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

Бакалаврська дипломна робота на тему:

«ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПЕРЕДБАЧЕННЯ ЦІНИ ТЕЛЕФОНІВ МЕТОДАМИ МАШИННОГО НАВЧАННЯ»

Виконав: студент 4 курсу, групи 2ІСТ-20б спеціальності 126 – «Інформаційні системи та технології»
_____ Олег НЕРЕУЦЬКИЙ
Керівник: к.т.н., доц. каф. САІТ
_____ Олексій КОЗАЧКО
«31» 05 2024 р.
Рецензент: к.т.н., доц. каф. КН
_____ Володимир ОЗЕРАНСЬКИЙ
«10» 06 2024 р.

Допущено до захисту
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
«03» 06 2024 р.

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій
Рівень вищої освіти – перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність 126 – «Інформаційні системи та технології»
Освітньо-професійна програма – Прикладні інформаційні технології

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
"05" 03 2024 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ
Нереуцькому Олегу Віталійовичу

1. Тема роботи: «Інформаційна технологія передбачення ціни телефонів методами машинного навчання»
керівник роботи: Козачко О. М., к.т.н., доц. каф. САІТ
затверджені наказом ВНТУ від «11» 03 2024 року № 80
2. Термін подання студентом роботи 31.05
3. Вихідні дані до роботи:
1) Набір даних для передбачення ціни телефонів.
4. Зміст текстової частини:
1) Аналіз предметної області;
2) Аналіз та обробка даних;
3) Розробка інформаційної технології передбачення ціни телефонів.
5. Перелік ілюстративного матеріалу:
1) Інтерактивні кругові діаграми категоріальних ознак;
2) Гістограми числових ознак;
3) Графіки залежності категоріальних змінних;
4) Box plots графіки розподілу основних ознак для різних цінових категорій;
5) Кореляційна матриця;
6) Результати передбачення моделей;
7) Оптимальна модель для передбачення ціни телефонів;
8) Блок-схема алгоритму інформаційної технології.

6. Консультанти розділів роботи:

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

7. Дата видачі завдання 11.03

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	11.03	31.03	вик
2	Вибір оптимальних інформаційних технологій	01.04	15.04	вик
3	Аналіз та обробка даних	16.04	30.04	вик
4	Розроблення інформаційної технології для передбачення ціни телефонів	01.05	15.05	вик
5	Оформлення матеріалів до захисту БДР	16.05	31.05	вик

Студент Олег НЕРЕУЦЬКИЙ

Керівник роботи Олексій КОЗАЧКО

АНОТАЦІЯ

Бакалаврська дипломна робота складається з 81 сторінки формату А4, на яких є 54 рисунки, список використаних джерел містить 22 найменуванья.

Метою роботи є розроблення інформаційної технології передбачення ціни телефонів.

Виконано аналіз предметної області, проаналізовано проблеми передбачення цін на мобільні телефони обґрунтовано актуальність дослідження.

Здійснено розвідувальний аналіз даних, під час якого було обрано ключові ознаки, які найбільше впливають на ціну телефонів.

В роботі використано декілька моделей машинного навчання Random Forest, XGBoost, AdaBoost, Extra Tree, здійснена їх реалізація для задачі передбачення ціни телефонів, розроблено інформаційну технологію.

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

ABSTRACT

The bachelor's thesis consists of 81 A4 pages with 54 figures and a list of references containing 22 titles.

The purpose of the work is to develop an information technology for predicting the price of phones.

The author analyzes the subject area, analyzes the problems of predicting prices for mobile phones, and substantiates the relevance of the study.

An exploratory analysis of the data was carried out, during which the key features that have the greatest impact on the price of phones were selected.

The paper uses several machine learning models, such as Random Forest, XGBoost, AdaBoost, Extra Tree, and implements them for the task of predicting the price of phones, and develops an information technology.

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

ЗМІСТ

ВСТУП

Актуальність теми. В сфері інформаційних технологій передбачення цін на мобільні телефони є важливою і актуальною темою, яка впливає як на споживачів, так і на виробників, продавців та інвесторів. Прогнозування цін на телефони може забезпечити низку переваг, таких як оптимізація витрат, покращення планування виробництва та маркетингових стратегій, а також підвищення конкурентоспроможності.

Для споживачів, розуміння тенденцій цін на мобільні телефони дозволяє робити більш обґрунтовані рішення щодо часу покупки, вибору моделі та отримання найкращої вартості за гроші. Це особливо важливо в умовах швидкого розвитку технологій, коли нові моделі з’являються на ринку майже щороку.

Для компаній-виробників та ритейлерів, передбачення цін дозволяє ефективніше управляти запасами, планувати випуск нових моделей і акційні кампанії, а також адаптуватися до змін у попиті та конкурентному середовищі. Аналітичні інструменти та алгоритми машинного навчання можуть допомогти в аналізі великих обсягів даних, враховуючи різні фактори, що впливають на ціноутворення, такі як сезонність, економічні показники, тренди на ринку та поведінка споживачів.

Для інвесторів, знання про можливі зміни цін на мобільні телефони може бути важливим аспектом при прийнятті рішень про вкладення коштів у технологічні компанії або стартапи. Прогнозування ринкових трендів і цін може допомогти в оцінці ризиків та потенційної вигоди від інвестицій.

Таким чином, передбачення цін на мобільні телефони є важливим інструментом для всіх учасників ринку, сприяючи прийняттю більш обґрунтованих і стратегічно вигідних рішень у сфері інформаційних технологій.

Мета і задачі дослідження. Метою дослідження є розроблення інформаційної технології для передбачення ціни мобільних телефонів.

Для досягнення поставленої мети необхідно розв’язати такі задачі:

– зробити аналіз об’єкта дослідження;

– здійснити вибір оптимальних інформаційних технологій;

– провести розвідувальний аналіз даних та виконати обробку вхідних даних;

– побудувати моделі для передбачення цін;

– розробити інформаційну технологію передбачення цін на мобільні телефони, використовуючи методи машинного навчання.

Об’єктом дослідження є процес передбачення цін на телефони методами машинного навчання.

Предметом дослідження є методи машинного навчання та інформаційна технологія передбачення цін на телефони.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз об’єкта дослідження

Об'єкт дослідження є прогнозування цін на мобільні телефони – це складна система, яка включає в себе безліч факторів, що впливають на вартість пристрою. Метою дослідження є розуміння та прогнозування коливань цін на мобільні телефони, що допомагає виробникам, продавцям та споживачам приймати обгрунтованні рішення.

Це поширене питання, яке задають керівники компаній: “Що впливає на ціни мобільних пристроїв?”

Якщо ви керуєте компанією, яка виробляє або продає мобільні телефони, або ж працюєте в роздрібній торгівлі, рекомендується мати принаймні базове уявлення про фактори, які можуть впливати на ціни. Можливо, ви чули фразу про те, що технології, дизайн та маркетинг – це все, але здебільшого це правда. Але це не повна історія.

На ціну телефону впливає безліч факторів. Деякі з них відіграють більшу роль, ніж інші. Деякі з них мають більший вплив на ціни в одному регіоні світу, ніж в іншому. Але в кінцевому рахунку, всі вони мають певний вплив.

Отже, давайте детальніше розглянемо, цо впливає на ціну телефону.

Це далеко не повний список, але він охоплює деякі основні фактори:

1. Технічна характеристика. Одним з найважливіших факторів, що впливають на ціну телефону, є його технічні характеристики. Збільшений обсяг оперативної пам'яті, потужні процесори, високоякісні камери та інші передові технології значно підвищують вартість пристрою.

2. Дизайн і матеріали. Використання високоякісних матеріалів, таких як скло і метал, а також інноваційний дизайн можуть суттєво вплинути на ціну телефону. Стильні та ергономічні пристрої часто коштують дорожче.

3. Маркетингова стратегія. Рекламні кампанії, позиціонування бренду та рівень маркетингових інвестицій також впливають на ціни. Бренди, які добре зарекомендували себе на ринку, можуть встановлювати більш високі ціни, для

підвищення обізнаності та лояльності споживачів.

4. Функціональні можливості. Додаткові функції, такі як захист від води, бездротова зарядка, підтримка 5g та інші інновації, додають цінності телефону.

5. Конкуренція на ринку. Рівень конкуренції на ринку також впливає на ціни. Якщо на ринку багато телефонів зі схожими характеристиками, виробники можуть знизити ціни, щоб залишатися конкурентоспроможними.

6. Економічність. Загальний стан економіки, обмінний курс та інфляція також відіграють важливу роль у формуванні ціни на телефон. Наприклад, збільшення вартості імпортних комплектуючих може призвести до збільшення кінцевої ціни виробу.

7. Сезонні фактори. Ціни на мобільні телефони можуть змінюватися залежно від сезону. Наприклад, під час святкового розпродажу або після виходу нової моделі ціна на попередню версію може знизитися.

8. Особливості регіону. Ціни можуть варіюватися в залежності від регіону. Рівень податків, імпортних мит, рівень доходів населення та інші регіональні особливості можуть зробити істотний вплив на вартість мобільних телефонів.

9. Інновації та дослідницька робота. Вартість досліджень, розробок та інновацій також може призвести до зростання ціни. Виробники, які активно інвестують в нові технології, часто підвищують ціни на свою продукцію.

10. Програми лояльності та знижки. Навність програм лояльності, знижок та рекламних пропозицій може вплинути на ціни. Програми, що пропонують бонуси, знижки та ексклюзивні пропозиції, можуть зробити ваш телефон більш доступгим для споживачів.

11. Екологічна відповідальність. Виробники, які впроваджують екологічні ініціативи, такі як використання перероблених матеріалів та зменшення вулгецевого сліду, можуть мати більші виробничі витрати, що може вплинути на кінцеву ціну їх продукції [1].

Це лише деякі з основних факторів, що впливають на ціну телефону. Розуміння цих факторів допомагає виробникам, продавцям та споживачам приймати обгрунтовані рішення щодо виробництва, продажу та придбання мобільних пристроїв.

1.2 Аналіз методів прогнозування

У мові Python існує безліч методів прогнозування, які можна використовувати для різних типів даних та завдань прогнозування. Оскільки основною задачею роботи є прогнозування цін на мобільні телефони, потрібно обрати методи прогнозування, які покажуть найбільш точний результат. Отже, наведемо деякі методи прогнозування Python:

1. Random Forest (Випадковий ліс) – це ансамблевий метод, який використовує велику кількість дерев рішень для підвищення точності прогнозування. Кожне дерево в ансамблі базується на випадковій вибірці даних та наборі випадкових ознак. Прогнози робляться шляхом голосування по всіх деревах (для класифікації) або за середніми значеннями (для регресії). Випадкові ліси ефективні в боротьбі з перенавчанням і забезпечують високу точність [2].

2. XGBoost (Extreme Gradient Boosting) – це алгоритм машинного навчання, який використовується для задач класифікації та регресії. Він об’єднує багато слабких дерев рішень для створення потужного та гнучкого класифікатора. Відомий своєю високою точністю та ефективністю, XGBoost став популярним рішенням для різних завдань машинного навчання [3].

3. AdaBoost – це комплексний метод, який послідовно створює слабкі моделі (наприклад, дерева рішень) і об’єднує їх для вдосконалення моделі на кожному кроці. Кожна нова модель фокусується на прикладах, які були неправильно класифіковані попередніми моделями. Це ефективно зменшує помилки та підвищує точність [4].

4. Extra Trees (Extremely Randomized Trees) – це варіант методу дерева рішень ensemble, який відрізняється від випадкових лісів тим, що розбиття вузлів виконується випадковим чином, а не оптимізується. Це забезпечує додаткову випадковість, покращує здатність моделі до узагальнення і зменшує варіабельність прогнозів [5].

5. Logistic Regression – це метод класифікації, який використовує логістичні функції для моделювання ймовірностей приналежності зразків до певного класу. Він використовується для бінарної класифікації і може бути

розширений для багатокласової класифікації. Логістична регресія проста в реалізації і добре працює з лінійно розділеними даними [6].

6. K-Nearest Neighbors (KNN) – це метод класифікації та регресії, який дозволяє робити прогнози, використовуючи відстань до найближчих сусідів. Прогноз для нової вибірки базується на K найближчих сусідів у навчальному наборі. Для класифікації вибирається найбільш поширений клас в околиці, а для регресії – середнє значення або медіана. Перевагою методу KNN є його простота та інтерпретованість [7].

7. Decision Trees (Дерева рішень) – є одним з найпоширеніших методів в машинному навчанні для задач класифікації та регресії. Основна ідея полягає в тому, щоб розділити навчальний набір на менші підгрупи шляхом встановлення правил прийняття рішень на основі вхідних ознак. Кожна розділена підгрупа стає "листком" дерева, в якому ми можемо зробити прогноз для нового вхідного запису [8].

Дерева рішень мають такі переваги, як проста інтерпретація, здатність робити висновки щодо важливості ознак та можливість роботи з категоріальними та числовими ознаками.

8. LightGBM (Light Gradient Boosting Machine) – це ефективний і швидкий алгоритм градієнтного бустінгу, розроблений компанією Microsoft. Він є одним з найпопулярніших алгоритмів машинного навчання у сфері конкурсів та реальних проектів. Основна особливість LightGBM полягає у його високій швидкості навчання та роботи, що робить його особливо ефективним для обробки великих обсягів даних [9].

Основними перевагами методу є висока швидкість роботи, навіть з великими обсягами даних, точність прогнозування, особливо коли використовуються великі набори даних та підтримка категоріальних ознак без необхідності їх перетворення.

У цілому, LightGBM є потужним і ефективним алгоритмом градієнтного бустінгу, який добре підходить для вирішення різноманітних задач класифікації та регресії.

9. Gradient Boosting – це потужний ансамблевий метод машинного

навчання, що використовується для класифікації та регресії. Він побудований на основі ідеї послідовного навчання слабких моделей (зазвичай, дерев рішень) для коригування помилок попередніх моделей. Основна мета Gradient Boosting - мінімізувати похибку моделі шляхом побудови нових моделей, які компенсують помилки попередніх [10].

До основних переваг методу відносяться, такі як висока точність, гнучкість та можливість роботи з різними типами даних.

1.3 Вибір оптимальних інформаційних технологій для прогнозування

Оскільки основним завданням даної роботи є прогнозування цін мобільних телефонів з використанням методів машинного навчання, доцільно використовувати мову програмування Python.

Python, створений Гвідо ван Россумом і вперше випущений в 1991 році, є високорівневою інтерпретованою об’єктно-орієнтованою мовою з динамічною семантикою, назва “ Python ” була обрана на честь британської комедійної групи Monty Python, тому мова стала простою та веселою. Python відомий своєю корисністю для початківців, оскільки дозволяє зосередитися на концепціях програмування, а не на деталях, які часто ускладнюють інші мови [12].

Python широко використовується у веб-розробці, розробці програмного забезпечення, математичних розрахунках та автоматизації систем. Його популярність обумовлена швидким створенням додатків і можливістю інтеграції з іншими мовами і технологіями. Завдяки багатим вбудованим структурам даних, динамічному набору тексту та макету, Python робить процес розробки дуже простим, швидким та ефективним. Акцент на зрозумілому синтаксисі та читабельності зменшує витрати на підтримку вашого коду. Крім того, підтримка модулів та пакетів полегшує створення модульних додатків та повторне використання коду. Python – це мова з відкритим кодом, яка сприяє активній розробці та розширенню бібліотек спільнотою розробників.

Приклади використання Python:

- Створення веб-додатків на сервері;

- Побудова робочих процесів, які можна використовувати в поєднанні з програмним забезпеченням;
- Розробка веб-додатків на стороні сервера;
- Автоматизація робочого процесу та написання сценаріїв;
- Взаємодія з базою даних;
- Обробка та аналіз файлів;
- Виконання складних математичних обчислень;
- Робота з великими даними;
- Розробка програмного забезпечення для виробництва.

На професійному рівні Python підходить для веб-розробки, аналізу даних, штучного інтелекту та наукових обчислень. Python також використовується для створення інструментів продуктивності, ігор та настільних додатків.

Переваги Python:

- Підтримка різних платформ, таки як Windows, Mac, Linux, Raspberry Pi та інші;
- Простий синтаксис, що дозволяє писати менше коду порівняно з іншими мовами програмування;
- Це інтерпретатор, який може виконувати код негайно, полегшуючи швидке створення прототипів;
- Python підтримує процедурний об’єктно-орієнтований та функціональний підходи до програмування.

Гнучкість Python дозволяє уникати суворих правил створення функцій і дає більше свободи у використанні різних методів для вирішення завдань. Python також дозволяє запускати програми в проблемних областях, використовуючи перевірку типу під час виконання.

Python та штучний інтелект (AI): Python часто вибирають для досліджень AI. Такі бібліотеки, як TensorFlow, scikit-learn і Keras, створюють міцну основну для розробки рішень в області штучного інтелекту, забезпечуючи зручність і гнучкість. Ці бібліотеки дозволяють розробникам зосередитися на інноваціях та вдосконаленні технологій.

Індекс пакетів Python (PyPI) – це сховище програмнго забезпечення, яке допомагає користувачам знаходити та встановлювати програми, створенні спільнотою розробників Python [16].

1.4 Висновки

У цьому розділі було проаналізовано об’єкт дослідження та визначено актуальність передбачення ціни на телефони за допомогою методів машинного навчання. Проведено опис та аналіз методів передбачення, в результаті чого було обрано декілька методів, які найкраще підходять для передбачення цін на мобільні телефони, а саме Random Forest, XGBoost, AdaBoost, Extra Trees.

2 АНАЛІЗ ТА ОБРОБКА ДАНИХ

2.1 Підготовка даних

Перш ніж приступити до роботи, необхідно підготувати дані та імпортувати бібліотеки. Першим кроком був імпорт усіх бібліотек, необхідних для виконання 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
from scipy import stats
```

Рисунок 2.1 – Основні бібліотеки

```
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 – Бібліотеки для візуалізації моделей

Наступним кроком було завантажено дані та виведено загальну інформацію про них, а також наявну кількість категоріальних та числових змінних (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 – Загальна інформація про дані

	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 – Кількість 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 – Кількість числових змінних

Даний датасет містить 2000 екземплярів даних. Загальна кількість змінних дорівнює 21, включаючи 20 незалежних і 1 залежну зміну. У наборі даних міститься 8 категоріальних та 13 числових змінних. Наведемо розшифровані абревіатури доступних змінних:

– battery_power – загальна кількість енергії, яку акумулятор може зберігати за один раз, вимірюється в mAh;

– blue – має Bluetooth чи ні;

– clock_speed – швидкість, з якою мікропроцесор виконує інструкції;

– dual_sim – має підтримку двох SIM-карт чи ні;

– fc – фронтальна камера к-сть мегапікселів;

– four_g – має 4G чи ні;

– int_memory – внутрішня пам'ять в гігабайтах;

– m_dep – глибина телефону;

– mobile_wt – вага мобільного телефону;

– n_cores – кількість ядер процесора;

– pc – основна камера к-сть мегапікселів;

– px_height – висота роздільної здатності пікселів;

– px_width – ширина роздільної здатності пікселів;

– ram – оперативна пам'ять в мегабайтах;

– sc_h – висота екрану мобільного в см;

– sc_w – ширина екрану мобільного в см;

– talk_time – найдовший час, на який вистачить одного заряду акумулятора;

– three_g – підтримує 3G чи ні;

– touch_screen – має сенсорний екран чи ні;

– wifi – є wifi чи ні;

– price_range – це цільова змінна зі значеннями 0 (низькі витрати), 1 (середні витрати), 2 (високі витрати) та 3 (дуже високі витрати).

Після чого було побудовано розподіл цільової ознаки, який показано на рисунку 2.6.

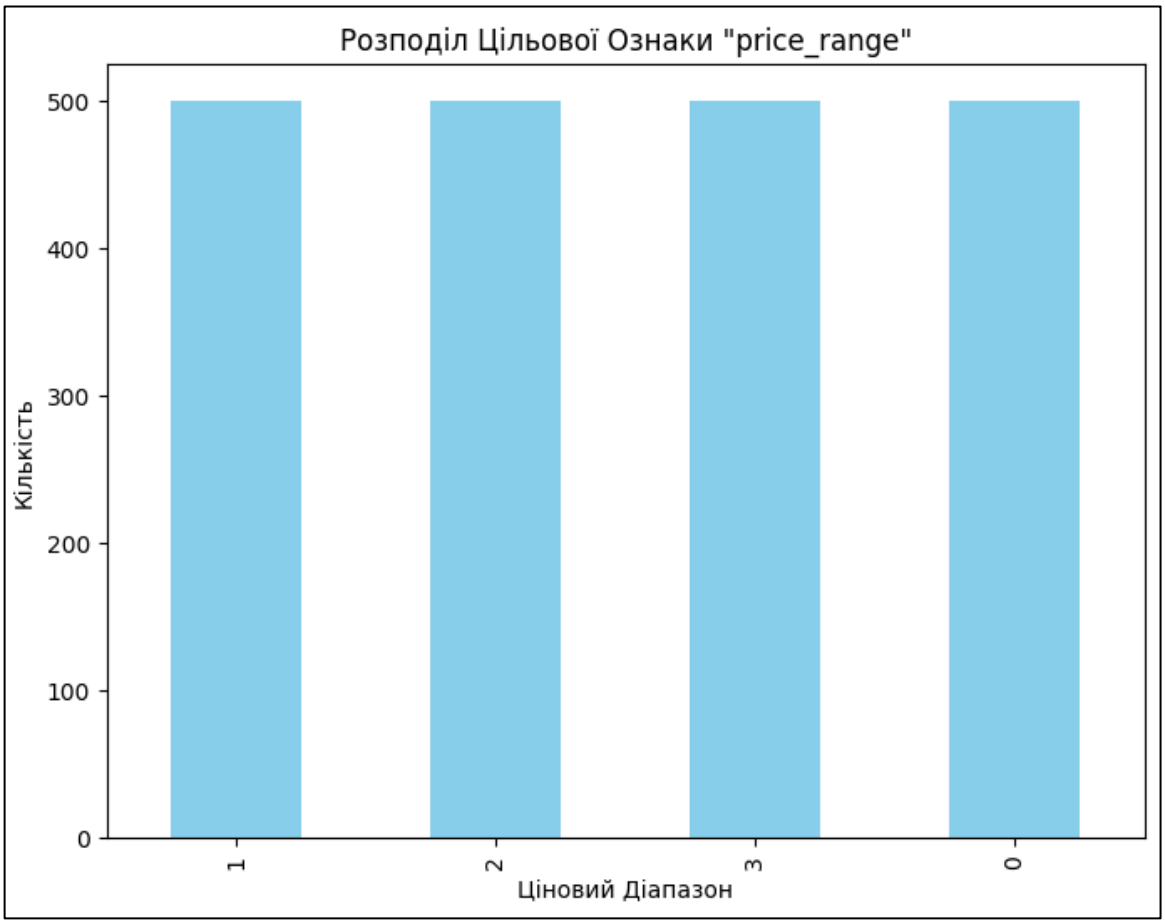


Рисунок 2.6 – Розподіл цільової ознаки price_range

Наведений графік цільової ознаки показує, що кількість спостережень у кожному з цінових діапазонів рівномірна. У кожному з діапазонів (0, 1, 2, 3) є приблизно однакова кількість даних (близько 500 спостережень). Це означає, що розподіл цільової ознаки "price_range" є збалансованим, тобто немає явної переваги будь-якого з діапазонів.

2.2 Розвідувальний аналіз даних

Розвідувальний аналіз даних (Exploratory Data Analysis, EDA) – це метод, який використовують фахівці з обробки даних для розуміння наборів даних перед початком моделювання. Іноді це називається дослідженням даних. Метою EDA є визначення характеристик набору даних. Впровадження EDA допомагає аналітикам даних робити прогнози та припущення щодо своїх даних. EDA зазвичай передбачає візуалізацію даних, особливо створення графіків, таких як

гістограми, діаграми розсіювання та прямокутні діаграми. Exploratory Data Analysis використовується для визначення взаємозв’язків між змінними в ситуаціях, коли апріорне уявлення про природу цих взаємозв’язків відсутнє або недостатнє. Як правило, в процесі аналізу розвідувальної інформації розглядається, порівнюється велика кількість змінних і використовуються різні методи для виявлення закономірностей.

Основні методи розвідувального аналізу даних:

- визначення основної структури;
- відбір найбільш важливих змінних;
- виявлення аномалій і відхилень від норми;
- перевірка основної гіпотези;
- розробка вихідної моделі.

Першим кроком розвідувального аналізу було побудовано інтерактивні кругові діаграми, для того щоб дослідити розподіл категоріальних ознак (рис 2.7).

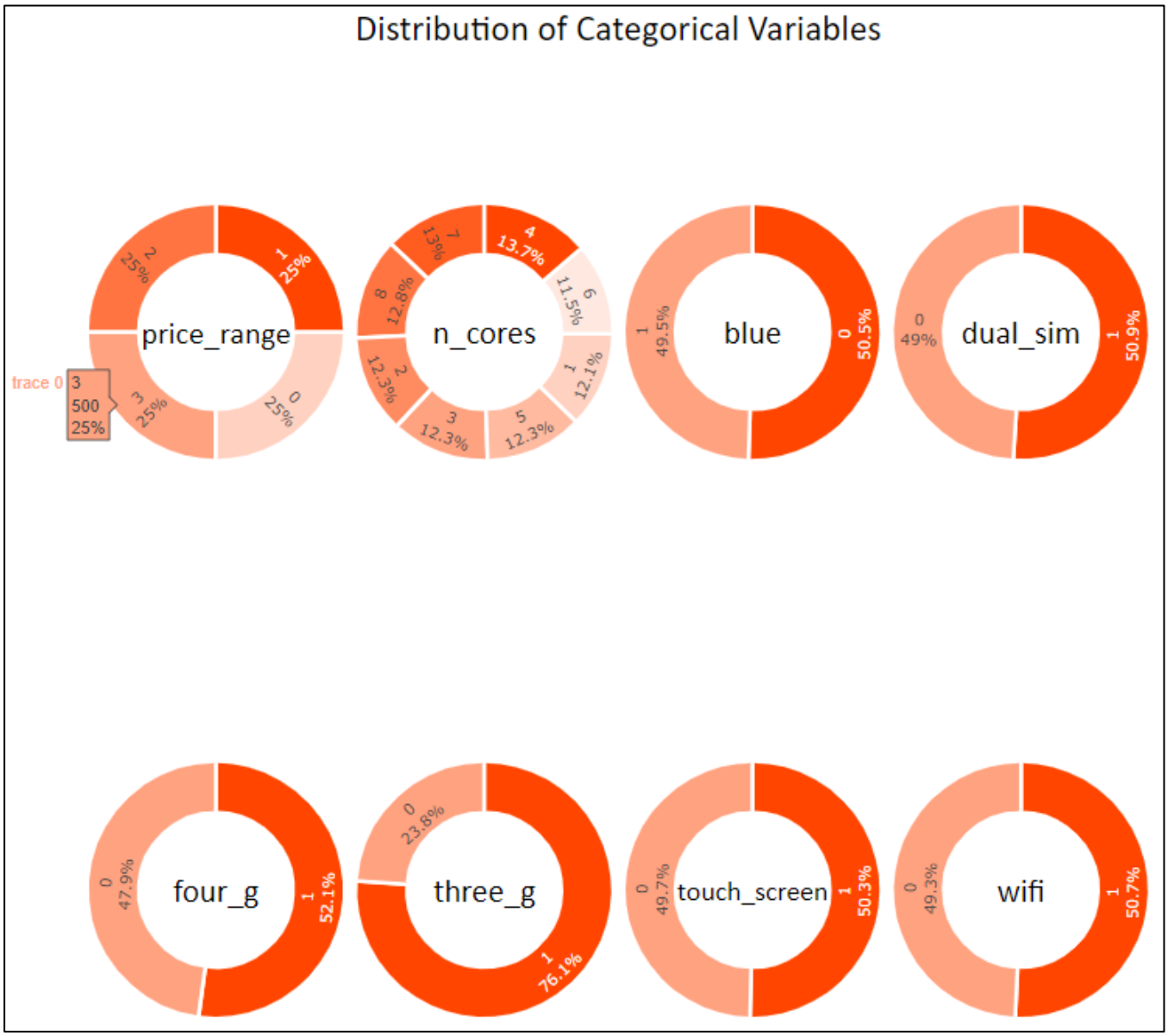


Рисунок 2.7 – Інтерактивні кругові діаграми категоріальних ознак

З наведених інтерактивних кругових діаграм категоріальних ознак можна зробити висновок, що мобільні телефони розподілені з однаковою частотою між чотирма класами price_range. Це свідчить про те, що набір даних є повністю збалансованим.

Також мобільні телефони в наборі даних мають майже однакову частоту з точки зору наявності або відсутності Bluetooth, 4G, сенсорного екрану, Wifi, а також підтримки двох SIM-карт і кількості використовуваних процесорних ядер.

Близько 76% мобільних телефонів підтримують 3G.

Наступним кроком було побудовано гістограми для дослідження розподілу числових ознак (рис 2.8-2.10).

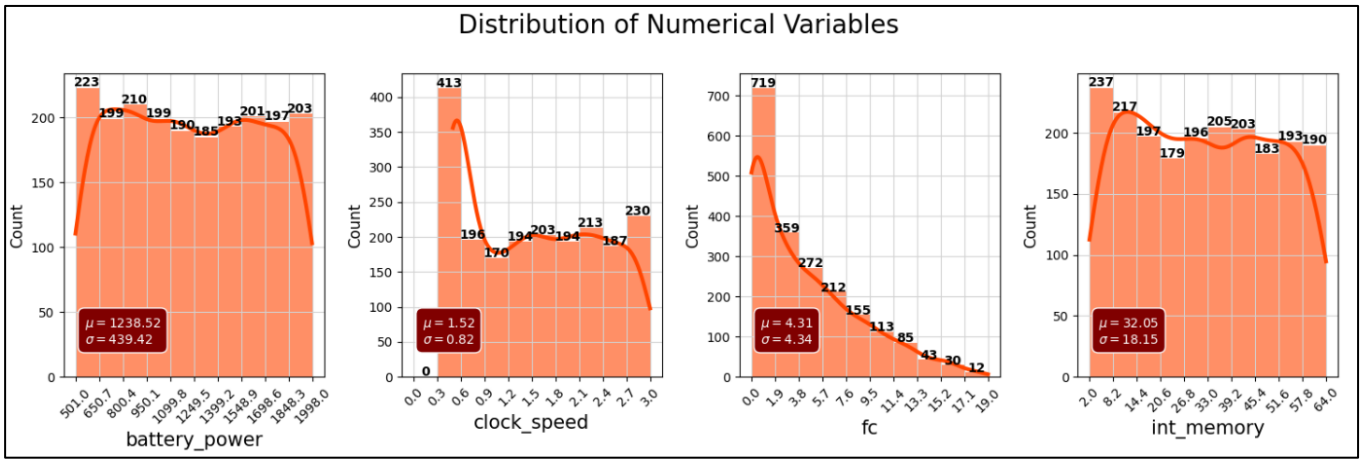


Рисунок 2.8 – Гістограми перших чотирьох числових ознак

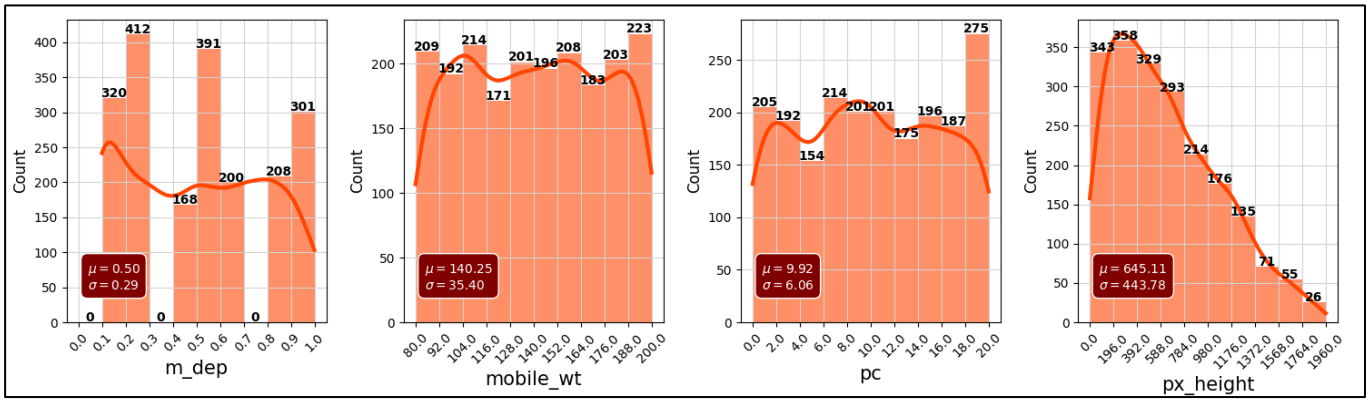


Рисунок 2.9 – Гістограми наступних чотирьох числових ознак

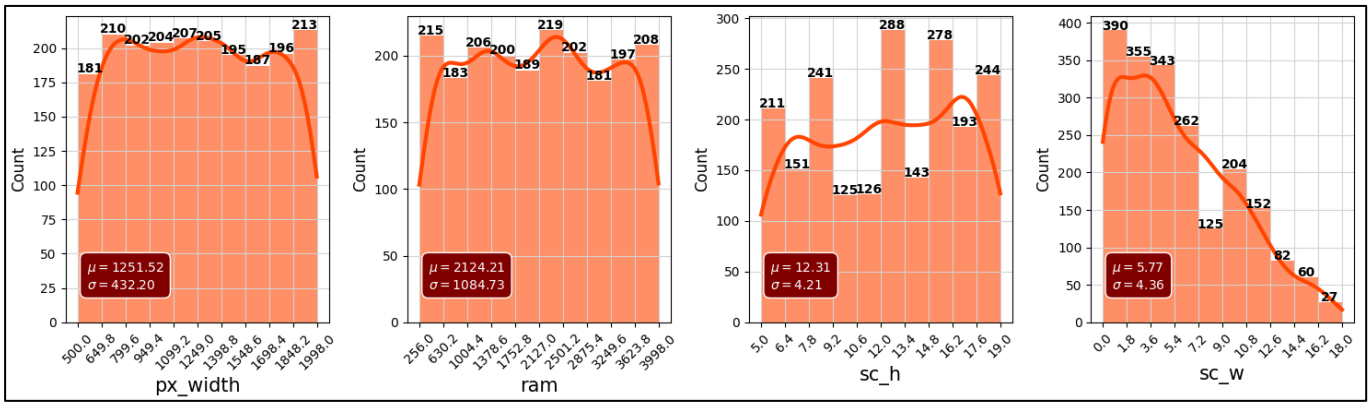


Рисунок 2.10 – Гістограми останніх числових ознак

Дивлячись на наведені графіки вище, ми можемо легко зрозуміти статистичні значення кожної ознаки, включаючи мінімальне і максимальне значення, а також середнє і стандартне відхилення. У деяких характеристиках,

зокрема px_height (висота роздільної здатності пікселів) та sc_w (ширина екрану мобільного в см), спостерігається багато значень, близьких до нуля, які здаються шумом.

Також в ході розвідувального аналізу було, побудовано три типи графіків, а саме: scatter plot, histogram та box plot, які дають можливість швидко порівняти взаємозв'язок та розподіл даних для аналізу цінових діапазонів.

Залежність між ємністю акумулятора та ціновим діапазоном показано на рисунку 2.11.

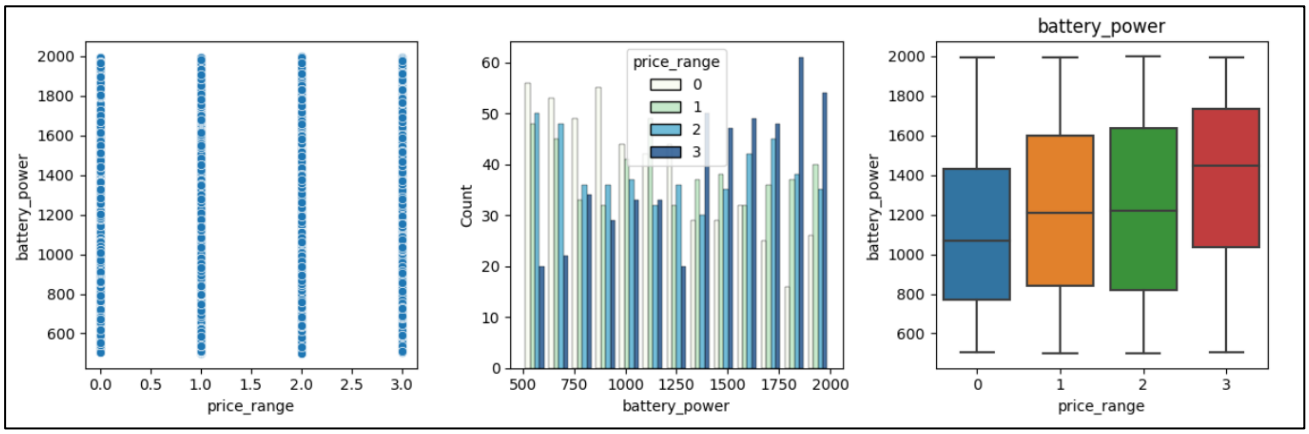


Рисунок 2.11 – Залежність між ємністю акумулятора та ціновим діапазоном

На основі наведених графіків можна зробити наступний висновок, який свідчить про те, що зі зростанням цінового діапазону мобільних телефонів збільшується потужність їхніх акумуляторів. У вищих цінових категоріях середня потужність акумулятора є значно вищою, що вказує на те, що дорожчі телефони зазвичай обладнані більш ємними акумуляторами. Це підтверджується як розподілом даних на скатерплоті та гістограмі, так і аналізом медіан та міжквартильних розмахів на коробковому графіку.

Залежність між наявністю функції Bluetooth та ціновим діапазоном показана на рисунку 2.12.

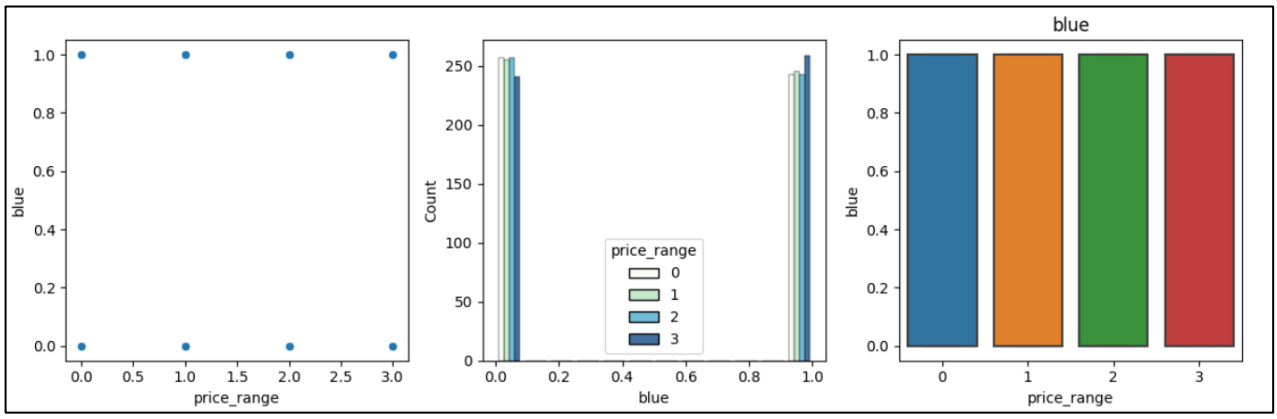


Рисунок 2.12 – Залежність між наявністю функції Bluetooth та ціновим діапазоном

До відповідно наведених графіків можна зробити висновок, що наявність Bluetooth у мобільних телефонах не залежить від їхнього цінового діапазону. У кожній ціновій категорії телефони з Bluetooth і без нього рівномірно розподілені, що свідчить про відсутність кореляції між наявністю цієї функції та ціною. Тобто, незалежно від цінового діапазону, телефони однаково часто оснащені Bluetooth.

Залежність між швидкістю роботи мікропроцесора та ціновим діапазоном зображено на рисунку 2.13.

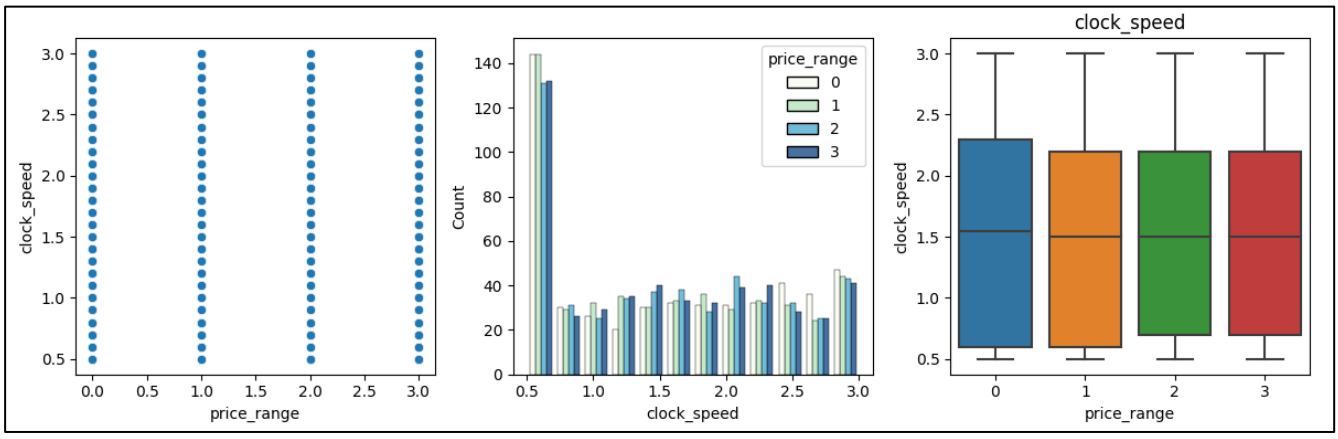


Рисунок 2.13 – Залежність між швидкістю роботи мікропроцесора та ціновим діапазоном

Дані графіки демонструють те, що існує пряма пропорційна залежність між швидкістю роботи мікропроцесора та ціною. Це свідчить про те, що телефони з вищою швидкістю мікропроцесора, як правило, мають і вищу ціну.

Залежність між кількістю SIM-карт та ціновим діапазоном відображена на рисунку 2.14.

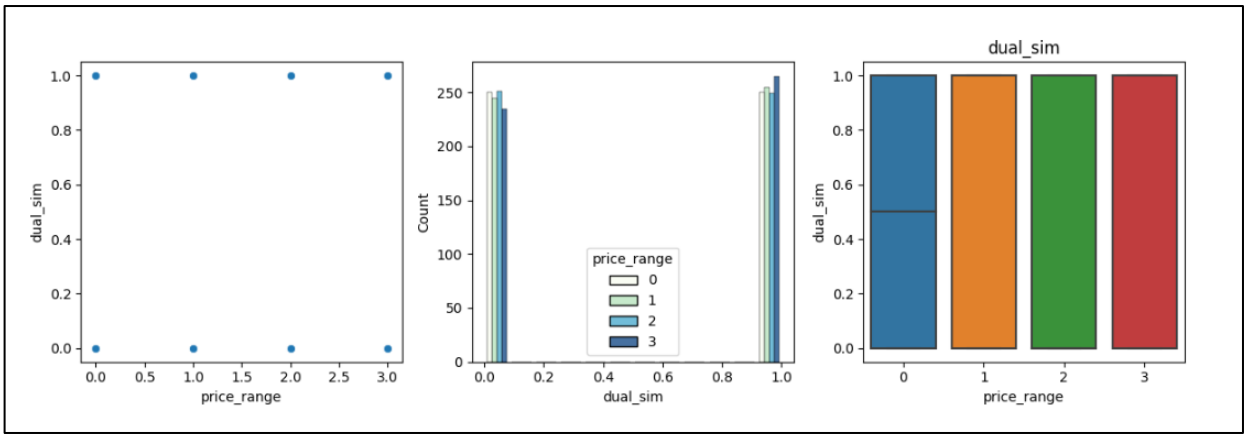


Рисунок 2.14 – Залежність між кількістю SIM-карт та ціновим діапазоном

Наведені графіки свідчать про те, що мобільні телефони з двома SIM-картками, як правило, трохи дорожчі, ніж мобільні телефони з однією SIM-карткою. Ця різниця в ціні може бути пов'язана з додатковими функціями та можливостями, які пропонують мобільні телефони з двома SIM-картками.

Залежність між кількістю фронтальних камер смартфона та ціновим діапазоном показано на рисунку 2.15.

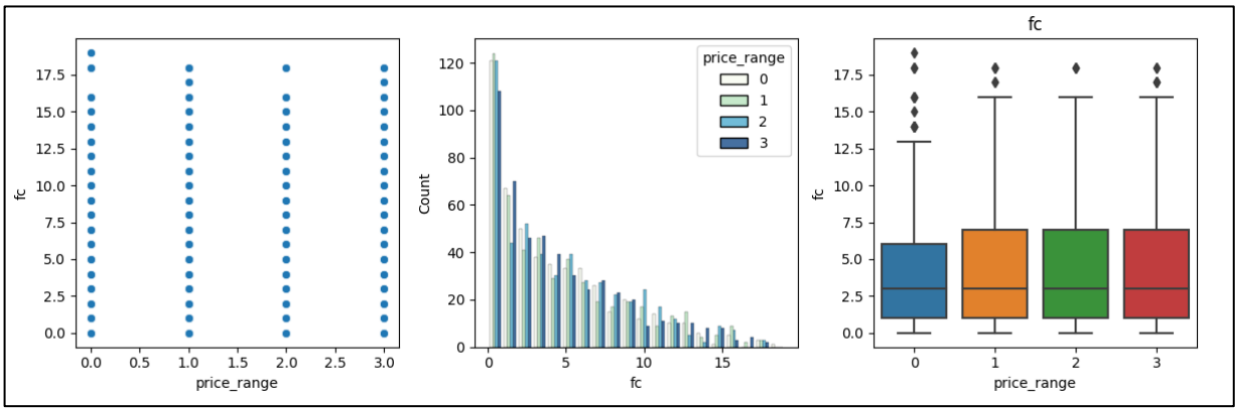


Рисунок 2.15 – Залежність між кількістю фронтальних камер смартфона та ціновим діапазоном

На даних графіках можна спостерігати, як кількість фронтальних камер на смартфонах змінюється в залежності від їх цінового діапазону. Можливо, більш дорогі моделі мають тенденцію мати більше фронтальних камер, щоб задовольнити потреби користувачів у якісних селфі та відеодзвінках.

Залежність між внутрішньою пам’яттю смартфона та ціновим діапазоном зображена на рисунку 2.16.

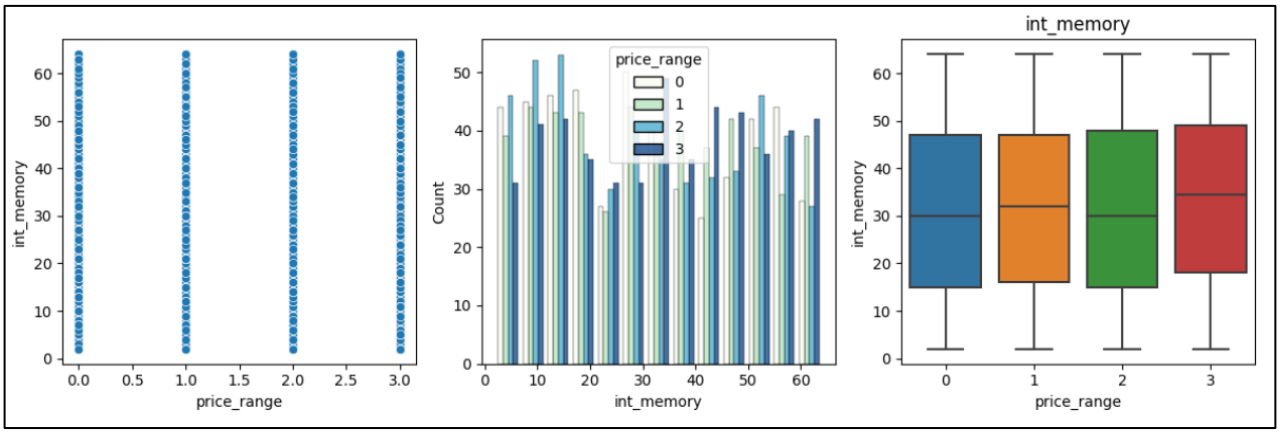


Рисунок 2.16 – Залежність між внутрішньою пам’яттю смартфона та ціновим діапазоном

Аналізуючи наведені графіки, можна зробити такі висновки, що значення обсягу внутрішньої пам'яті суттєво впливає на ціни мобільних телефонів. Це може бути пов'язано з особливостями деяких моделей або може бути наслідком різних стратегій виробників щодо включення певних характеристик у конкретні моделі.

Залежність між товщиною смартфона та ціновим діапазоном наведено на рисунку 2.17.

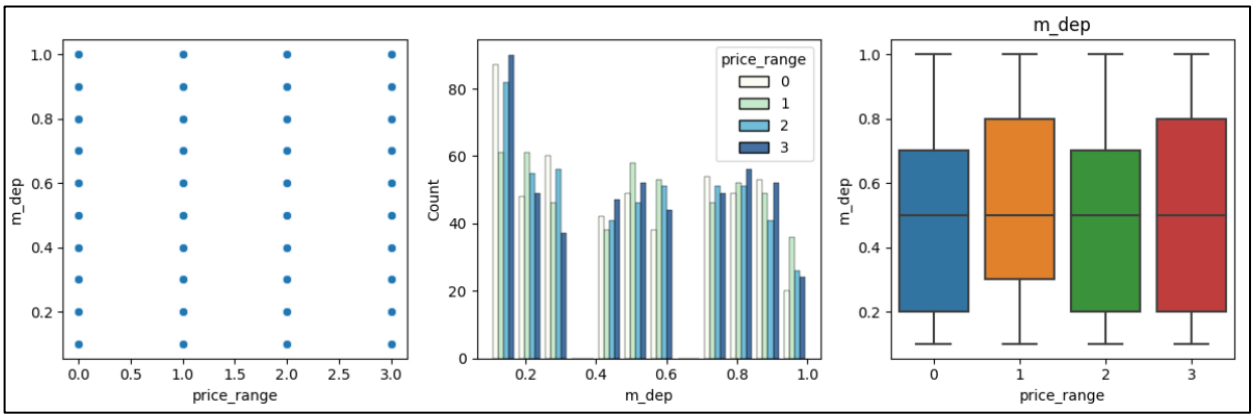


Рисунок 2.17 – Залежність між товщиною смартфона та ціновим діапазоном

Зображені графіки показують, як змінюється товщина смартфона в залежності від його цінового діапазону. Це вказує на те, що ціна може бути фактором, що впливає на розмір і дизайн смартфона. Наприклад, більш дорогі моделі можуть мати більш витончений дизайн, щоб забезпечити більшу елегантність і привабливість для покупців.

Залежність між вагою смартфона та ціновим діапазоном рисунок 2.18.

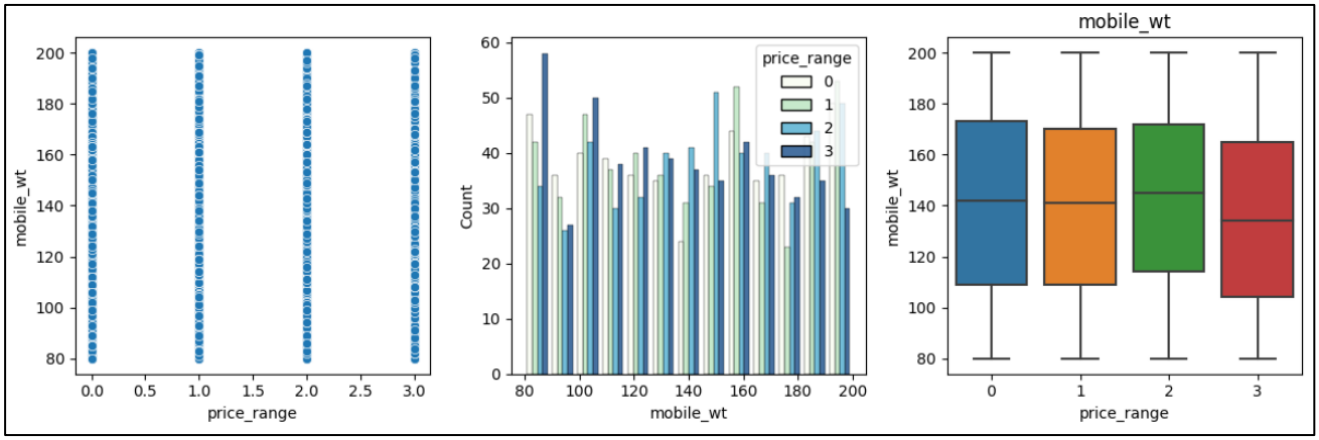


Рисунок 2.18 – Залежність між вагою смартфона та ціновим діапазоном

Ці графіки показують, як змінюється вага смартфона в залежності від його ціни. Можливо, дорожчі моделі, як правило мають меншу вагу, щоб забезпечити більшу мобільність та привабливість для клієнтів.

Залежність між кількістю ядер процесора та ціновим діапазоном показано на рисунку 2.19.

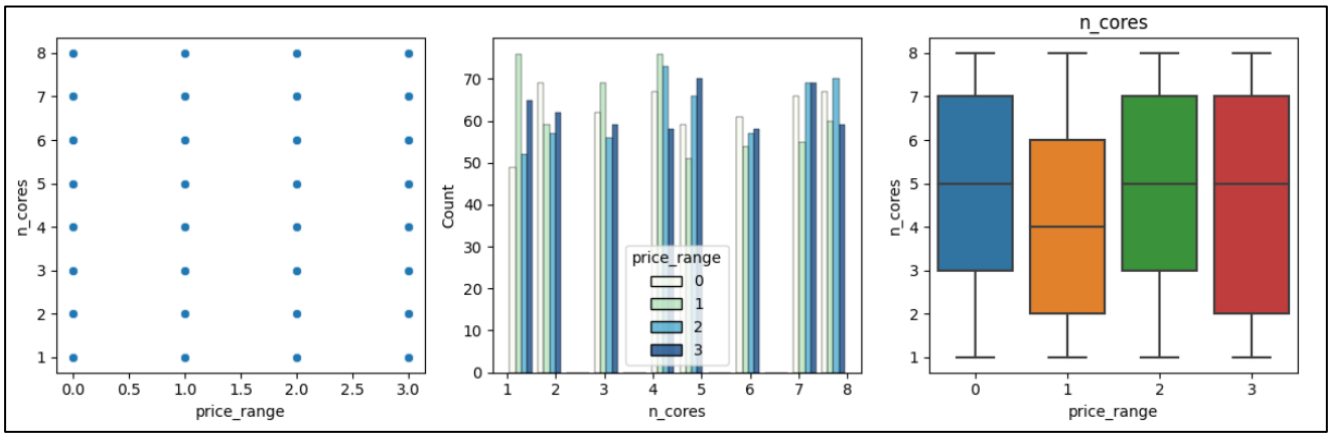


Рисунок 2.19 – Залежність між кількістю ядер процесора та ціновим діапазоном

Зображені графіки вказують на те, що мобільні телефони з більшою кількістю ядер процесора, як правило, дорожчі. Ця різниця в ціні може бути пов'язана з тим, що мобільні телефони з більшою кількістю ядер процесора, як правило, мають кращу продуктивність та багатозадачність.

Залежність між кількістю мегапікселів основної камери та ціновим діапазон наведено на рисунку 2.20.

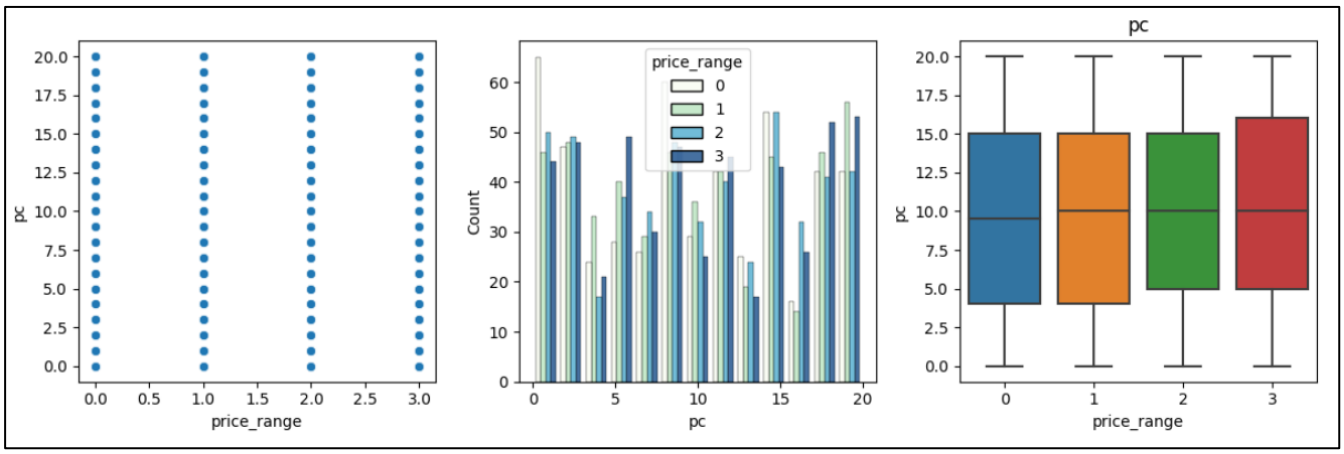


Рисунок 2.20 – Залежність між кількістю мегапікселів основної камери та ціновим діапазон

Наведені графіки свідчать про те, що мобільні телефони з більшою кількістю мегапікселів основної камери, як правило, дорожчі. Ця різниця в ціні може бути пов'язана з тим, що мобільні телефони з більшою кількістю мегапікселів основної камери роблять кращі фотографії та відео.

Залежність між висотою і шириною роздільної здатності та ціновим діапазоном показано на рисунках 2.21-2.22.

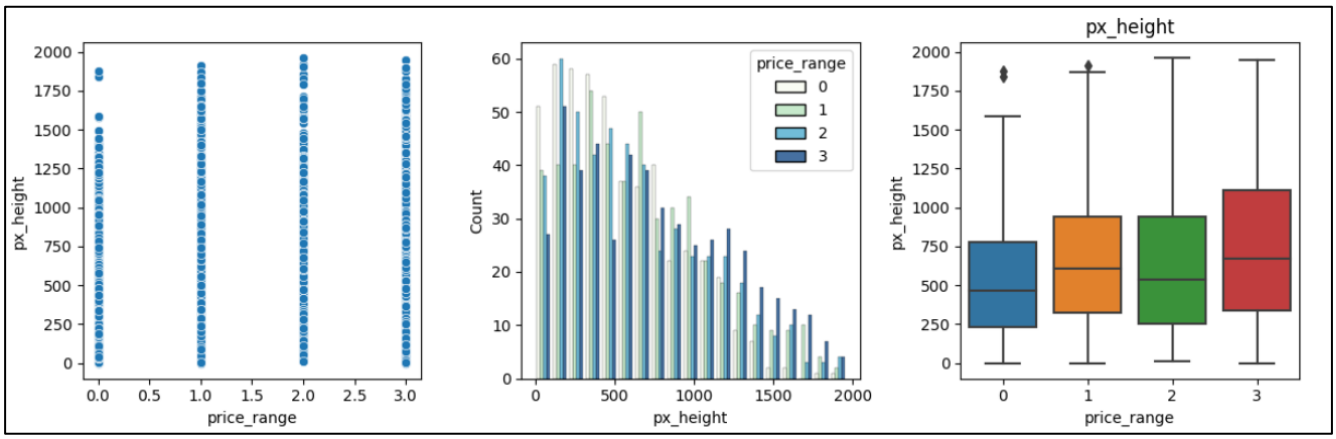


Рисунок 2.21 – Залежність між висотою роздільної здатності та ціновим діапазоном

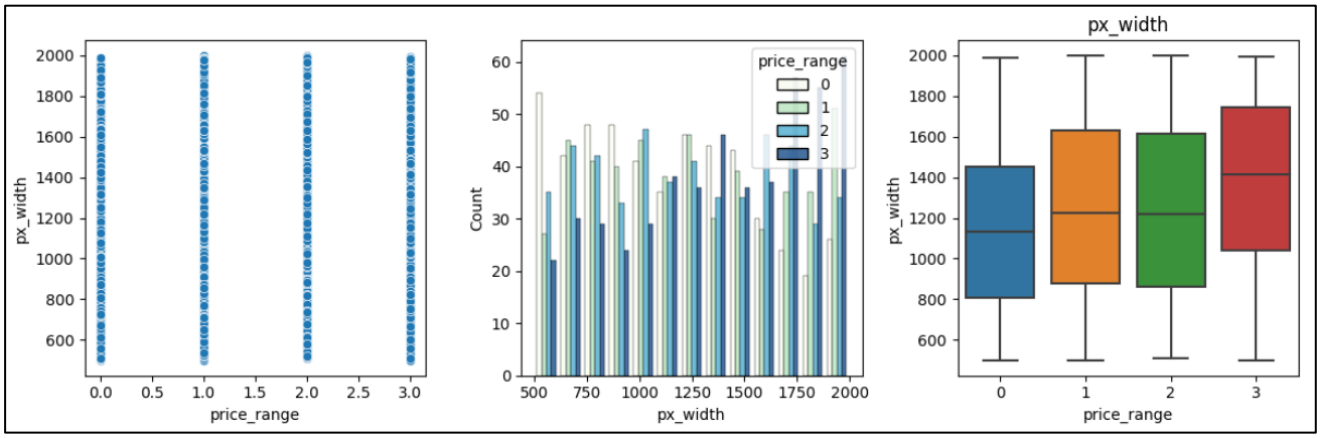


Рисунок 2.22 – Залежність між шириною роздільної здатності та ціновим діапазоном

Аналізуючи графіки, які показані на рисунках 2.21 та 2.22, можна зробити наступні висновки та побачити, що графіки дозволяють виявити, як різні цінові діапазони смартфонів відрізняються за шириною екрану та роздільною здатністю. Можливо, більш дорогі моделі мають більші ширини екрану та роздільну здатність або ж навпаки, що може бути важливою характеристикою для споживачів у виборі смартфону.

Залежність між оперативною пам’яттю та ціновим діапазоном показано на рисунку 2.23.

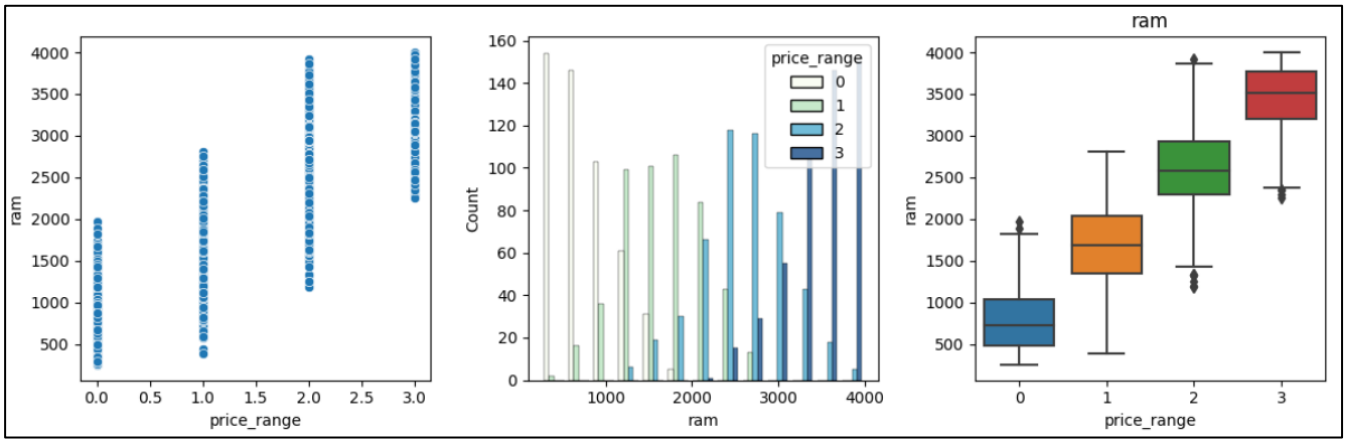


Рисунок 2.24 – Залежність між оперативною пам’яттю та ціновим діапазоном

З наведених графіків видно, що існує певна тенденція збільшення обсягу оперативної пам'яті (RAM) із зростанням ціни смартфона. Це може бути зумовлено включенням більш продуктивних компонентів у більш дорогі моделі.

Залежність між висотою сенсорного екрану та ціновим діапазоном наведено на рисунку 2.25.

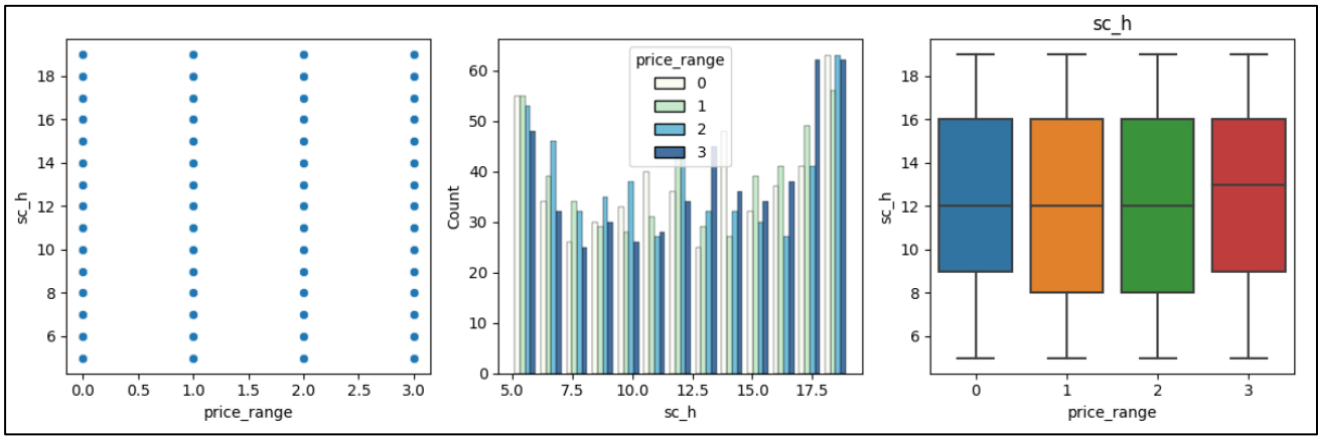


Рисунок 2.25 – Залежність між висотою сенсорного екрану та ціновим діапазоном

Відповідно наведених графіків, які показують розподіл висоти сенсорного екрану (sc_h) у різних цінових категоріях мобільних пристроїв (price_range), можна зробити наступгий висновок про те, що висота сенсорного екрану не є суттєвим фактором для прогнозування цінової категорії мобільного пристрою. Усі цінові категорії демонструють схожий розподіл висоти сенсорного екрану, з медіаною приблизно на рівні 12-14 одиниць та інтерквартильним розмахом (IQR) від 10 до 16 одиниць. Це вказує на те, що параметр висоти сенсорного екрану має незначний вплив на визначення цінової категорії мобільного пристрою.

Залежність між шириною сенсорного екрану та ціновим діапазоном показано на рисунку 2.26.

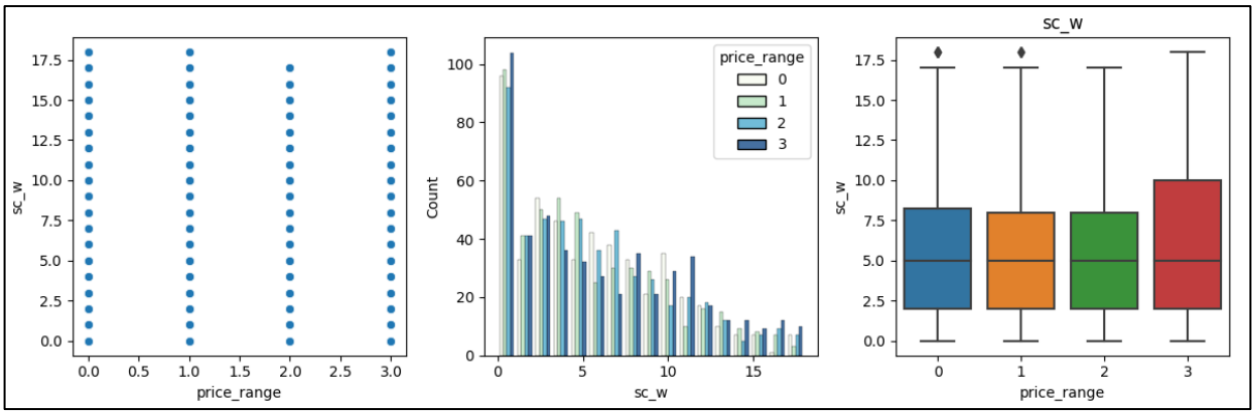


Рисунок 2.26 – Залежність між шириною сенсорного екрану та ціновим діапазоном

Згідно цих графіків, які показують розподіл ширини сенсорного екрану (sc_w) у різних цінових категоріях мобільних пристроїв (price_range), можна зробити висновок, що ширина сенсорного екрану (sc_w) не є суттєвим фактором для прогнозування цінової категорії мобільного пристрою. Хоча у найвищій ціновій категорії (3) спостерігається трохи більша варіативність і вища медіана, загальний розподіл ширини сенсорного екрану в усіх цінових категоріях є подібним. Це свідчить про те, що цей параметр має незначний вплив на визначення цінової категорії.

Залежність між розподілом часу розмови та ціновим діапазоном відображено на рисунку 2.27.

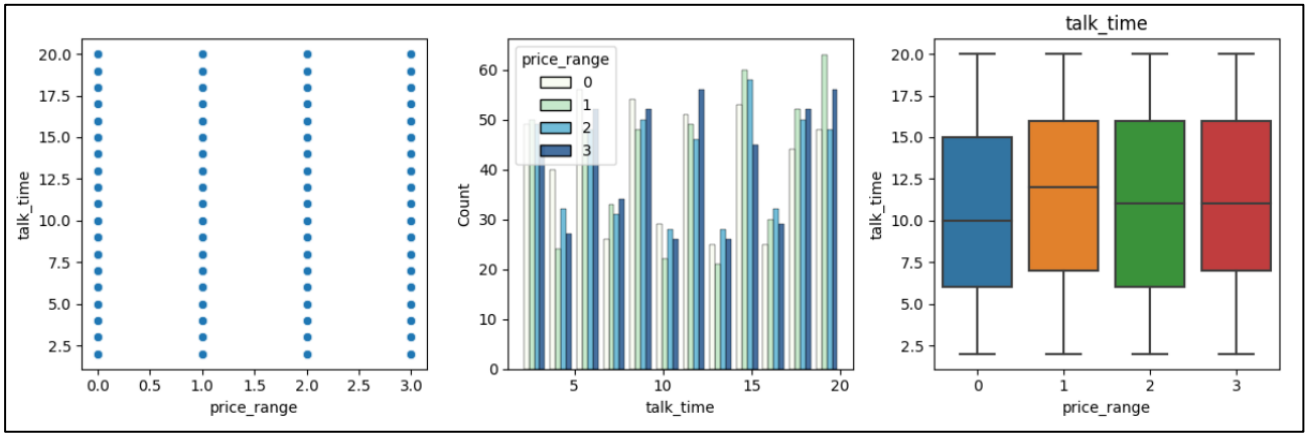


Рисунок 2.27 – Залежність між розподілом часу розмови та ціновим діапазоном

На основі аналізу наведених графіків, які показують розподіл часу розмови (talk_time) у різних цінових категоріях мобільних пристроїв (price_range), можна зробити висновок, що час розмови не є суттєвим фактором для прогнозування цінової категорії мобільного пристрою.

Залежність між наявністю 4 і 3G технологій та ціновим діапазон показано на рисунках 2.27-2.28.

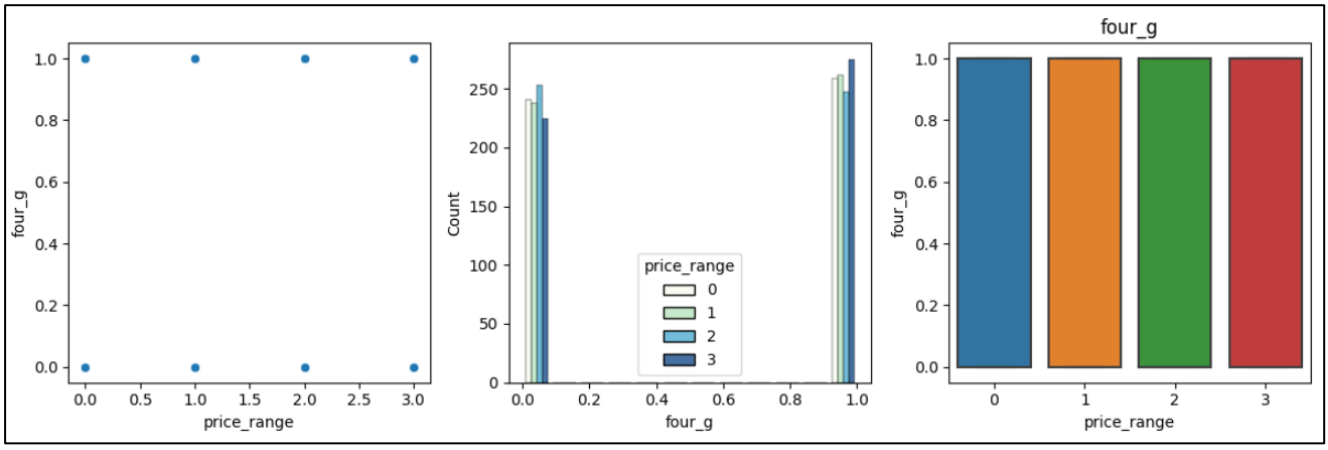


Рисунок 2.27 – Залежність між наявністю 4G та ціновим діапазон

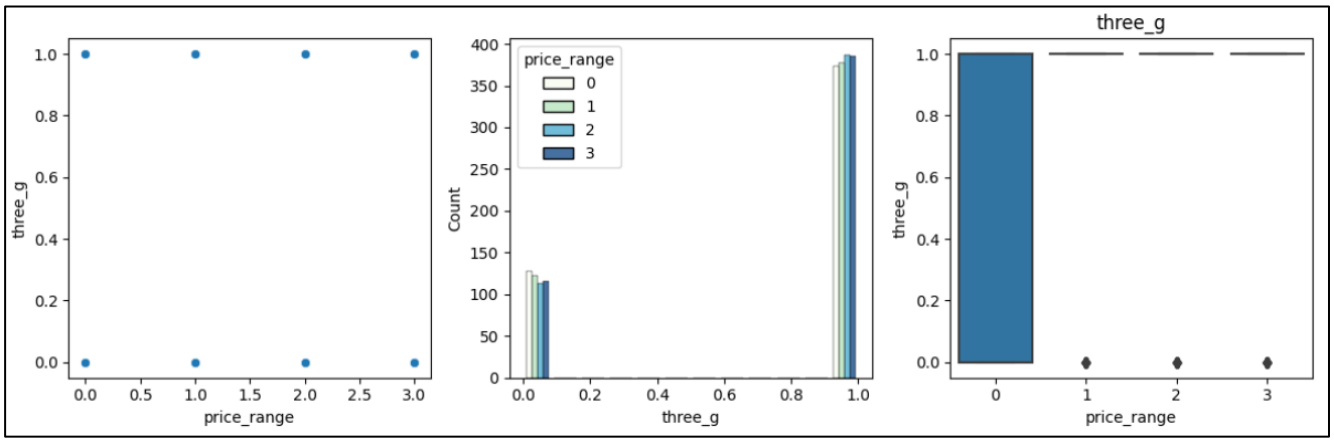


Рисунок 2.28 – Залежність між наявністю 3G та ціновим діапазон

Дані графіки, які зображені на рисунках 2.27 та 2.28 вказують на те, що мобільні телефони з функцією 3 та 4G, як правило, трохи дорожчі, ніж мобільні телефони без 3G. Ця різниця в ціні може бути пов'язана з тим, що 3G є більш просунутою технологією, яка забезпечує більш швидкий мобільний інтернет.

Залежність між наявністю сенсорного екрану та ціновим діапазоном показано на рисунку 2.29.

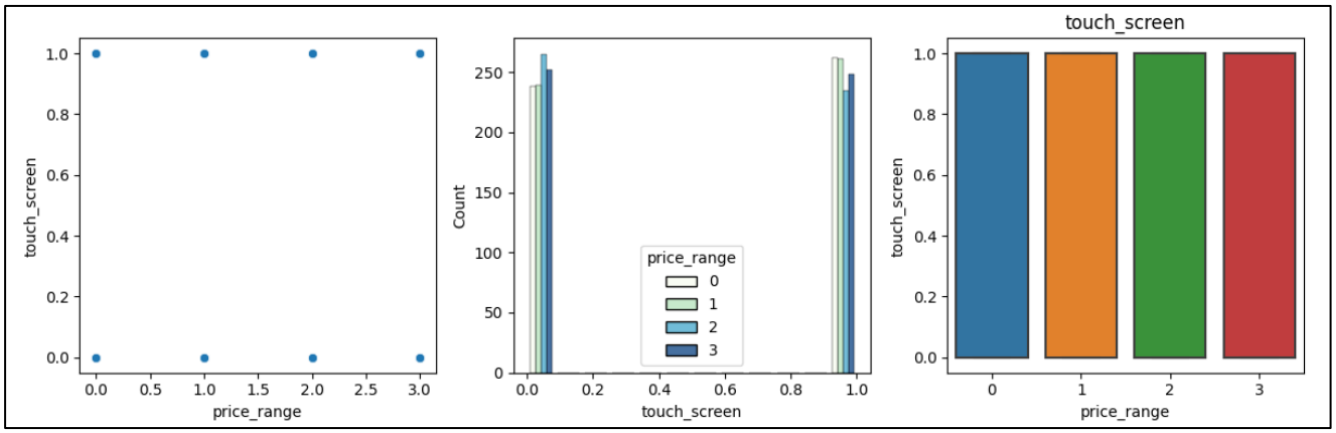


Рисунок 2.29 – Залежність між наявністю сенсорного екрану та ціновим діапазоном

З наведених графіків можна зробити наступний висновок, який свідчить про те, що мобільні телефони з сенсорним екраном, як правило, значно дорожчі, ніж мобільні телефони без сенсорного екрану. Ця різниця в ціні може бути пов'язана з тим, що сенсорні екрани є більш складною та дорогою технологією, ніж традиційні клавіатури.

Залежність між наявністю Wifi та ціновим діапазоном зображено на рисунку 2.30.

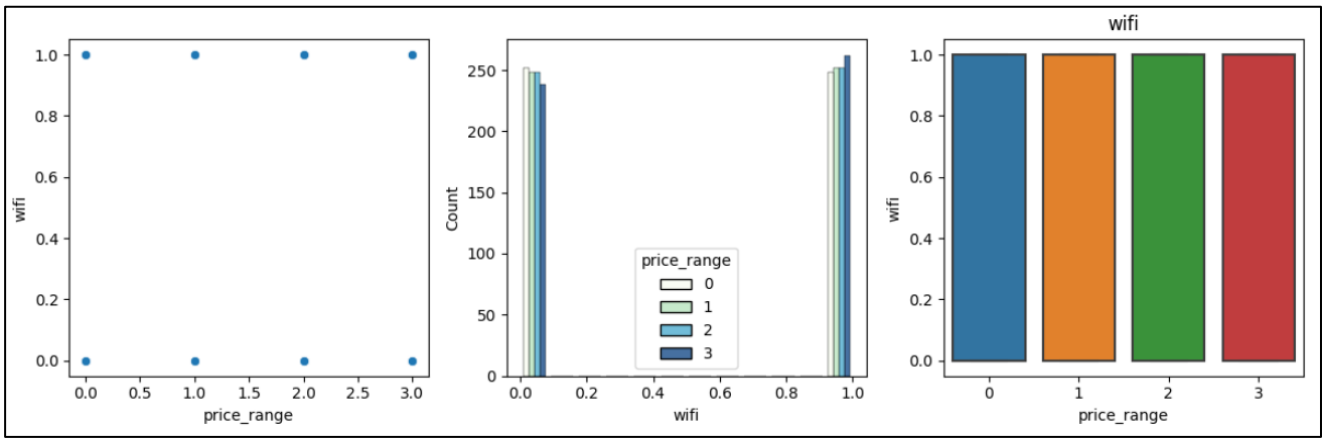


Рисунок 2.30 – Залежність між наявністю Wifi та ціновим діапазоном

Показані графіки показують те, що мобільні телефони з Wifi, як правило, трохи дорожчі, ніж мобільні телефони без Wifi. Ця різниця в ціні може бути пов'язана з тим, що Wifi є популярною функцією, яка робить мобільний телефон більш функціональним.

Також під час розвідувального аналізу було побудовано box plot графіки основних ознак, які суттєво впливають на розподіл цінових діапазонів для мобільних телефонів.

Box-plot графік розподілу потужності батареї для різних цінових категорій відображено на рисунку 2.31.

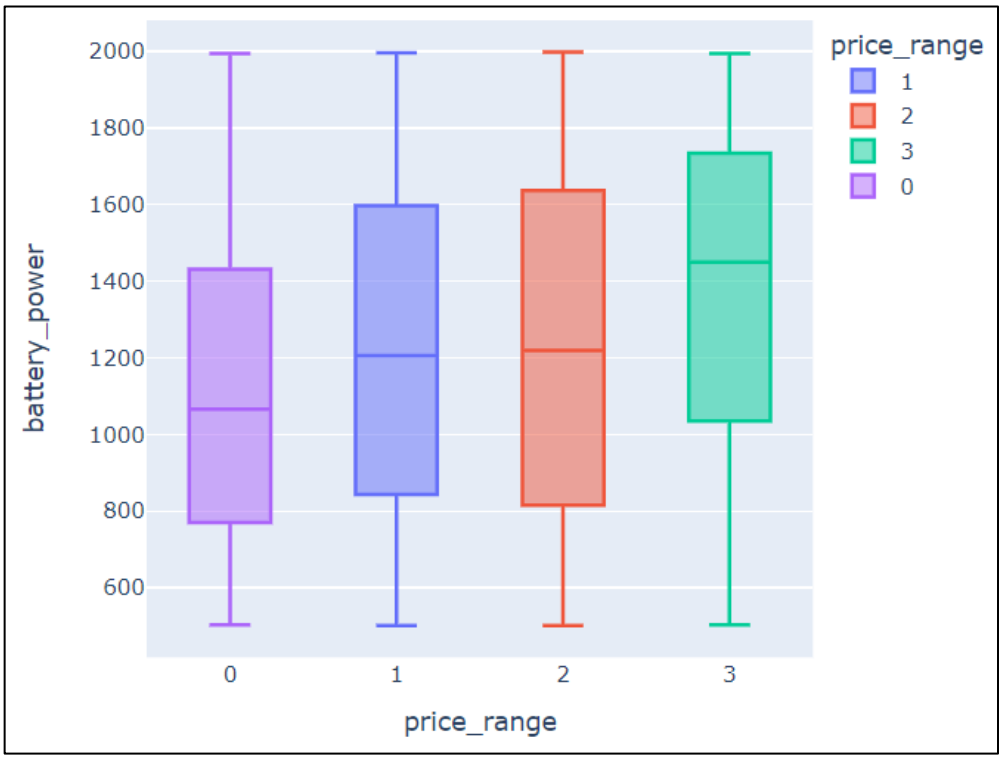


Рисунок 2.31 – Box-plot графік розподілу потужності батареї для різних цінових категорій

Аналізуючи наведений box-plot графік розподілу потужності батареї (battery_power) для різних цінових категорій (price_range) мобільних телефонів, можна зробити висновок, що медіана потужності батареї зростає зі збільшенням цінової категорії, що свідчить про тенденцію до кращої потужності батареї в дорожчих телефонах. У всіх категоріях спостерігається значна варіація потужності батареї, а максимальні значення потужності також зростають разом із ціновою категорією. Таким чином, потужність батареї є важливим фактором, що впливає на ціну телефону: більш дорогі телефони, як правило, оснащені більш потужними аккумуляторами.

Box-plot графік розподілу оперативної пам'яті для різних цінових категорій зображено на рисунку 2.32.

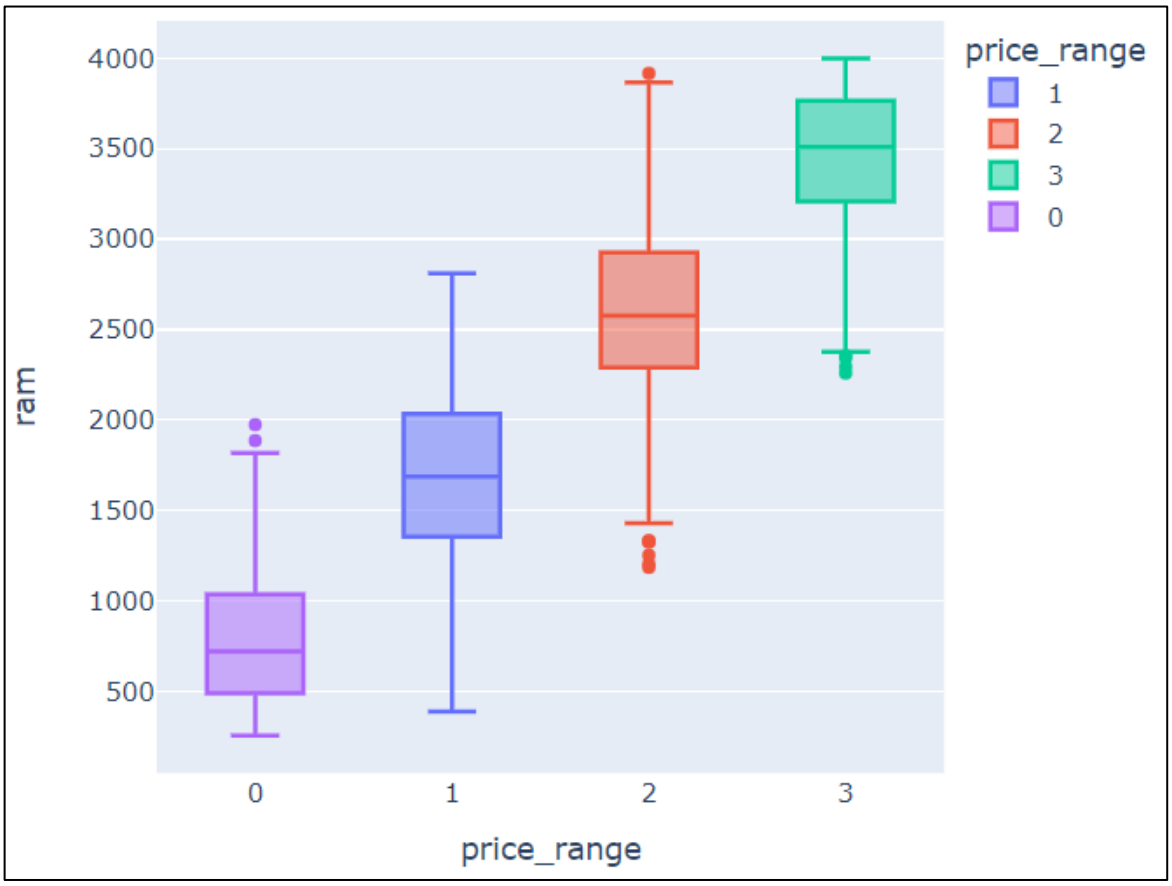


Рисунок 2.32 – Box-plot графік розподілу оперативної пам'яті для різних цінових категорій

Наведений графік відображає вплив обсягу оперативної пам'яті на ціни телефонів у різних цінових діапазонах. З аналізу графіка видно, що зі зростанням обсягу RAM збільшується середня ціна телефону, що свідчить про позитивний вплив цього параметра на ціну. Також спостерігається різноманітність цін у різних цінових діапазонах, а виявлені викиди можуть свідчити про наявність особливо дорогих або дешевих моделей телефонів з певним обсягом оперативної пам'яті. Отже, можна зробити висновок, що оперативна пам’ять є важливим фактором, що впливає на ціну телефону.

Box-plot графік розподілу внутрішньої пам'яті для різних цінових категорій наведено на рисунку 2.33.

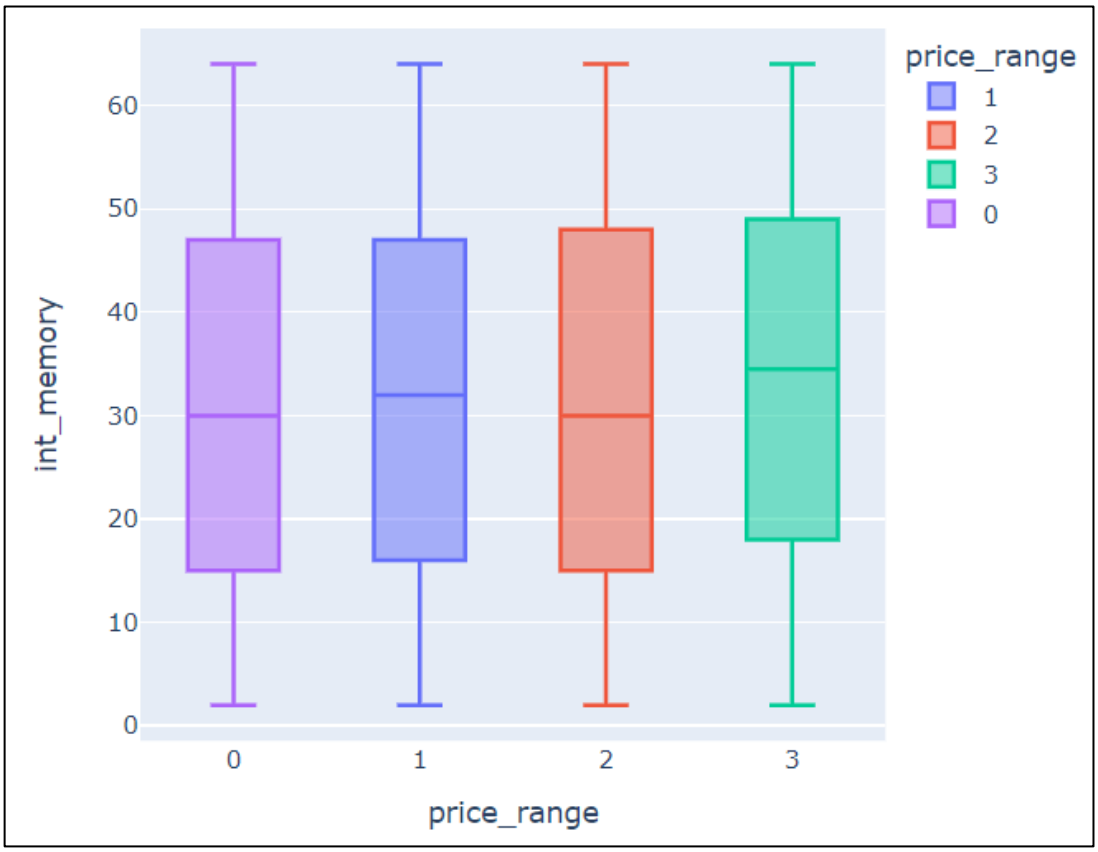


Рисунок 2.33 – Box-plot графік розподілу внутрішньої пам'яті для різних цінових категорій

З наведеного графіка, можна зробити такі висновки, що значення обсягу внутрішньої пам'яті суттєво впливає на ціни мобільних телефонів. Це може бути пов'язано з особливостями деяких моделей або може бути наслідком різних стратегій виробників щодо включення певних характеристик у конкретні моделі.

Box-plot графік розподілу швидкості мікропроцесора для різних цінових категорій показано на рисунку 2.34.

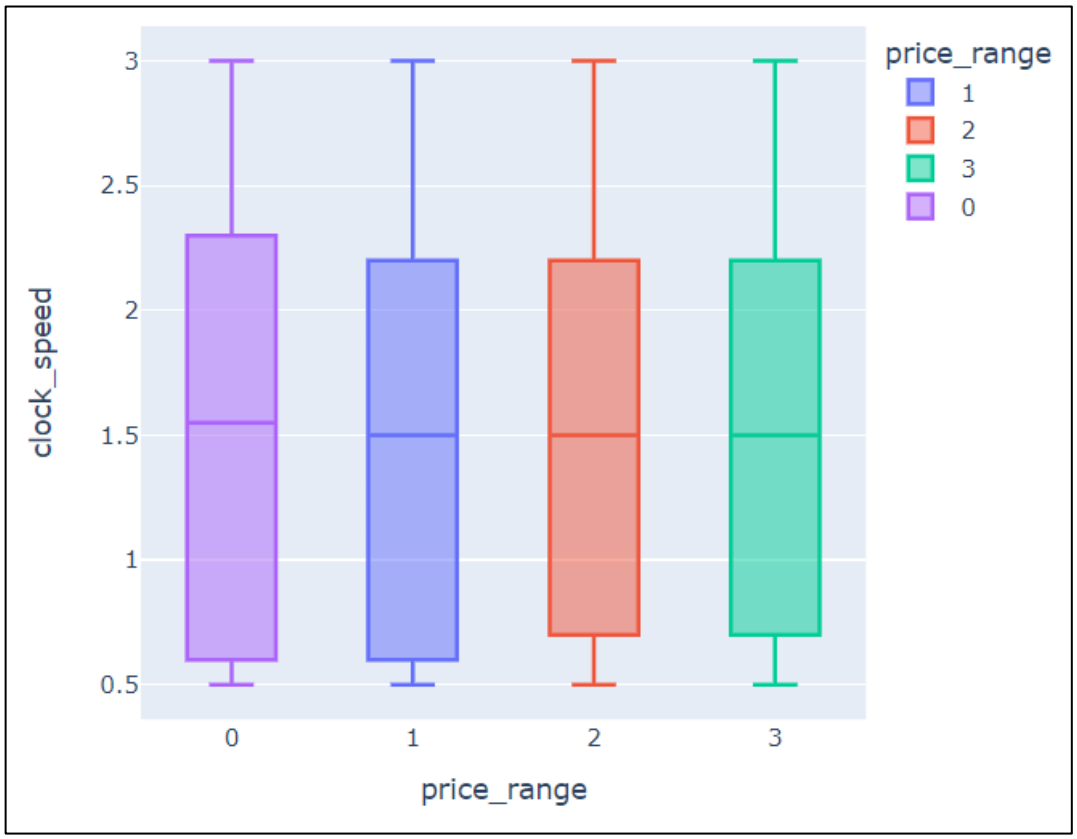


Рисунок 2.34 – Box-plot графік розподілу швидкості мікропроцесора для різних цінових категорій

Наведений графік демонструє, як тактова частота процесора змінюється в залежності від цінового діапазону смартфонів. Зазвичай, більш дорогі моделі мають потужніший процесор з вищою тактовою частотою, що може підвищити продуктивність і швидкість роботи пристрою.

Також в ході розвідувального аналізу було побудовано теплову карту, яка служить для візуалізації кореляційної матриці, де кожна комірка відображає коефіцієнт кореляції між двома змінними (рис. 2.35).

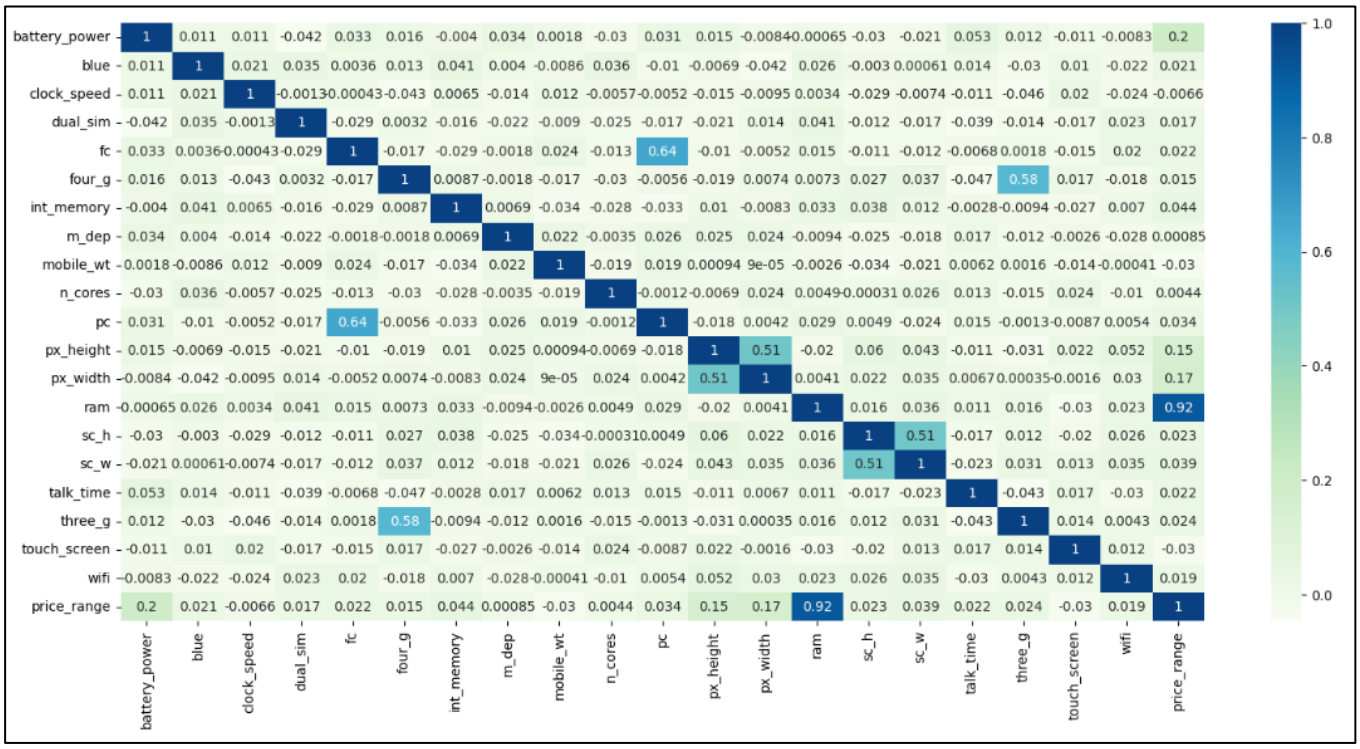


Рисунок 2.35 – Теплова карта

Відповідно наведеної теплової карти можна зробити наступні висновки про те, що існує багато сильних кореляцій між змінними. Але найбільш значуща кореляція спостерігається між об'ємом оперативної пам'яті (**ram**) та ціновою категорією пристрою (**price_range**). Така висока кореляція свідчить про те, що телефони з більшою кількістю оперативної пам’яті можуть бути доступні за більш високими цінами. Це пов'язано з тим, що оперативна пам'ять є важливим компонентом телефону, який впливає на його продуктивність. Більше оперативної пам'яті дозволяє телефону запускати більше програм одночасно, обробляти більше даних та працювати швидше. Тому виробники телефонів, як правило, встановлюють більше оперативної пам'яті у свої флагманські моделі, які також мають й вищу ціну.

Крім того, сильні кореляції між деякими іншими змінними, такими як **pc** та **fc**, **px_heigh** та **px_widht**, **sc_w** та **sc_h**, а також **tree_g** та **four_g**, вказують на те, що ці пари характеристик зазвичай зростають або зменшуються разом. Інші змінні мають слабку кореляцію між собою, що свідчить про незалежність їх впливу на досліджувані показники.

2.3 Висновки

В цьому розділі було здійснено підготовку даних та проведено розвідувальний аналіз даних, побудовано теплову карту та з її допомогою виявлено фактори, які найбільше впливають на збільшення ціни мобільних телефонів, а саме: ram (оперативна пам'ять), pc та fc (основна камера та фронтальна камера), px_heigh та px_widht (висота роздільної здатності та ширина роздільної здатності), sc_w та sc_h(ширина екрану та висота екрану), а також tree_g та four_g (наявність або ж відсутність 3 та 4G технологій).

3 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ ПЕРЕДБАЧЕННЯ ЦІНИ ТЕЛЕФОНІВ

3.1 Розробка математичних моделей

Опишемо моделі класифікаціїі, які було використано під час написання дипломної роботи: Random Forest, XGBoost, AdaBoost, ExtraTree. Спочатку було представлено алгоритм роботи кожної з моделей.

Random Forest – це ансамблевий метод, який використовує велику кількість дерев рішень для покращення точності і зниження ризику перенавчання. Наведемо математичне пояснення основних кроків алгоритму [20]:

1. Бутстрепінг (Bootstrap Sampling):

– Нехай D – початковий набір даних, що містить n зразків. Random Forest B бутстреп-відібраних наборів даних $D_1, D_2, \ldots, D_B$, кожен з яких містить n зразків, відібраних з D з поверненням. Це означає, що деякі зразки можуть з'являтися кілька разів у одному бутстреп-відібраному наборі, а деякі можуть бути не включені.

2. Побудова дерев рішень:

– Для кожного бутстреп-відібраного набору даних D_i будується дерево рішень T_i. Під час побудови дерева, на кожному вузлі, замість перевірки всіх можливих ознак, випадковим чином обирається підмножина ознак m з усіх p ознак, де $m \leq p$. Найкраща ознака для розділення вибирається з цієї підмножини.

– Нехай $X = \{x_1, x_1, \ldots, x_p\}$ – набір ознак. На кожному вузлі випадковим чином обирається підмножина $M \subset X$, де $|M| = m$.

3. Ріст дерев:

– Кожне дерево рішень росте без обрізання, поки кожен лист не стане чистим (всі зразки належать до одного класу) або не досягне мінімального розміру вузла.

4. Агрегація результатів:

– Для задачі класифікації: кожне дерево T_i робить своє передбачення класу для зразка x. Остаточне передбачення y визначається більшістю голосів:

$$y = \text{mode}\{ T_i(x) : i = 1, 2, ..., B\}$$

– Для задачі регресії: кожне дерево T_i передбачає значення y для x. Остаточне передбачення y визначається як середнє значення передбачень всіх дерев:

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

На рисунку 3.1 показано блок-схему методу Random Forest

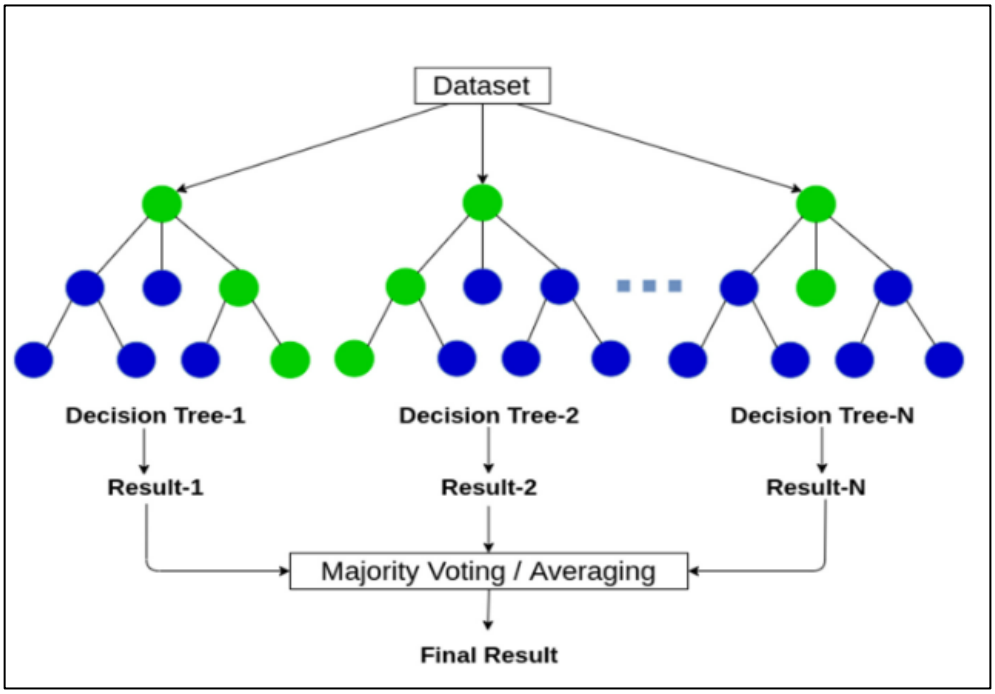


Рисунок 3.1 – Блок-схема методу Random Forest

XGBoost (Extreme Gradient Boosting) – це покращений алгоритм градієнтного бустингу, який є дуже ефективним для задач класифікації та регресії. Основні кроки алгоритму включають [21]:

1. Ініціалізація моделі:

– Початкова модель $F_0(x)$ встановлюється як константа, зазвичай обирається середнє значення цільової змінної (у випадку регресії) або логарифмічні шанси (у випадку класифікації).

2. Послідовне додавання дерев:

– Алгоритм додає нові дерева послідовно для покращення точності моделі. На кожній ітерації t, додається $h_t(x)$, яке мінімізує функцію втрат.

3. Обчислення залишків:

– На кожній ітерації обчислюються залишки (residuals) або псевдо-резідуали для поточної моделі $F_{t-1}(x)$. Псевдо-резідуали $r_i^{(t)}$ визначаються як негативний градієнт функції втрат L на поточних передбаченнях:

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

4. Побудова дерева:

– Нове дерево $h_t(x)$ будується для передбачення залишків $r_i^{(t)}$. Це дерево мінімізує функцію втрат для резідуалів. Математично, це означає, що будується нова модель $h_t(x)$, яка наближає градієнт втрат:

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

5. Оновлення моделі:

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

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

6. Регуляризація:

– Для запобігання перенавчанню, XgBoost використовує регуляризацію, яка включає L1 та L2 регуляризацію на ваги листів дерев. Регуляризаційний термін додається до функції втрат:

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

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

Блок-схема методу XGBoost зображено на рисунку 3.2

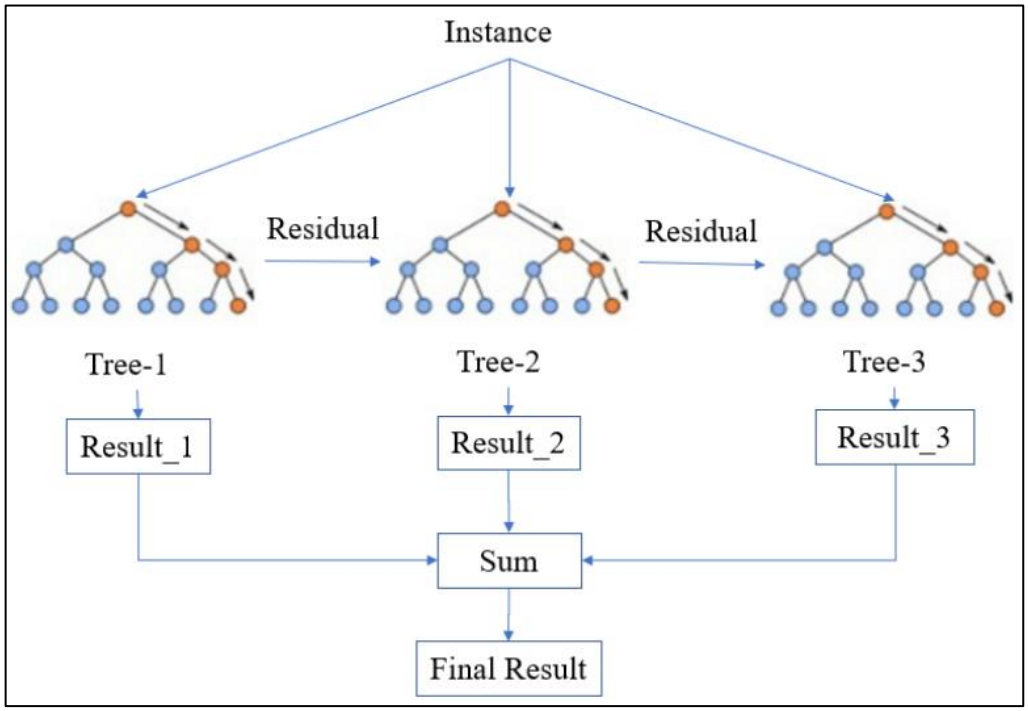


Рисунок 3.2 – Блок-схема методу XGBoost

Математична модель класифікації AdaBoost (Adaptive Boosting) є однією з популярних методів ансамблевого навчання, який використовує комбінацію простих (базових) моделей для створення сильної моделі. Основна ідея AdaBoost полягає в тому, щоб ітеративно навчати базові моделі, приділяючи більше уваги тим прикладам, які важче класифікувати.

Математична модель класифікації AdaBoost виглядає так [21]:

1. Ініціалізація ваг:

Кожному навчально прикладу присвоюється вага ω_i. На початку всі ваги рівні і здаються як $\omega_i = \frac{1}{N}$, де N – кількість навчальних прикладів.

2. Ітеративне навчання базових моделей:

Для кожної ітерації $t = 1, 2, \ldots, T$ (де T – кількість базових моделей):

– Навчання базового класифікатора:

Навчаємо базовий класифікатор h_t на навчальному наборі з вагами ω_i.

– Обчислення похибки класифікатора:

Вивчаємо похибку базового класифікатора h_t:

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

де $I\big(y_i \neq h_t(x_i)\big)$ – індикаторна функція, яка дорівнює 1, якщо класифікатор помилився на прикладі i, та 0 в іншому випадку.

– Обчислення ваги класифікатора:

Вага α_t базового класифікатора обчислюється як:

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

– Оновлення ваг навчальних прикладів:

Ваги оновлюються для кожного прикладу i наступним чином:

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

Після чого ваги нормалізуються так, щоб їх сума дорівнювала 1:

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

3. Формування кінцевої моделі:

Кінцевий класифікатор є комбінацією всіх базових класифікаторів:

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

Ця модель використовує вагові коефіцієнти α_t щоб підсилити внесок тих базових моделей, які добре класифікують важкі приклади, і зменшити внесок тих, що часто помиляються. У результаті отримується потужний ансамблевий класифікатор, здатний давати кращі результати, ніж окремі базові моделі.

Блок-схему методу показано на рисунку 3.3

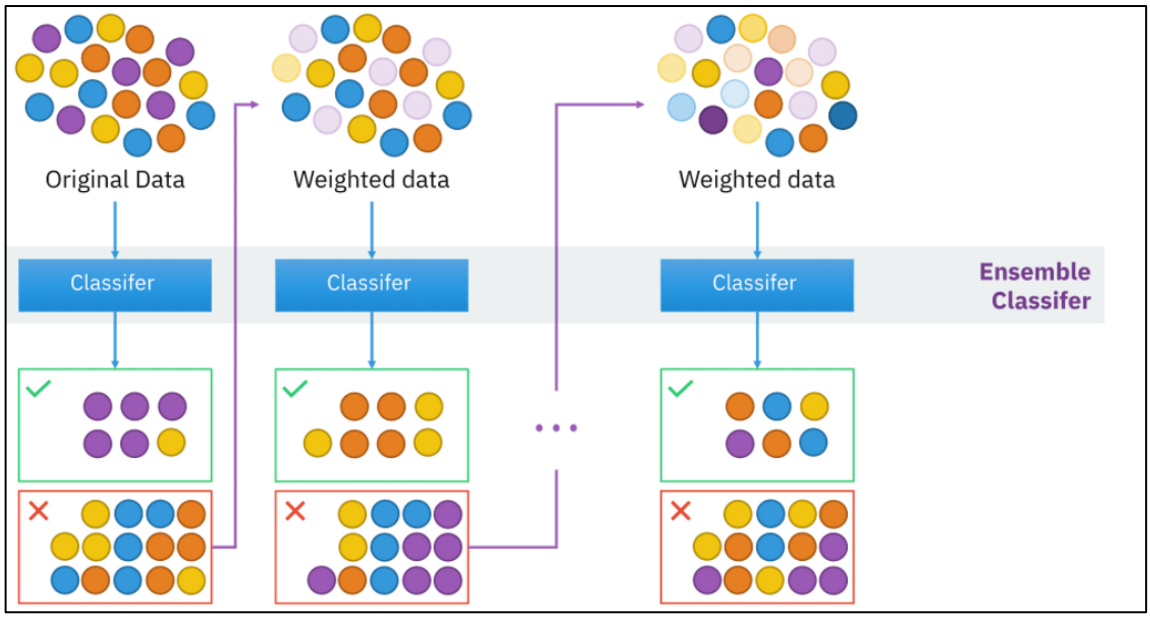


Рисунок 3.3 – Блок-схема методу AdaBoost

ExtraTrees (Extremely Randomized Trees) є методом ансамблевого навчання, який використовується для класифікації та регресії. Це різновид методу Random Forest, але з деякими відмінностями в способі побудови дерев. Основна ідея полягає в тому, щоб зробити побудову дерев максимально випадковою для зменшення розбіжності моделі.

Представлення математичної моделі виглядає наступним чином [20]:

1. Формування ансамблю дерев:

Ансамбль складається з M дерев рішень, де M – задана кількість дерев.

2. Випадковий вибір ознак:

На кожному вузлі дерева випадковим чином обирається підмножина ознак (характеристик) з початкового набору ознак. Кількість вибраних ознак зазвичай дорівнює $\sqrt{d}$, де d – загальна кількість ознак.

3. Випадковий вибір порогів:

Для кожної вибраної ознаки обирається кілька порогів розділення випадковим чином з можливих значень ознаки.

4. Розділення вузла:

Вибирається найкращий поріг з випадкових порогів для кожної ознаки, який максимізує критерій інформаційного приросту (наприклад, приріст інформації або індекс Джині). Вузол розділяється на два підвузли на основі цього порогу.

5. Побудова дерева:

Процес повторюється для кожного вузла, доки не досягнеться одне з умов зупинки:

– Досягнута максимальна глибина дерева;

– Кількість зразків у вузлі менша за мінімальний розмір вузла;

– Всі зразки у вузлі мають одну і ту ж мітку класу.

6. Класифікація з використанням ансамблю:

Для класифікації зразка x:

– Кожне дерево в ансамблі пророблює свій прогноз $h_m(x)$, де m – індекс дерева.

– Остаточний прогноз ансамблю є більшістю голосів від усіх дерев:

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

де I – індикаторна функція, яка дорівнює 1, якщо $h_m(x) = y$, і 0 в іншому випадку.

ExtraTrees відрізняється від Random Forest двома основними моментами:

– Вузли розділяються з використанням випадкових порогів;

– Використовуються всі зразки даних для навчання кожного дерева, тоді як у Random Forest використовується бутстрепінг (випадкове підмножина зразків з поверненням).

Це надає моделі ExtraTrees вищий рівень випадковості, що може призвести до зменшення розбіжності (variance) моделі, хоча може трохи збільшити зміщення (bias).

Блок-схема методу відображена на рисунку 3.4

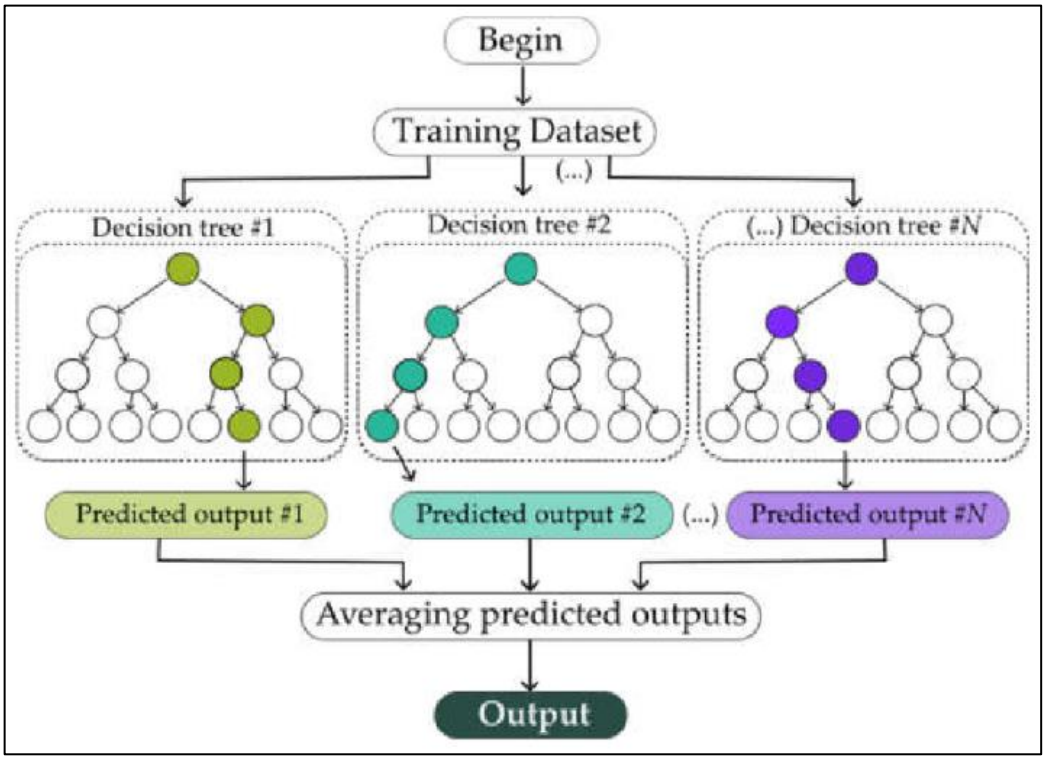


Рисунок 3.4 – Блок-схема методу Extra Tree

3.2 Реалізація математичних моделей

Для початку моделювання та тестування виконаємо попередню обробку даних, включаючи видалення непотрібних стовпців, кодування категоріальних змінних, масштабування числових ознак і розділення даних на тренувальний і тестовий набори для подальшого використання в моделюванні машинного навчання (рис 3.5).

```
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 3.5 – Обробка даних для моделювання та тестування

Після обробки даних було виконано прогнозування моделей та побудовано матриці невідповідностей.

Проведемо прогнозування за допомогою Random Forest Classifier (рис 3.6).

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

Рисунок 3.6 – Код для створення та тюнінгу моделі

Метрики моделі на тренувальних та тестових даних зображено на рисунку 3.7.

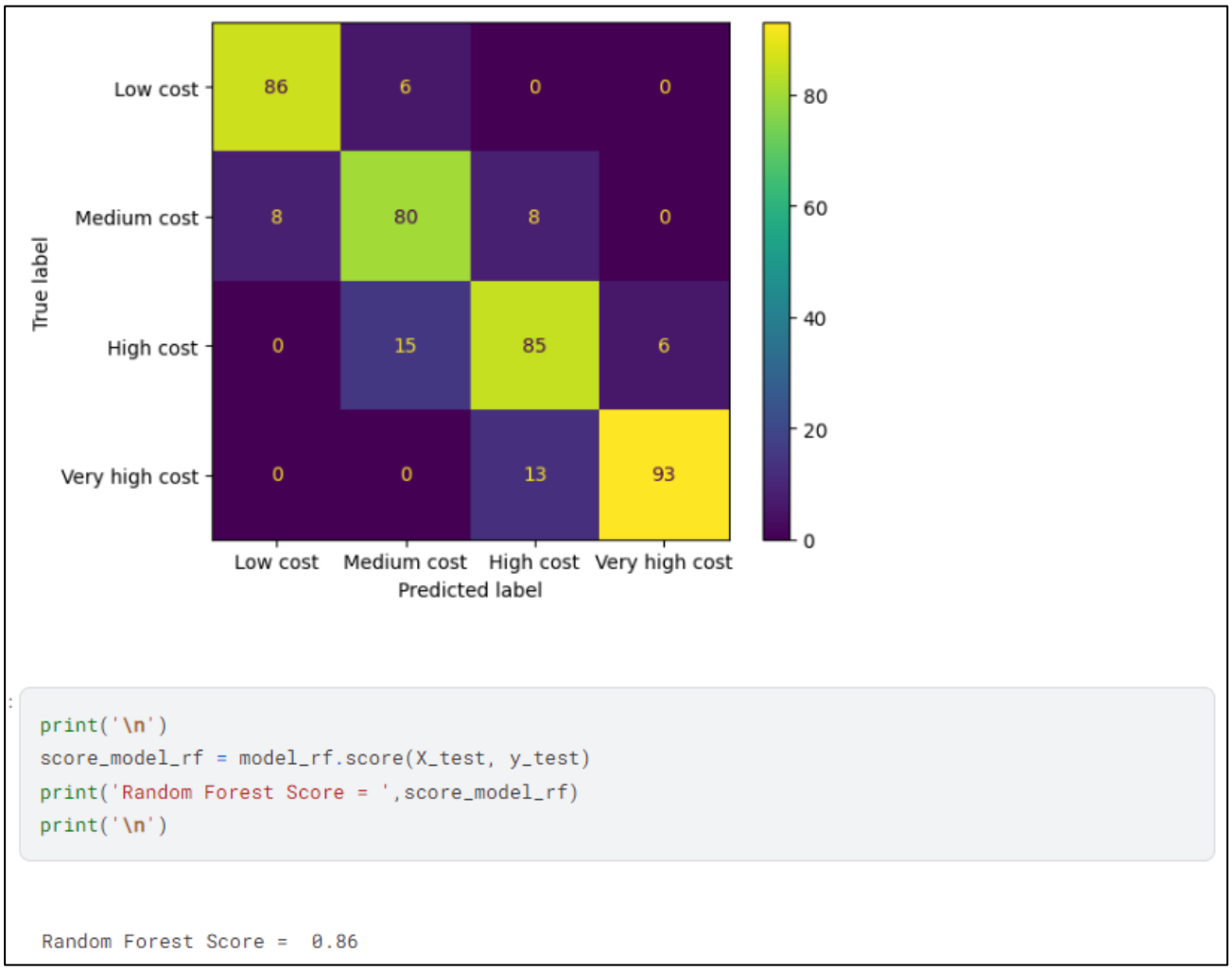
```
Train Data Metrics:
Precision :  1.0
Recall :  1.0
Accuracy :  1.0
F1 Score :  1.0
R2 Score :  1.0


Test Data Metrics:
Precision :  0.86
Recall :  0.86
Accuracy :  0.86
F1 Score :  0.8599999999999999
R2 Score :  0.8867107685460541
```

Рисунок 3.7 – Метрики моделі для тренувальних та тестових даних

Результат роботи випадкового лісу показує, що точність моделі становить 0.86%. Це свідчить про те, що модель правильно класифікувала приблизно 86% випадків у тестовому наборі даних.

Матриця невідповідностей надає більш детальну інформацію про те, як модель працює, показуючи кількість правильних і неправильних прогнозів для кожного класу (рис 3.8).




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

```
Random Forest Score =  0.86
```

Рисунок 3.8 – Матриця плутанини та результат для моделі Random Forest Classifier

Проведено налаштування параметрів моделі XGBoost Classifier, а також прогнозування за допомогою цієї моделі (рис 3.9).

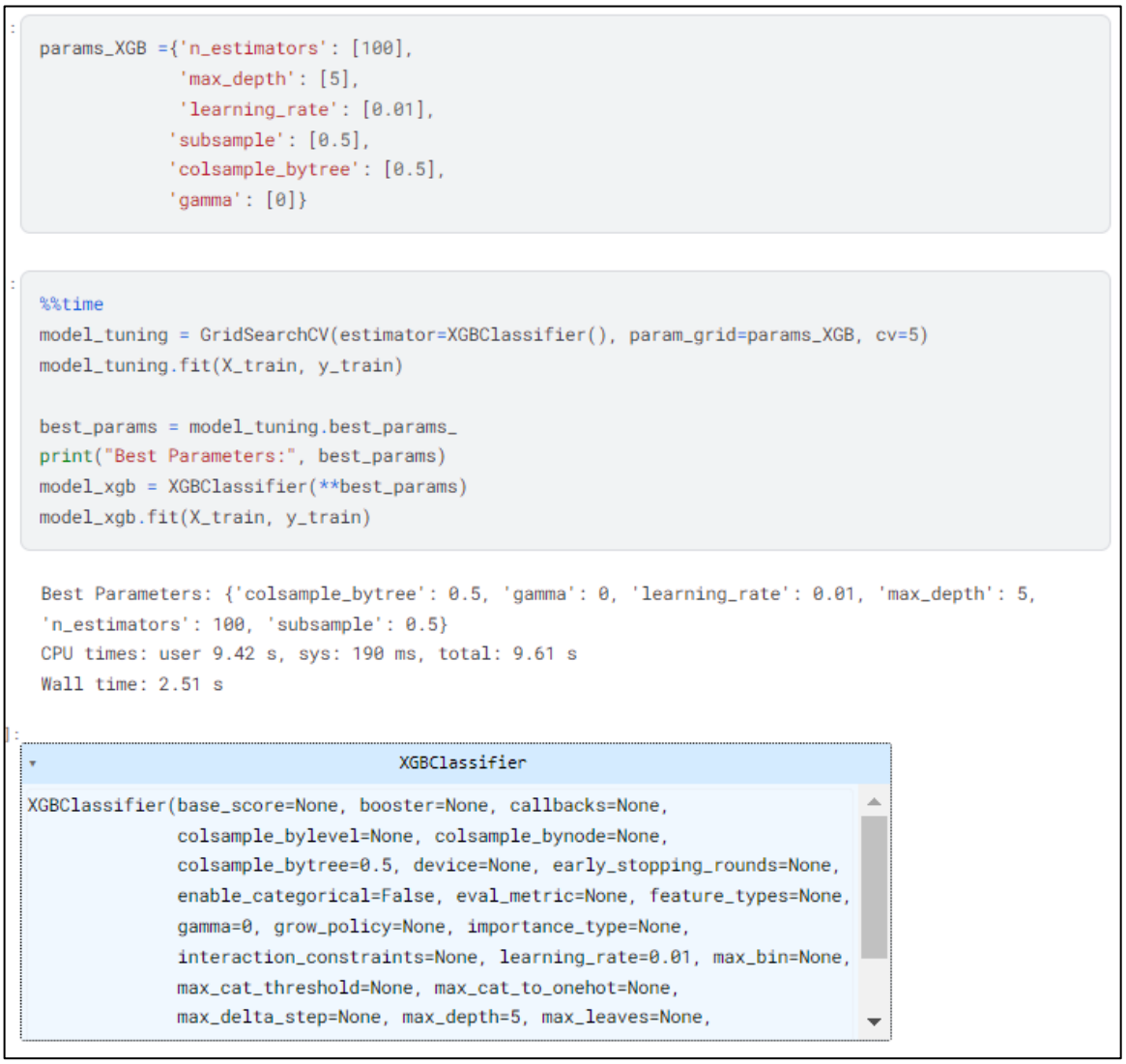


Рисунок 3.9 – Код для створення та тюнінгу моделі

Метрики моделі на тренувальних та тестових даних показано на рисунку 3.10.

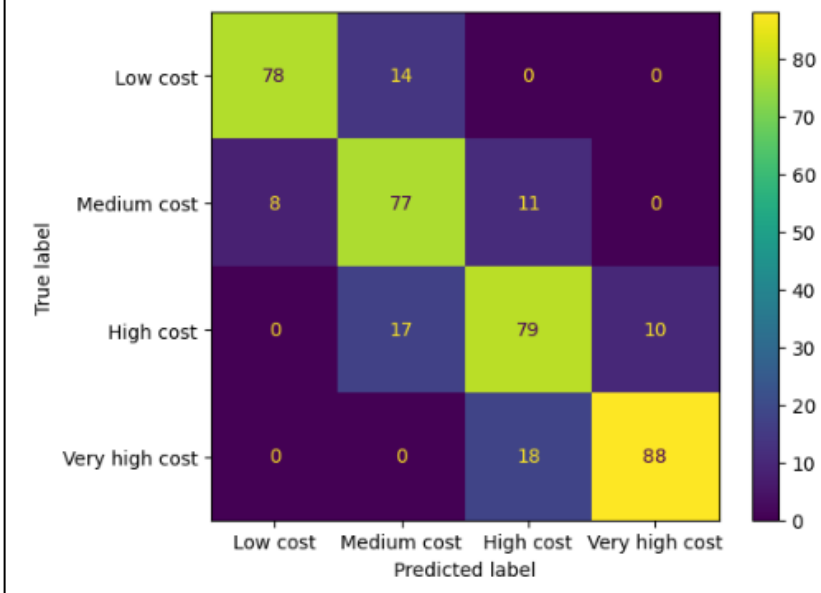
```
Train Data Metrics:
Precision :  0.924375
Recall :  0.924375
Accuracy :  0.924375
F1 Score :  0.924375
R2 Score :  0.9396080261432362


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

Рисунок 3.10 – Метрики моделі для тренувальних та тестових даних

Результат роботи моделі показує, що точність становить 0.805%.

Побудовано матрицю невідповідностей, яка надає більш детальну інформацію про те, як модель працює (рис 3.11).




```
print('\n')
score_model_xgb = model_xgb.score(X_test, y_test)
print('XGBoost Score = ',score_model_xgb)
print('\n')
```

```
XGBoost Score =  0.805
```

Рисунок 3.11 – Матриця плутанини та результат для моделі XGBoost Classifier

Використано модель ADABoost Classifier з налаштованими параметрами. Проведено пошук найкращих параметрів для моделі шляхом перебору вказаних значень параметрів (n_estimators, learning_rate, algorithm) (рис 3.12).

```
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 3.12 – Код для створення та тюнінгу моделі

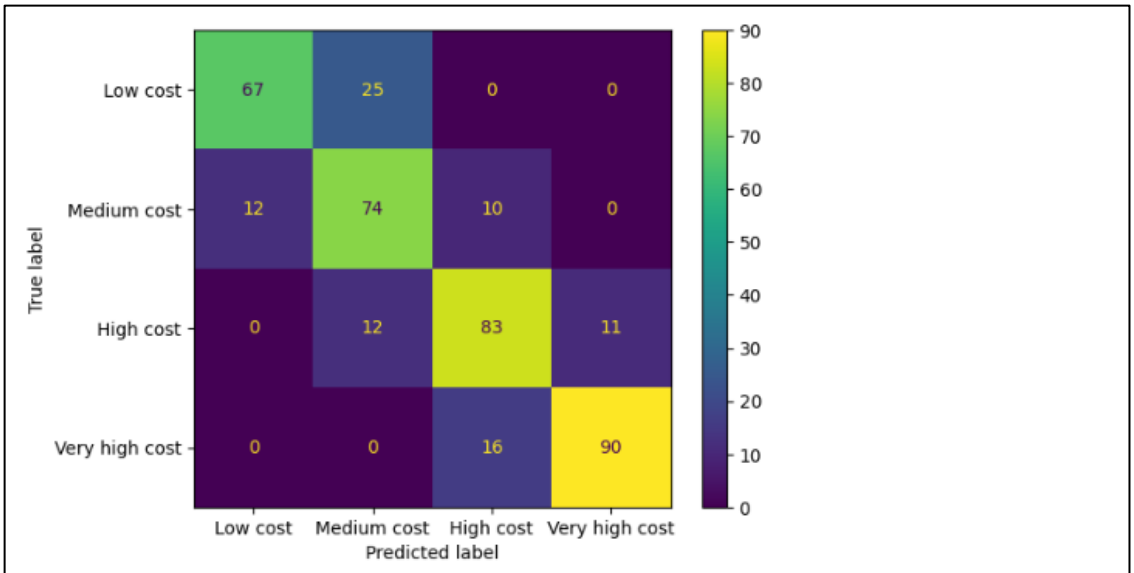
Метрики моделі на тренувальних та тестових даних продемонстровано на рисунку 3.13.

```
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 3.13 – Метрики моделі для тренувальних та тестових даних

Матрицю плутанини для моделі AdaBoost Classifier зображено на рисунку 3.14.




```
print('\n')
score_model_ada = model_ada.score(X_test, y_test)
print('Model ADA Score = ',score_model_ada)
print('\n')
```

```
Model ADA Score =  0.785
```

Рисунок 3.14 – Матриця плутанини та результат для моделі ADABoost Classifier

Використано модель ExtraTreesClassifier для класифікації тестових даих та проведено пошук найкращих параметрів. Після пошуку та встановлення найкращих параметрів виводиться відповідний результат (рис. 3.15).

```
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)
```

```
Best Parameters: {'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_
split': 5, 'n_estimators': 100}
CPU times: user 1.76 s, sys: 14.9 ms, total: 1.77 s
Wall time: 1.77 s
```

```
ExtraTreesClassifier
ExtraTreesClassifier(max_depth=10, min_samples_leaf=2, min_samples_split=5)
```

Рисунок 3.15 – Код для створення та тюнінгу моделі

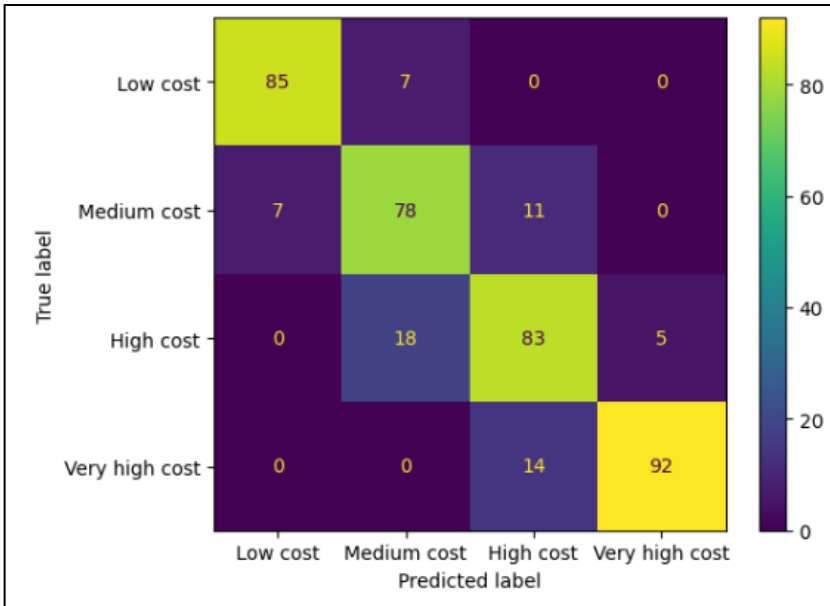
Метрики моделі на тренувальних та тестових даних показано на рисунку 3.16.

```
Train Data Metrics:
Precision :  1.0
Recall :  1.0
Accuracy :  1.0
F1 Score :  1.0
R2 Score :  1.0


Test Data Metrics:
Precision :  0.8275
Recall :  0.8275
Accuracy :  0.8275
F1 Score :  0.8275
R2 Score :  0.8604114826728166
```

Рисунок 3.16 – Метрики моделі для тренувальних та тестових даних

Матрицю плутанини для моделі ExtraTreesClassifier показано на рисунку 3.17.




```
print('\n')
score_model_etc = model_etc.score(X_test, y_test)
print('Extra trees Score = ',score_model_etc)
print('\n')
```

```
Extra trees Score =  0.845
```

Рисунок 3.17 – Матриця плутанини та результат для моделі ExtraTreesClassifier

Наступним кроком було проведено аналіз результатів використаних моделей машинного навчання, який показано на рисунку 3.18.

	model	f1_train	f1_test	r2_train	r2_test	short
0	RandomForest Classifier	1.0	0.86	1.0	0.886711	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	0.8275	1.0	0.860411	ExT

Рисунок 3.18 – Результати моделей

На основі даних, представлених у таблиці результатів, можна зробити висновок, що ADABoost Classifier є найкращою моделлю для передбачення ціни мобільних телефонів, інші моделі піддалися явищу перенавчання так як їхня різниця між тренувальними та тестовими даними складає більше як 10%.

3.3 Розроблення інформаційної технології передбачення ціни телефонів

Для розв’язання даного завдання створений алгоритм інформаційної технології з опрацюванням вхідних даних, налаштування і застосування моделі, який містить такі кроки:

1. Аналіз вхідних даних;
2. Візуалізація даних за допомогою графіків для того щоб отримати краще розуміння залежності між ознаками та їх цінами;
3. Обробка вхідних даних та їх перевірка на наявність пропущених значень;
4. Розбиття даних тренувальні й тестові набори;
5. Вибір моделей машинного навчання, які будуть використані для передбачення ціни телефонів;
6. Оптимізація моделей;
7. Оцінка точності моделей на тренувальних даних;
8. Формування прогнозів та графіків.

Блок-схему алгоритму інформаційної технології показано на рисунку (3.19)

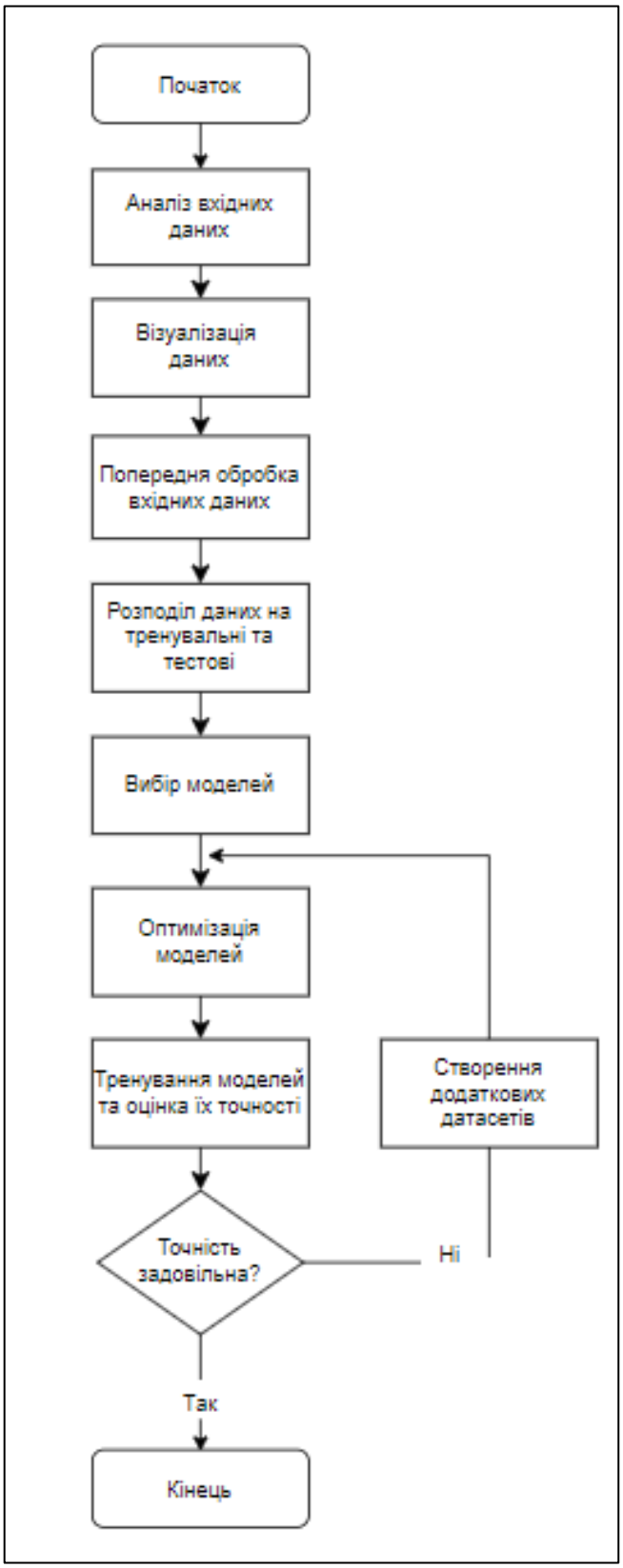


Рисунок 3.19 – Блок-схема алгоритму інформаційної технології передбачення ціни телефонів методами машинного навчання

3.4 Висновки

В даному розділі було розроблено інформаційну технологію для передбачення ціни мобільних телефонів на основі датасету Mobile Price Classifier. Описано розробку математичних моделей, які було використано, продемонстровано їхню реалізацію, а також наведено їхні результати та зроблено висновки, які вказують на те, що найкращою моделлю для передбачення ціни телефонів є AdaBoost Classifier з точністю передбачення 0.785 на тестових даних.

ВИСНОВКИ

В ході виконання бакалаврської дипломної роботи було проведено дослідження та прогнозування факторів ціни мобільних телефонів за допомогою методів машинного навчання.

Проведено аналіз об'єкта дослідження та аналіз методів прогнозування. Для ефективного прогнозування факторів цін на телефони необхідно використовувати методи машинного навчання, які можуть враховувати різні фактори та залежності.

Також було розглянуто вибір оптимальної інформаційної технології для прогнозування рішення та налаштувань для розв'язання поставленої задачі. Обрано мову програмування Python, оскільки вона надає широкі можливості для реалізації методів машинного навчання. Проведено розвідувальний аналіз даних, побудована кореляційна матриця, яка дозволила отримати уявлення про характеристики набору даних та їх взаємозв'язки.

Здійснено застосування моделей та прогнозування цін мобільних пристроїв. Було проведено моделювання, використовуючи різні методи машинного навчання. В результаті прогнозування модель AdaBoost Classifier показала найкращий результат з точністю 0.7606 на тренувальних даних та 0.785 на тестових, тому її доцільно використовувати для прогнозування факторів цін мобільних телефонів.

В цілому, результати дослідження з прогнозування цін на мобільні пристрої з використанням методів машинного навчання демонструють ефективність такого підходу. Отримана модель може бути використана для прогнозування цін на мобільні пристрої в майбутньому, що дозволить приймати більш обгрунтовані рішення в області електронної комерції.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Predicting the price of phones using machine learning methods. [Електронний ресурс]. URL: https://galinfo.com.ua/news/vartist_smartfonu_shcho_vplyvaie_na_formuvannya_tsiny_387977.html#google_vignette

2. Random forest in remote sensing: A review of applications and future directions [Електронний ресурс]. URL: Random forest in remote sensing: A review of applications and future directions - ScienceDirect

3. XGBoost Documentation, 2022. [Електронний ресурс]. URL: https://xgboost.readthedocs.io/en/stable/

4. AdaBoost Classifier in Python, 2022. [Електронний ресурс]. URL: https://www.datacamp.com/tutorial/adaboost-classifier-python

5. Extra Tree Scikit-learn, 2022. [Електронний ресурс]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier

6. 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

7. K-Nearest Neighbors (KNN) in Python, 2022. [Електронний ресурс]. URL: https://www.ibm.com/topics/knn

8. Decision Trees in Python, 2020. [Електронний ресурс]. URL: https://www.ibm.com/topics/decision-trees

9. lightgbm.LGBMClassifier, 2023. [Електронний ресурс]. URL: https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html

10. Gradient Boosting Scikit-learn, 2022. [Електронний ресурс]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

11. Machine learning, 2020. [Електронний ресурс]. URL: https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML

12. Amit Kumar Bishnoi, Sanjeev Kumar Mandal. Perfomance Analysis for

Mobile Price Prediction, 2023. [Електронний ресурс]. URL: https://ieeexplore.ieee.org/abstract/document/10466198

13. Danda B. Rawat, Lait K Awasthi, Valentina Emilia Balas. Comparison of Various Classification Model Using Machine Learning to Predict Mobile Phones Price Range, 2023. [Електронний ресурс]. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119905233.ch17

14. Agust Pratondo, Rio Korio Utoro, Toufan D. Tambunan, Aris Budianto. Predict of Operating System Preferences on Mobile Phone Using Machine Learning, 2023. [Електронний ресурс]. URL: https://ieeexplore.ieee.org/abstract/document/10335385

15. Бібліотека NumPy, 2020. [Електронний ресурс]. URL: https://www.w3schools.com/python/numpy/

16. Exploratory Data Analysis in Python, 2022. [Електронний ресурс]. URL: https://towardsdatascience.com/exploratory-data-analysis-in-python-a-step- by-step-process-d0dfa6bf94ee

17. 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

18. 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

19. Okmi M, Por LY, Ang TF, Ku CS. Mobile Phone Data: A Survey of Techniques, Features, and Applications. *Sensors*. 2023; 23(2):908. DOI: https://doi.org/10.3390/s23020908

20. Мокін О.Б., Мокін В.Б., і Мокін Б.І., «Алгоритм методу ідентифікації моделі авторегресії – ковзного середнього, який узагальнює методику Юла–Уокера, та його програмна python-реалізація», Вісник ВПІ, вип. 4, с. 41–55, Верес. 2022.

21. 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

22. Oleg Nereutskyi. Mobile Price Classifier 3.0. 2024 [Електронний ресурс]. URL: https://www.kaggle.com/code/olegnereutskyi/mobile-price-classifier-5-0

Додаток А

(обов’язковий)

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

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
“____”_____ 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на бакалаврську дипломну роботу

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПЕРЕДБАЧЕННЯ ЦІНИ ТЕЛЕФОНІВ МЕТОДАМИ МАШИННОГО НАВЧАННЯ

08-34.БДР.0011.00.000 ТЗ

Керівник: к.т.н., доц.
_____ Олексій КОЗАЧКО
«__» _____ 2024 р.
Розробив студент гр. 2ІСТ-20б
_____ Олег НЕРЕУЦЬКИЙ
«__» _____ 2024 р.

Вінниця 2024

1. Підстава для проведення робіт.

Підставою для виконання роботи є наказ №__ по ВНТУ від «__» _____2024р., та індивідуальне завдання на БДР, 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:

a) Аналіз предметної області _____ – _____
b) Вибір оптимальних інформаційних технологій _____ – _____
c) Тренування моделей машинного навчання _____ – _____
d) Створення інформаційної технології
e) Прогнозування даних _____ – _____
f) Оформлення матеріалів до захисту БДР _____ – _____

7. Очікувані результати та порядок реалізації

Отримання інформаційної технології передбачення ціни телефонів методами машинного навчання.

8. Вимоги до розробленої документації

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

9. Порядок приймання роботи

Публічний захист «___» _____ 2024 р.
Початок розробки «___» _____ 2024 р.
Граничні терміни виконання БДР «___» _____ 2024 р.

Розробив студент групи 2ІСТ-20б _____ Олег НЕРЕУЦЬКИЙ

Додаток Б

(обов'язковий)

Протокол перевірки бакалаврської дипломної роботи на наявність текстових запозичень

Назва роботи: «Інформаційна технологія передбачення ціни телефонів методами машинного навчання»
Тип роботи: бакалаврська дипломна робота
Підрозділ: кафедра САІТ

Показники звіту подібності Unicheck

Оригінальність 92.8% Схожість 7.2%

Аналіз звіту подібності (відмітити потрібне)

• Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
o Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.
o Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ (підпис) Сергій ЖУКОВ

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ (підпис) Олег НЕРЕУЦЬКИЙ

Керівник роботи _____ (підпис) Олексій КОЗАЧКО

Додаток В

(довідниковий)

Фрагмент лістингу програми

```
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, StratifiedKFol
d, 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, ExtraTree
sClassifier
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_scor
e, 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', 'thre
e_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_val
ues})

# 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_sp
acing=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", "w
hite"])
    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=a
x [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_wi
dth ()/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', edg
ecolor='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=co
lors)
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=c
olors)
plt.show();

numerical = ["battery_power","clock_speed","fc","int_memory","m_dep","mobile_wt","p
c","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_size=
0.2, random_state=1)
```

```
# Results of prediction
results = pd.DataFrame(columns = ['model', 'f1_train', 'f1_test', 'r2_train', 'r2_t
est'])

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)
```

Додаток Г

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ПЕРЕДБАЧЕННЯ ЦІНИ ТЕЛЕФОНІВ МЕТОДАМИ МАШИННОГО НАВЧАННЯ

Нормоконтроль: к.т.н., доцент

_____ Сергій ЖУКОВ

«___» _____ 2024 р.

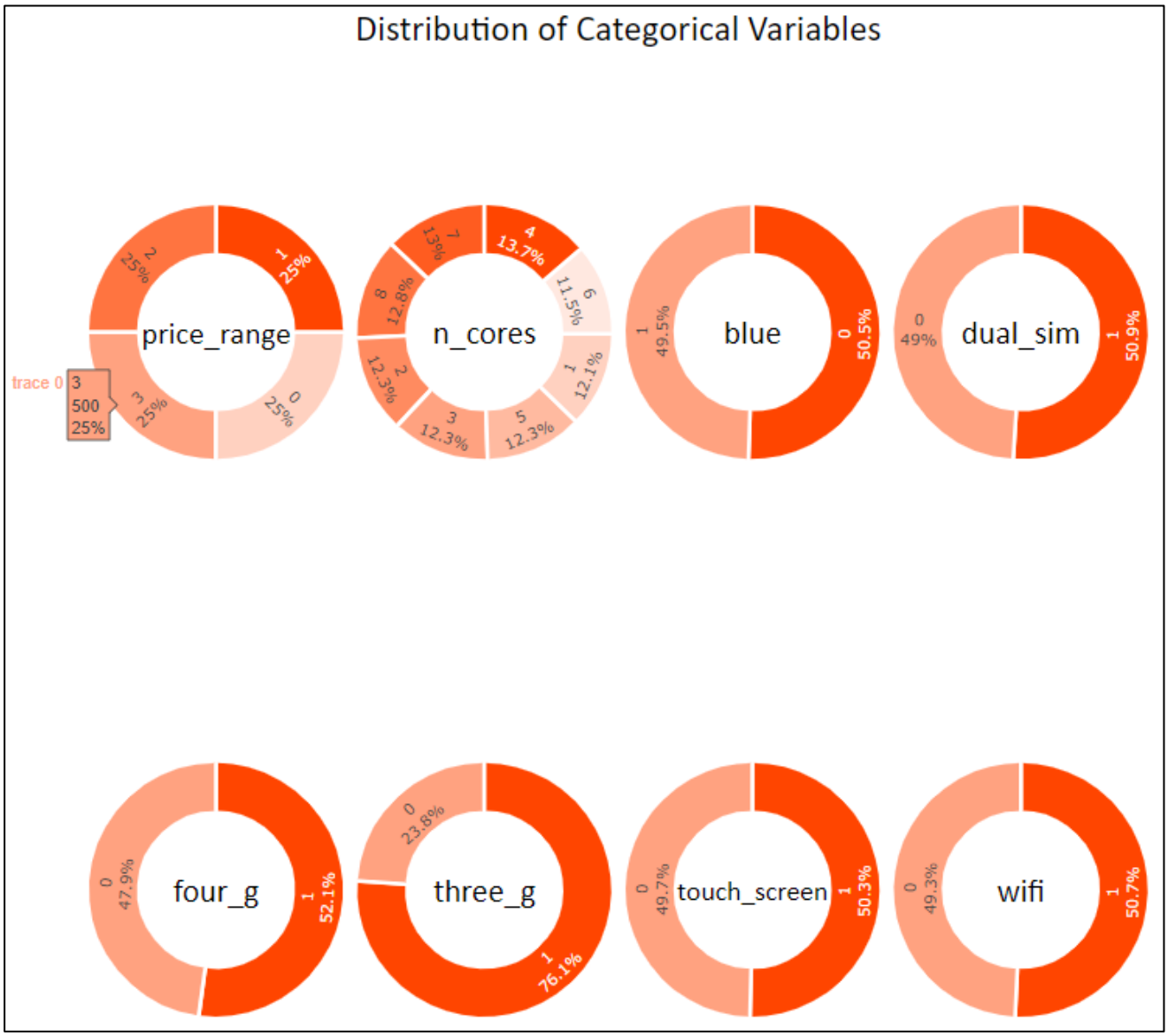


Рисунок Г.1 – Інтерактивні кругові діаграми категоріальних ознак

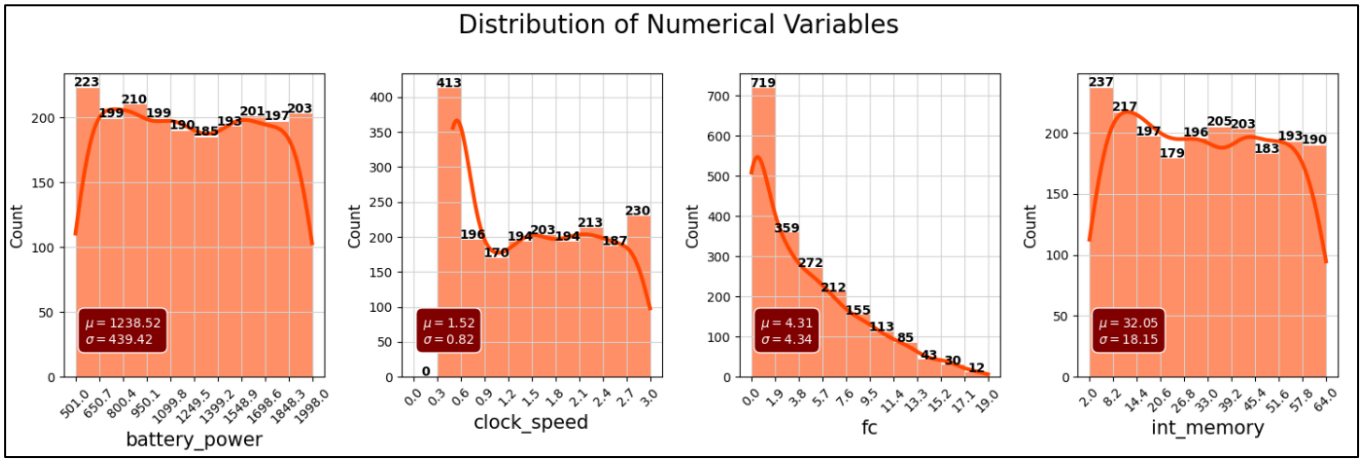


Рисунок Г.2 – Гістограми числових ознак

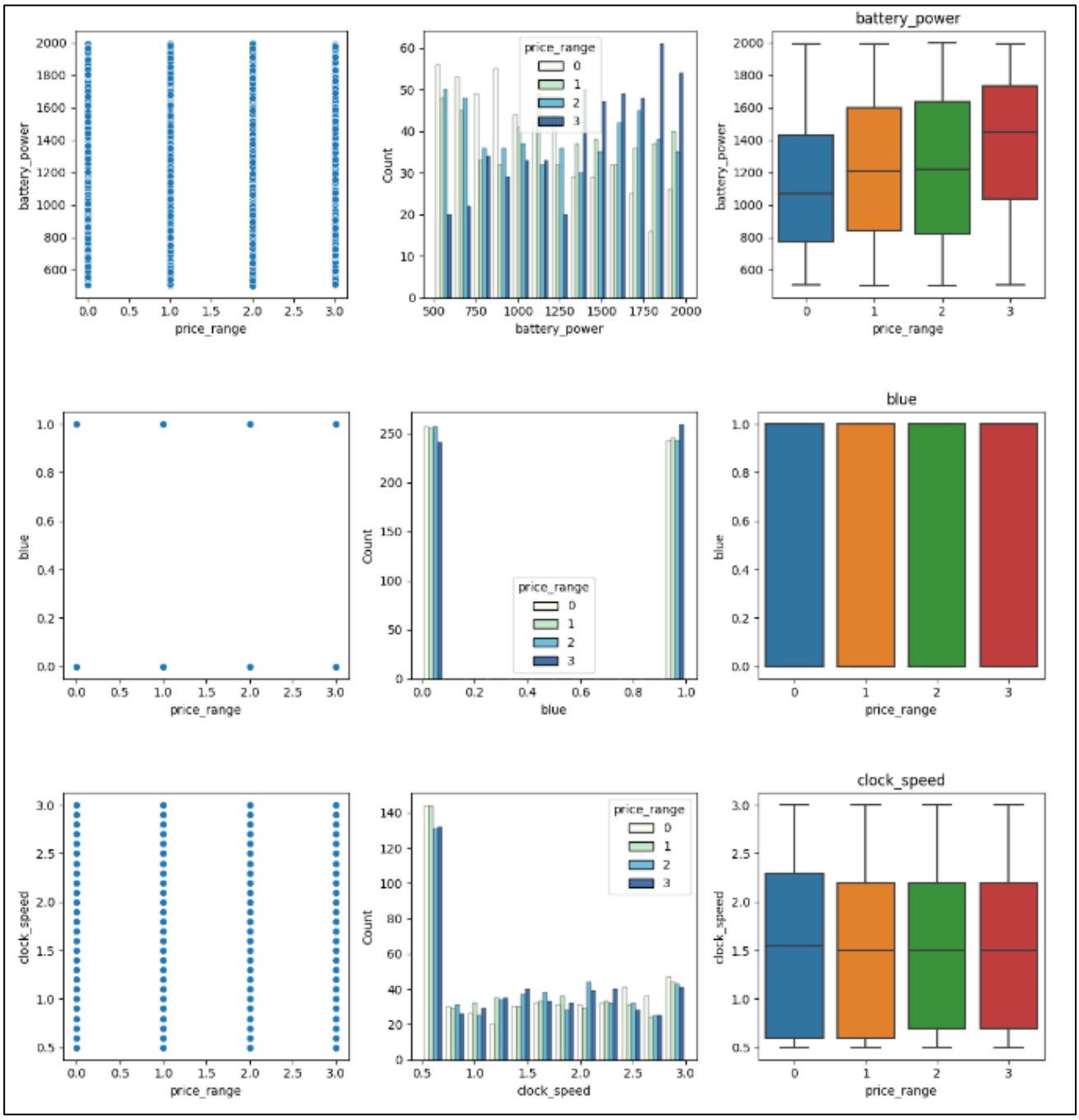


Рисунок Г.3 – Графіки залежностей категоріальних змінних

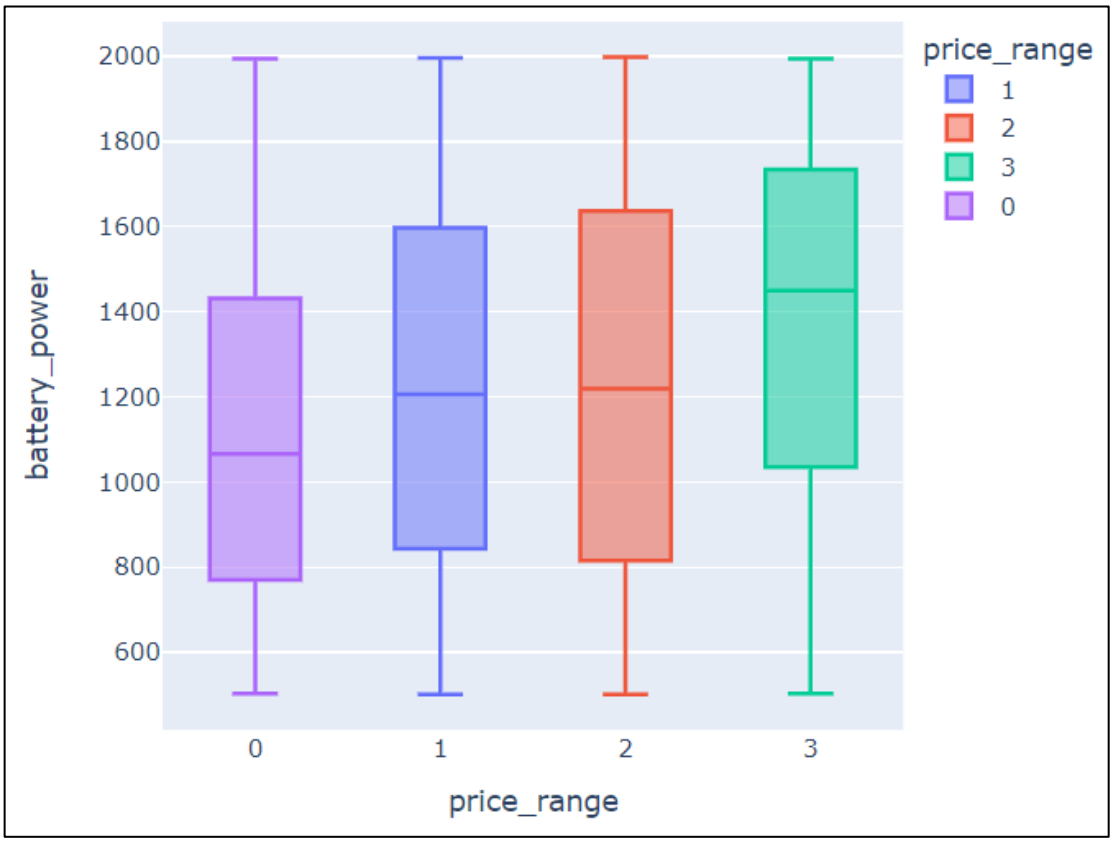


Рисунок Г.4 – Box-plot графік розподілу потужності батареї для різних цінових категорій

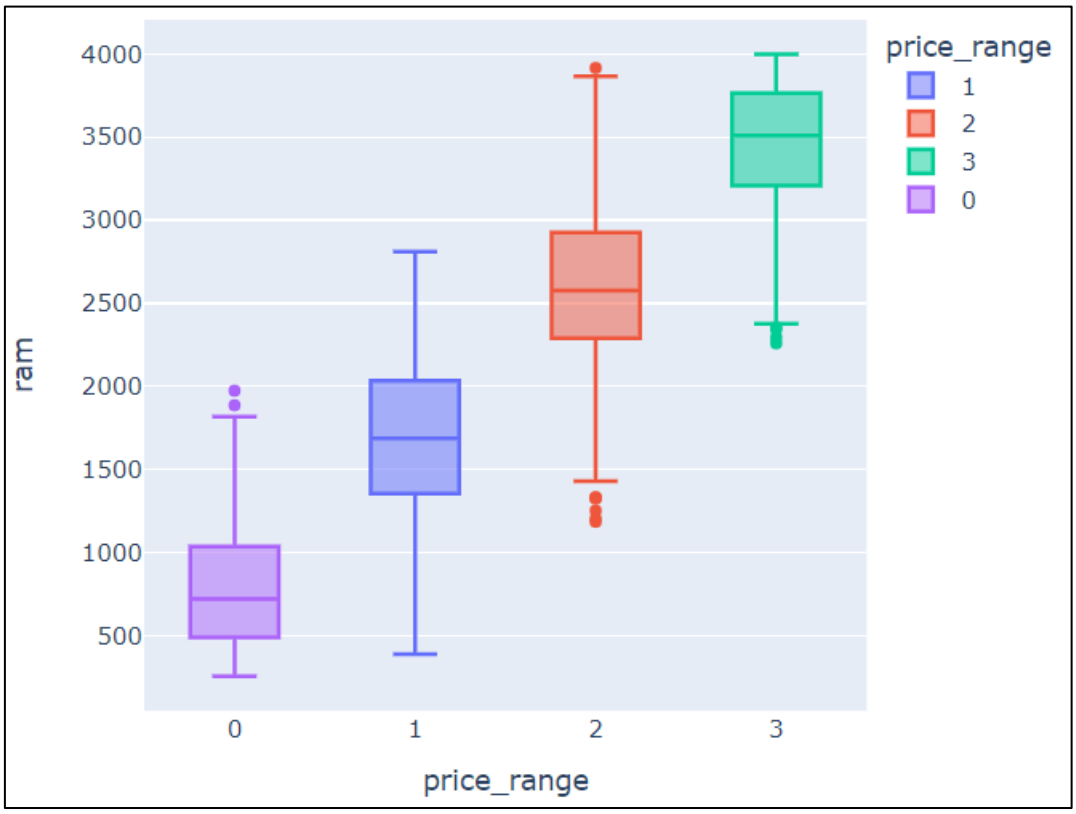


Рисунок Г.5 – Box-plot графік розподілу оперативної пам'яті для різних цінових категорій

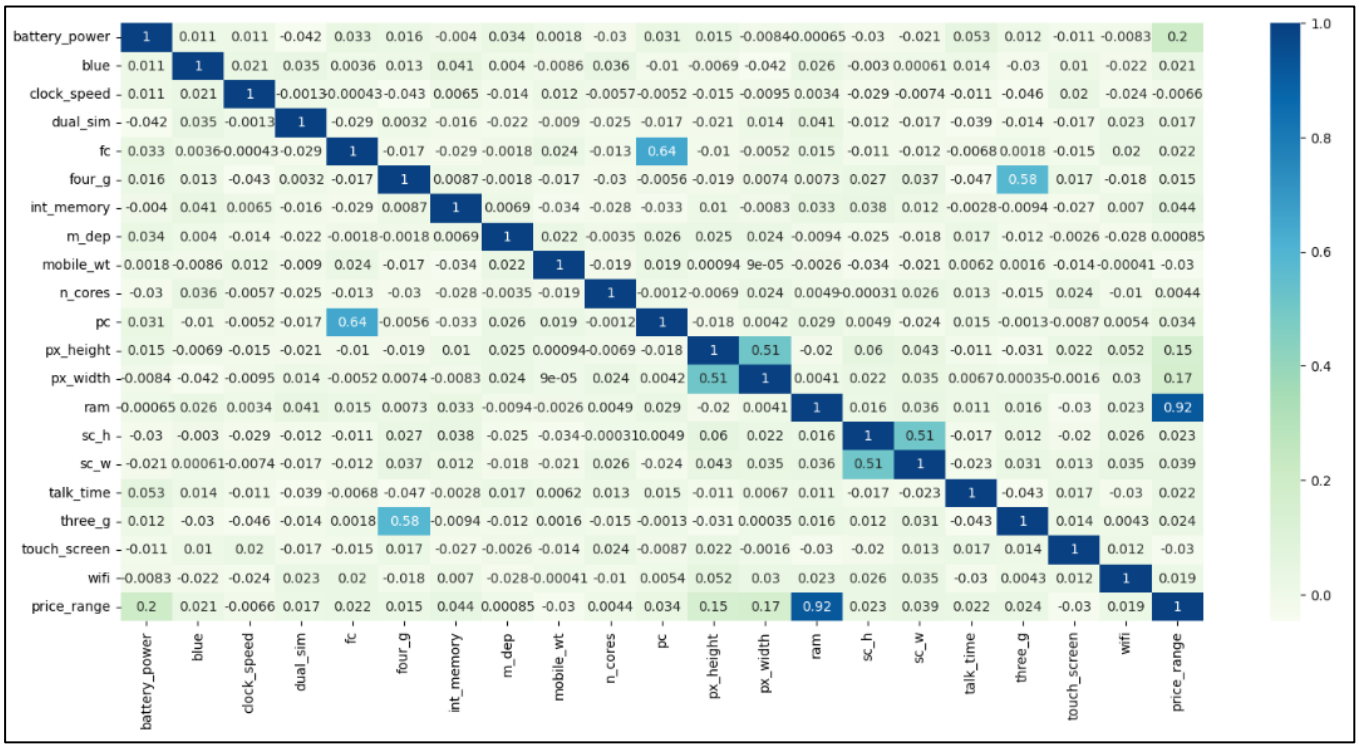
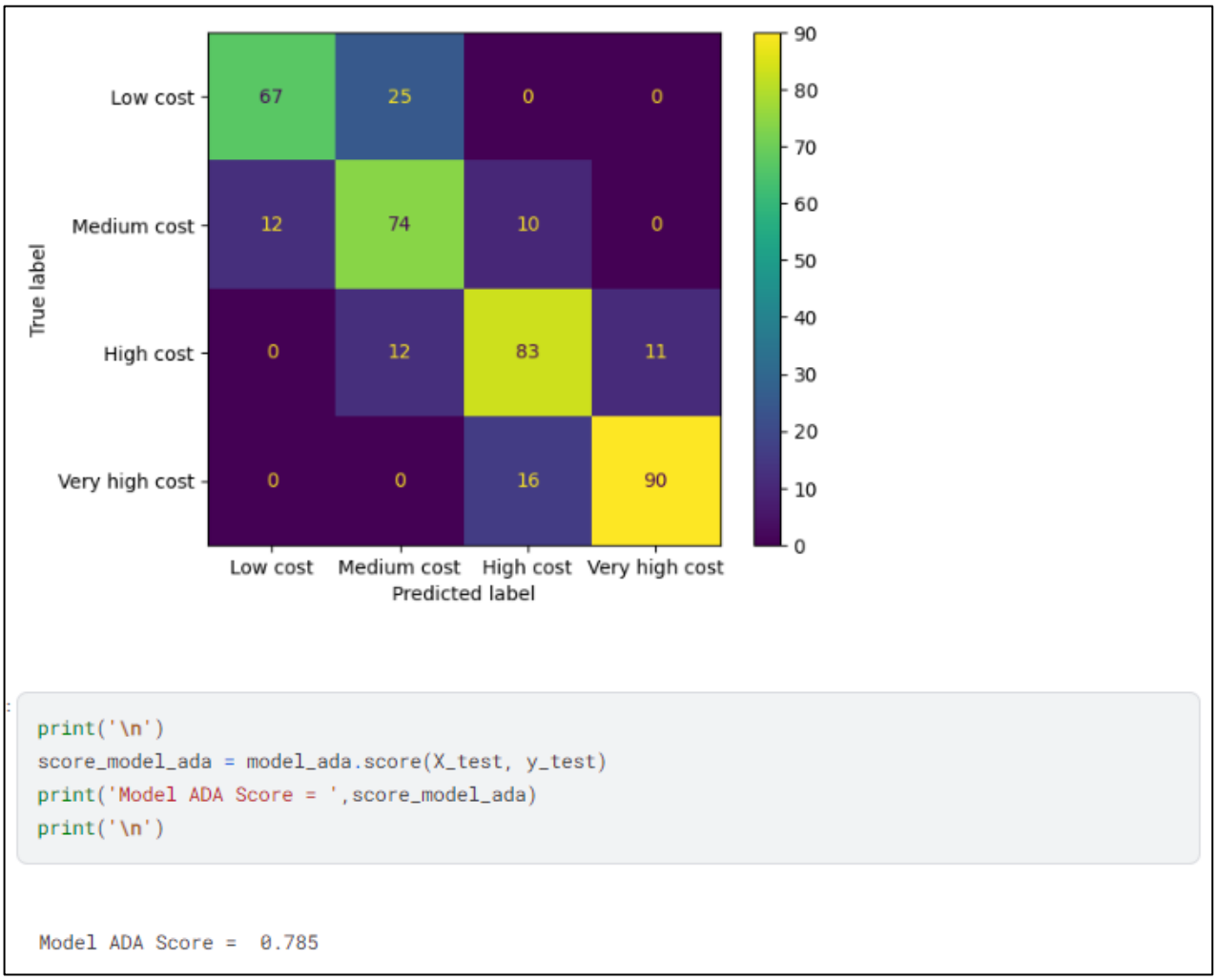


Рисунок Г.6 – Кореляційна матриця

	model	f1_train	f1_test	r2_train	r2_test	short
0	RandomForest Classifier	1.0	0.86	1.0	0.886711	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	0.8275	1.0	0.860411	ExT

Рисунок Г.7 – Результати передбачення моделей




```
print('\n')
score_model_ada = model_ada.score(X_test, y_test)
print('Model ADA Score = ',score_model_ada)
print('\n')
```

```
Model ADA Score =  0.785
```

Рисунок Г.8 – Оптимальна модель AdaBoost для передбачення ціни телефонів

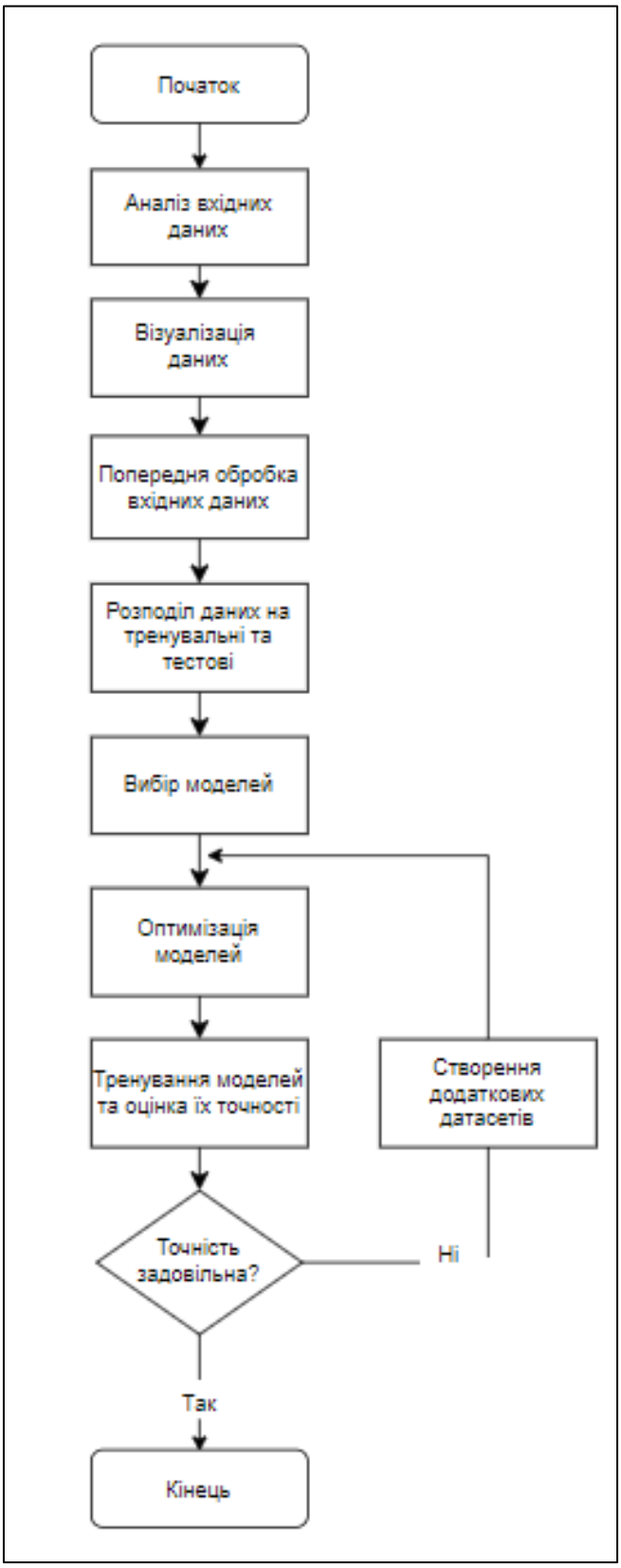


Рисунок Г.9 – Блок-схема алгоритму інформаційної технології