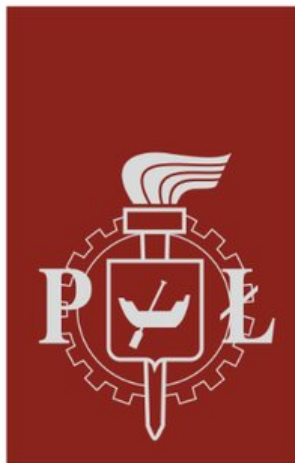


Politechnika Łódzka

Wydział Elektrotechniki Elektroniki Informatyki i Automatyki



sem, zimowy, r ak. 2024/2025

Sprawozdanie z projektu BigData „Predykcja cen samochodów używanych”



Mateusz Grzybek 240678

Kamil Młynarczyk 240757

17 grudnia 2024

Spis treści

1	Wstęp	2
1.1	Założenia projektowe	2
1.2	Komponenty	2
1.3	Konteneryzacja	2
1.4	Sposób uruchomienia	3
2	Diagramy	4
2.1	Diagram przypadków użycia	4
2.2	Diagram sekwencji zdarzeń	5
3	Aplikacja kliencka	6
3.1	Opis	6
3.2	Technologie	6
3.3	Widoki aplikacji	7
3.3.1	Strona	7
3.3.2	Okno z ceną	8
3.3.3	Okno z błędem	8
4	Komponent pośredniczący	9
4.1	Opis	9
4.2	Technologie	9
5	Komponent komunikacyjny	10
5.1	Opis	10
5.2	Technologie	10
6	Przygotowanie danych	11
6.1	Opis	11
6.2	Wizualizacja danych	11
6.3	Puste pola	13
6.4	Zestaw danych z wartościami odstającymi	14
6.5	Wykrywanie i usuwanie wartości odstających za pomocą metody IQR	14
6.6	Zestaw danych bez wartości odstających	15
6.7	Wyodrębnienie znaczących informacji o silniku	15
6.8	Faktoryzacja danych	16
7	Serwis predykcyjny	18
7.1	Opis	18
7.2	Technologie	18
7.3	Wybór modelu	18

Rozdział 1

Wstęp

1.1 Założenia projektowe

Celem projektu jest zaimplementowanie aplikacji webowej pozwalającej użytkownikom na predykcję ceny używanego samochodu na podstawie dostarczonego przez niego zestawu cech. Tematyka projektu daje możliwość wykorzystania różnorodnych technologii z dziedziny uczenia maszynowego, rozwoju aplikacji webowych, komunikacji pomiędzy serwisami, architektury oprogramowania oraz bierania i przetwarzania danych. W celu zrealizowania przewidywanych funkcjonalności, aplikacja została podzielona na cztery komponenty, każdy z nich odpowiedzialny za realizację innego aspektu aplikacji.

1.2 Komponenty

- Aplikacja kliencka — Interfejs graficzny użytkownika.
- Pośrednik — Komponent pośredniczący w komunikacji pomiędzy aplikacją kliencką i serwisem predykcyjnym
- Komponent komunikacyjny — Komponent zawierający szyny danych, które są wykorzystywane do dostarczania i odbierania informacji od serwisu predykcyjnego
- Serwis predykcyjny — Komponent dokonujący predykcji na podstawie dostarczonych danych, z wykorzystaniem nauczonego modelu.

1.3 Konteneryzacja

Wszystkie komponenty zostały skonteneryzowane za pomocą narzędzi **Docker**¹ i **Docker Compose**², co pozwala na uruchomienie projektu bez konieczności dodatkowej konfiguracji. **Obrazy**³ **kontenerów**⁴ dla aplikacji klienckiej oraz pośrednika zostały zdefiniowane za pomocą plików **Dockerfile**⁵, natomiast dla komponentu komunikacyjnego wykorzystano gotowe obrazy Apache Kafka i Zookeeper z rejestru Docker.io.

¹Narzędzie do tworzenia, uruchamiania i zarządzania aplikacjami w izolowanych środowiskach zwanych kontenerami.

²Narzędzie usprawniające zarządzanie wieloma kontenerami jednocześnie.

³Gotowy do uruchomienia szablon do tworzenia kontenerów, zawierający system plików, aplikację i jej zależności.

⁴Lekkie, izolowane środowisko uruchomieniowe, które zawiera aplikację wraz z jej zależnościami.

⁵Plik tekstowy zawierający zestaw instrukcji do zbudowania obrazu Docker.

1.4 Sposób uruchomienia

1. Zainstalować Docker i Docker Compose.
2. Aplikacja kliencka — Otworzyć katalog `frontend` i wewnątrz niego uruchomić skrypt `run.sh` lub uruchomić ręcznie komendy w terminalu. Wyłączenia kontenera można dokonać skryptem `clean.sh`.

- Skrypt `run.sh`:

```
#!/bin/bash

# Builds docker image from local Dockerfile
# and sets image name to "frontend-image"
docker build -t frontend-image .

# Creates and runs container with name "frontend" from
# frontend-image
# in detached mode and container-host port mapping to
# 9091
docker run --name frontend -d -p 9091:9091 frontend-image
```

- Skrypt `clean.sh`:

```
#!/bin/bash

# Stops and removes the "frontend" container
docker stop frontend
docker rm frontend
```

3. Pozostałe komponenty — Otworzyć katalog projektu i uruchomić komendę `docker compose up`. Do wyłączenia kontenerów należy użyć komendy `docker compose down`.

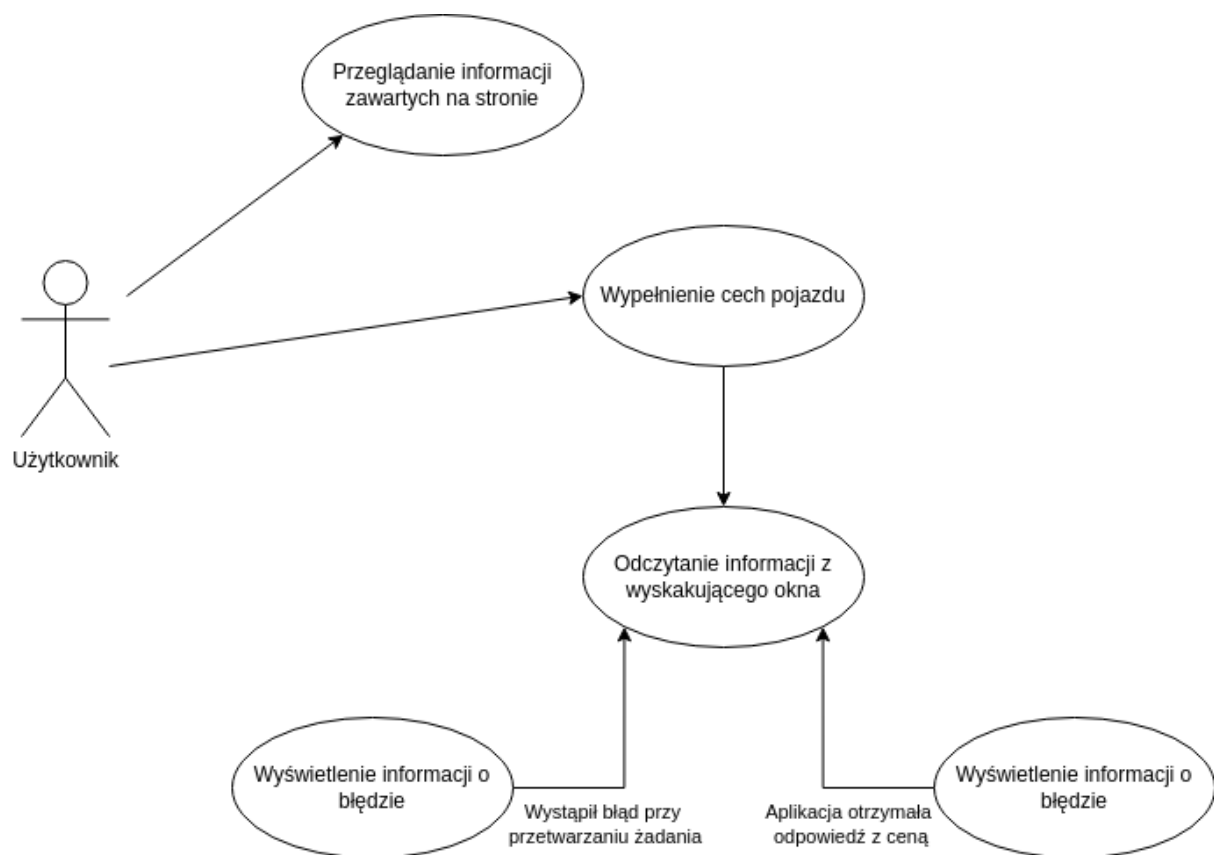
```
# Run containers
docker compose up

# Stop containers
docker compose down
```

Rozdział 2

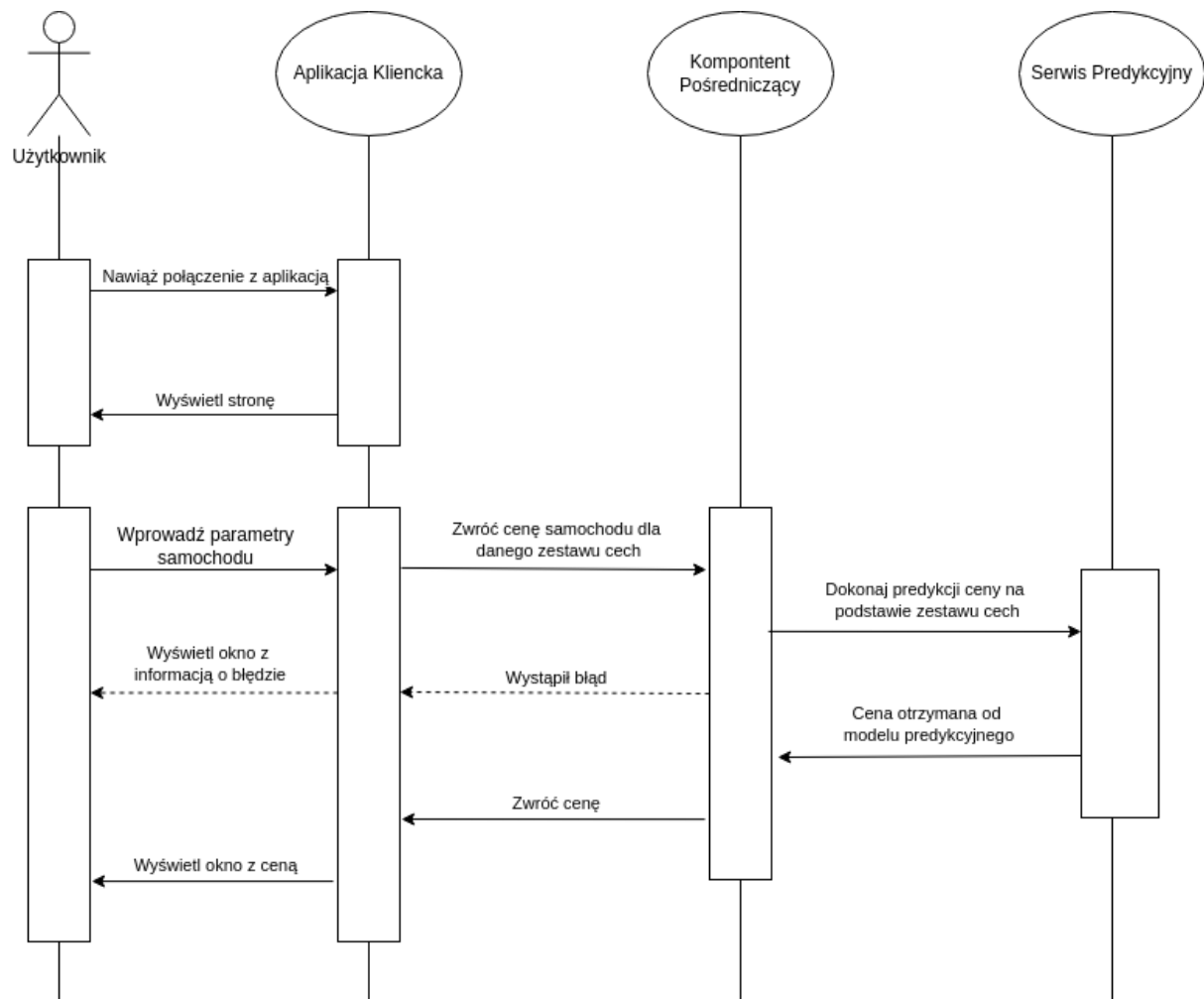
Diagramy

2.1 Diagram przypadków użycia



Rysunek 2.1: Przebieg interakcji użytkownika z aplikacją

2.2 Diagram sekwencji zdarzeń



Rysunek 2.2: Przebieg operacji komponentów i działań użytkownika podczas procesu predykcji ceny samochodu

Rozdział 3

Aplikacja kliencka

3.1 Opis

Aplikacja kliencka stanowi pojedynczą stronę dostępną za pośrednictwem przeglądarki, udostępnianą pod adresem **localhost**¹, na porcie **9091**. Strona zawiera informacje związane z aplikacją oraz pola do wprowadzania wartości, na podstawie których następnie dokonywana jest predykcja ceny samochodu. Aplikacja łączy się z komponentem middleware za pośrednictwem protokołu **HTTP**² w architekturze **REST**³.

3.2 Technologie

- React — Framework JavaScript do tworzenia interfejsów użytkownika w oparciu o komponenty.
- HTML — Język znaczników do tworzenia struktury strony internetowej.
- CSS — Język stylów wykorzystywany do definiowania wyglądu stron internetowych.
- JavaScript — Język programowania wykorzystywany do tworzenia dynamicznych i interaktywnych elementów stron internetowych.
- Axios — Biblioteka JavaScript służąca do wykonywania zapytań HTTP.
- Vite — Narzędzie do budowania i uruchamiania aplikacji front-endowych.

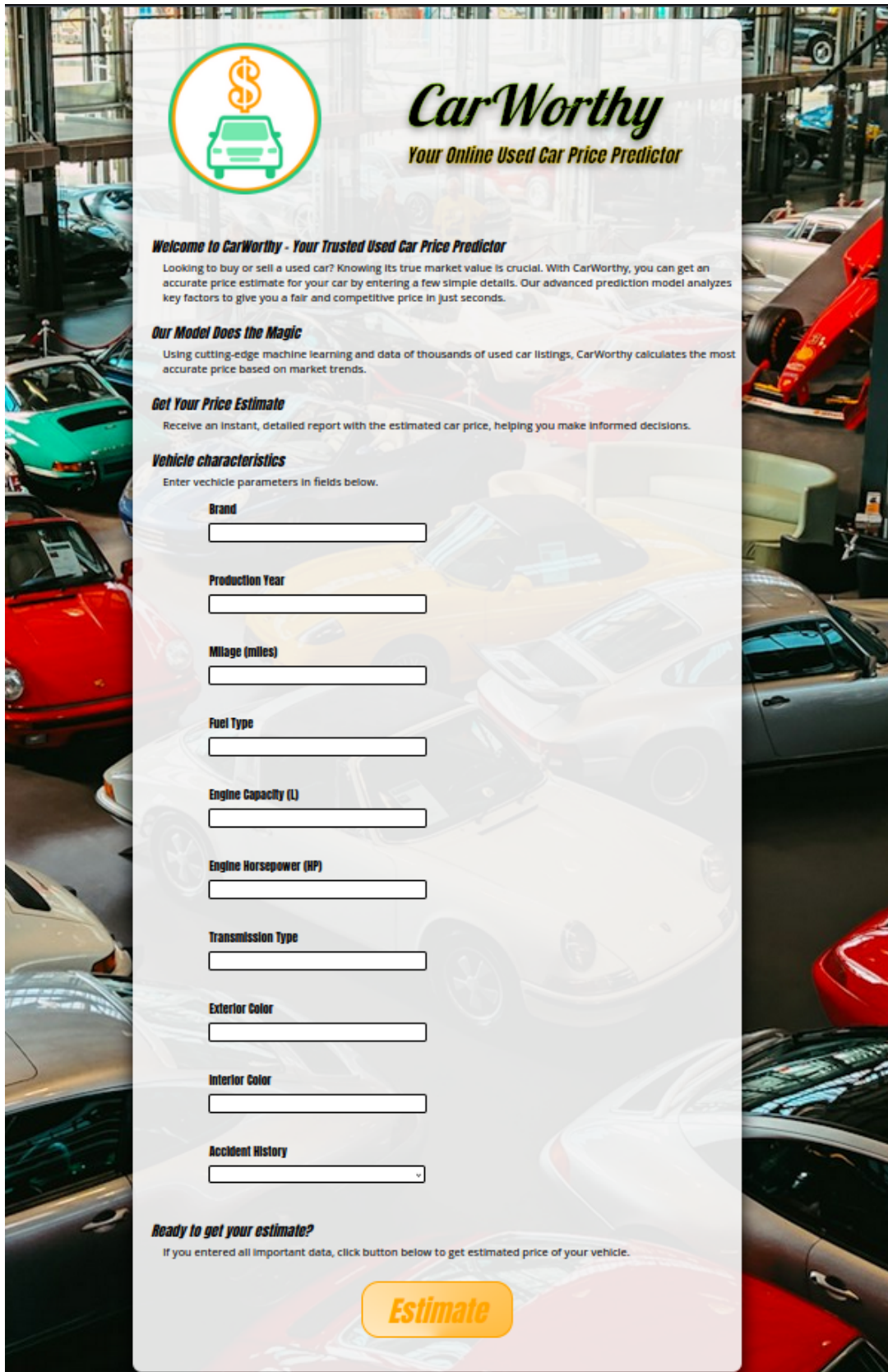
¹loopback address — adres pętli zwrotnej, który jest wykorzystywany do komunikacji urządzenia z samym sobą.

²HyperText Transfer Protocol — protokół komunikacyjny używany do przesyłania danych w sieci.

³Representational State Transfer — architektura komunikacji oparta o protokół HTTP definiujący sposoby identyfikacji i manipulacji zasobami za pomocą zapytań HTTP.

3.3 Widoki aplikacji

3.3.1 Strona



The screenshot displays the CarWorthy website, which is an online used car price predictor. The page features a light gray background with a faint image of a car dealership. At the top left, there is a logo consisting of a green circle with a white car icon and a dollar sign above it. To the right of the logo, the text "CarWorthy" is written in a large, bold, black font, and "Your Online Used Car Price Predictor" is written in a smaller, bold, black font below it.

Below the header, there is a section titled "Welcome to CarWorthy - Your Trusted Used Car Price Predictor". This section contains a paragraph of text: "Looking to buy or sell a used car? Knowing its true market value is crucial. With CarWorthy, you can get an accurate price estimate for your car by entering a few simple details. Our advanced prediction model analyzes key factors to give you a fair and competitive price in just seconds."

Next is a section titled "Our Model Does the Magic". It contains a paragraph: "Using cutting-edge machine learning and data of thousands of used car listings, CarWorthy calculates the most accurate price based on market trends."

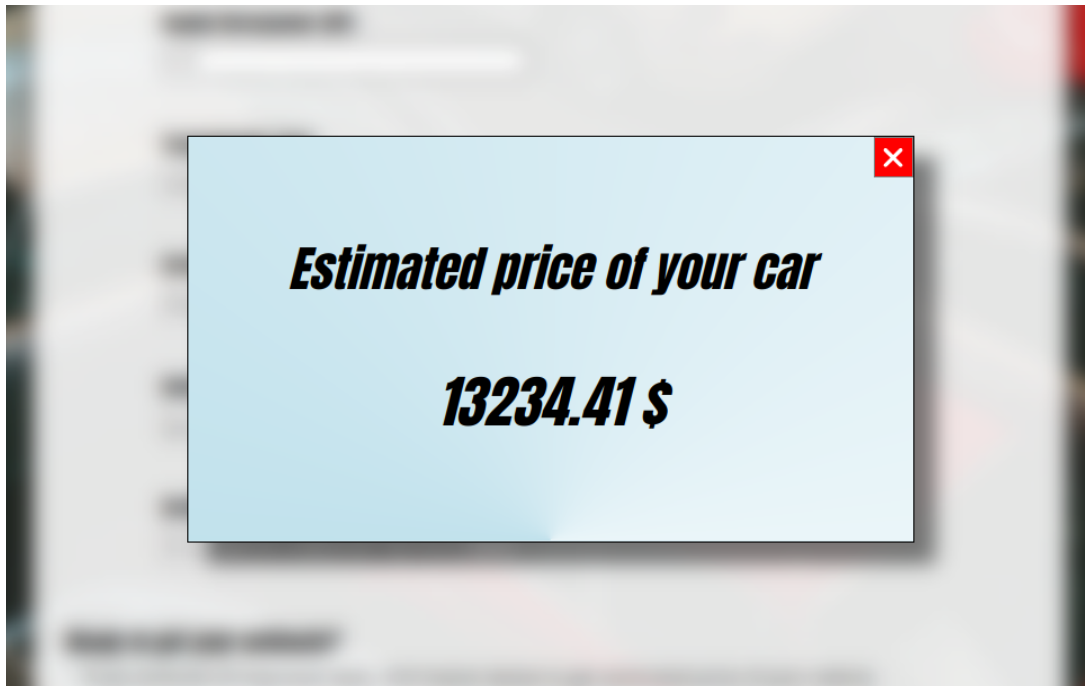
Below that is a section titled "Get Your Price Estimate". It contains a paragraph: "Receive an instant, detailed report with the estimated car price, helping you make informed decisions."

The main part of the form is titled "Vehicle characteristics". It starts with the instruction "Enter vehicle parameters in fields below." and then lists several fields with corresponding labels: "Brand", "Production Year", "Mileage (miles)", "Fuel Type", "Engine Capacity (L)", "Engine Horsepower (HP)", "Transmission Type", "Exterior Color", "Interior Color", and "Accident History". Each label is followed by a white input field with a black border. The "Accident History" field has a small downward arrow on the right side, indicating it is a dropdown menu.

At the bottom of the form, there is a section titled "Ready to get your estimate?". It contains a paragraph: "If you entered all important data, click button below to get estimated price of your vehicle." Below this text is a large, orange, rounded rectangular button with the word "Estimate" written in white, bold, italicized font.

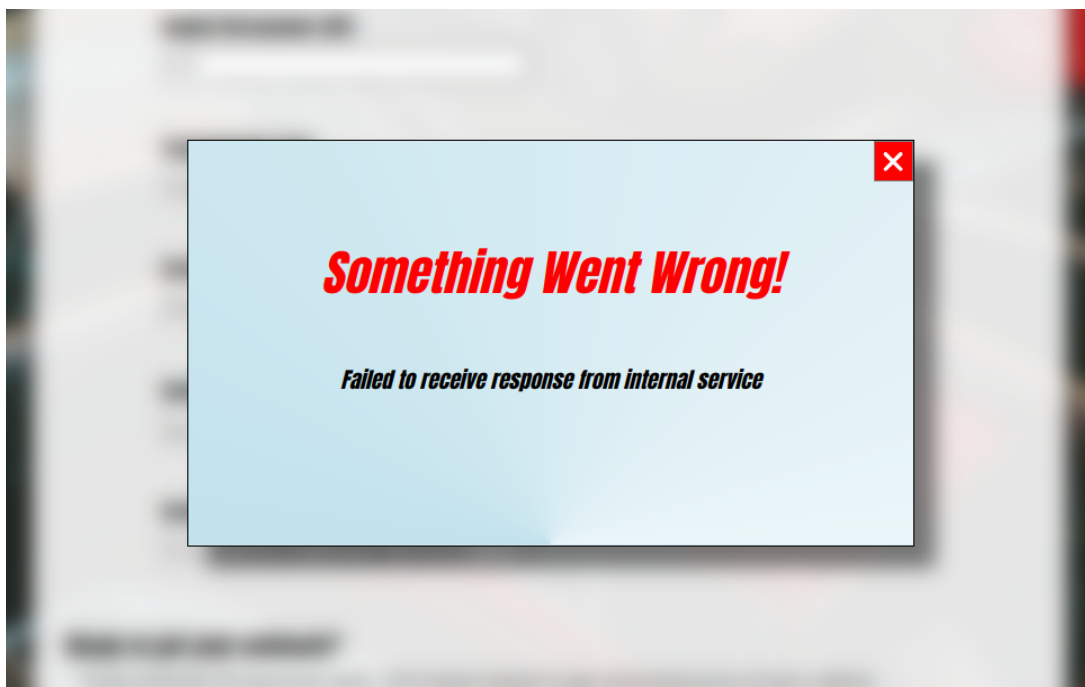
Rysunek 3.1: Widok strony

3.3.2 Okno z ceną



Rysunek 3.2: Widok okna z ceną

3.3.3 Okno z błędem



Rysunek 3.3: Widok okna z błędem

Rozdział 4

Komponent pośredniczący

4.1 Opis

Komponent pośredniczący pełni rolę pośrednika pomiędzy aplikacją kliencką i serwisem predykcyjnym. Otrzymywane od **frontendu**¹ dane w formie **JSON**² są w tym komponencie przetwarzane na wiadomości w formacie odpowiadającym wejściu modelu, z uwzględnieniem procesu **kodowania liczbowego**³ pól. Otrzymane w tym procesie wiadomości zapisywane są na **temat**⁴ wejściowy Kafki. Pośrednik jest również odpowiedzialny za odczytywanie danych z tematu wyjściowego i przekazywanie uzyskanych z nich informacji do klienta.

4.2 Technologie

- Java — Obiektowy język programowania.
- SpringBoot — Framework dla języka Java nastawiony na wytwarzanie aplikacji webowych i mikroservisów
- Gradle — Narzędzie do automatyzacji budowania projektów.

¹Część aplikacji, z którą użytkownik wchodzi w bezpośrednią interakcję, w tym wszystko co widzi oraz elementy wizualne i interaktywne.

²JavaScript Object Notation — format danych zapewniający kompaktowe rozmiary i jest czytelny dla ludzi i maszyn.

³Technika zamiany wartości danych tekstowych na wartości liczbowe, poprzez przypisanie unikalnej liczby każdej unikalnej wartości tekstowej.

⁴Podstawowy komponent Apache Kafka służący do kategoryzacji napływających wiadomości.

Rozdział 5

Komponent komunikacyjny

5.1 Opis

Komponent komunikacyjny odpowiedzialny jest za transport danych pomiędzy komponentem pośredniczącym i serwisem predykcyjnym. Wykorzystuje w tym celu skonteneryzowany **broker**¹ wiadomości Apache Kafka wraz z dwoma tematami input oraz output, wykorzystywanych odpowiednio do gromadzenia danych odczytywanych przez serwis predykcyjny i gromadzenia danych odczytywanych przez pośrednika. Do zarządzania brokerem wykorzystywany jest Apache Zookeeper.

5.2 Technologie

- Apache Kafka — Platforma przetwarzania danych w czasie rzeczywistym.
- Apache Zookeeper — Usługa koordynacyjna systemów rozproszonych.

¹Serwer Apache Kafka zawierający dane należące do tematów i partycji, na które może być podzielony temat

Rozdział 6

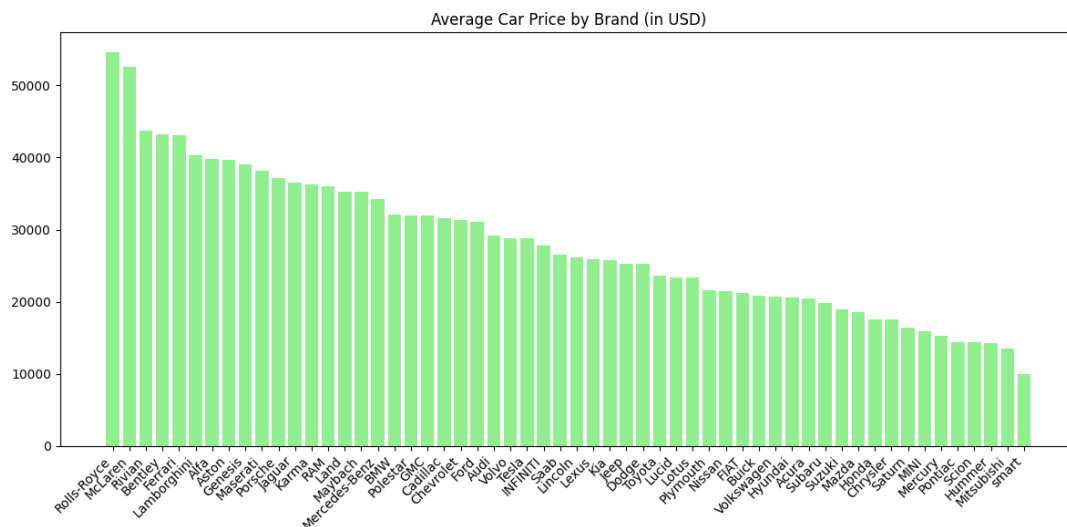
Przygotowanie danych

6.1 Opis

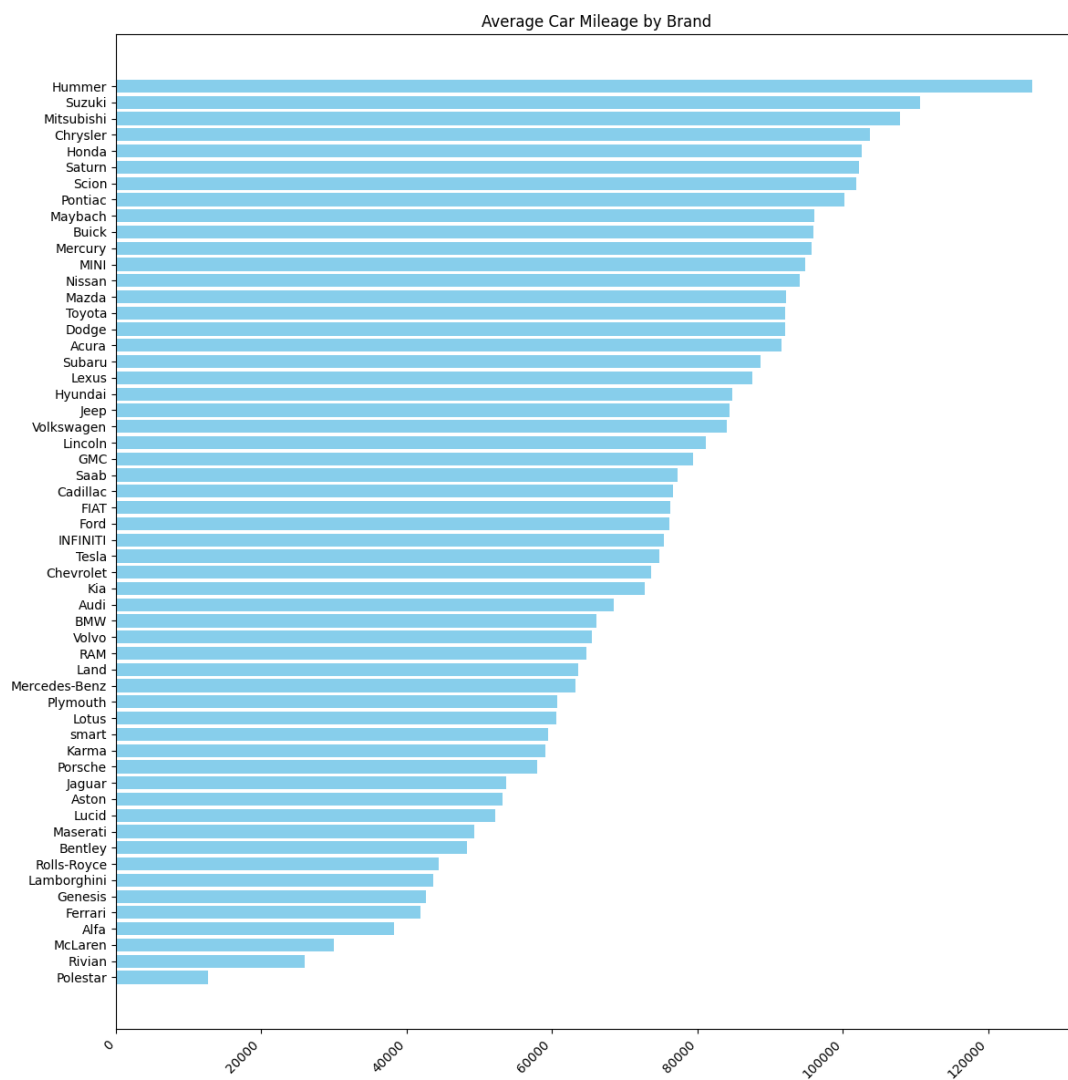
Model jest skuteczny, gdy dane na których się go trenuje są odpowiednio przygotowane. Początkowo należy przeanalizować potencjalne zagrożenia w postaci braków poszczególnych wartości w polach danych oraz wartości odstających, mogących zniekształcić miary statystyczne. Na końcu należy zfaktoryzować, a zatem znumeryzować dane kategoryczne tak aby model mógł się na nich uczyć.

6.2 Wizualizacja danych

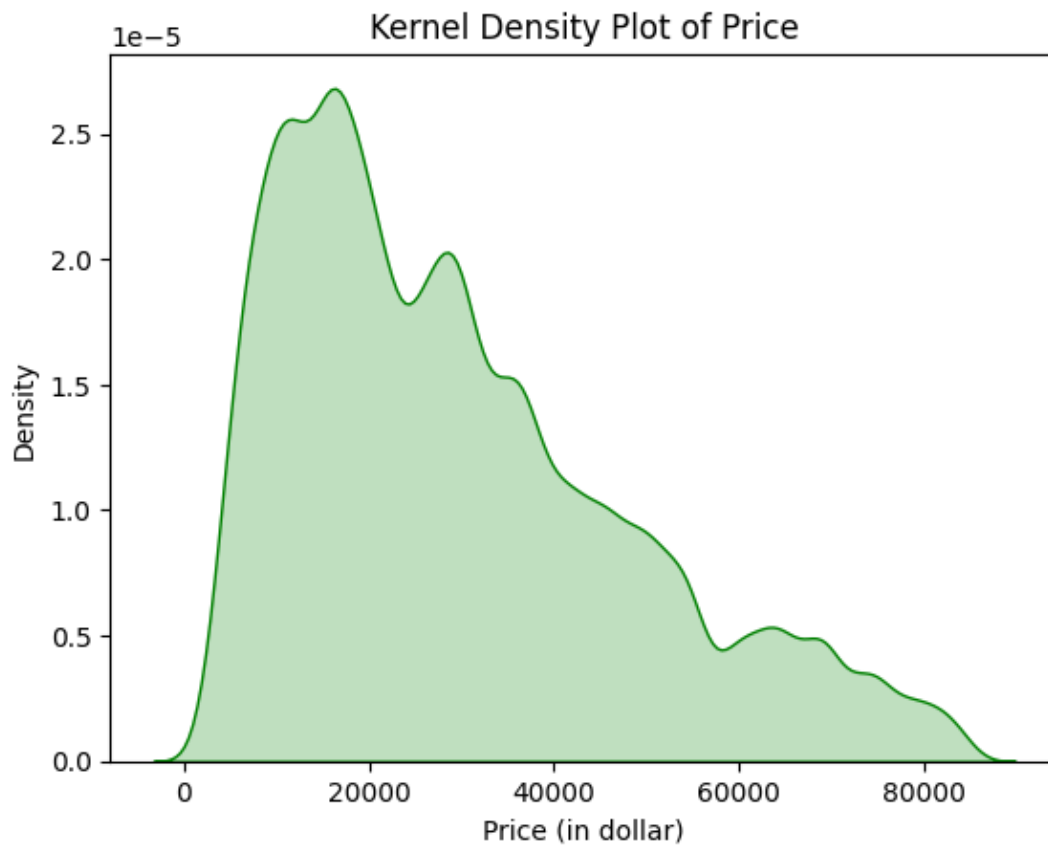
Kilka wykresów pokazujących zbiór danych



Rysunek 6.1: Średnia cena pojazdu danej marki



Rysunek 6.2: Średnia ilość przejechanych mil pojazdów danej marki



Rysunek 6.3: Wykres gęstości cen pojazdów, widać że najwięcej jest ich w okolicach 17 tyś. USD

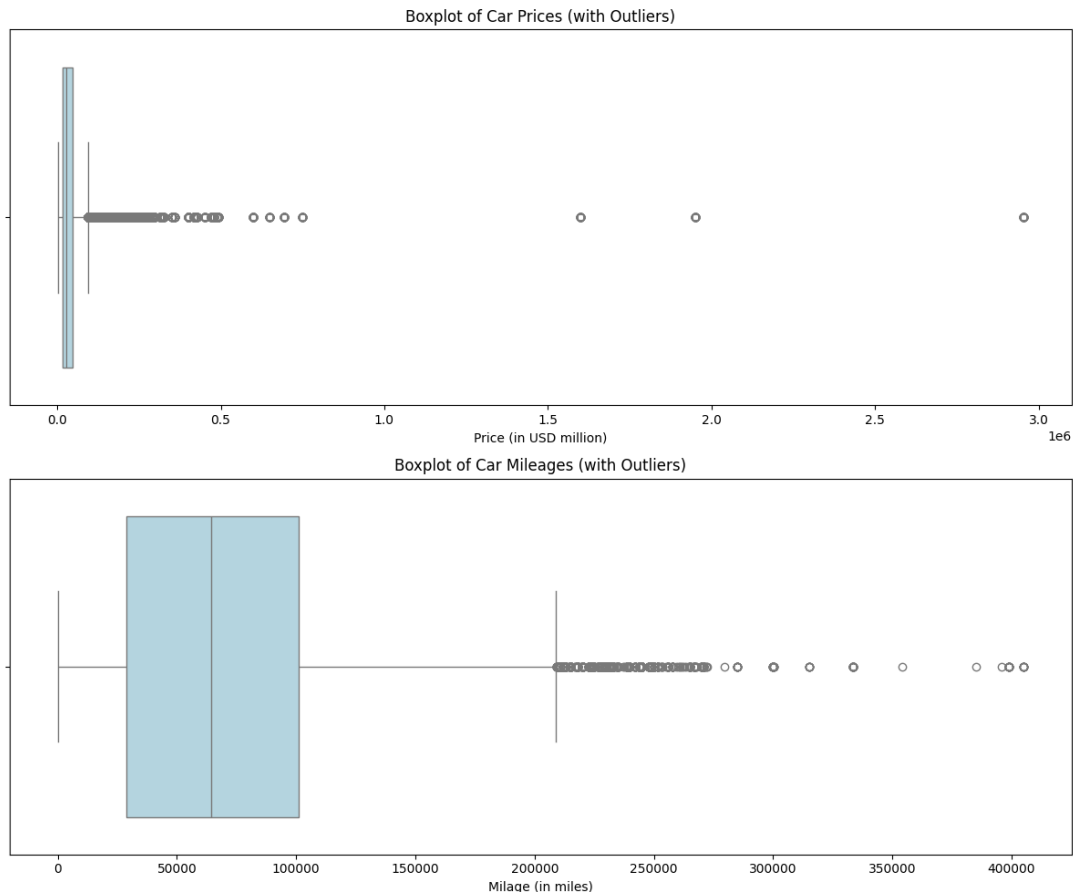
6.3 Puste pola

Postanowiliśmy usunąć puste pola zamiast stosować inne metody sztucznego ich wypełniania na przykład średnią, ponieważ zestaw danych jest obszerny i usunięcie próbek z pustymi wartościami nie odbije się na dokładności modelu.

Przed usunięciem wartości odstających	188533
Po usunięciu wartości odstających	162610

Tabela 6.1: Wielkość zbioru danych przed i po usunięciu wartości odstających

6.4 Zestaw danych z wartościami odstającymi



Rysunek 6.4: Niektóre wartości są zbyt duże, należy się ich pozbyć

6.5 Wykrywanie i usuwanie wartości odstających za pomocą metody IQR

Aby wykryć i usunąć wartości odstające w zbiorze danych na podstawie kolumn `price` oraz `milage`, wykonaliśmy następujące kroki:

1. Obliczenie kwartyli

Pierwszy kwartył (Q_1) oraz trzeci kwartył (Q_3) wyznaczone są dla każdej kolumny. Odpowiadają one 25. i 75. percentylowi:

$$Q_1 = 25. \text{ percentyl}, \quad Q_3 = 75. \text{ percentyl}$$

2. Obliczenie rozstępu międzykwartylowego (IQR)

Rozstęp międzykwartylowy (IQR) oblicza się jako:

$$IQR = Q_3 - Q_1$$

3. Definiowanie granic wartości odstających

Wartości odstające to te, które znajdują się poza przedziałem:

$$\text{Dolna granica} = Q_1 - 1.5 \cdot IQR, \quad \text{Górna granica} = Q_3 + 1.5 \cdot IQR$$

Dla kolumn `price` oraz `milage` wyznacza się odpowiednio:

$$\text{Dolna granica}_{\text{price}} = Q_{1,\text{price}} - 1.5 \cdot IQR_{\text{price}}, \quad \text{Górna granica}_{\text{price}} = Q_{3,\text{price}} + 1.5 \cdot IQR_{\text{price}}$$

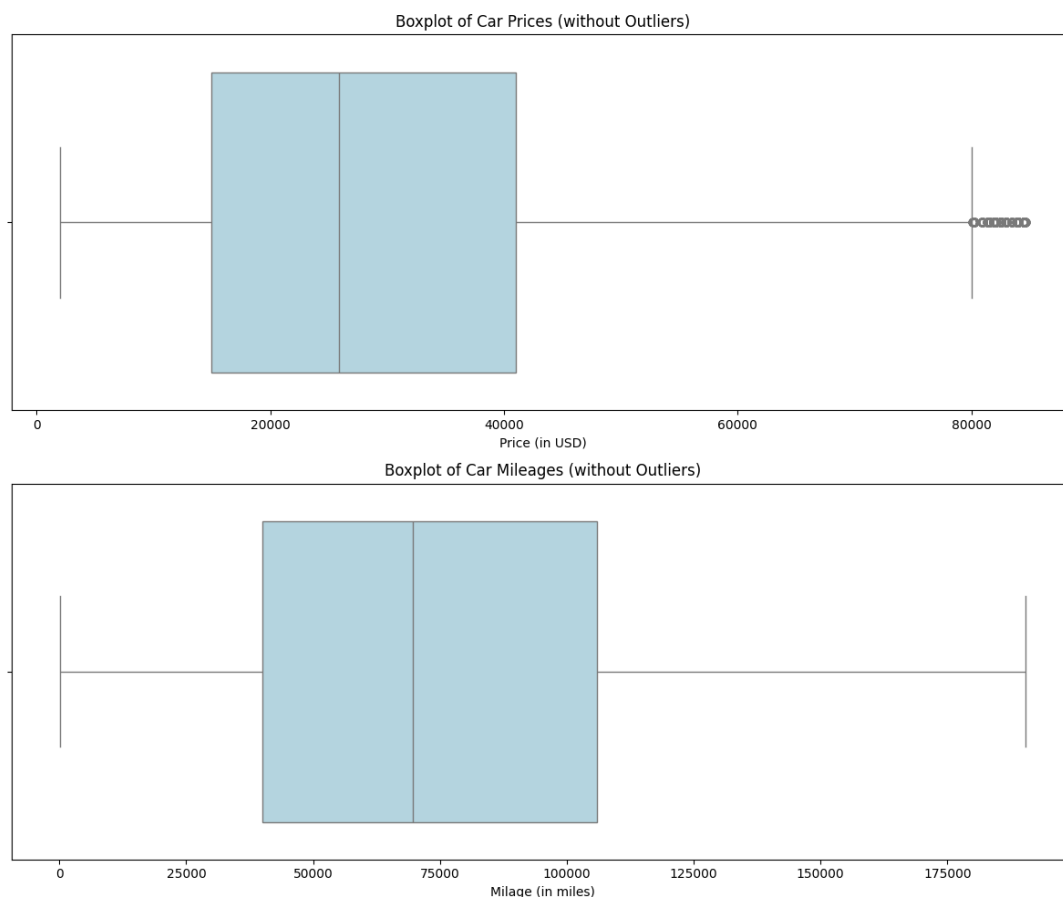
$$\text{Dolna granica}_{\text{milage}} = Q_{1,\text{milage}} - 1.5 \cdot IQR_{\text{milage}}, \quad \text{Górna granica}_{\text{milage}} = Q_{3,\text{milage}} + 1.5 \cdot IQR_{\text{milage}}$$

4. Filtrowanie wierszy bez wartości odstających

Zbiór danych jest filtrowany w taki sposób, aby wartości w kolumnach `price` oraz `milage` spełniały następujące warunki:

$$Q_1 - 1.5 \cdot IQR \leq X \leq Q_3 + 1.5 \cdot IQR$$

6.6 Zestaw danych bez wartości odstających



Rysunek 6.5: Brak lub mała ilość wartości odstających

6.7 Wyodrębnienie znaczących informacji o silniku

W zestawie danych istniała kolumna o nazwie `engine`, w której wartości przedstawiały krótki opis silnika, zawierający takie informacje jak liczba koni mechanicznych, pojemność silnika oraz inne cechy. Przykładowy opis to:

172.0HP 1.6L 4 Cylinder Engine Gasoline Fuel

Z takiego opisu, za pomocą wyrażeń regularnych, wyodrębniliśmy dwie główne informacje: pojemność silnika oraz liczbę koni mechanicznych.

Dla przykładu:

172.0HP 1.6L 4 Cylinder Engine Gasoline Fuel → 1.6, 172.0

Zastąpienie kolumny **engine** nowymi kolumnami:

Po wyodrębnieniu informacji o pojemności silnika oraz liczbie koni mechanicznych, zastąpiliśmy istniejącą kolumnę **engine** dwiema nowymi kolumnami:

- **engine_horsepower** – zawierającą moc silnika w koniach mechanicznych (HP).
- **engine_capacity** – zawierającą pojemność silnika w litrach (L).

6.8 Faktoryzacja danych

Modele uczenia maszynowego wymagają danych numerycznych jako wejścia. Wartości tekstowe (kategoryczne) muszą być przekonwertowane na liczby w sposób, który zachowa sens danych, ale jednocześnie nie wprowadzi sztucznej relacji między kategoriami. Do każdej wartości kategorycznej został przypisany numer, następnie dzięki słownikowi zamieniliśmy wszystkie próbki w wektory cech w następujący sposób:

Dane przed faktoryzacją

Przykład danych wejściowych przed faktoryzacją:

```
1 {  
2   "brand": "Toyota",  
3   "transmission": "Automatic",  
4   "fuel_type": "Gasoline",  
5   "ext_col": "Red",  
6   "int_col": "White",  
7   "accident": "No accident",  
8 }
```

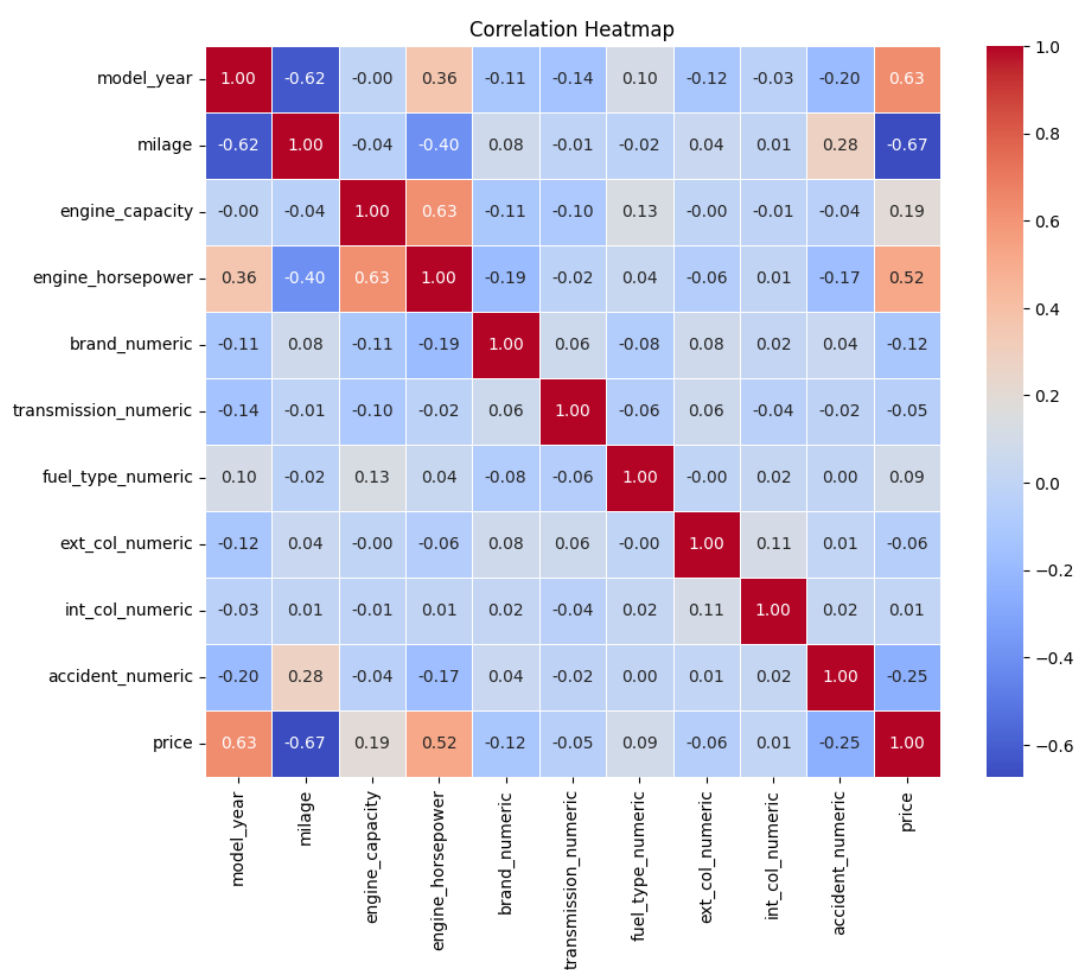
Dane po faktoryzacji

Po zastosowaniu faktoryzacji 'Label-Encoding' (przypisaniu liczb do kategorii), dane będą wyglądać następująco:

```
1 {  
2   "brand": 5,  
3   "transmission": 0,  
4   "fuel_type": 0,  
5   "ext_col": 3,  
6   "int_col": 5,  
7   "accident": 0,  
8 }
```

Przykładowy wiersz danych po faktoryzacji:

Nazwa zmiennej	Wartość
model_year	2002
milage	143250.0
price	4999
engine_capacity	3.9
engine_horsepower	252.0
brand_numeric	17.0
transmission_numeric	0.0
fuel_type_numeric	0.0
ext_col_numeric	3.0
int_col_numeric	1.0
accident_numeric	1.0



Rysunek 6.6: Macierz korelacji wszystkich cech, cechy korelujące się w największym stopniu z ceną: rok produkcji modelu, przebieg samochodu oraz konie mechaniczne

Rozdział 7

Serwis predykcyjny

7.1 Opis

Zadaniem serwisu predykcyjnego jest dokonanie predykcji ceny samochodu na podstawie dostarczonego zestawu cech. W tym celu wykorzystuje gotowy, zapisany model przygotowany przy użyciu modułu SparkML. Dane przekazywane do modelu są odczytywane z tematu input za pomocą frameworka Spark. Cena zwrócona przez model zostaje zapisana na temat wyjściowy output.

7.2 Technologie

- Python — Język skryptowy.
- Apache Spark — Framework do sprawnego przetwarzania zbiorów danych w pamięci.
- Apache SparkML — Moduł Apache Spark przeznaczony do uczenia maszynowego.

7.3 Wybór modelu

Podczas wyboru modelu kierowaliśmy się tym jak zostały zfaktoryzowane dane. Dokonałiśmy tego korzystając z 'Label encoding', czyli przypisując unikalną wartość do każdej danej kategorycznej, na przykład "BMW" → 0, "Audi" → 1.

W tym przypadku modele na bazie regresji liniowej nie będą skuteczne, ponieważ takie przypisanie może prowadzić do błędnych założeń o istnieniu relacji porównawczych między kategoriami np. różnica między 0 a 1 jest taka sama jak między 1 a 2, co niekoniecznie ma sens w przypadku zmiennych kategorycznych.

Rozwiązaniem będą modele bazujące na drzewach, z których do testowania wybraliśmy Drzewo decyzyjne oraz Las losowy.