Manchester Metropolitan University

6G7V0026 Principles of Data Science

Student ID: 22547202
Name: Neha Alam Hussain

Contents

1. Data Understanding and Exploration

Overview

The dataset that we have been provided to work on is called **Car Sales Adverts** by AutoTrader. The dataset (adverts.csv) has 402,005 rows and 12 columns. The dataset has a total of 12 features. This dataset tells us about the characteristics of the cars that are up for sale and their prices.

The features that we can find in this dataset are **public_reference**, **mileage**, **reg_code**, **standard_colour**, **standard_make**, **standard_model**, **vehicle_condition**, **year_of_registration**, **price**, **body_type**, **crossover_car_and_van**, and **fuel_type.**

```
import pandas as pd
dataset = pd.read_csv('https://raw.githubusercontent.com/nehahussain/Ds_ML_dataset/main/adverts.csv')

print('Total number of columns: ' + str(dataset.shape[1]) + "\n" + "Total number of rows: " + str(dataset.shape[0]))
print(dataset.info())
```

```
Total number of columns: 12
Total number of rows: 402005
<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
None
```

The features in the dataset provide us with details about the vehicles that are up for sale.

Features in the dataset

i. public_reference

This is a 15-digit integer value in the dataset. Every record has a unique value. This value could be the ad reference value that allows to uniquely identify each advertisement at AutoTrader.

```
In [12]: # this code statement helps us to find if there are any duplicate values in the column
         pd.Series(dataset['public_reference']).is_unique

Out[12]: True
```

ii. mileage

This is a float value that tells us the number of miles that the car has been driven. It is a continuous variable. New and used cars both can have zero miles according to the dataset. This is an important feature in the dataset because miles clocked on a vehicle has an effect on the price.

```
In [36]: new = dataset[dataset['mileage'] < 1]
         new['vehicle_condition'].value_counts()

Out[36]: NEW      15852
         USED       355
         Name: vehicle_condition, dtype: int64
```

The highest mileage in the dataset is 999,999 miles. The record with the highest mileage looks rather erroneous because the price of the car is 9999 as well.

```
In [39]: df = dataset.sort_values('mileage', ascending=False)
         df.head(1)

Out[39]:
```

	public_reference	mileage	reg_code	standard_colour	standard_make	standard_model	vehicle_condition	year_of_registration	price	body_type
43974	202008122406434	999999.0	63	White	Vauxhall	Astra	USED	2013.0	9999	Hatchback

These are the top five records with the highest mileage in the dataset:

```
df = dataset.sort_values('mileage', ascending=False)
df.head(5)
```

	public_reference	mileage	reg_code	standard_colour	standard_make	standard_model	vehicle_condition	year_of_registration	price	body_type	cros
43974	202008122406434	999999.0	63	White	Vauxhall	Astra	USED	2013.0	9999	Hatchback	
63569	202010235350884	990000.0	07	Red	Vauxhall	Zafira	USED	2007.0	1200	MPV	
85043	202010285542410	930000.0	56	Blue	BMW	1 Series	USED	2006.0	2500	Hatchback	
131508	202010064664710	788072.0	68	Blue	MINI	Countryman	USED	2018.0	22875	SUV	
115277	202010215262290	740000.0	14	Blue	Volvo	XC90	USED	2014.0	14795	SUV	

Cars that were new had no miles clocked on them. Hence, the lowest mileage is zero in the dataset.

```
df = dataset.sort_values('mileage')
df.head(5)
```

	public_reference	mileage	reg_code	standard_colour	standard_make	standard_model	vehicle_condition	year_of_registration	price	body_type	cros
0	202006039777689	0.0	NaN	Grey	Volvo	XC90	NEW	NaN	73970	SUV	
376481	202009163819142	0.0	NaN	Black	Volkswagen	Polo	NEW	NaN	17401	Hatchback	
11081	202007101107807	0.0	NaN	Black	Volvo	XC60	NEW	NaN	57645	SUV	
11082	202007221536420	0.0	NaN	White	Vauxhall	Grandland X	NEW	NaN	45995	SUV	
32906	202003278771163	0.0	NaN	White	Renault	Megane	NEW	NaN	63500	Hatchback	

The mean and standard deviation can be seen below:

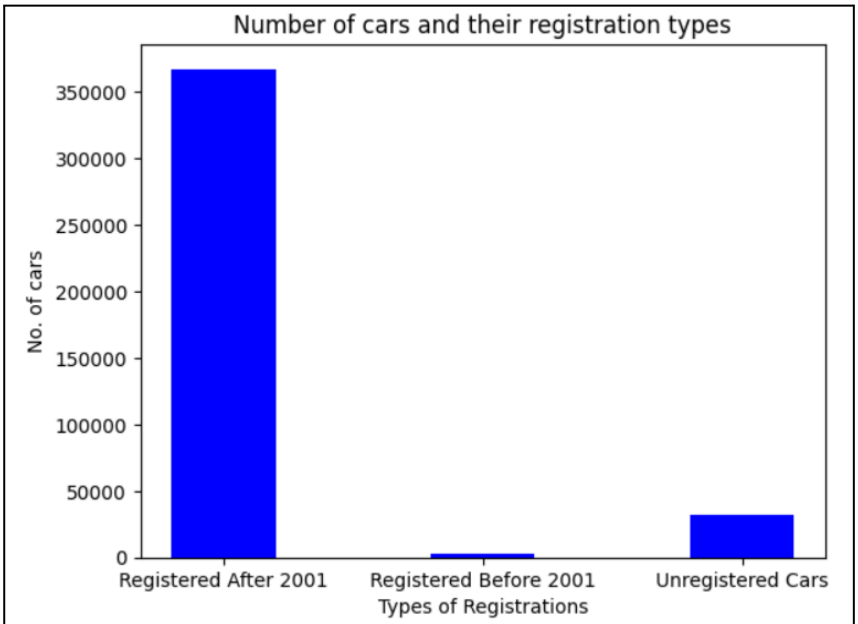
```
df_mean = np.mean(dataset["mileage"])
df_std = np.std(df["mileage"])
print("Mean: " + str(round(df_mean, 2)))
print("Standard Deviation: " + str(round(df_std, 2)))

Mean: 37743.6
Standard Deviation: 34831.68
```

iii. reg_code

This feature tells us about the age of the vehicle. Instead of the entire registration code of the vehicle, only the age identifier part of the code is available in this feature. There are two types of entries that can be found in this feature: **two-digit** integers and **single** alphabets. Vehicles registered from 1963 to 2001 had their age identifier in the alphabets. After 2001, the age identifier was changed to double digits.

The distribution of the data can be seen below:

```
import matplotlib.pyplot as plt

df_not_null = dataset[dataset['reg_code'].notnull() == True]
df_null = dataset[dataset['reg_code'].isna() == True]
df_alpha = df_not_null[df_not_null['reg_code'].str.isdigit() == False]
df_digits = df_not_null[df_not_null['reg_code'].str.isdigit() == True]

x_label = ['Registered After 2001', 'Registered Before 2001', 'Unregistered Cars']
y_label = [df_digits["reg_code"].count(), df_alpha["reg_code"].count(), df_null['reg_code'].isna().count()]
plt.bar(x_label, y_label, color ='blue',
        width = 0.4)
plt.xlabel("Types of Registrations")
plt.ylabel("No. of cars")
plt.title("Number of cars and their registration types")
plt.show()
print("Total entries: " + str(dataset.shape[0]))
print("Unregistered cars: " + str(df_null['reg_code'].isna().count()))
print("Cars registered after 2001: " + str(df_digits["reg_code"].count()))
print("Cars registered before 2001: " + str(df_alpha["reg_code"].count()))
```



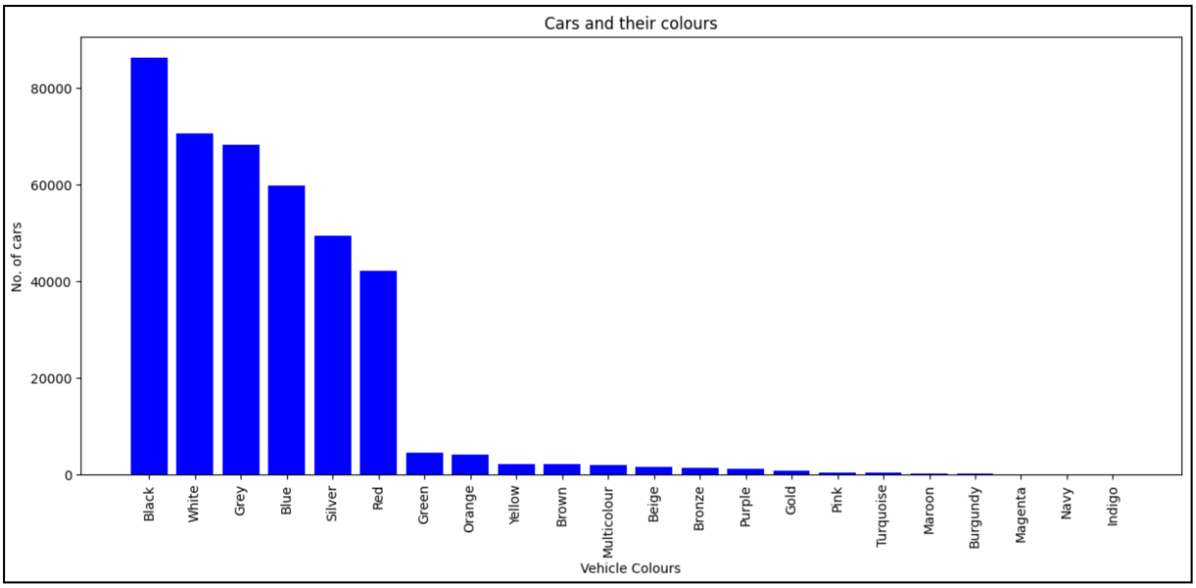

```
Total entries: 402005
Unregistered cars: 31857
Cars registered after 2001: 367233
Cars registered before 2001: 2915
```

New cars that have not been registered yet have **null** values in the dataset. The age of the vehicle has an effect on the price of the car.

iv. standard_colour

This feature tells us about the color of the vehicle. The following bar plot will tell about the distribution:

```
import matplotlib.pyplot as plt

vc = dataset['standard_colour'].value_counts()
vc.index = vc.index.astype(str)
vc_dict = vc.to_dict()
x_label = []
y_label = []
for key in vc_dict.keys():
    x_label.append(key)
    y_label.append(vc_dict[key])

plt.figure(figsize=(15, 6))
plt.bar(x_label, y_label, color ='blue',
        width = 0.8)
ax = plt.gca()
ax.set_xticklabels(labels=x_label,rotation=90);
plt.xlabel("Vehicle Colours")
plt.ylabel("No. of cars")
plt.title("Cars and their colours")
plt.show()
```



There are **5378** vehicles with no color mentioned. These values are represented as empty strings in the dataset.

```
dataset['standard_colour'].isnull().sum()

5378
```

```
dataset['standard_colour'].value_counts()

Black          86287
White          70535
Grey           68227
Blue           59784
Silver         49323
Red            42024
Green           4534
Orange          4088
Yellow          2097
Brown           2014
Multicolour     1854
Beige           1539
Bronze          1330
Purple          1211
Gold             818
Pink             410
Turquoise        307
Maroon           159
Burgundy          63
Magenta           15
Navy               7
Indigo             1
Name: standard_colour, dtype: int64
```

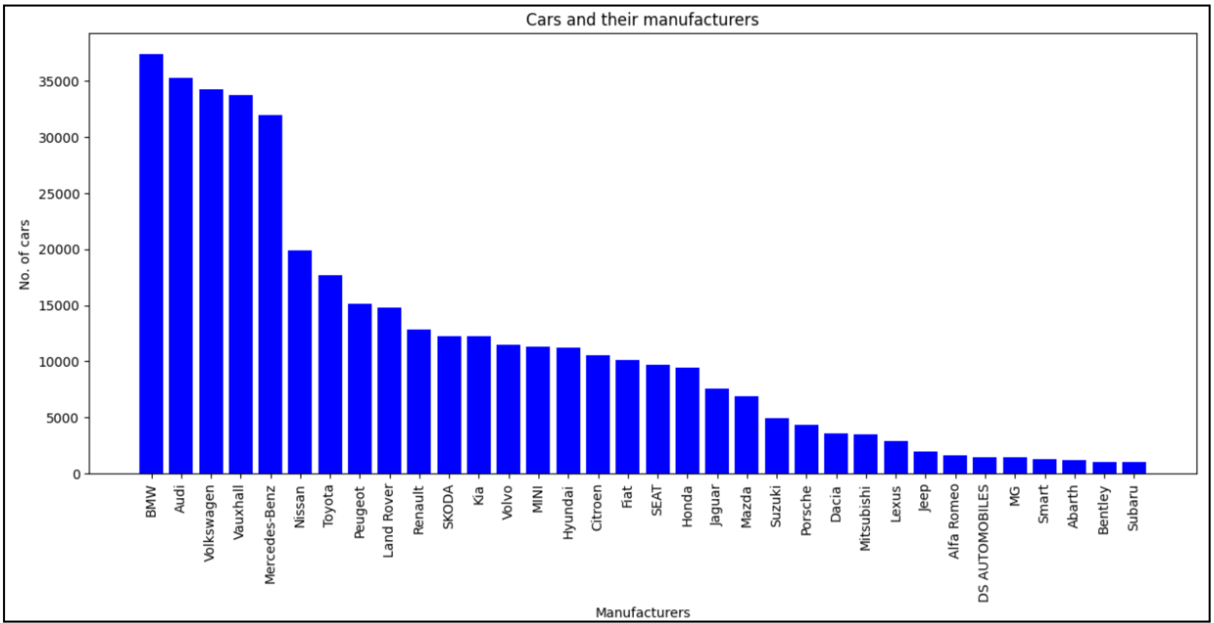
v. standard_make

This feature defines the manufacturer of the vehicle. It is a categorical variable. There are a total of 110 unique manufacturers in the dataset. German automakers BMW, Audi and Volkswagen are the manufacturers of most vehicles up for sale according to the dataset.

Due to a large number of different manufacturers, only those manufacturers are included in the figure below that have more than 1000 vehicles in the entire dataset.

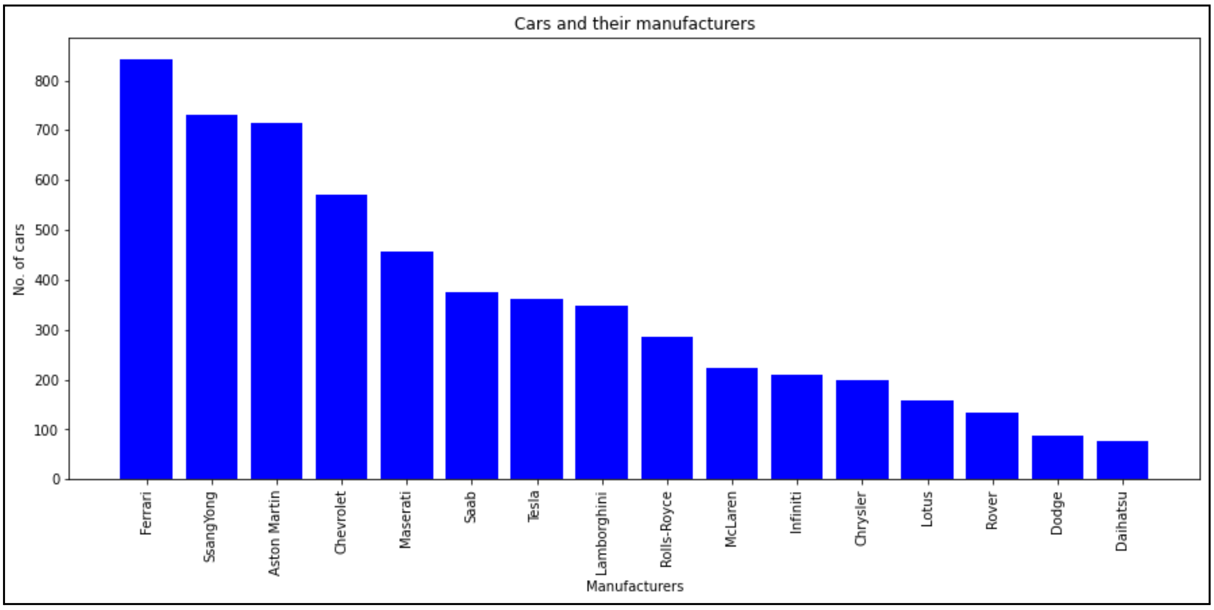
```
import matplotlib.pyplot as plt

vc = dataset['standard_make'].value_counts()
print("No. of manufacturers: " + str(len(vc)))
vc.index = vc.index.astype(str)
vc_dict = vc.to_dict()
x_label = []
y_label = []
for key in vc_dict.keys():
    if vc_dict[key] > 1000:
        x_label.append(key)
        y_label.append(vc_dict[key])

plt.figure(figsize=(15, 6))
plt.bar(x_label, y_label, color ='blue',
        width = 0.8)
ax = plt.gca()
ax.set_xticklabels(labels=x_label,rotation=90);
plt.xlabel("Manufacturers")
plt.ylabel("No. of cars")
plt.title("Cars and their manufacturers")
plt.show()
```

```
No. of manufacturers: 110
```



Luxury automakers such as Ferrari, Bugatti, and Pagani are also part of the dataset. Brands that make luxury cars tend to have higher prices.

```
x_label = []
y_label = []
for key in vc_dict.keys():
    if vc_dict[key] < 1000 and vc_dict[key] >50 :
        x_label.append(key)
        y_label.append(vc_dict[key])

plt.figure(figsize=(15, 6))
plt.bar(x_label, y_label, color ='blue',
        width = 0.8)
ax = plt.gca()
ax.set_xticklabels(labels=x_label,rotation=90);
plt.xlabel("Manufacturers")
plt.ylabel("No. of cars")
plt.title("Cars and their manufacturers")
plt.show()
```



vi. standard_model

This feature is closely related to the feature **standard_make**. It is a categorical feature. The model of the vehicle depends on the manufacturer. Volkswagen Golf has the most listings in the dataset.

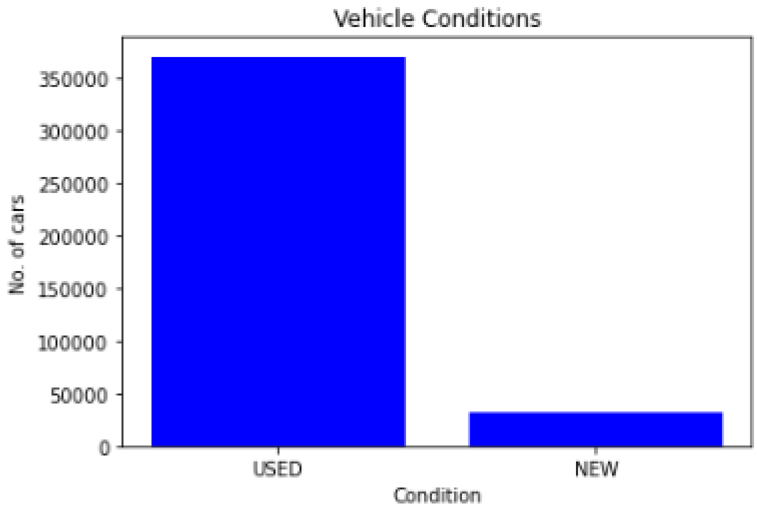
```
dataset['standard_model'].value_counts()
dataset.groupby(['standard_make','standard_model'])
group_size = dataset.groupby(['standard_make','standard_model']).size()
group_size.sort_values(ascending=False)
```

```
standard_make  standard_model
Volkswagen     Golf              11583
Vauxhall       Corsa             10646
Mercedes-Benz  C Class            8550
BMW            3 Series           8347
Volkswagen     Polo               7681
                                 ...
Toyota         Mark II Blitz         1
GMC            Pickup                1
Fiat           Uno                   1
Toyota         Paseo                 1
Porsche        917                   1
Length: 1217, dtype: int64
```

vii. vehicle_condition

This is a categorical feature. It has only two possible values: **new** and **used**. This feature has an effect on the price of the vehicle. There are 370,756 used and 31,249 new vehicles.

```
vc = dataset['vehicle_condition'].value_counts()
vc.index = vc.index.astype(str)
vc_dict = vc.to_dict()
x_label = []
y_label = []
for key in vc_dict.keys():
    x_label.append(key)
    y_label.append(vc_dict[key])

plt.bar(x_label, y_label, color ='blue',
        width = 0.8)
plt.xlabel("Condition")
plt.ylabel("No. of cars")
plt.title("Vehicle Conditions")
plt.show()
```




```
USED     370756
NEW       31249
Name: vehicle_condition, dtype: int64
```

viii. year_of_registration

This is a categorical variable that tells us the year in which the vehicle was registered. It can be used to identify the age of the vehicle as well. This feature can play a role in the price of vehicles. New vehicles do not have a year of registration value. Hence, there are missing values for new vehicles.

```
print("No. of records with no year of registration: " + str(dataset[(dataset['year_of_registration'].isna() == True)].shape[0]))
No. of records with no year of registration: 33311
```

Moreover, there are erroneous values in this column. For example, there are records where the year of registration column has values like 1015, 1515 etc. For example:

```
vc = dataset['year_of_registration'].value_counts()
dataset[dataset['year_of_registration'] < 1900]
```

	public_reference	mileage	reg_code	standard_colour	standard_make	standard_model	vehicle_condition	year_of_registration	price	body_type
59010	202006270588110	14000.0	07	Blue	Toyota	Prius	USED	1007.0	7000	Hatchback
69516	202010155035879	96659.0	65	Black	Audi	A4 Avant	USED	1515.0	10385	Estate
84501	202009163810376	37771.0	63	Black	Smart	fortwo	USED	1063.0	4785	Coupe
114737	202008102305925	30000.0	59	Red	Toyota	AYGO	USED	1009.0	4695	Hatchback
120858	202010064654489	27200.0	66	Black	MINI	Clubman	USED	1016.0	18990	Estate
190556	202010205206488	58470.0	10	Black	Fiat	Punto Evo	USED	1010.0	3785	Hatchback
199830	202009013167637	23000.0	59	Silver	MINI	Hatch	USED	1009.0	5995	Hatchback

The first record in the example above has 1007 in the **year_of_registration** column. We can use the reg_code to find out the correct year of registration. The 07 in the reg_code column means that the correct year of registration is **2007**. The same method can be used to find the correct year of registration for other entries as well.

Most vehicles were registered in the following years (according to the dataset):

```
vc = dataset['year_of_registration'].value_counts()
vc.sort_values(ascending=False)
```

```
2017.0    68790
2016.0    43483
2019.0    39236
2018.0    38300
2015.0    29019
          ...
1018.0        1
1006.0        1
1008.0        1
1515.0        1
1015.0        1
Name: year_of_registration, Length: 84, dtype: int64
```

The top 10 years when vehicles were registered the most.

```
vc.head(10)         #Top 10years when vehicles were registered the most
```

```
2017.0    68790
2016.0    43483
2019.0    39236
2018.0    38300
2015.0    29019
2020.0    28683
2014.0    23666
2013.0    19117
2012.0    15312
2011.0    12614
Name: year_of_registration, dtype: int64
```

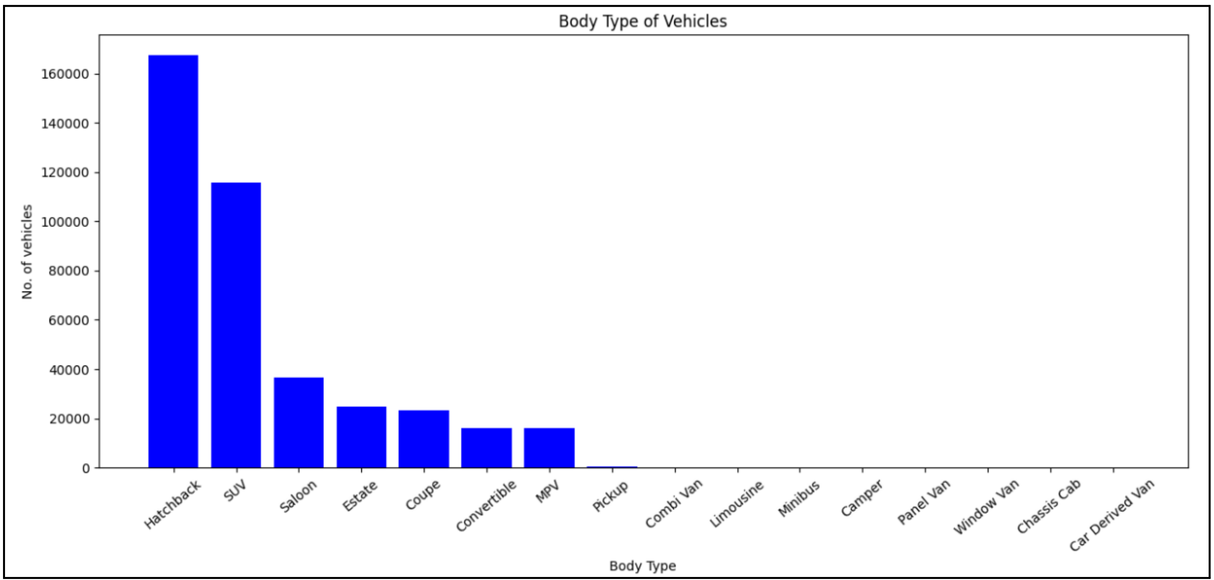
ix. body_type

This is another categorical variable. It tells us about the body type of the vehicle. The body type gives us an idea about the size of the vehicle. For example, hatchbacks are smaller cars in comparison to SUV or saloons. Vehicles with smaller body types are cheaper. We have some missing values in this column. These missing values can be attributed to the fact that they were not added when the advertisement was created.

```
dataset['body_type'].value_counts()
missing_val = dataset[dataset['body_type'].isna() == True]
print("Vehicles with missing body type: {}".format(missing_val.shape[0]))
print(dataset['body_type'].value_counts())
```

```
Vehicles with missing body type: 837
Hatchback          167315
SUV                115872
Saloon              36641
Estate              24692
Coupe               23258
Convertible         16038
MPV                 16026
Pickup                620
Combi Van             214
Limousine             159
Minibus               149
Camper                 77
Panel Van              61
Window Van             41
Chassis Cab             3
Car Derived Van         2
Name: body_type, dtype: int64
```

```
vc = dataset['body_type'].value_counts()
vc.index = vc.index.astype(str)
vc_dict = vc.to_dict()
x_label = []
y_label = []
for key in vc_dict.keys():
    x_label.append(key)
    y_label.append(vc_dict[key])

plt.figure(figsize=(15, 6))
ax = plt.gca()
ax.set_xticklabels(labels=x_label,rotation=40);
plt.bar(x_label, y_label, color ='blue',
        width = 0.8)
plt.xlabel("Body Type")
plt.ylabel("No. of vehicles")
plt.title("Body Type of Vehicles")
plt.show()
```

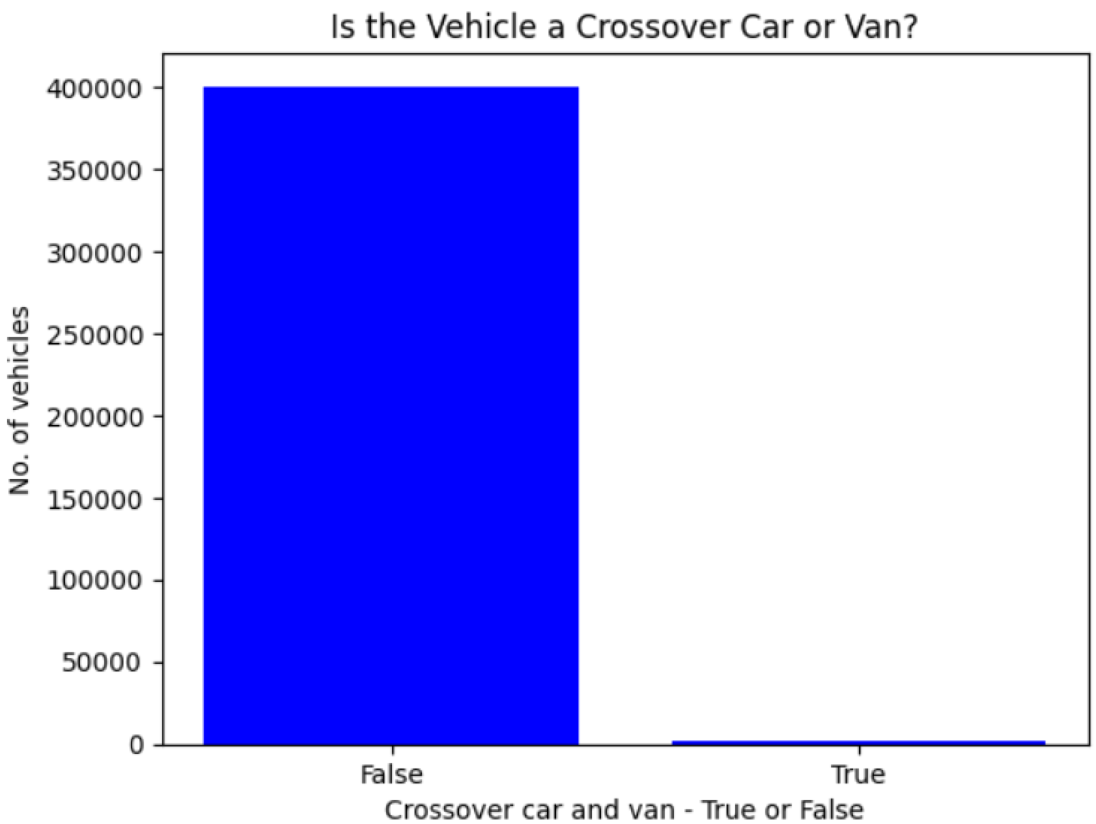


x. crossover_car_and_van

This feature is a categorical variable. It populates only two types of values: True and False. There are no missing values in the column. This feature informs whether the vehicle up for sale is a crossover or a van. A crossover is a type of vehicle which is a hybrid of a hatchback and an SUV. It is smaller than an SUV but bigger than a hatchback.

```
print(f"No. of missing values in crossover_car_and_van column: {dataset['crossover_car_and_van'].isnull().sum()}")
print(dataset['crossover_car_and_van'].value_counts())
x_label = ["False", "True"]
y_label = [(~dataset['crossover_car_and_van']).values.sum(), dataset['crossover_car_and_van'].values.sum()]
plt.bar(x_label, y_label, color ='blue',
        width = 0.8)
plt.xlabel("Crossover car and van - True or False")
plt.ylabel("No. of vehicles")
plt.title("Is the Vehicle a Crossover Car or Van?")
plt.show()
```

```
No. of missing values in crossover_car_and_van column: 0
False    400210
True       1795
Name: crossover_car_and_van, dtype: int64
```



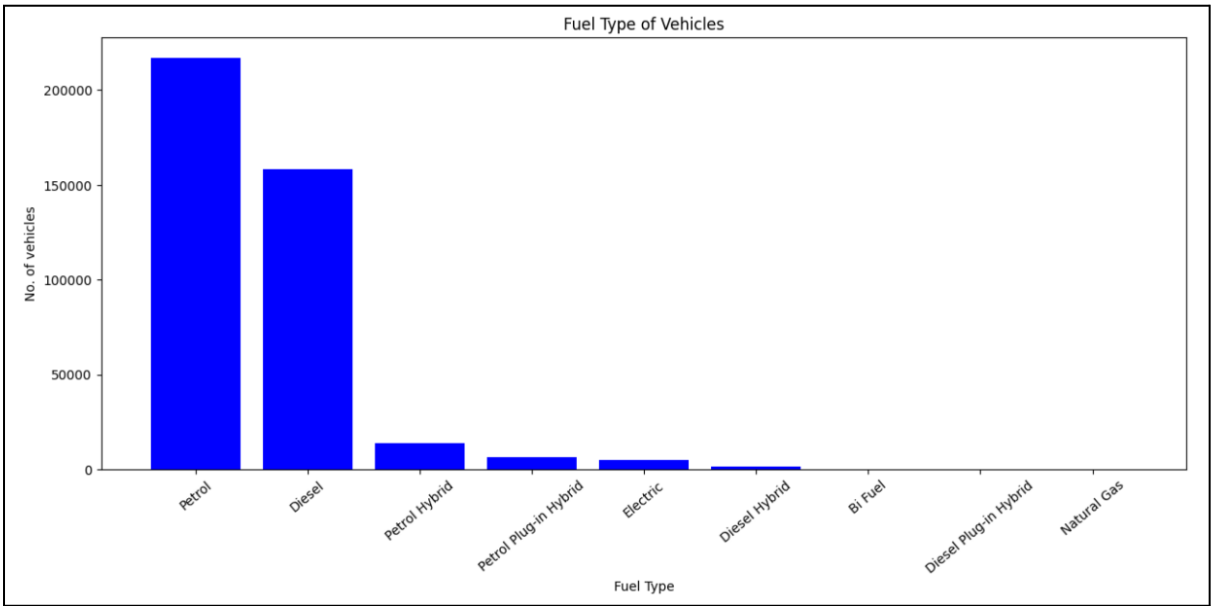
xi. fuel_type

This feature is categorical and informs about the type of fuel that the vehicle uses to run. There are different fuel type options available in the dataset. Modern fuel options such as hybrids can fetch higher prices for a vehicle in comparison to a conventional petrol engine.

There are some missing values in the dataset for this particular feature as well. These are defined as empty strings in the dataset. We have one outlier in this case. There is only one vehicle up for sale that uses natural gas as its fuel type. Most vehicles use either petrol or diesel. More in-depth analysis can be found below:

```
print(dataset['fuel_type'].value_counts())
print(f"No. of missing values in fuel_type column: {dataset['fuel_type'].isnull().sum()}")
vc = dataset['fuel_type'].value_counts()
vc.index = vc.index.astype(str)
vc_dict = vc.to_dict()
x_label = []
y_label = []
for key in vc_dict.keys():
    x_label.append(key)
    y_label.append(vc_dict[key])

plt.figure(figsize=(15, 6))
ax = plt.gca()
ax.set_xticklabels(labels=x_label,rotation=40);
plt.bar(x_label, y_label, color ='blue',
        width = 0.8)
plt.xlabel("Fuel Type")
plt.ylabel("No. of vehicles")
plt.title("Fuel Type of Vehicles")
plt.show()
```

```
Petrol                  216929
Diesel                  158120
Petrol Hybrid            13602
Petrol Plug-in Hybrid     6160
Electric                  4783
Diesel Hybrid             1403
Bi Fuel                    221
Diesel Plug-in Hybrid      185
Natural Gas                  1
Name: fuel_type, dtype: int64
No. of missing values in fuel_type column: 601
```



xii. price

This is a quantitative variable in the dataset. This feature determines the value of the vehicle that is up for sale. It is the target variable in the dataset. The other features that we have looked at up till now play their role in affecting this particular feature of the vehicle. For example, a new sports car from a luxury brand will have a higher price than a small, used hatchback.

There are no missing values in this column. Basic stat results of raw data show that there are possibly incorrect values in the dataset. These were the findings:

```
mode_val = dataset['price'].mode()
print(f"Mean price of vehicle: {dataset['price'].mean():.2f}")
print(f"Median price of vehicle: {dataset['price'].median():.2f}")
print(f"Mode price of vehicle: {mode_val[0]:.2f}")
print(f"Standard Deviation in price of vehicle: {dataset['price'].std():.2f}")
print(f"Max price of vehicle: {dataset['price'].max():.2f}")
print(f"Min price of vehicle: {dataset['price'].min():.2f}")
```

```
Mean price of vehicle: 17341.97
Median price of vehicle: 12600.00
Mode price of vehicle: 8995.00
Standard Deviation in price of vehicle: 46437.46
Max price of vehicle: 9999999.00
Min price of vehicle: 120.00
```

The maximum value in the column looks like an incorrect entry. The same could be said about the minimum price of the vehicle. These values are too high and too low to be considered true. Hence, it is necessary to analyze them to make sure that these entries are correct.

Deep diving into the anomaly helped us to figure out that these entries are the result of incomplete or incorrect data entry. These entries are missing **reg_code** and **year_of_registration** values as well. Registration code is unique to every vehicle and is assigned to each vehicle when it is brought on the road. These cars are used, hence, this clears that these cars must have had a registration code but the data is missing.

```
temp = dataset[dataset['price'] == 9999999]
print(f"Count of vehicles: {temp['price'].count()}")
temp
```

```
Count of vehicles: 6
```

	public_reference	mileage	reg_code	standard_colour	standard_make	standard_model	vehicle_condition	year_of_registration	price	body_type	cr
141833	202007050883898	87450.0	NaN	Red	Ferrari	250	USED	NaN	9999999	Coupe	
147188	202009103539048	100.0	NaN	Grey	Maserati	3500	USED	NaN	9999999	Convertible	
252505	202008112331147	46300.0	NaN	Blue	Ferrari	275	USED	NaN	9999999	Coupe	
305436	201812223434109	3600.0	NaN	Grey	Lamborghini	Miura	USED	NaN	9999999	Coupe	
336202	202001226429470	950.0	NaN	Black	Ferrari	LaFerrari	USED	NaN	9999999	Coupe	
336536	202006180262926	4400.0	NaN	White	Porsche	959	USED	NaN	9999999	Coupe	

We decided to look further into records that had one of the lowest prices in the dataset. These entries seem to be correct, and we can keep them.

```
res = dataset.sort_values(by='price', ascending=False)
res.tail()
```

	public_reference	mileage	reg_code	standard_colour	standard_make	standard_model	vehicle_condition	year_of_registration	price	body_type	cros:
91878	202009083473638	100000.0	56	Silver	Renault	Clio	USED	2006.0	200	Hatchback	
109133	202010295564975	117500.0	52	Green	Citroen	C3	USED	2002.0	180	Hatchback	
303316	202010195173556	159000.0	X	Bronze	Honda	HR-V	USED	2000.0	150	SUV	
300445	202011015671489	89000.0	W	Green	Vauxhall	Corsa	USED	2000.0	122	Hatchback	
332532	202010315653263	78000.0	W	Blue	Citroen	Saxo	USED	2000.0	120	Hatchback	

The 9,999,999 value in the price column affects the mean and standard deviation. After omitting the six rows that had 9,999,999 price, this is what the statistics look like now:

```
df = dataset[dataset['price'] < 9999999]
temp = df.sort_values(by='price', ascending=False)
mode_val = temp['price'].mode()
print(f"Mean price of vehicle: {temp['price'].mean():.2f}")
print(f"Median price of vehicle: {temp['price'].median():.2f}")
print(f"Mode price of vehicle: {mode_val[0]:.2f}")
print(f"Standard Deviation in price of vehicle: {temp['price'].std():.2f}")
print(f"Max price of vehicle: {temp['price'].max():.2f}")
print(f"Min price of vehicle: {temp['price'].min():.2f}")
```

```
Mean price of vehicle: 17192.97
Median price of vehicle: 12600.00
Mode price of vehicle: 8995.00
Standard Deviation in price of vehicle: 25866.50
Max price of vehicle: 3799995.00
Min price of vehicle: 120.00
```

There is a big difference in the standard deviation caused due to those anomalous entries. The mean was only slightly affected in comparison.

2. Data Processing

Detection of Erroneous and Missing Values

Our initial analysis of the dataset helped us to find the following issues with the data:

- 9,999,999 in the price column for six records. It was also found that the year of registration and registration code values for those six records are also missing. Details regarding the erroneous values are mentioned in the ‘price’ subheading.

- Another erroneous value was found when looking for the maximum value in the ‘mileage’ column. The highest mileage in the dataset is recorded as 999,999. While clocking 999,999 miles/km on a vehicle is possible, the record raises some concerns because the asking price of the vehicle is 9,999 and the year of registration is 2013. An in-depth look at this particular record is done in the report earlier.

- Registration code column has null values. Null values are mostly for vehicles that are new. New vehicles for sale do not have a registration code assigned to them yet. However, we did find records where the vehicle condition was used but there was no registration code mentioned. There were null values instead. There are a total of **608** records of this type.

```
dataset[(dataset['reg_code'].isna() == True) & (dataset['vehicle_condition'] == 'USED')]
```

	public_reference	mileage	reg_code	standard_colour	standard_make	standard_model	vehicle_condition	year_of_registration	price	body_type
630	202010275479166	54000.0	NaN	White	Toyota	Prius+	USED	2019.0	10900	MPV
682	202006019703585	103450.0	NaN	Bronze	BMW	3 Series	USED	2006.0	6000	Convertible
1131	202009274274693	74000.0	NaN	Silver	Mercedes-Benz	SL Class	USED	2004.0	15950	Convertible
1194	202010114878094	13000.0	NaN	White	Ferrari	599	USED	2017.0	79949	Coupe
1335	202004018824812	76000.0	NaN	White	Aston Martin	DB4	USED	2018.0	495000	Saloon
...	...	...	...	...	...	...	...	...	...	...
397788	202009214000713	119400.0	NaN	White	Toyota	Prius	USED	2017.0	8995	Hatchback
397947	202010265437718	147898.0	NaN	Blue	Toyota	Prado	USED	2005.0	3250	Estate
398865	202010074728813	74500.0	NaN	Silver	Toyota	Sienta	USED	2020.0	4250	MPV
399728	202008262969804	12812.0	NaN	Silver	Rolls-Royce	Wraith	USED	NaN	159950	Coupe
400536	202010094805399	40523.0	NaN	Red	Peugeot	108	USED	NaN	5999	Hatchback

608 rows × 12 columns

- There are **5,378** records that have standard_colour missing.

```
dataset[dataset['standard_colour'].isna() == True].shape[0]
```

5378

- Year of registration is assigned to a vehicle as soon as it is brought onto the road. New vehicles for sale do not have a year of registration value in the dataset. They are missing in that case but due to logical reasons. However, there are **2,062** records that do not have a year of registration mentioned for them even though their condition is marked as ‘USED’ in the advertisement. To fix these missing values, we can use the registration code column to figure out the year of registration value. This will only be possible in cases where the registration code column is populated. There are **321** used vehicles in the dataset that do not have registration codes and years of registration added in their respective fields.

```
dataset[(dataset['year_of_registration'].isna() == True) & (dataset['vehicle_condition'] == 'USED')]
```

	public_reference	mileage	reg_code	standard_colour	standard_make	standard_model	vehicle_condition	year_of_registration	price	body_type
54	202007030806426	30000.0	18	Red	Vauxhall	Insignia	USED	NaN	11990	Hatchback
83	202008222801747	42847.0	61	Red	Honda	Jazz	USED	NaN	5695	Hatchback
865	202010084741550	43130.0	66	White	Land Rover	Range Rover Sport	USED	NaN	35990	SUV
968	202010305607535	63369.0	17	Blue	SKODA	Rapid Spaceback	USED	NaN	7490	Hatchback
1256	202010225294466	18715.0	68	White	Volvo	V40	USED	NaN	16950	NaN
...	...	...	...	...	...	...	...	...	...	...
400536	202010094805399	40523.0	NaN	Red	Peugeot	108	USED	NaN	5999	Hatchback
400725	202008252905656	45.0	20	Silver	Land Rover	Range Rover Evoque	USED	NaN	46945	SUV
401314	202010315635541	12522.0	67	Red	Dacia	Sandero	USED	NaN	6300	Hatchback
401323	201909222504136	46000.0	13	Grey	Volkswagen	Caravelle	USED	NaN	22995	MPV
401357	202007111114611	10.0	20	Blue	Vauxhall	Corsa	USED	NaN	16000	Hatchback

2062 rows × 12 columns

```python
dataset[(dataset['year_of_registration'].isna() == True) & (dataset['vehicle_condition'] == 'USED') &
        (dataset['reg_code'].isna() == True)]
```

	public_reference	mileage	reg_code	standard_colour	standard_make	standard_model	vehicle_condition	year_of_registration	price	body_type
1510	202010054642656	13406.0	NaN	White	Land Rover	Range Rover Evoque	USED	NaN	26000	Coupe
2631	202010235350805	1000.0	NaN	Blue	Maserati	Levante	USED	NaN	63995	SUV
4766	202003238706011	NaN	NaN	Grey	Subaru	Outback	USED	NaN	35995	Estate
6998	202010225284269	160.0	NaN	Grey	McLaren	Senna	USED	NaN	699950	Coupe
7517	202009234093511	11413.0	NaN	NaN	MINI	Convertible	USED	NaN	14400	Convertible
...	...	...	...	...	...	...	...	...	...	...
392499	202010064681927	83555.0	NaN	Black	Land Rover	Range Rover Sport	USED	NaN	30995	SUV
392730	202009093528195	38796.0	NaN	Grey	Rover	110	USED	NaN	3150	Saloon
396985	202001256559400	29000.0	NaN	Black	Lamborghini	Gallardo	USED	NaN	77990	Coupe
399728	202008262969804	12812.0	NaN	Silver	Rolls-Royce	Wraith	USED	NaN	159950	Coupe
400536	202010094805399	40523.0	NaN	Red	Peugeot	108	USED	NaN	5999	Hatchback

321 rows × 12 columns

- There are **837** records that have null values in the body type column.

```python
dataset[dataset['body_type'].isna() == True]
```

	public_reference	mileage	reg_code	standard_colour	standard_make	standard_model	vehicle_condition	year_of_registration	price	body_type
307	202010245377951	33287.0	63	Red	Volkswagen	Caddy	USED	2013.0	10990	NaN
625	202010255419023	55000.0	65	Grey	Vauxhall	Vivaro	USED	2015.0	10995	NaN
1256	202010225294466	18715.0	68	White	Volvo	V40	USED	NaN	16950	NaN
1643	202010305596351	24920.0	17	Blue	BMW	4 Series	USED	2017.0	21980	NaN
1929	202010155047896	10.0	70	Blue	Lotus	Elise	USED	2020.0	47775	NaN
...	...	...	...	...	...	...	...	...	...	...
399677	202007111130539	5001.0	20	Black	Mercedes-Benz	GLC Class	USED	2020.0	36870	NaN
400624	202010285545599	322000.0	05	Black	London Taxis International	TXI	USED	2005.0	995	NaN
400643	202011015665976	10.0	NaN	White	Audi	A3	NEW	NaN	27845	NaN
400724	202008242879192	0.0	L	Silver	Porsche	911	USED	1973.0	175000	NaN
401788	202009103544825	67558.0	16	Red	Mercedes-Benz	220	USED	2016.0	12000	NaN

837 rows × 12 columns

- There are **601** records that have null values in their fuel_type column.

```python
print(f"{dataset[dataset['fuel_type'].isna() == True].shape[0]} records have null values in fuel_type column")
dataset[dataset['fuel_type'].isna() == True]
```

```
601 records have null values in fuel_type column
```

ileage	reg_code	standard_colour	standard_make	standard_model	vehicle_condition	year_of_registration	price	body_type	crossover_car_and_van	fuel_type
10.0	NaN	Black	BMW	5 Series	NEW	NaN	51395	Estate	False	NaN
0.0	NaN	NaN	Jaguar	XF	NEW	NaN	35990	Estate	False	NaN
10.0	NaN	Grey	Volvo	V60	NEW	NaN	31414	Estate	False	NaN
1568.0	14	Silver	Toyota	Prius	USED	2020.0	7995	Hatchback	False	NaN
10.0	NaN	Black	Mercedes-Benz	A Class	NEW	NaN	36584	Hatchback	False	NaN
...	...	...	...	...	...	...	...	...	...	...
0.0	NaN	NaN	Jaguar	XF	NEW	NaN	32585	Saloon	False	NaN
1.0	NaN	Black	Peugeot	508	NEW	NaN	39135	Hatchback	False	NaN
17.0	19	NaN	McLaren	Senna	USED	2019.0	799900	NaN	False	NaN
10.0	NaN	Black	Vauxhall	Grandland X	NEW	NaN	31190	SUV	False	NaN
1000.0	70	Silver	Mercedes-Benz	A Class	USED	2020.0	32000	Hatchback	False	NaN

- The year of registration column has some values that date back as far as 1007. This is an obvious error in data entry. On inspecting such records, it was easy to find the error. The registration code column provided the correct age and year of registration of the car. The first record in the example below has 1007 in the **year_of_registration** column. We can use the reg_code to find out the correct year of registration. The 07 means that the correct year of registration is **2007**. The same method can be used to find the correct year of registration for other entries as well.

```
vc = dataset['year_of_registration'].value_counts()
dataset[dataset['year_of_registration'] < 1900]
```

	public_reference	mileage	reg_code	standard_colour	standard_make	standard_model	vehicle_condition	year_of_registration	price	body_type
59010	202006270588110	14000.0	07	Blue	Toyota	Prius	USED	1007.0	7000	Hatchback
69516	202010155035879	96659.0	65	Black	Audi	A4 Avant	USED	1515.0	10385	Estate
84501	202009163810376	37771.0	63	Black	Smart	fortwo	USED	1063.0	4785	Coupe
114737	202008102305925	30000.0	59	Red	Toyota	AYGO	USED	1009.0	4695	Hatchback
120858	202010064654489	27200.0	66	Black	MINI	Clubman	USED	1016.0	18990	Estate
190556	202010205206488	58470.0	10	Black	Fiat	Punto Evo	USED	1010.0	3785	Hatchback
199830	202009013167637	23000.0	59	Silver	MINI	Hatch	USED	1009.0	5995	Hatchback

Handling Erroneous and Missing Values

Firstly, the six records that had 9,999,999 in the price column were removed. This was due to the erroneous price and the missing year of registration and registration code of these records.

```
# this statement drops the six records with the missing values and the erroneous price
dataset.drop(dataset[dataset['price'] == 9999999].index, inplace=True)
```

To handle missing values in our dataset, we will be using mean, median and mode to fill them. The decision to use either mean, median or mode will depend on the type of data and skewness. Categorical variables can be filled using the mode value of the column. If the feature is numeric and the data is not skewed, the mean can be used to fill the null values. Otherwise, we are going to use the median.
We ran some analysis to check for skewness in our **mileage** and **year_of_registration** column.

```
print("Records with null mileage value: " + str(dataset[dataset["mileage"].isna() == True].shape[0]))
print("Skewness in mileage column: " +
      str(dataset['mileage'].skew(axis=0, skipna=True).round(decimals=2)))
# As the mileage column is skewed, we will need to use median to fill the NaN (null) values
print(dataset['mileage'].median())   # mileage median

Records with null mileage value: 127
Skewness in mileage column: 1.45
28629.5
```

```
print("Records with null year_of_registration value: "
      + str(dataset[(dataset["year_of_registration"].isna() == True)
                   & (dataset["vehicle_condition"] == "USED")].shape[0]))
print("Skewness in year_of_registration column: " +
      str(dataset['year_of_registration'].skew(axis=0, skipna=True).round(decimals=2)))
# As the year_of_registration column is skewed, we will need to use median to fill the NaN (null) values
print(dataset['year_of_registration'].median())

Records with null year_of_registration value: 2062
Skewness in year_of_registration column: -87.91
2016.0
```

Moreover, there were about 17 records that had incorrect year_of_registration values (the year before the 1900s).

```
dataset[dataset["year_of_registration"]<1900]["year_of_registration"].value_counts().sum()
```

```
17
```

To fix these erroneous values, we use the reg_code column. The reg_code column has an age identifier that allows us to figure out the correct year_of_registration value. To fill the missing values in the column, a mapping was created using reg_code and information from this website [1].

The erroneous values were fixed first. Then, the missing values were filled in where the vehicle condition is used. New vehicles do not have a year of registration. Then, records that had missing reg_code and year_of_registration, in those cases, year_of_registration was filled with 2016 (median of year_of_registration).

```
dataset[ (dataset['reg_code'].isna() == False) & (dataset['year_of_registration'].isna() == True) ]
```

mileage	reg_code	standard_colour	standard_make	standard_model	vehicle_condition	year_of_registration	price	body_type	crossover_car_and_van	fuel_typ
30000.0	18	Red	Vauxhall	Insignia	USED	NaN	11990	Hatchback	False	Petr
42847.0	61	Red	Honda	Jazz	USED	NaN	5695	Hatchback	False	Petr
43130.0	66	White	Land Rover	Range Rover Sport	USED	NaN	35990	SUV	False	Dies
63369.0	17	Blue	SKODA	Rapid Spaceback	USED	NaN	7490	Hatchback	False	Dies
18715.0	68	White	Volvo	V40	USED	NaN	16950	NaN	False	Petr
...	...	...	...	...	...	...	...	...	...	
66287.0	63	Blue	Vauxhall	Astra GTC	USED	NaN	8400	Coupe	False	Petr
45.0	20	Silver	Land Rover	Range Rover Evoque	USED	NaN	46945	SUV	False	Dies
12522.0	67	Red	Dacia	Sandero	USED	NaN	6300	Hatchback	False	Petr
46000.0	13	Grey	Volkswagen	Caravelle	USED	NaN	22995	MPV	False	Dies
10.0	20	Blue	Vauxhall	Corsa	USED	NaN	16000	Hatchback	False	Petr

```
# Fixing year_of_registration column
reg_code_mapping = {
    '02': 2002,'03': 2003,'04': 2004,'05': 2005,'06': 2006,'07': 2007,'08': 2008,'09': 2009,
    '10': 2010,'11': 2011,'12': 2012,'13': 2013,'14': 2014,'15': 2015,'16': 2016,'17': 2017,'18': 2018,'19': 2019,'20': 2020,
    '51': 2001,'52': 2002, '53': 2003,'54': 2004,'55': 2005,'56': 2006,'57': 2007,'58': 2008,'59': 2009,
    '60': 2010,'61': 2011,'62': 2012,'63': 2013,'64': 2014,'65': 2015,'66': 2016,'67': 2017,'68': 2018,'69': 2019,'70': 2020
}

for index, row in dataset.iterrows():
    if row['reg_code'] in reg_code_mapping:
        dataset.at[index, 'year_of_registration'] = reg_code_mapping[row['reg_code']]
```

```
for index, row in dataset.iterrows():
    if pd.isnull(row['year_of_registration']) and row['vehicle_condition'] == 'USED':
        dataset.at[index, 'year_of_registration'] = 2016

# dataset[(dataset1['year_of_registration'].isna() == True) & (dataset['vehicle_condition'].isna() == 'USED')]
# the above statement will give empty dataframe to show no missing year_of_registration values for used vehicles
```

```
dataset[ (dataset['reg_code'].isna() == False) & (dataset['year_of_registration'].isna() == True)]  #no result, hence its filled
```

public_reference	mileage	reg_code	standard_colour	standard_make	standard_model	vehicle_condition	year_of_registration	price	body_type	crossover_ca

We have handled the year of registration by reg_code, but we can not do the vice versa as we also need the months with a year of registration to use it to fill reg_code since the reg_code age identifier, changes twice a year, on the 1st of March and September [1].

The rest of the columns are categorical variables, hence, we found out their respective modes.

```
print("Mode of standard_colour: " + str(dataset['standard_colour'].mode()[0]))
print("\nMode of body_type: " + str(dataset['body_type'].mode()[0]))
print("\nMode of fuel_type: " + str(dataset['fuel_type'].mode()[0]))

Mode of standard_colour: Black

Mode of body_type: Hatchback

Mode of fuel_type: Petrol
```

The null values were then replaced with their respective values.

```
dataset = dataset.fillna({'mileage': dataset['mileage'].median(), 'standard_colour': 'Black',
              'body_type': 'Hatchback', 'fuel_type': 'Petrol'})
```

Since we removed the six entries, we are left with 401999 rows and the below info shows that we have dealt with all the missing values of all the features. And we have also dealt with all the missing year of registration where the vehicle condition was used, and as for the new ones having no year of registration is justified so we did not fill them.

```
dataset.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 401999 entries, 0 to 402004
Data columns (total 12 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   public_reference       401999 non-null  int64
 1   mileage                401999 non-null  float64
 2   reg_code               370148 non-null  object
 3   standard_colour        401999 non-null  object
 4   standard_make          401999 non-null  object
 5   standard_model         401999 non-null  object
 6   vehicle_condition      401999 non-null  object
 7   year_of_registration   370750 non-null  float64
 8   price                  401999 non-null  int64
 9   body_type              401999 non-null  object
 10  crossover_car_and_van  401999 non-null  bool
 11  fuel_type              401999 non-null  object
dtypes: bool(1), float64(2), int64(2), object(7)
memory usage: 37.2+ MB
```

Feature Engineering

To help with predicting trends and patterns in our data, there can be new features engineered from existing features. Some new features were added to the dataset to provide better insights when looking at trends and averages of price when compared to different features.

mileage_type

This feature is created to provide low, medium, or high values depending on the mileage of the vehicle. Vehicles with low mileage are more likely to sell for a higher value. To check whether this trend exists in our dataset, this new feature was created.

Some analysis was required before a certain threshold could be set to decide whether the vehicle has low, medium, or high mileage. The feature engineering was done after the initial bit of data processing to add missing values and remove erroneous records.
The describe function on the mileage feature helps to provide a deep dive on the dispersion of the data.

```
dataset['mileage'].describe()
```

```
count    401982.000000
mean      37740.549127
std       34826.678659
min           0.000000
25%       10487.000000
50%       28630.000000
75%       56851.750000
max      999999.000000
Name: mileage, dtype: float64
```

According to the statistics above, the average mileage of the vehicles is about 38,000. The 25th quartile is about 10,500. Hence, vehicles with mileage lesser than 30,000 were termed as ‘low’. Vehicles that had mileage varying from 30,000 to 60,000 were termed as vehicles with ‘medium’ mileage. Lastly, vehicles with mileage greater than 60,000 were termed as ‘high’ mileage vehicles.

```
def categorize_mileage(mileage):
    if mileage < 30000:
        return 'low'
    elif mileage >= 30000 and mileage < 600000:
        return 'medium'
    else:
        return 'high'

dataset['mileage_type'] = dataset['mileage'].apply(categorize_mileage)
```

```
dataset[['standard_make','mileage','mileage_type', 'price']]
```

	standard_make	mileage	mileage_type	price
0	Volvo	0.0	low	73970
1	Jaguar	108230.0	medium	7000
2	SKODA	7800.0	low	14000
3	Vauxhall	45000.0	medium	7995
4	Land Rover	64000.0	medium	26995
...	...	...	...	...
402000	Peugeot	5179.0	low	10595
402001	Peugeot	110000.0	medium	2000
402002	Nissan	52760.0	medium	7250
402003	Abarth	10250.0	low	11490
402004	Audi	14000.0	low	20520

luxury_vehicle

Luxury vehicle brands are known for being expensive. Moreover, there are many other brands that make luxury models that appeal to a certain demographic. A luxury vehicle is expected to be more expensive than its non-luxury counterpart. For improved analysis of the correlation between vehicle manufacturer and price, we have made a new feature from two existing features: **standard_make** and **body_type**.

True values were set for vehicles that either belonged to luxury brands or had a body type that is considered luxury.

```
auto_manufacturers = list(dataset['standard_make'].unique())
luxury_manufacturers = ['Jaguar', 'Land Rover', 'Audi', 'Mercedes-Benz', 'BMW', 'Ferrari', 'Bentley', 'Aston Martin', 'McLaren',
                        'Porsche', 'Maserati', 'Rolls-Royce', 'Lexus', 'Jeep', 'Lamborghini', 'Bugatti', 'Maybach', 'Pagani',
                         'Alfa Romeo', 'Cadillac']
body_type = list(dataset['body_type'].unique())
luxury_body_type = ['Convertible', 'Limousine', 'Coupe', 'Saloon']
dataset['luxury_vehicle'] = ((dataset['standard_make'].isin(luxury_manufacturers)) |
(dataset['body_type'].isin(luxury_body_type)))
dataset[['standard_make', 'body_type', 'luxury_vehicle', 'price']]
```

	standard_make	body_type	luxury_vehicle	price
0	Volvo	SUV	False	73970
1	Jaguar	Saloon	True	7000
2	SKODA	SUV	False	14000
3	Vauxhall	Hatchback	False	7995
4	Land Rover	SUV	True	26995
...	...	...	...	...
402000	Peugeot	Hatchback	False	10595
402001	Peugeot	Hatchback	False	2000
402002	Nissan	SUV	False	7250
402003	Abarth	Hatchback	False	11490
402004	Audi	Estate	True	20520

vehicle_age

Age of the vehicle plays a part in the price of the vehicle. A new car will fetch a greater price than a car that is older. Except for some cases where the vehicle is rare, the price of the vehicle depreciates with age. This new feature can be a good indicator of the price of a vehicle. It is created using the year_of_registration column. To calculate the age of the vehicle, 2022 is used as the base year value. So, for example, if the year_of_registration is 2020, the vehicle age will be 2 years. New vehicles will have 0 in the column.

```
def vehicle_age_calculation(year_of_registration):
    if pd.isnull(year_of_registration):
        return 0
    else:
        return (2022-year_of_registration)

dataset['vehicle_age'] = dataset['year_of_registration'].apply(vehicle_age_calculation)
```

```
dataset.head()
```

rd_make	standard_model	vehicle_condition	year_of_registration	price	body_type	crossover_car_and_van	fuel_type	mileage_type	luxury_vehicle	vehicle_age
Volvo	XC90	NEW	NaN	73970	SUV	False	Petrol Plug-in Hybrid	low	False	0.0
Jaguar	XF	USED	2011.0	7000	Saloon	False	Diesel	medium	True	11.0
SKODA	Yeti	USED	2017.0	14000	SUV	False	Petrol	low	False	5.0
Vauxhall	Mokka	USED	2016.0	7995	Hatchback	False	Diesel	medium	False	6.0
nd Rover	Range Rover Sport	USED	2014.0	26995	SUV	False	Diesel	medium	True	8.0

Feature Selection

There are some features in our dataset that can be dropped as they do not provide much insight into whether they have any correlation with the price of vehicles on sale. Hence, some of the columns will be dropped from the dataset.

Firstly, public_reference will be dropped. It is a 15-digit code that is unique throughout the dataset. Each record has its unique public_reference. This ID seems to be the unique identifier for the advertisements. As this feature has no effect on the price of the vehicle, we can drop it.

Secondly, we can drop reg_code. The reason to drop reg_code is that it is similar to the year_of_registration column that is already present in the dataset. Both columns help to identify the age of the vehicle. We can choose among the two which one to use. The year_of_registration column is easier to use because it mentions the exact year while the reg_code column has codes that need to be decoded to find the exact year of registration.

Thirdly, we can drop crossover_car_and_van. This feature is very broad and does not provide enough information to be used efficiently or look for any patterns it may have with the price of the vehicle.

Lastly, we can also drop standard_model as it does not provide enough information that could be used to predict the price of the vehicle.

The final dataset should look something like this after removing some old columns and introducing some engineered features.

```
dataset = dataset.drop(['public_reference', 'reg_code', 'crossover_car_and_van', 'standard_model'], axis=1)
dataset.head()
```

	mileage	standard_colour	standard_make	vehicle_condition	year_of_registration	price	body_type	fuel_type	mileage_type	luxury_vehicle	vehicle_age
0	0.0	Grey	Volvo	NEW	NaN	73970	SUV	Petrol Plug-in Hybrid	low	False	0.0
1	108230.0	Blue	Jaguar	USED	2011.0	7000	Saloon	Diesel	medium	True	11.0
2	7800.0	Grey	SKODA	USED	2017.0	14000	SUV	Petrol	low	False	5.0
3	45000.0	Brown	Vauxhall	USED	2016.0	7995	Hatchback	Diesel	medium	False	6.0
4	64000.0	Grey	Land Rover	USED	2014.0	26995	SUV	Diesel	medium	True	8.0

3. Association and Group Difference Analysis

We ran an association and group difference analysis on the variables that are present in our dataset. This allows us to get useful insights regarding the variables and the relationship between them.

Quantitative - Quantitative

Mileage - Price

We are using mileage and price for this analysis. For association analysis between the two variables, we have used Pearson's correlation coefficient. It is a statistical method that helps to determine the direction and impact of the linear relationship between the two variables. According to our tests, we were able to conclude that there was a negative linear relationship. More mileage means less price. The p-value came out to be 0. The p-value helps to evaluate that the two variables are statistically significant. In simpler terms, we can conclude that this pattern did not occur by chance.

For group difference analysis, t-test was used. The t-test result allows us to conclude whether the difference between the average of the two variable groups is significant or not. In this case, the p-value was 0. This means that the t-statistic is significant.

Mileage - Price

```
import scipy.stats as stats

mileage = dataset['mileage']
price = dataset['price']

# Association Analysis using Pearson's correlation coefficient
corr, p_value = stats.pearsonr(mileage, price)
print('Pearson correlation coefficient:', corr)
print('p-value:', p_value)

# Group Difference Analysis using a t-test
t_stat, p_value = stats.ttest_ind(mileage, price)
print('t-statistic:', t_stat)
print('p-value:', p_value)
```

```
Pearson correlation coefficient: -0.28529740453924174
p-value: 0.0
t-statistic: 300.29541400426325
p-value: 0.0
```

Age of Vehicle - Price

Another pair of variables that were selected were the age of the vehicle and the price. The age of the vehicle was calculated and added to the **vehicle_age** feature. We have used Pearson's correlation coefficient for association analysis and t-test for group difference analysis. The weak negative correlation coefficient indicates that if the age increases the price decreases.

Age of vehicle - Price

```
vehicle_age = dataset['vehicle_age']
price = dataset['price']

# Association Analysis using Pearson's correlation coefficient
corr, p_value = stats.pearsonr(vehicle_age, price)
print('Pearson correlation coefficient:', corr)
print('p-value:', p_value)

# Group Difference Analysis using a t-test
t_stat, p_value = stats.ttest_ind(vehicle_age, price)
print('t-statistic:', t_stat)
print('p-value:', p_value)
```

```
Pearson correlation coefficient: -0.23083221165171208
p-value: 0.0
t-statistic: -421.264850881954
p-value: 0.0
```

Quantitative - Categorical

Mileage Type - Price

For association analysis, we use ANOVA. This will allow us to see if there is a significant difference in the means of the different groups present in our respective variables.

The group difference analysis is done using a t-test. A new column was made with the name **mileage_type_numeric**. The labels low, medium, and high were changed to 0,1 and 2 respectively. The data was transformed so that we could use the library **scipy** to get t-test results.

```
import scipy.stats as stats

def categorize_mileage_numeric(mileage):
    if mileage < 30000:
        return 0
    elif mileage >= 30000 and mileage < 600000:
        return 1
    else:
        return 2

dataset['mileage_type_numeric'] = dataset['mileage'].apply(categorize_mileage_numeric)

quantitative_var = dataset['price']
categorical_var = dataset['mileage_type_numeric']

# Association analysis using ANOVA
f_stat, p_value = stats.f_oneway(quantitative_var, categorical_var)
print(f'F-statistic:', f_stat.round(decimals=2))
print('p-value:', p_value)

# Group difference analysis using t-test
grouped_ttest = df.groupby(categorical_var)
print(grouped_ttest.apply(lambda x: stats.ttest_ind(x['price'], quantitative_var, equal_var=False)))
```

```
F-statistic: 177585.72
p-value: 0.0
mileage_type_numeric
0                      (77.55888712844308, 0.0)
1                     (-145.4423105061231, 0.0)
2    (-2.6671849263098206, 0.037151610991101564)
dtype: object
```

The F-statistic score tells us that there is a significant difference between the mean of different groups. This means that the average price of the vehicle greatly differed on the basis of the mileage of the vehicle.

The tuple values after the F-statistic and p-value tells us the t-test results of the groups that reside in our mileage_type column. A positive t-statistic value means that the average of the first group was greater than the other group. A negative t-statistic value means that the average of the second group was larger than the first group.

Luxury Vehicle - Price

The same statistical methods will be used as above. The data will be transformed so that t-test values can be easily evaluated using the scipy library. The False value will be replaced with zero and the True value will be represented by 1.

```
def boolean_to_numeric(luxury_vehicle):
    if luxury_vehicle == False:
        return 0
    else:
        return 1

dataset['luxury_vehicle_numeric'] = dataset['luxury_vehicle'].apply(boolean_to_numeric)

quantitative_var = dataset['price']
categorical_var = dataset['luxury_vehicle_numeric']

# Association analysis using ANOVA
f_stat, p_value = stats.f_oneway(quantitative_var, categorical_var)
print(f'F-statistic:', f_stat.round(decimals=2))
print('p-value:', p_value)

# Group difference analysis using t-test
grouped_ttest = df.groupby(categorical_var)
print(grouped_ttest.apply(lambda x: stats.ttest_ind(x['price'], quantitative_var, equal_var=False)))
```

```
F-statistic: 177587.64
p-value: 0.0
luxury_vehicle_numeric
0    (-116.31191572323398, 0.0)
1     (75.82062549995281, 0.0)
dtype: object
```

The high F-statistic value and the low p-value means that there is a significant difference between the means of the price of luxury vehicle groups. This is further confirmed from the results of the t-test.

Categorical - Categorical

Luxury Vehicle - Vehicle Condition

The luxury vehicle feature is a categorical feature that was engineered using the standard_make feature of the dataset. Pitting it against the vehicle condition for association and group difference analysis should give us some useful insight.

A positive chi-square value with a low p-value means that there is a significant association between the features: luxury vehicle and vehicle condition. The t-test for group difference analysis shows that there is a significant difference between the proportion of luxury cars that are new versus used.

```
luxury_vehicle = dataset['luxury_vehicle']
vehicle_condition = dataset['vehicle_condition']

# Association analysis using Chi-square test
chi2, p_value, degree_of_freedom, freq_dist = stats.chi2_contingency(pd.crosstab(luxury_vehicle, vehicle_condition))
print('Chi-square statistic:', chi2.round(decimals=2))
print(f"p-value: {p_value:.2f}")

# t-test prep
vehicle_condition_dummies = pd.get_dummies(vehicle_condition)
vehicle_condition_new = vehicle_condition_dummies['NEW']
vehicle_condition_used = vehicle_condition_dummies['USED']

# t-test for New in Vehicle Condition
t_stat_new, p_value_new = stats.ttest_ind(luxury_vehicle, vehicle_condition_new)
print('NEW t-statistic:', t_stat_new.round(decimals=2))
print('NEW p-value:', p_value_new.round(decimals=2))

# t-test for Old in Vehicle Condition
t_stat_used, p_value_used = stats.ttest_ind(luxury_vehicle, vehicle_condition_used)
print('USED t-statistic:', t_stat_used.round(decimals=2))
print('USED p-value:', p_value_used.round(decimals=2))
```

```
Chi-square statistic: 522.73
p-value: 0.00
NEW t-statistic: 357.05
NEW p-value: 0.0
USED t-statistic: -604.89
USED p-value: 0.0
```

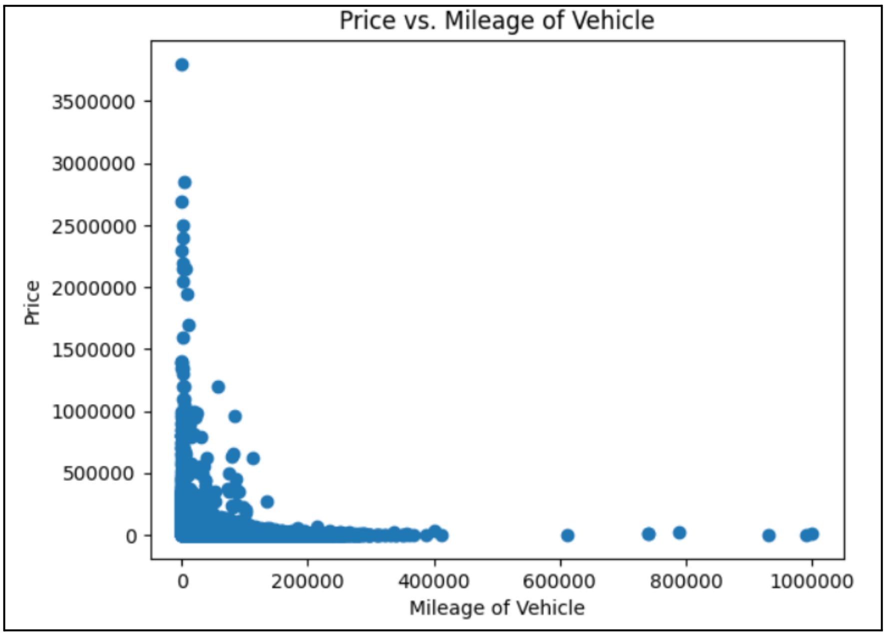
What are the best predictors of the price of a vehicle?

The analysis done up till now gives a clear picture about the features that play a significant role in the price of the vehicle. Hence, to support our findings, we will use visualizations to provide a better overall picture of how certain features are good predictors of the price of a vehicle.

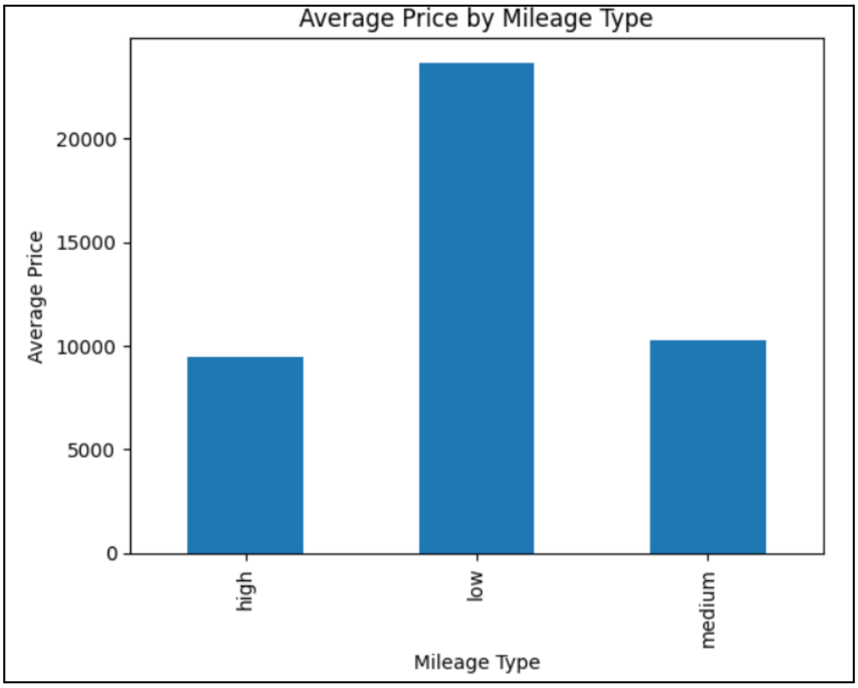
Mileage

Mileage is an important feature in the dataset. This figure tells us how much the vehicle has been used in the past. This feature gives a good idea about the condition of the vehicle for sale. Hence, it should be a good indicator of the price of a vehicle.

```python
# Extract the age and price variables
mileage = dataset['mileage']
price = dataset['price']

# Create the scatter plot
plt.scatter(mileage, price)
plt.xlabel('Mileage of Vehicle')
plt.ylabel('Price')
plt.title('Price vs. Mileage of Vehicle')
plt.ticklabel_format(style='plain', axis='y')
plt.ticklabel_format(style='plain', axis='x')
plt.show()
```



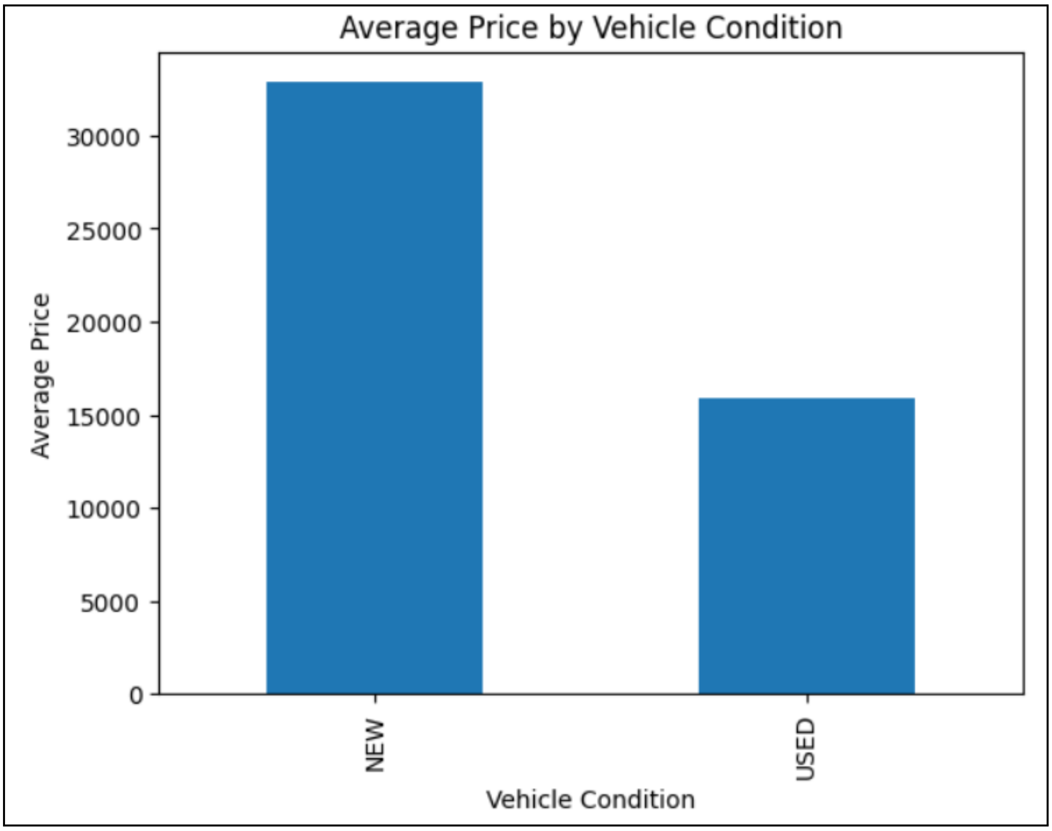

```python
# Calculate the average price for each mileage type
average_prices = dataset.groupby('mileage_type')['price'].mean()

# Plot the average prices as a bar chart
average_prices.plot(kind='bar')
plt.xlabel('Mileage Type')
plt.ylabel('Average Price')
plt.title('Average Price by Mileage Type')
plt.show()
```



Vehicle Condition

Vehicle condition plays a vital part in deciding the price of a vehicle. New vehicles fetch a higher price than used vehicles. The statistics also prove the previous statement.

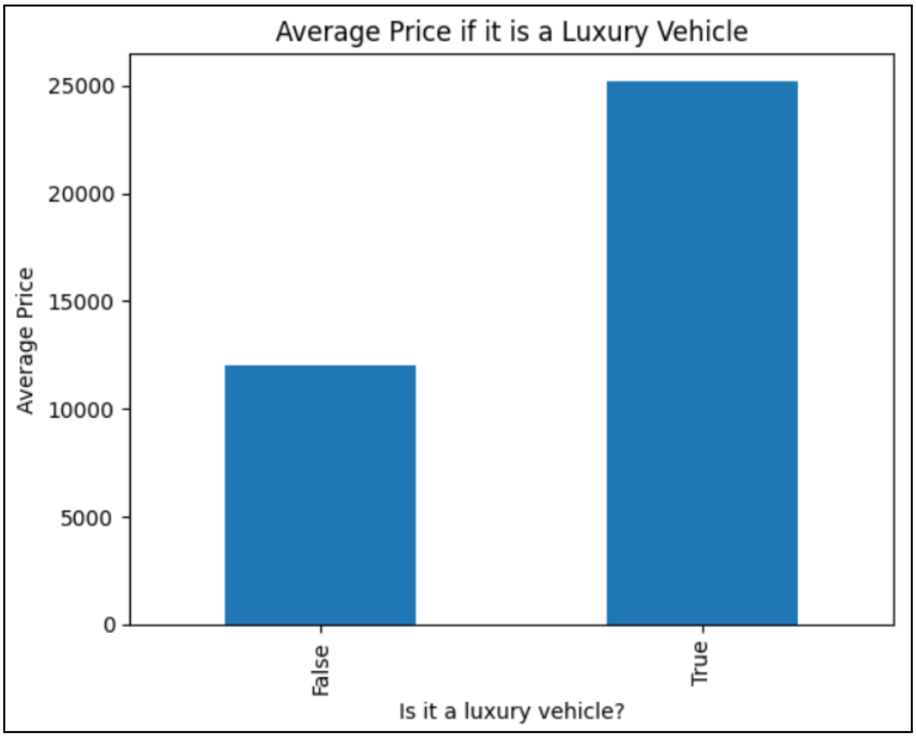
```
# Calculate the average price for each vehicle condition
average_prices = dataset.groupby('vehicle_condition')['price'].mean()

# Plot the average prices as a bar chart
average_prices.plot(kind='bar')
plt.xlabel('Vehicle Condition')
plt.ylabel('Average Price')
plt.title('Average Price by Vehicle Condition')
plt.show()
```



Luxury Vehicle

This feature was engineered using standard_make and body_type features present in the dataset. This engineered feature is a good predictor of the price of the vehicle as well. Vehicles made by specific manufacturers with certain body types have greater value than conventional vehicles.

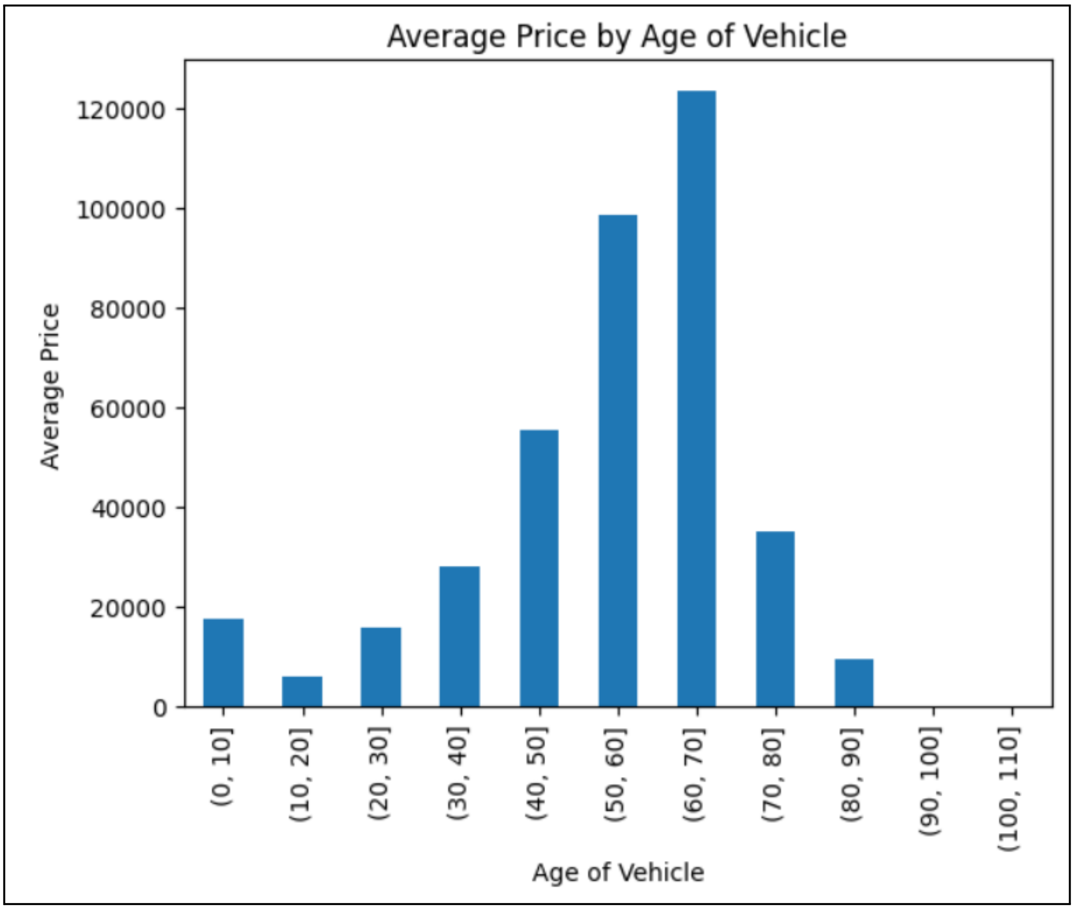
```
# Calculate the average price for luxury vehicle
average_prices = dataset.groupby('luxury_vehicle')['price'].mean()

# Plot the average prices as a bar chart
average_prices.plot(kind='bar')
plt.xlabel('Is it a luxury vehicle?')
plt.ylabel('Average Price')
plt.title('Average Price if it is a Luxury Vehicle')
plt.show()
```



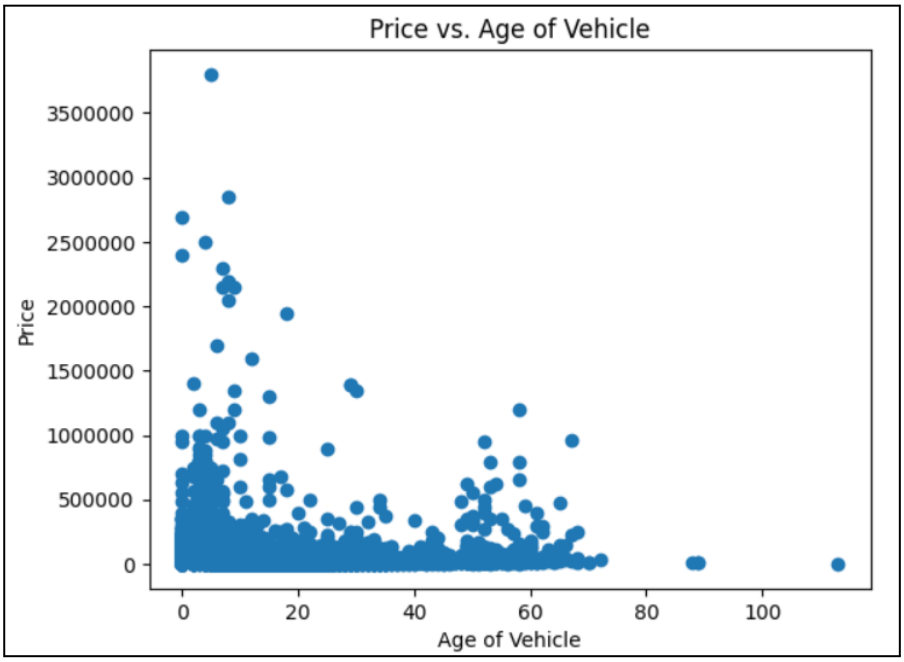
Age of Vehicle

This is an interesting feature of vehicles. It is understood that new vehicles are more expensive than older vehicles. However, after studying the dataset, we were able to conclude that the average price of the vehicle can greatly vary if the vehicle in question is antique or considered vintage. The trend shows that vehicles that are of age greater than 10 and less than 20 years have the lowest average price.

```
min_age = dataset['vehicle_age'].min().astype(int)
max_age = dataset['vehicle_age'].max().astype(int)

# Divide the age variable into bins based on the min and max values
bins = pd.cut(dataset['vehicle_age'], range(min_age, max_age+1, 10))

# Calculate the average price for each bin
average_prices = df.groupby(bins)['price'].mean()

# Plot the average prices as a bar chart
average_prices.plot(kind='bar')
plt.xlabel('Age of Vehicle')
plt.ylabel('Average Price')
plt.title('Average Price by Age of Vehicle')
plt.show()
```



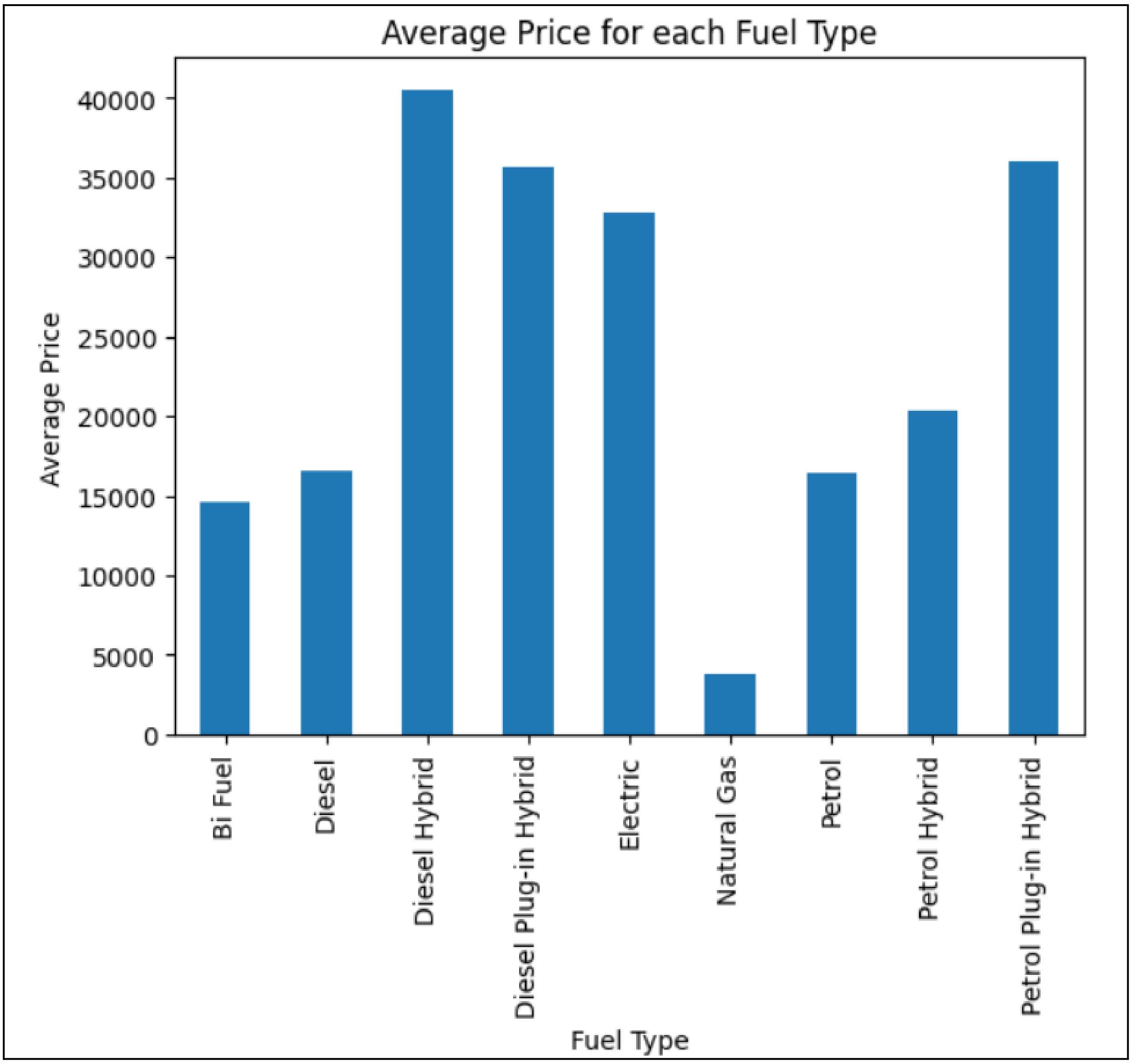

```
# Extract the age and price variables
age = dataset['vehicle_age']
price = dataset['price']

# Create the scatter plot
plt.scatter(age, price)
plt.xlabel('Age of Vehicle')
plt.ylabel('Price')
plt.title('Price vs. Age of Vehicle')
plt.ticklabel_format(style='plain', axis='y')
plt.show()
```



Fuel Type

This feature helped to understand that modern vehicles with new fuel types such as hybrids fetch a higher price. The use of cutting-edge technology requires a lot of research and development. This actually drives up the average price of the vehicles.

```
# Calculate the average price for each fuel type
average_prices = dataset.groupby('fuel_type')['price'].mean()

# Plot the average prices as a bar chart
average_prices.plot(kind='bar')
plt.xlabel('Fuel Type')
plt.ylabel('Average Price')
plt.title('Average Price for each Fuel Type')
plt.show()
```



Citation

[1] National Numbers. (n.d.). Year of Issue. Retrieved from https://www.nationalnumbers.co.uk/dvla-guide/year-of-issue-149.htm