КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра технологій управління

Спеціальність 122 – Комп'ютерні науки,
освітня програма «Інформаційна аналітика та впливи»

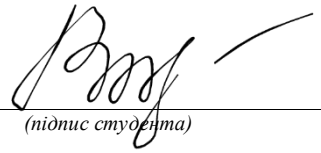
КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему:

«Інформаційне забезпечення прогнозування цін на автотранспортні засоби у сфері роздрібної торгівлі методами машинного навчання»

Студентки 2-го курсу групи ІАВ-21

Вікторія ЦИКУН
(ім'я, прізвище)



(підпис студента)

Науковий керівник:

д.т.н., професор
(науковий ступінь, вчене звання)

Юлія ХЛЕВНА
(ім'я, прізвище)

(дата) *(підпис)*

Попередній захист:

(Висновок: «До захисту в Екзаменаційній комісії»)

Завідувач кафедри
технологій управління, проф. _____ *Віктор МОРОЗОВ* _____
(підпис) *(ім'я, прізвище)* *(дата)*

Київ – 2022

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій

Кафедра технологій управління
Освітньо-кваліфікаційний рівень Магістр
Спеціальність 122 - Комп'ютерні науки
Освітня програма Інформаційна аналітика та впливи

ЗАТВЕРДЖУЮ
Завідувач кафедри
професор Віктор МОРОЗОВ
«____» _____20___року

З А В Д А Н Н Я
НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Студент: **Вікторія ЦИКУН**

Група: ІАВ-21

1. Тема кваліфікаційної роботи
Інформаційне забезпечення прогнозування цін на автотранспортні засоби у сфері роздрібної торгівлі методами машинного навчання

Затверджена протоколом від «17» листопада 2021 р. № 4.

2. Строк подання студентом готової роботи – «18» травня 2022 р.

3. Цільова установка та вихідні дані до роботи інформаційне забезпечення прогнозування цін на автотранспортні засоби у сфері роздрібної торгівлі, що побудоване за допомогою мови програмування Python і реалізоване за допомогою фреймворку для веб-додатків Flask на основі даних, отриманих з веб-ресурсів

4. Зміст роботи визначення предметної області дослідження, аналіз принципів формування цін на автотранспортні засоби, аналіз методів прогнозування цін на автотранспортні засоби, визначення математичного апарату прогнозування цін на автотранспортні засоби, реалізація математичного апарату прогнозування цін на автотраспортні засоби у комп'ютерних системах, формування набору даних, первинний аналіз набору даних, підготовка даних для моделювання та прогнозування, застосування моделей лінійної регресії, регресії опорних векторів, градієнтного бустингу, побудова моделей прогнозування цін на автотранспортні засоби з використання інформаційного забезпечення, порівняння отриманих результатів побудови моделей, вибір інструментів для реалізації інформаційного забезпечення прогнозування цін на автотранспортні засоби, розробка інформаційного забезпечення прогнозування цін на автотранспортні засоби, практичне використання інформаційного забезпечення прогнозування цін на автотранспортні засоби

5. Перелік графічного матеріалу (слайдів) 30 рисунків, 6 формул, 6 додатків, 19 слайдів презентації доповіді

6. Календарний план виконання роботи:

№ з/п	Назва частин роботи	%	Виконання роботи	
			За планом	Фактично
1.	Вибір теми кваліфікаційної роботи	3	01.10.2021	01.10.2021
2.	Протокол кафедри ТУ про затвердження тем кваліфікаційних робіт та призначення наукових керівників	1	09.11.2021	09.11.2021
3.	Формування переліку нормативних матеріалів, літератури з проблематики кваліфікаційної роботи	10	08.01.2022	08.01.2022
4.	Складання розгорнутого плану кваліфікаційної роботи	4	18.01.2022	17.01.2022
5.	Ознайомлення наукового керівника з розгорнутим планом кваліфікаційної роботи. Внесення змін	4	19.01.2022 – 20.01.2022	18.01.2022
6.	Підготовка розділу 1 «Теоретичні основи та особливості ціноутворення на автотранспортні засоби у сфері роздрібної торгівлі»	10	12.02.2022	12.02.2022
7.	Підготовка розділу 2 «Методи машинного навчання для проблеми прогнозування цін на автотранспортні засоби»	11	08.03.2022	08.03.2022
8.	Підготовка розділу 3 «Побудова моделей прогнозування цін на автотранспортні засоби та їх валідація»	23	01.04.2022	03.04.2022
9.	Підготовка розділу 4 «Розробка інформаційного забезпечення прогнозування цін на автотранспортні засоби»	15	20.04.2022	20.04.2022
10.	Оформлення кваліфікаційної роботи. Підготовка висновків і пропозицій	10	29.04.2022	29.04.2022
11.	Передача кваліфікаційної роботи науковому керівникові	2	04.05.2022	04.05.2022
12.	Передача кваліфікаційної роботи рецензенту для рецензування	2	05.05.2022	05.05.2022
13.	Попередній захист кваліфікаційної роботи	5	11.05.2022	11.05.2022

Дата видачі завдання «17» листопада 2021 р.

Керівник роботи д.т.н., професор Юлія ХЛЕВНА
(посада, ім'я, прізвище)

_____
(підпис)

Завдання прийняв до виконання здобувач освіти групи ІАВ-21

Вікторія ЦИКУН
(ім'я, прізвище)

_____
(підпис)

ЗМІСТ

АНОТАЦІЯ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра технологій управління
Спеціальність 122 - Комп'ютерні науки,
освітня програма "Інформаційна аналітика та впливи"

Дипломна робота магістра Цикун Вікторії Андріївни.

Тема роботи – «Інформаційне забезпечення прогнозування цін на автотранспортні засоби у сфері роздрібної торгівлі методами машинного навчання».

Мета дипломної роботи магістра – аналіз та розробка моделі прогнозування цін на автотранспортні засоби у сфері роздрібної торгівлі методами машинного навчання, створення інформаційного забезпечення прогнозування цін на основі побудованої моделі.

Об'єкт дослідження – процеси ціноутворення на автотранспортні засоби у сфері роздрібної торгівлі.

Предмет дослідження – інформаційні засоби, інструменти, методи і моделі управління даними і процесами при аналізі та прогнозуванні ціни на вживані автотранспортні засоби.

Наукова новизна роботи – створено модель прогнозування цін на автотранспортні засоби у сфері роздрібної торгівлі, яка на відміну від існуючих базується на даних, що сформовані з веб-ресурсів українського ринку.

Практична значимість у тому, що реалізовано програмну імплементацію технології прогнозування ціни на вживані автотранспортні засоби українського ринку, що базується на мові програмування Python для безпосереднього використання у бізнесі.

У роботі досліджуються існуючий математичний інструментарій та підходи до вирішення задач регресії. На основі отриманого математичного

апарату побудована програмна реалізація технології прогнозування ціни на вживані автомобілі.

Дипломна робота складається зі вступу, основної частини, яка включає чотири розділи, висновків та списку використаних джерел. Всього налічує 90 сторінок та перелік посилань з 39 джерел на 4 сторінках.

Ключові слова: роздрібна торгівля, ціни на вживані автотранспортні засоби, машинне навчання, лінійна регресія, метод опорних векторів, градієнтний бустинг, XGBoost, LightGBM.

ВСТУП

Актуальність дослідження. Історія роздрібної торгівлі почалася ще дуже давно, а роздрібна торгівля автотранспортними засобами – після початку виробництва перших авто. Наразі роздрібна торгівля автомобілями має сильні позиції на ринку в принципі. Не можна заперечувати, що майже кожна родина хоча б раз в житті продавала своє авто. За даними Edmunds.com, інтернет-сайту автомобільних оглядів, лише в Сполучених Штатах щорічно продається близько 40 мільйонів вживаних автомобілів [16]. За даними українських сайтів, 80% від усіх нових регістрацій автомобілів в Україні складали вживані машини [17]. А у грудні 2018 року ринок б/в автомобілів в Україні в 3 рази перевищив продажі нових авто [18]. Незважаючи на кризу ковідного періоду, за 2021 рік український ринок б/в автомобілів виріс на 18% [19].

Сфера постійно розвивається, то ж зрозуміло, що вона потребує розвиток технологій оцінювання вартості автомобілів. Наразі існують декілька способів, за якими експерти формують ціну авто. Зазвичай це певний відсоток від початкової вартості автомобіля, зважаючи на пройдений час від випуску та стан машини. Проте, для пересічного громадянина, який хоче продати свою автівку, ця оцінка є проблематичною, адже він не є експертом та не може коректно оцінити вартість автомобіля.

Також не можна не враховувати можливу недоброчесність експертів з оцінки чи дилерів, які роблять націнку. Виникає потреба в експертній системі, що буде надавати розраховану ціну вживаного автомобіля, аналізуючи його характеристики (пробіг, об’єм двигуна, тип кузову тощо).

Схожі системи та алгоритми вже існують, проте вони розраховані в більшості випадків на американський ринок та не мають майже нічого спільного з українськими реаліями. Тож майбутнє інформаційне забезпечення прогнозування цін на автомобілі має базуватися на даних українського ринку.

Створена експертна система буде сприяти розвитку українського автомобільного роздрібного ринку, допоможе ефективно продавати вживані

автомобілі, надаючи рекомендовану ціну, зважаючи лише на характеристики транспортного засобу.

Мета роботи - аналіз та розробка моделі прогнозування цін на автотранспортні засоби у сфері роздрібної торгівлі методами машинного навчання, створення інформаційного забезпечення прогнозування цін на основі побудованої моделі. Для реалізації поставленої мети необхідним є вирішення таких **завдань**:

— провести аналітику ціноутворення та виокремити основні показники, що впливають на ціну вживаного автомобіля;

— виокремити методи прогнозування, які можуть бути використані для моделювання цін на автотранспортні засоби;

— зібрати актуальні дані про вживані автомобілі з українських веб-ресурсів;

— розробити декілька моделей прогнозування цін на вживані автотранспортні засоби з використанням методів машинного навчання, порівняти їх та, базуючись на отриманих результатах, обрати найкращу;

— розробити програмну реалізацію технології прогнозування ціни на вживані автомобілі та оцінити практичне застосування створеного продукту у бізнесі.

Об’єктом дослідження є процеси ціноутворення на автотранспортні засоби у сфері роздрібної торгівлі.

Предметом дослідження є інформаційні засоби, інструменти, методи і моделі управління даними і процесами при аналізі та прогнозуванні ціни на вживані автотранспортні засоби.

Використаними у роботі **методами досліджень** є емпіричний метод дослідження, кореляційний аналіз, математичний апарат та моделі для вирішення задач регресії: лінійна регресія, метод опорних векторів, регресія дерева рішень з градієнтним бустингом, статистичні методи та інструменти для обробки та аналізу даних, інструменти візуалізаці даних, контейнеризація додатку.

Для вирішення завдань даної роботи використано мову програмування, обробки та аналізу даних Python.

Наукова новизна одержаних результатів

Розроблено модель прогнозування цін на автотранспортні засоби у сфері роздрібної торгівлі методами машинного навчання саме для українського ринку. Запропонована модель базується на базі даних, що була зібрана з українських веб-ресурсів по продажу вживаних автотранспортних засобів.

Практична значимість роботи полягає в тому, що реалізовано програмну імплементацію технології прогнозування ціни на вживані автотранспортні засоби, що базується на мові програмування Python, для безпосереднього використання на веб-ресурсах розміщення оголошень продажу автотранспортних засобів та бізнесу.

Апробація результатів роботи. Основні наукові положення, висновки і результати магістерської кваліфікаційної роботи знайшли відображення у тезах доповіді на конференції VII INTERNATIONAL CONFERENCE Information Technology and Interactions (Satellite).

Структура та обсяг роботи. Магістерська робота складається зі вступу, 4 розділів, висновків, списку використаних джерел з 39 найменувань та додатків. Загальний обсяг курсової становить 90 сторінок, із них 56 сторінок основного тексту, який містить 30 рисунків.

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ ТА ОСОБЛИВОСТІ ЦІНОУТВОРЕННЯ НА АВТОТРАНСПОРТНІ ЗАСОБИ У СФЕРІ РОЗДРІБНОЇ ТОРГІВЛІ

1.1 Визначення предметної області дослідження

Роздрібна торгівля - це процес продажу споживчих товарів або послуг клієнтам через кілька каналів розподілу з метою отримання прибутку. Роздрібні торговці задовольняють попит, виявлений через ланцюжок поставок. Термін «роздрібний торговець» зазвичай застосовується, коли постачальник послуг виконує невеликі замовлення багатьох осіб, які є кінцевими користувачами, а не великі замовлення невеликого числа оптових, корпоративних або державних клієнтів. Під покупками зазвичай розуміється покупка товарів. Іноді це робиться для отримання кінцевих товарів, включаючи предмети першої необхідності, такі як їжа і одяг; іноді це відбувається як розважальна діяльність.

У роздрібних ринків і магазинів дуже давня історія, що йде корінням в глибоку старовину. Деякі з перших роздрібних торговців були мандрівними розносниками. Протягом століть роздрібні магазини перетворилися з «грубих кіосків» в вишукані торгові центри сучасної епохи.

Більшість сучасних роздрібних продавців зазвичай приймають безліч рішень на стратегічному рівні, включаючи тип магазину, ринок, який буде обслуговуватися, оптимальний асортимент продукції, обслуговування клієнтів, допоміжні послуги та загальне положення магазину на ринку. Після розробки стратегічного плану роздрібної торгівлі роздрібні торговці розробляють структуру роздрібної торгівлі, що включає товар, ціну, місце, просування, персонал і презентацію. В епоху цифрових технологій все більше число роздрібних продавців прагнуть вийти на ширші ринки, продаючи товари через кілька каналів, включаючи як звичайні, так і роздрібні продажі через Інтернет. Цифрові технології також змінюють спосіб оплати споживачами товарів і послуг. Послуги підтримки роздрібної торгівлі можуть також включати надання

кредиту, послуги доставки, консультаційні послуги і ряд інших допоміжних послуг [21].

Дистрибуція автомобілів - це бізнес, який продає нові або старі автомобілі на роздрібному рівні [22]. Також можливий такий варіант, що продавцем виступає власник автомобіля, який продає свою стару машину використовуючи сайти та інші ринки збуту.

Роздрібна ціна - це те, скільки потенційний покупець може розраховувати заплатити за автомобіль при покупці [23].

1.2 Аналіз принципів формування ціни на автотранспортний засіб

Ціноутворення на вторинному ринку автомобілів - тема, яка цікавить як продавців, так і покупців транспортних засобів. Очевидно, що вартість уживаних авто знижується прямо пропорційно терміну їх експлуатації. Однак це далеко не єдиний фактор, який надає об'єктивний вплив на ціну продаваного транспортного засобу.

Кінцеву ціну реалізованого автомобіля встановлює власник, виходячи з власних уявлень про вартість свого транспортного засобу. Природньо, що пропозиції із завищеними цінами не викличуть інтересу у потенційних покупців, тому важливо максимально об'єктивно оцінити стан машини і, в відповідність з цим, виставити вартість. При цьому необхідно брати до уваги такі фактори:

— Рік випуску: насамперед на вартість авто впливає амортизаційний чинник. Чим старіше транспортний засіб, тим нижче його ціна на вторинному ринку;

— Технічний стан і стан кузова: автомобілі з технічними неполадками, подряпинами та пошкодженнями кузовних частин апріорі не можуть продаватися за середньоринковими цінами. Однак в процесі огляду потенційними покупцями реалізованого автомобіля багато нюансів залишаються непоміченими, тому іноді досить візуально підлатати транспортний засіб, щоб вигідно його продати;

— Пробіг: сьогодні власники транспортних засобів значно менше спантеличені величиною пробігу, адже автомобіль, перш за все, купується для активного пересування. Однак при інших рівних умовах продавцю машини з великим пробігом все ж доведеться скинути деяку грошову суму, щоб не втратити покупця;

— Особливості комплектації: зрозуміло, що кожна додаткова особливість в автомобілі автоматично «накидає» йому вартість. Наприклад, наявність парктроника або чотирьохзонного клімат-контролю гарантує деяку надбавку до середньоринкової вартості. Але існують також чинники, які здатні знизити ціну реалізованого транспортного засобу. Зокрема, велику роль в популярності авто серед потенційних покупців грає колір салон і кузова. Світлий салон менш затребуваний, тому що вимагає акуратної експлуатації і додаткового догляду. Нестандартні кольори кузова також можуть стати причиною зниження попиту на транспортний засіб. Не варто очікувати ажіотажу навколо продажу представницьких автомобілів з базовою комплектацією, моноприводних кросоверів і «легковиків» з двигуном, що працюють на дизельному паливі;

— Час продажу: як не дивно, навіть сезон, в який авто виставляється на реалізацію, впливає на попит і, відповідно, вартість. Наприклад, власникам кабріолета, які продають свій стильний автомобіль взимку, розраховувати на підвищений інтерес не доводиться. Краще почекати з продажем до настання весни. Стежити необхідно і за випуском оновлених моделей - в очікуванні їх релізу вартість попередніх версій вже помітно знижується;

— Затребуваність моделі на ринку: не всі марки і не всі моделі автомобілів однаково затребувані, і цей факт необхідно враховувати при встановленні ціни. Хороший приклад - автомобіль Hyundai Genesis. Будучи представником преміум-класу, він спочатку має досить високу ціну, за яку можна придбати авто більш престижної марки. Чи не кожен автолюбитель готовий піти на такий крок, особливо з огляду на ймовірні складності в реалізації транспортного засобу на вторинному ринку;

— Регіон продажу: вивчаючи пропозиції на ринку старих автомобілів, ви, напевно, помітили, що в кожному регіоні країни встановлюються свої середньоринкові ціни. Як правило, в столиці ціни значно нижче, ніж, припустимо, в Херсоні. На Далекому Сході вигідніше купувати японські автомобілі, а в Україні - європейські. Також слід враховувати затребуваність конкретних категорій транспортних засобів в окремих населених пунктах. Представницькі автомобілі не користуються попитом в малонаселених містах і селах, де більший інтерес викликають позашляховики і кросовери;

— Сервісна історія: технічне обслуговування у офіційного дилера проходять далеко не всі автомобілі - їх власники просто вважають, що дана послуга не коштує витрачених грошей. Можливо, такий підхід і має раціональне зерно, але при продажу порожня сервісна книжка буде коштувати продавцю певну помітно знижаючу ціну авто суму [20].

Перераховані вище фактори об'єктивно впливають на формування вартості автомобілів, реалізованих на вторинному ринку.

1.3 Аналіз методів прогнозування цін на вживані автотранспортні засоби в Україні

Для оцінки стану проблеми на українському ринку, проаналізуємо та дослідимо поточну ситуацію, а саме наявність та розповсюдженість програм чи систем, що вивчають, розраховують та надають рекомендації щодо ціноутворення на вживані автотранспортні засоби в Україні.

Одна з найбільших інтернет-платформ по розташуванню оголошень на продаж нових та вживаних автомобілів AUTO.RIA пропонує розроблений калькулятор розрахунку вартості автомобіля, для того, щоб дізнатися ринкову вартість автомобіля онлайн [37]. Цей калькулятор дозволяє заповнити параметри, такі як тип транспорту, тип кузова, марку і модель, пробіг, рік, тип кпп і палива. На основі введених параметрів онлайн калькулятор надає інформацію щодо середньої ціни на вказаний автомобіль (рис. 1.1).

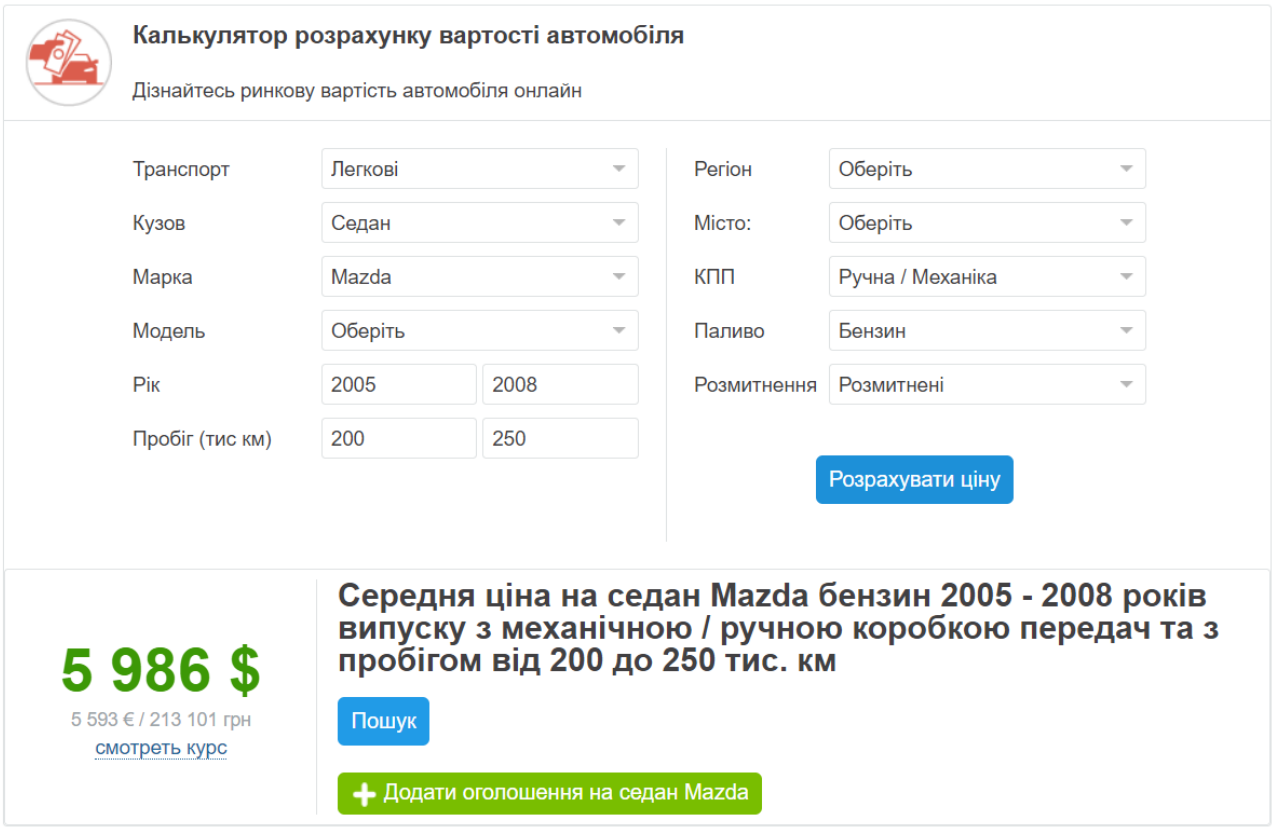


Рисунок 1.1 – Онлайн калькулятор з розрахунку середньої ціни на сайті AUTO.RIA

Як вказують самі розробники, калькулятор вартості розраховує середньоринкову ціну на підставі ціни, зазначеної в оголошеннях конкретних автомобілів на AUTO.RIA як по всій Україні, так і конкретного регіону. Тобто даний калькулятор не прогнозує ціну за вказаними параметрами, а лише фільтрує оголошення розміщені на AUTO.RIA за введеними параметрами в конкретний момент часу та розраховує середнє арифметичне цін отриманих автомобілів. Даний підхід не вирішує проблеми прогнозування, адже за відсутності наявних активних оголошень, що відповідають введеним параметрам, на платформі цей калькулятор не надасть ніякої інформації. Також, якщо таких оголошень буде мала кількість, отримане значення ціни буде упередженим.

Іншим знайденим калькулятором для розрухунку ціни автомобіля, став калькулятор на сайті Trade-in, що надає послуги оцінки, пошуку та реєстрації автомобілів [38]. Після детального аналізу, було виявлено, що цей калькулятор

не має власної формули чи моделі, розробники просто використовують вже наданий платформою AUTO.RIA сервіс. Тобто, калькулятор на сайті Trade-in повністю залежить від сайту AUTO.RIA та даних оголошень, розміщених на ньому.

У кінці 2015 року на сайті Міністерства економічного розвитку та торгівлі з'явився онлайн-калькулятор, який розраховує середньоринкову вартість автомобіля [39]. Таким чином, цей сервіс повинен допомогти власникам автомобілів дізнатися, скільки оцінюють їх транспортний засіб в уряді.

В алгоритмі розрахунку закладено кілька змінних: тип транспортного засобу (легкова машина, мотоцикл або мопед), марка, модель, рік випуску та пробіг. При цьому на сайті не вказано жодних подробиць про базову ціну або формулу, що знаходиться в основі розрахунку вартості авто з урахуванням змін, введених користувачем. Відсутній і сам опис калькулятора.

Відсутність вказаного алгоритму чи формули не дає можливості використовувати цей калькулятор у бізнесі, а те, що даний калькулятор було розроблено для розрахунку середньоринкової вартості транспортних засобів (легкових автомобілів, мотоциклів, мопедів) для цілей оподаткування операцій з продажу або обміну об'єктів рухомого майна не дозволяє покладатися на цю ціну під час продажу на роздрібному ринку.

Отже, проаналізувавши та дослідивши стан проблеми в Україні, було виявлено, що наразі український ринок не має працюючої системи оцінки вартості автомобіля за введеними параметрами, що була б доступна не тільки бізнесу чи уряду, а й звичайним громадянинам України.

1.3 Виявлення показників для формування ціни в моделях машинного навчання

Для передбачення вартості автомобіля за допомогою методів машинного навчання необхідно попередньо визначити показники, що мають відношення до прогнозування ціни. Літературні джерела повідомляють, що необхідно враховувати такі атрибути [24]:

— Виробник/клас: марка машини – це важливий момент у формуванні вартості авто при продажу. Кожна марка займає свій сегмент на ринку автомобілів. Також важливий клас авто: автомобіль представницького класу буде коштувати більше автомобіля економ класу;

— Тип трансмісії (КПП): автомобіль з автоматичною коробкою перемикання передач коштує більше ніж автомобіль з механічною КПП;

— Колір: автомобілі зі затребуваними кольорами (чорний, сріблястий, білий) коштують більше ніж авто інших кольорів;

— Пробіг: показник одометра як ніщо інше впливає на ціну транспортного засобу. Чим вище показник одометра, тим нижче ціна автомобіля;

— Рік випуску/вік: чим більше вік автомобіля тим менше він коштує;

— Тип двигуна/палива: дизельні автомобілі дорожче бензинових;

— Об'єм двигуна: чим більше об’єм тим більше ціна;

— Тип приводу: повний привід завжди збільшує ціну на машину, на відміну від переднього;

— Тип кузову: в середньому універсал та седан коштують менше ніж кабріолет та купе.

РОЗДІЛ 2 МЕТОДИ МАШИННОГО НАВЧАННЯ ДЛЯ ПРОБЛЕМИ ПРОГНОЗУВАННЯ ЦІН НА АВТОТРАНСПОРТНІ ЗАСОБИ

2.1 Задачі регресії та прогнозування

Регресійний аналіз – це статистичний метод для дослідження та моделювання зв’язку між змінними. Мета регресії полягає в тому, щоб визначити основну природу зв’язку між X та Y. Вона вирішує питання щодо того, чи X та Y безпосередньо пов’язані, оскільки обидва рухаються в одному напрямку, чи вони опосередковано пов’язані так, що одне збільшується, коли інше зменшується. Це не означає, що регресія пояснює, чому залежна змінна змінюється. Регресія не може визначити причинно-наслідкові зв'язки або змоделювати причинно-наслідковий зв'язок [4].

Застосування регресії численні і зустрічаються майже в кожній галузі, включаючи інженерію, фізичні та хімічні науки. Фактично, регресійний аналіз можна вважати найбільш широко використовуваним статистичним методом.

Регресійні моделі використовуються для кількох цілей, зокрема для опису даних, оцінки параметрів, прогнозу та оцінки, контролю.

Інженери та вчені часто використовують рівняння для узагальнення або опису набору даних. У розробці таких рівнянь допомагає регресійний аналіз. Наприклад, ми можемо зібрати значну кількість даних про час доставки та обсяги доставки, і регресійна модель, ймовірно, буде набагато зручнішим і кориснішим підсумком цих даних, ніж таблиця чи навіть графік.

Регресійні моделі також можна використовувати для цілей контролю. Наприклад, інженер-хімік міг би використати регресійний аналіз, щоб розробити модель, яка зв’язує міцність паперу на розрив із концентрацією листяних порід у целюлозі. Потім це рівняння можна використовувати для контролю міцності до відповідних значень, змінюючи рівень концентрації листяної деревини. Коли регресія використовується для цілей контролю, важливо, щоб змінні були пов’язані причинно-наслідковим чином. Проте варто зауважити, що причинно-наслідковий зв’язок може не бути необхідним, якщо модель буде

використовуватися лише для передбачення. У цьому випадку необхідно лише, щоб зв’язки, які існували в вихідних даних, використаних для побудови рівняння регресії, залишалися дійсними [3].

Специфікація моделі є критично важливою в регресійному аналізі, метою якого є оцінка відносного значення індивідуальних змінних-предикторів при прогнозуванні. Для того, щоб визначити, які змінні найбільше сприяють прогнозуванню відповіді, потрібно спочатку переконатися, що всі відповідні змінні містяться в базі даних і що рівняння прогнозування визначено у правильному функціональному вигляді для всіх змінних. Недотримання жодної з цих двох вимог може призвести до неправильних висновків.

Регресійний аналіз, який проводиться для того, щоб забезпечити хороші оцінки параметрів моделі, як правило, є найскладнішим у виконанні. Модель повинна бути не тільки правильно визначена, а прогноз - точним, але й база даних повинна забезпечувати хорошу оцінку. Певні характеристики бази даних (наприклад, мультиколінеарність) створюють проблеми при оцінці параметрів.

Обмеження бази даних і неможливість вимірювання всіх змінних-предикторів, які є релевантними для дослідження, обмежують використання рівнянь прогнозу. Це особливо вірно, коли оцінка параметрів є основною метою дослідження. Використання специфікації моделі вимагає обгрунтованих оцінок параметрів, але часто потрібні лише грубі порівняння важливості предикторів. Наприклад, у дослідженні рівня заробітної плати може не знадобитися мати точні оцінки параметрів - лише знання про те, які змінні є важливими предикторами, може бути достатнім [5].

Подібно до того, як існує багато корисних застосувань регресійного аналізу, деколи підібрані регресійні моделі використовуються невідповідно. Ці зловживання не завжди очевидні для аналітика даних. Деякі неправильні застосування взагалі є не результатом аналізу даних, а пов’язані з необґрунтованими або надмірно перебільшеними твердженнями щодо встановлених моделей.

2.2 Формування математичного апарату прогнозування цін на автотранспортні засоби

2.2.1 Лінійна регресія як базовий метод для вирішення проблеми прогнозування

Лінійна регресія - це основна форма регресійного аналізу. Передбачається, що існує лінійний зв'язок між залежною змінною і предиктором(ами). У регресії ми намагаємося обчислити лінію найкращої відповідності, яка описує взаємозв'язок між предикторами і прогнозуючою / залежною змінною [1].

Найпростіша форма рівняння регресії з однією залежною й однією незалежною змінною визначається формулою

$$y = c + b * x, \tag{2.1}$$

де y - оцінка залежної змінної;

c - константа;

b - коефіцієнт регресії;

x - оцінка незалежної змінної.

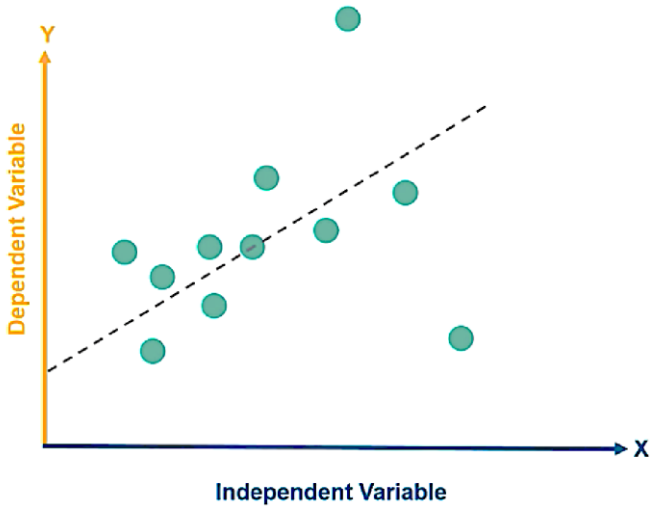


Рисунок 2.1 – Зв’язок залежної змінної від незалежної

Чи є модель «правильною», залежить від того, чи відповідають відносини між даними певним припущенням:

1. Лінійність. За моделлю лінійної регресії передбачення є лінійною комбінацією ознак, що є як її найбільшою силою, так і найбільшим обмеженням.

Лінійність призводить до інтерпретованих моделей. Лінійні ефекти легко кількісно оцінити та описати. Вони є адитивними, тому їх легко розділити. Якщо ви підозрюєте взаємодію змінних або нелінійний зв'язок змінної з цільовим значенням, ви можете додати умови взаємодії або використовувати сплайни регресії.

2. Нормальність. Передбачається, що цільовий результат відповідає нормальному розподілу. Якщо це припущення порушується, розрахункові довірчі інтервали ваг об'єктів є недійсними.

3. Гомоскедастичність. Вважається, що дисперсія похибок є постійною на всьому просторі ознак. Припустимо, ви хочете спрогнозувати вартість будинку з урахуванням житлової площі в квадратних метрах. Ви оцінюєте лінійну модель, яка передбачає, що, незалежно від розміру будинку, помилка навколо передбачення має однакову дисперсію. Це припущення часто порушується в реальності. У прикладі з будинком вірогідно, що дисперсія похибки прогнозованої ціни є вищою для великих будинків, оскільки ціни вищі і є більше можливостей для коливань цін. Припустимо, середня помилка (різниця між прогнозованою та фактичною ціною) у вашій моделі лінійної регресії становить 50 000 євро. Якщо ви припустите гомоскедастичність, ви припустите, що середня похибка 50 000 однакова для будинків, які коштують 1 мільйон, і для будинків, які коштують лише 40 000. Це не обгрунтовано, оскільки це означатиме, що ми можемо очікувати негативних цін на житло.

4. Незалежність. Передбачається, що кожен екземпляр незалежний від будь-якого іншого. Якщо ви виконуєте повторні вимірювання, наприклад, кілька аналізів крові на одного пацієнта, елементи даних не є незалежними. Для залежних даних потрібні спеціальні моделі лінійної регресії, такі як моделі змішаних ефектів або GEE. Якщо ви використовуєте «звичайну» модель лінійної регресії, ви можете зробити неправильні висновки з моделі.

5. Фіксовані змінні. Вхідні змінні вважаються «фіксованими». Фіксований означає, що вони розглядаються як «задані константи», а не як статистичні змінні. Це означає, що вони не мають помилок вимірювань. Це

досить нереалістичне припущення. Однак без цього припущення доведеться підбирати дуже складні моделі вимірювання помилок, які враховують похибку вимірювання ваших вхідних характеристик.

6. Відсутність мультиколінеарності. Сильно корельовані змінні не потрібні, оскільки це заважає оцінці ваг. У ситуації, коли дві ознаки сильно корелюють, стає проблематичною оцінка вагових коефіцієнтів, оскільки ефекти ознак є адитивними, і неможливо визначити, якій із корельованих ознак приписувати ефекти [6].

Однак лінійна регресія часто непридатна до реальних даних через занадто обмежені можливості і обмежену свободи маневру. Її часто використовують тільки в якості базової моделі для оцінки та порівняння з новими підходами в дослідженнях.

2.2.2 Використання методу опорних векторів для задач регресії

Метод опорних векторів є одним із найпопулярніших і широко використовуваних алгоритмів для вирішення проблем класифікації в машинному навчанні. Однак використання SVM в регресії не дуже добре задокументовано. Цей алгоритм визнає наявність нелінійності в даних і забезпечує кваліфіковану модель прогнозування [11].

Метод опорних векторів також може використовуватися як метод регресії, зберігаючи всі основні характеристики, що характеризують алгоритм. Регресія опорного вектора (SVR) використовує ті ж принципи, що й SVM для класифікації, лише з кількома незначними відмінностями. Перш за все, оскільки виходом є дійсне числом, стає дуже важко передбачити наявну інформацію, яка має нескінченні можливості. У разі регресії, межа допуску (епсилон) встановлюється в наближенні до SVM. Але крім цього є ще й більш складна причина - алгоритм складніший, тому його слід враховувати. Однак основна ідея завжди одна: мінімізувати помилку, індивідуалізуючи гіперплощину, яка максимізує запас, пам’ятаючи, що частина помилки допускається [10].

У машинному навчанні машини опорних векторів — це моделі навчання вчителем з відповідними алгоритмами навчання, які аналізують дані, що використовуються для класифікації та регресійного аналізу. У регресії опорного вектора пряма лінія, до якої необхідно прагнути в результаті, називається гіперплощиною.

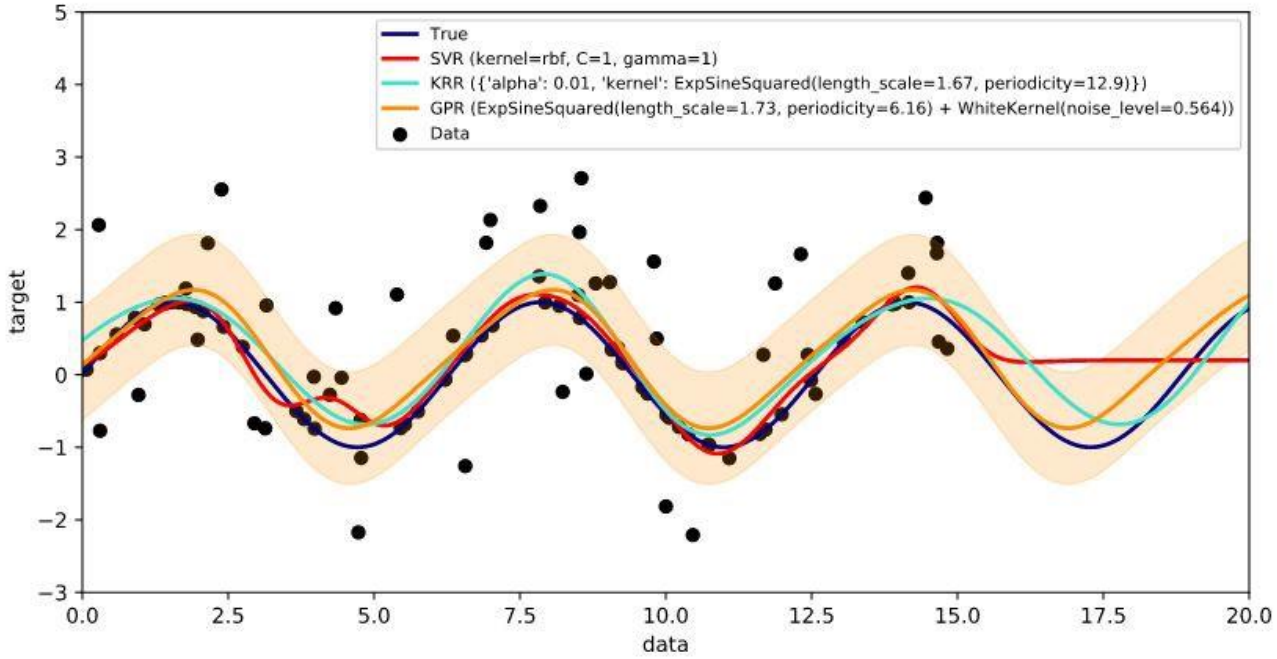


Рисунок 2.2 - Прогнози з різними моделями навчання

Метою алгоритму опорних векторів є знайти гіперплощину в n-вимірному просторі, яка чітко класифікує точки даних. Точки даних по обидва боки від гіперплощини, які є найближчими до гіперплощини, називаються опорними векторами. Вони впливають на положення та орієнтацію гіперплощини і, таким чином, допомагають побудувати SVM.

Існують різні гіперпараметри, які використовуються в регресії опорних векторів.

Гіперплощини — це межі прийняття рішень, які використовуються для прогнозування безперервного результату. Точки даних по обидва боки від гіперплощини, які є найближчими до гіперплощини, називаються опорними

векторами. Вони використовуються для побудови потрібної лінії, яка показує прогнозований результат алгоритму.

Ядро — це набір математичних функцій, який приймає дані як вхідні дані та перетворює їх у необхідну форму. Зазвичай вони використовуються для пошуку гіперплощини у просторі вищих вимірів. Найбільш широко використовувані ядра включають лінійне, нелінійне, поліноміальне, радіально-базисну функцію (RBF) і сигмовидну. За замовчуванням в якості ядра використовується RBF. Кожне з цих ядер використовується залежно від набору даних.

Граничні лінії - це дві лінії, які проведені навколо гіперплощини на відстані ε (епсилон). Він використовується для створення межі між точками даних.

На відміну від інших моделей регресії, які намагаються мінімізувати похибку між реальним і прогнозованим значенням, SVR намагається вмістити найкращу лінію в межах порогового значення. Порогове значення — це відстань між гіперплощиною та граничною лінією [11].

2.2.3 Дерева рішень та алгоритми бустингу

Дерево рішень - це перевернуте дерево, яке приймає рішення на основі умов, наявних в даних [7].

Дерево рішень має гілки, вузли, листя і т. д. Кореневий вузол - це початковий вузол, який представляє всю вибірку або сукупність, і він може бути далі розділений на інші вузли або однорідні безлічі. Вузол рішення складається з двох або більше вузлів, які представляють окремі значення атрибута, що перевіряється.

Листовий / кінцевий вузол не поділяється на інші вузли і являє собою рішення. Гілка або піддерево - це частина всього дерева. Поділ - це процес поділу вузла на дві або більше підсистеми. Протилежність поділу називається скороченням, тобто видаленням підсистемами вузла прийняття рішення. Батьківський вузол - це вузол, який ділиться на підвузли, а підвузол - це дочірній вузол.

Алгоритм дерева рішень використовує екземпляр даних і проходить через все дерево, задаючи правдиві / неправдиві питання. Починаючи з кореневого вузла, ставлять запитання, і для кожної відповіді створюються окремі гілки, і це триває до тих пір, поки не буде досягнутий кінцевий вузол. Рекурсивне розбиття використовується для побудови дерева [8].

Дерева прийняття рішень (decision tree) є універсальними алгоритмами машинного навчання, які можуть займатися завданнями класифікації і регресії, включаючи навіть багатовихідові завдання. Вони являють собою дуже потужні алгоритми, здатні пристосовуватися до складних наборів даних.

Дерева прийняття рішень висувають дуже мало припущень про навчальні дані (на противагу лінійним моделям, які очевидним чином припускають, що дані лінійні, наприклад). Залишена не зв'язана обмеженнями, деревоподібна структура буде адаптувати себе до навчальних даних, дуже близько підганяючи себе до них і, швидше за все, допускаючи перенавчання. Така модель часто називається непараметричної моделлю (nonparametric mode), але не через відсутність будь-яких параметрів (вони є і нерідко в достатку), а з тієї причини, що кількість параметрів перед навчанням не визначене, тому структура моделі вільна тісно прив'язуватися до даних . На противагу цьому параметрична модель (parametric mode, подібна лінійній моделі, має визначену кількість параметрів, так що її ступінь свободи обмежується, скорочуючи ризик перенавчання (але збільшуючи ризик недонавчання) [9].

Дерева рішень із градієнтним бустингом — це техніка машинного навчання, яка використовується для оптимізації прогнозної цінності моделі за допомогою послідовних кроків у процесі навчання. Кожна ітерація дерева рішень включає коригування значень коефіцієнтів, ваг або зміщень, що застосовуються до кожної з вхідних змінних, що використовуються для прогнозування цільового значення, з метою мінімізації функції втрат (міра різниці між прогнозованими і фактичними цільовими значеннями). Градієнт — це покрокове коригування, яке здійснюється на кожному етапі процесу;

підсилення – це метод прискорення підвищення точності прогнозування до достатньо оптимального значення.

Градієнтний бустинг є популярним методом розв’язування задач прогнозування як в області класифікації, так і в області регресії. Цей підхід покращує процес навчання за рахунок спрощення мети та зменшення кількості ітерацій для досягнення достатньо оптимального рішення. Моделі з градієнтним бустингом неодноразово зарекомендували себе на різноманітних змаганнях, які оцінювали як точність, так і ефективність, що робить їх основним компонентом у наборі інструментів науковця з даних [12].

Градієнтний бустинг — це один із варіантів методів ансамблю, коли ви створюєте кілька слабких моделей і об’єднуєте їх, щоб отримати кращу продуктивність в цілому [13].

Градієнтний бустинг включає три елементи: функцію втрат, яку потрібно оптимізувати, слабка модель, додаткова модель для додавання слабких учнів, щоб мінімізувати функцію втрат.

Використовувана функція втрат залежить від типу проблеми, що вирішується. Вона повинна бути диференційованою, але підтримується багато стандартних функцій втрат, і можна визначити власні.

Наприклад, регресія може використовувати квадратну помилку, а класифікація може використовувати логарифмічні втрати.

Перевага градієнтного бустингу полягає в тому, що новий алгоритм не потрібно виводити для кожної функції втрат, яку можна використовувати, натомість це досить загальна структура, щоб можна було використовувати будь-яку диференційовану функцію втрат.

Дерева рішень використовуються як слабкі «учні» у градієнтному бустингу. Зокрема, використовуються дерева регресії, які виводять реальні значення для розщеплень і вихідні дані яких можна додати разом, дозволяючи додавати вихідні дані наступних моделей і «коригувати» залишки в передбаченнях. Зазвичай слабкі моделі обмежуються певними способами, такими як максимальна кількість шарів, вузлів, розщеплень або листових вузлів.

Це робиться для того, щоб «учні» залишалися слабкими, але все ще могли бути сконструйованими.

Дерева додаються по одному, а існуючі дерева в моделі не змінюються. Для мінімізації втрат при додаванні дерев використовується процедура градієнтного спуску.

Традиційно градієнтний спуск використовується для мінімізації набору параметрів, таких як коефіцієнти в рівнянні регресії або ваги в нейронній мережі. Після обчислення помилки або втрат ваги оновлюються, щоб мінімізувати цю помилку.

Замість параметрів ми маємо слабкі підмоделі учнів або, точніше, дерева рішень. Після розрахунку втрат, щоб виконати процедуру градієнтного спуску, ми повинні додати до моделі дерево, яке зменшує втрати (тобто слідувати за градієнтом). Ми робимо це, параметризуючи дерево, потім змінюємо параметри дерева і рухаємося в правильному напрямку (зменшуючи залишкові втрати).

Зазвичай цей підхід називається функціональним градієнтним спуском або градієнтним спуском з функціями. Вихідні дані для нового дерева потім додаються до вихідних даних існуючої послідовності дерев, щоб виправити або покращити кінцевий результат моделі.

Додається фіксована кількість дерев або навчання припиняється, коли втрата досягає прийнятного рівня або набір даних зовнішньої перевірки більше не покращується [14].

Процес бустингу виглядає так: спочатку будується початкова модель з даними. Далі виконується передбачення для всього набору даних і відповідно обчислюється помилка, використовуючи прогнози і фактичні значення. Для наступного створення іншої моделі, більше ваги надається помилковим прогнозам, а нова модель буде намагатися виправити помилки останньої моделі. Наступним кроком є передбачення для всього набору даних за допомогою нової моделі. Далі створюється кілька моделей для кожної моделі, які спрямовані на виправлення помилок, створених попередньою. Таким чином, отримуємо остаточну модель, зваживши середнє значення всіх моделей.

XGBoost — це бібліотека для градієнтного бустингу, яка підтримується для мов програмування Java, Python, Java і C++, R і Julia. Вона також використовує ансамбль слабких дерев рішень.

Це лінійна модель, яка вивчає дерево за допомогою паралельних обчислень. Алгоритм також постачається з функціями для виконання перехресної перевірки та показу важливості функції. XGBoost приймає кілька типів вхідних даних, добре працює з розрідженими вхідними даними для дерева та лінійного підсилювача та підтримує використання налаштованих функцій цілі та оцінки.

LightGBM відрізняється від інших фреймворків градієнтного бустингу, оскільки використовує алгоритм росту дерева по листу. Відомо, що алгоритми росту дерев по листу сходяться швидше, ніж алгоритми росту по глибині. Однак вони більш схильні до перенавчання.

Алгоритм заснований на гістограмі, тому він розміщує безперервні значення в дискретних ящиках. Це призводить до швидшого навчання та ефективного використання пам’яті. Інші примітні особливості цього алгоритму включають в себе вбудовану підтримку категорійних функцій, здатність працювати з великомасштабними даними, обробку відсутніх значень за замовчуванням[15].

2.3 Критерії оцінки якості роботи моделей

Перевірка моделі є важливим кроком у процесі моделювання і допомагає оцінити надійність моделей перед тим, як їх можна буде використовувати для прийняття рішень. Правильний вибір змінних мінімізує помилку невідповідності моделі, тоді як вибір підходящої моделі зменшує помилку оцінки моделі. Моделі мають перевірятися, щоб звести до мінімуму помилку передбачення моделі. Визначення правильної форми моделі для зменшення помилки невідповідності моделі виконується на етапі побудови моделі, тоді як визначення правильного параметра моделі може бути досягнуто під час вибору та перевірки моделі.

Після того, як регресійна модель була побудована, важливо підтвердити відповідність моделі та статистичну значущість оцінених параметрів. Перевірка є важливою частиною побудови моделі, її застосування та рівень впевненості у використанні дуже важливі [27].

Під час пошуку найкращої моделі для прогнозування, неправильний вибір може зруйнувати усю попередню роботу. Тому для оцінки моделі використовуються різноманітні метрики.

R-квадрат (R^2) або коефіцієнт детермінації, скоригований R-квадрат (Adj R^2), середня абсолютна похибка (MAE), середня квадратична похибка (MSE) і середньоквадратична похибка (RMSE) є дуже популярними метриками для регресії.

R-квадрат регресії або коефіцієнт детермінації — це інтуїтивно зрозуміла статистична шкала, яка показує наскільки незалежні змінні пояснюють цільову змінну у моделі регресії. Іншими словами, він представляє силу відповідності, однак нічого не говорить про саму модель, чи гарна вона, чи обрані дані упереджені, чи метод моделювання обрано коректно.

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y}_i)^2}, \tag{2.2}$$

де y_i – значення залежних змінних;

$\widehat{y_l}$ – прогнозовані значення;

$\overline{y_l}$ – середнє значення цільової змінної.

Значення R-квадрату знаходяться в діапазоні від 0 до 1. Статична модель, яка завжди передбачає очікуване значення y, незалежно від вхідних характеристик, буде мати значення R-квадрата, що дорівнює 0, тоді як для ідеальної моделі цей показник дорівнює 1. Значення R-квадрат може бути негативним для моделі, яка «візуально» має дуже поганий ступінь відповідності.

Як правило, R-квадрат є мірою відносної відповідності моделі. Таким чином, високий R-квадрат не означає, що відповідність є хорошою або задовільною, це просто означає, що відхилення фактичних точок від підібраних точок, у середньому, невелике. Однак у цього показника є певні мінуси.

Наприклад, додаючи нові змінні у модель, підвищується можливість перенавчання моделі, коли модель намагається передбачити «шум» у даних. Це зменшує здатність моделі працювати з новими даними, яких вона не бачила. Також R-квадрат, як правило, завжди збільшується при додаванні нових змінних у модель, при цьому не обов'язково покращуючи роботу моделі. Щоб подолати цей недолік, перевагу надають іншій метриці, Adjusted R-квадрат (скоригований коефіцієнт детермінації).

$$R_{adj}{}^2 = 1 - \frac{(1-R^2)\,(n-1)}{n-k-1}, \tag{2.3}$$

де n – кількість спостережень у датасеті;

k – кількість незалежних змінних у датасеті.

Скоригований R-квадрат враховує ступені свободи. Він збільшується, коли додавання додаткових змінних до моделі покращує роботу моделі та зменшується в іншому випадку. Скоригований R-квадрат покращує R-квадрат, даючи уявлення про те, чи залежить значення R-квадрат моделі від того, наскільки добре вона передбачає, чи, скоріше, через її складність [29].

Середня абсолютна похибка (MAE) - це сума всіх величин помилок, поділена на кількість точок, тобто, по суті, середня похибка моделі. Отже, чим нижче середня абсолютна похибка, тим менше помилок у вашій моделі.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y_i}|, \tag{2.4}$$

де n – кількість спостережень у датасеті;

y – цільова змінна;

$\hat{y}$ – прогнозоване значення.

Середня квадратична похибка (MSE) або середньоквадратичне відхилення, — це міра близькості лінії регресії до набору точок. Вона вимірює відстані, які називаються помилками (тобто різниці реальних значень від передбачених) або дисперсією залишків від точок до лінії регресії, і зводить їх у квадрат, щоб видалити будь-які негативні ознаки.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \quad (2.5)$$

де n – кількість спостережень у датасеті;

y – цільова змінна;

$\hat{y}$ – прогнозоване значення.

Для обчислення середньоквадратичної похибки (RMSE) беруть квадратний корінь з середньоквадратичного відхилення і розглядають цю метрику на абсолютну відповідність моделі даним – наскільки близькі точки спостережуваних даних до прогнозованих значень моделі. RMSE відносно легше інтерпретувати, оскільки ця метрика знаходиться в тих же одиницях, що й цільова змінна. Нижчі значення RMSE означають, що лінія регресії близька до точок даних, що вказує на кращу відповідність. RMSE є хорошим показником того, наскільки точно модель прогнозує відповідь, і це найважливіший критерій відповідності, якщо головною метою моделі є передбачення [28].

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \quad (2.6)$$

де n – кількість спостережень у датасеті;

y – цільова змінна;

$\hat{y}$ – прогнозоване значення.

Використання середньої абсолютної похибки має деякі переваги.

По-перше, ми можемо захотіти розглядати невеликі помилки так само, як і великі помилки. Наприклад, припустимо, що ви підбираєте дані, які зазвичай не мають великих помилок, за винятком однієї великої аномальної точки даних. Якщо ви виберете модель лінійної регресії на основі мінімального RMSE, ваша модель може бути перенавченою, оскільки ви намагалися б уловити аномалію.

У такому випадку, враховуючи, що ваші дані загалом однорідні з незначною кількістю чи без видимих великих помилок, вибір регресійної моделі з найнижчим MAE може бути більш доречним [29].

Статистичні методи можуть спрямувати у правильному напрямку, але в кінцевому підсумку потрібно враховувати ще інші міркування. По-перше, можна дослідити, які рішення вже існують, врахувати плюси та помилки при побудові

своєї моделі. Перш ніж розрочати регресійний аналіз, потрібно сформулювати уявлення про важливі змінні, їх взаємозв'язки, величини ефекту.

По-друге, краще починати з простого. Багато досліджень показують, що простіші моделі зазвичай дають більш точні прогнози. Чим складнішою є модель, тим більша ймовірність, що модель просто адаптована до свого набору даних. Отже, при оцінці моделей потрібно розраховувати не лише на статичні методи, адже вони не розуміють основний процес чи предметну область дослідження [30].

2.4 Реалізація математичного апарату прогнозування цін на автотранспортні засоби у комп'ютерних системах

Для аналізу та побудови таких складних моделей, як в цьому дослідженні, є потреба в використанні можливостей комп'ютерних систем і програм, а саме їх обчислювальних потужностей. Для отримання відносно швидкого при обчисленні великого обсягу даних результату, використовується мова програмування Python. В середовищі мови програмування Python реалізовано багато бібліотек та пакетів, що містять математичні та статистичні методи, функції розрахунків похибок, класи для моделювання, методи візуалізації.

Для побудови моделей, аналізу бази даних, візуалізацій та графіків залежностей змінних будуть використовуватися бібліотеки з середовища мови програмування Python.

Першим прикладом такої бібліотеки є Pandas – програмна бібліотека для маніпулювання даними та їхнього аналізу. Pandas в основному використовується для аналізу даних і пов'язаних з ними маніпуляцій з табличними даними в Dataframes. Pandas дозволяє виконувати різноманітні операції маніпулювання даними, а також очищення даних. Можливості цієї бібліотеки будуть використовуватися для більш зручного та ефективного опрацювання даних, наприклад створення об'єктів DataFrame, зберігання таблиць (даних) у файл (.csv), зчитування даних з файлу, а також створення фіктивних змінних.

Для математичних розрахунків використовується бібліотека NumPy. З-за допомогою математичних функцій буде застосована логарифмічна трансформація значень змінних датасету.

Matplotlib – бібліотека, що дозволяє будувати візуалізації та різні типи графіків для первинного аналізу та кращого відображення залежностей. Бібліотека Seaborn надає інтерфейс для побудови статистичних графіків.

Певні перетворення можуть бути потрібні для змінних у наборі даних. Модуль re забезпечує операції зіставлення регулярних виразів. Тобто використовуючи регулярні вирази, можна шукати певні патерни, строки, потрібні частини речень та вилучати їх, формуючи нові атрибути.

Для отримання даних з серверу за запитами, використовується спеціальна HTTP бібліотека requests. Requests — це модуль Python, який можна використовувати для надсилання всіх видів HTTP-запитів. Це проста у використанні бібліотека з великою кількістю функцій, починаючи від передачі параметрів в URL-адресах до надсилання користувацьких заголовків і перевірки SSL.

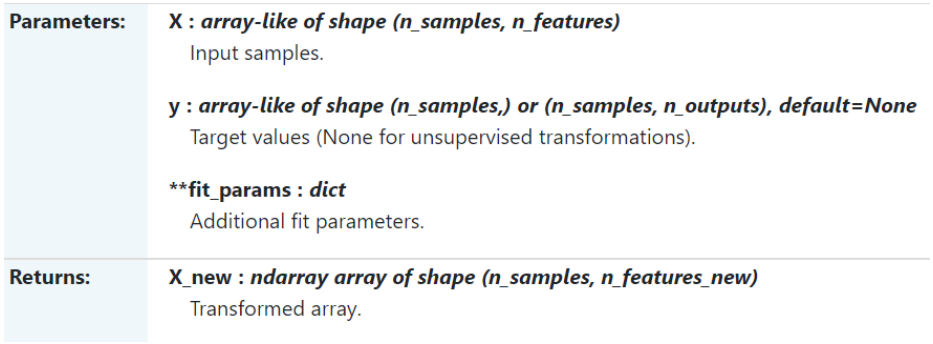
Найпоширеніша та найголовніша бібліотека, що використовується саме для побудови моделей та різних перетворень – Scikit-learn (Sklearn). Це безкоштовна бібліотека машинного навчання для мови програмування Python. Вона містить різні алгоритми класифікації, регресії та кластеризації, включаючи машини опорних векторів, градієнтний бустинг.

Для побудови моделей XGBoost та Light GBM використовуються спеціальні окремі бібліотеки xgboost та lightgbm відповідно.

Розглянемо основні методи та функції, що використовуються для аналізу, підготовки даних та створення моделей з бібліотеки sklearn.

1. Пакет sklearn.preprocessing, що включає методи масштабування, центрування, нормалізації, бінаризації. Пакет sklearn.preprocessing надає кілька загальних допоміжних функцій і класів трансформаторів для зміни необроблених векторів ознак на представлення, яке більше підходить для побудови моделей.

Для перетворення категоріальних змінних на числові (наприклад для проведення кластерного аналізу) використовується клас OrdinalEncoder, а саме метод fit_transform(). На рисунку 2.3 показані можливі параметри, що приймає цей метод та результат.



Parameters:	**X :** ***array-like of shape (n_samples, n_features)*** Input samples. **y :** ***array-like of shape (n_samples,) or (n_samples, n_outputs), default=None*** Target values (None for unsupervised transformations). ****fit_params :** ***dict*** Additional fit parameters.
Returns:	**X_new :** ***ndarray array of shape (n_samples, n_features_new)*** Transformed array.

Рисунок 2.3 – Можливі параметри методу fit_transform класу OrdinalEncoder

Одним з доступних методів нормалізації є клас MinMaxScaler. З-за допомогою його методів можна трансформувати значення атрибутів, масштабуючи кожне значення до заданого діапазону. За замовчуванням діапазон дорівнює [0, 1], що означає, що мінімальне та максимальне значення ознаки/змінної будуть 0 та 1 відповідно.

2. Пакет sklearn.model_selection включає методи оптимізації гіперпараметрів, методи валідації моделей та функції поділу наборів даних.

Метод train_test_split розділяє масиви або матриці на випадкові тренувальні та тестові підмножини. Даний метод використовується після підготовки даних до моделювання і безспосередньо для отримання 2 вибірок для тренування моделі та її валідації. Можливі параметри вказані на рисунку 2.4.

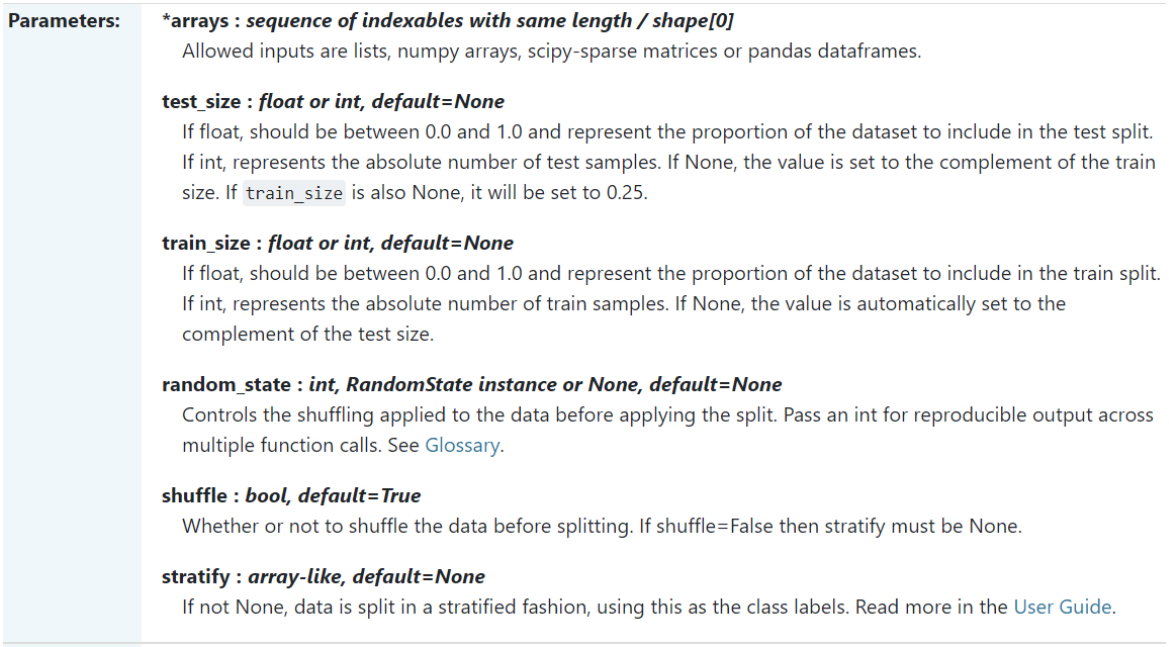


Рисунок 2.4 – Можливі параметри методу train_test_split

Для вирішення проблеми вибору параметрів для тренування модел використовується функція GridSearchCV. Ця функція допомагає перебирати попередньо визначені гіперпараметри та створює модель для кожної комбінації параметрів. Отже, зрештою, ви можете вибрати найкращі параметри з перерахованих гіперпараметрів. При створенні об’єкту GridSearchCV потрібно визначити кілька аргументів:

estimator: навчальна модель (об’єкт моделі з sklearn);

params_grid: список, який містить гіперпараметри, які ви хочете спробувати перебрати;

scoring: метрика оцінки, яку ви хочете використовувати, ви можете просто передати дійсний рядок/об'єкт метрики оцінки;

cv: кількість перехресних перевірок, які будуть проведені для кожного вибраного набору гіперпараметрів;

verbose: ви можете встановити значення 1, щоб отримувати детальний опис, що відбувається, під час підгонки даних до GridSearchCV;

n_jobs: кількість процесів, які ви хочете запускати паралельно для цього завдання, якщо воно -1, воно використовуватиме всі доступні процесори.

3. Пакет sklearn.metrics включає функції оцінки моделей, показники продуктивності та попарні метрики та обчислення відстані. Модуль sklearn.metrics також надає набір простих функцій, які вимірюють похибку передбачення з урахуванням істинних значень та передбачення. Так для розрахунку статистичних показників використовуються функції mean_squared_error, mean_absolute_error.

4. Клас sklearn.linear_model.LinearRegression використовується для моделювання лінійної регресії.

```
class sklearn.linear_model.LinearRegression(*, fit_intercept=True, normalize='deprecated', copy_X=True, n_jobs=None, positive=False) ¶ [source]
```

5. Клас sklearn.svm.SVR використовується для моделювання регресії опорних векторів, маючи параметри:

```
class sklearn.svm.SVR(*, kernel='rbf', degree=3, gamma='scale', coef0=0.0, tol=0.001, C=1.0, epsilon=0.1, shrinking=True, cache_size=200, verbose=False, max_iter=- 1) ¶ [source]
```

C - параметр регуляризації. Сила регуляризації обернено пропорційна C. Повинна бути строго позитивною;

epsilon - параметр контролює ширину ε-нечутливої зони, яка використовується для відповідності навчальним даним. Значення ε може впливати на кількість опорних векторів, що використовуються для побудови функції регресії. Чим більше ε, тим менше опорних векторів обирається.

6. Клас sklearn.ensemble.GradientBoostingRegressor використовується для моделювання градієнтного бустингу для регресії:

```
class sklearn.ensemble.GradientBoostingRegressor(*, loss='squared_error', learning_rate=0.1, n_estimators=100, subsample=1.0, criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_depth=3, min_impurity_decrease=0.0, init=None, random_state=None, max_features=None, alpha=0.9, verbose=0, max_leaf_nodes=None, warm_start=False, validation_fraction=0.1, n_iter_no_change=None, tol=0.0001, ccp_alpha=0.0) [source]
```

n_estimators - це в основному кількість етапів підвищення, які повинні виконуватися моделлю. Інакше кажучи, n_estimators означає кількість дерев у лісі. Більша кількість дерев допомагає краще вивчати дані. З іншого боку, більша кількість дерев може призвести до більшого часу навчання;

max_depth - глибина оцінки дерева рішень;

learning_rate - розмір кроку на кожній ітерації, рухаючись до мінімуму функції втрат;

criterion - використовується для вимірювання якості розбиття для дерева рішень;

loss - вказує функцію втрат, яку потрібно оптимізувати. Існують різні функції втрат, такі як ls, що означає регресію найменших квадратів. Найменше абсолютне відхилення, скорочено як lad, є ще однією функцією втрат. Третя функція втрат Губера — це комбінація регресії за найменшими квадратами та найменшого абсолютного відхилення;

subsample - частка вибірок, що використовуються для підбору окремих «учнів». Якщо subsample менша за 1, це призводить до зменшення дисперсії та збільшення упередженості.

Бібліотека xgboost, а саме XGBoostRegressor використовується для побудови моделі XGBoost, маючи такі параметри:

— booster – тип моделі для запуску після кожної ітерації. Може приймати значення gbtree чи gblinear;

— silent – визначає чи виводити повідомлення;

— nthread – кількість ядер в системі, що будуть використовуватися;

— eta – аналогічно learning_rate;

— min_child_weight – визначає мінімальну суму ваг усіх спостережень, використовується для контролю перенавчання;

— max_depth – максимальна глибина дерева, використовується для контролю перенавчання;

— gamma – мінімальне скорочення втрат, необхідне для розбиття вузла дерева;

— subsample - частка спостережень, що є випадковими вибірками для кожного дерева.

РОЗДІЛ 3 ПОБУДОВА МОДЕЛЕЙ ПРОГНОЗУВАННЯ ЦІН НА АВТОТРАНСПОРТНІ ЗАСОБИ ТА ЇХ ВАЛІДАЦІЯ

3.1 Формування набору даних та опис вхідних даних

Для подальшого аналізу та побудови моделей на базі математичного апарату, потрібно використати набір даних, що містить інформацію про вживані автомобілі. Набір даних має складатися з екземплярів машин: їх характеристик та цін, за які вони продаються.

Було проаналізовано вже існуючі набори даних на різних ресурсах (наприклад, Kaggle), що збиралися з різних веб-сайтів чи додатків. На жаль, більшість з них містили застарілу інформацію (збір був проведений більше 3 років тому), а також неактуальну інформацію для нашого регіону (Східна Європа – Україна). Тож було вирішено провести власний збір актуальної інформації з українського ринку роздрібної торгівлі автотранспортними засобами, а саме з сайту AUTO.RIA.

AUTO.RIA – це сайт (додаток), що слугує майданчиком для розміщення оголошень від приватних осіб і компаній про купівлю-продаж нових автомобілів, та автомобілів із пробігом. Даний ресурс надає додаткові можливості для розробників – пошук авто, пошук авто за фільтром, отримання списку замовлень, зміна деталей замовлення. Компанія створила певні API (інтерфейс програмування застосунків), що дозволяють створювати свої додатки, використовуючи можливості RIA.com (AUTO.RIA, DOM.RIA). Для роботи з даними потрібно зареєструватися на порталі та отримати персоналізований ключ. Для розробників великих спеціалізованих додатків є певні спрощення у роботі.

Для збору даних з веб-ресурсів також використовується веб-скрапінг. Проте в даному випадку було вирішено використовувати можливості AUTO.RIA, адже це значно спростить та пришвидшить збір потрібних даних.

Першим кроком була фільтрація усіх оголошень на сайті за такими параметрами: категорія «легкові», ціна у доларах, з розмитненням, не

конфісковані, без кредиту, не в аварійному стані (одразу після ДТП). Код, що використовувався для отримання ідентифікаційних номерів оголошень, наведено в додатку А.

Другим кроком по збору даних було виконання запитів за вже зібраними номерами оголошень, що дало можливість отримати усю інформацію з оголошень, а саме: марка та модель авто, пробіг, тип двигуна, тип кузову, тип приводу, коробку передач, колір та ціну. З отриманих даних було створено набір даних, що буде використовуватися в подальшому аналізі та побудові моделей. Код, що використовувався для збору даних з оголошень наведено також у додатку А.

Всього на момент збору даних на сайті AUTO.RIA, використовуючи визначені фільтри, було розміщено 114473 оголошень. З-за допомогою функцій RIA.com Developers було зібрано дані про 97043 оголошень.

Отриманий набір даних можна назвати значним, що може призводити до збільшення часу обробки та побудови моделей, проте має також дати більш точні результати та більше даних на тренувальний, тестовий чи валідаційний сети.

Цей набір містить 16 значень для кожного запису. Тлумачення кожного з показників наведено у Додатку Б.

3.2 Аналіз та підготовка даних для побудови моделей та прогнозування

Наступним етапом перетворення необроблених даних з сайту AUTO.RIA на набір даних, що готовий до подальшої побудови моделей на базі математичного апарату, є підготовка даних: вибір потрібних атрибутів, які можуть мати значення для моделі, очистити дані від неадекватних та пропущених значень, зробити нормалізацію та поділити набір даних на тренувальний і тестовий набори.

Спочатку завантажимо потрібний нам датасет, усі дані з сайту AUTO.RIA.

```
cars = pd.read_csv('raw_cars.csv')
```

На рисунку 3.1 показана частина зібраного датасету.

	auto_id	manufacturer_name	manufacturer_name_eng	manufacturer_name_id	model_name	model_name_id	gear_box	color	odometer_value	year	fuel	fuel_name	body_type	state	drivetrain	price_usd
0	32433984	Audi	audi	6	a4	47	2	gray	270	2008	1	Бензин, 2 л.	3	True	1	7899
1	32451567	Volkswagen	volkswagen	84	passat-alltrack	2805	2	black	262	2017	2	Дизель, 2 л.	2	False	1	19800
2	32380070	Audi	audi	6	a4	47	2	black	26	2018	1	Бензин, 2 л.	3	True	2	21000
3	32170213	Audi	audi	6	a4	47	2	gray	59	2017	1	Бензин, 2 л.	3	True	4	19700
4	32458292	Volkswagen	volkswagen	84	passat-b7	38063	2	other	86	2013	1	Бензин, 1.8 л.	3	True	2	11200

Рисунок 3.1 – Початковий датасет

У цьому наборі наявні змінні таких типів:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 97043 entries, 0 to 97042
Data columns (total 16 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   auto_id               97043 non-null  int64
 1   manufacturer_name     97033 non-null  object
 2   manufacturer_name_eng 97033 non-null  object
 3   manufacturer_name_id  97043 non-null  int64
 4   model_name            97004 non-null  object
 5   model_name_id         97043 non-null  int64
 6   gear_box              97043 non-null  int64
 7   color                 97043 non-null  object
 8   odometer_value        97043 non-null  int64
 9   year                  97043 non-null  int64
 10  fuel                  97043 non-null  int64
 11  fuel_name             97043 non-null  object
 12  body_type             97043 non-null  int64
 13  state                 97043 non-null  bool
 14  drivetrain            97043 non-null  int64
 15  price_usd             97043 non-null  int64
dtypes: bool(1), int64(10), object(5)
memory usage: 11.2+ MB
```

Рисунок 3.2 – Вивід усіх існуючих показників у датасеті

Перевіримо, чи є у наборі пропущені значення та при незначній їх кількості проведемо очищення (рис. 3.3).

```
cars = cars.dropna()
print(cars.isnull().sum())
```

```
auto_id                 0
manufacturer_name      10
manufacturer_name_eng  10
manufacturer_name_id    0
model_name             39
model_name_id           0
gear_box                0
color                   0
odometer_value          0
year                    0
fuel                    0
fuel_name               0
body_type               0
state                   0
drivetrain              0
price_usd               0
dtype: int64
```

```
auto_id                 0
manufacturer_name       0
manufacturer_name_eng   0
manufacturer_name_id    0
model_name              0
model_name_id           0
gear_box                0
color                   0
odometer_value          0
year                    0
fuel                    0
fuel_name               0
body_type               0
state                   0
drivetrain              0
price_usd               0
dtype: int64
```

Рисунок 3.3 – Очищення пропущених значень у датасеті

Для первинного поверхневого аналізу та описової статистики набору даних було створено декілька візуалізацій та графіків залежності змінних. Також є інформативним порівняння цих графіків до та після проведення операцій нормалізації, видалення викидів чи неадекватних значень, тощо.

	auto_id	manufacturer_name_id	model_name_id	gear_box	odometer_value	year	fuel	body_type	drivetrain	price_usd
count	9.700300e+04	97003.000000	97003.000000	97003.000000	97003.000000	97003.000000	97003.000000	97003.000000	97003.000000	97003.000000
mean	3.142680e+07	75.376576	13083.410977	1.830077	191.099791	2009.184644	2.028236	16.041380	2.064266	11741.211488
std	3.368696e+06	635.478121	18387.544257	1.193364	100.518688	6.766688	1.317496	55.642933	0.822819	12852.027672
min	1.788929e+06	2.000000	9.000000	1.000000	0.000000	1937.000000	0.000000	1.000000	1.000000	0.000000
25%	3.178217e+07	24.000000	649.000000	1.000000	123.000000	2006.000000	1.000000	3.000000	2.000000	5100.000000
50%	3.228397e+07	55.000000	2034.000000	2.000000	189.000000	2010.000000	2.000000	3.000000	2.000000	8400.000000
75%	3.239371e+07	79.000000	31567.000000	2.000000	246.000000	2014.000000	2.000000	5.000000	2.000000	14000.000000
max	3.246875e+07	55173.000000	63436.000000	6.000000	1000.000000	2022.000000	9.000000	315.000000	4.000000	499999.000000

Рисунок 3.4 – Описова статистика набору даних

Найстаріша машина у наборі була зроблена 1937 року, коли тільки 25% машин було зроблено до 2006 року. Також можна побачити, що набір містить записи, для яких цільова змінна (ціна) дорівнює 0. Також 75% усіх автомобілей коштують менше 14000, коли максимальне значення ціни дорівнює 499999. Цільова змінна потребує нормалізації та фільтрації.

Для графічної оцінки побудуємо графік залежності ціни автомобіля від пробігу (рис. 3.5).

```
cars.plot(kind='scatter', x='odometer_value', y='price_usd', figsize=(6,6))
```

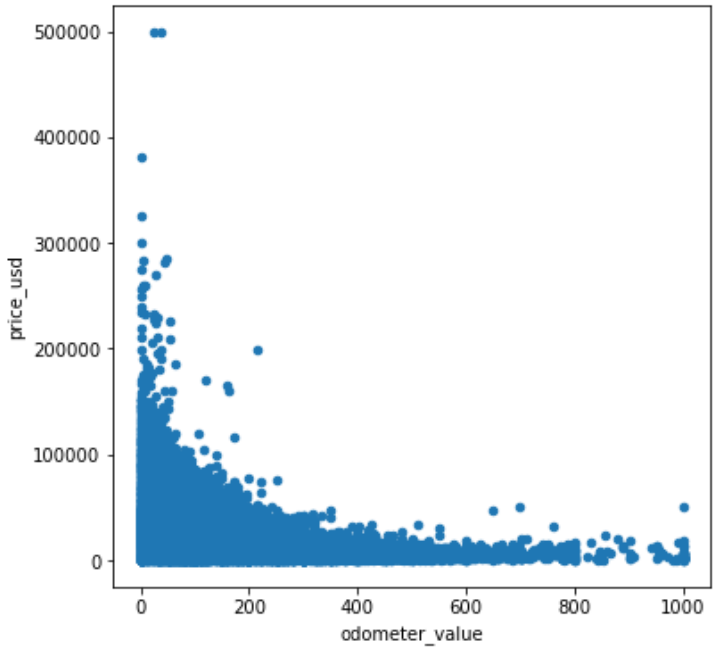


Рисунок 3.5 – Графік залежності ціни від пробігу

З графіку можна побачити, що змінна ціни має зворотній зв’язок зі значенням пробігу, що логічно. Також можна помітити, що наявні декілька викидів, які мають дуже високу ціну. Їх занадто мало, щоб можна було зробити коректне передбачення, але вони будуть впливати на коефіцієнти.

```
cars.plot(kind='scatter', x='year', y='price_usd', figsize=(6,6))
```

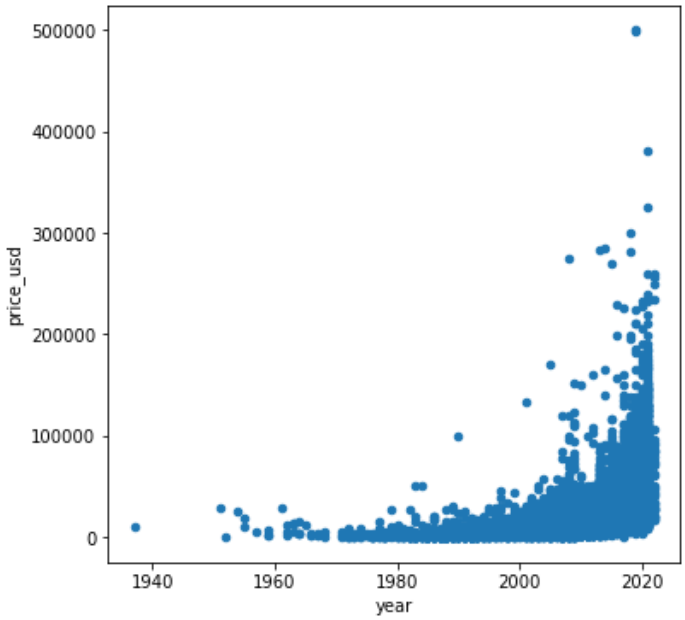


Рисунок 3.6 – Графік залежності ціни від року виробництва

З графіку можна побачити, що змінна ціни має прямий зв’язок зі значенням року виробництва – чим більший рік (менший вік відповідно) тиц ціна більша.

Для аналізу цільової змінної «Ціна» побудуємо гістограму розподілу (рис.3.7).

```
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.title('Car Price Distribution Plot')
sns.distplot(cars.price_usd)
plt.show()
```

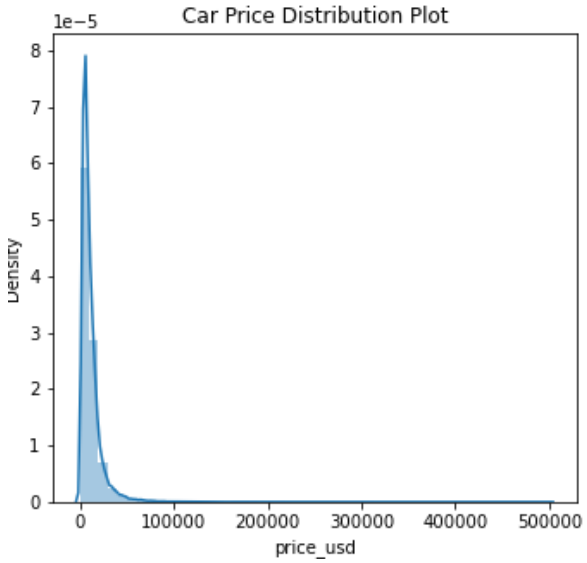


Рисунок 3.7 – Графік розподілу вартості автомобілів

Графік виглядає дуже зміщеним вправо, що означає, що абсолютна більшість цін в наборі даних значно нижча за максимальне значення. Точки даних сильно відрізняються від середнього значення, що вказує на великий розкид цін на автомобілі (75% цін нижче 14 000, тоді як решта 25% перебувають між 14 000 і 499 999).

Після початкового поверхневого аналізу потрібно зробити деякі зміни у самій структурі набору даних, видалити не важливі атрибути, а також провести його нормалізацію.

Спочатку видалимо неадекватні чи нерелевантні значення. Атрибут року виробництва є важливим, проте немає сенсу розглядати занадто старі машини, тому було вирішено видалити ті значення, для яких рік виробництва раніше ніж

2000 рік. Також створимо нову змінну «вік» (age) для більш зручного аналізу та моделювання.

```
cars = cars.loc[cars['year'] >= 2000]
```

```
cars['age'] = 2022 - cars['year']
```

Для нормалізації цільової змінної «Ціна» було видалено нульові значення та занадто великі. Також було проведена нормалізація з-за допомогою логарифмічної трансформації.

```
cars = cars.loc[(cars['price_usd'] > 300) & (cars['price_usd'] < 100000)]
```

При розвідці можливих значень деяких атрибутів було отримано інформацію, що атрибут fuel (тип палива) може містити деякі дуже рідкісні значення, наприклад як «газ пропан-бутан», тобто записів для побудови моделі дуже мало, тому було вирішено видалити записи з таким значенням. Також це стосується атрибуту body_type (тип кузову), що містить чи рідкісні значення чи взагалі помилкові (наприклад, був запис з типом кузову «Кран»).

```
cars = cars.loc[(cars['fuel'] != 0) & (cars['fuel'] != 7) & (cars['fuel'] != 8) & (cars['fuel'] != 9)]
```

```
cars.loc[(cars['body_type'] == 1), 'body_type'] = 28

cars = cars.loc[(cars['body_type'] != 87) & (cars['body_type'] != 95) & (cars['body_type'] != 36)]
```

Наступним кроком в процессі формалізації набору даних була трансформація чи створення нових атрибутів.

Так, наприклад, була проведена зміна значень назв виробника автомобілів (manufacturer_name), адже унікальних значень цього атрибуту дуже багато (рис.3.8).

```
{'manufacturer_name': 121,
 'manufacturer_name_eng': 121,
 'model_name': 1670,
 'color': 12,
 'fuel_name': 385}
```

Рисунок 3.8 – Кількість унікальних значень у колонках

Було проведено видалення усіх назв виробника автомобілів, що зустрічаються у датасеті менше 40 разів, адже це є недостатньою кількістю рядків для побудови моделі (рис.3.9).

```
manufacturer_count = (cars['manufacturer_name_eng'].value_counts() >= 40).to_dict()
cars['manufacturer_name_eng'] = cars['manufacturer_name_eng'].apply(
    lambda x: x if manufacturer_count[f"{x}"] else 'other'
)
```

```
{'manufacturer_name': 121,
 'manufacturer_name_eng': 54,
 'model_name': 1540,
 'color': 12,
 'fuel_name': 385}
```

Рисунок 3.9 – Кількість унікальних значень у колонках після фільтрації

Кількість різних назв виробників скоротилося з 121 до 54.

На жаль, дані, що поверталися з AUTO.RIA не мали інформації щодо літражу двигуна. Точніше, це значення було частиною іншої змінної – загальної інформації про двигун. Тому було створено функцію, що збирала потрібні нам значення і створювала нову змінну – літраж двигуна (engine_capacity).

```
def getNumber(string):
    return re.findall(r"[-+]?(?:\d*\.\d+|\d+)", string)

cars['engine_capacity'] = cars['fuel_name'].apply(
    lambda x: getNumber(x)[0] if len(getNumber(x)) == 1 else 0)
```

Для деяких записів інформації про літраж не було і атрибут engine_capacity для них дорівнював 0. Це не є прийнятним для побудови моделі, тому було вирішено заповнити ці нульові значення середніми за типом палива та типом кузову.

```
temp = cars.loc[cars['engine_capacity'] != 0]
mean_capacity = temp.groupby(['fuel', 'body_type'])['engine_capacity'].mean()

def f(x):
    if x['engine_capacity'] == 0 and x['fuel'] != 'electro':
        return mean_capacity.get(x['fuel']).get(x['body_type'])
    else: return x['engine_capacity']

cars['engine_capacity'] = cars.apply(f, axis=1)
```

Дані, що отримуються з AUTO.RIA не завжди приймають значення формат якого підходить для аналізу – атрибути коробки передач, типу палива, типу кузову та типу приводу. З сайту вони надходять закодовані, тобто приймають числове значення. Сервіс надає також розшифровку для цих значень, тому потрібно зробити дешифрування значень цих атрибутів.

```
gear_box_dict = {
    1: "manual",
    2: "automatic",
    3: "typtronik",
    4: "robotic",
    5: "variable",
    6: "other"
}

cars['gear_box'] = cars['gear_box'].apply(lambda x: gear_box_dict[x])
```

```
fuel_dict = {
    1: "petrol",
    2: "diesel",
    3: "gas",
    4: "gas/petrol",
    5: "hybrid",
    6: "electro",
}

cars['fuel'] = cars['fuel'].apply(lambda x: fuel_dict[x])
```

```
drivetrain_dict = {
    1: "all",
    2: "front",
    3: "back",
    4: "other"
}

cars['drivetrain'] = cars['drivetrain'].apply(lambda x: drivetrain_dict[x])
```

```
body_type_dict = {
    2: "station wagon",
    3: "sedan",
    4: "hatchback",
    5: "SUV/crossover",
    6: "coupe",
    7: "cabriolet",
    8: "minivan",
    9: "pickup",
    28: "other",
    252: "limousine",
    254: "passenger van",
    307: "liftback",
    315: "roadster"
}

cars['body_type'] = cars['body_type'].apply(lambda x: body_type_dict[x])
```

На рисунку 3. 10 зображено набір даних після усіх проведених перетворень.

	manufacturer	gear_box	odometer_value	fuel	body_type	state	drivetrain	price_usd	engine_capacity	age
0	audi	automatic	270	petrol	sedan	1	all	7899	2.0	14
1	volkswagen	automatic	262	diesel	station wagon	0	all	19800	2.0	5
2	audi	automatic	26	petrol	sedan	1	front	21000	2.0	4
3	audi	automatic	59	petrol	sedan	1	other	19700	2.0	5
4	volkswagen	automatic	86	petrol	sedan	1	front	11200	1.8	9
5	volkswagen	automatic	97	petrol	sedan	0	front	9999	1.8	11

Рисунок 3.10 – Набір даних після обробки

Для наглядності на рисунках 3.11, 3.12 та 3.13 зображено нові графіки залежності значень ціни від пробігу, ціни від року виробництва та графік розподілу ціни.

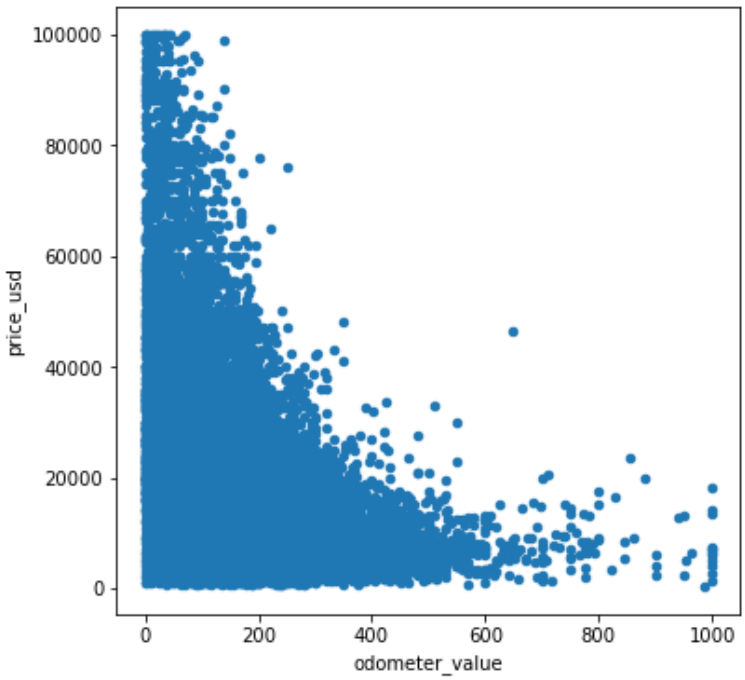


Рисунок 3.11 – Графік залежності ціни від пробігу після перетворень

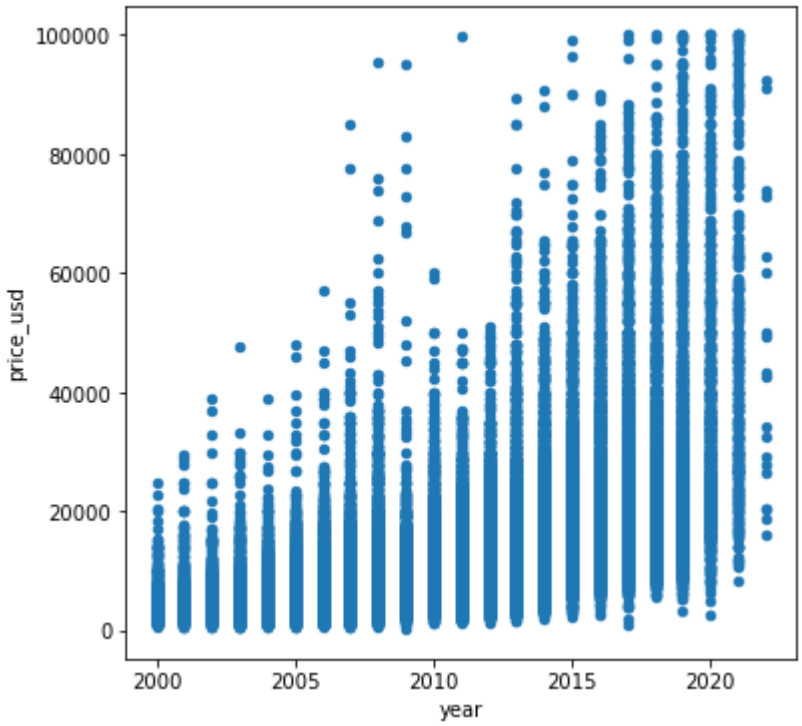


Рисунок 3.12 – Графік залежності ціни від року виробництва після перетворень

Рисунок 3.13 – Графік розподілу ціни після перетворень

З даних графіків можна побачити, що внаслідок перетворень та нормалізації набір даних приблизився до нормально розподілених, нормалізованих значень. Наступними кроками у цій роботі має бути побудова

матриці колінеарності, знаходження важливих атрибутів та саме побудова моделей, створення системи.

На рисунку 3.14 можна побачити графік, що показує середні значення для кожно виробника. За цим графіком можна виділити 3-4 класи машин, проте серед машин, що мають середнє значення менше за 10000 це є складним. Тому було прийняте рішення не замінювати назву виробника на клас машини.

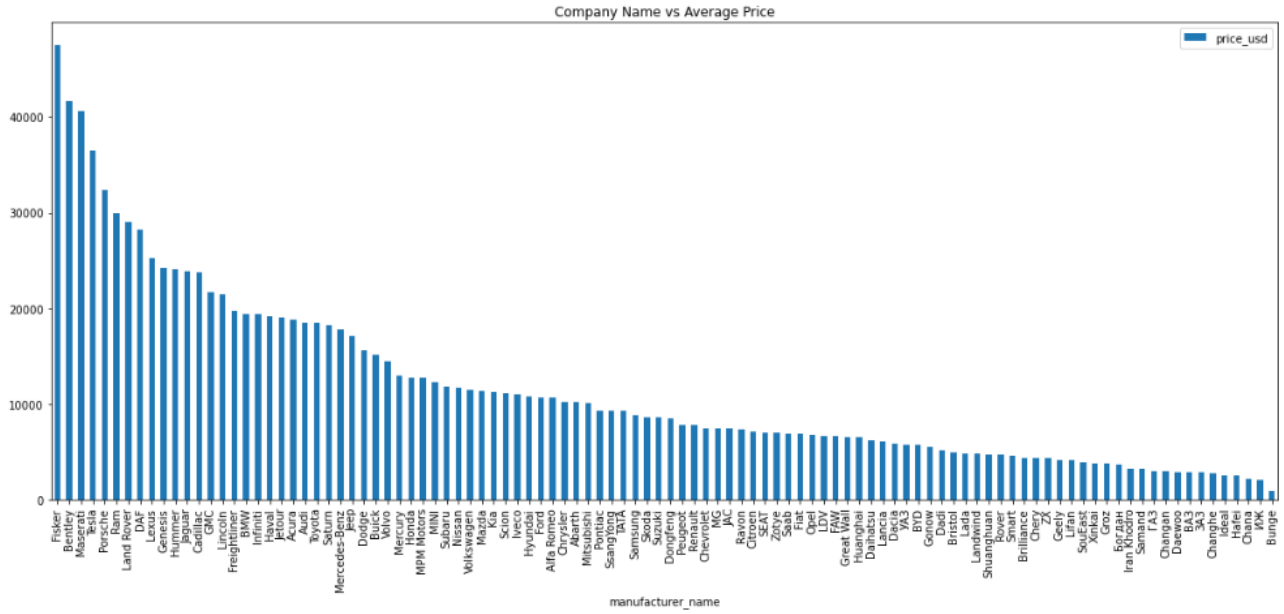


Рисунок 3.14 – Середнє значення ціни для кожного виробника

Перед проведенням кореляційного аналізу було видалено допоміжні показники, наприклад ідентифікатор машини, ідентифікатор виробника, рік виробництва.

Для подальшої оцінки впливу певних показників на ціноутворення, проведемо кореляційний аналіз: перекодуємо категоріальні показники у кількісні та створимо матрицю кореляції (рис.3.15):

```
from sklearn.preprocessing import OrdinalEncoder

corr_cars = cars.copy()
ord_enc = OrdinalEncoder()
corr_cars["manufacturer"] = ord_enc.fit_transform(corr_cars[["manufacturer"]])
corr_cars["model"] = ord_enc.fit_transform(corr_cars[["model"]])
corr_cars["gear_box"] = ord_enc.fit_transform(corr_cars[["gear_box"]])
corr_cars["color"] = ord_enc.fit_transform(corr_cars[["color"]])
corr_cars["fuel"] = ord_enc.fit_transform(corr_cars[["fuel"]])
corr_cars["body_type"] = ord_enc.fit_transform(corr_cars[["body_type"]])
corr_cars["drivetrain"] = ord_enc.fit_transform(corr_cars[["drivetrain"]])
```

```
plt.figure(figsize = (30, 25))
sns.set(font_scale=1.5)
nn = sns.heatmap(corr_cars.corr(), annot = True, cmap="YlGnBu")
plt.show()
```

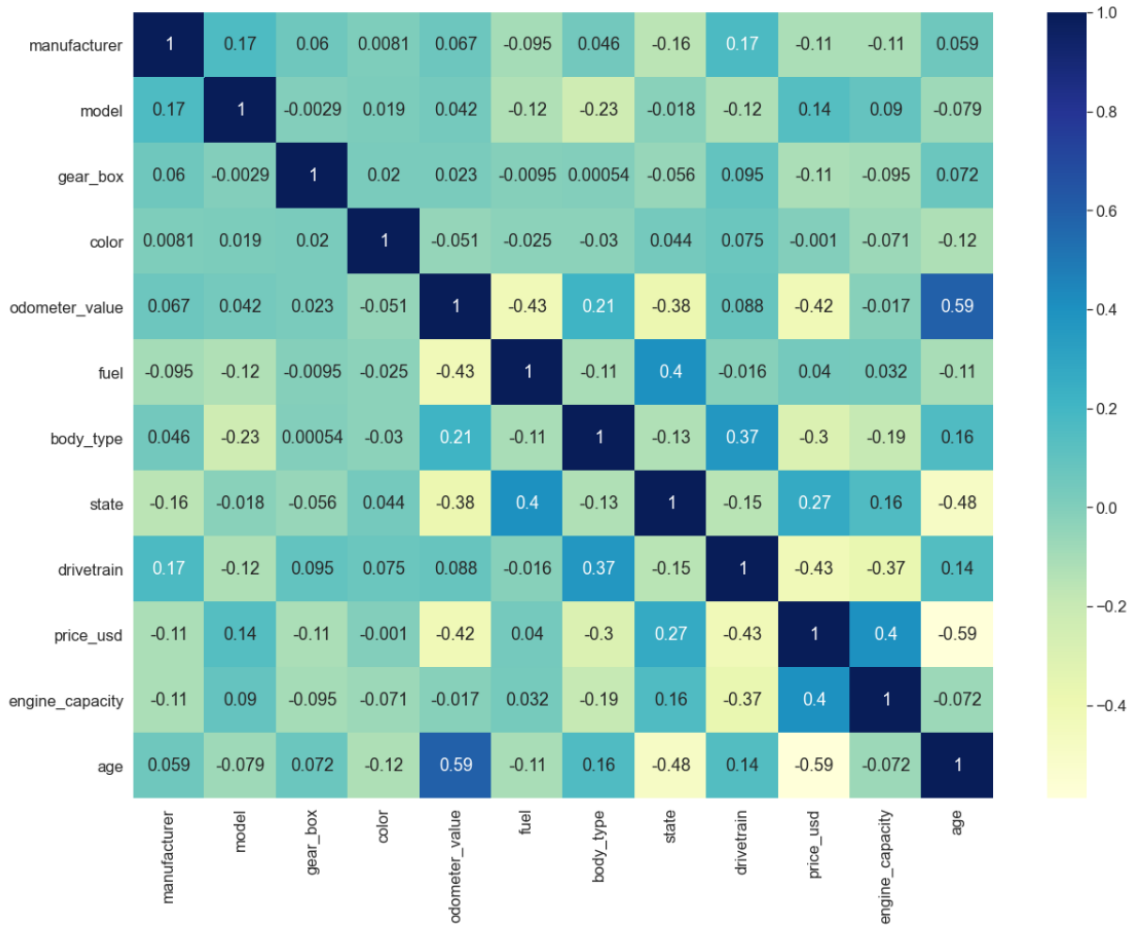


Рисунок 3.15 – Матриця кореляції

Використовуючи отриману матрицю колеряції, можна побачити, що вік має найбільшу кореляцію з ціною автомобіля. Також ціна корелює зі значенням пробігу, об’ємом двигуна та типом приводу. Проте коефіцієнти кореляції між кольором кузову та моделлю відносно малі. Також вік автомобіля та пробіг корелюють між собою, що очевидно, адже пробіг з часом постійно збільшується. Стан автомобіля (чи був в ДТП) корелює з віком, що логічно - чим більше часу автомобіль їздить по дорогах, тим більше в нього було можливості потрапити у ДТП.

На цьому етап очищення і перетворення даних завершено, наступний крок це підготовка набору даних до побудови моделей, а саме маштабування числових змінних та перетворення категоріальних змінних у кількісні.

Для цього створюємо функцію, що вводить фіктивні (dummies) змінні для категоріальних:

```
def set_dummies_vb(x, df):
    temp = pd.get_dummies(df[[x]])
    df = pd.concat([df,temp], axis=1)
    df.drop([x], axis=1, inplace=True)
    return df

cars = set_dummies_vb('manufacturer', cars)
cars = set_dummies_vb('gear_box', cars)
cars = set_dummies_vb('fuel', cars)
cars = set_dummies_vb('body_type', cars)
cars = set_dummies_vb('drivetrain', cars)
```

Тепер настала черга нормалізації даних, з-за допомогою MinMaxScaler з бібліотеки sklearn:

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
num_vars= ['odometer_value', 'engine_capacity', 'age']
cars[num_vars] = scaler.fit_transform(cars[num_vars])
```

Отже, атрибути та цільова змінна були підготовлені для подальшого прогнозування, дані були нормалізовані, позбавлені від викидів та нульових значень.

3.3 Навчання та тестування моделей прогнозування цін на автотранспортні засоби

Після підготовчого етапу розіб’ємо дані у тренувальний та тестувальний набори:

```
from sklearn.model_selection import train_test_split

X = cars.drop(columns = ['price_usd']).copy()
y = cars['price_usd']

x_train, x_test, y_train, y_test = train_test_split(X,y, train_size=0.7, random_state=42)
```

Для вирішення завдання регресії будуть використані лінійна регресія, метод опорних векторів, алгоритми градієнтного бустингу (xgboost, lightGBM). Для побудови моделей будуть використані можливості модуля scikit learn, xgboost та lightgbm. Для оцінки отриманих моделей, порівняння та вибору найкращої буде використовуватися значення середньоквадратичної похибки та коефіцієнт лінійної детермінації (R^2).

Регресія опорних векторів та алгоритми градієнтного бустингу приймають багато різних параметрів, що значно впливають на результативність моделей. Тому дуже важливо знайти найбільш підходящі параметри. Для цього використовується функція GridSearchCV, що перебирає попередньо визначені гіперпараметри та навчає модель постійно з різними параметрами. Отже, зрештою, обираються найкращі параметри з перерахованих гіперпараметрів.

Було створено об’єкт з різними гіперпараметрами для усіх моделей і використано функцію GridSearchCV. Кінцеві найкращі параметри для кожної з моделей показані на рисунку 3.16.

```
algos = {
    'LinearRegression' : {
        'model': LinearRegression(),
        'params': {
            'normalize': [True, False]
        }
    },
    'LGBMRegressor': {
        'model' : LGBMRegressor(),
        'params': {
            'boosting_type': ['gbdt', 'dart', 'goss'],
            'n_estimators': [i for i in range(10, 100, 20)],
            'num_leaves': [7, 14, 21, 28, 31, 50],
            'learning_rate': [0.1, 0.03, 0.003],
            'max_depth': [-1, 3, 5, 8, None],
        }
    },
    'XGBRegressor': {
        'model' : XGBRegressor(),
        'params': {
            'max_depth': [3, 5, 8, None],
            'n_estimators': [i for i in range(10, 100, 20)],
            'learning_rate': [0.1, 0.2, 0.03, 0.003],
        }
    },
    'GradientBoostingRegressor': {
        'model' : GradientBoostingRegressor(),
        'params': {
            'learning_rate': [0.01,0.03,0.003],
            'subsample'    : [0.9, 0.5, 0.2, 0.1],
            'n_estimators' : [100,500,1000],
            'max_depth'    : [4,6,8,10]
        }
    },
    'SVR': {
        'model' : SVR(),
        'params': {
            'C':[1.5, 10],
            'gamma': [1e-7, 1e-4],
            'epsilon':[0.1,0.2,0.5,0.3]
        }
    }
}
```

```
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import GridSearchCV

cv = ShuffleSplit(n_splits=5, test_size=0.2, random_state=0)
for algo_name, config in algos.items():
    gs = GridSearchCV(config['model'], config['params'], cv=cv, scoring='r2', return_train_score=False, verbose=2)
    gs.fit(x_train,y_train)
    best_params.append({
        'model': algo_name,
        'best_params': gs.best_params_,
        'best_estimator': gs.best_estimator_
    })
```

	model	best_params	best_estimator
0	LinearRegression	{'normalize': True}	LinearRegression(normalize=True)
1	LGBMRegressor	{'boosting_type': 'gbdt', 'learning_rate': 0.1...	LGBMRegressor(n_estimators=90, num_leaves=50)
2	XGBRegressor	{'learning_rate': 0.2, 'max_depth': 8, 'n_esti...	XGBRegressor(base_score=0.5, booster='gbtree',...
3	GradientBoostingRegressor	{'learning_rate': 0.03, 'max_depth': 8, 'n_est...	([DecisionTreeRegressor(criterion='friedman_ms...
4	SVR	{'C': 10, 'epsilon': 0.1, 'gamma': 0.0001}	SVR(C=10, gamma=0.0001)

Рисунок 3.16 – Найкращі гіперпараметри для кожної моделі

Отримавши найкращі параметри можна починати навчати моделі.

Першим алгоритмом, що буде використовуватися для навчання моделі є лінійна регресія:

```
lm = LinearRegression(normalize=True)
lm.fit(x_train, y_train)

y_pred_lr = lm.predict(x_test)

score_lr = lm.score(x_test, y_test)
mse_lr = mean_squared_error(y_test, y_pred_lr)
print('R-squared for Linear Regression: ', score_lr)
print('Mean squared error for Linear Regression: ', mse_lr)
```

```
R-squared for Linear Regression:  0.8674659623599584
Mean squared error for Linear Regression:  0.06301708323564058
```

Значення R-квадрат для лінійної регресіїї складає 86,7%, а MSE похибка приблизно 0.063.

Наступним алгоритмом для навчання є регресія опорних векторів:

```
svr = SVR()
svr.fit(x_train, y_train)

y_pred_svr = svr.predict(x_test)

score_svr = svr.score(x_test, y_test)
mse_svr = mean_squared_error(y_test, y_pred_svr)
print('R-squared for SVR: ', score_svr)
print('Mean squared error for SVR: ', mse_svr)
```

```
R-squared for SVR:  0.911374407152431
Mean squared error for SVR:  0.04213956249074474
```

Значення R-квадрат для регресії опорних векторів складає 91,1%, а MSE похибка приблизно 0.042.

Продовжимо дослідження побудовою моделей градієнтного бустингу. Спробуємо побудувати модель звичайного градієнтного бустингу:

```
gb = GradientBoostingRegressor(learning_rate=0.03, max_depth=8, n_estimators=1000,
                               subsample=0.5)
gb.fit(x_train, y_train)

y_pred_gb = gb.predict(x_test)

score_gb = gb.score(x_test, y_test)
mse_gb = mean_squared_error(y_test, y_pred_gb)
print('R-squared for Gradient Boosting: ', score_gb)
print('Mean squared error for Gradient Boosting: ', mse_gb)
```

```
R-squared for Gradient Boosting:  0.9262494826209536
Mean squared error for Gradient Boosting:  0.03506678416430282
```

Значення R-квадрат для моделі градієнтного бустингу складає 92,6%, а MSE похибка приблизно 0.035.

Для побудови наступної моделі Extreme Gradient Boosting (xgboost) використаємо бібліотеку xgboost та натренуємо модель:

```
xgb = XGBRegressor(learning_rate=0.2, max_depth=8, n_estimators=90)
xgb.fit(x_train, y_train)

y_pred_xgb = xgb.predict(x_test)

score_xgb = xgb.score(x_test, y_test)
mse_xgb = mean_squared_error(y_test, y_pred_xgb)
print('R-squared for XGBoost: ', score_xgb)
print('Mean squared error for XGBoost: ', mse_xgb)
```

```
R-squared for XGBoost:  0.920046491779564
Mean squared error for XGBoost:  0.038016172843031715
```

Значення R-квадрат для моделі XGBoost складає 92%, а MSE похибка приблизно 0.038.

Залишається остання модель – Light GBM з бібліотеки lightgbm:

```
lgbmr = LGBMRegressor(n_estimators=90, num_leaves=50)
lgbmr.fit(x_train, y_train)

y_pred_lgbmr = lgbmr.predict(x_test)

score_lgbmr = lgbmr.score(x_test, y_test)
mse_lgbmr = mean_squared_error(y_test, y_pred_lgbmr)
print('R-squared for LGBMR: ', score_lgbmr)
print('Mean squared error for LGBMR: ', mse_lgbmr)
```

```
R-squared for LGBMR:  0.9177191597106954
Mean squared error for LGBMR:  0.039122769165851036
```

Значення R-квадрат для моделі Light GBM складає 91,8%, а MSE похибка приблизно 0.039.

Повний код побудови моделей наведено у додатку В.

Отримавши значення похибок та коефіцієнтів лінійної детермінації можна визначити найкращу модель. На рисунках 3.17 – 3.19 можна побачити як реальні значення співвідносяться з прогнозованими для кожної побудованої моделі.

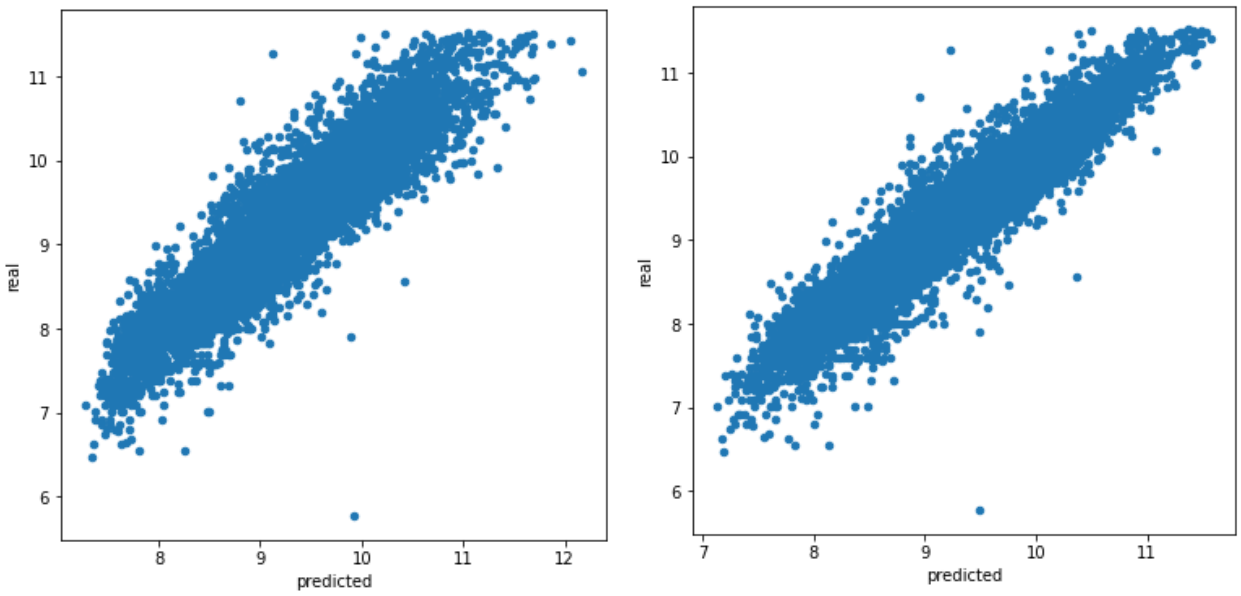


Рисунок 3.17 – Діаграма розсіювання справжніх значень до прогнозованих для лінійної регресії (зліва) та регресії опорних векторів (справа)

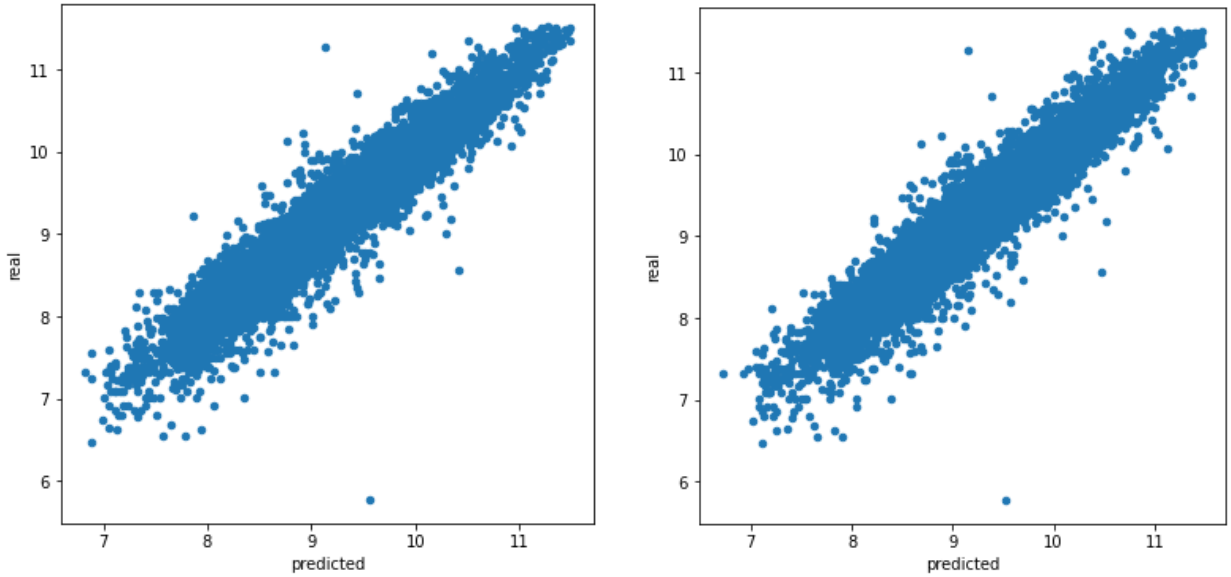


Рисунок 3.18 - Діаграма розсіювання справжніх значень до прогнозованих для градієнтного бустингу (зліва) та XGBoost (справа).

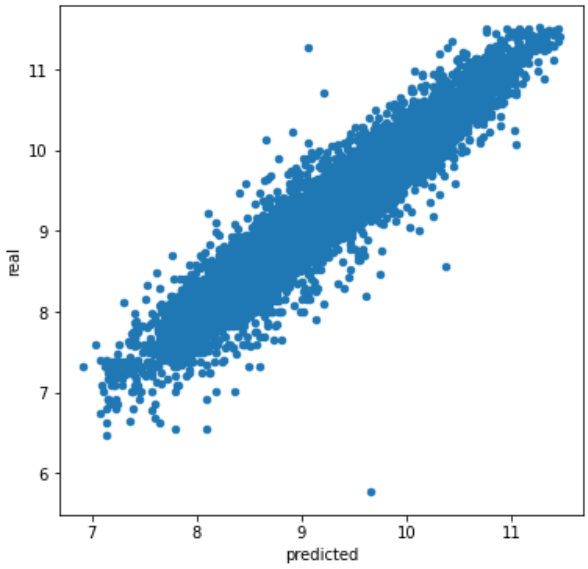


Рисунок 3.19 - Діаграма розсіювання справжніх значень до прогнозованих для Light GBM

Загалом, усі моделі показали гарні результати. Після побудови можна почати оцінювання та остаточний вибір для подальшого використання. На рисунку 3.20 можна побачити результати прогнозування моделей (коефіцієнт лінійної детермінації), а на рисунку 3.21 - середньоквадратичної похибки.

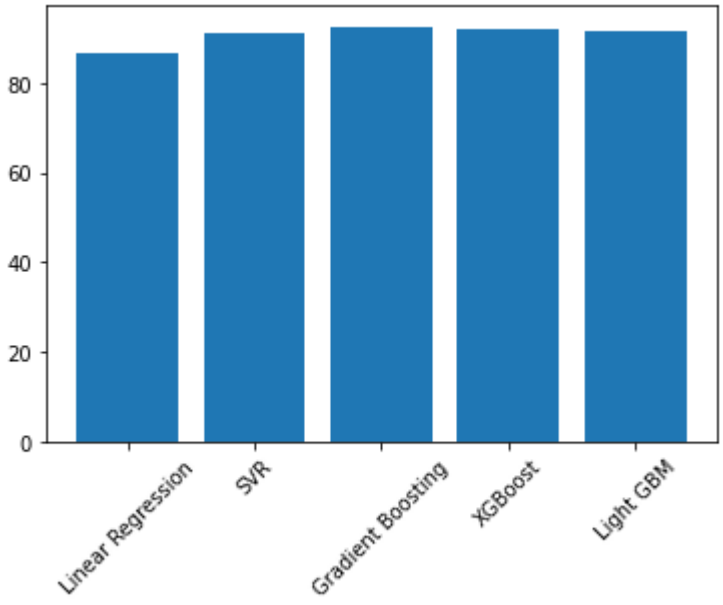


Рисунок 3.20 – Результат прогнозування моделей (R^2)

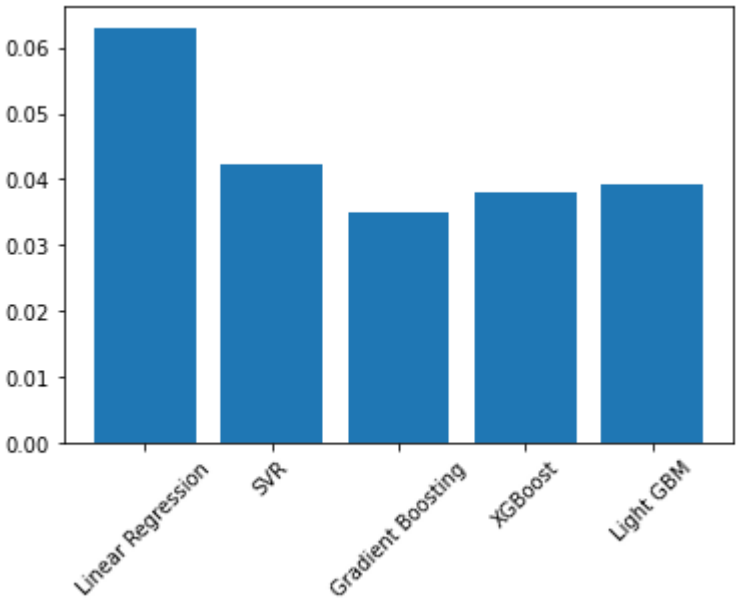


Рисунок 3.21 – Результат прогнозування моделей (MSE)

Отже, було побудовано п’ять моделей для прогнозування цін на автотранспортні засоби. Накращий результат продемонструвала модель градієнтного бустингу - коефіцієнт лінійної детермінації дорівнює 92,6%, а середньоквадратична похибка дорівню 0.035. Дана модель зі знайденими найкращими параметрами використовується для побудови інформаційного забезпечення.

РОЗДІЛ 4 РОЗРОБКА ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ ПРОГНОЗУВАННЯ ЦІН НА АВТОТРАНСПОРТНІ ЗАСОБИ

4.1 Вибір інструментів для реалізації інформаційного забезпечення прогнозування цін на автотранспортні засоби

Щоб створити складну веб-програму, ви можете створити власну програму з нуля або побудувати її, використовуючи фреймворк (який визначає структуру та надає набір бібліотек для загальних завдань). Rolling-your-own означає, що вам потрібно написати десять тисяч рядків стандартного коду, що вже передбачений фреймворком. З іншого боку, використання фреймворку означає, що вам потрібно витратити тижні або навіть місяці на читання та вивчення фреймворку, оскільки кожен фреймворк має свій власний «синтаксис», а отже, вимагає детального ознайомлення. Крім того, доступних фреймворків занадто багато, і вибір правильного фреймворка виявляється важким рішенням.

Flask — це невеликий мінімалістичний і легкий фреймворк Python Webapp. Завдання високого рівня, такі як доступ до бази даних, веб-форм та аутентифікація користувачів, підтримуються через «розширення». Фреймворк Flask надає інтерфейс відповідності WSGI, маршрутизацію URL-адрес і диспетчеризацію запитів, безпечні файли cookie та сеанси, вбудований веб-сервер і налагоджувач, клієнт модульного тестування для модульного тестування.

За допомогою Flask ви можете обробляти запити HTPP і AJAX, сеанси користувача між запитами, маршрутизувати запити до контролерів, оцінювати та перевіряти дані запиту, відповідати за допомогою HTML або JSON тощо [34].

Перевагами використання Flask можна назвати масштабованість. Так як Flask вважається мікрофреймоврком, це означає, що його можна використовувати для неймовірно швидкого розвитку проекту, наприклад веб-додатку. Якщо додаток на початку досить малий, але має потенціал для

швидкого розвитку, фреймворк Flask ідеальний вибір, адже він дуже простий у використанні та має досить невелику кількість залежностей, що дозволяє йому працювати безперебійно, навіть якщо програма збільшується і збільшується.

Основною функцією та перевагою Flask є його гнучкість. Мінімалістична природа Flask і його здатність до розробки малих веб-додатків робить його одним з найбільш гнучких фреймворків для розробки.

Flask також підтримує модульне програмування, де його функціональність може бути розділена на кілька взаємозамінних модулів. Кожен модуль діє як незалежний будівельний блок, який може виконувати одну частину функціональних можливостей. Разом це означає, що всі складові частини конструкції є гнучкими, рухомими та перевіряються самостійно.

Неминуче є деякі недоліки у «легкості» природи цього мікрофреймворка. Головним з них є те, що на відміну від Django, Flask не має великого набору інструментів. Це означає, що розробникам доведеться вручну додавати такі розширення, як бібліотеки. І якщо ви додасте величезну кількість розширень, це може почати сповільнювати саму програму через безліч запитів.

Також оскільки Flask настільки універсальний з точки зору того, з якими технологіями він може взаємодіяти, досить часто компанія, яка використовує Flask, несе додаткові витрати на підтримку цих технологій. Наприклад, якщо технологія, яка взаємодіє з вашим додатком Flask, застаріла або була припинена, компанії доведеться шукати нову сумісну. Чим складнішим стає додаток, тим вище потенційні витрати на обслуговування та впровадження.

Так як в результаті дослідження кваліфікаційної роботи магістра має бути створений веб-додаток прогнозування цін на автотранспортні засоби, а за планом цей веб-додаток буде мати не дуже багато фукнціональності та складної побудови, було вирішено використовувати фреймворк Flask для його розробки.

Docker — це метод контейнеризації, який використовується для розгортання програмних додатків як на хмарних платформах, так і на локальних серверах. За допомогою Docker досить легко створювати, розгортати та запускати програми.

За допомогою контейнерів розробники можуть упакувати свою програму в одну частину разом із усіма залежностями і мати можливість розгортати цю програму в будь-якому місці, не турбуючись про будь-які залежності. Це визначення контейнера, яке дає Docker: «Контейнер — це стандартна одиниця програмного забезпечення, яка упаковує код і всі його залежності, щоб програма швидко й надійно працювала від одного обчислювального середовища до іншого».

Heroku – це хмарна платформа, яка дозволяє розміщювати додатки в хмарі. Heroku повністю керована, що дає розробникам свободу зосередитися на своєму основному продукті, не відволікаючись на обслуговування серверів, обладнання чи інфраструктури. Heroku надає послуги, інструменти, робочі процеси та підтримку багатьох мов — усе розроблено для підвищення продуктивності розробників.

Flask, Docker, Heroku – це досить поширена зв'язка технологій для побудови веб-додатків у сфері Data Science. Остаточна архітектура додатку зображена на рисунку 4.1.

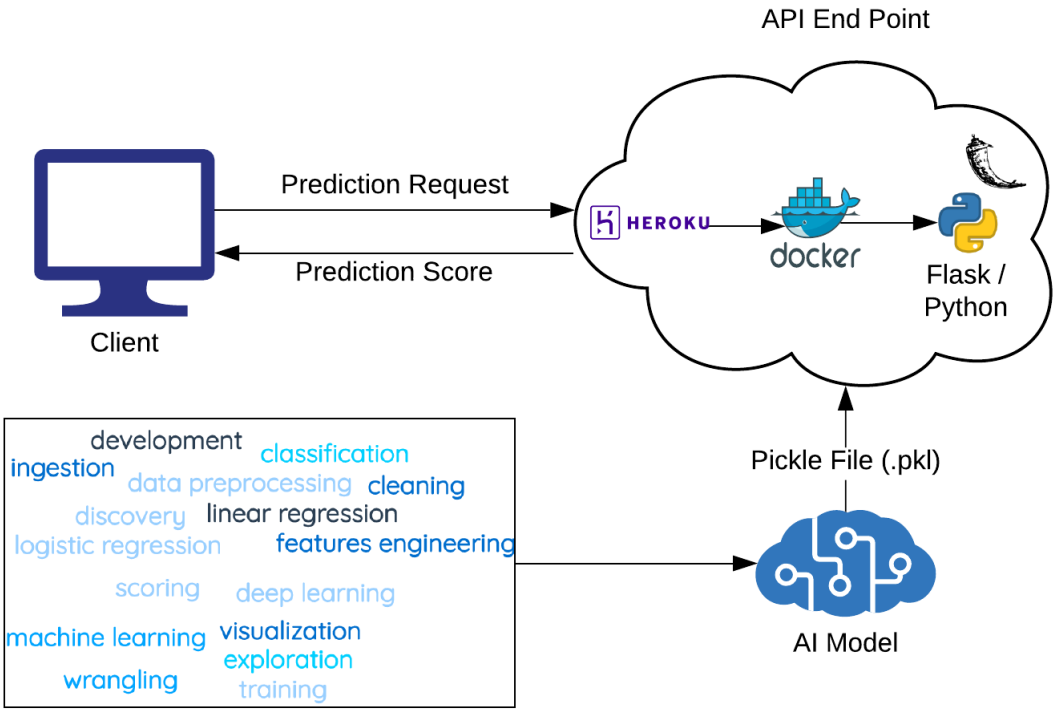


Рисунок 4.1 – Архітектура веб-додатку

4.2 Побудова інформаційного забезпечення для прогнозування

Для реалізації веб-додатку, що має інтерфейс для вводу параметрів вживаного автомобіля та виводу прогнозованої ціни, використовується веб-фреймворк Flask. Пропонований дизайн інтерфейсу додатку наведено на рисунку 4.2.

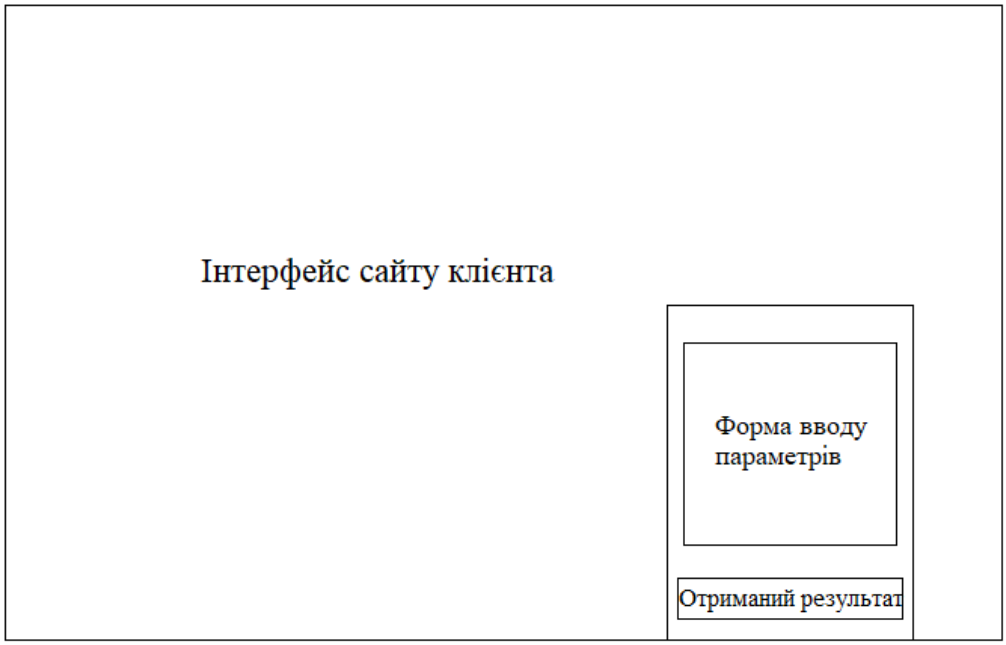


Рисунок 4.2 – Пропонований інтерфейс веб-додатку

Так як потрібний набір даних було вже зібрано, підготовлено, нормалізовано, вхідними даними для роботи будуть 2 файли csv формату, що містять тренувальний та тестувальний датасети. Також в попередній частині роботи було визначено найкращу модель та гіперпараметри, що вона приймає.

Тож завантажимо потрібні датасети та побудуємо потрібну нам модель використовуючи мову програмування Python.

```
train = pd.read_csv(TRAIN_CSV)
test = pd.read_csv(TEST_CSV)

X = train.drop("price_usd", axis=1)
y = train["price_usd"]

model = GradientBoostingRegressor(learning_rate=0.03, max_depth=8, n_estimators=1000, subsample=0.5)
model.fit(X, y)
with open('models/model.pickle', 'wb') as f:
    pickle.dump(model, f)
```

Отриману модель було збережено у файлі model.pickle, для подальшого використання у роботі.

Перевіримо продуктивність моделі.

```
test = pd.read_csv(TEST_CSV)
X = test.drop("price_usd", axis=1)
y = test["price_usd"]

loaded_model = pickle.load(open('models/model.pickle', 'rb'))
score = loaded_model.score(X, y)
print(score)
```

Для роботи з веб-додатком було створено два класи: DataLoader, який буде мати методи для попередньої обробки даних та Predictor, методи якого будуть реалізовувати завантаження моделі з файлу .pickle та передбачення результатів на базі цієї моделі. Повну структуру коду проекту веб-додатку наведено у Додатку Г.

Для початку роботи над класом DataLoader потрібно визначити вхідні параметри, що потрібні для попередньої обробки та подальшого передбачення. Вхідні параметри будуть визначати кількість, формат та поля у веб-застосунку. Аналізуючи перетворення даних у розділі 3 було виділено такі вхідні параметри:

— manufacturer – назва виробника з попередньо сформованого списку;

— drivetrain – тип приводу в розкодованому форматі;

— body_type – тип кузову з попередньо сформованого списку в розкодованому форматі;

— fuel – тип палива двигуна з попередньо сформованого списку в розкодованому форматі;

— engine_capacity – об’єм двигуна, числове значення;

— gear_box – тип коробки передач з попередньо сформованого списку в розкодованому форматі;

— year – рік виробництва автомобіля, числове значення;

— odometer_value - пробіг, числове значення;

— state – чи був автомобіль у ДТП, булеве значення.

Тепер визначимо операції обробки, що мають бути застосовані до цих значень. Для отримання атрибуту age потрібно провести перетворення над атрибутом year. За змінних, що визначаються з попередньо сформованих списків, створюються фіктивні змінні. Для числових змінних застосовано логарифмічну трансформацію. Код класу DataLoader відображено на рисунку 4.3.

```
class DataLoader(object):
  def fit(self, dataset):
    self.dataset = dataset.copy()

  def set_dummies_vb(self, x, df):
    dumblist = DUMMIES[x]
    all_dumb_features = [x + '_' + d for d in dumblist]
    df[all_dumb_features] = 0
    df[x + '_' + df[x].iloc[0]] = 1
    df = df.drop([x], axis=1)
    return df

  def load_data(self):
    self.dataset['age'] = 2022 - self.dataset['year']

    self.dataset = self.set_dummies_vb('manufacturer', self.dataset)
    self.dataset = self.set_dummies_vb('gear_box', self.dataset)
    self.dataset = self.set_dummies_vb('fuel', self.dataset)
    self.dataset = self.set_dummies_vb('body_type', self.dataset)
    self.dataset = self.set_dummies_vb('drivetrain', self.dataset)

    # drop columns
    drop_elements = ['year']

    self.dataset = self.dataset.drop(drop_elements, axis=1)

    self.dataset['odometer_value'] = np.log1p(self.dataset['odometer_value'])
    self.dataset['engine_capacity'] = np.log1p(self.dataset['engine_capacity'])
    self.dataset['age'] = np.log1p(self.dataset['age'])

    return self.dataset
```

Рисунок 4.3 – Код класу DataLoader

Наступним кроком у побудові Flask веб-додатку це створення API. Введення параметрів буде реалізовуватися через форму, отримані значення будуть сформовані у DataFrame, який після обробки методами класу DataLoader буде завантажено в модель. На виході маємо отримати прогнозоване значення ціни на автомобіль за введеними параметрами. Основний код програми, а саме ініціалізація Flask додатку та створення основного API наведено на рисунку 4.4.

app.py > ...

```python
import os
from flask import Flask, request, render_template
import pandas as pd
import numpy as np
from utils import Predictor
from utils import DataLoader

app = Flask(__name__)

@app.route("/")
def home():
    return render_template('index.html')

@app.route("/", methods=['POST'])
def predict():
    if request.method == 'POST':
        #access the data from form
        manufacturer = request.form['manufacturer']
        drivetrain = request.form['drivetrain']
        body_type = request.form['body_type']
        fuel = request.form['fuel']
        engine_capacity = float(request.form['engine_capacity'])
        gear_box = request.form['gear_box']
        year = int(request.form['year'])
        odometer_value = float(request.form['odometer_value'])
        if request.form.get('state') == None:
            state = 0
        else:
            state = 1

        df = pd.DataFrame(
            { 'manufacturer': manufacturer,
              'drivetrain': drivetrain,
              'body_type': body_type,
              'fuel': fuel,
              'engine_capacity': engine_capacity,
              'gear_box': gear_box,
              'year': year,
              'odometer_value': odometer_value,
              'state': state
            }, index=[0])

        loader = DataLoader()
        loader.fit(df)
        processed_df = loader.load_data()

        predictor = Predictor()
        response_dict = predictor.predict(processed_df).tolist()
        price = np.expm1(response_dict[0])
        output = round(price/100)*100
        return render_template("index.html", prediction_text='Прогнозована вартість автомобіля: $ {} - {}'.format(output, output + 200))


if __name__ == "__main__":
    port = int(os.environ.get('PORT', 5000))
    app.run(debug=True, host='0.0.0.0', port=port)
```

Рисунок 4.4 – Основний код програми

Наступним кроком у розробці веб-додатку є реалізація інтерфейсу користувача з-за допомогою веб-технологій – HTML, CSS, JavaScript. Фрагмент HTML шаблону для веб-додатку наведено у додатку Д.

Для демонстрації створеного інтерфейсу та імітації його роботи на реальному сайті з продажу автомобілей, було створено шаблон сайту. Демонстрація роботи веб-додатку наведено у додатку Ж.

Побудований веб-додаток прогнозування ціни на вживані автомобілі було розміщено у контейнер Docker та розгорнуто за допомогою хмарної платформи Heroku.

Реалізований веб-додаток можна знайти за посиланням - https://cars-price-dyploma.herokuapp.com/.

4.3 Практичне використання інформаційного забезпечення прогнозування цін на автотранспортні засоби у бізнесі

Існує велика різниця в цінах на конкретний автомобіль з певним пробігом і в певному стані. Знання того, яке значення використовувати, часто може збивати з пантелику.

Kelley Blue Book і Edmunds - два найвідоміших довідника з цінами на старі автомобілі в Сполучених Штатах. Kelley Blue Book і Edmunds пропонують дуже хорошу загальну довідкову інформацію, коли справа доходить до поточних ринкових умов та рівнів цін для конкретної марки і моделі уживаного автомобіля або вантажівки [16]. Однак значення, зазначені на кожному веб-сайті, ймовірно, будуть відрізнятися на кілька тисяч доларів.

Проте ринок уживаних автомобілів США відрізняється від українського, тому використовувати ці інструменти для ціноутворення в нашій країні не є доречним. Також для нашого ринку немає чіткого алгоритму та різні сайти і література пропонують різні способи визначення ціни автомобіля.

До того ж, продавці машин зазвичай не є експертами в автомобільній галузі і різноманітність показників, що можуть впливати на ціну зазвичай стають фатальними.

Реалізований веб-додаток на базі даних, що була створена з оголошень про продаж вживаних автомобілей в Україні, орієнтований на український ринок. Створений інтерфейс з формою введення параметрів дозволяє використовувати веб-додаток на будь-якому сайті чи веб-ресурсі. Такими сайтами можуть виступати як сайти зі списком оголошень на продаж автомобілей, так і власне сайти компаній, що займаються пошуком, оцінкою та оформленням автомобілей.

ВИСНОВКИ

В даний час спостерігається зростання вторинного ринку автомобілів в Україні. Проте з розвитком торгівлі, покупець не завжди може зрозуміти, чи відповідає ціна автомобіля його стану. У той же час продавці не завжди можуть адекватно оцінити вартість свого автомобіля.

В ході кваліфікаційної роботи магістра було розглянуто ціноутворення на вживані автомобілі. Було проаналізовано світову літературу, веб-ресурси, аналітику сайтів і наведено основні принципи формування ціни на автотранспортні засоби. Також були виокремлені основні показники, що впливають на ціну, які використовувалися при зборі інформації та формуванні датасету для моделювання.

На основі розглянутих досліджень з прогнозування цін на автомобілі, було вирішено застосовувати методи машинного навчання для вирішення задач регресії: лінійну регресію, регресію опорних векторів, методи градієнтного бустингу. Також була описана реалізація сформованого математичного апарату за допомогою комп'ютерних систем, а саме мови програмування Python та її бібліотек і пакетів.

База даних, на основі якої створюються моделі та аналізується ціноутворення, складається з інформації, зібраної з веб-ресурсу AUTO.RIA. Всього було зібрано 97043 екземпляри оголошень по продажу б/в авто. Цільовою змінною виступає «Ціна». Для моделювання цільової змінної використовується 9 незалежних змінних, таких як: назва виробника, тип коробки передач, пробіг, тип палива, тип кузову, стан машини (була в ДТП чи ні), тип приводу, об'єм двигуна та вік машини.

Перед моделюванням був проведений поверхневий аналіз отриманої бази даних, видалення пустих значень, заповнення пропусків, нормалізація та позбавлення від викидів. Також був проведений кореляційний аналіз, за результатами якого були виведені основні характеристики, що впливають на

ціну автомобіля: вік, пробіг, об’єм двигуна та тип приводу. Ці значення підтверджує аналіз літератури та досліджень ціноутворення на автомобілі.

Для прогнозування було обрано 5 моделей: лінійна регресія, модель градієнтного бустингу, регресія опорних векторів, XGBoost модель та Light GBM. Накращий результат продемонструвала модель градієнтного бустингу - коефіцієнт лінійної детермінації дорівнює 92,6%, а середньоквадратична похибка дорівнює 0.035.

На основі обраної моделі було побудований веб-додаток за допомогою комбінації інструментів Flask, Docker, Heroku та мови програмування Python. Розроблена програма може використовуватися як на різних сайтах (веб-ресурсах) з продажу вживаних автомобілей (AUTO.RIA, rst.ua, ab.ua), так і в компаніях, що надають послуги з продажі чи оцінки автотранспортних засобів.

Отриманий результат може бути покращений в майбутньому, наприклад, можна розглянути використання нейронних мереж для побудови моделей, збільшення датасету чи використання незадіяних атрибутів, як модель автомобіля. Також розроблений веб-додаток можна розширити, додавши нову функціональність: реалізувати постійне оновлення бази актуальною інформацією. З-за допомогою веб-розробників дизайн додатку може бути покращений та сама програма може бути випущена як додатковий сервіс для веб-сайтів.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Linear Regression in Machine Learning Definition, Advantage & uses [Електронний ресурс] – Режим доступу до ресурсу: https://www.mygreatlearning.com/blog/linear-regression-in-machine-learning.
2. What is Linear Regression? [Електронний ресурс] – Режим доступу до ресурсу: https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/what-is-linear-regression/.
3. Douglas C. Monthgomery. Introduction to Linear Regression Analysis / D. Montgomery, E. Peck, G. Vining. – New Jersey: John Wiley & Sons, Inc., 2013. – 672 с. – (5).
4. Webster A. Introductory Regression Analysis: with Computer Application for Business and Economics / Allen Webster., 2013. – 496 с.
5. Gunst R. Regression Analysis and its Application: A Data-Oriented Approach / R. Gunst, R. Mason., 2018. – 424 с.
6. Molnar C. Interpretable Machine Learning / Christoph Molnar., 2020. – 320 с.
7. Decision Tree Algorithm Explained with Examples [Електронний ресурс] – Режим доступу до ресурсу: https://www.mygreatlearning.com/blog/decision-tree-algorithm.
8. Pros and Cons of Decision Tree Regression in Machine Learning [Електронний ресурс] – Режим доступу до ресурсу: https://www.upgrad.com/blog/pros-and-cons-of-decision-tree-regression-in-machine-learning/.
9. Орельен Ж. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow / Жерон Орельен., 2018. – 688 с.
10. Support Vector Machine - Regression (SVR) [Електронний ресурс] – Режим доступу до ресурсу: https://www.saedsayad.com/support_vector_machine_reg.htm.

11. Unlocking the True Power of Support Vector Regression [Електронний ресурс] / Ashwin Raj. – 2020. – Режим доступу до ресурсу: https://towardsdatascience.com/unlocking-the-true-power-of-support-vector-regression-847fd123a4a0.

12. Gradient-Boosted Decision Trees (GBDT) [Електронний ресурс] – Режим доступу до ресурсу: https://c3.ai/glossary/data-science/gradient-boosted-decision-trees-gbdt/.

13. Masui T. All You Need to Know about Gradient Boosting Algorithm − Part 1. Regression [Електронний ресурс] / Tomonori Masui. – 2022. – Режим доступу до ресурсу: https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-1-regression-2520a34a502.

14. Brownlee J. A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning [Електронний ресурс] / Jason Brownlee. – 2016. – Режим доступу до ресурсу: https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/.

15. Mwiti D. Gradient Boosted Decision Trees [Guide]: a Conceptual Explanation [Електронний ресурс] / Derrick Mwiti – 2021. - Режим доступу до ресурсу: https://neptune.ai/blog/gradient-boosted-decision-trees-guide.

16. Edmunds. USED VEHICLE OUTLOOK 2019 / Edmunds.

17. Автопродажи в Украине в 2020 [Електронний ресурс] – Режим доступу до ресурсу: https://proautomoto.com/category/198-avtoprodazhi-v-ukraine-v-2020.

18. Автомобильный рынок Украины 2018 / 2019 [Електронний ресурс] – Режим доступу до ресурсу: https://inventure.com.ua/analytics/investments/avtomobilnyj-rynok-ukrainy-2018-2019.

19. Украинский рынок б/у авто бьет рекорды: какие модели популярны на "вторичке" [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://focus.ua/auto/496887-ukrainskiy-rynok-b-u-avto-bet-rekordy-kakie-modeli-populyarny-na-vtorichke.

20. Как формируется стоимость автомобиля [Електронний ресурс] – Режим доступу до ресурсу: https://vykupavto.pro/formirovanie-stoimosti-avtomobilya.html.

21. Retail definition [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Retail.

22. Car dealership definition [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Car_dealership.

23. How to Determine the Retail Value of Your Used Car [Електронний ресурс] – Режим доступу до ресурсу: https://retail-insider.com/articles/2020/08/how-to-determine-the-retail-value-of-your-used-car/.

24. Что влияет на стоимость автомобиля [Електронний ресурс] – Режим доступу до ресурсу: https://avtocod.ru/chto-vliyaet-na-stoimost-avtomobilya.

25. Mohri M. Foundations of Machine Learning, second edition / M. Mohri, A. Rostamizadeh, A. Talwlkar., 2018. – 504 с.

26. Ashish K. Learning Predictive Analytics with Python / Kumar Ashish., 2016. – 354 с.

27. Oredein A. ON VALIDATING REGRESSION MODELS WITH BOOTSTRAPS AND DATA SPLITTING TECHNIQUES [Електронний ресурс] / A. Oredein, T. Olatayo, A. Loyinmi // Global Journal of Science Frontier Research. – 2011. – Режим доступу до ресурсу: https://www.semanticscholar.org/paper/ON-VALIDATING-REGRESSION-MODELS-WITH-BOOTSTRAPS-AND-Oredein-Olatayo/80273bb965163d2ca189e66cccfced68f1aa375b.

28. Goja F. Predictive Modelers' Guide To Choosing The Best Fit Regression Model for Beginners [Електронний ресурс] / Freeman Goja. – 2020. – Режим доступу до ресурсу: https://towardsdatascience.com/predictive-modellers-guide-to-choosing-the-best-fit-regression-model-707120e502b4.

29. Nami Y. How to choose the best Linear Regression model [Електронний ресурс] / Yousef Nami. – 2020. – Режим доступу до ресурсу: https://towardsdatascience.com/how-to-choose-the-best-linear-regression-model-a-comprehensive-guide-for-beginners-754480768467.

30. How to Choose the Best Regression Model [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://blog.minitab.com/en/how-to-choose-the-best-regression-model.

31. Dhiraj K. Implementing Gradient Boosting in Python [Електронний ресурс] / K. Dhiraj. – 2020. – Режим доступу до ресурсу: https://blog.paperspace.com/implementing-gradient-boosting-regression-python/.

32. Jain A. Complete Guide to Parameter Tuning in XGBoost with codes in Python [Електронний ресурс] / A. Jain. – 2016. – Режим доступу до ресурсу: https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/.

33. Python: Data Analytics and Visualization / P.Vo, M. Czygan, A. Kumar, K. Raman., 2017. – 866 с.

34. Thanh N. Flask Web Development: Developing Web Applications with Python / Neos Thanh.. – 116 с.

35. Deery M. What Is Flask and How Do Developers Use It? [Електронний ресурс] / Matthew Deery. – 2021. – Режим доступу до ресурсу: https://careerfoundry.com/en/blog/web-development/what-is-flask/.

36. Wong K. Heroku + Docker in 10 Minutes [Електронний ресурс] / Kay Jan Wong. – 2022. – Режим доступу до ресурсу: https://towardsdatascience.com/heroku-docker-in-10-minutes-f4329c4fd72f#3032.

37. Калькулятор розрахунку вартості автомобіля. – Режим доступу до ресурсу: https://auto.ria.com/uk/price/average/.

38. On-line оценка автомобиля. – Режим доступу до ресурсу: http://trade-in.com.ua/.

39. Транспортні засоби: розрахунок вартості. – Режим доступу до ресурсу: https://www.me.gov.ua/Vehicles/CalculatePrice?lang=uk-UA.

ДОДАТКИ

ДОДАТОК А

Код для отримання ідентифікаційних номерів оголошень та збору даних з оголошень

```
import requests
import json
import numpy as np
import pandas as pd
import time

API_KEY = "NdtYlGe98VItX3IC0hJrtEzes5Ib64CcteUBvb9P"
ids = np.array([], dtype=object)
for x in range(1141):
    try:
        response =
requests.get(f"https://developers.ria.com/auto/search?api_key={API_KEY}&categor
y_id=1&currency=1&abroad=2&custom=1&damage=0&under_credit=0&confiscate
d_car=0&countpage=100&page={x}")
        result_ids = np.array(response.json()['result']['search_result']['ids'],
dtype=object)
        ids = np.append(ids, result_ids)
    except:
        print("Error!!!")

df = pd.DataFrame(columns = ['auto_id', 'manufacturer_name',
'manufacturer_name_eng', 'manufacturer_name_id', 'model_name', 'model_name_id',
'gear_box', 'color', 'odometer_value', 'year', 'fuel', 'fuel_name', 'body_type', 'state',
'drivetrain', 'price_usd'])
error_ids = np.array([], dtype=object)
```

```
i = 0
error_counter = 0
for x in range(114473):
    try:
        i = i + 1
        response =
requests.get(f"https://developers.ria.com/auto/info?api_key={API_KEY1}&auto_id=
{ids[x]}")
        result = response.json()
        autoData = result['autoData']

        df = df.append(
            {'auto_id': autoData['autoId'],
             'manufacturer_name': result['markName'],
             'manufacturer_name_eng': result['markNameEng'],
             'manufacturer_name_id': result['markId'],
             'model_name': result['modelNameEng'],
             'model_name_id': result['modelId'],
             'gear_box': autoData.get('gearBoxId', 6),
             'color': result['color'].get('eng', 'other'),
             'odometer_value': autoData['raceInt'],
             'year': autoData['year'],
             'fuel': autoData['fuelId'],
             'fuel_name': autoData['fuelName'],
             'body_type': autoData.get('bodyId', 1),
             'state': result['autoInfoBar']['damage'],
             'drivetrain': autoData.get('driveId', 4),
             'price_usd': result['USD']
```

```
            }, ignore_index = True)
        error_counter = 0
        if i % 100 == 0:
            print("Sleep for 30 seconds")
            time.sleep(30)
    except:
        print("Error!")
        error_counter = error_counter + 1
        error_ids = np.append(error_ids, ids[x])
ids = np.delete(ids, range(i))

df.to_csv('raw_cars.csv', index=False)
```

ДОДАТОК Б

Опис атрибутів бази даних

Назва	Опис	Тип	Структура
auto_id	Ідентифікаційний номер автомобіля в базі даних AUTO.RIA	int64	Унікальний номер
manufacturer_name	Назва виробника	object	Строка з оригінальною назвою виробника
manufacturer_name_eng	Назва виробника англійською	object	Строка з оригінальною назвою виробника англійською мовою
namufacturer_name_id	Ідентифікаційний номер виробника в базі даних AUTO.RIA	int64	0...55173
model_name	Назва моделі	object	Строка з оригінальною назвою моделі
model_name_id	Ідентифікаційний номер моделі в базі даних AUTO.RIA	int64	0...63436
gear_box	Тип коробки передач	int64	{ 1: "manual",

			2: "automatic", 3: "typtronik", 4: "robotic", 5: "variable", 6: "other" }
color	Колір кузову	object	array(['gray', 'black', 'other', 'darkblue', 'red', 'orange', 'white', 'beige', 'brown', 'purple', 'yellow', 'green'], dtype=object)
odometer_value	Пробіг	int64	0...1000
year	Рік випуску	int64	1937...2022

fuel	Тип палива двигуна	int64	{
			1: "petrol",
			2: "diesel",
			3: "gas",
			4: "gas/petrol",
			5: "hybrid",
			6: "electro",
			}

fuel_name	Опис двигуна з об’ємом двигуна	object	Строка з типом пального та об’ємом двигуна (літраж)
body_type	Тип кузову	int64	{

			2: "station wagon", 3: "sedan", 4: "hatchback", 5:"SUV/crossover", 6: "coupe", 7: "cabriolet", 8: "minivan", 9: "pickup", 28: "other", 252:"limousine", 254: "passenger van", 307: "liftback", 315: "roadster" }
state	Чи був в ДТП	bool	0 – не був у ДТП 1 – був в ДТП

drivetrain	Тип трансмісії (приводу)	int64	{
			1: "all",
			2: "front",
			3: "back",
			4: "other"
			}

price_usd	Ціна	int64	0…11111111

ДОДАТОК В

Код побудови моделей на мові Python

```
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.ensemble import GradientBoostingRegressor
from xgboost.sklearn import XGBRegressor
from lightgbm import LGBMRegressor
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import GridSearchCV

best_params = []
algos = {
    'LinearRegression' : {
        'model': LinearRegression(),
        'params': {
            'normalize': [True, False]
        }
    },
    'LGBMRegressor': {
        'model' : LGBMRegressor(),
        'params': {
            'boosting_type': ['gbdt', 'dart', 'goss'],
            'n_estimators': [i for i in range(10, 1000, 100)],
            'num_leaves': [7, 14, 50, 100, 150, 300],
            'learning_rate': [0.1, 0.03, 0.003],
            'max_depth': [3, 5, 8, None],
            'max_bin': [200, 300],
            'bagging_fraction': [0.2, 0.5, 0.95]
```

```
        }
    },
    'XGBRegressor': {
        'model' : XGBRegressor(),
        'params': {
            'eta': [0.01, 0.2, 0.3],
            'max_depth': [3, 5, 6, 8, None],
            'n_estimators': [i for i in range(10, 100, 20)],
            'learning_rate': [0.1, 0.2, 0.03, 0.003],
            'min_child_weight': [i for i in range(1, 12, 4)],
        }
    },
    'GradientBoostingRegressor': {
        'model' : GradientBoostingRegressor(),
        'params': {
            'learning_rate': [0.01,0.03,0.003],
            'subsample'    : [0.9, 0.5, 0.2, 0.1],
            'n_estimators' : [100,500,1000],
            'max_depth'    : [4,6,8,10],
        }
    },
    'SVR': {
        'model' : SVR(),
        'params': {
            'C':[1.5, 10, 100, 1000],
            'gamma': [1e-7, 1e-4, 0.1, 0.6],
            'epsilon':[0.1,0.2,0.5,0.3]
        }
    }
}
```

```
cv = ShuffleSplit(n_splits=5, test_size=0.2, random_state=0)
for algo_name, config in algos.items():
    gs = GridSearchCV(config['model'], config['params'], cv=cv, scoring='r2',
return_train_score=False, verbose=2)
    gs.fit(x_train,y_train)
    best_params.append({
        'model': algo_name,
        'best_params': gs.best_params_,
        'best_estimator': gs.best_estimator_
    })

lm = LinearRegression(normalize=True)
lm.fit(x_train, y_train)
y_pred_lr = lm.predict(x_test)
score_lr = lm.score(x_test, y_test)
mse_lr = mean_squared_error(y_test, y_pred_lr)
print('R-squared for Linear Regression: ', score_lr)
print('Mean squared error for Linear Regression: ', mse_lr)

svr = SVR()
svr.fit(x_train, y_train)
y_pred_svr = svr.predict(x_test)
score_svr = svr.score(x_test, y_test)
mse_svr = mean_squared_error(y_test, y_pred_svr)
print('R-squared for SVR: ', score_svr)
print('Mean squared error for SVR: ', mse_svr)

gb = GradientBoostingRegressor(learning_rate=0.03, max_depth=8,
n_estimators=1000, subsample=0.5)
gb.fit(x_train, y_train)
```

```
y_pred_gb = gb.predict(x_test)
score_gb = gb.score(x_test, y_test)
mse_gb = mean_squared_error(y_test, y_pred_gb)
print('R-squared for Gradient Boosting: ', score_gb)
print('Mean squared error for Gradient Boosting: ', mse_gb)

xgb = XGBRegressor(learning_rate=0.2, max_depth=8, n_estimators=90)
xgb.fit(x_train, y_train)
y_pred_xgb = xgb.predict(x_test)
score_xgb = xgb.score(x_test, y_test)
mse_xgb = mean_squared_error(y_test, y_pred_xgb)
print('R-squared for XGBoost: ', score_xgb)
print('Mean squared error for XGBoost: ', mse_xgb)

lgbmr = LGBMRegressor(n_estimators=90, num_leaves=50)
lgbmr.fit(x_train, y_train)
y_pred_lgbmr = lgbmr.predict(x_test)
score_lgbmr = lgbmr.score(x_test, y_test)
mse_lgbmr = mean_squared_error(y_test, y_pred_lgbmr)
print('R-squared for LGBMR: ', score_lgbmr)
print('Mean squared error for LGBMR: ', mse_lgbmr)
```

ДОДАТОК Г

Структура проекту веб-додатку

app

```
├── data                - містить тренувальний та тестувальний датасети
│   ├── train.csv       - тренувальний датасет
│   └── test.csv        - тестовий датасет
├── models              - містить навчену модель
│   └── model.pickle
│
├── static              - містить статичні файли для побудови інтерфейсу
│   ├── css             - файли стилів сторінки та веб-додатку
│   ├── images          - картинки для веб-сторінки
│   ├── js              - файли коду мови програмування JavaScript для
функціональності веб-сторінки
│
├── templates           - містить шаблони сторінки
│   ├── index.html      - розмітка сторінки та веб-додатку
├── utils               - містить інструменти, з-за допомогою яких, працюємо
з датасетом
│   ├── __init__.py     - файл ініціалізації пакету
│   ├── dataloader.py   - завантажувач даних (клас обробки даних)
│   ├── predictor.py    - клас роботи з моделлю
├── constants.py        - зберігання констант для їх зручного використання
│
├── app.py              - основний файл коду
├── requirements.txt    - список бібліотек, що використовуються для Docker
├── Procfile
└── Dockerfile
```

ДОДАТОК Д

Фрагмент HTML-коду для веб-додатку

```
<div class="predictor">
    <header class="clearfix">
      <h4>Дізнатися ціну на автомобіль</h4>
    </header>
    <div class="chat">
      <div class="chat-history">
        <form class="prediction-form" action="" method="post">
          <label for="manufacturer" colo>Виробник: </label>
          <select id="manufacturer" name="manufacturer" , required="required">
            <option value="audi">Audi</option>
            <option value="volkswagen">Volkswagen</option>
            <option value="mitsubishi">Mitsubishi</option>
            <option value="renault">Renault</option>
            <option value="toyota">Toyota</option>
            <option value="bmw">BMW</option>
            <option value="hyundai">Hyundai</option>
            <option value="citroen">Citroen</option>
            <option value="lexus">Lexus</option>
            <option value="mazda">Mazda</option>
            <option value="ford">Ford</option>
            <option value="jeep">Jeep</option>
            <option value="opel">Opel</option>
            <option value="porsche">Porsche</option>
            <option value="kia">KIA</option>
            <option value="honda">Honda</option>
            <option value="nissan">Nissan</option>
            <option value="skoda">Skoda</option>
```

<option value="mercedes-benz">Mercedes-Benz</option>
<option value="subaru">Subaru</option>
<option value="peugeot">Peugeot</option>
<option value="volvo">Volvo</option>
<option value="chevrolet">Chevrolet</option>
<option value="mini">Mini</option>
<option value="land-rover">Land-Rover</option>
<option value="daewoo">Daewoo</option>
<option value="vaz">ВАЗ</option>
<option value="dodge">Dodge</option>
<option value="infiniti">Infiniti</option>
<option value="dacia">Dacia</option>
<option value="jaguar">Jaguar</option>
<option value="cadillac">Cadillac</option>
<option value="zaz">ЗАЗ</option>
<option value="ssangyong">Ssangyong</option>
<option value="acura">Acura</option>
<option value="daihatsu">Daihatsu</option>
<option value="chery">Chery</option>
<option value="fiat">Fiat</option>
<option value="seat">Seat</option>
<option value="lincoln">Lincoln</option>
<option value="suzuki">Suzuki</option>
<option value="chrysler">Chrysler</option>
<option value="alfa-romeo">Alfa-romeo</option>
<option value="smart">Smart</option>
<option value="geely">Geely</option>
<option value="buick">Buick</option>
<option value="gaz">ГАЗ</option>
<option value="bogdan">Богдан</option>

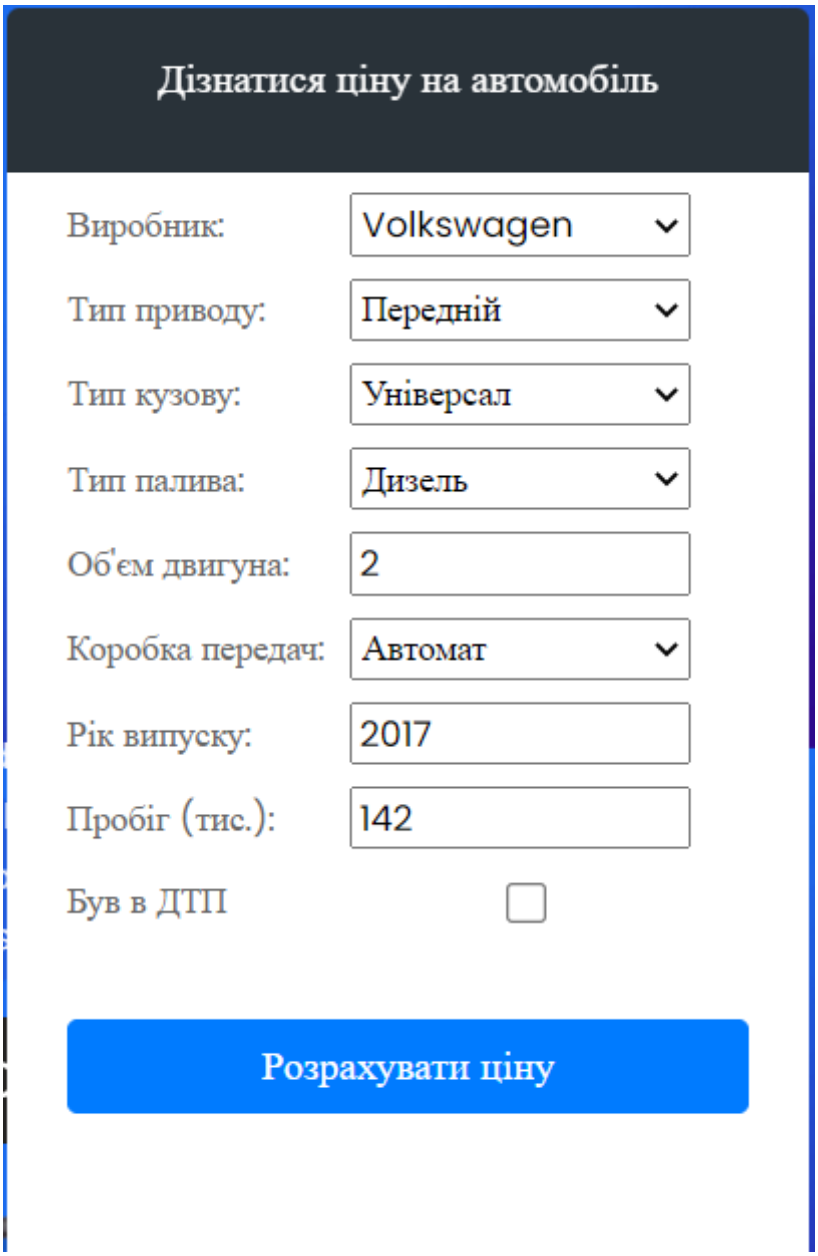
```
  <option value="lada">Lada</option>
  <option value="maserati">Maserati</option>
  <option value="gmc">GMC</option>
  <option value="great-wall">Great-Wall</option>
  <option value="tesla">Tesla</option>
  <option value="other">Інше</option>
</select>
<br>
<label for="drivetrain" colo>Тип приводу: </label>
<select id="drivetrain" , name="drivetrain" , required="required">
  <option value="all">Повний</option>
  <option value="front">Передній</option>
  <option value="back">Задній</option>
  <option value="other">Інше/Невідомо</option>
</select>
<br>
<label for="body_type" colo>Тип кузову: </label>
<select id="body_type" , name="body_type" , required="required">
  <option value="station wagon">Універсал</option>
  <option value="sedan">Седан</option>
  <option value="hatchback">Хетчбек</option>
  <option value="SUV/crossover">Позашляховик/Кросовер</option>
  <option value="coupe">Купе</option>
  <option value="cabriolet">Кабріолет</option>
  <option value="minivan">Мінівен</option>
  <option value="pickup">Пікап</option>
  <option value="limousine">Лімузин</option>
  <option value="passenger van">Легковий фургон</option>
  <option value="liftback">Ліфтбек</option>
  <option value="other">Інше/Невідомо</option>
```

```
        </select>
        <br>
        <label for="fuel">Тип палива: </label>
        <select id="fuel" name="fuel" , required="required">
          <option value="petrol">Бензин</option>
          <option value="diesel">Дизель</option>
          <option value="gas">Газ</option>
          <option value="gas/petrol">Газ/бензин</option>
          <option value="hybrid">Гібрид</option>
          <option value="electro">Електро</option>
        </select>
        <br>
        <label for="engine_capacity">Об'єм двигуна: </label>
        <input id="engine_capacity" name="engine_capacity" type="number" min="0"
step="0.1"
          required="required"></input>
        <br>
        <label for="gear_box">Коробка передач: </label>
        <select id="gear_box" name="gear_box" , required="required">
          <option value="manual">Механічна</option>
          <option value="automatic">Автомат</option>
          <option value="typtronik">Тіпротонік</option>
          <option value="robotic">Адаптивна</option>
          <option value="variable">Варіатор</option>
          <option value="other">Інше</option>
        </select>
        <br>
        <label for="year">Рік випуску: </label>
        <input id="year" name="year" type="number" min="1995" step="1"
required="required"></input>
```

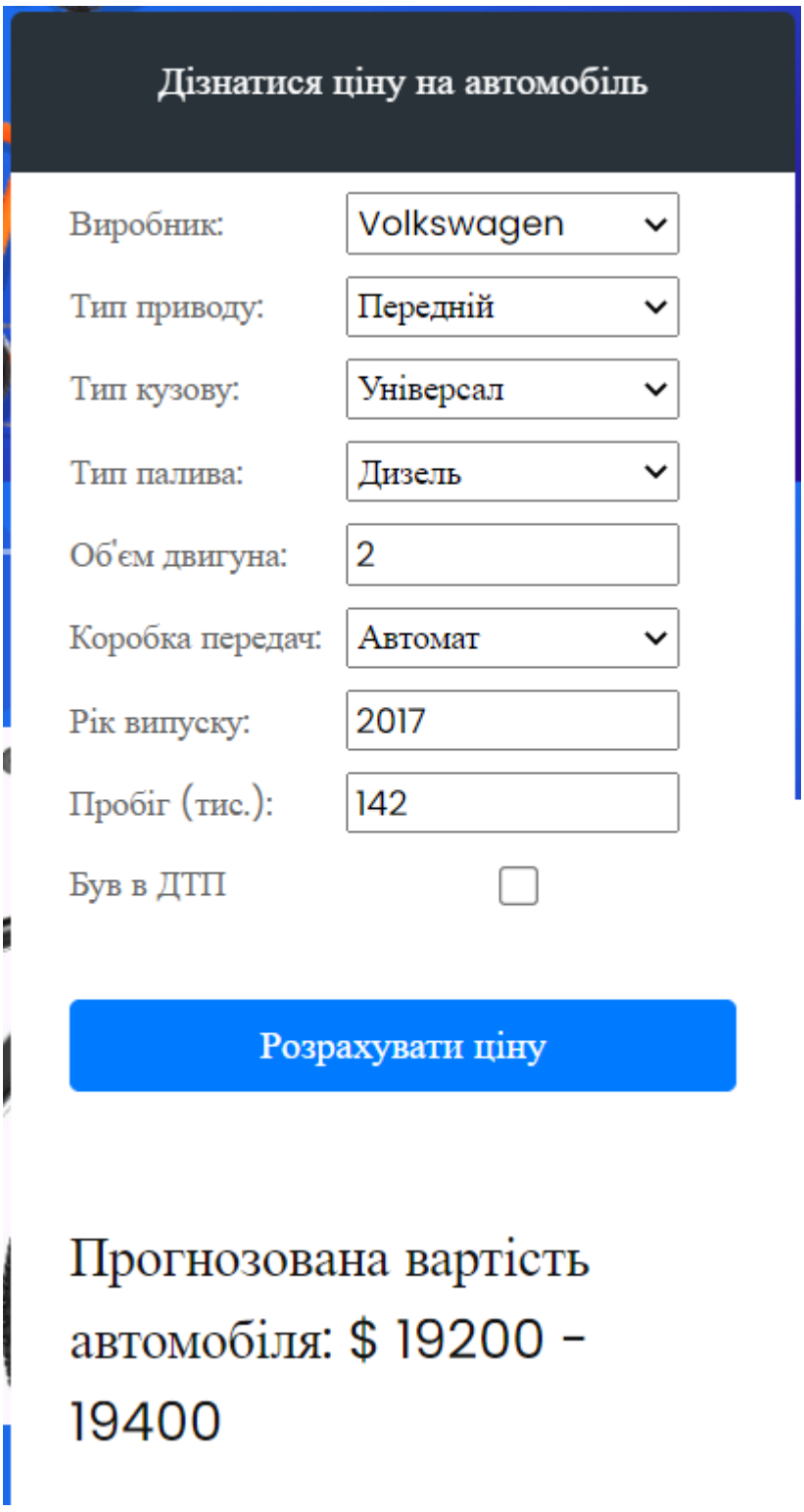
```
        <br>
        <label for="odometer_value">Пробіг (тис.): </label>
        <input id="odometer_value" name="odometer_value" type="number" min="0"
required="required"></input>
        <br>
        <div style="display: flex; align-items: center;">
          <label for="state">Був в ДТП</label>
          <input id="state" name="state" type="checkbox" value="off"
            style="margin-bottom: 0.5rem; margin-left: 6px; height: 16px"></input>
        </div>
        <br>
        <button type="submit" class="btn btn-primary btn-block btn-
large">Розрахувати ціну</button>
      </form>
      <br>
      <br>
      <h2>
        {{ prediction_text }}
      </h2>
    </div>
  </div>
</div>
```

ДОДАТОК Ж

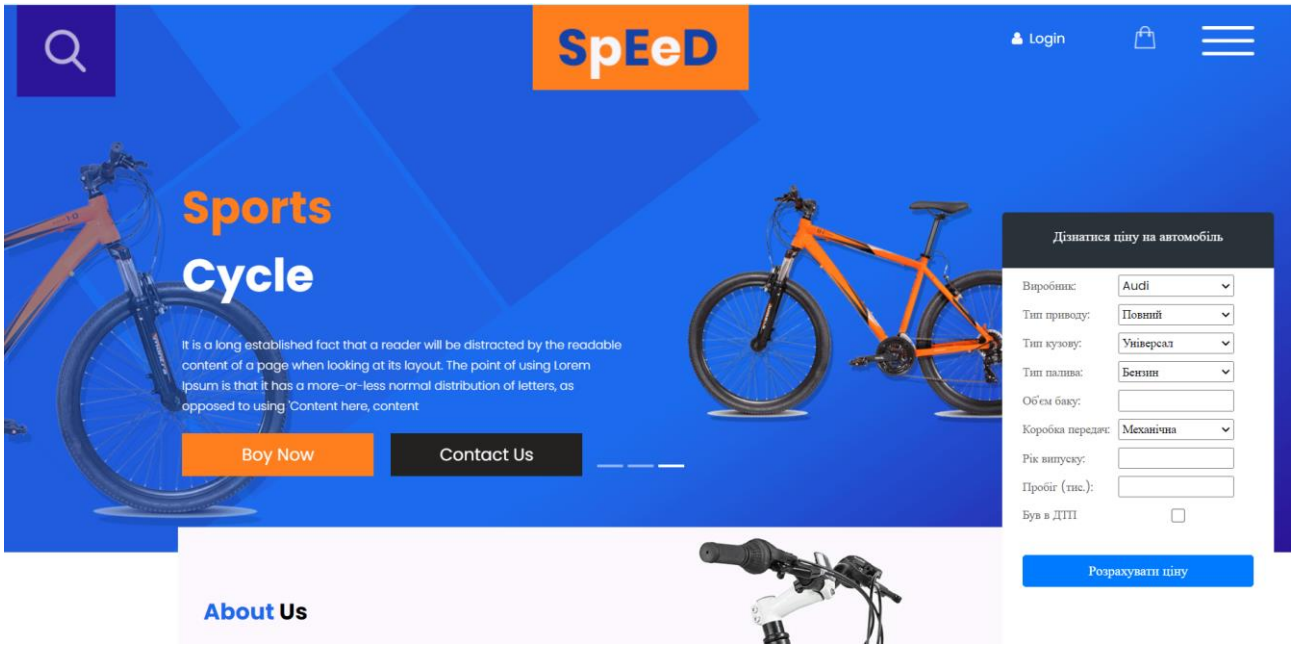
Приклад роботи створеного веб-додатку прогнозування ціни на автомобіль



Введення параметрів для розрахунку прогнозованої ціни



Отримана прогнозована вартість автомобіля в залежності від введених параметрів



Інтерфейс сайту шаблону та розробленого веб-додатку