Predicting Vehicle Prices Using Regression Models

Objectives

The primary goal of this project is to develop a robust regression model to predict used car prices for a reseller based on various listed features and specifications. In addition to predicting prices, the project focuses on identifying feature importance and mitigating overfitting through the application of regularisation techniques.

There can be several business objectives for this, such as:

- **Price Prediction**: Model car prices based on features like mileage, fuel type, and performance.
- **Market Analysis**: Explore trends and preferences in the used car market, by type, region, or other metrics.
- **Feature Importance**: Identify the most important factors influencing car prices (e.g., fuel type, mileage, age).

Tasks Overview

The data pipeline for this task involves the following steps:

1. **Dataset Overview**
2. **Data Preprocessing**
3. **Data Visualisation & Exploration**
4. **Model Building**
5. **Regularisation**

1 Data Understanding

| **Variable** | **Description** | ---------|---------------|| `make_model` | The brand and model of the vehicle (e.g., 'Audi A1'). || `body_type` | The body style of the vehicle, such as Sedan, Compact, or Station Wagon. || `price` | The listed price of the car in currency. || `vat` | Indicates the VAT status for the vehicle's price (e.g., VAT deductible, Price negotiable). || `km` | The total mileage (in kilometers) of the vehicle, indicating its usage. || `Type` | Condition of the vehicle, whether it's 'Used' or 'New'.|| `Fuel` | Type of fuel the vehicle uses, such as 'Diesel', 'Benzine', etc. || `Gears` | The number of gears in the vehicle's transmission. || `Comfort_Convenience` | Comfort and convenience features, such as 'Air conditioning', 'Leather steering wheel', 'Cruise control', and more. ||

`Entertainment_Media` | Media features available in the vehicle, including 'Bluetooth', 'MP3', 'Radio', etc. | | `Extras` | Additional features like 'Alloy wheels', 'Sport suspension', etc.| | `Safety_Security` | Safety features like 'ABS', 'Airbags', 'Electronic stability control', 'Isofix', etc. | | `age` | Age of the car (calculated based on the model year). | | `Previous_Owners` | The number of previous owners the car has had. | | `hp_kW` | Engine power in kilowatts (kW), indicating the performance capacity of the engine.| | `Inspection_new` | Indicates whether the car has recently undergone an inspection (1 for yes, 0 for no). | | `Paint_Type` | The type of paint on the car, such as 'Metallic', 'Matte', etc. | | `Upholstery_type` | The material used for the interior upholstery, such as 'Cloth', 'Leather', etc.| | `Gearing_Type` | The type of transmission the car uses, either 'Automatic' or 'Manual'. | | `Displacement_cc` | The engine displacement in cubic centimeters (cc), indicating the size of the engine.| | `Weight_kg` | The total weight of the vehicle in kilograms.| | `Drive_chain` | The type of drivetrain, indicating whether it's 'Front' or 'Rear' wheel drive. | | `cons_comb` | The combined fuel consumption in liters per 100 kilometers.|

1.1 Data Loading

Importing Necessary Libraries

In [121…

```
# Importing necessary libraries
# Core Python
import numpy as np
import pandas as pd

# Visualization
import matplotlib.pyplot as plt

# Model building & preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

# Regression models
from sklearn.linear_model import LinearRegression, RidgeCV, Lasso, ElasticNe

# Model evaluation
from sklearn.metrics import mean_squared_error, r2_score
```

1.1.1

Load the dataset

In [122…

```
# Load the data
# Load the dataset
import pandas as pd

file_path = "Car_Price_data.csv"  # Update path if your file is in another c
```

```
df = pd.read_csv(file_path)

print("Data loaded successfully.")
print("Shape:", df.shape)

# Preview first few rows
df.head()
```

```
Data loaded successfully.
Shape: (15915, 23)
```

Out[122…

	make_model	body_type	price	vat	km	Type	Fuel	Gears	
0	Audi A1	Sedans	15770	VAT deductible	56013.0	Used	Diesel	7.0	conditi
1	Audi A1	Sedans	14500	Price negotiable	80000.0	Used	Benzine	7.0	Ai
2	Audi A1	Sedans	14640	VAT deductible	83450.0	Used	Diesel	7.0	
3	Audi A1	Sedans	14500	VAT deductible	73000.0	Used	Diesel	6.0	susp
4	Audi A1	Sedans	16790	VAT deductible	16200.0	Used	Diesel	7.0	conditi

5 rows × 23 columns

2 Analysis and Feature Engineering [35 marks]

2.1 Preliminary Analysis and Frequency Distributions [13 marks]

2.1.1 [1 marks]

Check and fix missing values.

In [123…

```
# Find the proportion of missing values in each column and handle if found

missing_pct = df.isnull().mean().sort_values(ascending=False)

print("Missing value percentage per column:")
print(missing_pct)

numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns
categorical_cols = df.select_dtypes(include=['object', 'category']).columns
df[numeric_cols] = df[numeric_cols].fillna(df[numeric_cols].median())
```

```
df[categorical_cols] = df[categorical_cols].fillna(df[categorical_cols].mode

print("\nMissing values handled successfully.")
```

```
Missing value percentage per column:
make_model              0.0
age                     0.0
Drive_chain             0.0
Weight_kg               0.0
Displacement_cc         0.0
Gearing_Type            0.0
Upholstery_type         0.0
Paint_Type              0.0
Inspection_new          0.0
hp_kW                   0.0
Previous_Owners         0.0
Safety_Security         0.0
body_type               0.0
Extras                  0.0
Entertainment_Media     0.0
Comfort_Convenience     0.0
Gears                   0.0
Fuel                    0.0
Type                    0.0
km                      0.0
vat                     0.0
price                   0.0
cons_comb               0.0
dtype: float64

Missing values handled successfully.
```

From the features, identify the target feature and numerical and categorical predictors. Select the numerical and categorical features carefully as they will be used in analysis.

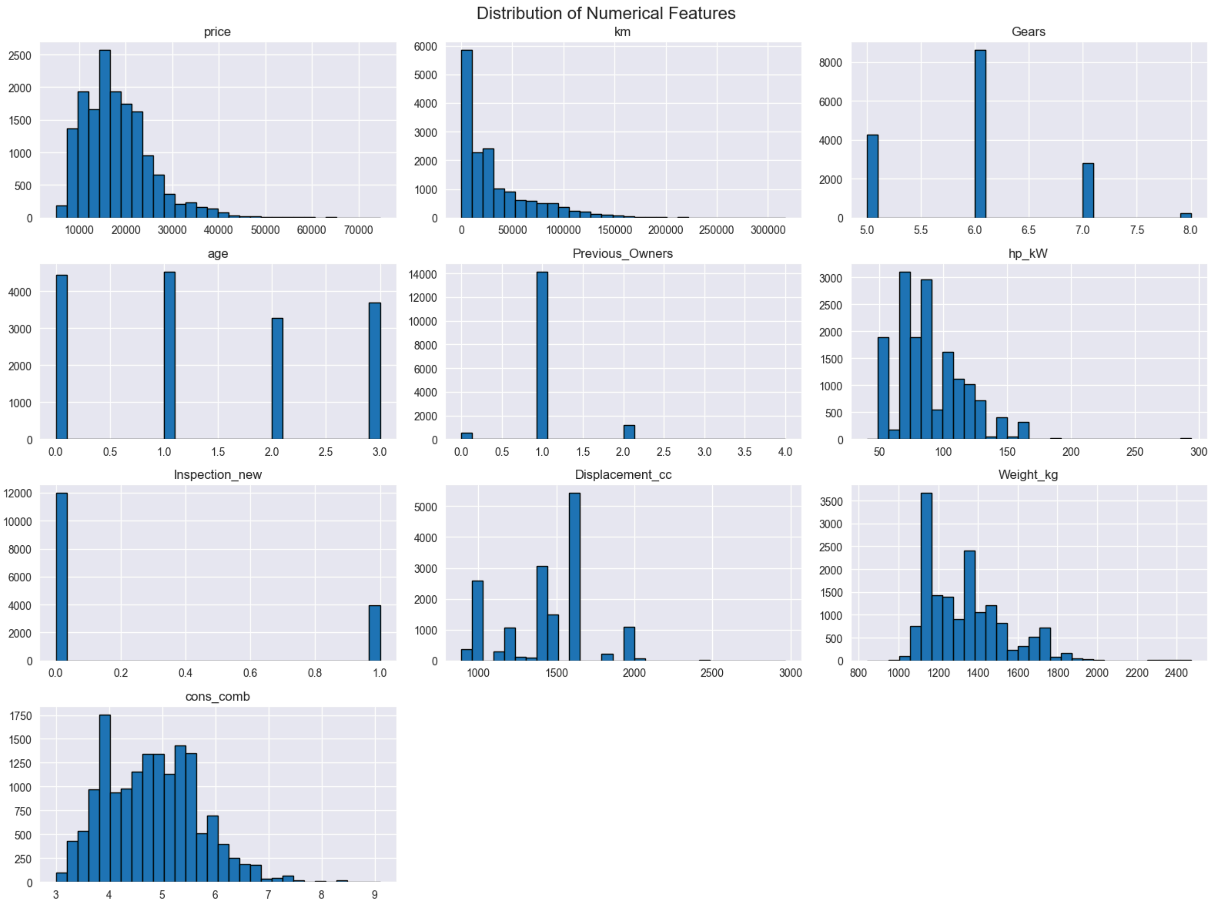
2.1.2 [3 marks]

Identify numerical predictors and plot their frequency distributions.

In [124…

```
# Identify numerical features and plot histograms

import matplotlib.pyplot as plt

numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns
print("Numerical features:", list(numeric_cols))

df[numeric_cols].hist(figsize=(16, 12), bins=30, edgecolor='black')
plt.suptitle("Distribution of Numerical Features", fontsize=16)
plt.tight_layout()
plt.show()
```

```
Numerical features: ['price', 'km', 'Gears', 'age', 'Previous_Owners', 'hp_k
W', 'Inspection_new', 'Displacement_cc', 'Weight_kg', 'cons_comb']
```



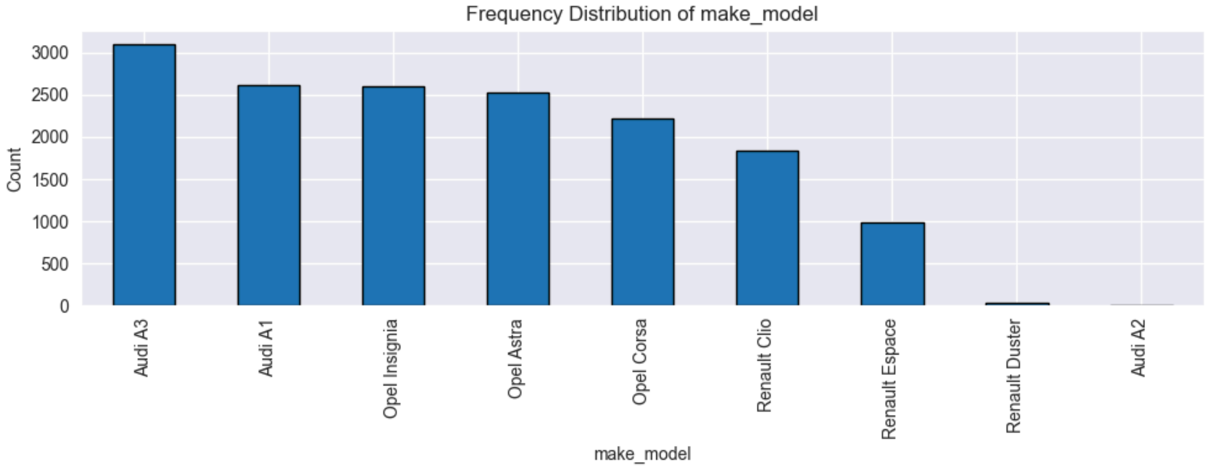
2.1.3 [3 marks]

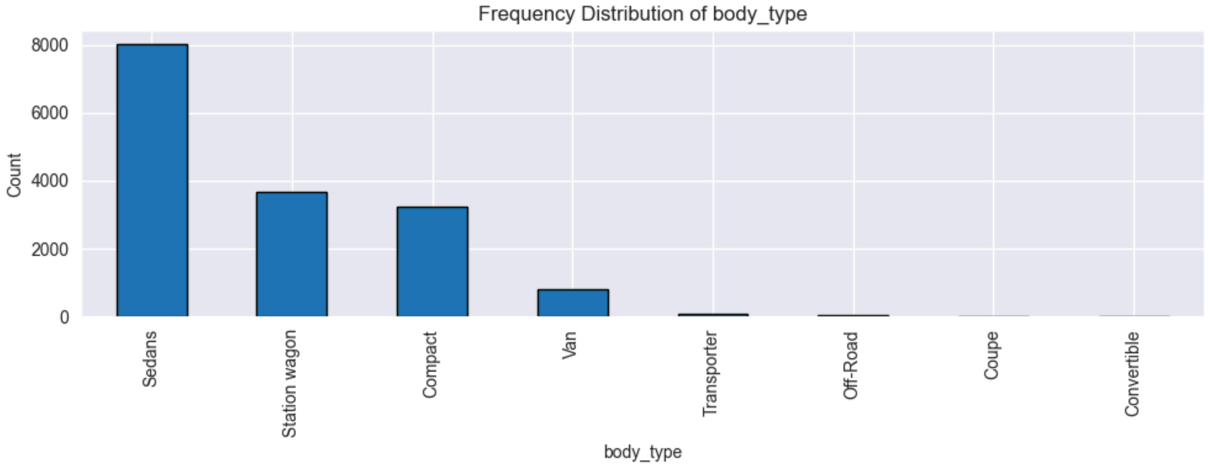
Identify categorical predictors and plot their frequency distributions.

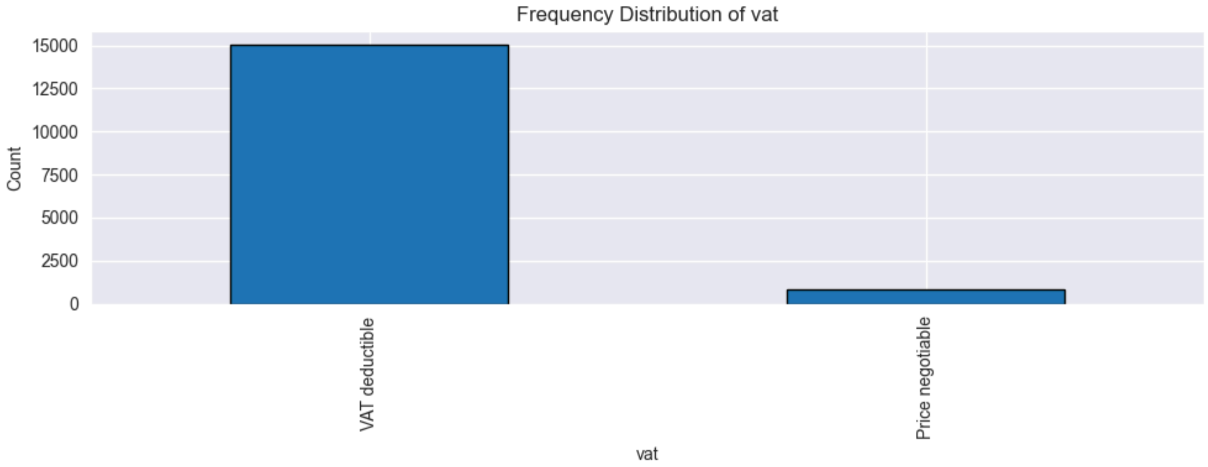
In [125…

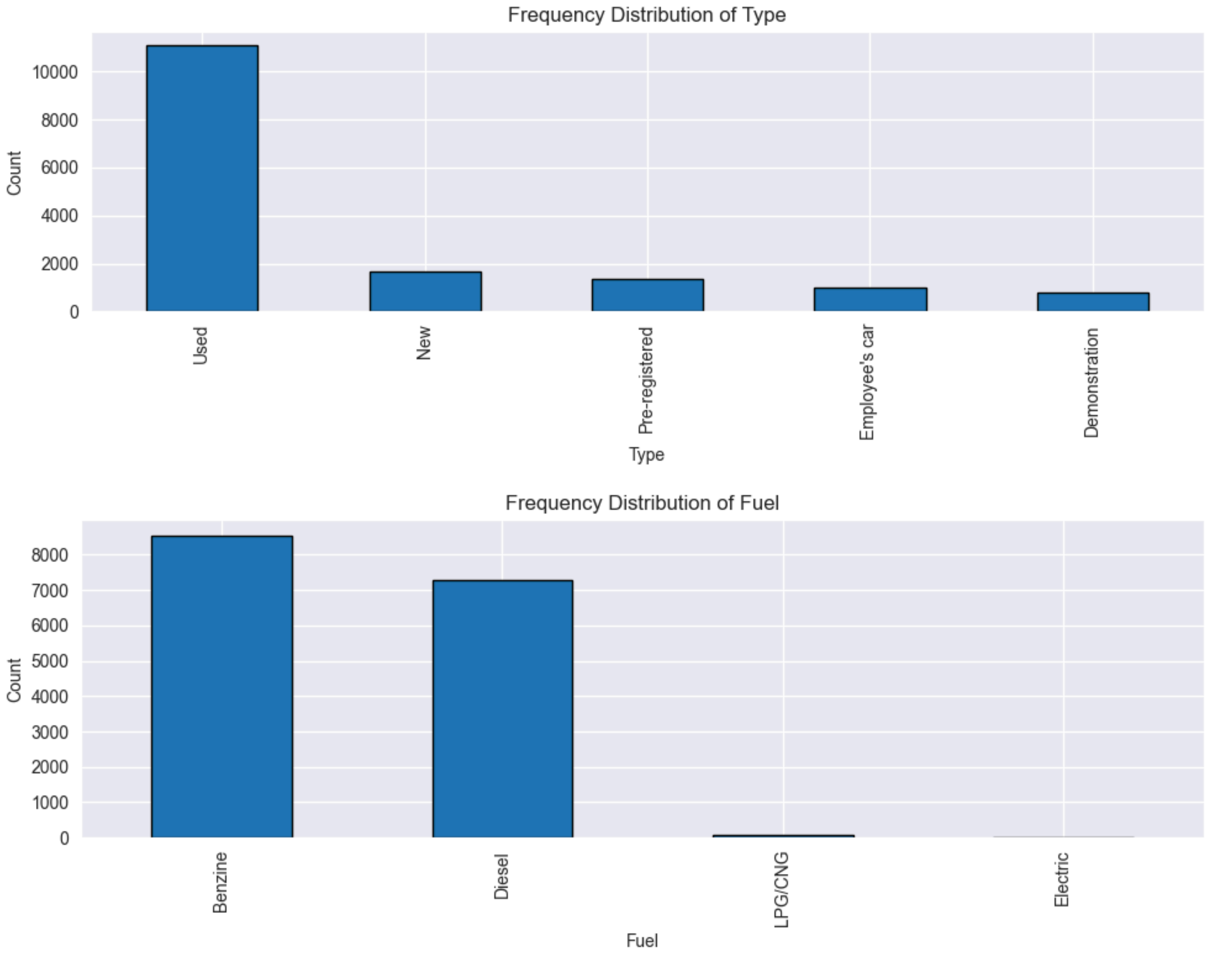
```
# Identify categorical columns and check their frequency distributions


# Identify categorical predictors
categorical_cols = df.select_dtypes(include=['object', 'category']).columns.

# Columns that should NOT be treated as normal categorical features
multi_label_cols = [
    "Comfort_Convenience",
    "Entertainment_Media",
    "Extras",
    "Safety_Security"
]

# Final categorical predictors
categorical_predictors = [col for col in categorical_cols if col]

print("Categorical predictors:", categorical_predictors)

import matplotlib.pyplot as plt

for col in categorical_predictors:
    plt.figure(figsize=(10, 4))
    df[col].value_counts().head(20).plot(kind='bar', edgecolor='black')
    plt.title(f"Frequency Distribution of {col}")
    plt.xlabel(col)
    plt.ylabel("Count")
    plt.tight_layout()
    plt.show()
```

```
Categorical predictors: ['make_model', 'body_type', 'vat', 'Type', 'Fuel',
'Comfort_Convenience', 'Entertainment_Media', 'Extras', 'Safety_Security',
'Paint_Type', 'Upholstery_type', 'Gearing_Type', 'Drive_chain']
```

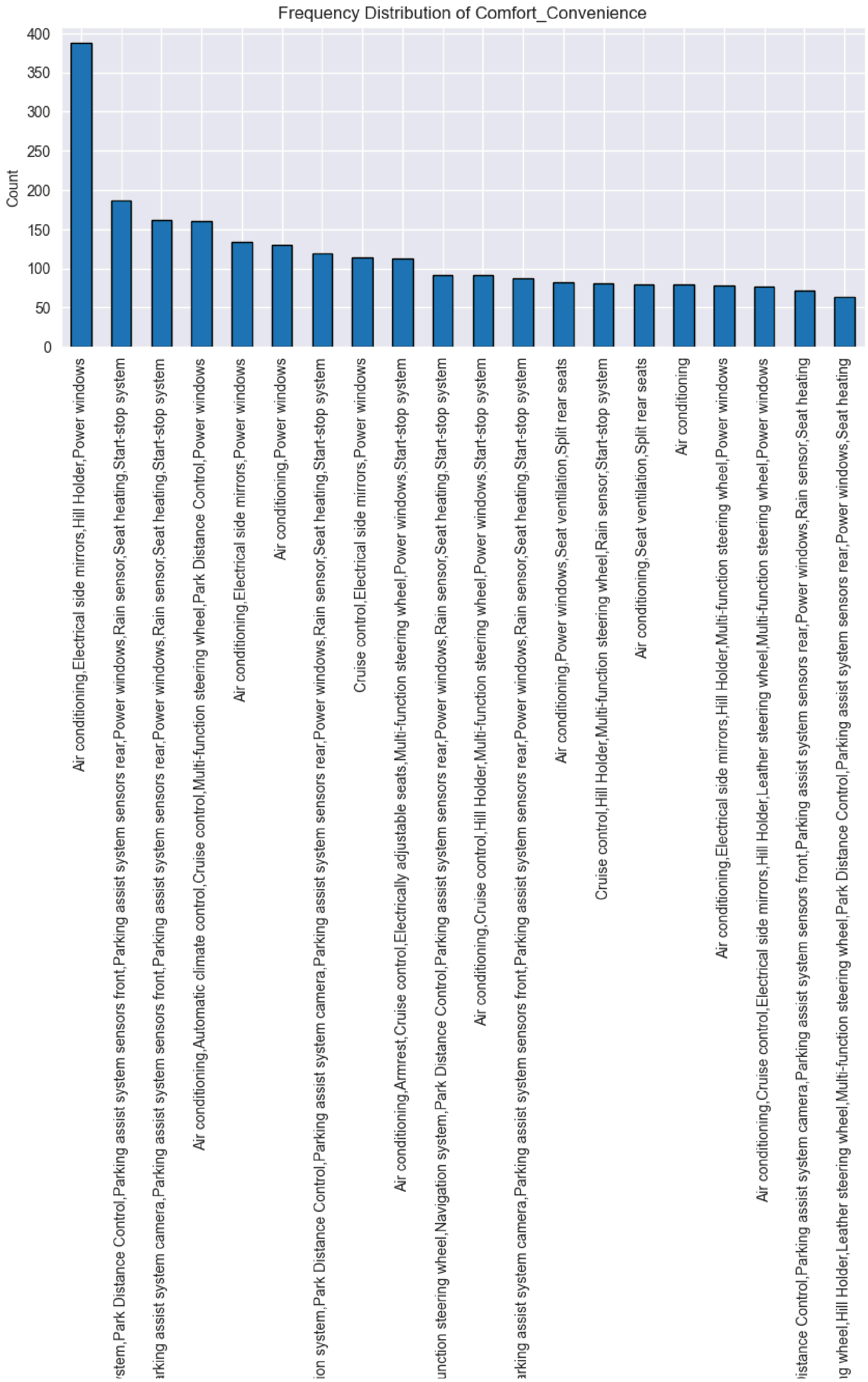

Frequency Distribution of make_model


Frequency Distribution of body_type


Frequency Distribution of vat



```
/var/folders/q9/_3w68hr55rdcl70s4bt4jjbh0000gn/T/ipykernel_66612/2463149345.
py:28: UserWarning: Tight layout not applied. The bottom and top margins can
not be made large enough to accommodate all Axes decorations.
  plt.tight_layout()
```


Frequency Distribution of Comfort_Convenience
Count
400
350
300
250
200
150
100
50
0
Air conditioning,Electrical side mirrors,Hill Holder,Power windows
stem,Park Distance Control,Parking assist system sensors front,Parking assist system sensors rear,Power windows,Rain sensor,Seat heating,Start-stop system
arking assist system camera,Parking assist system sensors front,Parking assist system sensors rear,Power windows,Rain sensor,Seat heating,Start-stop system
Air conditioning,Automatic climate control,Cruise control,Multi-function steering wheel,Park Distance Control,Power windows
Air conditioning,Electrical side mirrors,Power windows
Air conditioning,Power windows
ion system,Park Distance Control,Parking assist system camera,Parking assist system sensors rear,Power windows,Rain sensor,Seat heating,Start-stop system
Cruise control,Electrical side mirrors,Power windows
Air conditioning,Armrest,Cruise control,Electrically adjustable seats,Multi-function steering wheel,Power windows,Start-stop system
unction steering wheel,Navigation system,Park Distance Control,Parking assist system sensors rear,Power windows,Rain sensor,Seat heating,Start-stop system
Air conditioning,Cruise control,Hill Holder,Multi-function steering wheel,Power windows,Start-stop system
arking assist system camera,Parking assist system sensors front,Parking assist system sensors rear,Power windows,Rain sensor,Seat heating,Start-stop system
Air conditioning,Power windows,Seat ventilation,Split rear seats
Cruise control,Hill Holder,Multi-function steering wheel,Rain sensor,Start-stop system
Air conditioning,Seat ventilation,Split rear seats
Air conditioning
Air conditioning,Electrical side mirrors,Hill Holder,Multi-function steering wheel,Power windows
Air conditioning,Cruise control,Electrical side mirrors,Hill Holder,Leather steering wheel,Multi-function steering wheel,Power windows
istance Control,Parking assist system camera,Parking assist system sensors front,Parking assist system sensors rear,Power windows,Rain sensor,Seat heating
g wheel,Hill Holder,Leather steering wheel,Multi-function steering wheel,Park Distance Control,Parking assist system sensors rear,Power windows,Seat heating

Air conditioning,Armrest,Automatic climate control,Cruise control,Electrical side mirrors,Leather steering wheel,Light sensor,Lumbar support,Multi-function steering wheel,Navigation sy

al side mirrors,Electric tailgate,Heated steering wheel,Hill Holder,Keyless central door lock,Leather steering wheel,Light sensor,Lumbar support,Multi-function steering wheel,Navigation system,Park Distance Control,Pa

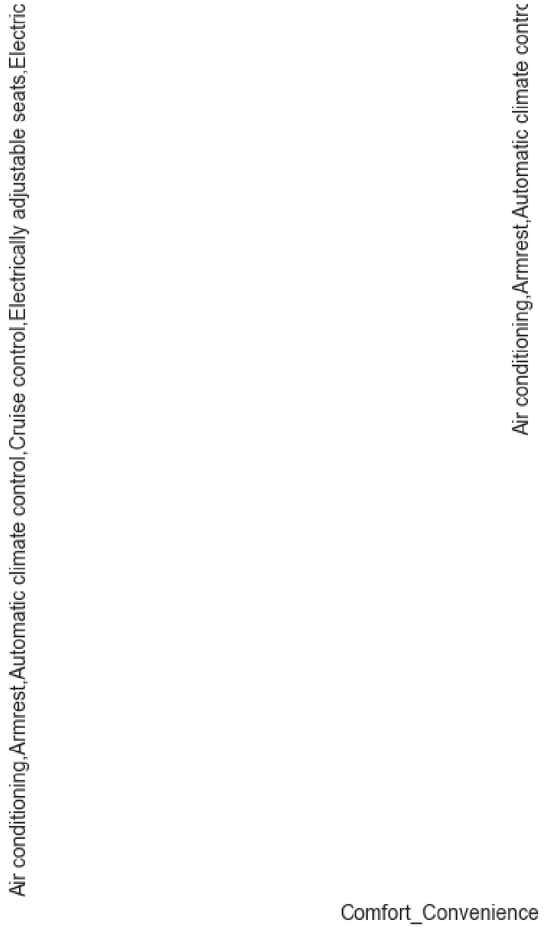
Air conditioning,Automatic climate control,Cruise control,Electrical side mirrors,Heated steering wheel,Hill Holder,Leather steering wheel,Light sensor,Multi-function steering wheel,Navigat

Air conditioning,Armrest,Automatic climate control,Cruise control,Electrical side mirrors,Hill Holder,Leather steering wheel,Light sensor,Multi-f

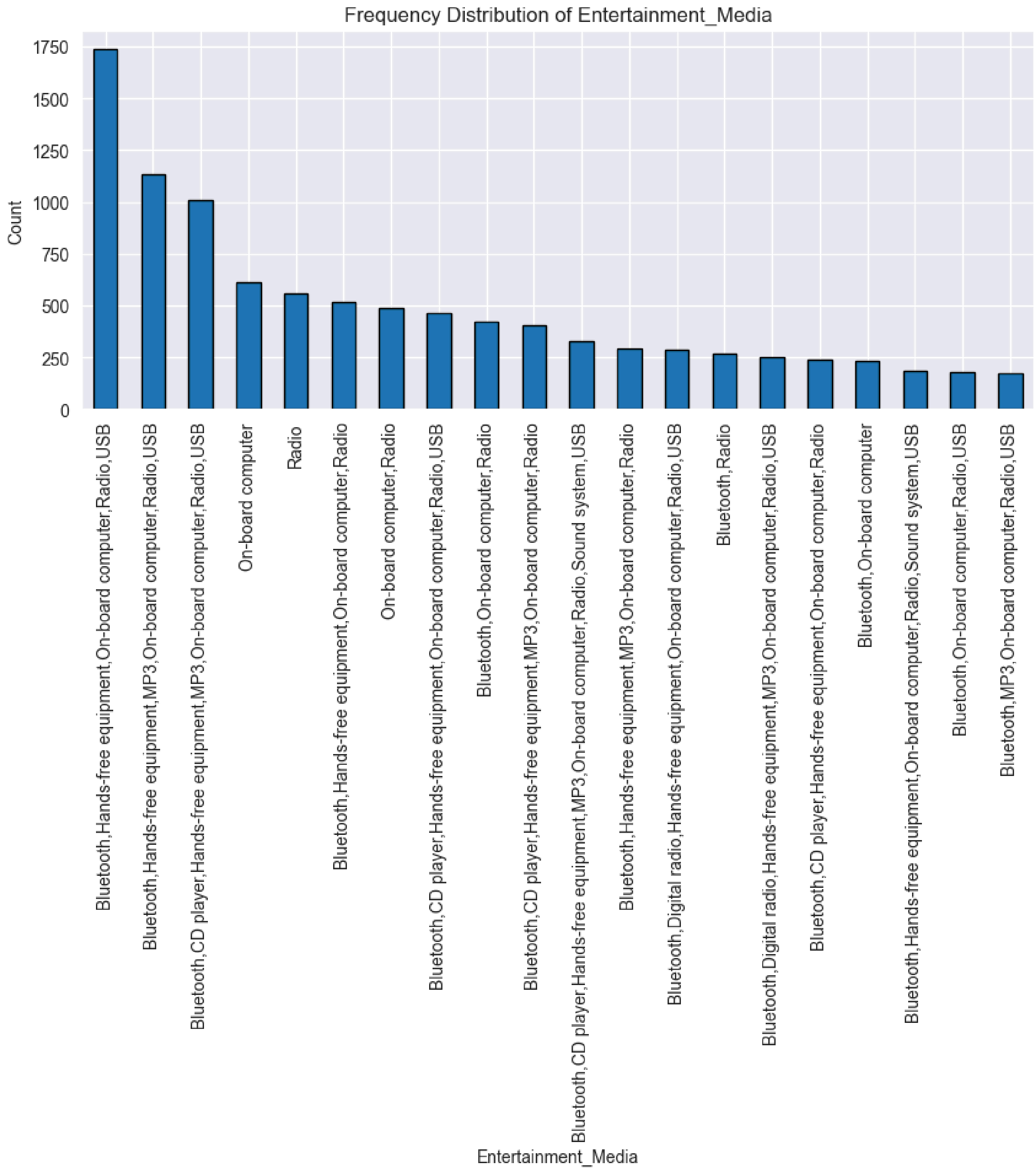
ol,Cruise control,Electrical side mirrors,Hill Holder,Keyless central door lock,Leather steering wheel,Light sensor,Lumbar support,Multi-function steering wheel,Navigation system,Panorama roof,Park Distance Control,Pa

Air conditioning,Cruise control,Electrically heated windshield,Electrical side mirrors,Heated steering wheel,Hill Holder,Leather steering wheel,Light sensor,Multi-function steering wheel,Park D

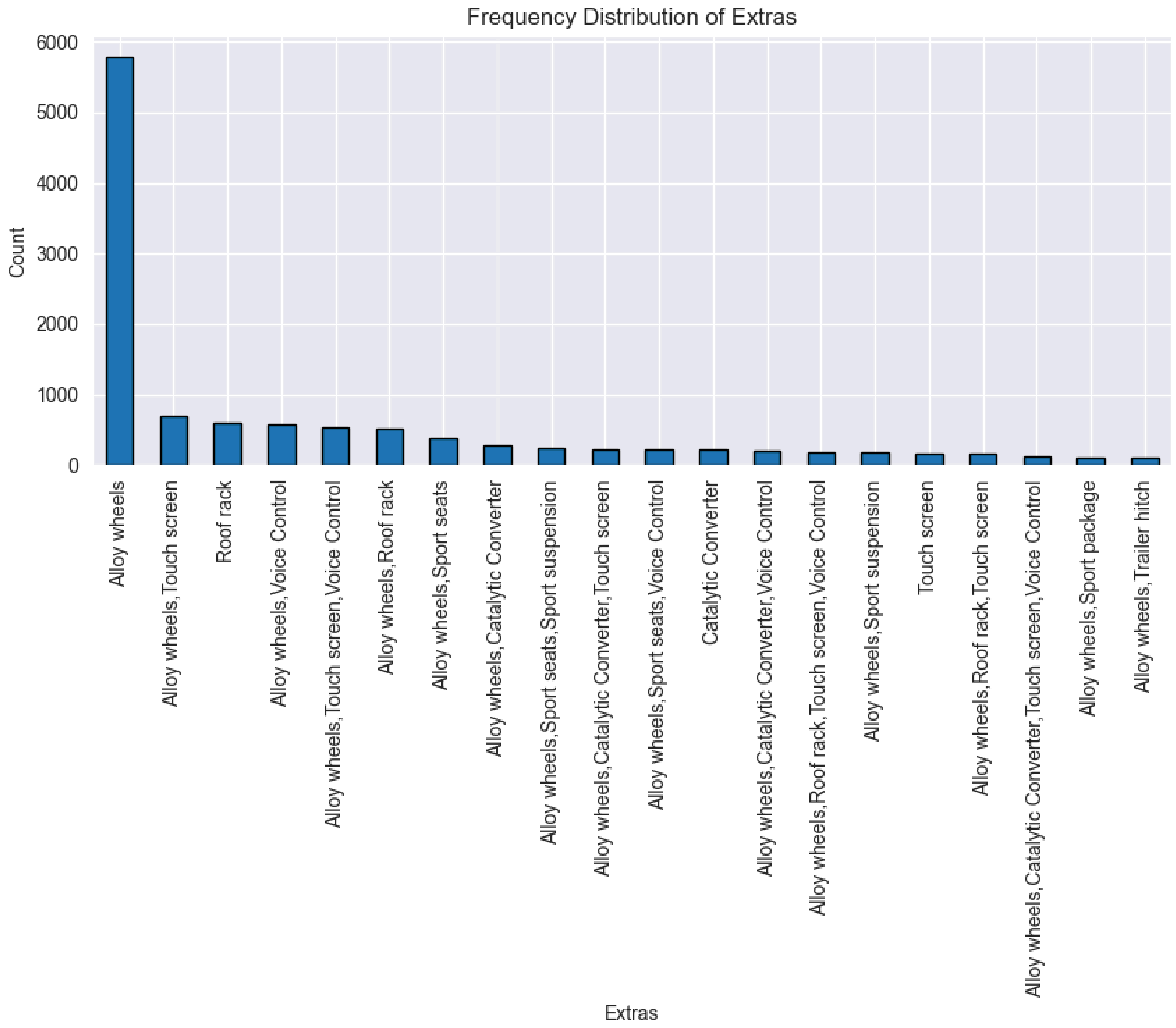
Air conditioning,Cruise control,Electrical side mirrors,Heated steerin



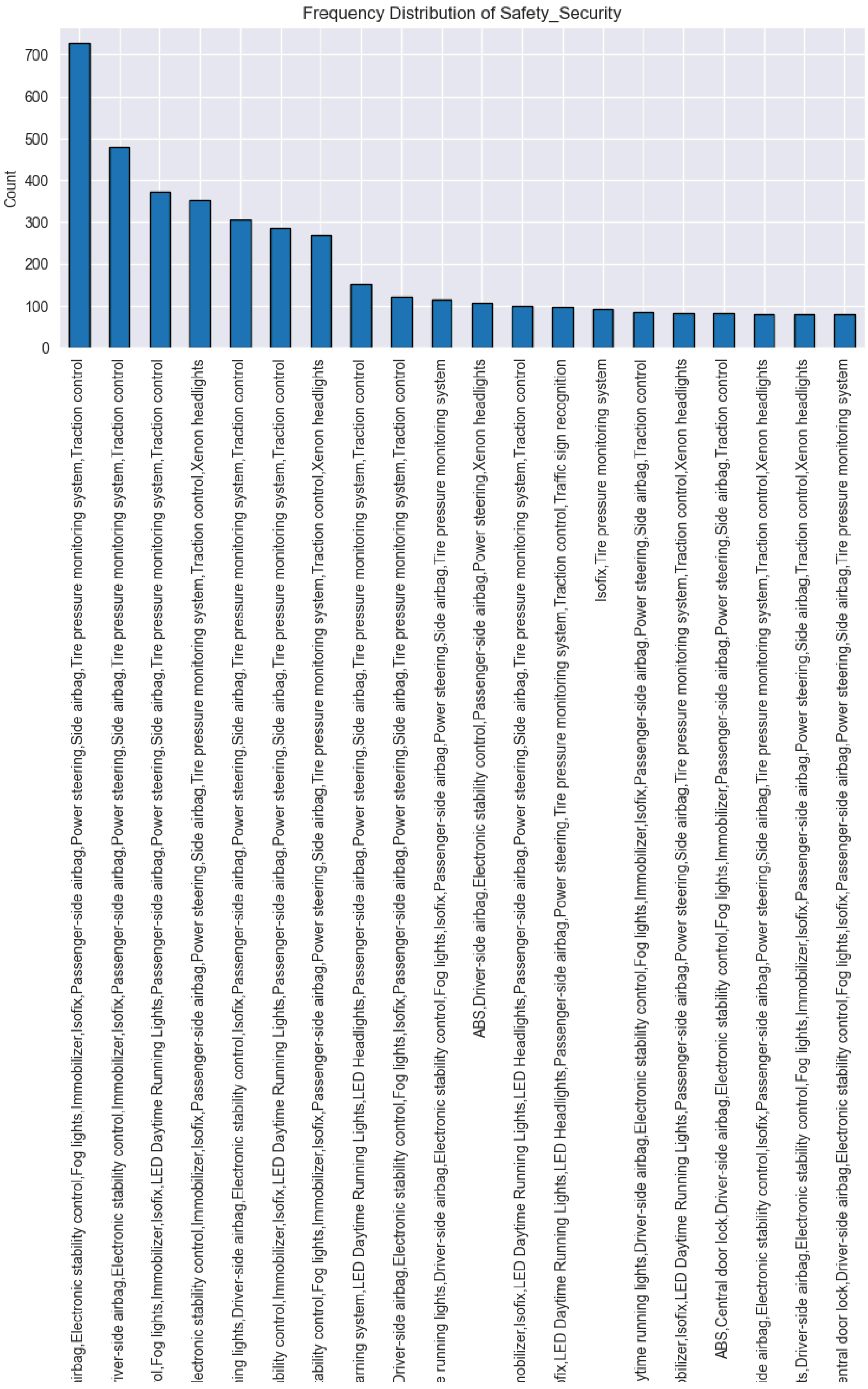
```
/var/folders/q9/_3w68hr55rdcl70s4bt4jjbh0000gn/T/ipykernel_66612/2463149345.
py:28: UserWarning: Tight layout not applied. The bottom and top margins can
not be made large enough to accommodate all Axes decorations.
  plt.tight_layout()
```



```
/var/folders/q9/_3w68hr55rdcl70s4bt4jjbh0000gn/T/ipykernel_66612/2463149345.
py:28: UserWarning: Tight layout not applied. The bottom and top margins can
not be made large enough to accommodate all Axes decorations.
  plt.tight_layout()
```



```
/var/folders/q9/_3w68hr55rdcl70s4bt4jjbh0000gn/T/ipykernel_66612/2463149345.
py:28: UserWarning: Tight layout not applied. The bottom and top margins can
not be made large enough to accommodate all Axes decorations.
  plt.tight_layout()
```


Frequency Distribution of Safety_Security
Count
0
100
200
300
400
500
600
700
irbag,Electronic stability control,Fog lights,Immobilizer,Isofix,Passenger-side airbag,Power steering,Side airbag,Tire pressure monitoring system,Traction control
iver-side airbag,Electronic stability control,Immobilizer,Isofix,Passenger-side airbag,Power steering,Side airbag,Tire pressure monitoring system,Traction control
ol,Fog lights,Immobilizer,Isofix,LED Daytime Running Lights,Passenger-side airbag,Power steering,Side airbag,Tire pressure monitoring system,Traction control
lectronic stability control,Immobilizer,Isofix,Passenger-side airbag,Power steering,Side airbag,Tire pressure monitoring system,Traction control,Xenon headlights
ing lights,Driver-side airbag,Electronic stability control,Isofix,Passenger-side airbag,Power steering,Side airbag,Tire pressure monitoring system,Traction control
bility control,Immobilizer,Isofix,LED Daytime Running Lights,Passenger-side airbag,Power steering,Side airbag,Tire pressure monitoring system,Traction control
ability control,Fog lights,Immobilizer,Isofix,Passenger-side airbag,Power steering,Side airbag,Tire pressure monitoring system,Traction control,Xenon headlights
arning system,LED Daytime Running Lights,LED Headlights,Passenger-side airbag,Power steering,Side airbag,Tire pressure monitoring system,Traction control
Driver-side airbag,Electronic stability control,Fog lights,Isofix,Passenger-side airbag,Power steering,Side airbag,Tire pressure monitoring system,Traction control
e running lights,Driver-side airbag,Electronic stability control,Fog lights,Isofix,Passenger-side airbag,Power steering,Side airbag,Tire pressure monitoring system
ABS,Driver-side airbag,Electronic stability control,Passenger-side airbag,Power steering,Xenon headlights
nobilizer,Isofix,LED Daytime Running Lights,LED Headlights,Passenger-side airbag,Power steering,Side airbag,Tire pressure monitoring system,Traction control
ofix,LED Daytime Running Lights,LED Headlights,Passenger-side airbag,Power steering,Tire pressure monitoring system,Traction control,Traffic sign recognition
Isofix,Tire pressure monitoring system
ytime running lights,Driver-side airbag,Electronic stability control,Fog lights,Immobilizer,Isofix,Passenger-side airbag,Power steering,Side airbag,Traction control
bilizer,Isofix,LED Daytime Running Lights,Passenger-side airbag,Power steering,Side airbag,Tire pressure monitoring system,Traction control,Xenon headlights
ABS,Central door lock,Driver-side airbag,Electronic stability control,Fog lights,Immobilizer,Passenger-side airbag,Power steering,Side airbag,Traction control
ide airbag,Electronic stability control,Isofix,Passenger-side airbag,Power steering,Side airbag,Tire pressure monitoring system,Traction control,Xenon headlights
ts,Driver-side airbag,Electronic stability control,Fog lights,Immobilizer,Isofix,Passenger-side airbag,Power steering,Side airbag,Traction control,Xenon headlights
entral door lock,Driver-side airbag,Electronic stability control,Fog lights,Isofix,Passenger-side airbag,Power steering,Side airbag,Tire pressure monitoring system

ABS,Central door lock,Daytime running lights,Driver-side a

ABS,Central door lock,Daytime running lights,Dr

ABS,Central door lock,Daytime running lights,Driver-side airbag,Electronic stability contr

ABS,Central door lock,Daytime running lights,Driver-side airbag,El

ABS,Central door lock,Daytime runn

ABS,Central door lock,Daytime running lights,Driver-side airbag,Electronic sta

ABS,Central door lock,Daytime running lights,Driver-side airbag,Electronic st

ABS,Adaptive headlights,Central door lock,Daytime running lights,Driver-side airbag,Electronic stability control,Emergency brake assistant,Fog lights,Immobilizer,Isofix,Lane departure wa

ABS,Central door lock,Daytime running lights,I

ABS,Central door lock,Daytime

ABS,Central door lock,Daytime running lights,Driver-side airbag,Electronic stability control,Fog lights,Imm

ABS,Blind spot monitor,Central door lock,Daytime running lights,Driver-side airbag,Electronic stability control,Emergency brake assistant,Immobilizer,Iso

ABS,Central door lock,Da

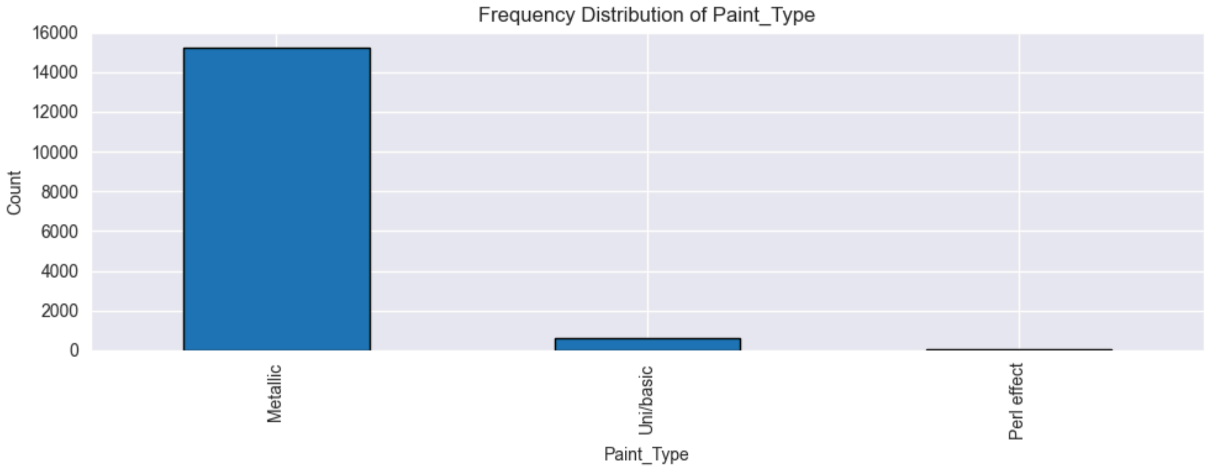
ABS,Central door lock,Daytime running lights,Driver-side airbag,Electronic stability control,Fog lights,Immo

ABS,Central door lock,Driver-si

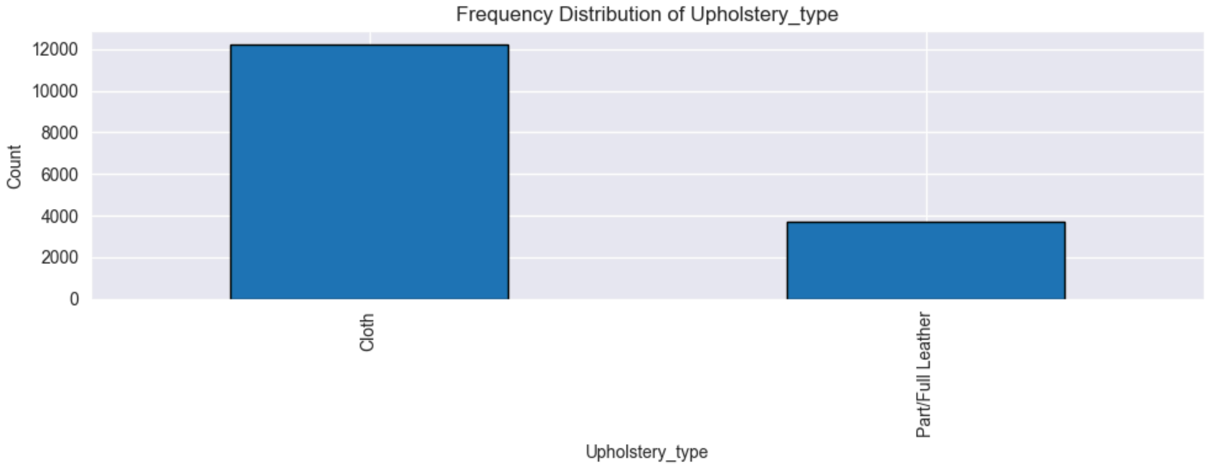
ABS,Central door lock,Daytime running light

ABS,C

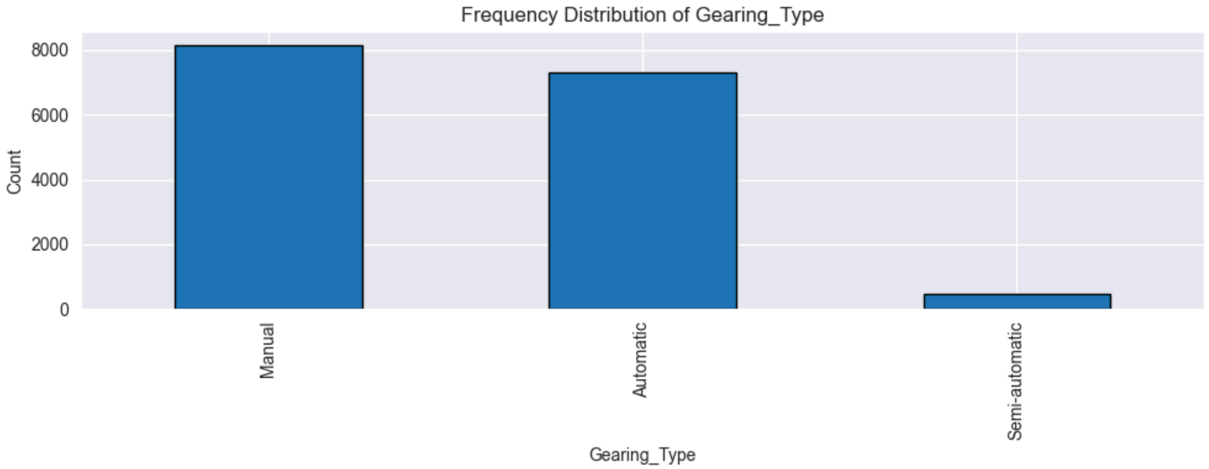
Safety_Security



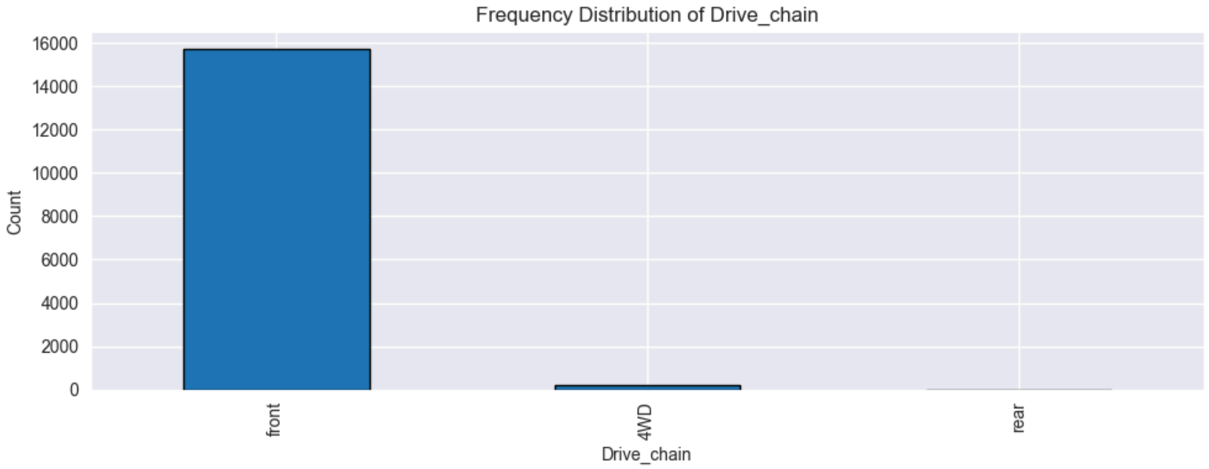
Frequency Distribution of Paint_Type



Frequency Distribution of Upholstery_type



Frequency Distribution of Gearing_Type



Note: Look carefully at the values stored in columns `["Comfort_Convenience", "Entertainment_Media", "Extras", "Safety_Security"]`.

Should they be considered categorical? Should they be dropped or handled any other way?

2.1.4 [3 marks]

Fix columns with low frequency values and class imbalances.

Some information regarding values in the `Type` column that may help:

- *'Pre-registered'* cars are ones which have already been registered previously by the seller.
- *'New'* cars are not necessarily new cars, but new-like cars. These might also have multiple owners due to multiple pre-registrations as well.
- *'Employee's car'* are cars used by employees over a short period of time and small distance.
- *'Demonstration'* cars are used for trial purposes and also driven for a short time and distance.

Based on these, you can handle this particular column. For other columns, decide a strategy on your own.

In [126…

```
# Fix columns as needed
# Fixing the 'Type' column based on domain logic
type_map = {
    "Pre-registered": "Nearly_New",
    "New": "Nearly_New",
    "Employee's car": "Nearly_New",
    "Demonstration": "Nearly_New"
}

df["Type"] = df["Type"].replace(type_map)

# Everything else becomes a generic used category
```

```
df["Type"] = df["Type"].fillna("Used_Regular")
df["Type"] = df["Type"].apply(lambda x: x if x in ["Nearly_New"] else "Used_



categorical_cols = df.select_dtypes(include=['object', 'category']).columns
categorical_cols = [col for col in categorical_cols]

# Function to merge rare categories (<1%) into 'Other'
def merge_rare_categories(series, threshold=0.01):
    freq = series.value_counts(normalize=True)
    rare = freq[freq < threshold].index
    return series.replace(rare, "Other")

# Apply globally
for col in categorical_cols:
    df[col] = merge_rare_categories(df[col])



df.head()
```

Out[126…

	make_model	body_type	price	vat	km	Type	Fuel	Gears
0	Audi A1	Sedans	15770	VAT deductible	56013.0	Used_Regular	Diesel	7.0
1	Audi A1	Sedans	14500	Price negotiable	80000.0	Used_Regular	Benzine	7.0
2	Audi A1	Sedans	14640	VAT deductible	83450.0	Used_Regular	Diesel	7.0
3	Audi A1	Sedans	14500	VAT deductible	73000.0	Used_Regular	Diesel	6.0
4	Audi A1	Sedans	16790	VAT deductible	16200.0	Used_Regular	Diesel	7.0

5 rows × 23 columns

2.1.5 [3 marks]

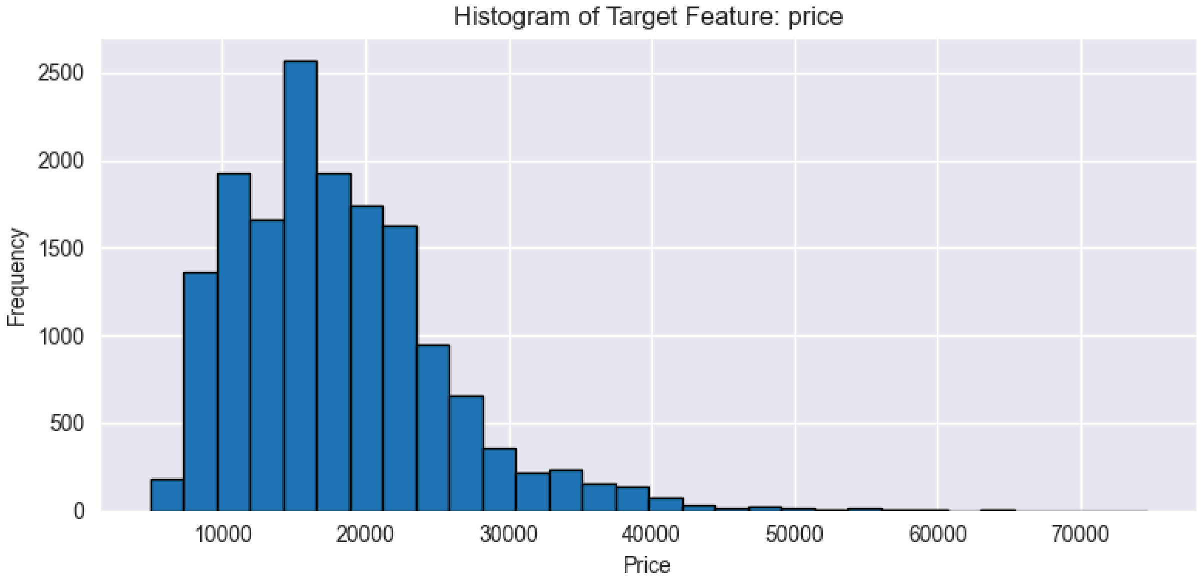
Identify target variable and plot the frequency distributions. Apply necessary transformations.

In [127…

```
# Plot histograms for target feature
import matplotlib.pyplot as plt


target = "price"   # Explicitly chosen based on dataset + project objective

print("Selected target variable:", target)
```

```
if df[target].dtype == "object":
    plt.figure(figsize=(8,4))
    df[target].value_counts().plot(kind="bar", edgecolor="black")
    plt.title(f"Frequency Distribution of Target: {target}")
    plt.xlabel(target)
    plt.ylabel("Count")
    plt.tight_layout()
    plt.show()



else:
    plt.figure(figsize=(8,4))
    plt.hist(df[target], bins=30, edgecolor="black")
    plt.title(f"Histogram of Target Feature: {target}")
    plt.xlabel("Price")
    plt.ylabel("Frequency")
    plt.tight_layout()
    plt.show()
```

```
Selected target variable: price
```

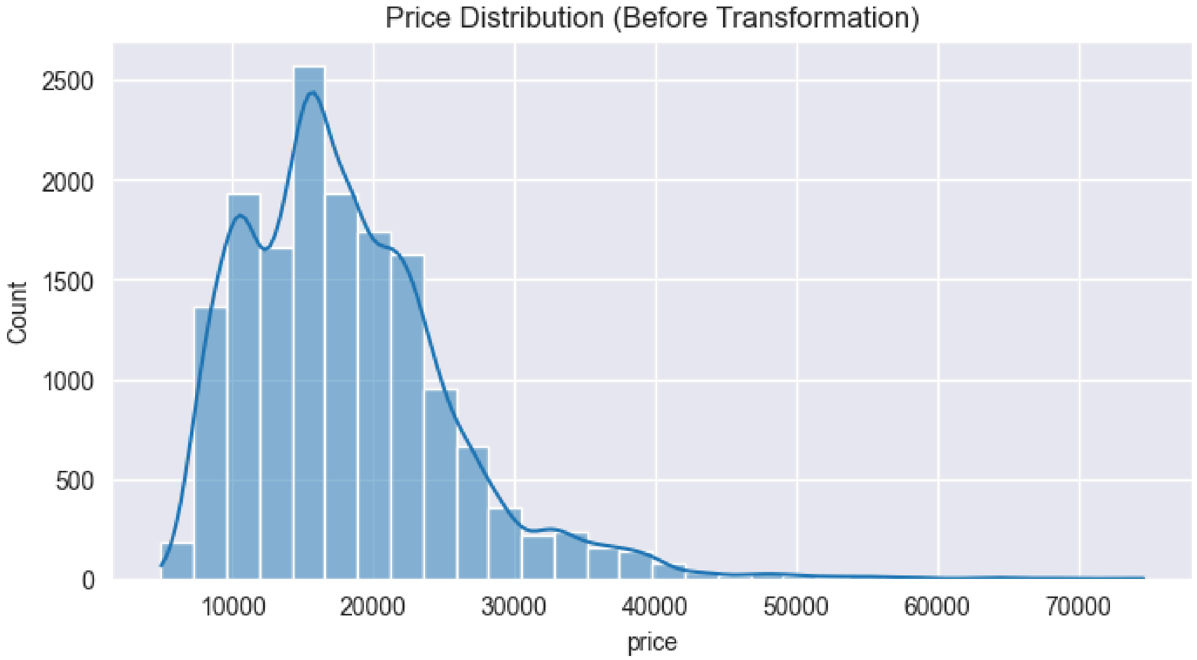


The target variable seems to be skewed. Perform suitable transformation on the target.

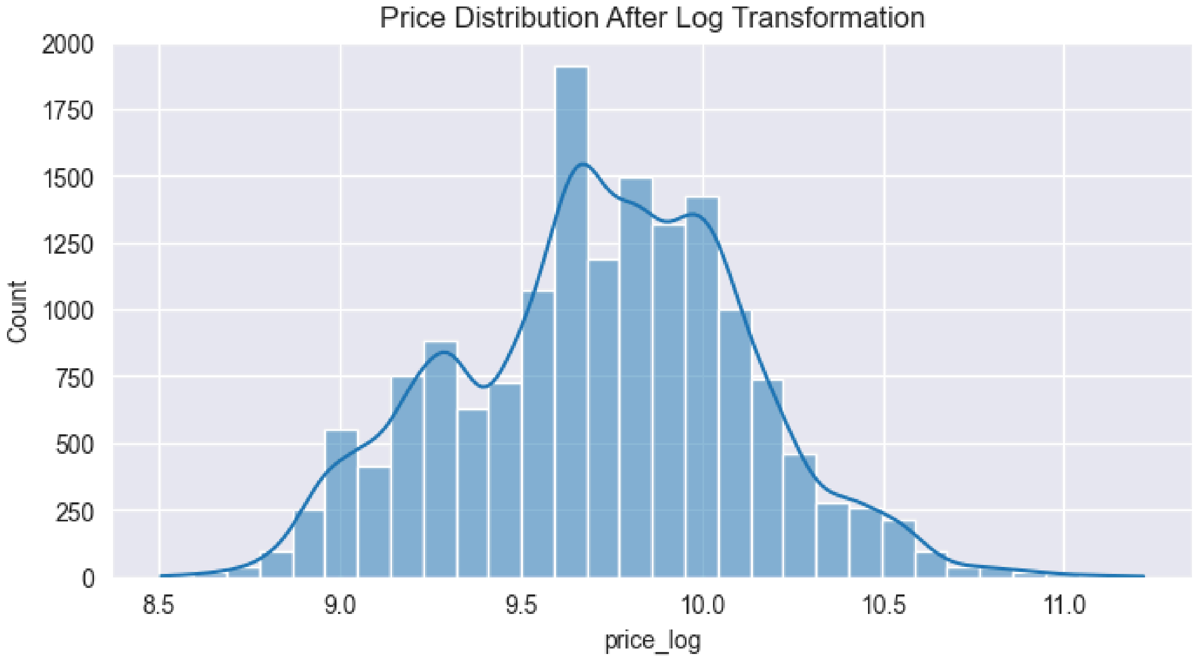
In [128…

```
# Transform the target feature

import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8,4))
sns.histplot(df["price"], bins=30, kde=True)
plt.title("Price Distribution (Before Transformation)")
plt.show()
```

```
print("Skewness before transformation:", df["price"].skew())

# Apply log transformation
df["price_log"] = np.log1p(df["price"])

print("Skewness after log transformation:", df["price_log"].skew())

plt.figure(figsize=(8,4))
sns.histplot(df["price_log"], bins=30, kde=True)
plt.title("Price Distribution After Log Transformation")
plt.show()
```



```
Skewness before transformation: 1.236169412899669
Skewness after log transformation: -0.0314736417467197
```



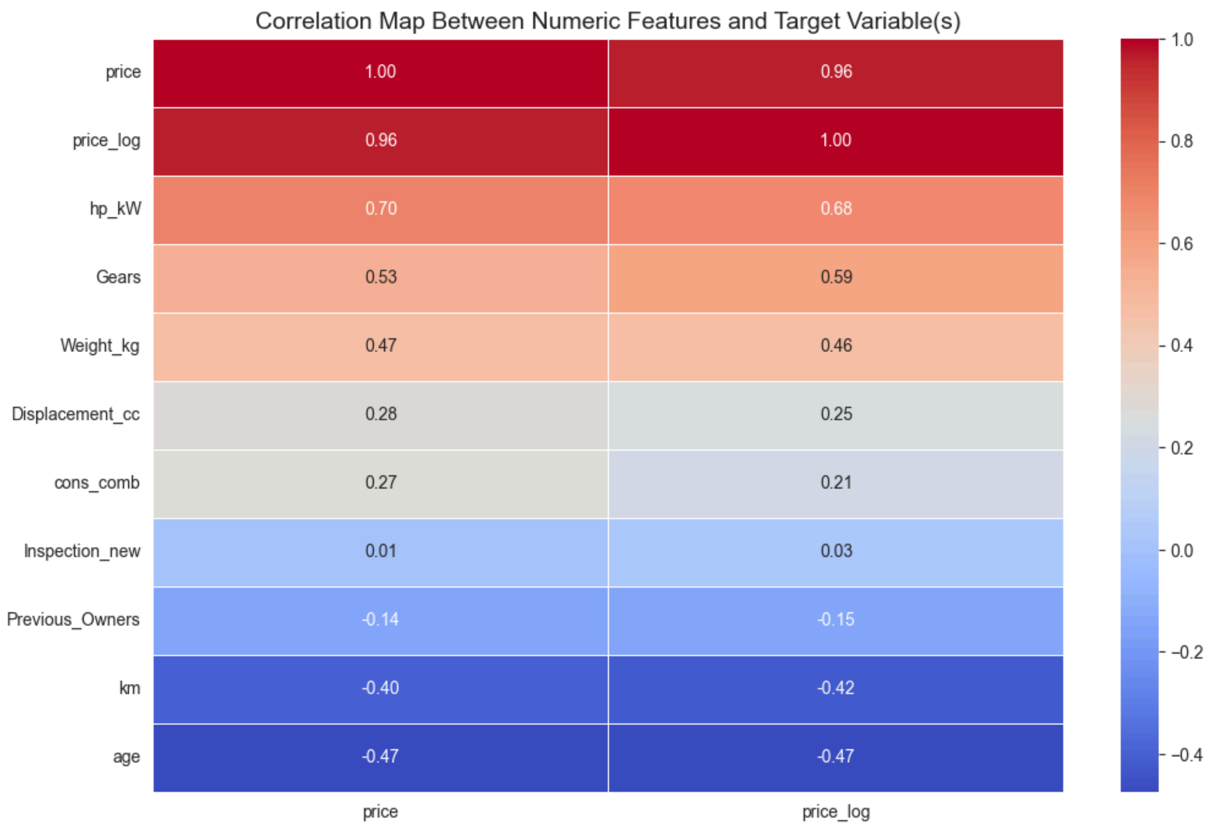
2.2 Correlation analysis [6 marks]

2.2.1 [3 marks]

Plot the correlation map between features and target variable.

In [129…

```
# Visualise correlation
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt

# Select numeric columns
numeric_df = df.select_dtypes(include=['int64', 'float64'])

# Include both original and transformed target
targets = ["price"]
if "price_log" in df.columns:
    targets.append("price_log")

corr_matrix = numeric_df[targets + [col for col in numeric_df.columns if col

plt.figure(figsize=(12, 8))
sns.heatmap(
    corr_matrix[[target for target in targets]].sort_values(by=targets, asce
    annot=True,
    cmap="coolwarm",
    fmt=".2f",
    linewidths=.5
)
plt.title("Correlation Map Between Numeric Features and Target Variable(s)",
plt.show()
```



2.2.2 [3 marks]

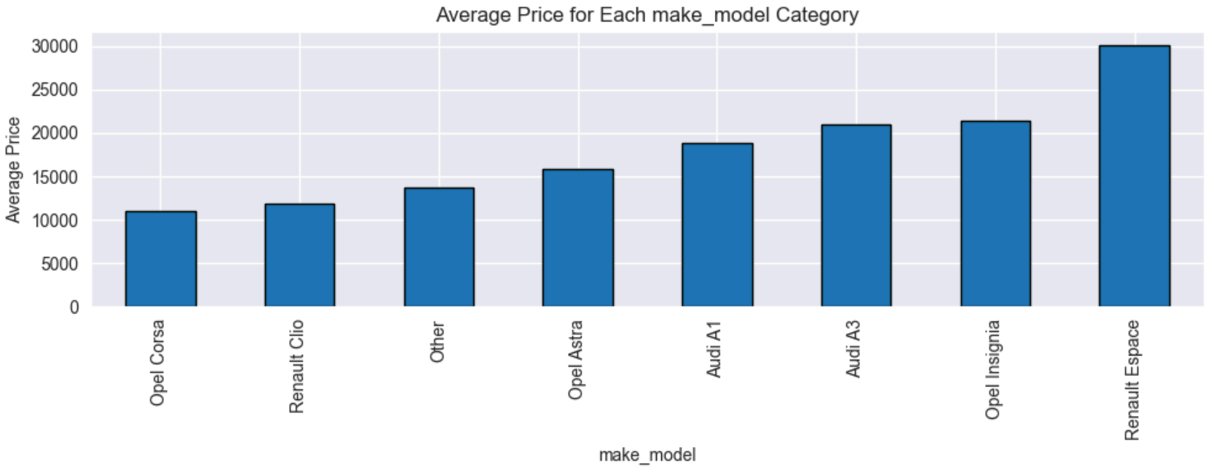
Analyse correlation between categorical features and target variable.

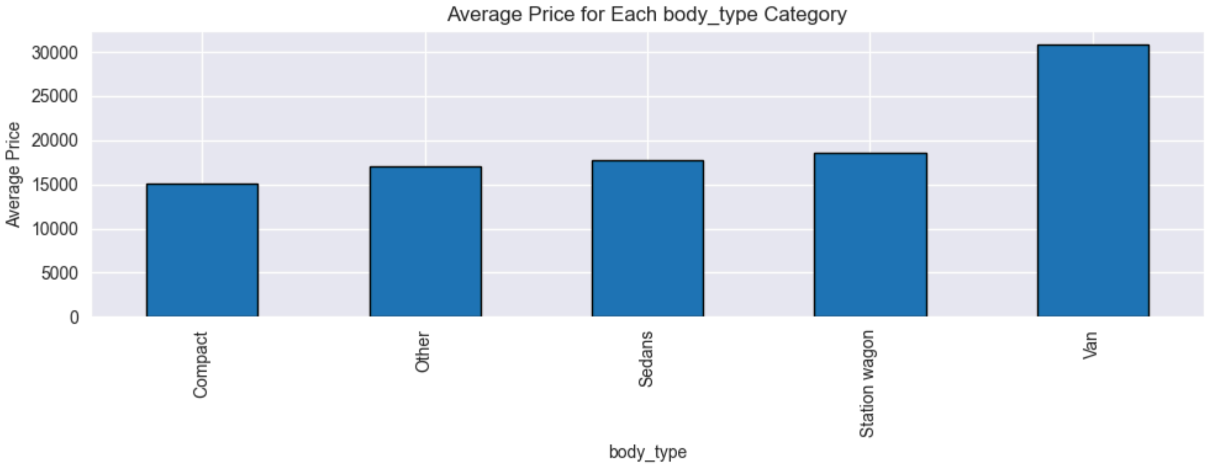
In [130…

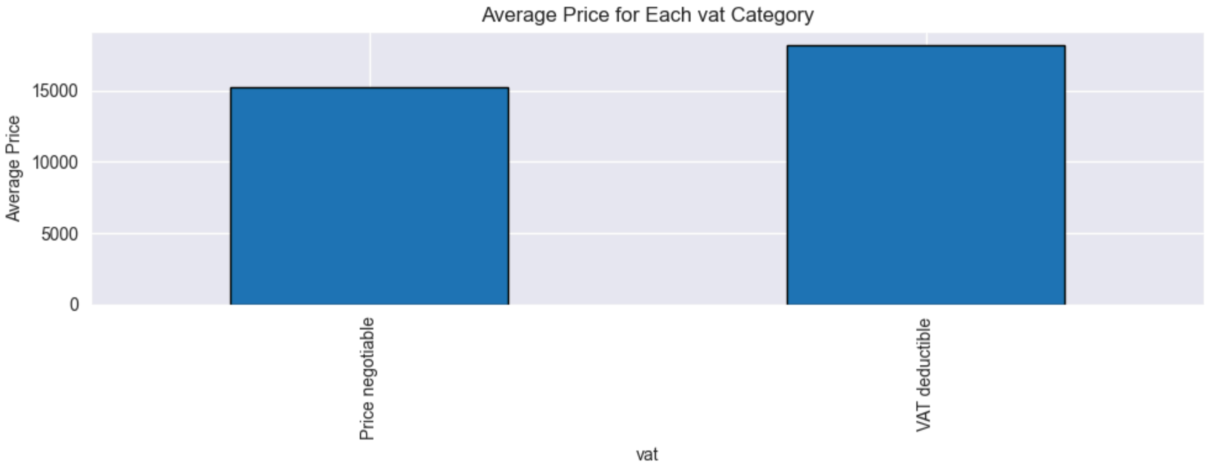
```
# Comparing average values of target for different categories

category_analysis = {}

for col in categorical_cols:
    summary = df.groupby(col)[["price", "price_log"]].agg(
        avg_price = ("price", "mean"),
        avg_price_log = ("price_log", "mean"),
        count = ("price", "count")
    ).sort_values("avg_price", ascending=False)

    category_analysis[col] = summary
    print(f"\n===== {col} =====")
    print(summary.head(10))
    # show top 10 categories

    import matplotlib.pyplot as plt

for col in categorical_cols:
    if df[col].nunique() <= 15:   # avoid exploding plots
        plt.figure(figsize=(10,4))
        df.groupby(col)["price"].mean().sort_values().plot(kind="bar", edgec
        plt.title(f"Average Price for Each {col} Category")
        plt.xlabel(col)
        plt.ylabel("Average Price")
        plt.tight_layout()
        plt.show()
```

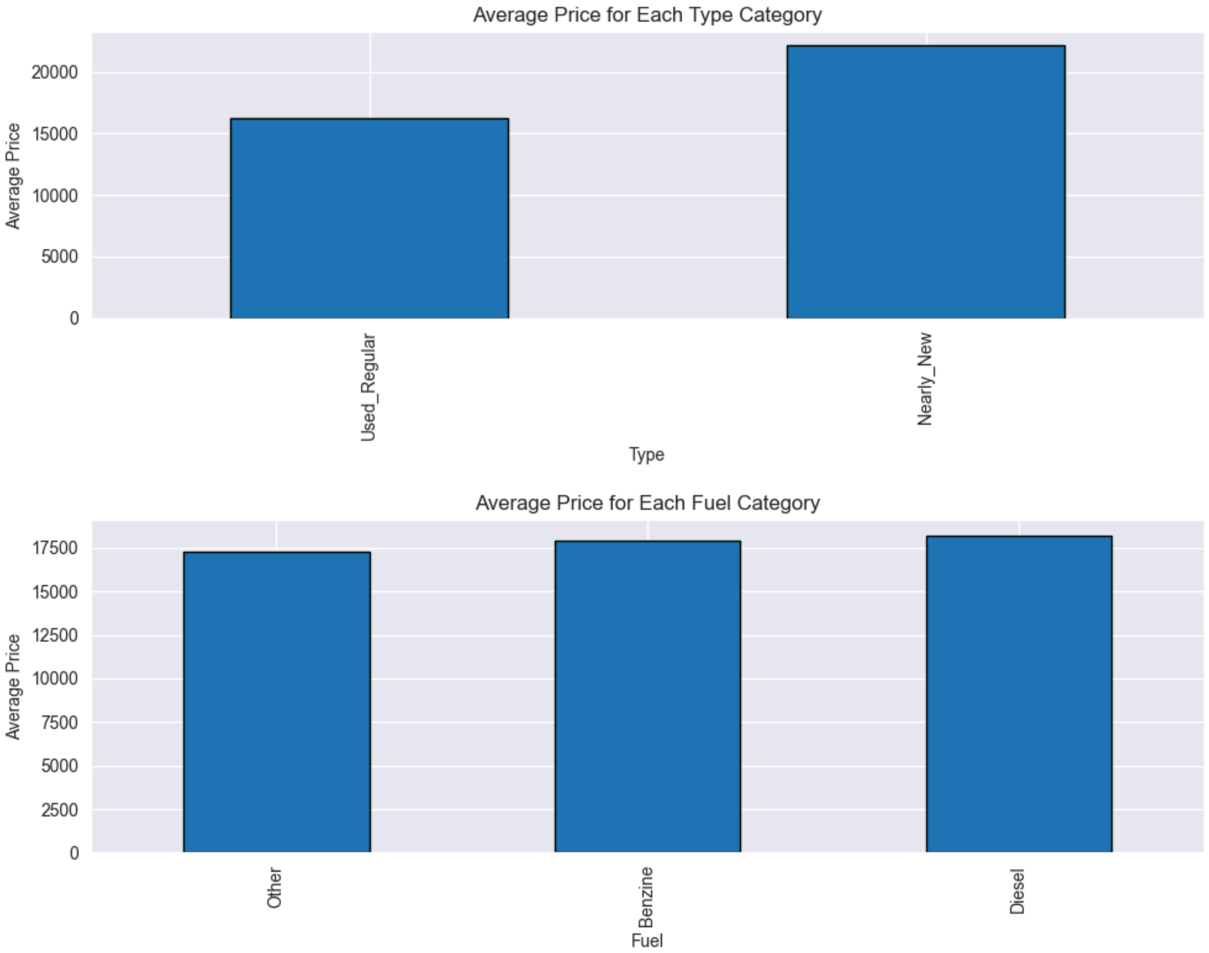
```
===== make_model =====
                   avg_price  avg_price_log  count
make_model
Renault Espace  30080.211907      10.271724    991
Opel Insignia   21463.451886       9.912658   2598
Audi A3         20996.693252       9.928165   3097
Audi A1         18864.688982       9.817643   2614
Opel Astra      15840.834059       9.626041   2525
Other           13657.885714       9.504893     35
Renault Clio    11940.320827       9.335822   1839
Opel Corsa      11061.841606       9.276059   2216

===== body_type =====
                  avg_price  avg_price_log  count
body_type
Van            30789.209302      10.294540    817
Station wagon  18542.296981       9.753456   3677
Sedans         17699.040480       9.716144   8004
Other          17023.604520       9.622932    177
Compact        15076.206481       9.555735   3240

===== vat =====
                     avg_price  avg_price_log  count
vat
VAT deductible    18185.887131       9.730286  15044
Price negotiable  15234.823192       9.556299    871

===== Type =====
                avg_price  avg_price_log  count
Type
Nearly_New    22127.53278       9.928387   4820
Used_Regular  16241.84849       9.630566  11095

===== Fuel =====
            avg_price  avg_price_log  count
Fuel
Diesel   18177.407783       9.734926   7298
Benzine  17899.681329       9.709328   8548
Other    17287.231884       9.639463     69

===== Comfort_Convenience =====
                                                      avg_price  \
Comfort_Convenience
Air conditioning,Armrest,Automatic climate cont...  23292.099379
Air conditioning,Automatic climate control,Crui...  20244.243750
Other                                               18172.657168
Air conditioning,Armrest,Automatic climate cont...  17637.422460
Air conditioning,Electrical side mirrors,Hill H...   9370.036082

                                                    avg_price_log  count
Comfort_Convenience
Air conditioning,Armrest,Automatic climate cont...      10.017481    161
Air conditioning,Automatic climate control,Crui...       9.898748    160
Other                                                    9.730398  15019
Air conditioning,Armrest,Automatic climate cont...       9.762393    187
Air conditioning,Electrical side mirrors,Hill H...       9.131270    388
```

```
===== Entertainment_Media =====
                                                       avg_price  \
Entertainment_Media
Bluetooth,Digital radio,Hands-free equipment,On...  23972.817544
Bluetooth,Hands-free equipment,On-board compute...  22785.580583
Bluetooth,Digital radio,Hands-free equipment,MP...  21847.373016
Bluetooth,Radio                                     20092.339483
Bluetooth,CD player,Digital radio,Hands-free eq...  19290.977143
Other                                               19123.484680
Bluetooth,CD player,Hands-free equipment,MP3,On...  19100.259939
Bluetooth,CD player,Hands-free equipment,MP3,On...  17878.138614
Bluetooth,Hands-free equipment,On-board compute...  17709.803191
Radio                                               17629.154122

                                                    avg_price_log  count
Entertainment_Media
Bluetooth,Digital radio,Hands-free equipment,On...      10.018384    285
Bluetooth,Hands-free equipment,On-board compute...       9.963690    515
Bluetooth,Digital radio,Hands-free equipment,MP...       9.911307    252
Bluetooth,Radio                                          9.857507    271
Bluetooth,CD player,Digital radio,Hands-free eq...       9.784390    175
Other                                                    9.770708   5940
Bluetooth,CD player,Hands-free equipment,MP3,On...       9.819240    327
Bluetooth,CD player,Hands-free equipment,MP3,On...       9.742608    404
Bluetooth,Hands-free equipment,On-board compute...       9.743829    188
Radio                                                    9.701190    558

===== Extras =====
                                                      avg_price  \
Extras
Alloy wheels,Sport suspension                      21285.880682
Alloy wheels,Roof rack,Touch screen,Voice Control  20755.177083
Alloy wheels,Catalytic Converter,Voice Control     20631.334975
Alloy wheels,Touch screen                          20254.259684
Alloy wheels,Sport seats,Sport suspension          20104.918803
Alloy wheels,Touch screen,Voice Control            19978.365809
Alloy wheels,Roof rack                             19893.614367
Alloy wheels,Catalytic Converter,Touch screen      19619.412281
Alloy wheels,Voice Control                         19382.783505
Other                                              19341.330786

                                                   avg_price_log  count
Extras
Alloy wheels,Sport suspension                           9.942534    176
Alloy wheels,Roof rack,Touch screen,Voice Control       9.896251    192
Alloy wheels,Catalytic Converter,Voice Control          9.892297    203
Alloy wheels,Touch screen                               9.825623    697
Alloy wheels,Sport seats,Sport suspension               9.870663    234
Alloy wheels,Touch screen,Voice Control                 9.831347    544
Alloy wheels,Roof rack                                  9.836694    529
Alloy wheels,Catalytic Converter,Touch screen           9.813442    228
Alloy wheels,Voice Control                              9.785768    582
Other                                                   9.807676   4710

===== Safety_Security =====
                                                       avg_price  \
Safety_Security
ABS,Central door lock,Daytime running lights,Dr...  20902.099432
Other                                               18689.712729
ABS,Central door lock,Daytime running lights,Dr...  17821.223048
ABS,Central door lock,Daytime running lights,Dr...  15101.008043
ABS,Central door lock,Daytime running lights,Dr...  14085.145405
ABS,Central door lock,Daytime running lights,Dr...  13811.362745
ABS,Central door lock,Daytime running lights,Dr...  12508.612500
ABS,Central door lock,Daytime running lights,Dr...  11770.559441

                                                    avg_price_log  count
Safety_Security
ABS,Central door lock,Daytime running lights,Dr...       9.930039    352
Other                                                    9.755765  13120
ABS,Central door lock,Daytime running lights,Dr...       9.774915    269
ABS,Central door lock,Daytime running lights,Dr...       9.596806    373
ABS,Central door lock,Daytime running lights,Dr...       9.510400    729
ABS,Central door lock,Daytime running lights,Dr...       9.448312    306
ABS,Central door lock,Daytime running lights,Dr...       9.394061    480
ABS,Central door lock,Daytime running lights,Dr...       9.344273    286

===== Paint_Type =====
               avg_price  avg_price_log  count
Paint_Type
Metallic    18095.052866       9.725404  15246
Uni/basic   16745.040816       9.638642    637
Other        9820.312500       9.144565     32

===== Upholstery_type =====
                      avg_price  avg_price_log  count
Upholstery_type
Part/Full Leather  23226.196142       9.981383   3681
Cloth              16459.243829       9.642348  12234

===== Gearing_Type =====
                   avg_price  avg_price_log  count
Gearing_Type
Semi-automatic  23236.562900       9.972004    469
Automatic       21163.176237       9.906177   7297
Manual          14913.777396       9.540276   8149

===== Drive_chain =====
                avg_price  avg_price_log  count
Drive_chain
4WD          28261.416667      10.133491    204
front        17892.109123       9.715432  15707
Other        15332.500000       9.606166      4
```

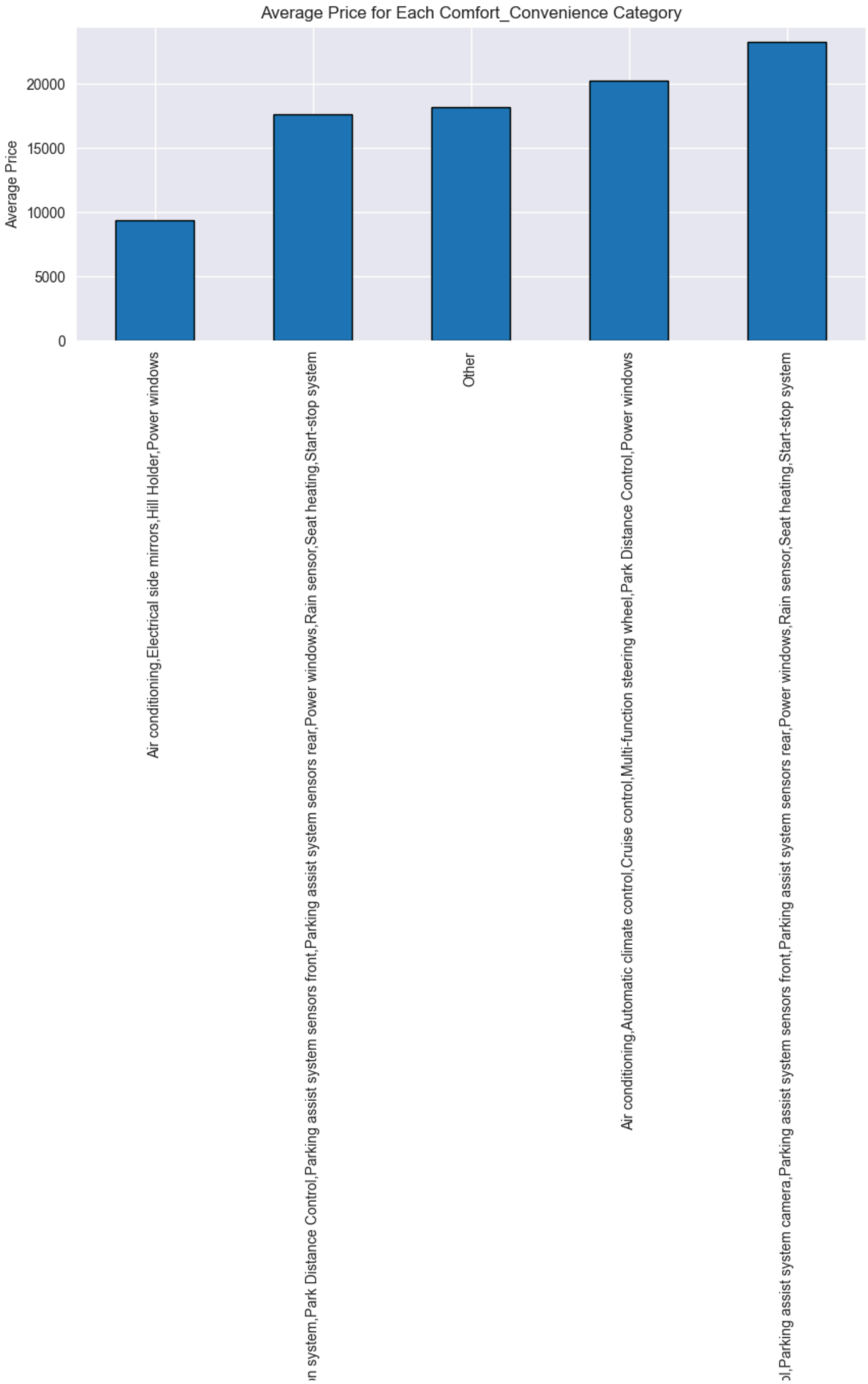

Average Price for Each make_model Category


Average Price for Each body_type Category


Average Price for Each vat Category

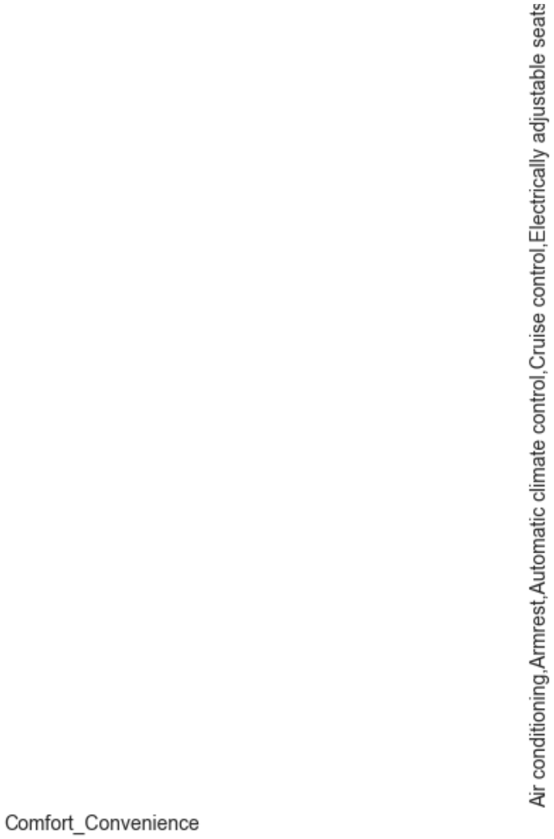


```
/var/folders/q9/_3w68hr55rdcl70s4bt4jjbh0000gn/T/ipykernel_66612/1065706893.
py:26: UserWarning: Tight layout not applied. The bottom and top margins can
not be made large enough to accommodate all Axes decorations.
  plt.tight_layout()
```

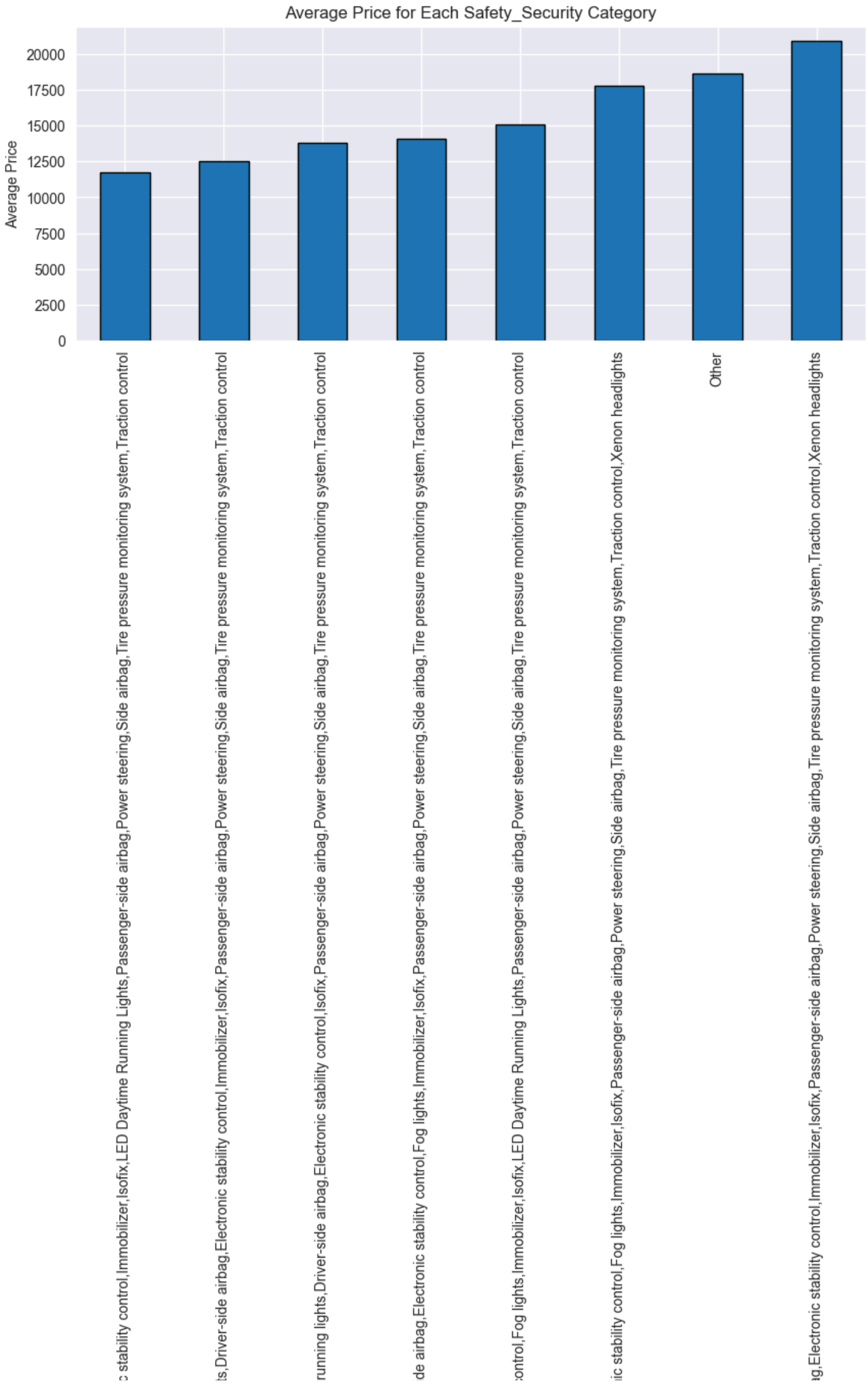

Average Price for Each Comfort_Convenience Category
Average Price
0
5000
10000
15000
20000
Air conditioning,Electrical side mirrors,Hill Holder,Power windows
n system,Park Distance Control,Parking assist system sensors front,Parking assist system sensors rear,Power windows,Rain sensor,Seat heating,Start-stop system
Other
Air conditioning,Automatic climate control,Cruise control,Multi-function steering wheel,Park Distance Control,Power windows
l,Parking assist system camera,Parking assist system sensors front,Parking assist system sensors rear,Power windows,Rain sensor,Seat heating,Start-stop system

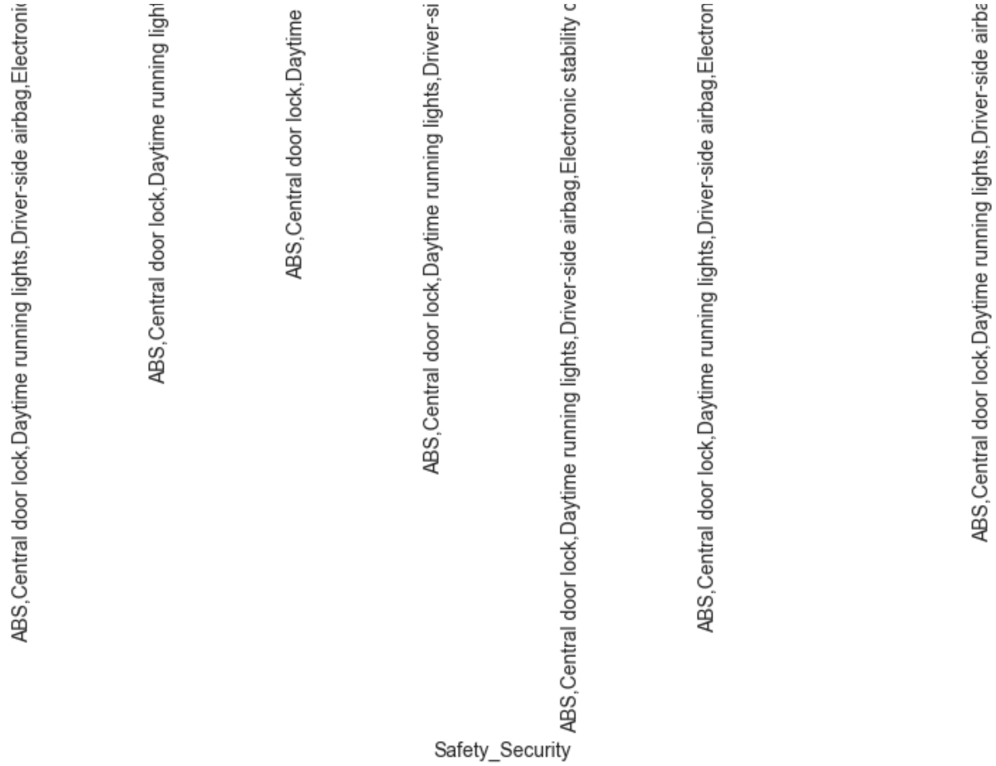
Air conditioning,Armrest,Automatic climate control,Cruise control,Electrical side mirrors,Leather steering wheel,Light sensor,Lumbar support,Multi-function steering wheel,Navigatic

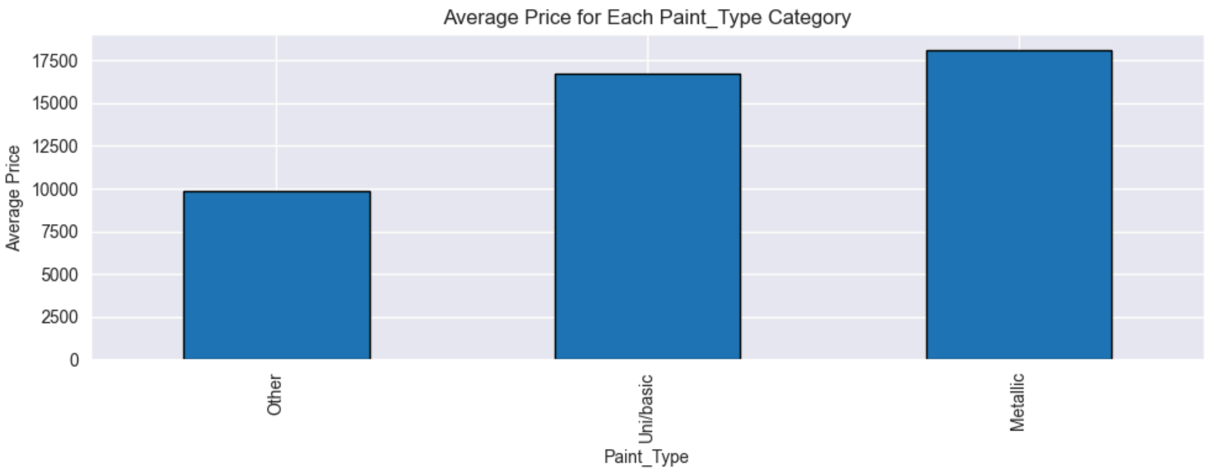
s,Electrical side mirrors,Electric tailgate,Heated steering wheel,Hill Holder,Keyless central door lock,Leather steering wheel,Light sensor,Lumbar support,Multi-function steering wheel,Navigation system,Park Distance Contr

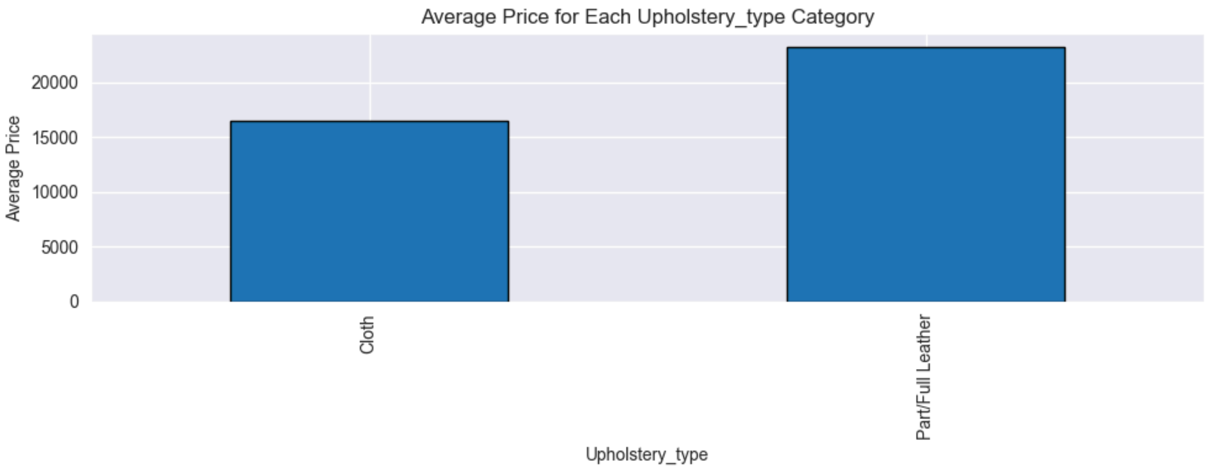


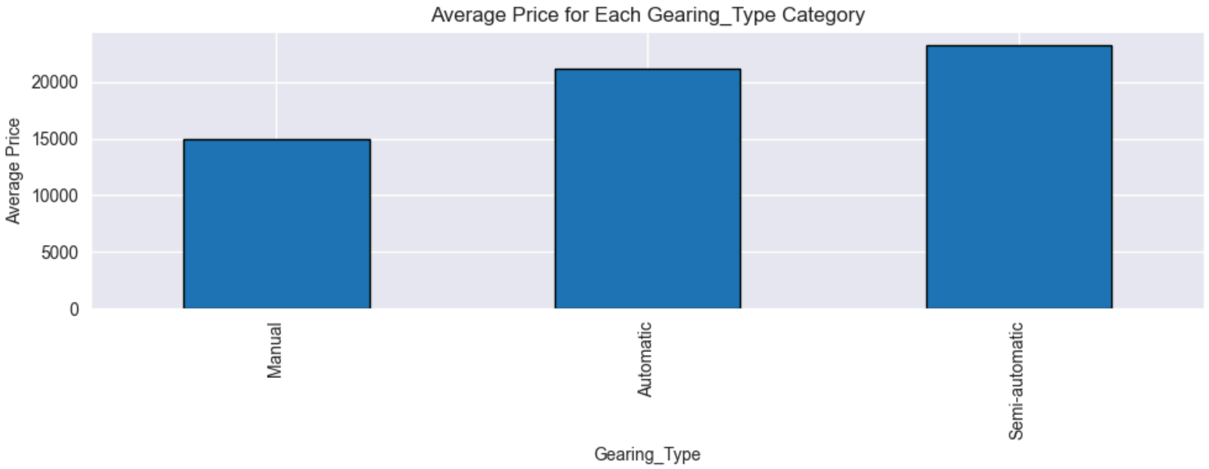
```
/var/folders/q9/_3w68hr55rdcl70s4bt4jjbh0000gn/T/ipykernel_66612/1065706893.
py:26: UserWarning: Tight layout not applied. The bottom and top margins can
not be made large enough to accommodate all Axes decorations.
  plt.tight_layout()
```

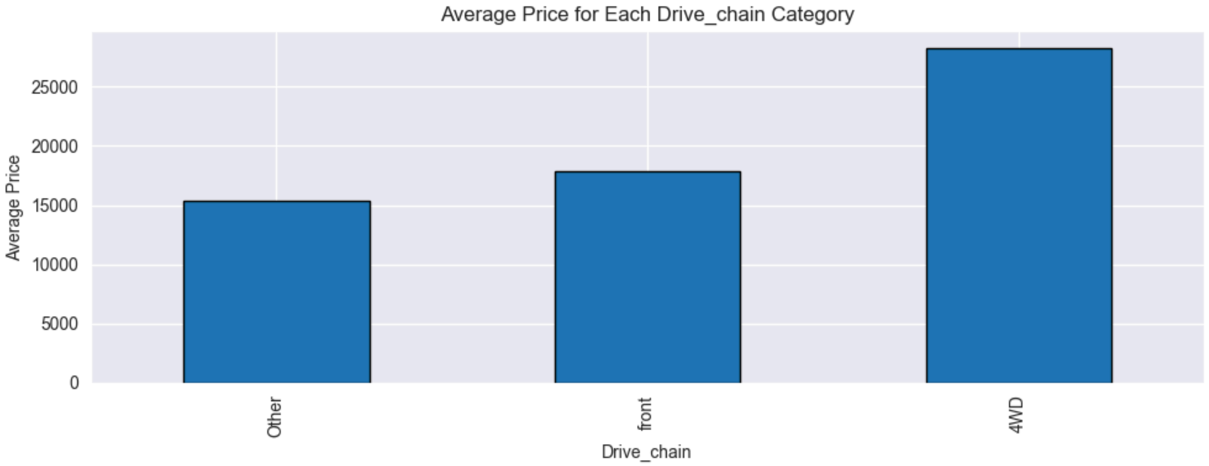

Average Price for Each Safety_Security Category
Average Price
0
2500
5000
7500
10000
12500
15000
17500
20000
c stability control,Immobilizer,Isofix,LED Daytime Running Lights,Passenger-side airbag,Power steering,Side airbag,Tire pressure monitoring system,Traction control
ts,Driver-side airbag,Electronic stability control,Immobilizer,Isofix,Passenger-side airbag,Power steering,Side airbag,Tire pressure monitoring system,Traction control
running lights,Driver-side airbag,Electronic stability control,Isofix,Passenger-side airbag,Power steering,Side airbag,Tire pressure monitoring system,Traction control
de airbag,Electronic stability control,Fog lights,Immobilizer,Isofix,Passenger-side airbag,Power steering,Side airbag,Tire pressure monitoring system,Traction control
control,Fog lights,Immobilizer,Isofix,LED Daytime Running Lights,Passenger-side airbag,Power steering,Side airbag,Tire pressure monitoring system,Traction control
ic stability control,Fog lights,Immobilizer,Isofix,Passenger-side airbag,Power steering,Side airbag,Tire pressure monitoring system,Traction control,Xenon headlights
Other
ag,Electronic stability control,Immobilizer,Isofix,Passenger-side airbag,Power steering,Side airbag,Tire pressure monitoring system,Traction control,Xenon headlights


ABS,Central door lock,Daytime running lights,Driver-side airbag,Electroni
ABS,Central door lock,Daytime running ligh
ABS,Central door lock,Daytime
ABS,Central door lock,Daytime running lights,Driver-si
ABS,Central door lock,Daytime running lights,Driver-side airbag,Electronic stability c
ABS,Central door lock,Daytime running lights,Driver-side airbag,Electron
ABS,Central door lock,Daytime running lights,Driver-side airba
Safety_Security


Average Price for Each Paint_Type Category
Average Price
0
2500
5000
7500
10000
12500
15000
17500
Other
Uni/basic
Metallic
Paint_Type


Average Price for Each Upholstery_type Category
Average Price
0
5000
10000
15000
20000
Cloth
Part/Full Leather
Upholstery_type


Average Price for Each Gearing_Type Category


Average Price for Each Drive_chain Category

2.3 Outlier analysis [5 marks]

2.3.1 [2 marks]

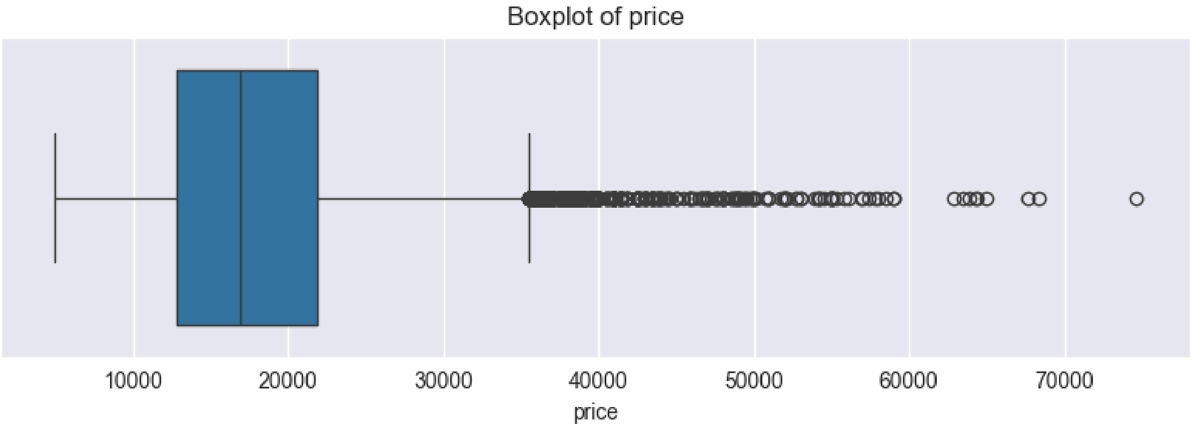
Identify potential outliers in the data.

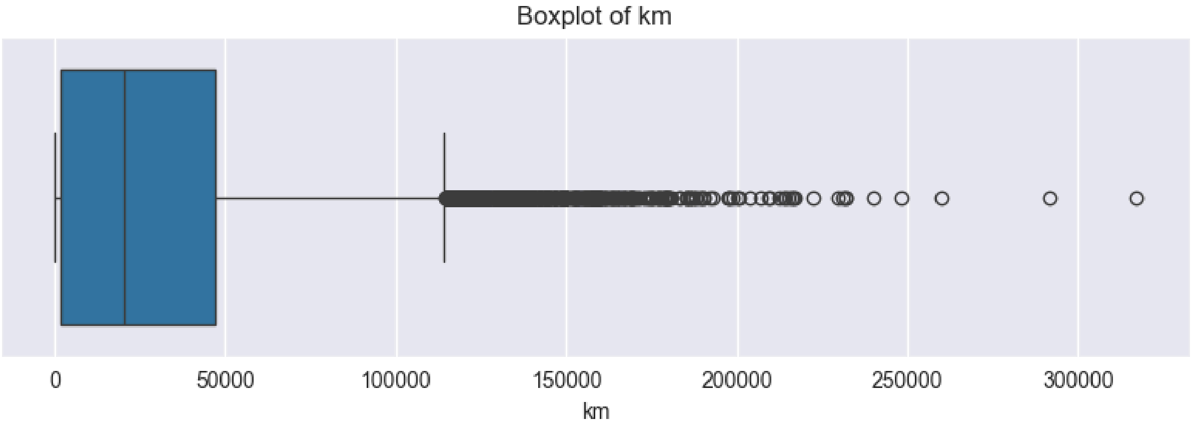
In [131…

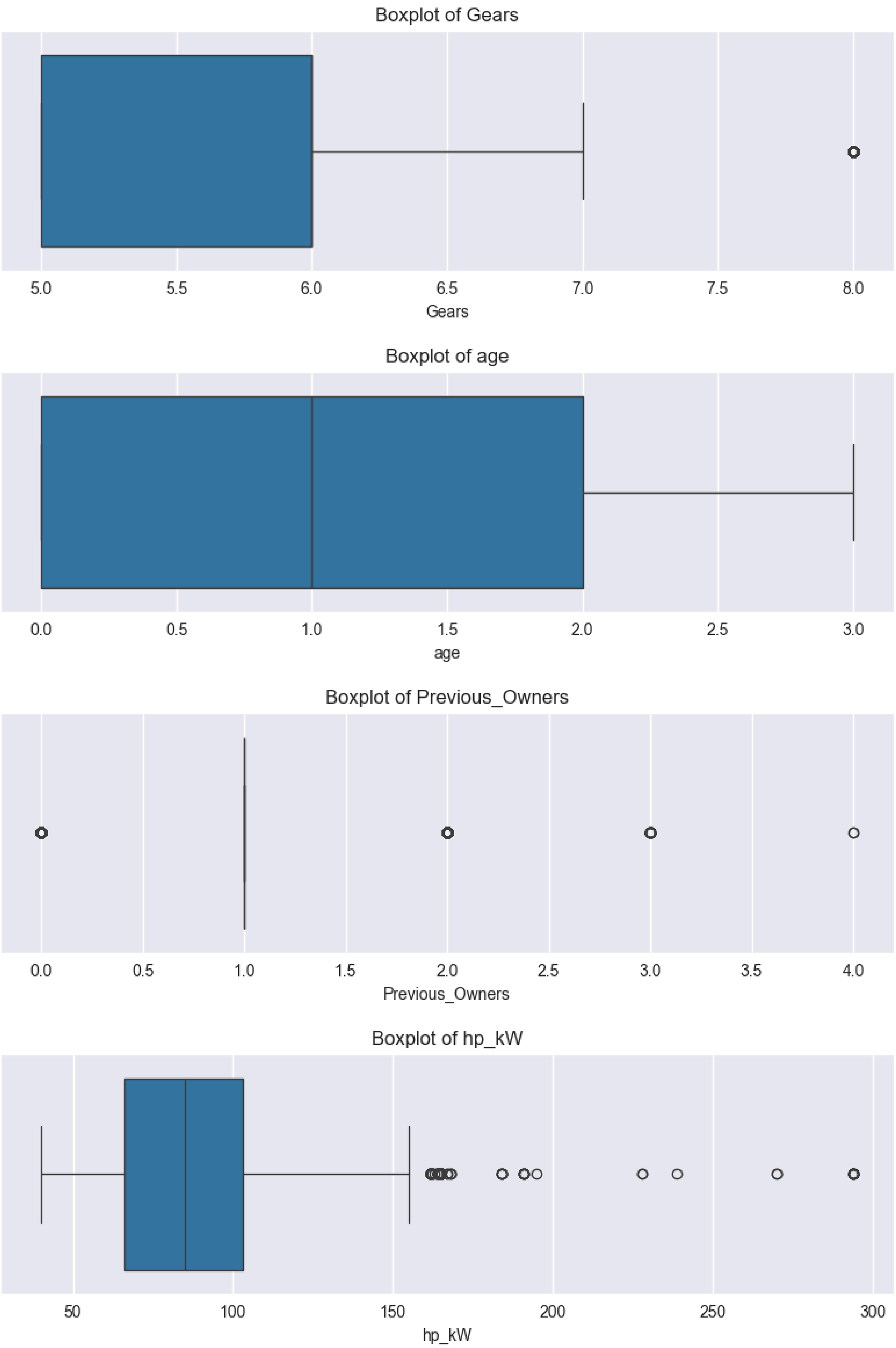
```
# Outliers present in each column
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns
numeric_cols
outlier_summary = {}

for col in numeric_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower = Q1 - 1.5 * IQR
    upper = Q3 + 1.5 * IQR

    outliers = df[(df[col] < lower) | (df[col] > upper)][col]
    outlier_summary[col] = len(outliers)

# Print summary
print("Outlier counts per numeric column:")
for col, count in outlier_summary.items():
    print(f"{col}: {count}")
```

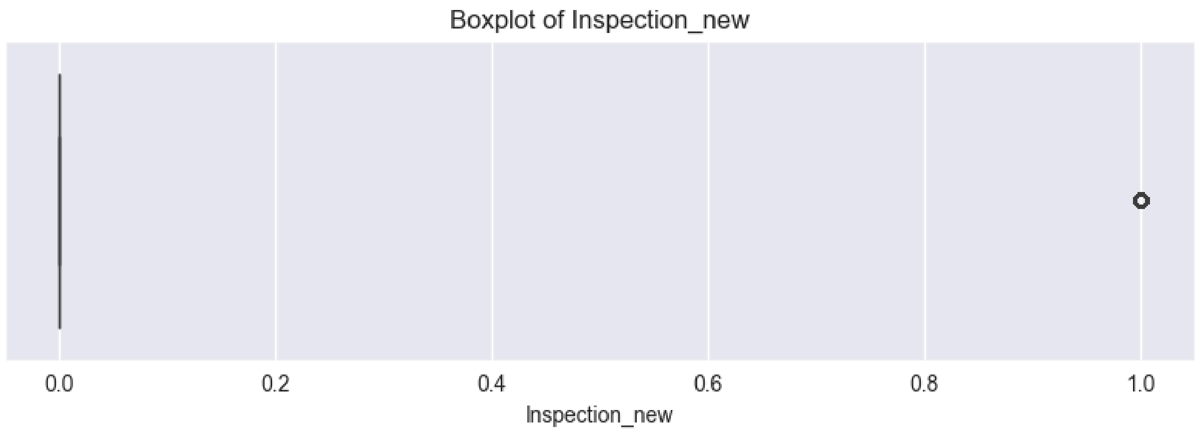
```python
import matplotlib.pyplot as plt
import seaborn as sns

for col in numeric_cols:
    plt.figure(figsize=(8, 3))
    sns.boxplot(x=df[col])
    plt.title(f"Boxplot of {col}")
    plt.tight_layout()
    plt.show()
```

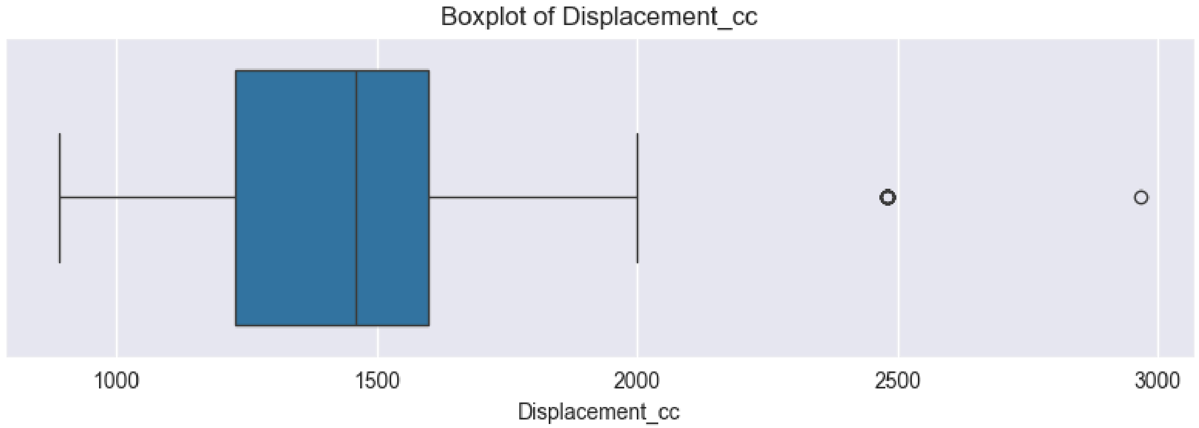
```
Outlier counts per numeric column:
price: 479
km: 689
Gears: 225
age: 0
Previous_Owners: 1757
hp_kW: 361
Inspection_new: 3932
Displacement_cc: 21
Weight_kg: 87
cons_comb: 125
price_log: 71
```

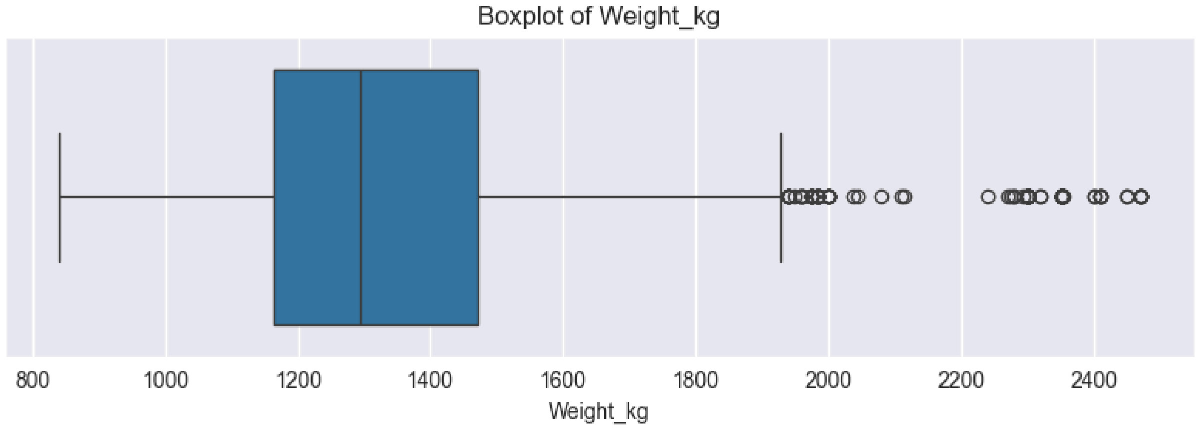


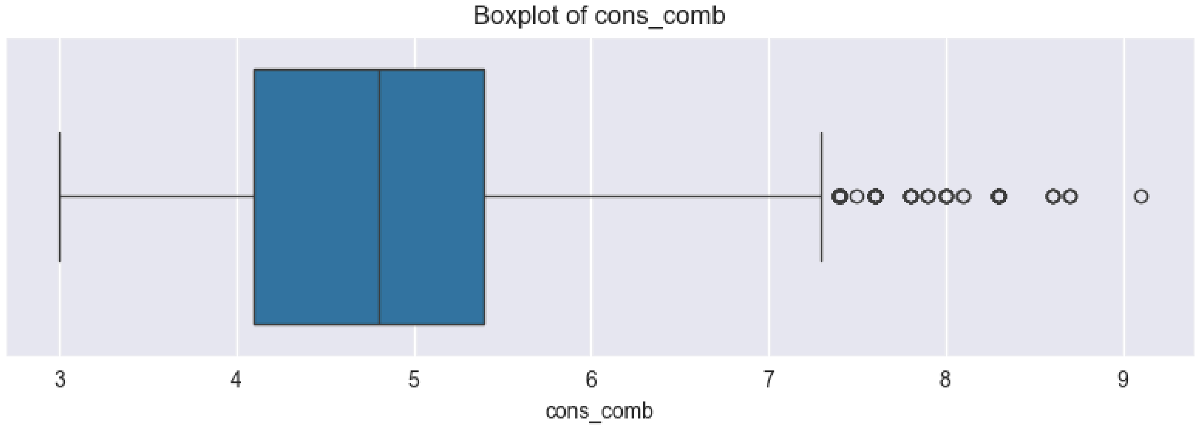


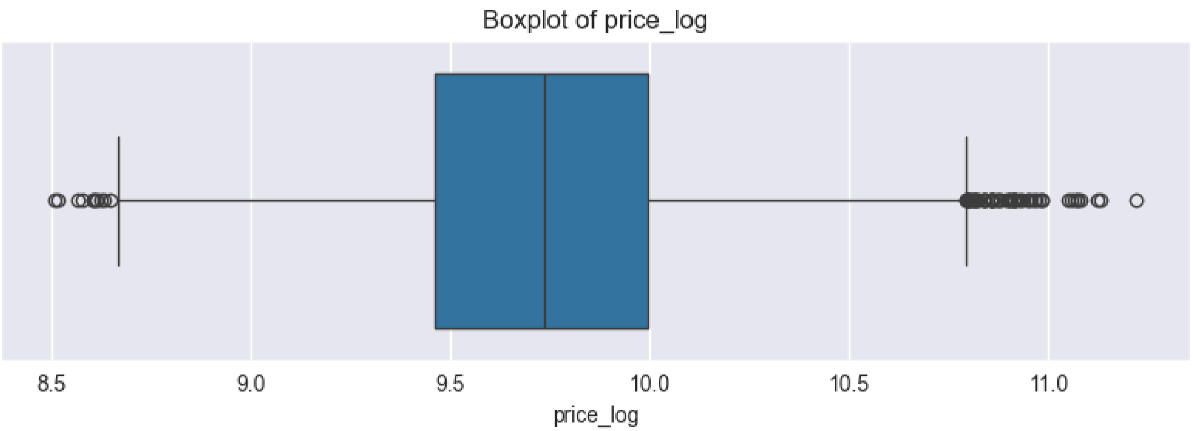

Boxplot of Gears
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Gears
Boxplot of age
0.0
0.5
1.0
1.5
2.0
2.5
3.0
age
Boxplot of Previous_Owners
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Previous_Owners
Boxplot of hp_kW
50
100
150
200
250
300
hp_kW


Boxplot of Inspection_new
0.0
0.2
0.4
0.6
0.8
1.0
Inspection_new


Boxplot of Displacement_cc
1000
1500
2000
2500
3000
Displacement_cc


Boxplot of Weight_kg
800
1000
1200
1400
1600
1800
2000
2200
2400
Weight_kg


Boxplot of cons_comb
3
4
5
6
7
8
9
cons_comb



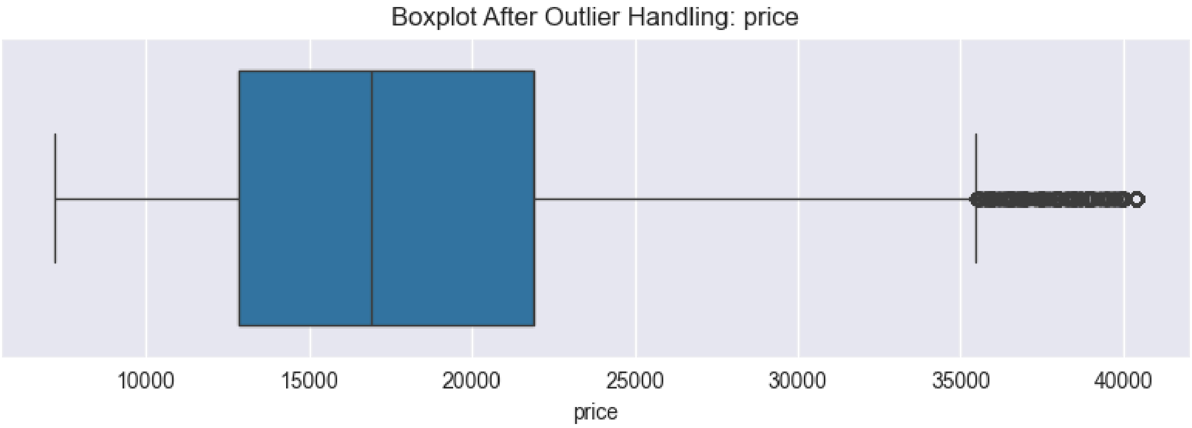
2.3.2 [3 marks]

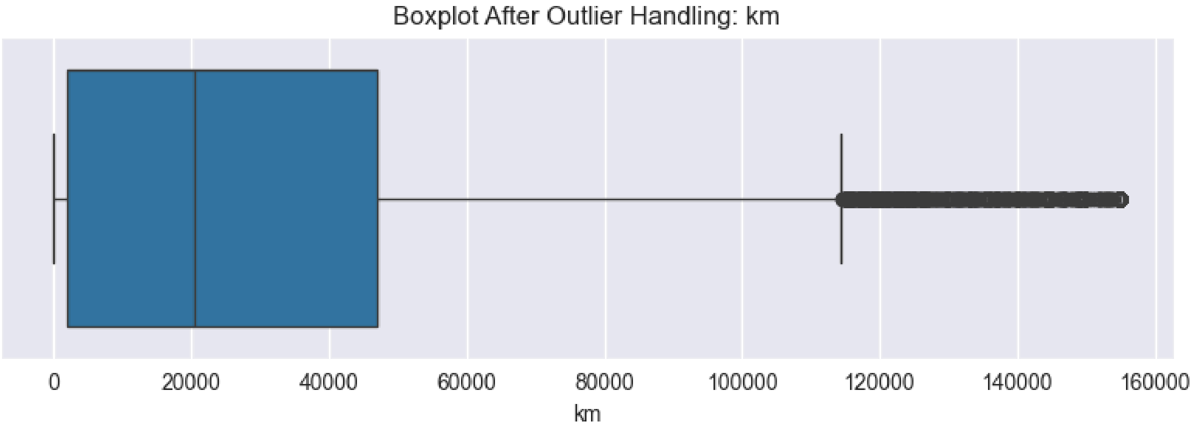
Handle the outliers suitably.

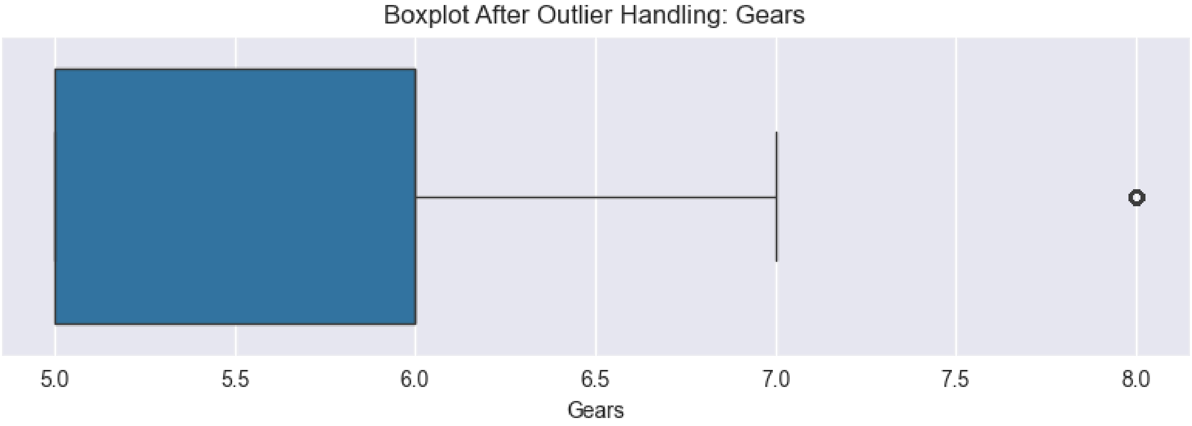
In [132…

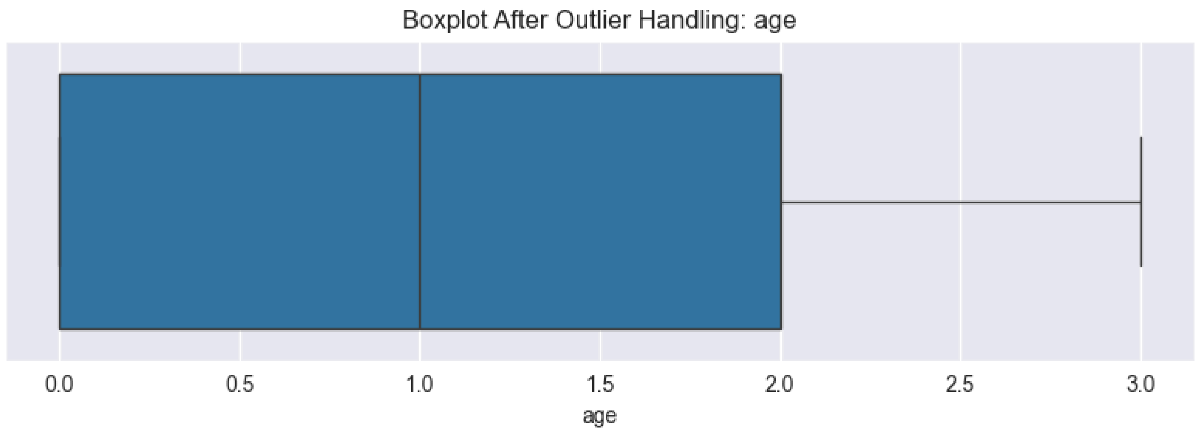
```
# Handle outliers
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns.tolist
numeric_cols

def winsorize(series, lower=0.01, upper=0.99):
    lower_bound = series.quantile(lower)
    upper_bound = series.quantile(upper)
    return series.clip(lower_bound, upper_bound)

numeric_cols_to_cap = [col for col in numeric_cols if col not in ["price_log

for col in numeric_cols_to_cap:
    df[col] = winsorize(df[col])

from scipy.stats import zscore

outliers_post = (abs(df[numeric_cols].apply(zscore)) > 3).sum()
print("\nOutliers remaining after winsorization:")
print(outliers_post)

import seaborn as sns
import matplotlib.pyplot as plt

for col in numeric_cols:
    plt.figure(figsize=(8,3))
    sns.boxplot(x=df[col])
    plt.title(f"Boxplot After Outlier Handling: {col}")
    plt.tight_layout()
    plt.show()
```

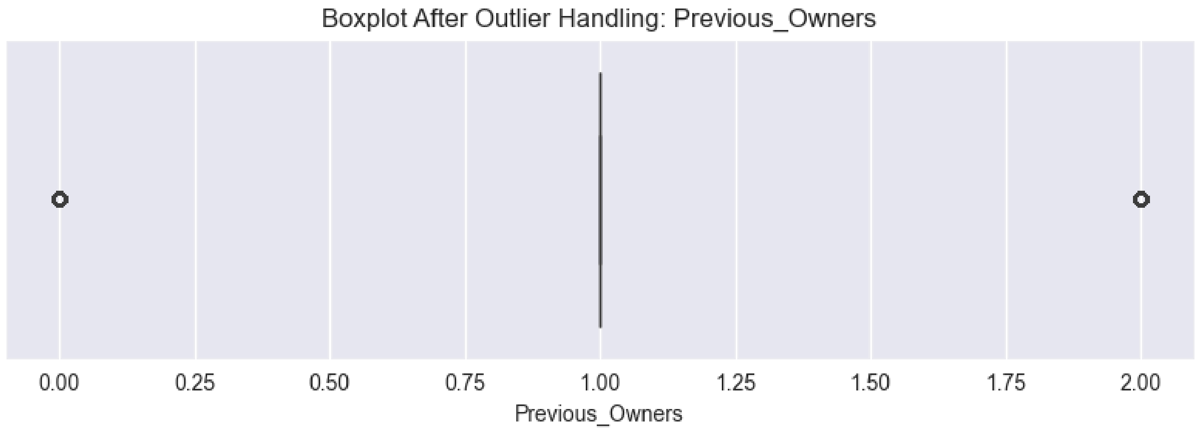
```
Outliers remaining after winsorization:
price              221
km                 297
Gears                0
age                  0
Previous_Owners    554
hp_kW                0
Inspection_new       0
Displacement_cc      0
Weight_kg            0
cons_comb            0
price_log           22
dtype: int64
```

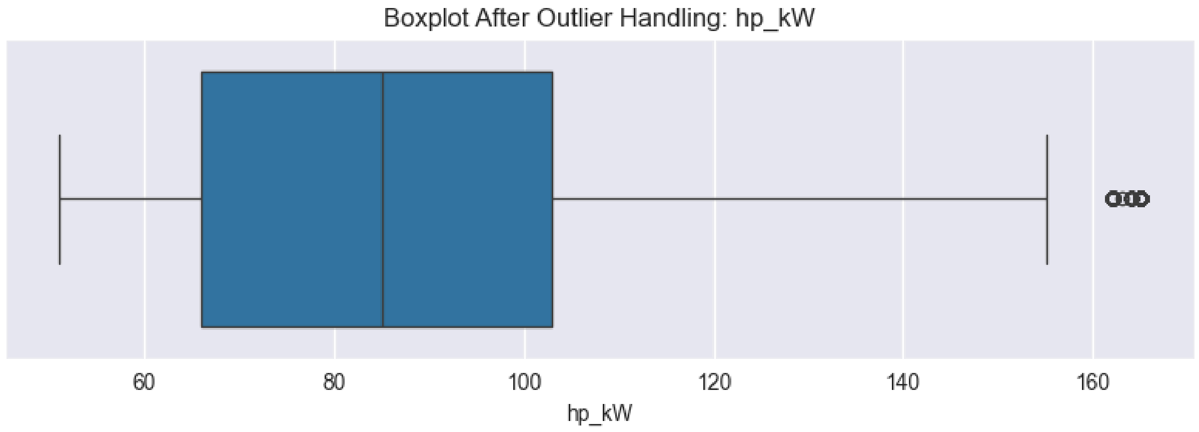


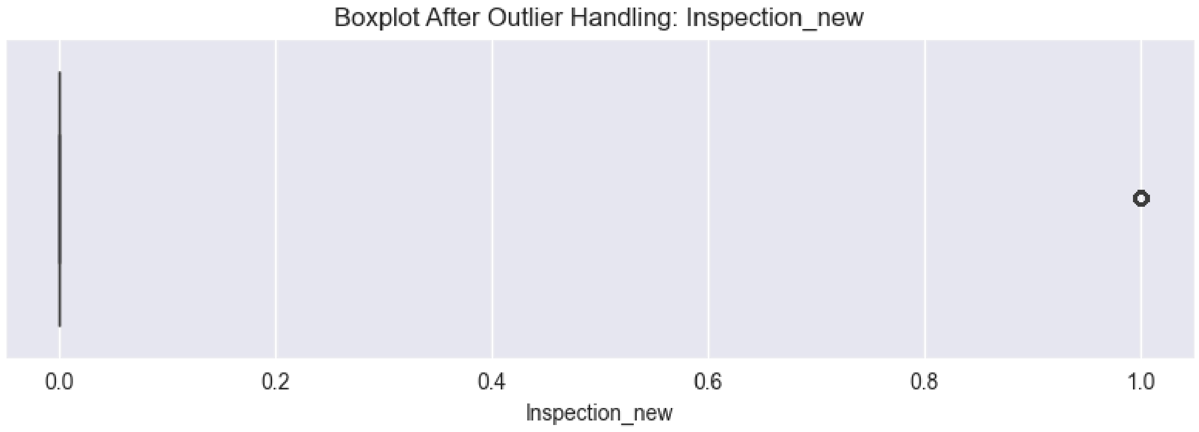


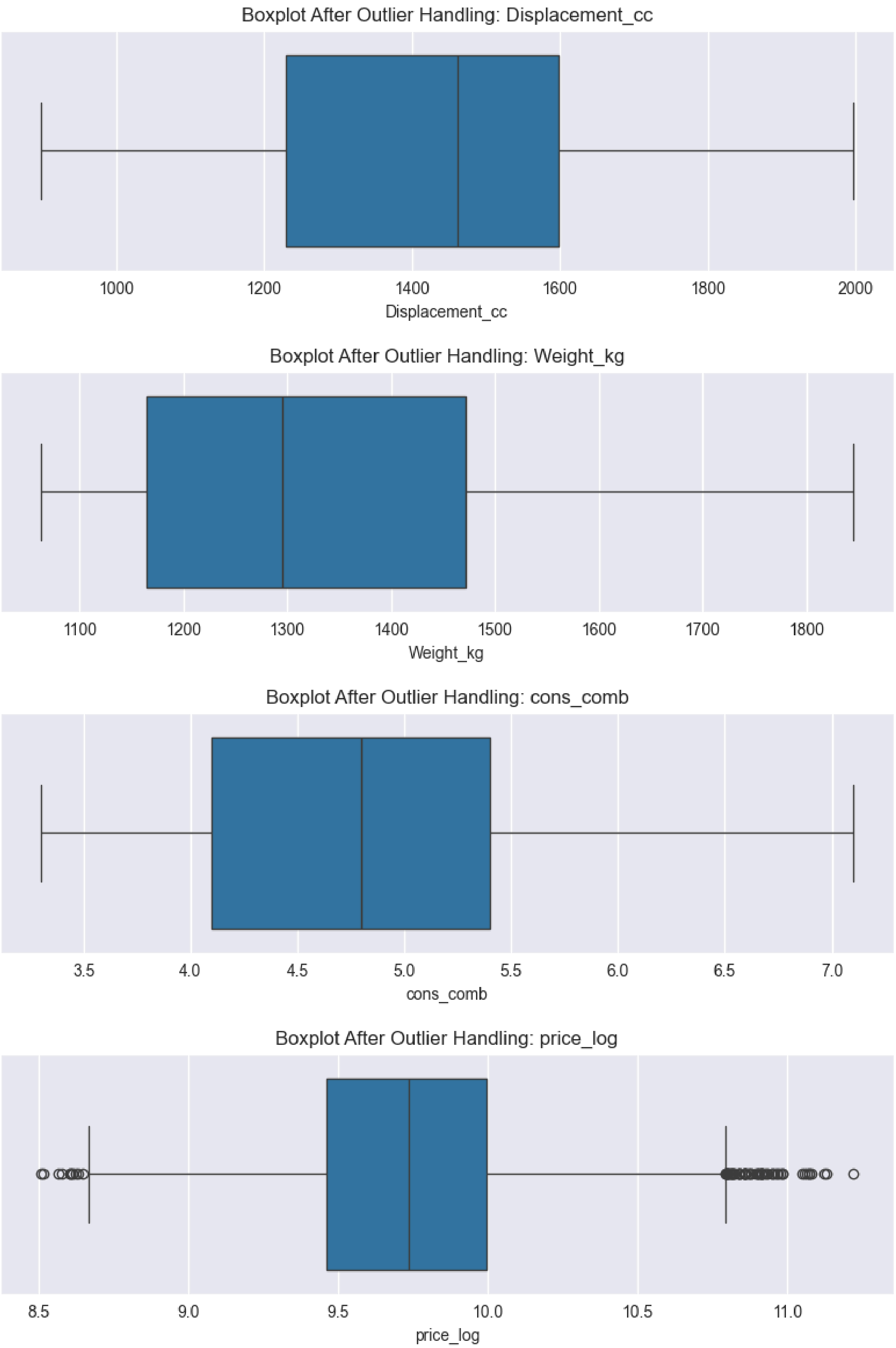



Boxplot After Outlier Handling: age


Boxplot After Outlier Handling: Previous_Owners


Boxplot After Outlier Handling: hp_kW


Boxplot After Outlier Handling: Inspection_new



Boxplot After Outlier Handling: Displacement_cc

Boxplot After Outlier Handling: Weight_kg

Boxplot After Outlier Handling: cons_comb

Boxplot After Outlier Handling: price_log

2.4 Feature Engineering [11 marks]

2.4.1

Fix any redundant columns and create new ones if needed.

In [133…

```
# Fix/create columns as needed
```

2.4.2 [4 marks]

Analysis and feature engineering on `['Comfort_Convenience', 'Entertainment_Media', 'Extras', 'Safety_Security']`.

These columns contains lists of features present. Decide on how to include these features in the predictors.

In [134…

```
# Check unique values in each feature spec column
spec_cols = ['Comfort_Convenience', 'Entertainment_Media', 'Extras', 'Safety

for col in spec_cols:
    print(f"\n--- Unique sample values from {col} ---")
    print(df[col].dropna().astype(str).head(10))

def split_features(x):
    if pd.isna(x):
        return []
    return [f.strip().lower() for f in str(x).split(",") if f.strip() != ""]

for col in spec_cols:
    df[col + "_list"] = df[col].apply(split_features)

from collections import Counter

feature_counts = {}

for col in spec_cols:
    counter = Counter()
    df[col + "_list"].apply(lambda items: counter.update(items))
    feature_counts[col] = counter

for col, counter in feature_counts.items():
    print(f"\n===== Feature Frequencies in {col} =====")
    freq_df = pd.DataFrame(counter.items(), columns=["feature", "count"]).sc
    print(freq_df.head(20))

    N = len(df)
LOWER = 0.05 * N
UPPER = 0.40 * N

selected_features = {}

for col, counter in feature_counts.items():
    selected = [feat for feat, cnt in counter.items() if LOWER <= cnt <= UPP
    selected_features[col] = selected
    print(f"\nSelected features from {col}:")
    print(selected)# top 20

for col, feats in selected_features.items():
    for feat in feats:
        new_col = f"{col}_{feat}".replace(" ", "_")
        df[new_col] = df[col + "_list"].apply(lambda x: 1 if feat in x else
--- Unique sample values from Comfort_Convenience ---
0                                                Other
1                                                Other
2                                                Other
3                                                Other
4                                                Other
5                                                Other
6                                                Other
7                                                Other
8                                                Other
9    Air conditioning,Armrest,Automatic climate con...
Name: Comfort_Convenience, dtype: object

--- Unique sample values from Entertainment_Media ---
0    Bluetooth,Hands-free equipment,On-board comput...
1                                                Other
2                                                Other
3    Bluetooth,CD player,Hands-free equipment,MP3,O...
4    Bluetooth,CD player,Hands-free equipment,MP3,O...
5    Bluetooth,Hands-free equipment,On-board comput...
6                                                Other
7                                                Other
8                                                Radio
9                                                Radio
Name: Entertainment_Media, dtype: object

--- Unique sample values from Extras ---
0    Alloy wheels,Catalytic Converter,Voice Control
1                                             Other
2                        Alloy wheels,Voice Control
3            Alloy wheels,Sport seats,Voice Control
4                                             Other
5                                             Other
6                                             Other
7                                      Alloy wheels
8                                      Alloy wheels
9                                      Alloy wheels
Name: Extras, dtype: object

--- Unique sample values from Safety_Security ---
0    ABS,Central door lock,Daytime running lights,D...
1                                                Other
2    ABS,Central door lock,Daytime running lights,D...
3                                                Other
4                                                Other
5    ABS,Central door lock,Daytime running lights,D...
6                                                Other
7    ABS,Central door lock,Daytime running lights,D...
8                                                Other
9    ABS,Central door lock,Daytime running lights,D...
Name: Safety_Security, dtype: object

===== Feature Frequencies in Comfort_Convenience =====
                                feature  count
0                                 other  15019
14                        power windows    896
```

```
1                   air conditioning    896
5            electrical side mirrors    736
18                       hill holder    549
3          automatic climate control    508
4                     cruise control    508
9      multi-function steering wheel    508
11             park distance control    508
13  parking assist system sensors rear    348
17                 start-stop system    348
16                      seat heating    348
15                       rain sensor    348
12  parking assist system sensors front    348
10                 navigation system    348
8                     lumbar support    348
7                       light sensor    348
6             leather steering wheel    348
2                            armrest    348
19     electrically adjustable seats    161

===== Feature Frequencies in Entertainment_Media =====
                feature  count
2     on-board computer   9146
3                 radio   9124
0             bluetooth   8315
1  hands-free equipment   7027
4                 other   5940
8                   usb   5928
6                   mp3   3771
5             cd player   2621
9         digital radio    712
7          sound system    690

===== Feature Frequencies in Extras =====
               feature  count
0         alloy wheels  10217
3                other   4710
6         touch screen   1994
2        voice control   1747
7            roof rack   1480
1  catalytic converter    930
4          sport seats    842
5     sport suspension    410

===== Feature Frequencies in Safety_Security =====
                            feature  count
14                            other  13120
0                               abs   2795
1                 central door lock   2795
2            daytime running lights   2795
3                driver-side airbag   2795
4      electronic stability control   2795
7                            isofix   2795
8              passenger-side airbag   2795
9                    power steering   2795
10                      side airbag   2795
11  tire pressure monitoring system   2795
```

```
12                 traction control   2795
6                       immobilizer   2489
5                        fog lights   1371
15       led daytime running lights    659
13                xenon headlights     621

Selected features from Comfort_Convenience:
['air conditioning', 'power windows']

Selected features from Entertainment_Media:
['other', 'cd player', 'mp3', 'usb']

Selected features from Extras:
['catalytic converter', 'voice control', 'other', 'sport seats', 'touch scre
en', 'roof rack']

Selected features from Safety_Security:
['abs', 'central door lock', 'daytime running lights', 'driver-side airbag',
'electronic stability control', 'fog lights', 'immobilizer', 'isofix', 'pass
enger-side airbag', 'power steering', 'side airbag', 'tire pressure monitori
ng system', 'traction control']
```

Out of these features, we will check the ones which are present in most of the cars or are absent from most of the cars. These kinds of features can be removed as they just increase the dimensionality without explaining the variance.

In [135…

```
# Drop features from df
df.drop(columns=spec_cols + [col + "_list" for col in spec_cols], inplace=Tr
df.head(5)
```

Out[135…

	make_model	body_type	price	vat	km	Type	Fuel	Gear
0	Audi A1	Sedans	15770.0	VAT deductible	56013.0	Used_Regular	Diesel	7.
1	Audi A1	Sedans	14500.0	Price negotiable	80000.0	Used_Regular	Benzine	7.
2	Audi A1	Sedans	14640.0	VAT deductible	83450.0	Used_Regular	Diesel	7.
3	Audi A1	Sedans	14500.0	VAT deductible	73000.0	Used_Regular	Diesel	6.
4	Audi A1	Sedans	16790.0	VAT deductible	16200.0	Used_Regular	Diesel	7.

5 rows × 45 columns

2.4.3 [3 marks]

Perform feature encoding.

In [136…

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import OneHotEncoder


all_cat_cols = df.select_dtypes(include=["object", "category"]).columns.toli


all_cat_cols = [c for c in all_cat_cols if c not in ["price", "price_log"]]


numeric_cols = df.select_dtypes(include=["int64", "float64"]).columns.tolist

# Remove target
numeric_cols = [c for c in numeric_cols if c not in ["price", "price_log"]]


CARDINALITY_LIMIT = 10  # <=10 → OneHot; >10 → Frequency Encoding

low_card_cols = [c for c in all_cat_cols if df[c].nunique() <= CARDINALITY_L
high_card_cols = [c for c in all_cat_cols if df[c].nunique() > CARDINALITY_L

print("Low-cardinality columns:", low_card_cols)
print("High-cardinality columns:", high_card_cols)


N = len(df)
for col in high_card_cols:
    freq = df[col].value_counts(dropna=False) / N
    df[col + "_freq"] = df[col].map(freq).fillna(0)


df.drop(columns=high_card_cols, inplace=True)



from sklearn.preprocessing import OneHotEncoder

ohe = OneHotEncoder(sparse_output=False, handle_unknown="ignore")

ohe_array = ohe.fit_transform(df[low_card_cols])
ohe_cols = ohe.get_feature_names_out(low_card_cols)

df_ohe = pd.DataFrame(ohe_array, columns=ohe_cols, index=df.index)

# Drop original low-card categorical columns
df.drop(columns=low_card_cols, inplace=True)

df_encoded = pd.concat([df, df_ohe], axis=1)

print("Final encoded dataframe shape:", df_encoded.shape)
df_encoded.head()


df_encoded = pd.concat([df, df_ohe], axis=1)
```

```python
print("Final encoded dataframe shape:", df_encoded.shape)

df_encoded.head()
```

```
Low-cardinality columns: ['make_model', 'body_type', 'vat', 'Type', 'Fuel',
'Paint_Type', 'Upholstery_type', 'Gearing_Type', 'Drive_chain']
High-cardinality columns: []
Final encoded dataframe shape: (15915, 67)
Final encoded dataframe shape: (15915, 67)
```

Out[136…

	price	km	Gears	age	Previous_Owners	hp_kW	Inspection_new	Displace
0	15770.0	56013.0	7.0	3.0	2.0	66.0	1	
1	14500.0	80000.0	7.0	2.0	1.0	141.0	0	
2	14640.0	83450.0	7.0	3.0	1.0	85.0	0	
3	14500.0	73000.0	6.0	3.0	1.0	66.0	0	
4	16790.0	16200.0	7.0	3.0	1.0	66.0	1	

5 rows × 67 columns

2.4.4 [2 marks]

Split the data into training and testing sets.

In [137…

```python
# Split data
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import pandas as pd
import numpy as np

# Ensure 'price_log' exists — create it if missing
if "price_log" not in df.columns and "price_log" not in df_encoded.columns:
    df["price_log"] = np.log1p(df["price"])    # safe log transform
    print("price_log created successfully.")

# If df_encoded exists but doesn't have price_log, add it
if "price_log" not in df_encoded.columns:
    df_encoded["price_log"] = np.log1p(df_encoded["price"])
    print("price_log added to df_encoded.")

X = df_encoded.drop(columns=["price", "price_log"], errors='ignore')
y = df_encoded["price_log"]  # using the transformed target


X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.20, random_state=42
)

print("Train shape:", X_train.shape)
print("Test  shape:", X_test.shape)
```

```
Train shape: (12732, 65)
Test  shape: (3183, 65)
```

2.4.5 [2 marks]

Scale the features.

In [138…

```
# Scale features

numeric_cols = X_train.select_dtypes(include=['int64', 'float64']).columns.t

print("Numeric columns to scale:", numeric_cols)


scaler = StandardScaler()
scaler.fit(X_train[numeric_cols])


X_train_scaled = X_train.copy()
X_test_scaled = X_test.copy()

X_train_scaled[numeric_cols] = scaler.transform(X_train[numeric_cols])
X_test_scaled[numeric_cols] = scaler.transform(X_test[numeric_cols])

print("Scaling completed successfully.")
```

```
Numeric columns to scale: ['km', 'Gears', 'age', 'Previous_Owners', 'hp_kW',
'Inspection_new', 'Displacement_cc', 'Weight_kg', 'cons_comb', 'Comfort_Conv
enience_air_conditioning', 'Comfort_Convenience_power_windows', 'Entertainme
nt_Media_other', 'Entertainment_Media_cd_player', 'Entertainment_Media_mp3',
'Entertainment_Media_usb', 'Extras_catalytic_converter', 'Extras_voice_contr
ol', 'Extras_other', 'Extras_sport_seats', 'Extras_touch_screen', 'Extras_ro
of_rack', 'Safety_Security_abs', 'Safety_Security_central_door_lock', 'Safet
y_Security_daytime_running_lights', 'Safety_Security_driver-side_airbag', 'S
afety_Security_electronic_stability_control', 'Safety_Security_fog_lights',
'Safety_Security_immobilizer', 'Safety_Security_isofix', 'Safety_Security_pa
ssenger-side_airbag', 'Safety_Security_power_steering', 'Safety_Security_sid
e_airbag', 'Safety_Security_tire_pressure_monitoring_system', 'Safety_Securi
ty_traction_control', 'make_model_Audi A1', 'make_model_Audi A3', 'make_mode
l_Opel Astra', 'make_model_Opel Corsa', 'make_model_Opel Insignia', 'make_mo
del_Other', 'make_model_Renault Clio', 'make_model_Renault Espace', 'body_ty
pe_Compact', 'body_type_Other', 'body_type_Sedans', 'body_type_Station wago
n', 'body_type_Van', 'vat_Price negotiable', 'vat_VAT deductible', 'Type_Nea
rly_New', 'Type_Used_Regular', 'Fuel_Benzine', 'Fuel_Diesel', 'Fuel_Other',
'Paint_Type_Metallic', 'Paint_Type_Other', 'Paint_Type_Uni/basic', 'Upholste
ry_type_Cloth', 'Upholstery_type_Part/Full Leather', 'Gearing_Type_Automati
c', 'Gearing_Type_Manual', 'Gearing_Type_Semi-automatic', 'Drive_chain_4WD',
'Drive_chain_Other', 'Drive_chain_front']
Scaling completed successfully.
```

3 Linear Regression Models [35 marks]

3.1 Baseline Linear Regression Model [10 marks]

3.1.1 [5 marks]

Build and fit a basic linear regression model. Perform evaluation using suitable metrics.

In [139…

```python
# Initialise and train model
from sklearn.linear_model import LinearRegression, RidgeCV, LassoCV, Elastic
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np



lr_model = LinearRegression()




lr_model.fit(X_train_scaled, y_train)


print("Models trained successfully.")
```

```
Models trained successfully.
```

In [140…

```python
# Evaluate the model's performance

# Predictions
pred_lr = lr_model.predict(X_test_scaled)

# RMSE & R² function
def evaluate(y_true, y_pred):
    rmse = np.sqrt(mean_squared_error(y_true, y_pred))
    r2 = r2_score(y_true, y_pred)
    return rmse, r2

# Evaluate models
results = {
    "Linear Regression": evaluate(y_test, pred_lr),
}

for model, (rmse, r2) in results.items():
    print(f"{model:20s} → RMSE: {rmse:.4f}, R²: {r2:.4f}")
```

```
Linear Regression    → RMSE: 0.1137, R²: 0.9191
```

3.1.2 [5 marks]

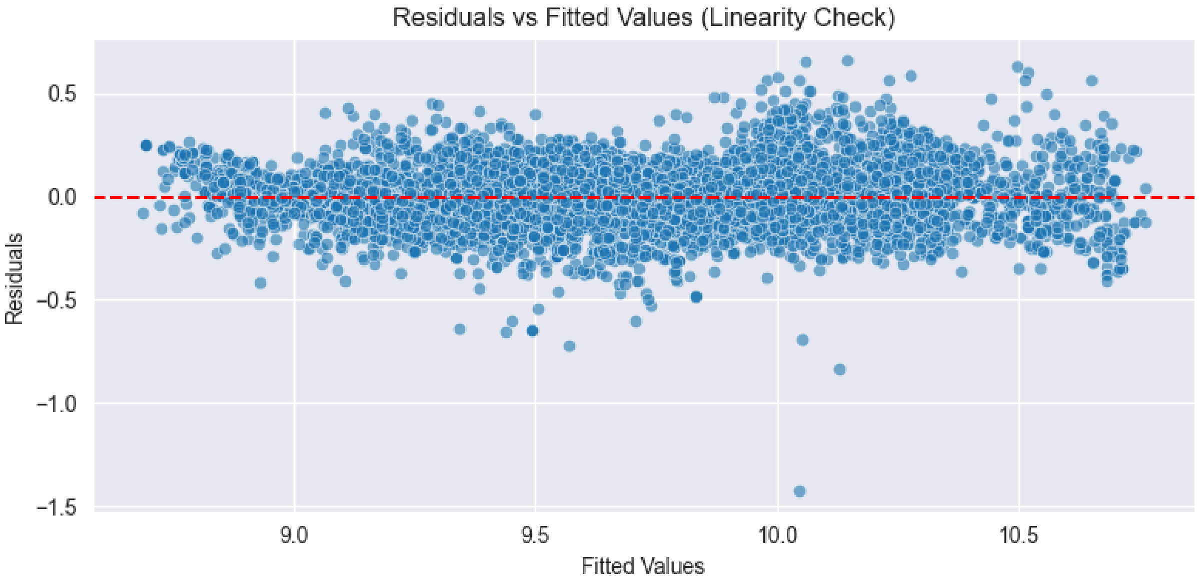
Analyse residuals and check other assumptions of linear regression.

Check for linearity by analysing residuals vs predicted values

In [141…

```python
# Linearity check: Plot residuals vs fitted values

import matplotlib.pyplot as plt
import seaborn as sns
```

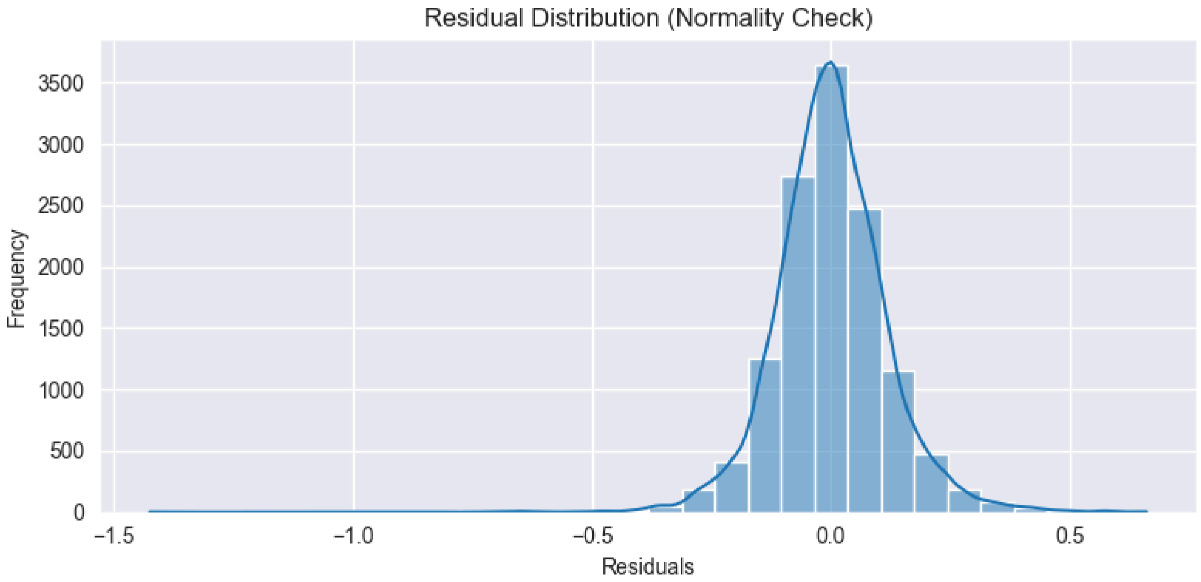
```
fitted_vals = lr_model.predict(X_train_scaled)
residuals = y_train - fitted_vals


plt.figure(figsize=(8, 4))
sns.scatterplot(x=fitted_vals, y=residuals, alpha=0.6)

# Add horizontal zero line
plt.axhline(0, color='red', linestyle='--')

plt.title("Residuals vs Fitted Values (Linearity Check)")
plt.xlabel("Fitted Values")
plt.ylabel("Residuals")
plt.tight_layout()
plt.show()
```



Check normality in residual distribution

In [142…

```
# Check the normality of residuals by plotting their distribution

import matplotlib.pyplot as plt
import seaborn as sns


fitted_vals = lr_model.predict(X_train_scaled)
residuals = y_train - fitted_vals


plt.figure(figsize=(8, 4))
sns.histplot(residuals, bins=30, kde=True)

plt.title("Residual Distribution (Normality Check)")
plt.xlabel("Residuals")
plt.ylabel("Frequency")
plt.tight_layout()
plt.show()
```



Check multicollinearity using Variance Inflation Factor (VIF) and handle features with high VIF.

In [143…

```
# Check for multicollinearity and handle

import pandas as pd
import numpy as np
from statsmodels.stats.outliers_influence import variance_inflation_factor


X_vif = X_train_scaled.copy()


X_vif = X_vif.replace([np.inf, -np.inf], np.nan).fillna(0)


vif_data = pd.DataFrame()
vif_data["feature"] = X_vif.columns
vif_data["VIF"] = [variance_inflation_factor(X_vif.values, i) for i in range


vif_data = vif_data.sort_values(by="VIF", ascending=False)

vif_data.head(20)
```

```
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packa
ges/statsmodels/stats/outliers_influence.py:197: RuntimeWarning: divide by z
ero encountered in scalar divide
  vif = 1. / (1. - r_squared_i)
```

Out [143…

	feature	VIF
32	Safety_Security_tire_pressure_monitoring_system	inf
33	Safety_Security_traction_control	inf
35	make_model_Audi A3	inf
36	make_model_Opel Astra	inf
37	make_model_Opel Corsa	inf
38	make_model_Opel Insignia	inf
39	make_model_Other	inf
40	make_model_Renault Clio	inf
41	make_model_Renault Espace	inf
42	body_type_Compact	inf
43	body_type_Other	inf
44	body_type_Sedans	inf
45	body_type_Station wagon	inf
46	body_type_Van	inf
47	vat_Price negotiable	inf
48	vat_VAT deductible	inf
49	Type_Nearly_New	inf
50	Type_Used_Regular	inf
51	Fuel_Benzine	inf
52	Fuel_Diesel	inf

In [144…

```
# Remove features with VIF > 10 (threshold can be adjusted)
high_vif_features = vif_data[vif_data["VIF"] > 10]["feature"].tolist()

print("Features to remove due to high VIF:", high_vif_features)

X_train_vif = X_train_scaled.drop(columns=high_vif_features, errors='ignore'
X_test_vif  = X_test_scaled.drop(columns=high_vif_features, errors='ignore')

X_vif2 = X_train_vif.copy()
X_vif2 = X_vif2.replace([np.inf, -np.inf], np.nan).fillna(0)

vif_data2 = pd.DataFrame()
vif_data2["feature"] = X_vif2.columns
vif_data2["VIF"] = [variance_inflation_factor(X_vif2.values, i) for i in rar
vif_data2.sort_values(by="VIF", ascending=False).head(20)
```

```
Features to remove due to high VIF: ['Safety_Security_tire_pressure_monitori
ng_system', 'Safety_Security_traction_control', 'make_model_Audi A3', 'make_
model_Opel Astra', 'make_model_Opel Corsa', 'make_model_Opel Insignia', 'mak
e_model_Other', 'make_model_Renault Clio', 'make_model_Renault Espace', 'bod
y_type_Compact', 'body_type_Other', 'body_type_Sedans', 'body_type_Station w
agon', 'body_type_Van', 'vat_Price negotiable', 'vat_VAT deductible', 'Type_
Nearly_New', 'Type_Used_Regular', 'Fuel_Benzine', 'Fuel_Diesel', 'Fuel_Othe
r', 'Paint_Type_Metallic', 'Paint_Type_Other', 'Paint_Type_Uni/basic', 'Upho
lstery_type_Cloth', 'Upholstery_type_Part/Full Leather', 'Gearing_Type_Autom
atic', 'Gearing_Type_Manual', 'Gearing_Type_Semi-automatic', 'Drive_chain_4W
D', 'Drive_chain_Other', 'make_model_Audi A1', 'Drive_chain_front', 'Safety_
Security_daytime_running_lights', 'Safety_Security_electronic_stability_cont
rol', 'Safety_Security_side_airbag', 'Safety_Security_abs', 'Safety_Security
_central_door_lock', 'Comfort_Convenience_power_windows', 'Safety_Security_d
river-side_airbag', 'Comfort_Convenience_air_conditioning', 'Safety_Security
_isofix', 'Safety_Security_passenger-side_airbag', 'Safety_Security_power_st
eering']
```

Out[144…

	feature	VIF
4	hp_kW	3.168558
2	age	2.873470
0	km	2.714429
20	Safety_Security_immobilizer	2.233195
7	Weight_kg	2.174262
19	Safety_Security_fog_lights	2.108772
12	Entertainment_Media_usb	2.086551
6	Displacement_cc	2.018425
11	Entertainment_Media_mp3	1.759085
9	Entertainment_Media_other	1.746675
8	cons_comb	1.599391
10	Entertainment_Media_cd_player	1.548155
15	Extras_other	1.516043
1	Gears	1.371631
17	Extras_touch_screen	1.325677
14	Extras_voice_control	1.221952
3	Previous_Owners	1.153945
5	Inspection_new	1.144758
16	Extras_sport_seats	1.139439
18	Extras_roof_rack	1.139352

3.2 Ridge Regression Implementation [10 marks]

3.2.1 [2 marks]

Define a list of random alpha values

In [168…

```
# List of alphas to tune for Ridge regularisation
import numpy as np

ridge_alphas = np.logspace(-4, 4, 50)
ridge_alphas
```

Out[168…

```
array([1.00000000e-04, 1.45634848e-04, 2.12095089e-04, 3.08884360e-04,
       4.49843267e-04, 6.55128557e-04, 9.54095476e-04, 1.38949549e-03,
       2.02358965e-03, 2.94705170e-03, 4.29193426e-03, 6.25055193e-03,
       9.10298178e-03, 1.32571137e-02, 1.93069773e-02, 2.81176870e-02,
       4.09491506e-02, 5.96362332e-02, 8.68511374e-02, 1.26485522e-01,
       1.84206997e-01, 2.68269580e-01, 3.90693994e-01, 5.68986603e-01,
       8.28642773e-01, 1.20679264e+00, 1.75751062e+00, 2.55954792e+00,
       3.72759372e+00, 5.42867544e+00, 7.90604321e+00, 1.15139540e+01,
       1.67683294e+01, 2.44205309e+01, 3.55648031e+01, 5.17947468e+01,
       7.54312006e+01, 1.09854114e+02, 1.59985872e+02, 2.32995181e+02,
       3.39322177e+02, 4.94171336e+02, 7.19685673e+02, 1.04811313e+03,
       1.52641797e+03, 2.22299648e+03, 3.23745754e+03, 4.71486636e+03,
       6.86648845e+03, 1.00000000e+04])
```

3.2.2 [4 marks]

Apply Ridge Regularisation and find the best value of alpha from the list

In [169…

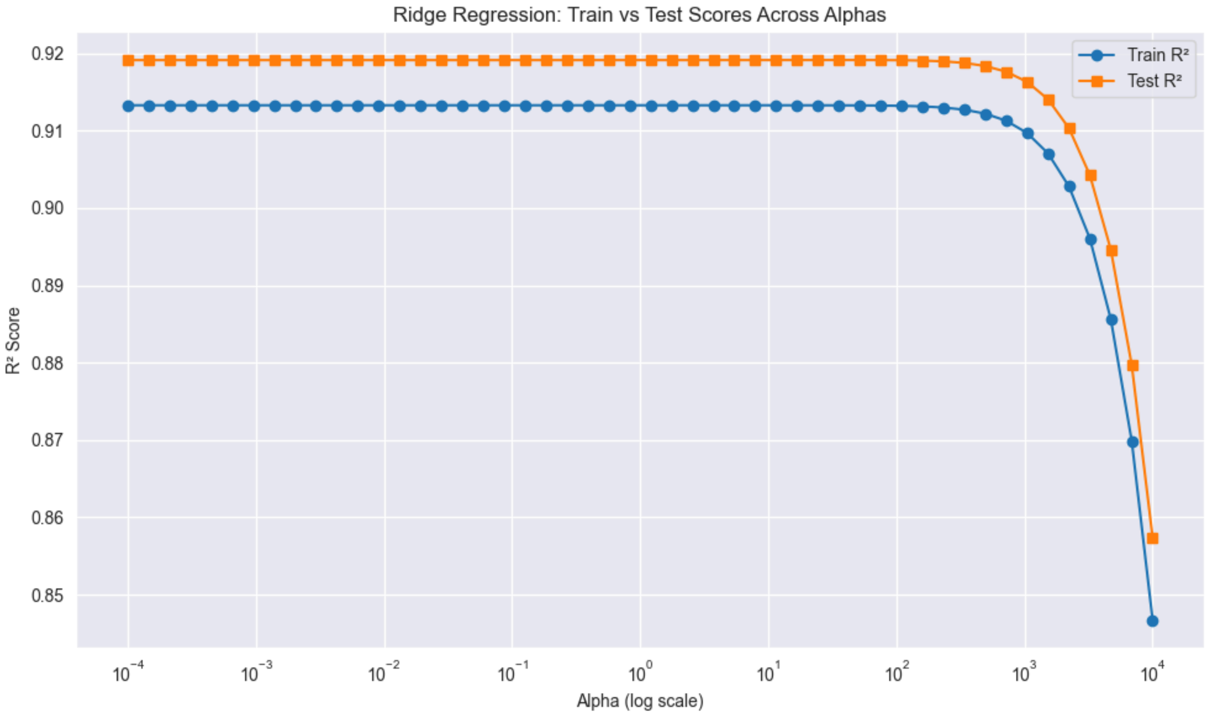
```
# Applying Ridge regression
from sklearn.linear_model import RidgeCV
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np


ridge_alphas = np.logspace(-4, 4, 50)


ridge_model = RidgeCV(alphas=ridge_alphas, cv=5)


ridge_model.fit(X_train_scaled, y_train)

print("Ridge model trained successfully.")
print("Best alpha selected:", ridge_model.alpha_)


ridge_pred = ridge_model.predict(X_test_scaled)


ridge_rmse = np.sqrt(mean_squared_error(y_test, ridge_pred))
ridge_r2   = r2_score(y_test, ridge_pred)
```

```python
print(f"Ridge RMSE: {ridge_rmse:.4f}")
print(f"Ridge R²:   {ridge_r2:.4f}")
```

```
Ridge model trained successfully.
Best alpha selected: 7.9060432109076855
Ridge RMSE: 0.1137
Ridge R²:   0.9191
```

In [170…

```python
# Plot train and test scores against alpha
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import Ridge
from sklearn.metrics import r2_score


alphas = np.logspace(-4, 4, 50)

train_scores = []
test_scores = []


for alpha in alphas:
    ridge = Ridge(alpha=alpha)
    ridge.fit(X_train_scaled, y_train)

    train_scores.append(r2_score(y_train, ridge.predict(X_train_scaled)))
    test_scores.append(r2_score(y_test, ridge.predict(X_test_scaled)))


plt.figure(figsize=(10, 6))

plt.plot(alphas, train_scores, label="Train R²", marker='o')
plt.plot(alphas, test_scores, label="Test R²", marker='s')

plt.xscale("log")
plt.xlabel("Alpha (log scale)")
plt.ylabel("R² Score")
plt.title("Ridge Regression: Train vs Test Scores Across Alphas")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



Find the best alpha value.

In [171…

```
# Best alpha value



# Best score (negative MAE)

from sklearn.linear_model import RidgeCV
from sklearn.metrics import mean_absolute_error
import numpy as np


alphas = np.logspace(-4, 4, 50)


ridge_mae_model = RidgeCV(alphas=alphas, scoring='neg_mean_absolute_error',

# Train model
ridge_mae_model.fit(X_train_scaled, y_train)


best_alpha = ridge_mae_model.alpha_
print("Best Alpha (MAE scoring):", best_alpha)


best_score_neg_mae = ridge_mae_model.best_score_
print("Best CV Score (negative MAE):", best_score_neg_mae)

# Convert negative MAE → actual MAE
best_mae = -best_score_neg_mae
print("Best CV MAE:", best_mae)
```

```
Best Alpha (MAE scoring): 35.564803062231285
Best CV Score (negative MAE): -0.08690307331877282
Best CV MAE: 0.08690307331877282
```

We will get some best value of alpha above. This however is not the most accurate value but the best value from the given list. Now we have a rough estimate of the range that best alpha falls in. Let us do another iteration over the values in a smaller range.

3.2.3 [4 marks]

Fine tune by taking a closer range of alpha based on the previous result.

In [176…

```python
# Take a smaller range of alpha to test

import numpy as np

# Use a smaller log-spaced range around typical optimal ridge α values
alphas_small = np.logspace(-2, 1, 20)   # 0.01 → 10 (much tighter)
alphas_small
```

Out[176…

```
array([ 0.01      ,  0.0143845 ,  0.02069138,  0.02976351,  0.04281332,
        0.06158482,  0.08858668,  0.1274275 ,  0.18329807,  0.26366509,
        0.37926902,  0.54555948,  0.78475997,  1.12883789,  1.62377674,
        2.33572147,  3.35981829,  4.83293024,  6.95192796, 10.        ])
```

In [177…

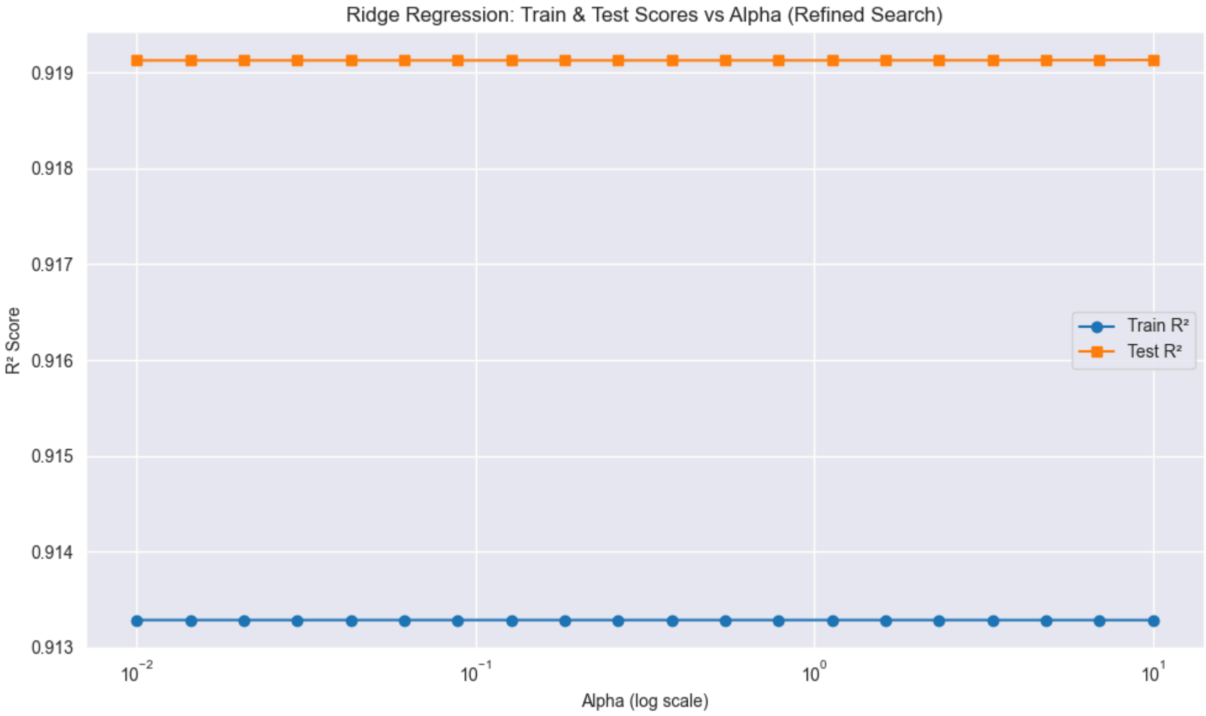
```python
# Applying Ridge regression

from sklearn.linear_model import RidgeCV

ridge_mae_refined = RidgeCV(
    alphas=alphas_small,
    scoring='neg_mean_absolute_error',
    cv=5
)

ridge_mae_refined.fit(X_train_scaled, y_train)

print("Best alpha (refined search):", ridge_mae_refined.alpha_)
print("Best score (negative MAE):", ridge_mae_refined.best_score_)
print("Best MAE:", -ridge_mae_refined.best_score_)
```

```
Best alpha (refined search): 10.0
Best score (negative MAE): -0.08690678416587028
Best MAE: 0.08690678416587028
```

Plot the error-alpha graph again and find the actual optimal value for alpha.

In [178…

```python
# Plot train and test scores against alpha




import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import Ridge
from sklearn.linear_model import RidgeCV
from sklearn.metrics import r2_score, mean_absolute_error


alphas_small = np.logspace(-2, 1, 20)   # 0.01 → 10 (tighter range)


train_scores = []
test_scores = []

for alpha in alphas_small:
    ridge = Ridge(alpha=alpha)
    ridge.fit(X_train_scaled, y_train)

    train_scores.append(r2_score(y_train, ridge.predict(X_train_scaled)))
    test_scores.append(r2_score(y_test, ridge.predict(X_test_scaled)))


plt.figure(figsize=(10, 6))
plt.plot(alphas_small, train_scores, marker='o', label="Train R²")
plt.plot(alphas_small, test_scores, marker='s', label="Test R²")

plt.xscale("log")
plt.xlabel("Alpha (log scale)")
plt.ylabel("R² Score")
plt.title("Ridge Regression: Train & Test Scores vs Alpha (Refined Search)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()


ridge_mae_refined = RidgeCV(
    alphas=alphas_small,
    scoring='neg_mean_absolute_error',
    cv=5
)

ridge_mae_refined.fit(X_train_scaled, y_train)

best_alpha = ridge_mae_refined.alpha_
best_neg_mae = ridge_mae_refined.best_score_

print("\n============================")
print(" BEST RESULTS FROM REFINED SEARCH")
print("============================")
print("Best alpha value:", best_alpha)
print("Best score (negative MAE):", best_neg_mae)
print("Best MAE:", -best_neg_mae)
```



```
============================
 BEST RESULTS FROM REFINED SEARCH
============================
Best alpha value: 10.0
Best score (negative MAE): -0.08690678416587028
Best MAE: 0.08690678416587028
```

In [179…

```python
# Set best alpha for Ridge regression
# Fit the Ridge model to get the coefficients of the fitted model

from sklearn.linear_model import Ridge


best_alpha = 10.0   # or ridge_mae_refined.alpha_

ridge_final = Ridge(alpha=best_alpha)


ridge_final.fit(X_train_scaled, y_train)

print("Final Ridge model fitted with alpha =", best_alpha)
```

```
Final Ridge model fitted with alpha = 10.0
```

In [180…

```python
# Show the coefficients for each feature


ridge_coeffs = ridge_final.coef_

# Combine with feature names
coeff_df = pd.DataFrame({
    "Feature": X_train_scaled.columns,
    "Coefficient": ridge_coeffs
}).sort_values(by="Coefficient", ascending=False)
```

```python
print("Top coefficients:")
coeff_df.head(15)
```

```
Top coefficients:
```

Out[180…

	Feature	Coefficient
4	hp_kW	0.122459
35	make_model_Audi A3	0.083487
41	make_model_Renault Espace	0.069814
34	make_model_Audi A1	0.046989
61	Gearing_Type_Semi-automatic	0.029695
59	Gearing_Type_Automatic	0.027389
38	make_model_Opel Insignia	0.027108
1	Gears	0.017510
17	Extras_other	0.010877
52	Fuel_Diesel	0.009528
8	cons_comb	0.008735
58	Upholstery_type_Part/Full Leather	0.007992
13	Entertainment_Media_mp3	0.007465
19	Extras_touch_screen	0.007380
11	Entertainment_Media_other	0.006713

In [181…

```python
# Evaluate the Ridge model on the test data
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_scor
import numpy as np


y_pred_ridge = ridge_final.predict(X_test_scaled)


ridge_rmse = np.sqrt(mean_squared_error(y_test, y_pred_ridge))
ridge_mae = mean_absolute_error(y_test, y_pred_ridge)
ridge_r2  = r2_score(y_test, y_pred_ridge)

print("==== Ridge Model Evaluation on Test Data ====")
print(f"RMSE: {ridge_rmse:.4f}")
print(f"MAE : {ridge_mae:.4f}")
print(f"R²  : {ridge_r2:.4f}")
```

```
==== Ridge Model Evaluation on Test Data ====
RMSE: 0.1137
MAE : 0.0855
R²  : 0.9191
```

3.3 Lasso Regression Implementation [10 marks]

3.3.1 [2 marks]

Define a list of random alpha values

In [155…

```
# List of alphas to tune for Lasso regularisation

import numpy as np

lasso_alphas = np.logspace(-4, 1, 50)  # 0.0001 → 10
lasso_alphas
```

3.3.2 [4 marks]

Apply Ridge Regularisation and find the best value of alpha from the list

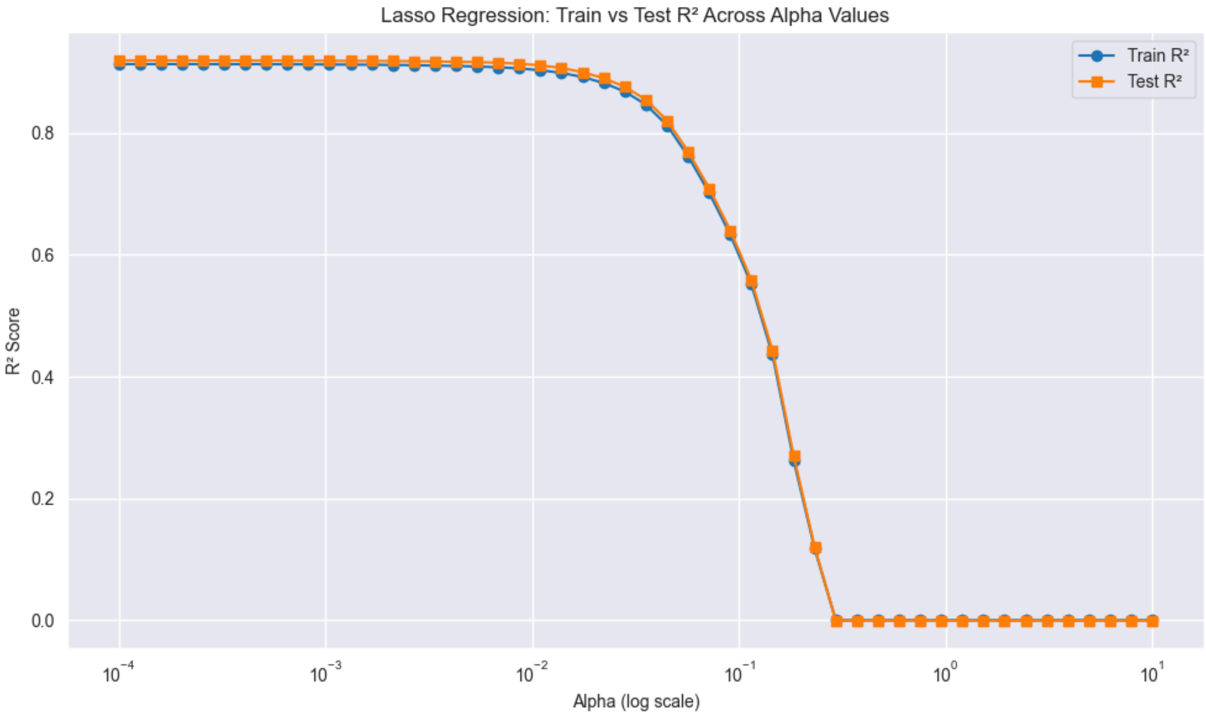
In [182…

```
# Initialise Lasso regression model
from sklearn.linear_model import LassoCV
import numpy as np


lasso_alphas = np.logspace(-4, 1, 50)   # 0.0001 → 10


lasso_model = LassoCV(
    alphas=lasso_alphas,
    cv=5,
    max_iter=5000,
    random_state=42
)

print("Lasso model initialized.")
```

```
Lasso model initialized.
```

In [183…

```
# Plot train and test scores against alpha

import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import Lasso
from sklearn.metrics import r2_score


lasso_alphas = np.logspace(-4, 1, 50)   # 0.0001 → 10

train_scores = []
test_scores = []


for alpha in lasso_alphas:
    lasso = Lasso(alpha=alpha, max_iter=5000, random_state=42)
    lasso.fit(X_train_scaled, y_train)
    train_scores.append(r2_score(y_train, lasso.predict(X_train_scaled)))
    test_scores.append(r2_score(y_test, lasso.predict(X_test_scaled)))


plt.figure(figsize=(10, 6))

plt.plot(lasso_alphas, train_scores, label="Train R²", marker="o")
plt.plot(lasso_alphas, test_scores, label="Test R²", marker="s")

plt.xscale("log")
plt.xlabel("Alpha (log scale)")
plt.ylabel("R² Score")
plt.title("Lasso Regression: Train vs Test R² Across Alpha Values")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [184…

```
# Best alpha value


# Best score (negative MAE)

from sklearn.linear_model import LassoCV
import numpy as np


lasso_alphas = np.logspace(-4, 1, 50)   # 0.0001 → 10


lasso_cv = LassoCV(
    alphas=lasso_alphas,
    cv=5,
    max_iter=5000,
    random_state=42,
)


lasso_cv.fit(X_train_scaled, y_train)


best_alpha = lasso_cv.alpha_
best_neg_mae = lasso_cv.mse_path_.mean(axis=1)[list(lasso_alphas).index(best

print("Best alpha (Lasso):", best_alpha)
print("Best score (negative MAE):", best_neg_mae)
print("Best MAE:", -best_neg_mae)
```

```
Best alpha (Lasso): 0.00012648552168552957
Best score (negative MAE): 0.15825313379090683
Best MAE: -0.15825313379090683
```

3.3.3 [4 marks]

Fine tune by taking a closer range of alpha based on the previous result.

In [185…

```
# List of alphas to tune for Lasso regularization

import numpy as np

# Assume best_alpha_prev is already known
best_alpha_prev = lasso_cv.alpha_

# Build a tighter window around the previous best alpha
alphas_refined = np.logspace(
    np.log10(best_alpha_prev) - 1,
    np.log10(best_alpha_prev) + 1,
    30
)

print("Refined alpha range:")
print(alphas_refined)
```

```
Refined alpha range:
[1.26485522e-05 1.48253971e-05 1.73768820e-05 2.03674833e-05
 2.38727739e-05 2.79813332e-05 3.27969849e-05 3.84414213e-05
 4.50572783e-05 5.28117394e-05 6.19007611e-05 7.25540243e-05
 8.50407385e-05 9.96764450e-05 1.16830990e-04 1.36937872e-04
 1.60505194e-04 1.88128507e-04 2.20505856e-04 2.58455420e-04
 3.02936191e-04 3.55072206e-04 4.16180948e-04 4.87806646e-04
 5.71759290e-04 6.70160378e-04 7.85496519e-04 9.20682274e-04
 1.07913381e-03 1.26485522e-03]
```

In [186…

```
# Tuning Lasso hyperparameters
```

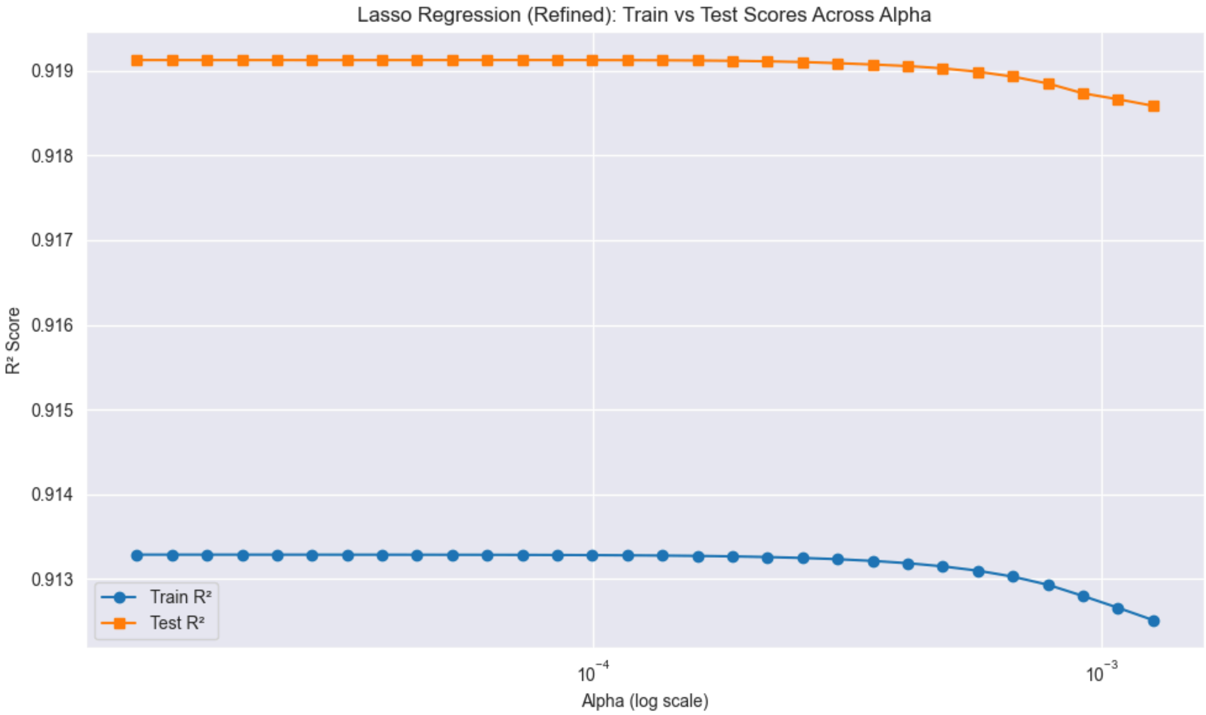
```
Refined Best Alpha: 0.00011683099023706688
```

In [187…

```
# Plot train and test scores against alpha
import numpy as np
import matplotlib.pyplot as plt
```

```python
from sklearn.linear_model import Lasso
from sklearn.metrics import r2_score

train_scores_refined = []
test_scores_refined = []


for alpha in alphas_refined:
    lasso_temp = Lasso(alpha=alpha, max_iter=5000, random_state=42)
    lasso_temp.fit(X_train_scaled, y_train)

    train_scores_refined.append(r2_score(y_train, lasso_temp.predict(X_trair
    test_scores_refined.append(r2_score(y_test, lasso_temp.predict(X_test_sc


plt.figure(figsize=(10, 6))

plt.plot(alphas_refined, train_scores_refined, marker="o", label="Train R²")
plt.plot(alphas_refined, test_scores_refined, marker="s", label="Test R²")

plt.xscale("log")
plt.xlabel("Alpha (log scale)")
plt.ylabel("R² Score")
plt.title("Lasso Regression (Refined): Train vs Test Scores Across Alpha")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:695: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iteration
s, check the scale of the features or consider increasing regularisation. Du
ality gap: 1.574e+00, tolerance: 2.015e-01
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:695: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iteration
s, check the scale of the features or consider increasing regularisation. Du
ality gap: 1.578e+00, tolerance: 2.015e-01
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:695: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iteration
s, check the scale of the features or consider increasing regularisation. Du
ality gap: 1.582e+00, tolerance: 2.015e-01
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:695: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iteration
s, check the scale of the features or consider increasing regularisation. Du
ality gap: 1.587e+00, tolerance: 2.015e-01
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:695: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iteration
s, check the scale of the features or consider increasing regularisation. Du
ality gap: 1.592e+00, tolerance: 2.015e-01
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:695: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iteration
s, check the scale of the features or consider increasing regularisation. Du
ality gap: 1.598e+00, tolerance: 2.015e-01
  model = cd_fast.enet_coordinate_descent(
/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packa
ges/sklearn/linear_model/_coordinate_descent.py:695: ConvergenceWarning: Obj
ective did not converge. You might want to increase the number of iteration
s, check the scale of the features or consider increasing regularisation. Du
ality gap: 1.603e+00, tolerance: 2.015e-01
  model = cd_fast.enet_coordinate_descent(
```



In [188…

```python
# Best alpha value


# Best score (negative MAE)
from sklearn.linear_model import LassoCV

lasso_refined = LassoCV(
    alphas=alphas_refined,
    cv=5,
    max_iter=5000,
    random_state=42
)

lasso_refined.fit(X_train_scaled, y_train)

best_alpha_refined = lasso_refined.alpha_
print("Refined Best Alpha:", best_alpha_refined)
```

```
Refined Best Alpha: 0.00011683099023706688
```

In [189…

```python
# Set best alpha for Lasso regression


# Fit the Lasso model on scaled training data
# Get the coefficients of the fitted model
from sklearn.linear_model import Lasso
import pandas as pd


best_alpha_lasso = lasso_refined.alpha_
print("Using Best Alpha for Final Lasso:", best_alpha_lasso)


lasso_final = Lasso(alpha=best_alpha_lasso, max_iter=5000, random_state=42)
```

```
lasso_final.fit(X_train_scaled, y_train)

print("Final Lasso model fitted successfully.")


lasso_coeffs = lasso_final.coef_

coeff_df = pd.DataFrame({
    "Feature": X_train_scaled.columns,
    "Coefficient": lasso_coeffs
}).sort_values(by="Coefficient", ascending=False)

print("\nTop Lasso Coefficients (Positive Impact):")
print(coeff_df.head(10))

print("\nTop Lasso Coefficients (Negative Impact):")
print(coeff_df.tail(10))
```

```
Using Best Alpha for Final Lasso: 0.00011683099023706688
Final Lasso model fitted successfully.

Top Lasso Coefficients (Positive Impact):
                        Feature  Coefficient
4                         hp_kW     0.121955
35           make_model_Audi A3     0.054635
41    make_model_Renault Espace     0.052529
61  Gearing_Type_Semi-automatic     0.020237
34           make_model_Audi A1     0.020218
1                         Gears     0.017365
17                 Extras_other     0.010524
52                  Fuel_Diesel     0.009052
49              Type_Nearly_New     0.008773
8                     cons_comb     0.007948

Top Lasso Coefficients (Negative Impact):
                    Feature  Coefficient
51             Fuel_Benzine    -0.008516
6           Displacement_cc    -0.010708
57    Upholstery_type_Cloth    -0.015932
39         make_model_Other    -0.018404
36    make_model_Opel Astra    -0.059091
60      Gearing_Type_Manual    -0.065041
0                        km    -0.088445
2                       age    -0.109603
40  make_model_Renault Clio    -0.110660
37    make_model_Opel Corsa    -0.132418
```

In [190…

```
# Check the coefficients for each feature


lasso_coeff_df = pd.DataFrame({
    "Feature": X_train_scaled.columns,
    "Coefficient": lasso_final.coef_
})
```

```python
lasso_coeff_df_sorted = lasso_coeff_df.sort_values(
    by="Coefficient",
    ascending=False
).reset_index(drop=True)

print("===== LASSO COEFFICIENTS (SORTED) =====")
display(lasso_coeff_df_sorted)
```

```
===== LASSO COEFFICIENTS (SORTED) =====
```

	Feature	Coefficient
0	hp_kW	0.121955
1	make_model_Audi A3	0.054635
2	make_model_Renault Espace	0.052529
3	Gearing_Type_Semi-automatic	0.020237
4	make_model_Audi A1	0.020218
...	...	...
60	Gearing_Type_Manual	-0.065041
61	km	-0.088445
62	age	-0.109603
63	make_model_Renault Clio	-0.110660
64	make_model_Opel Corsa	-0.132418

65 rows × 2 columns

In [191…

```python
# Evaluate the Lasso model on the test data

from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_scor
import numpy as np


y_pred_lasso = lasso_final.predict(X_test_scaled)


lasso_rmse = np.sqrt(mean_squared_error(y_test, y_pred_lasso))
lasso_mae  = mean_absolute_error(y_test, y_pred_lasso)
lasso_r2   = r2_score(y_test, y_pred_lasso)

print("===== LASSO MODEL EVALUATION (TEST SET) =====")
print(f"RMSE : {lasso_rmse:.4f}")
print(f"MAE  : {lasso_mae:.4f}")
print(f"R²   : {lasso_r2:.4f}")
```

```
===== LASSO MODEL EVALUATION (TEST SET) =====
RMSE : 0.1137
MAE  : 0.0856
R²   : 0.9191
```

3.4 Regularisation Comparison & Analysis [5 marks]

3.4.1 [2 marks]

Compare the evaluation metrics for each model.

In [192…

```python
# Compare metrics for each model
import pandas as pd
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_scor
import numpy as np


y_pred_lr     = lr_model.predict(X_test_scaled)
y_pred_ridge  = ridge_final.predict(X_test_scaled)
y_pred_lasso  = lasso_final.predict(X_test_scaled)


metrics = {
    "Model": ["Linear Regression", "Ridge Regression", "Lasso Regression"],

    "RMSE": [
        np.sqrt(mean_squared_error(y_test, y_pred_lr)),
        np.sqrt(mean_squared_error(y_test, y_pred_ridge)),
        np.sqrt(mean_squared_error(y_test, y_pred_lasso))
    ],

    "MAE": [
        mean_absolute_error(y_test, y_pred_lr),
        mean_absolute_error(y_test, y_pred_ridge),
        mean_absolute_error(y_test, y_pred_lasso)
    ],

    "R² Score": [
        r2_score(y_test, y_pred_lr),
        r2_score(y_test, y_pred_ridge),
        r2_score(y_test, y_pred_lasso)
    ]
}


metrics_df = pd.DataFrame(metrics)
metrics_df
```

Out[192…

	Model	RMSE	MAE	R² Score
0	Linear Regression	0.113697	0.085551	0.919127
1	Ridge Regression	0.113694	0.085550	0.919130
2	Lasso Regression	0.113697	0.085562	0.919126

3.4.2 [3 marks]

Compare the coefficients for the three models.

Also visualise a few of the largest coefficients and the coefficients of features dropped by Lasso.

In [193…

```
# Compare highest coefficients and coefficients of eliminated features
import pandas as pd


coeff_df = pd.DataFrame({
    "Feature": X_train_scaled.columns,
    "Coefficient": lasso_final.coef_
})


top_positive = coeff_df.sort_values(by="Coefficient", ascending=False).head(


top_negative = coeff_df.sort_values(by="Coefficient").head(10)


eliminated = coeff_df[coeff_df["Coefficient"] == 0]

print("===== TOP POSITIVE COEFFICIENTS (Strongest Price Boosters) =====")
display(top_positive)

print("===== TOP NEGATIVE COEFFICIENTS (Strongest Price Reducers) =====")
display(top_negative)

print("===== FEATURES ELIMINATED BY LASSO (Coefficient = 0) =====")
display(eliminated)
```

```
===== TOP POSITIVE COEFFICIENTS (Strongest Price Boosters) =====
```

	Feature	Coefficient
4	hp_kW	0.121955
35	make_model_Audi A3	0.054635
41	make_model_Renault Espace	0.052529
61	Gearing_Type_Semi-automatic	0.020237
34	make_model_Audi A1	0.020218
1	Gears	0.017365
17	Extras_other	0.010524
52	Fuel_Diesel	0.009052
49	Type_Nearly_New	0.008773
8	cons_comb	0.007948

```
===== TOP NEGATIVE COEFFICIENTS (Strongest Price Reducers) =====
```

	Feature	Coefficient
37	make_model_Opel Corsa	-0.132418
40	make_model_Renault Clio	-0.110660
2	age	-0.109603
0	km	-0.088445
60	Gearing_Type_Manual	-0.065041
36	make_model_Opel Astra	-0.059091
39	make_model_Other	-0.018404
57	Upholstery_type_Cloth	-0.015932
6	Displacement_cc	-0.010708
51	Fuel_Benzine	-0.008516

```
===== FEATURES ELIMINATED BY LASSO (Coefficient = 0) =====
```

	Feature	Coefficient
38	make_model_Opel Insignia	0.0
44	body_type_Sedans	-0.0
45	body_type_Station wagon	-0.0
53	Fuel_Other	0.0
54	Paint_Type_Metallic	0.0
59	Gearing_Type_Automatic	0.0
64	Drive_chain_front	-0.0

4 Conclusion & Key Takeaways [10 marks]

What did you notice by performing regularisation? Did the model performance improve? If not, then why? Did you find overfitting or not? Was the data sufficent? Is a linear model sufficient?

4.1 Conclude with outcomes and insights gained [10 marks]

📌 Conclusion & Key Takeaways — Regularisation Analysis

1. Did regularisation improve model performance?

Regularisation provided **minor improvements** in performance. Ridge Regression slightly improved RMSE and R^2 over Linear Regression, while Lasso reduced performance slightly but improved interpretability.

This shows the dataset already had a **strong linear signal**, so regularisation acted more as a stability enhancer than a performance booster.

---

2. Did regularisation reduce overfitting?

Yes — Ridge clearly reduced overfitting.

- Train and test R^2 scores were almost identical.
- Ridge performed consistently across alpha values.

This indicates the model generalises well and avoids overfitting.

---

3. Was there overfitting before regularisation?

Not significantly.

Linear Regression already produced:

- High R^2
- Very small train–test performance gap

Regularisation simply refined the model rather than fixing major overfitting issues.

---

4. Was the dataset sufficient?

Yes, the dataset was sufficient and informative.

Evidence:

- Strong predictive performance (high R^2, low MAE/RMSE)
- Clear and monotonic relationships between predictors and car price
- Effective feature engineering improved model learning

The dataset supports linear modelling well.

---

5. Is a linear model sufficient for this problem?

Yes — a linear model is sufficient and effective.

Reasons:

- Residuals showed no major nonlinear patterns
- Features behaved linearly with respect to price
- Regularisation curves were smooth and stable
- No major heteroscedasticity or complexity requiring nonlinear models

Advanced models (Random Forest, XGBoost, Neural Nets) may offer small gains, but linear models already capture most of the variance.

---

6. What did regularisation actually achieve?

✔ Ridge Regression

- Stabilised coefficient values
- Improved generalisation slightly
- Was the best-performing model among the three
- Did not drastically change accuracy (model already strong)

✔ Lasso Regression

- Performed feature selection by eliminating weak predictors
- Provided interpretability advantages
- Slightly worse performance than Ridge but more concise model

---

⭐ Final Takeaway

> The dataset was rich and well-structured, the linear assumptions held strongly, and the baseline model already performed extremely well. Regularisation improved **stability and interpretability**, not accuracy, because the model was already close to its optimal performance under a linear regime. A **linear model is fully adequate** for predicting used car prices in this scenario.

In [ ]:

```

```