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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія аналізу та передбачення ціни телефонів методами машинного навчання»

Виконав: студент 2 курсу, групи 2ІСТ-24м спеціальності 126 – «Інформаційні системи та технології»
_____ Олег НЕРЕУЦЬКИЙ

Керівник: к.т.н., доц. каф. САІТ
_____ Олексій КОЗАЧКО
«27» 11 _____ 2025 р.

Опонент: к.т.н., доц. каф. КН
_____ Юрій ПАНОЧИШИН
«03» 12 _____ 2025 р.

Допущено до захисту

Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
«28» 11 _____ 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 – другий (магістерський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 126 Інформаційні системи та технології
Освітньо-професійна програма – Інформаційні технології аналізу даних та зображень

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
д.т.н., проф. Віталій МОКІН
«25» 09 2025 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Нереуцькому Олегу Віталійовичу

1. Тема роботи: "Інформаційна технологія аналізу та передбачення ціни телефонів методами машинного навчання ",
керівник роботи: Олексій КОЗАЧКО, к.т.н., доц. каф. САІТ,
затверджені наказом ВНТУ від «24» 09 2025 року №313
2. Строк подання студентом роботи «28» 11 2025 року
3. Вихідні дані до роботи:
Kaggle Dataset "Mobile Price Classification":
https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification
4. Зміст текстової частини:
   - загальна характеристика об'єкту дослідження;
   - вибір оптимального рішення та налаштування для розв'язання поставленої задачі;
   - результати інтелектуального моделювання та передбачення;
   - економічна частина.
5. Перелік ілюстративного матеріалу:
   - box plots графіки розподілу основних ознак для різних цінових категорій;
   - кореляційна матриця;
   - результати передбачення моделей;
   - блок-схема алгоритму роботи інформаційної технологія.
   - UML-діаграма розгортання (Deployment Diagram)

6. Консультанти розділів МКР

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Олександр ЛЕСЬКО, к. е. н., проф. каф. ЕПВМ	30.11.24	30.11.

7. Дата видачі завдання «25» 09 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	15.09.	26.09.	вик
2	Вибір оптимальних інформаційних технологій	26.09	06.10	вик
3	Аналіз та обробка даних	06.10	17.10	вик
4	Розроблення інформаційної технології для передбачення ціни телефонів	17.10	26.10.	вик
5	Економічна частина	26.10	15.11	вик
6	Оформлення матеріалів до захисту МКР	15.11	25.11	вик

Студент _____ Олег НЕРЕУЦЬКИ

Керівник роботи _____ Олексій КОЗАЧКО

АНОТАЦІЯ

УДК 004.8:004.652

Нереуцький О. В. Інформаційна технологія аналізу та передбачення ціни телефонів методами машинного навчання. Магістерська кваліфікаційна робота зі спеціальності 126 – інформаційні системи та технології, освітньо-професійна програма – інформаційні технології аналізу даних та зображень. Вінниця: ВНТУ, 2025. 137 с.

На укр. мові. Бібліогр.: 23 назв; рис.: 69; табл.: 10.

В магістерській кваліфікаційній роботі проаналізовано можливість прогнозування цін на мобільні телефони з використанням методів машинного навчання на мові програмування Python та запропоновано підхід до вибору найбільш значущих ознак для підвищення точності моделей прогнозування. Об’єктом дослідження є процес аналізу характеристик мобільних телефонів та передбачення їхніх цінових категорій.

Ілюстративна частина магістерської роботи включає 10 плакатів, що відображають ключові етапи аналізу та передбачення цін на мобільні телефони.

У розділі економічної частини розглянуто питання доцільності розробки та впровадження інформаційної технології для прогнозування цін на мобільні телефони.

Ключові слова: інформаційна технологія, розвідувальний аналіз даних, передбачення цін, моделі, машинне навчання

ABSTRACT

Nereutsky, O. V. Information technology for analyzing and predicting phone prices using machine learning methods. Master's thesis in the field of 126 – Information Systems and Technologies, educational and professional program – Information Technologies for Data and Image Analysis. Vinnytsia: VNTU, 2025. 137 p.

In Ukrainian. Bibliography.: 23 titles; fig.: 69; table.: 10.

The master's thesis analyzes the possibility of predicting mobile phone prices using machine learning methods in the Python programming language and proposes an approach to selecting the most significant features to improve the accuracy of prediction models. The object of the study is the process of analyzing the characteristics of mobile phones and predicting their price categories.

The illustrative part of the master's thesis includes 10 posters reflecting the key stages of analysis and forecasting of mobile phone prices.

The economic section examines the feasibility of developing and implementing information technology for forecasting mobile phone prices.

Keywords: information technology, exploratory data analysis, price prediction, models, machine learning

ЗМІСТ

ВСТУП

Актуальність теми. У сучасній сфері інформаційних технологій прогнозування вартості мобільних телефонів набуває особливого значення, оскільки впливає на інтереси широкого кола учасників ринку — споживачів, виробників, продавців і інвесторів. Можливість передбачати зміну цін забезпечує низку важливих переваг: оптимізацію витрат, більш ефективне планування виробництва, формування маркетингових стратегій та підвищення конкурентоспроможності компаній.

Для кінцевих споживачів аналіз і розуміння цінових тенденцій дозволяє приймати обґрунтовані рішення щодо вибору моделі, часу покупки та отримання максимальної вигоди при придбанні пристрою. Такий підхід особливо актуальний в умовах стрімкого розвитку технологій, коли нові моделі смартфонів виходять на ринок практично щорічно.

Для виробників та ритейлерів здатність прогнозувати динаміку цін відкриває можливості для ефективного управління складськими запасами, планування випуску нових продуктів, призначення акційних пропозицій та адаптації до змін ринкового попиту й конкурентної ситуації. Сучасні аналітичні інструменти й алгоритми машинного навчання дозволяють опрацьовувати великі обсяги даних, ураховуючи широкий спектр чинників ціноутворення, таких як сезонність, економічні показники, ринкові тренди та поведінка покупців.

Для інвесторів володіння інформацією щодо можливих цінових коливань на мобільні телефони є важливим елементом прийняття інвестиційних рішень. Прогнозування ринкових трендів і вартості продукції допомагає оцінити потенційні ризики та очікувану дохідність від вкладень у технологічні компанії та стартапи.

Таким чином, передбачення цін на мобільні пристрої є важливим інструментом, що сприяє формуванню обґрунтованих та стратегічно вигідних рішень для всіх учасників ринку цифрової інфраструктури.

Мета і задачі дослідження. Основною метою даного дослідження є розробка інформаційної технології, яка дозволяє виконувати передбачення вартості мобільних телефонів.

Для досягнення поставленої мети необхідно розв’язати такі задачі:

– здійснити детальний аналіз об’єкта дослідження;

– визначити та обрати найбільш ефективні інформаційні технології;

– провести розвідувальний аналіз даних і виконати їх попередню обробку;

– сформувати та навчити моделі, спрямовані на передбачення цін;

– створити інформаційну технологію для передбачення вартості мобільних телефонів на основі методів машинного навчання.

Об’єктом дослідження є процес прогнозування вартості мобільних телефонів із застосуванням алгоритмів машинного навчання.

Предметом дослідження є методи машинного навчання та інформаційна технологія, що використовується для моделювання та прогнозування цін на мобільні телефони.

Наукова новизна роботи. Подальшого розвитку набула інформаційна технологія аналізу та передбачення ціни мобільних телефонів шляхом використання трьох моделей машинного навчання – Logistic Regression, SVM Classifier та MLP Classifier. У межах дослідження проведено оптимізацію їх гіперпараметрів, що дало змогу підвищити точність передбачення цін мобільних пристроїв. Запропонований підхід забезпечив удосконалення процесу оцінювання вартості мобільних телефонів та дав можливість визначити найбільш ефективну модель для вирішення поставленої задачі.

Публікації результатів магістерської кваліфікаційної роботи. Опубліковано тези на Всеукраїнській науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи» (Вінниця, 2025-2026 рр.) [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз об’єкта дослідження

Об’єктом даного дослідження є процес прогнозування вартості мобільних телефонів — багаторівнева, динамічна та складна система, яка складається з безлічі факторів, що прямо або опосередковано впливають на формування кінцевої ціни. Актуальність аналізу даного об’єкта зумовлена швидким розвитком ринку мобільних технологій, постійною появою нових моделей та високою конкуренцією між виробниками, що вимагає точного прогнозування коливань цін.

Прогнозування вартості смартфонів дає змогу глибше зрозуміти механізми ціноутворення, що є важливим для різних груп користувачів: виробників, дистриб’юторів, ритейлерів, споживачів та інвесторів. Часто керівники технологічних компаній ставлять питання: «Які чинники визначають ринкову ціну мобільного пристрою?» та «Як передбачити зміну цієї ціни?». Відповідь на ці питання потребує аналізу великої кількості характеристик, що можуть суттєво варіювати залежно від моделі, бренду, регіону продажу, економічної ситуації та навіть сезонних коливань [2].

Якщо ви займаєте керівну посаду у компанії, що спеціалізується на виробництві чи реалізації мобільних телефонів, або працюєте в галузі роздрібної торгівлі, важливо мати хоча б загальне уявлення про чинники, які визначають формування вартості цих пристроїв. Часто можна почути, що технології, дизайн і маркетинг визначають усе, і в більшості випадків це відповідає дійсності. Проте це лише частина загальної картини.

На вартість мобільного телефону впливає багато різних чинників. Деякі з них мають суттєвіший ефект, ніж інші, а їхній вплив може відрізнятися залежно від регіону. Проте в кінцевому підсумку кожен із цих факторів певним чином впливає на ціну.

Тож розглянемо докладніше основні аспекти, які впливають на формування вартості мобільного телефону. Наведений список не є вичерпним, однак він містить ключові фактори, що мають найбільший вплив на ціну пристрою:

1. Технічні характеристики. Одним із головних чинників є апаратна складова пристрою. Збільшений обсяг оперативної пам’яті, високопродуктивні процесори, сучасні камери та інноваційні технології безпосередньо підвищують собівартість і, відповідно, роздрібну ціну смартфона.

2. Дизайн і матеріали корпуса. Використання преміальних матеріалів (скло, метал), нестандартні конструктивні рішення та привабливий дизайн зазвичай збільшують ціну пристрою, оскільки потребують складніших технологічних процесів і дорожчої сировини.

3. Маркетингова політика. Активні рекламні кампанії, позиціонування бренду та формування його іміджу значною мірою впливають на ціну. Відомі бренди мають можливість встановлювати вищу вартість завдяки довірі споживачів і стабільному попиту [2].

4. Функціональні особливості. Наявність додаткових можливостей, таких як захист від вологи, підтримка 5G, бездротова зарядка чи розширені функції безпеки, збільшує цінність пристрою та підвищує його вартість на ринку.

5. Рівень конкуренції. В умовах сильного конкурентного тиску виробники можуть знижувати ціни для збереження позицій на ринку. Особливо це помітно у сегментах, де кілька моделей мають дуже схожі характеристики.

6. Економічні умови. Курсові коливання, інфляція, зміна вартості імпортних комплектуючих та інші макроекономічні чинники здатні істотно впливати на кінцеву ціну смартфонів.

7. Сезонність. У певні періоди року (передсвяткові акції, вихід нових моделей, сезонні розпродажі) ціни на смартфони можуть тимчасово

змінюватися, що обумовлюється ринковим попитом і маркетинговими стратегіями.

8. Регіональні особливості. Податки, мита, вартість логістики, рівень доходів населення та локальна конкуренція можуть суттєво відрізнятися в різних країнах і регіонах, що призводить до різниці в кінцевих цінах.

9. Вартість досліджень і розробок. Інвестиції у нові технології та інновації можуть спричиняти підвищення цін на смартфони. Виробники, які активно розробляють передові рішення, зазвичай закладають ці витрати у кінцеву вартість продукції.

10. Маркетингові пропозиції та програми лояльності. Наявність знижок, бонусів та спеціальних пропозицій може впливати на ціну. Програми лояльності та акційні кампанії роблять пристрої більш доступними для споживачів.

11. Екологічні ініціативи. Виробники, які впроваджують екологічні практики — наприклад, використовують перероблені матеріали або зменшують вуглецевий слід — можуть стикатися з вищими виробничими витратами, що впливає на ціну кінцевого продукту [2].

Перелічені вище чинники становлять лише частину основних аспектів, що впливають на формування вартості мобільного телефону. Розуміння їхнього впливу дає змогу виробникам, продавцям і покупцям ухвалювати більш обґрунтовані рішення щодо виготовлення, продажу та придбання таких пристроїв.

1.2 Аналіз методів прогнозування

Мова програмування Python пропонує широкий вибір інструментів і алгоритмів для прогнозування, які можна застосовувати до різноманітних типів даних та аналітичних задач. З огляду на те, що ключовою метою даної роботи є моделювання та передбачення вартості мобільних телефонів, важливо підібрати такі методи, які забезпечать максимальну точність

результатів. Нижче подано основні підходи до прогнозування, що можуть бути використані у Python:

1. Випадковий ліс (Random Forest) – ансамблевий підхід, що формує велику кількість дерев рішень, кожне з яких створюється на основі випадкової підвибірки даних та ознак. Остаточний прогноз отримують шляхом голосування (у класифікації) або усереднення результатів (у регресії). Метод вирізняється стійкістю до перенавчання та високим рівнем точності [3].

2. XGBoost (Extreme Gradient Boosting) – один із найефективніших алгоритмів градієнтного бустингу, який застосовується для класифікаційних та регресійних задач. Він об’єднує набір слабких дерев рішень у єдину сильну модель. XGBoost здобув популярність завдяки високій продуктивності та точності [4].

3. AdaBoost – ансамблевий метод, що послідовно формує слабкі моделі, кожна з яких зосереджується на прикладах, помилково класифікованих попередніми моделями. Такий підхід дозволяє значно зменшити похибку та підвищити точність алгоритму [5].

4. Extremely Randomized Trees (Extra Trees) – різновид ансамблевих дерев рішень, у якому вибір порогів для поділу вузлів виконується випадково, без оптимізації. Це сприяє збільшенню різноманітності дерев у моделі та підвищує її здатність до узагальнення, одночасно зменшуючи коливання прогнозованих результатів [6].

5. Логістична регресія (Logistic Regression) – алгоритм класифікації, що застосовує логістичну функцію для оцінки ймовірності належності об’єкта до певного класу. Підходить для бінарної класифікації та може бути адаптована до багатокласових задач. Модель проста в реалізації та ефективна для лінійно розділених даних. [7]

6. K-Nearest Neighbors (KNN) – метод, який базується на пошуку найближчих сусідів для визначення класу або числового прогнозу. Для класифікації обирається клас, найбільш поширений серед K сусідів, для регресії – середнє або медіанне значення. Метод вирізняється простотою та

зрозумілістю [8].

7. Decision Trees (Дерева рішень) – один із базових алгоритмів машинного навчання, що використовується для задач класифікації та регресії. Принцип роботи полягає в рекурсивному поділі даних на підгрупи на основі умовних правил, сформованих за ознаками. Листові вузли дерева дозволяють отримати прогноз для нових даних [9].

Дерева рішень легко інтерпретуються, дають змогу визначити важливість ознак і можуть обробляти як числові, так і категоріальні дані.

8. Light Gradient Boosting Machine (LightGBM) – високопродуктивний алгоритм градієнтного бустингу, розроблений компанією Microsoft. Вирізняється швидкістю та ефективністю, що робить його зручним для роботи з великими масивами даних. LightGBM підтримує категоріальні ознаки та забезпечує високу точність прогнозування [10].

Основними перевагами цього методу є швидка обробка великих обсягів даних, висока точність прогнозів, особливо на великих наборах даних, а також можливість роботи з категоріальними змінними без необхідності їх попереднього кодування.

Загалом, LightGBM представляє собою ефективний та продуктивний алгоритм, придатний для вирішення широкого спектра задач класифікації та регресії.

9. Gradient Boosting – ансамблевий метод, заснований на поетапному навчанні слабких моделей, які покращують результати попередніх шляхом коригування їхніх помилок. Головною метою є мінімізація похибки за рахунок поступового вдосконалення моделі [11].

До ключових переваг методу належать гнучкість, точність та здатність працювати з різноманітними типами даних.

10. Support Vector Machines – це алгоритм машинного навчання, який застосовується для вирішення задач класифікації та регресії. Основний принцип його роботи полягає у розташуванні даних на гіперплощині та їх розділенні за допомогою ліній або площин, що дозволяє визначити належність

об’єктів до певних класів. Коли надходять нові дані, модель використовує попередньо побудоване розділення на гіперплощині для виконання класифікації [12].

SVM особливо ефективний при роботі з невеликими, але різноманітними наборами даних. Алгоритм характеризується високою точністю та гнучкістю, однак має порівняно високу обчислювальну складність і чутливість до наявності аномальних даних.

11. Multi-Layer Perceptron, MLP – це поширений метод машинного навчання, відомий також як нейронна мережа з повною взаємною зв’язністю між шарами. Алгоритм використовує штучну нейронну мережу з кількома шарами для виконання задач класифікації та прогнозування.

Структура MLP включає вхідний шар, один або декілька прихованих шарів та вихідний шар. Вхідний шар приймає дані для обробки, приховані шари здійснюють обчислення та трансформації сигналів, а на вихідному шарі формується кінцевий прогноз на основі обробленої інформації. Основна мета алгоритму – мінімізувати помилку класифікації, тобто зменшити розбіжність між прогнозованими та фактичними класами.

До ключових переваг методу належать здатність виявляти складні й неочевидні залежності між даними, а також висока ефективність у задачах регресії та класифікації. Серед недоліків – значні вимоги до обчислювальних ресурсів і можливість перенавчання при недостатньому обсязі даних [13].

1.3 Вибір оптимальних інформаційних технологій для прогнозування

Оскільки ключовою метою даного дослідження є прогнозування вартості мобільних телефонів із застосуванням методів машинного навчання, найбільш доцільним вибором інструменту є мова програмування Python.

Python, розроблений Гвідо ван Россумом та вперше представлений у 1991 році, є високорівневою інтерпретованою об’єктно-орієнтованою мовою з

динамічною семантикою. Її назва походить від британського комедійного шоу Monty Python, що підкреслює прагнення автора зробити мову простою, зручною та приємною у використанні. Python відомий своєю доступністю для новачків, адже дає змогу зосередитися на концепціях програмування, не відволікаючись на складні технічні деталі, характерні для багатьох інших мов [14].

Ця мова має широке застосування у веброзробці, створенні програмного забезпечення, математичних обчисленнях, аналізі даних та автоматизації процесів. Її популярність зумовлена можливістю швидкої розробки застосунків, а також зручністю інтеграції з різними технологіями та мовами програмування. Завдяки розвинутим вбудованим структурам даних, динамічній типізації та гнучкій архітектурі, Python забезпечує простий, швидкий та ефективний процес розробки. Зрозумілий синтаксис і висока читабельність коду істотно зменшують витрати на його супровід. Крім того, підтримка модулів і пакетів спрощує створення модульних застосунків і сприяє повторному використанню коду. Python є мовою з відкритим вихідним кодом, що стимулює активний розвиток, підтримку та розширення бібліотек з боку світової спільноти розробників.

Приклади застосування Python:

- Розробка серверних веб-застосунків;
- Створення робочих процесів, що можуть інтегруватися з іншим програмним забезпеченням;
- Побудова веб-додатків, які функціонують на стороні сервера;
- Автоматизація різноманітних процесів і написання службових скриптів;
- Організація взаємодії та керування базами даних;
- Аналіз та обробка даних і різних типів файлів;
- Виконання складних математичних статистичних розрахунків;
- Опрацювання великих масивів даних;
- Створення програмних рішень для промислових та виробничих

систем.

На професійному рівні Python широко застосовується у веброзробці, аналізі даних, створенні систем AI та виконанні наукових обчислень. Крім того, мова використовується для розробки інструментів продуктивності, ігор та настільних застосунків.

Переваги Python:

- Підтримка роботи на різних операційних системах, включаючи Windows, macOS, Linux, Raspberry Pi та інші платформи;
- Лаконічний і зрозумілий синтаксис, який дозволяє створювати менш об’ємний код у порівнянні з багатьма іншими мовами програмування;
- Інтерпретований характер мови, що забезпечує можливість негайного виконання коду та значно спрощує процес швидкого створення прототипів;
- Підтримка процедурного, об’єктно-орієнтованого та функціонального стилів програмування.

Висока гнучкість Python дає змогу уникати жорстких обмежень під час створення функцій та обирати різні підходи для розв’язання задач. Мова також забезпечує можливість виконання програм у складних або спеціалізованих середовищах завдяки динамічній перевірці типів під час роботи програми.

Python та штучний інтелект (AI): Python є одним з найпоширеніших інструментів для досліджень і розробки у сфері штучного інтелекту. Такі бібліотеки, як scikit-learn, Keras та TensorFlow, формують потужну базу для створення рішень у галузі AI, забезпечуючи розробникам зручність, масштабованість та гнучкість. Використання цих бібліотек дозволяє зосередитися на інноваціях і вдосконаленні моделей, не витрачаючи зайвих ресурсів на реалізацію низькорівневих алгоритмів.

Індекс пакетів Python (PyPI) – це офіційне сховище програмного забезпечення для екосистеми Python, яке дозволяє користувачам знаходити, встановлювати та оновлювати пакунки, створені спільнотою розробників [15].

1.4 Висновки

У цьому розділі було здійснено аналіз об’єкта дослідження та обґрунтовано актуальність задачі прогнозування вартості мобільних телефонів із застосуванням методів машинного навчання. Наведено опис і проведено порівняльний аналіз різних алгоритмів прогнозування, на основі чого було визначено найбільш придатні моделі для розв’язання поставленої задачі. До таких методів віднесено Random Forest, XGBoost, AdaBoost, Extra Trees, Logistic Regression, SVM Classifier та MLP Classifier.

2 АНАЛІЗ ТА ОБРОБКА ДАНИХ

2.1 Підготовка даних

Перед початком роботи необхідно здійснити підготовку даних та виконати імпорт відповідних бібліотек. Першим етапом стало підключення всіх необхідних бібліотек, які забезпечують виконання поставленого завдання (рис. 2.1-2.2).

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import pickle
import xgboost as xgb
from mpl_toolkits import mplot3d
import matplotlib.style as style
import matplotlib.gridspec as gridspec
import matplotlib.colors as colors
from scipy import stats
import plotly.graph_objs as go
from plotly.subplots import make_subplots
from matplotlib import colors
from sklearn.model_selection import cross_val_score
from matplotlib.colors import ListedColormap, LinearSegmentedColormap
from sklearn.model_selection import train_test_split, GridSearchCV, StratifiedKFold, cross_val_sc
ore
```

Рисунок 2.1 – Перелік основних бібліотек

```
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, ExtraTreesClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from xgboost import XGBClassifier
from sklearn.metrics import precision_score, recall_score, accuracy_score, r2_score, f1_score, confusion_matrix,

from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV

import warnings
warnings.filterwarnings("ignore")
```

Рисунок 2.2 – Набір бібліотек призначених для візуалізації моделей

На наступному етапі було здійснено завантаження набору даних та отримано загальну інформацію про його структуру. Додатково визначено кількість наявних категоріальних і числових змінних, що є важливим для подальшої підготовки даних та вибору оптимальних методів обробки (рис. 2.3-2.5).

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   battery_power  2000 non-null   int64  
 1   blue           2000 non-null   int64  
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64  
 4   fc             2000 non-null   int64  
 5   four_g         2000 non-null   int64  
 6   int_memory     2000 non-null   int64  
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64  
 9   n_cores        2000 non-null   int64  
 10  pc             2000 non-null   int64  
 11  px_height      2000 non-null   int64  
 12  px_width       2000 non-null   int64  
 13  ram            2000 non-null   int64  
 14  sc_h           2000 non-null   int64  
 15  sc_w           2000 non-null   int64  
 16  talk_time      2000 non-null   int64  
 17  three_g        2000 non-null   int64  
 18  touch_screen   2000 non-null   int64  
 19  wifi           2000 non-null   int64  
 20  price_range    2000 non-null   int64  
dtypes: float64(2), int64(19)
memory usage: 328.2 KB
```

Рисунок 2.3 – Основні відомості щодо набору даних

	Number of Unique Values	Unique Values
price_range	4	[1, 2, 3, 0]
n_cores	8	[2, 3, 5, 6, 1, 8, 4, 7]
blue	2	[0, 1]
dual_sim	2	[0, 1]
four_g	2	[0, 1]
three_g	2	[0, 1]
touch_screen	2	[0, 1]
wifi	2	[1, 0]

Рисунок 2.4 – Структура категоріальних змінних

	count	mean	std	min	25%	50%	75%	max
battery_power	2000.0	1238.5	439.4	501.0	851.8	1226.0	1615.2	1998.0
clock_speed	2000.0	1.5	0.8	0.5	0.7	1.5	2.2	3.0
fc	2000.0	4.3	4.3	0.0	1.0	3.0	7.0	19.0
int_memory	2000.0	32.0	18.1	2.0	16.0	32.0	48.0	64.0
m_dep	2000.0	0.5	0.3	0.1	0.2	0.5	0.8	1.0
mobile_wt	2000.0	140.2	35.4	80.0	109.0	141.0	170.0	200.0
pc	2000.0	9.9	6.1	0.0	5.0	10.0	15.0	20.0
px_height	2000.0	645.1	443.8	0.0	282.8	564.0	947.2	1960.0
px_width	2000.0	1251.5	432.2	500.0	874.8	1247.0	1633.0	1998.0
ram	2000.0	2124.2	1084.7	256.0	1207.5	2146.5	3064.5	3998.0
sc_h	2000.0	12.3	4.2	5.0	9.0	12.0	16.0	19.0
sc_w	2000.0	5.8	4.4	0.0	2.0	5.0	9.0	18.0
talk_time	2000.0	11.0	5.5	2.0	6.0	11.0	16.0	20.0

Рисунок 2.5 – Структура числових змінних

Набір даних складається з 2000 прикладів, що описуються 21 змінною, з яких 20 є незалежними, а одна — залежною. Структура включає 8 категоріальних та 13 числових ознак. Нижче навдено розшифрування абревіатур доступних змінних:

– Battery_power – максимальний обсяг енергії, яку акумулятор здатен

утримувати за один цикл зарядки (mAh);

- Blue – наявність підтримки Bluetooth;
- Clock_speed – частота, з якою процесор виконує інструкції;
- Dual_sim – підтримка двох SIM-карт;
- Fc – роздільна здатність фронтальної камери (Мп);
- Four_g – наявність підтримки 4G;
- Int_memory – внутрішня пам'ять у гігабайтах;
- M_dep – товщина (глибина) корпусу;
- Mobile_wt – вага мобільного пристрою;
- N_cores – кількість процесорних ядер;
- Pc – роздільна здатність основної камери (Мп);
- Px_height – висота екранної роздільної здатності;
- Px_width – ширина екранної роздільної здатності;
- Ram – обсяг оперативної пам’яті (МБ);
- Sc_h – висота дисплея (см);
- Sc_w – ширина дисплея (см);
- Talk_time – максимальний час роботи пристрою від одного заряду;
- Three_g – наявність підтримки 3G;
- Touch_screen – наявність сенсорного екрану;
- Wifi – підтримка wifi;
- Price_range – це цільова змінна зі значеннями 0 (низький клас), 1 (середній клас), 2 (високий клас) та 3 (преміальний ціновий сегмент).

Наступним кроком було побудовано розподіл залежної змінної, що зображено на рисунку 2.6.

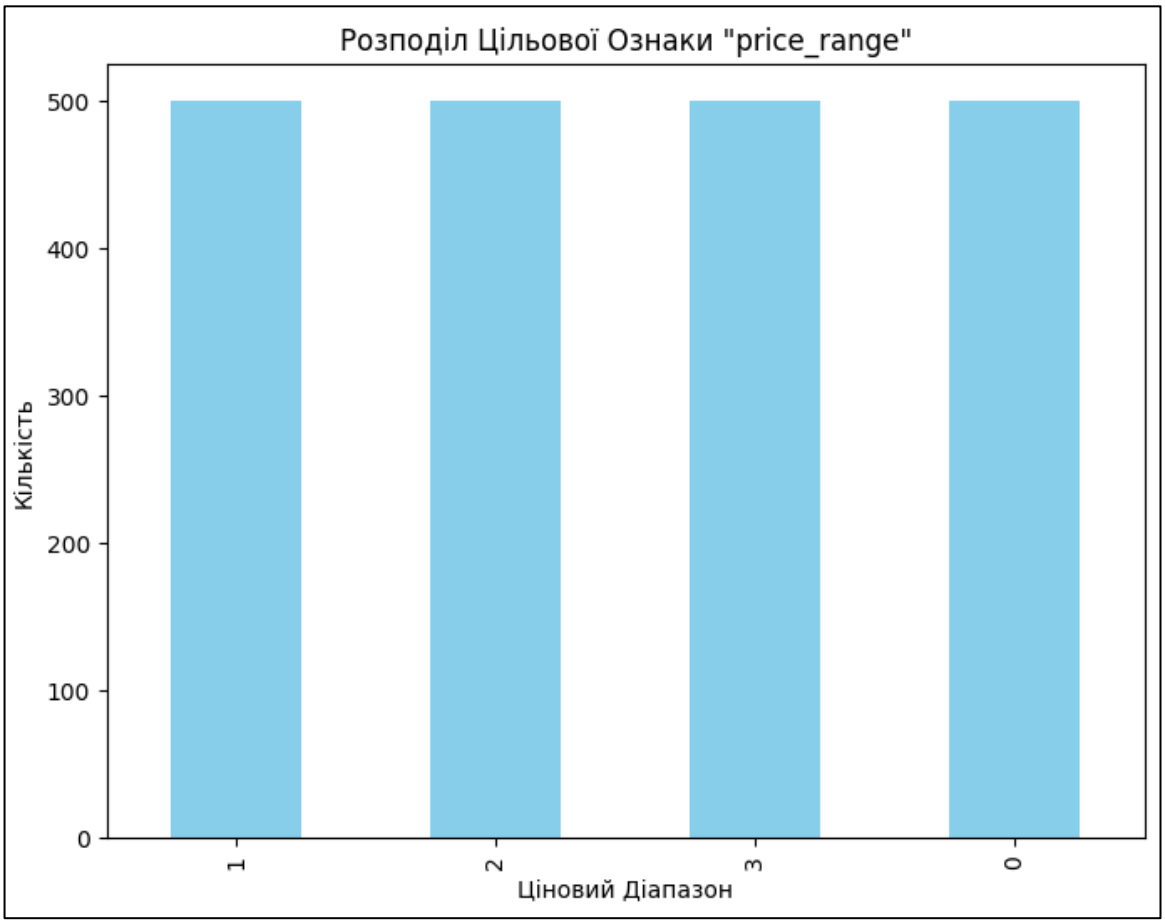


Рисунок 2.6 – Розподіл цільової ознаки price_range

Наведений графік залежної змінної чітко демонструє, що спостереження рівномірно розподілені між усіма ціновими категоріями. Кожен із діапазонів (0, 1, 2, 3) містить приблизно по 500 записів, що свідчить про збалансованість розподілу змінної price_range та відсутність домінування будь-якої категорії.

2.2 Розвідувальний аналіз даних

Розвідувальний аналіз даних (Exploratory Data Analysis, EDA) – це підхід, який застосовують фахівці з обробки даних для детального ознайомлення з набором даних перед початком побудови моделей. Іноді його також називають дослідженням даних. Основною метою EDA є виявлення основних характеристик та структури набору даних. Використання цього методу дозволяє аналітикам робити обґрунтовані припущення та

прогнозування щодо досліджуваних даних. Процес EDA зазвичай включає візуалізацію даних, наприклад, у вигляді гістограм, діаграм розсіювання та boxplot-діаграм. Метод дозволяє визначати взаємозв’язки між змінними у випадках, коли апріорна інформація про ці зв’язки відсутня або є неповною.

У процесі розвідувального аналізу зазвичай вивчають велику кількість змінних та застосовують різні методи для виявлення прихованих закономірностей у даних.

Головні підходи розвідувального аналізу даних включають:

– визначення ключових характеристик і структури набору даних;

– вибір найбільш значущих змінних для подальшого аналізу;

– виявлення аномалій та відхилень від очікуваних значень;

– перевірка початкових гіпотез щодо даних;

– розробка початкової або базової моделі для подальшого прогнозування.

На початковому етапі розвідувального аналізу було побудовано інтерактивні кругові діаграми для візуалізації розподілу категоріальних змінних (рис 2.7).

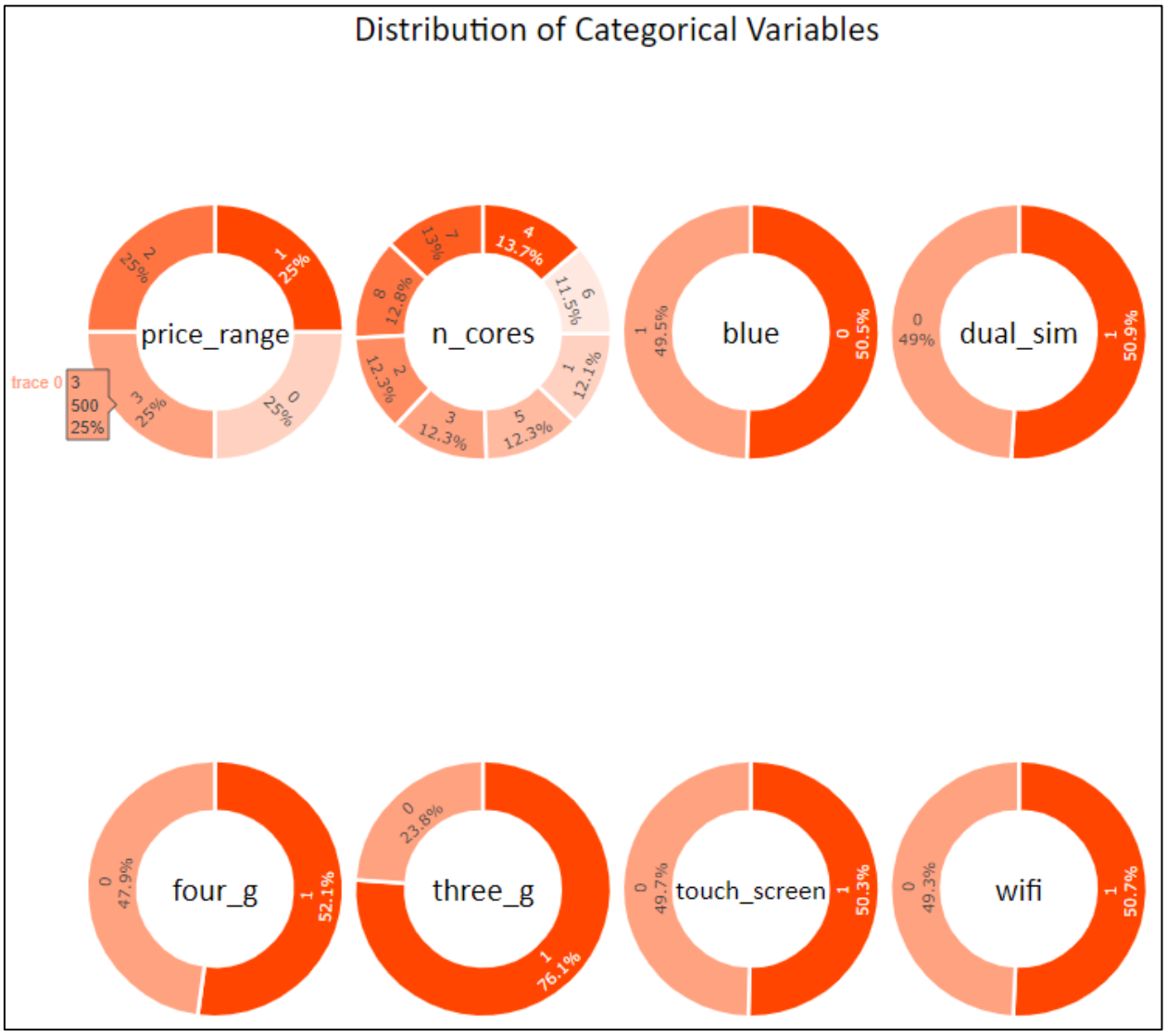


Рисунок 2.7 – Візуалізація розподілу категоріальних змінних за допомогою інтерактивних кругових діаграм

Наведений вище аналіз інтерактивних кругових діаграм категоріальних змінних чітко показує, що мобільні телефони рівномірно розподілені між чотирма категоріями змінної price_range, що свідчить про збалансованість набору даних.

Крім того, спостерігається майже рівномірний розподіл телефонів за наявністю або відсутністю таких характеристик, як Bluetooth, 4G, сенсорний екран, Wi-Fi, підтримка двох SIM-карт та кількість процесорних ядер. При цьому близько 76% моделей підтримують мережу 3G.

На наступному етапі проведено побудову гістограм для аналізу розподілу числових змінних (рис 2.8-2.10)

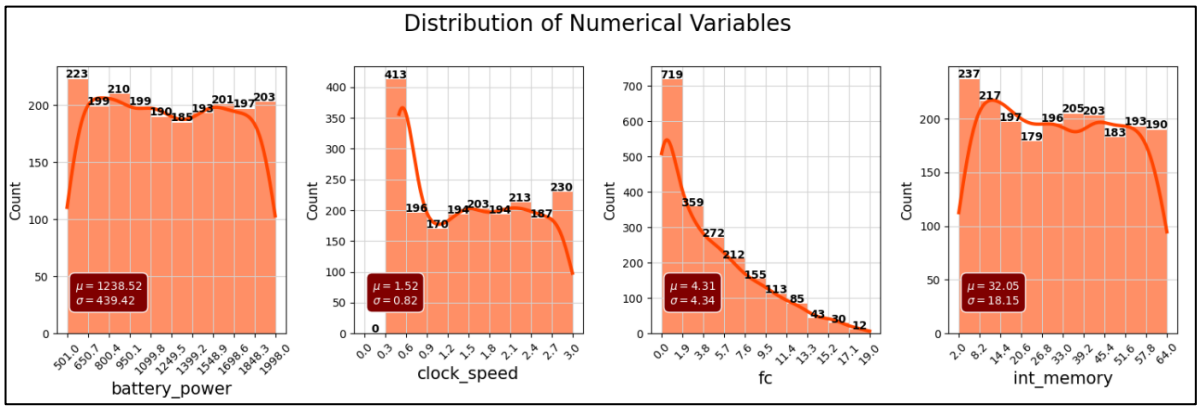


Рисунок 2.8 – Гістограми, що відображають розподіл перших чотирьох числових змінних

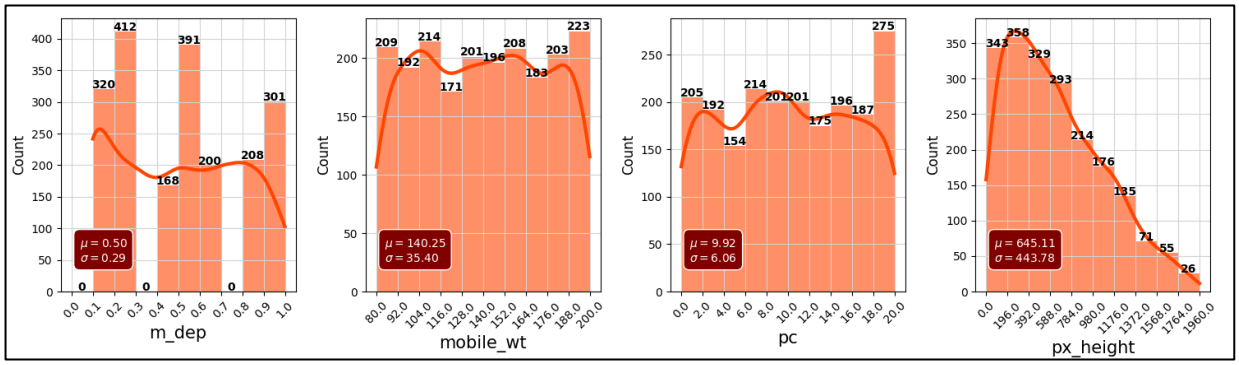


Рисунок 2.9 – Гістограми, які демонструють розподіл наступних чотирьох числових змінних

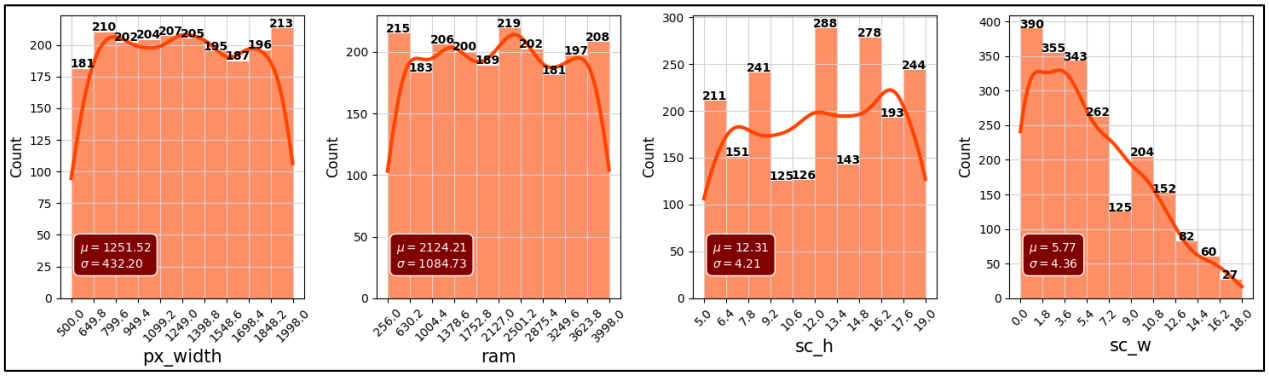


Рисунок 2.10 – Гістограми, що відображають розподіл останніх числових змінних

Аналізуючи наведені графіки вище, можна легко визначити ключові статистичні характеристики кожної ознаки, такі як мінімальні та максимальні значення, середнє та стандартне відхилення. Водночас деякі параметри, наприклад Px_height (висота екрана в пікселях) та Sc_w (ширина екрану в сантиметрах), мають значну кількість значень, близьких до нуля, що може свідчити про можливу наявність шуму або аномальних даних у наборі.

Також у процесі розвідувального аналізу були побудовані три види графіків — точкова діаграма (scatter plot), гістограма (histogram) та коробкова діаграма (box plot), що дозволяють наочно оцінити розподіл даних і взаємозв'язки між змінними для аналізу цінових категорій.

Співвідношення обсягу акумулятора та цінової групи наведено на рисунку 2.11.

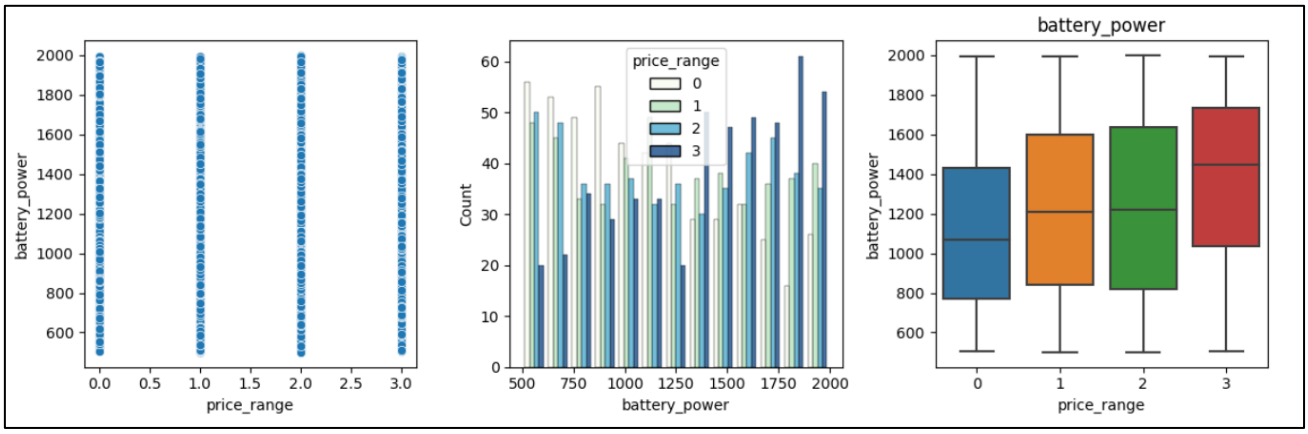


Рисунок 2.11 – Співвідношення обсягу акумулятора та цінової групи

З аналізу графіків можна зробити висновок, що із зростанням цінового сегмента збільшується і потужність батареї. У більш дорогих моделей середній обсяг акумулятора значно вищий, що вказує на тенденцію встановлення більш ємних батарей у преміальних пристроях. Це підтверджується як розподілом даних на скатерплоті та гістограмі, так і показниками медіани та міжквартильних інтервалів на boxplot-графіку.

Співвідношення функції Bluetooth та цінової групи зображено на рисунку 2.12.

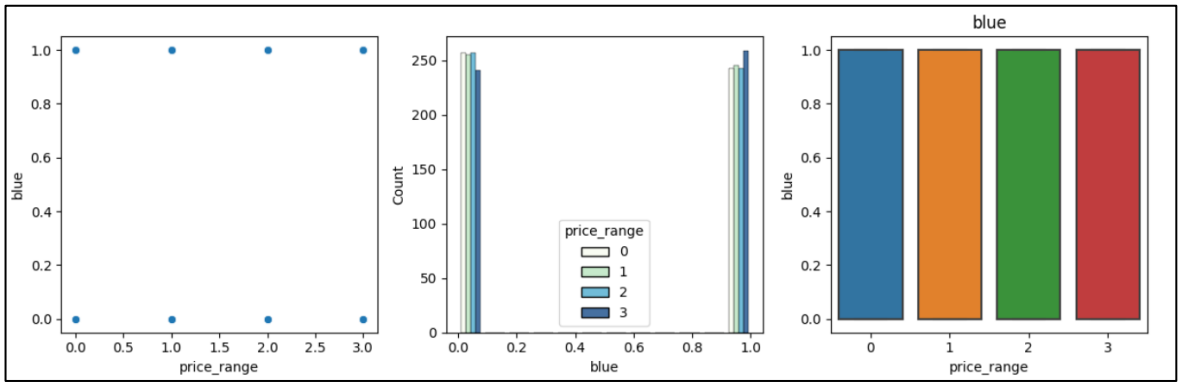


Рисунок 2.12 – Співвідношення функції Bluetooth та цінової групи

З графіків видно, що наявність Bluetooth не залежить від цінової категорії. Телефони з цією функцією зустрічаються приблизно однаково часто у всіх сегментах, що свідчить про відсутність явної кореляції між Bluetooth та ціною.

Співвідношення частоти процесора та цінової групи показана на рисунку 2.13.

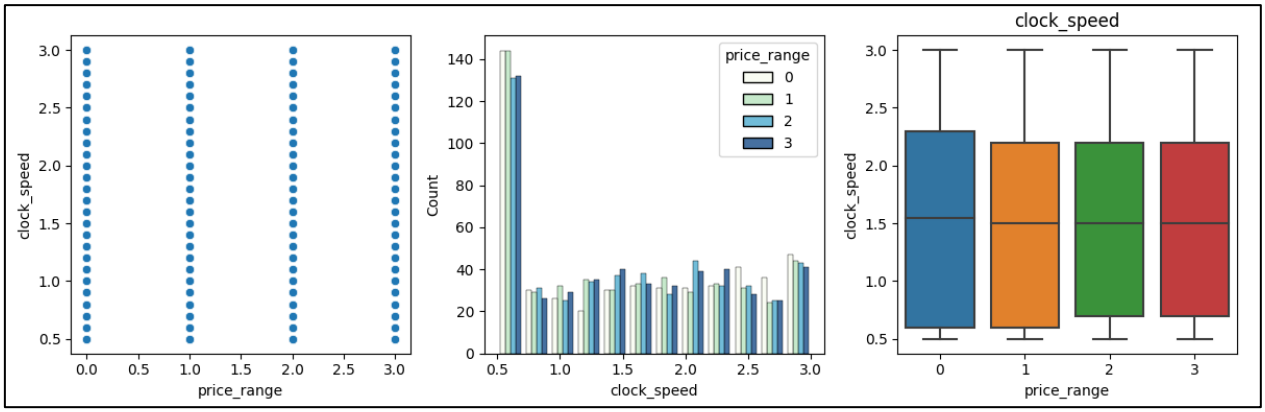


Рисунок 2.13 – Співвідношення частоти процесора та цінової групи

Наведені графіки показують, що між частотою роботи мікропроцесора та ціною смартфона прослідковується пряма залежність. Це означає, що моделі з вищою продуктивністю процесора зазвичай належать до дорожчого цінового сегмента.

Співвідношення кількості SIM та цінової групи рисунок 2.14.

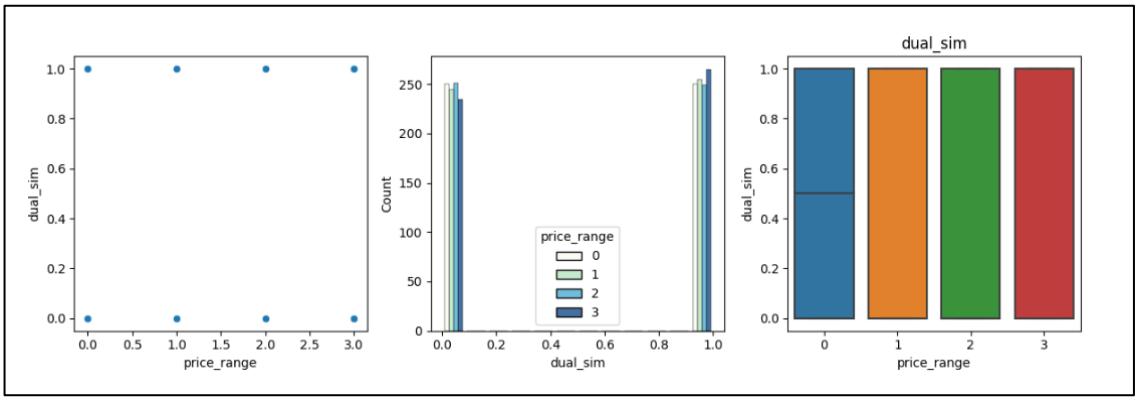


Рисунок 2.14 – Співвідношення кількості SIM та цінової групи

Вище продемонстровані графіки показують, що мобільні телефони з підтримкою двох SIM-карт зазвичай мають дещо вищу ціну порівняно з моделями, які оснащені лише однією SIM-карткою. Така різниця може пояснюватися додатковими можливостями та зручністю, які забезпечують пристрої з двома SIM-картами.

Співвідношення кількості фронтальних камер та цінової групи продемонстровано на рисунку 2.15.

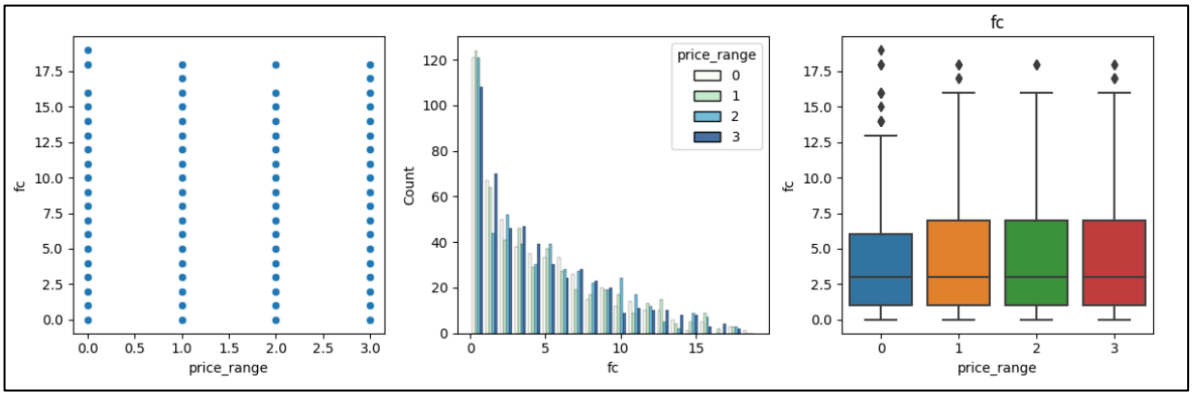


Рисунок 2.15 – Співвідношення кількості фронтальних камер та цінової групи

З представлених графіків видно, що кількість фронтальних камер у смартфонах варіюється залежно від їхнього цінового сегмента. Дорожчі моделі, ймовірно, частіше оснащуються більшою кількістю фронтальних

камер, що дозволяє забезпечити вищу якість селфі та відеозв'язку й таким чином краще задовольняти потреби користувачів.

Співвідношення внутрішньої пам'яті та цінової групи зображено на рисунку 2.16.

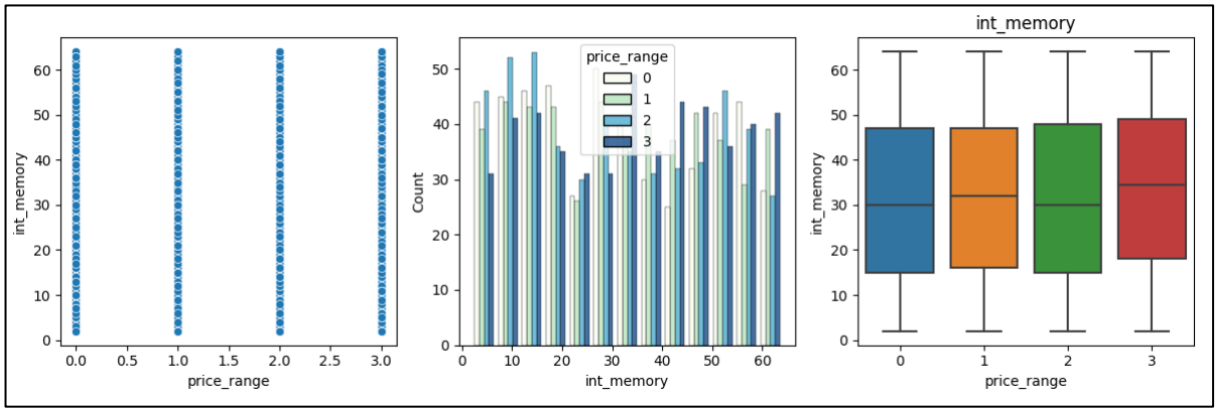


Рисунок 2.16 – Співвідношення внутрішньої пам'яті та цінової групи

Аналіз наведених графіків свідчить, що обсяг внутрішньої пам'яті має помітний вплив на ціну мобільних телефонів. Це може бути пов'язано як з особливостями окремих моделей, так і з різними підходами виробників до формування функціональності та позиціонування своїх пристроїв у певних цінових категоріях.

Співвідношення товщини смартфона та цінової групи наведено на рисунку 2.17.

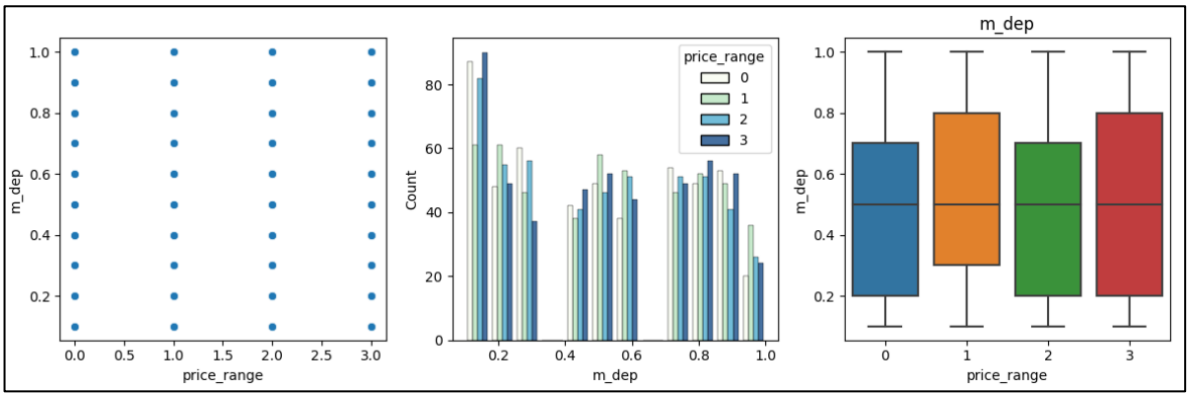


Рисунок 2.17 – Співвідношення товщини смартфона та цінової групи

Продемонстровані графіки ілюструють, як товщина смартфона змінюється залежно від його цінового сегмента. Це свідчить про те, що вартість може впливати на конструктивні особливості та дизайн пристрою. Зокрема, дорожчі моделі часто мають тонший і більш витончений корпус, що підкреслює їхню естетику та підвищує привабливість для покупців.

Співвідношення ваги смартфона та цінової групи рисунок 2.18.

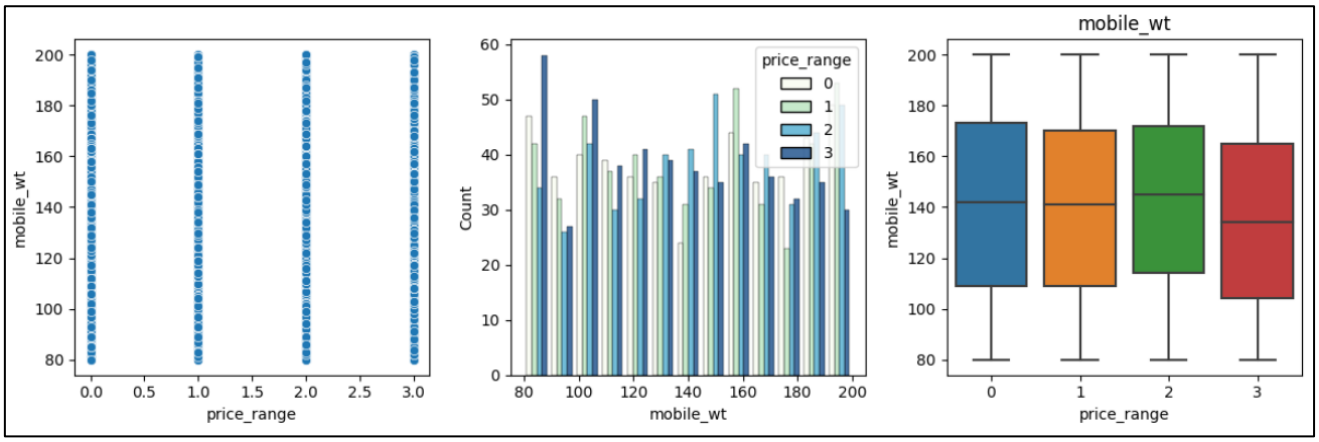


Рисунок 2.18 – Співвідношення ваги смартфона та цінової групи

Ці графіки ілюструють взаємозв’язок між вагою смартфона та його ціною. Спостерігається тенденція, що дорожчі моделі, як правило, мають меншу вагу, що підвищує їхню ергономічність та привабливість для користувачів.

Співвідношення кількості ядер процесора та цінової групи наведено на рисунку 2.19.

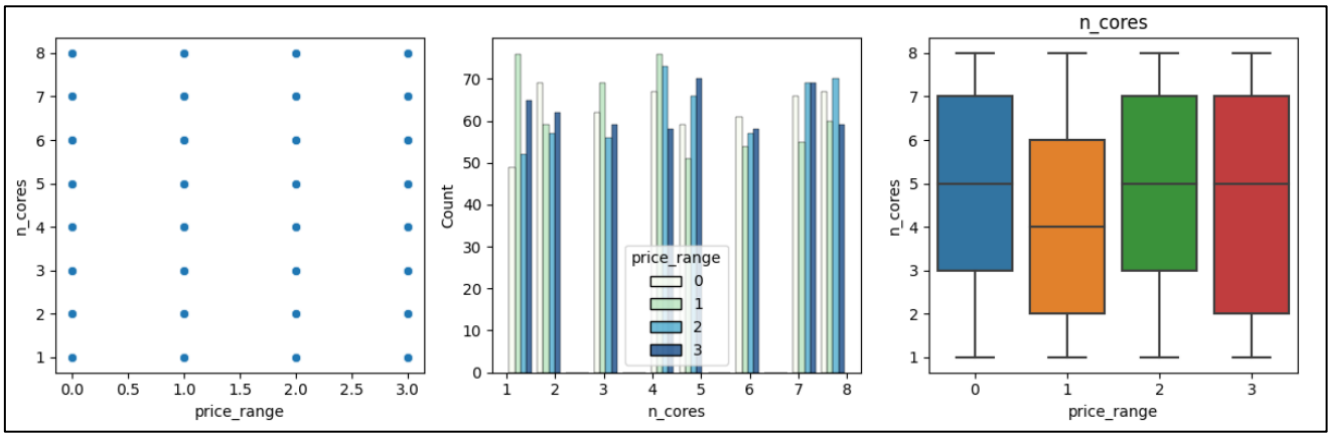


Рисунок 2.19 – Співвідношення кількості ядер процесора та цінової групи

Представлені графіки демонструють, що смартфони з більшою кількістю ядер процесора зазвичай коштують дорожче. Це пов’язано з тим, що такі моделі забезпечують вищу продуктивність та кращі можливості для багатозадачності, що підвищує їхню привабливість для користувачів.

Співвідношення кількості мегапікселів основної камери та цінової групи показано на рисунку 2.20.

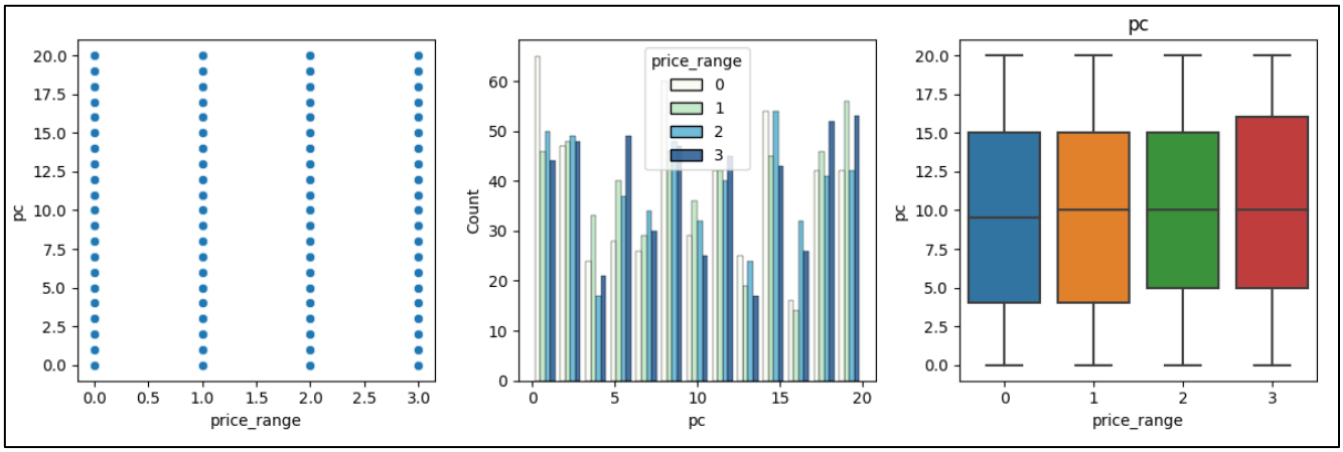


Рисунок 2.20 – Співвідношення кількості мегапікселів основної камери та цінової групи

Наведені діаграми ілюструють тенденцію, що мобільні телефони з вищою кількістю мегапікселів основної камери, як правило, мають більшу

вартість. Це пояснюється тим, що такі моделі забезпечують кращу якість фотографій і відео, що робить їх більш привабливими для користувачів.

Співвідношення роздільної здатності екрану (висота та ширина) та цінової групи наведено на рисунках 2.21-2.22.

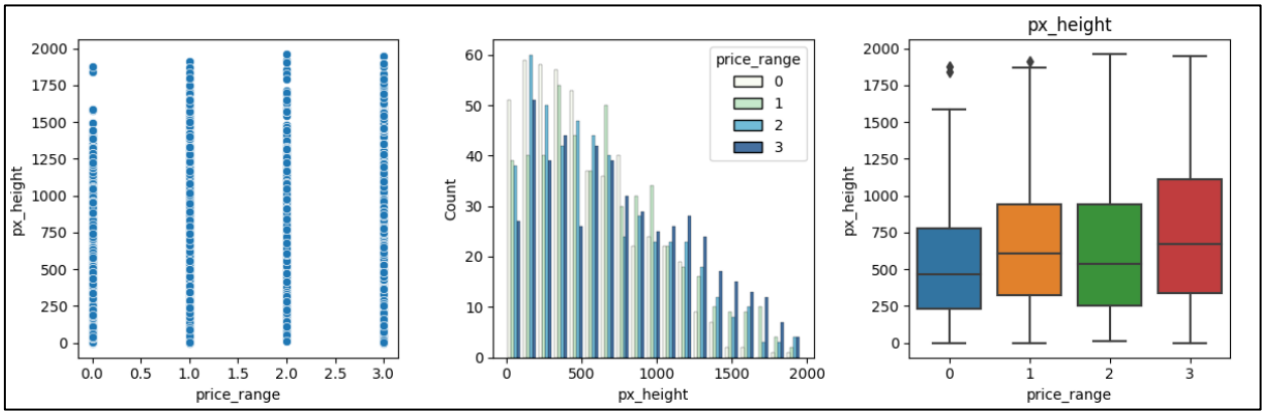


Рисунок 2.21 – Співвідношення висоти роздільної здатності та цінової групи

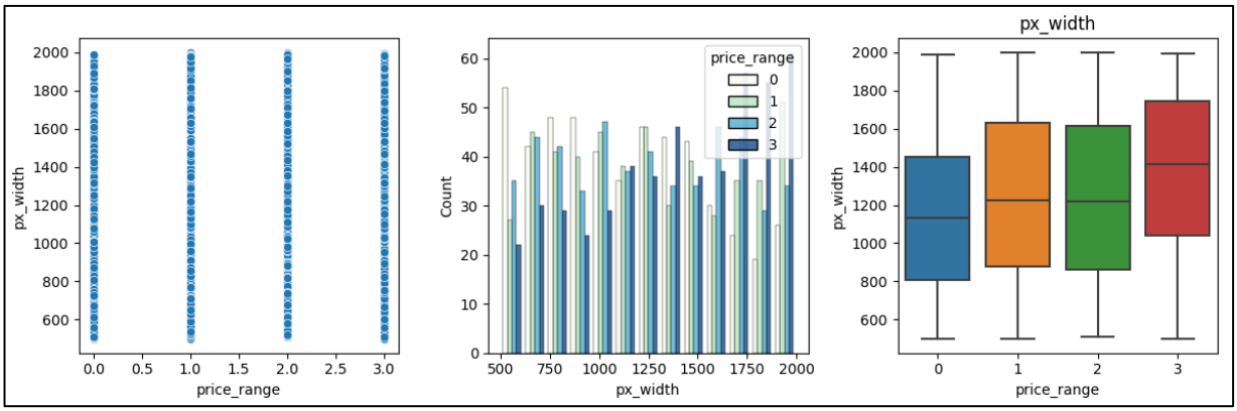


Рисунок 2.22 – Співвідношення ширини роздільної здатності та цінової групи

Аналіз графіків на рисунках 2.21 та 2.22 дозволяє зробити висновок, що вони демонструють залежність ширини екрану та роздільної здатності смартфонів від їхнього цінового діапазону. Можна помітити, що більш дорогі моделі, як правило, оснащені більшими екранами та мають вищу роздільну здатність, що може бути важливим фактором для споживачів при виборі смартфона.

Співвідношення оперативної пам’яті та цінової групи зображено на рисунку 2.23.

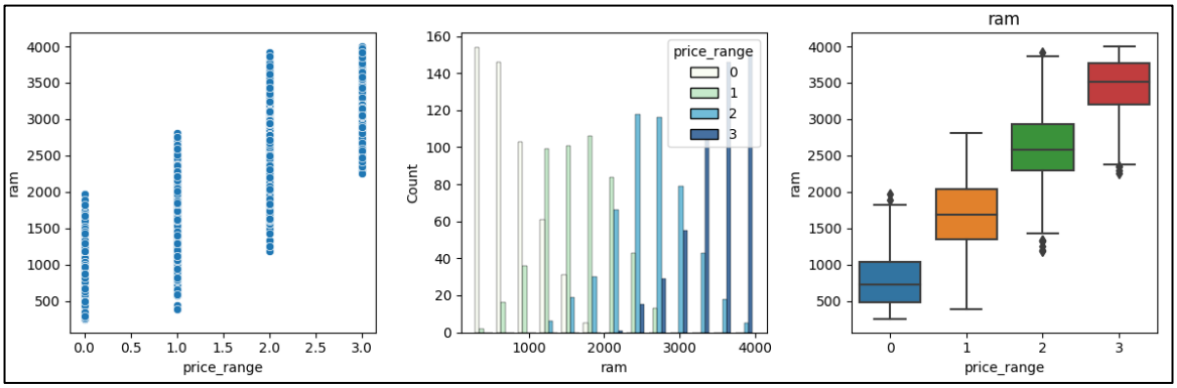


Рисунок 2.23 – Співвідношення оперативної пам’яті та цінової групи

З наведених графіків видно, що зі збільшенням ціни смартфонів спостерігається тенденція до зростання обсягу оперативної пам’яті (RAM). Це може пояснюватися тим, що в дорожчих моделях встановлюються більш продуктивні компоненти.

Співвідношення висоти сенсорного екрану та цінової групи рисунок 2.24.

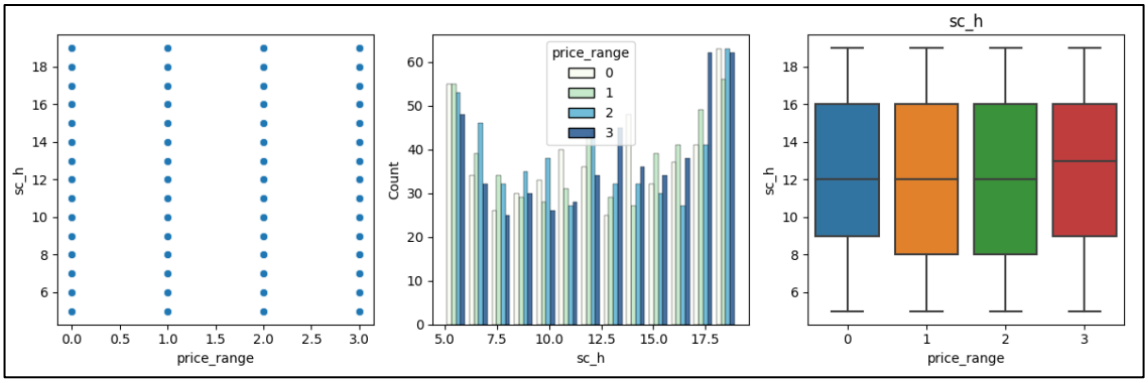


Рисунок 2.24 – Співвідношення висоти сенсорного екрану та цінової групи

Аналіз наведених графіків, що відображають розподіл висоти сенсорного екрану (Sc_h) у різних цінових категоріях мобільних пристроїв

(Price_range), показує, що цей параметр не є суттєвим фактором для прогнозування ціни. Усі категорії демонструють схожий розподіл висоти екрана, із медіаною приблизно 12–14 одиниць та міжквартильним розмахом від 10 до 16 одиниць. Це свідчить про те, що висота сенсорного екрану має мінімальний вплив на визначення цінової категорії смартфона.

Співвідношення ширини сенсорного екрану та цінової групи показано на рисунку 2.25.

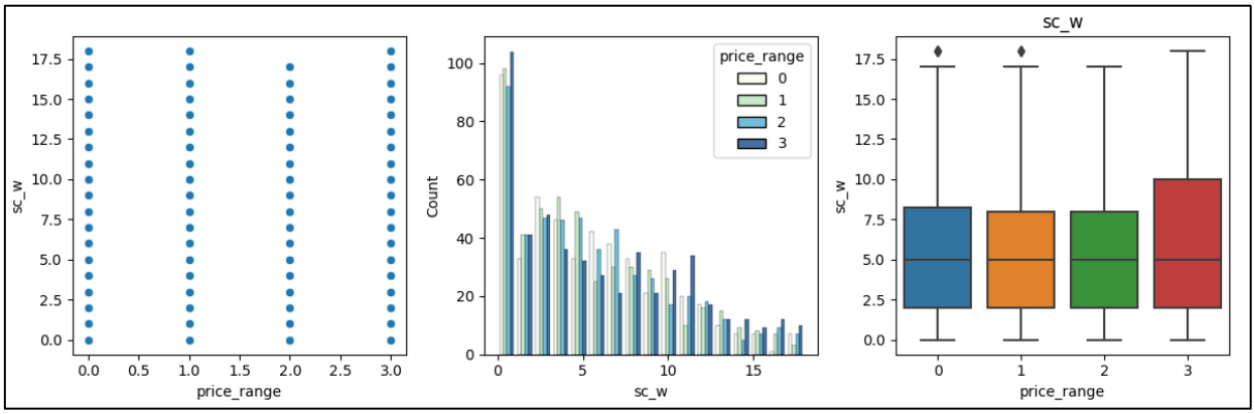


Рисунок 2.25 – Співвідношення ширини сенсорного екрану та цінової групи

Аналіз графіків, що відображають розподіл ширини сенсорного екрану (Sc_w) у різних цінових категоріях мобільних пристроїв (Price_range), дозволяє зробити висновок, що цей параметр не є визначальним для прогнозування ціни. Незважаючи на те, що у найвищій ціновій категорії (3) спостерігається дещо більша варіативність та вища медіана, загальний розподіл ширини екрана у всіх категоріях залишається схожим. Це свідчить про обмежений вплив даної характеристики на формування цінової категорії смартфона.

Співвідношення розподілу часу розмови та цінової групи наведено на рисунку 2.26.

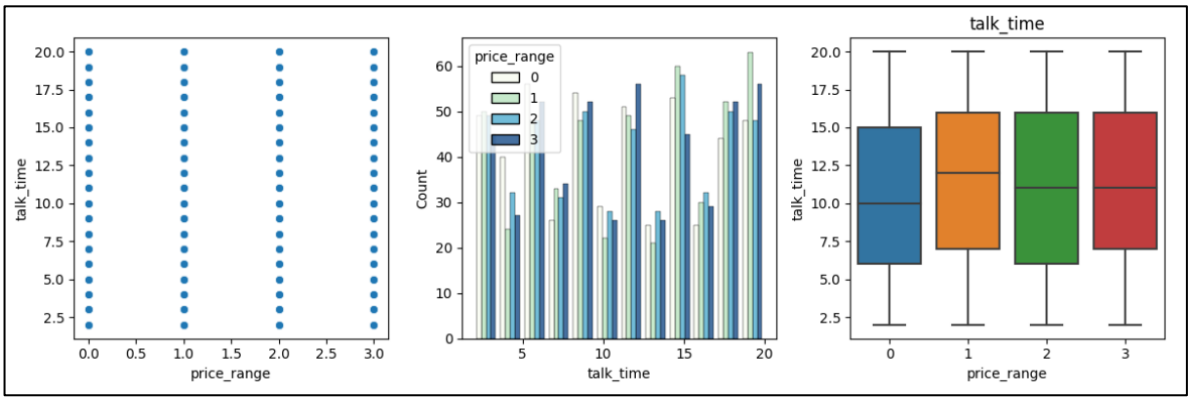


Рисунок 2.26 – Співвідношення розподілу часу розмови та цінової групи

Дослідження графіків, що ілюструють розподіл тривалості розмови (Talk_time) у різних цінових категоріях мобільних пристроїв (Price_range), показує, що цей показник не має суттєвого впливу на прогнозування цінової категорії смартфона.

Співвідношення підтримки 4 і 3G технологій та ціновою групою показано на рисунках 2.27-2.28.

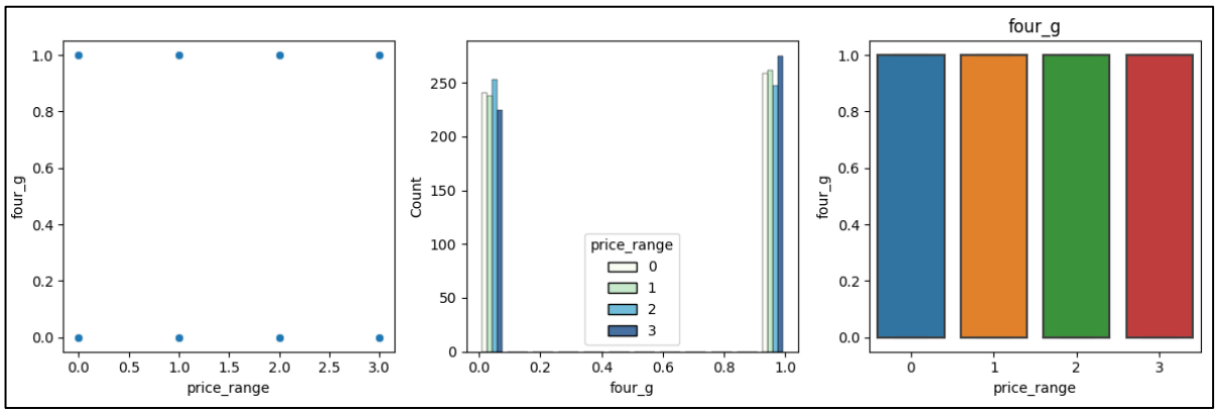


Рисунок 2.27 – Співвідношення підтримки 4G технології та ціновою групою

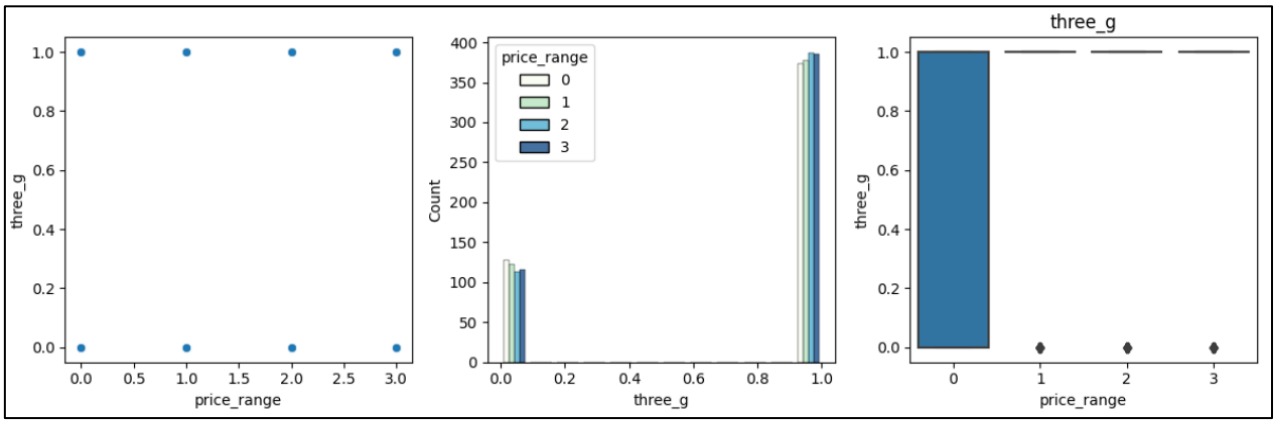


Рисунок 2.28 – Співвідношення підтримки 3G технології та ціновою групою

Наведені графіки вище (рис. 2.27-2.28) демонструють, що мобільні пристрої з підтримкою 3G та 4G зазвичай мають дещо вищу ціну порівняно з моделями без 3G. Це можна пояснити тим, що 3G і 4G є більш сучасними технологіями, які забезпечують швидший та більш стабільний мобільний інтернет.

Співвідношення наявності сенсорного екрану та цінової групи наведено на рисунку 2.29.

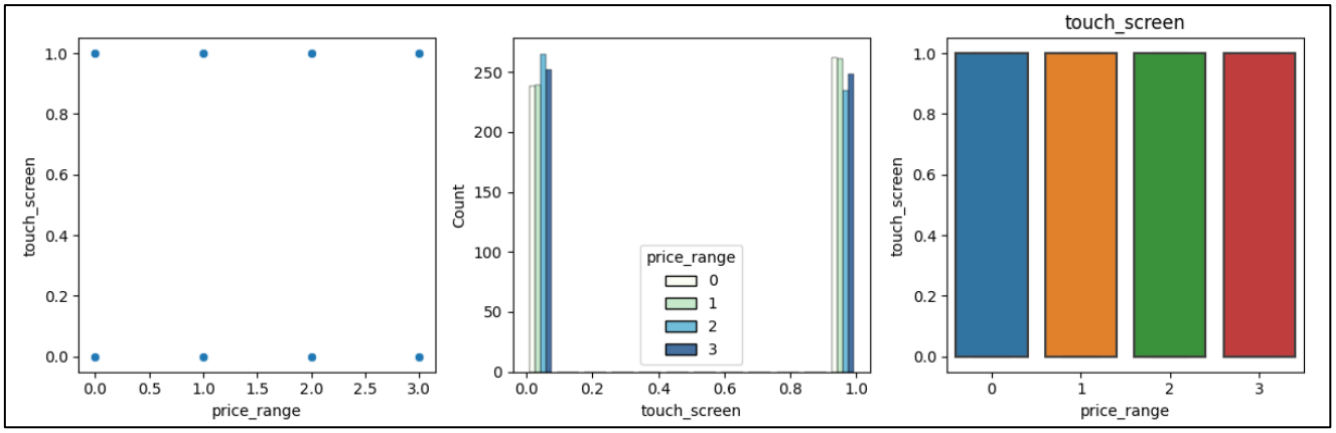


Рисунок 2.29 – Співвідношення наявності сенсорного екрану та цінової групи

З аналізу графіків видно, що смартфони з сенсорним екраном зазвичай коштують значно дорожче, ніж моделі без нього. Це пояснюється тим, що сенсорні екрани є технологічно складнішими та дорожчими у виробництві порівняно з традиційними клавіатурами.

Співвідношення наявності функції Wifi та ціновою групою продемонстровано на рисунку 2.30.

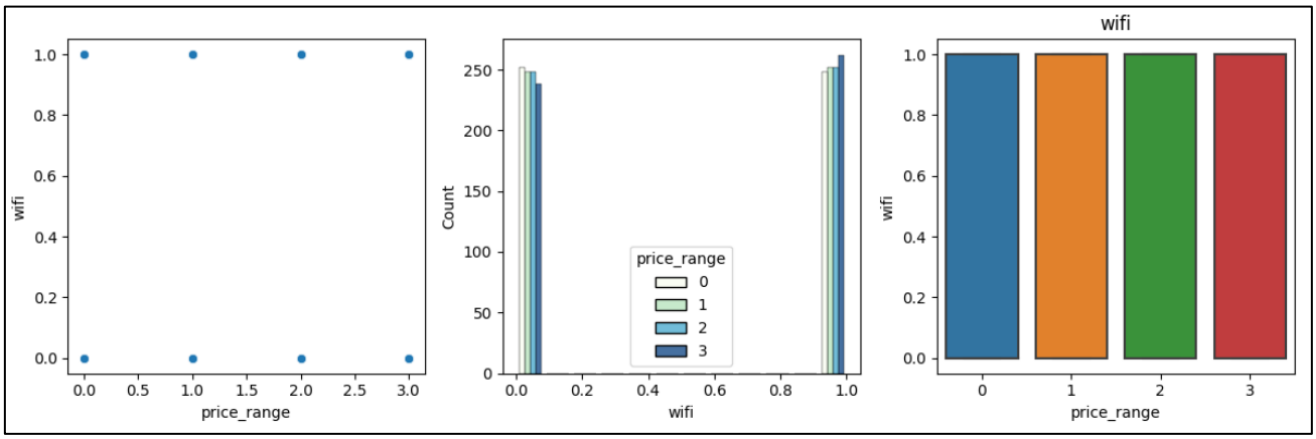


Рисунок 2.30 – Співвідношення наявності функції Wifi та ціновою групою

Показані графіки демонструють те, що мобільні телефони з підтримкою Wi-Fi зазвичай мають трохи вищу ціну порівняно з моделями без цієї функції. Це можна пояснити тим, що наявність Wi-Fi підвищує функціональність пристрою та робить його більш зручним для користувачів.

У межах розвідувального аналізу також було створено коробкові діаграми (box-plot) для ключових характеристик, що істотно впливають на розподіл мобільних телефонів за різними ціновими сегментами.

На рисунку 2.31 показано коробкову діаграму (box-plot), що ілюструє розподіл ємності акумулятора серед різних цінових сегментів.

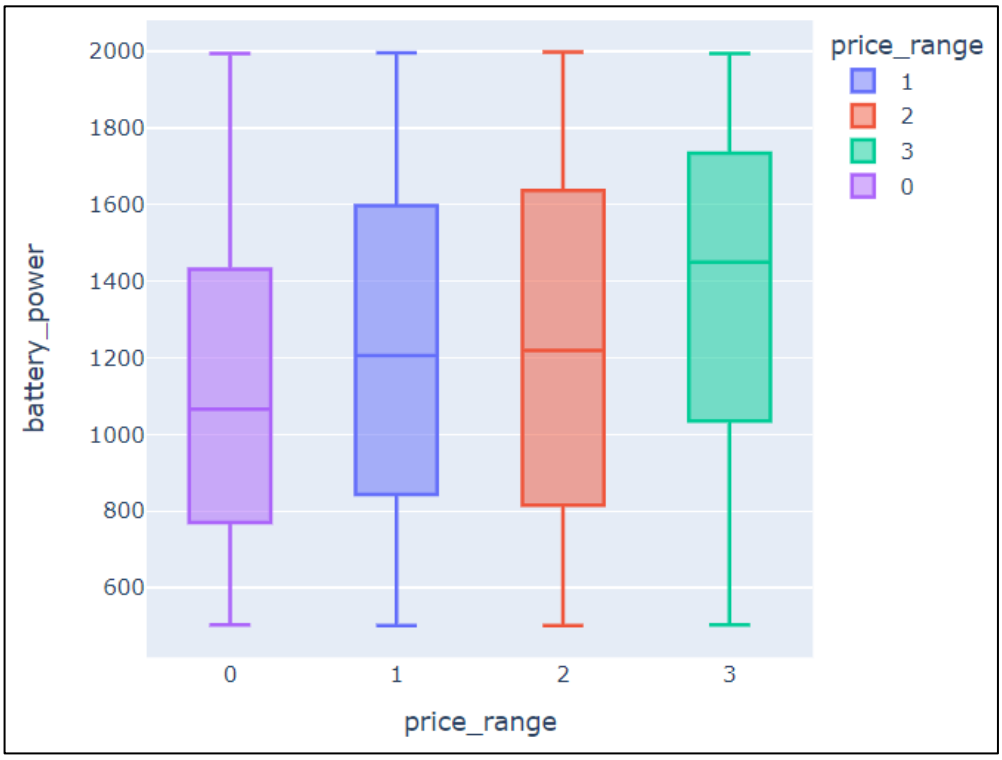


Рисунок 2.31 – Коробкова діаграма розподілу ємності акумулятора серед різних цінових сегментів

Проводячи аналіз представленого box-plot графіка розподілу ємності акумулятора (battery_power) у різних цінових сегментах (price_range) мобільних пристроїв, можна помітити, що зі зростанням вартості смартфонів підвищується й медіанний показник ємності батареї. Це означає, що дорожчі моделі переважно оснащені потужнішими акумуляторами. Водночас для всіх цінових груп характерна суттєва мінливість значень потужності батареї, а максимальні показники також збільшуються разом із ціною. Отже, ємність акумулятора виступає одним із ключових чинників, що впливають на формування вартості смартфонів.

Коробкова діаграма (box-plot) розподілу оперативної пам'яті серед різних цінових сегментів зображена на рисунку 2.32.

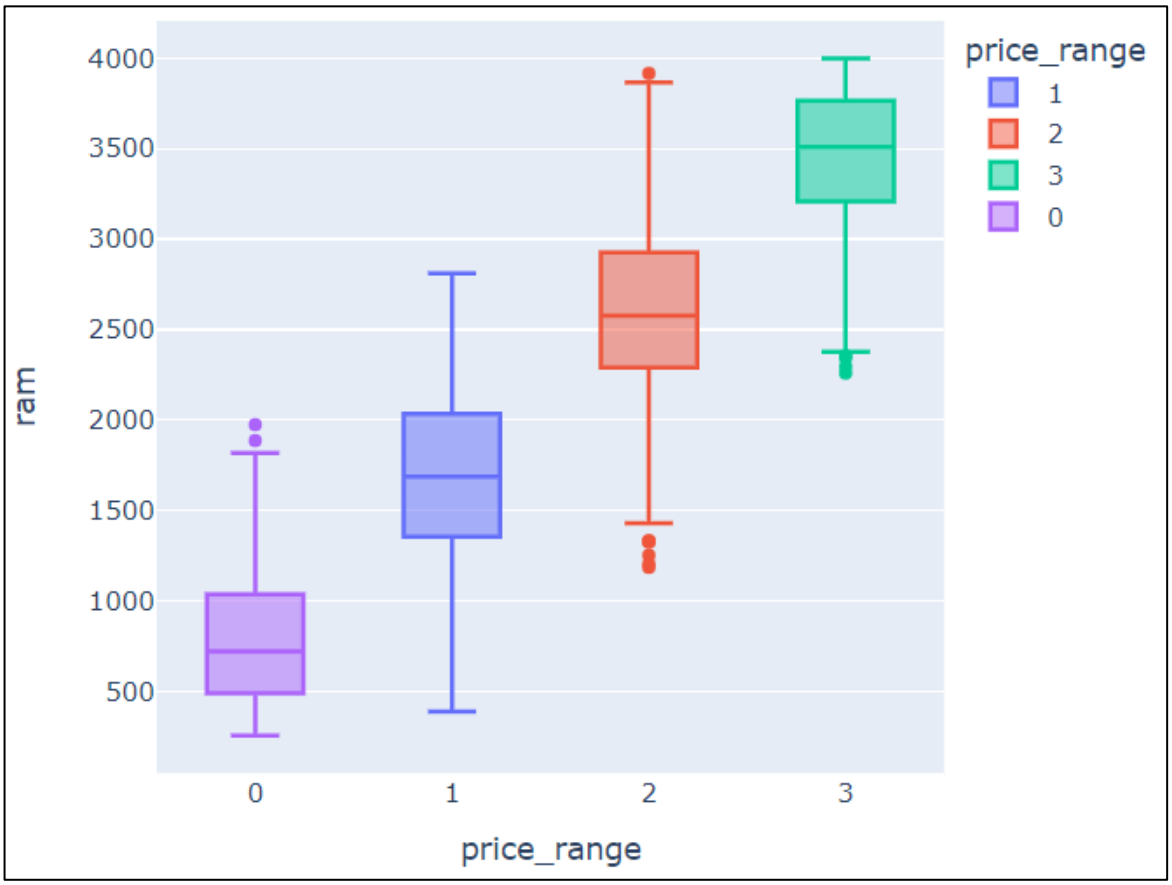


Рисунок 2.32 – Коробкова діаграма розподілу оперативної пам'яті серед різних цінових сегментів

Наведена діаграма демонструє вплив обсягу оперативної пам’яті (RAM) на ціни мобільних пристроїв у різних цінових сегментах. З аналізу видно, що зі збільшенням обсягу RAM зростає середня вартість пристрою, що свідчить про позитивний вплив цього параметра на ціну смартфона. Крім того, спостерігається помітна варіативність цін у межах різних категорій, а наявність викидів може вказувати на присутність особливо дорогих або дешевих моделей з певним обсягом оперативної пам’яті. Отже, можна зробити висновок, що RAM є одним із ключових факторів, що визначають цінність мобільного пристрою.

Рисунок 2.33 ілюструє коробкову діаграму (box-plot) розподілу внутрішньої пам'яті серед різних цінових сегментів.

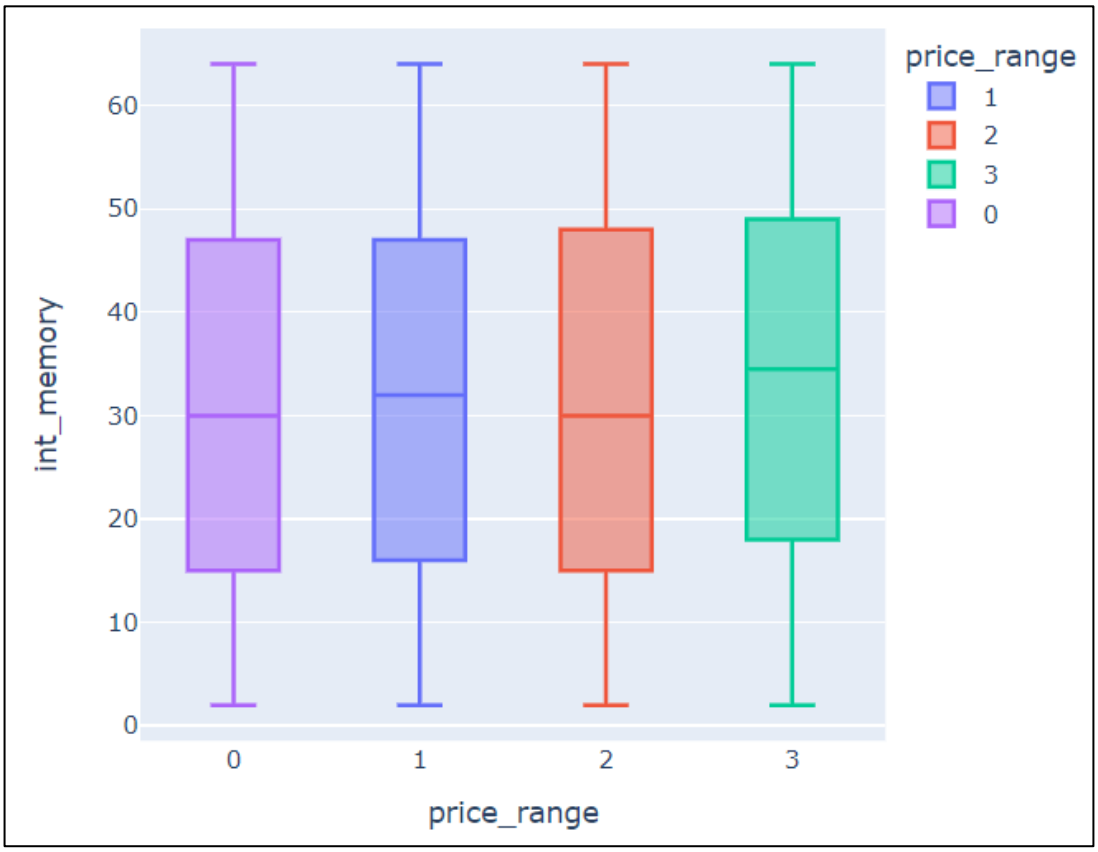


Рисунок 2.33 – Коробкова діаграма розподілу внутрішньої пам'яті серед різних цінових сегментів.

Аналіз наведеного графіка свідчить, що обсяг внутрішньої пам’яті має значний вплив на вартість мобільних телефонів. Це може пояснюватися як особливостями конкретних моделей, так і різними підходами виробників щодо комплектації пристроїв певними характеристиками.

Коробкова діаграма (box-plot) розподілу швидкості мікропроцесора серед різних цінових сегментів зображено на рисунку 2.34.

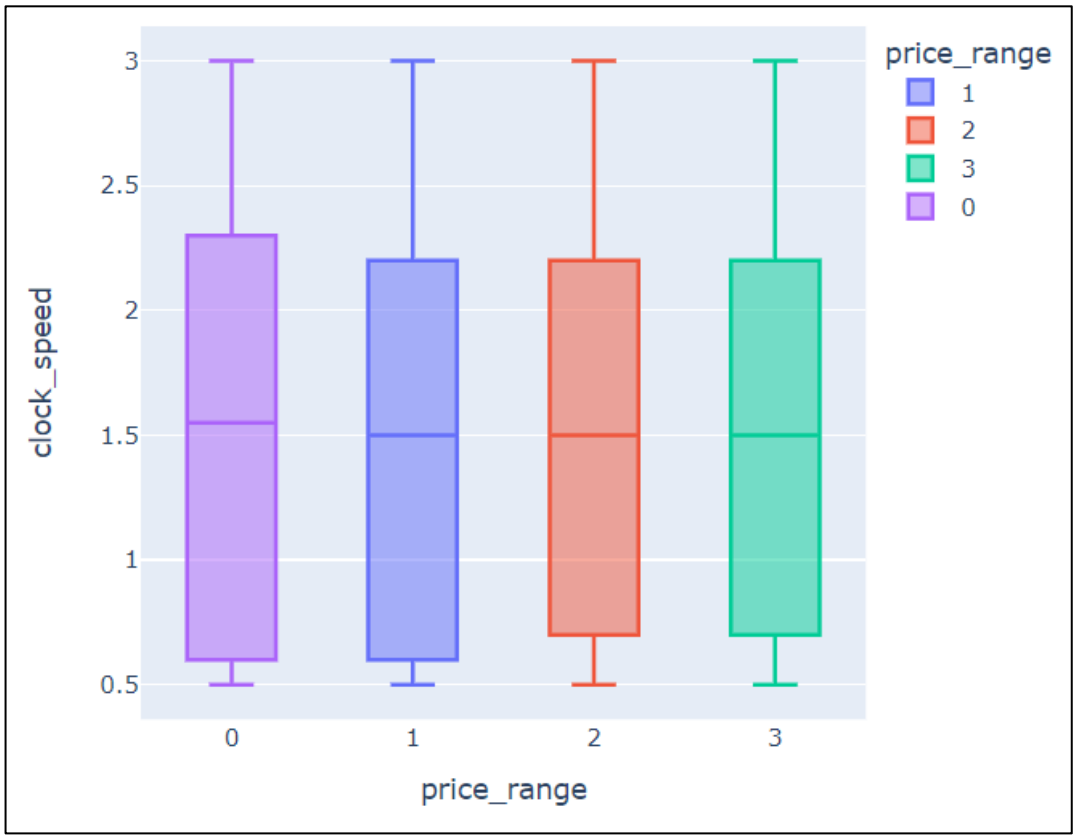


Рисунок 2.34 – Коробкова діаграма розподілу швидкості мікропроцесора серед різних цінових сегментів

Наведений графік ілюструє залежність тактової частоти процесора від цінового діапазону смартфонів. Зазвичай дорожчі моделі оснащені більш потужними процесорами з вищою тактовою частотою, що забезпечує підвищену продуктивність і швидкість роботи пристрою.

Коробкова діаграма (box-plot) розподілу висоти роздільної здатності екрана серед різних цінових сегментів продемонстрована на рисунку 2.35.

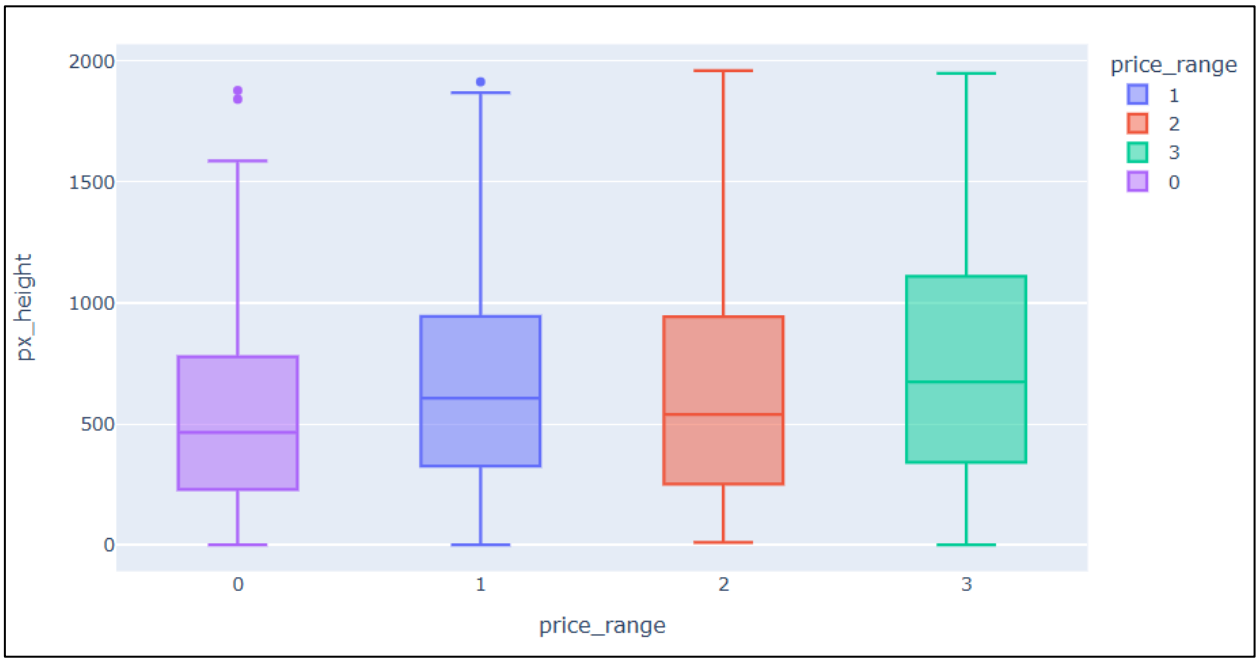


Рисунок 2.35 – Коробкова діаграма розподілу висоти роздільної здатності екрана серед різних цінових сегментів

Наведений графік добре відображає залежність висоти екрана смартфона від його цінової категорії. Із графіка видно, що зі збільшенням цінового діапазону зростає і медіанна висота екрана, що свідчить про позитивний вплив параметра px_height на формування ціни телефону. У дорожчих цінових сегментах спостерігається ширший діапазон значень та більша варіативність висоти екрана, що вказує на різноманіття моделей і наявність як стандартних, так і значно більших дисплеїв. Наявні викиди у всіх категоріях демонструють існування окремих моделей з нетипово високими або низькими значеннями px_height. Отже, можна зробити висновок, що висота екрана є важливим технічним параметром, який істотно впливає на кінцеву ціну смартфона та відрізняє моделі різних цінових діапазонів.

Коробкову діаграму (box-plot) розподілу ширини роздільної здатності екрана серед різних цінових сегментів показано на рисунку 2.36.

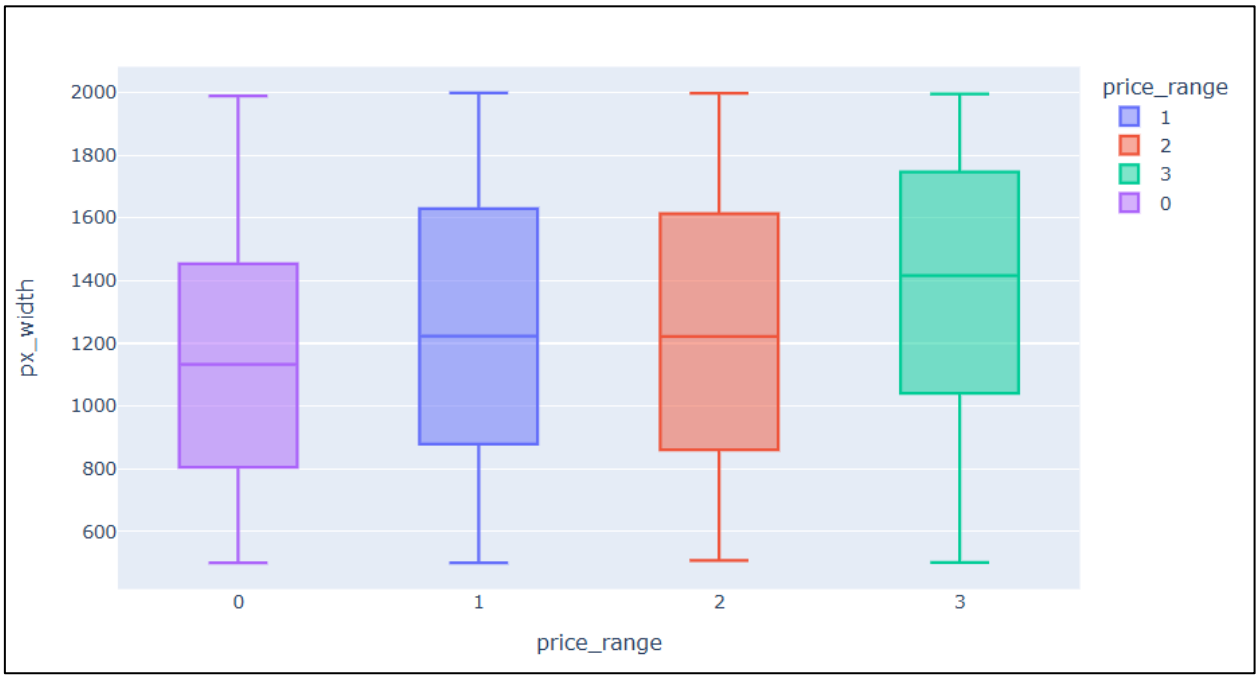


Рисунок 2.36 – Коробкова діаграма (box-plot) розподілу ширини роздільної здатності екрана серед різних цінових сегментів

Наведений графік чітко демонструє залежність ширини екрана смартфона від його цінового діапазону. З аналізу видно, що зі збільшенням цінової категорії поступово зростає середнє значення px_width, що вказує на позитивний вплив ширини дисплея на формування ціни телефону. У дорожчих сегментах спостерігається ширший діапазон значень та більша варіативність ширини екрана, що свідчить про різноманітність моделей – від стандартних до значно ширших дисплеїв. Наявні викиди демонструють існування окремих моделей з нетипово великими або малими значеннями px_width у кожному ціновому діапазоні. Отже, можна зробити висновок, що ширина екрана є важливим технічним параметром, який впливає на кінцеву ціну смартфона та допомагає відрізняти моделі різних цінових сегментів.

Коробкова діаграма (box-plot) розподілу ваги мобільних телефонів серед різних цінових сегментів представлена на рисунку 2.37.

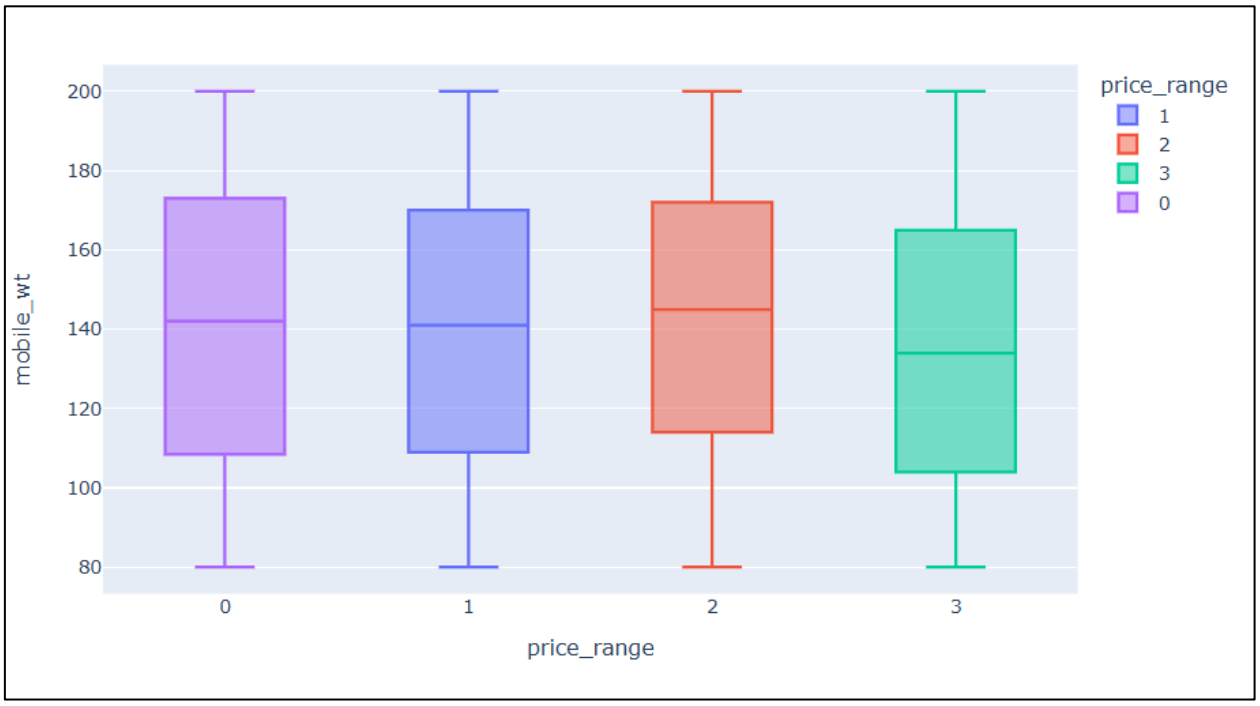


Рисунок 2.37 – Коробкова діаграма розподілу ваги мобільних телефонів серед різних цінових сегментів

Наведений графік відображає залежність ваги смартфона від його цінового діапазону. З аналізу видно, що середні значення mobile_wt у різних цінових категоріях залишаються близькими між собою, що свідчить про відсутність чіткої тенденції зростання або зменшення ваги смартфонів із підвищенням price_range. У всіх сегментах спостерігається подібний діапазон значень, а інтерквартильні розмахи майже збігаються, що вказує на однакову варіативність ваги незалежно від категорії.

Хоча окремі групи демонструють незначні відмінності у медіанних значеннях, ці коливання не є суттєвими та не формують стійкої залежності між вагою та ціновим сегментом. Наявні викиди у даних відображають існування окремих моделей із нетипово малою або великою вагою, але їх кількість незначна та не впливає на загальну картину.

Отже, можна зробити висновок, що вага смартфона не є визначальним технічним параметром у формуванні його цінової категорії. Показник

mobile_wt не демонструє сильної взаємозалежності з price_range та не дозволяє виразно відрізнити моделі різних цінових сегментів.

У процесі розвідувального аналізу була також побудована теплова карта, яка відображає кореляційну матрицю, де кожен елемент демонструє величину кореляційного зв'язку між відповідними змінними (рис. 2.38).

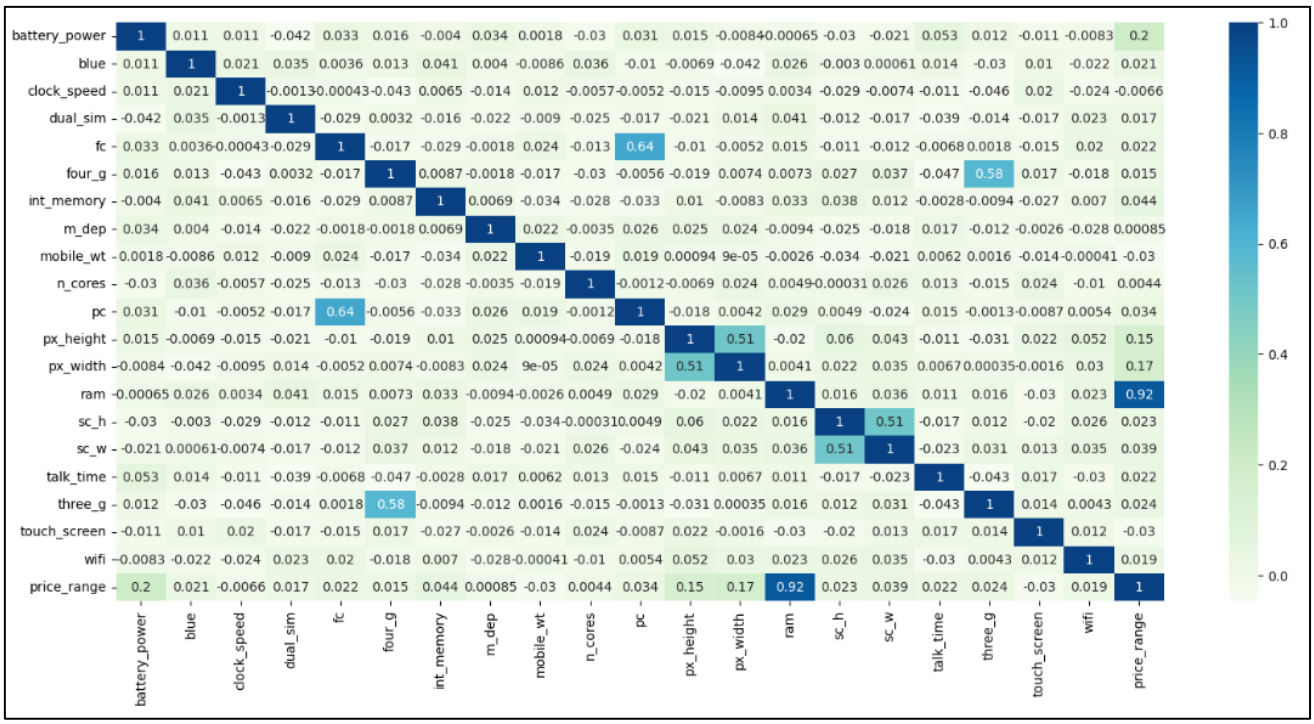


Рисунок 2.38 – Кореляційна теплова карта

Аналіз наведеної теплової карти дозволяє зробити висновок, що між багатьма змінними спостерігаються значні кореляційні зв'язки. Найсильніша взаємозалежність простежується між обсягом оперативної пам'яті (RAM) та ціновою категорією пристрою (price_range), що свідчить про те, що смартфони з більшою оперативною пам'яттю зазвичай відносяться до вищих цінових сегментів. Такий зв'язок обумовлений тим, що RAM є критично важливим для продуктивності: більший обсяг пам'яті дозволяє одночасно запускати більше додатків, обробляти великі обсяги даних та забезпечує більш швидку роботу пристрою. Тому виробники часто оснащують преміальні моделі більшою оперативною пам'яттю, що безпосередньо впливає на їхню вартість.

Крім того, помітні значущі кореляції спостерігаються між іншими парами характеристик, зокрема Pc та Fc, Px_height та Px_width, Sc_w та Sc_h, а також Three_g та Four_g, що вказує на узгоджене зростання або зменшення цих параметрів. Інші змінні демонструють слабкі зв’язки, свідчачи про їхню відносну незалежність та окремий внесок у формування досліджуваних показників.

2.3 Висновки

У цьому розділі виконано обробку даних із розвідувальним аналізом, зокрема створено теплову карту кореляцій, що дозволила визначити ключові характеристики, які найбільш суттєво впливають на зростання цін мобільних телефонів. До них відносяться: обсяг оперативної пам’яті (RAM), параметри основної та фронтальної камер (Pc та Fc), висота та ширина роздільної здатності дисплея (Px_height та Px_width), розміри екрану (Sc_w та Sc_h), а також підтримка мереж третього та четвертого покоління (Three_g та Four_g).

3 РОЗРОБЛЕННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ПЕРЕДБАЧЕННЯ ЦІНИ ТЕЛНФОНІВ

3.1 Розроблення математичних моделей

Розглянемо моделі класифікації, які застосовувалися під час виконання магістерської кваліфікаційної роботи: Random Forest Classifier, XGBoost Classifier, AdaBoost Classifier, ExtraTree Classifier, Logistic Regression, SVM Classifier, MLP Classifier. Спочатку було подано опис принципу роботи кожної з цих моделей.

Random Forest – це ансамблевий метод машинного навчання, який об’єднує значну кількість дерев рішень з метою покращення точності прогнозів і зменшення ймовірності перенавчання. Розглянемо основні кроки алгоритму з математичної точки зору [16]:

1. Бутстрепінг (Bootstrap Sampling):

– Нехай D – початковий набір даних, що містить n спостережень. Random Forest формує B бутстреп-відібраних піднаборів $D_1, D_2, \ldots, D_B$, кожен розміром n, шляхом випадкового вибору зразків з поверненням. Це означає, що деякі спостереження можуть повторюватися в наборі, а деякі – взагалі не потрапляти до нього.

2. Побудова дерев рішень:

– Для кожного піднабору D_i створюється дерево T_i. На кожному вузлі дерева замість перевірки всіх ознак випадковим чином обирається підмножина ознак $M \subset X$, де $|M| = m \leq p$. Найоптимальніша ознака для розділення вузла обирається з цієї підмножини.

3. Ріст дерев:

– Кожне дерево росте без обрізання до тих пір, поки всі зразки в листі належать до одного класу або не досягнуто мінімального розміру вузла.

4. Агрегація результатів:

– Для рішення задач класифікації: кожне дерево T_i видає свій прогноз класу для зразка x. Остаточне передбачення визначається більшістю голосів:

$$y = \text{mode}\{ T_1(x), T_2(x),..., T_B(x)\}$$

– Для рішення задач регресії: кожне дерево видає передбачене значення $T_i(x)$. Остаточне прогнозне значення обсилюється як середнє всіх передбачень:

$$y = \frac{1}{B}\sum_{i=1}^{B} T_i(x)$$

Таким чином, Random Forest поєднує результати багатьох незалежних дерев, що забезпечує більш стійкі та точні прогнози порівняно з одним деревом рішень.

Алгоритм методу Random Forest зображено на рисунку 3.1.

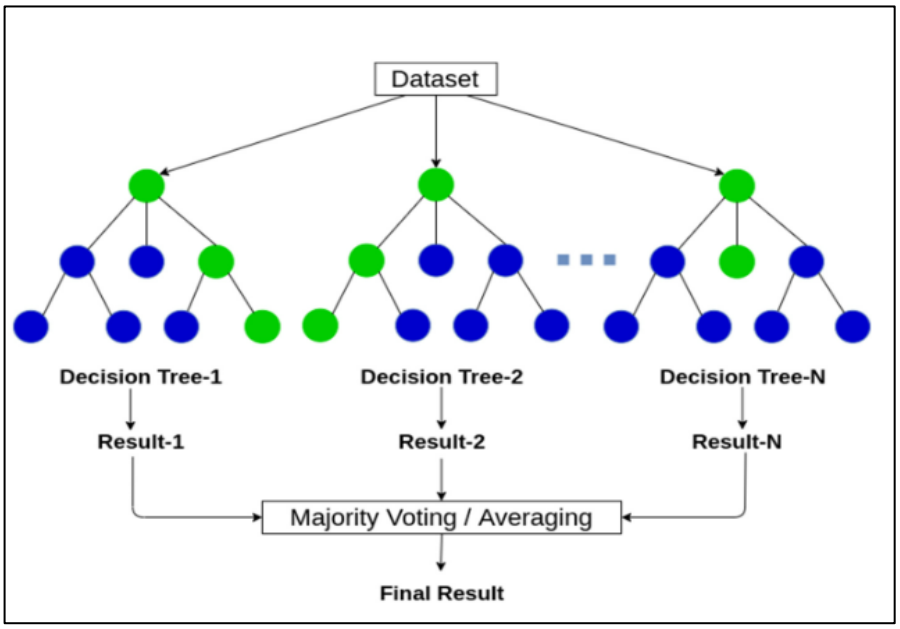


Рисунок 3.1 – Блок-схема роботи алгоритму Random Forest

XGBoost (Extreme Gradient Boosting) – це вдосконалений алгоритм градієнтного бустингу, який ефективно застосовується для задач класифікації та регресії. Основні кроки роботи алгоритму можна описати наступним чином [16]:

1. Ініціалізація моделі:

– Спершу будується початкова модель $F_0(x)$, яка зазвичай приймає значення константи. Для задач регресії це середнє значення цільової змінної, а для класифікації – логарифмічні шанси ймовірності належності до класу.

2. Послідовне додавання дерев:

– Алгоритм поступово додає нові дерева $h_t(x)$ на кожній ітерації t для покращення точності моделі.

3. Обчислення залишків:

– На кожній ітерації обчислюються залишки (residuals) або псевдо-резідуали для поточної моделі $F_{t-1}(x)$. Вони визначаються як негативний градієнт функції втрат L щодо передбачень:

$$r_i^{(t)} = -\left[\frac{\partial L(y_i, F_{t-1}(x_i))}{\partial F_{t-1}(x_i)}\right]$$

4. Побудова дерева:

– Нове дерево $h_t(x)$ будується для апроксимації залишків $r_i^{(t)}$, таким чином мінімізуючи функцію втрат для поточного набору даних:

$$h_t = argmin \sum_{i=1}^{n} L\left(y_i, F_{t-1}(x_i) + h(x_i)\right)$$

5. Оновлення моделі:

– Модель оновлюється шляхом додавання передбачень нового дерева з коефіцієнтом навчання η (learning rate):

$$F_t(x) = F_{t-1}(x) + \eta h_t(x)$$

6. Регуляризація:

– Для запобігання перенавчанню, XgBoost використовує регуляризаційний термін, який включає L1 та L2 регуляризацію ваг листів дерев. Функція втрат із регуляризацією має вигляд:

$$L(\theta) = \sum_{i=1}^{n} L(y_i, F_{t-1}(x_i) + h(x_i) + \Omega(h_t)$$

Де $\Omega(h_t)$ – регуляризаційний термін.

Таким чином XGBoost комбінує послідовне навчання дерев, градієнтне наближення функції втрат та регуляризацію для досягнення високої точності прогнозування та стійкості до перенавчання.

Алгоритм методу роботи моделі XGBoost показано на рисунку 3.2

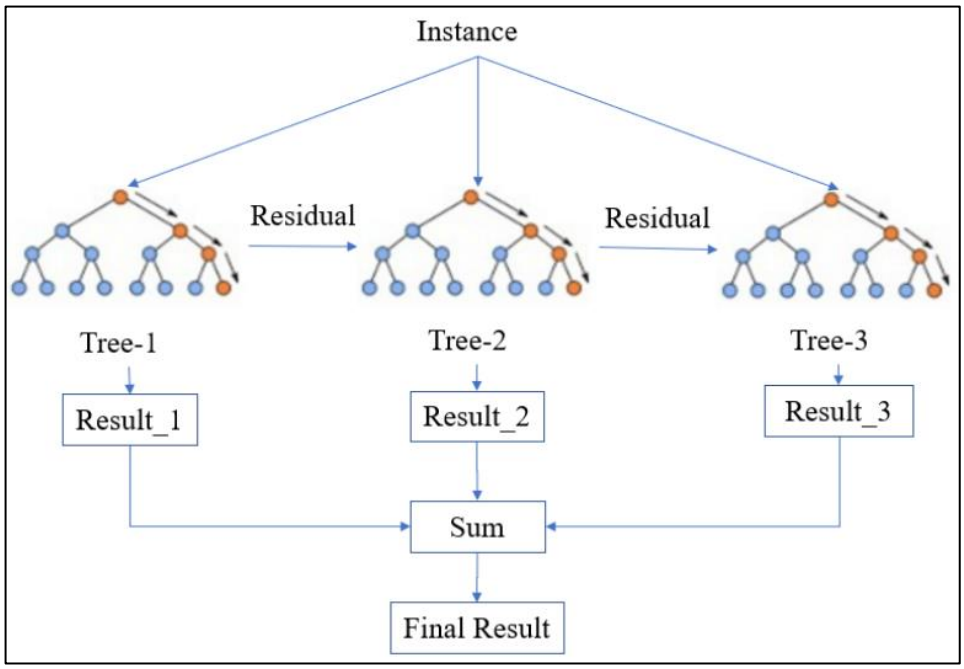


Рисунок 3.2 – Блок-схема роботи алгоритму XGBoost

AdaBoost (Adaptive Boosting) – це популярний метод ансамблевого навчання, який об’єднує декілька простих моделей (базових класифікаторів) для створення потужного прогнозуючого алгоритму. Головна задача AdaBoost полягає в ітеративному навчанні базових моделей із посиленою увагою до тих спостережень, які складніше правильно класифікувати.

Математично алгоритм працює наступним чином [16]:

1. Ініціалізація ваг:

Кожному з *N* навчальних прикладів призначається вага $\omega i = \frac{1}{N}$, тобто всі спостереження рівнозначні на початку навчання.

2. Послідовне навчання базових моделей:

Для кожної інерації *t = 1, 2,..., T* (де *T* – загальна кількість базових класифікаторів) виконуються такі кроки:

– Навчання базового класифікатора:

Створюємо базовий класифікатор h_t на поточному навчальному наборі із заданими вагами ω_i.

– Обчислення помилки клачифікатора:

Похибка ϵ_t визначається як зважена сума неправильних передбачень:

$$\epsilon_t = \sum_{i=1}^{N} \omega_i I\big(y_i \neq h_t(x_i)\big)$$

Де $I\big(y_i \neq h_t(x_i)\big)$ – індикаторна функція, яка приймає значення 1 у випадку помилки класифікатор та 0, якщо класифікатор визначив правильний клас.

– Розрахунок ваги класифікатора:

Значення ваги базового класифікатора α_t обчислюється за формулою:

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

– Коригування ваг тренувальних зразків:

Ваги прикладів змінюються так, щоб підкреслити важкі для уласифікації спостереження:

$$\omega_i \leftarrow \omega_i \exp\left(-\alpha_t y_i h_i(x_i)\right)$$

Далі ваги приводяться до нормалізованого вигляду, забезпечуючи суму рівну 1:

$$\omega_i \leftarrow \frac{\omega_i}{\sum_{j=1}^{N} \omega_i}$$

3. Формування кінцевої моделі:

Остаточний прогноз здійснюється за допомогою зваженої комбінації всіх базових моделей:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t\,(x)\right)$$

За рахунок вагових коефіцієнтів α_t AdaBoost посилює внесок тих класифікаторів, які добре працюють на складних прикладах, і зменшує вплив тих, що часто помиляються. Це дозволяє отримати високоточний ансамблевий класифікатор, що перевершує ефективність окремих базових моделей.

Алгоритм методу продемонстровано на рисунку 3.3

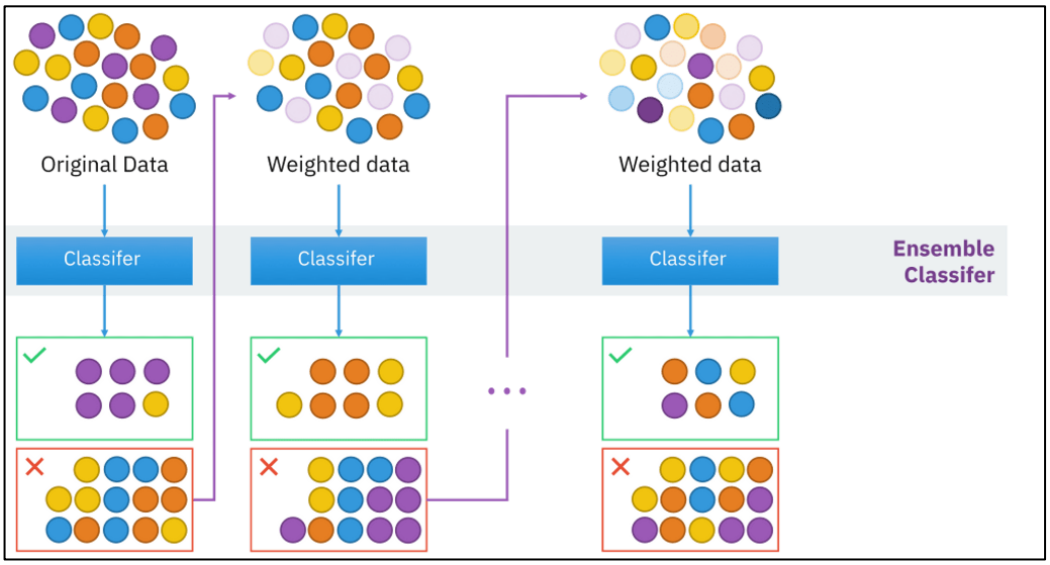


Рисунок 3.3 – Блок-схема роботи алгоритму AdaBoost

Метод ExtraTrees (Extremely Randomized Trees) є ансамблевим підходом для задач класифікації та регресії. Він є модифікацією Random Forest, проте відрізняється більшою випадковістю при побудові дерев, що допомагає знизити дисперсію моделі.

Математичну модель даного методу можна описати наступним чином [16]:

1. Створення ансамблю дерев:

Ансамбль включає M дерев рішень, де M – задана кількість дерев.

2. Випадковий відбір ознак:

На кожному вузлі дерева формується випадкова підмножина допустимих ознак. Кількість обраних ознак зазвичай визначається як $\sqrt{d}$, де d – загальна кількість ознак.

3. Випадкове визначення порогів:

Для кожної обраної ознаки створюється набір випадкових порогів розділення з можливих значень цієї ознаки.

4. Розділ вузла:

Серед усіх випадково обраних порогів вибирається той, який забезпечує максимальний приріст інформації або найкраще значення індексу Джині. На його основі вузол розбивається на два підвузли.

5. Побудова дерева:

Кожен вузол обробляється до тих пір, поки не досягнуто одного з критеріїв:

– Максимальна глибина дерева досягнута;

– Кількість об’єктів у вузлі менша за встановлений мінімальний поріг;

– Всі спостереження в вузлі відносяться до одного класу.

6. Агрегація прогнозів ансамблю:

Для класифікації нового зразка *x*:

– Кожне дерево $h_m(x)$ робить власний прогноз, де $m = 1,\ldots,M$;

– Остаточний результат визначається більшістю голосів:

$$H(x) = argmax \sum_{m=1}^{M} I(h_m(x) = y)$$

Де *I* – позначає індикаторну функцію, яка дорівнює 1, коли *y* відповідає певному класу, і 0 у всіх інших випадках.

Основні особливості ExtraTrees у порівнянні з Random Forest двома:

– Вузли розділяються за допомогою випадкових порогів, а не оптимального розділу;

– Для навчання кожного дерева використовується весь набір даних, а не бутстрепінг (випадкова вибірка з поверненням).

Завдяки цим особливостям ExtraTrees демонструє більшу випадковість, що знижує дисперсію моделі, хоча може трохи підвищувати зміщення.

Алгоритм методу показано на рисунку 3.4

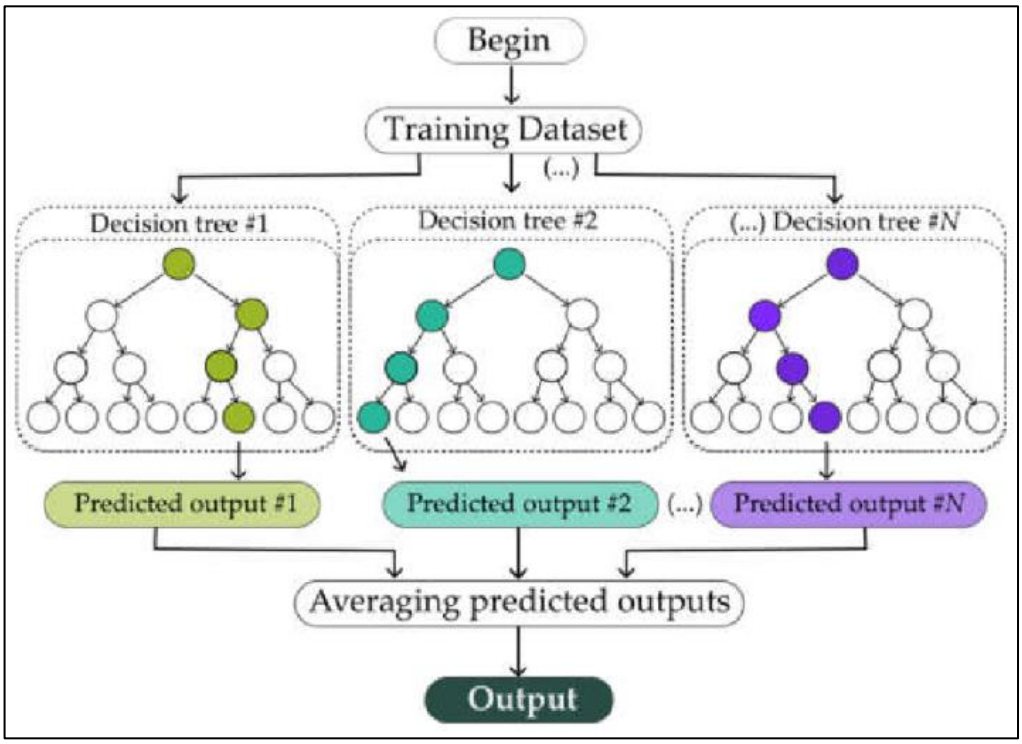


Рисунок 3.4 – Блок-схема роботи алгоритму Extra Tree

Logistic Regression – це статистичний алгоритм класифікації, який моделює ймовірність належності об'єкта до певного класу. Він широко використовується завдяки простоті, інтерпретованості та ефективності для лінійно відокремлюваних даних. Основні етапи роботи алгоритму включають [16]:

1. Ініціалізація моделі:

На початку алгоритм ініціалізує вектор ваг w та вільний член (bias) b. Зазвичай ваги встановлюються малими випадковими значеннями або нулями. Модель задається лінійною комбінацією ознак:

$$z = w^T x + b$$

2. Обчислення логістичної (сигмоїдної) функції:

Для перетворення лінійного виходу z у ймовірність використовується сигмоїдна функція:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Це дозволяє інтерпретувати результат як імовірність належності до класу 1:

$$\hat{y} = P(y = 1|x) = \sigma(w^T x + b)$$

3. Визначення функції втрат:

Для оцінки якості передбачення логістична регресія використовує Log Loss (бінарну крос-ентропію):

$$L(y, \hat{y}) = -(y ln(\hat{y}) + (1 - y)\, ln(1 - \hat{y}))$$

Сумарна функція втрат по всіх об’єктах:

$$J(w, b) = \frac{1}{n} \sum_{i=1}^{n} L(y_i, \hat{y}_i)$$

4. Обчислення градієнтів:

Алгоритм знаходить напрям, у якому потрібно змінити ваги, щоб зменшити втрати. Градієнт для ваг і зміщення:

$$\frac{\partial J}{\partial w} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i) x_i$$

$$\frac{\partial J}{\partial b} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)$$

5. Оновлення ваг (Gradient Descent):

На кожній ітерації алгоритм оновлює ваги за правилом градієнтного спуску:

$$w := w - \eta \frac{\partial J}{\partial w}$$

$$b := b - \eta \frac{\partial J}{\partial b}$$

Де η – коефіцієнт навчання (learning rate).

6. Регуляризація:

Для запобігання перенавчанню використовується L1 або L2 – регуляризація, яка додається до функції втрат:

L2 (Ridge):

$$J_{reg} = J + \lambda \|w\|^2$$

L1 (Lasso):

$$J_{reg} = J + \lambda \|w\|_1$$

Де λ – коефіцієнт регуляризації.

Це допомагає контролювати величини ваг, зменшуючи переадаптацію моделі.

7. Класифікація:

Після навчання модель застосовує поріг (звичайно 0,5):

$$\hat{y} = \begin{cases} 1, \text{якщо } \sigma(w^T x + b) \geq 0{,}5 \\ 0, \text{інше} \end{cases}$$

Отже, таким чином Logistic Regression створює лінійну модель, яка прогнозує ймовірність належності до класу за допомогою сигмоїдної функції. Алгоритм ітеративно оптимізує ваги через зменшення крос-ентропійної функції втрат. За рахунок простоти та інтерпретованості він є одним із базових та найпоширеніших методів класифікації.

Support Vector Machine (SVM) – це потужний алгоритм класифікації, що шукає оптимальну гіперплощину для поділу класів з максимальним зазором (margin). Він ефективний для лінійно та нелінійно відокремлюваних даних, а завдяки використанню ядрових функцій може працювати у складних просторах ознак. Математична модель даного методу виглядає наступним чином [16]:

1. Формування гіперплощини:

Знаходження гіперплощини, яка найкраще розділяє два класи:

$$w^T x + b = 0$$

2. Максимізація відстані між класами:

Алгоритм визначає опорні вектори – точки, найближчі до гіперплощини. Ціль: максимізувати maring – відстань між гіперплощиною та опорними точкаами.

Оптимізаційна задача:

$$\min_{w,b} \frac{1}{2} \|w\|^2$$

при умовах:

$$y_i(w^T x_i + b) \geq 1$$

3. Використання soft-maring (параметр C):

У реальних задачах дані часто не є повністю лінійно відокремленими. Тому вводять штрафний параметр C, який контролює баланс між:

– шириною miring;

– кількістю помилок класифікації.

Модифікована задача оптимізації:

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i$$

4. Ядрові функції:

Для нелінійних даних застосовуються ядрові перетворення, що переводять дані у простір вищої розмірності. Популярними ядрами є: RBF (Gaussian), Polynomial, Sigmoid, Linear.

Ядро обчислює подібність:

$$K(x_i, x_i)$$

без явного підняття вимірності, що забезпечує високу ефективність.

5. Розв’язання оптимізаційної задачі:

Задача зводиться до квадратичної оптимізації (Quadratic Programming):

$$\alpha_i = Lagrange\ multipliers$$

Обчислюються ваги:

$$w = \sum_{i=1}^{n} \alpha_i y_i \, x_i$$

Лише точка з $\alpha_i > 0$ є опорними векторами

6. Прийняття рішення:

Класифікація відбувається за правилом:

$$\hat{y} = \ sin(w^T x + b)$$

Виходячи з вище описаного, можна зробити висновок що Support Vector Machine (SVM) находить оптимальну гіперплощину з максимальним margin, забезпечуючи високу якість класифікації, особливо при використанні ядрових функцій. Це робить даний метод одним із найефективніших методів для складних, нелінійних та високовимірних даних.

MLP (Multi-Layer Perceptron) – це нейронна мережа з одним або кількома прихованими шарами, яка здатна моделювати складні нелінійні залежності. Для навчання використовується алгоритм зворотного поширення помилки (backpropagation). Математична модель даного методу представлена намтупним чином [16]:

1. Ініціалізація мережі:

Визначаються наступні параметри:

– кількість приховних шарів,
– кількість нейронів у кожному шарі,
– функції активації,
– початкові ваги (випадкові малі значення).

2. Прямий прохід (Forward Pass)

Дані передаються від входу до виходу для кожного шару:

$$z = Wx + b$$

$$a = f(z)$$

Де f – функція активації (ReLU, sigmoid, tanh тощо).

На вихідному шарі для класифікації застосовується softmax або sigmoid.

3. Обчислення функції втрат:

Для задач класифікації найчастіше використовується крос-ентропія:

$$L(y, \hat{y}) = -\sum_{k} y_k \, ln(\hat{y}_k)$$

4. Зворотне поширення помилки (Backpropagation):

Алгоритм обчислює похідні функції втрат відносно кожного параметра за правилом:

$$\frac{\partial L}{\partial W}, \qquad \frac{\partial L}{\partial b}$$

Це робиться пошарово від виходу до входу, використовуючи правило ланцюга.

5. Оновлення ваг (Gradient Descent):

Параметри оновлюються:

$$W := W - \eta \frac{\partial L}{\partial W}$$

$$b := b - \eta \frac{\partial L}{\partial b}$$

Де η – коефіцієнт навчання (learning rate).

6. Регуляризація й оптимізація:

MLP часто використовує:

– L2 – регуляризацію,

– dropout,

– оптимізатори Adam, RMSProp, SGD.

7. Ітеративне навчання:

Процес forward → loss → backward → update повторюється багато епох, доки модель не збіжиться.

Таким чином MLP (Multi-Layer Perceptron) є досить потужною нейронною моделлю, яка навчається шляхом зворотного поширення помилки та дозволяє ефективно моделювати складні, нелінійні залежності. Завдяки гнучкості архітектури та можливості роботи з різноманітними активаціями MLP широко використовується у задачах класифікації та регресії.

Алгоритм даного методу зображено на рисунку 3.5.

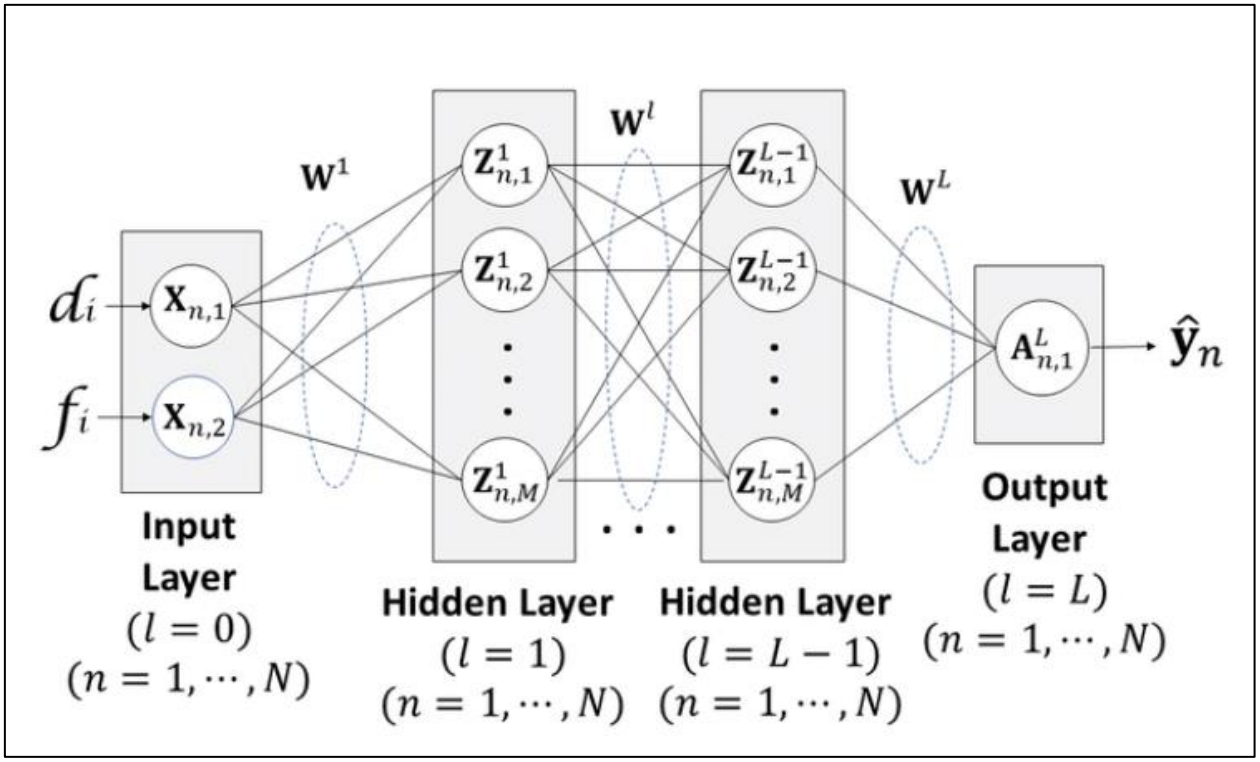


Рисунок 3.5 – Блок-схема методу MLP (Multi-Layer Perceptron)

3.2 Реалізація математичних моделей

Перед початком моделювання та тестування даних проводиться їх первинна обробка: видаляються непотрібні стовпці, категоріальні ознаки кодуються у числовий формат, числові змінні масштабуються для приведення їх до єдиного діапазону. Після цього дані розділяються на тренувальний та тестовий набори, що дозволяє ефективно навчати моделі та оцінювати їх продуктивність. Такий підхід забезпечує правильну підготовку даних і підвищує якість роботи алгоритмів машинного навчання (рис 3.6).

```
df_test.drop(columns='id',inplace=True)
```

```
X = df_train.drop(columns=['price_range','pc','three_g'])
y = df_train[['price_range']]

sts = StandardScaler()
stdscaler = sts.fit_transform(X)
minmax = MinMaxScaler()
minmaxscaler = minmax.fit_transform(X)
col = X.columns
df_stdscaler = pd.DataFrame(stdscaler,columns=col)
df_minmaxscaler = pd.DataFrame(minmaxscaler,columns=col)

X_train,X_test,y_train,y_test = train_test_split(df_minmaxscaler,y,test_size=0.2,random_state=1)
```

```
# Results of prediction
results = pd.DataFrame(columns = ['model', 'f1_train', 'f1_test', 'r2_train', 'r2_test'])
```

```
def performTest(y_pred):
    print("Test Data Metrics:")
    print("Precision : ", precision_score(y_test, y_pred, average = 'micro'))
    print("Recall : ", recall_score(y_test, y_pred, average = 'micro'))
    print("Accuracy : ", accuracy_score(y_test, y_pred))
    print("F1 Score : ", f1_score(y_test, y_pred, average = 'micro'))
    print("R2 Score : ", r2_score(y_test, y_pred))
    cm = confusion_matrix(y_test, y_pred)
    print("\n", cm)
    print("\n")
    print("**"*27 + "\n" + " "* 16 + "Classification Report\n" + "**"*27)
    print(classification_report(y_test, y_pred))
    print("**"*27+"\n")

    cm = ConfusionMatrixDisplay(confusion_matrix = cm, display_labels=['Low cost', 'Medium cost',
'High cost', 'Very high cost'])

    cm.plot( cmap='viridis', xticks_rotation='horizontal')
```

```
def performTrain(y_pred_train):
    print("Train Data Metrics:")
    print("Precision : ", precision_score(y_train, y_pred_train, average='micro'))
    print("Recall : ", recall_score(y_train, y_pred_train, average='micro'))
    print("Accuracy : ", accuracy_score(y_train, y_pred_train))
    print("F1 Score : ", f1_score(y_train, y_pred_train, average='micro'))
    print("R2 Score : ", r2_score(y_train, y_pred_train))
    print("\n")
```

Рисунок 3.6 – Підготовка даних для моделювання та тестування

Після проведеної обробки даних було здійснено прогнозування за допомогою моделей та сформовано відповідні матриці плутанин.

Виконаємо прогнозування, застосувавши модель Random Forest Classifier (рис 3.7).

```
from sklearn.model_selection import GridSearchCV
param_RF = {
    'n_estimators': [100],
    'max_depth': [10],
    'min_samples_split': [2],
}
```

```
%%time
model_tuning = GridSearchCV(estimator=RandomForestClassifier(), param_grid=param_RF, cv=5)
model_tuning.fit(X_train, y_train)

best_params = model_tuning.best_params_
print("Best Parameters:", best_params)
model_rf = RandomForestClassifier(**best_params)
model_rf.fit(X_train, y_train)
```

```
Best Parameters: {'max_depth': 10, 'min_samples_split': 2, 'n_estimators': 100}
CPU times: user 2.82 s, sys: 21.6 ms, total: 2.84 s
Wall time: 2.84 s
```

```
RandomForestClassifier(max_depth=10)
```

Рисунок 3.7 – Приклад коду для побудови та налаштування моделі

Порівняння метрик моделі на тренувальному та тестовому наборах даних зображено на рисунку 3.8.

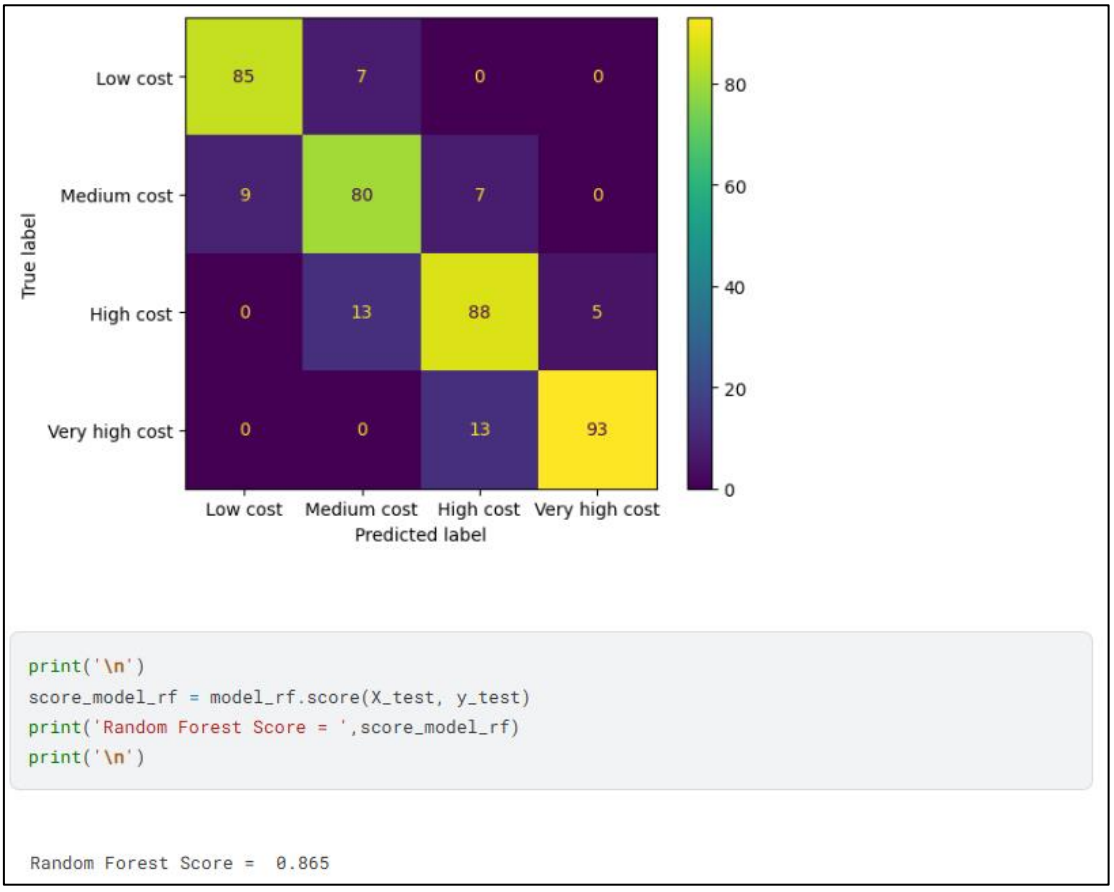
```
Train Data Metrics:
Precision :  0.9975
Recall :  0.9975
Accuracy :  0.9975
F1 Score :  0.9975
R2 Score :  0.9980035711121731


Test Data Metrics:
Precision :  0.865
Recall :  0.865
Accuracy :  0.865
F1 Score :  0.865
R2 Score :  0.8907568125265521
```

Рисунок 3.8 – Показники ефективності моделі на тренувальних і тестових наборах даних.

Результати роботи моделі випадкового лісу показують, що її точність становить 87%. Це означає, що модель правильно класифікувала приблизно 87% випадків у тестовому наборі даних.

Матриця плутанини дозволяє детально оцінити роботу моделі, показуючи кількість вірних та помилкових передбачень для кожного класу (рис 3.9).




```
print('\n')
score_model_rf = model_rf.score(X_test, y_test)
print('Random Forest Score = ',score_model_rf)
print('\n')
```

```
Random Forest Score =  0.865
```

Рисунок 3.9 – Матриця плутанини та результати роботи моделі Random Forest Classifier

Було здійснено налаштування параметрів моделі XGBoost Classifier та виконано прогнозування з її допомогою (рис 3.10).

```
params_XGB ={'n_estimators': [100],
             'max_depth': [5],
             'learning_rate': [0.01],
            'subsample': [0.5],
            'colsample_bytree': [0.5],
            'gamma': [0]}
```

```
%%time
model_tuning = GridSearchCV(estimator=XGBClassifier(), param_grid=params_XGB, cv=5)
model_tuning.fit(X_train, y_train)

best_params = model_tuning.best_params_
print("Best Parameters:", best_params)
model_xgb = XGBClassifier(**best_params)
model_xgb.fit(X_train, y_train)
```

```
Best Parameters: {'colsample_bytree': 0.5, 'gamma': 0, 'learning_rate': 0.01, 'max_depth': 5,
'n_estimators': 100, 'subsample': 0.5}
CPU times: user 9.42 s, sys: 190 ms, total: 9.61 s
Wall time: 2.51 s
```

```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.5, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=0, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.01, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=5, max_leaves=None,
```

Рисунок 3.10 – Приклад коду для побудови та налаштування моделі

Порівняння метрик моделі на тренувальному та тестовому наборах даних показано на рисунку на рисунку 3.11.

```
Train Data Metrics:
Precision :  0.924375
Recall :  0.924375
Accuracy :  0.924375
F1 Score :  0.924375
R2 Score :  0.9396080261432362


Test Data Metrics:
Precision :  0.805
Recall :  0.805
Accuracy :  0.805
F1 Score :  0.805
R2 Score :  0.8422042847605753
```

Рисунок 3.11 – Показники ефективності моделі на тренувальних і тестових наборах даних.

Результати роботи моделі демонструють точність на рівні 0.805%.

Побудована матриця плутанини дозволяє детальніше оцінити ефективність роботи моделі (рис 3.12).

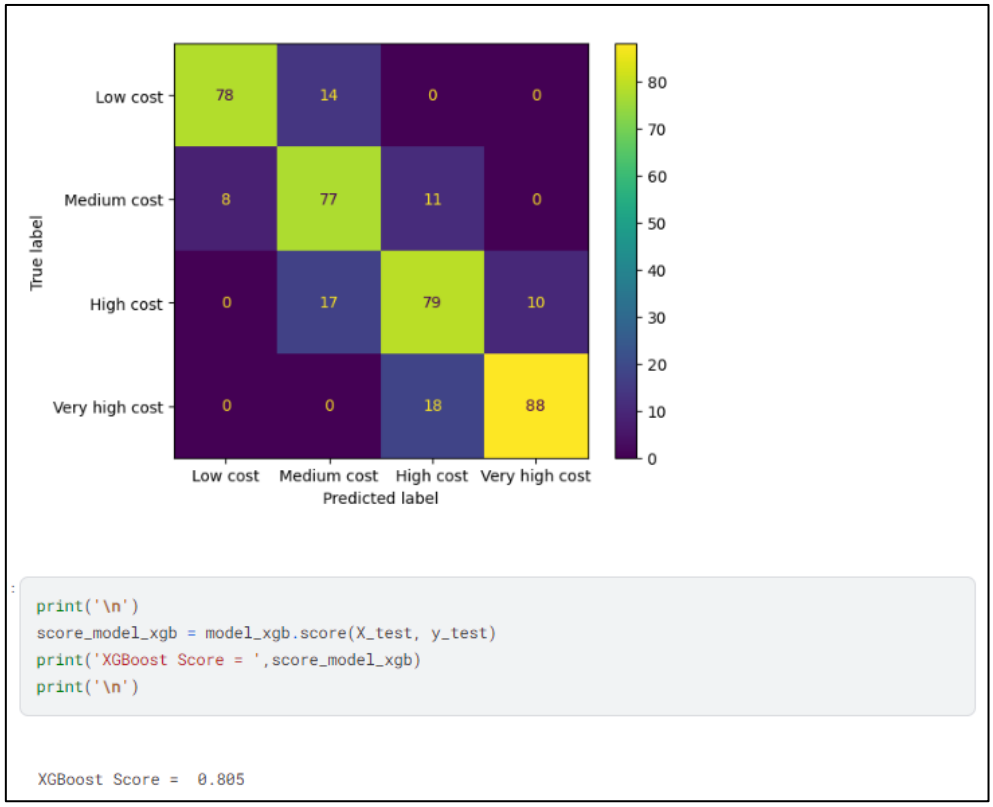


Рисунок 3.12 – Матриця плутанини та результати роботи моделі XGBoost Classifier

Застосовано модель ADABoost Classifier з оптимізованими параметрами. Для визначення найкращих значень параметрів (n_estimators, learning_rate, algorithm) проведено їх систематичний підбір у процесі пошуку оптимальної конфігурації моделі. (рис 3.13).

```
param_ADA = {
    'n_estimators': [350],
    'learning_rate': [1.0],
    'algorithm': ['SAMME.R'],
}
```

```
%%time
model_tuning = GridSearchCV(estimator=AdaBoostClassifier(), param_grid=param_ADA, cv=5)
model_tuning.fit(X_train, y_train)

best_params = model_tuning.best_params_
print("Best Parameters:", best_params)
model_ada = AdaBoostClassifier(**best_params)
model_ada.fit(X_train, y_train)
```

```
Best Parameters: {'algorithm': 'SAMME.R', 'learning_rate': 1.0, 'n_estimators': 350}
CPU times: user 8.98 s, sys: 31.5 ms, total: 9.01 s
Wall time: 9.01 s
```

```
AdaBoostClassifier
AdaBoostClassifier(n_estimators=350)
```

Рисунок 3.13 – Приклад коду для побудови та налаштування моделі

Порівняння метрик моделі на тренувальному та тестовому наборах даних зображено на рисунку 3.14.

```
Train Data Metrics:
Precision :  0.760625
Recall :  0.760625
Accuracy :  0.760625
F1 Score :  0.760625
R2 Score :  0.8088419339905744


Test Data Metrics:
Precision :  0.785
Recall :  0.785
Accuracy :  0.785
F1 Score :  0.785
R2 Score :  0.826020108838583
```

Рисунок 3.14 – Показники ефективності моделі на тренувальних і тестових наборах даних.

Матрицю плутанини для моделі AdaBoost Classifier представлена на рисунку 3.15.

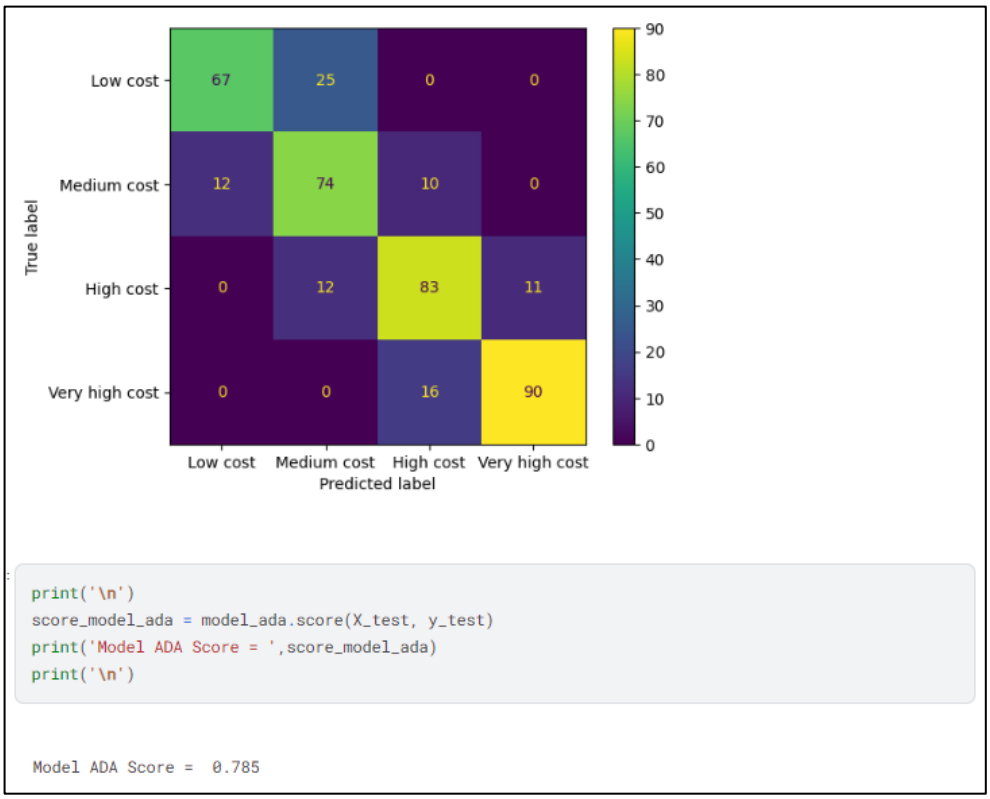


Рисунок 3.15 – Матриця плутанини та результати роботи моделі ADABoost Classifier

Для класифікації тестових даних застосовано модель ExtraTreesClassifier з проведеним пошуком оптимальних параметрів. Після підбору та встановлення найкращих значень параметрів отримано результати роботи моделі. (рис. 3.16).

Рисунок 3.16 – Приклад коду для побудови та налаштування моделі

Порівняння метрик моделі на тренувальному та тестовому наборах даних показано на рисунку 3.17.

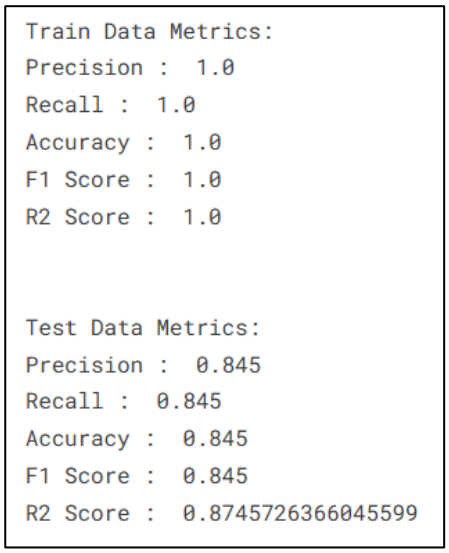


Рисунок 3.17 – Показники ефективності моделі на тренувальних і тестових наборах даних.

Матрицю плутанини для моделі ExtraTreesClassifier зображено на рисунку 3.18.

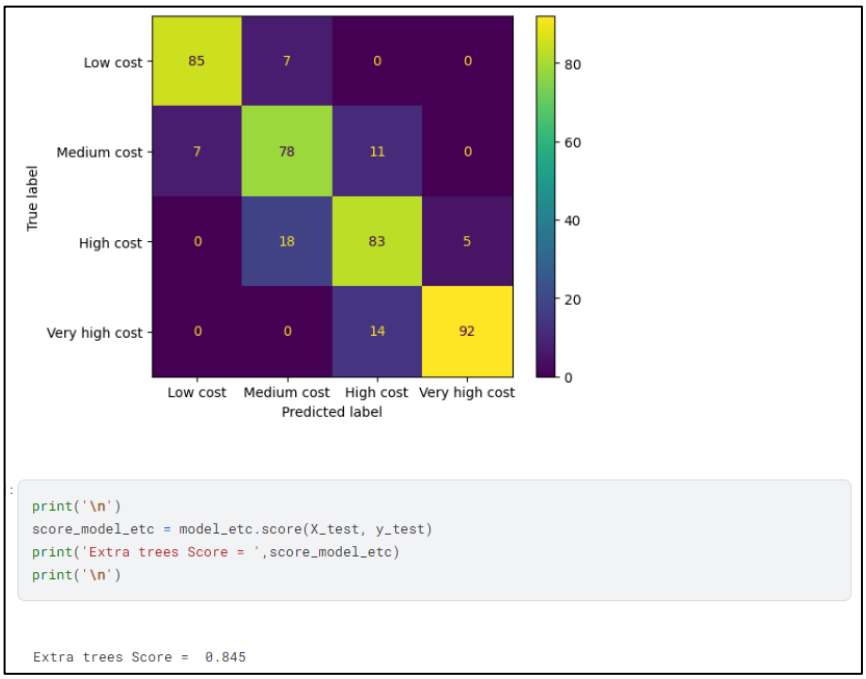


Рисунок 3.18 – Матриця плутанини та результати роботи моделі ExtraTreesClassifier

Виконано налаштування параметрів та прогнозування з використанням моделі Logistic Regression (рис 3.19).

Рисунок 3.19 – Приклад коду для побудови та налаштування моделі

Порівняння метрик моделі на тренувальному та тестовому наборах даних продемонстровано на рисунку 3.20.

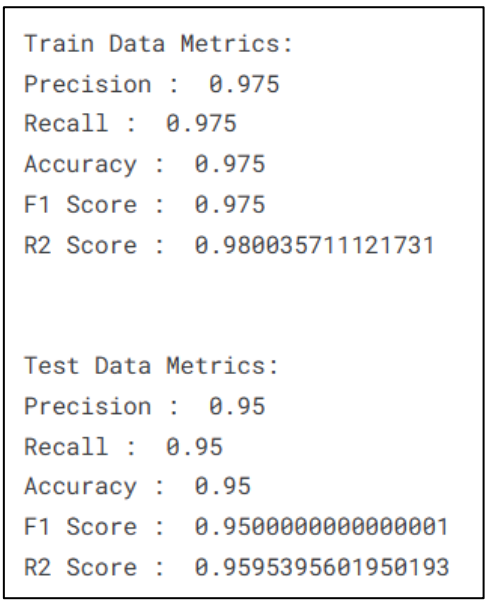


Рисунок 3.20 – Показники ефективності моделі на тренувальних і тестових наборах даних.

Модель продемонструвала точність 95%, що означає правильну класифікацію близько 95% зразків у тестовому наборі.

Матриця плутанини відображає роботу моделі більш детально, демонструючи кількість вірно та помилково класифікованих зразків у кожному класі. (рис 3.21).

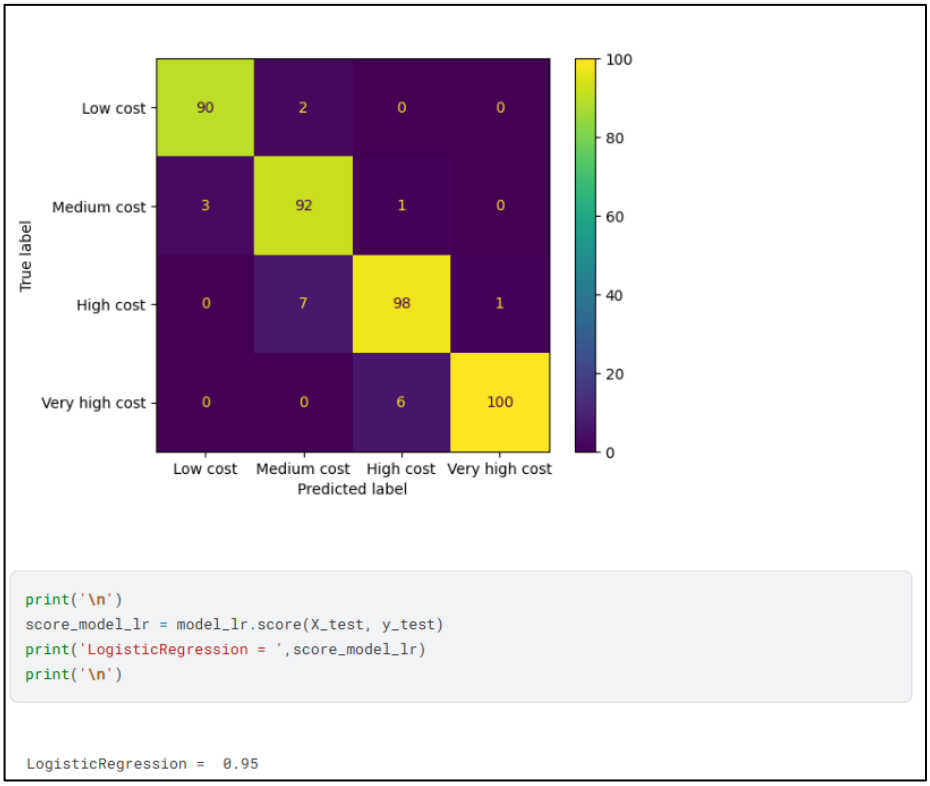


Рисунок 3.21 – Матриця плутанини та результати роботи моделі Logistic Regression

Було виконано оптимізацію параметрів моделі SVM Classifier та здійснено прогнозування на тестових даних за допомогою цієї моделі. (рис 3.22).

```
from sklearn.svm import SVC

params_svm = {
    'C': [0.5, 1, 5],
    'kernel': ['rbf', 'linear'],
    'gamma': ['scale', 'auto']
}
```

```
%%time
model_tuning = GridSearchCV(estimator=SVC(), param_grid=params_svm, cv=5)
model_tuning.fit(X_train, y_train)

best_params = model_tuning.best_params_
print("Best Parameters:", best_params)

model_svm = SVC(**best_params)
model_svm.fit(X_train, y_train)
```

```
Best Parameters: {'C': 5, 'gamma': 'scale', 'kernel': 'linear'}
CPU times: user 5.38 s, sys: 6.88 ms, total: 5.39 s
Wall time: 5.39 s
```

```
SVC
SVC(C=5, kernel='linear')
```

Рисунок 3.22 – Приклад коду для побудови та налаштування моделі

Порівняння метрик моделі на тренувальному та тестовому наборах даних наведено на рисунку 3.23.

```
Train Data Metrics:
Precision :  0.968125
Recall :  0.968125
Accuracy :  0.968125
F1 Score :  0.968125
R2 Score :  0.9745455316802071


Test Data Metrics:
Precision :  0.945
Recall :  0.945
Accuracy :  0.945
F1 Score :  0.945
R2 Score :  0.9554935162145213
```

Рисунок 3.23 – Показники ефективності моделі на тренувальних і тестових наборах даних.

Результати роботи моделі демонструють, що її точність дорівнює 94,5 %, що свідчить про правильну класифікацію більшості випадків у тестовому наборі даних.

Матриця плутанини відображає детальну роботу моделі, демонструючи кількість правильних та неправильних передбачень для кожного класу, що дозволяє оцінити її точність та виявити можливі помилки класифікації. (рис 3.24).

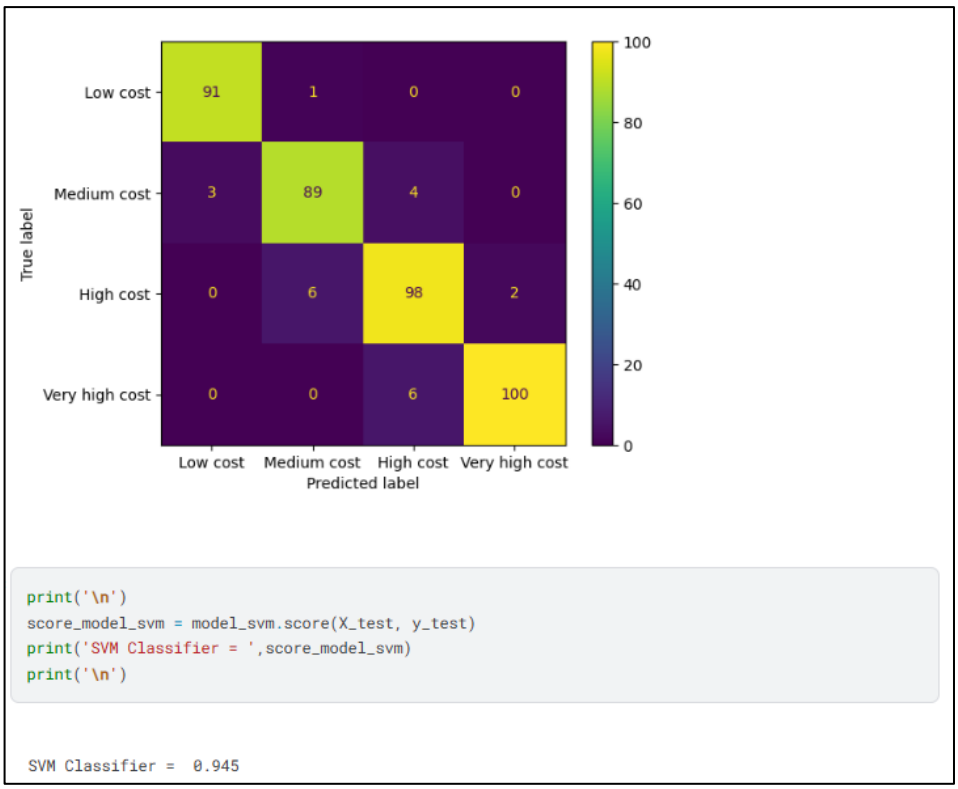
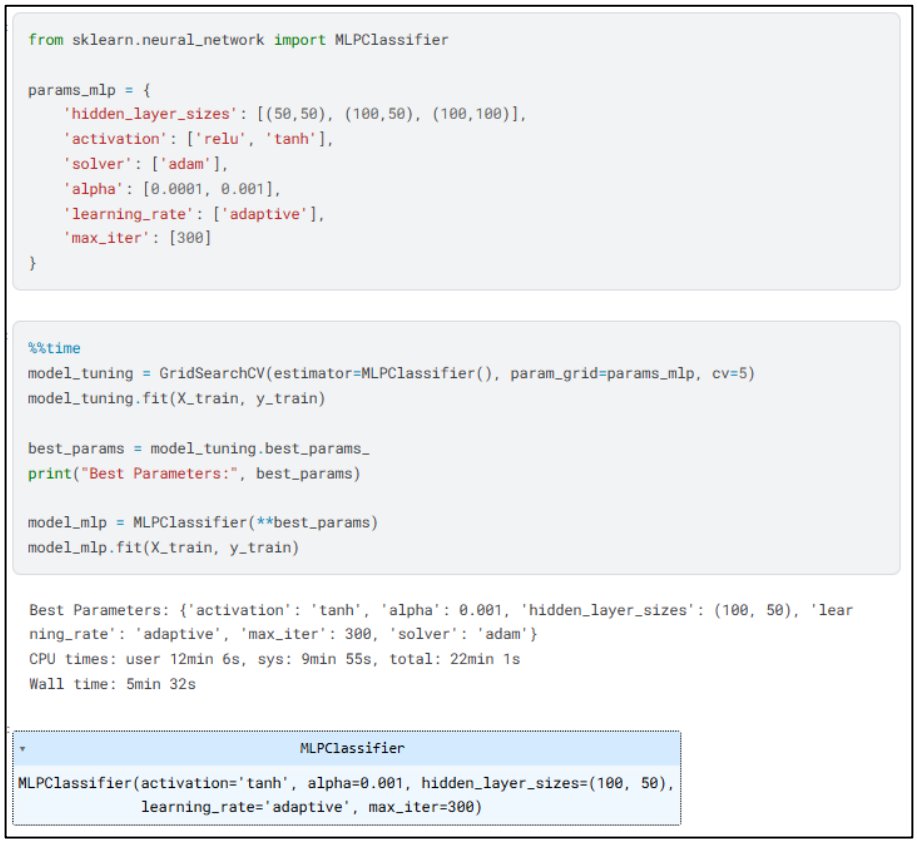


Рисунок 3.24 – Матриця плутанини та результати роботи моделі SVM Classifier

Використано модель Multi-Layer Perceptron (MLP) для класифікації тестових даих із пошуком оптимальних параметрів для підвищення точності та ефективності шляхом перебору [hidden_layer_sizes], [activation], [solver], [alpha], [learning_rate], [max_iter]. Після підбору та встановлення найкращих параметрів моделі виводиться відповідний результат (рис. 3.25).



```
from sklearn.neural_network import MLPClassifier

params_mlp = {
    'hidden_layer_sizes': [(50,50), (100,50), (100,100)],
    'activation': ['relu', 'tanh'],
    'solver': ['adam'],
    'alpha': [0.0001, 0.001],
    'learning_rate': ['adaptive'],
    'max_iter': [300]
}
```

```
%%time
model_tuning = GridSearchCV(estimator=MLPClassifier(), param_grid=params_mlp, cv=5)
model_tuning.fit(X_train, y_train)

best_params = model_tuning.best_params_
print("Best Parameters:", best_params)

model_mlp = MLPClassifier(**best_params)
model_mlp.fit(X_train, y_train)
```

```
Best Parameters: {'activation': 'tanh', 'alpha': 0.001, 'hidden_layer_sizes': (100, 50), 'lear
ning_rate': 'adaptive', 'max_iter': 300, 'solver': 'adam'}
CPU times: user 12min 6s, sys: 9min 55s, total: 22min 1s
Wall time: 5min 32s
```

```
MLPClassifier
MLPClassifier(activation='tanh', alpha=0.001, hidden_layer_sizes=(100, 50),
              learning_rate='adaptive', max_iter=300)
```

Рисунок 3.25 – Приклад коду для побудови та налаштування моделі

Порівняння метрик моделі на тренувальному та тестовому наборах даних зображено на рисунку 3.26.

```
Train Data Metrics:
Precision :  0.995
Recall :  0.995
Accuracy :  0.995
F1 Score :  0.995
R2 Score :  0.9960071422243462


Test Data Metrics:
Precision :  0.9625
Recall :  0.9625
Accuracy :  0.9625
F1 Score :  0.9625000000000001
R2 Score :  0.9696546701462645
```

Рисунок 3.26 – Показники ефективності моделі на тренувальних і тестових наборах даних.

Отриманий результат показує, що точність моделі MLP складає 96%, що підтверджує її високу ефективність у класифікації тестового набору та здатність точно прогнозувати цінові сегменти мобільних пристроїв.

Матриця плутанини для моделі Multi-Layer Perceptron (MLP) представлена на рисунку 3.27, що дозволяє детально оцінити роботу моделі, показуючи кількість правильних та неправильних прогнозів для кожного класу.

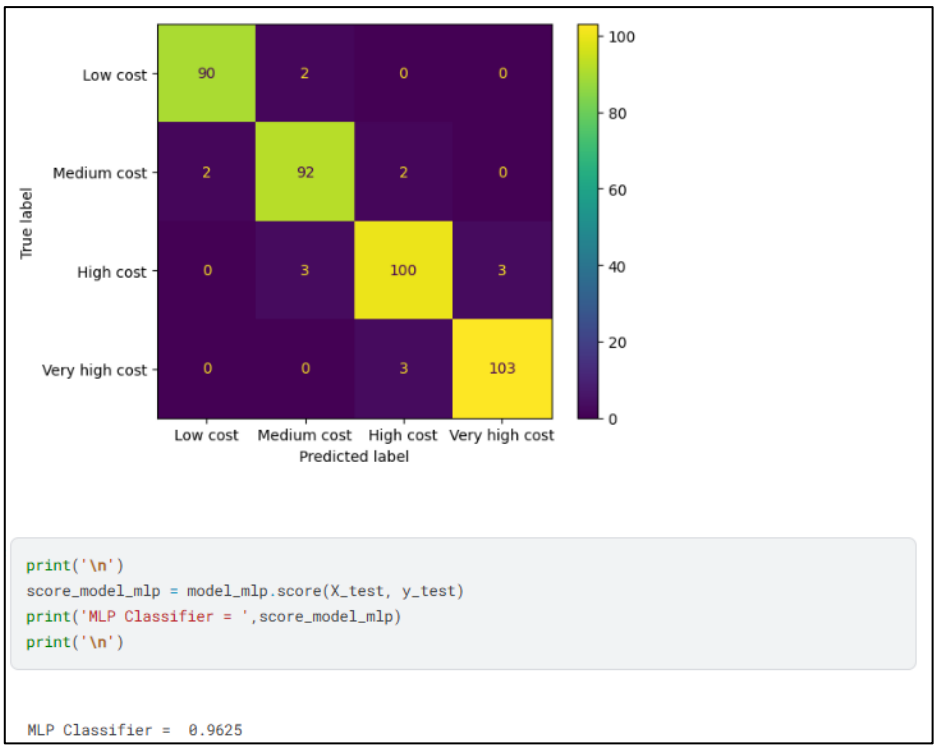


Рисунок 3.27 – Матриця плутанини та результати роботи моделі Multi-Layer Perceptron

На наступному етапі проведено аналіз результатів роботи всіх застосованих моделей машинного навчання, який наведено на рисунку 3.28, що дозволяє порівняти їх точність та ефективність у прогнозуванні цінових сегментів мобільних пристроїв.

	model	f1_train	f1_test	r2_train	r2_test	short
0	RandomForest Classifier	0.9975	0.865	0.998004	0.890757	RF
1	XGB Classifier	0.924375	0.805	0.939608	0.842204	XGB
2	ADABoost Classifier	0.760625	0.785	0.808842	0.82602	ADA
3	ExtraTrees Classifier	1.0	0.845	1.0	0.874573	ExT
4	Logistic Regression	0.975	0.95	0.980036	0.95954	LR
5	SVM Classifier	0.968125	0.945	0.974546	0.955494	SVM
6	MLP Classifier	0.995	0.9625	0.996007	0.969655	MLP

Рисунок 3.28 – Результати роботи застосованих моделей

На основі даних, представлених у таблиці результатів, можна зробити висновок, що MLP Classifier є найефективнішою моделлю для передбачення ціни мобільних телефонів, оскільки вона забезпечує високу точність та мінімальний розрив між тренувальними й тестовими даними.

Модель SVM Classifier також продемонструвала високі результати й не має ознак перенавчання, проте все ж дещо поступається MLP за рівнем точності. Logistic Regression, попри свою простоту та відсутність складних параметрів, показала стабільну й передбачувану роботу, що свідчить про її здатність добре адаптуватися до даних без перенавчання.

У той же час моделі RandomForest, ExtraTrees та XGBoost продемонстрували явне перенавчання, оскільки різниця між їхніми тренувальними та тестовими показниками перевищує 10%, що значно знижує їхню здатність до узагальнення, а от модель AdaBoost Classifier не перенавчилась, проте виявилась недостатньо ефективною порівняно з іншими алгоритмами.

Таким чином, найоптимальнішим вибором для вирішення задачі передбачення цін на мобільні телефони є модель MLP Classifier, тоді як моделі SVM та Logistic Regression можуть розглядатися як якісні альтернативи завдяки своїй стабільності та відсутності перенавчання.

3.3 Проєктування та моделювання інформаційної технології аналізу та передбачення ціни на мобільні телефони

Блок-схему алгоритму інформаційної технології представлено на рисунку 3.29, який демонструє покроковий процес обробки даних, налаштування та застосування моделей машинного навчання для прогнозування цін мобільних телефонів. Основні етапи алгоритму включають:

1. Аналіз отриманих вхідних даних;
2. Візуалізацію даних за допомогою графіків для виявлення залежностей між ознаками та цінами;
3. Попередню обробку даних та перевірку на наявність пропущених значень;
4. Розподіл даних на тренувальні та тестові набори;
5. Вибір відповідних моеделй машинного навчання для прогнозування цін;
6. Оптимазацію параметрів обраних моделей;
7. Оцінку точності моделей на тренувальному наборі даних;
8. Формування прогнозів та побудову графічних відображень результатів.

Таким чином, даний підхід забезпечує систематичну роботу з даними та ефективне використання моделей для передбачення цінових сегментів мобільних пристроїв.

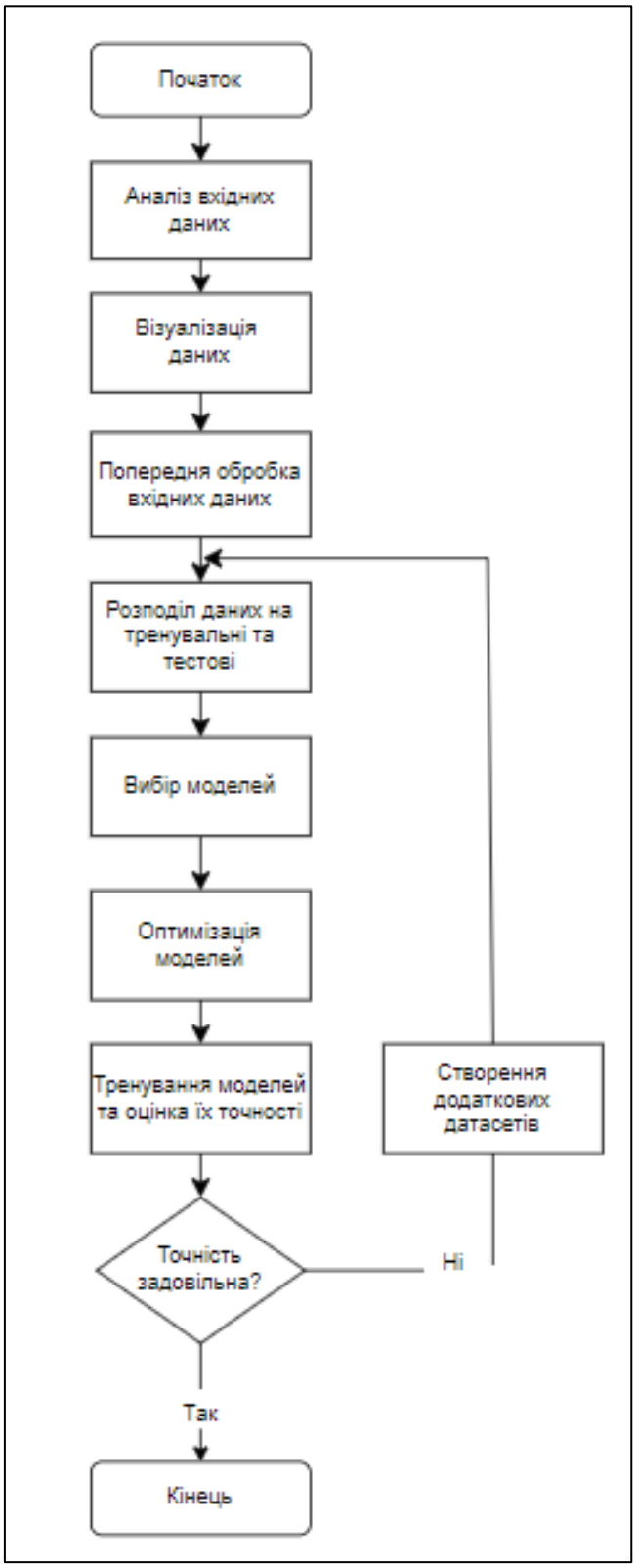


Рисунок 3.29 – Блок-схема алгоритму інформаційної технології передбачення ціни телефонів методами машинного навчання

Структурну організацію цієї інформаційної технології, а також взаємодію між користувачем, обчислювальним середовищем Kaggle та хмарною інфраструктурою подано на UML-діаграмі розгортання (Deployment Diagram), що наведена на рисунку 3.30.

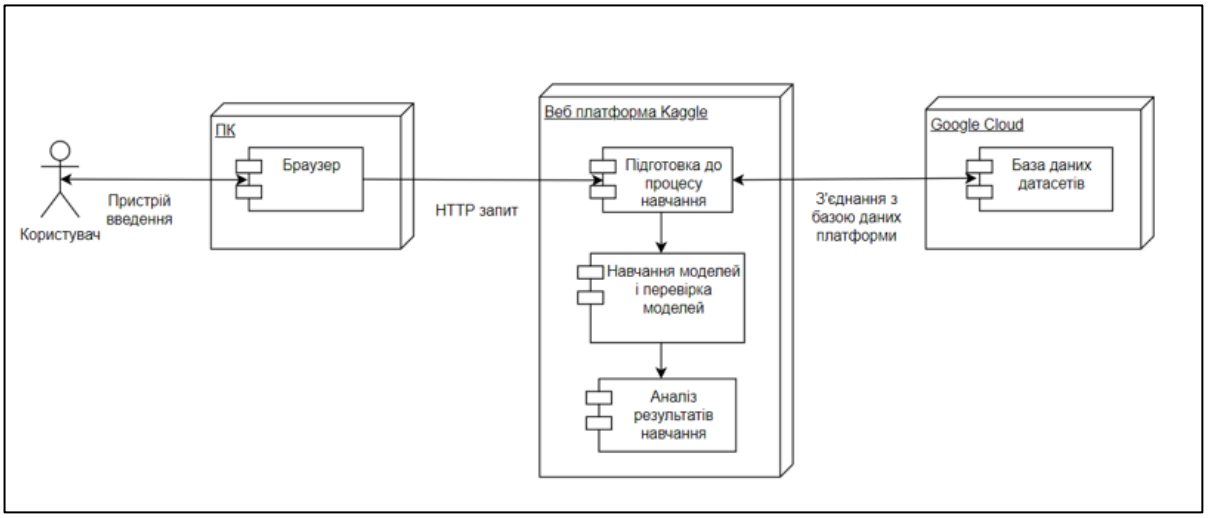


Рисунок 3.31 – UML-діаграма розгортання (Deployment Diagram) інформаційної технології

Наведена UML-діаграма розгортання відображає архітектуру інформаційної технології та демонструє розподіл функціональних компонентів між основними фізичними вузлами системи. На діаграмі виділено три ключові елементи інфраструктури: користувацький персональний комп’ютер, веб-платформу Kaggle та хмарну інфраструктуру Google Cloud, у якій зберігається датасет з даними.

Користувач взаємодіє із системою через веббраузер, який слугує інтерфейсом доступу до обчислювального середовища Kaggle. У межах Kaggle виконуються всі етапи машинного навчання, зокрема завантаження та попередня обробка даних, формування тренувальних і тестових вибірок, навчання моделей та аналіз отриманих результатів. Доступ до датасету забезпечується через хмарне сховище Google Cloud, що гарантує стабільність і швидкодію роботи системи.

Таким чином, UML-діаграма розгортання демонструє логічну та фізичну структуру інформаційної технології, а також відображає взаємодію між її компонентами, що забезпечує цілісний процес обробки даних та виконання моделей машинного навчання.

На рисунку 3.31 подано UML-діаграму варіантів використання інформаційної технології передбачення ціни мобільних телефонів. Діаграма відображає взаємодію користувача з основними функціональними можливостями системи та демонструє два ключові напрямки роботи — обробку даних і роботу з моделями машинного навчання.

Головним актором системи виступає користувач, який ініціює всі доступні випадки використання. У межах підсистеми *Робота з даними* користувач має можливість:

– завантажувати нові набори даних, необхідні для подальшого навчання моделей;

– виконувати попередню обробку даних, що включає очищення, нормалізацію та підготовку даних до навчання;

– візуалізувати дані, що дозволяє провести первинний аналіз та виявити потенційні залежності між ознаками.

Другий функціональний блок — *Робота з моделями* — містить такі варіанти використання:

– обрати параметри моделі, що дає змогу налаштувати алгоритм машинного навчання відповідно до типу задачі;

– запустити процес навчання, у ході якого система формує модель прогнозування;

– оцінити точність моделі шляхом аналізу метрик якості;

– передбачити ціну телефону на основі побудованої моделі.

Діаграма демонструє логічну послідовність дій та відображає, як користувач здійснює повний цикл роботи з інформаційною технологією — від завантаження й обробки даних до налаштування, навчання моделей та

отримання прогнозів. Таким чином, UML Use Case діаграма визначає основну функціональність системи та забезпечує чітке розуміння ролі користувача у процесі передбачення вартості мобільних пристроїв.

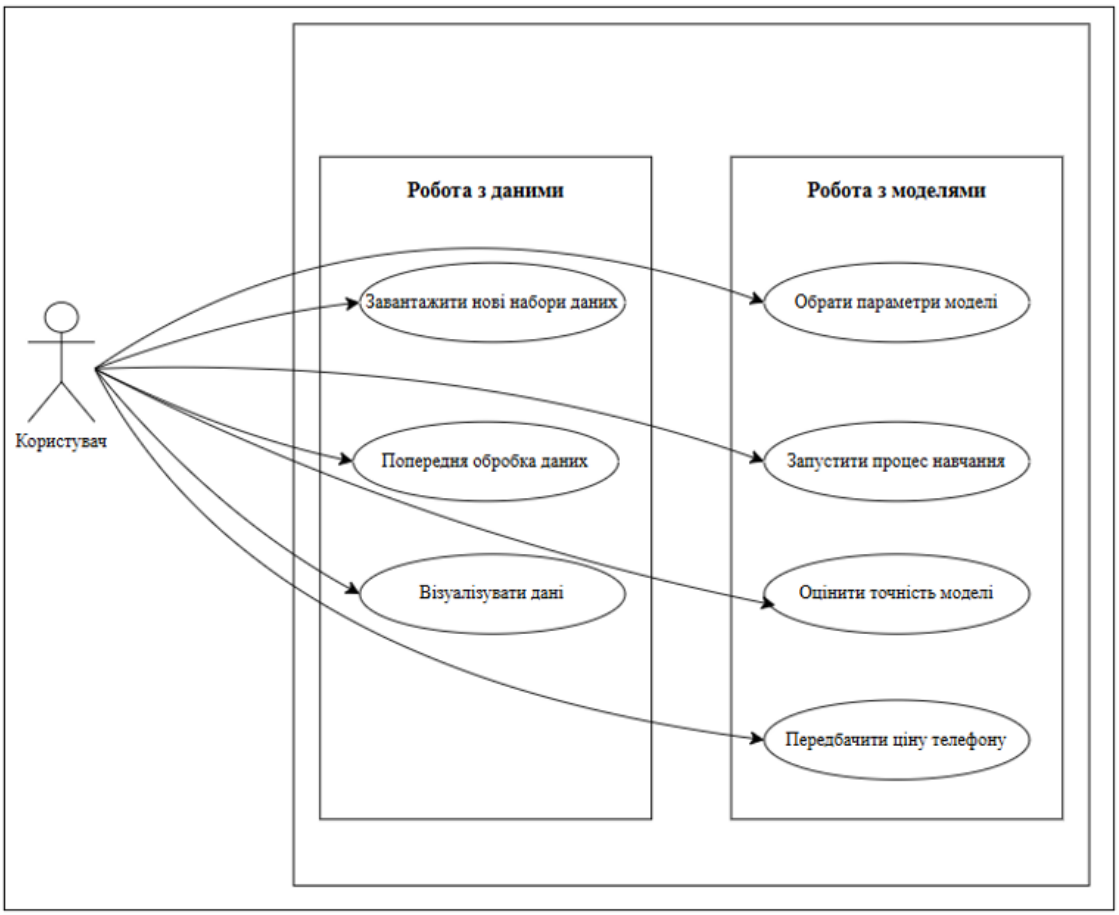


Рисунок 3.31 – UML-діаграма варіантів використання інформаційної технології

3.4 Висновки

У цьому розділі розроблення та комплексне тестування інформаційної технології для аналізу та передбачення цін мобільних телефонів із застосуванням методів машинного навчання на основі датасету Mobile Price Classifier. Було описано процес створення та впровадження математичних моделей, продемонстровано їх практичну реалізацію, наведено результати

прогнозування та сформульовано висновки. Проведений аналіз показав, що найвищу точність серед застосованих моделей досяг Multi-Layer Perceptron (MLP) Classifier, забезпечивши значення точності прогнозу 0.996.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Інформаційна технологія аналізу та передбачення ціни телефонів методами машинного навчання» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями [17].

Таблиця 4.1 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	5	4	5
2. Ринкові переваги (наявність аналогів)	4	4	3
3. Ринкові переваги (ціна продукту)	4	4	4
4. Ринкові переваги (технічні властивості)	3	4	4
5. Ринкові переваги (експлуатаційні витрати)	4	4	4
6. Ринкові перспективи (розмір ринку)	3	3	3
7. Ринкові перспективи (конкуренція)	3	2	2
8. Практична здійсненність (наявність фахівців)	4	4	4

Продовження таблиці 4.1

9. Практична здійсненність (наявність фінансів)	2	2	2
10. Практична здійсненність (необхідність нових матеріалів)	3	4	4
11. Практична здійсненність (термін реалізації)	4	4	4
12. Практична здійсненність (розробка документів)	3	2	3
Сума балів	42	41	42
Середньоарифметична сума балів $СБ_с$	41,7		

За результатами розрахунків, наведених в таблиці 4.1, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в [17].

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія аналізу та передбачення ціни телефонів методами машинного навчання» становить 41,7 бала, що, відповідно до [17], свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки високий).

4.2 Розрахунок узагальненого коефіцієнта якості розробки

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення розрахуємо за формулою [18]:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i \,, \tag{4.1}$$

де k – кількість найбільш важливих технічних показників, які впливають на якість нового технічного рішення;

α_i – коефіцієнт, який враховує питому вагу *i-го* технічного показника в загальній якості розробки. Коефіцієнт α_i визначається експертним шляхом і при цьому має виконуватись умова $\sum_{i=1}^{k} \alpha_i = 1$;

β_i – відносне значення *i-го* технічного показника якості нової розробки.

Результати порівняння зведемо до таблиці 4.2.

Таблиця 4.2 – Порівняння основних параметрів розробки та аналога.

Показники (параметри)	Одиниця вимірю-вання	Аналог	Проектований продукт	Відношення параметрів нової розробки до аналога	Питома вага показника
1. Кількість використаних моделей машинного навчання	од.	4	7	1,75	0,3
2. Попередня обробка та очистка даних	од.	1	1	1	0,1
3. Точність передбачення	%	91	99,6	1,09	0,25
4. Кількість графіків розвідувального аналізу	од.	5	15	3	0,2
5. Кількість використаних багатошарових нейронних мереж	од.	1	5	5	0,15

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення складе:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i = 1{,}75 \cdot 0{,}3 + 1 \cdot 0{,}1 + 1{,}09 \cdot 0{,}25 + 3 \cdot 0{,}2 + 5 \cdot 0{,}15 = 2{,}25.$$

Отже за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 2,25 рази.

4.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Інформаційна технологія аналізу та передбачення ціни телефонів методами машинного навчання», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

4.3.1 Витрати на оплату праці

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [17]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (4.2)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=21 дні.

$$З_о = 24850{,}00 \cdot 11 / 21 = 13016{,}67 \text{ (грн.)}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 4.3 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту (проєктний менеджер)	24850,00	1183,33	11	13016,67
Інженер-розробник автоматизованих інформаційних систем	22500,00	1071,43	21	22500,00
Інженер-розробник програмного забезпечення	23600,00	1123,81	35	39333,33
Консультант (маркетолог бізнес-аналітик)	23000,00	1095,24	7	7666,67
Технік 1-ї категорії	9150,00	435,71	14	6100,00
Всього				88616,67

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Інформаційна технологія передбачення ціни телефонів методами машинного навчання» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i , \qquad (4.3)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} , \qquad (4.4)$$

де $М_М$ – розмір мінімальної місячної заробітної плати, приймемо *Мм*=8000,00 грн;

$К_і$ – коефіцієнт міжкваліфікаційного співвідношення (табл. Б.2, додаток Б) [17];

$К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок;

$Т_р$ – середнє число робочих днів в місяці, приблизно $Т_р$ = 21 дн;

$t_{зм}$ – тривалість зміни, год.

$С_1$ = 8000,00 · 1,10 · 1,15 / (21 · 8) = 60,24 (грн.)

$З_{р1}$ = 60,24 · 8,00 = 481,90 (грн.)

Таблиця 4.4 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Установка обладнання для забезпечення моделювання методів прогнозування	8,00	2	1,10	60,24	481,90
Підготовка робочого місця інженера-розробника інформаційних систем	6,50	4	1,50	82,14	533,93
Інсталяція програмного забезпечення моделювання прогнозних даних	7,50	3	1,35	73,93	554,46
Компіляція програмних блоків	12,00	2	1,10	60,24	722,86
Налагодження програмних блоків	7,20	5	1,70	93,10	670,29

Продовження таблиці 4.4

Формування бази даних нейромережі	16,00	2	1,10	60,24	963,81
Попереднє тренування нейромережі	11,00	2	1,10	60,24	662,62
Тестування методів прогнозування на основі нейромережі	6,20	2	1,10	60,24	373,48
Інсталяція цифрових моделей нейромережі	7,65	4	1,50	82,14	628,39
Всього					5591,74

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.5)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод} = (88616,67 + 5591,74) \cdot 11 / 100\% = 10362,92$ (грн.)

4.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.6)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (88616,67 + 5591,74 + 10362,92) · 22 / 100% = 23005,69 (грн.)

4.3.3 Сировина та матеріали

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (4.7)$$

де $Н_j$ – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 ... 1,15*);

$В_j$ – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$М_1$ = 3,000 · 202,00 · 1,1 - 0 · 0 = 666,60 грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 4.5 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 од, грн	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Офісний папір Cristal A4 500	202,00	3	0	0	666,60

Продовження таблиці 4.5

Папір для записів Cristal LightPapers 65 A5	100,00	4	0	0	440,00
Органайзер офісний Cristal	195,00	4	0	0	858,00
Набір офісний Cristal Base	204,00	3	0	0	673,20
Картридж для принтера Canon LBP5000	1420,00	1	0	0	1562,00
Диск оптичний CD-R	31,00	2	0	0	68,20
Flesh-пам'ять Kingston 128 GB 10	369,00	1	0	0	405,90
Тека для паперів	72,00	6	0	0	475,20
Інші матеріали	140,00	1	0	0	154,00
Всього					5303,10

4.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Інформаційна технологія передбачення ціни телефонів методами машинного навчання», розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \qquad (4.8)$$

де H_j – кількість комплектуючих j-го виду, шт.;

$Ц_j$ – покупна ціна комплектуючих j-го виду, грн;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 … 1,15*).

$К_в = 1 \cdot 3599{,}00 \cdot 1{,}1 = 3958{,}90$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.6 – Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
Зовнішній жорсткий диск 2.5" 2TB Seagate (STGD2000200)	1	3599,00	3958,90
Концентратор Defender SEPTIMA SLIM (83505)	1	400,00	440,00
Кабель для передачі даних USB to COM 1.0m Patron (CAB-PN-USB-COM)	1	354,00	389,40
Маршрутизатор Asus RT-AX57 (90IG06Z0-MO3C00 / 90IG06Z0-MU2C00)	1	4499,00	4948,90
Всього			9737,20

4.3.5 Спецустаткування для наукових (експериментальних) робіт

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ , \qquad (4.9)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.і}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, (K_i = *1,10…1,12);*

k – кількість найменувань устаткування.

$B_{спец}$ = *48499,00 · 1 · 1,05 = 50923,95* (грн.)

Отримані результати зведемо до таблиці:

Таблиця 4.7 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Серверне обладнання на основі ЕОМ DELUX F43-A71BC	1	48499,00	50923,95
Всього			50923,95

4.3.6 Програмне забезпечення для наукових (експериментальних) робіт

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.і} \cdot K_i \ , \qquad (4.10)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.і}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (K_i = *1,10…1,12);*

k – кількість найменувань програмних засобів.

$B_{прг} = 9499,00 \cdot 1 \cdot 1,05 = 9973,95$ (грн.)

Отримані результати зведемо до таблиці:

Таблиця 4.8 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Тестова база даних нейромережі датасет "Mobile Price Classification"	1	9499,00	9973,95
Математичне середовище MatLab 12	1	9860,00	10353,00
Прикладне ПЗ Mathematica	1	7580,00	7959,00
Хмарний хостинг Cloud Professiona для отримання данних (доступ, місяць)	2	560,00	1176,00
Абонентна плата доступу до мережі Internet (за місяць)	2	400,00	840,00
Платформа Kaggle (доступ)	2	40,38	84,80
Підтримка домену (місяць)	2	40,38	84,80
VDS (Virtual Dedic Server) Сервер для середовища розробки та тестування	2	3450,00	7245,00
Всього			37716,55

4.3.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (4.11)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (48309{,}00 \cdot 2) / (3 \cdot 12) = 2683{,}83$ (грн.)

Проведені розрахунки зведемо до таблиці.

Таблиця 4.9 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Програмно-аналітичний комплекс комп'ютер ARTLINE Business B57 Windows 11 Pro (B57v88Win)	48309,00	3	2	2683,83
Персональний комп'ютер Ноутбук Lenovo IdeaPad Slim 3 15ABR8 (82XM013KRA) Arctic Grey / 15.6" IPS Full HD / AMD Ryzen 5 5625U / RAM 16 ГБ / SSD 512 ГБ	26299,00	3	2	1461,06

Продовження таблиці 4.9

Спеціалізоване робоче місце розробника інформаційної технології	9500,00	5	1	158,33
Пристрій виводу текстової інформації	10499,00	4	1	218,73
Оргтехніка Office	10599,00	5	1	176,65
Приміщення лабораторії розробки та дослідження	519000,00	35	1	1235,71
ОС Windows 11	8789,00	3	1	244,14
Прикладний пакет Microsoft Office 2021 Professional Plus	7759,00	3	1	215,53
Мережеве обладнання передачі цифрових даних	6799,00	4	1	141,65
Всього				6535,63

4.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (4.12)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; приймемо $Ц_е = 12,56$ грн;

$К_{вп і}$ – коефіцієнт, що враховує використання потужності, $К_{вп і} < 1$;

$\eta_і$ – коефіцієнт корисної дії обладнання, $\eta_і < 1$.

$B_е = 0,25 \cdot 280,0 \cdot 12,56 \cdot 0,95 / 0,97 = 879,20$ (грн.)

Проведені розрахунки зведемо до таблиці.

Таблиця 4.10 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Програмно-аналітичний комплекс комп'ютер ARTLINE Business B57 Windows 11 Pro (B57v88Win)	0,25	280,0	879,20
Персональний комп'ютер Ноутбук Lenovo IdeaPad Slim 3 15ABR8 (82XM013KRA) Arctic Grey / 15.6" IPS Full HD / AMD Ryzen 5 5625U / RAM 16 ГБ / SSD 512 ГБ	0,08	280,0	281,34
Спеціалізоване робоче місце розробника інформаційної технології	0,07	280,0	246,18
Пристрій виводу текстової інформації	0,12	2,0	3,01
Оргтехніка Office	0,52	1,8	11,76
Маршрутизатор Asus RT-AX57 (90IG06Z0-MO3C00 / 90IG06Z0-MU2C00)	0,10	280,0	351,68
Серверне обладнання на основі EOM DELUX F43-A71BC	0,02	280,0	70,34
Мережеве обладнання передачі цифрових даних	0,03	280,0	105,50
Всього			1949,01

4.3.9 Службові відрядження

Витрати за статтею «Службові відрядження» відсутні.

4.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (4.13)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 30%.

$B_{сп} = (88616,67 + 5591,74) \cdot 30 / 100\% = 28262,52$ (грн.)

4.3.11 Інші витрати

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.14)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 65%.

$I_в = (88616,67 + 5591,74) \cdot 65 / 100\% = 61235,46$ (грн.)

4.3.12 Накладні (загальновиробничі) витрати

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.15)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 115%.

$B_{нзв} = (88616,67 + 5591,74) \cdot 115 / 100\% = 108339,67$ (грн.)

Витрати на проведення науково-дослідної роботи на тему «Інформаційна технологія аналізу та передбачення ціни телефонів методами машинного навчання» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + A_{обл} + B_е + B_{св} + B_{сп} + I_в + B_{нзв}. \quad (4.18)$$

$B_{заг}$= 88616,67 +5591,74 +10362,92 +23005,69 +5303,10 +9737,20 +50923,95 + 37716,55 + 6535,63 +1949,01 +0,00 +28262,52 +61235,46 +108339,67 = 437580,11 (грн.)

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (4.16)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,95.

ЗВ = 437580,11 / 0,95 = 460610,64 (грн.)

4.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

Результати дослідження проведені за темою «Інформаційна технологія аналізу та передбачення ціни телефонів методами машинного навчання» передбачають комерціалізацію протягом 4-х років реалізації на ринку.

В цьому випадку основу майбутнього економічного ефекту будуть формувати:

ΔN – збільшення кількості споживачів яким надається відповідна інформаційна послуга у періоди часу, що аналізуються;

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	1100	1800	2000	500

N – кількість споживачів яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки, приймемо 22000 осіб;

$Ц_б$ – вартість послуги у році до впровадження інформаційної системи, приймемо 7659,00 грн;

$\pm\Delta Ц_о$ – зміна вартості послуги від впровадження результатів, приймемо 254,60 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [17]:

$$\Delta П_i = (\pm \Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.17)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2025 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ =42%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2025 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (254{,}60 \cdot 22000{,}00 + 7913{,}60 \cdot 1100) \cdot 0{,}83 \cdot 0{,}42 \cdot (1 - 0{,}18/100\%) = 4089460{,}96$ грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 = (254{,}60 \cdot 22000{,}00 + 7913{,}60 \cdot 2900) \cdot 0{,}83 \cdot 0{,}42 \cdot (1 - 0{,}18/100\%) = 8161275{,}34$ грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 = (254{,}60 \cdot 22000{,}00 + 7913{,}60 \cdot 4900) \cdot 0{,}83 \cdot 0{,}42 \cdot (1 - 0{,}18/100\%) = 12685513{,}54$ грн.

Збільшення чистого прибутку 4-го року:

$\Delta П_4 = (254{,}60 \cdot 22000{,}00 + 7913{,}60 \cdot 5400) \cdot 0{,}83 \cdot 0{,}42 \cdot (1 - 0{,}18/100\%) = 13816573{,}09$ грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.18)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,12;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП =$
$4089460,96/(1+0,12)^1+8161275,34/(1+0,12)^2+12685513,54/(1+0,12)^3+$
$+13816573,09/(1+0,12)^4=3651304,43+6506118,73+9029297,97+8780681,98=$
$=27967403,12$ грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \quad (4.19)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$ =2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 460610,64 грн.

$PV = k_{інв} \cdot ЗВ = 2 \cdot 460610,64 = 921221,28$ грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \quad (4.20)$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 27967403,12 грн;

PV – теперішня вартість початкових інвестицій, 921221,28 грн.

$E_{абс} = ПП - PV$ = 27967403,12 - 921221,28 = 27046181,84 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (4.21)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 27046181,84 грн;

PV – теперішня вартість початкових інвестицій, 921221,28 грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+27046181,84/921221,28)^{1/4} = 1,35.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \quad (4.22)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2025 році в Україні d =0,11;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,3.

$\tau_{мін}$ = 0,11+0,3 = 0,41 < 1,35 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Інформаційна технологія передбачення ціни телефонів методами машинного навчання» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \quad (4.23)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 1,35 = 0,74 р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

4.5 Висновки

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія аналізу та передбачення ціни телефонів методами машинного навчання» становить 41,7 бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки високий).

При оцінюванні за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 2,25 рази.

Також термін окупності становить 0,74 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційна технологія аналізу та передбачення ціни телефонів методами машинного навчання».

ВИСНОВКИ

У рамках виконання кваліфікаційної магістерської роботи проведено дослідження та прогнозування факторів, що впливають на вартість мобільних телефонів, із застосуванням методів машинного навчання. Було здійснено детальний аналіз об’єкта дослідження та методів прогнозування, підкреслено необхідність використання алгоритмів машинного навчання, здатних враховувати різноманітні фактори та їх взаємозв’язки для підвищення точності передбачень.

Особлива увага була приділена вибору оптимальної інформаційної технології та налаштувань для вирішення поставленої задачі. Для реалізації моделей машинного навчання обрана мова програмування Python, яка надає широкий спектр інструментів для побудови та оцінки моделей. Проведено розвідувальний аналіз даних, включно з побудовою кореляційної матриці, що дозволило оцінити характеристики набору даних та виявити взаємозв’язки між ознаками.

Для прогнозування цін було застосовано кілька моделей машинного навчання. Найвищу точність показала модель Multi-Layer Perceptron (MLP) Classifier — 0.996, що робить її найбільш ефективною для передбачення факторів, що впливають на ціни мобільних пристроїв.

Загалом результати дослідження підтверджують доцільність використання методів машинного навчання для прогнозування цін на мобільні телефони. Отримана модель може бути застосована для передбачення цін у майбутньому, що сприятиме прийняттю більш обґрунтованих рішень у сфері електронної комерції.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Нереуцький О. В., Козачко О. М., «РОЗВІДУВАЛЬНИЙ АНАЛІЗ ДАНИХ ДЛЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ АНАЛІЗУ ТА ПЕРЕДБАЧЕННЯ ЦІНИ ТЕЛЕФОНІВ МЕТОДАМИ МАШИННОГО НАВЧАННЯ». *Всеукраїнська науково-практична інтернет-конференція «Молодь в науці: дослідження, проблеми, перспективи» (Вінниця, 2025-2026 рр.)*. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2026/paper/view/26567/21954 (дата звернення: 08.11.2025)

2. Predicting the price of phones using machine learning methods. URL: https://galinfo.com.ua/news/vartist_smartfonu_shcho_vplyvaie_na_formuvannya_tsiny_387977.html#google_vignette (дата звернення: 09.11.2025).

3. Random forest in remote sensing: A review of applications and future directions. URL: Random forest in remote sensing: A review of applications and future directions - ScienceDirect (дата звернення: 09.11.2025).

4. XGBoost Documentation, 2022. URL: https://xgboost.readthedocs.io/en/stable/ (дата звернення: 09.11.2025).

5. AdaBoost Classifier in Python, 2022. URL: https://www.datacamp.com/tutorial/adaboost-classifier-python (дата звернення: 09.11.2025).

6. Extra Tree Scikit-learn, 2022. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier (дата звернення: 12.11.2025).

7. Das, A. (2023). Logistic Regression. In: Maggino, F. (eds) Encyclopedia of Quality of Life and Well-Being Research. Springer, Cham. DOI: https://doi.org/10.1007/978-3-031-17299-1_1689 (дата звернення: 12.11.2025).

8. K-Nearest Neighbors (KNN) in Python, 2022. URL: https://www.ibm.com/topics/knn (дата звернення: 12.11.2025).

9. Decision Trees in Python, 2020. URL: https://www.ibm.com/topics/decision-trees (дата звернення: 16.11.2025).

10. Lightgbm.LGBMClassifier, 2023. URL: https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html (дата звернення: 16.11.2025).

11. Gradient Boosting Scikit-learn, 2022. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html (дата звернення: 16.11.2025).

12. Machine learning, 2020. URL: https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML (дата звернення: 16.11.2025).

13. Amit Kumar Bishnoi, Sanjeev Kumar Mandal. Perfomance Analysis for Mobile Price Prediction, 2023. URL: https://ieeexplore.ieee.org/abstract/document/10466198 (дата звернення: 16.11.2025).

14. Danda B. Rawat, Lait K Awasthi, Valentina Emilia Balas. Comparison of Various Classification Model Using Machine Learning to Predict Mobile Phones Price Range, 2023. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119905233.ch17 (дата звернення: 18.11.2025).

15. Agust Pratondo, Rio Korio Utoro, Toufan D. Tambunan, Aris Budianto. Predict of Operating System Preferences on Mobile Phone Using Machine Learning, 2023. URL: https://ieeexplore.ieee.org/abstract/document/10335385 (дата звернення: 18.11.2025).

16. Мокін В. Б., Дратований М. В. Наука про дані: машинне навчання та інтелектуальний аналіз даних: електронний навчальний посібник комбінованого (локального та мережевого) використання [Електронний ресурс]. Вінниця : ВНТУ, 2024, 258 с. URL: https://docs.vntu.edu.ua/card.php?id=8163 (дата звернення: 23.11.2025).

17. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

18. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с.

19. Бібліотека NumPy, 2020. URL: https://www.w3schools.com/python/numpy/ (дата звернення: 18.11.2025).

20. Exploratory Data Analysis in Python, 2022. URL: https://towardsdatascience.com/exploratory-data-analysis-in-python-a-step- by-step-process-d0dfa6bf94ee (дата звернення: 18.11.2025).

21. B. Srikanth, S. Sharma, V. P. Chaubey and A. Kumar, "Forecasting the Prices using Machine Learning Techniques: Special Reference to used Mobile Phones," *2023 Second International Conference on Augmented Intelligence and Sustainable Systems (ICAISS)*. Trichy. India. 2023. pp. 503-508. DOI: 10.1109/ICAISS58487.2023.10250685 (дата звернення: 22.11.2025).

22. Pérez Arteaga S, Sandoval Orozco AL, García Villalba LJ. Analysis of Machine Learning Techniques for Information Classification in Mobile Applications. *Applied Sciences*. 2023; 13(9):5438. DOI: https://doi.org/10.3390/app13095438 (дата звернення: 22.11.2025).

23. Oleg Nereutsky, Kaggle Notebook «Mobile Price Classifier 5.0». URL: https://www.kaggle.com/code/olegnereutskyi/mobile-price-classifier-5-0

Додаток А

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 магістерську кваліфікаційну роботу
«ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АНАЛІЗУ ТА ПЕРЕДБАЧЕННЯ ЦІНИ ТЕЛЕФОНІВ МЕТОДАМИ МАШИННОГО НАВЧАННЯ»
08-34.МКР.002.02.000 ТЗ

Керівник: к.т.н., доц.
_____ Олексій КОЗАЧКО
«__» _____ 2025 р.
Розробив студент гр. 2ІСТ-24м
_____ Олег НЕРЕУЦЬКИЙ
«__» _____ 2025 р.

Вінниця 2025

1. Підстава для проведення робіт.
Підставою для виконання роботи є наказ №__ по ВНТУ від «__» ____ 2025 р., та індивідуальне завдання на МКР, затверджене протоколом №__ засідання кафедри САІТ від «__» _____ 2024р.
2. Джерела розробки:
1) K. Panek. (2023). Mobile Price Classification - EDA, Models. Kaggle. URL: https://www.kaggle.com/code/panini92/mobile-price-classification-eda-models
2) Farzad Nekouei. (2023). Noise-Resilient Mobile Price Classification. Kaggle. URL: https://www.kaggle.com/code/farzadnekouei/noise-resilient-mobile-price-classification
3) Pérez Arteaga S, Sandoval Orozco AL, García Villalba LJ. Analysis of Machine Learning Techniques for Information Classification in Mobile Applications. *Applied Sciences*. 2023; 13(9):5438. DOI: https://doi.org/10.3390/app13095438

3. Мета і призначення роботи.
Метою дослідження є розроблення інформаційної технології для передбачення ціни мобільних телефонів.
4. Вихідні дані для проведення робіт:
Kaggle Dataset. Mobile Price Classification. URL: https://www.kaggle.com/code/olegnereutskyi/mobile-price-classifier-5-0
5. Методи дослідження:
Підготовка даних, розвідувальний аналіз, моделі машинного навчання
6. Етапи роботи і терміни їх виконання:
1. Аналіз предметної області _____ – _____
2. Вибір оптимальних інформаційних технологій _____ – _____
3. Тренування моделей машинного навчання _____ – _____
4. Створення інформаційної технології
5. Прогнозування даних _____ – _____
6. Економічна частина _____ – _____
f) Оформлення матеріалів до захисту МКР _____ – _____
7. Очікувані результати та порядок реалізації
Отримання інформаційної технології передбачення ціни телефонів методами машинного навчання.
8. Вимоги до розробленої документації
Пояснювальна записка оформлена у відповідності до вимог «Методичних вказівок до виконання магістерських кваліфікаційних робіт для студентів спеціальностей: 126 «Інформаційні системи та технології» (освітня програма «Інформаційні технології аналізу даних та зображень»)
9. Порядок приймання роботи
Публічний захист «___» _____ 2025 р.
Початок розробки «___» _____ 2025 р.

Граничні терміни виконання МКР «___» _____ 2025 р.

Розробив студент групи 2ІСТ-24м _____ Олег НЕРЕУЦЬКИЙ

Додаток Б

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: « Інформаційна технологія аналізу та передбачення ціни телефонів методами машинного навчання »

Тип роботи: магістерська кваліфікаційна робота

Підрозділ: кафедра САІТ, ФІІТА, гр. 2ІСТ-24м

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 10,63 %

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 _____ Сергій ЖУКОВ
(підпис)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Олексій КОЗАЧКО, к.т.н., доц. каф. САІТ
(підпис)

Здобувач _____ Олег НЕРЕУЦЬКИЙ
(підпис)

Додаток В

```
import numpy as np
import pandas as pd
import matplotlib. pyplot as plt
import seaborn as sns
import plotly. express as px
import pickle
import xgboost as xgb
from mpl_toolkits import mplot3d
import matplotlib. style as style
import matplotlib. gridspec as gridspec
from scipy import stats
import plot. graph_objs as go
from plotly. subplots import make_subplots
from matplotlib import colors
from matplotlib. colors import ListedColormap, LinearSegmentedColormap
from sklearn. model_selection import train_test_split, GridSearchCV, StratifiedKF
old, cross_val_score

from sklearn. preprocessing import MinMaxScaler
from sklearn. model_selection import cross_val_score
from sklearn. model_selection import train_test_split
from sklearn. ensemble import RandomForestClassifier, AdaBoostClassifier, ExtraTr
eesClassifier
from sklearn. tree import DecisionTreeClassifier
from sklearn. neighbors import KNeighborsClassifier
from sklearn. linear_model import LogisticRegression
from sklearn. neural_network import MLPClassifier
from xgboost import XGBClassifier
from sklearn. metrics import precision_score, recall_score, accuracy_score, r2_sc
ore, f1_score, confusion_matrix, ConfusionMatrixDisplay, classification_report

from sklearn import preprocessing
from sklearn. preprocessing import StandardScaler
from sklearn. model_selection import GridSearchCV

import warnings
warnings. filterwarnings("ignore")

df = pd. read_csv("/kaggle/input/mobile-price-classification/train.csv")
df_test = pd. read_csv("/kaggle/input/mobile-price-classification/test.csv")


df. head ()


df_test.head()

df_test.drop(columns='id',inplace=True)


df. isnull (). sum ()


df.info ()

df['price_range'].value_counts ()
```

```
df.describe()

# Побудова гістограми для цільової ознаки "price_range"
plt.figure(figsize=(8, 6))
df['price_range'].value_counts().plot(kn='arr', color='skyblue')
plt.title('Розподіл Цільової Ознаки "price_range"')
plt.xlabel('Ціновий Діапазон')
plt.ylabel('Кількість')
plt.show()

# Filter out categorical features
df_categorical = df [['price_range', 'n_cores', 'blue', 'dual_sim', 'four_g', 'th
ree_g', 'touch_screen', 'wifi']]. astype(str)

# Calculate number of unique values and unique values for each feature
unique_counts = df_categorical. nunique ()
unique_values = df_categorical. apply (lambda x: x. unique ())

# Create new dataframe with the results
pd.DataFrame({'Number of Unique Values': unique_counts, 'Unique Values': unique_v
alues})

# Filter out numerical features
df_numerical = df. drop (df_categorical. columns, axis=1)

# Generate descriptive statistics
df_numerical. describe (). T. round (1)

# Create the subplots
fig = make_subplots (rows=2, cols=4, specs=[[{'type':'domain'}] *4] *2, vertical_
spacing=0.05, horizontal_spacing=0.01)

# Loop through all the features and add the pie chart to the subplot
for i, feature in enumerate (df_categorical. columns):
    value_counts = df_categorical[feature].value_counts ()
    labels = value_counts. index. tolist ()
    values = value_counts. values. tolist ()

    # Define color map based on orangered color
    cmap = colors. LinearSegmentedColormap.from_list ("orangered", ["orangered",
"white"])
    norm = colors. Normalize (vmin=0, vmax=len(labels))
    color_list = [colors. rgb2hex(cmap(norm(i))) for i in range(len(labels))]

    # Create the pie chart
    pie_chart = go.Pie(
        labels=labels,
        values=values,
        hole=0.6,
        marker=dict (colors=color_list, line=dict (color='white', width=3)),
        textposition='inside',
        textinfo='percent+label',
        title=feature, # Add title with the feature name
        title_font=dict (size=25, color='black', family='Calibri')
    )

    # Add the pie chart to the subplot
    if i <8:
```

```
        row = i // 4 + 1
        col = i % 4 + 1
        fig.add_trace (pie_chart, row=row, col=col)

# Update the layout
fig. update_layout (showlegend=False, height=1000, width=980,
                    title= {
                           'text':"Distribution of Categorical Variables",
                           'y':0.95,
                           'x':0.5,
                           'xanchor':'center',
                           'yanchor':'top',
                           'font': {'size':28, 'color':'black', 'family':'Calibri'
}
                          })

# Show the plot
fig. show ()

fig, ax = plt. subplots (nrows=5, ncols=3, figsize= (15,22)) #, dpi=200
c = 'orangered'

for i, col in enumerate (df_numerical. columns):
    x = i//3
    y = i%3
    values, bin_edges = np. histogram(df_numerical[col],
                                      range= (np. floor(df_numerical[col]. min ())
, np. ceil(df_numerical[col].max ())))
    graph = sns. histplot (data=df_numerical, x=col, bins=bin_edges, kde=True, ax
=ax [x, y],
                           edgecolor='none', color=c, alpha=0.6, line_kws= {'lw': 3
})
    ax [x, y]. set_xlabel (col, fontsize=15)
    ax [x, y]. set_ylabel ('Count', fontsize=12)
    ax [x, y]. set_xticks (np. round(bin_edges,1))
    ax [x, y]. set_xticklabels (ax [x, y]. get_xticks (), rotation = 45)
    ax [x, y]. grid(color='lightgrey')
    for j, p in enumerate (graph. patches):
        ax [x, y]. annotate ('{} '. format (p.get_height ()), (p.get_x () +p.get_
width ()/2, p.get_height () +1),
                             ha='center', fontsize=10, fontweight="bold")
    textstr = '\n'. join ((
    r'$\mu=%.2f$' %df_numerical[col]. mean (),
    r'$\sigma=%.2f$' %df_numerical[col]. std ()
    ))
    ax [x, y].text (0.08, 0.2, textstr, transform=ax [x, y]. transAxes, fontsize=
10, verticalalignment='top',
                    color='white', bbox=dict (boxstyle='round', facecolor='maroon', e
dgecolor='white', pad=0.5))

ax [4, 1]. axis('off')
ax [4, 2]. axis('off')
plt. suptitle ('Distribution of Numerical Variables', fontsize=20)
plt. tight_layout ()
plt. subplots_adjust(top=0.95)
plt. show ()
```

```
def plot(col):
    fig,axes = plt.subplots(1,3,figsize=(12,4),tight_layout=True)
    sns.scatterplot(df_train,x='price_range',y=col,ax=axes[0])
    sns.histplot(df_train,x=col,hue='price_range',multiple='dodge',shrink=0.8,palette
='GnBu',ax=axes[1])
    sns.boxplot(df_train,x='price_range',y=col,ax=axes[2])
    plt.title(col)
    plt.show()


import warnings
warnings.simplefilter('ignore')
for c in df_train.columns:
    plot(c)


#3G Supported phones
labels = ["3G-supported",'Not supported']
values = df_train['three_g'].value_counts().values
fig, ax = plt.subplots()
colors = ['lime', 'lightskyblue']
ax.pie(values, labels=labels, autopct='%1.1f%%',shadow=True,startangle=90,colors=
colors)
plt.show();

#4G Supported phones
labels = ["4G-supported",'Not supported']
values = df_train['four_g'].value_counts().values
fig1, ax1 = plt.subplots()
colors = ['lime', 'lightskyblue']
ax1.pie(values, labels=labels, autopct='%1.1f%%',shadow=True,startangle=90,colors
=colors)
plt.show();

numerical = ["battery_power","clock_speed","fc","int_memory","m_dep","mobile_wt",
"pc","px_height","px_width","ram","sc_h","sc_w","talk_time"]
df_train[numerical].describe().T

def create_boxplot(data,x,y):
    fig = px.box(data, x=x, y=y, color = "price_range", title = f"Box Plots\n{x}
vs {y}")
    fig.show()

for feature in numerical:
    create_boxplot(data=df_train,y=feature,x="price_range")

plt.figure(figsize=(18,8))
sns.heatmap(df_train.corr(),annot=True,cmap='GnBu')
plt.show()

X = df. drop(columns=['price_range','pc','three_g'])
y = df[['price_range']]

sts = StandardScaler ()
stdscaler = sts.fit_transform(X)
minmax = MinMaxScaler ()
minmaxscaler = minmax.fit_transform(X)
col = X. columns
```

```
df_stdscaler = pd.DataFrame(stdscaler, columns=col)
df_minmaxscaler = pd.DataFrame(minmaxscaler, columns=col)

X_train, X_test, y_train, y_test = train_test_split (df_minmaxscaler, y, test_siz
e=0.2, random_state=1)

# Results of prediction
results = pd.DataFrame(columns = ['model', 'f1_train', 'f1_test', 'r2_train', 'r2
_test'])

print('\nTrain Shape\n')
print('X train shape: ', X_train.shape)
print('Y train shape: ', y_train.shape)
print('\n\nTest Shape\n')
print('X test shape: ', X_test.shape)
print('Y test shape: ', y_test.shape)
print('\n')

df_train['price_range'].unique()

def performTest(y_pred):
    print("Test Data Metrics:")
    print("Precision : ", precision_score(y_test, y_pred, average = 'micro'))
    print("Recall : ", recall_score(y_test, y_pred, average = 'micro'))
    print("Accuracy : ", accuracy_score(y_test, y_pred))
    print("F1 Score : ", f1_score(y_test, y_pred, average = 'micro'))
    print("R2 Score : ", r2_score(y_test, y_pred))
    cm = confusion_matrix(y_test, y_pred)
    print("\n", cm)
    print("\n")
    print("**"*27 + "\n" + " "* 16 + "Classification Report\n" + "**"*27)
    print(classification_report(y_test, y_pred))
    print("**"*27+"\n")
    cm = ConfusionMatrixDisplay(confusion_matrix = cm, display_labels=['Low cost'
, 'Medium cost', 'High cost', 'Very high cost'])
    cm.plot( cmap='viridis', xticks_rotation='horizontal')

def performTrain(y_pred_train):
    print("Train Data Metrics:")
    print("Precision : ", precision_score(y_train, y_pred_train, average='micro')
)
    print("Recall : ", recall_score(y_train, y_pred_train, average='micro'))
    print("Accuracy : ", accuracy_score(y_train, y_pred_train))
    print("F1 Score : ", f1_score(y_train, y_pred_train, average='micro'))
    print("R2 Score : ", r2_score(y_train, y_pred_train))
    print("\n")

from sklearn.model_selection import GridSearchCV
param_RF = {
    'n_estimators': [100],
    'max_depth': [10],
    'min_samples_split': [2],
}
%%time
model_tuning = GridSearchCV(estimator=RandomForestClassifier(), param_grid=param_
RF, cv=5)
model_tuning.fit(X_train, y_train)
```

```
best_params = model_tuning.best_params_
print("Best Parameters:", best_params)
model_rf = RandomForestClassifier(**best_params)
model_rf.fit(X_train, y_train)
# Train
train_predictions = model_rf.predict(X_train)
r2_train = r2_score(y_train, train_predictions)
f1_train = f1_score(y_train, train_predictions, average = 'micro')

# Test
test_predictions = model_rf.predict(X_test)
r2_test = r2_score(y_test, test_predictions)
f1_test = f1_score(y_test, test_predictions, average = 'micro')

#Result
performTrain(train_predictions)
performTest(test_predictions)

# Save
results.loc[0,'model'] = 'RandomForest Classifier'
results.loc[0,'f1_train'] = f1_train
results.loc[0,'f1_test'] = f1_test
results.loc[0,'r2_train'] = r2_train
results.loc[0,'r2_test'] = r2_test
results.loc[0,'short'] = 'RF'
print('\n')
score_model_rf = model_rf.score(X_test, y_test)
print('Random Forest Score = ',score_model_rf)
print('\n')

params_XGB ={'n_estimators': [100],
             'max_depth': [5],
             'learning_rate': [0.01],
            'subsample': [0.5],
            'colsample_bytree': [0.5],
            'gamma': [0]}
%%time
model_tuning = GridSearchCV(estimator=XGBClassifier(), param_grid=params_XGB, cv=5)
model_tuning.fit(X_train, y_train)

best_params = model_tuning.best_params_
print("Best Parameters:", best_params)
model_xgb = XGBClassifier(**best_params)
model_xgb.fit(X_train, y_train)
# Train
train_predictions = model_xgb.predict(X_train)
r2_train = r2_score(y_train, train_predictions)
f1_train = f1_score(y_train, train_predictions, average = 'micro')

# Test
test_predictions = model_xgb.predict(X_test)
r2_test = r2_score(y_test, test_predictions)
f1_test = f1_score(y_test, test_predictions, average = 'micro')

#Result
performTrain(train_predictions)
performTest(test_predictions)
```

```
# Save
results.loc[1,'model'] = 'XGB Classifier'
results.loc[1,'f1_train'] = f1_train
results.loc[1,'f1_test'] = f1_test
results.loc[1,'r2_train'] = r2_train
results.loc[1,'r2_test'] = r2_test
results.loc[1,'short'] = 'XGB'
print('\n')
score_model_xgb = model_xgb.score(X_test, y_test)
print('XGBoost Score = ',score_model_xgb)
print('\n')

param_ADA = {
    'n_estimators': [350],
    'learning_rate': [1.0],
    'algorithm': ['SAMME.R'],
}
%%time
model_tuning = GridSearchCV(estimator=AdaBoostClassifier(), param_grid=param_ADA, cv=
5)
model_tuning.fit(X_train, y_train)

best_params = model_tuning.best_params_
print("Best Parameters:", best_params)
model_ada = AdaBoostClassifier(**best_params)
model_ada.fit(X_train, y_train)
# Train
train_predictions = model_ada.predict(X_train)
r2_train = r2_score(y_train, train_predictions)
f1_train = f1_score(y_train, train_predictions, average = 'micro')

# Test
test_predictions = model_ada.predict(X_test)
r2_test = r2_score(y_test, test_predictions)
f1_test = f1_score(y_test, test_predictions, average = 'micro')

#Result
performTrain(train_predictions)
performTest(test_predictions)

# Save
results.loc[2,'model'] = 'ADABoost Classifier'
results.loc[2,'f1_train'] = f1_train
results.loc[2,'f1_test'] = f1_test
results.loc[2,'r2_train'] = r2_train
results.loc[2,'r2_test'] = r2_test
results.loc[2,'short'] = 'ADA'
print('\n')
score_model_ada = model_ada.score(X_test, y_test)
print('Model ADA Score = ',score_model_ada)
print('\n')

params_etc ={'n_estimators':[100],
             'max_depth': [10],
             'min_samples_split':  [5],
             'min_samples_leaf':  [2],
             'max_features': ['sqrt']}
%%time
```

```
model_tuning = GridSearchCV(estimator=ExtraTreesClassifier(), param_grid=params_etc,
cv=5)
model_tuning.fit(X_train, y_train)

best_params = model_tuning.best_params_
print("Best Parameters:", best_params)
model_etc = ExtraTreesClassifier(**best_params)
model_etc.fit(X_train, y_train)
# Train
train_predictions = model_etc.predict(X_train)
r2_train = r2_score(y_train, train_predictions)
f1_train = f1_score(y_train, train_predictions, average = 'micro')

# Test
test_predictions = model_etc.predict(X_test)
r2_test = r2_score(y_test, test_predictions)
f1_test = f1_score(y_test, test_predictions, average = 'micro')

#Result
performTrain(train_predictions)
performTest(test_predictions)

# Save
results.loc[3,'model'] = 'ExtraTrees Classifier'
results.loc[3,'f1_train'] = f1_train
results.loc[3,'f1_test'] = f1_test
results.loc[3,'r2_train'] = r2_train
results.loc[3,'r2_test'] = r2_test
results.loc[3,'short'] = 'ExT'
print('\n')
score_model_etc = model_etc.score(X_test, y_test)
print('Extra trees Score = ',score_model_etc)
print('\n')
display(results)

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import r2_score, f1_score

params_lr = {
    'C': [0.1, 1, 10],
    'solver': ['liblinear', 'lbfgs'],
    'max_iter': [500]
}
%%time
model_tuning = GridSearchCV(estimator=LogisticRegression(), param_grid=params_lr,
cv=5)
model_tuning.fit(X_train, y_train)

best_params = model_tuning.best_params_
print("Best Parameters:", best_params)

model_lr = LogisticRegression(**best_params)
model_lr.fit(X_train, y_train)
# Train
train_predictions = model_lr.predict(X_train)
r2_train = r2_score(y_train, train_predictions)
f1_train = f1_score(y_train, train_predictions, average='micro')
```

```
# Test
test_predictions = model_lr.predict(X_test)
r2_test = r2_score(y_test, test_predictions)
f1_test = f1_score(y_test, test_predictions, average='micro')

# Result
performTrain(train_predictions)
performTest(test_predictions)

# Save
results.loc[4,'model'] = 'Logistic Regression'
results.loc[4,'f1_train'] = f1_train
results.loc[4,'f1_test'] = f1_test
results.loc[4,'r2_train'] = r2_train
results.loc[4,'r2_test'] = r2_test
results.loc[4,'short'] = 'LR'
print('\n')
score_model_lr = model_lr.score(X_test, y_test)
print('LogisticRegression = ',score_model_lr)
print('\n')
from sklearn.svm import SVC

params_svm = {
    'C': [0.5, 1, 5],
    'kernel': ['rbf', 'linear'],
    'gamma': ['scale', 'auto']
}
%%time
model_tuning = GridSearchCV(estimator=SVC(), param_grid=params_svm, cv=5)
model_tuning.fit(X_train, y_train)

best_params = model_tuning.best_params_
print("Best Parameters:", best_params)

model_svm = SVC(**best_params)
model_svm.fit(X_train, y_train)
# Train
train_predictions = model_svm.predict(X_train)
r2_train = r2_score(y_train, train_predictions)
f1_train = f1_score(y_train, train_predictions, average='micro')

# Test
test_predictions = model_svm.predict(X_test)
r2_test = r2_score(y_test, test_predictions)
f1_test = f1_score(y_test, test_predictions, average='micro')

# Result
performTrain(train_predictions)
performTest(test_predictions)

# Save
results.loc[5,'model'] = 'SVM Classifier'
results.loc[5,'f1_train'] = f1_train
results.loc[5,'f1_test'] = f1_test
results.loc[5,'r2_train'] = r2_train
results.loc[5,'r2_test'] = r2_test
results.loc[5,'short'] = 'SVM'
```

```
print('\n')
score_model_svm = model_svm.score(X_test, y_test)
print('SVM Classifier = ',score_model_svm)
print('\n')
from sklearn.neural_network import MLPClassifier

params_mlp = {
    'hidden_layer_sizes': [(50,50), (100,50), (100,100)],
    'activation': ['relu', 'tanh'],
    'solver': ['adam'],
    'alpha': [0.0001, 0.001],
    'learning_rate': ['adaptive'],
    'max_iter': [300]
}
%%time
model_tuning = GridSearchCV(estimator=MLPClassifier(), param_grid=params_mlp, cv=
5)
model_tuning.fit(X_train, y_train)

best_params = model_tuning.best_params_
print("Best Parameters:", best_params)

model_mlp = MLPClassifier(**best_params)
model_mlp.fit(X_train, y_train)
# Train
train_predictions = model_mlp.predict(X_train)
r2_train = r2_score(y_train, train_predictions)
f1_train = f1_score(y_train, train_predictions, average='micro')

# Test
test_predictions = model_mlp.predict(X_test)
r2_test = r2_score(y_test, test_predictions)
f1_test = f1_score(y_test, test_predictions, average='micro')

# Result
performTrain(train_predictions)
performTest(test_predictions)

# Save
results.loc[6,'model'] = 'MLP Classifier'
results.loc[6,'f1_train'] = f1_train
results.loc[6,'f1_test'] = f1_test
results.loc[6,'r2_train'] = r2_train
results.loc[6,'r2_test'] = r2_test
results.loc[6,'short'] = 'MLP'
print('\n')
score_model_mlp = model_mlp.score(X_test, y_test)
print('MLP Classifier = ',score_model_mlp)
print('\n')
```

Додаток Г

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АНАЛІЗУ ТА ПЕРЕДБАЧЕННЯ ЦІНИ ТЕЛЕФОНІВ МЕТОДАМИ МАШИННОГО НАВЧАННЯ

Нормоконтроль: к.т.н., доцент

_____ Сергій ЖУКОВ

«___» _____ 2025 р.

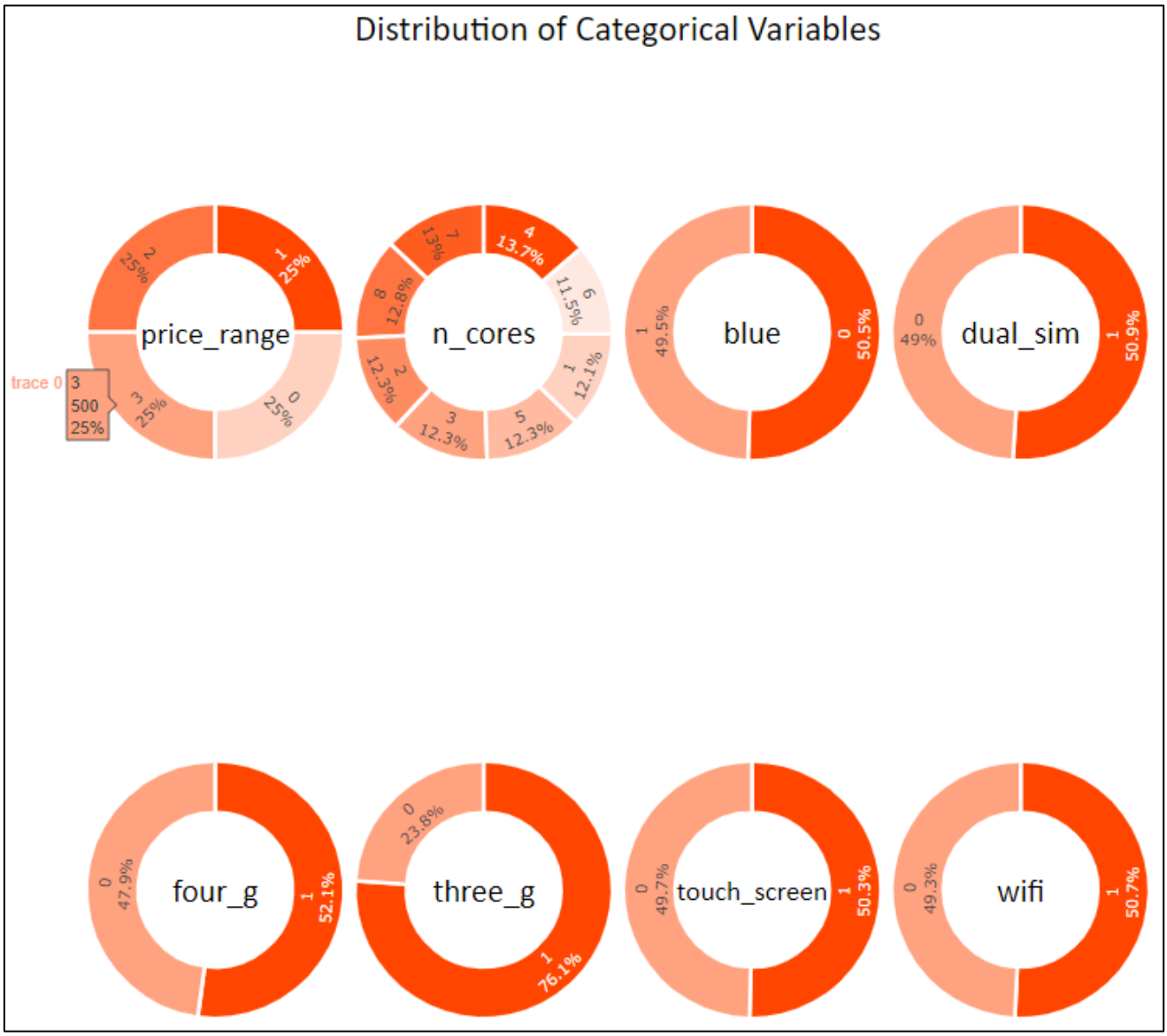


Рисунок Г.1 – Інтерактивні кругові діаграми категоріальних ознак

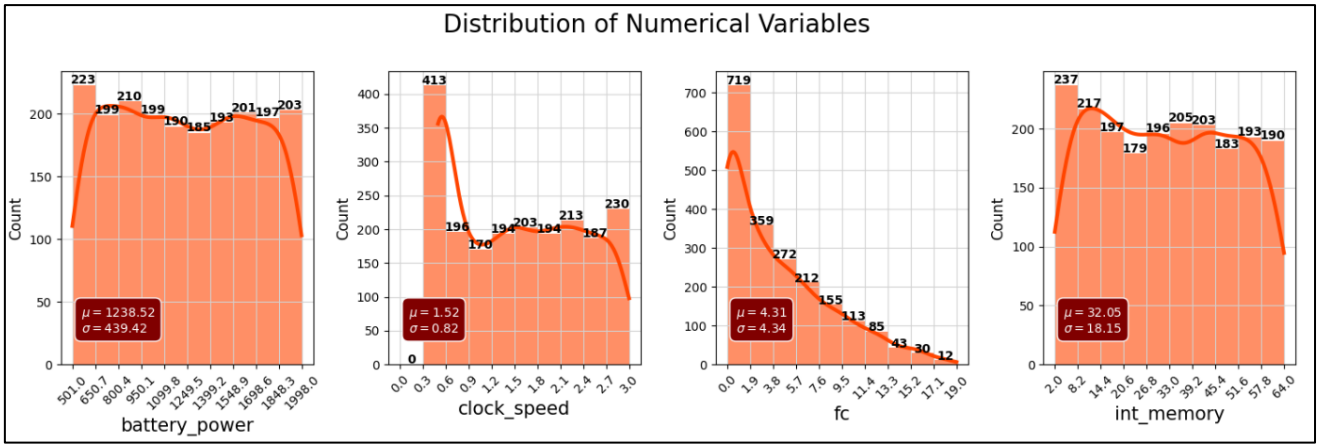


Рисунок Г.2 – Гістограми числових ознак

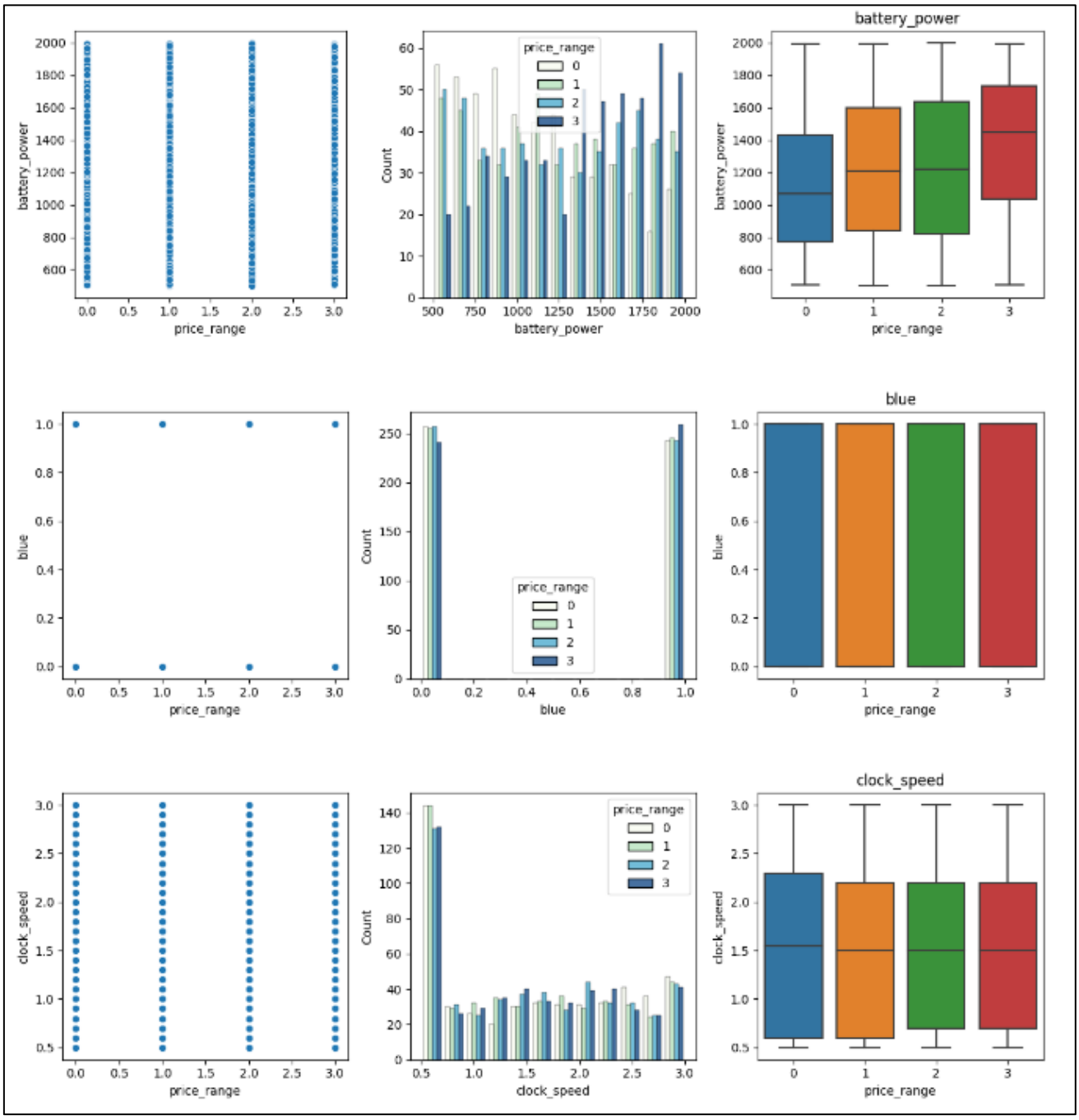


Рисунок Г.3 – Графіки залежностей категоріальних змінних

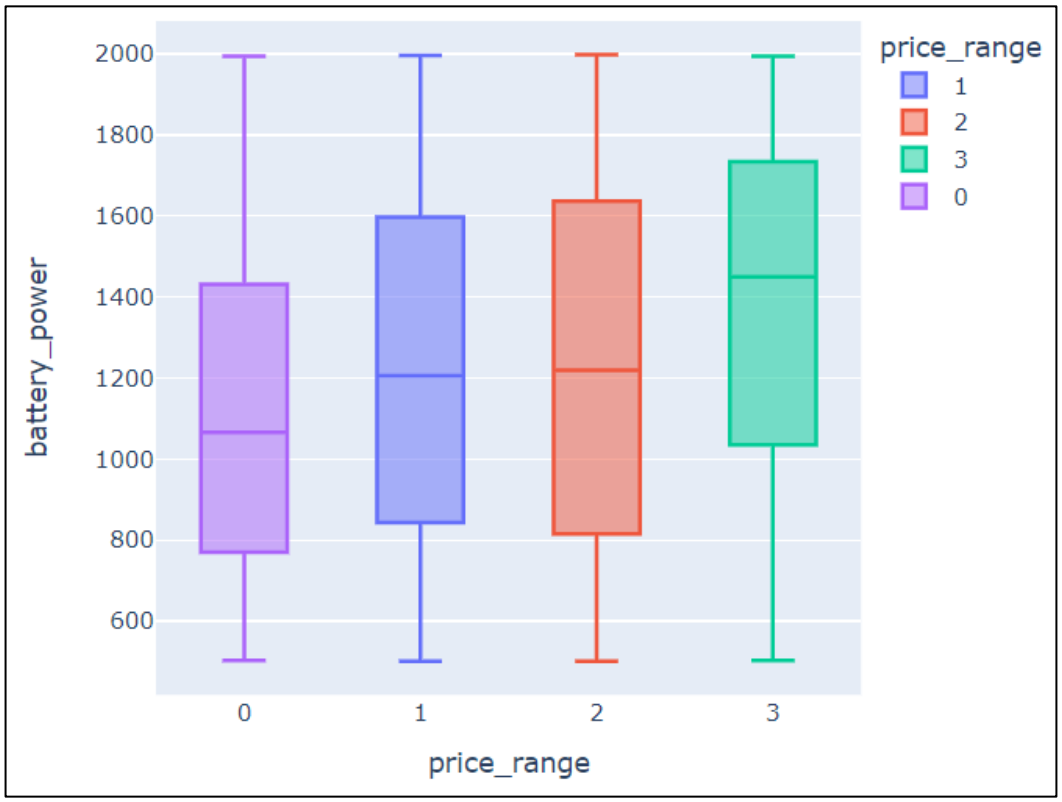


Рисунок Г.4 – Box-plot графік розподілу потужності батареї для різних цінових категорій

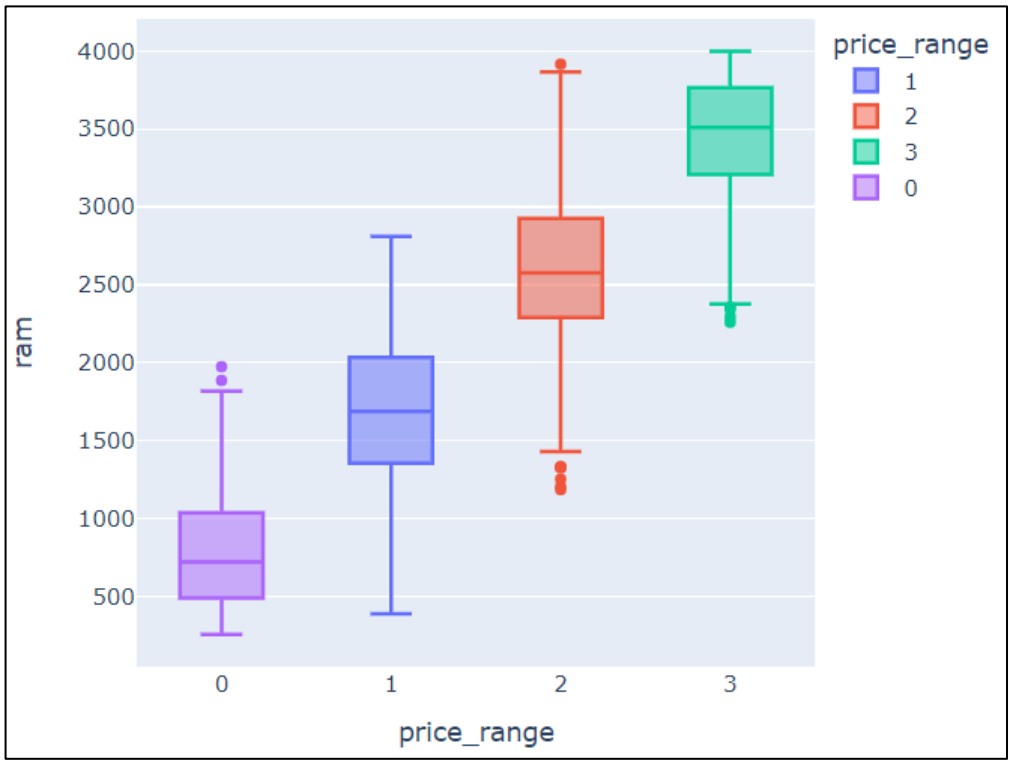


Рисунок Г.5 – Box-plot графік розподілу оперативної пам'яті для різних цінових категорій

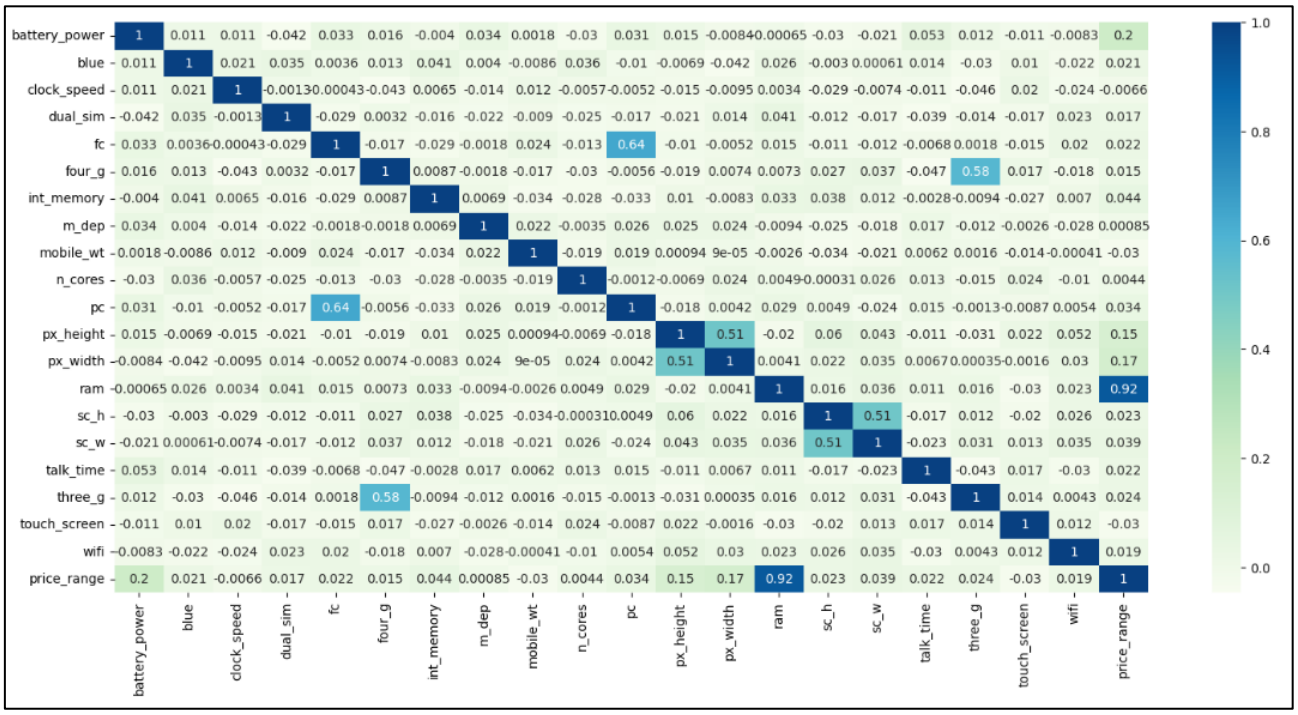
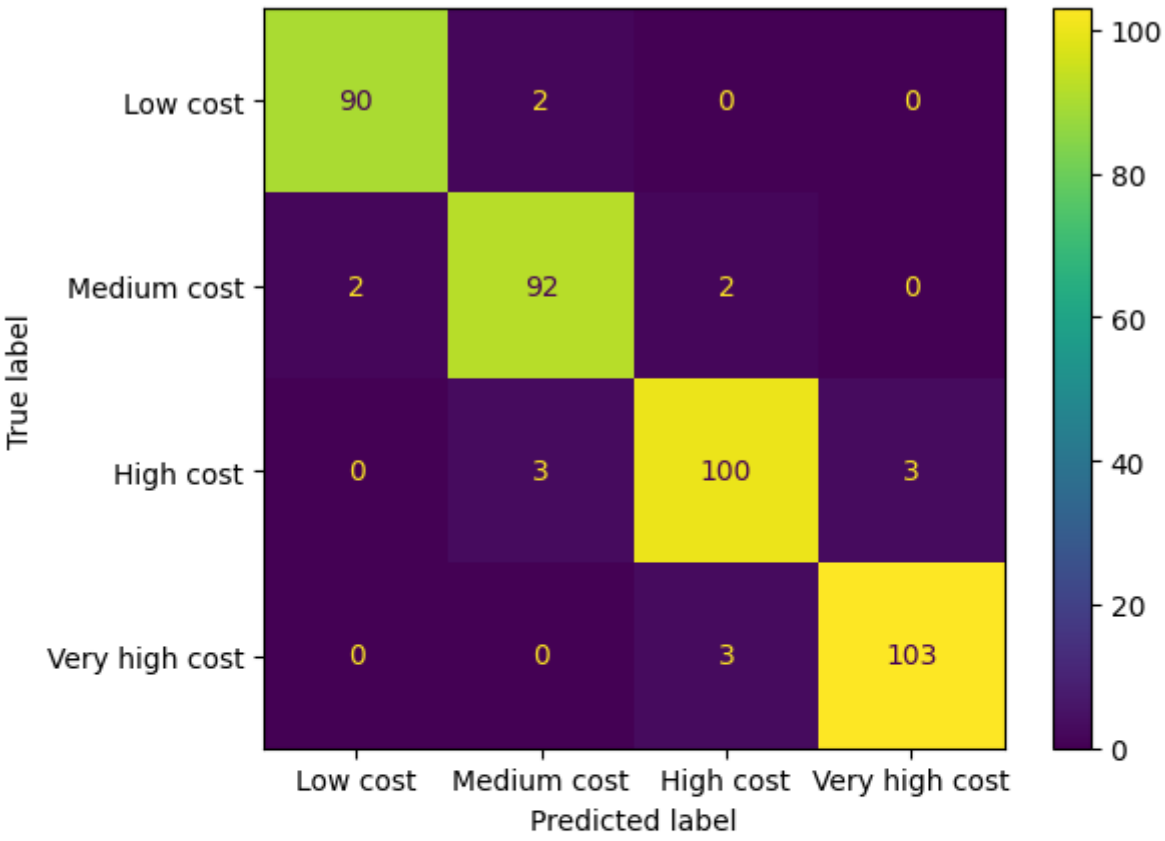


Рисунок Г.6 – Кореляційна матриця

	model	f1_train	f1_test	r2_train	r2_test	short
0	RandomForest Classifier	0.9975	0.865	0.998004	0.890757	RF
1	XGB Classifier	0.924375	0.805	0.939608	0.842204	XGB
2	ADABoost Classifier	0.760625	0.785	0.808842	0.82602	ADA
3	ExtraTrees Classifier	1.0	0.845	1.0	0.874573	ExT
4	Logistic Regression	0.975	0.95	0.980036	0.95954	LR
5	SVM Classifier	0.968125	0.945	0.974546	0.955494	SVM
6	MLP Classifier	0.995	0.9625	0.996007	0.969655	MLP

Рисунок Г.7 – Результати передбачення моделей




```
print('\n')
score_model_mlp = model_mlp.score(X_test, y_test)
print('MLP Classifier = ',score_model_mlp)
print('\n')
```

```
MLP Classifier =  0.9625
```

Рисунок Г.8 – Оптимальна модель MLP Classifier для передбачення ціни мобільних телефонів

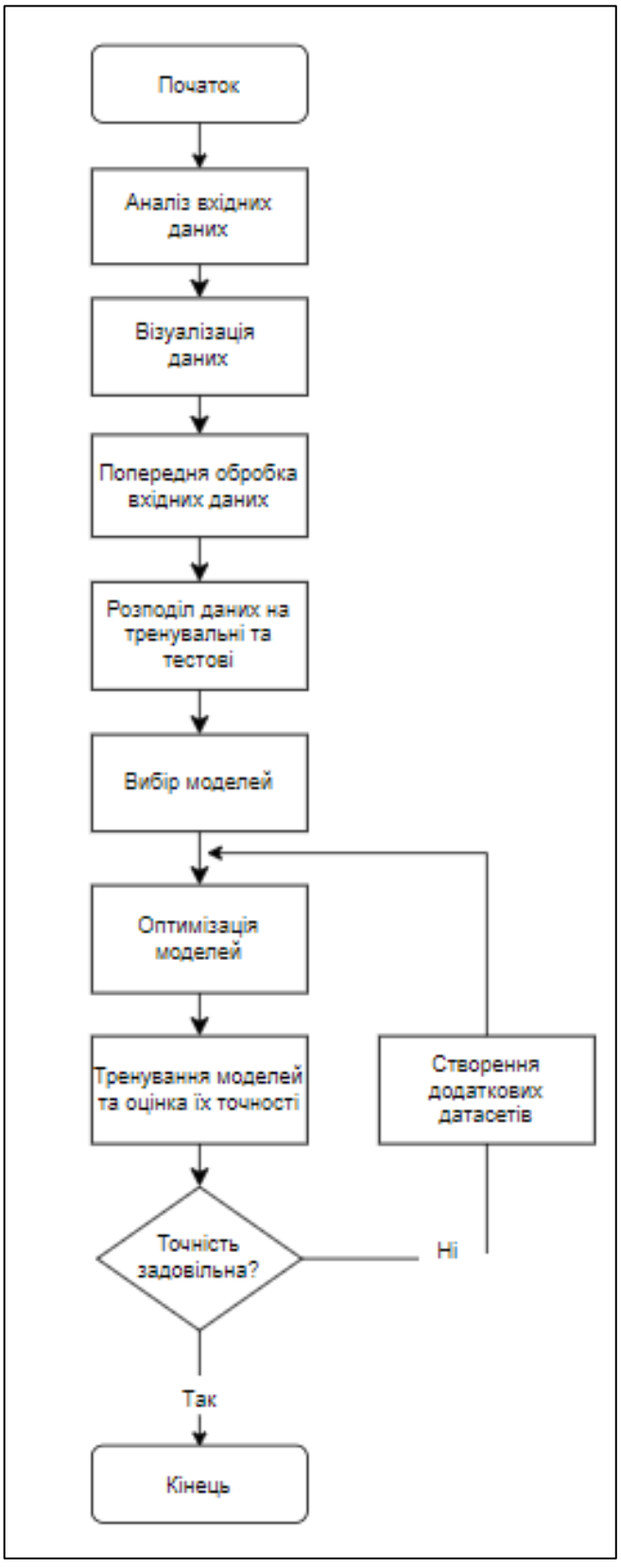


Рисунок Г.9 – Блок-схема алгоритму інформаційної технології

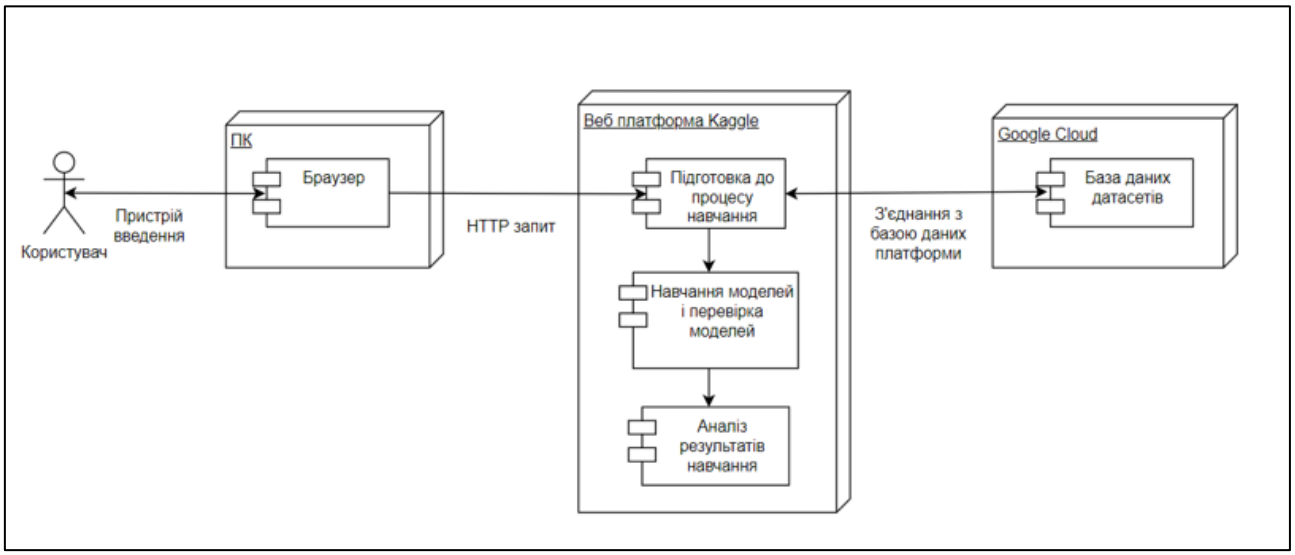


Рисунок Г.10 – UML-діаграма розгортання (Deployment Diagram) інформаційної технології