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

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

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Програмний засіб аналіз ринку мобільних телефонів засобами машинного навчання.

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

		Вітушинський А.І.
	(підпис)	(прізвище та ініціали)
Керівник		Небесний Р.М.
	(підпис)	(прізвище та ініціали)
Нормоконтроль		Липак Г.І.
	(підпис)	(прізвище та ініціали)
Завідувач кафедри		Боднарчук І.О.
	(підпис)	(прізвище та ініціали)
Рецензент		Коноваленко І.В.
	(підпис)	(прізвище та ініціали)

Тернопіль
2026

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

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

Кафедра комп'ютерних наук
(повна назва кафедри)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

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

студенту Вітушинський Андрій Іванович
(прізвище, ім'я, по батькові)

1. Тема роботи **Програмний засіб аналіз ринку мобільних телефонів засобами машинного навчання.**

Керівник роботи **PhD., доц. Небесний Руслан Михайлович**
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

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

2. Термін подання студентом завершеної роботи 26 червня 2026 р.

3. Вихідні дані до роботи Літературні джерела з тематики роботи

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

Вступ; 1 Теоретичні засади використаних моделей машинного навчання; 1.1 Метод кластеризації K-середніх; 1.2 Дерево рішень; 1.3 Логістична регресія; 1.4 Випадковий ліс (Random Forest); 1.5 Градієнтний бустинг XGBoost; 1.6 Ансамблеві методи; 1.7 Метрики оцінювання якості класифікації; 1.8 Принципи підготовки даних для машинного навчання; 1.9 Висновки до розділу 1; 2 Програмна реалізація доступу до веб-ресурсу; 2.1 Технологія веб-скрапінгу; 2.2 Бібліотека BeautifulSoup та Selenium; 2.3 Реалізація запиту до веб-ресурсу; 2.4 Структура HTTP-запиту та HTML-розмітки; 2.5 Висновки до розділу 2; 3 Опис роботи програмного коду; 3.1 Проєктування та створення бази даних; 3.2 Збір та парсинг даних; 3.3 Зчитування даних із бази даних; 3.4 Опис структури зібраного набору даних; 3.5 Візуалізація зібраних даних; 3.6 Побудова та навчання моделей машинного навчання; 3.7 Висновки до розділу 3; 4 Безпека життєдіяльності, основи охорони праці; Висновки; Список використаних джерел; Додатки

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

Титульний слайд; Актуальність теми; Мета та завдання дослідження; Об'єкт, предмет і методи дослідження; Конвеєр розпізнавання облич; Інструментальні засоби реалізації; Формування векторних ознак: FaceNet; Архітектура застосунку; Проєктування векторної бази даних; Алгоритм реєстрації особи; Розпізнавання у реальному часі та метрика; Сценарії використання та інтерфейс; Тестування та продуктивність; Висновки.

6. Консультанти розділів роботи

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

7. Дата видачі завдання 26 січня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	26.01.2026 – 27.01.2026	*Виконано*
2.	Підбір джерел по темі роботи	28.01.2026 – 01.04.2026	*Виконано*
3.	Оформлення першого розділу	15.04.2026	*Виконано*
4.	Оформлення другого розділу	20.04.2026	
5.	Оформлення третього розділу	30.04.2026	*Виконано*
6.	Виконання завдання до підрозділу "Безпека		
7.	життєдіяльності, основи охорони праці"	15.05.2026	*Виконано*
8.	Оформлення кваліфікаційної роботи	07.06.2026	*Виконано*
9.	Перевірка на плагіат	07.06.2026	*Виконано*
10.	Нормоконтроль	09.06.2026	*Виконано*
11.	Попередній захист кваліфікаційної роботи	11.06.2026	*Виконано*
12.	Захист кваліфікаційної роботи	27.06.2026	
13.			
14.			
15.			

Студент _____ Вітушинський А.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Небесний Р.М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

"Програмний засіб аналіз ринку мобільних телефонів засобами машинного навчання" // Кваліфікаційна робота освітнього рівня "Бакалавр" // Вітушинський Андрій Іванович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-41 // Тернопіль, 2026 // с. – 61, рис. – 11, таблиць – 10, джерел – 24, додатків – 1, сторінок додатків – 37.

Ключові слова: машинне навчання, аналіз ринку, веб-скрапінг, BeautifulSoup, Random Forest, кластеризація, класифікація, SQLite, Python, смартфони.

Кваліфікаційна робота присвячена розробці програмного засобу для аналізу ринку мобільних телефонів із застосуванням методів машинного навчання. Актуальність теми зумовлена тим, що сучасний ринок смартфонів характеризується значною кількістю моделей, широким діапазоном цін та різноманіттям технічних характеристик, унаслідок чого ручний порівняльний аналіз і обґрунтований вибір пристрою стають складними для пересічного користувача.

У роботі реалізовано повний цикл обробки даних: автоматизований збір інформації про товари з веб-ресурсу за допомогою бібліотеки BeautifulSoup, збереження зібраних даних у реляційній базі даних SQLite, попередня обробка та візуалізація даних засобами бібліотек pandas, matplotlib і seaborn, а також побудова та порівняння кількох класифікаційних моделей машинного навчання – дерева рішень, логістичної регресії, випадкового лісу (Random Forest), градієнтного бустингу XGBoost та ансамблевого голосування.

Окрему увагу приділено обґрунтуванню вибору інструмента доступу до веб-ресурсу: проведено порівняння бібліотеки BeautifulSoup із фреймворком

Selenium за критеріями швидкодії, споживання ресурсів та складності реалізації. За результатами порівняння для поставленої задачі обрано BeautifulSoup.

Найкращу точність класифікації показав алгоритм Random Forest – 97,59 % на реальній вибірці та 79,47 % на імітованій. На основі побудованої моделі сформовано рекомендацію щодо вибору оптимальної моделі смартфона за критерієм максимізації характеристик при мінімізації ціни.

ANNOTATION

"Software Tool for Mobile Phone Market Analysis Using Machine Learning" // Qualification work of the educational level "Bachelor" // Vitushynskyi Andrii // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Computer Science, Group CH-41 // Ternopil, 2026 // p. – 61, fig. – 11, tables – 10, references – 24, annexes – 1, annexes pages – 37.

Keywords: machine learning, market analysis, web scraping, BeautifulSoup, Random Forest, clustering, classification, SQLite, Python, smartphones.

This qualification thesis is devoted to the development of a software tool for analyzing the mobile phone market using machine learning methods. The relevance of the topic stems from the fact that the modern smartphone market is characterized by a large number of models, a wide range of prices, and a diversity of technical specifications, which makes manual comparative analysis and a well-grounded choice of device difficult for the average user.

The work implements a complete data processing pipeline: automated collection of product information from a web resource using the BeautifulSoup library, storage of the collected data in a relational SQLite database, preliminary processing and visualization of the data by means of the pandas, matplotlib, and seaborn libraries, as well as the construction and comparison of several machine learning classification models – a decision tree, logistic regression, random forest (Random Forest), gradient boosting (XGBoost), and ensemble voting.

Particular attention is given to justifying the choice of the tool for accessing the web resource: the BeautifulSoup library is compared with the Selenium framework according to the criteria of performance, resource consumption, and implementation complexity. Based on the results of the comparison, BeautifulSoup was chosen for the task at hand.

The highest classification accuracy was demonstrated by the Random Forest algorithm – 97.59% on the real sample and 79.47% on the simulated one. Based on the constructed model, a recommendation was formulated for choosing the optimal smartphone model according to the criterion of maximizing specifications while minimizing price.

ЗМІСТ

ВСТУП

Ринок мобільних телефонів є одним із найбільш динамічних сегментів сучасної електронної комерції. Щороку виробники випускають десятки нових моделей, які відрізняються за технічними характеристиками, ціновою категорією, операційною системою та позиціонуванням на ринку. У таких умовах перед потенційним покупцем постає нетривіальна задача: обрати пристрій, що забезпечує максимальну користь – найкраще співвідношення сукупності характеристик до ціни.

Складність задачі полягає в тому, що інформація про товари розосереджена між численними інтернет-магазинами, а сам обсяг пропозицій унеможливлює ручний аналіз. Крім того, ціна пристрою формується під впливом багатьох чинників – обсягу пам'яті, роздільної здатності камери, бренду, операційної системи тощо, – взаємозв'язки між якими не завжди очевидні. Це робить актуальним застосування методів машинного навчання, здатних автоматично виявляти закономірності у великих масивах даних.

Машинне навчання надає інструментарій для розв'язання задач класифікації, кластеризації та прогнозування. Поєднання технологій автоматизованого збору даних (веб-скрапінгу) із алгоритмами навчання дозволяє побудувати повноцінну аналітичну систему, яка від зчитування сирих даних із веб-ресурсу доходить до обґрунтованої рекомендації.

Аналіз існуючих рішень.

На ринку існує низка сервісів порівняння товарів (агрегатори цін, маркетплейси з фільтрами характеристик), які надають користувачеві можливість сортувати та фільтрувати пропозиції. Проте більшість із них обмежуються простим відображенням даних і не застосовують методів машинного навчання для виявлення прихованих закономірностей чи формування обґрунтованих рекомендацій. Користувач самостійно має зважувати десятки параметрів, що залишає задачу вибору оптимальної моделі фактично невирішеною.

Наукові та навчальні роботи у сфері аналізу даних зазвичай зосереджуються або на етапі збору даних (веб-скрапінг), або на етапі моделювання, рідко поєднуючи їх у єдиний наскрізний конвеєр. Особливістю цієї роботи є саме інтеграція всіх етапів – від автоматизованого збору сирих даних із реального веб-ресурсу до побудови моделі та формування практичної рекомендації, що робить її цілісним прикладом прикладної системи аналізу даних.

Мета та завдання роботи.

Метою кваліфікаційної роботи є розробка програмного засобу для аналізу ринку мобільних телефонів із застосуванням методів машинного навчання, який автоматизує збір даних, їх обробку та формування рекомендації щодо вибору оптимальної моделі.

Для досягнення поставленої мети сформульовано такі завдання:

1) проаналізувати предметну область і визначити джерело даних про мобільні телефони;

2) дослідити та обрати інструмент автоматизованого доступу до веб-ресурсу, порівнявши бібліотеку BeautifulSoup із фреймворком Selenium;

3) реалізувати модуль збору та збереження даних у базі даних SQLite;

4) виконати попередню обробку та візуалізацію зібраних даних;

5) дослідити теоретичні засади використаних моделей машинного навчання;

6) побудувати та порівняти кілька класифікаційних моделей, обрати найточнішу;

7) сформувати рекомендацію щодо вибору оптимальної моделі смартфона.

Об'єкт і предмет дослідження.

Об'єкт дослідження – процес аналізу ринку мобільних телефонів на основі даних інтернет-магазину.

Предмет дослідження – методи та засоби машинного навчання для класифікації та аналізу моделей смартфонів за їх характеристиками й ціною.

Методи дослідження.

У роботі застосовано методи веб-скрапінгу для збору даних, методи реляційного зберігання даних, статистичні методи описового аналізу та візуалізації, а також методи машинного навчання: кластеризацію (K-середніх), класифікацію (дерево рішень, логістична регресія, випадковий ліс, градієнтний бустинг) та ансамблеві методи.

Практичне значення роботи.

Розроблений програмний засіб може бути використаний як основа для систем підтримки прийняття рішень у сфері електронної комерції, для порівняльного аналізу товарних позицій, а також як навчальний приклад побудови повного конвеєра обробки даних – від збору до прогнозування.

1 ТЕОРЕТИЧНІ ЗАСАДИ ВИКОРИСТАНИХ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ

Машинне навчання (machine learning) – це підгалузь штучного інтелекту, що вивчає алгоритми, здатні автоматично покращувати свою роботу на основі досвіду, тобто шляхом обробки даних, а не за допомогою явно запрограмованих правил. Залежно від характеру наявних даних і поставленої задачі розрізняють навчання з учителем (supervised learning), навчання без учителя (unsupervised learning) та навчання з підкріпленням (reinforcement learning).

У межах цієї роботи використано обидва основні підходи: навчання без учителя застосовано для кластеризації моделей телефонів (виявлення прихованих груп у даних), а навчання з учителем – для класифікації, тобто віднесення пристрою до однієї з відомих моделей за його характеристиками. Нижче розглянуто теоретичні засади кожного з використаних алгоритмів.

1.1 Метод кластеризації K-середніх (K-Means)

Кластеризація – це задача навчання без учителя, що полягає в розбитті множини об'єктів на групи (кластери) таким чином, щоб об'єкти всередині однієї групи були максимально схожими між собою, а об'єкти різних груп – максимально відмінними. У роботі застосовано один із найпоширеніших алгоритмів кластеризації – метод K-середніх.

Алгоритм K-середніх працює ітеративно та складається з таких кроків:

1) задається кількість кластерів k та випадковим чином ініціалізуються центри кластерів (центроїди);

2) кожен об'єкт вибірки відноситься до того кластера, центроїд якого є найближчим (зазвичай за евклідовою відстанню);

3) для кожного кластера перераховується новий центроїд як середнє значення всіх віднесених до нього об'єктів;

4) кроки 2–3 повторюються доти, доки положення центроїдів не перестане суттєво змінюватися.

Метою алгоритму є мінімізація сумарної внутрішньокластерної дисперсії – суми квадратів відстаней від кожного об'єкта до центроїда його кластера. Ця величина в бібліотеці scikit-learn доступна як показник inertia та використовується для оцінювання якості розбиття.

Ключовою проблемою методу є вибір оптимальної кількості кластерів k. Для її розв'язання застосовують «метод ліктя» (elbow method): будують графік залежності внутрішньокластерної дисперсії від кількості кластерів і обирають таке значення k, після якого подальше збільшення кількості кластерів дає лише незначне зменшення помилки. Точка перегину графіка («лікоть») і вважається оптимальним вибором. У цій роботі метод K-середніх використано для поділу телефонів на цінові сегменти – бюджетний, середній та дорогий.

1.2 Дерево рішень (Decision Tree)

Дерево рішень – це алгоритм навчання з учителем, який представляє процес прийняття рішення у вигляді деревоподібної структури. Кожен внутрішній вузол дерева відповідає перевірці певної ознаки, кожна гілка – результату цієї перевірки, а кожен листовий вузол – підсумковому класу (або значенню). Класифікація нового об'єкта відбувається шляхом проходження від кореня дерева до листа відповідно до значень його ознак.

Побудова дерева полягає в послідовному розбитті навчальної вибірки на підмножини за тими ознаками, що дають найбільше зменшення «нечистоти» (impurity) даних. Як міру нечистоти найчастіше використовують індекс Джині (Gini impurity) або ентропію. Індекс Джині для вузла обчислюється як одиниця мінус сума квадратів часток кожного класу у вузлі; він приймає значення 0 для абсолютно однорідного вузла.

Для кореня дерева обирається та ознака, яка забезпечує найменшу нечистоту після розбиття. Далі процес рекурсивно повторюється для

отриманих підмножин, поки не буде досягнуто однорідності листів або іншого критерію зупинки. Для неперервних (числових) ознак алгоритм перебирає можливі порогові значення (наприклад, середні між сусідніми значеннями) та обирає найкращий поріг.

Перевагами дерев рішень є інтуїтивна зрозумілість та невибагливість до попередньої підготовки даних. Проте дерева схильні до перенавчання та високої чутливості: навіть незначна зміна вхідних даних може суттєво вплинути на кінцеву структуру дерева й, відповідно, на результат класифікації.

1.3 Логістична регресія (Logistic Regression)

Логістична регресія – це лінійна модель, призначена для розв'язання задач класифікації. На відміну від лінійної регресії, яка прогнозує неперервне числове значення за допомогою прямої лінії, логістична регресія прогнозує ймовірність належності об'єкта до певного класу, використовуючи логістичну (сигмоїдну) функцію. Сигмоїда перетворює будь-яке дійсне число у значення з інтервалу від 0 до 1, що інтерпретується як ймовірність.

Навчання логістичної регресії полягає в підборі вагових коефіцієнтів моделі шляхом максимізації функції правдоподібності (likelihood). Процес можна описати так:

1) обирається початкова ймовірність, масштабована ваговими коефіцієнтами;

2) на її основі обчислюється правдоподібність кожного спостереження;

3) правдоподібності всіх спостережень перемножуються, утворюючи загальну правдоподібність даних за поточної моделі;

4) параметри моделі коригуються та кроки повторюються, доки не буде досягнуто максимуму правдоподібності.

Класична логістична регресія найкраще працює для бінарної класифікації, проте за допомогою стратегії «один проти решти» (one-vs-rest) її можна застосувати й до багатокласових задач. Перевагою методу є простота

та інтерпретованість, однак він поступається в точності складнішим нелінійним алгоритмам, особливо коли межі між класами мають складну форму. Як буде показано в практичній частині, для досліджуваної задачі логістична регресія показала помітно нижчу точність порівняно з деревоподібними методами.

1.4 Випадковий ліс (Random Forest)

Випадковий ліс – це ансамблевий алгоритм навчання з учителем, що ґрунтується на побудові великої кількості дерев рішень та об'єднанні їхніх прогнозів. Ідея методу полягає в тому, що сукупність багатьох «слабких» (схильних до помилок) дерев у середньому дає значно стабільніший і точніший результат, ніж окреме дерево. Це дозволяє подолати головний недолік одиночного дерева рішень – схильність до перенавчання.

Алгоритм будує кожне дерево на основі випадкової підвибірки навчальних даних, отриманої методом бутстрепу (вибірка з поверненням). Крім того, у кожному вузлі дерева розглядається не вся множина ознак, а лише її випадкова підмножина. Така подвійна випадковість забезпечує різноманітність дерев та знижує кореляцію між їхніми помилками.

Під час класифікації нового об'єкта він пропускається через усі побудовані дерева, кожне з яких видає свій прогноз. Підсумковий клас визначається голосуванням більшості – тим класом, за який «проголосувала» найбільша кількість дерев. Принцип роботи алгоритму схематично показано на рисунку нижче.

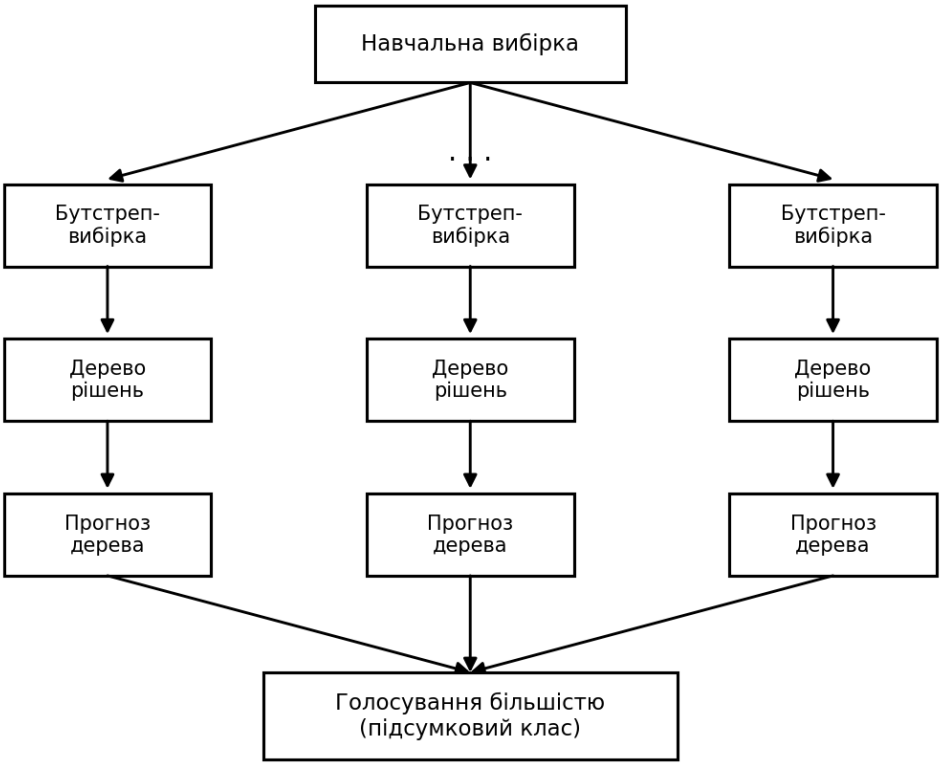


Рисунок 1.1 – Принцип роботи алгоритму Random Forest

Випадковий ліс особливо ефективний у випадках, коли дані мають високу щільність значень і складну, нелінійну структуру, де методи чисельного наближення або пошуку загальних закономірностей виявляються непридатними. Саме тому, як буде показано далі, цей алгоритм виявився найдоцільнішим для поставленої задачі класифікації моделей телефонів.

1.5 Градієнтний бустинг XGBoost

XGBoost (Extreme Gradient Boosting) – це високопродуктивна реалізація алгоритму градієнтного бустингу над деревами рішень. На відміну від випадкового лісу, де дерева будуються незалежно та паралельно, у бустингу дерева будуються послідовно: кожне наступне дерево намагається виправити помилки, допущені сукупністю попередніх дерев.

У градієнтному бустингу мінімізація помилки відбувається за рахунок руху в напрямку антиградієнта функції втрат: кожне нове дерево навчається

прогнозувати залишки (різницю між реальними та передбаченими значеннями) попередньої моделі. XGBoost доповнює класичний підхід регуляризацією, яка штрафує надмірну складність моделі та запобігає перенавчанню, а також низкою оптимізацій, що забезпечують високу швидкодію.

Завдяки поєднанню точності й ефективності XGBoost вважається одним із найпопулярніших алгоритмів для роботи зі структурованими (табличними) даними та часто перемагає в змаганнях з аналізу даних. У цій роботі XGBoost розглянуто як один із кандидатів серед класифікаційних моделей.

1.6 Ансамблеві методи (Voting Classifier)

Ансамблеві методи об'єднують прогнози кількох базових моделей задля отримання точнішого та надійнішого результату. Окремим випадком є голосувальний класифікатор (Voting Classifier), який комбінує прогнози кількох різнотипних алгоритмів.

Розрізняють два типи голосування. «Жорстке» голосування (hard voting) обирає клас, за який проголосувала більшість моделей. «М'яке» голосування (soft voting) усереднює спрогнозовані моделями ймовірності та обирає клас із найвищою усередненою ймовірністю. У роботі застосовано жорстке голосування, що поєднує дерево рішень та випадковий ліс.

Ансамблеві методи дозволяють компенсувати слабкі сторони окремих алгоритмів: помилки однієї моделі можуть бути виправлені правильними прогнозами інших. Проте, як показала практична частина, для досліджуваної задачі ансамбль не дав суттєвого виграшу порівняно з окремо взятим випадковим лісом.

1.7 Метрики оцінювання якості класифікації

Для об'єктивного порівняння побудованих моделей використовують метрики якості класифікації. Основними з них є:

– точність (accuracy) – частка правильно класифікованих об'єктів від загальної кількості; найпростіша та найнаочніша метрика;

– прецизійність (precision) – частка справді правильних серед усіх об'єктів, віднесених моделлю до певного класу;

– повнота (recall) – частка правильно знайдених об'єктів класу серед усіх об'єктів цього класу;

– F1-міра (f1-score) – гармонічне середнє прецизійності та повноти, що дає збалансовану оцінку;

– матриця плутанини (confusion matrix) – таблиця, що показує співвідношення між реальними та спрогнозованими класами й дозволяє виявити, які саме класи модель плутає між собою.

У практичній частині роботи всі побудовані моделі оцінюються за цими метриками, що дозволяє обґрунтовано обрати найкращий алгоритм.

Для систематизації розглянутих методів доцільно зіставити їхні сильні та слабкі сторони, а також придатність до задач різного типу. Узагальнене порівняння класифікаційних алгоритмів наведено в таблиці 1.1.

Таблиця 1.1 – Порівняння класифікаційних алгоритмів

Алгоритм	**Переваги**	**Недоліки**
Дерево рішень	Інтуїтивність, мінімальна підготовка даних	Схильність до перенавчання, висока чутливість до змін
Логістична регресія	Простота, інтерпретованість, швидкість	Низька точність на нелінійних даних
Випадковий ліс	Висока точність, стійкість до перенавчання	Менша інтерпретованість, вищі обчислювальні витрати
XGBoost	Дуже висока точність, регуляризація	Складніше налаштування гіперпараметрів
Ансамбль (Voting)	Компенсація помилок окремих моделей	Зростання складності без гарантії виграшу

Наведене порівняння підтверджує, що для задач зі складною нелінійною структурою даних деревоподібні ансамблеві методи (випадковий ліс, XGBoost) мають суттєву перевагу над лінійними моделями. Водночас остаточний вибір алгоритму потребує емпіричної перевірки на конкретних даних, що й виконано в практичній частині роботи.

1.8 Принципи підготовки даних для машинного навчання

Якість роботи будь-якого алгоритму машинного навчання значною мірою залежить від якості вхідних даних. Етап попередньої підготовки (preprocessing) часто є найбільш трудомістким у всьому процесі та включає очищення, перетворення й кодування даних. Без належної підготовки навіть найдосконаліший алгоритм здатний давати незадовільні результати, що відоме як принцип «сміття на вході – сміття на виході» (garbage in, garbage out).

Очищення даних полягає у виявленні та обробці пропущених, некоректних або аномальних значень. Пропущені значення можуть бути або вилучені разом із відповідними записами, або заповнені (імпутовані) – наприклад, середнім, медіаною чи прогнозованим значенням. У цій роботі товари без вказаної ціни вилучаються з навчальної вибірки, а для імітованої перевірки відсутні ціни заповнюються середнім значенням бренду з випадковим відхиленням.

Більшість алгоритмів машинного навчання працює виключно з числовими даними, тому категоріальні (текстові) ознаки потребують кодування. Найпоширенішими підходами є кодування мітками (label encoding), коли кожній категорії присвоюється унікальне ціле число, та пряме кодування (one-hot encoding), коли категоріальна ознака перетворюється на набір бінарних стовпців – по одному на кожну категорію. Останній підхід уникає хибного припущення про порядкову залежність між категоріями, тому застосовується для номінальних ознак, таких як розмір дисплея в цій роботі.

Окремою важливою процедурою є масштабування числових ознак, що приводить їх до співставних діапазонів. Воно критичне для алгоритмів, чутливих до масштабу (зокрема для методів на основі відстаней, таких як K-середніх, та для логістичної регресії), проте практично не впливає на роботу деревоподібних методів, які оперують пороговими розбиттями, а не абсолютними значеннями.

Однією з ключових проблем машинного навчання є перенавчання (overfitting) – ситуація, коли модель надмірно точно підлаштовується під навчальні дані, фактично «запам'ятовуючи» їх разом із випадковим шумом, замість того щоб виявити узагальнені закономірності. Перенавчена модель показує високу точність на навчальній вибірці, але погано працює на нових, раніше не бачених даних. Протилежним явищем є недонавчання (underfitting), коли модель занадто проста, щоб уловити структуру даних.

Для виявлення перенавчання та об'єктивної оцінки якості моделі застосовують розбиття наявних даних на навчальну та тестову вибірки: модель навчається на одній частині даних, а її якість оцінюється на іншій, яка не використовувалася під час навчання. У цій роботі застосовано розбиття у співвідношенні 75 % до 25 %, що є поширеною практикою. Додатковою перевіркою стійкості моделі слугує оцінювання на окремій імітованій вибірці.

Більш надійним методом оцінювання є перехресна перевірка (cross-validation), за якої дані багаторазово розбиваються на навчальну й тестову частини різними способами, а підсумкова оцінка усереднюється. Це зменшує залежність результату від конкретного випадкового розбиття. Боротьба з перенавчанням також забезпечується методами регуляризації (як в XGBoost) та використанням ансамблів (як у випадковому лісі), які за своєю природою стійкіші до цього явища.

1.9 Висновки до розділу 1

У розділі розглянуто теоретичні засади методів машинного навчання, застосованих у роботі. Описано метод кластеризації K-середніх для виявлення цінових сегментів, а також низку класифікаційних алгоритмів – дерево рішень, логістичну регресію, випадковий ліс, градієнтний бустинг XGBoost та голосувальний ансамбль. Окреслено принципи підготовки даних, проблему перенавчання та методи валідації, а також метрики оцінювання якості класифікації. Теоретичний аналіз дозволяє припустити, що для задачі з високою щільністю та складною структурою даних найдоцільнішими будуть деревоподібні ансамблеві методи, що й перевірено в подальших розділах.

2 ПРОГРАМНА РЕАЛІЗАЦІЯ ДОСТУПУ ДО ВЕБ-РЕСУРСУ

2.1 Технологія веб-скрапінгу

Першим етапом роботи будь-якої аналітичної системи є отримання вихідних даних. У межах цієї роботи джерелом даних обрано онлайн-каталог мобільних телефонів інтернет-магазину. Завдання модуля доступу – автоматично звернутися до веб-ресурсу, отримати HTML-розмітку сторінок каталогу та вилучити з неї структуровану інформацію про товари: артикул, назву, ціну, посилання на фотографію та технічні характеристики.

Веб-скрапінг (web scraping) – це автоматизоване вилучення даних із веб-сторінок. Технічно процес складається з двох основних кроків: завантаження HTML-коду сторінки (зазвичай за допомогою HTTP-запиту) та подальшого синтаксичного аналізу (парсингу) цього коду для виокремлення потрібних елементів. Для виконання HTTP-запитів у мові Python широко використовується бібліотека requests, а для парсингу HTML існує кілька підходів, головними з яких є бібліотека BeautifulSoup та фреймворк Selenium.

2.2 Бібліотека BeautifulSoup та Selenium

BeautifulSoup – це бібліотека мови Python, призначена для вилучення даних з HTML- та XML-документів. Вона перетворює сирий HTML-текст на деревоподібний об'єкт, навігація яким здійснюється за допомогою методів пошуку за тегами, атрибутами, класами та CSS-селекторами. BeautifulSoup не виконує JavaScript і працює виключно зі статичним HTML, отриманим у відповідь на HTTP-запит.

Перевагами BeautifulSoup є простота використання, мінімальні вимоги до ресурсів та висока швидкодія, оскільки бібліотека не запускає браузер і не очікує на завантаження динамічного вмісту. Бібліотека ідеально підходить для

роботи із сайтами, що повертають готову HTML-розмітку безпосередньо у відповіді сервера.

Selenium – це інструмент автоматизації браузера, що спочатку був створений для тестування веб-застосунків. Selenium керує реальним (або «безголовим») браузером через спеціальний драйвер (WebDriver), завдяки чому здатний повноцінно відтворювати поведінку користувача: натискати кнопки, прокручувати сторінки, заповнювати форми та, що найважливіше, виконувати JavaScript. Це робить Selenium незамінним для роботи з динамічними сайтами, на яких дані формуються вже після завантаження сторінки за допомогою клієнтських скриптів.

Водночас за ці можливості доводиться платити: запуск повноцінного браузера споживає значно більше оперативної пам'яті та процесорного часу, а очікування на рендеринг сторінок суттєво уповільнює процес збору даних. Крім того, конфігурація Selenium складніша – вона потребує встановлення відповідного драйвера браузера.

Для обґрунтованого вибору інструмента доступу до веб-ресурсу проведено порівняння BeautifulSoup та Selenium за низкою критеріїв. Результати порівняння наведено в таблиці 2.1.

Таблиця 2.1 – Порівняння BeautifulSoup та Selenium

Критерій	**BeautifulSoup**	**Selenium**
Виконання JavaScript	Ні	Так
Споживання ресурсів	Низьке	Високе
Швидкодія	Висока	Нижча
Складність налаштування	Низька	Вища (потрібен WebDriver)
Тип сторінок	Статичні HTML	Статичні та динамічні
Емуляція дій користувача	Ні	Так
Доцільність для каталогу	Висока	Надлишкова

Ключовим критерієм вибору є характер цільового веб-ресурсу. Аналіз сторінок обраного каталогу показав, що вся необхідна інформація про товари –

назва, ціна, характеристики – присутня безпосередньо в HTML-розмітці, яку сервер повертає у відповідь на звичайний HTTP-запит, без потреби у виконанні JavaScript. За таких умов застосування Selenium було б надлишковим: запуск повноцінного браузера лише уповільнив би роботу та збільшив споживання ресурсів, не надаючи жодних переваг.

Логіку вибору інструмента доступу узагальнено на рисунку 2.1. Оскільки для отримання даних не потрібен рендеринг JavaScript, оптимальним рішенням є зв'язка requests + BeautifulSoup.

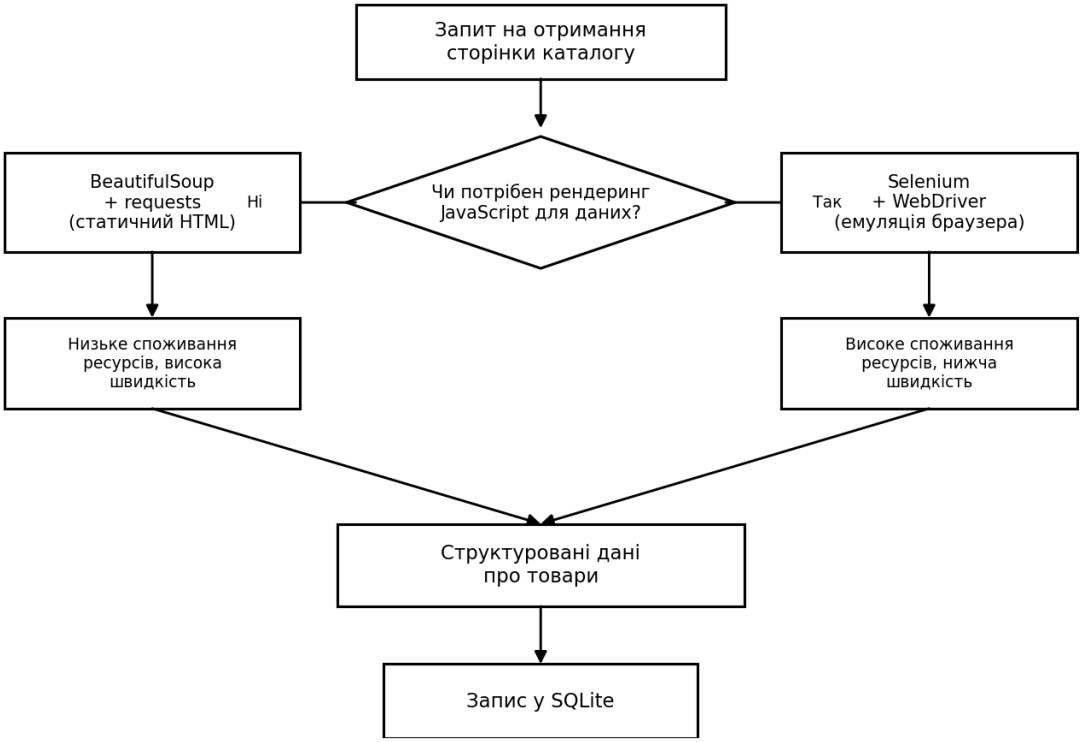


Рисунок 2.1 – Схема вибору інструмента доступу до веб-ресурсу

Таким чином, для реалізації модуля доступу обрано бібліотеку BeautifulSoup у поєднанні з бібліотекою requests. Цей вибір забезпечує найкраще співвідношення швидкодії, простоти реалізації та споживання ресурсів для поставленої задачі збору даних зі статичного каталогу.

2.3 Реалізація запиту до веб-ресурсу

Початковий етап роботи модуля доступу полягає у формуванні базової адреси каталогу та надсиланні першого HTTP-запиту за допомогою бібліотеки requests. Отриману відповідь зберігають в об'єкті response, статус-код якого дозволяє перевірити успішність запиту. Відповідний фрагмент коду наведено нижче.

Лістинг 2.1 – Формування адреси та надсилання HTTP-запиту

```
url_base = "https://can.ua/smartphones/c1538/"  # початкова сторінка з даними
url_page = "page="                               # додаток для отримання сторінки

# отримання першої сторінки
data = requests.get(url_base + url_page + str(1))
```

Після отримання відповіді програма перевіряє її статус-код. Значення 200 означає, що сервер успішно повернув дані, після чого текст відповіді конвертується в об'єкт BeautifulSoup для подальшого парсингу. У разі іншого статус-коду користувачеві виводиться відповідне повідомлення, і робота програми завершується.

2.4 Структура HTTP-запиту та HTML-розмітки

Для розуміння принципу роботи модуля доступу варто коротко розглянути основи взаємодії клієнта та сервера. Протокол HTTP (HyperText Transfer Protocol) є основою обміну даними у вебі та працює за моделлю «запит-відповідь»: клієнт надсилає запит на певну адресу (URL), а сервер повертає відповідь, що складається зі статус-коду, заголовків і тіла. Статус-код 200 означає успішне виконання запиту, тоді як коди серії 4xx вказують на помилки клієнта (наприклад, 404 – ресурс не знайдено), а 5xx – на помилки сервера.

Тіло відповіді для веб-сторінок зазвичай містить HTML-розмітку – ієрархічну структуру вкладених тегів, що описують вміст та структуру

документа. Кожен елемент може мати атрибути, зокрема class та id, які використовуються для стилізації та ідентифікації. Саме за цими атрибутами бібліотека BeautifulSoup здійснює пошук потрібних елементів за допомогою CSS-селекторів – компактних виразів, що описують шлях до елементів у дереві документа.

У реалізованому модулі параметр page= у складі URL використовується для навігації між сторінками каталогу, що дозволяє послідовно отримати всі товари. Після отримання HTML кожної сторінки програма за допомогою селекторів вилучає блоки окремих товарів та зчитує з них необхідні характеристики, як описано у наступному розділі.

2.5 Висновки до розділу 2

У розділі розглянуто технологію веб-скрапінгу та два основні інструменти доступу до веб-ресурсів – бібліотеку BeautifulSoup і фреймворк Selenium. Проведено їх порівняння за критеріями швидкодії, споживання ресурсів, складності налаштування та підтримки динамічного вмісту. Оскільки цільовий каталог повертає всі необхідні дані у статичній HTML-розмітці, для реалізації модуля доступу обрано BeautifulSoup у поєднанні з requests як найбільш ефективне та просте рішення. Наведено фрагменти коду формування запиту до веб-ресурсу.

3 ОПИС РОБОТИ ПРОГРАМНОГО КОДУ

У цьому розділі детально розглянуто роботу програмного коду, що реалізує збір, збереження та візуалізацію даних про мобільні телефони. Програму побудовано у вигляді послідовного конвеєра: від звернення до веб-ресурсу до отримання очищеного набору даних, придатного для подальшого машинного навчання. Узагальнену архітектуру програмного засобу наведено на рисунку 3.1.

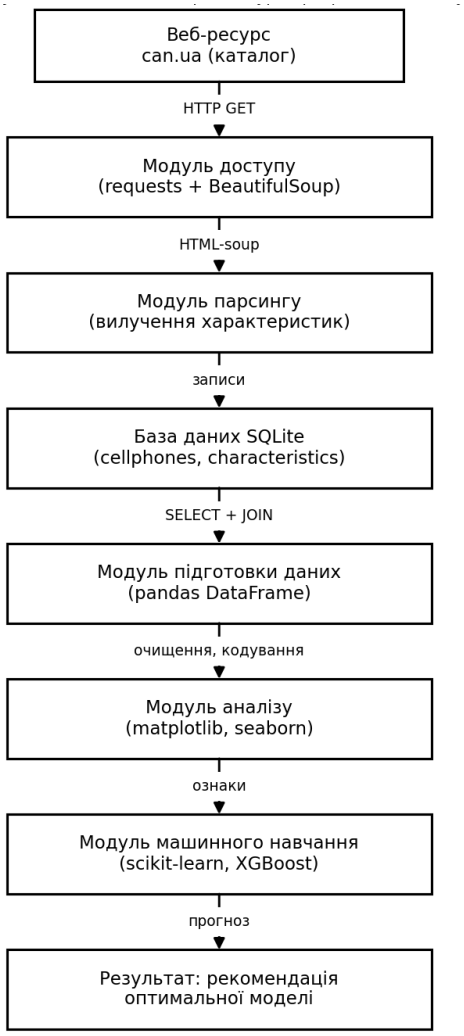


Рисунок 3.1 – Узагальнена архітектура програмного засобу

3.1 Проєктування та створення бази даних

Роботу програми розпочато з імпорту необхідних бібліотек. Бібліотека requests використовується для звернення до веб-ресурсів, BeautifulSoup – для конвертації відповіді сервера в об'єкт HTML, sqlite3 – для роботи з базою даних SQLite, а pandas – для роботи з табличними даними (датафреймами).

Лістинг 3.1 – Імпорт основних бібліотек

```
import requests                  # бібліотека для здійснення запитів
from bs4 import BeautifulSoup    # для парсингу html
import sqlite3                   # для роботи з БД, зокрема SQLite
import pandas as pd              # для роботи з датафреймами
```

Зібрані дані зберігаються у реляційній базі даних SQLite. Структуру бази даних спроєктовано з двох пов'язаних таблиць: таблиця cellphones містить загальну інформацію про товар (артикул, назву, посилання на фото, ціну та модель), а таблиця characteristics – його технічні характеристики (діагональ і розмір дисплея, кількість ядер, обсяг оперативної та внутрішньої пам'яті, роздільну здатність камери та операційну систему). Таблиці пов'язані за артикулом товару через зовнішній ключ. Схему бази даних, побудовану в середовищі Power BI, наведено на рисунку 3.2.

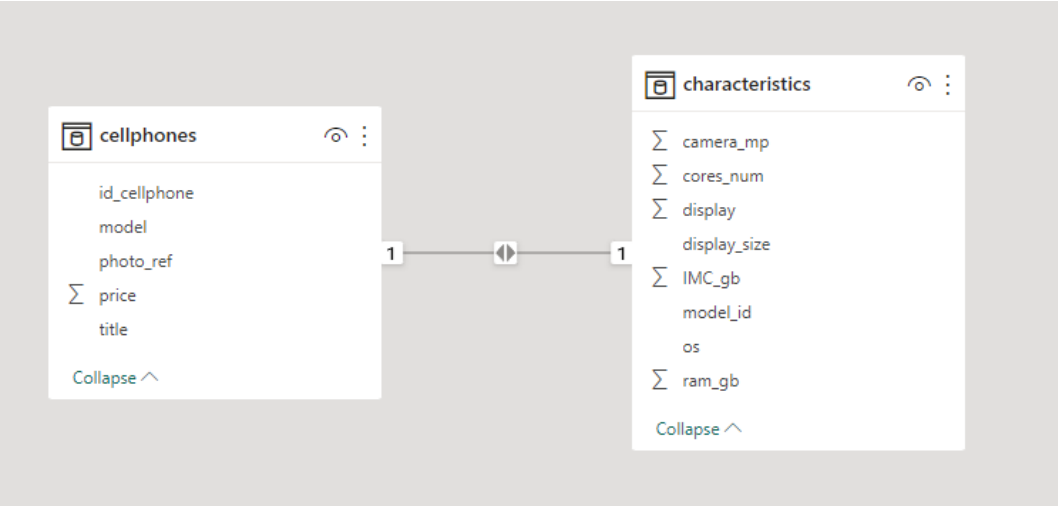


Рисунок 3.2 – Схема спроєктованої бази даних

Якщо сервер успішно повернув дані (статус-код 200), текст відповіді конвертується в об'єкт BeautifulSoup, після чого створюється з'єднання з базою даних і за допомогою SQL-запитів формуються обидві таблиці. Відповідний фрагмент коду наведено в лістингу 3.2.

Лістинг 3.2 – Підключення до БД та створення таблиць

```
if data.status_code == 200:
    print("Data was successfully retrieved (200)")
    # парсимо текст відповіді в об'єкт BeautifulSoup
    soup = BeautifulSoup(data.text, "html.parser")
    # створюємо базу даних або підключаємось до існуючої
    db = sqlite3.connect("cellphones.db")
    cur = db.cursor()

    cur.execute('''CREATE TABLE IF NOT EXISTS cellphones(
        id_cellphone INT NOT NULL PRIMARY KEY,
        title VARCHAR(100) NOT NULL,
        photo_ref VARCHAR(255) NULL,
        price INT NOT NULL,
        model VARCHAR(100) NOT NULL )''')

    cur.execute('''CREATE TABLE IF NOT EXISTS characteristics(
        model_id INT NOT NULL PRIMARY KEY,
        display FLOAT NOT NULL,
        display_size VARCHAR(45) NOT NULL,
        cores_num INT NOT NULL,
        ram_gb INT NOT NULL,
        IMC_gb INT NOT NULL,
        camera_mp INT NOT NULL,
        os VARCHAR(45) NOT NULL,
        FOREIGN KEY(model_id) REFERENCES cellphones(id_cellphone))''')
    db.commit()
    cur.close()
```

3.2 Збір та парсинг даних

Основний цикл збору даних спершу визначає загальну кількість сторінок каталогу як максимальний елемент списку пагінації, а також зчитує перелік доступних на сайті брендів (моделей). Далі програма у циклі проходить усі сторінки каталогу: для кожної сторінки надсилається запит, відповідь парситься в об'єкт BeautifulSoup, після чого з неї за допомогою CSS-селекторів вилучаються всі товари.

Для кожного товару зчитуються його артикул, назва, ціна (за відсутності – 0, що означає відсутність товару в наявності), посилання на фотографію, бренд (визначається пошуком назви бренду у назві товару) та технічні характеристики з прихованого блоку. Зчитані дані записуються в обидві таблиці бази даних. Скорочений фрагмент циклу збору даних наведено в лістингу 3.3.

Лістинг 3.3 – Цикл збору та запису даних

```
if data.status_code == 200:
    # кількість сторінок розділу
    pages_qty = int(soup.select_one(
        "ul[name='paginator'] li:nth-last-child(2)").get_text())
    # перелік брендів зі списку фільтрів сайту
    model_labels = soup.select("#desktop_sort-producer > ul > li > label")
    models = [m.get_text().split("(")[0].strip() for m in model_labels]

    for i in range(1, pages_qty + 1):              # цикл по всіх сторінках
        print("Fulfilling page #", i, "... of", pages_qty)
        if i != 1:                                 # перша сторінка вже отримана
            data = requests.get(url_base + url_page + str(i))
        soup = BeautifulSoup(data.text, "html.parser")
        page_products = soup.select(".catalog-tile li")

        for product in page_products:              # цикл по товарах сторінки
            id = product.select_one("div.info__code").get_text().strip()[5:]
            title = product.select_one("a.info__name").get_text()
            price = 0
            price_element = product.select_one("p.item__price")
            if price_element:
                price = price_element.get_text()[:-4].replace(" ", "")
            photo_ref = product.select_one("img")['src']
            model = ""
            for curr_model in models:              # визначення бренду
                if curr_model in title:
                    model = curr_model
                    break
            # зчитування технічних характеристик
            display = product.select_one(
                ".item__hidden > p > b:nth-child(1)").get_text()[:-2]
            # ... (інші характеристики зчитуються аналогічно)
            cur = db.cursor()
            cur.execute('INSERT INTO cellphones VALUES(?, ?, ?, ?, ?)',
                        (id, title, photo_ref, price, model))
            db.commit()
            cur.close()
    db.close()
    print("Done")
```

Для забезпечення цілісності даних операції запису обгорнуто в обробник винятків: у разі порушення унікальності первинного ключа (спроби повторного запису того самого товару) програма виводить відповідне

повідомлення та припиняє парсинг, що запобігає дублюванню записів у базі даних.

3.3 Зчитування даних із бази даних

Після завершення збору даних інформація з обох таблиць об'єднується за допомогою SQL-запиту з операцією LEFT JOIN за спільним ключем і завантажується у датафрейм pandas. Українські назви стовпців задаються явно для зручності подальшого аналізу. Відповідний фрагмент коду наведено в лістингу 3.4.

Лістинг 3.4 – Зчитування об'єднаних даних у датафрейм

```
db = sqlite3.connect("cellphones.db")
cur = db.cursor()
cur.execute('''SELECT id_cellphone, title, photo_ref, price, model,
                      display, display_size, cores_num, ram_gb,
                      IMC_gb, camera_mp, os
               FROM cellphones
               LEFT JOIN characteristics
                    ON cellphones.id_cellphone = characteristics.model_id''')
df = pd.DataFrame(cur.fetchall(),
    columns=["Артикул", "Назва", "Фото", "Ціна", "Модель", "Дисплей",
             "Розмір дисплею", "Кількість ядер", "Оперативна пам'ять",
             "Внутрішня пам'ять", "Камера", "Операційка"])
cur.close()
db.close()
```

3.4 Опис структури зібраного набору даних

Після виконання повного циклу збору отриманий набір даних містить інформацію про сотні моделей мобільних телефонів. Кожен запис описується дванадцятьма полями, що поділяються на ідентифікаційні (артикул, назва, посилання на фото), цінові (ціна, бренд) та технічні (діагональ і розмір дисплея, кількість ядер, обсяг оперативної та внутрішньої пам'яті, роздільна здатність камери, операційна система). Перед аналізом виконано перевірку набору на наявність пропущених значень, а також отримано описову статистику числових ознак за допомогою методу describe бібліотеки pandas, що дозволяє оцінити діапазони, середні значення та розкид характеристик.

Аналіз описової статистики показав значний розкид цін – від кількох тисяч до понад вісімдесяти тисяч гривень, що підтверджує необхідність сегментації моделей за ціновими категоріями. Технічні характеристики також мають широкий діапазон значень, особливо щодо обсягу внутрішньої пам'яті та роздільної здатності камери, які, як буде показано далі, найсильніше впливають на ціну пристрою.

3.5 Візуалізація зібраних даних

Для попереднього аналізу зібраних даних використано бібліотеки matplotlib і seaborn. Перед побудовою графіків із набору вилучено товари з нульовою ціною (відсутні в наявності). Першу групу візуалізацій присвячено аналізу брендів за двома показниками – середньою ціною та кількістю моделей. Результат наведено на рисунку 3.3.

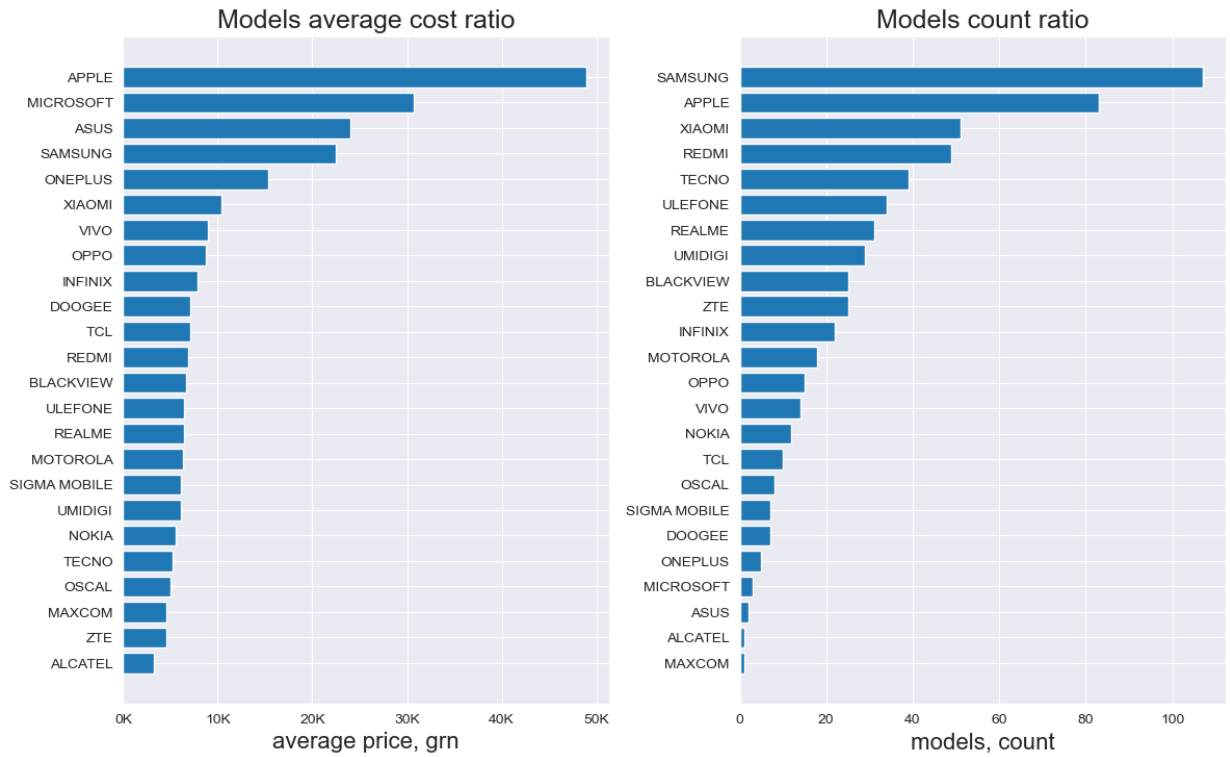


Рисунок 3.3 – Середня ціна та кількість моделей за брендами

З лівої діаграми видно, що найвищу середню ціну мають бренди APPLE, MICROSOFT та ASUS, тоді як права діаграма демонструє, що за кількістю представлених на ринку моделей лідирують SAMSUNG, APPLE, XIAOMI та REDMI. Поєднання цих двох показників на одній діаграмі розсіювання (рисунок 3.4) дозволяє оцінити одночасно і популярність, і цінове позиціонування кожного бренду.

Діаграма розсіювання чітко показує особливе положення бренду APPLE: за відносно великої кількості моделей він має найвищу середню ціну, що свідчить про значну надбавку за бренд. SAMSUNG, навпаки, поєднує найбільшу кількість моделей із помірною середньою ціною. Для подальшого аналізу обрано п'ять найпопулярніших брендів, представлених на кругових діаграмах на рисунку 3.5.

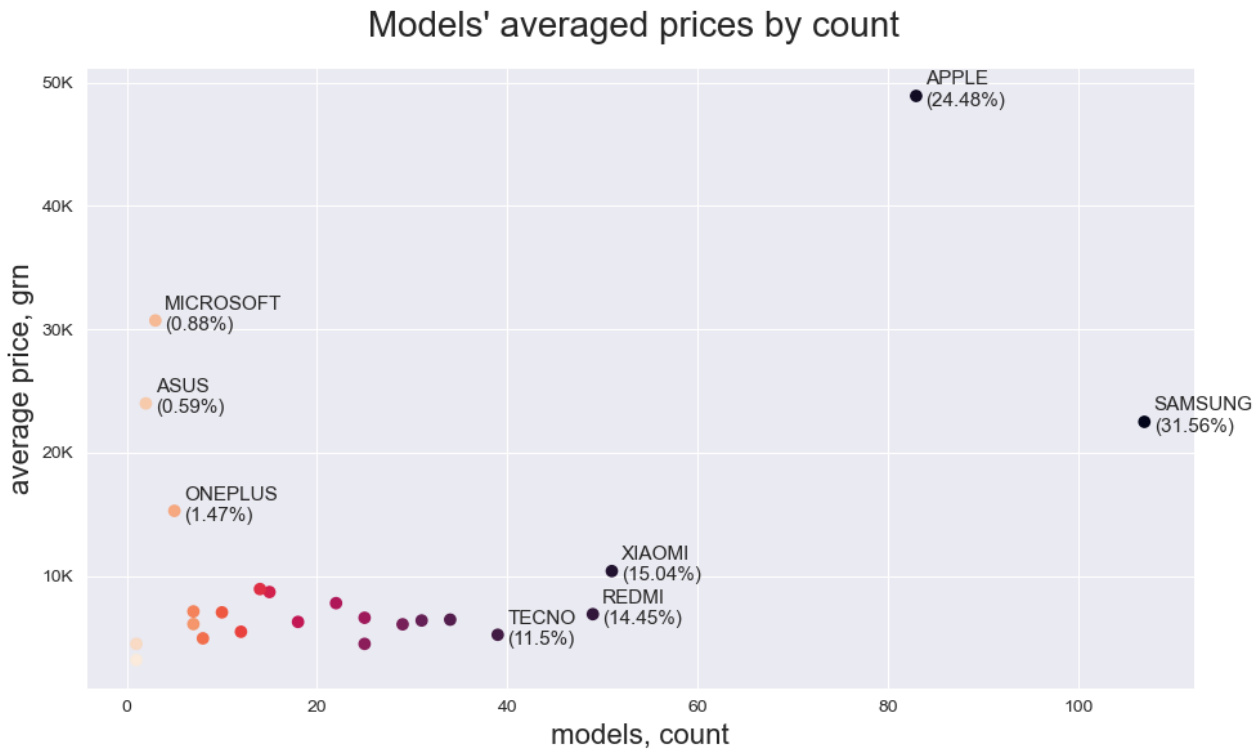


Рисунок 3.4 – Співвідношення середньої ціни та кількості моделей за брендами

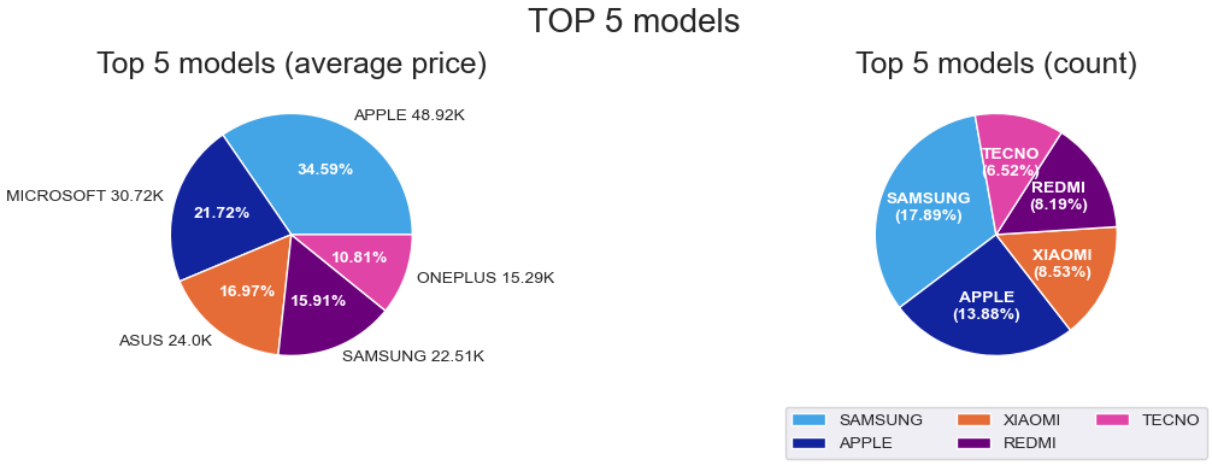


Рисунок 3.5 – Топ-5 брендів за середньою ціною та кількістю моделей

Аналіз показав, що до п'ятірки найпопулярніших за кількістю моделей увійшли SAMSUNG, APPLE, XIAOMI, REDMI та TECNO. Саме на цих брендах зосереджено подальше навчання класифікаційних моделей, оскільки решта брендів представлена недостатньою для надійного навчання кількістю зразків.

3.6 Побудова та навчання моделей машинного навчання

Заключний етап роботи програмного засобу полягає в застосуванні методів машинного навчання до підготовлених даних. Постановка задачі формулюється так: стоїть задача вибору телефону, коли фінансів вистачає навіть на найдорожчу модель, проте на першому місці – доцільність і релевантність витрати. Тобто необхідно розв'язати задачу максимізації користі телефону за його ціну. Для цього виконано низку підзадач: враховано репутацію та популярність кожної моделі на ринку, визначено принципи поділу телефонів на групи (що найбільше впливає на ціну) та навчено мережу розпізнавати модель телефону за його характеристиками.

3.6.1 Імпорт інструментів машинного навчання

Для реалізації моделей машинного навчання використано бібліотеку scikit-learn та бібліотеку xgboost. Імпортовано алгоритми кластеризації (KMeans) і класифікації (LogisticRegression,

Лістинг 3.5 – Імпорт алгоритмів машинного навчання

```
import random
from sklearn.cluster import KMeans
from sklearn.linear_model import LogisticRegression
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.metrics import (confusion_matrix,
                             classification_report, accuracy_score)
from sklearn.model_selection import train_test_split
from sklearn.ensemble import VotingClassifier
```

DecisionTreeClassifier, RandomForestClassifier, XGBClassifier), інструменти оцінювання якості (confusion_matrix, classification_report, accuracy_score), функцію розбиття вибірки train_test_split та ансамблевий класифікатор VotingClassifier.

3.6.2 Попередня обробка та кодування даних

Перед навчанням моделей дані приводяться до числового вигляду. Спершу вилучаються нерелевантні для аналізу стовпці – артикул, назва та посилання на фото. Стовпець розміру дисплея стандартизується, після чого до нього застосовується one-hot кодування (перетворення категоріальної ознаки на набір бінарних стовпців). Операційна система кодується числовими мітками: Android позначається значенням 0, iOS – 1. Конвеєр попередньої обробки та навчання узагальнено на рисунку 3.6.

Рисунок 3.6 – Конвеєр обробки даних та навчання моделей

Лістинг3.6 – Кодування категоріальних ознак

```
df = df.drop(['Артикул', 'Назва', 'Фото'], axis=1)   # вилучення зайвих стовпців
# стандартизація розміру дисплея
df['Розмір дисплею'] = df['Розмір дисплею'].apply(
    lambda x: x.split('(')[0].strip())
# one-hot кодування розміру дисплея
df = pd.get_dummies(df, columns=['Розмір дисплею'])
# кодування операційної системи
df['Операційка'] = df['Операційка'].apply(
    lambda x: 0 if x == 'Android' else (1 if x == 'iOS' else 'unfamiliar'))
```

Далі з вибірки для навчання вилучаються товари з нульовою ціною (відсутні в наявності, без інформації про вартість). Окремо ці товари зберігаються в допоміжний набір для подальшого прогнозування: відсутні ціни замінюються на середнє значення ціни для відповідного бренду з випадковим відхиленням у межах стандартного відхилення. Це дозволяє створити імітовану вибірку для додаткової перевірки якості моделі.

3.6.3 Кластеризація та визначення цінових сегментів

Для визначення оптимальної кількості цінових сегментів застосовано метод K-середніх із оцінюванням за методом ліктя. Побудовано графік залежності внутрішньокластерної дисперсії (зважених помилок) від кількості кластерів у діапазоні від 1 до 10 (рисунок 3.7).

Лістинг 3.7 – Метод ліктя для вибору кількості кластерів

```
distortions = []
for k in range(1, 11):
    kmean = KMeans(n_clusters=k, n_init=15).fit(df_learn)
    distortions.append(kmean.inertia_)  # сума квадратів відстаней
plt.plot(range(1, 11), distortions)
plt.xlabel('k, clusters'); plt.ylabel('Weighted errors')
plt.show()
```

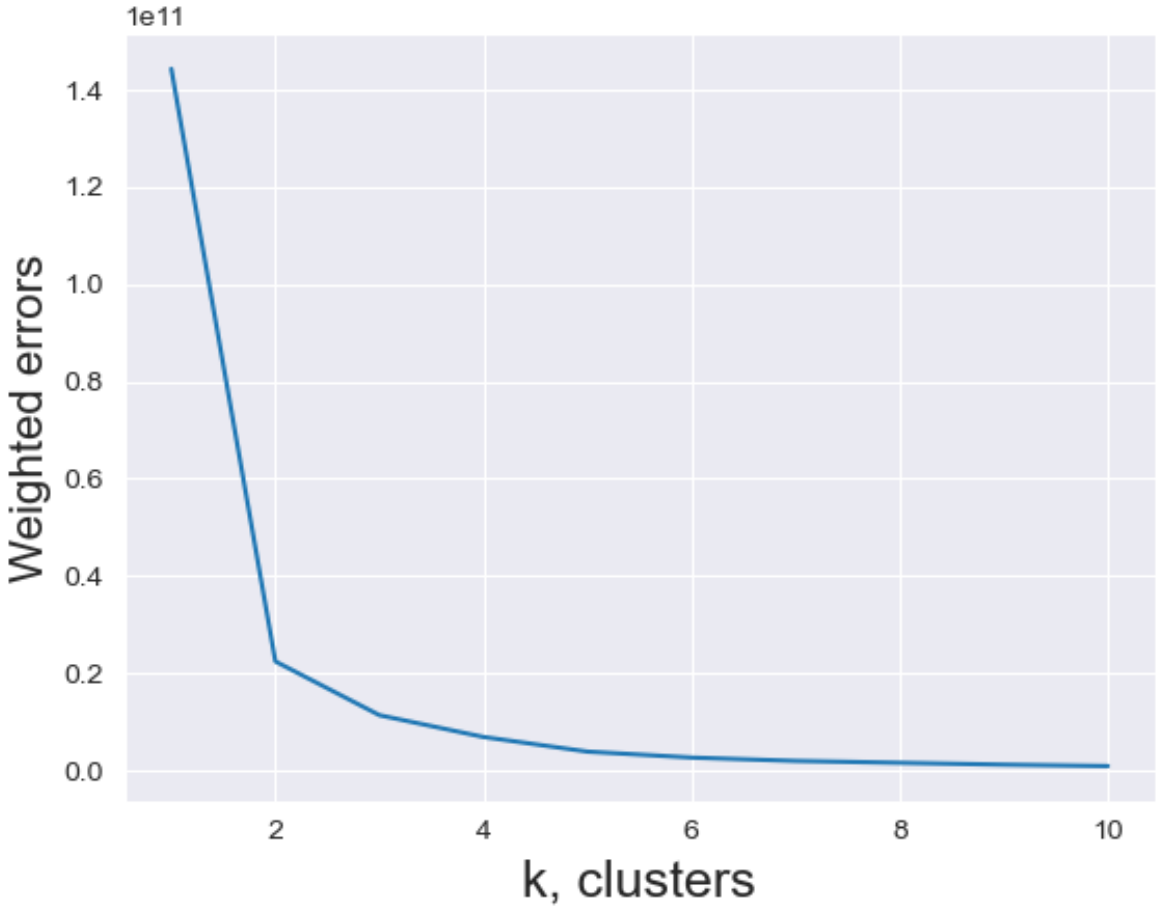


Рисунок 3.7 – Метод ліктя для вибору кількості кластерів

Графік демонструє різкий злам у районі двох-трьох кластерів, після чого крива згладжується. Для детальнішого аналізу розглянуто розбиття на два та три кластери. Результати кластеризації за ознаками ціни, внутрішньої пам'яті та камери наведено на рисунку 3.8.

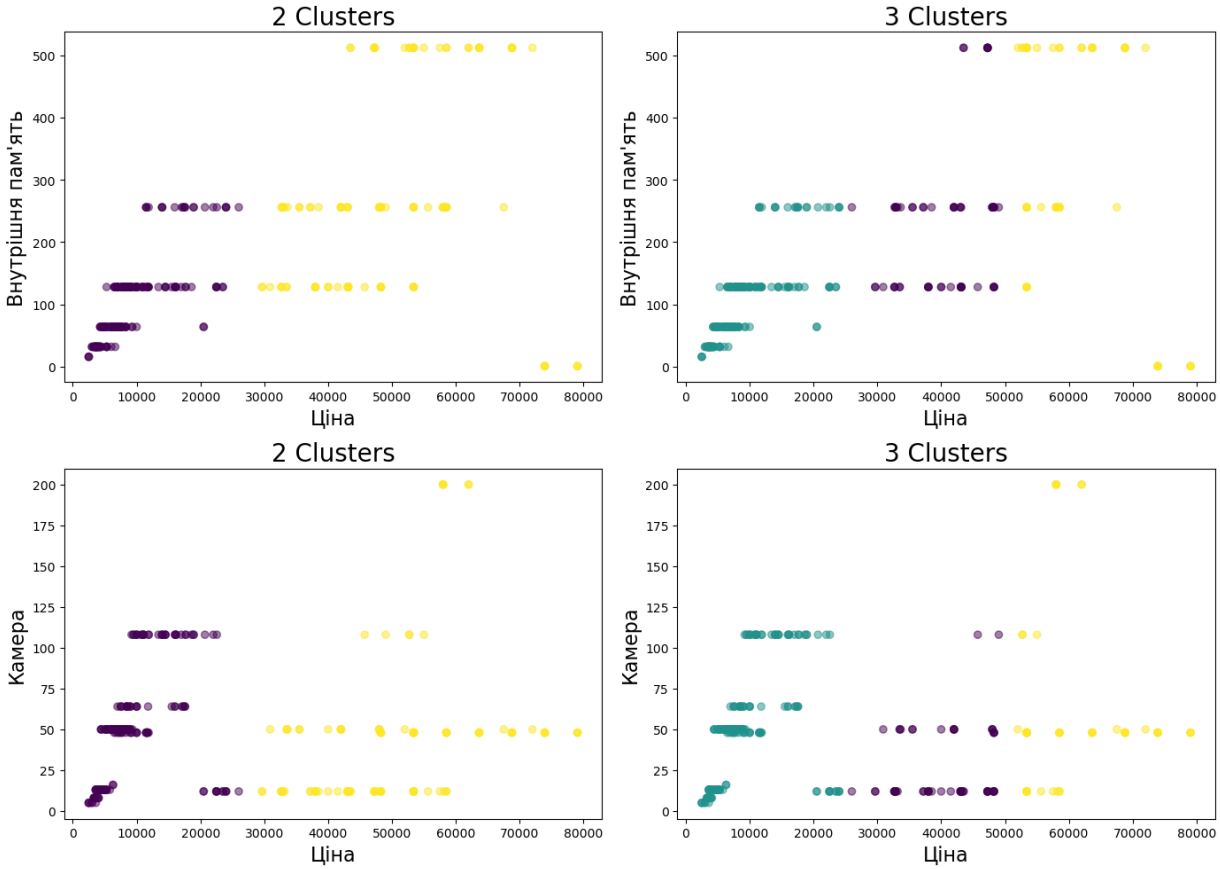


Рисунок 3.8 – Кластеризація моделей за ціною, пам'яттю та камерою

Аналіз показав, що поділ на три кластери найкраще відображає структуру даних, відповідаючи трьом ціновим сегментам: бюджетному, середньому та дорогому. На рисунку 3.9 ті самі дані розфарбовано за брендами, що підтверджує зв'язок між брендом, технічними характеристиками й ціною.

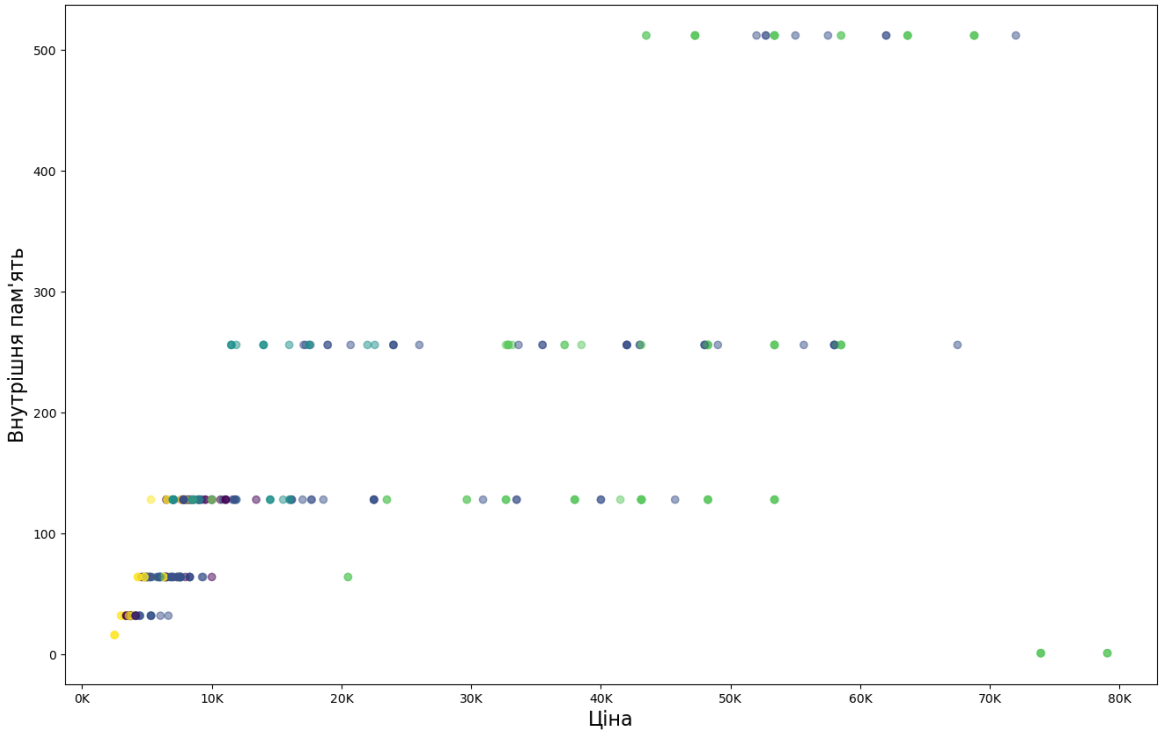


Рисунок 3.9 – Розподіл моделей за ціною та внутрішньою пам'яттю (за брендами)

3.6.4 Розбиття вибірки та навчання класифікаторів

Підготовлений набір даних розбивається на навчальну та тестову вибірки у співвідношенні 75 % до 25 % за допомогою функції train_test_split. Незалежними змінними виступають усі характеристики телефона разом із ціною, а залежною (цільовою) змінною – закодований індекс моделі.

Лістинг 3.8 – Розбиття вибірки на навчальну та тестову

```
X = df_learn.drop('Модель', axis=1).astype('float')  # незалежні змінні
Y = df_learn['Модель']                               # цільова змінна
X_train, X_test, y_train, y_test = train_test_split(
    X, Y, random_state=1, train_size=.75)
```

На підготовлених даних навчено кілька класифікаційних моделей. Першою побудовано дерево рішень – найпростішу модель із мінімальною підготовкою даних. Воно показало точність 95,18 %.

Лістинг 3.8 – Навчання дерева рішень

```
decision_tree = tree.DecisionTreeClassifier().fit(X_train, y_train)
dtree_predictions = decision_tree.predict(X_test)
print("Accuracy of Decision Tree: {0}%".format(
    accuracy_score(y_test, dtree_predictions) * 100))
# Accuracy of Decision Tree: 95.18072289156626%
```

Логістична регресія, навпаки, показала найнижчу серед усіх моделей точність – лише 55,42 %, що пояснюється складною нелінійною структурою даних, для якої лінійна модель погано підходить.

Лістинг 3.10 – Навчання логістичної регресії

```
fit_lg = LogisticRegression(penalty='none').fit(X_train, y_train)
lg_predictions = fit_lg.predict(X_test)
print("Accuracy of Logistic Regression: {0}%".format(
    accuracy_score(y_test, lg_predictions) * 100))
# Accuracy of Logistic Regression: 55.42168674698795%
```

Найкращий результат серед окремих моделей показав випадковий ліс із 800 дерев – точність 97,59 %. Завдяки ансамблевій природі та стійкості до перенавчання він найкраще впорався з високою щільністю та складною структурою даних.

Лістинг 3.11 – Навчання випадкового лісу

```
rf = RandomForestClassifier(n_estimators=800,
                            random_state=41).fit(X_train, y_train)
rf_predictions = rf.predict(X_test)
accuracy_score_rf = accuracy_score(y_test, rf_predictions)
print("Accuracy of Random Forest: {0}%".format(accuracy_score_rf * 100))
# Accuracy of Random Forest: 97.59036144578313%
```

Додатково випробувано ансамбль на основі жорсткого голосування, що поєднує дерево рішень та випадковий ліс.

Лістинг 3.12 – Ансамбль на основі голосування

```
estimators = [('decision_tree', decision_tree), ("RForest", rf)]
ensemble = VotingClassifier(estimators, voting='hard')
ensemble.fit(X_train, y_train)
Y_prediction = ensemble.predict(X_test)
print("Voting Ensemble:>" +
      str(accuracy_score(Y_prediction, y_test) * 100) + "%")
# Voting Ensemble:>97.59036144578313%
```

Його точність склала 97,59 %, тобто не перевищила результат окремо взятого випадкового лісу, що підтверджує достатність останнього.

3.6.5 Порівняння точності моделей

Зведені результати оцінювання точності всіх побудованих моделей наведено в таблиці 3.1. Окрім точності на реальній тестовій вибірці, для найкращої моделі (випадкового лісу) виконано перевірку на імітованій вибірці з відновленими цінами.

Таблиця 3.1 – Порівняння точності моделей

Модель	**Точність, %**
Дерево рішень (Decision Tree)	95,18
Логістична регресія (Logistic Regression)	55,42
Випадковий ліс (Random Forest)	97,59
Ансамбль голосування (Voting)	97,59
Random Forest (імітована вибірка)	79,47

Як видно з таблиці, найдоцільнішим методом навчання виявився Random Forest. Для цієї задачі не підійшли б методи чисельного наближення або пошуку загальних закономірностей, адже бюджетний сегмент має дуже високу щільність значень, частина з яких не пов'язана по групах. Точність навченої моделі склала 97,59 % для справжньої вибірки та 79,47 % для імітованої.

3.6.6 Формування підсумкової рекомендації

На основі найкращої моделі розв'язано задачу мінімізації ціни при максимізації характеристик окремо для кожної операційної системи. Для цього в навчену модель подано вектори з максимальними значеннями характеристик за мінімальної середньої ціни для Android та iOS відповідно.

За результатами прогнозування для операційної системи Android найкращим вибором є модель SAMSUNG, а для iOS – модель Apple. Оскільки за дослідженими даними для Apple значна частина вартості припадає на надбавку за бренд (найвище середнє значення ціни за досить великої кількості моделей), ця альтернатива виключається. Таким чином, фінальний вибір залишається за моделлю SAMSUNG як такою, що забезпечує найкраще співвідношення характеристик і ціни.

За результатами проведеного аналізу зібрані та оброблені дані доцільно узагальнити у вигляді таблиць, що дозволяють наочно представити як перелік найпопулярніших моделей, так і усереднені характеристики телефонів кожної цінової категорії. Таке табличне представлення підсумовує висновки розділів і слугує основою для остаточної рекомендації.

П'ять найпопулярніших брендів.

Враховуючи популярність кожної моделі (кількість представлених на ринку позицій), вибір було звужено до п'яти найпопулярніших брендів. Для кожного з них у таблиці 3.2 наведено середню ціну, обчислену за реальними (ненульовими) цінами наявних товарів.

Таблиця 3.2 – П'ять найпопулярніших брендів та їх середня ціна

№	Бренд	Середня ціна, грн
1	APPLE	48 919,27
2	SAMSUNG	22 506,96
3	XIAOMI	10 413,45
4	REDMI	6 920,57
5	TECNO	5 250,97

З таблиці 4.2 видно, що навіть серед п'яти найпопулярніших брендів спостерігається значний розрив у середній ціні — від приблизно 5,3 тис. грн у бренду TECNO до майже 49 тис. грн у бренду APPLE, тобто більш ніж у дев'ять разів. Бренд APPLE має найвищу середню ціну, незважаючи на велику

кількість моделей, що свідчить про вагому складову вартості, пов'язану саме з брендом, а не лише з технічними характеристиками. SAMSUNG посідає друге місце за ціною, поєднуючи при цьому найбільшу кількість моделей на ринку, тоді як бренди XIAOMI, REDMI і TECNO формують доступніший ціновий діапазон. Саме на цих п'яти брендах зосереджено подальше навчання та аналіз класифікаційних моделей.

Характеристики цінових сегментів.

Як показав кластерний аналіз методом K-середніх (підрозділ 4.3), ціна будь-якої моделі сильно залежить від роздільної здатності камери, обсягу внутрішньої пам'яті та бренду. На основі поділу на три кластери всі моделі було розподілено на три цінові сегменти: бюджетний, середній та дорогий. Для кожного сегмента обчислено усереднені значення основних характеристик, наведені в таблицях 4.3 – 4.5. Слід зазначити, що бренд та операційна система подані в кодованому (числовому) вигляді, який використовувався під час навчання моделі: операційна система 0 відповідає Android, 1 – iOS, тому проміжні значення відображають частку iOS-пристроїв у сегменті.

Бюджетний сегмент (таблиця 3.3) є найчисленнішим і характеризується найнижчою середньою ціною. До нього входять переважно Android-пристрої початкового та середнього рівня.

Таблиця 3.3 – Усереднені характеристики бюджетного сегмента

Характеристика	**Середнє значення**
Середня ціна, грн	8 584,57
Діагональ дисплея, дюйми	6,56
Кількість ядер	7,64
Оперативна пам'ять, ГБ	4,50
Внутрішня пам'ять, ГБ	96,38
Камера, Мп	47,89
Операційна система (0 — Android)	0,01

Бюджетний сегмент відрізняється помірними значеннями оперативної (близько 4,5 ГБ) та внутрішньої (близько 96 ГБ) пам'яті за відносно високої роздільної здатності камери (майже 48 Мп). Показник операційної системи, близький до нуля, свідчить про те, що сегмент майже повністю складається з пристроїв на Android. Висока кількість ядер процесора (понад 7) пояснюється поширеністю восьмиядерних чипсетів навіть у недорогих моделях.

Дорогий сегмент (таблиця 3.4) об'єднує пристрої з найбільшим обсягом внутрішньої пам'яті та помітною часткою iOS-моделей.

Таблиця 3.4 – Усереднені характеристики дорогого сегмента

Характеристика	**Середнє значення**
Середня ціна, грн	40 046,32
Діагональ дисплея, дюйми	6,13
Кількість ядер	6,62
Оперативна пам'ять, ГБ	5,95
Внутрішня пам'ять, ГБ	236,97
Камера, Мп	25,03
Операційна система (1 — iOS)	0,69

Дорогий сегмент характеризується значно більшим обсягом внутрішньої пам'яті (близько 237 ГБ) та найвищою серед усіх сегментів часткою iOS-пристроїв (показник операційної системи 0,69 означає, що майже 70 % моделей сегмента працюють під управлінням iOS). Цікаво, що середня роздільна здатність камери в цьому сегменті нижча (близько 25 Мп), ніж у бюджетному, що пояснюється маркетинговою політикою деяких преміальних виробників, які роблять акцент не на номінальній кількості мегапікселів, а на якості оптики та обробки зображення. Це підтверджує висновок про те, що значну частку вартості преміальних пристроїв формує бренд.

Окремий кластер (таблиця 3.5) об'єднує невелику групу найдорожчих флагманських пристроїв із максимальними характеристиками.

Таблиця 3.5 – Усереднені характеристики преміального (флагманського) сегмента

Характеристика	Середнє значення
Середня ціна, грн	61 905,92
Діагональ дисплея, дюйми	6,57
Кількість ядер	6,52
Оперативна пам'ять, ГБ	7,48
Внутрішня пам'ять, ГБ	317,60
Камера, Мп	62,00
Операційна система (1 — iOS)	0,74

Цей сегмент є найвищим за ціною (близько 62 тис. грн) і об'єднує флагманські моделі з максимальними характеристиками: найбільшим обсягом оперативної (близько 7,5 ГБ) та внутрішньої (понад 317 ГБ) пам'яті, найвищою роздільною здатністю камери (62 Мп) і значною часткою iOS-пристроїв (74 %). Саме поєднання найкращих технічних параметрів із преміальним позиціонуванням брендів зумовлює найвищу вартість пристроїв цієї групи.

Зведене порівняння сегментів.

Для зручності зіставлення усереднені характеристики всіх трьох цінових сегментів зведено в єдину порівняльну таблицю 3.6.

Таблиця 3.6 – Зведене порівняння цінових сегментів

Характеристика	Бюджетний	Дорогий	Преміальний
Середня ціна, грн	8 584,57	40 046,32	61 905,92
Діагональ дисплея, дюйми	6,56	6,13	6,57
Кількість ядер	7,64	6,62	6,52
Оперативна пам'ять, ГБ	4,50	5,95	7,48
Внутрішня пам'ять, ГБ	96,38	236,97	317,60
Камера, Мп	47,89	25,03	62,00
Частка iOS-пристроїв	0,01	0,69	0,74

Зведена таблиця 3.6 наочно демонструє основну закономірність, виявлену в роботі: при переході від бюджетного до преміального сегмента послідовно зростають обсяг оперативної та внутрішньої пам'яті, а також частка iOS-пристроїв. Водночас такі характеристики, як діагональ дисплея та кількість ядер, залишаються відносно стабільними в усіх сегментах, а отже, не є визначальними чинниками ціноутворення. Роздільна здатність камери не має монотонної залежності від ціни, що ще раз підкреслює: вартість пристрою формується комплексно, з вагомим внеском бренду й операційної системи, а не лише номінальних технічних параметрів.

Узагальнюючи, можна зробити висновок, що ключовими чинниками, які найсильніше впливають на ціну мобільного телефона, є обсяг внутрішньої пам'яті, бренд та операційна система. Саме врахування цих чинників дозволило побудованій моделі Random Forest з високою точністю розпізнавати модель пристрою за його характеристиками та сформувати обґрунтовану рекомендацію щодо оптимального вибору.

3.6.7 Інтерпретація результатів та обмеження моделі

Отримані результати дозволяють зробити декілька важливих висновків щодо чинників ціноутворення на ринку смартфонів. Висока точність класифікації моделі за її характеристиками й ціною свідчить про те, що між технічними параметрами пристрою та його брендом існує стійкий зв'язок. Найбільший вплив на ціну, як показав кластерний аналіз, чинять обсяг внутрішньої пам'яті, роздільна здатність камери та бренд – саме за цими ознаками найчіткіше розділяються цінові сегменти.

Особливо показовим є випадок бренду Apple: моделі цього виробника утворюють окремий ціновий кластер із найвищою середньою ціною, що значною мірою пояснюється надбавкою за бренд, а не лише технічними

перевагами. Це підтверджує доцільність виключення цієї альтернативи при розв'язанні задачі максимізації користі за ціну.

Водночас слід враховувати обмеження побудованої моделі. По-перше, дані зібрано з одного інтернет-магазину в конкретний момент часу, тому результати відображають поточну ринкову ситуацію саме цього ресурсу й можуть змінюватися. По-друге, помітне зниження точності на імітованій вибірці (з 97,59 % до 79,47 %) вказує на те, що штучно відновлені ціни не повністю відтворюють реальні закономірності, а отже, модель найнадійніше працює саме на фактичних, а не змодельованих даних. По-третє, набір ознак обмежений тими характеристиками, що доступні в каталозі; врахування додаткових параметрів (рік випуску, ємність акумулятора, відгуки користувачів) потенційно могло б підвищити якість прогнозу.

Незважаючи на ці обмеження, побудована система демонструє практичну придатність методів машинного навчання для автоматизованого аналізу товарного ринку та формування обґрунтованих рекомендацій, що відповідає поставленій меті роботи.

3.7 Висновки до розділу 3

У розділі детально описано роботу програмного коду на етапах збору, збереження та візуалізації даних. Реалізовано проєктування реляційної бази даних SQLite з двох пов'язаних таблиць, циклічний збір даних з усіх сторінок каталогу із вилученням характеристик товарів за допомогою CSS-селекторів BeautifulSoup, а також зчитування об'єднаних даних у датафрейм pandas. Засобами matplotlib і seaborn побудовано візуалізації, що дозволили проаналізувати цінове позиціонування й популярність брендів та обґрунтовано звузити подальший аналіз до п'яти найпопулярніших моделей.

Також у розділі описано побудову, навчання та порівняння класифікаційних моделей машинного навчання. Виконано попередню обробку та кодування даних, кластеризацію моделей на цінові сегменти методом K-

середніх, розбиття вибірки на навчальну й тестову та навчання чотирьох класифікаторів. Найвищу точність (97,59 %) показав алгоритм Random Forest, що підтверджує висновки теоретичного аналізу про переваги деревоподібних ансамблевих методів для даних зі складною структурою. На основі найкращої моделі сформовано обґрунтовану рекомендацію щодо вибору оптимальної моделі смартфона.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

В умовах прискореного розвитку інформаційних технологій, професії у сфері ІТ займають ключове місце в економічній структурі та соціальному устрої. Відповідно, набуває актуальності забезпечення адекватних умов праці та охорони здоров'я для спеціалістів цієї галузі. Попри загальноприйняту думку про порівняно низький ризик в роботі програмістів порівняно з іншими професіями, ця сфера містить специфічні ризики та особливості, що вимагають індивідуального підходу до питань охорони праці.

Аналіз робочого середовища ІТ-спеціалістів показує, що, незважаючи на зовнішню зручність, існують недоліки з точки зору ергономіки, психологічного навантаження та інших важливих аспектів. У цьому контексті важливим є вивчення основних аспектів охорони праці програмістів, аналіз потенційних ризиків та розробка рекомендацій щодо оптимізації умов праці для фахівців у галузі інформаційних технологій.

4.1 Питання щодо охорони праці

Фактори трудового середовища можуть істотно впливати на здоров'я та працездатність розробника програм. Неправильно організоване робоче місце може викликати проблеми із хребтом, шиєю, зап'ястям та іншими частинами тіла. Довгий час роботи за комп'ютером може призвести до тунельного синдрому зап'ястного каналу.

4.1.1 Ергономіка

У сучасних умовах трудової діяльності значуще місце займає організація робочого місця користувачів персональних комп'ютерів. Законодавство України наголошує на необхідності забезпечення безпечних та комфортних умов праці. Зокрема, відповідно до КЗпП та Закону України "Про охорону

праці", роботодавці мають обов'язки стосовно забезпечення працівників належними умовами.

При організації робочих місць із персональними комп'ютерами важливо забезпечити відстані між боковими частинами столів не менше ніж 1,2 м, а також враховувати, що відстань між задньою частиною одного комп'ютера та екраном іншого має бути 2,5 м. Структура робочого столу повинна бути розроблена відповідно до ергономічних стандартів, дозволяючи ефективно розташовувати необхідне обладнання, таке як дисплей, клавіатура, принтер, а також робочі документи.

Щодо параметрів робочого столу: його висота повинна знаходитися у діапазоні від 680 до 800 мм. Що стосується ширини та глибини, то вони повинні бути такими, щоб працівник міг з легкістю працювати в зоні доступності рук (для цього рекомендовані показники: ширини від 600 до 1400 мм, глибини від 800 до 1000 мм). Також необхідно передбачити комфортний простір для ніг користувача: висота – мінімум 600 мм, ширина – мінімум 500 мм, глибина на рівні колін – 450 мм і на рівні витягнутої ноги – не менше 650 мм.

Стілець на робочому місці повинен мати підйомно-поворотні характеристики, можливість регулювання по висоті, куту нахилу сидушки і спинки, а також відстані від спинки до зовнішнього краю сидіння. Поверхня сидіння має бути рівною, а її зовнішній край має мати округлу форму. Налаштування по кожному параметру повинно бути індивідуальним, інтуїтивним та надійно фіксуватися.

Інтервал регулювання частин стільця: для лінійних розмірів – 15-20 мм, для кутових 2-5°. Зусилля для регулювання не має перевищувати 20 Н. Висота сидіння повинна бути від 400 до 500 мм, а її ширина і глибина – не менше 400 мм. Сидіння має мати можливість нахилу до 15° вперед і до 5° назад. Висота спинки стільця – 300±20 мм, ширина – не менше 380 мм, радіус кривизни горизонтально – 400 мм. Кут нахилу спинки може регулюватися в діапазоні 1-

30° від вертикального положення. Відстань від спинки до сидіння – від 260 до 400 мм. Для зменшення напруження в руках рекомендується використовувати підлокітники довжиною від 250 мм, шириною 50-70 мм, що налаштовуються по висоті від 230 до 260 мм та по відстані між ними від 350 до 500 мм. Матеріал сидіння і спинки має бути напівтвердим, антиковзним, що пропускає повітря і легко миється, а також не збирає статичний заряд.

Робоча зона повинна бути обладнана підніжкою шириною мінімум 300 мм, глибиною не менше 400 мм, з можливістю регулювання по висоті до 150 мм і нахилом до 20°. На підніжці має бути рельєфна поверхня і маленький борт по зовнішньому краю висотою 10 мм. Екран комп'ютера слід розміщувати на оптимальній відстані від користувача, що варіюється в межах 500-700 мм, але не ближче ніж 500 мм, з урахуванням легкості читання тексту і зображень. Екран повинен бути розташований так, щоб забезпечити комфорт при спостереженні, кутом у 30° від вертикалі.

4.1.2 Освітлення

Ефективне та грамотне виробниче світло підвищує якість зорової діяльності, зменшує втомленість, стимулює зростання продуктивності, сприяє комфортному робочому середовищу, додаючи спокій та позитив працівникам, а також підсилює безпеку роботи, зменшуючи ризик травм. Недостатнє або надто яскраве освітлення може спричинити зорове перевтомлення і головний біль.

Недолік світла може призводити до перевантаження очей, зниження уваги та передчасної втоми. Занадто яскраве світло викликає сліпоту, незадоволення та відчуття дискомфорту в очах. Неправильне розташування світла може утворювати відблиски, контрастні тіні та дезорієнтувати працівника. Ці фактори можуть спричинити нещасний випадок або професійні захворювання, тому важливо правильно розраховувати інтенсивність світла.

Розрізняють три типи світла - природне, штучне та комбіноване.

Природне світло – це варіант освітлення, при якому денне світло проникає через вікна або інші прозорі елементи приміщення. Інтенсивність природного світла може сильно коливатися в залежності від доби, сезону та інших факторів. Штучне світло використовують у вечірній час або коли природне світло недостатнє. Якщо природне світло доповнюється штучним, таке освітлення називають змішаним.

За своїм призначенням штучне світло може бути робочим, евакуаційним, аварійним чи охоронним. Робоче світло поділяється на загальне та місцеве. При загальному освітленні світильники розташовані рівномірно по приміщенню. У системі комбінованого освітлення до загального додається додаткове місцеве світло.

Згідно норм "ДБН В.2.5-28:2018 Природне і штучне освітлення", в комп'ютерних залах слід застосовувати комбінований тип освітлення. Для виконання робіт високої зорової точності (об'єкт розрізнення 0,3 ... 0,5 мм) інтенсивність природного світла має бути не менше 1,5%. Для робіт середньої точності (об'єкт розрізнення 0,5 ... 1,0 мм) - не менше 1,0%. Як джерела штучного світла часто використовують лампи типу ЛБ або ДРЛ, об'єднані в світильниках над робочими столами.

Вимоги до освітленості для робочих місць з комп'ютерами такі: при високій точності зорової роботи – 300 лк (загальне) та 750 лк (комбіноване); при середній точності – 200 та 300 лк відповідно.

Також важливо, щоб усе поле зору було якісно освітлене. Світлова інтенсивність приміщення та яскравість екрану комп'ютера повинні бути близькими, адже надлишкове світло може збільшувати втомленість очей.

4.1.3 Параметри мікроклімату

Для створення комфортних умов праці в приміщеннях з комп'ютерною технікою важливо дотримуватися певних стандартів мікроклімату. В таблиці 4.1 наведені параметри мікроклімату для таких приміщень

Таблиця 4.1 – Параметри мікроклімату для приміщень, де встановлені комп'ютери

Параметри мікроклімату	Літній період	Зимовий період
Температура повітря, °С	20 – 24	22 – 24
Відносна вологість, %	40 – 60	40 – 60
Швидкість руху повітря, м/с	0,1 – 0,2	0,1 – 0,2
Рівень шуму, дБ	до 50	До 50

4.1.4 Емоційна психогігієна

У сфері ІТ охорона праці включає не лише фізичне, але й психічне здоров'я працівників. Фактори, що впливають на психіку працівників у галузі інформаційних технологій, є різноманітними та мають велике значення у створенні здорового робочого середовища.

Таблиця 4.2 – Допустимі значення параметрів неіонізуючого електромагнітного випромінювання

Найменування параметра	Допустимі значення
Напруженість електричної складової електромагнітного поля на відстані 50см від поверхні монітора	10 В / м
Напруженість магнітної складової електромагнітного поля на відстані 50см від поверхні монітора	0,3 А / м
Напруженість електростатичного поля	20кВ / м

До них належать:

– тривалість та інтенсивність робочого дня: довгі години роботи без відпочинку можуть призвести до перевтоми та стресу, впливаючи на психічне здоров'я;

– терміни виконання завдань і робочий тиск: нереалістичні терміни виконання завдань або надмірний робочий тиск можуть викликати стрес та тривогу;

– міжособистісні взаємодії: конфлікти в команді або нездорова робоча атмосфера можуть спричиняти емоційне вигорання;

– робота в умовах ізоляції або віддалена робота: відсутність соціальної взаємодії або підтримки може вплинути на почуття ізоляції та самотності;

– work-life баланс : нездатність знайти баланс між роботою та особистим життям може призвести до стресу та емоційного вигорання;

– значне розумове навантаження: постійна потреба в освоєнні нових технологій та адаптації до змін може бути стресовою;

– відсутність автономії або контролю над роботою: обмежений контроль над робочими процесами та відсутність автономії можуть викликати почуття безпорадності та фрустрації.

Для ефективної профілактики психоемоційних проблем, пов'язаних з роботою в IT-галузі, необхідно розробити комплексний підхід, який враховує різноманітність факторів, що впливають на психічне здоров'я працівників. Значущість цього підходу полягає в його спрямованості на зменшення стресорів у робочому середовищі та підвищення ресурсів для психічного відновлення.

Першочергово, акцентується увага на регулюванні тривалості робочого дня та інтенсивності роботи. Це включає розробку ефективних графіків роботи, що передбачають достатні перерви для відновлення, та уникнення перевантаження завданнями. Крім того, розглядається важливість балансу між роботою та особистим життям, який може бути підтриманий через гнучкі робочі години та можливість дистанційної роботи.

Ще одним важливим аспектом є оптимізація робочого середовища з точки зору ергономіки. Покращення умов роботи на робочому місці, включаючи забезпечення якісного обладнання та комфортних умов, сприятиме зниженню фізичного та психологічного дискомфорту.

Ключову роль відіграє також розвиток корпоративної культури, яка підтримує психологічне благополуччя. Це передбачає створення відкритого,

підтримуючого співробітництва та комунікації середовища, заохочення взаємодопомоги між колегами, та реалізацію програм з управління стресом.

Для забезпечення довгострокового психічного благополуччя, розглядається імплементація регулярних тренінгів та навчальних програм, спрямованих на розвиток навичок управління стресом та підвищення особистісної стійкості. Також важливим є застосування систематичного моніторингу психічного стану працівників з використанням психометричних інструментів для раннього виявлення потенційних проблем.

Розуміння та належне врахування цих факторів є ключовим для забезпечення психічного благополуччя працівників у галузі ІТ, що, у свою чергу, позитивно впливає на їх продуктивність та загальне задоволення роботою.

4.2 Питання щодо безпеки в надзвичайних ситуаціях

В умовах швидкого розвитку інформаційних технологій та зростання залежності бізнес-процесів від ІТ-систем, питання безпеки в сфері ІТ набуває особливої актуальності. Розглядаючи безпеку праці в ІТ, особливу увагу необхідно приділити розробці та імплементації процедур реагування на надзвичайні ситуації, що включають технічні збої, кібератаки, витік конфіденційної інформації, фізичні загрози обладнанню та перебої в електропостачанні, а також інші аварійні стани, здатні порушити звичний хід робочих процесів.

Відповідальність за реалізацію ефективної системи безпеки лежить на ІТ-спеціалістах, керівництві компаній та інших зацікавлених сторонах. Необхідно забезпечити встановлення чітких правил та інструкцій, які регламентують дії персоналу у випадку виникнення надзвичайних ситуацій. Це включає визначення швидких комунікаційних каналів, розробку планів відновлення роботи систем, а також проведення регулярних тренінгів та інструктажів з охорони праці для всіх працівників.

Плани евакуації та реагування мають бути адаптовані до можливих IT-ризиків. Вони повинні включати дії для захисту життя та здоров'я працівників, а також процедури збереження даних та обладнання. Плани мають бути детальними та зрозумілими для кожного члена команди, з чіткими інструкціями щодо дій в кризових ситуаціях.

Регулярні тренінги з охорони праці, які охоплюють надзвичайні ситуації в IT, є необхідними для підготовки персоналу до ефективного реагування на інциденти. Навчальні програми повинні включати імітації надзвичайних ситуацій, навчання з використанням спеціалізованого обладнання та вивчення найкращих практик у сфері кібербезпеки. Систематичний аналіз ризиків є важливим для ідентифікації потенційних вразливостей системи та розробки відповідних заходів запобігання. Встановлення систем моніторингу, які можуть виявити ознаки надзвичайних ситуацій на ранніх стадіях, допомагає зменшити потенційні збитки та забезпечити оперативне реагування.

Після кожної надзвичайної ситуації необхідно проводити детальний аналіз її причин, ефективності дій персоналу та наявних процедур реагування. На основі цього аналізу слід вносити корективи у плани надзвичайних дій, поліпшувати процедури безпеки та організовувати додаткове навчання. Ефективне управління надзвичайними ситуаціями в IT-сфері вимагає всебічного підходу, який поєднує в собі як технічні, так і організаційні аспекти. Відповідальність за це лежить на всіх рівнях організації, починаючи з IT-фахівців і закінчуючи вищим керівництвом. Завдяки встановленню та дотриманню відповідних процедур можна мінімізувати ризики та забезпечити безперебійну роботу в умовах, що постійно змінюються.

ВИСНОВКИ

У кваліфікаційній роботі розв'язано актуальну задачу розробки програмного засобу для аналізу ринку мобільних телефонів із застосуванням методів машинного навчання. Поставлену мету досягнуто, а всі сформульовані завдання – виконано.

Основні результати роботи такі:

1) Проаналізовано предметну область та обґрунтовано актуальність застосування методів машинного навчання для аналізу ринку смартфонів в умовах великого обсягу пропозицій та складних взаємозв'язків між характеристиками й ціною.

2) Досліджено теоретичні засади використаних моделей машинного навчання – методу кластеризації K-середніх, дерева рішень, логістичної регресії, випадкового лісу, градієнтного бустингу XGBoost та голосувального ансамблю, а також метрик оцінювання якості класифікації.

3) Проведено порівняння інструментів доступу до веб-ресурсу – бібліотеки BeautifulSoup та фреймворку Selenium – за критеріями швидкодії, споживання ресурсів і складності реалізації. Оскільки цільовий каталог повертає всі дані у статичній HTML-розмітці, для реалізації обрано BeautifulSoup як найбільш ефективне та просте рішення.

4) Реалізовано повний конвеєр обробки даних: автоматизований збір даних з усіх сторінок каталогу, збереження у реляційній базі даних SQLite з двох пов'язаних таблиць, зчитування об'єднаних даних у датафрейм pandas та їх візуалізацію засобами matplotlib і seaborn.

5) Побудовано та порівняно кілька класифікаційних моделей. Найвищу точність показав алгоритм Random Forest – 97,59 % на реальній вибірці та 79,47 % на імітованій, що значно перевершує результат логістичної регресії (55,42 %).

6) На основі найкращої моделі сформовано обґрунтовану рекомендацію: за критерієм максимізації характеристик при мінімізації ціни оптимальним вибором є модель SAMSUNG.

Практична цінність роботи полягає в тому, що розроблений програмний засіб демонструє повний цикл побудови аналітичної системи – від автоматизованого збору сирих даних до формування обґрунтованої рекомендації – і може слугувати основою для систем підтримки прийняття рішень у сфері електронної комерції.

Перспективами подальшого розвитку є розширення джерел даних за рахунок кількох інтернет-магазинів, врахування динаміки зміни цін у часі, а також додавання вебінтерфейсу для зручної взаємодії користувача із системою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Курчак Ю. Р. Моделювання та аналіз системи рекомендацій на веб-сайті на основі теорії графів та статистичних алгоритмів : робота на здобуття кваліфікаційного ступеня магістра, спец. 122 – комп'ютерні науки / Ю. Р. Курчак ; наук. кер. І. О. Боднарчук. – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2025. – 83 с.

2. Багрій М. Т. Розробка рекомендаційної системи для інтернет-магазину книг : робота на здобуття кваліфікаційного ступеня бакалавра, спец. 122 – комп'ютерні науки / М. Т. Багрій ; наук. кер. С. В. Марценко. – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2025. – 64 с.

3. Небесний Р. М. Рекомендаційна система формування команд виконавців з відповідними фаховими компетентностями : дис. ... д-ра філософії : 122 / Р. М. Небесний. – Тернопіль, 2023. – 253 с.

4. Ржеуський А., Кунанець Н., Стахів М. Рекомендаційна система інформаційного обслуговування користувачів бібліотек // Матеріали V науково-технічної конференції «Інформаційні моделі, системи та технології», 1–2 лютого 2018 року. – Тернопіль : ТНТУ, 2018.

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

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

7. Müller A. C., Guido S. Introduction to Machine Learning with Python: A Guide for Data Scientists. Sebastopol : O'Reilly Media, 2016. 392 p.

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

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

10. Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York : ACM, 2016. P. 785–794.

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

12. VanderPlas J. Python Data Science Handbook: Essential Tools for Working with Data. Sebastopol : O'Reilly Media, 2016. 548 p.

13. McKinney W. Python for Data Analysis. 3rd ed. Sebastopol : O'Reilly Media, 2022. 579 p.

14. Mitchell R. Web Scraping with Python: Collecting More Data from the Modern Web. 2nd ed. Sebastopol : O'Reilly Media, 2018. 308 p.

15. Richardson L. Beautiful Soup Documentation. URL: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (дата звернення: 10.05.2024).

16. Selenium Documentation. URL: https://www.selenium.dev/documentation/ (дата звернення: 10.05.2024).

17. SQLite Documentation. URL: https://www.sqlite.org/docs.html (дата звернення: 12.05.2024).

18. Pandas Documentation. URL: https://pandas.pydata.org/docs/ (дата звернення: 12.05.2024).

19. Hunter J. D. Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering. 2007. Vol. 9, No. 3. P. 90–95.

20. Waskom M. L. seaborn: statistical data visualization. Journal of Open Source Software. 2021. Vol. 6, No. 60. P. 3021.

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

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

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

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

25. Мельник, А., & Дмитроца, Л. (2026). Методи та архітектурні підходи до автоматизації тестування мобільних і вебзастосунків. вимірювальна та обчислювальна техніка в технологічних процесах, (2), 74-81. https://doi.org/10.31891/2219-9365-2026-86-9
26. Melnyk, A., Dmytrotsa, L., Palka, O., Vasylenko, Y., & Klymuk, N. (2025). Dynamic test case prioritisation for mobile applications based on real user behaviour data. Proceedings of the CITI 2025: The 3rd International Workshop on Computer Information Technologies in Industry 4.0 (Ternopil, Ukraine, June 11-12, 2025). CEUR Workshop Proceedings (CEURWS.org). 2025. Vol-4057, pp. 179-188. URL: https://ceur-ws.org/Vol-4057/paper12.pdf
27. Formation of an IT Project Team in the Context of PMBOK Requirements / R. Nebesnyi, N. Kunanets, R. Vaskiv, N. Veretennikova. 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT). 2021. Vol. 2. P. 431–436. URL: https://doi.org/10.1109/CSIT52704.2021.9615291 (дата звернення: 17.06.2026).
28. Project of an intelligent recommender system for parking vehicles in smart cities / Y. Pankiv, N. Kunanets, O. Artemenko, N. Veretennikova, R. Nebesnyi. 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT). 2021. Vol. 2. P. 419–422. URL: https://doi.org/10.1109/CSIT52704.2021.9648687 (дата звернення: 17.06.2026).

ДОДАТКИ

ДОДАТОК А

Програмний код

Final project "Analyzing phone market"

Part 1: Getting, parsing and saving the data

1. Імпортуємо бібліотеку requests (для звернення до веб-ресурсів), beautifulSoup (для конверсії відповіді response в об'єкт html-soup), sqlite3 - (для роботи з sqlite), pandas - (для роботи з датафреймами).

```
import requests                     # бібліотека для здійснення запитів
from bs4 import BeautifulSoup       # для парсингу html
import sqlite3                      # для роботи з БД, зокрема SQlite
import pandas as pd                 # для роботи з датафреймами
```

2. У якості використовуваного веб-ресурсу приймемо сторінку з гаджетами. Звернемося до неї та отримаємо першу сторінку:

```
url_base = "https://can.ua/smartphones/c1538/" # початкова сторінка з данними
url_page="page=" # додаток до адреси для отримання сторінки

data = requests.get(url_base + url_page + str(1)) # отримання першої сторінки
```

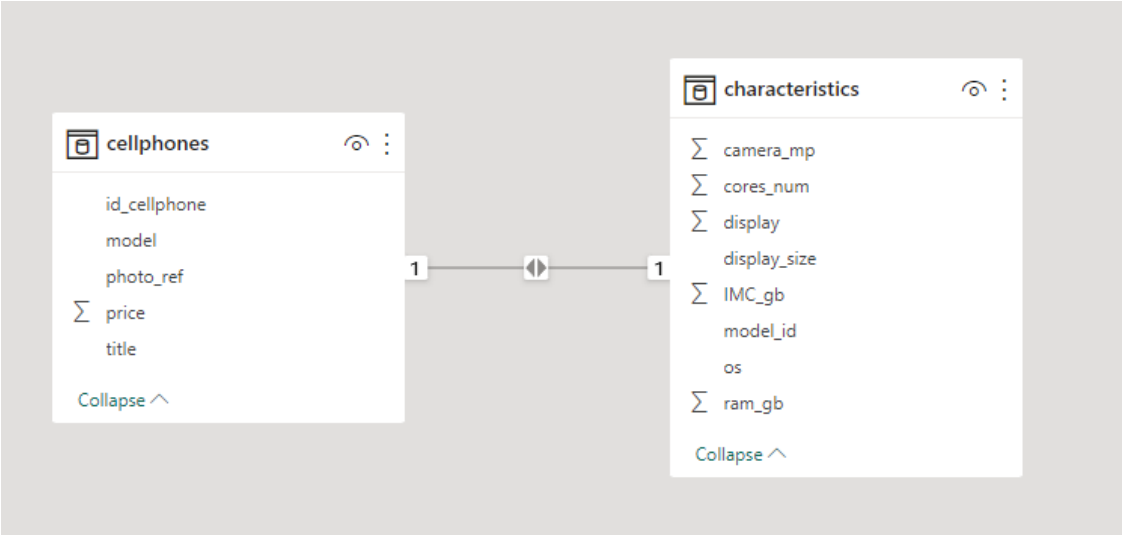
3. Якщо сайт дозволив отримати дані (статус код = 200), то конвертуємо дані в об'єкт BeautifulSoup для подальшої роботи і створюємо базу даних для запису отриманих даних. В іншому випадку - сповіщаємо, який статус код прийшов користувачу і завершуємо роботу програми:

```
if data.status_code == 200:
    print("Data was successfully retrieved (200)")

    soup = BeautifulSoup(data.text, "html.parser") # парсимо текст відповіді
в об'єкт ГарногоСупа

    db = sqlite3.connect("cellphones.db") # створили базу даних з назвою cell
phones, або підключились до існуючої (якщо вона була)
    cur = db.cursor() # отримали курсор (мідлуеар) для подальшого оперування
запитами

    # створюємо таблицю cellphones, у якій будемо зберігати:
        # id_cellphone - унікальний артикул товару з сайту
        # title - повна назва товару
        # photo_ref - посилання на головну фотографію товару
        # price - ціна товару
        # model - модель товару
    cur.execute('''CREATE TABLE IF NOT EXISTS cellphones(
        id_cellphone INT NOT NULL PRIMARY KEY,
        title VARCHAR(100) NOT NULL,
        photo_ref VARCHAR(255) NULL,
        price INT NOT NULL,
        model VARCHAR(100) NOT NULL ) ''')

    # створюємо таблицю characteristics, у якій будемо зберігати:
        # id_phone - унікальний ідентифікатор у таблиці, автоінкрементний
```

```
        # model_id - унікальний артикул товару з сайту, прив'язаний до таблиц
і cellphones
        # display - діагональ дісплею
        # display_size - розмір дісплею
        # cores_num - кількість ядер
        # ram_gb - оперативна пам'ять
        # IMC_gb - внутрішня пам'ять
        # camera_mp - мегапікселі камери
        # os - операційна система
    cur.execute('''CREATE TABLE IF NOT EXISTS characteristics(
        model_id INT NOT NULL PRIMARY KEY,
        display FLOAT NOT NULL,
        display_size VARCHAR(45) NOT NULL,
        cores_num INT NOT NULL,
        ram_gb INT NOT NULL,
        IMC_gb INT NOT NULL,
        camera_mp INT NOT NULL,
        os VARCHAR(45) NOT NULL,
        FOREIGN KEY(model_id) REFERENCES cellphones(id_cellphone))''')

    db.commit() # commit the transaction (saves changes in DB)
    cur.close() # позначаємо програмі кінець поточної операції з курсором, за
чиняючи його

else:
    print("Unable to retrieve the data. Response code = ", data.status_code)
```

```
Data was successfully retrieved (200)
```

Схема спроектованої бази даних з Power BI



4. Якщо статус код = 200, то отримуємо кількість сторінок як максимальний елемент списку сторінок на сайті. Розпочинаємо цикл по всім сторінкам для отримання даних. Створюємо БД та таблиці, записуємо в них отримані дані. Для подробиць, будь ласка, зверніться до коментарів у програмі:

```
if data.status_code == 200: # якщо сайт не проти
    print("Data was successfully retrieved (200)")

    # отримуємо кількість сторінок розділу
```

```
    pages_qty = int(soup.select_one("ul[name='paginator'] li:nth-last-child(2
)").get_text())
    # отримуємо label'и зі списку моделей сайту
    model_labels = soup.select("#desktop_sort-producer > ul > li > label")
    models = [] # список моделей з сайту (для подальшого їх аналізу у назві т
овару)

    for model in model_labels: # для кожної моделі зі списку
        models.append(model.get_text().split("(")[0].strip()) # парсимо лише
назву моделі у масив з моделями

    for i in range(1, pages_qty+1): # цикл по всім сторінкам
        print("Fulfilling page #", i, "... of", pages_qty) # виводимо номер п
оточної сторінки, щоб розуміти прогрес виконання програми
        if i != 1: # для першої сторінки нічого не робимо, тому що у якості т
естового запиту вона вже була отримана
            data = requests.get(url_base + url_page + str(i)) # а для інших с
торінок посилаємо запит на новую сторінку

        soup = BeautifulSoup(data.text, "html.parser") # парсимо текст відпов
іді в об'єкт ГарногоСупа

        page_products = soup.select(".catalog-tile li") # отримуємо всі товар
и з поточної (і) сторінки

        for product in page_products: # цикл по всіх продуктах поточної сторі
нки (за замовчуванням сайту 40 шт. на сторінці)
            id = product.select_one("div.info__code").get_text().strip()[5:]
# унікальний артикул товару з сайту
            title = product.select_one("a.info__name").get_text()  # повна на
зва товару
            price = 0 # за замовчуванням - ціна 0
            price_element = product.select_one("p.item__price") # елемент, з
якого читаємо ціну

            if price_element: # якщо такий елемент знайдено
                price = (price_element.get_text()[:-4]).replace(" ", '')  # ц
іна товару зберігається (відповідно, товар є в наявності)

            photo_ref = product.select_one("img")['src']  # посилання на фото
графію товару
            model = "" # модель товару, що обмежена значеннями

            for curr_model in models: # цикл по всім моделям
                if curr_model in title: # перевіряємо кожну і зупиняємося, ко
ли знайшли
                    model = curr_model # перед зупинкою зберігаємо значення,
що знайшли
                    break # сама зупинка

            # секція "додаткові параметри"
            display = product.select_one(".item__hidden > p >b:nth-child(1)")
.get_text()[:-2]  # діагональ дісплею
            display_size = product.select_one(".item__hidden > p >b:nth-child
(3)").get_text() # розмір дісплею
```

```
            cores_num = product.select_one(".item__hidden > p >b:nth-child(5)
").get_text() # кількість ядер
            ram = product.select_one(".item__hidden > p >b:nth-child(7)").get
_text()[:-3] # оперативна пам'ять
            imc = product.select_one(".item__hidden > p >b:nth-child(9)").get
_text()[:-3] # внутрішня пам'ять
            camera = product.select_one(".item__hidden > p >b:nth-child(11)")
.get_text()[:-3] # мегапікселі камери
            os = product.select_one(".item__hidden > p >b:nth-child(13)").get
_text() # операційна система

            cur = db.cursor() # оголошуємо курсор знову, позначаючи початок р
оботи із запитами
            # записали дані у першу таблицю
            try: # спробувати записати дані у БД
                cur.execute('INSERT INTO cellphones VALUES(?, ?, ?, ?, ?)', (
id,title, photo_ref, price,model))
                # записали дані у другу таблицю
                cur.execute('INSERT INTO characteristics VALUES(?, ?, ?, ?, ?
, ?, ?, ?)', (id, display,display_size,cores_num, ram, imc, camera, os))
            except sqlite3.IntegrityError as e: # якщо була помилка порушення
цілісності
                if 'UNIQUE constraint' in e.args[0]: # якщо дублюється унікал
ьний запис
                    print("Немає можливості додати запис(и), тому що вони пор
ушують унікальність коду БД. Спробуйте спочатку видалити ці записи або просто
видаліть БД та запустіть програму")
                else: # інша помилка
                    print("Error: ", e)

                break # при помилці не продовжувати парсинг сторінок і запис
у БД
            else: # якщо все пройшло без помилок
                db.commit() # зберігаємо зміни
            finally: # виконувати у будь-якому разі
                cur.close() # позначаємо кінець поточних транзакцій

    db.close() # завершуємо роботу з базою даних
    print("Done") # сповіщаємо користувача, що програма завершила роботу
else:
    print("Unable to retrieve data. Response code = ", data.status_code) # у
разі помилки із самого початку
```

```
Data was successfully retrieved (200)
Fulfilling page # 1 ... of 25
Fulfilling page # 2 ... of 25
Fulfilling page # 3 ... of 25
Fulfilling page # 4 ... of 25
Fulfilling page # 5 ... of 25
Fulfilling page # 6 ... of 25
Fulfilling page # 7 ... of 25
Fulfilling page # 8 ... of 25
Fulfilling page # 9 ... of 25
Fulfilling page # 10 ... of 25
```

```
Fulfilling page # 11 ... of 25
Fulfilling page # 12 ... of 25
Fulfilling page # 13 ... of 25
Fulfilling page # 14 ... of 25
Fulfilling page # 15 ... of 25
Fulfilling page # 16 ... of 25
Fulfilling page # 17 ... of 25
Fulfilling page # 18 ... of 25
Fulfilling page # 19 ... of 25
Fulfilling page # 20 ... of 25
Fulfilling page # 21 ... of 25
Fulfilling page # 22 ... of 25
Fulfilling page # 23 ... of 25
Fulfilling page # 24 ... of 25
Fulfilling page # 25 ... of 25
Done
```

5. Отримуємо дані із БД:

```
db = sqlite3.connect("cellphones.db") # підключаємося до БД cellphones.db
cur = db.cursor() # оголошуємо новий курсор

# за допомогою об'єднання таблиць отримуємо всі дані з БД
cur.execute('''SELECT id_cellphone, title, photo_ref, price, model, display,
display_size, cores_num, ram_gb, IMC_gb, camera_mp, os
               FROM cellphones
               LEFT JOIN characteristics ON cellphones.id_cellphone = charac
teristics.model_id''')
df = pd.DataFrame(cur.fetchall(), columns=["Артикул", "Назва", "Фото", "Ціна"
, "Модель", "Дисплей", "Розмір дисплею", "Кількість ядер", "Оперативна пам'ят
ь", "Внутрішня пам'ять", "Камера", "Операційка"])

cur.close() # кінець поточних транзакцій
db.close() # завершуємо роботу з базою даних

df.head()
```

```
   Артикул                                              Назва  \
0   345194                     Смартфон NOKIA G10 3/32GB Night
1   346475       Смартфон REDMI Note 10 Pro 6/128GB Onyx Gray
2   349393   Смартфон ONEPLUS Nord N100 4/64GB Midnight Frost
3   364817                 Смартфон VIVO Y31 4/64GB Ocean Blue
4   365575  Смартфон SAMSUNG Galaxy M52 6/128GB Light Blue...

                                                 Фото   Ціна   Модель  Дисплей
\
0  https://i.can.ua/images/244x190/goods/8341/834...   3999    NOKIA     6.50
1  https://i.can.ua/images/244x190/goods/8393/839...   9999    REDMI     6.67
2  https://i.can.ua/images/244x190/goods/8571/857...   6999  ONEPLUS     6.52
3  https://i.can.ua/images/244x190/goods/9284/928...   7173     VIVO     6.58
4  https://i.can.ua/images/244x190/goods/9327/932...  11790  SAMSUNG     6.70

         Розмір дисплею  Кількість ядер Оперативна пам'ять  Внутрішня пам'ять
\
0        1600x720 (HD+)               8                  3                 32
1  2400x1080 (Full HD+)               8                  6                128
```

```
2        1600x720 (HD+)              8                4                64
3  2400x1080 (Full HD+)              8                4                64
4  2400x1080 (Full HD+)              8                6               128

   Камера Операційка
0    13.0    Android
1   108.0    Android
2    13.0    Android
3    48.0    Android
4    64.0    Android
```

Part 2: Visualization

```
import matplotlib.pyplot as plt    # import the library for visualizing
import seaborn as sns              # for plots styling

display(df.describe()) # describe the data
df.isnull().values.any() # check for nulls (if any)
```

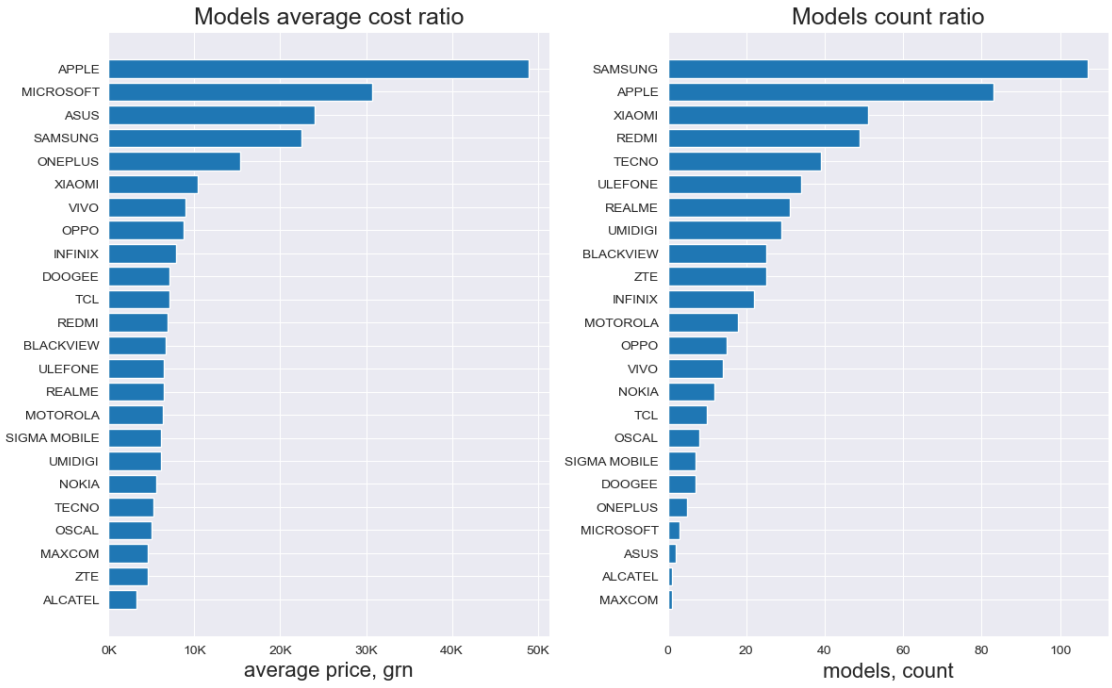
```
             Артикул          Ціна     Дисплей  Кількість ядер  \
count     982.000000    982.000000  982.000000      982.000000
mean   378000.014257   9626.064155    6.398656        7.303462
std     29910.138488  15827.538988    0.431751        1.346861
min     95294.000000      0.000000    4.700000        4.000000
25%    374083.250000      0.000000    6.367500        8.000000
50%    387305.500000   4684.500000    6.500000        8.000000
75%    394836.750000   8495.500000    6.600000        8.000000
max    406100.000000  79052.000000    8.300000        8.000000

       Внутрішня пам'ять      Камера
count          982.000000  982.000000
mean           119.845214   39.376171
std            102.782831   32.749700
min              1.000000    4.000000
25%             64.000000   13.000000
50%            128.000000   48.000000
75%            128.000000   50.000000
max            512.000000  200.000000

False
```

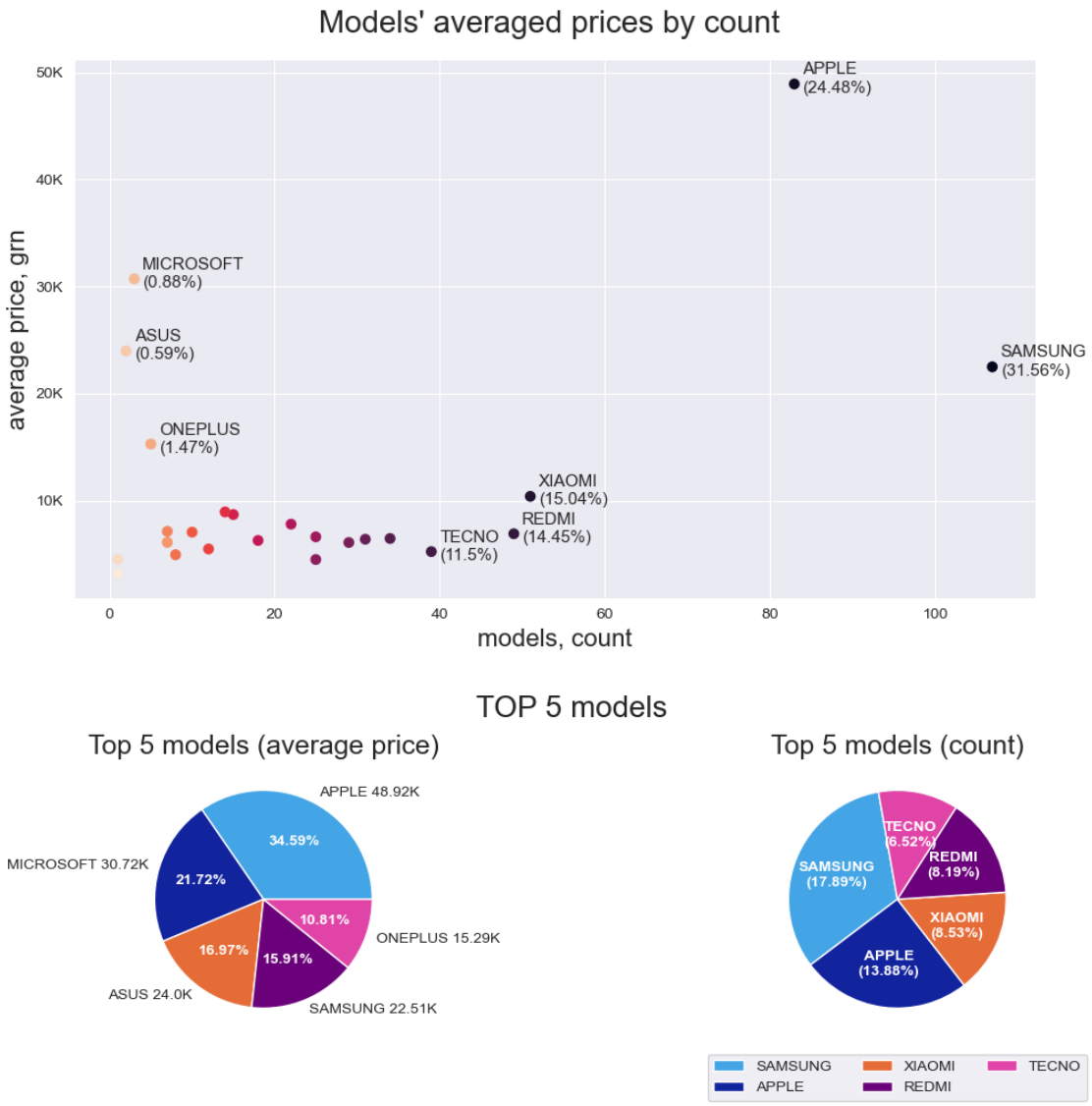
```
# function to get the string value percentage on the column
def get_percentage_str(data_frame_column, value, round_to):
    return str(round((value/sum(data_frame_column))*100,round_to))+"%"

sns.set_style('darkgrid') # dark grid style

df_vis = df[df['Ціна'] > 0] # not showing rows with 0 prices (out of stock)

avg_price_per_model = df_vis[['Модель','Ціна']].groupby('Модель', as_index=Fa
lse)['Ціна'].mean() # average price per model
avg_price_per_model = avg_price_per_model.sort_values('Ціна', ascending=False
) # sorting by price from biggest to lowest
count_per_model = df_vis['Модель'].value_counts() # models count per model
```

```
count_per_model = pd.DataFrame({'Модель': count_per_model.index, 'Середня кіл
ькість': count_per_model.values})
avg_price_count_per_model = pd.merge(avg_price_per_model, count_per_model, on
=['Модель', 'Модель'])
avg_price_count_per_model = avg_price_count_per_model.sort_values('Середня кі
лькість', ascending=False)


fig1 = plt.figure(figsize=(12,14)) # the first figure to plot on
ax1 = fig1.add_subplot(2,2,1) # add subplot to the fig1 (2 rows, 2 columns, 1
st plotting)
ax1.invert_yaxis() # invert y axis
ax1.xaxis.set_major_formatter(lambda x,p: str(int(x/1000))+"K") # custom form
at x axis labels

plt.barh(avg_price_per_model['Модель'].values, avg_price_per_model['Ціна'].va
lues) # horizontal bar chart model by price
plt.xlabel('average price, grn', fontsize=16) # x axis label
plt.title("Models average cost ratio", fontsize = 18) # chart title



ax2 = fig1.add_subplot(2,2,2) # add subplot to the fig1 (2 rows, 2 columns, 2
nd plotting)
ax2.invert_yaxis() # invert y axis
plt.barh(count_per_model['Модель'].values, count_per_model['Середня кількість
'].values) # horizontal bar chart model by count
plt.xlabel('models, count', fontsize=16) # x axis label
plt.title("Models count ratio", fontsize = 18) # chart title

fig1.tight_layout() # for the plots not to overlap



fig2 = plt.figure(figsize=(10,6)) # the second figure to plot on
fig2.suptitle("Models' averaged prices by count", fontsize = 20) # figure 2 t
itle
ax3 = fig2.add_subplot(1,1,1) # add subplot to the fig2 (1 row, 1 column, 1st
plotting)
ax3.yaxis.set_major_formatter(lambda x,p: str(int(x/1000))+"K") # custom form
at y axis labels
# scatter plot count by price
plt.scatter(avg_price_count_per_model['Середня кількість'].values, # x-values
: average count per model
            avg_price_count_per_model['Ціна'].values, # y-values: average pri
ce per model
            c=[i for i in range(1, len(count_per_model.values)+1)]) # separat
e color for each model
plt.xlabel('models, count', fontsize=16) # x axis label
plt.ylabel('average price, grn', fontsize=16) # y axis label

# filter models to get the most popular and/or expensive models
models_filtered = avg_price_count_per_model[((avg_price_count_per_model['Ціна
']>14000.0) & # prices bigger than 14K AND
                                            (avg_price_count_per_model['Серед
```

```
дня кількість']<20.0)) # count less that 20 OR
                                                | (avg_price_count_per_model['Сер
едня кількість']>37)] # count bigger than 37

for i, txt in enumerate(list(models_filtered['Модель'].values)): # for every
filtered model
    # render labels in the custom formatting: model_name+count_percentage
    ax3.annotate(txt + "\n(" + get_percentage_str(models_filtered['Середня кі
лькість'], models_filtered['Середня кількість'].values[i], 2) +")",
                 (models_filtered['Середня кількість'].values[i]+1, # label x
-val (+1 for better visual)
                  models_filtered['Ціна'].values[i]-800),  # label y-val (-80
0 for better visual)
                 size=11) # font-size

fig2.tight_layout() # for the plots not to overlap



fig3 = plt.figure(figsize=(12,10)) # the third figure to plot on
fig3.suptitle('TOP 5 models', fontsize=20) # figure 3 title
ax4 = fig3.add_subplot(3,2,1) # add subplot to the fig3 (3 rows, 2 columns, 1
st plotting)
_, __, texts = plt.pie(avg_price_per_model['Ціна'].head().values, # the first
5 most expensive average price values per model
                       colors=['#43A4E6', '#12239E', '#E66C37', '#6B007B', '#
E044A7'], # the colors for each model
                       autopct='%1.2f%%', # auto-estimation of percent in the
float formatting
                       #custom value format for labels: model_name+(price/100
0)K
                       labels=[k + " " + str(round(int(v)/1000,2))+"K" for k,
v in avg_price_per_model.head().values],
                       textprops={'fontsize':10}) # size of the labels

for text in texts: # for each of labels of the plt.pie
    text.set_horizontalalignment('center') # align labels center
    text.set_color('white') # font color
    text.set_weight('bold') # font weight

plt.title("Top 5 models (average price)", fontsize = 18) # the pie plot title



ax5 = fig3.add_subplot(3,2,2)  # add subplot to the fig3 (3 rows, 2 columns,
2nd plotting)
_, texts = plt.pie(count_per_model['Середня кількість'].head().values, # the
first 5 biggest average count values per model
                   colors=['#43A4E6', '#12239E', '#E66C37', '#6B007B', '#E044
A7'],  # the colors for each model
                   startangle=100, # starting angle for the plot to be shown
                   labeldistance=0.6, # distance between labels and the pie c
enter
                   # custom value format for labels: model_name+<br>+(percent
_from_the_whole)%
```

```
                    labels=[str(k)+"\n("+ get_percentage_str(count_per_model['
Середня кількість'], v, 2) +")"
                            for k,v in count_per_model.head().values],
                    textprops={'fontsize':10}) # size of the labels

for text in texts: # for each of labels of the plt.pie
    text.set_horizontalalignment('center')  # align labels center
    text.set_color('white') # font color
    text.set_weight('bold') # font weight

plt.title("Top 5 models (count)", fontsize = 18) # the pie plot title
# the pie plot legend
plt.legend(labels=count_per_model['Модель'].head().values, # labels are the f
ive top models
           loc='upper center', # legend location
           bbox_to_anchor=(0.5, -0.04), # the corner's location
           ncol=3) # number of columns

fig3.tight_layout() # for the plots not to overlap



plt.show() # show the plots
```





Part 3: Analyzing and model learning

Стоїть задача вибору телефону. Фінансів вистачає на найдорожчу модель у магазині, але доцільність і релевантність витрати стоїть на першому місці.
Тож, постає необхідність вирішення задачі максимізації користі телефону за його ціну.
Необхідними підзадачами до виконання є:

```
import random
# working with pseudo-random values
from sklearn.cluster import KMeans
# kmeans exclusive clustering alg-thm
from sklearn.linear_model import LogisticRegression
# logistic regression alg-thm
from sklearn import tree
# decision tree alg-thm
from sklearn.ensemble import RandomForestClassifier
# random forest alg-thm
from xgboost import XGBClassifier
# XGB alg-thm
from sklearn.metrics import confusion_matrix, classification_report, accuracy
```

```
_score   # analyzing learning results
from sklearn.model_selection import train_test_split
# splitting data for train/test
from sklearn.ensemble import VotingClassifier
# ensemble to use multiple alg-thms

display(df.head())
display(df.describe())
```

```
   Артикул                                          Назва  \
0   345194                  Смартфон NOKIA G10 3/32GB Night
1   346475       Смартфон REDMI Note 10 Pro 6/128GB Onyx Gray
2   349393   Смартфон ONEPLUS Nord N100 4/64GB Midnight Frost
3   364817               Смартфон VIVO Y31 4/64GB Ocean Blue
4   365575  Смартфон SAMSUNG Galaxy M52 6/128GB Light Blue...

                                                Фото   Ціна   Модель  Дисплей
\
0  https://i.can.ua/images/244x190/goods/8341/834...   3999    NOKIA     6.50
1  https://i.can.ua/images/244x190/goods/8393/839...   9999    REDMI     6.67
2  https://i.can.ua/images/244x190/goods/8571/857...   6999  ONEPLUS     6.52
3  https://i.can.ua/images/244x190/goods/9284/928...   7173     VIVO     6.58
4  https://i.can.ua/images/244x190/goods/9327/932...  11790  SAMSUNG     6.70

         Розмір дисплею  Кількість ядер Оперативна пам'ять  Внутрішня пам'ять
\
0        1600x720 (HD+)               8                  3                 32
1  2400x1080 (Full HD+)               8                  6                128
2        1600x720 (HD+)               8                  4                 64
3  2400x1080 (Full HD+)               8                  4                 64
4  2400x1080 (Full HD+)               8                  6                128

   Камера Операційка
0    13.0    Android
1   108.0    Android
2    13.0    Android
3    48.0    Android
4    64.0    Android

             Артикул          Ціна     Дисплей  Кількість ядер  \
count     982.000000    982.000000  982.000000      982.000000
mean   378000.014257   9626.064155    6.398656        7.303462
std     29910.138488  15827.538988    0.431751        1.346861
min     95294.000000      0.000000    4.700000        4.000000
25%    374083.250000      0.000000    6.367500        8.000000
50%    387305.500000   4684.500000    6.500000        8.000000
75%    394836.750000   8495.500000    6.600000        8.000000
max    406100.000000  79052.000000    8.300000        8.000000

       Внутрішня пам'ять      Камера
count          982.000000  982.000000
mean           119.845214   39.376171
std            102.782831   32.749700
min              1.000000    4.000000
25%             64.000000   13.000000
50%            128.000000   48.000000
```

```
75%           128.000000   50.000000
max           512.000000  200.000000
```

```
df = df.drop(['Артикул', 'Назва', 'Фото'], axis=1) # getting rid of useless c
olumns
 # splitting display size by "(" to standrdize it, removing whitespaces
df['Розмір дисплею'] = df['Розмір дисплею'].apply(lambda x: x.split('(')[0].s
trip())

display(df['Розмір дисплею'].unique()) # unique display sizes
df.head()
```

```
array(['1600x720', '2400x1080', '1650x720', '2712x1220', '2796x1290',
       '3088x1440', '2340x1080', '2408x1080', '2460x1080', '1640x720',
       '1280x600', '1440x720', '1280x720', '960x480', '1560x720',
       '2208x1768', '1920x1080', '1792x828', '2532x1170', '1520x720',
       '2700x1800', '3216x1440', '2688x1892', '1612x720', '2640x1080',
       '2412x1080', '2176x1812', '1014x480', '2560x1179', '2778x1284',
       '3200x1440', '960x540', '2160x1080', '2280x1080', '960x442',
       '2388x1080', '1480x720', '1334x750'], dtype=object)
```

```
    Ціна   Модель  Дисплей Розмір дисплею  Кількість ядер Оперативна пам'ять
\
0   3999    NOKIA     6.50       1600x720               8                   3
1   9999    REDMI     6.67      2400x1080               8                   6
2   6999  ONEPLUS     6.52       1600x720               8                   4
3   7173     VIVO     6.58      2400x1080               8                   4
4  11790  SAMSUNG     6.70      2400x1080               8                   6

   Внутрішня пам'ять  Камера Операційка
0                 32    13.0    Android
1                128   108.0    Android
2                 64    13.0    Android
3                 64    48.0    Android
4                128    64.0    Android
```

```
df = pd.get_dummies(df, columns=['Розмір дисплею']) # divide 'Розмір дисплею'
column into separate columns

display(df.head())
display(df['Операційка'].unique()) # unique values of OSes
```

```
    Ціна   Модель  Дисплей  Кількість ядер Оперативна пам'ять  \
0   3999    NOKIA     6.50               8                   3
1   9999    REDMI     6.67               8                   6
2   6999  ONEPLUS     6.52               8                   4
3   7173     VIVO     6.58               8                   4
4  11790  SAMSUNG     6.70               8                   6

   Внутрішня пам'ять  Камера Операційка  Розмір дисплею_1014x480  \
0                 32    13.0    Android                        0
1                128   108.0    Android                        0
2                 64    13.0    Android                        0
3                 64    48.0    Android                        0
4                128    64.0    Android                        0
```

```
   Розмір дисплею_1280x600  ...  Розмір дисплею_2700x1800  \
0                        0  ...                         0
1                        0  ...                         0
2                        0  ...                         0
3                        0  ...                         0
4                        0  ...                         0

   Розмір дисплею_2712x1220  Розмір дисплею_2778x1284  \
0                         0                         0
1                         0                         0
2                         0                         0
3                         0                         0
4                         0                         0

   Розмір дисплею_2796x1290  Розмір дисплею_3088x1440  \
0                         0                         0
1                         0                         0
2                         0                         0
3                         0                         0
4                         0                         0

   Розмір дисплею_3200x1440  Розмір дисплею_3216x1440  Розмір дисплею_960x442
\
0                         0                         0                       0
1                         0                         0                       0
2                         0                         0                       0
3                         0                         0                       0
4                         0                         0                       0

   Розмір дисплею_960x480  Розмір дисплею_960x540
0                       0                       0
1                       0                       0
2                       0                       0
3                       0                       0
4                       0                       0

[5 rows x 46 columns]
```

```
array(['Android', 'iOS',
       'Android (без поддержки Google Services та Play Market)'],
      dtype=object)
```

```
df['Операційка'] = df['Операційка'].apply(lambda x: 0 if x == 'Android (без п
оддержки Google Services та Play Market)' or x == 'Android' else (1 if x == '
iOS' else 'unfamiliar model'))

display(df['Операційка'].unique())
display(df.head())
display(df.describe())
```

```
array([0, 1], dtype=int64)
```

```
    Ціна   Модель  Дисплей  Кількість ядер Оперативна пам'ять  \
0   3999    NOKIA     6.50               8                  3
1   9999    REDMI     6.67               8                  6
2   6999  ONEPLUS     6.52               8                  4
```

```
3   7173     VIVO     6.58               8                  4
4  11790  SAMSUNG     6.70               8                  6

   Внутрішня пам'ять  Камера  Операційка  Розмір дисплею_1014x480  \
0                 32    13.0           0                        0
1                128   108.0           0                        0
2                 64    13.0           0                        0
3                 64    48.0           0                        0
4                128    64.0           0                        0

   Розмір дисплею_1280x600  ...  Розмір дисплею_2700x1800  \
0                        0  ...                         0
1                        0  ...                         0
2                        0  ...                         0
3                        0  ...                         0
4                        0  ...                         0

   Розмір дисплею_2712x1220  Розмір дисплею_2778x1284  \
0                         0                         0
1                         0                         0
2                         0                         0
3                         0                         0
4                         0                         0

   Розмір дисплею_2796x1290  Розмір дисплею_3088x1440  \
0                         0                         0
1                         0                         0
2                         0                         0
3                         0                         0
4                         0                         0

   Розмір дисплею_3200x1440  Розмір дисплею_3216x1440  Розмір дисплею_960x442
\
0                         0                         0                       0
1                         0                         0                       0
2                         0                         0                       0
3                         0                         0                       0
4                         0                         0                       0

   Розмір дисплею_960x480  Розмір дисплею_960x540
0                       0                       0
1                       0                       0
2                       0                       0
3                       0                       0
4                       0                       0

[5 rows x 46 columns]

               Ціна     Дисплей  Кількість ядер  Внутрішня пам'ять  \
count    982.000000  982.000000      982.000000         982.000000
mean    9626.064155    6.398656        7.303462         119.845214
std    15827.538988    0.431751        1.346861         102.782831
min        0.000000    4.700000        4.000000           1.000000
25%        0.000000    6.367500        8.000000          64.000000
50%     4684.500000    6.500000        8.000000         128.000000
```

```
75%     8495.500000    6.600000        8.000000         128.000000
max    79052.000000    8.300000        8.000000         512.000000

           Камера  Операційка  Розмір дисплею_1014x480  \
count  982.000000  982.000000               982.000000
mean    39.376171    0.116090                 0.001018
std     32.749700    0.320496                 0.031911
min      4.000000    0.000000                 0.000000
25%     13.000000    0.000000                 0.000000
50%     48.000000    0.000000                 0.000000
75%     50.000000    0.000000                 0.000000
max    200.000000    1.000000                 1.000000

       Розмір дисплею_1280x600  Розмір дисплею_1280x720  \
count               982.000000               982.000000
mean                  0.004073                 0.017312
std                   0.063725                 0.130496
min                   0.000000                 0.000000
25%                   0.000000                 0.000000
50%                   0.000000                 0.000000
75%                   0.000000                 0.000000
max                   1.000000                 1.000000

       Розмір дисплею_1334x750  ...  Розмір дисплею_2700x1800  \
count               982.000000  ...               982.000000
mean                  0.002037  ...                 0.002037
std                   0.045106  ...                 0.045106
min                   0.000000  ...                 0.000000
25%                   0.000000  ...                 0.000000
50%                   0.000000  ...                 0.000000
75%                   0.000000  ...                 0.000000
max                   1.000000  ...                 1.000000

       Розмір дисплею_2712x1220  Розмір дисплею_2778x1284  \
count               982.000000                 982.00000
mean                  0.006110                   0.01833
std                   0.077967                   0.13421
min                   0.000000                   0.00000
25%                   0.000000                   0.00000
50%                   0.000000                   0.00000
75%                   0.000000                   0.00000
max                   1.000000                   1.00000

       Розмір дисплею_2796x1290  Розмір дисплею_3088x1440  \
count               982.000000               982.000000
mean                  0.016293                 0.014257
std                   0.126665                 0.118607
min                   0.000000                 0.000000
25%                   0.000000                 0.000000
50%                   0.000000                 0.000000
75%                   0.000000                 0.000000
max                   1.000000                 1.000000

       Розмір дисплею_3200x1440  Розмір дисплею_3216x1440  \
count               982.000000               982.000000
```

```
mean                    0.002037                  0.001018
std                     0.045106                  0.031911
min                     0.000000                  0.000000
25%                     0.000000                  0.000000
50%                     0.000000                  0.000000
75%                     0.000000                  0.000000
max                     1.000000                  1.000000

       Розмір дисплею_960x442  Розмір дисплею_960x480  Розмір дисплею_960x540
count              982.000000              982.000000              982.000000
mean                 0.004073                0.014257                0.001018
std                  0.063725                0.118607                0.031911
min                  0.000000                0.000000                0.000000
25%                  0.000000                0.000000                0.000000
50%                  0.000000                0.000000                0.000000
75%                  0.000000                0.000000                0.000000
max                  1.000000                1.000000                1.000000

[8 rows x 44 columns]
```

```
df_learn = df.copy() # dataframe for alg-thms to be learnt

display(df_learn.shape) # sizes (rows, columns)

df_learn = df_learn.drop(df_learn[df_learn['Ціна']==0].index) # drop 0 prices
(out of stock; no info)

display(df_learn.shape) # sizes (rows, columns)
display(df_learn.head())
```

```
(982, 46)

(598, 46)

    Ціна   Модель  Дисплей  Кількість ядер Оперативна пам'ять  \
0   3999    NOKIA     6.50               8                   3
1   9999    REDMI     6.67               8                   6
2   6999  ONEPLUS     6.52               8                   4
3   7173     VIVO     6.58               8                   4
4  11790  SAMSUNG     6.70               8                   6

   Внутрішня пам'ять  Камера  Операційка  Розмір дисплею_1014x480  \
0                 32    13.0           0                        0
1                128   108.0           0                        0
2                 64    13.0           0                        0
3                 64    48.0           0                        0
4                128    64.0           0                        0

   Розмір дисплею_1280x600  ...  Розмір дисплею_2700x1800  \
0                        0  ...                         0
1                        0  ...                         0
2                        0  ...                         0
3                        0  ...                         0
4                        0  ...                         0

   Розмір дисплею_2712x1220  Розмір дисплею_2778x1284  \
```

```
0                          0                          0
1                          0                          0
2                          0                          0
3                          0                          0
4                          0                          0

   Розмір дисплею_2796x1290  Розмір дисплею_3088x1440  \
0                          0                          0
1                          0                          0
2                          0                          0
3                          0                          0
4                          0                          0

   Розмір дисплею_3200x1440  Розмір дисплею_3216x1440  Розмір дисплею_960x442
\
0                          0                          0                       0
1                          0                          0                       0
2                          0                          0                       0
3                          0                          0                       0
4                          0                          0                       0

   Розмір дисплею_960x480  Розмір дисплею_960x540
0                       0                       0
1                       0                       0
2                       0                       0
3                       0                       0
4                       0                       0

[5 rows x 46 columns]
```

```
model_counts = df_learn['Модель'].value_counts() # counts per model
```

```
model_counts
```

```
SAMSUNG         107
APPLE            83
XIAOMI           51
REDMI            49
TECNO            39
ULEFONE          34
REALME           31
UMIDIGI          29
BLACKVIEW        25
ZTE              25
INFINIX          22
MOTOROLA         18
OPPO             15
VIVO             14
NOKIA            12
TCL              10
OSCAL             8
SIGMA MOBILE      7
DOOGEE            7
ONEPLUS           5
MICROSOFT         3
```

```
ASUS              2
ALCATEL           1
MAXCOM            1
Name: Модель, dtype: int64
```

```
df_learn = df_learn[df_learn['Модель'].isin(model_counts.head().index)] # lea
ve only the top 5 most popular models
unique_models = df_learn['Модель'].unique() # check all the unique models in
df for learning
```

```
unique_models
```

```
array(['REDMI', 'SAMSUNG', 'XIAOMI', 'APPLE', 'TECNO'], dtype=object)
```

```
price_0_models = df[df['Ціна']==0] # save 0 prices to the separate df
# leave there the same models as in df for learning
price_0_models = price_0_models[price_0_models['Модель'].isin(unique_models)]
```

```
price_0_models.head()
```

```
     Ціна  Модель  Дисплей  Кількість ядер Оперативна пам'ять  \
599     0   APPLE     6.70               6                   6
607     0  XIAOMI     6.36               8                   8
608     0  XIAOMI     6.36               8                   8
609     0  XIAOMI     6.55               8                   8
610     0  XIAOMI     6.73               8                  12

     Внутрішня пам'ять  Камера  Операційка  Розмір дисплею_1014x480  \
599                  1    48.0           1                        0
607                256    50.0           0                        0
608                256    50.0           0                        0
609                256    50.0           0                        0
610                256    50.0           0                        0

     Розмір дисплею_1280x600  ...  Розмір дисплею_2700x1800  \
599                        0  ...                         0
607                        0  ...                         0
608                        0  ...                         0
609                        0  ...                         0
610                        0  ...                         0

     Розмір дисплею_2712x1220  Розмір дисплею_2778x1284  \
599                         0                         0
607                         0                         0
608                         0                         0
609                         0                         0
610                         0                         0

     Розмір дисплею_2796x1290  Розмір дисплею_3088x1440  \
599                         1                         0
607                         0                         0
608                         0                         0
609                         0                         0
610                         0                         0

     Розмір дисплею_3200x1440  Розмір дисплею_3216x1440  \
```

```
599                          0                          0
607                          0                          0
608                          0                          0
609                          0                          0
610                          1                          0

     Розмір дисплею_960x442  Розмір дисплею_960x480  Розмір дисплею_960x540
599                       0                       0                       0
607                       0                       0                       0
608                       0                       0                       0
609                       0                       0                       0
610                       0                       0                       0

[5 rows x 46 columns]
```

```
# std of price per model
std_price_per_model = df_learn[['Модель','Ціна']].groupby('Модель', as_index=
False)['Ціна'].std()
# in avg price leave the same models as in learning df
avg_price_per_model = avg_price_per_model[avg_price_per_model['Модель'].isin(
unique_models)]

display(avg_price_per_model)
display(std_price_per_model)
```

```
     Модель          Ціна
1     APPLE  48919.265060
15  SAMSUNG  22506.962617
22   XIAOMI  10413.450980
14    REDMI   6920.571429
18    TECNO   5250.974359

    Модель          Ціна
0    APPLE  13902.628540
1    REDMI   2520.550370
2  SAMSUNG  18996.748510
3    TECNO   2124.390420
4   XIAOMI   4306.303302
```

```
for model in unique_models: # for every top model
    curr_model_std = std_price_per_model[std_price_per_model['Модель']==model
]['Ціна'].values[0] # curr model's price std
    curr_model_avg = avg_price_per_model[avg_price_per_model['Модель'] == mod
el]['Ціна'].values[0] # curr model's price avg
    # for every price in 0 price df change price into average price for the m
odel +- its std price
    price_0_models.loc[price_0_models['Модель']==model, 'Ціна'] = price_0_mod
els[price_0_models['Модель']==model]['Ціна']\
    .apply(lambda x: curr_model_avg + random.uniform(-curr_model_std, curr_mo
del_std))

display(price_0_models.head())
display(df_learn.head())
```

```
             Ціна  Модель  Дисплей  Кількість ядер Оперативна пам'ять  \
599  56325.370466   APPLE     6.70               6                   6
```

```
607   7965.037372  XIAOMI     6.36              8                   8
608  11220.094291  XIAOMI     6.36              8                   8
609  10683.970295  XIAOMI     6.55              8                   8
610  12127.805662  XIAOMI     6.73              8                  12

     Внутрішня пам'ять  Камера  Операційка  Розмір дисплею_1014x480  \
599                   1    48.0           1                         0
607                 256    50.0           0                         0
608                 256    50.0           0                         0
609                 256    50.0           0                         0
610                 256    50.0           0                         0

     Розмір дисплею_1280x600  ...  Розмір дисплею_2700x1800  \
599                        0  ...                         0
607                        0  ...                         0
608                        0  ...                         0
609                        0  ...                         0
610                        0  ...                         0

     Розмір дисплею_2712x1220  Розмір дисплею_2778x1284  \
599                         0                         0
607                         0                         0
608                         0                         0
609                         0                         0
610                         0                         0

     Розмір дисплею_2796x1290  Розмір дисплею_3088x1440  \
599                         1                         0
607                         0                         0
608                         0                         0
609                         0                         0
610                         0                         0

     Розмір дисплею_3200x1440  Розмір дисплею_3216x1440  \
599                         0                         0
607                         0                         0
608                         0                         0
609                         0                         0
610                         1                         0

     Розмір дисплею_960x442  Розмір дисплею_960x480  Розмір дисплею_960x540
599                       0                       0                       0
607                       0                       0                       0
608                       0                       0                       0
609                       0                       0                       0
610                       0                       0                       0

[5 rows x 46 columns]

     Ціна   Модель  Дисплей  Кількість ядер Оперативна пам'ять  \
1    9999    REDMI     6.67               8                   6
4   11790  SAMSUNG     6.70               8                   6
6    6999    REDMI     6.71               8                   4
8    4208   XIAOMI     6.71               8                   3
11  22565   XIAOMI     6.67               8                   8
```

```
    Внутрішня пам'ять  Камера  Операційка  Розмір дисплею_1014x480  \
1                 128   108.0           0                        0
4                 128    64.0           0                        0
6                 128    50.0           0                        0
8                  32    13.0           0                        0
11                256   108.0           0                        0

    Розмір дисплею_1280x600  ...  Розмір дисплею_2700x1800  \
1                         0  ...                         0
4                         0  ...                         0
6                         0  ...                         0
8                         0  ...                         0
11                        0  ...                         0

    Розмір дисплею_2712x1220  Розмір дисплею_2778x1284  \
1                          0                         0
4                          0                         0
6                          0                         0
8                          0                         0
11                         1                         0

    Розмір дисплею_2796x1290  Розмір дисплею_3088x1440  \
1                          0                         0
4                          0                         0
6                          0                         0
8                          0                         0
11                         0                         0

    Розмір дисплею_3200x1440  Розмір дисплею_3216x1440  \
1                          0                         0
4                          0                         0
6                          0                         0
8                          0                         0
11                         0                         0

    Розмір дисплею_960x442  Розмір дисплею_960x480  Розмір дисплею_960x540
1                        0                       0                       0
4                        0                       0                       0
6                        0                       0                       0
8                        0                       0                       0
11                       0                       0                       0

[5 rows x 46 columns]
```

```
df_learn['Модель'] = df_learn['Модель'].apply(lambda x: list(unique_models).i
ndex(x)) # encode models as indexes
df_to_predict_res = price_0_models['Модель'].apply(lambda x: list(unique_mode
ls).index(x)) # encode models for 0 models also
df_to_predict = price_0_models.drop('Модель',axis=1) # delete value for predi
ction from the test df

display(df_learn.head())
display(df_to_predict.head())
display(df_to_predict_res.head())
```

```
     Ціна  Модель  Дисплей  Кількість ядер Оперативна пам'ять  \
1    9999       0     6.67               8                   6
4   11790       1     6.70               8                   6
6    6999       0     6.71               8                   4
8    4208       2     6.71               8                   3
11  22565       2     6.67               8                   8

    Внутрішня пам'ять  Камера  Операційка  Розмір дисплею_1014x480  \
1                 128   108.0           0                        0
4                 128    64.0           0                        0
6                 128    50.0           0                        0
8                  32    13.0           0                        0
11                256   108.0           0                        0

    Розмір дисплею_1280x600  ...  Розмір дисплею_2700x1800  \
1                         0  ...                         0
4                         0  ...                         0
6                         0  ...                         0
8                         0  ...                         0
11                        0  ...                         0

    Розмір дисплею_2712x1220  Розмір дисплею_2778x1284  \
1                          0                         0
4                          0                         0
6                          0                         0
8                          0                         0
11                         1                         0

    Розмір дисплею_2796x1290  Розмір дисплею_3088x1440  \
1                          0                         0
4                          0                         0
6                          0                         0
8                          0                         0
11                         0                         0

    Розмір дисплею_3200x1440  Розмір дисплею_3216x1440  \
1                          0                         0
4                          0                         0
6                          0                         0
8                          0                         0
11                         0                         0

    Розмір дисплею_960x442  Розмір дисплею_960x480  Розмір дисплею_960x540
1                        0                       0                       0
4                        0                       0                       0
6                        0                       0                       0
8                        0                       0                       0
11                       0                       0                       0

[5 rows x 46 columns]

             Ціна  Дисплей  Кількість ядер Оперативна пам'ять  \
599  56325.370466     6.70               6                   6
607   7965.037372     6.36               8                   8
608  11220.094291     6.36               8                   8
```

```
609  10683.970295     6.55               8                   8
610  12127.805662     6.73               8                  12

     Внутрішня пам'ять  Камера  Операційка  Розмір дисплею_1014x480  \
599                   1    48.0           1                        0
607                 256    50.0           0                        0
608                 256    50.0           0                        0
609                 256    50.0           0                        0
610                 256    50.0           0                        0

     Розмір дисплею_1280x600  Розмір дисплею_1280x720  ...  \
599                        0                        0  ...
607                        0                        0  ...
608                        0                        0  ...
609                        0                        0  ...
610                        0                        0  ...

     Розмір дисплею_2700x1800  Розмір дисплею_2712x1220  \
599                         0                         0
607                         0                         0
608                         0                         0
609                         0                         0
610                         0                         0

     Розмір дисплею_2778x1284  Розмір дисплею_2796x1290  \
599                         0                         1
607                         0                         0
608                         0                         0
609                         0                         0
610                         0                         0

     Розмір дисплею_3088x1440  Розмір дисплею_3200x1440  \
599                         0                         0
607                         0                         0
608                         0                         0
609                         0                         0
610                         0                         1

     Розмір дисплею_3216x1440  Розмір дисплею_960x442  Розмір дисплею_960x480
\
599                         0                       0                       0
607                         0                       0                       0
608                         0                       0                       0
609                         0                       0                       0
610                         0                       0                       0

     Розмір дисплею_960x540
599                       0
607                       0
608                       0
609                       0
610                       0

[5 rows x 45 columns]
```

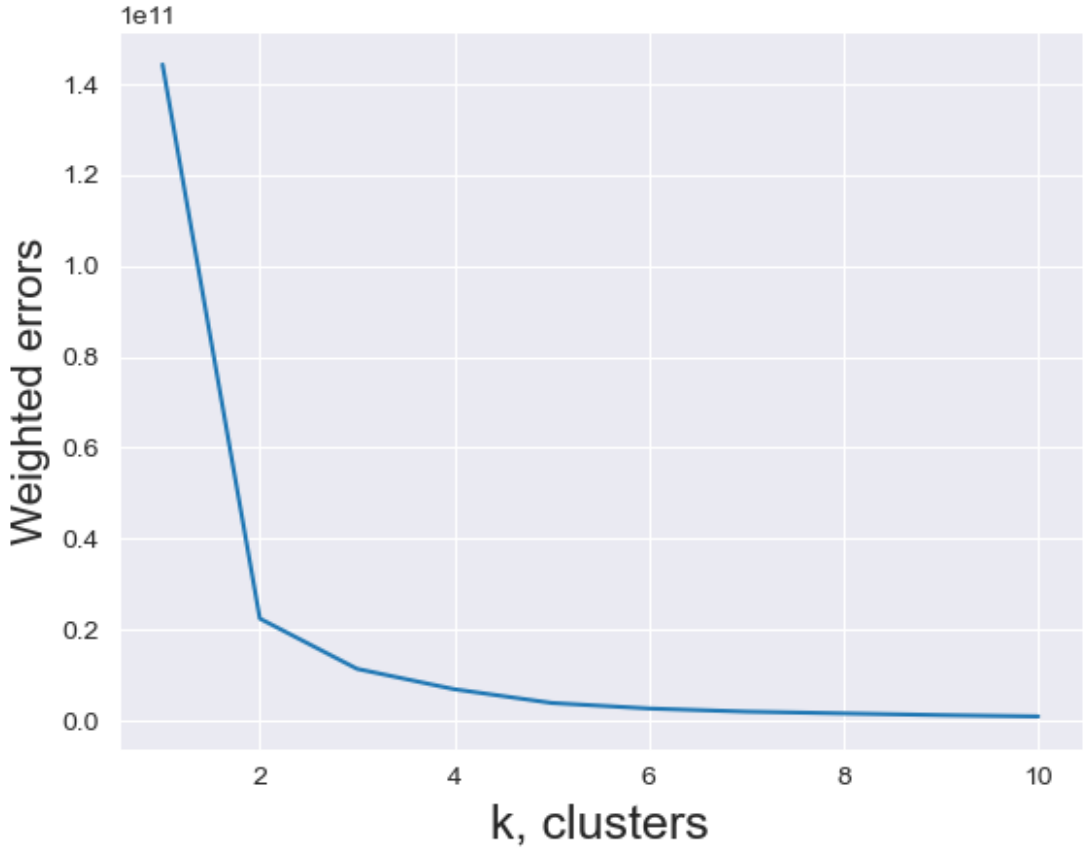
```
599    3
607    2
608    2
609    2
610    2
Name: Модель, dtype: int64
```

```
try:
    # trying to parse all columns in all dfs as floats to check for any remai
ning string-ish elements
    df_learn = df_learn.astype('float')
    df_to_predict = df_to_predict.astype('float')
    df_to_predict_res = df_to_predict_res.astype('float')
except Exception as e: # in case of error
    print(e) # print the error
finally: # anyway
    print("Done")
```

```
Done
```

```
distortions = [] # list for appending inertias for the elbow method
for k in range(1,11): # estimate kmeans for different number of clusters (fro
m 1 to 10)
    kmean = KMeans(n_clusters=k, # amount of clusters
                   n_init=15 # number of times to repeat algorithm to find better
centroids
                  ).fit(df_learn)

    distortions.append(kmean.inertia_) # Sum of squared distances of samples
to their closest cluster center

# plotting
plt.plot([i for i in range(1,11)], # x values
         distortions) # y values
plt.xlabel('k, clusters', fontsize=18) # x axis label
plt.ylabel('Weighted errors', fontsize=16) # y axis label

plt.show()
```

```
D:\Anaconda\lib\site-packages\sklearn\cluster\_kmeans.py:1036: UserWarning: K
Means is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment va
riable OMP_NUM_THREADS=2.
  warnings.warn(
```




```
# we're choosing between
kmeans_2_cl = KMeans(n_clusters=2, n_init=15).fit(df_learn) # 2 clusters kmea
ns
kmeans_3_cl = KMeans(n_clusters=3, n_init=15).fit(df_learn) # 3 clusters kmea
ns

df_2_clusters, df_3_clusters = df_learn.copy(), df_learn.copy() # copy df for
learning for each kmeans
df_2_clusters['Cluster'] = kmeans_2_cl.labels_ # create the 'Cluster' column
to associate each row (model) with its cluster
df_3_clusters['Cluster'] = kmeans_3_cl.labels_

df_grouped_2 = df_2_clusters.groupby('Cluster', as_index=False).mean() # grou
p by cluster and get means (for analysis)
df_grouped_3 = df_3_clusters.groupby('Cluster', as_index=False).mean()

display(df_grouped_2)
display(df_grouped_2.describe())
display(df_grouped_3)
display(df_grouped_3.describe())
```

```
   Cluster          Ціна    Модель   Дисплей  Кількість ядер  \
0        0   9100.774194  1.585253  6.552212        7.686636
1        1  49720.160714  2.410714  6.356250        6.589286

   Оперативна пам'ять  Внутрішня пам'ять     Камера  Операційка  \
0            4.686636         104.184332  48.953917    0.018433
1            6.714286         272.062500  42.089286    0.705357
```

```
   Розмір дисплею_1014x480  ...  Розмір дисплею_2700x1800  \
0                      0.0  ...                       0.0
1                      0.0  ...                       0.0

   Розмір дисплею_2712x1220  Розмір дисплею_2778x1284  \
0                  0.009217                  0.000000
1                  0.000000                  0.116071

   Розмір дисплею_2796x1290  Розмір дисплею_3088x1440  \
0                  0.000000                  0.000000
1                  0.133929                  0.098214

   Розмір дисплею_3200x1440  Розмір дисплею_3216x1440  Розмір дисплею_960x442
\
0                       0.0                       0.0                     0.0
1                       0.0                       0.0                     0.0

   Розмір дисплею_960x480  Розмір дисплею_960x540
0                     0.0                     0.0
1                     0.0                     0.0

[2 rows x 47 columns]

        Cluster          Ціна    Модель   Дисплей  Кількість ядер  \
count  2.000000      2.000000  2.000000  2.000000        2.000000
mean   0.500000  29410.467454  1.997984  6.454231        7.137961
std    0.707107  28722.243656  0.583689  0.138566        0.775944
min    0.000000   9100.774194  1.585253  6.356250        6.589286
25%    0.250000  19255.620824  1.791619  6.405240        6.863623
50%    0.500000  29410.467454  1.997984  6.454231        7.137961
75%    0.750000  39565.314084  2.204349  6.503221        7.412298
max    1.000000  49720.160714  2.410714  6.552212        7.686636

       Оперативна пам'ять  Внутрішня пам'ять     Камера  Операційка  \
count            2.000000           2.000000   2.000000    2.000000
mean             5.700461         188.123416  45.521601    0.361895
std              1.433765         118.707791   4.854027    0.485729
min              4.686636         104.184332  42.089286    0.018433
25%              5.193548         146.153874  43.805444    0.190164
50%              5.700461         188.123416  45.521601    0.361895
75%              6.207373         230.092958  47.237759    0.533626
max              6.714286         272.062500  48.953917    0.705357

       Розмір дисплею_1014x480  ...  Розмір дисплею_2700x1800  \
count                      2.0  ...                       2.0
mean                       0.0  ...                       0.0
std                        0.0  ...                       0.0
min                        0.0  ...                       0.0
25%                        0.0  ...                       0.0
50%                        0.0  ...                       0.0
75%                        0.0  ...                       0.0
max                        0.0  ...                       0.0

       Розмір дисплею_2712x1220  Розмір дисплею_2778x1284  \
count                  2.000000                  2.000000
```

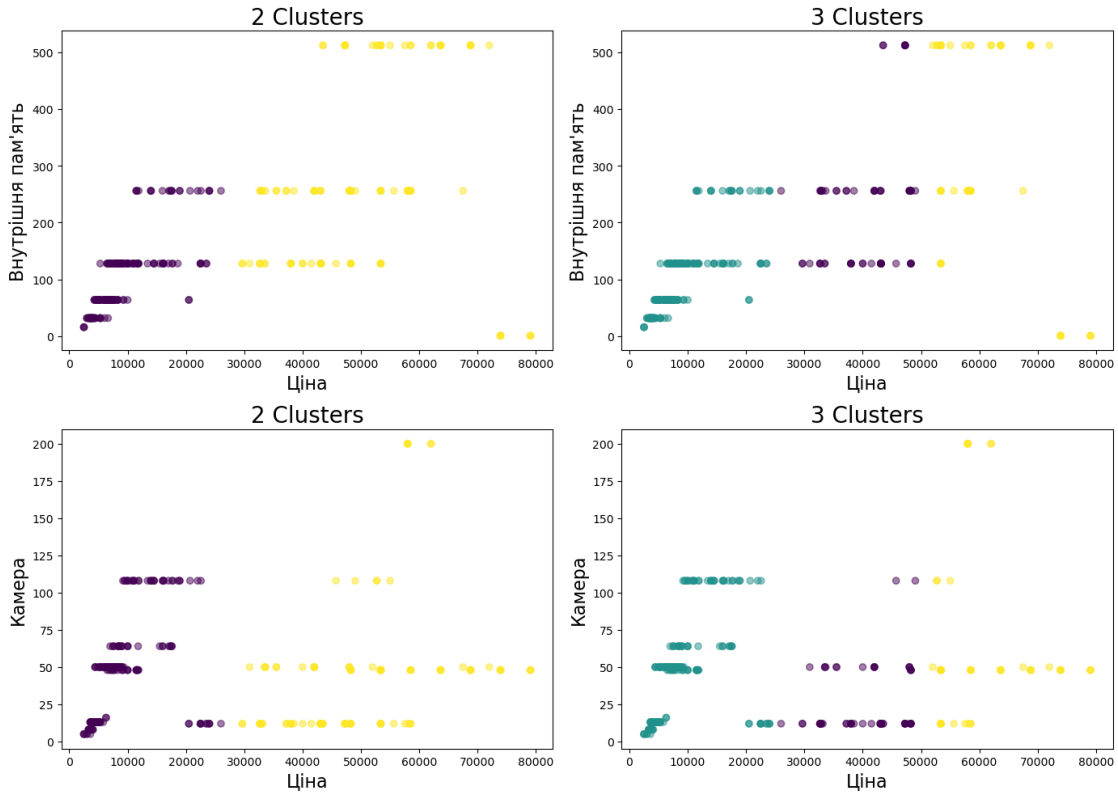
```
mean                    0.004608                    0.058036
std                     0.006517                    0.082075
min                     0.000000                    0.000000
25%                     0.002304                    0.029018
50%                     0.004608                    0.058036
75%                     0.006912                    0.087054
max                     0.009217                    0.116071

       Розмір дисплею_2796x1290  Розмір дисплею_3088x1440  \
count                  2.000000                  2.000000
mean                   0.066964                  0.049107
std                    0.094702                  0.069448
min                    0.000000                  0.000000
25%                    0.033482                  0.024554
50%                    0.066964                  0.049107
75%                    0.100446                  0.073661
max                    0.133929                  0.098214

       Розмір дисплею_3200x1440  Розмір дисплею_3216x1440  \
count                       2.0                       2.0
mean                        0.0                       0.0
std                         0.0                       0.0
min                         0.0                       0.0
25%                         0.0                       0.0
50%                         0.0                       0.0
75%                         0.0                       0.0
max                         0.0                       0.0

       Розмір дисплею_960x442  Розмір дисплею_960x480  Розмір дисплею_960x540
count                     2.0                     2.0                     2.0
mean                      0.0                     0.0                     0.0
std                       0.0                     0.0                     0.0
min                       0.0                     0.0                     0.0
25%                       0.0                     0.0                     0.0
50%                       0.0                     0.0                     0.0
75%                       0.0                     0.0                     0.0
max                       0.0                     0.0                     0.0

[8 rows x 47 columns]

   Cluster          Ціна    Модель   Дисплей  Кількість ядер  \
0        0  40745.812500  2.375000  6.184375        6.625000
1        1   9022.541667  1.587963  6.552917        7.685185
2        2  60957.653061  2.428571  6.581633        6.571429

   Оперативна пам'ять  Внутрішня пам'ять     Камера  Операційка  \
0            6.031250         228.000000  25.562500    0.687500
1            4.671296         103.481481  49.125000    0.018519
2            7.632653         329.285714  63.061224    0.714286

   Розмір дисплею_1014x480  ...  Розмір дисплею_2700x1800  \
0                      0.0  ...                       0.0
1                      0.0  ...                       0.0
2                      0.0  ...                       0.0
```

```
   Розмір дисплею_2712x1220  Розмір дисплею_2778x1284  \
0                  0.000000                  0.156250
1                  0.009259                  0.000000
2                  0.000000                  0.061224

   Розмір дисплею_2796x1290  Розмір дисплею_3088x1440  \
0                  0.000000                  0.031250
1                  0.000000                  0.000000
2                  0.306122                  0.183673

   Розмір дисплею_3200x1440  Розмір дисплею_3216x1440  Розмір дисплею_960x442
\
0                       0.0                       0.0                     0.0
1                       0.0                       0.0                     0.0
2                       0.0                       0.0                     0.0

   Розмір дисплею_960x480  Розмір дисплею_960x540
0                     0.0                     0.0
1                     0.0                     0.0
2                     0.0                     0.0

[3 rows x 47 columns]

       Cluster          Ціна    Модель   Дисплей  Кількість ядер  \
count      3.0      3.000000  3.000000  3.000000        3.000000
mean       1.0  36908.669076  2.130511  6.439641        6.960538
std        1.0  26179.318195  0.470624  0.221533        0.628134
min        0.0   9022.541667  1.587963  6.184375        6.571429
25%        0.5  24884.177083  1.981481  6.368646        6.598214
50%        1.0  40745.812500  2.375000  6.552917        6.625000
75%        1.5  50851.732781  2.401786  6.567275        7.155093
max        2.0  60957.653061  2.428571  6.581633        7.685185

       Оперативна пам'ять  Внутрішня пам'ять     Камера  Операційка  \
count            3.000000           3.000000   3.000000    3.000000
mean             6.111733         220.255732  45.916241    0.473435
std              1.482318         113.101141  18.954173    0.394197
min              4.671296         103.481481  25.562500    0.018519
25%              5.351273         165.740741  37.343750    0.353009
50%              6.031250         228.000000  49.125000    0.687500
75%              6.831952         278.642857  56.093112    0.700893
max              7.632653         329.285714  63.061224    0.714286

       Розмір дисплею_1014x480  ...  Розмір дисплею_2700x1800  \
count                      3.0  ...                       3.0
mean                       0.0  ...                       0.0
std                        0.0  ...                       0.0
min                        0.0  ...                       0.0
25%                        0.0  ...                       0.0
50%                        0.0  ...                       0.0
75%                        0.0  ...                       0.0
max                        0.0  ...                       0.0

       Розмір дисплею_2712x1220  Розмір дисплею_2778x1284  \
count                  3.000000                  3.000000
```

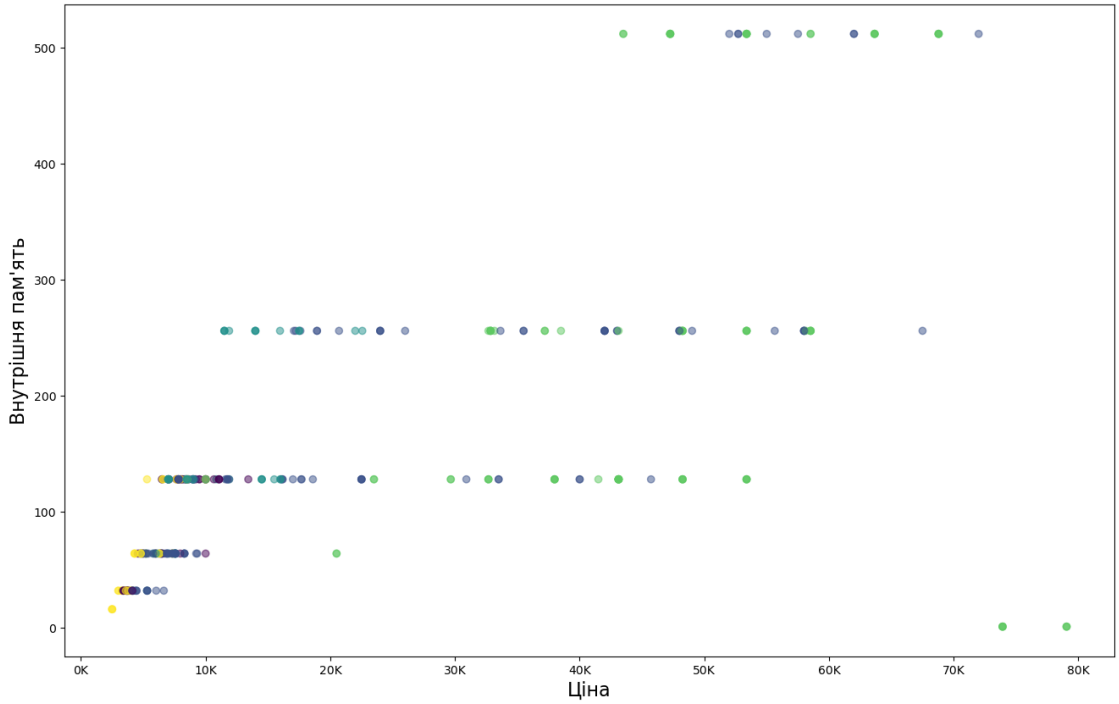
```
mean                    0.003086                    0.072491
std                     0.005346                    0.078732
min                     0.000000                    0.000000
25%                     0.000000                    0.030612
50%                     0.000000                    0.061224
75%                     0.004630                    0.108737
max                     0.009259                    0.156250

       Розмір дисплею_2796x1290  Розмір дисплею_3088x1440  \
count                  3.000000                  3.000000
mean                   0.102041                  0.071641
std                    0.176740                  0.098273
min                    0.000000                  0.000000
25%                    0.000000                  0.015625
50%                    0.000000                  0.031250
75%                    0.153061                  0.107462
max                    0.306122                  0.183673

       Розмір дисплею_3200x1440  Розмір дисплею_3216x1440  \
count                       3.0                       3.0
mean                        0.0                       0.0
std                         0.0                       0.0
min                         0.0                       0.0
25%                         0.0                       0.0
50%                         0.0                       0.0
75%                         0.0                       0.0
max                         0.0                       0.0

       Розмір дисплею_960x442  Розмір дисплею_960x480  Розмір дисплею_960x540
count                     3.0                     3.0                     3.0
mean                      0.0                     0.0                     0.0
std                       0.0                     0.0                     0.0
min                       0.0                     0.0                     0.0
25%                       0.0                     0.0                     0.0
50%                       0.0                     0.0                     0.0
75%                       0.0                     0.0                     0.0
max                       0.0                     0.0                     0.0

[8 rows x 47 columns]
```

```
# plotting clusters with c different colors for each separate cluster
sns.reset_orig() # get back to standard bg


fig = plt.figure(figsize=(14,10)) # the first figure to plot on
plt.subplot(2, 2, 1) # add subplot (2 rows, 2 columns, 1st plotting)
# scatter plot price by device memory colored separately for each cluster, ha
lf transparent (2 clusters)
plt.scatter(df_2_clusters['Ціна'], df_2_clusters["Внутрішня пам'ять"], c=df_2
_clusters['Cluster'], alpha=.5)
plt.title("2 Clusters", fontsize=20) # the scatter plot title
plt.xlabel('Ціна', fontsize=16) # x axis label
plt.ylabel("Внутрішня пам'ять", fontsize=16) # y axis label
```

```
plt.subplot(2, 2, 2) # add subplot (2 rows, 2 columns, 2nd plotting)
# scatter plot price by device memory colored separately for each cluster, ha
lf transparent (3 clusters)
plt.scatter(df_3_clusters['Ціна'], df_3_clusters["Внутрішня пам'ять"], c=df_3
_clusters['Cluster'], alpha=.5)
plt.title("3 Clusters", fontsize=20) # the scatter plot title
plt.xlabel('Ціна', fontsize=16) # x axis label
plt.ylabel("Внутрішня пам'ять", fontsize=16) # y axis label



plt.subplot(2, 2, 3) # add subplot (2 rows, 2 columns, 3rd plotting)
# scatter plot price by camera colored separately for each cluster, half tran
sparent (2 clusters)
plt.scatter(df_2_clusters['Ціна'], df_2_clusters["Камера"], c=df_2_clusters['
Cluster'], alpha=.5)
plt.title("2 Clusters", fontsize=20) # the scatter plot title
plt.xlabel('Ціна', fontsize=16) # x axis label
plt.ylabel("Камера", fontsize=16) # y axis label



plt.subplot(2, 2, 4) # add subplot (2 rows, 2 columns, 4th plotting)
# scatter plot price by camera colored separately for each cluster, half tran
sparent (3 clusters)
plt.scatter(df_3_clusters['Ціна'], df_3_clusters["Камера"], c=df_3_clusters['
Cluster'], alpha=.5)
plt.title("3 Clusters", fontsize=20) # the scatter plot title
plt.xlabel('Ціна', fontsize=16) # x axis label
plt.ylabel("Камера", fontsize=16) # y axis label

fig.tight_layout() # for the plots not to overlap


plt.show() # show the plots
```




```
fig = plt.figure(figsize=(16,10)) # the first figure to plot on
ax = plt.subplot(1,1,1) # add subplot (1 row, 1 column, 1st plotting)
# scatter plot price by device memory colored separately for each cluster, ha
lf transparent (3 clusters)
ax.scatter(df_3_clusters['Ціна'], df_3_clusters["Внутрішня пам'ять"], c=df_3_
clusters['Модель'], alpha=.5)
ax.xaxis.set_major_formatter(lambda x,p: str(int(x/1000))+"K") # custom forma
t y axis labels
plt.xlabel('Ціна', fontsize=16) # x axis label
plt.ylabel("Внутрішня пам'ять", fontsize=16) # y axis label

plt.show() # show the plot
```




```
# dividing df for learning into separate dfs
X = df_learn.drop('Модель', axis=1).astype('float') # independent (x) values

Y = df_learn['Модель'] # dependent value (y) for predicting

X_train, X_test, y_train, y_test = train_test_split(X, # independent variable
s
                                                    Y, # dependent variable (
expected output)
                                                    random_state=1, # pseudo-
random number parameter to reproduce the same train test split each time you
run the code
                                                    train_size = .75) # 75% i
s aimed for training, other 25% - for testing

# дерево рішень
# easy model with least data preparation. Based on multiple statistical choic
e,
# but even slight difference in data can significantly impact the final solut
ion
# - calculate gini impurity for all impure leafes (that are of ambigious choi
ce)
# for every independent variable: 1 - probability of true/n - probability of
false/n
# (for non-binary values we calculate average of every pair and check if the
current value
# less or bigger than the current average)
# - for the root element of the tree the least gini impurity is chosen
# - for further leafs reduce impurity by adding other calculated leafs
# - recalculate gini impurity for elements that were filtered by the root ele
ments
decision_tree = tree.DecisionTreeClassifier().fit(X_train, y_train) # fit the
splitted data from df_learn
dtree_predictions = decision_tree.predict(X_test) # test how good is this alg
```

```
orythm

print("Accuracy of Dicision Tree: {0}%".format(accuracy_score(y_test, dtree_p
redictions) * 100))
print(confusion_matrix(y_test, dtree_predictions))
print(classification_report(y_test, dtree_predictions))
```

```
Accuracy of Dicision Tree: 95.18072289156626%
[[11  0  1  0  0]
 [ 0 26  2  0  0]
 [ 0  1 14  0  0]
 [ 0  0  0 22  0]
 [ 0  0  0  0  6]]
              precision    recall  f1-score   support

         0.0       1.00      0.92      0.96        12
         1.0       0.96      0.93      0.95        28
         2.0       0.82      0.93      0.87        15
         3.0       1.00      1.00      1.00        22
         4.0       1.00      1.00      1.00         6

    accuracy                           0.95        83
   macro avg       0.96      0.96      0.96        83
weighted avg       0.96      0.95      0.95        83
```

```
# логістична регресія
# logistic regression has a sygmoid to fit the data rather than a stright lin
e
# it also predicts binary values, not discrete ones, lika the linear regressi
on
# instead of minimizing sum of residual squares (we can also get rid off non-
helping
# independent variables):
# 1. pick a probability, scaled by weight
# 2. using it calculate the likelihood of the prediction being "False"
# 3. do this for all data and multiply them - that is the likelihood of the d
ata
# according to the current sigmoid
# 4. shift the line and repeat 1-4 until we reach the maximum likelihood
fit_lg = LogisticRegression(penalty='none').fit(X_train, y_train) # fit the s
plitted data from df_learn
lg_predictions = fit_lg.predict(X_test) # test how good is this algorythm

print("Accuracy of Logistic Regression: {0}%".format(accuracy_score(y_test, l
g_predictions) * 100))
print(confusion_matrix(y_test, lg_predictions))
print(classification_report(y_test, lg_predictions))
```

```
Accuracy of Logistic Regression: 55.42168674698795%
[[ 4  2  5  0  1]
 [ 3 15  5  5  0]
 [ 4  5  4  0  2]
 [ 0  0  0 22  0]
 [ 1  2  2  0  1]]
              precision    recall  f1-score   support
```

```
              0.0       0.33      0.33      0.33        12
              1.0       0.62      0.54      0.58        28
              2.0       0.25      0.27      0.26        15
              3.0       0.81      1.00      0.90        22
              4.0       0.25      0.17      0.20         6

    accuracy                           0.55        83
   macro avg       0.45      0.46      0.45        83
weighted avg       0.54      0.55      0.54        83


D:\Anaconda\lib\site-packages\sklearn\linear_model\_logistic.py:814: Converge
nceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
sion
  n_iter_i = _check_optimize_result(
```

```
# random forest
# similar to the Tree Clasifier, but it may build hundreds of trees randomly
# and then, predictin the data, we run our test data through every tree and
# then venture the decision according to the most votes
rf = RandomForestClassifier(n_estimators = 800, random_state = 41).fit(X_trai
n, y_train) # fit the splitted data from df_learn
rf_predictions = rf.predict(X_test) # test how good is this algorythm
accuracy_score_rf = accuracy_score(y_test, rf_predictions)

print("Accuracy of Logistic Regression: {0}%".format(accuracy_score_rf * 100)
)
print(confusion_matrix(y_test, rf_predictions))
print(classification_report(y_test, rf_predictions))
```

```
Accuracy of Logistic Regression: 97.59036144578313%
[[11  0  1  0  0]
 [ 0 28  0  0  0]
 [ 0  1 14  0  0]
 [ 0  0  0 22  0]
 [ 0  0  0  0  6]]
              precision    recall  f1-score   support

         0.0       1.00      0.92      0.96        12
         1.0       0.97      1.00      0.98        28
         2.0       0.93      0.93      0.93        15
         3.0       1.00      1.00      1.00        22
         4.0       1.00      1.00      1.00         6

    accuracy                           0.98        83
   macro avg       0.98      0.97      0.97        83
weighted avg       0.98      0.98      0.98        83
```

```
# trying out ensemble
estimators = [('decision_tree', decision_tree), ("RForest", rf)] # set of alg
orythms to combine
ensemble = VotingClassifier(estimators, # the algorythms
                            voting='hard') # make predicitions by majority vo
te

ensemble.fit(X_train, y_train) # fit the splitted data from df_learn
Y_prediction = ensemble.predict(X_test) # test how good is this algorythm

print(classification_report(Y_prediction,y_test))
print("Voting Ensemble:>" + str(accuracy_score(Y_prediction,y_test)*100) + "%
")
```

```
              precision    recall  f1-score   support

         0.0       0.92      1.00      0.96        11
         1.0       1.00      0.97      0.98        29
         2.0       0.93      0.93      0.93        15
         3.0       1.00      1.00      1.00        22
         4.0       1.00      1.00      1.00         6

    accuracy                           0.98        83
   macro avg       0.97      0.98      0.97        83
weighted avg       0.98      0.98      0.98        83

Voting Ensemble:>97.59036144578313%
```

```
predictions = rf.predict(df_to_predict) # predicting 0 prices models with pse
udo-randomized prices using the best alg

print(classification_report(predictions,df_to_predict_res))
print("Result:>" + str(accuracy_score(predictions,df_to_predict_res)*100) + "
%")
```

```
              precision    recall  f1-score   support

         0.0       0.76      0.76      0.76        29
         1.0       0.86      0.84      0.85        77
         2.0       0.47      0.65      0.55        26
         3.0       1.00      1.00      1.00        31
         4.0       0.89      0.59      0.71        27

    accuracy                           0.79       190
   macro avg       0.79      0.77      0.77       190
weighted avg       0.82      0.79      0.80       190

Result:>79.47368421052632%
```

```
avg_price_per_model
```

	Модель	Ціна
1	APPLE	48919.265060
15	SAMSUNG	22506.962617
22	XIAOMI	10413.450980

```
14    REDMI   6920.571429
18    TECNO   5250.974359
```

```
# minimizing prices, maximizing attributes (for each OS)
predictions = rf.predict([[5250.974359, 7.6, 8, 12, 512, 200, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0],
                          [5250.974359, 7.6, 8, 12, 512, 200, 1, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0]
                          ])

display(predictions)
display(unique_models)
```

```
D:\Anaconda\lib\site-packages\sklearn\base.py:450: UserWarning: X does not ha
ve valid feature names, but RandomForestClassifier was fitted with feature na
mes
  warnings.warn(

array([1., 3.])

array(['REDMI', 'SAMSUNG', 'XIAOMI', 'APPLE', 'TECNO'], dtype=object)
```

Summary

Врахувавши популярність кожної моделі, вибір було звужено до наступних п'яти:

```
avg_price_per_model['Модель']
```

```
1       APPLE
15    SAMSUNG
22     XIAOMI
14      REDMI
18      TECNO
Name: Модель, dtype: object
```

Ціна будь-якої моделі сильно залежить від камери, внутрішньої пам'яті та бренду. Всі моделі було вирішено поділити на сегмент бюджетний:

```
df_grouped_3[df_grouped_3['Cluster']==2]
```

```
   Cluster         Ціна    Модель   Дисплей  Кількість ядер  \
2        2  8584.566038  1.561321  6.556557        7.641509

   Оперативна пам'ять  Внутрішня пам'ять     Камера  Операційка  \
2            4.504717          96.377358  47.886792    0.009434

   Розмір дисплею_1014x480  ...  Розмір дисплею_2700x1800  \
2                      0.0  ...                       0.0

   Розмір дисплею_2712x1220  Розмір дисплею_2778x1284  \
2                  0.009434                       0.0

   Розмір дисплею_2796x1290  Розмір дисплею_3088x1440  \
2                       0.0                       0.0
```

```
   Розмір дисплею_3200x1440  Розмір дисплею_3216x1440  Розмір дисплею_960x442
\
2                       0.0                       0.0                      0.0

   Розмір дисплею_960x480  Розмір дисплею_960x540
2                     0.0                     0.0

[1 rows x 47 columns]
```

середній:

```
df_grouped_3[df_grouped_3['Cluster']==1]
```

```
   Cluster      Ціна  Модель  Дисплей  Кількість ядер  Оперативна пам'ять  \
1        1  61905.92    2.48    6.566            6.52                7.48

   Внутрішня пам'ять  Камера  Операційка  Розмір дисплею_1014x480  ...  \
1              317.6    62.0        0.74                      0.0  ...

   Розмір дисплею_2700x1800  Розмір дисплею_2712x1220  \
1                       0.0                       0.0

   Розмір дисплею_2778x1284  Розмір дисплею_2796x1290  \
1                      0.08                      0.32

   Розмір дисплею_3088x1440  Розмір дисплею_3200x1440  \
1                      0.18                       0.0

   Розмір дисплею_3216x1440  Розмір дисплею_960x442  Розмір дисплею_960x480
\
1                       0.0                     0.0                      0.0

   Розмір дисплею_960x540
1                     0.0

[1 rows x 47 columns]
```

дорогий:

```
df_grouped_3[df_grouped_3['Cluster']==0]
```

```
   Cluster          Ціна    Модель   Дисплей  Кількість ядер  \
0        0  40046.324324  2.378378  6.131081        6.621622

   Оперативна пам'ять  Внутрішня пам'ять     Камера  Операційка  \
0            5.945946         236.972973  25.027027    0.689189

   Розмір дисплею_1014x480  ...  Розмір дисплею_2700x1800  \
0                      0.0  ...                       0.0

   Розмір дисплею_2712x1220  Розмір дисплею_2778x1284  \
0                       0.0                  0.121622

   Розмір дисплею_2796x1290  Розмір дисплею_3088x1440  \
0                       0.0                  0.040541
```

```
   Розмір дисплею_3200x1440  Розмір дисплею_3216x1440  Розмір дисплею_960x442
\
0                        0.0                       0.0                       0.0

   Розмір дисплею_960x480  Розмір дисплею_960x540
0                     0.0                     0.0

[1 rows x 47 columns]
```

Найдоцільнішим методом навчання мережі виявився Random Forest. Для данної моделі не підійшли б методи чисельного наближення або пошуку загальних закономірностей, адже бюджетний сегмент має дуже високу щільність значень, частина з яких не пов'язана по групах. Точність навченої моделі склала 97.59036144578313% для справжньої вибірки і 79.47368421052632% для імітованої. Тож, довіряючи системі можемо вирішити задачу мінімізації ціни при максимізації параметрів відповідно для кожної з ОС: Для Android найкращим вибором буде модель SAMSUNG; Для iOS найкращим вибором буде модель Apple; Оскільки, на основі досліджених даних для Apple переплата йде значною кількістю за бренд (найвище середнє значення ціни при досить великій кількості моделей), ця альтернатива виключається. Фінальний вибір залишається за моделлю SAMSUNG.