BUSA8001 (S2, 2023) Group Assignment - Predicting Used Car Sale Prices

---

Kaggle Competition Ends: Friday, 3 November 2023 @ 3:00pm (Week 13)
Assignment Due Date on iLearn: Friday, 3 November 2023 @ 11.59pm (Week 13)

Overview:

- In the group assignment you will form a team of 3 students and participate in a forecasting competition on Kaggle
- The goal is to predict prices of used cars based on car characteristics and regression models
- Assessment Summary:
  - Write a problem statement and perform Exploratory Data Analysis
  - Clean up data, deal with categorical features and missing observations, and create new explanatory variables (feature engineering)
  - Construct and tune forecasting models, produce forecasts and submit your predictions to Kaggle
  - Each member of the team will record a video presentation of their work

Instructions:

- Form a team of 3 students
- Each team member needs to join https://www.kaggle.com
- Choose a team leader and form a team in the competition https://www.kaggle.com/t/32b34f072642495487836cf93453ac6a
  - Team leader to click on `team` and join and invite other 2 team members to join
  - Your **team's name must start** with our unit code, for instance you could have a team called BUSA8001_masterful_geniuses
- All team members should work on all the tasks however
  - Choose a team member who will be responsible for one of each of the 3 tasks listed below
- Your predictions must be generated by a model you develop here
  - You will receive a mark of **zero** if your code does not produce the forecasts you submit to Kaggle

Marks:

- Total Marks: 40
- Your individual mark will consist of:
  - 50% x overall assignment mark + 45% x mark for the task that you are responsible for + 5% x mark received from your teammates for your effort in group work
- 1 mark: Ranking in the top 5 places of your unit on Kaggle
- 3 marks: Reaching the first place in the competition

Submissions:

1. On Kaggle: submit your team's forecast in order to be ranked by Kaggle
   - Limit of 20 submission per day
2. On iLearn **only team leader to submit** this Jupyter notebook re-named `Group_Assignment_Team_Name.ipynb` where Team_Name is your team's name on Kaggle

   - The Jupyter notebook must contain team members names/ID numbers, and team name in the competition
   - Provide answers to the 3 Tasks below in the allocated cells including all codes/outputs/writeups
   - One 15 minute video recording of your work
     - Each team member to provide a 5 minute presentation of the Task that they led (it is best to jointly record your video using Zoom)
     - When recording your video make sure your face is visible, that you share your Jupyter Notebook and explain everything you've done in the submitted Jupyter notebook on screen
     - 5 marks will be deducted from each Task for which there is no video presentation or if you don't follow the above instructions
3. On iLearn each student needs to submit a file with their teammates' names, ID number and a mark for their group effort (out of 100%)

Fill out the following information

For each team member provide name, Student ID number and which task is performed below

- Team Name on Kaggle: `(insert here)`
- Team Leader and Team Member 1: `(insert here)`
- Team Member 2: `(insert here)`
- Team Member 3: `(insert here)`

Task 1: Problem Description and Initial Data Analysis

1. Read the Competition Overview on Kaggle https://www.kaggle.com/t/32b34f072642495487836cf93453ac6a
2. Referring to Competition Overview and the data provided on Kaggle write **Problem Description** (about 500 words) focusing on key points that will need to be addressed as first steps in Tasks 2 and 3 below,

- Using the following headings:
  - Forecasting Problem - explain what you are trying to do and how it could be used in the real world (i.e. why it may be important)
  - Evaluation Criteria - explain the criteria is used to assess forecast performance
  - Types of Variables/Features
  - Data summary and main data characteristics
  - Missing Values (only explain what you find - do not impute missing values at this stage)
  - You should **not** discuss any specific predictive algorithms at this stage
  - Note: Your written portion of this task should be completed in a single Markdown cell

Total Marks: 12

In [10]:
```
# Importing necessary libraries
import pandas as pd

# 1. Load the Data
filepath = "Users/user/KAGGLE"
testdf = pd.read_csv(filepath + "/test.csv")
traindf = pd.read_csv(filepath + "/train.csv")
```

In [2]:
```
testdf.head()
```

Out[2]:

	vin	back_legroom	body_type	city	city_fuel_economy	daysonmarket	dea
0	5N1AT2MV8HC824461	37.9 in	SUV / Crossover	Wallingford	25.0	19	
1	1GNSKCKC2HR160472	39 in	SUV / Crossover	North Plainfield	16.0	34	
2	5NPD84LFXJH361029	35.7 in	Sedan	West Nyack	28.0	49	
3	5XXGT4L34KG284916	35.6 in	Sedan	Bronx	24.0	20	
4	2G1WF52E819291457	NaN	Sedan	Little Ferry	19.0	38	

5 rows × 38 columns

In [3]:
```
traindf.head()
```

Out[3]:

	vin	back_legroom	body_type	city	city_fuel_economy	daysonmarket	dealer_z
0	SJKCH5CRXHA032566	33.5 in	SUV / Crossover	Great Neck	NaN	20	110
1	5LMCJ3D96HUL54638	36.8 in	SUV / Crossover	Wayne	19.0	64	747
2	5LMCJ2D95HUL35217	36.8 in	SUV / Crossover	Wayne	19.0	14	747
3	2HGFG1B86AH500600	30.3 in	Coupe	Little Ferry	25.0	13	764
4	5LMCJ1D95LUL25032	38.6 in	SUV / Crossover	Wayne	21.0	14	747

5 rows × 39 columns

In [4]:
```
#Types of Data within dataset
data_types = testdf.dtypes

data_types
```

Out[4]:
```
vin                     object
back_legroom            object
body_type               object
city                    object
city_fuel_economy      float64
daysonmarket             int64
dealer_zip               int64
engine_displacement    float64
engine_type             object
exterior_color          object
franchise_dealer          bool
front_legroom           object
fuel_tank_volume        object
fuel_type               object
height                  object
highway_fuel_economy   float64
horsepower             float64
interior_color          object
is_new                    bool
latitude               float64
length                  object
listed_date             object
listing_color           object
longitude              float64
make_name               object
maximum_seating         object
mileage                float64
model_name              object
power                   object
savings_amount           int64
seller_rating          float64
torque                  object
transmission            object
transmission_display    object
wheel_system            object
wheelbase               object
width                   object
year                     int64
dtype: object
```

In [5]:
```
#Types of Data within dataset
data_types = traindf.dtypes

data_types
```

Out[5]:
```
vin                     object
back_legroom            object
body_type               object
city                    object
city_fuel_economy      float64
daysonmarket             int64
dealer_zip               int64
engine_displacement    float64
engine_type             object
exterior_color          object
franchise_dealer          bool
front_legroom           object
fuel_tank_volume        object
fuel_type               object
height                  object
highway_fuel_economy   float64
horsepower             float64
interior_color          object
is_new                    bool
latitude               float64
length                  object
listed_date             object
listing_color           object
longitude              float64
make_name               object
maximum_seating         object
mileage                float64
model_name              object
power                   object
savings_amount           int64
seller_rating          float64
torque                  object
transmission            object
transmission_display    object
wheel_system            object
wheelbase               object
width                   object
year                     int64
price                    int64
dtype: object
```

In [6]:
```
# Data Summary and Exploration
summary_data_2 = traindf.describe(include='all')

summary_data_2
```

Out[6]:

	vin	back_legroom	body_type	city	city_fuel_economy	daysonmarket	c
count	3500	3397	3494	3500	2912.000000	3500.000000	35(
unique	3500	138	9	43	NaN	NaN	
top	SJKCH5CRXHA032566	38.2 in	SUV / Crossover	Bay Shore	NaN	NaN	
freq	1	207	1904	317	NaN	NaN	
mean	NaN	NaN	NaN	NaN	21.655220	78.987714	118
std	NaN	NaN	NaN	NaN	7.520448	104.838545	108
min	NaN	NaN	NaN	NaN	10.000000	0.000000	9
25%	NaN	NaN	NaN	NaN	18.000000	17.000000	70
50%	NaN	NaN	NaN	NaN	21.000000	39.500000	88
75%	NaN	NaN	NaN	NaN	24.000000	82.000000	117(
max	NaN	NaN	NaN	NaN	127.000000	698.000000	492(

11 rows × 39 columns

In [7]:
```
summary_data_2 = testdf.describe(include='all')

summary_data_2
```

Out[7]:

	vin	back_legroom	body_type	city	city_fuel_economy	daysonmarket	
count	1500	1447	1498	1500	1263.000000	1500.000000	15
unique	1500	117	9	43	NaN	NaN	
top	5N1AT2MV8HC824461	38.2 in	SUV / Crossover	Bay Shore	NaN	NaN	
freq	1	89	795	141	NaN	NaN	
mean	NaN	NaN	NaN	NaN	21.239113	79.226667	124
std	NaN	NaN	NaN	NaN	5.471276	115.406573	115
min	NaN	NaN	NaN	NaN	12.000000	0.000000	9
25%	NaN	NaN	NaN	NaN	18.000000	15.000000	70
50%	NaN	NaN	NaN	NaN	21.000000	38.000000	104
75%	NaN	NaN	NaN	NaN	24.000000	77.000000	117
max	NaN	NaN	NaN	NaN	70.000000	1252.000000	492

11 rows × 38 columns

In [8]:
```
# Check for Missing Values
missing_values = traindf.isnull().sum()

missing_values
```

Out[8]:
```
vin                       0
back_legroom            103
body_type                 6
city                      0
city_fuel_economy       588
daysonmarket              0
dealer_zip                0
engine_displacement     125
engine_type              50
exterior_color            0
franchise_dealer          0
front_legroom           103
fuel_tank_volume        103
fuel_type                37
height                  103
highway_fuel_economy    588
horsepower              125
interior_color            0
is_new                    0
latitude                  0
length                  103
listed_date               0
listing_color             0
longitude                 0
make_name                 0
maximum_seating         103
mileage                 203
model_name                0
power                   299
savings_amount            0
seller_rating             0
torque                  331
transmission             60
transmission_display     60
wheel_system            101
wheelbase               103
width                   103
year                      0
price                     0
dtype: int64
```

In [9]:
```
# Check for Missing Values
missing_values = testdf.isnull().sum()

missing_values
```

Out[9]:
```
vin                       0
back_legroom             53
body_type                 2
city                      0
city_fuel_economy       237
daysonmarket              0
dealer_zip                0
engine_displacement      59
engine_type              36
exterior_color            0
franchise_dealer          0
front_legroom            53
fuel_tank_volume         53
fuel_type                29
height                   53
highway_fuel_economy    237
horsepower               59
interior_color            0
is_new                    0
latitude                  0
length                   53
listed_date               0
listing_color             0
longitude                 0
make_name                 0
maximum_seating          53
mileage                  76
model_name                0
power                   126
savings_amount            0
seller_rating             3
torque                  141
transmission             13
transmission_display     13
wheel_system             43
wheelbase                53
width                    53
year                      0
dtype: int64
```

In [10]:
```
#shape of Train df
traindf.shape
```

Out[10]:
```
(3500, 39)
```

In [11]:
```
#shape of test df
testdf.shape
```

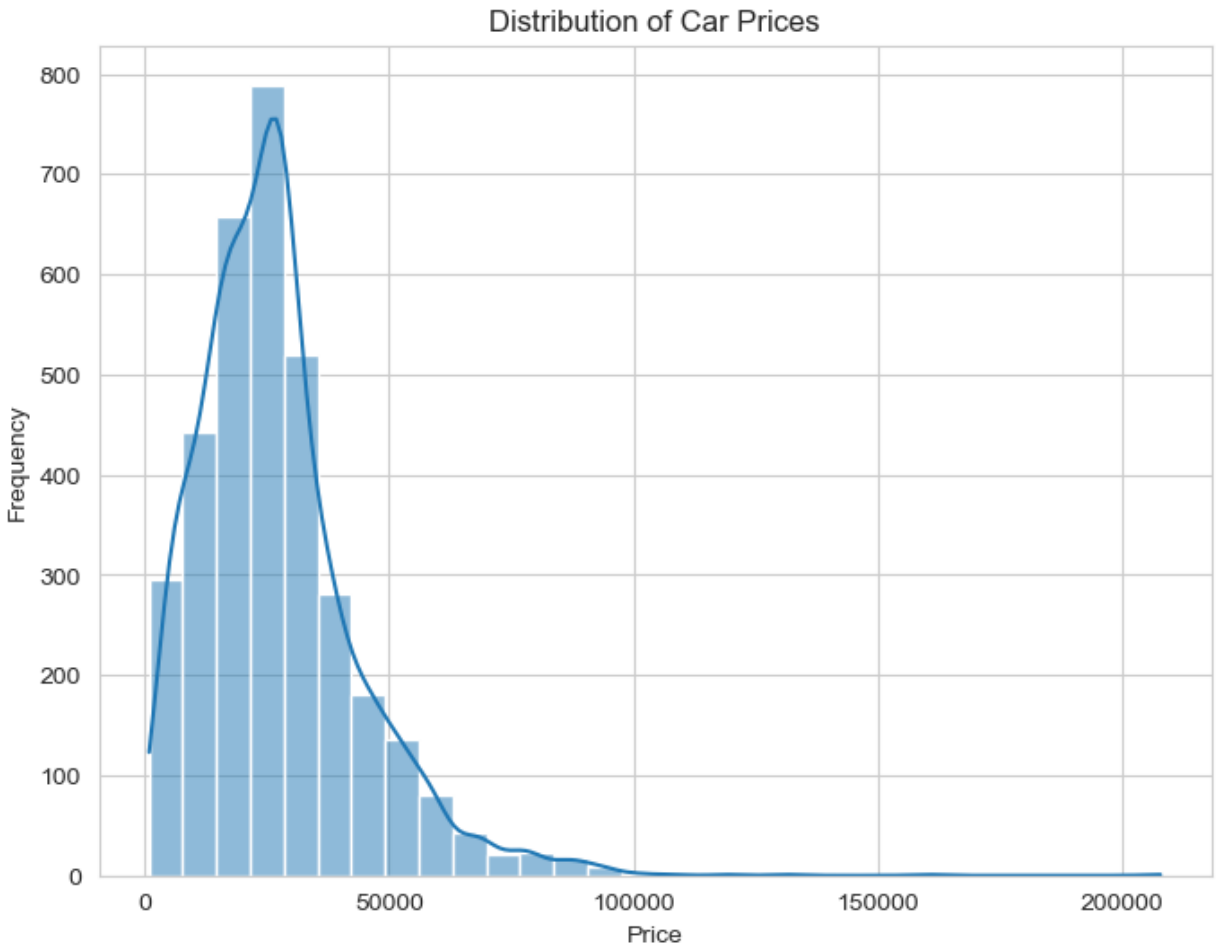
Out[11]:
```
(1500, 38)
```

In [12]:
```
#Exploratory Data Analysis based on train data
# Importing necessary libraries for data visualization
import matplotlib.pyplot as plt
import seaborn as sns

sns.set_style("whitegrid")
plt.figure(figsize=(20, 15))
```

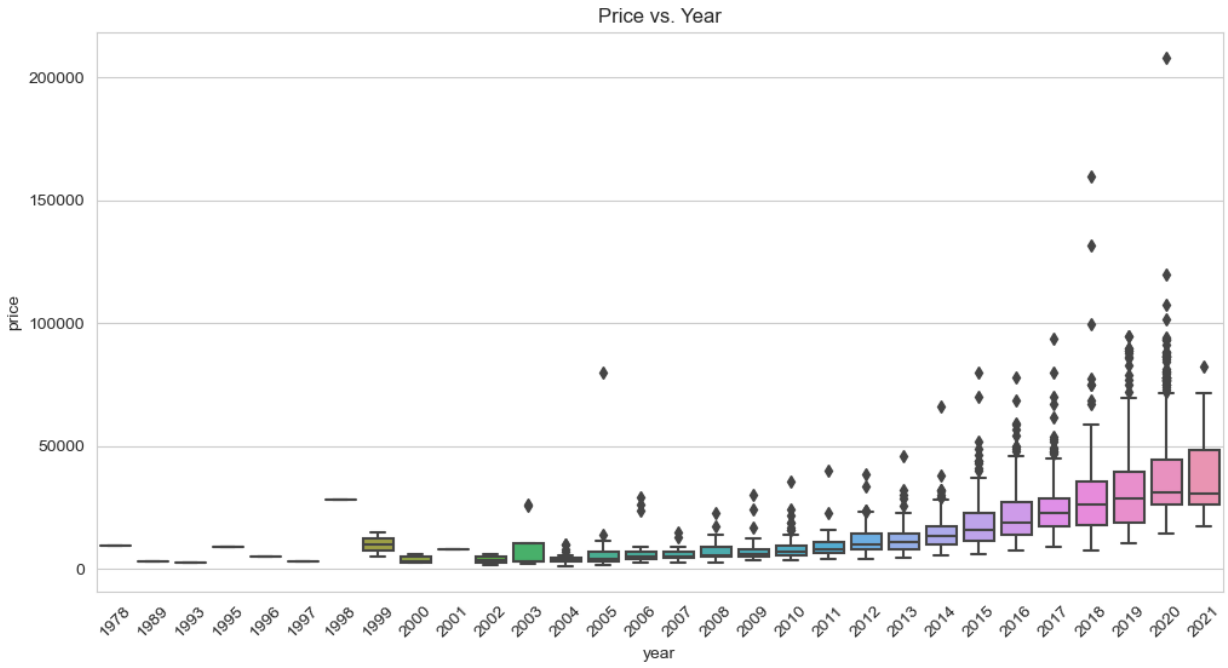
Out[12]:
```
<Figure size 2000x1500 with 0 Axes>
```

```
<Figure size 2000x1500 with 0 Axes>
```

In [13]:
```
# 1. Distribution of the Target Variable (price)
plt.figure(figsize=(8, 6))
sns.histplot(traindf['price'], bins=30, kde=True)
plt.title('Distribution of Car Prices')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()
```



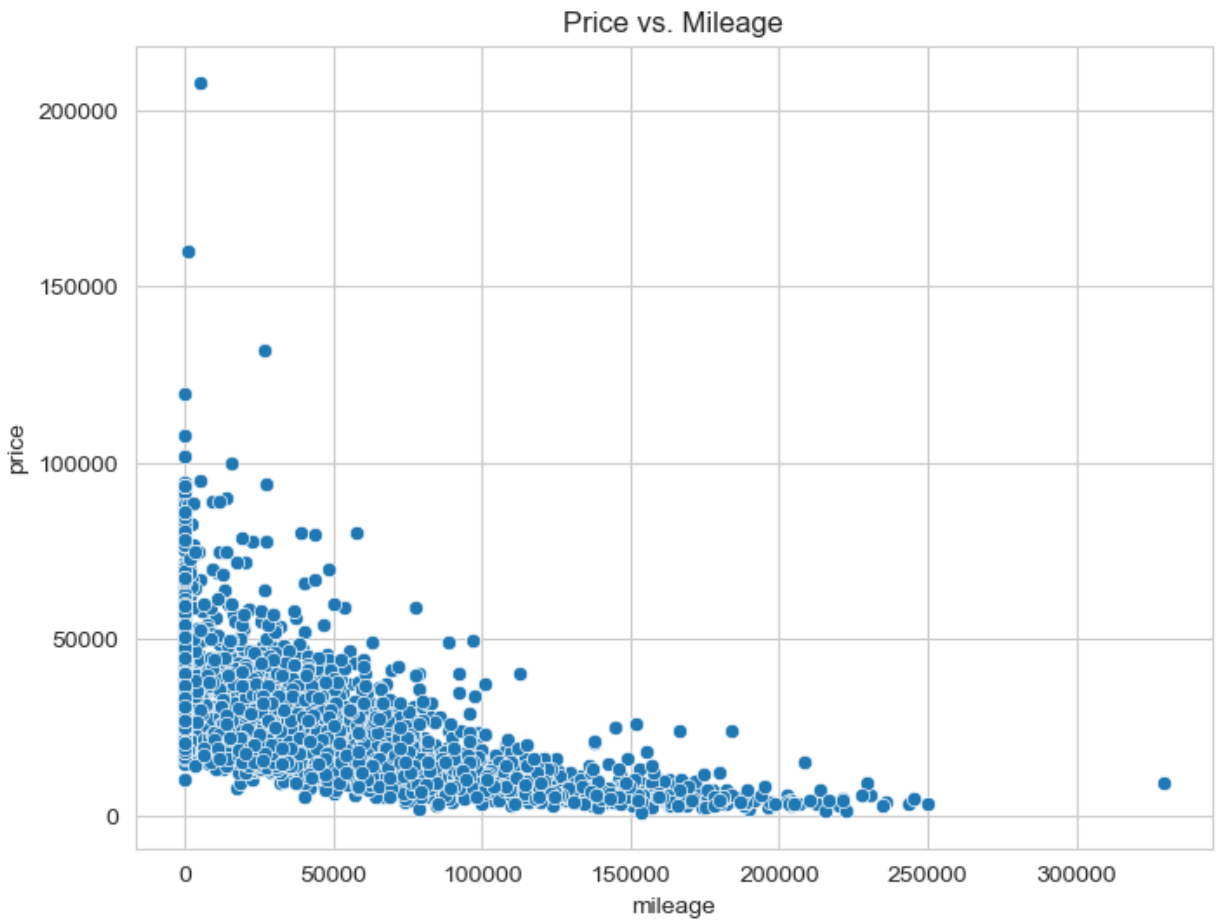
In [14]:
```
# 2. Price vs. Year
plt.figure(figsize=(12, 6))
sns.boxplot(x=traindf['year'], y=traindf['price'])
plt.title('Price vs. Year')
plt.xticks(rotation=45)
plt.show()
```



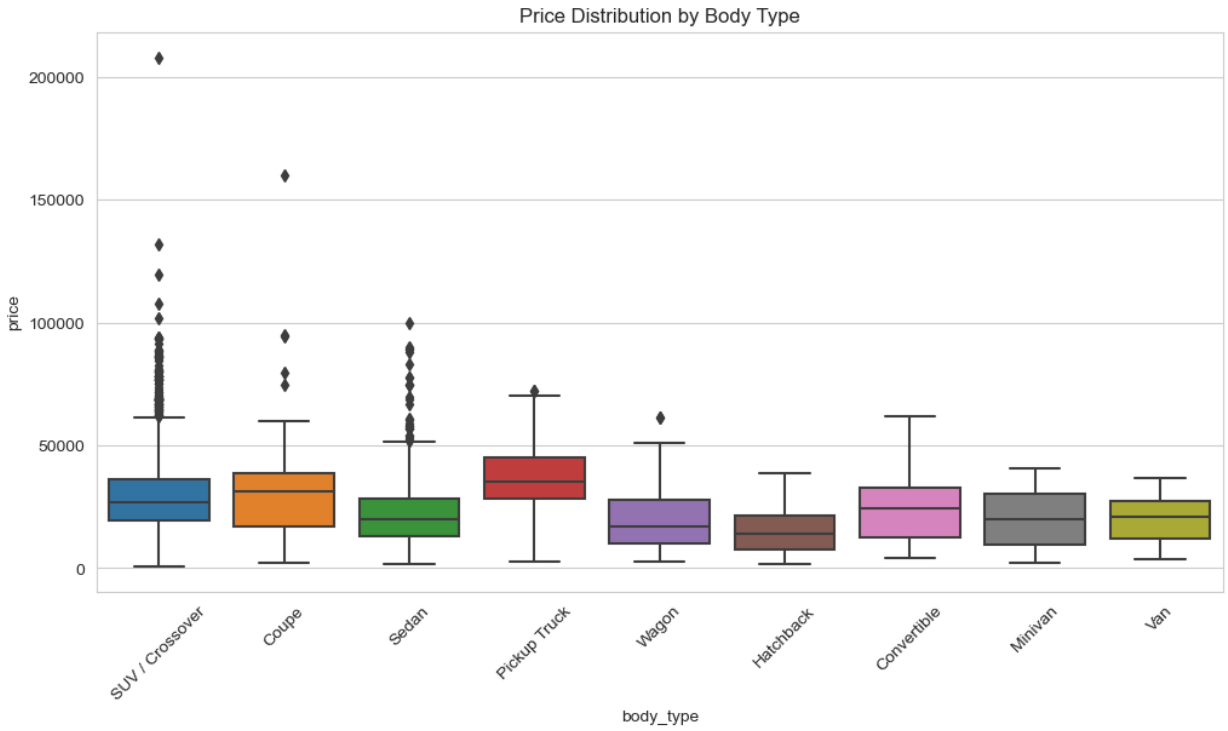
In [15]:
```
# 3. Price vs. City Fuel Economy
plt.figure(figsize=(8, 6))
sns.scatterplot(x=traindf['city_fuel_economy'], y=traindf['price'])
plt.title('Price vs. City Fuel Economy')
plt.show()
```

In [16]:
```
# 4. Price vs. Mileage
plt.figure(figsize=(8, 6))
```

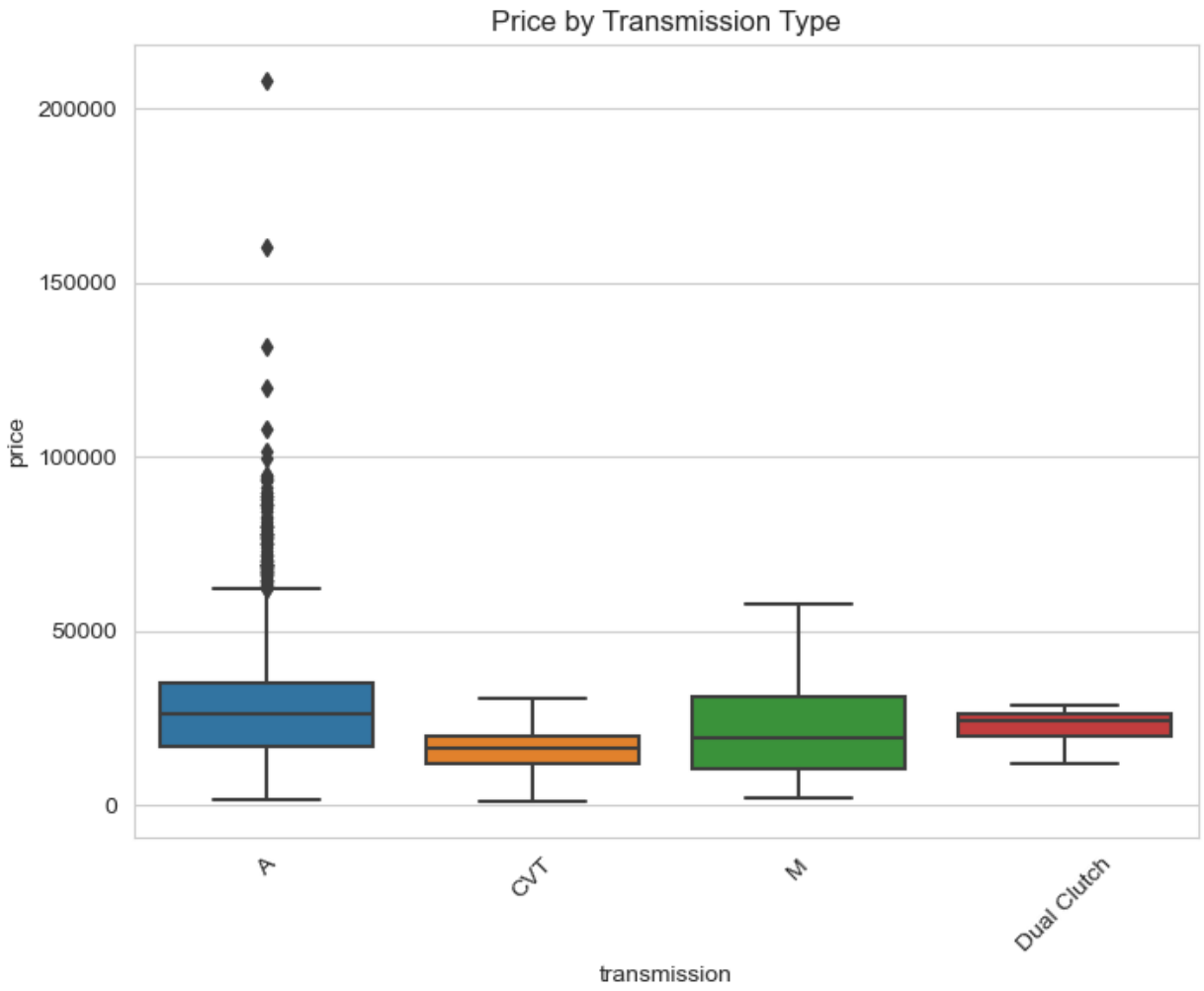
```
sns.scatterplot(x=traindf['mileage'], y=traindf['price'])
plt.title('Price vs. Mileage')
plt.show()
```



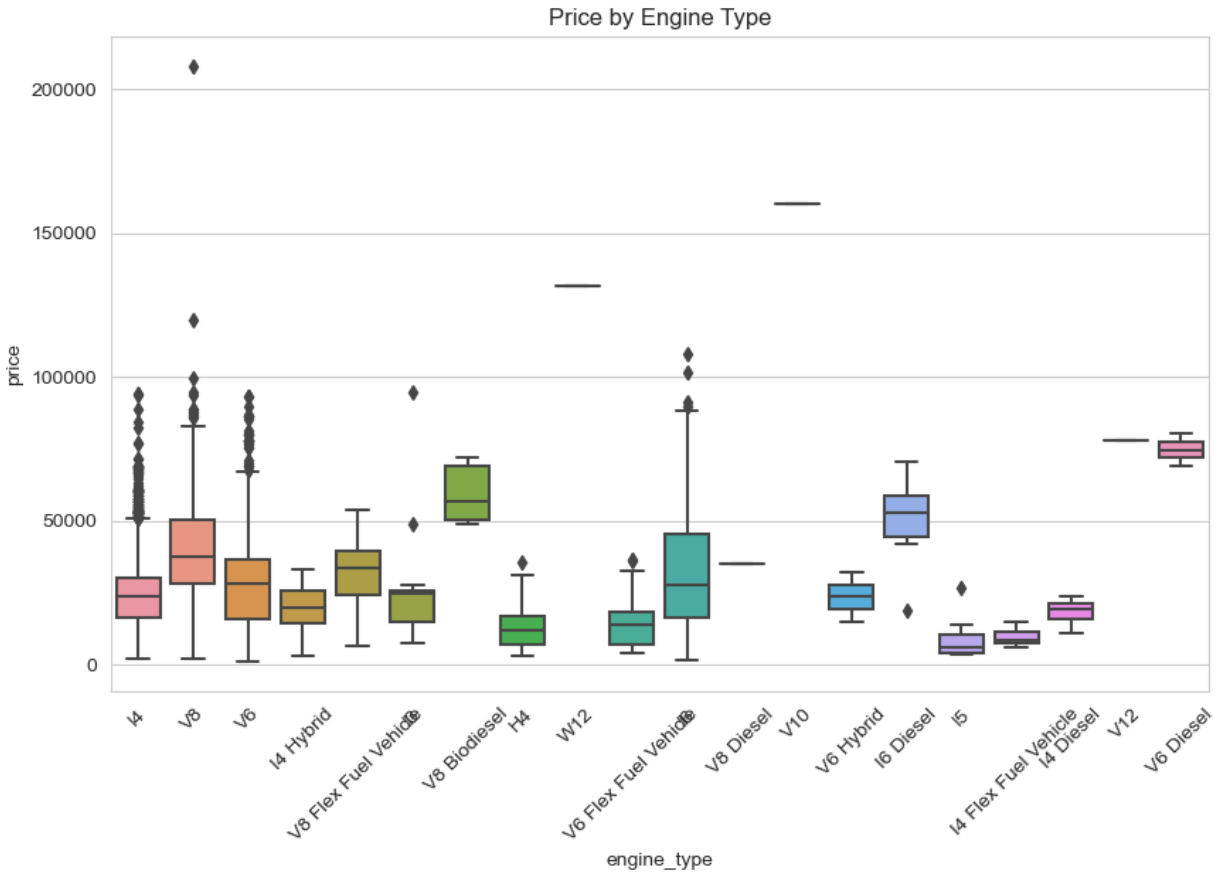
In [17]:
```
# 5. Body Type Distribution
plt.figure(figsize=(12, 6))
sns.boxplot(x=traindf['body_type'], y=traindf['price'])
plt.title('Price Distribution by Body Type')
plt.xticks(rotation=45)
plt.show()
```



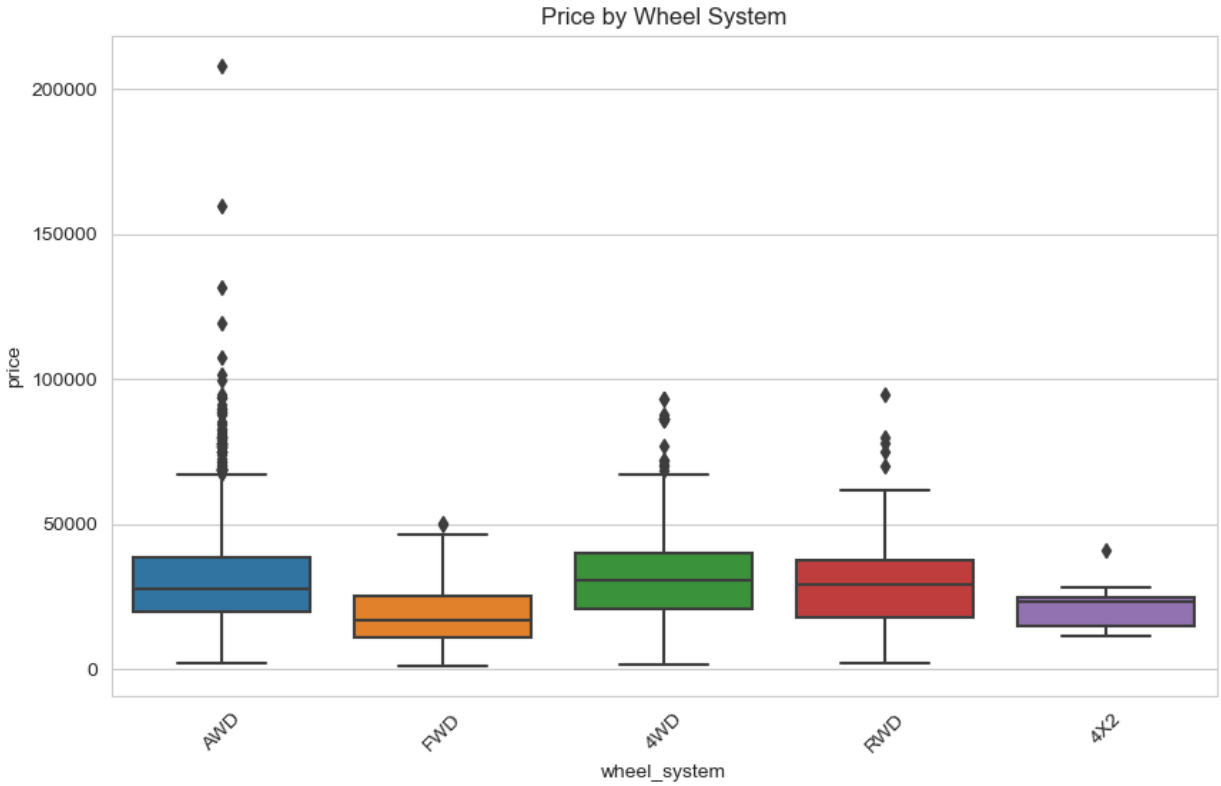
In [18]:
```
# 6. Transmission Type vs. Price
plt.figure(figsize=(8, 6))
sns.boxplot(x=traindf['transmission'], y=traindf['price'])
plt.title('Price by Transmission Type')
plt.xticks(rotation=45)
plt.show()
```



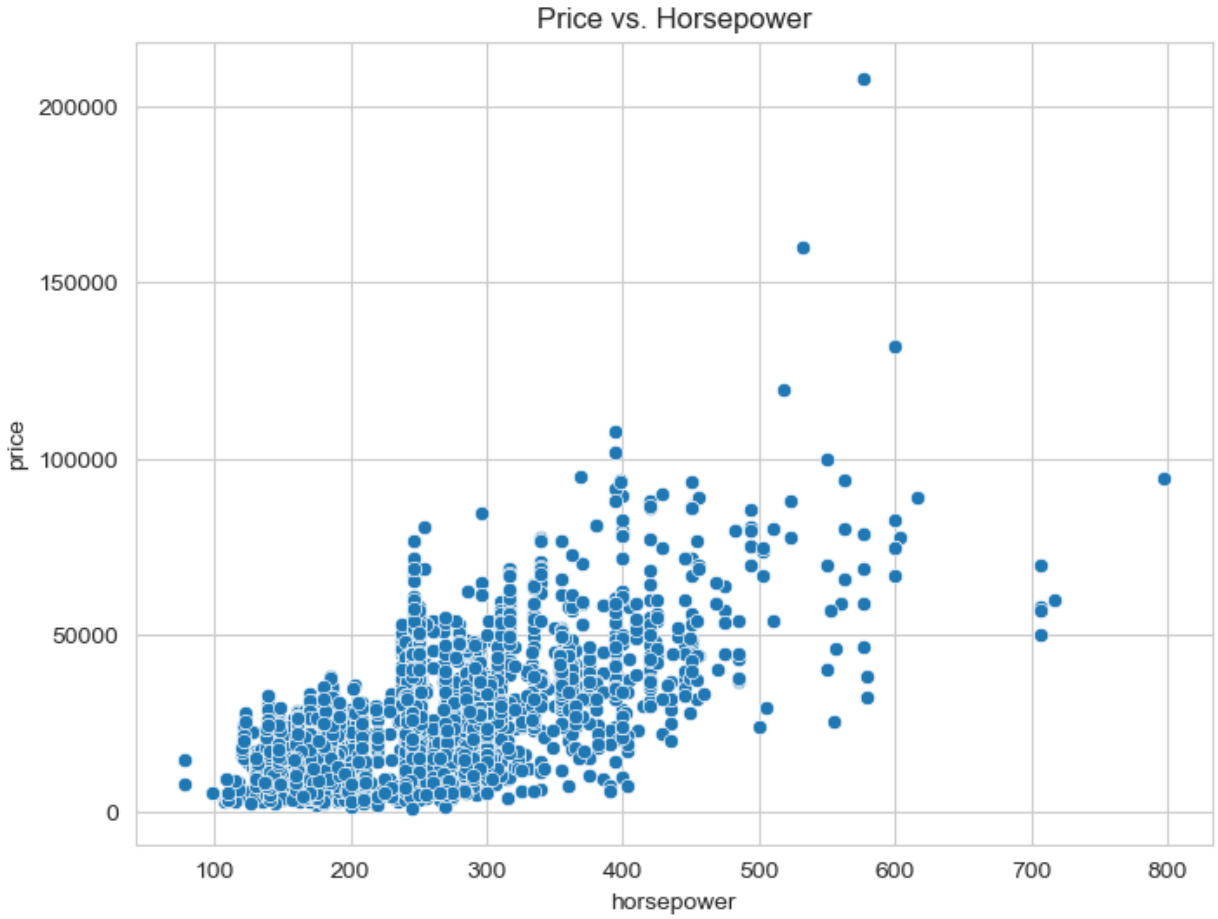
In [19]:
```
# 7. Price vs. Engine Type
plt.figure(figsize=(10, 6))
sns.boxplot(x=traindf['engine_type'], y=traindf['price'])
plt.title('Price by Engine Type')
plt.xticks(rotation=45)
plt.show()
```



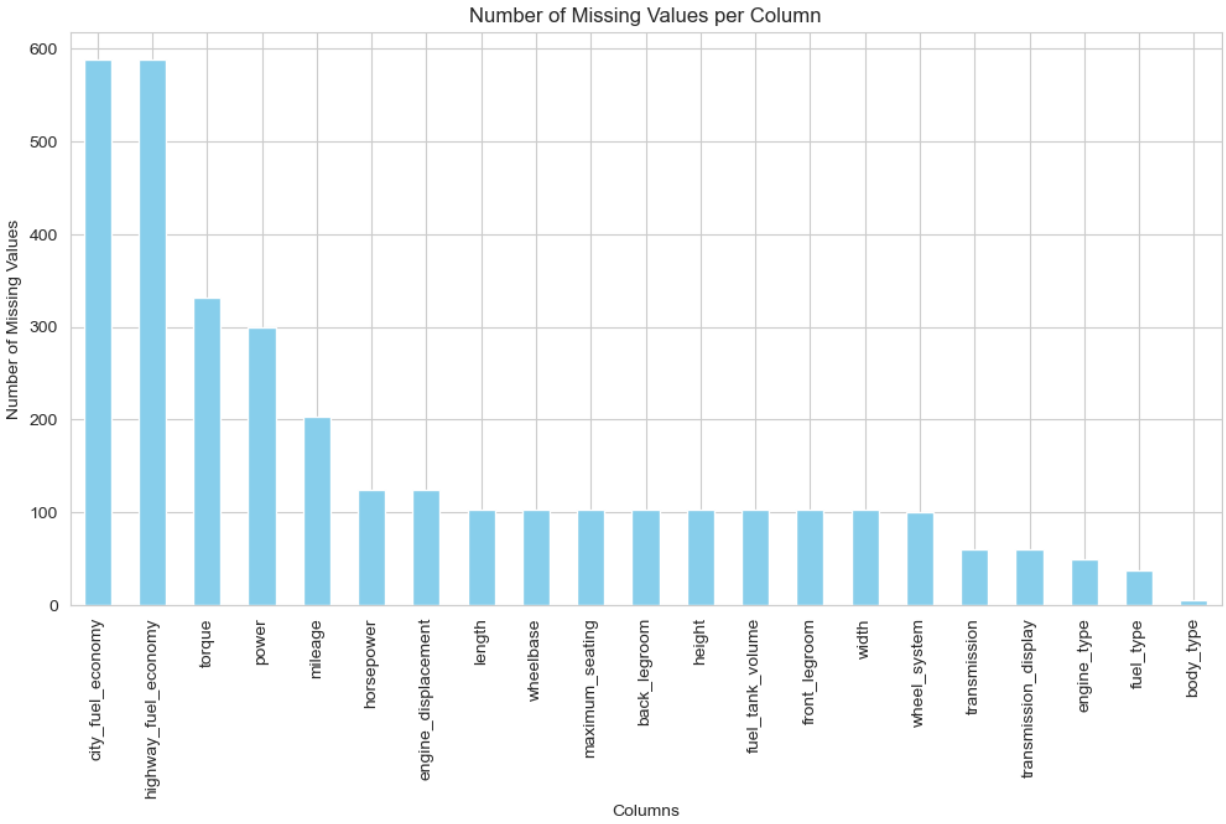
In [20]:
```
# 8. Distribution of Wheel System
plt.figure(figsize=(10, 6))
sns.boxplot(x=traindf['wheel_system'], y=traindf['price'])
plt.title('Price by Wheel System')
plt.xticks(rotation=45)
plt.show()
```



In [21]:
```
# 9. Horsepower vs. Price
plt.figure(figsize=(8, 6))
sns.scatterplot(x=traindf['horsepower'], y=traindf['price'])
plt.title('Price vs. Horsepower')
plt.show()
```



In [22]:
```
# 10. Number of Missing Values per Column
plt.figure(figsize=(12, 6))
missing_values_count = traindf.isnull().sum()
missing_values_count = missing_values_count[missing_values_count > 0].sort_values(asce
missing_values_count.plot(kind='bar', color='skyblue', title="Number of Missing Values
plt.ylabel('Number of Missing Values')
plt.xlabel('Columns')
plt.show()
```



Forecasting Problem

The objective is to forecast the cost of a used vehicle given its characteristics. To this end, we want to use data to develop a model that can predict a car's market worth based on a variety of criteria, including but not limited to its age, mileage, and engine type. In practical applications, this forecast is priceless. It guarantees a fair and lucrative pricing for the vendor. Instead, buyers may learn the vehicle's true market worth and avoid spending too much. The projections can also be useful for financial firms when determining loan amounts or insurance premiums. Thus, reliable pricing predictions improve honesty and confidence in the pre-owned vehicle trade.

Evaluation Criteria

Metrics that judge how well and how accurately forecasts were made are the backbone of any forecasting competition. Although traditional regression measures are not explicitly stated in the published requirements, the focus on regression approaches like random forest and gradient boosting implies that they may be used in this competition. Metrics like Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), which penalise greater deviations more severely, and R-squared, which represents the fraction of variance explained by the model, are frequently used for regression issues. It's also possible to evaluate the model based on how well it deals with missing data, how well it incorporates engineering into its design, and how resilient it is in general.

Types of Variables/Features

The dataset comprises a diverse set of variables that can be broadly categorized into the following types:

- Categorical Variables: These are variables that can take on one of a limited, and usually fixed, number of possible values. They represent qualitative data and are often non-numeric.
    - Examples from the dataset:
        - Sex: This can take values like 'Male' or 'Female'.
        - Marital status, Occupation, and Settlement size are other categorical variables in our dataset.
- Ordinal Categorical Variables: These are similar to categorical variables but have a clear order. They represent qualitative data where the order matters but the distance between categories is not defined.
    - Example from the dataset:
        - Education: It can have values like 'High School', 'Bachelor's', 'Master's', etc., where the order of education level is evident.
- Numerical Variables: These are variables that represent quantitative data and are numeric in nature. They can be further divided into:
    - Discrete: Variables that have a countable number of values. For instance, the number of doors in a car.
    - Continuous: Variables that can take any value within a range. For instance, age, income, or mileage of a car.
    - Examples from the dataset:
        - Age and Income are continuous numerical variables.
        - ID is a discrete numerical variable, representing unique identifiers for the entries.

Understanding the nature of these variables is fundamental as it guides the preprocessing steps, informs the choice of statistical methods, and impacts the model's performance. Different types of variables often require different handling techniques, especially in the realms of data visualization, feature engineering, and modeling.

Data Summary and Main Data Characteristics

The dataset, train.csv, comprises 3,500 entries, each representing a unique car listing. Across these listings, there are 39 variables capturing diverse attributes of the cars, ranging from technical specifications to broader categorizations. The main objective of this dataset is to predict the target variable, price, based on the other 38 feature variables.

Key Characteristics

- Diversity of Features: The dataset encapsulates a wide range of features, including categorical, ordinal, and numerical variables. This spectrum encompasses basic attributes like year and mileage to more nuanced ones such as body_type and engine_type.

- Distribution of Target Variable (price): Preliminary analysis indicates a right-skewed distribution, suggesting that a majority of cars are priced in the lower to mid-range bracket.
- Temporal Factor (year): The dataset spans various manufacturing years, potentially illustrating the depreciation of cars over time.
- Fuel Efficiency: Variables like city_fuel_economy and highway_fuel_economy provide insights into the car's fuel efficiency, a factor that might influence buyer decisions and consequently, the price.
- Variability in Car Types: The dataset is diverse in terms of car types, with multiple body_type classifications like SUV, Sedan, and Truck. Such variety caters to different market segments and can significantly influence pricing.
- Technical Specifications: Features like horsepower, engine_type, and transmission delve into the technical side of the vehicles, capturing performance and machinery aspects that can have direct implications on the car's value.
- Potential Data Quality Issues: While an in-depth cleaning process is yet to be undertaken, it's crucial to be vigilant about missing values, outliers, and potential inconsistencies in the dataset. An initial exploration did reveal some missing values, which will need addressing.

Understanding these characteristics is pivotal. It not only aids in formulating a suitable modeling strategy but also in conducting a comprehensive exploratory analysis, ensuring that the nuances of the dataset aren't overlooked.

Missing Values

The train.csv dataset exhibits missing values across multiple columns. Key columns include city_fuel_economy, highway_fuel_economy, and various technical specifications like torque and power. Some columns, such as transmission and engine_type, lack specific details about the car's machinery. Addressing these missing values is essential to ensure the robustness of predictive models.

Methods to Handle Missing Values:

For continuous variables like city_fuel_economy, imputation with the median or mean is recommended. Technical specifications, being continuous, can be similarly treated. For categorical columns like transmission and engine_type, the mode (most frequent value) is an appropriate imputation choice. In cases where relationships exist between variables, model-based imputation, using techniques like K-Nearest Neighbors or Random Forest, can be beneficial. For columns with a high proportion of missing values, consider dropping them. After imputation, validation through distribution checks is vital to ensure data integrity.

Task 2: Data Cleaning, Missing Observations and Feature Engineering

- In this task you will follow a set of instructions/questions listed below.
- Make sure you explain each answer carefully both in Markdown text and on your video.

Total Marks: 12

Task 2, Question 1: Clean **all** numerical features so that they can be used in training algorithms. For instance, back_legroom feature is in object format containing both numerical values and text. Extract numerical values (equivalently eliminate the text) so that the numerical values can be used as a regular feature.
(2 marks)

In [2]:
```
import pandas as pd
import re

# List of columns with mixed numeric and text values
columns_to_clean = ["back_legroom", "front_legroom", "fuel_tank_volume", "height", "le


# Function to extract numeric values and convert to float based on the format
def extract_numeric_value(text):
    if isinstance(text, str):
        numeric_part = re.search(r'\d+\.\d+', text)
        if numeric_part:
            return float(numeric_part.group())
    return None

for column in columns_to_clean:
    traindf[column] = traindf[column].apply(lambda x: extract_numeric_value(x))
    testdf[column] = testdf[column].apply(lambda x: extract_numeric_value(x))
```

I first identify the columns with mixed numeric and text values, and then create a function called extract_numeric_value. This function checks if a value is a text (string), and if so, it uses a regular expression to find and extract numeric patterns. The extracted numeric values are converted to floating-point numbers. If a valid numeric value is not found, the function returns None.

I then apply this extract_numeric_value function to clean the specified columns in both the traindf and testdf datasets. The result is that these columns will contain only the extracted numeric values, making them ready for analysis or modeling.

Task 2, Question 2 Create at least 5 new features from the existing numerical variables which contain multiple items of information, for example you could extract maximum torque and torque rpm from the torque variable.
(2 marks)

In [3]:
```
# Extract maximum torque from the 'torque' column
traindf['max_torque'] = traindf['torque'].str.extract(r'(\d+) lb-ft').astype(float)
testdf['max_torque'] = testdf['torque'].str.extract(r'(\d+) lb-ft').astype(float)
```

```
# Calculating the rear legroom ratio
traindf['rear_legroom_ratio'] = traindf['back_legroom'] / traindf['front_legroom']
testdf['rear_legroom_ratio'] = testdf['back_legroom'] / testdf['front_legroom']

# Calculating fuel efficiency of the average of city and highway fuel economy
traindf['fuel_efficiency'] = (traindf['city_fuel_economy'] + traindf['highway_fuel_ecc
testdf['fuel_efficiency'] = (testdf['city_fuel_economy'] + testdf['highway_fuel_econom

# Calculating the engine power ratio as horsepower / engine displacement
traindf['engine_power_ratio'] = traindf['horsepower'] / traindf['engine_displacement']
testdf['engine_power_ratio'] = testdf['horsepower'] / testdf['engine_displacement']

# Calculating car size as the product of length, width, and height
traindf['car_size'] = traindf['length'] * traindf['width'] * traindf['height']
testdf['car_size'] = testdf['length'] * testdf['width'] * testdf['height']
```

i created five features The maximum torque which indicates the maximum torque of cars, legroom ratio to show the ratio of rear legroom to front legroom, fuel efficiency of the average of city and highway fuel economy to reflects the average of city and highway fuel economy, engine power ratio as horsepower/engine displacement to denotes the ratio of horsepower to engine displacement, car sizes as product of length, width and height

Task 2, Question 3: Impute missing values for all features in both the training and test datasets. (3 marks)

In [5]:
```
# Imputing missing values for numerical features with the median value
numerical_features = ["city_fuel_economy", "engine_displacement", "horsepower", "milea
                      'car_age', 'max_torque', 'rear_legroom_ratio', 'fuel_efficiency'
                      "back_legroom", "front_legroom", "fuel_tank_volume", "height", "l
                       'highway_fuel_economy']
for feature in numerical_features:
    traindf[feature].fillna(traindf[feature].median(), inplace=True)
    testdf[feature].fillna(testdf[feature].median(), inplace=True)

# Imputing missing values for categorical features with the mode (most frequent value)
categorical_features = ["body_type", "engine_type","fuel_type","transmission", "transm
for feature in categorical_features:
    traindf[feature].fillna(traindf[feature].mode()[0], inplace=True)
    testdf[feature].fillna(testdf[feature].mode()[0], inplace=True)
```

Missing values in numerical features are imputed with the median value, ensuring central tendencies. For categorical features, imputation with the mode (most frequent value) ensures representative values. This data preprocessing enhances dataset completeness for analysis and modeling

Task 2, Question 4: Encode all categorical variables appropriately as discussed in class.

- Where multiple values are given for an observation encode the observation as 'other'.
- Where a categorical feature contains more than 5 unique values, map the features into 5 most frequent values + 'other' and then encode appropriately. For instance, map colours into 5 basic colours + 'other': [red, yellow, green, blue, purple, other] and then encode. (2 marks)

In [6]:
```
#custom mapping for 'listing_color'
listing_color_mapping = {
    'GRAY': 'GRAY',
    'BLACK': 'BLACK',
    'WHITE': 'WHITE',
    'SILVER': 'SILVER',
    'RED': 'RED'
}

# Encode 'listing_color'
traindf['listing_color'] = traindf['listing_color'].replace(listing_color_mapping)
testdf['listing_color'] = testdf['listing_color'].replace(listing_color_mapping)

other_colors = traindf['listing_color'].append(testdf['listing_color']).unique()
traindf['listing_color'] = traindf['listing_color'].apply(lambda x: x if x in other_co
testdf['listing_color'] = testdf['listing_color'].apply(lambda x: x if x in other_colo

#custom mapping for 'maximum_seating'
seating_mapping = {
    '1 seats': '1 seats',
    '2 seats': '2 seats',
    '3 seats': '3 seats',
    '4 seats': '4 seats',
    '5 seats': '5 seats'
}

# Encode 'maximum_seating'
traindf['maximum_seating'] = traindf['maximum_seating'].replace(seating_mapping)
testdf['maximum_seating'] = testdf['maximum_seating'].replace(seating_mapping)

other_seating = traindf['maximum_seating'].append(testdf['maximum_seating']).unique()
traindf['maximum_seating'] = traindf['maximum_seating'].apply(lambda x: x if x in othe
testdf['maximum_seating'] = testdf['maximum_seating'].apply(lambda x: x if x in other_
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_6856\688642629.py:15: FutureWarning: The s
eries.append method is deprecated and will be removed from pandas in a future versio
n. Use pandas.concat instead.
  other_colors = traindf['listing_color'].append(testdf['listing_color']).unique()
C:\Users\user\AppData\Local\Temp\ipykernel_6856\688642629.py:33: FutureWarning: The s
eries.append method is deprecated and will be removed from pandas in a future versio
n. Use pandas.concat instead.
  other_seating = traindf['maximum_seating'].append(testdf['maximum_seating']).unique
()
```

I applied custom encoding to the 'listing_color' variable, specifically mapping colors like gray, black, white, silver, and red to themselves, while categorizing all other colors as 'OTHERS.' Similarly, for the 'maximum_seating' variable, I encoded values ranging from 1 to 5 as they are, and categorized any values representing more seats as 'OTHERS.'

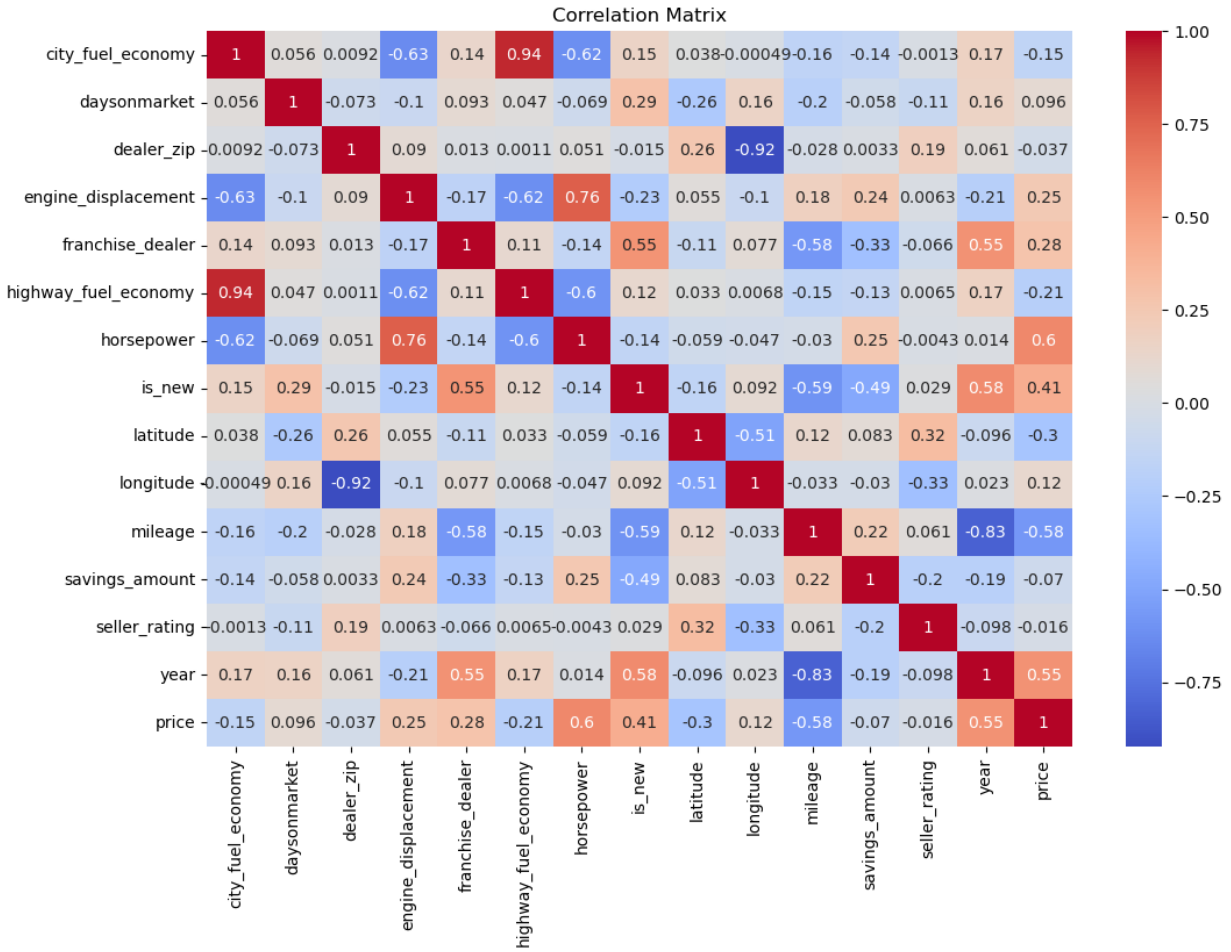
Task 2, Question 5: Perform any other actions you think need to be done on the data before constructing predictive models, and clearly explain what you have done.
(1 marks)

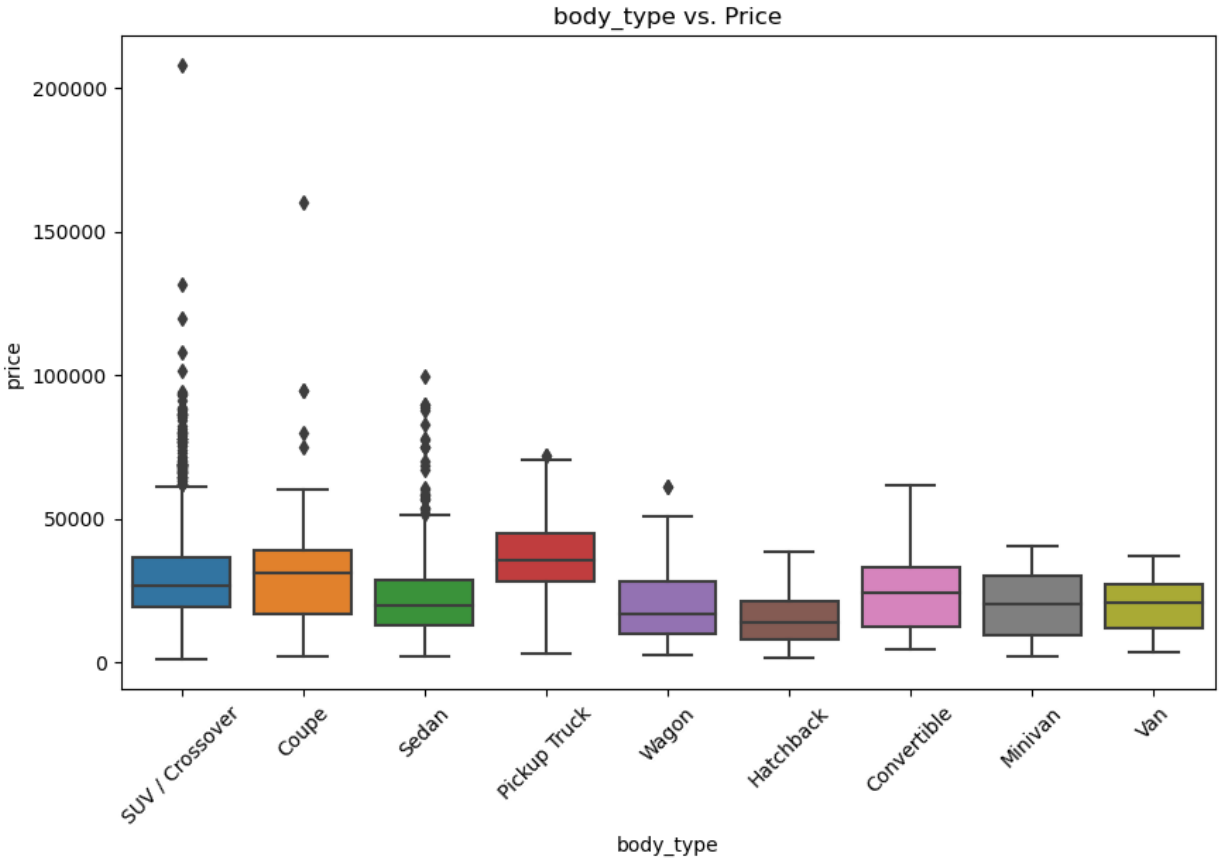
In [ ]:
```
# Calculating car age based on the 'year' column
traindf['car_age'] = 2023 - traindf['year']
testdf['car_age'] = 2023 - testdf['year']
```

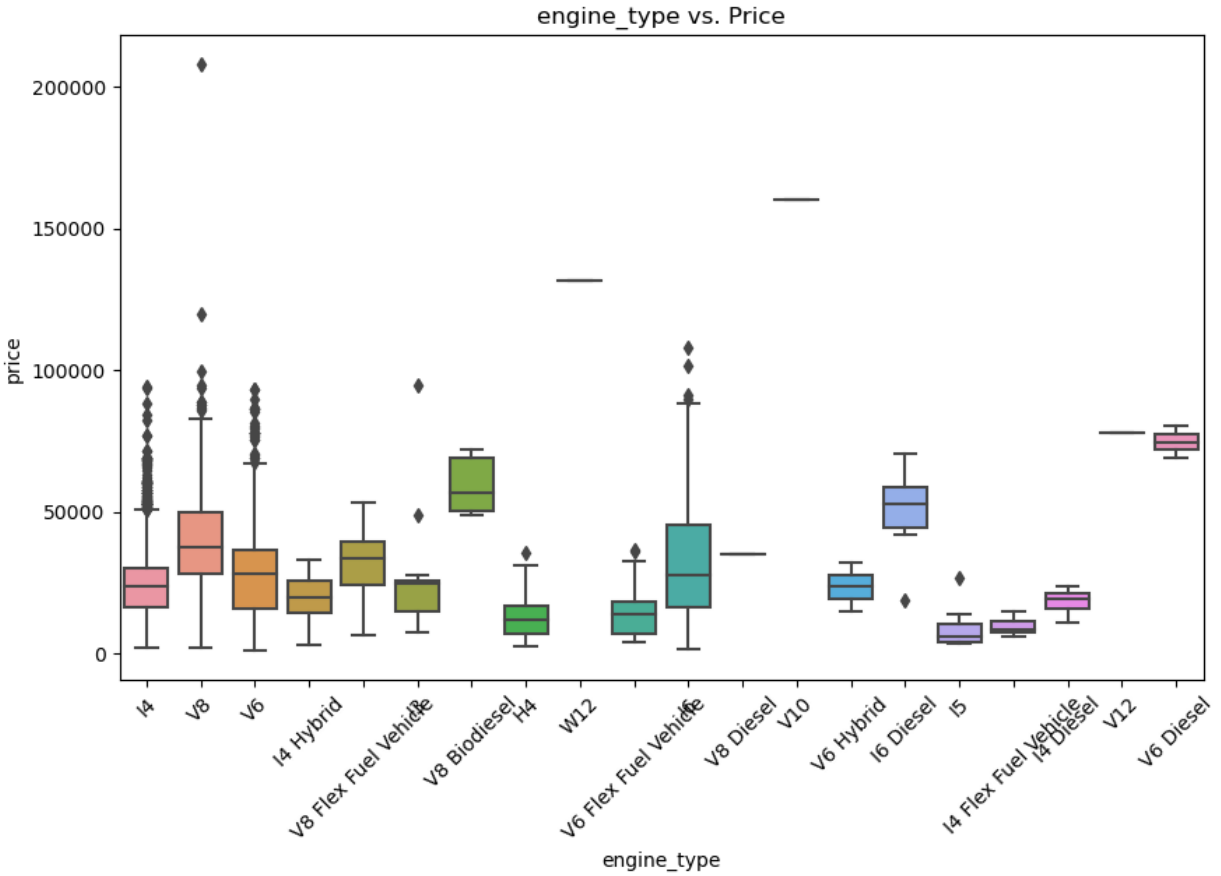
i created a new column 'car age' and added it to the traindf and testdf I believe that the "car age" column is essential and plays a crucial role as it directly reflects the car's durability and longevity.

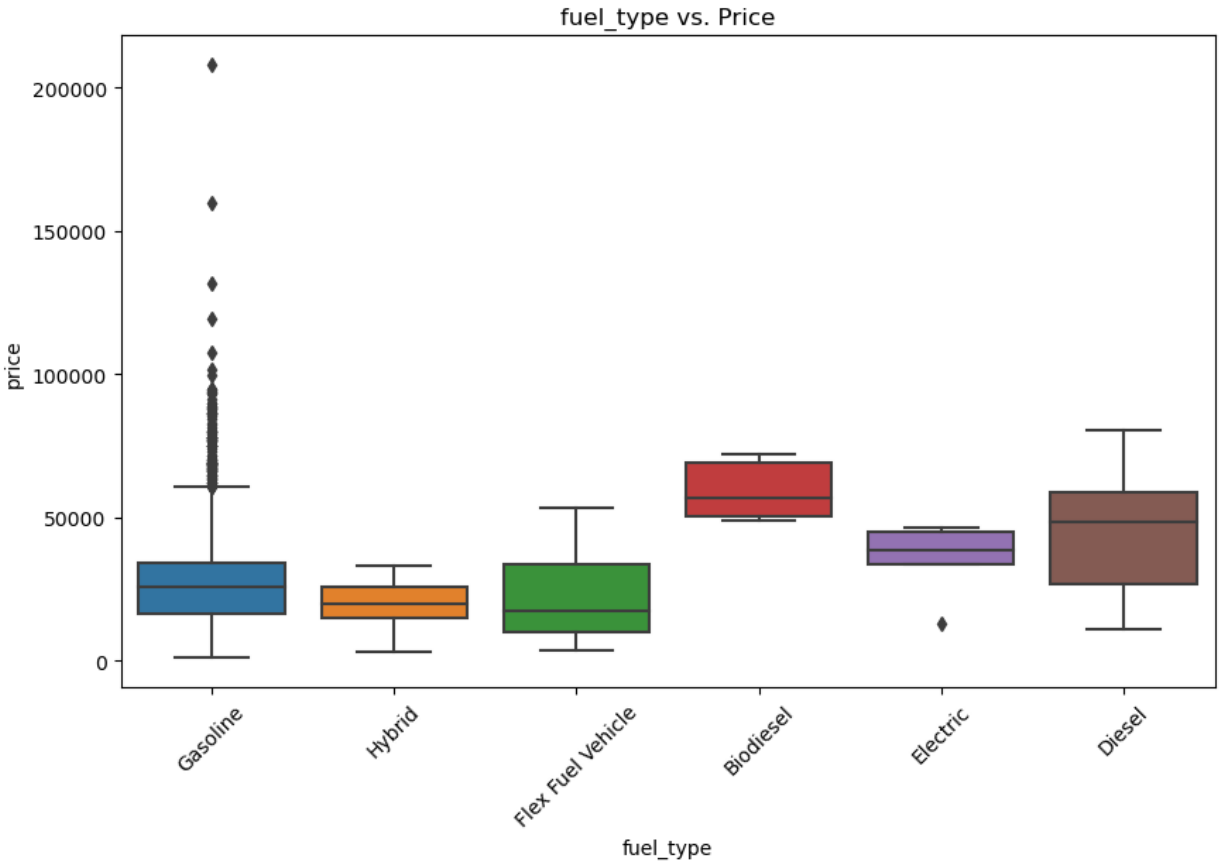
Task 2, Question 6: Perform exploratory data analysis to measure the relationship between the features and the target and carefully write up your findings. (2 marks)

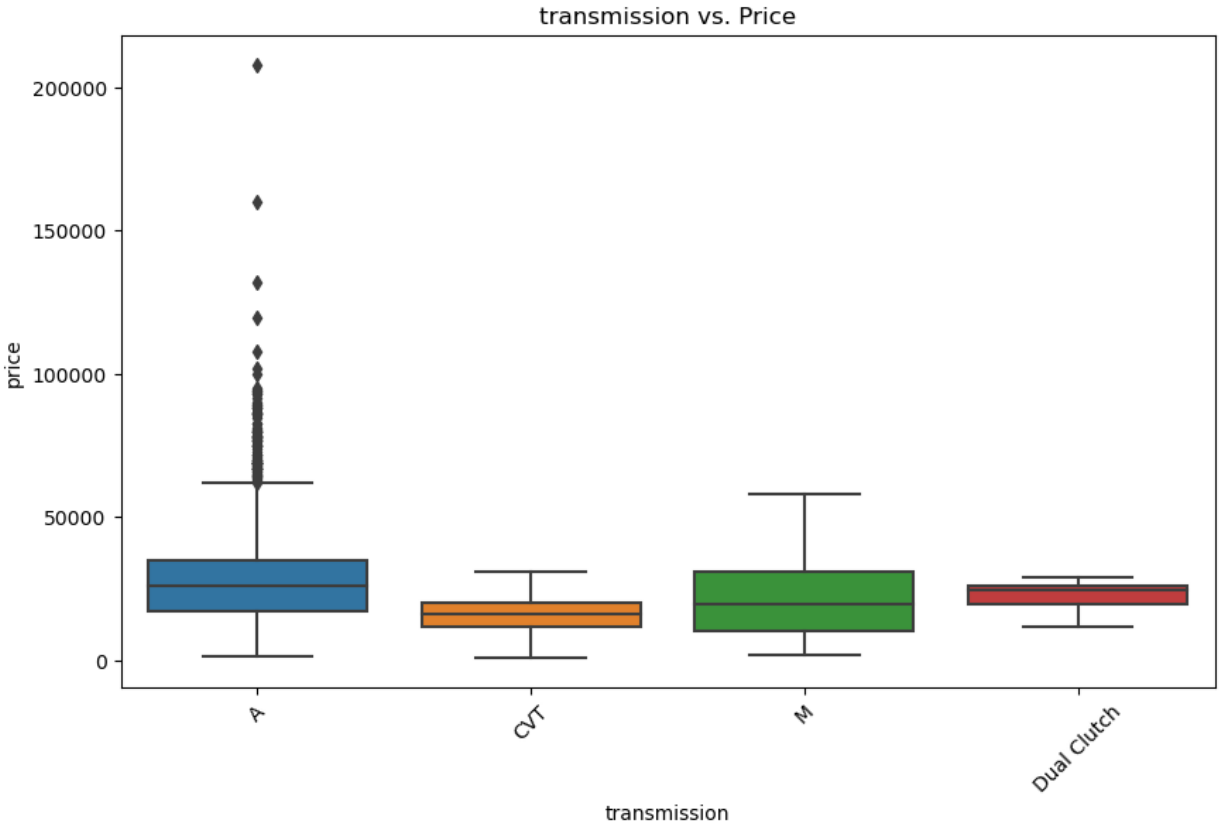
In [11]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Correlation matrix to measure the relationships between numerical features and the t
correlation_matrix = traindf.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()


# Boxplots for categorical features vs. target
categorical_features = ["body_type", "engine_type", "fuel_type",  "transmission", "tra
for feature in categorical_features:
    plt.figure(figsize=(10, 6))
    sns.boxplot(x=feature, y='price', data=traindf)
    plt.title(f'{feature} vs. Price')
    plt.xticks(rotation=45)
    plt.show()
```

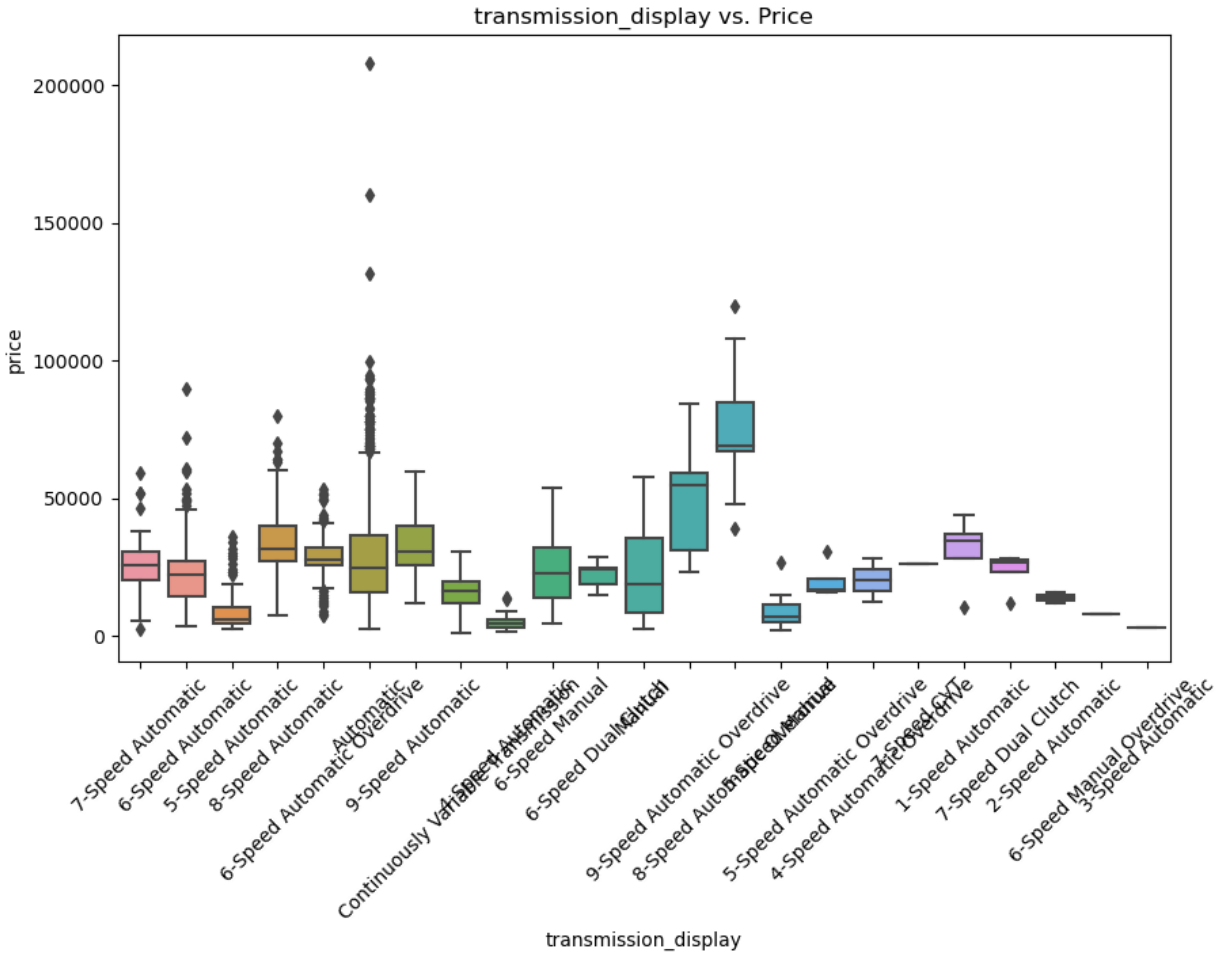


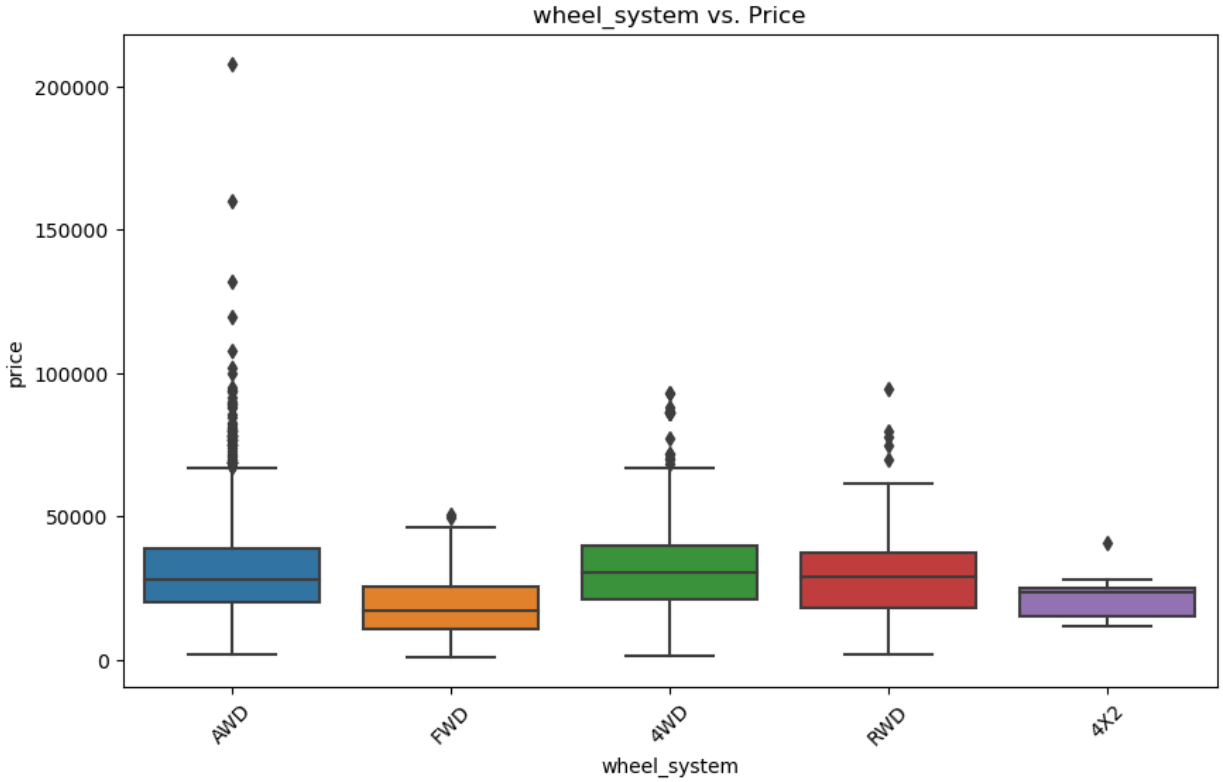

body_type vs. Price
price
200000
150000
100000
50000
0
SUV / Crossover
Coupe
Sedan
Pickup Truck
Wagon
Hatchback
Convertible
Minivan
Van
body_type


engine_type vs. Price
price
200000
150000
100000
50000
0
I4
V8
V6
I4 Hybrid
V8 Flex Fuel Vehicle
V8 Biodiesel
H4
W12
V6 Flex Fuel Vehicle
V8 Diesel
V10
V6 Hybrid
I6 Diesel
I5
I4 Flex Fuel Vehicle
I4 Diesel
V12
V6 Diesel
engine_type


fuel_type vs. Price
price
200000
150000
100000
50000
0
Gasoline
Hybrid
Flex Fuel Vehicle
Biodiesel
Electric
Diesel
fuel_type


transmission vs. Price
price
200000
150000
100000
50000
0
A
CVT
M
Dual Clutch
transmission





The heatmap displays the relationships, between variables using colors. The warmer colors like reds indicate correlations among all variables. In this case the variable "back_legroom" stands out as having the correlation with the target variable 'price.'

Correlation Strength: When a variable has a correlation it means that as that variables value increases the cars 'price' tends to increase as well. On the hand if there is a negative correlation, an increase in that variables value is associated with a decrease, in the 'price.'

Positive and Negative Correlations: By looking at the heatmap we can determine which variables have correlations (represented by colors) and which ones have negative correlations (represented by cool colors) with 'price.'

Comparing Categorical Features to Price:

When examining the boxplots for features we gain extra insights, into how these features are connected to the target variable (price):

Body Type and Price: The boxplot for 'body_type' indicates that pickup trucks generally have the highest average price, followed by coupes. However it's important to note that SUV/Crossover and Sedan categories have outliers suggesting that certain cars, in these categories have prices exceeding those of pickup trucks.

Engine. Price: By looking at the 'engine_type' boxplot we can observe the price ranges associated with engine types. This information helps us understand what we can expect in terms of car prices. Notably some mid range cars equipped with V6 Diesel engines appear to have prices.

Fuel Type vs. Price: The 'fuel_type' boxplot indicates that the most expensive cars are typically fueled by biodiesel. Gasoline-fueled cars are the most common, and there are outliers in various categories, suggesting that some cars in these categories can have notably high prices.

Transmission vs. Price: The 'transmission' boxplot shows that cars with 'A' (Automatic) transmission are the most common across all price ranges, including their outliers. Other transmission types appear to be more randomly scattered.

Transmission Display vs. Price: The 'transmission_display' boxplot displays different types of transmission displays and their distribution in the dataset. Notably, cars with '9-speed Automatic Overdrive' have the highest price range, while those with '8-speed Automatic' are spread out and contain outliers.

Wheel System vs. Price: The 'wheel_system' boxplot illustrates the distribution of various wheel systems in the dataset. 'AWD' (All-Wheel Drive) is the most common wheel system, followed by '4WD' (Four-Wheel Drive), and other categories.

Task 3: Fit and tune a forecasting model/Submit predictions/Report score and ranking

Make sure you **clearly explain each step** you do both in text and on the recoded video.
This task must not create any additional features and has to use on the dataset constructed in

Task 2.

1. Build at least 3 machine learning (ML) regression models taking into account the outcomes of Tasks 1 & 2 (Explain Carefully)
2. Fit the models and tune hyperparameters via cross-validation: make sure you comment and explain each step clearly
3. Select your best algorithm, create predictions using the test dataset, and submit your predictions on Kaggle's competition page
4. Provide Kaggle ranking and **score** (screenshot your best submission) and comment
5. Make sure your Python code works, so that a marker that can replicate your all Kaggle Score

- Hint: to perform well you will need to iterate Tasks 2 and Task 3.

Total Marks: 12

In [ ]:
```
#Task 3 code here
```

```
(Task 3 - insert more cells as required)
```

Marking Criteria

- Marking Rubrics
  - Problem Description - 12 marks
  - Data Cleaning - 12 marks
  - Building Forecasting models - 12 marks
  - Competition Points - 4 marks

- To receive full marks your solutions must satisfy the following criteria:
  - Provide Python solutions that follow the modelling methodology developed in BUSA8001
  - Written answers explain your logic and Python code in detail, and beformulated in easy to understand full sentences

---
