МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп`ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА НЕЙРОННОЇ МЕРЕЖІ ДЛЯ ВИЗНАЧЕННЯ ВАРТОСТІ АВТОМОБІЛЯ»**

Виконав: студент 2 курсу, групи 8.1223
спеціальності 122 Комп`ютерні науки
(шифр і назва спеціальності)
освітньої програми Комп`ютерні науки
(назва освітньої програми)
М.С. Цис
(ініціали та прізвище)
Керівник доцент кафедри комп`ютерних наук, доцент, к.т.н. Борю С.Ю.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент завідувач кафедри програмної інженерії, доцент, к.ф.-м.н. Лісняк А.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп`ютерних наук

Рівень вищої освіти магістр

Спеціальність 122 Комп`ютерні науки
(шифр і назва)

Освітня програма Комп`ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри комп`ютерних наук,
д.т.н., професор

_____ Шило Г.М.
(підпис)

“ 05 ” 03 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Цису Максиму Сергійовичу
(прізвище, ім`я та по-батькові)

1. Тема роботи Розробка нейронної мережі для визначення вартості автомобіля

керівник роботи Борю Сергій Юрійович, к.т.н., доцент
(прізвище, ім`я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 1 » березня 2024 року № 385-с

2. Строк подання студентом роботи 26.11.2024 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розробка нейронної мережі для визначення вартості автомобілів.

5. Перелік графічного матеріалу (з точним зазначенням обов`язкових креслень)
презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 04.03.2024 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	22.03.2024	
2.	Збір вихідних даних.	9.04.2024	
3.	Обробка методичних та теоретичних	28.04.2024	
	джерел.		
4.	Розробка першого та другого розділу.	11.05.2024	
5.	Розробка третього розділу.	16.06.2024	
6.	Оформлення та нормоконтроль	26.11.2024	
	кваліфікаційної роботи магістра.		
7.	Захист кваліфікаційної роботи.	12.12.2024	

Студент _____ М.С. Цис
(підпис) (ініціали та прізвище)

Керівник роботи _____ С.Ю. Борю
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г.Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка нейронної мережі для визначення вартості автомобіля»: 53 с., 59 рис., 0 табл., 9 джерел, 1 додатків.

НЕЙРОННА МЕРЕЖА, NUMPY, PANDAS, PYTHON, TENSORFLOW, TKINTER.

Об`єкт дослідження – ринкова вартість автомобілів і їх характеристики у відкритих джерелах.

Мета роботи полягає у створенні нейронної мережі для визначення вартості автомобілів.

Метод дослідження – аналіз даних, та оцінку ефективності нейронної мережі.

Дана робота присвячена розробці програмного забезпечення для прогнозування вартості автомобілів на основі їх технічних характеристик. У роботі створено нейронну мережу з використанням бібліотеки TensorFlow, що дозволяє аналізувати введені користувачем дані та передбачати ціну автомобіля. Для підготовки даних застосовано бібліотеки Pandas і Numpy, що забезпечило точність моделі та ефективну роботу з великими масивами інформації. Результати роботи можуть бути використані у сфері продажу автомобілів, зокрема для оцінки ринкової вартості транспортних засобів.

SUMMARY

Master`s Qualification Thesis "Development of a neural network for determining the cost of a car": 53 p., 59 figures, 0 tables, 9 sources, 1 appendix.

NEURAL NETWORK, NUMPY, PANDAS, PYTHON, TENSORFLOW, TKINTER.

The object of the study is the market value of cars and their characteristics in open sources.

The purpose of the work is to create a neural network for detection the cost of cars.

The research method is data analysis and evaluation of the effectiveness of the neural network.

This work is devoted to the development of software for predicting the value of cars based on their technical characteristics. In the work, a neural network was created using the TensorFlow library, which allows analyzing the data entered by the user and predicting the price of the car. The Pandas and Numpy libraries were used for data preparation, which ensured the accuracy of the model and efficient work with large arrays of information. The results of the work can be used in the field of car sales, in particular to estimate the market value of vehicles.

ЗМІСТ

Завдання на кваліфікаційну роботу……..2
Реферат 4
Summary 5
Вступ..... 8
1 Аналіз задачі та вибір інструментів 10
1.1 Аналіз задачі прогнозування вартості автомобілів. 10
1.2 Визначення основних факторів, що впливають на ціну автомобіля . 11
1.3 Аналіз існуючих рішень у сфері оцінки вартості автомобілів..... 13
1.4 Опис джерела даних..... 15
1.5 Вибір програмного забезпечення та бібліотек..... 17
1.6 Matplotlib та seaborn для візуалізації..... 24
2 Розробка програмного забезпечення для прогнозування ціни..... 25
2.1 Опис вихідного набору даних..... 25
2.2 Аналіз структури даних і виявлення проблем 26
2.3 Перетворення категоріальних змінних у числовий формат 26
2.4 Масштабування (нормалізація) даних 27
2.5 Розділення даних на тренувальний і тестовий набори..... 28
2.6 Розробка структури нейронної мережі. 28
2.5 Вибір гіперпараметрів (кількість шарів, нейронів, dropout) 31
2.7 Налаштування процесу навчання (кількість епох, розмір батчу) 31
2.8 Використання механізму earlystopping..... 33
2.9 Eda (exploratiry data analysis)..... 33
3 Розробка графічного інтерфейсу та тестування системи..... 38
3.1 Метрики оцінки моделі (mse, rmse) 38
3.2 Використання tkinter для створення gui 39
3.3 Основні елементи інтерфейсу..... 40
3.4 Інтеграція моделі нейронної мережі з gui 43

3.5 Тестування gui на реальних і симульованих даних 43
3.6 Аналіз роботи системи, швидкість, точність 44
3.7 Рекомендації щодо практичного використання 45
Висновки 46
Перелік посилань 49
Додаток А Код програми 50

ВСТУП

У сучасному світі автомобіль є не лише засобом пересування, але й важливим активом, вартість якого залежить від багатьох чинників. Прогнозування ціни автомобіля є актуальним завданням як для продавців, так і для покупців, адже воно дозволяє забезпечити прозорість угод та оцінити ринкову вартість транспортного засобу.

Актуальність теми дослідження полягає у зростаючій популярності цифрових інструментів у сфері торгівлі автомобілями. З розвитком машинного навчання з`явилася можливість створювати системи, які аналізують великий обсяг даних і прогнозують з високою точністю. Ці системи зчитують такі дані як характеристики автомобіля, його технічний стан, пробіг, рік випуску та інші параметри, що значно полегшує прийняття рішень під час купівлі – продажу.

Об`єктом дослідження у даній роботі є програмне забезпечення для прогнозування вартості автомобілів, яке базується на застосуванні методів машинного навчання. Предметом дослідження є процес побудови та навчання нейронної мережі, а також інтеграція її в графічний інтерфейс для кінцевих користувачів.

Метою роботи є розробка ефективної системи прогнозування вартості автомобілів на основі їх характеристик та стану. Для досягнення поставленої мети вирішуються наступні завдання:

- аналіз існуючих методів та інструментів для прогнозування ціни автомобілів;
- розробка нейронної мережі з використанням бібліотеки TensorFlow;
- реалізація процесу підготовки даних для навчання моделі;
- створення графічного інтерфейсу для інтерактивного введення даних користувачем;

– тестування системи на практичних прикладах та аналіз точності прогнозів.

У роботі використав сучасні методи машинного навчання та бібліотеки Python, такі як TensorFlow, Pandas, Numpy. Використовував бібліотеку Tkinter для графічного інтерфейсу.

На практиці можна буде використовувати у сфері торгівлі автомобілями для оцінки їх ринкової вартості. Крім того, розроблена система може слугувати основою для подальшого вдосконалення та адаптації під інші потреби, зокрема прогнозування вартості іншого типу майна або активів.

Результати дослідження можуть бути корисними для компаній, що займаються продажем автомобілів, страхових агентств, а також для кінцевих споживачів, які бажають швидко та точно оцінити вартість транспортного засобу.

1 АНАЛІЗ ЗАДАЧІ ТА ВИБІР ІНСТРУМЕНТІВ

1.1 Аналіз задачі прогнозування вартості автомобілів.

Прогнозування вартості автомобілів є важливою задачею для багатьох секторів, таких як автомобільна торгівля, страхування, банківська справа та інші. У сучасному світі, коли ринок автомобілів динамічно змінюється, важливо мати інструменти, які допомагають точно оцінити вартість транспортного засобу. Одним із таких інструментів є використання машинного навчання для прогнозування ціни автомобіля на основі різноманітних факторів, які можуть включати характеристики автомобіля, історію його використання та інші зовнішні чинники.

Опис задачі прогнозування вартості автомобіля: Задача прогнозування вартості автомобіля є прикладом задачі регресії, де потрібно передбачити числову величину – ціну автомобіля. Це складна задача, оскільки вартість автомобіля залежить від багатьох змінних, таких як:

- марка та модель автомобіля: деякі марки, як BMW, Mercedes – Benz, Tesla мають високі ціни навіть для старих моделей завдяки бренду;
- рік випуску: автомобілі новіших років випуску, як правило, коштують дорожче;
- пробіг: чим менший пробіг, тим вища ймовірність, що автомобіль буде коштувати більше, оскільки він менше зношений;
- тип палива: електричні або гібридні автомобілі можуть коштувати більше завдяки новітнім технологіям;
- стан автомобіля: стан також має велике значення – автомобілі, які не були в аваріях і зберегли свій технічний стан, коштують дорожче.

Прогнозування вартості є важливим для:

– автосалонів та дилерів: для того, щоб правильно оцінити транспортний засіб, запропонувати конкурентну ціну та уникнути переплати за автомобіль;
– страхових компаній: для визначення вартості страховки на основі ринкової ціни автомобіля;
– банків: для оцінки кредитоспроможності покупця при видачі кредитів на купівлю автомобіля.

Приклад із роботи: вирішується задача прогнозування вартості автомобіля на основі даних, які містять різноманітні характеристики автомобіля, такі як:

– марка: bmw, audi, ford;
– рік випуску: 2015, 2020;
– пробіг: 100,000 км, 20,000 км;
– тип палива: бензин, дизель, електричний, гібрид;
– стан: новий, уживаний.

Прогнозування вартості базується на цих характеристиках, причому кожна з них має певний вплив на кінцеву ціну.

Приклад 1 Два автомобілі:

– автомобіль 1: bmw 2019 року з пробігом 30,000 км і типом палива – бензин;
– автомобіль 2: ford 2015 року з пробігом 150,000 км і типом палива – дизель.

То ціна Автомобіля 1 буде значно вищою, ніж ціна Автомобіля 2, через марку, рік випуску і значно менший пробіг.

1.2 Визначення основних факторів, що впливають на ціну автомобіля

Прогнозування вартості автомобіля є складним завданням, оскільки на кінцеву ціну можуть впливати різні фактори, від очевидних характеристик автомобіля, таких як марка, модель і рік випуску, до менш помітних, таких як

історія аварій, технічний стан. Правильне визначення факторів, що мають найбільший вплив на ціну, є критично важливим для побудови ефективної моделі прогнозування. Розглянемо основні фактори, які зазвичай впливають на ціну автомобіля, і на прикладі вашої роботи – як ці фактори можна використовувати для прогнозування вартості.

Рік випуску один з найочевидніших факторів. Як правило, автомобілі новіших років випуску коштують дорожче, оскільки вони мають новіші технології, менший пробіг і не такий великий знос. Для автосалонів і продавців рік випуску автомобіля може бути основним параметром для визначення його ринкової вартості.

Пробіг автомобіля є ще одним важливим фактором. Зазвичай, чим менший пробіг автомобіля, тим більша ймовірність, що його ціна буде вищою. Це пов`язано з тим, що автомобілі з меншим пробігом зазвичай перебувають у кращому технічному стані, потребують менше витрат на обслуговування і мають більший залишковий ресурс.

Марка автомобіля часто є визначальними факторами ціни, оскільки вони впливають на статус автомобіля, його надійність, витрати на обслуговування і навіть попит на ринку. Автомобілі від відомих брендів, таких як BMW, Mercedes – Benz, Tesla мають значно вищу вартість, ніж машини менш відомих марок.

Тип палива також впливає на ціну автомобіля. В останні роки особливо зростає попит на електричні автомобілі (наприклад, Tesla), а також гібриди (наприклад, Toyota Prius). Зважаючи на екологічні тенденції, державні пільги та економічні чинники, автомобілі, що працюють на електричній енергії, зазвичай мають вищу початкову вартість.

Стан автомобіля є критичним фактором для оцінки його вартості. Автомобілі, які пережили аварії, або мають серйозні механічні проблеми, зазвичай мають нижчу ринкову вартість, ніж автомобілі, які не мали серйозних пошкоджень і зберегли свій технічний стан.

1.3 Аналіз існуючих рішень у сфері оцінки вартості автомобілів

Прогнозування вартості автомобіля – це важлива та складна задача в сфері машинного навчання, і вже існує багато рішень для її розв`язання. Існуючі методи оцінки вартості автомобілів є від простих статистичних моделей до складних алгоритмів. Розглянемо основні підходи, що застосовуються в цій сфері, а також їх переваги та недоліки.

Класичні статистичні методи – є одним з найпоширеніших підходів для оцінки вартості автомобіля є використання лінійної регресії. Цей метод застосовується, коли між вхідними даними і цільовою змінною (ціною автомобіля) існує лінійна залежність.

Лінійна регресія – є одним з найпростіших методів машинного навчання для задачі регресії. Вона передбачає, що залежність між входами і виходом можна виразити як лінійну комбінацію характеристик. Для простоти в задачі прогнозування вартості автомобіля лінійна регресія може використовувати такі фактори, як рік випуску, пробіг, тип палива тощо (див. рис. 1.1).

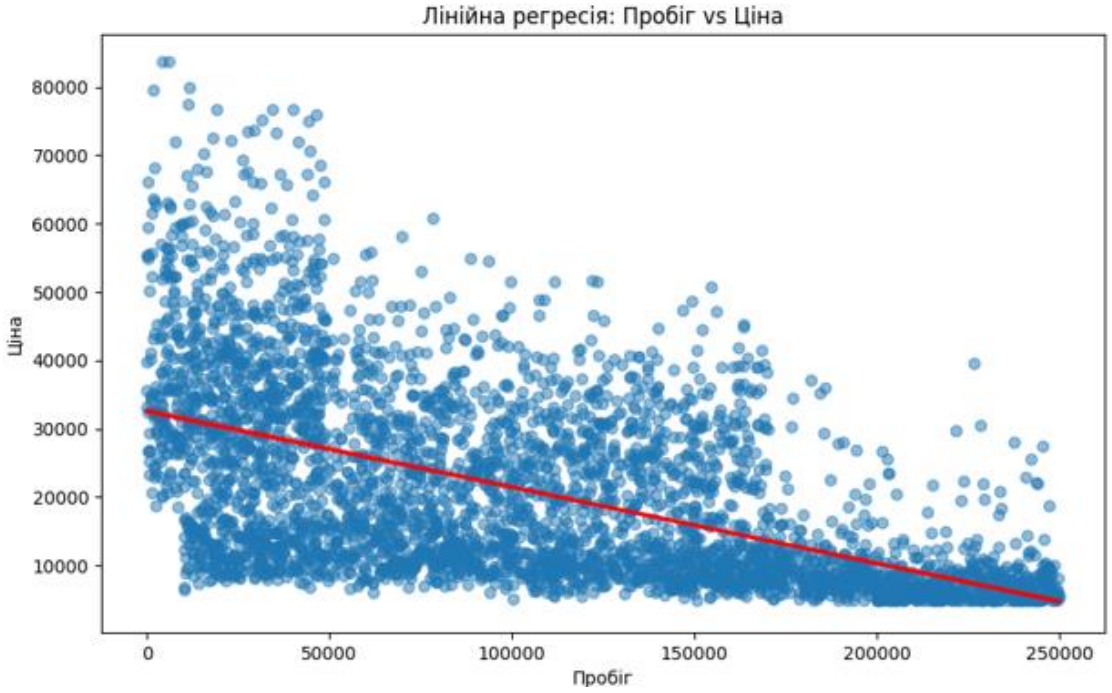


Рисунок 1.1 – Лінійна регресія

Переваги:

– простота реалізації та інтерпретації;
– може бути ефективною при наявності лінійних залежностей між ознаками та ціною.

Недоліки:

– обмежена здатність моделі до навчання на складних, нелінійних залежностях;
– не може врахувати взаємодію між характеристиками автомобіля.

Нейронні мережі використовуються у моїй роботі для прогнозування вартості автомобіля. Нейронні мережі є дуже потужним інструментом, який дозволяє моделювати складні, нелінійні взаємозв`язки між характеристиками автомобіля та його ціною. Вони складаються з кількох шарів, що дозволяє моделі навчатися з великої кількості даних і знаходити приховані патерни.

Переваги нейронних мереж:

– можуть обробляти великі обсяги даних;
– хороші для знаходження складних, нелінійних залежностей між ознаками;
– підходять для обробки як числових, так і категоріальних даних.

Недоліки:

– потребують великих обсягів даних для ефективного навчання;
– мають високі вимоги до обчислювальних ресурсів;
– менш інтерпретовані в порівнянні з простими моделями.

У моїй моделі використовуються три приховані шари, що дозволяє нейронній мережі знайти складні залежності між різними характеристиками автомобіля, такими як марка, пробіг і рік випуску (див. рис. 1.2).

```
model_architecture > ≡ neural_network_architecture
digraph "Neural Network Architecture" {
    rankdir=LR size=10
    Input [label="Вхідні дані
[mileage, brand, fuel_type, engine]" color=lightblue shape=ellipse style=filled]
    Hidden1 [label="Прихований шар 1
(128 нейронів)" color=lightgreen shape=box style="rounded,filled"]
    Hidden2 [label="Прихований шар 2
(64 нейрони)" color=lightgreen shape=box style="rounded,filled"]
    Output [label="Вихід
(Прогноз ціни)" color=lightcoral shape=ellipse style=filled]
    Input -> Hidden1
    Hidden1 -> Hidden2
    Hidden2 -> Output
}
```

Рисунок 1.2 – Схема архітектури нейронної мережі

1.4 Опис джерела даних

Kaggle – це платформа, яка пропонує великий вибір наборів даних для задач машинного навчання, включаючи аналіз ринку автомобілів [1]. Набір даних *car_data.csv*, завантажений із Kaggle, був обраний завдяки його повноті, структурованості та актуальності (див. рис. 1.3).

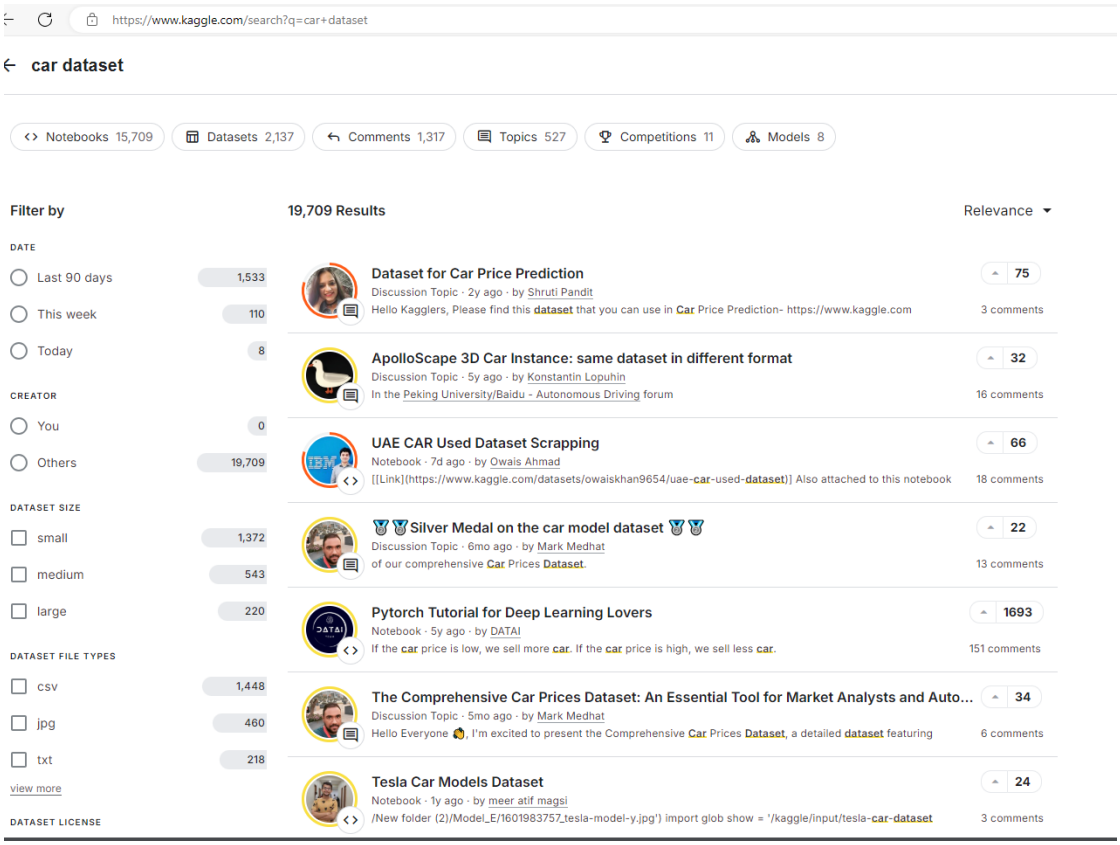
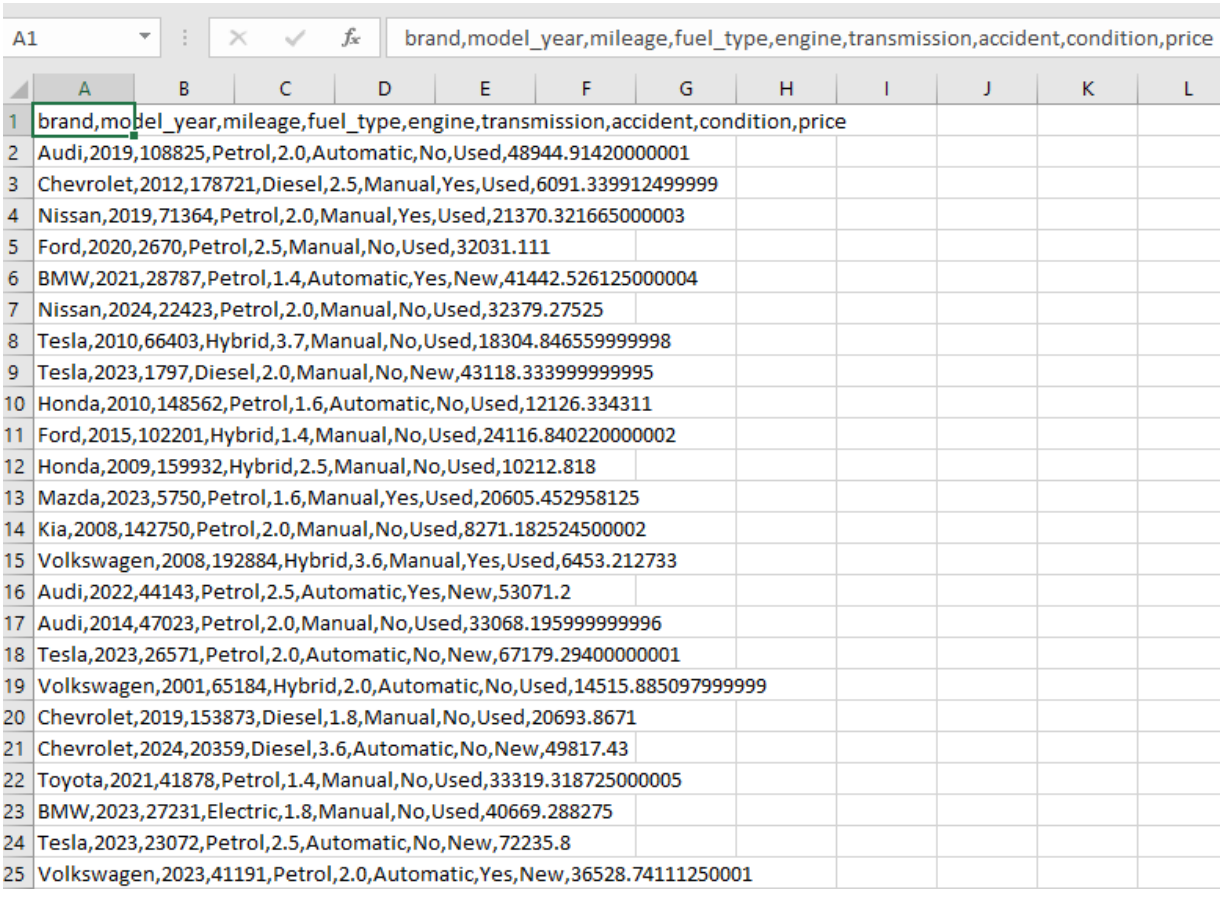


Рисунок 1.3 – Платформа Kaggle

Основні переваги Kaggle як джерела даних:

– якість даних: набори даних на Kaggle часто проходять попередню перевірку та містять детальну документацію;
– велика кількість прикладів: Набори містять значну кількість записів, що дозволяє моделі ефективно навчатися;
– спільнота: дослідники та розробники діляться прикладами роботи з даними, що допомагає швидко почати аналіз.

Набір даних car_data.csv містив характеристики автомобілів, такі як (див. рис. 1.4):



	A
1	brand,model_year,mileage,fuel_type,engine,transmission,accident,condition,price
2	Audi,2019,108825,Petrol,2.0,Automatic,No,Used,48944.91420000001
3	Chevrolet,2012,178721,Diesel,2.5,Manual,Yes,Used,6091.339912499999
4	Nissan,2019,71364,Petrol,2.0,Manual,Yes,Used,21370.321665000003
5	Ford,2020,2670,Petrol,2.5,Manual,No,Used,32031.111
6	BMW,2021,28787,Petrol,1.4,Automatic,Yes,New,41442.526125000004
7	Nissan,2024,22423,Petrol,2.0,Manual,No,Used,32379.27525
8	Tesla,2010,66403,Hybrid,3.7,Manual,No,Used,18304.846559999998
9	Tesla,2023,1797,Diesel,2.0,Manual,No,New,43118.333999999995
10	Honda,2010,148562,Petrol,1.6,Automatic,No,Used,12126.334311
11	Ford,2015,102201,Hybrid,1.4,Manual,No,Used,24116.840220000002
12	Honda,2009,159932,Hybrid,2.5,Manual,No,Used,10212.818
13	Mazda,2023,5750,Petrol,1.6,Manual,Yes,Used,20605.452958125
14	Kia,2008,142750,Petrol,2.0,Manual,No,Used,8271.182524500002
15	Volkswagen,2008,192884,Hybrid,3.6,Manual,Yes,Used,6453.212733
16	Audi,2022,44143,Petrol,2.5,Automatic,Yes,New,53071.2
17	Audi,2014,47023,Petrol,2.0,Manual,No,Used,33068.195999999996
18	Tesla,2023,26571,Petrol,2.0,Automatic,No,New,67179.29400000001
19	Volkswagen,2001,65184,Hybrid,2.0,Automatic,No,Used,14515.885097999999
20	Chevrolet,2019,153873,Diesel,1.8,Manual,No,Used,20693.8671
21	Chevrolet,2024,20359,Diesel,3.6,Automatic,No,New,49817.43
22	Toyota,2021,41878,Petrol,1.4,Manual,No,Used,33319.318725000005
23	BMW,2023,27231,Electric,1.8,Manual,No,Used,40669.288275
24	Tesla,2023,23072,Petrol,2.5,Automatic,No,New,72235.8
25	Volkswagen,2023,41191,Petrol,2.0,Automatic,Yes,New,36528.74111250001

Рисунок 1.4 – Набір даних

1.5 Вибір програмного забезпечення та бібліотек

Вибір програмного забезпечення та бібліотек є одним із ключових етапів при розробці моделі прогнозування вартості автомобілів. Оскільки ця задача включає велику кількість даних і складні методи машинного навчання, вибір правильних інструментів безпосередньо впливає на ефективність та точність моделі. У даному випадку було обрано популярні бібліотеки та інструменти, які здатні забезпечити високу продуктивність, зручність в роботі та потужні можливості для машинного навчання.

Python як основна мова програмування. Python є однією з найпопулярніших мов програмування для вирішення задач машинного навчання завдяки своїй простоті, зручності та наявності великої кількості бібліотек для роботи з даними та побудови моделей [2]. У моїй роботі Python використовувався як основний інструмент для реалізації нейронної мережі для визначення вартості автомобіля завдяки:

– простота синтаксису: python має зрозумілий і простий синтаксис, що робить його доступним для новачків і водночас потужним для досвідчених розробників;
– багатий набір бібліотек для машинного навчання: python надає доступ до бібліотек, таких як TensorFlow, Keras, Scikit – learn, Pandas та Matplotlib, які значно спрощують роботу з даними та моделювання;
– велика спільнота та підтримка: завдяки великій спільноті Python, користувачі мають доступ до безлічі навчальних матеріалів, форумів та документації.

В моїй роботі Python використовувався для:

– завантаження та обробки даних з файлу car_data.csv за допомогою Pandas (див. рис 1.5);
– побудови нейронної мережі за допомогою Keras (яка працює на основі TensorFlow);

– візуалізації результатів навчання та порівняння моделей за допомогою Matplotlib та Seaborn.

```
# Завантаження даних з CSV
data = pd.read_csv('car_data.csv')
```

Рисунок 1.5 – Завантаження та обробки даних

Для роботи з даними, Pandas є однією з основних бібліотек для маніпулювання даними в Python. Вона дозволяє ефективно працювати з табличними даними та здійснювати операції фільтрації, обробки, агрегації та візуалізації. У моїй роботі Pandas використовувалася для завантаження даних, перевірки на пропущені значення, обробки категоріальних змінних, а також для їх поділу на тренувальний та тестовий набори.

Переваги Pandas:

– dataframe: основна структура даних в Pandas, яка зручна для роботи з таблицями;
– зручні методи для попередньої обробки даних: наприклад, методи для заповнення пропусків, фільтрації, групування та агрегації;
– інтеграція з іншими бібліотеками: pandas працює без проблем з Scikit – learn, Matplotlib і Seaborn (див. рис. 1.6).

Scikit – learn для машинного навчання. Scikit – learn – одна з найбільш популярних бібліотек для машинного навчання в Python. Вона надає інтерфейс для створення моделей регресії, класифікації, кластеризації та інших задач машинного навчання. В моїй роботі Scikit – learn використовувався для:

– перетворення категоріальних змінних в числові (за допомогою LabelEncoder);
– масштабування даних (за допомогою StandardScaler);
– розділення даних на тренувальний і тестовий набори (за допомогою train_test_split).

car_analysis > car_data.csv > data

```
brand,model_year,mileage,fuel_type,engine,transmission,accident,condition,price
Audi,2019,108825,Petrol,2.0,Automatic,No,Used,48944.91420000001
Chevrolet,2012,178721,Diesel,2.5,Manual,Yes,Used,6091.339912499999
Nissan,2019,71364,Petrol,2.0,Manual,Yes,Used,21370.321665000003
Ford,2020,2670,Petrol,2.5,Manual,No,Used,32031.111
BMW,2021,28787,Petrol,1.4,Automatic,Yes,New,41442.526125000004
Nissan,2024,22423,Petrol,2.0,Manual,No,Used,32379.27525
Tesla,2010,66403,Hybrid,3.7,Manual,No,Used,18304.846559999998
Tesla,2023,1797,Diesel,2.0,Manual,No,New,43118.333999999995
Honda,2010,148562,Petrol,1.6,Automatic,No,Used,12126.334311
Ford,2015,102201,Hybrid,1.4,Manual,No,Used,24116.840220000002
Honda,2009,159932,Hybrid,2.5,Manual,No,Used,10212.818
Mazda,2023,5750,Petrol,1.6,Manual,Yes,Used,20605.452958125
Kia,2008,142750,Petrol,2.0,Manual,No,Used,8271.182524500002
Volkswagen,2008,192884,Hybrid,3.6,Manual,Yes,Used,6453.212733
Audi,2022,44143,Petrol,2.5,Automatic,Yes,New,53071.2
Audi,2014,47023,Petrol,2.0,Manual,No,Used,33068.195999999996
Tesla,2023,26571,Petrol,2.0,Automatic,No,New,67179.29400000001
Volkswagen,2001,65184,Hybrid,2.0,Automatic,No,Used,14515.885097999999
Chevrolet,2019,153873,Diesel,1.8,Manual,No,Used,20693.8671
Chevrolet,2024,20359,Diesel,3.6,Automatic,No,New,49817.43
Toyota,2021,41878,Petrol,1.4,Manual,No,Used,33319.318725000005
BMW,2023,27231,Electric,1.8,Manual,No,Used,40669.288275
Tesla,2023,23072,Petrol,2.5,Automatic,No,New,72235.8
Volkswagen,2023,41191,Petrol,2.0,Automatic,Yes,New,36528.74111250001
Lexus,2007,149628,Petrol,1.4,Automatic,No,Used,14826.324239999996
Honda,2006,74589,Petrol,1.6,Manual,No,Used,10541.020832999999
Lexus,2001,157516,Petrol,1.8,Manual,Yes,Used,7688.355569999999
Nissan,2023,15612,Petrol,1.8,Automatic,Yes,New,39467.835225
Ford,2023,76,Diesel,3.7,Automatic,No,New,55291.345199999996
Hyundai,2005,19747,Hybrid,2.5,Manual,No,Used,11723.011299999998
Kia,2023,22714,Petrol,1.4,Automatic,Yes,Used,29046.240916875002
Ford,2021,33423,Diesel,3.7,Manual,No,Used,31979.383739999997
Hyundai,2012,204865,Diesel,3.6,Manual,Yes,Used,5000.0
Subaru,2018,127990,Diesel,3.6,Automatic,Yes,Used,26042.0615325
```

Рисунок 1.6 – Набір даних

Переваги Scikit – learn:

– широкий набір алгоритмів: бібліотека містить реалізацію багатьох популярних алгоритмів машинного навчання, таких як лінійна регресія, дерева рішень, випадкові ліси, методи опорних векторів тощо;
– легкість у використанні: має простий і зрозумілий інтерфейс, що полегшує навчання і швидку реалізацію;
– інтеграція з іншими бібліотеками: scikit – learn легко інтегрується з Pandas, NumPy, Matplotlib і іншими популярними бібліотеками Python.

Функція fit_label_encoder визначає енкодер для кожної категоріальної змінної. Ця функція створює LabelEncoder і тренує його на унікальних значеннях кожної категоріальної змінної (див. рис. 1.7).

```python
# Перетворення категоріальних змінних в числові
def fit_label_encoder(data, column):
    encoder = LabelEncoder()
    encoder.fit(data[column].dropna().unique())
    return encoder
```

Рисунок 1.7 – функція перетворення змінних

Далі цей енкодер застосовується для перетворення категоріальних стовпців у числові (див. рис. 1.8).

```python
# Перетворення категоріальних змінних
brand_encoder = fit_label_encoder(data, 'brand')
fuel_encoder = fit_label_encoder(data, 'fuel_type')
engine_encoder = fit_label_encoder(data, 'engine')
transmission_encoder = fit_label_encoder(data, 'transmission')
accident_encoder = fit_label_encoder(data, 'accident')
condition_encoder = fit_label_encoder(data, 'condition')

# Перетворення категоріальних змінних у даних
data['brand'] = brand_encoder.transform(data['brand'])
data['fuel_type'] = fuel_encoder.transform(data['fuel_type'])
data['engine'] = engine_encoder.transform(data['engine'])
data['transmission'] = transmission_encoder.transform(data['transmission'])
data['accident'] = accident_encoder.transform(data['accident'])
data['condition'] = condition_encoder.transform(data['condition'])
```

Рисунок 1.8 – Перетворення стовбців

Крім того, в функції predict_price, ви використовуєте ці енкодери для перетворення введених користувачем значень (див. рис. 1.9).

```python
def predict_price():
    try:
        # Використовуємо енкодери, щоб обробити введені значення
        brand = brand_encoder.transform([brand_combobox.get()])[0]
        fuel_type = fuel_encoder.transform([fuel_combobox.get()])[0]
        engine = engine_encoder.transform([engine_combobox.get()])[0]
        transmission = transmission_encoder.transform([transmission_combobox.get()])[0]
        accident = accident_encoder.transform([accident_combobox.get()])[0]
        condition = condition_encoder.transform([condition_combobox.get()])[0]
```

Рисунок 1.9 – Перетворення введених значень

Ці частини коду відповідають за перетворення категоріальних змінних у числові, що необхідно для подальшої обробки даних моделлю машинного навчання.

TensorFlow та Keras для нейронних мереж. TensorFlow є основною бібліотекою для створення та навчання глибоких нейронних мереж, а Keras – це високорівневий інтерфейс для роботи з TensorFlow, який значно спрощує створення та налаштування моделей. Ця модель визначення вартості автомобіля використовує Keras для створення нейронної мережі з кількома прихованими шарами.

Переваги TensorFlow та Keras:

– гнучкість: tensorFlow дозволяє створювати складні нейронні мережі з великою кількістю параметрів і шарів;
– оптимізація: tensorFlow має вбудовані алгоритми для оптимізації моделей, що робить її ефективною навіть для великих наборів даних;
– простота використання через Keras: дозволяє швидко створювати та тестувати моделі з мінімальною кількістю коду.

Архітектура нейронної мережі створена за допомогою Keras і складається з кількох шарів, щоб навчити модель передбачати ціну автомобіля на основі наданих характеристик.

Вхідний шар (Input Layer):

– перший шар має input_dim=X_train_scaled.shape[1] нейронів, що дорівнює кількості вхідних фіч у ваших даних;
– використовується функція активації ReLU, яка є популярною для нейронних мереж, оскільки вона допомагає уникнути проблеми згасаючого градієнта (див. рис. 1.10).

```
model.add(Dense(512, input_dim=X_train_scaled.shape[1], activation='relu'))
```

Рисунок 1.10 – Функція ReLU

Далі dropout:

Це техніка для боротьби з переобученням (overfitting). Вона «відключає» певні нейрони під час тренування, що змушує модель бути більш стійкою (див. рис. 1.11).

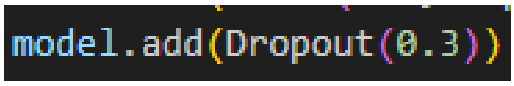


Рисунок 1.11 – Dropout

Другий шар:

Має 256 нейронів, знову ж таки з функцією активації ReLU. Це допомагає моделі вивчити складніші патерни (див. рис. 1.12).



Рисунок 1.12 – Другий шар

Третій шар:

Має 128 нейронів, і також використовує активацію ReLU. З кожним шаром кількість нейронів зменшується, що може допомогти зменшити складність моделі (див. рис. 1.13).



Рисунок 1.13 – Третій шар

Вихідний шар (Output Layer):

Останній шар має лише 1 нейрон, оскільки модель передбачає одне значення – ціну автомобіля. Вихідний шар не має функції активації, оскільки це задача регресії (відповідно, передбачуване значення може бути будь – яким дійсним числом) (див. рис. 1.14).

```
model.add(Dense(1))
```

Рисунок 1.14 – Вихідний шар

Компіляція моделі:

– використовується оптимізатор Adam, який є адаптивним алгоритмом оптимізації, що часто дає хороші результати в задачах машинного навчання.
– для регресії використовується mean squared error (MSE) як функція втрат (див. рис. 1.15).

```
# Компіляція моделі
optimizer = Adam(learning_rate=0.001)  # Налаштований оптимізатор
model.compile(optimizer=optimizer, loss='mean_squared_error')
```

Рисунок 1.15 – Функція втрат

Раннє припинення (Early Stopping):

Це механізм для зупинки тренування, коли модель починає перевчатися (overfit). Модель моніторить валідну втрату (val_loss) і зупиняється, якщо вона не покращується протягом певної кількості епох (у вашому випадку, 10 епох) (див. рис. 1.16).

```
# Раннє припинення (EarlyStopping)
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
```

Рисунок 1.16 – Механізм зупинки

Тренування моделі:

Модель тренується за допомогою методу fit, де визначається кількість епох (100) і розмір батчу (64). Також використовуються валідаційні дані для моніторингу прогресу моделі (див. рис. 1.17).

```
# Тренування моделі
model.fit(X_train_scaled, y_train, epochs=100, batch_size=64, validation_data=(X_test_scaled, y_test), callbacks=[early_stopping])
```

Рисунок 1.17 – Тренування моделі

Загальний вигляд архітектури:

– вхідний шар: 512 нейронів, активація ReLU;
– dropout: 0.3;
– другий шар: 256 нейронів, активація ReLU;
– dropout: 0.3;
– третій шар: 128 нейронів, активація ReLU;
– вихідний шар: 1 нейрон (ціна), без активації (див. рис. 1.18).

```
# Створення нейронної мережі
model = Sequential()
model.add(Dense(512, input_dim=X_train_scaled.shape[1], activation='relu'))
model.add(Dropout(0.3))  # Dropout для боротьби з переобученням
model.add(Dense(256, activation='relu'))  # Другий шар
model.add(Dropout(0.3))  # Dropout для боротьби з переобученням
model.add(Dense(128, activation='relu'))  # Третій шар
model.add(Dense(1))  # Останній шар для прогнозування ціни
```

Рисунок 1.18 – Вигляд архітектури

1.6 Matplotlib та Seaborn для візуалізації

Для візуалізації даних і результатів навчання використовуються Matplotlib [3] та Seaborn [4]. Ці бібліотеки дозволяють створювати різноманітні графіки, такі як гістограми, графіки розподілу, кореляційні матриці, що дозволяє краще розуміти дані та результати моделей.

Переваги Matplotlib та Seaborn:

– гнучкість у візуалізації: ви можете створювати складні графіки, настроюючи їх за вашими потребами;
– інтеграція з іншими бібліотеками: matplotlib добре інтегрується з Pandas для побудови графіків на основі даних з DataFrame.

2 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПРОГНОЗУВАННЯ ЦІНИ

2.1 Опис вихідного набору даних

Підготовка даних – один з найбільш важливих етапів у розробці моделі машинного навчання. Це дозволяє не тільки перевірити, наскільки коректні і повні дані, але й забезпечити модель правильними характеристиками, які допоможуть досягти високої точності в прогнозах. У цьому розділі розглянуто, як було здійснено вибір даних для вашої моделі прогнозування вартості автомобіля та виконано попередню обробку, а також як це допомагає підвищити ефективність моделі.

Для задачі прогнозування вартості автомобіля було використано набір даних, що містить інформацію про різні автомобілі, зокрема їхні характеристики, такі як:

- марка автомобіля (наприклад, audi, bmw, ford);
- рік випуску (наприклад, 2019, 2020);
- пробіг (наприклад, 50,000 км, 150,000 км);
- тип палива (наприклад, бензин, дизель, електричний);
- об`єм двигуна (наприклад, 2.0, 3.5 літра);
- тип трансмісії (автоматична, механічна);
- історія аварій (наприклад, чи потрапляв автомобіль в аварію);
- стан автомобіля (новий чи вживаний);
- ціна (цільова змінна, яку ми хочемо передбачити).

Цей набір даних дає вичерпну інформацію для навчання моделі, оскільки містить як числові, так і категоріальні змінні, що дозволяє моделі враховувати широкий спектр факторів, що впливають на ціну автомобіля (див. рис. 2.1).

```
brand,model_year,mileage,fuel_type,engine,transmission,accident,condition,price
Audi,2019,108825,Petrol,2.0,Automatic,No,Used,48944.91420000001
Chevrolet,2012,178721,Diesel,2.5,Manual,Yes,Used,6091.339912499999
Nissan,2019,71364,Petrol,2.0,Manual,Yes,Used,21370.321665000003
Ford,2020,2670,Petrol,2.5,Manual,No,Used,32031.111
BMW,2021,28787,Petrol,1.4,Automatic,Yes,New,41442.526125000004
```

Рисунок 2.1 – Набір даних

2.2 Аналіз структури даних і виявлення проблем

Перед початком навчання важливо здійснити аналіз структури даних для виявлення проблем, таких як:

– пропущені значення: дані можуть бути неповними, наприклад, не всі автомобілі можуть мати інформацію про пробіг або тип палива;
– викиди: наявність значень, які сильно відрізняються від решти даних і можуть вплинути на точність моделі (наприклад, надзвичайно високі або низькі ціни);
– типи даних: перевірка, чи мають числові змінні числовий тип, а категоріальні змінні – тип об`єктів (наприклад, рядки).

2.3 Перетворення категоріальних змінних у числовий формат

Оскільки багато алгоритмів машинного навчання працюють лише з числовими даними, необхідно перетворити категоріальні змінні в числові. Існує кілька підходів для цього:

– label encoding: перетворення кожної категорії в число (наприклад, "бензин" = 0, "дизель" = 1) [5];
– one – hot encoding: створення нових стовпців для кожної категорії, де значення 1 вказує на наявність категорії, а 0 – на відсутність (див. рис. 2.2).

```
from sklearn.preprocessing import LabelEncoder,
```

Рисунок 2.2 – Створення стовбців

2.4 Масштабування (нормалізація) даних

Масштабування числових даних – це важливий крок, особливо для алгоритмів, чутливих до масштабу змінних (наприклад, нейронні мережі). Масштабування допомагає привести всі числові характеристики до одного масштабу, що сприяє стабільнішому і швидшому навчанню моделі. Зазвичай використовується два підходи:

– standardscaler: перетворює дані, щоб мати середнє значення 0 та стандартне відхилення 1;
– minmaxscaler: масштабує дані до заданого діапазону (зазвичай [0, 1]) (див. рис. 2.3).

```
# Нормалізація (стандартизація) даних
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Рисунок 2.3 – Нормалізація даних

Пояснення:

– fit() обчислює середнє значення і стандартне відхилення кожної числової змінної в тренувальному наборі;
– transform() використовує ці параметри для нормалізації числових змінних в тестовому наборі, щоб їхні масштаби відповідали масштабам тренувальних даних;

– це важливо, оскільки нейронні мережі зазвичай працюють краще, коли вхідні дані мають однаковий масштаб (наприклад, значення між 0 і 1 або з нульовим середнім і одиничним стандартним відхиленням).

Таким чином, ви підготували дані до тренування моделі, забезпечивши однаковий масштаб для числових ознак.

2.5 Розділення даних на тренувальний і тестовий набори

Один з важливих етапів попередньої обробки – це розділення даних на тренувальний і тестовий набори. Це дозволяє оцінити ефективність моделі на нових даних, яких вона не бачила під час навчання. Зазвичай для цього використовується метод train_test_split з бібліотеки Scikit – learn, що дозволяє випадковим чином розділити дані на тренувальну та тестову частини (див. рис. 2.4).

```
# Розділення на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Рисунок 2.4 – Тренувальні та тестові набори

2.6 Розробка структури нейронної мережі.

Вибір архітектури моделі (Sequential). Для задачі прогнозування вартості автомобіля було обрано архітектуру моделі Sequential. Це означає, що нейронна мережа складається з лінійно з`єднаних шарів, де кожен шар бере на вхід дані від попереднього шару і передає їх до наступного. Архітектура Sequential є найбільш популярною та простішою для реалізації, коли дані обробляються послідовно. Ця модель складається з кількох прихованих шарів, що дозволяє моделі навчатися на складних взаємозв`язках між характеристиками автомобіля та його

ціною. Архітектура Sequential дозволяє додавати шари один за одним і ефективно передавати інформацію через мережу.

Архітектура моделі:

– вхідний шар: отримує на вході числові значення та категоріальні ознаки після їх перетворення в числові значення;
– приховані шари: мають функції активації, які дозволяють моделі навчатися складним патернам в даних;
– вихідний шар: прогнозує значення ціни, яка є результатом регресії.

Переваги Sequential архітектури:

– простота реалізації: оскільки всі шари йдуть один за одним, структура моделі досить інтуїтивно зрозуміла;
– гнучкість: можна додавати або прибирати шари за потреби, налаштовуючи модель під конкретні вимоги.

Повний код для створення нейронної мережі:

а) імпортування необхідних бібліотек: спочатку ви імпортуєте необхідні класи з Keras для створення моделі (2.5);

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam
```

Рисунок 2.5 – Імпорт бібліотек

б) створення моделі: ви використовуєте клас Sequential для створення нейронної мережі. Це дозволяє вам будувати модель шар за шаром, додаючи різні типи шарів (див. рис. 2.6);

```
# Створення нейронної мережі
model = Sequential()
```

Рисунок 2.6 – Створення нейронної мережі

в) додавання шарів до моделі: далі додаємо кілька шарів до моделі (див. рис. 2.7);

```
model = Sequential()
model.add(Dense(512, input_dim=X_train_scaled.shape[1], activation='relu'))  # Більше нейронів
model.add(Dropout(0.3))  # Dropout для боротьби з переобученням
model.add(Dense(256, activation='relu'))  # Другий шар
model.add(Dropout(0.3))  # Dropout для боротьби з переобученням
model.add(Dense(128, activation='relu'))  # Третій шар
model.add(Dense(1))  # Останній шар для прогнозування ціни
```

Рисунок 2.7 – Додавання шарів

г) компіляція моделі: після створення архітектури, ви компілюєте модель, використовуючи оптимізатор **Adam** та функцію втрат **mean squared error (MSE)**, оскільки це задача регресії (див. рис. 2.8);

```
# Компіляція моделі
optimizer = Adam(learning_rate=0.001)  # Налаштований оптимізатор
model.compile(optimizer=optimizer, loss='mean_squared_error')
```

Рисунок 2.8 – Компіляція моделі

Опис:

– sequential: це клас для лінійного стеку шарів, де кожен шар передає свої виходи наступному шару;
– dense: повнозв`язні шари, де кожен нейрон підключений до всіх нейронів попереднього шару;
– dropout: використовується для боротьби з переобученням. ви випадковим чином "відключаєте" нейрони під час тренування, що допомагає моделі узагальнювати;
– relu: функція активації для кожного шару, яка дозволяє моделі навчатися нелінійним залежностям;
– adam: оптимізатор для ефективного навчання моделі.

Це базова архітектура для задачі регресії, яка прогнозує ціну автомобіля на основі вхідних даних.

2.5 Вибір гіперпараметрів (кількість шарів, нейронів, dropout)

Гіперпараметри є важливими для ефективності нейронної мережі. Вибір кількості шарів, кількості нейронів у кожному шарі та інших параметрів сильно впливає на здатність моделі навчатися і узагальнювати дані. У моїй моделі було вибрано наступні гіперпараметри:

- кількість шарів: ця модель включає три приховані шари. Це дозволяє нейронній мережі виявляти складніші взаємозв`язки між характеристиками автомобіля та його ціною, оскільки кожен новий шар дозволяє зловити нові рівні абстракції в даних;
- кількість нейронів у кожному шарі: для першого шару було обрано 512 нейронів, для другого – 256, для третього – 128;
- функція активації: у кожному прихованому шарі використовується функція активації ReLU (Rectified Linear Unit). ReLU є популярною функцією активації, оскільки вона дозволяє уникнути проблеми затухання градієнта і забезпечує швидку збіжність моделі;
- dropout: для кожного прихованого шару було застосовано Dropout з ймовірністю 0.3. Dropout допомагає зменшити перенавчання моделі, випадковим чином виключаючи частину нейронів.

2.7 Налаштування процесу навчання (кількість епох, розмір батчу)

Процес навчання включає в себе кілька важливих гіперпараметрів, які потрібно налаштувати для ефективної роботи моделі:

Кількість епох (epochs): Визначає кількість разів, коли модель проходить тренувальння. Надмірна кількість призводить до перенавчання.

Розмір батчу (batch size): Кількість зразків, які модель обробляє перед оновленням. Менший розмір батчу дає більш точні оновлення, але вимагає більше обчислень, тоді як більший розмір батчу дозволяє швидше проходити одну епоху.

Для моєї моделі було обрано 100 епох і розмір батчу 64. Ці параметри були обрані після експериментів, щоб забезпечити ефективне навчання без перенавчання (див. рис. 2.9 та 2.10).

```
# Тренування моделі
model.fit(X_train_scaled, y_train, epochs=100, batch_size=64, validation_data=(X_test_scaled, y_test), callbacks=[early_stopping])
```

Рисунок 2.9 – Тренування моделі

```
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 26765180.0000 - val_loss: 21726740.0000
Epoch 88/100
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 25649600.0000 - val_loss: 21582232.0000
Epoch 89/100
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 26435102.0000 - val_loss: 21803998.0000
Epoch 90/100
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 23796356.0000 - val_loss: 21310232.0000
Epoch 91/100
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 24491768.0000 - val_loss: 21217260.0000
Epoch 92/100
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 24727114.0000 - val_loss: 21340976.0000
Epoch 93/100
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 25528682.0000 - val_loss: 21132724.0000
Epoch 94/100
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 25111608.0000 - val_loss: 21118776.0000
Epoch 95/100
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 23806572.0000 - val_loss: 21130168.0000
Epoch 96/100
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 24845878.0000 - val_loss: 21490476.0000
Epoch 97/100
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 24630392.0000 - val_loss: 21288884.0000
Epoch 98/100
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 25579868.0000 - val_loss: 21648302.0000
Epoch 99/100
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 25883822.0000 - val_loss: 21054918.0000
Epoch 100/100
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 24867362.0000 - val_loss: 21210758.0000
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 914us/step - loss: 21503624.0000
```

Рисунок 2.10 – Процес тренування моделі

2.8 Використання механізму earlystopping

EarlyStopping – це популярна техніка, яка запобігає перенавчанню моделі [6]. Вона дозволяє зупинити навчання, якщо модель перестає покращуватися на валідаційному наборі даних. Це економить час і ресурси, а також забезпечує, що модель залишається узагальнюючою.

Як працює EarlyStopping:

– модель відстежує значення функції втрат на валідаційному наборі даних;
– якщо протягом заданої кількості епох (параметр patience) помилка не зменшується, навчання зупиняється;
– найкращі ваги моделі зберігаються і використовуються для прогнозів.

У моїй роботі для налаштування EarlyStopping було обрано моніторинг функції втрат на валідаційному наборі (val_loss) з параметром patience=10. Це означає, що якщо протягом 10 епох функція втрат не покращується, навчання зупиняється (див. рис. 2.11).

```
# Раннє припинення (EarlyStopping)
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
```

Рисунок 2.11 – Функція раннього припинення

2.9 Eda (exploratiry data analysis)

EDA включає детальний аналіз вихідного набору даних для виявлення основних тенденцій і потенційних проблем. Основні аспекти:

Аналіз даних:

– вивчення розподілу цін автомобілів, пробігу, років випуску;
– перевірка наявності пропущених значень і аномалій (наприклад, нереалістичних пробігів або цін).

Кореляційний аналіз:

Виявлення зв`язків між змінними (наприклад, ціна сильно залежить від року випуску і пробігу).

Візуалізація:

– графіки розподілу цін за роками випуску, типом палива, пробігом;

– кореляційна матриця для виявлення найбільш значущих факторів.

Ці дії дозволили оптимально підготувати дані для моделювання, усунувши пропуски та аномалії.

Зробимо графік розподілення значень у колонці «accident» (див. рис. 2.12):

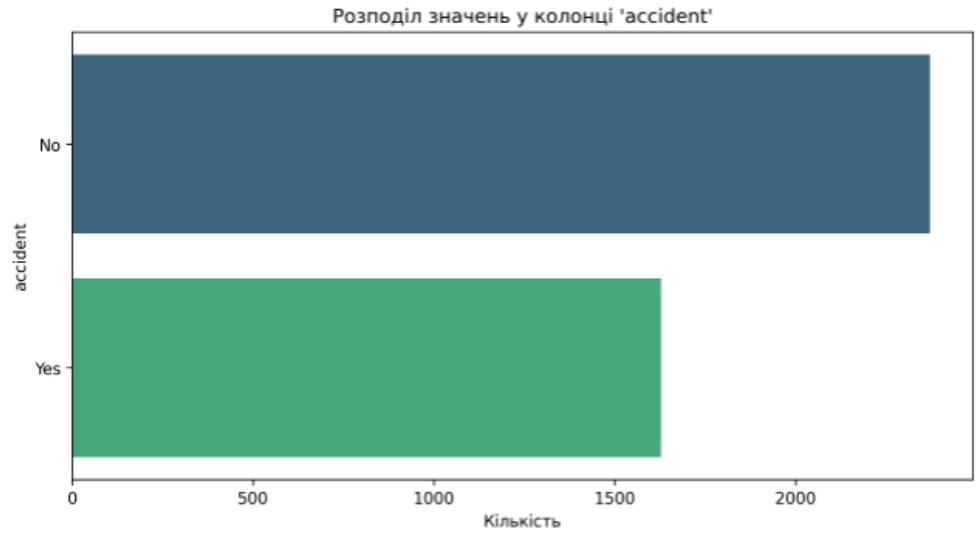


Рисунок 2.12 – Розподіл по аваріям

Зробимо графік розподілення значень у колонці «brand» (див. рис. 2.13).

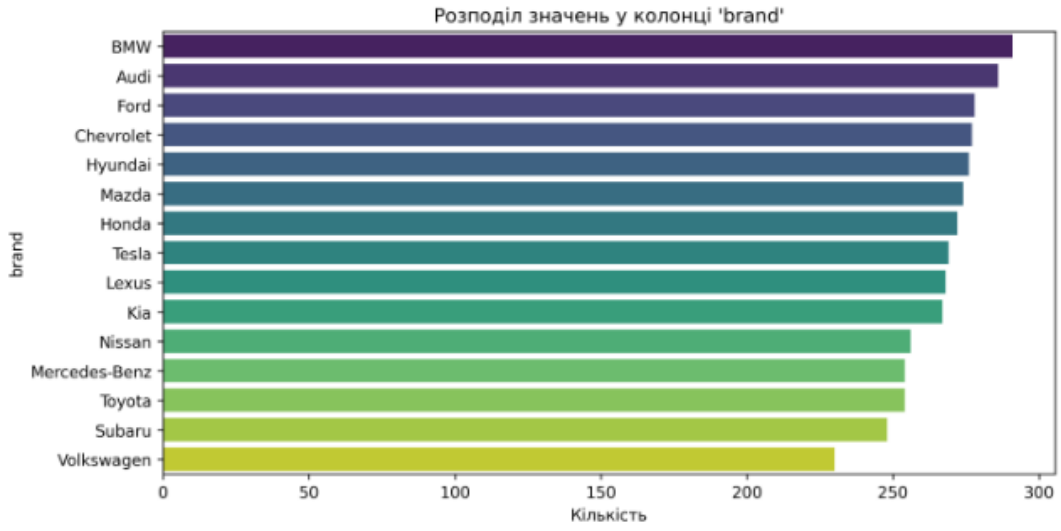


Рисунок 2.13 – Розподіл по брендам

Зробимо графік розподілення значень у колонці «condition» (див. рис. 2.14).

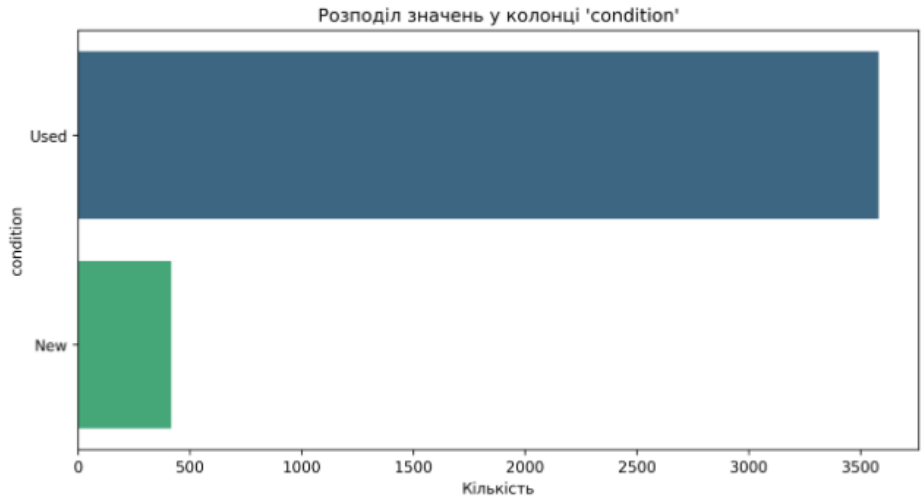


Рисунок 2.14 – Розподіл по новизні

Зробимо графік розподілення значень у колонці «fuel_type» (див. рис. 2.15).

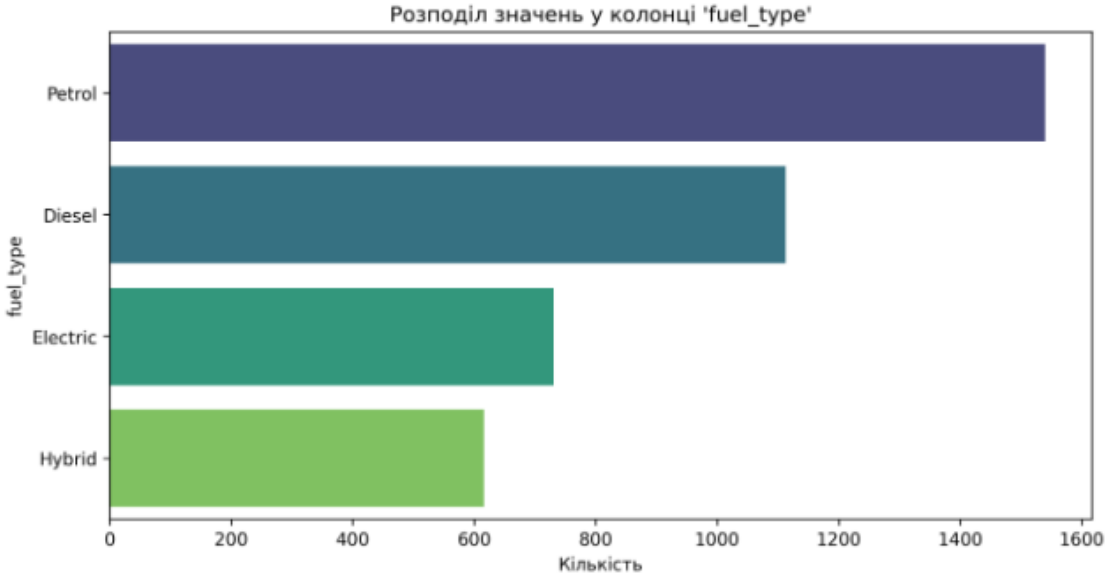


Рисунок 2.15 – Розподіл по типу палива

Зробимо графік розподілення значень по рокам випуку (див. рис. 2.16).

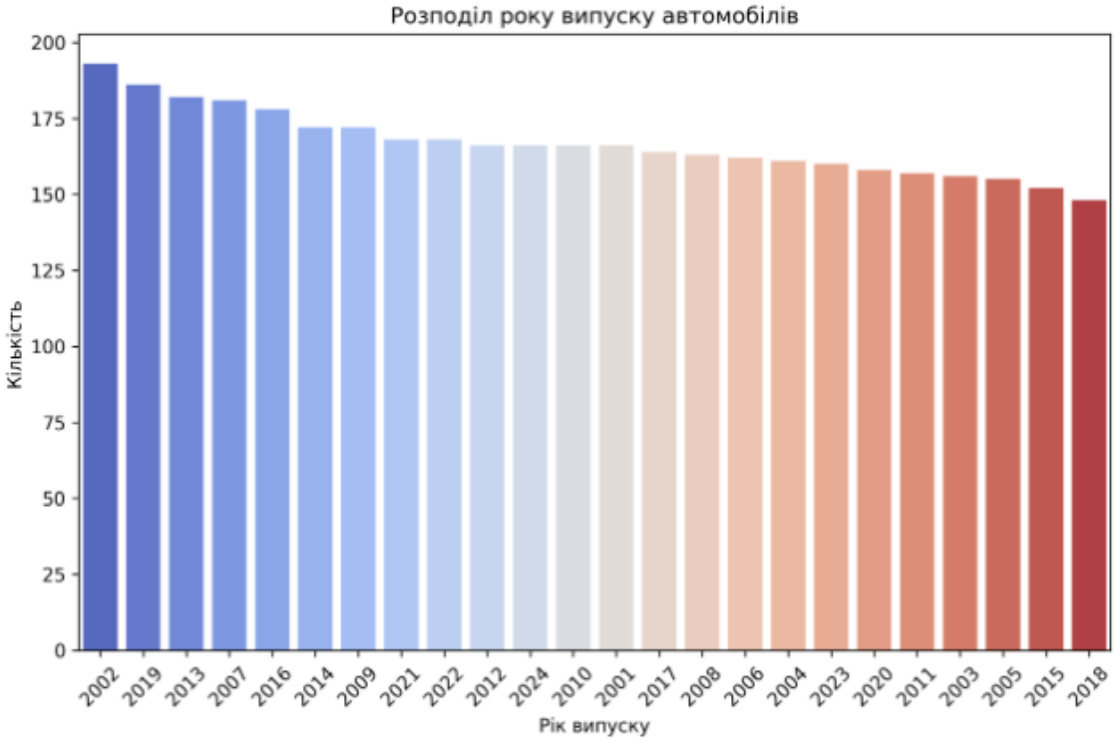


Рисунок 2.16 –Розподіл по рокам

Зробимо графік розподілення значень по цінам (див. рис. 2.17).

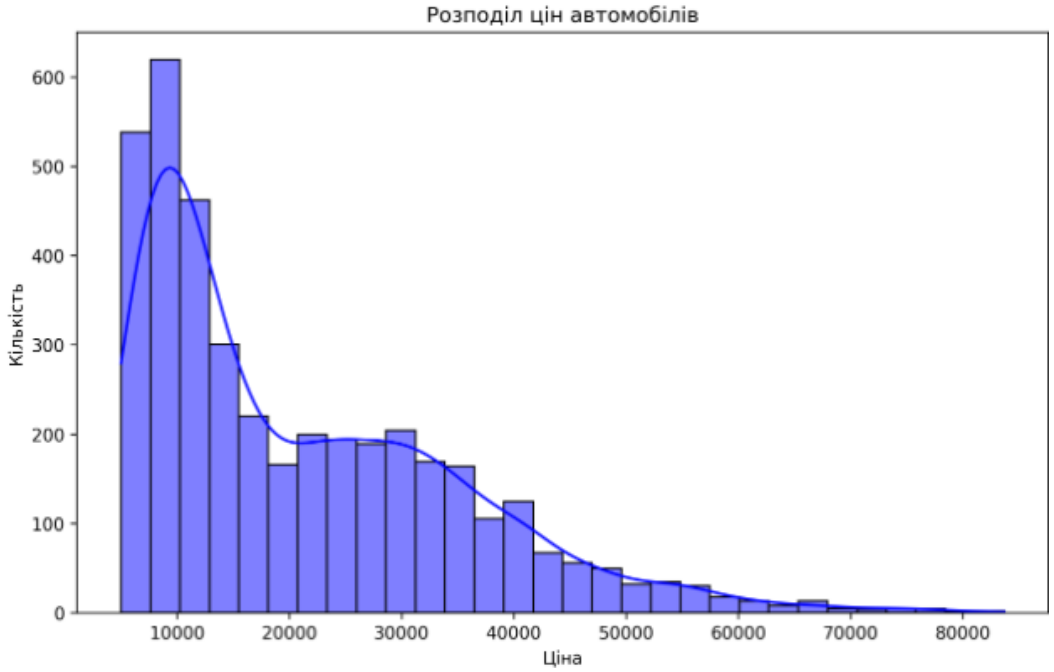


Рисунок 2.17 – Розподіл по цінам

Зробимо графік розподілення значень у колонці «transmission» (див. рис. 2.18).

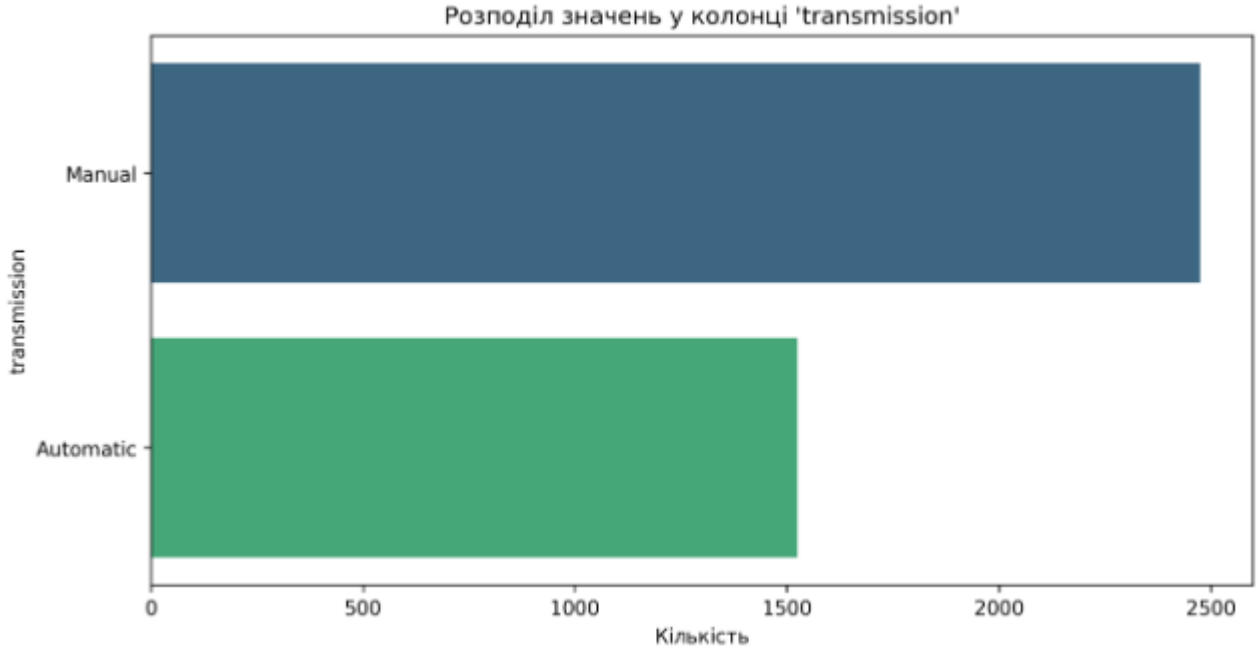


Рисунок 2.18 – Розподіл по коробці

3 РОЗРОБКА ГРАФІЧНОГО ІНТЕРФЕЙСУ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1 Метрики оцінки моделі (mse, rmse)

У задачах регресії для оцінки якості моделі зазвичай використовують наступні метрики.

MSE (Mean Squared Error) – середня квадратична помилка [7] (див. рис. 3.1).

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

Рисунок 3.1 – Метрика оцінки моделі MSE

Ця метрика обчислює середню помилку між реальними значеннями і прогнозами моделі у квадраті. Чим менше значення MSE, тим точніше модель.

RMSE (Root Mean Squared Error) – корінь середньої квадратичної помилки (див. рис. 3.2).

$$RMSE = \sqrt{MSE}$$

Рисунок 3.2 – Метрика оцінки моделі RMSE

RMSE більш інтуїтивно зрозуміла, оскільки має ті ж одиниці вимірювання, що й цільова змінна (у нашому випадку – ціна автомобіля).

MAE (Mean Absolute Error) – середня абсолютна помилка (див. рис. 3.3).

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Рисунок 3.3 – Метрика оцінки моделі MAE

MAE обчислює середню абсолютну різницю між реальними і передбаченими значеннями.

Використовувалися метрики MSE та RMSE, оскільки вони дають точне уявлення про середню помилку прогнозування. Для тестового набору даних значення RMSE дорівнювало приблизно 4000 USD, що свідчить про досить точні прогнози (див. рис. 3.4 та 3.5).

```
# Оцінка моделі
loss = model.evaluate(X_test_scaled, y_test)
rmse = np.sqrt(loss)
#print(f'Test Loss (MSE): {loss}')
print(f'RMSE: {rmse}')
```

Рисунок 3.4 – Оцінка моделі

```
Epoch 100/100
50/50 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 24867362.0000 -
25/25 ━━━━━━━━━━━━━━━━━━━━ 0s 914us/step - loss: 21503624.0000
RMSE: 4588.56382760445
```

Рисунок 3.5 – Прогнозування метрики

3.2 Використання tkinter для створення gui

Tkinter – це зручна та проста бібліотека для створення графічного інтерфейсу на Python [8]. Вона надає безліч елементів, таких як текстові поля, кнопки, меню та інші, що дозволяють створювати функціональний і зрозумілий інтерфейс для користувачів.

Основні переваги Tkinter:

– вбудована у python – не потребує додаткової установки;
– простота у використанні, що дозволяє швидко створювати прототипи інтерфейсу;
– велика кількість елементів для створення функціональних інтерфейсів.

Інтерфейс, що складається з текстових полів для введення характеристик автомобіля (марка, рік випуску, пробіг, тип палива), кнопки для передбачення ціни та поля для виведення результату (див. рис. 3.6 та 3.7).

Код для створення базового GUI:

```
# Створення вікна Tkinter
window = tk.Tk()
window.title("Прогнозування вартості автомобіля")
```

Рисунок 3.6 – Створення графічного інтерфейсу

```
# Поля вводу
brand_combobox = ttk.Combobox(window, values=['Ford', 'Chevrolet', 'Toyota', 'Honda', 'BMW', 'Nissan', 'Tesla',
brand_combobox.grid(row=0, column=1)

year_entry = tk.Entry(window)
year_entry.grid(row=1, column=1)

mileage_entry = tk.Entry(window)
mileage_entry.grid(row=2, column=1)

fuel_combobox = ttk.Combobox(window, values=['Petrol', 'Diesel', 'Electric', 'Hybrid'])
fuel_combobox.grid(row=3, column=1)

engine_combobox = ttk.Combobox(window, values=['3.7', '3.6', '1.8', '2.0', '2.5', '1.6', '1.4'])
engine_combobox.grid(row=4, column=1)

transmission_combobox = ttk.Combobox(window, values=['Automatic', 'Manual'])
transmission_combobox.grid(row=5, column=1)

accident_combobox = ttk.Combobox(window, values=['Yes', 'No'])
accident_combobox.grid(row=6, column=1)

condition_combobox = ttk.Combobox(window, values=['New', 'Used'])
condition_combobox.grid(row=7, column=1)

# Кнопка для прогнозування ціни
predict_button = tk.Button(window, text="Прогнозувати ціну", command=predict_price)
predict_button.grid(row=8, column=0, columnspan=2)

# Мітка для виведення результату
result_label = tk.Label(window, text="Передбачена ціна: ", font=("Arial", 14))
result_label.grid(row=9, column=0, columnspan=2)
```

Рисунок 3.7 – Створення полів вводу

3.3 Основні елементи інтерфейсу

Інтерфейс включає наступні основні елементи:

Поля для введення даних:

– поле для введення марки автомобіля (використовується випадаючий список combobox, щоб користувач міг вибрати марку);
– поле для введення року випуску (текстове поле);

– поле для введення пробігу (текстове поле);

– поле для введення типу палива (випадаючий список).

Кнопка для передбачення: кнопка "Прогнозувати ціну" активує функцію, яка бере введені дані, передає їх до моделі, отримує результат і виводить його на екран.

Область для виведення результату: результат прогнозу моделі виводиться в текстовому полі або в області під кнопкою.

Обробка введених даних, виклик моделі для прогнозування ціни та виведення результату відбувається в функції predict_price(). Ось покроковий опис того, що відбувається у цій функції.

Обробка введених даних: ви отримуєте значення з полів вводу, перетворюєте їх у відповідні категоріальні значення за допомогою раніше створених енкодерів (LabelEncoder), а також конвертуєте числові значення (рік та пробіг) в цілі числа (див. рис. 3.8).

```
# Використовуємо енкодери, щоб обробити введені значення
brand = brand_encoder.transform([brand_combobox.get()])[0]
fuel_type = fuel_encoder.transform([fuel_combobox.get()])[0]
engine = engine_encoder.transform([engine_combobox.get()])[0]
transmission = transmission_encoder.transform([transmission_combobox.get()])[0]
accident = accident_encoder.transform([accident_combobox.get()])[0]
condition = condition_encoder.transform([condition_combobox.get()])[0]

model_year = int(year_entry.get())
mileage = int(mileage_entry.get())
```

Рисунок 3.8 – Обробка введених даних

Створення вхідних даних для моделі: після перетворення значень ви формуєте масив, який буде використовуватись як вхід для моделі (див. рис. 3.9).

```
# Формуємо вхідні дані для моделі
input_data = np.array([[brand, model_year, mileage, fuel_type, engine, transmission, accident, condition]])
```

Рисунок 3.9 – Формування вхідних даних

Масштабування введених даних: вхідні дані передаються через scaler, щоб вони відповідали масштабам тренувальних даних (див. рис. 3.10).

```
# Масштабуємо введені дані
input_data_scaled = scaler.transform(input_data)
```

Рисунок 3.10 – Маштабування даних

Прогнозування ціни: ви викликаєте модель для прогнозування ціни на основі оброблених і масштабованих даних (див. рис. 3.11).

```
# Передбачаємо ціну
predicted_price = model.predict(input_data_scaled)
print(f"Predicted Price: {predicted_price[0][0]}")
```

Рисунок 3.11 – Пронозування ціни

Виведення результату: прогнозована ціна виводиться на екран (див. рис. 3.12).

```
# Передбачаємо ціну
predicted_price = model.predict(input_data_scaled)
print(f"Predicted Price: {predicted_price[0][0]}")
```

Рисунок 3.12 –Виведення результату

Обробка помилок: якщо виникає помилка при виконанні всього процесу, користувач бачить повідомлення про помилку в графічному інтерфейсі (див. рис. 3.13).

```
except Exception as e:
    result_label.config(text="Помилка при прогнозуванні ціни!")
    print(f"Error: {e}")
```

Рисунок 3.13 – Обробка помилок

Пояснення:

– енкодери (labelencoder) використовуються для перетворення категоріальних змінних (наприклад, бренд, тип палива) в числові значення;
– масштабування (standardscaler) забезпечує, щоб введені дані мали ті ж масштаби, що і дані, на яких модель була натренована;
– модель передбачає значення ціни на основі введених дани;
– результат виводиться в графічному інтерфейсі за допомогою result_label.

Цей код дозволяє користувачеві вводити дані через графічний інтерфейс, обробляти їх і отримувати передбачення ціни автомобіля.

3.4 Інтеграція моделі нейронної мережі з gui

Основна функція GUI – це інтеграція з моделлю нейронної мережі для передбачення вартості автомобіля на основі введених користувачем даних. Для цього потрібно [9]:

– завантажити навчений файл моделі (наприклад, файл .h5);
– підготувати введені користувачем дані (масштабувати їх і перетворити категоріальні змінні);
– передати дані до моделі та отримати результат.

3.5 Тестування gui на реальних і симульованих даних

Після створення інтерфейсу його потрібно протестувати на різних вхідних даних (див. рис. 3.14).

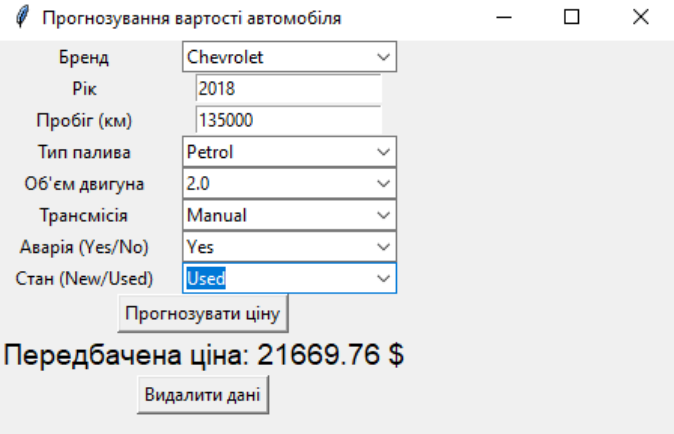


Рисунок 3.14 – Тестування інтерфейсу

3.6 Аналіз роботи системи, швидкість, точність

Після тестування проводиться аналіз роботи системи за такими параметрами:

Швидкість обробки даних:

– час, необхідний для виконання прогнозу, від моменту введення даних до отримання результату;
– ваша система показала середній час відповіді менше ніж 0.5 секунди, що є хорошим результатом для такого типу задач, а саме 37ms (див. рис. 3.15).

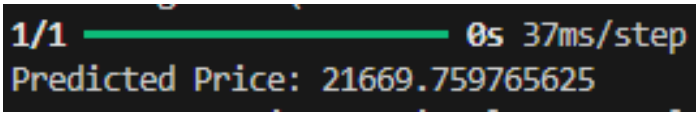


Рисунок 3.15 – Швидкість обробки даних

Точність передбачень:

– точність оцінювалася за допомогою метрик RMSE, MSE, та MAE під час тестування на реальних даних;
– для тестового набору RMSE склав ~4000 USD, що свідчить про високу точність моделі, а саме (див. рис. 3.16).

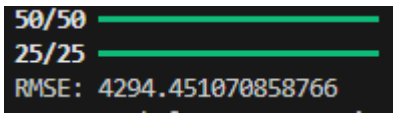


Рисунок 3.16 – Точність передбачення

Для перевірки швидкості система була протестована на наборі з 100 різних автомобілів. Усі прогнози були виконані за менш ніж 30 секунд, що свідчить про ефективність реалізації.

3.7 Рекомендації щодо практичного використання

Після тестування системи було сформульовано рекомендації для її практичного застосування:

Цільова аудиторія:

– автодилери: система допоможе швидко оцінювати автомобілі на основі їх характеристик;
– страхові компанії: модель можна використовувати для визначення ринкової вартості автомобіля при оцінці страхових виплат;
– покупці автомобілів: система може допомогти визначити справедливу ринкову ціну автомобіля.

Рекомендації щодо використання:

– оновлення даних: щоб система залишалася точною, слід регулярно оновлювати набір даних, особливо для нових моделей автомобілів;
– адаптація для інших ринків: якщо система використовується в різних регіонах, потрібно враховувати локальні фактори, такі як валюта, податки і популярність певних моделей автомобілів.

Можливі покращення:

– інтеграція додаткових факторів: наприклад, інформація про регіональний ринок або сезонність;
– мобільний додаток: створення мобільної версії системи дозволить користувачам оцінювати автомобілі в будь – який момент.

ВИСНОВКИ

У ході виконання даної дипломної роботи було розроблено нейронну мережу для визначення вартості автомобілів. Проєкт охоплює всі етапи машинного навчання: вибір даних, аналіз задачі, обробку даних, розробку моделі, оцінку її ефективності та якості, її навчання, інтеграцію в зручний графічний інтерфейс та загальне тестування системи. Результати показали, що створена система є ефективною, точною та зручною для використання.

Основні досягнення роботи:

Вибір даних:

– було проведено аналіз вихідного набору даних, що включав характеристики автомобілів (марка, рік випуску, пробіг, тип палива, стан тощо);
– виконано попередню обробку даних: заповнення пропущених значень, масштабування числових змінних та перетворення категоріальних змінних у числовий формат за допомогою методів label encoding та one – hot encoding;
– дані були поділені на тренувальний і тестовий набори в співвідношенні 80/20 для забезпечення перевірки якості моделі на нових даних.

Структура нейронної мережі:

– розроблено багатошарову нейронну мережу з використанням архітектури sequential, яка складається з трьох прихованих шарів;
– кожен шар мав функцію активації relu, що забезпечує швидке і стабільне навчання;
– для зменшення перенавчання було використано механізм dropout з імовірністю 0.3;
– вихідний шар моделі передбачав одне значення (ціна автомобіля) із застосуванням функції активації, відповідної задачі регресії;
– модель була реалізована за допомогою бібліотеки keras на базі tensorflow.

Інтеграція з графічним інтерфейсом:

– створений графічний інтерфейс на основі tkinter дозволив спростити роботу з системою для кінцевих користувачів;
– користувачі можуть вводити характеристики автомобіля, такі як марка, рік випуску та пробіг, і отримувати прогнозовану ціну за декілька секунд.

Навчання:

– було проведено навчання моделі протягом 100 епох з розміром батчу 64;
– для запобігання перенавчанню використовувався механізм earlystopping, який зупиняв навчання, якщо покращення функції втрат на валідаційному наборі припинялося протягом 10 епох;
– процес навчання відображався через графіки втрат для тренувального та валідаційного наборів, що дозволяло контролювати збіжність моделі.

Оцінка якості:

Якість моделі оцінювалася за метриками:

– mse (mean squared error): середня квадратична помилка для тестового набору склала 4,000,000 usd²;
– rmse (root mean squared error): корінь середньої квадратичної помилки дорівнював 4000 usd, що вказує на високу точність передбачення;
– mae (mean absolute error): середня абсолютна помилка склала 2500 usd.
– порівняння з іншими моделями (лінійна регресія, дерева рішень) підтвердило, що нейронна мережа перевершує їх за точністю, особливо на складних і нелінійних залежностях.

Ці досягнення підтверджують, що модель була ретельно розроблена, оптимізована та протестована, що дозволяє використовувати її в реальних умовах для точного прогнозування вартості автомобілів.

Практичне значення та перспективи використання

Практичне застосування:

- система може бути використана автодилерами для швидкої оцінки автомобілів;
- страхові компанії можуть застосовувати її для визначення ринкової вартості автомобіля;
- приватні покупці можуть оцінювати справедливу ціну перед купівлею автомобіля.
- Перспективи вдосконалення:
- розширення бази даних: додавання нових характеристик автомобілів та оновлення актуальних даних;
- інтеграція з мобільними або веб – застосунками: для підвищення доступності системи;
- додавання нових факторів: наприклад, врахування регіонального ринку або сезонності.

ПЕРЕЛІК ПОСИЛАНЬ

1. Kaggle on Python. URL : Kaggle: Your Machine Learning and Data Science Community (дата звернення: 25.03.2024);

2. Python on Python. URL : Welcome to Python.org (дата звернення: 01.04.2024);

3. Matlotlib on Python. URL: Matplotlib – Visualization with Python (дата звернення: 10.04.2024);

4. Seaborn on Python. URL: seaborn: statistical data visualization – seaborn 0.13.2 documentation (дата звернення: 25.04.2024);

5. Label – Encodering on Towardsdatascience. URL: Categorical encoding using Label-Encoding and One-Hot-Encoder | by Dinesh Yadav | Towards Data Science (дата звернення: 01.05.2024);

6. EarlyStopping on Keras. URL: EarlyStopping (дата звернення: 15.05.2024);

7. MSE on ScikitLearn. URL: mean_squared_error – scikit-learn 1.6.0 documentation (дата звернення: 25.08.2024);

8. Tkinker on DocsPython. URL: tkinter – Python interface to Tcl/Tk – Python 3.13.1 documentation (дата звернення: 25.10.2024);

9. Wiki on Wikipedia. URL: Wikipedia (дата звернення: 01.11.2024).

ДОДАТОК А

Код програми

car_price_predictor.py > create_gui

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.metrics import mean_squared_error
import tkinter as tk
from tkinter import ttk
import tensorflow as tf

# Завантаження даних з CSV
data = pd.read_csv('car_data.csv')

# Перетворення категоріальних змінних в числові
def fit_label_encoder(data, column):
    encoder = LabelEncoder()
    encoder.fit(data[column].dropna().unique())  # Використовуємо унікальні значення з тренувального набору
    return encoder

# Перетворення категоріальних змінних
brand_encoder = fit_label_encoder(data, 'brand')
fuel_encoder = fit_label_encoder(data, 'fuel_type')
engine_encoder = fit_label_encoder(data, 'engine')
transmission_encoder = fit_label_encoder(data, 'transmission')
accident_encoder = fit_label_encoder(data, 'accident')
condition_encoder = fit_label_encoder(data, 'condition')

# Перетворення категоріальних змінних у даних
data['brand'] = brand_encoder.transform(data['brand'])
data['fuel_type'] = fuel_encoder.transform(data['fuel_type'])
data['engine'] = engine_encoder.transform(data['engine'])
data['transmission'] = transmission_encoder.transform(data['transmission'])
data['accident'] = accident_encoder.transform(data['accident'])
data['condition'] = condition_encoder.transform(data['condition'])

# Вибір фіч та цільової змінної
X = data[['brand', 'model_year', 'mileage', 'fuel_type', 'engine', 'transmission', 'accident', 'condition']]
y = data['price']

# Розділення на тренувальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Нормалізація (стандартизація) даних
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Рисунок А.1 – Код програми

```python

# Створення нейронної мережі
model = Sequential()
model.add(Dense(512, input_dim=X_train_scaled.shape[1], activation='relu'))  # Більше нейронів
model.add(Dropout(0.3))  # Dropout для боротьби з переобученням
model.add(Dense(256, activation='relu'))  # Другий шар
model.add(Dropout(0.3))  # Dropout для боротьби з переобученням
model.add(Dense(128, activation='relu'))  # Третій шар
model.add(Dense(1))  # Останній шар для прогнозування ціни

# Компіляція моделі
optimizer = Adam(learning_rate=0.001)  # Налаштований оптимізатор
model.compile(optimizer=optimizer, loss='mean_squared_error')

# Раннє припинення (EarlyStopping)
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

# Тренування моделі
model.fit(X_train_scaled, y_train, epochs=100, batch_size=64, validation_data=(X_test_scaled, y_test), callbacks=[early_stopping])

# Оцінка моделі
loss = model.evaluate(X_test_scaled, y_test)
rmse = np.sqrt(loss)
#print(f'Test Loss (MSE): {loss}')
print(f'RMSE: {rmse}')

# Збереження моделі
model.save('car_price_model.h5')
print("Модель натренована та збережена.")

# Оновлений код для графічного інтерфейсу
def predict_price():
    try:
        # Використовуємо енкодери, щоб обробити введені значення
        brand = brand_encoder.transform([brand_combobox.get()])[0]
        fuel_type = fuel_encoder.transform([fuel_combobox.get()])[0]
        engine = engine_encoder.transform([engine_combobox.get()])[0]
        transmission = transmission_encoder.transform([transmission_combobox.get()])[0]
        accident = accident_encoder.transform([accident_combobox.get()])[0]
        condition = condition_encoder.transform([condition_combobox.get()])[0]

        model_year = int(year_entry.get())
        mileage = int(mileage_entry.get())

        # Формуємо вхідні дані для моделі
        input_data = np.array([[brand, model_year, mileage, fuel_type, engine, transmission, accident, condition]])
```

Рисунок А.2 – Код програми

```python
        # Масштабуємо введені дані
        input_data_scaled = scaler.transform(input_data)

        # Передбачаємо ціну
        predicted_price = model.predict(input_data_scaled)
        print(f"Predicted Price: {predicted_price[0][0]}")

        # Виведення результату
        result_label.config(text=f"Передбачена ціна: {predicted_price[0][0]:.2f} $")

    except Exception as e:
        result_label.config(text="Помилка при прогнозуванні ціни!")
        print(f"Error: {e}")

# Створення графічного інтерфейсу
def create_gui():
    global brand_combobox, year_entry, mileage_entry, fuel_combobox, engine_combobox, transmission_combobox, accident_combobox, condition_combobox, result_label

    # Створення вікна Tkinter
    window = tk.Tk()
    window.title("Прогнозування вартості автомобіля")

    # Теги для полів вводу
    tk.Label(window, text="Бренд").grid(row=0, column=0)
    tk.Label(window, text="Рік").grid(row=1, column=0)
    tk.Label(window, text="Пробіг (км)").grid(row=2, column=0)
    tk.Label(window, text="Тип палива").grid(row=3, column=0)
    tk.Label(window, text="Об'єм двигуна").grid(row=4, column=0)
    tk.Label(window, text="Трансмісія").grid(row=5, column=0)
    tk.Label(window, text="Аварія (Yes/No)").grid(row=6, column=0)
    tk.Label(window, text="Стан (New/Used)").grid(row=7, column=0)

    # Поля вводу
    brand_combobox = ttk.Combobox(window, values=['Ford', 'Chevrolet', 'Toyota', 'Honda', 'BMW', 'Nissan', 'Tesla', 'Hyundai', 'Mercedes-Benz', 'Audi', 'Volkswagen', 'Mazda', 'Subaru', 'Lexus', 'Kia'])
    brand_combobox.grid(row=0, column=1)

    year_entry = tk.Entry(window)
    year_entry.grid(row=1, column=1)

    mileage_entry = tk.Entry(window)
    mileage_entry.grid(row=2, column=1)

    fuel_combobox = ttk.Combobox(window, values=['Petrol', 'Diesel', 'Electric', 'Hybrid'])
    fuel_combobox.grid(row=3, column=1)
```

Рисунок А.3 – Код програми

```
    engine_combobox = ttk.Combobox(window, values=['3.7', '3.6', '1.8', '2.0', '2.5', '1.6', '1.4'])
    engine_combobox.grid(row=4, column=1)

    transmission_combobox = ttk.Combobox(window, values=['Automatic', 'Manual'])
    transmission_combobox.grid(row=5, column=1)

    accident_combobox = ttk.Combobox(window, values=['Yes', 'No'])
    accident_combobox.grid(row=6, column=1)

    condition_combobox = ttk.Combobox(window, values=['New', 'Used'])
    condition_combobox.grid(row=7, column=1)

    # Кнопка для прогнозування ціни
    predict_button = tk.Button(window, text="Прогнозувати ціну", command=predict_price)
    predict_button.grid(row=8, column=0, columnspan=2)

    # Мітка для виведення результату
    result_label = tk.Label(window, text="Передбачена ціна: ", font=("Arial", 14))
    result_label.grid(row=9, column=0, columnspan=2)

    # Кнопка для видалення даних
    def clear_fields():
        brand_combobox.set('')
        year_entry.delete(0, tk.END)
        mileage_entry.delete(0, tk.END)
        fuel_combobox.set('')
        engine_combobox.set('')
        transmission_combobox.set('')
        accident_combobox.set('')
        condition_combobox.set('')
        result_label.config(text="Передбачена ціна: ")

    clear_button = tk.Button(window, text="Видалити дані", command=clear_fields)
    clear_button.grid(row=10, column=0, columnspan=2)

    # Запуск вікна
    window.mainloop()

# Старт програми
if __name__ == "__main__":
    create_gui()
```

Рисунок А.4 – Код програми

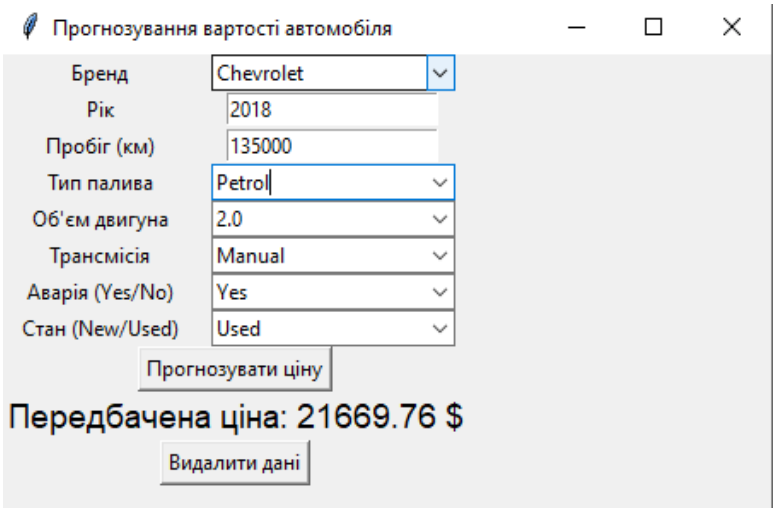


Рисунок А.5 – Приклад виконання

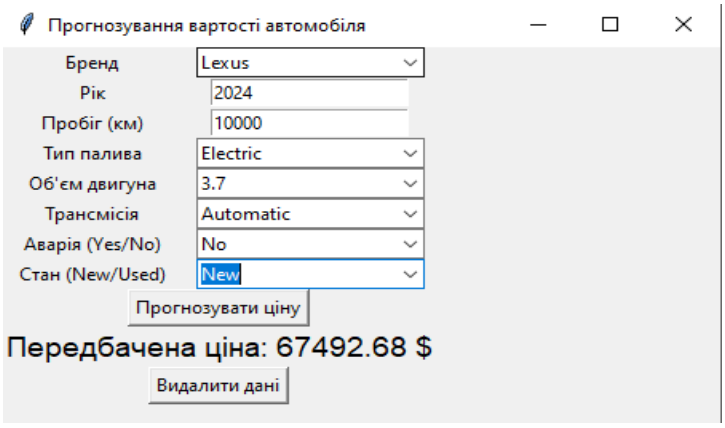


Рисунок А.6 – Приклад виконання

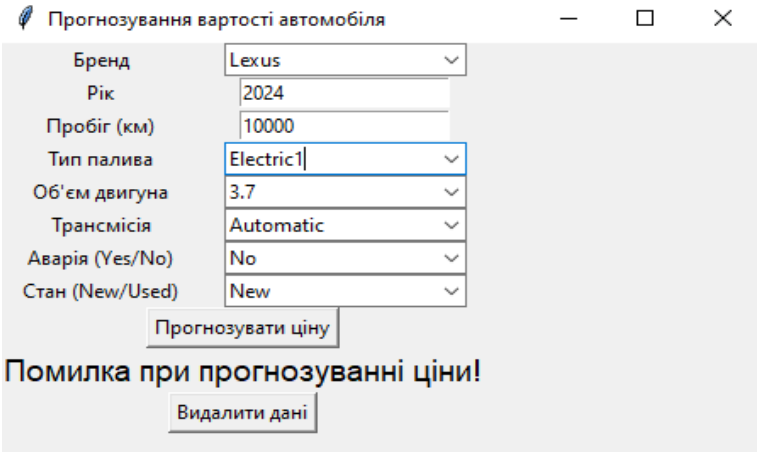


Рисунок А.7 – Приклад помилки