НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Навчально-науковий інститут прикладного системного аналізу
Кафедра штучного інтелекту

До захисту допущено:
Завідувач кафедри
_____ Олена ЧУМАЧЕНКО
«___»_____2023 р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системи і методи штучного інтелекту»
спеціальності 122 «Комп'ютерні науки»
на тему: «Прогнозування ціни вживаних автомобілів на основі їх характеристик з використанням алгоритмів машинного навчання»

Виконав:
студент IV курсу, групи КІ-93
Коваль Павло Сергійович _____

Керівник:
к.ф-.м.н., доцент Барановська Л.В. _____

Консультант з нормоконтролю:
фахівець першої категорії, Гончарук М.М. _____

Консультант з економічного розділу:
доцент, к.е.н., Рощина Н.В. _____

Рецензент:
доцент, к.т.н., Харченко К.В. _____

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

Київ, 2023

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

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 122 «Комп'ютерні науки»
Освітня програма «Системи і методи штучного інтелекту»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олена ЧУМАЧЕНКО
«__» _____ 2023 р.

ЗАВДАННЯ
на дипломну роботу студенту
Ковалю Павлу Сергійовичу

1. Тема роботи «Прогнозування ціни вживаних автомобілів на основі їх характеристик з використанням алгоритмів машинного навчання», керівник роботи Барановська Леся Валеріївна, доцент кафедри ММСА, затверджені наказом по університету від «30» травня 2023 р. № 2065-с.
2. Термін подання студентом роботи 16.06.2023.
3. Вихідні дані до роботи: дані продажів вживаних автомобілів на ринку Індії.
4. Зміст роботи: 1 – Дослідження предметної області; 2 – Методи штучного інтелекту для прогнозування ціни автомобілів; 3 – Програмна реалізація та аналіз результатів; 4 – Функціонально-вартісний аналіз програмного забезпечення.
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): схема роботи програми та алгоритмів машинного навчання.

6. Консультанти розділів роботи.

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	доцент, к.е.н., Рощина Н.В.		

7. Дата видачі завдання 20.02.2023

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Вивчення літератури за темою роботи.	18.04.2023	Виконано
2.	Підготовка першого розділу.	25.04.2023	Виконано
3.	Підготовка другого розділу.	01.05.2023	Виконано
4.	Початок розробки програмного продукту.	05.05.2023	Виконано
5.	Підготовка третього розділу.	29.05.2023	Виконано
6.	Підготовка економічної частини.	31.05.2023	Виконано
7.	Оформлення розділів відповідно до нормконтролю.	01.06.2023	Виконано
8.	Підготовка презентації доповіді.	05.06.2023	Виконано
9.	Оформлення дипломної роботи.	07.06.2023	Виконано

Студент Павло КОВАЛЬ

Керівник Леся БАРАНОВСЬКА

АНОТАЦІЯ

Дипломна робота: 120 с., 5 табл., 56 рис., 1 додаток, 18 джерел.

ПРОГНОЗУВАННЯ ЦІНИ, ШТУЧНИЙ ІНТЕЛЕКТ, МАШИННЕ НАВЧАННЯ, НАВЧАННЯ З ВЧИТЕЛЕМ, РИНОК ВЖИВАНИХ АВТОМОБІЛІВ.

Об’єкт дослідження – прогнозування ціни вживаних автомобілів на основі їх характеристик.

Предмет дослідження – методи штучного інтелекту, які використовуються для задач прогнозування ціни автомобілів.

Мета роботи – проаналізувати предмет дослідження, розробити програмний продукт за допомогою методів штучного інтелекту для визначення ціни автомобіля.

Методи дослідження – методи штучного інтелекту: алгоритми машинного навчання, а саме лінійна регресія та випадковий ліс.

Актуальність – задача розробки нової системи для покращення процесу оцінки ціни автомобіля. Зменшення витрат на помилкові рішення. Оптимізація роботи автосалонів за рахунок оперативності при оцінці вартості автомобілів.

Результати роботи – створено програмний продукт, який використовує методи штучного інтелекту, які можуть бути використані для прогнозування ціни автомобілів.

Шляхи подальшого розвитку предмету дослідження – використання більшого набору даних з залученням ринку вживаних автомобілів України, США та Європи. Створення повноцінного графічного інтерфейсу для спрощення роботи користувача з програмою, побудова складнішої та більш точної системи для прогнозування ціни автомобілів.

ANNOTATION

Diploma thesis: 120 p., 5 tabl., 56 fig., 1 appendice, 18 source.

PRICE FORECASTING, ARTIFICIAL INTELLIGENCE, MACHINE LEARNING, SUPERVISED LEARNING, USED CAR MARKET.

Object of research - predicting the price of used cars based on their characteristics.

The subject of the study is artificial intelligence methods used for car price forecasting tasks.

Purpose - to analyze the subject of research, develop a software product using artificial intelligence methods to determine the price of a car.

Research methods - artificial intelligence methods: machine learning algorithms, namely linear regression and random forest.

Relevance - the task of developing a new system to improve the process of estimating the price of a car. Reducing the cost of wrong decisions. Optimizing the work of car dealerships through efficiency in car valuation.

Results - a software product has been created that uses artificial intelligence methods that can be used to predict the price of cars.

Ways to further develop the subject of research - use of a larger data set involving the used car market in Ukraine, the United States and Europe. Creating a full-fledged graphical interface to simplify the user's work with the program, building a more complex and accurate system for predicting car prices.

ЗМІСТ

ВСТУП

Сьогодні ринок вживаних автомобілів розвивається дуже швидко, і це зробило проблему оцінки ціни автомобілів важливим питанням. Оцінка вартості автомобіля є складним процесом, що потребує детального аналізу його характеристик.

Збільшення кількості населення безперечно є однією з головних причин зростання попиту на автомобілі. Чим більше людей живуть в місті, тим більше стає потреба в зручних та ефективних засобах транспорту, що дозволяють людям легко дістатись до роботи, навчальних закладів та інших місць.

Зростання населення також призводить до збільшення окремих районів чи міст. Збільшення відстаней на які люди повинні їздити в рамках міста чи певного регіону зумовлює потребу в більшій кількості транспортних засобів. Це в свою чергу підсилює попит на автомобілі, оскільки вони є одним з найбільш доступних та популярних засобів транспорту, особливо в тій місцевості де громадський транспорт погано розвинений.

Під час виконання даної роботи необхідно буде виконати дослідження, яке включає аналіз різних характеристик автомобілів, таких як рік випуску, пробіг, об'єм двигуна, тип палива, кількість власників та інші. Буде використано моделі машинного навчання, такі як лінійна регресія та випадковий ліс, для прогнозування вартості автомобіля.

Також необхідно буде звернути увагу на дослідження важливості кожної з характеристик автомобіля для визначення його ціни. Це дозволить краще зрозуміти, які характеристики мають найбільший вплив на вартість автомобіля та як їх можна використовувати для зменшення ризиків вкладень під час покупки чи продажу.

Дослідження, яке проведено у представленій роботі, має практичне застосування в автомобільній індустрії та може бути використане як в особистих, так і в комерційних цілях. Це дозволяє економити час та гроші, які зазвичай витрачаються на оцінку вартості автомобіля вручну.

У даній роботі буде досліджено можливості використання аналітичних методів та машинного навчання для прогнозування вартості вживаних автомобілів на основі їх характеристик. Даний продукт буде мати важливе значення для покупців та продавців автомобілів, оскільки дозволить точно визначити ринкову вартість автомобіля.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність процесу прогнозування ціни автомобілів

Станом на зараз, світовий ринок автомобілів являє собою одни з найбільш високорозвинених товарних ринків у світі. На ньому зосереджено увагу великої кількісті виробників та продавців автотранспортних засобів. Даний ринок характеризується неперервною та жорсткою конкуренцією між виробниками, зокрема у секторі легкових авто. Світові автоконцерни активно борються за своїх споживачів та постійно шукають нові ринки збуту для своїх автомобілів.

Автомобільний ринок відіграє важливу роль у глобальній економіці. Щорічно у світі виробляється приблизно 82 мільйони легкових автомобілів, включаючи легкові комерційні автомобілі, на загальну суму близько 2 трильйонів доларів США. Разом з суміжними галузями, автомобільна промисловість складає майже 5% світового ВВП. Автомобільна галузь є унікальною складовою світової економіки, яка демонструє стійкий ріст та сприяє розвитку економіки в цілому. За останні 10 років глобальний автомобільний ринок показав 30% зростання. Якби автомобільна промисловість була окремою країною, вона займала б шосте місце за величиною економіки у світі. Світова автомобільна галузь, як один з основних секторів світового господарства, потребує значних трудових ресурсів [1]. Прямо в автомобільній промисловості працює приблизно 9 мільйонів людей, що становить більше 5% від загальної кількості зайнятого населення в промисловості світу. Кожне робоче місце в автомобільній промисловості впливає на забезпечення зайнятості для інших 5 працівників у суміжних галузях, що становить 50 мільйонів зайнятих осіб в автомобільній та суміжних індустріях. Виробництво автомобілів включає використання різних товарів інших галузей, таких як метал, сталь, скло, текстиль, каучук та інші матеріали [2].

Також є зрозумілим той факт, що населення планети постійно збільшується. Як наслідок це спричиняє збільшення чисельності та розмірів міст у світі. Не завжди

громадський транспорт встигає за таким темпом росту та залишає непокриті ділянки в місті чи пригороді. Тому у мешканців існує проблема у пересуванні. Це питання може вирішити автомобіль, який люди активно починають купляти.

Рисунок 1.1 - Кількість автомобілів у місті Києві за останні 7 років [3]

Як можна побачити з рис.1 то у 2022 році у місті Києві було зареєстровано більше 1,3 мільйона автомобілів. Тобто майже у кожного другого мешканця столиці

є хоча б по одному автомобілю. Зрозуміло, що така кількість створює великий ринок продажу та купівлі автомобілів.

Також як привило середню пересічну людину більше цікавлять вживані автомобілі, ніж нові. Це пояснюється тим, що платоспроможність на вживану автівку менше. Тобто треба менше заплатити грошей, щоб стати новоспеченим автомобілістом.

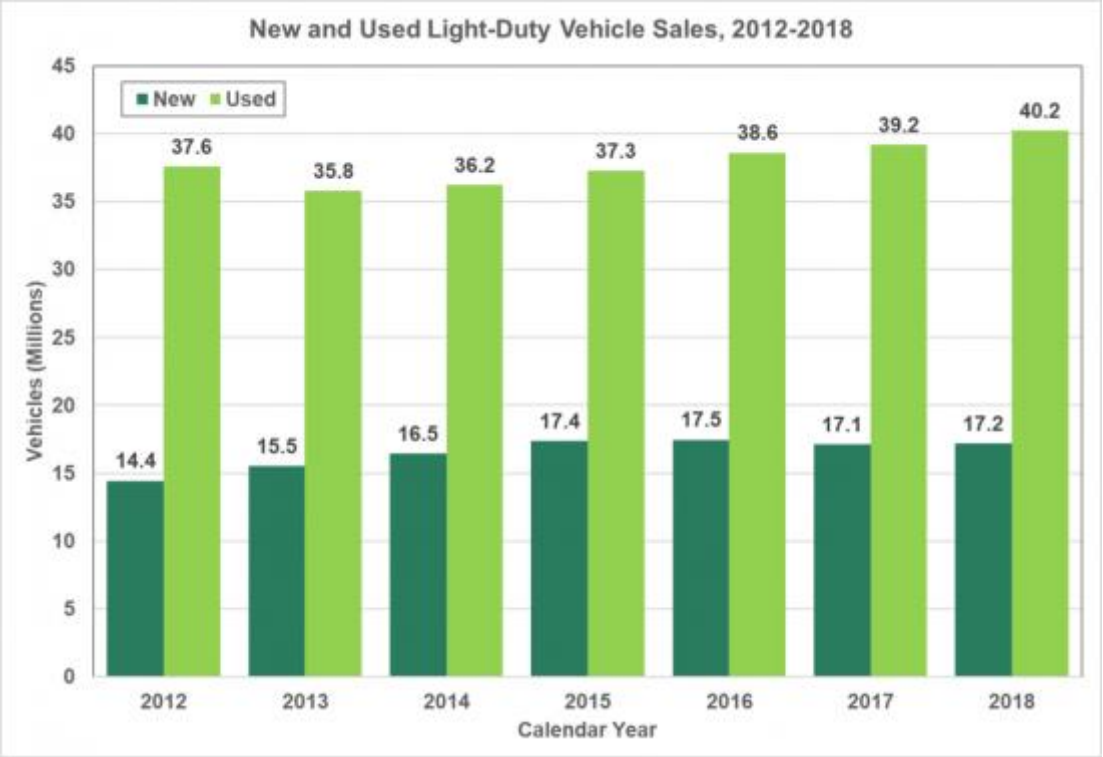

Рисунок 1.2 - Кількість проданих нових та вживаних автомобілів у США з 2012 по 2018 роки [4]

За даними, представленими на рис. 2, можна спостерігати, що у 2018 році кількість проданих вживаних автомобілів в США була більше ніж вдвічі перевищувала кількість проданих нових автомобілів.

Отже, можна зробити висновок, що дослідження та прогнозування вартості вживаних автомобілів є доволі актульним та затребуваним у наш час.

1.2 Основні поняття з автомобільної індустрії

Тематика автомобільного ринку є доволі багатогранною. Кожний день даний вид транспорту продається чи купується за певною вставленою ціною.
На формування даної вартості впливають багато чинників. Також згодом під впливом тих самих факторів вона може змінюватися. Нижче розпишемо основні поняття з теми автомобільної індустрії, які впливають на формування ціни автомобіля.

Почнемо з марки автомобіля. Вона відноситься до певного виробника, який виробляє цей транспортний засіб. Марка є ідентифікаційним елементом кожного автомобіля, оскільки вона поєдную політику, концепцію, дизайн, стиль та особливі технічні рішення конкретної компанії, яка стояла за виробництвом даного продукту. Приведемо приклад відомих автомобільних марок такі як: BMW, Toyota, Ford, Mercedes-Benz, Honda, Audi, Chevrolet та багато інших. Кожна з них має свою нішу або спеціалізацію, яка спрямована на конкретні сегменти ринку. Даними сегментами можуть виступати спортивні автомобілі, позашляховики, економічні автомобілі або автомобілі преміум-класу. Вибір марки є одним із головних аспектів при купівлі автомобіля, оскільки різні марки мають певну репутацію серед автомобілістів, яка була основана на принципах якості, надійності, стилю, простоти обслуговування та новаторства у залучених технологіях.

Модель автомобіля - це вже конкретний тип або версія автомобіля, яку випускає виробник під своєю маркою. Кожна модель вирізняється своїм дизайном, технічними характеристиками, розмірами, використаними матеріалами та особливостями конструкції. Також потрібно розуміти, що модель автомобіля може мати свою унікальну назву, яка уособлює певні особливості та призначення даної моделі. Розглянемо приклад такої відомої марки як BMW. У неї існують різні моделі, наприклад BMW 3 Series, BMW X5, BMW i8 тощо. Кожна модель може бути

вироблена у певному кузові (седан, купе, хетчбек, позашляховик і т.д.), з різними варіаціями двигуна (бензиновий, дизельний, електричний), у поєднанні з переднім або повним приводом. Вибір конкретної моделі автомобіля залежить від потреб та вимог покупця. Різні моделі мають різні рівні комфорту, технічні характеристики, технологічні новинки та інші функції. Як правило кожну модель створюють під її цільову аудиторію. Наприклад, молодь у більшості полюбляє швидкі автомобілі, у той час як люди старшого віку цінують практичність та невибагливість. Ознайомлення з властивостями модельного ряду певної марки допомагає покупцям зробити обґрунтований вибір автомобіля, який відповідає їхнім потребам, стилю життя та бюджету.

Рік випуску автомобіля вказує на календарний рік, коли конкретна модель автомобіля була виготовлена на заводі. Це є досить важливою та однією з головних характеристик, яка напряму впливає на вартість автомобіля. У світовій практиці заведено, що оновлення технологій та дизайну автомобіля відбуваються щорічно або через деякий період часу. З цього випливає, що нові моделі автомобілів стають більш технічно досконалими, отримують додаткові функції та оновлений зовнішній вигляд. Не слід також забувати, що рік виробництва впливає на доступність запасних частин та доступність сервісного обслуговування. Тобто, на старі автомобілі з кожним року стає все важче знайти запчастини, а при зверненні до станції технічного обслуговування потрібних фахівців, які проведуть якісний ремонт по даній застарілій моделі, може вже не бути. Рік випуску автомобіля також враховується при оцінці вартості. Тому нові моделі зазвичай мають вищу вартість, ніж старі. Проте, існують випадки, коли старі моделі можуть мати колекційну цінність, наприклад дану модель по світу у свій час виготовили обмеженою кількістю. Це може збільшити її вартість з часом.

Загальна назва, технічні характеристики автомобіля, включає різноманітні аспекти, які описують його конструкцію, розміри, потужність, паливну систему, передачі, підвіску та інші технічні параметри. Основні технічні характеристики автомобіля можуть включати наступні пункти:

- Двигун: Вказує на тип двигуна (бензиновий, дизельний, гібридний, електричний) та його об'єм (наприклад, 2.0 літра).
- Потужність: Вимірюється в кіловатах (кВ) або кінських силах (к.с.) і вказує на максимальну потужність, яку може розвивати двигун.
- Крутний момент: Вимірюється в ньютон-метрах (Нм) і описує силу, яку може виробити двигун на валу колінчастого вала. Крутний момент впливає на прискорення автомобіля та його потужність на дорозі.
- Трансмісія: Вказує на тип передачі, яка може бути механічною (механічна скринька передач), автоматичною (автоматична скринька передач) або роботизованою (передачі з керуванням електронікою).
- Паливна система: Вказує на тип палива, який використовує автомобіль (бензин, дизель, газ, електрика) та систему подачі палива.
- Розміри: Включає довжину, ширину, висоту та вагу автомобіля, що впливають на його маневреність, просторість та стійкість на дорозі.
- Підвіска: Описує тип підвіски (незалежна, залежна, пневматична тощо), яка впливає на комфорт та якість керування автомобілем.
- Розгін та максимальна швидкість: Вказує на час, за який автомобіль може прискоритись з нуля до певної швидкості (наприклад, від 0 до 100 км/год), а також на максимальну швидкість, яку він може досягти.
- Тип приводу: Вказує на тип приводу автомобіля - передній привід, задній привід або повний привід. Це впливає на тягові характеристики та поведінку автомобіля на дорозі.
- Витрата палива: Вказує на середню витрату палива автомобілем на певній відстані або в місті/за містом. Це важлива характеристика для оцінки економічності автомобіля.
- Тип кузова: Вказує на форму та стиль кузова автомобіля, такі як седан, купе, хетчбек, кросовер, вантажівка тощо. Він визначає просторість, функціональність та зовнішній вигляд автомобіля.

- Системи безпеки: Включає різні системи та пристрої безпеки, такі як система контролю стабільності (ESP), системи гальмування, подушки безпеки, адаптивний круїз-контроль, система помічників при паркуванні тощо.

Усі ці технічні характеристики допомагають покупцям оцінити вартість автомобіля перед придбанням. У залежності від вимог покупця кожна характеристика може мати як первинне так і вторинне значення.

Також при оцінці та прогнозуванні вартості автомобіля не слід забувати про такі параметри як загальний стан, пробіг та кількість власників. Загальний стан автомобіля описує його фізичний вигляд, технічну працездатність та рівень зносу його деталей чи обладнання. Він впливає на якість та надійність автомобіля. Пробіг вказує на відстань, яку автомобіль пройшов з моменту його виготовлення. Як правило вимірюється в кілометрах або милях. Пробіг може вказувати на те, як інтенсивно автомобіль використовувався та як добре було здійснено технічне обслуговування. Зазвичай, чим менший пробіг, тим кращий стан автомобіля. Кількість власників автомобіля вказує на те, скільки осіб володіло ним протягом часу його експлуатації. Даний факт впливає на історію обслуговування, рівень догляду та стан автомобіля. Загалом автомобілі з меншою кількістю власників вважаються більш привабливими, оскільки їх історія та період обслуговування можуть бути краще відомими.

Якщо підсумувати, то розуміння основних понять з автомобільної індустрії допомагає краще оцінити його вартість. Спираючись на проаналізовані параметри автомобіля, стає легше прийняти обґрунтовані рішення щодо його купівлі, продажу, страхування або фінансування.

1.3. Аналіз існуючих рішень для прогнозування вартості автомобіля

Загалом прогнозування вартості автомобіля є досить важливим та водночас важким процесом, оскільки необхідно звернути свою увагу на безліч параметрів та, проаналізувавши їх, прийти до якісної оцінки вартості.

На допомогу у вирішенні даної задачі приходять різні методи та підходи, з якими ознайомимося нижче.

Почнемо з статистичного аналізу даних. Даний підхід використовує аналізу зібраних даних для отримання закономірностей, тенденцій або інших значущих висновків. Він передбачає використання статистичних методів аналізу даних[5]. Насамперед це можуть бути середні значення, медіани, варіація та кореляція. Наприклад, можна прорахувати середній процент знецінення автомобілів певної марки та моделі протягом певного періоду часу.

Якщо перейти до регресійного аналізу, то це спосіб математичного визначення того, яка з зі змінних дійсно має вплив на вартість автомобіля. Він відповідає на питання: Які фактори мають найбільше значення? Які ми можемо ігнорувати? Як ці фактори взаємодіють між собою? І, можливо, найважливіше, наскільки ми впевнені у всіх цих факторах? [6].У рамках автомобільної тематики прогнозування вартості проводиться на основі незалежних змінних, таких як рік випуску, пробіг, марка та модель, тип двигуна тощо. Проводиться аналіз залежності та вплив кожної змінної на вартість автомобіля.

Основним методом, який буде використано у даній роботі, є метод машинного навчання. До його числа входять, наприклад, лінійна регресія, дерева рішень, випадковий ліс або нейронні мережі. Зазвичай використовуються великі набори даних, які містять історичні ціни та відомості про автомобілі для тренування моделей та передбачення майбутніх цін. Більш детально про даний метод та його користь буде описано у наступному пункті.

У світовій практиці існує ще метод експертної оцінки для прогнозування вартості автомобіля. Висновок експерта формується на основі знань та досвіду у даній галузі. Вони аналізують різні фактори, такі як ринкові тенденції, популярність марок та моделей, технічний стан, історію обслуговування та інші фактори, щоб зробити правильну та доволі точну оцінку. Проте, у час цифровізації рішення експерта хотілося би масштабувати для будь-кого та автоматизувати для кращої працеспроможності та швидкості. Тому необхідність у системах, які виконують оцінку вартості автомобіля є великою.

Існує вже доволі багато веб-ресурсів, які виконують даний аналіз та прогнозування вартості. Хочеться згадати про Carfax, Kelley Blue Book , CarGurus, TrueCar та AutoTrader. Усі ці додатки об’єднує те, що вони є одними з найвідоміших ресурсів для оцінки та прогнозування вартості автомобілів. Дані ресурси надають користувачам можливість ввести дані про автомобіль, такі як марка, модель, рік, пробіг, стан тощо, і отримати оцінку його поточної вартості. Використовують велику кількість даних та аналітику для визначення реальних цін автомобілів на ринку. Також надають можливість встановити фільтри та критерії пошуку. Досвід наведених застосунків був прийнятий та проаналізований для проведення дослідження у рамках обраної теми та створення власного продукту.

1.4. Штучний інтелект у прогнозуванні вартості автомобілів

З попереднього пункту нам стало відомо, що для прогнозування вартості автомобіля використовують методи машинного навчання. Розкриємо це питання більш детально.

Перш за все почнемо з того, що штучний інтелект (ШІ) - це галузь комп'ютерної науки, що займається розробкою комп'ютерних систем та алгоритмів, які демонструють інтелектуальні здібності, схожі на ті, що спостерігаються у людей. ШІ використовується для моделювання та розуміння розумових процесів, прийняття

рішень, вирішення проблем та виконання завдань, які зазвичай пов'язані з людським інтелектом.

У свою чергу машинне навчання (machine learning, ML) є підгалуззю штучного інтелекту, яка шляхом створення певних алгоритмів та моделей виявляє закономірності під час аналізу великих даних та використовує їх в подальшому для самонавчання. Головною особливістю та пріоритетним завданням машинного навчання є набуття загального інтелекту, здатності вчитися та виконувати різноманітні завдання на основі набутого досвіду. Замість простого вирішення однієї конкретної задачі, машинне навчання спрямоване на розвиток універсальних алгоритмів та моделей, які можуть застосовуватися для вирішення подібних завдань [7].

Загалом класичне навчання люблять ділити на дві категорії — з вчителем і без. Також можна зустріти їх англійські назви — Supervised і Unsupervised Learning [8]. У керованому алгоритмі машинного навчання, тобто навчанні з вчителем, відповіді або вихідні дані вже відомі та відображені для вхідних даних. Таким чином, модель навчається на великій кількості навчальних прикладів, де для кожного вхідного значення існує відповідний вихідний результат. Тренувальні дані використовуються для досягнення високої точності та підготовки моделі для обробки нових вхідних даних та прогнозування результатів. Після тренування модель стає готовою для використання на нових невідомих даних і здатною зробити прогнози або виконати класифікацію на основі набутого досвіду з тренувального процесу [8].

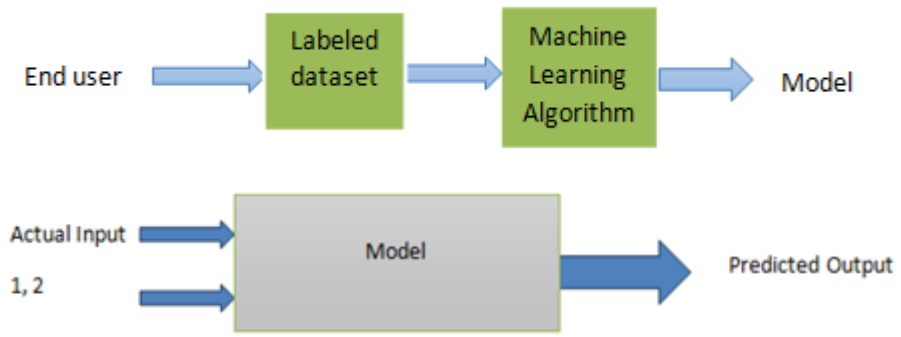

Рисунок 1.3 - Схема роботи алгоритму навчання з вчителем [8]

Слід додати, що у даній роботі буде використано саме керований алгоритм машинного навчання, навчання з вчителем.

Закінчуючи тезу про види машинного навчання, згадаємо про навчання без вчителя. Воно відбувається без допомоги керівника, подібно до того, як риба самостійно навчається плавати. Це процес, в якому модель навчається на вхідних даних без наявних відповідних вихідних міток або значень. Відсутність цільових значень у цій моделі означає, що система повинна самостійно виявляти складнощі, шаблони або групи в даних. Вона самостійно вчиться на основі введених даних та виявляє приховані закономірності, без заздалегідь визначених цілей або вихідних результатів [8].

Якщо повернутися до теми автомобілів, то використання штучного інтелекту у прогнозуванні вартості автомобілів має велике значення. Завдяки своїй здатності аналізувати велику кількості даних та виявляти складні взаємозв'язки, штучний інтелект (ШІ) може забезпечувати прогнози вартості автомобілів з вищою точністю та надійністю порівняно з традиційними методами прогнозування. Також слід додати, що штучний інтелект здатний враховувати індивідуальні характеристики автомобілів, такі як марка, модель, рік випуску, стан та інші, а також зовнішні фактори, включаючи ринкові тенденції та попит. Дана властивість дозволяє надавати користувачам більш персоналізовані та точні прогнози щодо вартості автомобіля. Постійне вдосконалення є важливою особливістю ШІ. Він може навчатися на основі нових даних та отримувати з них нові уявлення та знання. Коли штучний інтелект використовується для прогнозування вартості автомобілів, він може аналізувати реальні дані про ціни на автомобілі, тенденції ринку та інші впливові фактори. Це дозволяє моделі постійно оновлюватися та покращувати свої прогнози з часом.

Отже, з усього вищепереліченого можна дійти висновку, що вклад штучного інтелекту у розвиток автомобільної торгівлі є значним та відповідає всім викликам сьогодення.

1.5. Постановка задачі

Дана робота полягає в розробці та навчанні моделі машинного навчання, яка здатна ефективно прогнозувати вартість автомобіля на основі набору характеристик. Вхідні дані для моделі включають такі характеристики автомобіля, як назва автомобіля, місцезнаходження, рік випуску, пробіг, тип палива, тип коробки передач, кількість власників, витрати палива(економічність), об'єм двигуна, потужність двигуна, кількість місць, ціна.

Програмний продукт у представленій роботі буде розроблений на основі методів машинного навчання (навчання з учителем). Для його реалізації буде використано такі алгоритми машинного навчання, як лінійна регресія (Linear Regression) та регресор випадкового лісу (Random Forest Regressor). Розробка буде проходити від базового алгоритму лінійної регресії до кращого алгоритму Random Forest Regressor. У ході роботи буде продемонстровано, що моделі Linear Regression та Random Forest доволі гарно себе показують у задачах регресії.

Для створення програмного продукту буде використано набір даних, який знаходиться у вільному доступі в мережі Інтернет, автомобільного ринку Індії. Це пов'язано з тим, що у 2019 році Індія відзначилася як четвертий найбільший автомобільний ринок у світі за обсягом продажів, випередивши Німеччину. Тому задачі прогнозування вартості для нього є досить актуальними та важливими з огляду на масштабність та темпи росту.

У рамках подальшого розвитку та покращення програми, планується інтеграція та робота з даними продажів автомобілі по Європі та США. Окрему увагу також у майбутньому можна буде приділити автомобільному ринку України.

1.6. Висновки до розділу 1

Прогнозування вартості автомобілів відіграє ключову роль у сучасному автомобільному ринку, а його важливість і актуальність не можна недооцінювати. Прогнози вартості автомобілів дозволяють ефективно управляти ризиками, зробити обґрунтовані рішення щодо купівлі або продажу автомобіля та забезпечити раціональне планування фінансових інвестицій у даному секторі ринку.

Дослідження даної теми дозволить зрозуміти, які фактори впливають на ціну, такі як марка, модель, рік випуску, технічний стан, пробіг та інші характеристики. Це допоможить споживачам, автомобільним компаніям, дилерам та страховим компаніям зробити обґрунтовані рішення щодо подальших своїх дій з автомобілями.

Застосування штучного інтелекту у прогнозуванні вартості автомобілів дозволяє отримати більш точні та персоналізовані прогнози, які були основані на аналізі великих обсягів даних. Загалом це дозволить зробити кращі прогнози за рахунок методів машинного навчання, що забезпечить більшу надійність та ефективність в процесі прийняття рішень.

Постійне та невпинне зростання автомобільного ринку, зміна трендів споживання та нові технологічні інновації роблять прогнозування вартості автомобілів ще більш актуальним завданням. Відтак, компанії, споживачі та інші учасники ринку можуть скористатися перевагами прогнозування вартості автомобілів для покращення своєї діяльності та прийняття стратегічних рішень.

РОЗДІЛ 2 МЕТОДИ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ПРОГНОЗУВАННЯ ЦІНИ АВТОМОБІЛІВ

2.1 Основні поняття зі штучного інтелекту

Штучний інтелект (ШІ, англ. AI) є технологією, що відтворює та наслідує людські когнітивні функції. Це означає, що він здатний збирати, адаптувати та використовувати інформацію для навчання, прийняття рішень та формування висновків подібно до людини.

В контексті штучного інтелекту використовуються такі ключові терміни:

- Big Data: ШІ вимагає великих масивів даних для навчання та вдосконалення.
- Машинне навчання: Це підвідділ штучного інтелекту, який надає комп'ютерам можливість виконувати завдання, вивчаючи та аналізуючи дані, замість простого слідування заданим інструкціям.
- Глибоке навчання: Це розширення машинного навчання, яке використовує нейронні мережі для обробки і вивчення великої кількості неструктурованих даних, таких як текст, звук та відео.
- Обробка природної мови: Це важливий елемент штучного інтелекту, який спрямований на розбір тексту, аудіо- та відеофайлів, з метою вивчення та імітації природного людського мовлення.
- Нейронна мережа: Це система обчислень, яка навчається за допомогою аналізу прикладів та поступового вдосконалення своїх здібностей. Функціонування нейронної мережі базується на принципах роботи центральної нервової системи.

Модель штучного інтелекту - це система, що імітує розум, поведінку або інтелект людини. Ці системи можуть навчатися з даних, використовувати розуміння мови, розв'язувати проблеми, розпізнавати зображення, голос та інше. Вони здатні до адаптації, самонавчання та прийняття рішень.

Моделі AI можуть бути засновані на різних підходах, включаючи машинне навчання (ML), глибоке навчання (DL), посилене навчання (RL) та інше.

Зазвичай моделі розробляються для конкретних завдань та мають великий потенціал для автоматизації та покращення різноманітних процесів у багатьох областях, включаючи здоров'я, бізнес, освіту, транспорт та інше.

2.2. Метод навчання моделей

Метод навчання полягає у використанні навчальної вибірки для створення алгоритму, який відповідає заданій моделі. Тобто, алгоритм формується на основі доступних навчальних даних.

Загалом процес навчання моделей включає декілька етапів, щоб досягти оптимальної пристосованості моделі до даних:

1. Підготовка даних включає збір, очищення та підготовку даних для подальшого аналізу. Включається обробка відсутніх значень, видалення випадкових аномалій, нормалізація або стандартизація даних.
2. Обираємо моделі, які найкраще підходять для конкретної задачі прогнозування.
3. На етапі навчання моделі використовується навчальна вибірка (training data), на основі якої здійснюється оптимізація параметрів моделі. Навчальна вибірка є таким набором даних, на якій модель навчається та виявляє залежності між вхідними та вихідними змінними. Модель пристосовується до навчальних даних шляхом пошуку оптимальних значень параметрів, що забезпечують найкращу точність на навчальній вибірці. Після оптимізації параметрів, модель може бути застосована для прогнозування на нових даних.
4. Останній етап полягає в тестуванні моделі на незалежних тестових даних (test data), які модель раніше не бачила. Це дозволяє оцінити, наскільки

добре модель може прогнозувати на нових, реальних даних. Використовуються, наприклад, метрики MAE, MSE, RMSE та R^2 для оцінки якості моделі.

Метод навчання моделей є доволі важливим кроком у використанні машинного навчання для прогнозування вартості автомобілів. Він дозволяє моделі використовувати доступні дані для навчання, виявлення закономірностей та здійснення прогнозів на нових даних.

Для визначення кращих алгоритмів, які буде використано для навчання моделей, спочатку проведемо нижче огляд деяких основних методів машинного навчання та визначемо ті, які краще підходять для вирішення задач регресії.

2.3. Лінійна регресія

На сьогоднішній день алгоритм лінійної регресії є доволі популярним методом в області статистики та машинного навчання. Вона використовується для прогнозування зв'язку між двома змінними. Даний метод передбачає лінійний зв'язок між незалежною змінною та залежною змінною і має на меті знайти лінію, яка найкраще описує цей зв'язок. Лінія визначається шляхом мінімізації суми квадратів різниць між прогнозованими та фактичними значеннями.

Загалом лінійну регресію можна представити у вигляді рівняння:

$$Y(X) = P_0 + P_1 * X \qquad (2.1)$$

де Y - вихідна змінна. Змінна у представляє безперервне значення, яке намагається передбачити модель;

X - вхідна змінна. Змінна X представляє вхідну інформацію, надану моделі в будь-який момент часу;

P_0 - відрізок осі Y (член зміщення по осі OY);

P_1 - коефіцієнт регресії або масштабний фактор. У класичній статистиці P_1 є еквівалентом нахилу найкращої прямої лінії моделі лінійної регресії;

P_i - ваги (загалом).

Таким чином, регресійне моделювання полягає в пошуку значень для невідомих параметрів рівняння, тобто значень для P_0 і P_1 (ваги).

Рівняння для лінійної регресії можна візуалізувати так:

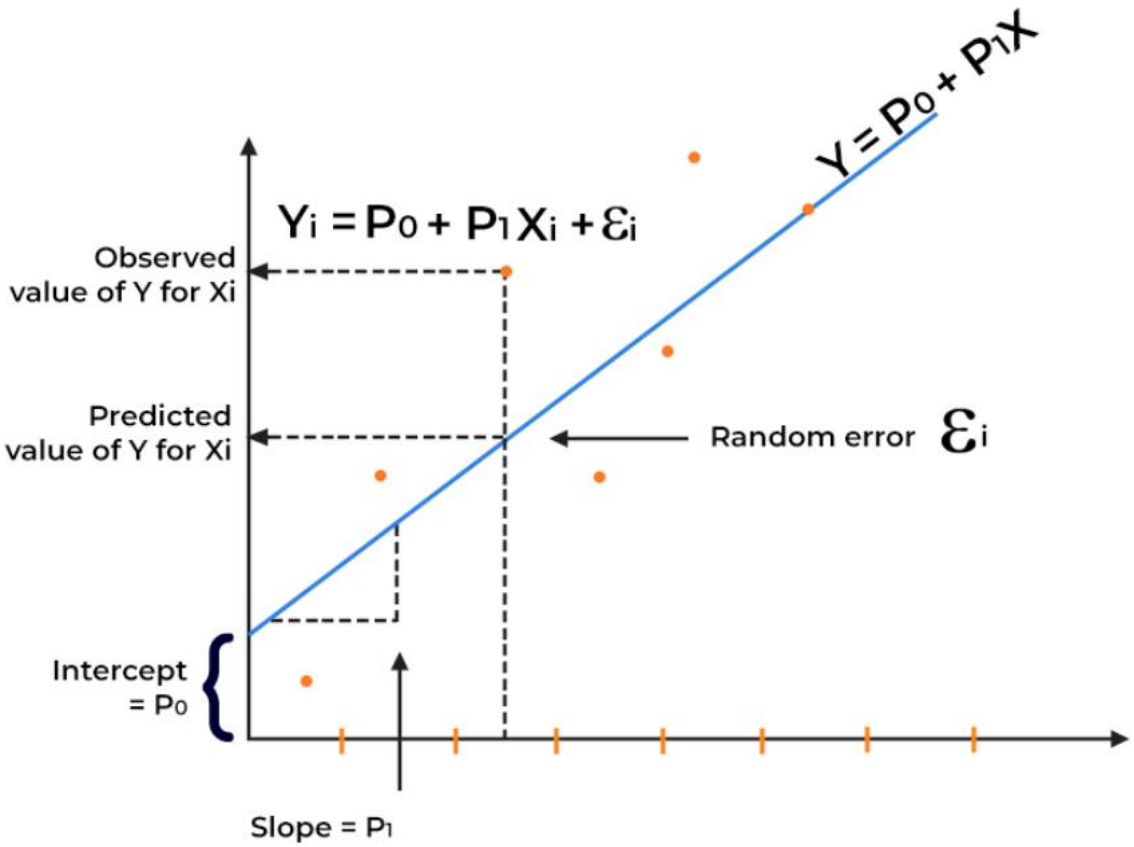

Рисунок 2.1 - Візуалізація рівняння лінійної регресії [9]

Також слід додати, що регресійну модель можна використовувати для кількох функцій, розширивши рівняння для кількості змінних, доступних у наборі даних. Рівняння множинної лінійної регресії виглядатиме так:

$$Y(X) = P_0 + P_1 * X_1 + P_2 * X_2 + \ldots + P_n * X_n \quad (2.2.)$$

Модель машинного навчання використовує згадану вище формулу та різноманітні вагові коефіцієнти для створення ліній відображення, залежно від даних. Для виявлення найбільш підходящої лінії модель оцінює різні комбінації ваг, що найкраще корелюють з даними, тим самим встановлюючи міцний зв'язок між змінними.

Для оцінки якості регресійної моделі використовуються різні метрики, наприклад середньоквадратична помилка (MSE).

2.4. Поліноміальна регресія

Поліноміальна регресія є варіантом регресійного аналізу, який використовується для моделювання зв'язку між незалежними та залежними змінними у нелінійному контексті. Хоча це і лінійний підхід, він дозволяє нам вписати нелінійну криву до наших даних.

Розглянемо ситуацію, коли у нас є набір даних, точки якого розташовані нелінійно. В таких умовах, лінійна регресія може бути неефективною, оскільки вона не зможе адекватно представити такі дані. Завдяки поліноміальній регресії, ми можемо описати таку нелінійність.

У поліноміальній регресії, вихідні змінні (ознаки) переформулюються до поліноміальних змінних відповідного порядку, після чого вони обробляються за допомогою лінійного моделювання. Такий підхід дозволяє нам максимально точно відобразити наші дані за допомогою поліноміальної лінії.

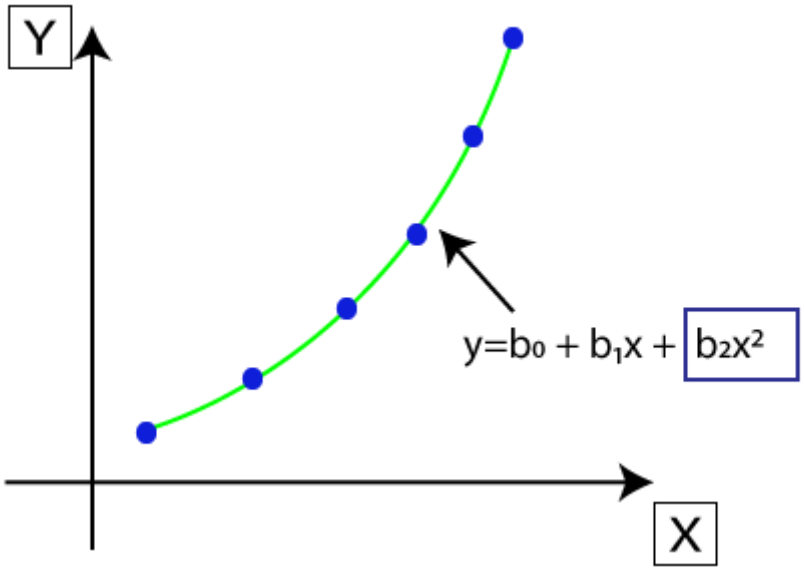

Рисунок 2.2 - Візуалізація рівняння поліноміальної регресії [10]

Рівняння для поліноміальної регресії буде мати наступний вигляд:

$$Y(X) = P_0 + b_1 * X_1 + b_2 * X_2^2 + \ldots + b_n * X_n^n \qquad (2.3)$$

де Y — прогнозований/цільовий вихід**;**

b_0 , b_1 ,... b_n — коефіцієнти регресії;

X — наша незалежна/вхідна змінна.

Поліноміальна регресія відрізняється від множинної лінійної регресії тим, що вона включає змінну, що представлена у різних ступенях, на відміну від множинної лінійної регресії, де кожна змінна представлена лише в одному ступені.

2.5. Дерево рішень

Дерево рішень (Decision Tree) - це алгоритм машинного навчання з вчителем, який може бути використаний як для задач класифікації, так і для задач регресії.

Воно представляє собою деревоподібну структуру, де внутрішні вузли представляють характеристики набору даних, гілки відображають правила прийняття рішень, а листові вузли представляють результат.

Дерева рішень можуть бути застосовані до наборів даних, що містять як числові, так і категоріальні атрибути. Вони добре працюють з нелінійними залежностями між ознаками та цільовою змінною. Однією з привабливих особливостей дерев рішень є їхня інтуїтивна зрозумілість, оскільки вони відповідають способу, яким люди приймають рішення.

В дереві рішень є два типи вузлів: вузли рішень і листові вузли. Вузли рішень використовуються для прийняття рішень і мають декілька гілок, в той час як листові вузли є кінцевим результатом цих рішень і не містять гілок.

Побудова дерева рішень зазвичай виконується за допомогою алгоритму CART (Classification and Regression Tree), який поєднує в собі як класифікацію, так і регресію. Алгоритм CART створює запитання, на які можна відповісти "Так" або "Ні", і на основі відповідей розбиває дерево на піддерева.

Нижче наведена схема, яка пояснює загальну структуру дерева рішень:

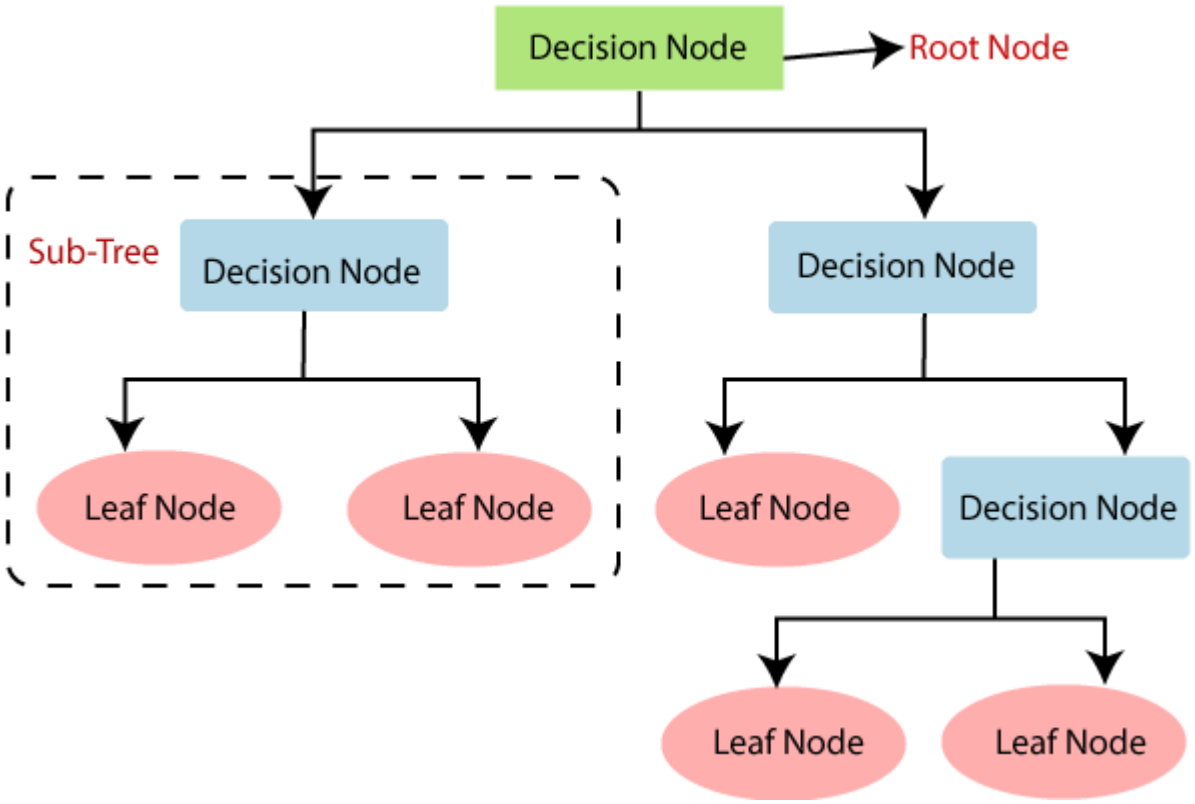

Рисунок 2.3 - Схема роботи алгоритму дерево рішень [11]

Загалом дерево рішень - це структура, в якій кожен вузол відображає характеристику, гілки представляють рішення, а листя - кінцеві результати (числові значення для регресії).

Щодо переваг, то дерева рішень легко зрозуміти та інтерпретувати, оскільки вони надають зрозумілу візуальну модель, що дозволяє інтуїтивно розуміти, як приймаються рішення на основі характеристик даних. Також існує можливість працювати як з числовими, так і з категоріальними ознаками без потреби в складних перетвореннях або кодуванні даних. Дерева рішень можуть працювати безпосередньо з оригінальними даними.

Якщо розглянути недоліки, то дерева рішень мають тенденцію до створення складних моделей, які можуть добре відповідати тренувальним даним, але погано узагальнюватися на нові дані. Це називається проблемою перенавчання, коли дерево стає занадто специфічним для тренувального набору даних і втрачає здатність до загальної класифікації. Також не слід забувати, що навіть невеликі зміни в даних можуть призводити до значних змін у структурі дерева рішень. Це може зробити модель нестабільною, особливо якщо вхідні дані мають шум або незначні зміни.

2.6. Випадковий ліс

Випадковий ліс є відомим і ефективним алгоритмом машинного навчання, що належить до категорії навчання з вчителем. Цей алгоритм може бути використаний для вирішення завдань класифікації або регресії в контексті машинного навчання. Його основа - принцип ансамблю, який включає комбінацію декількох класифікаторів для розв'язання складних завдань і підвищення ефективності моделі.

Випадковий ліс - це вид класифікатора, що включає в себе декілька дерев рішень, кожне з яких працює на своїй підмножині даних, і використовує середнє значення для покращення точності прогнозування. Таким чином, замість залежності

від одного дерева рішень, випадковий ліс бере прогнози з кожного дерева, враховуючи більшість голосів прогнозів, і на цій основі робить кінцеве передбачення.

Зазвичай, більша кількість дерев у лісі веде до вищої точності передбачення. Крім того, додавання дерев до лісу допомагає уникнути перенавчанню.

Нижче наведена схема, яка описує роботу алгоритму випадкового лісу:

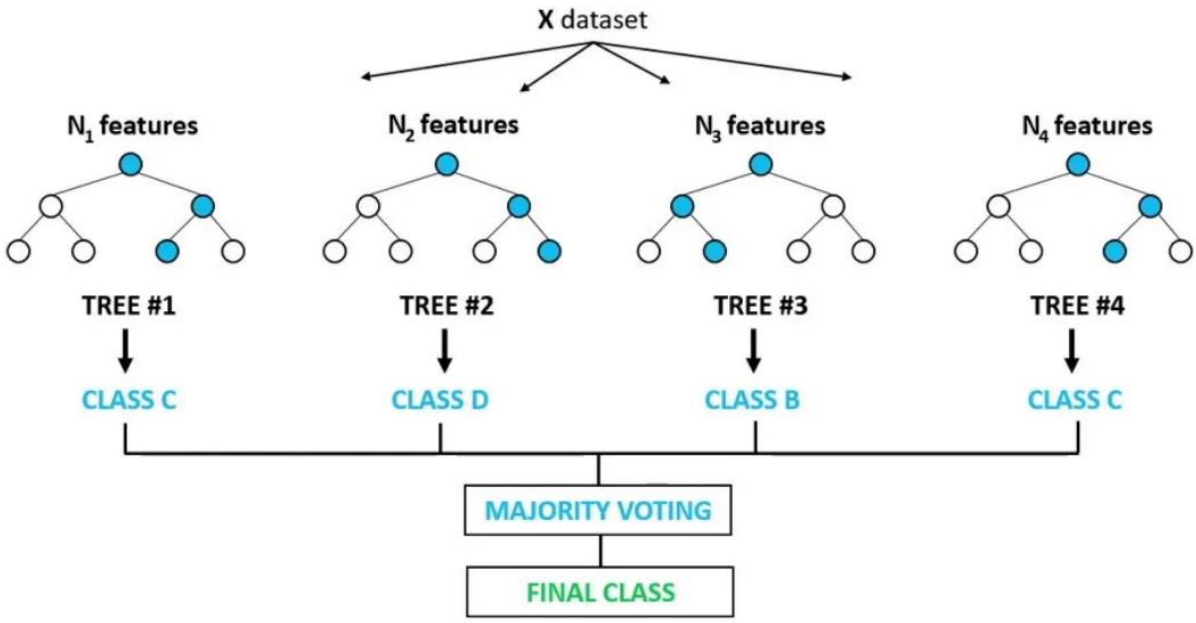

Рисунок 2.4 - Схема роботи алгоритму випадкового лісу [12]

Алгоритм випадкового лісу діє у два основні етапи:

• На першому етапі відбувається формування випадкового лісу шляхом об'єднання N дерев рішень.

• На другому етапі здійснюється створення прогнозів для кожного з побудованих де рев.

Цей процес можна описати наступними кроками:

Крок 1: Відбираємо випадковим чином K точок даних з навчального набору.

Крок 2: Створюємо дерева рішень на основі вибраних точок даних (підмножин).

Крок 3: Визначаємо число N дерев рішень, які ми бажаємо побудувати.

Крок 4: Повторюємо кроки 1 і 2.

Крок 5: Для нових точок даних робимо прогнози від кожного дерева рішень і віднос имо нові точки даних до тієї категорії, яка отримала більшість голосів.

Варто відзначити, що випадковий ліс використовує декілька дерев рішень для прогнозування класу даних. Таким чином, можливо, що окремі дерева рішень дають правильний прогноз, а деякі ні. Однак у загальному контексті всі дерева разом передбачають правильний вихід.

Також потрібно звернути увагу, що використання великих випадкових лісів з метою підвищення продуктивності може не бути оптимальним, оскільки це вимагає значних обчислювальних ресурсів, включаючи пам'ять та обчислювальний час.

2.7. **Регресія опорних векторів**

Регресія опорних векторів (SVR) - це метод регресії, призначений для роботи з безперервними величинами. Цей підхід використовує кілька важливих концепцій, які варто відзначити:

- Ядро: це функція, яка дозволяє нам перетворити дані з нижчої вимірності до вищої вимірності, що полегшує обробку складних даних.
- Гіперплощина: У контексті машини опорних векторів (SVM), це лінія, що відокремлює два класи. Однак у випадку SVR, це лінія, яка служить для прогнозування безперервних змінних, при цьому намагаючись включити якомога більше точок даних.
- Межі: Межі є двома лініями, які, на відміну від гіперплощини, створюють коридор для точок даних.
- Опорні вектори: це дані, що знаходяться найближче до гіперплощини і протилежного класу.

Мета SVR - визначити гіперплощину з максимальною відстанню до ближніх точок даних (запасом), щоб якомога більше точок було включено в цей запас.

Основний принцип SVR - забезпечити, що найбільша кількість точок даних знаходиться між межами, і гіперплощина (або лінія оптимального підходу) має найбільшу кількість точок даних.

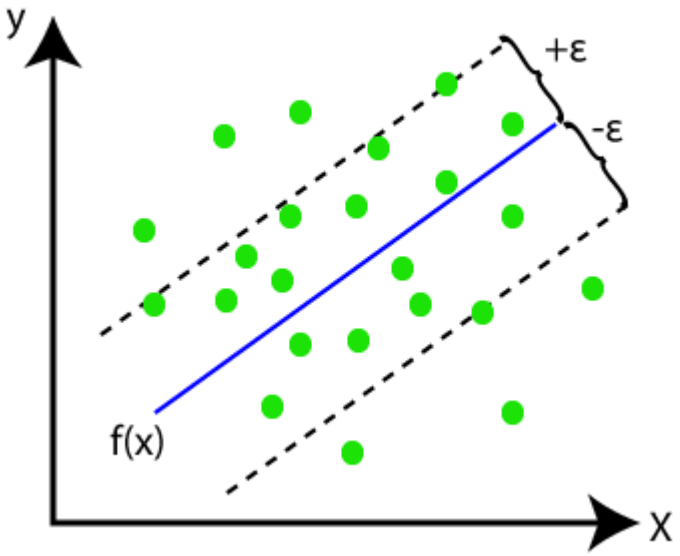

Рисунок 2.5 - Схема роботи алгоритму регресії опорних векторів [13]

На рисунку 2.5 лінія, яка зображена синім кольором, представляє гіперплощину, тоді як дві інші лінії відіграють роль обмежувальних ліній.

2.8. Метрики якості моделей

Метрики якості моделі машинного навчання відіграють важливу роль в оцінці ефективності моделі. Ці метрики визначають, наскільки добре модель впоралася з задачею прогнозування і допомагають визначити, чи вдалося досягнути бажаного результату. Опишемо декілька причин, чому метрики якості моделі є доволі необхідними:

- Відображення точності: Метрики якості допомагають визначити, наскільки точно модель передбачає результати. Це важливо для визначення, наскільки

надійна модель, і чи може вона бути використана для прогнозування реальних даних.

- Порівняння моделей: Метрики допомагають в порівнянні різних моделей. Ви можете тренувати різні моделі на одному і тому ж наборі даних і використовувати метрики якості, щоб визначити, яка з них виконує свою роботу найкраще.
- Оптимізація моделі: Використовуючи метрики якості, ви можете оптимізувати свою модель, роблячи вдосконалення і тонку настройку параметрів моделі, щоб покращити її продуктивність.
- Діагностика проблем: Метрики також можуть вказати на потенційні проблеми у моделі, такі як перенавчання, де модель занадто добре працює на тренувальних даних, але погано справляється з новими, невидимими даними.
- Повторюваність: Метрики якості забезпечують стандартні, повторювані міри для оцінки моделей, що допомагає при порівнянні результатів різних дослідників або різних підходів до розв'язання задачі.

Нижче проведемо огляд основних метрик якості моделі, які використовують при вирішенні задач регресії.

2.8.1 Середня абсолютна похибка (MAE)

Середня абсолютна помилка (MAE) — це простий показник, що вимірює різницю між реальними та прогнозованими значеннями.

Наприклад, у вас є набір вхідних та вихідних даних, і використовується модель лінійної регресії для зображення оптимальної лінії прогнозування. Щоб визначити MAE для даної моделі, необхідно врахувати величину помилок, які модель зробила, відомі як відхилення. Тобто, потрібно визначити різницю між реальними та прогнозованими значеннями, яка представляє абсолютну помилку.

Але для отримання MAE необхідно обчислити середнє абсолютне відхилення для всього набору даних.

Щоб це зробити, треба просумувати всі відхилення та поділити їх на загальну кількість спостережень. Результатом буде отримана помилка MAE. Ціль полягає в мінімізації MAE, оскільки воно представляє загальну втрату моделі. Формула даної метрики представлена нижче:

$$MAE = \frac{1}{n}\sum\left|Y - \hat{Y}\right| \quad (2.4)$$

де Y – фактичні значення;

$\hat{Y}$ - прогнозовані значення.

Переваги середньої абсолютної помилки (MAE):

- Одиниці MAE співпадають з одиницями змінної, яку прогнозують. Це робить MAE інтуїтивно зрозумілим.
- MAE є величиною, стійкою до аномальних значень у даних.

Недоліки середньої абсолютної помилки (MAE):

- Функція MAE не є диференційованою, тому для оптимізації можуть бути потрібні інші методи, такі як градієнтний спуск, який вимагає диференційованих функцій.

2.8.2 Середня квадратична помилка (MSE)

Середня квадратична помилка (MSE) - це часто використовувана метрика, яка має багато схожостей з середньою абсолютною помилкою, але з невеликою

модифікацією. Вона обчислює квадрат різниці між реальними і прогнозованими значеннями.

У порівнянні з MAE, де ми обчислювали абсолютну різницю, в MSE ми вимірюємо квадрат цієї різниці.

Таким чином, MSE фактично представляє собою квадрат відстані між дійсними та передбаченими значеннями. Робити цей квадрат необхідно, щоб уникнути видалення від'ємних значень, що є однією з переваг MSE. Формула даної метрики представлена нижче:

$$MSE = \frac{1}{n}\sum(Y - \hat{Y})^2 \qquad (2.5)$$

де Y – фактичні значення;

$\hat{Y}$ - прогнозовані значення.

Переваги середньої квадратичної помилки (MSE):

- Графік MSE можна диференціювати, що робить його підходящим для використання як функції втрат.

Недоліки середньої квадратичної помилки (MSE):

- Результат, який отримано після розрахунку MSE, вимірюється у квадраті вихідних одиниць. Наприклад, якщо вихідна змінна вимірюється у метрах, то значення MSE виражається в квадратних метрах.
- MSE дуже чутливий до викидів в даних, і ці викиди можуть призвести до великого зростання значення MSE. Тобто, MSE не стійкий до викидів, на відміну від MAE.

2.8.3 Середньоквадратична помилка (RMSE)

RMSE, як можна зрозуміти з назви, представляє собою квадратний корінь від середньоквадратичної помилки. Формула даної метрики представлена нижче:

$$RMSE = \sqrt{\frac{1}{n}\sum(Y - \hat{Y})^2} \quad (2.6)$$

де Y – фактичні значення;

$\hat{Y}$ - прогнозовані значення.

Переваги середньоквадратичної помилки (RMSE):

- Виходом RMSE є значення, яке має ту саму масштабувальну одиницю, що й вихідна змінна, що спрощує розуміння рівня помилок моделі.

Недоліки середньоквадратичної помилки (RMSE):

- RMSE менш стійкий до аномалій у порівнянні з MAE.

2.8.4 Коєфіцієнт детермінації (R^2)

Це статистична міра, яка показує, наскільки передбачувані дані підходять до реальних данних. Значення R^2 варіюються від 0 до 1, де 1 означає, що модель ідеально передбачає реальні дані.

R^2 обчислює, наскільки лінія регресії має бути кращою за середню лінію. Формула даної метрики представлена нижче:

$$R^2 = 1 - \frac{\sum(Y - \hat{Y})^2}{\sum(Y - \bar{Y})^2} \quad (2.7)$$

де Y – фактичні значення;

$\hat{Y}$ - прогнозовані значення;

$\bar{Y}$ - середні значення Y.

Переваги коєфіцієнту детермінації R^2:

- Стандартна метрика: R^2 є загальноприйнятим коефіцієнтом, що оцінює якість моделі у статистиці та машинному навчанні.
- Інтерпретованість: R^2 виражає долю варіації залежної змінної, яка пояснюється моделлю. Це робить R2 вкрай інтерпретованим, що є важливим для розуміння та обґрунтування результатів моделі.
- Масштабність: R^2 має фіксовані межі від 0 до 1 (або від 0% до 100%), що дозволяє легко порівнювати результати різних моделей.

Недоліки коефіцієнту детермінації R^2:

- Не завжди відображає точність прогнозів: Хоча R2 оцінює якість прогнозування моделі, він може бути високим у моделі, яка фактично не добре прогнозує дані. Це особливо відноситься до випадків з високою кількістю змінних.
- Вплив викидів: R2 може сильно змінюватися через викиди (аномальні дані), що може призвести до переоцінки або недооцінки якості моделі.
- Часто використовується неправильно: Багато людей помилково використовують R2 як єдиний показник якості моделі, хоча він має використовуватися разом з іншими метриками для повного розуміння результатів моделі.

2.9. Висновки до розділу 2

У даному розділі було розглянуто різні методи штучного інтелекту, що використовуються для прогнозування вартості автомобілів. Зокрема, описали такі алгоритми машинного навчання, як лінійна регресія, поліноміальна регресія, дерево рішень, випадковий ліс та регресія опорних векторів.

Лінійна регресія (Linear Regression) - це базовий алгоритм, що використовує лінійний підхід до моделювання взаємозв'язку між однією або кількома незалежними змінними та цільовою змінною. Вона проста у розумінні та інтерпретації, але може бути недостатньою при наявності нелінійних взаємозв'язків або високої розмірності даних.

Поліноміальна регресія (Polynomial Regression) використовує поліноміальні функції для апроксимації залежностей між залежною і незалежними змінними. Цей метод дозволяє моделювати нелінійні зв'язки та допомагає уникнути простої лінійної апроксимації, що не враховує складнішу структуру даних. Поліноміальна регресія може бути корисною для вирішення задач, де залежність між змінними має криволінійний або неоднорідний характер.

Дерево рішень (Decision Tree) - це модель машинного навчання, що використовує деревоподібну структуру для прийняття рішень на основі характеристик даних. Кожен вузол в дереві представляє питання або правило прийняття рішення, гілки відповідають можливим варіантам відповіді, а листові вузли містять результат або прогноз. Дерева рішень є інтуїтивно зрозумілими, легко інтерпретованими і можуть бути застосовані як для задач класифікації, так і для задач регресії.

Випадковий ліс (Random Forest) - це алгоритм, що базується на ансамблевому методі, що використовує багато дерев рішень для створення більш точної моделі. Він здатний впоратися з нелінійними даними і автоматично займається важливими

факторами, але може бути важчим для інтерпретації і вимагає більше часу для навчання.

Регресія опорних векторів (SVR) - це метод машинного навчання, який використовується для розв'язання задач регресії. В SVR головною метою є пошук оптимальної гіперплощини, яка найкраще апроксимує дані, з урахуванням допустимої помилки. Відмінністю SVR від класичного методу опорних векторів (SVM) є те, що в регресії опорні вектори розташовані у межах прогнозованого значення з допуском певної помилки, замість строгої класифікації на дві категорії.

У ході аналізу переваг та недоліків представлених вище алгоритмів було обрано, що лінійна регресія та випадковий ліс найкраще підходять для вирішення задачі прогнозування вартості автомобілів. Лінійна регресія є простою та ефективною моделлю, яка добре працює в випадках, коли залежність між змінними є лінійною або близькою до лінійної. Вона має низький ризик перенавчання і добре інтерпретується, дозволяючи визначити вплив кожної змінної на результат. У той час, як випадковий ліс є ансамблем дерев рішень, що комбінує декілька моделей для отримання кращої точності та стабільності прогнозу. Він може працювати з різними типами даних, не вимагає передпроцесінгу та впорається з нелінійними залежностями між змінними. Випадковий ліс також може оцінювати важливість змінних і використовувати багатокласову класифікацію.

Для оцінки якості моделей були обрані метрики MAE, RMSE та R^2. Метрика MAE вимірює середню абсолютну помилку між прогнозованими значеннями і справжніми значеннями. Вона дозволяє отримати інформацію про величину помилок в прогнозах без врахування їхнього напрямку. Чим менше значення MAE, тим краще модель. RMSE є популярною метрикою, яка враховує як величину, так і напрямок помилок прогнозу. Вона обчислює квадратний корінь з середньоквадратичної помилки. RMSE штрафує великі помилки сильніше, ніж MAE, що робить його чуттєвішим до великих відхилень в прогнозах. R2 вимірює відповідність моделі до варіації вихідних даних. Чим ближче значення R2 до 1, тим краще модель. Він вказує на частку варіації, пояснену моделлю, в порівнянні з

загальною варіацією даних. Використання R2 дозволяє оцінити, наскільки добре модель пояснює варіацію вихідних даних.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Обґрунтування вибору мови програмування

Вибір мови програмування є доволі важливим для успіху будь-якого проекту пов’язаного з розробкою програмного забезпечення, оскільки саме мова програмування безпосередньо впливає на швидкість розробки, доступність та якість підтримки сторонніх бібліотек, а також потенційну масштабованість і стійкість кінцевого продукту. Крім того, вибір мови також впливає на вартість проекту, оскільки різні мови мають різні рівні доступності ресурсів та вимог до обслуговування.

Під час аналізу існуючих мов програмування, було обрано, що мова програмування Python найбільше підходить для вирішення поставленої задачі.

Python - це високорівнева, інтерпретована мова програмування з динамічною типізацією та автоматичним управлінням пам'яттю. Вона була створена Гвідо ван Россумом і вперше випущена в 1991 році. Python підтримує кілька парадигм програмування, включаючи процедурне, об'єктно-орієнтоване та функціональне програмування [1].

Мова програмування Python є однією з найбільш уживаних для науки про дані та машинного навчання. Він має різноманітні бібліотеки, що впроваджують різні алгоритми машинного навчання та методи аналізу даних. Наприклад, pandas для обробки даних, NumPy для числових операцій, Scikit-learn для впровадження і навчання моделей машинного навчання, matplotlib та Seaborn для візуалізації даних, тощо.

Також варто згадати про те, що Python відомий своїм чистим та читабельним синтаксисом, що робить його легким для навчання та користування, особливо для новачків. Окрім цього, він має велику та активну спільноту розробників, яка надає велику кількість ресурсів для навчання та підтримки. Це включає в себе численні

онлайн-курси, книги, форуми та онлайн-групи, де можна отримати відповіді на питання та рішення проблем.

Звернімо увагу, що Python легко інтегрується з іншими системами та сервісами, що можуть бути корисними для збору, обробки та візуалізації даних. Загалом дана мова програмування дозволяє швидко розробляти та модифікувати код. За рахунок свого синтаксису та багатофункціональності, він ідеально підходить для використання в наукових дослідженнях та дипломних роботах, де потреби можуть швидко змінюватися або розвиватися.

Оскільки дана робота передбачає роботу з даними, то Python має велику стандартну бібліотеку, яка підтримує багато загальних задач програмування, що включають модулі для роботи з файлами, JSON, HTTP, датами та часом, CSV файлами та багато іншого.

Підсумовуючи все згадане вище, можна дійти висновку, що Python - це надзвичайно потужна, гнучка і доступна мова програмування, яка має широке застосування у різних сферах, починаючи від веб-розробки і закінчуючи науковими дослідженнями та аналітикою даних. Її простота синтаксису і читабельність коду робить її прекрасним вибором для новачків, а потужні вбудовані функції та обширна бібліотека модулів роблять її незамінним інструментом для досвідчених розробників. Python продовжує активно розвиватися і має сильну підтримку спільноти, що робить його вибором номер один для багатьох проектів і організацій.
Саме тому було прийнято рішення, що розробку програмного продукту з прогнозування вартості автомобілів необхідно проводити на мові програмування Python.

3.2. Опис використаних бібліотек

Виходячи з того, що у рамках представленої роботи необхідно провести прогнозування вартості, а також графічно зобразити результати, то розглянемо наступні бібліотеки мови програмування Python. Загалом їх декілька, які спрямовані на обробку даних та машинне навчання. Нижче розглянуто найбільш поширені з них.

Scikit-learn (Sklearn) - це популярна та надійна бібліотека для машинного навчання в мові Python. Вона надає багатий набір інструментів для статистичного моделювання та машинного навчання, включаючи техніки класифікації, регресії, кластеризації та зменшення розмірності. Все це пропонується через єдність інтерфейсу в Python. Скомпонована в основному на Python, ця бібліотека базується на таких фундаментальних пакетах, як NumPy, SciPy і Matplotlib.

Дана бібліотека має стандартний і зрозумілий для всіх Python API, який спрощує використання та розробку. Вона легко використовується в комбінації з іншими бібліотеками Python, включаючи NumPy та Pandas для обробки даних, а також бібліотеки для візуалізації даних, такі як Matplotlib і Seaborn.

Scikit-learn використовує NumPy для створення ефективних алгоритмів на Python. Вона також може використовувати багатопоточність, якщо це можливо, щоб прискорити обчислення.

Бібліотека sklearn містить багато вбудованих функцій і засобів для попередньої обробки даних, вибору та оцінки моделей, трансформації даних, завантаження датасетів та багатого іншого, що робить її вельми гнучким інструментом для науковців та інженерів, що працюють у сфері машинного навчання.

Основними модулями Scikit-learn є:

- Sklearn.datasets: Модуль для завантаження стандартних наборів даних, таких як набір даних iris та digits.
- Sklearn.model_selection: Модуль для розділення набору даних на тестові та навчальні датасети, крос-валідації тощо.
- Sklearn.preprocessing: Модуль для попередньої обробки - масштабування, центрування, нормалізації, бінаризації та імпутації.
- Sklearn.metrics: Модуль для оцінювання точності прогнозів.
- Sklearn.pipeline: Модуль для настройки машинного навчання.
- Sklearn.linear_model: Модуль для роботи з лінійними моделями.
- Sklearn.naive_bayes: Модуль для наївних Баєсівських моделей.
- Sklearn.ensemble: Модуль для методів ансамблю.
- Sklearn.svm: Модуль для методів Support Vector Machine.
- Sklearn.neighbors: Модуль для методів найближчих сусідів.
- Sklearn.cluster: Модуль для некерованих моделей кластеризації.

Загалом усі ці модулі, включені до Scikit-learn, роблять його потужним інструментом для машинного навчання та аналізу даних. Тому було прийнято рішення обрати цю бібліотеку для роботи над дипломним проектом.

Pandas - бібліотека, що пропонує зручні структури даних та методи обробки даних для швидкого та простого аналізу. Ім'я "Pandas" походить від терміну "Panel Data", що використовується в економетриці.

Опишемо деякі ключові особливості Pandas:

- Структури даних: Pandas надає дві основні структури даних: Series (одновимірний масив) і DataFrame (двовимірний масив).
- Обробка даних: Pandas може обробляти різні формати даних, включаючи CSV, Excel, SQL і багато інших. Вона надає широкий спектр функцій для обробки даних, таких як фільтрація, сортування, заміна пропущених значень тощо.
- Аналіз даних: Pandas надає функції для проведення різних статистичних аналізів. Вона також вміє з’єднувати і змішувати датасети, використовуючи функції, подібні до SQL.

- Візуалізація даних: Хоча Pandas не є власне бібліотекою візуалізації, вона має тісний зв'язок з Matplotlib і Seaborn, що дозволяє легко візуалізувати DataFrame і Series.

Загалом, Pandas є потужним інструментом, який робить Python зручною мовою для аналізу даних, і це робить його основним компонентом для будь-якого проекту, пов'язаного з даними, включаючи прогнозування цін на автомобілі.

Matplotlib - це відкрита бібліотека для візуалізації даних на Python, що робить задачу зображення даних більш простою та ефективною. Вона була створена Джоном Д. Гантером у 2003 році і з того часу стала стандартом в області візуалізації даних в Python.

Дана бібліотека використовується для створення статичних, інтерактивних та анімованих візуалізацій у Python. Вона має широкий спектр інструментів для візуалізації, що охоплює від простих діаграм розсіювання до складних тривимірних діаграм.

Однією з найважливіших особливостей Matplotlib є її можливість працювати в штатному режимі з багатьма операційними системами та графічними бекендами. Також вона включає підтримку широкого спектра діаграм, включаючи лінійні діаграми, стовпчасті діаграми, гістограми, діаграми розсіювання, кругові діаграми та ще багато іншого. Крім того, бібліотека надає API для вбудовування цих діаграм у GUI, такі як Tk, wxPython, Qt або GTK.

Окремо варто відзначити, що Matplotlib повністю інтегрована з бібліотеками NumPy та Pandas, що робить її незамінним інструментом для наукового програмування в Python.

Додатково звернемо увагу, що Matplotlib має активну спільноту розробників і користувачів, які підтримують велику кількість ресурсів для навчання і допомоги у вирішенні проблем. Є численні онлайн-ресурси, включаючи документацію, туторіали, приклади коду, форуми для обговорення та майстер-класи, які допомагають користувачам ефективно використовувати бібліотеку.

Отже, завдяки своїй гнучкості, потужності та широкому визнанню, Matplotlib є важливим інструментом для візуалізації даних у Python. Тому її використання при

реалізації продукту з прогнозування вартості та візуалізацією даних є обов’язковим та надзвичайно важливим.

Розглянемо також ще одну бібліотеку під назвою Seaborn. Вона є бібліотекою візуалізації даних на Python, яка надає високорівневий інтерфейс для створення красивих, інформативних статистичних графіків. Вона створена поверх Matplotlib і є частиною більш широкого екосистеми наукового програмування Python, що включає такі пакети, як NumPy, SciPy і Pandas.

Seaborn надає величезну кількість можливостей для візуалізації даних, включаючи автоматичне створення різних типів графіків, побудову складних візуалізацій, графіків та гістограм для однієї або кількох змінних, побудову матриць кореляцій і багато іншого.

Дана бібліотека також дуже зручна для роботи з Pandas, оскільки вона без проблем сприймає датафрейми Pandas і може використовувати метадані з цих структур для надання контексту графікам, які вона створює.

Бібліотека Seaborn особливо корисна для візуалізації статистичних моделей, що включає лінійні і нелинійні регресійні моделі, класифікаційні і кластеризаційні моделі, а також невід'ємною частиною процесу EDA (exploratory data analysis), який є дуже важливим етапом у роботі з даними.

Seaborn пропонує більш красивий та зрозумілий вивід графіків, порівняно з Matplotlib. Отже, для представлення даних у вигляді графіків часто використовують Seaborn, бо він дозволяє легко і просто робити графіки вищої якості.

Отже, Seaborn - це потужний і гнучкий інструмент для візуалізації даних в Python, який спрощує відображення та розуміння складних наборів даних. Тому її також було вирішено використати при розробці дипломного програмного продукту.

3.3. Обґрунтування вибору платформи реалізації програмного продукту

Вибір платформи для реалізації програмного продукту є одним з ключових рішень, яке необхідно прийняти при розробці програми. Це визначає не тільки функціональність та властивості створеного продукту, але і його подальшу підтримку, оптимізацію та загалом майбутнє. Під час перегляду та аналізу наявних платформ для роботи та прогнозування даних, було виявлено, що платформа Google Colab доволі сильно вирізняється на фоні інших. Нижче детальніше поговоримо про неї.

Google Colab (або Colaboratory) - це безкоштовне середовище Jupyter Notebook, що працює в хмарі і зберігається на Google Drive. Воно надає можливість писати та виконувати код Python в браузері з декількома великими перевагами:

- Вільний доступ до обчислювальних ресурсів: Google Colab пропонує безкоштовний доступ до обчислювальних засобів, включаючи графічні процесори (GPU) та тензорні процесори (TPU), що особливо корисно для завдань з машинного навчання.
- Хмарне середовище: Написані проекти зберігаються в хмарі, тому завжди є можливість отримати доступ до них з будь-якого комп'ютера і поділитися ними з іншими без будь-яких додаткових кроків.
- Сумісність: Colab повністю сумісний з бібліотеками Python, такими як Pandas, Scikit-learn, Matplotlib та TensorFlow. Кожний може імпортувати свої датасети, створювати візуалізації, побудовувати моделі машинного навчання та використовувати інші ресурси, які надає Python.
- Інтеграція з Google Drive і Github: Існує можливість легко завантажити проекти прямо з Google Drive або Github, а також зберігати роботу назад на ці платформи.

- Інтерактивні туторіали: Google Colab надає серію інтерактивних туторіалів, що допомагають швидко ознайомитися з середовищем та його особливостями.
- Легкість використання: Для роботи з Google Colab не потрібно жодної додаткової настройки. Немає необхідності встановлювати бібліотеки на свій персональний комп'ютер або турбуватися про версії бібліотек. Все це робить Google Colab привабливим інструментом для використання Python і машинного навчання.
- Велика продуктивність: Оскільки Google Colab використовує обчислювальну потужність Google, то із-за цього можна отримати перевагу від високої продуктивності, особливо при роботі з великими наборами даних або виконанні важких обчислень, таких як навчання моделей глибокого навчання.
- Широкий спектр використання: Від використання як інтерактивного середовища для дослідження даних до публікації робіт як інтерактивних проектів - Google Colab пропонує великий спектр можливостей для різних видів проектів.

Отже, Google Colab - це ефективний, надійний та гнучкий інструмент для розробки програмного продукту на Python, особливо в області машинного навчання та аналізу даних. Саме тому його було обрано як платформу для написання дипломного проекту.

3.4. Опис програмного продукту

У даному програмному продукті методи машинного навчання були застосовані для прогнозування цін автомобілів, використовуючи сформовану для цього вибірку даних про продані автомобілі на ринку Індії. Ці дані були взяті з платформи kaggle.com, яка є відомим ресурсом, де зібрані різноманітні набори даних з різних секторів економіки та бізнесу. На рисунку 3.1. представлена

отримана вибірка даних для використання у представленій роботі з виведеними п’ятьма випадковими рядками.

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	New_Price	Price
705	705	Hyundai Verna 1.6 VTVT	Kochi	2017	37688	Petrol	Manual	First	17.01 kmpl	1591 CC	121.3 bhp	5.00	NaN	7.88
4400	4400	Volkswagen Polo Petrol Highline 1.6L	Kolkata	2011	46000	Petrol	Manual	First	15.26 kmpl	1598 CC	103.5 bhp	5.00	NaN	2.35
4096	4096	Toyota Camry Hybrid 2.5	Coimbatore	2017	23249	Petrol	Automatic	First	19.16 kmpl	2487 CC	175.67 bhp	5.00	40.62 Lakh	26.11
329	329	Hyundai i20 1.4 CRDi Asta	Pune	2012	83000	Diesel	Manual	First	23.0 kmpl	1396 CC	90 bhp	5.00	NaN	5.50
6705	6705	Hyundai Accent GLX	Pune	2009	60000	Petrol	Manual	Second	13.2 kmpl	1495 CC	94 bhp	5.00	NaN	NaN

Рисунок 3.1 - Виведені 5 випадкових рядків початкових даних вибірки

Представлена таблиця має 7254 записи та 14 колонок. Опишемо кожний атрибут даної таблиці:

- S.No. – порядковий номер запису у таблиці;
- Name – назва автомобіля (марка та модель);
- Location – місце розташування автомобіля;
- Year – рік виробництва автомобіля;
- Kilometers_Driven – пробіг автомобіля у кілометрах;
- Fuel_Type – тип палива (бензин, дизель, газ на основі суміші пропану та бутану, газ на основі метану та електричний заряд);
- Transmission – тип коробки передач (автоматична або механічна);
- Owner_Type – кількість власників (перший, другий, третій, чотири та більше);
- Mileage – запас ходу, стандартний пробіг автомобіля, який зазвичай вимірюється в кілометрах на літр (км/л) або кілометрах на кілограм (км/кг) у випадку автомобілів, що працюють на стисненому природному газі. Є оцінкою відстані, яку автомобіль може подолати, використовуючи певну кількість палива;
- Engine – об’єм двигуна у $см^3$;
- Power – потужність двигуна, що вимірюється у “гальмівних” кінських силах (Brake horsepower). Цей показник беруть безпосередньо від двигуна, до втрати потужності, яка виникає в компонентах, таких як коробка передач, генератор і диференціал. Ось чому BHP іноді називають "потужністю автомобіля на колінвалі", тому що це потужність, що вимірюється на колінвалі двигуна;

- Seats – кількість місць у автомобілі;
- New_Price та Price – ціна на автомобіль коли він був новий та фактична ціна станом на теперішній час. Вимірюються у індійській валюті, для роботи програми вона не має суттєвого значення, тому будемо вважати що вартість автомобілів вимірюється у певних умовних одиницях.

Представлені дані мають пропуски, тому перед тим, як почати подальшу роботу, необхідно заповнити пусті місця середнім значенням даних з відповідного стовпця, в якому є пропуски. Також треба виділити чи є переважно пусті колонки. Якщо такі є, то їх належить видалити, оскільки вони не мають змісту. У даній таблиці є стовпець з більшістю пустих значень, New_Price, тому його було прибрано з неї.

Окрім цього, необхідно було провести приведення деяких даних до числової форми. Наприклад, у колонці Engine прибрати від чисел приставку CC (см3) та залишити тільки число, тобто 1799 CC перетворити на 1799. Аналогічну процедуру також було проведено з усіма іншими колонками, які мають численне та буквенне значення. Фінальний вигляд даних представлено на рисунку 3.2.

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage kmpl/kg	Engine CC	Power bph	Seats	Price
0	0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First	26.60	998.00	58.16	5.00	1.75
1	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	19.67	1582.00	126.20	5.00	12.50
2	2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First	18.20	1199.00	88.70	5.00	4.50
3	3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First	20.77	1248.00	88.76	7.00	6.00
4	4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second	15.20	1968.00	140.80	5.00	17.74

Рисунок 3.2 - Фінальний вигляд даних після проведення попередньої обробки

Після виконання робіт з підготовки даних до роботи, шлях подальшого виконання програми складається з двох напрямків, а саме аналізу даних та прогнозування цін автомобілів. Загальна схема роботи програмного продукту зображена на рисунку 3.3.

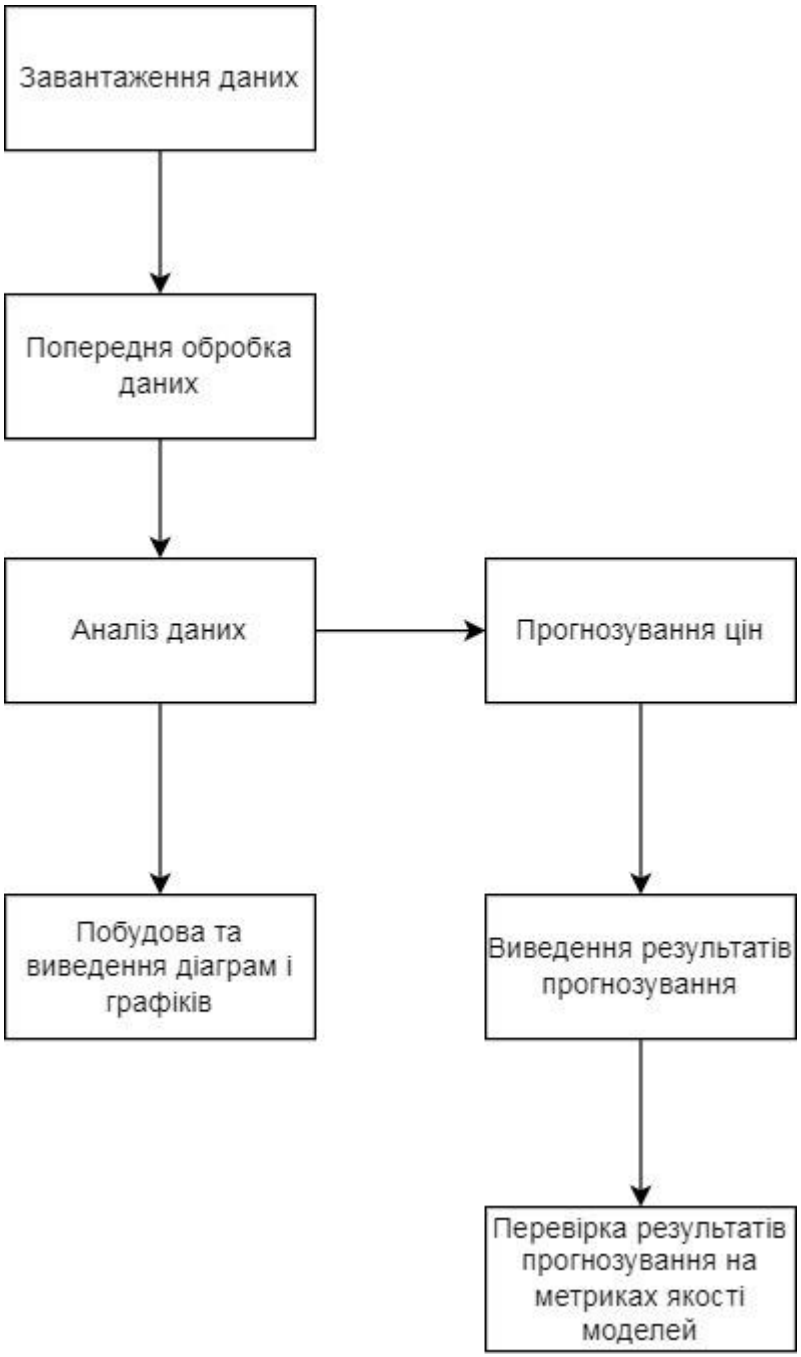

Рисунок 3.3 - Загальна схема роботи програми

Аналіз даних є важливим етапом у процесі прогнозування вартості автомобілів. Цей процес включає в себе збір, обробку та інтерпретацію великих наборів даних, що містять інформацію про автомобілі та їх ціни.

Перший крок у цьому процесі вже було зроблено та описано вище - це очищення даних, що включає в себе обробку відсутніх значень, виявлення та виправлення помилок у даних, зведення до єдиного числового формату значень, які мають числову та буквену частину.

Наступний крок - це аналіз даних, що може включати в себе статистичний аналіз, що допомагає зрозуміти розподіл даних, виявити будь-які аномалії та визначити кореляції між різними атрибутами.

Також можуть використовуватися методи візуалізації даних, такі як гістограми, графіки та діаграми, щоб допомогти визуально інтерпретувати дані та зрозуміти структуру та взаємозв'язки між атрибутами.

Нарешті, на основі аналізу даних, можна вибрати відповідні моделі машинного навчання для прогнозування цін на автомобілі. Для навчання моделей було зроблено розбиття даних на навчальну та тестову вибірку, де 70 процентів даних було віддано під тренувальну, а 30 процентів під контрольну. Використовуючи ці вибірки, було натреновано 10 моделей штучного інтелекту. Це 4 моделі простої лінійної регресії по основним параметрам автомобіля та одна модель множинної лінійної регресії з комбінацією цих параметрів. Аналогічна процедура була проведена для алгоритму випадкового лісу. Отримано 4 моделі випадкового лісу по одному з параметрів автомобіля та одна модель з комбінацією цих параметрів.

Для аналізу кожної моделі були використані метрики якості такі, як середня абсолютна похибка (MAE)**,** середньоквадратична помилка (RMSE) та коефіцієнт детермінації (R^2**)**.

3.5. Результати аналізу даних

Перед початком проведення аналізу даних необхідно побудувати кореляційну матрицю. Кореляційна матриця - це таблиця, яка показує кореляційні взаємозв'язки між кількома змінними. Кожен клітинка в таблиці показує кореляцію між двома змінними. Значення варіюються від -1 до +1. Значення +1 означає повну позитивну кореляцію (тобто, коли одна змінна зростає, інша теж зростає), -1 означає повну

негативну кореляцію (тобто, коли одна змінна зростає, інша спадає), а 0 означає відсутність кореляції.

Кореляційна матриця широко використовується в статистиці для оцінки сили взаємозв'язку між різними змінними. Вона може бути особливо корисною в машинному навчанні для визначення, які атрибути є найбільш пов'язаними з цільовою змінною, яку намагаємося спрогнозувати. За допомогою цього можна визначити, які особливості найкраще використовувати в моделі. На рисунку 3.4. зображена дана матриця для нашого набору даних.

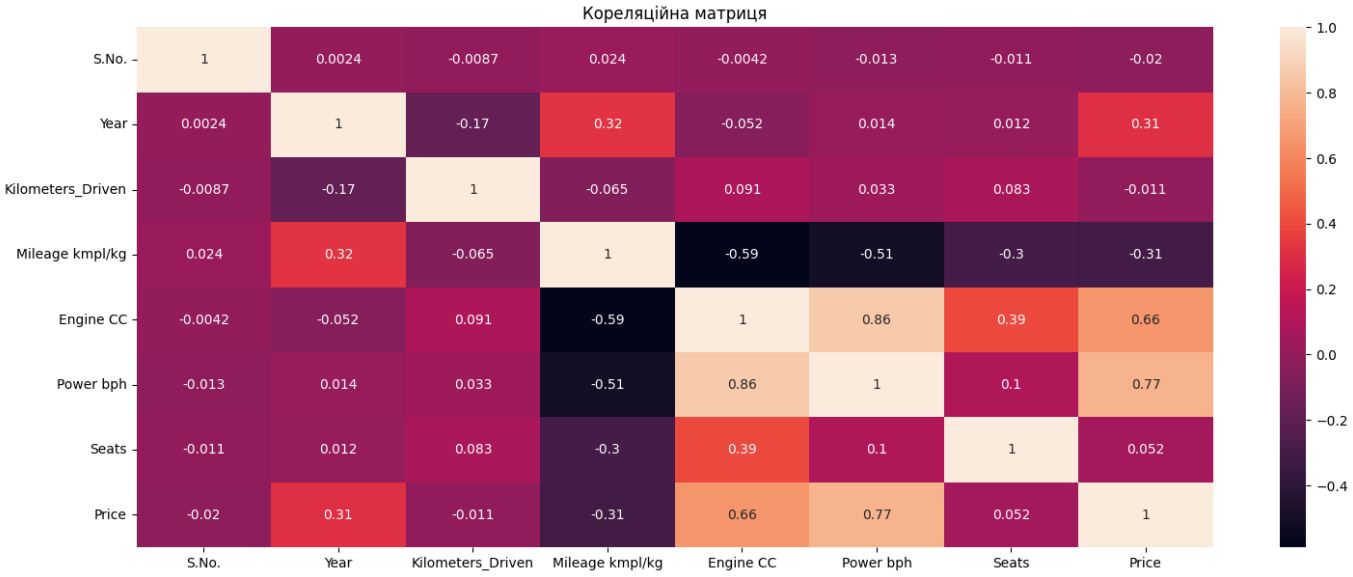

Рисунок 3.4 – Кореляційна матриця для набору даних про ціни на вживані автомобілі

Як можна побачити з представленої матриці, то найбільше пов’язаними з цільовою змінною Price є параметри Engine CC та Power bph. Отже, їх необхідно використовувати в моделі прогнозування. Додатково можна також додати параметри Year та Seats.

У ході подальшого процесу аналізу даних, було вирішено дослідити розподіл кожного параметру та виокремити певні залежності між змінними на основі яких можливо було б дійти до певних висновків. Почнемо з виведення топ-10 автобілів, які найчастіше зустрічаються у даному дата сеті. Зобразимо це на рисунку 3.5.

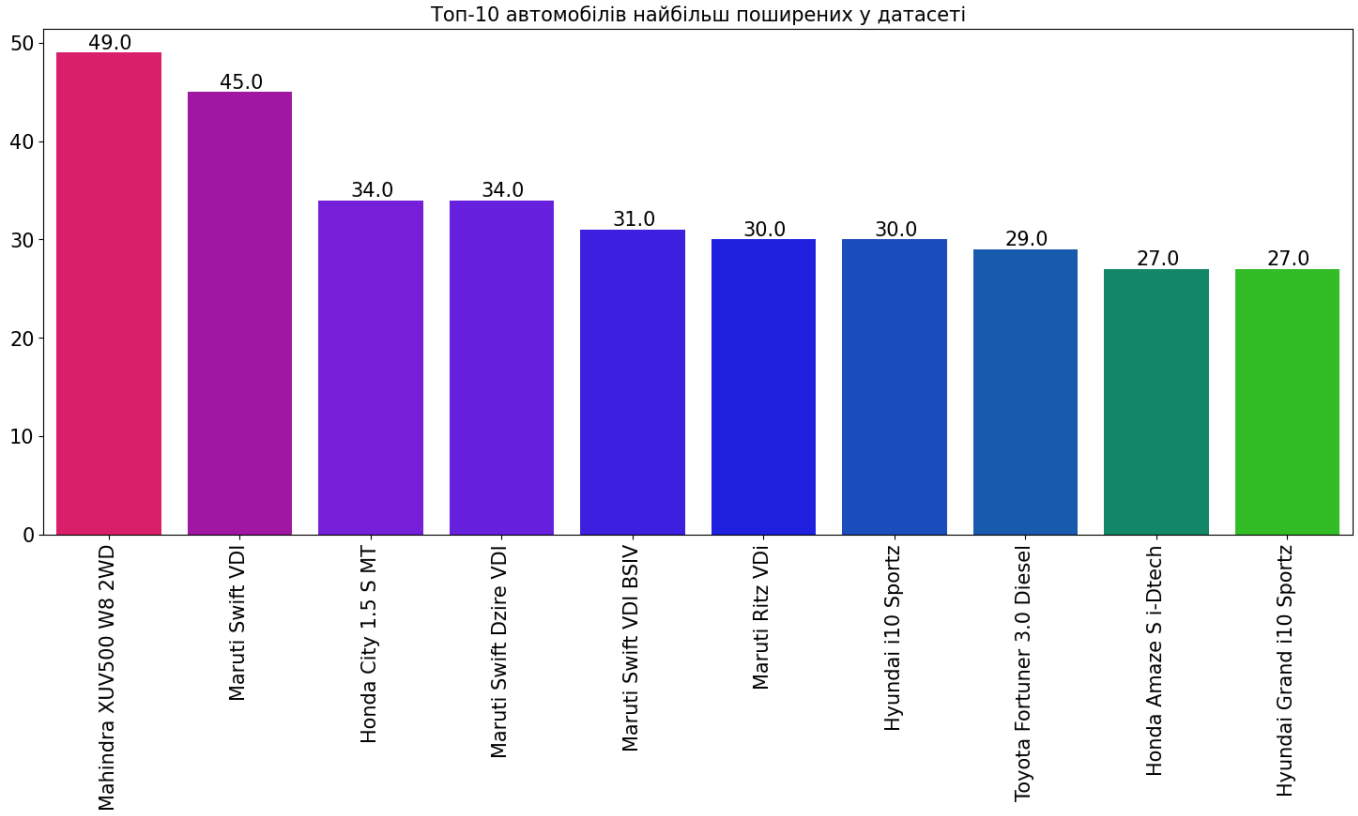

Рисунок 3.5 - Топ-10 автомобілів, які найчастіше зустрічаються у дата сеті

Можна побачити, що на індійському автомобільному ринку популярність мають автомобілі їх вітчизняних брендів Mahindra та Maruti. Тепер на рисунку 3.6 виведемо розподіл кількості проданих автомобілів за містами Індії.

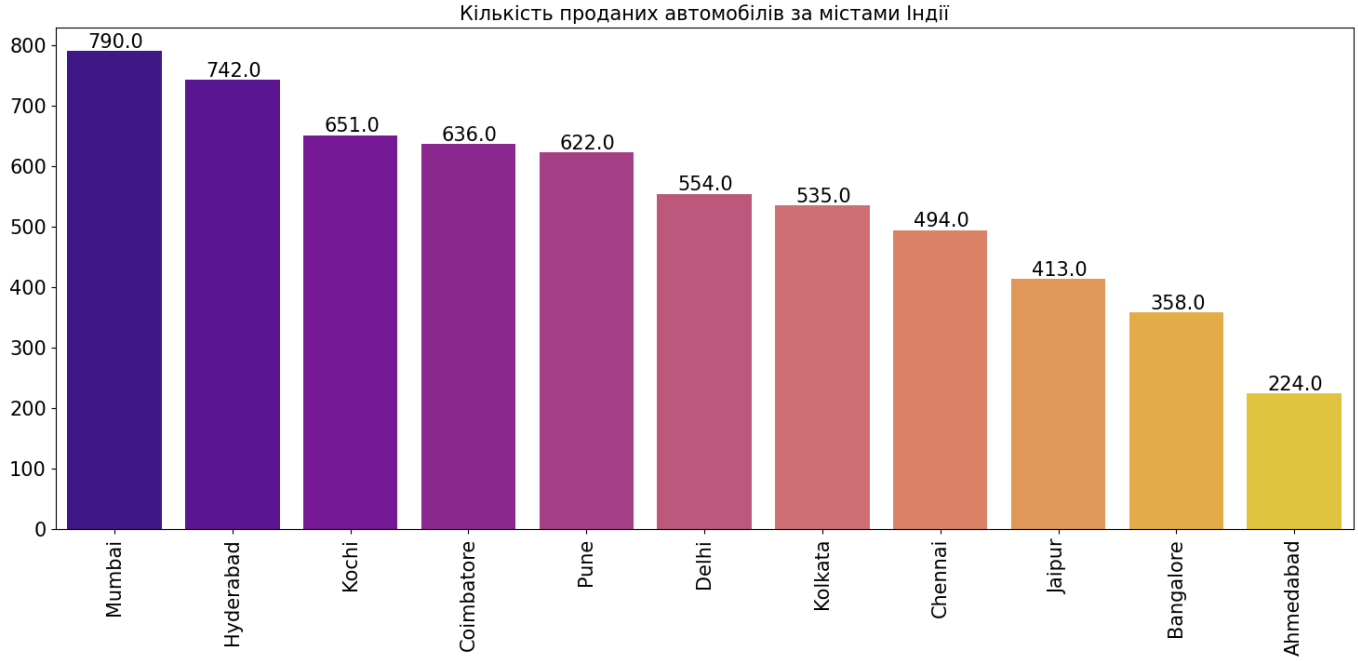

Рисунок 3.6 - Кількість проданих автомобілів за містами Індії

Найбільше проданих автомобілів було у місті Мумбай. Це і не дивно, оскільки дане місто, раніше відомий як Бомбей, є одним з найбільших міст у світі та є фінансовою столицею Індії. Місто розташоване на західному узбережжі Індії і є важливим портом на Аравійському морі. У місті Мумбай розташовані штаб-квартири багатьох з найбільших компаній Індії та численні фінансові установи, включаючи Бомбейську фондову біржу, Азійський Інститут Фінансів, Резервний банк Індії та Національну біржу Індії. Отже, як можна зрозуміти, то у цьому місті мешкає багато багатих та впливових людей, та й загалом середня платоспроможність пересічного мешканця Мумбай є доволі високою, тому там купляють багато автомобілів. Як можна побачити, то кількість проданих автомобілів допомагає зрозуміти рівень фінансовго життя та платоспрожності населення певного міста.

Далі візуалізуємо дані на рисунку 3.7 про кількість проданих автомобілів за роком виробництва.

Рисунок 3.7 - Кількість проданих автомобілів за роком виробництва

Загалом видно, що більшість автомобілів було продано 2014 року виробництва. Цей факт може свідчити про те, що у рамках даної вибірки з автомобільного ринку Індії, вік більшості автомобілів сягає 9 років. Це є доволі непоганим показником як для країни Сходу, що свідчить про фінансовий ріст Індії.

Тепер на рисунку 3.8 звернімо свою увагу на кількість проданих автомобілів за пробігом.

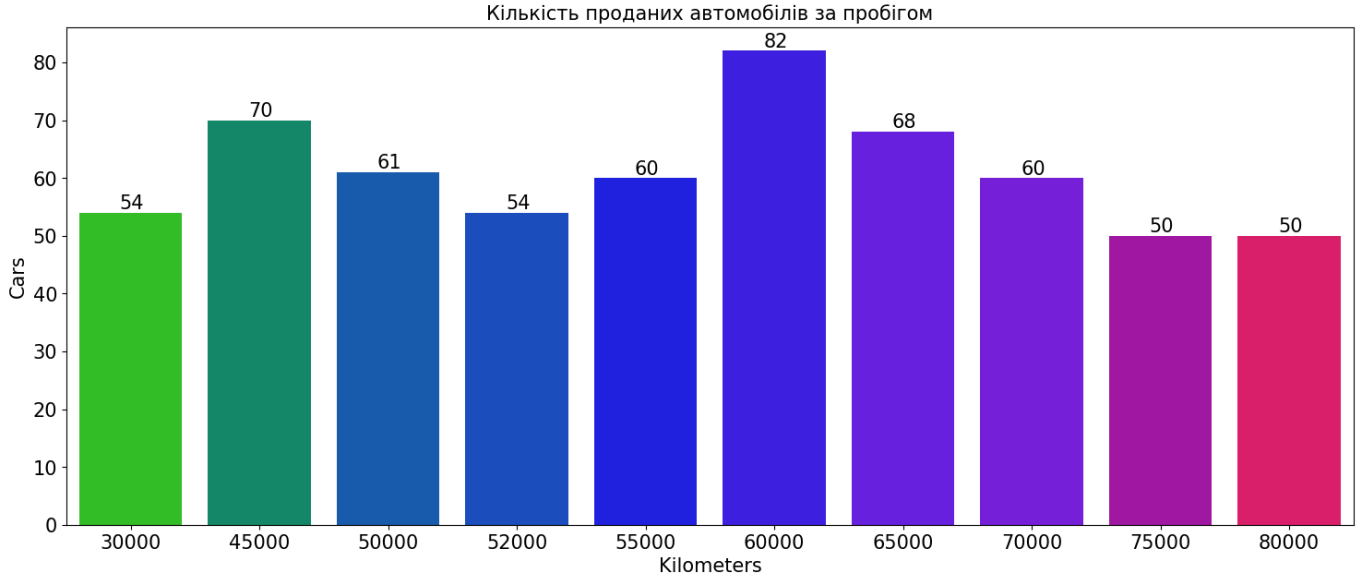

Рисунок 3.8 - Кількість проданих автомобілів за пробігом

Було виведені дані у межах від 30 до 80 тисяч кілометрів пробігу. Найбільша кількість автомобілів з пробігом 60 тисяч кілометрів, що є доволі невеликим показником. Можливо із-за відносно свіжих за роками автомобілями у вибірці пробіги не є великими. Тепер на рисунку 3.9 візуалізуємо дані про кількість автомобілів за типом палива.

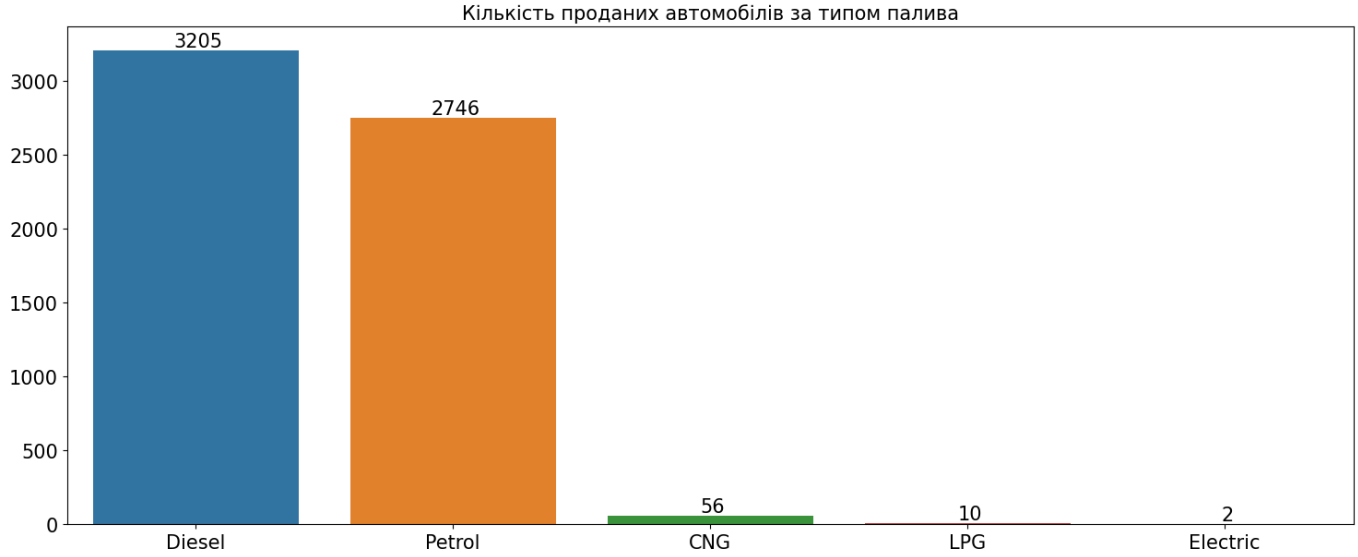

Рисунок - 3.9 Кількість проданих автомобілів за типом палива

Можна побачити, що у даній вибірці найбільша кількість проданих автомобілів використовує дизельне пальне. У той же час найменшу кількість займають електричні автомобілі. Автомобілі, які використовують у якості пального природний газ також не дуже багато. Загалом можемо констатувати, що у Індії найбільшу популярність мають автомобілі з дизелем та бензином. На рис. 3.10 виведемо кількість проданих автомобілів за типом коробки передач.

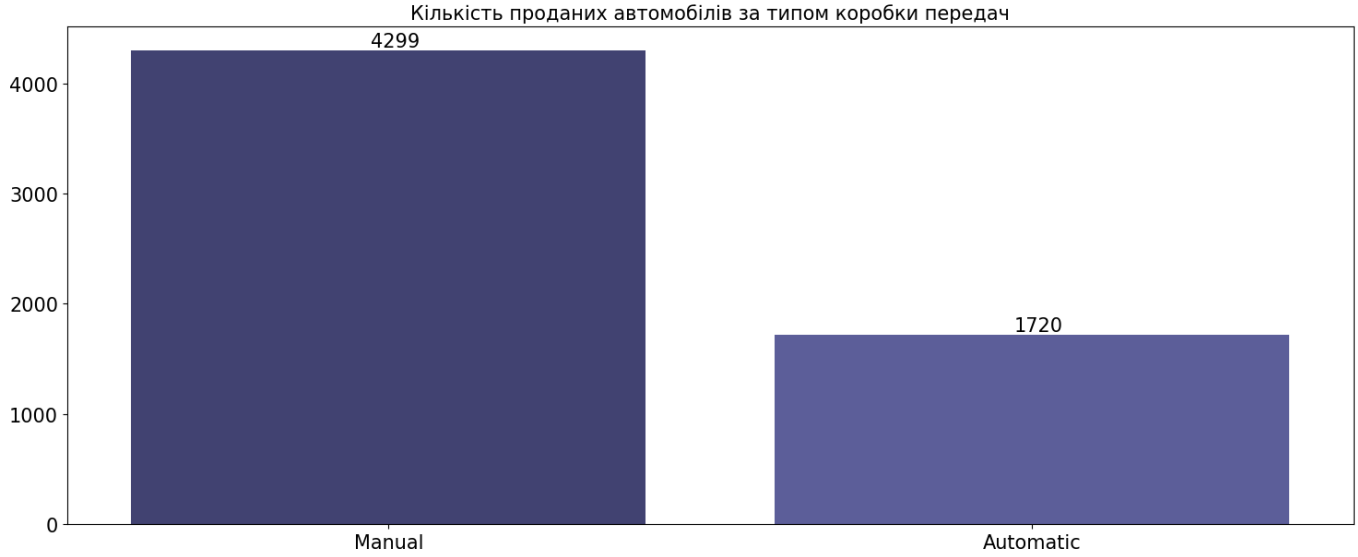

Рисунок - 3.10 Кількість проданих автомобілів за типом коробки передач

Найбільшу популярність серед проданих автомобілів має механічна коробка передач. Це можна пояснити тим, що загальні умови експлуатації автомобілів у Індії є доволі суровими, тому потенційні покупці віддають свою перевагу більш простій та надійній механічній коробці передач. На рис. 3.11 виведемо найбільшу кількість проданих автомобілів за об’ємом двигуна.

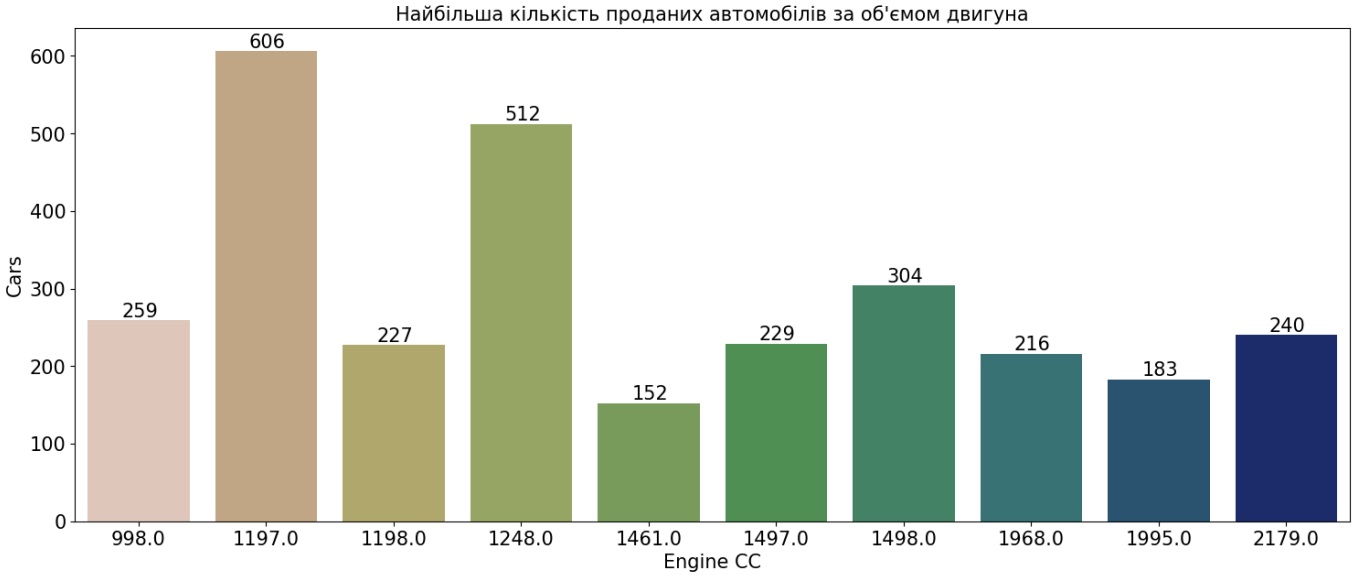

Рисунок - 3.11 Найбільша кількість проданих автомобілів за об’ємом двигуна

Більшість проданих автомобілів мають невеликі або середні об’єми. Найбільш популярним виявився двигун з об’ємом 1197 $см^3$, що свідчить про те, що індійці більше полюбляють економічні автомобілі з малим об’ємом двигуна. Із-за цього вивливають дані з рисунку 3.12 про найбільшу кількість проданих автомобілів за потужністю.

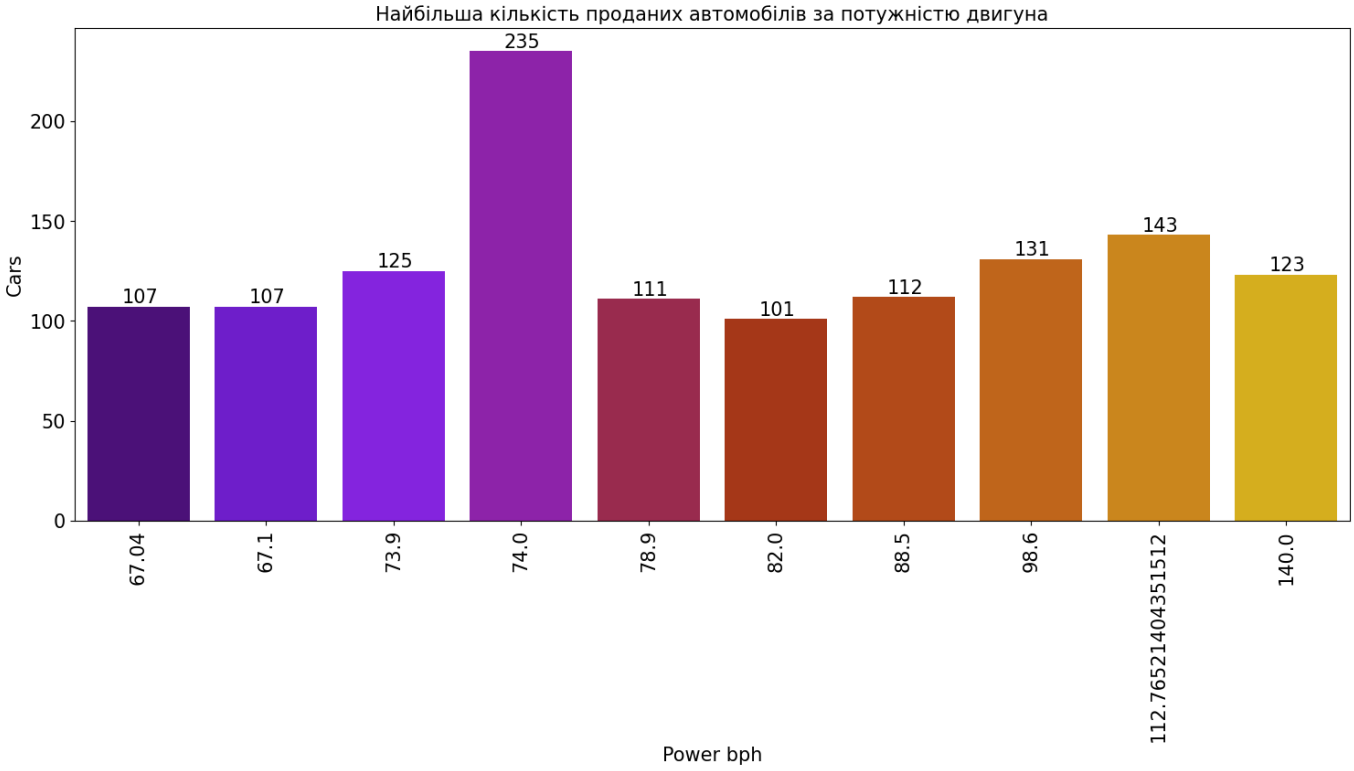

Рисунок - 3.12 Найбільша кількість проданих автомобілів за потужністю двигуна

Із-за популярності серед індійського автомобільного ринку малооб’ємних двигунів як наслідок маємо, що найбільша кількість проданих автомобілів має невелику потужність.

Тепер перейдемо до більш глибокого аналізу та виведемо на рисунках 3.13 та 3.14 відповідно конкретні продані автомобілі з найбільшим об’ємом двигуну та потужністю.

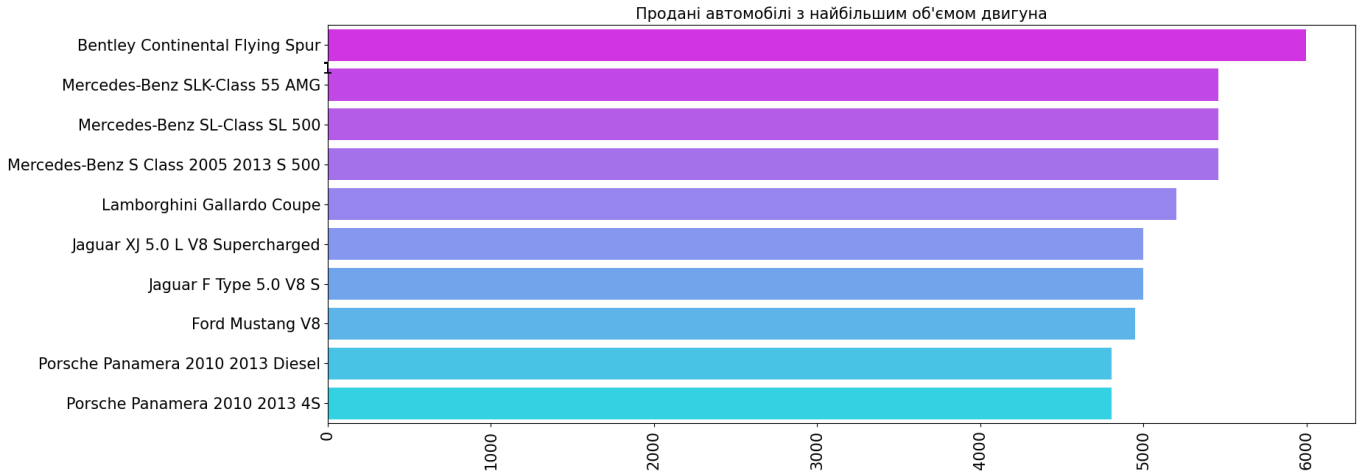

Рисунок 3.13 - Продані автомобілі з найбільшим об’ємом двигуна

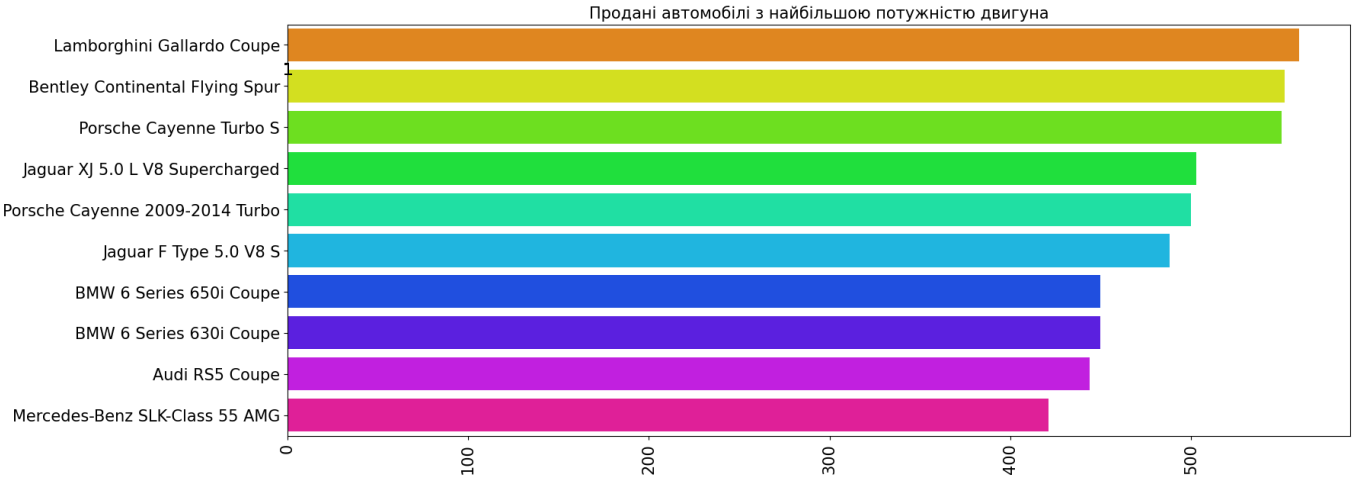

Рисунок 3.14 - Продані автомобілі з найбільшою потіжністю двигуна

Тепер на рисунку 3.15 виведемо мінімальну, середню та максимальну потужність автомобілів в залежності від типу палива.

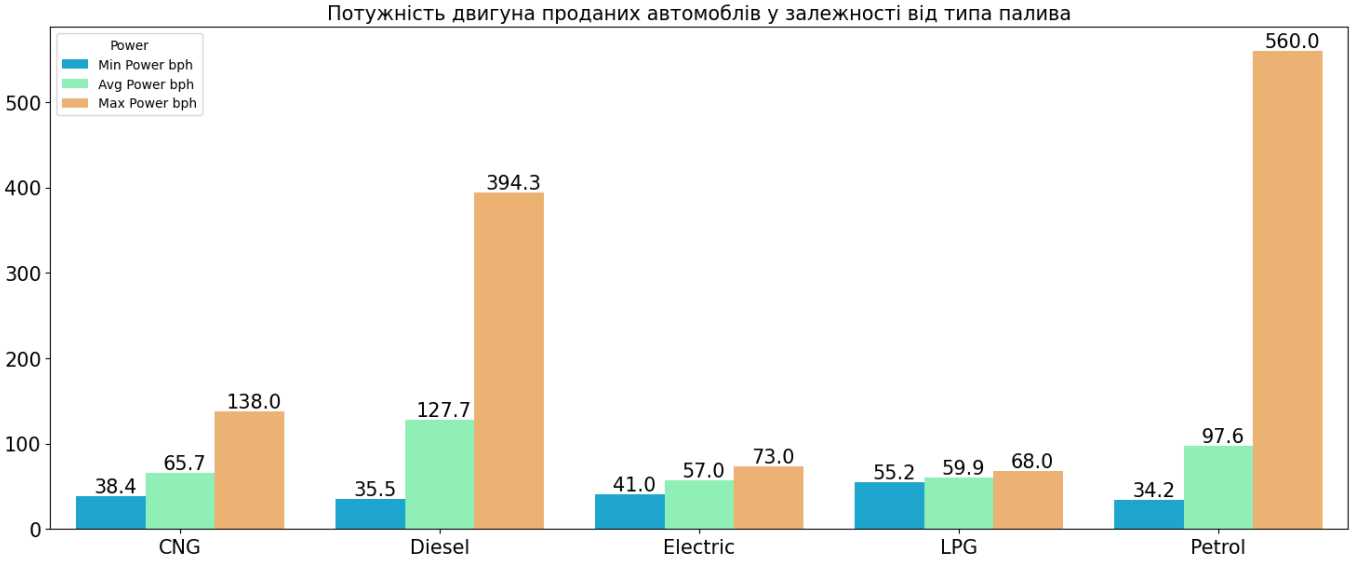

Рисунок 3.15 - Потужність двигуна проданих автомобілів у залежності від типа палива

Максимальний показник потужності двигуна був у автомобілів на бензині, у той час як найменші показник були у автомобілів на газу та електриці. Це пояснюється тим, що у автомобілів на газу чи електриці першочерговим завданням є економічність, тому велика потужність їм не потрібна. Також на рисунку 3.16 зобразимо співвідношення кількості проданих автомобілів за роком виробництва з механічною та автоматичною коробкою передач.

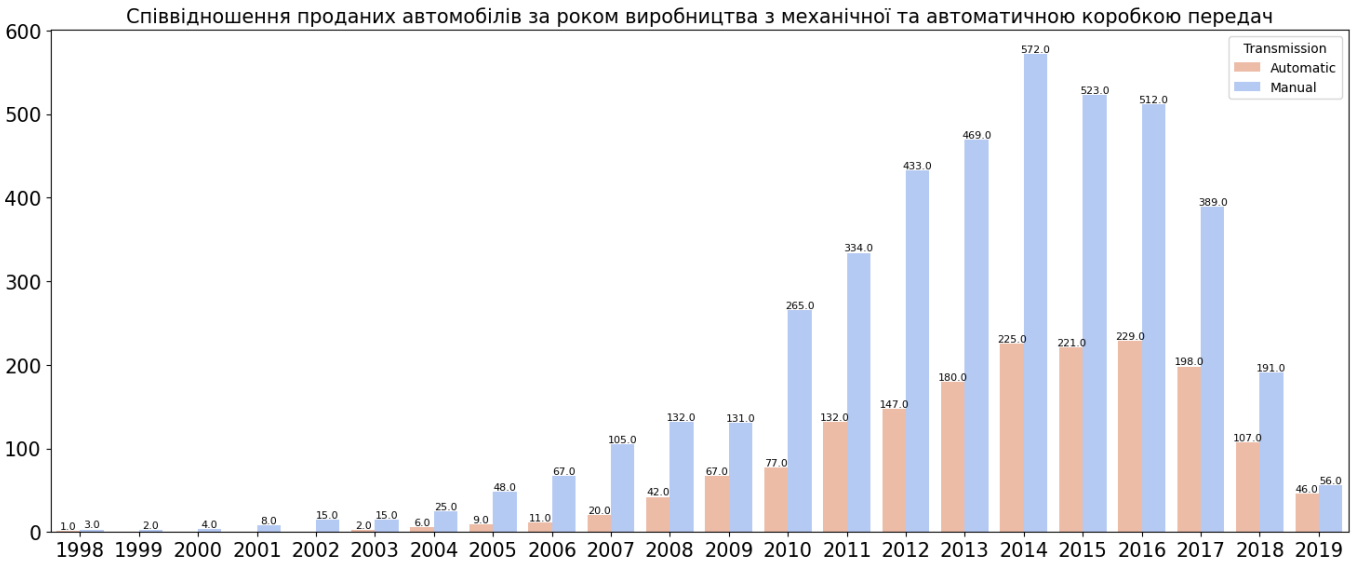

Рисунок 3.16 - Співвідношення проданих автомобілів за роком виробництва з механічною та автоматичною коробкою передач

Загалом можна побачити таку тенденцію, що чим старіший автомобіль, тим більше його купують з механічною коробкою передач, а ніж з автоматичною. Тобто, коли стоїть вибір купувати автомобіль, якому вже більше 10 років, то краще його купити на механічній коробці передач, оскільки вона більш надійна та в неї краще ресурс, ніж в автоматичній. На рисунку 3.17 покажемо тип коробки передач впливає на запас ходу автомобіля.

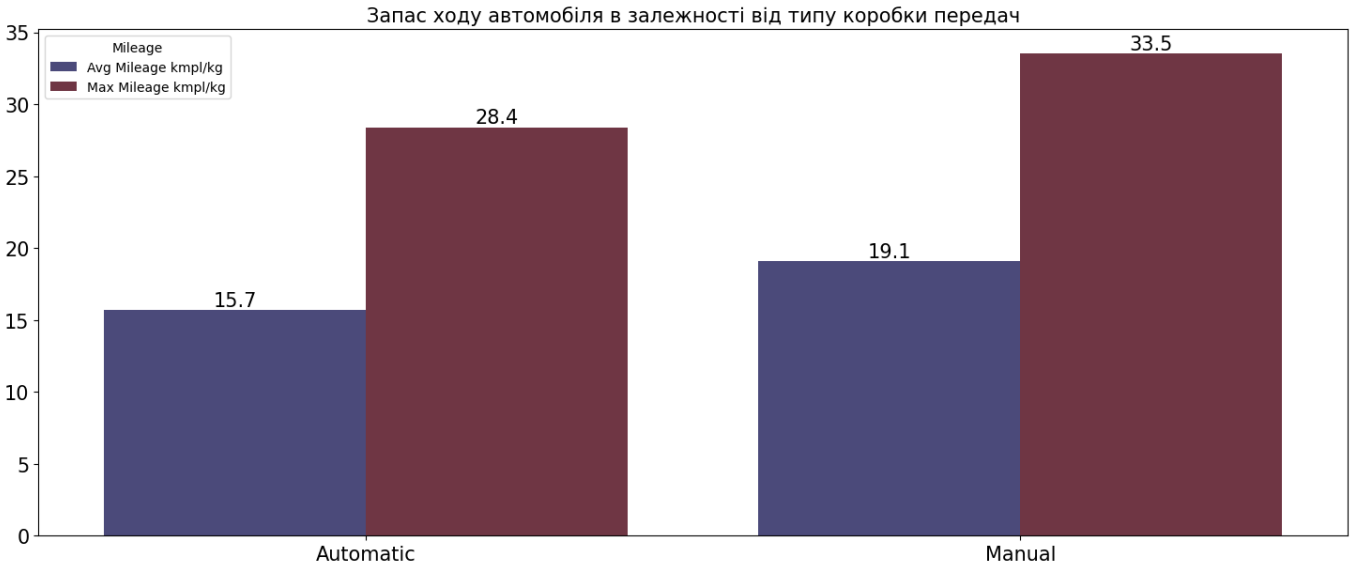

Рисунок 3.17 - Запас ходу автомобіля в залежності від типу коробки передач

Як можна побачити, то автомобілі з механічною коробкою передач більш економічні з кращим запасом ходу, тому вони користуються більшим попитом на вторинному автомобільному ринку Індії.

Загалом ми дійшли висновку, що індійці більше полюбляють вживані автомобілі з малим об’ємом двигуна, невеликої потужності з використанням механічної коробки передач. Переважаючим типом палива є дизель та бензин. Тобто, популярність мають прості, надійні та економічні автомобілі.

Тепер після завершення аналізу даних можемо перейти до побудови моделі прогнозування цін на вживані автомобілі. Розглянемо це у наступному пункті.

3.6. Результати прогнозування цін

З попереднього пункту за допомогою кореляційної матриці на рисунку 3.4. було визначено, що найбільший вплив на формування ціни автомобіля мають параметри Engine CC (об’єм двигуна) та Power bph (потужність). Додатково також можна розглянути параметри Year (рік виробництва) та Seats (кількість місць у автомобілі). На основі цих параметрів будемо будувати моделі машинного навчання, використовуючи алгоритми лінійної регресії та випадкового лісу.

Почнемо прогнозування з алгоритму лінійної регресії. Спочатку побудуємо моделі простої лінійної регресії за кожним з параметрів Year, Engine CC, Power bph та Seats. Нагадаємо, що під навчання було виділено 70 процентів вибірки, а під тестову – 30 процентів. Результати виведені на рисунках 3.18 – 3.25.

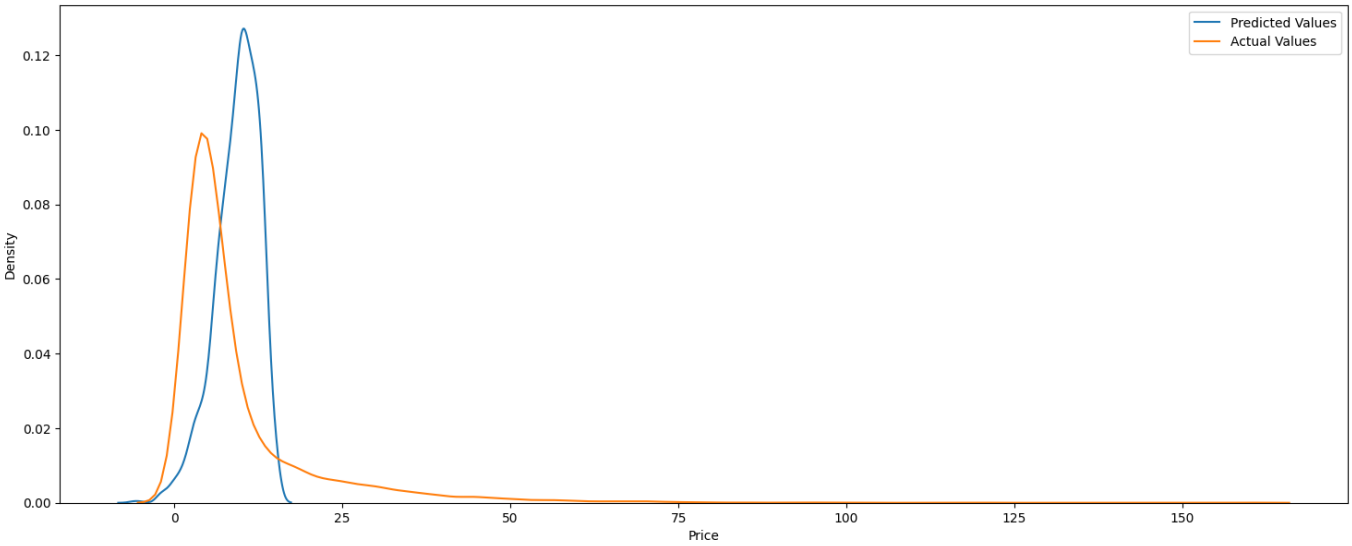

Рисунок 3.18 - Графік розподілу фактичних та прогнозованих значень ціни автомобіля для моделі лінійної регресії за параметром Year

```
Mean Squared Error =  110.53237998480884
Root Mean Squared Error =  10.513438066817574
Mean Absolute Error =  6.799484666973446
R-Squared: =  0.10492931084611967
```

Рисунок 3.19 - Результати метрик якості моделі лінійної регресії для параметру Year

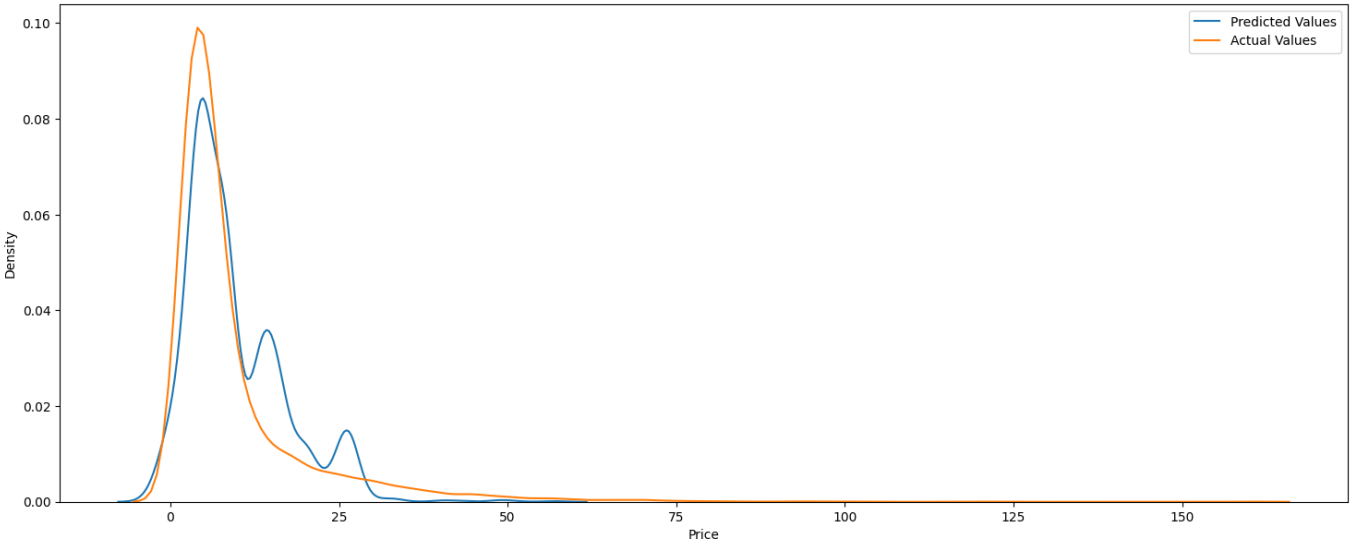

Рисунок 3.20 - Графік розподілу фактичних та прогнозованих значень ціни автомобіля для моделі лінійної регресії за параметром Engine CC

```
Mean Squared Error =  70.86117267463437
Root Mean Squared Error =  8.417907856150148
Mean Absolute Error =  5.050974470273945
R-Squared: =  0.4261793813825948
```

Рисунок 3.21 - Результати метрик якості моделі лінійної регресії для параметру Engine CC

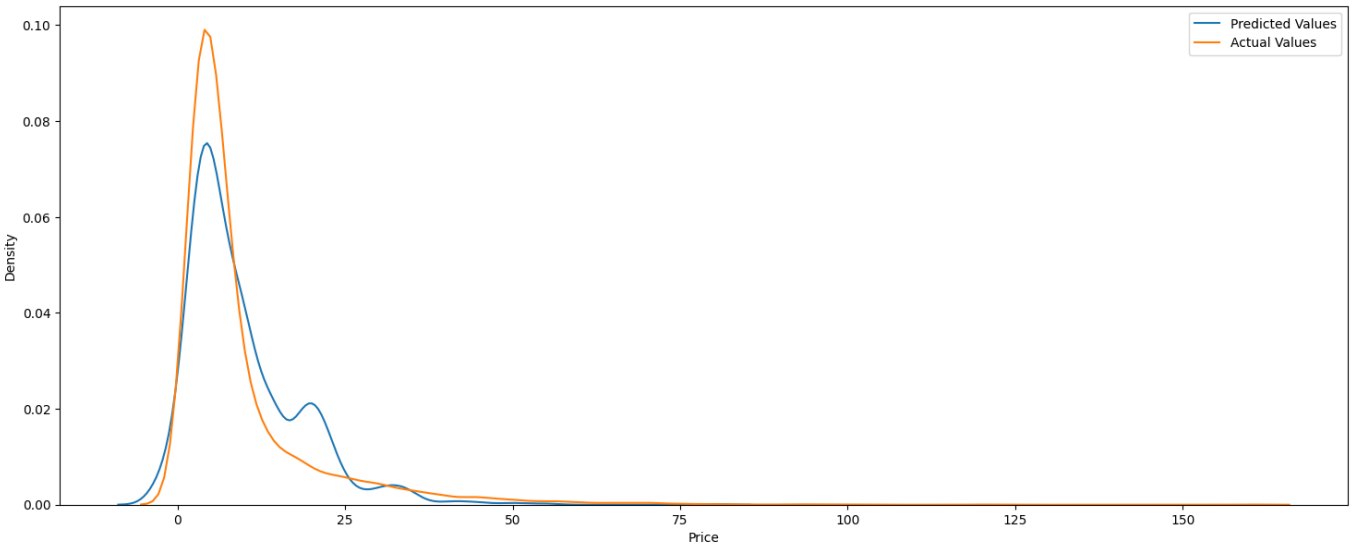

Рисунок 3.22 - Графік розподілу фактичних та прогнозованих значень ціни автомобіля для моделі лінійної регресії за параметром Power bph

```
Mean Squared Error =  49.60555770842773
Root Mean Squared Error =  7.043121304395355
Mean Absolute Error =  4.319770181309899
R-Squared: =  0.5983034045765845
```

Рисунок 3.23 - Результати метрик якості моделі лінійної регресії для параметру Power bph

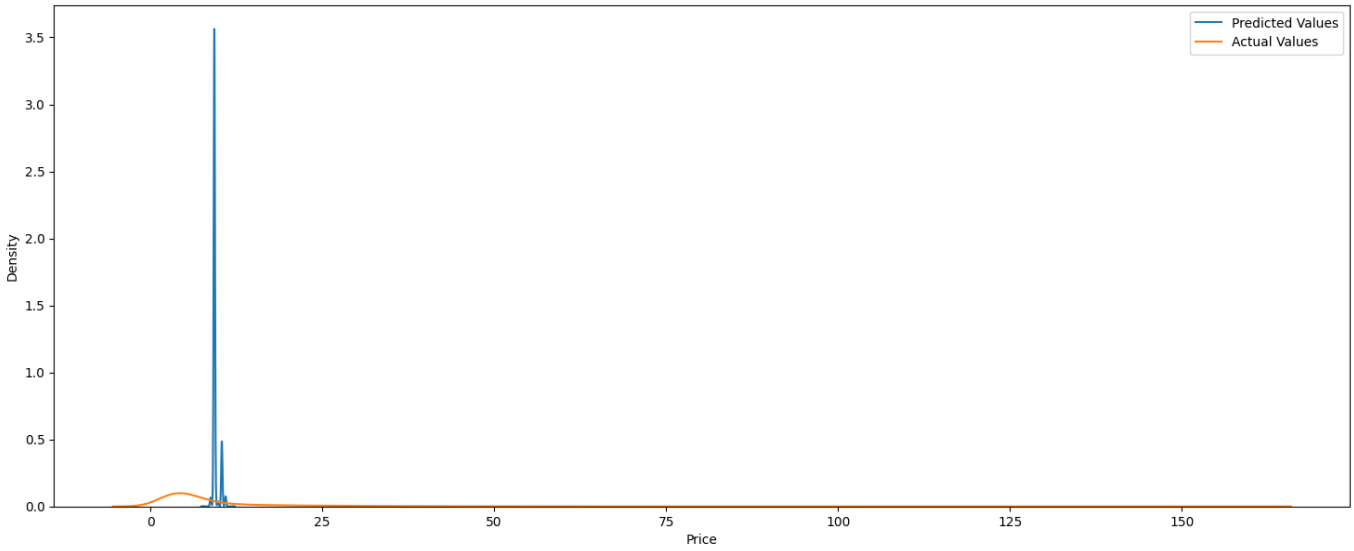

Рисунок 3.24 - Графік розподілу фактичних та прогнозованих значень ціни автомобіля для моделі лінійної регресії за параметром Seats

```
Mean Squared Error =  122.90133507192914
Root Mean Squared Error =  11.086087455542156
Mean Absolute Error =  7.132753701258022
R-Squared: =  0.00476780925297815
```

Рисунок 3.25 - Результати метрик якості моделі лінійної регресії для параметру Seats

Проаналізувавши отримані результати, можна дійти висновку що найбільш точний прогноз ціни автомобіля був за параметром Power bph, тобто потужність. Отже, з чотирьох обраних нами параметрів потужність автомобіля має більший вплив на формування ціни автомобіля, ніж інші параметри. Тепер спробуємо скомбінувати ці параметри разом та побудувати модель множинної лінійної регресії. Результати відображені на рисунках 3.26 – 3.28.

	Actual	Predicted
4859	5.38	8.71
2650	12.90	15.10
727	3.25	0.59
3960	10.82	10.68
421	4.35	7.29

Рисунок 3.26 - Фактична та прогнозована ціна для п’яти випадкових автомобілів для моделі множинної лінійної регресії

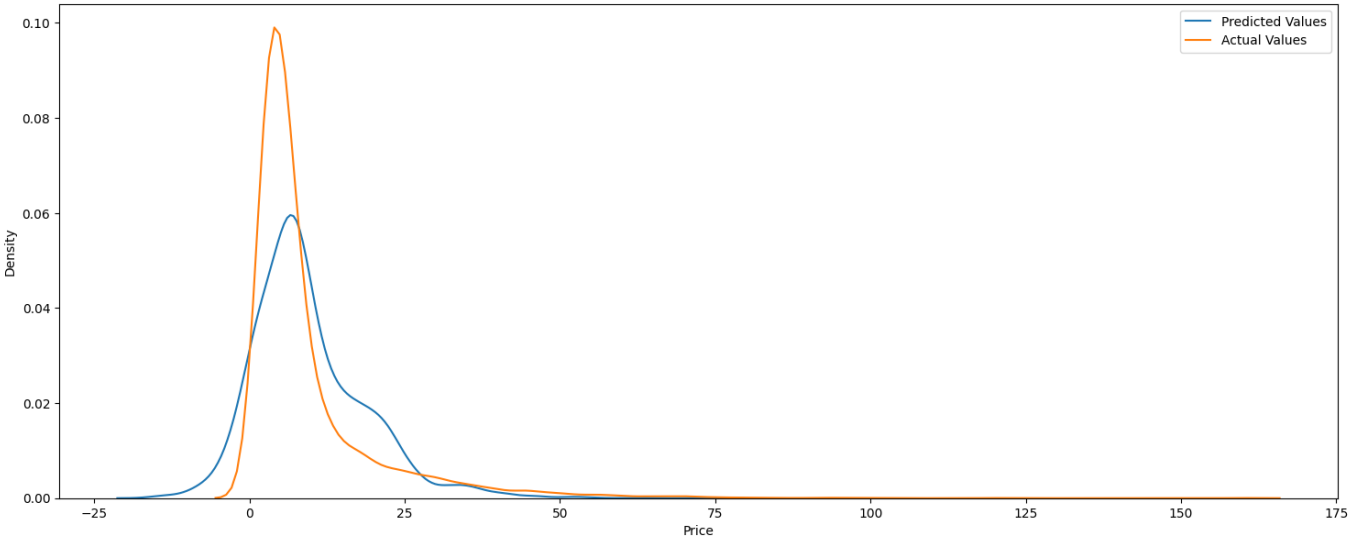

Рисунок 3.27 - Графік розподілу фактичних та прогнозованих значень ціни автомобіля для моделі множинної лінійної регресії

```
Mean Squared Error =  38.189323569345085
Root Mean Squared Error =  6.179751092830931
Mean Absolute Error =  3.8963336711624574
R-Squared: =  0.6907499488364215
```

Рисунок 3.28 - Результати метрик якості моделі множиної лінійної регресії з параметрами Year, Engine CC, Power bph та Seats

Можна побачити, що результати отримані від моделі множинної лінійної регресії набагато кращі, ніж результати прогнозування по кожному параметру

окремо. Тому підходимо до висновку, що для кращої оцінки вартості автомобіля необхідно зважати та враховувати на всі параметри разом, а не на кожний окремо.

Тепер повторимо процедуру, яка була зроблена для алгоритму лінійної регресії, з алгоритмом випадкого лісу. Перш за все це нам дасть можливість порівняти ці алгоритми та зрозуміти який з них кращий. Тому почнемо знову з прогнозування ціни автомобіля за кожним параметром окремо тільке вже за допомогою алгоритму випадкового лісу. Результати виведені на рисунках 3.29 – 3.40.

	Actual	Predicted
5150	7.25	9.59
2164	4.25	4.88
1136	3.90	7.11
5641	8.41	7.11
1515	13.48	13.75

Рисунок 3.29 - Фактична та прогнозована ціна для п’яти випадкових автомобілів за параметром Year

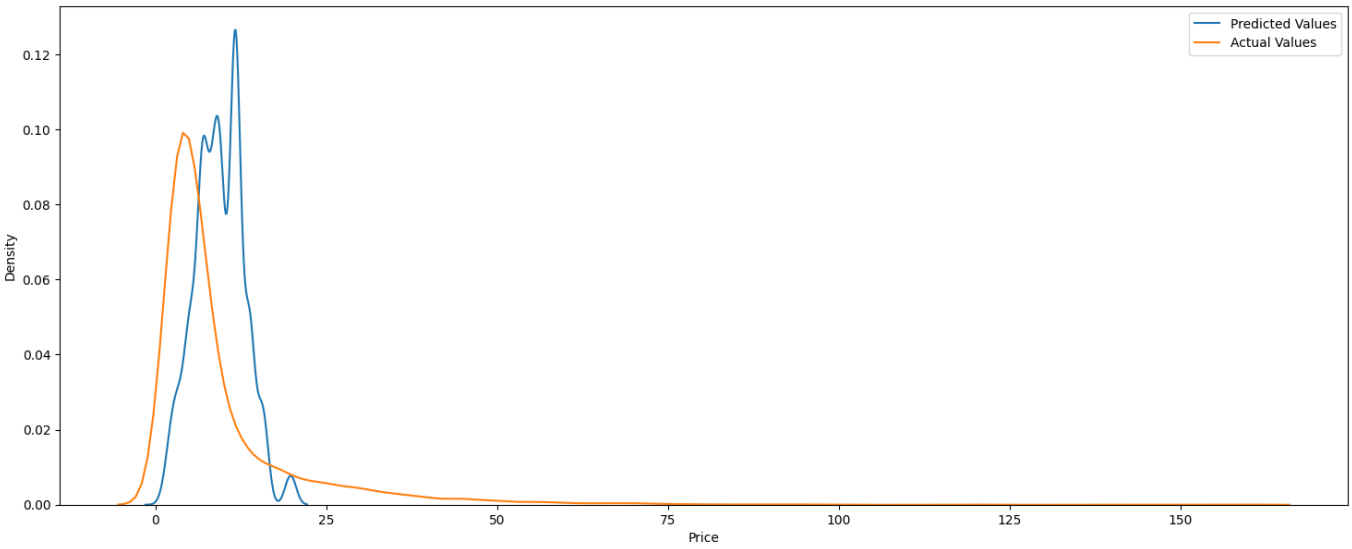

Рисунок 3.30 - Графік розподілу фактичних та прогнозованих значень ціни автомобіля для моделі випадкового лісу за параметром Year

```
Mean Squared Error =  116.1488492854837
Root Mean Squared Error =  10.77723755354236
Mean Absolute Error =  6.885058579110939
R-Squared: =  0.07521855278172873
```

Рисунок 3.31 - Результати метрик якості моделі випадкового лісу за параметром Year

	Actual	Predicted
5150	7.25	14.07
2164	4.25	5.68
1136	3.90	5.65
5641	8.41	7.22
1515	13.48	9.72

Рисунок 3.32 - Фактична та прогнозована ціна для п’яти випадкових автомобілів за параметром Engine CC

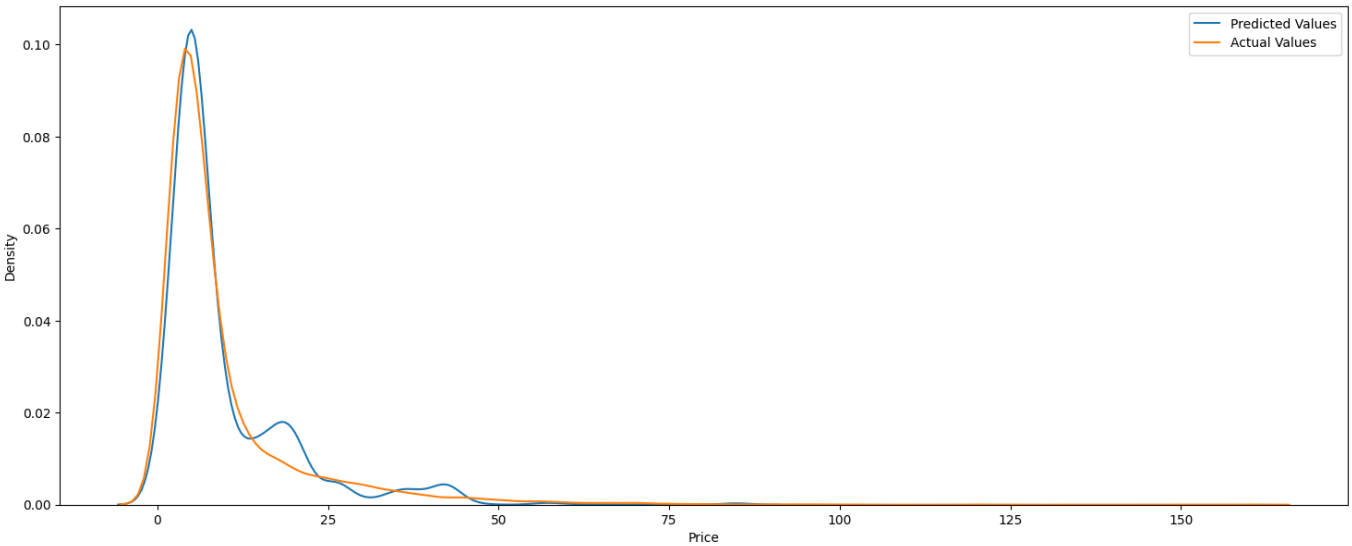

Рисунок 3.33 - Графік розподілу фактичних та прогнозованих значень ціни автомобіля для моделі випадкового лісу за параметром Engine CC

```
Mean Squared Error =  52.43186135844851
Root Mean Squared Error =  7.240984833463505
Mean Absolute Error =  3.71961937535334
R-Squared: =  0.5825355746036323
```

Рисунок 3.34 - Результати метрик якості моделі випадкового лісу за параметром Engine CC

	Actual	Predicted
5150	7.25	10.70
2164	4.25	4.33
1136	3.90	4.81
5641	8.41	11.01
1515	13.48	10.63

Рисунок 3.35 - Фактична та прогнозована ціна для п’яти випадкових автомобілів за параметром Power bph

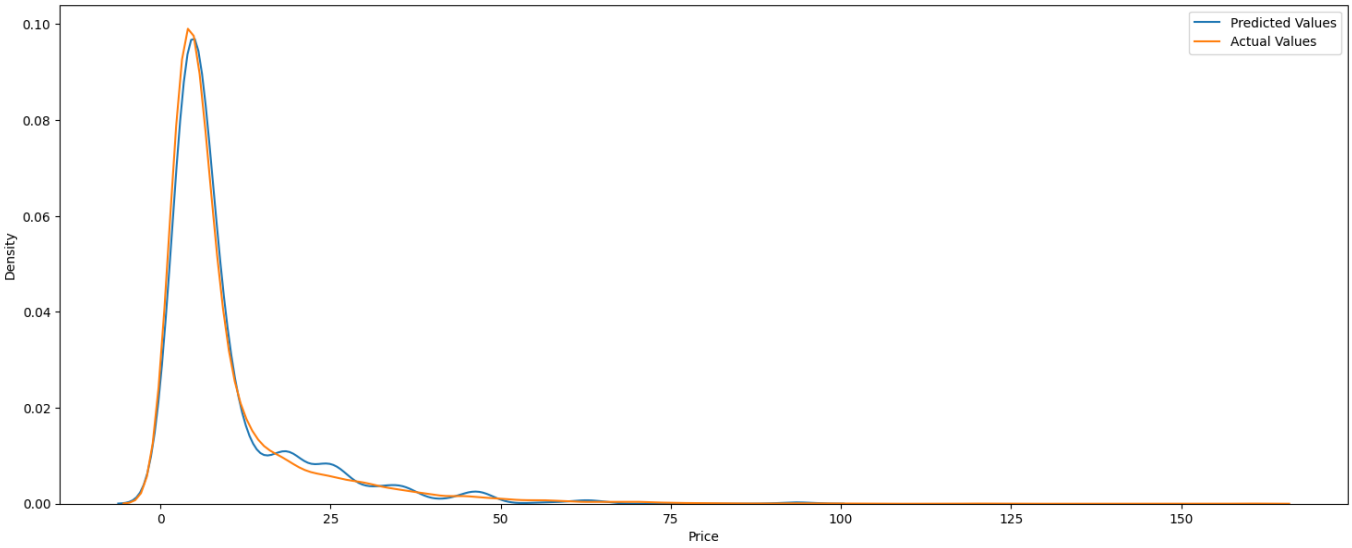

Рисунок 3.36 - Графік розподілу фактичних та прогнозованих значень ціни автомобіля для моделі випадкового лісу за параметром Power bph

```
Mean Squared Error =  36.11952408461719
Root Mean Squared Error =  6.009952086715599
Mean Absolute Error =  2.804879087196865
R-Squared: =  0.7124150091775197
```

Рисунок 3.37 - Результати метрик якості моделі випадкового лісу за параметром Power bph

	Actual	Predicted
5150	7.25	14.59
2164	4.25	8.51
1136	3.90	8.51
5641	8.41	14.59
1515	13.48	7.56

Рисунок 3.38 - Фактична та прогнозована ціна для п’яти випадкових автомобілів за параметром Seats

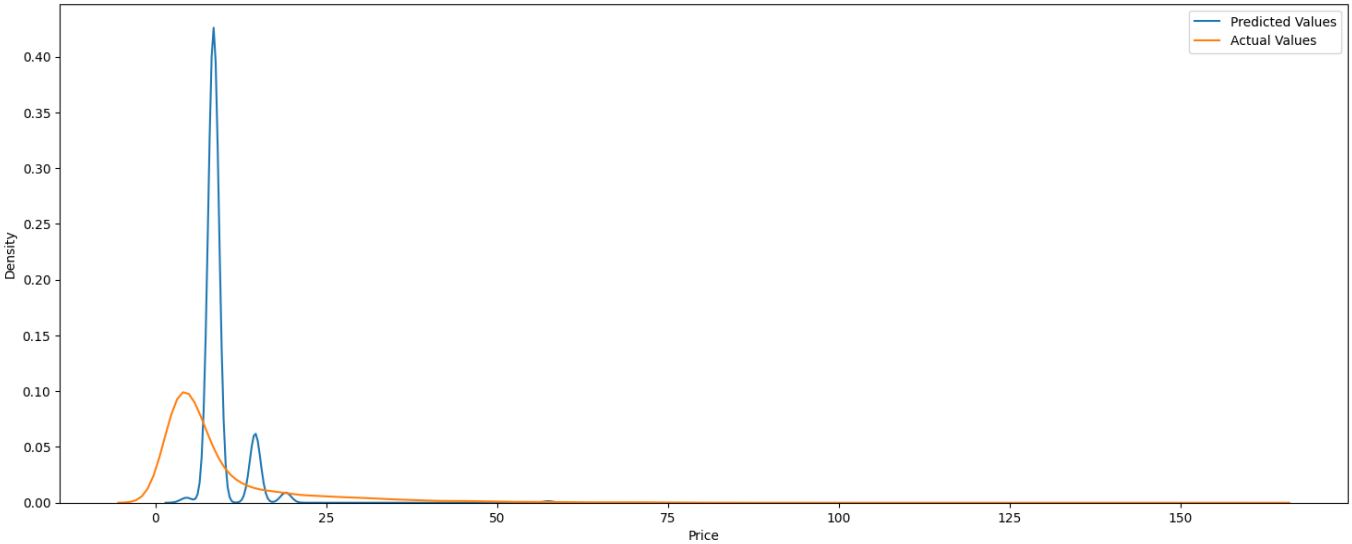

Рисунок 3.39 - Графік розподілу фактичних та прогнозованих значень ціни автомобіля для моделі випадкового лісу за параметром Seats

```
Mean Squared Error =  112.14260363846006
Root Mean Squared Error =  10.589740489665461
Mean Absolute Error =  6.706888631515043
R-Squared: =  0.10711642925797449
```

Рисунок 3.40 - Результати метрик якості моделі випадкового лісу за параметром Seats

Аналіз отриманих результатів показав, що алгоритм випадкового лісу для тих самих параметрів автомобіля дає кращі показники, ніж алгоритм лінійної регресії. Також знову був отриманий висновок, що найбільш точний прогноз ціни автомобіля був за параметром Power bph, тобто потужність. Тепер знову спробуємо скомбінувати ці параметри разом та побудувати модель випадкового лісу для Year, Engine CC, Power bph та Seats. Результати відображені на рисунках 3.41 – 3.43.

	Actual	Predicted
5150	7.25	8.40
2164	4.25	2.49
1136	3.90	4.38
5641	8.41	9.26
1515	13.48	14.24

Рисунок 3.41 - Фактична та прогнозована ціна для п’яти випадкових автомобілів для моделі випадкового лісу за декількома параметрами

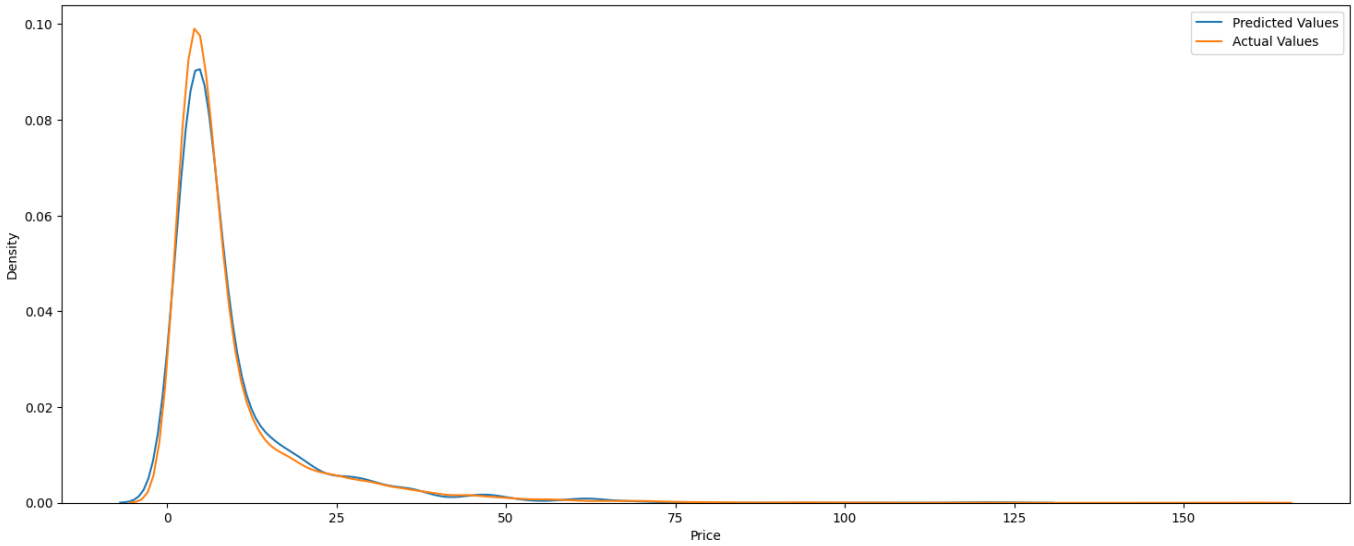

Рисунок 3.42 - Графік розподілу фактичних та прогнозованих значень ціни автомобіля для моделі випадкового лісу за декількома параметрами

```
Mean Squared Error =  23.041052751313625
Root Mean Squared Error =  4.800109660342525
Mean Absolute Error =  1.8776449500560701
R-Squared: =  0.8165462831541336
```

Рисунок 3.43 - Результати метрик якості моделі випадкового лісу за кількома параметрами

Загалом можна побачити, що результат прогнозування ціни автомобіля за параметрами Year, Engine CC, Power bph та Seats, набагато кращий при використанні алгоритму випадкового лісу, ніж лінійної регресії.

3.7 Висновки до розділу 3

У даному розділі представлено розробку програмного продукту з використанням Python в середовищі Google Colab. Для обробки даних та тренування моделей було використано бібліотеки, такі як Pandas, Scikit-learn, Matplotlib і Seaborn.

Дані були взяті з популярної платформи kaggle.com, де зібрані різноманітні вибірки за будь-якими темами. Проте перед початком роботи дані необхідно було попередньо обробити, тобто заповнити пропуски у колонках середніми значеннями по колонці, видалити майже пусті колонки якщо такі є та привести значення даних, що мають числову та буквенну частину до числової.

Після цього був початий аналіз даних. Виведена матриця кореляції, з якої стало зрозумілим, що найбільший зв'язок з цільовою прогнозованою змінної Price, мають параметри Engine CC та Power bph. Додатково також були взяті до уваги параметри Year та Seats. На основі цих параметрів було вирішено будувати моделі прогнозування.

Для процесу прогнозування вибірка даних була поділена на тренувальну (70 відсотків даних) та тестову (30 відсотків даних). Моделі були побудовані на алгоритмах лінійної регресії та випадкового лісу. Для розрахунку якості моделей були використані метрики MAE, MSE, RMSE та R^2.

Загалом результати роботи програми показали, що найбільший вплив на формування ціни автомобіля має параметр Power bph (потужність). Найкращі результати прогнозування ціни було отримано при використанні алгоритму випадкового лісу.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі буде проведено оцінювання основних характеристик для майбутнього програмного продукту, що спеціалізується на прогнозуванні ціни вживаних автомобілів на основі їх характеристик з використанням алгоритмів машинного навчання.

Дана реалізація сприятиме проведенню всіх необхідних досліджень і дозволить глибоко вивчити дане питання на прикладі автомобільного ринку Індії.

У представленому дослідженні відображені різноманітні підходи до реалізації, спрямовані на створення найбільш точної та оптимізованої стратегії вибору, яка враховує економічні фактори та сумісність з майбутнім програмним рішенням. Для цього використовувався метод функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість товару або послуги, незалежно від структури компанії. Головною метою ФВА є виявлення можливостей для зниження витрат, здійснюючи більш ефективні процеси виробництва та досягаючи оптимального балансу між споживчою вартістю продукту та витратами на його виготовлення. Для здійснення цього аналізу використовуються дані економічного, технічного та конструкторського характеру.

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 прогнозування ціни автомобіля на основі його характеристик. 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 класифікації.

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 програмного продукту

Головна функція F_0 – розробка програмного продукту, який вирішує задачу прогнозування ціни автомобіля на основі його характеристик. Беручи за основу цю функцію, можна виділити наступні:

F_1 – вибір мови програмування.

F_2 – вибір середовища програмування.

F_3 – вибір фреймворку машинного навчання.

Кожна з цих функцій має декілька варіантів реалізації:

Функція F_1:

а) Python.

б) R.

Функція F_2.

а) Kaggle (Google Colab).

б) PyCharm.

Функція F_3:

а) Scikit-learn (sklearn).

б) Statsmodels.

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

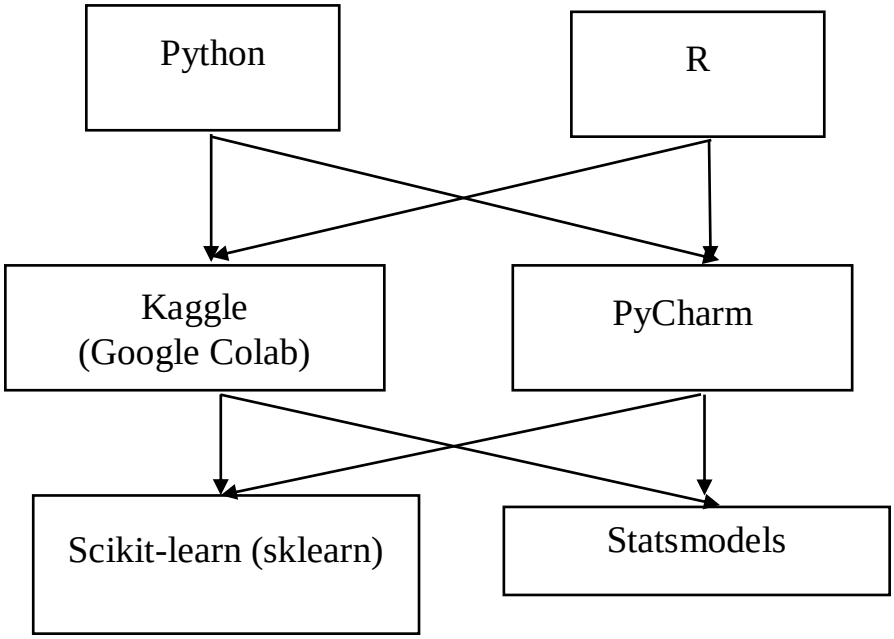

Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіантів основних функцій. Позитивно-негативна матриця показана в таблиці 4.1.

Таблиця 4.1.

Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	*А*	Зручність, багатофункціональність, широкий вибір бібліотек	Швидкодія, обмежена підтримка апаратного прискорення
	Б	Має величезні можливості для статистичного моделювання і аналізу даних	Володіє обмеженими можливостями для розробки загального призначення
F_2	*А*	Інтерактивність, можливість візуалізації даних та результатів,	Потребує багато ресурсів, обмежена підтримка великих обсягів

		зручність	даних
	Б	Підтримка автодоповнення, інструменти для оптимізації, робота з великими проектами	Висока ціна платної версії, проблеми з налаштуванням деяких бібліотек
F_3	*А*	Надає різноманітні алгоритми для навчання з учителем і без нього в зручному та єдиному форматі	Не підтримує глибоке навчання в повній мірі, особливо щодо моделей нейронних мереж
	Б	Надає детальні результати для кожної моделі, що дозволяє виконувати глибокий статистичний аналіз	Не надає широкого спектра алгоритмів машинного навчання, які доступні в інших бібліотеках, таких як scikit-learn

На основі аналізу позитивно-негативної матриці 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:

Перевагу даємо зручності, багатофункціональності, широкому вибору бібліотек. Для спрощення роботи по написанню коду варіант Б має бути відкинутий.

Функція F_2:

Програма допускає обрання обох варіантів. Можливо використати варіанти А чи Б.

Функція F_3:

Реалізація першого варіанту є сприйнятливою для програми. Це варіант А.

Таким чином, будемо розглядати такий варіанти реалізації ПП:

F_1а – F_2а – F_3а

F_1а – F_2*б* – F_3а

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.3 Обґрунтування системи параметрів програмного продукту

На основі даних, розглянутих вище, визначаються основні параметри вибору, які будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- *X1* – швидкодія мови програмування;
- *X2* – об’єм пам’яті для обчислень та збереження даних;
- *X3* – час навчання даних;
- *X4* – потенційний об’єм програмного коду.

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію програмного продукту, як показано у таблиці 4.2.

Таблиця 4.2 - Основні параметри програмного продукту

Назва Параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	X1	оп/мс	6000	10000	14000
Об’єм пам’яті	X2	Мб	256	128	64
Час попередньої обробки даних	X3	мс	30000	25000	20000
Потенційний об’єм програмного коду	X4	кількість рядків коду	800	600	400

За даними таблиці 4.2 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5.

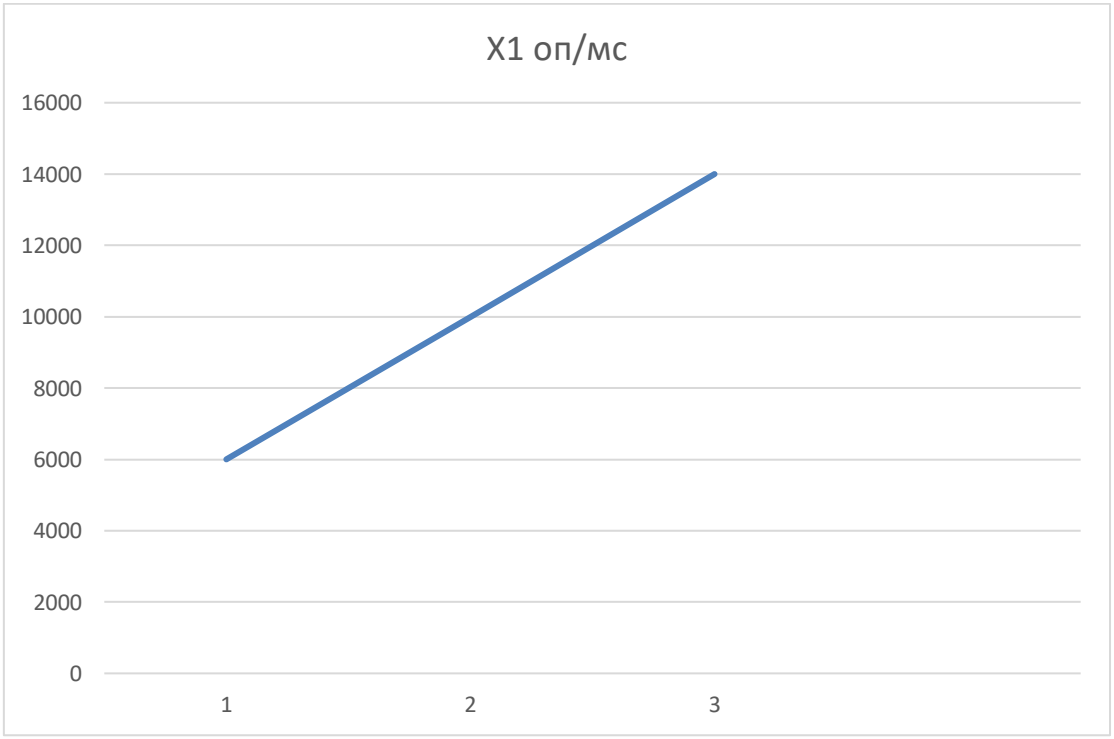

Рисунок 4.2 – X1, швидкодія мови програмування

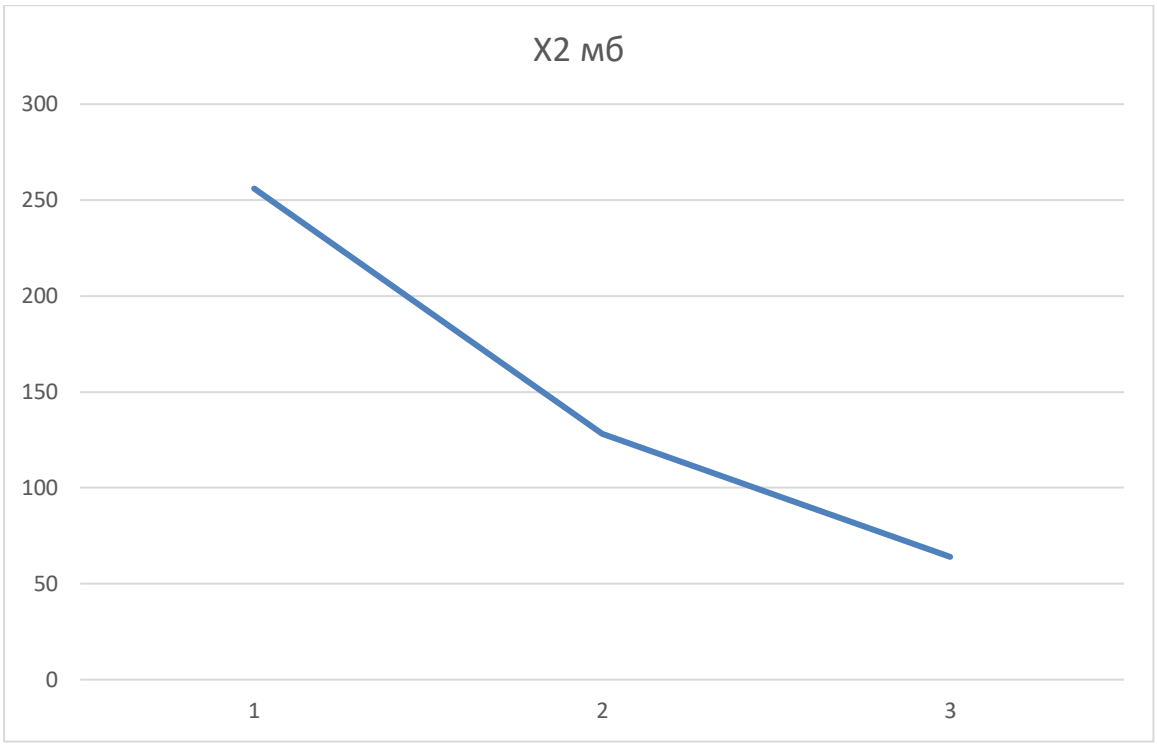

Рисунок 4.3 – X2, об’єм пам’яті

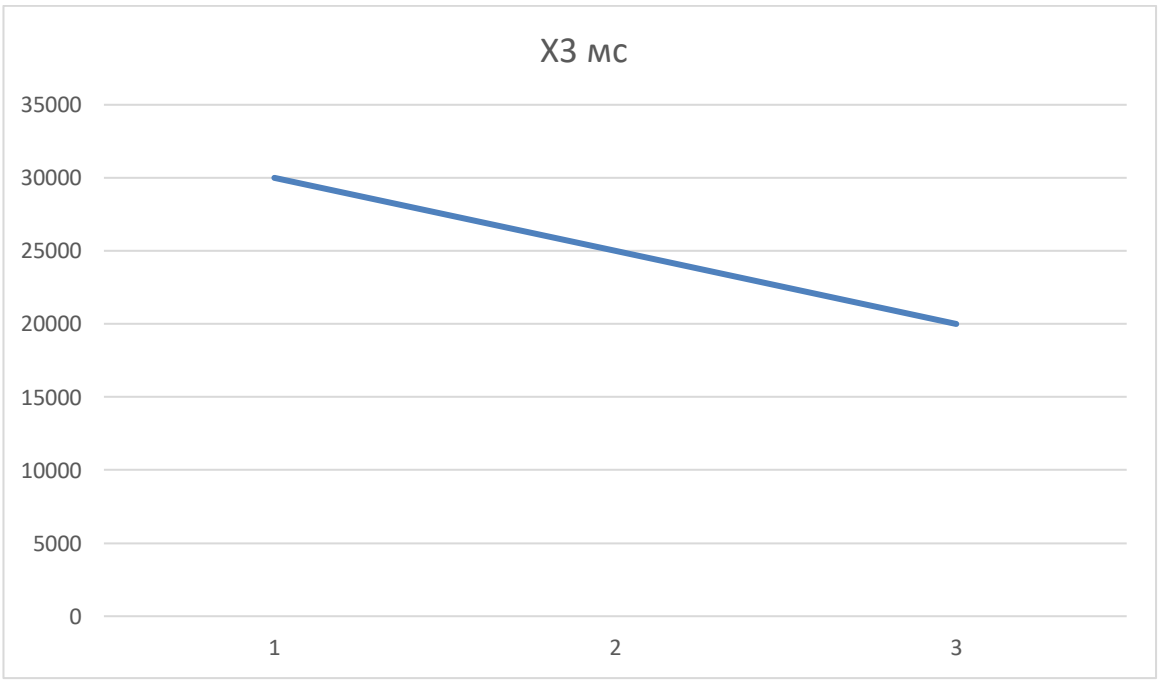

Рисунок 4.4 – X3, час попередньої обробки даних

Рисунок 4.5 – X4, потенційний об’єм програмного коду

4.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;

– перевірку придатності експертних оцінок для подальшого використання;

– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 - Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхи-лення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	2	1	1	2	1	1	3	11	-6,5	42,25
X2	Об’єм пам’яті	Mb	4	4	5	4	3	5	3	28	10,5	110,25
X3	Час попередньої обробки даних	мс	1	2	2	1	2	1	1	10	-7,5	56,25
X4	Потенційний об’єм програмного коду	Кількість рядків коду	3	3	2	3	4	3	3	21	3,5	12,25
	Разом		10	10	10	10	10	10	10	70	0	221

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70 \quad (4.1)$$

де N – число експертів,

n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17{,}5 \quad (4.2)$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T. \quad (4.3)$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 221. \quad (4.4)$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{10S}{N^2(n^3 - n)} = \frac{10 \cdot 221}{7^2(4^3 - 4)} = 0{,}752 > W_k = 0{,}67. \quad (4.5)$$

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 параметрів.

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

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

$$a_{ij} = \begin{cases} 1.5 \; при \; X_i > X_j \\ 1.0 \; при \; X_i = X_j \\ 0.5 \; при \; X_i < X_j \end{cases}. \quad (4.6)$$

З отриманих числових оцінок переваги складемо матрицю A=$\| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{ві}$ за наступними формулами:

$$K_{ві} = \frac{b_i}{\sum_{i=1}^{n} b_i} \qquad (4.7)$$

$$b_i = \sum_{i=1}^{N} a_{ij} \qquad (4.8)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{ві} = \frac{b_i^{'}}{\sum_{i=1}^{n} b_i^{'}}, \qquad (4.9)$$

$$b_i^{'} = \sum_{i=1}^{N} a_{ij} b_j \qquad (4.10)$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 - Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{вi}$	b_i^1	$K_{вi}^1$	b_i^2	$K_{вi}^2$
X1	1	0,5	1	0,5	3	0,19	11	0,19	40,75	0,185
X2	1,5	1	1,5	1,5	5,5	0,34	21,25	0,35	78,625	0,36
X3	1	0,5	1	0,5	3	0,19	11	0,19	40,75	0,185
X4	1,5	0,5	1,5	1	4,5	0,28	16,25	0,27	59,875	0,27
Всього:					16	1	59,5	1	220	1

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2* (Об'єм пам'яті), *X3* (час попередньої обробки даних) та *X4* (потенційний об'єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X1* (швидкість роботи мови програмування) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j} \qquad (4.11)$$

де n – кількість параметрів;

$K_{вi}$– коефіцієнт вагомості i–го параметра;

B_i – оцінка i–го параметра в балах.

Таблиця 4.6 - Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	А	X1	10000	21	0,185	3,885
F3	А	X2	64	17	0,36	6,12
	Б	X3	128	15	0,185	2,775
F4	А	X4	400	16	0,27	4,32

За даними з таблиці 4.6 за формулою:

$$K_K = K_{TY}[F_{1k}] + K_{TY}[F_{2k}] + \ldots + K_{TY}[F_{zk}] \quad (4.12)$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 3{,}885 + 6{,}12 + 4{,}32 = 14{,}325 ;$$

$$K_{K2} = 3{,}885 + 2{,}775 + 4{,}32 = 10{,}98 .$$

Як видно з розрахунків, кращим є 1 варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється як:

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М} \quad (4.13)$$

де T_P – трудомісткість розробки ПП;

$K_П$ – поправочний коефіцієнт;

$K_{СК}$ – коефіцієнт на складність вхідної інформації;

K_M – коефіцієнт рівня мови програмування;

$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: T_P =35 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П$ = 1.6. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.8. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 35 \cdot 1.6 \cdot 0.8 = 44{,}8 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто T_P =27 людино-днів, $К_П$ =1.4, $К_{СК}$ = 1, $К_{СТ}$ =0.9:

$$T_2 = 27 \cdot 1.4 \cdot 0.9 = 34{,}02 \text{ людино-днів.}$$

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

$$T_I = (44{,}8 + 34.02 + 5.4 + 34.02) \cdot 8 = 945{,}92 \quad \text{людино-годин.}$$

$$T_{II} = (44{,}8 + 34.02 + 7.22 + 34.02) \cdot 8 = 960{,}48 \quad \text{людино-годин.}$$

Найбільш високу трудомісткість має варіант II.

У розробці беруть участь два програмісти з окладом 27675 грн., один аналітик в області даних з окладом 31365 грн. Визначимо середню зарплату за годину за формулою:

$$С_{ч} = \frac{М}{T_m \cdot t} грн. \quad (4.14)$$

де М – місячний оклад працівників;

T_m – кількість робочих днів тиждень;

t – кількість робочих годин в день.

$$С_{ч} = \frac{27675 + 27675 + 31365}{3 \cdot 21 \cdot 8} = 172{,}05 \ \ грн. \quad (4.15)$$

Тоді, розрахуємо заробітну плату за формулою:

$$С_{ЗП} = C_ч \cdot T_i \cdot К_Д \quad (4.16)$$

де $С_ч$– величина погодинної оплати праці програміста;

T_i – трудомісткість відповідного завдання;

$К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП} = 172{,}05 \cdot 945{,}92 \cdot 1.2 = 195294{,}64$ грн.

II. $С_{ЗП} = 172{,}05 \cdot 960{,}48 \cdot 1.2 = 198300{,}701$ грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 195294{,}64 \cdot 0.22 = 42964{,}82$ грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22 = 198300,701 \cdot 0.22 = 43626,15$ грн.

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

Так як одна ЕОМ обслуговує одного програміста з окладом 27675 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 27675 \cdot 0,2 = 66420 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$С_{ЗП} = С_Г \cdot (1 + К_З) = 66420 \cdot (1 + 0.2) = 79704 \text{ грн.}$$

Відрахування на соціальний внесок:

$$С_{ВІД} = С_{ЗП} \cdot 0.22 = 79704 \cdot 0,22 = 17534,88 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 28500 грн.

$$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.2 \cdot 0.25 \cdot 28500 = 8550 \text{ грн.},$$

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$– річна норма амортизації;

$Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.2 \cdot 28500 \cdot 0.05 = 1710 \text{ грн.},$$

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{ЕФ} =(Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 8 – 11) \cdot 8 \cdot 0.7 = \\ = 1355,2 \text{ години},$$

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1355,2 \cdot 0,5 \cdot 0,2 \cdot 4,86 = 658,62 \text{ грн.},$$

де $N_С$ – середньо-споживча потужність приладу;

$К_З$– коефіцієнтом зайнятості приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_Н = Ц_{ПР} \cdot 0.67 = 28500 \cdot 0,67 = 19095 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н \quad (4.17)$$

$$C_{ЕКС} = 79704 + 17534{,}88 + 8550 + 1710 + 658{,}62 + 19095 = 127252{,}5 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$C_{М\text{-}Г} = C_{ЕКС} / T_{ЕФ} = 127252{,}5 \ / \ 1355{,}2 = 93{,}9 \text{ грн/год.}$$

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$C_М = C_{М-Г} \cdot T \quad (4.18)$$

I. $C_М = 93{,}9 \cdot 945{,}92 = 88821{,}89$ грн.

II. $C_М = 93{,}9 \cdot 960{,}48 = 90189{,}07$ грн.

Накладні витрати складають 67% від заробітної плати:

$$C_Н = C_{ЗП} \cdot 0{,}67 \quad (4.19)$$

I. $C_Н = 195294{,}64 \cdot 0{,}67 = 130847{,}4$ грн.

II. $C_Н = 198300{,}701 \cdot 0{,}67 = 132861{,}47$ грн.

Отже, вартість розробки ПП за варіантами становить:

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_М + C_Н \quad (4.20)$$

I. $С_{ПП}$ = 195294,64 + 42964,82 + 88821,89 + 130847,4 = 457928,75 грн.

II. $С_{ПП}$ = 198300,701 + 43626,15 + 90189,07 + 132861,47 = 475977,391 грн.

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕР_j} = К_{Кj} / С_{Фj} \qquad (4.21)$$

$$К_{ТЕР1} = 14,325 / 457928,75 = 3,129 \cdot 10^{-5},$$

$$К_{ТЕР2} = 10,98 / 475977,391 = 2,3 \cdot 10^{-5}.$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1}$= 3,129 $\cdot 10^{-5}$.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР}$ = 3,129 $\cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

- Мова програмування – Python;
- Інтерактивність, можливість візуалізації даних та результатів, зручність
- Різноманітні алгоритми для навчання з учителем і без нього в зручному та єдиному форматі

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, швидку та зручну реалізацію програми, доступний функціонал для роботи.

4.8 Висновки до розділу 4

У даній частині було проведено повний функціонально-вартісний аналіз програмного продукту. Також було знайдено оцінку основних функцій програмного продукту.

У результаті виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, було визначено та проведено оцінку основних функцій програмного продукту, а також знайдено параметри, які його характеризують.

На основі аналізу вибрано варіант реалізації програмного продукту.

ВИСНОВКИ

У даній дипломній роботі було досліджено можливість використання методів штучного інтелекту та машинного навчання для прогнозування ціни вживаних автомобілів на основі аналізу їх характеристик.

Отримане дослідження демонструє, що з використанням сучасних технік машинного навчання, таких як лінійна регресія та випадковий ліс, можливо створити доволі точні прогнози вартості вживаних автомобілів. Найкращий результат прогнозу був отриманий при використанні алгоритму випадкового лісу.

Було показано, що аналіз характеристик автомобіля може бути корисним інструментом для прогнозування його вартості на ринку вживаних автомобілів. Дані характеристики включають такі фактори, як назва автомобіля, місцезнаходження, рік випуску, пробіг, тип палива, тип коробки передач, кількість власників, запас ходу(економічність), об'єм двигуна, потужність двигуна та кількість місць. Загалом отримано, що потужність двигуна автомобіля має найбільш вплив на формування ціни.

Також, важливим внеском цієї роботи є розробка практичного інструменту, який може бути використаний дилерами автомобілів, оцінювачами, страховими компаніями та індивідуальними покупцями для прогнозування вартості автомобіля на основі його характеристик.

У майбутньому, можливо, ці методи можна було б вдосконалити шляхом включення додаткових даних та використання більш складних моделей машинного навчання. Проте вже зараз дана робота вносить значний внесок у зрозуміння того, як технології штучного інтелекту можуть бути використані для прогнозування ціни вживаних автомобілів.

ВИКОРИСТАНА ЛІТЕРАТУРА

1. Ринок легкових автомобілів в Україні та світі: тип, структура, сучасний стан та перспективи розвитку. URL: http://dspace.wunu.edu.ua/bitstream/316497/21568/1/%D0%9A.%D1%80.%20%20%D0%9E%D0%BB%D0%B5%D0%BD%D0%B8%D1%87%20%D0%9E.%D0%90..pdf
2. Світовий ринок легкових автомобілів та глобальна економіка [Електронний ресурс] / О. П. Савич // Економіка: реалії часу. Науковий журнал. – 2016. – № 3 (25). – С. 144-149. – Режим доступу до журн.: http://economics.opu.ua/files/archive/2016/n3.html
3. Кількість автомобілів у місті Києві за останні 7 років. URL: https://www.epravda.com.ua/news/2022/02/11/682305/
4. Кількість проданих нових та вживаних автомобілів у США. URL: https://www.greencarcongress.com/2019/07/20190716-fotw.html
5. Статистичний аналіз. URL: https://www.wallstreetmojo.com/statistical-analysis/
6. Регресійний аналіз. URL: https://hbr.org/2015/11/a-refresher-on-regression-analysis
7. Машинне навчання. URL: https://nachasi.com/tech/2019/01/31/yak-pratsyuye-machine-learning/
8. Навчання з вчителем та без. URL: https://uk.myservername.com/types-machine-learning
9. Візуалізація рівняння лінійної регресії. URL: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-linear-regression/
10. Візуалізація рівняння поліноміальної регресії. URL: https://www.javatpoint.com/regression-analysis-in-machine-learning

11. Схема роботи алгоритму дерево рішень. URL: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm
12. Схема роботи алгоритму випадкового лісу. URL: https://www.datacamp.com/blog/top-machine-learning-use-cases-and-algorithms
13. Схема роботи алгоритму регресії опорних векторів (SVR). URL: https://www.javatpoint.com/regression-analysis-in-machine-learning
14. Реалізація алгоритму лінійної регресії. URL: https://www.machinelearningnuggets.com/python-linear-regression/
15. Реалізація алгоритму випадкового лісу. URL: https://medium.com/@theclickreader/random-forest-regression-explained-with-implementation-in-python-3dad88caf165
16. Метрики якості моделей для задач регресії. URL: https://www.analyticsvidhya.com/blog/2021/05/know-the-best-evaluation-metrics-for-your-regression-model/
17. Мова програмування Python. URL: https://www.javatpoint.com/python-tutorial
18. Огляд платформи реалізації програмного продукту Google Colab. URL: https://www.tutorialspoint.com/google_colab/what_is_google_colab.htm

ДОДАТОК А ЛІСТИНГ КОДУ

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error

import warnings
warnings.filterwarnings("ignore")
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', -1)
pd.set_option('display.float_format', '{:.2f}'.format)

df = pd.read_csv("/content/used_cars_data.csv")
df.sample(5)

df.info()

df["Mileage"] = df["Mileage"].str.replace(r"([kmpl/g])",
"").astype("float")
df["Power"] = df["Power"].replace("null bhp", 0)
df["Power"] = df["Power"].str.replace(r"([ bhp])", "").astype("float")

mean_power = df[df["Power"] != 0]
```

```
df["Power"] = df["Power"].replace(0, mean_power["Power"].mean())
df["Power"].fillna(mean_power["Power"].mean(), inplace = True)
df["New_Price"] = df["New_Price"].str.replace(r"([LakhCr])",
"").astype("float")
df["New_Price"].fillna(0, inplace = True)
df["Seats"].fillna(df["Seats"].mean(), inplace = True)
df["Engine"] = df["Engine"].str.replace(r"([CC])", "").astype("float")
df["Engine"].fillna(df["Engine"].mean(), inplace = True)
df = df.drop(columns = ["New_Price"])
df = df[df['Price'].notna()]
df = df.rename(columns = {"Mileage": "Mileage kmpl/kg",
                          "Engine": "Engine CC",
                          "Power": "Power bph"})

df.info()
df.describe()

plt.figure(figsize = (18, 7))
correlation = df.corr()
sns.heatmap(correlation, annot = True)
plt.title("Матриця кореляції")
plt.show()
plt.close()
plt.show()

cars = df["Name"].value_counts().to_frame().reset_index().head(10)
cars.columns = ["Cars", "Frequency"]
cars

plt.figure(figsize = (18, 7))
graph = sns.barplot(x = "Cars", y = "Frequency", data = cars, palette
= "prism")
for p in graph.patches:
  graph.annotate('{:.01f}'.format(p.get_height()),
                 (p.get_x()+0.41, p.get_height()),
```

```
                ha='center', va='bottom', color='black', size = 15)
plt.title("Топ-10 автомобілів найбільш поширених у датасеті", size =
15)
plt.xticks(size = 15, rotation = 90)
plt.yticks(size = 15)
plt.xlabel(None)
plt.ylabel(None)
plt.show()

city = df["Location"].value_counts().reset_index()
city.columns = ["City", "Cars"]
city

plt.figure(figsize = (18, 7))
graph = sns.barplot(x="City", y = "Cars", data = city, palette =
"plasma")
for p in graph.patches:
  graph.annotate('{:.01f}'.format(p.get_height()),
                 (p.get_x()+0.41, p.get_height()),
                ha='center', va='bottom', color='black', size = 15)
plt.title("Кількість проданих автомобілів за містами Індії", size =
15)
plt.xticks(size = 15, rotation = 90)
plt.yticks(size = 15)
plt.xlabel(None)
plt.ylabel(None)
plt.show()



years = df["Year"].value_counts().to_frame().reset_index()
years.columns = ["Years", "Cars"]
years

plt.figure(figsize = (18, 7))
```

```
graph = sns.barplot(x = "Years", y = "Cars", data = years, palette =
"prism_r")
for p in graph.patches:
  graph.annotate('{:.0f}'.format(p.get_height()),
                 (p.get_x()+0.41, p.get_height()),
                ha='center', va='bottom', color='black', size = 15)
plt.title("Кількість проданих автомобілів за роком виробництва", size
= 15)
plt.xticks(size = 15, rotation = 90)
plt.yticks(size = 15)
plt.xlabel(None)
plt.ylabel(None)
plt.show()

Kilometers_Driven =
df["Kilometers_Driven"].value_counts().to_frame().reset_index().head(1
0)
Kilometers_Driven.columns = ["Kilometers Drive", "Cars"]
Kilometers_Driven

plt.figure(figsize = (18, 7))
graph = sns.barplot(x = "Kilometers Drive", y = "Cars", data =
Kilometers_Driven, palette = "prism_r")
for p in graph.patches:
  graph.annotate('{:.0f}'.format(p.get_height()),
                 (p.get_x()+0.41, p.get_height()),
                ha='center', va='bottom', color='black', size = 15)
plt.title("Кількість проданих автомобілів за пробігом", size = 15)
plt.xticks(size = 15)
plt.yticks(size = 15)
plt.xlabel("Kilometers", size = 15)
plt.ylabel("Cars", size = 15)
plt.show()

fuel_cars = df["Fuel_Type"].value_counts().to_frame().reset_index()
```

```
fuel_cars.columns = ["Fuel Type", "Cars"]
fuel_cars

plt.figure(figsize = (18, 7))
graph = sns.barplot(x = "Fuel Type", y = "Cars", data = fuel_cars)
for p in graph.patches:
  graph.annotate('{:.0f}'.format(p.get_height()),
                 (p.get_x()+0.41, p.get_height()),
                 ha='center', va='bottom', color='black', size = 15)
plt.title("Кількість проданих автомобілів за типом палива", size = 15)
plt.xticks(size = 15)
plt.yticks(size = 15)
plt.xlabel(None)
plt.ylabel(None)
plt.show()

transmission_cars =
df["Transmission"].value_counts().to_frame().reset_index()
transmission_cars.columns = ["Transmission Type", "Cars"]
transmission_cars

plt.figure(figsize = (18, 7))
graph = sns.barplot(x = "Transmission Type", y = "Cars", data =
transmission_cars, palette = "tab20b")
for p in graph.patches:
  graph.annotate('{:.0f}'.format(p.get_height()),
                 (p.get_x()+0.41, p.get_height()),
                 ha='center', va='bottom', color='black', size = 15)
plt.title("Кількість проданих автомобілів за типом коробки передач",
size = 15)
plt.xticks(size = 15)
plt.yticks(size = 15)
plt.xlabel(None)
plt.ylabel(None)
plt.show()
```

```
engine = df["Engine
CC"].value_counts().to_frame().reset_index().head(10)
engine.columns = ["Engine CC", "Cars"]
engine

plt.figure(figsize = (18, 7))
graph = sns.barplot(x = "Engine CC", y = "Cars", data = engine,
palette = "gist_earth_r")
for p in graph.patches:
  graph.annotate('{:.0f}'.format(p.get_height()),
                 (p.get_x()+0.41, p.get_height()),
                 ha='center', va='bottom', color='black', size = 15)
plt.title("Найбільша кількість проданих автомобілів за об'ємом
двигуна", size = 15)
plt.xticks(size = 15)
plt.yticks(size = 15)
plt.xlabel("Engine CC", size = 15)
plt.ylabel("Cars", size = 15)
plt.show()

power = df["Power bph"].value_counts().to_frame().reset_index()
power.columns = ["Power bph", "Cars"]
power = power.head(10)
power

plt.figure(figsize = (18, 7))
graph = sns.barplot(x = "Power bph", y = "Cars", data = power, palette
= "gnuplot")
for p in graph.patches:
  graph.annotate('{:.0f}'.format(p.get_height()),
                 (p.get_x()+0.41, p.get_height()),
                 ha='center', va='bottom', color='black', size = 15)
plt.title("Найбільша кількість проданих автомобілів за потужністю
двигуна", size = 15)
```

```
plt.xticks(size = 15, rotation = 90)
plt.yticks(size = 15)
plt.xlabel("Power bph", size = 15)
plt.ylabel("Cars", size = 15)
plt.show()

ce = df.groupby(["Engine CC",
"Name"])["Name"].agg(["count"]).reset_index().sort_values(by = "Engine
CC", ascending = False)
ce = ce.drop_duplicates("Name", keep = "first")
ce.columns = ["Engine CC", "Car Model", "Cars"]
ce = ce.head(10)
ce

plt.figure(figsize = (18, 7))
graph = sns.barplot(x = "Engine CC", y = "Car Model", data = ce,
palette = "cool_r")
for p in graph.patches:
  graph.annotate('{:.0f}'.format(p.get_height()),
                 (p.get_x()+0.05, p.get_height()),
                 ha='center', va='bottom', color='black', size = 15)
plt.title("Продані автомобілі з найбільшим об'ємом двигуна", size =
15)
plt.xticks(size = 15, rotation = 90)
plt.yticks(size = 15)
plt.xlabel(None)
plt.ylabel(None)
plt.show()

cp = df.groupby(["Power bph",
"Name"])["Name"].agg(["count"]).reset_index().sort_values(by = "Power
bph", ascending = False)
cp = cp.drop_duplicates("Name", keep = "first")
cp.columns = ["Power bph", "Car Model", "Cars"]
cp = cp.head(10)
```

```
cp

plt.figure(figsize = (18, 7))
graph = sns.barplot(x = "Power bph", y = "Car Model", data = cp,
palette = "hsv")
for p in graph.patches:
  graph.annotate('{:.0f}'.format(p.get_height()),
                 (p.get_x()+0.05, p.get_height()),
                 ha='center', va='bottom', color='black', size = 15)
plt.title("Продані автомобілі з найбільшою потужністю двигуна", size =
15)
plt.xticks(size = 15, rotation = 90)
plt.yticks(size = 15)
plt.xlabel(None)
plt.ylabel(None)
plt.show()


fp = df.groupby("Fuel_Type")["Power bph"].agg(["min", "mean",
"max"]).reset_index()
fp.columns = ["Fuel", "Min Power bph", "Avg Power bph", "Max Power
bph"]
fp

fp = fp.melt("Fuel", var_name = "Power", value_name = "bph")
plt.figure(figsize = (18, 7))
graph = sns.barplot(x = "Fuel", y = "bph", hue = "Power", data = fp,
palette = "rainbow")
for p in graph.patches:
  graph.annotate('{:.01f}'.format(p.get_height()),
                 (p.get_x()+0.15, p.get_height()),
                 ha='center', va='bottom', color='black', size = 15)
plt.title("Потужність двигуна проданих автомоблів у залежності від
типа палива", size = 15)
plt.xticks(size = 15)
```

```
plt.yticks(size = 15)
plt.xlabel(None)
plt.ylabel(None)
plt.show()

ty = df.groupby(["Year",
"Transmission"])["Transmission"].agg(["count"]).reset_index()
ty.columns = ["Year", "Transmission", "Cars"]
ty.sort_values(by = "Cars")

plt.figure(figsize = (18, 7))
graph = sns.barplot(x = "Year", y = "Cars", hue = "Transmission", data
= ty, palette = "coolwarm_r")
for p in graph.patches:
  graph.annotate('{:.01f}'.format(p.get_height()),
                 (p.get_x()+0.2, p.get_height()),
                 ha='center', va='bottom', color='black', size = 8)
plt.title("Співвідношення проданих автомобілів за роком виробництва з
механічної та автоматичною коробкою передач", size = 15)
plt.xticks(size = 15)
plt.yticks(size = 15)
plt.xlabel(None)
plt.ylabel(None)
plt.show()

tm = df.groupby("Transmission")["Mileage kmpl/kg"].agg(["count",
"sum", "min", "mean", "max"]).reset_index()
tm.columns = ["Transmission", "Cars", "Total Mileage kmpl/kg", "Min
Mileage kmpl/kg", "Avg Mileage kmpl/kg", "Max Mileage kmpl/kg"]
tm

tm = tm[["Transmission", "Avg Mileage kmpl/kg", "Max Mileage
kmpl/kg"]]
tm = tm.melt("Transmission", var_name = "Mileage", value_name =
"kmpl/kg")
```

```
plt.figure(figsize = (18, 7))
graph = sns.barplot(x = "Transmission", y = "kmpl/kg", hue =
"Mileage", data = tm, palette = "icefire")
for p in graph.patches:
  graph.annotate('{:.01f}'.format(p.get_height()),
                 (p.get_x()+0.2, p.get_height()),
                 ha='center', va='bottom', color='black', size = 15)
plt.title("Запас ходу автомобіля в залежності від типу коробки
передач", size = 15)
plt.xticks(size = 15)
plt.yticks(size = 15)
plt.xlabel(None)
plt.ylabel(None)
plt.show()

df.head()

X = df[["Year"]]
y = df["Price"]

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size =
0.3, random_state = 50)

standardScaler = StandardScaler()
standardScaler.fit(x_train)
x_train = standardScaler.transform(x_train)
x_test = standardScaler.transform(x_test)

lr = LinearRegression()

lr.fit(x_train, y_train)

y_hat = lr.predict(x_test)

pd.DataFrame({"Actual": y_test, "Predicted": y_hat}).head()
```

```
plt.figure(figsize=(18, 7))
ax1 = sns.distplot(y_hat, hist=False, label='Predicted Values')
sns.distplot(y, hist=False, label='Actual Values', ax=ax1)
plt.legend()
plt.show()
plt.close()
plt.show()

print("Mean Squared Error = ", mean_squared_error(y_test, y_hat,
squared = True))
print("Root Mean Squared Error = ", mean_squared_error(y_test, y_hat,
squared = False))
print("Mean Absolute Error = ", mean_absolute_error(y_test, y_hat))
print("R-Squared: = ", r2_score(y_test, y_hat))

X = df[["Engine CC"]]
y = df["Price"]

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size =
0.3, random_state = 50)

standardScaler = StandardScaler()
standardScaler.fit(x_train)
x_train = standardScaler.transform(x_train)
x_test = standardScaler.transform(x_test)

lr = LinearRegression()

lr.fit(x_train, y_train)

y_hat = lr.predict(x_test)

pd.DataFrame({"Actual": y_test, "Predicted": y_hat}).head()
plt.figure(figsize=(18, 7))
ax1 = sns.distplot(y_hat, hist=False, label='Predicted Values')
```

```
sns.distplot(y, hist=False, label='Actual Values', ax=ax1)
plt.legend()
plt.show()
plt.close()
plt.show()

print("Mean Squared Error = ", mean_squared_error(y_test, y_hat,
squared = True))
print("Root Mean Squared Error = ", mean_squared_error(y_test, y_hat,
squared = False))
print("Mean Absolute Error = ", mean_absolute_error(y_test, y_hat))
print("R-Squared: = ", r2_score(y_test, y_hat))

X = df[["Power bph"]]
y = df["Price"]

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size =
0.3, random_state = 50)

standardScaler = StandardScaler()
standardScaler.fit(x_train)
x_train = standardScaler.transform(x_train)
x_test = standardScaler.transform(x_test)

lr = LinearRegression()

lr.fit(x_train, y_train)

y_hat = lr.predict(x_test)

pd.DataFrame({"Actual": y_test, "Predicted": y_hat}).head()

plt.figure(figsize=(18, 7))
ax1 = sns.distplot(y_hat, hist=False, label='Predicted Values')
sns.distplot(y, hist=False, label='Actual Values', ax=ax1)
```

```
plt.legend()
plt.show()
plt.close()
plt.show()

print("Mean Squared Error = ", mean_squared_error(y_test, y_hat,
squared = True))
print("Root Mean Squared Error = ", mean_squared_error(y_test, y_hat,
squared = False))
print("Mean Absolute Error = ", mean_absolute_error(y_test, y_hat))
print("R-Squared: = ", r2_score(y_test, y_hat))

X = df[["Seats"]]
y = df["Price"]

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size =
0.3, random_state = 50)

standardScaler = StandardScaler()
standardScaler.fit(x_train)
x_train = standardScaler.transform(x_train)
x_test = standardScaler.transform(x_test)

lr = LinearRegression()

lr.fit(x_train, y_train)

y_hat = lr.predict(x_test)

pd.DataFrame({"Actual": y_test, "Predicted": y_hat}).head()
plt.figure(figsize=(18, 7))
ax1 = sns.distplot(y_hat, hist=False, label='Predicted Values')
sns.distplot(y, hist=False, label='Actual Values', ax=ax1)
plt.legend()
plt.show()
```

```
plt.close()
plt.show()

print("Mean Squared Error = ", mean_squared_error(y_test, y_hat,
squared = True))
print("Root Mean Squared Error = ", mean_squared_error(y_test, y_hat,
squared = False))
print("Mean Absolute Error = ", mean_absolute_error(y_test, y_hat))
print("R-Squared: = ", r2_score(y_test, y_hat))

X = df[["Year", "Engine CC", "Power bph", "Seats"]]
y = df["Price"]

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size =
0.3, random_state = 50)

standardScaler = StandardScaler()
standardScaler.fit(x_train)
x_train = standardScaler.transform(x_train)
x_test = standardScaler.transform(x_test)

lr = LinearRegression()

lr.fit(x_train, y_train)

y_hat = lr.predict(x_test)

pd.DataFrame({"Actual": y_test, "Predicted": y_hat}).head()

plt.figure(figsize=(18, 7))
ax1 = sns.distplot(y_hat, hist=False, label='Predicted Values')
sns.distplot(y, hist=False, label='Actual Values', ax=ax1)
plt.legend()
plt.show()
plt.close()
```

```
plt.show()

print("Mean Squared Error = ", mean_squared_error(y_test, y_hat,
squared = True))
print("Root Mean Squared Error = ", mean_squared_error(y_test, y_hat,
squared = False))
print("Mean Absolute Error = ", mean_absolute_error(y_test, y_hat))
print("R-Squared: = ", r2_score(y_test, y_hat))

df.head()

X = df[["Year"]]
y = df["Price"]

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size =
0.3, random_state = 0)

standardScaler = StandardScaler()
standardScaler.fit(x_train)
x_train = standardScaler.transform(x_train)
x_test = standardScaler.transform(x_test)

rf = RandomForestRegressor(n_estimators = 100, random_state = 1)

rf.fit(x_train, y_train)

y_hat = rf.predict(x_test)

pd.DataFrame({"Actual": y_test, "Predicted": y_hat}).head()

plt.figure(figsize=(18, 7))
ax1 = sns.distplot(y_hat, hist=False, label='Predicted Values')
sns.distplot(y, hist=False, label='Actual Values', ax=ax1)
plt.legend()
plt.show()
```

```
plt.close()
plt.show()

print("Mean Squared Error = ", mean_squared_error(y_test, y_hat,
squared = True))
print("Root Mean Squared Error = ", mean_squared_error(y_test, y_hat,
squared = False))
print("Mean Absolute Error = ", mean_absolute_error(y_test, y_hat))
print("R-Squared: = ", r2_score(y_test, y_hat))

X = df[["Engine CC"]]
y = df["Price"]

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size =
0.3, random_state = 0)

standardScaler = StandardScaler()
standardScaler.fit(x_train)
x_train = standardScaler.transform(x_train)
x_test = standardScaler.transform(x_test)

rf = RandomForestRegressor(n_estimators = 100, random_state = 1)

rf.fit(x_train, y_train)

y_hat = rf.predict(x_test)

pd.DataFrame({"Actual": y_test, "Predicted": y_hat}).head()

plt.figure(figsize=(18, 7))
ax1 = sns.distplot(y_hat, hist=False, label='Predicted Values')
sns.distplot(y, hist=False, label='Actual Values', ax=ax1)
plt.legend()
plt.show()
plt.close()
```

```
plt.show()

print("Mean Squared Error = ", mean_squared_error(y_test, y_hat,
squared = True))
print("Root Mean Squared Error = ", mean_squared_error(y_test, y_hat,
squared = False))
print("Mean Absolute Error = ", mean_absolute_error(y_test, y_hat))
print("R-Squared: = ", r2_score(y_test, y_hat))

X = df[["Power bph"]]
y = df["Price"]

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size =
0.3, random_state = 0)

standardScaler = StandardScaler()
standardScaler.fit(x_train)
x_train = standardScaler.transform(x_train)
x_test = standardScaler.transform(x_test)

rf = RandomForestRegressor(n_estimators = 100, random_state = 1)

rf.fit(x_train, y_train)

y_hat = rf.predict(x_test)

pd.DataFrame({"Actual": y_test, "Predicted": y_hat}).head()
plt.figure(figsize=(18, 7))
ax1 = sns.distplot(y_hat, hist=False, label='Predicted Values')
sns.distplot(y, hist=False, label='Actual Values', ax=ax1)
plt.legend()
plt.show()
plt.close()
plt.show()
```

```
print("Mean Squared Error = ", mean_squared_error(y_test, y_hat,
squared = True))
print("Root Mean Squared Error = ", mean_squared_error(y_test, y_hat,
squared = False))
print("Mean Absolute Error = ", mean_absolute_error(y_test, y_hat))
print("R-Squared: = ", r2_score(y_test, y_hat))

X = df[["Seats"]]
y = df["Price"]

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size =
0.3, random_state = 0)

standardScaler = StandardScaler()
standardScaler.fit(x_train)
x_train = standardScaler.transform(x_train)
x_test = standardScaler.transform(x_test)

rf = RandomForestRegressor(n_estimators = 100, random_state = 1)

rf.fit(x_train, y_train)

y_hat = rf.predict(x_test)

pd.DataFrame({"Actual": y_test, "Predicted": y_hat}).head()

plt.figure(figsize=(18, 7))
ax1 = sns.distplot(y_hat, hist=False, label='Predicted Values')
sns.distplot(y, hist=False, label='Actual Values', ax=ax1)
plt.legend()
plt.show()
plt.close()
plt.show()
```

```
print("Mean Squared Error = ", mean_squared_error(y_test, y_hat,
squared = True))
print("Root Mean Squared Error = ", mean_squared_error(y_test, y_hat,
squared = False))
print("Mean Absolute Error = ", mean_absolute_error(y_test, y_hat))
print("R-Squared: = ", r2_score(y_test, y_hat))

X = df[["Year", "Engine CC", "Power bph", "Seats"]]
y = df["Price"]

x_train, x_test, y_train, y_test = train_test_split(X, y, test_size =
0.3, random_state = 0)

standardScaler = StandardScaler()
standardScaler.fit(x_train)
x_train = standardScaler.transform(x_train)
x_test = standardScaler.transform(x_test)

rf = RandomForestRegressor(n_estimators = 100, random_state = 1)

rf.fit(x_train, y_train)

y_hat = rf.predict(x_test)

pd.DataFrame({"Actual": y_test, "Predicted": y_hat}).head()

plt.figure(figsize=(18, 7))
ax1 = sns.distplot(y_hat, hist=False, label='Predicted Values')
sns.distplot(y, hist=False, label='Actual Values', ax=ax1)
plt.legend()
plt.show()
plt.close()
plt.show()
```

```
print("Mean Squared Error = ", mean_squared_error(y_test, y_hat,
squared = True))
print("Root Mean Squared Error = ", mean_squared_error(y_test, y_hat,
squared = False))
print("Mean Absolute Error = ", mean_absolute_error(y_test, y_hat))
print("R-Squared: = ", r2_score(y_test, y_hat))
```