MACHINE LEARNING REPORT ON THE CAR SALE ADVERTS

TITLE	Machine Learning Concepts
NAME	GODWIN AJAYI
COURSE CODE	6G7V0015
ID	22553197

This is a coursework on the principles of data science

About the dataset

The Data set is from Auto Trader

Auto Trader prides itself on being the most trusted automotive marketplace. It's the go-to destination for car buyers and has been for the past 40 years.

Auto Trader has over 90% prompted brand awareness with consumers and attracts over 50 million cross platform visits each month. The audience is not only large but highly engaged with a 75% share of minutes spent across automotive platforms.

Import required libraries

In [1]:

```
import pandas as pd
import matplotlib as mlb
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn import preprocessing
from scipy import stats
!pip install pandas-profiling --quiet
!pip install phik==0.11.1 --quiet
!pip install missingno --quiet

# Supressing the warning messages
import warnings
```

Reading the data into python

In [2]:

1. Data Understanding and Exploration

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

Basic Data Exploration

This step is performed to guage the overall data. The volume of data, the types of columns present in the data.

There are four commands which are used for Basic data exploration in Python

- head() : This helps to see a few sample rows of the data
- info() : This provides the summarized information of the data
- describe() : This provides the descriptive statistical details of the data

- nunique(): This helps us to identify if a column is categorical or continuous

In [3]:

Out[3]:

	public_reference	mileage	reg_code	standard_colour	standard_make	standard_model	vehicle_condition	year_of_
0	202006039777689	0.0	NaN	Grey	Volvo	XC90	NEW	
1	202007020778260	108230.0	61	Blue	Jaguar	XF	USED	
2	202007020778474	7800.0	17	Grey	SKODA	Yeti	USED	
3	202007080986776	45000.0	16	Brown	Vauxhall	Mokka	USED	
4	202007161321269	64000.0	64	Grey	Land Rover	Range Rover Sport	USED	

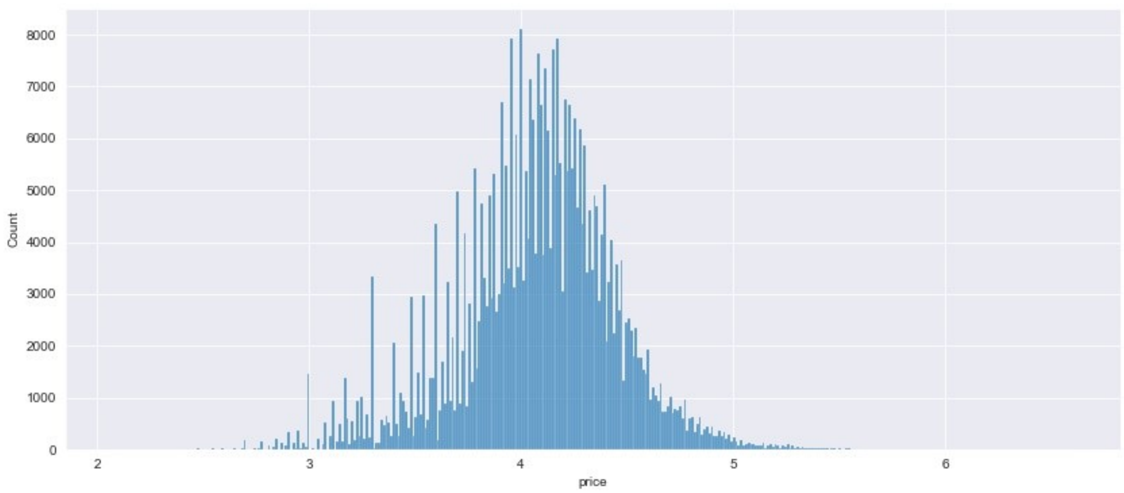
In [4]:

Out[4]:

```
public_reference         402005
mileage                   80634
reg_code                     72
standard_colour              22
standard_make               110
standard_model             1168
vehicle_condition             2
year_of_registration         84
price                     30578
body_type                    16
crossover_car_and_van         2
fuel_type                     9
dtype: int64
```

- Summary of data: 402005 rows, 12 columns

Looking at the distribution of Target variable



> We can remove the skewness of a random variable by transforming it with log transformation. In this case, i've applied log transformation to make price distribution look normal

The data distribution of the target variable is satisfactory to proceed further

1.1. Meaning and Type of Features

Data description

- Price: The Price of the cars. This is the Target Variable!
- mileage: aggregate length or distance in miles: such as. : the total miles traveled especially in a given period of time.
- reg code: The unique registration code for each cars
- Year of registration: The year the car was registered
- Standard make refers to the brand of the vehicle.
- Standard model refers to the specific vehicle model. Using the example of a Nissan Altima, Nissan is the make, while Altima is the model. ATV/UTV.
- vehicle_condition: this is the condition of the vehicle, with the value USED indicating that the vehicle is pre-owned.
- Fuel type: The type of fuel the car uses
- Body type: A car body type is a categorisation of a vehicle based on its design, shape and space.
- crossover_car_and_van: this is a boolean field that indicates whether the vehicle is a crossover between a car and a van.

Columns	Dtype
public reference	int64
mileage	float64
reg_code	object
standard_colour	object
standard_model	object
standard_make	object
vehicle_condition	object
year_of_registration	float64
price	int64
body_type	object
crossover_car_and_van	bool
fuel_type	object

Analysis of Distributions

Data columns that contain numerical values that can be used to measure or quantify a certain aspect of the data are referred to as quantitative columns, also known as numerical columns or continuous variables. These columns can be further classified into two groups: discrete and continuous, and they can take on any value within a continuous range.

Values that are different and distinct from one another, such as whole numbers, are contained in discrete quantitative columns. On the other hand, continuous quantitative columns, like decimal numbers, contain values that can take on any value within a continuous range.

Quantitative columns
public_reference
mileage
year_of_registration
price

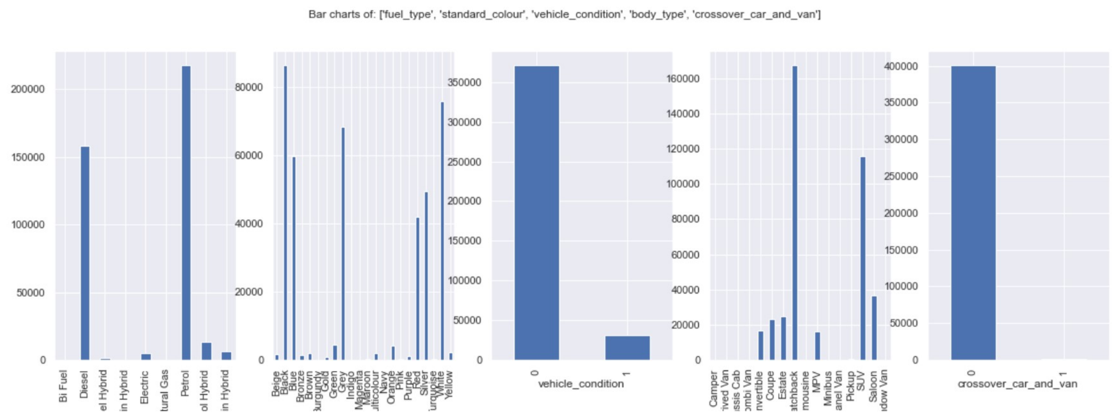
Describe ()

	public_reference	mileage	year_of_registration	price
count	4.020050e+05	401878.000000	368694.000000	4.020050e+05

mean	2.020071e+14	37743.595656	2015.006206	1.734197e+04
std	1.691662e+10	34831.724018	7.962667	4.643746e+04
min	2.013072e+14	0.000000	999.000000	1.200000e+02
25%	2.020090e+14	10481.000000	2013.000000	7.495000e+03
50%	2.020093e+14	28629.500000	2016.000000	1.260000e+04
75%	2.020102e+14	56875.750000	2018.000000	2.000000e+04
max	2.020110e+14	999999.000000	2020.000000	9.999999e+06

Qualitative columns, also known as categorical columns are data columns that contain values that represent categories or groups. These columns can take on a limited number of possible values, and the values are usually not numerical. Qualitative columns are often used to represent characteristics or features of the data that cannot be measured or quantified numerically.

Categorical Columns
standard_colour
standard_make
standard_model
vehicle_condition
body_type
crossover_car_and_van
fuel_type

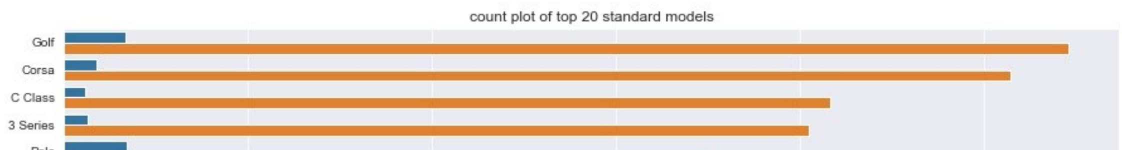


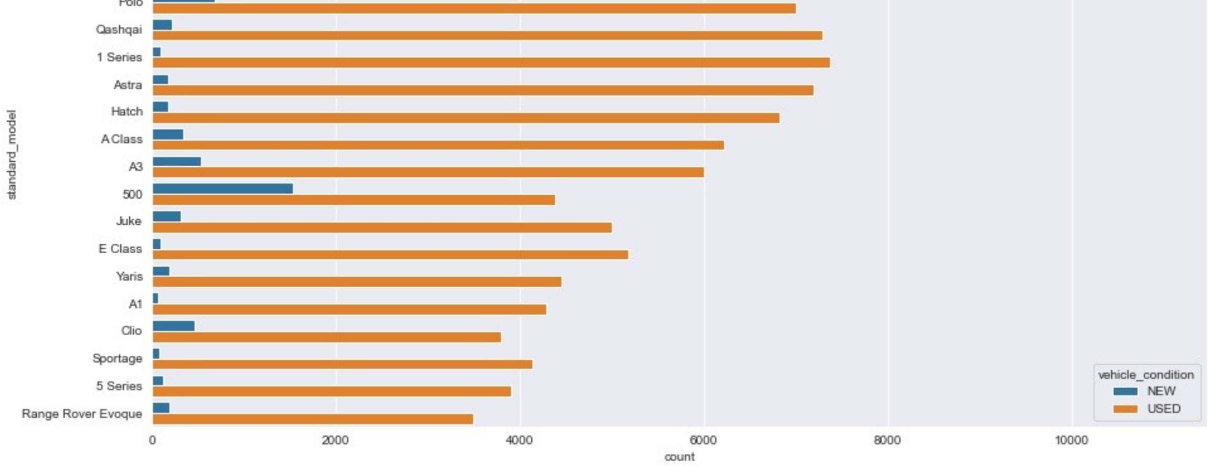
Bar charts to see how the data is distributed for these categorical columns.

Describe()

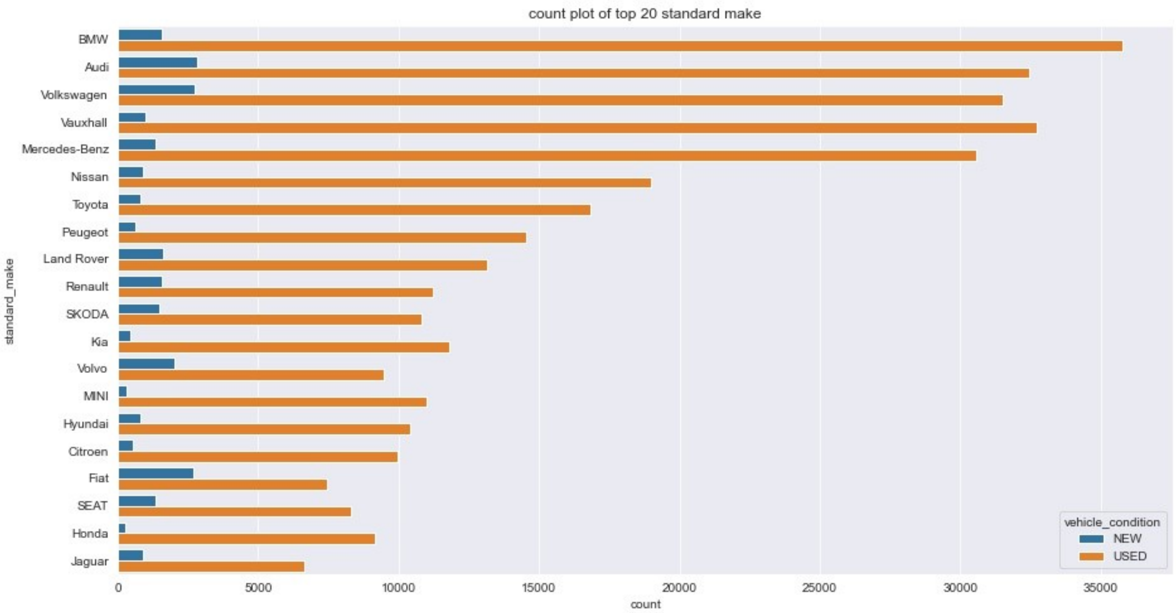
	standard_colour	standard_make	standard_model	vehicle_condition	body_type	crossover_car_and_van
count	396627	402005	402005	402005	401168	402005
unique	22	110	1168	2	16	2
top	Black	BMW	Golf	USED	Hatchback	False
freq	86287	37376	11583	370756	167315	400210

From the above, i can see that for standard colour, the Black colour occured the most. For Standard make, BMW comes top. For Vehicle condition, we have more of used cars and for fuel type, we have more of Petrol using cars. For standard model, Golf occurs the highest

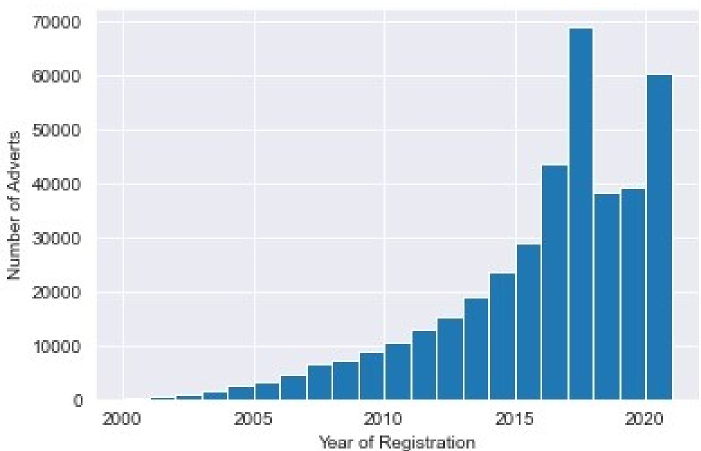




The above plot shows us maximum adverts of cars of different models. Therefore from above plot we can see that adverts number of cars models are more from Golf Model followed by Corsa till Range Rover evoque from first 20 most of standard model



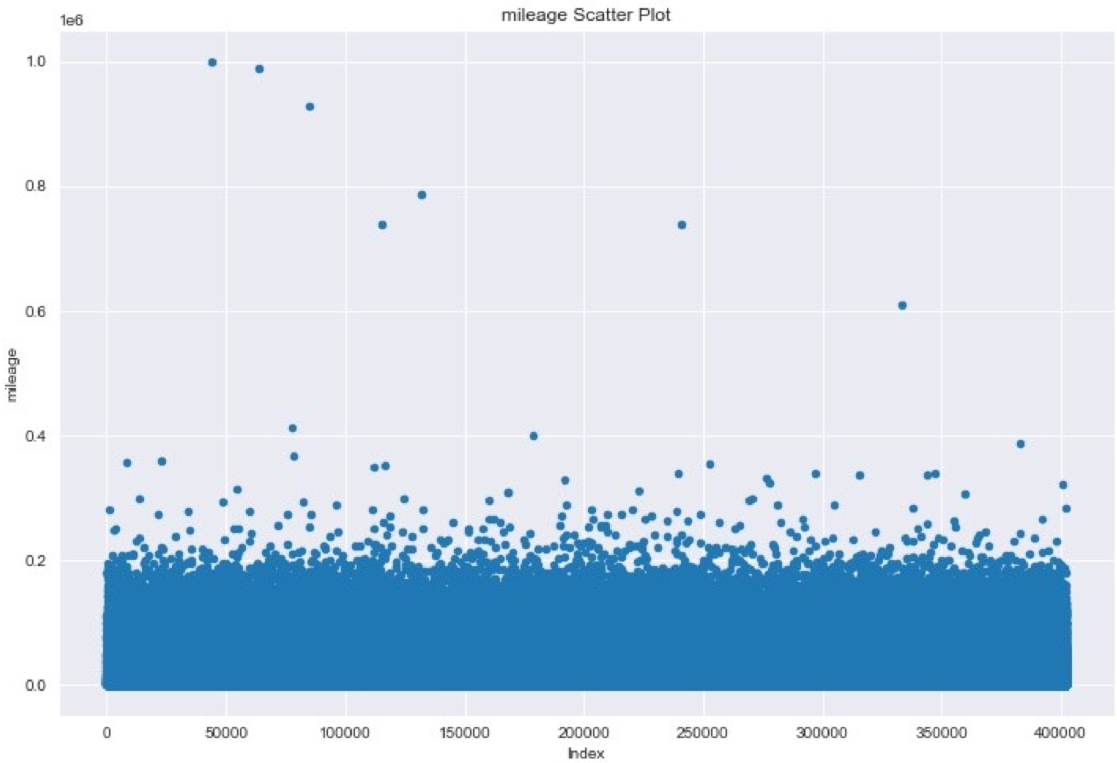
A plot of the top 20 standard car make shows BMW comes first

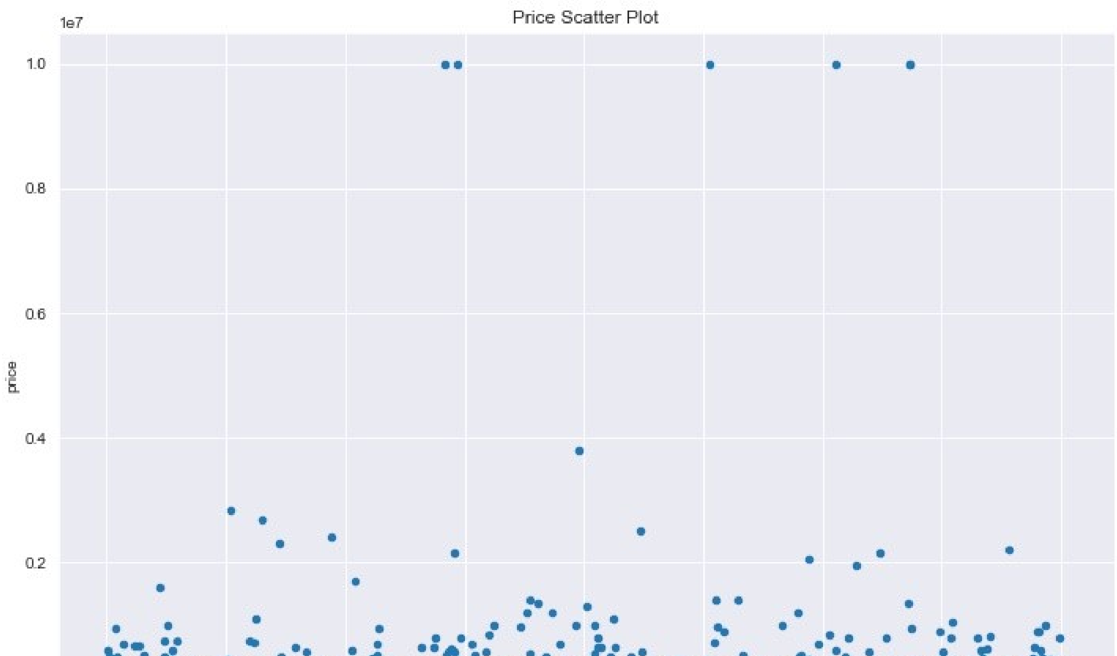


The above plot shows us adverts of cars in respective years from about 2000 to 2021. Therefore from above plot we can see that adverts number of cars was highest in 2017.

ANALYSIS OF UNIVARIATE DISTRIBUTION

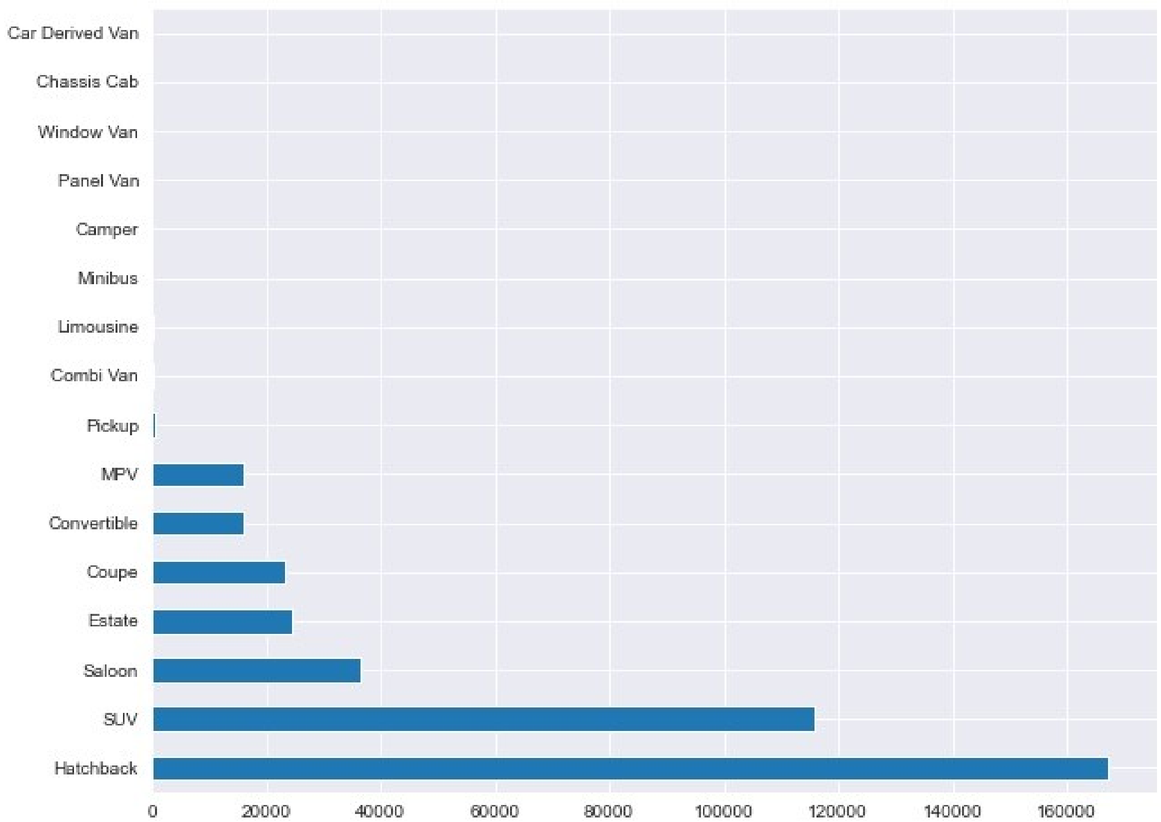
The scatter plot below shows the different distributions for the price and mileage column. The extending points are outliers.





Analysis of distribution in the body_type column

> Charts shows we have more of hatchbacks in the adverts followed by SUV



Analysis of Predictive Power of Features

ANOVA (Analysis of Variance) is a statistical method that can be used to examine the propensity of features in a model to predict outcomes. It is utilised to investigate the null hypothesis that the means of various groups are comparable. By comparing the means of the result variable for various levels of the feature, an ANOVA can be used to assess if a particular feature has a significant effect on the outcome variable in the context of feature selection. It is likely that the feature has a good capacity for prediction if the means for the various levels of the feature are significantly different. When features are chosen or eliminated based on their p-values.

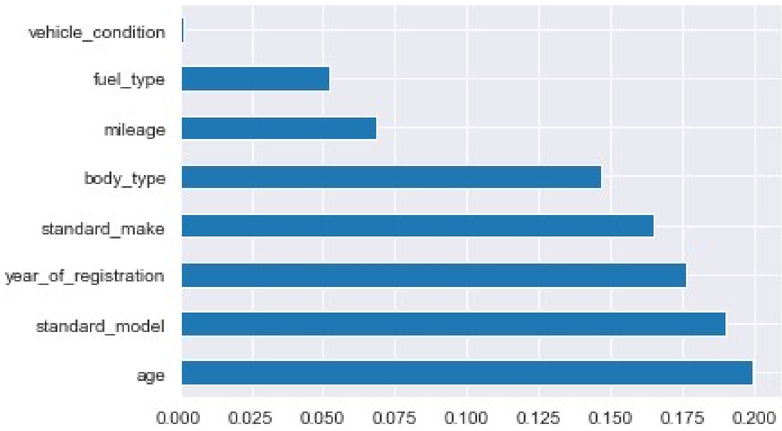
COLUMNS	ANOVA RESULTS
standard model	P-Value: 6.640052996047211e-68
standard make	P-Value: 3.879545153649509e-39
vehicle condition	P-Value: 1.1157287635872366e-28
body type	P-Value: 1.514409583672064e-40
fuel_type	P-Value: 4.856296225271252e-16
standard colour	P-Value: 0.0373904421915285

crossover_car_and_van P-Value: 0.16184389635027194

The standard model, standard make, vehicle condition, body style, and fuel type all seem to be connected with the price of the car, according to the ANOVA results. This indicates that these factors affect the vehicle's pricing in a statistically significant way. However, as the p-value is higher than the 0.05 cutoff, it does not appear that the crossover car and van group has a statistically significant effect on the cost of the vehicle. It's important to note that the standard colour category also has a high p-value, indicating that while it may have some influence on the pricing, it is not as powerful as the other predictors found by the study.

- Overall, these results suggest that these identified categorical variables should be included in the model as they have a statistically significant impact on the price of a vehicle.

Random Forest Feature Importance



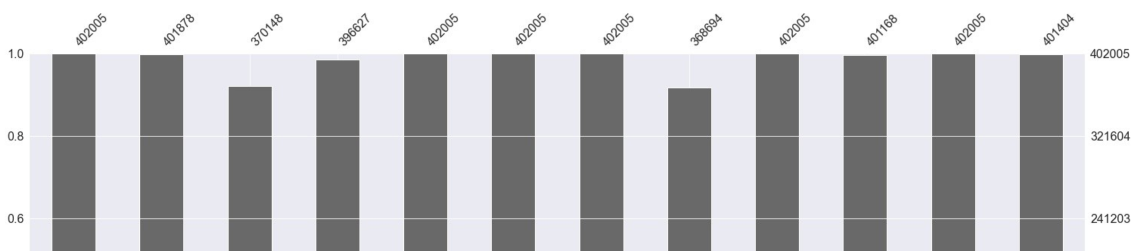
> This also Analyses the predictive power of features.

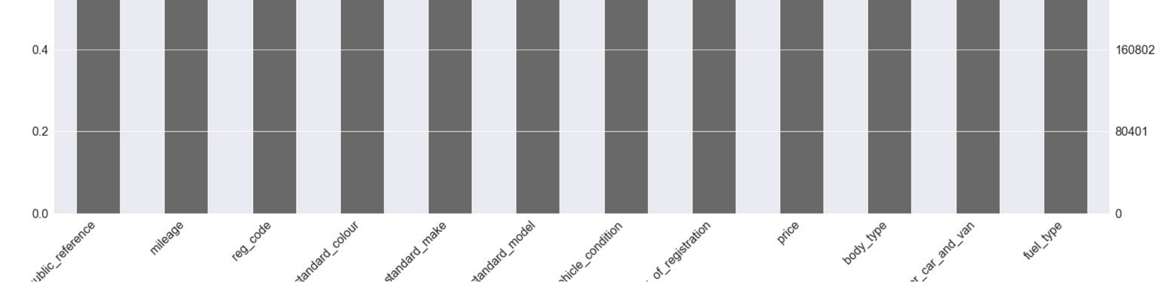
1.2. Identification/Commenting on Missing Values

Columns	No of missing values
mileage	127
reg_code	31857
standard_colour	5378
year_of_registration	33311
body_type	837
fuel_type	601

We cannot Ignore the missing values, because the percentage of missing values is not small and the missing values are not randomly distributed throughout the dataset. Ignoring may introduce bias into results as the missing values are not truly random.

- Visual representation of the missing data in the dataset



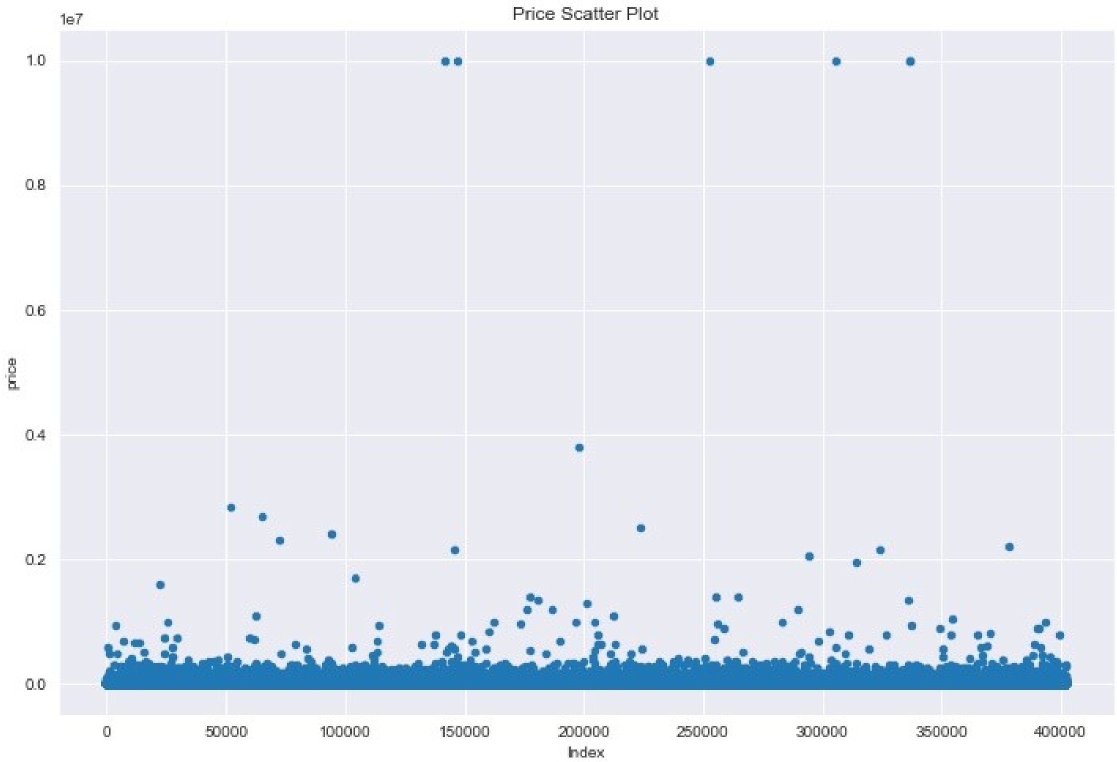


1.3. Identification/Commenting on Outliers and Noise

Outliers are extreme values in the data which are far away from most of the values.

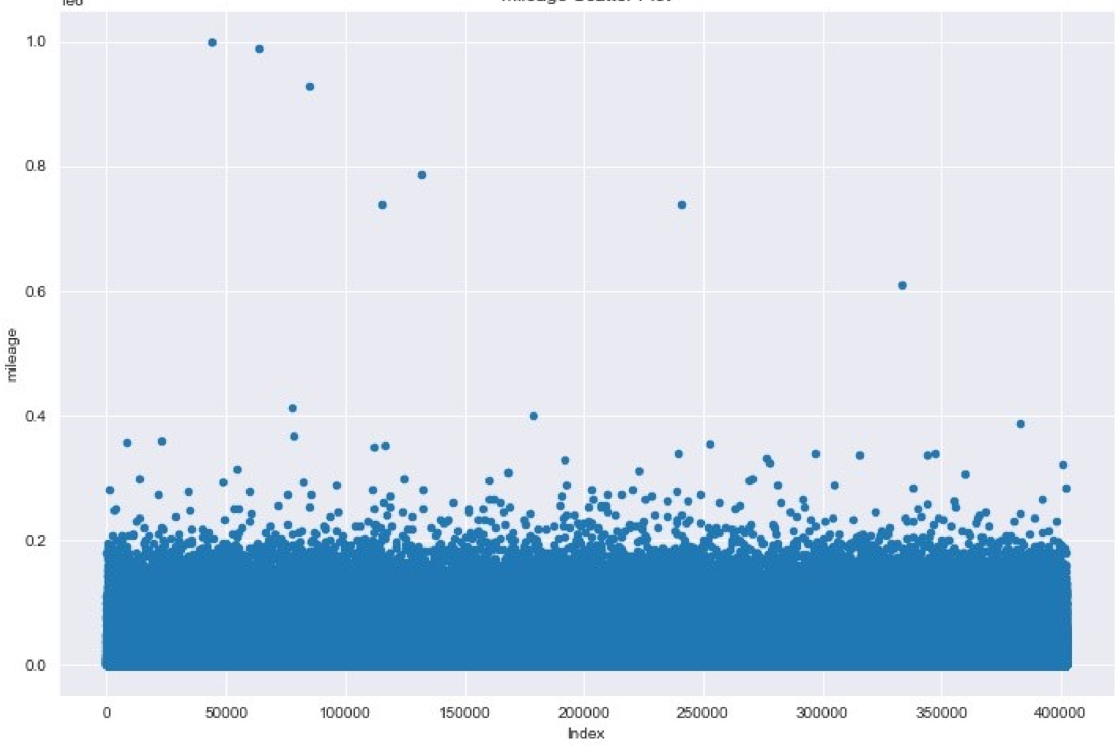
Outlier must be treated one column at a time. As the treatment will be slightly different for each column.

- scatter plots to identify outliers.



There are a number of factors that could contribute to outliers in the price of cars in this dataset. Some possible explanations include:

- The cost of new autos can be greater.
- The automobile's brand and model: Some car brands and models may be more well-liked or in high demand, which could lead to a price increase.
- The car's options and features: Cars with more options or features may cost more than those with fewer options or features.
- The location of the automobile's sale: Depending on the local economy or the demand for particular car models, the price of a car may fluctuate depending on the location where it is being sold.
- The seller: The seller may have an impact on a car's price.
- Pricing errors or mistakes: Outliers in price may be the result of pricing errors or mistakes, such as if the price listed is significantly higher or lower than the typical price for that make and model of car.



Outliers in mileage data may occur for a variety of reasons. Some possible explanations for outliers in mileage include:

- Due to factors like the owner's driving habits or the car's use, the vehicle may have been driven much more or less than the average vehicle (e.g. for commuting or for long distance travel).
- Higher mileage could be the result of the vehicle being operated in a way that was particularly difficult on the engine or other parts.
- The vehicle's mileage may have been omitted or erroneously recorded.

2. Data Processing for Machine Learning

2.1. Dealing with Missing Values, Outliers, and Noise

Columns	No of missing values
mileage	127
reg_code	31857
standard_colour	5378
year_of_registration	33311
body_type	837
fuel_type	601

Missing values are treated for each column separately.

- **Filling missing values in the standard_colour column**

Standard_colour
Grey
Blue
Brown
Red
Bronze

Standard_colour
Black
White
Silver
Purple
Green
Orange
Yellow
Turquoise
Gold
Multicolour
Beige
Burgundy
Pink
Maroon
Magenta
Navy

> The above are the unique values in the standard colour column after filling Nans

For each group of rows in the DataFrame that have the same values in the standard make and standard model columns, the missing values are filled in using the mode of the standard colour column. The standard make and standard model columns of the DataFrame are first grouped, and for each group, the mode of the standard colour column is computed. The first non-null value in the mode is utilised as the mode value if there are any. If the mode value is not null, it is then utilised to fill in the blanks in the datafram's standard colour column. The changed dataframe is then given back.

- **Filling missing values in the body_type column**

I used a function to fill the nans in the body_type column. The function groups the dataframe by standard_make, and for each group, it computes the mode of the body_type column. It then selects only the non-null values in the mode and uses the first one as the mode value. If the mode value is not null, the function fills null values in the body_type column with the computed mode. Finally, the function returns the modified dataframe.

	body_types_unique_val
0	SUV
1	Saloon
2	Hatchback
3	Convertible
4	Limousine
5	Estate
6	MPV
7	Coupe
8	Pickup
9	Combi Van
10	Panel Van
11	Minibus
12	Window Van
13	Camper

body_types_unique_val

Filling missing values in the fuel_type column

The mode value (i.e., the most frequently occurring value) in the fuel type column is used to fill the Nans

> **The year of registration had values that were wrongly inputed to assume. I created a dictionary to replace them all with appropriate values**

Original	Replacement
999	1999
1006	2006
1007	2007
1008	2008
1009	2009
1010	2010
1015	2015
1016	2016
1017	2017
1018	2018
1063	1963
1515	2015
1909	1999
1933	1953

Filling missing values in the year_of_registration column

An assumption is made here, that for vehicle that are new, and the year of registration and regcode values are Nan, the year of registration is 2021 and the corresponding reg_code value is assigned. So, the function below is used to fill

```
# Fill year_of_registration and regcode columns with 2021 and 21, respectively,
#where vehicle_condition is 'new' and year_of_registration is nan
def fill_year_of_registration(df):
    # Fill year_of_registration column with 2021 where vehicle_condition is 'new' and year_of_registration is nan
    df.loc[(df['vehicle_condition'] == 'NEW') & (df['year_of_registration'].isnull()),
           ['year_of_registration', 'reg_code']] = [2021, 21]
    return df
```

I loaded the Car year identifier and age identifier csv file into my notebook in my quest to fill the remaining Nans in my year of registration column

> Two dictionaries were created from the data in the age_identifier dataframe and then using those dictionaries to fill missing values in the year_of_registration column

The code is filling missing values in the year_of_registration column of the adverts DataFrame by using two dictionaries that are created from the age_identifier DataFrame and the reg_code column of the adverts data frame

```
# create dictionaries from the age identifier dataframe
first_col = dict(zip(age_identifier['1 March - 31 August'],age_identifier['Year']))
second_col = dict(zip(age_identifier['1 September - 28/29 February'],age_identifier['Year']))
```

```
second_col = dict(zip(age_identifier['1 September - 28/29 February'],age_identifier['Year']))
```

I created another dictionary by using the zip() function and two columns from the Pandas DataFrame year_identifier. The dictionary maps the values in the 'Letter' column to the corresponding values in the 'Year' column for rows 21 to 41 of the DataFrame.

```python
year_dict = dict(zip(year_identifier.loc[21:41, 'Letter'], year_identifier.loc[21:41, 'Year']))

# fillings
adverts['year_of_registration']= adverts['year_of_registration'].fillna(adverts['reg_code'].astype(str).map(year_dict))
```

Columns	No of missing values
year_of_registration	330

Checking, i still have my number of missing values in the year of registration column as 330

```python
def fill_year_of_reg_col_nans(df):
    # group the dataframe by make and model
    grouped = df.groupby(['standard_make', 'standard_model'])

    # for each group, compute the mode of the 'year_of_registration' column
    for name, group in grouped:
        # compute the mode of the 'year_of_registration' column
        mode = group['year_of_registration'].mode()

        # only consider non-null values in the mode
        non_null_mode = mode[mode.notnull()]

        # if there are any non-null values, use the first one as the mode
        if non_null_mode.empty:
            mode_value = None
        else:
            mode_value = non_null_mode.iloc[0]

        # if the mode value is not NaN, fill in the missing values in the 'year_of_registration' column
        # with the computed mode value using the fillna() method
        if not pd.isnull(mode_value):
            df['year_of_registration'].fillna(value=mode_value, inplace=True)

    return df
```

The function works by grouping standard_make and standard_model, and then for each group, it computes the mode of the year_of_registration column. If there are no non-null values in the mode, the mode value is set to None. If there are any non-null values, the first one is used as the mode value. Finally, the function uses the fillna() method to fill in the missing values in the year_of_registration column with the computed mode value, if the mode value is not NaN.

Filling missing values in the mileage column

The missing values in the mileage column is filled with median of the column

```python
def fill_nan_in_mileage(data, col):
    median = np.nanmedian(data[col])
    data[col].fillna(median, inplace=True)
```

Dealing with the Nans in the Regcode column

I will drop my reg_code columnn because it will serve the same function as the year_of_registration in the dataset

<u>Treating outliers in the price and mileage column</u>

Why I should treat the outliers?

Outliers bias the training of machine learning models. As the algorithm tries to fit the extreme value, it goes away from majority of the data.

There are below two options to treat outliers in the data.

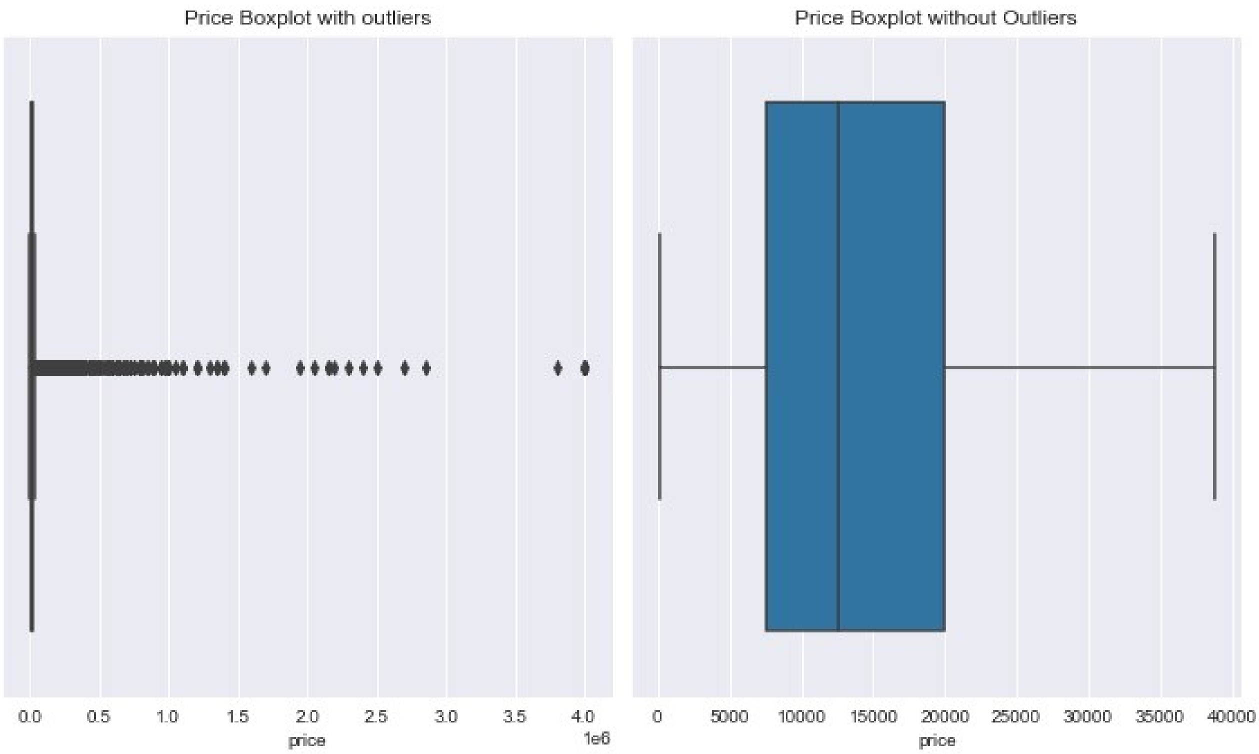
- Option-1: Delete the outlier Records. Only if there are just few rows lost.
- Option-2: Impute the outlier values with a logical business value

Finding out the most logical value to be replaced in place of outliers

```
[293]: adverts['price'][adverts['price']<5000000].sort_values(ascending=False)
```

```
[293]: 198060    3799995
       51741     2850000
       64910     2695000
       223835    2500000
       94033     2400000
                   ...
       91878         200
       109133        180
       303316        150
       300445        122
       332532        120
       Name: price, Length: 401999, dtype: int64
```

Based on the above output, the nearest logical value is 3799995, hence, replacing any value greater than 5000000£ with 4000000£



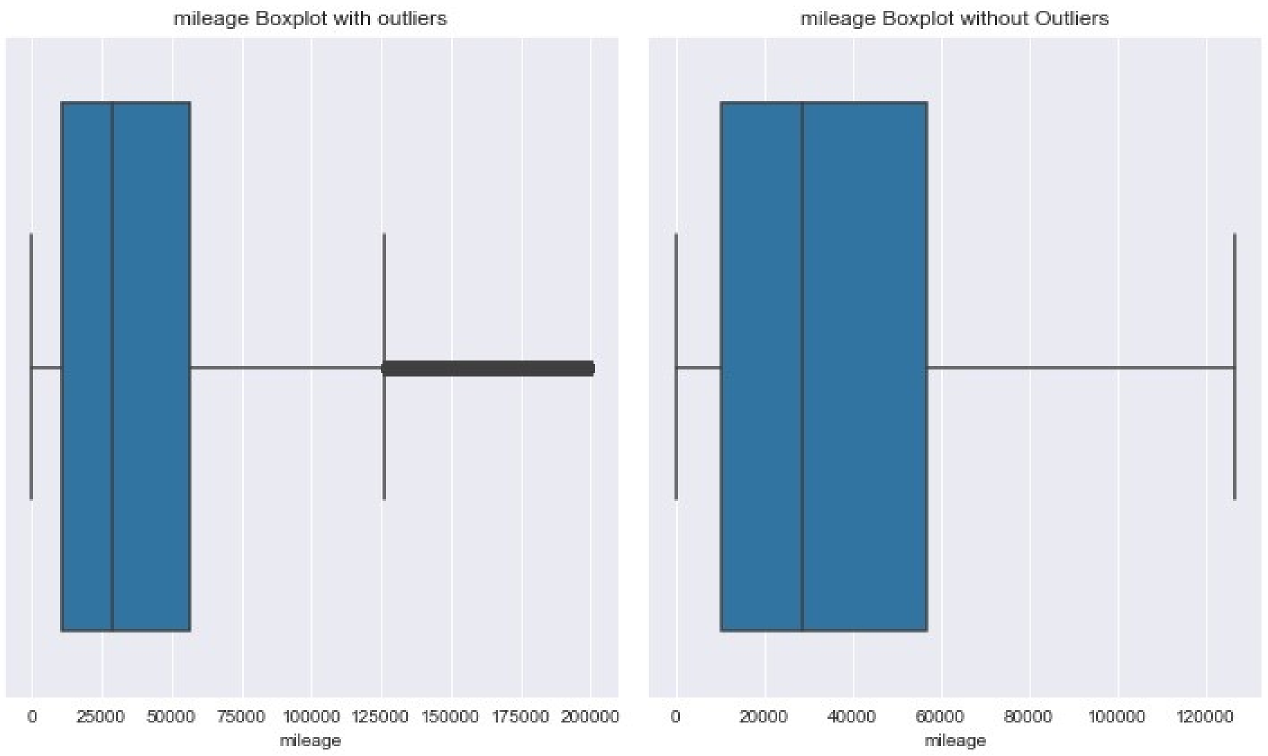
I proceeded to remove treat the outliers in price using the quartile range 0.90

```python
def drop_outliers_IQR(df, column):
    # Calculate the first and third quartiles
    Q1, Q3 = df[column].quantile([0.0, 0.90])

    # Calculate the interQuartile range
    IQR = Q3 - Q1

    # Calculate the lower and upper bounds
    lower_bound = Q1 - (1.5 * IQR)
    upper_bound = Q3 + (1.5 * IQR)

    # Return the data without outliers
    return df[(df[column] >= lower_bound) & (df[column] <=upper_bound)]
```

```python
adverts=drop_outliers_IQR(adverts, 'price')
```

Mileage

I used 200,000 as my logical value for mileage.



2.2. Feature Engineering, Data Transformations (2-3)

1. AGE:

Age of a car is calculated as (current year- year of registration)

- Current year is set to 2021 as that is the max year in the year of registration column

year_of_registration	age
2011	10
2017	4
2016	5
2015	6

This gives us the age of the vehicle as a new column

Data transformation

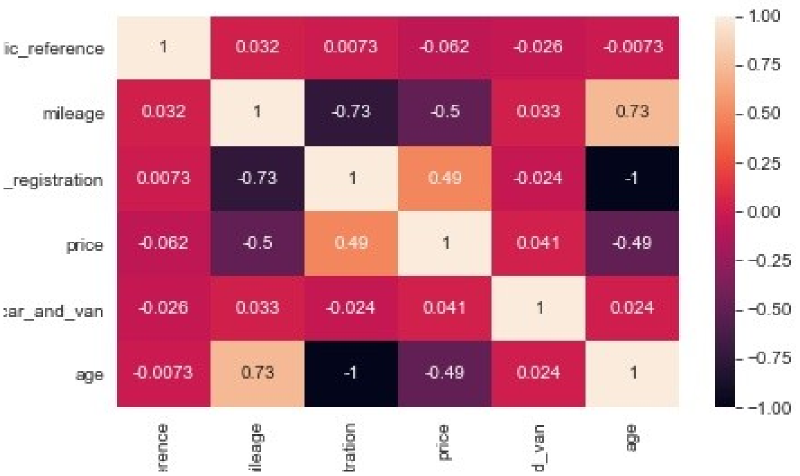
- Data transformation on the vehicle condition column and crossover_car_and_van column

```
adverts['vehicle_condition']=adverts['vehicle_condition'].map({"USED":0, "NEW":1})
```

Vehicle_condition	crossover_car_and_van
1	0
0	0
0	0
0	0
0	0
0	0

> **This will return a transformed column with numeric values for the condition 'USED', and 'NEW' same thing applies for the crossover_car_and_van column. Snippet above confirms**

Feature Selection



I used the correlation heat map to show correlation between numerical variable and target(price)

- From this heat map, we can conclude Mileage is correlated, same with year of registration and car age

> This confirms our correlations

Analysis of variance(ANOVA) is performed to check if there is any relationship between the TARGET variable and categorical variable

If the ANOVA P-Value is <0.05, that means we reject H0

Standard colour has the least correlation with target.

COLUMNS	ANOVA RESULTS
standard model	P-Value: 6.640052996047211e-68
standard make	P-Value: 3.879545153649509e-39
vehicle condition	P-Value: 1.1157287635872366e-28
body type	P-Value: 1.514409583672064e-40
fuel_type	P-Value: 4.856296225271252e-16
standard colour	P-Value: 0.0373904421915285
crossover_car_and_van	P-Value: 0.16184389635027194

Correlated Columns
standard_make
standard_model
vehicle_condition
body_type
fuel_type
standard colour

- crossover _car_and_van is not correlated with price and standard colour has low correlation

Model Building

- 3.1 Algorithm Selection, Model Instantiation and Configuration (1-2)

My base model is the Multiple linear regression and i got a low score for R2. It could be because of the following reason

- Non-linearity: MLR considers the connection between the predictor and outcome variables to be linear. If this presumption is incorrect, the model might not adequately match the data, which would lead to a poor R-squared score.
- Overfitting is a possibility with MLR models because of their sensitivity to outliers and significant observations. On the other hand, because random forest models create many decision trees and average the predictions, they are less prone to overfitting.

```
print('R2 Value:',metrics.r2_score(y_test, LREG.predict(X_test)))
```

```
LinearRegression()
R2 Value: 0.3914699753504456
```

```
Accuracy_Values=cross_val_score(RegModel, X , y, cv=10, scoring=custom_Scoring)
print('\nAccuracy values for 10-fold Cross Validation:\n',Accuracy_Values)
print('\nFinal Average Accuracy of the model:', round(Accuracy_Values.mean(),2))
```

```
##### Model Validation and Accuracy Calculations ##########
   price  Predictedprice
0  16995         18861.0
1   4000          3640.0
2   6980         12219.0
3   7995         14700.0
4  32750         23829.0
Mean Accuracy on test data: 40.43084852736895
Median Accuracy on test data: 61.348822200709904

Accuracy values for 10-fold Cross Validation:
 [40.67799608 40.27975459 40.41143213 40.60288347 40.20633509 40.37351876
  40.38946937 39.63680722 40.75704778 40.4196656 ]
```

```
 40.38048027 39.62680722 40.75794778 40.4196656 ]

Final Average Accuracy of the model: 40.37
```

The Final average Acccuracy of my base model is 40.37. I will compare the model with other model; Random forest and XGboost

Random Forest

```
RandomForestRegressor()
R2 Value: 0.9342675759018028
```

```
RandomForestRegressor()
R2 Value: 0.9342675759018028

##### Model Validation and Accuracy Calculations ##########
   price  Predictedprice
0  16995         15441.0
1   4000          5230.0
2   6980          6022.0
3   7995          7678.0
4  32750         31114.0
Mean Accuracy on test data: 85.56667406189575
Median Accuracy on test data: 90.8811748998665

Accuracy values for 10-fold Cross Validation:
 [85.68891932 85.50633648 85.84302316 85.70271313 85.76896267 85.88204705
 85.83823259 85.57020347 85.5179005  85.84605041]

Final Average Accuracy of the model: 85.72
```

A high R-squared value of 0.93 for a random forest model is generally a good indication that the model is able to explain a large amount of the variation in the target variable.

Cross-validation is done, which will give me a better estimate of the model's performance on unseen data

My cross validation score is 85.72

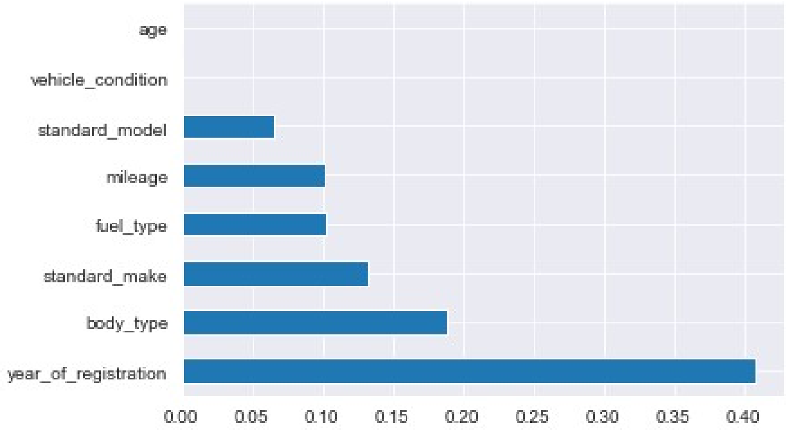
XGBOOST

```
R2 Value: 0.873497364286579
```

```
Accuracy values for 10-fold Cross Validation:
 [78.67165121 78.37803958 78.38872801 78.87623591 78.43376697 78.65596438
 78.45123265 78.51309797 79.14436379 78.61492074]

Final Average Accuracy of the model: 78.61
```

```
##### Model Validation and Accuracy Calculations ##########
   price  Predictedprice
0  16995         16419.0
1   4000          7077.0
2   6980          6659.0
3   7995          9381.0
```

```
4  32750          28790.0
Mean Accuracy on test data: 78.50169563293457
Median Accuracy on test data: 85.8205394744873
```



The model might not be as intricate as the random forest, and as a result, it could not be able to account for as much variation in the target variable.

3.2. Grid Search, and Model Ranking and Selection

GRID SEARCH ON RANDOM FOREST

A grid search using a Random Forest Regressor model to optimize the parameters of the model. It's using R2 score as the evaluation metric, which is a commonly used metric to evaluate the performance of regression models.

The grid search is searching through two parameters: 'max_depth' and 'n_estimators'. The possible values for 'max_depth' are [5, 10, 15], and the possible values for 'n_estimators' are [50, 100]. The grid search trainS a Random Forest model for each combination of these parameters and evaluate the model's performance using R2 score.

```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer, r2_score

# Define the custom scoring function for R2 score
def custom_scoring_function(y_true, y_pred):
    return r2_score(y_true, y_pred)

# Create the custom scoring function
custom_scorer = make_scorer(custom_scoring_function)

# Define the parameter grid
param_grid = {'max_depth': [5, 10, 15],
              'n_estimators': [50, 100]}

# Create the grid search object
grid_search = GridSearchCV(RandomForestRegressor(), param_grid, cv=5, scoring=custom_scorer)

# Fit the grid search to the data
grid_search.fit(X, y)

# Print the best parameters and the best score
print("Best Parameters: ", grid_search.best_params_)
print("Best Score: ", grid_search.best_score_)
```

```
Best Parameters:  {'max_depth': 15, 'n_estimators': 100}
Best Score:  0.9352734466641008
```

The grid search was done on just random forest because of the computational power of my computer.

Model ranking and selection

Model ranking and selection is the process of comparing different machine learning models and selecting the best one based on their performance

- My best model here is the Random forest after the grid search i still have a high r2 score of 0.9352734466641008.
- The Knn is doing well also. but the Final Average Accuracy of my random forest model supercedes my knn model.

Final Average Accuracy of the model: 85.72(Random Forest) Final Average Accuracy of the model: 84.83(KNN)

4. Model Evaluation and Analysis

- **4.1. Coarse-Grained Evaluation/Analysis (1-2) (e.g., with model scores)**

- The grid search is a coarse-grained evaluation method

Coarse-Grained Evaluation	R2 SCORE AFTER GRID SEARCH
Random Forest Regressor	0.9352734466641008

Based on the results of the grid search, it appears that the best set of parameters for the random forest model is a max_depth of 15 and n_estimators of 100. This combination of parameters resulted in an R2 score of 0.94, which is considered to be a high value and indicates that the model is able to explain 93% of the variability in the target variable.

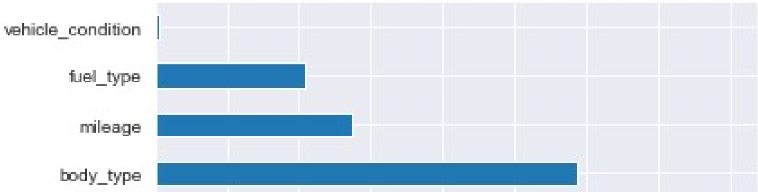
It is worth noting that the R2 score is a commonly used metric for evaluating the performance of regression models, and values close to 1 indicate that the model is able to accurately predict the target variable. A high R2 score such as 0.93 indicates that the model is able to make accurate predictions, and that the difference between the predicted values and the true values of the target variable is small.

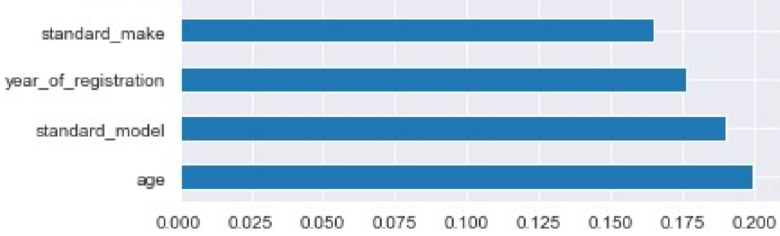
Metric	Value
Mean Accuracy	85.56696472307145
Median Accuracy	90.8868778280543

- It is worth noting that these metrics are calculated using the absolute percentage error (APE) and they are affected by outliers sometimes
- In conclusion, the model has a Mean Accuracy of 85.57% and a Median Accuracy of 90.89%. These values suggest that the model is making accurate predictions

- **4.2. Feature Importance**

A FEATURE IMPORTANCE PLOT FROM RANDOM FOREST



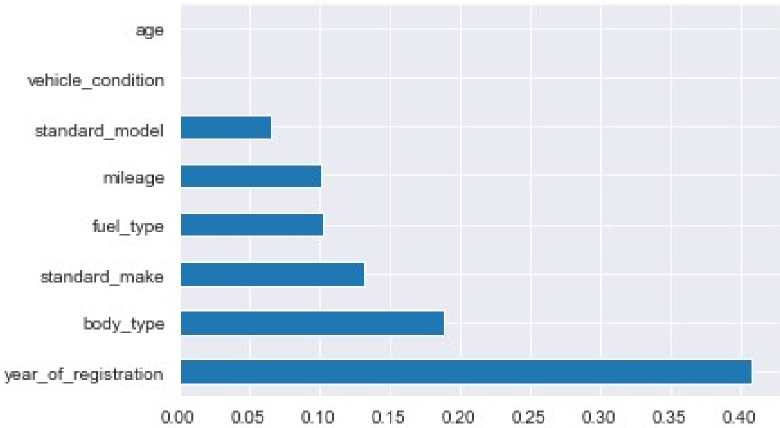


The age of the car and its standard model are the two factors that have the most impact on a car's price, as indicated by the feature importance plot. The feature importance plot, which the random forest model creates, displays the relative weights each feature has in affecting the target variable.

The car's age is the most crucial component because it has the greatest feature priority score. This shows that there is a significant relationship between the age of the car and its price. It makes sense that older vehicles would cost less than newer ones.

- The second most crucial element is the car's standard model, which has a high relevance value.
- Standard make also has a relatively high importance value, though not as high as age and standard model. This suggests that the make of the car also has some influence on the price of the car. This is also intuitive as different car makers tend to have different prices for their cars.

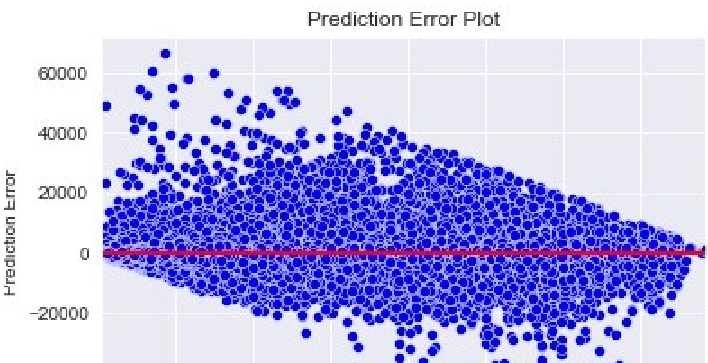
> A Feature importance plot from Xgboost



The Xg boost has got different features as its best, different from the random forest feature importance plot. The Year of registration comes first here

The year of registration can influence car prices, as the age of the car is one of the key factors that determine the price of a used car. Generally, newer cars will have a higher price than older cars. This is because newer cars typically have lower mileage, have been subject to fewer repairs, and have a more modern design. Additionally, cars that are newer are likely to be equipped with more advanced safety features and technologies, which can also affect their price.

- **Fine-Grained Evaluation (1-2) (e.g., with instance-level errors)**



In the prediction error plot, the line that runs through the middle of the points (y=0) represents the average difference between the predicted values and the true values. If the points are randomly distributed around this line, it means that the model is making accurate predictions on average.

In this case, since the line runs through the middle of the points, it indicates that the model is making accurate predictions on average. This is a good indication that the model is performing well.

CONCLUSION

Due to the fact that i removed outliers in price using the 90% percentile,. my models will not be able to predict price of cars in that range. And Random forest seems very much my best model here. It's accuracy is highest.