Used Cars Price Prediction

Context

There is a huge demand for used cars in the Indian Market today. As sales of new cars have slowed down in the recent past, the pre-owned car market has continued to grow over the past years and is larger than the new car market now. **Cars4U** is a budding tech start-up that aims to find footholes in this market.

In 2018-19, while new car sales were recorded at 3.6 million units, around 4 million second-hand cars were bought and sold. There is a slowdown in new car sales and that could mean that the demand is shifting towards the pre-owned market. In fact, some car owners replace their old cars with pre-owned cars instead of buying new ones. Unlike new cars, where price and supply are fairly deterministic and managed by **OEMs (Original Equipment Manufacturer)** except for dealership level discounts which come into play only in the last stage of the customer journey. Used cars are very different beasts with huge uncertainty in both pricing and supply. Keeping this in mind, the pricing scheme of these used cars becomes important in order to grow in the market.

Objectives:

- Explore and visualize the dataset
- Build a model to predict the prices of used cars
- Generate a set of insights and recommendations that will help the business

Data Dictionary

S.No. : Serial Number

Name : Name of the car which includes Brand name and Model name

Location : The location in which the car is being sold or is available for purchase (Cities)

Year : Manufacturing year of the car

Kilometers_driven : The total kilometers car has been driven by the previous owner(s) in KM

Fuel_Type : The type of fuel used by the car. (Petrol, Diesel, Electric, CNG, LPG)

Transmission : The type of transmission used by the car. (Automatic / Manual)

Owner : Type of ownership

Mileage : The standard mileage offered by the car company in kmpl or km/kg

Engine : The displacement volume of the engine in CC

Power : The maximum power of the engine in bhp

Seats : The number of seats in the car

New_Price : The price of a new car of the same model (units in INR 100,000)

Price : The price of the used car (units in INR 100,000) (**Target Variable**)

Loading libraries

In [3]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

# Import libraries to build linear model for statistical analysis and predic
from sklearn.linear_model import LinearRegression, Ridge, Lasso # Importing
from sklearn.tree import DecisionTreeRegressor # Importing Decision Tree Reg
from sklearn.ensemble import RandomForestRegressor # Importing Random Forest
from sklearn.model_selection import train_test_split # To split the data int

# Metrics to evaluate the model
from sklearn import metrics # To calculate the accuracy metrics

# For tuning the model
from sklearn.model_selection import GridSearchCV # For tuning parameters of

# To ignore warnings
import warnings
warnings.filterwarnings('ignore')

# Removes the limit from the number of displayed columns and rows
pd.set_option("display.max_columns", None)
pd.set_option("display.max_rows", None)
```

Loading and exploring the data

Loading the data into python to explore and understand it

In [4]:
```
#data = pd.read_csv("used_cars_data.csv")
```

Let us understand the data by observing a few rows

First and last 5 rows of the dataset

In [5]:
```
data.head()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[5], line 1
----> 1 data.head()

NameError: name 'data' is not defined
```

In [ ]:
```
data.tail()
```

Out[ ]:

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmissio
7248	7248	Volkswagen Vento Diesel Trendline	Hyderabad	2011	89411	Diesel	Manu
7249	7249	Volkswagen Polo GT TSI	Mumbai	2015	59000	Petrol	Automat
7250	7250	Nissan Micra Diesel XV	Kolkata	2012	28000	Diesel	Manu
7251	7251	Volkswagen Polo GT TSI	Pune	2013	52262	Petrol	Automat
7252	7252	Mercedes-Benz E-Class 2009-2013 E 220 CDI Avan...	Kochi	2014	72443	Diesel	Automat

In [ ]:
```
data.sample(2)
```

Out[ ]:

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission
1729	1729	Nissan Sunny XL	Chennai	2016	90000	Petrol	Manual
1432	1432	Ford Figo 2015-2019 1.2P Titanium Opt MT	Coimbatore	2017	25095	Petrol	Manual

Observations

- `S.No.` looks like an index for the data entry and such a column would not be useful for our analysis and we can drop it
- `Car names` contain a lot of model information. Let us check how many individual names we have. If they are too many, we can process this column to extract important information
- `New_Price` and our target variable `Price` have missing values

In [ ]:
```
# Removing S.No. column from data
data.drop(['S.No.'],axis = 1, inplace = True)
```

Let us check the data types and and missing values of each column

In [ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7253 entries, 0 to 7252
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Name               7253 non-null   object 
 1   Location           7253 non-null   object 
 2   Year               7253 non-null   int64  
 3   Kilometers_Driven  7253 non-null   int64  
 4   Fuel_Type          7253 non-null   object 
 5   Transmission       7253 non-null   object 
 6   Owner_Type         7253 non-null   object 
 7   Mileage            7251 non-null   object 
 8   Engine             7207 non-null   object 
 9   Power              7207 non-null   object 
 10  Seats              7200 non-null   float64
 11  New_Price          1006 non-null   object 
 12  Price              6019 non-null   float64
dtypes: float64(2), int64(2), object(9)
memory usage: 736.8+ KB
```

In [ ]:
```
data.isnull().sum()
```

Out[ ]:

```
Name                    0
Location                0
Year                    0
Kilometers_Driven       0
Fuel_Type               0
Transmission            0
Owner_Type              0
Mileage                 2
Engine                 46
Power                  46
Seats                  53
New_Price            6247
Price                1234
dtype: int64
```

Observations

- `Name`, `Location`, `Year`, `Kilometers_Driven`, `Fuel_Type`, `Transmission`, `Owner_Type` columns have no missing values
- `Mileage`, `Engine`, `Power`, `Seats`, `New_Price`, `Price` columns have missing values

In [ ]:

```
data.shape # rows and columns
```

Out[ ]:

```
(7253, 13)
```

Exploratory Data Analysis

Preprocessing the Data

In [ ]:

```
data.head(2)
```

Out[ ]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type
0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First
1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First

In [ ]:

```
# Some columns, which should have been numerical, are currently of object da
# We will extract only the numerical part from these columns to perform furt
data['Power'] = data['Power'].apply(lambda x: x.split(' ')[0] if pd.notnull(
data['Power'] = data['Power'].apply(lambda x: float(x) if x!= 'null' else np
data['Engine'] = data['Engine'].apply(lambda x: float(x.split(' ')[0]) if pc
```

```
In [ ]: data.sample(2)
```

Out[ ]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Ty
908	Maruti Swift Dzire VXI	Bangalore	2013	38623	Petrol	Manual	Fi
5475	Ford Figo Diesel EXI	Chennai	2011	76000	Diesel	Manual	Fi

```
In [ ]: 10,000,000 # 1 crore
        100,000 # 1 lakh
```

```
Out[ ]: (100, 0)
```

```
In [ ]: # Making unit same across whole column
def mileage_convert(x): # Function to convert km/kg to km per liter
    if type(x) == str: # if the data type is string
        if x.split()[-1] == 'km/kg': # If the unit is km/kg towards the end
            return float(x.split()[0])*1.40 # Formula for converting km/kg t
        elif x.split()[-1] == 'kmpl': # If the text is 'kmpl' instead of 'km
            return float(x.split()[0]) # Then convert that to float type for
    else:
        return x # If there is no 'kmpl' or 'km/kg', then we are good, no ac

def price_convert(x): # Function to extract the numerical price data from th
    if type(x) == str: # If the data type is string (text data a.k.a. object
        if x.split()[-1] == 'Cr': # Split the value in 'Cr', the last part c
            return float(x.split()[0])*100 # Formula for converting Crores t

        elif x.split()[-1] == 'Lakh': # If the string contains "Lakh", split
            return float(x.split()[0]) # Then keep the number part from the
    else:
        return x  # If neither "Lakh" nor "Cr." is present, keep the data as

data['Mileage'] = data['Mileage'].apply(mileage_convert) # Using the above c
data['New_Price'] = data['New_Price'].apply(price_convert) # Using the above
```

```
In [ ]: data.sample(2)
```

Out[ ]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_
7207	Maruti Swift Dzire ZDI	Hyderabad	2016	46372	Diesel	Manual	
2244	Hyundai i20 Magna 1.4 CRDi	Kochi	2015	64399	Diesel	Manual	

Let us now explore the summary statistics of numerical variables

It is important to understand the data statistically

In [ ]:

```
# Basic summary stats - Numeric variables
data.describe().T
```

Out[ ]:

	count	mean	std	min	25%	50%
Year	7253.0	2013.365366	3.254421	1996.00	2011.000	2014.0C
Kilometers_Driven	7253.0	58699.063146	84427.720583	171.00	34000.000	53416.0C
Mileage	7251.0	18.240986	4.839919	0.00	15.260	18.2C
Engine	7207.0	1616.573470	595.285137	72.00	1198.000	1493.0C
Power	7078.0	112.765214	53.493553	34.20	75.000	94.0C
Seats	7200.0	5.279722	0.811660	0.00	5.000	5.0C
New_Price	1006.0	22.779692	27.759344	3.91	7.885	11.57
Price	6019.0	9.479468	11.187917	0.44	3.500	5.64

Observations

- **The Manufacturing year of cars** varies from 1996 to 2019
- **At least 50% of the cars are 53416 kilometers_driven**, however, there are some extreme values, as the minimum value is 171 km and the maximum value is 6500000 km. We should check the extreme values to get a sense of the data
- **Average number of seats is around 5**
- **Average selling price of a used car is 9.47 lakh.** At least 50% of cars have been sold for 9.9 lakh or less, with the maximum selling price being 1 Cr 60 lakh
- **At least 75% of used cars have Mileage of 21 km or less** with the maximum value being 33.5 km. However, the minimum mileage of 0 is also troubling; we need to investigate this.

- The **mean of the new_price** is **22.77 lakh**, whereas **the median of the new_price** is **11.57 lakh**. This indicates that the new_price distribution is skewed towards the right side and explains that there are only a few very high range brands.

Let us also explore the summary statistics of all categorical variables and the number of unique observations in each category

In [ ]:

```
# Basic summary stats - Categorical variables
data.describe(include=['object']) # Alternatively, we can also do "exclude =
```

Out[ ]:

	Name	Location	Fuel_Type	Transmission	Owner_Type
count	7253	7253	7253	7253	7253
unique	2041	11	5	2	4
top	Mahindra XUV500 W8 2WD	Mumbai	Diesel	Manual	First
freq	55	949	3852	5204	5952

Number of unique observations in each category

It is necessary to undersand what are the count of unique values in each category, in the column of categorical data

In [ ]:

```
cat_cols = data.select_dtypes(include=['object']).columns[1:] # This variabl

for column in cat_cols: # For each individual column in the variable categor
    print("For column:",column) # Prints the name of the column joined with
    print(data[column].value_counts()) # Prints the count of the each indivi
    print('-'*50) # Prints 50 -s after each column value counts as a divider
```

```
For column: Location
Location
Mumbai        949
Hyderabad     876
Coimbatore    772
Kochi         772
Pune          765
Delhi         660
Kolkata       654
Chennai       591
Jaipur        499
Bangalore     440
Ahmedabad     275
Name: count, dtype: int64
--------------------------------------------------
For column: Fuel_Type
Fuel_Type
Diesel      3852
Petrol      3325
CNG           62
LPG           12
Electric       2
Name: count, dtype: int64
--------------------------------------------------
For column: Transmission
Transmission
Manual       5204
Automatic    2049
Name: count, dtype: int64
--------------------------------------------------
For column: Owner_Type
Owner_Type
First             5952
Second            1152
Third              137
Fourth & Above      12
Name: count, dtype: int64
--------------------------------------------------
```

Observations

- There are 2041 unique cars in our data.
- Most cars are from Mumbai and Hyderabad.
- Most of the cars have manual transmission.
- Most cars are first-owner vehicles.
- Very few cars use CNG, LPG, Electric Fuel_Type.

Check Kilometers_Driven extreme values

We observed from summary statistics that kilometers_driven column has extreme values . Let us check that column

In [ ]:
```
data.sort_values(by=["Kilometers_Driven"], ascending = False).head(10)
```

Out[ ]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Ow
2328	BMW X5 xDrive 30d M Sport	Chennai	2017	6500000	Diesel	Automatic	
340	Skoda Octavia Ambition Plus 2.0 TDI AT	Kolkata	2013	775000	Diesel	Automatic	
1860	Volkswagen Vento Diesel Highline	Chennai	2013	720000	Diesel	Manual	
358	Hyundai i10 Magna 1.2	Chennai	2009	620000	Petrol	Manual	
2823	Volkswagen Jetta 2013-2015 2.0L TDI Highline AT	Chennai	2015	480000	Diesel	Automatic	
3092	Honda City i VTEC SV	Kolkata	2015	480000	Petrol	Manual	
4491	Hyundai i20 Magna Optional 1.2	Bangalore	2013	445000	Petrol	Manual	
6921	Maruti Swift Dzire Tour LDI	Jaipur	2012	350000	Diesel	Manual	
3649	Tata Indigo LS	Jaipur	2008	300000	Diesel	Manual	
1528	Toyota Innova 2.5 G (Diesel) 8 Seater BS IV	Hyderabad	2005	299322	Diesel	Manual	

Observations

- In the first row, a car manufactured as recently as 2017 having been driven 6500000 km is almost impossible. It can be considered as data entry error, so we can remove this value/entry from the data
- The other observations that follow are also on a higher-end but kilometers driven by these cars are still reasonable as they are quite old. There is a good chance that these are outliers. We will look at this further while doing the univariate analysis

In [ ]:
```
# Removing this specific row from the above observation
data.drop(2328, inplace = True)
```

Check Mileage extreme values

We also observed from summary statistics that minimum mileage is zero. Let us check that column

In [ ]:
```
data.sort_values(by = ['Mileage'], ascending = True).head(10)
```

Out[ ]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owı
4234	Mercedes-Benz M-Class ML 350 4Matic	Chennai	2012	63000	Diesel	Automatic	
2053	Mahindra Jeep MM 550 PE	Hyderabad	2009	26000	Diesel	Manual	
6177	Mercedes-Benz M-Class ML 350 4Matic	Bangalore	2012	37000	Diesel	Automatic	
2542	Hyundai Santro GLS II - Euro II	Bangalore	2011	65000	Petrol	Manual	
262	Hyundai Santro Xing XL	Hyderabad	2006	99000	Petrol	Manual	
5426	Hyundai Santro Xing XL	Chennai	2006	85000	Petrol	Manual	
5943	Mahindra Jeep MM 540 DP	Chennai	2002	75000	Diesel	Manual	
4152	Land Rover Range Rover 3.0 D	Mumbai	2003	75000	Diesel	Automatic	
6633	Mahindra TUV 300 P4	Kolkata	2016	27000	Diesel	Manual	
2096	Hyundai Santro LP zipPlus	Coimbatore	2004	52146	Petrol	Manual	

Observations

- Mileage of cars **can not be 0**
- we should treat 0's as missing values

In [ ]:
```
data.loc[np.round(data['Mileage']) == 0, 'Mileage'] = np.nan
```

In [ ]:
```
data.sort_values(by = ['Mileage'], ascending = True).head(10)
```

Out[ ]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Ov
5781	Lamborghini Gallardo Coupe	Delhi	2011	6500	Petrol	Automatic	
5603	Porsche Panamera 2010 2013 Diesel	Delhi	2013	36400	Diesel	Automatic	
152	Mercedes-Benz S Class 2005 2013 S 500	Kolkata	2010	35277	Petrol	Automatic	
7057	BMW 6 Series 650i Coupe	Delhi	2009	64000	Petrol	Automatic	
4821	BMW 6 Series 630i Coupe	Mumbai	2011	5900	Petrol	Automatic	
4627	BMW 6 Series 650i Coupe	Kochi	2010	65329	Petrol	Automatic	
2978	Porsche Panamera 2010 2013 4S	Coimbatore	2010	42400	Petrol	Automatic	
4722	Mercedes-Benz SL-Class SL 500	Kolkata	2010	35000	Petrol	Automatic	
5218	BMW 3 Series 330d Convertible	Mumbai	2013	49000	Diesel	Automatic	
5868	BMW 3 Series 330d Convertible	Kochi	2014	51240	Diesel	Automatic	

Univariate Analysis

Univariate analysis is used to explore each variable in a data set, separately. It looks at the range of values, as well as the central tendency of the values. It can be done for both numerical and categorical variables

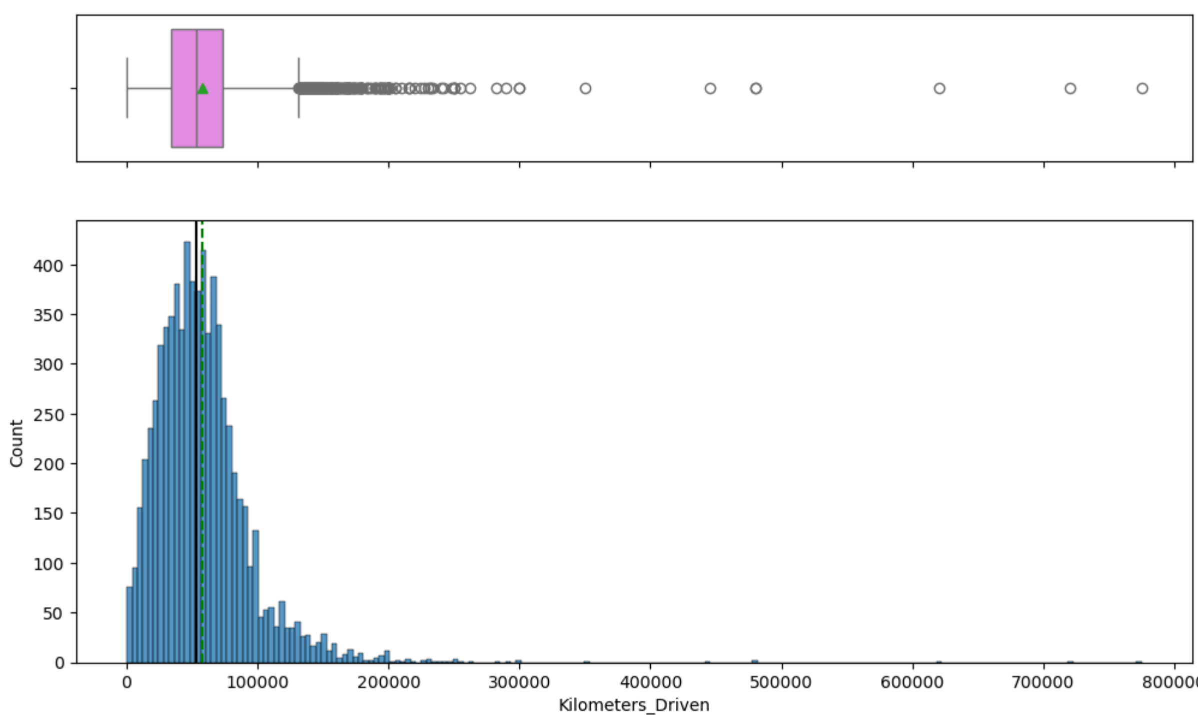
1.Univariate analysis - Numerical data

Histograms and box plots help to visualize and describe numerical data. We will use these to analyse the following numerical columns: `Kilometers_driven`, `power`, `price`, `mileage`.

In [ ]:
```
# Function to plot a boxplot and a histogram along the same scale

def histogram_boxplot(data, feature, figsize = (12, 7), kde = False, bins =
    """
    Boxplot and histogram combined

    data: dataframe
    feature: dataframe column
    figsize: size of figure (default (12,7))
    kde: whether to show density curve (default False)
    bins: number of bins for histogram (default None)
    """
    f2, (ax_box2, ax_hist2) = plt.subplots(
        nrows = 2,      # Number of rows of the subplot grid = 2
        sharex = True,  # x-axis will be shared among all subplots
        gridspec_kw = {"height_ratios": (0.25, 0.75)},
        figsize = figsize)                  # Creating the 2 subplots

    sns.boxplot(data = data, x = feature, ax = ax_box2, showmeans = True, cc
    sns.histplot(data = data, x = feature, kde = kde, ax = ax_hist2, bins =
    ax_hist2.axvline(data[feature].mean(), color = "green", linestyle = "--"
    ax_hist2.axvline(data[feature].median(), color = "black", linestyle = "-
```

Box plot and Histogram for Kilometers Driven

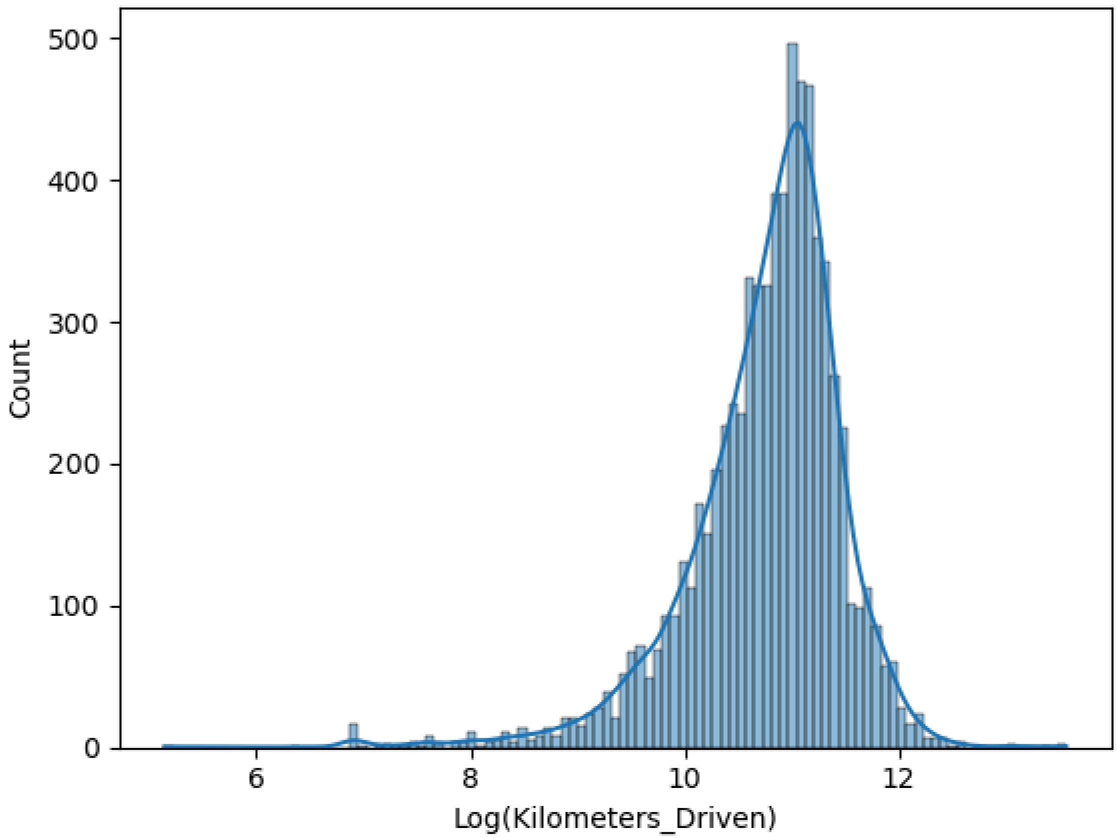
In [ ]:
```
# Plot histogram and box-plot for 'Kilometers_Driven'
histogram_boxplot(data, 'Kilometers_Driven')
```



- Kilometers_Driven is highly right-skewed. It is very difficult to interpret. Log transformation can be used to reduce/remove the skewness. Log transformed value can be used for analysis

In [ ]:
```
sns.histplot(np.log(data["Kilometers_Driven"]), kde=True)
plt.xlabel('Log(Kilometers_Driven)')
```

Out[ ]:
```
Text(0.5, 0, 'Log(Kilometers_Driven)')
```



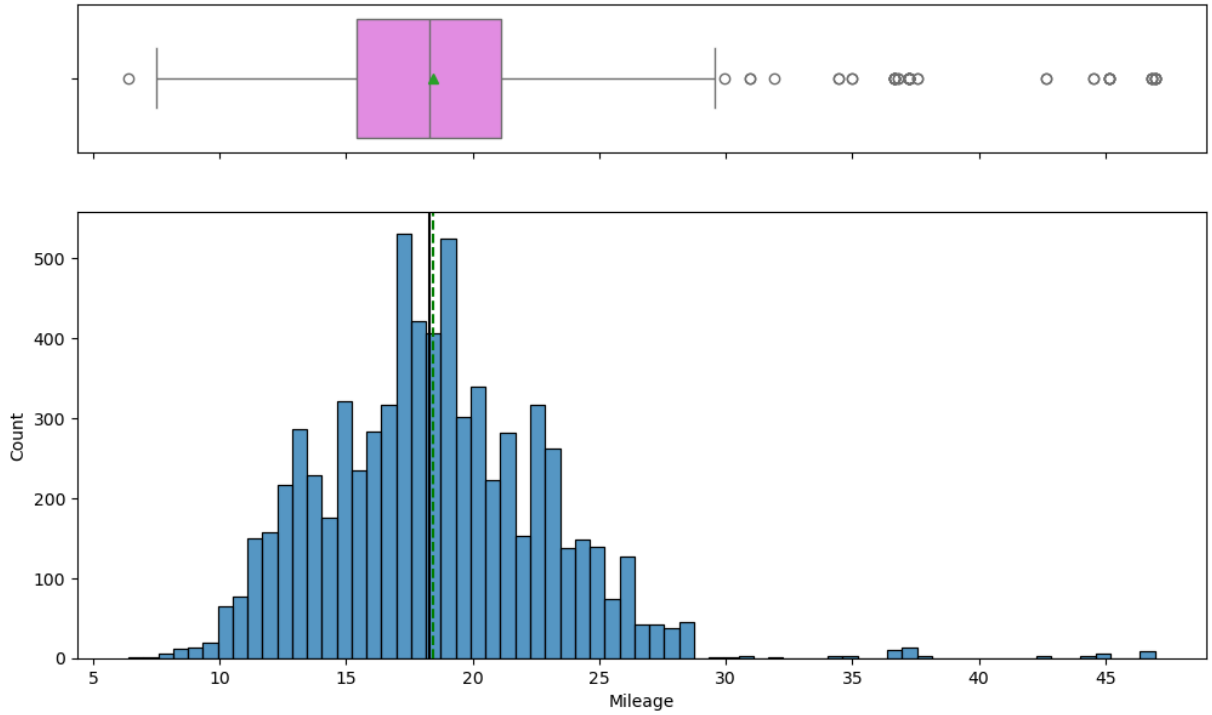
Observations

- Log transformation of data has reduced the extreme skewness
- From box-plot we can see the outliers, as we discussed in summary statistics

In [ ]:
```
# Adding a transformed kilometers_driven feature to the data.
data["kilometers_driven_log"] = np.log(data["Kilometers_Driven"])
```

Box plot and Histogram for Mileage

In [ ]:
```
histogram_boxplot(data, 'Mileage')
```

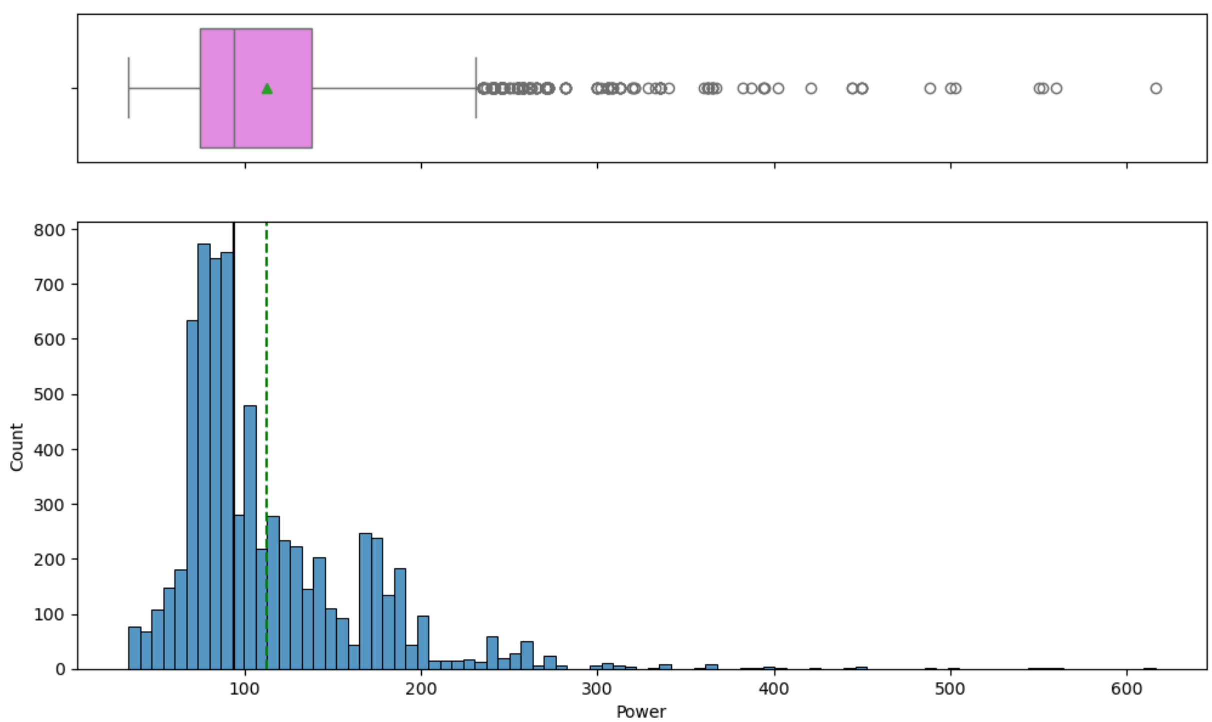


Observations

- The distribution of mileage looks fairly normally distributed if we ignore the cars with 0 mileage.
- From box plot also it is visible that the extreme values can be seen as outliers

Box plot and Histogram for Power

In [ ]:
```
histogram_boxplot(data, 'Power')
```

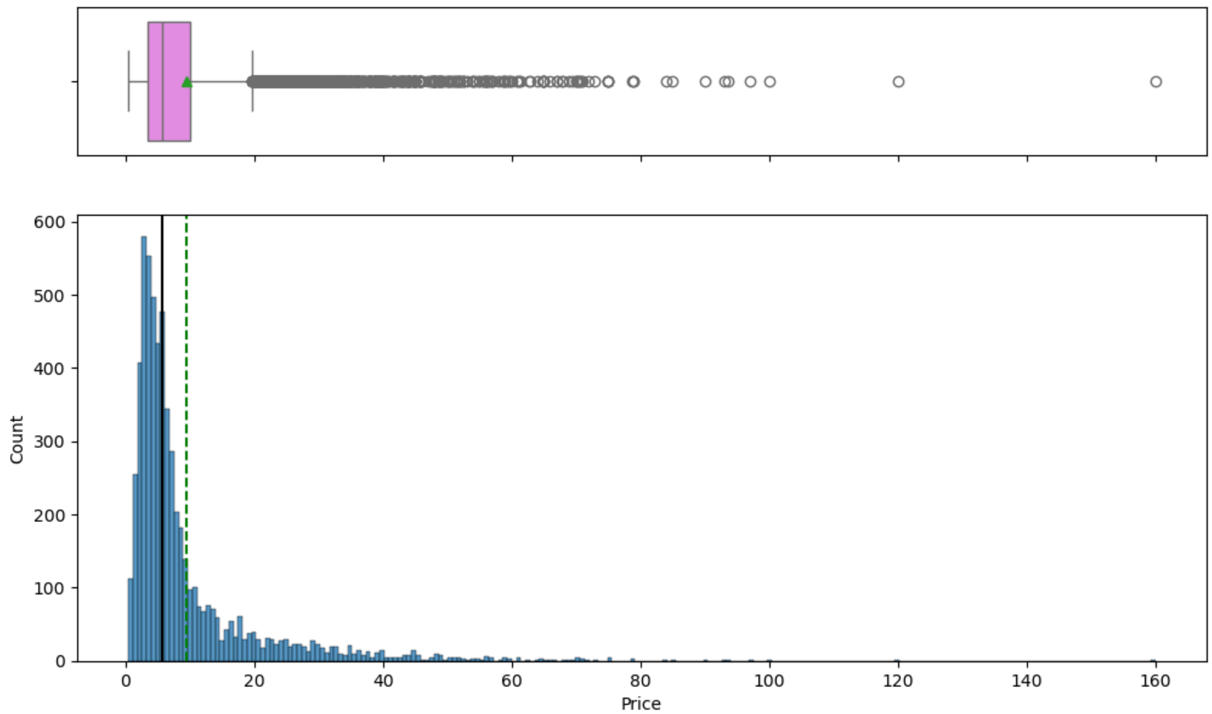


Observations

- Most cars have Power of engines between 90-100 bhp
- From the boxplot, we can see that there are many outliers in this variable - cars with more than 250 bhp are being considered as outliers in data

Box plot and Histogram for Price

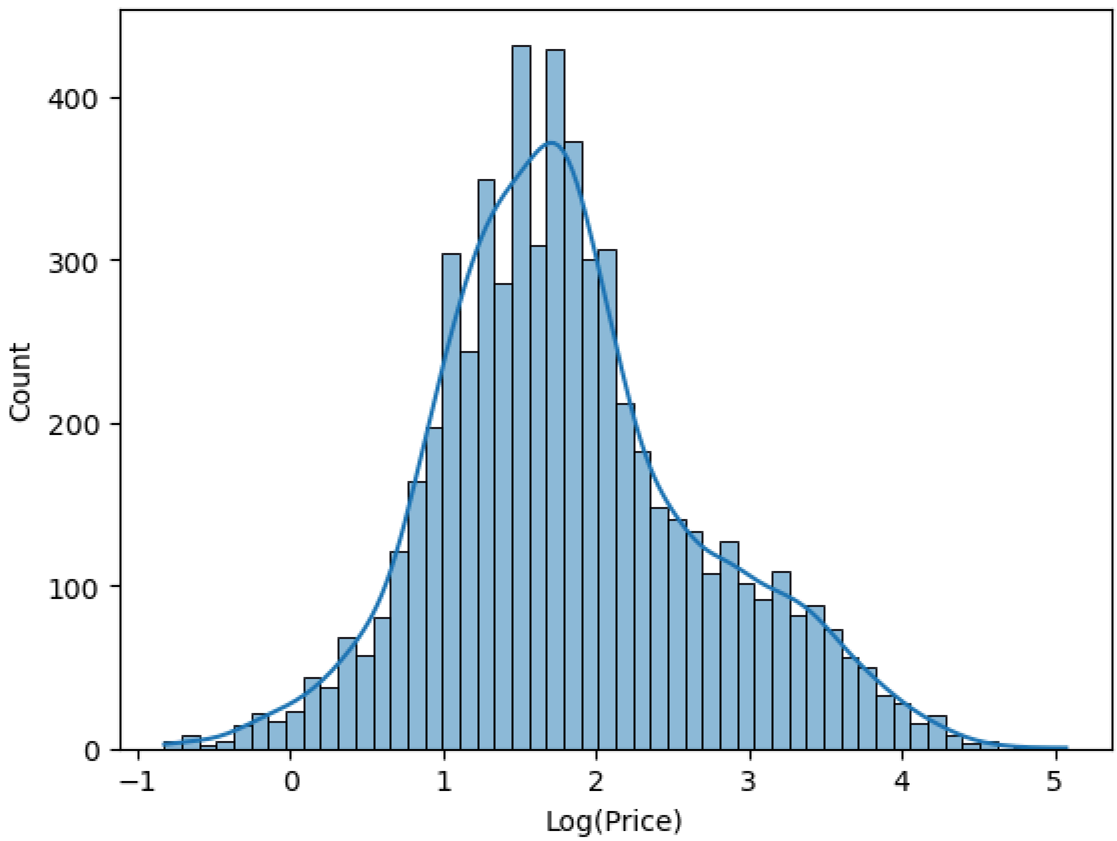
In [ ]:
```
histogram_boxplot(data, 'Price')
```



- The distribution of Price is highly skewed, we can use log transformation on this column to see if that helps normalize the distribution.

In [ ]:
```
sns.histplot(np.log(data["Price"]), kde=True)
plt.xlabel('Log(Price)')
```

Out[ ]:
```
Text(0.5, 0, 'Log(Price)')
```



Observations

- Log transformation helps to normalize the distribution
- It is observed that few extreme price values are there, as seen in summary statistics

In [ ]:
```
# Log Transformation has definitely helped in reducing the skew
# Creating a new column with the transformed variable.
data["price_log"] = np.log(data["Price"])
```

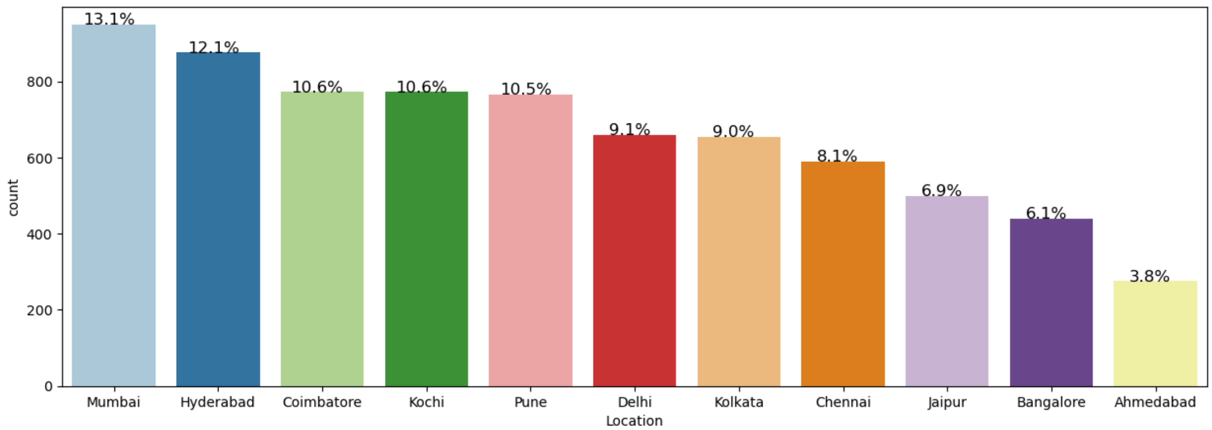
2.Univariate analysis - Categorical data

In [ ]:
```
# Let us write a function that will help us create barplots that indicate th
# This function takes the categorical column as the input and returns the ba

def perc_on_bar(data, z):
    '''
    plot
    data: DataFrame or Series
    z: categorical feature
    the function won't work if a column is passed in hue parameter
    '''

    total = len(data[z])  # Length of the column
    plt.figure(figsize=(15, 5))
    ax = sns.countplot(data=data, x=z, palette='Paired', order=data[z].value
    for p in ax.patches:
        percentage = '{:.1f}%'.format(100 * p.get_height() / total)  # Perce
```

```
        x = p.get_x() + p.get_width() / 2 - 0.05  # Width of the plot
        y = p.get_y() + p.get_height()  # Height of the plot

        ax.annotate(percentage, (x, y), size=12, ha='center')  # Annotate th

    plt.show()
```

Barplot for Location

In [ ]:
```
# % values has to have offset

perc_on_bar(data, 'Location')
```

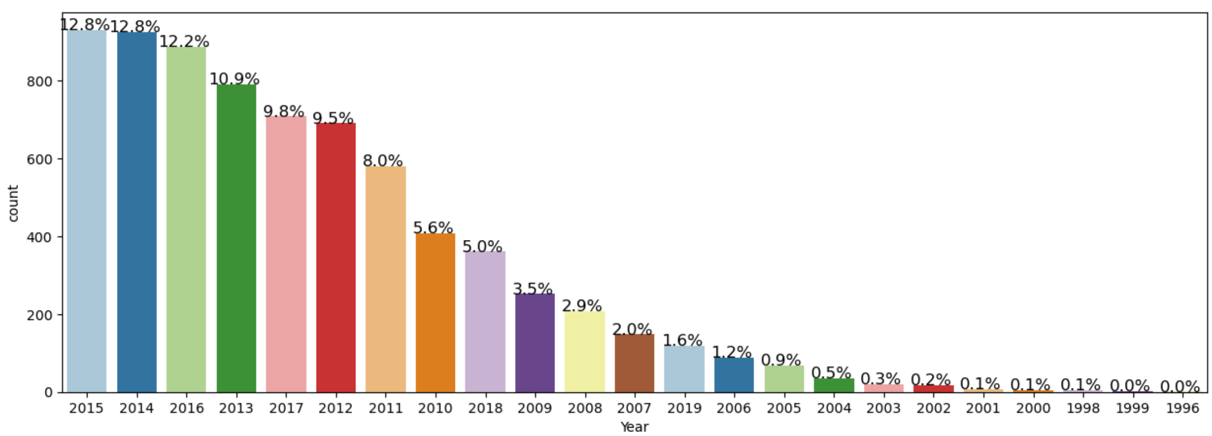


Observation

- 13.1% of the cars are from Mumbai followed by 12.1% of the cars from Hyderabad

Barplot for Year

In [ ]:
```
perc_on_bar(data, 'Year')
```

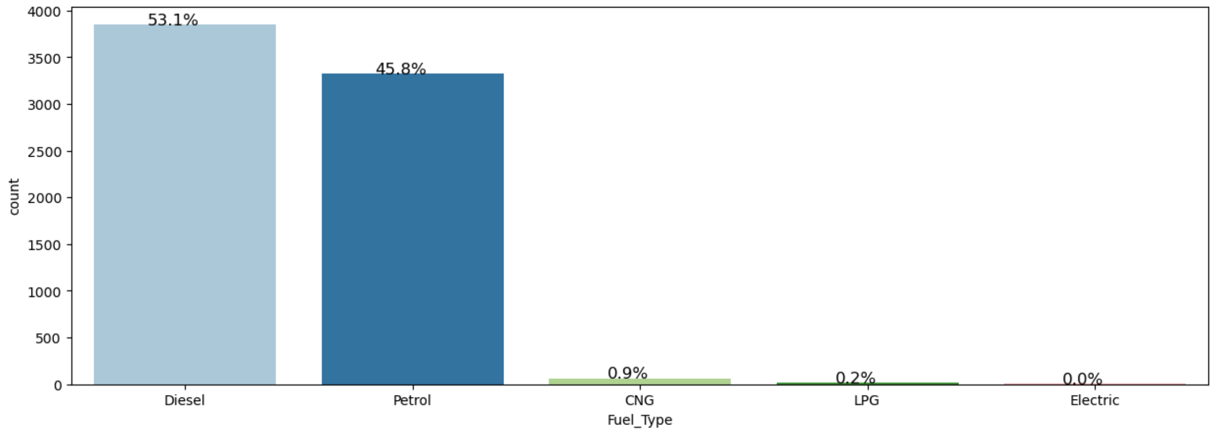


Observation

- About 38% of the cars in the data are from the year 2014 - 2016

Barplot for Fuel_Type

```
In [ ]: perc_on_bar(data, 'Fuel_Type')
```

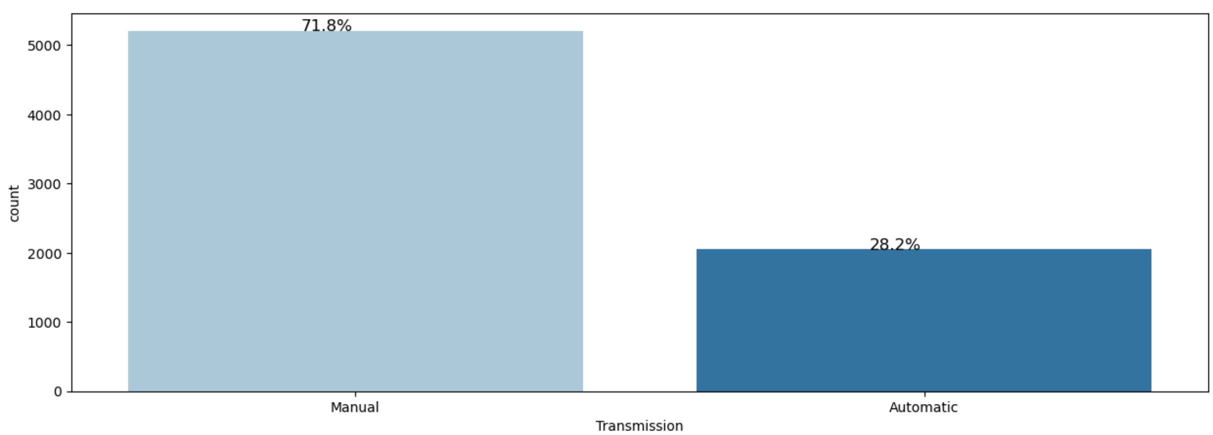


Observations

- Approximately 99% of cars are powered by Diesel and Petrol, while the remaining 1% use alternative fuels such as CNG, LPG, and electricity.

Barplot for Transmission

```
In [ ]: perc_on_bar(data, 'Transmission')
```

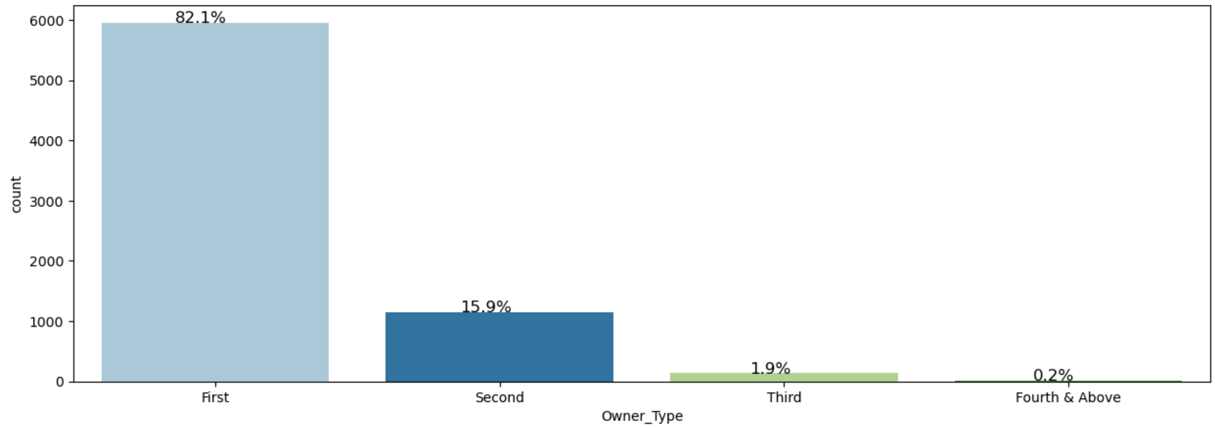


Observations

- 71.8% of the cars have a manual transmission

Barplot for Owner_Type

```
In [ ]: perc_on_bar(data, 'Owner_Type')
```



Observations

- 82.1% of the cars have first owners followed by 15.9% of the cars with second owners

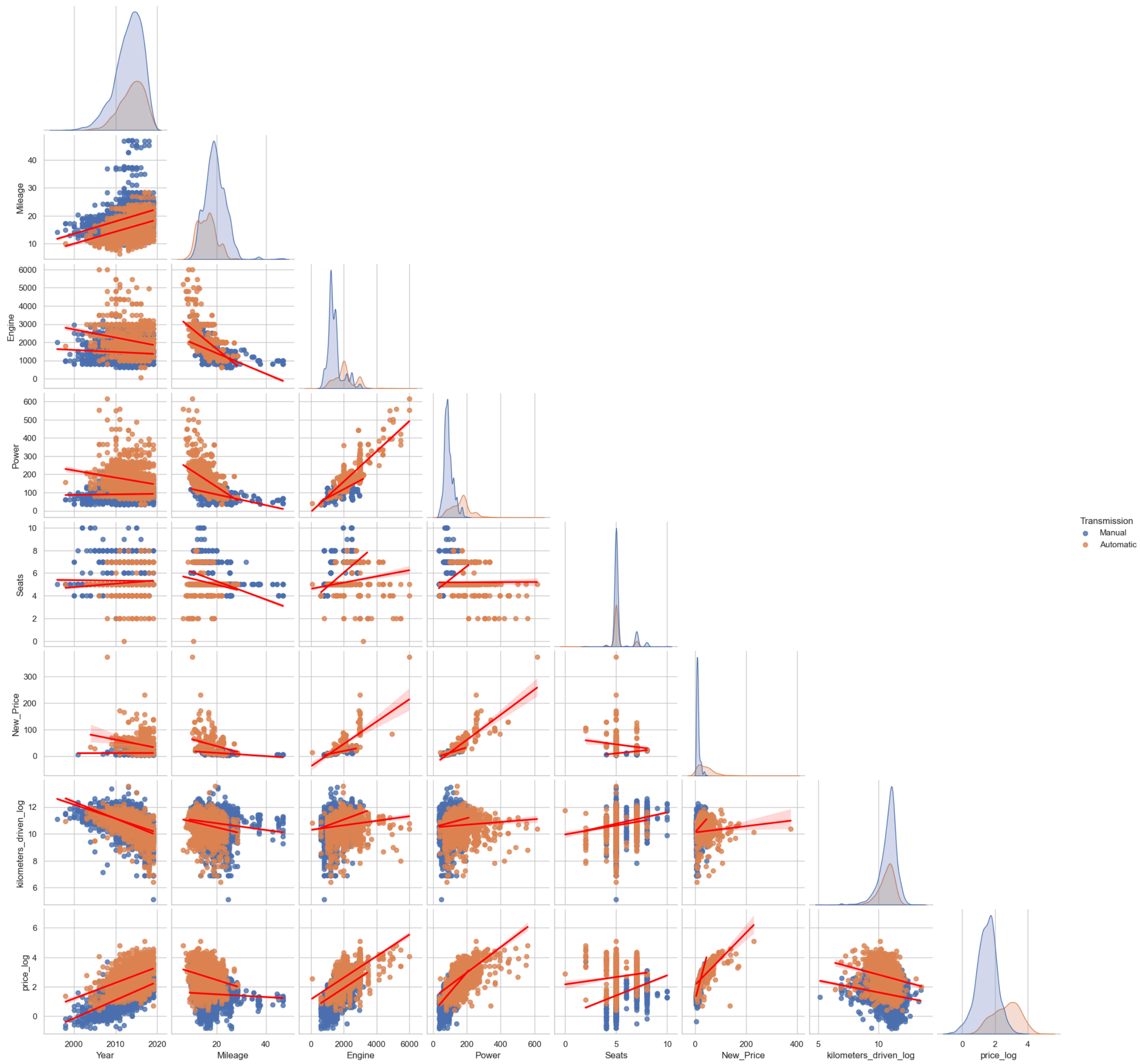
Bivariate Analysis

1. Pair plot

A pair plot allows us to see both distribution of single variables and relationships between two variables

Note: Use log transformed values 'kilometers_driven_log' and 'price_log'

In [ ]:
```
# Let us plot pair plot for the variables. We can include the log transforma

sns.set(style='ticks', color_codes=True)
sns.set(rc={'figure.figsize':(15,15)}) # Designing the size of the pairplot
sns.set_style("whitegrid")

sns.pairplot(data.drop(['Kilometers_Driven','Price'],axis = 1), kind = 'reg'
plt.show()
```



Observations

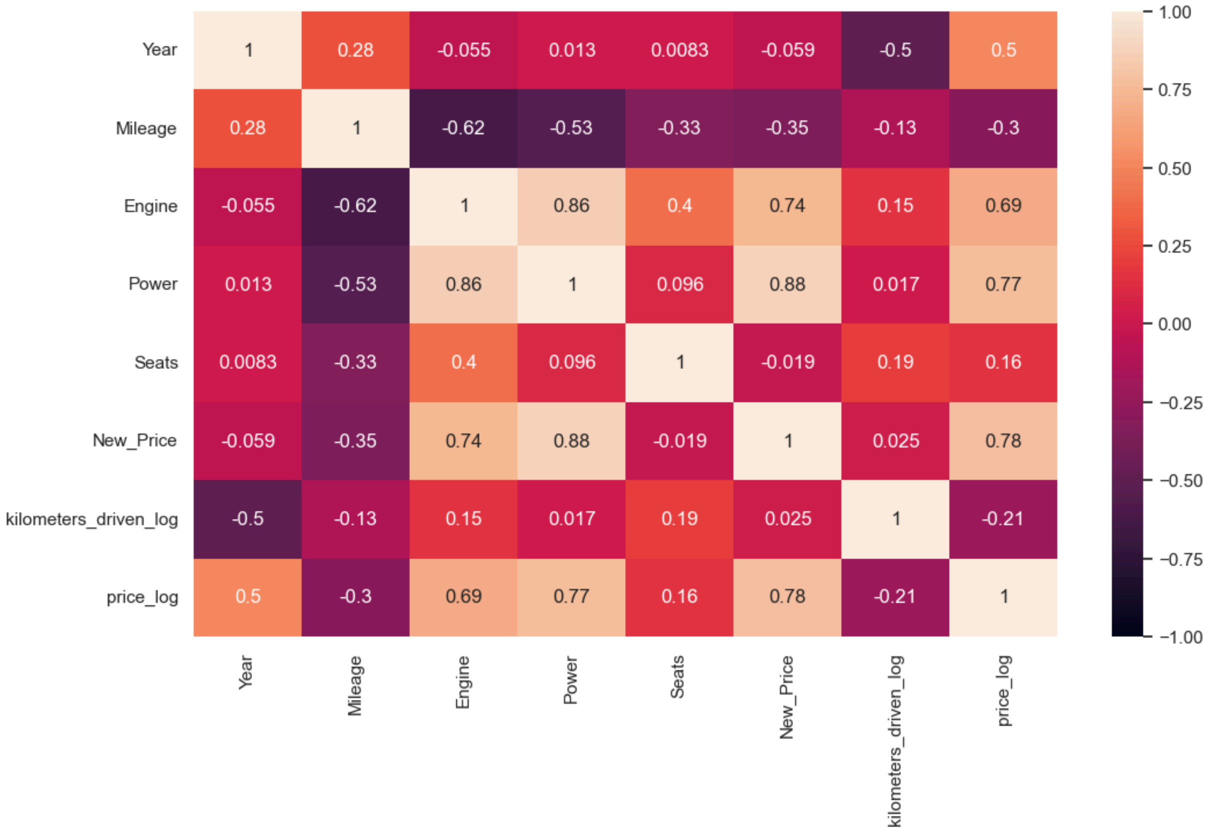
Zooming into these plots gives us a lot of information -

- Contrary to intuition, **Kilometers Driven** have **no relationship** with price
- Price has a **positive relationship with Year**. Newer the car, the higher the price
- 2 seater cars are all luxury variants. Cars with 8-10 seats are exclusively mid to high range
- Mileage does not seem to show much relationship with the price of used cars
- **Engine displacement and Power** of the car have a **positive relationship** with the price
- **New Price** and Used Car Price are also **positively correlated**, which is expected
- Kilometers Driven has a peculiar relationship with the Year variable. Generally, the newer the car lesser the distance it has traveled, but this is not always true
- Mileage and power of newer cars is increasing owing to advancement in technology
- **Mileage** has a **negative correlation** with engine displacement and power. More powerful the engine, the more fuel it consumes in general

2. Heat map

Heat map shows a 2D correlation matrix between two discrete features.

In [ ]:
```
# We can include the log transformation values and drop the original skewed
plt.figure(figsize = (12, 7))
sns.set_style("whitegrid")
# limits the values for 2 decimals
pd.set_option('display.float_format', lambda x: '%.2f' % x)

sns.heatmap(data.drop(['Kilometers_Driven','Price'],axis = 1).corr(numeric_c
plt.show()
```



- Power and engine are important predictors of price
- New_price is also a significant predictor of price

3. Box plot

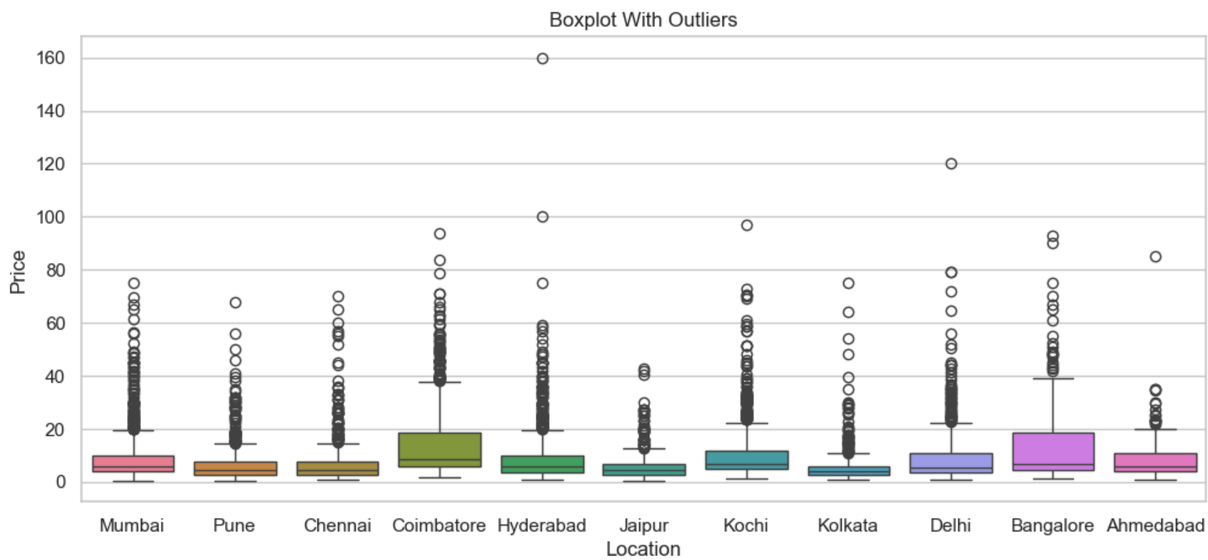
Performing Bi-variate analysis using Boxplot

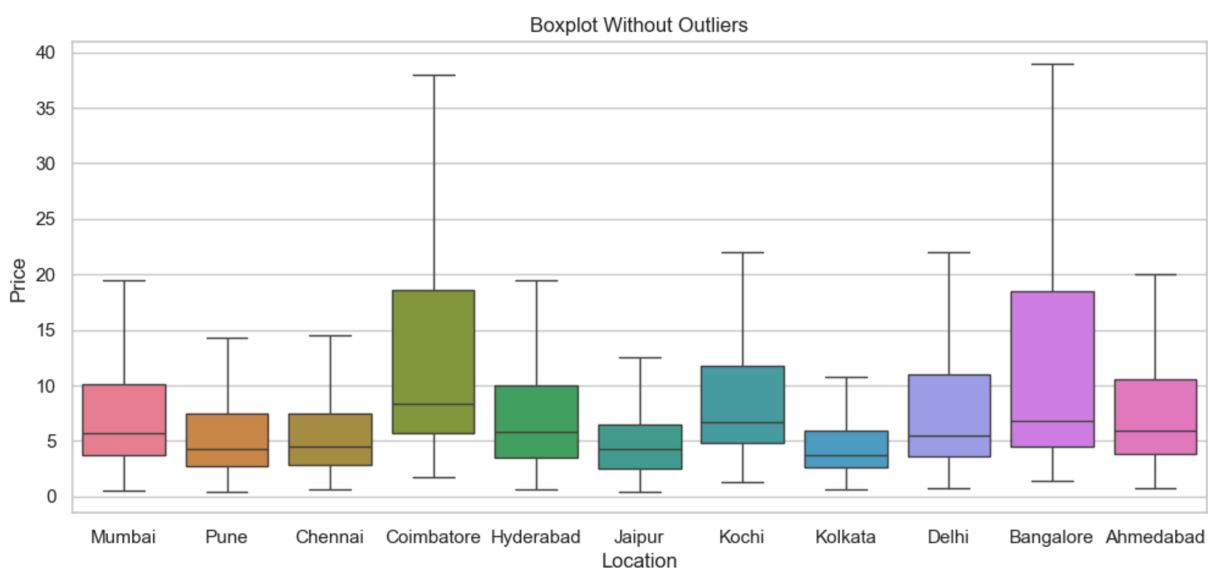
In [ ]:
```
# Function to plot boxplot w.r.t Price
def boxplot(z): # Here, z is the name of the column that you want to visuali

    # Boxplot with outliers
    plt.figure(figsize = (12, 5)) # setting size of boxplot
    # Put color legend on the column z
```

```
    plt.title('Boxplot With Outliers')
    sns.boxplot(x = z, y = data['Price'], hue=z) # Defining x and y
    plt.show()

    # Boxplot without outliers
    plt.figure(figsize = (12, 5))
    plt.title('Boxplot Without Outliers')
    sns.boxplot(x = z, y = data['Price'], showfliers = False, hue=z) # Turni
    plt.show()
```

Box Plot : Price vs Location

In [ ]:
```
boxplot(data['Location'])
```



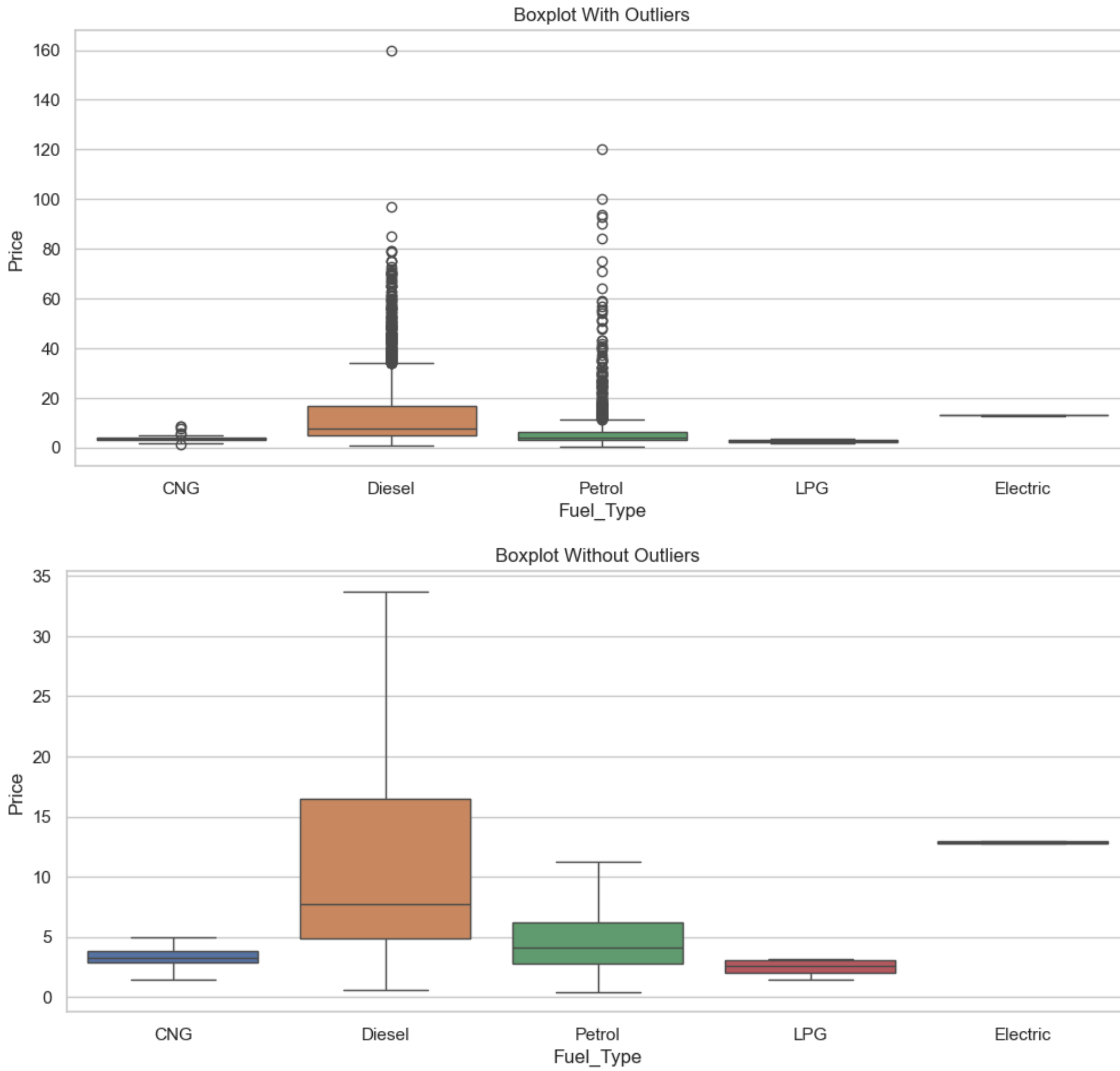


Observation

- Price of used cars has a large IQR in Coimbatore and Bangalore

Box Plot : Price vs Fuel_Type

In [ ]:
```
boxplot(data['Fuel_Type'])
```

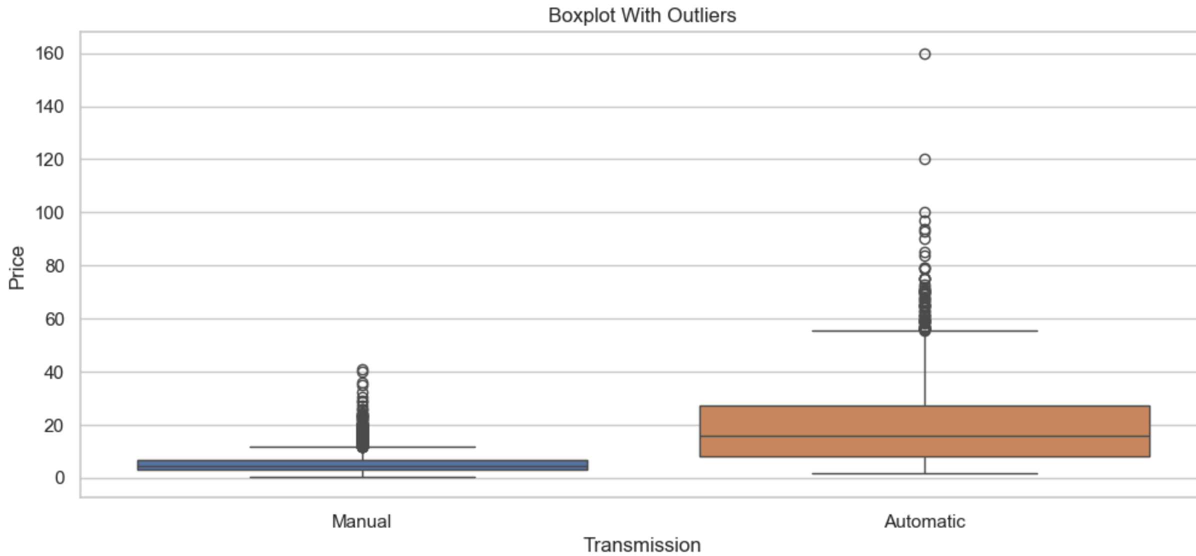


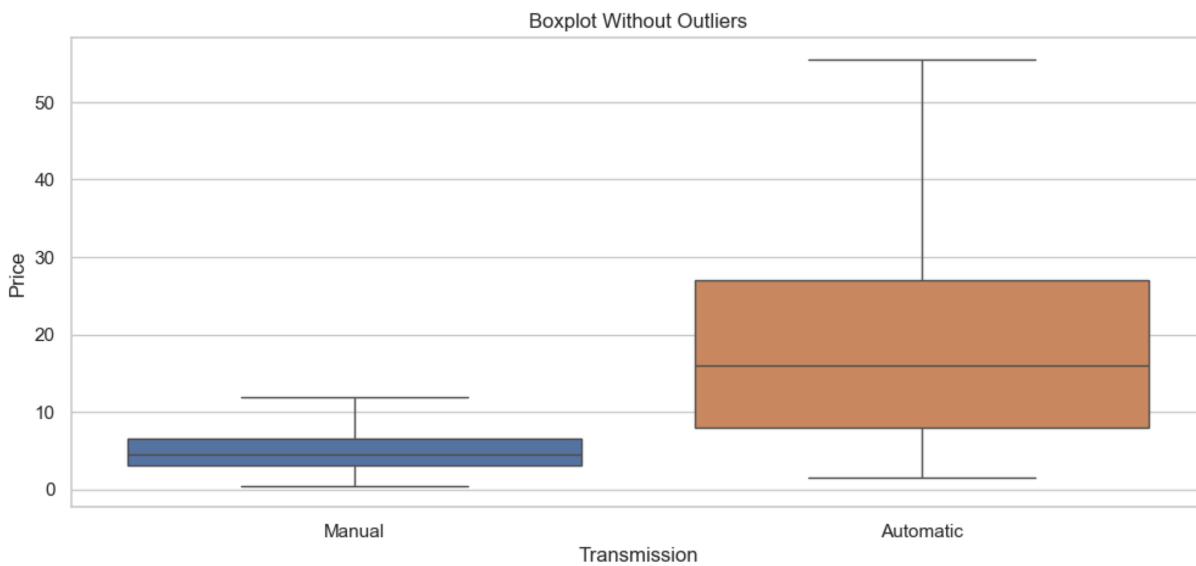
Observations

- Diesel cars are costlier than Petrol cars
- Electric cars are costlier than CNG and LPG cars

Box Plot : Price vs Transmission

In [ ]:
```
boxplot(data['Transmission'])
```



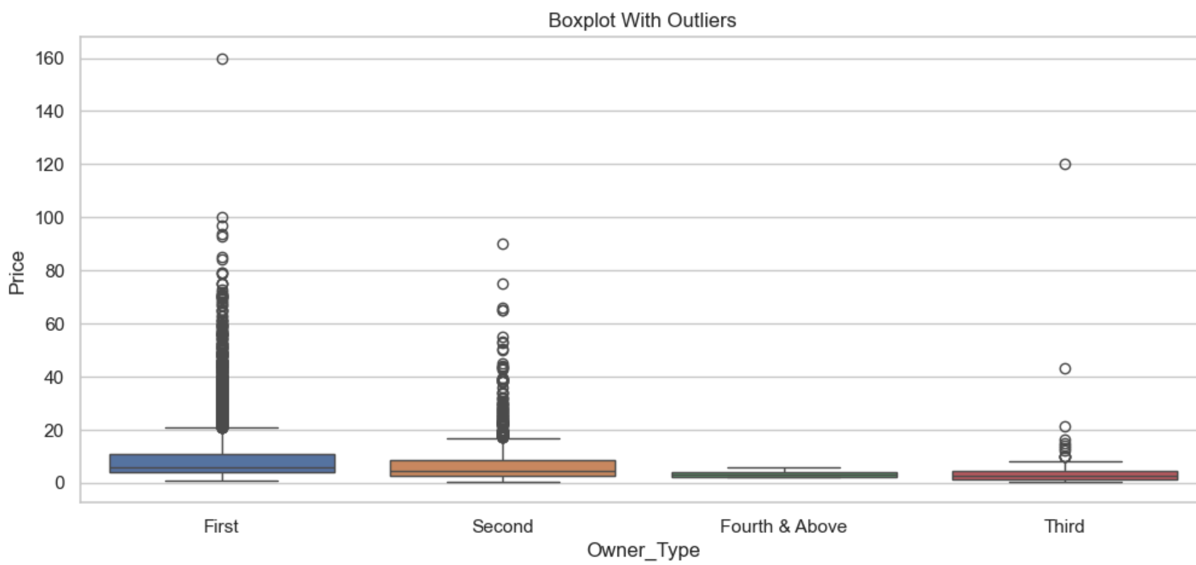


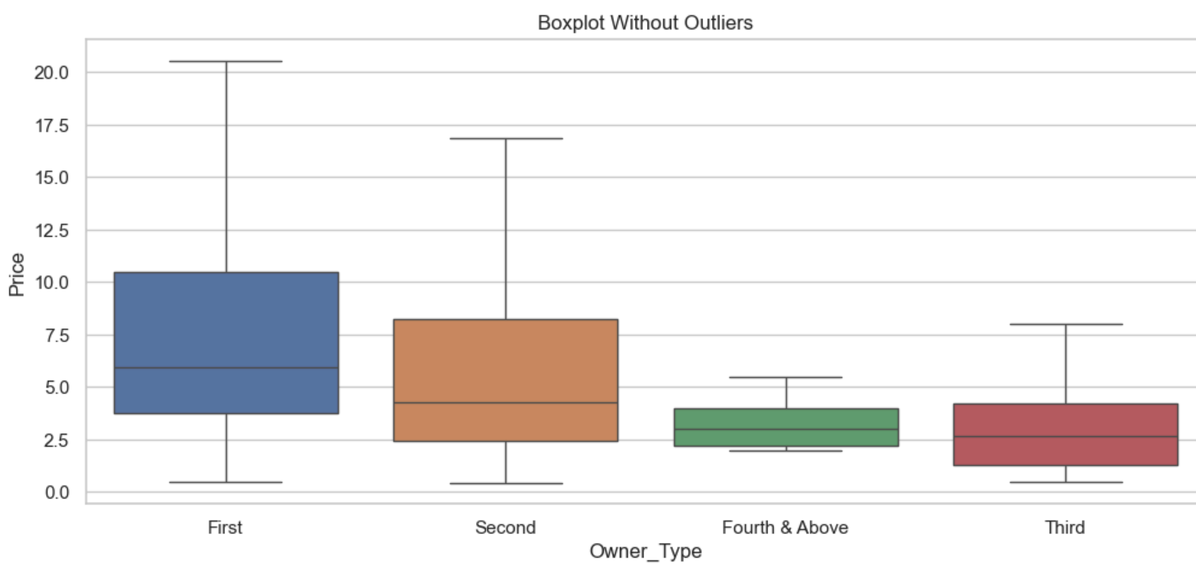
Observation

- Automatic transmission cars are very costly as compared to cars with manual transmission

Box Plot : Price vs Owner_Type

In [ ]:
```
boxplot(data['Owner_Type'])
```





Observation

- Cars with fewer previous owners tend to have higher prices. Notably, third-owner cars might exhibit outliers in price due to the presence of luxury vehicles in this category.

Feature engineering

The `Name` column, which includes both the brand name and model name of each vehicle, contains too many unique values. This high level of uniqueness limits its usefulness for predictive analysis.

In [ ]:
```
data["Name"].nunique()
```

Out[ ]:
```
2041
```

-

The 'car names' column contains 2041 unique names, making it a poor predictor of price in our current dataset. To improve this, we can process the column to extract meaningful information, which should help reduce the number of unique levels and potentially enhance the predictive power of this feature.

1. Car Brand Name

In [ ]:
```
data.head(2)
```

Out[ ]:

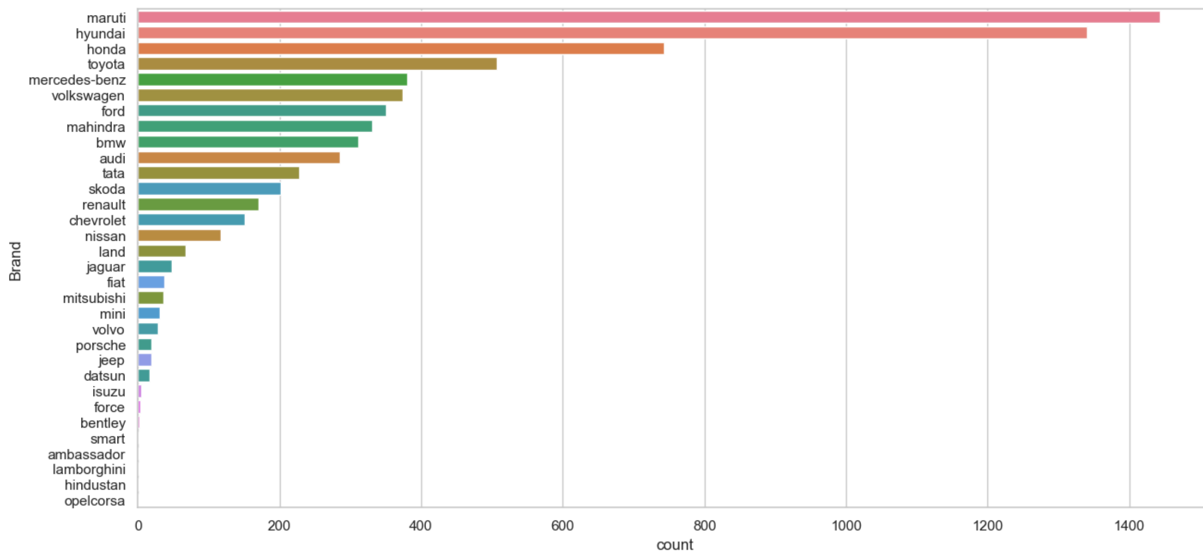
	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type
0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First
1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First

In [ ]:
```
# Extract Brand Names
data["Brand"] = data["Name"].apply(lambda x: x.split(" ")[0].lower()) # The

# Check the data
data["Brand"].value_counts()
```

```
Out[ ]: Brand
maruti           1444
hyundai          1340
honda             743
toyota            507
mercedes-benz     380
volkswagen        374
ford              351
mahindra          331
bmw               311
audi              285
tata              228
skoda             202
renault           170
chevrolet         151
nissan            117
land               67
jaguar             48
fiat               38
mitsubishi         36
mini               31
volvo              28
porsche            19
jeep               19
datsun             17
isuzu               5
force               3
bentley             2
smart               1
ambassador          1
lamborghini         1
hindustan           1
opelcorsa           1
Name: count, dtype: int64
```

Checking the brand of vehicle that appears most frequently in the dataset

```
In [ ]: plt.figure(figsize = (15, 7))
sns.countplot(y = "Brand", data = data, order = data["Brand"].value_counts()
plt.show()
```



Observation

- Most frequent brands in our data are Maruti and Hyundai

2. Car Model Name

When we look at the name of the vehicle, first word in the sentence is the brand, second word is the model, etc., so we need to capture the second word which is the model of the vehicle. Python index starts at 0, so the split value 1 will be the model of the vehicle

In [ ]:
```
# Extract Model Names
data["Model"] = data["Name"].apply(lambda x: x.split(" ")[1].lower()) # Extr

# Check the data
data["Model"].value_counts()
```

```
Out[ ]: Model
swift          418
city           318
i20            303
innova         203
verna          200
alto           183
grand          183
i10            181
wagon          178
polo           178
xuv500         131
vento          129
amaze          127
new            119
creta          118
fortuner       118
ecosport       117
figo           112
3              109
e-class        108
duster          97
santro          95
a4              90
5               86
ertiga          86
corolla         83
ciaz            83
brio            80
etios           80
eon             79
ritz            78
baleno          75
jazz            70
scorpio         69
xcent           68
rover           67
celerio         66
a6              66
rapid           58
superb          58
vitara          55
indica          54
beat            54
fiesta          49
micra           45
sx4             44
kwid            44
endeavour       43
q7              39
civic           39
q5              38
laura           36
indigo          36
x1              36
xf              36
```

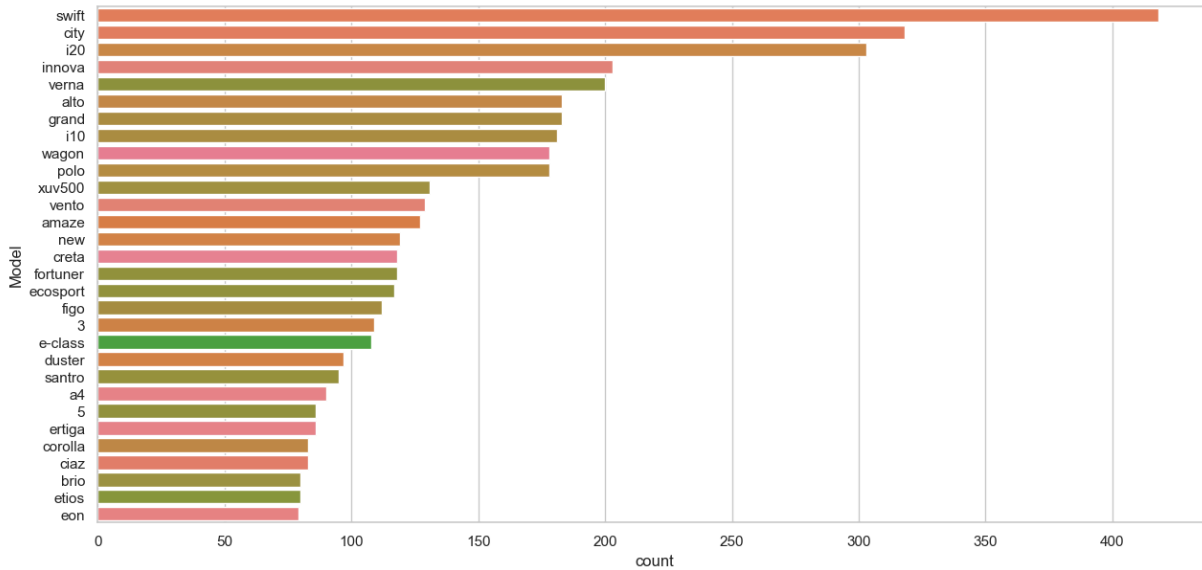
accord	35
q3	35
zen	33
octavia	33
elantra	32
sunny	32
cr-v	31
nano	31
xylo	30
terrano	30
pajero	29
jetta	28
dzire	28
cooper	28
m-class	27
zest	24
accent	24
x5	24
mobilio	24
cruze	23
omni	23
ameo	22
a-star	22
eeco	21
bolero	21
s	20
gla	20
kuv	20
manza	19
compass	19
elite	18
ikon	18
7	17
santa	17
x3	17
tiago	17
aveo	17
800	15
spark	15
linea	15
cla	15
camry	15
gl-class	15
b	15
ssangyong	14
gle	14
enjoy	13
optra	13
a	12
tuv	12
getz	12
fabia	11
ignis	10
passat	10
thar	10
sail	10

6	9
go	9
s60	9
safari	9
glc	9
pulse	9
sonata	9
panamera	8
brv	8
cayenne	8
x6	8
redi-go	7
xc60	7
punto	7
grande	7
tigor	7
avventura	6
xj	6
quanto	6
esteem	6
sumo	6
a3	6
yeti	6
verito	6
teana	5
br-v	5
qualis	5
wrv	5
xe	5
s-class	5
v40	5
scala	5
crosspolo	5
tucson	5
r-class	4
captur	4
aspire	4
hexa	4
fluence	4
koleos	4
xenon	4
bolt	4
xc90	4
freestyle	4
captiva	3
a7	3
1	3
classic	3
slc	3
s80	3
a8	3
renault	3
tt	3
one	3
slk-class	3
lodgy	3

```
nuvosport        3
c-class          3
d-max            3
x-trail          3
s-cross          3
nexon            3
tavera           3
estilo           3
lancer           2
z4               2
clubman          2
versa            2
jeep             2
logan            2
gls              2
rs5              2
cedia            2
outlander        2
cayman           2
continental      1
prius            1
countryman       1
wr-v             1
gallardo         1
1000             1
f                1
motors           1
flying           1
land             1
mu               1
370z             1
abarth           1
sl-class         1
fusion           1
siena            1
mux              1
tiguan           1
montero          1
petra            1
beetle           1
venture          1
xuv300           1
platinum         1
evalia           1
boxster          1
cls-class        1
fortwo           1
redi             1
e                1
mustang          1
1.4gsi           1
Name: count, dtype: int64
```

Creating countplot for a clearer and more understandable view of the information

In [ ]:
```
plt.figure(figsize = (15, 7))
sns.countplot(y = "Model", data = data, order = data["Model"].value_counts()
plt.show()
```



Observations

- It is clear from the above charts that our dataset contains used cars from luxury as well as budget-friendly brands
- We have extracted brand name and model name, we get a better understanding of the cars we have in our data

We need to also understand, on an average, **what's the price of the vehicle for a specific brand for better understanding of the data**

In [ ]:
```
# Grouping the data based on the brand to find the average price of cars per
data.groupby(["Brand"])["Price"].mean().sort_values(ascending = False)
```

Out[ ]:

```
Brand
lamborghini      120.00
bentley           59.00
porsche           48.35
land              39.26
jaguar            37.63
mini              26.90
mercedes-benz     26.81
audi              25.54
bmw               25.09
volvo             18.80
jeep              18.72
isuzu             14.70
toyota            11.58
mitsubishi        11.06
force              9.33
mahindra           8.05
skoda              7.56
ford               6.89
renault            5.80
honda              5.41
hyundai            5.34
volkswagen         5.31
nissan             4.74
maruti             4.52
tata               3.56
fiat               3.27
datsun             3.05
chevrolet          3.04
smart              3.00
ambassador         1.35
hindustan           NaN
opelcorsa           NaN
Name: Price, dtype: float64
```

Observations

- The output closely matches our expectations in terms of brand ranking. The average price of a used Lamborghini is 120 Lakhs, with other luxury brands following in descending order.
- Towards the lower end, we observe more affordable brands.
- We notice some missing data, which we will handle in subsequent steps.

Missing value treatment

In [ ]:

```
# Summing up the number of rows with missing values for each column.
data.isnull().sum()
```

Out[ ]:
```
Name                      0
Location                  0
Year                      0
Kilometers_Driven         0
Fuel_Type                 0
Transmission              0
Owner_Type                0
Mileage                  83
Engine                   46
Power                   175
Seats                    53
New_Price              6246
Price                  1234
kilometers_driven_log     0
price_log              1234
Brand                     0
Model                     0
dtype: int64
```

Observations

- Engine displacement information of 46 observations is missing and a maximum power of 175 entries is missing
- Information about the number of seats is not available for 53 entries
- New Price as we saw earlier has a huge missing count
- Price is also missing for 1234 entries. Since price is the response variable that we want to predict, we will have to drop these rows while building the model

Missing values in Seats

In [ ]:
```
# Checking the actual rows where the Seats column has missing values.
data[data['Seats'].isnull()]
```

Out[ ]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owı
194	Honda City 1.5 GXI	Ahmedabad	2007	60006	Petrol	Manual	
208	Maruti Swift 1.3 VXi	Kolkata	2010	42001	Petrol	Manual	
229	Ford Figo Diesel	Bangalore	2015	70436	Diesel	Manual	
733	Maruti Swift 1.3 VXi	Chennai	2006	97800	Petrol	Manual	
749	Land Rover Range Rover 3.0 D	Mumbai	2008	55001	Diesel	Automatic	
1294	Honda City 1.3 DX	Delhi	2009	55005	Petrol	Manual	
1327	Maruti Swift 1.3 ZXI	Hyderabad	2015	50295	Petrol	Manual	
1385	Honda City 1.5 GXI	Pune	2004	115000	Petrol	Manual	
1460	Land Rover Range Rover Sport 2005 2012 Sport	Coimbatore	2008	69078	Petrol	Manual	
1917	Honda City 1.5 EXI	Jaipur	2005	88000	Petrol	Manual	
2074	Maruti Swift 1.3 LXI	Pune	2011	24255	Petrol	Manual	
2096	Hyundai Santro LP zipPlus	Coimbatore	2004	52146	Petrol	Manual	
2264	Toyota Etios Liva V	Pune	2012	24500	Petrol	Manual	
2325	Maruti Swift 1.3	Pune	2015	67000	Petrol	Manual	

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Ow
	VXI ABS						
2335	Maruti Swift 1.3 VXi	Mumbai	2007	55000	Petrol	Manual	
2369	Maruti Estilo LXI	Chennai	2008	56000	Petrol	Manual	
2530	BMW 5 Series 520d Sedan	Kochi	2014	64158	Diesel	Automatic	
2542	Hyundai Santro GLS II - Euro II	Bangalore	2011	65000	Petrol	Manual	
2623	BMW 5 Series 520d Sedan	Pune	2012	95000	Diesel	Automatic	
2668	Maruti Swift 1.3 VXi	Kolkata	2014	32986	Petrol	Manual	
2737	Maruti Wagon R Vx	Jaipur	2001	200000	Petrol	Manual	
2780	Hyundai Santro GLS II - Euro II	Pune	2009	100000	Petrol	Manual	
2842	Hyundai Santro GLS II - Euro II	Bangalore	2012	43000	Petrol	Manual	
3272	BMW 5 Series 520d Sedan	Mumbai	2008	81000	Diesel	Automatic	
3404	Maruti Swift 1.3 VXi	Jaipur	2006	125000	Petrol	Manual	
3520	BMW 5 Series 520d Sedan	Delhi	2012	90000	Diesel	Automatic	
3522	Hyundai Santro GLS II - Euro II	Kochi	2012	66400	Petrol	Manual	

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owi
3800	Ford Endeavour Hurricane LE	Mumbai	2012	129000	Diesel	Automatic	
3810	Honda CR-V AT With Sun Roof	Kolkata	2013	27000	Petrol	Automatic	
3882	Maruti Estilo LXI	Kolkata	2010	40000	Petrol	Manual	
4011	Fiat Punto 1.3 Emotion	Pune	2011	45271	Diesel	Manual	
4152	Land Rover Range Rover 3.0 D	Mumbai	2003	75000	Diesel	Automatic	
4229	Hyundai Santro Xing XG	Bangalore	2005	79000	Petrol	Manual	
4577	BMW 5 Series 520d Sedan	Delhi	2012	72000	Diesel	Automatic	
4604	Honda Jazz Select Edition	Pune	2011	98000	Petrol	Manual	
4697	Fiat Punto 1.2 Dynamic	Kochi	2017	17941	Petrol	Manual	
4712	Hyundai Santro Xing XG	Pune	2003	80000	Petrol	Manual	
4952	Fiat Punto 1.4 Emotion	Kolkata	2010	47000	Petrol	Manual	
5015	Maruti Swift 1.3 VXi	Delhi	2006	63000	Petrol	Manual	
5185	Maruti Swift 1.3 LXI	Delhi	2012	52000	Petrol	Manual	

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owı
5270	Honda City 1.5 GXI	Bangalore	2002	53000	Petrol	Manual	
5893	Maruti Estilo LXI	Chennai	2008	51000	Petrol	Manual	
6042	Skoda Laura 1.8 TSI Ambition	Bangalore	2009	72000	Petrol	Manual	
6541	Toyota Etios Liva Diesel TRD Sportivo	Bangalore	2012	56600	Diesel	Manual	
6544	Hyundai i20 new Sportz AT 1.4	Bangalore	2012	58000	Petrol	Automatic	
6633	Mahindra TUV 300 P4	Kolkata	2016	27000	Diesel	Manual	
6643	BMW 5 Series 520d Sedan	Bangalore	2009	150000	Diesel	Automatic	
6651	Maruti Swift 1.3 VXi	Kolkata	2015	36009	Petrol	Manual	
6677	Fiat Punto 1.4 Emotion	Jaipur	2010	65000	Petrol	Manual	
6685	Maruti Swift 1.3 VXi	Pune	2010	115000	Petrol	Manual	
6880	BMW 5 Series 520d Sedan	Chennai	2009	95000	Diesel	Automatic	
6902	Toyota Etios Liva V	Kochi	2012	59311	Petrol	Manual	
6957	Honda Jazz 2020 Petrol	Kochi	2019	11574	Petrol	Manual	

In [ ]:

```
# We will impute these missing values one by one by taking the median number
# using the brand and model name.
```

```
data.groupby(["Brand", "Model"], as_index = False)["Seats"].median() # Check
```

Out[ ]:

	Brand	Model	Seats
0	ambassador	classic	5.00
1	audi	a3	5.00
2	audi	a4	5.00
3	audi	a6	5.00
4	audi	a7	5.00
5	audi	a8	5.00
6	audi	q3	5.00
7	audi	q5	5.00
8	audi	q7	7.00
9	audi	rs5	4.00
10	audi	tt	2.00
11	bentley	continental	4.00
12	bentley	flying	5.00
13	bmw	1	5.00
14	bmw	3	5.00
15	bmw	5	5.00
16	bmw	6	4.00
17	bmw	7	5.00
18	bmw	x1	5.00
19	bmw	x3	5.00
20	bmw	x5	5.00
21	bmw	x6	4.00
22	bmw	z4	2.00
23	chevrolet	aveo	5.00
24	chevrolet	beat	5.00
25	chevrolet	captiva	7.00
26	chevrolet	cruze	5.00
27	chevrolet	enjoy	8.00
28	chevrolet	optra	5.00
29	chevrolet	sail	5.00
30	chevrolet	spark	5.00
31	chevrolet	tavera	10.00

	Brand	Model	Seats
32	datsun	go	5.00
33	datsun	redi	5.00
34	datsun	redi-go	5.00
35	fiat	abarth	4.00
36	fiat	avventura	5.00
37	fiat	grande	5.00
38	fiat	linea	5.00
39	fiat	petra	5.00
40	fiat	punto	5.00
41	fiat	siena	5.00
42	force	one	7.00
43	ford	aspire	5.00
44	ford	classic	5.00
45	ford	ecosport	5.00
46	ford	endeavour	7.00
47	ford	fiesta	5.00
48	ford	figo	5.00
49	ford	freestyle	5.00
50	ford	fusion	5.00
51	ford	ikon	5.00
52	ford	mustang	4.00
53	hindustan	motors	5.00
54	honda	accord	5.00
55	honda	amaze	5.00
56	honda	br-v	7.00
57	honda	brio	5.00
58	honda	brv	7.00
59	honda	city	5.00
60	honda	civic	5.00
61	honda	cr-v	5.00
62	honda	jazz	5.00
63	honda	mobilio	7.00

	Brand	Model	Seats
64	honda	wr-v	5.00
65	honda	wrv	5.00
66	hyundai	accent	5.00
67	hyundai	creta	5.00
68	hyundai	elantra	5.00
69	hyundai	elite	5.00
70	hyundai	eon	5.00
71	hyundai	getz	5.00
72	hyundai	grand	5.00
73	hyundai	i10	5.00
74	hyundai	i20	5.00
75	hyundai	santa	7.00
76	hyundai	santro	5.00
77	hyundai	sonata	5.00
78	hyundai	tucson	5.00
79	hyundai	verna	5.00
80	hyundai	xcent	5.00
81	isuzu	d-max	5.00
82	isuzu	mu	7.00
83	isuzu	mux	7.00
84	jaguar	f	2.00
85	jaguar	xe	5.00
86	jaguar	xf	5.00
87	jaguar	xj	4.50
88	jeep	compass	5.00
89	lamborghini	gallardo	2.00
90	land	rover	5.00
91	mahindra	bolero	7.00
92	mahindra	e	5.00
93	mahindra	jeep	6.00
94	mahindra	kuv	6.00
95	mahindra	logan	5.00

	Brand	Model	Seats
96	mahindra	nuvosport	7.00
97	mahindra	quanto	7.00
98	mahindra	renault	5.00
99	mahindra	scorpio	8.00
100	mahindra	ssangyong	7.00
101	mahindra	thar	6.00
102	mahindra	tuv	7.00
103	mahindra	verito	5.00
104	mahindra	xuv300	5.00
105	mahindra	xuv500	7.00
106	mahindra	xylo	7.50
107	maruti	1000	5.00
108	maruti	800	4.00
109	maruti	a-star	5.00
110	maruti	alto	5.00
111	maruti	baleno	5.00
112	maruti	celerio	5.00
113	maruti	ciaz	5.00
114	maruti	dzire	5.00
115	maruti	eeco	5.00
116	maruti	ertiga	7.00
117	maruti	esteem	5.00
118	maruti	estilo	NaN
119	maruti	grand	5.00
120	maruti	ignis	5.00
121	maruti	omni	5.00
122	maruti	ritz	5.00
123	maruti	s	5.00
124	maruti	s-cross	5.00
125	maruti	swift	5.00
126	maruti	sx4	5.00
127	maruti	versa	8.00

	Brand	Model	Seats
128	maruti	vitara	5.00
129	maruti	wagon	5.00
130	maruti	zen	5.00
131	mercedes-benz	a	5.00
132	mercedes-benz	b	5.00
133	mercedes-benz	c-class	5.00
134	mercedes-benz	cla	5.00
135	mercedes-benz	cls-class	4.00
136	mercedes-benz	e-class	5.00
137	mercedes-benz	gl-class	7.00
138	mercedes-benz	gla	5.00
139	mercedes-benz	glc	5.00
140	mercedes-benz	gle	5.00
141	mercedes-benz	gls	7.00
142	mercedes-benz	m-class	5.00
143	mercedes-benz	new	5.00
144	mercedes-benz	r-class	7.00
145	mercedes-benz	s	5.00
146	mercedes-benz	s-class	5.00
147	mercedes-benz	sl-class	2.00
148	mercedes-benz	slc	2.00
149	mercedes-benz	slk-class	2.00
150	mini	clubman	5.00
151	mini	cooper	4.00
152	mini	countryman	5.00
153	mitsubishi	cedia	5.00
154	mitsubishi	lancer	5.00
155	mitsubishi	montero	7.00
156	mitsubishi	outlander	5.00
157	mitsubishi	pajero	6.00
158	nissan	370z	2.00
159	nissan	evalia	7.00

	Brand	Model	Seats
160	nissan	micra	5.00
161	nissan	sunny	5.00
162	nissan	teana	5.00
163	nissan	terrano	5.00
164	nissan	x-trail	5.00
165	opelcorsa	1.4gsi	5.00
166	porsche	boxster	2.00
167	porsche	cayenne	5.00
168	porsche	cayman	2.00
169	porsche	panamera	4.00
170	renault	captur	5.00
171	renault	duster	5.00
172	renault	fluence	5.00
173	renault	koleos	5.00
174	renault	kwid	5.00
175	renault	lodgy	8.00
176	renault	pulse	5.00
177	renault	scala	5.00
178	skoda	fabia	5.00
179	skoda	laura	5.00
180	skoda	octavia	5.00
181	skoda	rapid	5.00
182	skoda	superb	5.00
183	skoda	yeti	5.00
184	smart	fortwo	2.00
185	tata	bolt	5.00
186	tata	hexa	7.00
187	tata	indica	5.00
188	tata	indigo	5.00
189	tata	manza	5.00
190	tata	nano	4.00
191	tata	new	7.00

	Brand	Model	Seats
192	tata	nexon	5.00
193	tata	safari	7.00
194	tata	sumo	7.00
195	tata	tiago	5.00
196	tata	tigor	5.00
197	tata	venture	8.00
198	tata	xenon	5.00
199	tata	zest	5.00
200	toyota	camry	5.00
201	toyota	corolla	5.00
202	toyota	etios	5.00
203	toyota	fortuner	7.00
204	toyota	innova	7.00
205	toyota	land	7.00
206	toyota	platinum	5.00
207	toyota	prius	5.00
208	toyota	qualis	10.00
209	volkswagen	ameo	5.00
210	volkswagen	beetle	4.00
211	volkswagen	crosspolo	5.00
212	volkswagen	jetta	5.00
213	volkswagen	passat	5.00
214	volkswagen	polo	5.00
215	volkswagen	tiguan	5.00
216	volkswagen	vento	5.00
217	volvo	s60	5.00
218	volvo	s80	5.00
219	volvo	v40	5.00
220	volvo	xc60	5.00
221	volvo	xc90	7.00

It looks appropriate to fill the missing values in the 'seats' column with the median number of seats for each model

In [ ]:
```
# Impute missing Seats with the median of each model
data["Seats"] = data.groupby(["Brand", "Model"])["Seats"].transform(lambda x
```

Checking how many missing and non-missing rows do we have in the seat column by now

In [ ]:
```
print(f"{data['Seats'].isnull().value_counts()[0]} rows have non-missing val
```

```
7249 rows have non-missing values in Seats column
3 rows have missing values in Seats column
```

Now it's time to investigate why there are still missing rows in the 'Seats' column

In [ ]:
```
data[data['Seats'].isnull()]
```

Out[ ]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Ty
2369	Maruti Estilo LXI	Chennai	2008	56000	Petrol	Manual	Seco
3882	Maruti Estilo LXI	Kolkata	2010	40000	Petrol	Manual	Seco
5893	Maruti Estilo LXI	Chennai	2008	51000	Petrol	Manual	Seco

Based on the domain knowledge, we can fill out the most appropriate value for the missing seats for those 3 rows

In [ ]:
```
# Maruti Estilo can accomodate 5
data["Seats"] = data["Seats"].fillna(5.0)
```

In [ ]:
```
data.isnull().sum()
```

Out[ ]:
```
Name                      0
Location                  0
Year                      0
Kilometers_Driven         0
Fuel_Type                 0
Transmission              0
Owner_Type                0
Mileage                  83
Engine                   46
Power                   175
Seats                     0
New_Price              6246
Price                  1234
kilometers_driven_log     0
price_log              1234
Brand                     0
Model                     0
dtype: int64
```

Above info shows that there is no more missing data in seat column, however, we got some other columns with the missing values

Missing values for Mileage

In [ ]:
```
data[data['Mileage'].isnull()]
```

Out[ ]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Ov
14	Land Rover Freelander 2 TD4 SE	Pune	2012	85000	Diesel	Automatic	
67	Mercedes-Benz C-Class Progressive C 220d	Coimbatore	2019	15369	Diesel	Automatic	
79	Hyundai Santro Xing XL	Hyderabad	2005	87591	Petrol	Manual	
194	Honda City 1.5 GXI	Ahmedabad	2007	60006	Petrol	Manual	
229	Ford Figo Diesel	Bangalore	2015	70436	Diesel	Manual	
262	Hyundai Santro Xing XL	Hyderabad	2006	99000	Petrol	Manual	
307	Hyundai Santro Xing XL	Chennai	2006	58000	Petrol	Manual	
424	Volkswagen Jetta 2007-2011 1.9 L TDI	Hyderabad	2010	42021	Diesel	Manual	
443	Hyundai Santro GLS I - Euro I	Coimbatore	2012	50243	Petrol	Manual	
544	Mercedes-Benz New C-Class Progressive C 200	Kochi	2019	13190	Petrol	Automatic	
631	Hyundai Santro LS zipPlus	Chennai	2002	70000	Petrol	Manual	
647	Hyundai Santro Xing XP	Jaipur	2004	200000	Petrol	Manual	
707	Mercedes-Benz M-Class ML 350 4Matic	Pune	2014	120000	Diesel	Automatic	
749	Land Rover Range Rover 3.0 D	Mumbai	2008	55001	Diesel	Automatic	

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Ov
915	Smart Fortwo CDI AT	Pune	2008	103000	Diesel	Automatic	
962	Mercedes-Benz C-Class Progressive C 220d	Mumbai	2018	8682	Diesel	Automatic	
996	Hyundai Santro Xing GL	Pune	2008	93000	Petrol	Manual	
1059	Hyundai Santro Xing GL	Hyderabad	2010	58163	Petrol	Manual	
1259	Land Rover Freelander 2 TD4 S	Bangalore	2010	125000	Diesel	Automatic	
1271	Hyundai Santro GLS I - Euro II	Jaipur	2009	89000	Petrol	Manual	
1308	Mercedes-Benz M-Class ML 350 4Matic	Bangalore	2014	33000	Diesel	Automatic	
1345	Maruti Baleno Vxi	Pune	2005	70000	Petrol	Manual	
1354	Hyundai Santro Xing GL	Kochi	2011	20842	Petrol	Manual	
1385	Honda City 1.5 GXI	Pune	2004	115000	Petrol	Manual	
1419	Hyundai Santro Xing XL	Chennai	2007	82000	Petrol	Manual	
1460	Land Rover Range Rover Sport 2005 2012 Sport	Coimbatore	2008	69078	Petrol	Manual	
1764	Mercedes-Benz M-Class ML 350 4Matic	Pune	2015	69000	Diesel	Automatic	
1857	Hyundai Santro DX	Hyderabad	2007	96000	Petrol	Manual	

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Ov
2053	Mahindra Jeep MM 550 PE	Hyderabad	2009	26000	Diesel	Manual	
2096	Hyundai Santro LP zipPlus	Coimbatore	2004	52146	Petrol	Manual	
2130	Hyundai Santro GLS I - Euro II	Coimbatore	2012	51019	Petrol	Manual	
2267	Toyota Qualis RS E2	Pune	2004	215750	Diesel	Manual	
2343	Hyundai Santro AT	Hyderabad	2006	74483	Petrol	Automatic	
2542	Hyundai Santro GLS II - Euro II	Bangalore	2011	65000	Petrol	Manual	
2597	Hyundai Santro Xing XP	Pune	2007	70000	Petrol	Manual	
2681	Skoda Superb 3.6 V6 FSI	Hyderabad	2010	54000	Petrol	Automatic	
2780	Hyundai Santro GLS II - Euro II	Pune	2009	100000	Petrol	Manual	
2842	Hyundai Santro GLS II - Euro II	Bangalore	2012	43000	Petrol	Manual	
3033	Hyundai Santro Xing XP	Jaipur	2005	120000	Petrol	Manual	
3044	Hyundai Santro Xing GL	Kolkata	2009	60170	Petrol	Manual	
3061	Hyundai Santro GS	Ahmedabad	2005	58000	Petrol	Manual	
3093	Audi A7 2011-2015 Sportback	Kolkata	2012	24720	Diesel	Automatic	
3189	Hyundai Santro GS zipDrive - Euro II	Chennai	2002	67000	Petrol	Manual	

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Ov
3210	Mercedes-Benz M-Class ML 350 4Matic	Coimbatore	2016	22769	Diesel	Automatic	
3271	Hyundai Santro Xing GL	Bangalore	2008	35268	Petrol	Manual	
3516	Hyundai Santro GLS I - Euro I	Pune	2011	65400	Petrol	Manual	
3522	Hyundai Santro GLS II - Euro II	Kochi	2012	66400	Petrol	Manual	
3645	Hyundai Santro Xing XP	Bangalore	2004	167000	Petrol	Manual	
4152	Land Rover Range Rover 3.0 D	Mumbai	2003	75000	Diesel	Automatic	
4234	Mercedes-Benz M-Class ML 350 4Matic	Chennai	2012	63000	Diesel	Automatic	
4302	Hyundai Santro Xing GL	Delhi	2012	61449	Petrol	Manual	
4412	Mercedes-Benz M-Class ML 350 4Matic	Coimbatore	2016	27833	Diesel	Automatic	
4446	Mahindra E Verito D4	Chennai	2016	50000	Electric	Automatic	
4629	Fiat Siena 1.2 ELX	Jaipur	2001	70000	Petrol	Manual	
4687	Land Rover Freelander 2 TD4 SE	Jaipur	2012	119203	Diesel	Automatic	
4704	Mercedes-Benz M-Class ML 350 4Matic	Bangalore	2015	20000	Diesel	Automatic	
4904	Toyota Prius 2009-2016 Z4	Mumbai	2011	44000	Electric	Automatic	

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Ov
5016	Land Rover Freelander 2 TD4 HSE	Delhi	2013	72000	Diesel	Automatic	
5022	Land Rover Freelander 2 TD4 SE	Hyderabad	2013	46000	Diesel	Automatic	
5119	Hyundai Santro Xing XP	Kolkata	2008	45500	Petrol	Manual	
5270	Honda City 1.5 GXI	Bangalore	2002	53000	Petrol	Manual	
5311	Land Rover Freelander 2 TD4 SE	Hyderabad	2012	139000	Diesel	Automatic	
5374	Mercedes-Benz M-Class ML 350 4Matic	Ahmedabad	2012	66000	Diesel	Automatic	
5426	Hyundai Santro Xing XL	Chennai	2006	85000	Petrol	Manual	
5529	Hyundai Santro LP - Euro II	Chennai	2005	105000	Petrol	Manual	
5647	Toyota Qualis Fleet A3	Mumbai	2001	227000	Diesel	Manual	
5875	Mercedes-Benz C-Class Progressive C 220d	Ahmedabad	2019	4000	Diesel	Automatic	
5943	Mahindra Jeep MM 540 DP	Chennai	2002	75000	Diesel	Manual	
5972	Hyundai Santro Xing GL	Mumbai	2008	65000	Petrol	Manual	
6011	Skoda Superb 3.6 V6 FSI	Hyderabad	2009	53000	Petrol	Automatic	
6090	Hyundai Santro Xing GL	Ahmedabad	2013	63831	Petrol	Manual	
6093	Hyundai Santro Xing	Bangalore	2007	47000	Petrol	Manual	

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Ov
	XL						
6177	Mercedes-Benz M-Class ML 350 4Matic	Bangalore	2012	37000	Diesel	Automatic	
6205	Hyundai Santro Xing GL	Ahmedabad	2007	78000	Petrol	Manual	
6439	Hyundai Santro GLS I - Euro II	Bangalore	2011	43189	Petrol	Manual	
6454	Hyundai Santro LS zipDrive Euro I	Chennai	2002	120000	Petrol	Manual	
6491	Mercedes-Benz M-Class ML 350 4Matic	Coimbatore	2016	22177	Diesel	Automatic	
6576	Hyundai Santro LS zipPlus	Kolkata	2002	80000	Petrol	Manual	
6633	Mahindra TUV 300 P4	Kolkata	2016	27000	Diesel	Manual	
6697	Hyundai Santro Xing XL	Jaipur	2007	85000	Petrol	Manual	
6857	Land Rover Freelander 2 TD4 SE	Mumbai	2011	87000	Diesel	Automatic	
6957	Honda Jazz 2020 Petrol	Kochi	2019	11574	Petrol	Manual	
7226	Hyundai Santro Xing GL	Ahmedabad	2014	41000	Petrol	Manual	

It is possible that if the Vehicle is ELECTRIC vehicle, then we will not have a Fuel Mileage, so let's check if the missing mileage column is for the electric vehicle

In [ ]:
```
data[data['Fuel_Type']=='Electric']
```

Out[ ]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_
4446	Mahindra E Verito D4	Chennai	2016	50000	Electric	Automatic	
4904	Toyota Prius 2009-2016 Z4	Mumbai	2011	44000	Electric	Automatic	

Observation

- 2 Electric car variants don't have entries for Mileage

In this case, we can either drop those two rows, or we can adjust the number for now, just for the analysis. Let's proceed with putting median mileage so that we can keep the record in the dataset

In [ ]:
```
# Impute missing Mileage
data["Mileage"].fillna(data['Mileage'].median(), inplace = True)
```

Let's now, further validate the missing data

In [ ]:
```
data.isnull().sum()
```

Out[ ]:
```
Name                      0
Location                  0
Year                      0
Kilometers_Driven         0
Fuel_Type                 0
Transmission              0
Owner_Type                0
Mileage                   0
Engine                   46
Power                   175
Seats                     0
New_Price              6246
Price                  1234
kilometers_driven_log     0
price_log              1234
Brand                     0
Model                     0
dtype: int64
```

Looks like **Mileage** also no longer has missing values. We can continue working on the rest of the columns with missing values

Missing values for Engine

In [ ]:
```
# Impute missing Engine values with the median value.
data["Engine"].fillna(data['Engine'].median(), inplace = True)
```

Missing values for Power

In [ ]:
```
# Impute missing Power value with median value as well
data["Power"].fillna(data['Power'].median(), inplace = True)
```

Missing values for New_price

In [ ]:
```
# Impute missing New_price with the Median price as well, as the mean can be
data["New_Price"].fillna(data['New_Price'].median(), inplace = True)
```

In [ ]:
```
data.isnull().sum()
```

Out[ ]:
```
Name                     0
Location                 0
Year                     0
Kilometers_Driven        0
Fuel_Type                0
Transmission             0
Owner_Type               0
Mileage                  0
Engine                   0
Power                    0
Seats                    0
New_Price                0
Price                 1234
kilometers_driven_log    0
price_log             1234
Brand                    0
Model                    0
dtype: int64
```

We have got the price and price log column still has the missing value, however, we are responsible to predict the price, so it would not be a good idea to fill the missing values with any statistical number, instead for now, we will drop the rows with the missing values

In [ ]:
```
# Drop the rows where 'Price' == NaN
cars_data = data[data["Price"].notna()]
```

Let's now perform the final validation of the missing rows in all the columns

In [ ]:
```
cars_data.isnull().sum()
```

Out[ ]:
```
Name                    0
Location                0
Year                    0
Kilometers_Driven       0
Fuel_Type               0
Transmission            0
Owner_Type              0
Mileage                 0
Engine                  0
Power                   0
Seats                   0
New_Price               0
Price                   0
kilometers_driven_log   0
price_log               0
Brand                   0
Model                   0
dtype: int64
```

Observation

- All missing values have been treated.

Important Insights from EDA and Data Preprocessing

- Kilometers_Driven and Price is highly right-skewed. **Log transformation** can be used to **reduce/remove the skewness** and helps to **normalize the distribution**
- Kilometers Driven has a peculiar relationship with the Year variable. Generally, the newer the car lesser the distance it has traveled, but this is not always true
- From box-plots we can see the outliers
- The distribution of mileage looks fairly normally distributed if we ignore the cars with 0 mileage
- About **99%** of the cars run on Diesel and Petrol while the rest 1% cars run on CNG, LPG and electric
- About **38%** of the cars are in the data are for the year 2014 - 2016
- **71.7%** of the cars have a **manual transmission**
- **Automatic transmission** cars are very costly as compared to cars with manual transmission
- Price of used cars has a large IQR in Coimbatore and Bangalore
- Price has a **positive relationship with Year**. Newer the car, the higher the price
- Power and engine are important predictors of price
- New_price is also a significant predictor of price
- **New Price** and Used Car Price are also **positively correlated**, which is expected
- 2 seater cars are all luxury variants. Cars with 8-10 seats are exclusively mid to high range
- Mileage does not seem to show much relationship with the price of used cars

- Mileage and power of newer cars is increasing owing to advancement in technology
- **Mileage** has a **negative correlation** with engine displacement and power. More powerful the engine, the more fuel it consumes in general
- Most cars have Power of engines between 90-100 bhp
- **Engine displacement and Power** of the car have a **positive relationship** with the price
- **82%** of the cars have first owners followed by **15.9%** of the cars with second owners
- Cars with fewer owners have higher prices, outliers in third owner cars these might be the luxury cars

Building Various Models

1. What we want to predict is "Price". We will use the normalized version 'price_log' for modeling.
2. Before we proceed to the model, we'll have to **encode categorical features**. We will drop categorical features like Name.
3. We'll **split the data into train and test**, to be able to evaluate the model that we build on the train data.
4. Build **Regression** models using train data.
5. **Evaluate** the model performance.

Split the Data

- Step1: Separating the independent variables (X) and the dependent variable (y)
- Step2: Encode the categorical variables in X using pd.dummies
- Step3: Split the data into train and test using train_test_split

For the final validation and understanding of the data types, to ensure that the correct columns are used for encoding, let's check the information about the data using `.info`.

In [ ]:
```
cars_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 6018 entries, 0 to 6018
Data columns (total 17 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Name                   6018 non-null   object 
 1   Location               6018 non-null   object 
 2   Year                   6018 non-null   int64  
 3   Kilometers_Driven      6018 non-null   int64  
 4   Fuel_Type              6018 non-null   object 
 5   Transmission           6018 non-null   object 
 6   Owner_Type             6018 non-null   object 
 7   Mileage                6018 non-null   float64
 8   Engine                 6018 non-null   float64
 9   Power                  6018 non-null   float64
 10  Seats                  6018 non-null   float64
 11  New_Price              6018 non-null   float64
 12  Price                  6018 non-null   float64
 13  kilometers_driven_log  6018 non-null   float64
 14  price_log              6018 non-null   float64
 15  Brand                  6018 non-null   object 
 16  Model                  6018 non-null   object 
dtypes: float64(8), int64(2), object(7)
memory usage: 846.3+ KB
```

Time to split the training variables (X), and target feature (y)

In [ ]:

```python
X = cars_data.drop(['Name','Price','price_log','Kilometers_Driven'],axis = 1
"""
- Dropping the name (not necessary for the ML model as it is an identity fea
- Dropping the Price and Price Log columns. Price Log is the target variable
- We have also created the "kilometers_driven_log" column from the "Kilomete
"""
X = pd.get_dummies(X, columns = X.select_dtypes(include = ["object", "catego
"""
Further, we are creating dummy variables for the categorical variables in th
Additionally, we will remove the first column of dummies for each categorica
"""
y = cars_data[["price_log", "Price"]] # Target variable
```

In [ ]:

```python
X.sample(4)
```

Out[ ]:

	Year	Mileage	Engine	Power	Seats	New_Price	kilometers_driven_log	Loca
968	2008	17.50	1298.00	85.80	5.00	11.57	10.76	
2876	2014	18.53	1968.00	187.74	5.00	59.38	10.93	
5583	2014	14.70	1984.00	181.00	5.00	11.57	11.44	
2753	2013	18.50	1197.00	82.85	5.00	11.57	11.43	

We have already performed the X and y (training features and target variable) split above, now it is good time to split the data into training and test set using **scikit-learn**

framework's Train Test Split function

In [ ]:
```
# Splitting data into training and test set:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, r
"""
random_state=1 (or any number) is a seed value for the random number generat
producing the same results every time you run the code. This reproducibility
set the random_state, you will get different results each time you run the c
"""

print(X_train.shape, X_test.shape)
```

```
(4212, 264) (1806, 264)
```

Just to validate:

In [ ]:
```
print(f"{round(X_train.shape[0] / X.shape[0] * 100, 0)}% of the data is in t
```

```
70.0% of the data is in training set, and 30.0% of the data is in test set
```

In [ ]:
```
# Let us write a function for calculating r2_score and RMSE on train and tes
# This function takes model as an input on which we have trained particular

def get_model_score(model, flag = True):

    '''
    model : regressor to predict values of X
    '''
    # Defining an empty list to store train and test results
    score_list = []
    pred_train = model.predict(X_train) # Predict the y values of the traini
    pred_train_ = np.exp(pred_train) # Predict exponentiated value of price
    pred_test = model.predict(X_test) # Predict price for the unseen data
    pred_test_ = np.exp(pred_test) # Predict exponentiated value of price fo
    train_r2 = metrics.r2_score(y_train['Price'], pred_train_) # Getting tra
    test_r2 = metrics.r2_score(y_test['Price'], pred_test_) # Getting test R
    """
    What is R^2 (R-squared)?
    R-squared is a statistical measure (Goodness of the fit of the Model) th
    that's explained by an independent variable or variables in a regressior
    It is calculated as the ratio of the explained variance to the total var

                                    R^2 = Explained Variance / Total Varianc

    - Explained variance is the variance of the dependent variable that is p
    sum of squared differences between the actual values and the predicted v
    - Total Variance is the variance of the dependent variable due to its me
    - Value of R^2 ranges from 0 to 1, where 0 represents that the model exp
    represents that the model explains all the variability of the response c
    """
    train_rmse = np.sqrt(metrics.mean_squared_error(y_train['Price'], pred_t
    test_rmse = np.sqrt(metrics.mean_squared_error(y_test['Price'], pred_tes

    #Adding all scores in the list
    score_list.extend((train_r2, test_r2, train_rmse, test_rmse))
```

```
    # If the flag is set to True then only the following print statements wi
    if flag == True:
        print("R-square on training set : ", metrics.r2_score(y_train['Price
        print("R-square on test set : ", metrics.r2_score(y_test['Price'], p
        print("RMSE on training set : ", np.sqrt(metrics.mean_squared_error(
        print("RMSE on test set : ", np.sqrt(metrics.mean_squared_error(y_te

    # Returning the list with train and test scores
    return score_list
```

Fitting a linear model

Linear Regression can be implemented using:

1) Sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
2) Statsmodels: https://www.statsmodels.org/stable/regression.html

In [ ]:
```
# Initiating th linear regression model
lr = LinearRegression()
```

Fitting the data to train on the Linear Regression Model

In [ ]:
```
lr.fit(X_train,y_train['price_log'])
```

Out[ ]:
```
▾ LinearRegression
LinearRegression()
```

Getting the training and test scores that we defined above for the Linear Regression Model

In [ ]:
```
LR_score = get_model_score(lr)
```

```
R-square on training set :  0.9400205799079627
R-square on test set :  0.8684533474969429
RMSE on training set :  2.736228735246362
RMSE on test set :  4.042214333879621
```

Observation

- Linear regression has performed well on training and test set with no overfitting

Important variables of Linear Regression

In [ ]:
```
X_train1 = X_train.astype(float) # Converting all the training features into
y_train1 = y_train.astype(float) # Converting target variable as float type
```

Quick Notes:
The constant (also known as the intercept or bias term) is the value at which the

regression line crosses the y-axis. It represents the predicted value of the dependent variable (y) when all the independent variables (x) are equal to zero.

Mathematically, in a simple linear regression model, the equation is:

$$y = b0 + b1x, \text{ where}$$

- y is the dependent (target that we want to predict) variable,
- x is the independent variable,
- b0 is the intercept (constant), and,
- b1 is the slope.

For example, if you are modeling the relationship between temperature (x) and energy consumption (y), the constant might represent the base level of energy consumption when the temperature is zero.

Ordinary Least Squares (OLS) model is a method used to estimate the unknown parameters (coefficients) in a linear regression model. Its goal is to find the best-fitting line (or hyperplane in multiple dimensions) that minimizes the sum of the squared differences between the observed values (actual data points) and the predicted values (values on the regression line).

In [ ]:

```
# Import Statsmodels
import statsmodels.api as sm

# Statsmodel api does not add a constant by default. We need to add it expli
x_train = sm.add_constant(X_train1)

# Add constant to test data
x_test = sm.add_constant(X_test)

def build_ols_model(train):

    # Create the model
    olsmodel = sm.OLS(y_train1["price_log"], train)

    return olsmodel.fit()


# Fit linear model on new dataset
olsmodel1 = build_ols_model(x_train)

print(olsmodel1.summary())
```

```
                            OLS Regression Results
==============================================================================
==
Dep. Variable:              price_log   R-squared:                       0.9
59
Model:                            OLS   Adj. R-squared:                  0.9
56
Method:                 Least Squares   F-statistic:                     40
7.1
Date:                Sat, 01 Feb 2025   Prob (F-statistic):               0.
00
Time:                        13:44:15   Log-Likelihood:                 132
4.0
No. Observations:                4212   AIC:                            -219
0.
Df Residuals:                    3983   BIC:                            -73
6.9
Df Model:                         228
Covariance Type:            nonrobust
==============================================================================
=================
                               coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
-----------------
const                     -203.7569      2.880    -70.740      0.000    -20
9.404    -198.110
Year                         0.1069      0.001     72.173      0.000
0.104       0.110
Mileage                      0.0028      0.002      1.576      0.115      -
0.001       0.006
Engine                   -8.016e-05   2.21e-05     -3.633      0.000      -
0.000   -3.69e-05
Power                        0.0025      0.000     10.505      0.000
0.002       0.003
Seats                       -0.0116      0.014     -0.850      0.395      -
0.038       0.015
New_Price                   -0.0009      0.000     -2.360      0.018      -
0.002      -0.000
kilometers_driven_log       -0.0741      0.006    -13.178      0.000      -
0.085      -0.063
Location_Bangalore           0.1847      0.019      9.843      0.000
0.148       0.221
Location_Chennai             0.0590      0.018      3.292      0.001
0.024       0.094
Location_Coimbatore          0.1530      0.017      8.955      0.000
0.119       0.186
Location_Delhi              -0.0818      0.017     -4.729      0.000      -
0.116      -0.048
Location_Hyderabad           0.1474      0.017      8.860      0.000
0.115       0.180
Location_Jaipur             -0.0191      0.018     -1.043      0.297      -
0.055       0.017
Location_Kochi              -0.0171      0.017     -1.002      0.317      -
0.051       0.016
Location_Kolkata            -0.2205      0.018    -12.596      0.000      -
```

```
0.255      -0.186
Location_Mumbai                 -0.0507       0.017     -3.055       0.002      -
0.083      -0.018
Location_Pune                   -0.0240       0.017     -1.404       0.160      -
0.058       0.010
Fuel_Type_Diesel                 0.1018       0.040      2.549       0.011
0.024       0.180
Fuel_Type_Electric              -0.2751       0.088     -3.121       0.002      -
0.448      -0.102
Fuel_Type_LPG                   -0.0115       0.079     -0.146       0.884      -
0.166       0.143
Fuel_Type_Petrol                -0.0010       0.045     -0.023       0.982      -
0.089       0.087
Transmission_Manual             -0.1199       0.010    -11.704       0.000      -
0.140      -0.100
Owner_Type_Fourth & Above       -0.1134       0.087     -1.307       0.191      -
0.284       0.057
Owner_Type_Second               -0.0547       0.008     -6.504       0.000      -
0.071      -0.038
Owner_Type_Third                -0.1379       0.022     -6.245       0.000      -
0.181      -0.095
Brand_audi                      -6.9460       0.113    -61.676       0.000      -
7.167      -6.725
Brand_bentley                   -3.3616       0.111    -30.211       0.000      -
3.580      -3.143
Brand_bmw                       -8.2504       0.158    -52.324       0.000      -
8.560      -7.941
Brand_chevrolet                 -8.3580       0.114    -73.302       0.000      -
8.582      -8.134
Brand_datsun                    -7.4256       0.109    -68.438       0.000      -
7.638      -7.213
Brand_fiat                      -8.1192       0.111    -72.909       0.000      -
8.338      -7.901
Brand_force                     -4.2063       0.091    -46.337       0.000      -
4.384      -4.028
Brand_ford                      -7.9289       0.113    -70.282       0.000      -
8.150      -7.708
Brand_honda                     -8.2210       0.115    -71.465       0.000      -
8.447      -7.995
Brand_hyundai                   -9.2446       0.169    -54.813       0.000      -
9.575      -8.914
Brand_isuzu                     -5.6753       0.114    -49.647       0.000      -
5.899      -5.451
Brand_jaguar                    -5.5612       0.105    -52.819       0.000      -
5.768      -5.355
Brand_jeep                      -4.1957       0.069    -61.138       0.000      -
4.330      -4.061
Brand_lamborghini               -3.2145       0.114    -28.313       0.000      -
3.437      -2.992
Brand_land                      -3.7579       0.064    -58.735       0.000      -
3.883      -3.632
Brand_mahindra                  -8.4209       0.122    -69.149       0.000      -
8.660      -8.182
Brand_maruti                    -8.5604       0.134    -63.772       0.000      -
8.824      -8.297
Brand_mercedes-benz             -7.3221       0.131    -55.774       0.000      -
```

```
7.580      -7.065
Brand_mini                         -5.8896      0.112    -52.664      0.000      -
6.109      -5.670
Brand_mitsubishi                   -6.6640      0.114    -58.603      0.000      -
6.887      -6.441
Brand_nissan                       -7.6820      0.114    -67.108      0.000      -
7.906      -7.458
Brand_porsche                      -5.4148      0.106    -51.093      0.000      -
5.623      -5.207
Brand_renault                      -8.0890      0.114    -70.968      0.000      -
8.312      -7.866
Brand_skoda                        -7.5794      0.105    -72.060      0.000      -
7.786      -7.373
Brand_smart                        -4.5386      0.108    -42.090      0.000      -
4.750      -4.327
Brand_tata                         -8.4158      0.148    -57.030      0.000      -
8.705      -8.126
Brand_toyota                       -7.2235      0.113    -64.168      0.000      -
7.444      -7.003
Brand_volkswagen                   -7.7478      0.112    -68.946      0.000      -
7.968      -7.527
Brand_volvo                        -6.7516      0.114    -59.367      0.000      -
6.975      -6.529
Model_1000                      -6.286e-14   9.94e-16    -63.218      0.000   -6.4
8e-14   -6.09e-14
Model_3                             0.1268      0.108      1.176      0.240      -
0.085       0.338
Model_5                             0.4196      0.109      3.859      0.000
0.206       0.633
Model_6                             0.8222      0.154      5.334      0.000
0.520       1.124
Model_7                             0.8479      0.124      6.821      0.000
0.604       1.092
Model_800                          -1.4002      0.092    -15.160      0.000      -
1.581      -1.219
Model_a                            -0.8428      0.107     -7.855      0.000      -
1.053      -0.632
Model_a-star                       -0.8211      0.086     -9.577      0.000      -
0.989      -0.653
Model_a3                           -1.2113      0.089    -13.659      0.000      -
1.385      -1.037
Model_a4                           -1.1042      0.040    -27.712      0.000      -
1.182      -1.026
Model_a6                           -0.9700      0.042    -23.009      0.000      -
1.053      -0.887
Model_a7                        -7.755e-15   6.64e-16    -11.686      0.000   -9.0
6e-15   -6.45e-15
Model_a8                           -0.0414      0.166     -0.249      0.803      -
0.367       0.284
Model_accent                       -0.2696      0.135     -2.000      0.046      -
0.534      -0.005
Model_accord                       -0.5315      0.049    -10.790      0.000      -
0.628      -0.435
Model_alto                         -1.0984      0.064    -17.121      0.000      -
1.224      -0.973
Model_amaze                        -1.0099      0.033    -30.996      0.000      -
```

```
1.074      -0.946
Model_ameo                       -1.4900      0.056    -26.645      0.000       -
1.600      -1.380
Model_aspire                     -1.1822      0.090    -13.083      0.000       -
1.359      -1.005
Model_aveo                       -1.1864      0.061    -19.530      0.000       -
1.306      -1.067
Model_avventura                  -1.0143      0.102     -9.898      0.000       -
1.215      -0.813
Model_b                          -0.9160      0.102     -8.943      0.000       -
1.117      -0.715
Model_baleno                     -0.5964      0.067     -8.919      0.000       -
0.728      -0.465
Model_beat                       -1.2335      0.044    -27.895      0.000       -
1.320      -1.147
Model_beetle                  -3.645e-15    6.1e-16     -5.973      0.000    -4.8
4e-15   -2.45e-15
Model_bolero                     -0.3568      0.065     -5.465      0.000       -
0.485      -0.229
Model_bolt                       -1.1036      0.142     -7.787      0.000       -
1.381      -0.826
Model_boxster                  -8.63e-15   1.22e-15     -7.083      0.000     -1.
1e-14   -6.24e-15
Model_br-v                       -0.7541      0.122     -6.174      0.000       -
0.994      -0.515
Model_brio                       -1.1135      0.038    -29.354      0.000       -
1.188      -1.039
Model_brv                        -0.6625      0.081     -8.147      0.000       -
0.822      -0.503
Model_c-class                    -0.7098      0.145     -4.890      0.000       -
0.994      -0.425
Model_camry                      -0.9738      0.078    -12.451      0.000       -
1.127      -0.820
Model_captiva                    -0.5668      0.167     -3.400      0.001       -
0.894      -0.240
Model_captur                     -0.7671      0.120     -6.412      0.000       -
1.002      -0.533
Model_cayenne                    -2.6193      0.082    -31.833      0.000       -
2.781      -2.458
Model_cayman                     -1.2249      0.149     -8.240      0.000       -
1.516      -0.933
Model_cedia                   -7.756e-16   3.58e-16     -2.164      0.031    -1.4
8e-15   -7.28e-17
Model_celerio                    -0.9031      0.067    -13.408      0.000       -
1.035      -0.771
Model_ciaz                       -0.3886      0.066     -5.846      0.000       -
0.519      -0.258
Model_city                       -0.6964      0.027    -25.656      0.000       -
0.750      -0.643
Model_civic                      -0.7741      0.041    -19.098      0.000       -
0.854      -0.695
Model_cla                        -0.6814      0.087     -7.817      0.000       -
0.852      -0.510
Model_classic                    -8.9910      0.212    -42.365      0.000       -
9.407      -8.575
Model_cls-class                  -0.0957      0.192     -0.499      0.618       -
```

```
0.472       0.280
Model_clubman                -1.7794      0.148    -12.022      0.000      -
2.070      -1.489
Model_compass                -4.1957      0.069    -61.138      0.000      -
4.330      -4.061
Model_continental            -3.3616      0.111    -30.211      0.000      -
3.580      -3.143
Model_cooper                 -1.8994      0.079    -23.909      0.000      -
2.055      -1.744
Model_corolla                -1.4734      0.041    -35.985      0.000      -
1.554      -1.393
Model_countryman             -2.2108      0.149    -14.880      0.000      -
2.502      -1.919
Model_cr-v                   -0.1668      0.044     -3.769      0.000      -
0.254      -0.080
Model_creta                   0.5992      0.126      4.757      0.000
0.352       0.846
Model_crosspolo              -1.4605      0.117    -12.501      0.000      -
1.690      -1.231
Model_cruze                  -0.7111      0.063    -11.342      0.000      -
0.834      -0.588
Model_d-max                  -3.1360      0.114    -27.503      0.000      -
3.360      -2.912
Model_duster                 -0.7943      0.043    -18.512      0.000      -
0.878      -0.710
Model_dzire                  -0.5092      0.082     -6.195      0.000      -
0.670      -0.348
Model_e                   -9.272e-16   1.85e-16     -5.018      0.000   -1.2
9e-15   -5.65e-16
Model_e-class                -0.5412      0.072     -7.511      0.000      -
0.682      -0.400
Model_ecosport               -1.0009      0.043    -23.276      0.000      -
1.085      -0.917
Model_eeco                   -0.9829      0.084    -11.645      0.000      -
1.148      -0.817
Model_elantra                 0.6105      0.132      4.628      0.000
0.352       0.869
Model_elite                   0.2008      0.133      1.512      0.130      -
0.059       0.461
Model_endeavour              -0.2731      0.060     -4.520      0.000      -
0.392      -0.155
Model_enjoy                  -0.9265      0.073    -12.652      0.000      -
1.070      -0.783
Model_eon                    -0.3966      0.124     -3.187      0.001      -
0.641      -0.153
Model_ertiga                 -0.2687      0.073     -3.662      0.000      -
0.413      -0.125
Model_esteem                 -1.1222      0.101    -11.065      0.000      -
1.321      -0.923
Model_estilo                 -0.8300      0.121     -6.866      0.000      -
1.067      -0.593
Model_etios                  -1.9448      0.044    -43.888      0.000      -
2.032      -1.858
Model_evalia                 -1.6071      0.162     -9.890      0.000      -
1.926      -1.289
Model_f                      -1.6858      0.147    -11.431      0.000      -
```

```
1.975      -1.397
Model_fabia                    -1.7960      0.060    -29.988      0.000      -
1.913      -1.679
Model_fiesta                   -1.3259      0.049    -26.932      0.000      -
1.422      -1.229
Model_figo                     -1.4441      0.043    -33.576      0.000      -
1.528      -1.360
Model_fluence                  -1.0436      0.099    -10.494      0.000      -
1.239      -0.849
Model_fortuner                 -0.6756      0.042    -16.117      0.000      -
0.758      -0.593
Model_fortwo                   -4.5386      0.108    -42.090      0.000      -
4.750      -4.327
Model_freestyle                -0.4201      0.167     -2.510      0.012      -
0.748      -0.092
Model_fusion                   -1.0750      0.167     -6.438      0.000      -
1.402      -0.748
Model_gallardo                 -3.2145      0.114    -28.313      0.000      -
3.437      -2.992
Model_getz                     -0.2705      0.136     -1.982      0.048      -
0.538      -0.003
Model_gl-class                  0.0800      0.095      0.838      0.402      -
0.107       0.267
Model_gla                      -0.5909      0.090     -6.552      0.000      -
0.768      -0.414
Model_glc                      -0.2534      0.097     -2.623      0.009      -
0.443      -0.064
Model_gle                      -0.0527      0.092     -0.572      0.568      -
0.233       0.128
Model_gls                       0.0998      0.197      0.506      0.613      -
0.287       0.486
Model_go                       -2.2704      0.100    -22.700      0.000      -
2.466      -2.074
Model_grand                    -0.0733      0.121     -0.605      0.545      -
0.311       0.164
Model_grande                   -1.3156      0.102    -12.871      0.000      -
1.516      -1.115
Model_hexa                     -0.1298      0.200     -0.648      0.517      -
0.523       0.263
Model_i10                      -0.0651      0.123     -0.530      0.596      -
0.306       0.176
Model_i20                       0.1319      0.123      1.073      0.283      -
0.109       0.373
Model_ignis                    -0.9282      0.121     -7.703      0.000      -
1.164      -0.692
Model_ikon                     -1.5503      0.064    -24.376      0.000      -
1.675      -1.426
Model_indica                   -1.4454      0.101    -14.286      0.000      -
1.644      -1.247
Model_indigo                   -1.3450      0.104    -12.983      0.000      -
1.548      -1.142
Model_innova                   -1.0007      0.041    -24.171      0.000      -
1.082      -0.920
Model_jazz                     -0.9184      0.037    -25.128      0.000      -
0.990      -0.847
Model_jeep                     -0.2080      0.123     -1.685      0.092      -
```

```
0.450       0.034
Model_jetta                    -0.8697      0.051    -17.011      0.000      -
0.970      -0.769
Model_koleos                   -0.4737      0.101     -4.694      0.000      -
0.672      -0.276
Model_kuv                      -1.0151      0.082    -12.374      0.000      -
1.176      -0.854
Model_kwid                     -1.6755      0.049    -34.148      0.000      -
1.772      -1.579
Model_lancer                   -1.9794      0.119    -16.633      0.000      -
2.213      -1.746
Model_laura                    -1.2885      0.040    -32.133      0.000      -
1.367      -1.210
Model_linea                    -1.1998      0.074    -16.123      0.000      -
1.346      -1.054
Model_lodgy                    -0.8561      0.169     -5.079      0.000      -
1.187      -0.526
Model_logan                    -1.1648      0.125     -9.316      0.000      -
1.410      -0.920
Model_m-class                  -0.2923      0.084     -3.491      0.000      -
0.457      -0.128
Model_manza                    -1.1518      0.110    -10.471      0.000      -
1.367      -0.936
Model_micra                    -1.6458      0.057    -28.966      0.000      -
1.757      -1.534
Model_mobilio                  -0.7984      0.062    -12.942      0.000      -
0.919      -0.677
Model_montero                  -1.3565      0.156     -8.670      0.000      -
1.663      -1.050
Model_mustang                   0.3427      0.177      1.938      0.053      -
0.004       0.689
Model_mux                      -2.5393      0.137    -18.506      0.000      -
2.808      -2.270
Model_nano                     -1.8736      0.113    -16.604      0.000      -
2.095      -1.652
Model_new                      -0.6892      0.071     -9.766      0.000      -
0.828      -0.551
Model_nexon                    -0.5252      0.203     -2.582      0.010      -
0.924      -0.126
Model_nuvosport                -0.8232      0.124     -6.620      0.000      -
1.067      -0.579
Model_octavia                  -1.1188      0.040    -28.137      0.000      -
1.197      -1.041
Model_omni                     -1.2006      0.079    -15.201      0.000      -
1.355      -1.046
Model_one                      -4.2063      0.091    -46.337      0.000      -
4.384      -4.028
Model_optra                    -0.9435      0.058    -16.144      0.000      -
1.058      -0.829
Model_outlander                -1.8847      0.155    -12.153      0.000      -
2.189      -1.581
Model_pajero                   -1.4434      0.078    -18.497      0.000      -
1.596      -1.290
Model_panamera                 -1.5706      0.086    -18.290      0.000      -
1.739      -1.402
Model_passat                   -0.9531      0.079    -12.121      0.000      -
```

```
1.107      -0.799
Model_petra                   -1.5864      0.161     -9.849      0.000      -
1.902      -1.271
Model_platinum             -3.294e-16   9.27e-17     -3.551      0.000   -5.1
1e-16   -1.48e-16
Model_polo                    -1.4366      0.039    -37.283      0.000      -
1.512      -1.361
Model_prius                   -0.2751      0.088     -3.121      0.002      -
0.448      -0.102
Model_pulse                   -1.2749      0.089    -14.388      0.000      -
1.449      -1.101
Model_punto                   -1.4392      0.102    -14.113      0.000      -
1.639      -1.239
Model_q3                      -1.0736      0.052    -20.811      0.000      -
1.175      -0.972
Model_q5                      -0.7673      0.050    -15.363      0.000      -
0.865      -0.669
Model_q7                      -0.5464      0.055     -9.866      0.000      -
0.655      -0.438
Model_qualis                  -0.8800      0.126     -6.998      0.000      -
1.126      -0.633
Model_quanto                  -0.9017      0.090    -10.007      0.000      -
1.078      -0.725
Model_r-class                 -0.3612      0.127     -2.848      0.004      -
0.610      -0.113
Model_rapid                   -1.4688      0.038    -38.568      0.000      -
1.543      -1.394
Model_redi                    -2.6573      0.144    -18.513      0.000      -
2.939      -2.376
Model_redi-go                 -2.4979      0.086    -28.908      0.000      -
2.667      -2.328
Model_renault                 -0.7564      0.173     -4.368      0.000      -
1.096      -0.417
Model_ritz                    -0.7484      0.067    -11.194      0.000      -
0.880      -0.617
Model_rover                   -3.7579      0.064    -58.735      0.000      -
3.883      -3.632
Model_rs5                     -0.6908      0.131     -5.275      0.000      -
0.948      -0.434
Model_s                       -0.3109      0.064     -4.886      0.000      -
0.436      -0.186
Model_s-class                 -0.2236      0.112     -1.993      0.046      -
0.444      -0.004
Model_s-cross                 -0.4615      0.190     -2.433      0.015      -
0.833      -0.090
Model_s60                     -1.3706      0.081    -16.847      0.000      -
1.530      -1.211
Model_s80                     -1.8761      0.158    -11.889      0.000      -
2.185      -1.567
Model_safari                  -0.4697      0.120     -3.922      0.000      -
0.704      -0.235
Model_sail                    -1.0299      0.070    -14.681      0.000      -
1.167      -0.892
Model_santa                    0.8162      0.142      5.753      0.000
0.538       1.094
Model_santro                  -0.2217      0.125     -1.780      0.075      -
```

```
0.466       0.023
Model_scala                          -1.2037      0.088    -13.665      0.000      -
1.376      -1.031
Model_scorpio                        -0.1998      0.043     -4.624      0.000      -
0.285      -0.115
Model_siena                          -1.5638      0.162     -9.668      0.000      -
1.881      -1.247
Model_sl-class                        0.1628      0.198      0.822      0.411      -
0.226       0.551
Model_slc                            -0.2746      0.153     -1.795      0.073      -
0.574       0.025
Model_slk-class                      -0.0812      0.130     -0.623      0.533      -
0.337       0.174
Model_sonata                          0.6779      0.150      4.511      0.000
0.383       0.972
Model_spark                          -1.3252      0.074    -17.851      0.000      -
1.471      -1.180
Model_ssangyong                       0.0009      0.064      0.015      0.988      -
0.124       0.126
Model_sumo                           -0.5720      0.122     -4.703      0.000      -
0.810      -0.334
Model_sunny                          -1.4551      0.057    -25.483      0.000      -
1.567      -1.343
Model_superb                         -0.9445      0.036    -26.293      0.000      -
1.015      -0.874
Model_swift                          -0.5674      0.062     -9.126      0.000      -
0.689      -0.446
Model_sx4                            -0.5328      0.070     -7.613      0.000      -
0.670      -0.396
Model_tavera                         -0.4352      0.133     -3.263      0.001      -
0.697      -0.174
Model_teana                          -1.0459      0.162     -6.466      0.000      -
1.363      -0.729
Model_terrano                        -1.2276      0.057    -21.449      0.000      -
1.340      -1.115
Model_thar                           -0.4466      0.083     -5.413      0.000      -
0.608      -0.285
Model_tiago                          -1.1887      0.117    -10.136      0.000      -
1.419      -0.959
Model_tigor                          -1.0202      0.133     -7.682      0.000      -
1.281      -0.760
Model_tiguan                         -0.2493      0.162     -1.538      0.124      -
0.567       0.068
Model_tt                             -0.5411      0.122     -4.436      0.000      -
0.780      -0.302
Model_tucson                          0.7533      0.165      4.564      0.000
0.430       1.077
Model_tuv                            -0.6589      0.071     -9.274      0.000      -
0.798      -0.520
Model_v40                            -1.2901      0.102    -12.663      0.000      -
1.490      -1.090
Model_vento                          -1.2886      0.039    -33.270      0.000      -
1.365      -1.213
Model_venture                        -0.9655      0.202     -4.776      0.000      -
1.362      -0.569
Model_verito                         -0.9307      0.104     -8.976      0.000      -
```

```
1.134      -0.727
Model_verna                      0.2624      0.125      2.108      0.035       
0.018       0.507
Model_versa                     -0.4904      0.195     -2.516      0.012      -
0.873      -0.108
Model_vitara                    -0.3688      0.070     -5.254      0.000      -
0.506      -0.231
Model_wagon                     -0.8432      0.064    -13.200      0.000      -
0.968      -0.718
Model_wr-v                            0          0        nan        nan       
0           0
Model_wrv                       -0.7953      0.168     -4.728      0.000      -
1.125      -0.465
Model_x-trail                   -0.7005      0.118     -5.919      0.000      -
0.933      -0.468
Model_x1                         0.1122      0.112      0.999      0.318      -
0.108       0.332
Model_x3                         0.4590      0.119      3.849      0.000       
0.225       0.693
Model_x5                         0.7316      0.117      6.274      0.000       
0.503       0.960
Model_x6                         0.9439      0.136      6.959      0.000       
0.678       1.210
Model_xc60                      -1.3063      0.086    -15.192      0.000      -
1.475      -1.138
Model_xc90                      -0.9086      0.161     -5.655      0.000      -
1.224      -0.594
Model_xcent                     -0.0574      0.126     -0.455      0.649      -
0.304       0.190
Model_xe                              0          0        nan        nan       
0           0
Model_xenon                     -0.8971      0.139     -6.462      0.000      -
1.169      -0.625
Model_xf                        -2.2138      0.068    -32.428      0.000      -
2.348      -2.080
Model_xj                        -1.6616      0.083    -19.910      0.000      -
1.825      -1.498
Model_xuv300                    -0.1882      0.173     -1.085      0.278      -
0.528       0.152
Model_xuv500                    -0.0999      0.036     -2.778      0.005      -
0.170      -0.029
Model_xylo                      -0.6716      0.059    -11.473      0.000      -
0.786      -0.557
Model_yeti                      -0.9629      0.068    -14.134      0.000      -
1.096      -0.829
Model_z4                         0.8319      0.216      3.847      0.000       
0.408       1.256
Model_zen                       -0.9424      0.074    -12.820      0.000      -
1.087      -0.798
Model_zest                      -0.9745      0.107     -9.090      0.000      -
1.185      -0.764
============================================================================
==
Omnibus:                     1870.031   Durbin-Watson:                   2.0
18
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            99284.3
```

```
77
Skew:                          -1.339   Prob(JB):                         0.
00
Kurtosis:                      26.634   Cond. No.                      7.40e+
19
==============================================================================
==

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.
[2] The smallest eigenvalue is 5.27e-30. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

In [ ]:

```
# Calculate Odds Ratio and probability.
# Create a data frame to collate Odds ratio, probability, and p-value of the
olsmod = pd.DataFrame(olsmodel1.params, columns = ['coef'])
olsmod['pval'] = olsmodel1.pvalues
```

In [ ]:

```
# Filter by significant p-value (pval <0.05) and sort descending by Odds rat
olsmod = olsmod.sort_values(by = "pval", ascending = False)
pval_filter = olsmod['pval']<= 0.05
olsmod[pval_filter]
```

Out[ ]:

	coef	pval
Model_getz	-0.27	0.05
Model_s-class	-0.22	0.05
Model_accent	-0.27	0.05
Model_verna	0.26	0.04
Model_cedia	-0.00	0.03
New_Price	-0.00	0.02
Model_s-cross	-0.46	0.02
Model_freestyle	-0.42	0.01
Model_versa	-0.49	0.01
Fuel_Type_Diesel	0.10	0.01
Model_nexon	-0.53	0.01
Model_glc	-0.25	0.01
Model_xuv500	-0.10	0.01
Model_r-class	-0.36	0.00
Location_Mumbai	-0.05	0.00
Fuel_Type_Electric	-0.28	0.00
Model_prius	-0.28	0.00
Model_eon	-0.40	0.00
Model_tavera	-0.44	0.00
Location_Chennai	0.06	0.00
Model_captiva	-0.57	0.00
Model_m-class	-0.29	0.00
Model_platinum	-0.00	0.00
Engine	-0.00	0.00
Model_ertiga	-0.27	0.00
Model_cr-v	-0.17	0.00
Model_z4	0.83	0.00
Model_x3	0.46	0.00
Model_5	0.42	0.00
Model_safari	-0.47	0.00
Model_renault	-0.76	0.00
Model_tt	-0.54	0.00

	coef	pval
Model_sonata	0.68	0.00
Model_endeavour	-0.27	0.00
Model_tucson	0.75	0.00
Model_scorpio	-0.20	0.00
Model_elantra	0.61	0.00
Model_koleos	-0.47	0.00
Model_sumo	-0.57	0.00
Model_wrv	-0.80	0.00
Location_Delhi	-0.08	0.00
Model_creta	0.60	0.00
Model_venture	-0.97	0.00
Model_s	-0.31	0.00
Model_c-class	-0.71	0.00
Model_e	-0.00	0.00
Model_lodgy	-0.86	0.00
Model_vitara	-0.37	0.00
Model_rs5	-0.69	0.00
Model_6	0.82	0.00
Model_thar	-0.45	0.00
Model_bolero	-0.36	0.00
Model_xc90	-0.91	0.00
Model_santa	0.82	0.00
Model_ciaz	-0.39	0.00
Model_x-trail	-0.70	0.00
Model_beetle	-0.00	0.00
Model_br-v	-0.75	0.00
Model_dzire	-0.51	0.00
Owner_Type_Third	-0.14	0.00
Model_x5	0.73	0.00
Model_captur	-0.77	0.00
Model_fusion	-1.07	0.00
Model_xenon	-0.90	0.00

	coef	pval
Model_teana	-1.05	0.00
Owner_Type_Second	-0.05	0.00
Model_gla	-0.59	0.00
Model_nuvosport	-0.82	0.00
Model_7	0.85	0.00
Model_estilo	-0.83	0.00
Model_x6	0.94	0.00
Model_qualis	-0.88	0.00
Model_boxster	-0.00	0.00
Model_e-class	-0.54	0.00
Model_sx4	-0.53	0.00
Model_tigor	-1.02	0.00
Model_ignis	-0.93	0.00
Model_bolt	-1.10	0.00
Model_cla	-0.68	0.00
Model_a	-0.84	0.00
Model_brv	-0.66	0.00
Model_cayman	-1.22	0.00
Model_montero	-1.36	0.00
Location_Hyderabad	0.15	0.00
Model_baleno	-0.60	0.00
Model_b	-0.92	0.00
Location_Coimbatore	0.15	0.00
Model_verito	-0.93	0.00
Model_zest	-0.97	0.00
Model_swift	-0.57	0.00
Model_tuv	-0.66	0.00
Model_logan	-1.16	0.00
Model_a-star	-0.82	0.00
Model_siena	-1.56	0.00
Model_new	-0.69	0.00
Location_Bangalore	0.18	0.00

	coef	pval
Model_petra	-1.59	0.00
Model_q7	-0.55	0.00
Model_evalia	-1.61	0.00
Model_avventura	-1.01	0.00
Model_quanto	-0.90	0.00
Model_tiago	-1.19	0.00
Model_manza	-1.15	0.00
Model_fluence	-1.04	0.00
Power	0.00	0.00
Model_accord	-0.53	0.00
Model_esteem	-1.12	0.00
Model_ritz	-0.75	0.00
Model_cruze	-0.71	0.00
Model_f	-1.69	0.00
Model_xylo	-0.67	0.00
Model_eeco	-0.98	0.00
Model_a7	-0.00	0.00
Transmission_Manual	-0.12	0.00
Model_s80	-1.88	0.00
Model_clubman	-1.78	0.00
Model_passat	-0.95	0.00
Model_outlander	-1.88	0.00
Model_kuv	-1.02	0.00
Model_camry	-0.97	0.00
Model_crosspolo	-1.46	0.00
Location_Kolkata	-0.22	0.00
Model_enjoy	-0.93	0.00
Model_v40	-1.29	0.00
Model_zen	-0.94	0.00
Model_grande	-1.32	0.00
Model_mobilio	-0.80	0.00
Model_indigo	-1.35	0.00

	coef	pval
Model_aspire	-1.18	0.00
kilometers_driven_log	-0.07	0.00
Model_wagon	-0.84	0.00
Model_celerio	-0.90	0.00
Model_a3	-1.21	0.00
Model_scala	-1.20	0.00
Model_punto	-1.44	0.00
Model_yeti	-0.96	0.00
Model_indica	-1.45	0.00
Model_pulse	-1.27	0.00
Model_sail	-1.03	0.00
Model_countryman	-2.21	0.00
Model_800	-1.40	0.00
Model_xc60	-1.31	0.00
Model_omni	-1.20	0.00
Model_q5	-0.77	0.00
Model_fortuner	-0.68	0.00
Model_linea	-1.20	0.00
Model_optra	-0.94	0.00
Model_nano	-1.87	0.00
Model_lancer	-1.98	0.00
Model_s60	-1.37	0.00
Model_jetta	-0.87	0.00
Model_alto	-1.10	0.00
Model_spark	-1.33	0.00
Model_panamera	-1.57	0.00
Model_pajero	-1.44	0.00
Model_mux	-2.54	0.00
Model_duster	-0.79	0.00
Model_redi	-2.66	0.00
Model_civic	-0.77	0.00
Model_aveo	-1.19	0.00

	coef	pval
Model_xj	-1.66	0.00
Model_q3	-1.07	0.00
Model_terrano	-1.23	0.00
Model_go	-2.27	0.00
Model_a6	-0.97	0.00
Model_ecosport	-1.00	0.00
Model_cooper	-1.90	0.00
Model_innova	-1.00	0.00
Model_ikon	-1.55	0.00
Model_jazz	-0.92	0.00
Model_sunny	-1.46	0.00
Model_city	-0.70	0.00
Model_superb	-0.94	0.00
Model_ameo	-1.49	0.00
Model_fiesta	-1.33	0.00
Model_d-max	-3.14	0.00
Model_a4	-1.10	0.00
Model_beat	-1.23	0.00
Model_octavia	-1.12	0.00
Model_gallardo	-3.21	0.00
Brand_lamborghini	-3.21	0.00
Model_redi-go	-2.50	0.00
Model_micra	-1.65	0.00
Model_brio	-1.11	0.00
Model_fabia	-1.80	0.00
Brand_bentley	-3.36	0.00
Model_continental	-3.36	0.00
Model_amaze	-1.01	0.00
Model_cayenne	-2.62	0.00
Model_laura	-1.29	0.00
Model_xf	-2.21	0.00
Model_vento	-1.29	0.00

	coef	pval
Model_figo	-1.44	0.00
Model_kwid	-1.68	0.00
Model_corolla	-1.47	0.00
Model_polo	-1.44	0.00
Model_rapid	-1.47	0.00
Model_fortwo	-4.54	0.00
Brand_smart	-4.54	0.00
Brand_nissan	-7.68	0.00
Brand_fiat	-8.12	0.00
Brand_chevrolet	-8.36	0.00
Brand_bmw	-8.25	0.00
Brand_tata	-8.42	0.00
Brand_toyota	-7.22	0.00
Brand_audi	-6.95	0.00
Brand_volkswagen	-7.75	0.00
Brand_volvo	-6.75	0.00
Model_1000	-0.00	0.00
Model_rover	-3.76	0.00
Model_one	-4.21	0.00
Model_classic	-8.99	0.00
Model_compass	-4.20	0.00
Model_etios	-1.94	0.00
Year	0.11	0.00
Brand_datsun	-7.43	0.00
Brand_force	-4.21	0.00
Brand_mitsubishi	-6.66	0.00
Brand_ford	-7.93	0.00
Brand_honda	-8.22	0.00
Brand_hyundai	-9.24	0.00
Brand_isuzu	-5.68	0.00
Brand_jaguar	-5.56	0.00
Brand_jeep	-4.20	0.00

	coef	pval
Brand_skoda	-7.58	0.00
Brand_renault	-8.09	0.00
Brand_porsche	-5.41	0.00
Brand_land	-3.76	0.00
Brand_mahindra	-8.42	0.00
Brand_maruti	-8.56	0.00
Brand_mercedes-benz	-7.32	0.00
Brand_mini	-5.89	0.00
const	-203.76	0.00

In [ ]:

```
# We are looking for overall significant variables.

pval_filter = olsmod['pval']<= 0.05
imp_vars = olsmod[pval_filter].index.tolist()

# We are going to retrieve the original variables (non-one-hot encoded varia

sig_var = []
for col in imp_vars:
    if '' in col:
        first_part = col.split('_')[0]
        for c in data.columns:
            if first_part in c and c not in sig_var :
                sig_var.append(c)


start = '\033[1m'
end = '\033[95m'
print(start+'Most overall significant categorical varaibles of LINEAR REGRES
```

```
Most overall significant categorical varaibles of LINEAR REGRESSION  are  :
 ['Model', 'New_Price', 'Fuel_Type', 'Location', 'Engine', 'Owner_Type', 'Po
wer', 'Transmission', 'kilometers_driven_log', 'Brand', 'Year']
```

Ridge Regression

Also Known as **L2 Regularization**, shrinks the coefficients evenly but does not necessarily bring them to zero. This means that less significant features will still have some influence on the final prediction. L2 regularization can help reduce model complexity but may be less robust to outliers.

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html

Initializing the Ridge Regression Model

In [ ]:
```
rdg = Ridge()
```

Fitting the training data into Ridge Regression Model

In [ ]:
```
rdg.fit(X_train,y_train['price_log'])
```

Out[ ]:
```
▾ Ridge
Ridge()
```

Looking at the performance scores from Ridge Regression

In [ ]:
```
Ridge_score = get_model_score(rdg)
```

```
R-square on training set :  0.9306479472062843
R-square on test set :  0.8951556045464901
RMSE on training set :  2.9422591930718034
RMSE on test set :  3.6087105329004547
```

Observation

- Ridge regression is able to generalize well compared to Linear Regression

Decision Tree

Initializing the Decision Tree Regressor Machine Learning model

In [ ]:
```
dtree = DecisionTreeRegressor(random_state = 1)
```

Fitting the data into a DecisionTree regressor model

In [ ]:
```
dtree.fit(X_train,y_train['price_log'])
```

Out[ ]:
```
▾ DecisionTreeRegressor
DecisionTreeRegressor(random_state=1)
```

Getting the Performance Score for Decision Tree Model

In [ ]:
```
Dtree_model = get_model_score(dtree)
```

```
R-square on training set :  0.9999965696959587
R-square on test set :  0.8146535721565282
RMSE on training set :  0.020692719736775493
RMSE on test set :  4.798124884032137
```

Observation

- Decision Tree is overfitting on the training set and hence not able to generalize well on the test set

In [ ]:

```
# Importance of features in the tree building ( The importance of a feature 
# (normalized) total reduction of the criterion brought by that feature. It 
print(pd.DataFrame(dtree.feature_importances_, columns = ["Imp"], index = X_
```

	Imp
Power	0.61
Year	0.23
Engine	0.05
kilometers_driven_log	0.01
Mileage	0.01
Brand_honda	0.00
Brand_tata	0.00
Transmission_Manual	0.00
New_Price	0.00
Location_Kolkata	0.00
Seats	0.00
Model_rover	0.00
Brand_mini	0.00
Brand_audi	0.00
Brand_mahindra	0.00
Location_Hyderabad	0.00
Brand_skoda	0.00
Brand_land	0.00
Location_Coimbatore	0.00
Model_5	0.00
Brand_hyundai	0.00
Location_Delhi	0.00
Fuel_Type_Petrol	0.00
Owner_Type_Second	0.00
Model_polo	0.00
Model_creta	0.00
Location_Bangalore	0.00
Owner_Type_Third	0.00
Brand_toyota	0.00
Location_Jaipur	0.00
Model_swift	0.00
Location_Mumbai	0.00
Location_Pune	0.00
Location_Kochi	0.00
Location_Chennai	0.00
Brand_chevrolet	0.00
Model_xylo	0.00
Brand_mercedes-benz	0.00
Model_ertiga	0.00
Model_freestyle	0.00
Model_city	0.00
Model_nano	0.00
Model_beat	0.00
Brand_bmw	0.00
Model_i20	0.00
Model_prius	0.00
Brand_volkswagen	0.00
Model_q5	0.00
Model_amaze	0.00
Model_figo	0.00
Model_i10	0.00
Model_3	0.00
Model_innova	0.00
Model_manza	0.00
Model_800	0.00

```
Model_elantra          0.00
Model_ikon             0.00
Model_compass          0.00
Model_xenon            0.00
Model_brio             0.00
Model_fabia            0.00
Brand_fiat             0.00
Model_grand            0.00
Model_x5               0.00
Model_jetta            0.00
Brand_ford             0.00
Model_indica           0.00
Model_sx4              0.00
Model_new              0.00
Brand_maruti           0.00
Model_ecosport         0.00
Model_celerio          0.00
Model_b                0.00
Fuel_Type_Diesel       0.00
Model_getz             0.00
Model_s                0.00
Model_scorpio          0.00
Model_superb           0.00
Model_a4               0.00
Model_m-class          0.00
Model_etios            0.00
Model_corolla          0.00
Model_santro           0.00
Model_wagon            0.00
Model_optra            0.00
Model_eeco             0.00
Model_esteem           0.00
Model_tigor            0.00
Model_elite            0.00
Model_indigo           0.00
Model_q7               0.00
Model_sail             0.00
Model_sunny            0.00
Model_zen              0.00
Brand_volvo            0.00
Model_fiesta           0.00
Model_xf               0.00
Model_s80              0.00
Model_ritz             0.00
Brand_renault          0.00
Model_x6               0.00
Model_octavia          0.00
Model_omni             0.00
Model_bolero           0.00
Model_xcent            0.00
Model_terrano          0.00
Model_baleno           0.00
Model_wrv              0.00
Model_fluence          0.00
Model_x1               0.00
Model_e-class          0.00
```

```
Model_vitara                0.00
Model_vento                 0.00
Model_pulse                 0.00
Model_ignis                 0.00
Brand_force                 0.00
Model_alto                  0.00
Model_grande                0.00
Model_a6                    0.00
Fuel_Type_LPG               0.00
Model_go                    0.00
Model_mobilio               0.00
Model_scala                 0.00
Model_sonata                0.00
Model_gl-class              0.00
Owner_Type_Fourth & Above 0.00
Model_a3                    0.00
Model_verna                 0.00
Model_kwid                  0.00
Brand_nissan                0.00
Model_ciaz                  0.00
Model_s-class               0.00
Model_logan                 0.00
Model_civic                 0.00
Model_bolt                  0.00
Model_duster                0.00
Model_yeti                  0.00
Model_r-class               0.00
Model_eon                   0.00
Model_jazz                  0.00
Model_q3                    0.00
Model_micra                 0.00
Brand_datsun                0.00
Model_linea                 0.00
Model_s-cross               0.00
Brand_jaguar                0.00
Model_ssangyong             0.00
Model_aveo                  0.00
Model_rapid                 0.00
Model_laura                 0.00
Model_brv                   0.00
Model_gla                   0.00
Model_aspire                0.00
Model_cayenne               0.00
Model_passat                0.00
Model_tuv                   0.00
Model_redi-go               0.00
Model_xuv300                0.00
Model_accent                0.00
Model_hexa                  0.00
Model_endeavour             0.00
Model_jeep                  0.00
Model_a-star                0.00
Brand_mitsubishi            0.00
Model_quanto                0.00
Model_zest                  0.00
Model_cooper                0.00
```

Model_redi	0.00
Model_d-max	0.00
Model_cla	0.00
Model_s60	0.00
Model_slc	0.00
Brand_porsche	0.00
Model_tiago	0.00
Model_gle	0.00
Model_beetle	0.00
Model_ameo	0.00
Model_v40	0.00
Model_avventura	0.00
Model_tucson	0.00
Model_tt	0.00
Model_tiguan	0.00
Model_thar	0.00
Model_verito	0.00
Model_boxster	0.00
Model_teana	0.00
Model_tavera	0.00
Model_br-v	0.00
Model_c-class	0.00
Model_camry	0.00
Model_captiva	0.00
Model_sumo	0.00
Model_venture	0.00
Model_accord	0.00
Model_spark	0.00
Model_versa	0.00
Fuel_Type_Electric	0.00
Model_z4	0.00
Brand_bentley	0.00
Brand_isuzu	0.00
Model_xuv500	0.00
Brand_jeep	0.00
Model_xj	0.00
Brand_lamborghini	0.00
Model_xe	0.00
Brand_smart	0.00
Model_xc90	0.00
Model_xc60	0.00
Model_1000	0.00
Model_6	0.00
Model_x3	0.00
Model_7	0.00
Model_x-trail	0.00
Model_a	0.00
Model_wr-v	0.00
Model_a7	0.00
Model_a8	0.00
Model_captur	0.00
Model_lodgy	0.00
Model_cayman	0.00
Model_evalia	0.00
Model_f	0.00
Model_fortuner	0.00

```
Model_fusion                0.00
Model_platinum              0.00
Model_petra                 0.00
Model_gallardo              0.00
Model_panamera              0.00
Model_pajero                0.00
Model_outlander             0.00
Model_glc                   0.00
Model_one                   0.00
Model_gls                   0.00
Model_nuvosport             0.00
Model_nexon                 0.00
Model_koleos                0.00
Model_kuv                   0.00
Model_mux                   0.00
Model_mustang               0.00
Model_montero               0.00
Model_punto                 0.00
Model_estilo                0.00
Model_slk-class             0.00
Model_enjoy                 0.00
Model_lancer                0.00
Model_sl-class              0.00
Model_siena                 0.00
Model_cedia                 0.00
Model_classic               0.00
Model_santa                 0.00
Model_cls-class             0.00
Model_safari                0.00
Model_clubman               0.00
Model_continental           0.00
Model_countryman            0.00
Model_rs5                   0.00
Model_cr-v                  0.00
Model_crosspolo             0.00
Model_renault               0.00
Model_cruze                 0.00
Model_dzire                 0.00
Model_e                     0.00
Model_qualis                0.00
Model_fortwo                0.00
```

Observation

- Power, Year and Engine are the top 3 important features of decision tree model

Random Forest

RandomForest Regressor is a powerful ensemble learning method used for regression tasks in machine learning. It combines the predictions of multiple decision trees into a single, more accurate and robust prediction.
How it Works:

- **Bootstrap Sampling**: RandomForest creates multiple subsets of the training data through bootstrapping, which means sampling with replacement. Each subset is used to train a decision tree.
- **Random Feature Selection**: At each node of each decision tree, a random subset of features is considered for splitting. This adds randomness and diversity to the model, helping to reduce overfitting.
- **Aggregation**: The predictions from all the individual decision trees are aggregated to make the final prediction. This is typically done by averaging the predictions for regression tasks.

Initializing the RandomForest Regressor Model

In [ ]:
```
rf = RandomForestRegressor(random_state = 1,oob_score = True)
"""
oob_score (Out-of-Bag score) is a way to estimate the model's performance wi
It leverages the way Random Forests are constructed to provide a built-in ev
"""
```

Out[ ]:
```
"\noob_score (Out-of-Bag score) is a way to estimate the model's performanc
e without the need for cross-validation.\nIt leverages the way Random Fores
ts are constructed to provide a built-in evaluation mechanism.\n"
```

Fitting the data to the model

In [ ]:
```
rf.fit(X_train,y_train['price_log'])
```

Out[ ]:
```
RandomForestRegressor
RandomForestRegressor(oob_score=True, random_state=1)
```

Getting the performance scores with the RandomForest Regressor

In [ ]:
```
RandomForest_model = get_model_score(rf)
```

```
R-square on training set :  0.9765731414802244
R-square on test set :  0.8477045450827858
RMSE on training set :  1.7100479788669742
RMSE on test set :  4.349335488727944
```

Observation

- Random Forest model has performed well on training and test set

Feature Importance

In [ ]:
```
# Importance of features in the model ( The importance of a feature is compu
# (normalized) total reduction of the criterion brought by that feature. It
print(pd.DataFrame(rf.feature_importances_, columns = ["Imp"], index = X_tra
```

	Imp
Power	0.61
Year	0.23
Engine	0.04
kilometers_driven_log	0.02
Mileage	0.01
New_Price	0.01
Location_Kolkata	0.00
Transmission_Manual	0.00
Brand_tata	0.00
Brand_land	0.00
Seats	0.00
Model_rover	0.00
Brand_honda	0.00
Location_Hyderabad	0.00
Brand_mahindra	0.00
Brand_mercedes-benz	0.00
Location_Coimbatore	0.00
Brand_mini	0.00
Owner_Type_Second	0.00
Fuel_Type_Diesel	0.00
Location_Bangalore	0.00
Fuel_Type_Petrol	0.00
Model_creta	0.00
Brand_skoda	0.00
Location_Delhi	0.00
Brand_bmw	0.00
Location_Mumbai	0.00
Brand_audi	0.00
Model_5	0.00
Location_Pune	0.00
Model_swift	0.00
Location_Jaipur	0.00
Model_a4	0.00
Brand_hyundai	0.00
Brand_toyota	0.00
Owner_Type_Third	0.00
Location_Kochi	0.00
Location_Chennai	0.00
Model_innova	0.00
Brand_chevrolet	0.00
Brand_volkswagen	0.00
Model_cayenne	0.00
Model_polo	0.00
Model_cooper	0.00
Brand_maruti	0.00
Model_nano	0.00
Model_ertiga	0.00
Model_amaze	0.00
Model_accord	0.00
Model_indica	0.00
Model_beat	0.00
Model_q5	0.00
Model_superb	0.00
Brand_ford	0.00
Model_i20	0.00

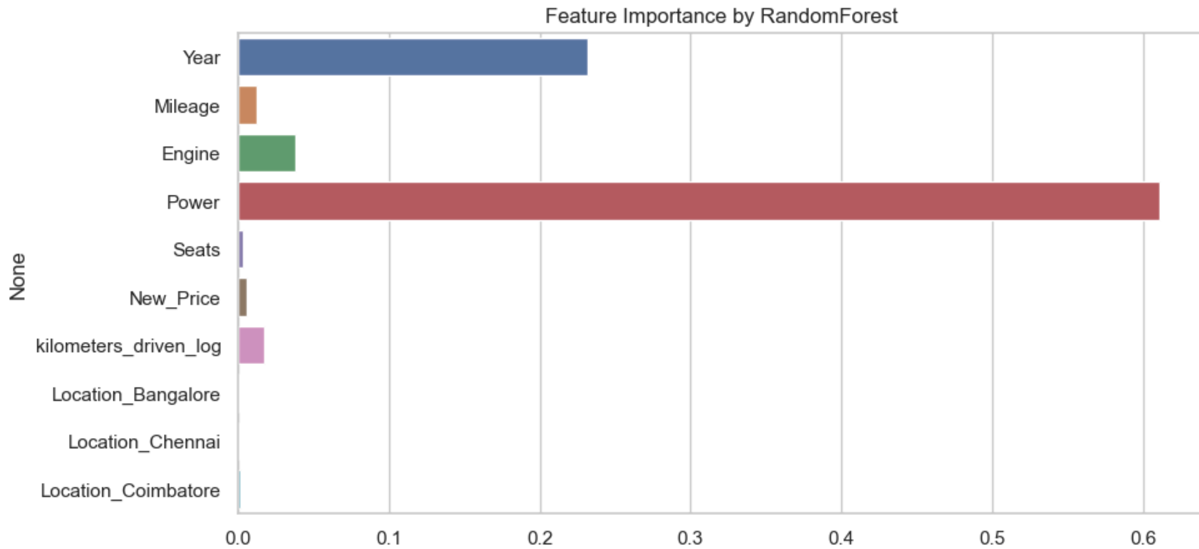
Model_figo	0.00
Model_etios	0.00
Brand_porsche	0.00
Model_city	0.00
Model_xylo	0.00
Model_new	0.00
Model_elantra	0.00
Model_corolla	0.00
Model_e-class	0.00
Model_i10	0.00
Model_cruze	0.00
Model_b	0.00
Model_ssangyong	0.00
Model_santa	0.00
Model_verna	0.00
Model_prius	0.00
Model_baleno	0.00
Model_3	0.00
Model_xenon	0.00
Model_sonata	0.00
Model_santro	0.00
Model_alto	0.00
Model_800	0.00
Model_xcent	0.00
Model_celerio	0.00
Model_m-class	0.00
Model_getz	0.00
Brand_renault	0.00
Model_manza	0.00
Model_ritz	0.00
Brand_nissan	0.00
Model_passat	0.00
Model_indigo	0.00
Fuel_Type_Electric	0.00
Model_x1	0.00
Model_laura	0.00
Model_zen	0.00
Model_7	0.00
Model_brio	0.00
Model_scorpio	0.00
Model_grand	0.00
Model_gl-class	0.00
Model_ecosport	0.00
Model_duster	0.00
Model_punto	0.00
Model_jazz	0.00
Model_ikon	0.00
Model_s	0.00
Brand_fiat	0.00
Model_wagon	0.00
Brand_mitsubishi	0.00
Model_fabia	0.00
Brand_jeep	0.00
Model_x5	0.00
Model_sx4	0.00
Model_vento	0.00

```
Model_optra                0.00
Model_compass              0.00
Model_bolero               0.00
Model_cayman               0.00
Brand_jaguar               0.00
Model_a6                   0.00
Model_xf                   0.00
Model_x3                   0.00
Model_fiesta               0.00
Model_elite                0.00
Brand_lamborghini          0.00
Model_jetta                0.00
Model_cr-v                 0.00
Model_gallardo             0.00
Model_accent               0.00
Model_octavia              0.00
Model_xuv500               0.00
Brand_volvo                0.00
Model_tigor                0.00
Owner_Type_Fourth & Above 0.00
Model_micra                0.00
Model_freestyle            0.00
Model_pajero               0.00
Model_civic                0.00
Model_terrano              0.00
Model_koleos               0.00
Model_glc                  0.00
Model_enjoy                0.00
Model_quanto               0.00
Model_s80                  0.00
Model_fortuner             0.00
Model_zest                 0.00
Model_linea                0.00
Model_sail                 0.00
Model_x6                   0.00
Model_sunny                0.00
Model_a8                   0.00
Model_gle                  0.00
Model_s-class              0.00
Model_fluence              0.00
Model_esteem               0.00
Model_ciaz                 0.00
Model_q3                   0.00
Model_omni                 0.00
Model_panamera             0.00
Model_endeavour            0.00
Model_a                    0.00
Model_logan                0.00
Model_rapid                0.00
Model_spark                0.00
Model_aveo                 0.00
Model_q7                   0.00
Model_sumo                 0.00
Model_eeco                 0.00
Model_s60                  0.00
Brand_datsun               0.00
```

Model_safari	0.00
Model_grande	0.00
Model_jeep	0.00
Model_6	0.00
Model_camry	0.00
Model_scala	0.00
Model_cla	0.00
Model_kuv	0.00
Model_estilo	0.00
Model_eon	0.00
Model_gla	0.00
Brand_isuzu	0.00
Model_continental	0.00
Model_yeti	0.00
Model_mobilio	0.00
Model_go	0.00
Model_teana	0.00
Model_aspire	0.00
Model_v40	0.00
Model_kwid	0.00
Model_xj	0.00
Model_rs5	0.00
Model_dzire	0.00
Model_lancer	0.00
Model_fortwo	0.00
Brand_smart	0.00
Model_tt	0.00
Model_wrv	0.00
Model_vitara	0.00
Model_qualis	0.00
Model_cls-class	0.00
Model_outlander	0.00
Model_xc60	0.00
Model_thar	0.00
Model_ameo	0.00
Model_ignis	0.00
Model_captur	0.00
Model_tiguan	0.00
Model_siena	0.00
Model_venture	0.00
Brand_force	0.00
Fuel_Type_LPG	0.00
Model_tavera	0.00
Model_r-class	0.00
Model_a3	0.00
Model_countryman	0.00
Model_hexa	0.00
Model_bolt	0.00
Model_pulse	0.00
Model_a-star	0.00
Model_mux	0.00
Model_xuv300	0.00
Model_tiago	0.00
Model_xc90	0.00
Model_redi-go	0.00
Model_tucson	0.00

```
Model_renault           0.00
Brand_bentley           0.00
Model_brv               0.00
Model_one               0.00
Model_d-max             0.00
Model_x-trail           0.00
Model_fusion            0.00
Model_petra             0.00
Model_nuvosport         0.00
Model_avventura         0.00
Model_tuv               0.00
Model_c-class           0.00
Model_crosspolo         0.00
Model_classic           0.00
Model_f                 0.00
Model_captiva           0.00
Model_verito            0.00
Model_mustang           0.00
Model_slk-class         0.00
Model_slc               0.00
Model_montero           0.00
Model_nexon             0.00
Model_gls               0.00
Model_lodgy             0.00
Model_redi              0.00
Model_br-v              0.00
Model_versa             0.00
Model_s-cross           0.00
Model_evalia            0.00
Model_z4                0.00
Model_clubman           0.00
Model_sl-class          0.00
Model_boxster           0.00
Model_xe                0.00
Model_a7                0.00
Model_beetle            0.00
Model_platinum          0.00
Model_1000              0.00
Model_e                 0.00
Model_cedia             0.00
Model_wr-v              0.00
```

In [ ]:
```
# Plotting the first 10 important features from RandomForest in the descendi
plt.figure(figsize = (10, 5))
sns.barplot(x = rf.feature_importances_[0:10], y = X_train.columns[0:10], hu
plt.title("Feature Importance by RandomForest")
plt.show()
```



Observation

- Power, Year and Engine are the top 3 important features of decision tree model

Hyperparameter Tuning - Decision Tree

In [ ]:
```
# Choose the type of regressor.
dtree_tuned = DecisionTreeRegressor(random_state = 1)

# Grid of parameters to choose from
parameters = {'max_depth': [5, 7, None],
             'min_samples_leaf': [1, 3, 5, 7],
             'max_leaf_nodes' : [2, 5, 7] + [None],
            }


# Type of scoring used to compare parameter combinations
scorer = metrics.make_scorer(metrics.r2_score)

# Run the grid search
grid_obj = GridSearchCV(dtree_tuned, parameters, scoring = scorer,cv = 5)
grid_obj = grid_obj.fit(X_train, y_train['price_log'])

# Set the model to the best combination of parameters
dtree_tuned = grid_obj.best_estimator_

# Fit the best algorithm to the data.
dtree_tuned.fit(X_train, y_train['price_log'])
```

Out[ ]:
```
▾ DecisionTreeRegressor
DecisionTreeRegressor(min_samples_leaf=3, random_state=1)
```

In [ ]:
```
dtree_tuned_score = get_model_score(dtree_tuned)
```

```
R-square on training set :  0.9523932899840497
R-square on test set :  0.7749466597662751
RMSE on training set :  2.43772860188823
RMSE on test set :  5.287156537830893
```

Observation

- Overfitting in decision tree is not there now.

Feature Importance

In [ ]:
```
# Importance of features in the tree building ( The importance of a feature
#(normalized) total reduction of the criterion brought by that feature. It i
print(pd.DataFrame(dtree_tuned.feature_importances_, columns = ["Imp"], inde
```

	Imp
Power	0.62
Year	0.24
Engine	0.05
Mileage	0.01
kilometers_driven_log	0.01
Transmission_Manual	0.00
Brand_honda	0.00
Brand_tata	0.00
Location_Kolkata	0.00
New_Price	0.00
Brand_mini	0.00
Brand_skoda	0.00
Fuel_Type_Diesel	0.00
Seats	0.00
Brand_mahindra	0.00
Model_a4	0.00
Location_Hyderabad	0.00
Location_Coimbatore	0.00
Model_5	0.00
Model_creta	0.00
Owner_Type_Second	0.00
Model_swift	0.00
Brand_hyundai	0.00
Brand_toyota	0.00
Location_Jaipur	0.00
Brand_audi	0.00
Owner_Type_Third	0.00
Model_xylo	0.00
Location_Delhi	0.00
Brand_mercedes-benz	0.00
Model_ertiga	0.00
Model_ssangyong	0.00
Location_Mumbai	0.00
Location_Bangalore	0.00
Model_nano	0.00
Model_beat	0.00
Location_Kochi	0.00
Fuel_Type_Petrol	0.00
Model_innova	0.00
Model_q5	0.00
Brand_chevrolet	0.00
Model_i10	0.00
Model_city	0.00
Model_amaze	0.00
Model_i20	0.00
Model_compass	0.00
Brand_ford	0.00
Model_x5	0.00
Model_brio	0.00
Model_new	0.00
Model_ecosport	0.00
Model_figo	0.00
Model_indica	0.00
Brand_maruti	0.00
Brand_bmw	0.00

```
Brand_volkswagen          0.00
Model_celerio             0.00
Model_3                   0.00
Model_scorpio             0.00
Model_rover               0.00
Model_duster              0.00
Location_Pune             0.00
Model_eeco                0.00
Model_elite               0.00
Model_7                   0.00
Brand_volvo               0.00
Model_vento               0.00
Model_terrano             0.00
Model_ritz                0.00
Brand_mitsubishi          0.00
Location_Chennai          0.00
Model_alto                0.00
Model_baleno              0.00
Model_x1                  0.00
Model_verna               0.00
Model_wagon               0.00
Model_corolla             0.00
Model_jazz                0.00
Model_q3                  0.00
Model_vitara              0.00
Model_a6                  0.00
Model_indigo              0.00
Model_grand               0.00
Model_gle                 0.00
Model_manza               0.00
Brand_nissan              0.00
Model_ciaz                0.00
Model_kwid                0.00
Model_yeti                0.00
Model_q7                  0.00
Model_redi                0.00
Model_redi-go             0.00
Model_qualis              0.00
Model_quanto              0.00
Model_punto               0.00
Model_r-class             0.00
Model_renault             0.00
Model_rapid               0.00
Model_x6                  0.00
Model_pulse               0.00
Model_outlander           0.00
Model_mustang             0.00
Model_mux                 0.00
Model_nexon               0.00
Model_nuvosport           0.00
Model_octavia             0.00
Model_omni                0.00
Model_one                 0.00
Model_optra               0.00
Model_pajero              0.00
Model_prius               0.00
```

```
Model_rs5                 0.00
Model_zen                 0.00
Model_z4                  0.00
Model_panamera            0.00
Model_passat              0.00
Model_petra               0.00
Model_platinum            0.00
Model_polo                0.00
Model_xuv500              0.00
Model_santa               0.00
Model_s                   0.00
Model_tucson              0.00
Model_tavera              0.00
Model_teana               0.00
Model_mobilio             0.00
Model_thar                0.00
Model_tiago               0.00
Model_tigor               0.00
Model_tiguan              0.00
Model_tt                  0.00
Model_xcent               0.00
Model_tuv                 0.00
Model_s-class             0.00
Model_v40                 0.00
Model_venture             0.00
Model_xc90                0.00
Model_verito              0.00
Model_versa               0.00
Model_wr-v                0.00
Model_wrv                 0.00
Model_x-trail             0.00
Model_x3                  0.00
Model_sx4                 0.00
Model_xe                  0.00
Model_superb              0.00
Model_sunny               0.00
Model_s-cross             0.00
Model_s60                 0.00
Model_s80                 0.00
Model_safari              0.00
Model_sail                0.00
Model_xc60                0.00
Model_santro              0.00
Model_xuv300              0.00
Model_scala               0.00
Model_siena               0.00
Model_sl-class            0.00
Model_xj                  0.00
Model_slc                 0.00
Model_slk-class           0.00
Model_sonata              0.00
Model_spark               0.00
Model_xf                  0.00
Model_xenon               0.00
Model_sumo                0.00
Model_montero             0.00
```

Model_fortwo	0.00
Model_micra	0.00
Model_m-class	0.00
Model_accord	0.00
Model_ameo	0.00
Model_aspire	0.00
Model_aveo	0.00
Model_avventura	0.00
Model_b	0.00
Model_beetle	0.00
Model_bolero	0.00
Model_bolt	0.00
Model_boxster	0.00
Model_br-v	0.00
Model_brv	0.00
Model_c-class	0.00
Model_camry	0.00
Model_captiva	0.00
Model_captur	0.00
Model_cayenne	0.00
Model_cayman	0.00
Model_cedia	0.00
Model_civic	0.00
Model_cla	0.00
Model_accent	0.00
Model_a8	0.00
Model_a7	0.00
Brand_jeep	0.00
Fuel_Type_Electric	0.00
Fuel_Type_LPG	0.00
Owner_Type_Fourth & Above	0.00
Brand_bentley	0.00
Brand_datsun	0.00
Brand_fiat	0.00
Brand_force	0.00
Brand_isuzu	0.00
Brand_jaguar	0.00
Brand_lamborghini	0.00
Model_a3	0.00
Brand_land	0.00
Brand_porsche	0.00
Brand_renault	0.00
Brand_smart	0.00
Model_1000	0.00
Model_6	0.00
Model_800	0.00
Model_a	0.00
Model_a-star	0.00
Model_classic	0.00
Model_cls-class	0.00
Model_clubman	0.00
Model_ignis	0.00
Model_gallardo	0.00
Model_getz	0.00
Model_gl-class	0.00
Model_gla	0.00

```
Model_glc                 0.00
Model_gls                 0.00
Model_go                  0.00
Model_grande              0.00
Model_hexa                0.00
Model_ikon                0.00
Model_freestyle           0.00
Model_jeep                0.00
Model_jetta               0.00
Model_koleos              0.00
Model_kuv                 0.00
Model_lancer              0.00
Model_laura               0.00
Model_linea               0.00
Model_lodgy               0.00
Model_logan               0.00
Model_fusion              0.00
Model_fortuner            0.00
Model_continental         0.00
Model_elantra             0.00
Model_cooper              0.00
Model_countryman          0.00
Model_cr-v                0.00
Model_crosspolo           0.00
Model_cruze               0.00
Model_d-max               0.00
Model_dzire               0.00
Model_e                   0.00
Model_e-class             0.00
Model_endeavour           0.00
Model_fluence             0.00
Model_enjoy               0.00
Model_eon                 0.00
Model_esteem              0.00
Model_estilo              0.00
Model_etios               0.00
Model_evalia              0.00
Model_f                   0.00
Model_fabia               0.00
Model_fiesta              0.00
Model_zest                0.00
```

Observation

- Power, Year and Engine are the top 3 important features of decision tree model

Hyperparameter Tuning - Random Forest

In [ ]:
```
# Choose the type of regressor

rf_tuned = RandomForestRegressor(random_state = 1,oob_score = True, n_jobs=-

# Grid of parameters to choose from
parameters = {
                'max_depth':[5,7,None],
```

```
                'max_features': ['sqrt','log2'],
                'n_estimators': [100, 200]
}

# Type of scoring used to compare parameter combinations
scorer = metrics.make_scorer(metrics.r2_score)

# Run the grid search
grid_obj = GridSearchCV(rf_tuned, parameters, scoring=scorer,cv=5)
grid_obj = grid_obj.fit(X_train, y_train['price_log'])

# Set the model to the best combination of parameters
rf_tuned = grid_obj.best_estimator_

# Fit the best algorithm to the data.
rf_tuned.fit(X_train, y_train['price_log'])
```

Out[ ]:

```
RandomForestRegressor

RandomForestRegressor(max_features='sqrt', n_estimators=200, n_jobs
=-1,
                      oob_score=True, random_state=1)
```

In [ ]:

```
rf_tuned_score = get_model_score(rf_tuned)
```

```
R-square on training set :  0.9695110418277427
R-square on test set :  0.8589538487865612
RMSE on training set :  1.9508440817108557
RMSE on test set :  4.18562250321649
```

Observation

- The Random Forest model does not perform any better after tuning.

Feature Importance

In [ ]:

```
# Importance of features in the tree building ( The importance of a feature
# (normalized) total reduction of the criterion brought by that feature. It

print(pd.DataFrame(rf_tuned.feature_importances_, columns = ["Imp"], index =
```

	Imp
Power	0.17
Engine	0.12
Year	0.11
Transmission_Manual	0.11
Mileage	0.05
kilometers_driven_log	0.04
Fuel_Type_Petrol	0.04
New_Price	0.03
Fuel_Type_Diesel	0.03
Brand_mercedes-benz	0.03
Seats	0.02
Brand_audi	0.02
Brand_bmw	0.01
Brand_maruti	0.01
Location_Coimbatore	0.01
Owner_Type_Second	0.01
Location_Kolkata	0.01
Brand_tata	0.01
Model_creta	0.01
Brand_land	0.00
Model_rover	0.00
Owner_Type_Third	0.00
Model_santro	0.00
Brand_hyundai	0.00
Brand_jaguar	0.00
Model_q7	0.00
Brand_toyota	0.00
Location_Kochi	0.00
Location_Jaipur	0.00
Brand_honda	0.00
Brand_chevrolet	0.00
Location_Hyderabad	0.00
Model_indica	0.00
Location_Pune	0.00
Model_fortuner	0.00
Brand_mahindra	0.00
Location_Mumbai	0.00
Model_alto	0.00
Model_e-class	0.00
Model_wagon	0.00
Location_Delhi	0.00
Model_innova	0.00
Location_Bangalore	0.00
Model_i10	0.00
Model_new	0.00
Brand_porsche	0.00
Model_5	0.00
Brand_skoda	0.00
Brand_mini	0.00
Location_Chennai	0.00
Model_nano	0.00
Brand_volkswagen	0.00
Model_zen	0.00
Model_3	0.00
Model_swift	0.00

Model_xuv500	0.00
Model_accent	0.00
Brand_ford	0.00
Model_xf	0.00
Model_indigo	0.00
Model_cooper	0.00
Model_figo	0.00
Model_city	0.00
Model_verna	0.00
Model_gle	0.00
Model_x5	0.00
Model_a6	0.00
Model_a4	0.00
Model_beat	0.00
Model_ciaz	0.00
Model_i20	0.00
Model_7	0.00
Model_corolla	0.00
Model_esteem	0.00
Model_ikon	0.00
Model_q5	0.00
Model_800	0.00
Model_superb	0.00
Model_baleno	0.00
Model_ecosport	0.00
Model_accord	0.00
Model_cayenne	0.00
Brand_renault	0.00
Model_octavia	0.00
Model_xj	0.00
Model_panamera	0.00
Model_ritz	0.00
Model_celerio	0.00
Model_getz	0.00
Model_laura	0.00
Model_scorpio	0.00
Model_cruze	0.00
Model_grand	0.00
Model_vitara	0.00
Model_vento	0.00
Model_gl-class	0.00
Model_polo	0.00
Model_aveo	0.00
Model_cr-v	0.00
Model_etios	0.00
Model_optra	0.00
Model_ertiga	0.00
Model_xylo	0.00
Model_slk-class	0.00
Model_m-class	0.00
Model_endeavour	0.00
Model_duster	0.00
Model_civic	0.00
Brand_fiat	0.00
Model_fiesta	0.00
Model_amaze	0.00

```
Model_manza                 0.00
Model_x6                    0.00
Model_jetta                 0.00
Model_brio                  0.00
Model_eon                   0.00
Model_omni                  0.00
Brand_nissan                0.00
Model_glc                   0.00
Model_x3                    0.00
Model_cla                   0.00
Model_elantra               0.00
Brand_lamborghini           0.00
Model_s                     0.00
Model_gla                   0.00
Model_camry                 0.00
Model_gallardo              0.00
Brand_jeep                  0.00
Model_sx4                   0.00
Brand_mitsubishi            0.00
Model_rapid                 0.00
Model_kwid                  0.00
Model_slc                   0.00
Model_q3                    0.00
Model_x1                    0.00
Model_cayman                0.00
Model_s-class               0.00
Model_compass               0.00
Model_bolero                0.00
Model_siena                 0.00
Model_ssangyong             0.00
Model_xcent                 0.00
Model_sonata                0.00
Brand_volvo                 0.00
Model_terrano               0.00
Model_tt                    0.00
Model_santa                 0.00
Model_pajero                0.00
Model_passat                0.00
Owner_Type_Fourth & Above   0.00
Model_elite                 0.00
Brand_bentley               0.00
Model_spark                 0.00
Model_6                     0.00
Model_fabia                 0.00
Model_micra                 0.00
Model_continental           0.00
Model_jazz                  0.00
Model_eeco                  0.00
Model_b                     0.00
Model_sumo                  0.00
Model_tucson                0.00
Model_clubman               0.00
Model_sunny                 0.00
Model_f                     0.00
Model_ameo                  0.00
Model_sail                  0.00
```

Model_a	0.00
Model_yeti	0.00
Model_a-star	0.00
Model_enjoy	0.00
Model_xenon	0.00
Model_zest	0.00
Model_captiva	0.00
Model_jeep	0.00
Model_quanto	0.00
Model_captur	0.00
Model_r-class	0.00
Brand_datsun	0.00
Model_punto	0.00
Model_koleos	0.00
Model_safari	0.00
Model_grande	0.00
Model_linea	0.00
Model_versa	0.00
Model_tigor	0.00
Model_petra	0.00
Model_estilo	0.00
Model_dzire	0.00
Model_xc60	0.00
Model_mobilio	0.00
Fuel_Type_Electric	0.00
Model_z4	0.00
Model_tiago	0.00
Model_lancer	0.00
Model_logan	0.00
Model_gls	0.00
Model_mustang	0.00
Model_a8	0.00
Model_tuv	0.00
Fuel_Type_LPG	0.00
Model_classic	0.00
Model_scala	0.00
Model_xc90	0.00
Model_redi-go	0.00
Model_freestyle	0.00
Model_thar	0.00
Model_a3	0.00
Model_brv	0.00
Model_countryman	0.00
Brand_isuzu	0.00
Brand_smart	0.00
Model_rs5	0.00
Model_qualis	0.00
Model_go	0.00
Model_prius	0.00
Model_x-trail	0.00
Model_d-max	0.00
Model_aspire	0.00
Model_v40	0.00
Model_verito	0.00
Model_s80	0.00
Model_cls-class	0.00

```
Model_s60                 0.00
Model_pulse               0.00
Model_sl-class            0.00
Model_bolt                0.00
Model_kuv                 0.00
Brand_force               0.00
Model_nexon               0.00
Model_tavera              0.00
Model_teana               0.00
Model_one                 0.00
Model_montero             0.00
Model_renault             0.00
Model_fluence             0.00
Model_venture             0.00
Model_fortwo              0.00
Model_wrv                 0.00
Model_xuv300              0.00
Model_tiguan              0.00
Model_outlander           0.00
Model_br-v                0.00
Model_ignis               0.00
Model_c-class             0.00
Model_nuvosport           0.00
Model_mux                 0.00
Model_s-cross             0.00
Model_crosspolo           0.00
Model_evalia              0.00
Model_hexa                0.00
Model_fusion              0.00
Model_avventura           0.00
Model_redi                0.00
Model_lodgy               0.00
Model_xe                  0.00
Model_1000                0.00
Model_wr-v                0.00
Model_platinum            0.00
Model_beetle              0.00
Model_boxster             0.00
Model_cedia               0.00
Model_e                   0.00
Model_a7                  0.00
```

Observation

- Power, Year and Engine are the top 3 important variables in predicting car price according to Random Forest

Conclusions and Recommendations

1. Comparison of various techniques and their relative performance based on chosen Metric (Measure of success):

Measures of success :

R-squared and RMSE can be used as a measure of success.

R-squared: This will tell us how much variation our predictive model can explain in data.

RMSE: This will give us a measure of how far off the model is predicting the original values on average.

In [ ]:
```
# Defining list of models
models = [lr,rdg,dtree, dtree_tuned, rf, rf_tuned]

# Defining empty lists to add train and test results
r2_train = []
r2_test = []
rmse_train= []
rmse_test= []

# Looping through all the models to get the rmse and r2 scores
for model in models:

    # Accuracy score
    j = get_model_score(model, False)
    r2_train.append(j[0])
    r2_test.append(j[1])
    rmse_train.append(j[2])
    rmse_test.append(j[3])
```

In [ ]:
```
comparison_frame = pd.DataFrame({'Model':['Linear Regression', 'Ridge Regres
                                          'Tuned Random Forest'],
                                          'Train_r2' : r2_train,'Test_r2' :
                                          'Train_RMSE' : rmse_train,'Test_RM
comparison_frame
```

Out[ ]:

	Model	Train_r2	Test_r2	Train_RMSE	Test_RMSE
0	Linear Regression	0.94	0.87	2.74	4.04
1	Ridge Regression	0.93	0.90	2.94	3.61
2	Decision Tree	1.00	0.81	0.02	4.80
3	Tuned Decision Tree	0.95	0.77	2.44	5.29
4	Random Forest	0.98	0.85	1.71	4.35
5	Tuned Random Forest	0.97	0.86	1.95	4.19

- Ridge Regression and Linear Regression have performed very well on data. However, Ridge Regression has given a more generalized model on training and test set
- There's still scope for improvement with tuning the hyperparameters of the Random Forest

2. Refined insights: **Name:**

- The `Name` column has 2041 unique values and this column would not be very useful in our analysis.

But the name contains both the brand name and the model name of the vehicle and we can process this column to extract Brand and Model names to reduce the number of levels

Extracting the car brands:

- After extracting the car brands from the name column we find that the most frequent brand in our data is Maruti and Hyundai

Extracting car model name:

- After extracting the car name it gets clear that our dataset contains used cars from luxury as well as budget-friendly brands
- The mean price of a used Lamborghini is 120 Lakhs and that of cars from other luxury brands follow in descending order and this output is very close to our expectation (domain knowledge), in terms of brand order.

Towards the bottom end, we have more budget friendly brands

Important variable with Linear Regression:

- According to the Linear Regresion model the most significant predictors of the price of used cars are -
    - Year
    - Power
    - New_price
    - Location
    - Kilometers_Driven
    - Fuel_Type
    - Owner_Type
    - Transmission

Important variable with Random Forest:

- According to the Random Forest model the most significant predictors of the price of used cars are

- Power of the engine

-The year of manufacturing -Engine -Mileage

3. Proposal for the final solution design:

Overall solution design :

The potential solution design would look like:

- Checking the data description to get the idea of basic statistics or summary of data
- Univariate analysis to see how data is spread out, getting to know about the outliers
- Bivariate analysis to see how different attributes vary with the dependent variable
- Outlier treatment if needed - In this case, outlier treatment is not necessary as outliers are the luxurious cars and in real world scenarios such cars would appear in data and we would want our predictive model to capture the underlying pattern for them
- Missing value treatment using appropriate techniques
- Feature engineering - transforming features, creating new features if possible
- Choosing the model evaluation technique - 1) R Squared 2) RMSE can be any other metrics related to regression analysis
- Splitting the data and proceeding with modeling
- Model tuning to see if the performance of the model can be improved further
- Since it is a regression problem we will first start with the parametric model - linear regression,Ridge Regression followed by the non-parametric models - Decision Tree and Random Forest

Best Model:

- The best solution can be determined by considering the combination of R-square and RMSE values for each

model on both the training and test datasets. A higher R-square indicates a better fit of the model to the data, while a lower RMSE indicates a lower error in the model's predictions. The model with the highest R-square and lowest RMSE on both the training and test sets would be considered the best solution

- Our final Ridge Regression model has an R-squared of ~0.89 on The test data, which means that our model can explain 89% variation in our data also the RMSE on test data is ~3.62 which means we can predict very closely to the original values. This is a very good model and we can use this model in production
- The model we should adopt is the Ridge Regression

model since it had a very good performance with both the train data and the test data

- Business can benefit by getting more cars under the hood:
  - From Tier 1 cities
  - First owner cars
  - Automatic transmission cars
  - High engine powered cars

- Some southern markets tend to have higher prices. It might be a good strategy to plan growth in southern cities using this information. Markets like Kolkata are very risky and we need to be careful about investments in these areas
- We will have to analyze the cost side of things before we can talk about profitability in the business. We should gather data regarding that
- The next step post that would be to cluster different sets of data and see if we should make multiple models for different locations/car types

- Now Car4U can price their cars competitively and maximize profit by predicting the optimal price for each car with the

Ridge Regression model