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-42
спеціальність 122 «Комп'ютерні науки»
освітньо-професійна програма «Інформатика»

Сенів В. Р.

Керівник Фролов В. В.

Рецензент Бережна Н. І.

Харків – 2023 року

ВСТУП

1.1. Формулювання мети роботи, задач та обґрунтування актуальності теми.

Мета даного дослідження полягає в створенні програмного модуля, який допоможе користувачам здійснювати оптимальний вибір автомобільної продукції з урахуванням їхніх вподобань, потреб та обмежень.

Основними задачами розробки модуля є:

1. Створення інформаційної і математичної моделей автомобіля.
2. Аналіз існуючих бібліотек і алгоритмі. Вибір відповідного алгоритму.
3. Реалізація алгоритмів і інтерфейсу користувача в програмному продукті.

Актуальність цієї теми можна обґрунтувати кількома факторами:

1. Зростаючий обсяг інформації: Сучасний автомобільний ринок пропонує широкий вибір автомобільних моделей з різними технічними характеристиками, вартістю, ефективністю палива та іншими параметрами. Потенційним покупцям стає важко аналізувати та порівнювати всю цю інформацію самостійно. Розробка програмного модуля зі штучним інтелектом дозволить автоматизувати цей процес та забезпечити швидкий та об'єктивний вибір автомобіля.
2. Розвиток штучного інтелекту: Штучний інтелект є одним з найбільш активно розвиваються напрямків в сфері технологій. Його застосування в автомобільній промисловості може принести значні переваги, включаючи автоматизацію процесу вибору автомобіля, покращення взаємодії з користувачами та підвищення задоволення від покупки автомобілів.
3. Конкурентний ринок автомобільної продукції: Конкуренція в автомобільній промисловості постійно зростає. Виробники автомобілів постійно пропонують нові моделі з різними характеристиками та функціями. Розробка програмного модуля зі штучним інтелектом, який допомагає зробити оптимальний вибір, може допомогти користувачам знайти автомобіль, який найкраще відповідає їхнім потребам, та надати конкурентну перевагу виробникам, які пропонують такий модуль.

4. Екологічні аспекти: У світлі зростаючої уваги до екологічних питань та зменшення впливу на навколишнє середовище, багато покупців шукають автомобілі з низьким викидом шкідливих речовин та високою енергоефективністю. Розробка програмного модуля зі штучним інтелектом, який враховує екологічні параметри автомобілів, дозволить користувачам здійснювати свідомий вибір та сприятиме розвитку екологічно-орієнтованих технологій в автомобільній промисловості.

Таким чином, із зростаючою потребою в зручних і ефективних інструментах вибору транспортних засобів, розробка програмних модулів штучного інтелекту для оптимізації вибору транспортних засобів привертає увагу. Задовольняти сучасний попит споживачів на інноваційні рішення їхніх проблем. Цей програмний модуль полегшує життя потенційних покупців автомобілів, допомагаючи їм збирати та аналізувати інформацію про різні моделі.

1.2. Відомості про одержані результати та їх новизна.

Було проведено тестування системи рекомендацій для інтернет-магазину Stall, використовуючи сервіс uXprice. Під час тестування, новий варіант списку схожих товарів був порівняний з попереднім суб'єктивним підходом. Результати показали, що нова система рекомендацій, заснована на алгоритмах машинного навчання та штучного інтелекту, була більш ефективною.

Для оцінки ефективності, було виміряно дохід, отриманий від користувачів з оригінальним списком схожих товарів і з варіантом від uXprice. Загальний дохід з новим варіантом був на 12% вище, ніж при сеансах з оригінальним списком. Також, дохід за сеанс з варіантом від uXprice склав у середньому 3,87 грн, що на 22% вище, ніж оригінальний список.

На жаль Netflix не надає результати будь яких тестувань цих алгоритмів, оскільки ця інформація не є загальнодоступною.

1.3. Теоретичне та практичне значення результатів, можливі області використання, результати.

Теоретичне значення: Результати дослідження рекомендаційної системи для автомобілів мають значний теоретичний внесок у сферу автомобільної індустрії та використання штучного інтелекту. Вивчення ефективності системи рекомендацій дозволяє розуміти, як алгоритми можуть забезпечити покращення процесу вибору автомобілів для користувачів. Крім того, це дослідження підтверджує потенціал застосування методів кластеризації та машинного навчання для покращення рекомендаційних систем.

Практичне значення: Результати дослідження рекомендаційної системи для автомобілів мають практичне значення для автомобільної індустрії та споживачів. Впровадження ефективної системи рекомендацій дозволить автомобільним компаніям підвищити задоволеність клієнтів та збільшити коефіцієнт транзакції. Користувачам надається зручний інструмент для вибору автомобілів, що відповідають їхнім потребам та вподобанням. Результати також свідчать про потенціал використання штучного інтелекту та машинного навчання в індустрії автомобілів.

Результати:

- Результати дослідження підтверджують ефективність використання рекомендаційної системи для автомобілів у процесі вибору автомобілів користувачами.
- Впровадження системи рекомендацій може покращити задоволеність покупців, збільшити продажі автомобілів та підвищити лояльність клієнтів.
- Результати дослідження підтверджують потенціал використання методів штучного інтелекту та машинного навчання для рекомендаційних систем у сфері автомобілів.

ОСНОВНА ЧАСТИНА

2.1. Постановка задачі

Постановка задачі для розробки модуля включає наступні пункти:

1. Створення інформаційної і математичної моделей автомобіля: необхідно розробити моделі, які відображатимуть основні характеристики та поведінку автомобіля, включаючи рух, реакції на керування, взаємодію з дорожнім покриттям та інші аспекти.
2. Аналіз існуючих бібліотек і алгоритмів: провести огляд доступних бібліотек та алгоритмів, які можуть бути використані для моделювання поведінки автомобіля. Оцінити їхню ефективність, точність і схожість для поставленої задачі.
3. Вибір відповідного алгоритму: на основі результатів аналізу обрати найбільш підходящий алгоритм для моделювання поведінки автомобіля. Врахувати його можливості, обмеження і відповідність до вимог проекту.
4. Реалізація алгоритмів і інтерфейсу користувача: розробити програмний продукт, який включатиме реалізацію вибраного алгоритму для моделювання поведінки автомобіля. Забезпечити зручний та інтуїтивно зрозумілий інтерфейс користувача для взаємодії з програмою та візуалізацію результатів моделювання.

2.2. Розвинутий огляд сучасного стану справ в області.

У рекомендаційних системах використовуються різні типи методів штучного інтелекту, кожен з яких відрізняється швидкістю роботи та точністю прогнозів. Ось деякі статистичні дані про їх використання:

1. Колаборативна фільтрація: Колаборативна фільтрація широко застосовується в рекомендаційних системах. Вона базується на припущенні, що люди з схожими уподобаннями в минулому матимуть аналогічні уподобання і в майбутньому. Вона має відносно швидку швидкість роботи, але її точність залежить від наявності достатньої кількості даних про минулі уподобання користувачів.
2. Фільтрація за змістом: Фільтрація за змістом - ще один популярний метод, який рекомендує елементи на основі їх атрибутів, таких як жанр або

ключові слова. Вона має швидку швидкість роботи, але її точність обмежена якістю та кількістю атрибутів об'єкта.

3. Матрична факторизація: Матрична факторизація - це метод машинного навчання, який використовується в рекомендаційних системах і намагається знайти приховані фактори, що пояснюють взаємодію користувача з елементом. Вона має меншу швидкість роботи, ніж колаборативна фільтрація та фільтрація за змістом, але вона може обробляти розріджені дані і покращити точність прогнозів.
4. Глибоке навчання: Глибоке навчання - це потужна техніка машинного навчання, яка може вивчати складні представлення даних. Воно застосовується в рекомендаційних системах для вивчення представлень користувачів та елементів. Воно має більш повільну швидкість роботи, ніж попередні згадані техніки, але дозволяє досягти високої точності прогнозів.
5. Гібридні рекомендаційні системи: Гібридні рекомендаційні системи поєднують кілька методів, таких як колаборативна фільтрація та фільтрація за змістом, для підвищення точності прогнозів. Вони мають меншу швидкість роботи, ніж окремі методики, але дозволяють досягти більш високої точності передбачень.

Отже, вибір методу штучного інтелекту в рекомендаційній системі залежить від швидкості роботи та необхідної точності прогнозів. Колаборативна фільтрація та фільтрація за змістом працюють швидко, але мають обмежену точність, тоді як матрична факторизація, глибоке навчання та гібридні рекомендаційні системи працюють повільніше, але можуть забезпечити вищу точність.

Далі приводиться пару прикладів подібних систем

1. Система рекомендацій для інтернет магазину Stall [13]

Ціль. Поліпшити список Схожих товарів для підвищення коефіцієнта транзакції скоротити час на вирішення рутинних завдань контент менеджерами.

Особливості проекту. В інтернет-магазині близько 3000 товарів. При додаванні нового товару на сайт досвідчені співробітники, які добре знають асортимент, визначають схожі на нього товари суб'єктивно. Як правило, вибираючи товари з однієї категорії.

Рішення. Керівництву магазину було запропоновано впровадити розумні товарні рекомендації подібних товарів uXprice. У її основі лежить кластеризація товарів за схожістю, виконана машинними алгоритмами з урахуванням штучного інтелекту. Для визначення більш ефективного списку між існуючим і новим варіантом, було вирішено провести A/B тестування за допомогою Google Optimize. Суб'єктивно нові списки схожих товарів сподобалися і співробітникам і керівництву набагато більше, ніж попередній варіант.

Результати цього дослідження надають підтвердження ефективності нової системи рекомендацій товарів в порівнянні з попереднім суб'єктивним підходом. Для об'єктивної оцінки та визначення переваги нової вибірки, було проведено A/B тестування з використанням Google Optimize. Експеримент тривав 8 днів, з 8 по 16 вересня 2020 року, і показав, що нова система рекомендацій має ймовірність переваги на рівні 99%.

Під час тестування було зареєстровано 55 066 сеансів. Половина відвідувачів сайту (27 738 сеансів) бачила оригінальний варіант списку схожих товарів, створений співробітниками, а інша половина (27 328 сеансів) бачила новий варіант від сервісу uXprice.

Для оцінки ефективності, найважливішим для бізнесу є отриманий дохід. Від користувачів з оригінальним списком схожих товарів було отримано дохід в сумі 87 851,82 грн. А від користувачів з варіантом від uXprice — 105 773,09 грн (на 12% більше). Перераховано на один сеанс, дохід за сеанс з оригінальним списком складав у середньому 3,17 грн, а з варіантом від uXprice — 3,87 грн. Таким чином, варіант списку схожих товарів від uXprice забезпечував дохід за сеанс на 22% вище, ніж оригінальний список. Загальний дохід з новим варіантом був на 12% вище, ніж при сеансах з оригінальним списком.

2. Система рекомендацій Netflix [12]

Netflix Prize був відкритим конкурсом на найкращий алгоритм спільної фільтрації для прогнозування рейтингу фільму користувача на основі попередніх оцінок інших користувачів. Не було жодної інформації про користувача чи фільм, лише ідентифікатор. У той час Netflix використовував Cinematch, власну систему рекомендацій, засновану на лінійній регресії із середньоквадратичною помилкою (RMSE) = 0,9525. У конкурсі учасникам було поставлено завдання знизити цей показник на 10%. Грошову премію отримувала команда, яка досягла або наблизилася до мети протягом року. Через рік, у 2007 році, переможцями Progress Prize (міжнародної нагороди, яка щорічно вручається учасникам Global Grad Show для визнання наступного покоління новаторів) стали переможці матричної факторизації (SVD) і обмеженої машини Больцмана (RBM). Вони отримали середньо квадратичне відхілення 0,88. Після деяких змін у коді Netflix запустив ці алгоритми у виробництво.

Зараз Netflix використовує різні ранкери, хоча конкретні деталі архітектури кожної моделі не вказуються. Ось короткий опис кожного з них:

Personalized Video Ranking (PVR) - цей алгоритм є загальним і зазвичай фільтрує каталог за певними критеріями (наприклад, насильствені телепрограми, американські телешоу, романтика тощо), поєднуваний з показниками користувача та популярністю.

Top-N Video Ranker - схожий на PVR, але він розглядає лише перші позиції рейтингу і аналізує весь каталог. Він оптимізований за допомогою метрик, що оцінюють перші позиції в рейтингу (наприклад, MAP@K, NDCG).

Trending Now Ranker - цей алгоритм враховує тимчасові тренди, які Netflix вважає сильними прогнозаторами. Ці короткострокові тренди можуть тривати від декількох хвилин до декількох днів. Це можуть бути:

1. Події, які мають сезонний тренд і повторюються (наприклад, День Святого Валентина призводить до збільшення перегляду романтичних відео).

2. Одноразові події короткострокового характеру (наприклад, коронавірус або інші катастрофи, що призводять до тимчасового інтересу до документальних фільмів про них).

Continue Watching Ranker - цей алгоритм аналізує елементи, які користувач переглянув, але не завершив, зазвичай:

1. Епізодичний контент (наприклад, драматичні серіали).
2. Непродовжуваний контент, який можна переглядати невеликими порціями (наприклад, фільми, які були переглянуті наполовину, серіали, що не залежать від епізодів, наприклад, "Чорне дзеркало").

Алгоритм обчислює ймовірність того, що користувач продовжить перегляд, і враховує інші контекстно-залежні сигнали (наприклад, час, що минув з моменту перегляду, точка припинення перегляду, пристрій, на якому переглядали).

Video-Video Similarity Ranker (також його називають "Because You Watched" Ranker) - цей алгоритм схожий на алгоритм фільтрації за вмістом. На основі елемента, який користувач переглянув, алгоритм обчислює інші подібні елементи (використовуючи матрицю подібності елементів) і повертає найбільш подібні елементи. Серед інших алгоритмів цей є неперсоналізованим, оскільки не використовується жодні інші показники користувача. Однак він персоналізований в тому сенсі, що це свідомий вибір показувати подібні елементи певного елемента на домашній сторінці користувача.

Використання кластерного аналізу в загальному вигляді складається з наступних етапів:

1. Відбір вибірки об'єктів для кластеризації.
2. Визначення множини змінних, за якими будуть оцінюватися об'єкти в вибірці. При необхідності - нормалізація значень змінних.
3. Обчислення значень міри схожості між об'єктами.
4. Застосування методу кластерного аналізу для створення груп подібних об'єктів (кластерів).

5. Представлення результатів аналізу.

Після отримання та аналізу результатів можлива корекція вибраної метрики та методу кластеризації для отримання оптимального результату. Після визначення вектора характеристик можна провести нормалізацію, щоб всі компоненти давали однаковий внесок при розрахунку "відстані". У процесі нормалізації всі значення приводяться до певного діапазону, наприклад, [-1, -1] або [0, 1]. Нарешті, для кожної пари об'єктів вимірюється "відстань" між ними - ступінь схожості. Існує безліч метрик, ось лише основні з них:

1. Евклідова відстань

   Евклідова відстань - найпоширеніша міра відстані між двома точками в багатовимірному просторі. Вона обчислюється як геометрична відстань між двома точками в прямій лінії. Формула для обчислення євклідової відстані:

$$\rho(x, x') = \sqrt{\sum_{i}^{n} (x_i - x_i')^2}$$

2. Квадрат євклідової відстані

Використовується для надання більшої ваги більш віддаленим один від одного об'єктам. Ця відстань обчислюється наступним чином:

$$\rho(x, x') = \sum_{i}^{n} (x_i - x_i')^2$$

2. Відстань між містами (манхеттенська відстань)

Ця відстань є середньою різницею за координатами. У більшості випадків ця міра відстані дає ті ж самі результати, що й звичайна євклідова відстань. Однак для цієї міри вплив окремих великих різниць (викидів) зменшується (тому що вони не підносяться до квадрату). Формула для розрахунку манхеттенської відстані:

$$\rho(x, x') = \sum_{i}^{n} | x_i - x_i' |$$

3. Відстань Чебишова

Відстань Чебишова є корисною, коли потрібно визначити два об'єкта як "різні", якщо вони відрізняються по одній конкретній координаті. Ця відстань обчислюється за допомогою спеціальної формули, яка враховує максимальну різницю між координатами об'єктів.

$$\rho(x, x') = \max(| x_i - x_i' |)$$

4. Степенева відстань

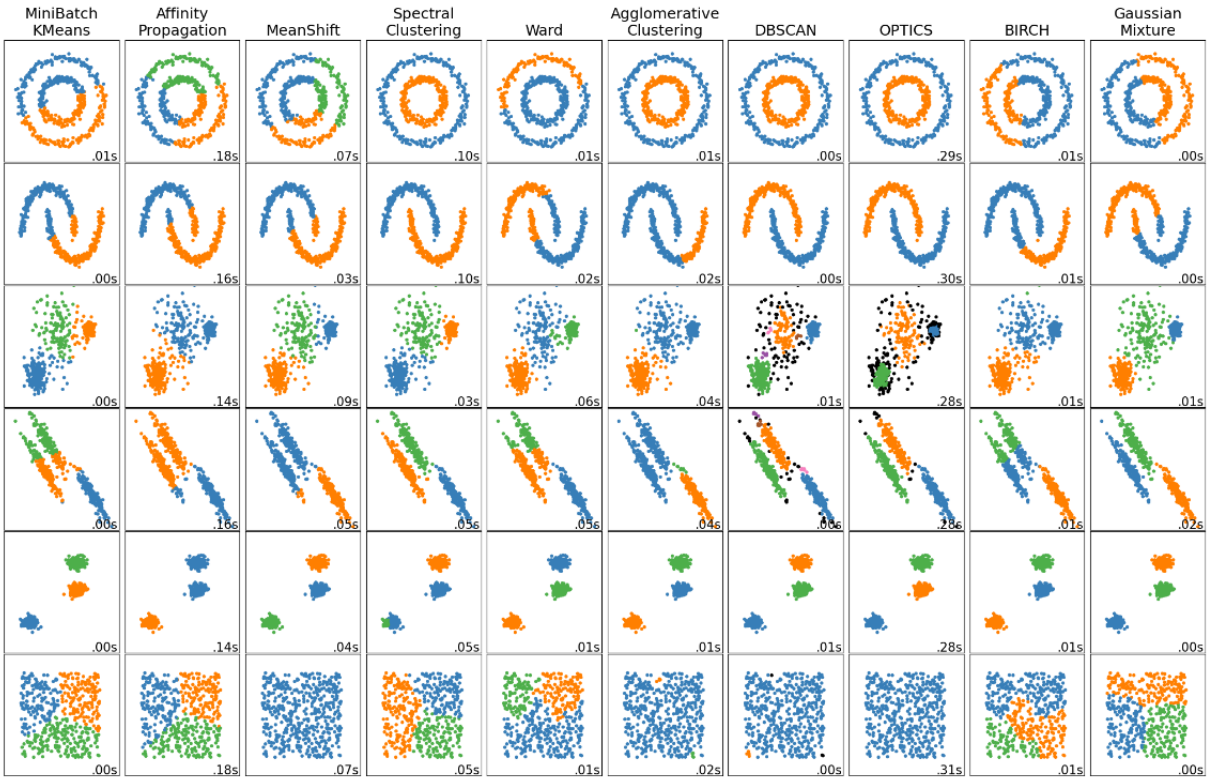
Степенева відстань застосовується тоді, коли потрібно враховувати вагу різних розмірностей, де об'єкти можуть значно відрізнятися. Ця відстань обчислюється за формулою, в якій враховуються параметри, встановлені користувачем. Ці параметри визначають, як враховувати різницю між координатами та вагу відстаней між об'єктами. При певних значеннях параметрів, степенева відстань може збігатися з євклідовою відстанню.

$$\rho(x, x') = \sqrt[r]{\sum_{i}^{n} (x_i - x_i')^p},$$

де r і p - параметри, що визначаються користувачем. Параметр p відповідає за поступове зважування різниць по окремим координатам, параметр r відповідає за прогресивне зважування великих відстаней між об'єктами. Якщо обидва параметри - r і p - дорівнюють двом, то ця відстань збігається з євклідовою відстанню. Вибір метрики повністю залежить від дослідника, оскільки результати кластеризації можуть суттєво відрізнятися при використанні різних метрик.

Найпопулярніші бібліотеки Python для кластеризації:

1. Scikit-learn є однією з найпопулярніших і широко використовуваних бібліотек Python для машинного навчання, включаючи кластеризацію. Вона надає широкий спектр алгоритмів кластеризації, таких як K-Means, DBSCAN, ієрархічна кластеризація та багато інших. Далі приведені таблиці було взято з [1].



A comparison of the clustering algorithms in scikit-learn

2. PyClustering є ще одною популярною бібліотекою для кластеризації, яка надає ряд алгоритмів, таких як K-Means, DBSCAN, ієрархічна кластеризація та інші. Вона також підтримує візуалізацію та оцінку кластерів.
3. HDBSCAN - це ієрархічний алгоритм кластеризації на основі густини, який може виявляти кластери з різною щільністю та розмірами. Це стійкий алгоритм, який добре працює з шумними та високорозмірними даними. Далі приведені таблиці(див. рис. 1, 2) було взято з [2].

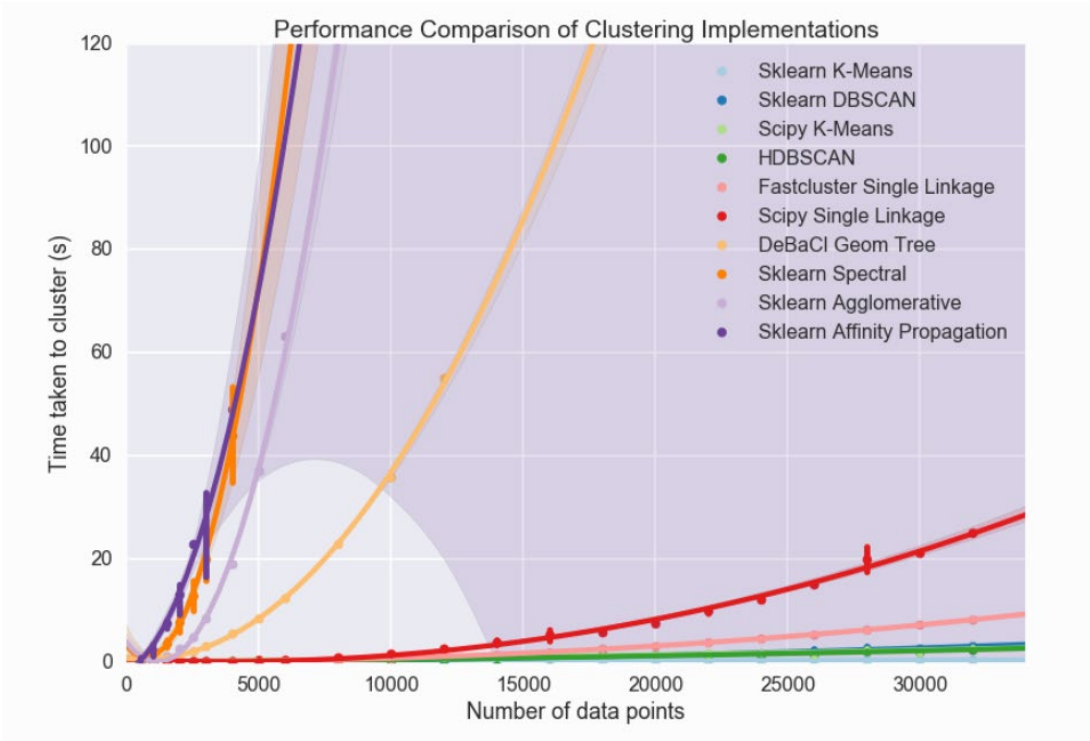


Рис. 1. Порівняння продуктивності реалізацій кластеризації

Рис. 2. Порівняння продуктивності найшвидших реалізацій кластеризації

4. Fastcluster - це бібліотека Python для кластеризації великих наборів даних. Вона надає ефективні алгоритми для ієрархічної кластеризації та кластеризації KMeans.

Таблиця нижче показує час роботи різних алгоритмів вище наведених бібліотек при 10000 вхідних даних:

Library Name	**Function**	**Input Data Size**	**Processing Time**
Scikit-learn	KMeans	10000	1.4 s
Scikit-learn	Agglomerative Clustering	10000	10.6 s
Scikit-learn	Spectral Clustering	10000	15.7 s
Scikit-learn	DBSCAN	10000	50.3 s
Scikit-learn	Birch	10000	0.15s
Scikit-learn	GaussianMixture	10000	4.4 s
Scikit-learn	OPTICS	10000	92.1 s
HDBSCAN	HDBSCAN	10000	6.5 s
HDBSCAN	HDBSCAN (approx)	10000	1.4 s
HDBSCAN	HDBSCAN (single linkage)	10000	11.2 s

Library Name	Function	Input Data Size	Processing Time
HDBSCAN	HDBSCAN (euclidean)	10000	19.7 s
HDBSCAN	HDBSCAN (manhattan)	10000	19.7 s
HDBSCAN	HDBSCAN (cosine)	10000	28.3 s
Fastcluster	Single linkage	10000	0.1 s
Fastcluster	Complete linkage	10000	0.2 s
Fastcluster	Average linkage	10000	0.4 s
Fastcluster	Ward linkage	10000	1.1 s
Fastcluster	DBSCAN (single linkage)	10000	2.7 s
Fastcluster	DBSCAN (complete linkage)	10000	3.6 s
Fastcluster	DBSCAN (average linkage)	10000	3.6 s
Fastcluster	DBSCAN (ward linkage)	10000	5.8 s

Отже, було вирішено використати бібліотеку scikit-learn, алгоритм KMeans для кластерізації, а потім застосувати Самоорганізовані карти Кохонена (Self-Organizing Maps, SOM). Самоорганізовані карти відрізняються від інших штучних

нейронних мереж тим, що вони застосовують конкурентне навчання замість навчання з виправленням помилок, такого як зворотне поширення з градієнтним спуском. Замість того, щоб коригувати ваги на основі точних цілевих значень, самоорганізуючіся карти використовують функцію сусідства для збереження топологічних властивостей вхідного простору.

Це означає, що при навчанні самоорганізуючихся карт немає явного навчання з використанням пари "вхід-вихід". Замість цього, карти самоорганізуються шляхом конкуренції між нейронами за вплив на вхідні дані. Нейрон, який найбільше схожий на вхідні дані, вважається переможцем і стає активованим. При цьому активований нейрон та його сусіди на карті піддаються оновленню ваг, що дозволяє карті адаптуватися до вхідних даних.

Переваги такого підходу:

1. Простота реалізації: Бібліотека scikit-learn надає легкий у використанні та добре документований інтерфейс для кластеризації даних за допомогою алгоритму KMeans. Вона має багато вбудованих функцій та параметрів, що полегшують налаштування та оптимізацію процесу кластеризації.
2. Швидкість виконання: Алгоритм KMeans є одним з найшвидших алгоритмів кластеризації, особливо для великих наборів даних. Він має лінійну складність за кількістю зразків та кількістю кластерів, що дозволяє швидко обробляти великі обсяги даних.
3. Чітка інтерпретація результатів: KMeans надає простий та зрозумілий спосіб розділення даних на кластери. Кожен об'єкт призначається до найближчого центру кластера, що дозволяє легко інтерпретувати результати і використовувати їх для подальшого аналізу.

Недоліки:

1. Залежність від початкового вибору центрів кластерів: Алгоритм KMeans чутливий до початкового вибору центрів кластерів. Результати кластеризації можуть відрізнятися в залежності від початкових умов, що

може вимагати декількох повторних запусків з різними початковими умовами для отримання більш стабільних результатів.

2. Проблема зі сферичними кластерами: Алгоритм KMeans припускає, що кластери мають сферичну форму та однакову дисперсію. Це може приводити до проблем, коли кластери мають складну форму або різну дисперсію, оскільки алгоритм може незадовільно розділяти такі кластери.
3. Відсутність знання про кількість кластерів: Алгоритм KMeans вимагає заздалегідь визначеної кількості кластерів. Якщо ви не знаєте точну кількість кластерів у ваших даних, ви можете використовувати різні підходи для вибору оптимальної кількості кластерів, такі як метод "ліктя" або інші евристичні методи.

2.3. Опис моделі вимог до інформаційної системи.

1. Функціональні вимоги:
   - Система повинна забезпечувати можливість створення та збереження інформаційних моделей автомобілів, що включають характеристики руху, динаміку, реакції на керування тощо.
   - Система має надавати можливість аналізувати та порівнювати різні інформаційні моделі автомобілів для оцінки їх ефективності та вибору оптимального рішення.
   - Інформаційна система повинна мати зручний інтерфейс користувача, що дозволяє легко створювати, редагувати та візуалізувати інформаційні моделі автомобілів.
2. Вимоги до надійності:
   - Система повинна забезпечувати стабільну та надійну роботу без збоїв або відмов.
   - Моделі автомобілів повинні бути збережені у безпечному та надійному сховищі, щоб запобігти втраті даних.
3. Вимоги до продуктивності:

- Система повинна працювати швидко та ефективно, забезпечуючи швидку відповідь на запити користувача.
- Завантаження та відображення інформаційних моделей автомобілів повинні відбуватись без помітних затримок.

2.4. Опис архітектури інформаційної системи.

Після аналізу різних бібліотек і вибору відповідних алгоритмів було перейдено до створення моделі автомобіля. За основу була взята наведена нижче схема автомобіля(див. рис. 3), на основі якої була побудована інформаційна модель автомобіля.

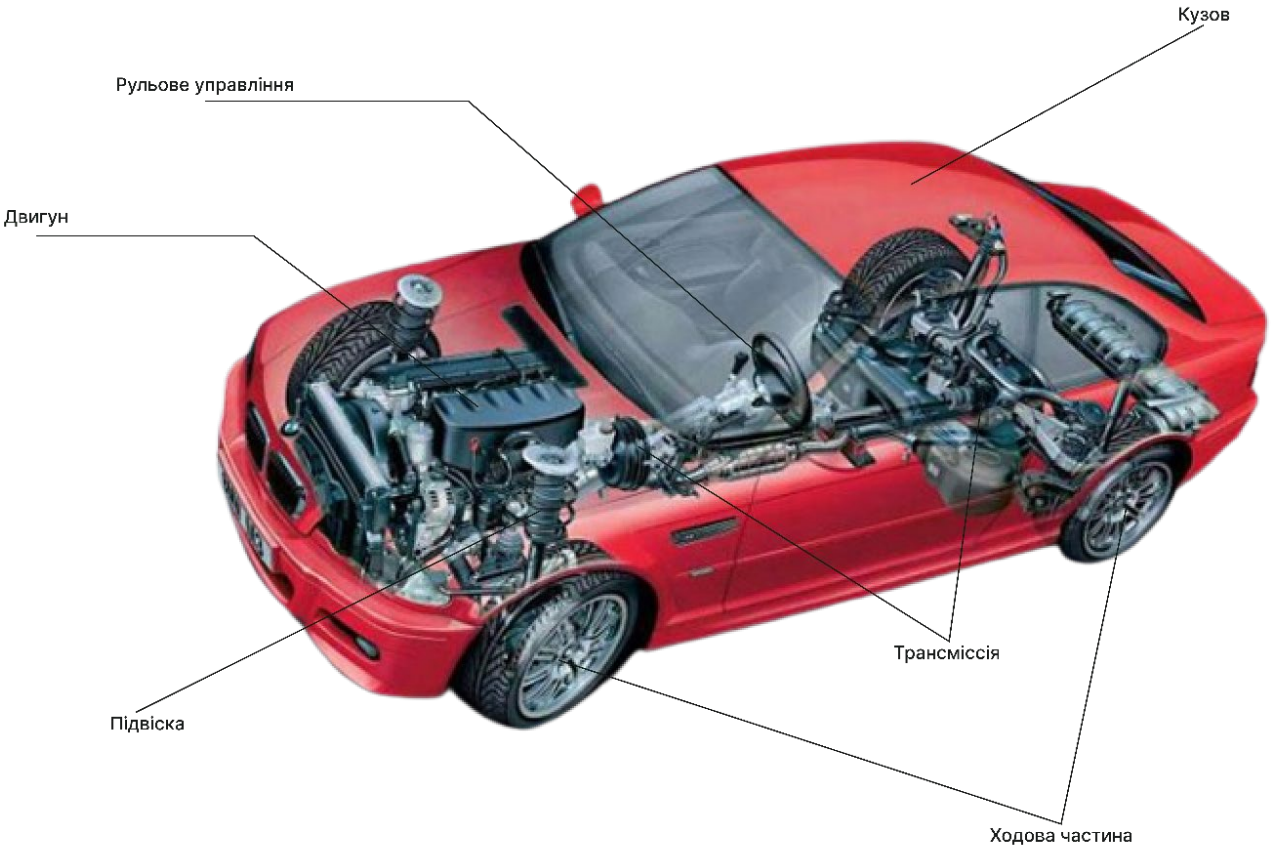


Рис. 3. Архітектура автомобіля

Інформаційна модель включає в себе компоненти та системи, які забезпечують обмін даними та передачу інформації між різними системами автомобіля.

Основні елементи цієї схеми включають:

- Кузов
- Двигун
- Ходова частина
- Габарити

Кожен із цих компонентів має власну внутрішню архітектуру, з якою вони взаємодіють і передають дані.

Кузов, наприклад, має з’єднання з пристроями безпеки та салоном. Наприклад, системи клімат-контролю, розподілу повітря, електропідсилювачі дверей та інші внутрішні елементи кузова.

Двигун є основним джерелом енергії автомобіля. Модель передачі даних містить дані про режими роботи двигуна, температуру, витрату палива та інші характеристики.

Шасі транспортного засобу містить такі компоненти, як гальма, рульове керування та системи керування. Ці компоненти підвищують комфорт водія, стабільність і керованість.

Розміри автомобіля визначають розміри автомобіля, такі як його довжина, ширина та висота. Ці вимірювання необхідні для вимірювання маневреності автомобіля, його прохідності та здатності маневрувати у важкодоступних місцях.

Усі ці компоненти та системи взаємодіють одна з одною, обмінюючись даними та інформацією, щоб зберегти безпеку, комфорт та ефективність автомобіля. Інформаційна модель транспортного засобу полегшує обмін даними та забезпечує оптимальну роботу всіх систем автомобіля. Однак пов’язати всі вищевказані фактори буде складно, і не всі з них є обов’язковими.(див. рис. 4)

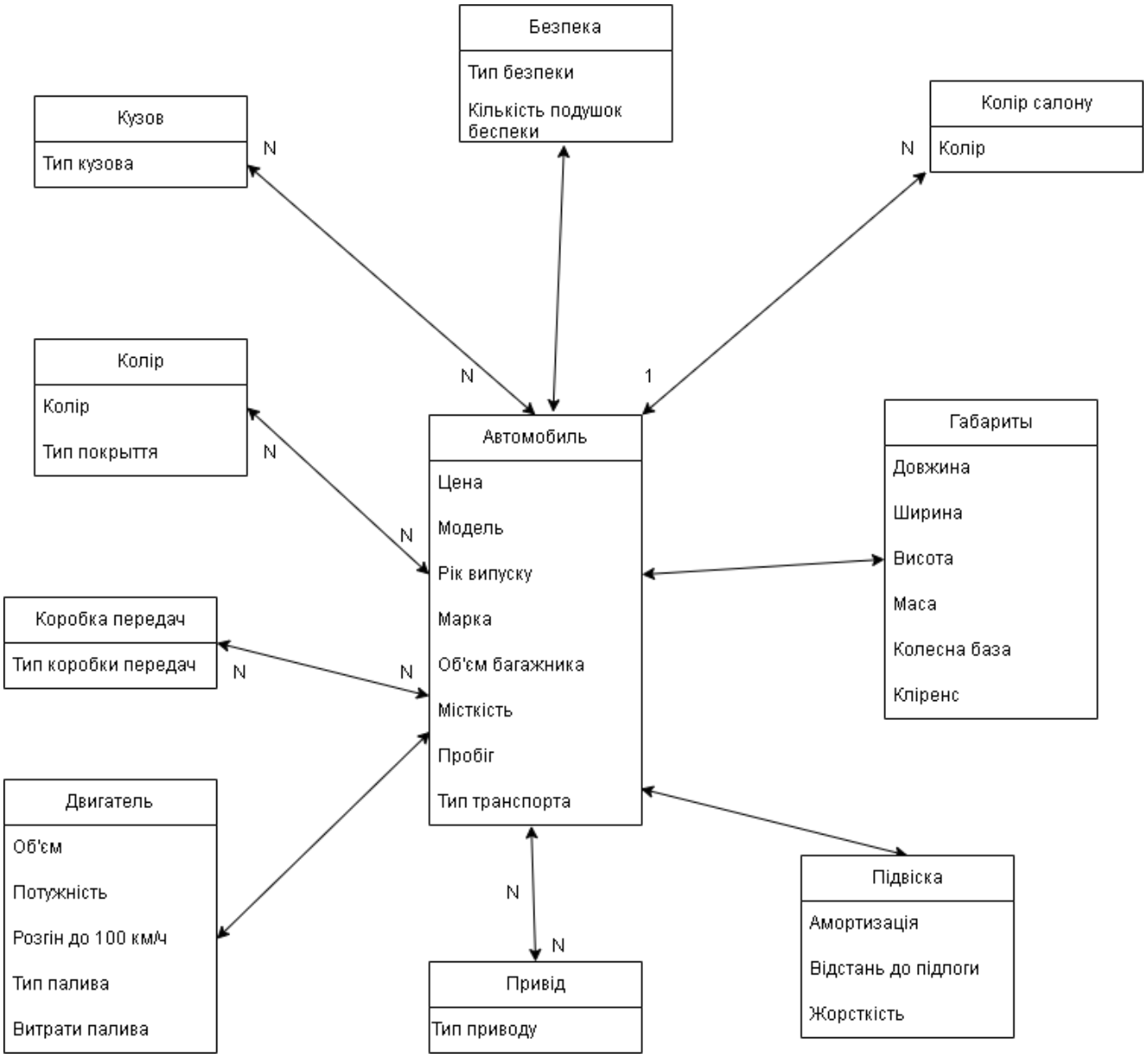


Рис. 4. Інформаційна модель автомобіля

При створені моделі у MySQL були внесені деякі правки, які в основному спрямовані на зміну структури таблиць.(див. рис. 5)

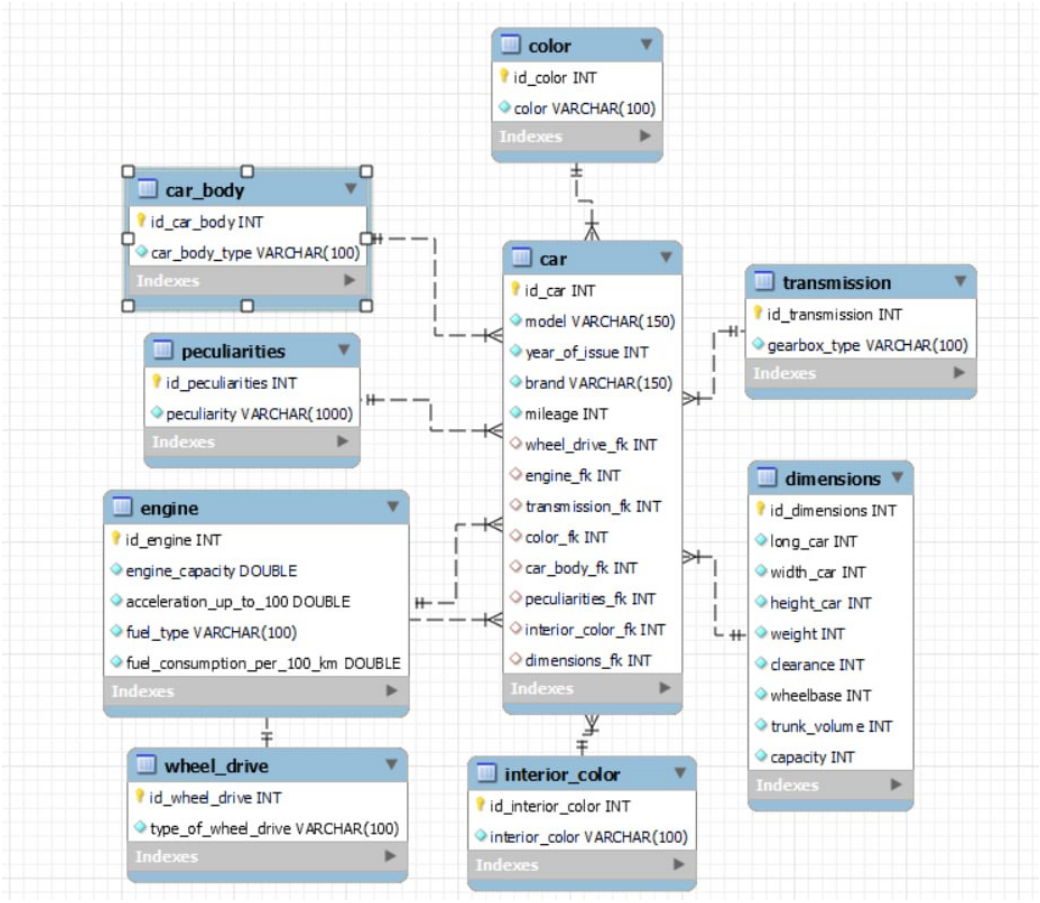


Рис. 5. Інформаційна модель автомобіля у MySQL

В роботі "ТЕОРЕТИЧНЕ ОБҐРУНТУВАННЯ ОЦІНКИ ЯКОСТІ ЛЕГКОВИХ АВТОМОБІЛІВ" авторства Т.О. Бажинової[5], було проведено обстеження та визначено повну оцінку якості автомобіля. У даній роботі основні показники комфорту, надійності, безпеки, технологічних інновацій та екологічності використовувалися для прийняття рішень. Розроблено набір показників для вимірювання ступеня якості вживаних автомобілів в країні. Показники якості та інтегральні показники мають математичні зв’язки.

Атрибут «Комфорт» можна розбити на такі складові показники як:

1. клас автомобіля;
2. тип кузова;

Автомобілі в інших країнах поділяються на класи (A, B, C, D, E, S). Розміри автомобіля (колісна база та загальні розміри) визначають, чи належить автомобіль до одного з цих класів.

При цьому найбільші розміри відповідають класу S, а найнижчі – класу A. Найпрестижніші автомобілі представницького класу часто мають надзвичайно великі розміри.

Надійність автомобіля визначається під час проектування, забезпечується під час виробництва, проявляється і підтримується під час експлуатації. Власнику необхідно витрачати гроші на обслуговування та ремонт, щоб автомобіль працював ефективно. Вартість експлуатації та обслуговування, за оцінками Об'єднаного німецького автомобільного клубу (ADAK), складається з багатьох компонентів:

- податок з власників транспортних засобів;
- поточні витрати (паливо, масло, ремонт і технічне обслуговування);
- амортизація як втрата вартості автомобіля з часом за певний пробіг;

Як результат, оцінка функціональності автомобіля за критерієм працездатності (надійності) повинна бути зосереджена на витратах на експлуатацію та обслуговування автомобіля та періодичності технологічних впливів. При оцінці якості транспортного засобу слід також враховувати активну та пасивну безпеку.

Вимірювання якості автомобіля вимагає вивчення його активних і пасивних функцій безпеки. Пасивну безпеку можна описати як здатність транспортного засобу захистити водія та пасажирів від серйозних травм у разі аварії. Це визначається здатністю транспортного засобу захистити пасажирів від смертельних випадків і серйозних травм під час зіткнення на швидкості до 50 км/год.

Здатність транспортного засобу запобігати зіткненню або аварії за допомогою навігації та забезпечення таких функцій, як керованість, стійкість і ефективність гальмування, називається активною безпекою. Це і плавність руху, що дозволяє надавати автомобілю керованість, стійкість і надійне гальмування навіть після незначних доопрацювань.

Як наслідок, оцінка якості автомобіля повинна враховувати як пасивну, так і активну безпеку, щоб запобігти шкоді водієві та пасажирам від несподіваних аварій.

"Технічне рішення" оцінюється з точки зору ефективності, інноваційності та технологічності використовуваних компонентів і систем. Основні компоненти "Технічного рішення" можуть включати двигун, трансмісію, систему керування, підвіску, гальма, електронні системи управління, комунікаційні системи та інші технічні деталі. Оцінка "Технічного рішення" враховує як продуктивність і потужність автомобіля, так і його надійність, ресурсність та легкість обслуговування.

"Екологічність" відноситься до впливу автомобіля на навколишнє середовище. Оцінка "Екологічності" зазвичай базується на таких факторах, як викиди шкідливих речовин (наприклад, вуглекислого газу, оксидів азоту, сажі, ртуті), рівень споживання палива та його ефективність, використання екологічно чистих матеріалів та технологій у виробництві автомобіля, наявність альтернативних енергетичних джерел (наприклад, гібридних або електричних систем).

Далі автор представив таблицю з формулами коефіцієнтів:

Показники якості	Математичний вираз	Умовне позначення
Комфорт	$К_ф = V_б * L_б/2,1 * V_c (1)$	V_c – об'єм салону, м3; $V_б$ – об'єм багажника, м3; $L_б$ – база автомобіля, м;
Надійність	$К_Н = З_{тор}/L_{ТО} \cdot Н_л \cdot С_Т (2)$	$З_{тор}$ – витрати на ТО и ремонт за міжсервісний пробіг, грн; $L_{то}$ – періодичність ТО; $Н_л$ – витрата палива,

		л/100 км; C_T – вартість літра палива, грн;
Безпека	$К_б = S_Т/S_{ТН}(n_{п.б.} + L_а/L_б)(3)$	$S_Т$ – гальмівний шлях при швидкості 100 км/год, м; $S_{ТН}$ – найменший гальмівний шлях серед усіх учасників експерименту автомобілів, м. $n_{п.б.}$ – кількість подушок безпеки; $L_а$ – довжина автомобіля, м; $L_б$ – довжина бази автомобіля, м;
Технічне рішення	$К_Т = 0.36 * H_{л.min}V_{max} * t_p * ρ_T/G_a(4)$	$H_{л.min}$ – мінімальна витрата палива, л/100 км; V_{max} – максимальна швидкість автомобіля, км/год; $t_р$ – час розгону до 100 км/год; $р_Т$ – густина палива кг/м3; $G_а$ – маса автомобіля, кг;

Екологічність	$К_{ек} = G_а * Э_Т / Н_л (5)$	$G_а$ – маса автомобіля, кг; $Н_л$ – витрата палива, л/100 км; $Э_т$ – еталонна енергія витрат на 100 км пробігу

Маючи параметри конкретного автомобіля можна обчислити інтегральний показник $К_{ІН} = К_ф + К_Н + К_б + К_Т + К_{ЭК}$.

Деяка інформація була відсутня у статті[5], але ці характеритики і формули були пізніше взяті з статті [6] і [7].

Після ознайомлення з дослідженнями було вирішено взяти їх за основу математичної моделі, яку приведено нижче(див. рис. 6).

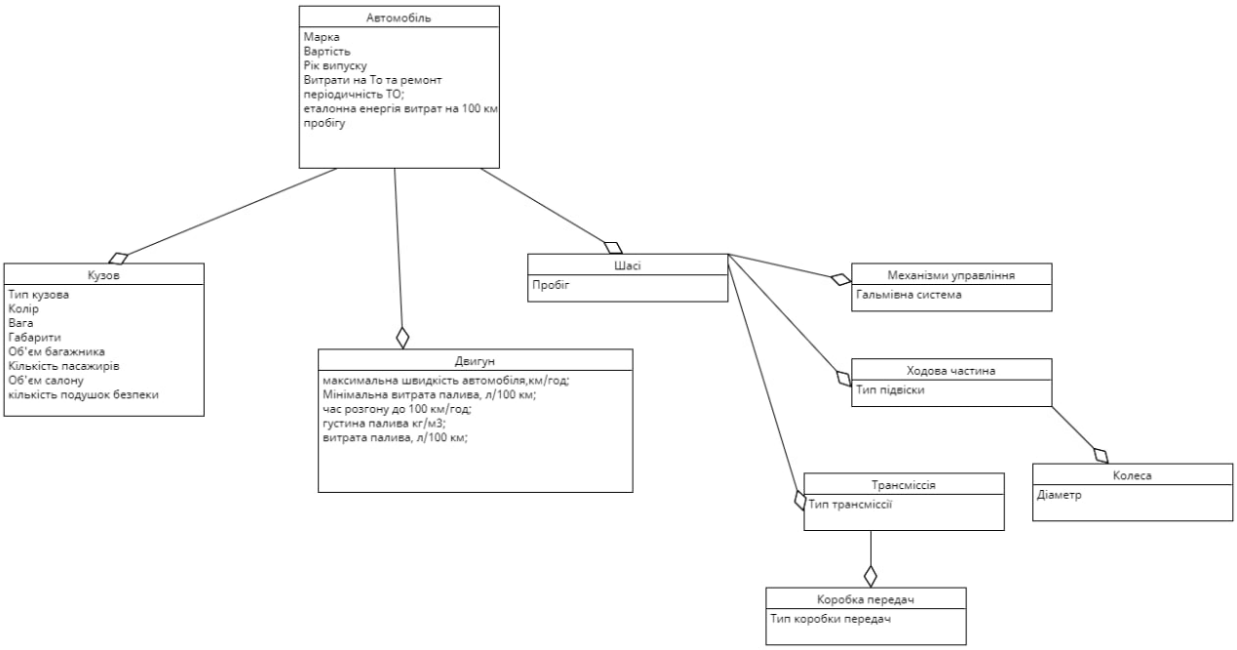


Рис. 6. Математична модель автомобіля

2.5. Опис функціональності системи.

1. Створення інформаційних моделей автомобілів: Система повинна надавати можливість користувачам створювати нові інформаційні моделі автомобілів. Це може включати введення характеристик автомобіля, даних про рух, динаміку та інші параметри.
2. Візуалізація інформаційних моделей: Система повинна забезпечувати візуальне відображення інформаційних моделей автомобілів, щоб користувачі могли переглядати та аналізувати їх характеристики.
3. Збереження та завантаження моделей: Користувачі мають можливість зберігати створені моделі автомобілів у системі та завантажувати їх для подальшого використання. Це дозволяє зберігати та обмінюватись моделями між користувачами.
4. Інтерфейс користувача: Система повинна мати зручний та інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам легко взаємодіяти з функціоналом системи, виконувати операції та отримувати результати.

2.6. Опис та обґрунтування алгоритмів.

Одним зі способів розподілу п'яти коефіцієнтів на дві групи може бути використання методу групування за схожістю впливу на інтегральний показник. Після аналізу було встановлено, що $К_ф$, $К_Н$, $К_б$ і $К_Т$ мають приблизно однаковий вплив на показник, подібний до впливу $К_{ЭК}$ (див. рис. 7). Таким чином, ці чотири коефіцієнти можуть бути об'єднані в одну групу, тоді як $К_{ЭК}$ залишиться окремо.

	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	0.15120274	0.10996563	0.14432989	0.17869415	0.41580756	1				
3	2	0.25	0.11	0.17	0.27	0.66	1.46	0.17123287	0.07534246	0.11643835	0.18493150	0.45205479	1				
4	3	0.1	0.12	0.08	0.17	0.528	0.998	0.10020040	0.12024048	0.08016032	0.17034068	0.52905811	1				
5	4	0.1	0.15	0.12	0.21	0.429	1.009	0.09910802	0.14866204	0.11892963	0.20812685	0.42517343	1				
6	5	0.22	0.15	0.2	0.25	0.682	1.502	0.14647137	0.09986684	0.13315579	0.16644474	0.45406125	1				
7	6	0.2	0.18	0.17	0.27	0.66	1.48	0.13513513	0.12162162	0.11486486	0.18243243	0.44594594	1				
8	7	0.21	0.17	0.14	0.3	0.693	1.513	0.13879709	0.11235955	0.09253139	0.19828155	0.45803040	1				
9	8	0.12	0.12	0.03	0.14	0.715	1.125	0.10666666	0.10666666	0.02666666	0.12444444	0.63555555	1				
10	9	0.16	0.2	0.04	0.13	0.737	1.267	0.12628255	0.15785319	0.03157063	0.10260457	0.58168902	1				
11	10	0.15	0.12	0.03	0.15	0.913	1.363	0.11005135	0.08804108	0.02201027	0.11005135	0.66984592	1				
12	11	0.18	0.17	0.02	0.18	1.144	1.694	0.10625737	0.10035419	0.01180637	0.10625737	0.67532467	1				
13	12	0.09	0.11	0.05	0.14	0.693	1.083	0.08310249	0.10156971	0.04616805	0.12927054	0.63988919	1				
14	13	0.26	0.21	0.1	0.17	0.792	1.532	0.16971279	0.13707571	0.06527415	0.11096605	0.51697127	1				
15	14	0.22	0.18	0.07	0.17	0.572	1.212	0.18151815	0.14851485	0.05775577	0.14026402	0.47194719	1				
16	15	0.07	0.14	0.14	0.19	0.682	1.222	0.05728314	0.11456628	0.11456628	0.15548281	0.55810147	1				
17	16	0.04	0.18	0.2	0.18	0.704	1.304	0.03067484	0.13803680	0.15337423	0.13803680	0.53987730	1				
18	17	0.18	0.13	0.21	0.24	0.616	1.376	0.13081395	0.09447674	0.15261627	0.17441860	0.44767441	1				
19	18	0.19	0.12	0.11	0.22	0.605	1.245	0.15261044	0.09638554	0.08835341	0.17670682	0.48594377	1				
20	19	0.19	0.2	0.14	0.24	0.671	1.441	0.13185287	0.13879250	0.09715475	0.16655100	0.46564885	1				
21	20	0.3	0.21	0.13	0.22	0.682	1.542	0.19455252	0.13618677	0.08430609	0.14267185	0.44228274	1				
22	21	0.1	0.19	0.06	0.23	0.572	1.152	0.08680555	0.16493055	0.05208333	0.19965277	0.49652777	1				
23	22	0.45	0.18	0.1	0.21	0.836	1.776	0.25337837	0.10135135	0.05630630	0.11824324	0.47072072	1				
24	23	0.07	0.12	0.12	0.18	0.627	1.117	0.06266786	0.10743061	0.10743061	0.16114592	0.56132497	1				
25	24	0.39	0.15	0.15	0.26	0.77	1.72	0.22674418	0.08720930	0.08720930	0.15116279	0.44767441	1				
26	25	0.15	0.14	0.15	0.18	0.704	1.324	0.11329305	0.10574018	0.11329305	0.13595166	0.53172205	1	Mean	Range	Std	
27						Mean	1.35648	0.13065663	0.11652962	0.08673423	0.15332538	0.51275411	1	0.2	0.42601988	0.17649263	9.78500899
28						Range	0.778	0.22270353	0.08958808	0.14156785	0.10552228	0.25951711					
29						Std	0.2156579:	0.05139819	0.02374020	0.04112643	0.03114283	0.07762714					

Рис. 7. Результати аналізу базових показників

Згрупувавши п'ять коефіцієнтів в дві категорії, значно спрощується процес оптимізації. Замість аналізування п'яти окремих показників, тепер маємо лише два зведені показники, які представляють ці категорії. Це значно зменшує складність завдання і дозволяє зосередитися на ключових аспектах оптимізації.

Маючи лише два зведені показники, можливо легше порівнювати їх значення та аналізувати вплив кожної групи коефіцієнтів на загальний результат. Це спрощує прийняття рішень, оскільки опрацьовується менша кількість факторів і можливо більш ефективно оцінювати, які з них мають найбільший вплив.

Такий підхід також полегшує порівняння різних стратегій або варіантів оптимізації. Замість аналізування п'яти показників окремо, можливо порівняти два зведені показники і оцінити, який з них краще відповідає цілям та вимогам.

Враховуючи лише два зведені показники, складність задачі оптимізації зводиться до більш зрозумілого та керованого процесу. Це допомагає зробити оптимальний вибір і прийняти обґрунтовані рішення щодо оптимізації.

Узагальнюючи, згрупувавши коефіцієнти в дві категорії, спрощується оптимізаційний процес, полегшується аналіз та порівняння, і звільняється час на те щоб зосередитися на ключових аспектах, що допомагає досягти кращих результатів.

У кінцевому результаті отримано два зведені показники, які відображатимуть вплив кожної групи коефіцієнтів на інтегральний показник (див. рис. 8). Це дозволяє зосередитися на головних аспектах оптимізації і приймати рішення, спрямовані на досягнення бажаного результату.

	A	B	C	D	E	F	G	H	I	J
1	id_car	K1	K2	K	DK1	DK2	Test			
2	1	0.85	0.605	1.455	0.58419243	0.41580756	1			
3	2	0.8	0.66	1.46	0.54794520	0.45205479	1			
4	3	0.47	0.528	0.998	0.47094188	0.52905811	1			
5	4	0.58	0.429	1.009	0.57482656	0.42517343	1			
6	5	0.82	0.682	1.502	0.54593874	0.45406125	1			
7	6	0.82	0.66	1.48	0.55405405	0.44594594	1			
8	7	0.82	0.693	1.513	0.54196959	0.45803040	1			
9	8	0.41	0.715	1.125	0.36444444	0.63555555	1			
10	9	0.53	0.737	1.267	0.41831097	0.58168902	1			
11	10	0.45	0.913	1.363	0.33015407	0.66984592	1			
12	11	0.55	1.144	1.694	0.32467532	0.67532467	1			
13	12	0.39	0.693	1.083	0.36011080	0.63988919	1			
14	13	0.74	0.792	1.532	0.48302872	0.51697127	1			
15	14	0.64	0.572	1.212	0.52805280	0.47194719	1			
16	15	0.54	0.682	1.222	0.44189852	0.55810147	1			
17	16	0.6	0.704	1.304	0.46012269	0.53987730	1			
18	17	0.76	0.616	1.376	0.55232558	0.44767441	1			
19	18	0.64	0.605	1.245	0.51405622	0.48594377	1			
20	19	0.77	0.671	1.441	0.53435114	0.46564885	1			
21	20	0.86	0.682	1.542	0.55771725	0.44228274	1			
22	21	0.58	0.572	1.152	0.50347222	0.49652777	1			
23	22	0.94	0.836	1.776	0.52927927	0.47072072	1			
24	23	0.49	0.627	1.117	0.43867502	0.56132497	1			
25	24	0.95	0.77	1.72	0.55232558	0.44767441	1			
26	25	0.62	0.704	1.324	0.46827794	0.53172205	1	Mean	Range	Std
27					0.48724588	0.51275411	1	0.5	0.02550823	0.01803704
28		0.95	1.144							

Рис. 8. Результати аналізу зведених показників

На основі аналізу доступних бібліотек Python і методів кластеризації було вирішено використати метод KMeans для кластеризації даних. Алгоритм K-means для кластеризації базується на розділенні набору даних на кластери. Він використовує підхід мінімізації суми квадратів відстаней між кожним об'єктом даних і центроїдом його відповідного кластера. Використання алгоритму K-means обґрунтовується простотою реалізації, швидкістю обчислень, ефективністю для задач кластеризації та стабільністю до шуму та випадкових варіацій у наборі даних. Однак, однією з проблем методу KMeans є необхідність передбачити кількість кластерів заздалегідь.

Для вирішення цієї проблеми було використано критерій Калінського-Харабасу, який є методом оцінки якості кластеризації. Його застосування полягає у обчисленні внутрішньокластерної дисперсії та зовнішньокластерної дисперсії.

Внутрішньокластерна дисперсія вимірює, наскільки схожі об'єкти є всередині кожного кластера. Максимальне зменшення внутрішньокластерної дисперсії свідчить про те, що об'єкти всередині кластерів добре схожі між собою.

Зовнішньокластерна дисперсія вимірює, наскільки різні кластери відокремлені один від одного. Максимізація зовнішньокластерної дисперсії означає, що кластери мають мінімальне перекриття та чітку відокремленість.

Оптимальне значення K, яке визначає найкращу кількість кластерів, досягається тоді, коли внутрішньокластерна дисперсія мінімальна, а зовнішньокластерна дисперсія максимальна.

Застосувавши критерій Калінського-Харабасу[10] до наших даних, було встановлено, що найбільш підходяща кількість кластерів для цієї задачі – 3 (див. рис. 9). Це означає, що можливо розділити дані на три групи або кластери, враховуючи їхні характеристики та взаємозв'язки між об'єктами.

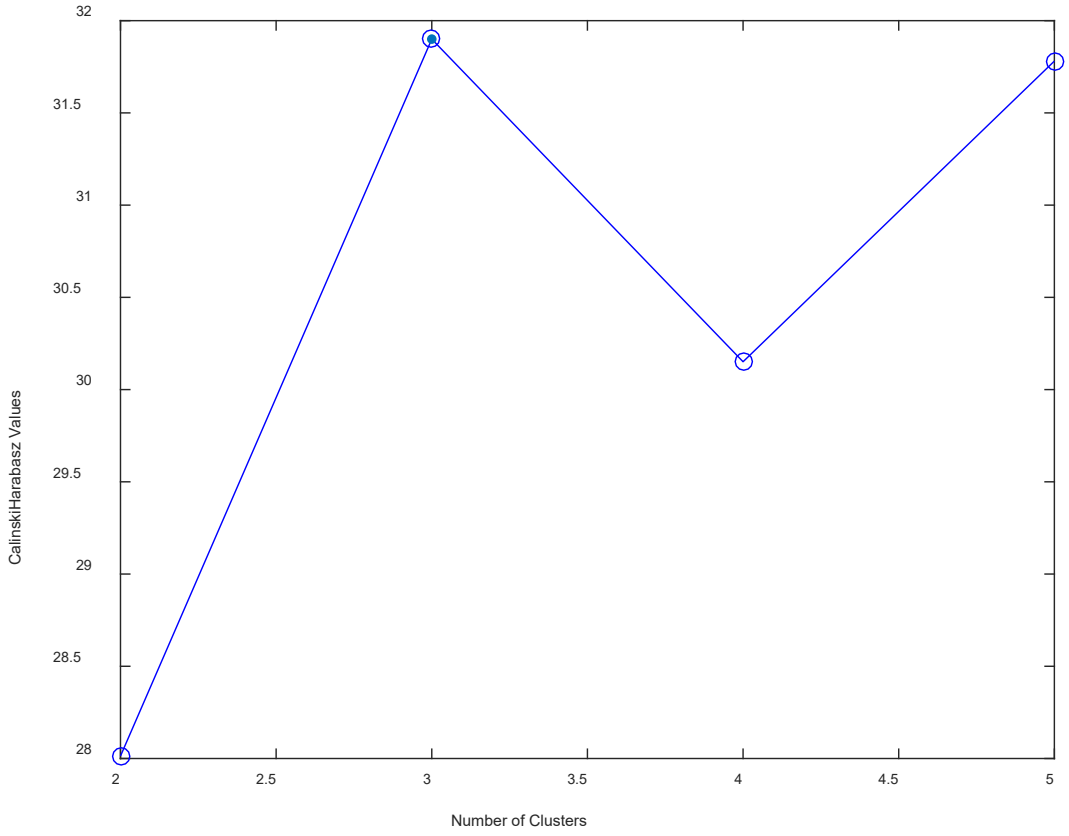


Рис. 9. Результати роботи алгоритму Калінські-Харабаса

Тепер коли були обрані метод для кластеризації, інформація про машини, коефіцієнти і оптимальна кількість кластерів, можливо застосувати метод KMeans до даних з трьома кластерами для подальшого аналізу або обробки інформації, що може допомогти зрозуміти особливості та структуру даних у кожному окремому кластері.

На першій картинці можна побачити розташування машин та області, в які вони входять за результатами кластеризації методом KMeans з трьома кластерами (див. рис. 10). Кожен кластер позначений окремим кольором, щоб візуально виділити їх.

Ця інформація дозволяє побачити, як дані розподілені по різних областях кластерів. Також можливо виявити, чи є в яких-небудь областях концентрація машин, чи можна виділити підгрупи машин з подібними характеристиками, або

якісь особливості, що характеризують кожен кластер.

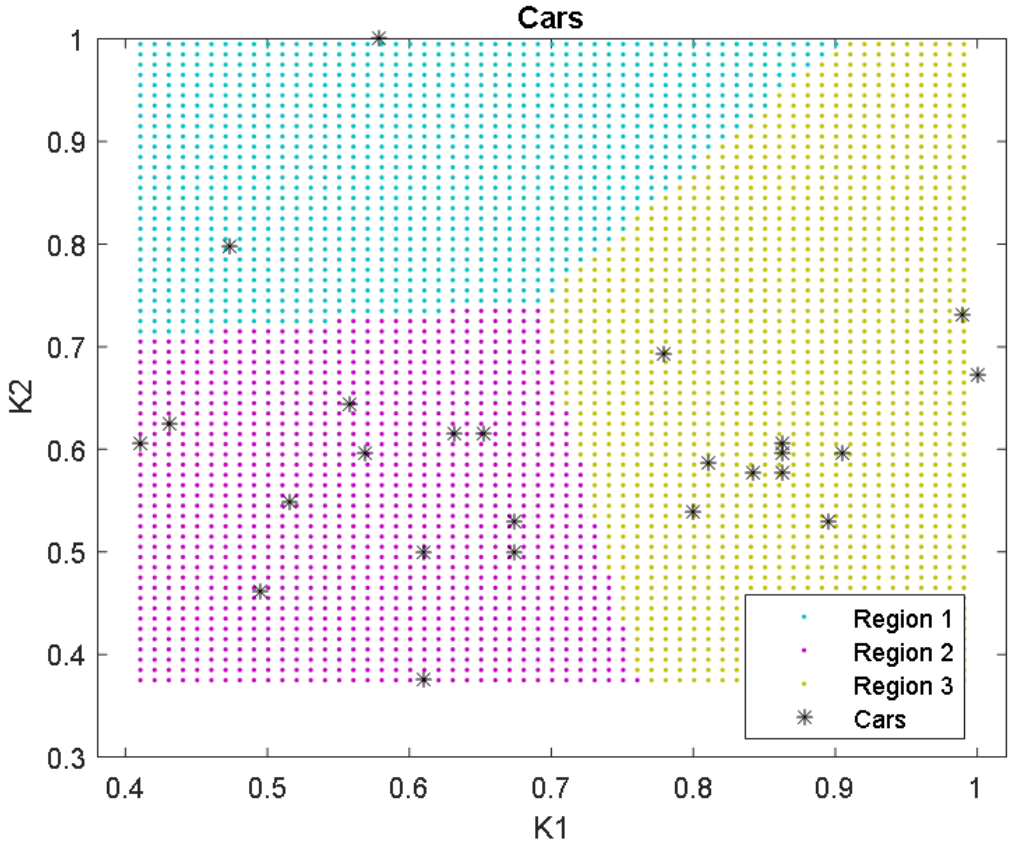


Рис. 10. Результати роботи алгоритму KMeans

Друга картинка демонструє територію кластерів і їх центри (див. рис. 11).

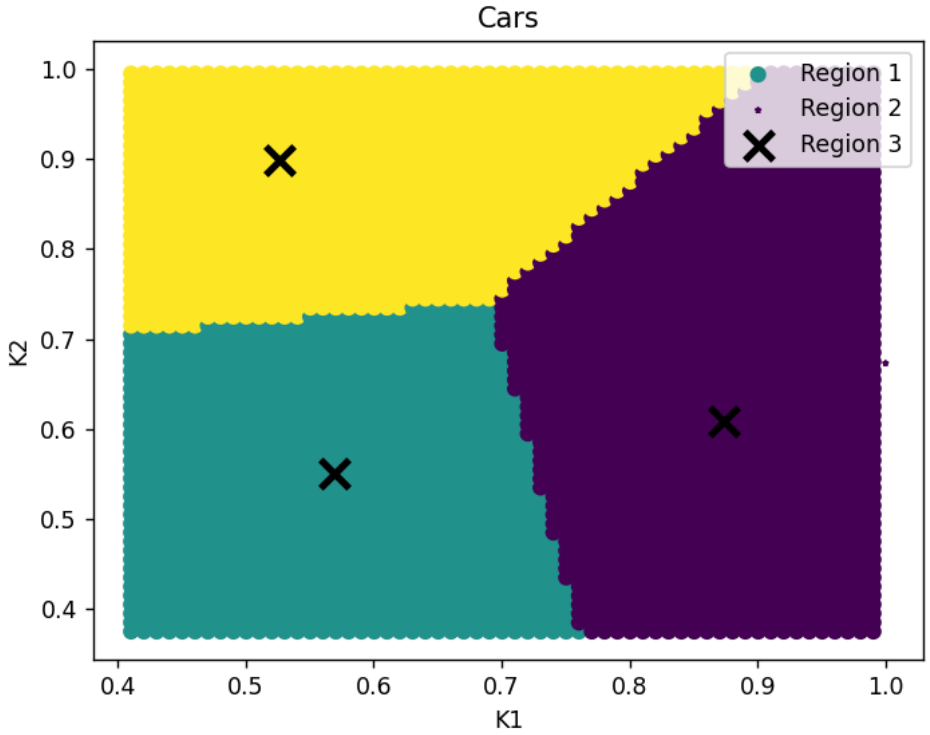


Рис. 11. Результати роботи алгоритму KMeans

Третя картинка додатково показує центри кластерів (див. рис. 12). Центри кластерів позначені спеціальними маркерами або символами, які вказують на їхнє місцезнаходження в просторі даних. Це дозволяє нам побачити, як центри розташовані щодо машин у кожному кластері. Ця інформація може бути корисною для подальшого аналізу, оскільки центри кластерів можуть вказувати на представницькі точки або характеристики кожного кластеру.

Разом ці три картинки надають візуальне уявлення про розташування машин у кластерах, а також їхні відносини з центрами кластерів. Це допомагає зрозуміти структуру даних, виділити групи або підгрупи з подібними характеристиками та здійснити подальший аналіз або вжити відповідні дії на основі цих візуальних висновків.

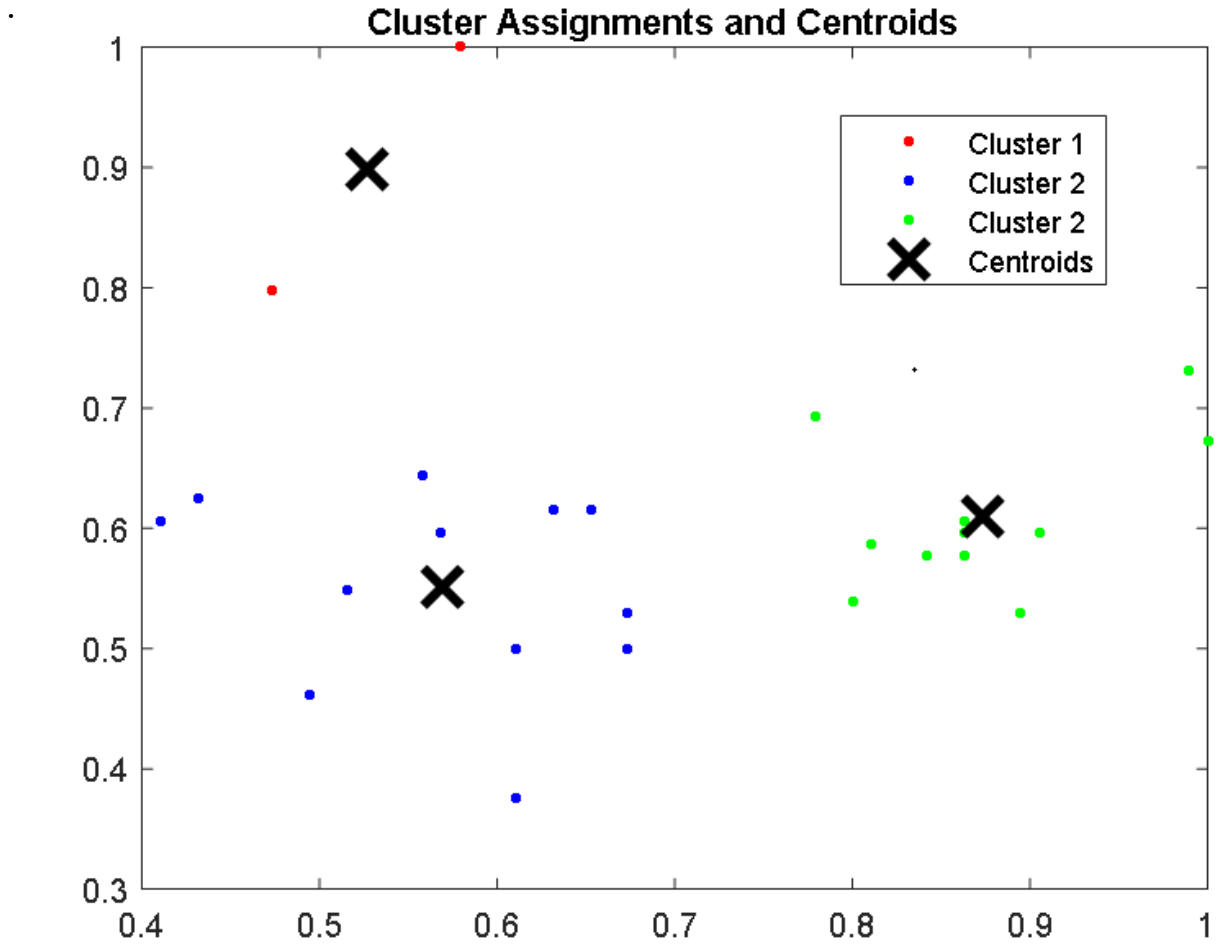


Рис. 12. Демонстрація кластерів і їх центрів

Самоорганізовані мережі Кохонена (SOM) є типом нейронних мереж, що використовуються для кластеризації і візуалізації даних. В ході свого навчання SOM аналізують характер розташування точок у вхідному просторі і намагаються зберегти топологічний порядок та метричну близькість вихідних даних на виході мережі. [9]

SOM можуть бути застосовані до даних будь-якої розмірності, але їх найбільш типове використання - це для візуалізації даних у двовимірному просторі. SOM складаються з групи нейронів (вузлів), розташованих на двовимірній сітці. Кожен нейрон має свій вектор ваг, який ініціалізується випадковим чином.

Під час навчання SOM здійснюється ітеративний процес, в якому випадково вибирається вхідний вектор даних, а потім знаходиться нейрон, вектор ваг якого найбільш близький до вхідного вектора. Цей нейрон і його сусіди на сітці оновлюють свої ваги, що дозволяє нейронам адаптуватися до структури даних. Поступово в процесі навчання сусідні нейрони на сітці постають захоплювати схожі дані, утворюючи кластери.

Одна з переваг SOM полягає в тому, що вони можуть зберігати топологічну інформацію про дані. Це означає, що близькі у вхідному просторі дані будуть мати схоже місцезнаходження на сітці SOM, що допомагає візуалізувати структуру даних та виявляти кластери.

Таким чином, використання самоорганізованих мереж Кохонена (SOM) в кластеризації даних дозволяє досягти топологічного порядку та метричної близькості між векторами, що може бути корисним для подальшого аналізу та візуалізації даних.

2.7. Використані технології та обґрунтування вибраного інструментарію.

Для розробки модуля були використані наступні технології та інструменти:

1. Мова програмування: Python - це популярна мова програмування з великим спектром бібліотек і фреймворків для обробки даних та машинного навчання. Вибір Python обґрунтовується його простотою використання, широким співтовариством розробників і наявністю потужних інструментів для роботи з даними.
2. Бібліотека машинного навчання: scikit-learn - це потужна бібліотека для машинного навчання в Python. Вона надає реалізації різних алгоритмів класифікації, регресії, кластеризації та багато іншого. Вибір scikit-learn обґрунтовується його широким функціоналом, документацією та ефективністю в роботі з великими обсягами даних.
3. Бібліотека для візуалізації даних: Matplotlib - це бібліотека для створення візуалізаційних графіків і діаграм в Python. Вона дозволяє зручно відображати дані, виявляти закономірності і структуру в наборі даних. Вибір Matplotlib обґрунтовується його широкими можливостями для візуалізації даних та зручним інтерфейсом.

Обґрунтування вибору цих технологій та інструментів полягає в їхній популярності, широкому співтоваристві користувачів та розробників, наявності документації та придатності для вирішення поставлених завдань у проекті. Використання цього інструментарію дозволить ефективно реалізувати функціональність модуля, проводити аналіз даних та візуалізацію результатів.

Але усе це являє з себе “Модель”, що є лише однією частиною шаблону MVC. Model -View-Controller (MVC) - це архітектурний шаблон, що розділяє додаток на три основні логічні компоненти: модель, представлення та контролер (див. рис. 13). Кожен із цих компонентів створений для обробки конкретних аспектів розробки програми.[8]

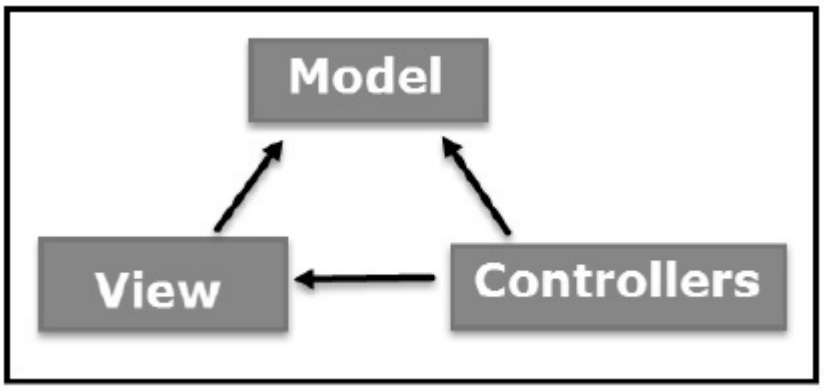


Рис. 13. Діаграма роботи шаблону MVC

Використовуючи нотацію UML можна описати цей шаблон так (див. рис. 14).

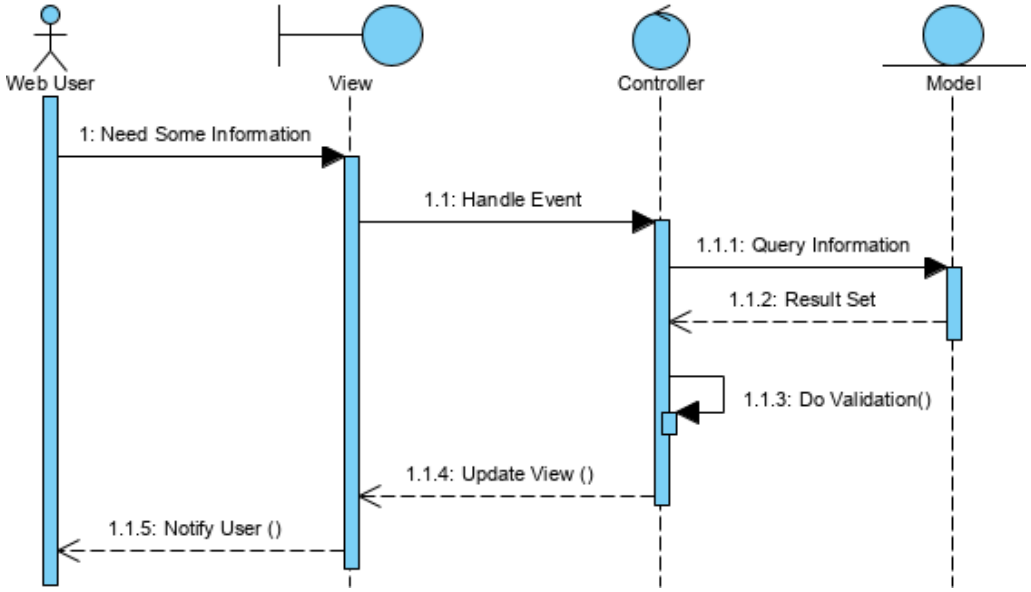


Рис. 14. UML нотація для шаблону MVC

Компонент "Модель" відповідає за логіку, пов'язану з даними, з якими користувач працює. Це можуть бути дані, які передаються між компонентами "Вид" і "Контролер", або будь-які інші дані, пов'язані з бізнес-логікою. Наприклад, об'єкт "Клієнт" буде витягувати інформацію про клієнта з бази даних, маніпулювати нею і оновлювати дані назад у базу даних або використовувати їх для відображення даних. Приклад роботи шаблону у програмному модулі дивіться на картинці нижче (див. рис. 15).

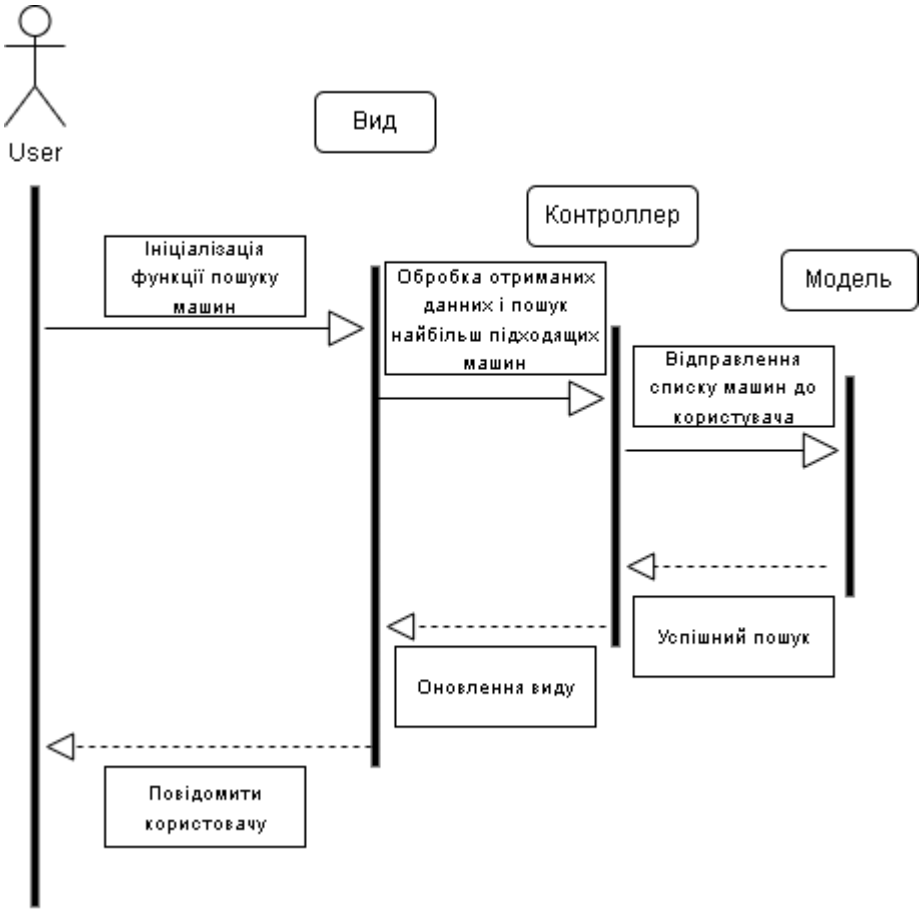


Рис. 15. Приклад використання шаблону MVC у програмі

Компонент "Вид" використовується для всієї логіки користувацького інтерфейсу додатку. Наприклад, представлення "Клієнт" включатиме всі компоненти користувацького інтерфейсу, такі як текстові поля, розкриваючіся списки тощо, з якими взаємодіє кінцевий користувач.

Контролери виступають як інтерфейс між компонентами моделі і представленням для обробки всієї бізнес-логіки та вхідних запитів, маніпулювання даними з використанням компонента моделі та взаємодії з представленнями для виведення кінцевого результату. Наприклад, контролер клієнта буде обробляти всі взаємодії та вхідні дані з представлення клієнта та оновлювати базу даних за допомогою моделі клієнта. Той же контролер буде використовуватися для перегляду даних клієнта.

Цей шаблон дозволяє розділити програму на логічні компоненти, що спрощує розробку, підтримку та розширення програмного забезпечення. Він

покращує перевикористання коду, робить програму більш модульною та полегшує зміни в інтерфейсі користувача без впливу на логіку даних.

Для реалізації компонента контролерів була використана бібліотека Flask. Flask є популярною бібліотекою Python для веб-розробки. Однією з основних переваг використання Flask є те, що вона проста у використанні і має лаконічний синтаксис. Вона дозволяє легко створювати веб-додатки і API, що реалізують бізнес-логіку та оброблюють вхідні запити.

Flask надає необхідні функціональність для реалізації маршрутів (URL-шляхів), обробки HTTP-запитів і відповідей. За допомогою Flask можна визначити різні функції-обробники для кожного маршруту, що відповідають на певні запити і виконують необхідні дії, пов'язані з бізнес-логікою програми. Flask також надає можливості для роботи з шаблонами, формами, автентифікацією та іншими аспектами веб-розробки.

Для компонента "Вид" були використані HTML і CSS. Загалом, використання Flask, HTML і CSS дозволяє створити ефективні контролери та забезпечити зрозумілий та привабливий користувацький інтерфейс для веб-додатків.

Також MVC шаблон дозволяє використовувати RESTful api. Архітектурний стиль REST (Representational State Transfer) використовується для забезпечення стандартів взаємодії між комп'ютерними системами в Інтернеті. Він спрощує взаємодію між системами, роблячи їх незалежними одна від одної (див. рис. 16). У стилі REST реалізація клієнта і реалізація сервера можуть працювати незалежно одна від одної, не знаючи про існування іншої сторони. [3]

Це означає, що код на стороні клієнта може бути змінений в будь-який момент без впливу на роботу сервера, а код на стороні сервера може бути змінений без впливу на роботу клієнта. Поки кожна сторона знає, який формат повідомлень надсилати іншій стороні, вони можуть залишатися модульними та незалежними. Розділяючи проблеми користувацького інтерфейсу від проблем зберігання даних, ми покращуємо гнучкість інтерфейсу на різних платформах і

поліпшуємо масштабованість шляхом спрощення серверних компонентів. Крім того, розподіленість дозволяє кожному компоненту розвиватись незалежно.

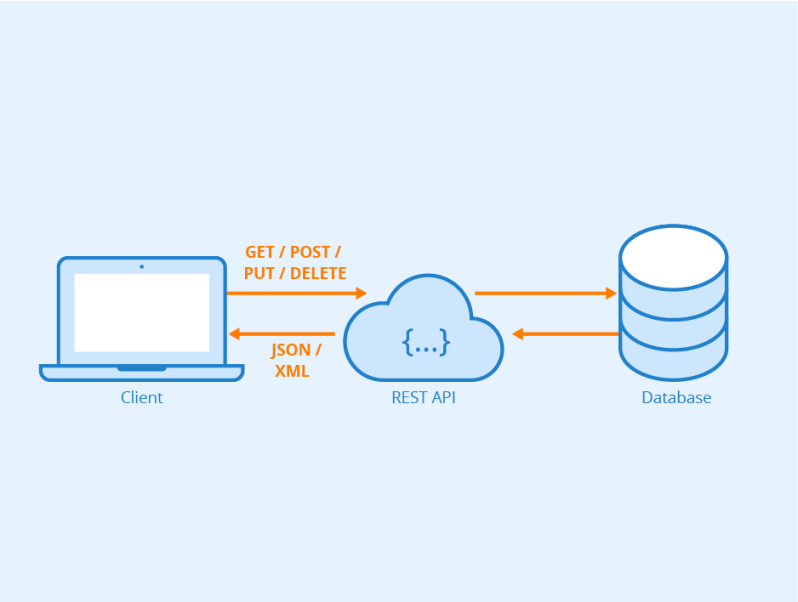


Рис. 16. Діаграма роботи архітектурного стилю REST

У REST-подібних API запити повинні містити шлях до ресурсу, над яким має виконуватися операція. У RESTful API шляхи повинні бути розроблені так, щоб допомагати клієнту зрозуміти, що відбувається. Це допомагає покращити зрозумілість інтерфейсу та спрощує взаємодію між клієнтом і сервером.

Простота інформативність інтерфейсу користувача є ключовими якостями з кількох причин. Простота інтерфейсу дозволяє користувачам легко орієнтуватися та використовувати програму без необхідності вивчати складні інструкції. Легкість використання і інтуїтивність інтерфейсу допомагають користувачам швидше виконувати завдання та досягати мети, що покращує їх задоволеність від використання програми.

Інформативність інтерфейсу означає, що користувачам надається достатня кількість потрібної інформації, щоб вони могли приймати інформовані рішення. Інтерфейс повинен надавати користувачам зрозумілі та чіткі повідомлення, що допоможуть їм розуміти поточний стан системи, виконувати необхідні дії та отримувати відповідну зворотну інформацію.

Комбінація простоти і інформативності забезпечує ефективність використання програми, допомагає уникнути помилок та недорозумінь, та сприяє позитивному користувацькому досвіду.

Загалом у користувача буде 2 екрани:

1. З формою для вводу характеристик які цікавлять користувача (див. рис. 17).

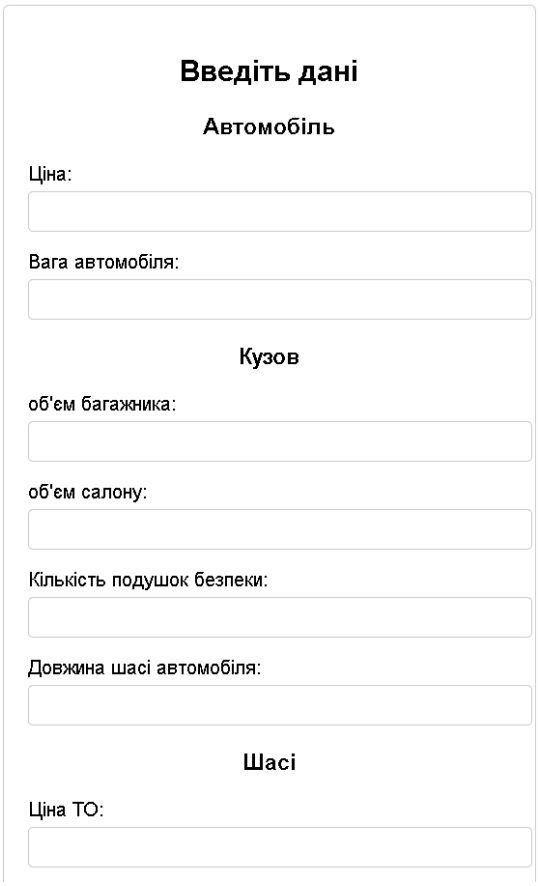


Рис. 17. Форма для введення данних користувачем

2. З результатами, тобто машини які найбільш підходять введеними користувачем характеристикам (див. рис. 18).

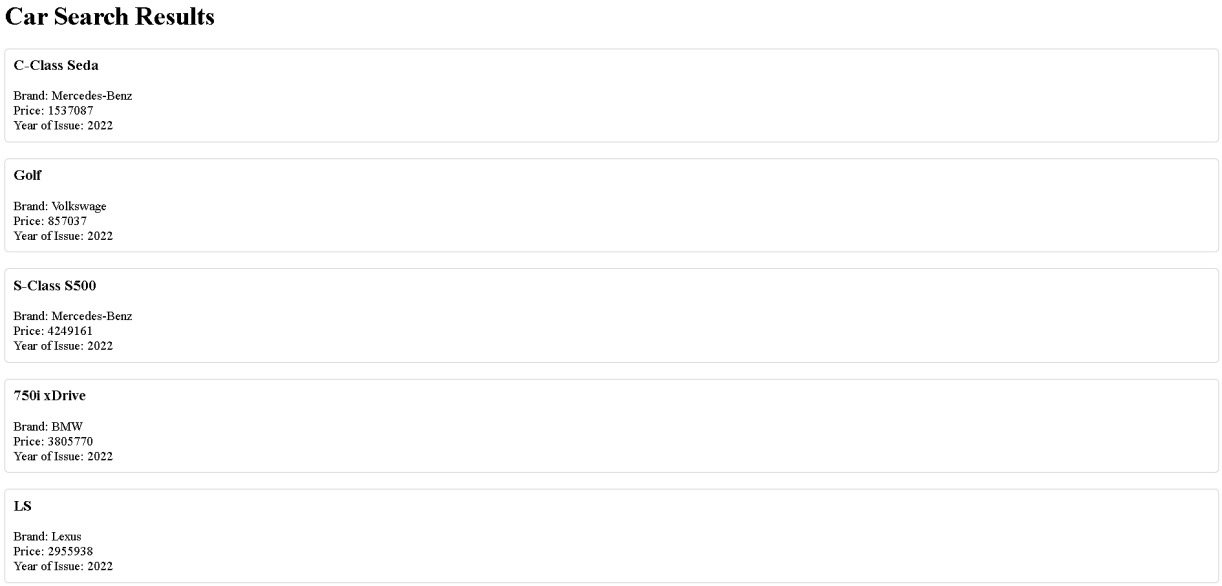


Рис. 18. Список рекомендованих машин

2.8. Результати тестування.

Приклад 1:

При введені наступних даних програма видає результат, який продемонстровано на картинці нижче (див. рис. 19).

Ціна: 50000

Вага автомобіля: 1500

об'єм багажника: 400

об'єм салону: 2500

Кількість подушок безпеки: 6

Довжина шасі автомобіля: 3

Ціна ТО: 200

Гарантійний пробіг: 10000

Гальмівний шлях: 40

Довжина бази автомобіля: 2

Витрата палива на 100 км: 9

Мінімальна витрата палива на 100 км: 6

Максимальна швидкість: 200

Розгін до 100 км/ч: 8

Густина палива: 1

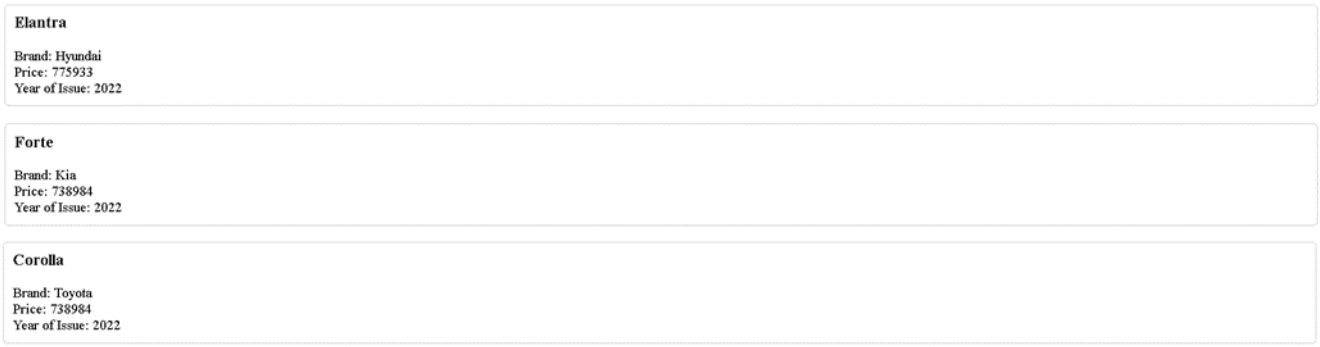


Рис. 19. Список рекомендованих машин(Приклад 1)

В даному випадку до введеної користувачем інформації найближчим кластером став 2. А найбільш підходящі машини це Hyundai Elantra, Kia Forte, Toyota Corolla.

Приклад 2:

При введені наступних даних програма видає результат, який продемонстровано на картинці нижче (див. рис. 20).

Ціна: 1000000

Вага автомобіля: 2000

Об'єм багажника: 300

Об'єм салону: 2200

Кількість подушок безпеки: 10

Довжина шасі автомобіля: 3

Ціна ТО: 500

Гарантійний пробіг: 15000

Гальмівний шлях: 35

Довжина бази автомобіля: 2

Витрата палива на 100 км: 10

Мінімальна витрата палива на 100 км: 8

Максимальна швидкість: 330

Розгін до 100 км/год: 4

Густина палива: 1

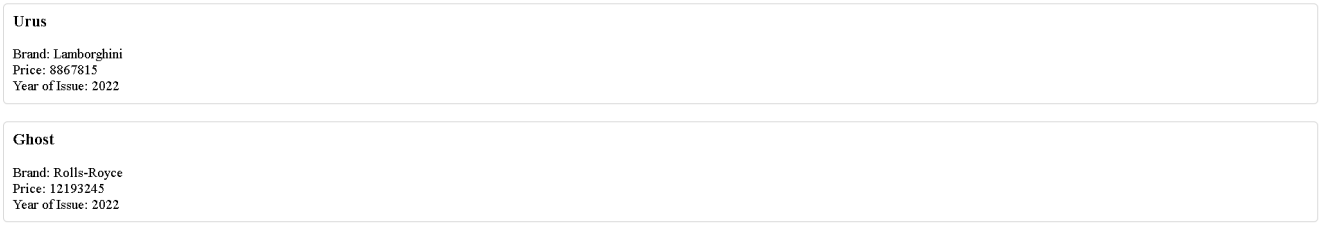


Рис. 20. Список рекомендованих машин(Приклад 2)

В даному випадку до введеної користувачем інформації найближчим кластером став 3. А найбільш підходящі машини це Lamborghini Urus і Rolls-Royce Ghost.

ВИСНОВКИ

У даній роботі було розроблено модуль, який здійснює аналіз і кластеризацію даних автомобіля. Для досягнення цієї мети було проведено дослідження і вибір відповідних алгоритмів, розроблено відповідну інформаційну модель та реалізовано алгоритми інтерфейсу користувача.

Загальною метою розробки модуля було створення інформаційної системи, яка забезпечує аналіз та кластеризацію даних автомобіля. Результатом роботи є реалізований модуль, який дозволяє користувачеві здійснювати кластерний аналіз даних автомобіля, отримувати результати візуалізації та здійснювати подальший аналіз отриманих кластерів.

Отже, розроблений модуль є потужним інструментом для аналізу та кластеризації даних автомобіля, що відкриває широкі можливості для виявлення внутрішніх зв'язків та структури даних, а також допомагає в зробити об'єктивні висновки на основі групування даних за схожими ознаками.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Scikit-learn Machine Learning in Python [Електронний ресурс]. URL: https://scikit-learn.org/stable/modules/clustering.html

2. Benchmarking Performance and Scaling of Python Clustering Algorithms [Електронний ресурс]. URL: https://hdbscan.readthedocs.io/en/latest/performance_and_scalability.html
3. What is REST? [Електронний ресурс]. URL: https://www.codecademy.com/article/what-is-rest
4. Understanding the Basics of REST APIs [Електронний ресурс]. URL: https://www.astera.com/type/blog/rest-api-definition/
5. Бажинова Т.О. ТЕОРЕТИЧНЕ ОБҐРУНТУВАННЯ ОЦІНКИ ЯКОСТІ ЛЕГКОВИХ АВТОМОБІЛІВ [Електронний ресурс]. URL: https://dspace.khadi.kharkov.ua/dspace/handle/123456789/1931
6. Бажинова Т.О. Розробка системи якості легкових автомобілів [Електронний ресурс]. URL: https://dspace.khadi.kharkov.ua/dspace/handle/123456789/5646
7. Оцінка автомобілів на етапі експлуатації зазвичай включає аналіз показників якості та стану автомобіля [Електронний ресурс]. URL: https://dspace.khadi.kharkov.ua/dspace/handle/123456789/2295
8. MVC Framework – Introduction [Електронний ресурс]. URL: https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm
9. Self Organizing Maps [Електронний ресурс]. URL: https://medium.com/@abhinavr8/self-organizing-maps-ff5853a118d4
10. Calinski-Harabasz Evaluation [Електронний ресурс]. URL: https://www.mathworks.com/help/stats/clustering.evaluation.calinskiharabaszevaluation.html
11. Товарна рекомендаційна система для інтернет-магазину, кластеризація товарів за схожістю [Електронний ресурс]. URL: https://uxprice.com/blog/ru/all-articles-ru/tovarnaja-rekomendatelnaja-sistema-dlja-internet-magazina-klasterizacija-tovarov-po-shozhesti-kejs-po-vnedreniju/

12. Deep Dive into Netflix’s Recommender System [Електронний ресурс]. URL: https://towardsdatascience.com/deep-dive-into-netflixs-recommender-system-341806ae3b48
13. Machine Learning for Recommender systems — Part 1 (algorithms, evaluation and cold start) [Електронний ресурс]. URL: https://medium.com/recombee-blog/machine-learning-for-recommender-systems-part-1-algorithms-evaluation-and-cold-start-6f696683d0ed