Background and 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 sellers 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.

Objective

1. Explore and visualize the dataset.
2. Build a linear regression model to predict the prices of used cars.
3. Generate a set of insights and recommendations that will help the business.

Variables to be analyzed

1. .No. : Serial Number
2. Name : Name of the car which includes Brand name and Model name
3. Location : The location in which the car is being sold or is available for purchase Cities
4. Year : Manufacturing year of the car
5. Kilometers_driven : The total kilometers driven in the car by the previous owner(s) in KM.
6. Fuel_Type : The type of fuel used by the car. (Petrol, Diesel, Electric, CNG, LPG)
7. Transmission : The type of transmission used by the car. (Automatic / Manual)
8. Owner : Type of ownership
9. Mileage : The standard mileage offered by the car company in kmpl or km/kg
10. Engine : The displacement volume of the engine in CC.
11. Power : The maximum power of the engine in bhp.
12. Seats : The number of seats in the car.
13. New_Price : The price of a new car of the same model in INR Lakhs.(1 Lakh = 100, 000)
14. Price : The price of the used car in INR Lakhs (1 Lakh = 100, 000)

Importing modules and packages

In [2]:
```
#import the important packages
import warnings
warnings.filterwarnings('ignore')
import pandas as pd #library used for data manipulation and analysis
import numpy as np # library used for working with arrays.
import matplotlib.pyplot as plt # library for plots and visualisations
import seaborn as sns # library for visualisations
import random
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn import linear_model
from scipy.stats import pearsonr
%matplotlib inline

import scipy.stats as stats # this library contains a large number of probability distributions as well as a grow
```

Reading and pre-processing the data

In [98]:
```
data = pd.read_csv('used_cars_data.csv', index_col = 0) #reading the data
np.random.seed(1)
data.sample(n = 10) #random sample of 10 values
```

Out[98]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	New_Price	Pric
S.No.													
	Ford												

2397	EcoSport 1.5 Petrol Trend	Kolkata	2016	21460	Petrol	Manual	First	17.0 kmpl	1497 CC	121.36 bhp	5.0	9.47 Lakh	6.
3777	Maruti Wagon R VXI 1.2	Kochi	2015	49818	Petrol	Manual	First	21.5 kmpl	1197 CC	81.80 bhp	5.0	5.44 Lakh	4.
4425	Ford Endeavour 4x2 XLT	Hyderabad	2007	130000	Diesel	Manual	First	13.1 kmpl	2499 CC	141 bhp	7.0	NaN	6.
3661	Mercedes-Benz E-Class E250 CDI Avantgrade	Coimbatore	2016	39753	Diesel	Automatic	First	13.0 kmpl	2143 CC	201.1 bhp	5.0	NaN	35.2
4514	Hyundai Xcent 1.2 Kappa AT SX Option	Kochi	2016	45560	Petrol	Automatic	First	16.9 kmpl	1197 CC	82 bhp	5.0	NaN	6.3
599	Toyota Innova Crysta 2.8 ZX AT	Coimbatore	2019	40674	Diesel	Automatic	First	11.36 kmpl	2755 CC	171.5 bhp	7.0	28.05 Lakh	24.8
186	Mercedes-Benz E-Class E250 CDI Avantgrade	Bangalore	2014	37382	Diesel	Automatic	First	13.0 kmpl	2143 CC	201.1 bhp	5.0	NaN	32.
305	Audi A6 2011-2015 2.0 TDI Premium Plus	Kochi	2014	61726	Diesel	Automatic	First	17.68 kmpl	1968 CC	174.33 bhp	5.0	NaN	20.7
4582	Hyundai i20 1.2 Magna	Kolkata	2011	36000	Petrol	Manual	First	18.5 kmpl	1197 CC	80 bhp	5.0	NaN	2.5
5434	Honda WR-V Edge Edition i-VTEC S	Kochi	2019	13913	Petrol	Manual	First	17.5 kmpl	1199 CC	88.7 bhp	5.0	9.36 Lakh	8.2

In [99]:
```
data.shape  #the shape of the data
```
Out[99]:
```
(7253, 13)
```

In [100…
```
data.info() #info of all the variables
```
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 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: 793.3+ KB
```

Observation:

1. Based on the above data types, Mileage, Engine, Power and New_Price have to be changed to float values
2. Name variable column can be further analyzed into useful data.
3. Year should be changed into date time/integer value or it can be changed into the age of the car since manfucturing till 2020.

In [101…
```
data.isnull().sum().sort_values(ascending=False) #sum of all the null values per variable
```
```
New_Price            6247
```

```
Out[101… New_Price              6247
         Price                  1234
         Seats                    53
         Power                    46
         Engine                   46
         Mileage                   2
         Owner_Type                0
         Transmission              0
         Fuel_Type                 0
         Kilometers_Driven         0
         Year                      0
         Location                  0
         Name                      0
         dtype: int64
```

```
In [102… data.drop(['New_Price'], axis = 1, inplace = True) #droppung the new_price variable
```

Observation:

1. New_Price has more than 50% of it's values missing, so that column can be dropped.
2. Seats, Power, Engine almost have equal number of null values, meaning atleast 3 or 4 variables should be missing per row.

```
In [103… def mileage_to_num(input): #converitng mileage to a number
    if isinstance(input, str):
        if input.split()[1] == 'km/kg':
            return float(input.replace('km/kg', ''))
        else:
            return float(input.replace('kmpl', ''))
    else:
        return np.nan


def engine_to_num(input): #converting engine to a number value
    if isinstance(input, str):
        return float(input.replace('CC', ''))
    else:
        return np.nan


def power_to_num(input): #converting power to a number value
    if isinstance(input, str):
        if input.split()[0] == 'null':
            return np.nan
        else:
            return float(input.replace('bhp', ''))
    else:
        return np.nan


def nprice_to_num(input):
    if isinstance(input, str):
        if input.split()[1] == 'Lakh':
            return float(input.replace('Lakh', ''))
        else:
            return float(input.replace('Cr', '')) * 100
    else:
        return np.nan
```

```
In [104… col_transforms = {
    'Mileage': mileage_to_num,
    'Engine': engine_to_num,
    'Power': power_to_num,
}
 #making a ddictiornary of all variables to be converted paired withe appropriate function
# k is the key, so the column name here

# v is the value, which a function in this case and is
#     either `height_to_num` or `weight_to_num`
for k,v in col_transforms.items():
    data[k] = data[k].map(v)
```

```
In [105… data.info() #checking the data info now
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7253 entries, 0 to 7252
Data columns (total 12 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   float64
 8   Engine             7207 non-null   float64
 9   Power              7078 non-null   float64
 10  Seats              7200 non-null   float64
 11  Price              6019 non-null   float64
dtypes: float64(5), int64(2), object(5)
memory usage: 736.6+ KB
```

In [106…

```
new_name  = [i.split()[0] for i in data.Name] #clearing up the Name column
data.Name = new_name
```

In [107…

```
age_cars = [(2020 - i) for i in data.Year] #Changing the Year column to Age of cars
data.Year = age_cars
data = data.rename(columns={'Year': 'Age_of_Car'})
```

In [108…

```
data.nunique(dropna = False) #unique values from each variable
```

Out[108…

```
Name                   33
Location               11
Age_of_Car             23
Kilometers_Driven    3660
Fuel_Type               5
Transmission            2
Owner_Type              4
Mileage               439
Engine                151
Power                 384
Seats                  10
Price                1374
dtype: int64
```

In [109…

```
data.isnull().sum().sort_values(ascending=False) #checking the total number of null values per variable
```

Out[109…

```
Price                1234
Power                 175
Seats                  53
Engine                 46
Mileage                 2
Owner_Type              0
Transmission            0
Fuel_Type               0
Kilometers_Driven       0
Age_of_Car              0
Location                0
Name                    0
dtype: int64
```

In [110…

```
data.isnull().sum(axis=1).value_counts() #null values per row
```

Out[110…

```
0    5872
1    1308
3      36
2      27
4      10
dtype: int64
```

In [111…

```
num_missing = data.isnull().sum(axis=1) #storing null values per row into num_missing
num_missing.value_counts()
```

Out[111…

```
0    5872
1    1308
3      36
2      27
4      10
dtype: int64
```

In [112…

```
# these are missing of Power, Seats and/or Price
data[num_missing == 2].sample(n=5)
```

Out[112…

	Name	Location	Age_of_Car	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	Price

S.No.												
6723	Ford	Kolkata	11	39408	Diesel	Manual	First	17.80	1399.0	NaN	5.0	NaN
3882	Maruti	Kolkata	10	40000	Petrol	Manual	Second	19.50	1061.0	NaN	NaN	2.50
6957	Honda	Kochi	1	11574	Petrol	Manual	First	0.00	1199.0	88.7	NaN	NaN
6896	Toyota	Hyderabad	7	86000	Diesel	Manual	First	23.59	1364.0	NaN	5.0	NaN
5893	Maruti	Chennai	12	51000	Petrol	Manual	Second	19.50	1061.0	NaN	NaN	1.75

In [113…

```
# these are missing Engine, Power and Seats
data[num_missing == 3].sample(n=5)
```

Out[113…

S.No.	Name	Location	Age_of_Car	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	Price
4604	Honda	Pune	9	98000	Petrol	Manual	First	16.70	NaN	NaN	NaN	3.15
2780	Hyundai	Pune	11	100000	Petrol	Manual	First	0.00	NaN	NaN	NaN	1.60
4577	BMW	Delhi	8	72000	Diesel	Automatic	Third	18.48	NaN	NaN	NaN	13.85
194	Honda	Ahmedabad	13	60006	Petrol	Manual	First	0.00	NaN	NaN	NaN	2.95
2530	BMW	Kochi	6	64158	Diesel	Automatic	First	18.48	NaN	NaN	NaN	17.89

In [114…

```
# these are missing Engine, Power, Seats, Price
data[num_missing == 4].sample(n=10)
```

Out[114…

S.No.	Name	Location	Age_of_Car	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	Price
6633	Mahindra	Kolkata	4	27000	Diesel	Manual	First	0.00	NaN	NaN	NaN	NaN
6643	BMW	Bangalore	11	150000	Diesel	Automatic	Second	18.48	NaN	NaN	NaN	NaN
6651	Maruti	Kolkata	5	36009	Petrol	Manual	First	16.10	NaN	NaN	NaN	NaN
6042	Skoda	Bangalore	11	72000	Petrol	Manual	Second	17.50	NaN	NaN	NaN	NaN
6677	Fiat	Jaipur	10	65000	Petrol	Manual	Third	14.60	NaN	NaN	NaN	NaN
6880	BMW	Chennai	11	95000	Diesel	Automatic	Second	18.48	NaN	NaN	NaN	NaN
6902	Toyota	Kochi	8	59311	Petrol	Manual	First	18.30	NaN	NaN	NaN	NaN
6541	Toyota	Bangalore	8	56600	Diesel	Manual	First	23.59	NaN	NaN	NaN	NaN
6544	Hyundai	Bangalore	8	58000	Petrol	Automatic	Second	15.00	NaN	NaN	NaN	NaN
6685	Maruti	Pune	10	115000	Petrol	Manual	Second	16.10	NaN	NaN	NaN	NaN

In [115…

```
for n in num_missing.value_counts().sort_index().index: #printing out where the variables are gone missing
    if n > 0:
        print(f'For the rows with exactly {n} missing values, NAs are found in:')
        n_miss_per_col = data[num_missing == n].isnull().sum()
        print(n_miss_per_col[n_miss_per_col > 0])
        print('\n\n')
```

```
For the rows with exactly 1 missing values, NAs are found in:
Mileage       2
Power       103
Seats         2
Price      1201
dtype: int64



For the rows with exactly 2 missing values, NAs are found in:
Power    26
Seats     5
Price    23
dtype: int64



For the rows with exactly 3 missing values, NAs are found in:
Engine    36
Power     36
Seats     36
dtype: int64
```

```
For the rows with exactly 4 missing values, NAs are found in:
Engine    10
Power     10
Seats     10
Price     10
dtype: int64
```

In [163…

```
data['Name'] = data['Name'].astype('category')
data['Location'] = data['Location'].astype('category')
data['Fuel_Type'] = data['Fuel_Type'].astype('category') #converting to categorical variables
data['Transmission'] = data['Transmission'].astype('category')
data['Owner_Type'] = data['Owner_Type'].astype('category')
data['Location'] = data['Location'].astype('category')
```

In [164…

```
data['Engine'].fillna(data['Engine'].median(), inplace=True)  # median imputation
data['Power'].fillna(data['Power'].median(), inplace=True)
data['Mileage'].fillna(data['Mileage'].median(), inplace=True)
data['Price'].fillna(data['Price'].median(), inplace=True)
data['Seats'].fillna(data['Seats'].median(), inplace=True)
```

In [165…

```
void_list = ['Engine', 'Power', 'Mileage', 'Price', 'Seats']  #checking the null values again
for i in void_list: #checking if all the null values have been filled
    print(f'{i} null values are: {data[i].isnull().sum()}')
```

```
Engine null values are: 0
Power null values are: 0
Mileage null values are: 0
Price null values are: 0
Seats null values are: 0
```

In [166…

```
data.isnull().sum().sort_values(ascending=False)
```

Out[166…

```
Price                0
Seats                0
Power                0
Engine               0
Mileage              0
Owner_Type           0
Transmission         0
Fuel_Type            0
Kilometers_Driven    0
Age_of_Car           0
Location             0
Name                 0
dtype: int64
```

In [167…

```
print(data.describe().T) #statistics of all the variables
```

```
                    count          mean           std      min       25%  \
Age_of_Car         7253.0      6.619054      3.198739    1.000      4.00   
Kilometers_Driven  7253.0  56277.365918  30187.237813  171.000  34000.00   
Mileage            7253.0     18.205135      4.304204    6.275     15.17   
Engine             7253.0   1608.226941    563.680643   72.000   1198.00   
Power              7253.0    110.402768     45.948236   34.200     77.00   
Seats              7253.0      5.000000      0.000000    5.000      5.00   
Price              7253.0      6.843612      4.214570    0.440      3.85   

                        50%       75%         max  
Age_of_Car             6.00      9.00      16.500  
Kilometers_Driven  53416.00  73000.00  131500.000  
Mileage               18.16     21.10      29.995  
Engine              1493.00   1968.00    3123.000  
Power                 94.00    138.03     229.575  
Seats                  5.00      5.00       5.000  
Price                  5.64      8.40      15.225  
```

Observation:

1. Age of car may be normally distributed as mean is very close to the median
2. Mileage is normally distributed as mean and median are equal.

3. For kilometers driven there is a huge range in the data
4. Power and Price huge range in data

In [168…

```
print(data.describe(include = 'category')) #categorical variables statistics
```

```
          Name Location Fuel_Type Transmission Owner_Type
count     7253     7253      7253         7253       7253
unique      33       11         5            2          4
top     Maruti   Mumbai    Diesel       Manual      First
freq      1444      949      3852         5204       5952
```

Observation:

1. There are 10 unique values for seats, with 5 being the most common, which makes sense.
2. Interesting that there are 5 differnt Fuel_Type cars, with disel being the most common
3. Only two types of transmission, which makes sense.
4. A car can have more than 4 owners which is very small as compared to the other values in the owner column

EDA

In [ ]:

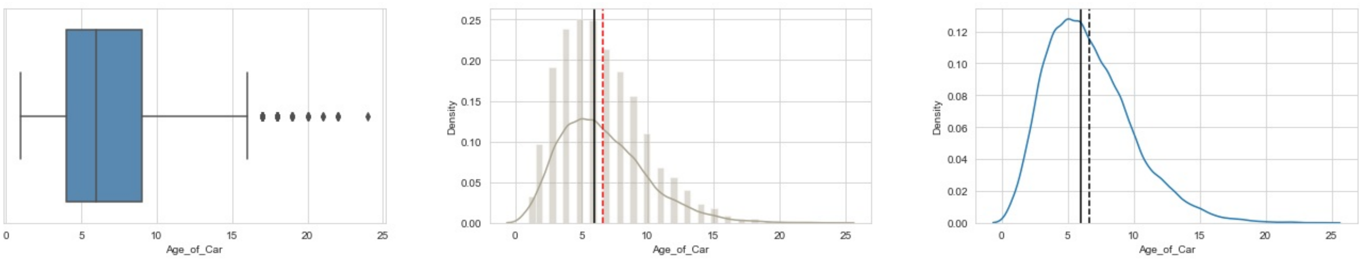
```
def histogram_boxplot(feature):
    """ Boxplot and histogram combined
    feature: 1-d feature array
    """
    figure, (ax_box2, ax_hist2, ax_hist3) = plt.subplots(
        nrows = 1, ncols=3,# Number of rows of the subplot grid= 2
        figsize  = (20,5)) # creating the 2 subplots
    figure.tight_layout(pad = 7)
    sns.boxplot(x = feature,ax=ax_box2, color = '#4B8BBE', orient = 'v') # boxplot will be created
    sns.distplot(feature, kde=True, ax=ax_hist2, color = '#a9a38f') # For histogram
    sns.distplot(feature, kde= True, ax=ax_hist3, hist = False) #Making an outline of the histogram
    ax_hist2.axvline(np.mean(feature), color='r', linestyle='--') # Add mean to the histogram
    ax_hist2.axvline(np.median(feature), color='black', linestyle='-') # Add median to the histogram
    ax_hist3.axvline(np.mean(feature), color = 'black', linestyle = '--') #Adding mean to second histogram
    ax_hist3.axvline(np.median(feature), color='black', linestyle='-') #Adding median to second histogram
```

Univariate Analysis

Observations on Age of Car

In [121…

```
histogram_boxplot(data.Age_of_Car)
```



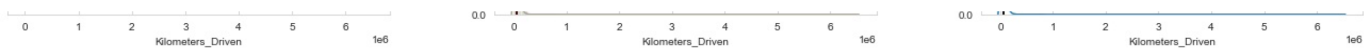
Observation:

1. There are some outliers
2. The graph is right skewed.

Observations on Kilometeres Driven

In [122…

```
histogram_boxplot(data['Kilometers_Driven'])
```



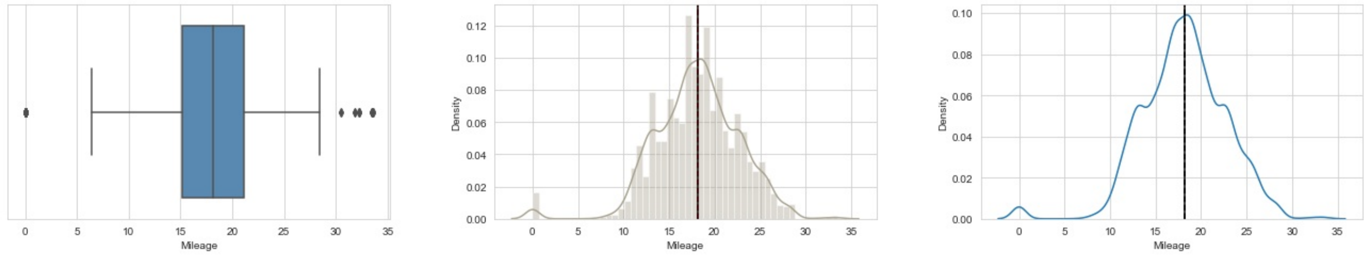
Observation:

1. Heavily right skewed graph
2. Maybe log scale transformation/arcsinh transformation may be applicable here.
3. There is clearly one distinct outlier

Observations on Mileage

In [123…

```
histogram_boxplot(data['Mileage'])
```



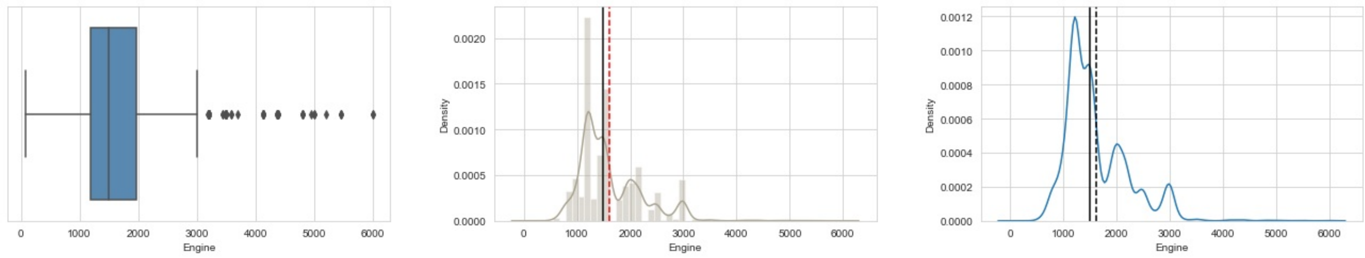
Observation:

1. One outlier towards the low end and few outliers towards the right side.
2. Not accounting for the outlier, the graph looks normally distributed.

Observations on Engine

In [124…

```
histogram_boxplot(data['Engine'])
```



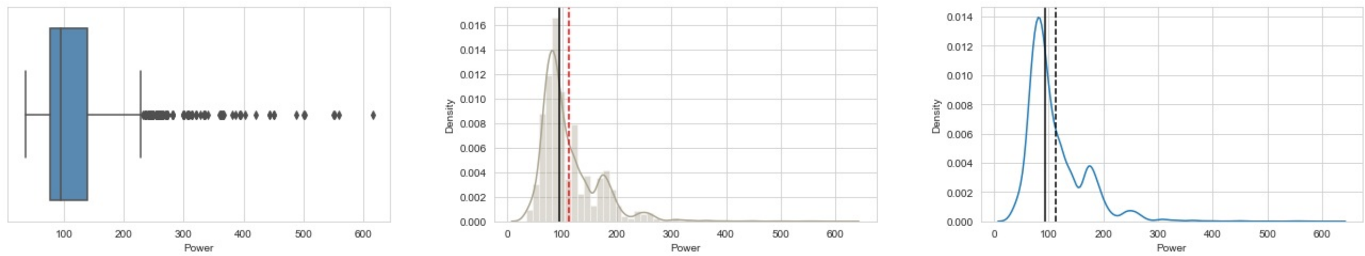
Observation:

1. There are many outliers.
2. Graph is slightly right skewed
3. The max value of engine 6000 is very close to the median.

Observations on Power

In [125…

```
histogram_boxplot(data['Power'])
```



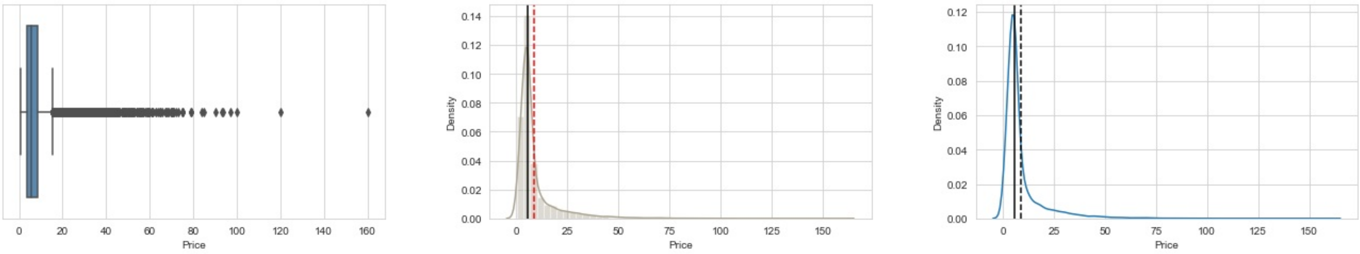
Observation:

1. Numerous Outliers present.
2. Heavily right skewed graph.
3. Outlier treatment like dropping outliers need to be considered.

Observations on Price

In [126…

```
histogram_boxplot(data['Price'])
```



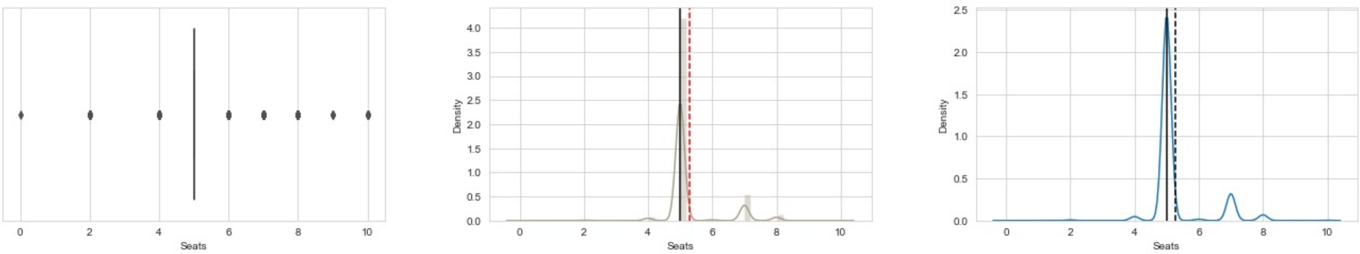
Observations:

1. Numerous outliers seen.
2. Right Skewed graph.
3. Log transformation can be completed along with outlier treatments.

Observation on Seats

In [127…

```
histogram_boxplot(data['Seats'])
```



Observation:

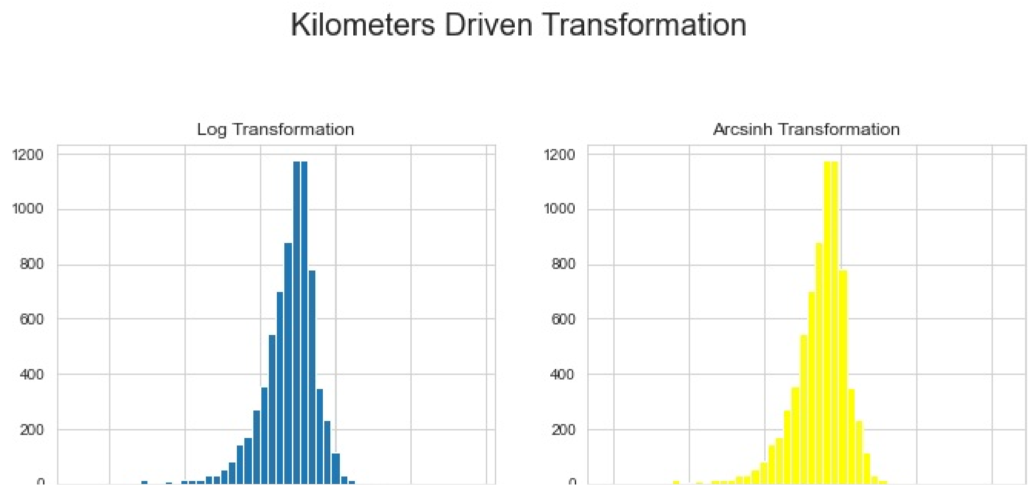
1. Many outliers visible.
2. Most of the data is centered at 5.
3. Shows a normal distribution with mean of 5.

Applying Log and Arcsinh Transformations

Transformation for Kilometers Driven

In [128…

```
fig, axs = plt.subplots(1, 2, figsize=(10,5)) #making a subplot
axs[0].hist(x = np.log(data['Kilometers_Driven']), bins = 50) #log transformation of km driven
axs[0].set_title('Log Transformation')
axs[1].hist(x = np.arcsinh(data['Kilometers_Driven']), bins = 50, color = 'yellow') #arcsinh transformation of km
axs[1].set_title('Arcsinh Transformation')
plt.suptitle('Kilometers Driven Transformation',fontsize=20)
fig.tight_layout(pad=3.0)
plt.show()
```



Observations:

1. The log and arcsinh look similar so the difference there is more be about interpretation.
2. The log transformation decreases the scale of the distributions, even with the huge range of Kilometeters driven. It seems the outliers caused the log-transformed distributions to still be a bit skewed, but it is closer to normal than the original distribution.

Transformation for Price

In [129…

```python
fig, axs = plt.subplots(1, 2, figsize=(10,5)) #making a subplot
axs[0].hist(x = np.log(data['Price']), bins = 50) #log transformation of km driven
axs[0].set_title('Log Transformation')
axs[1].hist(x = np.arcsinh(data['Price']), bins = 50, color = 'yellow') #arcsinh transformation of km driven
axs[1].set_title('Arcsinh Transformation')
plt.suptitle('Price Transformation',fontsize=20)
fig.tight_layout(pad=3.0)
plt.show()
```

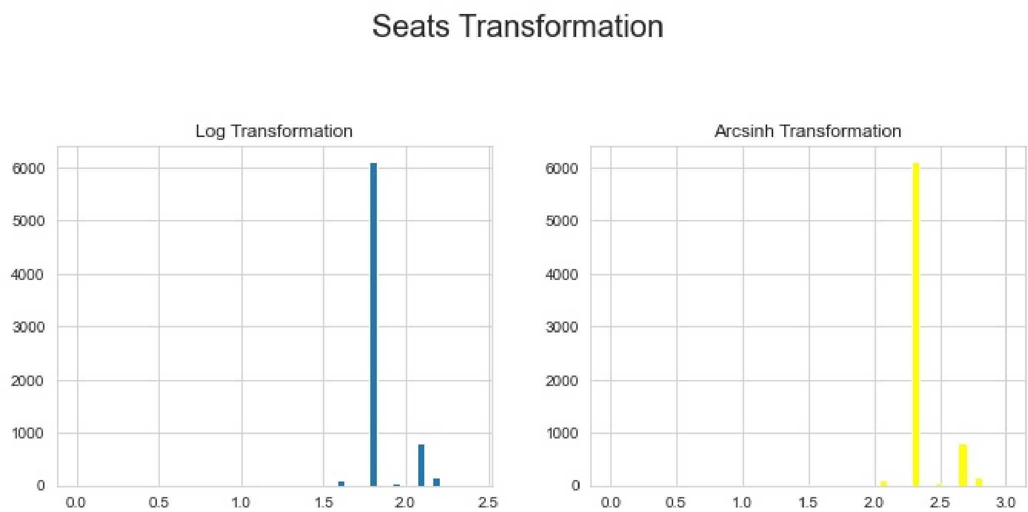
Observations:

1. The log and arcsinh look similar so the difference there is more be about interpretation.
2. There is onviously one value that is very different from others.
3. Maybe removing tht one huge value the graph may look more normally distributed.

Transformations for Seats

In [130…

```python
fig, axs = plt.subplots(1, 2, figsize=(10,5)) #making a subplot
axs[0].hist(x = np.log(data['Seats'] + 1), bins = 50) #log transformation of seats
axs[0].set_title('Log Transformation')
axs[1].hist(x = np.arcsinh(data['Seats']), bins = 50, color = 'yellow') #arcsinh transformation of seats
axs[1].set_title('Arcsinh Transformation')
plt.suptitle('Seats Transformation',fontsize=20)
fig.tight_layout(pad=3.0)
plt.show()
```



Observations:

1. From the above, there were a lot of 0 values so log(value + 1) transformation has to be applied.
2. There dosen't seem to be a big change in either of the two graphs, the outliers are pushing the graph to be more left skewed than previous graph.

Outlier treatments

In [131…

```
# Let's treat outliers by flooring and capping
def treat_outliers(df, col):
    """
    treats outliers in a variable
    col: str, name of the numerical variable
    df: dataframe
    col: name of the column
    """
    Q1 = df[col].quantile(0.25)  # 25th quantile
    Q3 = df[col].quantile(0.75)  # 75th quantile
    IQR = Q3 - Q1
    Lower_Whisker = Q1 - 1.5 * IQR
    Upper_Whisker = Q3 + 1.5 * IQR

    # all the values smaller than Lower_Whisker will be assigned the value of Lower_Whisker
    # all the values greater than Upper_Whisker will be assigned the value of Upper_Whisker
    df[col] = np.clip(df[col], Lower_Whisker, Upper_Whisker)

    return df


def treat_outliers_all(df, col_list):
    """
    treat outlier in all numerical variables
    col_list: list of numerical variables
    df: data frame
    """
    for c in col_list:
        df = treat_outliers(df, c)

    return df
```

In [132…

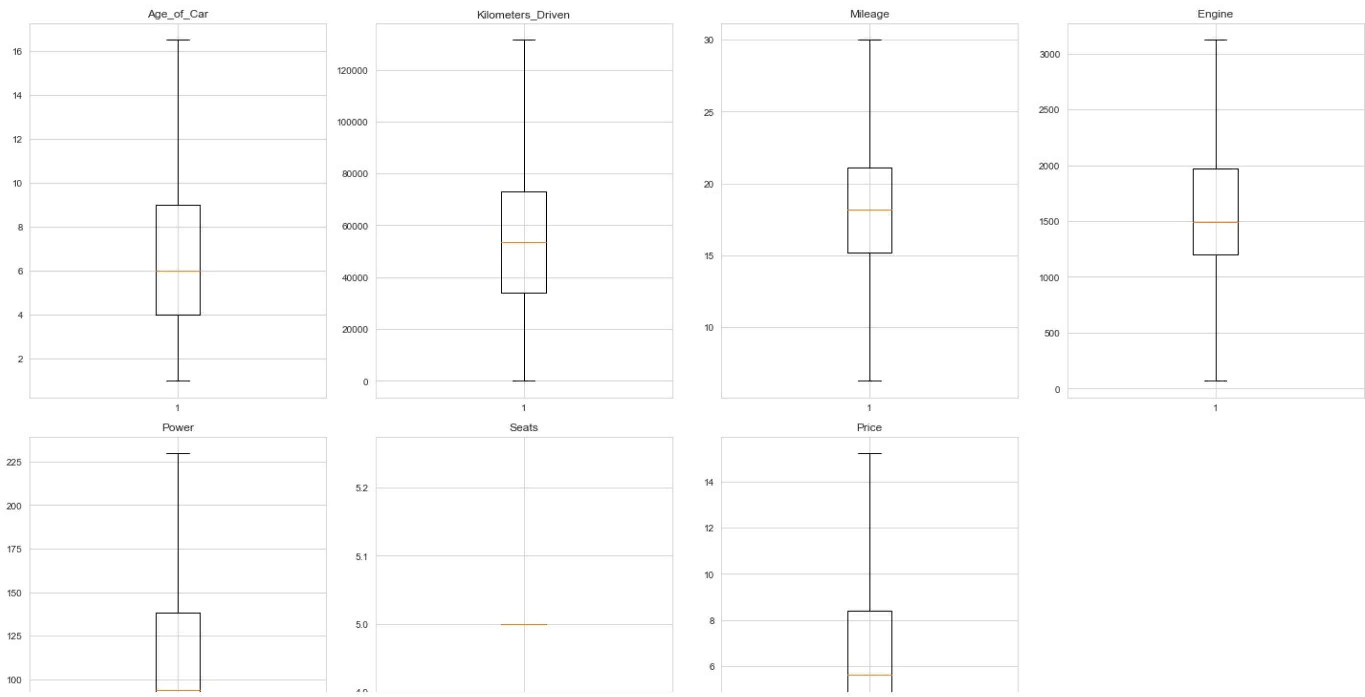
```
numeric_columns = data.select_dtypes(include=np.number).columns.tolist()#numberic columns
numerical_col = data.select_dtypes(include=np.number).columns.tolist()#converting numerical cols
data = treat_outliers_all(data, numerical_col) #treating outliers
```

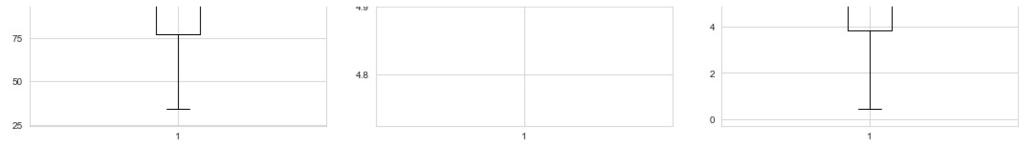
In [133…

```
plt.figure(figsize=(20, 30))

for i, variable in enumerate(numeric_columns): #boxplots subplots
    plt.subplot(5, 4, i + 1)
    plt.boxplot(data[variable], whis=1.5)
    plt.tight_layout()
    plt.title(variable)

plt.show()
```





Observation:

1. As it can be seen above, all outliers has been treated and no outliers remain.
2. The graph for seats only appear to be a straight line, or all the points seem to be at 5 seats.

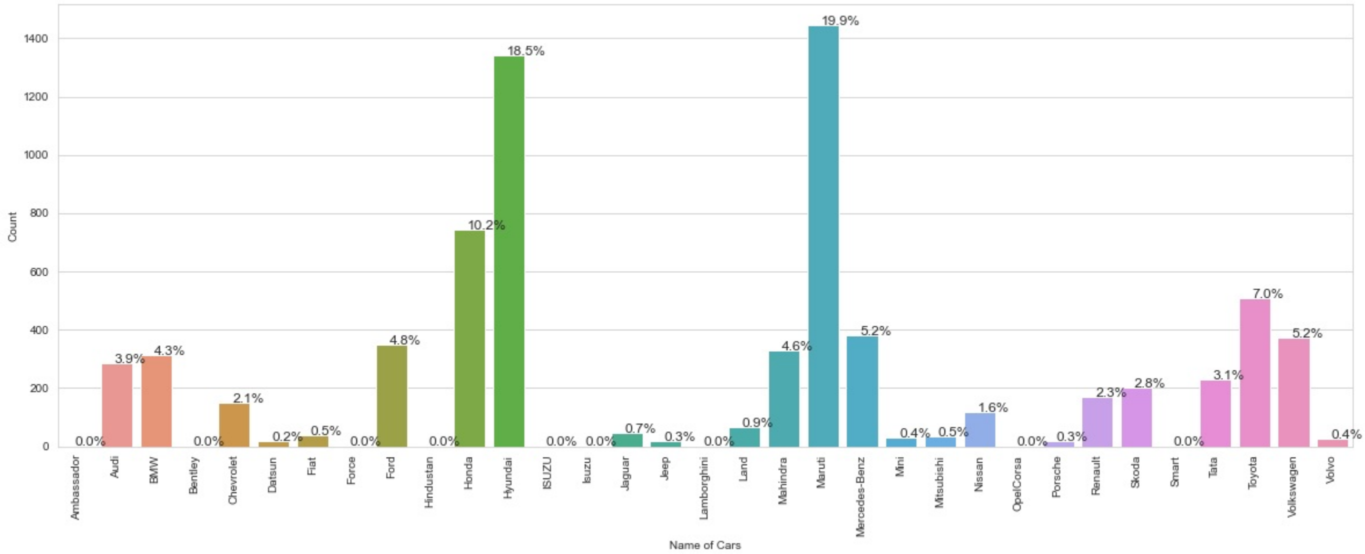
In [134…
```
data['Name'] = data['Name'].astype('category')
data['Location'] = data['Location'].astype('category')
data['Fuel_Type'] = data['Fuel_Type'].astype('category') #converting to categorical variables
data['Transmission'] = data['Transmission'].astype('category')
data['Owner_Type'] = data['Owner_Type'].astype('category')
data['Location'] = data['Location'].astype('category')
```

Categorical Variables

In [135…
```
def bar_perc(plot, feature):
    '''
    plot
    feature: 1-d categorical feature array
    '''
    total = len(feature) # length of the column
    for p in ax.patches:
        percentage = '{:.1f}%'.format(100 * p.get_height()/total) # percentage of each class of the category
        x = p.get_x() + p.get_width() / 2 - 0.05 # width of the plot
        y = p.get_y() + p.get_height()           # hieght of the plot
        ax.annotate(percentage, (x, y), size = 12) # annotate the percantage
```

Observations on Name

In [205…
```
plt.figure(figsize=(20,7))
ax = sns.countplot(data['Name']) #count plot for Name
plt.xlabel('Name of Cars')
plt.xticks(rotation=90)
plt.ylabel('Count')
bar_perc(ax,data['Name'])
```



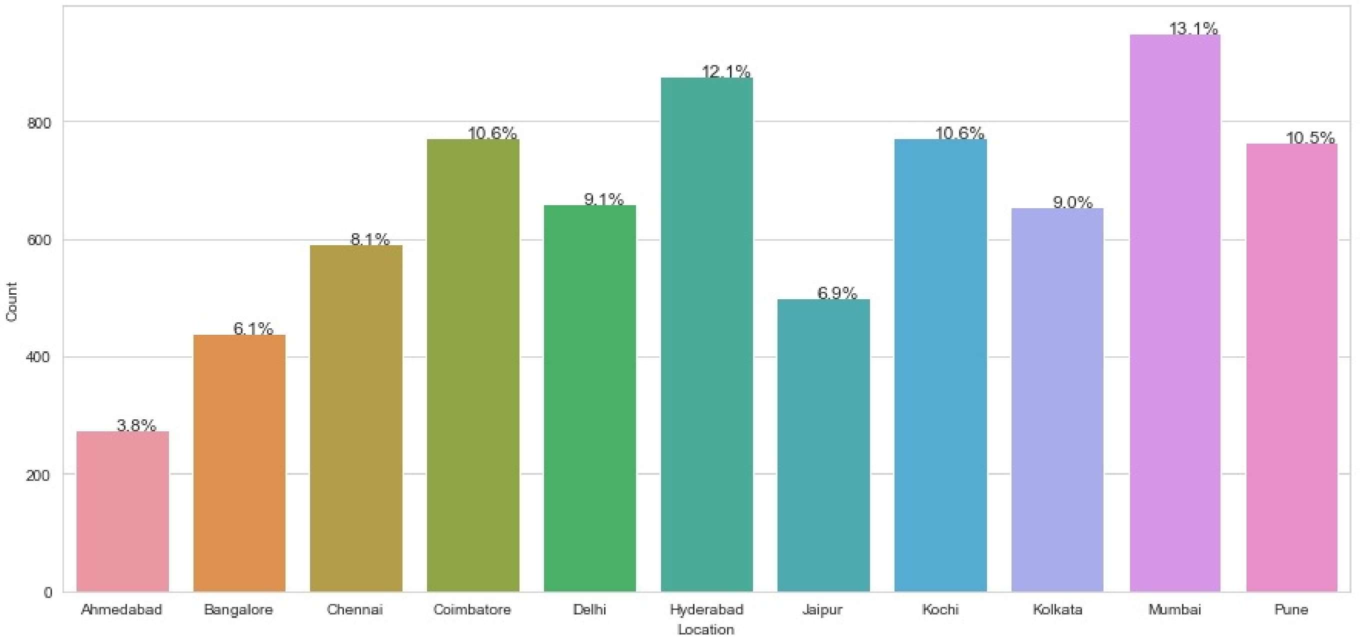
Observations:

1. Most common is Maruthi with 19.9%
2. Second most common car is Hyundar with 18.5 %
3. least common are Bentley, Ambassado, Jeep etc

Observations on Location

In [136…
```
plt.figure(figsize=(15,7))
ax = sns.countplot(data['Location']) #count plot for Location
plt.xlabel('Location')
```

```
plt.ylabel('Count')
bar_perc(ax,data['Location'])
```



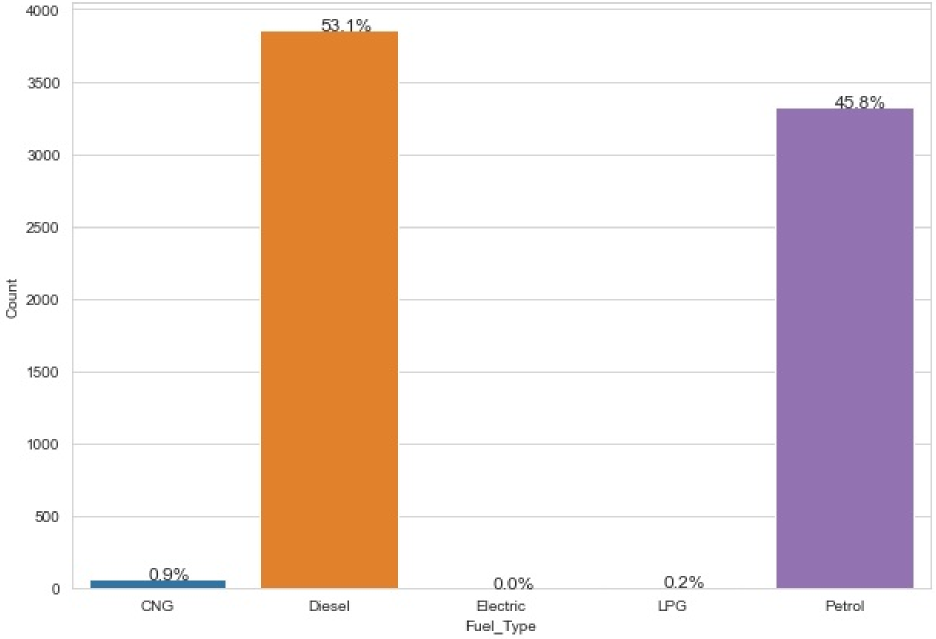
Observation:

1. Mumbai and Hyderbad are the most common Locations.
2. The least common location is Ahmedabad

Observation on Fuel Type

In [137…

```
plt.figure(figsize=(10,7))
ax = sns.countplot(data['Fuel_Type'])
plt.xlabel('Fuel_Type')
plt.ylabel('Count')
bar_perc(ax,data['Fuel_Type'])
```



Observation:

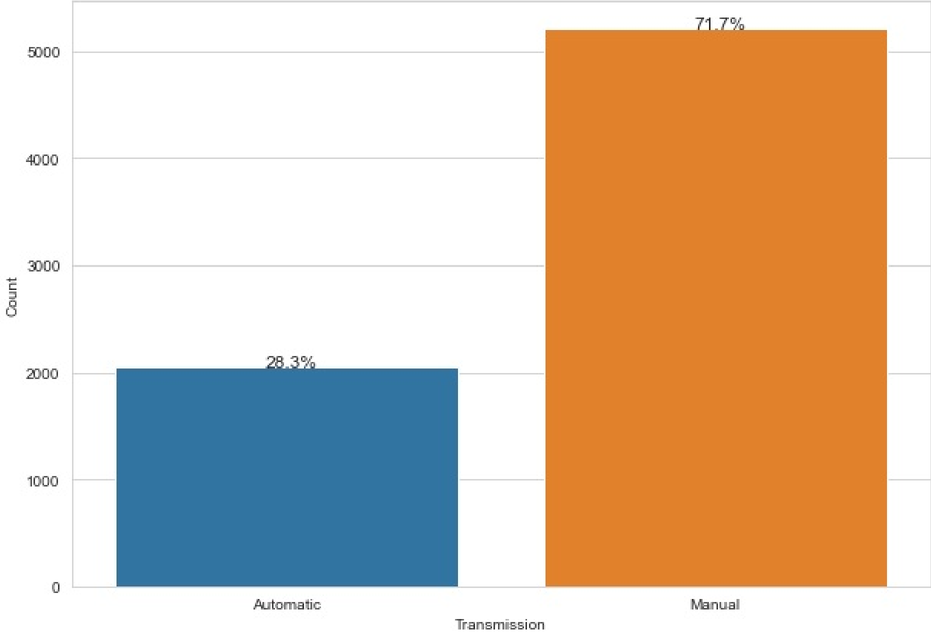
1. The most common type of Fuel is Diesel with 53.1% followed by petrol 45.8%
2. The least common fuel type is the electric car, there was only 2 values of electric car in the whole data.

Observation on Transmission Type

In [138…

```
plt.figure(figsize=(10,7))
ax = sns.countplot(data['Transmission'])
```

```
plt.xlabel('Transmission')
plt.ylabel('Count')
bar_perc(ax,data['Transmission'])
```

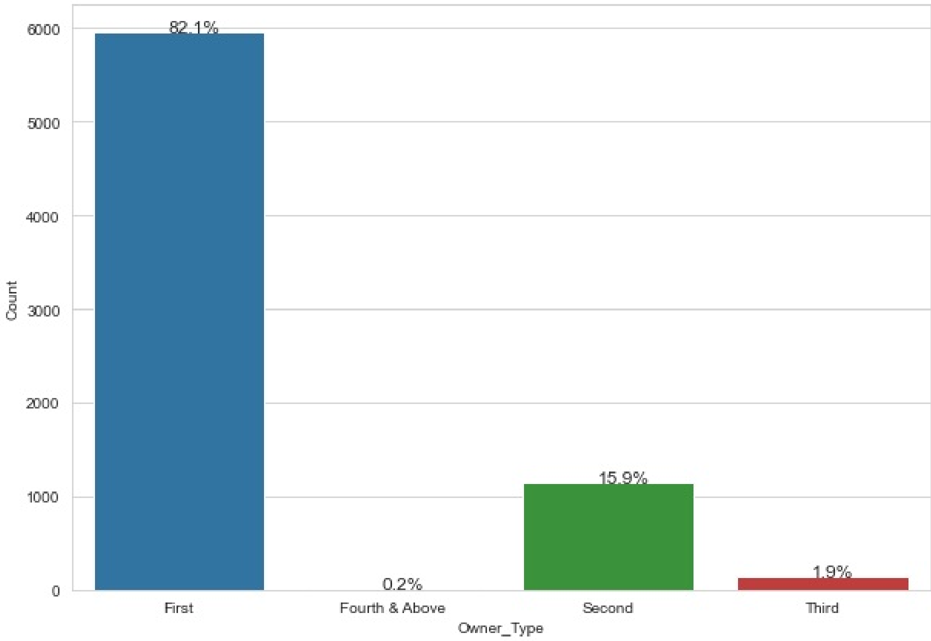


Observation:

1. Manual is the most common type of transmission with 71.%
2. Automatic is the least common type of transmission

Observation on Owner Type

In [139…
```
plt.figure(figsize=(10,7))
ax = sns.countplot(data['Owner_Type'])
plt.xlabel('Owner_Type')
plt.ylabel('Count')
bar_perc(ax,data['Owner_Type'])
```



Observation:

1. First hand owners are mostly popular with 82.1% with the least being 0.2%

Bivariate Analysis

In [141…
```
data.corr() #correlation of data
```

Out[141...

	Age_of_Car	Kilometers_Driven	Mileage	Engine	Power	Seats	Price
Age_of_Car	1.000000	0.519522	-0.321942	0.049200	-0.035288	NaN	-0.371188
Kilometers_Driven	0.519522	1.000000	-0.160162	0.186120	0.021601	NaN	-0.152816
Mileage	-0.321942	-0.160162	1.000000	-0.621567	-0.539260	NaN	-0.304916
Engine	0.049200	0.186120	-0.621567	1.000000	0.846878	NaN	0.646981
Power	-0.035288	0.021601	-0.539260	0.846878	1.000000	NaN	0.727125
Seats	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Price	-0.371188	-0.152816	-0.304916	0.646981	0.727125	NaN	1.000000

In [142...

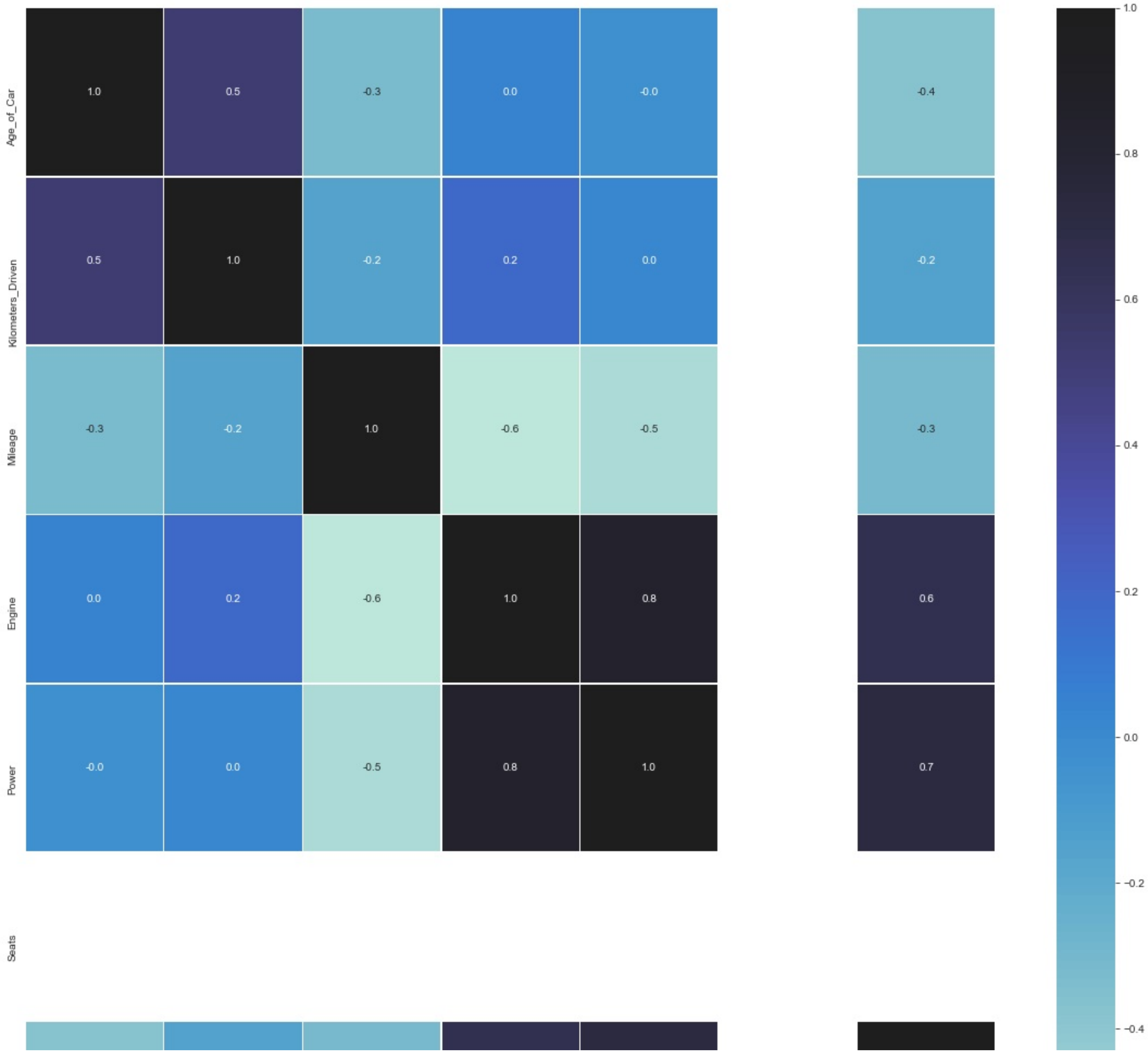
```
data.cov() #covariance of data
```

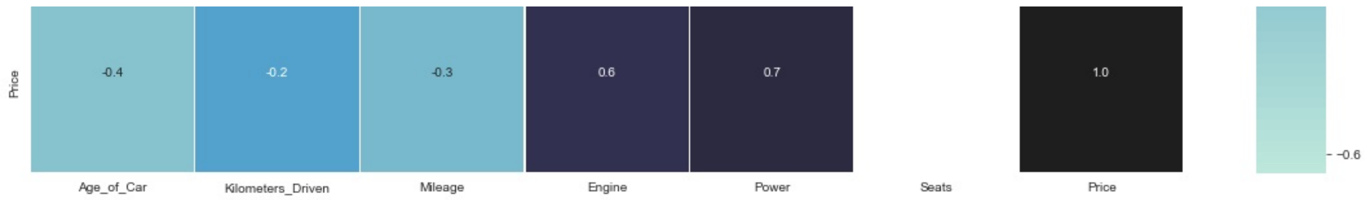
Out[142...

	Age_of_Car	Kilometers_Driven	Mileage	Engine	Power	Seats	Price
Age_of_Car	10.231929	5.016561e+04	-4.432510	8.871112e+01	-5.186477	0.0	-5.004103
Kilometers_Driven	50165.611008	9.112693e+08	-20810.125109	3.167007e+06	29961.524356	0.0	-19442.230069
Mileage	-4.432510	-2.081013e+04	18.526176	-1.508044e+03	-106.649682	0.0	-5.531284
Engine	88.711119	3.167007e+06	-1508.043931	3.177359e+05	21934.259628	0.0	1537.014171
Power	-5.186477	2.996152e+04	-106.649682	2.193426e+04	2111.240382	0.0	140.809224
Seats	0.000000	0.000000e+00	0.000000	0.000000e+00	0.000000	0.0	0.000000
Price	-5.004103	-1.944223e+04	-5.531284	1.537014e+03	140.809224	0.0	17.762604

In [143...

```
plt.figure(figsize=(20,20))
sns.heatmap(data.corr(), annot=True, linewidths=.5, fmt= '.1f', center = 1 )  # heatmap
plt.show()
```





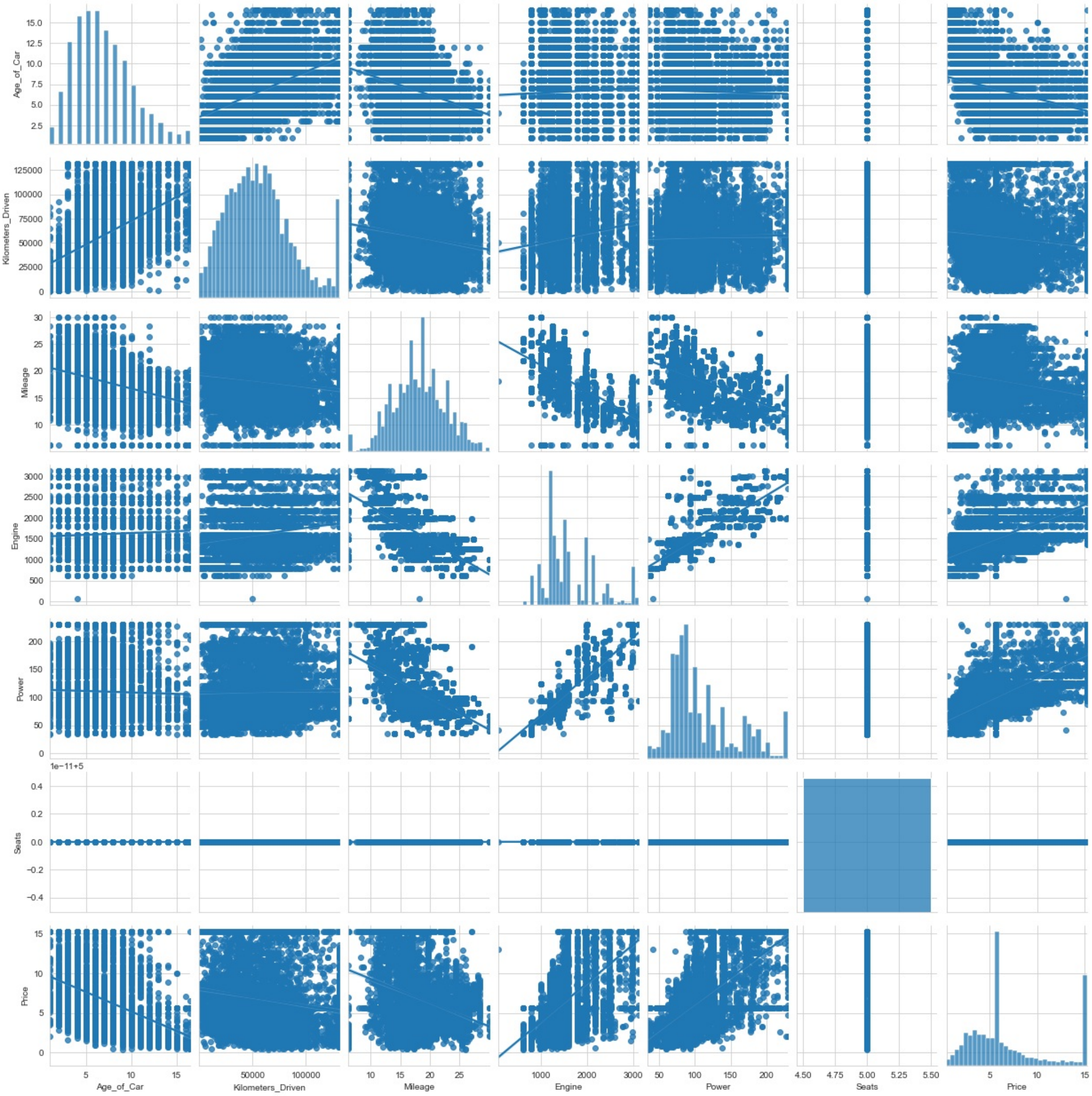
Observation:

1. Engine seems to have a high correlation with Price.
2. Power and Engine, Power and Price have high correlation
3. Seats and other variables don't really correlate well
4. Correlation does not imply cassaution.
5. Mileage and Engine correlate well negatively.

In [144…

```
plt.figure(figsize = (20,20))
sns.pairplot(data = data, kind = 'reg') #pairplot
plt.show()
```

```
<Figure size 1440x1440 with 0 Axes>
```



Observation:

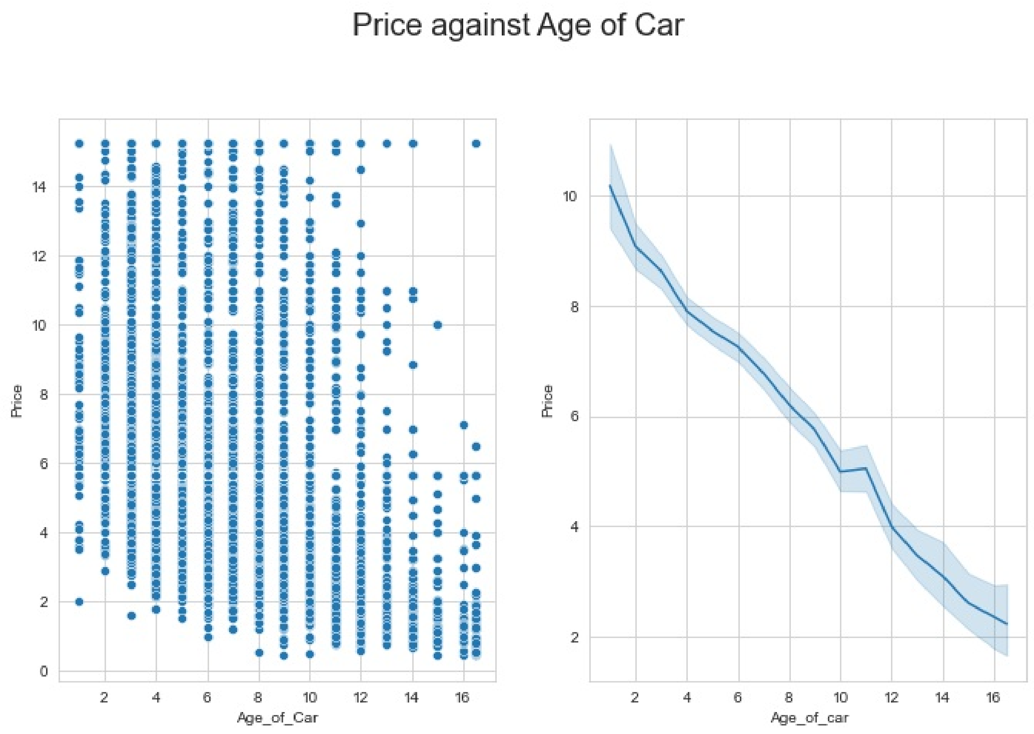
1. As indicated by the heat map most of the variables don't correlate that well with each other, other than the few exceptions.
2. Price and Power are positively correlated
3. Engine and Price are slightly positively correlated as well.
4. Power and Engine are positively correlated as well.
5. Power as a representation of itself is very rightly skewed.
6. Seats and Engine have a small positive correlation amongst each other.
7. Negative correlation for Mileage and Engine.
8. Negative correlation for Power and Mileage
9. Key variables that correlate well with Price are in the order of Power, Engine, Mileage and Kilometers Driven

Numerical vs Numerical

In [145…

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10,7)) #creating subplots
ax1 = sns.scatterplot(x = 'Age_of_Car', ax = ax1,  y = 'Price', data = data) #scatterplot fo age vs price
ax2 = sns.lineplot(x = 'Age_of_Car', ax = ax2, y = 'Price', data = data) #lineplot of age vs price
plt.suptitle('Price against Age of Car',fontsize=20)
plt.xlabel('Age_of_car')
fig.tight_layout(pad=3.0)
plt.ylabel('Price')
```

Out[145… Text(372.11363636363626, 0.5, 'Price')



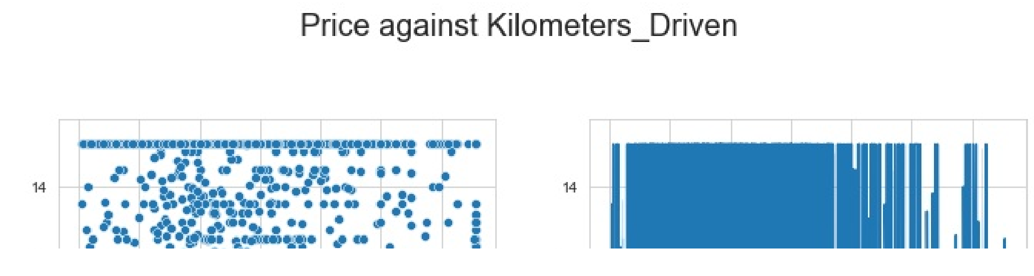
Observation:

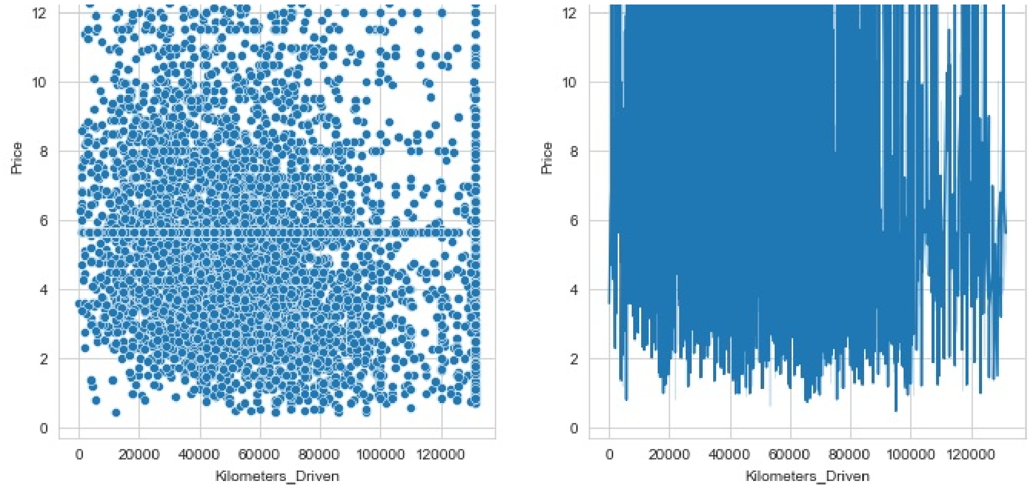
1. There is a negative correlation between price and age of car
2. The relationship may not be entirely linear but maybe of cubic or maybe quadratic.
3. There does seem to be distincive outliers.

In [146…

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10,7))
ax1 = sns.scatterplot(x = 'Kilometers_Driven', ax = ax1,  y = 'Price', data = data)
ax2 = sns.lineplot(x = 'Kilometers_Driven', ax = ax2, y = 'Price', data = data)
plt.suptitle('Price against Kilometers_Driven',fontsize=20)
plt.xlabel('Kilometers_Driven')
fig.tight_layout(pad=3.0)
plt.ylabel('Price')
```

Out[146… Text(372.11363636363626, 0.5, 'Price')



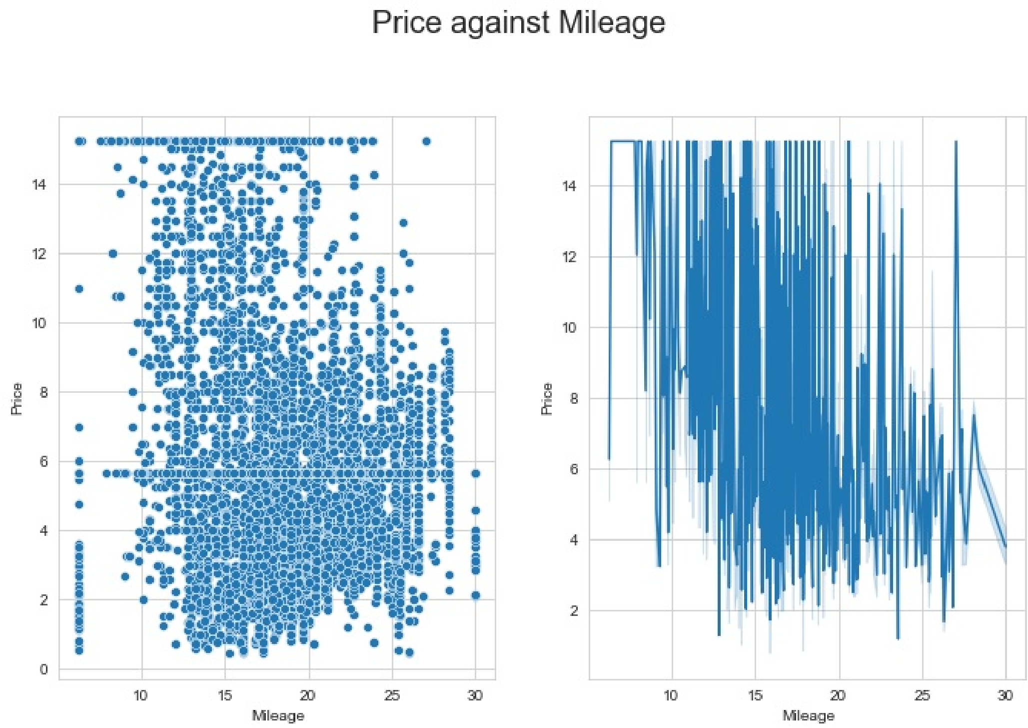


Observations:

1. A relationship can't be well established between the two variables.
2. Most of the data stack on each other beween 0 and 1 Kilometers driven while the price extends from 0 to 160.
3. There is a clear outlier in Kilometeres driven above 6 which is causing the line plot to extend to the right.

In [147…
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10,7))
ax1 = sns.scatterplot(x = 'Mileage', ax = ax1,  y = 'Price', data = data)
ax2 = sns.lineplot(x = 'Mileage', ax = ax2, y = 'Price', data = data)
plt.suptitle('Price against Mileage',fontsize=20)
plt.xlabel('Mileage')
fig.tight_layout(pad=3.0)
plt.ylabel('Price')
```

Out[147… `Text(372.11363636363626, 0.5, 'Price')`



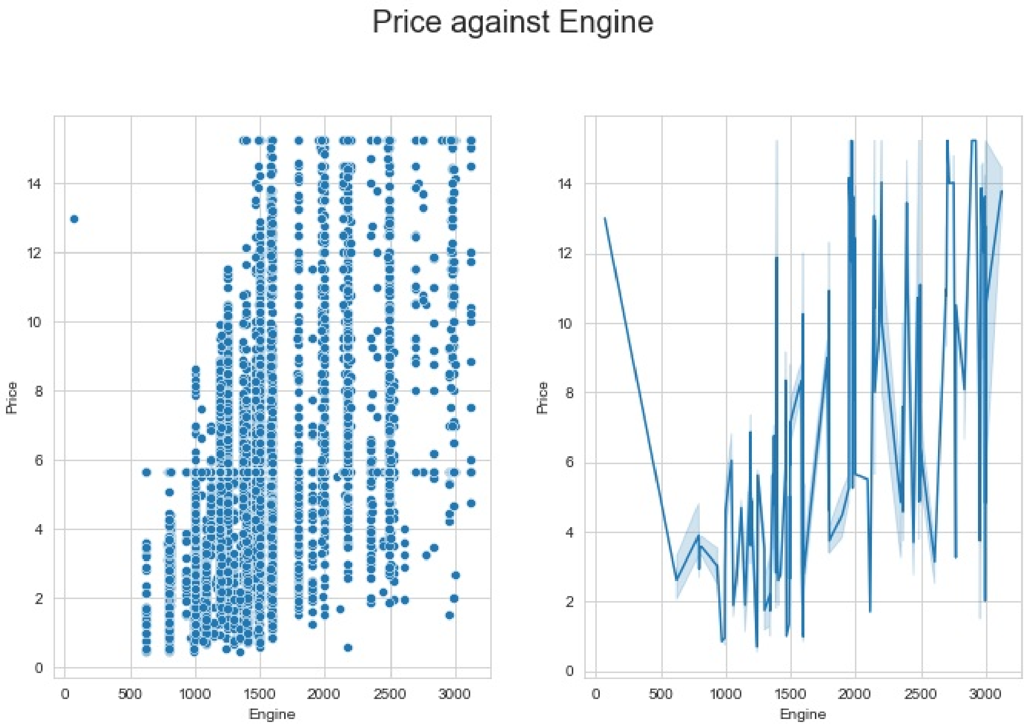
Observations:

1. There dosen't seem to be a clear strong relationship between the to variables.
2. A negative relationship can be slightly established between the two variables.
3. There is 2 clear distinctive outlier points.

In [148…
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10,7))
ax1 = sns.scatterplot(x = 'Engine', ax = ax1,  y = 'Price', data = data)
ax2 = sns.lineplot(x = 'Engine', ax = ax2, y = 'Price', data = data)
plt.suptitle('Price against Engine',fontsize=20)
plt.xlabel('Engine')
fig.tight_layout(pad=3.0)
```

```
plt.ylabel('Price')
```

Out[148… Text(372.11363636363626, 0.5, 'Price')

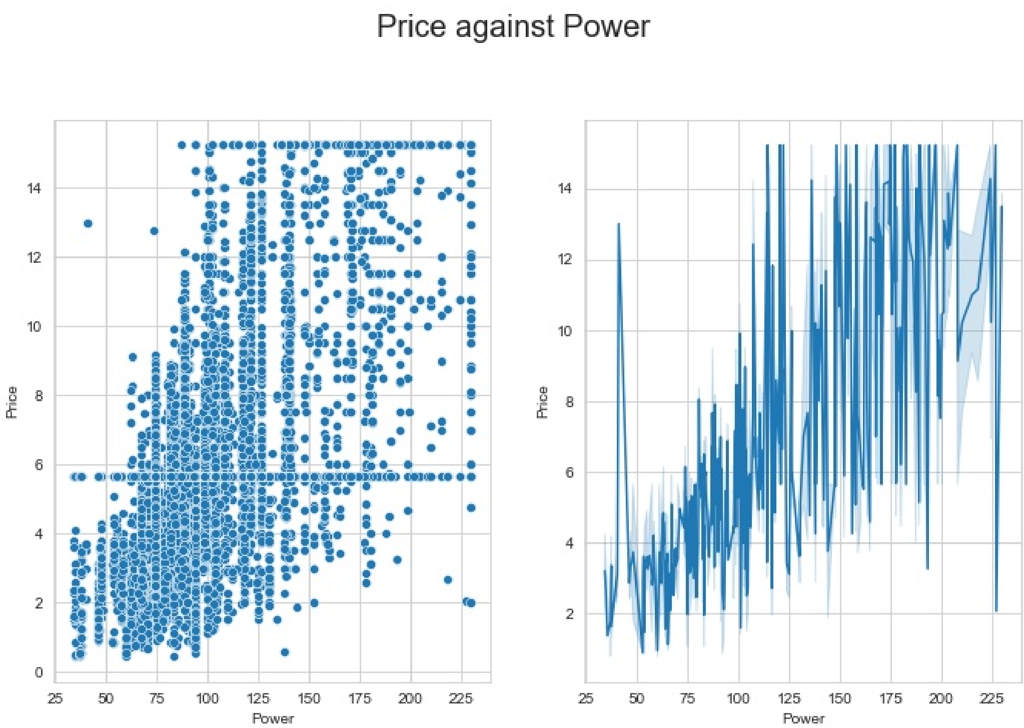


Observations:

1. There is definitely a relationship between the two variables.
2. Positive correlation can be established between the two variables.
3. Although there is a rise in price as the engine values increase it, isn't a steady rise in relationship.

In [149…
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(10,7))
ax1 = sns.scatterplot(x = 'Power', ax = ax1,  y = 'Price', data = data)
ax2 = sns.lineplot(x = 'Power', ax = ax2, y = 'Price', data = data)
plt.suptitle('Price against Power',fontsize=20)
plt.xlabel('Power')
fig.tight_layout(pad=3.0)
plt.ylabel('Price')
```

Out[149… Text(372.11363636363626, 0.5, 'Price')



Observation:

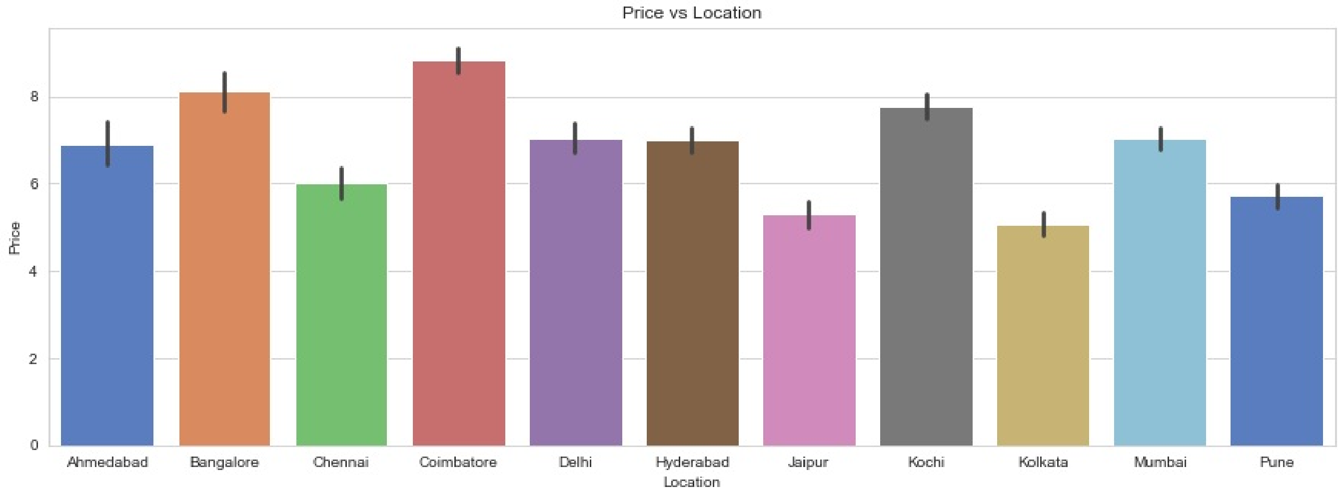
Observation:

1. Power and price do correlate well.
2. Positive correlation shown by the two relationships.
3. There isn't a steady positive trend, rather many up and down's between the two variables.

Numerical Vs Categorical

In [150…

```
plt.figure(figsize=(15,5))  # setting the figure size
plt.title('Price vs Location')
ax = sns.barplot(x='Location', y='Price', data=data, palette='muted')#barplot of location vs price
```

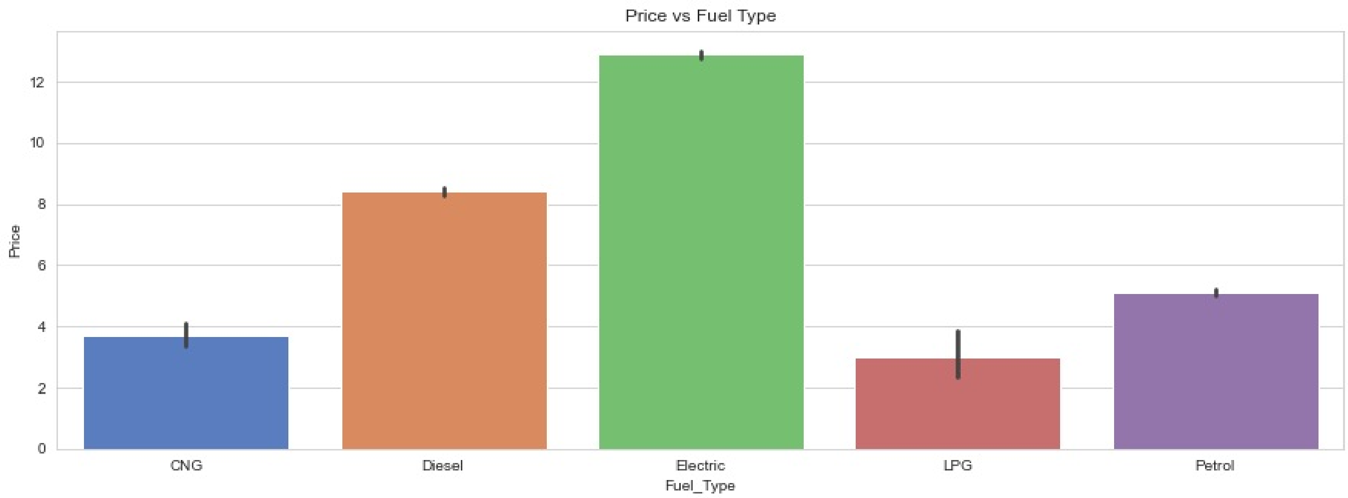


Observations:

1. Coimbatore has the highest price amongst other locations followed by Bangalore and Kochi.
2. Coimbatore and Kochi are in Kerala, which suggest that Kerala has the highest priced cars.
3. Lowest priced cars can be found in Jaipur and Kolkata and Chennai.

In [151…

```
plt.figure(figsize=(15,5))  # setting the figure size
plt.title('Price vs Fuel Type')
ax = sns.barplot(x='Fuel_Type', y='Price', data=data, palette='muted')
```

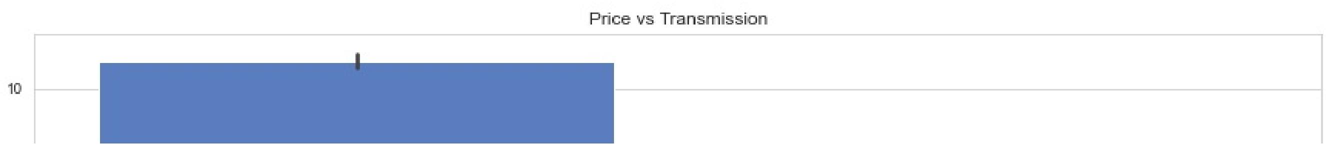


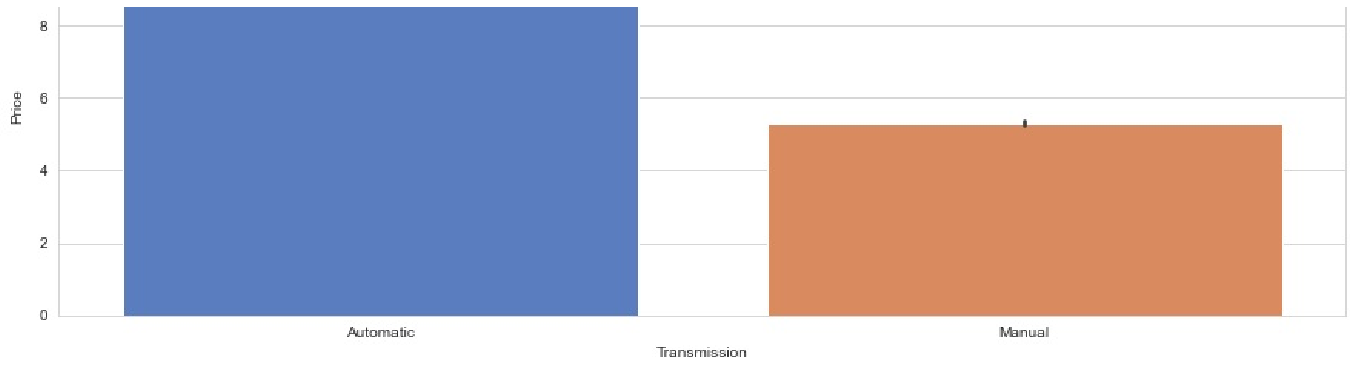
Observations:

1. Electric type cars are highly priced followed by DIelse and Petrol cars.
2. LPG cars are the lowest priced cars.

In [152…

```
plt.figure(figsize=(15,5))  # setting the figure size
plt.title('Price vs Transmission')
ax = sns.barplot(x='Transmission', y='Price', data=data, palette='muted')
```



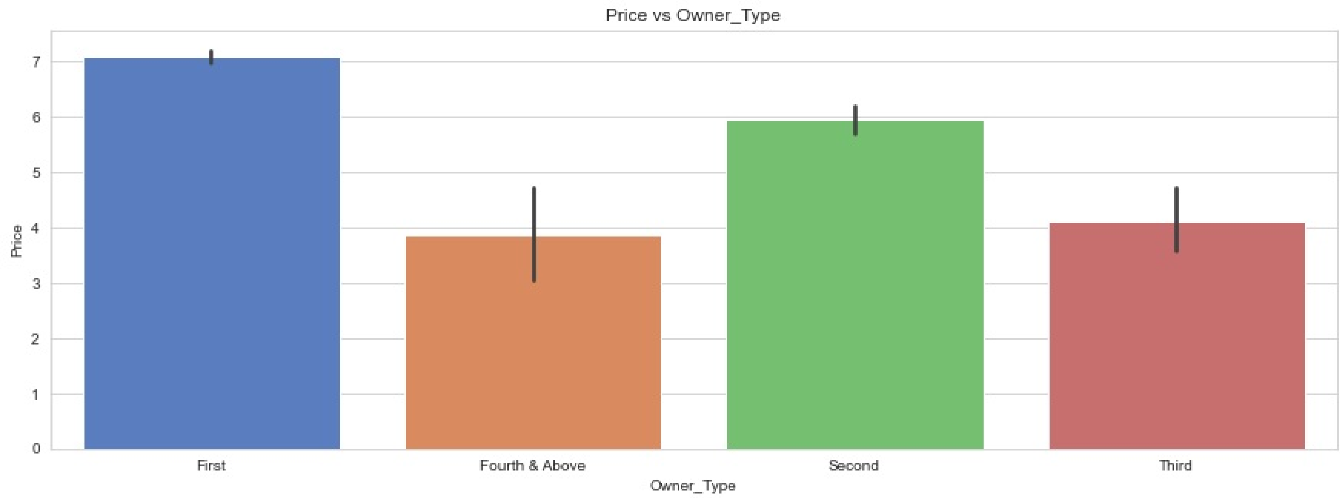


Observations:

1. Automatic cars are highest priced cars
2. Manual cars are the least priced cars.

In [153…

```
plt.figure(figsize=(15,5))  # setting the figure size
plt.title('Price vs Owner_Type')
ax = sns.barplot(x='Owner_Type', y='Price', data=data, palette='muted')
```

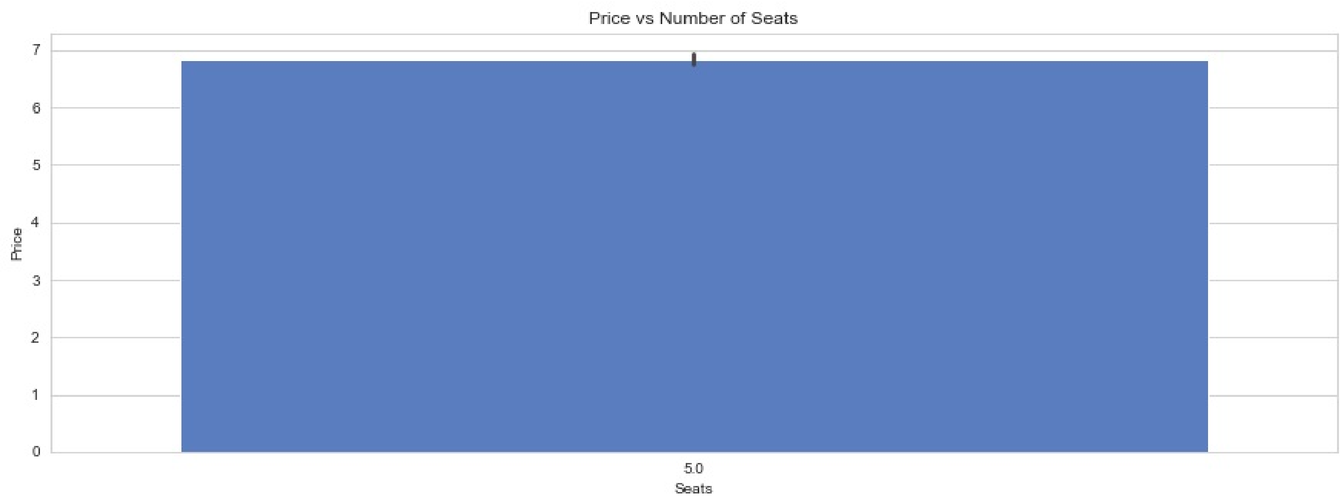


Observations:

1. First hand cars are highest priced, followed by second and third.
2. This confirms and makes sense, because pre-owned cars will have lower prices than fresh cars.

In [154…

```
plt.figure(figsize=(15,5))  # setting the figure size
plt.title('Price vs Number of Seats')
ax = sns.barplot(x='Seats', y='Price', data=data, palette='muted')
```



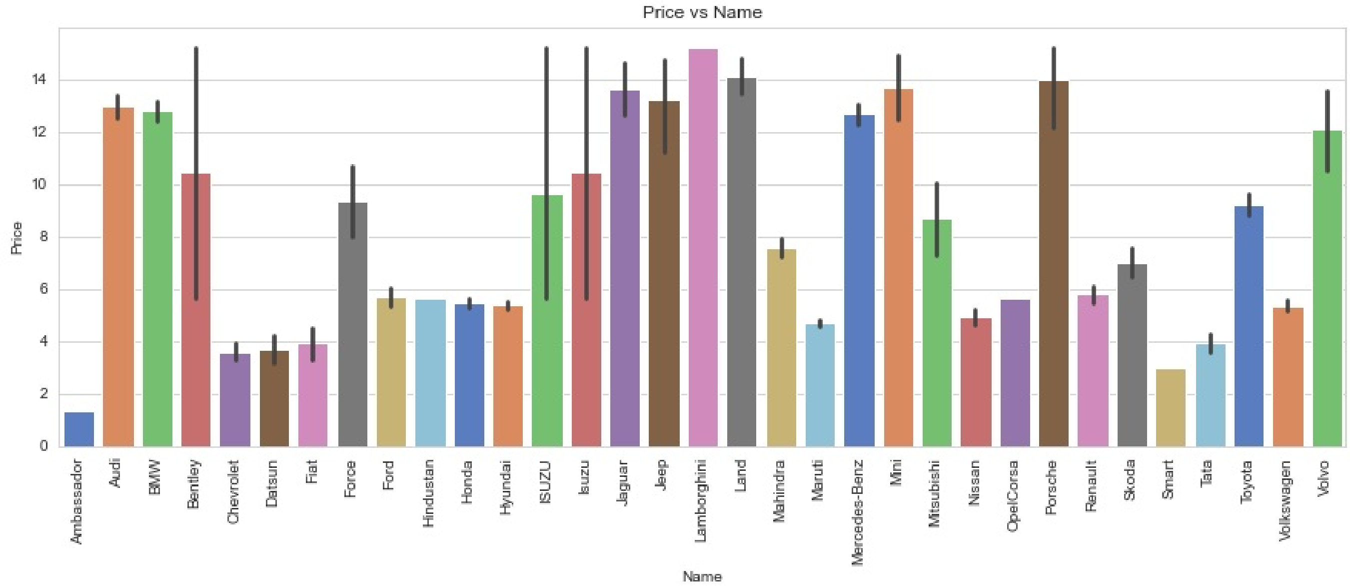
Observations:

1. Cars has only 5 seats as rest of the rows got filled with median values.

2. No remaining outliers

In [155…

```python
plt.figure(figsize=(15,5))  # setting the figure size
plt.title('Price vs Name')
plt.xticks(rotation=90)
ax = sns.barplot(x='Name', y='Price', data=data, palette='muted')
```



Observations:

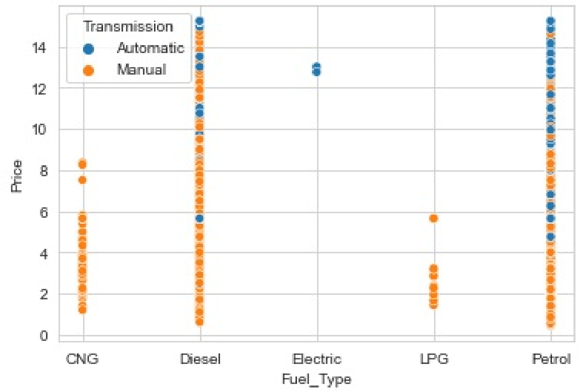
1. As expected Lamborghini has the highest price out of any cars, followed by Porsche, Bentley and Jaguar.
2. Least priced cars are the eco cars such as Maruti, Honda, Nissan, Hindustan, Ambassador, Smart etc.
3. The name gives straight forward insights, but I don't think it is necesscary to keep names column, as it won't provide any deep insights.

Two Categorical vs Numerical

In [156…

```python
sns.scatterplot(y = 'Price', x = 'Fuel_Type', data = data, hue = 'Transmission')
```

Out[156… <AxesSubplot:xlabel='Fuel_Type', ylabel='Price'>



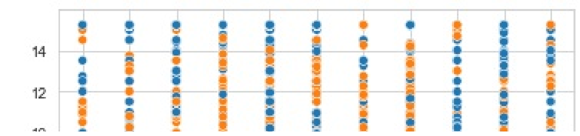
Observations:

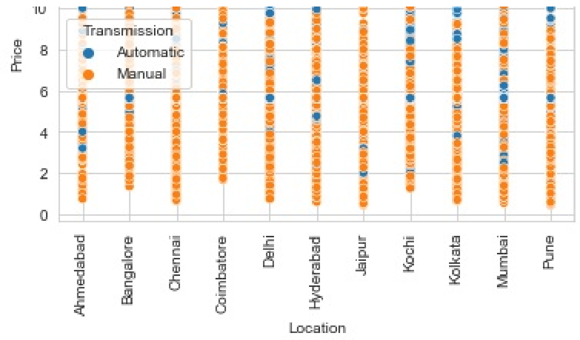
1. Most CNG and LPG cas are manual cars as they are very old cars.
2. Diesel and Petrol cars mostly comprise of Automatic cars rather than manual cars.

In [157…

```python
plt.xticks(rotation=90)
sns.scatterplot(y = 'Price', x = 'Location', data = data, hue = 'Transmission')
plt.figure(figsize = (15, 10))
```

Out[157… <Figure size 1080x720 with 0 Axes>






```
<Figure size 1080x720 with 0 Axes>
```

Observations:

1. Automatic cars are common accross every region
2. On average automatic cars are highly priced than Mnaual cars for every single region.

In [170…
```
data.to_csv('data1.csv') #converting the current data set into a new dataframe
```

Data Preparation for Modeling

In [171…
```
df = pd.read_csv('data1.csv')
df.head()
```

Out[171…

	Unnamed: 0	Name	Location	Age_of_Car	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	Pric
0	0	Maruti	Mumbai	10.0	72000	CNG	Manual	First	26.60	998.0	58.16	5.0	1.7
1	1	Hyundai	Pune	5.0	41000	Diesel	Manual	First	19.67	1582.0	126.20	5.0	12.5
2	2	Honda	Chennai	9.0	46000	Petrol	Manual	First	18.20	1199.0	88.70	5.0	4.5
3	3	Maruti	Chennai	8.0	87000	Diesel	Manual	First	20.77	1248.0	88.76	5.0	6.0
4	4	Audi	Coimbatore	7.0	40670	Diesel	Automatic	Second	15.20	1968.0	140.80	5.0	17.7

In [172…
```
X = df.drop(["Price", "Seats"], axis=1) #dropping Seats variable as there was not much correlation between Price
y = df[["Price"]]

print(X.head())
print(y.head())
```

```
   Unnamed: 0     Name    Location  Age_of_Car  Kilometers_Driven Fuel_Type  \
0           0   Maruti      Mumbai        10.0              72000       CNG   
1           1  Hyundai        Pune         5.0              41000    Diesel   
2           2    Honda     Chennai         9.0              46000    Petrol   
3           3   Maruti     Chennai         8.0              87000    Diesel   
4           4     Audi  Coimbatore         7.0              40670    Diesel   

  Transmission Owner_Type  Mileage  Engine   Power  
0       Manual      First    26.60   998.0   58.16  
1       Manual      First    19.67  1582.0  126.20  
2       Manual      First    18.20  1199.0   88.70  
3       Manual      First    20.77  1248.0   88.76  
4    Automatic     Second    15.20  1968.0  140.80  
   Price
0   1.75
1  12.50
2   4.50
3   6.00
4  17.74
```

In [173…
```
print(X.shape)
print(y.shape)
```

```
(7250, 11)
(7250, 1)
```

In [174…
```
# creating dummy variables
X = pd.get_dummies(X, columns=["Name","Location","Fuel_Type", "Transmission", "Owner_Type"], drop_first=True)
```

```
X.head()
```

Out[174…

	Unnamed: 0	Age_of_Car	Kilometers_Driven	Mileage	Engine	Power	Name_Audi	Name_BMW	Name_Bentley	Name_Chevrolet	...	Location_I
0	0	10.0	72000	26.60	998.0	58.16	0	0	0	0	...	
1	1	5.0	41000	19.67	1582.0	126.20	0	0	0	0	...	
2	2	9.0	46000	18.20	1199.0	88.70	0	0	0	0	...	
3	3	8.0	87000	20.77	1248.0	88.76	0	0	0	0	...	
4	4	7.0	40670	15.20	1968.0	140.80	1	0	0	0	...	

5 rows × 56 columns

In [175…

```
# split the data into train and test
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42) #splitting data into te
```

In [176…

```
X_train.head()
```

Out[176…

	Unnamed: 0	Age_of_Car	Kilometers_Driven	Mileage	Engine	Power	Name_Audi	Name_BMW	Name_Bentley	Name_Chevrolet	...	Lo
952	952	7.0	71000	17.000	1197.0	80.000000	0	0	0	0	...	
2264	2264	16.0	131500	6.275	2446.0	112.765214	0	0	0	0	...	
1504	1504	12.0	95000	12.900	1799.0	130.000000	0	0	0	0	...	
7210	7210	7.0	131500	22.300	1248.0	74.000000	0	0	0	0	...	
5538	5538	8.0	81000	25.440	936.0	56.300000	0	0	0	1	...	

5 rows × 56 columns

Choose, Train, Evaluate model

In [177…

```
# fitting the model on the train data (70% of the whole data)
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

linearregression = LinearRegression()
linearregression.fit(X_train, y_train)
```

Out[177… LinearRegression()

In [178…

```
# let us check the coefficients and intercept of the model

coef_df = pd.DataFrame(
    np.append(linearregression.coef_[0], linearregression.intercept_[0]),
    index=X_train.columns.tolist() + ["Intercept"],
    columns=["Coefficients"],
)

coef_df
```

Out[178…

	Coefficients
Unnamed: 0	2.532329e-04
Age_of_Car	-4.608980e-01
Kilometers_Driven	-9.727808e-06
Mileage	-9.918001e-02
Engine	9.691858e-04
Power	2.837856e-02
Name_Audi	2.872767e+00
Name_BMW	2.483456e+00
Name_Bentley	1.571339e+00
Name_Chevrolet	-1.842591e+00
Name_Datsun	-2.729279e+00
Name_Fiat	-1.396602e+00
Name_Force	-1.229473e+00

Name_Ford	-1.531774e+00
Name_Hindustan	2.930989e-14
Name_Honda	-1.390413e+00
Name_Hyundai	-1.216315e+00
Name_ISUZU	-4.441976e+00
Name_Isuzu	-4.871319e-01
Name_Jaguar	2.471685e+00
Name_Jeep	2.238565e+00
Name_Lamborghini	4.806032e+00
Name_Land	3.884227e+00
Name_Mahindra	-1.718561e+00
Name_Maruti	-9.543948e-01
Name_Mercedes-Benz	2.215802e+00
Name_Mini	5.773696e+00
Name_Mitsubishi	-4.531860e-01
Name_Nissan	-1.478800e+00
Name_OpelCorsa	5.160197e+00
Name_Porsche	2.571528e+00
Name_Renault	-1.488897e+00
Name_Skoda	-1.357686e+00
Name_Smart	-3.191215e+00
Name_Tata	-2.270336e+00
Name_Toyota	6.735057e-01
Name_Volkswagen	-1.715829e+00
Name_Volvo	1.623700e+00
Location_Bangalore	2.427726e-01
Location_Chennai	-7.186991e-02
Location_Coimbatore	1.925768e-01
Location_Delhi	-8.448755e-01
Location_Hyderabad	2.308306e-01
Location_Jaipur	-9.290503e-02
Location_Kochi	-3.416551e-01
Location_Kolkata	-1.026230e+00
Location_Mumbai	-3.248464e-01
Location_Pune	-1.797736e-01
Fuel_Type_Diesel	1.411251e+00
Fuel_Type_Electric	6.989043e+00
Fuel_Type_LPG	7.493960e-01
Fuel_Type_Petrol	1.271854e-01
Transmission_Manual	-7.722681e-01
Owner_Type_Fourth & Above	5.905974e-02
Owner_Type_Second	-2.429078e-01
Owner_Type_Third	-4.758598e-01
Intercept	8.152298e+00

Let's check the performance of the model using different metrics (MAE, MAPE, RMSE, R2).

We will be using metric functions defined in sklearn for RMSE, MAE, and R2. We will define a function to calculate MAPE. We will create a function which will print out all the above metrics in one go.

In [179…

```
# defining function for MAPE
def mape(targets, predictions):
    return np.mean(np.abs((targets - predictions)) / targets) * 100


# defining common function for all metrics
def model_perf(model, inp, out):
```

```
    """
    model: model
    inp: independent variables
    out: dependent variable
    """
    y_pred = model.predict(inp).flatten()
    y_act = out.values.flatten()

    return pd.DataFrame(
        {
            "MAE": mean_absolute_error(y_act, y_pred),
            "MAPE": mape(y_act, y_pred),
            "RMSE": np.sqrt(mean_squared_error(y_act, y_pred)),
            "R^2": r2_score(y_act, y_pred),
        },
        index=[0],
    )
```

In [180…

```
# Checking model performance on train set (seen 70% data)
print("Train Performance\n")
model_perf(linearregression, X_train, y_train)
```

```
Train Performance
```

Out[180…

	MAE	MAPE	RMSE	R^2
0	1.88127	31.671087	2.513503	0.737865

In [181…

```
# Checking model performance on test set (unseen 30% data)
print("Test Performance\n")
model_perf(linearregression, X_test, y_test)
```

```
Test Performance
```

Out[181…

	MAE	MAPE	RMSE	R^2
0	1.905165	32.523649	2.514116	0.741246

Observations:

1. The training data performance is 72% while the testing data performance is 74% both which are comparable which is good.
2. R-squared is 0.740 on the test set, i.e., the model explains 74% of total variation in the test dataset. So, overall the model is very satisfactory.
3. MAE indicates that our current model is able to predict Price within a mean error of 1.95 years on the test data.
4. MAPE on the test set suggests we can predict within 33% of Price.

Linear regression using statsmodel

In [182…

```
# Let's build linear regression model using statsmodel
# unlike sklearn, statsmodels does not add a constant to the data on its own
# we have to add the constant manually
import statsmodels.api as sm
X = sm.add_constant(X)
X_train1, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

olsmod0 = sm.OLS(y_train, X_train1)
olsres0 = olsmod0.fit()
print(olsres0.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.738
Model:                            OLS   Adj. R-squared:                  0.735
Method:                 Least Squares   F-statistic:                     256.9
Date:                Fri, 21 May 2021   Prob (F-statistic):               0.00
Time:                        14:14:51   Log-Likelihood:                -11879.
No. Observations:                5075   AIC:                         2.387e+04
Df Residuals:                    5019   BIC:                         2.424e+04
Df Model:                          55
Covariance Type:            nonrobust
===============================================================================================
                                coef    std err          t      P>|t|      [0.025      0.975]
-----------------------------------------------------------------------------------------------
```

```
const                        8.1523      2.621      3.110      0.002       3.013      13.292
Unnamed: 0                   0.0003   1.71e-05     14.816      0.000       0.000       0.000
Age_of_Car                  -0.4609      0.016    -28.320      0.000      -0.493      -0.429
Kilometers_Driven        -9.728e-06   1.64e-06     -5.936      0.000   -1.29e-05   -6.52e-06
Mileage                     -0.0992      0.015     -6.429      0.000      -0.129      -0.069
Engine                       0.0010      0.000      5.296      0.000       0.001       0.001
Power                        0.0284      0.002     13.239      0.000       0.024       0.033
Name_Audi                    2.8728      2.563      1.121      0.262      -2.152       7.898
Name_BMW                     2.4835      2.566      0.968      0.333      -2.546       7.513
Name_Bentley                 1.5713      3.131      0.502      0.616      -4.566       7.709
Name_Chevrolet              -1.8426      2.561     -0.720      0.472      -6.862       3.177
Name_Datsun                 -2.7293      2.655     -1.028      0.304      -7.934       2.476
Name_Fiat                   -1.3966      2.596     -0.538      0.591      -6.485       3.692
Name_Force                  -1.2295      2.940     -0.418      0.676      -6.993       4.534
Name_Ford                   -1.5318      2.554     -0.600      0.549      -6.538       3.475
Name_Hindustan            6.613e-15   2.04e-14      0.324      0.746   -3.34e-14    4.66e-14
Name_Honda                  -1.3904      2.555     -0.544      0.586      -6.399       3.618
Name_Hyundai                -1.2163      2.552     -0.477      0.634      -6.220       3.787
Name_ISUZU                  -4.4420      3.122     -1.423      0.155     -10.562       1.678
Name_Isuzu                  -0.4871      3.121     -0.156      0.876      -6.606       5.632
Name_Jaguar                  2.4717      2.598      0.951      0.342      -2.622       7.566
Name_Jeep                    2.2386      2.640      0.848      0.396      -2.937       7.414
Name_Lamborghini             4.8060      3.594      1.337      0.181      -2.240      11.852
Name_Land                    3.8842      2.582      1.504      0.133      -1.177       8.946
Name_Mahindra               -1.7186      2.554     -0.673      0.501      -6.726       3.289
Name_Maruti                 -0.9544      2.552     -0.374      0.708      -5.958       4.049
Name_Mercedes-Benz           2.2158      2.561      0.865      0.387      -2.805       7.237
Name_Mini                    5.7737      2.610      2.212      0.027       0.657      10.891
Name_Mitsubishi             -0.4532      2.611     -0.174      0.862      -5.572       4.666
Name_Nissan                 -1.4788      2.565     -0.577      0.564      -6.507       3.549
Name_OpelCorsa               5.1602      3.594      1.436      0.151      -1.886      12.206
Name_Porsche                 2.5715      2.643      0.973      0.331      -2.610       7.753
Name_Renault                -1.4889      2.561     -0.581      0.561      -6.509       3.531
Name_Skoda                  -1.3577      2.561     -0.530      0.596      -6.378       3.663
Name_Smart                  -3.1912      3.603     -0.886      0.376     -10.255       3.873
Name_Tata                   -2.2703      2.558     -0.888      0.375      -7.285       2.744
Name_Toyota                  0.6735      2.554      0.264      0.792      -4.334       5.681
Name_Volkswagen             -1.7158      2.554     -0.672      0.502      -6.723       3.292
Name_Volvo                   1.6237      2.612      0.622      0.534      -3.498       6.745
Location_Bangalore           0.2428      0.228      1.067      0.286      -0.203       0.689
Location_Chennai            -0.0719      0.218     -0.329      0.742      -0.500       0.356
Location_Coimbatore          0.1926      0.209      0.920      0.357      -0.218       0.603
Location_Delhi              -0.8449      0.213     -3.959      0.000      -1.263      -0.427
Location_Hyderabad           0.2308      0.205      1.125      0.261      -0.171       0.633
Location_Jaipur             -0.0929      0.226     -0.412      0.680      -0.535       0.349
Location_Kochi              -0.3417      0.210     -1.630      0.103      -0.753       0.069
Location_Kolkata            -1.0262      0.215     -4.783      0.000      -1.447      -0.606
Location_Mumbai             -0.3248      0.203     -1.597      0.110      -0.724       0.074
Location_Pune               -0.1798      0.209     -0.860      0.390      -0.589       0.230
Fuel_Type_Diesel             1.4113      0.394      3.580      0.000       0.638       2.184
Fuel_Type_Electric           6.9890      1.844      3.790      0.000       3.374      10.604
Fuel_Type_LPG                0.7494      0.896      0.837      0.403      -1.007       2.505
Fuel_Type_Petrol             0.1272      0.400      0.318      0.750      -0.657       0.911
Transmission_Manual         -0.7723      0.121     -6.404      0.000      -1.009      -0.536
Owner_Type_Fourth & Above    0.0591      0.851      0.069      0.945      -1.609       1.727
Owner_Type_Second           -0.2429      0.105     -2.322      0.020      -0.448      -0.038
Owner_Type_Third            -0.4759      0.281     -1.696      0.090      -1.026       0.074
==============================================================================
Omnibus:                      203.075   Durbin-Watson:                   2.013
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              632.573
Skew:                           0.049   Prob(JB):                    4.35e-138
Kurtosis:                       4.727   Cond. No.                     1.14e+16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.6e-19. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

Observations:

1. Negative values of the coefficient show that Price decreases with the increase of corresponding attribute value.
2. Positive values of the coefficient show that Price increases with the increase of corresponding attribute value.\
3. But these variables might contain multicollinearity, which will affect the p-values.So, we need to deal with multicollinearity and check the other assumptions of linear regression first, and then look at the p-values.

Checking Linear Regression Assumptions:

1. No Multicollinearity
2. Mean of residuals should be 0
3. No Heteroscedasticity
4. Linearity of variables
5. Normality of error terms

Test for Multi-colinerarity

In [183…

```
from statsmodels.stats.outliers_influence import variance_inflation_factor

vif_series1 = pd.Series(
    [variance_inflation_factor(X.values, i) for i in range(X.shape[1])], index=X.columns
)
print("VIF Scores: \n\n{}\n".format(vif_series1))
```

```
VIF Scores:

const                        7629.495563
Unnamed: 0                      1.006295
Age_of_Car                      2.161840
Kilometers_Driven               1.934936
Mileage                         3.489832
Engine                          8.339316
Power                           7.722786
Name_Audi                     279.130380
Name_BMW                      303.807552
Name_Bentley                    3.045368
Name_Chevrolet                150.502278
Name_Datsun                    18.187907
Name_Fiat                      39.238450
Name_Force                      4.038609
Name_Ford                     338.720203
Name_Hindustan                  2.014152
Name_Honda                    676.754533
Name_Hyundai                 1107.016577
Name_ISUZU                      4.043979
Name_Isuzu                      3.032690
Name_Jaguar                    49.525590
Name_Jeep                      20.262048
Name_Lamborghini                2.014608
Name_Land                      68.353416
Name_Mahindra                 320.478849
Name_Maruti                  1171.137374
Name_Mercedes-Benz            366.713732
Name_Mini                      32.339617
Name_Mitsubishi                37.230289
Name_Nissan                   117.467956
Name_OpelCorsa                  2.014707
Name_Porsche                   20.296725
Name_Renault                  168.120437
Name_Skoda                    199.970070
Name_Smart                      2.021905
Name_Tata                     224.434773
Name_Toyota                   478.338904
Name_Volkswagen               359.866054
Name_Volvo                     29.353022
Location_Bangalore              2.482514
Location_Chennai                2.965172
Location_Coimbatore             3.502153
Location_Delhi                  3.115341
Location_Hyderabad              3.734260
Location_Jaipur                 2.674546
Location_Kochi                  3.494423
Location_Kolkata                3.134109
Location_Mumbai                 3.947901
Location_Pune                   3.460740
Fuel_Type_Diesel               31.125435
Fuel_Type_Electric              1.045131
Fuel_Type_LPG                   1.211395
Fuel_Type_Petrol               31.636694
Transmission_Manual             2.310319
Owner_Type_Fourth & Above       1.015855
Owner_Type_Second               1.180549
Owner_Type_Third                1.118845
dtype: float64
```

Observations:

1. Some of the Name columns have high VIF values, values greater than 10.
2. Fuel_type_Diesel and Petrol have high VIF values (31).
3. Name_Maruti has the highest VIF value of 1170.

In [184…

```
# we drop the one with the highest vif values and check the adjusted R-squared
X_train2 = X_train1.drop("Name_Maruti", axis=1)
vif_series2 = pd.Series(
    [variance_inflation_factor(X_train2.values, i) for i in range(X_train2.shape[1])],
    index=X_train2.columns,
)
print("VIF Scores: \n\n{}\n".format(vif_series2))
```

```
VIF Scores: 

const                        375.615738
Unnamed: 0                     1.009592
Age_of_Car                     2.173371
Kilometers_Driven              1.937204
Mileage                        3.496700
Engine                         8.280492
Power                          7.548324
Name_Audi                      2.020632
Name_BMW                       2.253541
Name_Bentley                   1.027349
Name_Chevrolet                 1.120779
Name_Datsun                    1.018953
Name_Fiat                      1.036420
Name_Force                     1.013269
Name_Ford                      1.288179
Name_Hindustan                      NaN
Name_Honda                     1.587932
Name_Hyundai                   1.741116
Name_ISUZU                     1.018323
Name_Isuzu                     1.017891
Name_Jaguar                    1.237926
Name_Jeep                      1.075760
Name_Lamborghini               1.025266
Name_Land                      1.266129
Name_Mahindra                  1.703093
Name_Mercedes-Benz             2.368496
Name_Mini                      1.083688
Name_Mitsubishi                1.092721
Name_Nissan                    1.092437
Name_OpelCorsa                 1.006322
Name_Porsche                   1.132191
Name_Renault                   1.136538
Name_Skoda                     1.283632
Name_Smart                     1.020896
Name_Tata                      1.157518
Name_Toyota                    2.097303
Name_Volkswagen                1.283635
Name_Volvo                     1.133847
Location_Bangalore             2.431605
Location_Chennai               2.801673
Location_Coimbatore            3.360029
Location_Delhi                 2.905687
Location_Hyderabad             3.503152
Location_Jaipur                2.540102
Location_Kochi                 3.339118
Location_Kolkata               2.979559
Location_Mumbai                3.715593
Location_Pune                  3.303077
Fuel_Type_Diesel              30.734370
Fuel_Type_Electric             1.064201
Fuel_Type_LPG                  1.253517
Fuel_Type_Petrol              31.499005
Transmission_Manual            2.335851
Owner_Type_Fourth & Above      1.017545
Owner_Type_Second              1.179906
Owner_Type_Third               1.115901
dtype: float64
```

Observation:

1. Dropping the Name_Maruthi has seemed to help the VIF values.

2. Only Fuel_type Diesel/Petrol seem to have high VIF value.

In [185…

```
olsmod1 = sm.OLS(y_train, X_train2)
olsres1 = olsmod1.fit()
print(olsres1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.738
Model:                            OLS   Adj. R-squared:                  0.735
Method:                 Least Squares   F-statistic:                     261.7
Date:                Fri, 21 May 2021   Prob (F-statistic):               0.00
Time:                        14:14:54   Log-Likelihood:                -11879.
No. Observations:                5075   AIC:                         2.387e+04
Df Residuals:                    5020   BIC:                         2.423e+04
Df Model:                          54
Covariance Type:            nonrobust
==============================================================================================
                               coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------------
const                        7.2063      0.688     10.481      0.000       5.858       8.554
Unnamed: 0                   0.0003   1.71e-05     14.814      0.000       0.000       0.000
Age_of_Car                  -0.4608      0.016    -28.320      0.000      -0.493      -0.429
Kilometers_Driven        -9.745e-06   1.64e-06     -5.950      0.000    -1.3e-05   -6.53e-06
Mileage                     -0.0994      0.015     -6.451      0.000      -0.130      -0.069
Engine                       0.0010      0.000      5.301      0.000       0.001       0.001
Power                        0.0283      0.002     13.237      0.000       0.024       0.033
Name_Audi                    3.8264      0.258     14.834      0.000       3.321       4.332
Name_BMW                     3.4378      0.264     13.031      0.000       2.921       3.955
Name_Bentley                 2.5260      1.812      1.394      0.163      -1.026       6.078
Name_Chevrolet              -0.8902      0.265     -3.358      0.001      -1.410      -0.371
Name_Datsun                 -1.7755      0.737     -2.408      0.016      -3.221      -0.330
Name_Fiat                   -0.4439      0.496     -0.894      0.371      -1.417       0.529
Name_Force                  -0.2772      1.469     -0.189      0.850      -3.157       2.603
Name_Ford                   -0.5795      0.191     -3.037      0.002      -0.954      -0.205
Name_Hindustan           -6.803e-15   5.05e-15     -1.347      0.178   -1.67e-14     3.1e-15
Name_Honda                  -0.4366      0.146     -2.994      0.003      -0.722      -0.151
Name_Hyundai                -0.2629      0.121     -2.176      0.030      -0.500      -0.026
Name_ISUZU                  -3.4892      1.804     -1.934      0.053      -7.025       0.047
Name_Isuzu                   0.4655      1.803      0.258      0.796      -3.070       4.001
Name_Jaguar                  3.4259      0.491      6.976      0.000       2.463       4.389
Name_Jeep                    3.1927      0.678      4.710      0.000       1.864       4.521
Name_Lamborghini             5.7497      2.559      2.247      0.025       0.732      10.767
Name_Land                    4.8368      0.421     11.483      0.000       4.011       5.663
Name_Mahindra               -0.7669      0.219     -3.502      0.000      -1.196      -0.338
Name_Mercedes-Benz           3.1694      0.240     13.209      0.000       2.699       3.640
Name_Mini                    6.7279      0.550     12.236      0.000       5.650       7.806
Name_Mitsubishi              0.4965      0.607      0.818      0.414      -0.694       1.687
Name_Nissan                 -0.5261      0.296     -1.778      0.075      -1.106       0.054
Name_OpelCorsa               6.1127      2.535      2.411      0.016       1.142      11.083
Name_Porsche                 3.5250      0.695      5.069      0.000       2.162       4.888
Name_Renault                -0.5356      0.241     -2.220      0.026      -1.008      -0.063
Name_Skoda                  -0.4043      0.241     -1.676      0.094      -0.877       0.069
Name_Smart                  -2.2407      2.554     -0.877      0.380      -7.247       2.766
Name_Tata                   -1.3174      0.221     -5.958      0.000      -1.751      -0.884
Name_Toyota                  1.6257      0.202      8.029      0.000       1.229       2.023
Name_Volkswagen             -0.7631      0.183     -4.170      0.000      -1.122      -0.404
Name_Volvo                   2.5778      0.562      4.583      0.000       1.475       3.680
Location_Bangalore           0.2426      0.228      1.066      0.286      -0.204       0.689
Location_Chennai            -0.0699      0.218     -0.320      0.749      -0.498       0.358
Location_Coimbatore          0.1930      0.209      0.923      0.356      -0.217       0.603
Location_Delhi              -0.8447      0.213     -3.959      0.000      -1.263      -0.426
Location_Hyderabad           0.2308      0.205      1.125      0.261      -0.171       0.633
Location_Jaipur             -0.0933      0.226     -0.414      0.679      -0.535       0.349
Location_Kochi              -0.3412      0.210     -1.628      0.104      -0.752       0.070
Location_Kolkata            -1.0263      0.215     -4.783      0.000      -1.447      -0.606
Location_Mumbai             -0.3248      0.203     -1.596      0.110      -0.724       0.074
Location_Pune               -0.1801      0.209     -0.862      0.389      -0.590       0.229
Fuel_Type_Diesel             1.4126      0.394      3.584      0.000       0.640       2.185
Fuel_Type_Electric           6.9883      1.844      3.790      0.000       3.374      10.603
Fuel_Type_LPG                0.7481      0.896      0.835      0.404      -1.008       2.504
Fuel_Type_Petrol             0.1264      0.400      0.316      0.752      -0.657       0.910
Transmission_Manual         -0.7725      0.121     -6.407      0.000      -1.009      -0.536
Owner_Type_Fourth & Above    0.0575      0.851      0.068      0.946      -1.610       1.725
Owner_Type_Second           -0.2431      0.105     -2.324      0.020      -0.448      -0.038
Owner_Type_Third            -0.4661      0.279     -1.668      0.095      -1.014       0.082
==============================================================================
Omnibus:                      202.887   Durbin-Watson:                   2.014
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              631.576
Skew:                           0.049   Prob(JB):                    7.16e-138
Kurtosis:                       4.725   Cond. No.                     1.14e+16
==============================================================================
```

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.6e-19. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

Observation:

1. Dropping the Maruti value has increased the Adjusted R-squared value.

In [186…

```python
# we drop the one with the highest vif values and check the adjusted R-squared
X_train3 = X_train2.drop("Fuel_Type_Diesel", axis=1)
vif_series2 = pd.Series(
    [variance_inflation_factor(X_train3.values, i) for i in range(X_train3.shape[1])],
    index=X_train3.columns,
)
print("VIF Scores: \n\n{}\n".format(vif_series2))
```

```
VIF Scores: 

const                          263.935629
Unnamed: 0                       1.009571
Age_of_Car                       2.171764
Kilometers_Driven                1.936175
Mileage                          3.493694
Engine                           8.276285
Power                            7.546950
Name_Audi                        2.016326
Name_BMW                         2.248770
Name_Bentley                     1.027338
Name_Chevrolet                   1.117814
Name_Datsun                      1.018793
Name_Fiat                        1.035719
Name_Force                       1.013213
Name_Ford                        1.280584
Name_Hindustan                        NaN
Name_Honda                       1.579944
Name_Hyundai                     1.728310
Name_ISUZU                       1.018321
Name_Isuzu                       1.017890
Name_Jaguar                      1.237481
Name_Jeep                        1.075376
Name_Lamborghini                 1.025265
Name_Land                        1.265255
Name_Mahindra                    1.696692
Name_Mercedes-Benz               2.364810
Name_Mini                        1.083105
Name_Mitsubishi                  1.092432
Name_Nissan                      1.089720
Name_OpelCorsa                   1.006312
Name_Porsche                     1.132113
Name_Renault                     1.130976
Name_Skoda                       1.280291
Name_Smart                       1.020838
Name_Tata                        1.154024
Name_Toyota                      2.093541
Name_Volkswagen                  1.276603
Name_Volvo                       1.133354
Location_Bangalore               2.431211
Location_Chennai                 2.801063
Location_Coimbatore              3.359519
Location_Delhi                   2.905019
Location_Hyderabad               3.501627
Location_Jaipur                  2.537576
Location_Kochi                   3.338707
Location_Kolkata                 2.978139
Location_Mumbai                  3.713202
Location_Pune                    3.303057
Fuel_Type_Electric               1.018817
Fuel_Type_LPG                    1.027701
Fuel_Type_Petrol                 2.688654
Transmission_Manual              2.335844
Owner_Type_Fourth & Above        1.017527
Owner_Type_Second                1.179627
Owner_Type_Third                 1.115711
dtype: float64
```

Observation:

1. Dropping the Diesel value has brought all VIF values significantly down.
2. Engine and Power have slighty high VIF value of 7 and 8 respectively.

In [187…

```
olsmod2 = sm.OLS(y_train, X_train3)
olsres2 = olsmod2.fit()
print(olsres2.summary())
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.737
Model:                            OLS   Adj. R-squared:                  0.734
Method:                 Least Squares   F-statistic:                     265.7
Date:                Fri, 21 May 2021   Prob (F-statistic):               0.00
Time:                        14:14:55   Log-Likelihood:                -11885.
No. Observations:                5075   AIC:                         2.388e+04
Df Residuals:                    5021   BIC:                         2.423e+04
Df Model:                          53                                         
Covariance Type:            nonrobust                                         
==============================================================================================
                                coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------------
const                         8.5498      0.577     14.817      0.000       7.419       9.681
Unnamed: 0                    0.0003   1.71e-05     14.780      0.000       0.000       0.000
Age_of_Car                   -0.4624      0.016    -28.395      0.000      -0.494      -0.430
Kilometers_Driven          -9.61e-06   1.64e-06     -5.862      0.000   -1.28e-05    -6.4e-06
Mileage                      -0.1010      0.015     -6.551      0.000      -0.131      -0.071
Engine                        0.0010      0.000      5.376      0.000       0.001       0.001
Power                         0.0284      0.002     13.271      0.000       0.024       0.033
Name_Audi                     3.8691      0.258     14.997      0.000       3.363       4.375
Name_BMW                      3.4813      0.264     13.194      0.000       2.964       3.999
Name_Bentley                  2.5047      1.814      1.381      0.167      -1.051       6.061
Name_Chevrolet               -0.8413      0.265     -3.174      0.002      -1.361      -0.322
Name_Datsun                  -1.7423      0.738     -2.360      0.018      -3.189      -0.295
Name_Fiat                    -0.3976      0.497     -0.800      0.424      -1.372       0.577
Name_Force                   -0.2380      1.471     -0.162      0.871      -3.122       2.646
Name_Ford                    -0.5270      0.190     -2.766      0.006      -0.900      -0.154
Name_Hindustan            -8.057e-15   6.61e-15     -1.218      0.223     -2.1e-14    4.91e-15
Name_Honda                   -0.3995      0.146     -2.744      0.006      -0.685      -0.114
Name_Hyundai                 -0.2258      0.120     -1.874      0.061      -0.462       0.010
Name_ISUZU                   -3.4811      1.806     -1.928      0.054      -7.021       0.059
Name_Isuzu                    0.4738      1.805      0.262      0.793      -3.066       4.013
Name_Jaguar                   3.4592      0.492      7.037      0.000       2.496       4.423
Name_Jeep                     3.2386      0.678      4.773      0.000       1.908       4.569
Name_Lamborghini              5.7563      2.562      2.247      0.025       0.733      10.779
Name_Land                     4.8765      0.422     11.568      0.000       4.050       5.703
Name_Mahindra                -0.7188      0.219     -3.285      0.001      -1.148      -0.290
Name_Mercedes-Benz            3.2033      0.240     13.345      0.000       2.733       3.674
Name_Mini                     6.7735      0.550     12.308      0.000       5.695       7.852
Name_Mitsubishi               0.5319      0.608      0.875      0.382      -0.660       1.724
Name_Nissan                  -0.4733      0.296     -1.600      0.110      -1.053       0.107
Name_OpelCorsa                6.1412      2.538      2.419      0.016       1.165      11.118
Name_Porsche                  3.5458      0.696      5.093      0.000       2.181       4.911
Name_Renault                 -0.4751      0.241     -1.972      0.049      -0.947      -0.003
Name_Skoda                   -0.3602      0.241     -1.493      0.135      -0.833       0.113
Name_Smart                   -2.1719      2.557     -0.849      0.396      -7.184       2.840
Name_Tata                    -1.2738      0.221     -5.763      0.000      -1.707      -0.840
Name_Toyota                   1.6564      0.203      8.178      0.000       1.259       2.053
Name_Volkswagen              -0.7145      0.183     -3.911      0.000      -1.073      -0.356
Name_Volvo                    2.6198      0.563      4.654      0.000       1.516       3.723
Location_Bangalore            0.2530      0.228      1.110      0.267      -0.194       0.700
Location_Chennai             -0.0584      0.219     -0.267      0.790      -0.487       0.370
Location_Coimbatore           0.2023      0.209      0.966      0.334      -0.208       0.613
Location_Delhi               -0.8563      0.214     -4.009      0.000      -1.275      -0.437
Location_Hyderabad            0.2462      0.205      1.198      0.231      -0.157       0.649
Location_Jaipur              -0.0678      0.226     -0.300      0.764      -0.510       0.375
Location_Kochi               -0.3495      0.210     -1.666      0.096      -0.761       0.062
Location_Kolkata             -1.0095      0.215     -4.701      0.000      -1.431      -0.588
Location_Mumbai              -0.3433      0.204     -1.686      0.092      -0.742       0.056
Location_Pune                -0.1819      0.209     -0.870      0.384      -0.592       0.228
Fuel_Type_Electric            5.6238      1.806      3.114      0.002       2.083       9.165
Fuel_Type_LPG                -0.6142      0.812     -0.756      0.449      -2.206       0.978
Fuel_Type_Petrol             -1.2435      0.117    -10.635      0.000      -1.473      -1.014
Transmission_Manual          -0.7718      0.121     -6.394      0.000      -1.008      -0.535
Owner_Type_Fourth & Above     0.0703      0.852      0.083      0.934      -1.599       1.740
Owner_Type_Second            -0.2488      0.105     -2.377      0.018      -0.454      -0.044
Owner_Type_Third             -0.4791      0.280     -1.713      0.087      -1.027       0.069
==============================================================================
Omnibus:                      201.479   Durbin-Watson:                   2.013
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              625.279
Skew:                           0.046   Prob(JB):                    1.67e-136
Kurtosis:                       4.717   Cond. No.                     1.14e+16
==============================================================================
```

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.6e-19. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

In [188…

```
X_train4 = X_train3.drop("Engine", axis=1)
vif_series3 = pd.Series(
    [variance_inflation_factor(X_train4.values, i) for i in range(X_train4.shape[1])],
    index=X_train4.columns,
)
print("VIF Scores: \n\n{}\n".format(vif_series3))
```

```
VIF Scores:

const                         203.087484
Unnamed: 0                      1.009437
Age_of_Car                      2.171142
Kilometers_Driven               1.933292
Mileage                         2.875093
Power                           4.584163
Name_Audi                       2.005452
Name_BMW                        2.245323
Name_Bentley                    1.024193
Name_Chevrolet                  1.117175
Name_Datsun                     1.018206
Name_Fiat                       1.033625
Name_Force                      1.012416
Name_Ford                       1.275100
Name_Hindustan                       NaN
Name_Honda                      1.571491
Name_Hyundai                    1.722503
Name_ISUZU                      1.016563
Name_Isuzu                      1.012857
Name_Jaguar                     1.237107
Name_Jeep                       1.070163
Name_Lamborghini                1.024193
Name_Land                       1.264794
Name_Mahindra                   1.616945
Name_Mercedes-Benz              2.364229
Name_Mini                       1.082880
Name_Mitsubishi                 1.071009
Name_Nissan                     1.084946
Name_OpelCorsa                  1.006310
Name_Porsche                    1.110225
Name_Renault                    1.130176
Name_Skoda                      1.278347
Name_Smart                      1.011498
Name_Tata                       1.152938
Name_Toyota                     1.819513
Name_Volkswagen                 1.276551
Name_Volvo                      1.128738
Location_Bangalore              2.431160
Location_Chennai                2.800673
Location_Coimbatore             3.359499
Location_Delhi                  2.905012
Location_Hyderabad              3.501461
Location_Jaipur                 2.537576
Location_Kochi                  3.338676
Location_Kolkata                2.977845
Location_Mumbai                 3.712804
Location_Pune                   3.303000
Fuel_Type_Electric              1.013586
Fuel_Type_LPG                   1.024585
Fuel_Type_Petrol                2.125654
Transmission_Manual             2.335767
Owner_Type_Fourth & Above       1.017526
Owner_Type_Second               1.179485
Owner_Type_Third                1.115705
dtype: float64
```

Observation:

1. Dropping engine has brought down all the high VIF values
2. All VIF values are now below 5 indicating no multi-colinearity
3. Power's VIF vlaue dropped to below 5.

In [189

```
olsmod3 = sm.OLS(y_train, X_train4)
```

```
olsmod3 = sm.OLS(y_train, X_train4)
olsres3 = olsmod3.fit()
print(olsres2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.737
Model:                            OLS   Adj. R-squared:                  0.734
Method:                 Least Squares   F-statistic:                     265.7
Date:                Fri, 21 May 2021   Prob (F-statistic):               0.00
Time:                        14:14:56   Log-Likelihood:                -11885.
No. Observations:                5075   AIC:                         2.388e+04
Df Residuals:                    5021   BIC:                         2.423e+04
Df Model:                          53
Covariance Type:            nonrobust
==============================================================================================
                                coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------------
const                         8.5498      0.577     14.817      0.000       7.419       9.681
Unnamed: 0                    0.0003   1.71e-05     14.780      0.000       0.000       0.000
Age_of_Car                   -0.4624      0.016    -28.395      0.000      -0.494      -0.430
Kilometers_Driven          -9.61e-06   1.64e-06     -5.862      0.000   -1.28e-05    -6.4e-06
Mileage                      -0.1010      0.015     -6.551      0.000      -0.131      -0.071
Engine                        0.0010      0.000      5.376      0.000       0.001       0.001
Power                         0.0284      0.002     13.271      0.000       0.024       0.033
Name_Audi                     3.8691      0.258     14.997      0.000       3.363       4.375
Name_BMW                      3.4813      0.264     13.194      0.000       2.964       3.999
Name_Bentley                  2.5047      1.814      1.381      0.167      -1.051       6.061
Name_Chevrolet               -0.8413      0.265     -3.174      0.002      -1.361      -0.322
Name_Datsun                  -1.7423      0.738     -2.360      0.018      -3.189      -0.295
Name_Fiat                    -0.3976      0.497     -0.800      0.424      -1.372       0.577
Name_Force                   -0.2380      1.471     -0.162      0.871      -3.122       2.646
Name_Ford                    -0.5270      0.190     -2.766      0.006      -0.900      -0.154
Name_Hindustan            -8.057e-15   6.61e-15     -1.218      0.223    -2.1e-14    4.91e-15
Name_Honda                   -0.3995      0.146     -2.744      0.006      -0.685      -0.114
Name_Hyundai                 -0.2258      0.120     -1.874      0.061      -0.462       0.010
Name_ISUZU                   -3.4811      1.806     -1.928      0.054      -7.021       0.059
Name_Isuzu                    0.4738      1.805      0.262      0.793      -3.066       4.013
Name_Jaguar                   3.4592      0.492      7.037      0.000       2.496       4.423
Name_Jeep                     3.2386      0.678      4.773      0.000       1.908       4.569
Name_Lamborghini              5.7563      2.562      2.247      0.025       0.733      10.779
Name_Land                     4.8765      0.422     11.568      0.000       4.050       5.703
Name_Mahindra                -0.7188      0.219     -3.285      0.001      -1.148      -0.290
Name_Mercedes-Benz            3.2033      0.240     13.345      0.000       2.733       3.674
Name_Mini                     6.7735      0.550     12.308      0.000       5.695       7.852
Name_Mitsubishi               0.5319      0.608      0.875      0.382      -0.660       1.724
Name_Nissan                  -0.4733      0.296     -1.600      0.110      -1.053       0.107
Name_OpelCorsa                6.1412      2.538      2.419      0.016       1.165      11.118
Name_Porsche                  3.5458      0.696      5.093      0.000       2.181       4.911
Name_Renault                 -0.4751      0.241     -1.972      0.049      -0.947      -0.003
Name_Skoda                   -0.3602      0.241     -1.493      0.135      -0.833       0.113
Name_Smart                   -2.1719      2.557     -0.849      0.396      -7.184       2.840
Name_Tata                    -1.2738      0.221     -5.763      0.000      -1.707      -0.840
Name_Toyota                   1.6564      0.203      8.178      0.000       1.259       2.053
Name_Volkswagen              -0.7145      0.183     -3.911      0.000      -1.073      -0.356
Name_Volvo                    2.6198      0.563      4.654      0.000       1.516       3.723
Location_Bangalore            0.2530      0.228      1.110      0.267      -0.194       0.700
Location_Chennai             -0.0584      0.219     -0.267      0.790      -0.487       0.370
Location_Coimbatore           0.2023      0.209      0.966      0.334      -0.208       0.613
Location_Delhi               -0.8563      0.214     -4.009      0.000      -1.275      -0.437
Location_Hyderabad            0.2462      0.205      1.198      0.231      -0.157       0.649
Location_Jaipur              -0.0678      0.226     -0.300      0.764      -0.510       0.375
Location_Kochi               -0.3495      0.210     -1.666      0.096      -0.761       0.062
Location_Kolkata             -1.0095      0.215     -4.701      0.000      -1.431      -0.588
Location_Mumbai              -0.3433      0.204     -1.686      0.092      -0.742       0.056
Location_Pune                -0.1819      0.209     -0.870      0.384      -0.592       0.228
Fuel_Type_Electric            5.6238      1.806      3.114      0.002       2.083       9.165
Fuel_Type_LPG                -0.6142      0.812     -0.756      0.449      -2.206       0.978
Fuel_Type_Petrol             -1.2435      0.117    -10.635      0.000      -1.473      -1.014
Transmission_Manual          -0.7718      0.121     -6.394      0.000      -1.008      -0.535
Owner_Type_Fourth & Above     0.0703      0.852      0.083      0.934      -1.599       1.740
Owner_Type_Second            -0.2488      0.105     -2.377      0.018      -0.454      -0.044
Owner_Type_Third             -0.4791      0.280     -1.713      0.087      -1.027       0.069
==============================================================================
Omnibus:                      201.479   Durbin-Watson:                   2.013
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              625.279
Skew:                           0.046   Prob(JB):                    1.67e-136
Kurtosis:                       4.717   Cond. No.                     1.14e+16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.6e-19. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

Observation:

1. Since multi-colinerity has been tested, drop variables where value of p is greater than 0.05.
2. Since none of the of the individual values are greater than 0.05, no variable has to be dropped.
3. No categorical variables will be removed even though their p values are greater than 0.05 because it is from a categorical variable and there are other levels of this category that are significant.

Mean of residuals should be 0.

In [190…

```
residual = olsres3.resid #mean of residuals
np.mean(residual)
```

Out[190… 1.9748264764032868e-11

Observation:

1. Mean of residuals is close to 0.

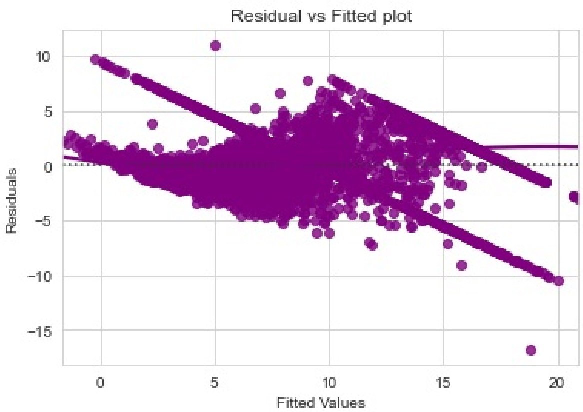
Test for linearity

In [191…

```
residual = olsres3.resid
fitted = olsres3.fittedvalues
```

In [192…

```
sns.set_style("whitegrid") #residuals vs fitted plot
sns.residplot(fitted, residual, color="purple", lowess=True)
plt.xlabel("Fitted Values")
plt.ylabel("Residuals")
plt.title("Residual vs Fitted plot")
plt.show()
```



Observations:

1. Scatter plot shows the distribution of residuals (errors) vs fitted values (predicted values).
2. If there exist any pattern in this plot, we consider it as signs of non-linearity in the data and a pattern means that the model doesn't capture non-linear effects.
3. We see no pattern in the plot above. Hence, the assumption is satisfied.

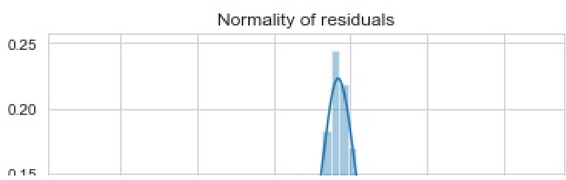
Test for Normality

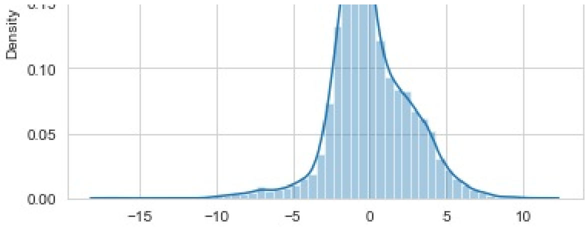
It can be checked via QQ Plot, Residuals following normal distribution will make a straight line plot otherwise not.

Other test to check for normality : Shapiro-Wilk test.

In [193…

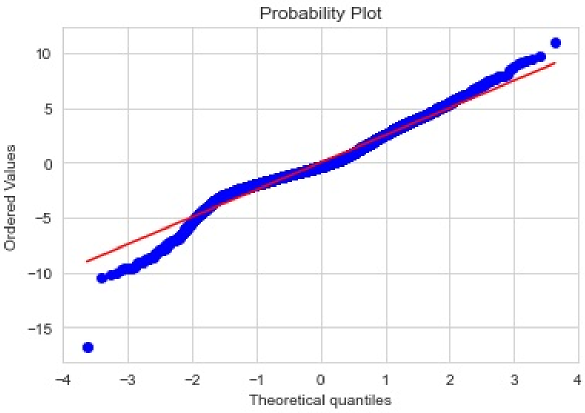
```
sns.distplot(residual) #desnisty plot
plt.title("Normality of residuals")
plt.show()
```





In [194…
```
import pylab
import scipy.stats as stats

stats.probplot(residual, dist="norm", plot=pylab)# probabilty plot
plt.show()
```



In [195…
```
stats.shapiro(residual) #shapiro test
```

Out[195… ShapiroResult(statistic=0.9686605334281921, pvalue=6.801046498356176e-32)

- The residuals are not normal as per shapiro test, but as per QQ plot they are approximately normal.
- The issue with shapiro test is when dataset is big, even for small deviations, it shows data as not normal.
- Hence we go with the QQ plot and say that residuals are normal.
- We can try to treat data for outliers and see if that helps in further normalizing the residual curve.

TEST FOR HOMOSCEDASTICITY

For goldfeldquandt test, the null and alternate hypotheses are as follows:

Null hypothesis : Residuals are homoscedastic Alternate hypothesis : Residuals have heteroscedasticity

In [196…
```
import statsmodels.stats.api as sms
from statsmodels.compat import lzip

name = ["F statistic", "p-value"]
test = sms.het_goldfeldquandt(residual, X_train4)
lzip(name, test)
```

Out[196… [('F statistic', 1.0625766900919138), ('p-value', 0.06509422887655633)]

Since p-value > 0.05, we can say that the residuals are homoscedastic. This assumption is therefore valid in the data.

Now we have checked all the assumptions and they are satisfied, so we can move towards the prediction part.

Predicting on the test data

In [197…
```
X_train4.columns
X_train4.shape
```

Out[197… (5075, 54)

In [198…
```
# Selecting columns from test data that we used to create our final model
```

```
X_test_final = X_test[X_train4.columns]
X_test_final.head()
```

Out[198...

	const	Unnamed: 0	Age_of_Car	Kilometers_Driven	Mileage	Power	Name_Audi	Name_BMW	Name_Bentley	Name_Chevrolet	...	Locatior
2952	1.0	2952	8.0	40000	20.77	88.80	0	0	0	0	...	
1634	1.0	1634	4.0	68193	17.11	174.33	1	0	0	0	...	
2622	1.0	2622	8.0	60000	19.40	86.80	0	0	0	0	...	
7048	1.0	7048	3.0	41200	28.09	88.50	0	0	0	0	...	
6732	1.0	6732	3.0	35000	24.30	88.50	0	0	0	0	...	

5 rows × 54 columns

In [206...

```
# Checking model performance on train set (seen 70% data)
print("Train Performance\n")
print(y_train)
model_perf(olsres3, X_train4.values, y_train)
```

```
Train Performance

          Price
952    3.650000
2264   3.500000
1504   3.000000
7210   9.479468
5538   2.500000
...         ...
3772   4.150000
5191   5.900000
5226  12.500000
5390   2.250000
860    4.750000

[5075 rows x 1 columns]
```

Out[206...

	MAE	MAPE	RMSE	R^2
0	1.888012	32.127677	2.523985	0.735674

In [200...

```
# Checking model performance on test set (seen 70% data)
print("Test Performance\n")
model_perf(olsres3, X_test_final.values, y_test)
```

```
Test Performance
```

Out[200...

	MAE	MAPE	RMSE	R^2
0	1.905936	32.76904	2.521289	0.739767

Observation:

1. Now we can see that the model has low test and train RMSE and MAE, and both the errors are comparable. So, our model is not suffering from overfitting.
2. The model is able to explain 73.9% of the variation on the test set, which is very good.
3. The MAPE on the test set suggests we can predict within 32% of the price model.

In [201...

```
# let us print the model summary

olsmod3 = sm.OLS(y_train, X_train4)
olsres3 = olsmod3.fit()
print(olsres3.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Price   R-squared:                       0.736
Model:                            OLS   Adj. R-squared:                  0.733
Method:                 Least Squares   F-statistic:                     268.8
Date:                Fri, 21 May 2021   Prob (F-statistic):               0.00
Time:                        14:15:03   Log-Likelihood:                -11900.
No. Observations:                5075   AIC:                         2.391e+04
Df Residuals:                    5022   BIC:                         2.425e+04
```

```
Df Model:                          52
Covariance Type:            nonrobust
============================================================================================
                               coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------
const                       10.0394      0.508     19.780      0.000       9.044      11.034
Unnamed: 0                   0.0003   1.72e-05     14.802      0.000       0.000       0.000
Age_of_Car                  -0.4638      0.016    -28.411      0.000      -0.496      -0.432
Kilometers_Driven         -9.27e-06   1.64e-06     -5.643      0.000   -1.25e-05   -6.05e-06
Mileage                     -0.1359      0.014     -9.689      0.000      -0.163      -0.108
Power                        0.0357      0.002     21.291      0.000       0.032       0.039
Name_Audi                    3.7672      0.258     14.602      0.000       3.261       4.273
Name_BMW                     3.4258      0.264     12.958      0.000       2.907       3.944
Name_Bentley                 3.0443      1.816      1.676      0.094      -0.516       6.605
Name_Chevrolet              -0.8754      0.266     -3.295      0.001      -1.396      -0.354
Name_Datsun                 -1.6471      0.740     -2.226      0.026      -3.098      -0.196
Name_Fiat                   -0.5177      0.498     -1.040      0.298      -1.494       0.458
Name_Force                  -0.0162      1.474     -0.011      0.991      -2.907       2.874
Name_Ford                   -0.4599      0.191     -2.413      0.016      -0.834      -0.086
Name_Hindustan            3.839e-14   1.48e-14      2.599      0.009    9.43e-15    6.74e-14
Name_Honda                  -0.3423      0.146     -2.350      0.019      -0.628      -0.057
Name_Hyundai                -0.2633      0.121     -2.183      0.029      -0.500      -0.027
Name_ISUZU                  -3.0777      1.809     -1.701      0.089      -6.625       0.469
Name_Isuzu                   1.1564      1.806      0.640      0.522      -2.384       4.697
Name_Jaguar                  3.5052      0.493      7.112      0.000       2.539       4.471
Name_Jeep                    2.9846      0.679      4.397      0.000       1.654       4.315
Name_Lamborghini             6.2018      2.568      2.415      0.016       1.167      11.236
Name_Land                    4.8332      0.423     11.436      0.000       4.005       5.662
Name_Mahindra               -0.4637      0.214     -2.165      0.030      -0.884      -0.044
Name_Mercedes-Benz           3.2235      0.241     13.394      0.000       2.752       3.695
Name_Mini                    6.7308      0.552     12.198      0.000       5.649       7.813
Name_Mitsubishi              0.9896      0.604      1.640      0.101      -0.194       2.173
Name_Nissan                 -0.3680      0.296     -1.243      0.214      -0.948       0.212
Name_OpelCorsa               6.1597      2.546      2.420      0.016       1.169      11.150
Name_Porsche                 4.0662      0.691      5.882      0.000       2.711       5.421
Name_Renault                -0.5095      0.241     -2.110      0.035      -0.983      -0.036
Name_Skoda                  -0.3097      0.242     -1.281      0.200      -0.784       0.164
Name_Smart                  -3.4868      2.552     -1.366      0.172      -8.490       1.516
Name_Tata                   -1.2374      0.222     -5.585      0.000      -1.672      -0.803
Name_Toyota                  2.0504      0.189     10.829      0.000       1.679       2.422
Name_Volkswagen             -0.7208      0.183     -3.935      0.000      -1.080      -0.362
Name_Volvo                   2.4267      0.563      4.308      0.000       1.322       3.531
Location_Bangalore           0.2586      0.228      1.132      0.258      -0.189       0.706
Location_Chennai            -0.0445      0.219     -0.203      0.839      -0.474       0.385
Location_Coimbatore          0.2050      0.210      0.976      0.329      -0.207       0.617
Location_Delhi              -0.8545      0.214     -3.990      0.000      -1.274      -0.435
Location_Hyderabad           0.2538      0.206      1.232      0.218      -0.150       0.658
Location_Jaipur             -0.0676      0.226     -0.299      0.765      -0.511       0.376
Location_Kochi              -0.3461      0.210     -1.645      0.100      -0.758       0.066
Location_Kolkata            -0.9980      0.215     -4.635      0.000      -1.420      -0.576
Location_Mumbai             -0.3319      0.204     -1.626      0.104      -0.732       0.068
Location_Pune               -0.1773      0.210     -0.845      0.398      -0.588       0.234
Fuel_Type_Electric           4.9280      1.807      2.728      0.006       1.386       8.470
Fuel_Type_LPG               -0.8545      0.813     -1.051      0.293      -2.448       0.739
Fuel_Type_Petrol            -1.5312      0.104    -14.687      0.000      -1.736      -1.327
Transmission_Manual         -0.7681      0.121     -6.345      0.000      -1.005      -0.531
Owner_Type_Fourth & Above    0.0654      0.854      0.077      0.939      -1.609       1.739
Owner_Type_Second           -0.2550      0.105     -2.429      0.015      -0.461      -0.049
Owner_Type_Third            -0.4825      0.280     -1.720      0.085      -1.032       0.067
==============================================================================
Omnibus:                      198.115   Durbin-Watson:                   2.012
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              612.123
Skew:                           0.029   Prob(JB):                    1.20e-133
Kurtosis:                       4.700   Cond. No.                     1.14e+16
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.6e-19. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

Conclusion

olsres3 is our final model which follows all the assumptions, and can be used for interpretations.

1. Owner Type, Transmission, Fuel type, location, km driven, age of car all decrease as price increases and vice versa as indicated by negative coeficent.
2. Fuel_type, Location and Name of the cars were big indicators in determining price, as change in any of these variables changed the

overall r-squared and adjusted r-squared values.
3. Increase in power, increases price value by 0.0357, which is about Rs.3570.
4. As age of car increased the price would decrease by Rs 46,380.
5. As mileage of the car increase, the price would again decrease by 0.1359 which is about Rs. 13, 590.
6. As the number of owners increased for a car, for the second and third car as the price would decrease from Rs 25500 to s 48250, but for owners 4 and above, the price would increase by Rs 6540.
7. As a car is indentified as Electric fuel type the price of the car would increase by Rs. 49,280
8. For high end luxury cars such as Volvo, Porsche, Lamborghini, BMW, Audi etc, the price would increase.
9. For low end cars, such as Maruthi, Hyundai, Honda etc the price would decrease.
10. Locations such as Coimbatore and Kochi had higher price increase as compared to other locations such as Kolkata and Mumbai where prices generally decreased.

Recommendation

1. Age of car, mileage and and kilometers driven have all negative impact on the price of car, i.e as any of these values increases the price of the car will significantly reduce.
2. Name of the car is a big indicator in how price is determined, as high-end luxury cars like Lamborghini and Porsche are much higher priced as compared to low-end budget cars like Maruthi or Honda.
3. Location wise the most priced cars are sold in Coimbatore and Kochi, while the low-end, low-price cars are sold in Kolkata, Mumbai etc.
4. Electric cars are much more expensive than any other fuel types and it will be better to sell electric cars as compared to Petrol or Diesel although both these fuel types heavily determine the price.
5. Owner type is also a big indication of the pricing, selling first-hand cars with no previous owners will yield higher price compared to pre-owned cars.
6. The number of seats did not have much of an impact in deciding the price of the car