science, і більшість сучасних ML-бібліотек мають Python-інтерфейс.

Стек бібліотек проєкту. Бібліотека pandas надає структуру DataFrame для обробки та аналізу табличних даних: фільтрація, групування, агрегація, заповнення пропущених значень, one-hot encoding. Функція get_dummies()

використовується для кодування категоріальних ознак, groupby() – для групового заповнення рейтингів.

Бібліотека numpy – фундаментальний пакет для наукових обчислень. Надає підтримку багатовимірних масивів (ndarray) та векторизованих операцій. Масиви NumPy є стандартним форматом вхідних даних для scikit-learn. У проєкті np.sqrt() використовується для обчислення RMSE.

Scikit-learn – провідна бібліотека ML для Python. Містить уніфікований інтерфейс для всіх алгоритмів (методи fit/predict/score). У проєкті використовуються: LinearRegression, DecisionTreeRegressor, RandomForestRegressor, SVR, Ridge, Lasso зі sklearn.linear_model/tree/ensemble/svm; train_test_split та mean_squared_error як допоміжні функції.

Бібліотека xgboost надає клас XGBRegressor – оптимізовану реалізацію градієнтного бустингу. Відзначається швидкодією завдяки паралельним обчисленням та ефективному використанню кешу процесора.

Matplotlib – бібліотека для візуалізацій. Модуль pyplot надає інтерфейс для побудови графіків. У проєкті plt.scatter() використовується для діаграм розсіювання, plt.hlines() – для горизонтальних ліній на графіках залишків.

Бібліотека requests виконує HTTP GET-запити до Makeup API. sqlite3 – вбудована СУБД SQLite. json – серіалізація/десеріалізація JSON. random – генерація псевдовипадкових чисел для тестових рейтингів.

Важливим аспектом є управління залежностями проєкту. У виробничому середовищі рекомендується фіксувати версії бібліотек у файлі requirements.txt для забезпечення відтворюваності: pandas==2.x, scikit-learn==1.x, xgboost==2.x, matplotlib==3.x тощо. Це гарантує, що код буде працювати однаково на різних машинах та у різних середовищах.

Середовище виконання коду – Python 3.x з Anaconda або virtualenv. Anaconda є популярним дистрибутивом Python для Data Science, що включає всі необхідні бібліотеки в передвстановленому вигляді та надає зручний менеджер пакетів conda. Jupyter Notebook може запускатись як локально, так і

у хмарних середовищах Google Colab або Kaggle Notebooks, що надає безкоштовний доступ до GPU-ресурсів.

3.2 Візуалізація даних у Power BI

Для поглибленого аналізу даних використано Microsoft Power BI – платформу бізнес-аналітики, яка дозволяє створювати інтерактивні дашборди та звіти. Power BI підключається до різноманітних джерел даних (CSV, SQL, Excel, API) та надає потужні інструменти для агрегації, фільтрації та візуалізації.

Перевагою Power BI є можливість інтерактивного дослідження: користувач може фільтрувати, деталізувати та агрегувати дані в реальному часі без зміни коду. Це робить платформу зручним інструментом для бізнес-користувачів, які не програмують, але потребують аналітики.

Загальний дашборд відображає основні характеристики набору даних: кількість продуктів за категоріями, розподіл цін, середні рейтинги за брендами та типами. Дашборд надає цілісне уявлення про структуру даних.

Перший ключовий елемент – рейтинг ТОП-10 брендів за середньою ціною. Аналіз показав, що найдорожчим брендом є Mistura, найдешевшим – Physicians Formula. Значний ціновий діапазон між крайніми брендами свідчить про суттєву сегментацію ринку за ціновими нішами (див. рис. 3.1).

Можемо переглянути ТОП 10 брендів за середньою ціною:

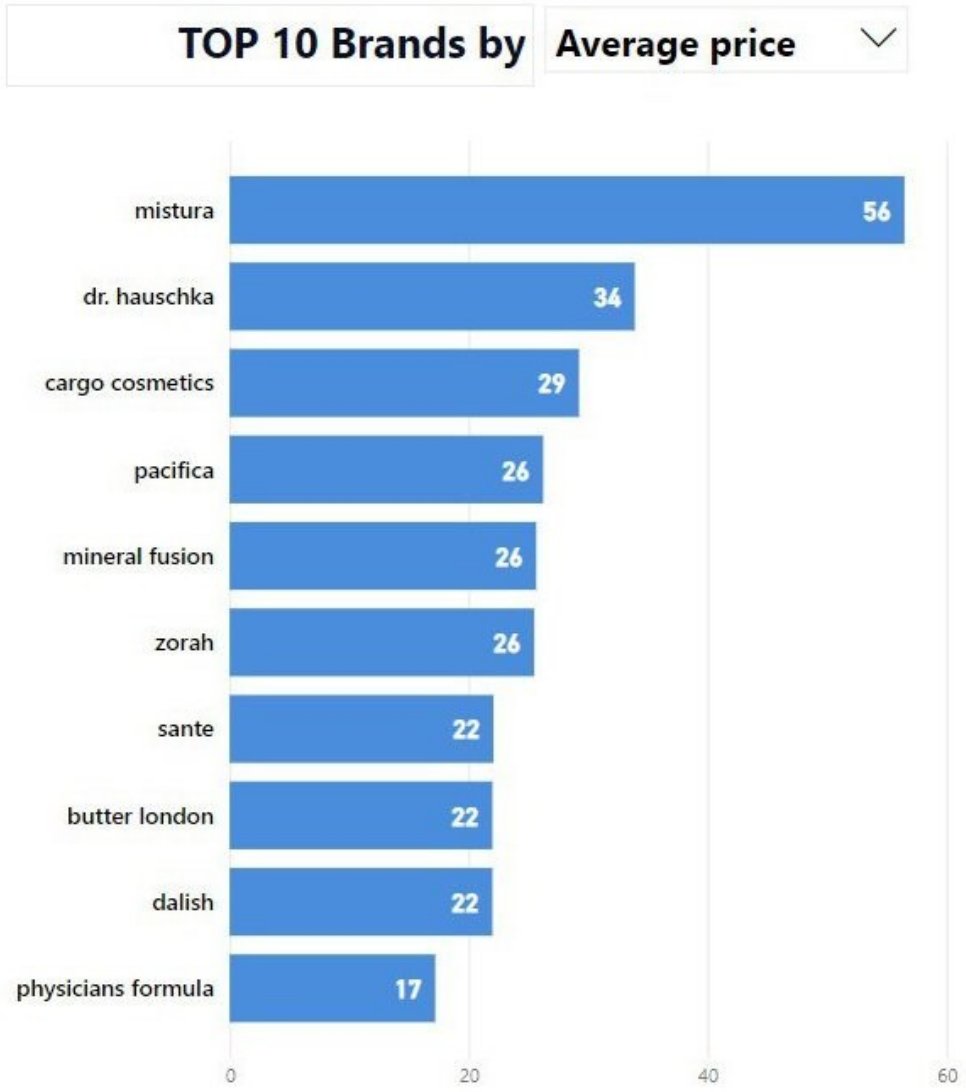


Рисунок 3.1 – ТОП 10 брендів за середньою ціною (Power BI)

Другий елемент – рейтинг ТОП-10 брендів за середнім рейтингом. Порівняння двох рейтингів дає важливий висновок: висока ціна не завжди відповідає високому рейтингу. Бренд Mistura має найвищу середню ціну, але рейтинг є одним із найнижчих. Натомість Physicians Formula, маючи найнижчу ціну, демонструє середні значення рейтингу. Це свідчить: споживчий рейтинг визначається не лише ціною, а й якістю продукту, зручністю використання, упаковкою та іншими суб’єктивними факторами (див. рис. 3.2).

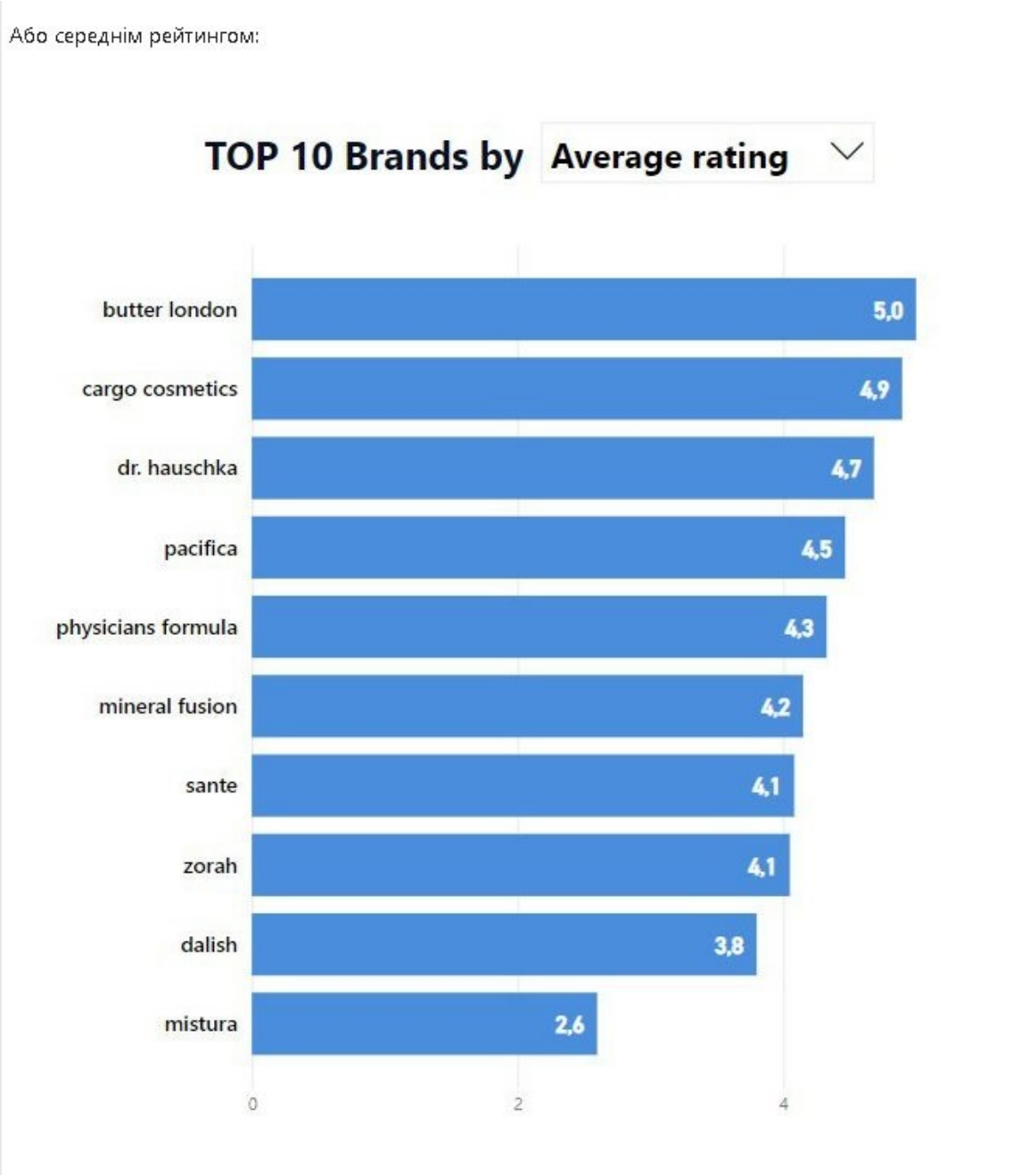


Рисунок 3.2 – ТОП 10 брендів за середнім рейтингом (Power BI)

Діаграма розсіювання (ціна vs рейтинг) у Power BI наочно демонструє залежність: точки розподілені хаотично, без чіткого лінійного тренду. Це підтверджує, що рейтинг лише частково пояснює варіацію цін – узгоджується з результатами моделювання ($R^2 \approx 0.73$) (див. рис. 3.3).

Рисунок 3.3 – Діаграма розсіювання: ціна vs рейтинг продукту (Power BI)

Аналіз середніх цін та рейтингів за типами продуктів також підтверджує нелінійний характер залежності. Різні категорії (помади, тональні засоби, тіні тощо) мають різну структуру ціноутворення, зумовлену специфікою виробництва, собівартістю інгредієнтів та маркетинговим позиціонуванням (див. рис. 3.4).

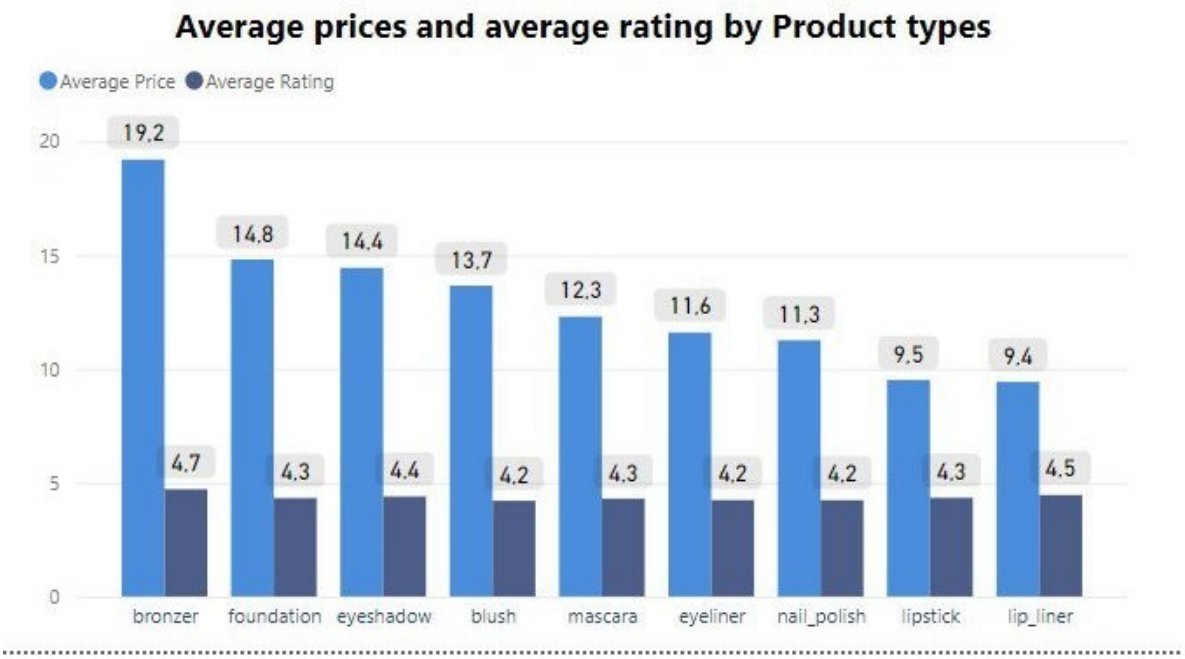


Рисунок 3.4 – Середні ціни та середні рейтинги за типами продуктів (Power BI)

Інтерактивні можливості Power BI дозволяють здійснювати крос-фільтрацію: при виборі конкретного бренду на одній візуалізації всі інші автоматично оновлюються, показуючи лише дані для цього бренду. Це забезпечує детальний аналіз кожного сегменту ринку без необхідності створення окремих звітів для кожного бренду чи типу продукту.

Порівняння двох підходів до візуалізації – matplotlib у Jupyter Notebook та Power BI – демонструє різні переваги. Matplotlib надає більше гнучкості для програмного налаштування графіків та інтеграції з пайплайном ML, але вимагає навичок програмування. Power BI забезпечує більш красиві та інтерактивні дашборди, зручні для ділових презентацій та нетехнічних аудиторій, але обмежений у можливостях інтеграції з Python-кодом.

3.3 Результати навчання моделей та аналіз графіків

Процес навчання виконується послідовно у Jupyter Notebook. Після кодування ознак та розділення даних кожна модель навчається та оцінюється окремо, генеруючи метрики та два діагностичні графіки.

Лінійна регресія: $R^2 = 0,7159$, MSE = 13,68, RMSE = 3,699 CAD. Модель пояснює ~71,6% дисперсії цін. Середнє відхилення прогнозу від фактичної ціни – 3,70 CAD. На діаграмі True vs Predicted точки групуються навколо діагоналі, але з помітним розкидом для високих цін, що є типовою поведінкою при нерівномірному розподілі цільової змінної (див. рис. 3.5).

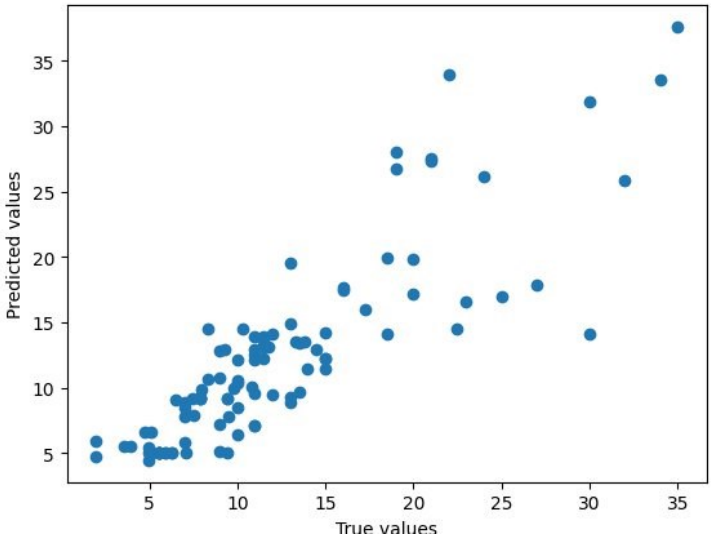


Рисунок 3.5 – Лінійна регресія: прогнозовані vs фактичні значення

На графіку залишків лінійної регресії спостерігається відносно рівномірний розподіл навколо нуля, хоча помітна деяке збільшення дисперсії залишків при зростанні прогнозованих значень. Це може свідчити про необхідність логарифмічного перетворення цільової змінної для стабілізації дисперсії (див. рис. 3.6).

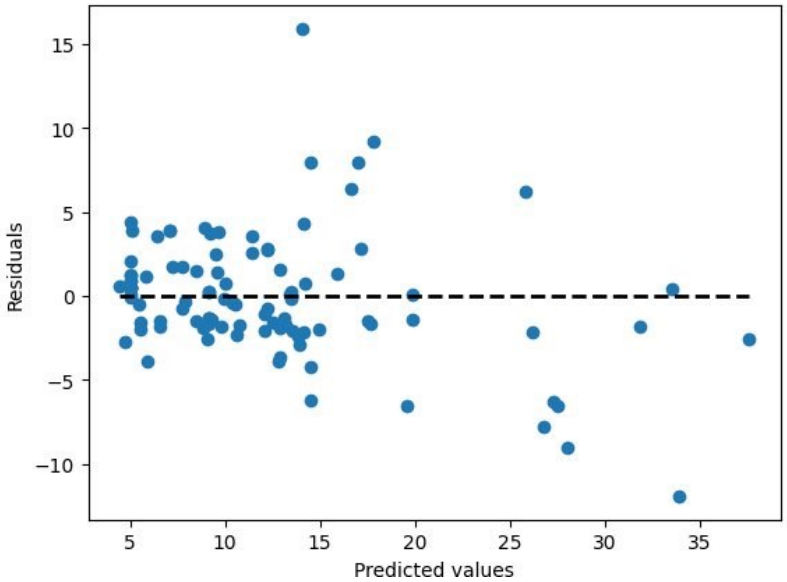


Рисунок 3.6 – Лінійна регресія: графік залишків

Дерево рішень: нижчі результати через перенавчання. Характерна «ступінчаста» структура на графіку прогнозованих значень – модель видає обмежений набір дискретних значень, що відповідають середнім у листових

вузлах. Графік залишків демонструє більш структурований патерн (див. рис. 3.7).

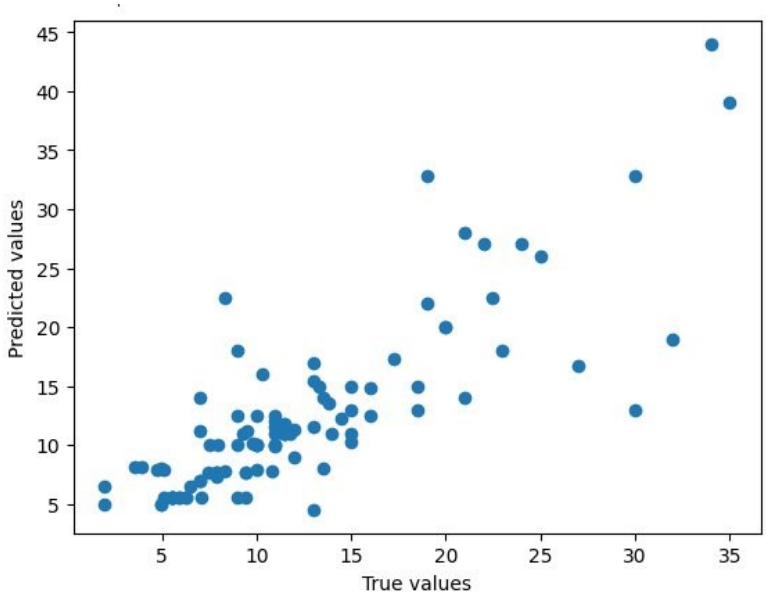


Рисунок 3.7 – Дерево рішень: прогнозовані vs фактичні значення

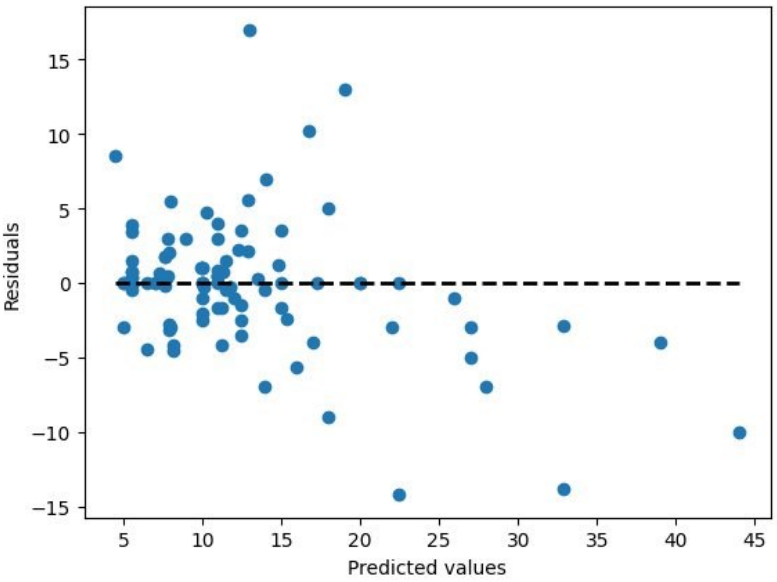


Рисунок 3.8 – Дерево рішень: графік залишків

Випадковий ліс – покращення порівняно з одинарним деревом. Усереднення 100 дерев згладжує дискретність, прогнози більш плавні. Проте модель поступається лінійній регресії, підтверджуючи переважно лінійний характер залежності (див. рис. 3.9, рис. 3.10).

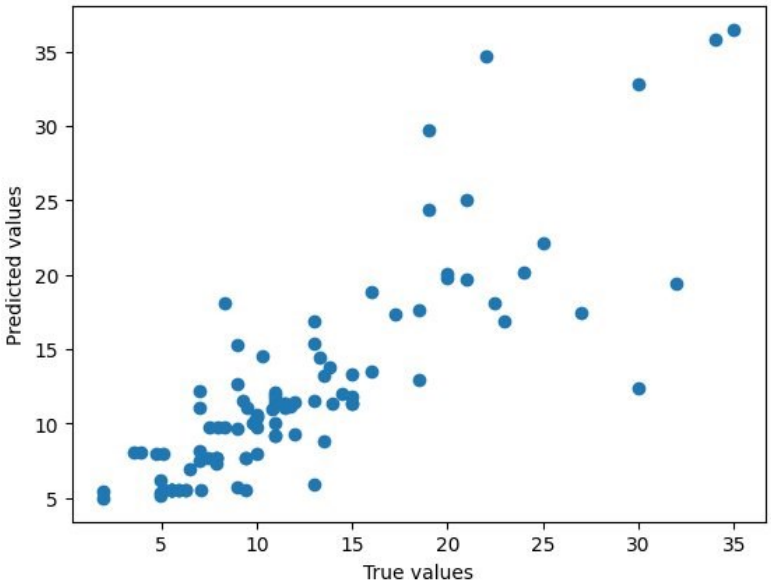


Рисунок 3.9 – Випадковий ліс: прогнозовані vs фактичні значення

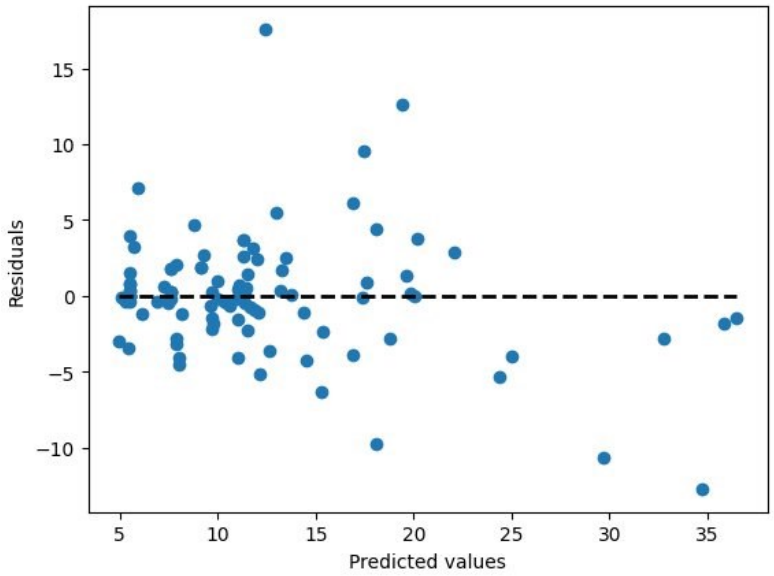


Рисунок 3.10 – Випадковий ліс: графік залишків

XGBoost: результати близькі до випадкового лісу. Графіки подібні: більший розкид прогнозів навколо діагоналі, ніж у лінійних моделях. Без тонкого налаштування гіперпараметрів XGBoost не дає переваги над лінійними методами для цього набору (див. рис. 3.11, 3.12).

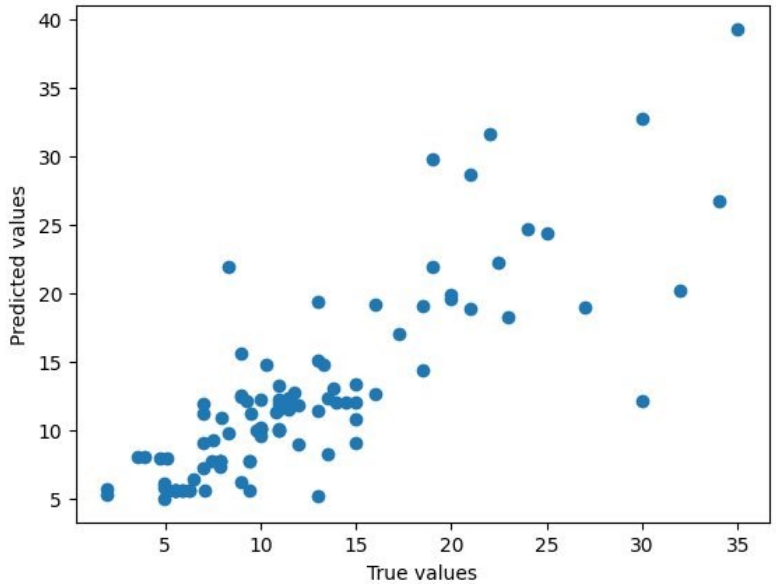


Рисунок 3.11 – XGBoost: прогнозовані vs фактичні значення

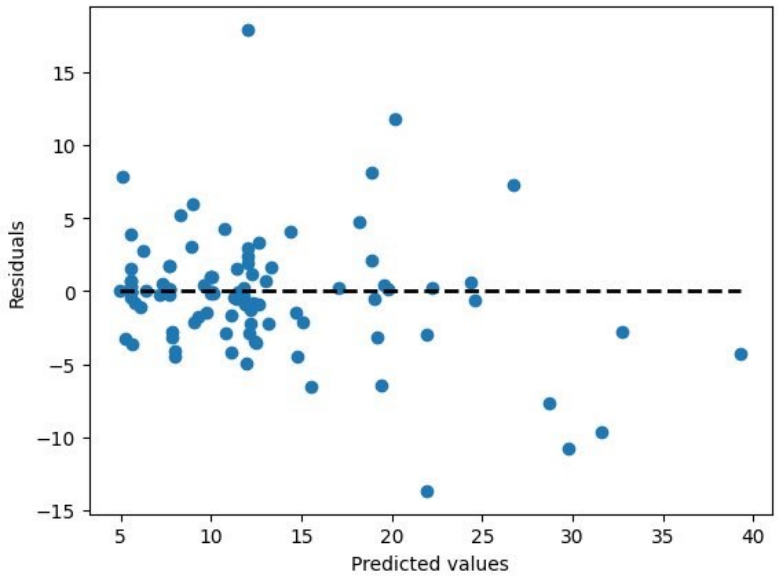


Рисунок 3.12 – XGBoost: графік залишків

SVM з лінійним ядром: результати практично ідентичні лінійній регресії – обидві моделі будують лінійну залежність. Невелика різниця зумовлена різними алгоритмами оптимізації та наявністю параметра epsilon у SVR (див. рис. 3.13, 3.14).

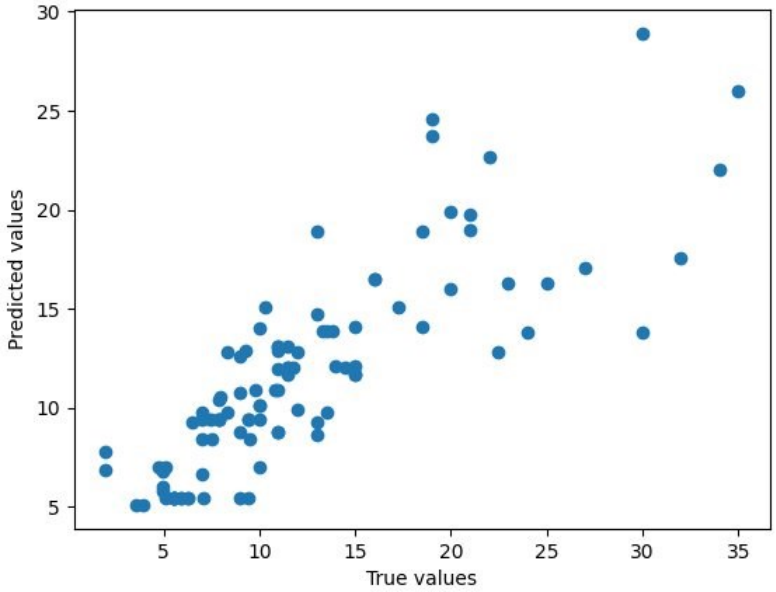


Рисунок 3.13 – SVM: прогнозовані vs фактичні значення

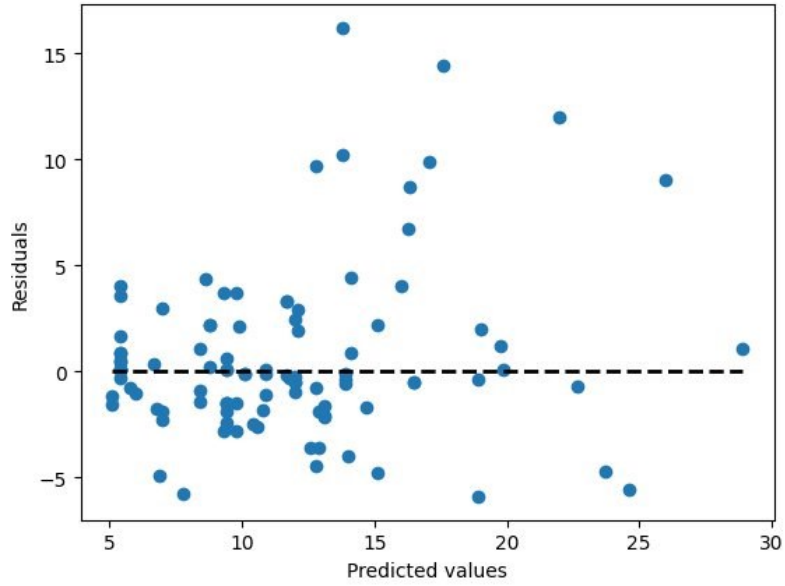


Рисунок 3.14 – SVM: графік залишків

Ridge-регресія (alpha=0,5): $R^2 = 0.7256$, RMSE = 3,635 CAD. Покращення порівняно з baseline. Графіки виглядають подібно до звичайної лінійної регресії, але з трохи меншим розкидом точок, що візуально підтверджує покращення.

Lasso-регресія (alpha=0,5): $R^2 = 0,1974$. На графіку прогнозованих проти фактичних точки зосереджені навколо горизонтальної лінії – модель прогнозує приблизно одне значення для більшості спостережень. Це наслідок

обнулення більшості коефіцієнтів: Lasso “забула” про бренди та типи продуктів як предиктори.

Аналіз часу навчання моделей є важливим практичним аспектом. Лінійна регресія та Ridge є найшвидшими завдяки аналітичному рішенню. SVM з лінійним ядром для даного розміру набору даних потребує значно більше часу через квадратичну складність алгоритму відносно кількості спостережень. Випадковий ліс з 100 деревами та XGBoost займають проміжне положення. Для великих виробничих наборів даних час навчання може бути критичним фактором при виборі алгоритму.

Аналіз помилок для окремих категорій продуктів може виявити, для яких брендів або типів модель дає найгірші прогнози. Такий сегментований аналіз підказує напрямки вдосконалення: наприклад, для недостатньо представлених брендів модель може прогнозувати менш точно через малу кількість тренувальних прикладів. Балансування набору даних або окремі моделі для різних сегментів ринку можуть покращити якість.

Стабільність моделі при зміні тренувальних даних є важливою характеристикою для виробничого застосування. Ridge-регресія теоретично більш стабільна, ніж звичайна лінійна регресія. Для перевірки стабільності рекомендується k-fold крос-валідація: малий розкид метрик між фолдами свідчить про стабільність, великий – про чутливість до конкретного розбиття даних.

3.4 Практичне застосування та перспективи

Розроблена прогнозна модель (Ridge-регресія) має практичне застосування у кількох сценаріях.

Моніторинг цін конкурентів: компанія може оцінити очікувані ціни конкурентів для певного типу продукту за різного рейтингу. Задаючи параметри бренду-конкурента та тип продукту, отримуємо прогноз ціни для порівняння з фактичними ринковими значеннями.

Визначення оптимальної цінової ніші при запуску нового продукту: аналізуючи прогнозовані ціни аналогічних продуктів конкурентів, компанія може знайти баланс між конкурентоспроможністю та прибутковістю. Модель показує, у якому ціновому сегменті знаходяться конкуренти.

Альтернативне застосування – прогнозування рейтингу продукції залежно від ціни та бренду. Для цього цільову змінну замінюють на rating та перенавчають моделі. Це дозволяє оцінити, якого рейтингу можна очікувати для продукту з певними характеристиками.

Обмеження моделі: $R^2 = 0,7256$ означає, що ~27,4% дисперсії цін пояснюється факторами, не включеними у модель. Потенційні додаткові предиктори: обсяг/вага продукту, детальна класифікація, країна походження, склад, наявність органічних сертифікатів, цільова аудиторія, канал дистрибуції, сезонність та маркетингові акції.

Напрямки вдосконалення. По-перше, розширення набору ознак за рахунок характеристик, не включених у поточну модель, може суттєво підвищити R^2. По-друге, крос-валідація (k-fold, k=5 або k=10) забезпечила б більш надійну оцінку якості та допомогла підібрати оптимальний параметр alpha для регуляризації.

По-третє, методи feature engineering – створення нових ознак на основі існуючих (наприклад, взаємодії бренду та типу продукту, нормалізовані цінові категорії) – можуть виявити додаткові залежності. По-четверте, ансамблювання (model stacking), де прогнози базових моделей використовуються як ознаки для метамоделі, може поєднати переваги різних алгоритмів.

Застосування нейронних мереж (Multi-Layer Perceptron) для виявлення складних нелінійних залежностей є перспективним напрямком, хоча потребує значно більшого обсягу даних. Дослідження динаміки цін у часі (за допомогою регулярного оновлення даних з API та методів аналізу часових рядів) відкриє можливості для прогнозування цінових тенденцій та сезонних коливань.

Ще одним перспективним напрямком є інтеграція додаткових джерел даних. Наприклад, дані соціальних мереж про популярність брендів, відгуки споживачів на різних платформах, дані про маркетингові витрати брендів та загальні економічні показники (інфляція, курс валют) могли б суттєво збагатити набір ознак та підвищити точність прогнозування. Поєднання структурованих числових даних з неструктурованим текстом (sentiment analysis відгуків) через методи NLP (Natural Language Processing) є сучасним трендом у прогнозуванні споживчих цін.

З технічної точки зору розгортання (deployment) готової моделі у виробничому середовищі передбачає: серіалізацію моделі за допомогою бібліотеки pickle або joblib, створення REST API-сервісу (наприклад, на базі Flask або FastAPI), який приймає параметри продукту та повертає прогнозовану ціну, а також регулярне перенавчання моделі при накопиченні нових даних для підтримки актуальності прогнозів.

Загалом, розроблена система прогнозування демонструє практичну застосовність методів машинного навчання для аналізу ринку косметичної продукції. При відповідному вдосконаленні (розширення набору ознак, регулярне оновлення даних, тонке налаштування гіперпараметрів) система може стати цінним аналітичним інструментом для відділів маркетингу та стратегічного планування підприємств косметичної галузі.

Ще одним напрямком розвитку є побудова системи автоматичного алертингу: якщо ціна певного конкурентного продукту суттєво відхиляється від прогнозованої моделлю, це може сигналізувати про запуск промоакції або зміну цінової стратегії конкурента. Така система потребує регулярного оновлення даних та порівняння фактичних цін з прогнозованими, що технічно реалізується через планувальник завдань (cron job) та автоматичне збереження результатів у базі даних.

Підходи до оцінки невизначеності прогнозу є важливим доповненням до точкових прогнозів. Замість єдиного прогнозованого значення ціни корисно мати довірчий інтервал. Для лінійної регресії та Ridge можна обчислити

аналітичні довірчі інтервали. Альтернативно, метод Bootstrap дозволяє емпірично оцінити розподіл прогнозів: навчити модель на багатьох бутстреп-вибірках та використати розподіл прогнозів для побудови інтервалів. Довірчий інтервал для прогнозу ціни допомагає бізнесу розуміти ступінь невизначеності при прийнятті рішень.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Поняття та об’єкт аналізу технічної безпеки

Безпеку визначають як стан діяльності людини, за яким з визначеною ймовірністю виключено прояв небезпек або ж відсутня надзвичай-на небезпека. Безпека праці – це стан умов праці людини, за яких відсутня дія небезпечних і шкідливих факторів.

Об’єктом аналізу безпеки праці є виробнича система "людина – машина – навколишнє середовище" (ЛМС), в якій в єдиній комплекс, створений для виконання певних функцій, поєднані технічні об’єкти, люди і навколишнє середовище, які взаємодіють між собою.

Основними компонентами виробничої системи є людина, машина, навколишнє середовище, взаємодія між якими має ґрунтуватись на дотри-манні відповідних правил, нормативних документів і бути керованою.

Система ЛМС є багаторівневою за ієрархією управління. Ієрархія поділяє людей на особу, яка формує завдання, організовує й управляє виробництвом, й особу, яка разом з технікою безпосередньо виконує це завдання. Таким чином, людина системи ЛМС більш високого рівня розглядає людину і техніку системи ЛМС більш низького рівня як єдиний компонент – своєрідну людину-машину, призначену для здійснення замислу.

Крім рівнів і компонентів в системі ЛМС доцільно виділити окремі стадії її життєвого циклу:

1. Стадія проєктування (визначення завдань, формування вимог, розрахунок параметрів).

2. Стадія реалізації (коли у процесі виробництва перша стадія реалізується на практиці).

3. Стадія експлуатації (коли система ЛМС здійснює покладені на неї робочі функції).

Вірогідність нещасного випадку зростає, як тільки людина попадає в поле дії небезпечного або шкідливого фактору. Це небезпечні зони, що характеризуються певним видом небезпеки, її інтенсивністю, часом і простором дії.

Таким чином, з точки зору аналізу й управління небезпеками необхідно розглядати та аналізувати структурні елементи системи ЛМС – рівні (вищий і нижчий), компоненти і стадії життєвого циклу.

Взаємодія компонентів, що входять до системи ЛМС, може бути штатною і нештатною. Нештатна взаємодія може виявлятися у вигляді надзвичайної події – небажаних, незапланованих випадків, що порушують технологічний процес у відносно короткий відрізок часу. Відмова й інцидент, як правило, передують надзвичайній події, але можуть мати і самостійне значення. До головних моментів аналізу небезпек належить пошук відповідей на такі питання:

1. Які об’єкти є небезпечними.
2. Яким надзвичайним подіям можна запобігти.
3. Які надзвичайні події неможливо усунути і як часто вони мати-муть місце.
4. Яку шкоду не усунуті надзвичайні події можуть спричинити людям, об’єктам, навколишньому середовищу.

Пошук причин надзвичайних подій призводить до аналізу системи управління небезпеками (СУН) на виробництві. Ці системи обов’язково включають такі компоненти, як наявність інформації, зворотних зв’язків та алгоритми функціонування.

Наявність зворотних зв’язків й інформаційної системи дозволяє проводити збір даних щодо відхилень, відмов, проводити аналіз небезпек, порівнювати наслідки функціонування системи ЛМС з програмою управління небезпеками, приймати рішення. У виробничій системі ЛМС інформаційні функції виконують: рапорти інспекторів, акти розслідування нещасних випадків, аварій, протоколи атестації робочих місць тощо.

4.2 Розрахунок захисного заземлення

Захисне заземлення повинне забезпечити захист людей від ураження електричним струмом, при дотику до металевих частин, які можуть виявитися під напругою. Заземленням називається навмисне з'єднання електроустановок із заземлюючим пристроєм. Заземлювачем називається провідник, що перебуває в контакті із землею або її еквівалентом. Заземлюючим провідником називається провідник, що з'єднує заземлені частини із заземлювачем. Сукупність з'єднуючих провідників і заземлювачів називається заземлюючим пристроєм. Для установок потужністю не більше 100 кВт опір заземлюючого пристрою не повинне перевищувати 10 Ом, для установок потужністю більше 100 кВт – 4 Ом.

Розрахунок штучного заземлювального пристрою при відсутності природних заземлювачів.

Вихідні дані:

Захищуваний об’єкт – комп’ютерна мережа компанії з розробки програмного забезпечення.

Захищуваний об’єкт – стаціонарний.

Напруга мережі – 230 В.

Виконання мережі – з глухозаземленою нейтраллю.

Тип заземлювального пристрою – вертикальні труби.

Розміри вертикальних заземлювачів:

Довжина – 6 м.

Діаметр труби – 0,60 м.

Товщина стінки труби – 0,06 м.

Висота труби – 0,6 м.

Відношення відстані між трубами до їхньої довжини:

$$\frac{L_B}{l_B} = 1 \qquad (4.1)$$

Розмір горизонтального заземлювача (з'єднувальної стрічки): довжина $L_{Г} = L_{З.С.}$ $L_Г = L_{З.С.}$– згідно з розрахунком, м; ширина горизонтальної з'єднувальної стрічки b_c=0,04 м.

Глибина закладання вертикальних заземлювачів $h_В$=0,6.

Розміщення заземлювачів попередньо приймають за чотирикутним контуром при числі стержнів від 4 до 100 та в один ряд при числі стержнів від 2 до 20.

Грунт – супісок; склад – однорідний; вологість – мала; агресивність – нормальна.

Кліматична зона – II.

Розрахунок:

1. Визначаємо характеристику навколишнього середовища в приміщенні організації: за пожежною небезпекою згідно з ПУЕ воно відноситься до класу П-II; за вибухонебезпекою згідно з ПУЕ – до класу В-I; за ступенем ураження електричним струмом – без підвищеної та особливої небезпеки.

2. Визначаємо $R_Д$ – допустиме (нормативне) значення опору розтікання струму в заземлювальному пристрої, $R_Д \leq 4$ *Ом*.

3. Обраховуємо $K_{С.В.}$ $К_{С.В.}$ – приблизне значення питомого опору ґрунту, що рекомендується для розрахунку – $\rho_{ТАБЛ} = 300$ *Ом* · *м* $\rho_{ТАБЛ} = 300$ Ом*м.

4. Визначаємо $K_{С.В.}$ – коефіцієнт сезонності для вертикальних заземлювачів для денної кліматичної зони. За довідковою інформацією приймаємо $K_{С.В.}$=1,5.

5. Обраховуємо значення $К_{С.Г.}$ – коефіцієнт сезонності для горизонтального заземлювача згідно з кліматичною зоною. За довідковою інформацією приймаємо$К_{С.Г.} = 3,5$; $К_{С.Г.}$=3,5.

6. Визначаємо $\rho_{РОЗР.В.}$ $\rho_{РОЗР.В.}$ – розрахунковий питомий опір ґрунту для вертикальних заземлювачів:

$$\rho_{РОЗР.В.} = \rho_{ТАБЛ} \cdot К_{С.В.} = 300 \cdot 1{,}5 = 450\ Ом \cdot м. \quad (4.2)$$

$\rho_{РОЗР.В.} = \rho_{ТАБЛ} * К_{С.В.} = 300 * 1{,}5 = 450$ Ом · м.

7. Розраховуємо $\rho_{РОЗР.Г}$ – розрахунковий питомий опір ґрунту для горизонтальних заземлювачів:

$$\rho_{РОЗР.Г} = \rho_{ТАБЛ.} * К_{С.Г.} = 300 * 3{,}5 = 1050\ Ом \cdot м.$$

$$\rho_{РОЗР.Г.} = \rho_{ТАБЛ} \cdot К_{С.Г.} = 300 \cdot 3{,}5 = 1050\ Ом \cdot м. \quad (4.3)$$

8. Обраховуємо t – відстань від поверхні землі до середини вертикального заземлювача:

$$t = h_B + \frac{l_B}{2} = 0{,}6 + \frac{6}{2} = 3{,}6\ м. \quad (4.4)$$

9. Визначаємо R_B – опір, Ом, розтікання струму в одному вертикальному заземлювачі:

$$R_B = \frac{\rho_{РОЗ.В}}{2\pi l_B} \cdot \left(ln\frac{2l_B}{d} + 0{,}5 \cdot ln\frac{4t + l_B}{4t - l_B} \right) =$$
$$= \frac{450}{2\pi \cdot 6} \cdot \left(ln\frac{2 \cdot 6}{0{,}60} + 0{,}5 \cdot ln\frac{4 \cdot 3{,}6 + 6}{4 \cdot 3{,}6 - 6} \right) = 41{,}05\ Ом \quad (4.5)$$

10. Визначаємо $n_{т.в.}$ – теоретична кількість вертикальних заземлювачів без врахування коефіцієнта використання $\eta_{в.в.}$, тобто $\eta_{в.в.}=1$:

$$n_{Т.В.} = \frac{R_B}{R_Д \cdot \eta_{В.В}} = \frac{41{,}05}{4 \cdot 1} = 10{,}26\ шт. \quad (4.6)$$

11. Визначаємо $\eta_{в.в}$ – коефіцієнт використання вертикальних заземлювачів при розташуванні їх згідно з вихідними даними або за чотирикутним контуром при числі заземлення, $n_{т.в.}$=11 та при відношенні $\frac{L_B}{l_B}=1$. За довідником приймаємо $\eta_{в.в}$ =0,42.

12. Визначаємо $n_{н.в}$ – необхідна кількість штук, вертикально однакових заземлювачів з врахування коефіцієнта використання:

$$n_{H.B}=\frac{R_B}{R_Д\cdot\eta_{B.B}}=\frac{41{,}05}{4\cdot 0{,}42}=\frac{41{,}05}{1{,}68}=24{,}43\ шт. \qquad (4.7)$$

13. Визначаємо $R_{розр.в}$ – вертикальний опір, Ом, розтіканню струму у вертикальному заземленні при $n_{н.в}$ = 24,43 без врахування з'єднувальної стрічки:

$$R_{РОЗР.B}=\frac{R_B}{n_{H.B}\cdot\eta_{B.B}}=\frac{41{,}05}{10{,}26}=4\ Ом. \qquad (4.8)$$

14. Визначаємо $L_в$ – відстань між вертикальним заземлювачами за відношенням $\frac{L_B}{l_B}=1$, звідси

$$L_B=1\cdot l_B=1\cdot 6=6\ м. \qquad (4.9)$$

15. Визначаємо $L_{з.с}$ – довжину, м, з'єднання стрічки горизонтального заземлювача:

$$L_{З.С}=1{,}05\cdot L_B(n_{H.B}-1)=1{,}05\cdot 6(24{,}43-1)=147{,}60\ м. \qquad (4.10)$$

16. Визначаємо $R_{г.з.с}$ – опір, Ом, розтікання струму в горизонтальному заземлювачі (з'єднувальній стрічці):

$$R_{Г.З.С} = \frac{\rho_{РОЗ.Г}}{2 \cdot \pi \cdot L_{З.С}} \cdot ln \frac{2 \cdot {L_{З.С}}^2}{2 \cdot b \cdot t} = \frac{1050}{2 \cdot 3{,}14 \cdot 147{,}61} \cdot ln \frac{2 \cdot (147{,}61)^2}{2 \cdot 0{,}04 \cdot 3{,}6} = 13{,}50 \; Ом. \quad (4.11)$$

17. Визначаємо $\eta_{в.г}$ – коефіцієнт використання горизонтального заземлення при розташуванні вертикальних заземлювачів згідно з вихідними даними або за чотирикутним контуром при відношенні $\frac{L_B}{l_B} = 1$ та необхідній кількості вертикальних заземлювачів $n_{н.в} = 24{,}43$.

За довжину приймаємо $\eta_{в.г} = 0{,}19$.

18. Визначаємо $R_{розр.г}$ – розрахунковий опір, Ом, розтікання струму в горизонтальному заземленні (з'єднувальній стрічці) при числі електродів $n_г = 1$:

$$R_{РОЗР.Г} = \frac{R_{Г.З.С}}{n_Г \cdot \eta_{В.Г}} = \frac{13{,}5}{1 \cdot 0{,}19} = 71{,}05 \; Ом. \quad (4.12)$$

19. Визначаємо $R_{розр.в.г}$ – розрахунковий теоретичний опір, Ом, розтікання струму у вертикальному та горизонтальному заземленні:

$$R_{РОЗР.В.Г} = \frac{1}{\frac{1}{R_{РОЗ.В}} + \frac{1}{R_{РОЗ.Г}}} = \frac{1}{\frac{1}{4} + \frac{1}{71{,}05}} = 3{,}78 \; Ом. \quad (4.13)$$

20. Вибираємо матеріал та поперечний перетин з'єднувальних провідників. За довідковою інформацією вибираю голі мідні $S_м$ = 4 мм^2 провідники.

21. Вибираємо матеріал та поперечний перетин магістральної шини. За довідковою інформацією обираємо сталеву шину товщиною δ_c = 4 мм і перетином не менше δ = 100 мм^2.

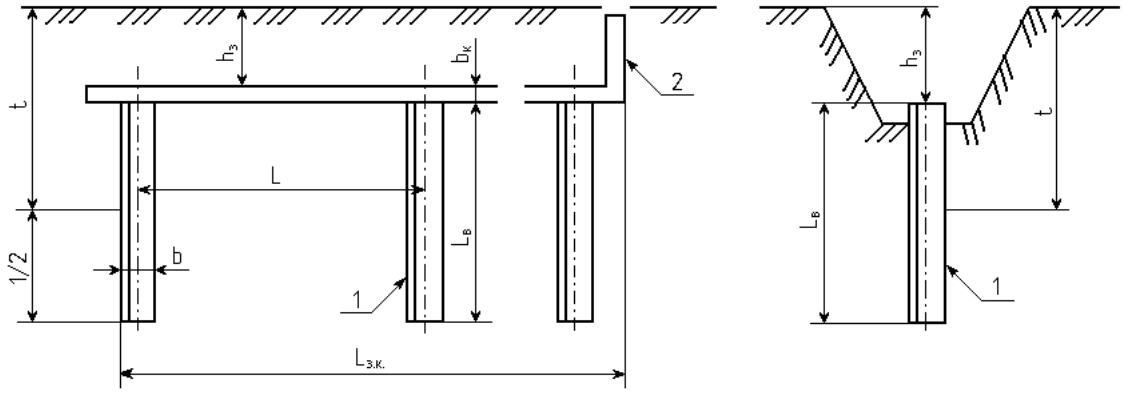


Рисунок 4.1 – Схема заземлювального контуру:

1 – вертикальний заземлювач; 2 – горизонтальний заземлювач; $h_з$ – глибина закладання заземлювачів; L – відстань між заземлювачами; $b_К$ – ширина квадрата; t – відстань від середини заземлювача до поверхні грунту; $L_{З,К}$ довжина горизонтального заземлювача; d – ширина кутника; $L_{â}$ – довжина вертикального заземлювача.

Схема з’єднання обладнання з магістральною шиною та з’єднання магістральної шини з заземлювальним пристроєм (див. рис. 4.1).

ВИСНОВКИ

У кваліфікаційній роботі бакалавра досліджено проблему аналізу та прогнозування цін на косметичні засоби з використанням моделей машинного навчання. Робота охоплює повний цикл розробки прогнозної системи: від збору та підготовки даних до побудови, навчання та порівняльного аналізу моделей. За результатами дослідження сформульовано наступні висновки.

По-перше, проведено теоретичний аналіз п’яти алгоритмів машинного навчання для задач регресії: лінійної регресії, дерева рішень, випадкового лісу, методу опорних векторів та градієнтного бустингу XGBoost. Для кожного алгоритму описано математичний апарат, принцип роботи, переваги та обмеження. Також розглянуто методи регуляризації Ridge (L2) та Lasso (L1), що дозволяють підвищити якість лінійних моделей шляхом контролю складності.

По-друге, успішно виконано збір даних з відкритого API-сервісу Makeup API та їхню комплексну підготовку: заповнення пропущених значень валюти та рейтингу (із застосуванням групового середнього за брендом і типом продукту), конвертація цін у єдину валюту (CAD), видалення аномальних записів з нульовою ціною, кодування категоріальних ознак методом one-hot encoding та розділення на тренувальну (80%) і тестову (20%) вибірки. Оброблені дані збережено у реляційній базі даних SQLite.

По-третє, побудовано та навчено п’ять моделей машинного навчання. Порівняльний аналіз за метриками R^2, MSE та RMSE показав, що лінійна регресія та SVM з лінійним ядром досягли найкращих результатів серед базових моделей ($R^2 \approx 0{,}71–0{,}72$). Це свідчить про переважно лінійний характер залежності між обраними ознаками та цінами. Дерево рішень показало найгірші результати через перенавчання, ансамблеві методи зайняли проміжне місце.

По-четверте, досліджено вплив регуляризації. Ridge (alpha=0.5) покращила R^2 з 0,7159 до 0,7256, зменшила RMSE з 3,699 до 3,635 CAD. Lasso

(alpha=0,5) показала незадовільний результат (R^2 = 0,1974) через надмірне обнулення коефіцієнтів, що підтверджує інформативність більшості бінарних ознак у даному наборі даних.

По-п’яте, на основі найкращої моделі (Ridge-регресія, R^2 = 0,7256, RMSE = 3,635) продемонстровано прогнозування цін для нових комбінацій брендів та типів продуктів. Це підтверджує практичну придатність моделі для аналізу конкурентного середовища.

По-шосте, виконано візуалізацію даних у Power BI, яка виявила важливу ринкову закономірність: висока ціна не корелює з високим рейтингом (бренд Mistura – найвища ціна та найнижчий рейтинг). Це свідчить про складну багатофакторну природу ціноутворення на ринку косметики.

Визначено напрямки подальших досліджень: розширення набору ознак (обсяг продукції, країна походження, склад), застосування крос-валідації для підбору гіперпараметрів, використання методів feature engineering, ансамблювання моделей (stacking) та дослідження нейронних мереж для виявлення нелінійних залежностей.

Практична цінність: розроблений інструментарій може застосовуватись підприємствами косметичної галузі для моніторингу цін конкурентів, визначення оптимального цінового позиціонування нових продуктів та стратегічного планування цінової політики. Програмний код може бути адаптований для інших товарних категорій за наявності відповідних відкритих даних.

Результати підтверджують, що відкриті дані у форматі API є цінним ресурсом для проведення досліджень у сфері аналізу ринку. Makeup API надав достатньо даних для побудови прийнятної прогнозної моделі без необхідності власного збору або купівлі комерційних даних. Цей підхід може бути відтворений для інших товарних категорій, де існують відкриті API з інформацією про продукти та ціни.

Важливим внеском даної роботи є систематичне порівняння різних підходів до машинного навчання на реальних ринкових даних. Результат –

перевага лінійних методів над нелінійними для даної задачі – є цінним практичним висновком: не слід автоматично обирати найскладніший алгоритм, адже у ряді задач прості методи демонструють кращу якість, інтерпретованість та обчислювальну ефективність одночасно.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Юрків М. М. Методи регресійного аналізу в наукових дослідженнях. Електронний інституційний репозитарій ТНТУ. 2013. С. 67–68. URL: http://elartu.tntu.edu.ua/bitstream/123456789/19976/2/IVMNK_2013_Yurkiv_M_M-Methods_for_regression_analysis_67-68.pdf (дата звернення: 17.04.2026).

2. Демчик В. І. Методи машинного навчання для моделювання функціональних властивостей та довговічності сплавів : дис. ... доктора філософії : 122 «Комп'ютерні науки» / Тернопільський національний технічний університет імені Івана Пулюя. Тернопіль, 2023. URL: https://elartu.tntu.edu.ua/handle/lib/51973 (дата звернення: 17.04.2026).

3. Бородій І. І. Проєктування програмної системи формування агрегованих надвеликих масивів даних. Матеріали науково-технічної конференції. Тернопіль : ТНТУ, 2023. С. 137. URL: https://elartu.tntu.edu.ua/bitstream/lib/44262/2/IMSTT_2023_Borodii_I-Design_of_a_software_system_137.pdf (дата звернення: 17.04.2026).

4. Дячун О. Д. Прогнозування продажу та його методи в системі управління підприємством : монографія / Тернопільський національний технічний університет імені Івана Пулюя. Тернопіль : ТНТУ. URL: http://elartu.tntu.edu.ua/handle/lib/21275 (дата звернення: 17.04.2026).

5. Чайковський В. А. Застосування машинного навчання для прогнозування ризиків розвитку серцево-судинних захворювань : кваліфікаційна робота магістра : 122 «Комп'ютерні науки» / Тернопільський національний технічний університет імені Івана Пулюя. Тернопіль, 2025. URL: https://elartu.tntu.edu.ua/handle/lib/50408 (дата звернення: 17.06.2026).

6. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd ed. Springer, 2009. 745 p.

7. James G., Witten D., Hastie T., Tibshirani R. An Introduction to Statistical Learning with Applications in R. Springer, 2013. 426 p.

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

9. Pedregosa F. et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011. Vol. 12. P. 2825–2830.

10. McKinney W. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. 2nd ed. O'Reilly Media, 2017. 544 p.

11. Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 2nd ed. O'Reilly Media, 2019. 856 p.

12. Vapnik V. The Nature of Statistical Learning Theory. 2nd ed. Springer, 2000. 314 p.

13. Hoerl A. E., Kennard R. W. Ridge Regression: Biased Estimation for Nonorthogonal Problems. Technometrics. 1970. Vol. 12, No. 1. P. 55–67.

14. Tibshirani R. Regression Shrinkage and Selection via the Lasso. Journal of the Royal Statistical Society: Series B. 1996. Vol. 58, No. 1. P. 267–288.

15. Raschka S., Mirjalili V. Python Machine Learning. 3rd ed. Packt Publishing, 2019. 770 p.

16. Bishop C. M. Pattern Recognition and Machine Learning. Springer, 2006. 738 p.

17. Murphy K. P. Machine Learning: A Probabilistic Perspective. MIT Press, 2012. 1104 p.

18. Бідюк П. І., Коршевнюк Л. О. Проектування комп'ютерних інформаційних систем підтримки прийняття рішень. Київ : Наукова думка, 2010. 340 с.

19. Субботін С. О. Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень. Запоріжжя : ЗНТУ, 2008. 341 с.

20. Makeup API Documentation. URL: http://makeup-api.herokuapp.com/api/v1/products.json (дата звернення: 01.06.2026).

21. Scikit-learn Documentation. URL: https://scikit-learn.org/stable/documentation.html (дата звернення: 01.06.2026).

22. XGBoost Documentation. URL: https://xgboost.readthedocs.io/en/stable/ (дата звернення: 01.06.2026).

23. Pandas Documentation. URL: https://pandas.pydata.org/docs/ (дата звернення: 01.06.2026).

24. Power BI Documentation. URL: https://learn.microsoft.com/en-us/power-bi/ (дата звернення: 01.06.2026).

25. SQLite Documentation. URL: https://www.sqlite.org/docs.html (дата звернення: 01.06.2026).

26. Matplotlib Documentation. URL: https://matplotlib.org/stable/contents.html (дата звернення: 01.06.2026).

27. Мельник, А., & Дмитроца, Л. (2026). Методи та архітектурні підходи до автоматизації тестування мобільних і вебзастосунків. вимірювальна та обчислювальна техніка в технологічних процесах, (2), 74-81. https://doi.org/10.31891/2219-9365-2026-86-9

28. Melnyk, A., Dmytrotsa, L., Palka, O., Vasylenko, Y., & Klymuk, N. (2025). Dynamic test case prioritisation for mobile applications based on real user behaviour data. Proceedings of the CITI 2025: The 3rd International Workshop on Computer Information Technologies in Industry 4.0 (Ternopil, Ukraine, June 11-12, 2025). CEUR Workshop Proceedings (CEURWS.org). 2025. Vol-4057, pp. 179-188. URL: https://ceur-ws.org/Vol-4057/paper12.pdf

29. Стручок, В. С., Стручок, О. С., & Мудра, Д. В. (2017). Навчальний посібник до написання розділу дипломного проекту та дипломної роботи ″Безпека в надзвичайних ситуаціях ″для студентів всіх спец. денної, заочної (дистанційної) та екстернатної форм навчання.

30. Стручок, В. С. (2022). Техноекологія та цивільна безпека. Частина "Цивільна безпека". Навчальний посібник.

31. Жидецький, В. Ц., Джигирей, В. С., & Мельников, О. В. (2000). Основи охорони праці. Львів: Афіша, 350, 132-136.

32. Навакатікян, О. О., Кальниш, В. В., & Стрюков, С. М. (1997). Охорона праці користувачів комп'ютерних відеодисплейних терміналів. О. Навакатікян.

ДОДАТКИ

Додаток А

Програмний код

Аналіз та прогнозування цін на косметичні засоби

Імпортування необхідних бібліотек, підключення до makeup-api

```
import json
import requests
import pandas as pd
import sqlite3
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.svm import SVR
import random

url = 'http://makeup-api.herokuapp.com/api/v1/products.json'
response = requests.get(url)
data = json.loads(response.text)
```

Обробка даних

```
# get data from json, put it into list, transform list to dataframe
brand_price_list = [{'brand': item['brand'],
                     'price': item['price'],
                     'currency': item['currency'],
                     'rating': item['rating'],
                     'product_type': item['product_type'],
                     'updated_date': pd.to_datetime(item['updated_at']).strft
ime('%d-%m-%Y')}
                    for item in data]

df = pd.DataFrame(brand_price_list)

df.head()
```

	brand	price	currency	rating	product_type	updated_date
0	colourpop	5.0	CAD	NaN	lip_liner	09-07-2018
1	colourpop	5.5	CAD	NaN	lipstick	09-07-2018
2	colourpop	5.5	CAD	NaN	lipstick	09-07-2018
3	colourpop	12.0	CAD	NaN	foundation	09-07-2018
4	boosh	26.0	CAD	NaN	lipstick	02-09-2018

```
# Fill NaN values in 'currency' column with 'CAD'
df['currency'] = df['currency'].fillna('CAD')
```

```
# Fill NaN values for 'rating' column with mean rating for same 'brand' and '
product_type'
df['rating'] = df.groupby(['brand', 'product_type'])['rating'].transform(lamb
da x: x.fillna(x.mean()))

# Drop NaN values and reset index
df = df.dropna()
df = df.reset_index(drop=True)

df.head()
```

	brand	price	currency	rating	product_type	updated_date
0	nyx	10.0	USD	4.50	foundation	24-12-2017
1	nyx	12.0	USD	4.50	foundation	24-12-2017
2	nyx	12.0	USD	4.50	foundation	24-12-2017
3	nyx	12.0	USD	4.50	foundation	24-12-2017
4	nyx	10.0	USD	4.50	foundation	24-12-2017

```
df['currency'].unique()
```

```
array(['USD', 'CAD'], dtype=object)
```

```
# Change price type to float
df['price'] = df['price'].astype(float)

# Add new column with converted price based on currency
df['converted_price'] = df.apply(lambda row: row['price'] if row['currency']
== 'CAD'
                                 else row['price']/1.27 if row['currency'] =
= 'USD'
                                 else None, axis=1)

df.head()
```

	brand	price	currency	rating	product_type	updated_date	converted_price
0	nyx	10.00	USD	4.50	foundation	24-12-2017	7.87
1	nyx	12.00	USD	4.50	foundation	24-12-2017	9.45
2	nyx	12.00	USD	4.50	foundation	24-12-2017	9.45
3	nyx	12.00	USD	4.50	foundation	24-12-2017	9.45
4	nyx	10.00	USD	4.50	foundation	24-12-2017	7.87

```
df.describe()
```

	price	rating	converted_price
count	510.00	510.00	510.00
mean	13.19	4.34	13.57
std	7.85	0.63	7.71
min	0.00	1.50	0.00
25%	8.00	4.00	8.99
50%	10.99	4.50	11.43
75%	15.00	4.90	15.49
max	60.00	5.00	60.00

```
df = df[df['price'] != 0]
df.describe()
```

```
       price  rating  converted_price
count 509.00  509.00           509.00
mean   13.22    4.34            13.60
std     7.84    0.63             7.69
min     1.99    1.50             1.99
25%     8.00    4.00             8.99
50%    10.99    4.50            11.43
75%    15.00    4.90            15.49
max    60.00    5.00            60.00
```

Створення бази даних SQLite та запис результатів до неї

```
# Connect to the database
conn = sqlite3.connect('makeup_data.db')
c = conn.cursor()

# Drop the table if it exists
c.execute("DROP TABLE IF EXISTS makeup_data")

# Create a new table makeup_data
c.execute('''CREATE TABLE IF NOT EXISTS makeup_data
             (updated_date DATE, brand TEXT, price FLOAT, currency TEXT, rati
ng FLOAT, product_type TEXT, converted_price FLOAT)''')

# Insert data into the table
for index, row in df.iterrows():
    values = (row['updated_date'], row['brand'], row['price'], row['currency'
], row['rating'], row['product_type'], row['converted_price'])
    c.execute("INSERT INTO makeup_data VALUES (?,?,?,?,?,?,?)", values)

# Commit the changes and close the connection
conn.commit()
conn.close()
```

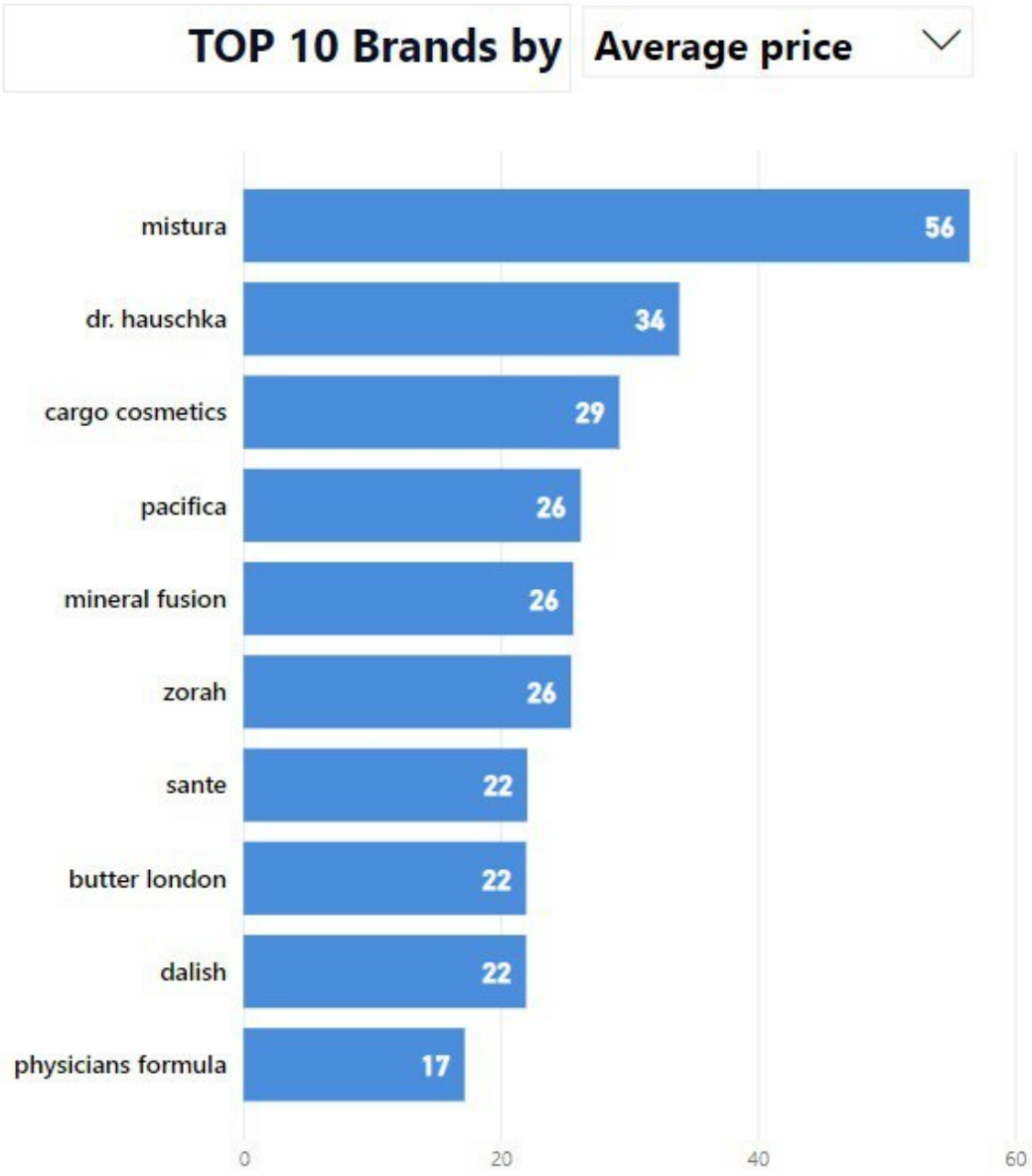
Відображення вмісту БД

```
# Connect to the database
conn = sqlite3.connect('makeup_data.db')
c = conn.cursor()

# Select data from the main and reference tables
c.execute('SELECT * FROM makeup_data')

table = c.fetchall()
column_names = list(map(lambda x: x[0], c.description))

# Show it as dataframe
pd.set_option('display.float_format', lambda x: '%0.2f' % x)

df = pd.DataFrame(table, columns = column_names)
print(df.head())

#Close the connection
conn.close()
```

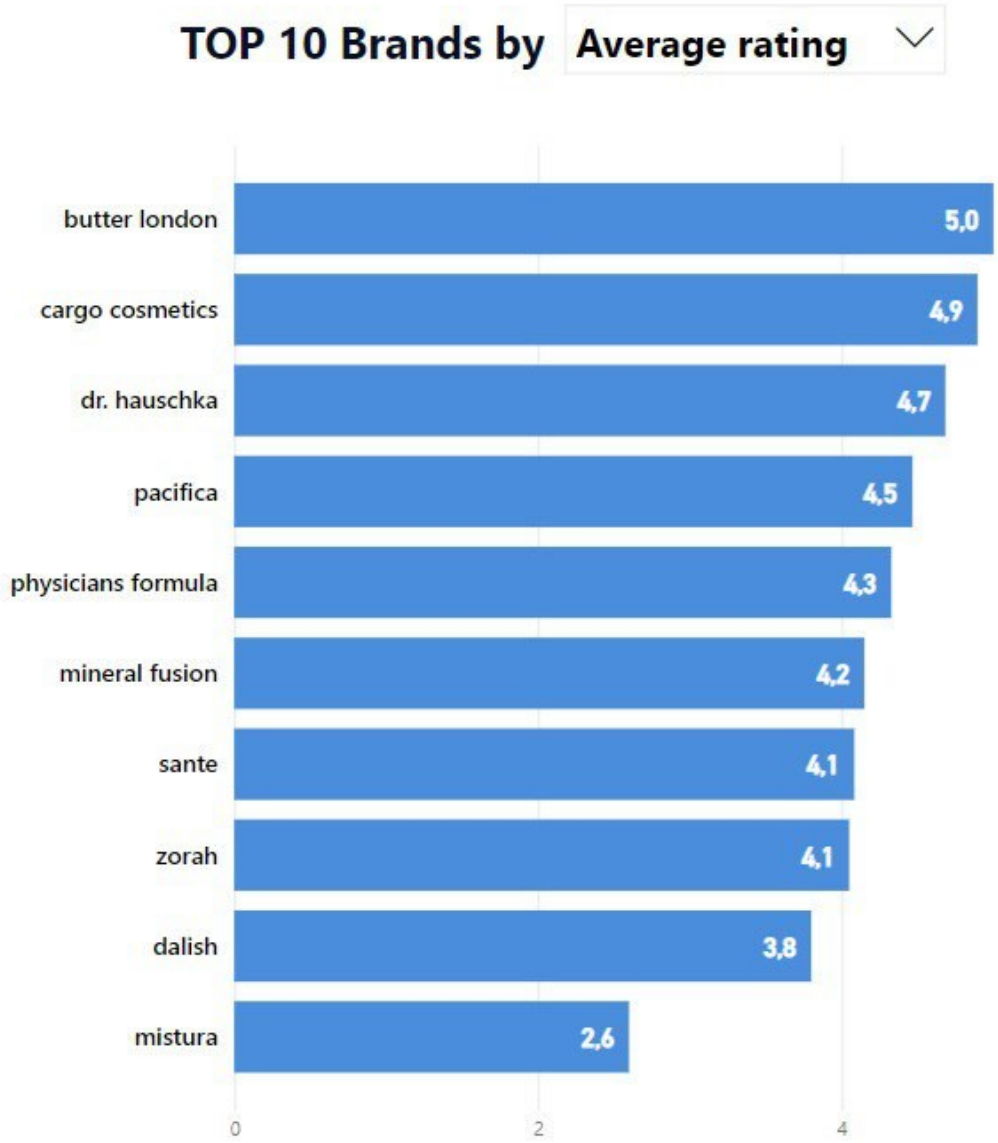
	updated_date	brand	price	currency	rating	product_type	converted_price
0	24-12-2017	nyx	10.00	USD	4.50	foundation	12.70
1	24-12-2017	nyx	12.00	USD	4.50	foundation	15.24
2	24-12-2017	nyx	12.00	USD	4.50	foundation	15.24
3	24-12-2017	nyx	12.00	USD	4.50	foundation	15.24
4	24-12-2017	nyx	10.00	USD	4.50	foundation	12.70

Візуалізація даних в Power BI

Можемо переглянути ТОП 10 брендів за середньою ціною:



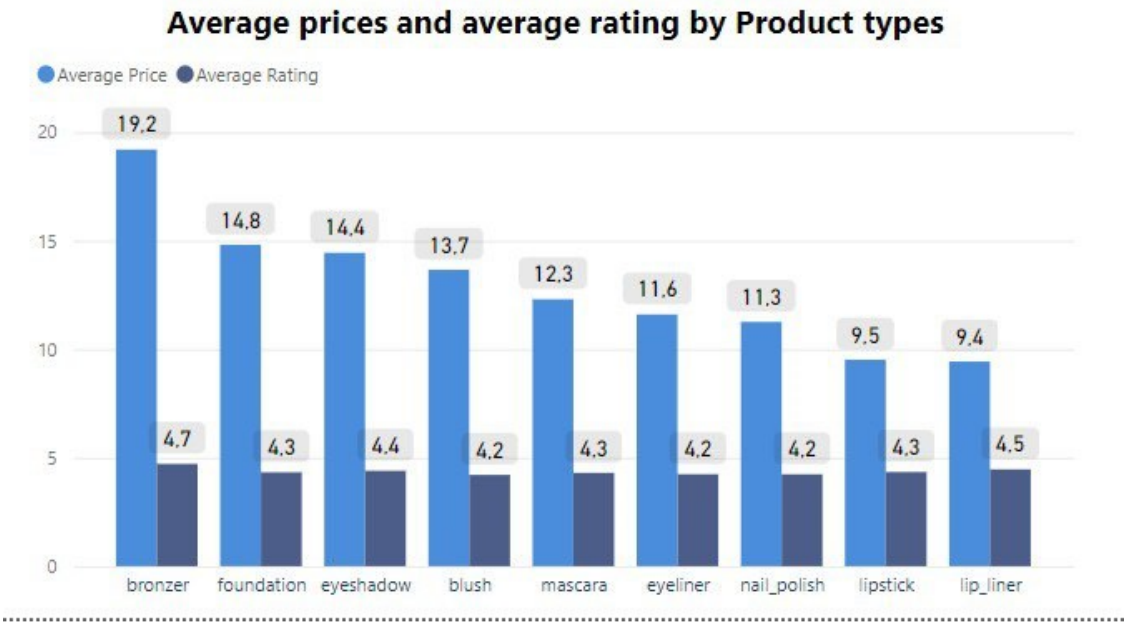
Або середнім рейтингом:



І відповідно відразу бачимо, що висока ціна не завжди відповідає високому рейтингу і навпаки. Бренд **Mistura** має найвищу ціну, але при цьому рейтинг найнижчий. Тоді як бренд **Physicians formula** має найнижчу ціну, але значення рейтингу при цьому середні.

На діаграмі розсіювання ми бачимо, що показник рейтингу лише частково пояснює значення ціни.

Те ж саме підтверджує аналіз середніх цін та середнього рейтингу по типам продуктів: не завжди високий рейтинг свідчить про високу ціну і навпаки.



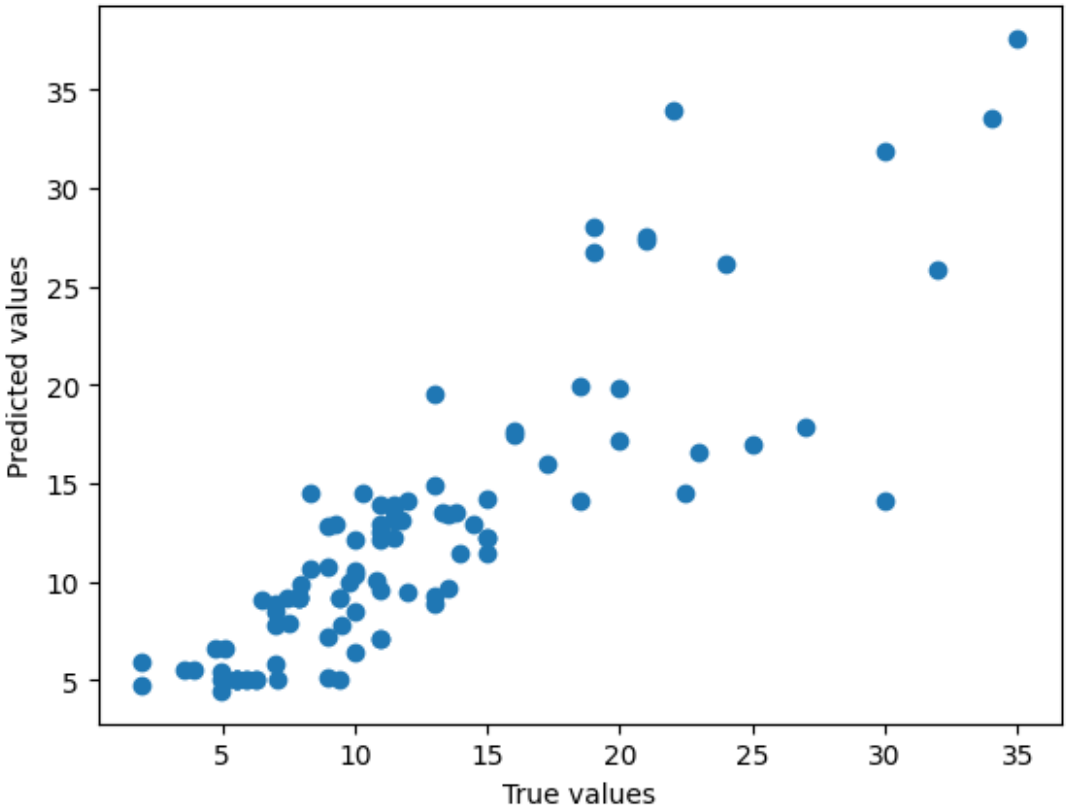
Навчання моделі

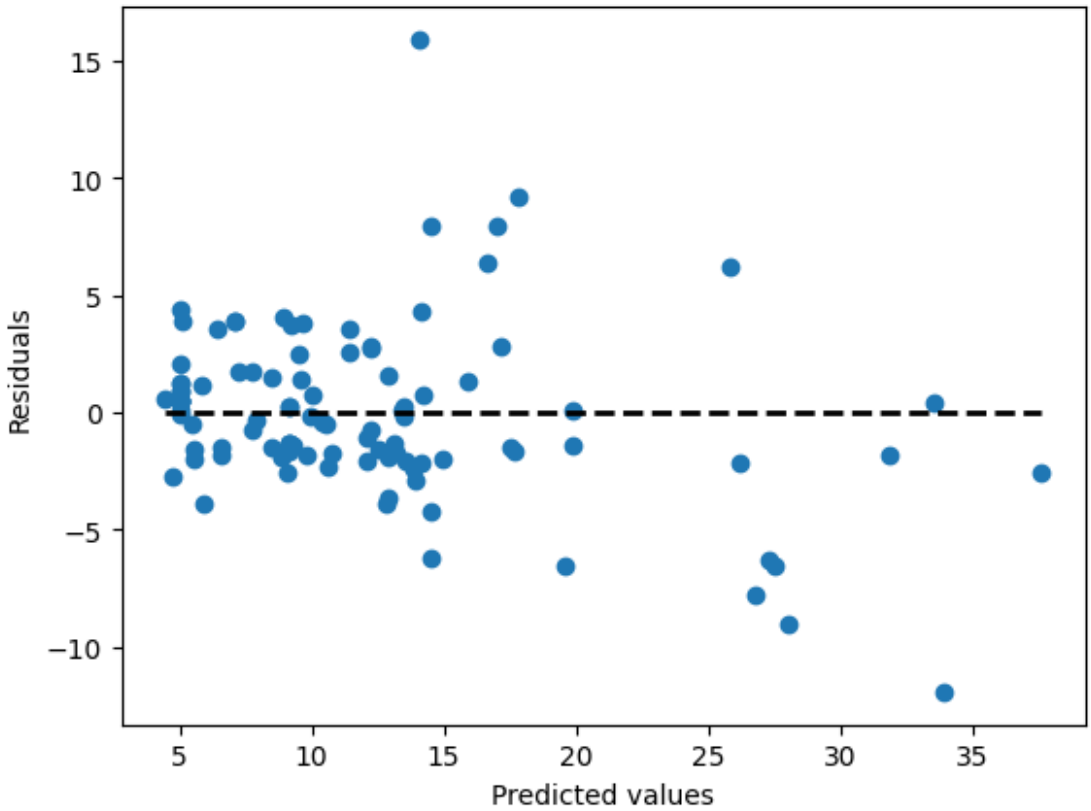
Лінійна регресія: тренування моделі машинного навчання, оцінка результатів.

```
# Encode the categorical features using pd.get_dummies()
df_encoded = pd.get_dummies(df, columns=['brand', 'product_type'])

# Split the data into training and testing sets
train_df, test_df = train_test_split(df_encoded, test_size=0.2, random_state=
42)

# Create numpy arrays for the features and target variables
X_train = train_df.drop(columns=['converted_price', 'updated_date', 'price',
```

```
'currency']).values
X_test = test_df.drop(columns=['converted_price', 'updated_date', 'price', 'c
urrency']).values
y_train = train_df['converted_price'].values
y_test = test_df['converted_price'].values

# Select a machine learning model and train it on the training data
model_lr = LinearRegression()
model_lr.fit(X_train, y_train)

# Evaluate the performance of the model on the testing data
y_pred = model_lr.predict(X_test)

r2 = model_lr.score(X_test, y_test)
print(f"R-squared: {r2}")

mse = mean_squared_error(y_test, y_pred)
print(f"Mean squared error: {mse}")

rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"Root mean squared error: {rmse}")

import matplotlib.pyplot as plt
plt.scatter(y_test, y_pred)
plt.xlabel("True values")
plt.ylabel("Predicted values")
plt.show()

plt.scatter(y_pred, y_test - y_pred)
plt.xlabel("Predicted values")
plt.ylabel("Residuals")
plt.hlines(y=0, xmin=y_pred.min(), xmax=y_pred.max(), colors='k', linestyles=
'--', lw=2)
plt.show()
```

```
R-squared: 0.7158721377479305
Mean squared error: 13.683220113603564
Root mean squared error: 3.69908368567184
```



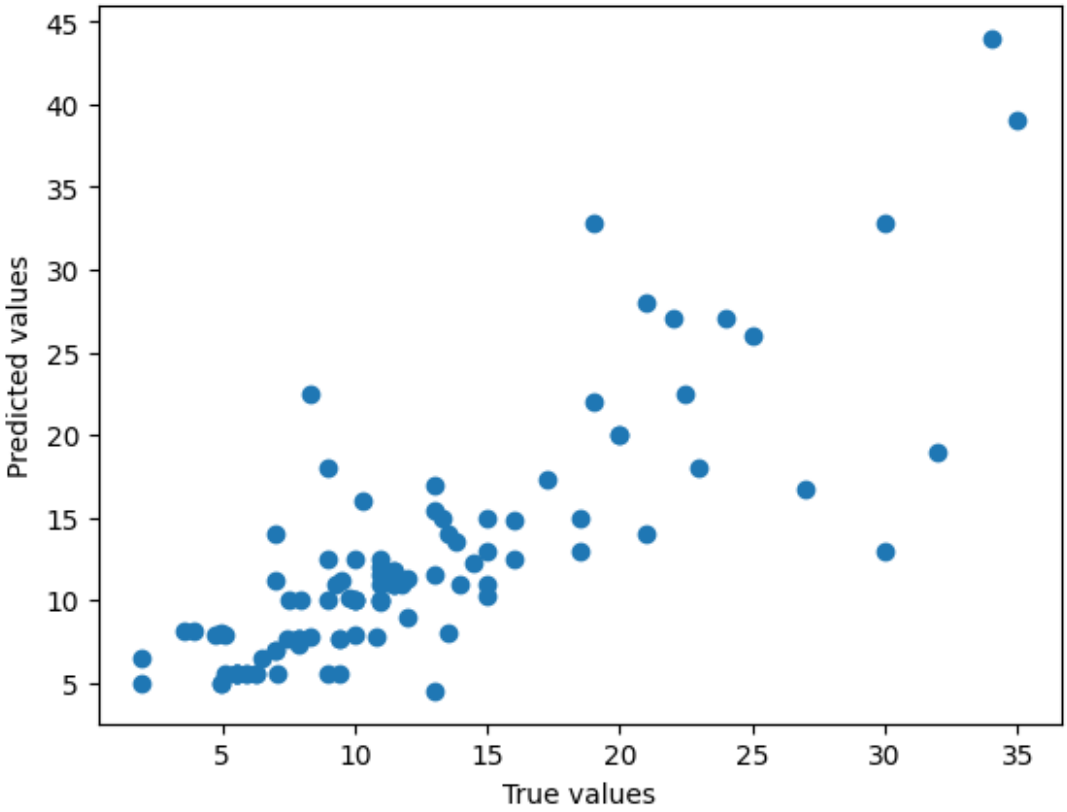


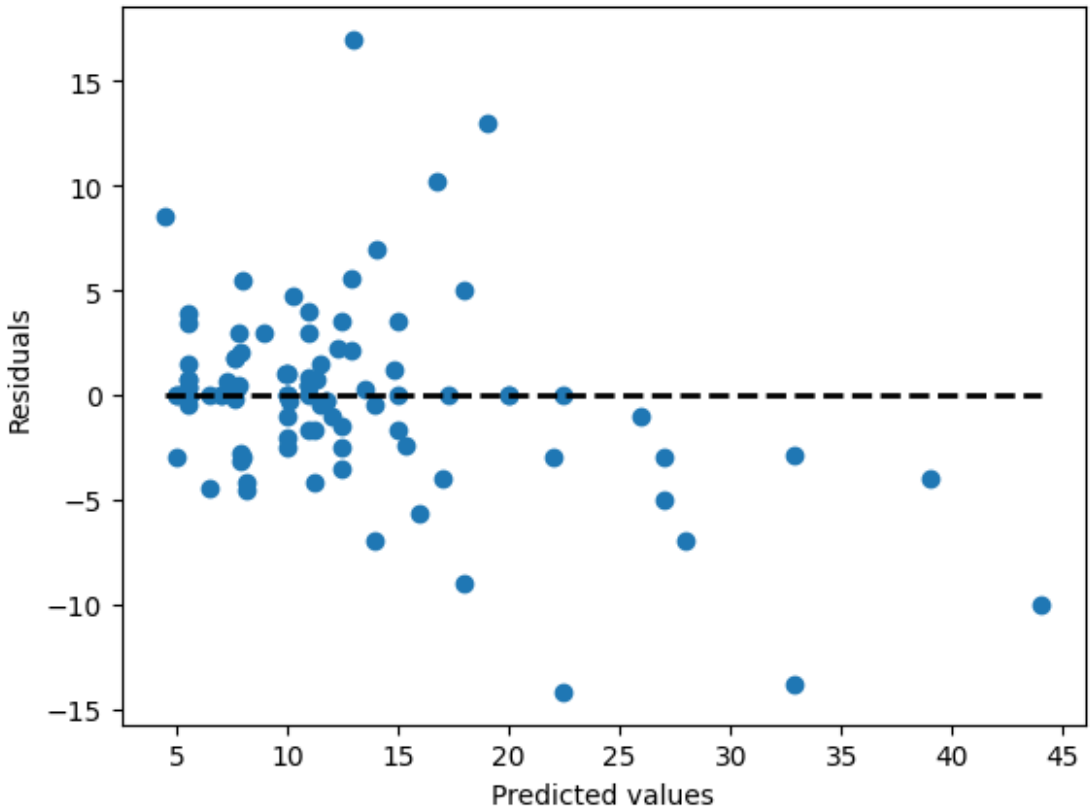
Дерево рішень: тренування моделі машинного навчання, оцінка результатів.

```
# Encode the categorical features using pd.get_dummies()
df_encoded = pd.get_dummies(df, columns=['brand', 'product_type'])
```

```
# Split the data into training and testing sets
train_df, test_df = train_test_split(df_encoded, test_size=0.2, random_state=
42)

# Create numpy arrays for the features and target variables
X_train = train_df.drop(columns=['converted_price', 'updated_date', 'price',
'currency']).values
X_test = test_df.drop(columns=['converted_price', 'updated_date', 'price', 'c
urrency']).values
y_train = train_df['converted_price'].values
y_test = test_df['converted_price'].values

# Select a machine learning model and train it on the training data
model_dt = DecisionTreeRegressor()
model_dt.fit(X_train, y_train)

# Evaluate the performance of the model on the testing data
y_pred = model_dt.predict(X_test)

r2 = model_dt.score(X_test, y_test)
print(f"R-squared: {r2}")

mse = mean_squared_error(y_test, y_pred)
print(f"Mean squared error: {mse}")

rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"Root mean squared error: {rmse}")

import matplotlib.pyplot as plt
plt.scatter(y_test, y_pred)
plt.xlabel("True values")
plt.ylabel("Predicted values")
plt.show()

plt.scatter(y_pred, y_test - y_pred)
plt.xlabel("Predicted values")
plt.ylabel("Residuals")
plt.hlines(y=0, xmin=y_pred.min(), xmax=y_pred.max(), colors='k', linestyles=
'--', lw=2)
plt.show()
```

```
R-squared: 0.6189667102409783
Mean squared error: 18.350056671871524
Root mean squared error: 4.28369661295843
```



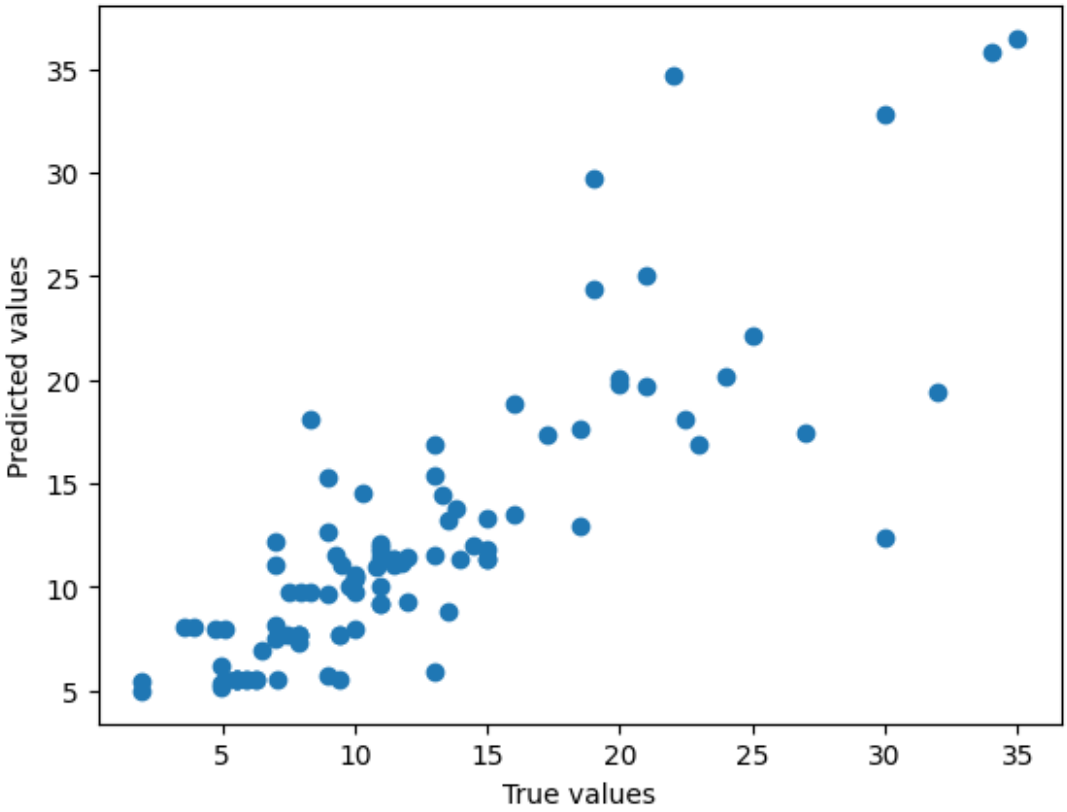


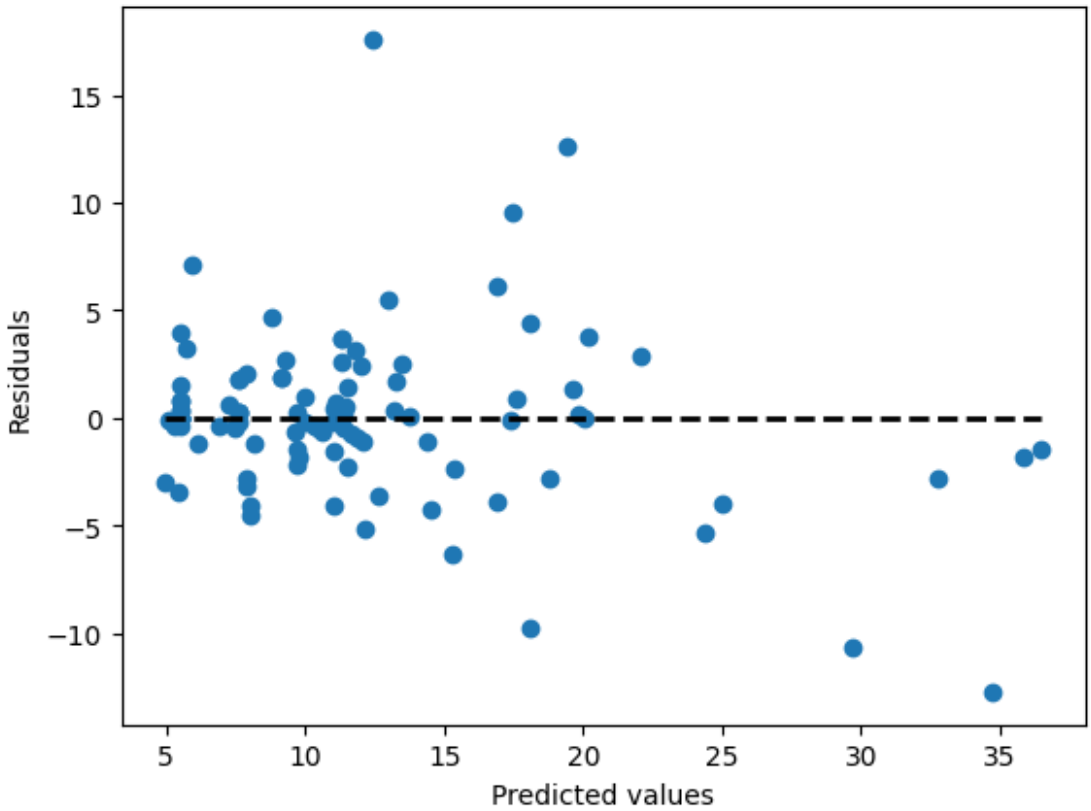
Random forest: тренування моделі машинного навчання, оцінка результатів.

```
# Encode the categorical features using pd.get_dummies()
df_encoded = pd.get_dummies(df, columns=['brand', 'product_type'])
```

```
# Split the data into training and testing sets
train_df, test_df = train_test_split(df_encoded, test_size=0.2, random_state=
42)

# Create numpy arrays for the features and target variables
X_train = train_df.drop(columns=['converted_price', 'updated_date', 'price',
'currency']).values
X_test = test_df.drop(columns=['converted_price', 'updated_date', 'price', 'c
urrency']).values
y_train = train_df['converted_price'].values
y_test = test_df['converted_price'].values

# Select a machine learning model and train it on the training data
model_rf = RandomForestRegressor(n_estimators=100, random_state=42)
model_rf.fit(X_train, y_train)

# Evaluate the performance of the model on the testing data
y_pred = model_rf.predict(X_test)

r2 = model_rf.score(X_test, y_test)
print(f"R-squared: {r2}")

mse = mean_squared_error(y_test, y_pred)
print(f"Mean squared error: {mse}")

rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"Root mean squared error: {rmse}")

import matplotlib.pyplot as plt
plt.scatter(y_test, y_pred)
plt.xlabel("True values")
plt.ylabel("Predicted values")
plt.show()

plt.scatter(y_pred, y_test - y_pred)
plt.xlabel("Predicted values")
plt.ylabel("Residuals")
plt.hlines(y=0, xmin=y_pred.min(), xmax=y_pred.max(), colors='k', linestyles=
'--', lw=2)
plt.show()
```

```
R-squared: 0.6868460744607181
Mean squared error: 15.081076732951784
Root mean squared error: 3.8834362017357495
```



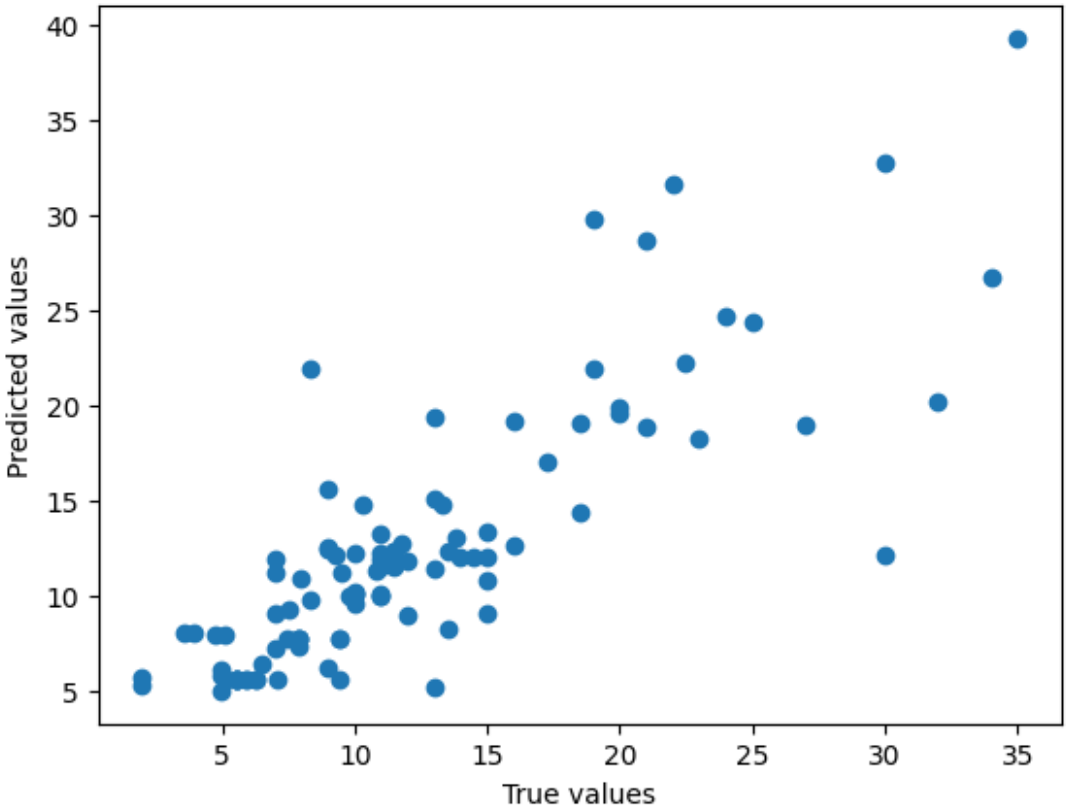


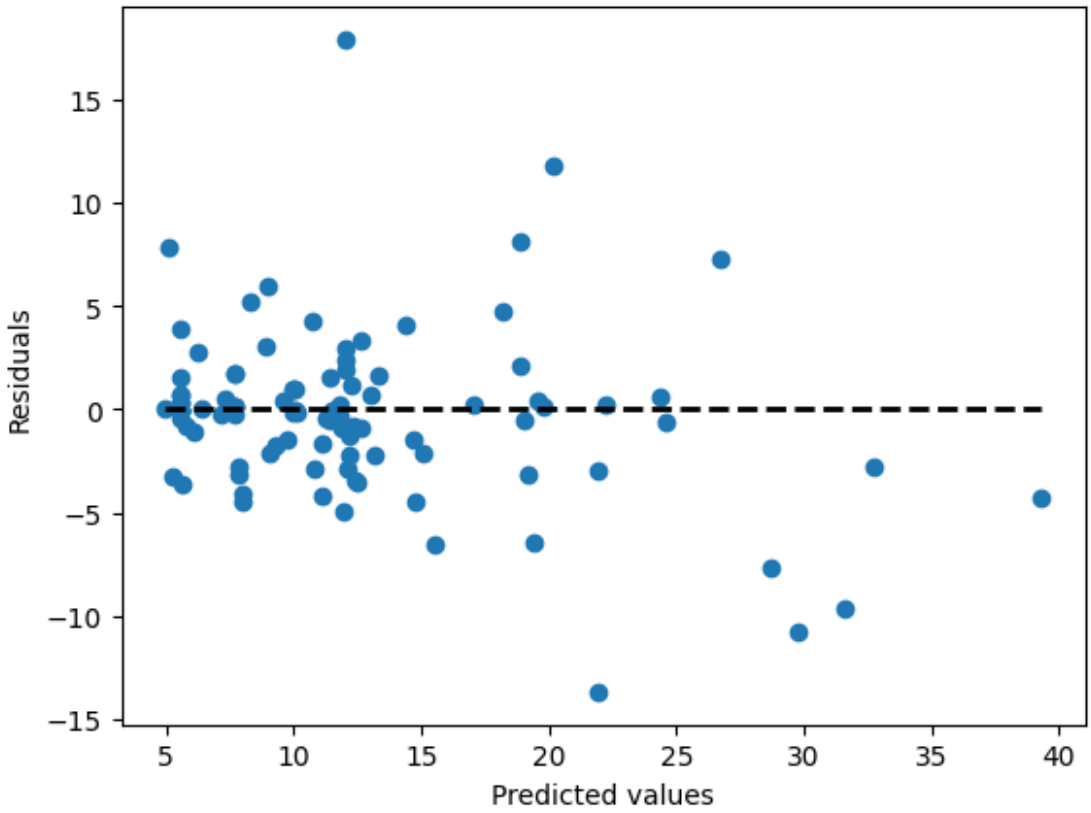
XGBoost: тренування моделі машинного навчання, оцінка результатів.

```
# Encode the categorical features using pd.get_dummies()
df_encoded = pd.get_dummies(df, columns=['brand', 'product_type'])
```

```
# Split the data into training and testing sets
train_df, test_df = train_test_split(df_encoded, test_size=0.2, random_state=
42)

# Create numpy arrays for the features and target variables
X_train = train_df.drop(columns=['converted_price', 'updated_date', 'price',
'currency']).values
X_test = test_df.drop(columns=['converted_price', 'updated_date', 'price', 'c
urrency']).values
y_train = train_df['converted_price'].values
y_test = test_df['converted_price'].values

# Select a machine learning model and train it on the training data
model_xgb = XGBRegressor(random_state=42)
model_xgb.fit(X_train, y_train)

# Evaluate the performance of the model on the testing data
y_pred = model_xgb.predict(X_test)

r2 = model_xgb.score(X_test, y_test)
print(f"R-squared: {r2}")

mse = mean_squared_error(y_test, y_pred)
print(f"Mean squared error: {mse}")

rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"Root mean squared error: {rmse}")

import matplotlib.pyplot as plt
plt.scatter(y_test, y_pred)
plt.xlabel("True values")
plt.ylabel("Predicted values")
plt.show()

plt.scatter(y_pred, y_test - y_pred)
plt.xlabel("Predicted values")
plt.ylabel("Residuals")
plt.hlines(y=0, xmin=y_pred.min(), xmax=y_pred.max(), colors='k', linestyles=
'--', lw=2)
plt.show()
```

```
R-squared: 0.6645087533292957
Mean squared error: 16.1568124223941
Root mean squared error: 4.019553759112335
```



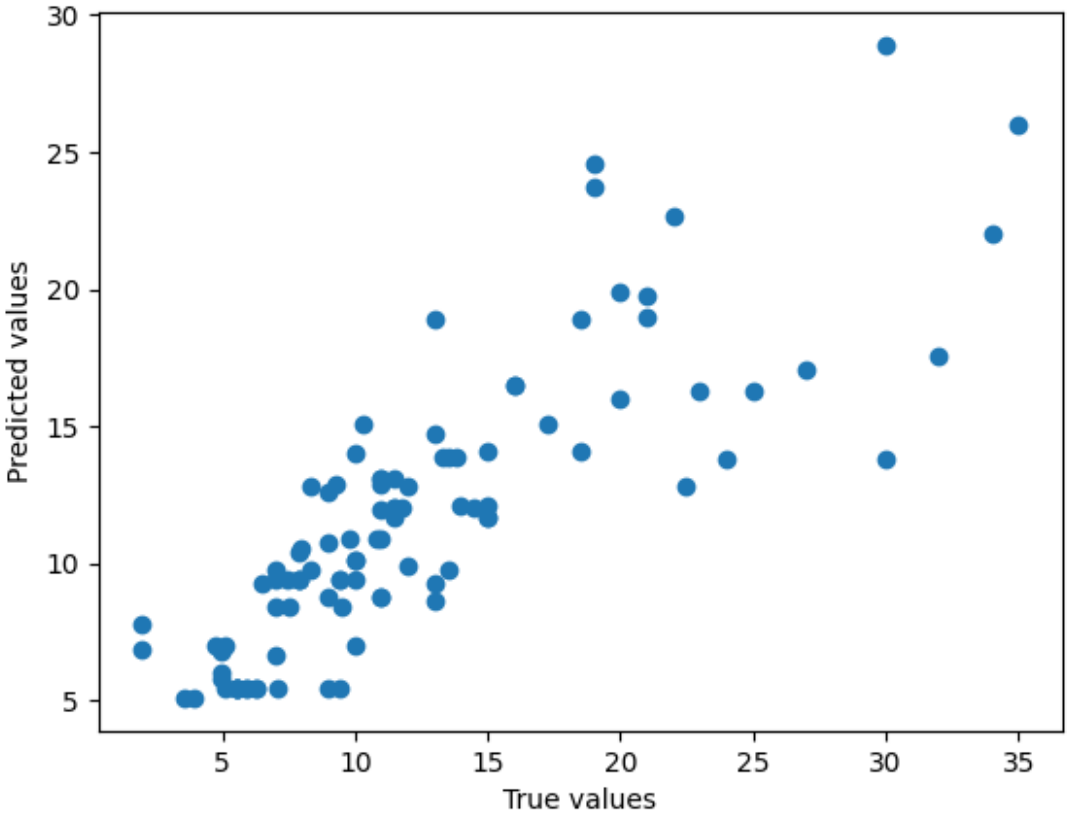


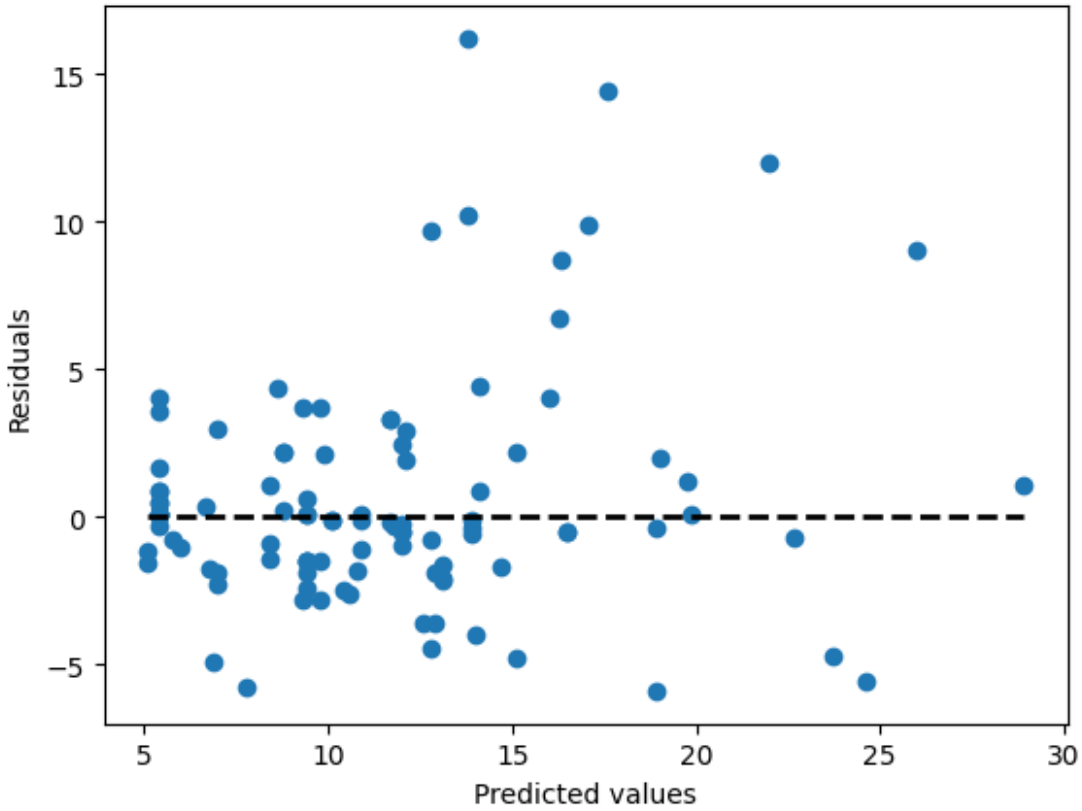
SVM: тренування моделі машинного навчання, оцінка результатів.

```
# Encode the categorical features using pd.get_dummies()
df_encoded = pd.get_dummies(df, columns=['brand', 'product_type'])
```

```
# Split the data into training and testing sets
train_df, test_df = train_test_split(df_encoded, test_size=0.2, random_state=
42)

# Create numpy arrays for the features and target variables
X_train = train_df.drop(columns=['converted_price', 'updated_date', 'price',
'currency']).values
X_test = test_df.drop(columns=['converted_price', 'updated_date', 'price', 'c
urrency']).values
y_train = train_df['converted_price'].values
y_test = test_df['converted_price'].values

# Select a machine learning model and train it on the training data
model_svm = SVR(kernel='linear')
model_svm.fit(X_train, y_train)

# Evaluate the performance of the model on the testing data
y_pred = model_svm.predict(X_test)

r2 = model_svm.score(X_test, y_test)
print(f"R-squared: {r2}")

mse = mean_squared_error(y_test, y_pred)
print(f"Mean squared error: {mse}")

rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"Root mean squared error: {rmse}")

import matplotlib.pyplot as plt
plt.scatter(y_test, y_pred)
plt.xlabel("True values")
plt.ylabel("Predicted values")
plt.show()

plt.scatter(y_pred, y_test - y_pred)
plt.xlabel("Predicted values")
plt.ylabel("Residuals")
plt.hlines(y=0, xmin=y_pred.min(), xmax=y_pred.max(), colors='k', linestyles=
'--', lw=2)
plt.show()
```

```
R-squared: 0.6679113205939602
Mean squared error: 15.992949306454973
Root mean squared error: 3.999118566191177
```





На основі використаних вище моделей можна зробити висновок щодо найбільшої доцільності використання лінійної регресії. Для підвищення точності спробуємо застосувати моделі, що використовують регуляризацію.

Ridge regularization

```
from sklearn.linear_model import Ridge

# Create a Ridge model with alpha = 0.5
ridge_model = Ridge(alpha=0.5)

# Train the model on the training data
ridge_model.fit(X_train, y_train)

# Evaluate the performance of the model on the testing data
y_pred = ridge_model.predict(X_test)

ridge_r2 = ridge_model.score(X_test, y_test)
print(f"R-squared with Ridge regularization: {ridge_r2}")

mse = mean_squared_error(y_test, y_pred)
print(f"Mean squared error with Ridge regularization: {mse}")

rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"Root mean squared error with Ridge regularization: {rmse}")
```

```
R-squared with Ridge regularization: 0.7255914115006548
Mean squared error with Ridge regularization: 13.215152810915349
Root mean squared error with Ridge regularization: 3.635265163769398
```

Lasso regularization

```
from sklearn.linear_model import Lasso

# Create a Lasso model with alpha = 0.5
lasso_model = Lasso(alpha=0.5)

# Train the model on the training data
lasso_model.fit(X_train, y_train)


# Evaluate the performance of the model on the testing data
y_pred = lasso_model.predict(X_test)

lasso_r2 = lasso_model.score(X_test, y_test)
print(f"R-squared with Lasso regularization: {lasso_r2}")

mse = mean_squared_error(y_test, y_pred)
print(f"Mean squared error with Lasso regularization: {mse}")

rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"Root mean squared error with Lasso regularization: {rmse}")
```

```
R-squared with Lasso regularization: 0.19738994558548784
Mean squared error with Lasso regularization: 38.65263319442412
Root mean squared error with Lasso regularization: 6.2171241900435055
```

Виходячи з цих результатів, регулярізація Ріджа трохи покращила продуктивність моделі, оскільки значення R-квадрат збільшилося з 0,7159 до 0,7256, а середня квадратична помилка та корінь з середньої квадратичної помилки зменшилися з 13,6832 до 13,2152 і з

3,6991 до 3,6353, відповідно. З іншого боку, регулярізація Лассо не показала належного результату на цьому наборі даних, оскільки значення R-квадрат значно впало з 0,7159 до 0,1974, а середня квадратична помилка та корінь з середньої квадратичної помилки значно зросли з 13,6832 до 38,6526 і з 3,6991 до 6,2171, відповідно. Це вказує на те, що регулярізація Лассо могла усунути важливі змінні або функції з моделі, що призвело до низької продуктивності. Підводячи підсумок, регулярізація Ріджа є хорошим вибором для покращення продуктивності моделі лінійної регресії, тоді як регулярізація Ласо може не підходити для цього конкретного набору даних.

Створимо новий набір даних та спрогнозуємо для нього ціну на основі Ridge regression

```
# Create a new dataframe with unique brands and product types and random valu
es for their rating
df_predict = df[['brand', 'product_type']].drop_duplicates()
df_predict['rating'] = [random.uniform(1, 5) for _ in range(len(df_predict))]

# Reset index for this dataframe
df_predict = df_predict.reset_index(drop=True)

# Encode the data for prediction
new_data_encoded = pd.get_dummies(df_predict, columns=['brand', 'product_type
']).values
new_predictions = ridge_model.predict(new_data_encoded)

# Create a new dataframe to store the predictions
df_pred = df_predict.copy()
df_pred['prediction'] = new_predictions.round(2)

# Print the new dataframe
print(df_pred)
```

```
            brand product_type  rating  prediction
0             nyx   foundation    2.72        9.25
1             nyx      bronzer    4.99        9.39
2             nyx        blush    2.71        6.91
3             nyx    lip_liner    1.67        5.54
4             nyx     lipstick    1.16        5.17
..            ...          ...     ...         ...
126    wet n wild      mascara    1.44        5.79
127  dr. hauschka      mascara    4.56       32.27
128       suncoat      mascara    2.01       18.23
129      pacifica      mascara    2.17       26.60
130    pure anada      mascara    3.81       12.79

[131 rows x 4 columns]
```

Загалом продуктивність моделі здається прийнятною, зі значенням R-квадрат 0,7256, що вказує на те, що модель може пояснити значну кількість дисперсії цільової змінної. Проте все ще є можливості для вдосконалення з точки зору зменшення середньоквадратичної помилки.

Таким чином, дана модель може бути використана при аналізі цін конкурентів та побудови власної політики ціноутворення. При запуску нового продукту за допомогою

даної моделі можна задати параметри, аналогічні до параметрів власного нового продукту, для брендів конкурентів та спрогнозувати їхні ціни за різного рівня рейтингу, і далі цю інформацію використати при прийнятті рішення про встановлення ціни на власний продукт.

Крім того, дану модель можна перебудувати на прогнозування рейтингу продукції в залежності від інших параметрів.

Проте, варто зазначити, що для побудови якіснішої моделі необхідна більша кількість інформації, більша кількість характеристик товарів (об’єм або вага продукції, класифікація, країна походження тощо), що дозволить більш точно прогнозувати дані.