Principle of Data Science-6G7V0026

CAR SALE ADVERT

Submitted by- Mitali Arun Patle (22537720)
Date of Submission: 13/01/2022

Contents

1. Data Understanding and Exploration

The Data set provided on which we are working is Car Sale Advert dataset from an Autotrader. The dataset contains collective information of cars like Brand, type, Mileage, Color, Year of registration and Selling Price.

The Data set has a wide range of data having details of more than 4 million vehicles. Our aim is to clean the data set from noises and outliers and visually analyze the distribution of data and then make predictions based on the correlation between different features and targets.

❖ Types of features: Analysis and Distribution

In the data provided we have the following details of a vehicle-

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

```
ds.sample(3)
```

	public_reference	mileage	reg_code	standard_colour	standard_make	standard_model	vehicle_condition	year_of_registration	price	body_type	crossover_car_and_van	fuel_type
164087	202006240462769	8700.0	66	Grey	Audi	R8	USED	2016.0	84995	Convertible	False	Petrol
290542	202010245374219	25349.0	67	Blue	Kia	Sportage	USED	2017.0	11800	SUV	False	Diesel
320241	202010074721477	24943.0	66	Red	Fiat	500X	USED	2016.0	8680	SUV	False	Petrol

We have the following data of a vehicle-

1. Public reference
2. Mileage
3. Registration code
4. Standard Color
5. Brand
6. Model
7. Vehicle Condition
8. Year of registration
9. Price
10. Body type
11. Cross over of Car or Van
12. Fuel type

First, we will identify the features and the target from the above data. The only value which can be predicted and analysed based on the other data is “Price”. Therefore, price will be our target and other data will be our features.

❖ Identifying the range and shape of data

```
ds.shape
```

```
(402005, 12)
```

```
ds.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

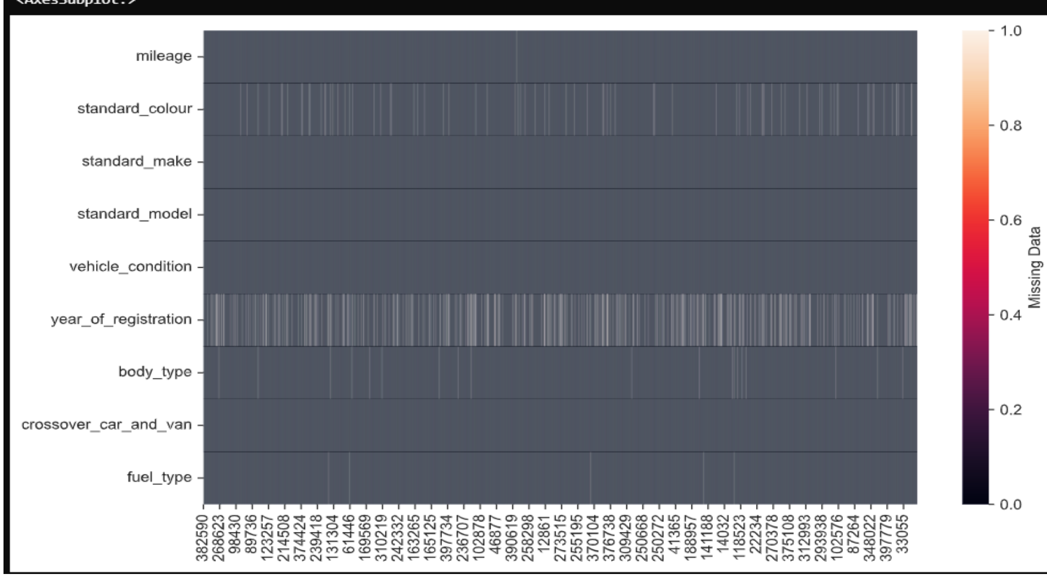
Shape tells us the size of our dataset. Here the number of rows is 402005 and number of columns are 12.
Using describe() function I can check the range of all our quantitative data. It gives us descriptive statistics and measures of dispersion.
This shows the maximum and the minimum values of Price, Mileage and Year of Registration.

❖ Identification of Missing values

The data provided to us have a lot of null values, i.e., no entries Ire made for that specific section. I can visualize these null values using a Heatmap-

```
plt.figure(figsize=(10,6))
sns.heatmap(dss.isna().transpose(),
            cbar_kws={'label': 'Missing Data'})
```

```
<AxesSubplot:>
```



Observations:

From the above heatmap I can see that there are lots of null values, especially in the 'Year of registration', 'Standard color', 'Body type', 'Fuel type' and 'Mileage'.

If we see the actual numbers of the null values in the Data-

```
features.isna().sum()

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

There are more than 30,000 entries in the dataset with null value. These null values can make aur predictions and analysis incorrect.

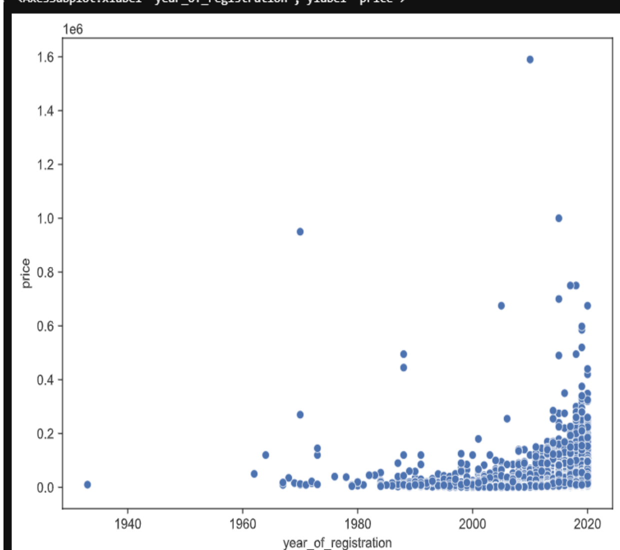
Since the numbers are very high so I cannot drop these rows. Therefore, our next step would be to substitute these null values with a specific value which does not disturb the distribution of our data.

❖ Detecting Outliers

As our data is very huge, there are lots of chances that our data will have outliers/incorrect values, now we will try to analyze outliers in our dataset using scatter plot.

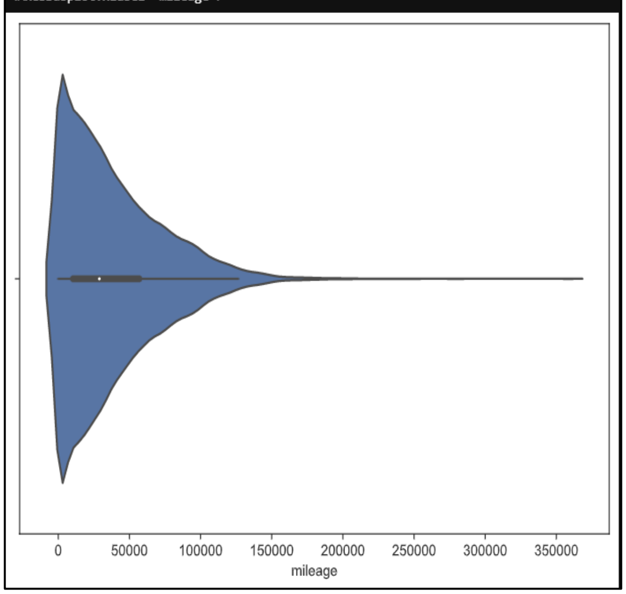
```
dss=advert.iloc[:40000,:]
sns.scatterplot(data=dss, x='year_of_registration',y='price')
```

```
<AxesSubplot:xlabel='year_of_registration', ylabel='price'>
```




```
sns.violinplot(data= dss, x='mileage',hue='price')
```

```
<AxesSubplot:xlabel='mileage'>
```



Observations:

1. The scatter plots show us that there are many points which are far away from the cluster and are not in the patterns. These are the outliers in our data.
2. The violin plot shows the distribution of the mileage and the box plot inside the violin shape shows the quartiles of the mileage. By observing the violin plot, I can see that there are many outliers as they fall outside the violin shape and far from the box plot.

2. Data Processing

Data processing is a very important part when you are trying to analyze the data. In the previous part we have already identified that our data has lots of Noise and outliers. Before proceeding ahead, we will deal with all the missing values and outliers.

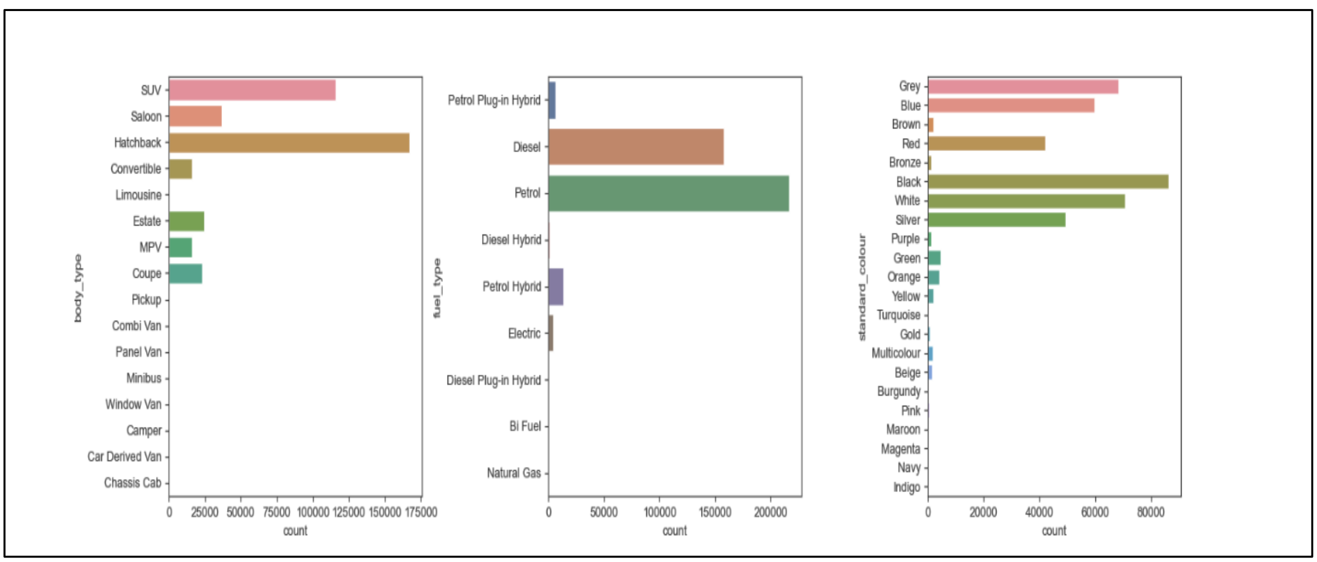
❖ Filling Missing Values

Now we will fill the null values with a meaningful value so that our data distribution does get disturbed. For numerical data we will replace the null values will the mean of the data using fillna() function.

```
features['mileage']= features['mileage'].fillna(features['mileage'].mean())

features['year_of_registration']=features['year_of_registration'].fillna(features['year_of_registration'].mean())
features['year_of_registration']=features['year_of_registration'].astype(int)
```

However, for categorical data I will replace the missing values with the maximum occurring values.



From the above bar plots, we can observe that for Body type the most occurred value is “Hatchback”, for Fuel type is “Petrol” and for colour is “Black”. Therefore, we will replace the null values using fillna() function.

```
features['body_type']=features['body_type'].fillna("Hatchback")
```

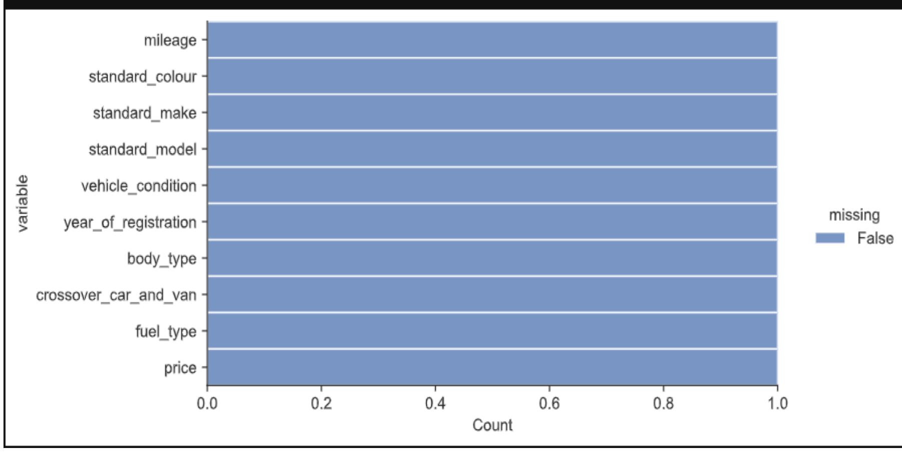
```
features['fuel_type']=features['fuel_type'].fillna("Petrol")
```

```
features['standard_colour']=features['standard_colour'].fillna("Black")
```

Now all our Null values are replaced with a meaningful data. We can visualize this by using a Distplot- Distplot is basically used to visualize the distribution of a single variable, here we are try to visualize the distribution of NaN values.

```
sns.displot(data=advert.isna().melt(value_name="missing"),y="variable",hue="missing",multiple="fill",aspect=2,height=3.75)
```

```
<seaborn.axisgrid.FacetGrid at 0x2434622bc70>
```



From this Graph we can clearly observe that there are no null values in our Data.

❖ Removing outliers

We have observed that our data have outliers/incorrect values in the data set, we will now try to remove these outliers. There are several methods to remove outliers, however, we will be using the Inter Quantile Range (IQR) method.

We are using this method because it is based on the distribution of the data, rather than on a fixed value or an assumption about the data.
To use this method, we have created user defined function to calculate the range of our data-

```
def quantile(x):
    global Q1
    global Q3
    Q1=x.quantile(0.25)
    Q3=x.quantile(0.75)
    print('Q1=',Q1,'\nQ3=',Q3)
    global IQR
    IQR = Q3 -Q1
    print ('IQR=', IQR)
    global Lowerlimit
    Lowerlimit= Q1 - 1.5*IQR
    global Upperlimit
    Upperlimit= Q3 + 1.5*IQR
    print ("Lower limit=", Lowerlimit, "\nUpper limit=", Upperlimit)
```

Using this function, we have calculated the upper range and the lower range of price, year of registration and mileage.

```
print("IQR for price-")
quantile(advert.price)

IQR for price-
Q1= 7495.0
Q3= 20000.0
IQR= 12505.0
Lower limit= -11262.5
Upper limit= 38757.5
```

```
print("IQR for Year of Registration-")
quantile(advert.year_of_registration)

IQR for Year of Registration-
Q1= 2013.0
Q3= 2018.0
IQR= 5.0
Lower limit= 2005.5
Upper limit= 2025.5
```

```
print("IQR for Mileage-")
quantile(advert.mileage)

IQR for Mileage-
Q1= 12126.0
Q3= 56492.0
IQR= 44366.0
Lower limit= -54423.0
Upper limit= 123041.0
```

We now have the upper and loIr IQ range of our quantitative data. I will trim our data by using query().

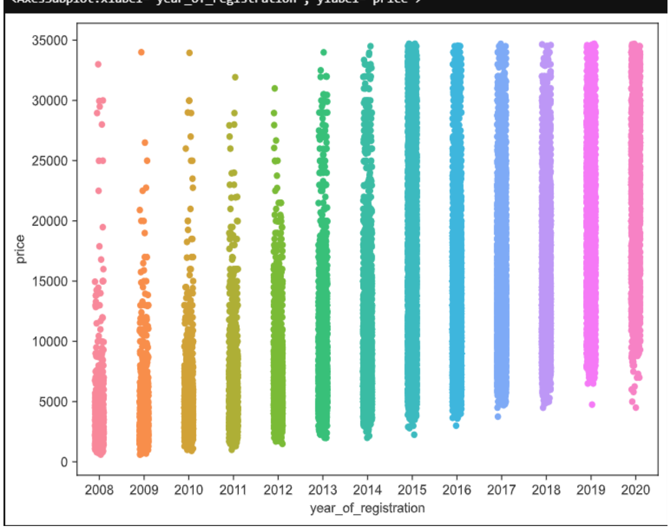
```
advert=advert.query("year_of_registration <= @Upperlimit and year_of_registration >= @Lowerlimit")
```

```
advert=advert.query("mileage <= @Upperlimit")
```

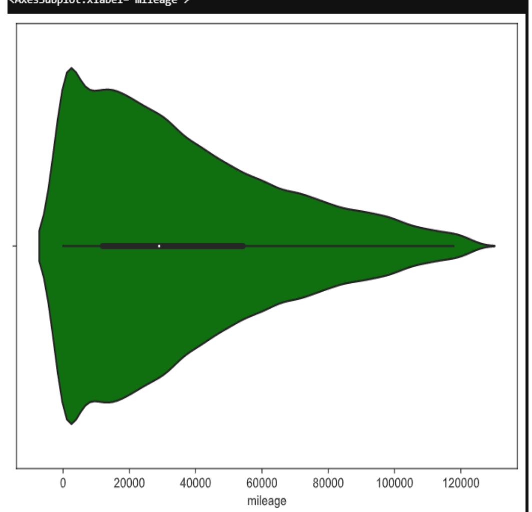
```
advert=advert.query("price <= @Upperlimit")
```

Now all the outliers has been removed from the data.

```
sns.stripplot(data=dss, x='year_of_registration',y='price')
<AxesSubplot:xlabel='year_of_registration', ylabel='price'>
```




```
sns.violinplot(data=dss,x='mileage',color='green')
<AxesSubplot:xlabel='mileage'>
```



Observation:

1. The strip plot shows that our data is now alligned with no outliers/noise. The value of price is continous and evenly distributed throughout the years.

2. There is no points outside the violin plot and the blox plot is inside the violin which shows that the outliers from the mileage are now removed.

❖ Feature Engineering and Data Transformation

Feature engineering is the most important part of our process. We will now try to convert all our raw data and categorical data into numeric values to find the correlation between all the features and between features and target.
Based on our correlation we will try to understand what features best fit for our analysis and prediction of the target.

Finding correlation between the features and the target

Here we are using the .astype('category') method to convert a column in the Data Frame from one data type to categorical type.

```
df=advert #taking backup of original data
```

```
df2_cat = df.astype({"standard_colour":"category","standard_make": "category", "standard_model": "category","vehicle_condition": "category","body_type": "category","crossover_car_and_van":"category","fue
```

```
df2_cat['standard_colour'] =df['standard_colour'].astype('category').cat.codes
df2_cat['standard_make'] =df['standard_make'].astype('category').cat.codes
df2_cat['standard_model'] =df['standard_model'].astype('category').cat.codes
df2_cat['vehicle_condition'] =df['vehicle_condition'].astype('category').cat.codes
df2_cat['body_type'] =df['body_type'].astype('category').cat.codes
df2_cat['crossover_car_and_van']=df['crossover_car_and_van'].astype('category').cat.codes
df2_cat['fuel_type'] =df['fuel_type'].astype('category').cat.codes
```

The data is now in categorical form using which we will find the correlations.

```
df2_cat.sample(5)
```

	mileage	standard_colour	standard_make	standard_model	vehicle_condition	year_of_registration	body_type	crossover_car_and_van	fuel_type	price
222959	75371.0	17	14	159	1	2010	7	0	1	3490
281325	6000.0	20	34	646	1	2020	13	0	5	14500
23256	105000.0	2	6	26	1	2010	14	0	1	3995
198046	43000.0	20	74	503	1	2016	7	0	5	8995
248704	27059.0	17	73	127	1	2016	7	0	5	7295

Observations:

1. This heatmap shows us the correlation of all the features with each other and with the target, i.e., price.
2. The highest correlation is between “Year of registration” and “Price”.
3. Mileage and vehicle condition have 32% correlation.
4. Body type & Price have 30% correlation.
5. Standard make and standard model have correlation of 27%
6. Fuel type & Standard make have 16% and Standard Model & Body type have correlation of 17%.

Based on these observations we can consider smaller subsets of our dataset and will try to make visual Analysis between categorical and quantitative features.

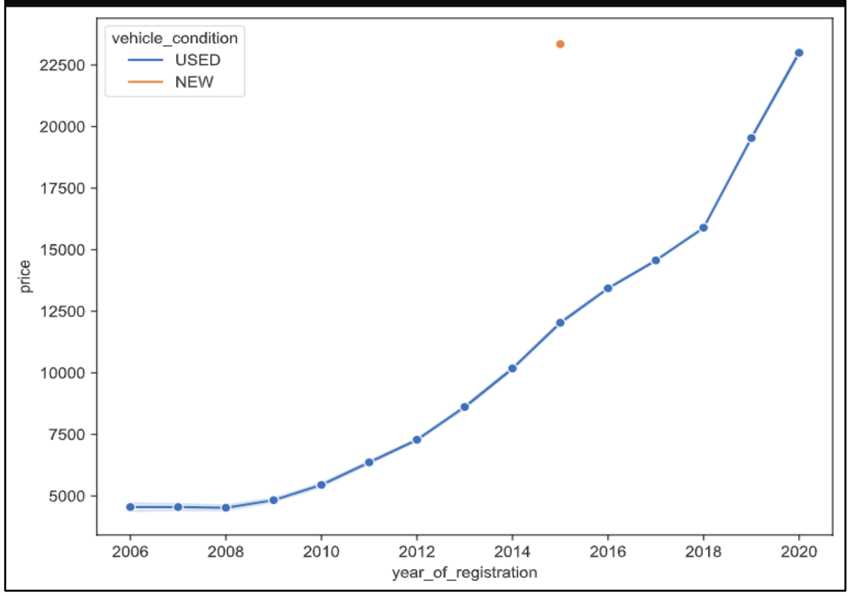
3. Association and Group Differences Analysis

❖ Quantitative -Quantitative

- **Price Vs Year of Registration**:

```
sns.lineplot(data=advert,x='year_of_registration',y='price',hue='vehicle_condition',marker= 'o')
```

```
<AxesSubplot:xlabel='year_of_registration', ylabel='price'>
```

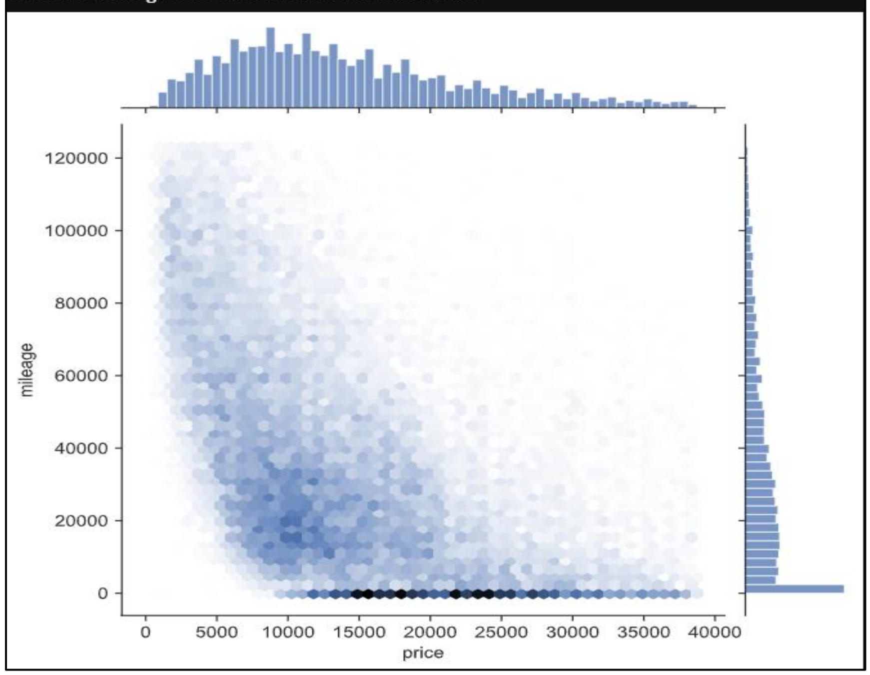


Observation:
-The line plot shows the variation of price of cars based on their year of registration.
-The prices of vehicles have increased exponentially throughout the years.
-Vehicles with New condition are only from year 2015 and have the highest price.
-Vehicles from the year 2020 are the costliest as their age is very less.
-Vehicles with greater age have lesser price.

- **Mileage Vs Price**

```
sns.jointplot(y = 'mileage', x = 'price',data = dss,kind = 'hex')
```

```
<seaborn.axisgrid.JointGrid at 0x255ccb226d0>
```



Observation:
- The Hexbin plot shows the relationship between the mileage and the price of a car.
- The plot shows the density of the points for price range, with the darker hexagons indicating a higher density of cars with lower mileage and high price.
-The cars with lower mileage have higher price.
-However, from this plot it can be seen that there is very less correlation between price and mileage and the price of car does not depends on the mileage of car.

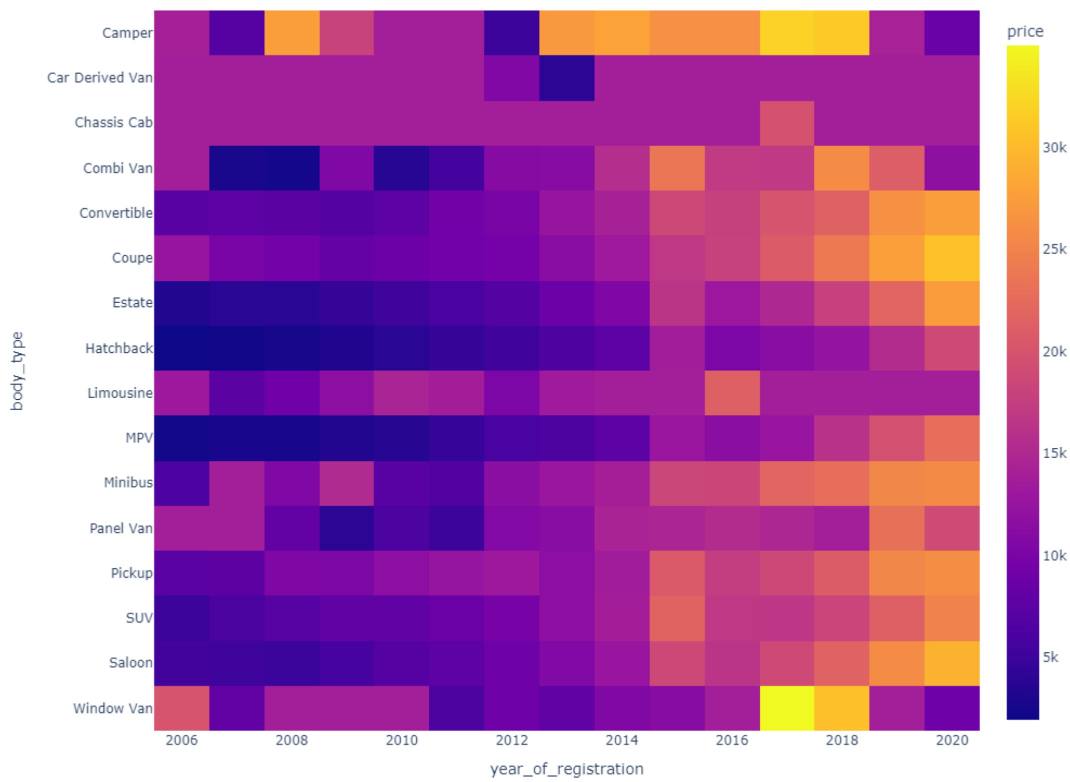
❖ Quantitative -Categorical

- **Body type Vs Year of registration & Price**

```
price_details = (
advert.groupby( ['body_type', 'year_of_registration'])['price'].mean()
    .unstack('year_of_registration').fillna(advert['price'].mean())

)
```

```
fig =px.imshow(price_details,labels=dict(x='year_of_registration', y='body_type', color='price'))
fig.update_layout(width=1000,height=800)
fig.show()
```

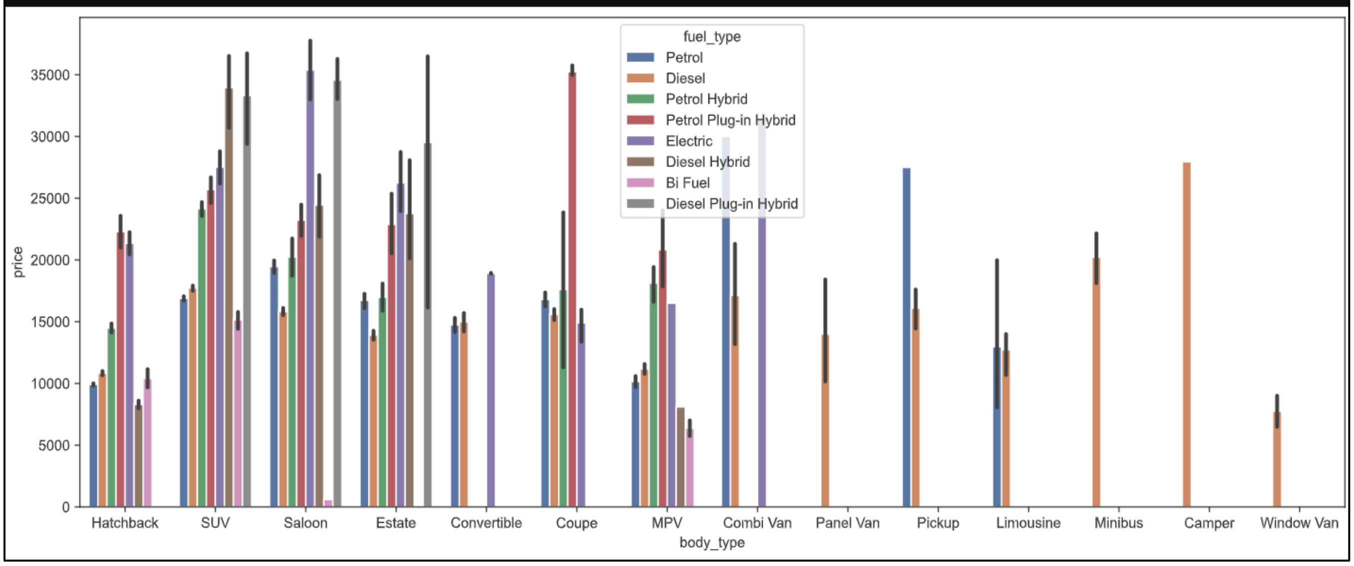


- **Observations**:
  The above heatmap is created using a subset of the data, grouped by Body type and Year of registration.
- Using this heatmap it can be observed that the prices are increasing gradually for all the Body types.
- Body type, such as Estate, Hatchback and MPV have comparatively lower price except from 2018 to 2020.
- Window Van body type is the only Body type whose price was highest in the year 2006 and lower in 2020.

- **Body type vs Price**

```
plt.figure(figsize=(16,6))
sns.barplot(data=dss, x='body_type',y='price',hue='fuel_type')
```

```
<AxesSubplot:xlabel='body_type', ylabel='price'>
```

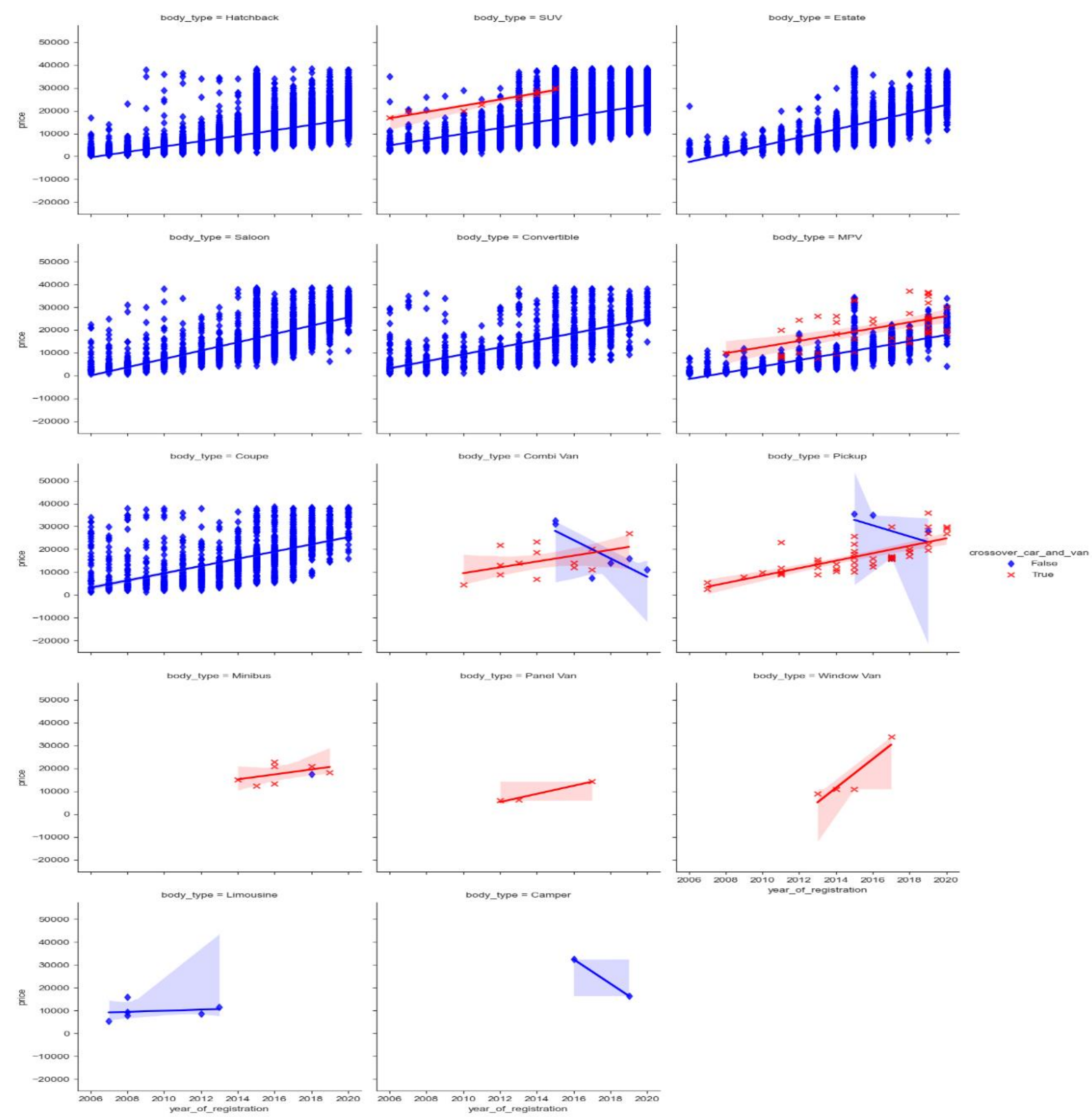


- **Observations-**
  The barplot shows the variation in the price of each body type based on the fuel type of the vehicle.
- SUV, Hatchback and Saloon are the only body type which has cars with 8 different fuel types.
- All the body types have Petrol and Diesel Fuel type vehicles with “Petrol Plug-in Hybrid”, “Electric” and “Diesel Hybrid” being the costliest.
- “Saloon”, “SUV”, “Estate” and “Coupe” Body type are the costliest ones.

❖ Categorical -Categorical

- Body Type-Crossover Car & Van Vs Price

```
sns.lmplot(x='year_of_registration',y='price',data=dss, hue='crossover_car_and_van',markers=['d','x'], palette=['blue','red'],col='body_type',col_wrap=3, height=4)
plt.savefig("Body Type VS Cross Over C&V-Price")
```



Observations:

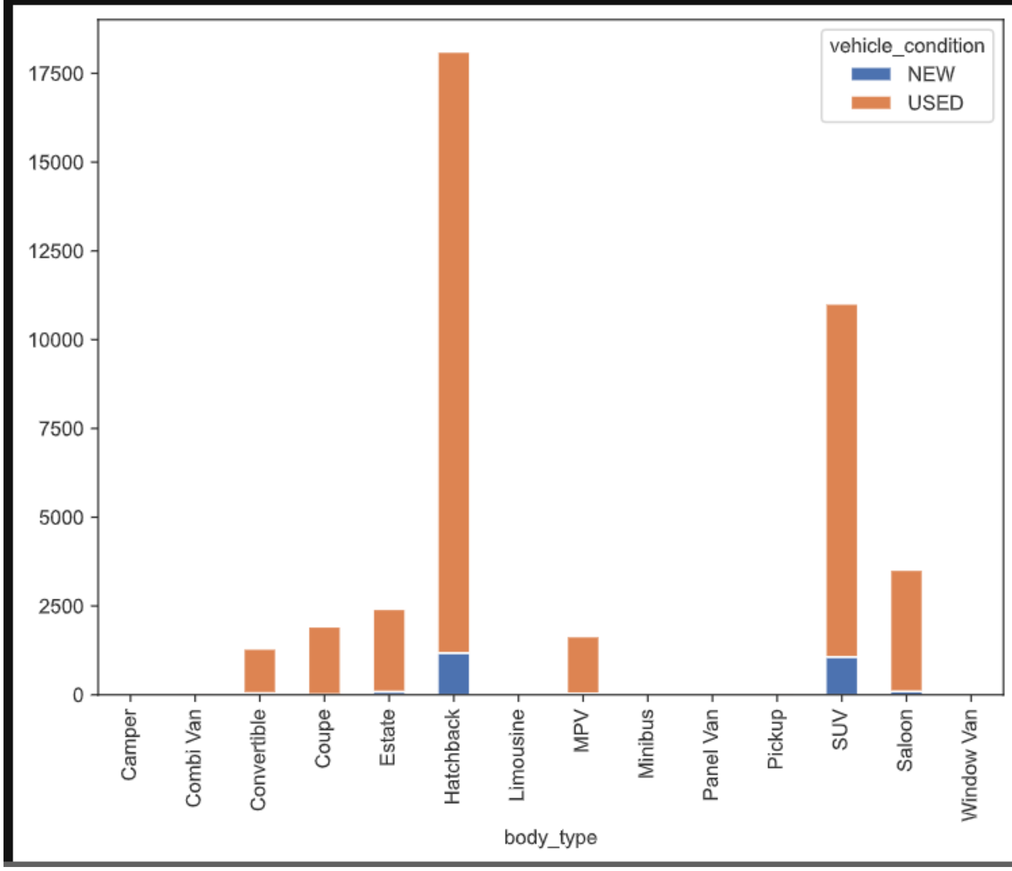
The linear regression plot shows the relationship between Body type and Price and based on, if they are cross over of Car & Van.

The price of the cars which are cross over of car and van are greater than the one which are not.

- Body Type Vs Vehicle condition

```
dss.groupby( ['body_type', 'vehicle_condition']).size().unstack('vehicle_condition').plot.bar(stacked=True)
```

```
<AxesSubplot:xlabel='body_type'>
```



Observation:

- The above stacked bar plot shows us how many new cars and how many used cars are available in the database.
- The new cars are only ‘Hatchback’ and ‘SUV’. Also, the cars available in the database are mostly ‘Hatchback’.