Course Title

Machine Learning Concepts (6G7V0015_2223_9F)

Assignment Title

Machine Learning Project

Full Name

Adebo Dolapo Kingsley

MMU ID

22539980

Table of Content

1. Introduction

This is a Coursework project for Machine Learning Concepts. 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/Domain 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

Descriptive Statistics of Categorical Features

	reg_code	standard_colour	standard_make	standard_model	vehicle_condition	body_type	fuel_type
count	370148	396627	402005	402005	402005	401168	401404
unique	72	22	110	1168	2	16	9
top	17	Black	BMW	Golf	USED	Hatchback	Petrol
freq	36738	86287	37376	11583	370756	167315	216929

Analysis of Univariate 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.

2.1 Analysis of Predictive Power of Features

Using `ppscore` library to get the predictive power of features with the target variable(`price`).

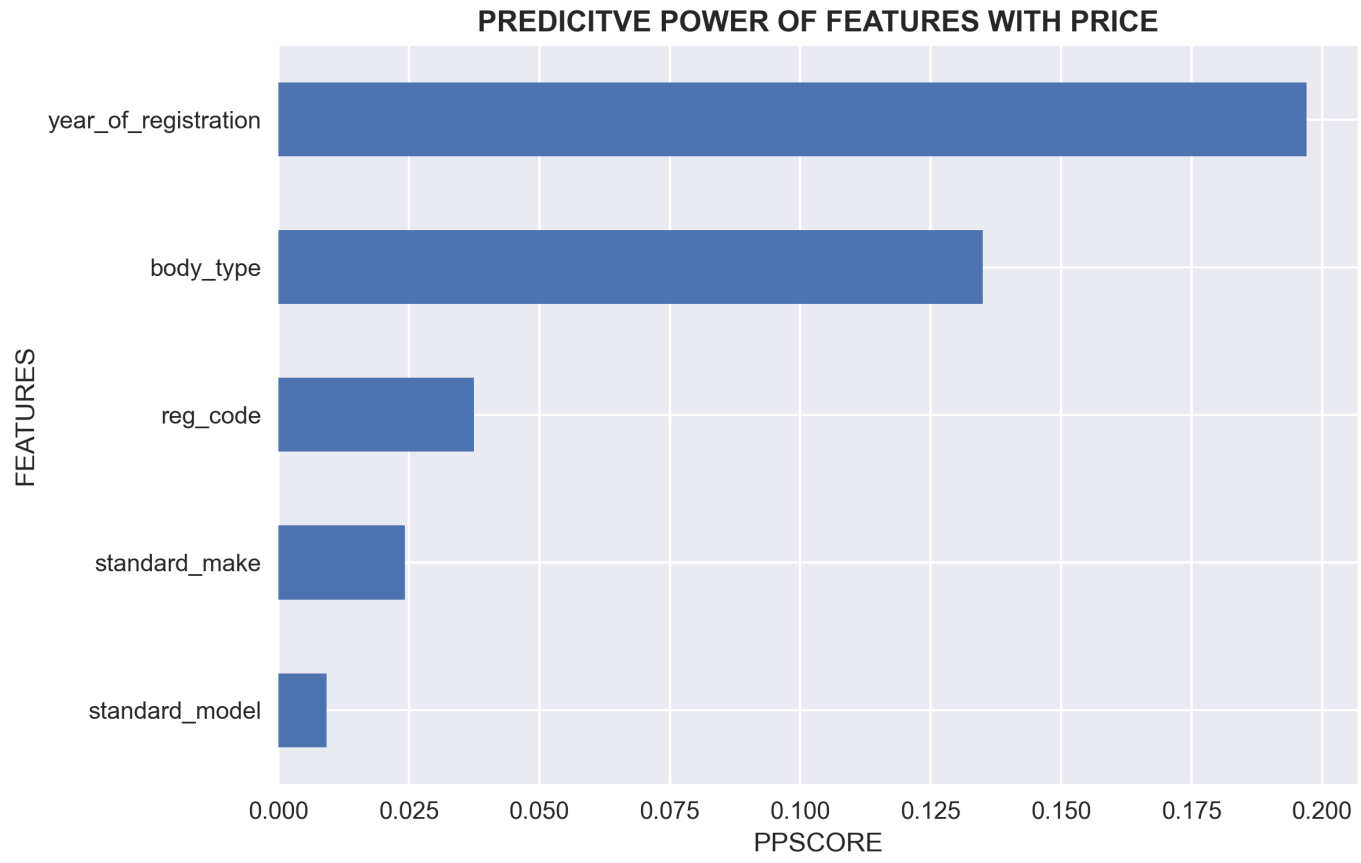

Fig.2 Analysis of Predictive Power of Features)

`Year_of_registration`,and `body_type` both have a predictive power score of over **0.1**. Features like `mileage`, `fuel_type`, and `vehicle_condition` have a score of **0**. It's difficult at this stage to agree that these features have no relationship with price. However, let's continue with the pre-processing.

2.3 Data Processing for Data Exploration and Visualisation

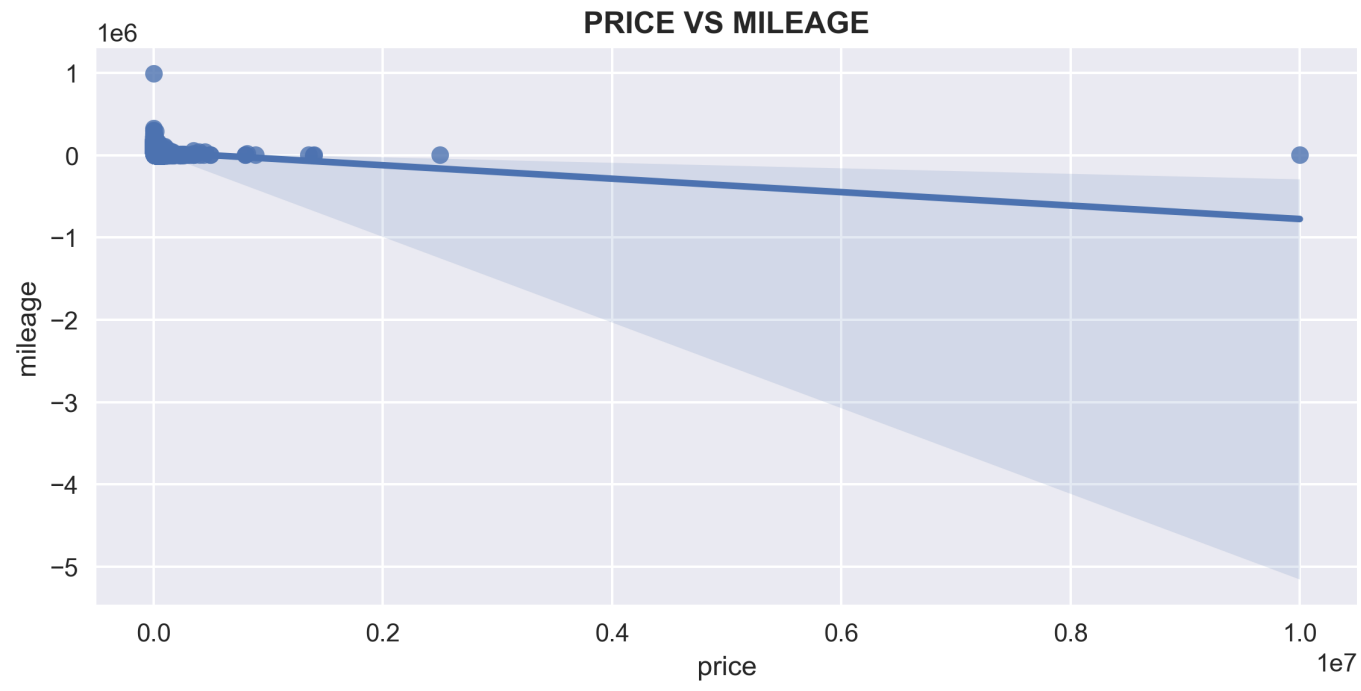

Fig.3 Price vs Mileage

From the plot above, `mileage` is inversely proportional to `price`, i.e as `mileage` increase, there's a corresponding decrease in the price of a car. In a few cases, where `mileage` is low and the price of the car is high, these cars are luxurious/vintage cars.

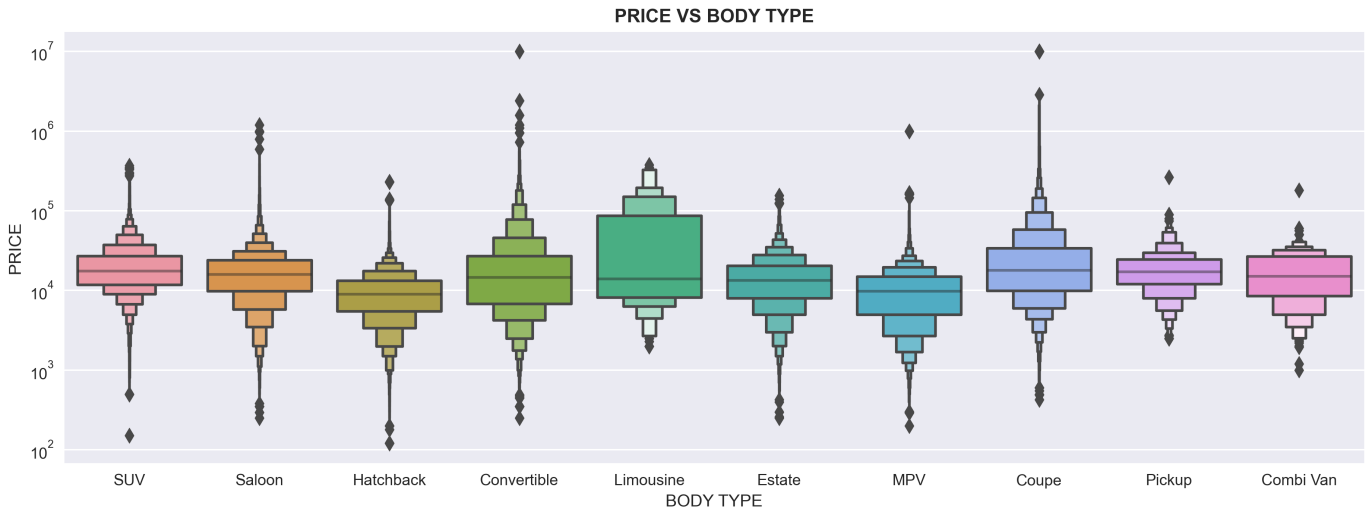

Fig.4 Price vs Body Type

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.

3. Data Processing for Machine Learning

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 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 `Year_of_registration` for years less than **1900** with **2013** because the `reg_code` for those observations was mapped to the year 2013.

Dealing with Outliers and Noise

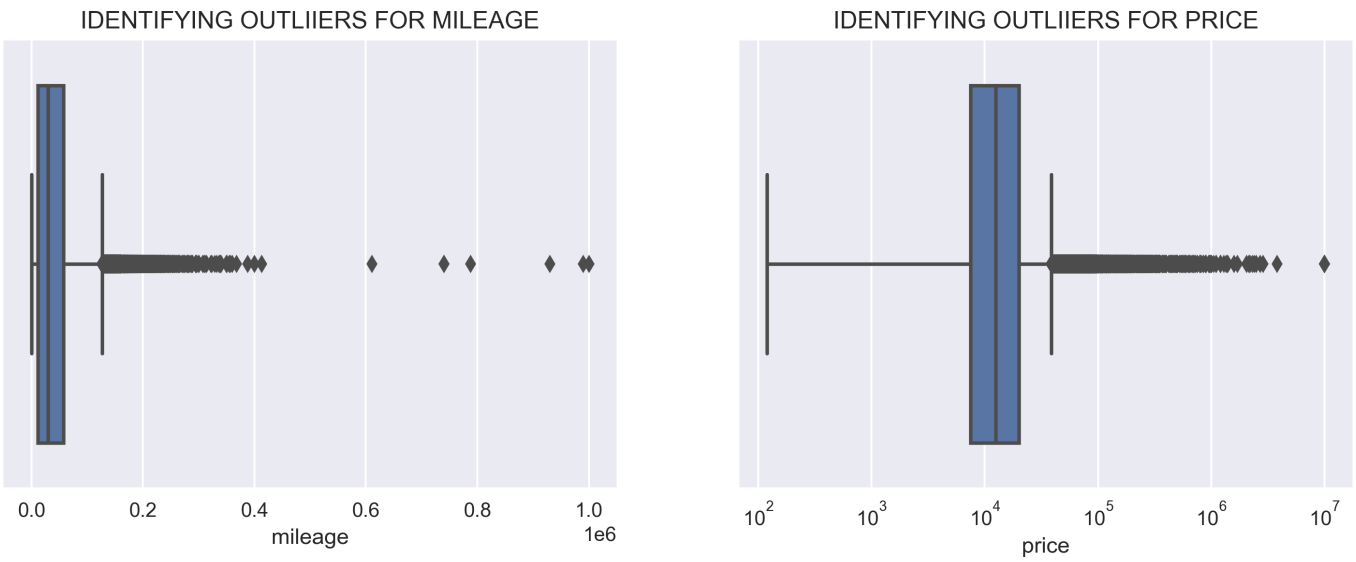

Fig.5 Boxplots Before Dealing with Outliers

For `Mileage`, I capped outliers for values greater than the 99th percentile which is 143,000 miles and replaced the outliers with value at the 99th percentile(143,000 miles).

For `price`, most of the top outliers in the price feature are expensive and luxurious cars, with most of them having low mileage. Handling the outliers in price was a tricky one. The car prices classified as outliers are actually expensive cars. For the purpose of this project, I excluded these expensive cars. I handled outliers for each subsets of vehicle conditions, capping the quantiles at different values.

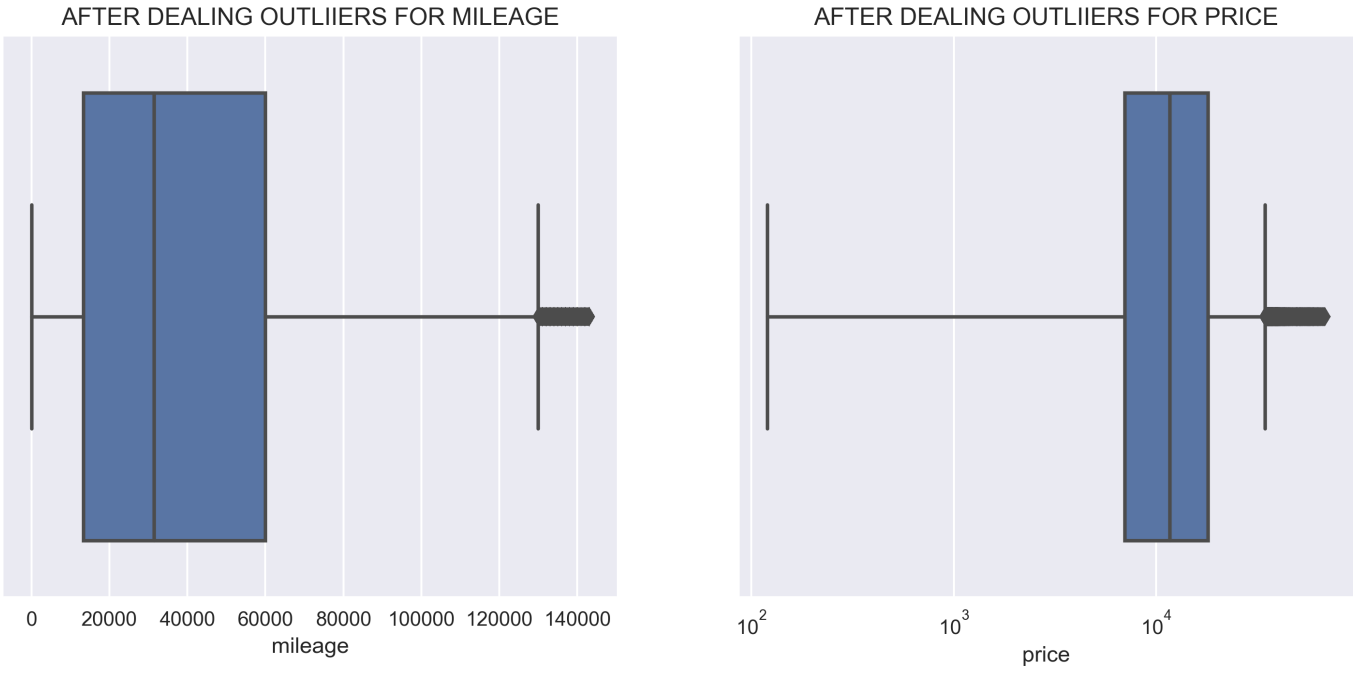

Fig.6 Boxplots After Dealing with Outliers

3.2. Feature Engineering, Data Transformations, Feature Selection

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('
','',''))
```

I encoded categorical variables using `OrdinalEncoder()`from `sklearn.preprocessing` library` which is very similar to target encoding but excludes the current row's target when calculating the mean target for a level to reduce the effect of outliers.

Feature Selection

There are several feature selection techniques, however, to gauge the feature importance, the novel correlation coefficient, ϕK's was used so because it is suitable for data sets with mixed variable types, e.g. where some variables are ordinal or categorical.

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.

Some features have a perfect correlation coefficient of 1 with each other. These feature prairs are:`year_of_registration` and `Vehicle_age` ,`reg_code` and `vehicle_condition`, `standard_make` and `standard_model`. However, ,`reg_code` and `vehicle_condition`, `standard_make` and `standard_model` have different score with the traget variable(`price`). I dropped `vehicle_age` `when selecting features for predictions.

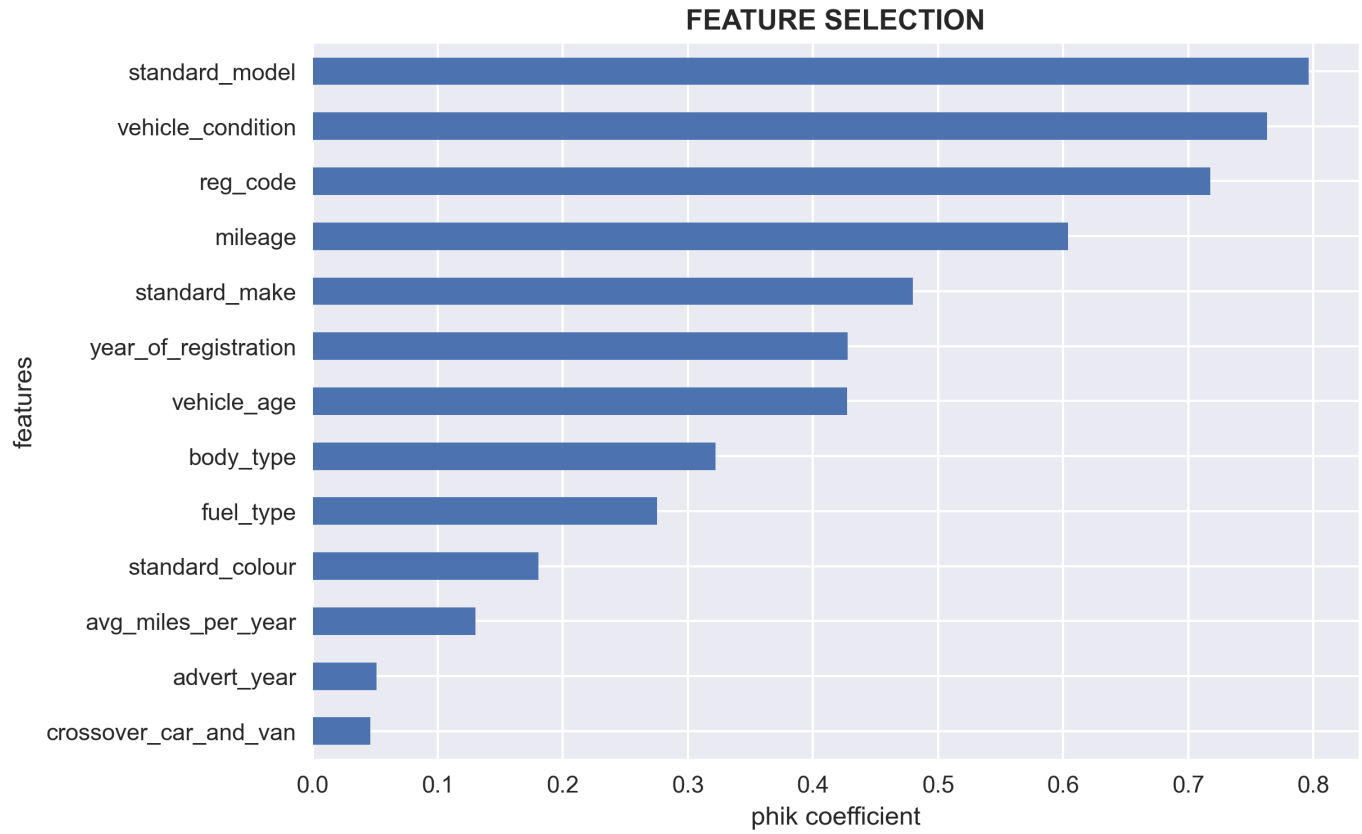

Fig.7 Feature Importance

4. Model Building

4.1 Algorithm Selection, Model Instantiation and Configuration

After data processing, **365,373** rows was left. The data was splitted into train and test set, **80%** and **20%** respectively. `LinearRegression()` was used as the baseline model. The table below contains the results.

train_score	test_score	mean_absolute_error	mean_squared_error	RMSE	r2_score
0.644174	0.640771	3895.338058	2.870305e+07	5357.522325	0.640771

Two more regression algorithms were also tested. `RandomForestRegressor()` and `DecisionTreeRegressor()`. Below are the results from all three models:

model	train_score	test_score	mean_absolute_error	mean_squared_error	RMSE	r2_score	time_taken
linear_regression	0.644174	0.640771	3895.338058	2.870305e+07	5357.522325	0.640771	1.799855
random_forest	0.988125	0.942380	1401.745475	4.603981e+06	2145.688836	0.942380	277.857838
decision_tree	0.994916	0.906525	1761.245323	7.468831e+06	2732.916209	0.906525	3.761690

Random Forest Regression model has the best overall model performance, i.e highest accuracy score, lowest mean absolute error, mean squared error, and root mean squared error. However, the computational time is really on a high side. Decision Tree Regressor is also performing well on the data set. With the right hyperparameters tunning, we should can get a better score at the end of the gridsearch iteration.

4.2 Grid Search, and Model Ranking and Selection

Due to to how fast the Decision Tree Regression model computes fast and saves computational memory, It was selected as the best model. The following parameters will be tunned to get a better overall score:

```
 def params_selection(X,y):
    model = {
        'decision_tree':{
            'model':DecisionTreeRegressor(random_state=seed),
            'params':{
                'max_depth':[20,25,30],
                'min_samples_split':[2,4,6,],
                'min_samples_leaf':[5,7,8]
            }
        }
    }
    for model_name, config in model.items():
        grid = GridSearchCV(config['model'], config['params'],
                                return_train_score=True,scoring='neg_mean_absolute_error')
        grid.fit(X,y)

    return pd.DataFrame(grid.cv_results_,columns=
['mean_test_score','mean_train_score','params','rank_test_score']).sort_values(by='rank_test_score')
```

Result from the Grid Search

mean_test_score	mean_train_score	params	rank_test_score
-1516.022460	-1208.188953	{'max_depth': 20, 'min_samples_leaf': 7, 'min_...	1
-1516.022460	-1208.188953	{'max_depth': 20, 'min_samples_leaf': 7, 'min_...	1
-1516.022460	-1208.188953	{'max_depth': 20, 'min_samples_leaf': 7, 'min_...	1
-1516.866563	-1235.059740	{'max_depth': 20, 'min_samples_leaf': 8, 'min_...	4
-1516.866563	-1235.059740	{'max_depth': 20, 'min_samples_leaf': 8, 'min_...	4

From the table of the grid search result above, looking at the scores for the last 2 rows, we can see it has a better trade off between mean train score and mean test score compared to the first three rows. Therefore, the best parameters for our Decision Tree Regressor is:

```
{'max_depth': 20, 'min_samples_leaf': 8, 'min_samples_split': 2}
```

using the new parameters to retrain our Decision Tree model,

```
grid_dtr = DecisionTreeRegressor(max_depth=20, min_samples_leaf=8, min_samples_split=2)
model_performance(grid_dtr)
```

train_score	test_score	mean_absolute_error	mean_squared_error	RMSE	r2_score
0.955015	0.932353	1502.842868	5.405140e+06	2324.895614	0.932353

5. Model Evaluation and Analysis

5.1 Coarse-Grained Evaluation/Analysis

The model got a train score of **95%** and a test score of **93%**.
The four metrics used above are commonly used metrics to evaluate the performance of a regression model.

Mean Absolute Error (MAE): This metric measures the average absolute difference between the predicted prices and the true prices. In this case, the MAE value of 1502.842868 implies that the model's predictions are off by an average of 1502.842868 units . A lower MAE value indicates a better fit.

R-Squared Score (R2): The R-squared value of the model is 0.932353, i.e 93.23%. This is considered a high value, indicating that the model has a good fit to the data. However, it's worth noting that a high R-squared value alone does not guarantee that the model is a good fit, other factors such as overfitting, underfitting, and outliers should be also considered.

A cross validation score of **-1491.508** was achieved on a 5 k-Fold splits which means when the data was splitted into 5 equal folds.

Fig.7 Actual vs Predicted Price

5.2 Feature Importance

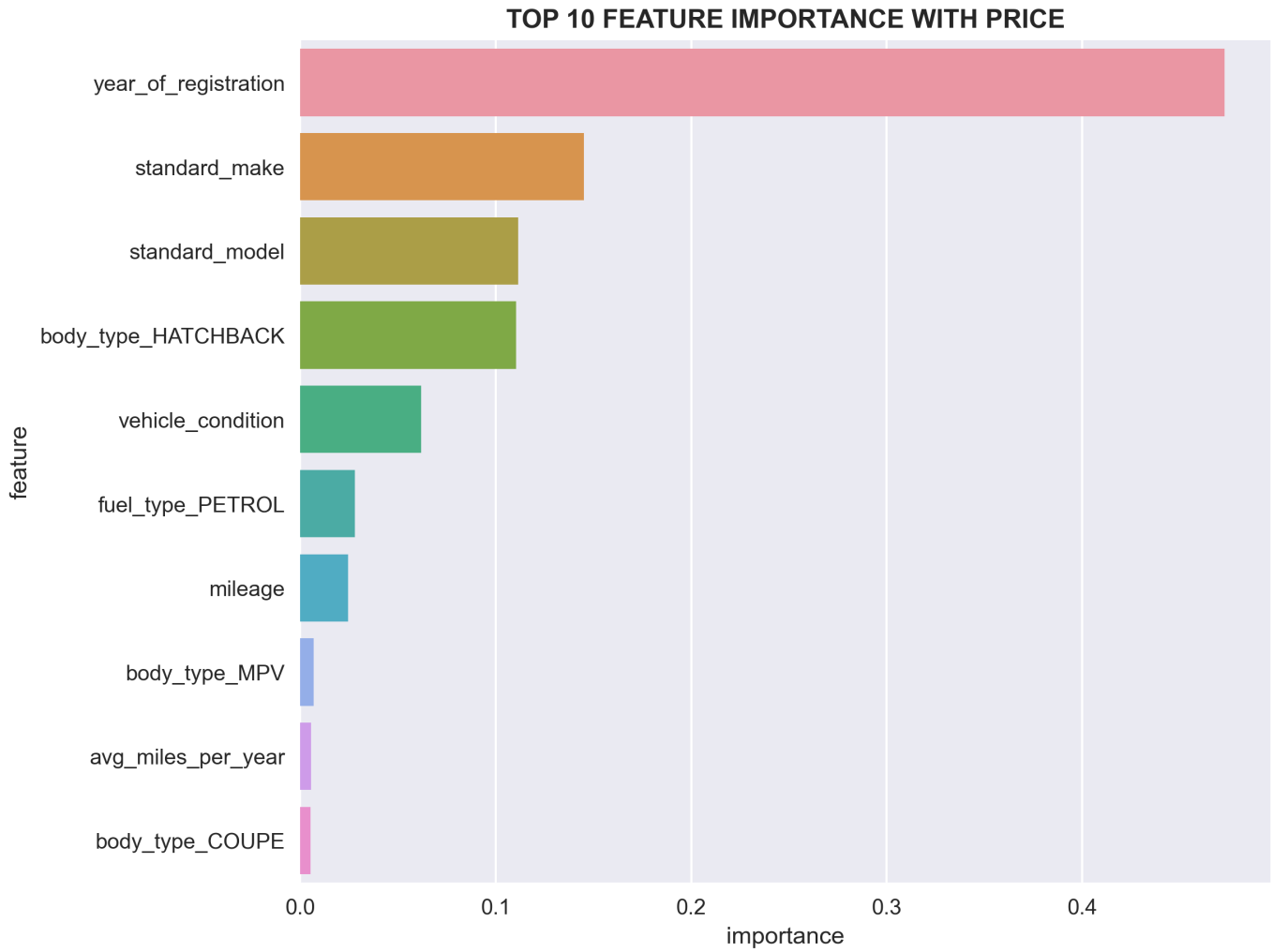

Fig.8 Top 10 Feature Importance with Price

The top features after the modelling are also the same with the features gotten from the predictive power score and also feature selection. Other features were genrerated after processing through encoding. It's not surprising that `year_of _registration`,`standard_make`, `standard_model`, and `vehicle_condition` are really good predictors of price.

5.3 Fine-Grained Evaluation

Below is a table showing the measures and score of the model on the overall prediction on the dataset:

mean_absolute_error	mean_squared_error	RMSE	MAPE	r2_score
1280.99	3935238.53	1983.74	0.13	0.95

Root Mean Squared Error (RMSE): RMSE is useful metric for measuring errors in predictions, because it is in the same units as the original data. The RMSE value of 1983.74 suggests that the model's predictions are on average off by **£1983.74** for a car.

156,057 rows have an absolute error greater than 1000 units, thats about **42%** of the entire dataset. Below is a scatter plot showing the data points for predicted and actual prices for these rows.

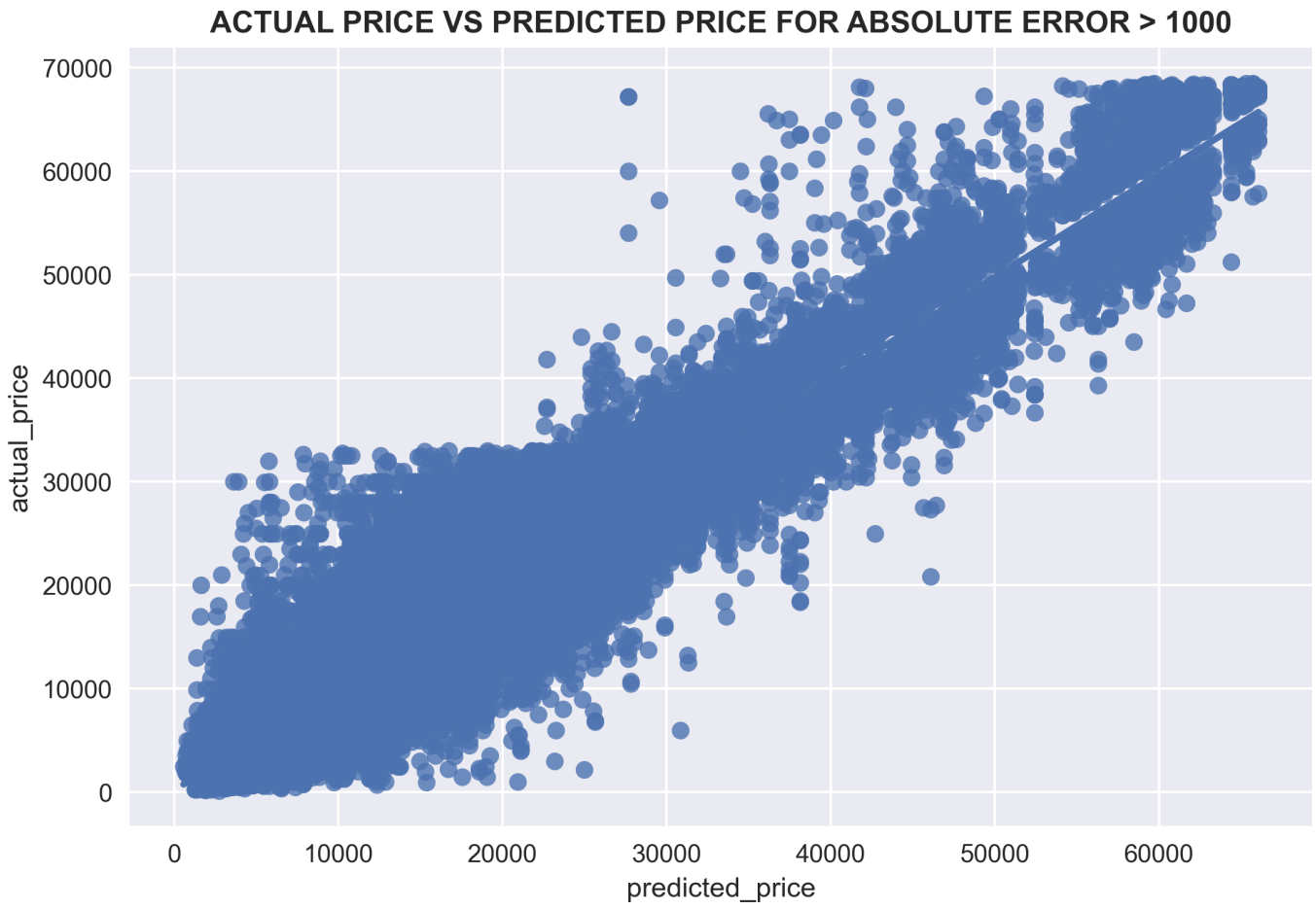

Fig.9 Actual Price vs Predicted Price for Absolute error greater than 1000.

209,301 rows have an absolute error greater than 1000 units, thats about **57%** of the entire dataset. Below is a scatter plot showing the data points for predicted and actual prices for these rows.

Fig.10 Actual Price vs Predicted Price for Absolute error less than 1000.

The mean absolute percentage error(**MAPE**) of the model is **13%**.

Using SHAP to explain two individual predicitons

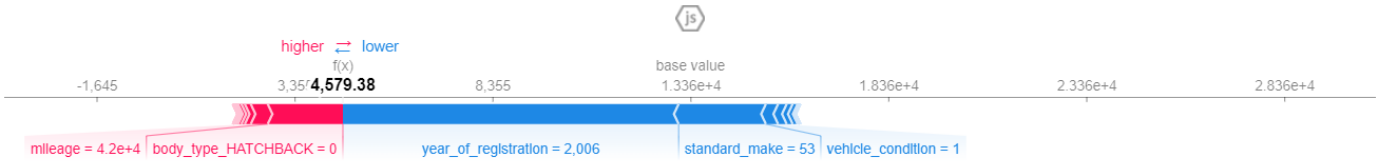

Fig.11 SHAP Explainer of an Individual Prediction of a car.

In Fig.11, the Vehicle is not a Hatchback which conrtibuted to the increase in the price of the car. However, the year of registration (2006) and the stanadard make which is a Mazda contributed to the reduction in the price of the car.

Let's check one more individual prediction;

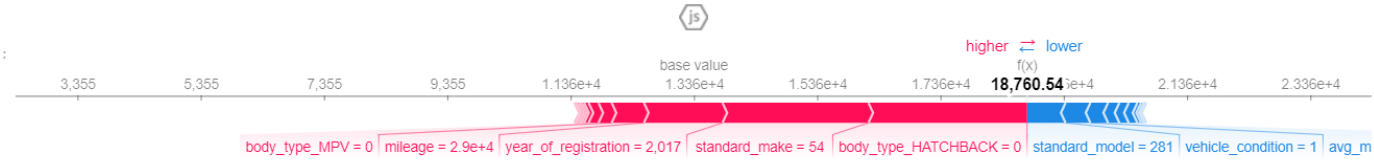

Fig.12 SHAP Explainer of an Individual Prediction of a car.

In Fig.12, the Vehicle is Mercedez Benz and the body type is not a Hatchback which greatly conrtibuted to the increase in the price of the car. Also, However, features that reduces the price of the car are, the model of the car(C Class), and the vechicle_condition is a new car reduces the price of the car.

Recommendation/Conclusion

In conclusion, the model seem to perform well on the overall data with a **r2_score of 95%**. Let's not forget we excluded expensive cars (above 90,000) from the dataset so as to get a good model fit. A different model can be built to predict expensive cars.

Finally, the three important features of prediciting the price of a vehicle are the body type, the year of registration, make and model of the car.