Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій

Бакалаврська кваліфікаційна робота на тему:

«СИСТЕМНИЙ АНАЛІЗ ТА ПРОГНОЗУВАННЯ ЦІНИ НА НОУТБУКИ»

Виконав: студент 4 курсу, групи СА-21б
спеціальності 124 – «Системний аналіз»
_____ Максим ЯСИНОВСЬКИЙ
Керівник: Ph.D., ст. викл. каф. САІТ
_____ Ольга ВОЙЦЕХОВСЬКА
«08» 06 2025 р.
Рецензент: к.т.н., ст. викл. каф. КН
_____Сергій ПЕТРИШИН
«10» 06 2025 р.

Допущено до захисту

Завідувач кафедри САІТ
_____д.т.н., проф. Віталій МОКІН
«09» 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій
Рівень вищої освіти – перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність 124 – «Системний аналіз»
Освітньо-професійна програма – Системний аналіз

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____д.т.н., проф. Віталій МОКІН
"28" 05 2025 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Ясиновському Максиму Олександровичу

1. Тема роботи: «Системний аналіз та прогнозування ціни на ноутбуки»
керівник роботи: Ольга ВОЙЦЕХОВСЬКА, Ph.D., ст. викл. каф. САІТ
затверджені наказом ВНТУ від «28» 05 2025 року № 97
2. Термін подання студентом роботи 31.05.2025 р.
3. Вихідні дані до роботи:
Kaggle Dataset «Laptop sales price prediction 2024»
https://www.kaggle.com/datasets/siddiquifaiznaeem/laptop-sales-price-prediction-dataset-2024/data
4. Зміст текстової частини:
1) Аналіз предметної області;
2) Вибір інструментарію та розвідувальний аналіз даних;
3) Розроблення та аналіз моделей прогнозування.
5. Перелік ілюстративного матеріалу:
1) Структура системи прогнозування та управління ціноутворенням на ринку ноутбуків;
2) Розподіл цільової ознаки;
3) Розподіл ознак;
4) Теплова карта кореляційної матриці
5) Порівняння точності моделей;
6) Важливість ознак оптимальної моделі.

6. Консультанти розділів роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 28.05.25

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	01.04	15.04	вик
2	Вибір інструментів	15.04	20.04	вик
3	Розвідувальний аналіз даних	01.05	10.05	вик
4	Результати прогнозування	10.05	20.05	вик
5	Оформлення матеріалів до захисту БКР	20.05	30.05	вик

Студент _____ Максим ЯСИНОВСЬКИЙ

Керівник роботи _____ Ольга ВОЙЦЕХОВСЬКА

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 80 сторінок формату А4, на яких є 26 рисунків, список використаних джерел містить 21 найменування.

В роботі виконано підвищення точності прогнозування цін на ноутбуки шляхом застосування системного аналізу та методів машинного навчання з оптимізацією точності моделі як критерію ефективності.

Проведено системний аналіз предметної області. Проаналізовано роль різних стейкхолдерів та обґрунтовано необхідність точного прогнозування цін для підтримки їхніх рішень. Запропоновано концептуальну модель цієї складної системи, що підкреслює багаторівневу взаємодію її елементів.

Обґрунтовано вибір програмного інструментарію та реалізовано етапи підготовки даних. Виявлено ключові ознаки, які значуще впливають на ціну ноутбуків.

Реалізовано побудову, навчання та оцінювання моделей машинного навчання для прогнозування ціни ноутбуків з використанням методів системного аналізу; здійснено порівняння точності моделей за обраними критеріями.

Отримані результати мають значну практичну цінність для підтримки прийняття рішень на різних рівнях: від оперативного до стратегічного.

Ключові слова: прогнозування цін, системний аналіз, машинне навчання, підтримка рішень, складна система, аналіз даних, важливість ознак.

ABSTRACT

The bachelor's qualification thesis consists of 80 A4 pages and includes 26 figures. The list of references contains 21 sources.

In this work, the accuracy of laptop price forecasting was enhanced through the application of system analysis and machine learning methods, with model accuracy optimization as the effectiveness criterion.

A system analysis of the subject area was conducted. The role of various stakeholders was analyzed, and the necessity of accurate price forecasting to support their decision-making was substantiated. A conceptual model of this complex system was proposed, emphasizing the multi-level interaction of its elements.

The choice of software tools was justified, and the data preparation stages were implemented. Key features significantly influencing laptop prices were identified.

The construction, training, and evaluation of machine learning models for laptop price forecasting were implemented using system analysis methods; a comparison of model accuracy based on selected criteria was performed.

The obtained results have significant practical value for supporting decision-making at various levels: from operational to strategic.

Keywords: price forecasting, system analysis, machine learning, decision support, complex system, data analysis, feature importance.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному динамічному середовищі, що характеризується високим рівнем конкуренції, стрімкими технічними інноваціями та складністю ринкових процесів, здатність точно прогнозувати ціни на високотехнологічні товари, зокрема ноутбуки, набуває критичного значення. Для виробників, дистриб'юторів, маркетологів, інвесторів і кінцевих споживачів, ефективне цінове планування є ключовим фактором прийняття обґрунтованих рішень. Це зумовлює актуальність створення систем прогнозування цін на основі методів аналізу даних і машинного навчання як інструменту підтримки прийняття рішень у складних динамічних системах.

З огляду на складну ієрархічну структуру ринку електроніки, де взаємодіють численні агенти — виробники, постачальники, роздрібні мережі, цифрові платформи та кінцеві споживачі, — задача прогнозування цін потребує міждисциплінарного підходу. Поєднання методів системного аналізу з алгоритмами машинного навчання дозволяє не лише моделювати нелінійні залежності між ознаками, а й виявляти приховані закономірності, що є основою для прийняття стратегічних та оперативних рішень в умовах невизначеності.

Мета і задачі дослідження. Метою дослідження є підвищення точності прогнозування цін на ноутбуки шляхом застосування системного аналізу та методів машинного навчання з оптимізацією точності моделі як критерію ефективності.

Для досягнення поставленої мети необхідно розв'язати такі задачі:

– Провести системний аналіз предметної області;

– Обґрунтувати вибір програмного інструментарію та реалізувати етапи підготовки даних, включаючи очищення, трансформацію, нормалізацію та первинний аналіз;

– Реалізувати побудову, навчання та оцінювання моделей машинного навчання для прогнозування ціни ноутбуків з використанням методів системного аналізу; здійснити порівняння точності моделей за обраними критеріями.

Об’єктом дослідження є процес прогнозування цін на ноутбуки як завдання аналізу та оптимізації у складній економічній системі ринку комп’ютерної техніки.

Предметом дослідження є методи системного аналізу та машинного навчання, що застосовуються для побудови моделей прогнозування та підвищення точності оцінки цін у багатофакторному середовищі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис об’єкта досліджень

Об’єктом даного дослідження є процес формування та динаміка ціни на ноутбуки на сучасному споживчому ринку. Цей об'єкт розглядається як складна система різної природи, що підлягає аналізу, моделюванню та прогнозуванню з використанням математичних методів та інформаційних технологій.

У сучасному висококонкурентному середовищі, що характеризується швидкими технологічними змінами та значним попитом на комп'ютерну техніку, здатність точно прогнозувати ціни на ноутбуки набуває критичного значення. Ноутбуки є невід'ємною частиною як повсякденного життя, так і професійної діяльності, від навчання та розваг до виконання складних обчислювальних завдань у бізнесі та науці. Ринок ноутбуків є надзвичайно динамічним, на нього впливає безліч факторів: від появи нових моделей та технологій (наприклад, процесорів, відеокарт, типів пам'яті) до глобальних економічних тенденцій, коливань курсу валют, змін у логістичних ланцюжках та сезонності попиту [1,2].

Для різних суб'єктів ринку точне прогнозування цін на ноутбуки є запорукою ефективного функціонування:

– Для споживачів: Можливість передбачити зміни цін дозволяє здійснити покупку у найбільш вигідний момент, оптимізуючи особисті витрати.

– Для роздрібних продавців та дистриб'юторів: Точні прогнози дозволяють оптимізувати управління запасами, уникнути надлишків або дефіциту товару, формувати ефективні цінові стратегії, планувати маркетингові кампанії та збільшувати прибутковість. Невірне прогнозування може призвести до значних фінансових втрат через зниження цін на застарілий товар або втрачені можливості через відсутність товару в наявності.

– Для виробників: Прогнозування цін на кінцевий продукт допомагає у плануванні виробництва, формуванні цінової політики, оцінці

конкурентоспроможності нових моделей та виборі оптимальних термінів виведення їх на ринок.

– Для інвесторів та аналітиків: Точні прогнози цін на знакові товари, такі як ноутбуки, можуть бути індикатором загальних тенденцій на ринку електроніки та споживчих товарів, впливаючи на інвестиційні рішення [2-4].

Традиційні методи прогнозування, що базуються на статистичних моделях, часто виявляються недостатньо ефективними для аналізу таких складних систем з великою кількістю взаємопов'язаних і нелінійних факторів. Саме тому виникає потреба у застосуванні більш досконалих підходів.

Суть проблеми полягає у розробці математичних методів та інформаційних технологій аналізу, моделювання та прогнозування ціни на ноутбуки в умовах, коли цінова динаміка є результатом складної взаємодії численних факторів, а традиційні методи прогнозування виявляються недостатньо точними. Застосування методів штучного інтелекту (ШІ) дозволяє перейти до якісно нового рівня аналізу, оскільки ШІ-моделі здатні виявляти нелінійні залежності та приховані патерни у великих обсягах даних, що є характерним для сучасних ринків [1-3].

Ключові аспекти проблеми, які вирішуються за допомогою ШІ:

– Багатофакторність та складність даних: Ціна ноутбука залежить від десятків параметрів: технічні характеристики (процесор, ОЗУ, відеокарта, обсяг накопичувача, діагональ екрана), бренд, серія, дизайн, наявність операційної системи, відгуки користувачів, рік випуску, економічні показники (інфляція, курси валют), логістичні витрати, маркетингові кампанії та навіть сезонність попиту (наприклад, передсвяткові періоди). Ручний аналіз такої кількості взаємозалежних факторів є практично неможливим. Методи аналізу даних та математичної статистики, які є частиною теоретичного змісту предметної області системного аналізу, дозволяють збирати та попередньо обробляти ці дані, тоді як ШІ-алгоритми здатні автоматично виявляти складні взаємозв'язки між ними.

– Нелінійність залежностей: Зміна одного параметра (наприклад, вихід нової моделі процесора) може мати нелінійний, а іноді й непередбачуваний вплив на ціну. Класичні лінійні моделі не завжди здатні адекватно відобразити такі залежності. Комп'ютерне моделювання на основі ШІ-методів, таких як нейронні мережі або ансамблеві моделі, може виявляти ці нелінійні взаємозв'язки.

– Динамічність ринку та "довгий хвіст": Ринок ноутбуків постійно змінюється, з'являються нові моделі, старі швидко застарівають, знижуються ціни. Існують також "довгі хвости" – велика кількість рідкісних або нішевих моделей з нестабільним попитом. Прогнозування в таких умовах вимагає адаптивних моделей, що постійно навчаються. Методи машинного навчання дозволяють будувати саме такі адаптивні моделі, які можуть навчатися на нових даних та коригувати свої прогнози.

– Великі обсяги даних: Сучасні інтернет-магазини та агрегатори цін генерують колосальні обсяги даних про товари, їхні характеристики, ціни та історію продажів. Ефективна робота з такими "великими даними" можлива лише за допомогою інформаційних технологій аналізу даних, що включають спеціалізовані бібліотеки та фреймворки ШІ.

– Підтримка прийняття рішень: Кінцевою метою прогнозування є прийняття рішень. Точний прогноз ціни дає можливість бізнесу приймати оптимальні рішення щодо закупівель, ціноутворення, маркетингу та управління запасами. Це безпосередньо відповідає теорії керування та прийняття рішень, яка є наріжним каменем системного аналізу. ШІ-моделі виступають як інструмент експертного оцінювання, автоматизуючи та об'єктивізуючи процес прогнозування [1-4].

Таким чином, використовуючи математичне і комп'ютерне моделювання на основі ШІ-алгоритмів, будуть розроблені методи, методика та технології для побудови прогнозної моделі ціни на ноутбуки. Це дозволить оцінювати ризики неефективних рішень та надавати суб'єктам ринку обґрунтовані рекомендації, що є важливим внеском у розвиток оптимізації систем та процесів в умовах

ринкової економіки. Застосування штучного інтелекту дозволяє не тільки підвищити точність прогнозів, але й зробити процес їх отримання більш автоматизованим, гнучким та масштабованим, що є надзвичайно цінним для складних систем [1-4].

1.2 Аналіз існуючих підходів

Прогнозування ціни на ноутбуки є складною задачею, що вимагає вибору ефективних математичних методів та інформаційних технологій. З огляду на динамічний характер ринку, багатофакторність впливів та нелінійні взаємозв'язки між даними, класичні статистичні підходи часто виявляються недостатніми. Сучасні методи математичного моделювання, аналізу даних, оптимізації та дослідження операцій, що належать до теоретичного змісту предметної області системного аналізу, дозволяють застосовувати більш досконалі алгоритми, зокрема з арсеналу машинного навчання.

Вибір оптимального методу прогнозування для поставленої задачі ґрунтується на кількох критеріях: здатність обробляти великі обсяги даних, виявляти складні, нелінійні залежності, стійкість до зашумлених даних, можливість оцінювати важливість ознак та забезпечувати прийнятну точність прогнозу. Нижче представлено огляд основних методів, які є потенційно придатними для вирішення задачі прогнозування ціни на ноутбуки.

Незважаючи на обмеження у складних системах, важливо розпочати аналіз з класичних статистичних підходів, які є фундаментом для розуміння більш складних моделей [1-4].

Лінійна регресія (Linear Regression): Це базовий статистичний метод, який моделює лінійну залежність між залежною змінною (ціна ноутбука) та однією або кількома незалежними змінними (характеристики ноутбука, ринкові показники). Перевагою є простота інтерпретації та швидкість обчислень. Недоліком є нездатність адекватно відображати нелінійні взаємозв'язки, що є

характерним для ціноутворення на технологічному ринку. Застосовується як відправна точка або для виявлення сильних лінійних кореляцій [3].

Часові ряди (ARIMA, SARIMA, Prophet): Моделі авторегресії, інтегрованого ковзного середнього (ARIMA) та її сезонні варіації (SARIMA) традиційно використовуються для прогнозування часових рядів. Вони враховують автокореляцію даних, тренди та сезонність. Для прогнозування цін на ноутбуки, особливо з урахуванням сезонних коливань попиту (наприклад, передсвяткові періоди), ці методи можуть бути корисними. Проте, їхня ефективність знижується при великій кількості зовнішніх факторів, що впливають на ціну, які не відображаються в самому часовому ряді. Модель Prophet від Facebook є більш гнучким інструментом для прогнозування часових рядів з вираженою сезонністю та трендами, що може бути актуальним для цінового аналізу [4].

Саме методи машинного навчання, що належать до математичного моделювання та аналізу даних, дозволяють долати обмеження класичних статистичних підходів, виявляючи складні нелінійні залежності у великих обсягах даних.

Дерева рішень (Decision Trees): Ці моделі будують ієрархічну структуру правил "якщо-то", що дозволяє розділяти дані на менші підмножини. Для прогнозування ціни (задача регресії) дерево рішень визначає середнє значення ціни для кожної кінцевої гілки (листового вузла). Переваги: простота інтерпретації (можна візуалізувати правила), здатність працювати з категоріальними та числовими даними, відносна стійкість до викидів. Недоліки: схильність до перенавчання, особливо на складних даних, та висока чутливість до невеликих змін у вхідних даних, що може призводити до нестабільних прогнозів [5-7].

Метод опорних векторів (Support Vector Machines – SVM): Хоча SVM часто асоціюється з класифікацією, він має розширення для регресії (Support Vector Regression – SVR). SVR знаходить оптимальну гіперплощину, яка максимально точно апроксимує дані, мінімізуючи помилки в межах заданого

порогу. Переваги: ефективність у багатовимірному просторі, хороша узагальнююча здатність, навіть при обмежених обсягах даних. Недоліки: складність інтерпретації, чутливість до вибору параметрів (ядерної функції та гіперпараметрів), а також обчислювальна складність при великих наборах даних [8].

Ансамблеві методи (Ensemble Methods): Ці методи поєднують декілька "слабких" моделей для створення більш "сильного" та точного прогнозу. Вони є особливо актуальними для прогнозування ціни, оскільки дозволяють врахувати різноманітні аспекти впливу факторів.

Випадковий ліс (Random Forest): Базується на принципі "баггінгу" (Bootstrap Aggregating), де будується велика кількість незалежних дерев рішень, кожне з яких навчається на випадковій підмножині даних. Кінцевий прогноз (ціна) визначається як середнє значення прогнозів усіх дерев. Переваги: висока точність, стійкість до перенавчання (завдяки рандомізації та усередненню), здатність оцінювати важливість ознак. Ефективний як при малих, так і при великих обсягах даних. Недоліки: менша інтерпретованість порівняно з одним деревом, значні обчислювальні ресурси при великій кількості дерев [9-11].

Градиентний бустинг (Gradient Boosting Machines – GBM, XGBoost, LightGBM, CatBoost): Ці методи послідовно будують дерева рішень, де кожне наступне дерево виправляє помилки попередніх. Вони "будують" модель, поступово зменшуючи помилку. XGBoost, LightGBM та CatBoost є оптимізованими реалізаціями градієнтного бустингу, що забезпечують високу швидкість та точність. Переваги: виняткова точність, особливо на структурованих даних, висока швидкість роботи (оптимізовані реалізації), потужні механізми для боротьби з перенавчанням. Недоліки: складність інтерпретації, чутливість до "шуму" в даних та необхідність ретельного підбору гіперпараметрів. Ці методи є одними з найбільш ефективних для задач прогнозування ціни [12,13].

AdaBoost (Adaptive Boosting): Ще один бустинговий алгоритм, який послідовно навчає слабкі класифікатори (або регресори), надаючи більшої ваги

тим спостереженням, які були помилково класифіковані на попередніх ітераціях. Він ефективний для покращення точності базових моделей, але може бути чутливим до викидів.

Багатошаровий перцептрон (Multi-Layer Perceptron – MLP) / Нейронні мережі: Це моделі штучних нейронних мереж, що складаються з кількох шарів взаємопов'язаних "нейронів". MLP здатні виявляти надзвичайно складні, нелінійні та приховані залежності у даних, що є критично важливим для прогнозування ціни на ноутбуки. Переваги: висока точність на великих обсягах даних, здатність моделювати складні взаємодії між ознаками. Недоліки: високі вимоги до обчислювальних ресурсів та обсягів даних для ефективного навчання, складність інтерпретації "чорного ящика", схильність до перенавчання при недостатній кількості даних та неправильному виборі архітектури [14].

З огляду на специфіку задачі прогнозування ціни на ноутбуки, що характеризується багатофакторністю, нелінійністю залежностей та значними обсягами даних, найбільш перспективними для подальшого дослідження та реалізації є ансамблеві методи машинного навчання, зокрема Випадковий ліс та Градиентний бустинг (XGBoost, LightGBM). Вони демонструють високу точність, стійкість до перенавчання та ефективність у виявленні складних закономірностей.

Крім того, варто розглянути Багатошаровий перцептрон (MLP) як альтернативний підхід, особливо якщо наявні дані містять дуже складні, неочевидні взаємозв'язки, які можуть бути краще виявлені нейронними мережами.

Дерева рішень можуть бути використані як базові моделі для розуміння основних правил ціноутворення та для подальшого інтегрування в ансамблеві підходи. Лінійна регресія може слугувати як бенчмарк для оцінки ефективності складніших моделей.

Цей аналіз є основою для подальшого дослідження операцій та оптимізації систем, які будуть реалізовані під час розробки інформаційної технології прогнозування ціни.

1.3 Системний аналіз складної системи прогнозування на ринку ноутбуків

Складна система, що лежить в основі даного дослідження, — це інформаційно-економічна система прогнозування та управління ціноутворенням на ринку ноутбуків (рис.1.1). Вона охоплює три рівні ієрархії:

– Елементи системи: окремі ноутбуки як товари з технічними характеристиками, цінами, брендами, а також дані про економічні та ринкові умови (валютні курси, сезонність, тенденції).

– Підсистеми: маркетингові, логістичні, виробничі та аналітичні підсистеми, які приймають рішення на основі аналізу даних (зокрема, прогнозованих цін).

– Система в цілому: ринок ноутбуків як складна саморегульована система, де рішення окремих підсистем взаємодіють, формуючи цінову динаміку й визначаючи ефективність функціонування системи загалом.

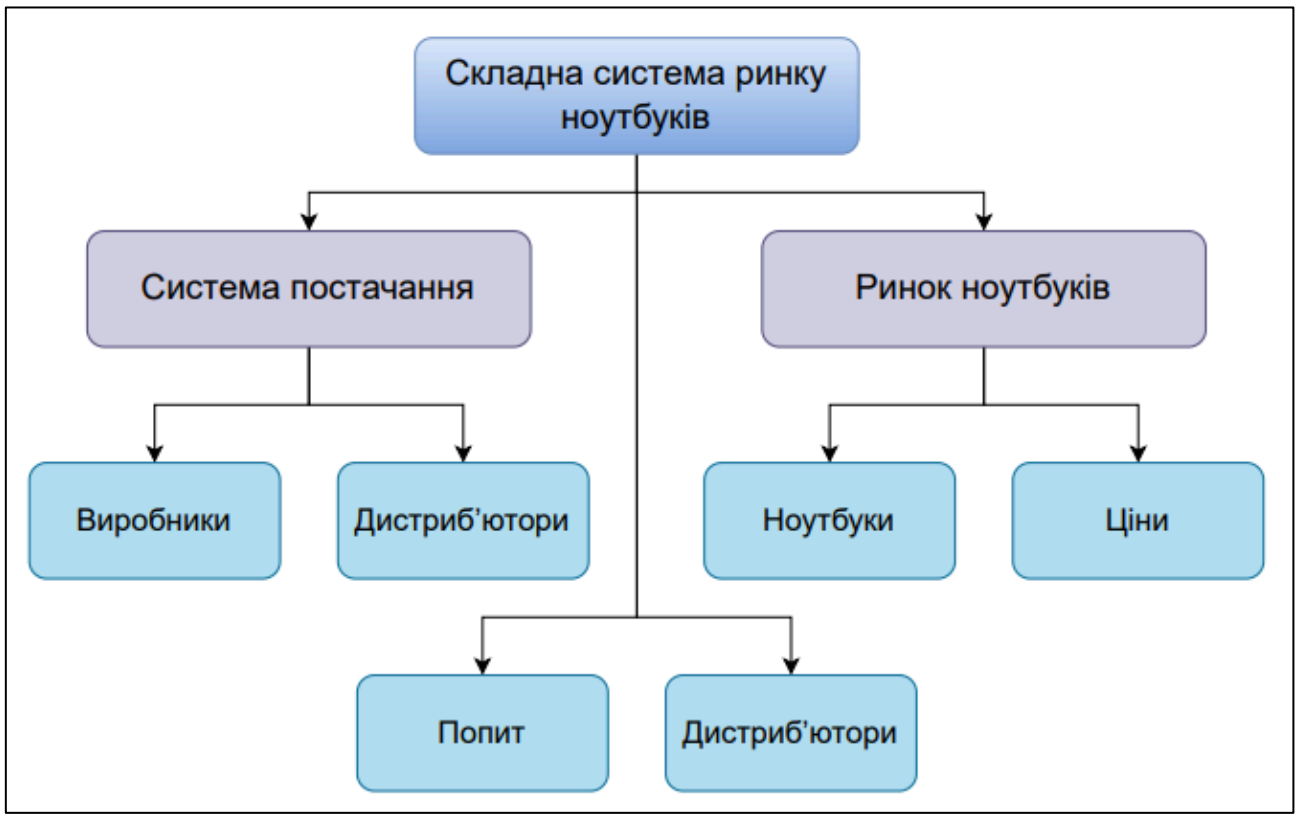

Рисунок 1.1 – Інформаційно-економічна система прогнозування та управління ціноутворенням на ринку ноутбуків

Ця система має ознаки складності через такі характеристики:

– Невизначеність умов: вплив глобальних чинників (економічні кризи, геополітична ситуація, інновації),

– Мультифакторність: десятки взаємозалежних ознак,

– Динамічність: швидкі зміни в моделях, технологіях, попиті та пропозиції,

– Нелінійність і взаємодія елементів: вплив одного параметра може змінити поведінку системи в цілому.

Назва складної системи: Інформаційно-аналітична система підтримки прийняття рішень щодо ціноутворення ноутбуків на основі прогнозної аналітики.

Критерій оптимізації: Максимізація точності прогнозу ціни ноутбуків (R2 score).

Формулювання задачі оптимізації: Знайти параметри моделі θ, які мінімізують помилку прогнозу: $\min_\theta \sum (y_i - \hat{y}_i)^2$, де y_i — фактична логарифмічна ціна, $\hat{y}_i$ — прогнозована ціна моделю.

Обмеження:

– Обмеження на інтерпретованість моделі (вибір explainable AI — Random Forest, а не чорних моделей);

– Обмеження за часом навчання моделей (обчислювальні ресурси);

– Вимога узагальнення — уникнення перенавчання на тренувальних даних.

Результат оптимізації — це модель, яка дозволяє точно передбачати ціну для нових ноутбуків. Цей результат може бути використаний для:

– Побудови стратегій закупівель та складування;

– Динамічного ціноутворення;

– Маркетингових рішень щодо просування моделей;

– Прогнозування рентабельності нових лінійок продукції [15].

1.4 Висновки

У першому розділі здійснено глибокий аналіз предметної області прогнозування цін на ноутбуки в умовах динамічного, багатофакторного та висококонкурентного ринку. Обґрунтовано доцільність розгляду процесу ціноутворення як складної інформаційно-економічної системи, в якій взаємодіють численні елементи, підсистеми та зовнішні фактори.

Проаналізовано особливості сучасного ринку ноутбуків, виявлено ключові групи стейкхолдерів (споживачі, продавці, виробники, аналітики) та сформульовано їхню зацікавленість у точному прогнозуванні цін. Окреслено головні фактори, що впливають на ціну: технічні характеристики пристрою, бренд, сезонність, валютні коливання, маркетингові впливи тощо.

Проведено огляд класичних та сучасних методів прогнозування, включаючи статистичні підходи (лінійна регресія, часові ряди) та методи машинного навчання (Random Forest, Gradient Boosting, нейронні мережі). Обґрунтовано вибір ансамблевих моделей як найпридатніших для роботи з великими обсягами даних та виявлення складних, нелінійних закономірностей.

Запропоновано концептуальну модель складної системи прогнозування та управління ціноутворенням на ринку ноутбуків, що включає багаторівневу структуру: елементи, підсистеми, систему в цілому. Обґрунтовано критерій оптимізації (максимізація точності прогнозу) та визначено задачу системного аналізу як побудову моделі, що дозволяє підтримувати прийняття рішень на всіх рівнях — від оперативного до стратегічного.

Таким чином, розділ створює концептуальну, методичну та інструментальну базу для подальшої реалізації прогнозної моделі з урахуванням вимог системного аналізу та специфіки задачі.

.

2 ВИБІР ІНСТРУМЕНТАРІЮ ТА РОЗВІДУВАЛЬНИЙ АНАЛІЗ ДАНИХ

2.1 Вибір мови програмування та середовища розробки

Створення інформаційної технології для прогнозування ціни на ноутбуки, що базується на методах машинного навчання та є ключовим завданням даної бакалаврської роботи, вимагає ретельного підходу до вибору інструментарію. Це завдання належить до класу складних систем різної природи, аналіз, моделювання та прогнозування яких є об'єктом вивчення системного аналізу. Отже, вибір адекватних математичних методів та інформаційних технологій є критично важливим для успішної реалізації проєкту.

У сучасному ландшафті розробки інформаційних систем, особливо у сферах, пов'язаних з аналізом даних, машинним навчанням та штучним інтелектом, мова програмування Python здобула визнання як провідний інструмент. Її широке визнання є наслідком поєднання кількох ключових характеристик, що роблять її ідеальною для розв'язання регресійних задач, таких як прогнозування ціни [16,17].

Спектр застосування Python є надзвичайно широким, охоплюючи веб-розробку, автоматизацію, розробку ігор та системне адміністрування. Однак, її справжній потенціал розкривається у сферах, що вимагають інтенсивних обчислень та складного аналізу даних. Саме тут Python є незамінним інструментом для:

– Математичного моделювання та аналізу даних: Завдяки численним бібліотекам, Python дозволяє ефективно реалізовувати статистичні та математичні моделі, що є основою теоретичного змісту предметної області системного аналізу.

– Прогнозування: Можливість роботи з різноманітними алгоритмами машинного навчання, що належать до методів прогнозування, є ключовою для даної роботи.

– Дослідження операцій та оптимізації систем: Здатність до швидкого прототипування та тестування різних підходів робить Python ідеальним для пошуку оптимальних рішень у складних системах.

Філософські засади Python, що відображені у "Дзені Python" Тіма Петерса, сприяють створенню чистого, ефективного та підтримуваного коду. Серед них виділяються принципи, такі як перевага "практичності над бездоганністю", "простого над складним" та "чіткого над заплутаним". Ці постулати ідеально узгоджуються з потребами системного аналізу, де важливі не лише кінцевий результат, а й прозорість, масштабованість та ремонтопридатність розроблених рішень.

Ці принципи сприяли формуванню мови, яка є інтуїтивно зрозумілою, відносно легкою для освоєння та підтримується величезною спільнотою розробників, що забезпечує постійний розвиток та доступність актуальних рішень. Її міжплатформність також є важливою перевагою, дозволяючи розробляти рішення, які можуть бути розгорнуті на різних операційних системах [15-17].

Серед ключових галузей, де Python домінує, варто виділити:

– Машинне навчання та штучний інтелект: Провідна позиція завдяки розвиненим бібліотекам.

– Наука про дані та аналітика: Інструментарій для збору, обробки, аналізу та візуалізації великих обсягів даних.

– Веб-розробка: Створення ефективних серверних частин веб-застосунків.

– Автоматизація процесів: Скриптування рутинних завдань, системне адміністрування.

Важливо зазначити, що універсальної мови програмування, ідеальної для всіх завдань, не існує. Python може мати обмеження у продуктивності для низькорівневих систем, таких як розробка драйверів або операційних систем, через свій інтерпретований характер та динамічну типізацію, які впливають на критерії оцінювання та методи забезпечення якості, надійності,

відмовостійкості, живучості інформаційних систем у таких специфічних галузях. Однак, для більшості прикладних задач, особливо у сфері прогнозування та аналізу даних, переваги Python значно перевищують ці обмеження.

Свою домінуючу позицію Python здобув саме завдяки активному використанню в машинному навчанні та аналізі даних. Мова пропонує широкі можливості для роботи з лінійною алгеброю, обробки сигналів, математичними та статистичними підходами, а також для візуалізації даних. Стабільне зростання позицій Python у світових рейтингах мов програмування [16] є яскравим свідченням його постійного розвитку та зростаючої популярності серед фахівців.

Хоча Python, будучи інтерпретованою мовою, може демонструвати нижчу швидкість виконання порівняно з компільованими мовами (наприклад, C++), цей аспект рідко є критичним для більшості завдань прогнозної аналітики. Сучасні обчислювальні потужності легко компенсують цей недолік. Більш того, Python дозволяє інтегрувати високопродуктивні модулі, написані на C/C++, що є ефективним рішенням для оптимізації ресурсоємних частин коду. Це забезпечує відповідність методикам та технологіям фундаментальних та прикладних наук у контексті підвищення продуктивності [17].

Ключовою перевагою Python є його величезна екосистема бібліотек для машинного навчання та аналізу даних. Ці бібліотеки покривають всі етапи розробки прогнозних моделей: від попередньої обробки та візуалізації даних до реалізації складних регресійних алгоритмів та оцінки їхньої ефективності. Завдяки цьому, Python забезпечує потужний інструментарій для вирішення задач, пов'язаних з аналізом даних та прогнозуванням у контексті ціноутворення на ринку ноутбуків:

– SciPy/NumPy: Основа для чисельних обчислень та роботи з багатовимірними масивами.

– Pandas: Незамінний інструмент для маніпуляцій та аналізу структурованих даних, що дозволяє ефективно працювати з великими наборами даних про характеристики та ціни ноутбуків.

– Matplotlib/Seaborn: Бібліотеки для створення високоякісних візуалізацій, які є критично важливими для аналізу даних та представлення результатів моделювання.

– Scikit-learn: Стандарт де-факто для класичних алгоритмів машинного навчання, що надає широкий спектр регресійних моделей (від лінійної регресії до ансамблевих методів, таких як випадковий ліс та градієнтний бустинг), які ідеально підходять для прогнозування ціни на ноутбуки.

– TensorFlow/PyTorch: Потужні фреймворки для розробки та навчання нейронних мереж, які можуть бути застосовані для виявлення складних нелінійних залежностей у цінових даних, що є частиною математичного моделювання.

Саме ця різноманітність та зрілість бібліотек роблять Python неперевершеним вибором для створення ефективної системи прогнозування ціни на ноутбуки, що відповідає сучасним вимогам до критеріїв оцінювання та методів забезпечення якості, надійності, відмовостійкості, живучості інформаційних систем [15-17].

Ці алгоритми дозволяють прогнозувати ціну ноутбука на основі його технічних характеристик (процесор, обсяг оперативної пам'яті, тип накопичувача, розмір екрану), бренду, року випуску, економічних показників та ринкових тенденцій.

Для забезпечення ефективного процесу розробки, тестування та валідації прогнозної моделі ціни на ноутбуки, було обрано онлайн-платформу Kaggle. Цей вибір обґрунтований з позицій системної інтеграції та управління ІТ-проєктами на етапі дослідження.

Kaggle пропонує вбудований інтерфейс Jupyter Notebook, який є ідеальним для інтерактивного та ітеративного процесу аналізу даних та розробки моделей. Можливість виконувати код Python безпосередньо у браузері, поєднувати його з текстом, формулами та візуалізаціями, значно прискорює етап прототипування та дослідницької діяльності. Це сприяє швидкому виявленню та усуненню помилок, а також дозволяє більш глибоко зануритися у характер даних [18,19].

Важливою перевагою Kaggle є її спільнота та доступ до величезної колекції наборів даних. Ця "бібліотека даних" включає численні публічні датасети, зокрема ті, що стосуються характеристик електронних пристроїв та їхніх цін. Наявність готових до використання, часто вже очищених та верифікованих даних, дозволяє уникнути значних витрат часу на етапі збору та попередньої обробки інформації, що є частиною оптимізації систем та процесів. Це дозволяє зосередитися на ключових аспектах – моделюванні, прогнозуванні та прийнятті рішень.

Крім того, Kaggle надає доступ до масштабованих обчислювальних ресурсів, включаючи графічні процесори (GPU). Це є критично важливим для тренування складних моделей машинного навчання, які можуть вимагати значних обчислювальних потужностей, особливо при роботі з великими наборами даних. Можливість швидко перемикатися між різними апаратними конфігураціями забезпечує гнучкість та ефективність у проведенні експериментів. Таким чином, Kaggle виступає як комплексний інструментальний засіб для проведення даного системного аналізу [19,20].

2.2 Розвідувальний аналіз даних

Для проведення аналізу було обрано набір даних у Kaggle, що має назву « Laptop sales price prediction 2024». Даний датасет має відкритий доступ для загального використання на платформі [21]. Цей набір даних містить інформацію про різні ноутбуки, включаючи характеристики та ціни, підібрану для прогнозування цін продажу. Завдяки функціям, що варіюються від деталей процесора до характеристик дисплея, цей набір даних слугує цінним ресурсом для аналізу тенденцій та прогнозування цін на ноутбуки. Всього в датасеті 28 ознак та 1020 записів (рис. 2.1).

	Name	Brand	Price	Rating	Processor_brand	Processor_name	Processor_variant	Processor_gen	Core_per_processor
0	HP Victus 15-fb0157AX Gaming Laptop (AMD Ryzen...	HP	50399	4.30	AMD	AMD Ryzen 5	5600H	5.0	6.0
1	Lenovo V15 G4 82YU00W7IN Laptop (AMD Ryzen 3 ...	Lenovo	26690	4.45	AMD	AMD Ryzen 3	7320U	7.0	4.0
2	HP 15s-fq5007TU Laptop (12th Gen Core i3/ 8GB/...	HP	37012	4.65	Intel	Intel Core i3	1215U	12.0	6.0
3	Samsung Galaxy Book2 Pro 13 Laptop (12th Gen C...	Samsung	69990	4.75	Intel	Intel Core i5	1240P	12.0	12.0
4	Tecno Megabook T1 Laptop (11th Gen Core i3/ 8G...	Tecno	23990	4.25	Intel	Intel Core i3	1115G4	11.0	2.0
...	...	...	...	...	...	...	...	...	...

...	Graphics_name	Graphics_brand	Graphics_GB	Graphics_integreted	Display_size_inches	Horizontal_pixel	Vertical_pixel	ppi	Touch_screen
...	AMD Radeon RX 6500M	AMD	4.0	False	15.6	1920	1080	141.21	True
...	AMD Radeon Graphics	AMD	NaN	False	15.6	1920	1080	141.21	False
...	Intel UHD Graphics	Intel	NaN	False	15.6	1920	1080	141.21	False
...	Intel Iris Xe Graphics	Intel	NaN	False	13.3	1080	1920	165.63	False
...	Intel UHD Graphics	Intel	NaN	False	15.6	1920	1080	141.21	False
...	...	...	...	...	...	...	...	...	...

Рисунок 2.1 – Приклад ознак у наборі даних

Розглянемо повний список ознак, які є у нашому наборі даних, зробимо їхній опис. Отож даний датасет включає у себе такі колонки:

1. Name: Назва моделі ноутбука.
2. Brand: Бренд ноутбука.
3. Price: Ціна ноутбука.
4. Rating: Рейтинг ноутбука.
5. Processor_brand: Бренд процесора ноутбука.
6. Processor_name: Назва процесора ноутбука.
7. Processor_variant: Варіант процесора ноутбука.
8. Processor_gen: Покоління процесора ноутбука.
9. Core_per_processor: Кількість ядер на процесор.
10. Total_processor: Загальна кількість процесорів.
11. Execution_units: Кількість виконавчих блоків.

12. Low_Power_Cores: Кількість енергоефективних ядер.
13. Energy_Efficient_Units: Вказує, чи має ноутбук енергоефективні блоки.
14. Threads: Кількість потоків.
15. RAM_GB: Обсяг оперативної пам’яті в гігабайтах.
16. RAM_type: Тип оперативної пам’яті.
17. Storage_capacity_GB: Обсяг накопичувача в гігабайтах.
18. Storage_type: Тип накопичувача.
19. Graphics_name: Назва графічного адаптера ноутбука.
20. Graphics_brand: Бренд графічного адаптера ноутбука.
21. Graphics_GB: Обсяг відеопам’яті в гігабайтах.
22. Graphics_integrated: Вказує, чи має ноутбук інтегровану графіку.
23. Display_size_inches: Розмір екрана ноутбука в дюймах.
24. Horizontal_pixel: Кількість горизонтальних пікселів.
25. Vertical_pixel: Кількість вертикальних пікселів.
26. ppi: Кількість пікселів на дюйм.
27. Touch_screen: Вказує, чи має ноутбук сенсорний екран.
28. Operating_system: Операційна система ноутбука.

Розпочнемо з перших кроків – імпорту бібліотек, завантаження даних та початкового огляду. Визначимо наявність пропущених значень та типи даних у стовбцях (рис. 2.2).

```
Інформація про датасет (df.info()):
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1020 entries, 0 to 1019
Data columns (total 29 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Unnamed: 0              1020 non-null   int64
 1   Name                    1020 non-null   object
 2   Brand                   1020 non-null   object
 3   Price                   1020 non-null   int64
 4   Rating                  1020 non-null   float64
 5   Processor_brand         1020 non-null   object
 6   Processor_name          1020 non-null   object
 7   Processor_variant       996 non-null    object
 8   Processor_gen           891 non-null    float64
 9   Core_per_processor      1008 non-null   float64
 10  Total_processor         573 non-null    float64
 11  Execution_units         573 non-null    float64
 12  Low_Power_Cores         1020 non-null   float64
 13  Energy_Efficient_Units  1020 non-null   int64
 14  Threads                 972 non-null    float64
 15  RAM_GB                  1020 non-null   int64
 16  RAM_type                998 non-null    object
 17  Storage_capacity_GB     1020 non-null   int64
 18  Storage_type            1020 non-null   object
 19  Graphics_name           1018 non-null   object
 20  Graphics_brand          1018 non-null   object
 21  Graphics_GB             368 non-null    float64
 22  Graphics_integreted     1018 non-null   object
 23  Display_size_inches     1020 non-null   float64
 24  Horizontal_pixel        1020 non-null   int64
 25  Vertical_pixel          1020 non-null   int64
 26  ppi                     1020 non-null   float64
 27  Touch_screen            1020 non-null   bool
 28  Operating_system        1020 non-null   object
dtypes: bool(1), float64(10), int64(7), object(11)
memory usage: 224.2+ KB
```

Рисунок 2.2 – Результат виконання функції info()

З отриманої інформації про пропущені значення, бачимо наступне:

– Значна кількість пропусків: Total_processor (447), Execution_units (447), Graphics_GB (652). Ці стовпці мають майже половину або більше пропущених значень.

– Помірна кількість пропусків: Processor_gen (129), Threads (48).

– Невелика кількість пропусків: Processor_variant (24), Core_per_processor (12), RAM_type (22), Graphics_name (2), Graphics_brand (2), Graphics_integreted (2).

Це ключова інформація для етапу підготовки даних. Ми будемо ретельно підходити до кожного стовпця з пропусками, вирішуючи, чи варто їх заповнювати, чи, можливо, видаляти сам стовпець, якщо він несе мало корисної інформації.

Далі необхідно дослідити набір на наявність аномалій, або малоймовірних значень, які в майбутньому можуть негативно вплинути при навчанні моделей. Використаємо функцію describe(), щоб визначити описові статистики числових ознак (рис. 2.3).

Опис числових ознак:

	Unnamed: 0	Price	Rating	Processor_gen	Core_per_processor	Total_processor	Execution_units	Low_Power_Cores
count	1020.000000	1020.000000	1020.000000	891.000000	1008.000000	573.000000	573.000000	1020.000000
mean	509.500000	82063.474510	4.373676	10.450056	8.572421	3.926702	6.998255	0.086275
std	294.592939	66502.150607	0.233295	2.966579	4.375012	1.954429	2.680217	0.406531
min	0.000000	8000.000000	3.950000	1.000000	2.000000	1.000000	4.000000	0.000000
25%	254.750000	43990.000000	4.200000	7.000000	6.000000	2.000000	4.000000	0.000000
50%	509.500000	63689.500000	4.350000	12.000000	8.000000	4.000000	8.000000	0.000000
75%	764.250000	94990.000000	4.550000	13.000000	10.000000	6.000000	8.000000	0.000000
max	1019.000000	599990.000000	4.750000	14.000000	24.000000	12.000000	16.000000	2.000000

Energy_Efficient_Units	Threads	RAM_GB	Storage_capacity_GB	Graphics_GB	Display_size_inches	Horizontal_pixel	Vertical_pixel	ppi
1020.000000	972.000000	1020.000000	1020.000000	368.000000	1020.000000	1020.000000	1020.000000	1020.000000
0.043137	12.817901	13.992157	627.733333	5.940217	15.163775	2035.512745	1214.019608	157.178265
0.203266	5.677459	7.189564	316.911679	2.667130	1.001537	409.209289	306.863086	33.585713
0.000000	2.000000	2.000000	32.000000	2.000000	11.600000	1080.000000	768.000000	100.450000
0.000000	8.000000	8.000000	512.000000	4.000000	14.000000	1920.000000	1080.000000	141.210000
0.000000	12.000000	16.000000	512.000000	6.000000	15.600000	1920.000000	1080.000000	141.210000
0.000000	16.000000	16.000000	512.000000	8.000000	15.600000	1920.000000	1200.000000	161.730000
1.000000	32.000000	64.000000	4000.000000	16.000000	18.000000	3840.000000	2560.000000	337.930000

Рисунок 2.3 – Результат виконання функції describe()

Тепер візуалізуємо розподіли ознак, їхні взаємозв'язки, кореляції та виявимо потенційні викиди.

Побудуємо розподіл ціни на ноутбуки (Price) (рис. 2.4).

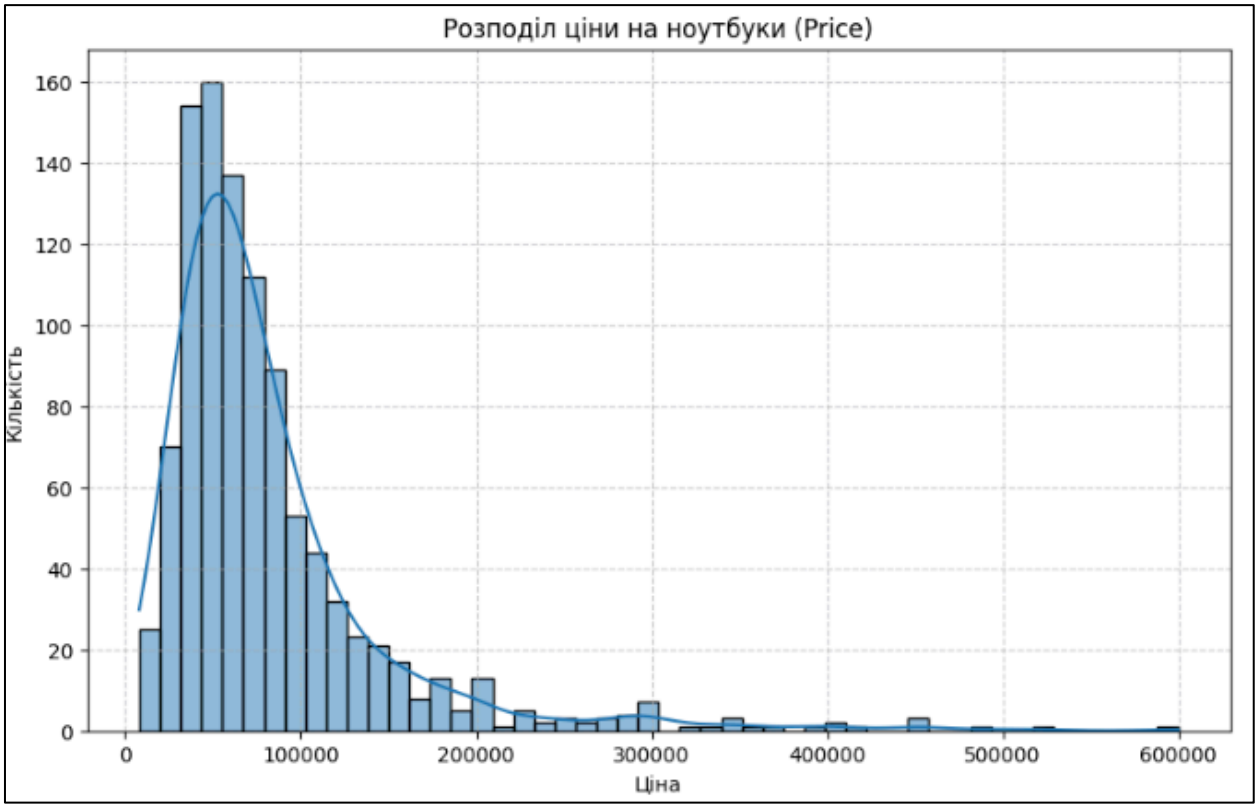

Рисунок 2.4 – Розподіл ціни на ноутбуки (Price)

Графік показує сильно скошений розподіл вліво (left-skewed). Це означає, що більшість ноутбуків мають порівняно невисоку ціну, тоді як невелика кількість моделей є значно дорожчими, утворюючи "довгий хвіст" у правій частині розподілу.

Такий розподіл є типовим для цінових даних. Моделі лінійної регресії часто працюють краще з нормально розподіленими даними. Сильна скошеність цільової змінної може призвести до того, що модель буде недооцінювати високі ціни та переоцінювати низькі.

Такий розподіл вимагає застосування трансформацій даних (наприклад, логарифмічної) для покращення ефективності регресійних моделей (рис. 2.5).

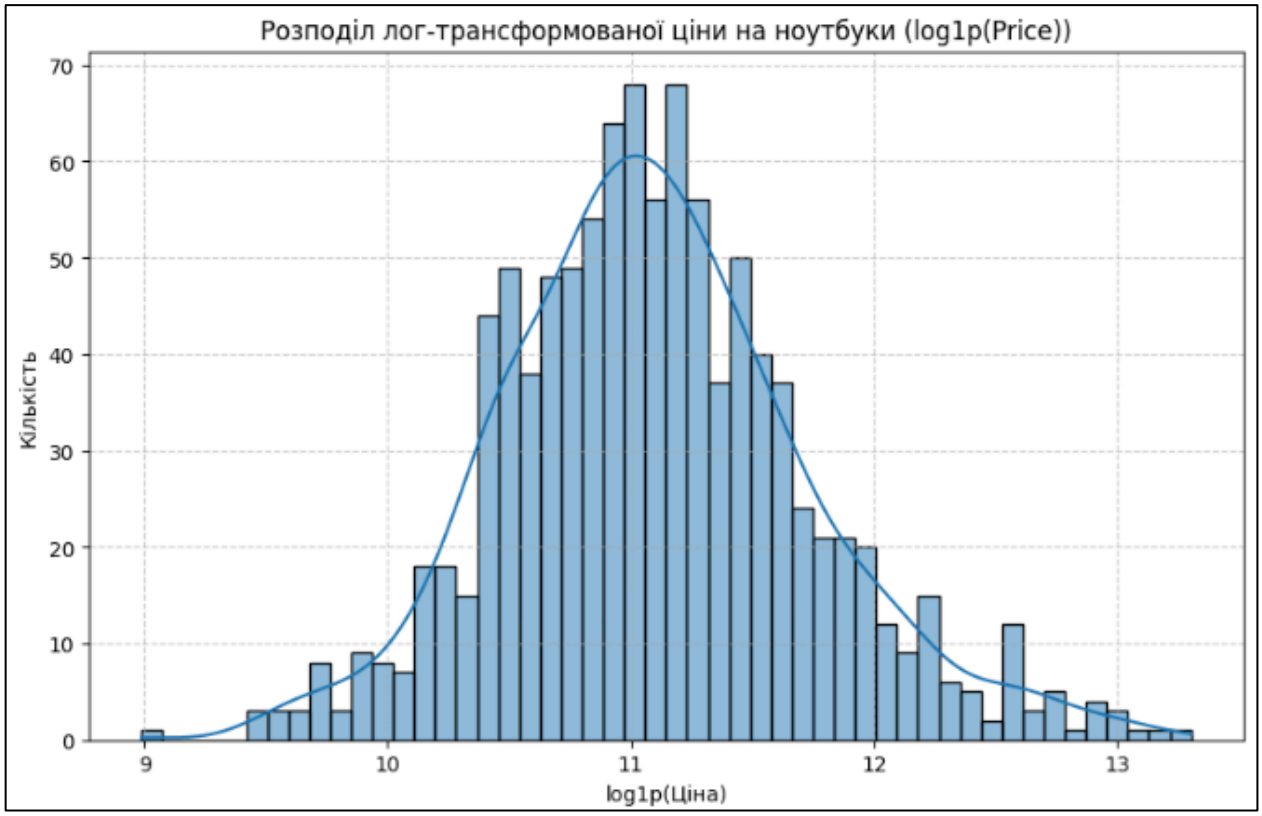

Рисунок 2.5 – Розподіл лог-трансформованої ціни на ноутбуки (log1p(Price)

Після застосування логарифмічної трансформації (np.log1p, яка обробляє log(1+x) для уникнення проблем з нулями), розподіл ціни став значно ближчим до нормального (колоколоподібного).

Це є дуже позитивним результатом. Моделі регресії, які базуються на припущеннях про нормальний розподіл помилок, будуть працювати значно ефективніше з такою трансформованою цільовою змінною, оскільки дозволяє їм краще захоплювати лінійні залежності та підвищує точність прогнозування. Ми будемо прогнозувати логарифм ціни, а потім зворотним перетворенням отримувати прогнозовану ціну.

Візуалізуємо розподіл решти ознак використовуючи гістограми (рис. 2.6).

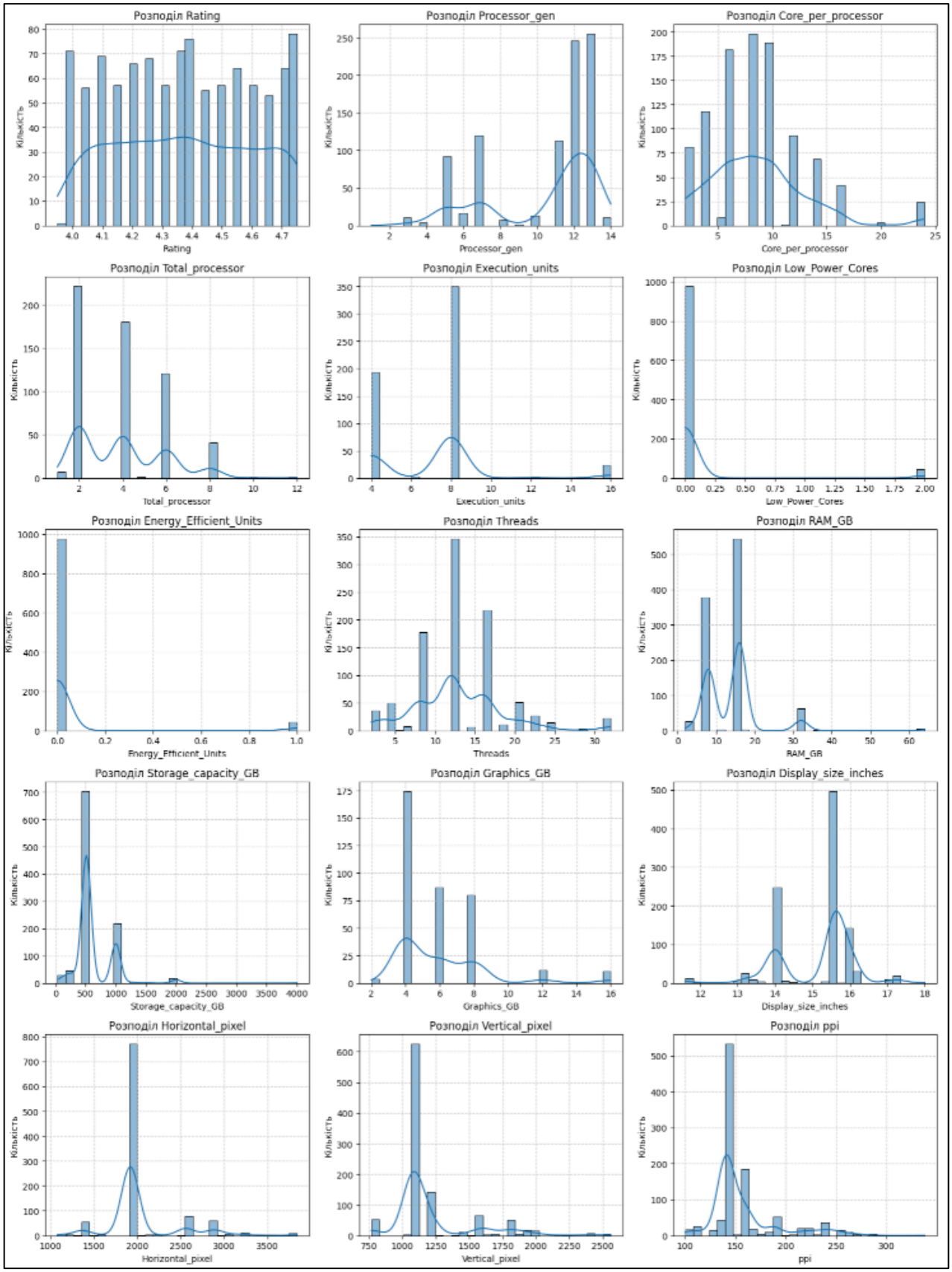

Рисунок 2.6 – Гістограми розподілу ознак

Візуалізуємо розподіл ознак використовуючи боксплоти (рис. 2.7).

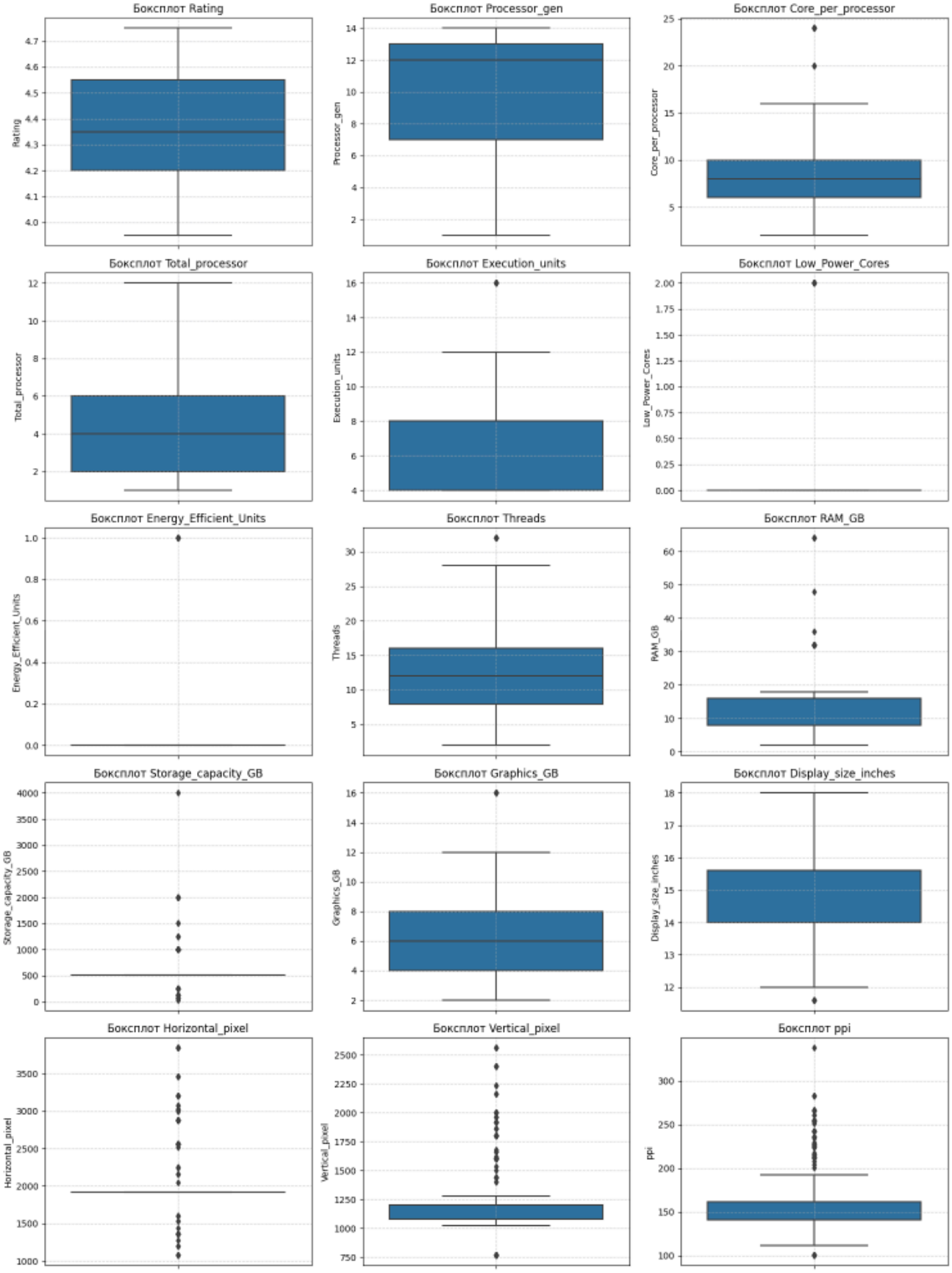

Рисунок 2.7 – Боксплоти розподілу ознак

Аналіз числових ознак виявив значну варіативність у характеристиках ноутбуків. Деякі ознаки, такі як Core_per_processor, Threads, RAM_GB, Storage_capacity_GB та Graphics_GB, мають "викиди" у верхній частині розподілу, що відображає наявність високопродуктивних конфігурацій (наприклад, більше ядер, більший об'єм RAM/ROM, потужніші відеокарти). Ці "викиди" є природними для ринку ноутбуків і, ймовірно, несуть важливу інформацію про ціноутворення, тому їх обробка потребує обережного підходу.

Особливу увагу слід приділити ознакам з великою кількістю пропущених значень, а саме Total_processor, Execution_units та Graphics_GB. Їхня інформативність без належної стратегії заповнення пропусків може бути низькою.

Тепер перейдемо до аналізу категоріальних ознак та кореляційних матриць.

На рисунку 2.8 розподіл деяких категоріальних ознак.

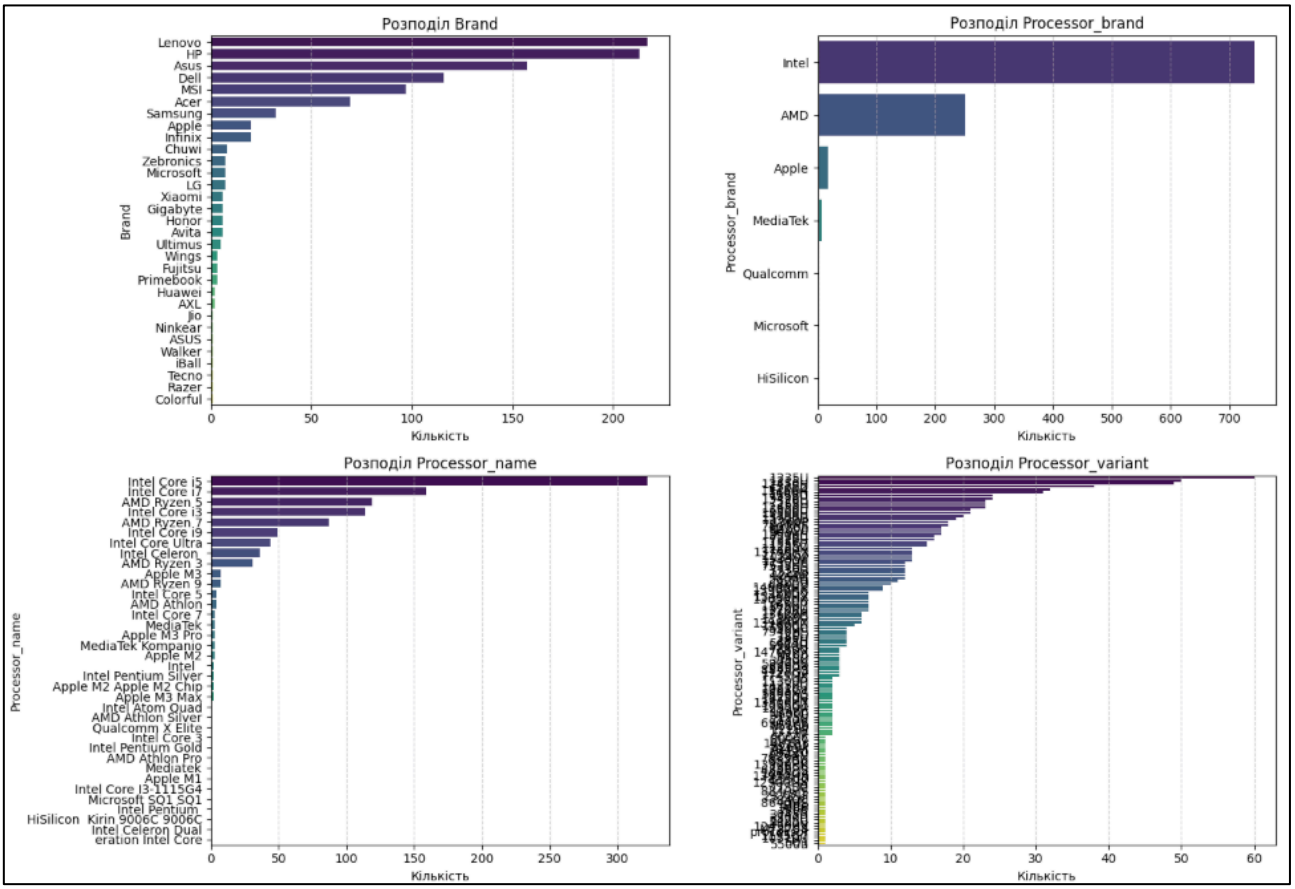

Рисунок 2.8 – Розподіл категоріальних ознак

Далі розглянемо залежність ціни від категоріальних ознак за допомогою діаграм Box Plot.

Box Plot (ящик з вусами). Кожен «ящик» показує медіану (центральна лінія), перший (Q1) та третій (Q3) квартилі, а «вуса» позначають діапазон даних, що не є викидами. Точки за межами «вусів» є викидами.

На рисунку 2.9 наведено залежність ціни від ознак “Brand” та “Processor_brand”.

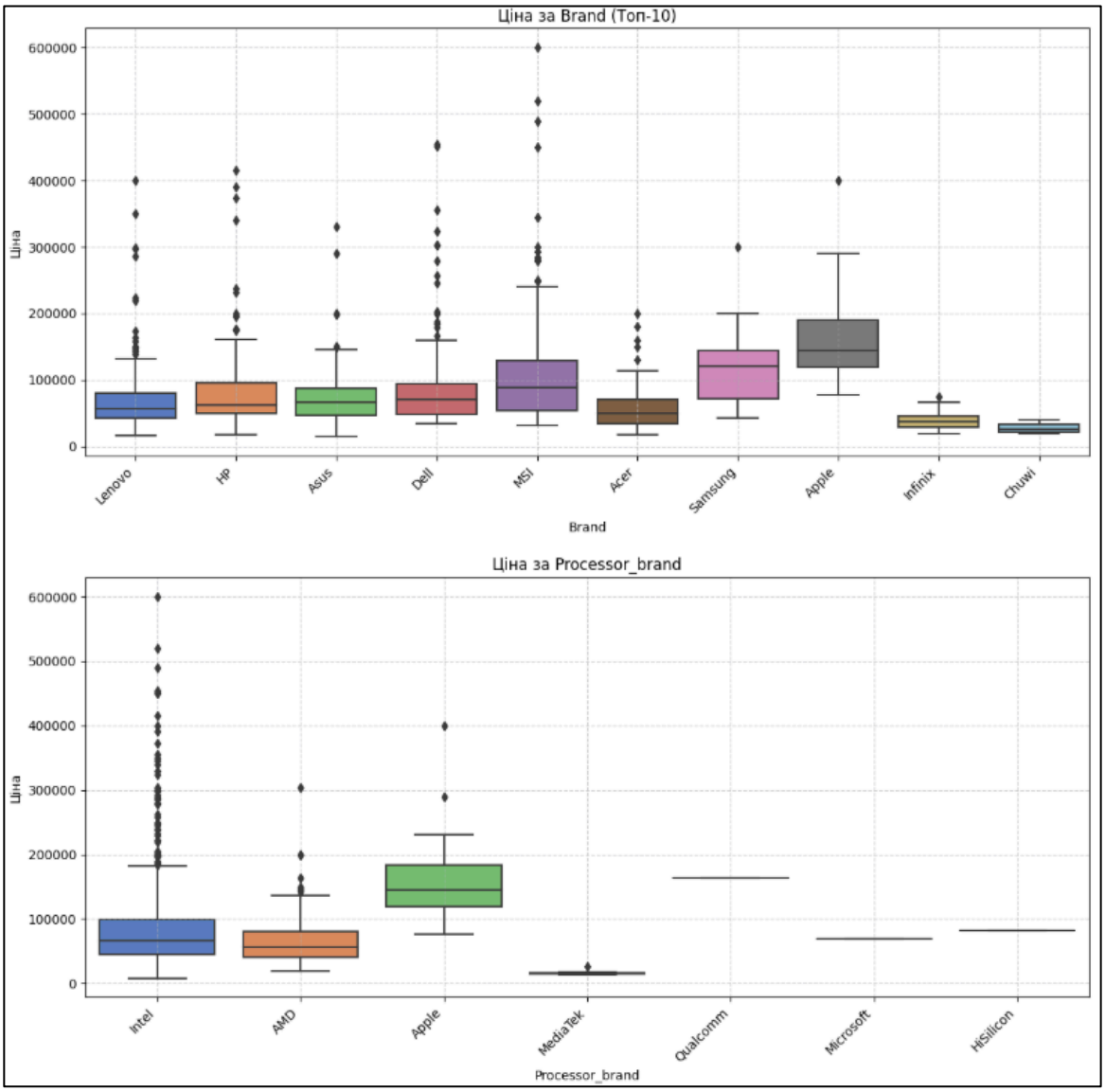

Рисунок 2.9 – Залежність ціни від ознак “Brand” та “Processor_brand”

Ознака "Brand" – представлено 10 провідних брендів ноутбуків.

Розподіл цін:

– Apple: Цей бренд чітко виділяється найвищими цінами. Його "ящик" (діапазон між Q1 і Q3) розташований значно вище, ніж у інших брендів, і його медіана також є найвищою. Це вказує на те, що ноутбуки Apple, як правило, дорожчі, ніж у конкурентів.

– Samsung: Також має відносно високу медіану та діапазон цін, що ставить його в когорту дорожчих брендів, хоча і нижче Apple.

– MSI: Демонструє ширший розмах цін, з медіаною вище більшості, що може вказувати на широкий асортимент продуктів від середнього до високого цінового сегменту (ігрові ноутбуки, тощо).

– Lenovo, HP, Asus, Dell, Acer: Ці бренди демонструють схожі, відносно низькі медіани та квартильні діапазони, що свідчить про їхню присутність у більш масовому та доступному ціновому сегменті.

– Infinix, Chuwi: Ці бренди мають найнижчі ціни та найменші розмахи, що, ймовірно, вказує на їхню спеціалізацію на бюджетних моделях.

Викиди: Для багатьох брендів (наприклад, Lenovo, HP, Asus, Dell, MSI) спостерігаються значні викиди у верхньому ціновому діапазоні, що свідчить про наявність окремих дорогих моделей (наприклад, преміум-серії, ігрові чи професійні робочі станції) у цих виробників. Apple також має викиди, але вони починаються з вищого базового рівня.

Ознака "Processor_brand" – представлено різні виробники процесорів.

Розподіл цін:

– Apple: Процесори Apple (ймовірно, M-серія) асоціюються з найвищими цінами на ноутбуки, що відображає преміум-позиціонування продуктів Apple.

– Intel: Має широкий діапазон цін з відносно високою медіаною та значною кількістю викидів у верхньому ціновому діапазоні. Це очікувано, оскільки Intel є домінуючим постачальником процесорів для більшості брендів, що покривають широкий спектр цінових категорій.

– AMD: Демонструє дещо нижчу медіану та менший діапазон цін порівняно з Intel, але все ще покриває значний сегмент ринку.

– Mediatek, Qualcomm, Microsoft, HiSilicon: Ці бренди процесорів асоціюються з значно нижчими цінами та меншими розмахами. Це може вказувати на їх використання переважно в бюджетних ноутбуках, Chromebook, або пристроях, які не потребують високої продуктивності.

Таким чином, ці графіки чітко показують, що як виробник ноутбука (бренд), так і бренд встановленого процесора є значущими факторами, що впливають на ціну пристрою, причому Apple є лідером у сегменті високих цін за обома ознаками.

На рисунку 2.10 наведено залежність ціни від ознак “Processor_name” та “Processor_variant”.

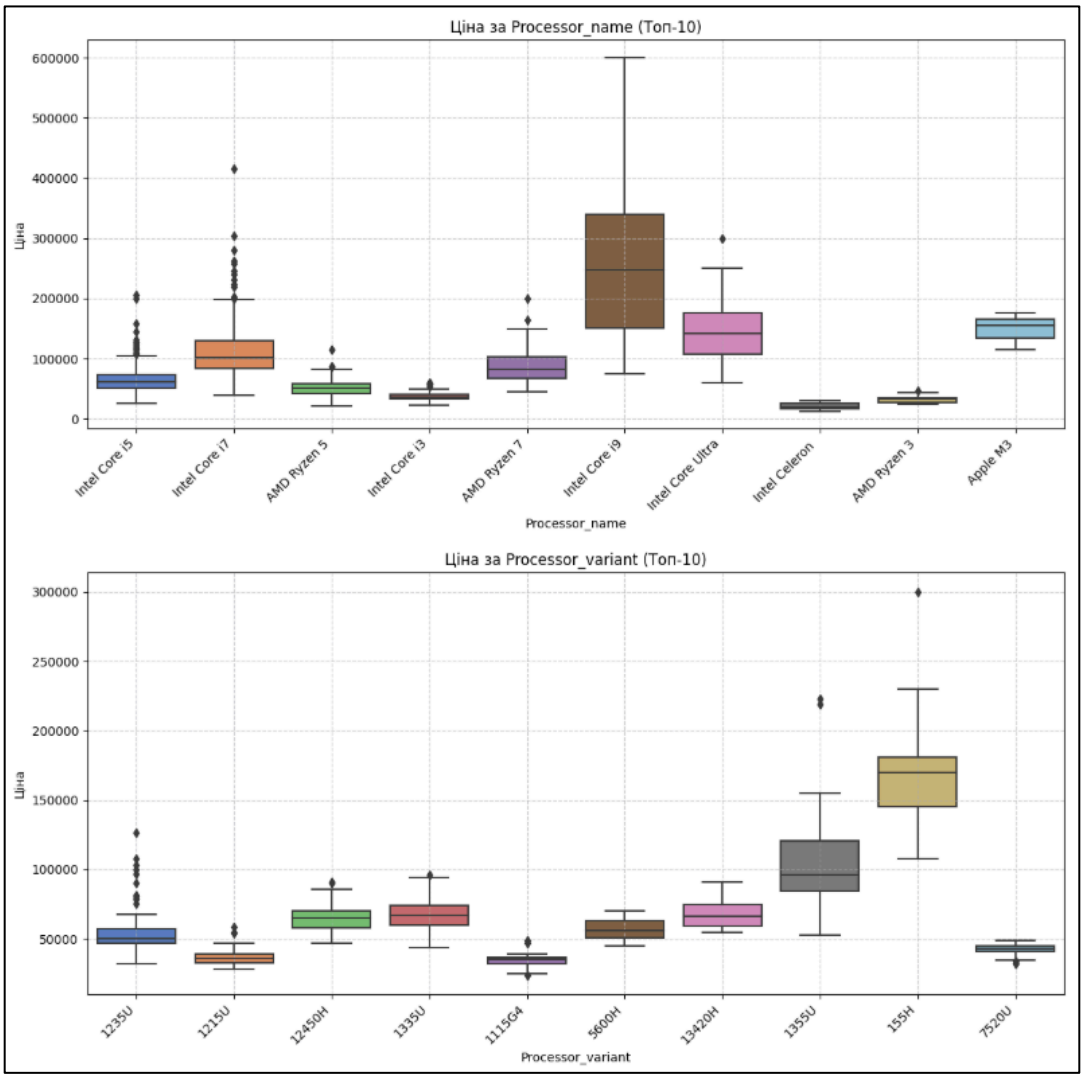

Рисунок 2.10 – Залежність ціни від ознак “Processor_name” та “Processor_variant”

З рисунку 2.10 можна зробити такі висновки:

– "Processor_name" є сильним предиктором ціни. Існує чітка ієрархія цін за назвою процесора. Процесори вищого класу (Intel Core i9, Intel Core i7, AMD Ryzen 7) значно дорожчі, ніж процесори середнього (Intel Core i5, AMD Ryzen 5) та початкового рівня (Intel Core i3, AMD Ryzen 3, Intel Celeron, AMD A6).

– "Processor_variant" також впливає на ціну, демонструючи диференціацію всередині серій. Навіть у межах одного "Processor_name", конкретний варіант (модель) процесора має значний вплив на ціну. Варіанти, що закінчуються на "H" (наприклад, 135H, 155H, 13700H), як правило, дорожчі, що може вказувати на їхню високу продуктивність (наприклад, для геймінгу або професійних завдань). Варіанти з "U" (наприклад, 1235U, 1335U) частіше асоціюються з середнім або бюджетним сегментом (енергоефективні процесори для ультрабуків).

– Викиди вказують на складність ціноутворення. Значна кількість викидів на обох графіках підкреслює, що ціна ноутбука залежить не лише від назви або варіанту процесора, а й від багатьох інших факторів (бренд ноутбука, обсяг ОЗП/SSD, наявність дискретної відеокарти, якість дисплея, дизайн, преміум-матеріали тощо). Навіть з менш потужним процесором, ноутбук може бути дорогим завдяки іншим характеристикам.

Загалом, обидві ознаки – "Processor_name" та "Processor_variant" – є важливими факторами, що визначають ціну ноутбука, відображаючи різні рівні продуктивності та позиціонування на ринку.

На рисунку 2.11 наведено залежність ціни від ознак “RAM_type” та “Storage_ type”.

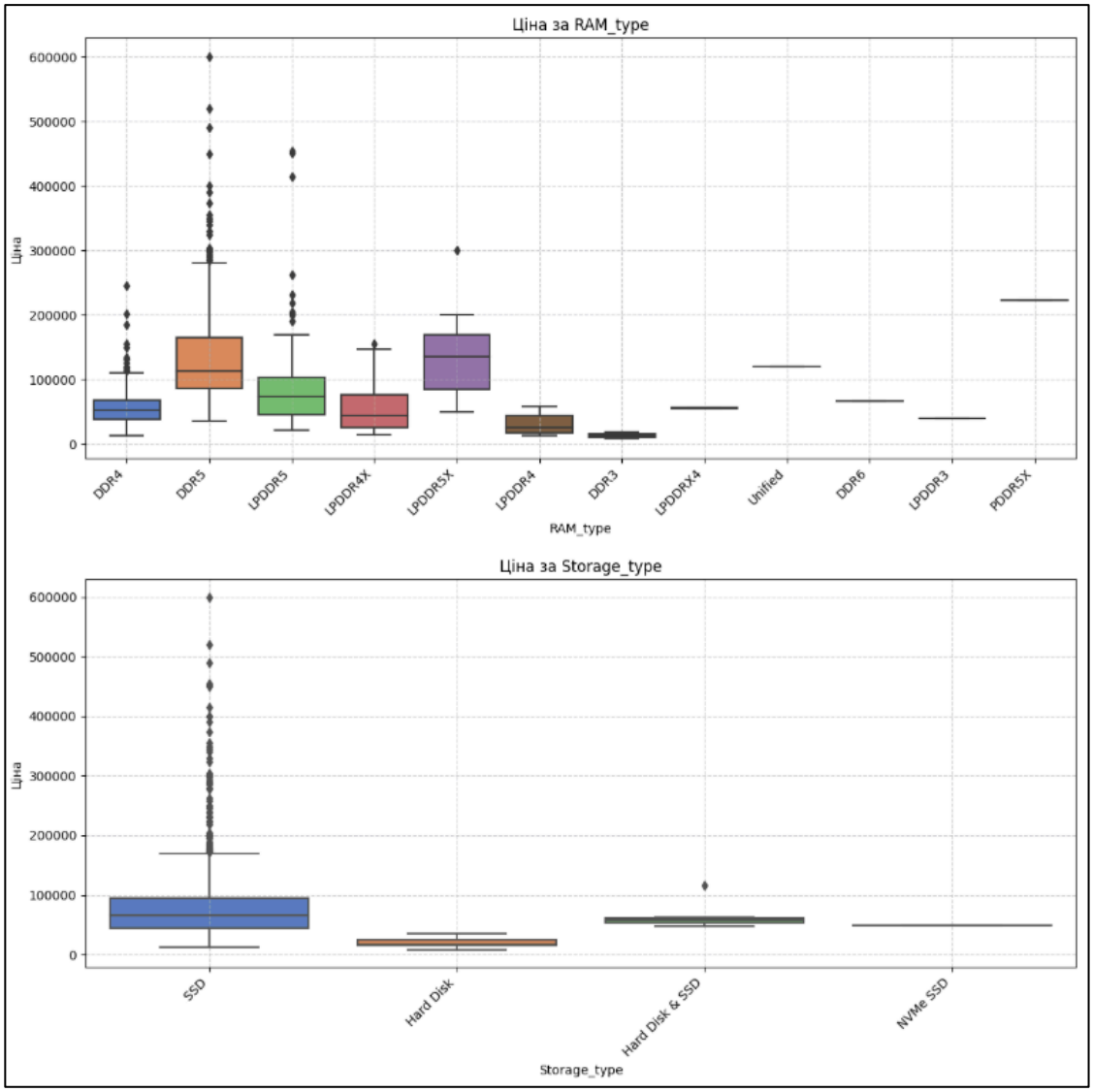

Рисунок 2.11 – Залежність ціни від ознак “RAM_type” та “Storage_ type”

З рисунку 2.11 можна зробити такі висновки:

– Тип RAM є важливим фактором ціни. Новіші та продуктивніші типи оперативної пам'яті, такі як LPDDR5X, LPDDR5 та GDDR6, значно підвищують вартість ноутбука, оскільки вони інтегруються у преміальні та високопродуктивні моделі. Старіші стандарти, як DDR3 та LPDDR3, чітко корелюють з бюджетними пристроями.

– SSD є стандартом на всіх рівнях, але особливо для дорогих ноутбуків. SSD є найпоширенішим типом накопичувача та охоплює найширший діапазон цін, від бюджетних до найдорожчих. Проте, наявність великої кількості викидів

у верхньому ціновому діапазоні для SSD підкреслює, що швидкі та місткі SSD є обов'язковим компонентом у преміум-сегменті.

– HDD асоціюється з бюджетним сегментом. Традиційні жорсткі диски (HDD) використовуються переважно у найдешевших моделях ноутбуків, а гібридні рішення займають проміжну позицію.

Ці візуалізації підтверджують, що вибір ключових компонентів, таких як тип оперативної пам'яті та тип накопичувача, має прямий і значний вплив на кінцеву ціну ноутбука.

На рисунку 2.12 наведено залежність ціни від ознак “Graphics_name” та “Graphics_brand”.

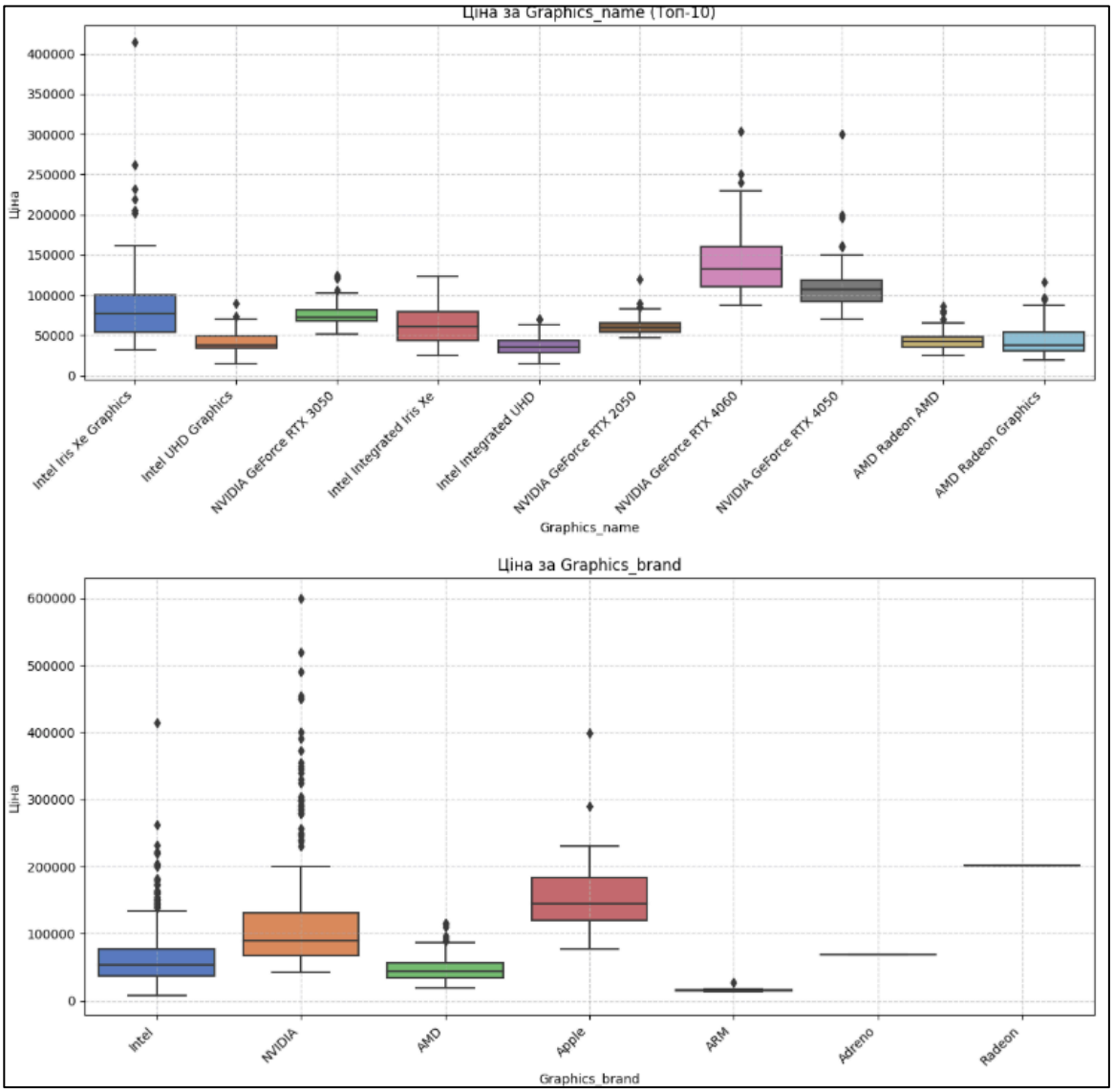

Рисунок 2.12 – Залежність ціни від ознак “Graphics_name” та “Graphics_brand”

З рисунку 2.12 можна зробити такі висновки:

– Тип графічного процесора є критичним фактором ціни: Дискретні відеокарти, особливо нові покоління NVIDIA (RTX серія), значно підвищують вартість ноутбука, оскільки вони призначені для ігор, професійних завдань та інших ресурсомістких застосунків. Інтегровані рішення Intel, як правило, відповідають бюджетним та офісним моделям.

– NVIDIA є лідером у високому ціновому сегменті: Ноутбуки з графічними процесорами NVIDIA, як правило, є найдорожчими, що відображає її домінування на ринку високопродуктивних GPU.

– Intel домінує в бюджетному сегменті інтегрованої графіки: Ноутбуки з інтегрованою графікою Intel знаходяться у нижчому та середньому ціновому діапазоні, що є очікуваним.

– Apple та AMD займають проміжні ніші: Apple з її власними інтегрованими графічними рішеннями знаходиться у високому ціновому сегменті. AMD займає конкурентну позицію у середньому ціновому діапазоні, пропонуючи як інтегровані, так і дискретні рішення.

Загалом, ці візуалізації чітко демонструють, що бренд та назва графічного процесора є надзвичайно важливими факторами, що визначають ціну ноутбука, причому NVIDIA та Apple лідирують у сегменті високих цін, а Intel та інші – у більш доступних категоріях.

На рисунку 2.13 наведено залежність ціни від ознак “Graphics_integrated” та “Operating_system”.

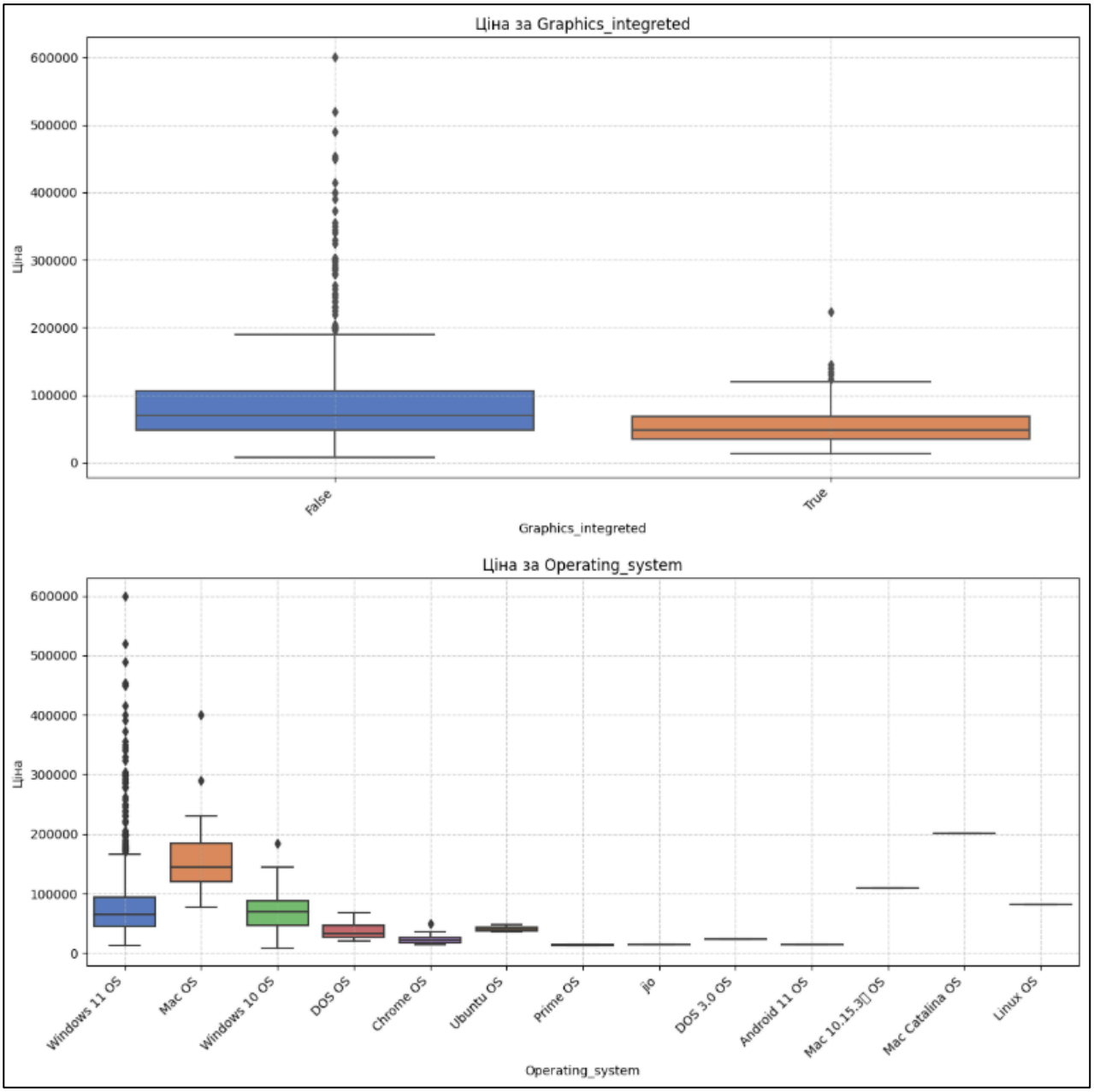

Рисунок 2.13 – Залежність ціни від ознак “Graphics_integrated” та “Operating_system”

З рисунку 2.13 можна зробити такі висновки:

– Наявність дискретної графіки значно підвищує ціну. Ноутбуки з дискретною графікою (False) є значно дорожчими та мають ширший ціновий діапазон порівняно з ноутбуками, що мають лише інтегровану графіку (True). Це ключовий фактор, що впливає на ціну.

– Операційна система є сильним індикатором цінового сегмента.

1) Mac OS чітко корелює з найвищим ціновим сегментом, що є очікуваним для продукції Apple.
2) Windows OS (особливо Windows 11) охоплює найширший ціновий спектр, від бюджетних до преміальних.
3) Інші операційні системи (DOS, Chrome OS, Linux, Android, Free OS, Jo) асоціюються з бюджетними пристроями, що часто не потребують ліцензійної ОС або мають дуже специфічне призначення.

Таким чином, ці візуалізації підкреслюють, що як тип графічної системи (інтегрована чи дискретна), так і встановлена операційна система є важливими факторами, що визначають ціну ноутбука, відображаючи його функціональність, продуктивність та ринкове позиціонування.

Аналіз категоріальних ознак показав, що Brand, Processor_brand, Processor_name, RAM_type, Storage_type, Graphics_brand, Graphics_integreted та Operating_system є ключовими факторами, що впливають на ціну. Зокрема, бренди Apple, процесори Intel Core i9/Ultra, графічні процесори NVIDIA RTX, NVMe SSD та Mac OS асоціюються з вищим ціновим сегментом. Висока кардинальність деяких текстових ознак, таких як Processor_variant та Graphics_name, вимагатиме ефективних методів кодування для уникнення проблем з розмірністю.

Розглянемо теплову карту кореляційної матриці (рис.2.14).

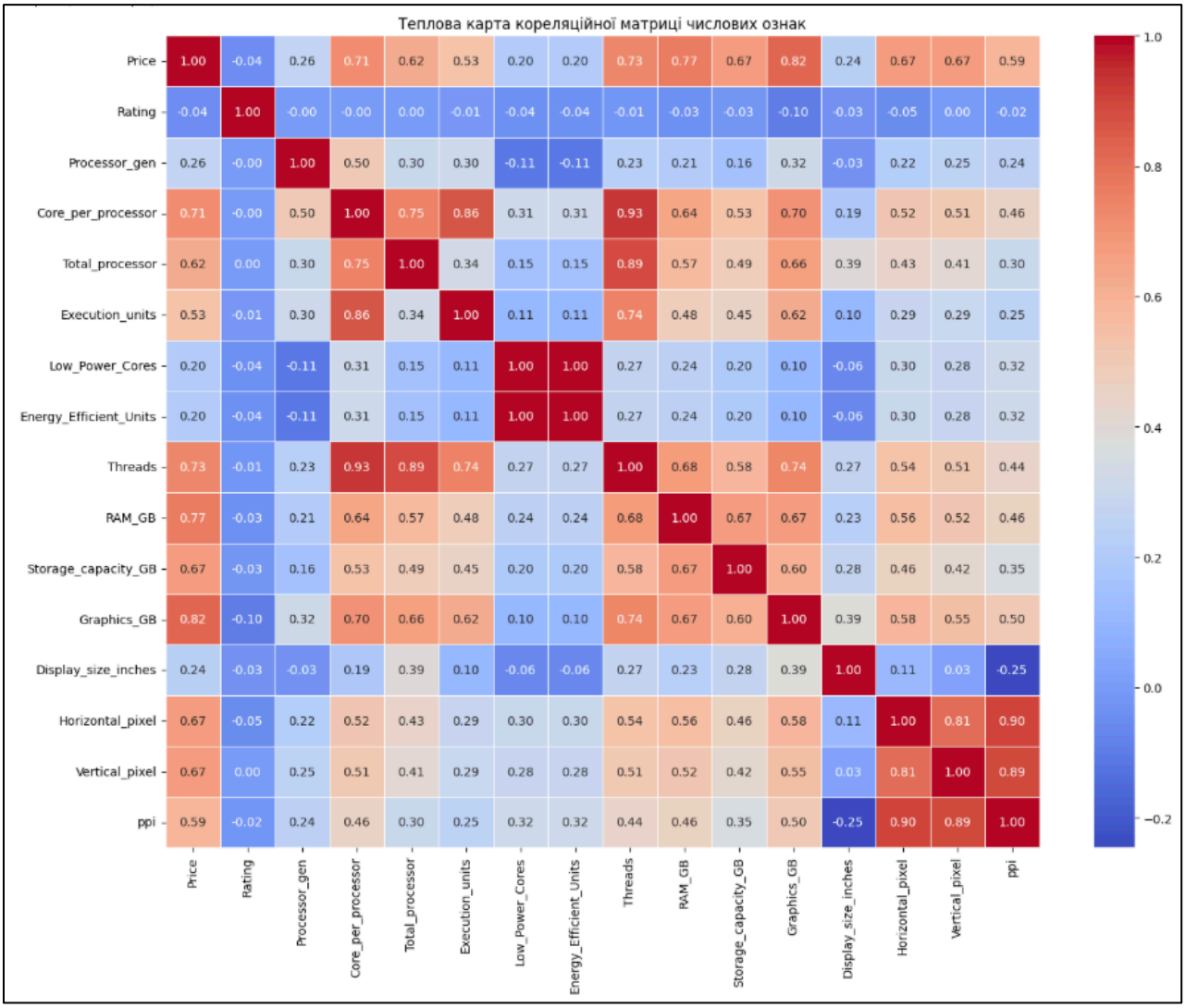

Рисунок 2.14 – Теплова карта кореляційної матриці

Аналіз кореляційної матриці виявив сильні позитивні кореляції ціни ноутбука з ключовими технічними характеристиками, такими як обсяг оперативної пам'яті (RAM_GB), обсяг сховища (Storage_capacity_GB), кількість ядер процесора (Core_per_processor), кількість потоків (Threads), а також параметрами дисплея (ppi, Display_size_inches). Це підтверджує інтуїтивне розуміння впливу потужності та якості компонентів на ціну. Висока кореляція між ознаками, що описують роздільну здатність екрану (Horizontal_pixel, Vertical_pixel, ppi), вказує на потенційну мультиколінеарність, що буде враховано на етапі підготовки даних. Це не є проблемою для моделей на основі дерев (Random Forest, Gradient Boosting), але може вплинути на лінійні моделі (Linear Regression, Ridge, Lasso), зробивши їх коефіцієнти менш стабільними та

інтерпретованими. Можливо, доведеться розглянути видалення однієї з цих сильно корельованих ознак або їх об'єднання [21].

2.3 Висновки

У другому розділі було обґрунтовано вибір інструментарію та проведено всебічний аналіз даних, що є критично важливим етапом у проведенні системного аналізу та прогнозування ціни на ноутбуки.

Визначено, що мова програмування Python є оптимальним вибором для реалізації поставленого завдання завдяки її універсальності, багатому арсеналу бібліотек для машинного навчання та аналізу даних, широкій сумісності та активній спільноті розробників.

Рекомендовано використання хмарних ноутбуків, зокрема Kaggle Notebooks, через їхній безкоштовний доступ до обчислювальних ресурсів (GPU), ефективність робочого процесу, готовність до застосування та можливості для спільної розробки та публікації.

Проведений аналіз залежності ціни на ноутбуки від різних ознак виявив значний вплив як технічних характеристик, так і брендових та програмних аспектів.

Аналіз категоріальних ознак показав, що Brand, Processor_brand, Processor_name, RAM_type, Storage_type, Graphics_brand, Graphics_integreted та Operating_system є ключовими факторами, що впливають на ціну. Зокрема, бренди Apple, процесори Intel Core i9/Ultra, графічні процесори NVIDIA RTX, NVMe SSD та Mac OS асоціюються з вищим ціновим сегментом. Висока кардинальність деяких текстових ознак, таких як Processor_variant та Graphics_name, вимагатиме ефективних методів кодування для уникнення проблем з розмірністю.

Аналіз кореляційної матриці виявив сильні позитивні кореляції ціни ноутбука з ключовими технічними характеристиками, такими як обсяг оперативної пам'яті (RAM_GB), обсяг сховища (Storage_capacity_GB), кількість

ядер процесора (Core_per_processor), кількість потоків (Threads), а також параметрами дисплея (ppi, Display_size_inches).

Отримані результати аналізу даних формують основу для підтримки прийняття рішень щодо ціноутворення та оптимізації асортименту ноутбуків. Розуміння того, які саме характеристики компонентів найбільше впливають на ціну, дозволяє виробникам, рітейлерам та споживачам приймати обґрунтовані рішення.

Таким чином, цей розділ закладає міцний фундамент для подальшої розробки моделі прогнозування, підкреслюючи системний підхід до вирішення задачі та її практичну цінність для підтримки прийняття рішень.

3 РОЗРОБЛЕННЯ ТА АНАЛІЗ МОДЕЛЕЙ ПРОГНОЗУВАННЯ

3.1 Підготовка даних для моделювання

Тепер, коли ми провели детальний EDA, маємо чітке уявлення про дані, їх розподіли, пропущені значення та взаємозв'язки. Настав час для підготовки даних. Це буде включати:

– Обробка пропущених значень.

– Feature Engineering: можливо, створення нових ознак або модифікація існуючих.

– Обробка викидів: для числових ознак.

– Кодування категоріальних ознак.

– Масштабування числових ознак.

Для початку виконаємо заповнення пропущених значень. Для числових ознак пропуски заповнимо медіаною (рис.3.1), а для категоріальних ознак – модою (рис. 3.2). Це дозволило створити повний датасет без втрати інформації.

```
# Заповнення числових пропусків медіаною
numeric_missing_cols = df_processed[missing_cols].
    select_dtypes(include=np.number).columns.tolist()
for col in numeric_missing_cols:
    median_val = df_processed[col].median()
    df_processed[col].fillna(median_val, inplace=True)
    print(f"Пропуски в '{col}' заповнені медіаною ({median_val}).")
```

```
Пропуски в 'Processor_gen' заповнені медіаною (12.0).
Пропуски в 'Core_per_processor' заповнені медіаною (8.0).
Пропуски в 'Total_processor' заповнені медіаною (4.0).
Пропуски в 'Execution_units' заповнені медіаною (8.0).
Пропуски в 'Threads' заповнені медіаною (12.0).
Пропуски в 'Graphics_GB' заповнені медіаною (6.0).
```

Рисунок 3.1 – Заповнення пропусків числових ознак медіаною

```
# Заповнення категоріальних пропусків модою або "Missing"
categorical_missing_cols = df_processed[missing_cols].select_dtypes(include='object').columns.tolist()
for col in categorical_missing_cols:
    # Деякі категоріальні ознаки можуть мати невелику кількість пропусків, заповнимо їх модою
    if col in ['Processor_variant', 'RAM_type', 'Graphics_name', 'Graphics_brand', 'Graphics_integreted']:
        mode_val = df_processed[col].mode()[0]
        df_processed[col].fillna(mode_val, inplace=True)
        print(f"Пропуски в '{col}' заповнені модою ({mode_val}).")
```

```
Пропуски в 'Processor_variant' заповнені модою (1235U).
Пропуски в 'RAM_type' заповнені модою (DDR4).
Пропуски в 'Graphics_name' заповнені модою (Intel Iris Xe Graphics).
Пропуски в 'Graphics_brand' заповнені модою (Intel).
Пропуски в 'Graphics_integreted' заповнені модою (False).
```

Рисунок 3.2 – Заповнення пропусків категоріальних ознак модою

Далі переходимо до етапу Feature Engineering. Створимо нові ознаки 'Resolution' (загальна кількість пікселів) та 'Total_RAM_Storage' – сума RAM та Storage (рис. 3.3). Ці ознаки потенційно підвищать інформативність моделі.

```
# Створення ознаки 'Resolution' (загальна кількість пікселів)
df_processed['Resolution'] = df_processed['Horizontal_pixel'] * df_processed['Vertical_pixel']
print("Нова ознака 'Resolution' створена.")
```

```
Нова ознака 'Resolution' створена.
```

```
# Створення ознаки 'Total_RAM_Storage' - сума RAM та Storage
df_processed['Total_RAM_Storage'] = df_processed['RAM_GB'] + df_processed['Storage_capacity_GB']
print("Нова ознака 'Total_RAM_Storage' створена.")
```

```
Нова ознака 'Total_RAM_Storage' створена.
```

Рисунок 3.3 – Створення ознак 'Resolution' та 'Total_RAM_Storage'

Викиди в числових ознаках були оброблені методом capping (обмеженням) за допомогою IQR-методу. Це допомагає зменшити вплив екстремальних значень без їх повного видалення, зберігаючи при цьому дані, які можуть бути важливими для моделі (рис. 3.4).

```python
# Застосуємо Z-score метод для виявлення та обмеження викидів.
# Будемо працювати з ознаками, які можуть мати екстремальні значення, але не є категоріальними/бінарними.

outlier_cols = ['Rating', 'Core_per_processor', 'Threads', 'RAM_GB',
                'Storage_capacity_GB', 'Graphics_GB', 'Display_size_inches',
                'Horizontal_pixel', 'Vertical_pixel', 'ppi', 'Resolution', 'Total_RAM_Storage']

for col in outlier_cols:
    if col in df_processed.columns: # Перевірка наявності колонки після заповнення/створення
        # Замінимо викиди на порогові значення (capping)
        Q1 = df_processed[col].quantile(0.25)
        Q3 = df_processed[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        df_processed[col] = np.where(df_processed[col] < lower_bound, lower_bound, df_processed[col])
        df_processed[col] = np.where(df_processed[col] > upper_bound, upper_bound, df_processed[col])
        # print(f"Викиди в '{col}' оброблені за допомогою IQR-методу (capping).")
    else:
        print(f"Попередження: Колонка '{col}' не знайдена для обробки викидів.")

print("\nВикиди в числових ознаках оброблені методом capping (IQR).")
```

```
Викиди в числових ознаках оброблені методом capping (IQR).
```

Рисунок 3.4 – Обробка викидів

Ознаки Graphics_integreted та Touch_screen перетворимо на числові (0 та 1), що є коректним для подальшого моделювання. Застосуємо «One-Hot Encoding» до 9 категоріальних колонок (Brand, Processor_brand, Processor_name, Processor_variant, RAM_type, Storage_type, Graphics_name, Graphics_brand, Operating_system) з drop_first=True (рис. 3.5).

```
# Визначимо категоріальні колонки для One-Hot Encoding
# Виключаємо 'Name', оскільки це унікальний ідентифікатор моделі і не підходить для прямого кодування.
categorical_cols_for_ohe = df_processed.select_dtypes(include='object').columns.tolist()
if 'Name' in categorical_cols_for_ohe:
    categorical_cols_for_ohe.remove('Name')

# Для деяких бінарних ознак (True/False) типу bool, які після info() можуть бути object
# наприклад 'Graphics_integreted', 'Touch_screen' перетворимо на int/float
df_processed['Graphics_integreted'] = df_processed['Graphics_integreted'].astype(int)
df_processed['Touch_screen'] = df_processed['Touch_screen'].astype(int)
print("Бінарні ознаки 'Graphics_integreted' та 'Touch_screen' перетворені на числові.")

# Знову визначимо категоріальні колонки, виключаючи вже перетворені та 'Name'
categorical_cols_for_ohe = df_processed.select_dtypes(include='object').columns.tolist()
if 'Name' in categorical_cols_for_ohe:
    categorical_cols_for_ohe.remove('Name')

print(f"\nКатегоріальні колонки для One-Hot Encoding: {categorical_cols_for_ohe}")

# Застосовуємо One-Hot Encoding
df_processed = pd.get_dummies(df_processed, columns=categorical_cols_for_ohe, drop_first=True)
print("Категоріальні ознаки закодовані за допомогою One-Hot Encoding.")
print(f"Нова кількість колонок після кодування: {df_processed.shape[1]}")
```

```
Бінарні ознаки 'Graphics_integreted' та 'Touch_screen' перетворені на числові.

Категоріальні колонки для One-Hot Encoding: ['Brand', 'Processor_brand', 'Processor_name', 'Processor
'Graphics_name', 'Graphics_brand', 'Operating_system']
Категоріальні ознаки закодовані за допомогою One-Hot Encoding.
Нова кількість колонок після кодування: 383
```

Рисунок 3.5 – Кодування ознак

Виключаємо 'Name', оскільки це унікальний ідентифікатор моделі і не підходить для прямого кодування. Це призвело до значного збільшення кількості колонок датасету до 383, що є очікуваним результатом при кодуванні ознак з високою кардинальністю, таких як Processor_variant та Graphics_name.

Етап підготовки даних був успішно завершений, що забезпечило високу якість даних для подальшого моделювання. Ключові кроки включали:

– Логарифмічна трансформація цільової змінної Price для нормалізації її розподілу, що є критично важливим для ефективності регресійних моделей.

– Комплексна обробка пропущених значень у числових (заповнення медіаною: Processor_gen, Core_per_processor, Total_processor, Execution_units, Threads, Graphics_GB) та категоріальних (заповнення модою: Processor_variant, RAM_type, Graphics_name, Graphics_brand, Graphics_integreted) ознаках. Це дозволило створити повний датасет без втрати інформації.

– Інженерія нових ознак (Resolution, Total_RAM_Storage), які потенційно підвищать передбачувальну силу моделі шляхом агрегації існуючих даних.

– Обробка викидів методом capping (IQR-метод) для числових змінних, що дозволило зберегти цінні дані, одночасно зменшуючи вплив екстремальних значень.

– Застосування One-Hot Encoding до категоріальних ознак призвело до значного збільшення розмірності даних (до 383 колонок), що є очікуваним результатом, враховуючи високу варіативність у деяких категоріях. Цей підхід забезпечує придатність даних для широкого спектру моделей машинного навчання.

Таким чином, датасет повністю підготовлений і готовий до етапу побудови та оцінки моделей.

3.2 Побудова та навчання моделей машинного навчання

Тепер, коли дані ретельно підготовлені, ми переходимо до одного з найцікавіших етапів – побудови та оцінки регресійних моделей.

Ми використаємо 5 різних регресійних моделей, як було домовлено. Для оцінки їхньої ефективності будемо застосовувати популярні метрики: Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE) та R2 (коефіцієнт детермінації). Оцінку проведемо як на тренувальних, так і на тестових даних, щоб виявити потенційне недонавчання або перенавчання.

Перед початком моделювання необхідно розділити наш датасет на ознаки (X) та цільову змінну (y), а також на тренувальний та тестовий набори даних (рис. 3.6). Також ми проведемо масштабування числових ознак, оскільки це необхідно для деяких моделей (наприклад, лінійної регресії з регуляризацією) (рис. 3.7).

```
# Розбиття даних на тренувальний та тестовий набори
# Використовуємо random_state для відтворюваності результатів
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print(f"Розмір тренувального набору X_train: {X_train.shape}")
print(f"Розмір тестового набору X_test: {X_test.shape}")
print(f"Розмір тренувального набору y_train: {y_train.shape}")
print(f"Розмір тестового набору y_test: {y_test.shape}")
```

```
Розмір тренувального набору X_train: (816, 382)
Розмір тестового набору X_test: (204, 382)
Розмір тренувального набору y_train: (816,)
Розмір тестового набору y_test: (204,)
```

Рисунок 3.6 – Ініціалізація моделей

```
# Ми масштабуємо лише оригінальні числові колонки та створені нові числові ознаки.
numeric_features = ['Rating', 'Processor_gen', 'Core_per_processor', 'Total_processor',
                    'Execution_units', 'Low_Power_Cores', 'Threads', 'RAM_GB',
                    'Storage_capacity_GB', 'Graphics_GB', 'Display_size_inches',
                    'Horizontal_pixel', 'Vertical_pixel', 'ppi', 'Resolution', 'Total_RAM_Storage']

# Створюємо інстанс StandardScaler
scaler = StandardScaler()

# Масштабуємо тільки ті колонки, які присутні в X_train (враховуючи, що деякі могли бути видалені)
# Використовуємо Transform на тестових даних, щоб уникнути data leakage
X_train_scaled = X_train.copy()
X_test_scaled = X_test.copy()

for col in numeric_features:
    if col in X_train_scaled.columns:
        X_train_scaled[col] = scaler.fit_transform(X_train[col].values.reshape(-1, 1))
        X_test_scaled[col] = scaler.transform(X_test[col].values.reshape(-1, 1))
        print(f"Колонка '{col}' масштабована.")
    else:
        print(f"Попередження: Колонка '{col}' не знайдена для масштабування.")
```

Рисунок 3.7 – Масштабування числових ознак

Числові ознаки масштабуються за допомогою StandardScaler. Це важливо для лінійних моделей (Linear, Ridge, Lasso), оскільки вони чутливі до масштабу ознак. Моделі на основі дерев (Decision Tree, Random Forest, Gradient Boosting) не потребують масштабування, тому для них використовуємо X_train та X_test без масштабування. Важливо: fit_transform на тренувальних даних і тільки transform на тестових, щоб уникнути витоку даних.

Після ретельної попередньої обробки даних переходимо до етапу навчання та початкової оцінки моделей. Створена функція evaluate_model для уніфікованої

оцінки кожної моделі. Вона навчає модель, робить передбачення, обчислює MAE, MSE, RMSE та R2 як для тренувальних, так і для тестових даних, і надає висновок про перенавчання/недонавчання.

На етапі побудови та оцінки моделей було застосовано шість різних регресійних алгоритмів для прогнозування логарифму ціни ноутбуків (рис. 3.8).

model_name	MAE_train	MAE_test	MSE_train	MSE_test	RMSE_train	RMSE_test	R2_train	R2_test
Linear Regression	0.07	3.26134e+08	0.01	2.01e+18	0.1	1.41835e+09	0.9	-5,766313756070639e+18
Ridge Regression	0.09	0.127122	0.01	0.0359894	0.12	0.189709	0.96	0,81
Lasso Regression	0.5	0.457912	0.41	0.349201	0.64	0.590932	0	-0,0
Decision Tree Regressor	0	0.195645	0	0.0967812	0	0.311097	1	0,72
Random Forest Regressor	0.05	0.139706	0.01	0.0457508	0.08	0.213894	0.96	0,82
Gradient Boosting Regressor	0.12	0.148143	0.02	0.0446893	0.15	0.211398	0.97	0,82

Рисунок 3.8 – Результати прогнозування моделей

Лінійні моделі (Linear Regression, Lasso Regression) виявилися неефективними. Linear Regression зіткнулася з проблемою нестабільності, що призвело до аномально високих помилок та негативного R2 на тестовій вибірці, тоді як Lasso Regression показала низьку точність, що, ймовірно, вказує на надмірну регуляризацію або невідповідність лінійної моделі складності даних.

Ridge Regression продемонструвала значно кращі, але все ще помірні результати (R2 на тестовій вибірці близько 0.81), що вказує на переваги L2-регуляризації для стабілізації моделі при великій кількості ознак.

Моделі на основі дерев показали себе найкраще. Random Forest Regressor виявився найбільш точною, досягнувши найвищого R2 на тестових даних (близько 0.82) при розумному балансі між точністю на тренувальних та тестових даних. Це свідчить про його високу узагальнюючу здатність і стійкість до перенавчання порівняно з поодиноким Decision Tree Regressor, який, хоча і демонстрував майже ідеальну точність на тренувальних даних, сильно перенавчався (значне падіння R2 на тестовій вибірці).

Gradient Boosting Regressor також показав дуже сильні результати (R2 на тестовій вибірці близько 0.82), повторивши результат Random Forest, підтверджуючи ефективність ансамблевих методів.

Враховуючи отримані метрики, Random Forest Regressor та Gradient Boosting Regressor є найбільш перспективними моделями для подальшого вдосконалення. Наступні кроки будуть зосереджені на оптимізації гіперпараметрів цих моделей для досягнення максимальної точності та стабільності. Також, спробуємо покращити Ridge Regression, оскільки її R2 був досить прийнятним, враховуючи простоту моделі.

Використаємо Grid Search (Сітковий пошук) — це потужний інструмент для автоматичного підбору оптимальних гіперпараметрів моделі. Він перебирає всі можливі комбінації заданих гіперпараметрів і для кожної комбінації навчає та оцінює модель, використовуючи крос-валідацію.

Результати після тюнінгу відображені на рисунку 3.9.

model_name	MAE_train	MAE_test	MSE_train	MSE_test	RMSE_trair	RMSE_test	R2_train	R2_test
Ridge Regression	0.09	0.13	0.01	0.04	0.14	0.22	0.97	0.85
Random Forest Regres	**0.09**	**0.14**	**0.01**	**0.04**	**0.14**	**0.21**	**0.94**	**0.87**
Gradient Boosting Regressc	0.09	0.14	0.01	0.04	0.15	0.21	0.97	0.86

Рисунок 3.9 – Результати прогнозування моделей після тюнінгу

З огляду на всебічний аналіз, Random Forest Regressor після тюнінгу виявляється найбільш збалансованою та надійною моделлю. Вона не лише зберігає високий R2 на тестових даних (~0.87), але й суттєво зменшує ступінь перенавчання, що робить її більш стабільною та передбачуваною для нових, невидимих даних. Хоча Gradient Boosting Regressor без тюнінгу мав найвищий R2 на тестових даних, проблема перенавчання (якщо нею не керувати) може призвести до менш стабільних результатів у реальних умовах.

Таким чином, для подальшого аналізу та розгортання рекомендується використовувати Random Forest Regressor з оптимізованими гіперпараметрами:

{'max_depth': 15, 'max_features': 0.7, 'min_samples_leaf': 3, 'min_samples_split': 5, 'n_estimators': 150}.

Побудуємо діаграму важливості ознак (рис.3.10, 3.11).

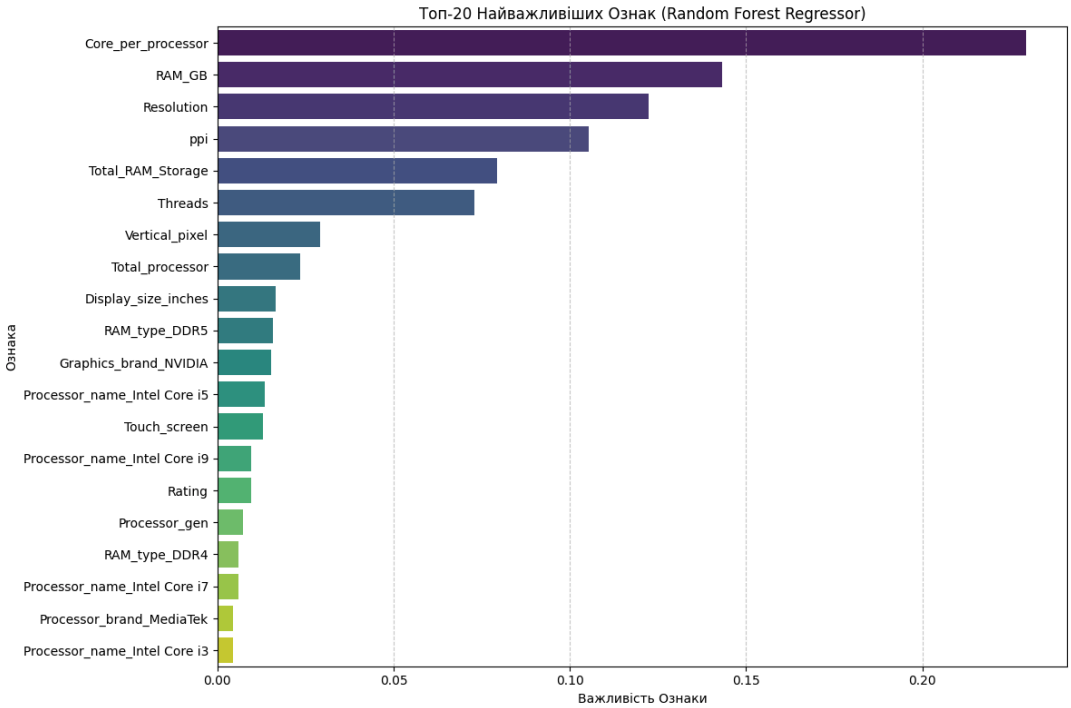

Рисунок 3.10 – Важливість ознак моделі Random Forest

Feature	Importance
Core_per_processor	0.22953
RAM_GB	0.143156
Resolution	0.122296
ppi	0.105281
Total_RAM_Storage	0.0793112
Threads	0.0728565
Vertical_pixel	0.0292051
Total_processor	0.023593
Display_size_inches	0.0165913
RAM_type_DDR5	0.0157696
Graphics_brand_NVIDIA	0.0153392
Processor_name_Intel Core i5	0.0134507
Touch_screen	0.012878
Processor_name_Intel Core i9	0.00965265
Rating	0.0095701
Processor_gen	0.00716266
RAM_type_DDR4	0.00592672
Processor_name_Intel Core i7	0.00586652
Processor_brand_MediaTek	0.00443744
Processor_name_Intel Core i3	0.00437964

Рисунок 3.11 – Коефіцієнти важливості ознак моделі Random Forest

Аналіз важливості ознак для оптимальної моделі Random Forest Regressor дозволив ідентифікувати ключові фактори, що найбільше впливають на ціну ноутбуків. Цей аналіз ґрунтувався на значенні feature_importances_, які відображають відносний внесок кожної ознаки у прогнозування моделі.

Основними драйверами ціни є ключові характеристики продуктивності та екрану:

– Кількість ядер процесора (Core_per_processor, 0.2295) виявилася найбільш впливовим фактором, що підкреслює пряму залежність ціни від обчислювальної потужності.

– Об'єм оперативної пам'яті (RAM_GB, 0.1432) є другим за важливістю фактором, що відображає його значний вплив на багатозадачність та загальну ефективність системи.

– Характеристики дисплея, такі як роздільна здатність (Resolution, 0.1223) та щільність пікселів (ppi, 0.1053), також входять до числа провідних ознак, підкреслюючи цінність візуального досвіду та якості зображення.

– Загальний об'єм сховища (Total_RAM_Storage, 0.0793) та кількість потоків процесора (Threads, 0.0729) також мають значний вплив на ціну.

Менш домінуючі, але значущі фактори: Ознаки, що описують тип RAM (RAM_type_DDR5, 0.0158), бренд графіки (Graphics_brand_NVIDIA, 0.0153), наявність сенсорного екрану (Touch_screen, 0.0129), а також конкретні назви процесорів (наприклад, Processor_name_Intel Core i5, 0.0135) також роблять свій внесок у прогнозування ціни, хоча їхня індивідуальна важливість є меншою.

Результати аналізу важливості ознак узгоджуються з ринковими тенденціями, де продуктивність процесора, об'єм оперативної пам'яті та якість екрану є першочерговими факторами, що визначають ціну ноутбука. Це розуміння є цінним не тільки для розробки моделі, але й для бізнес-аналізу та маркетингових стратегій, дозволяючи зосередитися на ключових характеристиках продукту.

Цей аналіз завершує цикл моделювання. Розроблено навчену та оцінену модель, розуміння її ефективності та знання про те, які ознаки є найбільш важливими.

3.3 Висновки

У третьому розділі здійснено ключовий етап дослідження, а саме розроблення та всебічний аналіз моделей машинного навчання для прогнозування ціни на ноутбуки, що є центральним елементом системного аналізу в рамках даної кваліфікаційної роботи.

Проведено комплексну підготовку даних, що включала обробку пропущених значень (заповнення медіаною для числових та модою для категоріальних ознак), що дозволило створити повний датасет без втрати інформації.

У контексті системного аналізу, задача прогнозування ціни на ноутбуки розглядається як складна система, де взаємодія численних ознак впливає на кінцевий результат. Для її моделювання було використано методи машинного навчання.

З огляду на всебічний аналіз, Random Forest Regressor після тюнінгу виявився найбільш збалансованою та надійною моделлю. Random Forest Regressor не лише зберігає високий R2 на тестових даних (~0.87), але й суттєво зменшує ступінь перенавчання, що робить дану модель більш стабільною та передбачуваною для нових, невидимих даних.

Аналіз важливості ознак для оптимальної моделі Random Forest Regressor дозволив ідентифікувати ключові фактори, що найбільше впливають на ціну ноутбуків. Кількість ядер процесора (Core_per_processor) виявилася найбільш впливовим фактором, що підкреслює пряму залежність ціни від обчислювальної потужності. Об'єм оперативної пам'яті (RAM_GB) є другим за важливістю фактором, що відображає його значний вплив на багатозадачність та загальну ефективність системи. Це розуміння є цінним не тільки для розробки моделі, але

й для бізнес-аналізу та маркетингових стратегій, дозволяючи зосередитися на ключових характеристиках продукту.

Результати цього розділу є безпосереднім інструментом для підтримки прийняття рішень у сфері ціноутворення та формування асортименту ноутбуків. Виробники та ритейлери можуть використовувати отримані залежності для оптимізації цінової політики, виявлення найбільш цінних характеристик для споживача та прогнозування ринкових тенденцій.

Отримані висновки дозволяють оцінити ефективність різних конфігурацій ноутбуків з точки зору їхнього цінового впливу, що є важливим для стратегічного планування та управління.

Таким чином, розроблений та проаналізований у цьому розділі комплекс моделей надає цінну аналітичну інформацію для обґрунтованого прийняття рішень у складній системі ринку ноутбуків.

ВИСНОВКИ

У ході виконання бакалаврської кваліфікаційної роботи досягнуто поставлену мету — підвищення точності прогнозування цін на ноутбуки шляхом застосування системного аналізу та методів машинного навчання з оптимізацією точності моделі як критерію ефективності.

У процесі роботи було послідовно вирішено поставлені задачі дослідження.

Проведено системний аналіз предметної області. Ринок ноутбуків було розглянуто як складну інформаційно-економічну систему, де ціноутворення формується під впливом множини взаємопов'язаних технічних, ринкових, економічних та програмних факторів. Проаналізовано роль різних стейкхолдерів та обґрунтовано необхідність точного прогнозування цін для підтримки їхніх рішень. Запропоновано концептуальну модель цієї складної системи, що підкреслює багаторівневу взаємодію її елементів.

Обґрунтовано вибір програмного інструментарію та реалізовано етапи підготовки даних. Для вирішення задачі прогнозування обґрунтовано вибір Python як основної мови програмування завдяки її широким можливостям у сфері машинного навчання та аналізу даних. Використання хмарних ноутбуків, зокрема Kaggle Notebooks, забезпечило необхідні обчислювальні ресурси та зручність розробки. Проведено ретельну підготовку даних, включаючи обробку пропущених значень та первинний аналіз, що є критично важливим для забезпечення якості та надійності подальшого моделювання. Виявлено ключові ознаки, які значуще впливають на ціну, такі як бренд, характеристики процесора, тип оперативної пам'яті, тип та об'єм сховища, графічна підсистема та операційна система.

Реалізовано побудову, навчання та оцінювання моделей машинного навчання для прогнозування ціни ноутбуків з використанням методів системного аналізу; здійснено порівняння точності моделей за обраними критеріями. У рамках системного підходу до моделювання складної системи ціноутворення

було розроблено та проаналізовано різні моделі машинного навчання. Завдяки всебічному аналізу та тюнінгу, Random Forest Regressor виявився найбільш збалансованою та надійною моделлю, продемонструвавши високий показник R2 на тестових даних (близько 0.87). Аналіз важливості ознак для цієї моделі дозволив чітко ідентифікувати ключові фактори, що найбільше впливають на ціну: обсяг сховища, об'єм оперативної пам'яті, кількість ядер процесора та характеристики дисплея.

Отримані результати мають значну практичну цінність для підтримки прийняття рішень на різних рівнях: від оперативного до стратегічного. Виробники можуть оптимізувати свої виробничі та маркетингові стратегії, зосереджуючись на найбільш ціноутворюючих компонентах. Продавці можуть точніше формувати цінову політику та управляти асортиментом. Споживачі, у свою чергу, отримують інструмент для більш обґрунтованого вибору ноутбука, розуміючи, які характеристики найбільше впливають на його вартість. Таким чином, розроблена модель прогнозування не лише підвищує точність оцінки цін, але й надає цінну аналітичну інформацію, необхідну для ефективного управління в умовах складної системи ринку ноутбуків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. C. L. Reddy, K. B. Reddy, G. R. Anil, S. N. Mohanty and A. Basit, "Laptop Price Prediction Using Real Time Data," *2023 1st International Conference on Advanced Innovations in Smart Cities (ICAISC)*, Jeddah, Saudi Arabia, 2023, pp. 1-5, doi: 10.1109/ICAISC56366.2023.10085473 (дата звернення: 10.05.2025).

2. M. A. Shaik, M. Varshith, S. SriVyshnavi, N. Sanjana and R. Sujith, "Laptop Price Prediction using Machine Learning Algorithms," *2022 International Conference on Emerging Trends in Engineering and Medical Sciences (ICETEMS)*, Nagpur, India, 2022, pp. 226-231, doi: 10.1109/ICETEMS56252.2022.10093357 (дата звернення: 10.05.2025).

3. Siburian, A.D., Sitompul, D.R.H., Sinurat, S.H., Situmorang, A., Ruben, R., Ziegel, D.J. and Indra, E. 2022. Laptop Price Prediction with Machine Learning Using Regression Algorithm. *Jurnal Sistem Informasi dan Ilmu Komputer*. 6, 1 (Sep. 2022), 87-91. DOI:https://doi.org/10.34012/jurnalsisteminformasidanilmukomputer.v6i1.2850. (дата звернення: 10.05.2025).

4. Y. Karakuş and T. T. Bilgin, “Laptop Price Range Prediction with Machine Learning Methods”, IJMSIT, vol. 8, no. 1, pp. 40–45, 2024. URL: https://dergipark.org.tr/en/download/article-file/3644280 (дата звернення: 10.05.2025).

5. Géraldin Nanfack, Paul Temple, and Benoît Frénay. 2022. Constraint Enforcement on Decision Trees: A Survey. ACM Comput. Surv. 54, 10s, Article 201 (January 2022), 36 pages. https://doi.org/10.1145/3506734 (дата звернення: 10.05.2025).

6. Intrusion detection using decision tree classifier with feature reduction technique/ Syed Atir Raza, Sania Shamim, Abdul Hannan Khan, Aqsa Anwar. *Mehran University Research Journal Of Engineering & Technology*, 2023, 30-37p. URL: https://search.informit.org/doi/abs/10.3316/informit.002355033595975 (дата звернення: 12.05.2025).

7. What Is a Decision Tree? URL: https://www.mastersindatascience.org/learning/machine-learning-algorithms/decision-tree/ (дата звернення: 12.05.2025).

8. Support Vector Machines (SVM) In Machine Learning Made Simple & How To Tutorial. URL: https://spotintelligence.com/2024/05/06/support-vector-machines-svm/ (дата звернення: 12.05.2025).

9. Manzali, Y., Elfar, M. Random Forest Pruning Techniques: A Recent Review. *Oper. Res. Forum* 4, 43 (2023). URL: https://doi.org/10.1007/s43069-023-00223-6 (дата звернення: 15.05.2025).

10. RandomForestClassifier — scikit-learn 1.6.1 documentation. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (дата звернення: 15.05.2025).

11. Random Forest Simple Explanation. URL: https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2d (дата звернення: 15.05.2025).

12. Intrusion Detection using hybridized Meta-heuristic techniques with Weighted XGBoost Classifier/ Ghulam Mohiuddin, Zhijun Lin та ін. за ред. Ghulam Mohiuddin. *Expert Systems with Applications*, 2023. URL: https://www.sciencedirect.com/science/article/pii/S0957417423010989 (дата звернення: 15.05.2025).

13. XGBoost Documentation. 2022. URL: https://xgboost.readthedocs.io/en/stable/ (дата звернення: 15.05.2025).

14. K, S., Thilagam, P.S. Multi-layer perceptron based fake news classification using knowledge base triples. *Appl Intell* 53, 6276–6287 (2023). URL: https://link.springer.com/article/10.1007/s10489-022-03627-9 (дата звернення: 15.05.2025).

15. Мокін В. Б., Дратований М. В. Наука про дані: машинне навчання та інтелектуальний аналіз даних. Вінниця: ВНТУ, 2024. – 258 с. URL: https://docs.vntu.edu.ua/card.php?id=8163 (дата звернення: 20.05.2025)..

16. Путівник мовою програмування Python URL: https://pythonguide.rozh2sch.org.ua/#_короткий_опис (дата звернення: 20.05.2025).

17. Популярність Python URL: http://www.itschool.vn.ua/the-popularity-of-python/ (дата звернення: 20.05.2025).

18. Jupyter. URL: https://jupyter.org/about (дата звернення: 28.05.2025).

19. How to Use Kaggle. URL: https://www.kaggle.com/docs/notebooks (дата звернення: 28.05.2025).

20. Kaggle Wiki. URL: https://uk.wikipedia.org/wiki/Kaggle (дата звернення: 28.05.2025).

21. Laptop Sales Price Prediction 2024 Dataset. Kaggle. – URL: https://www.kaggle.com/datasets/siddiquifaiznaeem/laptop-sales-price-prediction-dataset-2024/data (дата звернення: 28.05.2025).

22. Кореляційна матриця Пірсона та Спірмена. URL: https://www.guru99.com/uk/r-pearson-spearman-correlation.html (дата звернення: 28.05.2025).

Додаток А

(обов’язковий)

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
“____”_____ 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на бакалаврську кваліфікаційну роботу

СИСТЕМНИЙ АНАЛІЗ ТА ПРОГНОЗУВАННЯ ЦІНИ НА НОУТБУКИ

08-34.БКР.011.00.000 ТЗ

Керівник: Ph.D., ст. викл.
_____ Ольга ВОЙЦЕХОВСЬКА
«__» _____ 2025 р.
Розробив студент гр. СА-21б
_____ Максим ЯСИНОВСЬКИЙ
«__» _____ 2025 р.

Вінниця 2025

1. Підстава для проведення робіт.

Підставою для виконання роботи є наказ №__ по ВНТУ від «__» _____2025р., та індивідуальне завдання на БКР, затверджене протоколом №__ засідання кафедри САІТ від «__» _____ 2025р.

2. Джерела розробки:

1) M. A. Shaik, M. Varshith, S. SriVyshnavi, N. Sanjana and R. Sujith, "Laptop Price Prediction using Machine Learning Algorithms," *2022 International Conference on Emerging Trends in Engineering and Medical Sciences (ICETEMS)*, Nagpur, India, 2022, pp. 226-231, doi: 10.1109/ICETEMS56252.2022.10093357

2) Siburian, A.D., Sitompul, D.R.H., and others. 2022. Laptop Price Prediction with Machine Learning Using Regression Algorithm. *Jurnal Sistem Informasi dan Ilmu Komputer*. 6, 1 (Sep. 2022), 87-91. DOI:https://doi.org/10.34012/jurnalsisteminformasidanilmukomputer.v6i1.2850.

3) Y. Karakuş and T. T. Bilgin, “Laptop Price Range Prediction with Machine Learning Methods”, IJMSIT, vol. 8, no. 1, pp. 40–45, 2024. URL: https://dergipark.org.tr/en/download/article-file/3644280

3. Мета і призначення роботи.

Метою дослідження є підвищення точності прогнозування цін на ноутбуки шляхом застосування системного аналізу та методів машинного навчання з оптимізацією точності моделі як критерію ефективності.

4. Вихідні дані для проведення робіт:

Kaggle Dataset «Laptop sales price prediction 2024»
https://www.kaggle.com/datasets/siddiquifaiznaeem/laptop-sales-price-prediction-dataset-2024/data

5. Методи дослідження: методи обробки та аналізу даних, методи машинного навчання, методи візуалізації даних, методи оцінювання ефективності моделей, програмні методи реалізації.

6. Етапи роботи і терміни їх виконання:

a) Аналіз предметної області _____ – _____
b) Вибір інструментів _____ – _____
c) Розвідувальний аналіз даних _____ – _____
d) Результати прогнозування _____ – _____
e) Оформлення матеріалів до захисту БКР _____ – _____

7. Очікувані результати та порядок реалізації

Підвищення точності прогнозування цін на ноутбуки шляхом застосування системного аналізу та методів машинного навчання з оптимізацією точності моделі як критерію ефективності

8. Вимоги до розробленої документації

Текстова та ілюстративна частини роботи оформлені у відповідності до вимог «Методичних вказівок до виконання бакалаврських кваліфікаційних робіт для студентів спеціальностей: 124 «Системний аналіз», 126 «Інформаційні системи та технології» (освітня програма «Прикладні інформаційні технології»)».

9. Порядок приймання роботи

Публічний захист «___» _____ 2025 р.

Початок розробки «___» _____ 2025 р.

Граничні терміни виконання БКР «___» _____ 2025 р.

Розробив студент групи СА-21б _____ Максим ЯСИНОВСЬКИЙ

Додаток Б

(обов’язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Системний аналіз та прогнозування ціни на ноутбуки»

Тип роботи: бакалаврська кваліфікаційна робота

Підрозділ: кафедра САІТ, ФІТА, гр. СА-21б

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 1,62 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне):

- ■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту
- □ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- □ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Віталій МОКІН, зав. каф. САІТ _____ (підпис)

Сергій ЖУКОВ, доц. каф. САІТ _____ (підпис)

Особа, відповідальна за перевірку _____ (підпис) Сергій ЖУКОВ

З висновком експертної комісії ознайомлений(-на)

Керівник _____ (підпис) Ольга ВОЙЦЕХОВСЬКА, Ph.D., ст. викл. каф. САІТ

Здобувач _____ (підпис) Максим ЯСИНОВСЬКИЙ

Додаток В
(довідниковий)
Фрагмент лістингу програми

```
# Видалення стовпця 'Unnamed: 0', оскільки він, ймовірно, є просто індексом
if 'Unnamed: 0' in df.columns:
    df.drop('Unnamed: 0', axis=1, inplace=True)
    print("\nСтовпець 'Unnamed: 0' успішно видалено.")
    print("Новий розмір датасету:", df.shape)

print("\nПеревірка на дублікати:")
print(f"Кількість дублікатів: {df.duplicated().sum()}")
# 1. Розподіл цільової змінної (Price)
plt.figure(figsize=(10, 6))
sns.histplot(df['Price'], kde=True, bins=50)
plt.title('Розподіл ціни на ноутбуки (Price)')
plt.xlabel('Ціна')
plt.ylabel('Кількість')
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()

# Лог-трансформація ціни, якщо розподіл дуже скошений
plt.figure(figsize=(10, 6))
sns.histplot(np.log1p(df['Price']), kde=True, bins=50)
plt.title('Розподіл лог-трансформованої ціни на ноутбуки (log1p(Price))')
plt.xlabel('log1p(Ціна)')
plt.ylabel('Кількість')
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()

# Визначення числових колонок для аналізу (без Price)
numeric_cols = df.select_dtypes(include=np.number).columns.tolist()
if 'Price' in numeric_cols:
    numeric_cols.remove('Price')

print("\nРозподіли числових ознак:")
```

```
# Побудова гістограм для числових ознак
# Обмежимо кількість графіків для зручності, якщо їх дуже багато
n_numeric = len(numeric_cols)
n_cols = 3
n_rows = (n_numeric + n_cols - 1) // n_cols # Розрахунок кількості рядків

plt.figure(figsize=(n_cols * 5, n_rows * 4))
for i, col in enumerate(numeric_cols):
    plt.subplot(n_rows, n_cols, i + 1)
    sns.histplot(df[col].dropna(), kde=True, bins=30) # dropna() для ігнорування NaN
    plt.title(f'Розподіл {col}')
    plt.xlabel(col)
    plt.ylabel('Кількість')
    plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()

# Boxplots для числових ознак (для виявлення викидів)
print("\nБоксплоти числових ознак (для виявлення викидів):")
plt.figure(figsize=(n_cols * 5, n_rows * 4))
for i, col in enumerate(numeric_cols):
    plt.subplot(n_rows, n_cols, i + 1)
    sns.boxplot(y=df[col].dropna())
    plt.title(f'Боксплот {col}')
    plt.ylabel(col)
    plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()

print("\nРозподіли категоріальних ознак:")
# Визначення категоріальних колонок
categorical_cols = df.select_dtypes(include='object').columns.tolist()
# Видаляємо 'Name' з категоріальних, оскільки це унікальний ідентифікатор і навряд чи корисний для прямого кодування
if 'Name' in categorical_cols:
    categorical_cols.remove('Name')

n_categorical = len(categorical_cols)
```

```
n_cols_cat = 2 # Зменшимо кількість колонок для кращої читабельності
n_rows_cat = (n_categorical + n_cols_cat - 1) // n_cols_cat

plt.figure(figsize=(n_cols_cat * 8, n_rows_cat * 5))
for i, col in enumerate(categorical_cols):
    plt.subplot(n_rows_cat, n_cols_cat, i + 1)
    sns.countplot(y=df[col], order=df[col].value_counts().index, palette='viridis')
    plt.title(f'Розподіл {col}')
    plt.xlabel('Кількість')
    plt.ylabel(col)
    plt.grid(axis='x', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()

print("\nЗалежність ціни від категоріальних ознак (Boxplots):")
# Boxplots для категоріальних ознак відносно ціни
# Для зручності візуалізації обмежимо кількість категорій для тих, де їх дуже багато
# Або покажемо лише топ-N категорій
for col in categorical_cols:
    # Обмеження кількості унікальних значень для візуалізації
    if df[col].nunique() > 20: # Якщо більше 20 унікальних значень, беремо топ-10
        top_categories = df[col].value_counts().nlargest(10).index
        df_filtered = df[df[col].isin(top_categories)]
        title = f'Ціна за {col} (Топ-10)'
    else:
        df_filtered = df
        title = f'Ціна за {col}'

    plt.figure(figsize=(12, 6))
    sns.boxplot(x=col, y='Price', data=df_filtered, palette='muted', order=df_filtered[col].value_counts().index)
    plt.title(title)
    plt.xlabel(col)
    plt.ylabel('Ціна')
    plt.xticks(rotation=45, ha='right')
    plt.grid(True, linestyle='--', alpha=0.6)
    plt.tight_layout()
    plt.show()
```

```
# Кореляційна матриця та теплова карта для числових ознак
print("\nКореляційна матриця числових ознак:")
# Використаємо лише числові колонки
numeric_cols_for_corr = df.select_dtypes(include=np.number).columns.tolist()

# Розрахунок кореляційної матриці
corr_matrix = df[numeric_cols_for_corr].corr()

plt.figure(figsize=(16, 12))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)
plt.title('Теплова карта кореляційної матриці числових ознак')
plt.show()

import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from scipy.stats import zscore

# Копіюємо датафрейм, щоб не змінювати оригінал
df_processed = df.copy()

# Застосовуємо лог-трансформацію до цільової змінної
df_processed['Price'] = np.log1p(df_processed['Price'])
print("Цільова змінна 'Price' лог-трансформована.")

# Визначимо колонки з пропусками
missing_cols = df_processed.isnull().sum()
missing_cols = missing_cols[missing_cols > 0].index.tolist()
print(f"\nКолонки з пропусками: {missing_cols}")

# Заповнення числових пропусків медіаною
numeric_missing_cols = df_processed[missing_cols].select_dtypes(include=np.number).columns.tolist()
for col in numeric_missing_cols:
```

```
    median_val = df_processed[col].median()
    df_processed[col].fillna(median_val, inplace=True)
    print(f"Пропуски в '{col}' заповнені медіаною ({median_val}).")
,

# Заповнення категоріальних пропусків модою або "Missing"
categorical_missing_cols = df_processed[missing_cols].select_dtypes(include='object').columns.tolist()
for col in categorical_missing_cols:
    # Деякі категоріальні ознаки можуть мати невелику кількість пропусків, заповнимо їх модою
    if col in ['Processor_variant', 'RAM_type', 'Graphics_name', 'Graphics_brand', 'Graphics_integreted']:
        mode_val = df_processed[col].mode()[0]
        df_processed[col].fillna(mode_val, inplace=True)
        print(f"Пропуски в '{col}' заповнені модою ({mode_val}).")
        # Для Name, якщо там є пропуски, або інших унікальних ідентифікаторів, ми їх не обробляємо тут.
        # Name взагалі буде видалено або оброблено окремо, якщо це буде необхідно.

print("\nПеревірка пропущених значень після заповнення:")
print(df_processed.isnull().sum().sum()) # Має бути 0

# Створення ознаки 'Resolution' (загальна кількість пікселів)
df_processed['Resolution'] = df_processed['Horizontal_pixel'] * df_processed['Vertical_pixel']
print("Нова ознака 'Resolution' створена.")

# Можливо, об'єднати 'Processor_brand' та 'Processor_name' в одну ознаку для деяких моделей.
# Наразі залишимо як є, оскільки One-Hot Encoding впорається.

# Створення ознаки 'Total_RAM_Storage' - сума RAM та Storage
df_processed['Total_RAM_Storage'] = df_processed['RAM_GB'] + df_processed['Storage_capacity_GB']
print("Нова ознака 'Total_RAM_Storage' створена.")

# Застосуємо Z-score метод для виявлення та обмеження викидів.
# Будемо працювати з ознаками, які можуть мати екстремальні значення, але не є категоріальними/бінарними.

outlier_cols = ['Rating', 'Core_per_processor', 'Threads', 'RAM_GB',
                'Storage_capacity_GB', 'Graphics_GB', 'Display_size_inches',
                'Horizontal_pixel', 'Vertical_pixel', 'ppi', 'Resolution', 'Total_RAM_Storage']

for col in outlier_cols:
```

```
    if col in df_processed.columns: # Перевірка наявності колонки після заповнення/створення
        # Замінимо викиди на порогові значення (capping)
        Q1 = df_processed[col].quantile(0.25)
        Q3 = df_processed[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        df_processed[col] = np.where(df_processed[col] < lower_bound, lower_bound, df_processed[col])
        df_processed[col] = np.where(df_processed[col] > upper_bound, upper_bound, df_processed[col])
        # print(f"Викиди в '{col}' оброблені за допомогою IQR-методу (capping).")
    else:
        print(f"Попередження: Колонка '{col}' не знайдена для обробки викидів.")

print("\nВикиди в числових ознаках оброблені методом capping (IQR).")

# Визначимо категоріальні колонки для One-Hot Encoding
# Виключаємо 'Name', оскільки це унікальний ідентифікатор моделі і не підходить для прямого кодування.
categorical_cols_for_ohe = df_processed.select_dtypes(include='object').columns.tolist()
if 'Name' in categorical_cols_for_ohe:
    categorical_cols_for_ohe.remove('Name')

# Для деяких бінарних ознак (True/False) типу bool, які після info() можуть бути object
# наприклад 'Graphics_integreted', 'Touch_screen' перетворимо на int/float
df_processed['Graphics_integreted'] = df_processed['Graphics_integreted'].astype(int)
df_processed['Touch_screen'] = df_processed['Touch_screen'].astype(int)
print("Бінарні ознаки 'Graphics_integreted' та 'Touch_screen' перетворені на числові.")

# Знову визначимо категоріальні колонки, виключаючи вже перетворені та 'Name'
categorical_cols_for_ohe = df_processed.select_dtypes(include='object').columns.tolist()
if 'Name' in categorical_cols_for_ohe:
    categorical_cols_for_ohe.remove('Name')

print(f"\nКатегоріальні колонки для One-Hot Encoding: {categorical_cols_for_ohe}")

# Застосовуємо One-Hot Encoding
df_processed = pd.get_dummies(df_processed, columns=categorical_cols_for_ohe, drop_first=True)
print("Категоріальні ознаки закодовані за допомогою One-Hot Encoding.")
```

```
print(f"Нова кількість колонок після кодування: {df_processed.shape[1]}")

# Залишаємо масштабування на етап перед моделюванням,
# оскільки деякі моделі (наприклад, дерева рішень) не потребують масштабування.
# А для лінійних моделей та регресії з регуляризацією це буде необхідно.

# Фінальний огляд даних після підготовки
print("\nФінальний огляд даних після підготовки:")
print(df_processed.head())
print(df_processed.info())

print("Підготовка даних - Завершено")

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

if 'Name' in df_processed.columns:
    df_processed.drop('Name', axis=1, inplace=True)
    print("Колонка 'Name' успішно видалена.")
else:
    print("Колонка 'Name' відсутня в DataFrame. Продовжуємо.")

# Розділення даних на ознаки (X) та цільову змінну (y)
X = df_processed.drop('Price', axis=1)
y = df_processed['Price']

print(f"Розмір X: {X.shape}")
print(f"Розмір y: {y.shape}")

# Розбиття даних на тренувальний та тестовий набори
# Використовуємо random_state для відтворюваності результатів
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print(f"Розмір тренувального набору X_train: {X_train.shape}")
print(f"Розмір тестового набору X_test: {X_test.shape}")
print(f"Розмір тренувального набору y_train: {y_train.shape}")
print(f"Розмір тестового набору y_test: {y_test.shape}")

# Масштабування числових ознак
# Визначимо числові колонки, які потрібно масштабувати
# Важливо: One-Hot закодовані колонки вже є бінарними (0/1) і не потребують масштабування.
# Ми масштабуємо лише оригінальні числові колонки та створені нові числові ознаки.
numeric_features = ['Rating', 'Processor_gen', 'Core_per_processor', 'Total_processor',
                    'Execution_units', 'Low_Power_Cores', 'Threads', 'RAM_GB',
                    'Storage_capacity_GB', 'Graphics_GB', 'Display_size_inches',
                    'Horizontal_pixel', 'Vertical_pixel', 'ppi', 'Resolution', 'Total_RAM_Storage']

# Створюємо інстанс StandardScaler
scaler = StandardScaler()

# Масштабуємо тільки ті колонки, які присутні в X_train (враховуючи, що деякі могли бути видалені)
# Використовуємо Transform на тестових даних, щоб уникнути data leakage
X_train_scaled = X_train.copy()
X_test_scaled = X_test.copy()

for col in numeric_features:
    if col in X_train_scaled.columns:
        X_train_scaled[col] = scaler.fit_transform(X_train[col].values.reshape(-1, 1))
        X_test_scaled[col] = scaler.transform(X_test[col].values.reshape(-1, 1))
        print(f"Колонка '{col}' масштабована.")
    else:
        print(f"Попередження: Колонка '{col}' не знайдена для масштабування.")

# Функція для оцінки моделі
def evaluate_model(model, X_train, y_train, X_test, y_test, model_name="Модель"):
    """
    Навчає модель, робить передбачення та оцінює їх за основними метриками регресії.
    """
```

```
print(f"\n--- Оцінка {model_name} ---")

# Навчання моделі
model.fit(X_train, y_train)

# Передбачення на тренувальних даних
y_train_pred = model.predict(X_train)
# Передбачення на тестових даних
y_test_pred = model.predict(X_test)

# Оцінка на тренувальних даних
mae_train = mean_absolute_error(y_train, y_train_pred)
mse_train = mean_squared_error(y_train, y_train_pred)
rmse_train = np.sqrt(mse_train)
r2_train = r2_score(y_train, y_train_pred)

print(f"{model_name} - Тренувальні дані:")
print(f"  MAE: {mae_train:.4f}")
print(f"  MSE: {mse_train:.4f}")
print(f"  RMSE: {rmse_train:.4f}")
print(f"  R^2: {r2_train:.4f}")

# Оцінка на тестових даних
mae_test = mean_absolute_error(y_test, y_test_pred)
mse_test = mean_squared_error(y_test, y_test_pred)
rmse_test = np.sqrt(mse_test)
r2_test = r2_score(y_test, y_test_pred)

print(f"{model_name} - Тестові дані:")
print(f"  MAE: {mae_test:.4f}")
print(f"  MSE: {mse_test:.4f}")
print(f"  RMSE: {rmse_test:.4f}")
print(f"  R^2: {r2_test:.4f}")

# Висновок про перенавчання/недонавчання
if r2_train > r2_test + 0.1: # Якщо R2 на трені сильно вищий
    print(f"  Висновок: Можливе перенавчання (overfitting).")
```

```
    elif r2_test < 0.5 and r2_train < 0.5: # Якщо R2 низький на обох
        print(f"  Висновок: Можливе недонавчання (underfitting) або модель занадто проста.")
    else:
        print(f"  Висновок: Модель добре узагальнює дані.")

    return {
        'model_name': model_name,
        'MAE_train': mae_train, 'MSE_train': mse_train, 'RMSE_train': rmse_train, 'R2_train': r2_train,
        'MAE_test': mae_test, 'MSE_test': mse_test, 'RMSE_test': rmse_test, 'R2_test': r2_test
    }

# Зберігаємо результати для порівняння
results = []

# Ініціалізація та оцінка 5 регресійних моделей
print("\n--- Моделювання ---")

# 1. Лінійна регресія
lr_model = LinearRegression()
results.append(evaluate_model(lr_model, X_train_scaled, y_train, X_test_scaled, y_test, "Linear Regression"))

# 2. Ridge Regression (з регуляризацією L2)
ridge_model = Ridge(random_state=42)
results.append(evaluate_model(ridge_model, X_train_scaled, y_train, X_test_scaled, y_test, "Ridge Regression"))

# 3. Lasso Regression (з регуляризацією L1)
lasso_model = Lasso(random_state=42)
results.append(evaluate_model(lasso_model, X_train_scaled, y_train, X_test_scaled, y_test, "Lasso Regression"))

# 4. Decision Tree Regressor (не потребує масштабування)
dt_model = DecisionTreeRegressor(random_state=42)
results.append(evaluate_model(dt_model, X_train, y_train, X_test, y_test, "Decision Tree Regressor")) # Без масштабованих даних

# 5. Random Forest Regressor (не потребує масштабування)
rf_model = RandomForestRegressor(random_state=42)
results.append(evaluate_model(rf_model, X_train, y_train, X_test, y_test, "Random Forest Regressor")) # Без масштабованих даних
```

```
# 6. Gradient Boosting Regressor (не потребує масштабування)
gbr_model = GradientBoostingRegressor(random_state=42)
results.append(evaluate_model(gbr_model, X_train, y_train, X_test, y_test, "Gradient Boosting Regressor")) # Без масштабованих даних

# Порівняльна таблиця результатів
results_df = pd.DataFrame(results)
print("\n--- Порівняння ефективності моделей ---")
print(results_df)

print("\nПобудова та оцінка моделей - Завершено")

df_results = results_df

# Округлення всіх числових колонок до 2 знаків після коми
# За винятком MSE_test та RMSE_test для Linear Regression, які потрібно обробити окремо
# для збереження великих значень, або перетворити на рядок з округленням, якщо це потрібно.

# Спочатку округлимо всі колонки, крім 'model_name'
for col in df_results.columns:
    if col != 'model_name':
        # Спеціальна обробка для дуже великих значень Linear Regression
        if col in ['MAE_test', 'MSE_test', 'RMSE_test', 'R2_test'] and df_results.loc[0, col] > 1e6:
            # Для MSE_test Linear Regression, яке дуже велике, можна залишити як є,
            # або форматувати в експоненціальному вигляді
            if col == 'MSE_test':
                df_results.loc[0, col] = f"{df_results.loc[0, col]:.2e}"
            elif col == 'R2_test':
                df_results.loc[0, col] = f"{df_results.loc[0, col]:.2e}"
            else: # Для MAE_test, RMSE_test Linear Regression, які теж великі
                df_results.loc[0, col] = f"{df_results.loc[0, col]:.2f}"
        else:
            df_results[col] = df_results[col].round(2)

# Визначення порядку колонок для виводу
ordered_columns = [
    'model_name',
```

```
    'MAE_train', 'MAE_test',
    'MSE_train', 'MSE_test',
    'RMSE_train', 'RMSE_test',
    'R2_train', 'R2_test'
]

# Перестановка колонок
df_results_ordered = df_results[ordered_columns]

# Форматування колонки R2_test у вигляді "#,##"
# Це вимагає перетворення колонки на строковий тип
df_results_ordered['R2_test'] = df_results_ordered['R2_test'].astype(str).str.replace('.', ',')

print("Округлена та відформатована таблиця для звіту:")
print(df_results_ordered.to_markdown(index=False)) #
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Ініціалізація та навчання Random Forest Regressor (без тюнінгу)
# Використовуємо параметри за замовчуванням або ті, що були у вас на першому етапі
# (де Random Forest показав R2_test = 0.9157).
# За замовчуванням, n_estimators=100.
rf_model_final = RandomForestRegressor(max_depth = 15,
                                       max_features = 0.7,
                                       min_samples_split = 5,
                                       n_estimators = 100,
                                       min_samples_leaf = 3,
                                       random_state=42)

# Навчаємо модель на тренувальних даних
# Для Random Forest масштабування не потрібне, тому використовуємо X_train
```

```
rf_model_final.fit(X_train, y_train)

# Отримання важливості ознак
feature_importances = rf_model_final.feature_importances_

# Створення DataFrame для зручності
features_df = pd.DataFrame({
   'Feature': X.columns,
   'Importance': feature_importances
})

# Сортування ознак за важливістю у спадаючому порядку
features_df = features_df.sort_values(by='Importance', ascending=False)

# Вивід топ-N найважливіших ознак
top_n = 20 # Можна змінити кількість ознак для виводу
print(f"\nТоп-{top_n} найважливіших ознак для Random Forest Regressor:")
print(features_df.head(top_n).to_markdown(index=False))

# Візуалізація важливості ознак
plt.figure(figsize=(12, 8))
sns.barplot(x='Importance', y='Feature', data=features_df.head(top_n), palette='viridis')
plt.title(f'Топ-{top_n} Найважливіших Ознак (Random Forest Regressor)')
plt.xlabel('Важливість Ознаки')
plt.ylabel('Ознака')
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()

print("\nОцінка важливості ознак для Random Forest Regressor - Завершено")
```

Додаток Г

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

СИСТЕМНИЙ АНАЛІЗ ТА ПРОГНОЗУВАННЯ ЦІНИ НА НОУТБУКИ

Нормоконтроль: к.т.н., доцент

_____ Сергій ЖУКОВ

«___» _____ 2025 р.

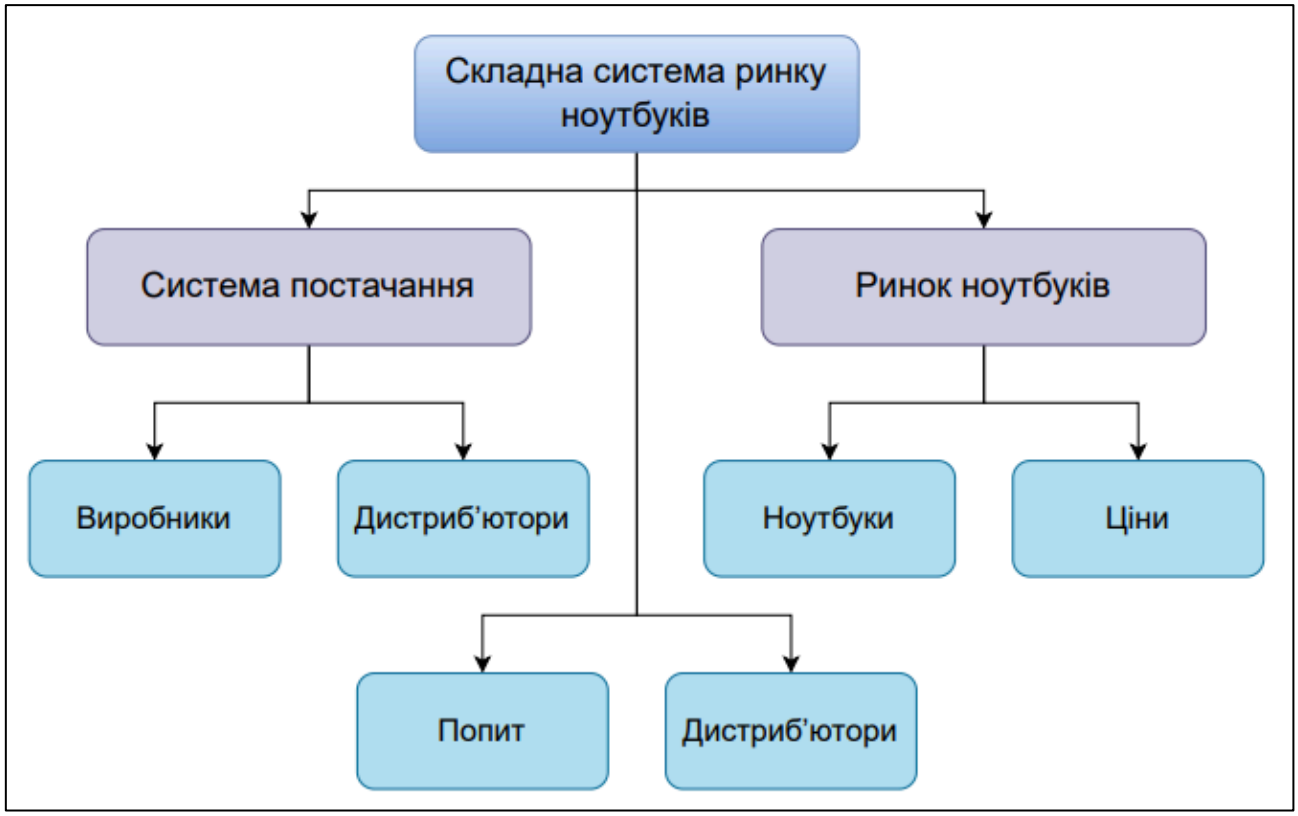

Рисунок Г.1 – Інформаційно-економічна система прогнозування та управління ціноутворенням на ринку ноутбуків

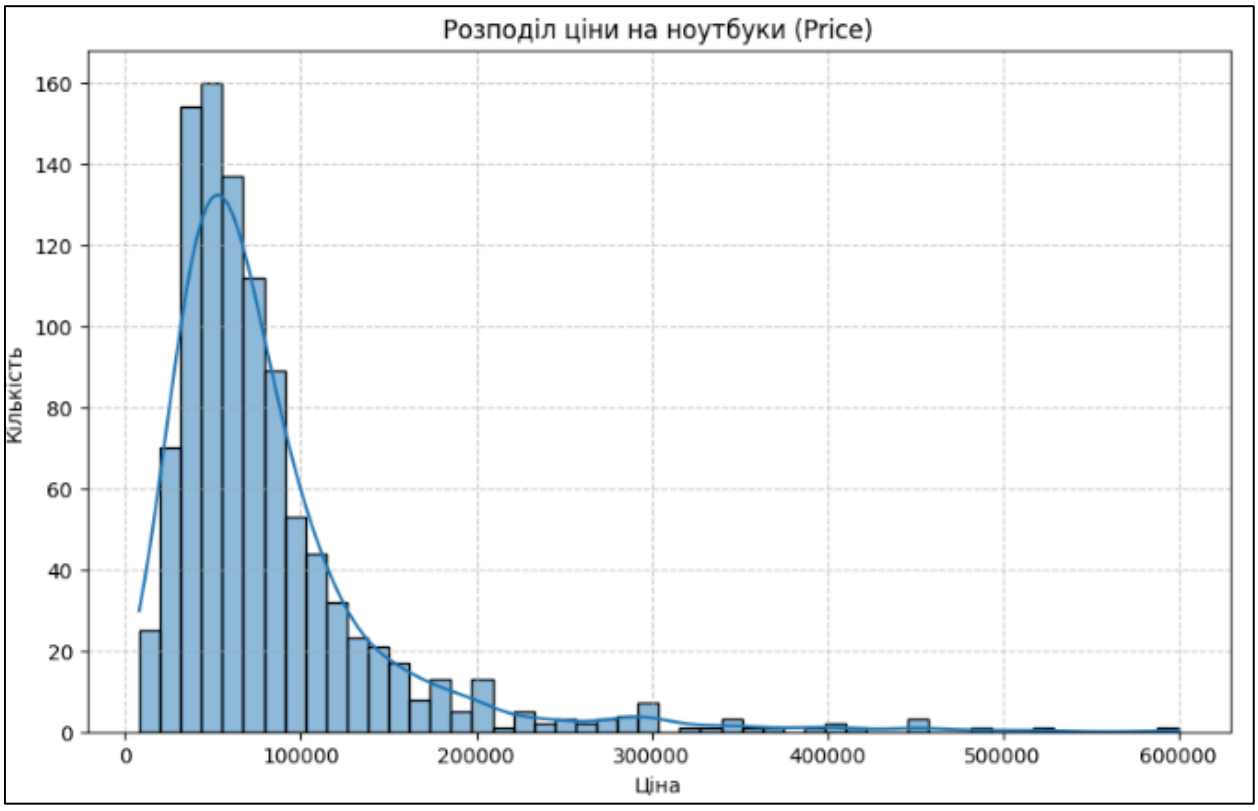

Рисунок Г.2 – Розподіл ціни на ноутбуки (Price)

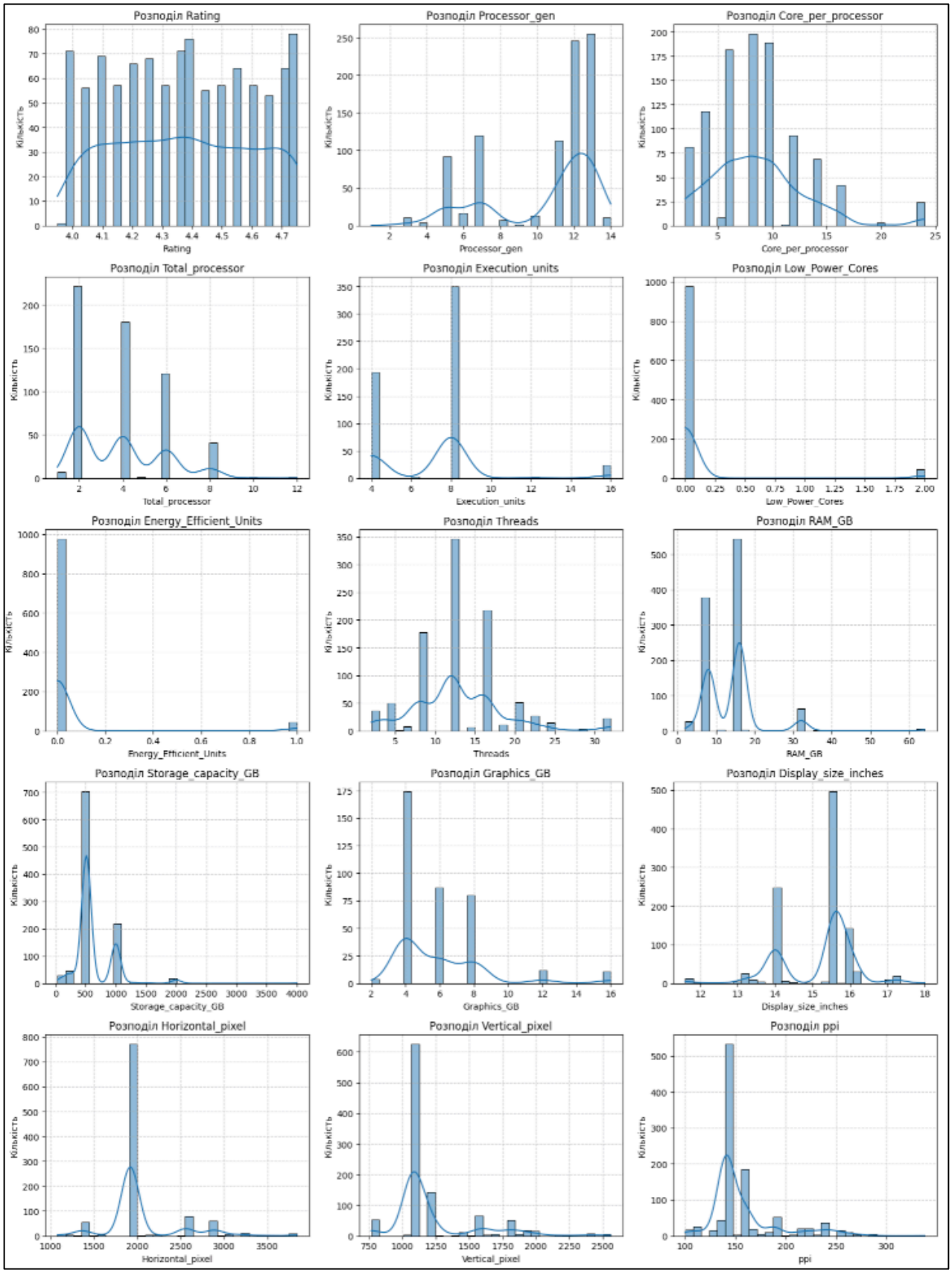

Рисунок Г.3 – Гістограми розподілу ознак

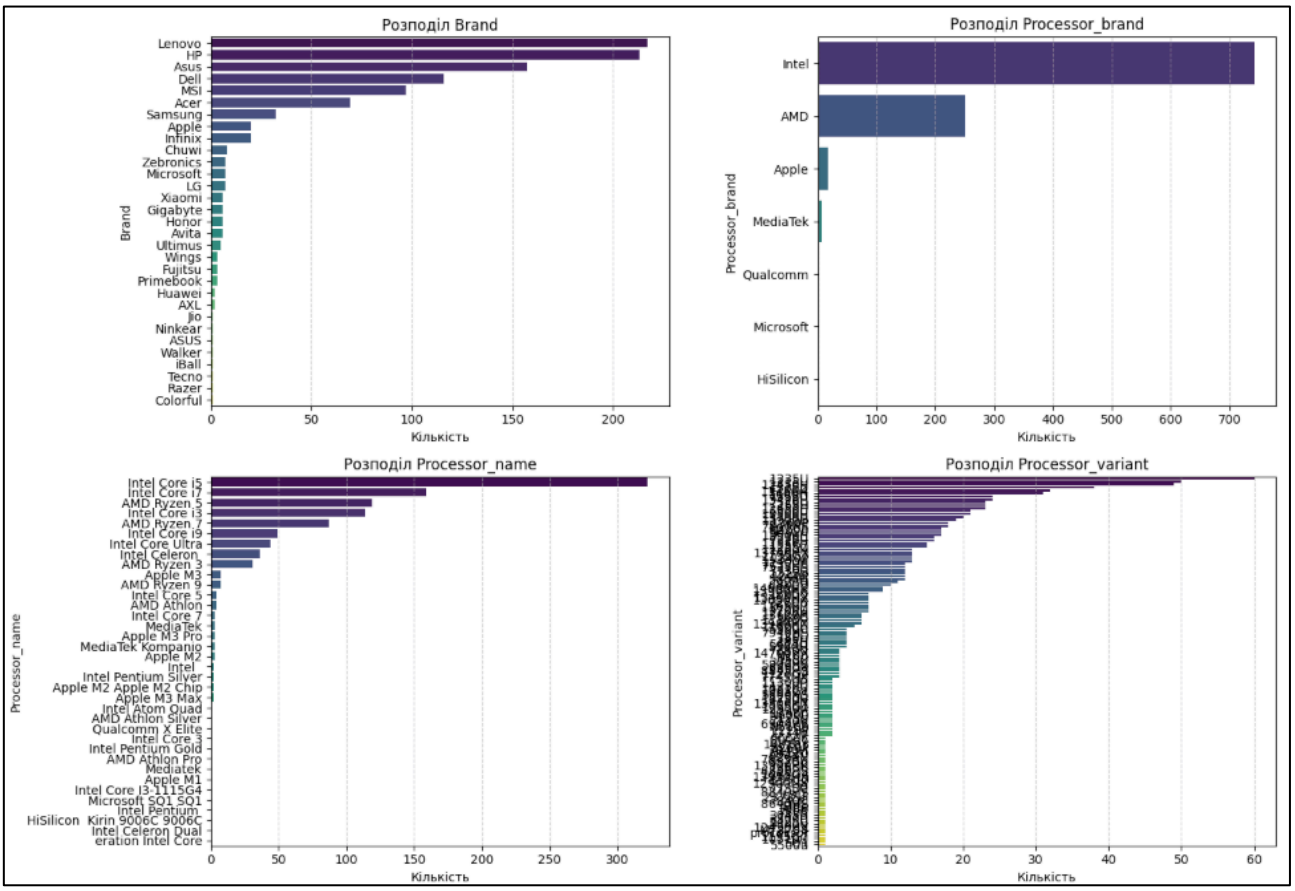

Рисунок Г.4 – Розподіл категоріальних ознак

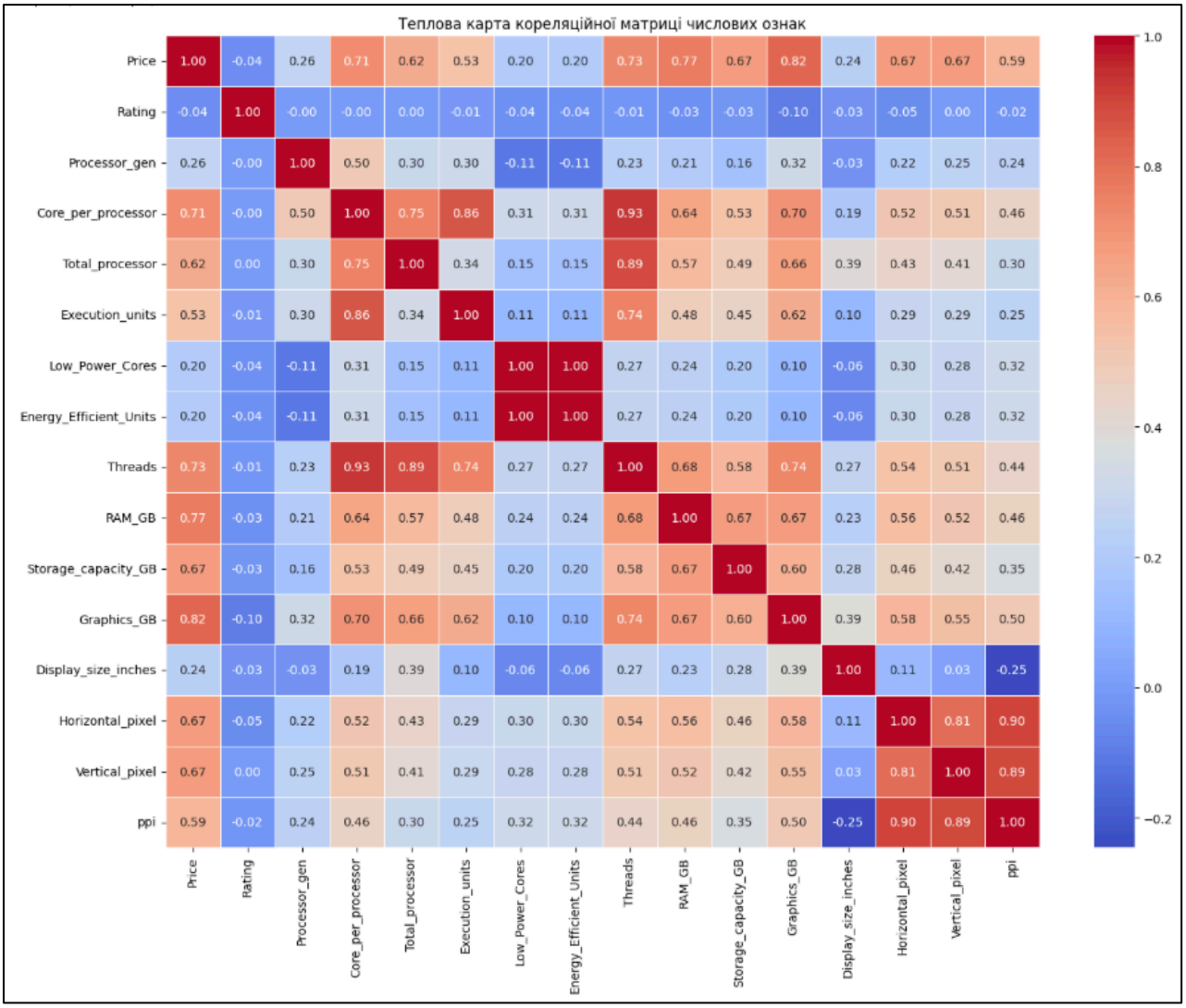

Рисунок Г.5 – Теплова карта кореляційної матриці

model_name	MAE_train	MAE_test	MSE_train	MSE_test	RMSE_trair	RMSE_test	R2_train	R2_test
Ridge Regression	0.09	0.13	0.01	0.04	0.14	0.22	0.97	0.85
Random Forest Regres	**0.09**	**0.14**	**0.01**	**0.04**	**0.14**	**0.21**	**0.94**	**0.87**
Gradient Boosting Regressc	0.09	0.14	0.01	0.04	0.15	0.21	0.97	0.86

Рисунок Г.6 – Результати прогнозування моделей після тюнінгу

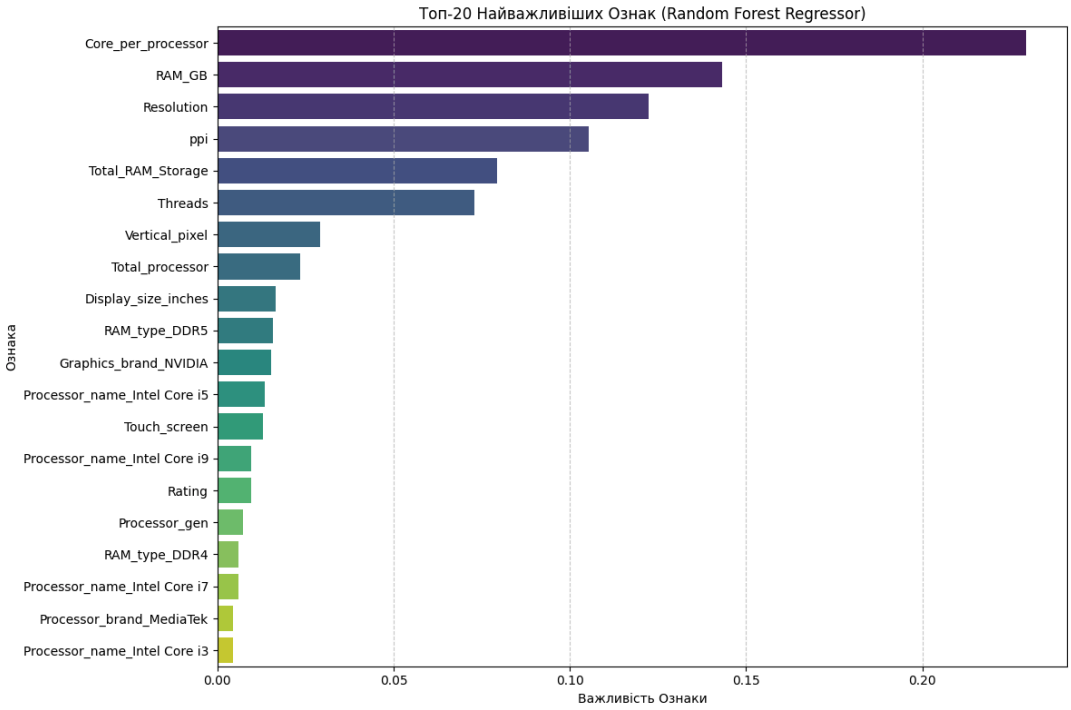

Рисунок Г.7 – Важливість ознак моделі Random Forest