Report On Used Car Price Analysis & Prediction

Content

1 Introduction

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 foot holes 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. We have to come up with a pricing model that can effectively predict the price of used cars and can help the business in devising profitable strategies using differential pricing.

1.1 Problem Statement

Identify the factors that affect a **second-hand car's value**, leading us to create a *car price prediction model* in near future, which may help the buyers to learn the actual market value of a car before buying or selling. Before we create our own car price prediction model, let's understand on what really affects a car's price.

Questions we will be answering here before any prediction model:

- Does various predicating factors affect the price of the used car .?
- What all independent variables effect the pricing of used cars?
- Does name of a car have any effect on pricing of car.?
- How does type of Transmission effect pricing?
- Does Location in which the car being sold has any effect on the price?
- Do kilometres Driven; Year of manufacturing have negative correlation with price of the car?
- Does Mileage, Engine and Power have any effect on the pricing of the car?
- How does number of seats, Fuel type effect the pricing.?

1.2 About Dataset

Data set used here consist record of used car from 1996-2015 from India. In consist various columns which describe Car name (consist of model and brand of it), location, Kilometres Driven, year of Manufacturing, Owner type, Mileage, Engine, Power, Seats, New Price, Price. Here we first analysis each factor affects on price of car in present and past as per data available in dataset.

Data set view: -

```
In [3]: data.head()
```

Out[3]:

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	New_Price	Price
0	0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First	26.6 km/kg	998 CC	58.16 bhp	5.0	NaN	1.75
1	1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	19.67 kmpl	1582 CC	126.2 bhp	5.0	NaN	12.50
2	2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First	18.2 kmpl	1199 CC	88.7 bhp	5.0	8.61 Lakh	4.50
3	3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First	20.77 kmpl	1248 CC	88.76 bhp	7.0	NaN	6.00
4	4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second	15.2 kmpl	1968 CC	140.8 bhp	5.0	NaN	17.74

```
In [4]: data.tail()
```

Out[4]:

	S.No.	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	New_Price	Price
7248	7248	Volkswagen Vento Diesel Trendline	Hyderabad	2011	89411	Diesel	Manual	First	20.54 kmpl	1598 CC	103.6 bhp	5.0	NaN	NaN
7249	7249	Volkswagen Polo GT TSI	Mumbai	2015	59000	Petrol	Automatic	First	17.21 kmpl	1197 CC	103.6 bhp	5.0	NaN	NaN
7250	7250	Nissan Micra Diesel XV	Kolkata	2012	28000	Diesel	Manual	First	23.08 kmpl	1461 CC	63.1 bhp	5.0	NaN	NaN
7251	7251	Volkswagen Polo GT TSI	Pune	2013	52262	Petrol	Automatic	Third	17.2 kmpl	1197 CC	103.6 bhp	5.0	NaN	NaN
7252	7252	Mercedes-Benz E-Class 2009-2013 E 220 CDI Avan...	Kochi	2014	72443	Diesel	Automatic	First	10.0 kmpl	2148 CC	170 bhp	5.0	NaN	NaN

Fig. 1

Dataset has about 7253 Rows and 14 columns, there are lot missing value and mismanaged value which need to check first.

List of column present with respective data set in dataset:

```
In [6]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7253 entries, 0 to 7252
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   S.No.              7253 non-null   int64
 1   Name               7253 non-null   object
 2   Location           7253 non-null   object
 3   Year               7253 non-null   int64
 4   Kilometers_Driven  7253 non-null   int64
 5   Fuel_Type          7253 non-null   object
 6   Transmission       7253 non-null   object
 7   Owner_Type         7253 non-null   object
 8   Mileage            7251 non-null   object
 9   Engine             7207 non-null   object
 10  Power              7207 non-null   object
 11  Seats              7200 non-null   float64
 12  New_Price          1006 non-null   object
 13  Price              6019 non-null   float64
dtypes: float64(2), int64(3), object(9)
memory usage: 793.4+ KB
```

Observed Insights:

- Here the variables Mileage, Engine, Power, Seats, New_Price, and Price have missing values.
- Numeric variables like Mileage, Power, engine, New_Price are of datatype are showing object dtype need to change.
- Categorical variables like Location, Fuel_Type, Transmission, and Owner Type are of object data type.

1.3 More Data Understanding:

➢ Analysis Missing value at first

In [8]:
```
# Missing values Calculation
data.isnull().sum()
```

Out[8]:
```
S.No.                  0
Name                   0
Location               0
Year                   0
Kilometers_Driven      0
Fuel_Type              0
Transmission           0
Owner_Type             0
Mileage                2
Engine                46
Power                 46
Seats                 53
New_Price           6247
Price               1234
dtype: int64
```

In [8]:
```
# calculate the percentage of missing values in each column
(data.isnull().sum()/(len(data)))*100
```

Out[8]:
```
S.No.                 0.000000
Name                  0.000000
Location              0.000000
Year                  0.000000
Kilometers_Driven     0.000000
Fuel_Type             0.000000
Transmission          0.000000
Owner_Type            0.000000
Mileage               0.027575
Engine                0.634220
Power                 0.634220
Seats                 0.730732
New_Price            86.129877
Price                17.013650
dtype: float64
```

The percentage of missing values for the columns New_Price and Price is ~86% and ~17%, respectively.

Initial Insights: -

- **New_Price** has only 1006 values. 86 % values are missing
- **Price**, which is a Target variable 17 % missing values.This needs to be analysed further.
- **Seats** has only 53 values missing and number of seats can be one of key factor in deciding price.
- **Power** and **Engine** has 46 missing values.
- **Mileage** only has two values missing.
- **Mileage,Power,Engine,New_Price** we know are quantitative variables but are of object dtype here and needs to to converted to numeric.

Analysis of unique value in different categorical column: Code below

Making a list of all categorical variables

cat_col = ["Fuel_Type","Location", "Transmission", "Seats", "Year", "Owner_Type",]

Printing number of count of each unique value in each column

for column in cat_col:

    print(data[column].value_counts())

Result after execution of code-

```
]
# Printing number of count of each unique value in each column
for column in cat_col:
    print(data[column].value_counts())
    print("#" * 40)
```

```
Diesel       3852
Petrol       3325
CNG            62
LPG            12
Electric        2
Name: Fuel_Type, dtype: int64
########################################
Mumbai        949
Hyderabad     876
Coimbatore    772
Kochi         772
Pune          765
Delhi         660
Kolkata       654
Chennai       591
Jaipur        499
Bangalore     440
Ahmedabad     275
Name: Location, dtype: int64
########################################
Manual       5204
Automatic    2049
Name: Transmission, dtype: int64
########################################
```

```
########################################
5.0     6047
7.0      796
8.0      170
4.0      119
6.0       38
2.0       18
10.0       8
9.0        3
0.0        1
Name: Seats, dtype: int64
########################################
2015    929
2014    925
2016    886
2013    791
2017    709
2012    690
2011    579
2010    407
2018    361
2009    252
2008    207
2007    148
2019    119
2006     89
2005     68
2004     35
2003     20
2002     18
2001      8
2000      5
1998      4
1999      2
1996      1
Name: Year, dtype: int64
########################################
```

```
########################################
First             5952
Second            1152
Third              137
Fourth & Above      12
Name: Owner_Type, dtype: int64
########################################
```

Observed Insights ¶

- Maximum car being sold have fuel type as Diesel.
- Mumbai has highest numbers of car availabe for purchase.
- 5204 cars with Manual transmission are available for purchase.
- Most of the cars are 5 seaters and First owned.
- Years of car ranges form 1996- 2015

2. Data Pre-processing

2.1. Data variable transformation – as we observed that dataset consist some unit with respective variable such as power, engine, mileage which need to remove as it obstructs the analysis and even the model building stage. There are also some variables having data value as zero example zero mileage which is not feasible for analysis so we convert it in (NaN) value, which can handle later stage under missing value treatment stage

In [7]:
```
# checking data
data[['Engine','Power','Mileage']]
```

Out[7]:

	Engine	Power	Mileage
0	998 CC	58.16 bhp	26.6 km/kg
1	1582 CC	126.2 bhp	19.67 kmpl
2	1199 CC	88.7 bhp	18.2 kmpl
3	1248 CC	88.76 bhp	20.77 kmpl
4	1968 CC	140.8 bhp	15.2 kmpl
...	...	...	...
7248	1598 CC	103.6 bhp	20.54 kmpl
7249	1197 CC	103.6 bhp	17.21 kmpl
7250	1461 CC	63.1 bhp	23.08 kmpl
7251	1197 CC	103.6 bhp	17.2 kmpl
7252	2148 CC	170 bhp	10.0 kmpl

7253 rows × 3 columns

```
data_car['Power'].value_counts()
```

```
74 bhp        280
98.6 bhp      166
73.9 bhp      152
140 bhp       142
null bhp      129
             ...
152.88 bhp      1
74.96 bhp       1
199.3 bhp       1
68.1 bhp        1
181.04 bhp      1
Name: Power, Length: 386, dtype: int64
```

In [13]:
```
typeoffuel=['CNG','LPG']
data_car.loc[data_car.Fuel_Type.isin(typeoffuel)].head(10)
```

Out[13]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	New_Price	Price
0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First	26.6 km/kg	998 CC	58.16 bhp	5.0	NaN	1.75
5	Hyundai EON LPG Era Plus Option	Hyderabad	2012	75000	LPG	Manual	First	21.1 km/kg	814 CC	55.2 bhp	5.0	NaN	2.35
127	Maruti Wagon R LXI CNG	Pune	2013	89900	CNG	Manual	First	26.6 km/kg	998 CC	58.16 bhp	5.0	NaN	3.25
328	Maruti Zen Estilo LXI Green (CNG)	Pune	2008	42496	CNG	Manual	First	26.3 km/kg	998 CC	67.1 bhp	5.0	NaN	1.40
440	Maruti Eeco 5 STR With AC Plus HTR CNG	Kochi	2017	31841	CNG	Manual	First	15.1 km/kg	1196 CC	73 bhp	5.0	NaN	4.70
839	Maruti Alto Green LXi (CNG)	Delhi	2012	65537	CNG	Manual	First	26.83 km/kg	796 CC	38.4 bhp	5.0	NaN	2.10
893	Hyundai Accent Executive CNG	Hyderabad	2010	95637	CNG	Manual	Second	13.2 km/kg	1495 CC	93.7 bhp	5.0	NaN	1.90
936	Maruti Wagon R LXI LPG BSIV	Hyderabad	2012	72000	LPG	Manual	First	26.2 km/kg	998 CC	58.2 bhp	5.0	NaN	2.85
987	Maruti Wagon R LXI DUO BSIII	Mumbai	2008	64226	LPG	Manual	First	17.3 km/kg	1061 CC	57.5 bhp	5.0	NaN	1.45
1135	Maruti Zen Estilo LXI Green (CNG)	Ahmedabad	2011	76000	CNG	Manual	First	26.3 km/kg	998 CC	67.1 bhp	5.0	NaN	2.00

- Power has some values as "null bhp". Mileage also has some observations as 0. For fuel type and CNG and LPG mileage is measured in km/kg where as for other type it is measured in kmpl. Since those units are in km for both of them no need of conversion. Dropping units from mileages, Engine and Power.

Code for removing unit

For Mileage

In [8]:
```
data_car["Mileage"] = data_car["Mileage"].str.rstrip(" kmpl")
data_car["Mileage"] = data_car["Mileage"].str.rstrip(" km/g")
```

For Engine

In [9]:
```
data_car["Engine"] = data_car["Engine"].str.rstrip(" CC")
```

For Power

In [10]:
```
data_car["Power"] = data_car["Power"].str.rstrip(" bhp")
data_car["Power"] = data_car["Power"].replace(regex="null", value = np.nan)
```

Verifying data

In [11]:
```
#verify the data
data_car[['Engine','Power','Mileage']].sample(10)
```

Out[11]:

	Engine	Power	Mileage
4183	1248	74	22.3
1405	1248	88.8	20.77
4030	1498	89.84	22.7
6934	998	66.1	19.0
5318	2179	120	15.4
2072	2698	179.5	12.4
7238	1968	147.51	16.55
7118	1497	118	17.0
3004	1248	73.94	23.2
4283	1582	126.32	22.32

Checking of Zero in variable and replacing it with NaN which handle as missing value.

In [12]:
```
data_car.query("Mileage == '0.0'")['Mileage'].count()
```

Out[12]: 81

In [13]:
```
#Converting this observations to Nan so we will remember to handle them when handling missing values.
data_car.loc[data_car["Mileage"]=='0.0','Mileage']=np.nan
```

In [15]:
```
## Processing seat to check 0 seat if any
data_car.query("Seats == 0.0")['Seats']
```

Out[15]:
```
3999    0.0
Name: Seats, dtype: float64
```

In [16]:
```
#seats cannot be 0 so changing it to nan and will be handled in missing value
data_car.loc[3999,'Seats'] =np.nan
```

New list of null value

```
In [20]: data_car.isnull().sum()

Out[20]: Name                    0
         Location                0
         Year                    0
         Kilometers_Driven       0
         Fuel_Type               0
         Transmission            0
         Owner_Type              0
         Mileage                83
         Engine                 46
         Power                 175
         Seats                  54
         New_Price            6247
         Price                1234
         dtype: int64
```

There are 46 missing values in Engine, 175 in Power,83 in Mileage.

Removing unit from new price

```
In [18]: data_car["New_Price"] = data_car["New_Price"].str.rstrip(" Lakh")
         data_car["New_Price"] = data_car["New_Price"].str.rstrip(" Cr")

In [19]: data_car["New_Price"]

Out[19]: 0        NaN
         1        NaN
         2       8.61
         3        NaN
         4        NaN
                 ...
         7248     NaN
         7249     NaN
         7250     NaN
         7251     NaN
         7252     NaN
         Name: New_Price, Length: 7253, dtype: object
```

2.2. Feature Engineering:

Feature engineering refers to the process of using domain knowledge to select and transform the most relevant variables from raw data when creating a predictive model using machine learning or statistical modelling. The main goal of Feature engineering is to create meaningful data from raw data.

Creating Features-We will play around with the variables Year and Name in our dataset. If we see the sample data, the column “Year” shows the manufacturing year of the car.

Getting Car age column using year column date function

In [23]:
```
# Coverting Year in Car_age as it will easy to compare car on base of year old car since it greatly effect the price
from datetime import date
date.today().year
data_car['Car_Age']=date.today().year-data['Year']
data_car.head()
```

Out[23]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	New_Price	Price	Car_Age
0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First	26.60	998.0	58.16	5.0	NaN	1.75	12
1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	19.67	1582.0	126.20	5.0	NaN	12.50	7
2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First	18.20	1199.0	88.70	5.0	8.61	4.50	11
3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First	20.77	1248.0	88.76	7.0	NaN	6.00	10
4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second	15.20	1968.0	140.80	5.0	NaN	17.74	9

Split Name in Brand and model of cars

In [25]:
```
# Brands do play an important role in Car selection and Prices. So extracting brand names from the Name.
data_car['Brand'] = data_car.Name.str.split().str.get(0)
data_car['Model'] = data_car.Name.str.split().str.get(1) + data_car.Name.str.split().str.get(2)
data_car[['Name','Brand','Model']]
```

Out[25]:

	Name	Brand	Model
0	Maruti Wagon R LXI CNG	Maruti	WagonR
1	Hyundai Creta 1.6 CRDi SX Option	Hyundai	Creta1.6
2	Honda Jazz V	Honda	JazzV
3	Maruti Ertiga VDI	Maruti	ErtigaVDI
4	Audi A4 New 2.0 TDI Multitronic	Audi	A4New
...	...	...	...
7248	Volkswagen Vento Diesel Trendline	Volkswagen	VentoDiesel
7249	Volkswagen Polo GT TSI	Volkswagen	PoloGT
7250	Nissan Micra Diesel XV	Nissan	MicraDiesel
7251	Volkswagen Polo GT TSI	Volkswagen	PoloGT
7252	Mercedes-Benz E-Class 2009-2013 E 220 CDI Avan...	Mercedes-Benz	E-Class2009-2013

7253 rows × 3 columns

3. EDA (Exploratory Data Analysis)

3.1. Statistical Inference

In [33]:
```
data_car.describe().T
```

Out[33]:

	count	mean	std	min	25%	50%	75%	max
Year	7252.0	2013.366520	3.253162	1996.00	2011.000	2014.00	2016.00	2019.00
Kilometers_Driven	7252.0	58700.262686	84433.480370	171.00	34000.000	53429.00	73000.00	6500000.00
Mileage	7169.0	18.347106	4.157912	6.40	15.300	18.20	21.10	33.54
Engine	7206.0	1616.605051	595.320408	72.00	1198.000	1493.00	1968.00	5998.00
Power	7077.0	112.768713	53.496523	34.20	75.000	94.00	138.10	616.00
Seats	7198.0	5.280495	0.809376	2.00	5.000	5.00	5.00	10.00
New_Price	1006.0	19.894324	19.813947	1.00	7.635	11.27	23.64	99.92
Price	6019.0	9.479468	11.187917	0.44	3.500	5.64	9.95	160.00
Car_Age	7252.0	8.633480	3.253162	3.00	6.000	8.00	11.00	26.00

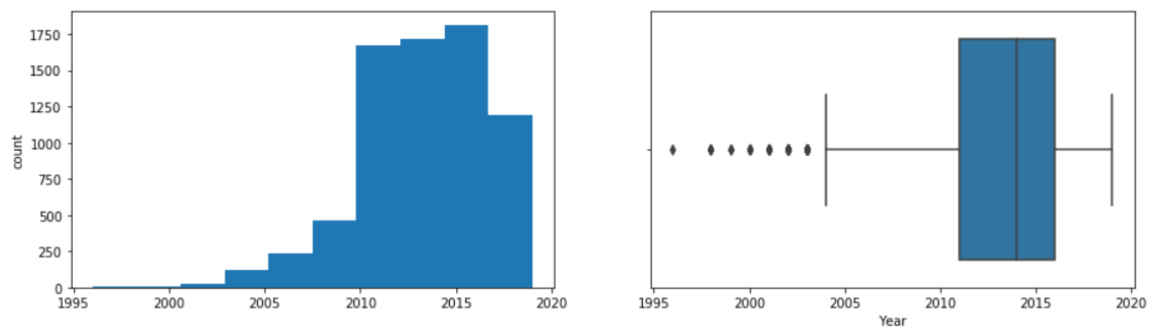
Observed insights-:

- age of car ranges from 3 to 25+ yrs where oldest car is 26 yrs old and avg. age of car is about 8.6 yr.
- On average of Kilometres-driven in Used cars are ~58k KM. The range shows a huge difference between min and max as max values show 650000 KM shows the evidence of an outlier. This record can be removed.
- Mileage is almost Normally distributed.
- Engine type (in cc) is right skewed and there may be outliers on higher and lower end
- There may also be some outlier in power & price.
- Price of car max. 160lakh high is very against other data which may be due outlier.

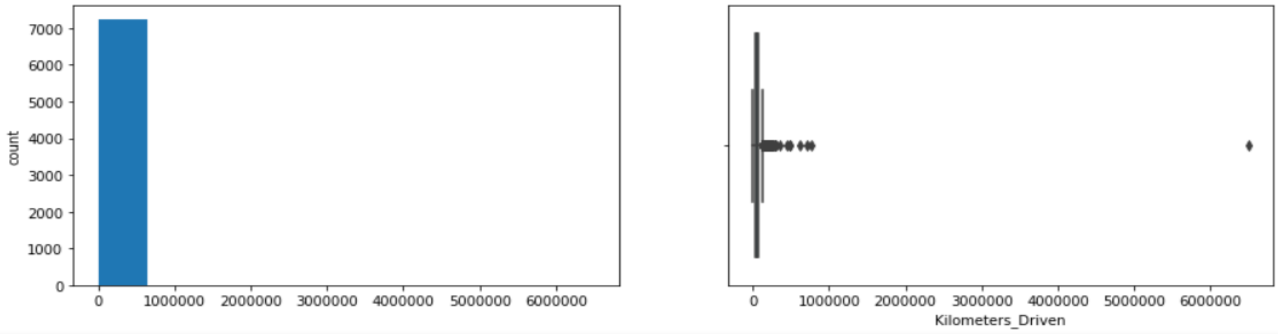
3.2. Univariate Analysis

For numerical variable

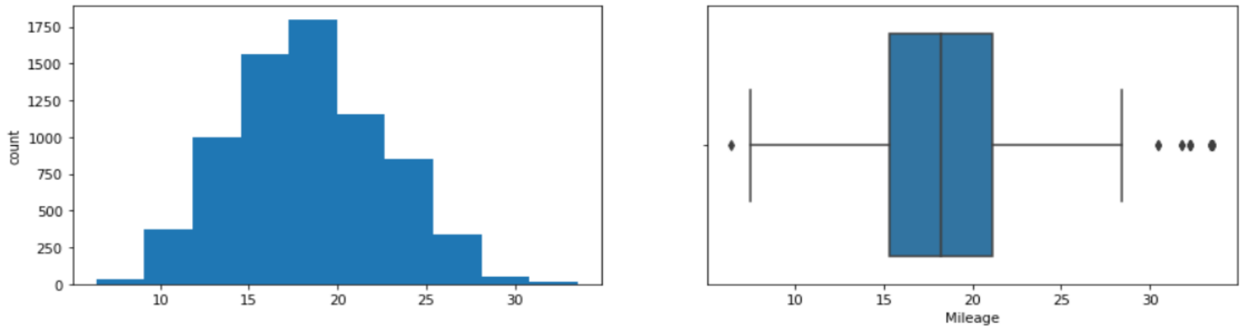
```
Year
Skew :  -0.84
```



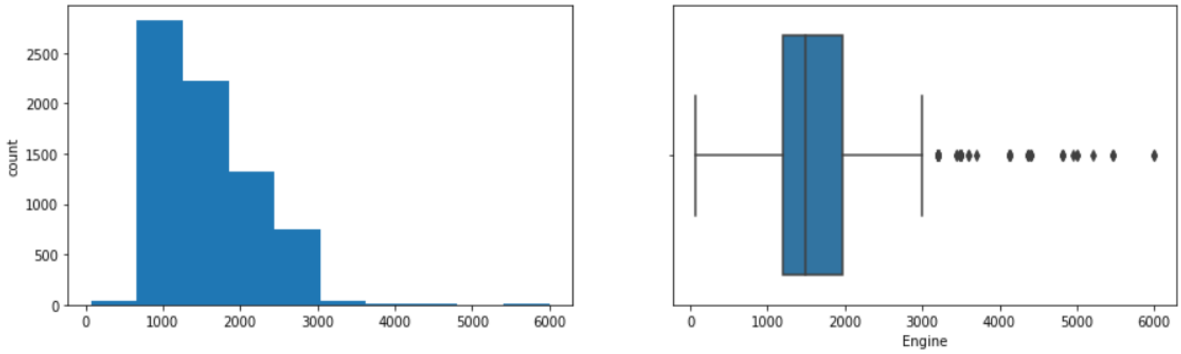
Kilometers_Driven
Skew : 61.58



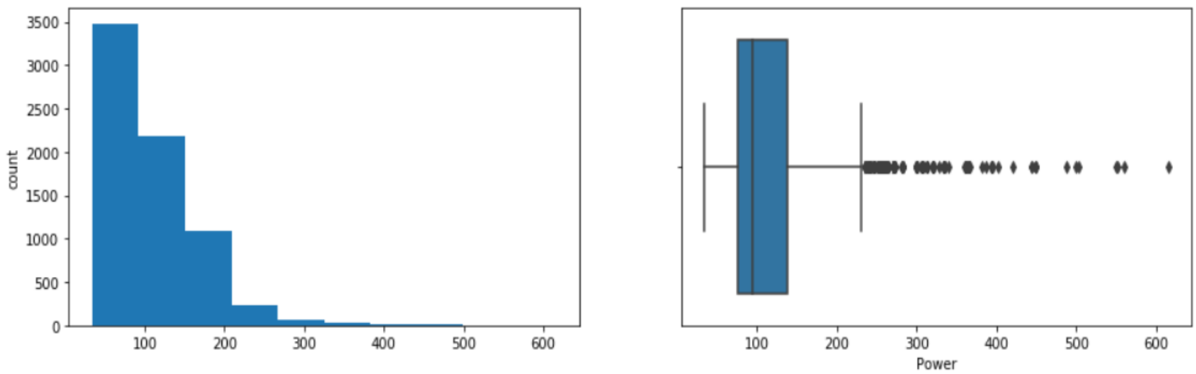
Mileage
Skew : 0.2



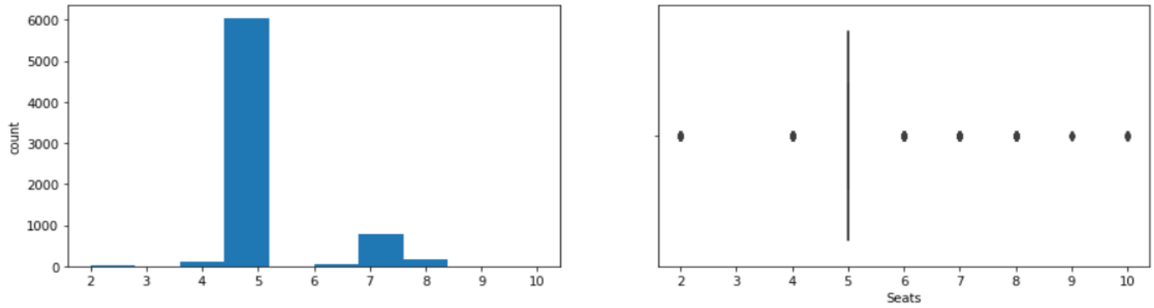
Engine
Skew : 1.41



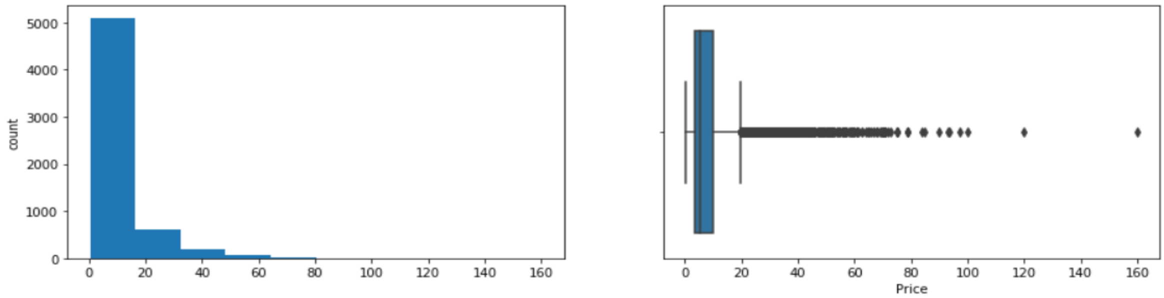
Power
Skew : 1.96



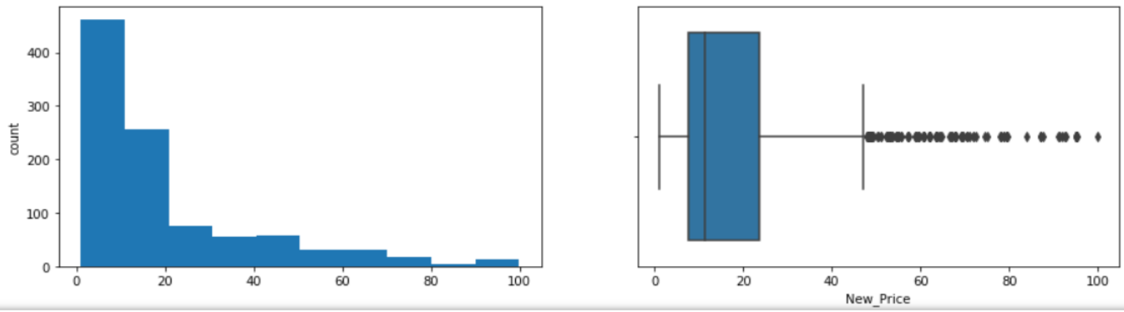
```
Seats
Skew : 1.95
```



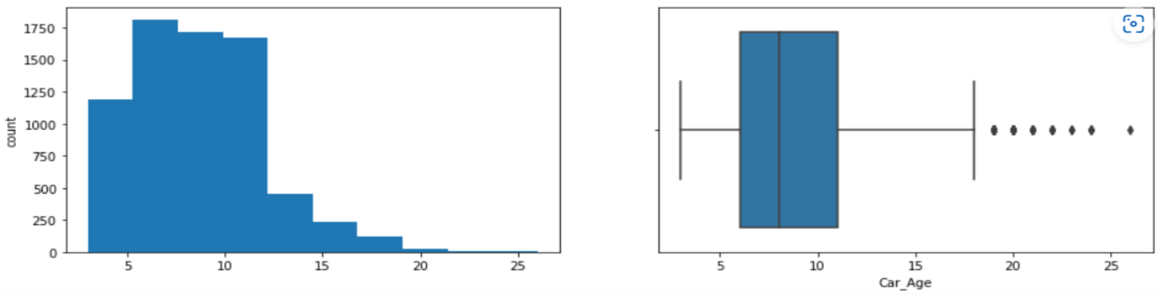

```
Price
Skew : 3.34
```




```
New_Price
Skew : 1.84
```



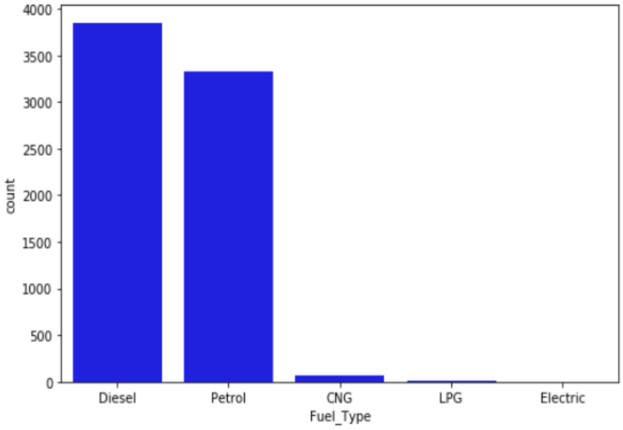

```
Car_Age
Skew : 0.84
```

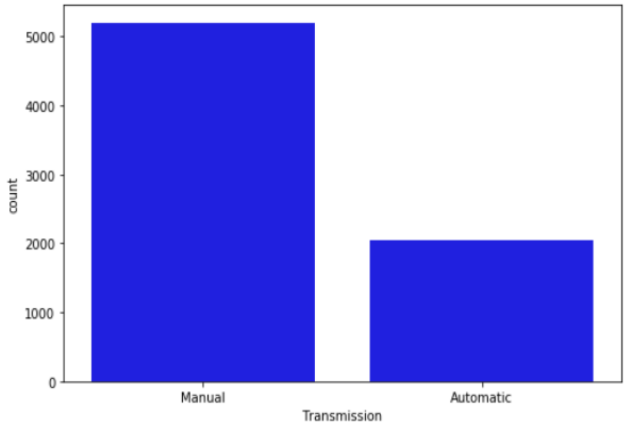


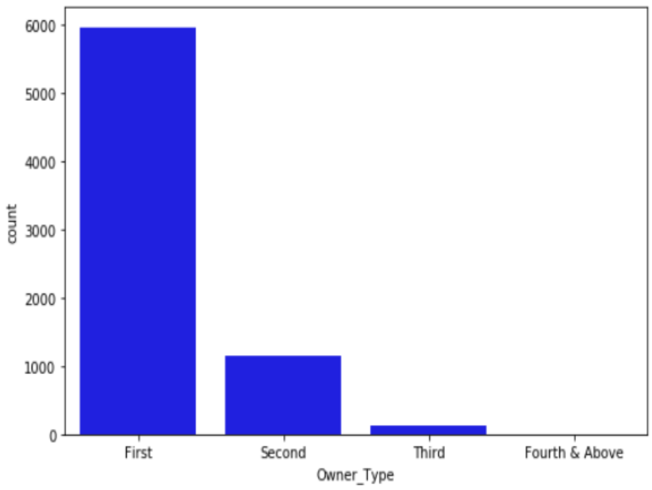
Observed insights:

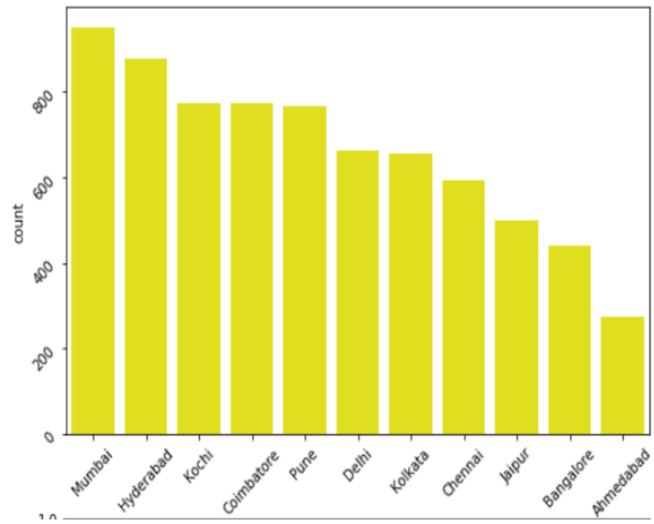
- Variable column of Price, New price, Kilometres Driven & power are right skewed for this data to be transformed, and all outliers will be handled during imputation.
- Mileage variable only show good normal distribution curve which is fit for model.

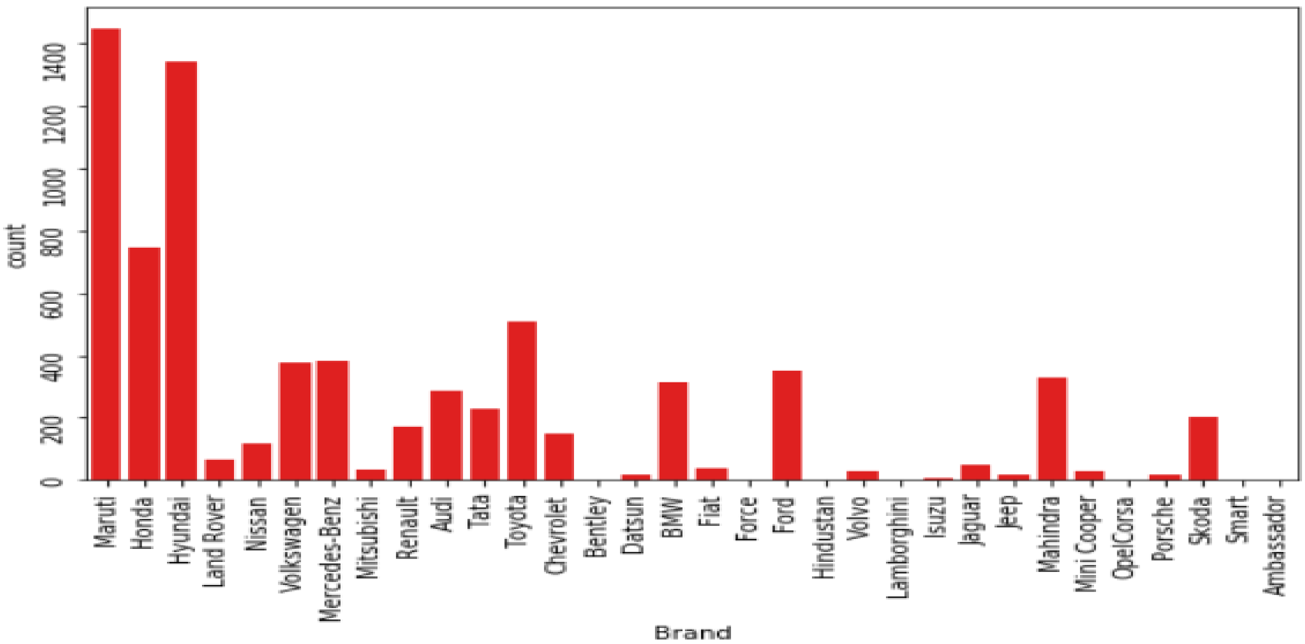
Analysis for categorical variable



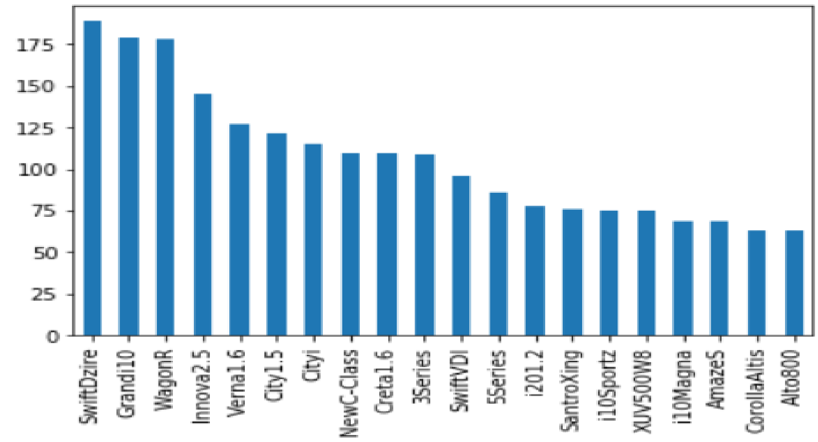








In [40]:
```
# For Model
data_car['Model'].value_counts().nlargest(20).plot(kind='bar');
plt.show()
```



Observed Insights

- Mumbai has the highest number of cars available for purchase, followed by Hyderabad and Coimbatore
- 53% of cars have fuel type as Diesel this shows diesel cars provide higher performance
- 72% of cars have manual transmission
- 82 % of cars are First owned cars. This shows most of the buyers prefer to purchase first-owner cars
- 20% of cars belong to the brand Maruti followed by 19% of cars belonging to Hyundai swiftDzire ranks first among all models which are available for purchase.

3.3. Data Transformation as observed in about visual there is some variable which is highly skew i.e. we need to transform data especially in case of price and kilometre driven of cars, here we used log transformation

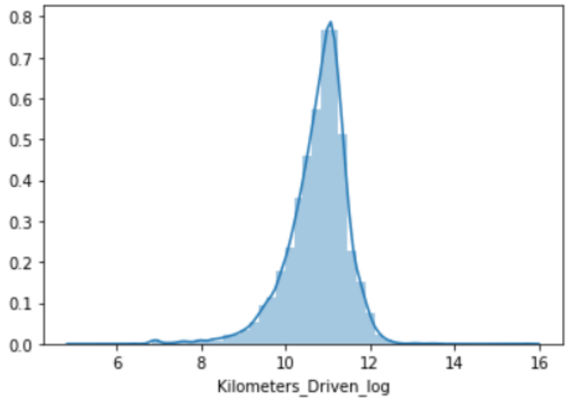
Used code;

In [36]:
```
# Function for log transformation of the column
def log_transform(data_car,col):
    for colname in col:
        if (data_car[colname] == 1.0).all():
            data_car[colname + '_log'] = np.log(data_car[colname]+1)
        else:
            data_car[colname + '_log'] = np.log(data_car[colname])
    data_car.info()
```

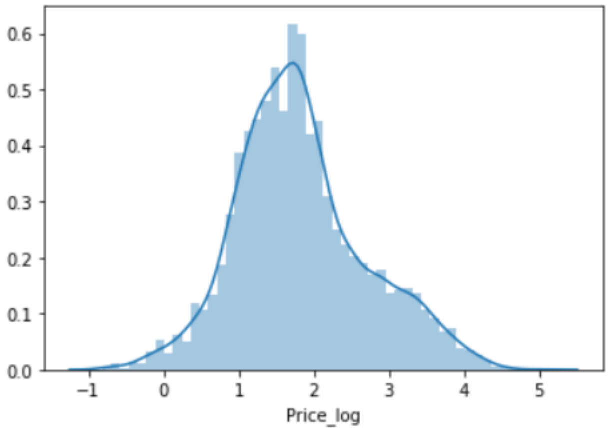
In [37]:
```
log_transform(data_car,['Kilometers_Driven','Price'])
```

Checking transformation

```
In [38]: #Log transformation of the feature 'Kilometers_Driven'
         sns.distplot(data_car["Kilometers_Driven_log"], axlabel="Kilometers_Driven_log");
```




```
In [44]: #Log transformation of the feature 'price'
         sns.distplot(data_car["Price_log"], axlabel="Price_log");
```

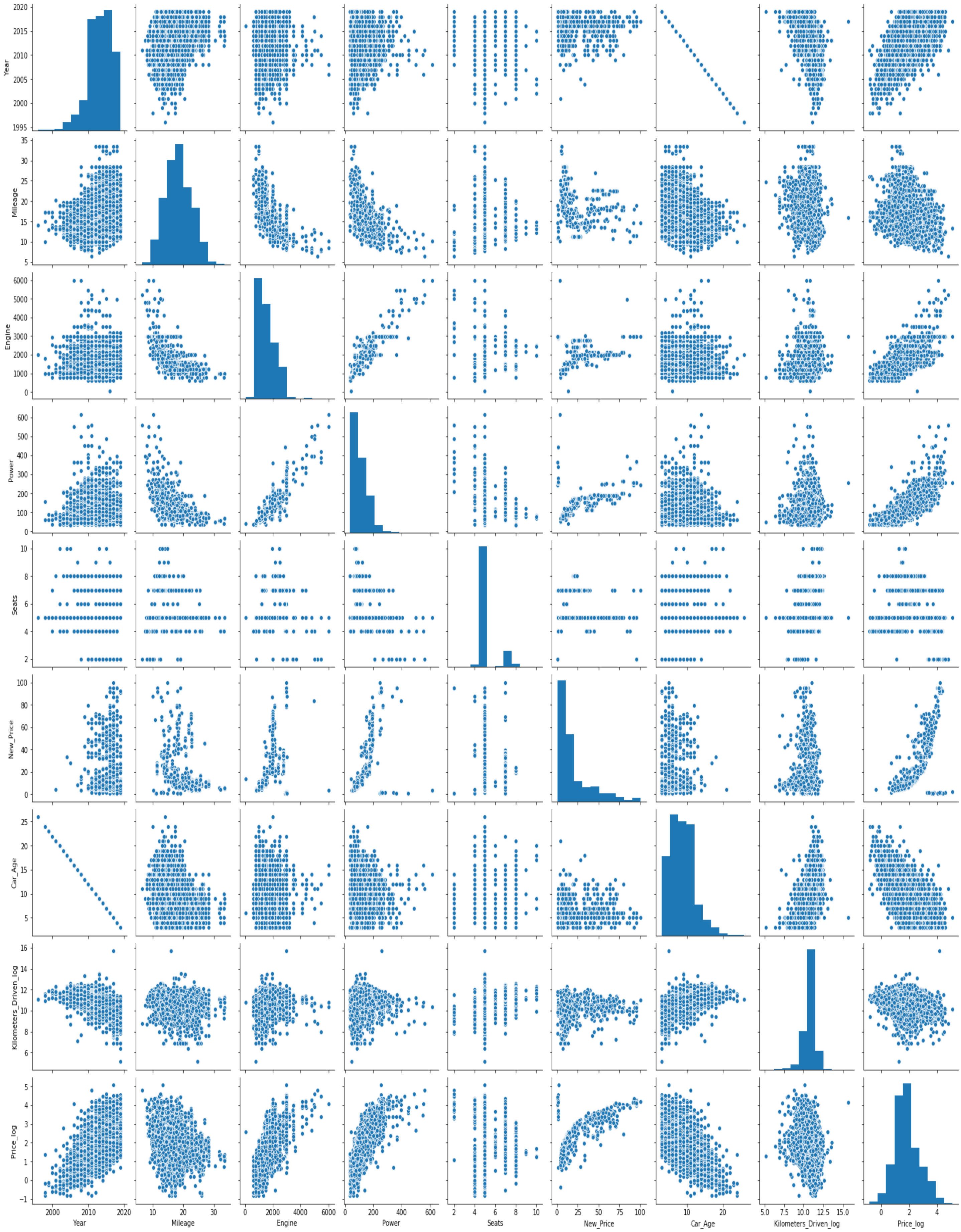


3.4. Bivariate Analysis

Checking for Numerical variable using pair plot

Code;

```
n [48]: plt.figure(figsize=(15,18))
        sns.pairplot(data=data_car.drop(['Kilometers_Driven','Price'],axis=1))
        plt.show()
```

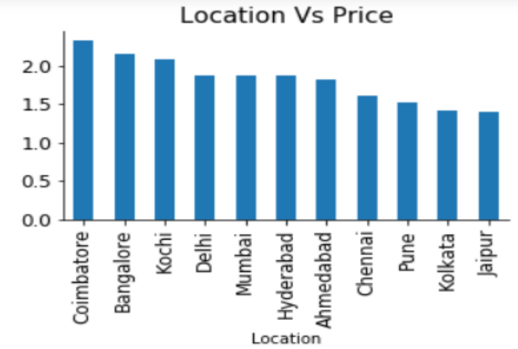


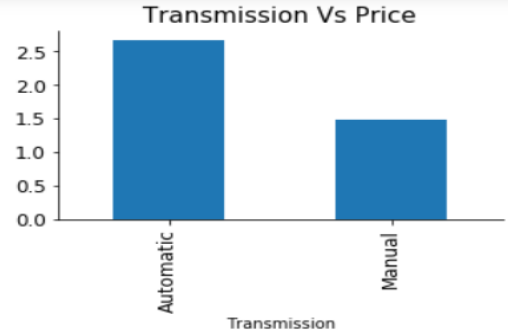
Year
Mileage
Engine
Power
Seats
New_Price
Car_Age
Kilometers_Driven_log
Price_log

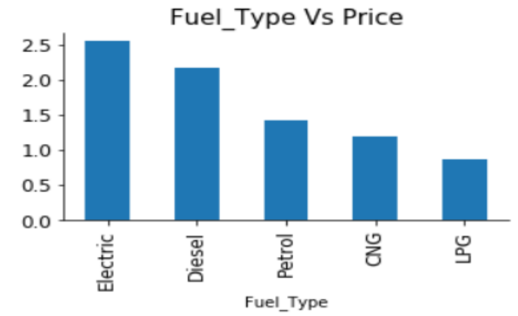
Pair Plot based insights:

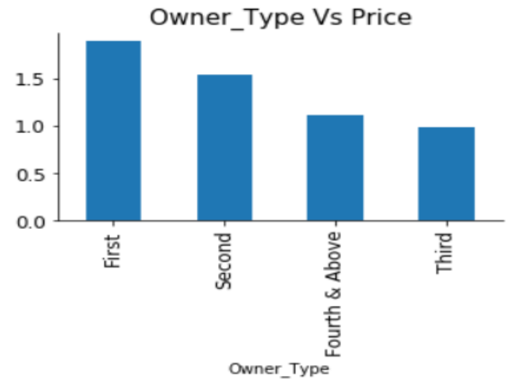
- The variable Year has a positive correlation with price and mileage
- A year has a Negative correlation with kilometres-Driven which mean with inc. in age of car performance decreases
- Mileage is negatively correlated with Power, As power increases, mileage decreases
- Car with recent make is higher at prices. As the age of the car increases price decreases
- Engine and Power increase, and the price of the car increases
- As engine inc. Power against increase.

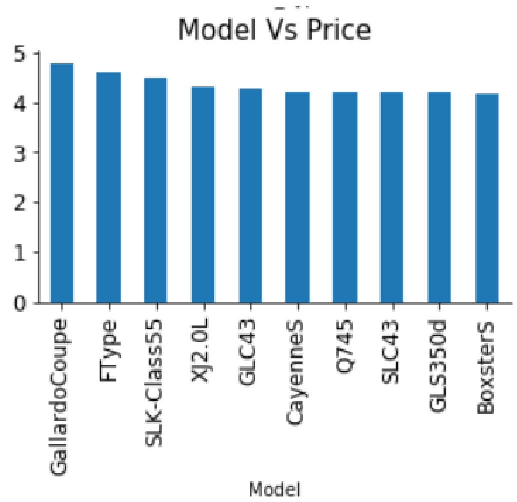
Analysis relation between categorical variable with price:

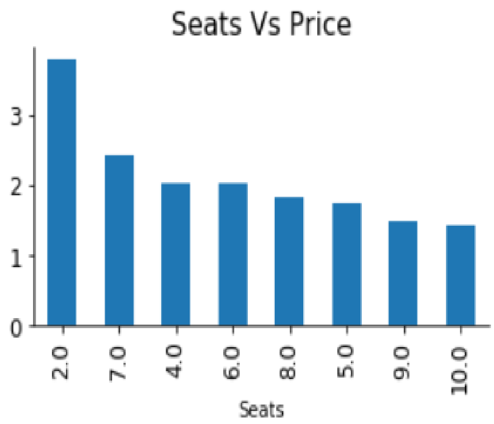


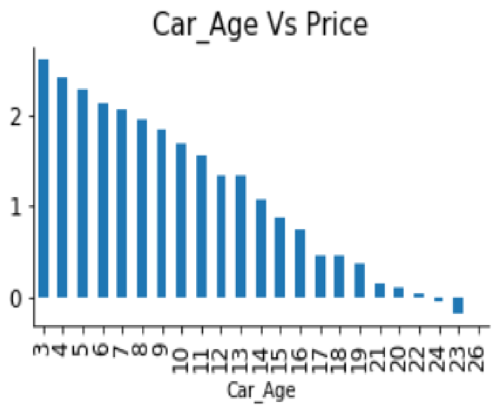










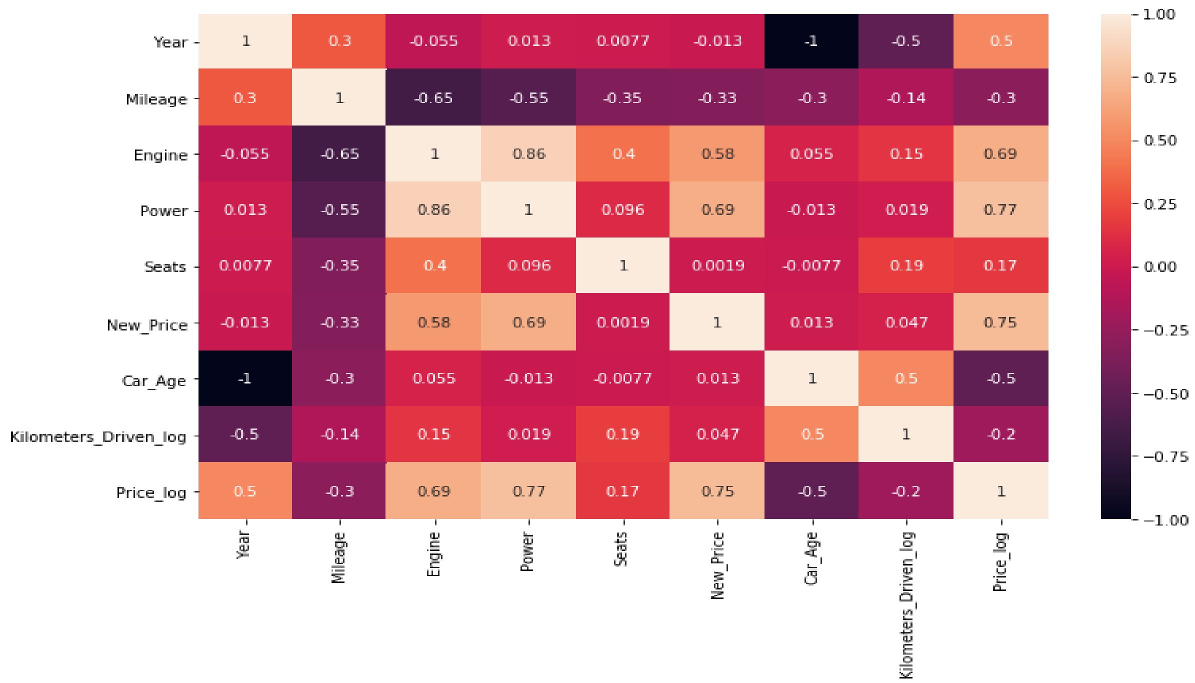


Observation:

- The price of cars is high in Coimbatore, Bangalore and less price in Kolkata and Jaipur
- Automatic cars have more price than manual cars.
- Electric cars price is maximum which follow by diesel, and LPG cars have the lowest price
- First-owner cars are higher in price, followed by a second, The third owner’s price is lesser than the Fourth and above
- Lamborghini brand is the highest in price
- Gallardocoupe Model is the highest in price
- 2 Seater has the highest price followed by 7 Seater
- The latest model cars are high in price.

3.5. Multivariate Analysis

Multivariate analysis looks at more than two variables using heatmap, Heat Map gives the correlation between the variables, whether it has a positive or negative correlation.



Observed Insights from heatmap;

- The engine has a strong positive correlation to Power 0.86
- Price has a positive correlation to Engine 0.69 as well Power 0.77
- Mileage has correlated to Engine, Power, and Price negatively
- Price is moderately positive in correlation to year.
- Kilometre driven has a negative correlation to year not much impact on the price
- Car age has a negative correlation with Price
- car Age is positively correlated to Kilometres-Driven as the Age of the car increases; then the kilometre will also increase of car has a negative correlation with Mileage this makes sense

4. Missing Value Imputation

Checking missing value first;

In [39]:
```
data_car.isnull().sum()
```

Out[39]:
```
Name                      0
Location                  0
Year                      0
Kilometers_Driven         0
Fuel_Type                 0
Transmission              0
Owner_Type                0
Mileage                  83
Engine                   46
Power                   175
Seats                    54
New_Price              6246
Price                  1233
Car_Age                   0
Brand                     0
Model                     0
Kilometers_Driven_log     0
Price_log              1233
dtype: int64
```

Here we used median value of respective variable for imputing missing value.

In [40]:
```
data_car["Seats"].fillna(data_car["Seats"].median(),inplace=True)
```

In [41]:
```
data_car["Mileage"].fillna(data_car["Mileage"].median(),inplace=True)
```

In [42]:
```
data_car["Engine"].fillna(data_car["Engine"].median(),inplace=True)
data_car["Power"].fillna(data_car["Power"].median(),inplace=True)
data_car["Seats"].fillna(data_car["Seats"].median(),inplace=True)
data_car["Price"].fillna(data_car["Price"].median(),inplace=True)
data_car["New_Price"].fillna(data_car["New_Price"].median(),inplace=True)
data_car["Price_log"].fillna(data_car["Price_log"].median(),inplace=True)
```

Now lets check whether all the missing values are filled in the dataset.

In [43]:
```
data_car.isnull().sum(axis=0)
```

Out[43]:
```
Name                     0
Location                 0
Year                     0
Kilometers_Driven        0
Fuel_Type                0
Transmission             0
Owner_Type               0
Mileage                  0
Engine                   0
Power                    0
Seats                    0
New_Price                0
Price                    0
Car_Age                  0
Brand                    0
Model                    0
Kilometers_Driven_log    0
Price_log                0
dtype: int64
```

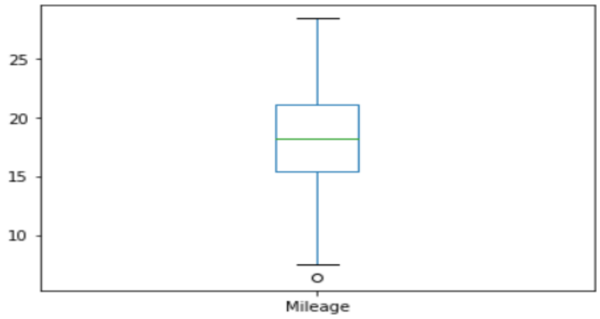
5. Outlier Treatment

For Mileage

```
In [46]: # treating mileage column
         data_car.loc[data_car['Mileage']>30,'Mileage']=np.mean(data_car['Mileage'])

In [47]: data_car['Mileage'].plot.box()

Out[47]: <matplotlib.axes._subplots.AxesSubplot at 0x13bcf781908>
```

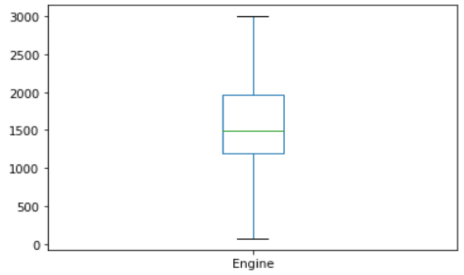


For Engine

```
In [46]: # Treating Engine Column
         data_car.loc[data_car["Engine"]>3000,"Engine"]=np.mean(data_car["Engine"])

In [47]: data_car['Engine'].plot.box()

Out[47]: <matplotlib.axes._subplots.AxesSubplot at 0x204002a8708>
```

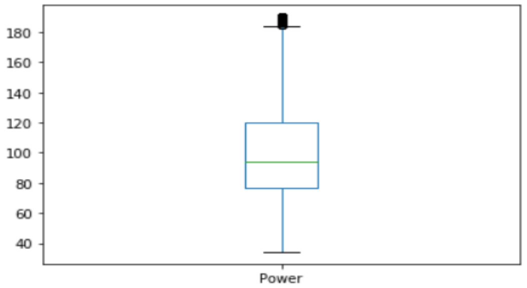


For Power

```
In [48]: # Treating Power Column
         data_car.loc[data_car["Power"]>190,"Power"]=np.mean(data_car["Power"])

In [49]: data_car["Power"].plot.box()

Out[49]: <matplotlib.axes._subplots.AxesSubplot at 0x204002cbd08>
```

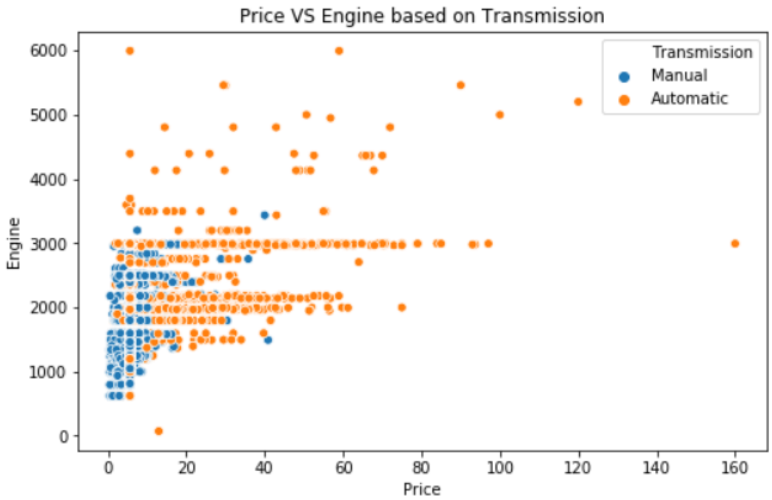


6. Analysing relation for two, three variables with price

In [64]:
```
# understand relation ship of Engine vs Price and Transmimssion
plt.figure(figsize=(8,5))

plt.title("Price VS Engine based on Transmission")
sns.scatterplot(y='Engine', x='Price', hue='Transmission', data=data_car)
```

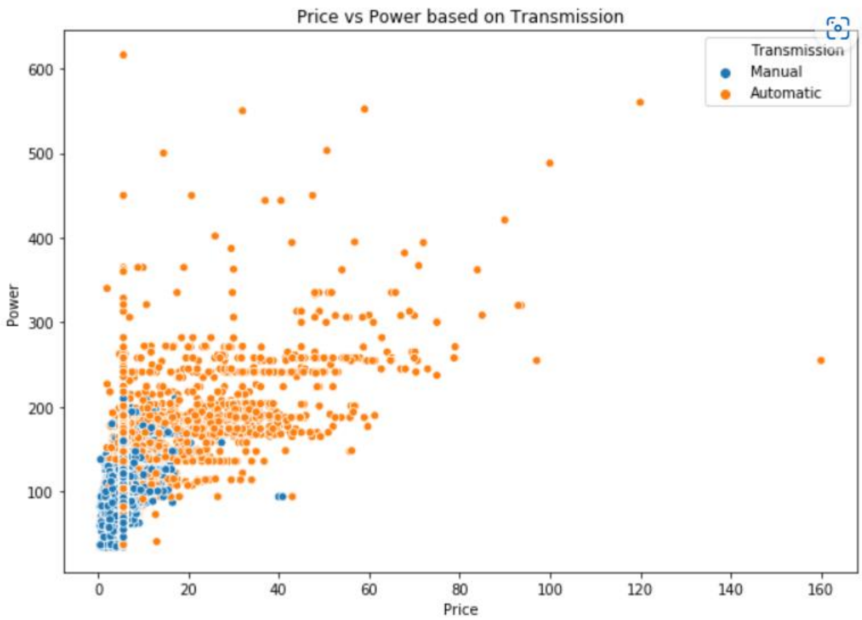
Out[64]: <matplotlib.axes._subplots.AxesSubplot at 0x1ae9df50748>



Price Vs Power vs Transmission

In [65]:
```
#understand relationship betweem Price and Power
plt.figure(figsize=10,7))
plt.title("Price vs Power based on Transmission")
sns.scatterplot(y='Power', x='Price', hue='Transmission', data=data_car)
```

Out[65]: <matplotlib.axes._subplots.AxesSubplot at 0x1ae9dfcbe88>

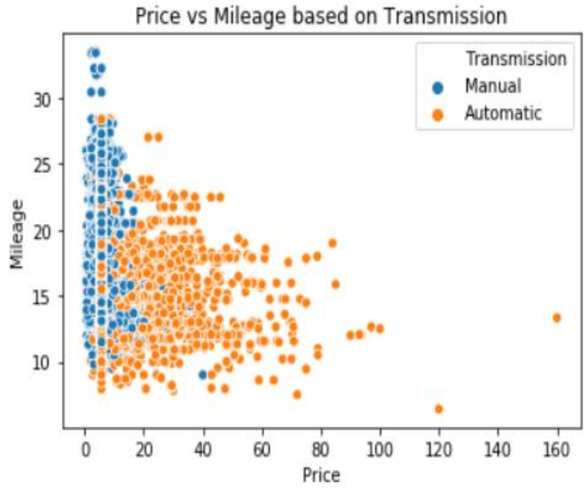


Price Vs Mileage Vs Transmission

In [70]:
```
# Understand the relationships  between mileage and Price

plt.title("Price vs Mileage based on Transmission")
sns.scatterplot(y='Mileage', x='Price', hue='Transmission', data=data_car)
```

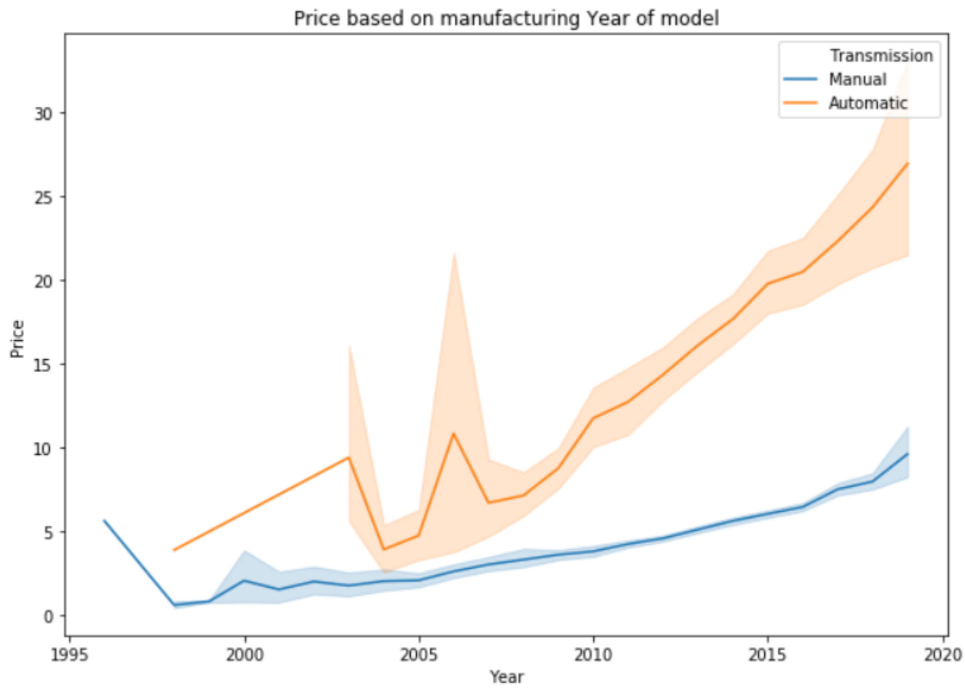
Out[70]: <matplotlib.axes._subplots.AxesSubplot at 0x1ae9cc20b48>



Price Vs Year Vs Transmission

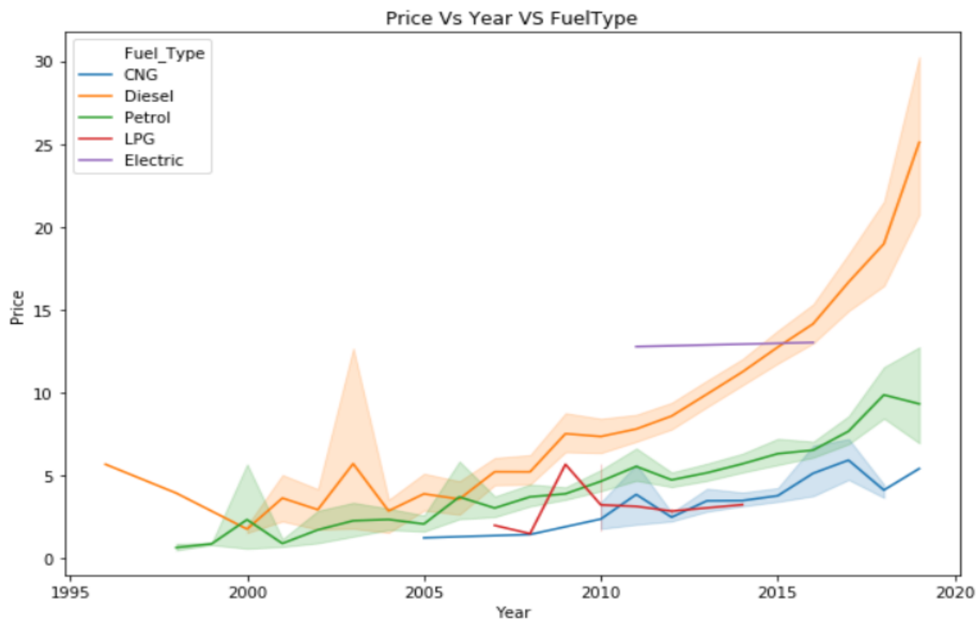
In [71]:
```
# Impact of years on price
plt.figure(figsize=(10,7))
plt.title("Price based on manufacturing Year of model")
sns.lineplot(x='Year', y='Price',hue='Transmission', data=data_car)
```

Out[71]: <matplotlib.axes._subplots.AxesSubplot at 0x1ae9e2c9148>



```
In [72]: # Impact of years on price
         plt.figure(figsize=(10,7))
         plt.title("Price Vs Year VS FuelType")
         sns.lineplot(x='Year', y='Price',hue='Fuel_Type', data=data_car)
```

```
Out[72]: <matplotlib.axes._subplots.AxesSubplot at 0x1ae9e066108>
```



Insights based on EDA

- Expensive cars are in Coimbatore and Bangalore.
- 2-Seater cars are more expensive.
- Here Diesel Fuel type car are more expensive compared to other fuel type.
- As expected, older model are sold cheaper compared to latest model
- Automatic transmission vehicle have a higher price than manual transmission vehicles.
- Vehicles with more engine capacity have higher prices.
- Customers prefer to purchase the First owner rather than the Second or Third.
- Automatic transmission require high engine and power.
- Prices for Cars with fuel type as Diesel has increased with recent models
- Engine, Power, how old the car his, Mileage, Fuel type, location, Transmission effect the price.

7. Model Building

7.1. Model 1

Dropping some variable

```
In [54]: # droping column which not be used in model building as keeping some feature can create lot of du
         data_pred.drop(['Name','Model','Year','Brand','New_Price'],axis=1,inplace=True)
```

Creating Train and test of our data set, later on creating dummies variable for categorical data as model can process only when there is numerical data.

```
X= data_pred.drop(['Price','Price_log'],axis=1)
y = data_pred[['Price','Price_log']]
```

Creating dummy variables

```
def encode_cat_vars(x):
    x = pd.get_dummies(
        x,
        columns=x.select_dtypes(include=["object", "category"]).columns.tolist(),
        drop_first=True,
    )
    return x
```

Splitting in train and test dataset

```
: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
X_train.reset_index()
print("X_train:",X_train.shape)
print("X_test:",X_test.shape)
print("y_train:",y_train.shape)
print("y_test:",y_test.shape)
```

```
X_train: (5076, 25)
X_test: (2176, 25)
y_train: (5076, 2)
y_test: (2176, 2)
```

Using statsmodel api and fit model over it to get summary regression results

```
In [61]: # Statsmodel api does not add a constant by default. We need to add it explicitly.
         import statsmodels.api as sm
         X_train = sm.add_constant(X_train)
         # Add constant to test data
         X_test = sm.add_constant(X_test)


         def build_ols_model(train):
             # Create the model
             olsmodel = sm.OLS(y_train["Price_log"], train)
             return olsmodel.fit()
```

```
In [62]: #fit statmodel
         olsmodel1 = build_ols_model(X_train)
         print(olsmodel1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              Price_log   R-squared:                       0.709
Model:                            OLS   Adj. R-squared:                  0.707
Method:                 Least Squares   F-statistic:                     491.2
Date:                Tue, 29 Nov 2022   Prob (F-statistic):               0.00
Time:                        20:03:37   Log-Likelihood:                -2878.6
No. Observations:                5076   AIC:                             5809.
Df Residuals:                    5050   BIC:                             5979.
Df Model:                          25
Covariance Type:            nonrobust
------------------------------------------------------------------------------------------------
```

```
==============================================================================================
                               coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------------------
const                        2.9944      0.200     14.964      0.000       2.602       3.387
Kilometers_Driven        -7.556e-07   2.87e-07     -2.629      0.009   -1.32e-06   -1.92e-07
Mileage                     -0.0167      0.003     -6.195      0.000      -0.022      -0.011
Engine                       0.0005   2.28e-05     22.834      0.000       0.000       0.001
Power                        0.0033      0.000     13.113      0.000       0.003       0.004
Seats                       -0.0857      0.009     -9.098      0.000      -0.104      -0.067
Car_Age                     -0.0979      0.003    -36.367      0.000      -0.103      -0.093
Kilometers_Driven_log       -0.0501      0.016     -3.065      0.002      -0.082      -0.018
Location_Bangalore           0.0901      0.040      2.250      0.025       0.012       0.169
Location_Chennai             0.0008      0.038      0.021      0.983      -0.074       0.076
Location_Coimbatore          0.0958      0.037      2.602      0.009       0.024       0.168
Location_Delhi              -0.0668      0.037     -1.792      0.073      -0.140       0.006
Location_Hyderabad           0.0790      0.036      2.184      0.029       0.008       0.150
Location_Jaipur             -0.0549      0.039     -1.396      0.163      -0.132       0.022
Location_Kochi              -0.0182      0.037     -0.492      0.623      -0.091       0.054
Location_Kolkata            -0.1832      0.038     -4.872      0.000      -0.257      -0.109
Location_Mumbai             -0.0506      0.036     -1.409      0.159      -0.121       0.020
Location_Pune               -0.0516      0.037     -1.394      0.163      -0.124       0.021
Fuel_Type_Diesel             0.1838      0.065      2.809      0.005       0.056       0.312
Fuel_Type_Electric           1.0492      0.310      3.386      0.001       0.442       1.657
Fuel_Type_LPG                0.0823      0.150      0.549      0.583      -0.212       0.376
Fuel_Type_Petrol            -0.0296      0.064     -0.459      0.646      -0.156       0.097
Transmission_Manual         -0.3269      0.018    -18.286      0.000      -0.362      -0.292
Owner_Type_Fourth & Above    0.3182      0.163      1.952      0.051      -0.001       0.638
Owner_Type_Second           -0.0593      0.018     -3.301      0.001      -0.094      -0.024
Owner_Type_Third            -0.1887      0.047     -4.034      0.000      -0.280      -0.097
==============================================================================
Omnibus:                      324.814   Durbin-Watson:                   1.970
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1510.420
Skew:                          -0.040   Prob(JB):                         0.00
Kurtosis:                       5.671   Cond. No.                     3.59e+06
==============================================================================
```

- Both the R-squared and Adjusted R squared of our model are Above average. This is a clear indication that we have been able to create a good model that is able to explain variance in price of used cars for up to 71%

Checking performance of test data

```python
# Checking model performance
model_pref(olsmodel1, X_train, X_test)
```

	Data	RMSE	MAE	MAPE
0	Train	6.992667	3.145990	33.663252
1	Test	7.182994	3.108323	33.607426

- Root Mean Squared Error of train and test data is not different, indicating that our model is not overfitting the train data.
- Mean Absolute Error indicates that our current model is able to predict used cars prices within mean error of 3 lakhs(app.) on test data.
- The units of both RMSE and MAE are same - Lakhs in this case. But RMSE is greater than MAE because it peanalises the outliers more.
- Mean Absolute Percentage Error is ~34% on the test data.

Fitting data in linear regression model for predicting test value (model score -46%)

```
In [64]: lreg = LinearRegression()

In [65]: lreg.fit(X_train,y_train)
Out[65]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

In [66]: lreg.predict(X_test)
Out[66]: array([[ 2.21152777,  1.19439399],
               [ 5.28827439,  1.60275052],
               [11.01895654,  1.79375985],
               ...,
               [ 9.66324963,  1.94362444],
               [ 6.00609491,  1.65537922],
               [ 2.08435939,  1.42042433]])
```

7.2. Model 2

Dropping lesser no. of variable for model

In [71]:
```
data_p.drop(['Name','Year','New_Price'],axis=1,inplace=True)
```

In [72]:
```
data_p.drop(['Model'],axis=1,inplace=True)
```

Creating dummy and splitting data set

In [83]:
```
X2_train, X2_test, y2_train, y2_test = train_test_split(X2, y2, test_size=0.3, random_state=42)
X_train.reset_index()
print("X2_train:",X2_train.shape)
print("X2_test:",X2_test.shape)
print("y2_train:",y2_train.shape)
print("y2_test:",y2_test.shape)
```

```
X2_train: (5076, 56)
X2_test: (2176, 56)
y2_train: (5076, 2)
y2_test: (2176, 2)
```

Fitting model and predicting score

In [84]:
```
reg = LinearRegression()
```

In [85]:
```
reg.fit(X2_train,y2_train)
```

Out[85]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [88]:
```
# Predict X2 test result
reg.predict(X2_test)
```

Out[88]:
```
array([[ 2.89698806,  1.26817036],
       [ 6.15056432,  1.63330306],
       [16.01900155,  2.12944412],
       ...,
       [ 8.62030878,  1.9932958 ],
       [ 5.42577495,  1.66783093],
       [ 4.42842159,  1.52961865]])
```

In [89]:
```
# Model Score
reg.score(X2_test,y2_test)
```

Out[89]:
```
0.584083600158599
```

7.3. Model 3

Dropping column and splitting dataset

In [93]:
```
data_pr.drop(['Name','Year','New_Price'],axis=1,inplace=True)
```

```
In [98]: # splitting dataset
X3_train, X3_test, y3_train, y3_test = train_test_split(X3, y3, test_size=0.3, random_state=42)
X3_train.reset_index()
print("X3_train:",X3_train.shape)
print("X3_test:",X3_test.shape)
print("y3_train:",y3_train.shape)
print("y3_test:",y3_test.shape)

X3_train: (5076, 781)
X3_test: (2176, 781)
y3_train: (5076, 2)
y3_test: (2176, 2)
```

Fitting model and Predicting score(model-64%)

```
In [99]: lrg =LinearRegression()

In [100]: lrg.fit(X3_train,y3_train)

Out[100]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

In [103]: lrg.predict(X3_test)

Out[103]: array([[ 1.82684512,  1.28713415],
       [ 5.65987037,  1.77158843],
       [14.71710425,  2.56415784],
       ...,
       [ 5.93599662,  1.81032671],
       [ 6.98368606,  1.87414187],
       [ 5.9562716 ,  1.30299947]])

In [104]: lrg.score(X3_test,y3_test)

Out[104]: 0.6439106592573801
```

OLS Regression summary

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              Price_log   R-squared:                       0.835
Model:                            OLS   Adj. R-squared:                  0.809
Method:                 Least Squares   F-statistic:                     32.80
Date:                Tue, 29 Nov 2022   Prob (F-statistic):               0.00
Time:                        20:09:02   Log-Likelihood:                -1440.1
No. Observations:                5076   AIC:                             4236.
Df Residuals:                    4398   BIC:                             8665.
Df Model:                         677
Covariance Type:            nonrobust
```

- Both the R-squared and Adjusted R squared of our model are very high. This is a clear indication that we have been able to create a very good model that is able to explain variance in price of used cars for upto 83%
- The model is not an underfitting or overfitting model.

Checking performance of test data

```python
# Checking model performance
model_pref(olsmodel1, X3_train, X3_test)
```

	Data	RMSE	MAE	MAPE
0	Train	4.742457	2.020217	23.057994
1	Test	6.265627	2.489694	29.699576

- Root Mean Squared Error of train and test data is not different, indicating that our model is not overfitting the train data.
- Mean Absolute Error indicates that our current model is able to predict used cars prices within mean error of 2.4 lakhs on test data.
- The units of both RMSE and MAE are same - Lakhs in this case. But RMSE is greater than MAE because it peanalises the outliers more.
- Mean Absolute Percentage Error is ~29% on the test data.

Final Observation from models;

1. After observing all 3 models we came know that the performance of model improves as when number of variables are more, especially including categorical variable.
2. Compare to 1st model in 3rd model number of variable column inc. from 56 to 781 in total
3. In model 3 we have captured ~83% data from Linear regression model.
4. The model indicates that the most significant predictors of price of used cars are -

    - Age of the car
    - Number of seats in the car
    - Power of the engine
    - Mileage
    - Kilometres Driven
    - Location
    - Fuel Type
    - Owner Type
    - Model
    - Brand
    - Transmission - Automatic/Manual

5. In finale model Both the R-squared and Adjusted R squared of our model are very high. This is a clear indication that we have been able to create a very good model that is able to explain variance in price of used cars for upto 83%
6. Model performance firstly Root Mean Squared Error of train and test data is not different, indicating that our model is not overfitting the train data.
7. Mean Absolute Error indicates that our current model is able to predict used cars prices within mean error of 2.4 lakhs on test data.
8. The units of both RMSE and MAE are same - Lakhs in this case. But RMSE is greater than MAE because it peanalises the outliers more.
9. Mean Absolute Percentage Error is ~29% on the test data.
10. Model 3 score predicted for test variable - 0.64 which about 0.47 for model1.

8. Recommendations

- Our final Linear Regression model has a MAPE of 29% on the test data, which means that we are able to predict within 23% of the price value. This is a very good model but can be further improved.
- Some southern markets tend to have higher prices. It might be a good strategy to plan growth in southern cities using this information.
- Markets like Kolkata (coeff = -0.1832 to -0.16) are very risky and we need to be careful about investments in this area.
- Based on Analysis, we can to divide our cars into 3 segment Low, Medium and High budget.
- Brands like Maruti, Hyundai, Honda are low budget and very popular brands in used car market.
- Brands like BMW, Bentley, Jaguar, Land Rover, Mercedes Benz,Porche,Mini Cooper are high budget cars and are mostly bought by car enthusiast who are ready to buy a two user owned car at higher price as well.
- Brands like Toyota, Volvo can be medium budget cars.
- Automatic transmission car earns more profit, as these cars sell for higher prices.
- With Increasing petrol rates diesel car are in more demand in recent years, acquiring and selling them can high profits
- We can provide Car maintenance packages where customers pays a small upfront fees and can bring the car for servicing anytime in a year to attract more customers.

9. References of work:

1. Kaggle data set- used cars data
2. Blog- Step by step Exploratory Data Analysis using Python https://www.analyticsvidhya.com/blog/2022/07/step-by-step-exploratory-data-analysis-eda-using-python/
3. Blog - https://jovian.ai/rayankazi/eda-used-cars#C1

4. Project reference file on predicting house price