NAME: ANADI MISHRA

BATCH: BATCH 3

GROUP: ML043B11

EMAIL ID : anadimishra231@gmail.com

DATA ANALYSIS AND PREDICTION OF THE COST OF USED CARS

First we need to import the needed libraries to perform the specific set of tasks.

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [3]:
```
import cufflinks as cf
from plotly.offline import download_plotlyjs,init_notebook_mode,plot,iplot
init_notebook_mode(connected = True)
cf.go_offline()
```

Plotly and cufflinks are used here to visualize more and to be clear.

In [4]:
```
expr = pd.read_excel('Data_Train.xlsx')
```

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

Out[5]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Mileag
0	Maruti Wagon R LXI CNG	Mumbai	2010	72000	CNG	Manual	First	26 km/l
1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	Diesel	Manual	First	19.(km
2	Honda Jazz V	Chennai	2011	46000	Petrol	Manual	First	18 km
3	Maruti Ertiga VDI	Chennai	2012	87000	Diesel	Manual	First	20.7 km
4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	Diesel	Automatic	Second	15 km

In [6]:
```
expr.columns.tolist()
```

Out[6]:
```
['Name',
 'Location',
 'Year',
 'Kilometers_Driven',
 'Fuel_Type',
 'Transmission',
 'Owner_Type',
 'Mileage',
 'Engine',
 'Power',
 'Seats',
 'Price']
```

In [7]:
```
expr.describe()
```

Out[7]:

	Year	Kilometers_Driven	Seats	Price
count	6019.000000	6.019000e+03	5977.000000	6019.000000
mean	2013.358199	5.873838e+04	5.278735	9.479468
std	3.269742	9.126884e+04	0.808840	11.187917
min	1998.000000	1.710000e+02	0.000000	0.440000
25%	2011.000000	3.400000e+04	5.000000	3.500000
50%	2014.000000	5.300000e+04	5.000000	5.640000
75%	2016.000000	7.300000e+04	5.000000	9.950000
max	2019.000000	6.500000e+06	10.000000	160.000000

In [8]:
```
expr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6019 entries, 0 to 6018
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Name               6019 non-null   object 
 1   Location           6019 non-null   object 
 2   Year               6019 non-null   int64  
 3   Kilometers_Driven  6019 non-null   int64  
 4   Fuel_Type          6019 non-null   object 
 5   Transmission       6019 non-null   object 
 6   Owner_Type         6019 non-null   object 
 7   Mileage            6017 non-null   object 
 8   Engine             5983 non-null   object 
 9   Power              5983 non-null   object 
 10  Seats              5977 non-null   float64
 11  Price              6019 non-null   float64
dtypes: float64(2), int64(2), object(8)
memory usage: 564.4+ KB
```

In [9]:
```
expr.Location.nunique()
expr.Owner_Type.nunique()
```

Out[9]:
```
4
```

In [10]:
```
expr.Owner_Type.unique()
```

Out[10]:
```
array(['First', 'Second', 'Fourth & Above', 'Third'], dtype=object)
```

In [11]:
```
expr.Seats.unique()
```

Out[11]:
```
array([ 5.,  7.,  8.,  4.,  6.,  2., nan, 10.,  9.,  0.])
```

In [7]:
```
expr.apply(lambda x: sum(x.isnull()),axis=0)
```

Out[7]:
```
Name                 0
Location             0
Year                 0
Kilometers_Driven    0
Fuel_Type            0
Transmission         0
Owner_Type           0
Mileage              2
Engine              36
Power               36
Seats               42
Price                0
dtype: int64
```

Since the dataset has approx 6000 Rows of data so dropping this much amount of values will not create much problem.

In [8]:
```
expr.dropna(inplace= True)
expr.isnull().sum()
```

Out[8]:
```
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
Price                0
dtype: int64
```

In [9]:
```
expr.dtypes
```

Out[9]:
```
Name                  object
Location              object
Year                   int64
Kilometers_Driven      int64
Fuel_Type             object
Transmission          object
Owner_Type            object
Mileage               object
Engine                object
Power                 object
Seats                float64
Price                float64
dtype: object
```

Since the Mileage Power and Engine are a object type in the data set so firstly we will try to convert that object data type to a float data type for visualization and for our model also.

In [10]:
```
expr['Mileage'] = expr['Mileage'].replace('km/kg', '',regex = True)
```

In [11]:
```
expr['Mileage'] = expr['Mileage'].replace('kmpl', '',regex = True)
```

In [12]:
```
expr.Mileage
```

Out[12]:
```
0        26.6
1       19.67
2        18.2
3       20.77
4        15.2
        ...
6014     28.4
6015     24.4
6016     14.0
6017     18.9
6018    25.44
Name: Mileage, Length: 5975, dtype: object
```

In [13]:
```
Mileage_Float = expr.Mileage.astype(float)
Mileage_Float
```

Out[13]:
```
0       26.60
1       19.67
2       18.20
3       20.77
4       15.20
        ...
6014    28.40
6015    24.40
6016    14.00
6017    18.90
6018    25.44
Name: Mileage, Length: 5975, dtype: float64
```

In [14]:
```
expr['Mileage_Float'] = Mileage_Float
```

In [15]:
```
expr.Mileage_Float
```

Out[15]:
```
0       26.60
1       19.67
2       18.20
3       20.77
4       15.20
        ...
6014    28.40
6015    24.40
6016    14.00
6017    18.90
6018    25.44
Name: Mileage_Float, Length: 5975, dtype: float64
```

In [16]:
```
expr.columns.tolist()
```

Out[16]:
```
['Name',
 'Location',
 'Year',
 'Kilometers_Driven',
 'Fuel_Type',
 'Transmission',
 'Owner_Type',
 'Mileage',
 'Engine',
 'Power',
 'Seats',
 'Price',
 'Mileage_Float']
```

In [17]:
```
expr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5975 entries, 0 to 6018
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Name               5975 non-null   object 
 1   Location           5975 non-null   object 
 2   Year               5975 non-null   int64  
 3   Kilometers_Driven  5975 non-null   int64  
 4   Fuel_Type          5975 non-null   object 
 5   Transmission       5975 non-null   object 
 6   Owner_Type         5975 non-null   object 
 7   Mileage            5975 non-null   object 
 8   Engine             5975 non-null   object 
 9   Power              5975 non-null   object 
 10  Seats              5975 non-null   float64
 11  Price              5975 non-null   float64
 12  Mileage_Float      5975 non-null   float64
dtypes: float64(3), int64(2), object(8)
memory usage: 653.5+ KB
```

In [18]:
```
del expr['Mileage']
```

In [19]:
```
expr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5975 entries, 0 to 6018
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Name               5975 non-null   object 
 1   Location           5975 non-null   object 
 2   Year               5975 non-null   int64  
 3   Kilometers_Driven  5975 non-null   int64  
 4   Fuel_Type          5975 non-null   object 
 5   Transmission       5975 non-null   object 
 6   Owner_Type         5975 non-null   object 
 7   Engine             5975 non-null   object 
 8   Power              5975 non-null   object 
 9   Seats              5975 non-null   float64
 10  Price              5975 non-null   float64
 11  Mileage_Float      5975 non-null   float64
dtypes: float64(3), int64(2), object(7)
memory usage: 606.8+ KB
```

In [20]:
```
expr['Engine'] = expr['Engine'].replace('CC', '',regex = True)
```

In [21]:
```
expr.Engine
```

Out[21]:
```
0        998
1       1582
2       1199
3       1248
4       1968
        ...
6014    1248
6015    1120
6016    2498
6017     998
6018     936
Name: Engine, Length: 5975, dtype: object
```

In [22]:
```
Engine_Float = expr.Engine.astype(float)
Engine_Float
```

Out[22]:
```
0        998.0
1       1582.0
2       1199.0
3       1248.0
4       1968.0
         ...
6014    1248.0
6015    1120.0
6016    2498.0
6017     998.0
6018     936.0
Name: Engine, Length: 5975, dtype: float64
```

In [23]:
```
expr['Engine_Float'] = Engine_Float
```

In [24]:
```
del expr['Engine']
```

In [25]:
```
expr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5975 entries, 0 to 6018
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Name               5975 non-null   object 
 1   Location           5975 non-null   object 
 2   Year               5975 non-null   int64  
 3   Kilometers_Driven  5975 non-null   int64  
 4   Fuel_Type          5975 non-null   object 
 5   Transmission       5975 non-null   object 
 6   Owner_Type         5975 non-null   object 
 7   Power              5975 non-null   object 
 8   Seats              5975 non-null   float64
 9   Price              5975 non-null   float64
 10  Mileage_Float      5975 non-null   float64
 11  Engine_Float       5975 non-null   float64
dtypes: float64(4), int64(2), object(6)
memory usage: 606.8+ KB
```

In [26]:
```
expr['Power'] = expr['Power'].replace('bhp', '',regex = True)
```

In [27]:
```
expr.Power
```

Out[27]:
```
0       58.16
1       126.2
2        88.7
3       88.76
4       140.8
        ...
6014      74
6015      71
6016     112
6017    67.1
6018    57.6
Name: Power, Length: 5975, dtype: object
```

In [28]:
```
Power_Float = expr.Power.astype(float)
Power_Float
```

Out[28]:
```
0        58.16
1       126.20
2        88.70
3        88.76
4       140.80
         ...
6014     74.00
6015     71.00
6016    112.00
6017     67.10
6018     57.60
Name: Power, Length: 5975, dtype: float64
```

In [29]:
```
expr['Power_Float'] = Power_Float
```

In [30]:
```
del expr['Power']
```

In [31]:
```
expr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5975 entries, 0 to 6018
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Name               5975 non-null   object 
 1   Location           5975 non-null   object 
 2   Year               5975 non-null   int64  
 3   Kilometers_Driven  5975 non-null   int64  
 4   Fuel_Type          5975 non-null   object 
 5   Transmission       5975 non-null   object 
 6   Owner_Type         5975 non-null   object 
 7   Seats              5975 non-null   float64
 8   Price              5975 non-null   float64
 9   Mileage_Float      5975 non-null   float64
 10  Engine_Float       5975 non-null   float64
 11  Power_Float        5975 non-null   float64
dtypes: float64(5), int64(2), object(5)
memory usage: 606.8+ KB
```

Now we have our data set as a float values. Observed that in Power column and the Mileage column some of the values are zero. And if the values are zero no on will technically purchase a car which has Zero Mileage and Power. So Basically it is an outlier so we will remove it before moving to our data visualization.Since we have made this observation right now so dealing with them right now might be a good option.

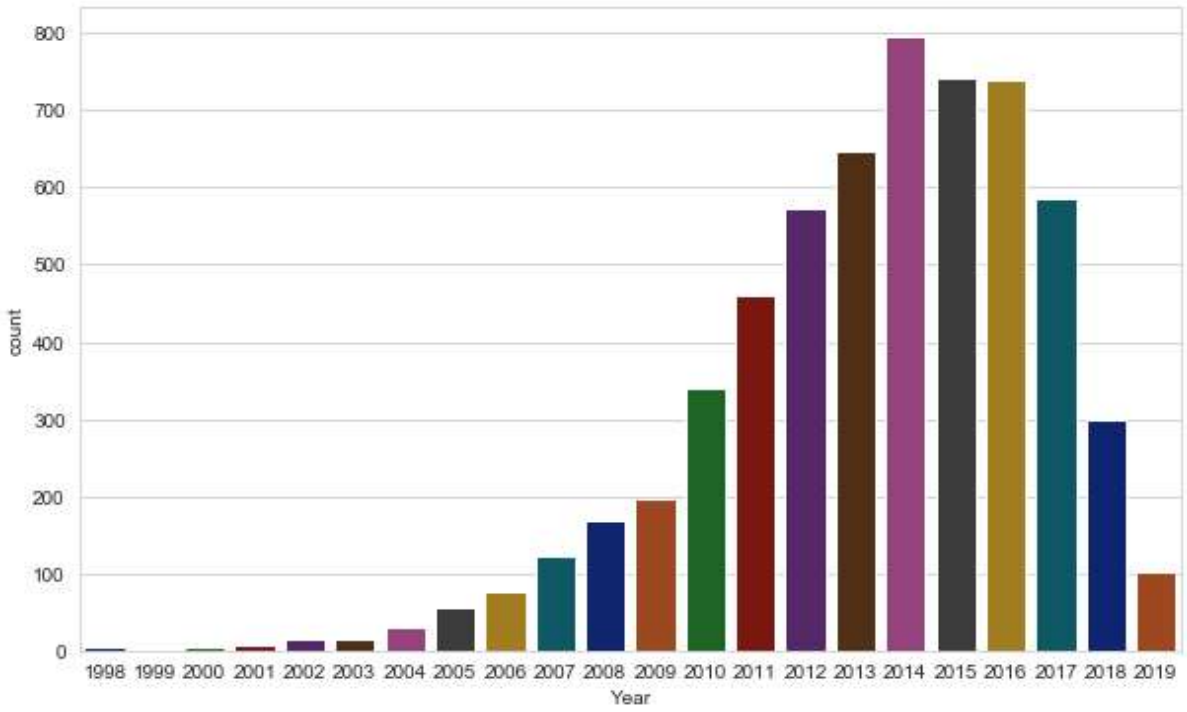
VISUAL SUMMARY OF THE DATASET

UNIVARIET ANALYSIS FIRST

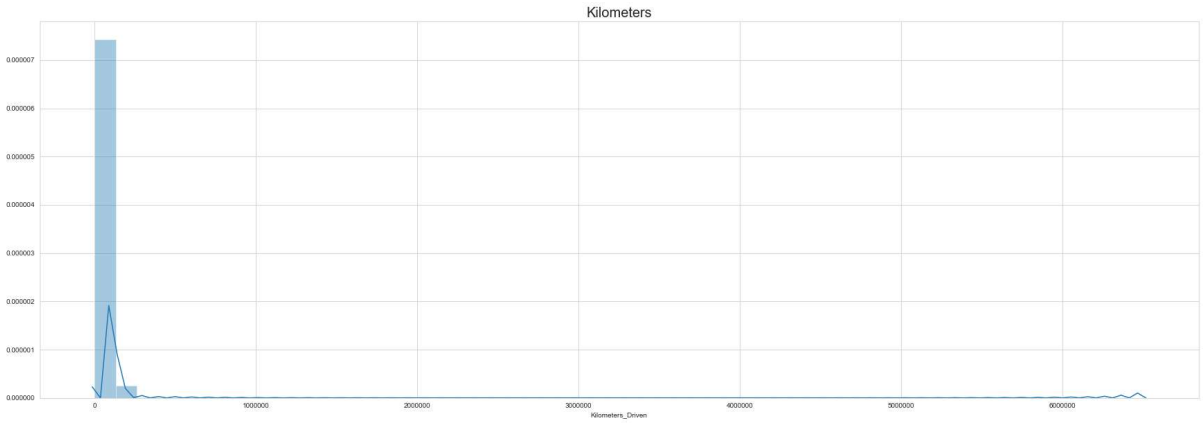
In [32]:
```
sns.set_style('whitegrid')
```

In [58]:
```
plt.figure(figsize = (10,6))
sns.countplot(expr['Year'],palette='dark')
```

Out[58]: <matplotlib.axes._subplots.AxesSubplot at 0x29b43778208>

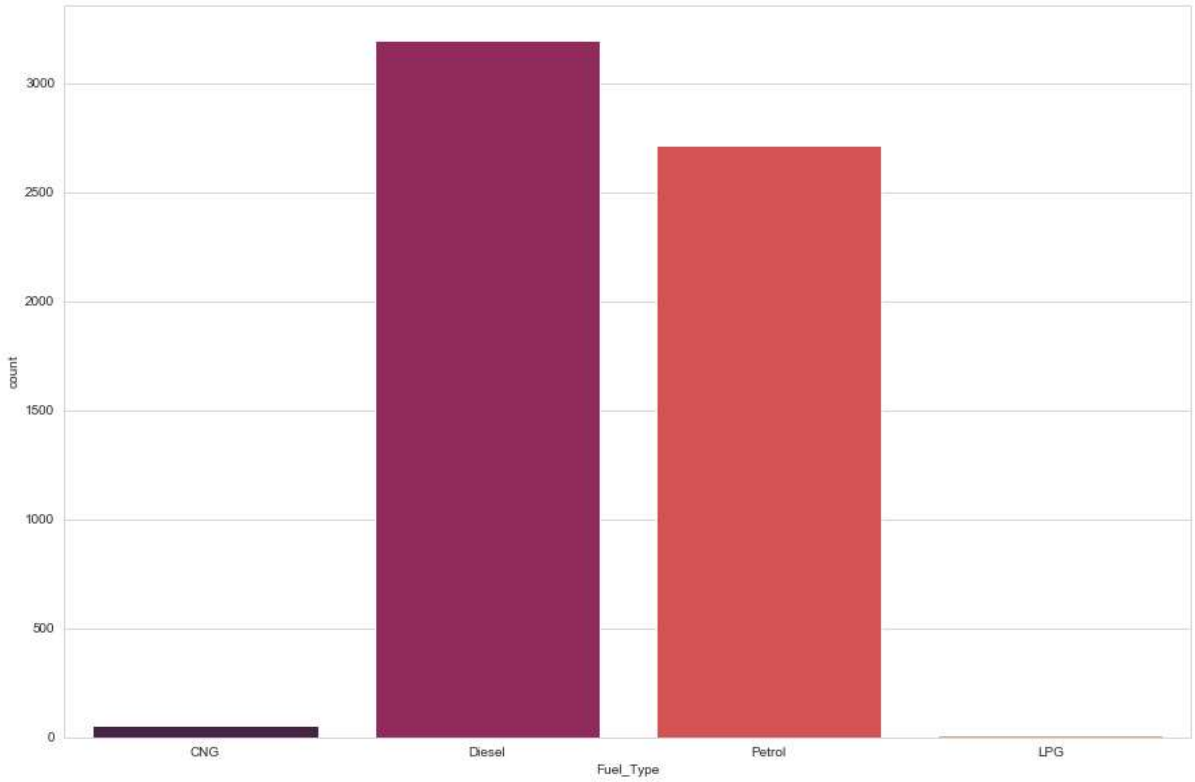


In [57]:
```
plt.rcParams['figure.figsize'] = (30,10)
sns.distplot(expr['Kilometers_Driven'])
plt.title("Kilometers",fontsize = 20)
plt.show()
```



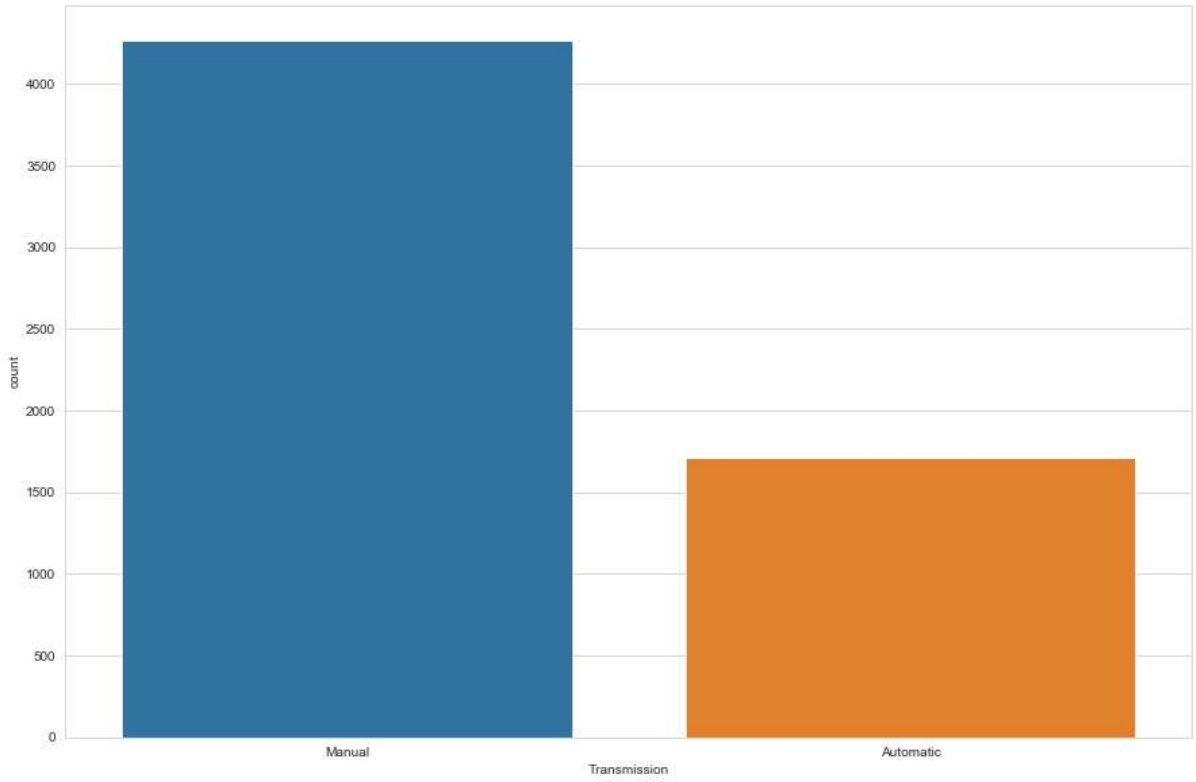
In [56]:
```
plt.figure(figsize = (15,10))
sns.countplot(expr['Fuel_Type'],palette= 'rocket')
```

Out[56]: <matplotlib.axes._subplots.AxesSubplot at 0x29b425609c8>



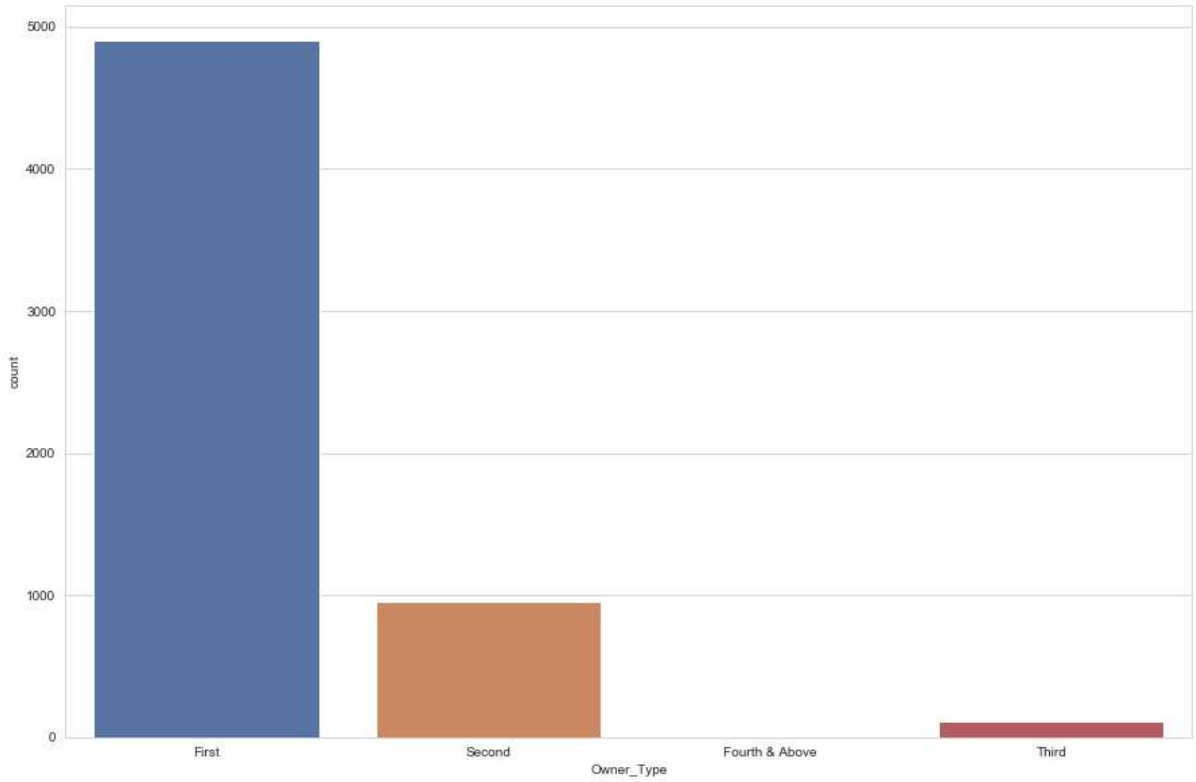
In [55]:
```
plt.figure(figsize = (15,10))
sns.countplot(expr['Transmission'])
```

Out[55]: <matplotlib.axes._subplots.AxesSubplot at 0x29b41d77748>



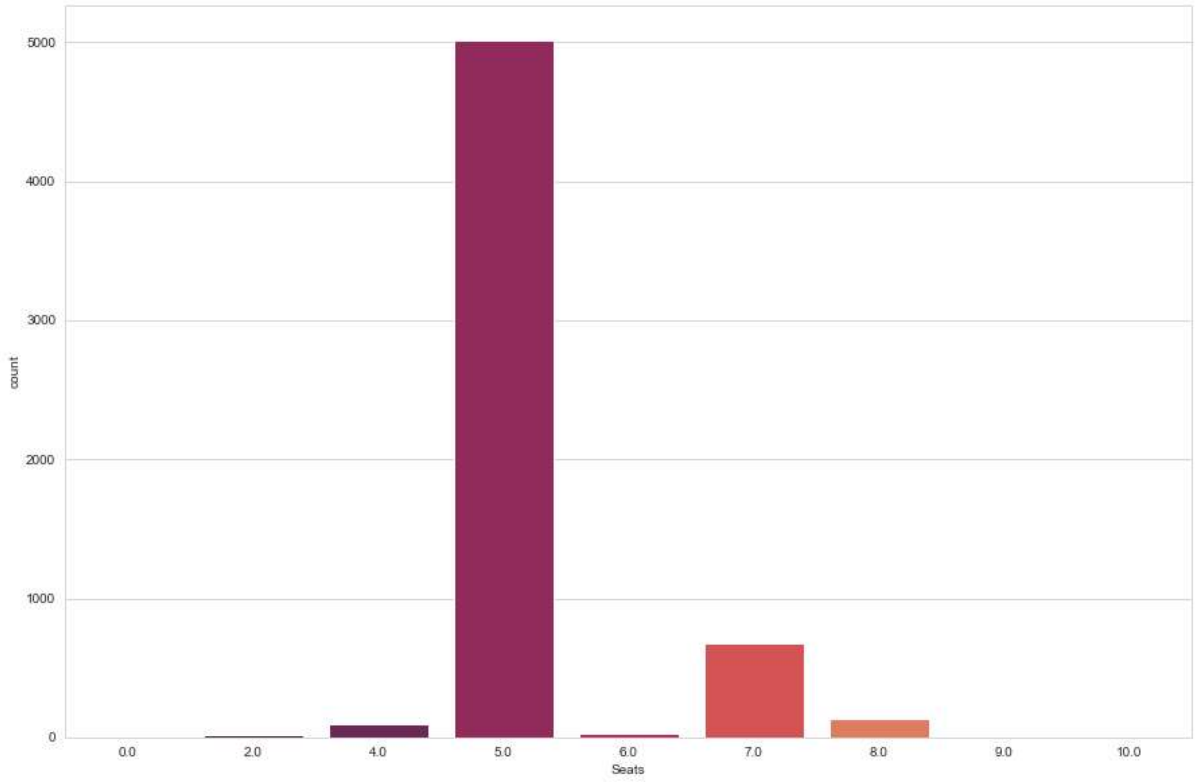
In [53]:
```
plt.figure(figsize = (15,10))
sns.countplot(expr['Owner_Type'],palette="deep")
```

Out[53]: <matplotlib.axes._subplots.AxesSubplot at 0x29b41ca1a88>



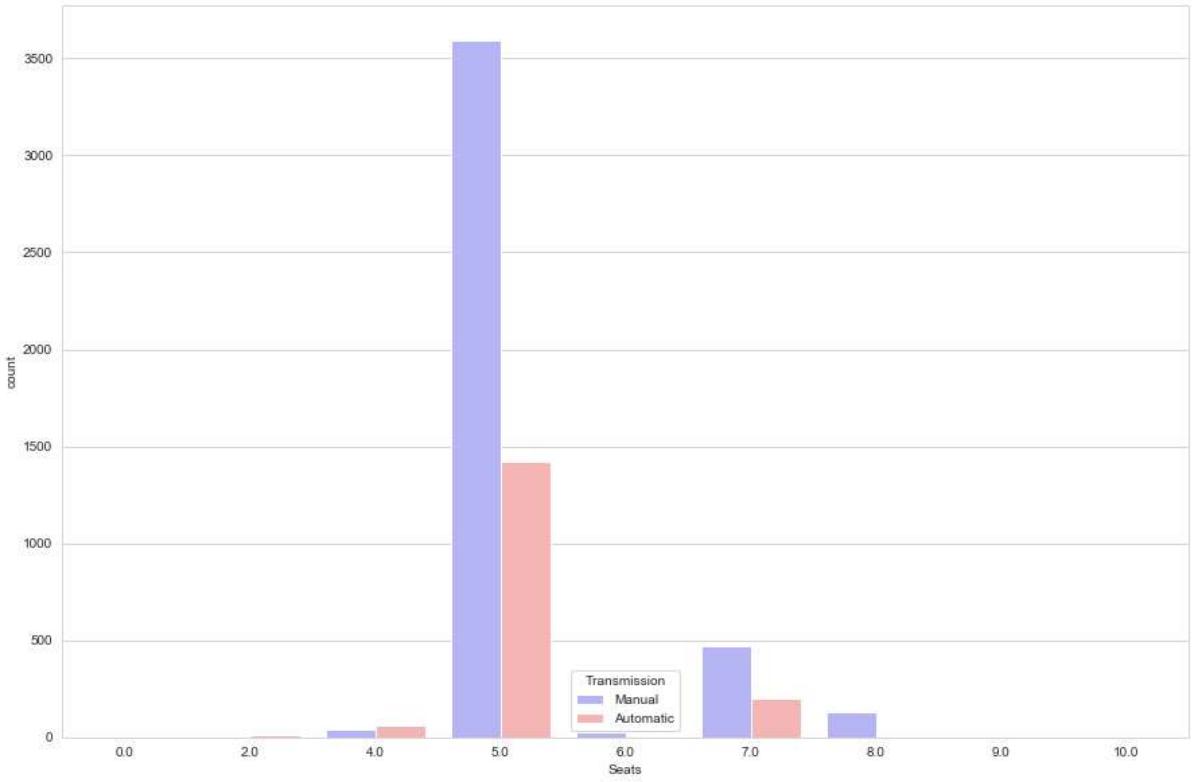
In [52]:
```
plt.figure(figsize = (15,10))
sns.countplot(expr['Seats'],palette= 'rocket')
```

Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0x29b41bed688>

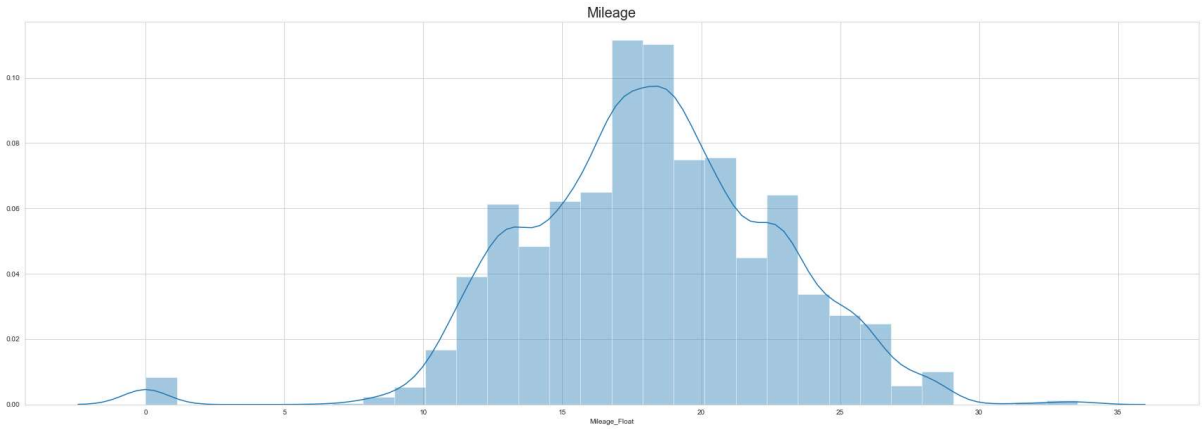


In [132]:
```
plt.figure(figsize = (15,10))
sns.countplot(x='Seats',hue='Transmission',palette= 'bwr',data =expr)
```

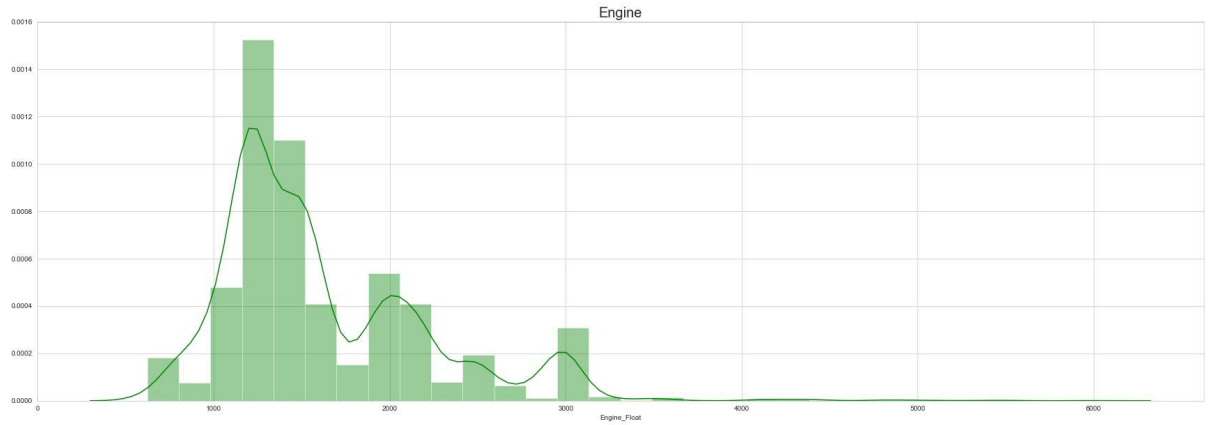
Out[132]: <matplotlib.axes._subplots.AxesSubplot at 0x1b3919d1e88>



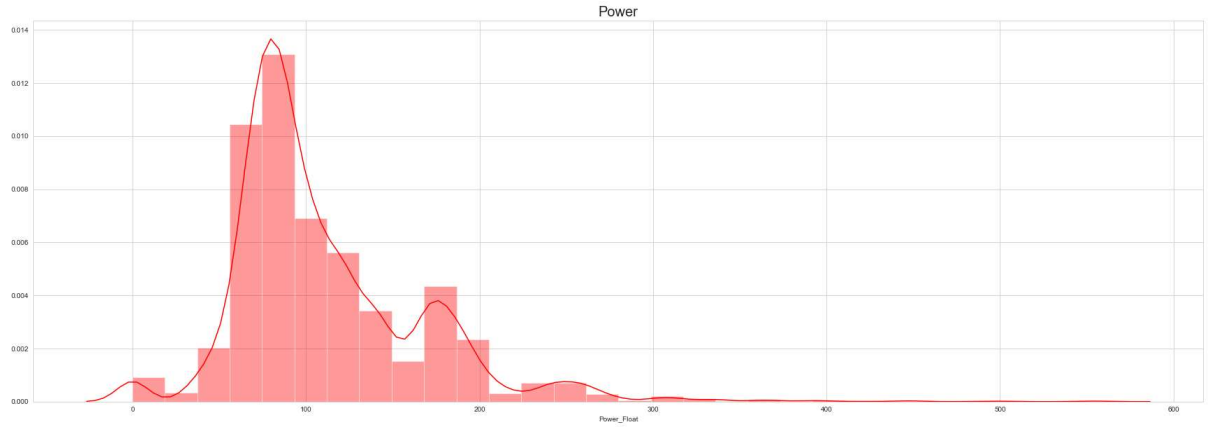
In [49]:
```
plt.rcParams['figure.figsize'] = (30,10)
sns.distplot(expr['Mileage_Float'],bins=30)
plt.title("Mileage",fontsize = 20)
plt.show()
```



In [48]:
```
plt.rcParams['figure.figsize'] = (30,10)
sns.distplot(expr['Engine_Float'],bins=30,color ='g')
plt.title("Engine",fontsize = 20)
plt.show()
```

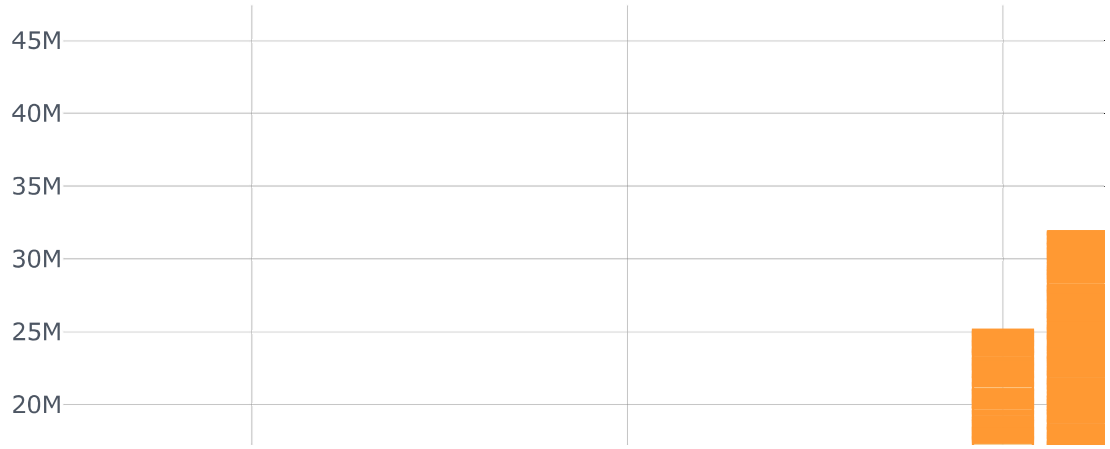


In [47]:
```
plt.rcParams['figure.figsize'] = (30,10)
sns.distplot(expr['Power_Float'],bins=30,color ='r')
plt.title("Power",fontsize = 20)
plt.show()
```



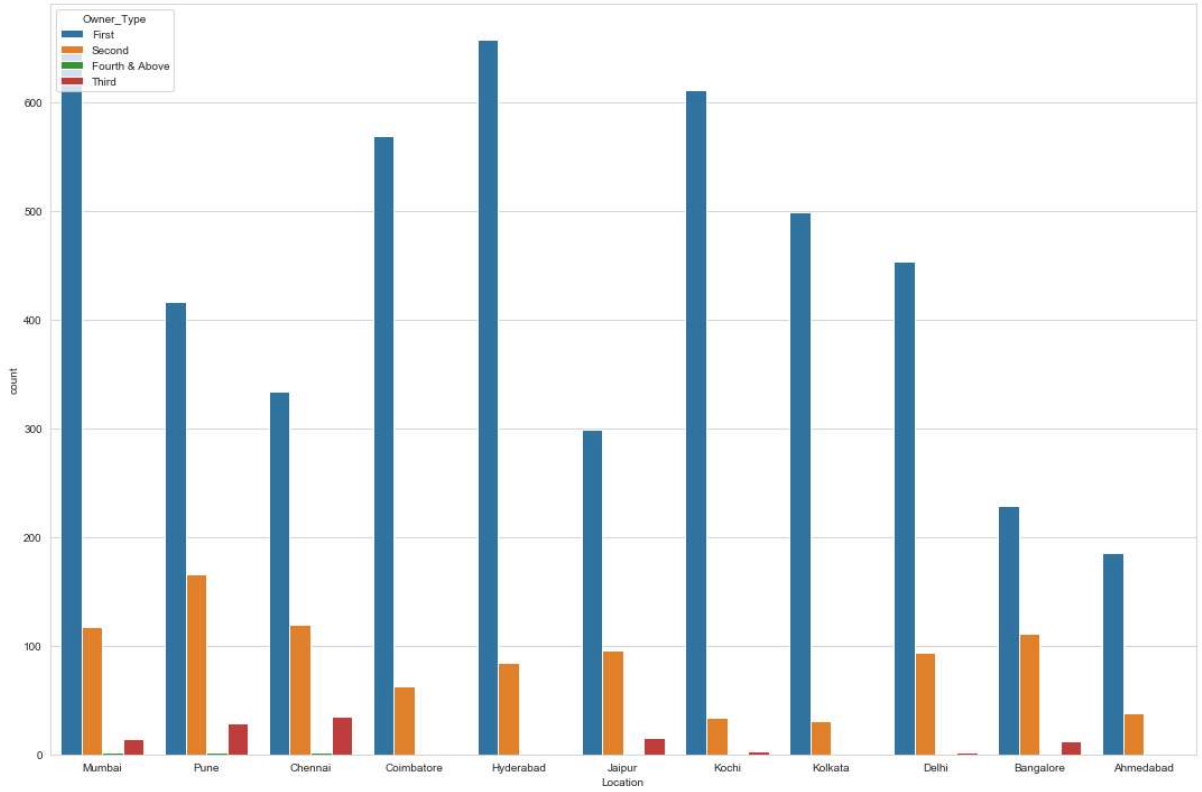
Bivariet Analysis Now

In [33]:
```
expr.iplot(kind = 'bar', x= 'Year', y= 'Kilometers_Driven')
```



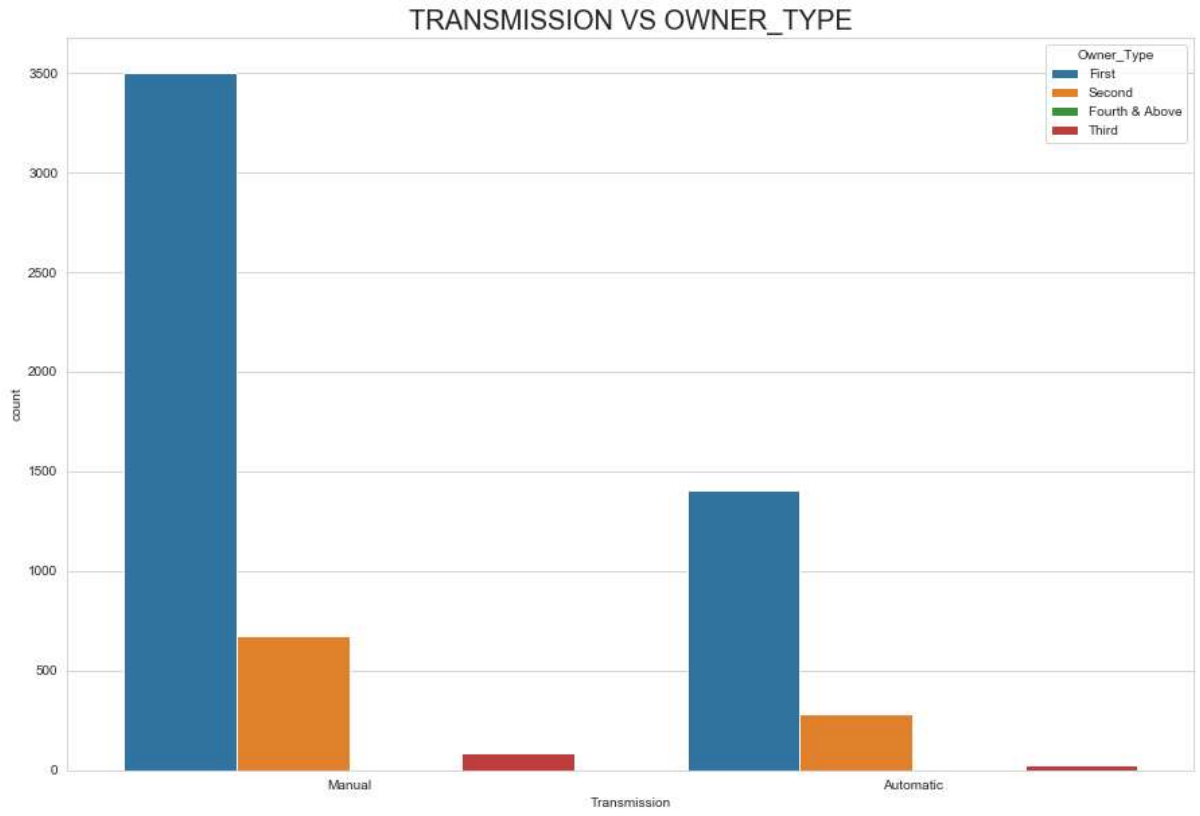
In [34]:
```
expr.iplot(kind = 'bar', x= 'Fuel_Type', y= 'Owner_Type',colors = '#079b91',xT
itle = ' Fuel',yTitle = 'Owner')
```

In [46]:
```
plt.figure(figsize = (15,10))
sns.countplot(x='Location',hue = 'Owner_Type',data = expr)
plt.tight_layout()
```



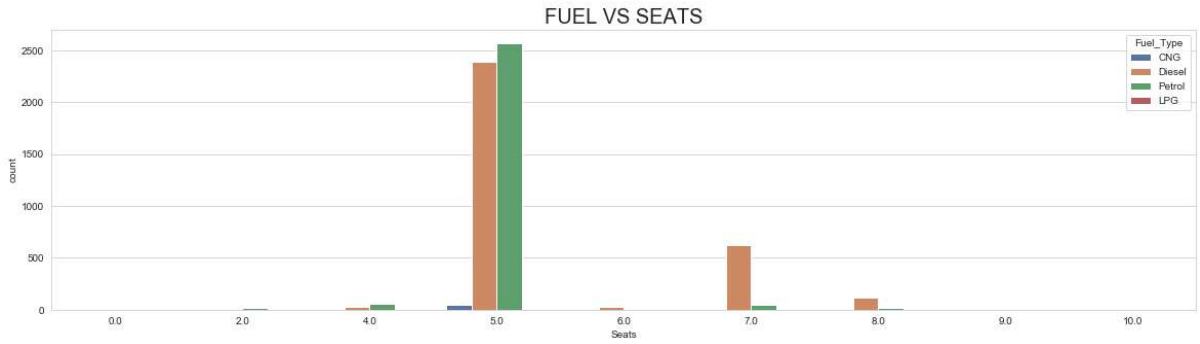
In [45]:
```
plt.figure(figsize = (15,10))
sns.countplot(x= 'Transmission',hue = 'Owner_Type',data = expr)
plt.title('TRANSMISSION VS OWNER_TYPE',fontsize = 20)
```

Out[45]: Text(0.5, 1.0, 'TRANSMISSION VS OWNER_TYPE')



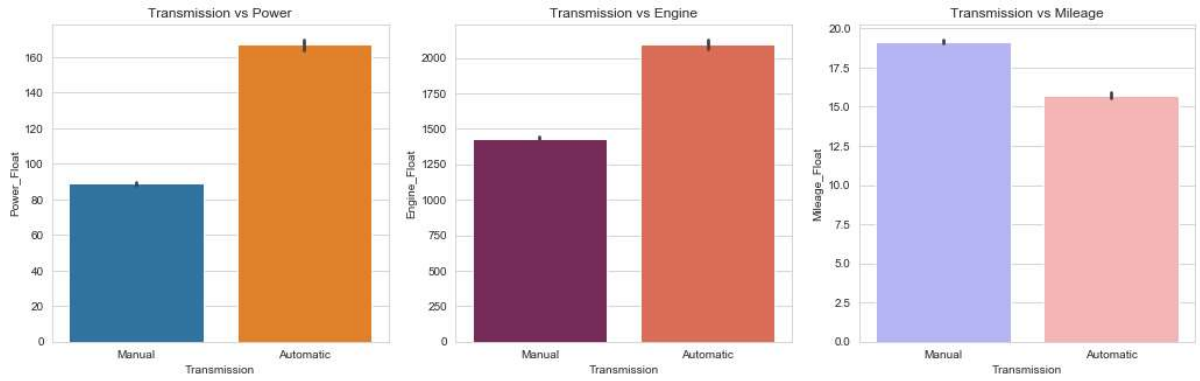
In [44]:
```
plt.figure(figsize= (20,5))
sns.countplot(x = 'Seats',hue = 'Fuel_Type',data = expr,palette = 'deep')
plt.title('FUEL VS SEATS',fontsize = 20)
```

Out[44]: Text(0.5, 1.0, 'FUEL VS SEATS')



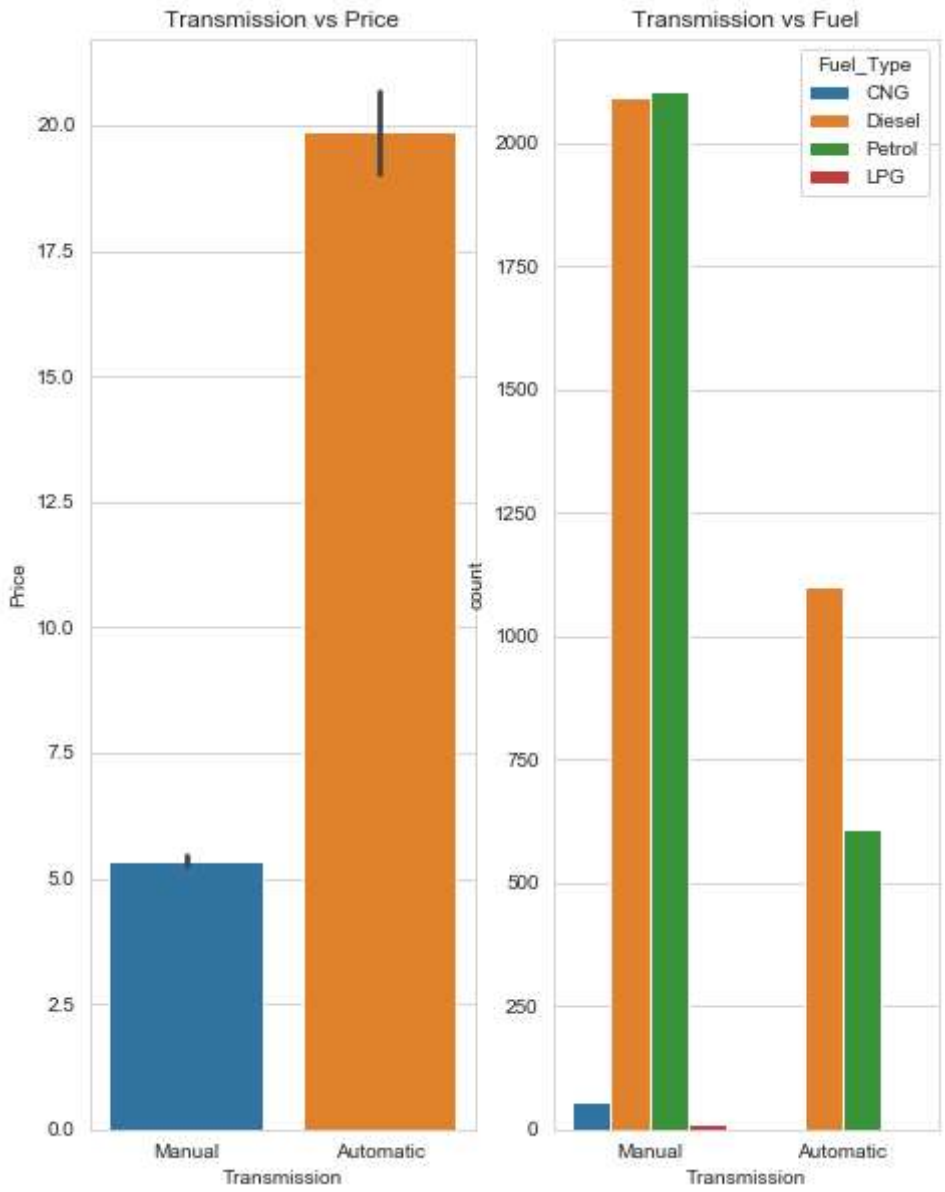
In [43]:
```
plt.figure(figsize= (30,5))
plt.subplot(151)
plt.title('Transmission vs Power')
sns.barplot(x ='Transmission',y = 'Power_Float',data = expr)
plt.subplot(152)
plt.title('Transmission vs Engine')
sns.barplot(x ='Transmission',y = 'Engine_Float',data = expr,palette= 'rocket'
)
plt.subplot(153)
plt.title('Transmission vs Mileage')
sns.barplot(x ='Transmission',y = 'Mileage_Float',data = expr,palette = 'bwr')
```

Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0x29b413bf248>



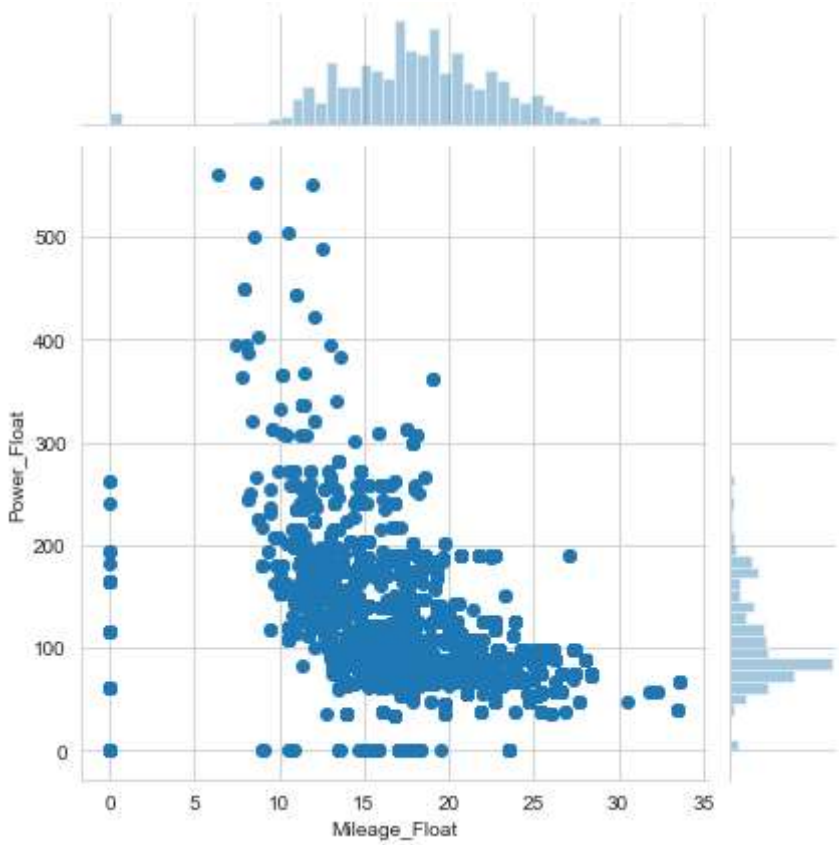
In [130]:
```
plt.figure(figsize= (20,10))
plt.subplot(151)
plt.title('Transmission vs Price')
sns.barplot(x ='Transmission',y = 'Price',data = expr)
plt.subplot(152)
plt.title('Transmission vs Fuel')
sns.countplot(x ='Transmission',hue = 'Fuel_Type',data = expr)
```

Out[130]: <matplotlib.axes._subplots.AxesSubplot at 0x1b39184bb48>



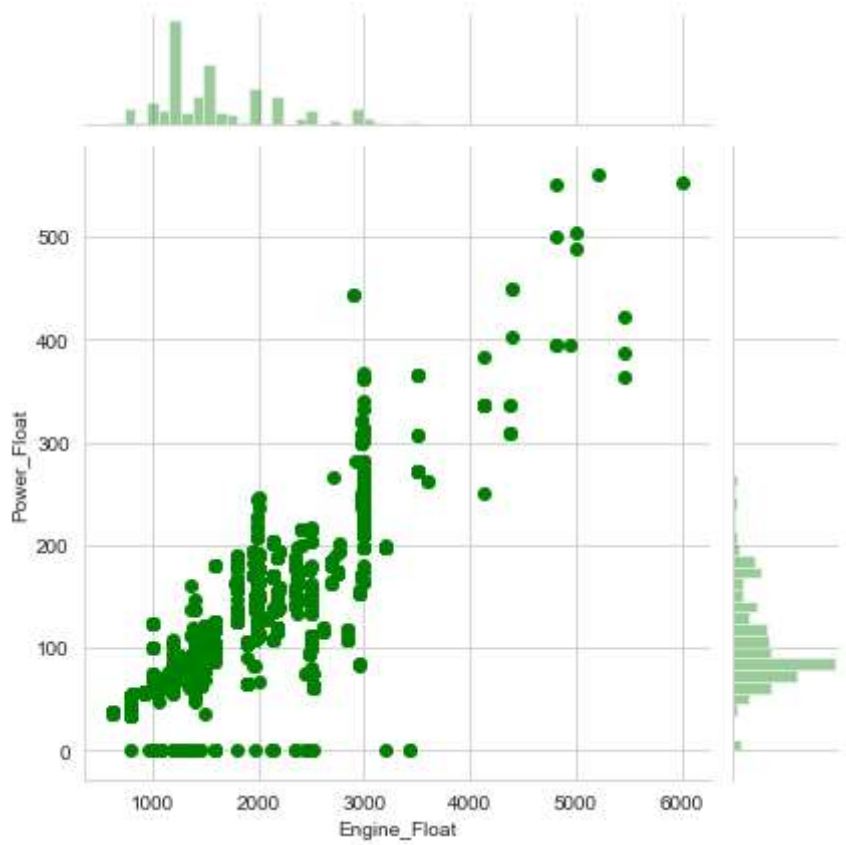
In [42]:
```
sns.jointplot(x ='Mileage_Float',y = 'Power_Float',data = expr)
```

Out[42]: <seaborn.axisgrid.JointGrid at 0x29b3f5deb48>



In [41]:
```
sns.jointplot(x ='Engine_Float',y = 'Power_Float',data = expr,color = 'g')
```

Out[41]: <seaborn.axisgrid.JointGrid at 0x29b3f5d0d08>

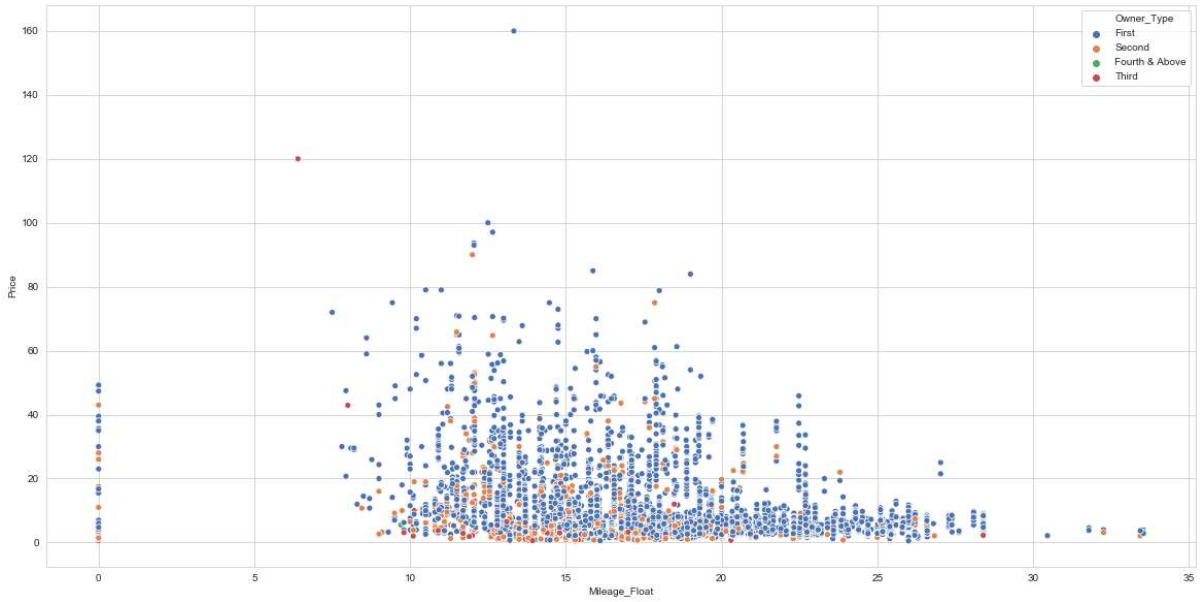


In [443]:
```
plt.figure(figsize = (10,5))
sns.barplot(x= 'Owner_Type',y = 'Price',data = expr,palette = 'bwr')
plt.title('OWNER VS PRICE')
```

Out[443]: Text(0.5, 1.0, 'OWNER VS PRICE')

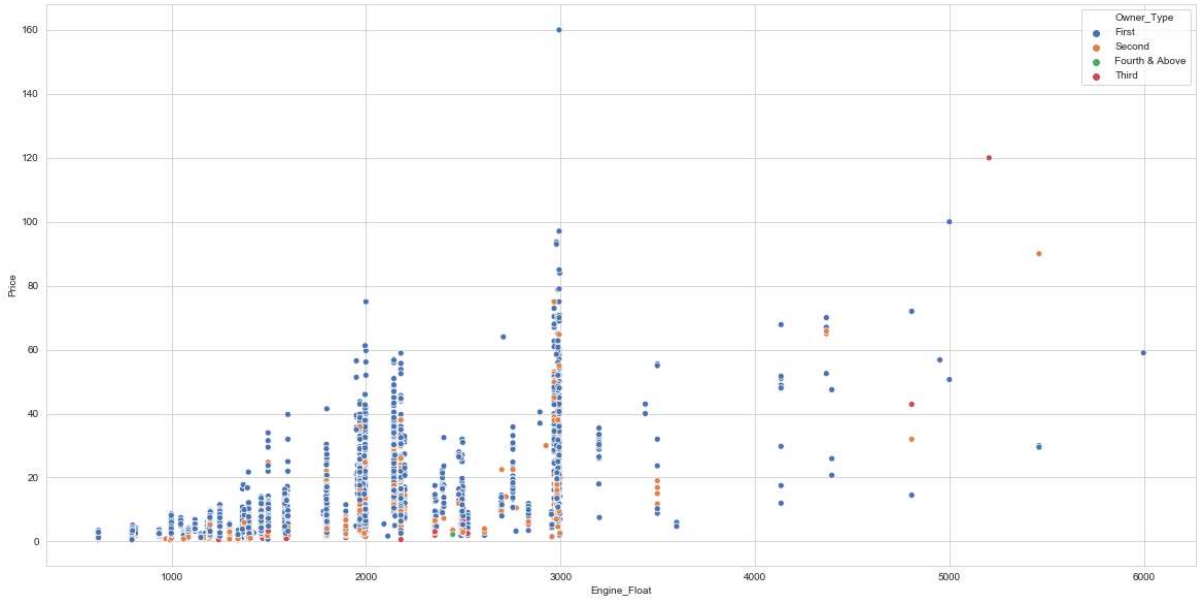
In [116]:
```
plt.figure(figsize = (20,10))
sns.scatterplot(x='Mileage_Float',y = 'Price',hue = 'Owner_Type',data = expr,p
alette = 'deep')
```

Out[116]: <matplotlib.axes._subplots.AxesSubplot at 0x1b38a50b508>



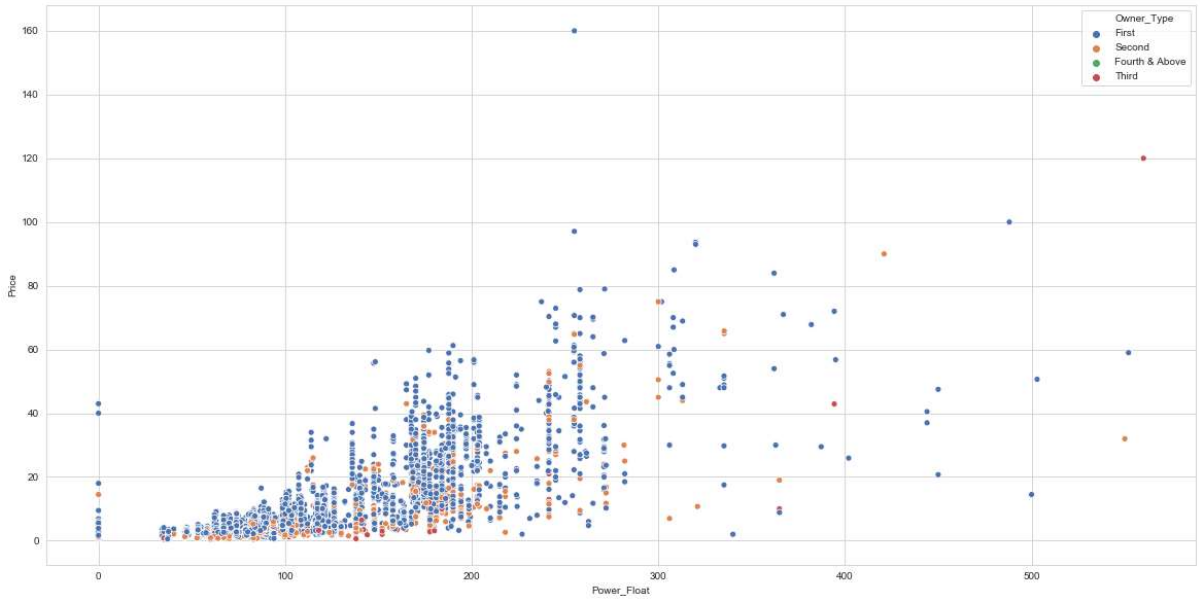
In [118]:
```
plt.figure(figsize = (20,10))
sns.scatterplot(x='Engine_Float',y = 'Price',hue = 'Owner_Type',data = expr,pa
lette = 'deep')
```

Out[118]: <matplotlib.axes._subplots.AxesSubplot at 0x1b38a827fc8>



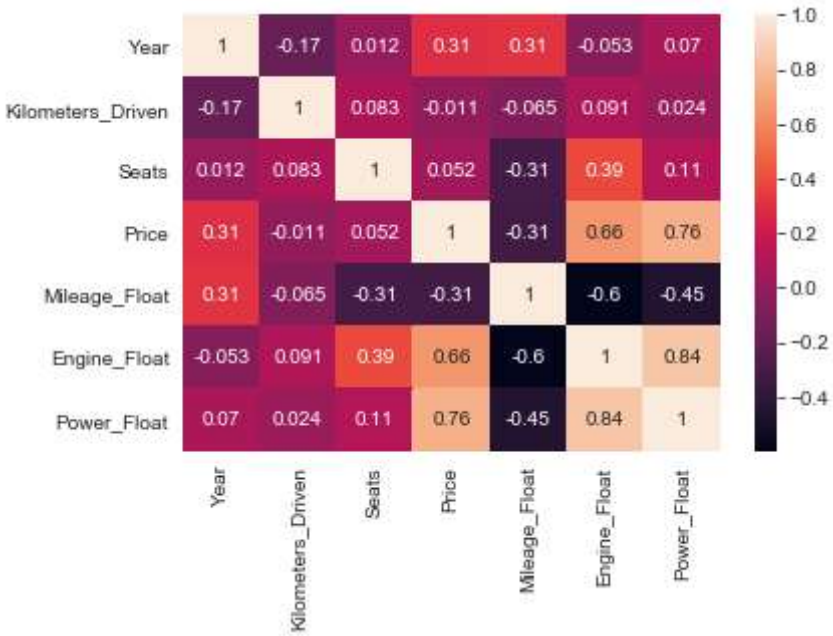
In [119]:
```
plt.figure(figsize = (20,10))
sns.scatterplot(x='Power_Float',y = 'Price',hue = 'Owner_Type',data = expr,pal
ette = 'deep')
```

Out[119]: <matplotlib.axes._subplots.AxesSubplot at 0x1b38abeaa08>



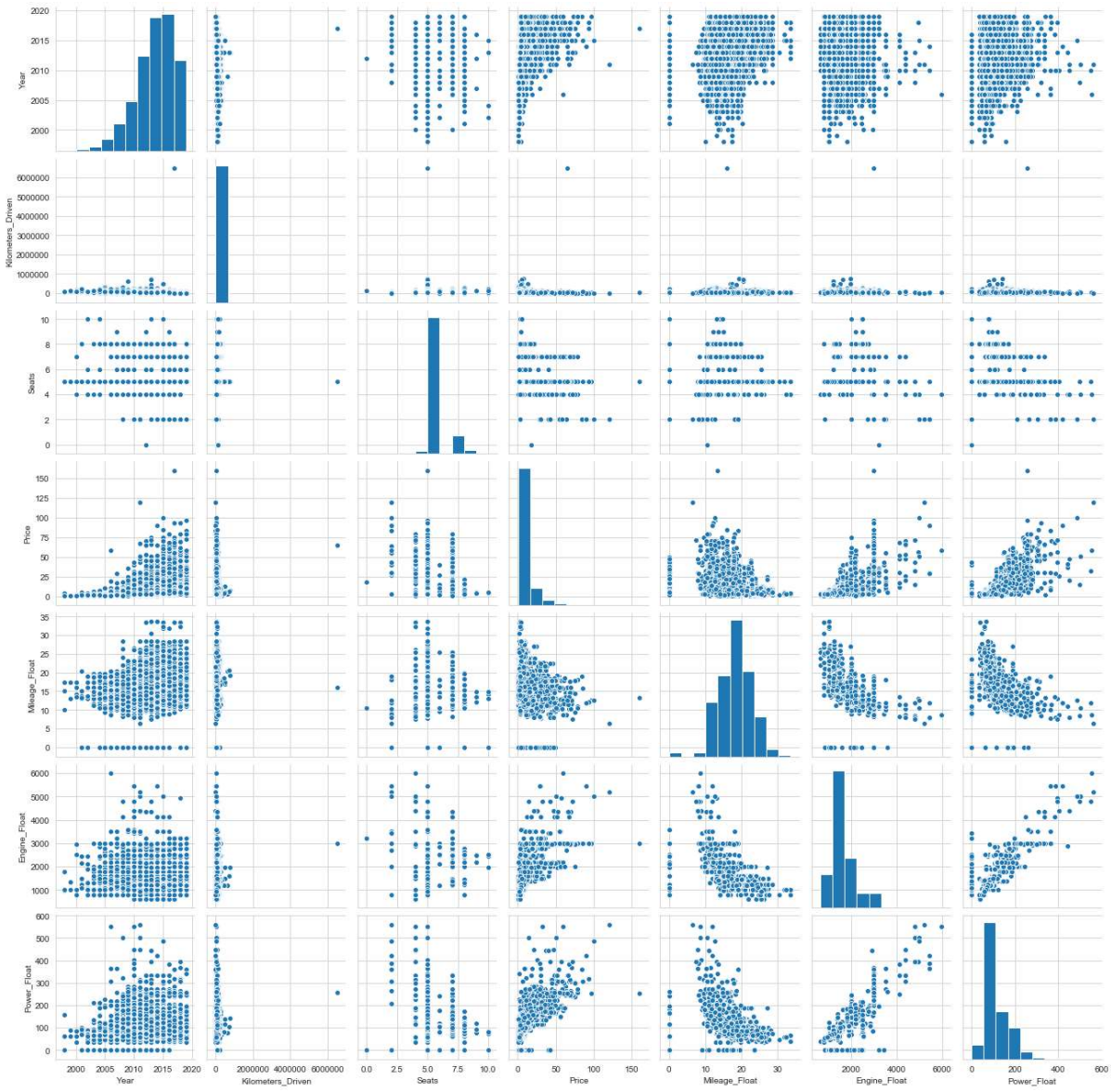
In [120]:
```
ex= expr.corr()
sns.heatmap(ex,annot = True)
```

Out[120]: <matplotlib.axes._subplots.AxesSubplot at 0x1b38abd9e08>



In [122]:
```
sns.pairplot(expr,palette='bwr')
```

Out[122]:
```
<seaborn.axisgrid.PairGrid at 0x1b38b2a0c88>
```



OBSERVATIONS:

- 2014 Year based cars are more in number. And cars from year 2015 and 2016 are almost same.
- Diesel cars are more than petrol. LPG is negligble.
- 5 No. of seats are more then comes 7.
- Manual Transmission are more than automatic transmission.
- First Class Owner type are more than second then third and then fourth. Fourth and above is very less.
- Most Owner prefer manual transmission.
- Fuel Type of 5 seater are more because of more number of 5 seat type. In that also petrol type is more than diesel.
- LPG fuel type is negligble.
- Power of automatic type are more than manual.
- Engine of automatic type vehicles are more than manual.
- Mileage of manual type is more than automatic type.
- Price of automatic type transmission is more than manual.
- 1000-2000 cc capacity of engines are more.
- Price range is far more for first class owner type then any other type.
- 100-200 bhp power cars are more and has priced their cars between 60k.

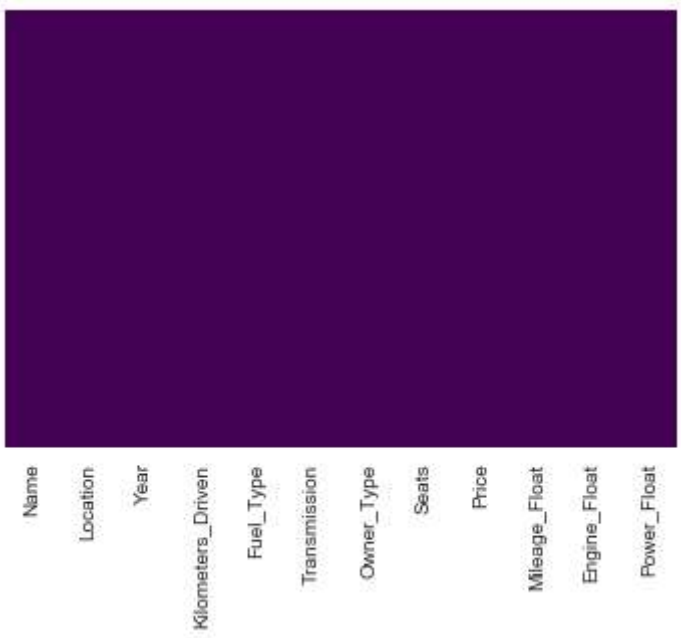
Inferences made from data visualization

- Outliers present in Kilometers_Driven.
- Outliers present in Mileage and Power.
- Name, Location columns does not have any significant values in the dataset.
- Owners prefers manual type over automatic type because :
  - **Mileage is more in manual type transmission**.
  - **Price is less for manual type cars**.
- When it comes to performance automatic type cars performs better than manual cars.
- Manual cars are more than automatic cars.
- **For models we need to clear the outliers first otherwise the model will not be accurate.**

Identification of outliers and removal.

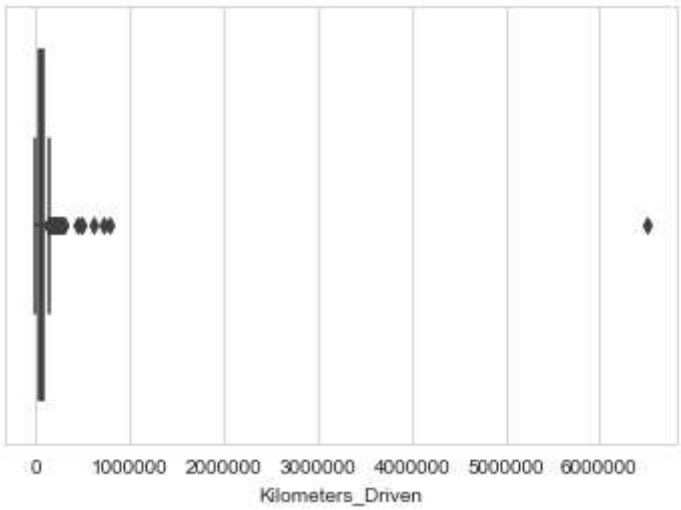
In [35]:
```
sns.heatmap(expr.isnull(),yticklabels=False,cbar= False,cmap = 'viridis')
#This indicates we dont have any null values.
```

Out[35]: <matplotlib.axes._subplots.AxesSubplot at 0x15daaf64788>



In [36]:
```
sns.boxplot(x ='Kilometers_Driven',data = expr)
```

Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x15dab4eef48>



Clearly it is visible that it contains a outlier.

In [37]:
```
Q1 = expr.Kilometers_Driven.quantile(0.25)
print(Q1)
Q3 = expr.Kilometers_Driven.quantile(0.75)
print(Q3)
IQR = Q3 - Q1
print(IQR)
print(Q1 - (1.5 * IQR))
print(Q3 + (1.5 * IQR))
```

```
33908.0
73000.0
39092.0
-24730.0
131638.0
```

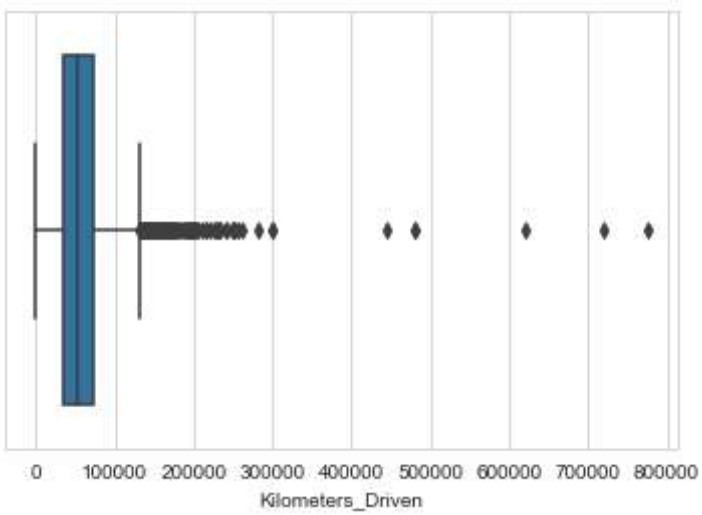
In [38]:
```
expr[~((expr.Kilometers_Driven < (Q1 - 1.5 * IQR)) |(expr.Kilometers_Driven > (Q3 + 1.5 * IQR)))].Kilometers_Driven.median()
```

Out[38]:
```
52000.0
```

In [39]:
```
index = expr[(expr['Kilometers_Driven'] >= 1000000)].index
expr.drop(index, inplace=True)
```

In [40]:
```
sns.boxplot(x ='Kilometers_Driven',data = expr)
```

Out[40]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x15dab202f48>
```

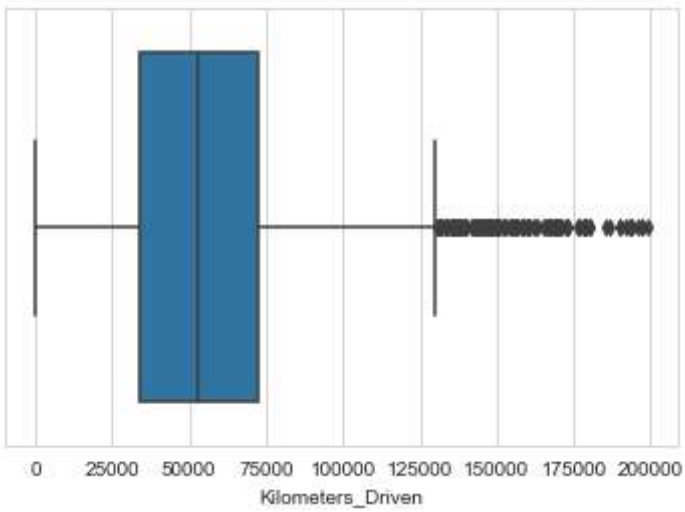


Since now we can see more precision in outliers we can remove them as well.

In [41]:
```
index = expr[(expr['Kilometers_Driven'] >= 200000)].index
expr.drop(index, inplace=True)
```

In [42]:
```
sns.boxplot(x ='Kilometers_Driven',data = expr)
```

Out[42]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x15dac69c988>
```

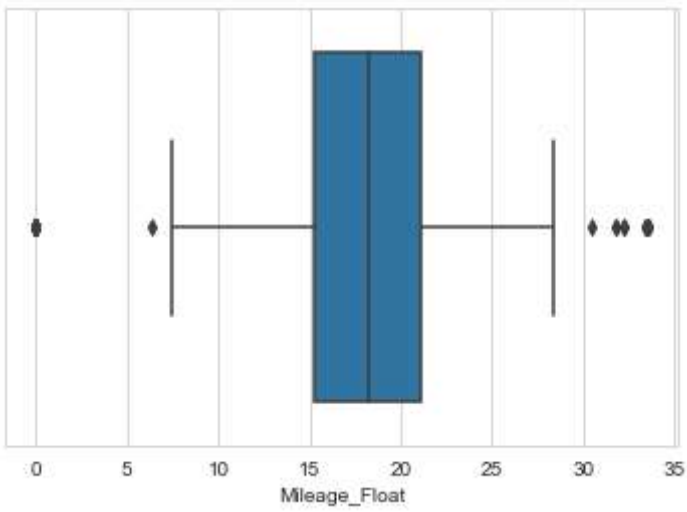


These points cannot be considered as outliers now because they are more in number and removing it will cost accuracy of the dataset.

Now checking outliers from the Mileage column.

In [43]:
```
sns.boxplot(x ='Mileage_Float',data = expr)
```

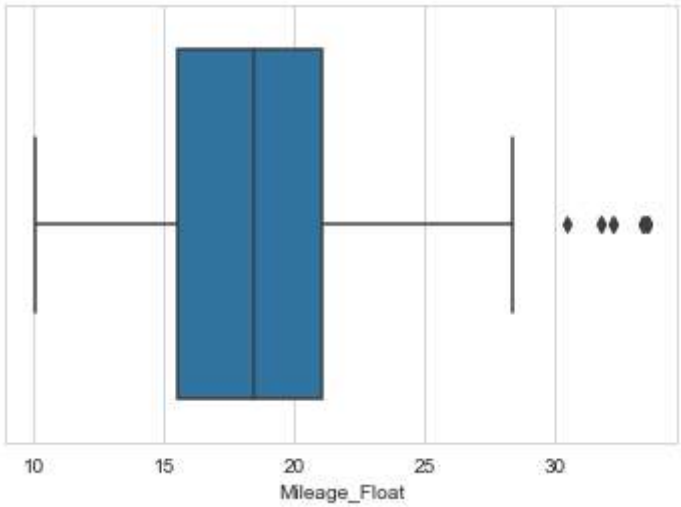
Out[43]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x15dab620cc8>
```



In [44]:
```
index = expr[(expr['Mileage_Float'] <=10)].index
expr.drop(index, inplace=True)
```

In [45]:
```
sns.boxplot(x ='Mileage_Float',data = expr)
```

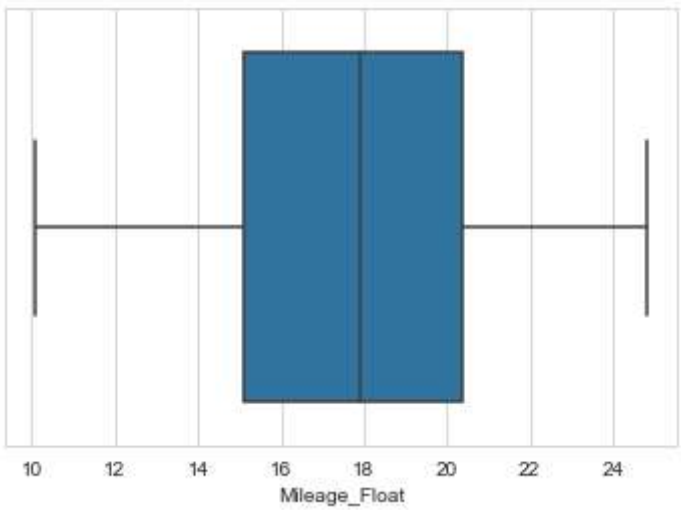
Out[45]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x15dad1bae08>
```



In [46]:
```
index = expr[(expr['Mileage_Float'] >=25)].index
expr.drop(index, inplace=True)
```

In [47]:
```
sns.boxplot(x ='Mileage_Float',data = expr)
```

Out[47]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x15dad203408>
```

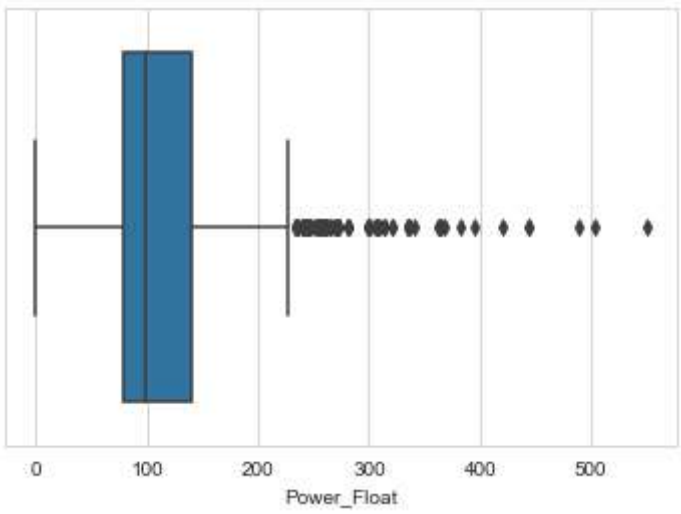


Outliers are removed from the Mileage also.

Now checking outliers from Power Column.

In [48]:
```
sns.boxplot(x ='Power_Float',data = expr)
```

Out[48]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x15dad283088>
```



In [49]:
```
Q1 = expr.Power_Float.quantile(0.25)
print(Q1)
Q3 = expr.Power_Float.quantile(0.75)
print(Q3)
IQR = Q3 - Q1
print(IQR)
print(Q1 - (1.5 * IQR))
print(Q3 + (1.5 * IQR))
```

```
78.9
140.0
61.099999999999994
-12.749999999999986
231.64999999999998
```

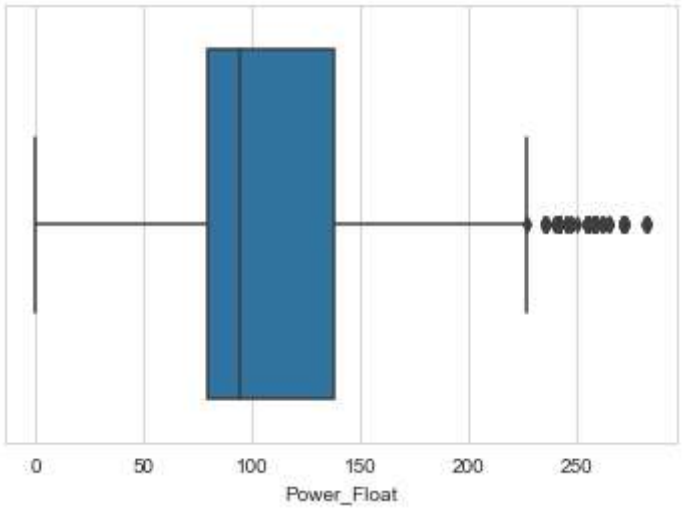
In [50]:
```
expr[~((expr.Power_Float < (Q1 - 1.5 * IQR)) |(expr.Power_Float > (Q3 + 1.5 *
IQR)))].Power_Float.median()
```

Out[50]:
```
90.0
```

In [51]:
```
index = expr[(expr['Power_Float'] >=300)].index
expr.drop(index, inplace=True)
```

In [52]:
```
sns.boxplot(x ='Power_Float',data = expr)
```

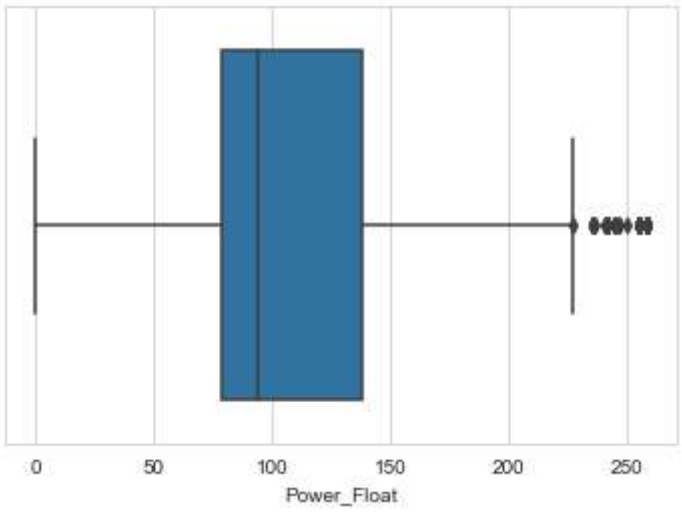
Out[52]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x15dad2e2688>
```



In [53]:
```
index = expr[(expr['Power_Float'] >=260)].index
expr.drop(index, inplace=True)
```

In [54]:
```
sns.boxplot(x ='Power_Float',data = expr)
```

Out[54]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x15dad336c08>
```



If we want we can completely remove these outliers if we go in the range 230-250 but doing so will cost a large number of data points to go missing and will result in accuracy so we will keep it like this.

****Now we have removed the outliers so we can go for machine learning for model predictions.****

MACHINE LEARNING ALGORITHMS FOR PREDICTIONS

Since from the data visualization it was quite clear that the transmission column and owner type column is playing an important role in the dataset so we will have to consider it as well. But it is a **string based** column so we will create some dummy variable to satisfy the needs of a machine learning model as machine learning model does not understand the String based text.

In [55]:
```
cleanup_nums = {"Transmission":     {"Manual": 0, "Automatic": 1},
                "Owner_Type": {"First": 1, "Second": 2, "Third": 3, "Fourth &
 Above": 4},
                "Fuel_Type": {"Diesel": 1, "Petrol": 2, "CNG": 3, "LPG": 4},
               }
```

In [56]:
```
expr.replace(cleanup_nums, inplace=True)
expr.head()
```

Out[56]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Seats
1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	1	0	1	5.0
2	Honda Jazz V	Chennai	2011	46000	2	0	1	5.0
3	Maruti Ertiga VDI	Chennai	2012	87000	1	0	1	7.0
4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	1	1	2	5.0
5	Hyundai EON LPG Era Plus Option	Hyderabad	2012	75000	4	0	1	5.0

In [57]:
```
expr.columns.tolist()
```

Out[57]:
```
['Name',
 'Location',
 'Year',
 'Kilometers_Driven',
 'Fuel_Type',
 'Transmission',
 'Owner_Type',
 'Seats',
 'Price',
 'Mileage_Float',
 'Engine_Float',
 'Power_Float']
```

In [58]:
```
expr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5311 entries, 1 to 6017
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Name               5311 non-null   object 
 1   Location           5311 non-null   object 
 2   Year               5311 non-null   int64  
 3   Kilometers_Driven  5311 non-null   int64  
 4   Fuel_Type          5311 non-null   int64  
 5   Transmission       5311 non-null   int64  
 6   Owner_Type         5311 non-null   int64  
 7   Seats              5311 non-null   float64
 8   Price              5311 non-null   float64
 9   Mileage_Float      5311 non-null   float64
 10  Engine_Float       5311 non-null   float64
 11  Power_Float        5311 non-null   float64
dtypes: float64(5), int64(5), object(2)
memory usage: 539.4+ KB
```

Now we will change the order of the dataframe so that the target column is at the last of the dataset. And also we will get rid of the useless column like name and location.

In [59]:
```
expr = expr[['Name','Location','Year','Kilometers_Driven','Fuel_Type','Transmi
ssion','Owner_Type','Seats','Mileage_Float','Engine_Float','Power_Float','Pric
e']]
```

In [60]:
```
expr.head()
```

Out[60]:

	Name	Location	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Seats
1	Hyundai Creta 1.6 CRDi SX Option	Pune	2015	41000	1	0	1	5.0
2	Honda Jazz V	Chennai	2011	46000	2	0	1	5.0
3	Maruti Ertiga VDI	Chennai	2012	87000	1	0	1	7.0
4	Audi A4 New 2.0 TDI Multitronic	Coimbatore	2013	40670	1	1	2	5.0
5	Hyundai EON LPG Era Plus Option	Hyderabad	2012	75000	4	0	1	5.0

In [61]:
```
expr.drop('Name',axis=1,inplace = True)
```

In [62]:
```
expr.drop('Location',axis=1,inplace =True)
```

In [63]:
```
expr.head()
```

Out[63]:

	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Seats	Mileage_Float	Engine
1	2015	41000	1	0	1	5.0	19.67	
2	2011	46000	2	0	1	5.0	18.20	
3	2012	87000	1	0	1	7.0	20.77	
4	2013	40670	1	1	2	5.0	15.20	
5	2012	75000	4	0	1	5.0	21.10	

Now we will fit our data for the training and testing purpose.

In [259]:
```
X = expr[['Year','Kilometers_Driven','Fuel_Type','Transmission','Owner_Type',
'Seats','Mileage_Float','Engine_Float','Power_Float']]
```

In [260]:
```
y = expr['Price']
```

In [261]:
```
from sklearn.model_selection import train_test_split
```

In [262]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size= 0.3,random_sta
te = 101)
```

FIRST MODEL WE WILL APPLY IS LINEAR REGRESSION

In [263]:
```
from sklearn.linear_model import LinearRegression
```

In [264]:
```
lm = LinearRegression()
```

In [265]:
```
lm.fit(X_train,y_train)
```

Out[265]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=Fals
e)
```

In [266]:
```
lm.coef_
```

Out[266]:
```
array([ 7.19824676e-01, -3.52025184e-05, -2.82770065e+00,  3.16480285e+00,
       -4.48940259e-02, -8.41924044e-01, -1.68290615e-01,  2.41201350e-03,
        9.43691729e-02])
```

In [267]:
```
X_train.columns
```

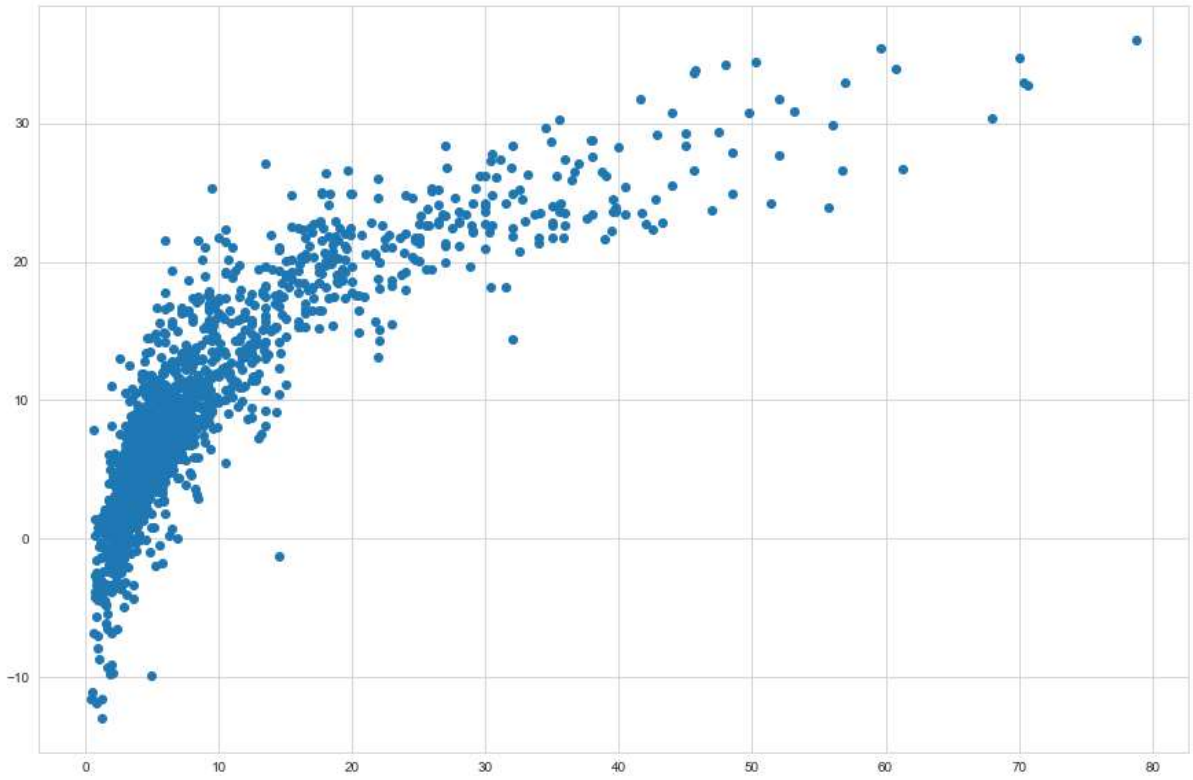
Out[267]:
```
Index(['Year', 'Kilometers_Driven', 'Fuel_Type', 'Transmission', 'Owner_Typ
e',
       'Seats', 'Mileage_Float', 'Engine_Float', 'Power_Float'],
      dtype='object')
```

In [268]:
```
cdf = pd.DataFrame(lm.coef_,X.columns,columns=['coeff'])
```

In [269]:
```
pred = lm.predict(X_test)
```

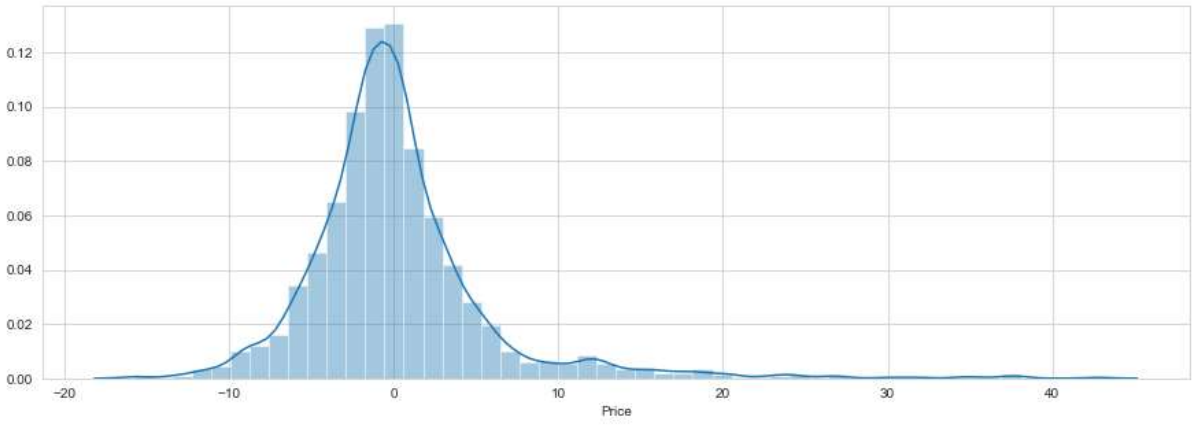
In [270]:
```
plt.figure(figsize = (15,10))
plt.scatter(y_test,pred)
```

Out[270]: <matplotlib.collections.PathCollection at 0x15dcc473bc8>



In [271]:
```
plt.figure(figsize = (15,5))
sns.distplot((y_test-pred))
```

Out[271]: <matplotlib.axes._subplots.AxesSubplot at 0x15dcc74c648>



In [272]:
```
from sklearn import metrics
```

In [273]:
```
print('MAE:', metrics.mean_absolute_error(y_test, pred))
print('MSE:', metrics.mean_squared_error(y_test, pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, pred)))
```

```
MAE: 3.5997218932270134
MSE: 32.057123228268814
RMSE: 5.661901026004323
```

By observation if we remove the year column the RMSE will change to 5.6 and with year it will change into 5.2.

SECOND MODEL WILL BE DECISION TREE

In [368]:
```
from sklearn.model_selection import train_test_split
```

In [369]:
```
X = expr.drop('Price',axis = 1)
```

In [370]:
```
y = expr['Price']
```

In [397]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3)
```

In [372]:
```
from sklearn.tree import DecisionTreeRegressor
```

In [373]:
```
dtree = DecisionTreeRegressor()
```

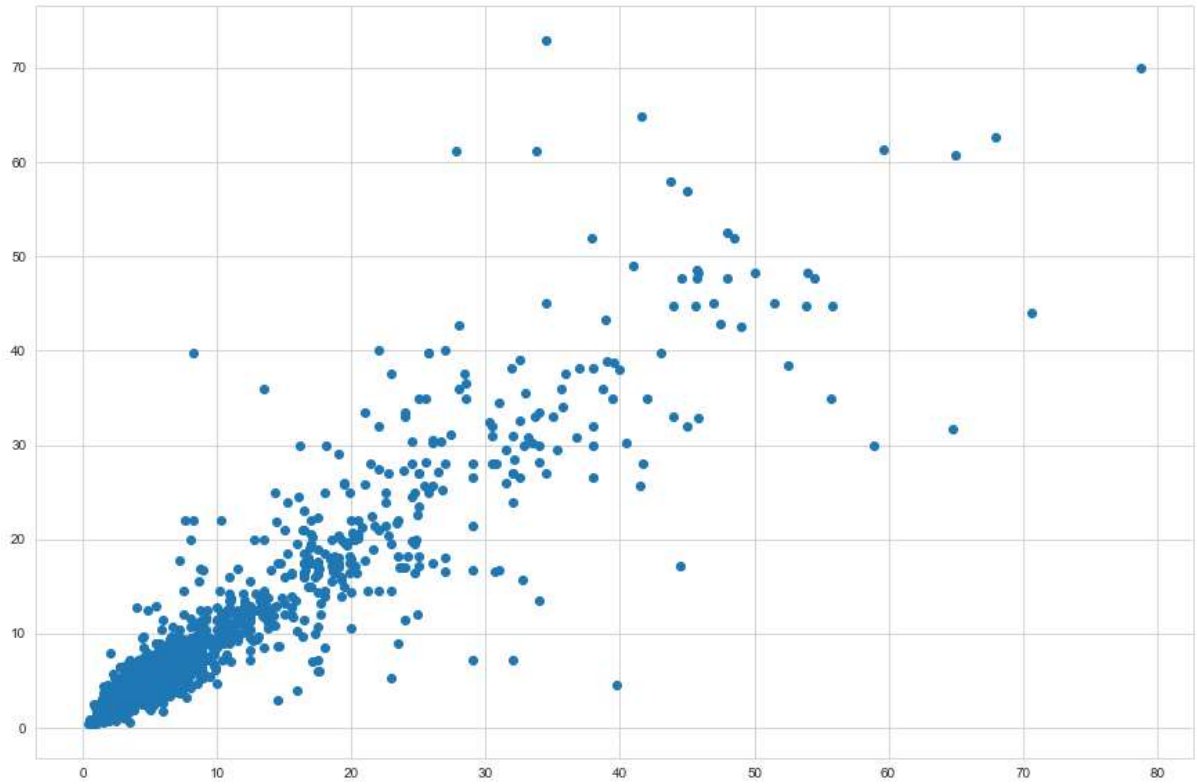
In [374]:
```
dtree.fit(X_train,y_train)
```

Out[374]:
```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=None,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=None, splitter='best')
```

In [375]:
```
pred = dtree.predict(X_test)
```

In [376]:
```
plt.figure(figsize = (15,10))
plt.scatter(y_test,pred)
```

Out[376]: <matplotlib.collections.PathCollection at 0x15dda11b1c8>



In [377]:
```
from sklearn import metrics
```

In [378]:
```
print('MAE:', metrics.mean_absolute_error(y_test, pred))
print('MSE:', metrics.mean_squared_error(y_test, pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, pred)))
```

```
MAE: 1.972766624843162
MSE: 16.876807967377665
RMSE: 4.108139234176182
```

THIRD MODEL WILL BE RANDOM FOREST

In [398]:
```
from sklearn.ensemble import RandomForestRegressor
```

In [399]:
```
rfr = RandomForestRegressor(n_estimators=200)
```

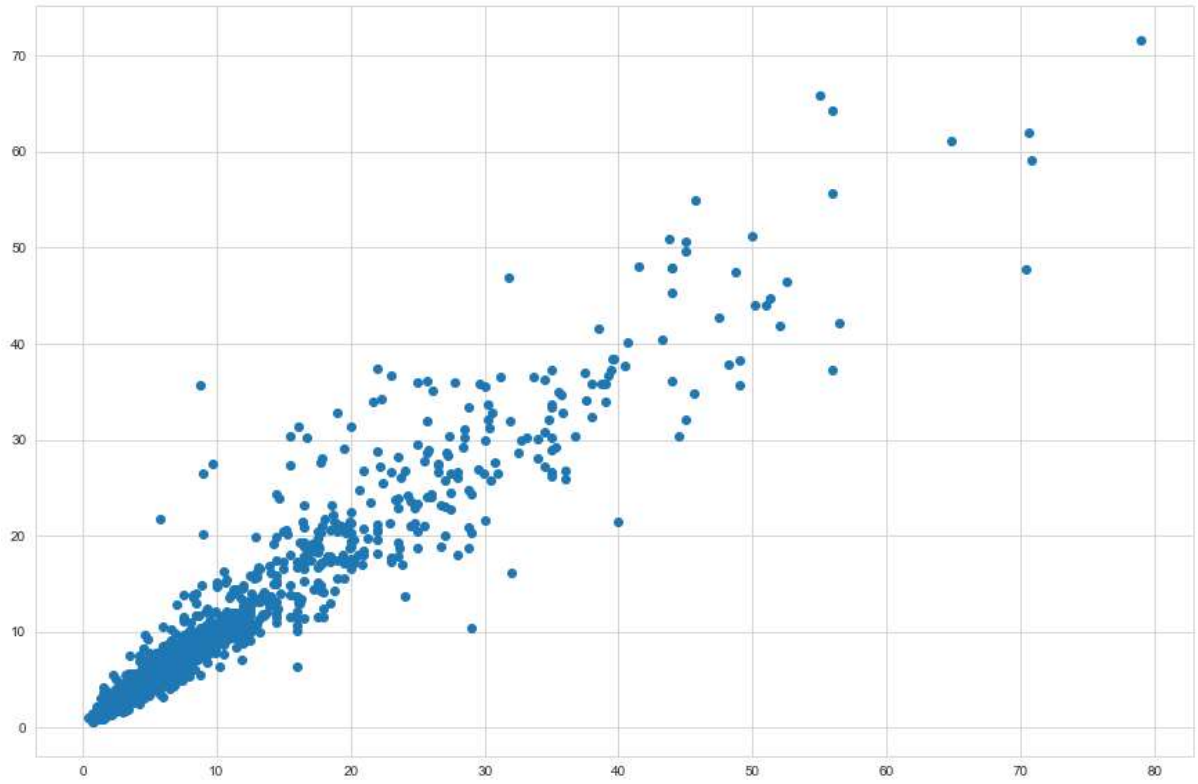
In [400]:
```
rfr.fit(X_train,y_train)
```

Out[400]:
```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=Non
e,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=200, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

In [401]:
```
rfr_pred = rfr.predict(X_test)
```

In [402]:
```
plt.figure(figsize = (15,10))
plt.scatter(y_test,rfr_pred)
```

Out[402]:
```
<matplotlib.collections.PathCollection at 0x15de6b2f188>
```



In [403]:
```
print('MAE:', metrics.mean_absolute_error(y_test, rfr_pred))
print('MSE:', metrics.mean_squared_error(y_test, rfr_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, rfr_pred)))
```

```
MAE: 1.5440944733225785
MSE: 8.735450521189101
RMSE: 2.9555795575807293
```

FOURTH MODEL WILL BE KNN

In [404]:
```
from sklearn.preprocessing import StandardScaler
```

In [405]:
```
scaler = StandardScaler()
```

In [406]:
```
scaler.fit(expr.drop('Price',axis =1))
```

Out[406]:
```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

In [407]:
```
scaled_features = scaler.transform(expr.drop('Price',axis = 1))
```

In [408]:
```
expr_feat = pd.DataFrame(scaled_features,columns = expr.columns[:-1])
```

In [409]:
```
expr_feat.head()
```

Out[409]:

	Year	Kilometers_Driven	Fuel_Type	Transmission	Owner_Type	Seats	Mileage_Float
0	0.515031	-0.490386	-0.961120	-0.639630	-0.446039	-0.388354	0.519210
1	-0.720857	-0.332682	0.982715	-0.639630	-0.446039	-0.388354	0.097054
2	-0.411885	0.960491	-0.961120	-0.639630	-0.446039	2.102645	0.835110
3	-0.102913	-0.500794	-0.961120	1.563403	1.778293	-0.388354	-0.764490
4	-0.411885	0.582001	4.870385	-0.639630	-0.446039	-0.388354	0.929880

In [410]:
```
X_train, X_test, y_train, y_test = train_test_split(scaled_features,expr['Price'],
                                                    test_size=0.30)
```

In [411]:
```
from sklearn.neighbors import KNeighborsRegressor
```

In [412]:
```
knn = KNeighborsRegressor(n_neighbors=1)
```

In [413]:
```
#Assuming the value of k =1
```

In [414]:
```
knn.fit(X_train,y_train)
```

Out[414]:
```
KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                    weights='uniform')
```

In [415]:
```
pred = knn.predict(X_test)
```

In [416]:
```
print('MAE:', metrics.mean_absolute_error(y_test, pred))
print('MSE:', metrics.mean_squared_error(y_test, pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, pred)))
```

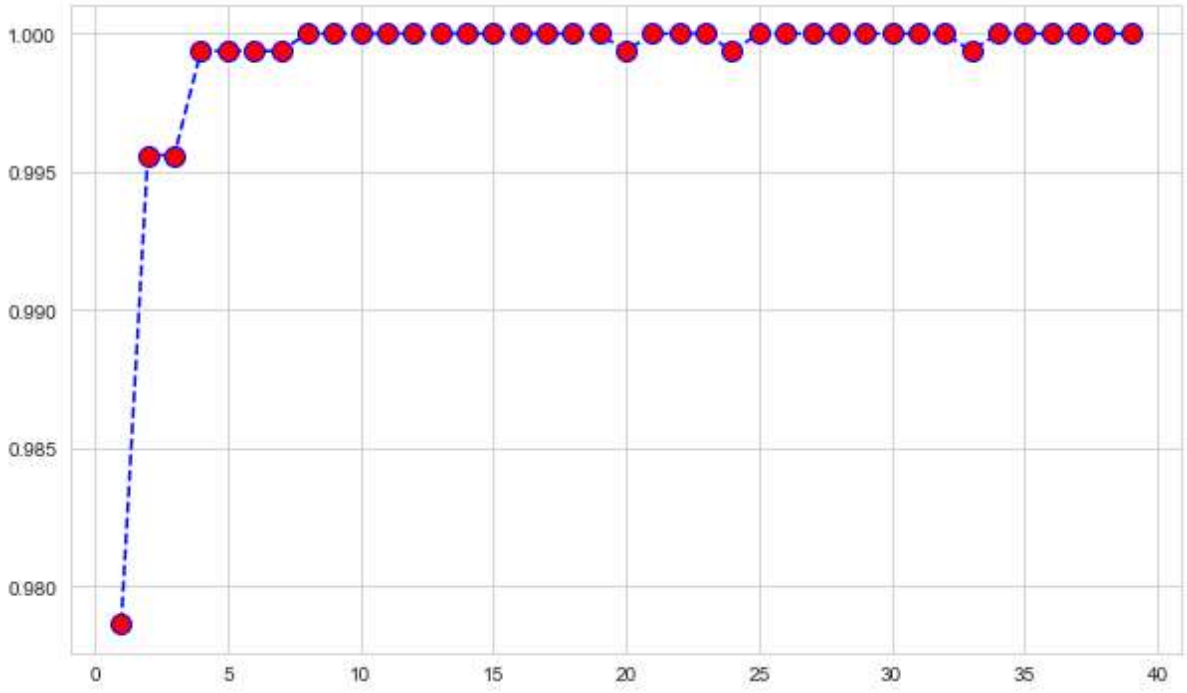
```
MAE: 1.9551380175658721
MSE: 14.854957904642411
RMSE: 3.8542130071705185
```

In [227]:
```
#The RMSE here is 3.85 so we will try to find the best suited value for k so t
hat the error becomes less.
```

In [417]:
```
error_rate = []
for i in range(1,40):
    knn = KNeighborsRegressor(n_neighbors=i)
    knn.fit(X_train,y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))
```

In [418]:
```
plt.figure(figsize = (10,6))
plt.plot(range(1,40),error_rate,color = 'blue',linestyle = 'dashed',marker =
'o',markerfacecolor='red', markersize=10)
```

Out[418]:
```
[<matplotlib.lines.Line2D at 0x15de6da38c8>]
```



In [237]:
```
#By looking at the graph it is observed that the error will keep on increasing
for higher values. So we will use k =3
```

In [419]:
```
knn = KNeighborsRegressor(n_neighbors=3)
knn.fit(X_train,y_train)
```

Out[419]:
```
KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                    weights='uniform')
```

In [420]:
```
pred = knn.predict(X_test)
```

In [421]:
```
print('MAE:', metrics.mean_absolute_error(y_test, pred))
print('MSE:', metrics.mean_squared_error(y_test, pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, pred)))
```

```
MAE: 1.8629381012128818
MSE: 14.21102952042381
RMSE: 3.769751917623202
```

In [244]:
```
#RMSE changed from 3.85 to 3.62 which is not a big difference but anyways!
```

FIFTH MODEL WILL BE SVM(SUPPORT VECTOR MACHINE)

In [422]:
```
from sklearn.model_selection import train_test_split
```

In [423]:
```
X = expr.drop('Price',axis = 1)
```

In [424]:
```
y= expr['Price']
```

In [425]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_stat
e = 101)
```

In [426]:
```
from sklearn.svm import SVR
```

In [427]:
```
model = SVR()
```

In [428]:
```
model.fit(X_train,y_train)
```

Out[428]:
```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma='scale',
    kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

In [429]:
```
predictions = model.predict(X_test)
```

In [430]:
```
from sklearn import metrics
```

In [431]:
```
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 5.770447571193602
MSE: 115.96697061162506
RMSE: 10.768796154242361
```

In [170]:
```
#The value of RMSE is not good at all we need to adjust the parameters of SVR
 like 'C' & 'gamma'
```

In [432]:
```
from sklearn.model_selection import GridSearchCV
```

In [433]:
```
param_grid = {'C':[0.1,1,10,100,1000],'gamma':[1,0.1,0.01,0.001,0.0001]}
```

In [434]:
```
grid = GridSearchCV(SVR(),param_grid,verbose = 3)
```

In [436]:
```
grid.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 25 candidates, totalling 125 fits
[CV] C=0.1, gamma=1 ..................................................

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent worke
rs.

[CV] ..................... C=0.1, gamma=1, score=-0.106, total=   0.7s
[CV] C=0.1, gamma=1 ..................................................

[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.6s remaining:    0.
0s

[CV] ..................... C=0.1, gamma=1, score=-0.120, total=   0.7s
[CV] C=0.1, gamma=1 ..................................................

[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    1.3s remaining:    0.
0s
```

```
[CV] ..... C=0.1, gamma=1, score=-0.103, total=   0.7s
[CV] C=0.1, gamma=1 ..................................................
[CV] ..... C=0.1, gamma=1, score=-0.137, total=   0.7s
[CV] C=0.1, gamma=1 ..................................................
[CV] ..... C=0.1, gamma=1, score=-0.134, total=   0.7s
[CV] C=0.1, gamma=0.1 ................................................
[CV] ..... C=0.1, gamma=0.1, score=-0.105, total=   0.7s
[CV] C=0.1, gamma=0.1 ................................................
[CV] ..... C=0.1, gamma=0.1, score=-0.119, total=   0.7s
[CV] C=0.1, gamma=0.1 ................................................
[CV] ..... C=0.1, gamma=0.1, score=-0.103, total=   0.7s
[CV] C=0.1, gamma=0.1 ................................................
[CV] ..... C=0.1, gamma=0.1, score=-0.136, total=   0.7s
[CV] C=0.1, gamma=0.1 ................................................
[CV] ..... C=0.1, gamma=0.1, score=-0.134, total=   0.7s
[CV] C=0.1, gamma=0.01 ...............................................
[CV] ..... C=0.1, gamma=0.01, score=-0.103, total=   0.7s
[CV] C=0.1, gamma=0.01 ...............................................
[CV] ..... C=0.1, gamma=0.01, score=-0.117, total=   0.7s
[CV] C=0.1, gamma=0.01 ...............................................
[CV] ..... C=0.1, gamma=0.01, score=-0.101, total=   0.7s
[CV] C=0.1, gamma=0.01 ...............................................
[CV] ..... C=0.1, gamma=0.01, score=-0.134, total=   0.7s
[CV] C=0.1, gamma=0.01 ...............................................
[CV] ..... C=0.1, gamma=0.01, score=-0.132, total=   0.7s
[CV] C=0.1, gamma=0.001 ..............................................
[CV] ..... C=0.1, gamma=0.001, score=-0.100, total=   0.7s
[CV] C=0.1, gamma=0.001 ..............................................
[CV] ..... C=0.1, gamma=0.001, score=-0.114, total=   0.7s
[CV] C=0.1, gamma=0.001 ..............................................
[CV] ..... C=0.1, gamma=0.001, score=-0.099, total=   0.7s
[CV] C=0.1, gamma=0.001 ..............................................
[CV] ..... C=0.1, gamma=0.001, score=-0.131, total=   0.7s
[CV] C=0.1, gamma=0.001 ..............................................
[CV] ..... C=0.1, gamma=0.001, score=-0.129, total=   0.7s
[CV] C=0.1, gamma=0.0001 .............................................
[CV] ..... C=0.1, gamma=0.0001, score=-0.095, total=   0.7s
[CV] C=0.1, gamma=0.0001 .............................................
[CV] ..... C=0.1, gamma=0.0001, score=-0.108, total=   0.7s
[CV] C=0.1, gamma=0.0001 .............................................
[CV] ..... C=0.1, gamma=0.0001, score=-0.094, total=   0.7s
[CV] C=0.1, gamma=0.0001 .............................................
[CV] ..... C=0.1, gamma=0.0001, score=-0.124, total=   0.7s
[CV] C=0.1, gamma=0.0001 .............................................
[CV] ..... C=0.1, gamma=0.0001, score=-0.123, total=   0.7s
[CV] C=1, gamma=1 ....................................................
[CV] ..... C=1, gamma=1, score=-0.104, total=   0.8s
[CV] C=1, gamma=1 ....................................................
[CV] ..... C=1, gamma=1, score=-0.117, total=   0.8s
[CV] C=1, gamma=1 ....................................................
[CV] ..... C=1, gamma=1, score=-0.102, total=   0.8s
[CV] C=1, gamma=1 ....................................................
[CV] ..... C=1, gamma=1, score=-0.135, total=   0.7s
[CV] C=1, gamma=1 ....................................................
[CV] ..... C=1, gamma=1, score=-0.131, total=   0.8s
[CV] C=1, gamma=0.1 ..................................................
[CV] ..... C=1, gamma=0.1, score=-0.097, total=   0.8s
```

```
[CV] C=1, gamma=0.1 ..................................................
[CV] ..................... C=1, gamma=0.1, score=-0.111, total=   0.8s
[CV] C=1, gamma=0.1 ..................................................
[CV] ..................... C=1, gamma=0.1, score=-0.097, total=   0.7s
[CV] C=1, gamma=0.1 ..................................................
[CV] ..................... C=1, gamma=0.1, score=-0.126, total=   0.8s
[CV] C=1, gamma=0.1 ..................................................
[CV] ..................... C=1, gamma=0.1, score=-0.124, total=   0.8s
[CV] C=1, gamma=0.01 .................................................
[CV] ..................... C=1, gamma=0.01, score=-0.084, total=   0.8s
[CV] C=1, gamma=0.01 .................................................
[CV] ..................... C=1, gamma=0.01, score=-0.094, total=   0.8s
[CV] C=1, gamma=0.01 .................................................
[CV] ..................... C=1, gamma=0.01, score=-0.081, total=   0.7s
[CV] C=1, gamma=0.01 .................................................
[CV] ..................... C=1, gamma=0.01, score=-0.110, total=   0.8s
[CV] C=1, gamma=0.01 .................................................
[CV] ..................... C=1, gamma=0.01, score=-0.110, total=   0.8s
[CV] C=1, gamma=0.001 ................................................
[CV] ................... C=1, gamma=0.001, score=-0.066, total=   0.8s
[CV] C=1, gamma=0.001 ................................................
[CV] ................... C=1, gamma=0.001, score=-0.070, total=   0.7s
[CV] C=1, gamma=0.001 ................................................
[CV] ................... C=1, gamma=0.001, score=-0.067, total=   0.8s
[CV] C=1, gamma=0.001 ................................................
[CV] ................... C=1, gamma=0.001, score=-0.092, total=   0.8s
[CV] C=1, gamma=0.001 ................................................
[CV] ................... C=1, gamma=0.001, score=-0.090, total=   0.8s
[CV] C=1, gamma=0.0001 ...............................................
[CV] ................... C=1, gamma=0.0001, score=-0.025, total=   0.8s
[CV] C=1, gamma=0.0001 ...............................................
[CV] ................... C=1, gamma=0.0001, score=-0.024, total=   0.8s
[CV] C=1, gamma=0.0001 ...............................................
[CV] ................... C=1, gamma=0.0001, score=-0.029, total=   0.8s
[CV] C=1, gamma=0.0001 ...............................................
[CV] ................... C=1, gamma=0.0001, score=-0.042, total=   0.8s
[CV] C=1, gamma=0.0001 ...............................................
[CV] ................... C=1, gamma=0.0001, score=-0.044, total=   0.8s
[CV] C=10, gamma=1 ...................................................
[CV] ...................... C=10, gamma=1, score=-0.019, total=   1.0s
[CV] C=10, gamma=1 ...................................................
[CV] ...................... C=10, gamma=1, score=-0.015, total=   0.9s
[CV] C=10, gamma=1 ...................................................
[CV] ...................... C=10, gamma=1, score=-0.016, total=   1.0s
[CV] C=10, gamma=1 ...................................................
[CV] ...................... C=10, gamma=1, score=-0.020, total=   0.9s
[CV] C=10, gamma=1 ...................................................
[CV] ...................... C=10, gamma=1, score=-0.021, total=   0.9s
[CV] C=10, gamma=0.1 .................................................
[CV] ..................... C=10, gamma=0.1, score=0.017, total=   0.9s
[CV] C=10, gamma=0.1 .................................................
[CV] ..................... C=10, gamma=0.1, score=0.016, total=   0.9s
[CV] C=10, gamma=0.1 .................................................
[CV] ..................... C=10, gamma=0.1, score=0.004, total=   0.9s
[CV] C=10, gamma=0.1 .................................................
[CV] ..................... C=10, gamma=0.1, score=0.018, total=   0.9s
[CV] C=10, gamma=0.1 .................................................
```

```
[CV] ..................... C=10, gamma=0.1, score=0.012, total=   0.9s
[CV] C=10, gamma=0.01 .................................................
[CV] ..................... C=10, gamma=0.01, score=0.057, total=   0.9s
[CV] C=10, gamma=0.01 .................................................
[CV] ..................... C=10, gamma=0.01, score=0.064, total=   0.9s
[CV] C=10, gamma=0.01 .................................................
[CV] ..................... C=10, gamma=0.01, score=0.069, total=   0.9s
[CV] C=10, gamma=0.01 .................................................
[CV] ..................... C=10, gamma=0.01, score=0.052, total=   0.9s
[CV] C=10, gamma=0.01 .................................................
[CV] ..................... C=10, gamma=0.01, score=0.040, total=   0.9s
[CV] C=10, gamma=0.001 ................................................
[CV] ................... C=10, gamma=0.001, score=0.115, total=   0.9s
[CV] C=10, gamma=0.001 ................................................
[CV] ................... C=10, gamma=0.001, score=0.127, total=   0.9s
[CV] C=10, gamma=0.001 ................................................
[CV] ................... C=10, gamma=0.001, score=0.115, total=   0.9s
[CV] C=10, gamma=0.001 ................................................
[CV] ................... C=10, gamma=0.001, score=0.103, total=   0.9s
[CV] C=10, gamma=0.001 ................................................
[CV] ................... C=10, gamma=0.001, score=0.106, total=   0.9s
[CV] C=10, gamma=0.0001 ...............................................
[CV] ................... C=10, gamma=0.0001, score=0.208, total=   0.9s
[CV] C=10, gamma=0.0001 ...............................................
[CV] ................... C=10, gamma=0.0001, score=0.225, total=   0.9s
[CV] C=10, gamma=0.0001 ...............................................
[CV] ................... C=10, gamma=0.0001, score=0.197, total=   0.9s
[CV] C=10, gamma=0.0001 ...............................................
[CV] ................... C=10, gamma=0.0001, score=0.223, total=   0.9s
[CV] C=10, gamma=0.0001 ...............................................
[CV] ................... C=10, gamma=0.0001, score=0.212, total=   0.9s
[CV] C=100, gamma=1 ...................................................
[CV] ..................... C=100, gamma=1, score=0.008, total=   1.0s
[CV] C=100, gamma=1 ...................................................
[CV] ..................... C=100, gamma=1, score=0.022, total=   1.0s
[CV] C=100, gamma=1 ...................................................
[CV] ..................... C=100, gamma=1, score=0.005, total=   1.0s
[CV] C=100, gamma=1 ...................................................
[CV] ..................... C=100, gamma=1, score=0.012, total=   1.0s
[CV] C=100, gamma=1 ...................................................
[CV] ..................... C=100, gamma=1, score=0.023, total=   1.0s
[CV] C=100, gamma=0.1 .................................................
[CV] ................... C=100, gamma=0.1, score=0.070, total=   1.0s
[CV] C=100, gamma=0.1 .................................................
[CV] ................... C=100, gamma=0.1, score=0.072, total=   1.0s
[CV] C=100, gamma=0.1 .................................................
[CV] ................... C=100, gamma=0.1, score=0.039, total=   1.0s
[CV] C=100, gamma=0.1 .................................................
[CV] ................... C=100, gamma=0.1, score=0.074, total=   1.0s
[CV] C=100, gamma=0.1 .................................................
[CV] ................... C=100, gamma=0.1, score=0.070, total=   1.0s
[CV] C=100, gamma=0.01 ................................................
[CV] ................... C=100, gamma=0.01, score=0.122, total=   1.0s
[CV] C=100, gamma=0.01 ................................................
[CV] ................... C=100, gamma=0.01, score=0.131, total=   1.0s
[CV] C=100, gamma=0.01 ................................................
[CV] ................... C=100, gamma=0.01, score=0.138, total=   1.0s
```

```
[CV] C=100, gamma=0.01 ...............................................
[CV] .................. C=100, gamma=0.01, score=0.102, total=   1.0s
[CV] C=100, gamma=0.01 ...............................................
[CV] .................. C=100, gamma=0.01, score=0.113, total=   1.0s
[CV] C=100, gamma=0.001 ..............................................
[CV] .................. C=100, gamma=0.001, score=0.178, total=   1.0s
[CV] C=100, gamma=0.001 ..............................................
[CV] .................. C=100, gamma=0.001, score=0.183, total=   1.0s
[CV] C=100, gamma=0.001 ..............................................
[CV] .................. C=100, gamma=0.001, score=0.188, total=   1.0s
[CV] C=100, gamma=0.001 ..............................................
[CV] .................. C=100, gamma=0.001, score=0.159, total=   1.0s
[CV] C=100, gamma=0.001 ..............................................
[CV] .................. C=100, gamma=0.001, score=0.161, total=   1.0s
[CV] C=100, gamma=0.0001 .............................................
[CV] .................. C=100, gamma=0.0001, score=0.278, total=   1.1s
[CV] C=100, gamma=0.0001 .............................................
[CV] .................. C=100, gamma=0.0001, score=0.294, total=   1.2s
[CV] C=100, gamma=0.0001 .............................................
[CV] .................. C=100, gamma=0.0001, score=0.250, total=   1.1s
[CV] C=100, gamma=0.0001 .............................................
[CV] .................. C=100, gamma=0.0001, score=0.287, total=   1.1s
[CV] C=100, gamma=0.0001 .............................................
[CV] .................. C=100, gamma=0.0001, score=0.244, total=   1.2s
[CV] C=1000, gamma=1 .................................................
[CV] ..................... C=1000, gamma=1, score=0.008, total=   1.0s
[CV] C=1000, gamma=1 .................................................
[CV] ..................... C=1000, gamma=1, score=0.022, total=   1.0s
[CV] C=1000, gamma=1 .................................................
[CV] ..................... C=1000, gamma=1, score=0.005, total=   1.0s
[CV] C=1000, gamma=1 .................................................
[CV] ..................... C=1000, gamma=1, score=0.012, total=   1.0s
[CV] C=1000, gamma=1 .................................................
[CV] ..................... C=1000, gamma=1, score=0.023, total=   1.0s
[CV] C=1000, gamma=0.1 ...............................................
[CV] .................. C=1000, gamma=0.1, score=0.070, total=   1.0s
[CV] C=1000, gamma=0.1 ...............................................
[CV] .................. C=1000, gamma=0.1, score=0.072, total=   1.0s
[CV] C=1000, gamma=0.1 ...............................................
[CV] .................. C=1000, gamma=0.1, score=0.039, total=   1.0s
[CV] C=1000, gamma=0.1 ...............................................
[CV] .................. C=1000, gamma=0.1, score=0.074, total=   1.0s
[CV] C=1000, gamma=0.1 ...............................................
[CV] .................. C=1000, gamma=0.1, score=0.070, total=   1.0s
[CV] C=1000, gamma=0.01 ..............................................
[CV] .................. C=1000, gamma=0.01, score=0.122, total=   1.2s
[CV] C=1000, gamma=0.01 ..............................................
[CV] .................. C=1000, gamma=0.01, score=0.131, total=   1.1s
[CV] C=1000, gamma=0.01 ..............................................
[CV] .................. C=1000, gamma=0.01, score=0.137, total=   1.0s
[CV] C=1000, gamma=0.01 ..............................................
[CV] .................. C=1000, gamma=0.01, score=0.102, total=   1.0s
[CV] C=1000, gamma=0.01 ..............................................
[CV] .................. C=1000, gamma=0.01, score=0.112, total=   1.0s
[CV] C=1000, gamma=0.001 .............................................
[CV] .................. C=1000, gamma=0.001, score=0.175, total=   1.2s
[CV] C=1000, gamma=0.001 .............................................
```

```
[CV] ................ C=1000, gamma=0.001, score=0.183, total=   1.1s
[CV] C=1000, gamma=0.001 .............................................
[CV] ................ C=1000, gamma=0.001, score=0.188, total=   1.2s
[CV] C=1000, gamma=0.001 .............................................
[CV] ................ C=1000, gamma=0.001, score=0.156, total=   1.1s
[CV] C=1000, gamma=0.001 .............................................
[CV] ................ C=1000, gamma=0.001, score=0.158, total=   1.3s
[CV] C=1000, gamma=0.0001 ............................................
[CV] ................ C=1000, gamma=0.0001, score=0.268, total=   1.4s
[CV] C=1000, gamma=0.0001 ............................................
[CV] ................ C=1000, gamma=0.0001, score=0.288, total=   1.6s
[CV] C=1000, gamma=0.0001 ............................................
[CV] ................ C=1000, gamma=0.0001, score=0.213, total=   1.6s
[CV] C=1000, gamma=0.0001 ............................................
[CV] ................ C=1000, gamma=0.0001, score=0.252, total=   1.5s
[CV] C=1000, gamma=0.0001 ............................................
[CV] ................ C=1000, gamma=0.0001, score=0.206, total=   1.5s

[Parallel(n_jobs=1)]: Done 125 out of 125 | elapsed:  1.9min finished
```

Out[436]:
```
GridSearchCV(cv=None, error_score=nan,
             estimator=SVR(C=1.0, cache_size=200, coef0=0.0, degree=3,
                           epsilon=0.1, gamma='scale', kernel='rbf',
                           max_iter=-1, shrinking=True, tol=0.001,
                           verbose=False),
             iid='deprecated', n_jobs=None,
             param_grid={'C': [0.1, 1, 10, 100, 1000],
                         'gamma': [1, 0.1, 0.01, 0.001, 0.0001]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=3)
```

In [437]:
```
grid.best_params_
```

Out[437]:
```
{'C': 100, 'gamma': 0.0001}
```

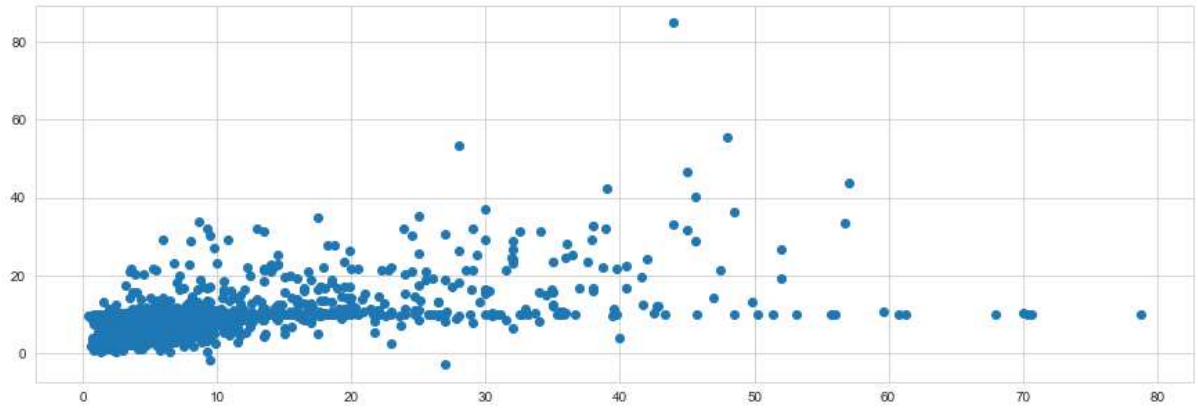
In [438]:
```
grid.best_estimator_
```

Out[438]:
```
SVR(C=100, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma=0.0001,
    kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)
```

In [439]:
```
grid_predictions = grid.predict(X_test)
```

In [440]:
```
plt.figure(figsize = (15,5))
plt.scatter(y_test,grid_predictions)
```

Out[440]: <matplotlib.collections.PathCollection at 0x15db0a30608>



In [441]:
```
print('MAE:', metrics.mean_absolute_error(y_test, grid_predictions))
print('MSE:', metrics.mean_squared_error(y_test, grid_predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, grid_predictions)))
```

```
MAE: 5.128937863948183
MSE: 75.84739959293746
RMSE: 8.709041255668586
```

In [180]:
```
#The RMSE dropped to 2% by adjusting the best parameters so this is the best S
upport Vector Machine can do.
```

FINAL INFERENCES:

- **We are judging on the basis of root mean square error (RMSE), which measures the average deviation of the actual data points from the regression line.**
- Linear Regression Model has a RMSE of 5.6619.
- Decision Tree Model has a RMSE of 4.108.
- Random Tree Model has a RMSE of 2.955.
- KNN Model has a RMSE of 3.769.
- SVM Model has the RMSE of 8.71 with best parameters.

Random Forest is the right choice for this kind of dataset. Because it has the least RMSE which is required for a good prediction model.

THANKYOU!!