Name : Tushar Shirsath

Roll No : 220940325083

Q.1

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv(r'C:\Users\Dell\Desktop\ML_Module_Exam\Data\car.csv')
df
```

Out[2]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0
3	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0
4	swift	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0
...	...	...	...	...	...	...	...	...	...
296	city	2016	9.50	11.60	33988	Diesel	Dealer	Manual	0
297	brio	2015	4.00	5.90	60000	Petrol	Dealer	Manual	0
298	city	2009	3.35	11.00	87934	Petrol	Dealer	Manual	0
299	city	2017	11.50	12.50	9000	Diesel	Dealer	Manual	0
300	brio	2016	5.30	5.90	5464	Petrol	Dealer	Manual	0

301 rows × 9 columns

1.Data understanding and exploration

In [3]:
```
df.shape
```

Out[3]: (301, 9)

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Car_Name       301 non-null    object 
 1   Year           301 non-null    int64  
 2   Selling_Price  301 non-null    float64
 3   Present_Price  301 non-null    float64
 4   Kms_Driven     301 non-null    int64  
 5   Fuel_Type      301 non-null    object 
 6   Seller_Type    301 non-null    object 
 7   Transmission   301 non-null    object 
 8   Owner          301 non-null    int64  
dtypes: float64(2), int64(3), object(4)
memory usage: 21.3+ KB
```

In [5]:
```
df.describe()
```

Out[5]:

	Year	Selling_Price	Present_Price	Kms_Driven	Owner
count	301.000000	301.000000	301.000000	301.000000	301.000000
mean	2013.627907	4.661296	7.628472	36947.205980	0.043189
std	2.891554	5.082812	8.644115	38886.883882	0.247915
min	2003.000000	0.100000	0.320000	500.000000	0.000000
25%	2012.000000	0.900000	1.200000	15000.000000	0.000000
50%	2014.000000	3.600000	6.400000	32000.000000	0.000000
75%	2016.000000	6.000000	9.900000	48767.000000	0.000000
max	2018.000000	35.000000	92.600000	500000.000000	3.000000

In [6]:
```
df.describe(include='O')
```

Out[6]:

	Car_Name	Fuel_Type	Seller_Type	Transmission
count	301	301	301	301
unique	98	3	2	2
top	city	Petrol	Dealer	Manual
freq	26	239	195	261

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Kms_Driven       0
Fuel_Type        0
Seller_Type      0
Transmission     0
Owner            0
dtype: int64
```

In [8]:
```
df.nunique()
```

Out[8]:
```
Car_Name          98
Year              16
Selling_Price    156
Present_Price    147
Kms_Driven       206
Fuel_Type          3
Seller_Type        2
Transmission       2
Owner              3
dtype: int64
```

In [10]:
```
df.shape
```

Out[10]:
```
(301, 9)
```

In [11]:
```
df.isnull().sum()/len(df) * 100
```

Out[11]:
```
Car_Name         0.0
Year             0.0
Selling_Price    0.0
Present_Price    0.0
Kms_Driven       0.0
Fuel_Type        0.0
Seller_Type      0.0
Transmission     0.0
Owner            0.0
dtype: float64
```

In [ ]:
```
# No null values are present
```

In [13]:
```
df.columns
```

Out[13]:
```
Index(['Car_Name', 'Year', 'Selling_Price', 'Present_Price', 'Kms_Driven',
       'Fuel_Type', 'Seller_Type', 'Transmission', 'Owner'],
      dtype='object')
```

In [12]:
```
# Car Name can be drop
```

In [14]:
```
df.drop(['Car_Name'], axis=1, inplace=True)
```

In [15]:
```
df.shape
```

Out[15]:
```
(301, 8)
```

In [16]:
```
df.Year.value_counts()
```

Out[16]:
```
2015    61
2016    50
2014    38
2017    35
2013    33
2012    23
2011    19
2010    15
2008     7
2009     6
2006     4
2005     4
2003     2
2007     2
2018     1
2004     1
Name: Year, dtype: int64
```

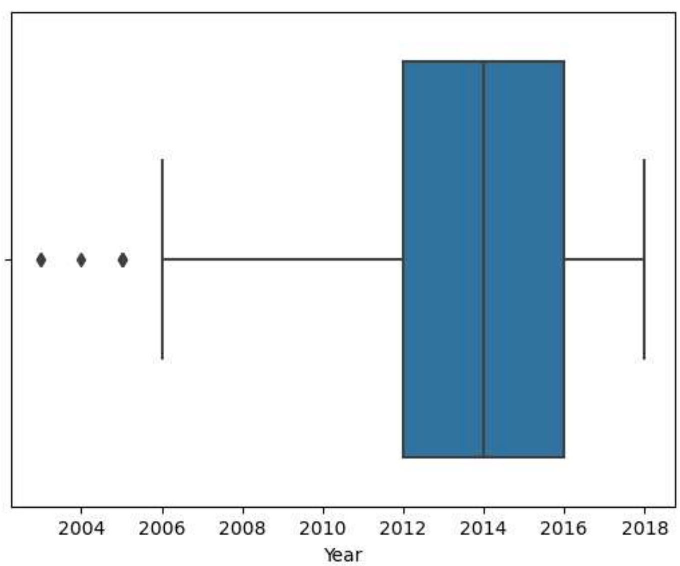
In [17]:
```
df.Year.nunique()
```

Out[17]:
```
16
```

In [18]:
```
sns.boxplot(df.Year)
```

```
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword ar
g: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit key
word will result in an error or misinterpretation.
  warnings.warn(
```

Out[18]:
```
<AxesSubplot:xlabel='Year'>
```



In [19]:
```
df.columns
```

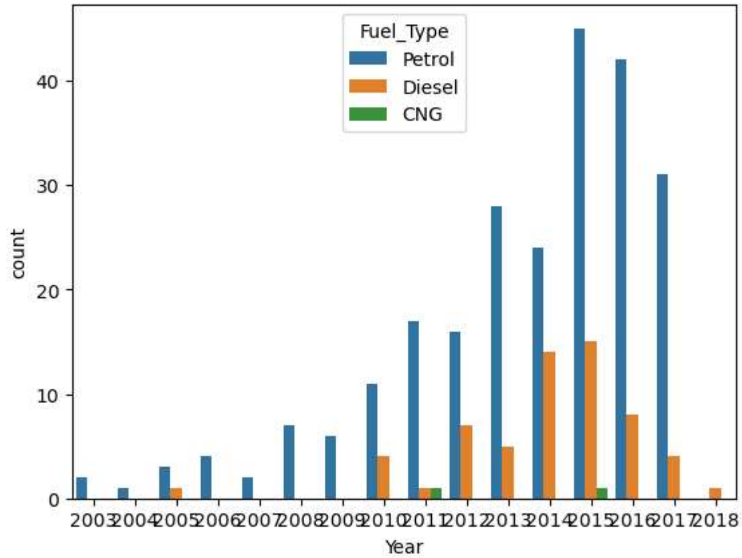
Out[19]:
```
Index(['Year', 'Selling_Price', 'Present_Price', 'Kms_Driven', 'Fuel_Type',
       'Seller_Type', 'Transmission', 'Owner'],
      dtype='object')
```

In [20]:
```
# Selling price vs Year
```

In [25]:
```
sns.countplot('Year', hue = 'Fuel_Type', data=df)
```

```
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword ar
g: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit key
word will result in an error or misinterpretation.
  warnings.warn(
```

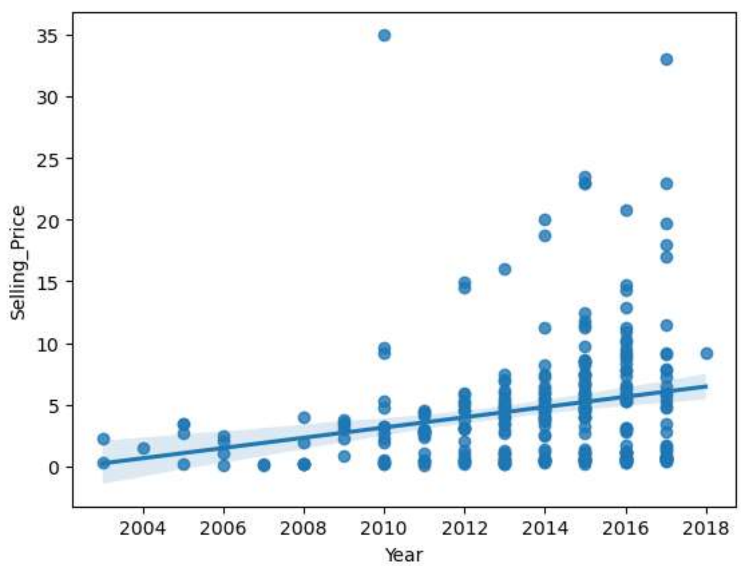
Out[25]:
```
<AxesSubplot:xlabel='Year', ylabel='count'>
```



In [21]:
```
sns.regplot('Year', 'Selling_Price', data = df)
```

```
C:\Users\Dell\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword arg
s: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[21]:
```
<AxesSubplot:xlabel='Year', ylabel='Selling_Price'>
```



In [ ]:
```
# As the year increases selling price also increases
```

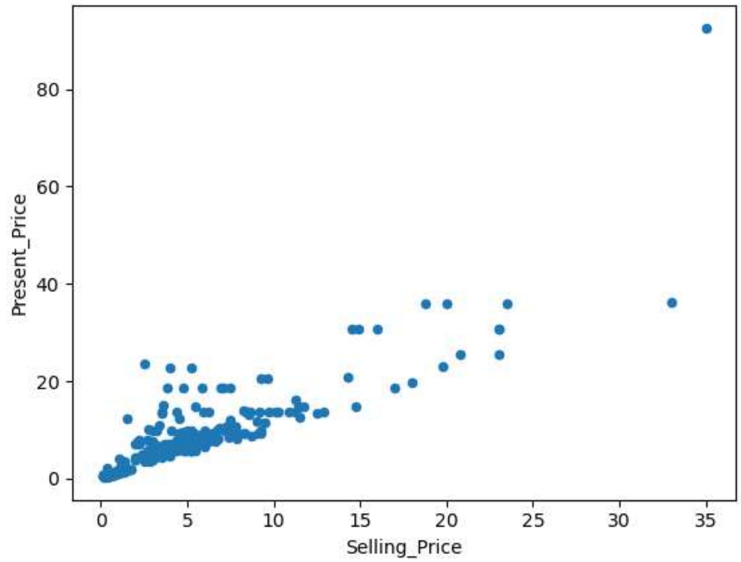
In [22]:
```
df.Selling_Price.describe()
```

Out[22]:
```
count    301.000000
mean       4.661296
std        5.082812
min        0.100000
25%        0.900000
50%        3.600000
75%        6.000000
max       35.000000
Name: Selling_Price, dtype: float64
```

In [23]:
```
# Selling Price vs Present Price
```

In [24]:
```
df.plot.scatter(x='Selling_Price', y= 'Present_Price')
```

Out[24]: <AxesSubplot:xlabel='Selling_Price', ylabel='Present_Price'>

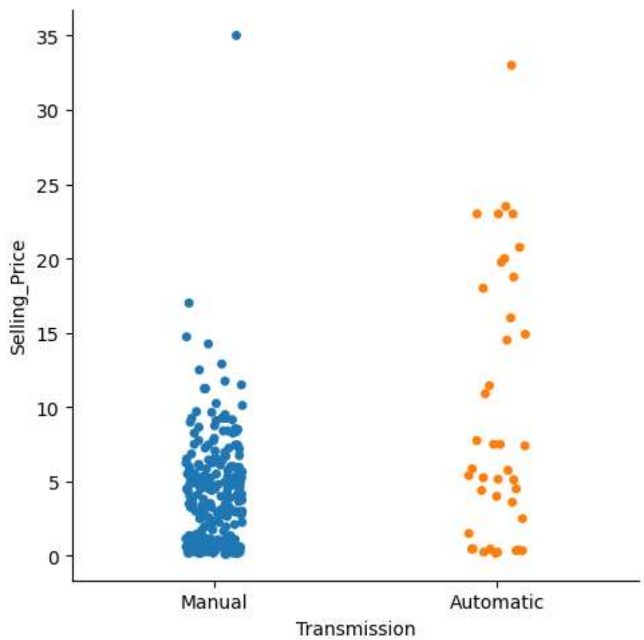


In [ ]:
```

```

In [26]:
```
sns.catplot(data=df, x='Transmission', y='Selling_Price')
```

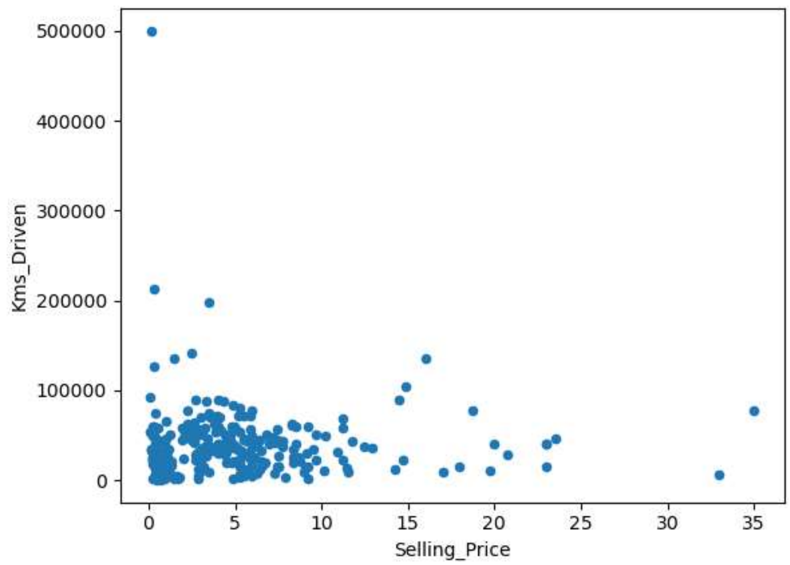
Out[26]: <seaborn.axisgrid.FacetGrid at 0x195c7447d30>



In [ ]:
```
# Automatic transmission has high selling price as compared to Manual Transmission
```

In [27]:
```
df.plot.scatter('Selling_Price', 'Kms_Driven')
```

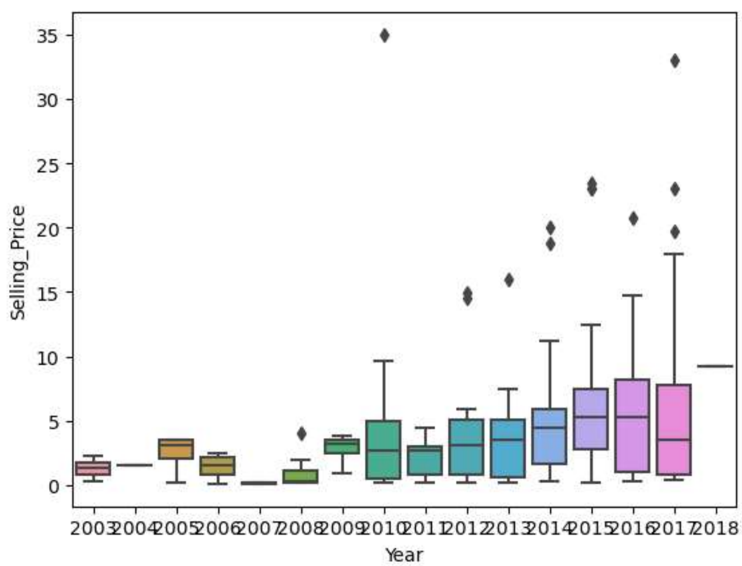
Out[27]: <AxesSubplot:xlabel='Selling_Price', ylabel='Kms_Driven'>



In [ ]:

In [30]:
```
sns.boxplot(x='Year', y='Selling_Price', data=df)
```

Out[30]: <AxesSubplot:xlabel='Year', ylabel='Selling_Price'>



2.Data cleaning

In [31]:
```
df.columns
```

Out[31]:
```
Index(['Year', 'Selling_Price', 'Present_Price', 'Kms_Driven', 'Fuel_Type',
       'Seller_Type', 'Transmission', 'Owner'],
      dtype='object')
```

In [32]:
```
df.head()
```

Out[32]:

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0

In [33]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Year           301 non-null    int64  
 1   Selling_Price  301 non-null    float64
 2   Present_Price  301 non-null    float64
 3   Kms_Driven     301 non-null    int64  
 4   Fuel_Type      301 non-null    object 
 5   Seller_Type    301 non-null    object 
 6   Transmission   301 non-null    object 
 7   Owner          301 non-null    int64  
dtypes: float64(2), int64(3), object(3)
memory usage: 18.9+ KB
```

In [34]:
```
df.Year.value_counts()
```

Out[34]:
```
2015    61
2016    50
2014    38
2017    35
2013    33
2012    23
2011    19
2010    15
2008     7
2009     6
2006     4
2005     4
2003     2
2007     2
2018     1
2004     1
Name: Year, dtype: int64
```

In [35]:
```
# Converting year column into number of years selling car is old
```

In [36]:
```
df['New_Year'] = 2022
df['Years'] = df.New_Year - df.Year
```

In [37]:
```
df.head(1)
```

Out[37]:

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	New_Year	Years
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0	2022	8

In [38]:
```
df.drop(['Year','New_Year'], axis=1, inplace = True)
```

In [39]:
```
df.head(1)
```

Out[39]:

	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Years
0	3.35	5.59	27000	Petrol	Dealer	Manual	0	8

In [40]:
```
df.shape
```

Out[40]: (301, 8)

In [41]:
```
df = pd.get_dummies(df, drop_first=True)
df.head()
```

Out[41]:

	Selling_Price	Present_Price	Kms_Driven	Owner	Years	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
0	3.35	5.59	27000	0	8	0	1	0	1
1	4.75	9.54	43000	0	9	1	0	0	1
2	7.25	9.85	6900	0	5	0	1	0	1
3	2.85	4.15	5200	0	11	0	1	0	1
4	4.60	6.87	42450	0	8	1	0	0	1

In [42]:
```
df.shape
```

Out[42]: (301, 9)

In [43]:
```
# sns.pairplot(data=df)
```

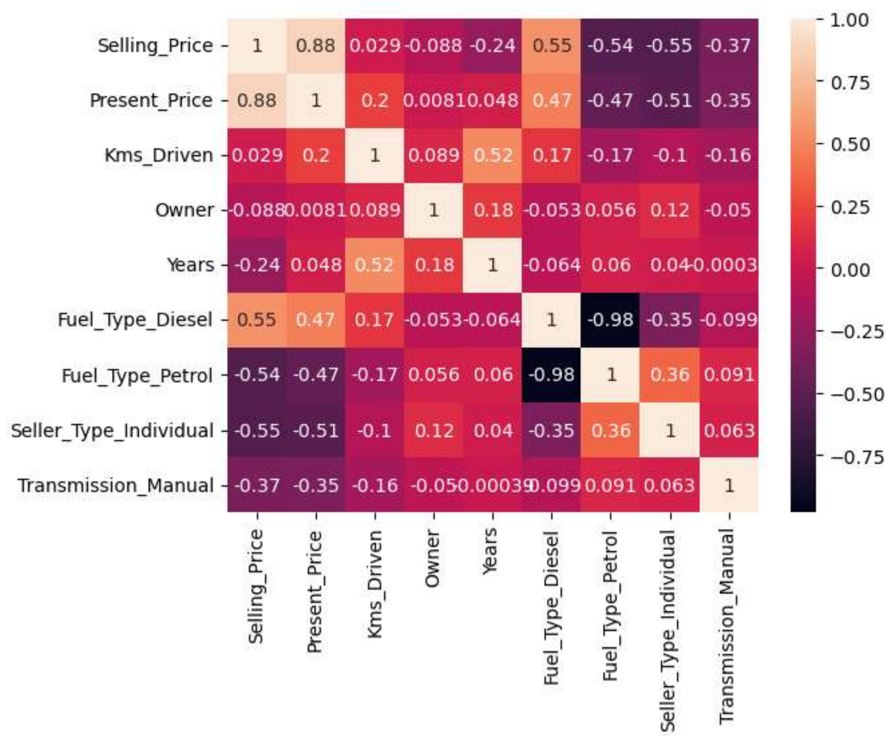
In [44]:
```
df.corr()
```

Out[44]:

	Selling_Price	Present_Price	Kms_Driven	Owner	Years	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_
Selling_Price	1.000000	0.878983	0.029187	-0.088344	-0.236141	0.552339	-0.540571	-0.550724	-0.
Present_Price	0.878983	1.000000	0.203647	0.008057	0.047584	0.473306	-0.465244	-0.512030	-0.
Kms_Driven	0.029187	0.203647	1.000000	0.089216	0.524342	0.172515	-0.172874	-0.101419	-0.
Owner	-0.088344	0.008057	0.089216	1.000000	0.182104	-0.053469	0.055687	0.124269	-0.
Years	-0.236141	0.047584	0.524342	0.182104	1.000000	-0.064315	0.059959	0.039896	-0.
Fuel_Type_Diesel	0.552339	0.473306	0.172515	-0.053469	-0.064315	1.000000	-0.979648	-0.350467	-0.
Fuel_Type_Petrol	-0.540571	-0.465244	-0.172874	0.055687	0.059959	-0.979648	1.000000	0.358321	0.
Seller_Type_Individual	-0.550724	-0.512030	-0.101419	0.124269	0.039896	-0.350467	0.358321	1.000000	0.
Transmission_Manual	-0.367128	-0.348715	-0.162510	-0.050316	-0.000394	-0.098643	0.091013	0.063240	1.

In [45]:
```
sns.heatmap(df.corr(), annot=True)
```

Out[45]: <AxesSubplot:>



3.Data preparation for Model Building

In [46]:
```
x = df.drop('Selling_Price', axis=1)
y = df['Selling_Price']
```

In [47]:
```
x
```

Out[47]:

	Present_Price	Kms_Driven	Owner	Years	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
0	5.59	27000	0	8	0	1	0	1
1	9.54	43000	0	9	1	0	0	1
2	9.85	6900	0	5	0	1	0	1
3	4.15	5200	0	11	0	1	0	1
4	6.87	42450	0	8	1	0	0	1
...	...	...	...	...	...	...	...	...
296	11.60	33988	0	6	1	0	0	1
297	5.90	60000	0	7	0	1	0	1
298	11.00	87934	0	13	0	1	0	1
299	12.50	9000	0	5	1	0	0	1
300	5.90	5464	0	6	0	1	0	1

301 rows × 8 columns

In [48]:
```
y
```

Out[48]:
```
0       3.35
1       4.75
2       7.25
3       2.85
4       4.60
       ...  
296     9.50
297     4.00
298     3.35
299    11.50
300     5.30
Name: Selling_Price, Length: 301, dtype: float64
```

In [49]:
```
x.shape, y.shape
```

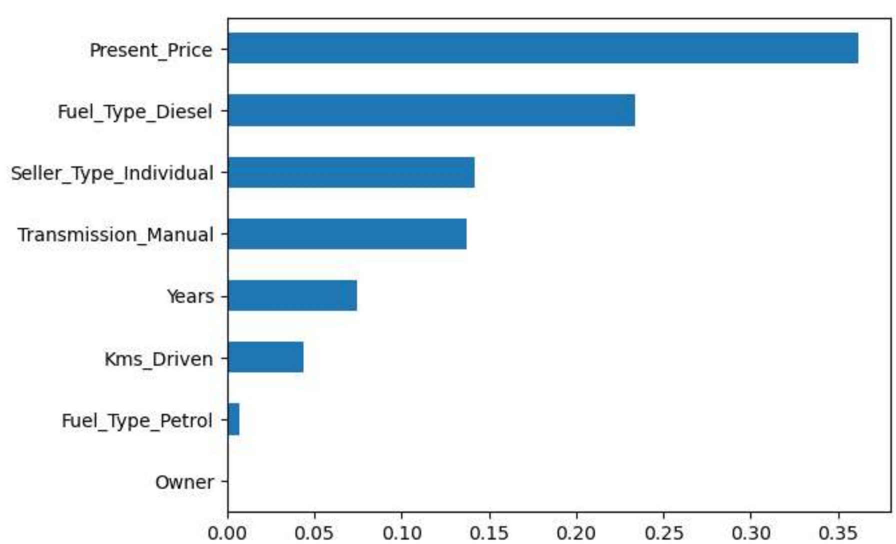
Out[49]:
```
((301, 8), (301,))
```

In [ ]:

In [50]:
```
from sklearn.ensemble import ExtraTreesRegressor
etr = ExtraTreesRegressor()
etr.fit(x,y)
```

Out[50]:
```
ExtraTreesRegressor()
```

In [51]:
```
feature = pd.Series(etr.feature_importances_, index=x.columns).sort_values(ascending=True)
feature.plot(kind = 'barh')
plt.show()
```



In [ ]:

In [52]:
```
from sklearn.model_selection import train_test_split
```

In [53]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,train_size=0.8, random_state=42)
```

In [54]:
```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

Out[54]:
```
((240, 8), (61, 8), (240,), (61,))
```

In [ ]:

In [55]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [56]:
```
sc = MinMaxScaler()
```

In [57]:
```
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [58]:
```
x_train, x_test
```

Out[58]:
```
(array([[0.00465973, 0.05105105, 0.33333333, ..., 1.        , 1.        ,
         1.        ],
        [0.00682705, 0.00600601, 0.        , ..., 1.        , 1.        ,
         1.        ],
        [0.00506068, 0.0990991 , 0.        , ..., 1.        , 1.        ,
         1.        ],
        ...,
        [0.03391851, 0.03203203, 0.33333333, ..., 1.        , 1.        ,
         1.        ],
        [0.10489814, 0.13781982, 0.        , ..., 1.        , 0.        ,
         1.        ],
        [0.01582141, 0.00700701, 0.        , ..., 1.        , 1.        ,
         1.        ]]),
 array([[ 0.00270915,  0.04704705,  0.        ,  0.07142857,  0.        ,
          1.        ,  1.        ,  0.        ],
        [ 0.14390984,  0.02098098,  0.        ,  0.07142857,  0.        ,
          1.        ,  0.        ,  1.        ],
        [ 0.09839619,  0.11911912,  0.        ,  0.35714286,  1.        ,
          0.        ,  0.        ,  1.        ],
```

In [ ]:

4.Model building and evaluation

Random Forest Regressor

In [63]:
```
from sklearn.ensemble import RandomForestRegressor
```

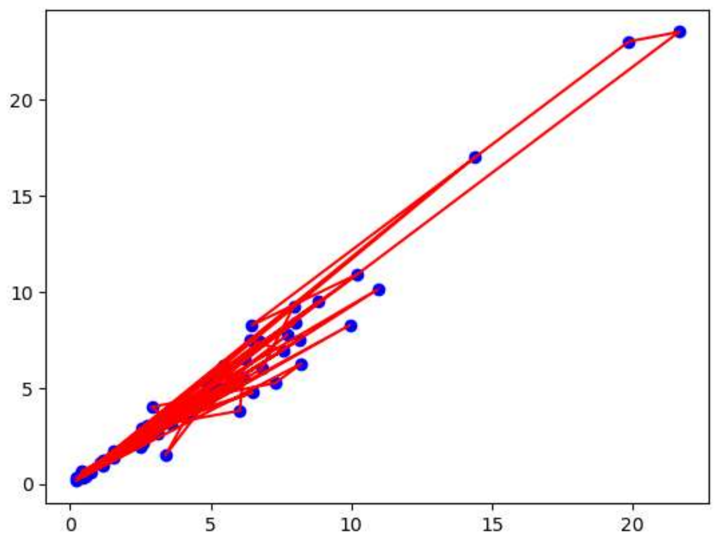
In [64]:
```
model = RandomForestRegressor()
model.fit(x_train, y_train)
model.score(x_test, y_test)
```

Out[64]:
```
0.9590803492264692
```

In [71]:
```
y_pred = model.predict(x_test)
y_pred
```

Out[71]:
```
array([ 0.4437, 10.9463,  4.9125,  0.2212,  7.5805,  6.41  ,  1.0574,
        0.57  ,  0.4713,  6.8295,  7.9604,  1.0773,  8.1506,  0.4491,
        5.44  ,  2.6155,  1.158 , 14.4091,  0.4771,  1.5455,  0.3337,
        7.9959,  4.85  ,  2.746 ,  0.5072,  3.4855,  5.3825,  3.139 ,
        1.2138,  1.1585,  0.4162,  9.9475,  0.4603,  2.5245,  7.7289,
        4.2315,  6.1575,  6.036 ,  2.5585,  6.504 ,  4.1598,  3.4018,
        4.971 ,  0.5703,  6.2125,  0.7568,  8.2245,  7.3145,  2.9165,
        3.6135,  5.02  ,  1.552 , 21.6622, 19.8743,  6.472 , 10.2125,
        5.066 ,  8.8477,  2.593 ,  6.7579,  0.2407])
```

In [72]:
```
plt.scatter(y_pred, y_test, color = 'Blue')
plt.plot(y_pred, y_test, color = 'red')
```
Out[72]:
```
[<matplotlib.lines.Line2D at 0x195cbcd0550>]
```



In [ ]:

5.Result with error calculation

In [73]:
```
from sklearn import metrics
```

In [79]:
```
# Mean Absolute error
round(metrics.mean_absolute_error(y_test, y_pred),2)
```
Out[79]:
```
0.64
```

In [80]:
```
# Mean Squared error
round(metrics.mean_squared_error(y_test, y_pred),2)
```
Out[80]:
```
0.94
```

In [81]:
```
# Median Absolute Error
round(metrics.median_absolute_error(y_test, y_pred),2)
```
Out[81]:
```
0.43
```

In [82]:
```
# Explain Varience Factor
round(metrics.explained_variance_score(y_test, y_pred),2)
```
Out[82]:
```
0.96
```

In [83]:
```
# Model Score
model.score(x_test, y_test)
```
Out[83]:
```
0.9590803492264692
```

In [84]:
```
# Model r2- score
metrics.r2_score(y_test, y_pred)
```
Out[84]:
```
0.9590803492264692
```

In [ ]:

Conclusion

```
- For model building, top 3 important features are - 1.Present_Price, 2.Fuel Type Diesel, 3.Seller Type Indidvidual
- I have got accuracy upto 95 % using Random Forest Regressor Model
- Automatic transmission has high selling price as compared to Manual Transmission
- As the year increases selling price also increases
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: