1. Data Understanding and Exploration

1.1 Importing all the important packages

The first step is to import all the important packages such as pandas, numpy, matplotlib, seaborn, etc.

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
pd.options.display.float_format = '{:.2f}'.format
```

1.2 Load the Data

To load the dataset, I will use pandas.

```
adv = pd.read_csv(r"C:\Users\asus\Desktop\Data_Science\Principles of Data Science\adverts.csv")
```

1.3 Sample Observation

We can use ". head()" to get an understanding of the data frame.

```
adv.head(2)
```

	public_reference	mileage	reg_code	standard_colour	standard_make	standard_model	vehicle_condition	year_of_registration	price	body_type	crossove
0	202006039777689	0.00	NaN	Grey	Volvo	XC90	NEW	NaN	73970	SUV	
1	202007020778260	108230.00	61	Blue	Jaguar	XF	USED	2011.00	7000	Saloon	

1.4 Meaning and Type of Features

To identify the column names and types of each column we can use ".info()".

```
adv.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 402005 entries, 0 to 402004
Data columns (total 12 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   public_reference      402005 non-null  int64  
 1   mileage               401878 non-null  float64
 2   reg_code              370148 non-null  object 
 3   standard_colour       396627 non-null  object 
 4   standard_make         402005 non-null  object 
 5   standard_model        402005 non-null  object 
 6   vehicle_condition     402005 non-null  object 
 7   year_of_registration  368694 non-null  float64
 8   price                 402005 non-null  int64  
 9   body_type             401168 non-null  object 
 10  crossover_car_and_van 402005 non-null  bool   
 11  fuel_type             401404 non-null  object 
dtypes: bool(1), float64(2), int64(2), object(7)
memory usage: 34.1+ MB
```

So, we have 402,005 rows and 12 columns in this data set.

Meaning and types of each feature

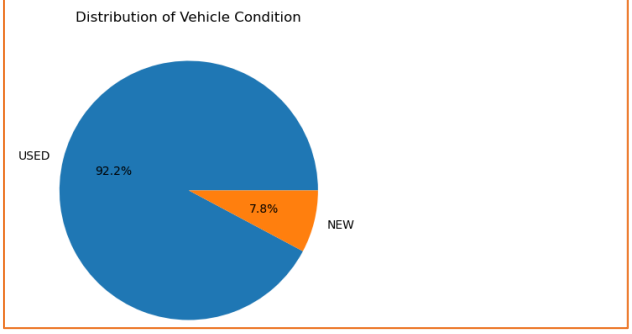
1. **public_reference (integer)**: A unique identification number for every vehicle.
2. **mileage (float):** The total distance that the vehicle has travelled.
3. **reg_code (object):** The vehicle's registration code to identify the year of registration.
4. **standard_color (object):** The exterior colour of the vehicle.
5. **standard_make (object)**: The vehicle's brand or manufacturer.
6. **standard_model (object):** The vehicle's specific model within the brand.
7. **vehicle_condition (object):** Determines whether the vehicle is new or used.
8. **year_of_registration (float)**: The year when the vehicle was first registered.
9. **price (integer):** The selling price of a vehicle.
10. **body_type (object):** The shape or style of the vehicle's body, such as SUV or sedan.
11. **crossover_car_and_van (boolean):** To distinguish between a car and a van.
12. **fuel_type (object):** The type of fuel the vehicle uses (e.g., petrol, diesel, hybrid, etc).

1.5 Analysis of Distributions

1.5.1 Pie-Chart of Vehicle Condition

```
plt.figure(figsize=(15, 5)) # Setting graph size
plt.pie(vehicle_condition_counts, labels=vehicle_condition_counts.index, autopct='%1.1f%%')

# Setting graph title and x and y labels
plt.title('Distribution of Vehicle Condition')
plt.show() # To display the graph
```



According to the pie chart, we can clearly say that the majority of the cars in our dataset are used cars.

I have created a variable with all the numeric columns to check the statistical summary of the data frame such as count, mean, std, min, and max.

1.5.2 Statistical Summary of Numeric Columns

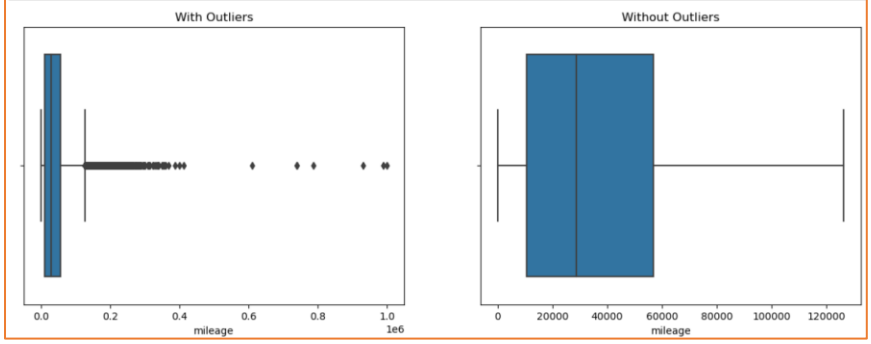
```
numeric_stats = adv[['mileage', 'year_of_registration', 'price']].describe()
numeric_stats
```

	mileage	year_of_registration	price
count	401878.00	368694.00	402005.00
mean	37743.60	2015.01	17341.97
std	34831.72	7.96	46437.46
min	0.00	999.00	120.00
25%	10481.00	2013.00	7495.00
50%	28629.50	2016.00	12600.00
75%	56875.75	2018.00	20000.00
max	999999.00	2020.00	9999999.00

Now, Let's visualize the above statistical summary in a boxplot

1.5.3 Box-Plot of Mileage

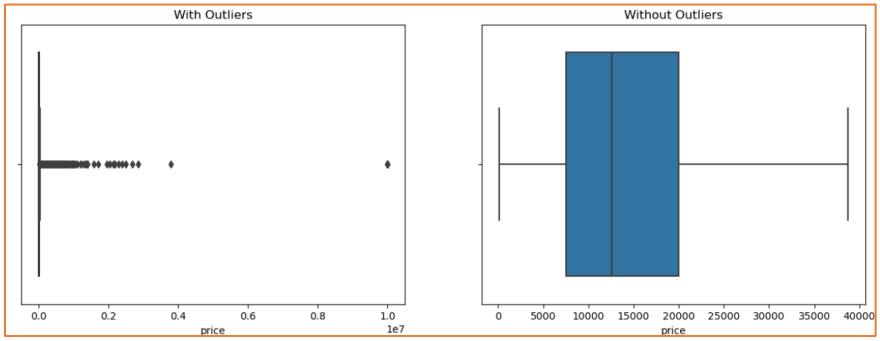
```
fig, axes = plt.subplots(1, 2, figsize=(15, 5)) # Setting graph size
# Visualization of Mileage column without outliers
sns.boxplot(data = adv, x='mileage', ax=axes[0])
axes[0].set_title('With Outliers')
# Visualization of Mileage column without outliers
sns.boxplot(data = adv, x='mileage', showfliers=False, ax=axes[1])
axes[1].set_title('Without Outliers')
plt.show() # To display the graph
```



Mileage with Outliers: The vehicle mileage distribution is shown in a boxplot graph, with multiple outliers extending towards high mileage values and a median that is noticeably lower than the upper quartile. Up to a million mileage values are displayed on the scale, with the majority of the data being concentrated in the lower half of the range.

Mileage without Outlier: The mileage distribution of vehicles is shown in the boxplot with no outliers, where most data points fall into a moderate range. The vehicle mileage values appear symmetrically distributed, with the median value lying approximately in the middle of the range.

1.5.4 Box-Plot of Price



Price with Outliers: The boxplot displays the distribution of prices for the items, with a few extreme outliers showing higher prices and the majority of prices concentrated at the lower end of the scale. There appears to be a skew towards lower values as the median price is closer to the lower quartile.

Price without outliers: The majority of the prices are concentrated in a smaller range, as seen by this boxplot, which displays the range of prices excluding outliers. A line in the middle of the box represents the median price and suggests the dataset's central tendency.

1.5.5 Box-Plot of Year of Registration

Year of Registration with Outliers: The year of registration distribution for cars is shown in a boxplot, which places an average of registration years closer to the present and a few outliers representing much older cars.

Year of Registration without Outliers: The year of registration for a group of cars is displayed in this boxplot, which demonstrates a close grouping of years towards the most recent end of the scale.

2. Data Pre-Processing

Data pre-processing is the process of preparing raw data for analysis by correcting or eliminating errors, inconsistencies, and irrelevant data points. This process involves recognizing and fixing issues such as missing values, outliers, duplicate entries, and errors in data entry or formatting. The goal of data cleaning is to ensure that the data is accurate, complete, and in an analysis-ready format, resulting in more reliable and meaningful insights from the data. This is an important step because data quality has a direct impact on the results of any subsequent analyses.

2.1 Data Cleaning

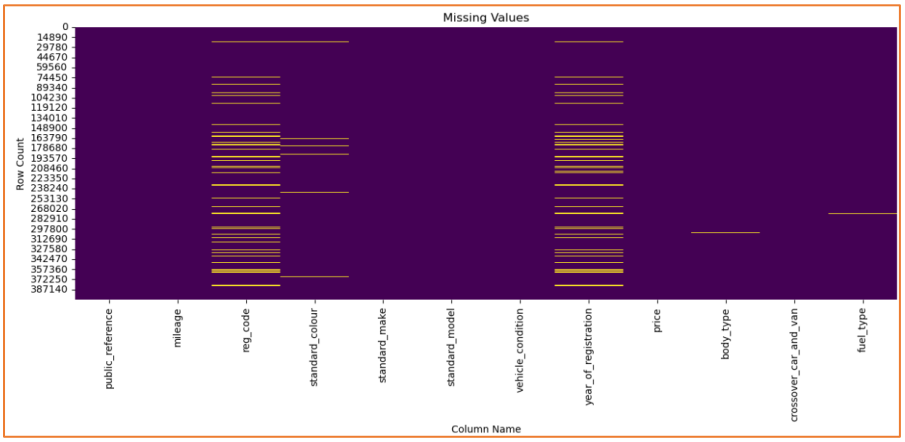
Data cleaning is an important step in data preparation because it ensures that the data is accurate, complete, and of high quality before being used for analysis. Good data leads to sound conclusions and informed decisions.

2.1.1 Filling Missing Values

```
1  adv.isnull().sum()
```

```
public_reference            0
mileage                   127
reg_code                31857
standard_colour          5378
standard_make               0
standard_model              0
vehicle_condition           0
year_of_registration    33311
price                       0
body_type                 837
crossover_car_and_van       0
fuel_type                 601
dtype: int64
```

Let's visualize the above data in a heatmap



Now here we can see there are a few columns with missing values that are important for analysis. We need to fill those in for further and accurate analysis. To begin with year of registration column has around 33311 rows with missing values and the latest year in the data set is 2020 so by using condition I will fill a few rows with the year 2020. So, the condition would be vehicle condition is new and yor isnull.

```
1  adv.loc[(adv['vehicle_condition'] == 'NEW') & (adv['year_of_registration'].isnull()), 'year_of_registration'] = 2020
```

I am filling in missing values in the reg_code with 20 because as per the link cars that are registered in the year 2020 they have reg_code as 20 and 69. The reason I am using 20 here is that the cars that are sold from 1st March - 31st August the reg_code is 20 and as per the condition the cars that have mileage less than 1000 would be the cars that are registered in this period.

```
1  con1 = adv.loc[(adv['reg_code'].isna() | (adv['reg_code'] == '')) &
2          (adv['year_of_registration'] == 2020) &
3          (adv['mileage'] <= 1000), 'reg_code'] = '20'
```

I am filling in missing values in the reg_code with 69 because as per the link cars that are registered in the year 2020 they have reg_code as 20 and 69. The reason I am using 69 here is because for the cars that are sold from 1st September - 28/29th February the reg_code is 69 and as per the condition the cars that have mileage greater than 1000 would be the cars that are registered in this period.

```
con2 = adv.loc[(adv['reg_code'].isna() | (adv['reg_code'] == '')) &
        (adv['year_of_registration'] == 2020) &
        (adv['mileage'] >= 1000), 'reg_code'] = '69'
```

Now, to fill in the remaining values of the year of registration and mileage column I will be using a median number for each column.

```
median_mileage = adv['mileage'].median()
median_yor = adv['year_of_registration'].median()
print('Median of Mileage column is:', median_mileage)
print('Median of Mileage column is:', median_yor)
```

```
Median of Mileage column is: 28629.5
Median of Mileage column is: 2017.0
```

```
# Filling missing values using median for mileage and year of registration
adv['mileage'] = adv['mileage'].fillna(median_mileage)
adv['year_of_registration'] =adv['year_of_registration'].fillna(median_yor)
```

Now, I'll mark categorical columns like body_type, fuel_type, and standard_colour as Unknown. I could also fill in those with the most common value from each column, but I'm not doing that because it would be incorrect data and could affect our analysis, and these are the columns with the fewest missing values.

```
# Filling missing values of "body_type" column
adv['body_type'] = adv['body_type'].fillna('Unknown')
# Filling missing values of "fuel_type" column
adv['fuel_type'] = adv['fuel_type'].fillna('Unknown')
# Filling missing values of "standard_colour" column
adv['standard_colour'] = adv['standard_colour'].fillna('Unknown')
```

```
adv.isnull().sum()
```

```
public_reference         0
mileage                  0
reg_code                 0
standard_colour          0
standard_make            0
standard_model           0
vehicle_condition        0
year_of_registration     0
price                    0
body_type                0
crossover_car_and_van    0
fuel_type                0
dtype: int64
```

I have filled in all the missing values of each column the next step is to deal with outliers.

2.1.2 Dealing with Outliers

I will use the Interquartile Range (IQR) to deal with outliers in mileage, year of registration, and price column. The Interquartile Range (IQR) is a statistical dispersion measure calculated as the difference between a dataset's 75th (Q3) and 25th (Q1) percentiles. It represents the range in which the middle 50% of the data falls. Potential outliers include values that are significantly lower than Q1 – (1.5*IQR) or higher than Q3 + (1.5*IQR). This method is beneficial because it is less influenced by extreme values than other measures.

Step 1 (Price)

```
# Finding 25 and 75 percentile of price
prices = adv['price'].values
q1_price,q3_price = np.percentile(prices,[25,75])
print('Price range',q1_price,q3_price)
```

```
Price range 7495.0 20000.0
```

Step 2 (Price)

```
# Find IQR for price

price_iqr = q3_price - q1_price
print('IQR for price is',price_iqr)
```

```
IQR for price is 12505.0
```

Step 3 (Price)

```
# Finding Lower and Upper bound for price

lower_bound_price_val = q1_price - (1.5 * price_iqr)
upper_bound_price_val = q3_price + (1.5 * price_iqr)
print('lower bound value for price',lower_bound_price_val)
print('Upper bound value for price',upper_bound_price_val)
```

```
lower bound value for price -11262.5
Upper bound value for price 38757.5
```

Step 1 (Mileage)

```
# Finding 25 and 75 percentile of mileage
mileage = adv['mileage'].values
q1_mileage,q3_mileage = np.percentile(mileage,[25,75])
print('Mileage range', q1_mileage,q3_mileage)
```

```
Mileage range 10487.0 56852.0
```

Step 2 (Mileage)

```
# Find IQR for mileage

mileage_iqr = q3_mileage - q1_mileage
print('IQR for mileage is',mileage_iqr)
```

```
IQR for mileage is 46365.0
```

Step 3 (Mileage)

```
# Finding Lower and Upper bound for mileage

lower_bound_mileage_val = q1_mileage - (1.5 * mileage_iqr)
upper_bound_mileage_val = q3_mileage + (1.5 * mileage_iqr)
print('lower bound value for mileage',lower_bound_mileage_val)
print('Upper bound value for mileage',upper_bound_mileage_val)
```

```
lower bound value for mileage -59060.5
Upper bound value for mileage 126399.5
```

Step 1 (Year of Registration)

```
# Finding 25 and 75 percentile of year of registration
yor = adv['year_of_registration'].values
q1_yor,q3_yor = np.percentile(yor,[25,75])
print('Year of registration range', q1_yor,q3_yor)
```

```
Year of registration range 2014.0 2018.0
```

Step 2 (Year of Registration)

```
# Find IQR for Year of registration

yor_iqr = q3_yor-q1_yor
print('IQR for year of registration is', yor_iqr)
```

```
IQR for year of registration is 4.0
```

Step 3 (Year of Registration)

```
# Finding Lower and Upper bound for year of registration

lower_bound_yor_val = q1_yor - (1.5 * yor_iqr)
upper_bound_yor_val = q3_yor + (1.5 * yor_iqr)
print('lower bound value for year of registration',lower_bound_yor_val)
print('Upper bound value for year of registration',upper_bound_yor_val)
```

```
lower bound value for year of registration 2008.0
Upper bound value for year of registration 2024.0
```

By using the output of step 3, the upper and lower bounds of each column I will remove the outliers

```
# Using upper and lower bound to remove the outliers from the dataset
adv = adv.query('price <= @upper_bound_price_val')
adv = adv.query('mileage <= @upper_bound_mileage_val')
adv = adv.query('year_of_registration >= @lower_bound_yor_val')
```

After removing the outliers my new dataset length is 348523 rows and 12 columns.

```
adv.shape
```

```
(348523, 12)
```

2.2 Feature Engineering

In data science, feature engineering is the process of converting raw data into more useful and informative features for further analysis. It entails creating new variables or modifying existing ones to better capture the underlying patterns and relationships in the data, resulting in more accurate and insightful analytical outcomes. This step is critical in preparing data for effective analysis and modelling.

```
# Here, I am changing values of new and used vehicle condition for further analysis
adv['vehicle_condition'] = adv['vehicle_condition'].map({'NEW': True, 'USED': False})
```

```
# Changing data type of year of registration coloumn as integer
adv['year_of_registration'] = adv['year_of_registration'].astype(int)
```

I have changed the data type of the column vehicle condition and year of registration for further analysis.

```
adv.head(1)
```

index	public_reference	mileage	reg_code	standard_colour	standard_make	standard_model	vehicle_condition	year_of_registration	price	body_type	cr
0	1	202007020778260	108230.00	61	Blue	Jaguar	XF	False	2011	7000	Saloon

I am creating a new column “vehicles_age” to identify the age of a vehicle by subtracting the current year (i.e. 2020) from the year of registration for further analysis.

```
adv['vehicles_age'] = 2023 - adv['year_of_registration']
adv.head(1)
```

ode	standard_colour	standard_make	standard_model	vehicle_condition	year_of_registration	price	body_type	crossover_car_and_van	fuel_type	vehicles_age
61	Blue	Jaguar	XF	False	2011	7000	Saloon	False	Diesel	12

2.3 Subsetting (e.g., feature selection and row sampling)

Subsetting is the process of selecting certain parts of your data for further analysis. This process can include two major aspects. Feature selection is the process of selecting only specific columns from a dataset, and row sampling is the process of selecting a subset of the dataset's rows. Both of these practices are important for making data more manageable and increasing analysis efficiency.

2.3.1 Feature Selection

I am removing the crossover_car_and_van, public_reference, and reg_code columns from the dataset because it is obvious that they play no role in predicting the vehicle's price. I am removing that column from the dataset because it is an irrelevant feature.

```
adv.drop('public_reference', axis=1, inplace=True)
adv.drop('crossover_car_and_van', axis=1, inplace=True)
adv.drop('reg_code', axis=1, inplace=True)
```

After removing the unnecessary columns, I will rearrange the dataset for better data framing and understanding.

```
# Rearranging the columns
adv_temp = ['standard_make', 'standard_model', 'standard_colour', 'body_type',
            'fuel_type', 'vehicle_condition', 'mileage', 'year_of_registration', 'vehicles_age', 'price']
```

I have created a new variable to rearrange the columns. So, I will reassign the new variable to my actual dataset name (i.e. adv)

```
adv_temp = adv[adv_temp]
```

```
adv = adv_temp
```

After doing feature engineering and feature selection below is the new data frame which we will use for analysis.

```
adv.head(1)
```

	standard_make	standard_model	standard_colour	body_type	fuel_type	vehicle_condition	mileage	year_of_registration	vehicles_age	price
0	Jaguar	XF	Blue	Saloon	Diesel	False	108230.00	2011	12	7000

2.3.2 Row Sampling and Subsetting

I have created a variable called jaguar_details to get all the information of the manufacturer Jaguar.

```
jaguar_details = adv[adv['standard_make'] == 'Jaguar']
jaguar_details.head(3)
```

	standard_make	standard_model	standard_colour	body_type	fuel_type	vehicle_condition	mileage	year_of_registration	vehicles_age	price
0	Jaguar	XF	Blue	Saloon	Diesel	False	108230.00	2011	12	7000
141	Jaguar	XF	White	Saloon	Diesel	False	15264.00	2017	6	18725
201	Jaguar	E-PACE	Red	SUV	Diesel	False	5000.00	2020	3	31900

I have created 2 subsets of the dataset called new_cars and used_cars.

```
new_cars = adv[adv['vehicle_condition'] == True]
new_cars.head(1)
```

	standard_make	standard_model	standard_colour	body_type	fuel_type	vehicle_condition	mileage	year_of_registration	vehicles_age	price
15	Nissan	X-Trail	Unknown	SUV	Diesel	True	5.00	2020	3	27595

```
used_cars = adv[adv['vehicle_condition'] == False]
used_cars.shape
```

```
(325577, 10)
```

3. Analysis of Associations and Group Differences

The analysis of association and group differences employs statistical methods to examine and quantify the relationships between variables, as well as to compare differences between groups. Analysis of Association aims to determine whether and how variables are related to one another, whereas Analysis of Group Differences seeks to identify and quantify differences among distinct groups within the data.

3.1 Correlation Matrix

I have already created a variable for numeric columns (mileage, year of registration and price), I will use ".append()" to add vehicles_age and vehicle_condition. This correlation matrix aims to identify the best predictor of the price of a vehicle.

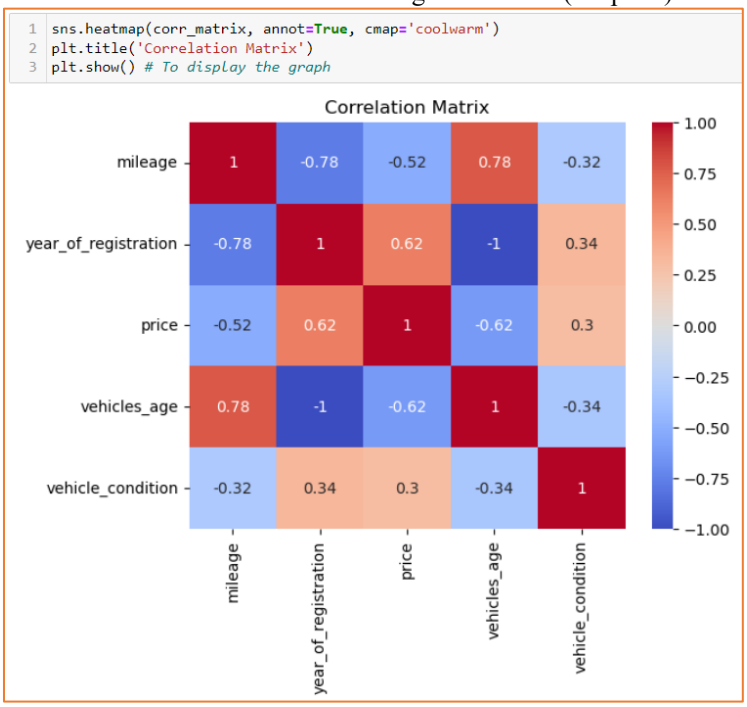
```
numeric_cols.append('vehicles_age')
numeric_cols.append('vehicle_condition')
```

After creating a new variable and using ".corr()" I will get a correlation summary of all the columns we have created in that variable.

```
# Creating a new variable with all the numeric columns we have to check the best price predictor
numeric_data = adv[numeric_cols]
corr_matrix = numeric_data.corr()
```

I will create a heatmap to visualise the correlation with our targeted column (i.e. price)

```
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show() # To display the graph
```



Based on the correlation matrix the best price predictors appear to be

1. year_of_registration (0.62): This correlates positively with price, indicating that newer vehicles are more expensive. Because the value is so close to 1, this is most likely one of the strongest predictors in the matrix.

2. vehicles_age (-0.62): This has a strong negative correlation with price, implying that vehicle prices decrease as they age. It is the inverse of year_of_registration and has the same absolute value, making it equally effective as a predictor.

3. vehicle_condition (0.3): There is a weak positive correlation, suggesting that the condition of the vehicle has a less pronounced but still positive relationship with the price.

4. mileage (-0.52): There is a moderate negative correlation between mileage and price, indicating that vehicles with higher mileage are more likely to be cheaper.

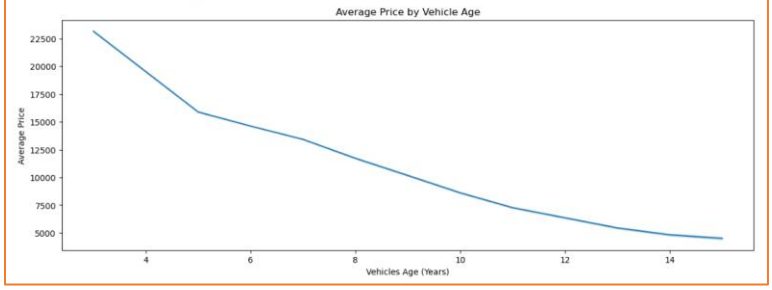
3.2 Group Analysis

3.2.1 Quantitative-Quantitative

3.2.1.1 Vehicles Age Vs. Price

I am comparing a vehicle's age with price to understand how vehicle age affects price after year by year.

```
# Creating a visualisation of vehicles_age and price to identify how vehicles age impact the price
plt.figure(figsize=(15, 5)) # Setting graph size
sns.lineplot(x='vehicles_age', y='price',data=adv)
# Setting graph title and x and y labels
plt.title('Average Price by Vehicle Age')
plt.xlabel('Vehicles Age (Years)')
plt.ylabel('Average Price')
plt.show() # To display the graph
```

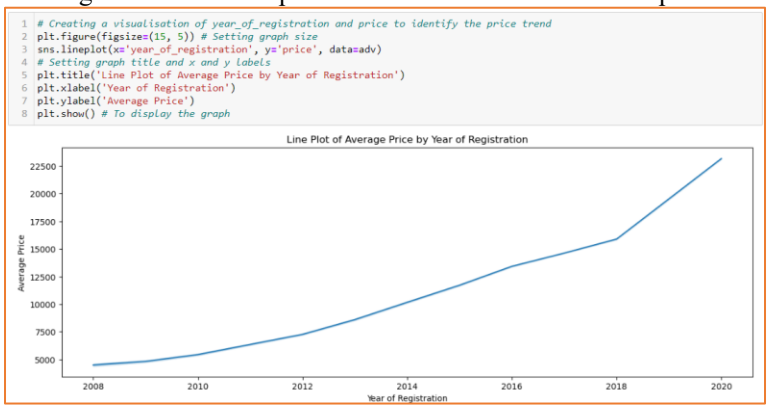


The graph shows a declining trend in the average price of vehicles as they age, from about 3 to 14 years. As vehicles get older, their value decreases.

3.2.1.2 Year of Registration Vs. Price

I am comparing the year of registration with the price to understand the trend in the price of new cars.

```
# Creating a visualisation of year_of_registration and price to identify the price trend
plt.figure(figsize=(15, 5)) # Setting graph size
sns.lineplot(x='year_of_registration', y='price', data=adv)
# Setting graph title and x and y labels
plt.title('Line Plot of Average Price by Year of Registration')
plt.xlabel('Year of Registration')
plt.ylabel('Average Price')
plt.show() # To display the graph
```



The average price of cars has been increasing over time. There is a noticeable acceleration in the average price starting around the year 2016.

3.2.1.3 Year of Registration Vs. Price based on Fuel Type

I am comparing the Year of Registration with Price based on fuel type to understand the trends and changes in the average prices across different fuel types over time.

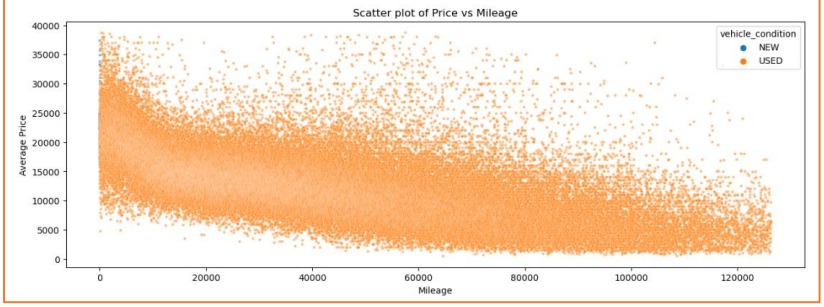
```
# Comparing year_of_registration and average price of each year based on fuel type
grouped_data1 = adv.groupby(['year_of_registration', 'fuel_type'])['price'].mean().reset_index()
plt.figure(figsize=(15, 5)) # Setting graph size
sns.barplot(x='year_of_registration', y='price', hue='fuel_type', data=grouped_data1)
# Setting graph title and x and y labels
plt.title('Average price of a car based on fuel type over the years')
plt.xlabel('Year of Registration')
plt.ylabel('Average Price')
plt.xticks(rotation=90)
plt.show() # To display the graph
```

As we can see from the graph certain fuel types such as electric or hybrid vehicles, show an upward trend in average prices, which could indicate increased demand or technological advancements. Conversely, traditional fuel types like petrol or diesel might show different trends. This graph provides a comparative view of how car prices vary with fuel type and registration year.

3.2.1.4 Mileage vs. Price based on Vehicle Condition

I am comparing Mileage with price based on vehicle condition. The purpose of this comparison is to analyse how mileage affects car prices differently for new and used vehicles.

```
# Visualising how mileage impacts the car price based on vehicle condition
grouped_data2 = adv.groupby(['mileage', 'vehicle_condition'])['price'].mean().reset_index()
plt.figure(figsize=(15, 5)) # Setting graph size
sns.scatterplot(x='mileage', y='price', hue='vehicle_condition',data=grouped_data2, alpha=0.5, s=10)
# Setting graph title and x and y labels
plt.title('Scatter plot of Price vs Mileage')
plt.xlabel('Mileage')
plt.ylabel('Average Price')
plt.show() # To display the graph
```



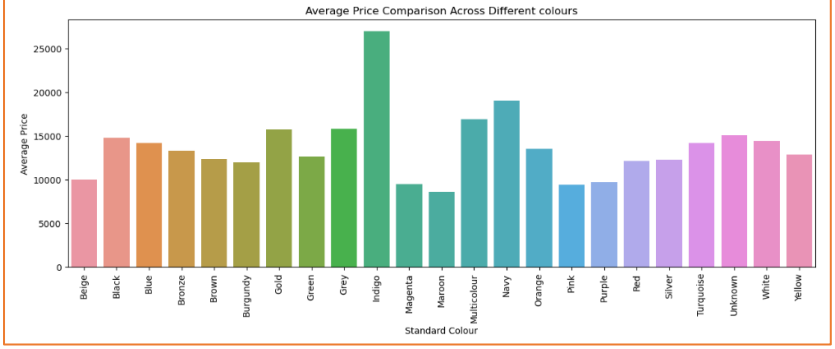
The scatter plot shows that, as a car's mileage increases, its average price decreases, indicating that its value will depreciate with use. New cars have lower mileage and are often more expensive, whereas used cars have a wide range of mileage and prices, with high-mileage used cars typically being less expensive.

3.2.2 Quantitative-Categorical

3.2.2.1 Standard Colour Vs. Price

I am comparing the standard colour with the average price to analyse how much colour affects the price of a car.

```
# Visualising how colours affect the price of a car
grouped_data3 = adv.groupby('standard_colour')['price'].mean().reset_index()
plt.figure(figsize=(15, 5)) # Setting graph size
sns.barplot(x='standard_colour', y='price', data=grouped_data3)
# Setting graph title and x and y labels
plt.title('Average Price Comparison Across Different colours')
plt.xlabel('Standard Colour')
plt.ylabel('Average Price')
plt.xticks(rotation=90)
plt.show() # To display the graph
```

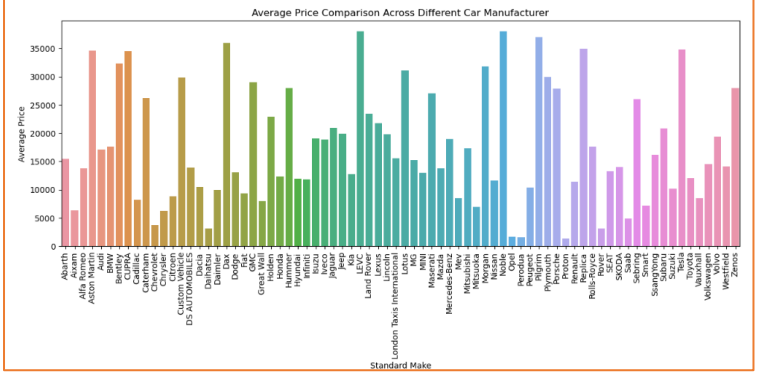


The graph above is useful for quickly identifying which colours are more expensive on average. According to the graph, indigo is the most expensive among the others.

3.2.2.2 Standard Make Vs. Price

The purpose of this graph is to compare the average prices of cars from various manufacturers.

```python
# Grouping standard_make with price to understand average price of each manufacturer
grouped_data4 = adv.groupby('standard_make')['price'].mean().reset_index()
plt.figure(figsize=(15, 5)) # Setting graph size
sns.barplot(x='standard_make', y='price', data=grouped_data4)
# Setting graph title and x and y labels
plt.title('Average Price Comparison Across Different Car Manufacturer')
plt.xlabel('Standard Make')
plt.ylabel('Average Price')
plt.xticks(rotation=90)
plt.show() # To display the graph
```



The graph represents the average prices of cars from various manufacturers. We can see that the average price of LEVC, Noble, Pilgrim and Dax are higher compare to others.

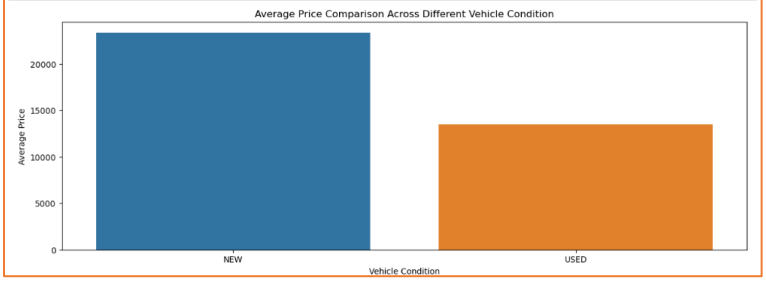
3.2.2.3 Vehicle Condition Vs. Price

This comparison is to identify how vehicle condition helps to predict the price of a vehicle.

```python
# Grouping vehicle condition by average price to compare the difference of used car and new car price
grouped_data5 = adv.groupby('vehicle_condition')['price'].mean().reset_index()

plt.figure(figsize=(15, 5)) # Setting graph size
sns.barplot(x='vehicle_condition', y='price', data=grouped_data5)

# Setting graph title and x and y labels
plt.title('Average Price Comparison Across Different Vehicle Condition')
plt.xlabel('Vehicle Condition')
plt.ylabel('Average Price')
plt.show() # To display the graph
```

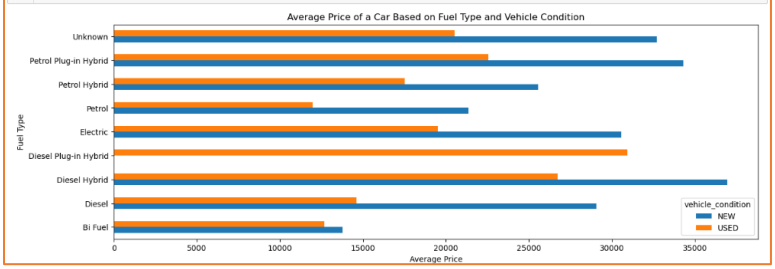


The graph clearly shows that the average price of new cars is nearly 50% higher than that of used cars. So, used cars are 50% less expensive than new cars.

3.2.2.4 Fuel Type Vs. Price based on Vehicle condition

The purpose of this comparison is to identify how fuel type helps to decide the price of a car.

```python
# Creating a grouping of fuel type with average price based on vehicle condition
grouped_data6 = adv.groupby(['fuel_type', 'vehicle_condition'])['price'].mean().reset_index()
pivot_adv = grouped_data6.pivot(index='fuel_type', columns='vehicle_condition', values='price')
pivot_adv.plot(kind='barh', stacked=False, figsize=(15, 5)) # Setting graph size
# Setting graph title and x and y Labels
plt.title('Average Price of a Car Based on Fuel Type and Vehicle Condition')
plt.xlabel('Average Price')
plt.ylabel('Fuel Type')
plt.show()  # To display the graph
```



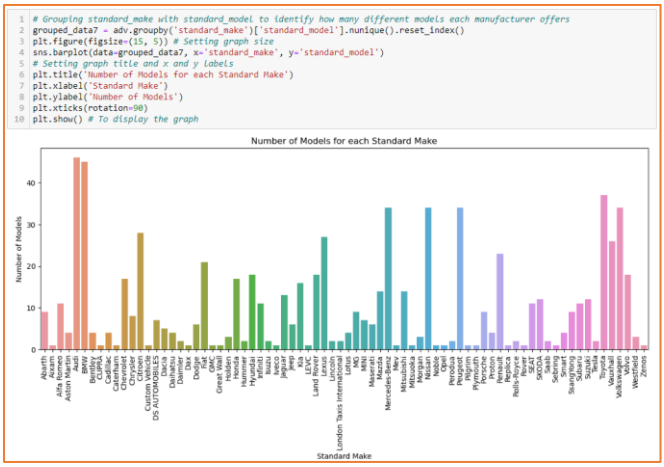
From the graph, we can say that for all fuel types, the blue bars (new cars) are longer than the orange bars (used cars). This implies that, on average, new cars are more expensive than used cars for every fuel type.

3.2.3 Categorical -Categorical

3.2.3.1 Standard Make Vs. Standard Model

The comparison of standard makes and the standard model is to identify which manufacturer offers more options for different models.

```
# Grouping standard_make with standard_model to identify how many different models each manufacturer offers
grouped_data7 = adv.groupby('standard_make')['standard_model'].nunique().reset_index()
plt.figure(figsize=(15, 5)) # Setting graph size
sns.barplot(data=grouped_data7, x='standard_make', y='standard_model')
# Setting graph title and x and y Labels
plt.title('Number of Models for each Standard Make')
plt.xlabel('Standard Make')
plt.ylabel('Number of Models')
plt.xticks(rotation=90)
plt.show() # To display the graph
```

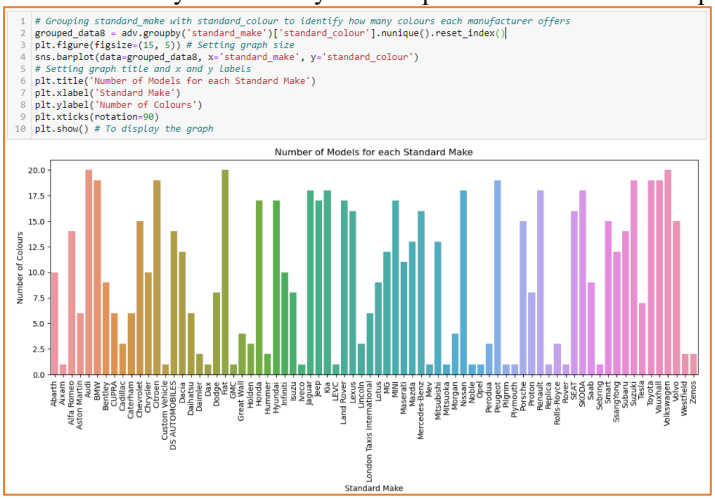


The graph shows that Standard make Audi offers more models than other manufacturers, followed by BMW.

3.2.3.2 Standard Make Vs. Standard Colour

The purpose of this comparison is to analyse how many colour options each manufacturer provides.

```
# Grouping standard_make with standard_colour to identify how many colours each manufacturer offers
grouped_data8 = adv.groupby('standard_make')['standard_colour'].nunique().reset_index()
plt.figure(figsize=(15, 5)) # Setting graph size
sns.barplot(data=grouped_data8, x='standard_make', y='standard_colour')
# Setting graph title and x and y labels
plt.title('Number of Models for each Standard Make')
plt.xlabel('Standard Make')
plt.ylabel('Number of Colours')
plt.xticks(rotation=90)
plt.show() # To display the graph
```



The chart helps to compare how many colour options are offered by each car manufacturer at a glance. As per the graph, some manufacturers have a wide range of colour options, as shown by taller bars, while others offer fewer choices, indicated by shorter bars.

4. Key Findings

1. The correlation matrix indicates that the 'year of registration' is really important for figuring out how much a car is worth. This is followed by the vehicle's age and its condition (NEW/USED). It makes sense that newer cars, or those registered more recently, cost more. As cars get older, they usually drop in price.
2. Fuel type is another key thing that affects a car's price. From the data, it looks like the average price for every type of fuel is going up over time. Maybe because of technological advancements or rising inflation.
3. When we look at how mileage affects price, there's a clear trend, the more a car is driven (and the higher its mileage), the less it tends to be worth. This is probably because more use means more wear and tear on the car.

In short, a bunch of things play into how much a car is worth, year of registration/vehicle age, vehicle condition, and mileage of the car are some of the biggest factors.