Import Python Libraries

Pandas and Numpy have been used for Data Manipulation and numerical Calculations

Matplotlib and Seaborn have been used for Data visualizations.

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
#to ignore warnings
import warnings
warnings.filterwarnings('ignore')
```

Reading Dataset

In [2]:
```python
data = pd.read_csv(r'C:\Users\Vineeta Shrivastava\Downloads\cars4u\used_cars_data.csv')
```

In [3]:
```python
#Analyzing the Data
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]:
```python
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

In [5]:
```python
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
```

Check for Duplication

In [6]:
```python
data.nunique()
```

Out[6]:
```
S.No.                7253
Name                 2041
Location               11
Year                   23
Kilometers_Driven    3660
Fuel_Type               5
Transmission            2
Owner_Type              4
Mileage               450
Engine                150
Power                 386
Seats                   9
New_Price             625
Price                1373
dtype: int64
```

Missing Values Calculation

In [7]:
```python
data.isnull().sum()
```

Out[7]:
```
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]:
```python
(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
```

Data Reduction

In [9]:
```python
# Remove S.No. column from data
data = data.drop(['S.No.'], axis = 1)
data.info()
```

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

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 modeling. The main goal of Feature engineering is to create meaningful data from raw data.

Creating Features

variables Year and Name in our dataset. If we see the sample data, the column “Year” shows the manufacturing year of the car.It would be difficult to find the car’s age if it is in year format as the Age of the car is a contributing factor to Car Price. Introducing a new column, “Car_Age” to know the age of the car

In [10]:
```python
from datetime import date
date.today().year
data['Car_Age']=date.today().year-data['Year']
data.head()
```

Out[10]:

	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	26.6 km/kg	998 CC	58.16 bhp	5.0	NaN	1.75	14
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	9
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	13
3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First	20.77 kmpl	1248 CC	88.76 bhp	7.0	NaN	6.00	12
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	11

Since car names will not be great predictors of the price in our current data. But we can process this column to extract important information using brand and Model names. Let’s split the name and introduce new variables “Brand” and “Model”

In [11]:
```python
data['Brand'] = data.Name.str.split().str.get(0)
data['Model'] = data.Name.str.split().str.get(1) + data.Name.str.split().str.get(2)
data[['Name','Brand','Model']]
```

Out[11]:

	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

Data Cleaning/Wrangling

In [12]:
```python
#Some names of the variables are not relevant and not easy to understand. Some data may have data entry errors, and some variables may need data type conversion. We need to fix this issue in the data.In the example, The brand nam
```

In [13]:
```python
print(data.Brand.unique())
print(data.Brand.nunique())
```

```
['Maruti' 'Hyundai' 'Honda' 'Audi' 'Nissan' 'Toyota' 'Volkswagen' 'Tata'
 'Land' 'Mitsubishi' 'Renault' 'Mercedes-Benz' 'BMW' 'Mahindra' 'Ford'
 'Porsche' 'Datsun' 'Jaguar' 'Volvo' 'Chevrolet' 'Skoda' 'Mini' 'Fiat'
 'Jeep' 'Smart' 'Ambassador' 'Isuzu' 'ISUZU' 'Force' 'Bentley'
 'Lamborghini' 'Hindustan' 'OpelCorsa']
33
```

In [14]:
```python
searchfor = ['Isuzu' ,'ISUZU','Mini','Land']
data[data.Brand.str.contains('|'.join(searchfor))].head(5)
```

Out[14]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileage	Engine	Power	Seats	New_Price	Price	Car_Age	Brand	Model
13	Land Rover Range Rover 2.2L Pure	Delhi	2014	72000	Diesel	Automatic	First	12.7 kmpl	2179 CC	187.7 bhp	5.0	NaN	27.00	10	Land	RoverRange
14	Land Rover Freelander 2 TD4 SE	Pune	2012	85000	Diesel	Automatic	Second	0.0 kmpl	2179 CC	115 bhp	5.0	NaN	17.50	12	Land	RoverFreelander
176	Mini Countryman Cooper D	Jaipur	2017	8525	Diesel	Automatic	Second	16.6 kmpl	1998 CC	112 bhp	5.0	NaN	23.00	7	Mini	CountrymanCooper
191	Land Rover Range Rover 2.2L Dynamic	Coimbatore	2018	36091	Diesel	Automatic	First	12.7 kmpl	2179 CC	187.7 bhp	5.0	NaN	55.76	6	Land	RoverRange
228	Mini Cooper Convertible S	Kochi	2017	26327	Petrol	Automatic	First	16.82 kmpl	1998 CC	189.08 bhp	4.0	44.28 Lakh	35.67	7	Mini	CooperConvertible

In [15]:
```python
data["Brand"].replace({"ISUZU": "Isuzu", "Mini": "Mini Cooper","Land":"Land Rover"}, inplace=True)
```

We have done the fundamental data analysis, Featuring, and data clean-up. Let’s move to the EDA process

Our Data is ready to perform EDA.

EDA Exploratory Data Analysis

Exploratory Data Analysis refers to the crucial process of performing initial investigations on data to discover patterns to check assumptions with the help of summary statistics and graphical representations.

EDA can be leveraged to check for outliers, patterns, and trends in the given data. EDA helps to find meaningful patterns in data. EDA provides in-depth insights into the data sets to solve our business problems. EDA gives a clue to impute missing values in the dataset

Statistics Summary

In [18]:
```python
data.describe().T
```

Out[18]:

	count	mean	std	min	25%	50%	75%	max
Year	7253.0	2013.365366	3.254421	1996.00	2011.0	2014.00	2016.00	2019.0
Kilometers_Driven	7253.0	58699.063146	84427.720583	171.00	34000.0	53416.00	73000.00	6500000.0
Seats	7200.0	5.279722	0.811660	0.00	5.0	5.00	5.00	10.0
Price	6019.0	9.479468	11.187917	0.44	3.5	5.64	9.95	160.0
Car_Age	7253.0	10.634634	3.254421	5.00	8.0	10.00	13.00	28.0

In [19]:
```python
#describe(include='all') provides a statistics summary of all data, include object, category etc
```

In [20]:
```python
data.describe(include='all').T
```

Out[20]:

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
Name	7253	2041	Mahindra XUV500 W8 2WD	55	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Location	7253	11	Mumbai	949	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Year	7253	NaN	NaN	NaN	2013.37	3.25442	1996	2011	2014	2016	2019
Kilometers_Driven	7253	NaN	NaN	NaN	58699.1	84427.7	171	34000	53416	73000	6.5e+06
Fuel_Type	7253	5	Diesel	3852	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Transmission	7253	2	Manual	5204	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Owner_Type	7253	4	First	5952	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Mileage	7251	450	17.0 kmpl	207	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Engine	7207	150	1197 CC	732	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Power	7207	386	74 bhp	280	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Seats	7200	NaN	NaN	NaN	5.27972	0.81166	0	5	5	5	10
New_Price	1006	625	63.71 Lakh	6	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Price	6019	NaN	NaN	NaN	9.47947	11.1879	0.44	3.5	5.64	9.95	160
Car_Age	7253	NaN	NaN	NaN	10.6346	3.25442	5	8	10	13	28
Brand	7253	32	Maruti	1444	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Model	7252	726	SwiftDzire	189	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Before we do EDA, lets separate Numerical and categorical variables for easy analysis

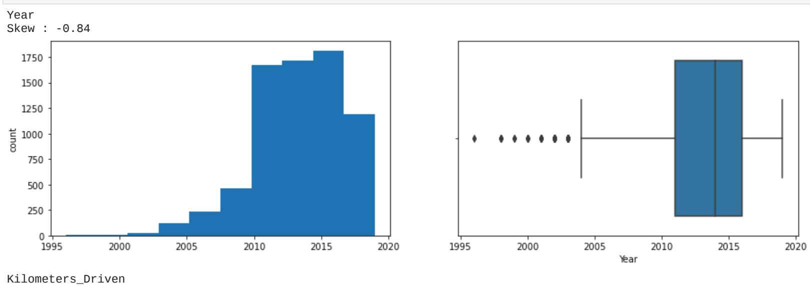
In [21]:
```python
cat_cols=data.select_dtypes(include=['object']).columns
num_cols = data.select_dtypes(include=np.number).columns.tolist()
print("Categorical Variables:")
print(cat_cols)
print("Numerical Variables:")
print(num_cols)
```

```
Categorical Variables:
Index(['Name', 'Location', 'Fuel_Type', 'Transmission', 'Owner_Type',
       'Mileage', 'Engine', 'Power', 'New_Price', 'Brand', 'Model'],
      dtype='object')
Numerical Variables:
['Year', 'Kilometers_Driven', 'Seats', 'Price', 'Car_Age']
```

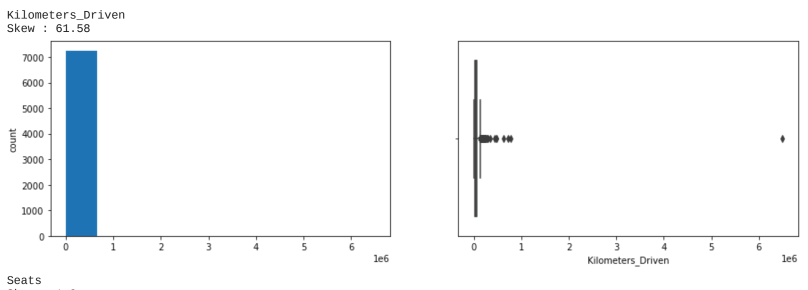
EDA Univariate Analysis

In [22]:
```python
for col in num_cols:
    print(col)
    print('Skew :', round(data[col].skew(), 2))
    plt.figure(figsize = (15, 4))
    plt.subplot(1, 2, 1)
    data[col].hist(grid=False)
    plt.ylabel('count')
    plt.subplot(1, 2, 2)
    sns.boxplot(x=data[col])
    plt.show()
```

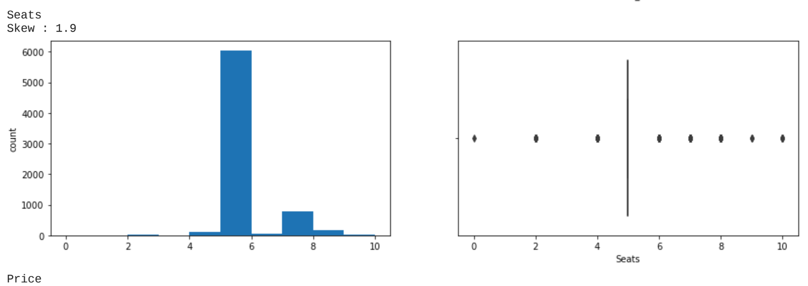
```
Year
Skew : -0.84
```



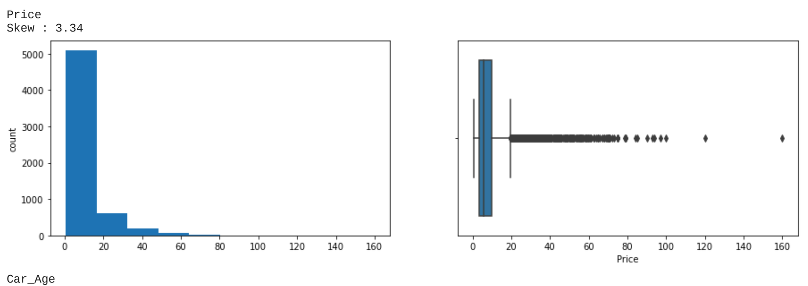

```
Kilometers_Driven
Skew : 61.58
```



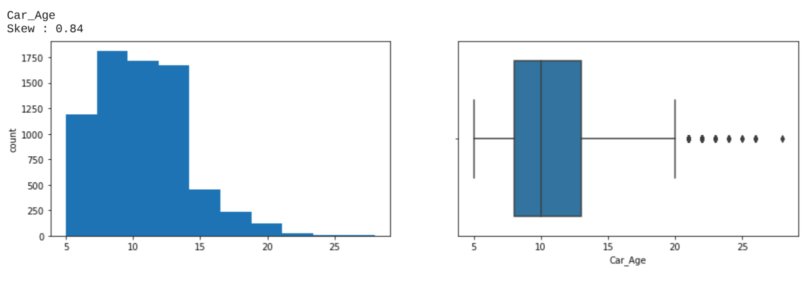

```
Seats
Skew : 1.9
```




```
Price
Skew : 3.34
```

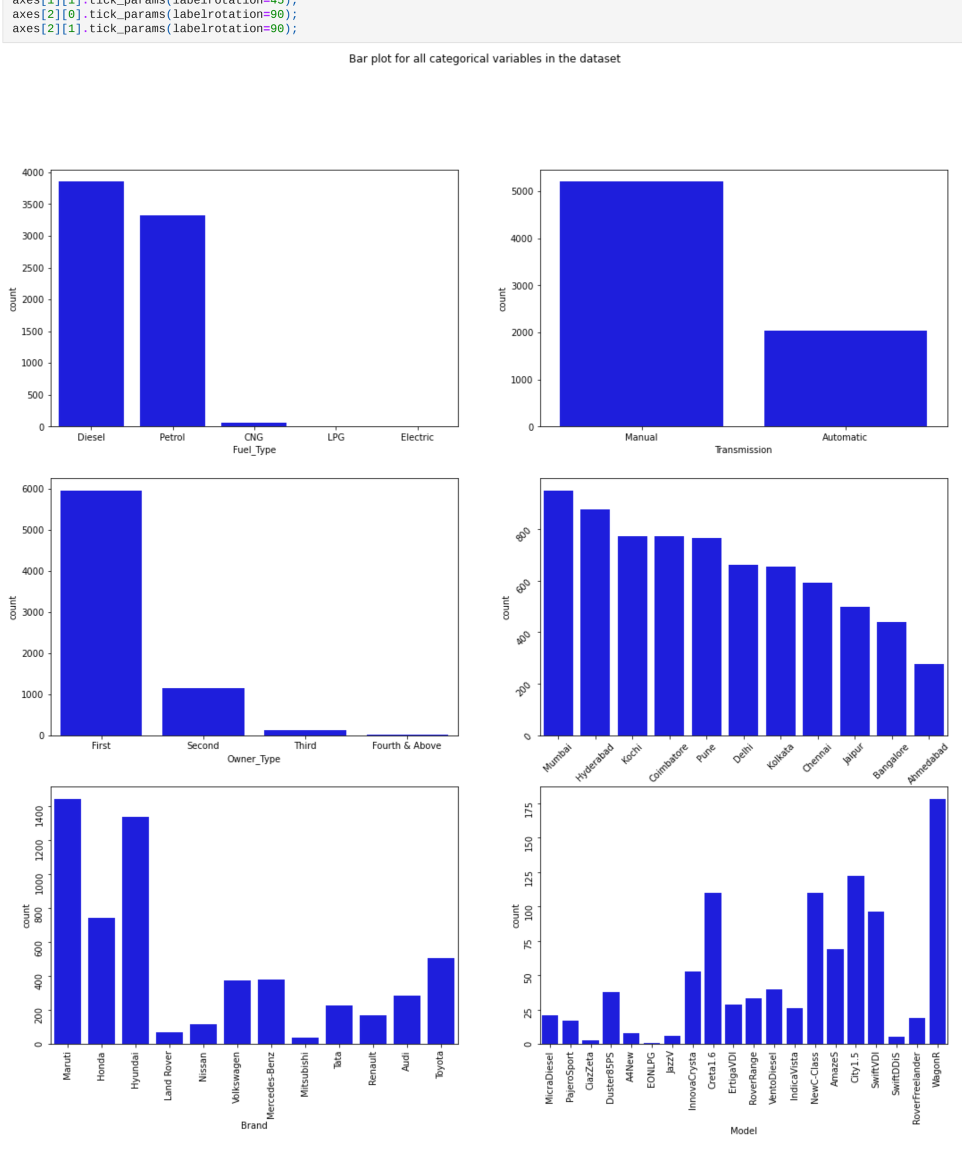



```
Car_Age
Skew : 0.84
```



categorical variables are being visualized using a count plot. Categorical variables provide the pattern of factors influencing car price

In [23]:
```python
fig, axes = plt.subplots(3, 2, figsize = (18, 18))
fig.suptitle('Bar plot for all categorical variables in the dataset')
sns.countplot(ax = axes[0, 0], x = 'Fuel_Type', data = data, color = 'blue',
              order = data['Fuel_Type'].value_counts().index);
sns.countplot(ax = axes[0, 1], x = 'Transmission', data = data, color = 'blue',
              order = data['Transmission'].value_counts().index);
sns.countplot(ax = axes[1, 0], x = 'Owner_Type', data = data, color = 'blue',
              order = data['Owner_Type'].value_counts().index);
sns.countplot(ax = axes[1, 1], x = 'Location', data = data, color = 'blue',
              order = data['Location'].value_counts().index);
sns.countplot(ax = axes[2, 0], x = 'Brand', data = data, color = 'blue',
              order = data['Brand'].head(20).value_counts().index);
sns.countplot(ax = axes[2, 1], x = 'Model', data = data, color = 'blue',
              order = data['Model'].head(20).value_counts().index);
axes[1][1].tick_params(labelrotation=45);
axes[2][0].tick_params(labelrotation=90);
axes[2][1].tick_params(labelrotation=90);
```



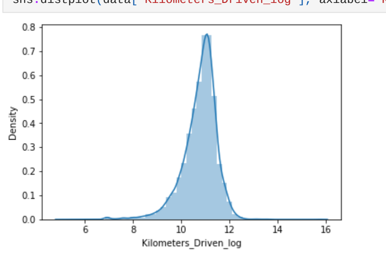
Data Transformation

In [26]:
```python
# Function for log transformation of the column
def log_transform(data,col):
    for colname in col:
        if (data[colname] == 1.0).all():
            data[colname + '_log'] = np.log(data[colname]+1)
        else:
            data[colname + '_log'] = np.log(data[colname])
    data.info()
```

In [27]:
```python
log_transform(data,['Kilometers_Driven','Price'])
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7253 entries, 0 to 7252
Data columns (total 18 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
 13  Car_Age                7253 non-null   int64
 14  Brand                  7253 non-null   object
 15  Model                  7252 non-null   object
 16  Kilometers_Driven_log  7253 non-null   float64
 17  Price_log              6019 non-null   float64
dtypes: float64(4), int64(3), object(11)
memory usage: 1020.1+ KB
```

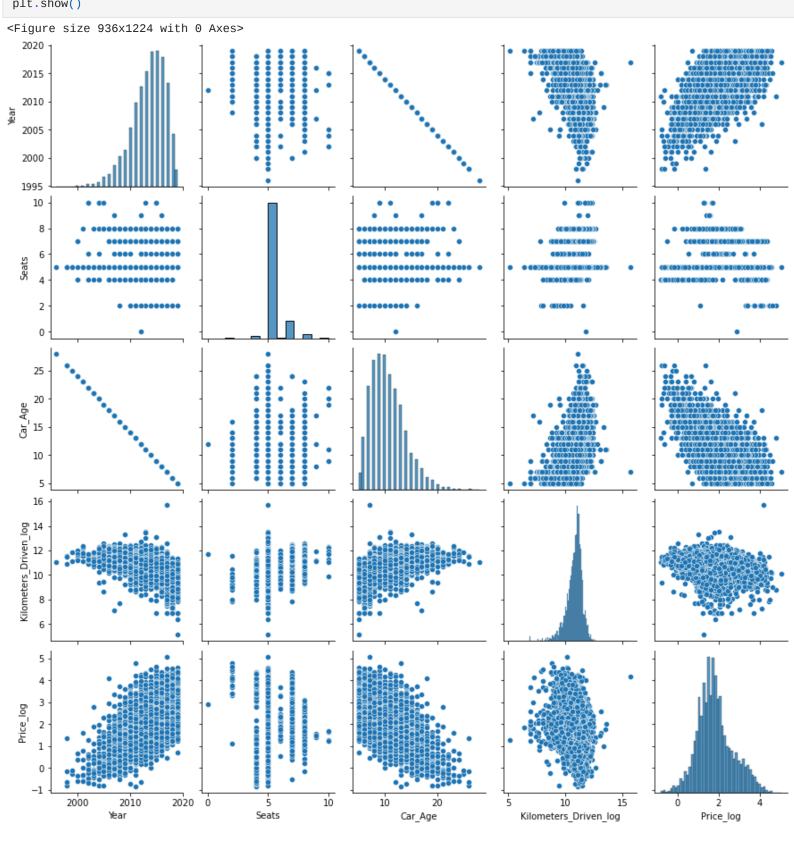
In [28]:
```python
#Log transformation of the feature 'Kilometers_Driven'
sns.distplot(data["Kilometers_Driven_log"], axlabel="Kilometers_Driven_log");
```



EDA Bivariate Analysis

In [29]:
```python
plt.figure(figsize=(13,17))
sns.pairplot(data=data.drop(['Kilometers_Driven','Price'],axis=1))
plt.show()
```

```
<Figure size 936x1224 with 0 Axes>
```



A bar plot can be used to show the relationship between Categorical variables and continuous variables

In [30]:
```python
fig, axarr = plt.subplots(4, 2, figsize=(12, 18))
data.groupby('Location')['Price_log'].mean().sort_values(ascending=False).plot.bar(ax=axarr[0][0], fontsize=12)
axarr[0][0].set_title("Location Vs Price", fontsize=18)
data.groupby('Transmission')['Price_log'].mean().sort_values(ascending=False).plot.bar(ax=axarr[0][1], fontsize=12)
axarr[0][1].set_title("Transmission Vs Price", fontsize=18)
data.groupby('Fuel_Type')['Price_log'].mean().sort_values(ascending=False).plot.bar(ax=axarr[1][0], fontsize=12)
axarr[1][0].set_title("Fuel_Type Vs Price", fontsize=18)
data.groupby('Owner_Type')['Price_log'].mean().sort_values(ascending=False).plot.bar(ax=axarr[1][1], fontsize=12)
axarr[1][1].set_title("Owner_Type Vs Price", fontsize=18)
data.groupby('Brand')['Price_log'].mean().sort_values(ascending=False).head(10).plot.bar(ax=axarr[2][0], fontsize=12)
axarr[2][0].set_title("Brand Vs Price", fontsize=18)
data.groupby('Model')['Price_log'].mean().sort_values(ascending=False).head(10).plot.bar(ax=axarr[2][1], fontsize=12)
axarr[2][1].set_title("Model Vs Price", fontsize=18)
data.groupby('Seats')['Price_log'].mean().sort_values(ascending=False).plot.bar(ax=axarr[3][0], fontsize=12)
axarr[3][0].set_title("Seats Vs Price", fontsize=18)
data.groupby('Car_Age')['Price_log'].mean().sort_values(ascending=False).plot.bar(ax=axarr[3][1], fontsize=12)
axarr[3][1].set_title("Car_Age Vs Price", fontsize=18)
plt.subplots_adjust(hspace=1.0)
plt.subplots_adjust(wspace=.5)
sns.despine()
```

EDA Multivariate Analysis

In [31]:
```python
#A heat map is widely been used for Multivariate Analysis
```

In [32]:
```python
plt.figure(figsize=(12, 7))
sns.heatmap(data.drop(['Kilometers_Driven','Price'],axis=1).corr(), annot = True, vmin = -1, vmax = 1)
plt.show()
```