Table of Content

1. Introduction

This is a Coursework project for Principles of Data Science. The dataset is a **Car Sale Adverts** dataset provided by **Auto Trader**, one of Manchester Metropolitan University-industry partners.

An overview of the business

Auto Trader is the UK and Ireland's largest digital automotive marketplace. It's the go-to destination for car buyers and has been for the past 40 years. **Auto Trader** exists to Drive change together. Responsibly, Its aim is to grow both car-buying and selling audiences. It also aims to change how the UK shops for cars by providing the best online car-buying experience and enabling all retailers to sell online. It aims to build stronger partnerships with its customers, use its voice and influence to drive more environmentally friendly vehicle choices, and create a diverse and inclusive culture. **Auto Trader** was listed on the London Stock Exchange in March 2015 and is now a member of the FTSE 100 Index Auto Trader's Website.

2. Data Understanding and Exploration

The dataset contains an anonymised collection of adverts with information on vehicles such as brand, type, colour, mileage, as well as the selling price.

2.1 Meaning and Type of Features; Analysis of Distributions

Features Definitions

- **public_reference:** The unique ID for each car advert.
- **mileage:** The total distance covered by the car to date.
- **reg_code:** This is the unique code for each year of registration.
- **standard_color:** The colour of the car.
- **standard_make:** The brand of the car.
- **standard_model:** The brand model of the car.
- **vehicle_condition:** The vehicle condition; either *NEW* or *USED*.
- **year_of_registration:** The first registration year of the car.
- **price:** The price of the car in pounds(£).
- **body_type:** The body type of the car i.e SUV, Saloon, Minibus, and so on.
- **crossover_car_and_van:** A crossover is a type of automobile with an increased ride height that is built on unibody chassis construction shared with passenger cars, as opposed to traditional sport utility vehicles (SUV) which are built on a body-on-frame chassis construction similar to pickup trucks.
- **fuel_type:** The type of fuel the car runs on.

General information about features, non_missing values count, data types, and also the shape of the dataset.

```
<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
```

The dataset consists of 402,005 rows and 12 columns.

Descriptive statistics of numerical features

	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

Analysis of Features Distribution

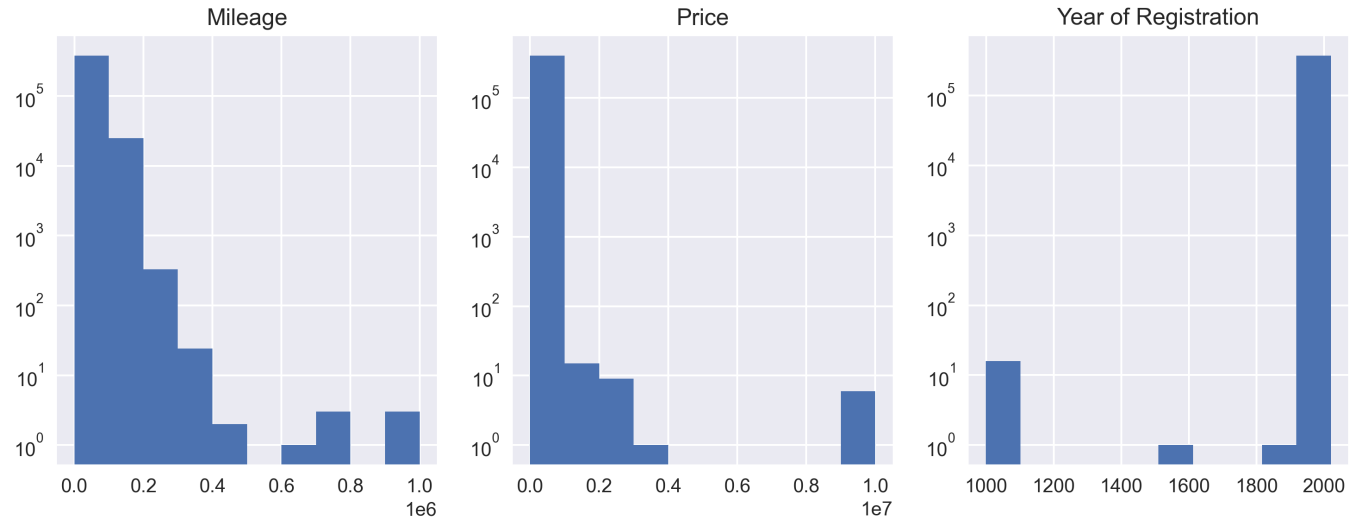

Fig.1 Analysis of Features Distribution for (mileage, price, and year_of_registration)

Both the `Mileage` and `Price` histogram plots are rightly skewed. This implies that the mean of these features is greater than its median. Also, for the `Mileage`, **77%** of cars have mileages 0km/h and 60,000km/h, **10%** of cars have between 60,000km/h and 80,000km/h, and **13%** have mileages greater than 80,000km/h. For `Price`, **94%** of the car prices are between £120 and £40,000. **6%** of cars have prices above £40,000. For the `Year_of_registration`, **91%** of cars were registered between the years 2000 and 2020.

2.2. Identification/Commenting on Missing Values

```
pd.DataFrame(dict(missing_value=adverts.isnull().sum(),
                  missing_value_proportion=adverts.isnull().sum() * 100 / len(adverts))
)
```

	missing_value	missing_value_proportion
public_reference	0	0.000000
mileage	127	0.031592
reg_code	31857	7.924528
standard_colour	5378	1.337794
standard_make	0	0.000000
standard_model	0	0.000000
vehicle_condition	0	0.000000
year_of_registration	33311	8.286215
price	0	0.000000
body_type	837	0.208206
crossover_car_and_van	0	0.000000
fuel_type	601	0.149501

There are some columns in the dataset that doesn't have complete observations of 402,005. About **18%** of the dataset has missing values. From the data frame above, 6 of the columns have missing values with `year_of_registration` and `reg_code` having the highest percentage of missing values, ***8.29%*** and ***7.92%*** respectively. Other columns with missing values are **(`standard_colour`, 1.34%), (`mileage`, 0.03%), (`body_type`, 0.20%), and (`fuel_type`, 0.15%).**

2.3. Identification/Commenting on Outliers and Noise

Identifying outliers by visualizing the data points using `seaborn.boxplot()`.

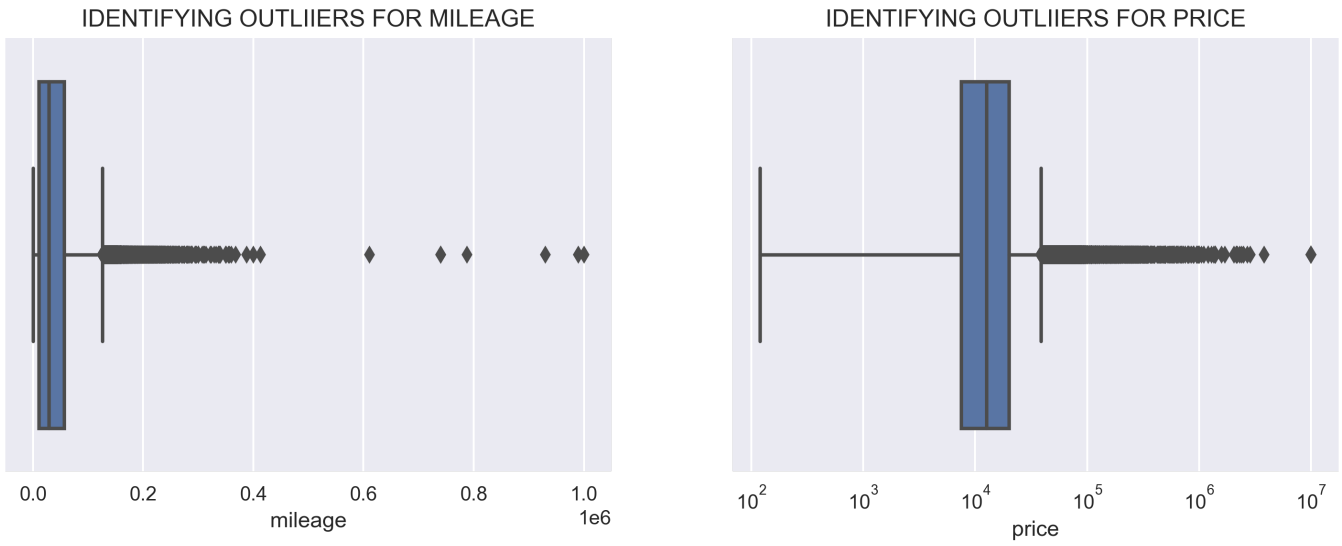

Fig.2 Identifying Outliers using Boxplot

The maximum `mileage` is **999,999km/h** while the maximum `price` of a car is **£9,999,999**. It's obvious from the boxplot, the majority of the data points sit between **0km/h** and less than **120,000km/h** for `mileage`. For `price`, I used a logarithm scale so as to be able to view the data points clearly. The majority of the data points sit around **£8,000** and **£40,000**.

3. Data Processing

In this section, data processing or data preparation simply means performing processes on raw data to prepare the data for further processing, analysis, or modeling. It is an important step in the data mining process. There are several methods of data processing and these includes but not all: Sampling, transformation, Imputation, normalization/Standardisation, feature engineering, and Removing Outliers.

3.1. Dealing with Missing Values, Outliers, and Noise

Dealing with missing values

- `Year of Registration` & `Reg Code`

Dealing with missing values in the `Year_of_registration` and `Reg_Code` columns by scraping the UK's car registration codes for age and year identifiers tables from the Wikipedia page UK Vehicle Registration Codes and mapping the reg_code from the Wikipedia table to fill in the missing year of registration.

There are some human errors in the **Year of Registration** column. For example, we have 1008,1009, and 1018 instead of 2008, 2009, and 2018. Replaced the wrongly imputed years with the correct ones

```
adverts['year_of_registration'].replace([1006.0, 1007.0, 1008.0, 1009.0, 1010.0,
1015.0, 1016.0, 1017.0, 1018.0],
                                        [2006.0, 2007.0, 2008.0, 2009.0, 2010.0,
2015.0, 2016.0, 2017.0, 2018.0], inplace=True)
```

There are also missing values in the `reg_code` column and some other columns. The `vehicle_condition` also seems to have majorly ***NEW*** cars. About **31,570** missing values in the `reg_code` column also have their `year_of_registration` to be missing.

For every ***NEW*** car, there's no year of registration and reg_code attached to it. This is because it hasn't been purchased and registered by any user. So, I filled the missing `Year_of_registration` for **NEW** cars with ***2020*** and `Reg_Code` for ***NEW*** cars with ***20***. This is because the maximum `year_of_registration` in the dataset is 2020 and also the **Public_reference** column indicates the year, month, and day the advert was made, this also aids in my decision to use ***2020*** as the year of registration for new cars.

```
len(adverts[(adverts['year_of_registration'].isnull()) & ~
(adverts['reg_code'].isnull())])
```

The code snippet above indicates that there are **1741** observations where **Year of Registration** was missing and **Reg_code** isn't. Only **USED** cars have their year_of_registration missing but have some values for reg_code. After filling the missing **Year of Registration** by mapping the **Reg_code** and **Year of Registration** gotten from the UK's Car Vehicle Registration Code using the codes below:

```
    reg_code_not_missing['year_of_registration'] =
reg_code_not_missing.index.map(modified_age_identifier.set_index('code')['year'])
```

By applying the result to the main data frame using conditional variable assignment,

```
    (adverts.loc[(adverts['year_of_registration'].isnull()) & ~
(adverts['reg_code'].isnull()), 'year_of_registration'] )=
missing_year_of_reg['year_of_registration']
```

there were still missing values in the `Year_of_registration` column and their vehicle condition was **USED**. So I filled the rows with the mode of the `year_of_registration` of USED cars. Then I dropped the `reg_code` column after filling the `Year_of_registration` column and also dropped years less than **1900**.

- Categorical Features (`Fuel_type, Body_type, Standard_colour`)

Missing values in categorical features were filled using the mode of the grouby result of their `standard_make`(brands) and `standard_model`(brand model). Scipy's stats mode

function(`scipy.stats.mode()`) was used to fill in the missing values.

```
def fillna_with_mode(missing_col):
    adverts[missing_col] = adverts.groupby(['standard_make', 'standard_model'])
[missing_col].apply(lambda x: x.fillna(scipy.stats.mode(x)[0][0]))
    return adverts
```

After filling in the mode, there were still some missing values. These missing values were observations that have **NaN** as the mode of such unique make and model or the make and model only appeared once in the dataset and doesn't have a category assigned to it. Created another function to handle the above situation, by grouping the make and model of each car and performing value counts on the missing column, then sorting the counts in descending order. Thereafter, I took the first index to replace the missing values.

```
def refillna_with_mode(missing_col):

    missing_values = adverts[adverts[missing_col].isnull()]
    brand_model = missing_values[['standard_make','standard_model']]

    results = []
    for i, j in brand_model.iterrows():
        try:
            result = (adverts[(adverts['standard_make']==j['standard_make']) &
(adverts['standard_model']==j['standard_model'])][missing_col]
            .value_counts().sort_values(ascending=False).index[0])
            results.append(result)
        except IndexError:
            results.append("no value")
    brand_model[missing_col] = results
    return brand_model
```

Dealing with Outliers and Noise

For `Mileage`, I capped outliers for values greater than 200,000km/h and replaced the outliers with 200,000km/h. In general, most modern cars can cross 200,000 miles without any major issues, provided the vehicle is well-maintained. However, it comes as no surprise that many automobiles with 400,000 and even 500,000 miles on them have undergone incredible care and upkeep, frequently with the owners performing the necessary maintenance work themselves. Some cars having mileages greater than 200,000km/h might be taxis, road trip vans, cars used for hire purchases, and public transportation.

For `price`, most of the top outliers in the price feature are expensive and luxurious cars, with most of them having low mileage. However, cars having an almost price of £10 million and are also USED don't look real. I verified some of these cars' prices and used the year of registration as the production year to check the value of the NEW cars, and most of them were a little over £3million. Therefore, I capped outliers at values greater than £4million and dropped the outliers.

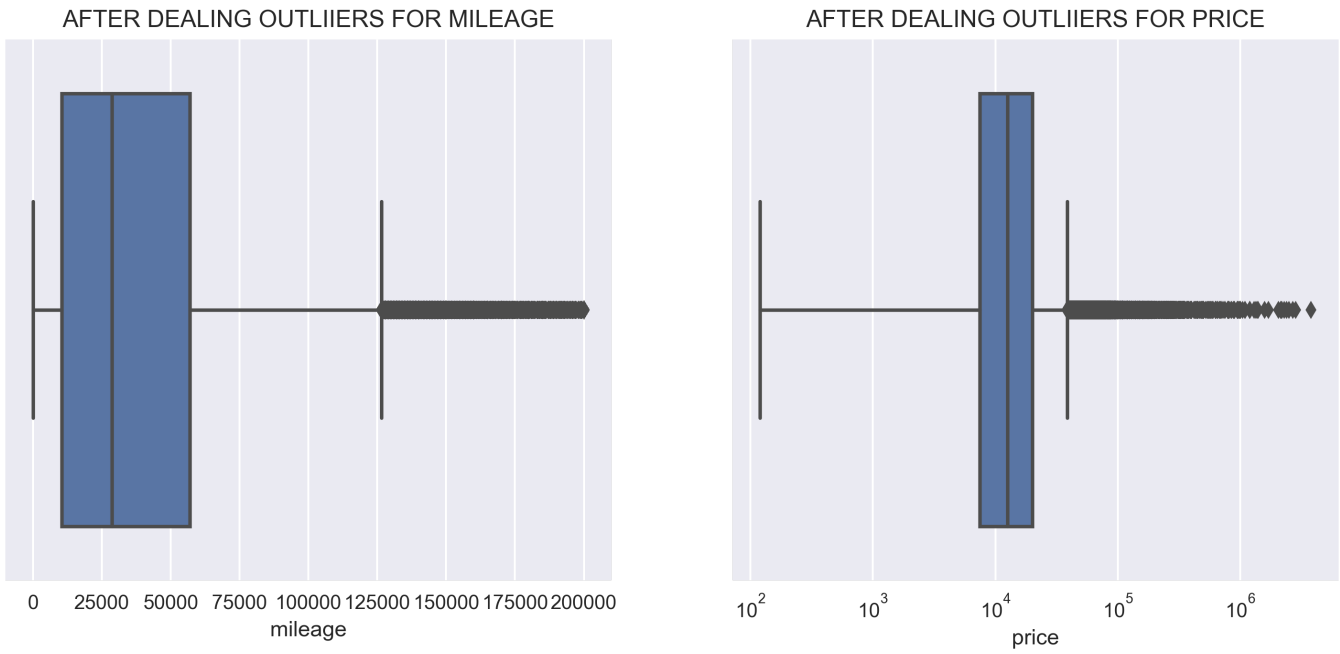

Fig.3 Boxplots After Dealing with Outliers

3.2. Feature Engineering, Data Transformations

Feature Engineering

A critical look at the `Public_reference` column shows the year, month, day, and some ID of each advert. I created a new column; `advert_year` by grabbing the first 4 digits from the `public_reference` column.

```
adverts['advert_year'] =
adverts['public_reference'].astype(str).str[:4].astype('int64')
```

Another column was created by subtracting the `Year_of_registration` from the `advert_year`. A third column was created for `annual_mileage` by dividing the `mileage` by the `vehicle_age`.

Data Transformations

Transformed some features(e.g. `mileage`, `annual_mileage`) to the correct data types and also encoded a categorical variable(`crossover_car_and_van`). I removed extra spaces from all categorical variable columns and also transformed all categorical features to Upper case.

```
    # transforming all object values to upper case
    adverts[categorical_feat] = adverts[categorical_feat].apply(lambda x:
x.astype(str).str.upper())

    # remove spaces inbetween words to form a uniform name
    adverts[categorical_feat] = adverts[categorical_feat].apply(lambda x:
x.astype(str).str.replace(' ',''))
```

3.3 Subsetting (e.g., Feature Selection, Data Sampling)

Data Sampling

A representative sample of the data was taken using the stratified sampling method. Below is the stratified sampling code:

```
def stratified_sample(df,col, N, seed=42):
    grouped = df.groupby(col, group_keys=False)
    rows = grouped.apply(lambda x: x.sample(int(np.rint(N*len(x)/len(df)))))
    samples = rows.sample(frac=1,random_state=seed)
    return samples.reset_index(drop=True)
```

Subsetting

Subsets were made on vehicle conditions i.e New and Used. It is worth having a deep understanding of how some features(e.g. `price`, `mileage`, etc) of used and new cars differ. For new cars, 48.5% of the cars have SUVs as the most advertised body type, and **34.6%** of cars have HATCHBACK. However, for Used cars, HATCHBACK happens to be the most advertised car body type, followed by SUV.

Feature Selection

There are several feature selection techniques, however, to gauge the feature importance, I'll use Pearson's Correlation Coefficient because the important variables have a strong correlation with the target variable, and correlation can be used to select features. Variables should also be uncorrelated among themselves while being correlated with the target variable.

We can anticipate one variable from another if the two are correlated. As a result, if two features are correlated, the model only actually requires one of them as the other does not provide any new information.

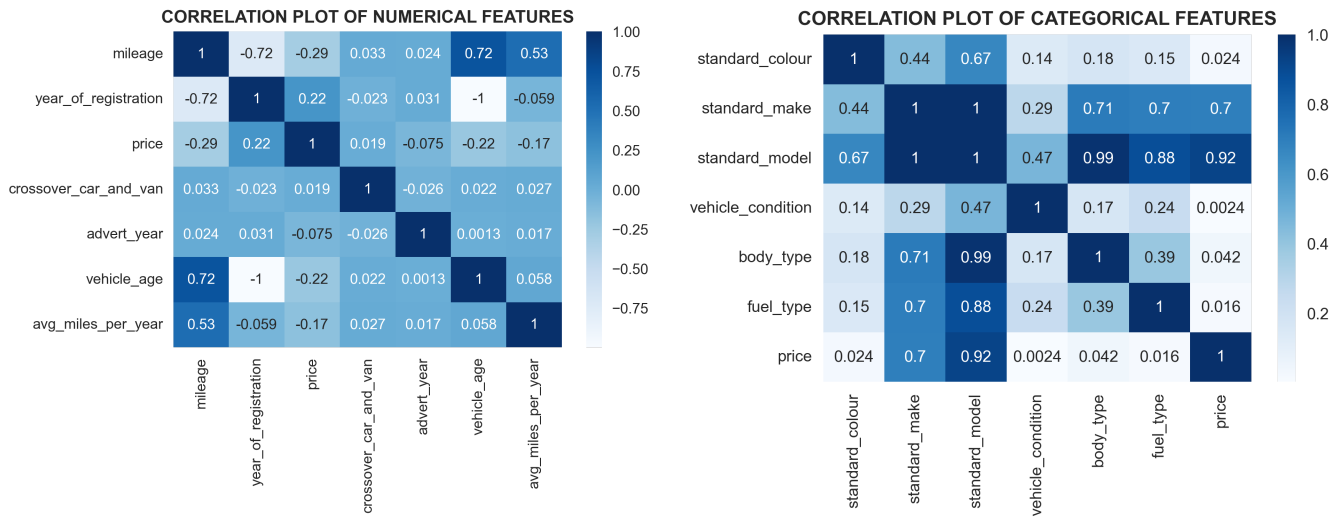

Fig.4 Correlation plots of numerical and categorical features.

4. Association and Group Differences Analysis

Quantitative - Quantitative

Relationship between Price and Mileage & Price and Vehicle Age

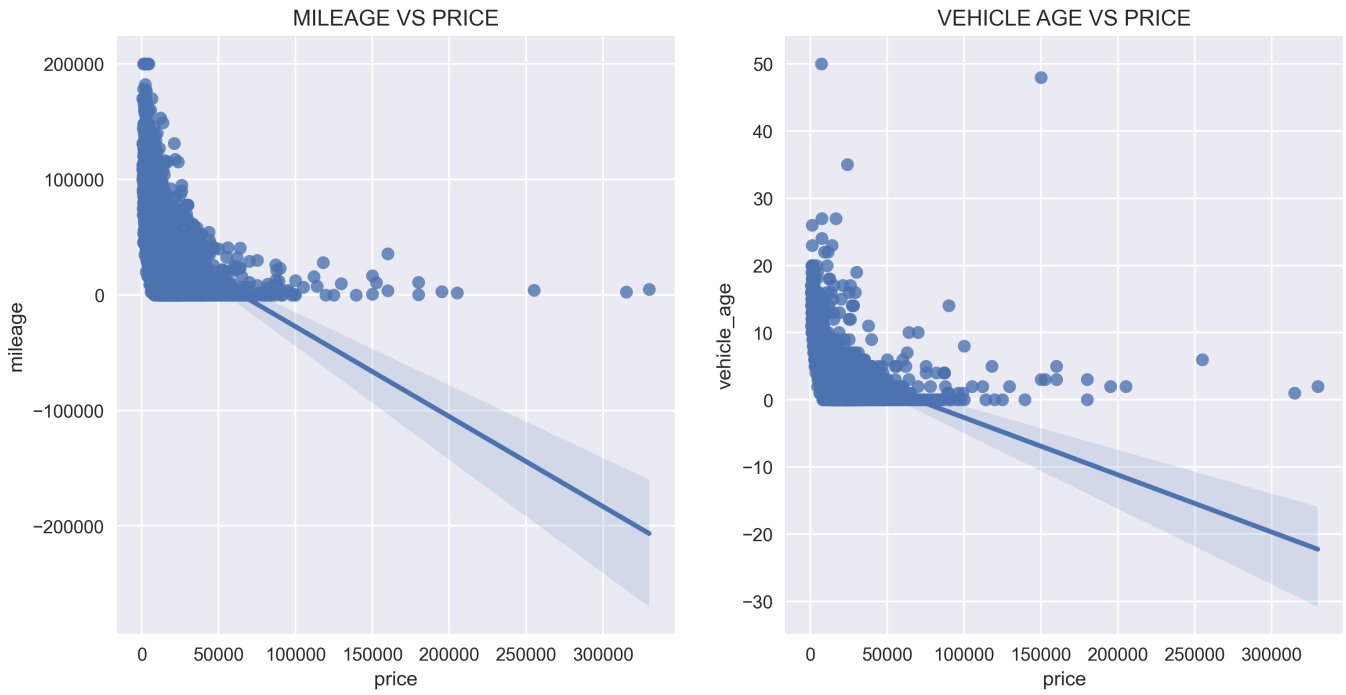

Fig.5 Relationship between mileage and vehicle age with price.

From the plot above, `mileage` and `vehicle_age` both have a strong negative correlation coefficient with `price`. This simply implies that as `mileage` and `vehicle_age` increase, there's a corresponding decrease in the price of a car. In a few cases, where `mileage` and `vehicle_age` is low and the price of the car is high, these cars are antics/vintage cars.

Relationship between Price, year of registration, and Fuel type

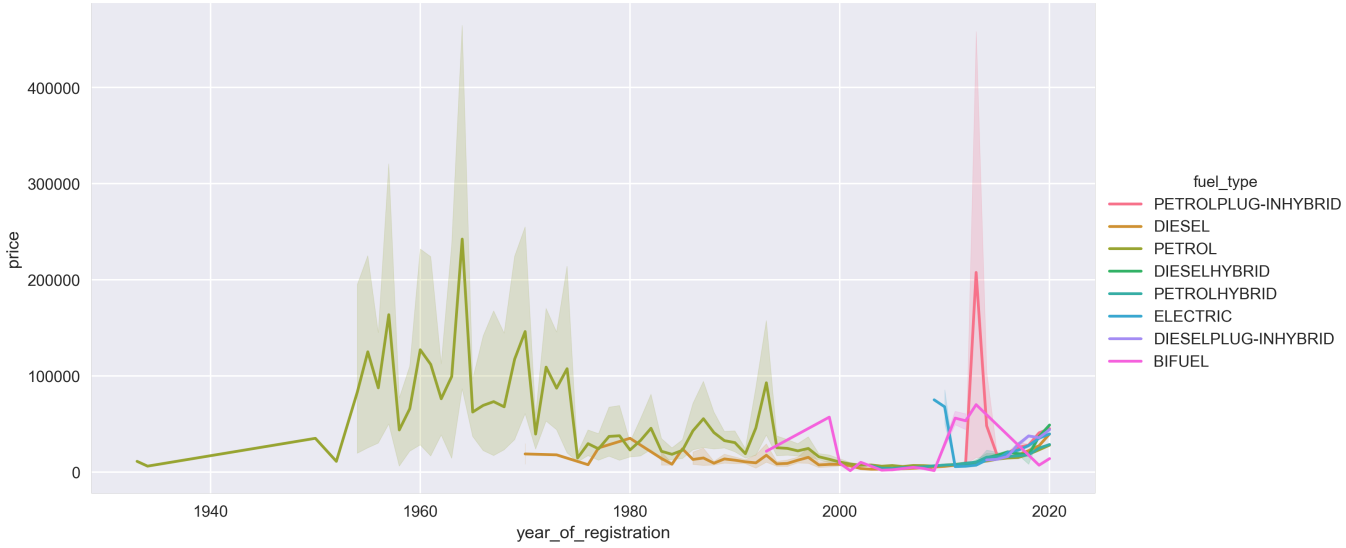

Fig.6 Relationship between Price, year of registration, and Fuel type

Between 1954 and 1977, Petrol was the only available combustion fuel type. It's no surprise that cars from that year happen to have high prices compared to other cars of the same fuel type (Petrol) because most of them are vintage cars. Starting from 1993, BiFuel or Natural Gas combustion engines were built in cars. Then Hybrid came in the late 90s and became popular in the early 20s. In the last 5 years, there has been a high demand for hybrid cars, this is because the world is evolving and it's trying to reduce air pollution from Petrol and Diesel cars. There's also an upward trend in the prices of hybrid cars.

Quantitative - Categorical

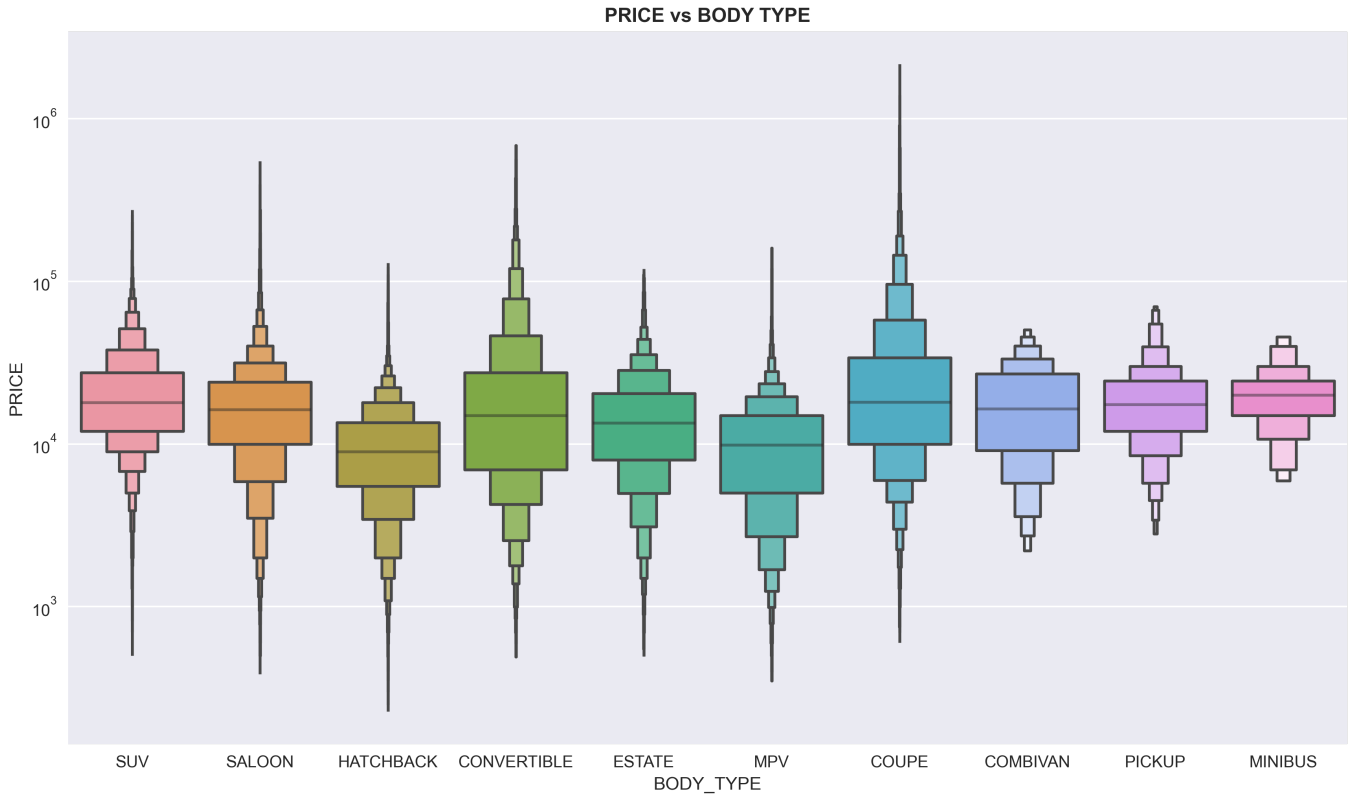

Fig.7 Boxplot showing relationships between Price, and Body Type

I used Boxenplot because interpreting them can be more straightforward. The thicker boxes represent a bigger part of the total population which makes it easier to interpret and understand what's going on with the data. In Fig.7, the majority of the body type have a median value between £8,000 and £40,000. On average, SUVs are more expensive than Hatchbacks and Saloon cars.

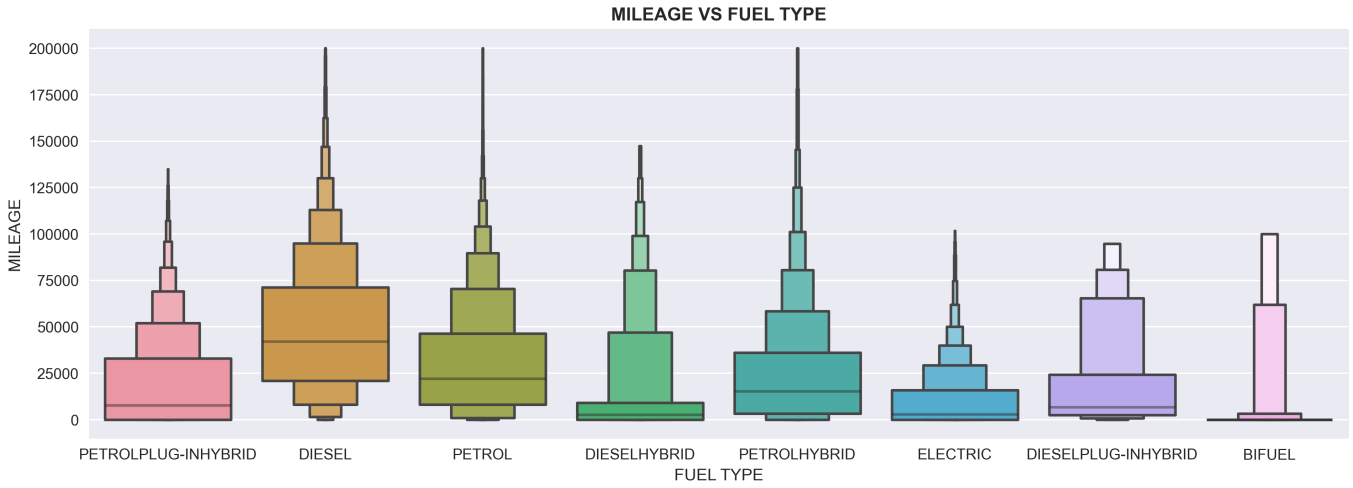

Fig.9 Boxplot showing relationships between Mileage and Fuel type

Diesel cars are more economical to fuel than petrol cars, giving them better mileage for the money. This is due to the fact that when compared like for like, diesel fuel has a higher energy content than petrol. As a result, even while diesel is more expensive per litre than petrol, if you frequently drive long distances, you'll wind up paying more for fuel altogether. In Fig.9, the Majority of diesel cars are around 21,000 to 74,000 miles.

Categorical - Categorical

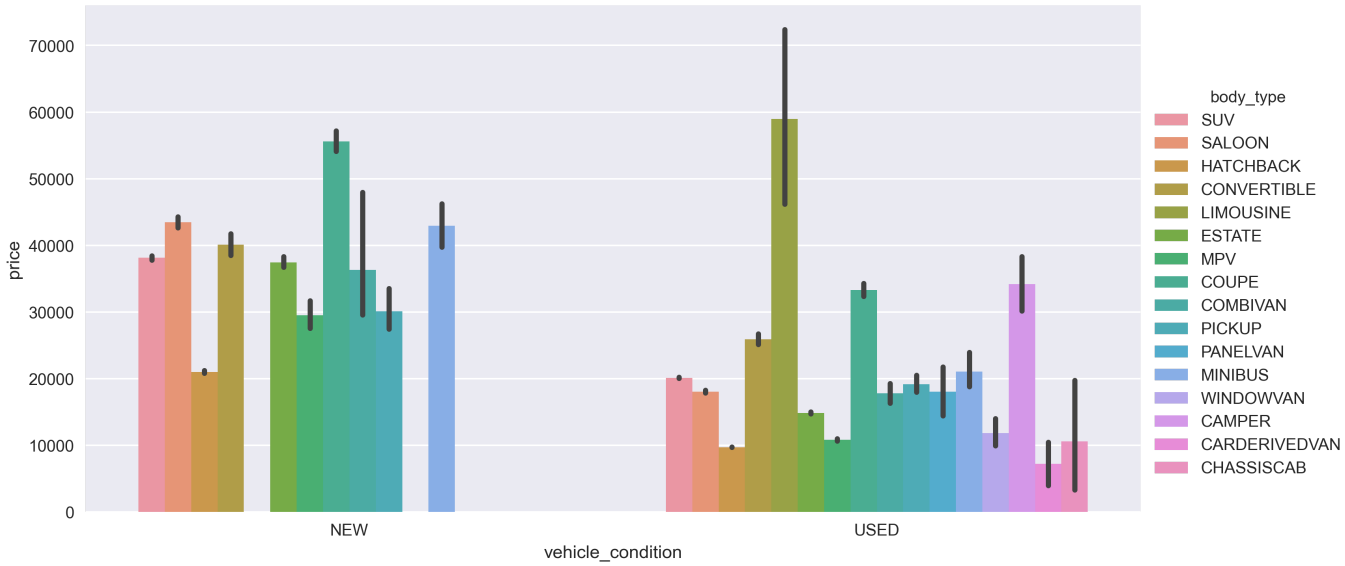

Fig.10 Barplot showing relationships between Price, Vehicle condition, and Body type

The plot above shows how `body_type` and `vehicle_condition` is distributed with `price`. It shows cars with a Coupe body type car is the most expensive on average for new cars. The hatchback is a little over £20,000 pounds on average for new cars, while SUVs, Saloons, and Convertibles are £38,000, £43,000, and £39,000 on average.

For Used cars, from Fig.10, Limousine cars happen to have the highest price on average. Some examples of these cars are ROLLS-ROYCE PHANTOM, MAYBACH 62, and DAIMLER LIMOUSINE. While Hatchbacks on average have a price between £9,000 and £10,000.

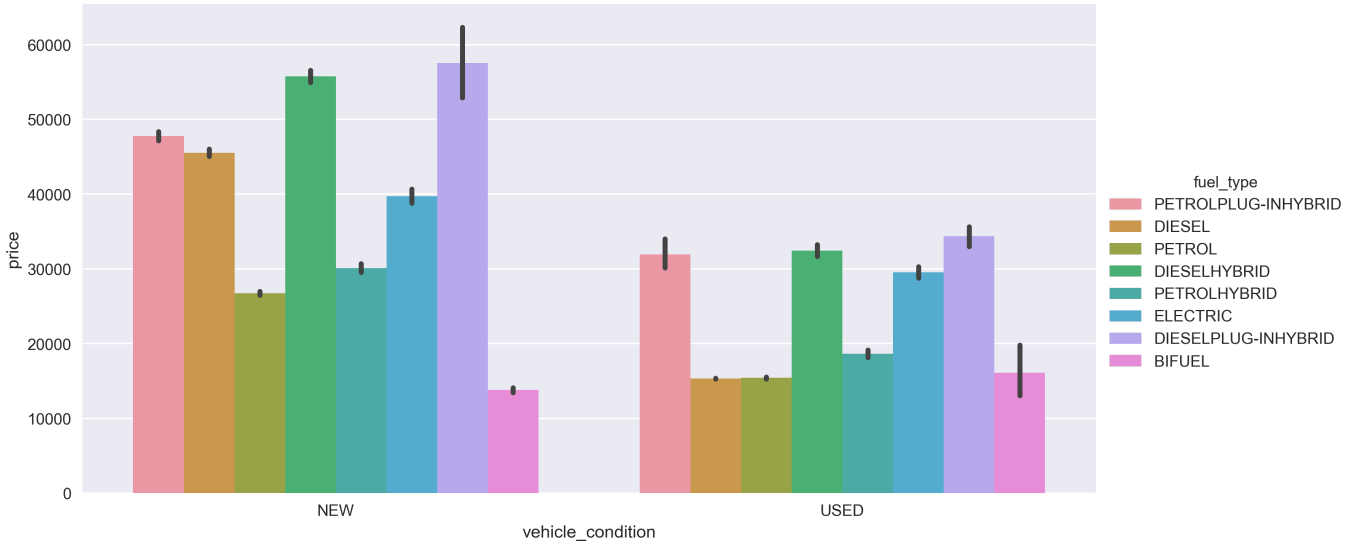

Fig.11 Barplot showing relationships between Price, Vehicle condition, and Fuel type

Generally, for new cars, prices of Hybrid cars are higher compared to other fuel types. For new cars, cars that run on diesel are more expensive compared to cars of other fuel types. New Diesel Plug-in Hybrid cars have over £55,000 on average. A quick look at the used cars, Hybrid cars also happen to have high car prices compared to cars of other fuel types.

Recommendation/Conclusion

From the overall analysis, the best predictors of car price are as follows; `year_of_registration`, `mileage`, `vehicle_age`, `standard_make`, `standard_model`, and `body_type`. A few of them have a negative correlation with price. For `year_of_registration`, the younger the year of registration of a car, the more expensive the price of the car. `Mileage` also is a good predictor of price. From the correlation plot, it has a strong negative correlation with price, i.e the higher the mileage of a car, the lesser the price of that car. `Body_type` is also a good predictor of price. The plot in Fig. 10, shows how price is affected by different car body types. The `standard_make` and `standard_model` of a car are very strong predictors of price. Brands like Ferrari, Lamborghini, Rolls-Royce, Mercedes Benz, BMW, Maybach and other luxurious and executive cars are more expensive than regular brands like Volkswagen, Mazda, Opel, Suzuki, and the like.

It is interesting to note that generally Hybrid cars are expensive on average in comparison to other fuel types. However, there are still some exceptions to big brand and luxurious cars like Ferrari, Lamborghini, Rolls-Royce, Bugatti, and Mclaren that run on petrol or diesel. it is also worth noting that, Diesel fueled cars (including hybrid) have high average mileage covered per year due to their low consumption of fuel which enables cars that run on diesel to travel longer distances. Also, There's a strong positive correlation between `vehicle_age` and `mileage`, i.e as the vehicle age increases, so does the mileage of the vehicle increase.

With the thorough analysis report above, Autotrader can verify the authenticity of any vehicle before posting the advert on the website. If a car is old(`vehicle_age`) and it has a low mileage, it simply means the engine of that vehicle has been replaced. Lastly, the detailed analysis can help in the valuation of incoming vehicles before being placed on adverts.