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

Магістерська кваліфікаційна робота на тему:

«ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АНАЛІЗУ ТА ПЕРЕДБАЧЕННЯ ЦІН НА ВЖИВАНІ АВТОМОБІЛІ»

Виконав: студент 2 курсу, групи 2ІСТ-22м
Спеціальності 126 – «Інформаційні системи та технології»
_____ Владислав ГІЖЕВСЬКИЙ

Керівник: к.т.н., доц. каф. САІТ
_____ Сергій ЖУКОВ
«30» _____11_____ 2023 р.

Опонент: к.т.н., доц. каф. КН
_____ Валерій ДЕНИСЮК
«05» _____12_____ 2023 р.

Допущено до захисту

Завідувач кафедри САІТ
_____ д.т.н., проф. Віталій МОКІН
«01» _____12_____ 2023 р.

Вінниця ВНТУ – 2023 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій
Рівень вищої освіти – другий (магістерський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 126 Інформаційні системи та технології
Освітньо-професійна програма – Інформаційні технології аналізу даних та зображень

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
д.т.н., проф. Віталій МОКІН
«04» 09 2023 р.

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Гіжевському Владиславу Віталійовичу

1. Тема роботи: «Інформаційна технологія аналізу та передбачення цін на вживані автомобілі»,
керівник роботи: Сергій ЖУКОВ, к.т.н., доц. каф. САІТ,
затверджені наказом ВНТУ від «18» 09 2023 року №247
2. Термін подання студентом роботи «30» 11 2023 року
3. Вихідні дані до роботи:
Набір даних з платформи Kaggle по продажу вживаних автомобілів на американському веб-сайті Craigslist періодом до 2022 року;
4. Зміст текстової частини:
   1) Аналіз предметної області передбачення цін на вживані автомобілі;
   2) Огляд оптимальних інформаційних технологій та вхідного набору даних;
   3) Побудова моделей та передбачення цін на вживані автомобілі;
   4) Економічна частина.
5. Перелік ілюстративного матеріалу:
   - Машинне навчання та його складові;
   - Архітектура LightGBM;
   - Теплова карта відсутніх даних;
   - Тривимірна діаграма співвідношення ознак;
   - Інтерактивна мапа розташування автомобілів;
   - Кореляційна матриця ознак;
   - Графік точності моделей за коефіцієнтом детермінації;
   - Графік відносної похибки моделей;
   - Графік середньоквадратичної похибки моделей;
   - Результат передбачення ціни та різниця із справжньою.

6. Консультанти розділів МКР

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Наталія БУРЄННІКОВА, д.е.н., проф. каф. ЕПВМ	20.10.23	04.11.23

7. Дата видачі завдання «04» 09 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів МКР	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області передбачення цін на вживані автомобілі	04.09	10.09	
2	Огляд оптимальних інформаційних технологій та вхідного набору даних	10.09	01.10	
3	Побудова моделей та передбачення цін на вживані автомобілі	01.10	20.10	
4	Економічна частина	20.10	05.11	
5	Оформлення матеріалів до захисту МКР	05.11	30.11	

Студент Владислав ГІЖЕВСЬКИЙ

Керівник роботи Сергій ЖУКОВ

АНОТАЦІЯ

УДК 004.09

Гіжевський В. В. Інформаційна технологія аналізу та передбачення цін на вживані автомобілі. Магістерська кваліфікаційна робота зі спеціальності 126 – інформаційні системи та технології, освітньо-професійна програма – інформаційні технології аналізу даних та зображень. Вінниця: ВНТУ, 2023. 146 с.

На укр. мові. Бібліогр.: 55 назв; рис.: 103; табл.: 9.

В магістерській кваліфікаційній роботі проведено аналіз предметної області передбачення цін на вживані автомобілі, попередньо запропоновано ознаки, які мають вплив на ціноутворення вживаних автомобілів. Здійснено огляд аналогічних рішень, запропоновано алгоритм створення ІТ передбачення цін на вживані автомобілі, на основі якого проведено створення ІТ. Виконано вибір та опис набору даних, проведено попереднє очищення даних. Проведено розвідувальний аналіз даних, запропоновано правила фільтрації аномальних значень, обрано моделі машинного навчання, здійснено їх тренування та визначено оптимальну модель, точність якої – 0.92, що є більшим за 0.89, як у найкращого аналога ІТ.

Ілюстративна частина складається з 10 плакатів.

У розділі економічної частини піднято питання та дана відповідь про доцільність розроблення та впровадження інформаційної технології аналізу та передбачення цін на вживані автомобілі за даними США методами машинного навчання.

Ключові слова: вживані автомобілі, інформаційні технології, машинне навчання, аналіз даних, передбачення ціни, ознаки, моделі машинного навчання, передбачення цін.

ABSTRACT

Hizhevskyi V. V. Information technology for analysis and prediction of used car prices. Master's qualification work in the specialty 126 – information systems and technologies, educational and professional program – information technologies for data and image analysis. Vinnytsia: VNTU. 2023. 146 p.

In Ukrainian language. Bibliogr: 55 titles; Fig: 103; tables: 9.

In the master's qualification work, an analysis of the subject area of used car price prediction was carries out, and preliminary features that affect the pricing of used cars were proposed. A review of similar solutions was carried out, and an algorithm for creating IT for predicting the prices of used cars was proposed, on the basis of which IT was created. The selection and description of the data set has been made, and the preliminary cleaning of the data has been carried out. Exploratory data analysis was carried out, rules for filtering anomalous values were proposed, machine learning models were selected, they were trained and the optimal model was determined, the prediction accuracy of which is 0.92, which is higher than 0.89 in the best IT analogue.

The illustrative part consists of 10 posters.

In the economic part section, the issue of the feasibility of developing and implementing an information technology for the analysis and prediction of used car prices using machine learning methods based on US data is raised and answered.

Keywords: used cars, information technologies, machine learning, data analysis, price prediction, features, machine learning models, price prediction.

ЗМІСТ

ВСТУП

Актуальність теми. Сучасний світ перебуває в постійному русі, де інформаційні технології стають ключовими для вирішення різних завдань і завдяки ним створюються нові можливості в багатьох сферах. Однією зі сфер, де інформаційні технології здійснюють значний вплив, є ринок вживаних автомобілів. Підвищення доступності інформації в інтернеті, а також застосування аналітичних і алгоритмічних методів у сфері економіки та фінансів відкривають перед дослідниками та бізнесменами нові перспективи для аналізу та передбачення цін на вживані автомобілі.

Актуальність теми дослідження полягає в кількох ключових аспектах:

1. Зростання ринку вживаних автомобілів: Ринок вживаних автомобілів постійно росте, як в розвинених, так і в країнах що розвиваються. Це робить його важливим об'єктом дослідження та управління.

2. Зміна споживацьких звичок: Сучасні споживачі все більше віддають перевагу покупці вживаних автомобілів, через їхню доступність і менші витрати. Знання про ціни та передбачення їхньої динаміки стають критичними для покупців та продавців.

3. Зростання даних та обчислювальної потужності: Споживання та зберігання даних про ринок вживаних автомобілів значно зросло, що робить можливим застосування сучасних аналітичних методів для зрозуміння ринкової динаміки та передбачення цінових тенденцій.

4. Фінансовий аспект: Ринок вживаних автомобілів є значущою галуззю у фінансовому сенсі, і правильне управління цінами може призвести до збільшення прибутку та оптимізації ресурсів.

5. Споживчі переваги: Дослідження цінових тенденцій на ринку вживаних автомобілів може допомогти споживачам знайти оптимальні пропозиції, зберігаючи їхні фінанси та задоволення потреб в транспорті.

Зважаючи на ці аспекти, дослідження в галузі інформаційних технологій для аналізу та передбачення цін на вживані автомобілі має важливе значення і

може призвести до покращення ефективності на ринку та сприяти сталому розвитку автомобільної індустрії в цілому.

Зручність та користь від такої системи для покупців полягає у тому, що аналіз і передбачення цін на вживані автомобілі допомагають їм отримувати доступ до актуальної та об'єктивної інформації про ринок. Що дозволяє зробити осмислені рішення та знайти оптимальну ціну. Покупці можуть визначити, які моделі та марки автомобілів найбільше вигідні з точки зору ціни. Це допомагає їм зекономити більше грошей та вибрати автомобіль, який найкраще відповідає їхнім потребам. За допомогою аналізу цін покупці можуть знайти вживані автомобілі з найкращим співвідношенням ціни та якості. Це допомагає їм оптимізувати свої витрати на транспорт. Інформація про історію та попередніх власників автомобіля може бути важливою для покупця. Аналіз цін може включати такі дані, що сприяють більш відповідальному вибору.

Користь, яку дана система, принесете для продавця, повністю залежить від аналізу ринкових даних, що дозволить продавцям налаштовувати ціни на вживані автомобілі так, щоб вони були конкурентоспроможними і водночас максимізували прибуток. За допомогою аналітичних інструментів продавці зможуть швидко реагувати на зміни в ринковій ситуації, змінюючи ціни відповідно до попиту та пропозиції.

Мета та завдання роботи. Метою даної роботи є підвищення точності передбачення цін на вживані автомобілі за даними США методами машинного навчання шляхом створення інформаційної технології аналізу та передбачення цін на вживані автомобілі.

Для досягнення поставленої мети потрібно розв’язати такі задачі:

– аналіз інтелектуальних технологій та систем рекомендування цін;

– вибір оптимальних інформаційних технологій;

– передобробка датасету, огляд основних ознак та попереднє очищення даних;

– проведення розвідувального аналізу даних;

– вибір оптимальної моделі, створення інформаційної технології для аналізу та передбачення цін на вживані автомобілі.

Об’єктом дослідження магістерської кваліфікаційної роботи є процес створення інформаційної технології аналізу та передбачення цін на вживані автомобілі на ринку США.

Предметом дослідження магістерської кваліфікаційної роботи є методи машинного навчання та інформаційна технологія аналізу та передбачення цін на вживані автомобілі на ринку США.

Методи дослідження. У дослідженні використовувались наступні методи: розвідувальний аналіз даних, моделі машинного навчання, такі як Linear Regression, Support Vector Machines, Linear SVR, Stochastic Gradient Decent, Decision Tree Regressor, RandomForestRegressor, тощо.

У процесі виконання роботи проведено аналіз та передбачення цін на вживані автомобілі, використовуючи мову програмування Python у системі Kaggle.

Новизна одержаних результатів. Дістала подальший розвиток інформаційна технологія аналізу та передбачення цін вживаних автомобілів, за рахунок удосконалення передобробки даних, детальнішого розвідувального аналізу даних, та кращою оптимізацією гіперпараметрів моделей, що дозволило підвищити точність передбачення в порівнянні з аналогічними інформаційними технологіями.

Апробація результатів магістерської кваліфікаційної роботи. Результати роботи доповідались на LIII Всеукраїнській науково-технічній конференції факультету інтелектуальних інформаційних технологій та автоматизації ВНТУ (м. Вінниця, 2023-2024 рр.)**.**

Публікації. За тематикою дослідження опубліковано доповідь на тему «Інформаційна технологія аналізу та передбачення цін на вживані автомобілі» на LIII Науково-технічній конференції факультету інтелектуальних інформаційних технологій та автоматизації ВНТУ з публікацією тез (м. Вінниця, 2023-2024 рр.) [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ПЕРЕДБАЧЕННЯ ЦІН НА ВЖИВАНІ АВТОМОБІЛІ

1.1 Аналіз предметної області передбачення цін на вживані автомобілі

1.1.1 Зріст попиту на вживані автомобілі.

В останні роки спостерігається зростання попиту на вживані автомобілі. Цей тренд пов'язаний з кількома факторами, такими як економічні обмеження, ринкова насиченість і екологічні аспекти [2].

Вживані автомобілі є популярним вибором для багатьох людей. Вони пропонують більш доступну ціну, ніж нові автомобілі, і можуть бути хорошим варіантом для тих, хто шукає надійний і економічний транспортний засіб.

Визначення понять:

– Вживаний автомобіль – це автомобіль, який був у вжитку раніше, тобто, мав минулих власників та певний період експлуатації;

– Попит на вживані автомобілі – це кількість вживаних автомобілів, які потенційні покупці готові придбати за певну ціну.

Економічне обмеження є одним із основних факторів, що впливає на ріст попиту на вживані автомобілі. Внаслідок зростання цін на нові авто, споживачі все частіше починають шукати більш доступні варіанти автомобілів. Вживані автомобілі можуть бути набагато дешевшими за нові, що робить їх привабливішими для споживачів з обмеженим бюджетом.

Ринкова насиченість – інший фактор, що впливає на зріст попиту на вживані автомобілі. На даний момент і в подальшому ринок нових автомобілів багатьох країн світу перенасичений, через це ціни на нові автомобілі знижується, хоч і не дуже сильно. Це ж, в свою чергу, робить вживані автомобілі більш привабливими для придбання.

1.1.2 Зміни відношення між брендами та моделями

За даними Міжнародної асоціації виробників автомобілів (OICA), у 2022 році на вживані автомобілі припадало понад 20% світового ринку автомобілів [3]. Серед найбільш популярних брендів вживаних автомобілів у 2022 році були:

– Toyota;
– Honda;
– Volkswagen;
– Renault;
– Nissan.

Ці бренди мають довгу історію виробництва надійних і економічних автомобілів. Вони також мають розвинену мережу сервісних центрів, що забезпечує доступність запасних частин і обслуговування цих самих автомобілів.

Серед найбільш популярних моделей вживаних автомобілів у 2022 році були:

– Toyota Corolla;
– Honda Civic;
– Volkswagen Golf;
– Renault Megane;
– Nissan Qashqai.

Ці моделі є популярними у всьому світі. Вони пропонують хороший баланс між ціною, надійністю та комфортом.

Зростання попиту на вживані автомобілі певних брендів та моделей можна пояснити наступними факторами:

– Надійність. Вживані автомобілі цих брендів і моделей мають репутацію надійних і довговічних;

– Економічність. Вживані автомобілі цих брендів і моделей, як правило, економніші, в порівнянні з новими;

– Доступність. Ці бренди і моделі широко представлені на ринку, що робить їх більш доступними для покупців.

В 2023 році, аналітики компаній, надали дані, що на ринку вживаних автомобілів спостерігається зростання популярності таких брендів, як [4]:

– Toyota;
– Honda;
– Volkswagen;
– Mercedes-Benz;
– BMW.

Ці бренди відомі своїми надійністю та високою якістю. Вони також пропонують широкий вибір моделей, які відповідають потребам різних покупців.

Серед популярних моделей вживаних автомобілів у 2023 році можна виділити:

– Toyota Corolla;
– Honda Civic;
– Volkswagen Golf;
– Mercedes-Benz C-Class;
– BMW 3 Series.

І вже в даному випадку, на популярність брендів та моделей вживаних автомобілів впливають такі фактори, як:

– Надійність;
– Якість;
– Широкий вибір моделей;
– Економічність;
– Комфорт;
– Наявність опцій та комплектацій.

Зміни відношення між брендами та моделями на ринку вживаних автомобілів відбуваються під впливом різних факторів. Важливо враховувати ці фактори при виборі вживаного автомобіля, щоб отримати найбільш вигідну

угоду. А інформація про те, які автомобілі стали популярнішими та через які фактори, може бути дуже корисною інформацією.

1.1.3 Фактори впливу на ринкову цінову динаміку

На ціни вживаних автомобілів впливають такі економічні та фінансові фактори, як [5]:

- Ставка облікової ставки – підвищення ставки облікової ставки може призвести до зниження попиту на вживані автомобілі, оскільки вони стають більш дорогими в кредитуванні;
- Інфляція – інфляція може призвести до підвищення цін на вживані автомобілі, оскільки вони стають дорожчими в порівнянні з новими автомобілями;
- Рівень доходів населення – підвищення рівня доходів населення може призвести до зростання попиту на вживані автомобілі, оскільки люди мають більше коштів для їх придбання;
- Стан економіки – економічна криза може призвести до зниження цін на вживані автомобілі, оскільки люди мають менше коштів для їх придбання.

На ціни конкретних вживаних автомобілів також впливають такі фактори, як:

- Вік – автомобілі старшого віку, як правило, мають нижчу вартість, ніж автомобілі молодшого віку;
- Пробіг – автомобілі з малим пробігом, як правило, мають нижчу вартість, ніж автомобілі з великим пробігом;
- Технічний стан – автомобілі з хорошим технічним станом, як правило, мають більш високу вартість, ніж автомобілі з поганим технічним станом;
- Історія обслуговування – автомобілі з регулярною історією обслуговування, як правило, мають більш високу вартість, ніж автомобілі без регулярної історії обслуговування;

– Популярність моделі – автомобілі популярних моделей, як правило, мають більш високу вартість, ніж автомобілі непопулярних моделей.

Фактори, що впливають на ціни на вживані автомобілі, є складними і взаємопов'язаними. Для того, щоб точно прогнозувати зміни цін, необхідно враховувати всі ці фактори.

1.1.4 Вплив змін на ринку автомобільної індустрії

Одним з найважливіших трендів у автомобільній індустрії є перехід до електромобілів. Це пов'язано з кількома факторами, такими як зростаюча увага до екологічних питань, розвиток технологій та державні програми підтримки електромобілів [6].

Перехід до електромобілів може призвести до зростання цін на вживані автомобілі з традиційними двигунами. Це пов'язано з тим, що електромобілі, як правило, дорожчі за автомобілі з традиційними двигунами. Крім того, зростання попиту на електромобілі може призвести до зниження пропозиції вживаних автомобілів з традиційними двигунами.

Ще одним важливим трендом у автомобільній індустрії є впровадження нових технологій безпеки. Ці технології, такі як автономне водіння, системи попередження про зіткнення та системи контролю за станом водія, можуть зробити автомобілі більш безпечними.

Впровадження нових технологій безпеки може призвести до зростання цін на вживані автомобілі, які оснащені цими технологіями. Це пов'язано з тим, що ці технології, як правило, підвищують вартість автомобіля. Крім того, споживачі можуть бути готові платити більше за автомобілі, які оснащені безпечними технологіями.

Впровадження нових технологій може мати як позитивний, так і негативний вплив на цінові тенденції на ринку вживаних автомобілів.

З одного боку, впровадження нових технологій може призвести до зростання цін на вживані автомобілі, які оснащені цими технологіями. Це

пов'язано з тим, що ці технології, як правило, підвищують вартість автомобіля і роблять його більш привабливим для споживачів.

З іншого боку, впровадження нових технологій може призвести до зниження цін на вживані автомобілі, які не оснащені цими технологіями. Це пов'язано з тим, що ці автомобілі можуть стати менш привабливими для споживачів, які хочуть мати найсучасніші технології.

Поступальний розвиток ринку електронної комерції обумовлений появою нових веб-платформ, які спеціалізуються на продажі різноманітних товарів. Ці платформи приваблюють користувачів своєю доступністю, швидкістю та зручністю, оскільки вони пропонують широкий вибір товарів, починаючи від звичайних дрібниць до нерухомості.

У 2020 році обсяг світового ринку електронної комерції автомобілів становив 43,62 мільярда доларів США. Однак пандемія COVID-19 негативно вплинула на галузь, і ринок скоротився на 5,9%. Експерти прогнозують, що ринок зросте до 51,04 мільярда доларів США в 2021 році і до 202,94 мільярда доларів США до 2028 року, що відповідає CAGR 21,8%. Це зростання пояснюється попитом на автомобілі, який повернеться до допандемічного рівня після завершення пандемії [6].

CAGR – сукупний середньорічний темп зростання. Виражається у відсотках і показує, на скільки відсотків за рік приростає параметр, що вивчається.

Ринок електронної комерції – це сфера діяльності, пов'язана з купівлею та продажем товарів та послуг через Інтернет.

Прогнозовану динаміку ринку автомобілів та автозапчастин у США зображено на рисунку 1.1.

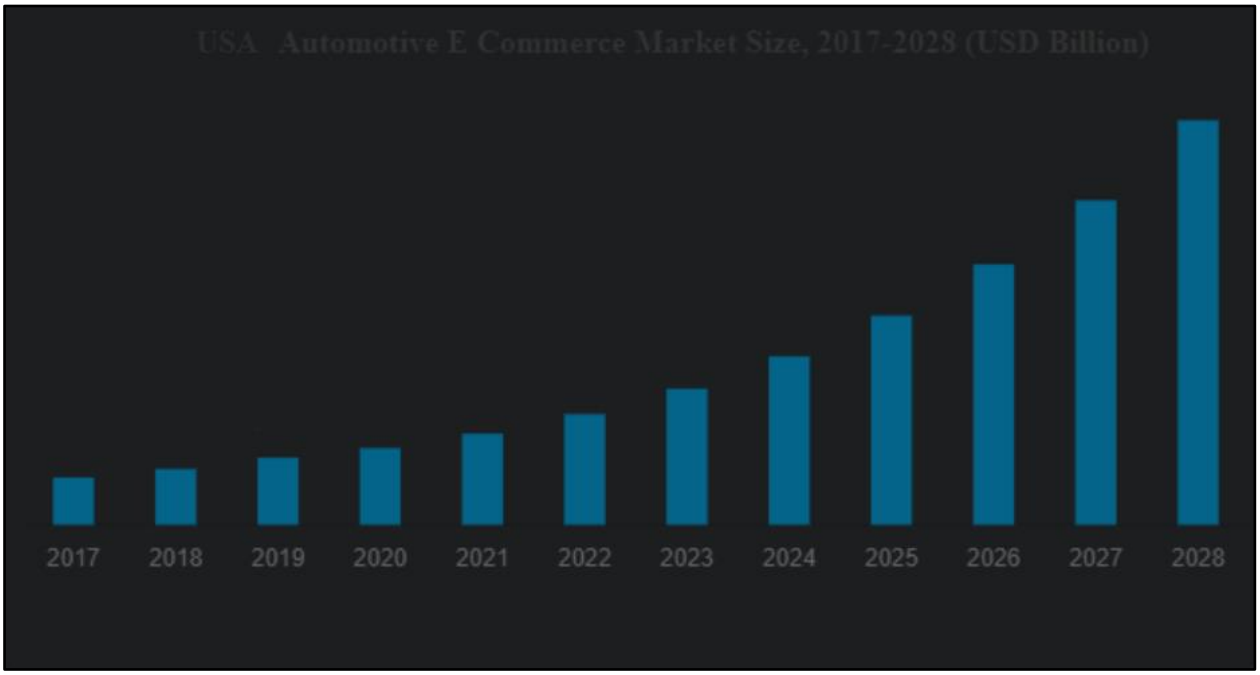

Рисунок 1.1 – Прогнозована динаміка ринку автомобілів та автозапчастин у США

Більшість виробників устаткування (OEM) стикаються з деякими проблемами в своїх операціях, такими як порушення в ланцюгу поставок, зниження імпорту та експорту автомобілів, а також обмеження на перевезення певних автомобілів. Ринок зростає завдяки глобальному зростанню електронної комерції, збільшенню продажів автомобілів та цифровізації інтерфейсів і каналів. Крім того, зростання доходу на душу населення та поширення Інтернету сприяли зростанню онлайн-продажів автомобілів і комплектуючих, що в кінцевому підсумку призвело до зростання ринку в прогнозований період [6].

OEM-виробники: це виробники, які проектують і виробляють устаткування для інших компаній.

Зростання світового ринку електронної комерції автомобілів також пояснюється збільшенням попиту на автомобільні компоненти та запчастини, оскільки старіючий автопарк стає все більш поширеним. Такі фактори, як більш прозорі ціни, зручний досвід покупок і більший вибір запчастин, спонукали клієнтів робити покупки в Інтернеті. Багато онлайн-пропозицій,

стратегії низьких цін і можливість порівняти деталі в Інтернеті за гарантією, ціною та технічними характеристиками ще більше стимулюють зростання ринку [2].

Серед найбільших компаній США, що займаються електронною комерцією автомобілів, виділяються наступні:

- O'Reilly Auto Parts (Міссурі, США)
- Amazon.com, Inc. (Вашингтон, США)
- AutoZone, Inc. (Теннессі, США)
- Advance Auto Parts (Північна Кароліна, США)
- eBay Inc. (Каліфорнія, США)
- Walmart (Арканзас, США)
- Craigslist (Каліфорнія, США)

Всі ці компанії мають веб-портали, які дозволяють покупцям і продавцям взаємодіяти між собою. Покупцям доступні різноманітні інструменти пошуку, а продавцям — зручні можливості для каталогізації та розміщення товарів.

Метою розробників таких веб-порталів є створення найкращого можливого користувацького досвіду. Вони прагнуть до того, щоб веб-сайти завантажувалися швидко, щоб користувачі могли легко знайти потрібні товари, а також щоб продавці могли легко додавати та оновлювати інформацію про свої товари.

Покращення користувацького досвіду призводить до збільшення продажів і кількості активних користувачів, що, в свою чергу, сприяє розвитку бізнесу.

Усі онлайн-покупці, після отримання результатів пошуку, проводять порівняння товарів самостійно. Однак, якщо покупець не має достатньої інформації про ціноутворення, він може зробити невигідну покупку, якщо натрапить на завищений цінник. Продавці також ризикують втратити прибуток, якщо неправильно встановлять ціну на свій товар.

Цей ризик особливо високий на вторинних ринках, де ціна товару залежить від багатьох факторів. Для вирішення цієї проблеми можна використовувати інтелектуальну систему аналізу та передбачення цін. Така система допоможе продавцям встановити справедливу ціну, а покупцям — оцінити її доцільність.

Розробка та впровадження цієї технології є важливою задачею для розвитку бізнес-порталів, що продають товари. Зокрема, вона може бути застосована на ринку автомобілів, де ціна залежить від багатьох факторів, таких як модель, рік випуску, стан та пробіг.

1.2 Переваги та недоліки купівлі вживаних автомобілів

Зважаючи на вигоди та обмеження, пов'язані з придбанням нового чи вживаного автомобіля, можливо, вам було б корисно визначити, яке рішення краще відповідає вашим потребам. Отже, давайте розглянемо переваги і недоліки придбання нового або вживаного автомобіля [7]:

Переваги придбання нового автомобіля:

– Найновіші технології: Залежно від обраного пакету, ви отримаєте доступ до передових технологій та інновацій.

– Надійність: Ваш новий автомобіль, як правило, вважається більш надійним, оскільки ви будете першим власником, і його надійність підтверджується заводськими гарантіями.

– Спеціальні умови: При купівлі нового автомобіля ви можете скористатися спеціальними знижками та пропозиціями від виробників.

Недоліки придбання нового автомобіля:

– Висока ціна: Вартість нового автомобіля зазвичай вища, ніж вартість вживаного еквівалента.

– Великий податок з продажу: При купівлі нового автомобіля ви маєте бути готовими до вищих податків з продажу.

– Втрата вартості: Нові автомобілі зазвичай втрачають до 20% своєї вартості миттєво після виїзду з автосалону.

Переваги придбання вживаного автомобіля [8]:

– Уникання великої втрати вартості: Придбавши вживаний автомобіль, ви уникаєте значної амортизації, яку проходять нові авто відразу після виходу з салону.

– Зменшені податки та страхові внески: Податки та страхові премії для вживаних моделей зазвичай нижчі.

– Доступ до преміальних функцій за меншу ціну: Ви все ще можете отримати преміальні функції за менше грошей.

Недоліки придбання вживаного автомобіля [9]:

– Немає гарантії щодо якості та надійності: Навіть з ретельним оглядом, якість та надійність вживаних автомобілів не гарантуються.

– Немає заводських гарантій для старших моделей: Якщо ваш вживаний автомобіль старший, він може бути поза періодом заводської гарантії.

– Обмежений вибір щодо особливостей та конфігурації: Придбавши вживаний автомобіль, ви маєте менше можливостей щодо вибору особливостей, таких як оббивка сидінь, колір кузова, тощо.

Отже, при виборі між новим та вживаним автомобілем важливо врахувати всі переваги та недоліки кожної опції, і зробити обдумане рішення, яке відповідає вашим потребам та обставинам. Нагадаємо, що вживані автомобілі мають своє місце на ринку та можуть бути вигідними в певних ситуаціях.

1.3 Огляд та аналіз існуючих програмних аналогів

Наразі не існує такої моделі, яка могла б точно спрогнозувати ціну на вживаний автомобіль. Основним завданням науковців, які займаються

дослідженням передбачення цін на автомобілі та інші товари, є визначення всіх факторів, які впливають на прогноз, але цього недостатньо.

Розглядаючи тему передбачення цін на вживані автомобілі, варто звертати увагу на теоретичні та практичні аспекти, які описуються в роботах дослідників, які також займаються передбаченням цін на вживані товари, зокрема, на автомобілі. Також варто використовувати веб-сервіси, які надають повну та статистичну інформацію про кожну угоду.

Перш за все, варто розглянути теоретичні та практичні аспекти в роботах інших дослідників, а лише потім тих, що використовуються на веб-сервісах. Адже тема передбачення цін на вживані автомобілі є досить популярною в різних країнах. В більшій частині країн йде активна робота ринків перепродажу вживаних автомобілів. Через незручність в очікуванні створення нових автомобілів, та через те, що ціни на нові автомобілі більші за вживані, перевага на придбання вживаних автомобілів зростає. Та навіть сам факт того, що автомобіль, який тільки виїхав з автосалону, вже вважається вживаним і дешевшає з нарощуванням значення пробігу досить швидко, одна подряпина, або якась вм’ятина сильно зменшую ціну на досить новий автомобіль. Варто пам’ятати, що є ті, хто полюбляє раритетні автомобілі, їх реставрує і часто, в подальшому, перепродає. Звідси випливає, що передбачення цін на вживані автомобілі, це насправді цікава тема, яка з кожним роком стає все більше і більше актуальнішою, та може мати багато цікавих факторів та ситуацій, які можуть відкритись при дослідженні роботи ринків цього плану.

Перша робота, яку розглянемо, має свою статтю, що опублікована в журналі «Вісник» [10].

У статті "Розвиток технологій інтелектуального аналізу ринку вживаних автомобілів" досліджено важливу проблему точного передбачення цін на вживані автомобілі та розвитку технологій, що можуть покращити цей процес. Автор використовує розвідувальний аналіз великих наборів даних, враховуючи різні параметри, що характеризують автомобілі, та будує моделі машинного навчання для передбачення цін.

Важливим аспектом є вказівка на необхідність попереднього розвідувального аналізу для ефективного фільтрування та відсіювання помилкових даних. Результати дослідження підтверджують високий потенціал інтелектуальних технологій у вдосконаленні процесу продажу вживаних автомобілів.

Висновки статті свідчать про досягнення високої точності передбачення цін, особливо за допомогою конкретних моделей машинного навчання. Загалом, дослідження вказує на перспективи використання інтелектуальних технологій для оптимізації ринку вживаних автомобілів та покликане служити основою для подальших наукових досліджень в даній області [9].

Усі інші дослідження аналогів проводяться по нотбуках, що знаходяться на платформі Kaggle, в яких досліджується датасет «Used Cars Dataset» в якому міститься інформація про автомобілі, виставлені на продаж на сервісі електронних оголошень Craigslist.com, та в яких проводяться інші, аналогічні до нашої теми заходи [11].

Було вирішено, першим розглянути блокнот(Notebook) «Craigslist_Car Data Analysis» Kaggle нотбук Мастера (Notebook Master), якого звати JeongBin Park [12].

Пройдемось по основним, важливим моментам дослідження.

Перше що досліджує JeongBin Park, так це кількість автомобілів по рокам (рис. 1.2). Що звучить дуже просто, але насправді, це робить дещо корисне для дослідження.

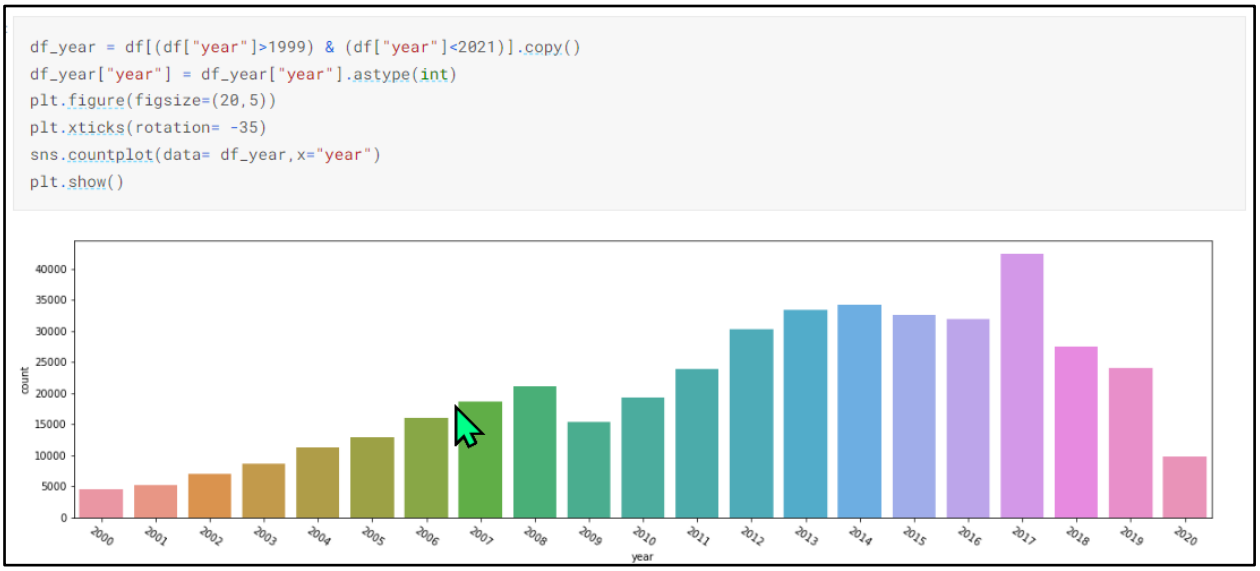

Рисунок 1.2 – Графік кількості автомобілів по рокам

Користь в тому, що дослідник видаляє із набора даних дуже старі автомобілі, тобто усі автомобілі по 2000 рік. Чим це може бути обумовлено? Старі автомобілі, в більшості випадків приблизно до 1990 року, вважаються раритетними. А як ми знаємо, ринок раритетних автомобілів має на мені зовсім інші цілі, ніж просто купівля та експлуатація автомобіля [12].

Справжні поціновувачі старих автомобілів обираються їх за декількох причин:

– Історія та естетика. Раритетні автомобілі часто мають цікаву історію та естетичну привабливість, яка може бути відсутня в сучасних автомобілях. Вони можуть бути свідками певної епохи або дати, або ж вони можуть просто мати унікальний дизайн, який не зустрічається більше ніде.

– Рідкість. Раритетні автомобілі часто є рідкісними, що може зробити їх більш бажаними для деяких людей. Це може бути тому, що вони були випущені обмеженим тиражем, або тому, що вони були занадто дорогими для більшості людей, тому їх було випущено в обмеженій кількості.

– Ексклюзивність. Раритетні автомобілі часто є ексклюзивними, що може зробити їх більш бажаними для деяких людей. Це може бути тому, що

вони були розроблені для певного клієнта, або тому, що вони були випущені обмеженим тиражем для певного ринку.

– Цінність. Раритетні автомобілі часто мають високу цінність, що може зробити їх хорошим вкладенням. Ціна раритетних автомобілів може зростати з часом, що може зробити їх хорошим способом інвестувати гроші.

Люди, які полюбляють раритетні автомобілі, можуть переслідувати різні цілі. Ось ще деякі з них:

– Насолоджуватися історією та естетикою автомобіля. Деякі люди просто люблять насолоджуватися історією та естетикою раритетної машини. Вони можуть насолоджуватися її дизайном, її механікою або її історією.

– Похвастатися своїм автомобілем. Деякі люди полюбляють похвалитися своїм автомобілем. Вони можуть бути пишними тим, що володіють рідкісним або дорогим автомобілем.

– Зробити інвестицію. Деякі люди бажають зробити інвестицію в раритетну машину. Бажання обумовлене тим, що ціна автомобіля зростатиме з часом, що дозволить їм отримати прибуток.

Звідси слідує, що передбачення цін раритетних автомобілів в цілі нашої задачі не входить, більше того, ціни таких автомобілів, часом, не піддаються логіці, адже історію важко оцінювати. Ще, можна зробити припущення, що старі дорогі, або старі дешеві автомобілі будуть тільки погано впливати на передбачення цін вживаних сучасних автомобілів. Але і варто додати, що автор надто сильно обрізав дані, адже є чудові автомобілі, до 2000 року, але не старіше 1990, які досі у вжитку і їх варто використовувати при дослідженні. Видалені автомобілі, починаючи з 2021 року, теж трохи дивна ситуація, але, можливо, це пов’язано із дуже великими цінами «нових» вживаних автомобілів

Цікаво ще те, що кількість вживаних автомобілів в 2009 та 2010 впала. Можливо через якусь ситуацію в світі, а можливо це просто так складений набір даних, що тоді ніякої цінності для дослідження не несе.

Далі йде порівняння середньої ціни автомобілів за роками (рис 1.3). І тут є одна, дуже цікава, закономірність – з кожним роком, середня ціна вживаних автомобілів зростає [12].

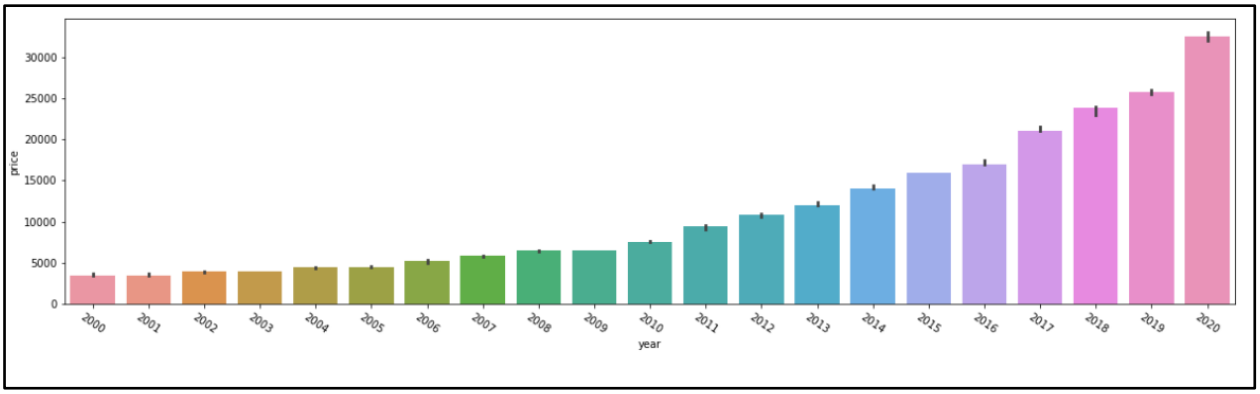

Рисунок 1.3 – Графік середньої ціни автомобілів по рокам

Як ми можемо побачити, сильний приріст в ціні, починається з 2010 по 2011 роки і досі зростає. Це можна пояснити декількома причинами:

– Зростаюча вартість виробництва автомобілів. Вартість виробництва автомобілів постійно зростає, оскільки виробники автомобілів повинні враховувати зростаючі витрати на сировину, робочу силу та технології.

– Зростаюча попит на автомобілі. Попит на автомобілі також зростає, оскільки населення зростає, а люди все частіше переміщуються з одного місця в інше.

– Зростаючі екологічні стандарти. Виробники автомобілів повинні вкладати кошти в розробку нових технологій, які відповідають жорстким екологічним стандартам.

– Нестача напівпровідників. Нестача напівпровідників, яка виникла під час пандемії COVID-19, також вплинула на ціни на автомобілі, оскільки виробники автомобілів не можуть виробляти стільки автомобілів, скільки вони б хотіли.

– Вартість сталі, алюмінію та інших матеріалів, що використовуються в автомобілях, зросла на 40-50% за останні кілька років. Це пов'язано з

зростанням попиту на ці матеріали в інших галузях, таких як будівництво та промисловість.

– Вартість робочої сили також зростає, оскільки працівники вимагають вищої заробітної плати. Це пов'язано з тим, що працівники автомобілебудівної галузі є висококваліфікованими і мають високий попит.

– Розробка нових технологій, таких як електромобілі та автономне водіння, вимагає значних інвестицій. Це також сприяє зростанню цін на автомобілі.

Неможливо певно сказати, як довго ціни на автомобілі будуть продовжувати зростати. Однак, ймовірно, що найближчі роки вони залишаться високими.

Доказом всьому вище сказаному є графік, що було зображено раніше (рисунку 1.2), де видно що кількість «нових» вживаних автомобілів досить мала і стає все менше і менше. Логічно, що «нових» вживаних автомобілів буде куди менше на ринку ніж старих вживаних. Це можна пояснити тим, що виробники гарантують якість автомобілів до 10 років служби і не багато людей спішить їх продавати 3-10 років від початку експлуатації.

На рисунку 1.4 зображено, окремо по роках 2000, 2005, 2010 та 2020, кількість автомобілів по показнику «Виробник» [12].

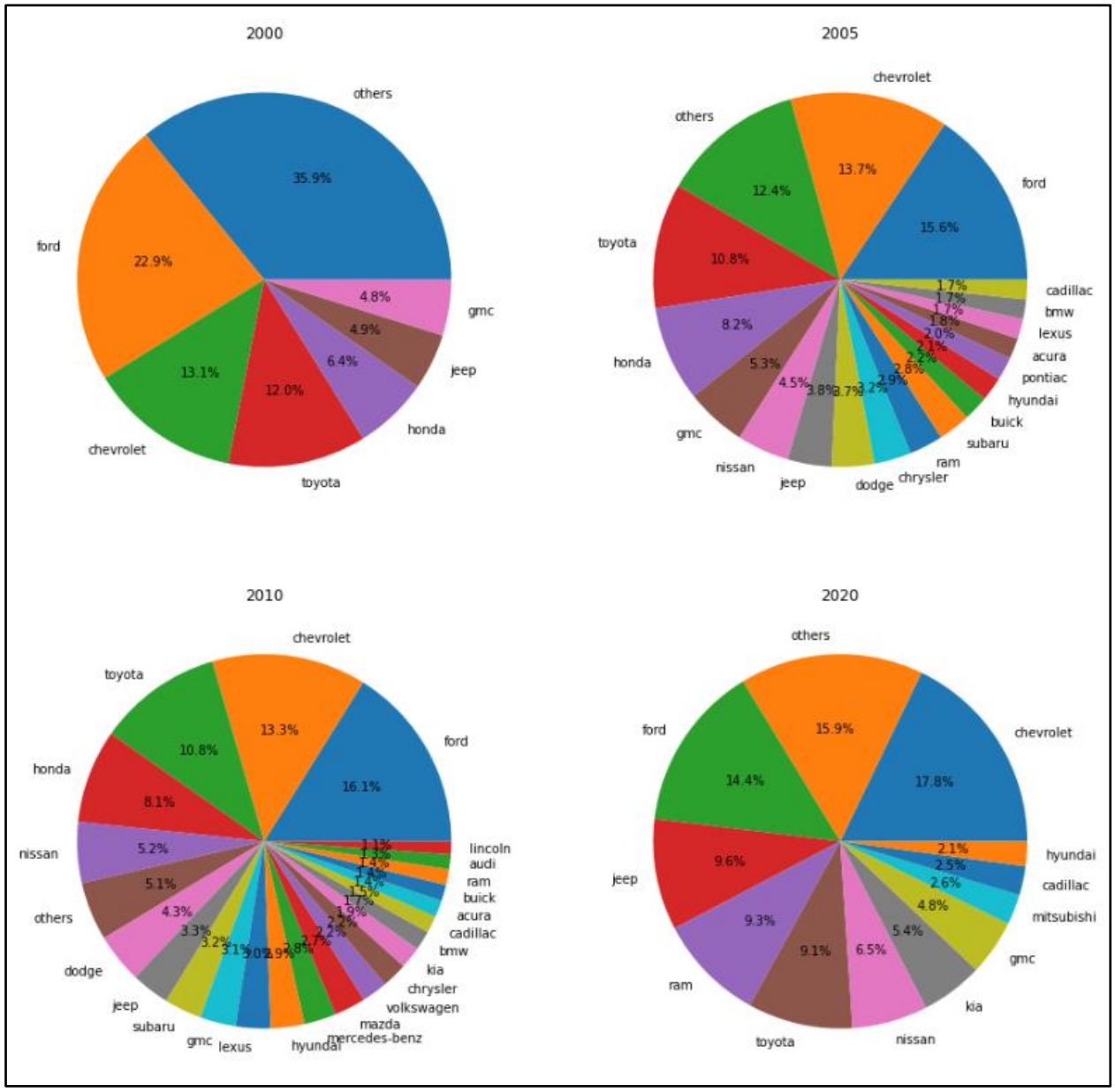

Рисунок 1.4 – Найпопулярніші марки автомобілів по рокам

На рисунку 1.4 можемо побачити, що найбільша кількість автомобілів таких виробників: Ford,Chevrolet, Toyota, Honda, Jeep,GMC, Nissan та Dodge.

Що дивно, так це відсутність великої кількості різних виробників в 2000 та 2020 роках. А цікавим є те, що кількість автомобілів Chevrolet переважило кількість Ford.

Таку популярність автомобілів цих виробників можна пояснити багатьма чинниками, але на даний момент, це ніякої цінності не несе.

Дослідник створив стовпчикову діаграму 10-ти найпопулярніших моделей автомобілів в наборі даних зображено на рисунку 1.5 [12].

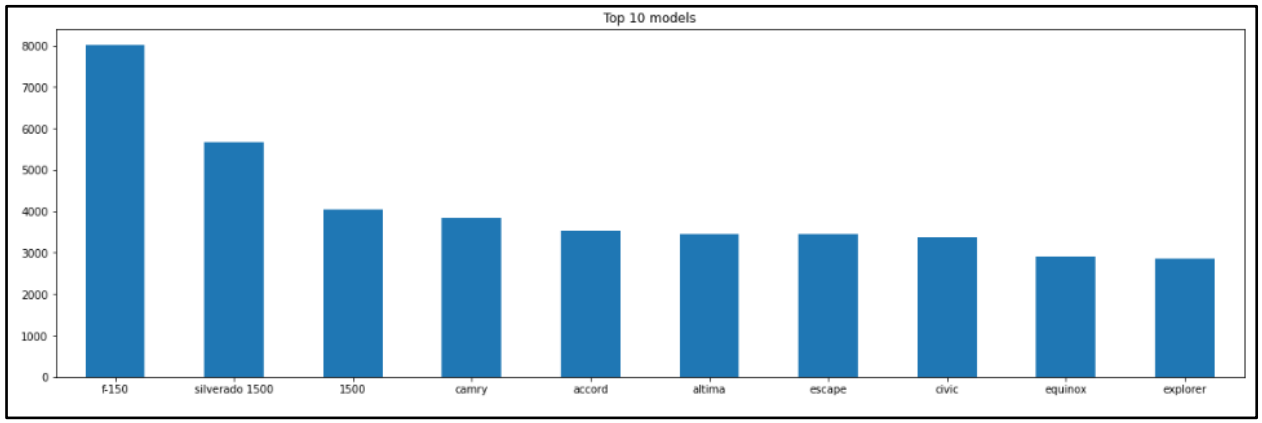

Рисунок 1.5 – Топ 10 найпопулярніших моделей автомобілів в наборі даних

Як видно з рисунку 1.5, особливо виділяються Ford f-150 та Chevrolet Silverado 1500, GMC Sierra 1500, Toyota Camry, Honda Accord, Nissan Altima, Ford Escape.

В більшості випадків, на ринку США, зустрічаються автомобілі типу седан, SUV, позашляховик пікап, траки(вантажівки) та хетчбеки, що чудово ілюструє графік, що зображено на рисунку 1.6.

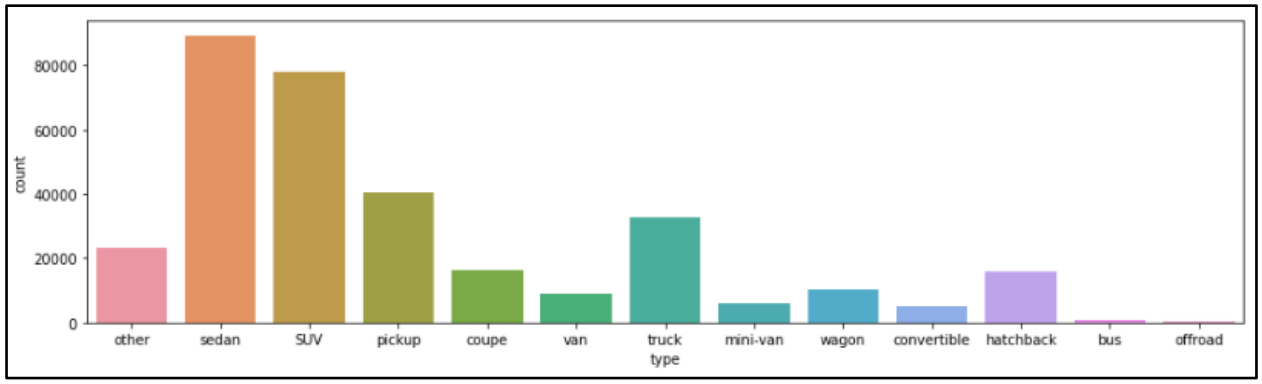

Рисунок 1.6 – Кількість автомобілів в датасеті по їх типу

Найцікавіший графік, який навів дослідник, це порівняння по регіонам, де який тип автомобіля використовується найчастіше. Ідея чудова, але реалізована не правильно. Бо краще показувати не точками, де вони

знаходяться, а використовувати Heat Map. Це краще покаже, де і скільки є автомобілів. Якщо потрібно, можна скласти багато таких карт, по окремим типам автомобілів. Але, що ще не правильно зробив дослідник, так це за зря видалив певну кількість автомобілів, які знаходяться поза межами певної частини США (рис. 1.7) [12].

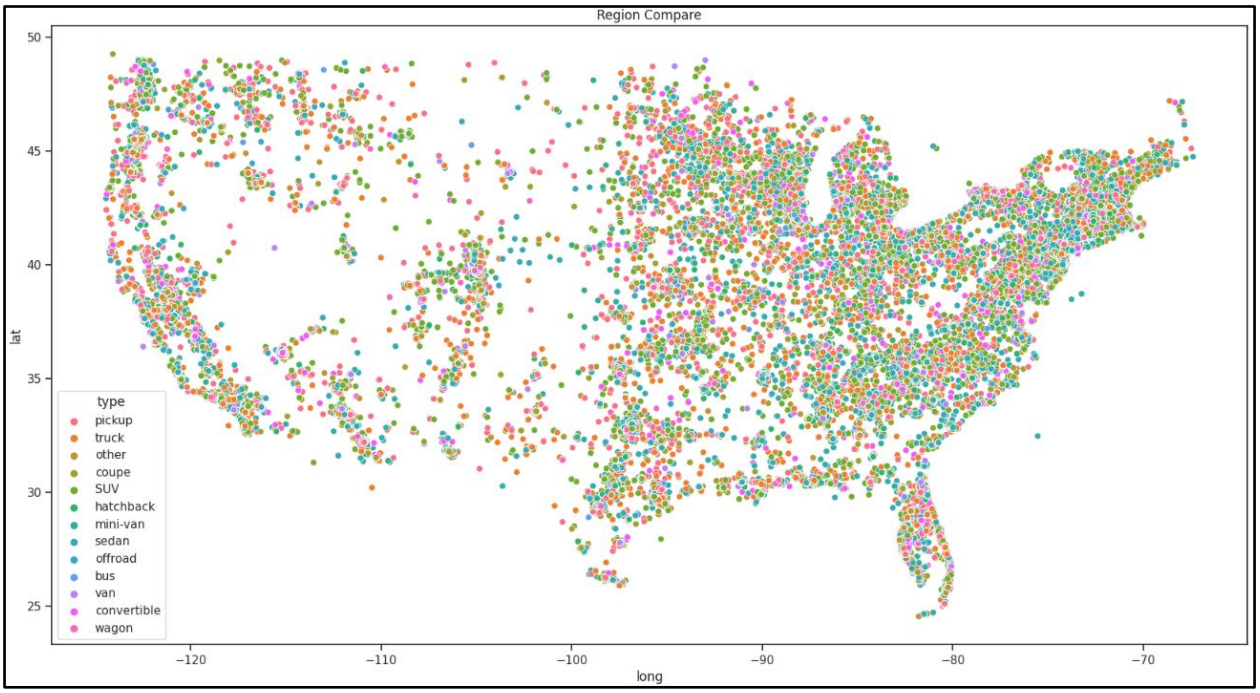

Рисунок 1.7 – Графік порівняння типів автомобілів по регіонам

Цей крок може бути обумовлений неправильно виставленим значенням довготи та широти, де знаходиться автомобіль, але автор не врахував іншої частини США та просто відкинув більше 5 тисяч автомобілів. На швидку руку, було побудовано карту і прив’язано до неї усі автомобілі (рис. 1.8).

Рисунок 1.8 – Карта знаходження усіх автомобілів із датасету

На рисунку 1.8 можемо помітити, що багато автомобілів знаходиться просто в морях та на інших материках, нонсенс, адже сервіс електронних оголошень працює в межах США. Але провівши огляд усіх точок, можна спокійно сказати, що автомобілі, що знаходяться в ділянці червоних кружечків, є правильними, або близькими до правди, усі ж інші «сині», «зелені» та «жовті», варто відфільтрувати.

При дослідженні та очистці даних, інший дослідник зміг зробити візуалізацію положення усіх автомобілів куди краще (рис. 1.9). Дослідник İsmail Sefa Akdeniz, Notebooks Master, Kaggle нотбук: Used Cars Data Analysis and Visualization (EDA) [13].

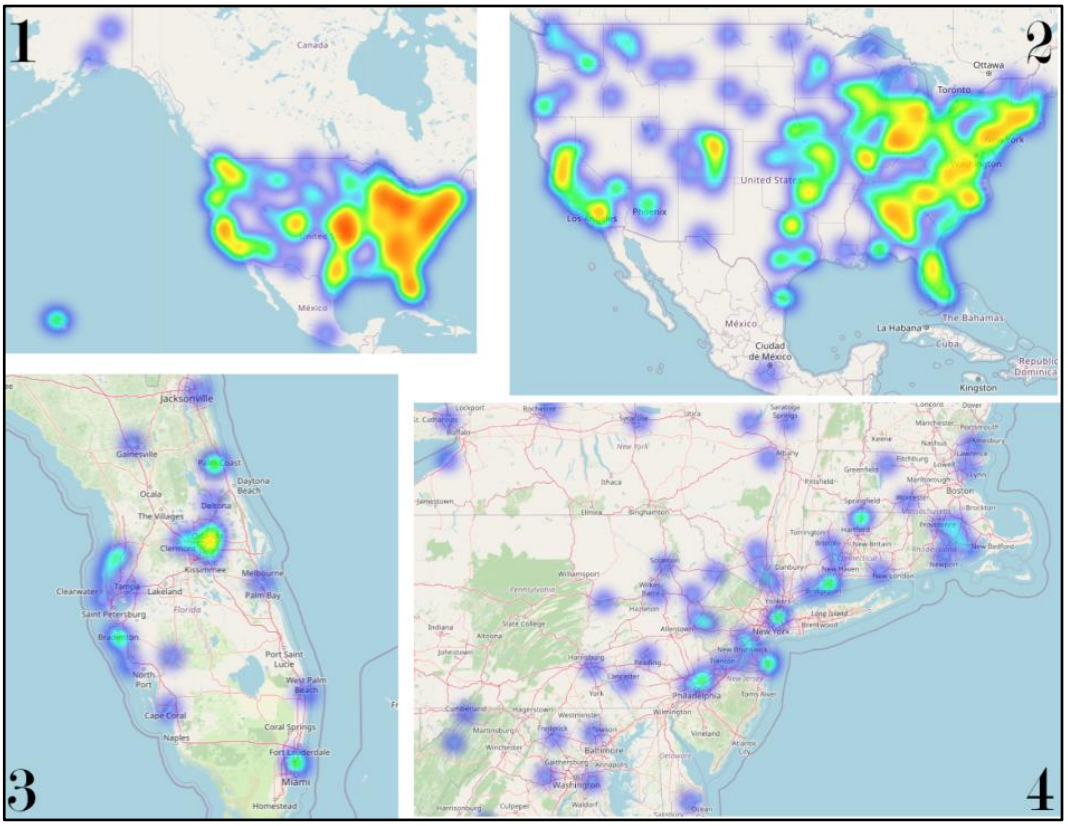

Рисунок 1.9 – Heat map кількості автомобілів в США

На рисунку 1.9 зображено Heat Map усіх автомобілів в США. Зображення поділено на 4 частини. Перший малюнок показує Heat Map усіх автомобілів в «північній», «південній» частині США та на островах (Штат Гаваї). Другий малюнок показує Heat Map автомобілів в «південній» частині США, звідси чудово видно, що більшість автомобілів розкидано по таким штатам як: Флорида, Каліфорнія, Колорадо, Індіана, Огайо, Вірджинія, Нью-Джерсі, Пенсільванія, Коннектикут, Вісконсин та Массачусетс. Третій малюнок показує як розкидані автомобілі по штату Флорида. Четвертий малюнок показує розкиданість автомобілів в правій частині, прилеглій до моря США, ближче до Нью-Йорку.

Інформація про місцезнаходження автомобіля може бути дуже корисною, адже, в залежності від штату, один і той самий автомобіль та із одними й тими самими характеристиками може коштувати по різному, десь дорожче на декілька сотень доларів, а десь дешевше на декілька тисяч.

Окрім цього, дослідник (İsmail Sefa Akdeniz) показує ще такі цікаві графіки (рис.1.10 - 1.11).

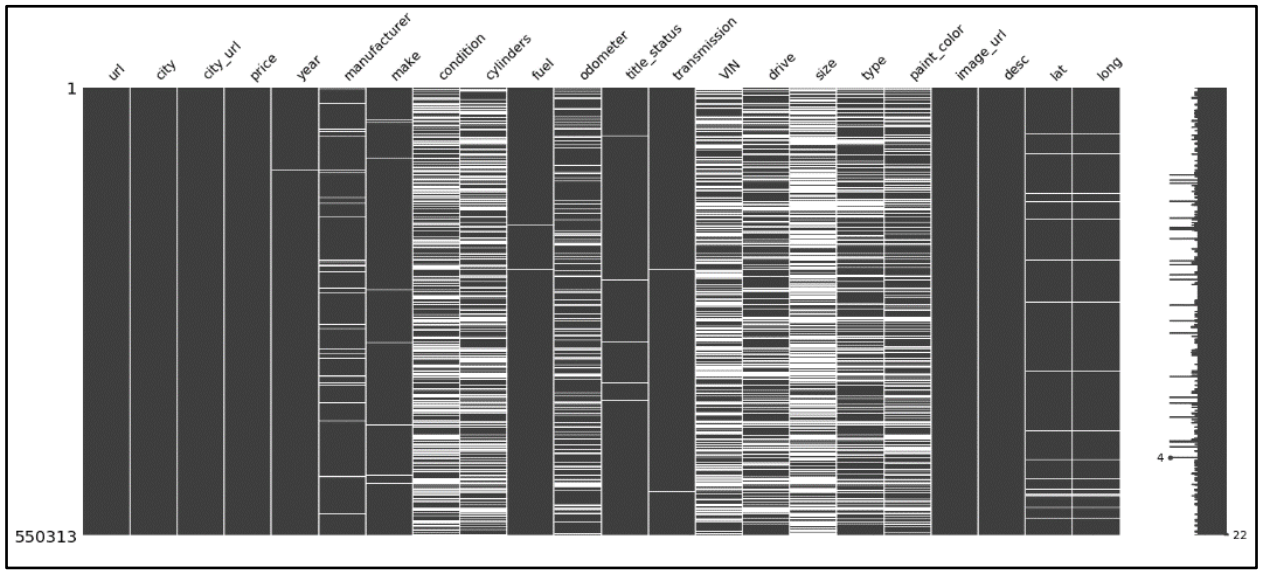

Рисунок 1.10 – Графік відсутності значень

На рисунку 1.10, ми бачимо що досить багато значень в нашому наборі відсутні (Білим кольором позначено відсутні), набір даних складається з 550313 рядків та 22 стовпців. Назви стовпців та що вони означають:

– url: URL-адреса сторінки з оголошенням про продаж автомобіля;

– city: Місто, в якому знаходиться автомобіль;

– city_url: URL-адреса сторінки з інформацією про місто, в якому знаходиться автомобіль;

– price: Ціна автомобіля;

– year: Рік випуску автомобіля;

– manufacturer: Виробник автомобіля;

– make: Модель автомобіля;

– condition: Стан автомобіля;

– cylinders: Кількість циліндрів у двигуні автомобіля;

– fuel: Тип палива, яке використовує автомобіль;

– odometer: Стан кілометражу автомобіля;

– title_status: Стан реєстрації автомобіля;
– transmission: Тип трансмісії автомобіля;
– VIN: Ідентифікаційний номер автомобіля;
– drive: Тип приводу автомобіля;
– size: Розмір автомобіля;
– type: Тип автомобіля;
– paint_color: Колір автомобіля;
– image_url: URL-адреса зображення автомобіля;
– desc: Опис автомобіля;
– lat: Широта місця розташування автомобіля;
– long: Довгота місця розташування автомобіля.

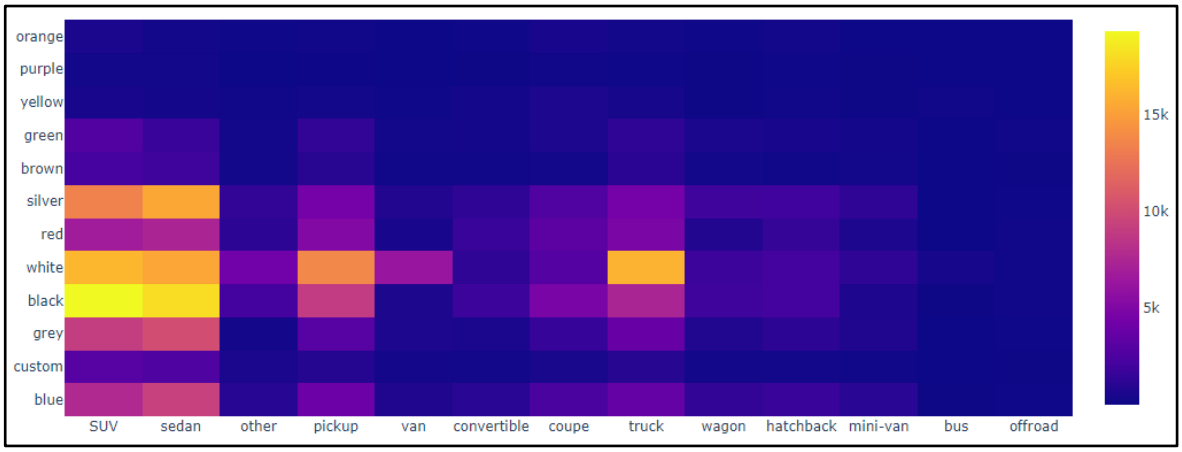

Рисунок 1.11 – Графік кількості автомобілів за типом автомобіля та його кольору

З рисунку 1.11, можемо побачити, що вподобання на кольори автомобілів різні, але деякі з них виділяються. Так автомобілі типу SUV, седан, пікап та трак(вантажівка) переважно мають колір: білий, чорний та сірий, після чого, ще можна виділити декілька тисяч інших автомобілів із такими кольорами: блакитний, срібний та зелений.

Не факт, що колір автомобіля впливає на його ціну, але, можна зробити припущення, що в залежності від кольору автомобіля, покупець краще обере той автомобіль, який такого кольору, який йому більше подобається.

Є ще одна, досить велика і гарна робота, дослідника Abhash Panwar, але яка знаходиться на платформі Github і має назву Used Car Price Prediction (End-to End Project) [14].

В ній є все, від початку до кінця, і навіть колись була реалізація програми, що була викладена на сервер і працювала як сайт, що рекомендує ціни на автомобілі, потрібно було лише ввести усі значення. На жаль, на сьогоднішній день сайт не працює.

Сама ж робота складається з 4 частин:

а) Data Prerpocessing;

б) Data Visualization;

в) Model Implementations;

г) Model deployment using Django.

В самій роботі є досить багато різних гарно побудованих та інформативних графіків для аналізу, дослідження та виявлення певних закономірностей між ознаками. Але приділимо увагу до Реалізації моделі (рис. 1.12).

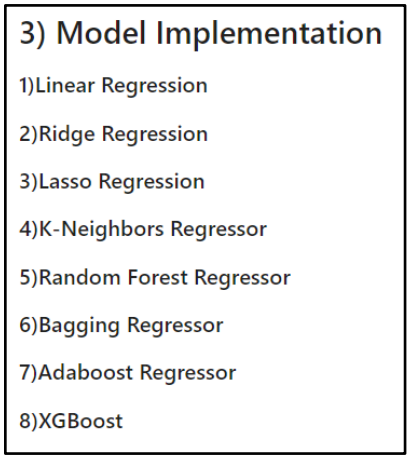

Рисунок 1.12 – Моделі машинного навчання

Бачимо, що дослідник вирішив використати одразу 8 регресійних моделей для передбачення цін на вживані автомобілі. Розглянемо найкращу з поміж усіх а саме XGBoost Regressor (рис. 1.13) [14].

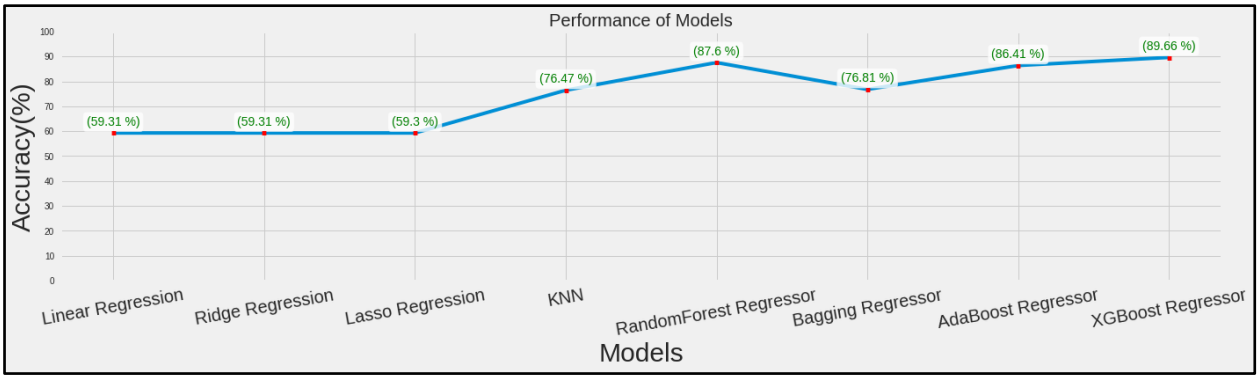

Рисунок 1.13 – Продуктивність моделей машинного навчання

З рисунку 1.13 бачимо, що дослідник досяг «точності» передбачення цін 89.66%. Проаналізуємо побудову моделі.

8) XGBOOST

In [42]:
```
#model implementation and fitting data
xg_reg = xgb.XGBRegressor(objective ='reg:squarederror', learning_rate = 0.4,
                max_depth = 24, alpha = 5, n_estimators = 200)
xg_reg.fit(X_train,y_train)
y_pred = xg_reg.predict(X_test)
```

In [43]:
```
#model evaluation
y_test_1,y_pred_1=remove_neg(y_test,y_pred)
r8_xg=result(y_test_1,y_pred_1)
print("MSLE : {}".format(r8_xg[0]))
print("Root MSLE : {}".format(r8_xg[1]))
print("R2 Score : {} or {}%".format(r8_xg[2],r8_xg[3]))
```

```
MSLE : 0.0006504702126268066
Root MSLE : 0.02550431752913233
R2 Score : 0.896623162653403 or 89.6623%
```

Рисунок 1.14 – Побудова, тренування та результат роботи моделі

На рисунку 1.14 видно, як була побудована модель та те, які результати передбачення вона дає. Помилка в візуалізації так званої «точності», R2 Score

– це міра того, наскільки добре модель регресії пояснює дані. R2 Score також називається коефіцієнтом детермінації. R2_score може приймати значення від 0 до 1, де 0 означає, що модель не пояснює жодних даних, а 1 означає, що модель ідеально пояснює всі дані. І множити це значення на 100, щоб сказати, що це «точність» передбачення в відсотках, не правильно, але помилка не критична, це лише візуальна помилка.

Далі, він показує важливість ознак моделі XGBoost Regressor (рис. 1.15). Де добре видно, що такі ознаки як odometer(Пробіг), lat(Широта місця розташування автомобіля), long(Довгота місця розташування автомобіля), model(Модель автомобіля), region (Регіон або Штат де знаходиться автомобіь), year (Рік автомобіля), manufacturer (Виробник автомобіля) дуже сильно впливають на передбачення ціни автомобіля, що варто взяти до уваги при створенні своєї інформаційної системи аналізу та передбачення цін на вживані автомобілі.

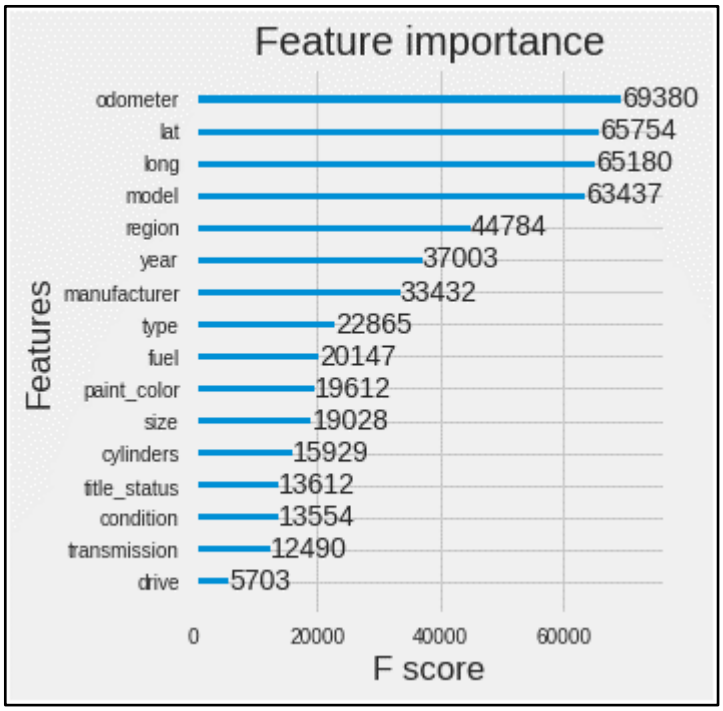

Рисунок 1.15 – Важливість ознак моделі XGBoost Regressor

Потрібно наголосити, що Важливість ознак (Feature importance) – це міра важливості кожної ознаки для моделі. Вона використовується для оцінки того, наскільки кожна ознака впливає на передбачення моделі. Feature importance є важливим інструментом для машинного навчання. Він може бути використаний для поліпшення продуктивності моделей, зменшення перенавчання та інтерпретації моделей.

Результати передбачення цін на вживані автомобілі за тестовими даними, яких досяг дослідник, зображено на рисунку 1.16

	Linear Regression	Ridge Regression	Lasso Regression	KNN	RandomForest Regressor	Bagging Regressor	AdaBoost Regressor	XGBoost Regressor
MSLE	0.002434	0.002434	0.002434	0.001440	0.000778	0.001432	0.000845	0.000650
Root MSLE	0.049336	0.049336	0.049336	0.037948	0.027895	0.037841	0.029065	0.025504
R2 Score	0.593051	0.593051	0.593050	0.764681	0.875979	0.768090	0.864084	0.896623
Accuracy(%)	59.305100	59.305100	59.305000	76.468100	87.597900	76.809000	86.408400	89.662300

Рисунок 1.16 – Результати передбачення цін на вживані автомобілі іншими моделями

На рисунку 1.16 показано результат передбачення цін на вживані автомобілі за тестовими (валідаційними) даними. Бачимо що себе, також, добре показали такі моделі: AdaBoostRegressor та RandomForestRegressor. Можливо, дослідник провів не достатню фільтрацію даних, чи ще щось, тому найкраща «точність», описана коефіцієнтом R2 Score становить 0.8966 [14].

Тому, нашою задачею буде покращення аналізу датасету, фільтрації та точності моделей передбачення даних. Задача складна, потребує багато зусиль, але актуальність задачі повинна мотивувати досягати більших результатів.

На жаль, неможливо порівняти системи передбачення або рекомендації цін на вживані автомобілі, що використовуються на торгових майданчиках США, оскільки ця інформація не є загальнодоступною. Тому давайте розглянемо таку систему на прикладі одного з найпопулярніших українських

веб-сервісів, Auto.RIA, та даних з веб-сервісу CraigsList, який надає частковий доступ до не засекречених даних.

На українському ринку існує понад 12 веб-сайтів, які пропонують прямий продаж та купівлю автомобілів. Серед них Auto.RIA є найпопулярнішим для продажу вживаних автомобілів [15].

API сайту AUTO.RIA дозволяє користувачам здійснювати запити до бази даних сайту. Зокрема, можна запитати середню ціну будь-якого транспортного засобу за його параметрами [16].

Наприклад, запит на середню ціну Honda Accord в Києві буде виглядати так:

http://api.auto.ria.com/average?marka_id=28&model_id=262&city_id=9

Результатом запиту буде середнє значення цін всіх автомобілів Honda Accord, що продаються в Києві. Результат запиту буде представлений у вигляді JSON-файлу [17].

Приклад результату запиту зображено на рисунку 1.17.

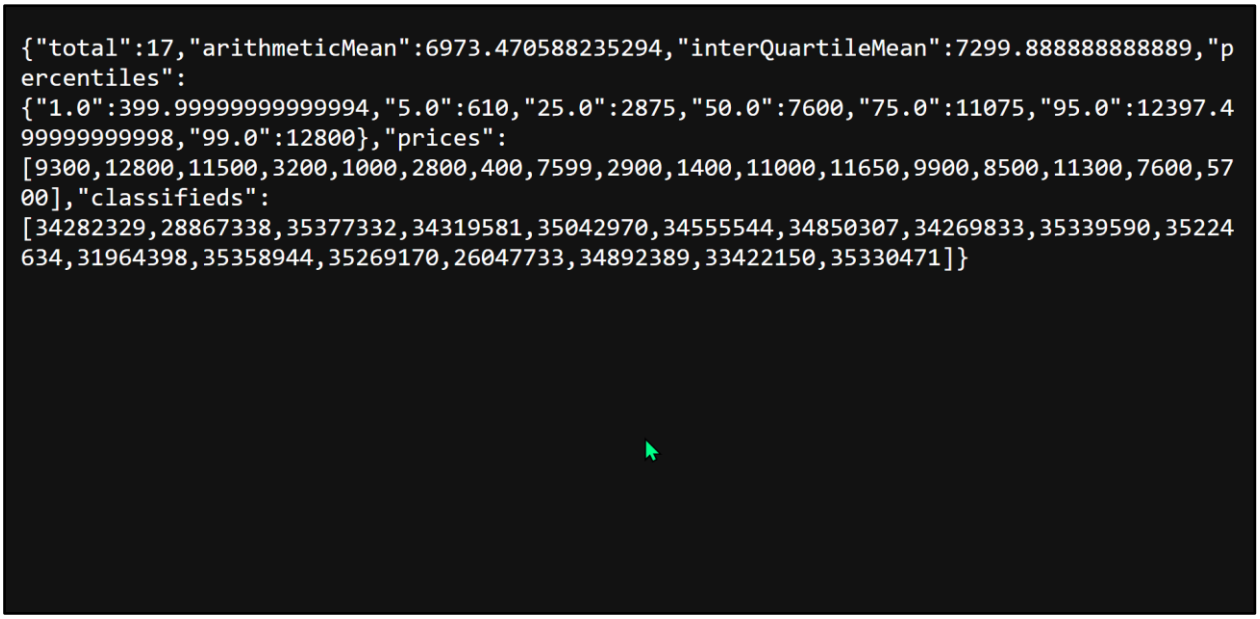

Рисунок 1.17 – Вигляд результату запиту на середню ціну автомобіля

Як видно з рисунка 1.17, запит повертає певну статистичну інформацію, а саме: total, arithmeticMean, interQuartileMean, percentiles, prices, classifieds.

Розберемо кожне із значень:

– total: це міра того, скільки оголошень було використано для розрахунку. Чим більше оголошень, тим точніше буде результат.

– arithmeticMean: це найпростіший спосіб розрахунку середньої ціни. Він обчислюється шляхом додавання всіх цін і поділу на загальну кількість оголошень.

– interQuartileMean: це більш точний спосіб розрахунку середньої ціни. Він обчислюється шляхом додавання цін, які знаходяться між першим і четвертим квантилями. Квантиль — це значення, яке розділяє дані на 4 рівні частини.

– percentiles: це значення, які показують, яка частка оголошень має ціну нижче певного значення. Наприклад, для даного прикладу 25% всіх оголошень мають ціну нижче за $3500.

– prices: це список фактичних цін оголошень, які використовувалися для розрахунку. Цей список може бути корисним для аналізу даних.

– classifieds: це ідентифікатори оголошень, до яких належать ціни відповідно. Цей список може бути корисним для подальшого дослідження оголошень.

Якщо по якійсь причині не вдалося розрахувати середню ціну, то відповідь матиме статус 400 Bad Request. Це може статися, якщо в базі даних немає оголошень, які відповідають заданим параметрам. Тіло JSON-файлу виглядатиме, як зображено на рисунку 1.18 [17].

Рисунок 1.18 – Відповідь на запит на середню ціну автомобіля з помилкою

Наявність обмеженого функціоналу API сайту AUTO.RIA свідчить про те, що вирішення задачі передбачення цін на вживані автомобілі є актуальним.

Для вирішення цієї задачі важливо виділити закономірності, які утворюють ціну на вживані автомобілі. Для цього необхідно провести розвідувальний аналіз даних, обробку даних та навчання моделей.

Розвідувальний аналіз даних допоможе виявити основні ознаки, які впливають на ціну автомобіля.

Обробка даних дозволить очистити дані від шуму та виправити помилки.

Навчання моделей за допомогою методів машинного навчання дозволить отримати модель, яка буде здатна прогнозувати ціни на вживані автомобілі.

Для підвищення точності прогнозу можна використовувати кілька моделей, отриманих за допомогою різних методів машинного навчання. Також можна використовувати ансамблі моделей, які поєднують прогнози декількох моделей.

Наприклад, для передбачення ціни на вживаний автомобіль можна використовувати такі ознаки, як:

- Марка і модель автомобіля
- Рік випуску
- Пробіг
- Стан автомобіля
- Комплектація
- Місто розташування

За допомогою розвідувального аналізу даних можна виявити, які з цих ознак є найважливішими для ціноутворення. Після цього можна обробити дані та навчити модель передбачення цін.

1.4 Висновки

В першому розділі даниї роботи проведено огляд ринку онлайн-продажів вживаних автомобілів в США та Україні. Проаналізовано прогнозовані тенденції розвитку цього ринку в США. Наведено переваги та

недоліки купівлі вживаних та нових автомобілів. Розглянуто сервіси, що надають інформацію про вживані автомобілі.

На основі проведеного аналізу було зроблено висновок, що існуючі сервіси не забезпечують достатньої інформації для точного передбачення цін на вживані автомобілі. Це пов'язано з тим, що в їх базах даних містяться лише рекомендації по наявних схожих продажах. Такий підхід не дозволяє врахувати всі фактори, які впливають на ціну автомобіля, такі як:

– Марка і модель автомобіля;
– Рік випуску;
– Пробіг;
– Стан автомобіля;
– Комплектація;
– Місто розташування.

Але роботи інших дослідників показують досить чудові результати, які можна тільки покращувати, за що варто взятись та досягти кращих результатів передбачення цін на вживані автомобілі.

Для розв’язання цієї проблеми було вирішено реалізувати інформаційну технологію аналізу та передбачення цін на вживані автомобілі із використанням моделей машинного навчання. Така технологія дозволятиме враховувати всі важливі фактори, які впливають на ціну автомобіля. Це дозволить покупцям і продавцям вжитих автомобілів приймати більш обґрунтовані рішення про купівлю або продаж. А також сприятиме розвитку ринку онлайн-продажів вживаних автомобілів.

2 ОГЛЯД ОПТИМАЛЬНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ВХІДНОГО НАБОРУ ДАНИХ

2.1 Вибір оптимальних інформаційних технологій

Інформаційна система аналізу та передбачення цін на вживані автомобілі повинна бути здатною обробляти великі обсяги даних, які містять інформацію про продажі вжитих автомобілів, характеристики вжитих автомобілів та ринкові умови. Для цього необхідно використовувати ефективні інформаційні технології.

Інформація – це відомості, які можуть бути отримані, оброблені та використані для прийняття рішень. Інформація може бути отримана з навколишнього середовища, передана іншим системам або збережена всередині системи для подальшого використання.

Інформаційна технологія – це комплекс методів, процесів та засобів для створення, зберігання, обробки та передачі інформації за допомогою комп'ютерів та комунікаційних мереж. Інформаційні технології використовуються для підвищення ефективності діяльності людей та організацій.

До основних інформаційних технологій, які можна використовувати для реалізації інформаційної системи аналізу та передбачення цін на вживані автомобілі, відносяться [18]:

- Системи управління базами даних (СУБД);
- Системи машинного навчання (СММ);
- Системи аналізу даних (САД).

СУБД використовуються для зберігання та керування даними. Вони забезпечують ефективний доступ до даних, а також дозволяють проводити різні операції над даними, такі як додавання, видалення та модифікація даних.

СММ використовуються для розробки моделей машинного навчання, які здатні вирішити задачу прогнозування цін на вживані автомобілі.

Машинне навчання – це галузь штучного інтелекту, яка займається розробкою методів, що дозволяють комп'ютерам автоматично навчатися на даних без явного програмування.

САД використовуються для аналізу даних. Вони дозволяють проводити різні статистичні та візуальні аналізи даних, що допомагає виявити закономірності та тенденції в даних.

Для вибору оптимальних інформаційних технологій необхідно враховувати такі фактори, як:

- Обсяг даних;
- Складність задачі;
- Вимоги до точності та своєчасності прогнозу.

Обсяг даних є важливим фактором, оскільки він визначає, які СУБД та СММ можна використовувати. Для великих обсягів даних необхідно використовувати СУБД та СММ, які підтримують паралельні обчислення.

Складність задачі також є важливим фактором, оскільки він визначає, які методи машинного навчання можна використовувати. Для складних задач необхідно використовувати СММ, які підтримують різні методи машинного навчання [18].

Вимоги до точності та своєчасності прогнозу є важливим фактором, оскільки вони впливають на вибір СММ. Для забезпечення високої точності прогнозу необхідно використовувати СММ, які навчені на великому наборі даних. Для забезпечення своєчасності прогнозу необхідно використовувати СММ, які можуть швидко навчатися та прогнозувати ціни.

Розв’язання, практично, будь-якої задачі аналізу та передбачення (прогнозування) даних з використанням інтелектуальних методів машинного навчання зазвичай здійснюється у такі етапи:

– Збір даних. На цьому етапі необхідно зібрати дані, які будуть використовуватися для навчання і тестування моделі. Дані можуть бути зібрані з різних джерел, таких як веб-сайти, бази даних, інтернет-аукціони та ін.

– Очищення даних. Після збору даних їх необхідно очистити від помилок і некоректних значень. Це важливо для забезпечення точності моделі.

– Аналіз даних. На цьому етапі необхідно проаналізувати дані, щоб визначити, які фактори впливають на результат, який необхідно передбачити.

– Вибір моделі. Після аналізу даних необхідно вибрати модель, яка найкраще підходить для даної задачі. Існує багато різних моделей машинного навчання, кожна з яких має свої переваги та недоліки.

– Навчання моделі. На цьому етапі модель навчається на даних, які були зібрані та очищені.

– Тестування моделі. Після навчання модель необхідно протестувати на незалежних даних, щоб оцінити її точність.

– Впровадження моделі. Після успішного тестування модель можна впровадити в реальний світ.

У деяких випадках може бути потрібно виконати додаткові етапи, такі як [18]:

- Відбір ознак. На цьому етапі необхідно вибрати найбільш важливі ознаки для моделі.
- Перетворення ознак. На цьому етапі необхідно перетворити ознаки таким чином, щоб модель могла їх зрозуміти.
- Врегулювання параметрів моделі. На цьому етапі необхідно налаштувати параметри моделі, щоб забезпечити її найкращу точність.

Розв’язання задач аналізу та передбачення (прогнозування) даних з використанням інтелектуальних методів машинного навчання є складним процесом, який вимагає наявності певних знань і навичок. Однак, дотримуючись описаних вище етапів, можна підвищити шанси на успіх.

2.1.1 Вибір мови програмування для розробки інформаційної технології

У галузі машинного навчання існує широкий спектр мов програмування, які дозволяють працювати з даними та машинним навчанням. Серед них можна виділити кілька основних і найпопулярніших мов програмування, спрямованих на аналіз даних та/або машинне навчання: Python, SQL, R і MATLAB. Далі ми розглянемо, чим ці мови такі популярні, і яка з них найкраще підійде для вирішення нашої задачі.

Розглядаючи MATLAB як інструмент для чисельного аналізу даних, важливо враховувати, що це програмне забезпечення має значний вплив на наукові дослідження та розробку. Система розроблена компанією The Math Works. MATLAB відомий розширеними можливостями для обробки та аналізу даних, включаючи математичні розрахунки, візуалізацію даних та створення моделей (рис. 2.1) [19].

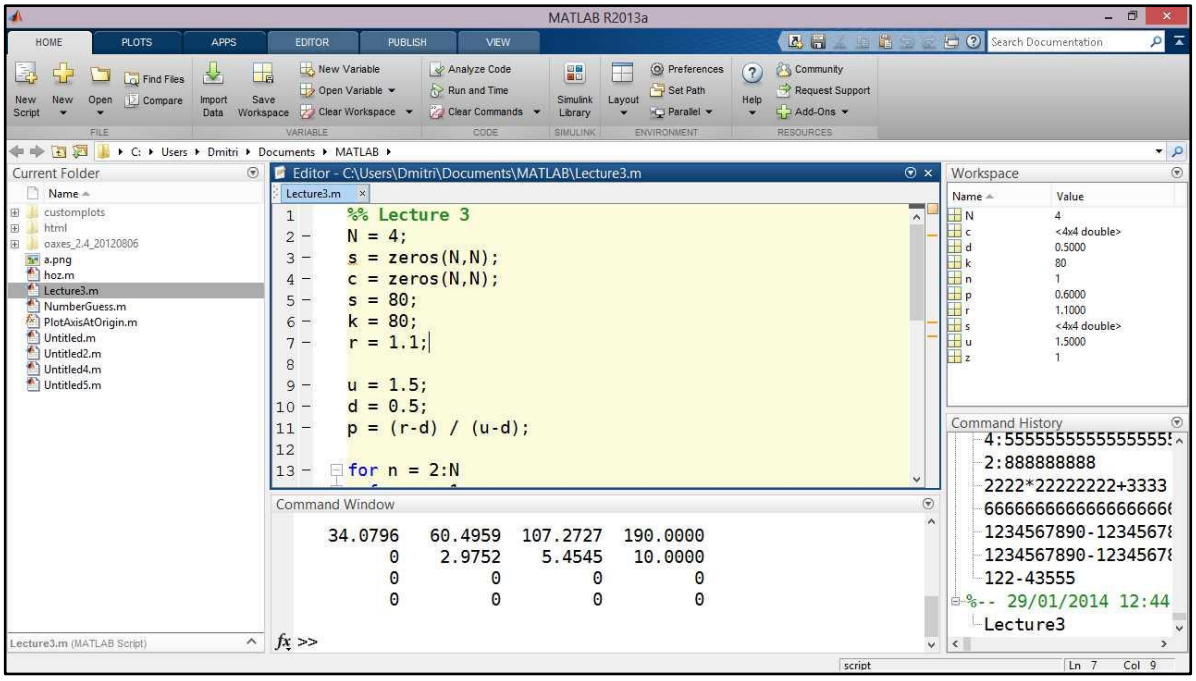

Рисунок 2.1 – Середовище розробки та синтаксис в MATLAB

Однак, MATLAB також має свої переваги і недоліки. До переваг відносяться його потужна мова програмування, широкий спектр вбудованих функцій та відкритість. Це робить його популярним інструментом для наукових досліджень і розробки.

З іншого боку, до недоліків MATLAB належать висока вартість ліцензії, яку програма вимагає придбати, можливість помилок та необхідність навчання для ефективного використання.

У вас також є можливість користуватися спеціальними пропозиціями для навчальних закладів, що може знизити вартість ліцензії.

У підсумку, MATLAB є потужним інструментом для аналізу даних, який може бути використаний в різних сферах, включаючи наукові дослідження, розробку, фінанси і інженерію. Однак, перед використанням його слід обгрунтовано врахувати переваги та недоліки цього програмного забезпечення.

Мова програмування Python є широковідомою та популярною мовою загального призначення, яка була розроблена Гвідо ван Россумом у 1991 році. Починаючи з 2010 року, Python стрімко зростає в популярності, і в 2018 році став найпопулярнішою мовою за пошуками в Google (рис. 2.2) [20].

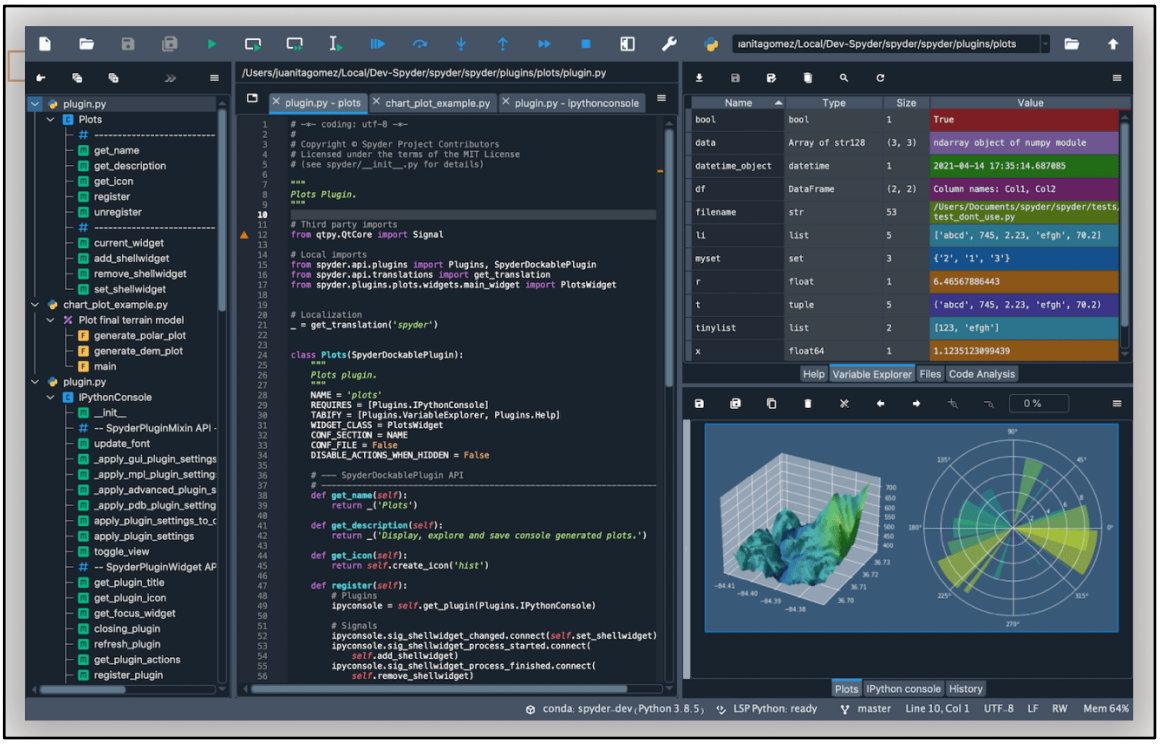

Рисунок 2.2 – Середовище розробки (Spyder) та синтаксис мови Python

Основною причиною популярності Python є його простота. Синтаксис Python є лаконічним і легким для розуміння, що робить його придатним для розробки як простих, так і складних програм. Python також є інтерпретованою мовою, що означає, що програми не потрібно компілювати, що робить їх більш швидкими в розробці.

Python є інтерпретованою, високорівневою, об’єктно-орієнтованою мовою програмування із строгою динамічною типізацією та загальним призначенням.

Інтерпретована мова означає, що програми Python не потрібно компілювати, а просто інтерпретувати. Це робить Python більш швидким у розробці, але менш ефективним, ніж компільовані мови.

Високорівнева мова означає, що Python абстрагує багато деталей низького рівня, таких як управління пам'яттю та управління потоком. Це робить Python більш продуктивним і менш схильним до помилок.

Об’єктно-орієнтована мова означає, що програми Python можуть бути побудовані з використанням об’єктів. Об’єкти є самодостатніми одиницями коду, які можуть містити дані та поведінку.

Строга динамічна типізація означає, що типи даних змінних визначаються під час виконання програми. Це може призвести до помилок, якщо програміст не врахує можливі типи даних, які можуть бути присвоєні змінній.

Python є мовою загального призначення, яка може використовуватися в широкому спектрі сфер, включаючи [21]:

– Науку і дослідження: Python широко використовується в науковій галузі для обробки даних, моделювання та аналізу.

– Розробку веб-додатків: Python є популярною мовою для розробки веб-додатків, оскільки він простий у вивченні та використанні.

– Машинне навчання і штучний інтелект: Python є популярною мовою для розробки моделей машинного навчання та систем штучного інтелекту.

– Інфраструктуру: Python використовується для створення та управління інфраструктурою, наприклад, серверами та мережами.

– Бізнес: Python використовується для розробки бізнес-додатків, таких як системи управління клієнтами та управління запасами.

Python є відкритою мовою програмування, що означає, що її код доступний для всіх. Це дозволяє розробникам вносити свій вклад у розвиток мови та створювати власні модулі та інструменти. Python також є безкоштовною мовою, що робить її доступною для широкого кола розробників.

Порівняно із статичними мовами програмування C, C++, Python працює в сотні, а то й навіть в тисячі разів повільніше. Чому ж це являється специфічним мінусом? Все через неймовірно великий функціонал та простоту мови. А швидкість можна і прискорити за допомогою спеціальних модулів, хоч і потрібно навчитись їх використовувати.

Python має велику і активну спільноту розробників, яка створює та підтримує широкий спектр модулів і інструментів. Більшість цих модулів мають вичерпну документацію, яка допомагає розробникам швидко та легко їх використовувати [20].

Python має ряд переваг, які роблять його популярною мовою для широкого спектру задач:

– Простота: Python має простий синтаксис, який легко зрозуміти та вивчити.

– Модульність: Python є модульною мовою, що означає, що програми можуть бути розбиті на дрібніші модулі, які можна повторно використовувати в інших програмах.

– Оптимізація: Python можна оптимізувати для підвищення швидкості виконання.

– Активна спільнота: Python має велику і активну спільноту розробників, яка створює та підтримує широкий спектр модулів і інструментів.

Перераховані вище фактори є не повним списком переваг Python серед інших мов програмування, особливо в сфері науки. Але, всі ці чинники роблять цю мову неймовірно зручною для швидкої розробки як прототипів програм, так і для розробки повноцінних проектів.

Мова програмування Python, найкращий вибір для розробки нашої ІТ аналізу та передбачення цін вживаних автомобілів. Вибір зроблено, але варто розповісти про інші, користі, потужні та унікальні в своєму роді мови програмування та роботи з даними.

Мова програмування R – це інтерпретована мова програмування, яка використовується для статистичного аналізу даних. R має простий синтаксис і широкий спектр вбудованих функцій, що робить його доступним для початківців. Однак R також підтримує широкий спектр сторонніх бібліотек, які надають додаткові можливості для більш просунутих користувачів (рис. 2.3) [22].

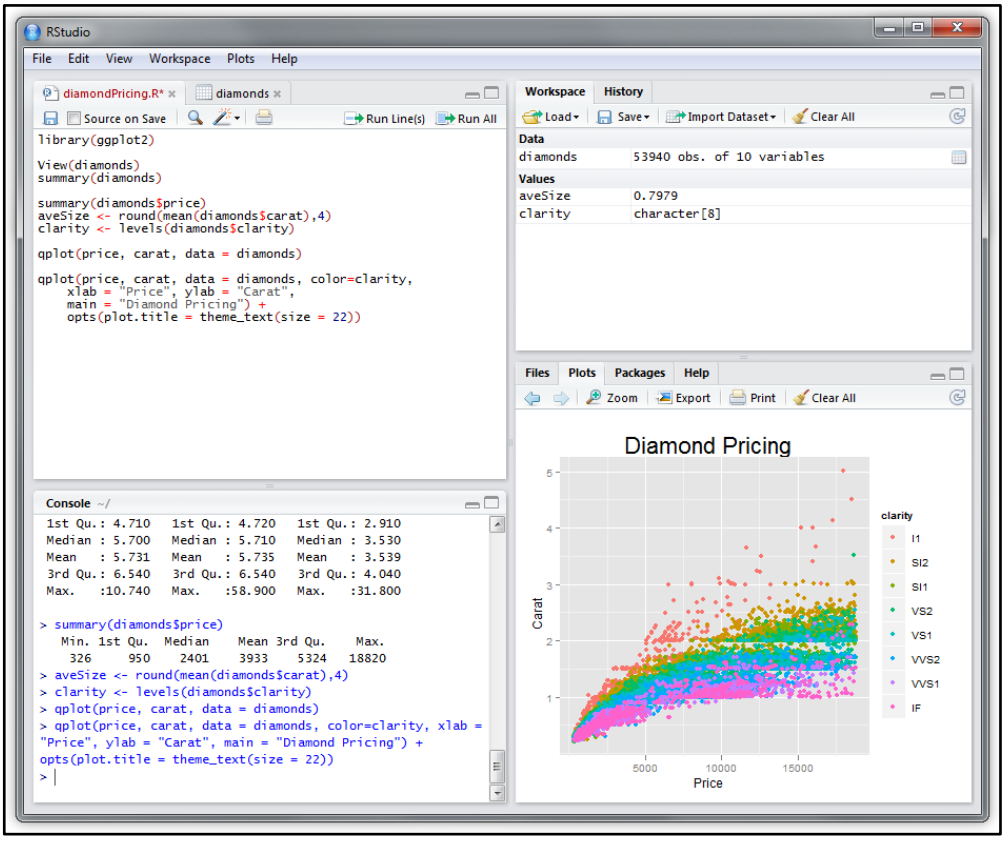

Рисунок 2.3 – Середовище розробки та синтаксис мови R

R часто порівнюють з MATLAB і Python, двома іншими популярними мовами програмування для статистичного аналізу даних. MATLAB є більш потужною мовою, ніж R, але також більш складною і дорогою. Python є більш всебічною мовою, ніж R, але не має такої спеціалізації в статистиці.

Переваги мови R:

- Простий синтаксис;
- Широкий спектр вбудованих функцій;
- Підтримка сторонніх бібліотек;
- Безкоштовна.

Недоліки мови R:

- Не так потужна, як MATLAB;
- Не така всебічна, як Python.

R є потужним і універсальним інструментом для статистичного аналізу даних. Він є хорошим вибором для початківців і просунутих користувачів.

Важливо згадати про SQL-подібні мови, оскільки вони часто використовуються для аналізу, фільтрації та обробки даних. Поєднання Python із SQL-подібними мовами може значно прискорити роботу із даними (рис. 2.4) [23].

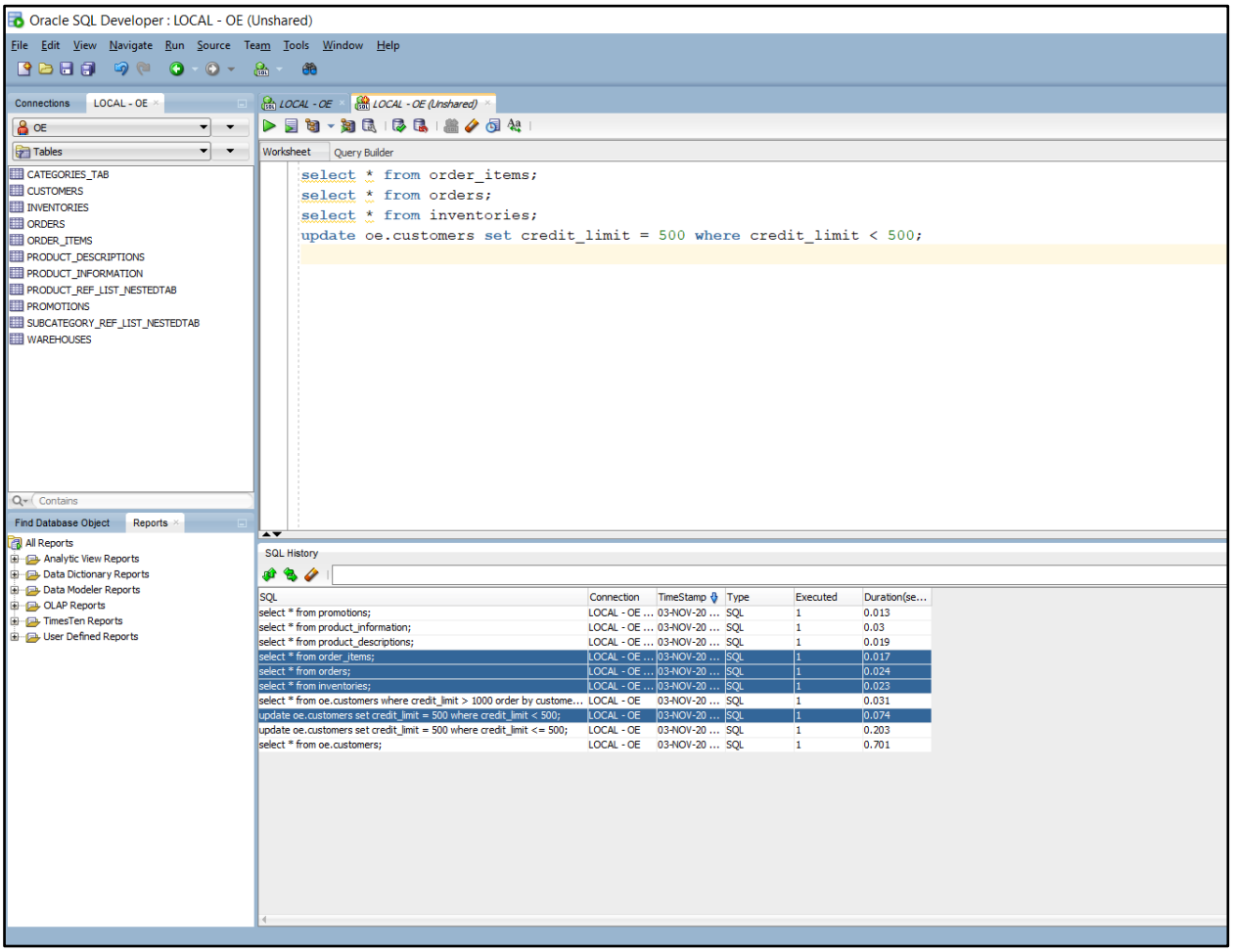

Рисунок 2.4 – Середовище розробки та синтаксис декларативної мови програмування SQL

Враховуючи універсальність, широку підтримку та потужні пакети та модулі для роботи з даними, для подальшої розробки інтелектуальної технології аналізу та передбачення цін на вживані автомобілі було обрано мову програмування Python.

Перед початком розробки програмного коду на Python необхідно також обрати середовище розробки та модулі для безпосередньої обробки даних та машинного навчання.

2.1.2 Вибір середовища розробки для ІТ аналізу та передбачення цін на вживані автомобілі

Немає кращого способу створювати програми на Python, ніж використовувати IDE (інтегроване середовище розробки). Вони не тільки роблять вашу роботу набагато легшою та логічнішою, але й покращують досвід кодування та ефективність (рис. 2.5) [24].

Звичайно, це знає кожен. Проте проблема полягає в тому, як вибрати найкраще середовище розробки, коли є так багато варіантів? Чи є один , такий редактор для Python, що буде кращим за усі інші? Ці питання часто стають проблемою, з якою стикаються початківці розробники. Проте, щоб вибрати найбільш підходящий варіант, потрібно знати, для чого він нам потрібен і над якими завданнями ми будете з ним працювати.

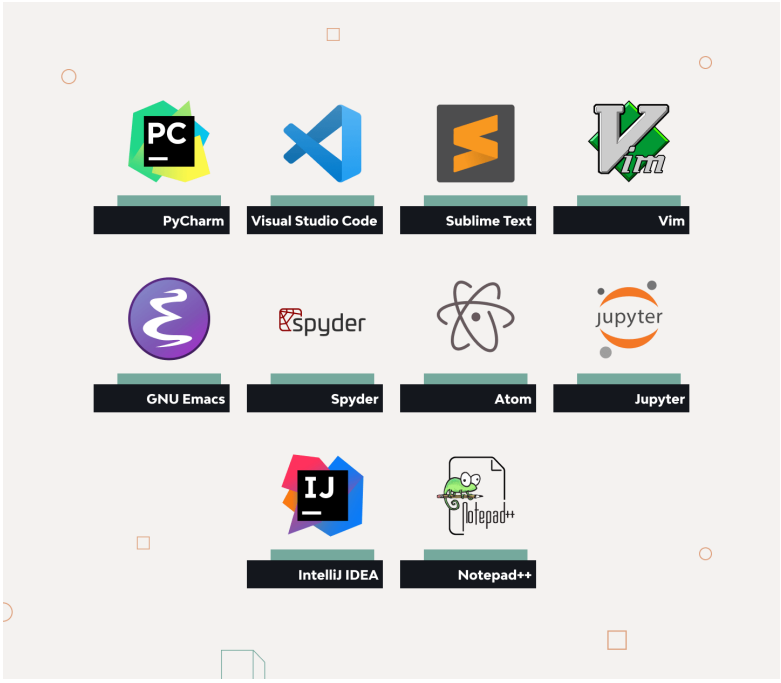

Рисунок 2.5 – Найкращі Python IDE для розробки

Вбудоване середовище розробки (IDE) — це програмний пакет, який розробники використовують для створення програм. Він призначений для підвищення продуктивності програміста шляхом поєднання тісно пов'язаних компонентів з простими інтерфейсами користувача. По суті, це інструмент, який покращує процес створення, тестування та налагодження вихідного коду — він робить роботу легшою. Простіше кажучи, ви можете зосередитися на важливіших речах, а IDE займеться всією рутиною [24].

Ось деякі з хороших інструментів Python IDE:

- Текстовий редактор;
- Компілятор та/або інтерпретатор;
- Засоби автоматизації збірки;
- Відладчик.

IDE є основним інструментом для роботи з кодом, тому вибір одного з них є дуже важливим. Він надає велику допомогу як при написанні, так і при читанні коду. Розробник повинен вибрати IDE на основі його функцій та завдань, які він повинен виконувати. Ось на що слід звернути увагу при пошуку найкращого текстового редактора для розробки на Python [25]:

- Простота використання, залежно від виконуваних завдань. Наприклад, навігація по коду, стилі кольорів коду та автозаповнення;
- Підтримка функцій та сервісів. Вони залежать від конкретного проекту чи потреб розробника. Наприклад, одному може знадобитися функція підсвічування синтаксису для конкретних мов, а іншому потрібна система тестування, відладчик, система контролю версій, інтеграція з Docker, доступ до терміналу тощо;
- Швидкість запуску IDE чи текстового редактора. Якщо IDE має багато функцій, але погано оптимізований, він буде запускатися, обробляти та доповнювати код дуже повільно, що є величезним недоліком;
- Екосистема. Важливо, щоб IDE для Python регулярно оновлювалося новими функціями та виправленнями помилок. Спільнота користувачів також має значення, оскільки вона дозволяє спілкуватися з розробниками,

пропонувати покращення функцій та надавати їм зворотний зв'язок у реальному часі. Хоча не всі розробники є користувачами, продукт повинен постійно оновлюватися та покращуватися, що майже неможливо без спільноти. В іншому випадку ви ризикуєте зіткнутися з обмеженнями застарілої версії, деяких технологій або цілої версії мови, яка не підтримується;

– Система плагінів. Зазвичай базові можливості IDE задовольняють понад 90% потреб розробника. Решта досягається за допомогою сторонніх плагінів;

– Ціна. Є кілька варіантів, які можна обрати: IDE з відкритим вихідним кодом, безкоштовна версія запатентованого продукту, одноразова оплата та підписка. Тут кожна компанія чи розробник робить свій власний вибір за співвідношенням ціни та якості. Тому, обираючи найкращий IDE для Python для Mac, Windows або Linux, ви будете поєднувати функції, які ви очікуєте від нього, та бюджет, який ви маєте на нього.

Існує термін, який є схожим на інтегроване середовище розробки — текстовий редактор. Давайте спочатку обговоримо відмінності та спільні риси обох інструментів.

Текстовий редактор для коду - це текстовий редактор, який підсвічує синтаксис та форматує код. Розширені текстові редактори для коду можуть розробляти та модифікувати код [24].

Які функції мають спільного IDE та текстові редактори для коду? Вони дозволяють розробникам:

– Зберігати та повторно відкривати скрипти;

– Запускати свій код на ньому;

– Відлагоджувати;

– Підсвічувати синтаксис.

Відмінні риси IDE та текстових редакторів для коду наведенно в таблиці 1.

Таблиця 1 – Відмінні риси IDE та текстових реакторів

Ознака	IDE	Текстовий редактор для коду
Надає високопродуктивні бібліотеки або інструментарій для розширеного кодування	Так	Ні
Автоматизований	Так	Ні
Широкий функціонал	Так	Менше функцій

Для написання програм на Python існує безліч середовищ, які можна класифікувати за функціональністю та сферою застосування.

Прості середовища, такі як Python IDLE та Thonny, підходять для початківців.

Середовища для веб-розробки, такі як VS Code та Atom, мають широкий функціонал і підходять для створення веб-додатків.

Середовища для наукових досліджень, такі як Spyder, PyCharm та Jupyter Notebook, мають спеціалізовані інструменти для роботи з даними [25].

Для поставленої задачі, яка полягає в дослідженні даних, побудові та тренуванні моделей, підходять будь-які з останніх трьох середовищ. Однак для максимальної швидкості та зручності було обрано Jupyter Notebook, який налаштований на платформі Kaggle (рис. 2.6) [26].

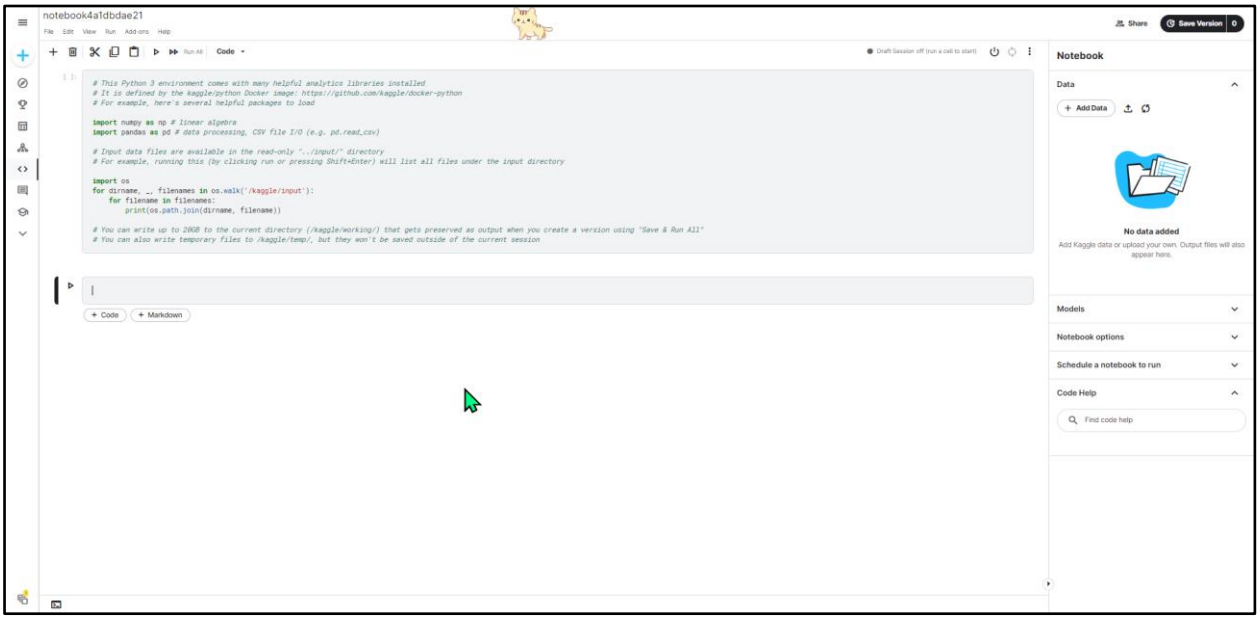

Рисунок 2.6 – Jupyter Notebook на платформі Kaggle

Jupyter Notebook має ряд переваг, зокрема:

- Зручний спосіб опису ходу дослідження.
- Підтримка від спільноти науковців.
- Можливість легко ділитися результатами дослідження.

Система Kaggle відкриває можливість проводити дослідження та виконувати код машинного навчання завдяки платформі Kaggle Notebooks. Це хмарне обчислювальне середовище, яке забезпечує відтворюваність та спільний аналіз проектів. Крім того, завдяки Kaggle Notebooks, ви можете запускати код на хмарному обчислювальному середовищі, яке має виділені ресурси оперативної пам'яті, а також можливість зберігати до майже 20 ГБ вихідних даних з ноутбука на диск у /kaggle/working. Ці дані автоматично зберігаються і можуть бути використані для подальших проектів [26].

Kaggle Notebooks - це більше, ніж просто інтерактивний редактор коду. Це обчислювальне середовище, створене для спрощення наукової роботи з даними. В Kaggle Notebooks ви отримуєте доступ до інтерактивних сеансів в контейнері Docker з попередньо встановленими пакетами. Ви можете змінювати версійні джерела даних та налаштовувати ресурси обчислення, такі як графічні процесори.

Однією з важливих переваг Kaggle є можливість спільної розробки коду та проектів. Ви можете запрошувати інших користувачів стати співавторами та надавати їм дозвіл на редагування коду в ноутбуці.

Однак найбільш вагомою перевагою Kaggle є доступ до широкого вибору наборів даних у вільному доступі, які можна використовувати для проектів. Ця можливість дозволяє користувачам працювати з різноманітними даними, обговорювати їх та вдосконалювати вже існуючі проекти або створювати власні.

Таким чином, для виконання завдання щодо передбачення цін на продаж будинків було обрано мову програмування Python як найбільш підходящу для роботи з методами машинного навчання. Для розробки програмного коду на Python було обрано середовище Kaggle IDE Notebooks та безкоштовні хмарні ресурси для обчислень, які виявилися повністю достатніми для виконання завдання.

2.1.3 Визначення виду та вибір інструментів машинного навчання

Завдання, яке було сформульоване в попередньому розділі, потребує ретельнішого розгляду, щоб уточнити вигляд нашої майбутньої програми та визначити, які методи машинного навчання слід використовувати. Перед початком фактичної розробки, необхідно зрозуміти певні терміни та концепції, пов'язані з аналізом даних і машинним навчанням.

Машинне навчання означає створення математичних моделей на основі дослідження даних. Процес "навчання" починається з налаштування параметрів моделей, які можна адаптувати для відображення даних. У суті, програма "навчається" на даних. Якщо ці моделі навчаться на доступних даних спостережень, їх можна використовувати для прогнозування та розуміння різних аспектів нових наборів даних [27].

Перш ніж приступати до роботи з машинним навчанням, необхідно мати базові знання та розуміння теорії ймовірностей та математичної статистики.

Також важливо мати навички роботи з диференціюванням складних функцій та матриць, лінійною алгеброю і, як мінімум, бути здатними працювати з мовою програмування Python.

Машинне навчання можна узагальнити у три основні категорії задач [28]:

– Задачі класифікації: Це ситуації, де потрібно призначити вхідні дані до певного класу або категорії. Наприклад, це може включати розділення зображень на котів та собак, розпізнавання вимовленого слова або встановлення діагнозу на основі медичних даних. Всі ці завдання відносяться до категорії класифікації.

– Задачі регресії: Ці завдання використовуються для прогнозування числових значень на основі вхідних даних. Наприклад, це може бути передбачення обмінного курсу валют на основі історичних даних або прогнозування обсягу продажу певного товару. Іншими словами, регресія допомагає побудувати прогнози числових значень на основі наявних даних.

– Задачі ранжування: Ця категорія задач передбачає впорядкування вхідного набору даних відповідно до певного критерію. Класичним прикладом є пошукові системи, які ранжують результати пошуку за релевантністю для кожного конкретного користувача та запиту. Задачі ранжування спрямовані на впорядкування даних з урахуванням конкретних вимог.

Всі ці задачі вимагають використання алгоритмів машинного навчання, оскільки вхідні дані мають складну структуру, і для їх обробки потрібно розробити складні критерії. Людям, зазвичай, важко створити ці критерії вручну, тому саме тут машинне навчання входить в гру, навчаючись на заздалегідь підготовленому наборі даних.

У машинному навчанні існують три основні та один додатковий компоненти:

– Дані: Це вихідні результати вимірювань, які становлять основу для роботи з машинним навчанням.

– Признаки (features): Це важливі характеристики або властивості, які виділяються з даних для подальшого аналізу та використання.

– Алгоритм: Це програмне забезпечення, яке може навчатись на основі вхідних даних та виконувати завдання, для яких воно було навчене.

Ці компоненти разом створюють основу для процесу машинного навчання, де дані служать джерелом інформації, признаки допомагають виокремити важливі аспекти цих даних, а алгоритми забезпечують аналіз та виконання завдань на основі навчання.

Алгоритми навчання розділяються на різні категорії відповідно до наступних основних видів (рис. 2.7) [27]:

– Навчання з учителем (supervised learning): Цей підхід включає моделювання ознак даних разом з відповідними мітками. Після побудови моделі її можна використовувати для призначення міток новим, раніше невідомим даним. Навчання з учителем поділяється на завдання класифікації і завдання регресії. При класифікації мітки представляють собою дискретні категорії, тоді як при регресії вони є безперервними величинами.

– Навчання без учителя (unsupervised learning): Цей підхід включає моделювання ознак набору даних без будь-яких міток та описується фразою "Нехай набір даних говорить сам за себе". Ці моделі включають такі завдання, як кластеризація (clustering) та зниження розмірності (dimensionality reduction). Алгоритми кластеризації служать для виділення окремих груп даних, у той час як алгоритми зниження розмірності призначені для створення більш лаконічних представлень даних.

– Навчання із підкріпленням (Reinforcement Learning): Ця категорія ґрунтується на математичних основах теорії ігор. Алгоритм може виконувати певні дії, змінювати навколишнє середовище і отримувати сигнали від зовнішнього середовища, на основі яких обчислюється кінцева нагорода. Головною метою навчання із підкріпленням є навчання алгоритму здійснювати дії, які призводять до найбільшої нагороди.

– Часткове навчання (Semi-supervised Learning): Цей метод розташований приблизно посередині між навчанням з учителем і навчанням без учителя.

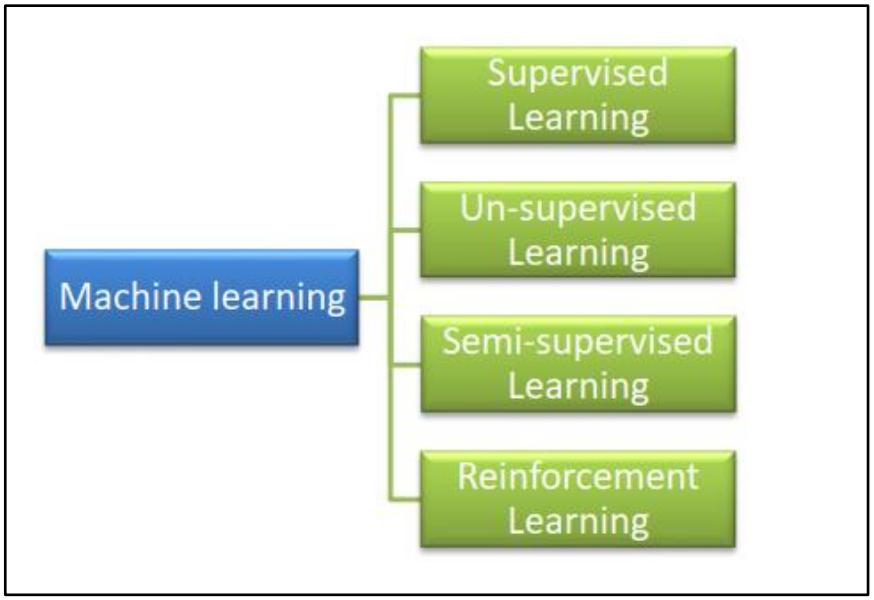

Рисунок 2.7 – Машинне навчання та його складові

Класифікація та регресія - це завдання, які вирішуються шляхом використання контрольованого навчання, відомого як навчання з "учителем".

В даному випадку людина, яка має знання про ціни будинків, є "вчителем" для машини. Вчитель навчає машину, проставляючи "мітки" для кожного будинку. Мітка - це ціна будинку. Машина сама обирає, які характеристики будинку використовувати для передбачення ціни. Алгоритм, який навчила машина, можна використовувати для вирішення інших задач, які мають схожий характер.

Класифікація - це задача, в якій машина навчається розділяти об'єкти на категорії. Це схоже на те, як дитина вчиться розрізняти предмети за їхніми характеристиками [28].

Класифікація - це задача, в якій машина навчається визначати категорію об'єкта на основі набору ознак. Машина використовує ці ознаки для сортування даних за категоріями. Але при класифікації, все зводиться до

передбачення бінарних ознак, наприклад, «Чи був проданий автомобіль», так – 1, або ні – 0.

Ціна автомобіля - це не бінарна ознака, оскільки вона може приймати будь-яке значення. Тому класифікатор не є тим методом, який підходить для передбачення ціни вживаного автомобіля [29].

Але регресія є тим самим методом для передбачення ціни вживаного автомобіля, оскільки вона може прогнозувати значення змінної на основі набору ознак.

Регресія - це задача машинного навчання, в якій машина навчається прогнозувати значення змінної на основі набору ознак. Регресія може працювати лише з числовими даними [28].

Регресія може використовуватися для прогнозування чисельних величин, таких як ціна автомобіля або обсяг ринку, тощо.

В даному випадку машина навчається на наборі даних, в якому ціни вживаних автомобілів вже відомі. Цей метод називається машинним навчанням з учителем.

Регресія та дерева рішень – це, саме ті, важливі, два підходи до вирішення задач регресії. Обидва вони можуть бути ефективними для передбачення ціни на вживані автомобілі.

2.1.4 Опис та вибір моделей машинного навчання

Багато моделей машинного навчання можуть бути використані для вирішення даного завдання. Однак для цієї реалізації було обрано 12 регресійних моделей, а саме:

- Linear Regression;
- Support Vector Machines;
- Linear SVR;
- Stochastic Gradient Decent;
- Decision Tree Regressor;

- RandomForestRegressor;
- XGBoostRegressor;
- LGBM;
- GradientBoostingRegressor;
- RidgeRegressor;
- BaggingRegressor;
- ExtraTreesRegressor;
- AdaBoostRegressor;
- VotingRegressor.

Linear Regression. Лінійна регресія представляє собою простий і основний метод статистичної регресії, який використовується в машинному навчанні для прогнозного аналізу. Цей метод виявляє лінійний зв'язок між незалежною (прогнозною) змінною, яка позначена як X, і залежною (вихідною) змінною, позначеною як Y. Цей вид регресії відомий як лінійна регресія. Якщо є лише одна вхідна змінна X (незалежна змінна), то таку лінійну регресію називають простою лінійною регресією [30].

Вигляд лінійної залежності між вихідними та предикторними змінними, зображено на рисунку 2.8.

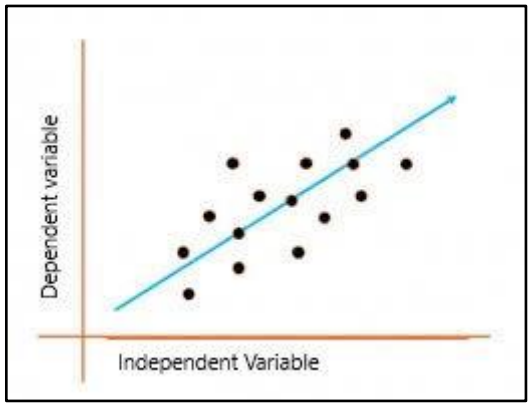

Рисунок 2.8 – Лінійна залежність між вихідними та предикторними змінними

На рисунку 2.8 показано синю лінію, яку називаються найкращою «прямою» лінією . На основі заданих точок даних ми намагаємося побудувати лінію, яка найкраще відповідає точкам.

Support Vector Machines (SVM) представляють собою контрольовані моделі навчання з відповідними алгоритмами, які аналізують дані для здійснення класифікації та регресійного аналізу. При наявності набору навчальних вибірок, де кожна позначена як належна до однієї з двох категорій, алгоритм навчання SVM формує модель, що призначає нові тестові вибірки до однієї з категорій, що робить його ефективним двійковим лінійним класифікатором [31].

Linear SVR. Регресія опорних векторів (SVR) — це тип машини опорних векторів (SVM), яка використовується для завдань регресії. Він намагається знайти функцію, яка найкраще прогнозує безперервне вихідне значення для заданого вхідного значення [32].

SVR може використовувати як лінійні, так і нелінійні ядра. Лінійне ядро — це простий скалярний добуток між двома вхідними векторами, тоді як нелінійне ядро — це складніша функція, яка може вловлювати складніші моделі в даних. Вибір ядра залежить від характеристик даних і складності завдання.

Stochastic Gradient Decent. Градієнтний спуск — це ітеративний процес оптимізації, який шукає оптимальне значення цільової функції (мінімальне/максимальне). Це один із найбільш використовуваних методів зміни параметрів моделі з метою зменшення функції витрат у проектах машинного навчання [33].

Основною метою градієнтного спуску є визначення параметрів моделі, які забезпечують максимальну точність як для навчальних, так і для тестових наборів даних. У градієнтному спуску градієнт — це вектор, що вказує в загальному напрямку найкрутішого зростання функції в певній точці. Алгоритм може поступово опускатися до нижчих значень функції, рухаючись у протилежному напрямку градієнта, до досягнення мінімуму функції.

Як правило, існує три типи градієнтного спуску:

- Пакетний градієнтний спуск;
- Стохастичний градієнтний спуск;
- Міні-серія Градієнтний спуск.

Стохастичний градієнтний спуск (SGD) — це варіант алгоритму градієнтного спуску , який використовується для оптимізації моделей машинного навчання. Він усуває обчислювальну неефективність традиційних методів градієнтного спуску при роботі з великими наборами даних у проектах машинного навчання [33].

У SGD замість використання всього набору даних для кожної ітерації вибирається лише один випадковий приклад навчання (або невелика партія) для обчислення градієнта та оновлення параметрів моделі. Цей випадковий вибір вводить випадковість у процес оптимізації, тому термін «стохастичний» у стохастичному градієнтному спуску

Перевагою використання SGD є його обчислювальна ефективність, особливо при роботі з великими наборами даних. Використовуючи один приклад або невелику партію, обчислювальна вартість однієї ітерації значно знижується порівняно з традиційними методами градієнтного спуску, які вимагають обробки всього набору даних.

Приклад шляху пройденого стохастичним градієнтним спуском зображено на рисунку 2.9.

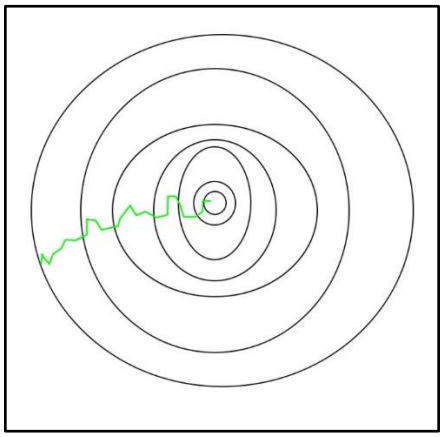

Рисунок 2.9 – Шлях пройдений стохастичним градієнтним спуском

Одна річ, яку слід зазначити, полягає в наступному, оскільки SGD, як правило, шумніший, ніж типовий градієнтний спуск, зазвичай потрібна більша кількість ітерацій, щоб досягти мінімумів, через випадковість його спуску. Незважаючи на те, що для досягнення мінімумів потрібна більша кількість ітерацій, ніж звичайний градієнтний спуск, він все одно набагато дешевший з обчислювальної точки зору, ніж звичайний градієнтний спуск. Отже, у більшості сценаріїв для оптимізації алгоритму навчання перевагу надають SGD, а не пакетному градієнтному спуску.

Decision Tree Regressor. Дерево рішень - це інструмент для прийняття рішень, який використовує структуру дерева, схожу на блок-схему, або це модель рішень та всіх їх можливих результатів, включаючи наслідки, витрати на вхідні дані та корисність [34].

Алгоритм дерева рішень належить до категорії алгоритмів керованого навчання. Він працює як для неперервних, так і для категоріальних вихідних змінних.

Гілки/ребра представляють результат вузла, а вузли мають або:

- Умови [Вузли рішень];
- Результат [Кінцеві вузли].

Гілки/ребра представляють істинність/хибність твердження і приймають рішення на основі цього, як у наведеному нижче прикладі, який показує дерево рішень, яке оцінює найменше з трьох чисел (рис. 2.10).

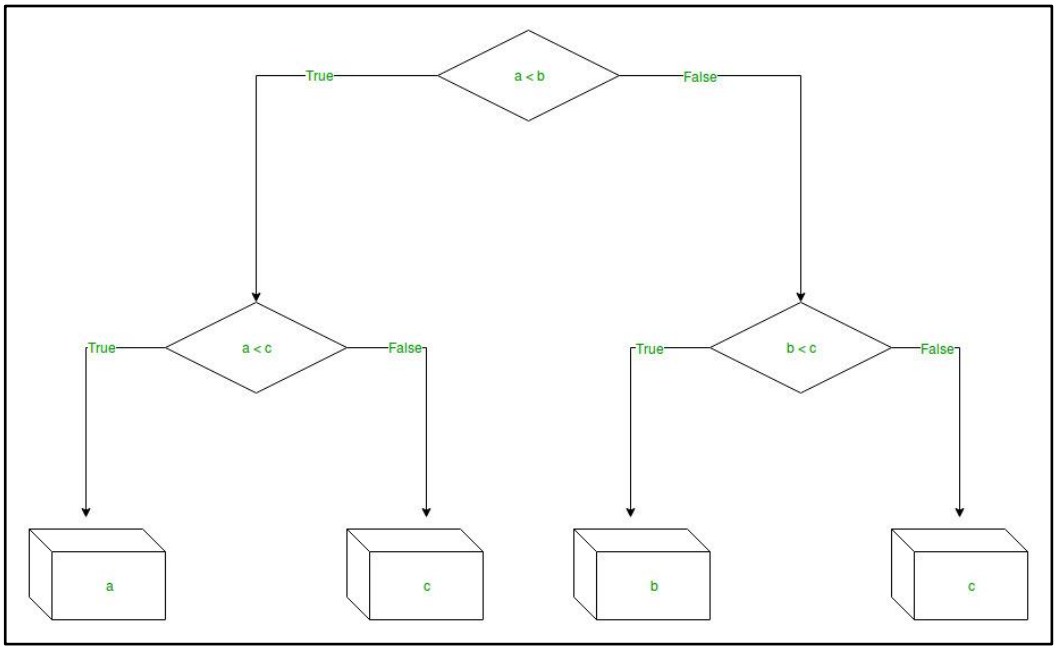

Рисунок 2.10 – Зображення дерева рішень для оцінки найменшого з трьох чисел

Регресія на дереві рішень. Регресія на дереві рішень спостерігає за ознаками об'єкта і навчає модель у структурі дерева для прогнозування даних у майбутньому для отримання значущого безперервного виходу. Безперервний вихід означає, що вихід/результат не є дискретним, тобто він не представлений лише дискретним, відомим набором чисел або значень [34].

Приклад дискретного виходу: модель прогнозування погоди, яка прогнозує, чи буде дощ у конкретний день.

Приклад безперервного виходу: модель прогнозування прибутку, яка визначає ймовірний прибуток, який може бути отриманий від продажу продукту.

Тут безперервні значення прогнозуються за допомогою моделі регресії на дереві рішень.

RandomForestRegressor. Кожне дерево рішень має високу дисперсію, але коли ми об’єднуємо їх усі разом паралельно, результуюча дисперсія є низькою, оскільки кожне дерево рішень ідеально навчено на цій конкретній вибірці даних, і, отже, результат залежить не від одного дерева рішень, а від

кілька дерев рішень. У разі проблеми з класифікацією кінцевий результат отримується за допомогою класифікатора більшості голосів. У випадку задачі регресії кінцевий результат є середнім значенням усіх результатів. Ця частина називається агрегацією [35].

Випадковий ліс — це метод ансамблю , здатний виконувати задачі як регресії, так і класифікації з використанням кількох дерев рішень і методу під назвою Bootstrap і Aggregation, широко відомого як bagging . Основна ідея цього полягає в тому, щоб об’єднати кілька дерев рішень для визначення кінцевого результату, а не покладатися на окремі дерева рішень (рис. 2.11).

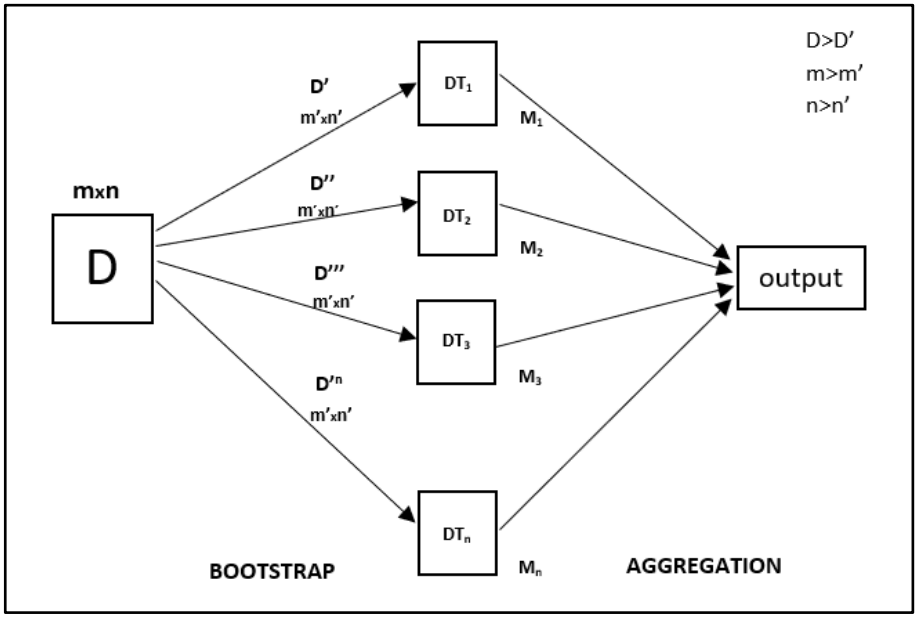

Рисунок 2.11 – Працює регресійна модель випадкового лісу

Random Forest має кілька дерев рішень як базові моделі навчання. Ми випадково виконуємо вибірку рядків і ознак із набору даних, формуючи вибіркові набори даних для кожної моделі. Ця частина називається Bootstrap.

XGBoost – це метод ансамблевого навчання, який поєднує прогнози кількох слабких моделей для створення більш точного прогнозу. XGBoost розшифровується як «Extreme Gradient Boosting» і став одним із найпопулярніших і широко використовуваних алгоритмів машинного навчання, завдяки своїй здатності обробляти великі набори даних і досягати

найсучаснішої продуктивності в багатьох завданнях машинного навчання, таких як класифікація та регресія [36].

LGBM. LightGBM — це структура для посилення градієнта, заснована на деревах рішень для підвищення ефективності моделі та зменшення використання пам'яті [37].

Він використовує дві нові техніки:

- Одностороння вибірка на основі градієнта (GOSS);
- Ексклюзивний пакет функцій (EFB).

Ці методи відповідають обмеженням алгоритму на основі гістограми, який в основному використовується в усіх структурах GBDT (Gradient Boosting Decision Tree). Дві методики GOSS і EFB, описані нижче, формують характеристики алгоритму LightGBM. Вони поєднуються разом, щоб зробити модель ефективнішою та надати їй передову перевагу над іншими фреймворками GBDT

Архітектура LightGBM.

LightGBM розбиває дерево на листи, на відміну від інших алгоритмів посилення, які ростуть на рівні дерева. Він вибирає для вирощування лист із максимальною втратою дельти. Оскільки аркуш є фіксованим, листковий алгоритм має менші втрати порівняно з порівневим алгоритмом. Зростання дерева по листах може збільшити складність моделі та призвести до переобладнання невеликих наборів даних (рис 2.12).

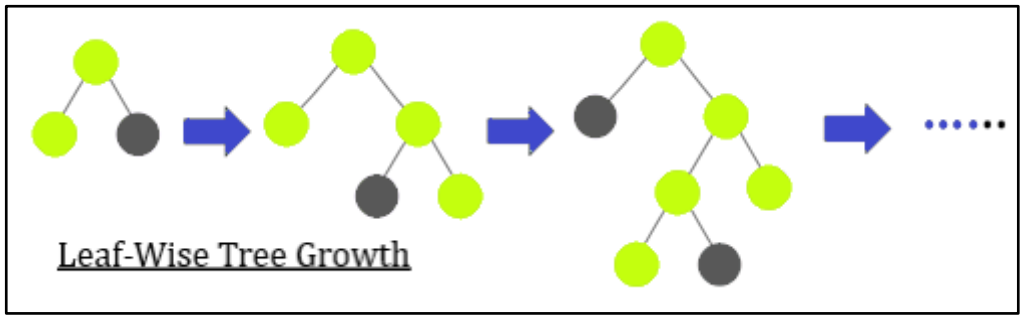

Рисунок 2.12 – Архітектура LightGBM

GradientBoostingRegressor. Gradient Boosting — це популярний алгоритм підвищення в машинному навчанні, який використовується для завдань класифікації та регресії. Підвищення — це один із методів ансамблевого навчання , який послідовно навчає модель, і кожна нова модель намагається виправити попередню модель. Він поєднує кількох слабких учнів у сильних. Існує два найпопулярніших алгоритму підвищення, а саме:

1. AdaBoost;

2. Gradient boosting.

Gradient Boosting — це потужний алгоритм підвищення, який об’єднує кількох слабких учнів у сильних, у якому кожна нова модель навчена мінімізувати функцію втрат, таку як середня квадратична помилка або крос-ентропія попередньої моделі за допомогою градієнтного спуску. На кожній ітерації алгоритм обчислює градієнт функції втрат щодо прогнозів поточного ансамблю, а потім навчає нову слабку модель, щоб мінімізувати цей градієнт. Прогнози нової моделі потім додаються до ансамблю, і процес повторюється, доки не буде виконано критерій зупинки [38].

На відміну від AdaBoost , ваги екземплярів навчання не змінюються, замість цього кожен предиктор навчається з використанням залишкових помилок попередника як міток. Існує техніка під назвою Gradient Boosted Trees , базовим учням якої є CART (дерева класифікації та регресії). На наведеній нижче діаграмі (рис. 2.13) пояснюється, як підсилені градієнтом дерева навчаються для проблем регресії.

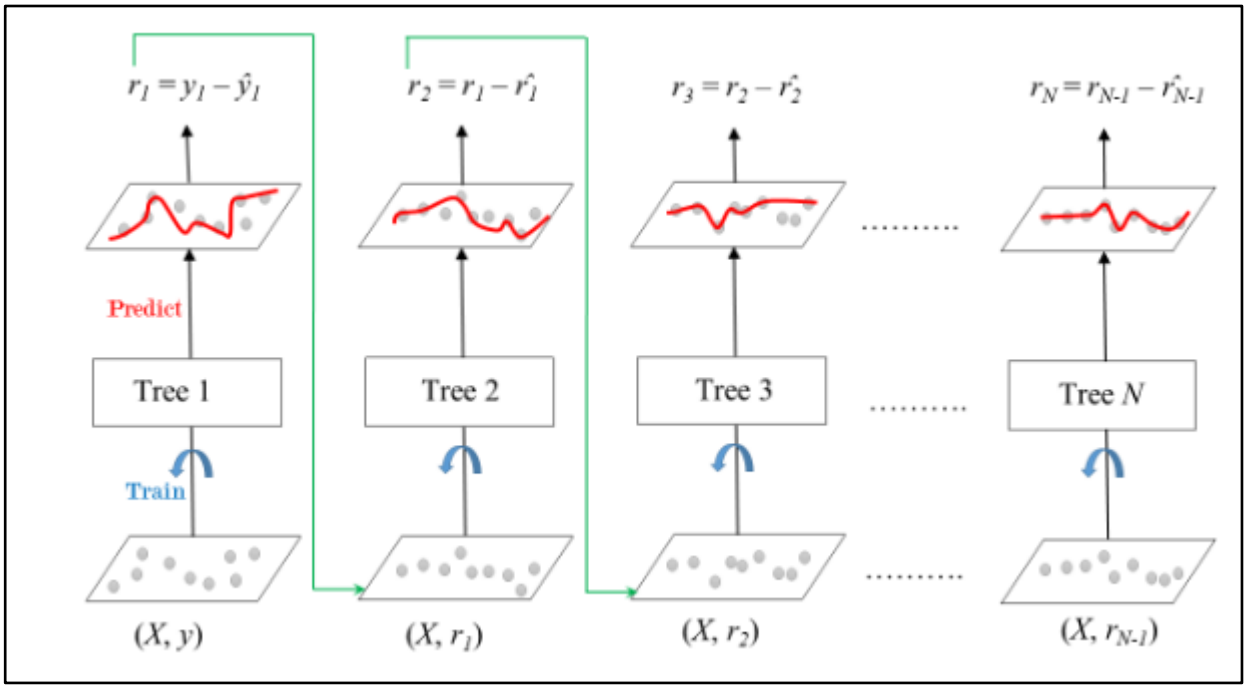

Рисунок 2.13 – Дерева з посиленням градієнта для регресії

Ансамбль складається з M дерев. Tree1 навчається за допомогою матриці ознак X і міток y .Прогнози, позначені y1(hat), використовуються для визначення залишкових помилок r1 навчального набору .Потім Tree2 навчається з використанням матриці ознак Xi залишкових помилок r1 Tree1 як міток. Прогнозовані результати r1(hat) потім використовуються для визначення залишку r2. Процес повторюється, доки всі M дерев, що утворюють ансамбль, не будуть навчені. У цій техніці використовується важливий параметр, відомий як усадка. Усадкавідноситься до того факту, що передбачення кожного дерева в ансамблі скорочується після того, як воно помножено на швидкість навчання (eta), яка коливається від 0 до 1. Існує компроміс між eta та кількістю оцінювачів, зменшуючи швидкість навчання необхідно компенсувати збільшенням оцінок, щоб досягти певної продуктивності моделі. Оскільки зараз усі дерева навчені, можна робити прогнози. Кожне дерево передбачає мітку, а остаточний прогноз дається за формулою,

RidgeRegressor. Регресор Ріджа в основному є регуляризованою версією лінійного регресора, тобто до вихідної функції вартості лінійного регресора

ми додаємо регуляризований член, який змушує алгоритм навчання відповідати даним і допомагає зберегти ваги якомога нижчими. Регуляризований член має параметр «альфа», який контролює регуляризацію моделі, тобто допомагає зменшити дисперсію оцінок [39].

BaggingRegressor. Ансамбль означає групу елементів, які розглядаються як ціле, а не окремо. Метод Ensemble створює кілька моделей і поєднує їх для вирішення. Методи ансамблю допомагають покращити надійність/узагальненість моделі. У цій статті ми обговоримо деякі методи з їх реалізацією в Python. Для цього ми вибираємо набір даних із сховища UCI [40].

Bagging: це також відомий як метод завантаження. Базові моделі працюють на пакетах, щоб отримати справедливий розподіл усього набору даних. Сумка — це підмножина набору даних разом із заміною, щоб зробити розмір сумки таким самим, як і весь набір даних. Остаточний вихід формується після об'єднання результатів усіх базових моделей.

ExtraTreesRegressor. Цей клас реалізує метаоцінювач, який підходить для ряду рандомізованих дерев рішень (так званих додаткових дерев) на різних підвибірках набору даних і використовує усереднення для покращення точності прогнозування та контролю перенавчання(overfitting) [41].

У випадкових лісах кожне дерево в ансамблі будується із зразка, взятого із заміною (тобто початкового зразка) із навчального набору.

Крім того, під час розбиття кожного вузла під час побудови дерева найкраще розбиття визначається або з усіх вхідних функцій, або з випадкової підмножини розміром max_features.

Метою цих двох джерел випадковості є зменшення дисперсії оцінки лісу. Дійсно, окремі дерева рішень зазвичай виявляють високу дисперсію та мають тенденцію до перепідбору. Впроваджена випадковість у лісах дає дерева рішень із дещо роз’єднаними помилками передбачення. Взявши середнє значення цих прогнозів, деякі помилки можна нівелювати. Випадкові ліси досягають зменшеної дисперсії шляхом комбінування різноманітних

дерев, іноді ціною незначного збільшення зміщення. На практиці зменшення дисперсії часто є значним, отже, дає загальну кращу модель.

AdaBoostRegressor. Скорочення AdaBoost від Adaptive Boosting — це ансамблеве навчання, яке використовується в машинному навчанні для задач класифікації та регресії. Основна ідея AdaBoost полягає в ітераційному навчанні слабкого класифікатора на навчальному наборі даних, причому кожен наступний класифікатор надає більшої ваги точкам даних, які неправильно класифіковані. Остаточна модель AdaBoost визначається шляхом об'єднання всіх слабких класифікаторів, які використовувалися для навчання, з вагою, наданим моделям відповідно до їхньої точності. Слабкій моделі з найвищою точністю надається найвища вага, тоді як моделі з найнижчою точністю надається менша вага [42].

Технології AdaBoost поєднують багато слабких моделей машинного навчання для створення потужної моделі класифікації результату (рис. 2.14).

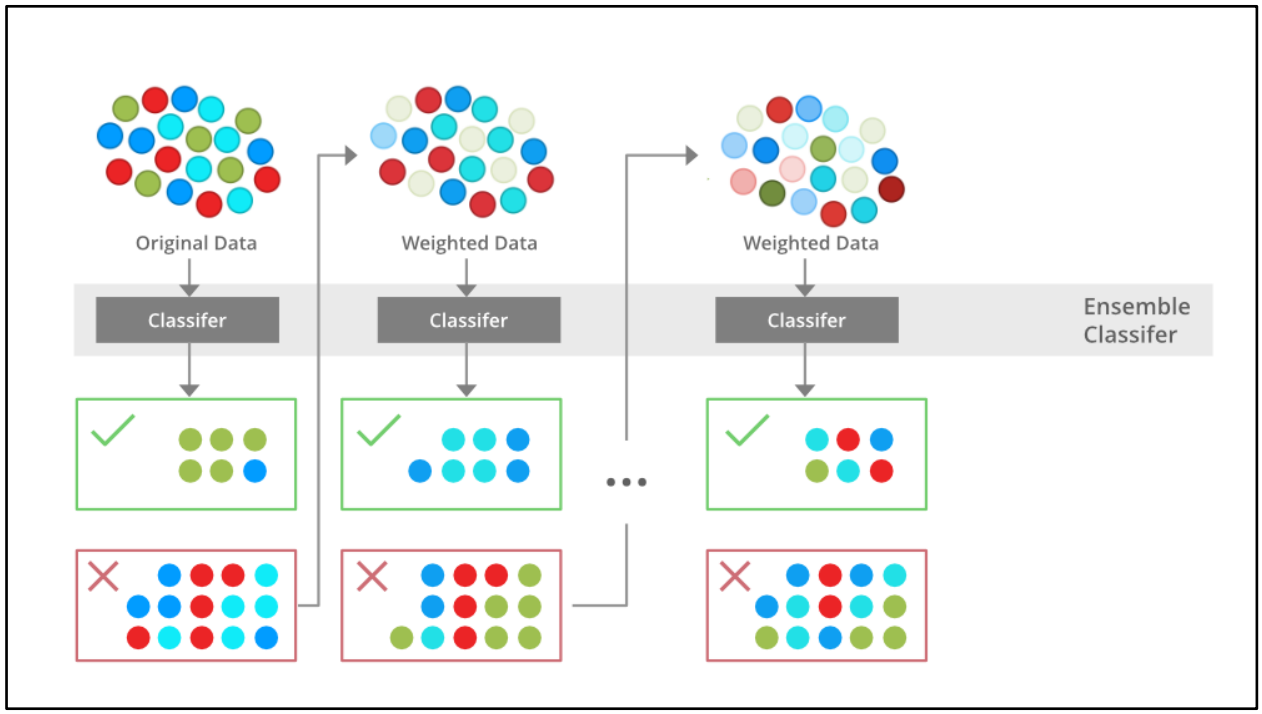

Рисунок 2.14 – Алгоритм підвищення

Етапи побудови та комбінування цих моделей такі:

Крок 1 – Ініціалізація ваг

– Для набору даних із N екземплярів навчальних точок даних ініціалізуйте NW_{i} вагових коефіцієнтів для кожної точки даних за допомогою W_{i} = \frac{1}{N}

Крок 2 – Навчіть слабкі класифікатори

– Навчіть слабкий класифікатор M_k, де k — поточна ітерація

– Слабкий класифікатор, який ми навчаємо, повинен мати точність більше 0,5, що означає, що він повинен працювати краще, ніж наївне припущення

Крок 3 – Обчисліть частоту помилок і важливість кожної слабкої моделі M_k [42]

– Обчисліть швидкість error_rate для кожного слабкого класифікатора M_k у навчальному наборі даних

– Обчисліть важливість кожної моделі α_k за формулою \alpha_k = \frac{1}{2} \ln{\frac{1 – error_k}{error_k}}

Крок 4 – Оновіть вагу точки даних для кожної точки даних W_i

– Після застосування моделі слабкого класифікатора до навчальних даних ми оновимо вагу, призначену точкам, використовуючи точність моделі. Формула для оновлення ваг буде такою: w_i = w_i \exp{(-\alpha_k y_i M_k(x_i))}. Тут y_i— справжній вихід, а X_i — відповідний вхідний вектор

Крок 5 – нормалізуйте вагу екземпляра

– Ми нормалізуємо вагу екземпляра, щоб їх можна було підсумувати до 1 за формулою W_i = W_i / sum(W)

Крок 6 – повторіть кроки 2-5 для K ітерацій

– Ми навчимо K класифікаторів і розрахуємо важливість моделі та оновимо ваги екземплярів за допомогою наведеної вище формули

– Остаточна модель M(X) буде моделлю ансамблю, яка отримана шляхом об’єднання цих слабких моделей, зважених за їхніми вагами моделей

VotingRegressor Класифікатор голосування — це модель машинного навчання, яка навчається на сукупності численних моделей і передбачає вихід

(клас) на основі їх найвищої ймовірності обраного класу як результату. Він просто агрегує результати кожного класифікатора, переданого в Класифікатор голосування, і прогнозує вихідний клас на основі найбільшої більшості голосів. Ідея полягає в тому, що замість створення окремих спеціалізованих моделей і пошуку точності для кожної з них ми створюємо єдину модель, яка навчається за цими моделями та прогнозує результат на основі їх спільної більшості голосів для кожного класу результатів [43].

Класифікатор голосування підтримує два типи голосувань.

– Жорстке голосування: під час жорсткого голосування прогнозований вихідний клас – це клас із найбільшою більшістю голосів, тобто клас, який мав найвищу ймовірність бути передбаченим кожним із класифікаторів. Припустімо, що три класифікатори передбачили вихідний клас (A, A, B), тож тут більшість передбачила A як вихідний. Отже, A буде остаточним прогнозом.

– М'яке голосування: у м'якому голосуванні клас вихідних даних є прогнозом на основі середньої ймовірності, наданої цьому класу. Припустімо, що отримано певні дані для трьох моделей, імовірність прогнозу для класу A = (0,30, 0,47, 0,53) і B = (0,20, 0,32, 0,40). Таким чином, середнє значення для класуA становить 0,4333, а B – 0,3067. Переможцем є клас A, оскільки він мав найвищу ймовірність, середню для кожного класифікатора.

2.1.5 Вибір та опис Python бібліотек

В мові програмування Python існує неймовірно велика кількість бібліотек, які можуть бути використані для реалізації різних завдань. Однак, щоб уникнути зайвої складності та залежності від сторонніх модулів, у даній магістерській роботі для реалізації основної задачі будуть використовуватися наступні модулі:

NumPy - це бібліотека для роботи з матрицями та тензорами. Вона надає широкий спектр функцій для створення, обробки та обчислення, що пов’язані

з матрицями, включаючи операції над векторами, скалярами, матрицями та тензорами, а також операції над матрицями, такі як додавання, віднімання, множення, ділення, транспонування, квадратний корінь, визначник та інші. NumPy також надає функції для генерування випадкових чисел, роботи з файлами та графіками [44].

Pandas - це бібліотека для роботи з даними у вигляді таблиць. Вона надає широкий спектр функцій для читання, запису, обробки та аналізу даних у вигляді таблиць, включаючи операції над даними, такими як сортування, фільтрація, об'єднання, злиття, обчислення статистики та ін. Pandas також надає функції для візуалізації даних та роботи з часом [45].

Seaborn - це бібліотека для візуалізації даних на основі Matplotlib. Вона надає широкий спектр функцій для створення естетичних і інформативних графіків, включаючи графіки ліній, стовпчасті графіки, кругові графіки, гістограми, розкиди та ін. Seaborn також надає функції для візуалізації даних у вигляді карт, 3D-графіків та ін. [46].

Matplotlib - це бібліотека для візуалізації даних. Вона надає широкий спектр функцій для створення графіків, діаграм, карт та інших візуальних представлень даних, включаючи лінії, стовпці, кола, графіки розкиду, гістограми, 3D-графіки та ін. Matplotlib також надає функції для інтерактивної візуалізації даних [47].

Scikit-learn - це бібліотека для машинного навчання. Вона надає широкий спектр алгоритмів машинного навчання, включаючи класифікацію, регресію, кластеризацію, розпізнавання образів та ін. Scikit-learn також надає функції для оцінки алгоритмів машинного навчання та обробки результатів машинного навчання [48].

SciPy - це бібліотека для наукових обчислень. Вона надає широкий спектр функцій для обчислень, включаючи числеві методи, обробку сигналів, обробку зображень, обробку даних та ін. SciPy також надає функції для візуалізації даних [49].

LightGBM - це бібліотека для швидкого та ефективного навчання лісів рішення. Вона є популярною бібліотекою для машинного навчання, яка використовується для різних завдань, включаючи класифікацію, регресію та кластеризацію. LightGBM є особливо ефективною для завдань, де кількість даних велика [50].

XGBoost - це бібліотека для швидкого та ефективного навчання лісів рішень. Вона є популярною бібліотекою для машинного навчання, яка використовується для різних завдань, включаючи класифікацію, регресію та кластеризацію. XGBoost є особливо ефективною для завдань, де кількість даних велика і є деяка нерівномірність у даних [51].

Folium - це бібліотека для створення інтерактивних карт на основі HTML, CSS та JavaScript. Вона надає широкий спектр функцій для створення карт, включаючи додавання маркерів, ліній, областей, тексту та ін. Folium також надає функції для інтерактивної взаємодії з картами [52].

2.2 Набір даних та опис його ознак

На етапі вибору набору даних для аналізу, обробки даних та моделювання інформаційної системи було використано дата сет “Used Cars Dataset”, що містить широкий спектр інформації по продажу автомобілів, яку надає компанія Craigslist, США. Дані викладені у вільний доступ на платформі Kaggle, користувачем Austin Reese в Травні 2021 року (рис. 2.15) [53].

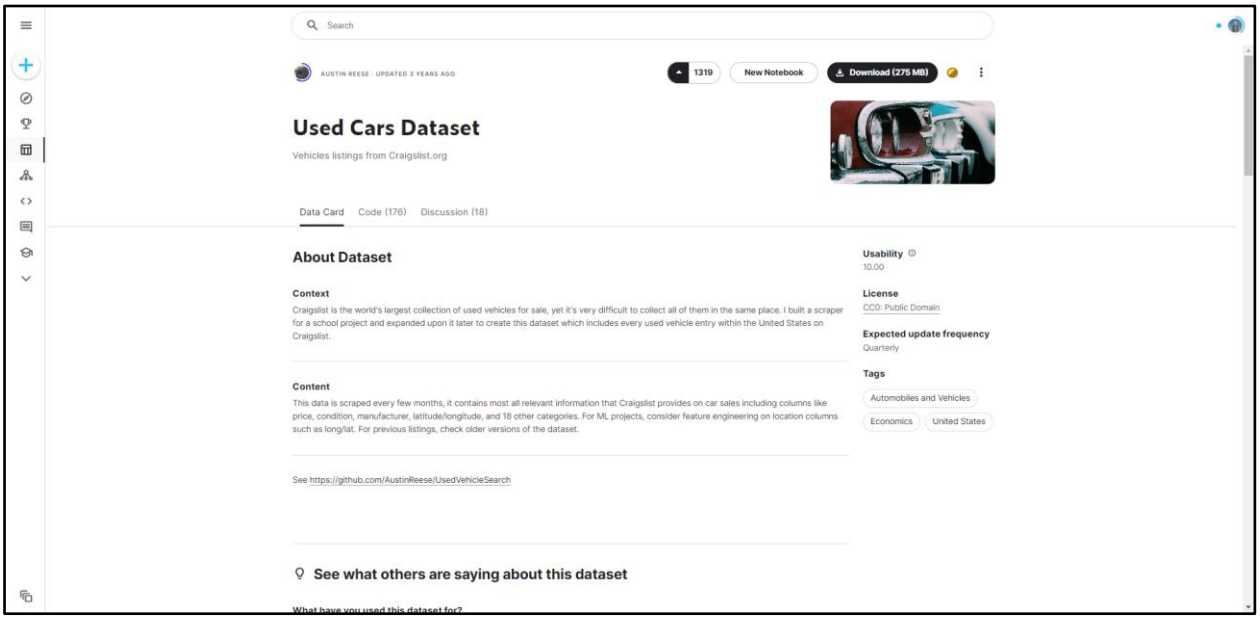

Рисунок 2.15 – Сторінка датасету на платформі Kaggle

Датасет досить популярний серед інших дослідників – більше 1300 вподобань та більше 170 відкритий (публічних) рішень.

Датасет доступний через створення записника (Notebook) через сторінку розміщених даних, через вбудований пошук наборів в записнику та через завантаження.

Датасет містить такі стовпці (ознаки):

- Унікальний ідентифікатор автомобіля в наборі («id»);
- Посилання на веб-сторінку оголошення про продаж автомобіля («url»);
- region – Регіон, в якому продається автомобіль;
- region_url – Посилання на веб-сторінку регіону, в якому продається автомобіль;
- price – Ціна автомобіля;
- year – Рік випуску автомобіля;
- manufacturer – Виробник автомобіля;
- model – Модель автомобіля;
- condition – Стан автомобіля;
- cylinders – Кількість циліндрів двигуна автомобіля;

- fuel – Тип палива, на якому працює автомобіль;
- odometer – Показник кілометражу автомобіля;
- title_status – Стан реєстрації автомобіля (документів);
- transmission – Тип коробки передач автомобіля;
- VIN – Ідентифікаційний номер транспортного засобу;
- drive – Тип приводу автомобіля;
- size – Розмір автомобіля;
- type – Тип кузова автомобіля;
- paint_color – Колір автомобіля;
- image_url – Посилання на зображення автомобіля;
- description – Опис автомобіля;
- county – Округ, в якому продається автомобіль (Помилково внесено в датасет);
- state – Штат, в якому продається автомобіль;
- lat – Широта місцезнаходження автомобіля;
- long – Довгота місцезнаходження автомобіля;
- posting_date – Дата публікації оголошення про продаж автомобіля.

Додатково:

- Ознаки long (довгота) та lat (широта) можуть бути корисними для візуалізації географічного розподілу автомобілів, що продавались на Craigslist. Це може бути корисно для завдань, таких як виявлення регіонів з високим попитом на автомобілі певного типу.
- Колонка posting_date (дата публікації) може бути використана для дослідження динаміки попиту на автомобілі протягом часу. Це може бути корисним для завдань прогнозування цін на автомобілі.

2.3 Зчитування та загальна інформація набору даних

Першим етопам в розв'язані задачі машинного навчання в Python є імпортування необхідних бібліотек та модулів, які, в свою чергу, будуть використовуватись на різних етапах розв'язання задачі, починаючи від зчитування даних, закінчуючи побудовою моделей.

Приклад імпортування бібліотек та модулів для нашій задачі наведено на рисунку 2.16.

```
import os
import glob
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
# import statsmodels.api as sm

# Importing seaborn library.
import seaborn as sns

# from mpl_toolkits.mplot3d import Axes3D # <--- This is important for 3d plotting
import matplotlib.pyplot as plt
import matplotlib.style as style
import matplotlib.gridspec as gridspec

# Preprocessing
from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.model_selection import train_test_split, GridSearchCV
from scipy import stats

# Pandas Profiling
from pandas_profiling import ProfileReport

# Models
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor, ExtraTreesRegressor
from sklearn.ensemble import BaggingRegressor, AdaBoostRegressor, VotingRegressor
# from sklearn.linear_model import LogisticRegression
from sklearn.metrics import mean_squared_error as MSE
from sklearn.metrics import mean_absolute_error as MAE

from sklearn.linear_model import LinearRegression, SGDRegressor, RidgeCV
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error, accuracy_score
from sklearn.svm import SVR, LinearSVR
from sklearn.neural_network import MLPRegressor

import xgboost as xgb
import lightgbm as lgbm

from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

# map visualization
import folium
from folium import plugins

%matplotlib notebook
%matplotlib inline

# model tuning
from hyperopt import STATUS_OK, Trials, fmin, hp, tpe, space_eval

import warnings
warnings.filterwarnings("ignore")

valid_part = 0.3
# pd.set_option('display.max_columns', 100)
```

Рисунок 2.16 – Імпорт модулів та бібліотек для розв'язання задачі

Наступним етапом розв'язання задачі є зчитування набору даних. Зчитування виконуємо за допомогою бібліотеки pandas, використовуючи метод read_csv, для читання csv-файлів. Приклад зчитування зображено на рисунках 2.17, 2.18.

```
df_main = pd.read_csv('../input/craigslist-carstrucks-data/vehicles.csv')

# First 5 rows and last 5 rows of our data
df = df_main[:]
df
```

Рисунок 2.17 – Фрагмент коду зчитування та відображення набору даних

	id	url	region	region_url	price	year	manufacturer	model	condition	cylinders	...	size	type	paint_color	image_url	description	county	state	lat
0	7222695916	https://prescott.craigslist.org/cto/d/prescott...	prescott	https://prescott.craigslist.org	6000	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	az	NaN
1	7218891961	https://fayar.craigslist.org/ctd/d/bentonville...	fayetteville	https://fayar.craigslist.org	11900	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	ar	NaN
2	7221797935	https://keys.craigslist.org/cto/d/summerland-k...	florida keys	https://keys.craigslist.org	21000	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	fl	NaN
3	7222270760	https://worcester.craigslist.org/cto/d/west-br...	worcester / central MA	https://worcester.craigslist.org	1500	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	ma	NaN
4	7210384030	https://greensboro.craigslist.org/cto/d/trinit...	greensboro	https://greensboro.craigslist.org	4900	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	nc	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
426875	7301591192	https://wyoming.craigslist.org/ctd/d/atlanta-2...	wyoming	https://wyoming.craigslist.org	23590	2019.0	nissan	maxima s sedan 4d	good	6 cylinders	...	NaN	sedan	NaN	https://images.craigslist.org/00o0o_iiraFnHg8q...	Carvana is the safer way to buy a car During t...	NaN	wy	33.786500
426876	7301591187	https://wyoming.craigslist.org/ctd/d/atlanta-2...	wyoming	https://wyoming.craigslist.org	30590	2020.0	volvo	s60 t5 momentum sedan 4d	good	NaN	...	NaN	sedan	red	https://images.craigslist.org/00x0x_15sbgnxCIS...	Carvana is the safer way to buy a car During t...	NaN	wy	33.786500
426877	7301591147	https://wyoming.craigslist.org/ctd/d/atlanta-2...	wyoming	https://wyoming.craigslist.org	34990	2020.0	cadillac	xt4 sport suv 4d	good	NaN	...	NaN	hatchback	white	https://images.craigslist.org/00L0L_farM7bxnxR...	Carvana is the safer way to buy a car During t...	NaN	wy	33.779214
426878	7301591140	https://wyoming.craigslist.org/ctd/d/atlanta-2...	wyoming	https://wyoming.craigslist.org	28990	2018.0	lexus	es 350 sedan 4d	good	6 cylinders	...	NaN	sedan	silver	https://images.craigslist.org/00z0z_bKnlVGLkDT...	Carvana is the safer way to buy a car During t...	NaN	wy	33.786500
426879	7301591129	https://wyoming.craigslist.org/ctd/d/atlanta-2...	wyoming	https://wyoming.craigslist.org	30590	2019.0	bmw	4 series 430i gran coupe	good	NaN	...	NaN	coupe	NaN	https://images.craigslist.org/00Y0Y_JEUocjyRxa...	Carvana is the safer way to buy a car During t...	NaN	wy	33.779214

426880 rows × 26 columns

Рисунок 2.18 – Перші та останні 5 рядків зчитаних даних

З рисунку 2.18 можна побачити, що набір даних начислює 426880 рядків(елементів) та 26 стовпців(ознак). Є досить багато пустих значень показаних лише на початкових та кінцевих п’яти елементів, що стосується інших елементів, будемо розглядати пізніше.

Наступним кроком, виводимо загальну інформацію по кожному стовпчику(ознаці) нашого набору функцією info(). Загальну інформацію набору даних наведено на рисунку 2.19.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 426880 entries, 0 to 426879
Data columns (total 26 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   id            426880 non-null  int64
 1   url           426880 non-null  object
 2   region        426880 non-null  object
 3   region_url    426880 non-null  object
 4   price         426880 non-null  int64
 5   year          425675 non-null  float64
 6   manufacturer  409234 non-null  object
 7   model         421603 non-null  object
 8   condition     252776 non-null  object
 9   cylinders     249202 non-null  object
 10  fuel          423867 non-null  object
 11  odometer      422480 non-null  float64
 12  title_status  418638 non-null  object
 13  transmission  424324 non-null  object
 14  VIN           265838 non-null  object
 15  drive         296313 non-null  object
 16  size          120519 non-null  object
 17  type          334022 non-null  object
 18  paint_color   296677 non-null  object
 19  image_url     426812 non-null  object
 20  description   426810 non-null  object
 21  county        0 non-null       float64
 22  state         426880 non-null  object
 23  lat           420331 non-null  float64
 24  long          420331 non-null  float64
 25  posting_date  426812 non-null  object
dtypes: float64(5), int64(2), object(19)
memory usage: 84.7+ MB
```

Рисунок 2.19 – Загальна інформація по ознакам із набору даних

2.4 Фільтрація аномальних даних та викидів

Для простого виявлення аномальних даних використовується функція describe(). Ця функція виводить основні статистичні параметри. Особливість цього методу полягає в можливості оглядати дані, взявши певний відсоток від них. Ці відсотки також відомі як квантилі. Наприклад, якщо відсоток становить чверть, тобто 25% або 75%, його називають квартилем.

Виведення елементарної статистики із автоматично налаштованими квантилями наведено на рисунку 2.20.

```
df.describe()
```

	id	price	year	odometer	county	lat	long
count	4.268800e+05	4.268800e+05	425675.000000	4.224800e+05	0.0	420331.000000	420331.000000
mean	7.311487e+09	7.519903e+04	2011.235191	9.804333e+04	NaN	38.493940	-94.748599
std	4.473170e+06	1.218228e+07	9.452120	2.138815e+05	NaN	5.841533	18.365462
min	7.207408e+09	0.000000e+00	1900.000000	0.000000e+00	NaN	-84.122245	-159.827728
25%	7.308143e+09	5.900000e+03	2008.000000	3.770400e+04	NaN	34.601900	-111.939847
50%	7.312621e+09	1.395000e+04	2013.000000	8.554800e+04	NaN	39.150100	-88.432600
75%	7.315254e+09	2.648575e+04	2017.000000	1.335425e+05	NaN	42.398900	-80.832039
max	7.317101e+09	3.736929e+09	2022.000000	1.000000e+07	NaN	82.390818	173.885502

Рисунок 2.20 – Фрагмент елементарних статистичних одиниць набору даних

З рисунку 2.20 видно, що в даних спостерігається значний перепад між мінімальними та максимальними цінами автомобілів, а також між значеннями пробігу (odometer). Цей різницевий характер може вказувати на наявність аномальних "шматків" даних, які впливають на результати навчання моделі та точність передбачення цін.

В процесі аналізу даних часто виявляються випадки присутності нульових чи аномальних значень. Такі значення можуть виникати з різних причин, таких як відсутність інформації, помилки в даних або навмисне приховування інформації. Фільтрація цих нульових і аномальних даних є важливою частиною процесу підготовки даних до аналізу. Врахування таких значень може спотворити результати аналізу та призвести до неточностей у висновках (рис. 2.21).

```
df.isnull().sum()
```

```
id                  0
url                 0
region              0
region_url          0
price               0
year             1205
manufacturer    17646
model            5277
condition      174104
cylinders      177678
fuel             3013
odometer         4400
title_status     8242
transmission     2556
VIN            161042
drive          130567
size           306361
type            92858
paint_color    130203
image_url          68
description        70
county         426880
state               0
lat              6549
long             6549
posting_date       68
dtype: int64
```

Рисунок 2.21 – Ряд стовпців із відповідною кількість нульових значень

На рисунку 2.21 зображено, кількість пустих значень в нашому наборі даних. Але важливе одне маленьке уточнення, що, хоч ознака price, не має відсутніх значень, ще не означає, що все там чудово і нічого не треба робити.

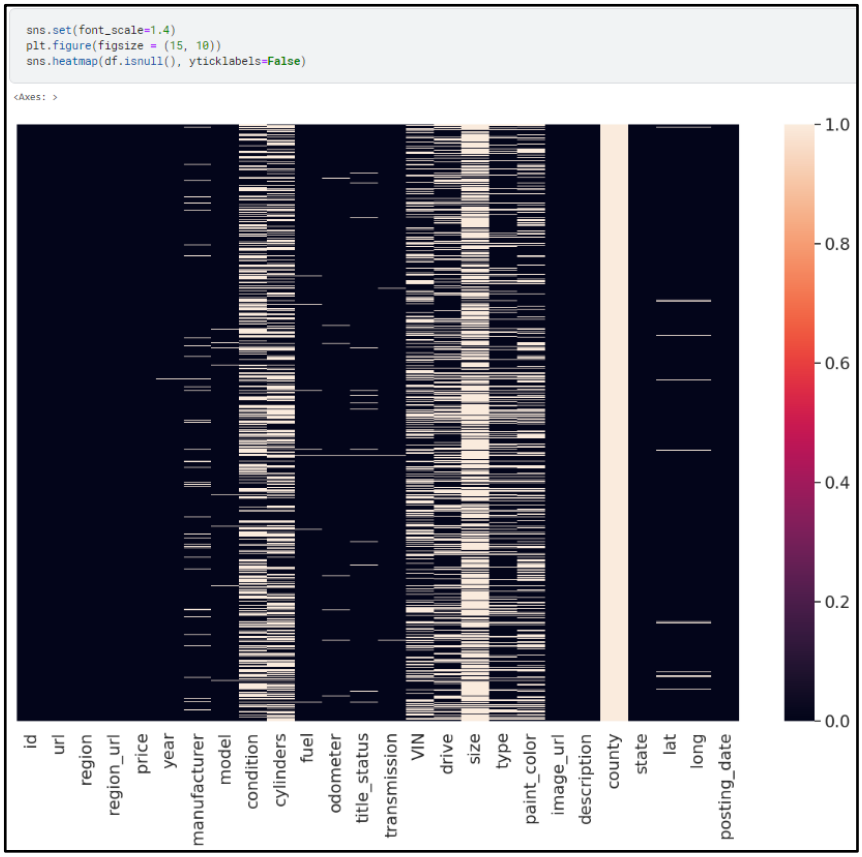

Рисунок 2.22 – Теплова карта відсутніх даних по ознакам

На рисунку 2.22, можна побачити, як розподілені відсутні значення по усім автомобілям. Із більше ніж чотирьохсот тисяч елементів, досить багато із них мають пусті значення по декільком ознакам (Білий колір – відсутнє значення).

Для чистоти роботи із набором даних, було вирішено видалимо ознаки, що мають найменший вплив, або ж не несуть ніякої цінності на ціноутворення та передбачення ціни автомобіля (рис. 2.23). Поправка на те, що місце знаходження автомобіля, може мати як сильний так і слабкий вплив на ціноутворення. В даній роботі, ми не будемо використовувати ці ознаки, в міру того, що ми не будемо вивчати та будувати навчання моделей навколо або разом із ціноутворення автомобілів за розташуванням, районом чи штатом.

```
drop_columns = ['id', 'url', 'region', 'region_url', 'title_status', 'VIN', 'paint_color', 'image_url', 'description', 'county', 'state', 'lat', 'long', 'posting_date']

df = df.drop(columns = drop_columns)
df
```

	price	year	manufacturer	model	condition	cylinders	fuel	odometer	transmission	drive	size	type
0	6000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	11900	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	21000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	1500	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	4900	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...
426875	23590	2019.0	nissan	maxima s sedan 4d	good	6 cylinders	gas	32226.0	other	fwd	NaN	sedan
426876	30590	2020.0	volvo	s60 t5 momentum sedan 4d	good	NaN	gas	12029.0	other	fwd	NaN	sedan
426877	34990	2020.0	cadillac	xt4 sport suv 4d	good	NaN	diesel	4174.0	other	NaN	NaN	hatchback
426878	28990	2018.0	lexus	es 350 sedan 4d	good	6 cylinders	gas	30112.0	other	fwd	NaN	sedan
426879	30590	2019.0	bmw	4 series 430i gran coupe	good	NaN	gas	22716.0	other	rwd	NaN	coupe

426880 rows × 12 columns

Рисунок 2.23 – Вилучення ряду ознак із датасету

Як виглядає набір даних тепер, після видалення частини ряду ознак, зображено на рисунку 2.24.

```
sns.set(font_scale=1.4)
plt.figure(figsize = (15, 10))
sns.heatmap(df.isnull(), yticklabels=False)
```

```
<Axes: >
```

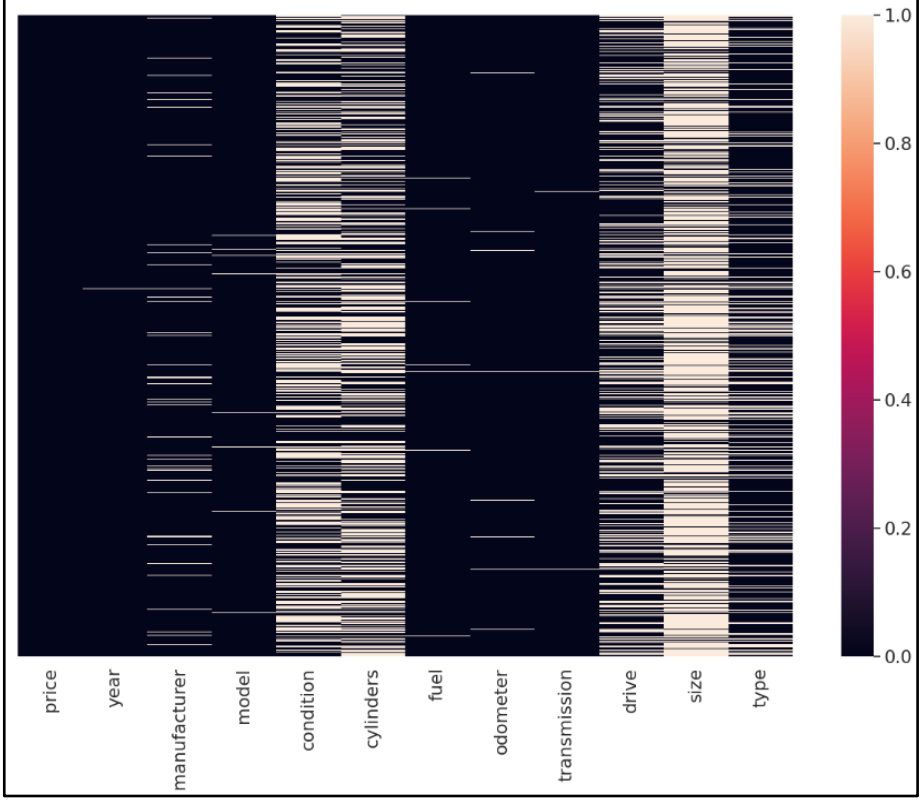

Рисунок 2.24 – Теплова карта відсутніх даних по ознакам

Повторний опис даних, за допомогою метода describe, показує наступний результат (рис. 2.25).

```
df.describe()
```

	price	year	odometer
count	4.268800e+05	425675.000000	4.224800e+05
mean	7.519903e+04	2011.235191	9.804333e+04
std	1.218228e+07	9.452120	2.138815e+05
min	0.000000e+00	1900.000000	0.000000e+00
25%	5.900000e+03	2008.000000	3.770400e+04
50%	1.395000e+04	2013.000000	8.554800e+04
75%	2.648575e+04	2017.000000	1.335425e+05
max	3.736929e+09	2022.000000	1.000000e+07

Рисунок 2.25 – Фрагмент елементарних статистичних одиниць набору даних

Що можна побачити, так це присутність лише трьох ознак в табличці. Означає це лише одне, усі інші ознаки, є категоріальними, тобто це текстові ознаки. В подальшому вони будуть перетворені на числові, для правильної роботи моделей машинного навчання.

Наступним кроком буде видалення тих рядків (автомобілів) із набору даних, що містять хоч одне відсутнє значення серед року випуску автомобіля (year) або виробника автомобіля (manufacturer) або моделі автомобіля (model) (рис. 2.26).

```
df = df.dropna(subset=['year', 'manufacturer', 'model'])
df
```

	price	year	manufacturer	model	condition	cylinders	fuel	odometer	transmission	drive	size	type
27	33590	2014.0	gmc	sierra 1500 crew cab slt	good	8 cylinders	gas	57923.0	other	NaN	NaN	pickup
28	22590	2010.0	chevrolet	silverado 1500	good	8 cylinders	gas	71229.0	other	NaN	NaN	pickup
29	39590	2020.0	chevrolet	silverado 1500 crew	good	8 cylinders	gas	19160.0	other	NaN	NaN	pickup
30	30990	2017.0	toyota	tundra double cab sr	good	8 cylinders	gas	41124.0	other	NaN	NaN	pickup
31	15000	2013.0	ford	f-150 xlt	excellent	6 cylinders	gas	128000.0	automatic	rwd	full-size	truck
...	...	...	...	...	...	...	...	...	...	...	...	...
426875	23590	2019.0	nissan	maxima s sedan 4d	good	6 cylinders	gas	32226.0	other	fwd	NaN	sedan
426876	30590	2020.0	volvo	s60 t5 momentum sedan 4d	good	NaN	gas	12029.0	other	fwd	NaN	sedan
426877	34990	2020.0	cadillac	xt4 sport suv 4d	good	NaN	diesel	4174.0	other	NaN	NaN	hatchback
426878	28990	2018.0	lexus	es 350 sedan 4d	good	6 cylinders	gas	30112.0	other	fwd	NaN	sedan
426879	30590	2019.0	bmw	4 series 430i gran coupe	good	NaN	gas	22716.0	other	rwd	NaN	coupe

404020 rows × 12 columns

Рисунок 2.26 – Оновлений набір даних

На рисунку 2.26 зображено код, що використовує метод dropna() для видалення рядків, де хоча б одне з значень у стовпцях 'year' або 'manufacturer', або 'model' є відсутнім. Метод subset=['year', 'manufacturer', 'model'] вказує, що ми хочемо перевіряти відсутність значень тільки у зазначених стовпцях.

Перед видаленням пустих значень, датафрейм начисляв 426880 рядків, після очищення пустих значень по трьом ознакам, набір став начислювати 404020 рядків. Тобто, було вилучено 22860 рядків, досить багато, але ці три ознаки, не можуть бути пустими, причиною цьому є наступне:

– Неможливість правильної ідентифікації автомобіля: Відсутність інформації про виробника або моделі може ускладнити ідентифікацію конкретного автомобіля. Це може бути проблемою, особливо, якщо покупець має конкретні вимоги до моделі або виробника.

– Неможливість визначення року випуску: Рік випуску автомобіля може суттєво вплинути на його вартість, стан та інші характеристики. Відсутність цієї інформації може призвести до неможливості оцінити історію автомобіля та здійснити об'єктивний вибір.

– Втрата можливості порівняння: Відсутність інформації про виробника чи модель може ускладнити порівняння між різними автомобілями та їхніми характеристиками. Покупець може втратити можливість визначити, які конкретні особливості або виробники відповідають його потребам.

Всі три пункти пояснюють важливість очищення пустих значень по ознакам year, manufacturer та model.

Результат про успішне очищення пустих значень по вище перекисленим ознакам наведено на рисунку 2.27.

```
df.isnull().sum()
```

```
price              0
year               0
manufacturer       0
model              0
condition     164342
cylinders     166951
fuel            2553
odometer        4137
transmission    2271
drive         120920
size          290544
type           84072
dtype: int64
```

Рисунок 2.27 – Ряд стовпців із відповідною кількість нульових значень

Наступним, важливим, параметром на вилучення пустих значень йде ознака – стан автомобіля (condition). Можна було б подумати, що стан автомобіля, особливо на ціну останнього ніяк не впливає. Але візьмемо до уваги рисунки 2.28 та 2.29, на яких зображено автомобілі за однаковими ознаками, такими як, виробник (manufacturer), модель (model) та рік (year).

```
df.loc[(df["manufacturer"] == "ford") & (df["model"] == "f-150") & (df["year"] == 2000.0)]
```

	price	year	manufacturer	model	condition	cylinders	fuel	odometer	transmission	drive	size	type
9435	5795	2000.0	ford	f-150	excellent	8 cylinders	gas	150000.0	automatic	rwd	full-size	truck
9560	5795	2000.0	ford	f-150	excellent	8 cylinders	gas	150000.0	automatic	rwd	full-size	truck
14330	2500	2000.0	ford	f-150	good	8 cylinders	gas	155000.0	automatic	4wd	full-size	truck
19723	3300	2000.0	ford	f-150	NaN	NaN	gas	170000.0	automatic	4wd	NaN	truck
23981	7995	2000.0	ford	f-150	NaN	6 cylinders	gas	83792.0	automatic	rwd	NaN	NaN
30683	5000	2000.0	ford	f-150	excellent	8 cylinders	gas	180000.0	automatic	rwd	full-size	truck
31967	2600	2000.0	ford	f-150	NaN	NaN	gas	200.0	automatic	NaN	NaN	NaN
41966	5600	2000.0	ford	f-150	excellent	6 cylinders	gas	234000.0	manual	rwd	full-size	pickup
53633	4500	2000.0	ford	f-150	NaN	NaN	gas	114000.0	automatic	NaN	NaN	NaN
82176	6700	2000.0	ford	f-150	good	8 cylinders	gas	124500.0	automatic	4wd	NaN	truck
91591	3500	2000.0	ford	f-150	fair	8 cylinders	gas	210100.0	automatic	4wd	NaN	truck
92626	0	2000.0	ford	f-150	good	NaN	gas	187000.0	automatic	NaN	NaN	truck
97981	3000	2000.0	ford	f-150	salvage	8 cylinders	gas	182000.0	automatic	4wd	full-size	pickup
99925	3000	2000.0	ford	f-150	salvage	8 cylinders	gas	182000.0	automatic	4wd	full-size	pickup
108102	450	2000.0	ford	f-150	NaN	NaN	gas	10000000.0	automatic	NaN	NaN	NaN
108797	450	2000.0	ford	f-150	NaN	NaN	gas	10000000.0	automatic	NaN	NaN	NaN
109793	450	2000.0	ford	f-150	NaN	NaN	gas	10000000.0	automatic	NaN	NaN	NaN
109840	450	2000.0	ford	f-150	NaN	NaN	gas	10000000.0	automatic	NaN	NaN	NaN
110253	450	2000.0	ford	f-150	NaN	NaN	gas	10000000.0	automatic	NaN	NaN	NaN
110623	450	2000.0	ford	f-150	NaN	NaN	gas	10000000.0	automatic	NaN	NaN	NaN
110715	450	2000.0	ford	f-150	NaN	NaN	gas	10000000.0	automatic	NaN	NaN	NaN
110964	450	2000.0	ford	f-150	NaN	NaN	gas	10000000.0	automatic	NaN	NaN	NaN
150512	2750	2000.0	ford	f-150	fair	8 cylinders	gas	213000.0	automatic	4wd	NaN	truck
157546	2900	2000.0	ford	f-150	fair	8 cylinders	gas	210715.0	automatic	4wd	full-size	pickup

Рисунок 2.28 – Фільтрація автомобілів за однаковим виробником, моделлю та роком

161863	1600	2000.0	ford	f-150	NaN	NaN	gas	203000.0	automatic	NaN	NaN	NaN
172689	2500	2000.0	ford	f-150	fair	8 cylinders	gas	240000.0	automatic	4wd	full-size	offroad
210481	5999	2000.0	ford	f-150	NaN	NaN	gas	220000.0	automatic	NaN	NaN	NaN
228462	1200	2000.0	ford	f-150	fair	8 cylinders	gas	231489.0	automatic	rwd	full-size	truck
237708	5300	2000.0	ford	f-150	good	8 cylinders	gas	245000.0	automatic	rwd	full-size	truck
239498	6200	2000.0	ford	f-150	excellent	8 cylinders	gas	195679.0	automatic	4wd	full-size	pickup
247075	5823	2000.0	ford	f-150	excellent	8 cylinders	gas	235793.0	automatic	4wd	full-size	truck
249844	3300	2000.0	ford	f-150	NaN	NaN	gas	183000.0	automatic	NaN	NaN	NaN
262044	5000	2000.0	ford	f-150	good	8 cylinders	gas	183579.0	automatic	4wd	NaN	truck
289964	2750	2000.0	ford	f-150	fair	4 cylinders	gas	139452.0	automatic	NaN	NaN	pickup
301657	3000	2000.0	ford	f-150	fair	NaN	gas	91000.0	automatic	4wd	full-size	pickup
303048	6900	2000.0	ford	f-150	NaN	NaN	other	93763.0	other	rwd	NaN	truck
303980	6300	2000.0	ford	f-150	excellent	8 cylinders	gas	232000.0	automatic	4wd	full-size	truck
304403	3500	2000.0	ford	f-150	good	8 cylinders	gas	212000.0	automatic	rwd	NaN	truck
307165	1850	2000.0	ford	f-150	fair	8 cylinders	gas	0.0	automatic	rwd	NaN	pickup
309589	5999	2000.0	ford	f-150	good	8 cylinders	gas	216300.0	automatic	4wd	full-size	pickup
336578	6900	2000.0	ford	f-150	NaN	NaN	other	93763.0	other	rwd	NaN	truck
341662	3500	2000.0	ford	f-150	NaN	NaN	gas	198645.0	manual	NaN	NaN	NaN
362105	5250	2000.0	ford	f-150	NaN	NaN	gas	174055.0	automatic	fwd	NaN	pickup
364677	4250	2000.0	ford	f-150	good	6 cylinders	gas	161000.0	automatic	NaN	NaN	NaN
369301	19950	2000.0	ford	f-150	excellent	8 cylinders	gas	151007.0	automatic	rwd	full-size	NaN
386428	5823	2000.0	ford	f-150	excellent	8 cylinders	gas	235793.0	automatic	4wd	full-size	truck
390420	900	2000.0	ford	f-150	NaN	NaN	gas	200000.0	other	NaN	NaN	NaN
390545	900	2000.0	ford	f-150	NaN	NaN	gas	200000.0	other	NaN	NaN	NaN
390904	900	2000.0	ford	f-150	NaN	NaN	gas	200000.0	other	NaN	NaN	NaN
391104	900	2000.0	ford	f-150	NaN	NaN	gas	200000.0	other	NaN	NaN	NaN
391272	900	2000.0	ford	f-150	NaN	NaN	gas	200000.0	other	NaN	NaN	NaN
391568	900	2000.0	ford	f-150	NaN	NaN	gas	200000.0	other	NaN	NaN	NaN

Рисунок 2.29 – Фільтрація автомобілів за однаковим виробником, моделлю та роком

Як видно з рисунків 2.28 та 2.29, виконавши фільтрація автомобілів за однаковим значенням виробника (“ford”), моделлю (“f-150”) та роком (2000.0), ми отримали автомобілі із різними показниками стану, але із однаковими іншими ознаками, наприклад циліндри, одометр, трансмісія, тощо. Якщо уважно перечитати та проаналізувати результат наведеної таблиці, можна побачити, що автомобілі із різним станом, але однаковими іншими ознаками

мають різну ціну, трохи більше, трохи менше, часом, навіть, ціна «на голову вище». Все через те, що ми маємо одразу декілька різних показників стану, а саме: «добре»(good), «відмінно»(excellent), «задовільно»(fair), «як новий»(like new), «новий»(new), NaN, «утилізований»(salvage). Назва кожного стану автомобіля досить добре пояснює, чому ціна може варіюватись від декількох сотень доларів, до декількох тисяч доларів за одну й ту саму марку автомобіля. Навряд чи хтось захотів би обрати серед двох однакових автомобілів той, стан якого «добре» за 2000 доларів, якщо є можливість взяти інший, такий самий, тієї ж самої моделі, пробігу, тощо, але стан якого «новий», але ціна якого буквально трохи більше за попередній. Це говорить про те, що стан, дуже важливо впливає на ціноутворення вживаного автомобіля.

Тож, було прийнято, логічне рішення, не заповнювати пусті значення ознаки стану, а просто видалити ті рядки, де стан автомобіля не вказаний. Не заповнювати пусті значення стану, були прийнято через причину недоречності, не обґрунтованості даної дії та можливості спотворення важливості цієї ознаки підставляннями значень «навмання». А чи вказаний стан правильно чи ні, не є задачею даної роботи. Результат очищення значень зображено на рисунку 2.30.

```
print("Dataset shape (rows, columns) before dropping NaN data by 'condition' subset:", df.shape)
no_na_condition = df.dropna(subset="condition")
print("Dataset shape (rows, columns) after dropping NaN data by 'condition' subset:", no_na_condition.shape)

print("Number of removed columns:", df.shape[0] - no_na_condition.shape[0])

df = df.dropna(subset="condition")
```

```
Dataset shape (rows, columns) before dropping NaN data by 'condition' subset: (404020, 12)
Dataset shape (rows, columns) after dropping NaN data by 'condition' subset: (239678, 12)
Number of removed columns: 164342
```

Рисунок 2.30 – Очищення даних за умовою «condition»

З рисунку 2.30 видно, що очистивши дані із пустим значенням «condition», було вилучено цілих 164342 рядки, досить великий «шмат» даних, але він же міг стати причиною поганого прогнозування моделей.

Наступним кроком буде ціни автомобілів. Так, ця ознака не має пустих значень, але є аномальні, не правильно або навмання вказані, тож варто їх проаналізувати та очистити.

Перш за все, варто видалити дані, що є аномально великими. Приклад таких даних, зображено на рисунку 2.31.

```
df.loc[(df['price'] >= 400_000)]
```

	price	year	manufacturer	model	condition	cylinders	fuel	odometer	transmission	drive	size	type
29386	1111111111	1999.0	ford	f350 super duty lariat	good	8 cylinders	diesel	149000.0	automatic	rwd	full-size	pickup
68935	2000000	2002.0	saturn	l-series l200 4dr sedan	good	4 cylinders	gas	164290.0	automatic	fwd	mid-size	sedan
79088	655000	2010.0	chrysler	town & country	good	6 cylinders	gas	106000.0	automatic	NaN	NaN	NaN
136516	17000000	2007.0	ram	2500	good	8 cylinders	diesel	170000.0	automatic	4wd	full-size	pickup
137807	123456789	1999.0	buick	regal	like new	6 cylinders	gas	96000.0	automatic	fwd	full-size	sedan
153082	1234567890	2006.0	volvo	vnl	fair	NaN	other	200000.0	manual	NaN	NaN	NaN
155421	1234567	2006.0	jeep	wrangler	like new	6 cylinders	gas	123456.0	automatic	4wd	mid-size	SUV
193736	123456789	2015.0	chevrolet	cruze	like new	NaN	gas	64181.0	automatic	fwd	compact	sedan
194292	1234567	2010.0	lincoln	mkt ecoboost	like new	NaN	gas	85653.0	automatic	NaN	full-size	SUV
219241	1111111	1970.0	dodge	challenger	fair	8 cylinders	gas	42000.0	automatic	rwd	full-size	coupe
230753	135008900	2008.0	nissan	titan se kingcab	like new	8 cylinders	gas	110500.0	automatic	4wd	full-size	truck
307488	123456789	1996.0	gmc	sierra 2500	fair	8 cylinders	gas	320000.0	automatic	4wd	full-size	pickup
318592	3736928711	2007.0	toyota	tundra	excellent	8 cylinders	gas	164000.0	automatic	4wd	full-size	pickup
356716	3736928711	1999.0	toyota	4runner	fair	6 cylinders	gas	211000.0	automatic	4wd	mid-size	NaN
379133	25003000	1991.0	chevrolet	camaro	fair	6 cylinders	gas	200000.0	automatic	NaN	NaN	NaN

Рисунок 2.31 – Приклад автомобілів за ціною (price) вище 400,000

Як бачимо з рисунку 2.31, хоч і не багато, але декілька автомобілів все ж мають дуже велике значення ціни, що є не припустимим, а також, видно ціни, що введені навмання, от як ціна на Ford f350 super duty – 1111111111.

Варто зазначити, що автомобілі бізнес класу, або спортивні, можуть певним чином, погано, вплинути на навчання моделі. Але щоб сильно не обмежувати наш датасет, видалимо автомобілі, ціни на які більше за 200000 (двісті тисяч). Окрім цього, як показано на рисунках 2.32 та 2.33.

```
zeros = df[df.price == 0].shape
print(f'Amount of zero values for price is equal to {zeros}')
```

```
Amount of zero values for price is equal to (9741, 12)
```

+ Code + Markdown

```
df.loc[(df['price'] == 0)]
```

	price	year	manufacturer	model	condition	cylinders	fuel	odometer	transmission	drive	size	type
46	0	2011.0	jeep	compass	excellent	NaN	gas	99615.0	automatic	NaN	full-size	SUV
126	0	2018.0	chevrolet	express cargo van	like new	6 cylinders	gas	68472.0	automatic	rwd	full-size	van
127	0	2019.0	chevrolet	express cargo van	like new	6 cylinders	gas	69125.0	automatic	rwd	full-size	van
128	0	2018.0	chevrolet	express cargo van	like new	6 cylinders	gas	66555.0	automatic	rwd	full-size	van
191	0	2015.0	nissan	sentra	excellent	4 cylinders	gas	99505.0	automatic	fwd	NaN	sedan
...	...	...	...	...	...	...	...	...	...	...	...	...
426763	0	2009.0	toyota	prius	excellent	4 cylinders	hybrid	271000.0	automatic	fwd	NaN	NaN
426812	0	2006.0	toyota	scion tc	excellent	4 cylinders	gas	195000.0	automatic	fwd	NaN	NaN
426832	0	2004.0	toyota	prius	excellent	4 cylinders	hybrid	239000.0	automatic	fwd	NaN	NaN
426836	0	2018.0	ram	2500	excellent	6 cylinders	diesel	20492.0	automatic	4wd	full-size	truck
426868	0	2010.0	toyota	venza	excellent	6 cylinders	gas	155000.0	automatic	4wd	NaN	NaN

9741 rows × 12 columns

Рисунок 2.32 – Фільтрація за ціною «0 (нуль)»

```
df.loc[(df['price'] == 1)]
```

	price	year	manufacturer	model	condition	cylinders	fuel	odometer	transmission	drive	size	type
1754	1	2005.0	gmc	envoy	excellent	8 cylinders	gas	181000.0	automatic	4wd	full-size	SUV
5017	1	2010.0	chevrolet	traverse lt awd	excellent	6 cylinders	gas	120000.0	automatic	4wd	mid-size	SUV
5971	1	2010.0	chevrolet	traverse lt awd	excellent	6 cylinders	gas	120000.0	automatic	4wd	mid-size	SUV
6368	1	1995.0	ford	f-350	excellent	NaN	gas	130000.0	manual	NaN	NaN	NaN
8418	1	1988.0	ford	f350 dually	good	NaN	diesel	130000.0	automatic	4wd	full-size	truck
...	...	...	...	...	...	...	...	...	...	...	...	...
422102	1	2020.0	mercedes-benz	porsche. bmw. ect.	new	other	other	1.0	other	rwd	mid-size	other
423876	1	2008.0	buick	lacrosse premium	good	6 cylinders	gas	9000.0	automatic	fwd	mid-size	sedan
425626	1	2012.0	chevrolet	express	excellent	8 cylinders	gas	123.0	automatic	4wd	full-size	van
425627	1	2012.0	chevrolet	express	excellent	8 cylinders	gas	1.0	automatic	NaN	full-size	van
425628	1	2012.0	chevrolet	express	excellent	NaN	gas	123.0	automatic	NaN	full-size	van

1085 rows × 12 columns

Рисунок 2.33 – Фільтрація за ціною «1 (один)»

На рисунках 2.32 та 2.33, показано явний приклад того, що досить багато даних ціни, було внесено не коректно. Але це далеко не все, є також ціни що дорівнюють 2, 111, 222 і так далі. Тож прийнято рішення, видалити автомобілі, ціни на які менше за показник 1500. Ціна не велика, але за таку ціну ще можна знайти більш менш нормальне вживане авто на певний, короткий період експлуатації.

Фінальним рішенням буде залишити автомобілі, ціни на які знаходять в діапазоні від 1500 до 200 тисяч доларів. Результат очистки зображено на рисунку. 2.34.

Рисунок 2.34 – Фільтрація за ціною в діапазоні від 1500 до 200,000

Як показано на рисунку 2.34, до очистки, набір начисляв 239678 рядків (автомобілів), а вже після очистки 223493 рядків. Таким чином, було видалено 16185 елементів. Багато, як для даних, із некоректно введеною та малою ціною на автомобілі.

Додатково було видалено 26 автомобілів, ціна на які дорівнювала таким значенням як: 12345, 123456, 111111. Що особливо не вплине на навчання моделі, але краще, виявлені нюанси виправити.

Наш набір, також, містить досить цікаву ознаку – рік випуску автомобіля. Цікавий він тим, що «найстарше» значення року автомобіля становить 1900, а «наймолодше» - 2022. Із автомобілями «нового покоління» все нормально, а от щодо старих автомобілів, варто зробити фільтрацію.

Найкращим рішенням буде видалити автомобілі, що старше за 2000 рік випуску, адже таким автомобілям вже більше 20 років, але щоб сильно не обмежувати дані, вирішено видалити автомобілі старше за 1990 рік випуску. Чому в додаткових, детальніших експериментах в даному напрямку варто брати менший рік випуску? Відповідь дуже проста, виробники автомобілів, зазвичай дають гарантію на автомобіль, в середньому 5 років. Звичайно, що якщо піклуватись про стан автомобіля, він може прослужити і всі 20, а того й більше, але тут теж варто зазначити один важливий нюанс. Автомобілі, старшого року випуску, в певних випадках, можна вважати раритетними. А всі чудово знають, що ціни на такі автомобілі, дуже часто завишаються. Такі автомобілі складно оцінювати по стану, по пробігу, моделі та виробнику. Тож очищення даних, що зображено на рисунках 2.35 та 2.36, враховує всі, вище перераховані критерії.

```
df = df[df["year"] >= 1990]
df["year"].describe()
```

```
count    217321.000000
mean       2012.210389
std           5.946063
min        1990.000000
25%        2008.000000
50%        2013.000000
75%        2017.000000
max        2022.000000
Name: year, dtype: float64
```

Рисунок 2.35 – Розподіл років вище 1990 року в датасеті

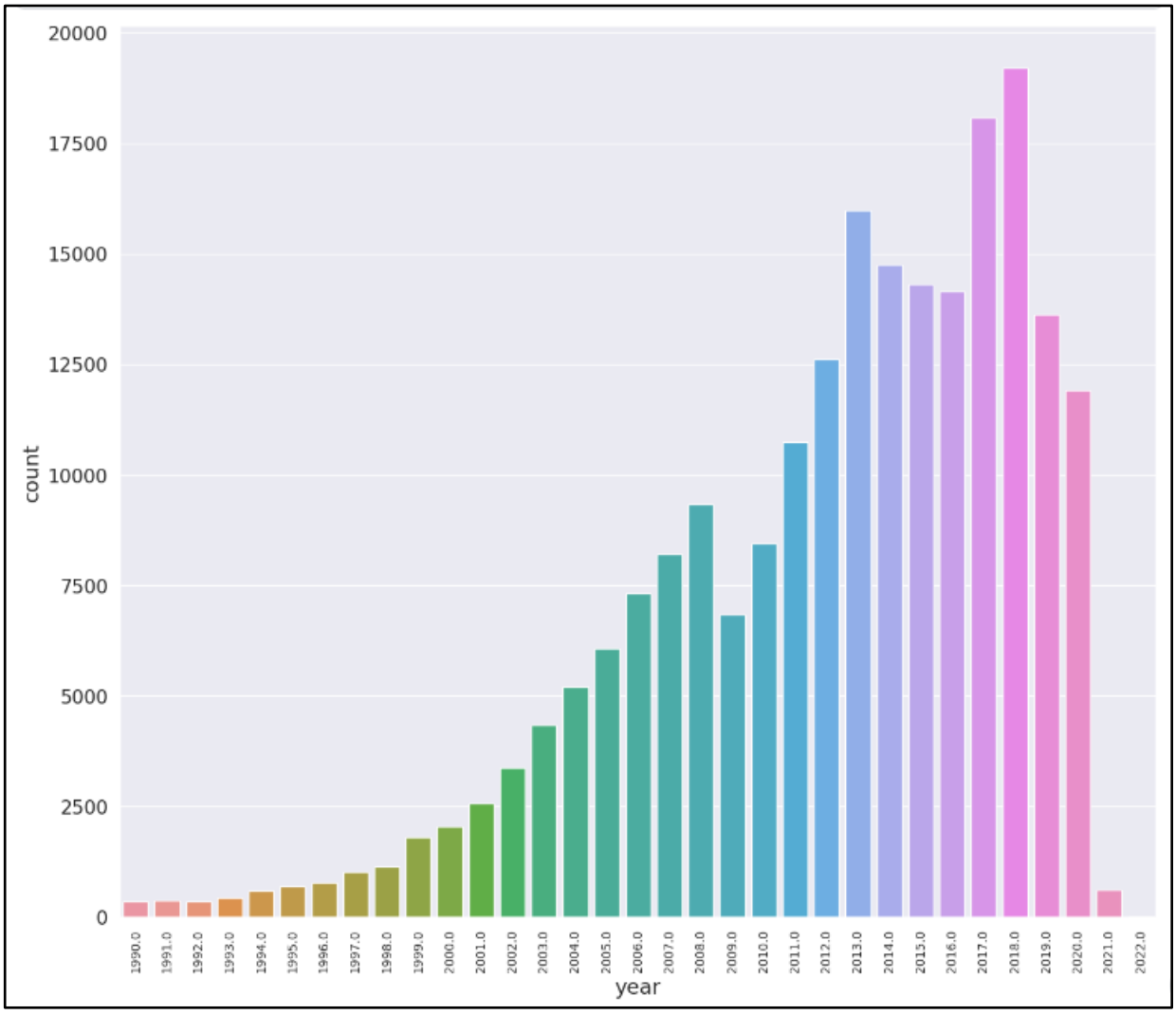

Рисунок 2.36 – Розподіл років автомобілів у датасеті за допомогою графіка countplot

Фільтрація одометра виконується досить просто, якщо пробіг автомобіля більше 200 тисяч, такий автомобіль видаляється. Причиною такої фільтрації, є складне рішення ціноутворення автомобіля із великим пробігом. Показник пробігу більше 150 тисяч вже є не бажаним для багатьох покупців, що говорити про пробіг більше 200 тисяч.

Результат очищення значень по ознаці пробігу (odometer) зображено на рисунках 2.37 та 2.38.

```
old_shape = df.shape
df = df[df["odometer"] <= 200_000]
print(old_shape[0] - df.shape[0])
```

```
13492
```

+ Code + Markdown

```
plt.figure(figsize = (8, 12))
sns.boxplot(y = 'odometer', data = df)
```

```
<Axes: ylabel='odometer'>
```

Рисунок 2.37 – Фільтрація по ознаці одометр та побудова boxplot

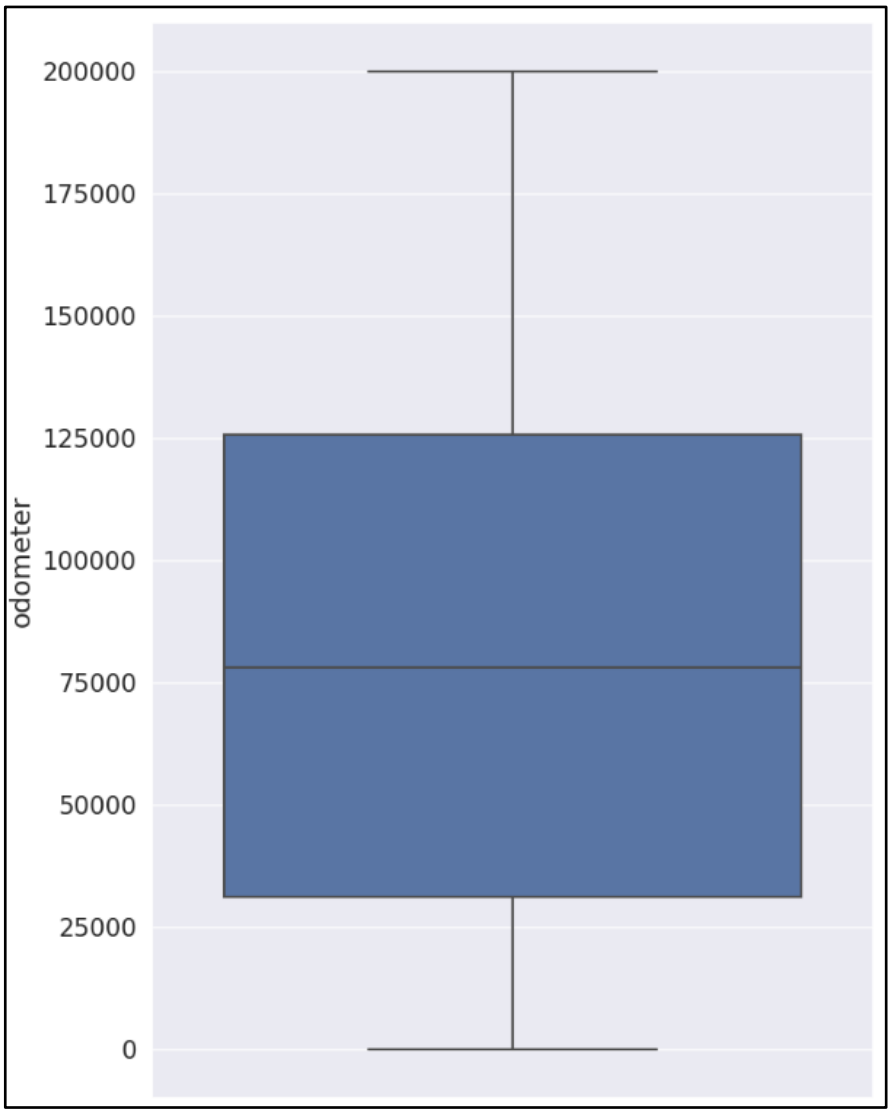

Рисунок 2.38 – Boxplot графік розподілу пробігу (odometer) автомобілів

З рисунку 2.38 видно, що більшість даних по ознаці odometer, знаходиться приблизно між 30 тисячами та 126 тисячами, але є і невеликі викиди. Середнє значення близько 77 тисяч.

Усі ж інші ознаки (рис 2.39), а саме, cylinders, drive, size, type, заповнимо значенням, таким як “unknown”, тобто невідоме.

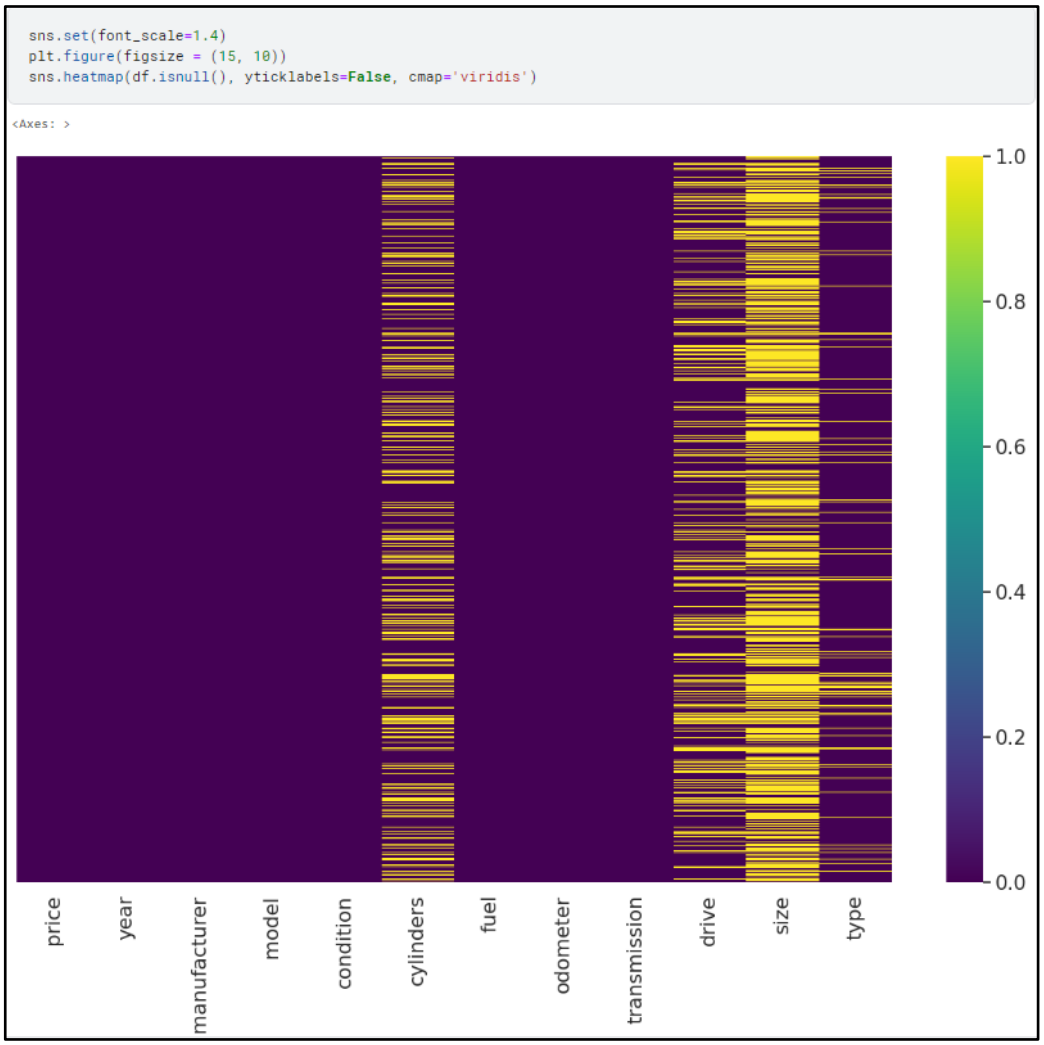

Рисунок 2.39 – Heatmap відсутніх значень у датасеті.

Код заповнення усіх пустих значень по останнім ознакам та результат заповнення., зображено на рисунку 2.40.

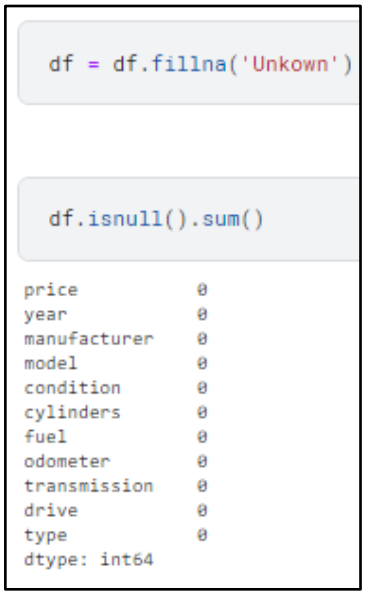

Рисунок 2.40 – Заповнення відсутніх значень та візуалізація після процедури заповнення

Ми не можемо визначити точно, скільки в автомобілі може бути циліндрів, який тип трансмісії (автомат, ручний, тощо), та тип приводу (передній, задній, повний), тощо. Тому і було прийнято рішення, про заповненням «невідомим» значенням кожного елемента по ознакам, що містять пусті значення.

Після застосування усіх заходів з очищення та фільтрації набору даних, вибуло видалено 426880 - 203829 = 223051 рядків даних. Більше половини даних було видалено. Побачимо що покаже нам розвідувальний аналіз даних, про наш набір.

2.5 Розвідувальний аналіз даних

EDA (Exploratory data analysis або Розвідувальний аналіз даних) – процес вивчення основних характеристик набору даних для отримання загального, повного уявлення про їх структуру, властивості та закономірності. EDA – забезпечує глибоке розуміння даних, перед їх використанням у моделях аналізу чи машинного навчання.

Побудуємо гістограми для відображення того, як розподілені значення цін на вживані автомобілі в нашому наборі даних. Для цього використовуємо гістограму цін, графік розподілу цін (рис 2.41) та графік «коробка з вусами» цін автомобілів (рис. 2.42).

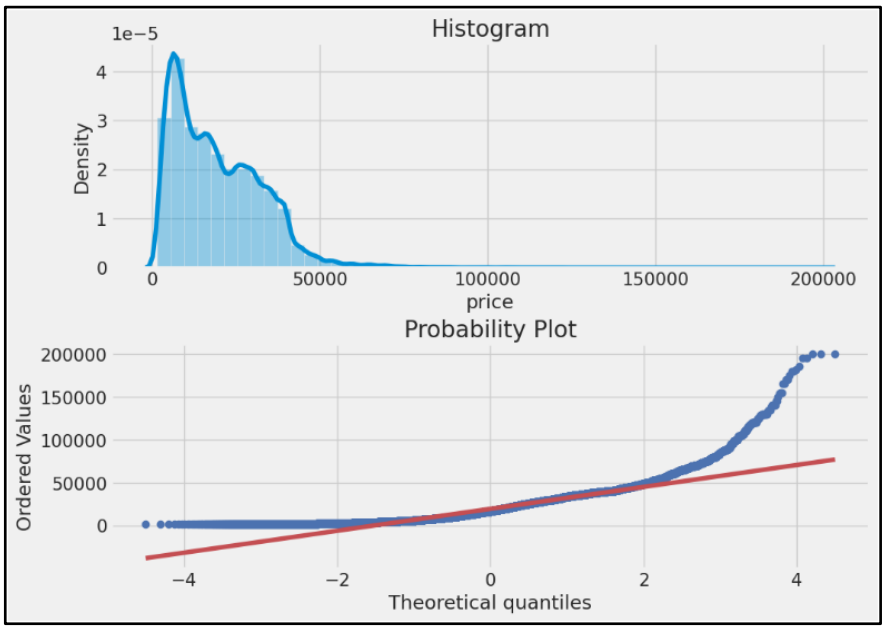

Рисунок 2.41 – Перевірка цінової ознаки на аномальні дані на гістограмі та графіку розподілу

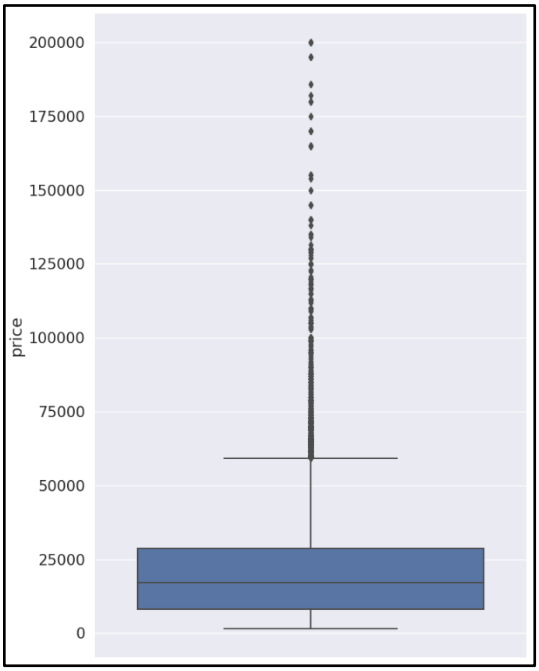

Рисунок 2.42 – Перевірка цінової ознаки на графіку «коробка з вусами»

На рисунку 2.41 зображено гістограму цін (перший графік), на якій видно, що більша частина автомобілів має ціну від, приблизно нуля, але насправді від 1500 доларів і до 50 тисяч доларів. Автомобілі з ціною вище на 50 тисяч доларів, трапляються досить рідко, підтвердженням цього є діаграма «коробка з вусами» (рис.2.42), з неї видно, що середня ціна автомобілів в нашому датасеті знаходиться між 10 тисячами доларів та 30 тисячами доларів.

З графіку ймовірності видно, що наша лінія проходить через синусоїду, приблизно 60-70% усіх даних знаходиться навколо даної лінії, але нехай графік не вводить нікого в оману, найбільша частина даних «скупчилась» саме між цінами від 2500 до 50 тисяч і знаходження даних в цій частині, становить від 80 до 90 відсотків, це означає, що ціни на автомобілі в цьому діапазоні будуть досить добре передбачуватись моделями, всі ж інші можуть мати певну похибку. Можна було б тоді видалити ці викиди, щоб моделі машинного навчання могли краще передбачувати ціни на вживані автомобілі, але ціною меншою за 70 тисяч. Насправді, можна було б, але давайте залишимо і ті ціни, що складають певний викид, щоб модель могла передбачувати ціни на автомобілі дорожче 70 тисяч, хоч і точність деяких передбачень буде трохи гірше.

На діаграмі «коробка з вусами» також видно, що на значені, приблизно, 60 тисяч закінчуються «вуса коробки», що свідчить про те, що усе що знаходиться вище – викид. Причина того, що ми залишаємо ці значення в тому, що наша модель не зможе передбачити ціну на автомобіль дорожче 100000 доларів, якщо вона буде навчатись лише на даних автомобілів, ціна яких менше 60 тисяч.

На рисунку 2.43 зображено побудову діаграми та саму тривимірну діаграму значення осей якої є: ціна, рік випуску та пробіг автомобіля.

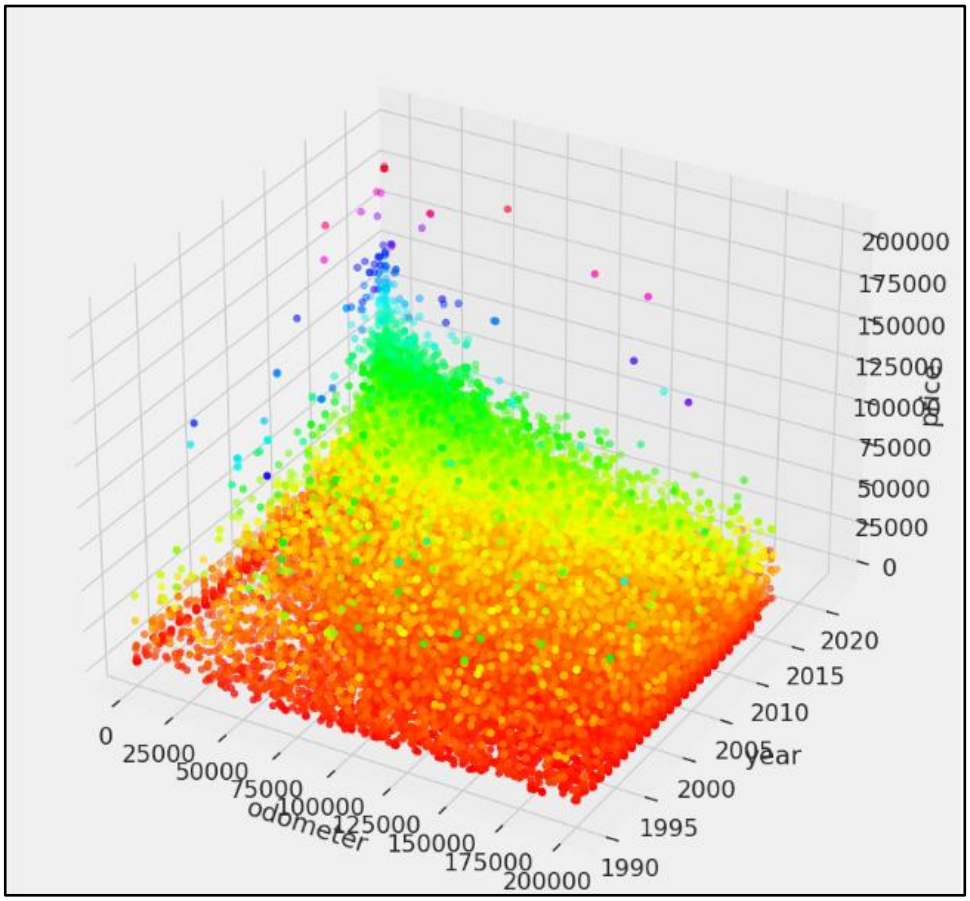

Рисунок 2.43 – Побудова тривимірної діаграми (ціна, рік, пробіг)

З рисунку 2.43 видно, що переважна кількість автомобілів знаходиться після 2005 року. Сильний зріст кількості автомобілів видно з 2008 по 2020 роки. Також видно, як з роками, ціна на новіші автомобілі, зростає, досить різко. Також, в нижньому лівому куті діаграми, де пробіг автомобілів, що є досить старими, знаходиться між значенням нуля та 50 тисяч зустрічається рідше. Але в той же час, автомобілі, що мають недавній рік випуску, та малий пробіг – більшість на цій діаграмі.

Графік розподілу кількості автомобільних пропозицій за виробниками зображено на рисунку 2.44

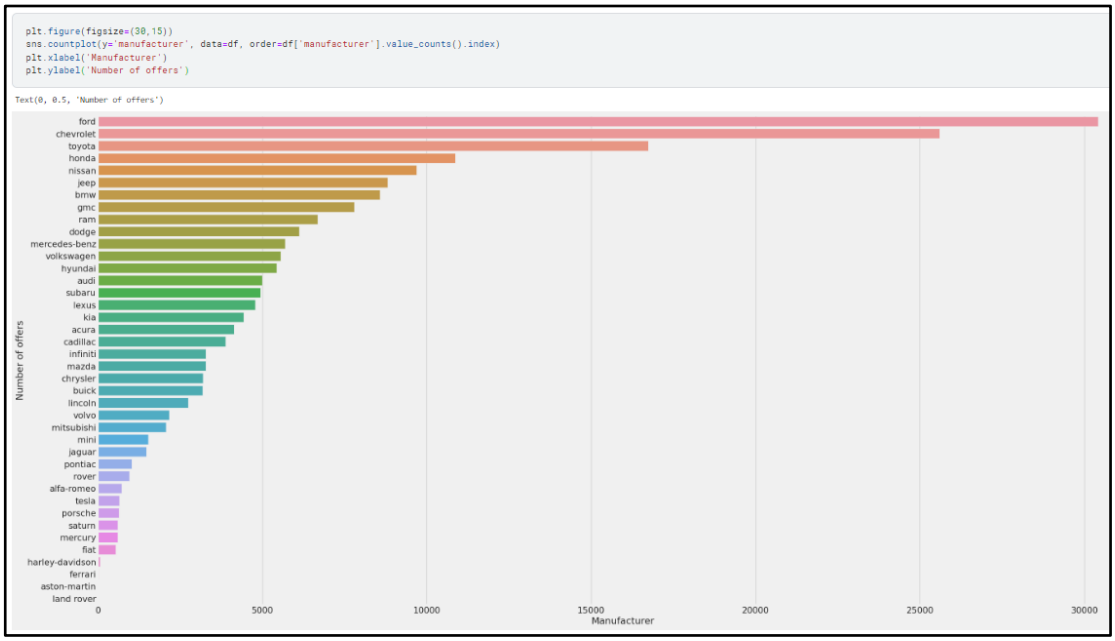

Рисунок 2.44 – Розподіл кількості автомобільних пропозицій за виробником

Як видно з рисунку 2.44, на графіку відображена кількість автомобілів для кожного виробника у порядку зменшення кількості пропозицій. Найчастіше зустрічаються Ford, Chevrolet, Toyota.

Взаємозв’язок трьох ознак (ціна, рік, пробіг) зображено на рисунку 2.45.

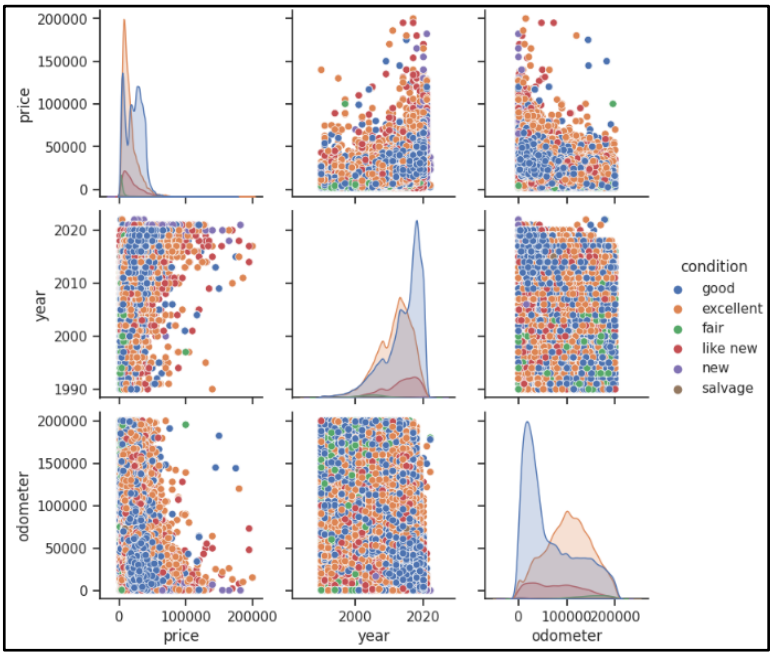

Рисунок 2.45 – Pair Plot взаємозв’язків ознак (ціна, рік, пробіг) у датасеті з розфарбуванням по стану автомобіля

Графік pairplot (рис. 2.45), візуалізує взаємозв'язки між усіма можливими парами ознак в нашому наборі, враховуючи кожен унікальний стан, яку ми позначили за допомогою кольору.

Кожна точка на діаграмі розсіювання представляє пару ознак, і її розташування вказує на залежність між цими ознаками. Діагональні графіки відображають розподіл кожної окремої ознаки.

Колір точок на графіку вказує на значення умови (значення у стовпці «condition»). Це дозволяє нам визначити, як взаємозв'язки між ознаками змінюються або виявляють закономірності в залежності від умови.

Створено стовпчасту діаграму, яка відображає вплив стану на середні ціни на автомобілі за різними типами палива (рис. 2.46).

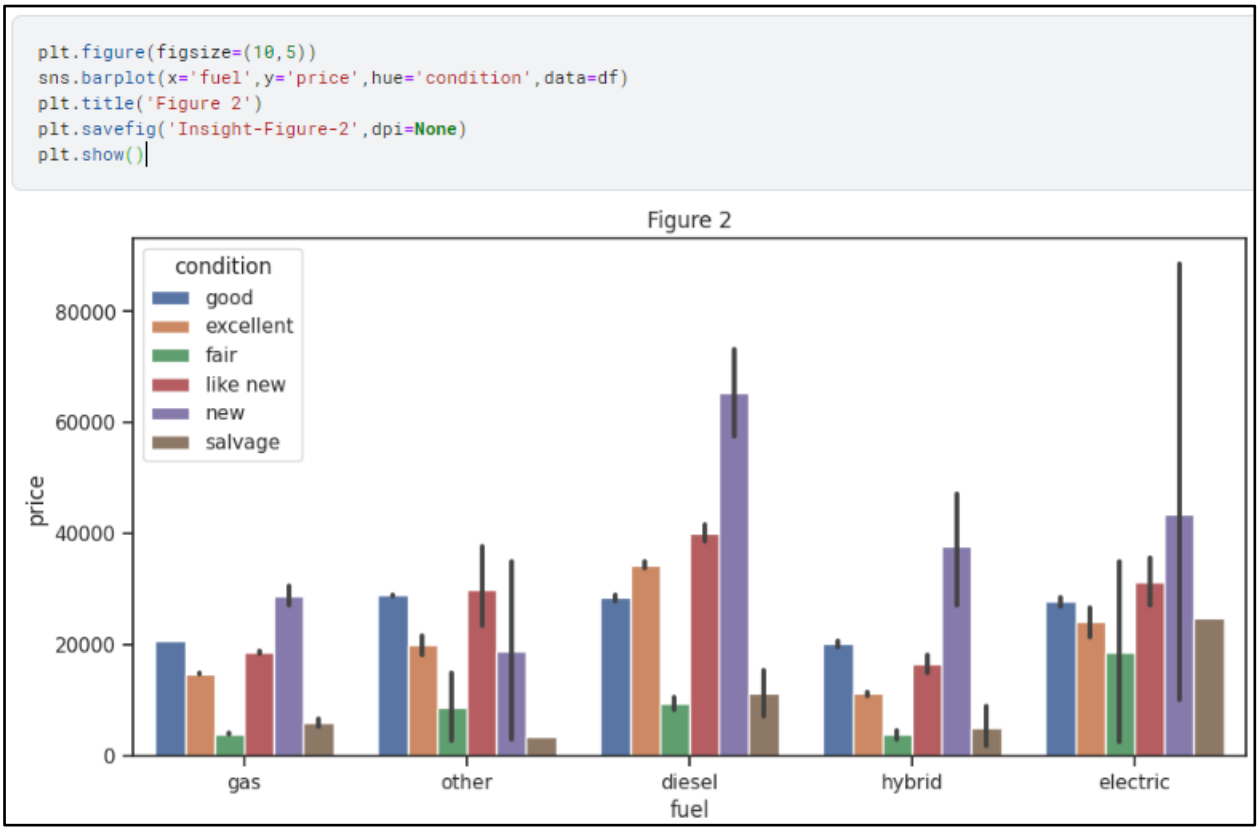

Рисунок 2.46 – Вплив стану на ціни автомобілів за типом палива

На рисунку 2.46 видно яскраво виражені в ціні автомобілі станом «новий» що мають тип палива «електро», «дизель» та «гібрид».

Створено 2 графіки: перший показує вплив року випуску на ціни автомобілів, а другий – кількість автомобілів за кожен рік випуску. Два графіки показують зростання як кількості автомобілів, так і ціни автомобілів (рис. 2.47).

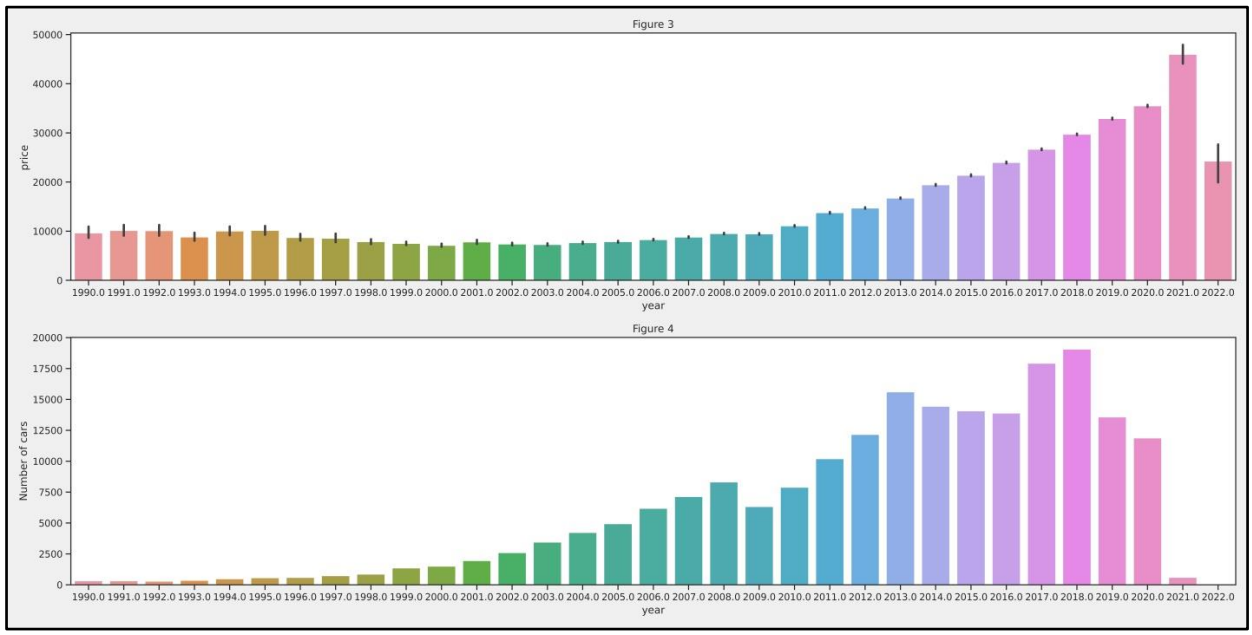

Рисунок 2.47 – Вплив року випуску на ціни та кількість автомобілів

Створено два графіки: перший відображає вплив стану автомобіля на ціни, а другий - вплив розміру автомобіля на ціни з урахуванням умови (рис. 2.48).

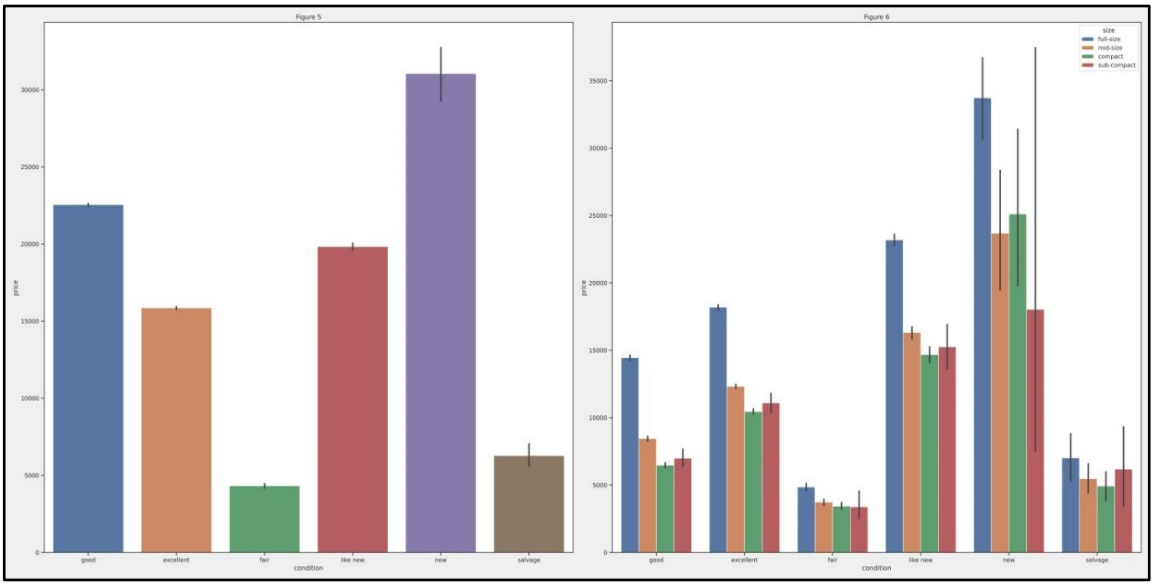

Рисунок 2.48 – Вплив стану автомобіля та його розміру на ціни

На рисунку 2.48 видно, що практично усі автомобілі станом «новий» та «як новий» мають найбільший показник ціни, серед автомобілів інших станів. Найменші ціни на автомобілі із станом «fair». Чудовий приклад, чому показник condition є важливим. Зліва направо йдуть такі значення стану: “good”, “excellent”, “fair”, “like new”, “new:, “salvage”. На графіку справа, значення ознаки “size”, теж йдуть зліва направо в такому порядку: “full-size”, “mid-size”, “compact”, “sub-compact”.

Графік впливу розміру автомобіля на ціну та розподіл за типом приводу зображено на рисунку 2.49.

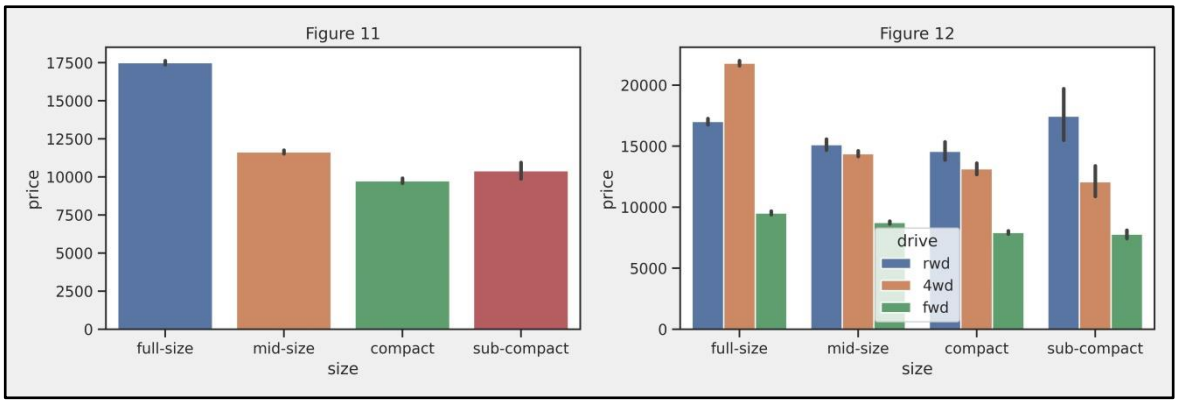

Рисунок 2.49 – Вплив розміру автомобіля на ціни та розподіл за типом приводу

Важко сказати, наскільки сильно показник приводу впливає на ціну автомобіля (рис. 2.49), але що точно можна зауважити, так це те, що на американському ринку, переважають автомобілі великих розмірів та повного або заднього приводу.

Розподіл кількості автомобілів за станом та типом трансмісії зображено на рисунку 2.50.

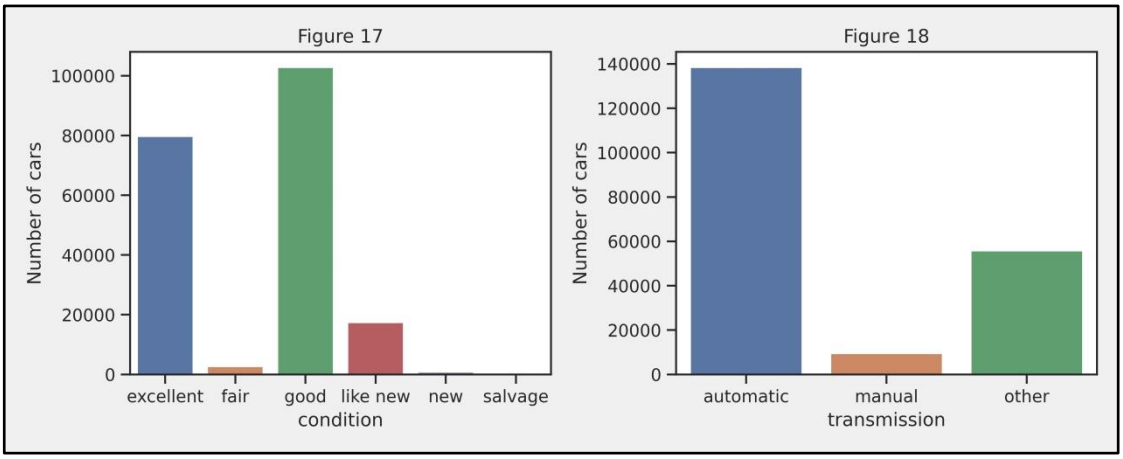

Рисунок 2.50 – Розподіл кількості автомобілів за станом та типом трансмісії

На рисунку 2.50 зображено два графіки, що показують розподіл кількості автомобілів за станом (рис. зліва) та типом трансмісії (рис. справа). З цього рисунка видно, що в нашому наборі, дуже сильно переважають два стану автомобілів: “excellent” та “good”. А також,сильно переважає тип трансмісії “automatic”. Можна зробити висновок, що більше половини набору, складають автомобілі чудового та відмінного стану із автоматичною коробкою передач. І тоді, можна зробити припущення, що ціни на такі автомобілі, будуть передбачуватись куди краще.

Побудовано два графіки, що показують розподіл кількості автомобілів за типом палива та розміром (рис. 2.51).

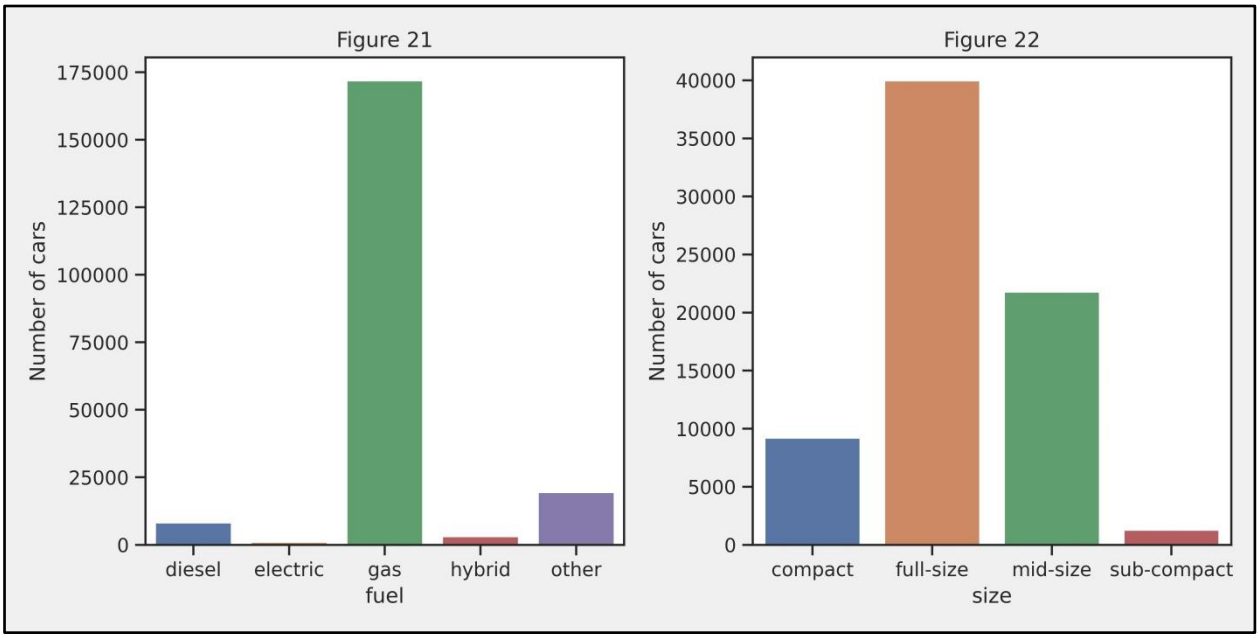

Рисунок 2.51 – Розподіл кількості автомобілів за типом палива та розміром

На рисунку 2.51 можна побачити, на лівому графіку, що більше половини усіх автомобілів, значення по ознаці «паливо» дорівнює бензин (“gas”). На правому графіку видно, що в наборі, здебільшого зустрічаються автомобілі “full-size“ та “mid-size”.

Маючи ознаки lat та long, можемо вивести місця розташування кожного автомобіля на карті. Побудова карти та результат візуалізації карти наведено на рисунках 2.52 та 2.53.

```
map_df = df.dropna(subset="lat")
car_map = folium.Map(location = [map_df['lat'].mean(), map_df['long'].mean()], zoom_start = 10)
lat_long_data = map_df[['lat', 'long']].values.tolist()
c_cluster = folium.plugins.FastMarkerCluster(lat_long_data,).add_to(car_map).add_to(car_map)
car_map
```

Рисунок 2.52 – Код програми для побудови карти розташування автомобілів

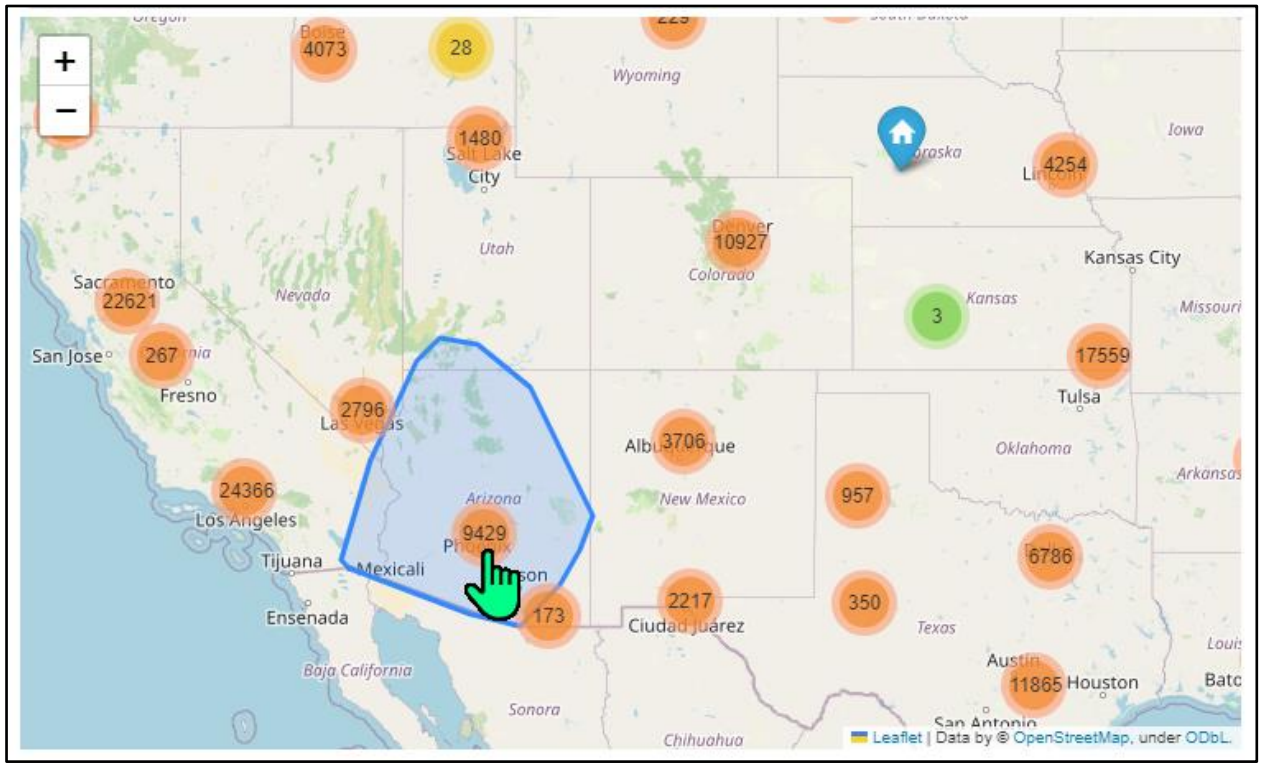

Рисунок 2.53 - Розташування автомобілів на мапі

На рисунку показано «кружечки» із числами. Кожен такий круг займає певну область із автомобілями. Якщо приближатись, то кожен раз, кружечків стає все більше і більше а число на них менше, відбувається це до тих пір, поки ви не дійдете до самого автомобіля та його положення (рис. 2.54).

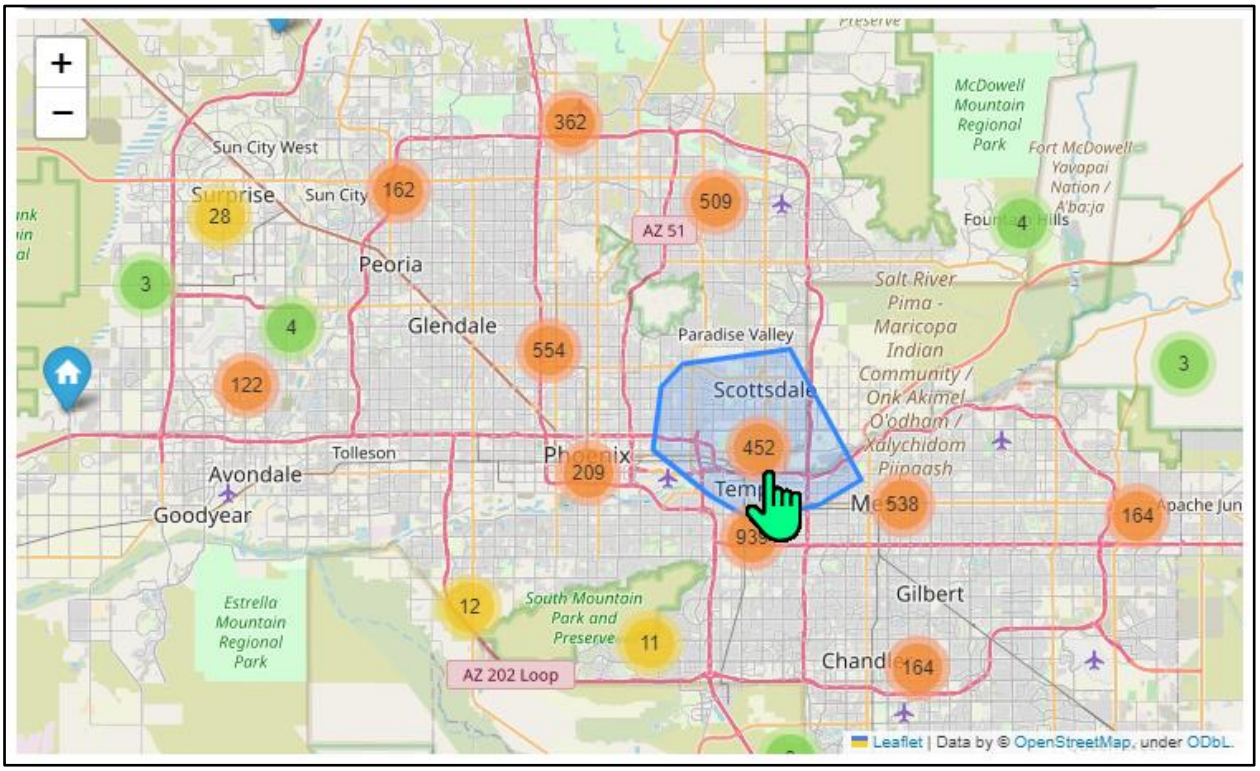

Рисунок 2.54 – Розташування автомобілів на мапі (штат Арізона)

Розташування автомобілів можна використовувати для виявлення залежності ціни автомобіля, від його розташування за штатами, містами, тощо. В даній роботі, це не досліджується.

Останній, важливий, крок перед побудовою моделей – це перетворення категоріальних ознак, про які говорилось раніше. Код виконання перетворення наведений на рисунку 2.55

```
# from the my kernel: https://www.kaggle.com/vbmokin/automatic-selection-from-20-classifier-models
# Determination categorical features
numerics = ['int8', 'int16', 'int32', 'int64', 'float16', 'float32', 'float64']
categorical_columns = []
features = df.columns.values.tolist()
for col in features:
    if df[col].dtype in numerics: continue
    categorical_columns.append(col)

# Encoding categorical features
for col in categorical_columns:
    if col in df.columns:
        le = LabelEncoder()
        le.fit(list(df[col].astype(str).values))
        df[col] = le.transform(list(df[col].astype(str).values))
```

Рисунок 2.55 – Кодування категоріальних ознак в числовий формат

На рисунку 2.55 зображено код, що призначений для перетворення категоріальних ознак у числовий формат за допомогою LabelEncoder. Це корисно для алгоритмів машинного навчання, які вимагають числових даних. Кодування дозволяє алгоритмам легше розуміти та обробляти категоріальні змінні в процесі навчання моделі.

Побудуємо кореляційну матрицю ознак (рис. 2.56).

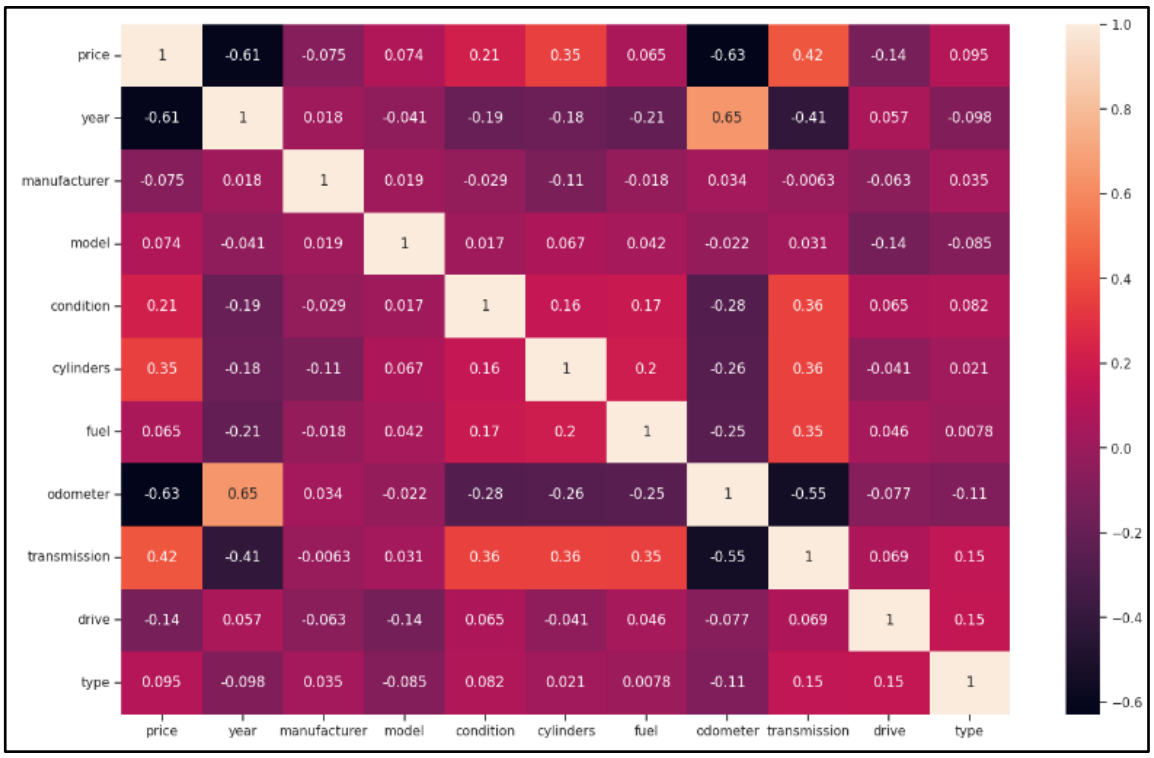

Рисунок 2.56 – Кореляційна матриця ознак

З кореляції видно ступінь залежності між наступними ознаками:

- odometer & year (0.65);
- transmission & price (0.42);
- condition & transmission (0.36);
- cylinders & transmission (0.36);
- fuel & transmission (0.35);
- cylinders & price (0.35);
- condition & price (0.21);
- cylinders & fuel (0.2).

2.6 Висновки

В даному розділі було здійснено вибір оптимальних інформаційних технологій, обрано мову програмування Python, як мову, за допомогою якої здійснюється робота з даними, їх очистка та фільтрація, побудова графіків та робота з методами машинного навчання. В якості IDE (середовища розробки) обрано систему Kaggle Notebooks, що дає доступ до великої кількості бібліотек, модулів та зручного інтерфейсу для написання коду та виведення результатів у вигляді таблиць, графіків, тощо.

Визначено, що задача передбачення цін на вживані автомобілі відноситься до виду машинного навчання з вчителем. Визначено, що наша задача відноситься до задачі регресії, оскільки нам потрібно передбачувати значення змінної на основі набору ознак. Прийнято рішення, додатково провести дослідження на моделях що побудовані на основі дерев рішень.

Здійснено вибір та опис моделей машинного навчання, які допоможуть нам, здійснити передбачення цін на вживані автомобілі. Здійснено опис Python модулів та бібліотек, що використовуються під час розв’язання задачі.

Проведено опис ознак (стовпців) датасету, що використовується для вирішення задачі. Здійснено зчитування та опис вхідного набору даних, його подальша фільтрація та очищення від аномальних та некоректних значень.

Здійснено розвідувальний аналіз даних, побудовано графіки, діаграми, інтерактивні візуалізації та карти. Проведено детальний опис набору даних для виявлення деяких закономірностей, викидів та аномалій, перед подальшим його використанням в побудові моделей машинного навчання.

3 ПОБУДОВА МОДЕЛЕЙ ТА ПЕРЕДБАЧЕННЯ ЦІН НА ВЖИВАНІ АВТОМОБІЛІ

3.1 Підготовка вхідного набору даних для побудови моделей

Створюємо два датафрейми, перший буде мати усі ознаки по автомобілям, а другий лише цільове значення, тобто ціну (рис. 3.1).

```
X = df.drop(['price'], axis=1)
y = df['price']

print(X.shape)
```

```
(203829, 10)
```

Рисунок 3.1 – Створення набору даних з відокремленою цільовою ознакою

Лістинг коду для розподілу даних на тестову та тренувальну вибірки зображено на рисунку 3.2.

```
# Synthesis test0 from train0
train0, test0, train_target0, test_target0 = train_test_split(X, y, test_size=0.2, random_state=42)

print(train0.shape, test0.shape)
```

```
(163063, 10) (40766, 10)
```

Рисунок 3.2 – Розбиття даних на навчальний та тестовий набори train0 та test0

Створення валідаційного набору даних для моделей бустингу зображено на рисунку 3.3

```
# For boosting model
train0b = train0
train_target0b = train_target0
# Synthesis valid as test for selection models
trainb, testb, targetb, target_testb = train_test_split(train0b, train_target0b, test_size=valid_part

print(trainb.shape, testb.shape)
```

```
(114144, 10) (48919, 10)
```

Рисунок 3.3 – Створення валідаційного набору для Boosting моделей

На рисунку 3.4 зображено код стандартизації даних для моделей, що використовуються з бібліотеки Scikit-learn. Для цього використовується клас StandardScaler, що приводить дані до стандартного розподілу з середнім значенням 0 і стандартним відхиленням 1.

```
#For models from Sklearn
scaler = StandardScaler()
train0 = pd.DataFrame(scaler.fit_transform(train0), columns = train0.columns)
```

Рисунок 3.4 – Стандартизація даних для моделей Scikit-learn

Створення валідаційного набору даних для усіх інших моделей зображено на рисунку 3.5

```
# Synthesis valid as test for selection models
train, test, target, target_test = train_test_split(train0, train_target0, test_size=valid_part, ranc
```

Рисунок 3.5 – Створення валідаційного набору для інших моделей

Тренувальні набори даних для моделей бустингу та усіх інших включають 114144 записи, тоді як тестовий набір містить 48919 записів. Додатково, були створені функції, що допомагають обраховувати відносну похибку між передбаченим значенням y_pred та виміряними значеннями

y_meas, а також корінь середньоквадратичної похибки (RMSE), між y_pred та y_meas. На рисунку 3.6 зображено код функцій.

```
def acc_d(y_meas, y_pred):
    # Relative error between predicted y_pred and measured y_meas values
    y_meas, y_pred = np.array(y_meas), np.array(y_pred)
    return np.mean(np.abs((y_pred - y_meas) / y_meas)) #MAPE

def acc_rmse(y_meas, y_pred):
    # RMSE between predicted y_pred and measured y_meas values
    return (MSE(y_meas, y_pred))**0.5
```

Рисунок 3.6 – Функції оцінки точності MAPE та RMSE

3.2 Оптимізація параметрів та тюнінг моделей

На цьому етапі ми готові провести навчання моделей та здійснити передбачення цін на вживані автомобілі. Згадаємо, що у нашому випадку, ми стикаємось з проблемою регресії, тобто намагаємось визначити взаємозв'язок між виходом (ціна автомобіля) та іншими змінними чи характеристиками (рік, марка, пробіг…). Використовується контрольоване навчання, адже навчаємо моделі на заданому наборі даних.

Побудова та тюнінг моделі Linear Regression , та виведення результатів точності та похибок, зображено на рисунку 3.7.

```
# Linear Regression

linreg = LinearRegression()
linreg.fit(train, target)
acc_model(0,linreg,train,test)
```

```
target =  [16995  9000  9650 22590  5995]
ytrain =  [14154.63087109 12267.00219808  6659.63742985 22766.48034208
  4092.87207615]
acc(r2_score) for train = 0.551
acc(relative error) for train = 52.048
acc(rmse) for train = 8938.505
target_test = [ 5500 52990 26990 36590  8900]
ytest = [18382.24301988 37577.26826531 30236.13963076 33733.4553228
 19689.9844449 ]
acc(r2_score) for test = 0.537
acc(relative error) for test = 52.456
acc(rmse) for test = 9118.344
```

Рисунок 3.7 – Тюнінг та результат навчання моделі Linear Regression

З рисунку 3.7 видно, результатом тренування моделі, точність передбачення значень на тренувальній вибірці становить 0.551, а на тестовій – 0.537. Налаштування моделі стандартне, результат передбачення поганий.

Побудова та тюнінг моделі Support Vector Regression, а також, результат роботи моделі наведено на рисунку 3.8.

```
# Support Vector Machines

svr = SVR()
svr.fit(train, target)
acc_model(1,svr,train,test)
```

```
target =  [16995  9000  9650 22590  5995]
ytrain =  [15403.60077062 11966.43673361  9973.11210681 22441.82663061
 12632.90256796]
acc(r2_score) for train = 0.333
acc(relative error) for train = 61.329
acc(rmse) for train = 10897.065
target_test = [ 5500 52990 26990 36590  8900]
ytest = [15281.67161664 25620.44476917 22818.20521372 23246.42938103
 17258.28371437]
acc(r2_score) for test = 0.325
acc(relative error) for test = 61.169
acc(rmse) for test = 11007.605
```

Рисунок 3.8 – Тюнінг та результат навчання моделі SVR

З рисунку 3.8 видно, результат тренування моделі, точність передбачення значень на тренувальній вибірці становить 0.33, а на тестовій – 0.325 Налаштування моделі стандартне, результат передбачення поганий.

Побудова та тюнінг моделі Linear SVR, а також, результат роботи моделі наведено на рисунку 3.9.

```
# Linear SVR

linear_svr = LinearSVR()
linear_svr.fit(train, target)
acc_model(2,linear_svr,train,test)
```

```
target =  [16995  9000  9650 22590  5995]
ytrain =  [ 7977.67267674 10437.37269224  6194.80795676 23188.31281637
  3352.63174146]
acc(r2_score) for train = 0.48
acc(relative error) for train = 42.613
acc(rmse) for train = 9614.85
target_test = [ 5500 52990 26990 36590  8900]
ytest = [15526.10367965 32740.37887786 27015.9472532  29836.20769399
 16920.41253577]
acc(r2_score) for test = 0.467
acc(relative error) for test = 43.016
acc(rmse) for test = 9786.254
```

Рисунок 3.9 – Тюнінг та результат навчання моделі Linear SVR

З рисунку 3.9 видно, результат тренування моделі, точність передбачення значень на тренувальній вибірці становить 0.48, а на тестовій – 0.467. Налаштування моделі стандартне, результат передбачення поганий.

Побудова та тюнінг моделі Stochastic Gradient Descent, а також, результат роботи моделі наведено на рисунку 3.10.

```
# Stochastic Gradient Descent

sgd = SGDRegressor()
sgd.fit(train, target)
acc_model(3,sgd,train,test)
```

```
target =  [16995  9000  9650 22590  5995]
ytrain =  [13450.45957177 11933.2726742   6842.88671088 23709.20170178
  5015.49357736]
acc(r2_score) for train = 0.549
acc(relative error) for train = 51.97
acc(rmse) for train = 8955.223
target_test = [ 5500 52990 26990 36590  8900]
ytest = [17899.12726349 37325.19446695 30548.28497682 33123.61196372
 19792.70219897]
acc(r2_score) for test = 0.536
acc(relative error) for test = 52.368
acc(rmse) for test = 9129.049
```

Рисунок 3.10 – Тюнінг та результат навчання моделі Stochastic Gradient Descent

З рисунку 3.10 видно, результат тренування моделі, точність передбачення значень на тренувальній вибірці становить 0.549, а на тестовій – 0.536. Налаштування моделі стандартне, результат передбачення поганий.

Побудова та тюнінг моделі Decision Tree Regressor, а також, результат роботи моделі наведено на рисунку 3.11.

```
# Decision Tree Regression

decision_tree = DecisionTreeRegressor()
decision_tree.fit(train, target)
acc_model(4,decision_tree,train,test)
```

```
target =  [16995  9000  9650 22590  5995]
ytrain =  [16995.  9000.  9650. 22590.  5995.]
acc(r2_score) for train = 1.0
acc(relative error) for train = 0.11
acc(rmse) for train = 142.386
target_test = [ 5500 52990 26990 36590  8900]
ytest = [26995. 52990. 26990. 36590. 11999.]
acc(r2_score) for test = 0.819
acc(relative error) for test = 19.548
acc(rmse) for test = 5697.891
```

Рисунок 3.11 – Тюнінг та результат навчання моделі Decision Tree Regressor

З рисунку 3.11 видно, результат тренування моделі, точність передбачення значень на тренувальній вибірці становить 1.0, а на тестовій – 0.819.Налаштування моделі стандартне, результат передбачення нормальний, але спостерігається перенавчання.

Побудова та тюнінг моделі Random Forest Regressor, а також, результат роботи моделі наведено на рисунку 3.12.

```
# Random Forest

#random_forest = GridSearchCV(estimator=RandomFores
random_forest = RandomForestRegressor()
random_forest.fit(train, target)
acc_model(5,random_forest,train,test)
```

```
target =  [16995  9000  9650 22590  5995]
ytrain =  [14452.89  8434.    9409.8  22590.    5532.25]
acc(r2_score) for train = 0.988
acc(relative error) for train = 6.244
acc(rmse) for train = 1488.05
target_test = [ 5500 52990 26990 36590  8900]
ytest = [20925.03  52990.    26990.    36590.     9854.015]
acc(r2_score) for test = 0.899
acc(relative error) for test = 16.737
acc(rmse) for test = 4267.664
```

Рисунок 3.12 – Тюнінг та результат навчання моделі Random Forest Regressor

З рисунку 3.12 видно, результат тренування моделі, точність передбачення значень на тренувальній вибірці становить 0.988, а на тестовій – 0.899.Налаштування моделі стандартне, результат передбачення чудовий, але спостерігається невелике перенавчання.

Побудова та тюнінг моделі XGBoost, а також, результат роботи моделі наведено на рисунку 3.13.

```
xgb_clf = xgb.XGBRegressor(objective='reg:squarederror') # # # # # # # # # # # # # # # # # # # # # # # # # # #
parameters = {
    'n_estimators': [60, 100, 120, 140],
    'learning_rate': [0.01, 0.1],
    'max_depth': [5, 7],
    'reg_lambda': [0.5]}

xgb_reg = GridSearchCV(estimator=xgb_clf, param_grid=parameters, cv=5, n_jobs=-1).fit(trainb, targetb)
print("Best score: %0.3f" % xgb_reg.best_score_)
print("Best parameters set:", xgb_reg.best_params_)

acc_boosting_model(7,xgb_reg,trainb,testb)
```

```
Best score: 0.889
Best parameters set: {'learning_rate': 0.1, 'max_depth': 7, 'n_estimators': 140, 'reg_lambda': 0.5}
target =  [16995  9000  9650 22590  5995]
ytrain =  [14046.203   8567.29    7983.6533 23892.758   5224.1675]
acc(r2_score) for train = 0.912
acc(relative error) for train = 20.388
acc(rmse) for train = 3949.687
target_test = [ 5500 52990 26990 36590  8900]
ytest = [12198.615 49834.586 26115.588 31426.496 10423.533]
acc(r2_score) for test = 0.876
acc(relative error) for test = 21.505
acc(rmse) for test = 4720.918
```

Рисунок 3.13 – Тюнінг та результат навчання моделі XGB

З рисунку 3.13 видно, результат тренування моделі, точність передбачення значень на тренувальній вибірці становить 0.912, а на тестовій – 0.876. Налаштування моделі є оптимальними, результат передбачення чудовий.

Побудова та тюнінг моделі LGBM, а також, результат роботи моделі наведено на рисунках 3.14 та 3.15.

```
# %% split training set to validation set
X_train, X_test, y_train, y_test = train_test_split(trainb, targetb, test_size = 0.2, random_state=42)
print(X_train.shape, X_test.shape)


train_set = lgbm.Dataset(X_train, y_train, silent=False)
valid_set = lgbm.Dataset(X_test, y_test, silent=False)

params = {
        'boosting_type':'gbdt',
        'objective': 'regression',
        'num_leaves': 51,
        'learning_rate': 0.05,
        'max_depth': -1,
        'subsample': 0.8,
        'bagging_fraction' : 1,
        'max_bin' : 5000 ,
        'bagging_freq': 20,
        'colsample_bytree': 0.6,
        'metric': 'rmse',
        'min_split_gain': 0.5,
        'min_child_weight': 1,
        'min_child_samples': 10,
        'scale_pos_weight':1,
        'zero_as_missing': True,
        'seed':0,
    }

modelL = lgbm.train(params, train_set = train_set, num_boost_round=8_000,
                    early_stopping_rounds=7_000,verbose_eval=500, valid_sets=valid_set)

acc_boosting_model(8,modelL,trainb,testb,modelL.best_iteration)
```

Рисунок 3.14 – Тюнінг моделі LGBM

```
Did not meet early stopping. Best iteration is:
[9982]  valid_0's rmse: 3539.72
target =  [16995  9000  9650 22590  5995]
ytrain =  [17061.98099732  7356.90428712  8660.90943251 22579.19028039
  5425.48994226]
acc(r2_score) for train = 0.979
acc(relative error) for train = 8.943
acc(rmse) for train = 1953.521
target_test = [ 5500 52990 26990 36590  8900]
ytest = [ 7796.24935277 53013.11861    26783.96890653 35398.59677937
 10914.57693916]
acc(r2_score) for test = 0.921
acc(relative error) for test = 15.416
acc(rmse) for test = 3773.229
```

Рисунок 3.15 – Результат тренування моделі LGBM

З рисунку 3.15 видно, результат тренування моделі, точність передбачення значень на тренувальній вибірці становить 0.979, а на тестовій – 0.921. Налаштування моделі є оптимальними, результат передбачення

чудовий. Додатково, було побудовано графік важливості ознак для даної моделі (рис. 3.16).

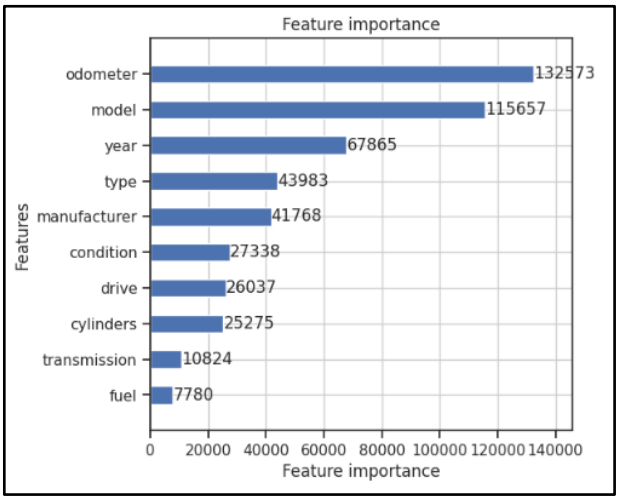

Рисунок 3.16 – Важливість ознак моделі LGBM

Побудова та тюнінг моделі Gradient Boosting Regressor, а також, результат роботи моделі наведено на рисунку 3.17.

```
# params = space_eval(space_gb, best)
params = {'max_depth': 2, 'n_estimators': 152}
params
```

```
{'max_depth': 2, 'n_estimators': 152}
```

```
# Gradient Boosting Regression

gradient_boosting = GradientBoostingRegressor(**params)
gradient_boosting.fit(train, target)
acc_model(9,gradient_boosting,train,test)
```

```
target =  [16995  9000  9650 22590  5995]
ytrain =  [24812.5841056  11520.73645548  7159.48043232 25860.41924315
  2293.29061019]
acc(r2_score) for train = 0.755
acc(relative error) for train = 33.548
acc(rmse) for train = 6599.327
target_test = [ 5500 52990 26990 36590  8900]
ytest = [17864.2735104  35768.156335   29563.50383942 31428.56288766
 13222.36439336]
acc(r2_score) for test = 0.742
acc(relative error) for test = 33.457
acc(rmse) for test = 6808.291
```

Рисунок 3.17 – Тюнінг та результат навчання моделі GradientBoostingRegressor

З рисунку 3.17 видно, результат тренування моделі, точність передбачення значень на тренувальній вибірці становить 0.755, а на тестовій – 0.742. Налаштування моделі стандартне, результат передбачення поганий.

Побудова та тюнінг моделі Ridge Regressor, а також, результат роботи моделі наведено на рисунку 3.18.

```
# Ridge Regressor

ridge = RidgeCV(cv=5)
ridge.fit(train, target)
acc_model(10,ridge,train,test)

target =  [16995  9000  9650 22590  5995]
ytrain =  [14154.13731992 12267.47676088  6660.58459517 22766.91502868
  4093.96052826]
acc(r2_score) for train = 0.551
acc(relative error) for train = 52.048
acc(rmse) for train = 8938.505
target_test = [ 5500 52990 26990 36590  8900]
ytest = [18382.24806168 37576.36793946 30235.76858791 33732.87609115
 19690.13192523]
acc(r2_score) for test = 0.537
acc(relative error) for test = 52.456
acc(rmse) for test = 9118.341
```

Рисунок 3.18 – Тюнінг та результат навчання моделі RidgeRegressor

З рисунку 3.18 видно, результат тренування моделі, точність передбачення значень на тренувальній вибірці становить 0.551, а на тестовій – 0.537. Налаштування моделі стандартне, результат передбачення поганий.

Побудова та тюнінг моделі Bagging Regressor, а також, результат роботи моделі наведено на рисунку 3.19.

```
# Bagging Regressor

bagging = BaggingRegressor()
bagging.fit(train, target)
acc_model(11,bagging,train,test)
```

```
target =  [16995  9000  9650 22590  5995]
ytrain =  [14236.   7809.   9249.5 22590.   5531.5]
acc(r2_score) for train = 0.983
acc(relative error) for train = 6.821
acc(rmse) for train = 1757.003
target_test = [ 5500 52990 26990 36590  8900]
ytest = [16910.5 52990.  26990.  35790.  10466.5]
acc(r2_score) for test = 0.887
acc(relative error) for test = 17.464
acc(rmse) for test = 4513.614
```

Рисунок 3.19 – Тюнінг та результат навчання моделі BaggingRegressor

З рисунку 3.19 видно, результат тренування моделі, точність передбачення значень на тренувальній вибірці становить 0.983, а на тестовій – 0.887. Налаштування моделі стандартне, результат передбачення добрий, але спостерігається перенавчання.

Побудова та тюнінг моделі Extra Trees Regressor, а також, результат роботи моделі наведено на рисунку 3.20.

```
# Extra Trees Regressor

etr = ExtraTreesRegressor()
etr.fit(train, target)
acc_model(12,etr,train,test)
```

```
target =  [16995  9000  9650 22590  5995]
ytrain =  [16995.  9000.  9650. 22590.  5995.]
acc(r2_score) for train = 1.0
acc(relative error) for train = 0.111
acc(rmse) for train = 142.407
target_test = [ 5500 52990 26990 36590  8900]
ytest = [22723.25 52990.   26990.   36590.   11957.34]
acc(r2_score) for test = 0.895
acc(relative error) for test = 16.124
acc(rmse) for test = 4352.558
```

Рисунок 3.20 – Тюнінг та результат навчання моделі ExtraTreesRegressor

З рисунку 3.20 видно, результат тренування моделі, точність передбачення значень на тренувальній вибірці становить 1.0, а на тестовій – 0.895. Налаштування моделі стандартне, результат передбачення добрий, але спостерігається перенавчання.

Побудова та тюнінг моделі AdaBoost Regressor, а також, результат роботи моделі наведено на рисунку 3.21.

```
# AdaBoost Regression

Ada_Boost = AdaBoostRegressor()
Ada_Boost.fit(train, target)
acc_model(13,Ada_Boost,train,test)
```

```
target =  [16995  9000  9650 22590  5995]
ytrain =  [23425.         21651.48783935 21651.48783935 34176.20948063
 19355.77894737]
acc(r2_score) for train = 0.139
acc(relative error) for train = 120.927
acc(rmse) for train = 12376.571
target_test = [ 5500 52990 26990 36590  8900]
ytest = [26976.79414848 33872.61793722 28984.50023599 31841.27694266
 28984.50023599]
acc(r2_score) for test = 0.134
acc(relative error) for test = 119.867
acc(rmse) for test = 12472.833
```

Рисунок 3.21 – Тюнінг та результат навчання моделі AdaBoostRegressor

З рисунку 3.21 видно, результат тренування моделі, точність передбачення значень на тренувальній вибірці становить 0.139, а на тестовій – 0.134. Налаштування моделі стандартне, результат передбачення поганий.

Побудова та тюнінг моделі Voting Regressor, а також, результат роботи моделі наведено на рисунку 3.22.

```
Voting_Reg = VotingRegressor(estimators=[('lin', linreg), ('ridge', ridge), ('sgd', sgd)])
Voting_Reg.fit(train, target)
acc_model(14,Voting_Reg,train,test)
```

```
target =  [16995  9000  9650 22590  5995]
ytrain =  [13920.08246069 12162.38111774  6494.54526543 22904.36319341
  3965.67446778]
acc(r2_score) for train = 0.551
acc(relative error) for train = 51.956
acc(rmse) for train = 8940.029
target_test = [ 5500 52990 26990 36590  8900]
ytest = [18256.60724493 37788.87984464 30441.85239882 33822.3223312
 19671.92956015]
acc(r2_score) for test = 0.537
acc(relative error) for test = 52.381
acc(rmse) for test = 9121.369
```

Рисунок 3.22 – Тюнінг та результат навчання моделі VotingRegressor

З рисунку 3.22 видно, результат тренування моделі, точність передбачення значень на тренувальній вибірці становить 0.551, а на тестовій – 0.537. Налаштування моделі стандартне, результат передбачення поганий.

3.3 Оцінка точності передбачення та порівняння моделей

Створено датафрейм, що містить результати навчання моделей (рис.3.23).

```
print('Prediction accuracy for models by R2 criterion - r2_test')
models.sort_values(by=['r2_test', 'd_test', 'rmse_test'], ascending=False)
```

Prediction accuracy for models by R2 criterion - r2_test

	Model	r2_test	d_test	rmse_test
7	LGBM	0.92	15.42	3,773.20
5	Random Forest	0.90	16.71	4,268.87
11	ExtraTreesRegressor	0.90	16.11	4,332.88
10	BaggingRegressor	0.89	17.50	4,436.09
6	XGB	0.88	21.50	4,720.92
4	Decision Tree Regressor	0.82	19.47	5,675.98
8	GradientBoostingRegressor	0.74	33.46	6,808.29
0	Linear Regression	0.54	52.46	9,118.34
9	RidgeRegressor	0.54	52.46	9,118.34
13	VotingRegressor	0.54	52.34	9,120.19
3	Stochastic Gradient Decent	0.54	53.37	9,135.60
2	Linear SVR	0.47	42.99	9,789.91
12	AdaBoostRegressor	0.35	89.89	10,832.13
1	Support Vector Machines	0.33	61.17	11,007.60

Рисунок 3.23 – Значення R2_Score та похибок моделей

Створено такий самий датафрейм, для порівняння результатів, але, де показано результати навчання моделей на тренувальному наборі даних та валідаційному (рис. 3.24).

```
print('Prediction accuracy for models by R2 criterion - r2_test')
models.sort_values(by=['r2_test', 'r2_train'], ascending=False)
```

```
Prediction accuracy for models by R2 criterion - r2_test
```

	Model	r2_train	r2_test	d_train	d_test	rmse_train	rmse_test
7	LGBM	0.98	0.92	8.94	15.42	1,953.52	3,773.23
5	Random Forest	0.99	0.90	6.23	16.76	1,467.67	4,274.37
11	ExtraTreesRegressor	1.00	0.90	0.11	16.12	142.41	4,352.56
10	BaggingRegressor	0.98	0.89	6.82	17.46	1,757.00	4,513.61
6	XGB	0.91	0.88	20.39	21.50	3,949.69	4,720.92
4	Decision Tree Regressor	1.00	0.82	0.11	19.44	142.39	5,633.08
8	GradientBoostingRegressor	0.76	0.74	33.55	33.46	6,599.33	6,808.29
0	Linear Regression	0.55	0.54	52.05	52.46	8,938.50	9,118.34
3	Stochastic Gradient Decent	0.55	0.54	52.24	52.66	8,940.37	9,118.41
9	RidgeRegressor	0.55	0.54	52.05	52.46	8,938.50	9,118.34
13	VotingRegressor	0.55	0.54	51.96	52.38	8,940.03	9,121.37
2	Linear SVR	0.48	0.47	42.61	43.02	9,614.85	9,786.25
1	Support Vector Machines	0.33	0.33	61.33	61.17	10,897.07	11,007.60
12	AdaBoostRegressor	0.14	0.13	120.93	119.87	12,376.57	12,472.83

Рисунок 3.24 – Порівняння результатів на тренувальному та валідаційному наборах даних

Створено три графіки: графік точності моделей на вибірці із тренувальних (синя лінія) та тестових (червона лінія) даних за коефіцієнтом детермінації R2 (рис. 3.25), графік середньої відносної похибки моделей на вибірці із тренувальних та тестових даних (рис. 3.26), графік кореневої середньоквадратичної похибки моделей (RMSE) (рис. 3.27).

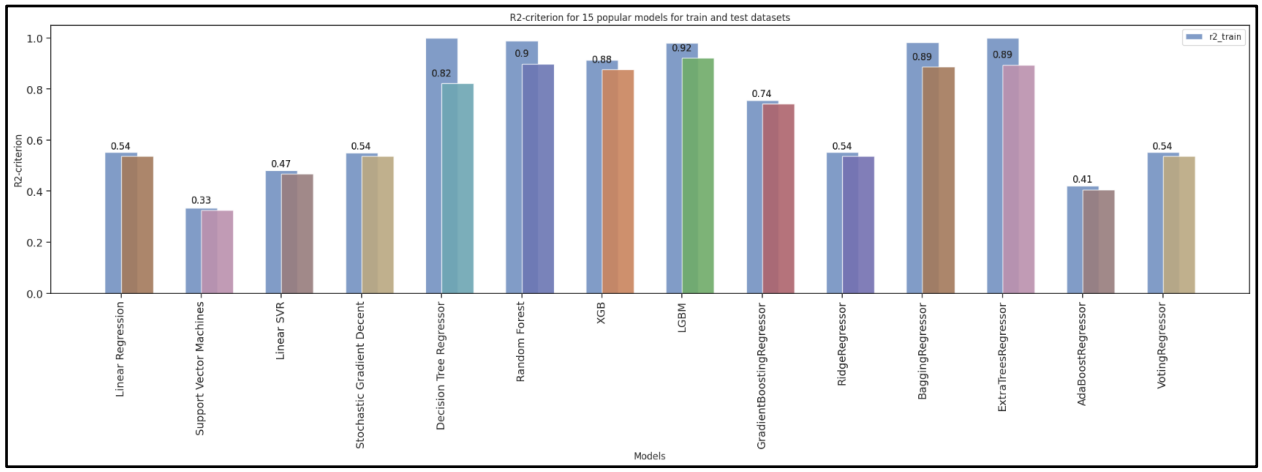

Рисунок 3.25 – Графік точності по 14 моделям за коефіцієнтом детермінації

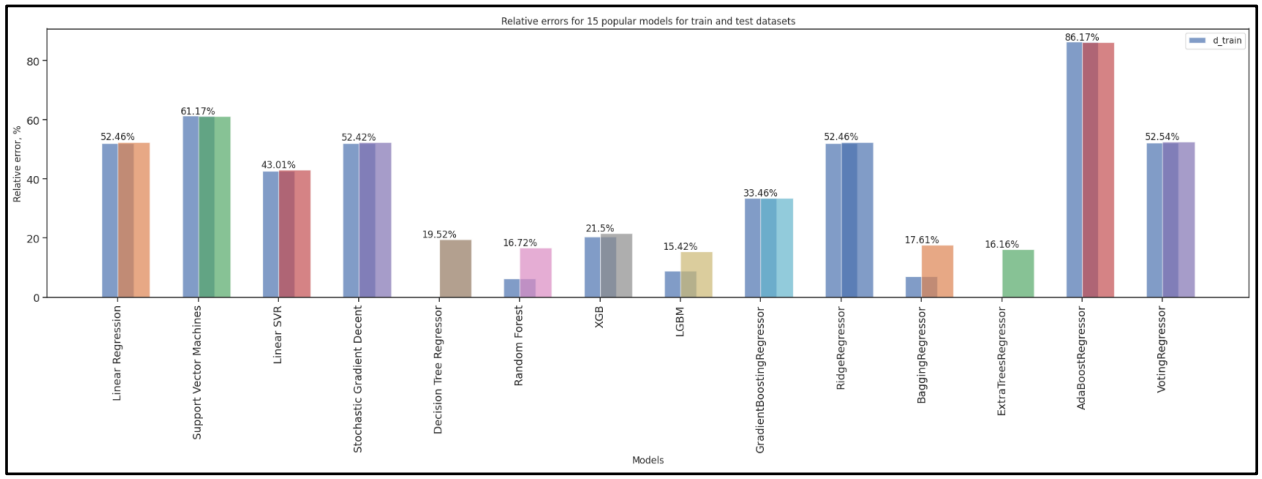

Рисунок 3.26 – Графік середньої відносної похибки по 14 моделям

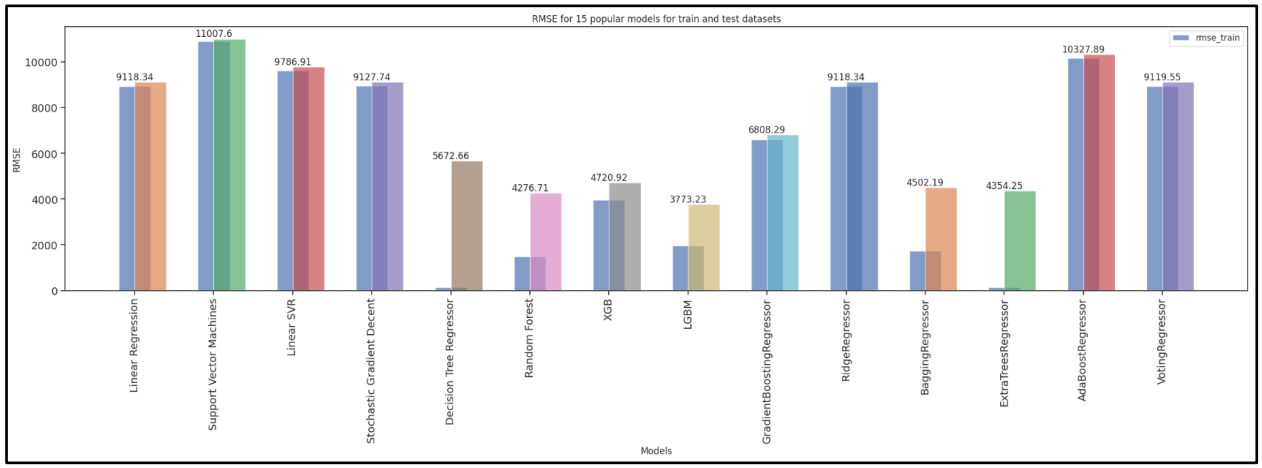

Рисунок 3.27 – Графік кореневої середньоквадратичної похибки по 14 моделям

На всіх графіках видно, що по усім параметрам, найкращий коефіцієнт детермінації (R2) – (0.92), середня відносна похибка (MAPE) – (15.42) та коренева середньоквадратична похибка (RMSE) – (3,773.20) показує модель LGBM.

Виведено датафрейм (рис. 3.28) та діаграму (рис. 3.29) порівняння значень між справжньою ціною автомобілів та передбаченим значенням на основі моделі LGBM, як найкращої моделі.

	Real_price	predicted_prices	difference
27209	35590	35672	82
26582	44990	45198	208
36067	12850	10868	1982
39664	11900	10613	1287
12981	27590	27493	97
14354	12995	14672	1677
23865	13901	13436	465
40202	16995	16727	268
32380	79990	78692	1298
30042	7499	6681	818

Рисунок 3.28 – Результат передбачення ціни та різниця із справжньою

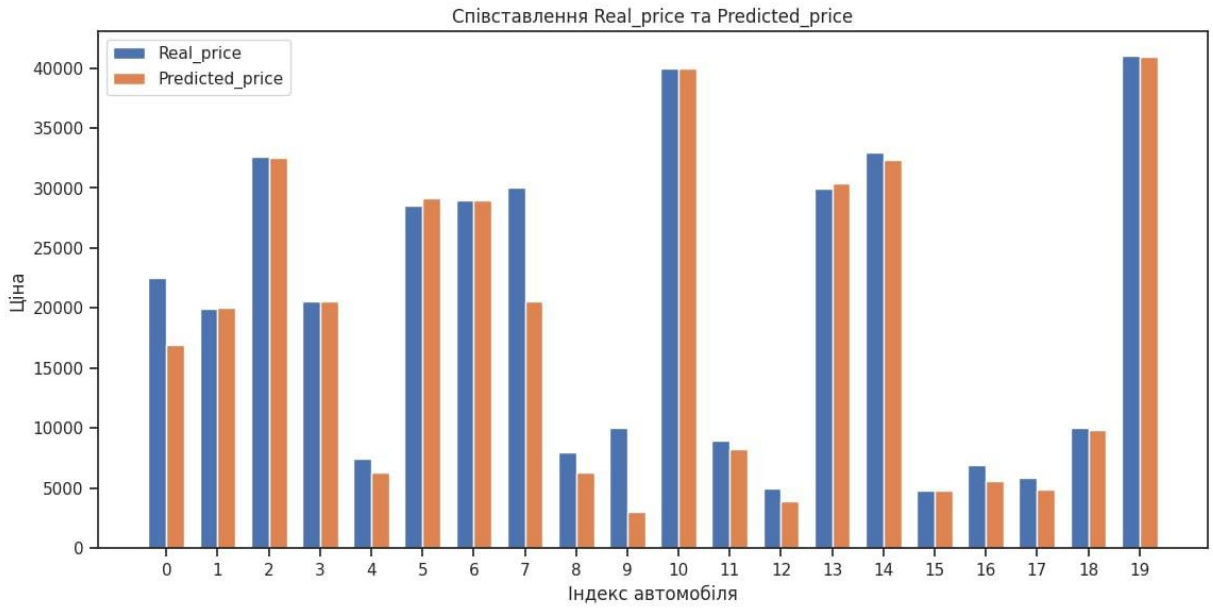

Рисунок 3.29 – Стовпчикова діграма порівняння передбаченої та справжньої ціни

3.4 Висновки

В третьому розділі набір даних було розбито на тренувальні та тестові дані для подальшого тренування регресійних моделей машинного навчання та дерев рішень. Створено функції для обчислення оцінки точності передбачення моделей MAPE та RMSE.

Побудовано та налагоджено 14 моделей машинного навчання, серед яких:

- Linear Regression;
- Support Vector Machines;
- Linear SVR;
- Stochastic Gradient Decent;
- Decision Tree Regressor;
- RandomForestRegressor;
- XGBoostRegressor;
- LGBM;
- GradientBoostingRegressor;
- RidgeRegressor;
- BaggingRegressor;
- ExtraTreesRegressor;
- AdaBoostRegressor;
- VotingRegressor.

Проведено навчання моделей та передбачення цін на вживані автомобілі, порівняно моделі за декількома критеріями, серед яких, найкращою моделлю виявилась LGBM із такими показниками: коефіцієнт детермінації (R2) – 0.92, середньою відносною похибкою (MAPE) – 15.42 та коренева середньоквадратична похибка (RMSE) – 3,773.20. Ці показники свідчать про успішність розв’язання задачі.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

У останні роки відбулося значне зростання обсягів продажу вживаних автомобілів на ринку. Цей ріст обумовлений зростанням попиту на вживані автомобілі, які були придбані за кордоном і ввезені на територію України з наступним розмитненням.

За умов такого стрімкого росту ринку виникає питання про формування конкурентоспроможної ціни на вживаний автомобіль. Окрім традиційних методів визначення ціни експертами, продавці можуть скористатися можливістю порівняти ціни на аналогічні автомобілі за допомогою веб-порталів, спеціалізованих на продажу вживаних авто. Деякі з цих порталів також пропонують інструменти для отримання статистичної інформації щодо ціноутворення на конкретну модель автомобіля.

Однак важливо враховувати, що доступність такої інформації залежить від наявності достатньої кількості записів про конкретні автомобілі. У випадку відсутності необхідних даних від користувача, може бути використано інструменти, що здатні прогнозувати ціни на автомобілі на основі введених даних.

З метою максимально точного прогнозування комерційних переваг від можливої реалізації розробки, планується проведення технологічного аудиту розробленої системи. Такий аудит включатиме оцінку наукового рівня розробки та визначення можливостей для комерційного використання. Технічний аудит буде здійснено експертним методом, залучивши фахівців, які є експертами в галузі наукових досліджень, зокрема у вивченні масивів даних та розробці прогностичних моделей.

Критерії оцінювання наукового-технічного рівня та комерційного потенціалу розробки наведено в таблиці 4.1 [54].

Таблиця 4.1 – Критерії оцінювання наукового-технічного рівня та комерційного потенціалу розробки та бальна оцінка

Бали (за п’ятибальною шкалою)					
	0	1	2	3	4
1	*2*	*3*	*4*	*5*	*6*
Технічна здійсненність концепції					
1	Достовірність Концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздат-ність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижча за ціни аналогів	Ціна продукту значно нижча за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатацій-них витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Продовження таблиці 4.1

Бали (за п’ятибальною шкалою)						
	0	1	2	3	4	
Практична здійсненність						
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї	
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування	
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використо-вуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використо-вуються у виробництві	
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більший 5-ти років	Термін реалізації ідеї менший 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менший 3-х років. Термін окупності інвестицій менший 3-х років	
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що потребує значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту потребує не значних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту	

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 4.2

Таблиця 4.2 Результат оцінювання комерційного потенціалу розробки

Критерії	Експерт		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	4	3
2. Ринкові переваги (наявність аналогів)	3	4	4
3. Ринкові переваги (ціна продукту)	4	4	3
4. Ринкові переваги (технічні властивості)	3	3	4
5. Ринкові переваги (експлуатаційні витрати)	3	2	3
6. Ринкові перспективи (розмір ринку)	3	4	4
7. Ринкові перспективи (конкуренція)	4	4	4
8. Практична здійсненність (наявність фахівців)	4	3	3
9. Практична здійсненність (наявність фінансів)	3	3	3
10. Практична здійсненність (необхідність нових матеріалів)	3	3	3
11. Практична здійсненність (термін реалізації)	3	4	3
12. Практична здійсненність (розробка документів)	4	3	4
Сума балів	$СБ_1$=41	$СБ_2$=41	$СБ_3$=41
Середньоарифметична сума балів $СБ_с$	$СБ_с = \frac{\sum_{i=1}^{3} СБ_i}{3} = \frac{123}{3} = 41$		

За даними таблиці 4.2 можна зробити висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 4.3 [30].

Таблиця 4.3 Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вищий середнього
21…30	Середній
11…20	Нижчий середнього
0…10	Низький

Як видно з таблиці, науково-технічний рівень та комерційний потенціал розробки є високим, що досягається за рахунок того, що оцінка вартості вживаного автомобіля – послуга, без якої важко обійтись в багатьох випадках. Фактично будь-який вживаний автомобіль вимагає розрахунку його ринкової вартості. Високий рівень комерційного потенціалу розробленої інформаційної технології досягається за рахунок того, що остання використовує методи машинного навчання, які дозволяють підвищити точність передбачення у порівнянні з аналогічними технологіями [54].

4.2 Розрахунок витрат на проведення науково-технічної роботи

4.2.1 Витрати на оплату праці

Основна заробітна плата дослідників.

Витрати на основну заробітну плату дослідників ($З_о$) розраховують відповідно до посадових окладів працівників, за формулою [54]:

$$З_0 = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (4.1)$$

де k – кількість посад дослідників, залучених до процесу досліджень;

$М_{пі}$ – місячний посадовий оклад конкретного дослідника, грн;

t_i – кількість днів роботи конкретного дослідника, дн.;

$T_р$ – середня кількість робочих днів в місяці, Тр=21…23 дні.

Проведені розрахунки зведено у таблицю 4.4.

Таблиця 4.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Кількість днів роботи	Витрати на заробітну плату, грн
Керівник проєкту	33600	1600	42	67200
Програміст	29400	1400	42	58800
Всього				126000

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт на тему «Інформаційна технологія аналізу та передбачення цін на вживані автомобілі» розраховується за формулою:

$$З_р = \sum_{i=1}^{n} C_i * t_i, \quad (4.2)$$

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

t_i – час роботи робітника на виконання певної роботи, год.

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

$$C_i = \frac{M_M \cdot K_i \cdot K_C}{T_Р \cdot t_{ЗМ}}, \quad (4.3)$$

де $М_М$ – розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати (залежно від діючого законодавства), приймемо значення 6700 грн;

$К_і$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду [54];

$К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати [54];

$Т_р$ – середня кількість робочих днів в місяці, приблизно Тр = 21…23 дні;

$t_{зм}$ – тривалість зміни, год.

$$C_I = \frac{6700 \cdot 1{,}1 \cdot 1{,}65}{21 \cdot 8} = 72{,}38 \text{ (грн).}$$

$$З_р = 72{,}38 \cdot 8{,}3 = 600{,}79 \text{ (грн)}$$

Проведені розрахунки зведено у таблицю 4.5.

Таблиця 4.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника, грн
Підготовка робочого місця	8,3	2	1,1	72,38	600,79
Інсталяція програмного забезпечення	6,2	3	1,35	88,83	550,78
Компіляція програмних частин	7,5	5	1,7	111,87	839,00
Всього					1990,56

Додаткова заробітна плата дослідників та робітників

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

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.4)$$

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

$$З_{дод} = (126000 + 1990{,}56) \cdot \frac{12}{100\%} = 15358{,}87 \text{ (грн)}.$$

4.2.2 Відрахування на соціальні заходи

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

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

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

$$З_н = (126000 + 1990{,}56 + 15358{,}87) \cdot \frac{22}{100\%} = 31536{,}88 \text{ (грн)}.$$

4.2.3 Сировина та матеріали

Витрати на матеріали (М) у вартісному вираженні розраховуємо окремо для кожного виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j - \sum_{j=1}^{n} В_j \cdot Ц_j, \quad (4.6)$$

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

n – кількість видів матеріалів;

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

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

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

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

Проведені розрахунки зведемо в таблицю 4.6.

Таблиця 4.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Набір канцелярський офісний	325,0	2,0	-	-	715,0
Органайзер офісний	500,0	3,0	-	-	1650,0
Офісний папір А4 500	120,0	3,0	-	-	396,0
Папір для записів А5 250	70,0	3,0	-	-	231
Картридж для принтера	980,0	1,0	-	-	1078
Диск оптичний CD-RW	15,0	2,0	-	-	33
Flash-пам’ять 32GB	150,0	1,0	-	-	165
Всього					4268

4.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі відсутні.

4.2.5 Спецустаткування для наукових (експериментальних) робіт

Витрати на спецустаткування для наукових (експериментальних) робіт відсутні.

4.2.6 Програмне забезпечення на наукових (експериментальних) робіт

Витрати на програмне забезпечення на наукових (експериментальних) робіт відсутні.

4.2.7 Амортизація обладнання, програмних засобів та приміщень

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

$$\text{А}_{\text{обл}} = \frac{\text{Ц}_{\text{б}}}{\text{Т}_{\text{в}}} \cdot \frac{t_{\text{вик}}}{12}, \quad (4.7)$$

де $\text{Ц}_{\text{б}}$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

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

$\text{Т}_{\text{в}}$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$\text{А}_{\text{обл}} = \frac{19500 \cdot 4}{2 \cdot 12} = 3250$ (грн).

Проведені розрахунки зведено в таблицю 4.7.

Таблиця 4.7 – Амортизація відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Персональний комп'ютер проведення розробки та моделювання	195000	2	4	3250.0
Робоче місце інженера розробника	5200	5	2	173,3
Офісне приміщення	900000	20	2	7500
ОС Windows 11	5000	2	2	416,7
Прикладний пакет Microsoft Office 2016	4000	2	2	333,3
Всього				11673,3

4.2.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію ($В_е$) розраховуємо за формулою:

$$В_е = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_е \cdot К_{впі}}{\eta_i}, \quad (4.8)$$

де – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; приймемо Це = 7,50 грн;

$К_{впі}$ – коефіцієнт, що враховує використання потужності, Квпі <1;

η_i – коефіцієнт корисної дії обладнання, η_i<1.

$В_е = 0,25 \cdot 320,0 \cdot 7,50 \cdot 0,95 / 0,97 = 587,63$ (грн).

Проведені розрахунки зведено в таблицю 4.8.

Таблиця 4.8 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер проведення розробки та моделювання	0,25	320,0	587,63
Робоче місце інженера	0,12	300,0	264.43
Всього			852,06

4.2.9 Службові відрядження

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{св} = (З_0 + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (4.9)$$

де Нсв – норма нарахування за статтею «Службові відрядження», приймемо Нсв = 25%.

$$В_{св} = (126000 + 1990,56) \cdot \frac{25}{100\%} = 31997,64 \text{ (грн)}.$$

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

Витрати на роботи, які виконують сторонні підприємства, установи і організації відсутні.

4.2.11 Інші витрати

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

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.10)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$I_в = (126000 + 1990{,}56) \cdot \frac{50}{100\%} = 63995{,}28$ (грн).

4.2.12 Накладні (загальновиробничі) витрати

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

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.11)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 100%.

$В_{нзв}$ = (126000+ 1990,56) · 100 / 100% = 127990,56 (грн).

Витрати на проведення науково-технічної роботи на тему «Інформаційна технологія аналізу та передбачення цін на вживані автомобілі» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$В_{заг} = З_О + З_Р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}, \quad (4.12)$$

$$В_{заг} = 126000 + 1990{,}56 + 15358{,}87 + 31536{,}88 + 4268 + 0 + 0 + 0 + 3250 + 852{,}06 + 31997{,}64 + 0 + 63995{,}28 + 127990{,}56 = 315445{,}71 \text{ (грн)}.$$

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

$$ЗВ = \frac{В_{заг}}{\eta}, \quad (4.13)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-технічної роботи, приймемо η =0,90.

$$ЗВ = \frac{315445{,}71}{0{,}90} = 350495{,}23 \text{ (грн)}.$$

4.3 Оцінювання важливості та наукової значимості науково-технічної роботи фундаментального чи пошукового характеру

Для обґрунтування доцільності виконання науково-технічної роботи використовується спеціальний комплексний показник, що враховує важливість, результативність роботи, можливість впровадження її результатів у виробництво, величину витрат на роботу.

Комплексний показник $К_Р$ рівня науково-технічної роботи розраховується за формулою:

$$К_р = \frac{I^n \cdot T_C \cdot R}{B \cdot t}, \quad (4.14)$$

де I – коефіцієнт важливості роботи, I = 2...5;

n – коефіцієнт використання результатів роботи; n = 0, коли результати роботи не будуть використовуватись; n =1, коли результати роботи будуть використовуватись частково; n = 2, коли результати роботи будуть використовуватись в дослідно-конструкторських розробках; n = 3, коли результати можуть використовуватись навіть без проведення дослідно-конструкторських розробок;

T_C – коефіцієнт складності роботи, T_C =1...3;

R – коефіцієнт результативності роботи; якщо результати роботи плануються вище відомих, то R = 4 ; якщо результати роботи відповідають відомому рівню, то R = 3; якщо нижче відомих результатів, то R =1;

B – вартість науково-технічної роботи, тис. грн;

t – час проведення дослідження, років.

Визначення показників I, n, T_C, R, B, t здійснюється експертним шляхом або на основі нормативів.

Якщо KP >1, то науково-технічну роботу можна вважати ефективною з високим науковим, технічним і економічним рівнями.

$$K_p = \frac{4^2 \cdot 2 \cdot 4}{350{,}49523 \cdot 0{,}2} = 1.8267.$$

З результату обрахунку показника видно, що науково-технічна робота є ефективною, має високий науковий, технічний і економічний рівні.

4.4 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

а) вказати, з якого часу можуть бути впроваджені результати науково-технічної розробки;

б) зазначити, протягом скількох років після впровадження цієї науково-технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 4-х років після її впровадження);

в) кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб’єктів (зацікавлених осіб) цього попиту;

г) визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

При розрахунку економічної ефективності потрібно обов’язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу.

При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

– абсолютного економічного ефекту (чистого дисконтованого доходу);

– внутрішньої економічної дохідності (внутрішньої норми дохідності);

– терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором можна об'єднати, враховуючи визначені ситуації з відповідними умовами.

Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних [54]:

$$\Delta П_i = (\pm \Delta Ц_о + N + Ц_о \cdot \Delta N_i)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (4.15)$$

де $\pm\Delta Ц_о$ – зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай, таким показником може бути зміна ціни реалізації одиниці нової розробки в аналізованому році (відносно року до впровадження цієї розробки); $\pm\Delta Ц_о$ може мати як додатне, так і від'ємне значення (від'ємне – при зниженні ціни відносно року до впровадження цієї розробки, додатне – при зростанні ціни). Приймемо зростання на 155,50 грн;

N – основний кількісний показник, який визначає величину попиту на аналогічні чи подібні розробки у році до впровадження результатів нової науково-технічної розробки. Кількість споживачів приймемо 35000 осіб;

$Ц_о$ – основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році, $Ц_о = Ц_б \pm \Delta Ц_о$;

$Ц_б$ – основний якісний показник, який визначає ціну реалізації існуючої (базової) науково-технічної розробки у році до впровадження результатів. Приймемо 1850,00 грн;

ΔN – зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай таким показником може бути зростання попиту на науково-технічну розробку в аналізованому році (відносно року до впровадження цієї розробки);

λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість становить 18%, а коефіцієнт λ =0,8333;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту (послуги). Рекомендується брати ρ = 0,2...0,5. Приймемо ρ = 0,35;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2021 році ϑ =18%

Збільшення чистого прибутку 1-го року:

$$\Delta П_1 = (155{,}50 \cdot 30000{,}0 + 2005{,}50 \cdot 3000) \cdot 0{,}83 \cdot 0{,}35 \cdot \left(1 - \frac{0{,}18}{100\%}\right) =$$

3096850,22 (грн).

Збільшення чистого прибутку 2-го року:

$$\Delta П_2 = (155{,}50 \cdot 30000{,}0 + 2005{,}50 \cdot 8000) \cdot 0{,}83 \cdot 0{,}35 \cdot \left(1 - \frac{0{,}18}{100\%}\right) =$$

6012610,20 (грн).

Збільшення чистого прибутку 3-го року:

$$\Delta П_3 = (155{,}50 \cdot 30000{,}0 + 2005{,}50 \cdot 15000) \cdot 0{,}83 \cdot 0{,}35 \cdot \left(1 - \frac{0{,}18}{100\%}\right) =$$

10063959,70 (грн).

Збільшення чистого прибутку 4-го року:

$$\Delta П_4 = (155{,}50 \cdot 30000{,}0 + 2005{,}50 \cdot 19000) \cdot 0{,}83 \cdot 0{,}35 \cdot \left(1 - \frac{0{,}18}{100\%}\right) =$$

12375581,08 (грн).

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

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (4.16)$$

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

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

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ = 0,05…0,15. Приймемо τ =0,15;

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

$$ПП = \frac{3096850,22}{(1+0,15)^1} + \frac{6012610,20}{(1+0,15)^2} + \frac{10063959,70}{(1+0,15)^3} + \frac{12375581,08}{(1+0,15)^4} = 2687991,55 + 4541847,16 + 6614360,47 + 7080778,56 = 20944977,74 \text{ (грн)}.$$

Далі розраховують величину початкових інвестицій PV , які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{\text{інв}} \cdot ЗВ, \quad (4.17)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}$ = 2...5, але може бути і більшим. Приймемо $k_{інв}$ = 2,1;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 350495,23 грн.

$$PV = k_{\text{інв}} \cdot ЗВ = 2,1 \cdot 350495,23 = 735039,98 \text{ (грн)}.$$

Тоді абсолютний економічний ефект $Е_{абс}$ або чистий приведений дохід (NPV, Net Present Value) для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{\text{абс}} = \text{ПП} - PV, \qquad (4.18)$$

де ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 20944977,74 грн;

PV – теперішня вартість початкових інвестицій, 735039,98 грн.

$E_{\text{абс}} = 20944977{,}74 - 735039{,}98 = 20209937{,}76$ (грн).

Внутрішня економічна дохідність інвестицій Ев , які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою:

$$E_{\text{в}} = \sqrt[T_{\text{ж}}]{1 + \frac{E_{\text{абс}}}{PV}} - 1, \qquad (4.19)$$

де $Е_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 20209937.76 грн;

PV – теперішня вартість початкових інвестицій, 735039.98 грн;

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

$$E_{\text{в}} = \sqrt[T_{\text{ж}}]{1 + \frac{E_{\text{абс}}}{PV}} - 1 = (1 + 20209937{,}76/735039{,}98)^{1/4} = 2{,}42.$$

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

$$\tau_{\text{мін}} = d + f, \tag{4.20}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,16;

f – показник, що характеризує ризикованість вкладення інвестицій; зазвичай величина f =0,05...0,5, але може бути і значно вищою. Приймемо 0,3.

$\tau_{\text{мін}} = d + f = 0{,}16 + 0{,}3 = 0{,}46 < 2{,}42$, що свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, вища мінімальної внутрішньої дохідності. Тобто, потенційних інвестор може бути зацікавлений у фінансуванні науково-технічної розробки за темою «Інформаційна технологія аналізу та передбачення цін на вживані автомобілі» та виведенні її на ринок, тобто в її комерціалізації.

Період окупності інвестицій $T_{ок}$ (DPP, Discounted Payback Period), які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{\text{ок}} = \frac{1}{E_{\text{в}}}, \tag{4.21}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{\text{ок}} = \frac{1}{2{,}42} = 0{,}41$ р.

Якщо $T_{ок}$ < 3-х років, то це свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок [55].

4.5 Висновки

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія аналізу та передбачення цін на вживані автомобілі» становить 41 бал, що свідчить про комерційну важливість

проведення даних досліджень (рівень комерційного потенціалу розробки високий).

Термін окупності становить 0,41 року, що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже, можна зробити висновок про доцільність проведення науково-технічної роботи на тему «Інформаційна технологія аналізу та передбачення цін на вживані автомобілі».

ВИСНОВКИ

В кваліфікаційній магістерській роботі розроблено інформаційну технологію аналізу та передбачення цін на вживані автомобілі США методами машинного навчання.

Проведено аналіз інтелектуальних технологій та систем рекомендування цін, огляд ринку онлайн-продажів вживаних автомобілів в США та Україні. Проаналізовано прогнозовані тенденції розвитку цього ринку в США. Наведено переваги та недоліки купівлі вживаних та нових автомобілів. Розглянуто сервіси, що надають інформацію про вживані автомобілі.

На основі проведеного аналізу було зроблено висновок, що існуючі сервіси не забезпечують достатньої інформації для точного передбачення цін на вживані автомобілі. Це пов'язано з тим, що в їх базах даних містяться лише рекомендації по наявних схожих продажах. Такий підхід не дозволяє врахувати всі ключові фактори, які впливають на ціну автомобіля, такі як:

– Марка і модель автомобіля;
– Рік випуску;
– Пробіг;
– Стан автомобіля;
– Комплектація;
– Місто розташування.

Результатом огляду та аналізу популярних аналогічних рішень у системі Kaggle та просторі інтернет, визначено, що точність аналогів, які використовують схожі методи машинного навчання, складає 0.89 і менше.

Здійснено вибір оптимальних інформаційних технологій та запропоновано, обрано мову програмування Python, як мову, за допомогою якої здійснювалась робота з даними, їх очистка та фільтрація, побудова графіків і не тільки, робота з методами машинного навчання. В якості IDE (середовища розробки) обрано систему Kaggle Notebooks, що дає доступ до

великої кількості бібліотек, модулів та зручного інтерфейсу для написання коду та виведення результатів у вигляді таблиць, графіків, тощо.

Визначено, що проблема (задача) передбачення цін на вживані автомобілі відноситься до виду машинного навчання з вчителем. Визначено, що наша задача відноситься до задачі регресії, оскільки нам потрібно передбачувати значення змінної на основі набору ознак, але також, нам потрібні моделі які побудовані на основі дерев рішень.

Здійснено вибір та опис моделей машинного навчання, які допомагали нам, здійснювати передбачення цін на вживані автомобілі. Здійснено опис Python модулів та бібліотек, що використовувались під час розв'язання задачі.

Проведено опис ознак (стовпців) датасету, що використовувався для вирішення проблеми. Здійснено зчитування та опис вхідного набору даних.

Виконано передобробку даних, а саме фільтрацію та очищення від аномальних та некоректних значень.

На етапі розвідувального аналізу даних, побудовано графіки, діаграми, інтерактивні візуалізації та карти. Проведено поверхневий та детальний опис набору даних для виявлення деяких закономірностей, викидів та аномалій, перед подальшим його використанням в побудові моделей машинного навчання.

Набір даних було розбито на тренувальні та тестові дані для подальшого тренування регресійних моделей машинного навчання та дерев рішень. Створено функції для обчислення оцінки точності передбачення моделей MAPE та RMSE.

Побудовано та налагоджено 14 моделей машинного навчання, серед яких:

- Linear Regression;
- Support Vector Machines;
- Linear SVR;
- Stochastic Gradient Decent;
- Decision Tree Regressor;

- RandomForestRegressor;
- XGBoostRegressor;
- LGBM;
- GradientBoostingRegressor;
- RidgeRegressor;
- BaggingRegressor;
- ExtraTreesRegressor;
- AdaBoostRegressor;
- VotingRegressor.

Проведено навчання моделей та передбачення цін на вживані автомобілі, порівняно моделі за декількома критеріями, серед яких, найкращою моделлю виявилась LGBM із такими показниками: коефіцієнт детермінації (R2) – 0.92, середньою відносною похибкою (MAPE) – 15.42 та коренева середньоквадратична похибка (RMSE) – 3,773.20.. Ці показники є кращими за показники найкращих аналогів, що свідчить про успішність розв’язання задачі.

Економічна частина роботи містить розрахунок доцільності проведення науково-технічної роботи. Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія аналізу та передбачення цін на вживані автомобілі» становить 41 бал, що свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки високий).

Термін окупності становить.0,41 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже, можна зробити висновок про доцільність проведення науково-технічної роботи на тему «Інформаційна технологія аналізу та передбачення цін на вживані автомобілі».

Таким чином, дістала подальший розвиток інформаційна технологія передбачення цін на вживані автомобілі із використанням моделей машинного навчання. Така технологія здатна враховувати більшість факторів, які впливають на ціну автомобіля, і надає більш точне передбачення цільової ознаки. А отже, поставлене завдання на магістерську кваліфікаційну роботу виконано в повному обсязі.

За тематикою дослідження опубліковано доповідь на тему «Інформаційна технологія аналізу та передбачення цін на вживані автомобілі» на LIII Науково-технічній конференції факультету інтелектуальних інформаційних технологій та автоматизації ВНТУ з публікацією тез (м. Вінниця, 2023-2024 рр.).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гіжевський В.В Жуков С.О. Інформаційна технологія аналізу та передбачення цін на вживані автомобілі. LIII Наукова-технічна конференція факультету інтелектуальних інформаційних технологій та автоматизації: зб. Матеріалів конференції. Вінниця, 2023. URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2024/paper/view/19626/16249 (Дата звернення: 01.10.2023)

2. Стасюк Б. Б. АНАЛІЗ ОСНОВНИХ ТЕНДЕНЦІЙ НА АВТОМОБІЛЬНОМУ РИНКУ УКРАЇНИ: Вісник НУВГП, м. Рівне, 1-23 червня 2023 р. Рівне, 2023. С. 37-39. (Дата звернення: 01.10.2023)

3. OICA підсумувала обсяги глобального автовиробництва, Укравтопром, 14 березня 2023. URL: https://ukrautoprom.com.ua/oica-pidsumuvala-obsyagy-globalnogo-avtovyrobnycztva (Дата звернення: 05.10.2023)

4. Автомобілебудування. Транспортні засоби, USAID, 2021 ТОВ «Ернст енд Янг». URL: https://ukraineinvest.gov.ua/wp-content/uploads/2021/08/FDI-Strategy-Section-2-Advanced-Manufacturing-Sector-UKR-1.pdf (Дата звернення: 06.10.2023)

5. Чи справді в Україні подешевшали вживані авто: дослідження. Станіслав Бучацький. Auto24, 27 липня 2022. URL: https://auto.24tv.ua/vzhyvani_avto_v_ukraini_2022_tsiny_ohliad_zminy_doslidzhennia_n39889 (Дата звернення: 08.10.2023)

6. Чичерін Валентин Вікторович Розвиток вітчизняного автомобілебудування на основі міжнародного трансферу технологій, Магістерська дисертація. КПІ. Київ, 2022. URL: https://ela.kpi.ua/bitstream/123456789/58418/1/Chycherin_magistr.pdf (Дата звернення: 15.11.2023)

7. Pros and cons of buying a second hand car. Autodeal, Dec 10, 2021. URL: https://www.autodeal.com.ph/articles/car-features/pros-and-cons-buying-second-hand-car (Дата звернення: 23.11.2023)

8. Buying a New vs. Pre-Owned Car, Truck, or SUV in Countryside, IL . URL: https://www.westfieldford.com/research/buying-new-vs-used.htm (Дата звернення: 25.11.2023)

9. Нові, чи вживані. Які автомобілі краще: вивчаємо переваги та недоліки. РБК-Україна, 27 листопада 2023. URL: https://www.rbc.ua/rus/news/novi-chi-vzhivani-ki-avtomobili-krashche-1701029466.html (Дата звернення 29.11.2023)

10. В. Б. Мокін, А. В. Лосенко, і М. В. Дратований, «ІНТЕЛЕКТУАЛЬНА ТЕХНОЛОГІЯ АНАЛІЗУ ТА ПЕРЕДБАЧЕННЯ ЦІН НА ВЖИВАНІ АВТОМОБІЛІ», Вісник ВПІ, вип. 6, с. 62–72, Груд. 2019. (Дата звернення 05.10.2023)

11. Craigslist. Wikipedia. URL: https://uk.wikipedia.org/wiki/Craigslist (Дата звернення: 02.10.2023)

12. Craigslist_Car Data Analysis, Kaggle Notebook. JEONGBIN PARK, 02.03.2021. URL: https://www.kaggle.com/code/jeongbinpark/craigslist-car-data-analysis (Дата звернення: 15.10.2023)

13. Used Cars Data Analysis and Visualization (EDA), Kaggle Notebook, İSMAIL SEFA AKDENIZ, 24.03.2020 . URL: https://www.kaggle.com/code/ismailsefa/used-cars-data-analysis-and-visualization-eda/notebook (Дата звернення: 16.10.2023)

14. Used Car Price Prediction (End-to End Project), Github, Abhash Panwar, 25.04.2021 . URL: https://github.com/abhashpanwar/used-car-price-prediction/blob/master/README.md (Дата звернення: 17.10.2023)

15. Auto RIA . URL: https://auto.ria.com/uk/ (Дата звернення: 15.10.2023)

16. Підрахунок середньої ціни . URL: https://api-docsv2.readthedocs.io/ru/latest/auto_ria/used_cars/average-price.html (Дата звернення: 16.10.2023)

17. REST API AUTO.RIA . URL: https://github.com/riacom/auto-ria-rest-api (Дата звернення: 20.10.2023)

18. Збірник наукових праць за матеріалами І Всеукраїнської науково-практичної конференції студентів і аспірантів «ТЕОРЕТИЧНІ ТА ПРИКЛАДНІ АСПЕКТИ РОЗРОБКИ КОМП'ЮТЕРНИХ СИСТЕМ. НУБіП України. Київ, 29 березня 2018 року. URL: https://nubip.edu.ua/sites/default/files/u214/proceedingstaacsd2018.pdf (Дата звернення: 23.10.2023)

19. MATLAB, History and Syntax. Wikipedia. URL: https://en.wikipedia.org/wiki/MATLAB (Дата звернення: 25.10.2023)

20. Головний сайт спільноти розробників Python. URL: https://www.python.org/ (Дата звернення: 25.10.2023)

21. What Is Python Used For? A Beginner`s Guide. Coursera. Coursera Staff, 25 жовт. 2023. URL: https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python (Дата звернення: 26.10.2023)

22. R (programming language). Wikipedia. URL: https://en.wikipedia.org/wiki/R_(programming_language) (Дата звернення: 28.10.2023)

23. SQL. SQL fundamentals. Atlassian. URL: https://www.atlassian.com/data/sql (Дата звернення: 01.11.2023)

24. 10 Best Python IDE & Code Editors. Carl Matheous Simpao. Hackr.io. URL: https://hackr.io/blog/best-python-ide (Дата звернення: 01.11.2023)

25. Best Python IDE for Development. URL: https://djangostars.com/blog/python-ide/ (Дата звернення: 02.11.2023)

26. How to use Kaggle. Notebooks. URL: https://www.kaggle.com/docs/notebooks (Дата звернення: 08.10.2023)

27. Ethem Alpaydin Introduction to Machine Learning, third edition. Cambridge, Massachusetts London, England, 2014. URL: https://dl.matlabyar.com/siavash/ML/Book/Ethem%20Alpaydin-Introduction%20to%20Machine%20Learning-The%20MIT%20Press%20(2014).pdf (Дата звернення: 10.10.2023)

28. Trevor Hastie, Robert Tibshirani, and Jerome Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition 1 січня 2016 . URL: https://www.amazon.com/Elements-StatisticalLearning-Prediction-Statistics/ (Дата звернення: 15.10.2023)

29. Машинне навчання. Вікіпедія . URL: https://uk.wikipedia.org/wiki/MachineLearning (Дата звернення: 15.11.2023)

30. Linear Regression (Python Implementation). GeeksForGeeks. URL: https://www.geeksforgeeks.org/linear-regression-python-implementation/ (Дата звернення: 15.11.2023)

31. Classifying data using Support Vector Machines(SVMs) in Python. GeeksForGeeks. URL: https://www.geeksforgeeks.org/classifying-data-using-support-vector-machinessvms-in-python/ (Дата звернення: 15.11.2023)

32. Support Vector Regression (SVR) using Linear and Non-Linear Kernels in Scikit Learn. GeeksForGeeks. URL: https://www.geeksforgeeks.org/support-vector-regression-svr-using-linear-and-non-linear-kernels-in-scikit-learn/ (Дата звернення: 15.11.2023)

33. Stochastic Gradient Descent Algorithm With Python and NumPy. Mirko Stojiljvic. Real Python. URL: https://realpython.com/gradient-descent-algorithm-python/ (Дата звернення: 15.11.2023)

34. Decision Tree Regression Explained with Implementation in Python. The Click Reader. Medium, Oct 19, 2021. URL: https://medium.com/@theclickreader/decision-tree-regression-explained-with-implementation-in-python-1e6e48aa7a47 (Дата звернення: 15.11.2023)

35. Random Forest in Python. Will Koehrsen. Towards Data Science, Dec 27, 2017. URL: https://towardsdatascience.com/random-forest-in-python-24d0893d51c0 (Дата звернення: 16.11.2023)

36. XGBoost explanation. GeeksForGeeks. URL: https://www.geeksforgeeks.org/xgboost/ (Дата звернення: 16.11.2023)

37. Light Gradient Boosting Machine. James Lamb. Github. URL: https://github.com/microsoft/LightGBM (Дата звернення: 16.11.2023)

38. All You Need to Know about Gradient Boosting Algorithm − Part 1. Regression. Tomonori Masui. Towards Data Science, Jan 20, 2022. URL: https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-1-regression-2520a34a502 (Дата звернення: 16.11.2023)

39. Ordinary Least Squares and Ridge Regression Variance in Scikit Learn. GeeksForGeeks. URL: https://www.geeksforgeeks.org/ordinary-least-squares-and-ridge-regression-variance-in-scikit-learn/ (Дата звернення: 16.11.2023)

40. Bagging. Bootstrap resampling. Aggregating. Bagging in scikit-learn. Github. URL: https://inria.github.io/scikit-learn-mooc/python_scripts/ensemble_bagging.html (Дата звернення: 16.11.2023)

41. Regression Example with an Extra-Trees Method in Python. DataTechNotes. URL: https://www.datatechnotes.com/2020/06/regression-example-with-extratrees-in-python.html (Дата звернення: 17.11.2023)

42. AdaBoostRegressor and Classifier with python and r programming. Yashwanth Reddy. Medium, Apr 2, 2023. URL: https://medium.com/@reddyyashu20/adaboostregressor-and-classifier-with-python-and-r-programming-8c4d49f3a50d (Дата звернення: 17.11.2023)

43. Voting Regressor. Step-by-step implementation. GeeksForGeeks. URL: https://www.geeksforgeeks.org/voting-regressor/ (Дата звернення: 17.11.2023)

44. Numpy Introduction. W3Schools. URL: https://www.w3schools.com/python/numpy/numpy_intro.asp (Дата звернення: 17.11.2023)

45. Pandas official. User Guide. URL: https://pandas.pydata.org/docs/user_guide/index.html (Дата звернення: 18.11.2023)

46. Seaborn: statistical data visualization. URL: https://seaborn.pydata.org/ (Дата звернення: 18.11.2023)

47. Matplotlib: Visualization with Python . URL: https://matplotlib.org/ (Дата звернення: 18.11.2023)

48. Scikit-Learn: Machine Learning in Python . URL: https://scikit-learn.org/stable/index.html (Дата звернення: 18.11.2023)

49. SciPy. Fundamental algorithms for scientific computing in Python. Eric Jones, Pearu Peterson, Travis Oliphant. 2023. URL: https://scipy.org/ (Дата звернення: 18.11.2023)

50. LightGBM documentation. Microsoft Corporation. 2023 . URL: https://scikit-learn.org/stable/index.html (Дата звернення: 16.11.2023)

51. XGBoost Documentstion. Github. Xgboost developers. 2022 . URL: https://xgboost.readthedocs.io/en/stable/ (Дата звернення: 17.11.2023)

52. Folium. Python data. Leaflet.js maps. Github. Rob Story. 2013 . URL: https://python-visualization.github.io/folium/latest/ (Дата звернення: 25.11.2023)

53. Used Cars Dataset. Kaggle, Austin Reese, 2021 . URL: https://www.kaggle.com/datasets/austinreese/craigslist-carstrucks-data (Дата звернення: 01.10.2023)

54. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с. (Дата звернення: 25.11.2023)

55. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с. (Дата звернення: 25.11.2023)

Додаток А

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

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
____ д.т.н., проф. Віталій МОКІН
«__» _____2023 року

ТЕХНІЧНЕ ЗАВДАННЯ
на магістерську кваліфікаційну роботу
ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АНАЛІЗУ ТА ПЕРЕДБАЧЕННЯ ЦІН НА ВЖИВАНІ АВТОМОБІЛІ
08-34.МКР.002.00.000 ТЗ

Керівник: к.т.н., доц.
_____ Сергій ЖУКОВ
«__» _____ 2023 р.
Розробив студент гр. 2ІСТ-22м
_____Владислав ГІЖЕВСЬКИЙ
«__» _____ 2023 р.

Вінниця 2023

1. Підстава для проведення робіт.
Підставою для виконання роботи є наказ №__ по ВНТУ від «__» _____ 2023р., та індивідуальне завдання на МКР, затверджене протоколом №__ засідання кафедри САІТ від «__» _____ 2023р.
2. Джерела розробки:
1) Used Cars Dataset. Kaggle, Austin Reese, 2021. URL: https://www.kaggle.com/datasets/austinreese/craigslist-carstrucks-data
2) Used Car Price Prediction. URL: https://www.kaggle.com/code/hizhevskyivladyslav/used-car-price-prediction-r2-score-0-925?scriptVersionId=152914987
3. Мета і призначення роботи.
Метою даної роботи є підвищення точності передбачення цін на вживані автомобілі за даними США методами машинного навчання шляхом створення інформаційної технології аналізу та передбачення цін на вживані автомобілі.
4. Вихідні дані для проведення робіт.
Набір даних з платформи Kaggle по продажу вживаних автомобілів на американському веб-сайті Craigslist періодом до 2022 року;
5. Методи дослідження.
Методи машинного навчання, регресійні моделі та моделі на основі дерев рішень.
6. Етапи роботи і терміни їх виконання:
а) Аналіз предметної області передбачення цін на вживані автомобілі
____ – ____
б) Огляд оптимальних інформаційних технологій та вхідного набору даних
____ – ____
в) Побудова моделей та передбачення цін на вживані автомобілі
____ – ____
г) Економічна частина ____ – ____
д) Оформлення матеріалів до захисту МКР ____ – ____
7. Очікувані результати та порядок реалізації.
Інформаційна технологія аналізу та передбачення цін на вживані автомобілі, результати передбачення якої є кращими порівняно з аналогами.
8. Вимоги до розробленої документації.
Текстова та ілюстративна частини роботи оформлені у відповідності до вимог «Методичних вказівок до виконання магістерських кваліфікаційних робіт для студентів спеціальності 126 «Інформаційні системи та технології» (освітня програма «Інформаційні технології аналізу даних та зображень»).
9. Порядок приймання роботи.
Публічний захист «___» _____ 2023 р.
Початок розробки «___» _____ 2023 р.
Граничні терміни виконання МКР «___» _____ 2023 р.
Розробив студент групи 2ІСТ-22м _____ Владислав ГІЖЕВСЬКИЙ

Додаток Б

Протокол перевірки магістерської кваліфікаційної роботи на наявність текстових запозичень

Назва роботи: «Інформаційна технологія аналізу та передбачення цін на вживані автомобілі»
Тип роботи: магістерська кваліфікаційна робота
Підрозділ: кафедра САІТ

Показники звіту подібності Unicheck

Оригінальність 94,94 % Схожість 5,06 %

Аналіз звіту подібності (відмітити потрібне)

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

Особа, відповідальна за перевірку (підпис) Сергій ЖУКОВ

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

Автор роботи (підпис) Владислав ГІЖЕВСЬКИЙ

Керівник роботи (підпис) Сергій ЖУКОВ

Додаток В

Лістинг програми

```
X = df.drop(['price'], axis=1)
y = df['price']

print(X.shape)
X

# Synthesis test0 from train0
train0, test0, train_target0, test_target0 = train_test_split(X, y, test_size=0.2, random_state=42)

print(train0.shape, test0.shape)
train0

# For boosting model
train0b = train0
train_target0b = train_target0
# Synthesis valid as test for selection models
trainb, testb, targetb, target_testb = train_test_split(train0b, train_target0b, test_size=valid_part, random_state=0)

print(trainb.shape, testb.shape)
trainb

#For models from Sklearn
scaler = StandardScaler()
train0 = pd.DataFrame(scaler.fit_transform(train0), columns = train0.columns)

train0.head()

len(train0)

# Synthesis valid as test for selection models
train, test, target, target_test = train_test_split(train0, train_target0, test_size=valid_part, random_state=0)

train.head(3)

train.info()

test.head(3)

test.info()


cc_train_r2 = []
acc_test_r2 = []
acc_train_d = []
acc_test_d = []
acc_train_rmse = []
acc_test_rmse = []

def acc_d(y_meas, y_pred):
    # Relative error between predicted y_pred and measured y_meas values
```

```
    y_meas, y_pred = np.array(y_meas), np.array(y_pred)
    return np.mean(np.abs((y_pred - y_meas) / y_meas)) #MAPE

def acc_rmse(y_meas, y_pred):
    # RMSE between predicted y_pred and measured y_meas values
    return (MSE(y_meas, y_pred))**0.5

def acc_boosting_model(num,model,train,test,num_iteration=0):
    # Calculation of accuracy of boosting model by different metrics

    global acc_train_r2, acc_test_r2, acc_train_d, acc_test_d, acc_train_rmse, acc_test_rmse

    if num_iteration > 0:
        ytrain = model.predict(train, num_iteration = num_iteration)
        ytest = model.predict(test, num_iteration = num_iteration)
    else:
        ytrain = model.predict(train)
        ytest = model.predict(test)

    print('target = ', targetb[:5].values)
    print('ytrain = ', ytrain[:5])

    acc_train_r2_num = round(r2_score(targetb, ytrain), 3)
    print('acc(r2_score) for train =', acc_train_r2_num)
    acc_train_r2.insert(num, acc_train_r2_num)

    acc_train_d_num = round(acc_d(targetb, ytrain) * 100, 3)
    print('acc(relative error) for train =', acc_train_d_num)
    acc_train_d.insert(num, acc_train_d_num)

    acc_train_rmse_num = round(acc_rmse(targetb, ytrain), 3)
    print('acc(rmse) for train =', acc_train_rmse_num)
    acc_train_rmse.insert(num, acc_train_rmse_num)

    print('target_test =', target_testb[:5].values)
    print('ytest =', ytest[:5])

    acc_test_r2_num = round(r2_score(target_testb, ytest), 3)
    print('acc(r2_score) for test =', acc_test_r2_num)
    acc_test_r2.insert(num, acc_test_r2_num)

    acc_test_d_num = round(acc_d(target_testb, ytest) * 100, 3)
    print('acc(relative error) for test =', acc_test_d_num)
    acc_test_d.insert(num, acc_test_d_num)

    acc_test_rmse_num = round(acc_rmse(target_testb, ytest), 3)
    print('acc(rmse) for test =', acc_test_rmse_num)
    acc_test_rmse.insert(num, acc_test_rmse_num)

def acc_model(num,model,train,test):
    # Calculation of accuracy of model акщь Sklearn by different metrics

    global acc_train_r2, acc_test_r2, acc_train_d, acc_test_d, acc_train_rmse, acc_test_rmse
```

```
    ytrain = model.predict(train)
    ytest = model.predict(test)

    print('target = ', target[:5].values)
    print('ytrain = ', ytrain[:5])

    acc_train_r2_num = round(r2_score(target, ytrain), 3)
    print('acc(r2_score) for train =', acc_train_r2_num)
    acc_train_r2.insert(num, acc_train_r2_num)

    acc_train_d_num = round(acc_d(target, ytrain) * 100, 3)
    print('acc(relative error) for train =', acc_train_d_num)
    acc_train_d.insert(num, acc_train_d_num)

    acc_train_rmse_num = round(acc_rmse(target, ytrain), 3)
    print('acc(rmse) for train =', acc_train_rmse_num)
    acc_train_rmse.insert(num, acc_train_rmse_num)

    print('target_test =', target_test[:5].values)
    print('ytest =', ytest[:5])

    acc_test_r2_num = round(r2_score(target_test, ytest), 3)
    print('acc(r2_score) for test =', acc_test_r2_num)
    acc_test_r2.insert(num, acc_test_r2_num)

    acc_test_d_num = round(acc_d(target_test, ytest) * 100, 3)
    print('acc(relative error) for test =', acc_test_d_num)
    acc_test_d.insert(num, acc_test_d_num)

    acc_test_rmse_num = round(acc_rmse(target_test, ytest), 3)
    print('acc(rmse) for test =', acc_test_rmse_num)
    acc_test_rmse.insert(num, acc_test_rmse_num)

# Linear Regression

linreg = LinearRegression()
linreg.fit(train, target)
acc_model(0,linreg,train,test)

# WORKS FOR TOO LONG
# Support Vector Machines

svr = SVR()
svr.fit(train, target)
acc_model(1,svr,train,test)

# Linear SVR

linear_svr = LinearSVR()
linear_svr.fit(train, target)
acc_model(2,linear_svr,train,test)

# Stochastic Gradient Descent
```

```
sgd = SGDRegressor()
sgd.fit(train, target)
acc_model(3,sgd,train,test)

# Decision Tree Regression

decision_tree = DecisionTreeRegressor()
decision_tree.fit(train, target)
acc_model(4,decision_tree,train,test)

# Random Forest

#random_forest = GridSearchCV(estimator=RandomForestRegressor(), param_grid={'n_estimators': [100, 1000]},
cv=5)
random_forest = RandomForestRegressor()
random_forest.fit(train, target)
acc_model(5,random_forest,train,test)

# # MLPRegressor

# mlp = MLPRegressor()
# param_grid = {'hidden_layer_sizes': [i for i in range(2,20)],
#               'activation': ['relu'],
#               'solver': ['adam'],
#               'learning_rate': ['constant'],
#               'learning_rate_init': [0.01],
#               'power_t': [0.5],
#               'alpha': [0.0001],
#               'max_iter': [1000],
#               'early_stopping': [True],
#               'warm_start': [False]}
# mlp_GS = GridSearchCV(mlp, param_grid=param_grid,
#                       cv=10, verbose=True, pre_dispatch='2*n_jobs')
# mlp_GS.fit(train, target)
# acc_model(6,mlp_GS,train,test)

xgb_clf = xgb.XGBRegressor(objective='reg:squarederror') # # # # # # # # # # # # # # # # # # # # # # # # # # # # #
# # # # # # #
parameters = {
    'n_estimators': [60, 100, 120, 140],
    'learning_rate': [0.01, 0.1],
    'max_depth': [5, 7],
    'reg_lambda': [0.5]}

xgb_reg = GridSearchCV(estimator=xgb_clf, param_grid=parameters, cv=5, n_jobs=-1).fit(trainb, targetb)
print("Best score: %0.3f" % xgb_reg.best_score_)
print("Best parameters set:", xgb_reg.best_params_)

acc_boosting_model(7,xgb_reg,trainb,testb)

# LGBM_train, LGBM_test, LGBM_train_target, LGBM_test_target = train_test_split(X, y, test_size=0.2,
random_state=42)

# train_set = lgbm.Dataset(LGBM_train, LGBM_train_target, silent=False) ##########
```

```
# valid_set = lgbm.Dataset(LGBM_test, LGBM_test_target, silent=False) ##########

# params = {
#         'boosting_type':'gbdt',
#         'objective': 'regression',
#         'num_leaves': 51,
#         'learning_rate': 0.05,
#         'max_depth': -1,
#         'subsample': 0.8,
#         'bagging_fraction' : 1,
#         'max_bin' : 5000 ,
#         'bagging_freq': 20,
#         'colsample_bytree': 0.6,
#         'metric': 'rmse',
#         'min_split_gain': 0.5,
#         'min_child_weight': 1,
#         'min_child_samples': 10,
#         'scale_pos_weight':1,
#         'zero_as_missing': True,
#         'seed':0,
#     }

# modelL = lgbm.train(params, train_set = train_set, num_boost_round=10_000,
#                     early_stopping_rounds=9_000,verbose_eval=500, valid_sets=valid_set)

# acc_boosting_model(8,modelL,trainb,testb,modelL.best_iteration)

# %% split training set to validation set
X_train, X_test, y_train, y_test = train_test_split(trainb, targetb, test_size = 0.2, random_state=42)
print(X_train.shape, X_test.shape)

train_set = lgbm.Dataset(X_train, y_train, silent=False)
valid_set = lgbm.Dataset(X_test, y_test, silent=False)

params = {
        'boosting_type':'gbdt',
        'objective': 'regression',
        'num_leaves': 51,
        'learning_rate': 0.05,
        'max_depth': -1,
        'subsample': 0.8,
        'bagging_fraction' : 1,
        'max_bin' : 5000 ,
        'bagging_freq': 20,
        'colsample_bytree': 0.6,
        'metric': 'rmse',
        'min_split_gain': 0.5,
        'min_child_weight': 1,
        'min_child_samples': 10,
        'scale_pos_weight':1,
        'zero_as_missing': True,
        'seed':0,
    }
```

```
modelL = lgbm.train(params, train_set = train_set, num_boost_round=10_000,
                    early_stopping_rounds=9_000,verbose_eval=500, valid_sets=valid_set)

acc_boosting_model(8,modelL,trainb,testb,modelL.best_iteration)

fig =  plt.figure(figsize = (5,5))
axes = fig.add_subplot(111)
lgbm.plot_importance(modelL,ax = axes,height = 0.5)
plt.show();
plt.close()

# def hyperopt_gb_score(params):
#     clf = GradientBoostingRegressor(**params)
#     current_score = cross_val_score(clf, train, target, cv=10).mean()
#     print(current_score, params)
#     return current_score

# space_gb = {
#             'n_estimators': hp.choice('n_estimators', range(100, 1000)),
#             'max_depth': hp.choice('max_depth', np.arange(2, 10, dtype=int))
#         }

# best = fmin(fn=hyperopt_gb_score, space=space_gb, algo=tpe.suggest, max_evals=10)
# print('best:')
# print(best)

# params = space_eval(space_gb, best)
params = {'max_depth': 2, 'n_estimators': 152}
params

# Gradient Boosting Regression

gradient_boosting = GradientBoostingRegressor(**params)
gradient_boosting.fit(train, target)
acc_model(9,gradient_boosting,train,test)

# Ridge Regressor

ridge = RidgeCV(cv=5)
ridge.fit(train, target)
acc_model(10,ridge,train,test)

# Bagging Regressor

bagging = BaggingRegressor()
bagging.fit(train, target)
acc_model(11,bagging,train,test)

# Extra Trees Regressor

etr = ExtraTreesRegressor()
etr.fit(train, target)
acc_model(12,etr,train,test)
```

```
# AdaBoost Regression

Ada_Boost = AdaBoostRegressor()
Ada_Boost.fit(train, target)
acc_model(13,Ada_Boost,train,test)

Voting_Reg = VotingRegressor(estimators=[('lin', linreg), ('ridge', ridge), ('sgd', sgd)])
Voting_Reg.fit(train, target)
acc_model(14,Voting_Reg,train,test)

print(len(acc_train_r2))
print(len(acc_test_r2))
print(len(acc_train_d))
print(len(acc_test_d))
print(len(acc_train_rmse))
print(len(acc_test_rmse))

#no MLPRegressor

models = pd.DataFrame({
   'Model': ['Linear Regression',
             'Support Vector Machines',
             'Linear SVR',
#              'MLPRegressor',
             'Stochastic Gradient Decent',
             'Decision Tree Regressor',
             'Random Forest',
             'XGB',
             'LGBM',
             'GradientBoostingRegressor',
             'RidgeRegressor',
             'BaggingRegressor',
             'ExtraTreesRegressor',
             'AdaBoostRegressor',
             'VotingRegressor'],

   'r2_train': acc_train_r2,
   'r2_test': acc_test_r2,
   'd_train': acc_train_d,
   'd_test': acc_test_d,
   'rmse_train': acc_train_rmse,
   'rmse_test': acc_test_rmse
                 })

pd.options.display.float_format = '{:,.2f}'.format

# print('Prediction accuracy for models by R2 criterion - r2_test')
# models.sort_values(by=['r2_test', 'r2_train', 'd_train', 'd_test', 'rmse_train', 'rmse_test'], ascending=False)

print('Prediction accuracy for models by R2 criterion - r2_test')
models.sort_values(by=['r2_test', 'r2_train'], ascending=False)

print('Prediction accuracy for models by relative error - d_test')
```

```
models.sort_values(by=['d_test', 'd_train'], ascending=True)

print('Prediction accuracy for models by RMSE - rmse_test')
models.sort_values(by=['rmse_test', 'rmse_train'], ascending=True)

# # Plot
# plt.figure(figsize=[25,6])
# xx = models['Model']
# plt.tick_params(labelsize=14)
# plt.plot(xx, models['r2_train'], label = 'r2_train')
# plt.plot(xx, models['r2_test'], label = 'r2_test')
# plt.legend()
# plt.title('R2-criterion for 15 popular models for train and test datasets')
# plt.xlabel('Models')
# plt.ylabel('R2-criterion')
# plt.xticks(xx, rotation='vertical')
# plt.savefig('graph1.png')
# plt.show()

# Plot with Bar Columns
plt.figure(figsize=[25,6])
xx = models['Model']
plt.tick_params(labelsize=14)
plt.bar(xx, models['r2_train'], label='r2_train', width=0.4, align='center')
plt.bar(xx, models['r2_test'], label='r2_test', width=0.4, align='edge')
plt.legend()
plt.title('R2-criterion for 15 popular models for train and test datasets')
plt.xlabel('Models')
plt.ylabel('R2-criterion')
plt.xticks(xx, rotation='vertical')
plt.savefig('bar_graph.png')
plt.show()

# Plot with Bar Columns and Values
plt.figure(figsize=[25,6])
xx = models['Model']
plt.tick_params(labelsize=14)

# Plotting the bars for r2_train
plt.bar(xx, models['r2_train'], label='r2_train', width=0.4, align='center', alpha=0.7)

# Plotting the bars for r2_test with text labels
for i, value in enumerate(models['r2_test']):
    plt.bar(xx[i], value, width=0.4, align='edge', alpha=0.7)
    plt.text(xx[i], value + 0.02, str(round(value, 2)), ha='center', va='bottom', fontsize=12)

# Plotting the bars for r2_test with text labels
for i, value in enumerate(models['r2_test']):
    plt.bar(xx[i], value, width=0.4, align='edge', alpha=0.7)
    plt.text(xx[i], value + 0.02, str(round(value, 2)), ha='center', va='bottom', fontsize=12)

plt.legend()
plt.title('R2-criterion for 15 popular models for train and test datasets')
plt.xlabel('Models')
```

```
plt.ylabel('R2-criterion')
plt.xticks(xx, rotation='vertical')
plt.savefig('bar_graph_with_values.png')
plt.show()

# # Plot
# plt.figure(figsize=[25,6])
# xx = models['Model']
# plt.tick_params(labelsize=14)
# plt.plot(xx, models['d_train'], label = 'd_train')
# plt.plot(xx, models['d_test'], label = 'd_test')
# plt.legend()
# plt.title('Relative errors for 15 popular models for train and test datasets')
# plt.xlabel('Models')
# plt.ylabel('Relative error, %')
# plt.xticks(xx, rotation='vertical')
# plt.savefig('graph2.png')
# plt.show()

# Plot with Bar Columns
plt.figure(figsize=[25,6])
xx = models['Model']
plt.tick_params(labelsize=14)
plt.bar(xx, models['d_train'], label='d_train', width=0.4, align='center')
plt.bar(xx, models['d_test'], label='d_test', width=0.4, align='edge')
plt.legend()
plt.title('Relative errors for 15 popular models for train and test datasets')
plt.xlabel('Models')
plt.ylabel('Relative error, %')
plt.xticks(xx, rotation='vertical')
plt.savefig('bar_graph2.png')
plt.show()


 Plot with Bar Columns and Values
plt.figure(figsize=[25,6])
xx = models['Model']
plt.tick_params(labelsize=14)

# Plotting the bars for d_train
plt.bar(xx, models['d_train'], label='d_train', width=0.4, align='center', alpha=0.7)

# Plotting the bars for d_test with text labels
for i, value in enumerate(models['d_test']):
    plt.bar(xx[i], value, width=0.4, align='edge', alpha=0.7)
    plt.text(xx[i], value + 0.02, f"{round(value, 2)}%", ha='center', va='bottom', fontsize=12)

plt.legend()
plt.title('Relative errors for 15 popular models for train and test datasets')
plt.xlabel('Models')
plt.ylabel('Relative error, %')
plt.xticks(xx, rotation='vertical')
plt.savefig('bar_graph2_with_values.png')
plt.show()
```

```
# # Plot
# plt.figure(figsize=[25,6])
# xx = models['Model']
# plt.tick_params(labelsize=14)
# plt.plot(xx, models['rmse_train'], label = 'rmse_train')
# plt.plot(xx, models['rmse_test'], label = 'rmse_test')
# plt.legend()
# plt.title('RMSE for 15 popular models for train and test datasets')
# plt.xlabel('Models')
# plt.ylabel('RMSE')
# plt.xticks(xx, rotation='vertical')
# plt.savefig('graph3.png')
# plt.show()

# Plot with Bar Columns
plt.figure(figsize=[25,6])
xx = models['Model']
plt.tick_params(labelsize=14)
plt.bar(xx, models['rmse_train'], label='rmse_train', width=0.4, align='center')
plt.bar(xx, models['rmse_test'], label='rmse_test', width=0.4, align='edge')
plt.legend()
plt.title('RMSE for 15 popular models for train and test datasets')
plt.xlabel('Models')
plt.ylabel('RMSE')
plt.xticks(xx, rotation='vertical')
plt.savefig('bar_graph3.png')
plt.show()

# Plot with Bar Columns and Values
plt.figure(figsize=[25,6])
xx = models['Model']
plt.tick_params(labelsize=14)

# Plotting the bars for rmse_train
plt.bar(xx, models['rmse_train'], label='rmse_train', width=0.4, align='center', alpha=0.7)

# Plotting the bars for rmse_test with text labels
for i, value in enumerate(models['rmse_test']):
    plt.bar(xx[i], value, width=0.4, align='edge', alpha=0.7)
    plt.text(xx[i], value + 0.02, f"{round(value, 2)}", ha='center', va='bottom', fontsize=12)

plt.legend()
plt.title('RMSE for 15 popular models for train and test datasets')
plt.xlabel('Models')
plt.ylabel('RMSE')
plt.xticks(xx, rotation='vertical')
plt.savefig('bar_graph3_with_values.png')
plt.show()

#no MLPRegressor

models = pd.DataFrame({
    'Model': ['Linear Regression',
```

```
            'Support Vector Machines',
            'Linear SVR',
#             'MLPRegressor',
            'Stochastic Gradient Decent',
            'Decision Tree Regressor',
            'Random Forest',
            'XGB',
            'LGBM',
            'GradientBoostingRegressor',
            'RidgeRegressor',
            'BaggingRegressor',
            'ExtraTreesRegressor',
            'AdaBoostRegressor',
            'VotingRegressor'],

    'r2_test': acc_test_r2,
    'd_test': acc_test_d,
    'rmse_test': acc_test_rmse
                })

print('Prediction accuracy for models by R2 criterion - r2_test')
models.sort_values(by=['r2_test', 'd_test', 'rmse_test'], ascending=False)

test0.info()

test0.head(3)

#For models from Sklearn
testn = pd.DataFrame(scaler.transform(test0), columns = test0.columns)
testn.head()

#Linear Regression model for basic train
linreg_predict = linreg.predict(testn)
linreg_predict[:3]

# Random Forest model for basic train
rf_predict = random_forest.predict(testn)
rf_predict[:3]

#Bagging Regressor model for basic train
bg_predict = bagging.predict(testn)
bg_predict[:3]

# XGB Regression model for basic train
xgb_predict = xgb_reg.predict(testn)
xgb_predict[:3]

# LGB Regression model for basic train
lgb_predict = modelL.predict(test0)
lgb_predict[:3]

# Extra Trees Regressor model for basic train
etr_predict = etr.predict(testn)
etr_predict[:3]
```

```
final_df = test_target0.values
final_df = pd.DataFrame(final_df,columns=['Real_price'])
final_df['predicted_prices'] = lgb_predict.astype(int)
final_df['difference'] = abs(final_df['Real_price'] - final_df['predicted_prices']).astype(int)
# final_df.head(20)
final_df.sample(10)

r2_score(test_target0, lgb_predict.astype(int))

mean_real_price = round(final_df['Real_price'].mean(), 0)
mean_predicted_prices = round(final_df['predicted_prices'].mean(), 0)
mean_difference = round(final_df['difference'].mean(), 0)
# Create and append mean values to DataFrame
mean_val = []
mean_val.append(('real_price', mean_real_price))
mean_val.append(('predicted_prices', mean_predicted_prices))
mean_val.append(('difference', mean_difference))
pd.DataFrame(mean_val, columns = ('Name', 'Average'))
```

Додаток Г

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ АНАЛІЗУ ТА ПЕРЕДБАЧЕННЯ ЦІН НА ВЖИВАНІ АВТОМОБІЛІ

Нормоконтроль: к.т.н., доцент

_____ Сергій ЖУКОВ

«___» _____ 2023 р.

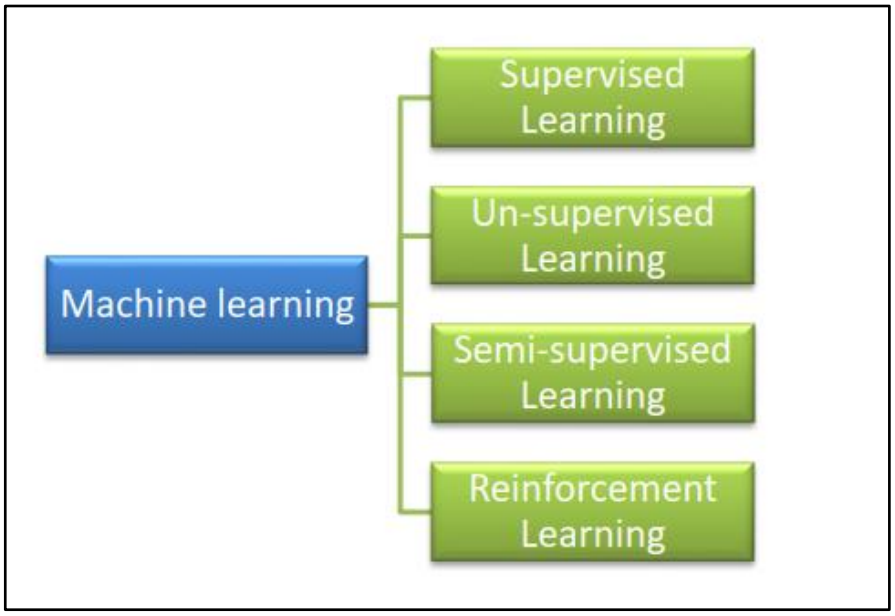

Рисунок Г.1 – Машинне навчання та його складові

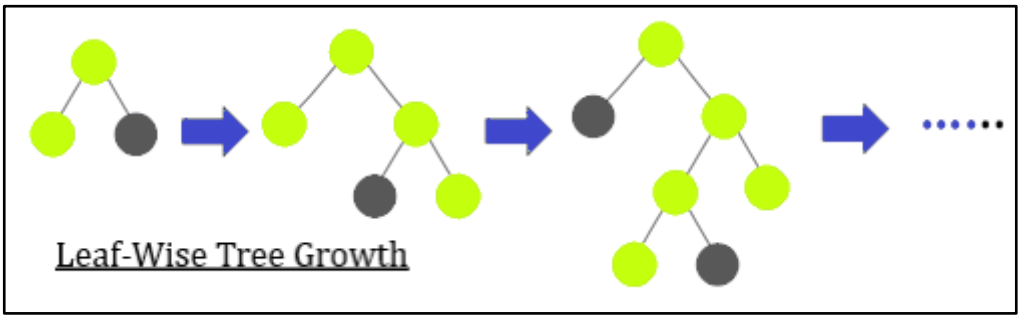

Рисунок Г.2 – Архітектура LightGBM

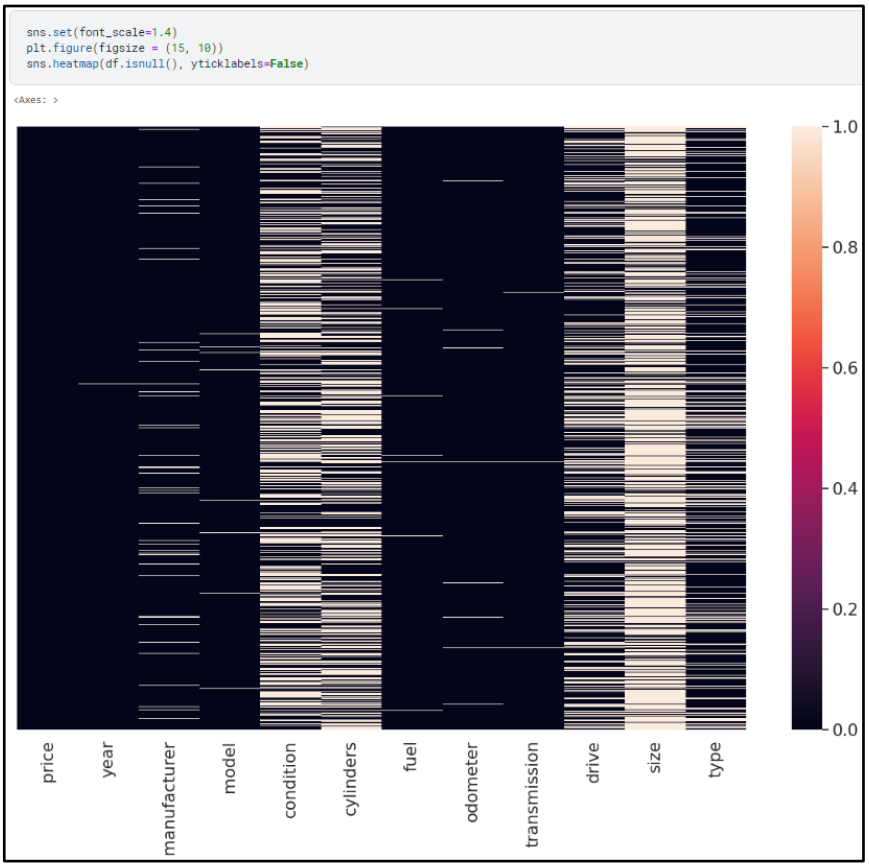

Рисунок Г.3 – Теплова карта відсутніх даних

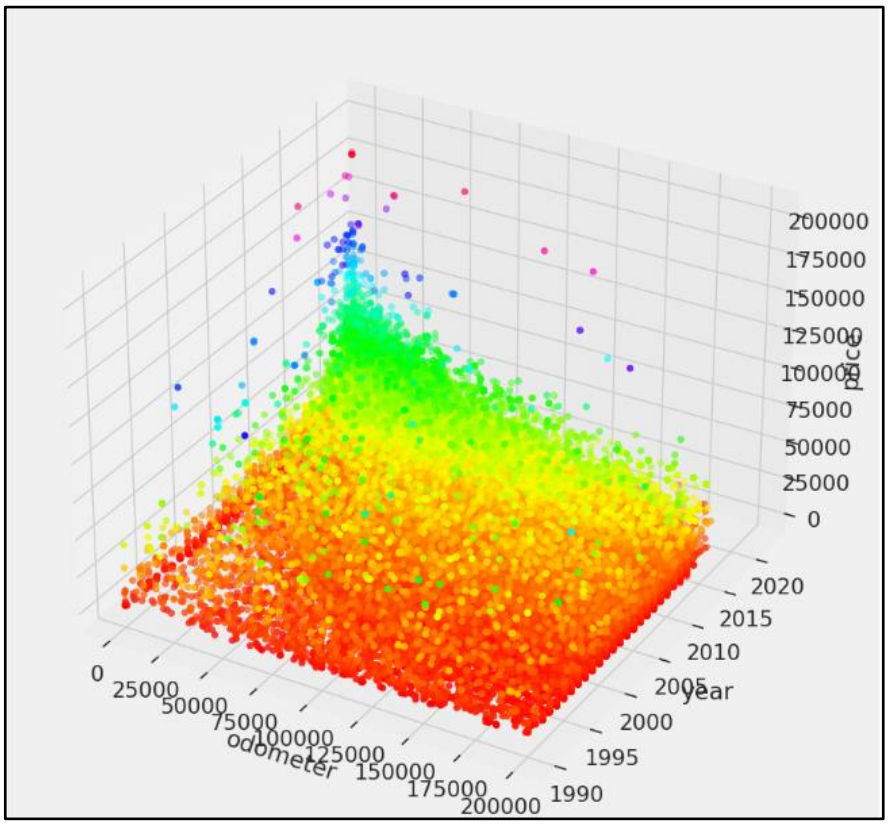

Рисунок Г.4 – Тривимірна діаграма співвідношення ознак

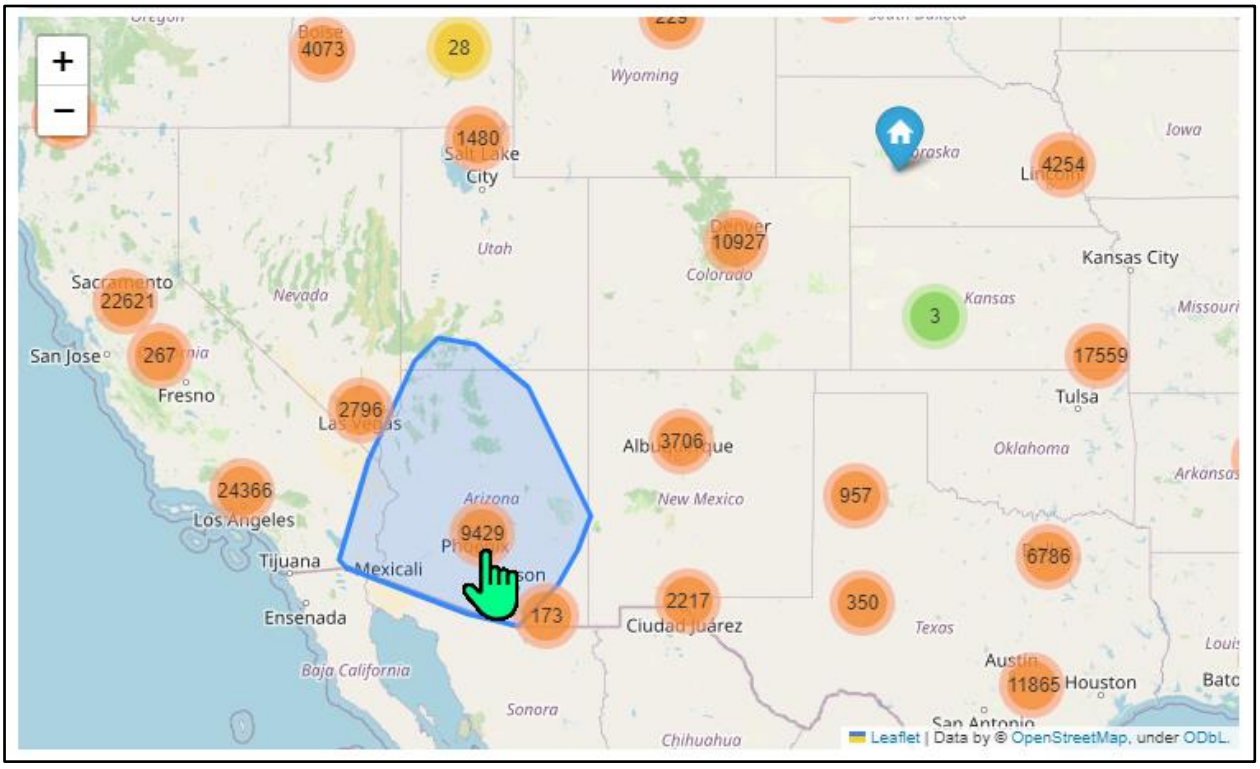

Рисунок Г.5 – Інтерактивна мапа розташування автомобілів

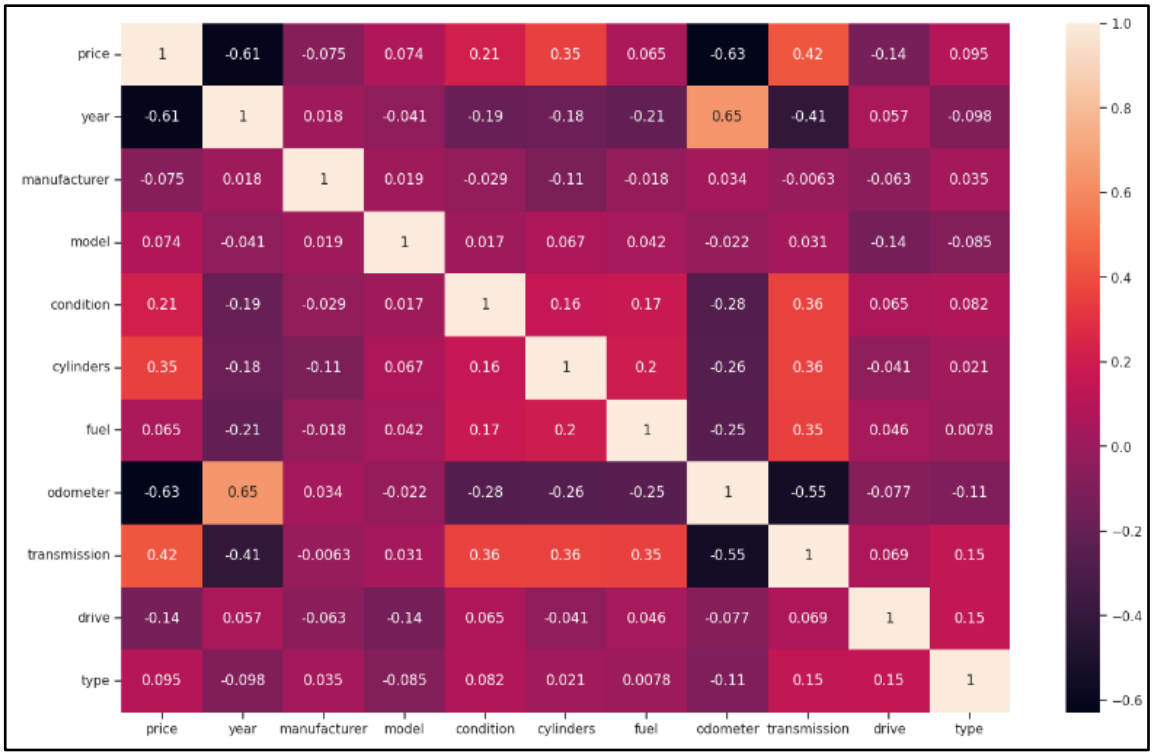

Рисунок Г.6 – Кореляційна матриця ознак

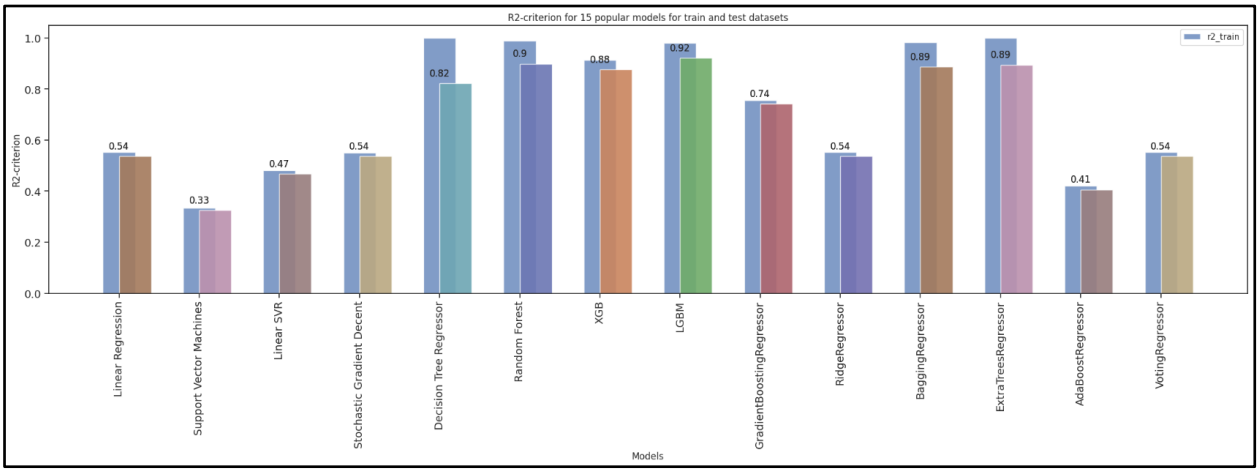

Рисунок Г.7 – Графік точності моделей за коефіцієнтом детермінації

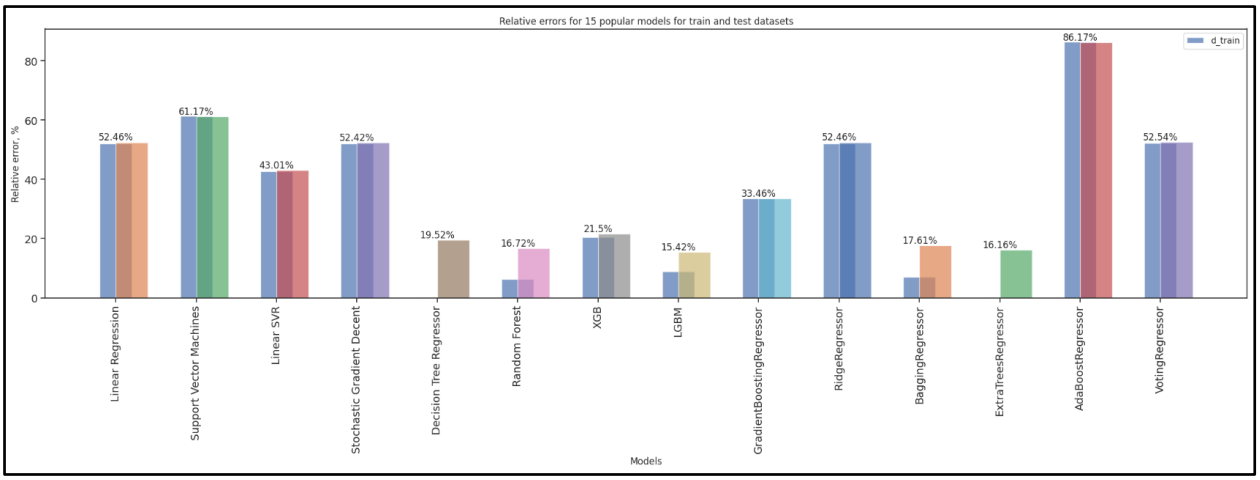

Рисунок Г.8 – Графік відносної похибки моделей

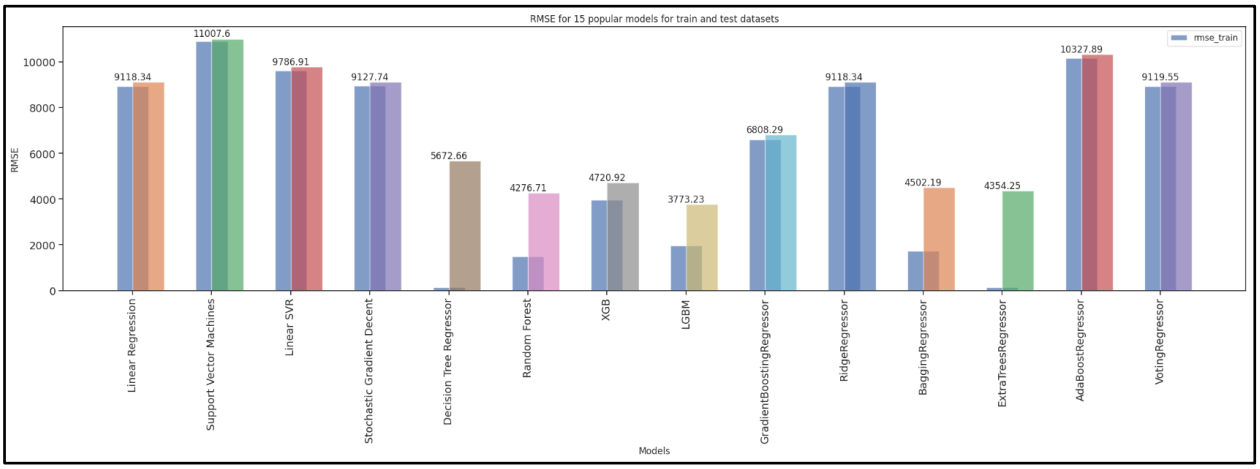

Рисунок Г.9 – Графік середньоквадратичної похибки моделей

	Real_price	predicted_prices	difference
27209	35590	35672	82
26582	44990	45198	208
36067	12850	10868	1982
39664	11900	10613	1287
12981	27590	27493	97
14354	12995	14672	1677
23865	13901	13436	465
40202	16995	16727	268
32380	79990	78692	1298
30042	7499	6681	818

Рисунок Г.10 – Результат передбачення ціни та різниця із справжньою

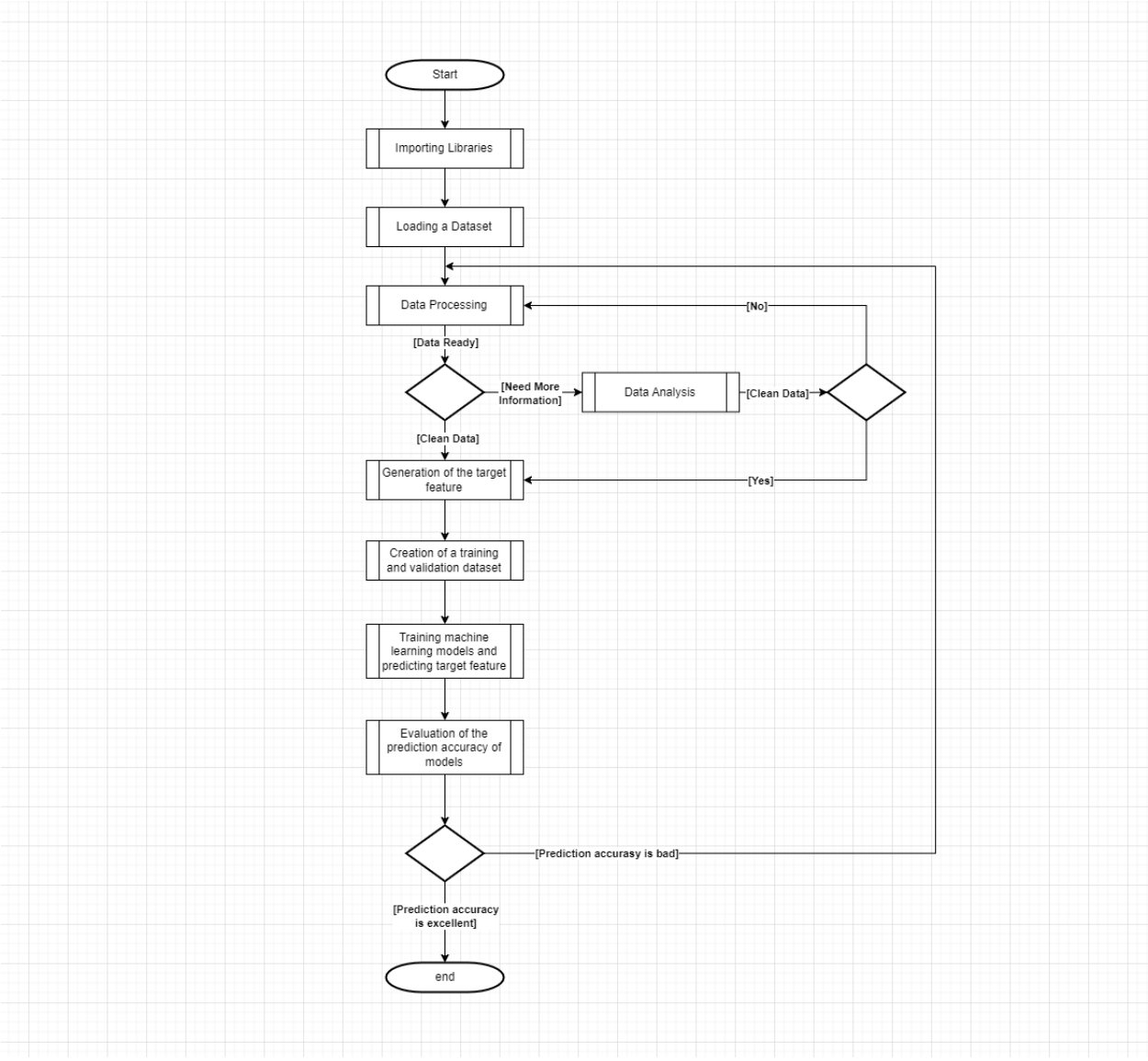

Рисунок Г.11 – Алгоритм роботи інформаційної технології