НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

До захисту допущено:
В.о. завідувач кафедри
_____ Оксана ТИМОЩУК
«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системи та методи штучного інтелекту»
спеціальності 122 «Комп'ютерні науки»
на тему: «Прогнозування діапазону цін на мобільні телефони методами інтелектуального аналізу даних»

Виконав:
студент IV курсу, групи КА-76
Борбела Артур Юрійович _____

Керівник:
доцент, д.т.н. Недашківська Надія Іванівна _____

Консультант з нормконтролю:
доцент, к.т.н. Коваленко Анатолій Єпіфанович _____

Консультант з економічного розділу:
доцент, к.е.н Рощина Надія Василівна _____

Рецензент:
доцент кафедри СП, к.т.н. Гіоргізова-Гай В.Ш. _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ - 2021 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма «Системи та методи штучного інтелекту»

ЗАТВЕРДЖУЮ
Завідувача кафедри
_____ Оксана ТИМОЩУК
«26» травня 2020 р.

ЗАВДАННЯ
на дипломну роботу студенту
Борбелі Артуру Юрійовичу

1. Тема роботи «Прогнозування діапазону цін на мобільні телефони методами інтелектуального аналізу даних» , керівник роботи доцент кафедри ММСА, Недашківська Надія Іванівна, затверджені наказом по університету від «26» травня 2021 р. № 1143-с.
2. Термін подання студентами роботи 8.06.2021.
3. Вихідні дані до роботи
4. Зміст роботи
5. Перелік ілюстративного матеріалу
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Рощина Надія Василівна, канд.економ.наук, доцент		

7. Дата видачі завдання

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів	Примітка
1	Затвердження теми ДР	15.04.2021-21.04.2021	Виконано
2	Аналіз актуальності задачі дослідження	15.04.2021-28.04.2021	Виконано
3	Збір та аналіз даних	21.04.2021-28.04.2021	Виконано
4	Формулювання задачі дослідження	28.04.2021-05.05.2021	Виконано
5	Розробка програмного продукту	05.05.2021-12.05.2021	Виконано
6	Аналіз та впорядкування результатів	05.05.2021-19.05.2021	Виконано
7	Оформлення пояснювальної записки	19.05.2021-26.05.2021	Виконано
8	Оформлення презентації для демонстрації	19.05.2021-26.05.2021	Виконано

Студент Артур БОРБЕЛА

Керівник роботи Надія НЕДАШКІВСЬКА

РЕФЕРАТ

Дипломна робота 124 ст., 41 рис., 38 табл., 2 додатки., 17 джерела.

КЛАСИФІКАЦІЯ, МАШИННЕ НАВЧАННЯ, НЕЙРОННА МЕРЕЖА, БУСТИНГ, ДЕРЕВА РІШЕНЬ, БЕГГІНГ, ГРАДІЄНТНИЙ СПУСК.

У дослідженні використано всі найбільш розповсюджені та відомі методи машинного навчання. Досліджені алгоритми методи їх переваги та недоліки. Результати роботи обраних мною методів були розглянуті на практиці в задачі класифікації мобільних телефонів до певної цінової категорії.

Об'єктом дослідження стали зібрані дані характеристик мобільних телефонів та їхньої цінової категорії.

Предметом дослідження стали математичні методи машинного навчання такі, як дерева рішень, градієнтний спуск, беггінг, бустінг для проведення класифікації на основі статистичних даних.

ABSTRACT

Thesis 124 art., 41 fig., 38 table., 2 appendices., 17 sources.

CLASSIFICATION, MACHINE LEARNING, NEURAL NETWORK, BUSTING, DECISION TREES, BAGGING, GRADIENT DOWNHILL.

The study used all the most common and well-known methods of machine learning. Research algorithms methods their advantages and disadvantages. The results of my chosen methods were considered in practice in the problem of classifying mobile phones into a certain price category.

The object of the study was to collect data on the characteristics of mobile phones and their price category.

The subject of the study were mathematical methods of machine learning such as decision trees, gradient descent, beggling, boosting for classification based on statistical data.

ЗМІСТ

ВСТУП

Мобільні телефони стали невід'ємною частиною повсякденного життя. Зараз важко когось здивувати новинками техніки, але кілька років тому середньостатистична людина не могла і мріяти про такий винахід, як мобільний телефон. Цей девайс дозволяє бути завжди на зв’язку та використовувати різні необхідні програми. Мобільні телефони спростили наше життя. Тепер немає потреби бути прив’язаним до певного місця, якщо є бажання поговорити з іншою людиною.

На даний момент світові виробники пропонують широкий вибір цього пристрою з різними технічними характеристиками з кожним роком зростає кількість нових технологій які намагаються влаштувати в цей пристрій або розвивають старі роблячи їх кращими. Також оновлюється програмне забезпечення яке в свою чергу також потребує від девайсу певних технічних характеристик. Отже телефон це пристрій якій кожній людині приходиться оновлювати в середньому раз на два роки й прогнозування цінових категорій є важливою задачею для кращого вибору цього пристрою.

Тематика першого розділу присвячена огляду сфери та застосування в ній машинного навчання.

У другому розділі присутній огляд всіх найбільш відомих моделей класифікації з вказанням переваг та недоліків кожної.

Третій розділ розглядає результати роботи обраних методів та порівняння їх .

У четвертому розділі був проведений функціонально-вартісний аналіз програмного продукту.

Предметом дослідження є задача класифікації цінових категорій для мобільного телефону.

РОЗДІЛ 1 ОГЛЯД СФЕРИ ТА ВИКОРИСТАННЯ В НІЙ МЕТОДІВ МАШИННОГО НАВЧАННЯ В ЗАДАЧІ КЛАСИФІКАЦІЇ

1.1 Поняття задачі класифікації

Класифікація - це процес прогнозування класу точок для певних елементів з вибірки. Класи зазвичай називають цілями, мітками або категоріями. Класифікаційне прогнозне моделювання є завдання наближення відображення функції (f) від вхідних даних (x) до дискретних змінних (y).

Наприклад, виявлення спаму у постачальників послуг електронної пошти можно віднести до задач класифікації. Це двійкова класифікація, оскільки існує лише 2 класи такі як спам та не спам. Класифікатор використовує деякі навчальні дані, щоб зрозуміти, як вхідні змінні в даному випадку це листи співвідносяться з класом. У цьому випадку в якості навчальних даних повинні використовуватись відомі електронні листи які вже мають мітку спам або не спам. Коли класифікатор навчений і добре відпрацьовує його можна використовувати для виявлення спаму в нових листах електронної пошти.

Класифікація відноситься до категорії алгоритмів навчання з учителем де цілі також забезпечуються навчальними даними. Є багато застосувань класифікації у різних сферах таких як медична діагностика, цільовий маркетинг, кредитування тощо.

Алгоритмів для класифікації в залежності від процесу навчання можна розділити на дві категорії:

Із “ледачим” навчанням алгоритм зберігає дані навчання та чекає поки не з'явиться дані для тесту. Коли це відбувається, класифікація проводиться на основі найбільш пов'язаних даних, що зберігаються в навчальних даних. У порівнянні з прогресивним навчанням у цього виду меньше часу витрачається на навчання але більше часу на прогнозування.

З прогресивним навчанням алгоритм будує модель класифікації на основі поданих навчальних даних перед тим як отримувати дані для класифікації. Він повинен мати змогу визначити хоча б одну гіпотезу яка охоплює всю вибірку за допомогою якої вже й буде відбуватись класифікація тестових даних. Завдяки побудові моделі цим типам алгоритмів потрібно багато часу на навчання і менше часу на прогнозування.

Зараз існує багато алгоритмів класифікації, але неможливо зробити висновки, який з них перевершує інші. Це залежить від застосування та характеру наявного набору даних. Наприклад, якщо класи лінійно відокремлені лінійні класифікатори такі як логістична регресія, лінійний дискримінант Фішера, можуть перевершувати складні моделі і навпаки.

1.2 Застосування методів штучного інтелекту в сфері прогнозування цін

Прогнозування цін є корисною функцією як для споживачів так і для виробників. Інструмент прогнозування цін спонукає користувачів взаємодіяти з виробником або оцінювати пропозиції, щоб розумно витратити свої гроші. Прогнозування цін дозволяє підприємствам встановлювати ціни таким чином, щоб залучати нових клієнтів та бути лояльними до старих. В машинному навчанні зазвичай дану проблему вирішують за допомогою регресійного аналізу але також можна й за допомогою класифікації якщо нам не важлива точна ціна, а лише цінова категорія. Наприклад розглянемо три різні галузі в яких має місце прогнозування цін.

Авіакомпанія та готельна індустрія: ціни на авіалінії та готелі коливаються залежно від сезону, конкретних днів тижня, свят або інших

змінних. Наприклад під час Різдвяних сезону попит на готелі та рейси зростає і як результат ціни зростають також. І навпаки коли готель намагається заповнити свої номери ціни падають. Яскравим прикладом є ефект COVID-19. Пандемія нанесла сильний удар по авіакомпаніям та сектору туризму, що призвело до зниження цін у всій галузі для спроб залучити більше клієнтів.

На ціни також негативно можуть вплинути ріст та вдосконалення технологій чи використання нових для задоволення бажань клієнтів. Крім того, конкуренти грають велику роль коригуючи свої цінові стратегії для отримання конкурентних переваг.

Ефективні програми ML для авіакомпаній та готельного господарства: повторювана нейронна мережа (RNN) або XGBoost є найбільш часто використовуваними алгоритмами ML для прогнозування авіарейсів та цін на готелі. RNN - це нейронна мережа для послідовних даних, таких як часові ряди, текст, відео, мова або фінансові дані. Тому він може точно прогнозувати майбутні значення. В якості альтернативи XGBoost - це деревоподібний керований алгоритм навчання. Він використовує набір слабших моделей (дерев) для прогнозування цільової змінної з більшою точністю шляхом усереднення їхніх оцінок.

Ці алгоритми ML можна використовувати для прогнозування цін на авіакомпанії та готелі на термін до шести місяців наперед - з високим ступенем точності. Це допомагає споживачам вирішити, купувати зараз чи чекати вигідної угоди.

Житлова промисловість: до атрибутів, що безпосередньо впливають на ціни на житло, належать: кількість кімнат, рік будівництва, якість будівництва, кількість ванних кімнат, кухонних приладів та багато іншого. Крім того, зовнішні фактори, такі як економічні та політичні обставини, процентні ставки та місцевий клімат, також можуть впливати на ринок житла.

Інші фактори на макрорівні можуть включати індекс споживчих цін (ІСЦ), показник валового внутрішнього продукту (ВВП), рівень безробіття тощо. Ці фактори, якщо взяти їх у цілому, можуть мати глибокий вплив на місцеві та регіональні ринки житла.

Ефективні програми ML для житлової промисловості: дослідники використовували різні комбінації дерев регресії, k-найближчих сусідів, опорні векторні машини та глибокі мережі для прогнозування цін на житло. Правильна та інтенсивна розробка особливостей надзвичайно важлива для підвищення точності прогнозування. Моделі, що складаються з ансамблів дерев регресії, прогнозують ціни з найвищим рівнем точності.

Складний алгоритм ML кращий, ніж модель лінійної регресії, оскільки похибки значно менші. Також використовується моделювання часових рядів, оскільки це може значно покращити ефективність прогнозування моделі.

Фондовий ринок: покупець акцій хотів би обґрунтовано вирішити, коли купувати акції, а коли продавати їх, щоб отримати прибуток. Однак численні фактори, такі як попит та пропозиція на акції, глобальна економічна та політична ситуація, історичні ціни, стихійне лихо, інфляція та дефляція ускладнюють прогнозування структури цін на акції. Більше того, прибуток на акцію (EPS), показник ефективності діяльності компанії, безпосередньо впливає на ціни акцій.

Ціни акцій також можуть зазнати несподіваних максимумів і падінь через особливі події, які інвесторам дуже важко передбачити. Це явище, відоме як “ринкові настрої”, часто призводить до того, що інвестори приймають нераціональні інвестиційні рішення. Для прогнозування майбутнього руху ціни акцій можна застосовувати методи штучного інтелекту, використовуючи дані ринку та новин.

Ефективні програми ML для акцій: моделювання часових рядів може бути застосовано для поліпшення здатності прогнозувати ціни акцій з

розумним рівнем точності. Хороша модель ML та AI розглядає історію послідовності даних і правильно передбачає, якими будуть майбутні елементи послідовності. Щодо цін на акції, у даних немає послідовних закономірностей для моделювання цін на акції з часом. У цьому випадку модель із точністю 60% може забезпечити надійну віддачу.

Розширені алгоритми ML для прогнозування запасів складаються з алгоритмів прогнозування часових рядів, таких як авторегресійна інтегрована змінна середнього (ARIMA), сезонна авторегресійна інтегрована змінна середнього (SARIMA), векторна авторегресія (VAR) тощо. Метод довгострокової короткочасної пам'яті (LSTM) - популярний алгоритм глибокого навчання, що використовується для прогнозування ціни акцій.

Дослідження також показали покращені результати з використанням технології Vector Vector Machine (SVM) з трюком ядра.

Оскільки фінансові установи починають застосовувати технології ШІ, підходи ML у поєднанні з іншими зовнішніми факторами, такими як поганими або позитивними настроями новин можуть бути використані для досягнення більш високого ступеня точності прогнозування ціни акцій.

Прогнозування цін є важливим інструментом для клієнтів, підприємців, підприємств та продавців нерухомості. Це може полегшити прийняття рішень у повсякденних операціях та довгостроковому плануванні. Припускаючи, що користувач має доступ до поточних та якісних даних, алгоритми ML може бути використаний для надання важливих даних прогнозу цін. Вибір правильного алгоритму є надзвичайно важливим, оскільки алгоритм повинен мати можливість фіксувати суть даних і надавати точні оцінки майбутньої ціни.

1.3 Попередня обробка та аналіз даних

Взагалі попередня перевірка та обробка даних дозволяє аналітикам збільшити якість результатів аналізу шляхом покращення самих даних.

На даний час більша кількість компаній турбується про якість даних, оскільки погана якість даних може негативно вплинути на правильність та коректність рішення що у свою чергу негативно вплине на бажаний результат. Із-за цього компанії розпочали приймати певні міри для уникнення цієї проблеми.

Особливо питання якості даних піднімається в сфері фінансових послуг. Пошкоджені або неправдиві дані не можуть бути використані в оцінці ризиків або прогнозуванні цін. Із-за цього безпосередньо необхідно приділити увагу на рішення даної проблеми. В основному аналітики не звертають свою увагу на попередню перевірку даних маючи на увазі, що саме так званий власник даних має надавати коректні дані. Проте наразі зараз цей погляд змінюється і компанії для підвищення якості аналізу спочатку роблять етап очищення даних. Очищення в основному складається з видалення дубльованих даних, неможливих, недостовірних та заповнення пустих даних. Також деякі проводять етап оптимізації на якому зменшується розмірність даних шляхом видалення не значущих ознак.

Етап очищення даних

Оскільки від впорядкованості інформації залежить якість результату аналізу потрібно ці дані нормалізувати тобто замінити неправдиві дані. Поява цих неправдивих даних спричиняється із-за різних чинників наприклад людського фактору або системного збою в роботі техніки. Але найрозповсюдженими являються:

а) помилки при заповненні даних людиною

б) використання різних форматів даних які погано конвертуються

в) використання різних одиниць вимірювань

г) невідповідність до певної структури

д) невдала робота з даними що призвела до їхньої модифікації

Основними варіантами помилок в даних вважають :

а) пропуски або так звані null значення

б) аномальні або неможливі значення

в) суперечливі дані

г) шум

Якщо виявлено суперечність в даних перше що потрібно зробити це видалити їх. Але такий спосіб негативно вплине на загальні дані а отже й на подальший результат. Тому зараз намагаються не видаляти їх а виправляти як варіант обчислити ймовірність появи кожної з суперечливих подій і обрати найбільш вірогідний цей варіант зазвичай використовують на практиці.

Пропуски або так звані null значення

Найпоширеніша проблема з якою стикаються при роботі з великими даними це їх не відповідність певній структурі тобто нехватка деяких параметрів, яку при об'єднанні даних заповнюють пустими значеннями. Із-за цього прогнозування може значно погіршуватись. Для виправлення цього застосовують наступні методи:

а) Зі всіх даних вибирається найбільш правдоподібне в залежності від характеристик і замінює цей пропуск;

б) Апроксимують значення тобто коли значення не доступне в певній точці береться її околиця і обчислюється значення яке потім замінює пропуск;

Аномальні або неможливі значення

Неможливими даними називають дані які не входять в діапазон коректних даних для певної характеристики або не відповідають одиниці вимірювання яку в свою чергу не можливо конвертувати в необхідну. Такі

дані потрібно редагувати та приводити до найімовірнішого значення, бо аномальні теж негативно можуть вплинути на результат роботи. Для запобігання цього застосовують:

а) видалення значення

б) заміна значення на найближче граничне

Шум

Шумом зазвичай називають відхил від середнього значення для певної характеристики, оскільки дане відхилення не несе в собі ніякої додаткової або корисної інформації його можна запобігти що значно краще може вплинути на подальший результат. Методи запобігання цьому:

а) авторегресійні методи - в основному використовуються для аналізу числових рядів і зводиться до обчислення функції результат роботи якої це опис процесу та шум. Після знаходження шуму його можна відкинути залишивши лише основне значення;

б) спектральний аналіз - головна його мета відсікти високочастотні складові дані. Зазвичай це часті й незначні відхилення від основного значення;

Етап оптимізації даних

Гарно впливає на результат оптимізація даних перед прогнозуванням. В основному ця оптимізація складається з використання методу узагальнення змінних на вибірці. Це спричиняє зниження розмірності даних. Головними методами оптимізації являються:

а) метод факторного аналізу - в основі якого дві мети: класифікувати дані позбавившись від непотрібної характеристики або просто вилучити найменш значущу характеристику;

б) метод головних компонент - головна мета якого зменшити розмірність змінних з мінімальною втратою корисності інформації. Метод відображує елементи у вигляді спадної регресії кількості характеристик в них;

1.4 Етапи розробки моделі машинного навчання

Етапи розробки моделі можна розділити на наступні:

а) відбір ключових характеристик в залежності від потреби алгоритму;

б) вибір типу вирішуваної задачі та її альтернативи

в) тренуванням алгоритму та конфігурація гіперпараметрів

г) тестування моделі та внесення певних змін для кращої роботи;

Якщо кінцевий результат задовольняє певні наші вимоги то ми використовуємо дану модель на практиці.

На першому етапі маємо відібрати для нас важливі характеристики даних, оскільки від них напряму залежить результат роботи. Перевірити чи є якість характеристики які не несуть інформаційної цінності при даній задачі та відкинути їх. Наприклад якщо ми оцінюємо чи надати клієнтові кредит чи ні нам не важливий колір його волосся нас більше цікавить його дохід. Оцінити вплив інших характеристик на результат, можливо існують якісь інші приховані закономірності, потім обравши оптимальну структуру даних вже переходити на етап вибору задачі. На вибору задачі ми маємо точно визначити яка перед нами задача, їх лише 5 основних типів: класифікація, кластеризація, регресія, ранжування або знаходження аномалій. Далі перед самим процесом навчання ми повинні розбити нашу вибірку на навчальну та тестову зазвичай в співвідношенні 80% та 20% .

Навчальна вибірка передається алгоритму й вже на її основі буде сконфігурована модель, а на тестовій вибірці ми перевіряємо якість роботи нетренованої моделі.

1.5 Постановка задачі

Проблема оптимального вибору телефону виникала у кожного адже всім при покупці хочеться розуміти чи коштує даний пристрій з певним набором характеристик таку суму чи то просто велика комісія посередників які збувають даний товар. Отже можливість приблизно знати цінову категорію даного девайсу допоможить оптимально зробити свій вибір. А вибір в даній галузі дуже великий і майже завжди знайдеться аналог точно ж з такими характеристиками, але вже можливо в іншій ціновій категорії чи від більш надійного виробника.
Метою дипломної роботи є дослідження методів машинного навчання для класифікації мобільних телефонів до цінових категорій.

Для дипломної роботи було обрано дані про мобільні телефони їхні характеристики та цінові категорії за період 2017 року. Дані містять як бінарні значення для відображення присутності чи відсутності технологій та неперервні для значень різних технічних вимірювань.
Необхідно розглянути наступні задачі:

а) зробити аналіз базових методів

б) дослідити найрозповсюдженіші алгоритми класифікації та їх модифікації

в) за результатами вибрати модель яка найкраще відпрацювала

1.6 Висновки

Алгоритму штучного інтелекту, машиного навчання надають можливість або значно оптимізують процес прогнозування та класифікацій у фінансовій сфері надаючи можливість аналітикам швидко шукати та перевіряти певні закономірності на великих даних, що значно продвигає

їхню роботу вперед. Майже в любій фінансовій сфері він вже має своє застосування, що призводить до меншої необхідності витрат з боку людини, виключає людській фактор але при деяких задачах все ж таки відпрацьовує не досконало.

У розділі було розглянуто застосування та актуальність дослідження, а також методи обробки та аналізу великих вибірок даних. Було акцентовано погляд на перспективності сучасних розробок для задачі прогнозування цін саме для задачі класифікації методами машинного навчання та інтелектуального аналізу даних. Також розглянуто різні оцінки аналізу даних та сформовано постановку задачі.

РОЗДІЛ 2 ОГЛЯД НАЙВІДОМІШИХ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ ДЛЯ ЗАДАЧІ КЛАСИФІКАЦІЇ

2.1 Постановка задачі класифікації

Що таке задача класифікації було визначено в попередньому розділі, тепер потрібно точно сформулювати чому для нашого випадку це саме вона й що ми хочемо в кінцевому випадку отримати. Як вже було сказано наша вибірка містить дані про мобільні телефони а саме їх технічні характеристики та цінові категорії. Оскільки в даних містяться лише цінові категорії, а не точні дані про ціну ми не можемо інтерпретувати її до задачі регресії, адже реально ціна телефону зазвичай точно не визначена й може коливатись в залежності від точки продажу або відстані від країни виробника тощо. Але це коливання нам не суттєве тому й вірним рішенням було збирати дані у вигляді цінових категорій. І оскільки мета нашого дослідження це знайти домінуючі або не суттєві характеристики які саме впливають ціну ми використаємо задачу класифікації яку намагатимемось вирішити методами штучного інтелекту результатом цього має бути добре навченна модель яка надалі нам може робити прогноз на на нові дані.

Задача класифікації є однією з добре досліджених й найпоширеніших задач прикладної статистики та машинного навчання. Вона на даний час являється найбільш простою для інтерпретації оскільки класифікуючі правила можуть бути описані або сформульовані на природній мові.

Також до даної задачі ми можемо застосувати тільки навчання з учителем, а для цього нам потрібно розділити нашу вибірку на навчальну та тестову зазвичай найкращий результат показує розбиття на дані вибірки в співвідношенні 80 % та 20%. Оскільки нам потрібно щоб модель навчалась на якомога більше даних, але повністю всю вибірку ми віддати не можемо через те, що оцінка якості роботи моделі буде не точною.

Теж можна відмітити що дана задача широко використовується в цьому напрямі й з робіт інших досліджень можна явно зразу вибрати собі фаворитів які на практиці вже використовуються, але все рівно потрібно перевірити роботу інших методів і обгрунтувати чому для нашого випадку цей алгоритм не є найкращим. Так як у нас вибірка доволі малих розмір і час на навчання у нас буде невеликий ми можемо це собі дозволити, але у випадку з великими даними потрібно одразу вибирати явно фаворитів наприклад штук три і вже з ними працювати, а не перевіряти всі можливі рішення для вирішення задачі. У останньому випадку нам це може дорого коштувати й ніхто не захоче витрачати свій час та технічний ресурс на дослідження яке всерівно не принесе корисної інформації, а тільки підтвердить напрацювання попередників.

Отже в наступних підрозділах будемо розглядати всі можливі алгоритми рішень задачі класифікації які нам надали у відкритому доступі одна з найбільших бібліотек для машинного навчання Scikit-learn яка до речі нормально написана з більш менш доброю документацією в порівнянні з іншими аналогами. Наведемо переваги та недоліки їх роботи.

2.1.1 Наївний Байєсівський класифікатор

Наївні байєсівські класифікатори - це методи статичної класифікації основані на теоремі Байєса. Являється одним із найпростіших алгоритмів навчання з вчителем. Характеризується як швидкий, точний і надійний при роботі з великими наборами даних. В його основну ідею входить те що ефект відповідної функції в класі не залежить від інших функцій. Навіть якщо ці функції взаємно залежні вони все рівно розглядатимуться як незалежні. Ця гіпотеза набагато спрощує складні обрахунки із-за цього й

цей алгоритм отримав свою назву як наївний. Також цю гіпотезу називають умовною незалежністю від класса.

Розраховується наступним чином:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

В алгоритмі можна виділити чотири основні кроки:

- вирахувати апріорну ймовірність для міток даного класу
- знайти ймовірність правдоподібності для кожного атрибуту кожного класу;
- з попередніх значень за формулою Байєса обчислити апостеріорну ймовірність;
- визначити який клас має саму найбільшу ймовірність для даного випадку;

До складу наївних класифікаторів входять гаусівський, поліноміальний та бернуллі. Основними різницями між ними є те що:

- поліноміальний розглядає вектор ознак який залежить від кількості раз повторення елементу тобто частоту;
- бернуллі це бінарний алгоритм використаний в незалежності від того чи є функція чи ні;
- гаусівський базується на гаусівському непервному розподілу;

До негативної сторони даної групи алгоритмів відносять те що:

- на практиці майже ніколи неможливо щоб модель мала набір повністю незалежних предикатів. Із-за цього головна гіпотеза є неправдивою що значно сказується на роботі моделі;
- якщо немає набору набору даних відповідного класу це призводить до нульової апостеріорної ймовірності. Із-за цього модель не може робити прогнози;

2.1.2 Логістична регресія

Оскільки модель лінійної регресії погано підходить для задачі класифікації із-за того що прогнозований нею результат це не ймовірність, а лінійна інтерполяція між точками то не існує значення порога при якому ми можемо відрізнити один клас від іншого. Ще й до цього ми можемо її примінити тільки на задачах класифікації з двома класами. Рішенням для класифікації стала логістична регресія. Замість апроксимації прямої чи гіперплощини в її основу входить логістична функція яка зжимає вихідні дані лінійного рівняння в діапазон від 0 до 1 й розраховується за формулою:

$$\text{logistic}(\eta) = \frac{1}{1 + exp(-\eta)}$$

Шаг від лінійної регресії до логістичної дуже простий. В моделі лінійної регресії ми змоделювали взаємозв'язок між результатом і характеристиками за допомогою лінійного рівняння:

$$\hat{y}^{(i)} = \beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_p x_p^{(i)}$$

Для класифікації ми надаємо перевагу ймовірності від 0 до 1, із-за цього ми заключаємо праву частину нашого рівняння в логістичну функцію яка визначається як:

$$P(y^{(i)} = 1) = \frac{1}{1 + exp(-(\beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_p x_p^{(i)}))}$$

Багато переваг та недоліків моделі лінійної регресій також містять в собі й логістична регресія. Ще один недолік логістичної регресії в тому що етап інтерпретації більш складний оскільки сама інтерпретація являється мультиплікативною а не аддитивной. Також в ній є проблема повного розбиття коли є функція яка добре розбиває на два класи модель вже не може далі навчатися. Це пов'язано з тим що вага для даної функції буде безкінечною. Така проблема може бути вирішена шляхом введення штрафу за ваги або визначення апріорного розподілу ймовірності ваг. До переваг можна віднести те що модель надає нам не тільки класифікацію а ще ймовірність з якою вона класифікує даний компонент.

2.1.3 Метод опорних векторів (SVM)

Ще один простий алгоритм який часто використовувати адже він може забезпечити високу точність при малій ресурсозатратності. Використовується як для задач регресії так і для задач класифікації, але здебільшого має пріоритет для останнього. Мета даного алгоритму знайти гіперплощину в N-мірному просторі (N - кількість функцій), яка добре класифікує точки даних (рис. 2.1) [7].

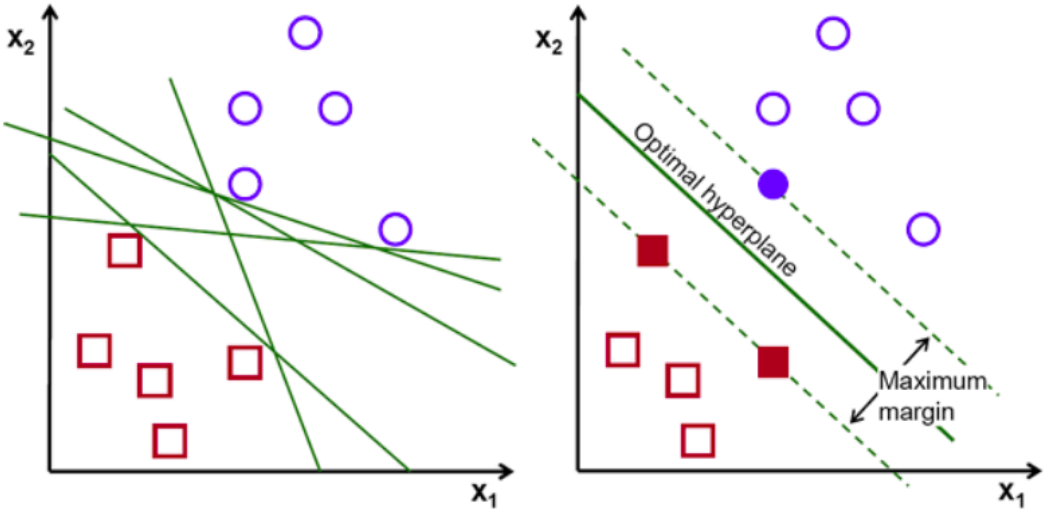


Рисунок 2.1 - Візуалізація роботи метода опорних векторів

Для того щоб розділити два класи точок даних можна вибрати множину можливих гіперплощин. Наша мета - знайти площину с максимальною відстанню між точками даних обох класів.

Гіперплощина та опорні вектори (рис. 2.2) [7]

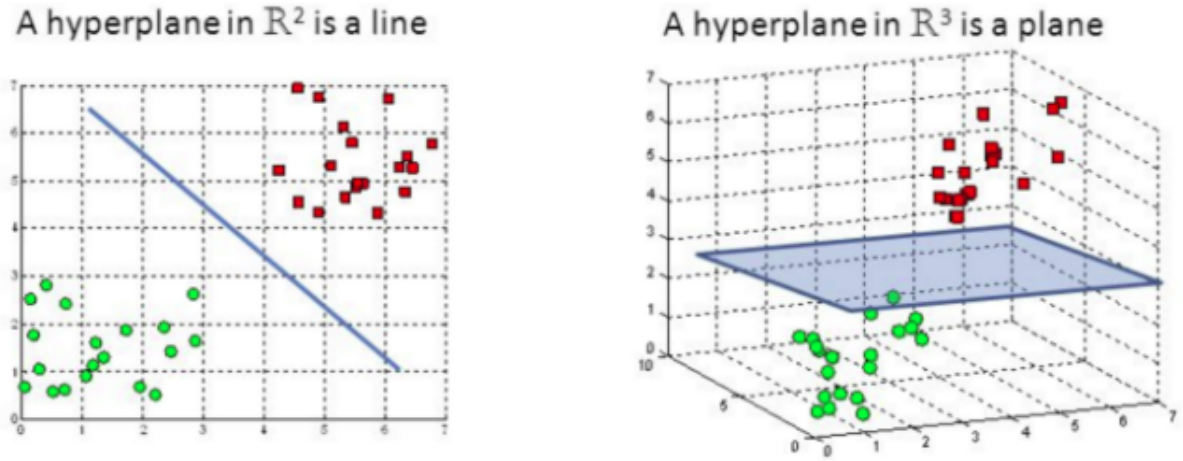


Рисунок 2.2 - Візуалізація гіперплощини яку утворює алгоритм

Гіперплощина - це границя розв'язків які допомагають класифікувати точки даних. Точки даних розташовані по обидва сторонам від неї можна віднести до різних класів. Якщо кількість вхідних об'єктів рівно 2 то гіперплощина представляє собою лінію. Якщо кількість об'єктів

рівно 3 то гіперплощина становиться двовимірній і так далі до n-1 вимірної площини (рис. 2.3) [7].

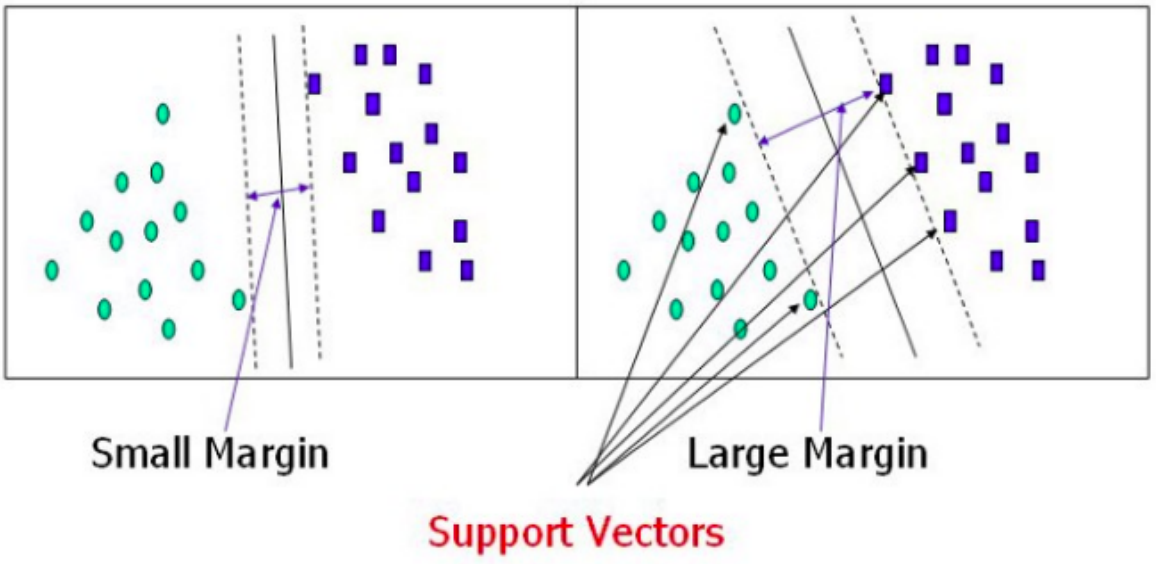


Рисунок 2.3 - Візуалізація виміру відстаней до роздільної гіперплощини

Опорні вектори - це точки даних які знаходяться найближче до гіперплощини і впливають на розташування й кут нахилу площини. Використовуючи ці опорні вектори ми збільшуємо запас класифікатора. Видалення опорних векторів змінить положення гіперплощини.

До переваг даної моделі можна віднести те що вона добре працює з чітким розбиттям, ефективна на даних з багатьма характеристиками, із-за використання підмножини навчальних точок в функції прийняття рішення ефективний з точки зору пам'яті. До недоліків відносять погано роботу на великих даних із-за великого часу навчання, погано відпрацьовує якщо в дані мають багато викидів, не дає напряму оцінку ймовірностей вони вираховуються за допомогою ресурсозатратної перевірки.

2.1.4 Дискримінантний аналіз

Методи дискримінантний аналізу умовно можна розділити на: лінійний дискримінантний аналіз, дискримінантний аналіз Фішера та квадратичний дискримінантний аналіз.

Лінійний дискримінантний аналіз (ЛДА), а також пов'язаний з ним лінійний дискримінант Фішера - методи статистики та машинного навчання для знаходження лінійних комбінацій признаків які найкращим чином розбивають два чи більше класів елементів чи подій. В результаті цього отримана комбінація може бути використана в якості лінійного класифікатора або для зменшення розмірності простору характеристик перед наступною класифікацією.

Дискримінантний аналіз Фішера - це просто LDA в ситуації двух класів. Коли є тільки два класи обраховані вручну и аналіз напряму пов'язаний з множинною регресією. LDA являється прямим розширенням ідеє Фішера про ситуацію любої кількості числа класів і використовує методи матричної алгебри такі як власне розкладання. Таким чином термін дискримінантний аналіз Фішера можна вважати застарівшим. Фішер використав так звані функції класифікації Фішера для класифікації елементів після обрахування дискримінантної функції. На даний момент для класифікації елементів використовується більш загальний байєсівський підхід в рамках процедури LDA.

Також LDA потребує припущення про рівні дисперсійно-коваріаційні матриці між вхідними змінними класів. Це припущення важливе на стадії аналізу. Якщо матриці значно відрізняються, елементи будуть відноситись до класу де змінність більша. Щоб вирішити дану проблему був винайдений метод квадратичного дисперсійного аналізу QDA. QDA - це модифікація LDA яка враховує вище вказану неоднорідність матриць класів.

Якщо у нас є неоднорідність і немає можливості використати QDA, ми все рівно можемо використати LDA в режимі використання окремих коваріаційних матриць, а не об'єднаної. Це частково вирішує проблему, хоча і менш ефективно, ніж в QDA, тому що це матриці між дискримінантами, а не вихідними змінними.

ЛДА являє собою розділ багатовимірного статистичного аналізу, вмістом якого являється розробка методів рішення задач відмінності (дискримінації) елементів спостереження по набору характеристик. Іншими словами, він дозволяє дослідити відмінності між двома або більшою кількістю класів елементів по деяким характеристикам водночас.

ЛДА тісно пов'язаний з дисперсійним та регресійним аналізом, також намагаючись виділити яку небудь змінну через лінійну комбінацію інших признаків або вимірів. В цих двух методах залежна змінна - чисельна величина, а в ЛДА вона являється номінальною величиною тобто метрикою класу. Крім цього, ЛДА має схожість с методом головних компонент та факторіальним аналізом, які шукають лінійні комбінації величин, найкращим чином описуючи дані.

Можно виділити три вида задач дискримінантного аналізу:

- визначення дискримінуючих признаків тобто це ті ознаки які дозволяють віднести віднести елемент до відповідної групи;
- побудова дискримінуючої функції;
- прогнозування майбутніх подій, пов'язаних з потраплянням об'єкта в той чи інший клас на основі значень його характеристик;

Основною ціллю дискримінації являється пошук лінійної комбінації характеристик які називають дискримінантними ознаками, які дозволили би найкращим чином розділити розглянуті класи.

ЛДА широко використовується для вирішення задач класифікації і розпізнавання об'єктів, пониження розмірності вхідних даних. Хоч він і працює з інформацією, яка описує належність елемента до одного з класів,

но сам по собі класифікатором не являється, а використовується як частина лінійної класифікаційної моделі.

До переваг метода можна віднести порівнянну простоту реалізації та інтерпретованості результатів. З недоліків можна відмітити чутливість до розподілу вихідних даних, коли навіть не велика їх зміна призводить до значущих змін результатів класифікації.

2.1.5 Беггінг та Випадковий ліс (Random Forest)

Характеризується як гибкий і простий алгоритм. Алгоритм будує сукупність дерев рішень навчаних методом “bagging”. Головна ідея даного методу в тому що об'єднання навчених моделей покращує результат тобто випадковий ліс будує кілька дерев рішень й об'єднує їх разом щоб отримати більш точніший та стабільний прогноз. Одним з найбільших переваг випадкового лісу в тому що його можна використовувати й в задачах регресії та класифікації. Цей алгоритм додає додаткову випадковість на етапі створення нового дерева. Замість того щоб шукати найбільш важливу функцію при розділі вузла, алгоритм шукає кращу у випадковій множині. Це призводить до більшої варіації моделі що зазвичай супроводжується її покращенням.

Із цього можна зробити висновок, що тільки випадкові признаки враховуються на етапі розділу вузла. Ще одной гарною характеристикою даного алгоритму являється те що доволі просто виміряти відносну важливість кожної функції для прогнозу. Подивившись на важливість функції ми можемо вирішити які з них потрібно видалити а які залишити.

Це важливо адже чим більше не потрібних функцій тим вища вірогідність що наша модель постраждає від перенавчання. Одне з найбільших переваг випадкового лісу це його універсальність й

можливість подивитись відносну важливість яку він надає вхідним функціям. А основним недоліком є те що велика кількість дерев може зробити алгоритм дуже повільним та не ефективним для прогнозу. Цей алгоритм швидко навчається але дає повільно прогнози. Для більшої частини випадків використання даного алгоритму ці недоліки нехтуються.

Випадковий ліс краще описати як інструмент прогнозного моделювання а не інструмент пошуку взаємозв'язків у даних що на перший погляд може здатися гарною ідеєю.

2.1.6 Бустинг (Boosting)

На відміну від багатьох моделей машинного навчання які фокусуються на прогнозуванні за допомогою лише однієї моделі алгоритми boosting намагаються покращувати силу прогнозування шляхом навчання послідовності слабких моделей кожна з яких компенсує недоліки своїх попередників (рис. 2.4) [15].

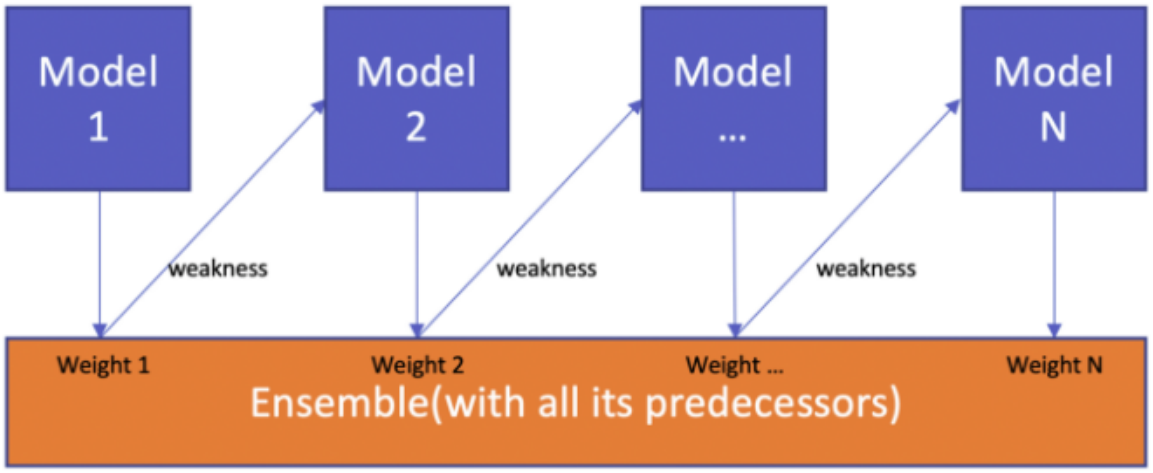


Рисунок 2.4 - Візуалізація мети алгоритма бустінга

Для того щоб зрозуміти що таке boosting важливо пам'ятати що boosting це загальний алгоритм а не конкретна модель. На першому етапі потрібно визначити слабку модель а вже потім покращувати її. В цих алгоритмах виділяють дві основні моделі :

AdaBoost - це спеціальний алгоритм розроблений для задач класифікації. На кожній ітерації алгоритм ідентифікує неправильно класифіковані точки даних і для них збільшує ваги при цьому зменшуючи у правильно класифікованих. Коли слабкі місця виявлено наступним кроком виконується об'єднання послідовності моделей щоб зробити ансамбль сильнішим з часом. AdaBoost навчає послідовність моделей зі збільшеними вагами вибірки, генеруючи коефіцієнти достовірності Alpha для окремих класифікаторів на основі похибок. Низькі оцінки призводять до великого значення Alpha що значить більш високу важливість при відборі.

Gradient Boosting - дана модель вирішує дану проблему по іншому. Замість того щоб коректувати ваги точок даних вона фокусується на різниці між прогнозним та істинним значенням. Модель потребує функції диференційованих втрат і працює як для класифікації так і для регресії. Для функції втрат в основному використовують метод найменших квадратів.

До переваг даної групи моделей належить: стійкість до перенавчання, легкість в інтерпретації, краща прогнозованість на відміну від всіх інших ансамблів. До недоліків можна віднести: чутливість к викидам оскільки кожний слабкий класифікатор назначений для виправлення недоліків своїх попередників із-за цього модель може виділяти дуже багато часу викидам. Також володіє поганою масштабованістю а саме коли кожна оцінка побудована на своїх попередниках процес дуже важко розпаралелити.

2.1.7 K-найближчих сусідів (Nearest Neighbors KNN)

Алгоритм к-ближніх сусідів - це простий алгоритм, який легко реалізується, відноситься до алгоритмів навчання з вчителем, підходить як для задач класифікації так і для регресії. Основна гіпотеза алгоритму в тому що схожі елементи знаходяться безпосередньо в близькості один до одного.

Алгоритм має наступні етапи:

а) вибрати значення параметру k та вибрати відповідно k елементів

б) виміряти відстані між вибраним елементом та всіма іншими

г) відсортувати всі виміряні відстані в порядку зростання.

д) вибрати перші k записів із відсортованої колекції

е) отримати мітки вибраних на попередньому кроці k записів й визначити яка була найчастішою;

Для вибору кращого значення параметру k ми запускаємо алгоритм декілька разів з різними значеннями цього параметру й вибираємо те значення при якому буде найменше помилок зберігаючи спроможність алгоритму точно робити прогнози коли йому трапляються дані які він ще не зустрічав. Перевагами даного алгоритму являються зручність та легкість в реалізації, відсутність необхідності довго будувати та настроювати багато параметрів або робити додаткові гіпотези, універсальність алгоритму. З недоліків є його повільність по мірі збільшення кількості елементів в датасете та характеристик.

2.1.8 Стохастичний градієнтний спуск (Stochastic Gradient Descent)

Стохастичний градієнтний спуск - дуже популярний і розповсюджений алгоритм який використовують в різних алгоритмах машиного навчанняю.

Це ітераційний алгоритм який починається із випадкової точки функції і поступово спускається вниз поки не досягне самої нижньої точки цієї функції. Його потрібно розглядати в якості оптимізатора для ефективного навчання лінійних класифікаторів при великої кількості даних та характеристик. Також гарно підходить для постійного навчання наприклад у випадках коли нам необхідно щоб алгоритм динамічно адаптувався до нових шаблонів даних. В інших випадках можливо будуть краще інші класифікатори.

Даний класифікатор чутливий до масштабування характеристик і потребує точного настроювання ряда гіперпараметрів, включає параметр регуляризації і кількості ітерацій для гарної продуктивності. Нам завжди потрібно використовувати нормалізацію функцій і метод пошуку по сітці щоб знайти найбільш оптимальні гіперпараметри.

2.1.9 Дерева прийняття рішень (Decision Tree Classifier)

Дерево рішень - це доволі простий алгоритм який вирішує задачу класифікації та регресії. Головна його ідея неперервно розбити даних відповідно до деяких параметрів.

Дерево рішень складається із:

- вузлів: перевірка значення відповідної характеристики
- ребра/ гілки відповідає результату теста на вузлі та направляє перехід на інший вузол;
- кінечні вузли вузли які прогнозують результат тобто при задачі класифікації відповідатимуть класам;

Дерева рішень будуються за допомогою евристиці яка називається рекурсивним розбиттям. Цей підхід широко відомий як розбивай та господарюй, із-за того що він розбиває дані на підмножини які потім ще

багато раз розбиваються на ще більш мілкіші і так далі поки процес не зупиниться коли процес вирішить, що дані в підмножині доволі однорідні чи досягнут інший категорій зупинки.

Базовий алгоритм складається з наступних етапів:

а) вибираємо характеристику та тест для неї. Створюємо гілки для кожного ймовірного результату;

б) розбити екземпляри на підмножини відповідно до теста на вузлі.

в) повторюємо це рекурсивно для кожної гілки використовуючи підмножини які досягли до певної гілки;

г) зупинити рекурсію гілки якщо дана підмножину буде складатись з одного класу;

Переваги дерев рішень полягають в тому що їх не дорого будувати, швидко класифікують невідомі записи, легко інтерпретуються для невеликих дерев, доволі висока точність на невеликих та простих даних в порівнянні з іншими методами класифікації також даний алгоритм відкидає не важливі характеристики датасету. До недоліків можна віднести їхню схильність до перенавчання, склоність розбиття по функціям з великим рівнями складності, не стійкість до невеликих змін у навчальній вибірці, складність при інтерпретації великих дерев.

2.2 Метрики оцінювання роботи алгоритмів

Для вибору кращої моделі машинного навчання будемо використовувати метрики які описані нище в розділі.

Classification Accuracy - відношення яке в чисельнику містить кількість правильних прогнозів до всієї вибірки над якої робиться прогноз.

Розраховується як

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

Метрика яка добре описує якість моделі лише якщо у вибірці немає домінуючого класса тобто вибірка повинна складатись з приблизно однакової за обсягом елементів у кожному класі.

Логарифмічна втрата (Logarithmic Loss)

Головна мета цієї метрики позбутися неправдивих класифікацій. Гарно підходить коли класифікація відбувається на декілька груп. Метрика дає імовірнісну оцінку кожної із груп на які відбувається класифікація.

$$LogarithmicLoss = \frac{-1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} * \log(p_{ij})$$

Значення даної метрики не обмежена зверху та має нижнє значення 0. Чим ближче оцінка до 0, тим краща робота моделі. Зазвичай дану метрику використовують лише в задачах класифікацій.

Матриця похибок (Confusion Matrix)

Матриця яка використовується для точної оцінки моделі. Наприклад у нас є задача класифікація на два класи: “Yes”, “No”. На вибірці яка складається з 165 записів і наша метрика видає наступний результат

Таблиця 2.1 - Приклад матриці похибок

	Predicted: “No”	**Predicted: “Yes”**
Actual: “No”	50	10
Actual: “Yes”	5	100

З цього результату можна зробити наступні висновки:

- True Positives: випадок коли модель класифікувала елемент до класу “Yes” і це було вірно;
- True Negatives: випадок коли модель класифікувала до класу “Yes”, а насправді він належить до класу “No”;
- False Positives: випадок коли модель класифікувала елемент до класу “No” і це було вірно;
- False Negatives: випадок коли модель класифікувала до класу “No”, а насправді він належить до класу “Yes”;

Таким чином точність вираховується за формулою:

$$Accuracy = \frac{TruePositive + TrueNegative}{TotalSample}$$

F-міра (F1 Score)

Метрика F1 вимірює точність моделі на тестових даних. Значення даної метрики належить діапазону від 0 до 1, показує наскільки надійний прогноз моделі и обраховується по наступній формулі:

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

Метою даної метрики являється знаходження відношення між метрикою precision та метрикою recall які в свою чергу визначаються як :

- відношення правильно прогнозованих випадків до суми всіх випадків класифікованих так само. Розраховується як:

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

- відношення правильно класифікованих випадків до справжньої кількості цього класу. Розраховується як:

$$Precision = \frac{TruePositives}{TruePositives + FalseNegatives}$$

Метрика AUC

В основному використовують для задач класифікації. Вираховується як ймовірність того що позитивний випадок буде на ранжований вище чим коли алгоритм неправильно класифікує. Для визначення даної метрики потрібно точно визначити наступні поняття:

- правдивий позитивний коефіцієнт або його інша назва коефіцієнт чутливості: визначається як TP / (FN + TP). Цей коефіцієнт показує долю точок позитивних даних які правильно вважаються позитивними до відношення всіх точок позитивних даних. Розраховується як:

$$TruePositiveRate = \frac{TruePositive}{FalseNegative + TruePositive}$$

- правдивий негативний коефіцієнт або його також називають специфічністю : визначається як TN / (FP + TN) і відповідає долі негативних точок даних які правильно вважаються негативними до відношення до всіх негативних точок даних. Розраховується як:

$$TrueNegativeRate = \frac{TrueNegative}{TrueNegative + FalsePositive}$$

- рівень хибних випадків: визначається як FP / (FP + TN) і відповідає долі від'ємних точок даних які хибно вираховуються по відношенню до всіх від'ємних точок даних. Розраховується як:

$$FalsePositiveRate = \frac{FalsePositive}{TrueNegative + FalsePositive}$$

Частота хибно позитивних та правдиво позитивних результатів має значення в діапазоні [0, 1]. FPR и TPR вираховується при різних порогових значень і будується графік. Метрика AUC - це площа під кривою графіка хибно позитивних результатів і правдиво позитивних в різних точках. Й чим краще значення метрики AUC тим краще відпрацьовує дана модель.
Середня абсолютна похибка

Середня абсолютна похибка - це середнє значення похибки прогнозованих даних. Дає наглядно зрозуміти наскільки точними або не точними були наші прогнози відносно реальних значень. Однак головною проблемою даної метрики вважається те що вона не дає нам поняття про напрямок похибки тобто чи ми прогнозуємо більше значення чи меньше. Математично обчислюється як :

$$MeanAbsoluteError = \frac{1}{N}\sum_{j=1}^{N}|y_j - \hat{y}_j|$$

Середньоквадратична похибка

Середньоквадратична похибка (MSE) подібна до середньо абсолютної але в ній визначається середнє значення квадрату різниці між реальними значеннями та прогнозованими. Головним пріоритетом даної метрики в тому що набагато легше і швидше порахувати градієнт тоді як середня абсолютна похибка потребує тяжких розрахунків лінійного програмування для обрахунків того ж градієнту. Оскільки ми беремо

квадрат похибки то вплив великих похибок буде набагато більшим чим меньших із-за цього модель може зараз більш фокусуватись на великих похибках. Розраховується як:

$$MeanSquaredError = \frac{1}{N}\sum_{j=1}^{N}(y_j - \hat{y}_j)^2$$

2.3 Вибір засобів для програмного продукту

Безпосередньо головним фаворитом для написання даного продукту являється мова програмування Python оскільки для неї написано найбільші бібліотеки для машинного навчання. Маючи доволі високий рівень абстракції та простоти використання даних методів написати програмний продукт доволі просто й незатратно.

2.3.1 Python

Популярність Python швидко зростає протягом останніх кількох років. У 2019 році опитування Stackoverflow показало, що Python є найпопулярнішою мовою програмування та другою за популярністю після Rust. Дані RedMonk за 2020 рік свідчать про те, що Python є другою за популярністю мовою за кількістю проектів на GitHub. Згідно з Індексом TIOBE, Python також піднявся зі свого 23-місця у 2000 році до третьої за популярністю мови програмування всього за двадцять років [16].

Крім того, Python вважається одним з найкращих рішень, коли мова йде про машинне навчання та науку даних.

Сам по собі Python - це звичайна інтерпретована мова програмування. Це не блискавично швидко порівняно з мовами, які компілюються, і це неймовірно важлива причина, що робить його непридатним для багатьох цілей.

Справжня сила Python полягає у величезній кількості бібліотек машинного навчання, доступних для різних завдань. Набір інструментів, який Python пропонує розробникам машинного навчання, надзвичайно гнучкий та здатний до масштабування. Бібліотеки Python не тільки дозволяють розробникам працювати ефективніше, але вони також дозволяють компаніям наймати працівників із набором навичок, який точно відповідає потребам компанії. Подібним чином розробники можуть стати досвідченими в багатьох бібліотеках, але приділяють найбільшу увагу тим, які є найбільш корисними у їхній роботі.

Кількість інструментів машинного навчання, які Python надає в різних бібліотеках, дозволяє розробникам скоротити час, який вони витрачають на кодування, з нуля. Хоча здатність програмувати необхідну функціональність вручну є важливою і може свідчити про майстерність розробника.

Python можна використовувати для найрізноманітніших завдань, що робить його важливим для багатьох компаній, які не спеціалізуються на машинному навчанні. Простота використання та різноманітність зовнішніх фреймворків та бібліотек, які взаємодіють з Python, дозволяють розробникам використовувати мову для науки про дані, створення хмарних сховищ і навіть для розробки настільних та мобільних додатків.

Крім того, розробники часто використовують Python, коли необхідно спочатку створити прототип програми. Оскільки мова легко читається і легко засвоюється, вона витрачає менше часу та інших ресурсів на розробку проекту програми для презентації. Python дозволяє оцінити

функціональність, ефективність та зовнішній вигляд програми, перш ніж реалізувати її іншою мовою.

2.3.2 Scikit-learn

Scikit-learn - один з найбільш широко використовуваних пакетів Python для Data Science та Machine Learning. Він дозволяє виконувати безліч операцій і надає безліч алгоритмів. Scikit-learn також пропонує добру документацію про використання своїх класів, методів і функцій, а також гарний опис використовуваних алгоритмів.

Scikit-Learn підтримує:

- попередню обробку даних
- зменшення розмірності
- вибір моделі
- регресії
- класифікації
- кластерний аналіз

Він також надає кілька наборів даних, які ви можна використовувати для тестування моделей.

Але scikit-learn не реалізує все, що пов'язано з машинним навчанням. Наприклад, він не має комплексної підтримки для:

- нейронних мереж
- самоорганізованих карт (мереж Кохонена)
- навчання асоціативним правилам
- навчання з підкріпленням (reinforcement learning)

Scikit-learn побудований на NumPy і SciPy, тому необхідно зрозуміти хоча б ази цих двох бібліотек, щоб ефективно застосовувати Scikit-learn.

Scikit-learn - це пакет з відкритим вихідним кодом. Як і більшість матеріалів з екосистеми Python, він безкоштовний навіть для комерційного використання. Він ліцензований під ліцензією BSD.

2.3.3 Seaborn

Візуалізація даних - це метод, який дозволяє фахівцям з аналізу даних перетворювати сирі дані в діаграми і графіки, які несуть цінну інформацію. Діаграми зменшують складність даних і роблять більш зрозумілими для будь-якого користувача.

Є безліч інструментів для візуалізації даних, таких як Tableau, Power BI, ChartBlocks та інших, які є no-code інструментами. Вони дуже потужні, і у кожного своя аудиторія. Однак для роботи з сирими даними, які вимагають обробки, а також в якості пісочниці, Python підійде найкраще.

Незважаючи на те, що цей шлях складніший і вимагає вміння програмувати, Python дозволить вам провести будь-які маніпуляції, перетворення і візуалізувати ваші дані. Він ідеально підходить для фахівців з аналізу даних.

Python - кращий інструмент для data science і цього багато причин, але найважливіша - це його екосистема бібліотек. Для роботи з даними в Python є багато чудових бібліотек, таких як numpy, pandas, matplotlib, tensorflow.

Matplotlib, ймовірно, найвідоміша бібліотека для побудови графіків, яка доступна в Python та іншими мовами програмування, таких як R. Саме її рівень кастомізації і зручності у використанні ставить її на перше місце.

Однак з деякими діями і кастомізації під час її використання буває впоратися нелегко.

Розробники створили нову бібліотеку на основі matplotlib, яка називається seaborn. Seaborn така ж потужна, як і matplotlib, але в той же час надає велику абстракцію для спрощення графіків і привносить деякі унікальні функції.

Seaborn - це бібліотека для створення статистичних графіків на Python. Вона грунтується на matplotlib і тісно взаємодіє зі структурами даних pandas.

Архітектура Seaborn дозволяє вам швидко вивчити і зрозуміти свої дані. Seaborn захоплює цілі фрейми даних або масиви, в яких містяться всі ваші дані, і виконує всі внутрішні функції, потрібні для семантичного маппінга і статистичної агрегації для перетворення даних в інформативні графіки.

Вона абстрагує складність, дозволяючи вам проектувати графіки відповідно до ваших потреб.

2.3.4 Numpy

NumPy - це основний пакет наукових обчислень на Python. Це бібліотека Python, яка забезпечує багатовимірний об'єкт масиву та різні похідні об'єкти (наприклад, масковані масиви та матриці) та асортимент підпрограм для швидких операцій над масивами, включаючи математичні, логічні, маніпуляції з фігурами, сортування, вибір, введення / виведення , дискретні перетворення Фур'є, основні методи лінійної алгебри, основні статистичні операції, випадкове моделювання та багато іншого.

В основі пакета NumPy лежить об'єкт ndarray . Це інкапсулює *n* -вимірні масиви однорідних типів даних, причому багато операцій

виконуються в скомпільованому коді для продуктивності. Існує кілька важливих відмінностей між масивами NumPy та стандартними послідовностями Python:

- масиви NumPy мають фіксований розмір при створенні, на відміну від списків Python (які можуть динамічно зростати). Зміна розміру ndarray створить новий масив і видалить оригінал;
- всі елементи масиву NumPy повинні мати однаковий тип даних, і, отже, матимуть однаковий розмір у пам'яті. Виняток: можна мати масиви об’єктів (Python, включаючи NumPy), що дозволяє мати масиви елементів різного розміру;
- масиви NumPy сприяють вдосконаленим математичним та іншим типам операцій над великою кількістю даних. Як правило, такі операції виконуються ефективніше і з меншим кодом, ніж це можливо за допомогою вбудованих послідовностей Python;
- зростаюче безліч науково-математичних пакетів на основі Python використовують масиви NumPy, хоча вони, як правило, підтримують введення послідовності Python, вони перетворюють такий вхід у масиви NumPy перед обробкою, і вони часто виводять масиви NumPy. Іншими словами, для ефективного використання більшої частини (можливо, навіть більшості) сучасного науково-математичного програмного забезпечення на основі Python, просто знання того, як використовувати вбудовані типи послідовностей Python, недостатньо - потрібно також знати, як використовувати масиви NumPy;

2.3.5 Pandas

Pandas - це пакет Python, який забезпечує швидкі, гнучкі та виразні структури даних, призначені для полегшення та інтуїтивної роботи зі структурованими (табличними, багатовимірними, потенційно неоднорідними) даними також добре підходить для роботи з даними часових рядів. Його основна мета стати фундаментальним будівельним інструментом високого рівня для практичного аналізу даних у світі на Python. Крім того, розробники даної бібліотеки мають більш широку ціль, а саме стати найпотужнішим та гнучким інструментом аналізу / маніпуляції з відкритим кодом, доступним будь-якою мовою. І вони зробили велику роботу на шляху до цієї мети.

З переваг даного пакета можна відмітити:

- простоту обробку відсутніх даних незважаючи на формат цих даних
- змінюваність розміру, можна динамічно видаляти та вставляти нові стовпці;
- автоматичне та явне вирівнювання даних, тобто об'єкти можуть бути явно вирівняні за набором міток або користувач може просто ігнорувати мітки і дозволити пакету це зробити автоматично;
- потужний та гнучкий функціонал
- легке перетворення з однієї структури даних в іншу
- ієрархічне маркування осей
- надійні інструменти вводу-виводу для завантаження даних з файлів

2.4 Висновки

У даному розділі було розглянуті найпопулярніші алгоритми машинного навчання для задачі класифікації. також хочу помітити, що в

наступному розділі буде розглянуто й порівняння їх роботу на нашій вибірці. Матеріал в основі містить переведенні статті безпосередньо авторів та співучасників які приймали участь в розробці даних технологій. Також розглянуто й обгрунтовані метрики за допомогою яких виявляють якість роботи алгоритмів та порівняють їх між собою. Найчастіше метрики вибираються в залежності від задачі та чого ми хочемо досягти.

В останній частині розділу було розглянуто інструменти за допомогою яких ми і будемо проводити наше дослідження. Беззаперечно дані технології вважаються фаворитами в машинному навчанні, ні одна з інших бібліотек чи мов програмування не мають таких досягненні у даному направленні. Також дані технології допомогли значно зменшити витрати часу на розробку програмного продукту для данного дослідження.

РОЗДІЛ 3 ПОРІВНЯННЯ ТА ОЦІНКА РЕЗУЛЬТАТІВ РОБОТИ АЛГОРИТМІВ

3.1 Попередній аналіз датасету

Датасет був взятий з відкритого джерела. Він містить в собі записи структура яких виглядає наступним чином 20 features variables and 1 target variable. Змінні характеристик складають із 6 бінарних та 14 неперервних.

Бінарні характеристики:

- наявність в пристрої технології Bluetooth (“blue”)
- наявність можливості використовувати в пристрої дві сім карти (“dual_sim”);
- наявність технології 4G (“four_g”)
- наявність технології 3G (“three_g”)
- наявність підтримки технології Wi-Fi (“wi-fi”)
- наявність сенсорного екрану (“touch_screen”)

Значення “1” в бінарній характеристиці говорить нам про наявність певної технології а значення “0” означає її відсутність.

Неперервні характеристики:

- швидкість виконання процесором певної кількості інструкцій (“clock_speed”);
- загальна електроємність яку аккумулятор может втримувати за одну підзарядку в mAh (“battery_power”);
- розширення фронтальна камера в mega pixels (“fc”)
- об'єм внутрішньої пам'яті в гигабайтах (“int_memory”)
- розмір висоти телефону в см (“m_dep”)
- вага мобільного телефону (“mobile_wt”)
- кількість ядрів на одному процесорі (“n_cores”)
- розширення основної камери в мегапікселях (“pc”)
- висота роздільної здатності екрану в пікселях (“px_height”)

- ширина роздільної здатності екрану в пікселях (“px_width”)
- розмір оперативної пам'яті в мегабайтах (“ram”)
- довжина екрана в сантиметрах (“sc_h”)
- ширина экрана в сантиметрах (“sc_w”)
- час на якій вистачає неперервної роботи телефону в стані дзвінку за один заряд аккумулятора (“talk_time”)

Цільова змінна показує до якої цінової категорії відноситься дана модель. Можливі значення:

- значення “0” говорить про належність до категорії низької вартості;
- значення “1” говорить про належність до категорії середньої вартості;
- значення “2” говорить про належність до категорії високої вартості;
- значення “3” говорить про належність до категорії дуже високої вартості;

Перевірка датасету на цілісність та правдивість.

По кожній характеристиці обраховуємо наступні метрики

- середнє значення (mean)
- максимальне значення (max)
- мінімальне значення (min)
- кількість записів (count)
- квартилі (0.25, 0.5, 0.75)
- наявність null значень

Також графічно відобразимо:

а) для неперервних даних:

- графік щільності розподілу як спільний так і для кожної категорії;

- графік ядерної оцінки плотності (Kernel Density Estimation) як спільний так і для кожної категорії;

б) для бінарних даних:

- Гістограму розподілу як спільну так і для кожної категорії.

Бінарні характеристики

Наявність в пристрої технології Bluetooth (табл. 3.1, рис. 3.1).

Таблиця 3.1 - Результати попереднього аналізу по характеристиці Bluetooth

count	mean	std	min	25%	50%	75%	max
2000	0.4950	0.5001	0	0	0	1	1

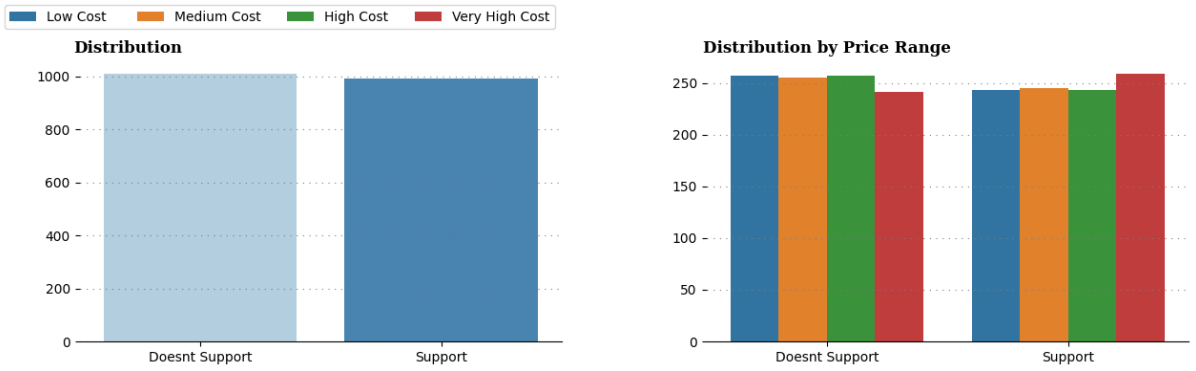


Рисунок 3.1 - Гістограма розподілу по характеристиці Bluetooth

Наявність можливості використовувати в пристрої дві сім карти (табл. 3.2, рис. 3.2).

Таблиця 3.2 - Результати попереднього аналізу по характеристиці двох карт

count	mean	std	min	25%	50%	75%	max
2000	0.50950	0.500035	0	0	1	1	1

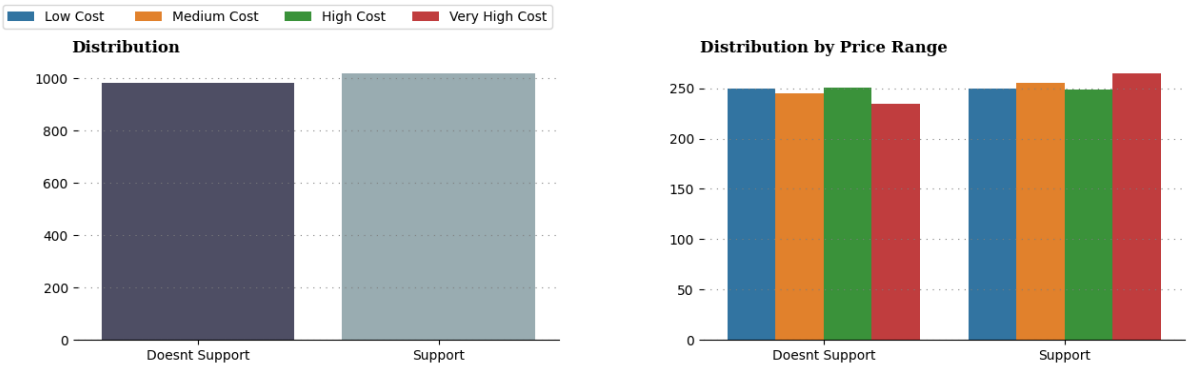


Рисунок 3.2 - Гістограма розподілу по характеристиці двох сім карт

Наявність технології 4G (табл. 3.3, рис. 3.3).

Таблиця 3.3 - Результати попереднього аналізу по характеристиці 4G

count	mean	std	min	25%	50%	75%	max
2000	0.521500	0.499662	0	0	1	1	1

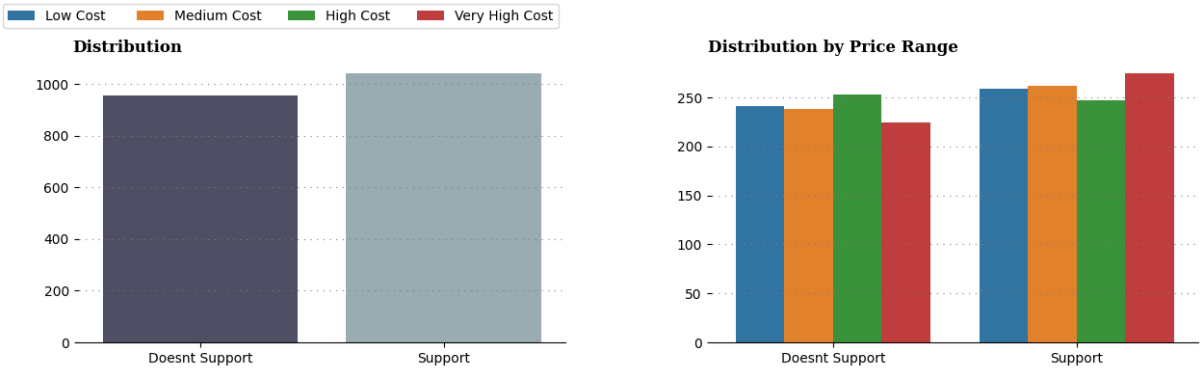


Рисунок 3.3 - Гістограма розподілу по характеристиці 4G

Наявність технології 3G (табл. 3.4, рис. 3.4).

Таблиця 3.4 - Результати попереднього аналізу по характеристиці 3G

count	mean	std	min	25%	50%	75%	max
2000	0.761500	0.426273	0	1	1	1	1

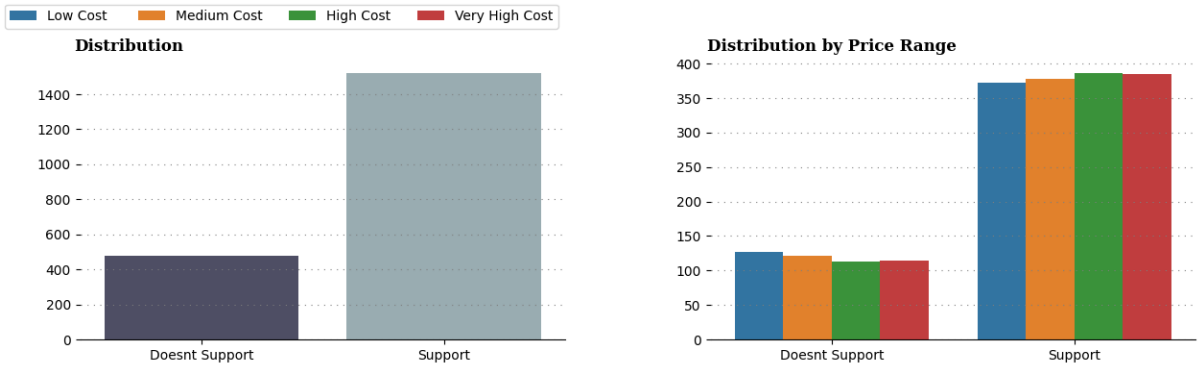


Рисунок 3.4 - Гістограма розподілу по характеристиці 3G

Наявність підтримки технології Wi-Fi (табл. 3.5, рис. 3.5).

Таблиця 3.5 - Результати розподілу по характеристиці Wi-Fi

count	mean	std	min	25%	50%	75%	max
2000	0.507000	0.500076	0	0	1	1	1

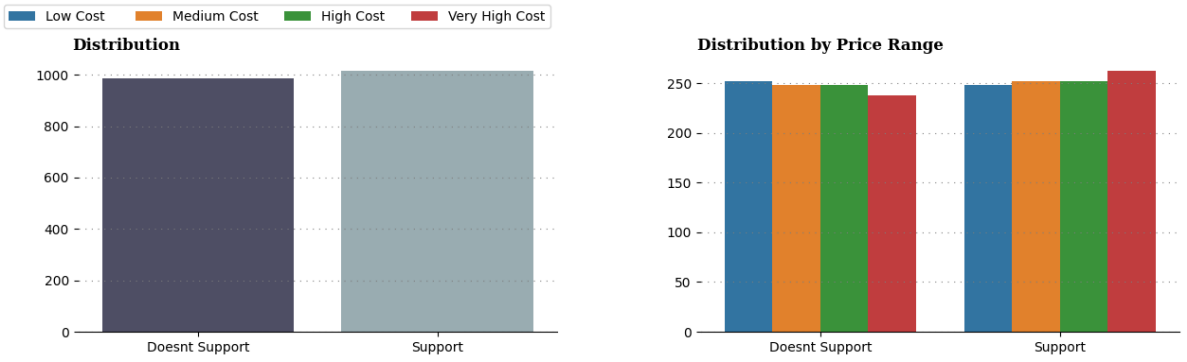


Рисунок 3.5 - Гістограма розподілу по характеристиці Wi-Fi

Наявність сенсорного екрану (табл. 3.6, рис. 3.6).

Таблиця 3.6 - Результати попереднього аналізу по характеристиці екрану

count	mean	std	min	25%	50%	75%	max
2000	0.50300	0.500116	0	0	1	1	1

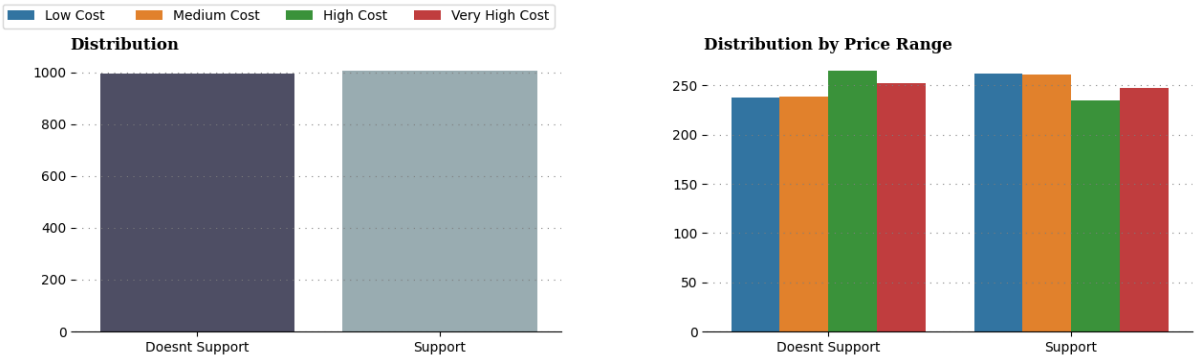


Рисунок 3.6 - Гістограма розподілу по характеристиці сенсорного екрану

Неперервні характеристики

Швидкість виконання процесором певної кількості інструкцій (табл. 3.7, рис. 3.7).

Таблиця 3.7 - Результати аналізу по характеристиці процессора

count	mean	std	min	25%	50%	75%	max
2000	1.522250	0.816004	0.5	0.7	1.5	2.2	3

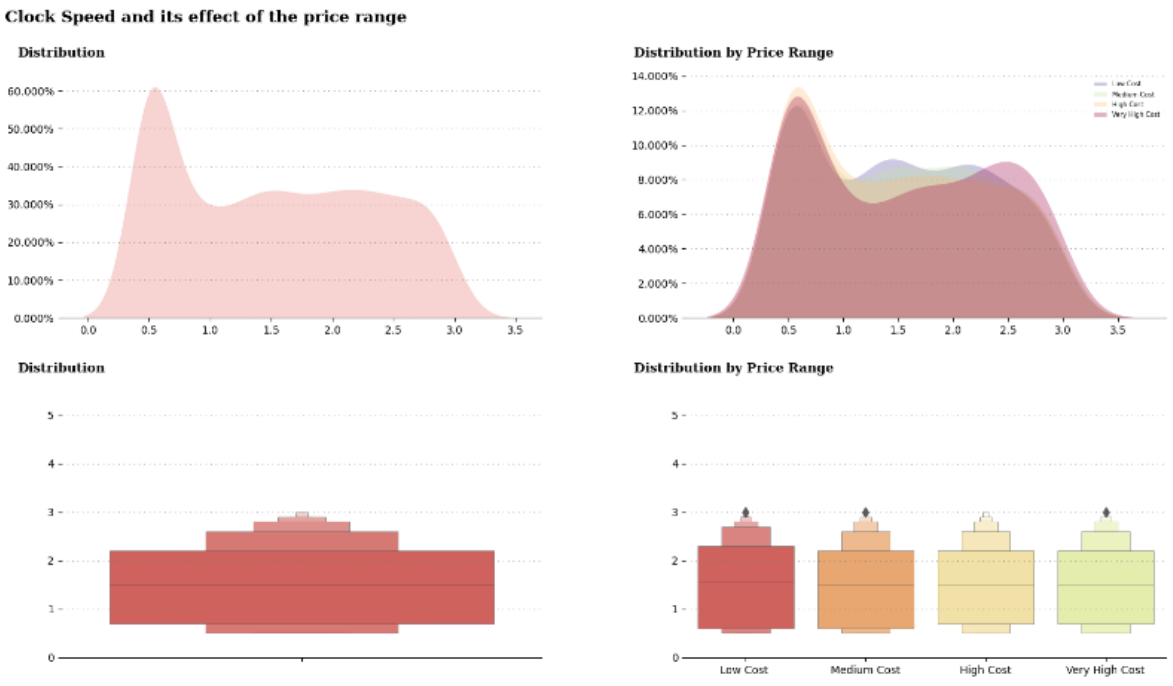


Рисунок 3.7 - Графіки щільності розподілу та оцінки плотності по характеристиці процессора

Загальна електроємність яку аккумулятор может утримувати за одну підзарядку (табл. 3.8, рис. 3.8).

Таблиця 3.8 - Результати аналізу по характеристиці електроємності

count	mean	std	min	25%	50%	75%	max
2000	1238.5185	439.418206	501.00	851.75	1226.00	1615.25	1998.00

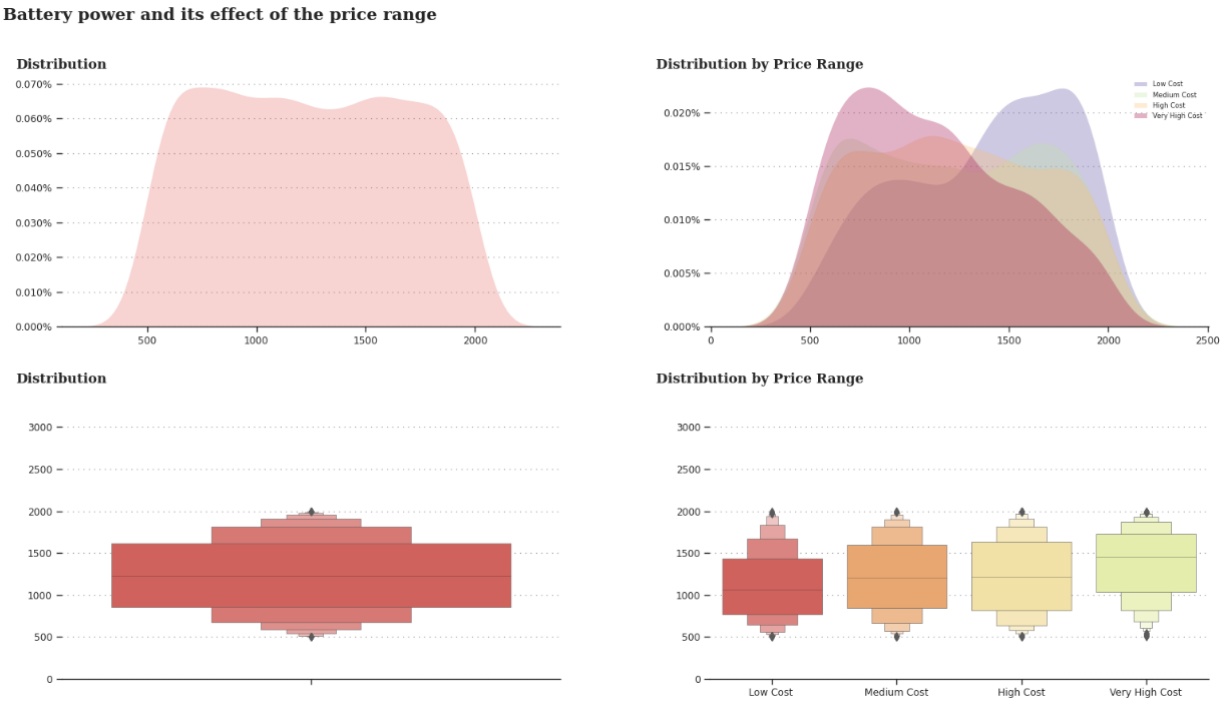


Рисунок 3.8 - Графіки щільності розподілу та оцінки плотності по характеристиці електроємності аккумулятора

Розширення фронтальної камери (табл. 3.9, рис. 3.9).

Таблиця 3.9 - Результати аналізу по характеристиці фронтальної камери

count	mean	std	min	25%	50%	75%	max
2000	4.309500	4.34144	0.00	1.00	3.00	7.00	19.00

Рисунок 3.9 - Графіки щільності розподілу та оцінки плотності по характеристиці фронтальної камери

Об'єм внутрішньої пам'яті (табл. 3.10, рис. 3.10).

Таблиця 3.10 - Результати аналізу по характеристиці внутрішньої пам'яті

count	mean	std	min	25%	50%	75%	max
2000	32.046500	18.145715	2.00	16.00	32.00	48.00	64.00

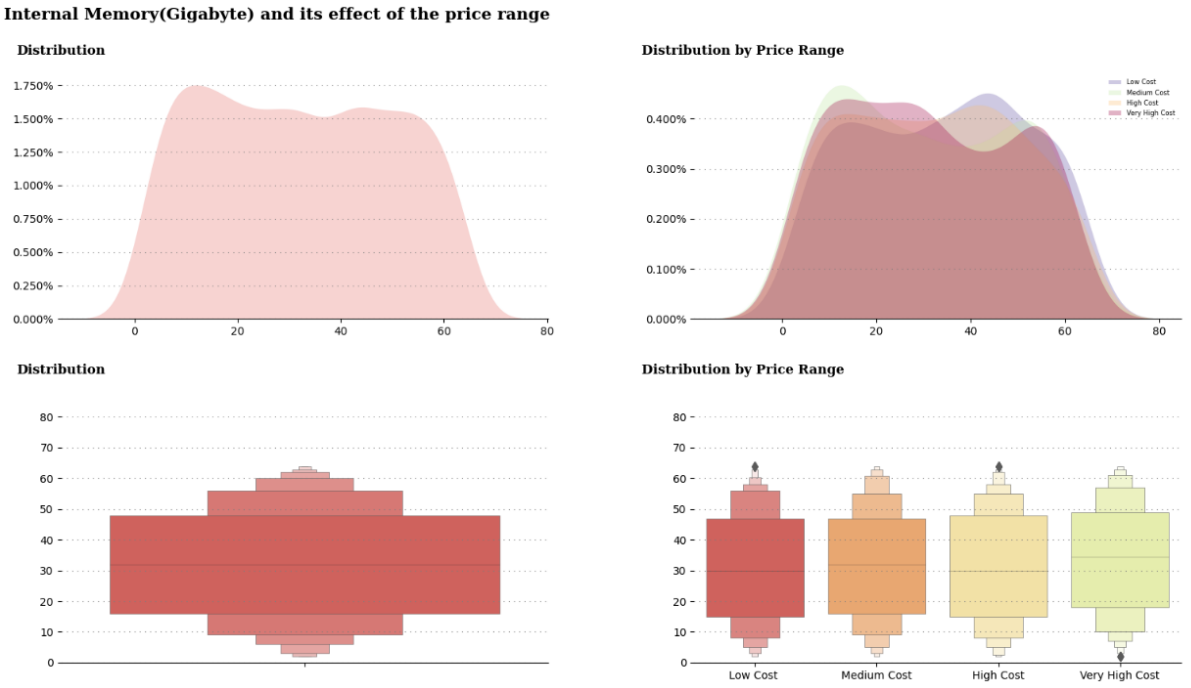


Рисунок 3.10 - Графіки щільності розподілу та оцінки плотності по характеристиці внутрішньої пам'яті

Розмір висоти телефону (табл. 3.11, рис. 3.11).

Таблиця 3.11 - Результати аналізу по характеристиці висоти телефону

count	mean	std	min	25%	50%	75%	max
2000	0.501750	0.288416	0.100	0.200	0.500	0.800	1.00

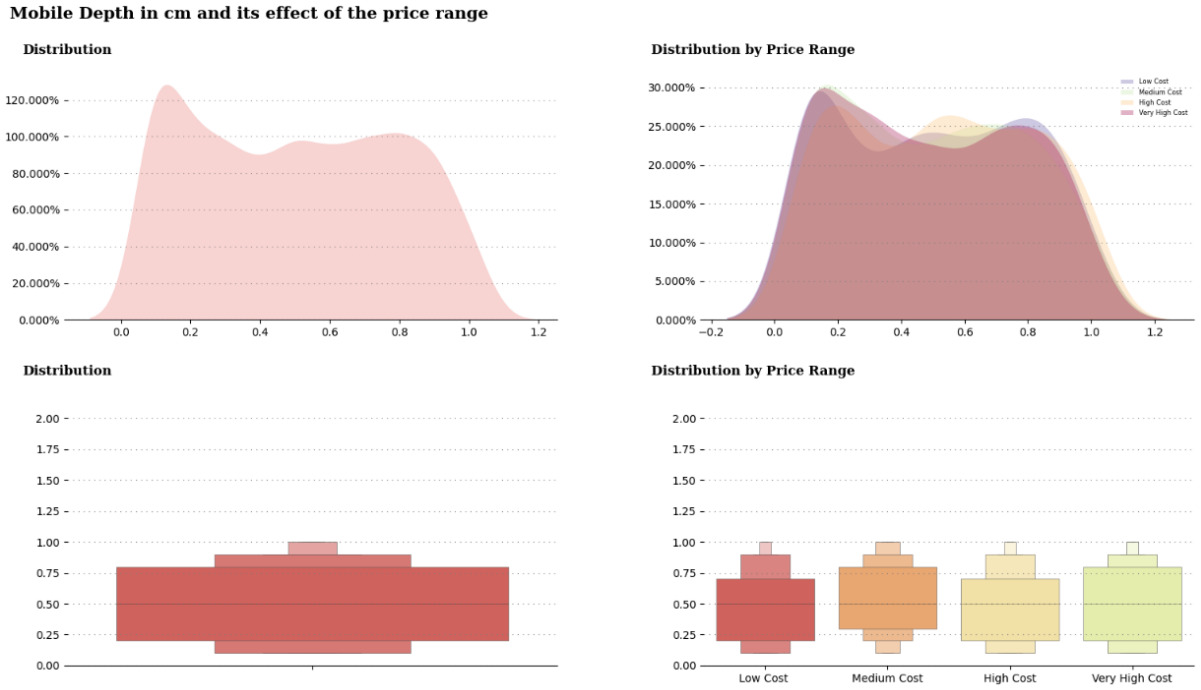


Рисунок 3.11 - Графіки щільності розподілу та оцінки плотності по характеристиці висоти телефону

Вага мобільного телефону (табл. 3.12, рис. 3.12).

Таблиця 3.12 - Результати аналізу по характеристиці ваги телефону

count	mean	std	min	25%	50%	75%	max
2000	140.24900	35.39965	80.00	109.00	141.000	170.00	200.00

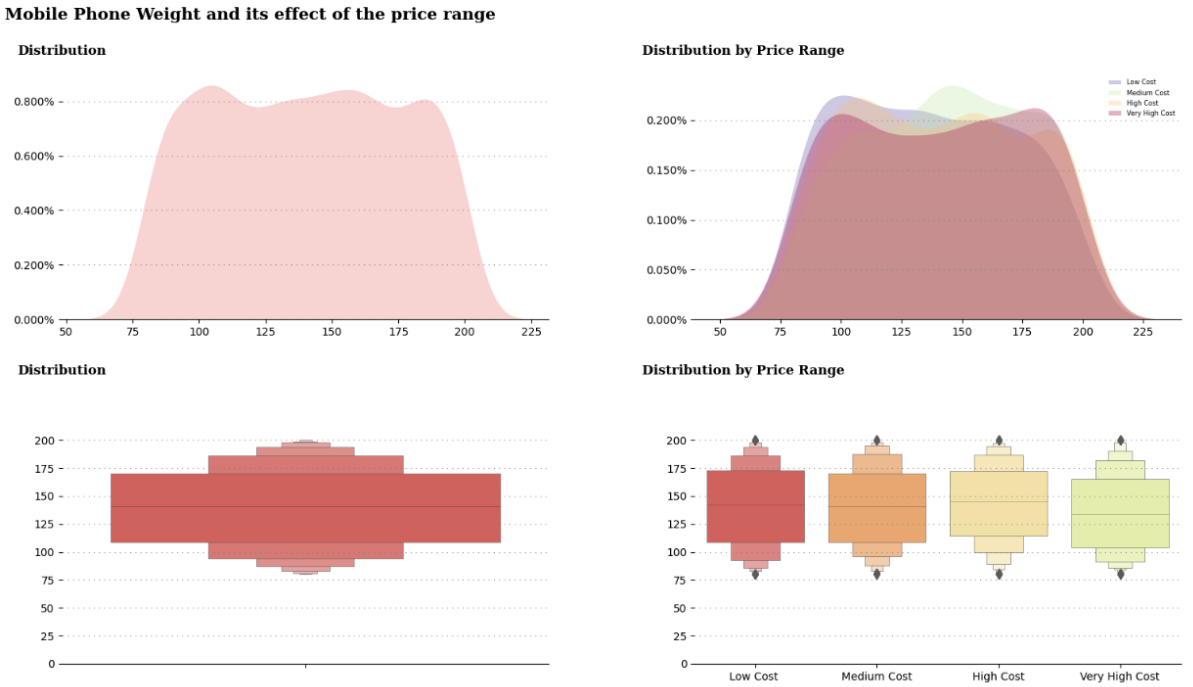


Рисунок 3.12 - Графіки щільності розподілу та оцінки плотності по характеристиці ваги телефону

Кількість ядрів на одному процесорі (табл. 3.13, рис. 3.13).

Таблиця 3.13 - Результати аналізу по характеристиці кількості ядер

count	mean	std	min	25%	50%	75%	max
2000	4.520500	2.287837	1.00	3.00	4.00	7.00	8.00

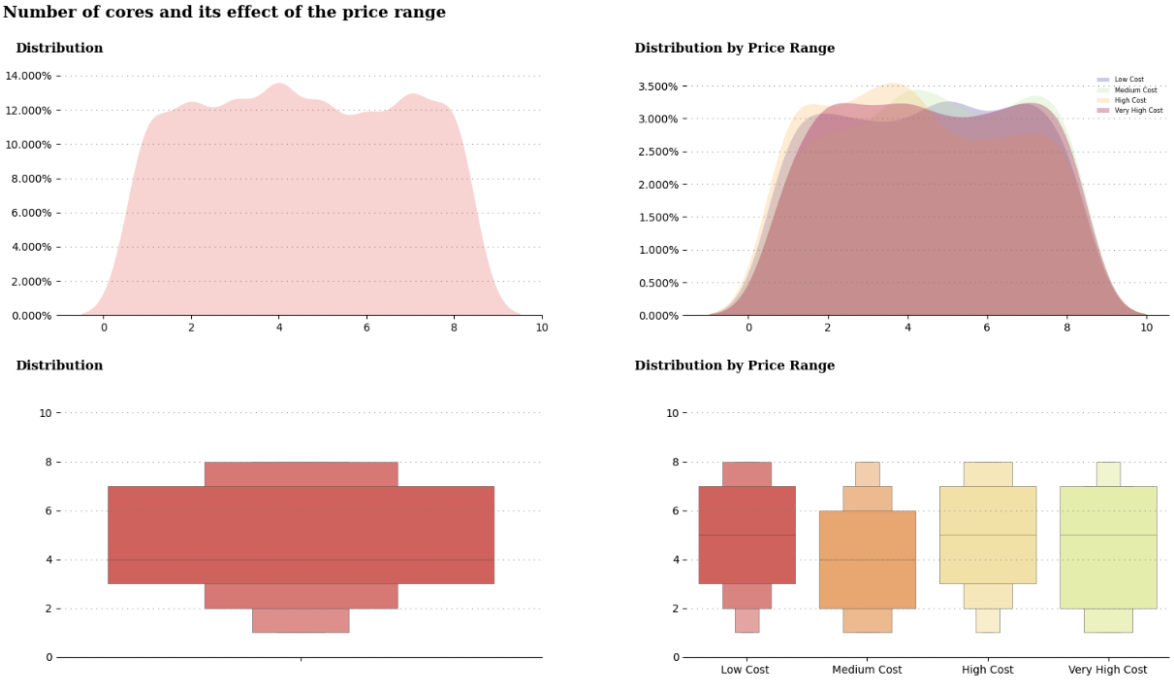


Рисунок 3.13 - Графіки щільності розподілу та оцінки плотності по характеристиці кількості ядер

Розширення основної камери (табл. 3.14, рис. 3.14).

Таблиця 3.14 - Результати аналізу по характеристиці основної камери

count	mean	std	min	25%	50%	75%	max
2000	9.9116500	6.064315	0.00	5.00	10.00	15.00	20.00

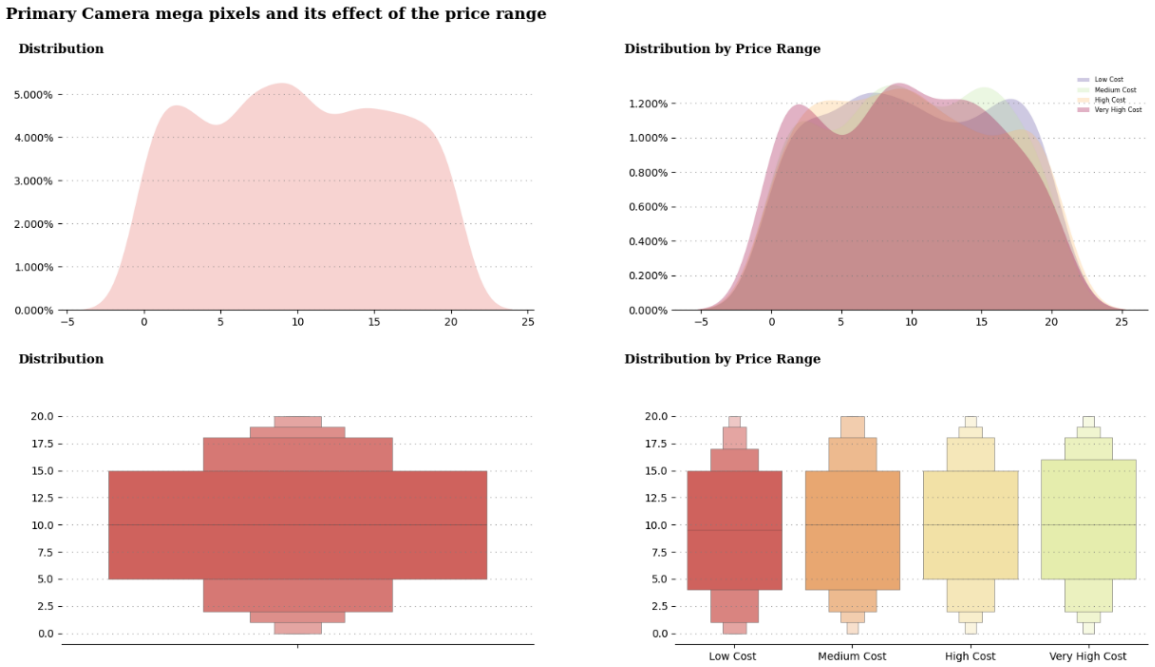


Рисунок 3.14 - Графіки щільності розподілу та оцінки плотності по характеристиці основної камери

Висота роздільної здатності екрану (табл. 3.15, рис. 3.15).

Таблиця 3.15 - Результати аналізу по характеристиці висоти екрану

count	mean	std	min	25%	50%	75%	max
2000	645.10800	443.780811	0.00	282.75	564.00	947.2500	1960.00

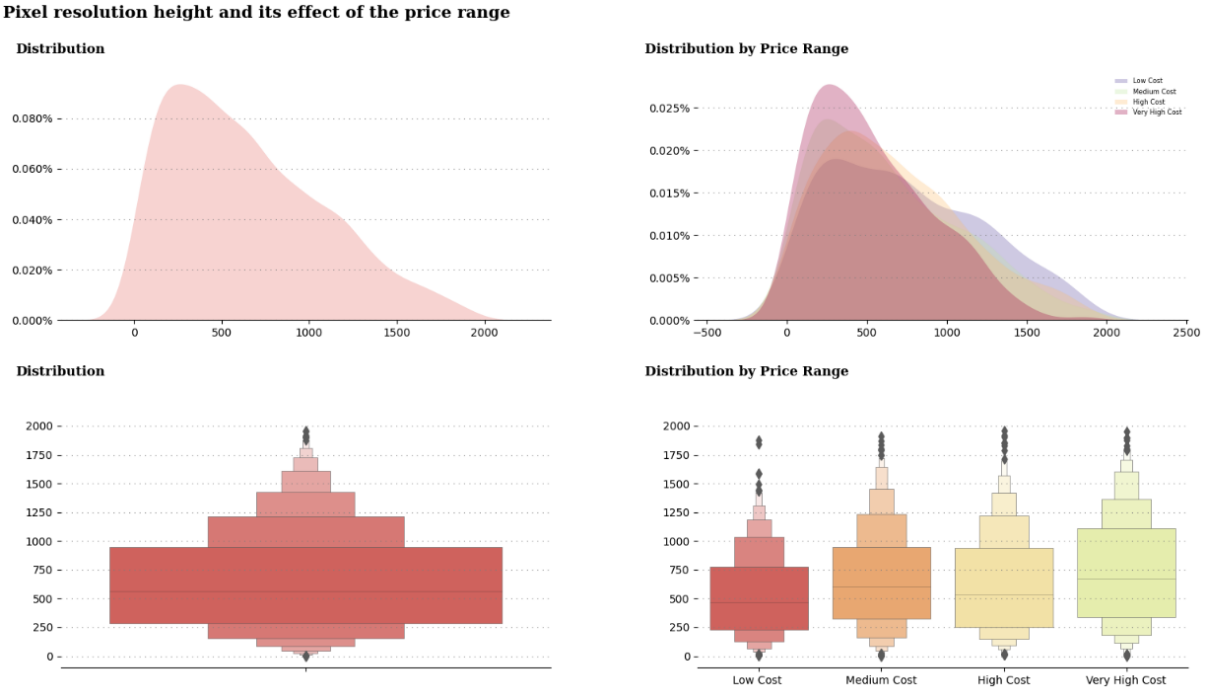


Рисунок 3.15 - Графіки щільності розподілу та оцінки плотності по характеристиці висоти роздільної здатності екрану

Ширина роздільної здатності екрану в пікселях (табл. 3.16, рис. 3.16).

Таблиця 3.16 - Результати аналізу по характеристиці ширина екрану

count	mean	std	min	25%	50%	75%	max
2000	1251.51550	432.199447	500.00	874.750	1247.00	1633.00	1998.00

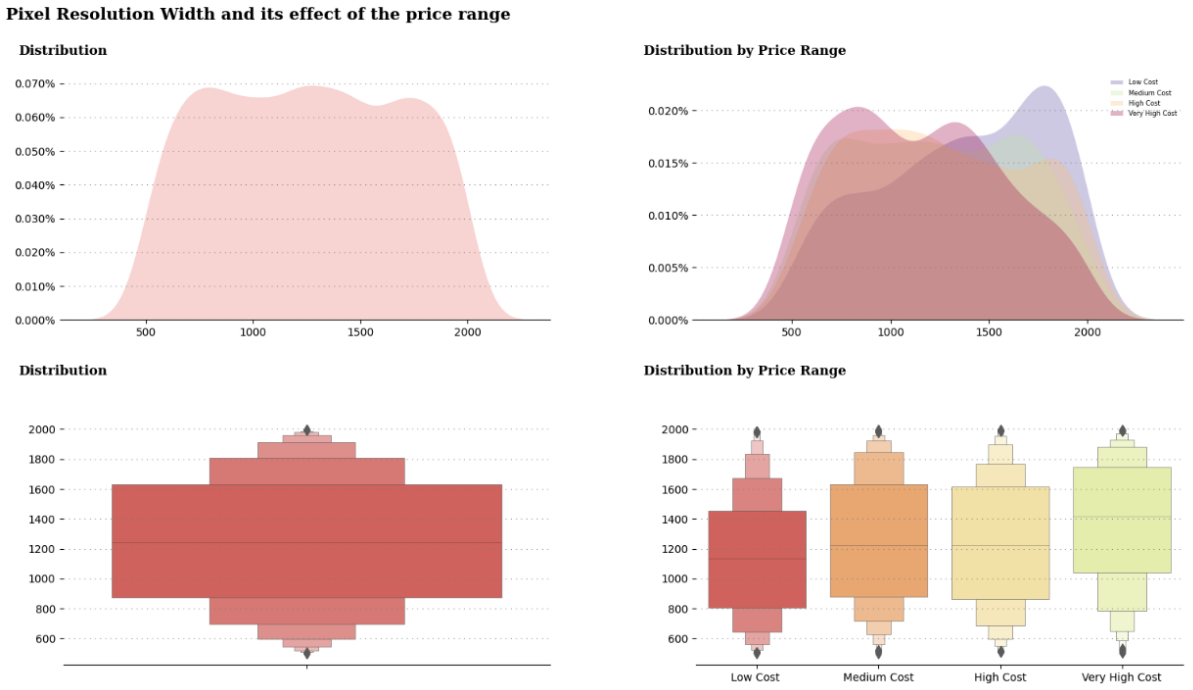


Рисунок 3.16 - Графіки щільності розподілу та оцінки плотності по характеристиці ширина роздільної здатності екрану

Розмір оперативної пам'яті (табл. 3.17, рис. 3.17).

Таблиця 3.17 - Результати аналізу розміру оперативної пам'яті

count	mean	std	min	25%	50%	75%	max
2000	2124.21300	1084.732044	256.00	1207.50	2146.500	3064.500	3998.00

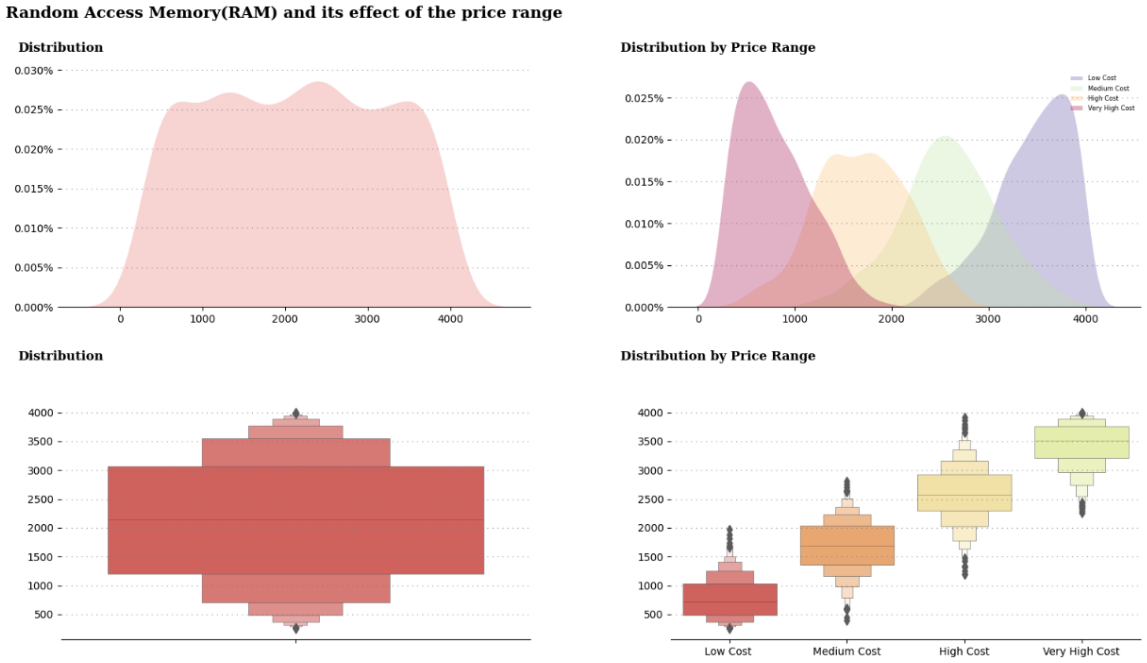


Рисунок 3.17 - Графіки щільності розподілу та оцінки плотності по характеристиці розміру оперативної пам'яті

Довжина екрану (табл. 3.18, рис. 3.18).

Таблиця 3.18 - Результати по характеристиці довжини екрану

count	mean	std	min	25%	50%	75%	max
2000	12.306500	4.213245	5.00	9.00	12.00	16.00	19.00

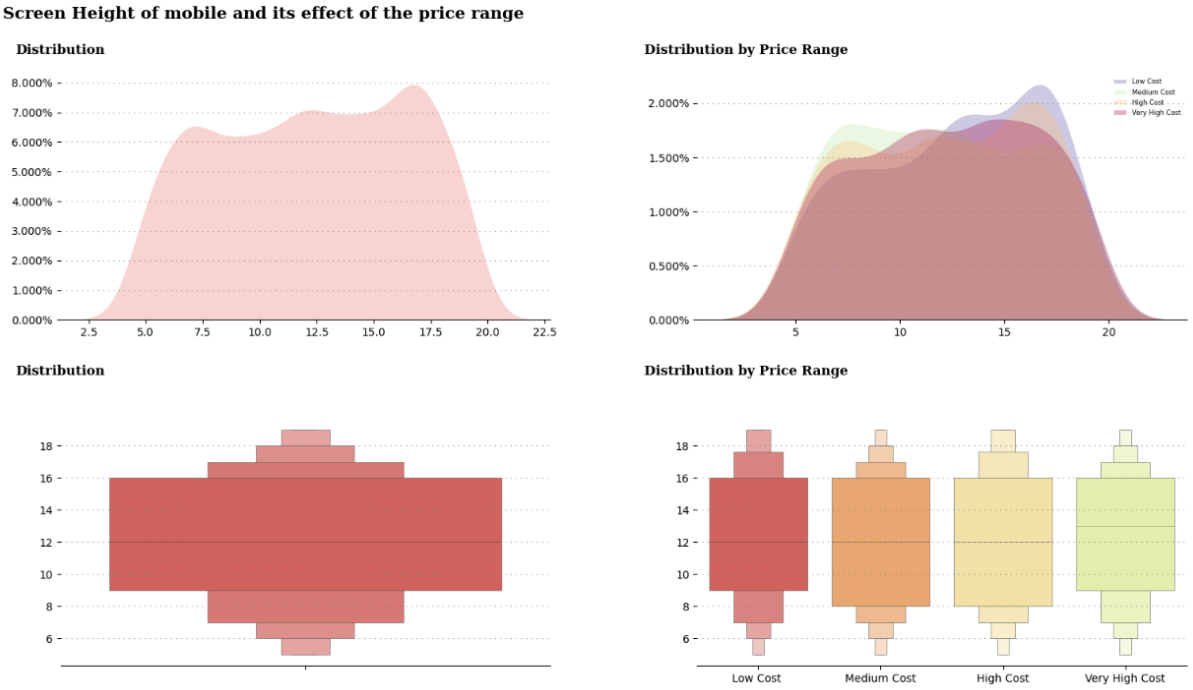


Рисунок 3.18 - Графіки щільності розподілу та оцінки плотності по характеристиці довжини екрану

Ширина екрану (табл. 3.19, рис. 3.19).

Таблиця 3.19 - Результати аналізу по характеристиці ширина екрану

count	mean	std	min	25%	50%	75%	max
2000	5.767000	4.356398	0.00	2.00	5.00	9.00	18.00

Рисунок 3.19 - Графіки щільності розподілу та оцінки плотності по характеристиці ширина екрану

Час на якій вистачає неперервної роботи телефону в стані дзвінку за один заряд аккумулятора. (табл. 3.20, рис. 3.20).

Таблиця 3.20 - Результати аналізу по характеристиці часу роботи

count	mean	std	min	25%	50%	75%	max
2000	11.01100	5.463955	2.00	6.00	11.00	16.00	20.00

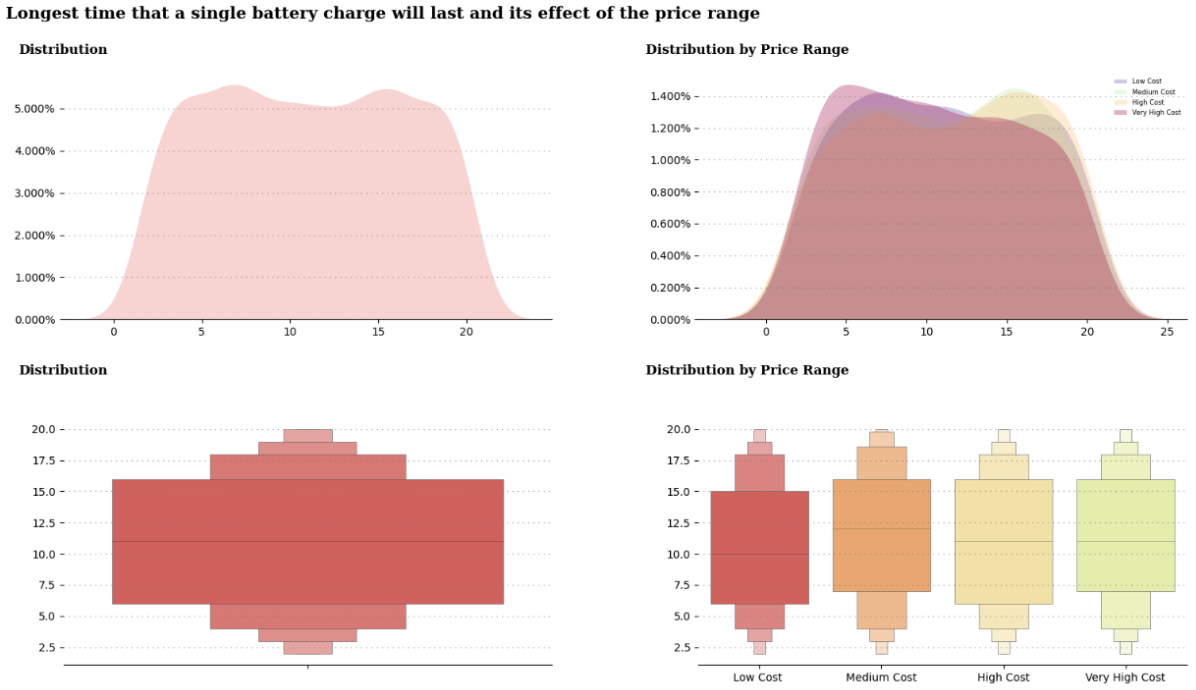


Рисунок 3.20 - Графіки щільності розподілу та оцінки плотності по характеристиці часу роботи

3.2 Результати роботи обраних алгоритмів

Зараз потрібно обрати модель яка буде найкраще класифікувати цей датасет. Для цього скористаємось всімома моделями які доступні нам в бібліотеці sklearn.

- Naive Bayes Classification Gaussian
- Naive Bayes classifier Multinomial
- Logistic Regression
- Random Forest Classifier
- Ada Boosting Classifier
- Support Vector Machine
- Decision Tree Classifier
- Nearest Neighbors
- Gradient Boosting Classifier

- Stochastic Gradient Descent
- Linear Discriminant Analysis
- Quadratic Discriminant Analysis

Результати роботи Naive Bayes Classification (табл. 3.21, рис. 3.21).

11	1	0	0
5	19	0	0
0	7	16	1
0	0	2	18

Рисунок 3.21 - Матриця похибок для результатів відпрацювання алгоритму Naive Bayes Classification Gaussian

Таблиця 3.22 - Метрики для оцінки роботи Naive Bayes Classification

	precision	recall	f1-score	support
0	0.69	0.92	0.79	12
1	0.70	0.79	0.75	24
2	0.89	0.67	0.76	24
3	0.95	0.90	0.92	20
accuracy			0.80	80
macro avg	0.81	0.82	0.80	80
weighted avg	0.82	0.80	0.80	80

Результати роботи Naive Bayes Multinomial (табл. 3.22, рис. 3.22).

12	8	0	0
4	10	0	0
0	7	11	4
0	2	7	15

Рисунок 3.22 - Матриця похибок для результатів відпрацювання алгоритму Naive Bayes Multinomial

Таблиця 3.22 - Метрики для оцінки Naive Bayes Multinomial

	precision	recall	f1-score	support
0	0.75	0.60	0.67	20
1	0.37	0.71	0.49	14
2	0.61	0.50	0.55	22
3	0.79	0.62	0.70	24
accuracy			0.60	80
macro avg	0.63	0.61	0.60	80
weighted avg	0.66	0.60	0.61	80

Результати роботи Logistic Regression (табл. 3.23, рис. 3.23).

14	0	0	0
2	27	0	0
0	7	17	0
0	0	1	19

Рисунок 3.23 - Матриця похибок для результатів відпрацювання алгоритму Logistic Regression

Таблиця 3.23 - Метрики для оцінки Logistic Regression

	precision	recall	f1-score	support
0	0.88	1.00	0.93	14
1	1.00	0.93	0.96	29
2	0.94	1.00	0.97	17
3	1.00	0.95	0.97	20
accuracy			0.96	80
macro avg	0.95	0.97	0.96	80
weighted avg	0.97	0.96	0.96	80

Результати роботи Random Forest Classifier (табл. 3.24, рис. 3.24).

14	1	0	0
2	20	0	0
0	6	18	1
0	0	0	18

Рисунок 3.24 - Матриця похибок для результатів відпрацювання алгоритму Random Forest Classifier

Таблиця 3.24 - Метрики для оцінки Random Forest Classifier

	precision	recall	f1-score	support
0	0.88	0.93	0.90	15
1	0.74	0.91	0.82	22
2	1.00	0.72	0.84	25
3	0.95	1.00	0.97	18
accuracy			0.88	80
macro avg	0.89	0.89	0.88	80
weighted avg	0.89	0.88	0.87	80

Результати роботи Ada Boosting Classifier (табл. 3.25, рис. 3.25).

12	3	0	0
4	22	0	0
0	2	16	10
0	0	1	9

Рисунок 3.25 - Матриця похибок для результатів відпрацювання алгоритму

Таблиця 3.25 - Метрики для оцінки роботи Ada Boosting Classifier

	precision	recall	f1-score	support
0	0.75	0.80	0.77	15
1	0.81	0.81	0.81	27
2	0.89	0.57	0.70	28
3	0.47	0.90	0.62	10
accuracy			0.74	80
macro avg	0.73	0.77	0.73	80
weighted avg	0.79	0.74	0.74	80

Результати роботи Support Vector Machine (табл. 3.26, рис. 3.26).

15	1	0	0
1	21	1	0
0	5	17	1
0	0	0	18

Рисунок 3.26 - Матриця похибок для результатів відпрацювання алгоритму Support Vector Machine

Таблиця 3.26 - Метрики для оцінки Support Vector Machine

	precision	recall	f1-score	support
0	0.94	0.94	0.94	16
1	0.78	0.91	0.84	23
2	0.94	0.74	0.83	23
3	0.95	1.00	0.97	18
accuracy			0.89	80
macro avg	0.90	0.90	0.89	80
weighted avg	0.90	0.89	0.89	80

Результати роботи Decision Tree Classifier (табл. 3.27, рис. 3.27).

12	0	0	0
4	24	1	0
0	3	15	1
0	0	2	18

Рисунок 3.27 - Матриця похибок для результатів відпрацювання алгоритму Decision Tree Classifier

Таблиця 3.27 - Метрики для оцінки Decision Tree Classifier

	precision	recall	f1-score	support
0	0.75	1.00	0.86	12
1	0.89	0.83	0.86	29
2	0.83	0.79	0.81	19
3	0.95	0.90	0.92	20
accuracy			0.86	80
macro avg	0.85	0.88	0.86	80
weighted avg	0.87	0.86	0.86	80

Результати роботи Nearest Neighbors (табл. 3.28, рис. 3.28).

10	6	2	0
5	14	10	1
1	7	3	7
0	0	3	11

Рисунок 3.28 - Матриця похибок для результатів відпрацювання алгоритму Nearest Neighbors

Таблиця 3.28 - Метрики для оцінки Nearest Neighbors

	precision	recall	f1-score	support
0	0.62	0.56	0.59	18
1	0.52	0.47	0.49	30
2	0.17	0.17	0.17	18
3	0.58	0.79	0.67	14
accuracy			0.48	80
macro avg	0.47	0.49	0.48	80
weighted avg	0.47	0.47	0.47	80

Результати роботи Gradient Boosting Classifier (табл. 3.29, рис. 3.29).

14	0	0	0
2	22	0	0
0	5	17	1
0	0	1	18

Рисунок 3.29 - Матриця похибок для результатів відпрацювання алгоритму

Таблиця 3.29 - Метрики для оцінки Gradient Boosting Classifier

	precision	recall	f1-score	support
0	0.88	1.00	0.93	14
1	0.81	0.92	0.86	24
2	0.94	0.74	0.83	23
3	0.95	0.95	0.95	19
accuracy			0.89	80
macro avg	0.90	0.90	0.89	80
weighted avg	0.89	0.89	0.89	80

Результати роботи Stochastic Gradient Descent (табл. 3.30, рис. 3.30).

16	0	0	0
0	10	6	0
0	17	11	0
0	0	1	19

Рисунок 3.30 - Матриця похибок для результатів відпрацювання алгоритму Stochastic Gradient Descent

Таблиця 3.30 - Метрики для оцінки роботи Stochastic Gradient Descent

	precision	recall	f1-score	support
0	1.00	1.00	1.00	16
1	0.37	0.62	0.47	16
2	0.61	0.39	0.48	28
3	1.00	0.95	0.97	20
accuracy			0.70	80
macro avg	0.75	0.74	0.73	80
weighted avg	0.74	0.70	0.70	80

Результати роботи Linear Discriminant Analysis (табл. 3.31, рис. 3.31).

14	0	0	0
2	26	0	0
0	1	18	0
0	0	0	19

Рисунок 3.31 - Матриця похибок для результатів відпрацювання алгоритму

Таблиця 3.31 - Метрики для оцінки Linear Discriminant Analysis

	precision	recall	f1-score	support
0	0.88	1.00	0.93	14
1	0.96	0.93	0.95	28
2	1.00	0.95	0.97	19
3	1.00	1.00	1.00	19
accuracy			0.96	80
macro avg	0.96	0.97	0.96	80
weighted avg	0.97	0.96	0.96	80

Результати роботи Quadratic Discriminant Analysis (табл. 3.32, рис. 3.32).

15	0	0	0
1	25	0	0
0	2	18	0
0	0	0	19

Рисунок 3.32 - Матриця похибок для результатів відпрацювання алгоритму Quadratic Discriminant Analysis

Таблиця 3.32 - Метрики для оцінки Quadratic Discriminant Analysis

	precision	recall	f1-score	support
0	0.94	1.00	0.97	15
1	0.93	0.96	0.94	26
2	1.00	0.90	0.95	19
3	1.00	1.00	1.00	19
accuracy			0.96	80
macro avg	0.96	0.97	0.96	80
weighted avg	0.97	0.96	0.96	80

3.3 Висновки

У роботі було розглянуті всі можливі варіанти рішення задачі класифікації доступні в бібліотеці Scikit-learn, яка я хочу помітити являється найкращою серед бібліотек машинного навчання. Кожна модель мала свої недоліки чи переваги, але оскільки наше дослідження було більше для саморозвитку, а не вирішувало чітко сформульовано прикладну задачу вибір кращої моделі перетворився на вибір в залежності від умов, які якраз би з'явилися при прикладній задачі. Деякі моделі виявились нестійкими до перенавчання або великої вибірки. Також був проведена попередню перевірка яка в нашому випадку зіграла велику роль.

4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик програмного продукту, розробленого для вирішення задач класифікації цінової категорії мобільних девайсів.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. ФВА проводиться з метою виявлення резервів зниження витрат за рахунок ефективніших варіантів виробництва, кращого співвідношення між споживчою вартістю виробу та витратами на його виготовлення. Для проведення аналізу використовується економічна, технічна та конструкторська інформація.

Алгоритм функціонально-вартісного аналізу включає в себе визначення послідовності етапів розробки продукту, визначення повних витрат (річних) та кількості робочих часів, визначення джерел витрат та кінцевий розрахунок вартості програмного продукту.

4.1 Постановка задачі проектування

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи класифікації цінової категорії для мобільних телефонів. Оскільки рішення стосовно проектування та реалізації компонентів, що розробляється, впливають на всю систему, кожна окрема підсистема має її задовольняти. Тому

фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу даних по коронавірусу.

Технічні вимоги до програмного продукту є наступні:

- функціонування на персональних комп'ютерах із стандартним набором компонентів;
- зручність та зрозумілість для користувача;
- швидкість обробки даних та доступ до інформації в реальному часі;
- можливість зручного масштабування та обслуговування;
- мінімальні витрати на впровадження програмного продукту.

4.2 Обґрунтування функцій та параметрів програмного продукту

Головна функція F_0 – розробка програмного продукту, який вирішує задачу класифікації цінової категорії для мобільного телефону. Беручи за основу цю функцію, можна виділити наступні:

F_1 – програмна реалізація

F_2 – спосіб введення даних

F_3 – обробка вхідних даних

F_4 - вивід графіків

Кожна з цих функцій має декілька варіантів реалізації:

Функція F_1:

а) Python

б) C++

в) R

Функція F_2:

а) зчитування з текстового файлу

б) зчитування з файлу .csv

Функція F_3:

а) власна реалізація існуючих методів

Функція F_4:

а) на одній вкладці всі результати

б) результати різних методів на різних вкладках

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1).

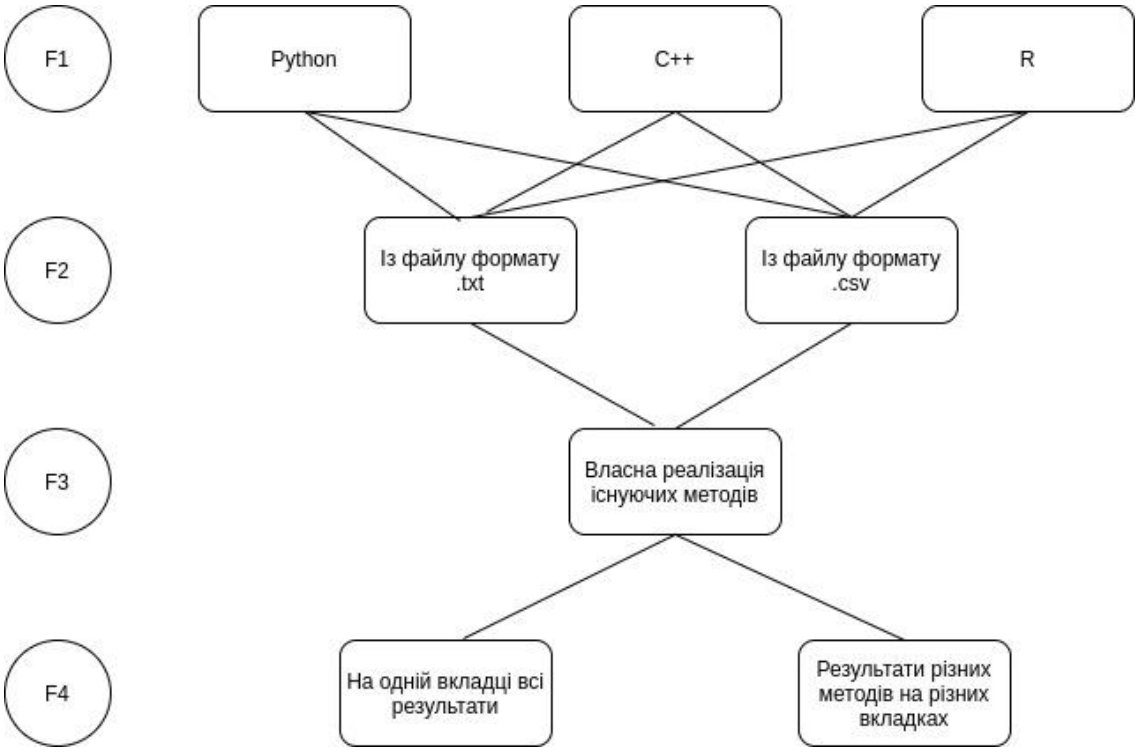


Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіанти основних функцій (табл. 4.1).

Таблиця 4.1 - Позитивно-негативна карта

Основні функції	Варіанти реалізації	Переваги	Недоліки
F1	а	Швидке виконання модулів на які мова поділена	Потрібна чітка структуризація ПО
	б	Швидкодія	Відносно складна структура коду, потрібно багато часу на розробку
	в	Швидко виконується код	Мала кількість бібліотек та реалізованих функцій
F2	а	Використовується при великому об'ємі інформації	Складно використовувати
	б	Широко поширений	Складний в обробці
F3	а	Можливість послідовного обчислення	Необхідність написання кожного методу
F4	а	Можливість зробити аналіз не переключаючи сторінки	Маленький масштаб
	б	Великий масштаб	Потрібно прикручувати сторінки

На основі цієї карти будуємо позитивно-негативну матрицю варіантів основних функцій (Таб.4.2). Робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути,

тому що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F_1:

Оскільки час на розробку програмного продукту є обмеженими серед необхідних досліджень є й інші задачі, то варіант б) відкидаємо, також необхідно відкинути варіант в) через низьку швидкодію.

Функція F_2:

Так, як наведені варіанти реалізації мають свої значні переваги один над одним, варто розглянути та проаналізувати обидва.

Обидва варіанти можна використовувати в розробці.

Функція F_4:

Оскільки програма не призначена для одночасної роботи з багатьма методами одночасно, то варіант а) можна відкинути .

Таким чином будемо використовувати наступні варіанти реалізації програмного продукту:

1. F1а – F2а – F3а – F4а;

2. F1а – F2б – F3а – F4б;

3. F1а – F2б – F3а – F4а;

4. F1а – F2а – F3а – F4б.

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче (таблиця 4.2.) Будемо розглядати три типи варіантів значення параметрів. Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію.

Таблиця 4.2 - Основні параметри програмного продукту

Назва параметру	Умовні позначення	Одиниці виміру	Значення параметру		
			гірші	середні	кращі
Об'єм пам'яті для збереження даних	X1	Мб	32	16	8
Час обробки даних	X2	Мб	800	420	60
Точність розв'язку	X3	%	10	5	1
Потенційний об'єм програмного продукту	X4	кількість комірок коду	1600	1200	800

Графічна характеристика параметру X1 наведена на рисунку 4.2:

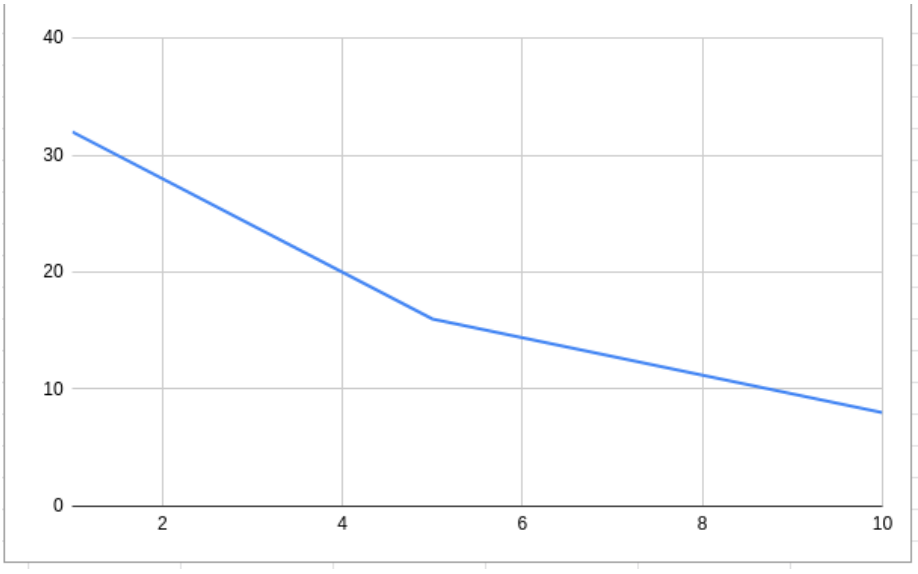


Рисунок 4.2 - X1, об’єм пам’яті для збереження даних

Графічна характеристика параметру X2 наведена на рисунку 4.3:

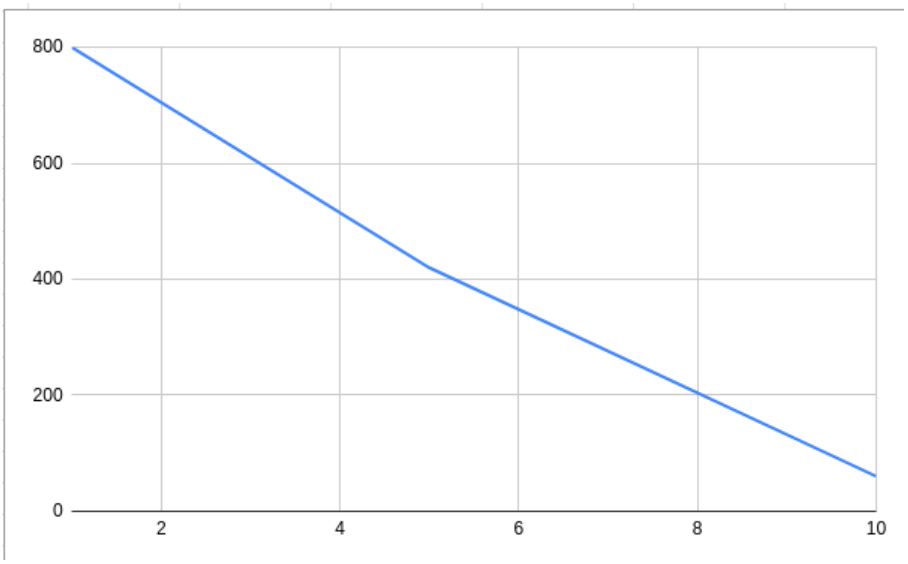


Рисунок 4.3 - X2, час обробки даних

Графічна характеристика параметру X3 наведена на рисунку 4.4:

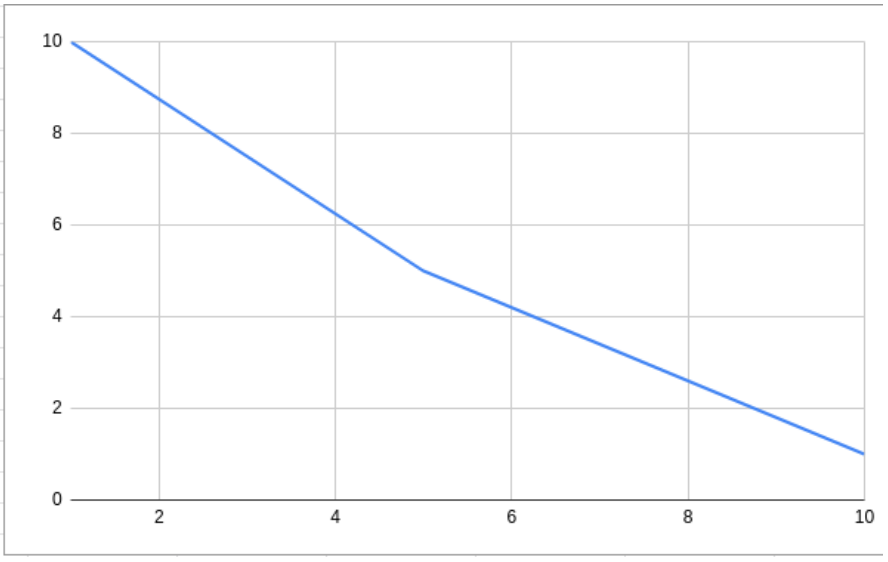


Рисунок 4.4 - X3, точність розв’язку

Графічна характеристика параметру X4 наведена на рисунку 4.5:

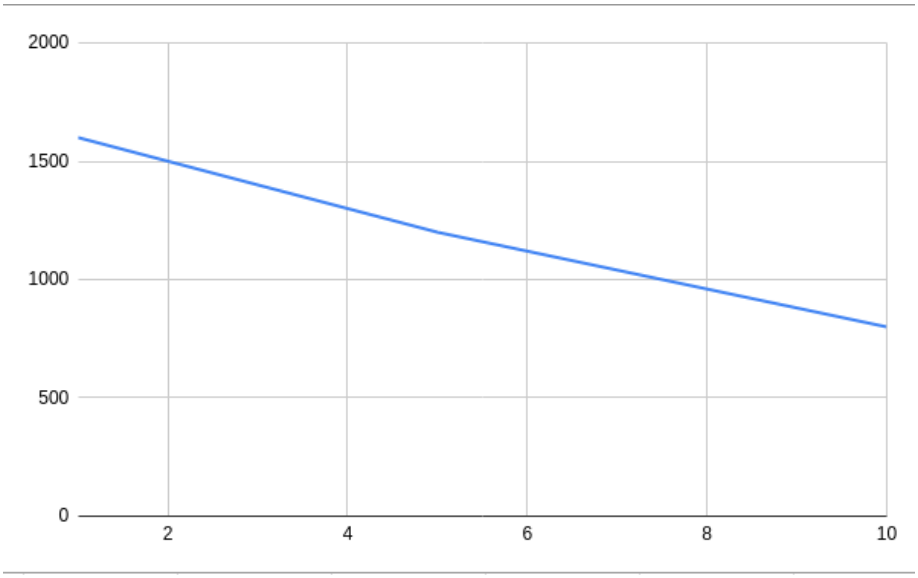


Рисунок 4.5 - X4, потенційний об’єм програмного коду

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 - Результати ранжування параметрів

Позначення параметру	Ранг параметру за оцінкою експерта							Сума рангів	Відхилення	Δ_i^2
	1	2	3	4	5	6	7			
X1	2	2	1	2	2	2	1	12	-5,5	30,25
X2	3	3	4	3	4	3	3	23	5,5	30,25
X3	1	1	2	1	1	1	2	9	-8,5	72,25
X4	4	4	3	4	3	4	4	26	8,5	72,25
Разом	10	10	10	10	10	10	10	70	0	205

Порахуємо коефіцієнт узгодженості:

$$W=\frac{12S}{N^2(n^3-n)}=\frac{12\cdot 205}{7^2(4^3-4)}=0.84>W_k=0,67.$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця 4.4 - Попарне порівняння показників

Параметри	Експерти							Підсумкова оцінка	Числове значення
	1	2	3	4	5	6	7		
X1, X2	<	<	<	<	<	<	<	<	0,5
X1, X3	>	>	<	>	>	>	<	>	1,5
X1, X4	<	<	<	<	<	<	<	<	0,5
X2, X3	>	>	>	>	>	>	>	>	1,5
X2, X4	<	<	>	<	>	<	<	<	0,5
X3, X4	<	<	>	<	<	>	<	<	0,5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij}визначається по формулі:

$$a_{ij}=\begin{cases}1.5 \text{ при } X_i > X_j\\ 1.0 \text{ при } X_i = X_j.\\ 0.5 \text{ при } X_i < X_j\end{cases}$$

У таблиці 4.5. представлені результати розрахунку вагомості.

Таблиця 4.5 - Розрахунок вагомості параметрів

Параметри	Параметри				Перша ітер.		Друга ітер.		Третя ітер.	
	X1	X2	X3	X4	b_i	$К_{вi}$	b_i^1	$\hat{E}^1_{\hat{a}^3}$	b_i^2	$\hat{E}^2_{\hat{a}^3}$
X1	1,0	0,5	1,5	0,5	3,5	0,219	12,25	0,208	44,875	0,207
X2	1,5	1,0	1,5	0,5	4,5	0,281	16,25	0,275	59,125	0,273
X3	0,5	0,5	1,0	0,5	2,5	0,156	9,25	0,157	34,125	0,160
X4	1,5	1,5	1,5	1,0	5,5	0,344	21,25	0,360	77,875	0,360
Параметри	Параметри				Перша ітер.		Друга ітер.		Третя ітер.	
Всього					16	1	59	1	216	1

Як видно з таблиці, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

4.3 Аналіз рівня якості варіантів реалізації функцій

Варіант функції F1 описується параметром X1, F2 параметром X2, F3 параметром X3, F4 параметром X4.

Таблиця 4.6 - Розрахунок показників рівня якості варіантів реалізації

Основні функції	Варіант реалізації функції	Абсолютне значення параметру	Бальна оцінка параметру	Коефіцієнт вагомості параметру	Коефіцієнт рівня якості
F1	А	16	5,8	0,207	1,2006
F2	А	520	4	0,273	1,092
	Б	200	8	0,273	2,184
F3	А	7	3	0,160	0,48
F4	А	1000	7,5	0,360	2,7
	Б	900	9,7	0,360	3,492

Враховуючи дані з порівнянь варіантів реалізацій функцій можна виключити з реалізацій функцій наступні варіанти: F1(б, в).

Залишаються наступні варіанти:

1.F1(а)=> F2(а=)> F3(а)=>F4(а)

2.F1(а)=>F2(б)=>F3(а)=>F4(б)

3.F1(а)=>F2(б)=>F3(а)=>F4(а)

4.F1(а)=>F2(а)=>F3(а)=>F4(б)

За даними з таблиці 4.6 за формулою:

$$K_K = K_{ТУ}[F_{1k}] + K_{ТУ}[F_{2k}] + \ldots + K_{ТУ}[F_{zk}],$$

визначаємо рівень якості кожного з варіантів:

КК1 = 1,2006+1,092+0,48+2,7=5,4726

КК2 = 1,2006+2,184+0,48+3,492=7,3566

КК3=1,2006+2,184+0,48+2,7=6,5626

КК4= 1,2006+1,092+0,48+3,492=6,2646

Як видно з розрахунків, кращим є другий варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.4 Економічний аналіз варіантів розробки ПП

Для оцінки трудомісткості розробки спочатку проведемо розрахунок трудомісткості. Усі варіанти мають наступні основні завдання:

- програмна реалізація за допомогою мови Python
- графічне представлення результатів за допомогою програмних засобів.

Також кожний з варіантів має два додаткових завдання, які є реалізаціями розгалужених варіантів розробки незалежного модуля. Далі наведено варіанти додаткових завдань(два завдання, які мають номери 3 в реалізаціях та два завдання, які мають номери 4 в реалізаціях).Також кожний з варіантів має два додаткових завдання, які є реалізаціями розгалужених варіантів розробки незалежного модуля. Далі наведено варіанти додаткових завдань(два завдання, які мають номери 3 в реалізаціях та два завдання, які мають номери 4 в реалізаціях).

3.1) завантаження даних із формату .txt;

3.2) завантаження даних із формату.csv;

4.1) оцінювання результату за допомогою сторонніх модулів;

4.2) оцінювання результату за допомогою власних функцій.

В варіанті 1 присутні наступні додаткові завдання під номерами 3.1 та 4.1

В варіанті 2 присутні наступні додаткові завдання під номерами 3.2 та 4.1

В варіанті 3 присутні наступні додаткові завдання під номерами 3.1 та 4.2

В варіанті 4 присутні наступні додаткові завдання під номерами 3.2 та 4.2

За ступенем новизни до групи А відноситься задання 1, 4.2, до групи Б – 4.1, до групи В відносяться завдання 3.2,3.1 , до групи Г — 2.

За складністю алгоритмів до групи 1 відносяться завдання 1, 4.2 до групи 2 відноситься завдання 3.2, 4.1, до групи 3 – 3.1.

Якщо при розробці ПП використовуються стандартні бібліотеки та(чи) пакети прикладних програм, норми часу коректуються з допомогою коефіцієнта Кст,значення якого лежать в діапазоні 0,6-0,8. У нашому випадку Кст =0,7.

Для першого завдання (степінь новизни А, група складності – 1), тобто ТР=90 людино-днів, КП = 1,7, КСК = 1; КСТ = 0.7.

Тоді, загальна трудомісткість програмування першого завдання дорівнює:

$T1 = 90 \cdot 1,7 \cdot 1 \cdot 0,7 = 107,1$ людино-днів

Для другого завдання (степінь новизни Г, група складності – 2), тобто ТР= 12 людино-днів,КП = 0,72; КСК = 1; КСТ = 0.7:

$T2 = 12 \cdot 0,72 \cdot 1 \cdot 0,7 = 6,1$ людино-днів.

Для завдання 3.1 (степінь новизни В, група складності – 3), тобто ТР=12 людино-днів, КП = 0,6 ; КСК = 1; КСТ = 0.7:

$T3.1 = 12 \cdot 0,6 \cdot 1 \cdot 0.7 = 5,04$ людино-днів.

Для завдання 3.2(степінь новизни В, складність алгоритмів 2) , тобто ТР= 19 людино-днів, КП =0,72 ; КСК = 1; КСТ = 0.7

$T3.2 = 19 \cdot 0,72 \cdot 1 \cdot 0.7 = 9,58$ людино-днів.

Для завдання 4.1 (степінь новизни Б, група складності – 2), тобто ТР=27 людино-днів,КП = 1.08; КСК = 1; КСТ = 0.7:

$T4.1 = 27 \cdot 1,08 \cdot 1 \cdot 0.7 = 20,41$ людино-днів.

Для завдання 4.2(степінь новизни А, група складності — 1), тобто ТР=90 людино-днів, КП = 1,7 , КСК = 1; КСТ = 0.7.

$T4.2 = 90 \cdot 2,84 \cdot 1 \cdot 0.7 = 178,92$ людино-днів

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

ТІ = (107,1 + 6,1 + 5,04+20,41) · 8 = 1109,2 людино-годин;

ТІІ = (107,1 + 6,1 + 9,58 + 20, 41) · 8 = 1145,5 людино-годин.

ТІІІ = (107,1 + 6,1 + 5,04 + 178,92) · 8 = 2377,28 людино-годин.

ТІV = (107,1 + 6,1 + 9,58 + 178,92) · 8 = 2413, 6 людино-годин.

Найбільш високу трудомісткість має варіант IV.

В розробці беруть участь два програмісти з окладом 10250 грн., один фінансовий аналітик з окладом 12900грн. Визначимо зарплату за годину:

$$СЧ = \frac{10250+10250+12900}{3 \cdot 21 \cdot 8} = 66,27 \text{ грн}$$

Тоді, заробітну плата розробників за варіантами становить:

І. СЗП = 66,27 · 1109,2 · 1.2 = 88208,02грн.

ІІ. СЗП = 66,27 · 1145,5 · 1.2 =91070,88грн.

ІІІ. СЗП = 66,27 · 2377,28 · 1.2 =189050,81 грн.

ІV. СЗП = 66,27 · 2413,6 · 1.2 = 191939,12грн.

Відрахування на єдиний соціальний внесок становить 22%:

І. СВІД = СЗП · 0.22 = 88208,02 · 0.22 =19405,76 грн.

ІІ. СВІД = СЗП · 0.22 = 91070,88 · 0.22 = 20035,59грн.

ІІІ. СВІД = СЗП · 0.22 = 189050 · 0.22 = 41591грн.

ІV. СВІД = СЗП · 0.22 = 191939,12 · 0.22 =42226,6 грн.

Тепер, визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 10250 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 10250 \cdot 0,2 = 24600$грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г \cdot (1 + К_З) = 24600 \cdot (1 + 0.2) = 29520$ грн.

Відрахування на єдиний соціальний внесок:

$С_{ВІД}= С_{ЗП} \cdot 0.22 = 29520 \cdot 0,22 = 6492,4$ грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 15250 грн.

$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 15250 = 4384,38$ грн.,

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 15250 \cdot 0.05 = 876,88 \text{ грн.},$$

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$Т_{ЕФ} =(Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 142 – 16) \cdot 8 \cdot 0.9 = 1324,8$ год.,

Витрати на оплату електроенергії розраховуємо за формулою:

$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} =1324,8 \cdot 0,3 \cdot 0,6 \cdot 3,51 = 837,01$ грн.,

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 15250 \cdot 0,67 =10217,5 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} =С_{ЗП}+ С_{ВІД}+ С_А + С_Р+ С_{ЕЛ} + С_Н$$

$С_{ЕКС} = 29520 + 6492,4+ 4384,38+ 876,88+837,01+ 10217,5 = 52328,17$грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$С_{М-Г} = С_{ЕКС}/ Т_{ЕФ} = 52328,17/1324,8 = 39,50$ грн/час.

Отже, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$С_М = С_{М-Г} \cdot Т$$

I.$С_М = 39,50 \cdot 1109,2 = 43813,4$ грн.

II. $С_М$ = 39,50 ·1145,5 = 45247,25 грн.

III. $С_М$ =39,50 ·2377,28 = 93902,56 грн

IV. $С_М$ = 39,50 ·2413, 6 = 95337,2 грн.

Накладні витрати складають 67% від заробітної плати:

СН = СЗП · 0,67

I. $С_Н$ = 88208,02· 0,67 =59099,37 грн.

II. $С_Н$ =91070,88 · 0,67 = 61017,48грн.

III.$С_Н$ =189050,81· 0,67 =126664,04 грн.

IV. $С_Н$ = 191939,12· 0,67 = 128599,21грн.

Отже, вартість розробки ПП за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н$$

I. $С_{ПП}$ = 88208,02 + 19405,76 + 43813,4 + 59099,37 = 210526,55 грн.;

II. $С_{ПП}$ = 91070,88 + 20035,59 + 45247,25 + 61017,48 = 217371,2 грн.

III. $С_{ПП}$ = 189050,81 + 41591+93902,56 + 126664,04 = 451208,41 грн.

IV.$С_{ПП}$ = 191939,12 + 42226,6 + 95337,2 +128599,21 = 458102,13 грн.

4.5 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕРj} = К_{Кj} / С_{ППj},$$

$К_{ТЕР1}$ = 5,4726 / 210526,55 =2,59 $\cdot 10^{-5}$

$К_{ТЕР2}$ = 7,3566 /217371,2 = 3,38$\cdot 10^{-5}$

$К_{ТЕР3}$= 6,5626/ 451208,41 = 1,45 $\cdot 10^{-5}$

$К_{ТЕР4}$ =6,2646 / 458102,13 = 1,36$\cdot 10^{-5}$

Як бачимо, найбільш ефективним є другий варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР2}$ =3,38·10^{-5}

4.6 Висновки

Після виконання функціонально-вартісного аналізу програмного комплексу, що розробляється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є другий варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості:

$К_{ТЕР2}$ = 7,3566 / 217371,2 =3,38·10^{-5}

До другого варіанту реалізації відносяться такі задачі:програмна реалізація за допомогою мови Python;завантаження даних із формату .csv;оцінювання результату за допомогою сторонніх модулів;результати різних методів представлені на різних вкладках.

ВИСНОВКИ

На етах написання дипломної роботи було розглянуто такі аспекти як актуальність обраної теми та позиція машинного навчання в такій ніші як прогнозування цін.

Для повного аналізу та рішення обраної задачі були виконанні наступні етапи.

1. Постановка та аналіз актуальності даної задачі, пошук літератури та досліджень в даному напрямі.
2. Розгляд всіх доступних варіантів вирішення даної задачі.
3. Використано всі доступні метрики для перевірки даних вибірки на цілісність та правдивість.
4. Використання алгоритмів класифікації та порівняння їх між собою за певними метриками та постановками задачами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Naive Bayes Classification using Scikit-learnю. URL: https://www.datacamp.com/community/tutorials/naive-bayes-scikit-learn?utm_source=adwords_ppc&utm_campaignid=1455363063&utm_adgroupid=65083631748&utm_device=c&utm_keyword=&utm_matchtype=b&utm_network=g&utm_adpostion=&utm_creative=278443377095&utm_targetid=aud-299261629574:dsa-429603003980&utm_loc_interest_ms=&utm_loc_physical_ms=1030367&gclid=CjwKCAjwhMmEBhBwEiwAXwFoEcnfYgLaqPYX5d29qAk9xqxJ79iQtLQTJGQgbAZYHyUTPhLh2M01bxoCfpsQAvD_BwE
2. Logistic Regression. URL: https://christophm.github.io/interpretable-ml-book/logistic.html#advantages-and-disadvantages
3. Multinomial Naive Bayes Explained. URL: https://www.mygreatlearning.com/blog/multinomial-naive-bayes-explained/
4. Naive Bayes Classification using Scikit-learn. URL: https://www.datacamp.com/community/tutorials/naive-bayes-scikit-learn?utm_source=adwords_ppc&utm_campaignid=1455363063&utm_adgroupid=65083631748&utm_device=c&utm_keyword=&utm_matchtype=b&utm_network=g&utm_adpostion=&utm_creative=278443377095&utm_targetid=aud-299261629574:dsa-429603003980&utm_loc_interest_ms=&utm_loc_physical_ms=9061018&gclid=CjwKCAjw7diEBhB-EiwAskVi1

8kAm0ffoowjdlBTp-dnZCi_Q9VupAdPfqKagEhLJQxXqsz7hEj2axoCa-0QAvD_BwE

5. A complete guide to the random forest algorithm. URL: https://builtin.com/data-science/random-forest-algorithm
6. Boosting Algorithms Explained. URL: https://towardsdatascience.com/boosting-algorithms-explained-d38f56ef3f30
7. Support Vector Machine — Introduction to Machine Learning Algorithms. URL: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47
8. Decision Tree Classification. URL: https://medium.com/swlh/decision-tree-classification-de64fc4d5aac
9. Machine Learning Basics with the K-Nearest Neighbors Algorithm. URL: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761
10. Using Stochastic Gradient Descent to Train Linear Classifiers. URL: https://towardsdatascience.com/using-stochastic-gradient-descent-to-train-linear-classifiers-c80f6aeaff76
11. Дискриминантный линейный анализ (Linear discriminant analysis). URL: https://wiki.loginom.ru/articles/linear-discriminant-analysis.html
12. Три версии дискриминантного анализа: различия и как их использовать. URL: https://qastack.ru/stats/71489/three-versions-of-discriminant-analysis-differences-and-how-to-use-them
13. Жизненный цикл модели машинного обучения. URL: http://neerc.ifmo.ru/wiki/index.php?title=%D0%96

14. Використання алгоритмів машинного навчання для прогнозування тенденцій ціноутворення. URL: https://www.onetech.ai/en/blog/using-machine-learning-algorithms-to-predict-pricing-trends#:~:text=With%20Machine%20Learning%20(ML)%20technology,multiple%20independent%20(interdependent)%20variables.
15. Price Prediction using Machine Learning Regression — a case study https://towardsdatascience.com/mercari-price-suggestion-97ff15840dbd
16. Gradient Boosting Trees for Classification: A Beginner's Guide. URL: https://medium.com/swlh/gradient-boosting-trees-for-classification-a-beginners-guide-596b594a14ea
17. Price Forecasting: Applying Machine Learning Approaches to Electricity, Flights, Hotels, Real Estate, and Stock Pricing. URL: https://www.altexsoft.com/blog/business/price-forecasting-machine-learning-based-approaches-applied-to-electricity-flights-hotels-real-estate-and-stock-pricing/

ДОДАТОК А ЛІСТИНГ ПРОГРАМИ

- main.py

```
import numpy as np
import pandas as pd
import pandas
import seaborn as sns
import matplotlib.pyplot as plt
from collections import Counter
import seaborn as sns; sns.set()

from sklearn import tree
import graphviz
import os
import preprocessing

import numpy as np
import pandas as pd
from plotly.offline import init_notebook_mode, iplot, plot
import plotly as py
import plotly.graph_objs as go
from wordcloud import WordCloud
import matplotlib.pyplot as plt

from pandas_profiling import ProfileReport

from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2, f_classif
from sklearn.model_selection import KFold
from sklearn.feature_selection import SelectFromModel
from sklearn.svm import LinearSVC

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict

from sklearn.preprocessing import normalize
from sklearn.preprocessing import StandardScaler, LabelEncoder,
OneHotEncoder
from sklearn.model_selection import train_test_split

from sklearn.pipeline import Pipeline, make_pipeline
from sklearn.decomposition import PCA
```

```
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import CategoricalNB
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.cluster import KMeans
from sklearn.ensemble import RandomForestClassifier,
AdaBoostClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.preprocessing import StandardScaler
from xgboost import XGBClassifier

from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.metrics import precision_score, recall_score,
f1_score
from sklearn.metrics import classification_report
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.model_selection import GridSearchCV

from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import SGDClassifier, LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn import metrics
from sklearn.neural_network import MLPClassifier
from xgboost import XGBClassifier, XGBRFClassifier
from xgboost import plot_tree, plot_importance

from sklearn.metrics import confusion_matrix, accuracy_score,
roc_auc_score, roc_curve
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import RFE

from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import
LinearDiscriminantAnalysis, QuadraticDiscriminantAnalysis


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

```
import os

for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))


train =
pd.read_csv('/home/artur/PycharmProject/deplom_str/data/train.csv')
test =
pd.read_csv('/home/artur/PycharmProject/deplom_str/data/test.csv')
print(train.describe())

columns = list(train.columns)
columns.remove('price_range')

features = columns
label = ['price_range']

X = train[features]
y = train[label]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=101)
X_valid, X_test, y_valid, y_test = train_test_split(X_test,
y_test, test_size=0.2, random_state=42)

print(f'Total # of sample in whole dataset: {len(X)}')
print(f'Total # of sample in train dataset: {len(X_train)}')
print(f'Total # of sample in validation dataset: {len(X_valid)}')
print(f'Total # of sample in test dataset: {len(X_test)}')

sc=StandardScaler()

X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

cv_results_acc = []
models = {
    'GaussianNB': GaussianNB(),
    'BernoulliNB': BernoulliNB(),
    'LogisticRegression': LogisticRegression(),
    'RandomForestClassifier': RandomForestClassifier(),
    'Ada Boosting' : AdaBoostClassifier(),
    'SupportVectorMachine': SVC(),
    'DecisionTreeClassifier': DecisionTreeClassifier(),
    'KNeighborsClassifier': KNeighborsClassifier(),
    'GradientBoostingClassifier': GradientBoostingClassifier(),
```

```
    'Stochastic Gradient Descent': SGDClassifier(max_iter=5000,
random_state=0),
    'Linear Discriminant' : LinearDiscriminantAnalysis(),
    'Quadratic Discriminant' : QuadraticDiscriminantAnalysis(),
}

modelNames = ['GaussianNB', 'BernoulliNB', 'LogisticRegression',
'RandomForestClassifier', 'Ada Boosting Classifier',
'SupportVectorMachine',
              'DecisionTreeClassifier', 'KNeighborsClassifier',
'GradientBoostingClassifier',
              'Stochastic Gradient Descent', 'Linear Discriminant',
'Quadratic Discriminant']

trainScores = []
validationScores = []
testScores = []

for m in models:
    model = models[m]
    model.fit(X_train, y_train)
    score = model.score(X_valid, y_valid)
    # print(f'{m} validation score => {score*100}')

    print(f'{m}')
    train_score = model.score(X_train, y_train)
    print(f'Train score of trained model: {train_score * 100}')
    trainScores.append(train_score * 100)

    validation_score = model.score(X_valid, y_valid)
    print(f'Validation score of trained model: {validation_score *
100}')
    validationScores.append(validation_score * 100)

    test_score = model.score(X_test, y_test)
    print(f'Test score of trained model: {test_score * 100}')
    testScores.append(test_score * 100)
    print(" ")

    y_predictions = model.predict(X_test)
    conf_matrix = confusion_matrix(y_predictions, y_test)

    print(f'Confussion Matrix: \n{conf_matrix}\n')

    predictions = model.predict(X_test)
    cm = confusion_matrix(predictions, y_test)

    print("")
```

```
    print(f'Classification Report:
\n{classification_report(predictions, y_test)}\n')
    print("")

    for m in range(1):
        current = modelNames[m]
        modelNames.remove(modelNames[m])
        cv_score = cross_val_score(model, X_train, y_train,
scoring="accuracy", cv=10)
        cv_results_acc.append(cv_score.mean() * 100)
        print("Cross Validation Accuracy: %s: %f " % (current,
cv_score.mean()))

    preds = model.predict(X_test)
    confusion_matr = confusion_matrix(y_test, preds)   # normalize =
'true'

print("##################################################################
###############")
    print("")
    print("")
    print("")
```

- 3g.py

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.ticker as mtick

pd.set_option('display.max_columns', None)



import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

train = pd.read_csv('../../data/train.csv')

fig = plt.figure(figsize=(15,4))
gs = fig.add_gridspec(1, 2)
gs.update(wspace=0.3, hspace=0)
fig.text(0.120,1.1,'3G and its effect on the price ',
fontfamily='serif',fontsize=15, fontweight='bold')
ax0 = fig.add_subplot(gs[0, 0])
ax1 = fig.add_subplot(gs[0, 1])
```

```
sns.countplot(x='three_g',
          data=train,
           palette='bone',
          ax=ax0)
ax0.grid(color='gray', linestyle=':', axis='y', zorder=0,
dashes=(1,5))
ax0.set_title('Distribution',fontsize=12,fontfamily='serif',fontwei
ght='bold',x=0.1,y=1)
ax0.spines['top'].set_visible(False)
ax0.spines['right'].set_visible(False)
ax0.spines['left'].set_visible(False)
ax0.set_xticklabels(["Doesnt Support","Support"])
ax0.set_xlabel("")
ax0.set_ylabel("")

sns.countplot(x='three_g',
            data=train,
            hue='price_range',
            ax=ax1)
ax1.grid(color='gray', linestyle=':', axis='y', zorder=0,
dashes=(1,5))
ax1.set_title('Distribution by Price
Range',fontsize=12,fontfamily='serif',fontweight='bold',x=0.25,y=1)
ax1.spines['top'].set_visible(False)
ax1.spines['right'].set_visible(False)
ax1.spines['left'].set_visible(False)
ax1.get_legend().remove()
legend_labels, _= ax1.get_legend_handles_labels()
ax1.legend(legend_labels, ['Low Cost ', 'Medium Cost','High
Cost','Very High Cost'], ncol=4, bbox_to_anchor=(-0.30, 1.22))
ax1.set_xticklabels(["Doesnt Support","Support"])
ax1.set_xlabel("")
ax1.set_ylabel("")

fig.show()
plt.show()
```

- 4g.py

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.ticker as mtick

pd.set_option('display.max_columns', None)
```

```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
   for filename in filenames:
       print(os.path.join(dirname, filename))

train = pd.read_csv('../../data/train.csv')

fig = plt.figure(figsize=(15,4))
gs = fig.add_gridspec(1, 2)
gs.update(wspace=0.3, hspace=0)
fig.text(0.120,1.1,'4G and its effect on the price ',
fontfamily='serif',fontsize=15, fontweight='bold')
ax0 = fig.add_subplot(gs[0, 0])
ax1 = fig.add_subplot(gs[0, 1])


sns.countplot(x='four_g',
          data=train,
           palette='bone',
          ax=ax0)
ax0.grid(color='gray', linestyle=':', axis='y', zorder=0,
dashes=(1,5))
ax0.set_title('Distribution',fontsize=12,fontfamily='serif',fontwei
ght='bold',x=0.1,y=1)
ax0.spines['top'].set_visible(False)
ax0.spines['right'].set_visible(False)
ax0.spines['left'].set_visible(False)
ax0.set_xticklabels(["Doesnt Support","Support"])
ax0.set_xlabel("")
ax0.set_ylabel("")

sns.countplot(x='four_g',
            data=train,
            hue='price_range',
            ax=ax1)
ax1.grid(color='gray', linestyle=':', axis='y', zorder=0,
dashes=(1,5))
ax1.set_title('Distribution by Price
Range',fontsize=12,fontfamily='serif',fontweight='bold',x=0.25,y=1)
ax1.spines['top'].set_visible(False)
ax1.spines['right'].set_visible(False)
ax1.spines['left'].set_visible(False)
ax1.get_legend().remove()
legend_labels, _= ax1.get_legend_handles_labels()
ax1.legend(legend_labels, ['Low Cost ', 'Medium Cost','High
Cost','Very High Cost'], ncol=4, bbox_to_anchor=(-0.30, 1.22))
ax1.set_xticklabels(["Doesnt Support","Support"])
```

```
ax1.set_xlabel("")
ax1.set_ylabel("")

fig.show()
plt.show()
```

- battery-pover.py

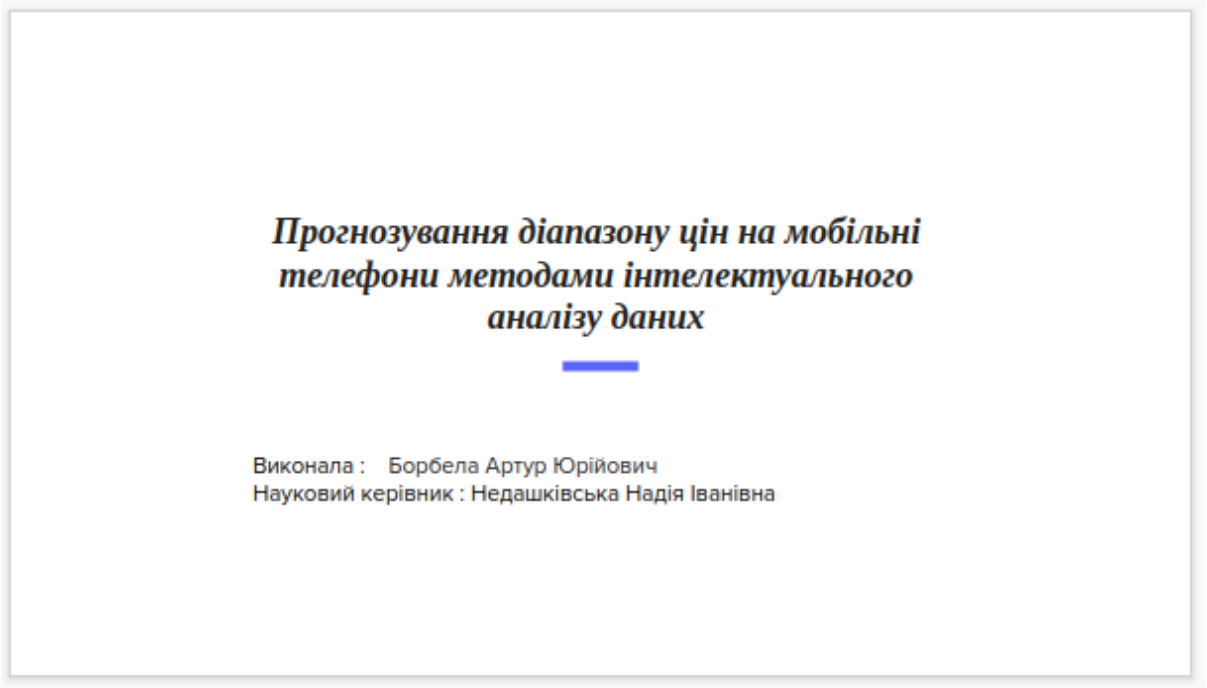
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.ticker as mtick

pd.set_option('display.max_columns', None)



import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

train = pd.read_csv('../../data/train.csv')

sns.set(rc={"figure.dpi": 100})
sns.set_context('paper')
sns.set_style("ticks")

fig = plt.figure(figsize=(15, 12))
gs = fig.add_gridspec(2, 2)
gs.update(wspace=0.3, hspace=0.4)
fig.text(0.085, 0.95, 'Battery power and its effect of the price
range ', fontfamily='serif', fontsize=15,
         fontweight='bold')
sns.set_palette('Spectral')

ax0 = fig.add_subplot(gs[0, 0])
ax1 = fig.add_subplot(gs[0, 1])
ax2 = fig.add_subplot(gs[1, 0], ylim=(0, 3000), xlim=(0, 5))
ax3 = fig.add_subplot(gs[1, 1], ylim=(0, 3000))

# Axis 0

ax0.grid(color='gray', linestyle=':', axis='y', zorder=0,
dashes=(1, 5))
sns.kdeplot(x='battery_power',
            data=train,
            shade=True,
```

```
            ax=ax0,
            linewidth=0
            )
ax0.yaxis.set_major_formatter(mtick.PercentFormatter(1,
decimals=3))
ax0.set_xlabel("")
ax0.set_ylabel("")
ax0.set_title('Distribution', fontsize=12, fontfamily='serif',
fontweight='bold', x=0, y=1)
ax0.spines['top'].set_visible(False)
ax0.spines['right'].set_visible(False)
ax0.spines['left'].set_visible(False)

# Axis 1

ax1.grid(color='gray', linestyle=':', axis='y', zorder=0,
dashes=(1, 5))
sns.kdeplot(x='battery_power',
            hue='price_range',
            shade=True,
            data=train,
            palette='Spectral',
            ax=ax1,
            fill=True,
            alpha=.3,
            linewidth=0

            )
ax1.yaxis.set_major_formatter(mtick.PercentFormatter(1,
decimals=3))
ax1.set_xlabel("")
ax1.set_ylabel("")
ax1.set_title('Distribution by Price Range', fontsize=12,
fontfamily='serif', fontweight='bold', x=0.1, y=1)
ax1.legend(['Low Cost ', 'Medium Cost', 'High Cost', 'Very High
Cost'], fontsize=6, frameon=False)
ax1.spines['top'].set_visible(False)
ax1.spines['right'].set_visible(False)
ax1.spines['left'].set_visible(False)

# Axis 2

ax2.grid(color='gray', linestyle=':', axis='y', zorder=0,
dashes=(1, 5))
sns.boxenplot(y='battery_power',
              data=train,
              ax=ax2,
              zorder=3,
              linewidth=0.4)
```

```
ax2.set_xlabel("")
ax2.set_ylabel("")
ax2.set_title('Distribution', fontsize=12, fontfamily='serif',
fontweight='bold', x=0, y=1.15)
ax2.spines['top'].set_visible(False)
ax2.spines['right'].set_visible(False)
ax2.spines['left'].set_visible(False)

# Axis3

ax3.grid(color='gray', linestyle='-', axis='y', zorder=0,
dashes=(1, 5))
sns.boxenplot(x='price_range',
              y='battery_power',
              data=train,
              ax=ax3,
              zorder=3,
              linewidth=0.4

              )

ax3.set_xlabel("")
ax3.set_ylabel("")
ax3.set_title('Distribution by Price Range', fontsize=12,
fontfamily='serif', fontweight='bold', x=0.1, y=1.15)
ax3.set_xticklabels(['Low Cost ', 'Medium Cost', 'High Cost', 'Very
High Cost'])
ax3.spines['top'].set_visible(False)
ax3.spines['right'].set_visible(False)
ax3.spines['left'].set_visible(False)

fig.show()
plt.show()
```

ДОДАТОК Б ГРАФІЧНОГО МАТЕРІАЛУ ДО ЗАХИСТУ

Слайд 1



Слайд 2

Предмет та об’єкт досліджень

Об’єкт *– дані характеристик мобільних телефонів та їх цінові категорії.*

Предмет дослідження *– моделі інтелектуального аналізу даних для проведення класифікацій на основі обранних даних.*

Мета роботи *– порівняння роботи різних моделей класифікацій та розробка такого програмного продукту, який би міг прогнозувати цінову категорію в залежності від технічних характеристик пристрою.*

Слайд 3

Актуальність роботи

Проблема оптимального вибору телефону виникала у кожного адже всім при покупці хочеться розуміти чи коштує даний пристрій з певним набором характеристик таку суму чи то просто велика комісія посередників які збувають даний товар. Отже можливість приблизно знати цінову категорію даного девайсу допоможить оптимально зробити свій вибір. А вибір в даній галузі дуже великий і майже завжди знайдеться аналог точно ж з такими характеристиками, але вже можливо в іншій ціновій категорії чи від більш надійного виробника.

Слайд 4

Постановка задачі

- Виконати порівняльний аналіз найбільш популярних методів класифікації.
- Вивчити різні метрики для оцінки якості роботи алгоритмів класифікації, визначити переваги кожної метрики.
- Виконати попередній аналіз вхідних даних та зробити висновки про його цілісність та достовірність.
- Обрати найкращі моделі за метриками якості для вирішення практичної задачі прогнозування цінових категорій.

Слайд 5

Наївний Байєсівський класифікатор

Заснований на застосуванні теореми Байєса із наївними припущеннями про незалежність атрибутів:

$$P(c \vee x) = \frac{P(c)\,P(c)}{P(x)}$$

$$P(c|X) = P(x_1 \vee c \times P(c) \times \ldots \times P(x_n \vee c) \times P(c),$$

де P (c | x) - апостеріорна ймовірність даного класу при заданому значенні ознаки x; P (c) - апріорна ймовірність даного класу; P (x | c) – ймовірність даного значення ознаки при даному класі (правдоподібність); P (x) - апріорна ймовірність даного значення ознаки.

Переваги:

- можуть навчатися дуже ефективно,
- мала кількість даних необхідних для навчання, оцінки параметрів і класифікації.

Слайд 6

К-найближчих сусідів(KNN)

- алгоритм запам'ятовує всі вектори ознак і відповідні їм мітки класів
- обчислюється відстань між вектором нового спостереження і існуючим раніше
- вибирається k найближчих векторів до нового об'єкту і новий відноситься до класу, якому належить більшість з них

Недолік: висока обчислювальна трудомісткість, яка збільшується квадратично з ростом числа навчальних прикладів.

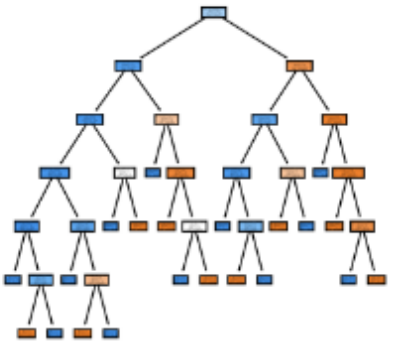
$$a(u) = argmax_{y \in Y} \sum_{i=1}^{m} [y(x_{i;u}) = y] w(i,u)$$

Слайд 7

Беггінг та Random Forest

Метою беггінгу є створення ансамблю моделей та усереднення їх прогнозів для отримання моделі з меншим розкидом,що стане більш надійною за окремі складові.

Випадковий ліс при побудові кожного дерева замість вибору всіх ознак для генерації вибірки обирає та зберігає тільки випадкову їх підмножину для побудови дерева. Алгоритм випадкового лісу поєднує в собі концепції беггінга і вибору підпростору випадкових об'єктів для створення більш стійких моделей.



Слайд 8

Бустинг

- Модель будується по збірці даних.
- Ця модель робить передбачення для всього набору даних.
- За прогнозами і істинним значенням обчислюються помилки.
- Нова модель будується з урахуванням помилок як цільових змінних. При цьому ми прагнемо знайти краще розділення для мінімізації помилки.
- Передбачення, зроблені за допомогою цієї нової моделі, поєднуються з передбаченнями попередніх.
- Знову обчислюються помилки з використанням цих передбачених значень і істинних значень.
- Цей процес повторюється, поки функція помилки не перестане змінюватися або поки не буде досягнуто максимальної кількості предикторів.

Слайд 9

Дерева прийняття рішень (Decision Tree Classifier)

Дерева рішень будуються за допомогою евристиці яка називається рекурсивним розбиттям. Цей алгоритм розбиває дані на підмножини які потім ще багато раз розбиваються на ще більш мілкіші і так далі поки процес не зупиниться коли процес вирішить, що дані в підмножині доволі однорідні чи досягнут інший категорій зупинки.

Переваги: не дорого будувати, швидко класифікують невідомі записи, легко інтерпретуються для невеликих дерев, доволі висока точність на невеликих та простих даних.

Недоліки: схильність до перенавчання, склоність розбиття по функціям з великим рівнями складності, не стійкість до невеликих змін у навчальній вибірці, складність при інтерпретації великих дерев.

Слайд 10

Стохастичний градієнтний спуск

Стохастичний градієнтний спуск - дуже популярний і розповсюджений алгоритм який використовують в різних алгоритмах машиного навчанняю.

Його потрібно розглядати в якості оптимізатора для ефективного навчання лінійних класифікаторів при великої кількості даних та характеристик. Також гарно підходить для постійного навчання наприклад у випадках коли нам необхідно щоб алгоритм динамічно адаптувався до нових шаблонів даних.

Недоліки: чутливий до масштабування характеристик, потребує точного настроювання ряда гіперпараметрів, включає параметр регуляризації і кількості ітерацій для гарної продуктивності.

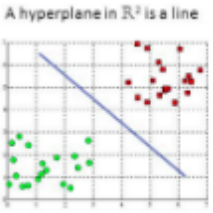
Слайд 11

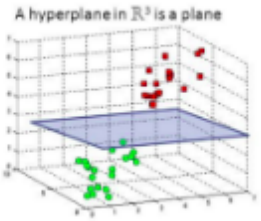
Метод опорних векторів (SVM)

Мета даного алгоритму знайти гіперплощину в N-мірному просторі (N - кількість функцій), яка добре розбиває точки даних на класи.

Переваги: добре працює з чітким розбиттям, ефективна на даних з багатьма характеристиками, ефективний з точки зору пам'яті.

Недоліки: погана робота на великих даних із-за великого часу навчання, погано відпрацьовує якщо в дані мають багато викидів, не дає напряму оцінку ймовірностей вони вираховуються за допомогою ресурсозатратної перевірки.





Слайд 12

Логістична регресія

Оскільки модель лінійної регресії погано підходить для задачі класифікації із-за того що прогнозований нею результат це не ймовірність, а лінійна інтерполяція між точками то не існує значення порога при якому ми можемо відрізнити один клас від іншого. Замість апроксимації прямої чи гіперплощини в її основу входить логістична функція яка зжимає вихідні дані лінійного рівняння в діапазон від 0 до 1 й розраховується за формулою:

$$\text{logistic}(\eta) = \frac{1}{1 + exp(-\eta)}$$

Багато переваг та недоліків моделі лінійної регресій також містять в собі й логістична регресія. Ще один недолік логістичної регресії в тому що етап інтерпретації більш складний оскільки сама інтерпретація являється мультиплікативною а не аддитивной. Також в ній є проблема повного розбиття коли є функція яка добре розбиває на два класи модель вже не може далі навчатися.

Слайд 13

Дискримінантний аналіз

Лінійний дискримінантний аналіз (ЛДА), а також пов'язаний з ним лінійний дискримінант Фішера - методи статистики та машинного навчання для знаходження лінійних комбінацій признаків які найкращим чином розбивають два чи більше класів елементів чи подій. В результаті цього отримана комбінація може бути використана в якості лінійного класифікатора або для зменшення розмірності простору характеристик перед наступною класифікацією.

Переваги: порівняна простоту реалізації та інтерпретованості результатів.
Недоліки: чутливість до розподілу вихідних даних, коли навіть не велика їх зміна призводить до значущих змін результатів класифікації.

Слайд 14

Обрані для роботи моделі

- Naive Bayes Classification Gaussian
- Naive Bayes classifier Multinomial
- Logistic Regression
- Random Forest Classifier
- Ada Boosting Classifier
- Support Vector Machine
- Decision Tree Classifier
- Nearest Neighbors
- Gradient Boosting Classifier
- Stochastic Gradient Descent
- Linear Discriminant Analysis
- Quadratic Discriminant Analysis

Слайд 15

Використані метрики

1.Accuracy $$Accuracy = \frac{TruePositive + TrueNegative}{TotalSample}$$

2.F1-міра $$F1 - 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

3.Precision $$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

4.Recall $$\frac{TruePositives}{TruePositives + FalseNegatives}$$

5.Confusion Matrix

	Predicted: "No"	Predicted: "Yes"
Actual: "No"	50	10
Actual: "Yes"	5	100

Слайд 16

Опис набору даних

Датасет був взятий з відкритого джерела. Він містить в собі записи структура яких виглядає наступним чином 20 features variables and 1 target variable. Змінні характеристик складають із 6 бінарних та 14 неперервних.

80% даних були відібрані для навчальної вибірки,решта 20% - для тестування.

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	...	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	...	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.046500	...	12.306500	5.767000	11.011000	0.761500	0.503000	0.507000	1.500000
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.145715	...	4.213245	4.356398	5.463955	0.426273	0.500116	0.500076	1.118314
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.000000	...	5.000000	0.000000	2.000000	0.000000	0.000000	0.000000	0.000000
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.000000	...	9.000000	2.000000	6.000000	1.000000	0.000000	0.000000	0.750000
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.000000	...	12.000000	5.000000	11.000000	1.000000	1.000000	1.000000	1.500000
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.000000	...	16.000000	9.000000	16.000000	1.000000	1.000000	1.000000	2.250000
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.000000	...	19.000000	18.000000	20.000000	1.000000	1.000000	1.000000	3.000000

Слайд 17

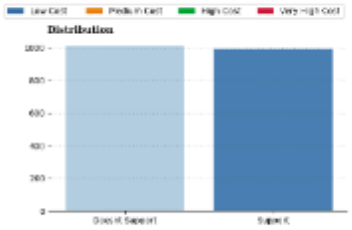
Опис атрибутів для кожного об’єкта

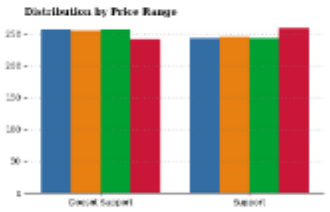
1. Наявність в пристрої технології Bluetooth (“blue”).
2. Наявність можливості використовувати в пристрої дві сім карти (“dual_sim”).
3. Наявність технології 4G (“four_g”).
4. Наявність технології 3G (“three_g”).
5. Наявність підтримки технології Wi-Fi (“wi-fi”).
6. Наявність сенсорного екрану (“touch_screen”).
7. Швидкість виконання процесором певної кількості інструкцій (“clock_speed”).
8. Загальна електроємність яку аккумулятор может втримувати за одну підзарядку в mAh. (“battery_power”)
9. Розширення фронтальна камера в mega pixels. (“fc”)
10. Об’єм внутрішньої пам’яті в гигабайтах (“int_memory”).
11. Розмір висоти телефону в см (“m_dep”).
12. Вага мобільного телефону (“mobile_wt”).
13. Кількість ядрів на одному процесорі (“n_cores”).
14. Розширення основної камери в мегапікселях (“pc”).
15. Висота роздільної здатності екрану в пікселях (“px_height”).
16. Ширина роздільної здатності екрану в пікселях (“px_width”).
17. Розмір оперативної пам’яті в мегабайтах (“ram”).
18. Довжина екрана в сантиметрах (“sc_h”).
19. Ширина екрана в сантиметрах (“sc_w”).
20. Час на якій вистачає неперервної роботи телефону в стані дзвінку за один заряд аккумулятора. (“talk_time”).

Слайд 18

Наявність в пристрої технології Bluetooth

count	mean	std	min	25%	50%	75%	max
2000	0.4950	0.5001	0	0	0	1	1

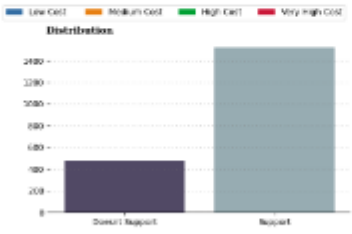


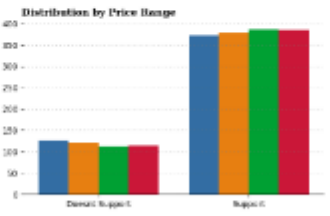


Слайд 19

Наявність технології 3G

count	mean	std	min	25%	50%	75%	max
2000	0.761500	0.426273	0	1	1	1	1

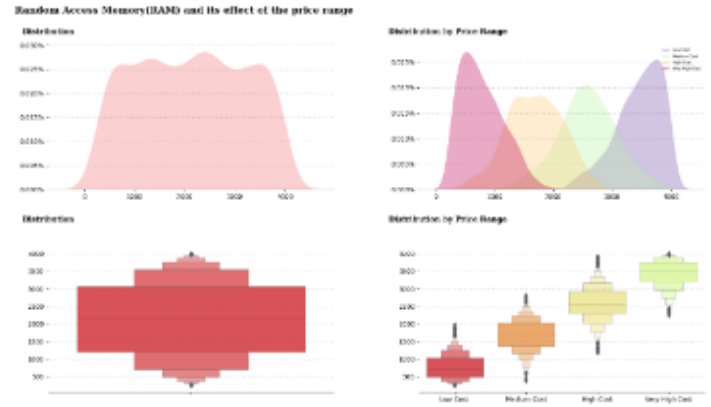




Слайд 20

Розмір оперативної пам'яті

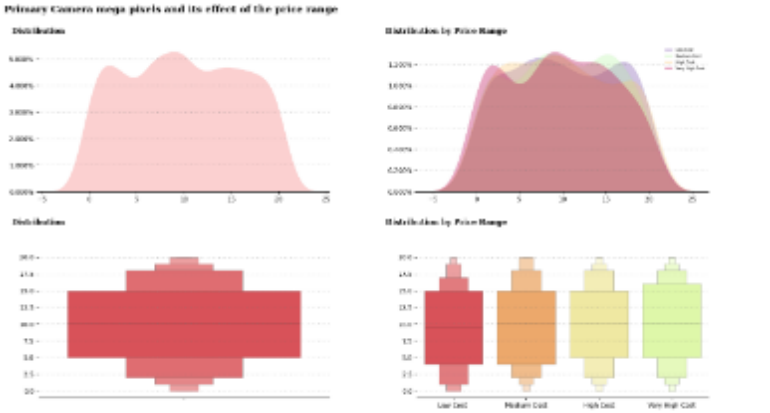
count	mean	std	min	25%	50%	75%	max
2000	2124.21300	1084.732044	256.00	1207.50	2146.500	3064.500	3998.00



Слайд 21

Розширення основної камери

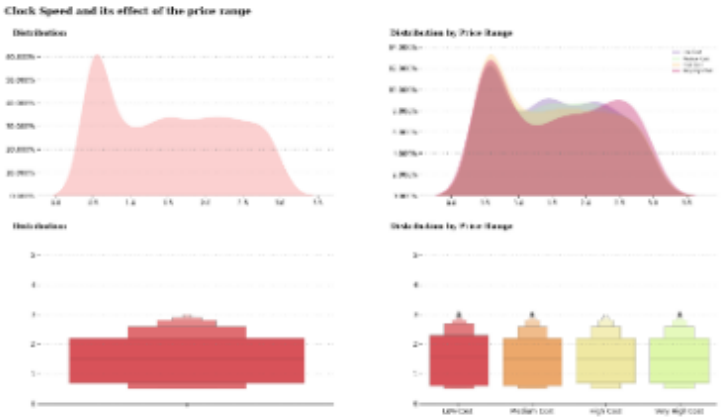
count	mean	std	min	25%	50%	75%	max
2000	9.9116500	6.064315	0.00	5.00	10.00	15.00	20.00



Слайд 22

Швидкість виконання процесором певної кількості інструкцій

count	mean	std	min	25%	50%	75%	max
2000	1.522250	0.816004	0.5	0.7	1.5	2.2	3

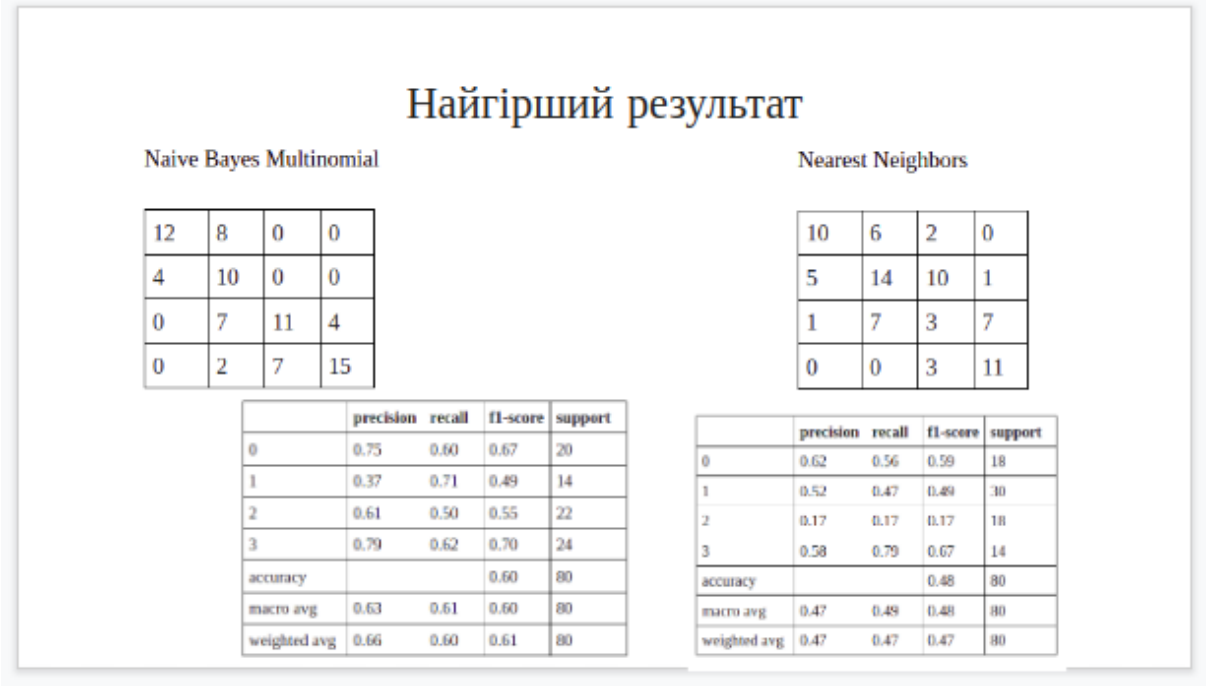


Слайд 23

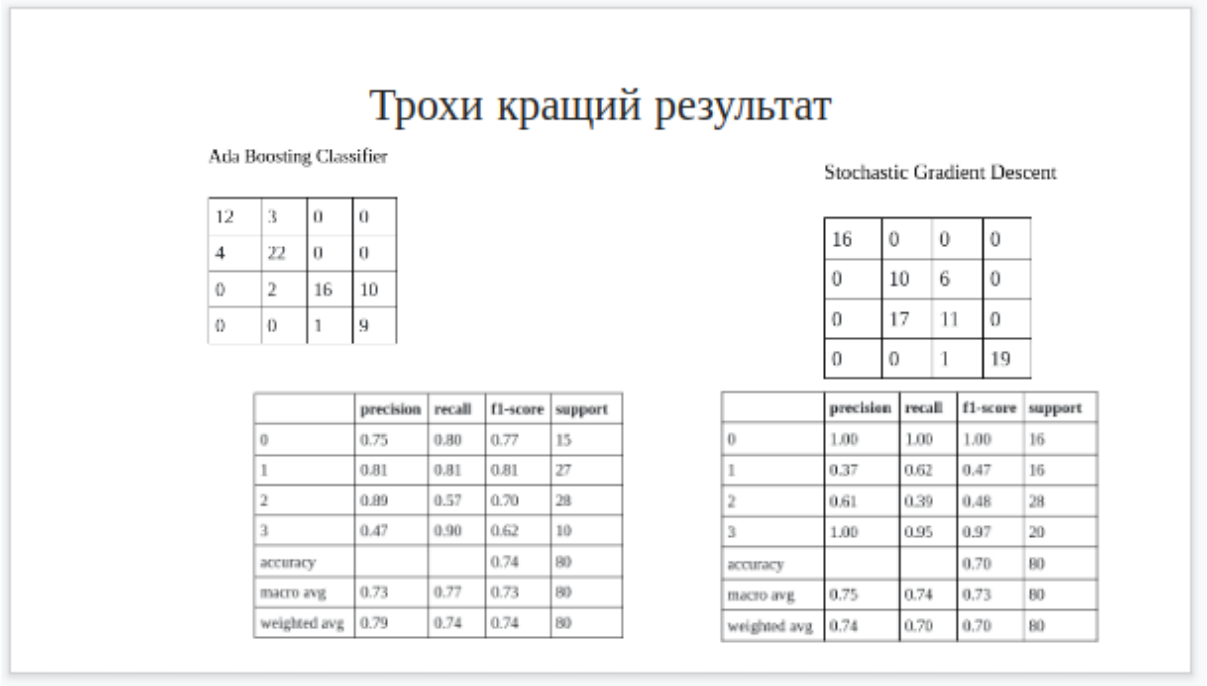
Порівняння результатів роботи класифікації алгоритмів

1. Naive Bayes Classification Gaussian, Decision Tree Classifier
2. Naive Bayes classifier Multinomial, K Nearest Neighbors
3. Logistic Regression, Random Forest Classifier
4. Ada Boosting Classifier, Stochastic Gradient Descent
5. Support Vector Machine, Gradient Boosting Classifier
6. Linear Discriminant Analysis, Quadratic Discriminant Analysis

Слайд 24



Слайд 25



Слайд 26

Трохи кращий результат

Naive Bayes Classification

11	1	0	0
5	19	0	0
0	7	16	1
0	0	2	18

	precision	recall	f1-score	support
0	0.69	0.92	0.79	12
1	0.70	0.79	0.75	24
2	0.89	0.67	0.76	24
3	0.95	0.90	0.92	20
accuracy			0.80	80
macro avg	0.81	0.82	0.80	80
weighted avg	0.82	0.80	0.80	80

Decision Tree Classifier

12	0	0	0
4	24	1	0
0	3	15	1
0	0	2	18

	precision	recall	f1-score	support
0	0.75	1.00	0.86	12
1	0.89	0.83	0.86	29
2	0.83	0.79	0.81	19
3	0.95	0.90	0.92	20
accuracy			0.86	80
macro avg	0.85	0.88	0.86	80
weighted avg	0.87	0.86	0.86	80

Слайд 27

Кращий результат

Logistic Regression

14	0	0	0
2	27	0	0
0	7	17	0
0	0	1	19

	precision	recall	f1-score	support
0	0.88	1.00	0.93	14
1	1.00	0.93	0.96	29
2	0.94	1.00	0.97	17
3	1.00	0.95	0.97	20
accuracy			0.96	80
macro avg	0.95	0.97	0.96	80
weighted avg	0.97	0.96	0.96	80

Random Forest Classifier

14	1	0	0
2	20	0	0
0	6	18	1
0	0	0	18

	precision	recall	f1-score	support
0	0.88	0.93	0.90	15
1	0.74	0.91	0.82	22
2	1.00	0.72	0.84	25
3	0.95	1.00	0.97	18
accuracy			0.88	80
macro avg	0.89	0.89	0.88	80
weighted avg	0.89	0.88	0.87	80

Слайд 28

Кращий результат

Support Vector Machine

15	1	0	0
1	21	1	0
0	5	17	1
0	0	0	18

	precision	recall	f1-score	support
0	0.94	0.94	0.94	16
1	0.78	0.91	0.84	23
2	0.94	0.74	0.83	23
3	0.95	1.00	0.97	18
accuracy			0.89	80
macro avg	0.90	0.90	0.89	80
weighted avg	0.90	0.89	0.89	80

Gradient Boosting Classifier

14	0	0	0
2	22	0	0
0	5	17	1
0	0	1	18

	precision	recall	f1-score	support
0	0.88	1.00	0.93	14
1	0.81	0.92	0.86	24
2	0.94	0.74	0.83	23
3	0.95	0.95	0.95	19
accuracy			0.89	80
macro avg	0.90	0.90	0.89	80
weighted avg	0.89	0.89	0.89	80

Слайд 29

Найкращий результат

Linear Discriminant Analysis

14	0	0	0
2	26	0	0
0	1	18	0
0	0	0	19

	precision	recall	f1-score	support
0	0.88	1.00	0.93	14
1	0.96	0.93	0.95	28
2	1.00	0.95	0.97	19
3	1.00	1.00	1.00	19
accuracy			0.96	80
macro avg	0.96	0.97	0.96	80
weighted avg	0.97	0.96	0.96	80

Quadratic Discriminant Analysis

15	0	0	0
1	25	0	0
0	2	18	0
0	0	0	19

	precision	recall	f1-score	support
0	0.94	1.00	0.97	15
1	0.93	0.96	0.94	26
2	1.00	0.90	0.95	19
3	1.00	1.00	1.00	19
accuracy			0.96	80
macro avg	0.96	0.97	0.96	80
weighted avg	0.97	0.96	0.96	80

Слайд 30

Висновки

- Виконано порівняльний аналіз найбільш популярних методів класифікації.
- Вивчено різні метрики для оцінки якості роботи алгоритмів класифікації, визначити переваги кожного метрики.
- Виконано попередній аналіз вхідних даних та перевірено цілісність та правдивість даної вибірки.
- Побудувано моделі нейронних мереж та різних ансамблів для опису вибірки даних характеристик мобільних телефонів.
- Обрано найкращі моделі за метриками якості для вирішення практичної задачі прогнозування цінових категорій мобільних телефонів.