Вінницький національний технічний університет
Факультет комп'ютерних систем і автоматики
Кафедра системного аналізу, комп'ютерного моніторингу
та інженерної графіки

ІНФОРМАЦІЙНА СИСТЕМА РЕКОМЕНДУВАННЯ ЦІНИ ВЖИВАНОГО АВТО

Пояснювальна записка до магістерської кваліфікаційної роботи

Виконав: студент 2 курсу, групи ІСТ-18м
спеціальності 126 – «Інформаційні системи та технології»
Лосенко А.В.

Керівник: д.т.н., проф. Мокін В.Б.
Рецензент: к.т.н. доц. Маслій Р.В.

Вінниця ВНТУ – 2019 року

РЕФЕРАТ

Магістерська кваліфікаційна робота: 81 ст., 7 табл., 19 рис., 25 джерел.

Об’єкт досліджень – процес створення інформаційної системи рекомендування ціни вживаного авто.

Мета роботи – систематизувати сучасні методи розвідувального аналізу даних на Python, запропонувати технологію аналізу та передбачення ціни на вживані авто та перевірити її за даними зі США та України.

Здійснено порівняльний аналіз існуючих систем для рекомендування ціни вживаного авто в Україні. Здійснена систематизація методів розвідувального аналізу даних. Проведено порівняльний аналіз алгоритмів для вирішення задачі прогнозування ціни вживаного авто. Визначено оптимальні технології для реалізації інформаційної системи (її модулів розвідувального аналізу та прогнозування ціни автомобіля). Виконано порівнняння точності застосованих алгоритмів.

Прогнозні припущення про розвиток об’єкта дослідження – розробка інформаційної системи рекомендування ціни на вживане авто, яка на відміну від існуючих, здійснює миттєву рекомендацію ціни вживаного авто на основі даних, які введені користувачем.

Галузь застосування – веб-сервіси, що спеціалізуються на продажі вживаних автомобілів по всій Україні.

ІНФОРМАЦІЙНА СИСТЕМА РЕКОМЕНДУВАННЯ ЦІНИ, ІНТЕЛЕКТУАЛЬНА ТЕХНОЛОГІЯ, ПРОГНОЗУВАННЯ ЦІНИ, РОЗВІДУВАЛЬНИЙ АНАЛІЗ ДАНИХ, МОДЕЛІ МАШИННОГО НАВЧАННЯ.

ABSTRACT

Master's qualification work: 81 pages, 7 tables, 19 pictures, 25 sources.

The object of research – the process of creating an informational system for recommending a used car price.

The purpose of the work – to systematize modern methods of exploratory data analysis on Python, to offer the technology of analysis and price prediction for used cars and to check it according to the data from the USA and Ukraine.

Comparative analysis of existing systems for recommending used car prices in Ukraine is carried out. Systematization of methods of exploratory data analysis is carried out. A comparative analysis of algorithms for solving the problem of forecasting the price of a used car is carried out. Optimal technologies for implementation of the informational system (its modules of exploratory data analysis and forecasting of the car price) are determined. The accuracy of the applied algorithms is compared.

Estimated assumptions about the development of the object of study - the development of an informational system for forecasting of a used car price, which, unlike existing ones, implements an instant recommendation of the used car price based on the data entered by the user.

Scope of application – web services specializing in used car sales throughout Ukraine.

INFORMATIONAL SYSTEM FOR PRICE RECOMMENDATION, INTELLECTUAL TECHNOLOGY, PRICING FORECAST, EXPLORATORY DATA ANALYSIS, MACHINE LEARNING MODELS

ЗМІСТ

ВСТУП

Для успішного продажу вживаного автомобіля необхідно, перш за все, зібрати достовірну та актуальну інформацію про технічний стан автомобіля, умови його використання, та історію використання попередніми власниками. Дана інформація зазвичай міститься в багатьох установах одночасно: у банках даних, а також тематичних веб-порталах. Причому, ці дані потрібні як і продавцям, так і потенційним покупцям автомобілів.

Існує багато веб-порталів, що спеціалізуються на купівлі-продажу автомобілів, які володіють великим об’ємом таких даних. Зазвичай такі веб-портали діють в межах однієї країни, але також є і міжнародні ресурси. Свідчення купівлі або продажу того чи іншого автомобіля зазвичай зберігають в банку даних, який дає змогу агрегувати та аналізувати ці дані у майбутньому.

Одним з найбільших веб-сервісів такого виду в Україні є сайт Auto RIA: http://auto.ria.com. Крім власне самого сайту, на якому можна вибрати потрібний транспортний засіб шляхом зручного пошуку, даний веб-сервіс також надає доступ до власного прикладного програмного інтерфейсу (Application Programming Interface, API). За допомогою даного ресурсу можна отримати доступ до деяких історичних даних попередніх угод купівлі/продажу.

Даний портал є лише прикладом одного з багатьох, оскільки ринок продажу вживаних авто в Україні має тенденцію до зростання: на січень-вересень 2019 обсяг продажу зріс в 4,5 рази, в порівнянні з аналогічним періодом минулого року, й становить 321,6 тисячі авто [1].

Враховуючи шо багато таких ресурсів мають на меті приваблення аудиторії та полегшення користувацького досвіду, часто постає питання оптимізації тих чи інших процесів роботи системи, користуючись вже отриманими раніше даними. Це можуть бути зміни в інтерфейсі системи, функціональні зміни роботи самого додатку або зміни у наданні послуг підтримки для користувача.

Серед недоліків, які були виявлені впродовж аналізу даних ресурсів було виявлено, що користувацький досвід не достатньо інтерактивний, вимагає від користувача введення даних, що знаходять у відкритому доступі, або можуть бути отримані при використанні ресурсів. Введення цієї інформації збільшує час публікації оголошення про продаж автомобіля, шо може у свою чергу призвести до переходу користувача до іншого веб-порталу, де даний процес налаштований кращим чином.

Одним з важливих кроків при реєстрації автомобіля на продаж є введення його ціни, зазвичай цю суму користувач повинен формулювати власноруч, спираючись на власний попередній досвід. Але якщо такого попереднього досвіду у нього немає, ціна може бути сформульована не вигідно для власника (або занадто висока або занадто низька відносно ринку). Потрібно врахувати, що є багато чинників, що впливають на ціноутворення вживаного авто: його загальний стан, умови використання, навіть регіон де він автомобіль знаходився під час свого використання.

Для допомоги з формуванням ціни на автомобіль доцільно створити спеціалізовану систему рекомендування ціни автомобіля, з використанням попередньо введеної інформації. Однак, для забезпечення ефективності її функціонування, спершу потрібно побудувати її інформаційну модель.

Отже, розробка інформаційної системи рекомендування ціни вживаного авто, яка буде забезпечувати формування максимально актуальної ціни автомобіля, є актуальною.

Об’єктом дослідження є процес створення інформаційної системи рекомендування ціни вживаного авто. **Предметом дослідження** є інформаційна система рекомендування ціни вживаного авто. **Мета дослідження:** систематизувати сучасні методи розвідувального аналізу даних на Python, запропонувати технологію аналізу та передбачення ціни на вживані авто та перевірити її за даними зі США та України.

Наукова новизна одержаних результатів. Дістала подальший розвиток інтелектуальна технологія аналізу та передбачення ціни на вживані авто, за рахунок удосконалення параметрів фільтрів, вибраних під час

розвідувального аналізу даних, та підходу щодо вибору оптимальної моделі із багатьох, отриманих у т.ч. з оптимізацією гіперпараметрів, що дозволило підвищити точність передбачення ціни автомобілів за їх параметрами.

Практичне значення одержаних результатів полягає у наступному:

- реалізований програмний модуль розвідувального аналізу та визначення ключових ознак;
- реалізований програмний модуль рекомендування ціни вживаного авто.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки задач, коректним застосуванням математичних методів під час доведення наукових положень, строгим виведенням аналітичних співвідношень, порівнянням результатів з відомими, та збіжністю результатів математичного моделювання з результатами, що отримані під час впровадження розробленого програмного засобу.

Публікації. За результатами магістерської кваліфікаційної роботи опубліковано: 1 стаття в науковому журналі «Вісник Вінницького політехнічного інституту».

1 АНАЛІЗ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА СИСТЕМ РЕКОМЕНДУВАННЯ ЦІНИ

1.1 Аналіз предметної області

Ринок інтернет-торгівлі розширюється з кожним роком, це відбувається за рахунок створення різноманітних онлайн-платформ, що спеціалізуються на продажах товарів. Такі платформи обираються користувачами через свою зручність, оскільки вони дають доступ до чисельних каталогів різних товарів, від побутової техніки до нерухомості.

У зв’язку з розширенням ринку зростає і частка продажів у мережі Інтернет. Середній показник обсягу продажів онлайн у 2019 році зріс на 20,7% в порівнянні з попереднім роком, причому найбільший ріст спостерігається у країнах, що розвивається [2]. В свою чергу, за прогнозами експертів, такий показник в Україні буде становити 25% [3], що є досить значним прогнозом. Це свідчить про зростання попиту на створення нових платформ для продажів товарів, а також покращення вже існуючих.

Найбільшими компаніями цього напрямку в Україні є компанії Rozetka, Prom.ua та OLX. Кожна з них має в основі свого бізнесу веб-портал, який в тій чи іншій формі дозволяє реалізовувати товари різних типів. Ці платформи надають продавцям зручний спосіб розповсюдження своїх товарів, а покупцям багатий набір інструментів для пошуку підходящого товару. Розробники таких платформ ставлять за мету формування якнайкращого користувацького досвіду, починаючи від швидкого завантаження веб-сторінок сайту, закінчуючи розміщенням фільтрів товарів за своєю актуальністю. Кожне нововведення в користувацький досвід системи це зростання продажів та збільшення числа активних користувачів, що безумовно є запорукою розвитку бізнесу.

Потенційний покупець, під час покупки у мережі Інтернет, проводить самостійне порівняння отриманих варіантів товару. Часто при виборі товару покупець не володіє достатнім обсягом інформації, щоб орієнтуватись у

особливостях його ціноутворення, а тому може прийняти не вигідне для себе рішення, у випадку якщо натрапить на занадто дешевий або занадто дорогий товар, що був не чесно оцінений продавцем. Схожий ризик може спіткати і продавців також, тому що вони можуть помилково сформувати ціну власних товарів, керуючись лише особистим досвідом, що призведе до збитків для бізнесу.

Такий ризик особливо явний на вторинному ринку, коли товар зазвичай вже був у вжитку, і його ціна може варіюватись. В цьому випадку ше більш важливим є застосування докладного аналізу товарних пропозицій тієї чи іншої категорії, з наступним прогнозуванням актуальної ціни на основі характеристик товару. З використанням такого методу покупець може отримати додаткові відомості про ціноутворення пропозиції, що його цікавить, а покупець, в свою чергу, пересвідчитись чи є запропонована ним ціна прийнятною.

Для перевірки доцільності впровадження методу аналізу ознак товару задля прогнозування актуальної ціни на нього, було вирішено впровадити такий метод у сфері продажів вживаних авто, оскільки при оцінці автомобіля враховується велика кількість ознак, кожна з яких значним чином варіюється, що відчутно впливає на вартість автомобіля на вторинному ринку.

1.2 Огляд існуючих програмних продуктів

Порівняємо, яким чином реалізовані системи рекомендування ціни на вживані авто на торгових майданчиках з продажу вживаних автомобілів. На українському автомобільному ринку діє декілька веб-платформ з продажу вживаних автомобілів, але найбільшими серед них є два сайти: AUTO.RIA та RST.UA. Кожен з цих сайтів в той чи інший спосіб надає інформацію про рекомендовану ціну автомобіля.

Наприклад, на сайті AUTO.RIA створений окремий прикладний програмний інтерфейс (application programming interface, API) для використання сторонніми розробниками, в якому є можливість робити

пошукові запити до бази даних сайту. Також даний API надає можливість запиту середньої ціни вживаного авто за його параметрами (виробник, модель, тип палива, тип трансмісії, і т.д.)[4]. Результатом запиту є середнє значення ціни автомобіля, отримане керуючись цінами автомобілів з схожими ознаками, що були до цього розміщені на сайті. Приклад результату запиту наведений на рисунку 1.1.


```
{
    total: 12,
    arithmeticMean: 16305.882352941177,
    interQuartileMean: 8483.333333333334,
    percentiles: {
        1.0: 1944,
        5.0: 2520,
        25.0: 3500,
        50.0: 8000,
        75.0: 23500,
        95.0: 53539.999999999985,
        99.0: 64868
    },
    prices: [
        67700,
        27000,
        3000,
        23500,
        3500,
        8100,
        10000,
        11000,
        45800,
        50000,
        1800,
        4350,
    ],
    classifieds: [
        14663610,
        14226353,
        14138132,
        13969588,
        14697569,
        13386778,
        14059841,
        14290096,
        14890250,
        14159441,
        14759141,
        13559841,
    ]
}
```


Рисунок 1.1 – Приклад результату запиту до прикладного програмного інтерфейсу сайту AUTO.RIA

Як бачимо з рисунку, сервіс AUTO.RIA надає деякі статистичні дані (загальну кількість оголошень, медіани, квантилі вибірки). Середня ціна

формується суто за рахунок вже наявних раніше записів у базі даних, та, власне, їх кількості. Такий підхід може суттєво вплинути на остаточний результат, оскільки така залежність ставить під сумнів достовірність ціни, наприклад, коли відповідних записів у базі вкрай мало, або вони відсутні.

Також слід зазначити, що до недавнього часу, сайт RST.UA додавав до кожного розміщеного оголошення коментар біля ціни авто, вказуючи чи вона завищена або занижена, відносно моделей даного автомобіля з схожими ознаками. Скоріше за все, даний коментар формувався за рахунок схожого запису до бази даних сервісу, як це зроблено у AUTO.RIA.

Наявність такого функціоналу (або спроби його впровадження) вказують на актуальність вирішення задачі прогнозування ціни автомобіля.

Враховуючи такі обставини, для вирішення такої задачі слід виявити, в чому виражені закономірності утворення ціни на вживаний автомобіль, які його ознаки більш значущі відносно інших. Для виявлення таких закономірностей та ознак варто скористатись методами машинного навчання та обробки даних.

Використання методів машинного навчання дозволить отримати модель, що може прогнозувати ціну вживаного автомобіля, користуючись закономірності, шо будуть виявлені впродовж процесу її “тренування” за допомогою тренувального набору даних, враховуючи що дані були оброблені належним чином.

1.3 Висновки

В даному розділі був зроблений короткий огляд відомостей про ринок онлайн-продажів в Україні та світі, зазначені основні показники росту ринку.

Проведено порівняння сервісів, що реалізували тим чи іншим чином рекомендування ціни вживаного авто і зроблено висновок, що дані сервіси у своїх рекомендаціях покладаються лише на наявність схожих записів у базі даних, тому було вирішено реалізувати модуль рекомендування ціни вживаного авто з використанням моделей машинного навчання, з метою

отримання універсального програмного рішення.

2 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ РЕКОМЕНДУВАННЯ ЦІНИ ВЖИВАНОГО АВТО

2.1 Ідея програмної реалізації інформаційної системи рекомендування ціни вживаного авто

Завдання, поставлене у попередньому розділі потребує більш ширшого пояснення, які саме методи машинного навчання будуть використані, а також який загальний вигляд матиме майбутній програмний продукт. Для цього, спочатку, потрібно пояснити основні терміни, що входять, в поняття машинного навчання.

Машинне навчання, є одним з розділів теорії штучного інтелекту (на рівні з машинним зором та технологією обробки природної мови), що застосовує різноманітні методи статистики, а також алгоритми для аналізу даних, які дозволяють покращувати продуктивність комп'ютеру у вирішенні поставленої задачі. Хоча дана галузь виникла ще в в 50-х роках минулого століття, лише у найближчі роки вона опинилась на піку популярності. Це пояснюється накопиченням великої кількості даних різними компаніями, та бажанням аналізувати ці дані для автоматизації робочих процесів, розуміння тенденцій розвитку бізнесу, або задля допомоги в прийнятті рішень [5].

Набір даних, що аналізується методами машинного навчання, є нічим іншим як списком об'єктів, кожен з яких має однаковий набір характеристик з певними їх значеннями. Характеристики (або ознаки) цих об'єктів можуть бути числами, рядком тексту, або складовими деякої множини. Для ефективного застосування методів машинного навчання об'єкти повинні бути представлені за допомогою зрозумілих для комп'ютеру типів даних (зазвичай чисел). Кожен набір таких формалізованих ознак представляється у вигляді числового вектору. Такий підхід спрощує аналіз даних, оскільки в основі багатьох алгоритмів машинного навчання лежать операції лінійної алгебри, які оперують саме векторами чисел.

Слід також пояснити, яким чином відбувається "навчання" моделі

машинного навчання. Для виявлення закономірностей між ознаками об’єкту та наступної перевірки точності результатів, вихідний масив даних розділяють на дві частини: на тренувальну та тестову вибірку даних. Тренувальна вибірка є набором прикладів, які використовуються для формування закономірностей, що стануть основою прогнозу. Після обробки тренувальної вибірки оцінюється точність прогнозу, використовуючи об’єкти тестової вибірки: прогноз порівнюється з дійсною величиною прогнозованої ознаки з тестового набору даних. Перевірка точності здійснюється за допомогою декількох критеріїв для підтвердження результатів моделі.

Зрештою, для формування припущення на основі даних, застосовується та чи інша модель машинного навчання. Зазвичай моделлю є функція, яка на вхід приймає вектор ознак об’єкту, а на виході видає значення прогнозу. На етапі “тренування” моделі, у неї поступово вводяться всі дані з тренувальної вибірки, після чого відбувається тестування точності прогнозу. Керуючись результатами тестів дослідники приймають рішення, чи задовольняють вони умови поставленої задачі, чи потрібно внести зміни у вхідні дані, або параметри моделі [6].

Поставлена задача з прогнозування ціни вживаного автомобіля, являє собою пошук залежності між ознаками деякого автомобіля та його ціни. Пошук такої залежності є ціллю задачі регресії, тому для вирішення поставленої задачі скористаємося кількома відомими методами регресії для формування відповідних моделей машинного навчання.

Слід зауважити, що розв’язанню поставленої задачі повинні передувати етапи очищення даних та розвідувальний аналіз даних. Ці етапи надзвичайно важливі у розробці подібної системи, оскільки від розгорнутого розвідувального аналізу даних безпосередньо залежить точність передбачень моделі.

Але перш за все потрібно почати з вибору технології для розробки інформаційної системи рекомендування ціни вживаного авто.

2.2 Огляд та вибір технологій для розробки інформаційної системи рекомендування ціни вживаного авто

Після формування задачі та методів її вирішення варто зайнятись оглядом доступних технологій у сфері машинного навчання, після чого обрати серед них найбільш ефективне та зручне програмне рішення. Крім власне вирішення поставленої задачі, розроблене програмний модуль повинний бути портативним, щоб мати можливість легкого використання потенційними користувачами.

У сфері машинного навчання найбільш популярними є мови програмування Python, R та MATLAB. Наведемо коротку довідкову інформацію про кожну з цих технологій.

Python є інтерпретованою мовою програмування, що базується на принципах об’єктно-орієнтованого парадигми. Типи даних визначають динамічно, тобто лише під час виконання коду програми, при цьому також використовуються структури даних високого рівня. Також у Python багатий вибір різноманітних бібліотек, призначених не лише для наукових досліджень та обробки даних, а й веб-розробки, адміністрування серверів, та розробки додатків з графічним користувацьким інтерфейсом. Код Python зручним чином конструюється в програмні модулі, що легко розповсюджуються та інтегруються між собою [7]. Всі ці чинники роблять цю мову досить зручною для швидкої розробки як прототипів, так і повноцінних високонавантажених додатків.

У свою чергу, MATLAB є пропрієтарним пакетом прикладних програм, та мовою програмування, яка використовується в даному пакеті. Сфера застосування мови суто наукова, спеціалізується на чисельному обчислюванні, дозволяє працювати з математичними матрицями та різноманітними алгоритмами, вирішенні статистичних задач, а також візуалізацією функцій та даних. Дана мова програми є потужним інструментом для наукових досліджень, при цьому відповідний пакет програм включає в себе набір корисних бібліотек та інструментів, які не

потребують додаткового встановлення. Такий спосіб розповсюдження даного пакету робить його зручним для використання науковцями, оскільки він не потребує додаткових маніпуляцій при конфігурації та встановленні, тому дослідник має змогу відразу приступити до роботи [8].

Мова програмування R у своїй реалізації є дечим “середнім” між Python та MATLAB, вона також вузькоспеціалізована для задач статистичних обрахунків та аналізу даних, має графічне середовище розробки, але також з її використанням реалізовано широкий спектр сторонніх бібліотек, що легко інтегруються у програмні продукти. R також підтримує різноманітні парадигми програмування: об’єктно-орієнтовану, функціональну та процедурну. Ця мова стає дедалі популярнішою в колах дослідників даних та розробників, що дає надії на її подальний розвиток та розповсюдження[9].

Враховуючи універсальність мови, велике міжнародне співтовариство, та наявність потужних бібліотек для обробки даних, для розробки інформаційної системи рекомендування ціни вживаного авто було вирішено скористатись мовою програмування Python.

Після вибору мови програмування залишилось лише обрати середовище розробки. Оскільки поставлене завдання потребує дослідження даних, застосування моделей машинного навчання та візуалізації проміжних результатів було вирішено обрати середовище розробки Jupyter Notebook. Дане середовище розробки являє собою веб-інтерфейс, поєднує в собі текстовий редактор та інтерпретатор коду, дозволяє зручно описувати хід дослідження та виконувати код у ізольованому середовищі [10]. Такий формат представлення проектів наочно зображує процес дослідження та є зручним для співпраці дослідників між собою.

Визначившись з технологіями розробки інформаційної системи рекомендування ціни вживаного авто перейдемо до пошуку набору даних для використання розробленим програмним модулем.

2.3 Вибір набору даних для тренування інформаційної системи рекомендування ціни вживаного авто

Для найбільш точного результату прогнозування ціни вживаного авто слід перш за все провести детальний пошук серед уже існуючих наборів даних, що розповсюджуються публічно. Одним з найбільших ресурсів що надає доступ до таких наборів даних є веб-платформа kaggle.com. Одним з розділів сайту є каталог наборів даних, що були розміщені іншими користувачами сервісу. В ньому міститься зручне поле пошуку, у якому можна формувати запити за безпосередньою назвою набору, темою що цікавить, а також форматом файлу з набором даних. Вигляд інтерфейсу форми пошуку наведений на рисунку 2.1.

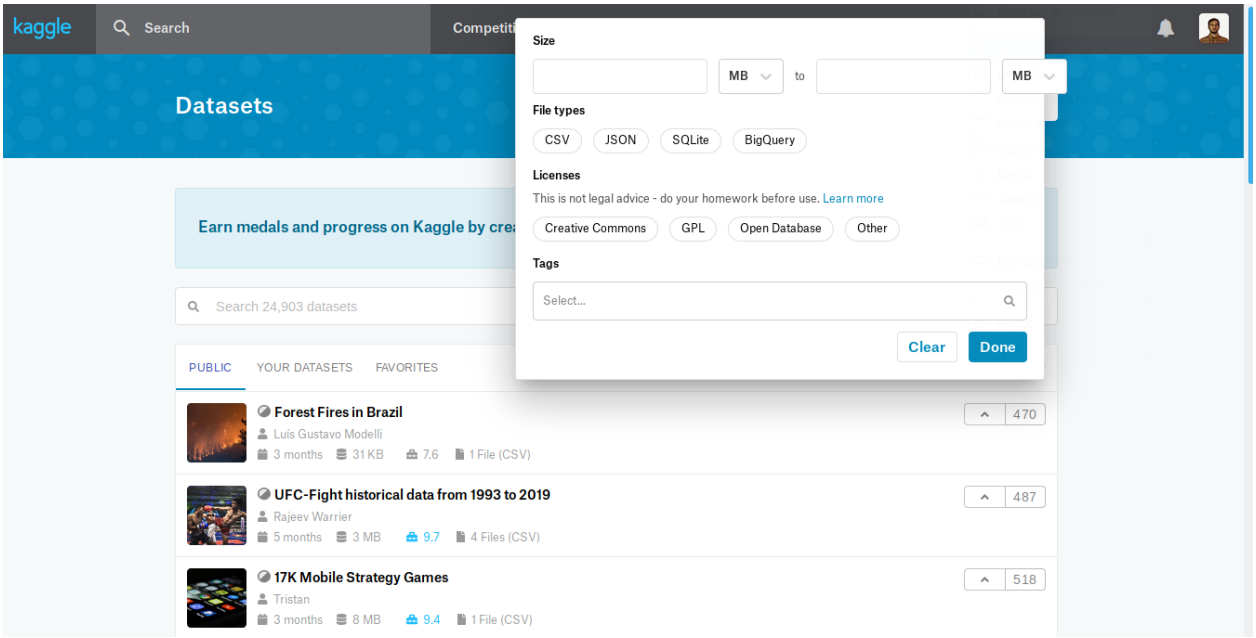


Рисунок 2.1 – Інтерфейс пошуку наборів даних на сайті kaggle.com

Скориставшись даною формою пошуку був знайдений набір даних, зібраний з розділу продажу автомобілів американського сайту craigslist.com. Знайдений набір даних вміщує в собі інформацію про 525839 оголошення продажу автомобілів, що були розміщені на сайті[11]. Розповсюджується

дана інформація за ліцензією «CC0: Public Domain», тобто без обмежень на копіювання і використання.

Переглянемо знайдений масив даних та оцінимо доцільність його застосування в розробці інформаційної системи.

Кожен об’єкт даної вибірки містить наступні ознаки:

- веб-посилання на оголошення про автомобіль на сайті(“url”);
- місто, де розміщене оголошення (“city”);
- посилання на сторінку сайту з фільтром по місту (“city_url”);
- ціна автомобіля (“price”);
- рік випуску (“year”);
- виробник автомобіля (“manufacturer”);
- модель автомобіля (“make”);
- стан автомобіля (“condition”)
- кількість циліндрів (“cylinders”);
- вид палива (“fuel”);
- пробіг автомобіля (“odometer”);
- статус заголовку оголошення (“titile_status”);
- трансмісія (“transmission”);
- номер VIN (“VIN”);
- привід автомобіля (“drive”);
- розмір оголошення (“size”);
- тип кузову (“type”);
- колір автомобіля (“paint_color”);
- посилання на фото (“image_url”);
- опис оголошення (“desc”);
- широта (“lat”);
- довгота (“long”).

Хоча серед ознак містяться ознаки що не належать до безпосередніх характеристик автомобіля (посилання на оголошення, місто, де розміщене оголошення, довгота, широта, опис оголошення, і т.д.), але разом з тим є достатньо ознак, що детально характеризують будь-який автомобіль (пробіг,

тип кузову, тип палива, привід, рік випуску, колір автомобіля, трансмісія, кількість циліндрів). Ці ознаки або вже представлені у вигляді чисел, або можуть бути представлені у вигляді чисел, тому що здебільшого являються елементом множини можливих варіантів, що також спрощує їх використання моделлю машинного навчання.

Також для дослідження тенденцій ціноутворення в Україні були використані дані веб-системи медіа-корпорації “RIA” по Україні (набір даних становить 5432 автомобілях), які були оброблені в межах договору про науково-технічне співробітництво між компанією "RIA" та ВНТУ. Обраний масив даних містить такі ознаки:

- ціна автомобіля (“price”);
- рік випуску (“year”);
- виробник автомобіля (“manufacturer”);
- модель автомобіля (“make”);
- вид палива (“fuel”);
- пробіг автомобіля (“odometer”);
- трансмісія (“transmission”);
- привід автомобіля (“drive”);
- тип кузову (“type”).

Як бачимо, ознаки об’єктів датасету включають лише характеристики автомобіля, чого більш ніж достатньо для використання у дослідженні. Враховуючи розміри масивів даних, та наявність у них характерних ознак, варто їх у розробці інформаційної системи рекомендування ціни вживаного авто.

2.4 Розробка модулю розвідувального аналізу та визначення ключових ознак вибірки даних

Як було зазначено вище, розв’язанню поставленої задачі на Python повинні передувати етапи очищення даних та розвідувальний аналіз даних. Для задач обробки та аналізу даних прийнято використовувати бібліотеку

мови Python під назвою Pandas. Принцип її роботи полягає в інтерпретації масивів даних у вигляді об’єкту під назвою датафрейм [10].

Такий датафрейм являє собою багатовимірний масив, що може в собі містити дані різного типу: цілі числа, числа з плаваючою комою, а також рядки тексту. Кожен такий датафрейм має доступ до різноманітних методів, що дають змогу отримати більше відомостей з масиву даних, що досліджується.

Проведено систематизацію сучасних підходів до автоматизації оброблення даних на цих етапах. Як правило, для аналізу дата фреймів використовуються такі підходи, методи та технології:

1) базова статистика за допомогою методу Describe: інформація по кожній ознаці, наприклад, кількість значень, мінімальне, максимальне, середнє і середнє квадратичне значення та значення квантилів, які можна задавати яким завгодно списком (за замовчуванням: 25%, 50%, 75%) [12];
2) метод ProfileReport, що входить у склад бібліотеки pandas-profiling, який автоматично виконує більшість типових операцій аналізу даних, з яких починається вивчення датасету [13]:
   – наводить загальну статистику: кількість ознак (стовпців) загалом і по кожній ознаці зокрема; кількість спостережень (кількість рядків); кількість та відсоток пропущених даних; кількість дублікатів; обсяг пам’яті, яку займає датасет і кожен його рядок в середньому; типи даних ознаки;
   – наводить статистику по кожній ознаці окремо (у структурованій гіпертекстовій форми): кількість унікальних значень, пропущених, середнє, мінімальне, максимальне, сума, кількість нульових, квантилі (5%, 25%, 50%, 75%, 95%), середньоквадратичне відхилення, дисперсія, коефіцієнт ексцесу, коефіцієнт асиметрії, графік гістограми, 10 найбільш частих значень, 5 найбільших і 5 найменших значень тощо;
   – будує і візуалізує кореляційні матриці з використанням як

відомих методів Пірсона, Спірмена і Кендалла, так і метода Крамера для категоріальних ознак та найновішого метода ϕk , запропонованого у 2018 р. у роботі [14], для аналізу кореляції значень різного типу (числових, категоріальних та ін.), між якими можуть бути як лінійні, так і нелінійні залежності;

- будує і наводить статистику по пропущених даних у масиві даних у вигляді гістограми, матриці, теплової карти та дендрограми;
- наводить 10 перших і 10 останніх рядків датасету.

3) бібліотеки Matplotlib та Plotly дозволяють побудувати графіки різних видів для відображення різних особливостей датасету в цілому та окремих його ознак зокрема, переважно двовимірних, хоча за допомогою бібліотеки Mpl_toolkits.mplot3d («MPL» – це скорочення від «MatPlotLib») дозволяє будувати й тривимірні графіки [15];
4) бібліотека Seaborn дозволяє будувати різні графіки для аналізу статистичних особливостей даних, наприклад графік для вивчення особливостей взаємозв'язку двох показників, коли по одній осі відкладається одна гістограма, по іншій – інша, а між ними двовимірна функція їх взаємного розподілу;
5) бібліотека Sklearn дозволяє здійснювати інтелектуальний аналіз та очищення і доповнення даних, наприклад, масштабування і стандартизацію даних, їх імпутинг (інтерполяцію за різними методами по сусідніх даних); побудову різних моделей штучного інтелекту та, за ними – діаграм важливості ознак (як правило, на основі дерев рішень та регресійних моделей), що потім дозволяє з них відібрати найважливіші; кластеризацію та класифікацію даних за різними критеріями тощо;
6) бібліотека Scipy.stats містить багато статистичних функцій, у т.ч. закони розподілу та їх аналіз за -критерієм і критерієм Стьюдента, кореляційний, регресійний, дисперсійний і факторний аналіз, перетворення Бокса-Кокса для перетворення заданого закону розподілу

на нормальний та ін;

7) ще більше можливостей дають інтерактивні динамічні технології: інтерактивні графіки за допомогою методів бібліотеки Bokeh;
8) кластеризація даних та виявлення їх прихованих закономірностей, тощо, за допомогою інтерактивного сервісу https://projector.tensorflow.org;
9) базові бібліотеки Python, наприклад Pandas та NumPy, для роботи з основними типами даних у подібних задачах, теж мають ряд методів для виявлення пропущених, помилкових даних аналізу їх типів, статистичних даних та їх виправлення за певними алгоритмами;
10) інші бібліотеки від різних розробників, у т.ч. Microsoft (наприклад, lightgbm для побудови дерев рішень методом бустингу та аналізу важливості ознак у них), теж мають багато потужних можливостей, які часто перевищують можливості наведених вище технологій і методів;
11) у разі, якщо точність передбачення вийшла низькою, тоді можна збільшити кількість ознак, за рахунок їх генераторів за допомогою бібліотек Feature Tools (додає статистичні показники до кожної ознаки з відповідним агрегування по середньому, дисперсії, мінімуму та ін. та/або AutoML (застосовує різні математичні функції до значень ознак: піднесення у степінь, логарифм, тригонометричні функції та ін.), а потім будується діаграма важливості і мало важливі ознаки відкидаються [16].

В даній задачі можна використовувати усі ці технології, але, враховуючи поставлену мету варто скористатись наступним підходом до аналізу та передбачення ціни на вживані авто на основі Python:

1) провести попередню очистку даних, відкинути ознаки які не є суттєвими для оцінки вартості автомобіля, після чого видалити об’єкти які мають хоча б одну порожню ознаку;
2) здійснити перетворення текстових значень об’єктів у числові за допомогою кодування категоріальних даних (label encoding);

3) пройти ознайомлювальний етап EDA використанням pandas-profiling, в якому, в першу чергу, проаналізувати кількість пропущених даних, найбільші і найменші значення, гістограму значень та кореляційну матрицю – на основі цього, розробити правила для фільтрування помилкових, пропущених та аномальних даних;
4) побудувати дво- і тривимірні графіки, які дозволять уточнити по яких саме ознаках варто робити фільтрування аномальних даних;
5) побудувати базову статистику та оцінити різні значення квантилів: 5%, 10%, 25%, 50%, 75%, 90%, 95% – відібрати такі значення межі фільтру, за яких основна кількість даних залишиться в основній вибірці, але аномальні значення (які суттєво відрізняються від інших), будуть видалені.

Тепер детальніше опишемо кожен з кроків цієї процедури аналізу даних. Почнемо з очистки даних, для цього оберемо ознаки об’єкту, які не є суттєвими у оцінці вартості автомобіля. Такими ознаками є всі ті, що не відносяться до безпосередніх характеристик автомобіля, а саме:

– веб-посилання на оголошення про автомобіль на сайті(“url”);
– місто, де розміщене оголошення (“city”);
– посилання на сторінку сайту з фільтром по місту (“city_url”);
– статус заголовку оголошення (“titile_status”);
– номер VIN (“VIN”);
– посилання на фото (“image_url”);
– опис оголошення (“desc”);
– широта (“lat”);
– довгота (“long”).

Ці ознаки ніяким чином не характеризують автомобіль, а лише є додатковою інформацією про оголошення продажу цього автомобілю. Код реалізації видалення не актуальних ознак зображений на рисунку 2.2:

```
drop_columns = ['url', 'city', 'city_url', 'make', 'title_status', 'VIN', 'size', 'image_u
rl', 'desc', 'lat','long']
train0 = train0.drop(columns = drop_columns)
```

Рисунок 2.2 — Код реалізації видалення не актуальних ознак об’єкту даних

Наступним кроком, буде видалення даних, що мають хоча б одну ознаку з порожнім значенням. Нагадаємо, що обраний набір даних містить записи про 525839 автомобілів. Після очистки порожніх даних залишилось 161824 об’єктів, що все ще залишається достатньою кількістю використання цього набору, оскільки гарантує різноманіття значень при тренуванні моделі.

Далі здійснимо кодування текстових ознак у числові за допомогою використання класу LabelEncoder бібліотеки Scikit-Learn [17]. Це допоміжний клас трансформує значення ознак текстового типу (наприклад, колір автомобіля, привід, вид палива, тип трансмісій) у числові значення. Такими ознаками з текстовими значеннями також називають категоріальними. Фрагмент коду, що виконує цю трансформацію зображений на рисунку 2.3:

```
numerics = ['int8', 'int16', 'int32', 'int64', 'float16', 'float32', 'float64']
categorical_columns = []
features = train0.columns.values.tolist()
for col in features:
    if train0[col].dtype in numerics: continue
    categorical_columns.append(col)
# Encoding categorical features
for col in categorical_columns:
    if col in train0.columns:
        le = LabelEncoder()
        le.fit(list(train0[col].astype(str).values))
        train0[col] = le.transform(list(train0[col].astype(str).values))
```

Рисунок 2.3 — Фрагмент коду, що виконує трансформацію категоріальних даних у числові

Після виконаних дій слід приступити до безпосереднього розвідувального аналізу даних. Розвідувальний аналіз даних це набір методів, що застосовується при попередньому аналізі даних для виявлення загальних закономірностей, властивостей даних аналізу, а також розподілення значень у наборі даних. Для визначення аномальних даних побудуємо гістограми ознак датасету, для прикладу візьмемо ціну автомобіля. Графіки розподілення значень ціни автомобіля зображена на рисунку 2.4:

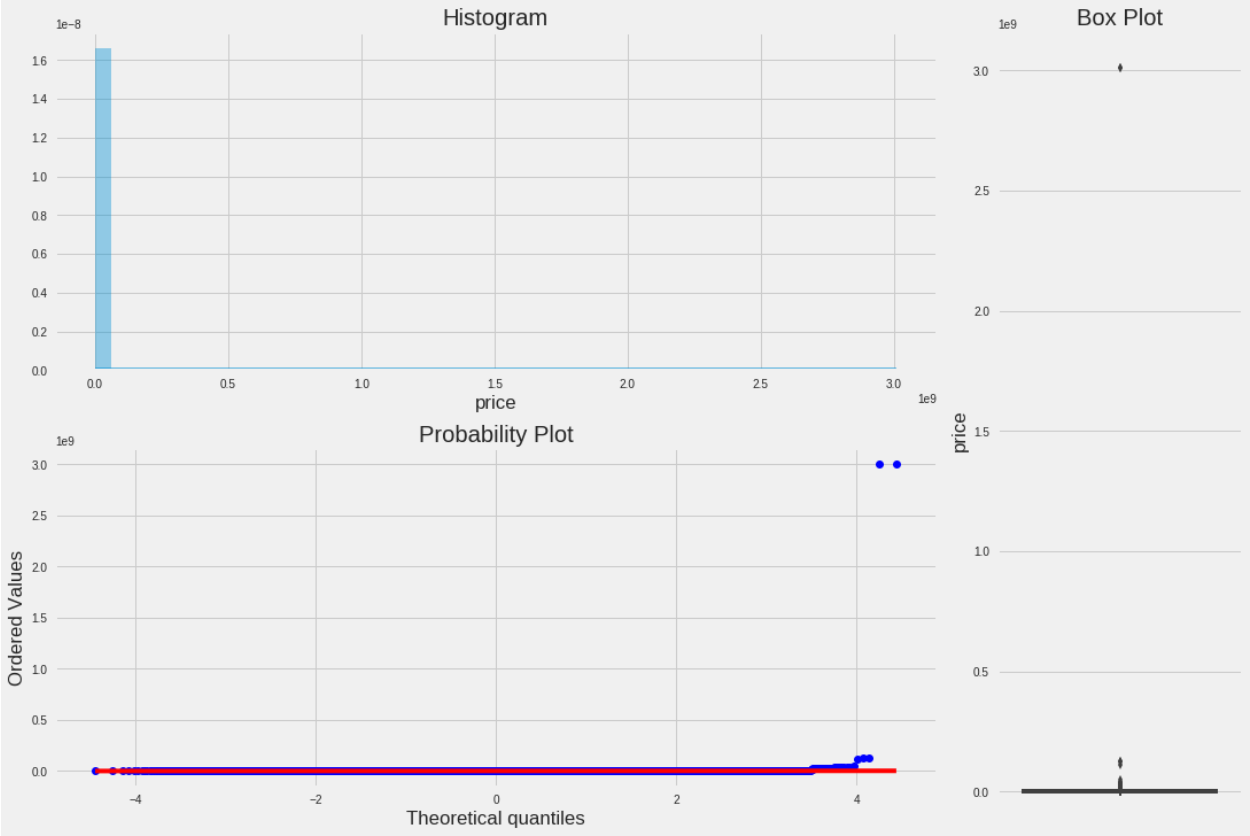


Рисунок 2.4 — Графіки розподілу ціни автомобіля

Побудовані графіки не є інформативними, оскільки простір значень ціни автомобіля містить велику кількістю аномальних даних. Продовжимо розвідувальний аналіз даних, та дізнаємося, які ще ознаки містять велику кількість аномальних даних, для цього побудуємо матрицю кореляції між ознаками набору даних. Для наочності, зобразимо дану матрицю у вигляді теплової карти (heatmap). Зображення матриці кореляції на рисунку 2.5:

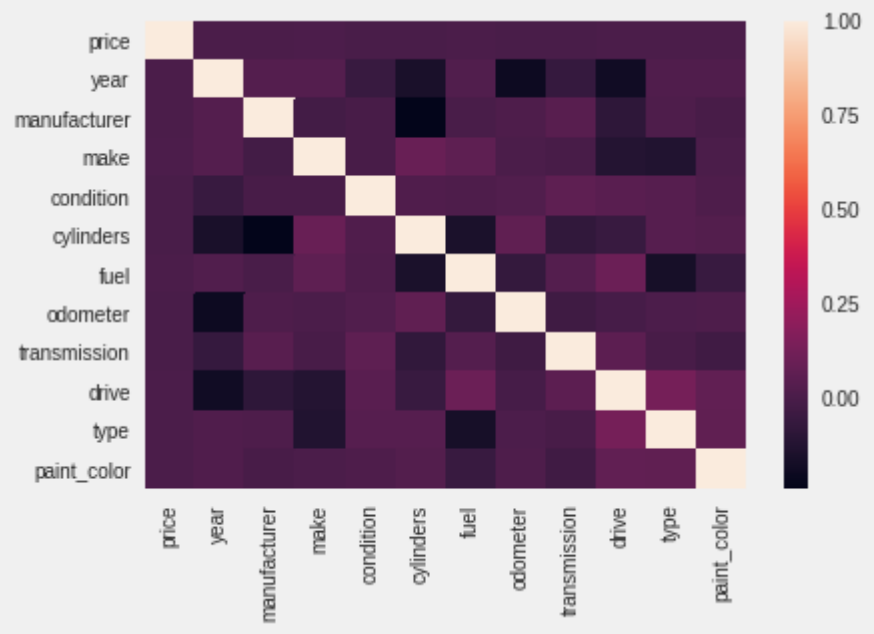


Рисунок 2.5 — Матриця кореляції між ознаками набору даних, зображена у вигляді теплової карти

Як бачимо з графіку вище, дані погано між собою корелюють, знову ж таки через велику кількістю аномальних даних. Зобразимо також відношення ціни, пробігу та року випуску автомобіля, таким чином ми дізнаємось, яким чином розподіляються дані у рамках цих трьох ознак. Графік цього відношення зображений на рисунку 2.6.

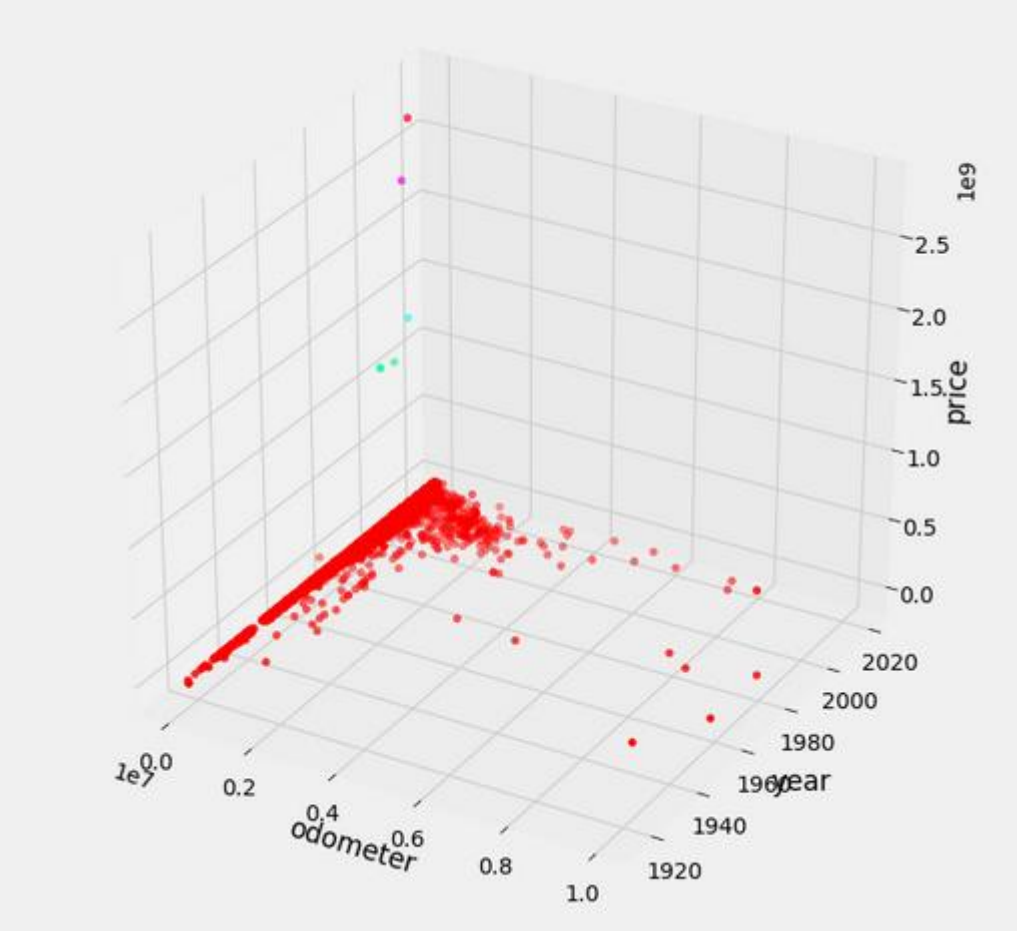


Рисунок 2.6 — Графік співвідношення значень ціни, пробігу та року випуску

Даний графік вказує на одну з можливих причин поганої кореляції – помилкові або аномальні значення і по пробігу (10 млн. км), і по вартості (2,5 млрд. дол. – явна помилка), і по року випуску (1920 р.), отже потрібно побудувати правило для фільтрування таких даних, оскільки, наприклад, ціна авто за 2015 рік з порівняно невеликим пробігом явно формується не так, як ціна на авто 1920 року чи з пробігом в 10 млн. км. Таким чином варто здійснити фільтрацію за аномальними значеннями пробігу, року випуску, а також ціною. Для цього побудуємо базову статистику набору даних, користуючись методом Describe (рис. 2.7).

	price	year	manufacturer	condition	cylinders	fuel	odometer	transmission
count	1.449030e+05	144903.0	144903.000000	144903.000000	144903.000000	144903.000000	1.449030e+05	144903.000000
mean	6.198665e+04	NaN	18.252176	1.078190	4.658013	1.886310	1.134756e+05	0.117375
std	1.120434e+07	NaN	10.939406	1.163766	1.280567	0.535248	1.235779e+05	0.386070
min	0.000000e+00	1900.0	0.000000	0.000000	0.000000	0.000000	0.000000e+00	0.000000
10%	1.750000e+03	2001.0	7.000000	0.000000	3.000000	2.000000	3.000000e+04	0.000000
50%	8.495000e+03	2010.0	14.000000	0.000000	5.000000	2.000000	1.070000e+05	0.000000
90%	2.520000e+04	2016.0	37.000000	3.000000	6.000000	2.000000	1.900000e+05	0.000000
max	3.009549e+09	2020.0	39.000000	5.000000	7.000000	4.000000	1.000000e+07	2.000000

Рисунок 2.7 — Базова статистика набору даних за допомогою методу Describe

Наведений графік вказує на те, як сильно відрізняється мінімальний рік випуску від року з квантилем 10%, максимальний пробіг від пробігу з квантилем 90%, мінімальна ціна від ціни з квантилем 10% та максимальна ціна від ціни з квантилем 90%. Це дозволило визначити правило для фільтрування аномальних значень, код реалізації якого зображений на рисунку 2.8:

```
#Filter: price (upper (90%) and lower (10%)), year (lower - 10%) and odometer (upper - 90%)
train = train[((train['price'] >= 1500)
               & (train['price'] < 25000)
               & (train['year'] >= 2001)
               & (train['odometer'] < 2000000))]
```

Рисунок 2.8 — Застосування фільтрів за ціною, пробігом, та роком випуску

Після фільтрування, масив даних зменшився до 115186 об’єктів. Тепер зобразимо графіки розподілу ціни автомобіля після фільтрування (рис 2.9).

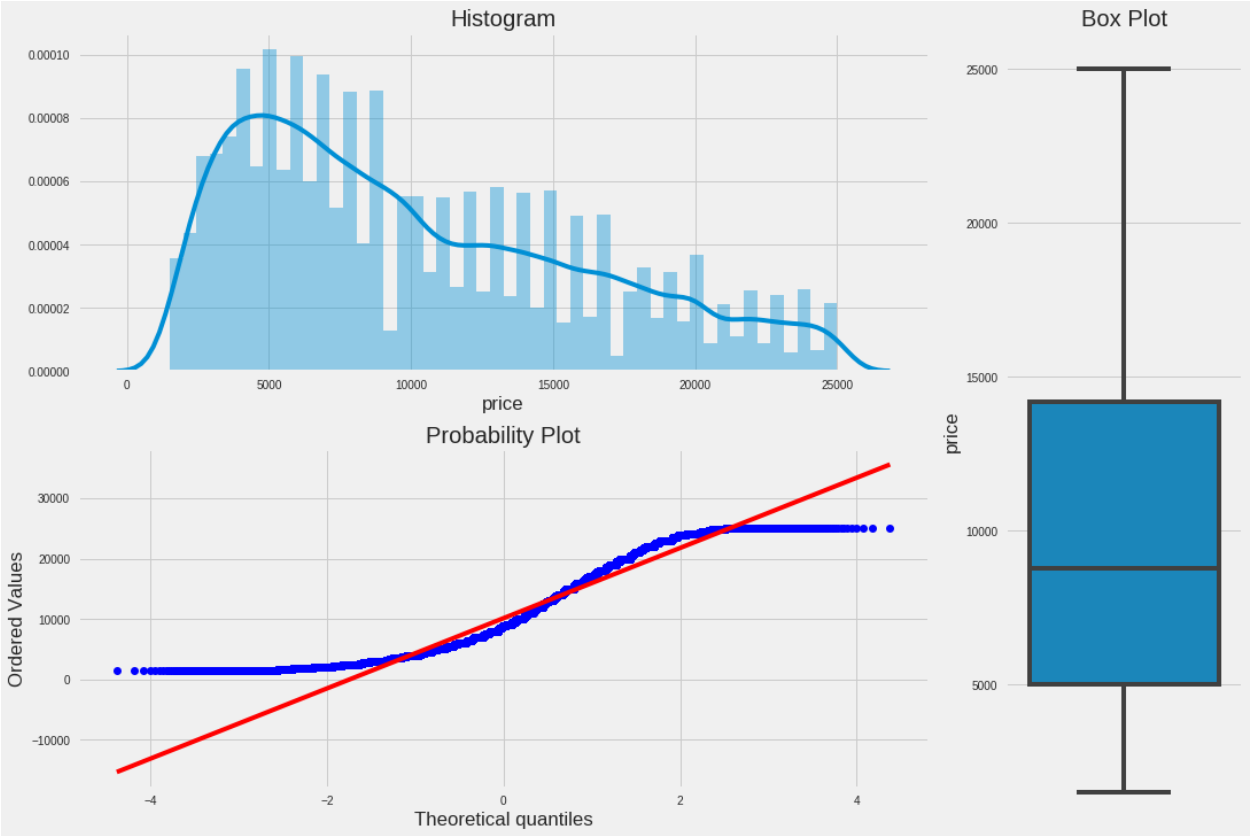


Рисунок 2.9 — Графіки розподілу ціни автомобіля після застосування фільтрів

Тепер ситуація помітно покращилась, ціна автомобіля розподілена в певних межах, які є реальними показниками ціни. Це безумовно повинно допомогти на етапі тренування моделей.

В якості додаткової перевірки зобразимо матрицю кореляції ще раз, використовуючи вже дані після фільтрації (рис 2.10):

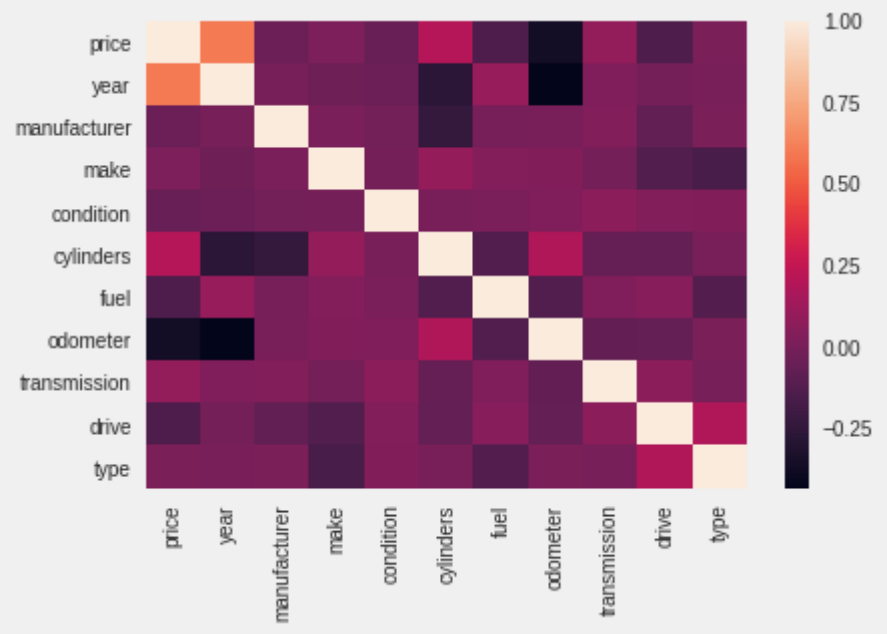


Рисунок 2.10 — Матриця кореляції між ознаками набору даних після фільтрування, зображена у вигляді теплової карти

Даний графік також надає покращені результати, прослідковується більша кореляція між ознаками набору даних, про це свідчить поява світліших областей на перетині значень ознак, у порівнянні з попереднім графіком. На сам кінець, зобразимо графік відношення ціни, року випуску автомобіля та його пробігу після фільтрування (рис 2.11):

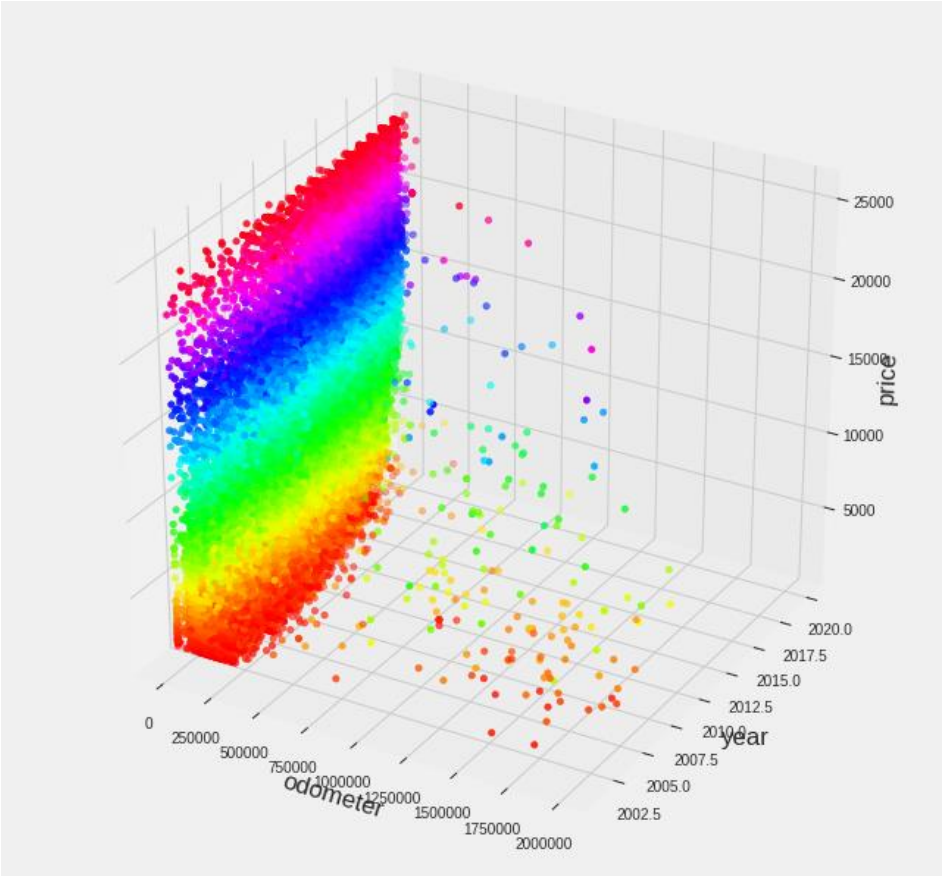


Рисунок 2.11 — Графік співвідношення значень ціни, пробігу та року випуску, після застосування фільтрів

Судячи з графіку вище, фільтрування позитивно вплинуло на співвідношення між цими ознаками, оскільки тепер значення знаходяться в більш реальних межах, та не сконцентровані в одній області значень.

Після здійснення розвідувального аналізу даних, було досліджено взаємозв’язки між ознаками об’єктів датасету, здійснено очищення датасету від порожніх та аномальних даних. В результаті, був отриманий датасет, у якому краще прослідковується взаємозв’язок між ознаками, а також містяться лише актуальні дані. Даний датасет тепер готовий до використання в тренуванні моделей машинного навчання.

2.5 Вибір моделей машинного навчання для вирішення поставленої задачі

Оскільки для вирішення поставленої задачі було вибраний метод

регресії, опишемо властивості найпопулярніших регресійних моделей.

Лінійна регресія - це модель, що використовує лінійний підхід до моделювання зв'язку між залежною змінною та однією або кількома незалежними змінними. У випадку пошуку залежності до однієї незалежної змінної процес називають простою лінійною регресією. Для більш ніж однієї незалежної змінної процес називається множинною нелінійною регресією.

Метод опорних векторів (Support Vector Machine) належить до методів з керованим навчання, що може використовуватись як для класифікації так і для регресійного аналізу. Аналізуючи набір навчальних зразків, кожен з яких позначений як належний до тієї чи іншої з двох категорій, алгоритм навчання SVM будує модель, яка присвоює нові тестові зразки тій чи іншій категорії, роблячи її двійковим лінійним класифікатором.

Лінійний SVR (Support Vector Regression) аналогічний методу SVM. Він також ґрунтується на застосуванні ядерного трюку, але також підходить і для некерованого навчання.

Стохастичний градієнтний спуск (Stochastic Gradient Descent) - є ітеративним методом оптимізації цільової функції за допомогою застосування стохастичного наближення, використовуючи обмежений за розміром тренувальний набір даних, що обирається випадково при кожній ітерації. Особливо корисним цей метод є у задачах аналізу великих даних, оскільки зменшує обчислювальне навантаження.

Метод побудови дерев рішень (Decision Tree) також використовується у задачах регресії, як модель прогнозування, яка відображає особливості (гілки дерев) до висновків про цільове значення (листя дерев). Якщо цільова змінна може приймати безперервні значення, то таке дерево рішень називається деревами регресії.

Random Forest є методом машинного навчання, шо також використовуються як для класифікації так і для регресії, а також інших завдань, які оперують за допомогою побудови численних дерев прийняття рішень і продукують моду для класів (класифікацій) або прогноз (регресія) побудованих дерев. Недоліком цього методу є схильність до перенавчання.

Gradient Boosting є методом машинного навчання для вирішення проблем регресії та класифікації, що формує свою модель у вигляді ансамблю слабших моделей прогнозування, як правило, дерев рішень. Модель будується поетапно, дозволяючи оптимізувати довільну диференційовану функцію втрат. Даний метод лежить в основі потужної бібліотеки XGBoost [18], а також LGBM [19].

Регуляризація Тихонова, слугує для пошуку наближеної відповіді рівняння без єдиного рішення. Такий тип проблем дуже поширений у завданнях машинного навчання, де «найкраще» рішення потрібно вибирати з використанням обмежених даних. Якщо існує унікальне рішення, алгоритм поверне оптимальне значення. Однак якщо існує кілька рішень, він може вибрати будь-яке з них. Даний метод лежить в основі класу Ridge Regressor [20].

Для підвищення стабільності моделей також використовують метод під назвою Bagging. Це ансамблевий алгоритм, що зазвичай застосовується до моделей на основі дерев рішень, але його можна використовувати разом штучними нейронними мережами та регресійними моделями [21].

Основним принципом AdaBoost є встановлення послідовності “слабких учнів” (тобто моделей, передбачення яких лише трохи кращі, ніж випадкові припущення, наприклад, невеликі дерева рішень) у неодноразово змінених версіях даних. Прогнози від усіх моделей потім поєднуються за допомогою зваженого голосування більшості (або суми) для отримання остаточного прогнозу. Модифікація даних при кожній, так званій, підсилювальний ітерації полягає у застосуванні N-ваг до кожного з навчальних зразків. Спочатку ці ваги встановлюються на значення 1 / N, так що перший крок просто навчає слабкого учня на початкових даних. Для кожної наступної ітерації вибіркові ваги індивідуально змінюються і алгоритм навчання повторно застосовується до повторно зважених даних. На даному кроці ті приклади тренувань, які були неправильно спрогнозовані прискореною моделлю, індукованою на попередньому уроці, збільшують їх вагу, тоді як ваги зменшуються для тих, які були прогнозовані правильно. У процесі

ітерацій приклади, які важко передбачити, отримують все більший вплив. Кожен наступний слабкий учень змушений концентруватися на прикладах, пропущених попередніми в послідовності [22].

Враховуючи велику кількість наведених моделей та методів, пропонується використати кожен з них, щоб визначити, яка з них буде найбільш точною для вирішення поставленої задачі. Моделі, що засновуються на принципах, описаних вище, будуть використані на етапі розробки модулю рекомендування ціни вживаного авто.

2.6 Розробка модулю рекомендування ціни вживаного авто

Обравши моделі машинного навчання, приступимо до безпосередньої розробки модулю рекомендування ціни вживаного авто. Враховуючи описані моделі та принципи були обрані наступні 15 моделей машинного навчання, які реалізовані в бібліотеці sklearn, або є частинами окремих бібліотек, таких xgboost та lightgbm. Для вирішення поставленої задачі пропонується застосувати моделі-регресори на основі дерев рішень (Decision Tree Regressor, Extra Trees Regressor, Random Forest) або регресійних моделей (Linear Regression, Ridge Regressor, Bagging Regressor, AdaBoost Regressor, XGB, LGBM, Voting Regressor, Gradient Boosting Regreesor, MLPRegressor, Support Vector Machine, Linear SVR).

Процес порівняння моделей буде побудований такий чином: кожна модель буде запускатись з тренувальною вибіркою даних, після чого її точність буде порівнюватись за трьома критеріями (коефіцієнт детермінації, середньоквадратична похибка та відносна похибка). Результати тестів кожної моделі за кожним критерієм буде зберігатись до завершення тренування усіх моделей, після чого на основі отриманих значень буде створена таблиця для порівняння значень точності моделей, яка і виявить найбільш ефективну модель серед усіх відібраних. Наведем код реалізації тренування кількох моделей, а саме моделей Decision Tree Regressor, XGBoost, Support Vector Regressor (SVR), а також LGBM(рис. 2.12, 2.13, 2.14, 2.15):

```
# Decision Tree Regression
from sklearn.tree import DecisionTreeRegressor

decision_tree = DecisionTreeRegressor()
decision_tree.fit(train, target)
acc_model(5,decision_tree,train,test)
```

Рисунок 2.12 — Код реалізації навчання моделі Decision Tree Regressor

```
import xgboost as xgb


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)
```

Рисунок 2.13 — Код реалізації навчання моделі XGBoost Regressor

```
# Support Vector Machines
from sklearn.svm import SVR

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

Рисунок 2.14 — Код реалізації навчання моделі SVR (Support Vector Regressor)

```
import lightgbm as lgb

#%% split training set to validation set
Xtrain, Xval, Ztrain, Zval = train_test_split(trainb, targetb, test_size=0.2, random_state=0)
train_set = lgb.Dataset(Xtrain, Ztrain, silent=False)
valid_set = lgb.Dataset(Xval, Zval, silent=False)

params = {
        'boosting_type':'gbdt',
        'objective': 'regression',
        'num_leaves': 31,
        'learning_rate': 0.01,
        '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 = lgb.train(params, train_set = train_set, num_boost_round=10000,
                   early_stopping_rounds=50,verbose_eval=10, valid_sets=valid_set)
```

Рисунок 2.15 — Код реалізації навчання моделі LGBM

З фрагментів коду видно, ще результати тренування моделі обробляються функціями acc_model або acc_boosting_model, саме ці функції тестують результати за критеріями, названими вище. Код реалізації інших моделей, а також допоміжних функцій acc_model та acc_boosting_model наведені у додатку В (Лістинг програми). Після програмної реалізації всіх моделей машинного навчання та їх тренування на попередньо оброблених даних, слід приступити до аналізу отриманих результатів та вибору найточнішої моделі для вирішення поставленої задачі.

2.7 Аналіз отриманих результатів та визначення найефективнішої моделі

На попередніх етапах роботи було відфільтровано дані, відібрано

оптимальні ознаки, визначено типи моделей, які варто використовувати для передбачення даних. Для розв'язання задачі передбачення використано авторський досвід, зокрема різноманітні програми-кернели на веб-платформі Kaggle [23]. Їх застосування дозволило ранжувати усі дані по США по точності R^2-критерію (sklearn.r2_test [24]) (рис. 2.16).

	Model	r2_train	r2_test	d_train	d_test	rmse_train	rmse_test
8	LGBM	91.06	86.12	12.53	14.56	179,382.56	208,609.17
12	ExtraTreesRegressor	99.95	84.19	0.13	13.73	12,790.79	227,775.52
6	Random Forest	97.17	84.11	5.82	14.51	97,206.65	225,093.18
11	BaggingRegressor	97.19	84.10	5.80	14.50	96,970.11	224,957.57
7	XGB	88.33	83.54	14.57	15.84	204,923.27	223,658.23
5	Decision Tree Regressor	99.95	77.11	0.13	16.98	12,789.23	288,694.89
3	MLPRegressor	67.37	68.04	21.65	21.74	297,952.31	297,286.15
9	GradientBoostingRegressor	62.21	62.57	21.81	21.97	296,603.86	297,399.42
14	VotingRegressor	28.82	30.39	29.35	29.36	389,901.74	387,985.65
0	Linear Regression	26.95	28.58	29.45	29.45	389,782.46	387,856.11
10	RidgeRegressor	26.92	28.56	29.45	29.45	389,782.47	387,856.51
4	Stochastic Gradient Decent	22.25	23.99	29.58	29.56	390,531.29	388,697.83
2	Linear SVR	11.13	12.93	29.68	29.74	409,624.83	407,699.79
13	AdaBoostRegressor	-102.47	-102.86	34.83	35.04	414,316.82	416,350.42
1	Support Vector Machines	-1,017.94	-1,020.26	38.45	38.60	508,854.61	510,915.06

Рисунок 2.16 — Ранжування за R^2-критерієм результатів передбачення моделей, натренованих на американських даних

Точність моделей оцінюється за трьома критеріями: RMSE (root-mean-square-error або середньоквадратична похибка), R^2 (R-squared або коефіцієнт детермінації) і за відносною похибкою на основі вбудованої функції MAE (абсолютне значення середньої похибки).

Пояснимо, яким чином обчислюються ці методи оцінювання точності. RMSE обчислюється у вигляді квадратного кореня з середнього значення квадратів різниці результатів натренованої моделі, та даних, що містяться у тренувальному наборі даних [25]:

$$RMSE = \sqrt{\frac{\sum_{t=1} T(\hat{y}_t - y_t)^2}{T}}, \quad (2.1)$$

де T— загальна кількість результатів (рядків даних);

$\hat{y}_t$— результат обчислення моделі;

y_t — відповідна величина з тренувального набору даних.

Коефіцієнт детермінації є статистичним методом, що використовується в статистичних моделях як показник залежності варіації залежної змінної від варіації незалежних змінних, що вказує на скільки отримані спостереження підтверджують модель:

$$R^2(y, \hat{y}) = 1 = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}, \quad (2.2)$$

де y_i— істинне значення з тренувального набору даних;

$\hat{y}_i$— передбачене значення відповідного набору ознак;

$\bar{y}_i$ — середнє арифметичне істинних значень з тренувального масиву даних.

Відносна похибка обчислюється за відомою формулою:

$$\delta = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{\sum_{i=1}^{n}|y_i|} * 100\%, \quad (2.3)$$

де$\hat{y}_t$— результат обчислення моделі;

y_t — відповідна величина з тренувального набору даних.

Аналіз отриманих результатів показав, що найкращою моделлю за R^2-критерієм є модель LGBM. Її застосування до даних США дозволило отримати точність передбачення 86,12%.

Результат аналогічного застосування запропонованої технології до українського набору даних наведено на рисунку 2.17. Найкращою моделлю за R^2-критерієм є знову LGBM. Її точність передбачення склала: 85,55%.

	Model	r2_train	r2_test	d_train	d_test	rmse_train	rmse_test
4	LGBM	92.51	85.55	31.92	36.05	38,031.21	42,672.70
5	GradientBoostingRegressor	94.14	85.45	6.45	35.26	10,965.31	46,303.93
6	BaggingRegressor	87.26	83.61	14.22	35.21	20,641.95	46,322.15
3	XGB	91.98	83.12	29.38	35.23	35,710.24	42,575.48
2	Random Forest	92.35	81.38	14.33	34.85	20,998.43	46,093.82
1	Decision Tree Regressor	84.16	78.43	1.40	38.18	8,257.61	57,887.24
0	MLPRegressor	60.75	57.71	53.76	53.12	53,822.12	53,586.36
7	ExtraTreesRegressor	27.75	25.63	1.40	35.70	8,257.72	49,729.96

Рисунок 2.17 — Ранжування за R^2-критерієм результатів передбачення моделей, натренованих на американських даних

2.8 Висновки

Дослідження масиву даних, що містить інформацію про продажі вживаних авто в США, за запропонованою інтелектуальною технологією на основі бібліотек Python показало, що, по-перше, для точного передбачення ціни необхідно провести докладний попередній розвідувальний аналіз і відфільтрувати помилкові та аномальні дані, а також відкинути ознаки, які у них відрізняються, щоб результати можна було порівнювати. Після цього можна переходити до тренування моделей та порівняння їхньої точності для вибору оптимальної.

Для вирішення поставленої задачі найкраще обрати моделі-регресори, оскільки завдання передбачення великої кількості класів ціни полягає в аналізі залежності деякої величини (у нашому випадку ціни автомобіля) відносно інших ознак (виробник, модель, обсяг пробігу, трансмісія тощо).

Були обрані 15 моделей і натреновані на американських українських даних. Результат показав співставні результати: оптимальною моделлю в обох випадках є модель LGBM з бібліотеки lightgbm, з точністю 85-86%

3 ЕКОНОМІЧНА ЧАСТИНА

3.1 Технологійчний аудит розробленої системи

В останні кілька років збільшився об’єм продажів на ринку вживаних авто. Такий ріст спричинений тенденцією до покупки вживаних авто з-за кордону, з наступним розмитненням на території України.

Внаслідок такого швидкого росту ринку постало питання формування конкурентоспроможної ціни на вживаний автомобіль.

Крім формування ціни за допомогою звернення експертів, продавці мають змогу порівняти ціну на схожі автомобілі за допомогою веб-порталів, що спеціалізуються на продажу вживаних авто. Деякі з таких веб-порталів також пропонують інструменти для запитів статистичної інформації про ціноутворення на ту чи іншу модель автомобіля, але ця інформація залежить від достатньої кількості наявних записів про автомобіль, що цікавить користувача. Тому також варто враховувати, що для такого випадку, потрібно скористатись інструментами, що можуть прогнозувати ціну автомобіля на основі введених користувачем даних.

З метою якнайточнішого прогнозування комерційних ефектів від можливої реалізації результатів розробки слід провести технологічний аудит нашої розробк. Такий аудит передбачає оцінювання наукового рівня розробки, а також визначення потенційних можливостей для її комерційного використання. Технічний аудит вирішено провести за допомогою експертного методу. Експертами запросимо: к.т.н., доцента Козачко О.М., к.т.н., доцента Ящолта А.Р. та к.т.н., доцента Крижановського Є.М.

Запрошені експерти є фахівцями у даній галузі наукових досліджень, а саме у дослідженні масивів даних та розробці моделей прогнозування.

Критерії оцінювання наукового та комерційного потенціалу розробки наведені у таблиці 3.1.

Таблиця 3.1 – Критерії оцінювання наукового та комерційного потенціалу розробки та їх бальна оцінка.

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність 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 нижче за ціни аналогів	Ціна продукту значно нижче 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
Ринкові перспективи					
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Практична здійсненність					
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 3.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Практична здійсненність					
8	Відсутні фахі-вці як з технічної, так і з комерційної реалізації ідеї	Необхідно на-ймати фахівців або витрачати значні кошти та час на на-вчання наявних фахівців	Необхідне не-значне 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 фінансуванн я ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансуванн я є	Потрібні незначні фінансові ресурси. Джерела фінансуванн я є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуютьс я у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуютьс я у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років.	Термін реалізації ідеї від 3-х до 5-ти років.	Термін реалізації ідеї менше 3-х років.	Термін реалізації ідеї менше 3-х років.
12	Необхідна розробка регламентни х документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки пові-домлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу розробки потрібно

звести в таблицю за зразком таблиці 3.2.

Таблиця 3.2 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	Козачко О. М	Крижановський Є.М.	Ящолт А.Р.
	Бали, виставлені експертами:		
1	4	3	4
2	3	4	4
3	4	4	3
4	4	2	3
5	3	3	4
6	2	4	2
7	4	3	3
8	3	4	3
9	3	3	3
10	3	4	3
11	3	4	3
12	4	3	4
Сума балів	$СБ_1$=40	$СБ_2$=41	$СБ_1$=39
Середньоарифметична сума балів СБ	$\acute{СБ}\frac{\sum_1^3 СБ_i}{3} = \frac{120}{3} = 40.$		

За даними таблиці 3.2 можна зробити висновок, щодо рівня комерційного потенціалу розробки. Зважимо на результат й порівняємо його з рівнями комерційного потенціалу розробки, що представлено в таблиці 3.3.

Таблиця 3.3 – Рівні комерційного потенціалу розробки.

Середньоарифметична сума балів СБ,	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Рівень комерційного потенціалу розробки, становить 40 балів, що відповідає рівню «вище середнього».

3.2 Розрахунок витрат на виконання даної роботи

Проведемо прогнозування витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи для розробки програмного забезпечення, яке складається з таких етапів:

1-й етап: розрахунок витрат, які безпосередньо стосуються виконавців даного розділу роботи;

2-й етап: розрахунок загальних витрат на виконання даної роботи;

3-й етап: прогнозування загальних витрат на виконання та впровадження результатів даної роботи.

Виконаємо розрахунок витрат приймаючи до уваги те, що для розробки інформаційної технології було залучено одного розробника програмного забезпечення. Основна заробітна плата кожного із розробників (дослідників) $З_о$, якщо вони працюють в наукових установах бюджетної сфери:

$$З_о = \frac{М}{T_p} \cdot t \text{ [грн]}, \quad (3.1)$$

де М – місячний посадовий оклад конкретного розробника (інженера, до-слідника, науковця тощо), грн;

$Т_р$ – число робочих днів в місяці; приблизно $Т_р$ = (20…23) дні;

t – число робочих днів роботи розробника (дослідника), розробка програмного забезпечення триває 60 днів.

Зроблені розрахунки внесені до таблиці 3.4.

Таблиця 3.4 – Основна заробітна плата розробників.

Найменування посади виконавця	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на оплату праці, грн.
Розробник	10 000	454.54	60	27267
Всього				27267

Додаткова заробітна плата $З_Д$ всіх розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12%) від суми основної заробітної плати розробників та робітників, розраховується наступним чином:

$$З_Д = 0.10 \cdot 27267 = 2726{,}7(\text{грн}).$$

Нарахування на заробітну плату $Н_{ЗП}$ розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується за формулою:

$$Н_{зп} = (З_0 + З_Д) \cdot \frac{\beta}{100} \ [\text{грн}], \qquad (3.2)$$

де $З_0$ – основна заробітна плата розробника, грн.;

$З_Д$ – додаткова заробітна плата розробника, грн.;

β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування – 22 %.

Розрахуємо нарахування на заробітну плату:

$$Н_{ЗП} = (27267 + 2726.7) \cdot 0{,}22 = 6598{,}61(\text{грн}).$$

Амортизація обладнання, комп'ютерів та приміщень А, які використовувались під час (чи для) виконання даного етапу роботи.

Дані відрахування розраховують по кожному виду обладнання, приміщенням тощо.

У спрощеному вигляді амортизаційні відрахування А в цілому бути розраховані за формулою:

$$А = \frac{Ц \cdot Т}{12 \cdot Т_В}[\text{грн}], \qquad (3.3)$$

де Ц – загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, що використовувались для виконання даного етапу роботи, грн;

Т – фактична тривалість використання, міс;

$Т_В$– термін, використання обладнання, приміщень тощо, місяці, роки.

Зроблені розрахунки наведено в таблиці 3.5.

Таблиця 3.5 – Амортизаційні відрахування

Найменування	Балансова вартість, грн	Термін використання, роки	Фактична тривалість використання, міс.	Величина амортизаційних відрахувань, грн
Офісне приміщення	12000	1	3	4000
Ноутбук	15000	1	3	5000
Всього				9000

Інформацію про матеріали, що використовуються при виготовленні даного інноваційного продукту внесено до таблиці 3.6.

Таблиця 3.6 – Матеріали, що використовуються при виготовленні даного продукту

Найменування матеріалу	Ціна за одиницю, грн.	Витрачено, шт.	Вартість витраченого матеріалу з урахуванням доставки, грн
Папір (пачка)	90,00	1	99,00
Канцтовари	40,00	1	44,00
Всього	143,00		

Під час розробки програмного продукту використовували безкоштовні програмні продукти. Здійснювалась оплата послуги інтернет для пошуку інформації. Програмні послуги, що використовуються при виготовленні даного продукту внесено до таблиці 3.7.

Таблиця 3.7 – Програмні послуги, що використовуються при виготовленні даного продукту

Найменування послуги	Ціна за 1 міс., грн.	Кількість місяців	Вартість використаної послуги.
Інтернет-провайдер	150	3	450
Всього			450

Витрати на силову електроенергію $В_е$ розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п \ [грн], \quad (3.4)$$

де В – вартість 1 кВт-год. електроенергії, 1,8 грн/кВт;

П – установлена потужність обладнання, кВт;

Ф – фактична кількість годин роботи обладнання, годин;

$К_п$ – коефіцієнт використання потужності;

Потужність використовуваного комп'ютера становить П=0.6 кВт.

Фактична кількість годин роботи обладнання – 480 год (60 робочих днів по 8 годин на день).

$$В_е = 1{,}8 \cdot 0{,}6 \cdot 480 \cdot 0{,}6 = 311{,}04(грн).$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини розділу роботи В.

$$B = 27267 + 2726.7 + 6598.61 + 9000 + 143 + 377.04 + 450 = 46562.35(грн).$$

2-й етап: розрахунок загальних витрат на виконання даної роботи.

$$В_{заг} = \frac{В}{\alpha}[грн], \quad (3.5)$$

$$В_{заг} = \frac{46562{,}35}{1} = 46562{,}35(грн),$$

3-й етап. Прогнозування загальних витрат на виконання та впровадження результатів виконаної роботи. Прогнозування витрат ЗВ на виконання та впровадження виконаної роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta}[грн], \quad (3.6)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Оскільки розробка знаходиться:

- на стадії розробки дослідного зразка, то $\beta \approx 0,5$
- на стадії технічного проектування , то $\beta \approx 0,2$
- на стадії розробки конструкторської документації то $\beta \approx 0,3$
- на стадії розробки технології, то $\beta \approx 0,4$
- на стадії науково-дослідних робіт, то $\beta \approx 0,1$
- на стадії промислового зразка, $\beta \approx 0,7$
- на стадії впровадження, то $\beta \approx 0,9$

$$ЗВ = \frac{46562,35}{0,7} = 66517.64 \text{ (грн)}.$$

Отже, прогноз загальних витрат на виконання та впровадження результатів становить 66517.64 грн.

3.3 Прогнозування комерційних ефектів від можливої реалізації результатів розробки

У даному підрозділі проведемо кількісне прогнозування, яку вигоду, зиск можна отримати у майбутньому від впровадження результатів виконаної наукової роботи. В умовах ринку узагальнюючим позитивним результатом, що його отримує підприємство від впровадження результатів тієї чи іншої розробки, є збільшення чистого прибутку підприємства. Зростання

чистого прибутку можна оцінити у теперішній вартості грошей.

Зростання чистого прибутку забезпечить підприємству надходження додаткових коштів, які дозволять покращити фінансові результати діяльності.

Виконання даної наукової роботи та впровадження її результатів складає приблизно 1 рік.

Позитивні результати від впровадження розробки очікуються вже в перший рік впровадження.

Проведемо детальніше прогнозування позитивних результатів та кількісне їх оцінювання по роках.

Обчислимо збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_я = \sum_1^n (\Delta П_я \cdot N + П_я \cdot \Delta N)_n \text{ [грн]}, \quad (3.7)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

Припустимо, що внаслідок впровадження результатів наукової розробки покращується якість, що дозволяє збільшити прибуток підприємства на 100грн, а кількість одиниць реалізованої послуги збільшиться: протягом першого року – на 200 од., протягом другого року – ще на 300 од., протягом третього року – ще на 350 од. Орієнтовно: реалізація

послуг до впровадження результатів наукової розробки складала 25 шт., а її прибуток підприємства – 500рн.

Спрогнозуємо збільшення чистого прибутку підприємства від впровадження результатів наукової розробки у кожному році відносно базового.

Збільшення чистого прибутку підприємства $\Delta П_1$ протягом першого року складе:

$$\Delta П_1=25\cdot 500 + (500+100) \cdot 200 = 132\ 500 \text{ (грн)}.$$

Обчислимо збільшення чистого прибутку підприємства $\Delta П_2$ протягом другого року:

$$\Delta П_2=25\cdot 500+(500+100) \cdot (200+300) = 312\ 500 \text{ (грн)}.$$

Збільшення чистого прибутку підприємства $\Delta П_3$ протягом третього року становитиме:

$$\Delta П_3=25\cdot 500 + (500+100) \cdot (200+300+350) = 522\ 500 \text{ (грн)}.$$

Розрахунки показують, що відповідно прогнозуванню комерційний ефект від впровадження розробки виражається у значному збільшенні чистого прибутку підприємства.

Основними показниками, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Розрахунок ефективності вкладених інвестицій передбачає:

1-й крок. Розрахунок теперішньої вартості інвестицій PV, що вкладаються в наукову розробку. Такою вартістю ми можемо вважати прогнозовану величину загальних витрат ЗВ на виконання та впровадження результатів НДДКР, тобто ЗВ = PV = 66517.64 (грн).

2-й крок. Розрахунок очікуваного збільшення прибутку $\Delta П_i$, що його отримає підприємство (організація) від впровадження результатів наукової розробки, для кожного із років, починаючи з першого року впровадження проведено вище.

3-й крок. Будуємо вісь часу, на якій відображаємо всі платежі (інвестиції та прибутки), що мають місце під час виконання науково-дослідної роботи та впровадження її результатів. Платежі показуємо у ті терміни, коли вони здійснюються.

Припустимо, що загальні витрати ЗВ на виконання та впровадження результатів НДДКР (або теперішня вартість інвестицій PV) дорівнює 66517.64 грн. Результати вкладених у наукову розробку інвестицій почнуть з'являтися протягом трьох років.

Ці результати виявляться у тому, що у першому році підприємство отримає збільшення чистого прибутку на 132500 грн відносно базового року, у другому році – збільшення чистого прибутку на 312500 грн (відносно базового року), у третьому році – збільшення чистого прибутку на 522500 грн (відносно базового року).

Тоді рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, наведений на рис. 3.1.

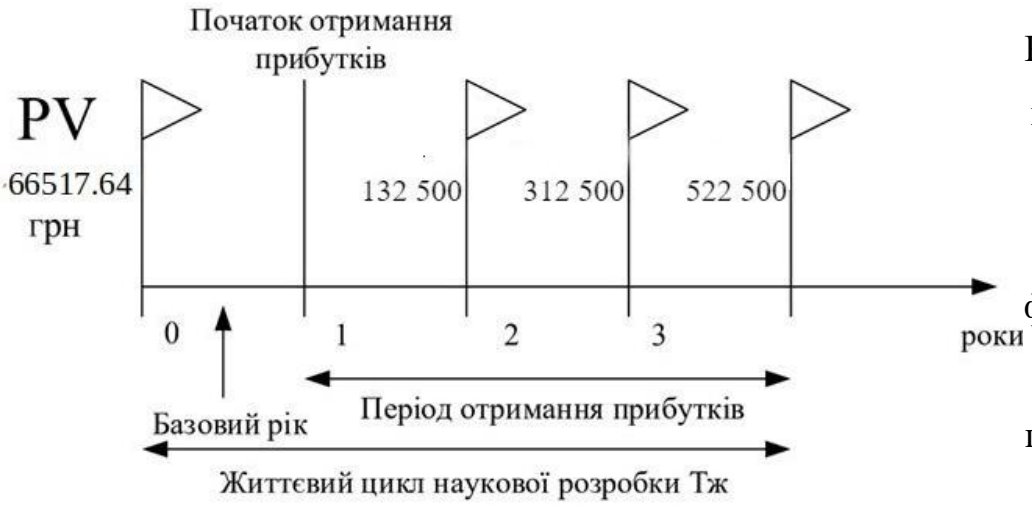


Рисунок 3.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн.

Приведена вартість всіх чистих прибутків ПП розраховується за формулою:

$$ПП = \sum_1^t \frac{\Delta П_i}{(1+\tau)^t}, \quad (3.8)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

t – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні - 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки „0”.

$$ПП = \frac{132500}{(1+0{,}1)^1} + \frac{312500}{(1+0{,}1)^2} + \frac{522500}{(1+0{,}1)^3} = 771280{,}99(грн).$$

$Е_{абс} = 771280{,}99 - 66517.64 = 704763.35$ (грн).

Оскільки $Е_{абс} > 0$, результат від проведення наукових досліджень щодо розробки програмного продукту та їх впровадження принесе прибуток, тобто є доцільним, але це ще не свідчить про те, що інвестор буде зацікавлений у фінансуванні даної програми.

5-й крок. Розраховують відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$ за формулою:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1, \quad (3.9)$$

де $Е_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$Т_ж$ – життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{1 + \frac{693342{,}23}{77938{,}76}} - 1 = \sqrt[3]{10.59} - 1 = 1{,}19 \text{ або } 119{,}00\%.$$

Порівняємо $Е_в$ з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть.

Спрогнозуємо величину $\tau_{мін}$ У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f, \quad (3.10)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; d = 0,14;

f – показник, що характеризує ризикованість вкладень; величина f =

0,3.

$$\tau = 0{,}14 + 0{,}3 = 0{,}44.$$

Припустимо, що за даних умов прибуток буде збільшуватись, то у інвестора є потенційна зацікавленість у фінансуванні даної наукової розробки.

6-й крок. Розраховують термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ за формулою:

$$T_{ок} = \frac{1}{E_в} \text{ [роки]}. \quad (3.11)$$

$$T_{ok} = \frac{1}{1{,}19} = 0{,}84 \text{ (роки)}.$$

Оскільки термін окупності вкладених у реалізацію наукового проекту інвестицій менше трьох років ($T_{ок} < 3$ років), то фінансування нової розробки є доцільним.

3.4 Висновки

В даному розділі було здійснено оцінювання комерційного потенціалу розробки інформаційної системи рекомендування ціни вживаного авто.

Проведено технологічний аудит з залученням трьох експертів. Аналіз експертних даних показав, що рівень комерційного потенціалу розробки вище середнього. Дослідження комерційного потенціалу розробки показав, що програмний продукт за своїми характеристиками випереджає аналогічні програмні продукти і є перспективною розробкою. Він має кращі функціональні показники, а тому є конкурентоспроможним товаром на ринку.

На основі зроблених підрахунків в економічній частині магістерської кваліфікаційної роботи досягнуті наступні результати:

- витрати на розробку та її впровадження складають 66517,64 грн;
- абсолютний ефект розробки складає 704763,35 грн;
- за здійсненими підрахунками було виявлено, що внутрішня норма дохідності була досягнута і складає 119%;
- термін окупності системи, що розробляється складає 0,84 років.

Отже, дана розробка є економічно вигідною та доцільною для вкладання в неї коштів інвесторів

ВИСНОВКИ

Розглянуто проблему прогнозування ціни на вживаний автомобіль, шляхом обробки масиву даних методами розвідувального аналізу, з наступним використанням методів машинного навчання. За рахунок розвідувального аналізу були досліджені обрані датасети, здійснене очищення та фільтрування даних з метою покращення якості його вмісту. Також були досліджені загальні закономірності між значеннями певних характеристик, зазначених у наборі даних, а також вибрані ключові ознаки, які будуть використані у процесі прогнозу ціни. Далі був обраний набір регресійних моделей, кожна з яких була натренована на підготовленому під час розвідувального аналізу наборі даних. Результати прогнозів моделей були порівняні за кількома критеріями, після чого була обрана модель з найкращим показником точності. Серед обраного набору моделей такою стала модель LGBM, з точністю 86%.

Продемонстровано ефективність розробленого програмного модулю, та методів розвідувального аналізу. Описані застосовані бібліотеки та методи.

У ході виконання економічної частини кваліфікаційної роботи на основі розрахунків було доведено, що продукт є економічно доцільним, оскільки витрати на розробку вказаного засобу з використанням відповідної технології становлять 66517,64 грн. Показник ефективності вкладених інвестицій $E_{абс}$ = 704763,35 грн, відносної – 119%, а термін окупності інвестицій становить 0,84 роки.

За результатами магістерської кваліфікаційної роботи опублікована науково-технічна стаття. Отже, поставлені задачі магістерської кваліфікаційної роботи були виконані в повному обсязі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Статистика продажів автомобілів у 2019 році. [Електронний ресурс] — Режим доступу до ресурсу: http://ukrautoprom.com.ua/uk/statistika/statistika-2019
2. Global Ecommerce 2019. [Електронний ресурс] — Режим доступу до ресурсу: https://www.emarketer.com/content/global-ecommerce-2019
3. Ukraine Country Commercial Guide [Електронний ресурс] — Режим доступу до ресурсу: https://www.export.gov/apex/article2?id=Ukraine-eCommerce
4. Подсчёт средней цены [Електронний ресурс] — Режим доступу до ресурсу: https://api-docs-v2.readthedocs.io/ru/latest/auto_ria/used_cars/average-price.html
5. A Very Brief Introduction to Machine Learning With Applications to Communication Systems / Osvaldo Simeone // IEEE Transactions on Cognitive Communications and Networking. — Vol. 4. — No. 4. — p.p.— 648-664. —Nov.2018.
6. Introduction to machine learning for brain imaging / Steven Lemm, Benjamin Blankertz, Thorsten Dickhaus, Klasu-Robert Muller // Neuroimage — Vol. 56. — No. 2. — p.p.—387-399. — May 2011
7. About Python™ | Python.org [Електронний ресурс] — Режим доступу до ресурсу: https://www.python.org/about/
8. Getting Started with MATLAB [Електронний ресурс] — Режим доступу до ресурсу: https://www.mathworks.com/help/matlab/getting-started-with-matlab.html
9. Introduction to R [Електронний ресурс] — Режим доступу до ресурсу: https://www.r-project.org/about.html
10. Project Jupyter | About Us [Електронний ресурс] — Режим доступу до ресурсу: https://jupyter.org/about
11. Used Cars Dataset - Vehicles listings from Craigslist.org [Електронний

ресурс] — Режим доступу до ресурсу: https://www.kaggle.com/austinreese/craigslist-carstrucks-data

12. Pandas Dataframe API reference [Електронний ресурс]— Режим доступу до ресурсу: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html
13. Module pandas_profiling [Електронний ресурс]— Режим доступу до ресурсу: https://pandas-profiling.github.io/pandas-profiling/docs/
14. A new correlation coefficient between categorical, ordinal and interval variables with Pearson characteristics [Електронний ресурс]— Режим доступу до ресурсу: https://arxiv.org/abs/1811.11440
15. Matplotlib API Overview [Електронний ресурс]— Режим доступу до ресурсу: https://matplotlib.org/api/index.html
16. Comprehensive Data Exploration with Python [Електронний ресурс]— Режим доступу до ресурсу: https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python
17. sklearn.preprocessing.LabelEncoder — scikit-learn 0.22 documentation [Електронний ресурс] — Режим доступу до ресурсу: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.htm
18. About XGBoost [Електронний ресурс] — Режим доступу до ресурсу: https://xgboost.ai/about
19. LGBM Documentation [Електронний ресурс] — Режим доступу до ресурсу: https://lightgbm.readthedocs.io/en/latest/
20. Ridge and Lasso Regression: A Complete Guide with Scikit-Learn [Електронний ресурс] — Режим доступу до ресурсу: https://towardsdatascience.com/ridge-and-lasso-regression-a-complete-guide-with-python-scikit-learn-e20e34bcbf0b
21. Bootstrap aggregating (bagging) [Електронний ресурс] — Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Bootstrap_aggregating
22. Ensemble methods [Електронний ресурс] — Режим доступу до ресурсу: https://scikit-learn.org/stable/modules/ensemble.html#adaboost

23. Used Cars Price Prediction by 15 models [Електронний ресурс]— Режим доступу до ресурсу: https://www.kaggle.com/vbmokin/used-cars-price-prediction-by-15-models
24. Sklearn.metrics.r2_score [Електронний ресурс]— Режим доступу до ресурсу:https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html
25. SPOTting Model Parameters Using a Ready-Made Python Package [Електронний ресурс] — Режим доступу до ресурсу: https://doi.org/10.1371/journal.pone.0145180

Додаток А

Технічне завдання

Вінницький національний технічний університет
Факультет комп'ютерних систем і автоматики

ЗАТВЕРДЖУЮ
Завідувач кафедри САКМІГ
_____ д.т.н.,проф. В.Б. Мокін
(підпис)
"__" _____ 2019

ТЕХНІЧНЕ ЗАВДАННЯ

на магістерську кваліфікаційну роботу

ІНФОРМАЦІЙНА СИСТЕМА РЕКОМЕНДУВАННЯ ЦІНИ ВЖИВАНОГО АВТО

08-53.МКР.005.02.000 ТЗ

Керівник магістерської
кваліфікаційної роботи
д.т.н., проф.
_____В.Б. Мокін
(підпис)
"__" _____ 2019 р.
Розробив студент гр. ІСТ-18м
_____А.В. Лосенко
(підпис)
"__" _____ 2019 р.

Вінниця 2019

1. Підстава для проведення робіт

Підставою для виконання роботи є наказ № __ по ВНТУ від «__» _____ 201_ р., та індивідуальне завдання на МКР, затверджене протоколом № __ засідання кафедри САКМІГ від «__» _____ 201_ р.

2. Джерела розробки

Серед джерел, які будуть використані в ході розробки магістерської кваліфікаційної роботи є: Державні стандарти України (ДСТУ) 3008-2015 «Інформація та документація. Звіти у сфері науки і техніки. Структура і правила оформлювання», 2481-94 “Системи оброблення інформації. Інтелектуальні інформаційні технології. Терміни та визначення”, завдання на МКР, довідкова та технічна література.

3. Мета і призначення роботи.

Метою дослідження є систематизація сучасних методів розвідувального аналізу даних на Python, запропонування технологій аналізу та передбачення ціни на вживані авто та їх перевірка за даними зі США та України.

4. Вихідні дані для проведення робіт

1) Набір даних про продажі вживаних автомобілів з веб-платформи Kaggle, у форматі CSV.

2) Набір даних про продажі вживаних автомобілів з веб-системи медіа-корпорації “RIA”, у форматі CSV.

5. Методи дослідження

1) Розвідувальний аналіз даних.

2) Регресійні моделі машинного навчання.

6. Етапи роботи і терміни їх виконання

1) Аналіз методів розрахунку рекомендованої ціни..……..__. – __

2) Розробка модулю розвідувального аналізу та визначення ключових ознак вибірки даних.....…….…..__. – __

3) Розробка модулю рекомендування ціни вживаного авто…...__. – __

7. Очікувані результати та порядок реалізації

Розробка інформаційної системи рекомендування ціни на вживане авто, яка на відміну від існуючих, здійснює миттєву рекомендацію ціни вживаного авто на основі даних, які введені користувачем.

8. Вимоги до розробленої документації

Пояснювальна записка оформлена у відповідності до вимог «Методичних вказівок до виконання та оформлення магістерських кваліфікаційних робіт для студентів спеціальності 126 – «Інформаційні системи та технології» денної форми навчання».

9. Порядок приймання роботи

Публічний захист…….__.__.201_ р.

Початок розробки «___» _____ 201_ р.

Граничні терміни виконання МКР «___» _____201_ р.

Розробив студент групи ІСТ-18м _____ Лосенко А. В.

Додаток Б

Лістинг програми

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

# preprocessing
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV,
StratifiedKFold
import pandas_profiling as pp

# models
from sklearn.linear_model import LinearRegression, SGDRegressor, RidgeCV
from sklearn.svm import SVR, LinearSVR
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor,
ExtraTreesRegressor
from sklearn.ensemble import BaggingRegressor, AdaBoostRegressor, VotingRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.tree import DecisionTreeRegressor
import sklearn.model_selection
from sklearn.model_selection import cross_val_predict as cvp
from sklearn import metrics
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error,
accuracy_score
import xgboost as xgb
import lightgbm as lgb

import matplotlib.style as style
import matplotlib.gridspec as gridspec
from scipy import stats
from mpl_toolkits import mplot3d
import seaborn as sns

# 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('max_columns',100)

train0 = pd.read_csv('/kaggle/input/craigslist-carstrucks-data/craigslistVehicles.csv')
train0.head(5)

drop_columns = ['url', 'city', 'city_url', 'title_status', 'VIN', 'size', 'image_url', 'desc', 'lat','long',
'paint_color']
```

```
train0 = train0.drop(columns = drop_columns)

train0 = train0.dropna()
train0.head(5)
# 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 = train0.columns.values.tolist()
for col in features:
    if train0[col].dtype in numerics: continue
    categorical_columns.append(col)
# Encoding categorical features
for col in categorical_columns:
    if col in train0.columns:
        le = LabelEncoder()
        le.fit(list(train0[col].astype(str).values))
        train0[col] = le.transform(list(train0[col].astype(str).values))

# Thanks to : https://www.kaggle.com/aantonova/some-new-risk-and-clusters-features
def reduce_mem_usage(df, verbose=True):
    numerics = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']
    start_mem = df.memory_usage().sum() / 1024**2
    for col in df.columns:
        col_type = df[col].dtypes
        if col_type in numerics:
            c_min = df[col].min()
            c_max = df[col].max()
            if str(col_type)[:3] == 'int':
                if c_min > np.iinfo(np.int8).min and c_max < np.iinfo(np.int8).max:
                    df[col] = df[col].astype(np.int8)
                elif c_min > np.iinfo(np.int16).min and c_max < np.iinfo(np.int16).max:
                    df[col] = df[col].astype(np.int16)
                elif c_min > np.iinfo(np.int32).min and c_max < np.iinfo(np.int32).max:
                    df[col] = df[col].astype(np.int32)
                elif c_min > np.iinfo(np.int64).min and c_max < np.iinfo(np.int64).max:
                    df[col] = df[col].astype(np.int64)
            else:
                if c_min > np.finfo(np.float16).min and c_max < np.finfo(np.float16).max:
                    df[col] = df[col].astype(np.float16)
                elif c_min > np.finfo(np.float32).min and c_max < np.finfo(np.float32).max:
                    df[col] = df[col].astype(np.float32)
                else:
                    df[col] = df[col].astype(np.float64)
    end_mem = df.memory_usage().sum() / 1024**2
    if verbose: print('Mem. usage decreased to {:5.2f} Mb ({:.1f}% reduction)'.format(end_mem, 100 * (start_mem - end_mem) / start_mem))
    return df

train0 = reduce_mem_usage(train0)
```

```
train0.head(10)
#Thanks to https://www.kaggle.com/masumrumi/a-detailed-regression-guide-with-house-pricing
def plotting_3_chart(df, feature):
    ## Importing seaborn, matplotlab and scipy modules.
    style.use('fivethirtyeight')

    ## Creating a customized chart. and giving in figsize and everything.
    fig = plt.figure(constrained_layout=True, figsize=(15,10))
    ## creating a grid of 3 cols and 3 rows.
    grid = gridspec.GridSpec(ncols=3, nrows=3, figure=fig)

    ## Customizing the histogram grid.
    ax1 = fig.add_subplot(grid[0, :2])
    ## Set the title.
    ax1.set_title('Histogram')
    ## plot the histogram.
    sns.distplot(df.loc[:,feature], norm_hist=True, ax = ax1)

    # customizing the QQ_plot.
    ax2 = fig.add_subplot(grid[1, :2])
    ## Set the title.
    ax2.set_title('QQ_plot')
    ## Plotting the QQ_Plot.
    stats.probplot(df.loc[:,feature], plot = ax2)

    ## Customizing the Box Plot.
    ax3 = fig.add_subplot(grid[:, 2])
    ## Set title.
    ax3.set_title('Box Plot')
    ## Plotting the box plot.
    sns.boxplot(df.loc[:,feature], orient='v', ax = ax3 );

plotting_3_chart(train0, 'price')

#Thanks to https://towardsdatascience.com/an-easy-introduction-to-3d-plotting-with-matplotlib-801561999725

fig = plt.figure(figsize=(10,10))
ax = plt.axes(projection="3d")

z_points = train0['price']
x_points = train0['odometer']
y_points = train0['year']
ax.scatter3D(x_points, y_points, z_points, c=z_points, cmap='hsv');

ax.set_xlabel('odometer')
ax.set_ylabel('year')
ax.set_zlabel('price')

plt.show()
```

```
# correlation matrix before the filtering
corr_matrix = train0.corr()
import seaborn as sns
import matplotlib.pyplot as plt

sns.heatmap(corr_matrix,
        xticklabels=corr_matrix.columns,
        yticklabels=corr_matrix.columns)
train0.describe(percentiles = [.1, .9])
# thanks to https://www.kaggle.com/vbmokin/used-cars-fe-eda-with-3d
# Filter: price (upper (90%) and lower (10%)), year (lower - 10%) and odometer (upper - 90%)
train0 = train0[((train0['price'] >= 1500) & (train0['price'] < 25000) & (train0['year'] >= 2001) &
(train0['odometer'] < 2000000))]
train0.info()
y = np.array(train0.price)
plt.subplot(131)
plt.plot(range(len(y)),y,'.');plt.ylabel('price');plt.xlabel('index');
plt.subplot(132)
sns.boxplot(y=train0.price)
#Thanks to https://towardsdatascience.com/an-easy-introduction-to-3d-plotting-with-matplotlib-
801561999725

fig = plt.figure(figsize=(10,10))
ax = plt.axes(projection="3d")

z_points = train0['price']
x_points = train0['odometer']
y_points = train0['year']
ax.scatter3D(x_points, y_points, z_points, c=z_points, cmap='hsv');

ax.set_xlabel('odometer')
ax.set_ylabel('year')
ax.set_zlabel('price')

target_name = 'price'
train_target0 = train0[target_name]
train0 = train0.drop([target_name], axis=1)

# Synthesis test0 from train0
train0, test0, train_target0, test_target0 = train_test_split(train0, train_target0, test_size=0.2,
random_state=0)

# 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)

#For models from Sklearn
scaler = StandardScaler()
```

```
train0 = pd.DataFrame(scaler.fit_transform(train0), columns = train0.columns)

train0.head(3)

# 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)
acc_train_r2 = []
acc_test_r2 = []
acc_train_d = []
acc_test_d = []
acc_train_rmse = []
acc_test_rmse = []
acc_train_score = []
acc_test_score = []

def acc_d(y_pred, y_meas):
    # Relative error between predicted y_pred and measured y_meas values
    return mean_absolute_error(y_pred, y_meas)*len(y_meas)/sum(abs(y_meas))

def acc_rmse(y_pred, y_meas):
    # RMSE between predicted y_pred and measured y_meas values
    return (mean_squared_error(y_pred, y_meas))**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
    global acc_train_score, acc_test_score

    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)

    if num == 8:
        # model as Booster (lgbm) and hasn't model.score
        acc_train_score_num = np.nan
        acc_test_score_num = np.nan
    else:
        acc_train_score_num = round(model.score(train, targetb) * 100, 2)
        acc_test_score_num = round(model.score(test, target_testb) * 100, 2)

    print('target = ', targetb[:5].values)
    print('ytrain = ', ytrain[:5])

    acc_train_r2_num = round(r2_score(targetb, ytrain) * 100, 2)
    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(ytrain, targetb) * 100, 2)
    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(ytrain, targetb) * 100, 2)
    print('acc(rmse) for train =', acc_train_rmse_num)
    acc_train_rmse.insert(num, acc_train_rmse_num)

    print('acc(score) for train =', acc_train_score_num)
    acc_train_score.insert(num, acc_train_score_num)

    print('target_test =', target_testb[:5].values)
    print('ytest =', ytest[:5])

    acc_test_r2_num = round(r2_score(ytest, target_testb) * 100, 2)
    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(ytest, target_testb) * 100, 2)
    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(ytest, target_testb) * 100, 2)
    print('acc(rmse) for test =', acc_test_rmse_num)
    acc_test_rmse.insert(num, acc_test_rmse_num)

    print('acc(score) for test =', acc_test_score_num)
    acc_test_score.insert(num, acc_test_score_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
    global acc_train_score, acc_test_score
    ytrain = model.predict(train)
    ytest = model.predict(test)
    acc_train_score_num = round(model.score(train, target) * 100, 2)
    acc_test_score_num = round(model.score(test, target_test) * 100, 2)
    print('target = ', target[:5].values)
    print('ytrain = ', ytrain[:5])
    acc_train_r2_num = round(r2_score(ytrain, target) * 100, 2)
    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(ytrain, target) * 100, 2)
    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(ytrain, target) * 100, 2)
    print('acc(rmse) for train =', acc_train_rmse_num)
    acc_train_rmse.insert(num, acc_train_rmse_num)
    print('acc(score) for train =', acc_train_score_num)
    acc_train_score.insert(num, acc_train_score_num)
```

```
    print('target_test =', target_test[:5].values)
    print('ytest =', ytest[:5])
    acc_test_r2_num = round(r2_score(ytest, target_test) * 100, 2)
    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(ytest, target_test) * 100, 2)
    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(ytest, target_test) * 100, 2)
    print('acc(rmse) for test =', acc_test_rmse_num)
    acc_test_rmse.insert(num, acc_test_rmse_num)
    print('acc(score) for test =', acc_test_score_num)
    acc_test_score.insert(num, acc_test_score_num)

# Linear Regression

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

# 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(3,mlp_GS,train,test)
# Stochastic Gradient Descent

sgd = SGDRegressor()
```

```
sgd.fit(train, target)
acc_model(4,sgd,train,test)

# Decision Tree Regression

decision_tree = DecisionTreeRegressor()
decision_tree.fit(train, target)
acc_model(5,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)
# print(random_forest.best_params_)
acc_model(6,random_forest,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)
#%% split training set to validation set
Xtrain, Xval, Ztrain, Zval = train_test_split(trainb, targetb, test_size=0.2, random_state=0)
train_set = lgb.Dataset(Xtrain, Ztrain, silent=False)
valid_set = lgb.Dataset(Xval, Zval, silent=False)

params = {
        'boosting_type':'gbdt',
        'objective': 'regression',
        'num_leaves': 31,
        'learning_rate': 0.01,
        '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 = lgb.train(params, train_set = train_set, num_boost_round=10000,
```

```
                early_stopping_rounds=50,verbose_eval=10, 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)
lgb.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

# 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)
```

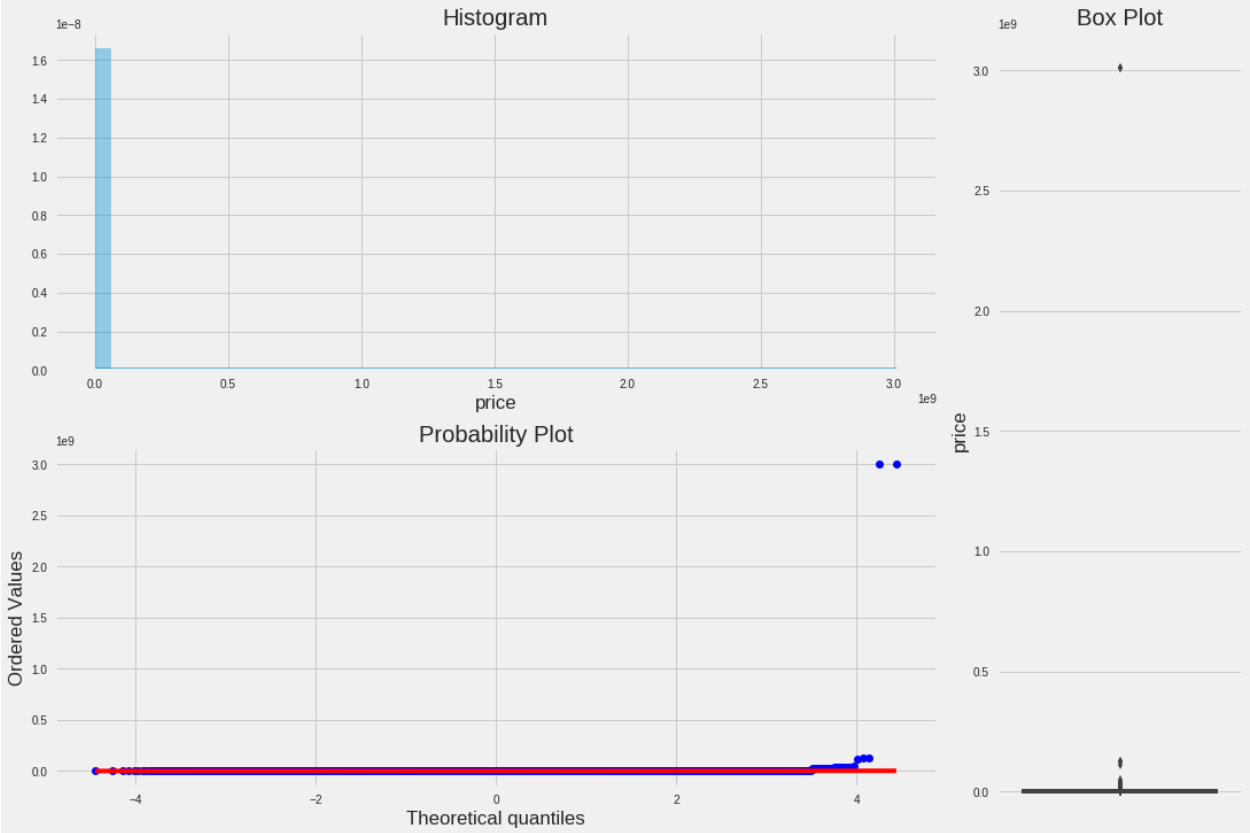
```
Voting_Reg = VotingRegressor(estimators=[('lin', linreg), ('ridge', ridge), ('sgd', sgd)])
Voting_Reg.fit(train, target)
acc_model(14,Voting_Reg,train,test)
models = ['Linear Regression',
          'Support Vector Machines',
          'Linear SVR',
          'MLPRegressor',
          'Stochastic Gradient Decent',
          'Decision Tree Regressor',
          'Random Forest',
          'XGB',
          'LGBM',
          'GradientBoostingRegressor',
          'RidgeRegressor',
          'BaggingRegressor',
          'ExtraTreesRegressor',
          'AdaBoostRegressor',
          'VotingRegressor',]

models = pd.DataFrame({
    'Model': models,
   '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'], ascending=False)

print('Prediction accuracy for models by relative error - d_test')
models.sort_values(by=['d_test', 'd_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('graph.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('graph.png')
plt.show()
```

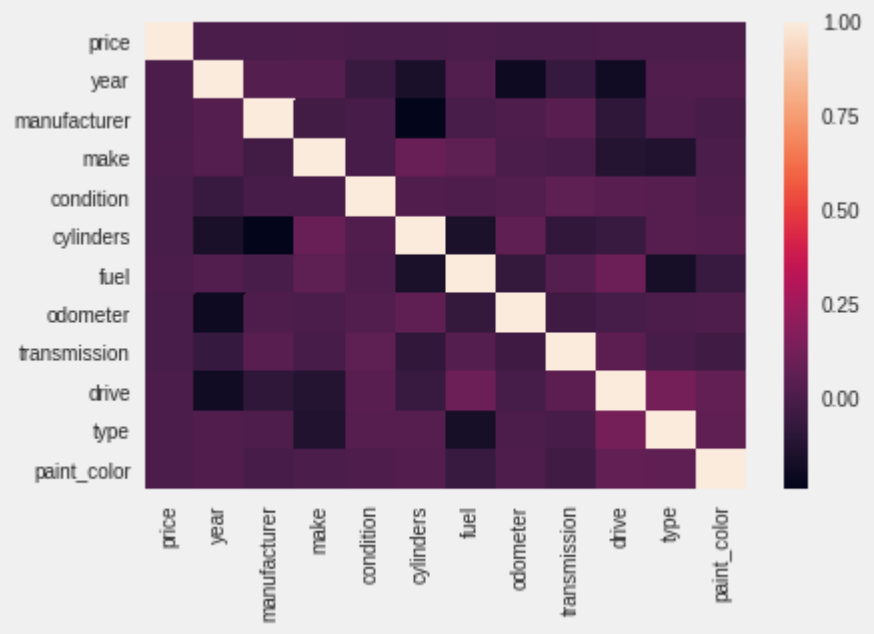
Додаток В

Графічна частина

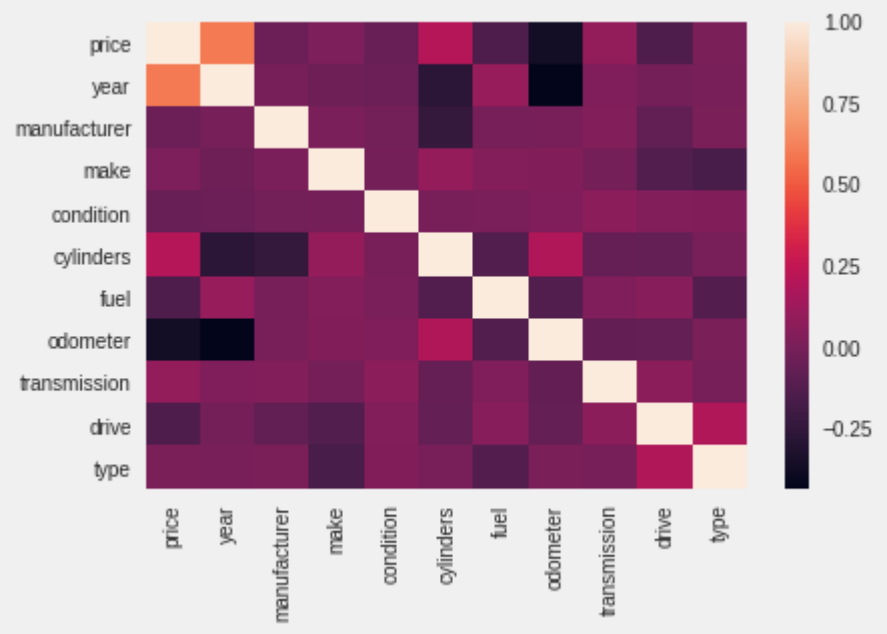
Графіки розподілу ціни автомобіля



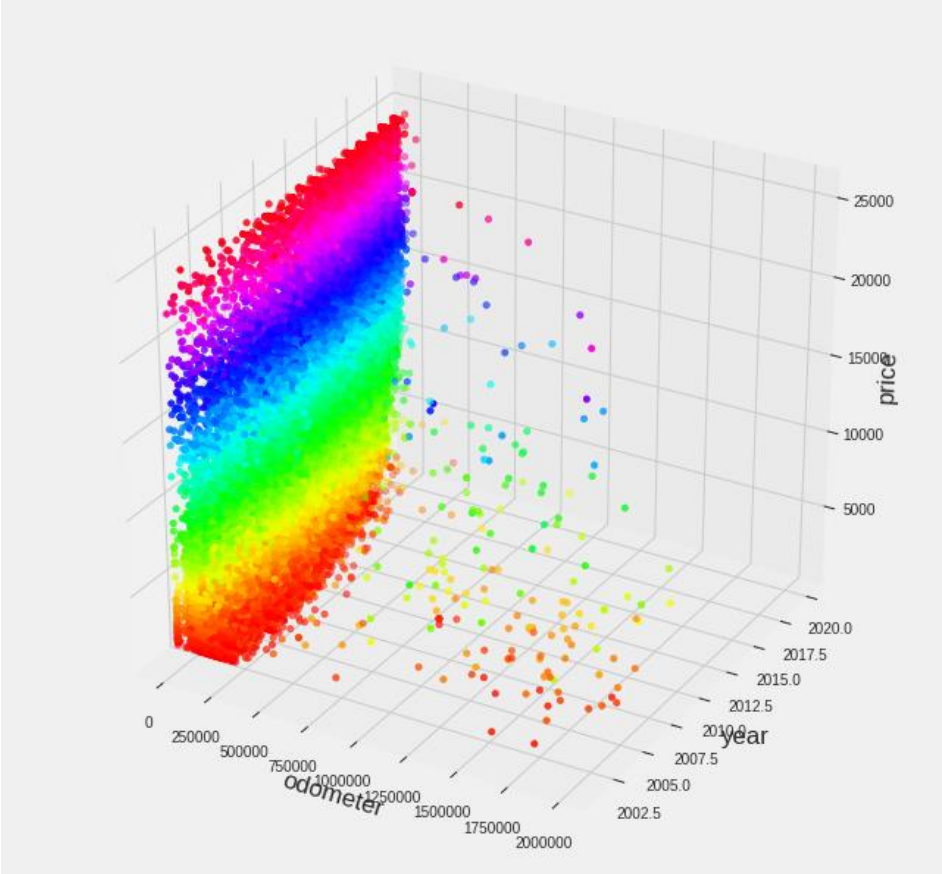
Матриця кореляції між ознаками набору даних, зображена у вигляді теплової карти



Матриця кореляції між ознаками набору даних після фільтрування, зображена у вигляді теплової карти



Графік співвідношення значень ціни, пробігу та року випуску, після застосування фільтрів



Ранжування за R^2-критерієм результатів передбачення моделей, натренованих на американських даних

	Model	r2_train	r2_test	d_train	d_test	rmse_train	rmse_test
8	LGBM	91.06	86.12	12.53	14.56	179,382.56	208,609.17
12	ExtraTreesRegressor	99.95	84.19	0.13	13.73	12,790.79	227,775.52
6	Random Forest	97.17	84.11	5.82	14.51	97,206.65	225,093.18
11	BaggingRegressor	97.19	84.10	5.80	14.50	96,970.11	224,957.57
7	XGB	88.33	83.54	14.57	15.84	204,923.27	223,658.23
5	Decision Tree Regressor	99.95	77.11	0.13	16.98	12,789.23	288,694.89
3	MLPRegressor	67.37	68.04	21.65	21.74	297,952.31	297,286.15
9	GradientBoostingRegressor	62.21	62.57	21.81	21.97	296,603.86	297,399.42
14	VotingRegressor	28.82	30.39	29.35	29.36	389,901.74	387,985.65
0	Linear Regression	26.95	28.58	29.45	29.45	389,782.46	387,856.11
10	RidgeRegressor	26.92	28.56	29.45	29.45	389,782.47	387,856.51
4	Stochastic Gradient Decent	22.25	23.99	29.58	29.56	390,531.29	388,697.83
2	Linear SVR	11.13	12.93	29.68	29.74	409,624.83	407,699.79
13	AdaBoostRegressor	-102.47	-102.86	34.83	35.04	414,316.82	416,350.42
1	Support Vector Machines	-1,017.94	-1,020.26	38.45	38.60	508,854.61	510,915.06

Ранжування за R^2-критерієм результатів передбачення моделей, натренованих на українських даних

	Model	r2_train	r2_test	d_train	d_test	rmse_train	rmse_test
4	LGBM	92.51	85.55	31.92	36.05	38,031.21	42,672.70
5	GradientBoostingRegressor	94.14	85.45	6.45	35.26	10,965.31	46,303.93
6	BaggingRegressor	87.26	83.61	14.22	35.21	20,641.95	46,322.15
3	XGB	91.98	83.12	29.38	35.23	35,710.24	42,575.48
2	Random Forest	92.35	81.38	14.33	34.85	20,998.43	46,093.82
1	Decision Tree Regressor	84.16	78.43	1.40	38.18	8,257.61	57,887.24
0	MLPRegressor	60.75	57.71	53.76	53.12	53,822.12	53,586.36
7	ExtraTreesRegressor	27.75	25.63	1.40	35.70	8,257.72	49,729.96

Код реалізації навчання моделі LGBM

```
import lightgbm as lgb

#%% split training set to validation set
Xtrain, Xval, Ztrain, Zval = train_test_split(trainb, targetb, test_size=0.2, random_state=0)
train_set = lgb.Dataset(Xtrain, Ztrain, silent=False)
valid_set = lgb.Dataset(Xval, Zval, silent=False)

params = {
        'boosting_type':'gbdt',
        'objective': 'regression',
        'num_leaves': 31,
        'learning_rate': 0.01,
        '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 = lgb.train(params, train_set = train_set, num_boost_round=10000,
                   early_stopping_rounds=50,verbose_eval=10, valid_sets=valid_set)
```

Код реалізації навчання моделі XGBoost Regressor

```
import xgboost as xgb


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)
```