МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В.Н.Каразіна
Факультет математики і інформатики
Кафедра теоретичної та прикладної інформатики

Кваліфікаційна робота

бакалавр

на тему: Розробка програмного модуля оптимізації вибору автомобільної продукції з урахуванням штучного інтелекту

Виконав: студент 4 курсу, групи МФ-41
спеціальність 122 «Комп'ютерні науки»
освітньо-професійна програма «Інформатика»

Цибулько В.О.

Керівник Фролов В.В.

Харків – 2023 року

ЗМІСТ

ВСТУП

1.1 Мета і завдання дослідження

Метою цієї дипломної роботи є створення програмного модуля, спрямованого на оптимізацію вибору автомобільної продукції для компаній, які продають автомобілі. Головна мета полягає у поліпшенні швидкості та точності процесу пошуку автомобілів, які відповідають вимогам водіїв. Основні завдання цієї програми включають:

1. Вирахування коефіцієнтів автомобіля: програма буде розраховувати коефіцієнти, такі як комфорт, екологічність, технічні рішення та інші, для кожного автомобіля в базі даних.
2. Розробка штучного інтелекту: буде створено модуль штучного інтелекту, який здатний кластеризувати базу даних за допомогою вирахуваних коефіцієнтів.
3. Реалізація роботи з базою даних через графічний інтерфейс сайту: програмний модуль буде мати веб-інтерфейс, що дозволить адміністратору взаємодіяти з базою даних та налаштовувати параметри оптимізації.
4. Знаходження оптимальної кількості кластерів: штучний інтелект буде використовувати алгоритми кластеризації для визначення оптимального числа кластерів автомобілів.
5. Вивід різноманітних графіків: програмний модуль буде створювати графіки, які відображатимуть різні аспекти роботи програми. Наприклад, графіки можуть показувати розподіл автомобілів за кластерами, залежність коефіцієнтів якості від певних параметрів автомобілів, зміни кластеризації при зміні параметрів оптимізації тощо. Ці графіки надають адміністратору зручний спосіб аналізу та визначення результатів роботи програми.

1.2 Актуальність цієї теми

Головними пунктами актуальності теми використання нейронних мереж у продажу автомобілів є:

1. Зростаюча кількість доступних моделей: Автомобільний ринок постійно розширюється, пропонуючи різноманітні моделі автомобілів з різними характеристиками, функціями та цінами. Вибір оптимального автомобіля серед такого широкого асортименту може бути складним для споживача. Використання нейронних мереж допомагає аналізувати великі обсяги даних та рекомендувати автомобілі, що найкраще відповідають вимогам та перевагам користувача.

2. Покращення точності та швидкості вибору: Нейронні мережі можуть використовувати складні алгоритми машинного навчання для аналізу великого обсягу інформації про автомобілі, включаючи технічні характеристики, відгуки користувачів, економічні показники тощо. Це дозволяє забезпечити більш точні та персоналізовані рекомендації щодо вибору автомобіля. Крім того, використання нейронних мереж може значно прискорити процес вибору, зменшуючи час, який споживач витрачає на пошук та аналіз інформації.

3. Автоматизація процесу вибору: Застосування нейронних мереж для оптимізації вибору автомобілів може зробити цей процес більш автоматизованим. Наприклад, за допомогою мобільних додатків або онлайн-сервісів, користувачі можуть ввести свої вимоги та уподобання, а нейронна мережа буде пропонувати найкращі варіанти автомобілів на основі цих вхідних даних. Це може заощадити час та зусилля покупця, спрощуючи процес вибору.

1.3 Постановка задачі

1. Сформулювати алгоритми та методи, які будуть використовуватися для розрахунку коефіцієнтів якості автомобілів на основі їх характеристик. Врахувати різні фактори, такі як вартість, безпека, паливна ефективність тощо.

2. Розробити програмний модуль, який буде використовуватися для збору та обробки даних про автомобілі. Використати нейронні мережі для кластеризації бази даних автомобілів з метою знаходження груп схожих моделей.

3. Виконати факторний аналіз для визначення основних факторів, що впливають на якість автомобіля. Здійснити статистичний аналіз та побудувати відповідні моделі, що враховують ці фактори.

4. Розробити інтерфейс адміністратора, який буде забезпечувати зручну та ефективну роботу з програмним модулем. Інтерфейс повинен дозволяти адміністратору вводити нові дані про автомобілі, редагувати чинні записи та

1.4 Розвинутий огляд сучасного стану справ в області

Зміни на ринку автомобілів: Ринок автомобілів в останні роки підвищив свою конкурентоспроможність та розширився завдяки появі нових технологій, електричних та гібридних автомобілів, а також зростанню інтересу до зелених технологій. Це призвело до збільшення розмаїття автомобільної продукції та вибору для споживачів.

1. Використання штучного інтелекту для продажу автомобілів: Штучний інтелект знайшов широке застосування в автомобільній індустрії, зокрема у сфері продажу автомобілів. Він використовується для аналізу даних клієнтів, рекомендаційних систем, персоналізованих пропозицій та

прогнозування попиту. Штучний інтелект допомагає покращити взаємодію з клієнтами та забезпечити більш точний та ефективний продаж автомобілів.

2. Використання методів класифікації та кластерного аналізу: Методи класифікації та кластерного аналізу дозволяють групувати автомобілі за подібними характеристиками та критеріями. Це допомагає покупцям знайти автомобіль, який найкраще відповідає їхнім вимогам та потребам. Ці методи використовуються для створення моделей рекомендацій та персоналізованого підбору автомобілів для клієнтів.

3. Використання штучного інтелекту та методів оптимізації для продажу в загальному: Штучний інтелект та оптимізаційні методи знайшли широке застосування у сфері продажу різних продуктів. Вони допомагають виробникам та продавцям аналізувати дані споживачів, розуміти їх поведінку та вподобання, прогнозувати попит та розробляти ефективні стратегії маркетингу. Використання штучного інтелекту та методів оптимізації дозволяє покращити процеси продажу, збільшити ефективність рекламних кампаній, підвищити задоволення споживачів та збільшити конкурентоспроможність на ринку.

4. Розширення застосування на інші галузі: Крім автомобільної індустрії, штучний інтелект та методи оптимізації знаходять своє застосування в багатьох інших галузях для продажу різних продуктів. Наприклад, в електронній комерції вони використовуються для персоналізації пропозицій, рекомендаційних систем та оптимізації ціноутворення. У сфері роздрібної торгівлі вони допомагають управляти запасами та передбачити попит. Такі технології також використовуються у сфері туризму, фінансів, медицини та інших галузях.

Штучний інтелект (ШІ) стає все більш поширеним інструментом для автомобільних компаній та сервісів для оптимізації процесів продажу автомобілів та поліпшення взаємодії з клієнтами. Ось кілька прикладів компаній та сервісів, які використовують ШІ для продажу автомобілів:

1. Tesla: Компанія Tesla, яка виробляє електричні автомобілі, використовує штучний інтелект для ряду цілей. Їх система автопілота базується на нейронних мережах та алгоритмах машинного навчання, що допомагають автомобілю виконувати автономні функції. Крім того, Tesla використовує ШІ для аналізу даних клієнтів та персоналізації пропозицій під час продажу автомобілів.

2. Carvana: Carvana є онлайн-платформою для купівлі та продажу автомобілів, яка використовує штучний інтелект для поліпшення процесу продажу. Вони використовують алгоритми машинного навчання для аналізу даних про клієнтів, оцінки автомобілів, автоматичного складання контрактів та інші цифрові рішення для зручності клієнтів.

3. CarGurus: CarGurus - це онлайн-платформа, яка допомагає покупцям знаходити та порівнювати автомобілі. Вони використовують штучний інтелект для аналізу та ранжування оголошень про автомобілі, що допомагає покупцям знайти оптимальний варіант відповідно до їхніх вимог та бюджету.

4. Amazon Alexa та Google Assistant: Ці голосові помічники використовують штучний інтелект для інтеграції з автомобілями та полегшення процесу продажу. Користувачі можуть взаємодіяти з цими голосовими помічниками, щоб отримати інформацію про доступні автомобілі, розрахувати вартість, запланувати тест-драйв або навіть здійснити покупку через підключені до них сервіси. Використання штучного інтелекту допомагає забезпечити зручну та персоналізовану взаємодію з потенційними покупцями.

5. Shift Technologies: Shift Technologies є платформою для купівлі та продажу автомобілів, яка використовує штучний інтелект для поліпшення процесу продажу. Вони застосовують алгоритми машинного навчання для оцінки автомобілів, персоналізованих рекомендацій покупцям, автоматизованого ціноутворення та визначення найбільш привабливих пропозицій для клієнтів.

Штучний інтелект (ШІ) широко використовується компаніями в різних галузях для оптимізації процесів продажу та покращення взаємодії з клієнтами.

Ось кілька прикладів компаній, які використовують ШІ для продажу товарів, включаючи не автомобілі:

1. Amazon: Amazon використовує штучний інтелект для персоналізації пропозицій та рекомендацій покупцям. За допомогою алгоритмів машинного навчання та аналізу даних про покупців, Amazon може пропонувати індивідуальні товари, які максимально відповідають інтересам та потребам кожного клієнта.

2. Alibaba: Alibaba, один з найбільших електронних комерційних майданчиків у світі, використовує штучний інтелект для розпізнавання образів, обробки мови та аналізу даних. Це допомагає покупцям швидше знаходити необхідні товари, а також дозволяє продавцям пропонувати більш персоналізовані пропозиції.

3. Netflix: Netflix використовує штучний інтелект для рекомендацій фільмів і серіалів. ШІ аналізує дані про перегляди, пристрасті та попередні вподобання глядачів, щоб рекомендувати контент, який найбільше відповідає їхнім смакам, покращуючи таким чином користувальницький досвід.

4. eBay: eBay використовує штучний інтелект для поліпшення пошуку та рекомендацій товарів. ШІ допомагає аналізувати мільйони оголошень та враховувати інтереси покупців, їхні попередні покупки та зворотний зв'язок, щоб пропонувати найбільш відповідні товари та покращувати користувацький досвід.

5. Sephora: Sephora, компанія з продажу косметики, використовує штучний інтелект для надання персоналізованих рекомендацій. За допомогою візуального аналізу обличчя та алгоритмів машинного навчання, Sephora може рекомендувати клієнтам продукти, які найкраще відповідають їхньому типу шкіри, тону обличчя та іншим індивідуальним характеристикам.

1.5 Структура дипломної роботи

Робота складається зі вступу, двох глав, загальних висновків, списку використаної літератури (9). Зміст роботи висвітлено на 46 сторінках основного тексту і містить 24 рисунки.

РОЗРОБКА МОДЕЛІ АВТОМОБІЛЯ ТА ПОБУДОВА БАЗИ ДАНИХ

2.1 Розробка моделі автомобіля

За статтею [1 - 2] можна виділити 3 основні частини автомобіля: шасі, двигун та кузов. Крім того, з цих статей можна виділити основні характеристики автомобіля, такі як ціна, маса, витрати на технічне обслуговування, періодичність технічного обслуговування, гальмівний шлях в метрах, довжина колісної бази, середня витрата палива на 100 км, мінімальна витрата палива на 100 км, максимальна швидкість у кілометрах на годину, розгін до 100 кілометрів на годину, об'єм багажника, об'єм салону, довжина бази автомобіля та кількість подушок безпеки, а також вид палива (А-92, А-95 і т. д.). (див. рис 1)

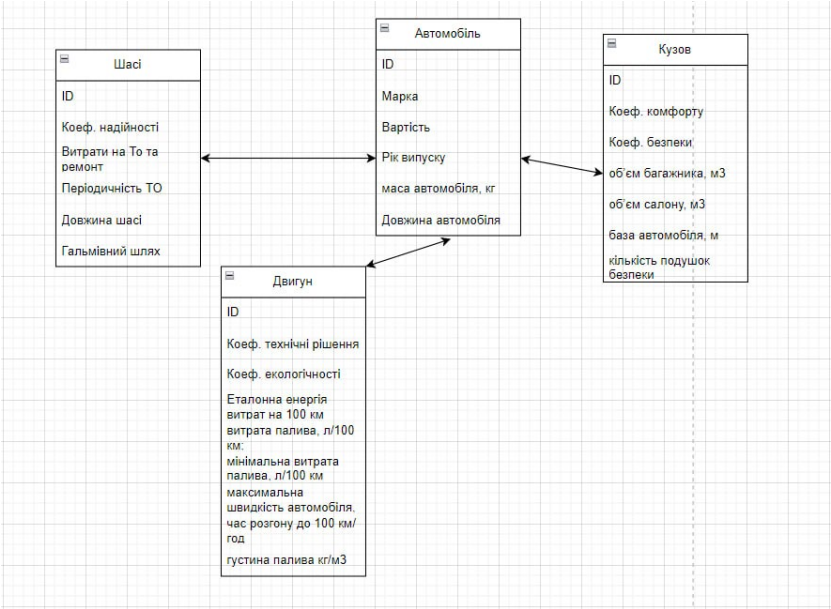


Рис. 1. Складові частини автомобіля та характеристики для вирахування коефіцієнтів

Так це буде виглядати на макеті реального автомобіля(див. рис. 2)

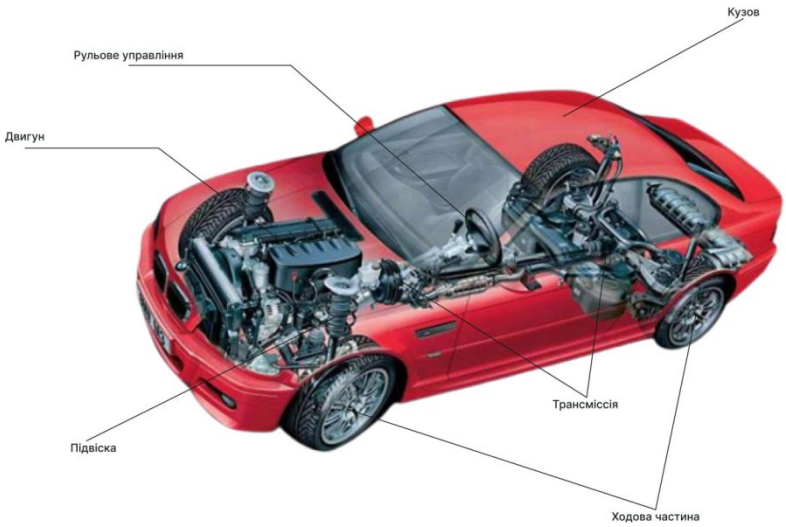


Рис. 2. Модель автомобіля

Ці характеристики дозволяють розрахувати коефіцієнти, за допомогою яких можна оцінити автомобіль з різних аспектів, таких як комфорт, надійність, безпека, технічні рішення та екологічність.

Ці коефіцієнти рахуються за наступними формулами

Коефіцієнт комфорту

$$Кф = \frac{Vб \times Lб}{2.1Vc}$$

Де Vб – об’єм багажника, м3 Vc – об’єм салону, м3 Lб – база автомобіля, м

Інший коефіцієнт — це коефіцієнт безпеки (Кб), який враховує кількість подушок безпеки, довжину колісної бази, гальмівний шлях і мінімальний гальмівний шлях серед усіх автомобілів. Формула для розрахунку Кб

$$Кб = \frac{St}{Sth(n\ п.б + \frac{La}{Lб})}$$

Де n п.б – кількість подушок безпеки, Lб – база автомобіля, м, Lф – довжина колії автомобіля, м, St – гальмівний шлях, Sth – мінімальний гальмівний шлях серед всіх автомобілів

Третій коефіцієнт — це коефіцієнт надійності (Кн), який враховує вартість нового автомобіля, вартість одного літра палива, мінімальну витрату палива автомобілем, гарантійний пробіг автомобіля та інші фактори. Формула для розрахунку Кн:

$$Кн = \frac{0{,}01\ Hl.min \times Ct \times Lгар}{Kв \times Савт}$$

Де Савт – вартість нового автомобіля, грн,СТ – вартість одного літра палива, грн, Нл.min - мінімальна витрата палива автомобілем, л/100 км, Lгар - гарантійний пробіг автомобіля, км

Четвертий коефіцієнт — це коефіцієнт технічних рішень (Кт), який враховує мінімальну витрату палива автомобілем, розгін до 100 км/год, густину палива, максимальну швидкість та масу автомобіля. Формула для розрахунку Кт:

$$Кт = \frac{0{,}036 Hl.\min \times t \times p \times Vmax}{Ga}$$

Де Нл.min - мінімальна витрата палива автомобілем, л/100 км, t – розгін до 100 км/год с., p – густина палива кг/м3, Vmax – максимальна швидкість км/год, Ga – маса автомобіля, кг.

П'ятий коефіцієнт — це коефіцієнт екологічності (Ке), який враховує мінімальну витрату палива автомобілем, максимальну швидкість та допустиму норму оксиду азоту згідно з екологічним стандартом. Формула для розрахунку Ке

$$Ке = \frac{0{,}0033\ Hl.min \times Vmax}{Kno \times Va}$$

Де Нл.min - мінімальна витрата палива автомобілем, Vmax – максимальна швидкість км/год, Kno – допустима норма оксиду азоту за стандартом (Євро-6), г/км. Va = 0,5·Vmax

Оцінку якості автомобіля можна здійснити шляхом формування інтегрального показника (KI).

Знаючи параметри оцінюваного автомобіля та значення показників якості, можна розрахувати інтегральний показник. Інтегральний показник (KI) об'єднує усі коефіцієнти комфорту, надійності, безпеки, технічних рішень та екологічності в один показник. Метою розрахунку інтегрального показника є визначення загальної оцінки якості автомобіля.

Формула для розрахунку інтегрального показника KI: KI = Кф + Кн + Кб + Ке + Кт → max.

Отже, шляхом обчислення кожного коефіцієнта і додавання їх значень можна отримати інтегральний показник, який відображатиме загальну якість

автомобіля з урахуванням комфорту, надійності, безпеки, технічних рішень та екологічних аспектів.

Цей підхід дозволяє перетворити багатовимірну задачу оцінки якості на двовимірну, що спрощує процес порівняння різних автомобілів та вибору оптимального варіанту.

Розрахунок інтегрального показника KI допомагає автомобільним експертам, покупцям та виробникам зробити об'єктивний аналіз і порівняння різних моделей автомобілів з урахуванням різних аспектів їх характеристик і властивостей.

Отже, використовуючи вищезазначені формули та показники, можна систематично оцінювати автомобілі та здійснювати раціональний вибір, враховуючи власні пріоритети та потреби.

2.2 Факторний аналіз

Факторний аналіз є статистичним методом, який використовується для виявлення прихованих взаємозв'язків або факторів серед набору змінних. Його основна мета полягає в знаходженні невидимих факторів, які пояснюють спільну варіативність між спостережуваними змінними [3].

При факторному аналізі змінні групуються в "фактори" відповідно до ступеня кореляції між ними. Кожен фактор є латентною (прихованою) змінною, яка не може бути виміряна безпосередньо, але може бути виявлена через кореляції між спостережуваними змінними.

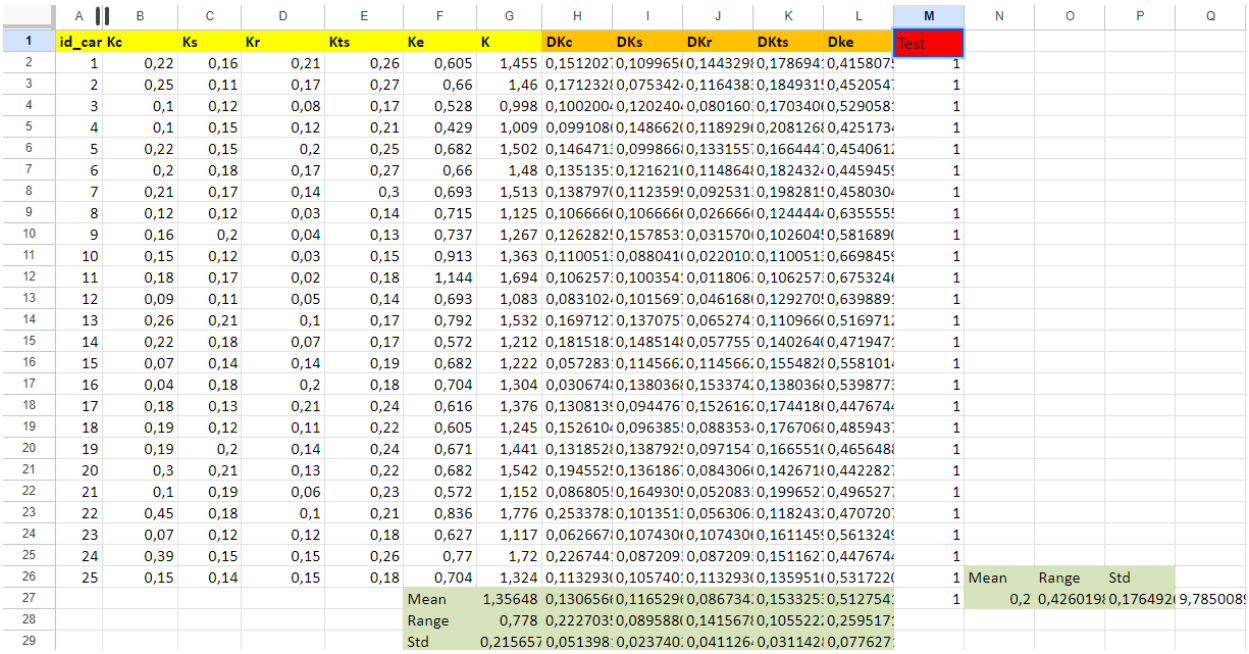
Факторний аналіз може мати кілька цілей:

1. Скорочення розмірності даних: Факторний аналіз дозволяє зменшити кількість змінних, зберігаючи при цьому інформацію про спільну варіативність між ними. Це дозволяє зробити подальший аналіз даних більш простим і зрозумілим.

2. Виявлення структури: Факторний аналіз може допомогти виявити приховану структуру або організацію в наборі змінних. Наприклад, він може допомогти виявити, які змінні схильні до спільності та групуватися разом.

3. Конструктна валідація: Факторний аналіз допомагає перевірити, чи відповідають дані певним концептуальним конструктам або теоретичним моделям. Він може підтвердити або спростувати гіпотези про взаємозв'язки між змінними.

Узявши характеристики 25 машин та підрахувавши їх коефіцієнти за формулами, які були приведенні до цього, було перенесено ці дані в Excel-таблицю та отримано такий результат (див. рис. 3).



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	id_car	Kc	Ks	Kr	Kts	Ke	K	DKc	DKs	DKr	DKts	Dke	Test				
2	1	0,22	0,16	0,21	0,26	0,605	1,455	0,151202	0,109965	0,144329	0,178694	0,415807	1				
3	2	0,25	0,11	0,17	0,27	0,66	1,46	0,171232	0,075342	0,116438	0,184931	0,452054	1				
4	3	0,1	0,12	0,08	0,17	0,528	0,998	0,100200	0,120240	0,080160	0,170340	0,529058	1				
5	4	0,1	0,15	0,12	0,21	0,429	1,009	0,099108	0,148662	0,118929	0,208126	0,425173	1				
6	5	0,22	0,15	0,2	0,25	0,682	1,502	0,146471	0,099866	0,133155	0,166444	0,454061	1				
7	6	0,2	0,18	0,17	0,27	0,66	1,48	0,135135	0,121621	0,114864	0,182432	0,445945	1				
8	7	0,21	0,17	0,14	0,3	0,693	1,513	0,138797	0,112359	0,092531	0,198281	0,458030	1				
9	8	0,12	0,12	0,03	0,14	0,715	1,125	0,106666	0,106666	0,026666	0,124444	0,635555	1				
10	9	0,16	0,2	0,04	0,13	0,737	1,267	0,126282	0,157853	0,031570	0,102604	0,581689	1				
11	10	0,15	0,12	0,03	0,15	0,913	1,363	0,110051	0,088041	0,022010	0,110051	0,669845	1				
12	11	0,18	0,17	0,02	0,18	1,144	1,694	0,106257	0,100354	0,011806	0,106257	0,675324	1				
13	12	0,09	0,11	0,05	0,14	0,693	1,083	0,083102	0,101569	0,046168	0,129270	0,639889	1				
14	13	0,26	0,21	0,1	0,17	0,792	1,532	0,169712	0,137075	0,065274	0,110966	0,516971	1				
15	14	0,22	0,18	0,07	0,17	0,572	1,212	0,181518	0,148514	0,057755	0,140264	0,471947	1				
16	15	0,07	0,14	0,14	0,19	0,682	1,222	0,057283	0,114566	0,114566	0,155482	0,558101	1				
17	16	0,04	0,18	0,2	0,18	0,704	1,304	0,030674	0,138036	0,153374	0,138036	0,539877	1				
18	17	0,18	0,13	0,21	0,24	0,616	1,376	0,130813	0,094476	0,152616	0,174418	0,447674	1				
19	18	0,19	0,12	0,11	0,22	0,605	1,245	0,152610	0,096385	0,088353	0,176706	0,485943	1				
20	19	0,19	0,2	0,14	0,24	0,671	1,441	0,131852	0,138792	0,097154	0,166551	0,465648	1				
21	20	0,3	0,21	0,13	0,22	0,682	1,542	0,194552	0,136186	0,084306	0,142671	0,442282	1				
22	21	0,1	0,19	0,06	0,23	0,572	1,152	0,086805	0,164930	0,052083	0,199652	0,496527	1				
23	22	0,45	0,18	0,1	0,21	0,836	1,776	0,253378	0,101351	0,056306	0,118243	0,470720	1				
24	23	0,07	0,12	0,12	0,18	0,627	1,117	0,062667	0,107430	0,107430	0,161145	0,561324	1				
25	24	0,39	0,15	0,15	0,26	0,77	1,72	0,226744	0,087209	0,087209	0,151162	0,447674	1				
26	25	0,15	0,14	0,15	0,18	0,704	1,324	0,113293	0,105740	0,113293	0,135951	0,531722	1	Mean	Range	Std	
27						Mean	1,35648	0,130656	0,116529	0,086734	0,153325	0,512754	1	0,2	0,426019	0,176492	9,785008
28						Range	0,778	0,222703	0,089588	0,141567	0,105522	0,259517					
29						Std	0,215657	0,051398	0,023740	0,041126	0,031142	0,077627					

Рис. 3. Таблиця з коефіцієнтами

У цій таблиці зазначено наступні коефіцієнти: Kc – коефіцієнт комфорту, Ks – коефіцієнт безпеки, Kr – коефіцієнт надійності, Kts – коефіцієнт технічних рішень, Ke – коефіцієнт екологічності, K – сума усіх коефіцієнтів. Також було підраховано середнє значення кожного коефіцієнта.

Далі було помічено таку закономірність: якщо скласти середнє значення Kc, Ks, Kr, Kts, то воно буде приблизно дорівнювати Ke. Наприклад, 0.13 + 0.11 + 0.08 + 0.15 = 0.47, тоді як середнє значення Ke дорівнює 0.51. З цього зроблено

висновок, що значення коефіцієнтів не збалансовані, оскільки один коефіцієнт більший за інші. Це сталося через те, що в статтях[1 - 2], які були використанні, не були наведені формули для розрахунку збалансованих коефіцієнтів.

Незбалансовані коефіцієнти для оцінки якості автомобіля можуть мати декілька негативних наслідків:

1. Непрозорість оцінки: Якщо один коефіцієнт має значно більше ваги, ніж інші, то кінцевий результат оцінки може бути спотворений і непрозорий. Це ускладнює розуміння того, як саме було здійснене порівняння і визначено кращий варіант.

2. Втрата об'єктивності: Незбалансовані коефіцієнти можуть спотворити об'єктивність процесу оцінки. Якщо певному коефіцієнту надано надмірну вагу, це може спричинити приховану перевагу або недооцінку конкретного аспекту якості автомобіля.

3. Неврахування індивідуальних потреб: Різні люди мають різні пріоритети та вимоги до автомобілів. Якщо один коефіцієнт переважає над рештою, це може призвести до ігнорування індивідуальних потреб користувачів. Наприклад, комфорт може бути менш важливим для деяких людей, ніж безпека або екологічність.

4. Втрата комплексного підходу: Комплексний підхід до оцінки якості автомобіля передбачає зважження різних аспектів і їх взаємозв'язок. Незбалансовані коефіцієнти можуть спотворити цей підхід і втратити цілісну картину про якість автомобіля.

Отже, збалансованість коефіцієнтів є важливою для забезпечення об'єктивної, прозорої та комплексної оцінки якості автомобіля, яка враховує індивідуальні потреби. Незбалансовані коефіцієнти можуть призвести до підвищення значущості одних аспектів якості автомобіля за шкоду іншим. Це може привести до вибору автомобіля, який не відповідає повністю потребам та вимогам користувача.

Щоб це виправити, було об'єднано перші чотири коефіцієнти в один кумулятивний коефіцієнт, що збалансує ці коефіцієнти. Крім того, це дуже

допоможе нам, оскільки тепер нам не потрібно буде створювати п'ятивимірну сітку, а лише двовимірну, що значно прискорить алгоритм та полегшить написання програми.

Тепер будемо розробляти подальшу програму на основі нового кумулятивного коефіцієнта, який складається з Kc – коефіцієнта комфорту, Ks – коефіцієнта безпеки, Kr – коефіцієнта надійності, Kts – коефіцієнта технічних рішень, а Ke – коефіцієнт екологічності буде окремо.

2.3 Побудова бази даних

Беручи до уваги усі аспекти які виникли при розробці моделі автомобіля та факторного аналізу було побудовано базу даних на основі реляційної бази даних. За основу вибрана СУБД MySql(див. рис. 4).

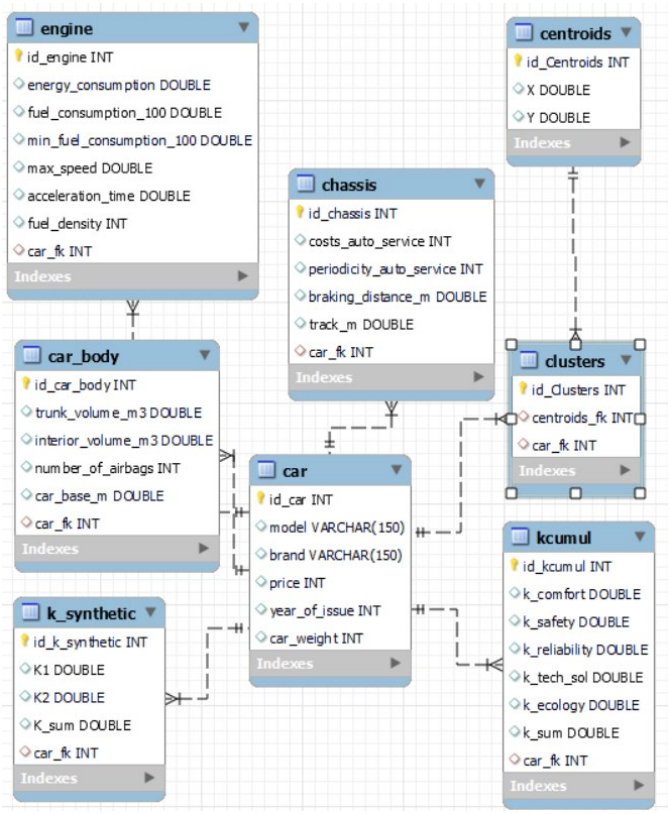


Рис. 4. Діаграма бази даних

Стаття [4] зазначає що Реляційна база даних має декілька переваг порівняно з іншими типами баз даних, особливо з урахуванням того, як був проведений факторний аналіз. Ось деякі переваги реляційних баз даних:

1. Структурованість та організація даних: Реляційна база даних заснована на табличній структурі, де дані представлені у вигляді таблиць з рядками і стовпцями. Це дозволяє логічно організувати дані, розділити їх на окремі таблиці та встановити відношення між ними за допомогою ключів. Це сприяє простоті та зрозумілості бази даних, що полегшує проведення факторного аналізу.

2. Запити та аналітичні можливості: Реляційні бази даних підтримують потужну мову запитів (наприклад, SQL), що дозволяє виконувати складні запити до даних. Це надає вам гнучкість та можливість проводити різні аналітичні операції, включаючи факторний аналіз, обчислення статистичних показників, сортування, фільтрацію та групування даних.

3. Інтеграція та спільний доступ до даних: Реляційна база даних дозволяє кільком користувачам одночасно працювати з даними та здійснювати спільний доступ до них. Це важливо для колективної роботи над факторним аналізом, де різні експерти можуть працювати з однією базою даних, вносити зміни та аналізувати результати.

У цій базі даних можна зазначити декілька таблиць, які включають такі основні складові автомобіля: "car" (автомобіль), "car_body" (кузов автомобіля), "chassis" (шасі) та "engine" (двигун). У цих таблицях зберігаються дані щодо параметрів, які були враховані при розробці моделі автомобіля. Також, ці таблиці містять характеристики, за якими вираховуються коефіцієнти якості автомобіля.

Таблиця "kcumul" використовується для зберігання всіх коефіцієнтів автомобіля, а також для обчислення сумарного значення цих коефіцієнтів. Це дозволяє нам отримати загальний показник якості автомобіля шляхом зіставлення та аналізу різних параметрів.

Таблиця "k_syntetic" містить стовпці "K1", "K2" та "K_sum". "K1" є кумулятивним коефіцієнтом, який складається з таких компонентів: "Kc" -

коефіцієнт комфорту, "Ks" - коефіцієнт безпеки, "Kr" - коефіцієнт надійності та "Kts" - коефіцієнт технічних рішень. "K2" представляє коефіцієнт екологічності, а "K_sum" є сумою значень "K1" та "K2". Ця таблиця дозволяє нам об'єднати різні аспекти якості автомобіля в один загальний показник.

Таблиця "centroid" містить координати "X" та "Y", які вказують на центри кластерів. Ця таблиця використовується для проведення кластерного аналізу, де автомобілі групуються в кластери в залежності від їх характеристик. Вона допомагає нам зрозуміти, які автомобілі належать до одного кластера та визначити подібність між ними. Це дозволяє нам отримати глибше розуміння структури та характеристик автомобільного ринку.

Таблиця "clusters" є проміжною таблицею, яка встановлює зв'язок між автомобілем і кластером, в якому він знаходиться. Кожен автомобіль може бути призначений до одного кластеру, в залежності від схожості його характеристик з іншими автомобілями. Це допомагає нам краще розуміти групу автомобілів, які мають схожі властивості і можуть бути спрямовані на певний ринковий сегмент або споживацьку групу.

Загалом, ця база даних надає зручність та організованість для зберігання та обробки даних про автомобілі. Вона дозволяє виконувати факторний аналіз, розраховувати коефіцієнти якості та проводити кластерний аналіз, що сприяє отриманню цінної інформації щодо якості, характеристик та структури автомобільного ринку.

РОЗРОБКА ПРОГРАМИ

3.1 Кластеризація та Нейронні мережі

Щоб почати процес кластеризації, важливо визначити оптимальну кількість кластерів для бази даних. Один з широко використовуваних критеріїв для визначення оптимальної кількості кластерів є критерій Калинського (Calinski-Harabasz).

За статтею [5] можна виділити що критерій Калинського є одним із числових показників, який використовується для оцінки якості кластеризації. Він вимірює подібність між даними точками в межах кожного кластера та відмінність між кластерами. Цей критерій є мірою компактності внутрішньокластерних відстаней та розсіяння міжкластерних відстаней.

Використання критерію Калинського допомагає визначити оптимальну кількість кластерів або кращу конфігурацію кластерів, а також оцінити ефективність кластеризаційного алгоритму. Він дозволяє порівняти різні моделі кластеризації та обрати найкращу за критерієм оптимальної поділки на кластери.

Для реалізації критерію Калинського використовується мова програмування Python. Використовуємо бібліотеку clustergram для реалізації кластерограми та метод calinski_harabasz_score() для обчислення значення критерію Калинського.

Кластерограма — це графічне представлення результатів кластерного аналізу, де по горизонтальній вісі відображаються різні кількості кластерів, а по вертикальній вісі — значення критерію Калинського. Цей графік дозволяє нам визначити оптимальну кількість кластерів для даних.

Метод calinski_harabasz_score() обчислює значення критерію Калинського для кожної кількості кластерів і повертає результат у вигляді числової оцінки. Чим вище значення критерію, тим краща кластеризація.

Стаття [6] говорить що знаючи кількість кластерів які будуть в базі даних можна зробити розбивку на кластери завдяки SOM (Self-Organizing Map)

SOM (Self-Organizing Map), також відома як картка Кохонена, є одним з найпоширеніших методів навчання без учителя для виявлення структури в наборі даних. Вона використовується для кластеризації та візуалізації даних.

SOM є нейромережевим алгоритмом, який використовується для перетворення простору високої розмірності даних на двовимірну (або навіть тривимірну) сітку нейронів, відому як "карта". Кожен нейрон на карті відповідає певним векторам вхідних даних і має ваги, які оновлюються під час процесу навчання.

Процес навчання SOM включає наступні кроки:

1. Ініціалізація: Ваги нейронів ініціалізуються випадковими значеннями або за допомогою іншого алгоритму ініціалізації.

2. Вибір вхідного вектора: З набору даних вибирається випадковий вхідний вектор.

3. Вибір найближчого нейрона: Обчислюється відстань між вхідним вектором і вагами нейронів на карті, і обирається нейрон з найближчими вагами (найближчий сусід).

4. Оновлення ваг нейрона: Нейрон з найближчими вагами та його сусіди оновлюють свої ваги, щоб бути більш подібними до вхідного вектора. Це оновлення здійснюється з урахуванням функції впливу, яка залежить від відстані між нейронами на карті.

5. Повторення: Кроки 2-4 повторюються для кожного вхідного вектора протягом заданої кількості епох навчання.

6. Фінальна структура: Після завершення процесу навчання SOM формує фінальну структуру, яка відображає структуру даних на карті. Кожен нейрон на карті представляється своїми оновленими вагами, які відображають характеристики, що найкраще відповідають вхідним даним. Таким чином, суміжні нейрони на карті зазвичай відповідають схожим кластерам або групам даних.

SOM має декілька важливих параметрів:

1. Розмір картки: Визначає кількість нейронів або розмір картки, яка визначає вимір простору, в якому будуть відображатися дані.

2. Кількість епох навчання: Визначає кількість повторень кроків навчання, під час яких ваги нейронів оновлюються.

3. Функція впливу (learning rate function): Визначає, як швидко ваги нейронів оновлюються під час навчання. Зазвичай використовуються функції зі зменшенням швидкості навчання з плином часу або епох.

4. Радіус впливу (influence radius): Визначає радіус околу найближчого нейрона, в якому ваги нейронів оновлюються під час навчання. Це дозволяє розповсюджувати оновлення на сусідні нейрони та забезпечує формування кластерів на карті.

Для реалізації SOM (Self-Organizing Map) у Python використовується бібліотека MiniSom. Ця бібліотека надає зручний інтерфейс для створення і навчання SOM.

У реалізації SOM треба вказати деякі параметри, такі як sigma (радіус впливу нейронів), learning_rate (швидкість навчання) та train_random (кількість епох). Ці параметри дозволяють налаштувати процес навчання SOM під конкретну задачу.

У програмі встановлено наступні значення параметрів: sigma – 1.0, learning_rate – 0.5, train_random – 1000. Вважається, що ця конфігурація є найкращою для цієї задачі.

Після навчання SOM використовуючи критерій Калинського, отримано певний результат, який можна побачити(див. рис 5).

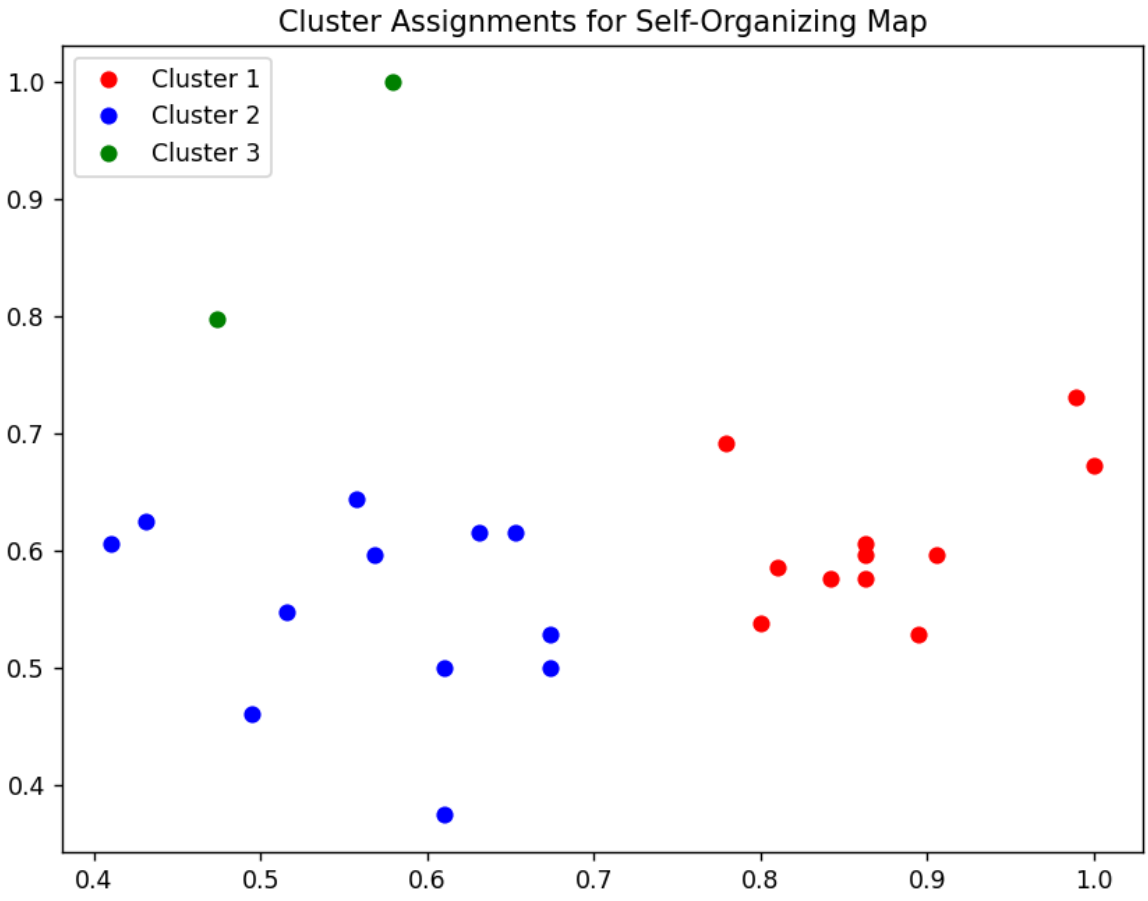


Рис. 5. Кластеризація

На графіку, що представлений (рис. 5), відображається розташування всіх автомобілів на площині, де вісь X представляє нормований кумулятивний коефіцієнт якості автомобіля, а вісь Y відповідає нормованому коефіцієнту екологічності.

Застосування критерію Калинського дозволило визначити оптимальну кількість кластерів, якою є 3 в цьому випадку. Використовуючи SOM, розділено всі автомобілі на 3 кластери. Кластеризація допомагає групувати автомобілі зі схожими характеристиками, що сприяє отриманню інформації про їхні залежності та взаємозв'язки.

3.2 Побудова графіків

Виведення різноманітних графіків для аналізу є важливою складовою процесу кластеризації для адміністратора. Графіки надають візуальну інформацію, яка допомагає зрозуміти структуру даних, відношення між кластерами та характеристики кожного кластера. Ось кілька причин, чому виведення різноманітних графіків є важливим:

1. Візуалізація кластерів: Графіки дозволяють адміністратору побачити розподіл даних в кластерах. Наприклад, графік розсіювання може відобразити положення зразків у просторі та дозволяти виділяти кластери різними кольорами або маркерами. Це допомагає адміністратору зрозуміти, які кластери існують у даних та як вони розташовані.

2. Візуалізація відношень: Графіки можуть показати взаємозв'язки між різними змінними та їх вплив на кластеризацію. Наприклад, графік кореляції може показати ступінь взаємозв'язку між різними ознаками та виявити, які ознаки можуть бути важливими для визначення кластерів.

3. Аналіз кількості: Графіки можуть показати розподіл кількості зразків у кожному кластері. Наприклад, стовпчиковий графік може відобразити кількість автомобілів у кожному кластері, дозволяючи адміністратору зрозуміти, як різні кластери відрізняються за розміром або кількістю зразків.

4. Візуалізація порівнянь: Графіки можуть допомогти адміністратору порівняти різні аспекти кластерів. Наприклад, діаграма розсіювання з двома ознаками може відобразити розподіл кластерів у просторі та вказати, які кластери мають схожі або відмінні характеристики.

Всі ці графіки та візуалізації допомагають адміністратору отримати глибше розуміння структури та характеристик кластерів у базі даних. Вони дозволяють легше спостерігати та аналізувати дані, виявляти закономірності, знаходити аномалії та приймати обґрунтовані рішення на основі цих даних. Графіки є

потужним інструментом в аналізі кластеризації, оскільки вони доповнюють числові дані та роблять аналіз більш доступним та зрозумілим.

В програмі додано декілька графіків, які надають корисну інформацію адміністратору для аналізу результатів кластеризації.

На графіку "Усі автомобілі на площині" (див. рис. 6) відображені всі нормовані автомобілі як точки на площині. Цей графік дозволяє адміністратору побачити розподіл автомобілів у просторі ознак та виявити можливі взаємозв'язки між ними.

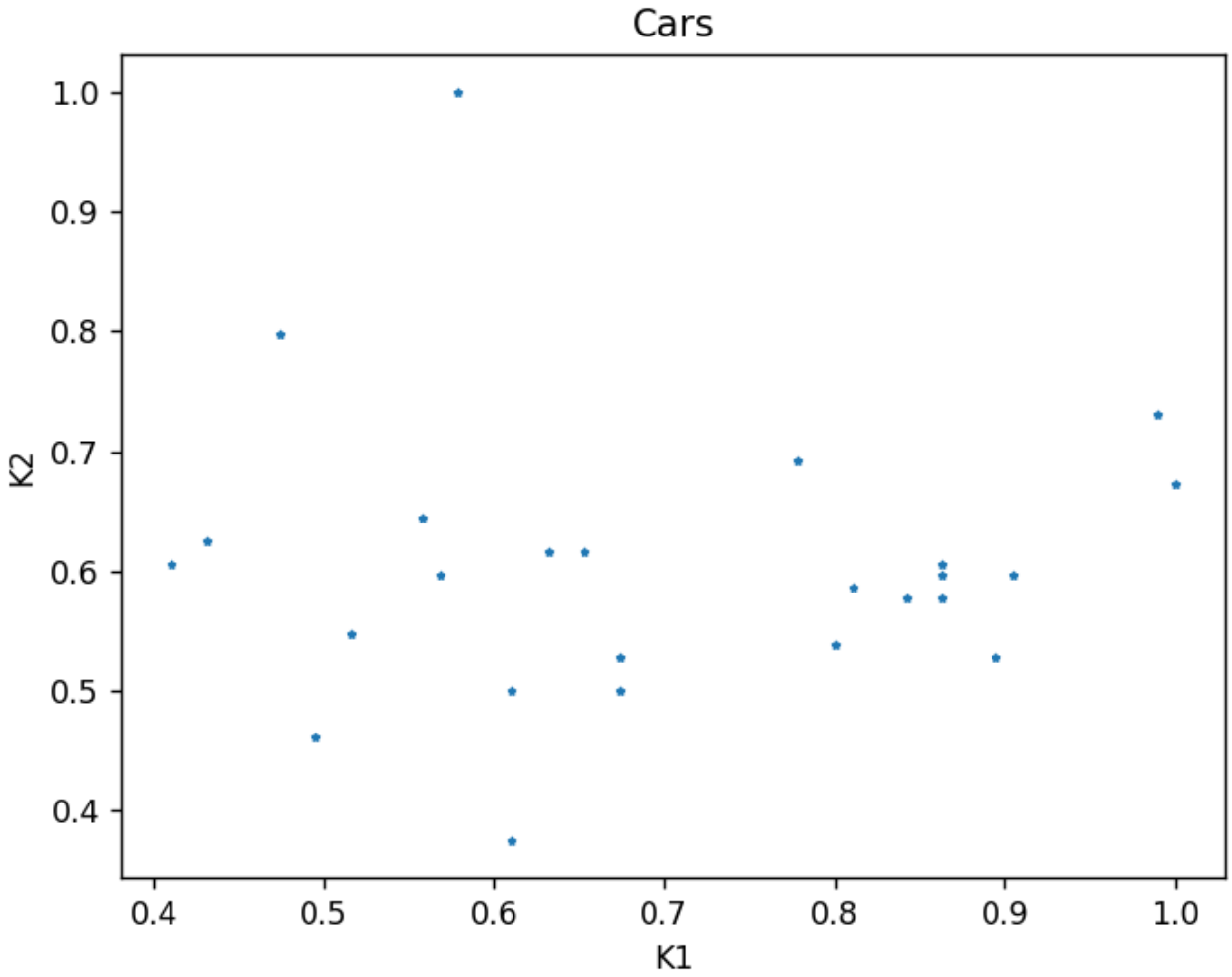


Рис. 6. Усі автомобілі на площині

На графіку "Візуалізація території кластерів" (див. рис. 7) відображені центри кластерів та території, які ці кластери охоплюють. Цей графік допомагає адміністратору отримати уявлення про розміщення кластерів у просторі ознак та

їх географічний розподіл.

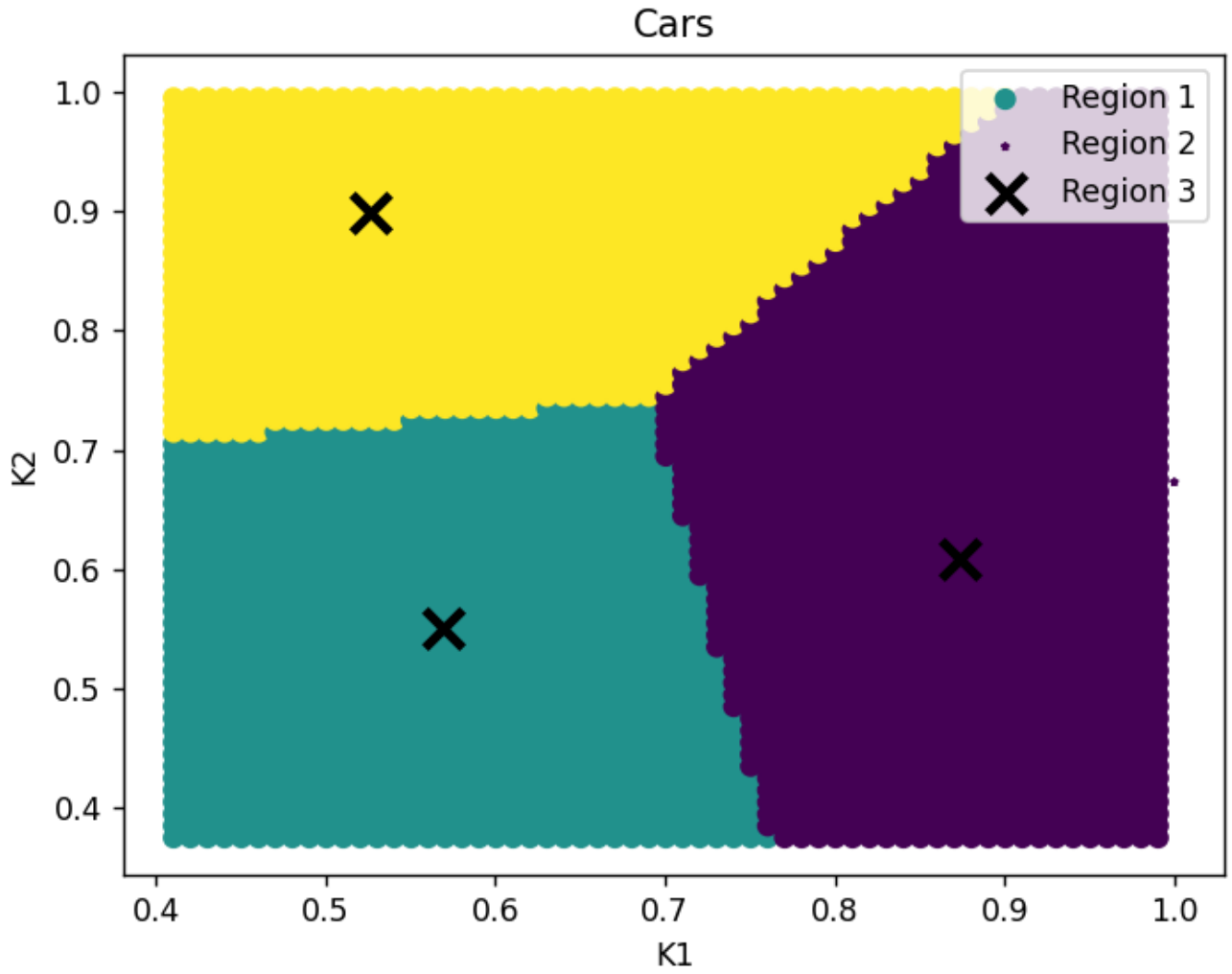


Рис. 7. Візуалізація території кластерів.

Графік "Візуалізація центрів кластерів та машин, які до них належать" (див. рис. 8) відображає центри кластерів та машини, що належать до кожного кластера. Цей графік допомагає адміністратору визначити структуру кластерів, виявити схожість та відмінності між кластерами, а також зрозуміти, які автомобілі входять до кожного кластера.

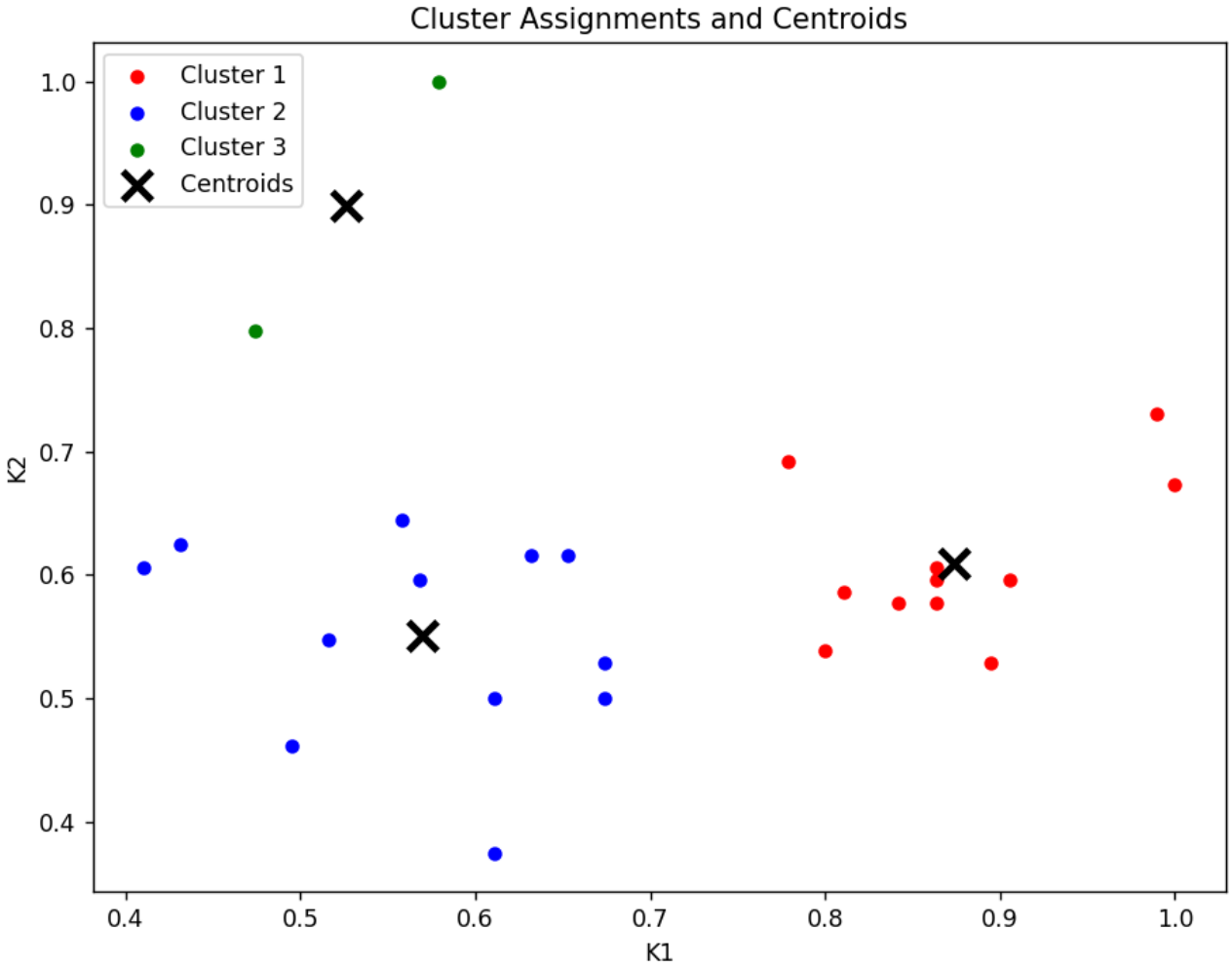


Рис. 8. Візуалізація центрів кластерів та машин, які до них належать.

Ці графіки візуалізують інформацію про кластери та розподіл даних, що допомагає адміністратору зробити об'єктивний аналіз результатів кластеризації та прийняти відповідні рішення на основі цих даних.

3.3 Реалізація роботи з базою даних через графічний інтерфейс

Головна мета цієї задачі зробити можливість адміністратору додавати, редагувати, проглянути всю базу даних та видаляти автомобілі через інтерфейс сайту. Для цього використана бібліотека Flask для розробки сайту, бібліотека mysql.connector для роботи с базою даних MySQL.

mysql.connector є бібліотекою для з'єднання з СУБД MySQL з використанням Python. Вона надає зручні методи для взаємодії з базою даних, виконання запитів та отримання результатів. Завдяки mysql.connector можна легко під'єднатисядо бази даних MySQL і виконувати різноманітні операції, такі як додавання, редагування та видалення даних.

Flask є одним з найпопулярніших фреймворків веб-розробки для мови програмування Python. Він надає простий та елегантний спосіб створення веб-додатків та API. Flask відомий своєю лаконічністю, гнучкістю та легкістю в освоєнні, що робить його привабливим для початківців та досвідчених розробників[7].

Основні особливості та переваги Flask:

1. Мінімалістичний підхід: Flask пропонує мінімальний набір необхідних інструментів для веб-розробки. Він не нав'язує жорстких правил та структури проєкту, що дозволяє розробникам використовувати свої підходи та бібліотеки за власним бажанням.

2. Маршрутизація: Flask надає простий механізм маршрутизації, який дозволяє визначати URL-шаблони та пов'язані з ними функції-обробники. Це дозволяє веб-додатку відповідати на різні запити та виконувати необхідні дії.

3. Вбудований сервер розробки: Flask має вбудований сервер розробки, що дозволяє швидко запустити та перевірити веб-додаток без необхідності налаштування окремого веб-сервера. Це робить процес розробки простішим та швидшим.

4. Шаблонізація: Flask підтримує використання шаблонів для генерації динамічного контенту. За допомогою шаблонів, розробники можуть створювати структуровані та привабливі сторінки з використанням HTML, CSS та JavaScript.

Далі було написано декілька HTML сторінок, щоб забезпечити красиве оформлення та дружній інтерфейс користувача. Для досягнення цього використано фреймворк Bootstrap.

Bootstrap - це популярний фреймворк для розробки веб-інтерфейсів, який забезпечує широкий спектр готових компонентів, стилів і інструментів для швидкого створення естетичного та респонсорного дизайну. Завдяки використанню Bootstrap, можна швидко і просто створити привабливі сторінки зі спільними елементами дизайну, такими як навігаційне меню, кнопки, таблиці, форми та інші.

Використання Bootstrap разом з Flask, забезпечує привабливий та функціональний зовнішній вигляд моїх сторінок та інтерфейсу користувача. Це дозволило адміністратору легко взаємодіяти з базою даних автомобілів, додавати нові записи, редагувати чинний та видаляти їх, все це через зручний та інтуїтивно зрозумілий інтерфейс. Завдяки використанню Flask, можна створити маршрути та функції, які відповідають на запити користувача та виконують відповідні дії з базою даних.

За допомогою Flask, було налаштовано сервер, який слухає певний порт та приймає HTTP-запити від користувачів. При отриманні запиту, Flask виконує відповідну функцію, яка обробляє запит та повертає відповідь у вигляді HTML-сторінки. Це дозволяє відображати сторінки з даними автомобілів, формами для додавання або редагування даних, а також повідомлення про успішне виконання операцій.

Перша частина веб-додатку відповідає за перегляд бази даних. Для цього виконано запит до таблиць car, car_body, engine та schassis, отримано всі стовпці цих таблиць і надіслано їх на HTML-сторінку для відображення (див. рис. 9).

ID	Brand	Model	Price	Year of Issue	Car Weight	Trunk Volume	Interior Volume	Number of Airbags	Car Base	Costs of Auto Service	Periodicity of Auto Service	Braking Distance	Track	Energy Consumption	Fuel Consumption (100 km)	Minimum Fuel Consumption (100 km)	Maximum Speed	Acceleration Time	Fuel Density
1	Corolla	Toyota	738984	2022	1310	0.47	2.83	7	2.7	10952	100000	36.0	1.55	1.0	6.3	5.5	180.0	9.5	750
2	Civic	Honda	812883	2022	1297	0.519	2.766	10	2.7	12778	90000	36.0	1.5	1.0	7.1	6.0	200.0	8.2	760
3	C-Class Seda	Mercedes-Benz	1537087	2022	1565	0.455	1.21	9	2.84	27382	85000	34.3	1.57	1.0	7.4	4.8	250.0	6.2	750
4	Golf	Volkswage	857037	2022	1284	0.38	1.27	6	2.64	21905	95000	30.3	1.54	1.0	5.9	3.9	250.0	7.8	750
5	Mazda3	Mazda	886781	2022	1424	0.444	2.805	7	2.73	14603	100000	34.0	1.6	1.0	7.3	6.2	195.0	8.1	750
6	Elantra	Hyundai	775933	2022	1367	0.408	2.888	6	2.72	12778	850000	36.0	1.5	1.0	7.1	6.0	200.0	8.3	760
7	Forte	Kia	738984	2022	1305	0.433	3.035	6	2.7	12778	60000	34.0	1.6	1.0	7.4	6.3	200.0	8.6	750
8	S-Class S500	Mercedes-Benz	4249161	2022	2070	0.51	1.617	10	3.165	54764	75000	40.0	1.62	1.0	9.5	6.5	250.0	5.0	750

Рис. 9. Вивід таблиці з даними на HTML-сторінку

Друга частина веб-додатку відповідає за створення нового запису автомобіля в базі даних. Для цього було створено форму вводу на веб-сторінці, яка дозволяє адміністратору ввести необхідні дані про автомобіль, такі як марка, модель, рік випуску та інші характеристики.

У HTML-файлі, використовується тег <form> для створення форми, атрибут method="POST" вказує, що дані з форми будуть передані на сервер за допомогою методу POST.

При натисканні на кнопку "Відправити", дані з форми будуть надіслані на сервер за адресою / submit. У Flask додатку, визначається маршрут / submit та метод POST, щоб обробити дані з форми:

У функції submit(), використовується об'єкт request для отримання значень полів форми. Залежно від потреби, можна додати додаткові поля та обробити їх значення у функції submit (). Далі можна виконати необхідні дії, а саме зберегти дані в базі даних.

Так була виконана форма для заповнення(див. рис. 10)

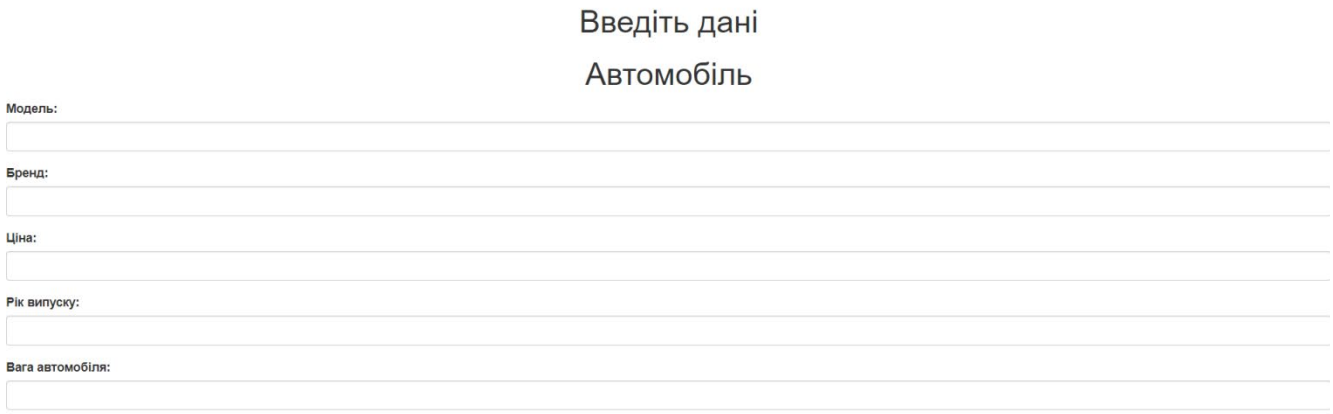


Рис. 10. Частина форми для заповнення

Третя частина це редагування базі даних. Якщо адміністратору треба змінити дані про якийсь автомобіль, то він повинен просто ввести у форму для заповнення ID автомобіля який хоче змінити та нові дані про автомобіль. Принцип дії цього алгоритму такий же як і у минулому пункті(див. рис. 11).

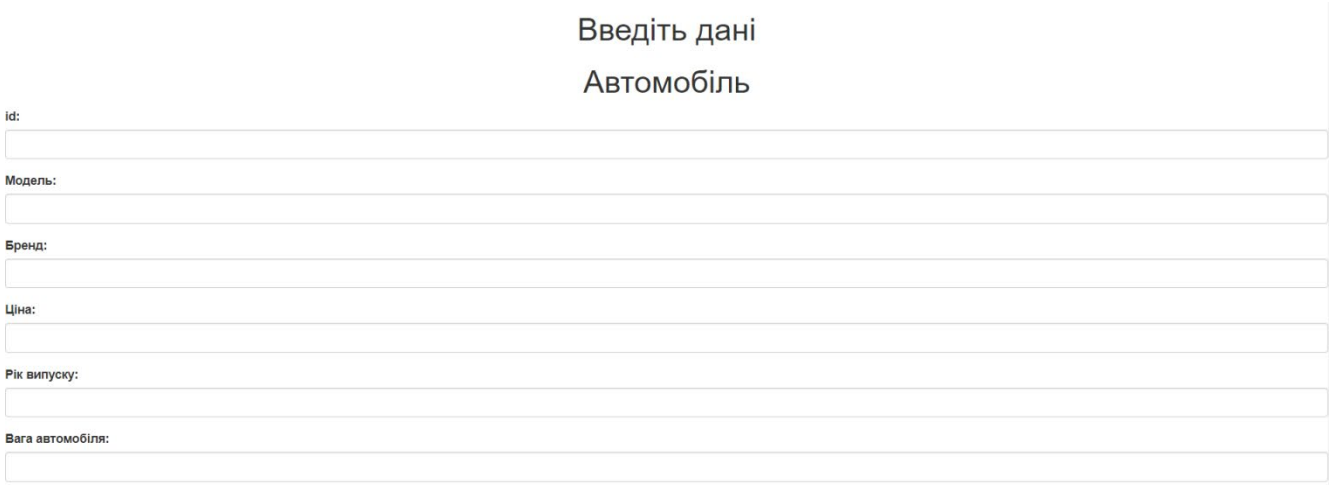


Рис. 11. Частина форми для редагування

Остання частина — це видалення автомобіля з бази даних, потрібно створити HTML-сторінку з формою, яка дозволить адміністратору ввести ID автомобіля, який потрібно видалити. Після натискання кнопки "Видалити" дані форми будуть відправлені на сервер для подальшої обробки.

3.4 Вивід графіків та кластеризація в веб-додадку.

Вище був описаний алгоритм кластеризації та роботи нейронних мереж, але це дуже ресурсозатратний алгоритм, тому не можна його проводити після кожного нового доданого автомобіля. Для розв'язання цієї проблеми, було додано окрему сторінку, на якій зможемо здійснити кластеризацію після накопичення декількох доданих автомобілів. Для цього адміністратор просто натисне кнопку, яка проведе кластеризацію після стількох накопичених машин, скільки накопичилось у ході додавання нових автомобілів.

Для забезпечення відображення всіх графіків разом і аналізу їх адміністратором, було додано веб-сторінку, на якій розмістив усі графіки, які були згадані вище. Для виведення графіків на сторінку використовувався підхід з використанням бібліотеки base64. Ця бібліотека дозволяє закодувати графік у

формат base64, після чого отримане закодування можна передати на HTML-сторінку.

Таким чином, замість використання посилань на зовнішні файли графіків, графіки були закодовані безпосередньо в рядок base64. Цей рядок може бути вставлений у HTML-код сторінки за допомогою тегу <img>. При завантаженні сторінки браузер декодує рядок base64 і відображає графіки безпосередньо на сторінці.

Такий підхід дозволяє забезпечити відображення графіків без необхідності зберігати їх як окремі зовнішні файли та зменшує кількість запитів до сервера для завантаження графіків. Адміністратор може легко переглядати всі графіки на одній сторінці та проводити їх аналіз без зайвих кліків та переходів між сторінками.

3.5 Концепція повної програми по шаблону MVC

Після усіх попередніх етапів розробки програми, таких як проєктування бази даних, розробка бізнес-логіки, інтеграція з базою даних і взаємодія з адміністратором, залишилося тільки зібрати всі модулі разом, щоб отримати повноцінний функціональний додаток. Для цього було використано шаблон MVC.

За інтернет-ресурсом [8] MVC (Model-View-Controller) - це архітектурний шаблон, що використовується для розробки програмного забезпечення. Він розділяє компоненти програми на три основних ролі: Модель (Model), Вид (View) і Контролер (Controller). Кожна з цих ролей виконує свої функції та має свою відповідальність у системі.

Основні концепції MVC такі:

Модель (Model): Модель представляє дані та бізнес-логіку програми. Вона відповідає за доступ до даних, їх обробку та валідацію. Модель може включати

різні класи або об'єкти, які представляють дані, базу даних, API-запити, алгоритми обчислень та інше.

Вид (View): Вид відповідає за відображення даних моделі користувачу або взаємодію з ним. Він може бути графічним інтерфейсом, веб-сторінкою, шаблоном HTML або будь-яким іншим способом представлення даних користувачу. Відображення зазвичай залежить від даних, що надаються моделлю.

Контролер (Controller): Контролер обробляє взаємодію користувача з системою. Він приймає вхідні дані від користувача, взаємодіє з моделлю для отримання або збереження даних та забезпечує оновлення відображення відповідно до змін у моделі. Контролер може містити логіку обробки подій, маршрутизації запитів та керування потоком даних. Він взаємодіє з Видом та Моделлю, передаючи дані між ними та керуючи виконанням операцій.

Основна ідея MVC полягає у відокремленні логіки програми, даних та їх відображення для полегшення розробки, зміни та тестування коду. Цей шаблон дозволяє зберігати код чистим, організованим та зручним для розуміння.

У програмі:

1. Вид (View) відповідає за відображення HTML-сторінок, на яких розміщені форми для роботи з базою даних, такі як додавання автомобіля, редагування, видалення, або сторінки з графіками та сторінка, що проводить кластеризацію. Вид взаємодіє з Контролером для отримання та передачі даних.
2. Контролер (Controller) включає всі методи, які були написані на Flask. Ці методи отримують дані з веб-сторінок, обробляють їх, виконують валідацію та інші операції, а потім надсилають оброблені дані до Моделі. Контролер відповідає за керування потоком даних та взаємодію з Видом і Моделлю.
3. Модель (Model) відповідає за роботу з базою даних. Вона забезпечує виведення даних з таблиць для передачі на веб-сторінку, додавання автомобіля, редагування, видалення тощо. Модель також обчислює всі коефіцієнти якості автомобіля та записує їх до бази даних. Крім того, Модель відповідає за малювання графіків, які використовуються адміністраторами для аналізу та роботи з нейронною мережею для обробки функції кластеризації.

Модель виконує роботу з базою даних, забезпечуючи доступ до неї та виконуючи операції з даними відповідно до бізнес-логіки програми.

Організація програми за шаблоном MVC дозволяє досягти розділення відповідальностей між компонентами та забезпечити гнучкість і легкість розширення. Зміни в одному компоненті не впливають безпосередньо на інші, що спрощує тестування, модифікацію та підтримку коду.

При використанні шаблону MVC у програмі, є можливість легко додавати нові функціональності, модифікувати чинний та розширювати функціональні можливості додатка без необхідності редагувати всю структуру. Кожен компонент виконує свою конкретну роль і може бути розроблений та тестований незалежно від інших компонентів.

Загальна схема роботи програми з використанням шаблону MVC може бути такою:

Користувач взаємодіє з Видом, заповнюючи форми на HTML-сторінках. Наприклад, користувач може ввести дані про новий автомобіль у форму на сторінці додавання автомобіля.

Введені користувачем дані передаються до Контролера, який отримує ці дані та виконує валідацію, перевірку на коректність та інші операції, необхідні для обробки даних перед подальшою роботою з ними.

Після обробки Контролер взаємодіє з Моделлю, передаючи їй необхідні дані та інструкції. Модель виконує операції з базою даних, такі як додавання, редагування або видалення даних, а також проводить розрахунки коефіцієнтів якості автомобіля.

Після завершення операцій з базою даних Модель повертає результати Контролеру. Контролер вирішує, які дані потрібно відобразити користувачу та які дії виконати далі.

Остаточно, Контролер передає дані до Виду, який відповідає за відображення даних користувачу. Вид використовує отримані дані та шаблони HTML для створення веб-сторінок з необхідною інформацією, включаючи форми, графіки та інші елементи інтерфейсу.

Ця циклічна взаємодія між компонентами Модель-Вид-Контролер дозволяє програмі працювати ефективно, забезпечуючи чітку структуру та відокремлення відповідальностей. Крім того, ви можете легко розширити або модифікувати функціональність програми, працюючи з окремими компонентами без необхідності внесення змін до інших частин коду.

3.6 Розробка повної програми по шаблону MVC

Цю модель-вид-контролер можна найкраще описати візуально за допомогою аналізу надійності в стереотипній нотації UML [9](див. рис. 12).

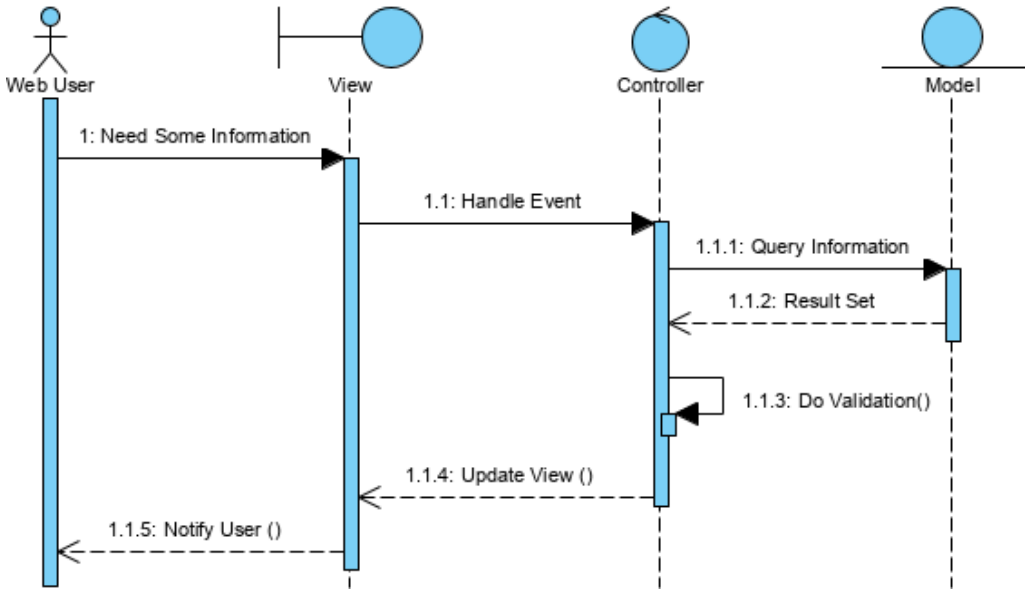


Рис. 12. UMl нотація для шаблону MVC

Ось проста і гіпотетична діаграма послідовності для MVC. На діаграмі вище зазначено те, що веб-користувач ініціює запит і генерує подію, яка обробляється контролером для отримання необхідної інформації з моделі, перевірки інформації та передачі набору результатів назад у представлення.

По цій схемі було реалізовано MVC патерн. Для використання функції добавлення нового автомобілю UML діаграму можна представити наступним чином.(див. рис. 13)

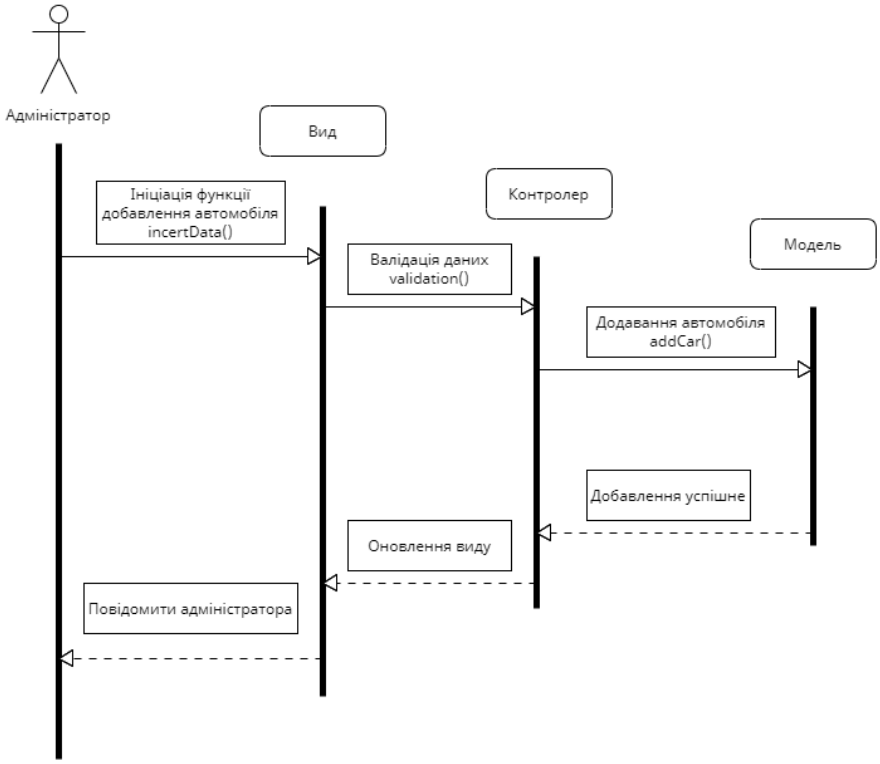


Рис. 13. UMl діаграма для функції додавання

На цій діаграмі можна побачити процес додавання нового автомобіля, який ініціюється адміністратором. Детальний опис кожного етапу процесу:

1. Адміністратор відкриває сторінку Виду, де відображається форма для додавання нового автомобіля.
2. Адміністратор заповнює форму з необхідними даними про автомобіль і натискає кнопку "Додати".
3. Контролер отримує дані, які були введені адміністратором і проводить їх обробку. Він перевіряє дані на наявність можливих помилок або некоректних значень.
4. Якщо дані відповідають вимогам, контролер передає їх до Моделі для додавання автомобіля до бази даних.
5. Модель отримує дані від контролера і виконує операцію додавання автомобіля до бази даних. Після цього, модель надсилає повідомлення контролеру про успішне додавання.

6. Контролер оновлює Вид, щоб відобразити повідомлення про успішне додавання автомобіля.

7. Адміністратор бачить оновлений Вид і отримує повідомлення про успішне додавання автомобіля.

Таким чином, весь процес включає взаємодію між Видом, Контролером та Моделлю, де Вид відображає форму, Контролер обробляє дані та координує взаємодію, а Модель здійснює зміни в базі даних та повертає статус операції.

Основну роботу з додавання автомобіля робить функція add() вона виглядає так (див. рис. 14)

```
query = """
    INSERT INTO Car (id_car, model, brand, price, year_of_issue, car_weight)
    VALUES (%s, %s, %s, %s, %s, %s)
"""
values = (id, model, brand, price, year_of_issue, car_weight)
cursor.execute(query, values)

query = """
    INSERT INTO Car_body(car_fk, trunk_volume_m3, interior_volume_m3, number_of_airbags, car_base_m)
    VALUES (%s, %s, %s, %s, %s)
"""
values = (id, trunk_volume_m3, interior_volume_m3, number_of_airbags, car_base_m)
cursor.execute(query, values)

query = """
    INSERT INTO Chassis(car_fk, costs_auto_service, periodicity_auto_service, braking_distance_m, track_m)
    VALUES (%s, %s, %s, %s, %s)
"""
values = (id, costs_auto_service, periodicity_auto_service, braking_distance_m, track_m)
cursor.execute(query, values)

query = """
    INSERT INTO Engine(car_fk, energy_consumption, fuel_consumption_100, min_fuel_consumption_100,
     max_speed, acceleration_time, fuel_density)
    VALUES (%s, %s, %s, %s, %s, %s, %s)
"""
values = (id, 0.06, fuel_consumption_100, min_fuel_consumption_100, max_speed, acceleration_time, fuel_density)
cursor.execute(query, values)
```

Рис. 14. Функція add()

У функції приймаються дані які висилає адміністратор та додаються в базу даних.

Таким же чином працює функція видалення та функція редагування(див. рис. 15 та 16).

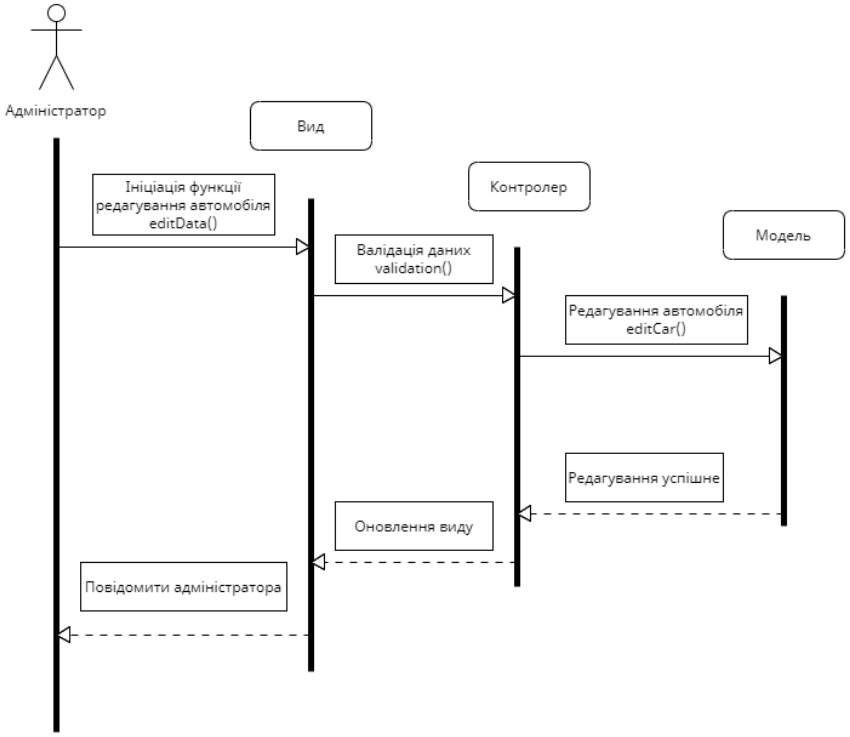


Рис. 15. UMl діаграма для функції редагування

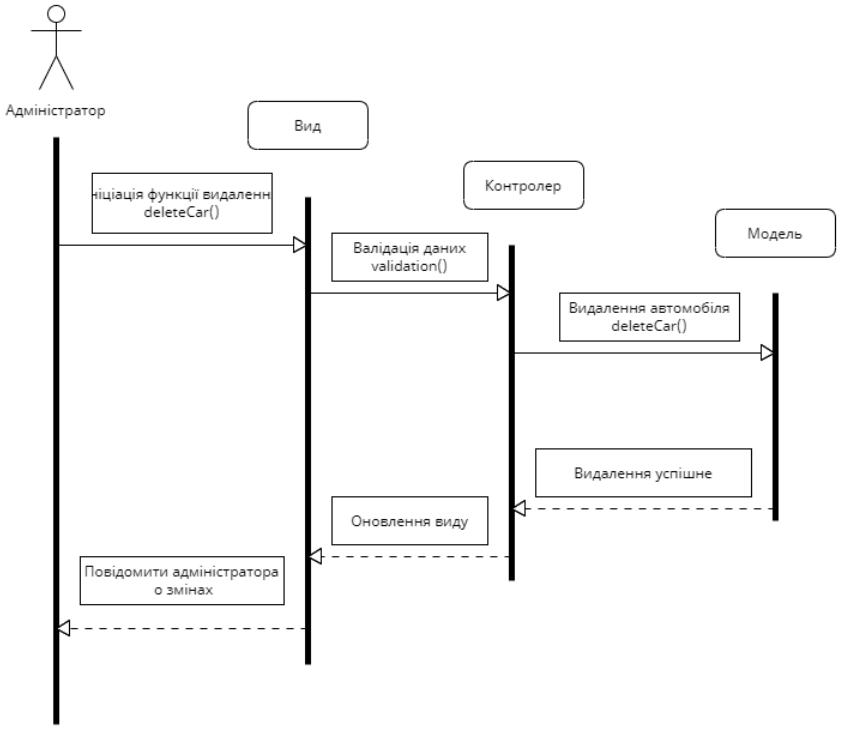


Рис. 16. UMl діаграма для функції видалення

Функція editCar() працює так само як і функція додавання, але замість функції INCERT використовується UPDATE для взаємодії с базою даних. Натомість функція deleteCar() працює по іншому вона видаляє автомобіль з усіх таблиць по ID(див рис. 17)

```python
@app.route('/delete', methods=['GET', 'POST'])
def delete():
    if request.method == 'POST':
        id = request.form['id']
        tables = ['Car_body', 'Chassis', 'Engine', 'K_synthetic', 'Kcumul', 'Clusters', 'Car']

        for table in tables:
            query = f"DELETE FROM {table} WHERE car_fk = %s"
            cursor.execute(query, (id,))

        cnx.commit()
        return render_template('delete.html')
    else:
        return render_template('delete.html')
```

Рис. 17. Функція delete()

Наступна діаграма показує як діє програма при будуванні таблиці(див. рис. 18)

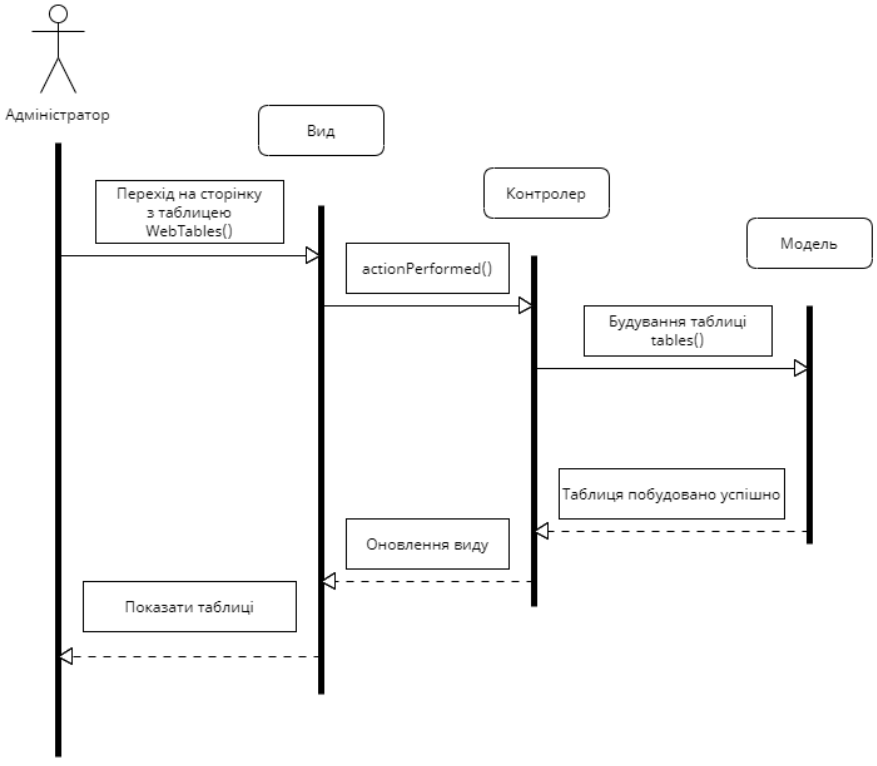


Рис. 16. UMl діаграма для функції перегляду таблиці

Після того, як адміністратор виконує перехід на сторінку з таблицею, відбувається взаємодія між Видом, Контролером та Моделлю для оновлення та відображення таблиць на веб-сторінці. Ось як цей процес можна описати:

1. Вид (View) генерує запит до Контролера для отримання таблиці.
2. Контролер (Controller) отримує запит від Виду та передає його до Моделі.
3. Модель (Model) виконує запит до бази даних для отримання таблиці.
4. Модель отримує дані з бази даних і формує таблиці для передачі назад до Контролера.
5. Контролер отримує таблицю від Моделі та передає їх назад до Виду.
6. Вид отримує таблиці від Контролера та оновлює веб-сторінку, відображаючи таблиці для адміністратора.

Таким чином, після виконання всіх кроків адміністратор може бачити оновлені таблиці на веб-сторінці.

Функція tables() працює наступним чином(див. рис. 17)

```
@app.route('/tables')
def tables():
    cursor.execute("""
        SELECT C.*, CB.trunk_volume_m3, CB.interior_volume_m3, CB.number_of_airbags, CB.car_base_m,
               CH.costs_auto_service, CH.periodicity_auto_service, CH.braking_distance_m, CH.track_m,
               E.energy_consumption, E.fuel_consumption_100, E.min_fuel_consumption_100,
               E.max_speed, E.acceleration_time, E.fuel_density
        FROM Car AS C
        JOIN Car_body AS CB ON C.id_car = CB.car_fk
        JOIN Chassis AS CH ON C.id_car = CH.car_fk
        JOIN Engine AS E ON C.id_car = E.car_fk;
    """)

    cars = cursor.fetchall()

    return render_template('tables.html', rows=cars)
```

Рис. 17. Функція tables()

Функція tables() просто дістає всі дані с таблиць бази даних та виводить їх.

Так само працює і функція яка виводить графіки тільки вона не бере дані с бази даних, а вираховує графіки та виводить їх.

Останньою функцією програми для якої використовується MVC шаблон – це функція кластеризації. Для неї діє такий принцип(див. рис. 18)

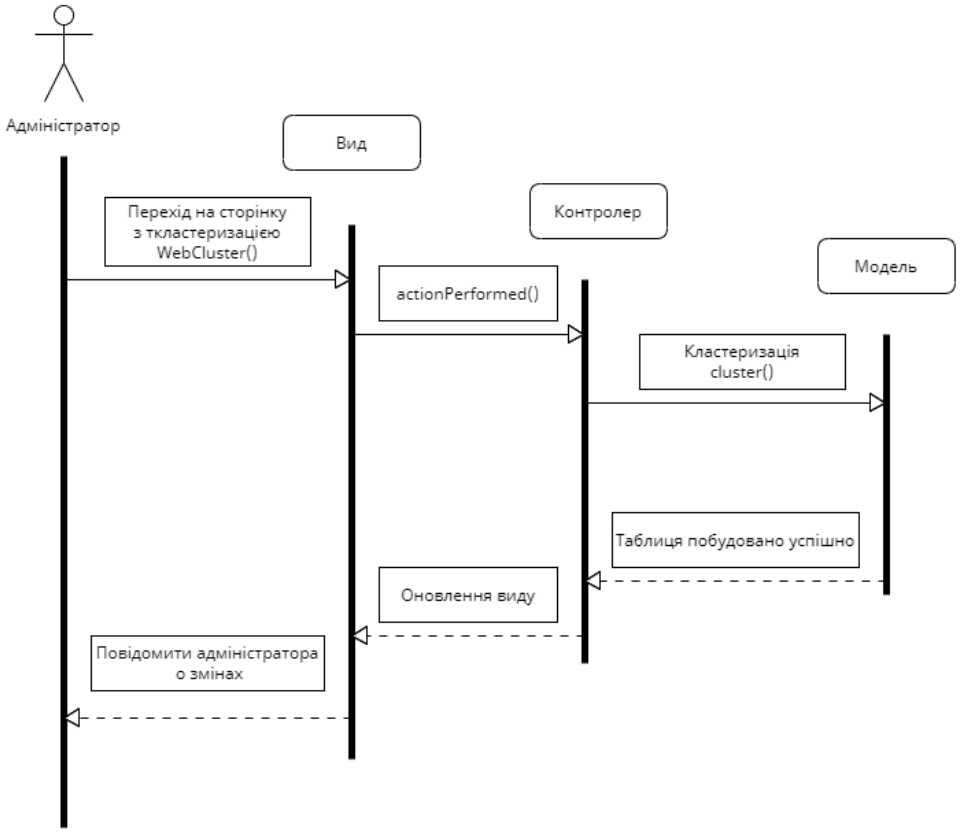


Рис. 18. UMl діаграма для функції проведення кластеризації

Після того, як адміністратор переходить на сторінку з кластеризацією, він ініціює процес проведення кластеризації. Це викликає взаємодію між Видом, Контролером і Моделлю, яка приводить до оновлення та відображення результатів кластеризації на веб-сторінці. Ось як можна описати цей процес:

1. Адміністратор переходить на сторінку з кластеризацією і запускає процес.
2. Вид генерує запит до Контролера для проведення кластеризації.
3. Контролер отримує запит від Виду і передає його до Моделі.
4. Модель виконує кластеризацію на основі отриманих даних.
5. Модель передає результати кластеризації через Контролер до Виду.

6. Вид отримує результати від Контролера та оновлює веб-сторінку, відображаючи їх для адміністратора.

Таким чином, адміністратор може бачити результати проведеної кластеризації на веб-сторінці після завершення процесу.

3.7 Приклад використання програми

Останньою частиною написання програми це буде тестування та перевірка роботи функцій програми.

Додамо нову машину з такими характеристиками:

1. Модель: TestModel
2. Марка: TestBrand
3. Ціна: 500000
4. Рік випуску: 2014
5. Вага автомобіля: 1500
6. об'єм багажника: 1
7. об'єм салону: 2
8. Кількість подушок безпеки: 6
9. Довжина шасі автомобіля: 3
10. Ціна ТО: 12000
11. Гарантійний пробіг: 100000
12. Гальмівний шлях: 40
13. Довжина бази автомобіля: 2
14. Витрата палива на 100 км: 8
15. Мінімальна витрата палива на 100 км: 6
16. Максимальна швидкість: 200
17. Розгін до 100 км/год: 7
18. Густина палива: 750

Тестовий автомобіль успішно додався до бази даних(див. рис. 19)

22	X3 xDrive30i	BMW	1588816	2022	1945	1.011	2.654	6	2.864	36509	80000	36.0	1.676	1.0	9.1	7.6	240.0	6.3	760
23	C-Class C 300 4MATIC	Mercedes-Benz	1614681	2022	1945	0.357	1.106	9	2.84	36509	100000	35.0	1.567	1.0	8.3	6.7	250.0	5.7	750
24	RAV4 XLE	Toyota	1063953	2022	1565	1.064	2.078	8	2.69	10587	90000	39.0	1.54	1.0	8.3	7.0	200.0	8.2	760
25	Mustang EcoBoost Fastback	Ford	1005203	2022	1615	0.382	2.193	8	2.72	36509	90000	37.0	1.59	1.0	8.0	6.4	232.0	5.5	760
26	TestModel	TestBrand	500000	2014	1500	1.0	2.0	6	3.0	12000	100000	40.0	2.0	0.06	8.0	6.0	200.0	7.0	750

Рис. 19. Доданий автомобіль з Id 26

Перевіримо функцію редагування і змінимо назву тестового автомобіля(див. рис. 20).

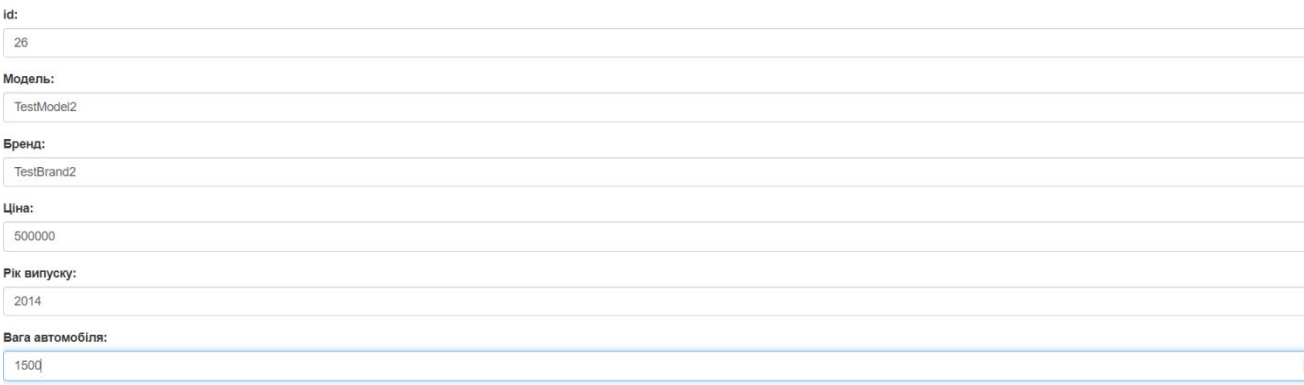


Рис. 20. Заповнення форми редагування.

Відправимо зміни на сервер та перевіримо базу даних. Та побачимо що зміни відбулись(див. рис. 21).

22	X3 xDrive30i	BMW	1588816	2022	1945	1.011	2.654	6	2.864	36509	80000	36.0	1.676	1.0	9.1	7.6	240.0	6.3	760
23	C-Class C 300 4MATIC	Mercedes-Benz	1614681	2022	1945	0.357	1.106	9	2.84	36509	100000	35.0	1.567	1.0	8.3	6.7	250.0	5.7	750
24	RAV4 XLE	Toyota	1063953	2022	1565	1.064	2.078	8	2.69	10587	90000	39.0	1.54	1.0	8.3	7.0	200.0	8.2	760
25	Mustang EcoBoost Fastback	Ford	1005203	2022	1615	0.382	2.193	8	2.72	36509	90000	37.0	1.59	1.0	8.0	6.4	232.0	5.5	760
26	TestModel2	TestBrand2	500000	2014	1500	1.0	2.0	6	3.0	12000	100000	40.0	2.0	0.06	8.0	6.0	200.0	7.0	750

Рис. 21. Відредагований автомобіль.

Тепер перевіримо функцію видалення автомобіля для цього просто вводимо на веб-сторінці id автомобіля котрий хочемо видалити(див. рис. 22).

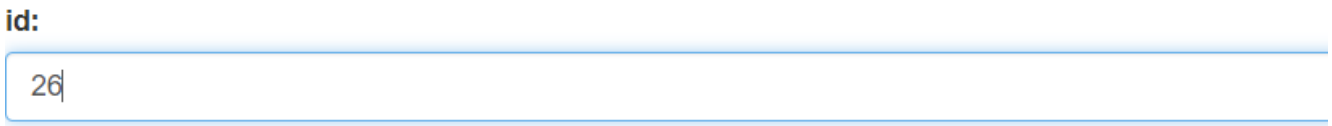


Рис. 22. Заповнення форми для видалення автомобіля

Знову робимо перевірку бази даних та перевіряємо видала програма автомобіль чи ні(див. рис. 23).

22	X3 xDrive30i	BMW	1588816	2022	1945	1.011	2.654	6	2.864	36509	80000	36.0	1.676	1.0	9.1	7.6	240.0	6.3	760
23	C-Class C 300 4MATIC	Mercedes-Benz	1614681	2022	1945	0.357	1.106	9	2.84	36509	100000	35.0	1.567	1.0	8.3	6.7	250.0	5.7	750
24	RAV4 XLE	Toyota	1063953	2022	1565	1.064	2.078	8	2.69	10587	90000	39.0	1.54	1.0	8.3	7.0	200.0	8.2	760
25	Mustang EcoBoost Fastback	Ford	1005203	2022	1615	0.382	2.193	8	2.72	36509	90000	37.0	1.59	1.0	8.0	6.4	232.0	5.5	760

Рис. 23. Перевірка видалення автомобіля.

І останній аспект перевірки програми це процес кластеризації та показу графіків(див. рис. 24).

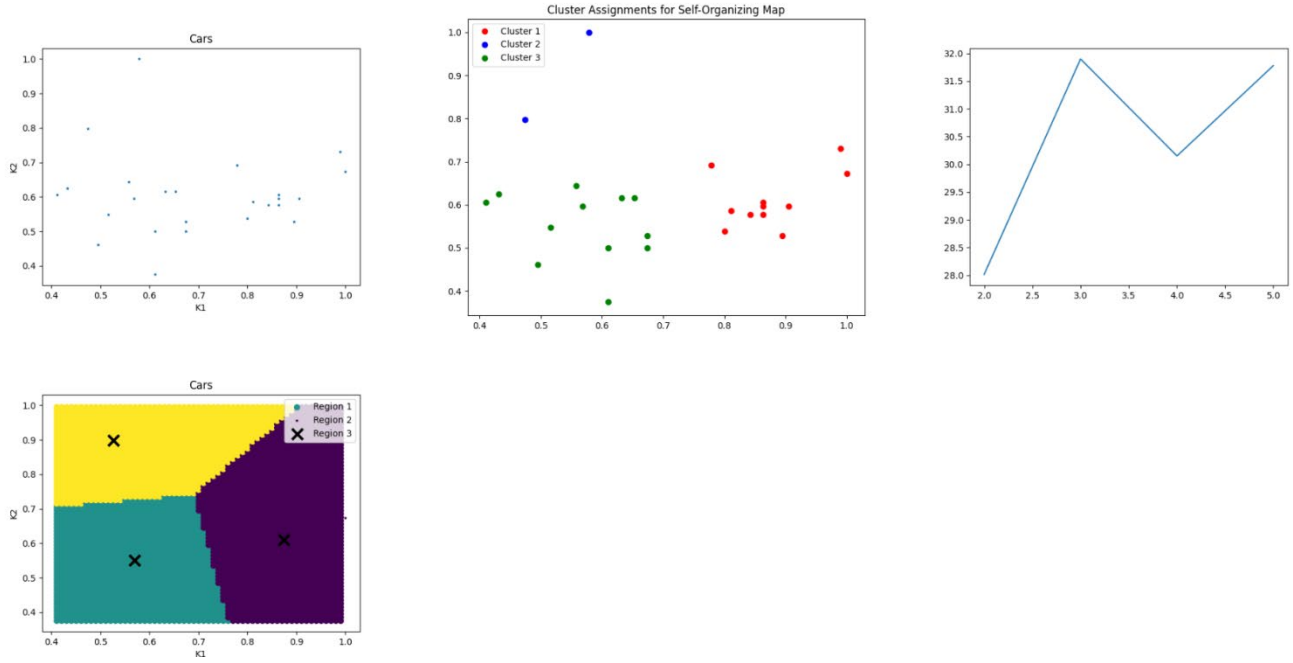


Рис. 24. Перевірка виводу графіків на сайті.

Як бачимо, всі графіки вивело коректно. І був зроблений висновок що всі модулі програми виконані коректно.

ВИСНОВКИ

У даній дипломній роботі була поставлена мета створення програмного модуля, спрямованого на оптимізацію вибору автомобільної продукції для компаній, які продають автомобілі. Головна мета полягала у поліпшенні швидкості та точності процесу пошуку автомобілів, які відповідають вимогам водіїв. Для досягнення цієї мети були сформульовані та реалізовані основні завдання програмного модуля.

Перше завдання передбачало вирахування коефіцієнтів автомобіля, таких як комфорт, екологічність, технічні рішення та інші, для кожного автомобіля в базі даних. Це дозволяє отримати об'єктивну оцінку якості автомобілів та визначити їхню придатність для певних вимог водіїв.

Друге завдання полягало у розробці модуля штучного інтелекту, який здатний кластеризувати базу даних за допомогою вирахуваних коефіцієнтів. Це дозволяє системі автоматично групувати автомобілі зі схожими характеристиками та спрощує процес пошуку відповідних автомобілів для покупців.

Третє завдання передбачало реалізацію роботи з базою даних через графічний інтерфейс сайту. Цей інтерфейс дозволяє адміністратору зручно взаємодіяти з базою даних, встановлювати параметри оптимізації та налаштовувати роботу програмного модуля.

Четверте завдання передбачало знаходження оптимальної кількості кластерів за допомогою алгоритмів кластеризації. Це важливий етап, оскільки правильне визначення кількості кластерів допомагає досягти оптимального розподілу автомобілів та підвищити ефективність процесу вибору.

П'яте завдання полягало у виведенні різноманітних графіків, які відображали різні аспекти роботи програмного модуля. Графіки надавали адміністратору зручний спосіб аналізу та визначення результатів роботи програми. Наприклад, розподіл автомобілів за кластерами, залежність

коефіцієнтів якості від певних параметрів автомобілів та зміни кластеризації при зміні параметрів оптимізації.

Отже, на основі проведеного дослідження та реалізації програмного модуля можна зробити висновок, що він успішно виконує свої завдання. Створений програмний модуль дозволяє оптимізувати процес вибору автомобільної продукції для компаній, що продають автомобілі, шляхом швидкого та точного пошуку автомобілів, що відповідають вимогам водіїв.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Бажинова Т.О. ТЕОРЕТИЧНЕ ОБҐРУНТУВАННЯ ОЦІНКИ ЯКОСТІ ЛЕГКОВИХ АВТОМОБІЛІВ / Т.О. Бажинова – К. ХНАДУ, 2016. – с. 95 - 99
2. Бажинова Т.О. ОЦІНКА ЛЕГКОВИХ АВТОМОБІЛІВ ЗА РАХУНОК ВИЗНАЧЕННЯ ПОКАЗНИКІВ ЯКОСТІ НА Етапі експлуатації/ Т.О. Бажинова – К. ХНАДУ, 2018. – 181 с.
3. A PRACTICAL INTRODUCTION TO FACTOR ANALYSIS: EXPLORATORY FACTOR ANALYSIS [Електронний ресурс]. URL: https://stats.oarc.ucla.edu/spss/seminars/introduction-to-factor-analysis/a-practical-introduction-to-factor-analysis/
4. What's the Difference? Relational vs Non-Relational Databases [Електронний ресурс]. URL: https://insightsoftware.com/blog/whats-the-difference-relational-vs-non-relational-databases/
5. Calinski-Harabasz Evaluation [Електронний ресурс]. URL: https://www.mathworks.com/help/stats/clustering.evaluation.calinskiharabaszevaluation.html
6. Self Organizing Maps [Електронний ресурс]. URL: https://medium.com/@abhinavr8/self-organizing-maps-ff5853a118d4
7. An introduction to the Flask Python web app framework [Електронний ресурс]. URL: https://opensource.com/article/18/4/flask
8. The Model View Controller Pattern – MVC Architecture and Frameworks Explained [Електронний ресурс]. URL: https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/
9. Jacobson I. Object-oriented Software Engineering: A Use Case Driven Approach/ I. Jacobson, 1992. 552 с. (Addison-Wesley)