In [64]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
```

In [65]:
```
df=pd.read_csv("E:/Full Stack Data Scientist Bootcamp/project resources-2023
```

In [66]:
```
df
```

Out[66]:

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

301 rows × 9 columns

In [67]:
```
df.columns
```

Out[67]:
```
Index(['Car_Name', 'Year', 'Selling_Price', 'Present_Price', 'Kms_Driven',
       'Fuel_Type', 'Seller_Type', 'Transmission', 'Owner'],
      dtype='object')
```

In [68]:
```
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 [69]:
```
df.isna().sum()
```

Out[69]:
```
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 [70]:
```
df['Car_Name'].value_counts()
```

Out[70]:
```
city                        26
corolla altis               16
verna                       14
fortuner                    11
brio                        10
                            ..
Honda CB Trigger             1
Yamaha FZ S                  1
Bajaj Pulsar 135 LS          1
Activa 4g                    1
Bajaj Avenger Street 220     1
Name: Car_Name, Length: 98, dtype: int64
```

In [71]:
```
df.head()
```

Out[71]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Selle
0	ritz	2014	3.35	5.59	27000	Petrol	
1	sx4	2013	4.75	9.54	43000	Diesel	
2	ciaz	2017	7.25	9.85	6900	Petrol	
3	wagon r	2011	2.85	4.15	5200	Petrol	
4	swift	2014	4.60	6.87	42450	Diesel	

In [72]:
```
df.groupby(['Car_Name','Year','Selling_Price','Present_Price','Kms_Driven','
```

Out[72]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Sell
0	800	2003	0.35	2.280	127000	Petrol	I
1	Activa 3g	2008	0.17	0.520	500000	Petrol	I
2	Activa 3g	2016	0.45	0.540	500	Petrol	I
3	Activa 4g	2017	0.40	0.510	1300	Petrol	I
4	Bajaj ct 100	2015	0.18	0.320	35000	Petrol	I
5	Bajaj Avenger 150	2016	0.75	0.800	7000	Petrol	I
6	Bajaj Avenger 150 street	2016	0.60	0.800	20000	Petrol	I
7	Bajaj Avenger 220	2016	0.72	0.950	500	Petrol	I
8	Bajaj Avenger 220	2017	0.75	0.950	3500	Petrol	I
9	Bajaj Avenger 220	2017	0.90	0.950	1300	Petrol	I
10	Bajaj Avenger 220 dtsi	2010	0.45	0.950	27000	Petrol	I
11	Bajaj Avenger 220 dtsi	2015	0.60	0.950	16600	Petrol	I
12	Bajaj Avenger Street 220	2011	0.45	0.950	24000	Petrol	I
13	Bajaj Discover 100	2013	0.27	0.470	21000	Petrol	I
14	Bajaj Discover 125	2011	0.15	0.570	35000	Petrol	I
15	Bajaj Discover 125	2012	0.20	0.570	25000	Petrol	I
16	Bajaj Dominar 400	2017	1.45	1.600	1200	Petrol	I
17	Bajaj Pulsar NS 200	2014	0.60	0.990	25000	Petrol	I

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Sell
18	Bajaj Pulsar 135 LS	2014	0.40	0.640	13700	Petrol	I
19	Bajaj Pulsar 150	2006	0.10	0.750	92233	Petrol	I
20	Bajaj Pulsar 150	2008	0.20	0.750	60000	Petrol	I
21	Bajaj Pulsar 150	2008	0.25	0.750	26000	Petrol	I
22	Bajaj Pulsar 150	2015	0.65	0.740	5000	Petrol	I
23	Bajaj Pulsar 220 F	2010	0.52	0.940	45000	Petrol	I
24	Bajaj Pulsar 220 F	2016	0.51	0.940	24000	Petrol	I
25	Bajaj Pulsar NS 200	2012	0.45	0.990	14500	Petrol	I
26	Bajaj Pulsar NS 200	2012	0.50	0.990	13000	Petrol	I
27	Bajaj Pulsar NS 200	2013	0.50	0.990	45000	Petrol	I
28	Bajaj Pulsar RS200	2016	1.05	1.260	5700	Petrol	I
29	Hero CBZ Xtreme	2008	0.20	0.787	50000	Petrol	I
30	Hero Ignitor Disc	2013	0.20	0.650	24000	Petrol	I
31	Hero Extreme	2013	0.65	0.787	16000	Petrol	I
32	Hero Extreme	2014	0.55	0.787	15000	Petrol	I
33	Hero Glamour	2013	0.25	0.570	18000	Petrol	I
34	Hero Honda CBZ extreme	2011	0.38	0.787	75000	Petrol	I
35	Hero Honda	2012	0.30	0.510	60000	Petrol	I

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Sell
	Passion Pro						
36	Hero Hunk	2007	0.20	0.750	49000	Petrol	I
37	Hero Passion Pro	2015	0.40	0.550	6700	Petrol	I
38	Hero Passion Pro	2016	0.45	0.550	1000	Petrol	I
39	Hero Passion X pro	2016	0.50	0.550	31000	Petrol	I
40	Hero Splender Plus	2016	0.30	0.480	50000	Petrol	I
41	Hero Splender iSmart	2015	0.40	0.540	14000	Petrol	I
42	Hero Splender iSmart	2016	0.45	0.540	14000	Petrol	I
43	Hero Super Splendor	2005	0.20	0.570	55000	Petrol	I
44	Honda Activa 125	2016	0.35	0.570	24000	Petrol	I
45	Honda Activa 4G	2017	0.45	0.510	4000	Petrol	I
46	Honda Activa 4G	2017	0.48	0.510	4300	Petrol	I
47	Honda CB Hornet 160R	2016	0.60	0.870	15000	Petrol	I
48	Honda CB Hornet 160R	2017	0.75	0.870	11000	Petrol	I
49	Honda CB Hornet 160R	2017	0.80	0.870	3000	Petrol	I
50	Honda CB Shine	2007	0.12	0.580	53000	Petrol	I
51	Honda CB Shine	2013	0.30	0.580	30000	Petrol	I
52	Honda CB Trigger	2013	0.42	0.730	12000	Petrol	I
53	Honda CB Unicorn	2015	0.38	0.720	38600	Petrol	I

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Sell
54	Honda CB twister	2010	0.16	0.510	33000	Petrol	I
55	Honda CB twister	2013	0.25	0.510	32000	Petrol	I
56	Honda CBR 150	2013	0.60	1.200	32000	Petrol	I
57	Honda CBR 150	2014	0.65	1.200	23500	Petrol	I
58	Honda Dream Yuga	2017	0.48	0.540	8600	Petrol	I
59	Honda Karizma	2010	0.31	1.050	213000	Petrol	I

In [73]:
```
df[df['Car_Name']=='800']
```

Out[73]:

	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Sell
37	800	2003	0.35	2.28	127000	Petrol	I

In [74]:
```
df1=pd.concat([df,df],axis=0)
```

In [75]:
```
df1
```

Out[75]:

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

602 rows × 9 columns

In [76]:
```
df1.groupby(['Car_Name','Year','Selling_Price','Present_Price','Kms_Driven',
```

Out[76]:
```
Car_Name         299
Year             299
Selling_Price    299
Present_Price    299
Kms_Driven       299
Fuel_Type        299
Seller_Type      299
Transmission     299
Owner            299
0                299
dtype: int64
```

In [77]:
```
df
```

Out[77]:

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

301 rows × 9 columns

In [78]:
```
df=df.drop('Car_Name',axis=1)
```

In [79]:
```
df
```

Out[79]:

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

301 rows × 8 columns

In [80]:
```
# Which arre the categroical feature in our data set
# i think the Fuel_Type,Transsmission,Owner,Seller_Type
# are the Categorical Column in Our dataste

# so count and see the unique value in the categorical column:
```

In [81]:
```
df['Fuel_Type'].unique()
```

Out[81]:
```
array(['Petrol', 'Diesel', 'CNG'], dtype=object)
```

In [82]:
```
df['Owner'].unique()
```

Out[82]:
```
array([0, 1, 3], dtype=int64)
```

In [83]:
```
df['Transmission'].unique()
```

Out[83]:
```
array(['Manual', 'Automatic'], dtype=object)
```

In [84]:
```
df['Seller_Type'].unique()
```

Out[84]:
```
array(['Dealer', 'Individual'], dtype=object)
```

In [85]:
```
# Making the new column from the Year Colmn
```

In [86]:
```
df['Year'].unique()
```

Out[86]:
```
array([2014, 2013, 2017, 2011, 2018, 2015, 2016, 2009, 2010, 2012, 2003,
       2008, 2006, 2005, 2004, 2007], dtype=int64)
```

In [87]:
```
df['Current_Year']=2020
```

In [88]:
```
# df['Year_count']= 2020 - ( [for i in df['Year']])
df['no_year']=df['Current_Year'] -df['Year']
```

In [89]:
```
df.head()
```

Out[89]:

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

In [90]:
```
list_year_drop=['Year','Current_Year']
df.drop(list_year_drop,axis=1,inplace=True)
```

In [91]:
```
df
```

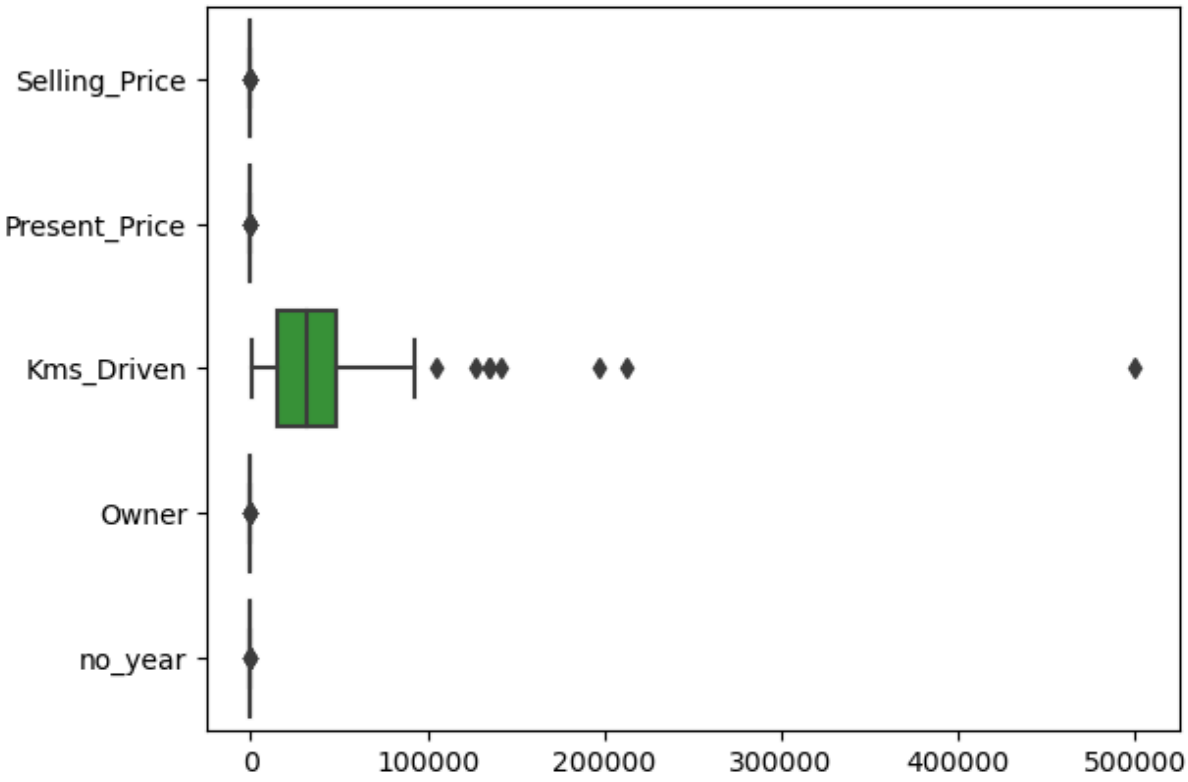
Out[91]:

	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transm
0	3.35	5.59	27000	Petrol	Dealer	M
1	4.75	9.54	43000	Diesel	Dealer	M
2	7.25	9.85	6900	Petrol	Dealer	M
3	2.85	4.15	5200	Petrol	Dealer	M
4	4.60	6.87	42450	Diesel	Dealer	M
...	...	...	...	...	...	
296	9.50	11.60	33988	Diesel	Dealer	M
297	4.00	5.90	60000	Petrol	Dealer	M
298	3.35	11.00	87934	Petrol	Dealer	M
299	11.50	12.50	9000	Diesel	Dealer	M
300	5.30	5.90	5464	Petrol	Dealer	M

301 rows × 8 columns

In [92]:
```
sns.boxplot(data=df,orient='h')
```

Out[92]:
```
<AxesSubplot:>
```



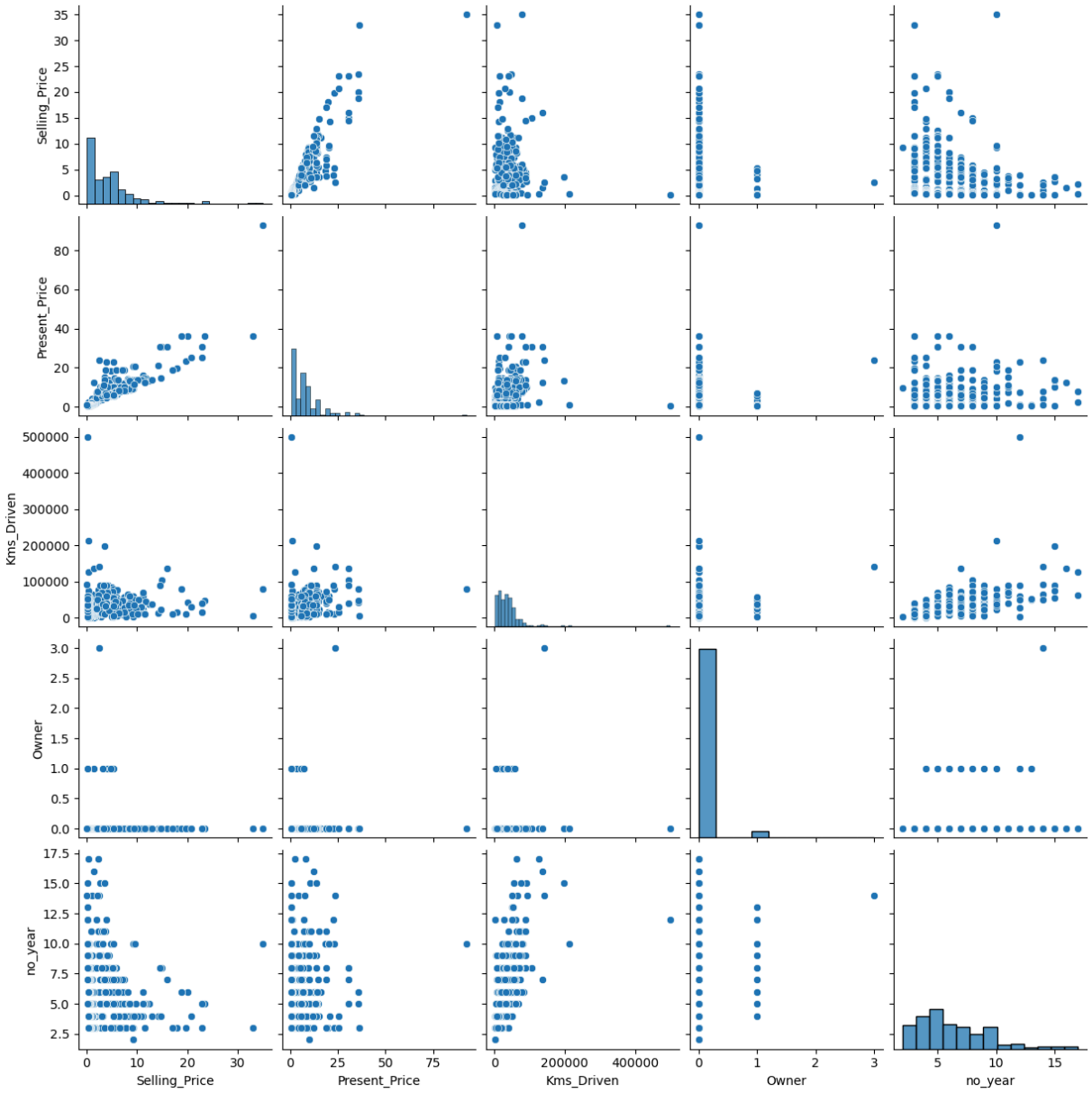
In [93]:
```
df.describe()
```

Out[93]:

	Selling_Price	Present_Price	Kms_Driven	Owner	no_year
count	301.000000	301.000000	301.000000	301.000000	301.000000
mean	4.661296	7.628472	36947.205980	0.043189	6.372093
std	5.082812	8.644115	38886.883882	0.247915	2.891554
min	0.100000	0.320000	500.000000	0.000000	2.000000
25%	0.900000	1.200000	15000.000000	0.000000	4.000000
50%	3.600000	6.400000	32000.000000	0.000000	6.000000
75%	6.000000	9.900000	48767.000000	0.000000	8.000000
max	35.000000	92.600000	500000.000000	3.000000	17.000000

In [94]:
```
sns.pairplot(df)
```

Out[94]:
```
<seaborn.axisgrid.PairGrid at 0x1e44719f640>
```



In [95]:
```
df=pd.get_dummies(df,drop_first=True)
```

In [96]:
```
df
```

Out[96]:

	Selling_Price	Present_Price	Kms_Driven	Owner	no_year	Fuel_Type_Die
0	3.35	5.59	27000	0	6	
1	4.75	9.54	43000	0	7	
2	7.25	9.85	6900	0	3	
3	2.85	4.15	5200	0	9	
4	4.60	6.87	42450	0	6	
...	...	...	...	...	...	
296	9.50	11.60	33988	0	4	
297	4.00	5.90	60000	0	5	
298	3.35	11.00	87934	0	11	
299	11.50	12.50	9000	0	3	
300	5.30	5.90	5464	0	4	

301 rows × 9 columns

In [97]:
```
df.columns
```

Out[97]:
```
Index(['Selling_Price', 'Present_Price', 'Kms_Driven', 'Owner', 'no_year',
       'Fuel_Type_Diesel', 'Fuel_Type_Petrol', 'Seller_Type_Individual',
       'Transmission_Manual'],
      dtype='object')
```

In [98]:
```
df.corr()
```

Out[98]:

	Selling_Price	Present_Price	Kms_Driven	Owner	n
Selling_Price	1.000000	0.878983	0.029187	-0.088344	-0.
Present_Price	0.878983	1.000000	0.203647	0.008057	0.
Kms_Driven	0.029187	0.203647	1.000000	0.089216	0.
Owner	-0.088344	0.008057	0.089216	1.000000	0.
no_year	-0.236141	0.047584	0.524342	0.182104	1.
Fuel_Type_Diesel	0.552339	0.473306	0.172515	-0.053469	-0.
Fuel_Type_Petrol	-0.540571	-0.465244	-0.172874	0.055687	0.
Seller_Type_Individual	-0.550724	-0.512030	-0.101419	0.124269	0.
Transmission_Manual	-0.367128	-0.348715	-0.162510	-0.050316	-0.

In [99]:
```
df.describe()
```

Out[99]:

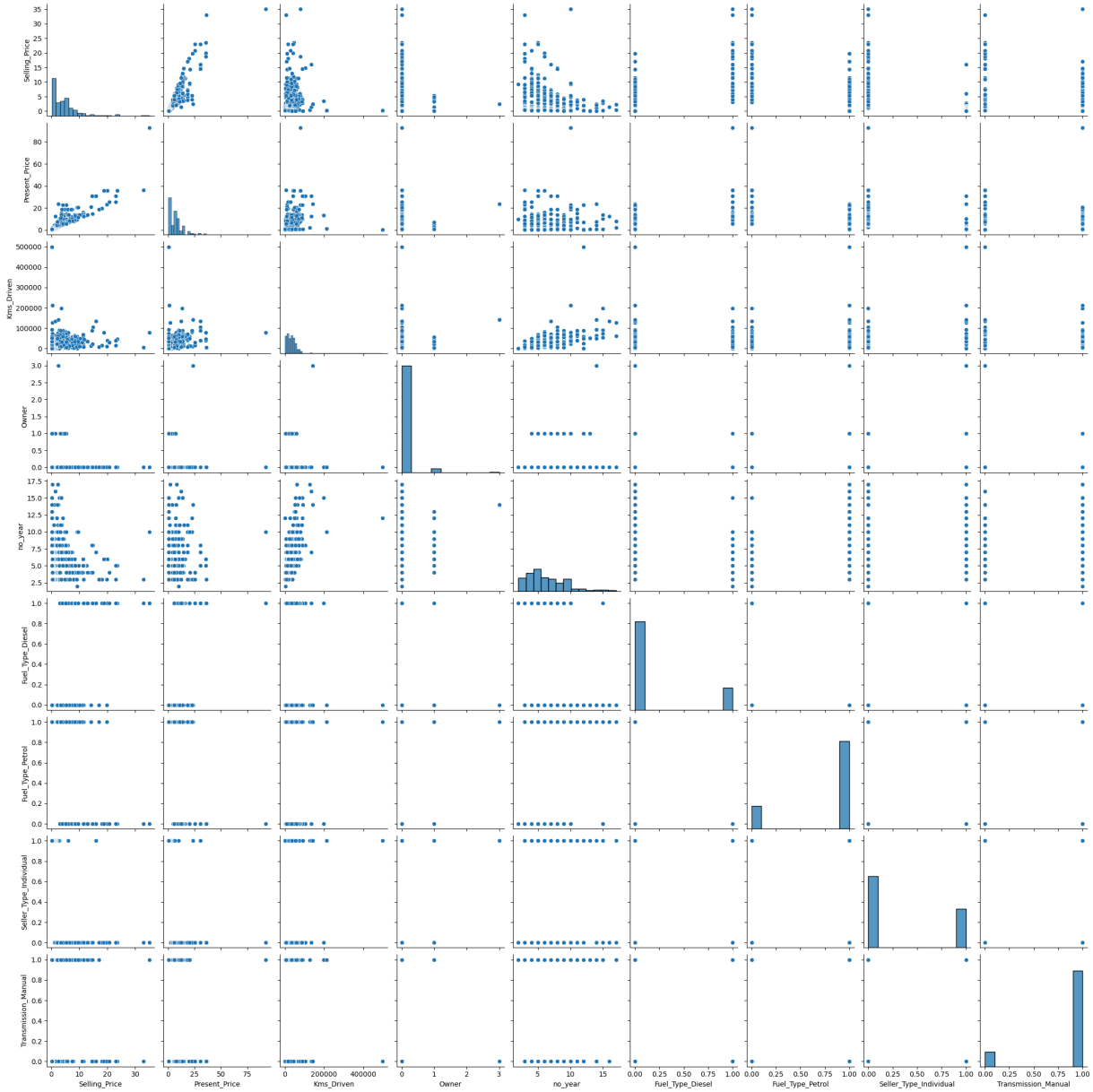
	Selling_Price	Present_Price	Kms_Driven	Owner	no_year	Fu
count	301.000000	301.000000	301.000000	301.000000	301.000000	
mean	4.661296	7.628472	36947.205980	0.043189	6.372093	
std	5.082812	8.644115	38886.883882	0.247915	2.891554	
min	0.100000	0.320000	500.000000	0.000000	2.000000	
25%	0.900000	1.200000	15000.000000	0.000000	4.000000	
50%	3.600000	6.400000	32000.000000	0.000000	6.000000	
75%	6.000000	9.900000	48767.000000	0.000000	8.000000	
max	35.000000	92.600000	500000.000000	3.000000	17.000000	

In [100…

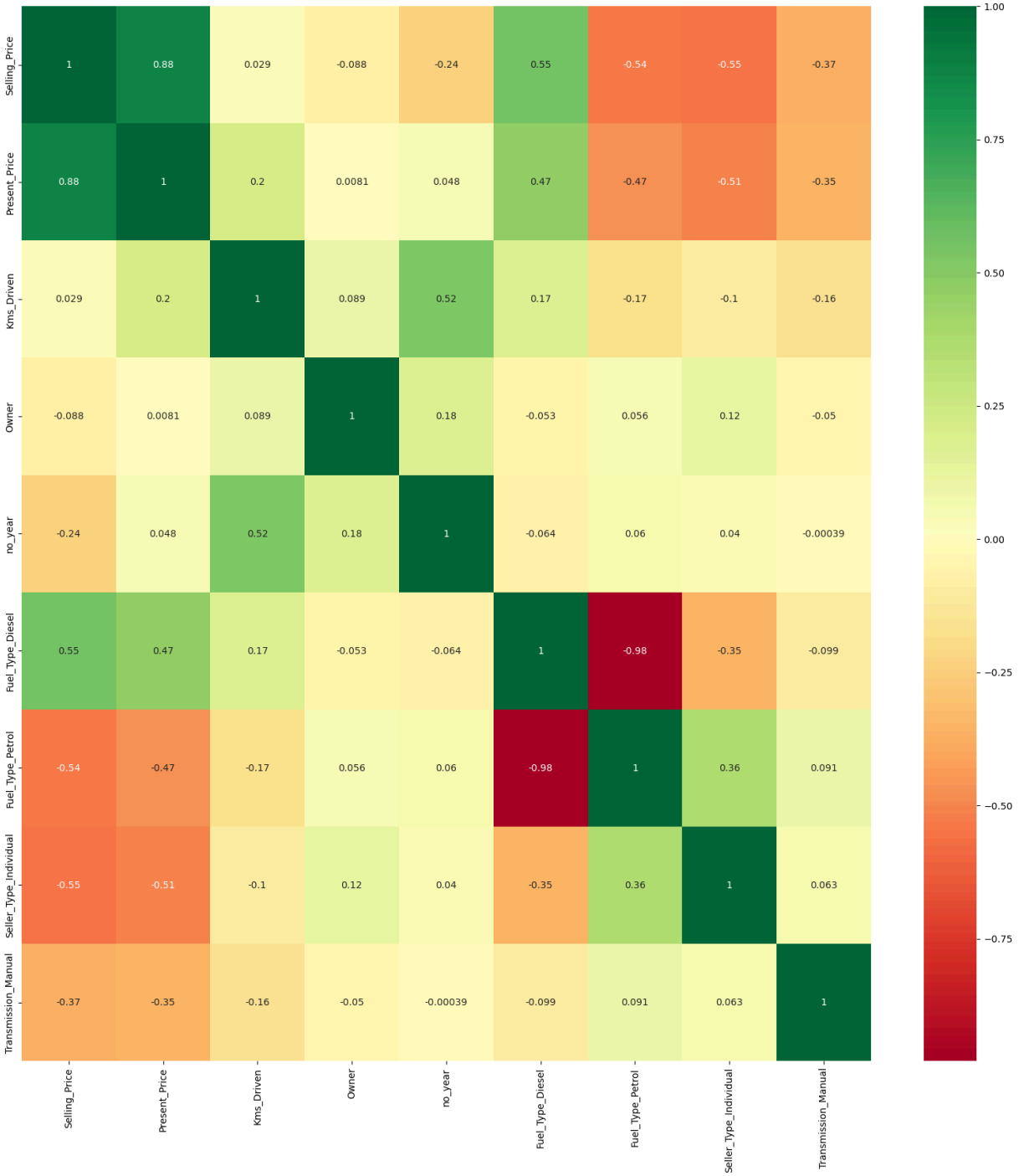
```
sb.pairplot(df)
```

Out[100…

```
<seaborn.axisgrid.PairGrid at 0x1e44bda6160>
```



In [101…]
```
corrmat=df.corr()
top_corr_features=corrmat.index
plt.figure(figsize=(20,20))
g=sb.heatmap(df[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```



In [102…]
```
# dividing the dataset in to the dependent and independent dataset

x=df.iloc[:,1:]
```

In [103…]
```
y=df.iloc[:,0]
```

In [104…]
```
x.head()
```

Out[104…

	Present_Price	Kms_Driven	Owner	no_year	Fuel_Type_Diesel	Fuel_Type_Pe
0	5.59	27000	0	6	0	
1	9.54	43000	0	7	1	
2	9.85	6900	0	3	0	
3	4.15	5200	0	9	0	
4	6.87	42450	0	6	1	

In [105…
```
y.head()
```

Out[105…
```
0    3.35
1    4.75
2    7.25
3    2.85
4    4.60
Name: Selling_Price, dtype: float64
```

In [106…
```
from sklearn.ensemble import ExtraTreesRegressor
model=ExtraTreesRegressor()
model.fit(x,y)
```

Out[106…
```
ExtraTreesRegressor()
```

In [107…
```
model.score(x,y)
```

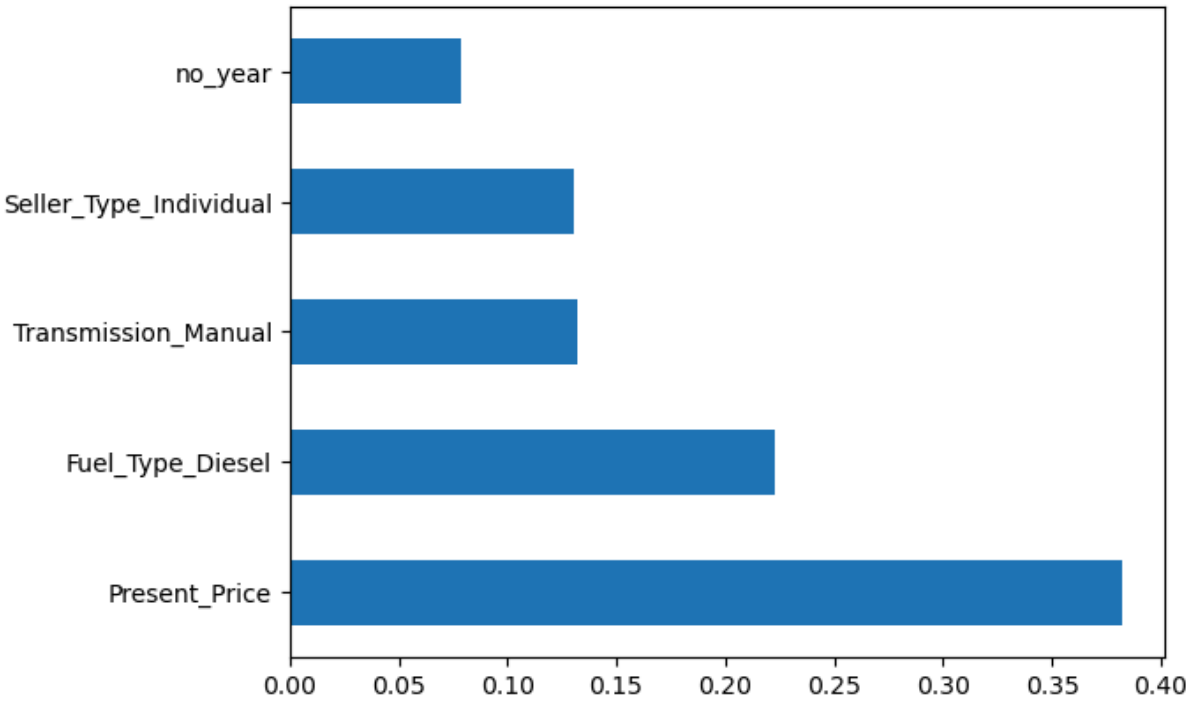
Out[107…
```
1.0
```

In [108…
```
model.feature_importances_
```

Out[108…
```
array([0.38244606, 0.04093561, 0.000415  , 0.07862471, 0.22260659,
       0.01267155, 0.13051966, 0.13178082])
```

In [109…
```
feat_importance=pd.Series(model.feature_importances_,index=x.columns)
feat_importance.nlargest(5).plot(kind='barh')
```

Out[109…
```
<AxesSubplot:>
```



In [110…
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.20)
```

In [111…
```
x_train.shape
```

Out[111… (240, 8)

In [112…
```
x_test.shape
```

Out[112… (61, 8)

In [113…
```
from sklearn.ensemble import RandomForestRegressor
model_2=RandomForestRegressor()
```

In [114…
```
# Grid SearchCV
# for n_estimator
n_estimators=[int(x) for x in np.linspace(start=100, stop=1200,num=12)]

# for max_featrue
max_features=['auto','sqrt']

# for depth
max_depth=[int(x) for x in np.linspace(5,30,num=6)]

# for min_sample_split
min_samples_split=([2,5,10,15,100])

# fro min_sample_leaf
min_samples_leaf=[1,2,5,10]
```

In [115…
```
from sklearn.model_selection import RandomizedSearchCV
```

```
random_grid={
    'n_estimators':n_estimators,
    'max_features':max_features,
    'max_depth':max_depth,
    'min_samples_split':min_samples_split,
    'min_samples_leaf':min_samples_leaf
}
```

In [116…

```
rf=RandomizedSearchCV(estimator=model_2, param_distributions=random_grid, sc
```

In [117…

```
rf.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_sp
lit=5, n_estimators=900; total time=   2.8s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_sp
lit=5, n_estimators=900; total time=   3.2s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_sp
lit=5, n_estimators=900; total time=   4.0s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_sp
lit=5, n_estimators=900; total time=   2.6s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_sp
lit=5, n_estimators=900; total time=   3.2s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_sp
lit=10, n_estimators=1100; total time=   3.5s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_sp
lit=10, n_estimators=1100; total time=   4.8s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_sp
lit=10, n_estimators=1100; total time=   3.1s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_sp
lit=10, n_estimators=1100; total time=   3.9s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_sp
lit=10, n_estimators=1100; total time=   7.0s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_sp
lit=100, n_estimators=300; total time=   1.3s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_sp
lit=100, n_estimators=300; total time=   1.5s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_sp
lit=100, n_estimators=300; total time=   2.2s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_sp
lit=100, n_estimators=300; total time=   1.2s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_sp
lit=100, n_estimators=300; total time=   1.1s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_sp
lit=5, n_estimators=400; total time=   1.4s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_sp
lit=5, n_estimators=400; total time=   1.2s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_sp
lit=5, n_estimators=400; total time=   1.2s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_sp
lit=5, n_estimators=400; total time=   1.2s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_sp
lit=5, n_estimators=400; total time=   1.6s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_s
plit=5, n_estimators=700; total time=   2.8s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_s
plit=5, n_estimators=700; total time=   2.3s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_s
plit=5, n_estimators=700; total time=   1.7s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_s
plit=5, n_estimators=700; total time=   1.2s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_s
plit=5, n_estimators=700; total time=   1.3s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_sp
lit=2, n_estimators=1000; total time=   2.1s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_sp
lit=2, n_estimators=1000; total time=   2.7s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_sp
```

```
lit=2, n_estimators=1000; total time=   2.8s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_sp
lit=2, n_estimators=1000; total time=   2.1s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_sp
lit=2, n_estimators=1000; total time=   2.6s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_sp
lit=15, n_estimators=1100; total time=   3.4s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_sp
lit=15, n_estimators=1100; total time=   2.5s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_sp
lit=15, n_estimators=1100; total time=   3.1s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_sp
lit=15, n_estimators=1100; total time=   3.5s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_sp
lit=15, n_estimators=1100; total time=   2.4s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_sp
lit=15, n_estimators=300; total time=   0.5s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_sp
lit=15, n_estimators=300; total time=   0.5s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_sp
lit=15, n_estimators=300; total time=   0.9s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_sp
lit=15, n_estimators=300; total time=   1.7s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_sp
lit=15, n_estimators=300; total time=   2.0s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_spl
it=10, n_estimators=700; total time=   4.1s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_spl
it=10, n_estimators=700; total time=   3.8s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_spl
it=10, n_estimators=700; total time=   4.1s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_spl
it=10, n_estimators=700; total time=   3.1s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_spl
it=10, n_estimators=700; total time=   2.8s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_sp
lit=15, n_estimators=700; total time=   3.5s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_sp
lit=15, n_estimators=700; total time=   3.1s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_sp
lit=15, n_estimators=700; total time=   2.6s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_sp
lit=15, n_estimators=700; total time=   2.3s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_sp
lit=15, n_estimators=700; total time=   2.0s
```

Out[117…
```
RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(), n_jobs=1,
                   param_distributions={'max_depth': [5, 10, 15, 20, 25, 3
0],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_leaf': [1, 2, 5, 10],
                                        'min_samples_split': [2, 5, 10, 15,
                                                              100],
                                        'n_estimators': [100, 200, 300, 40
0,
                                                         500, 600, 700, 80
0,
                                                         900, 1000, 1100,
                                                         1200]},
                   random_state=42, scoring='neg_mean_squared_error',
                   verbose=2)
```

In [118…
```
rf.best_params_
```

Out[118…
```
{'n_estimators': 700,
 'min_samples_split': 15,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth': 20}
```

In [119…
```
y_pred=rf.predict(x_test)
```

In [120…
```
y_pred
```

Out[120…
```
array([23.27699711,  3.69874845,  0.68444389,  0.2847245 ,  0.62459747,
        0.25029553,  0.41243903,  4.49915519,  3.8583009 ,  1.16288033,
       23.27699711, 23.28469344,  7.43330901,  2.77303057,  7.27768673,
        0.3060368 ,  5.28840354,  7.33407163,  4.93966888,  5.35142082,
        0.68476858,  5.28558101,  0.59793313,  1.17992504,  0.56746235,
       10.50297087,  2.97811358, 17.97085404,  8.75905635,  0.30876405,
        2.8108766 ,  1.1635106 ,  4.54067581,  6.20926818,  2.70491992,
        2.81115306,  0.57032066, 10.0128442 ,  0.36521691,  1.16133974,
        0.65264802,  7.47208922,  7.17726758,  5.67686153,  9.35655894,
        2.76377987,  4.34621969,  0.29295418,  1.97251475,  0.33698417,
        4.43117365,  0.32672353,  0.41374821,  4.73191599,  3.30874455,
        4.99163158, 11.50917179, 23.27699711,  2.99640056,  6.39369547,
       10.69921968])
```

In [121…
```
rf.score(x_test,y_test)
```

Out[121…
```
-8.154630980730952
```

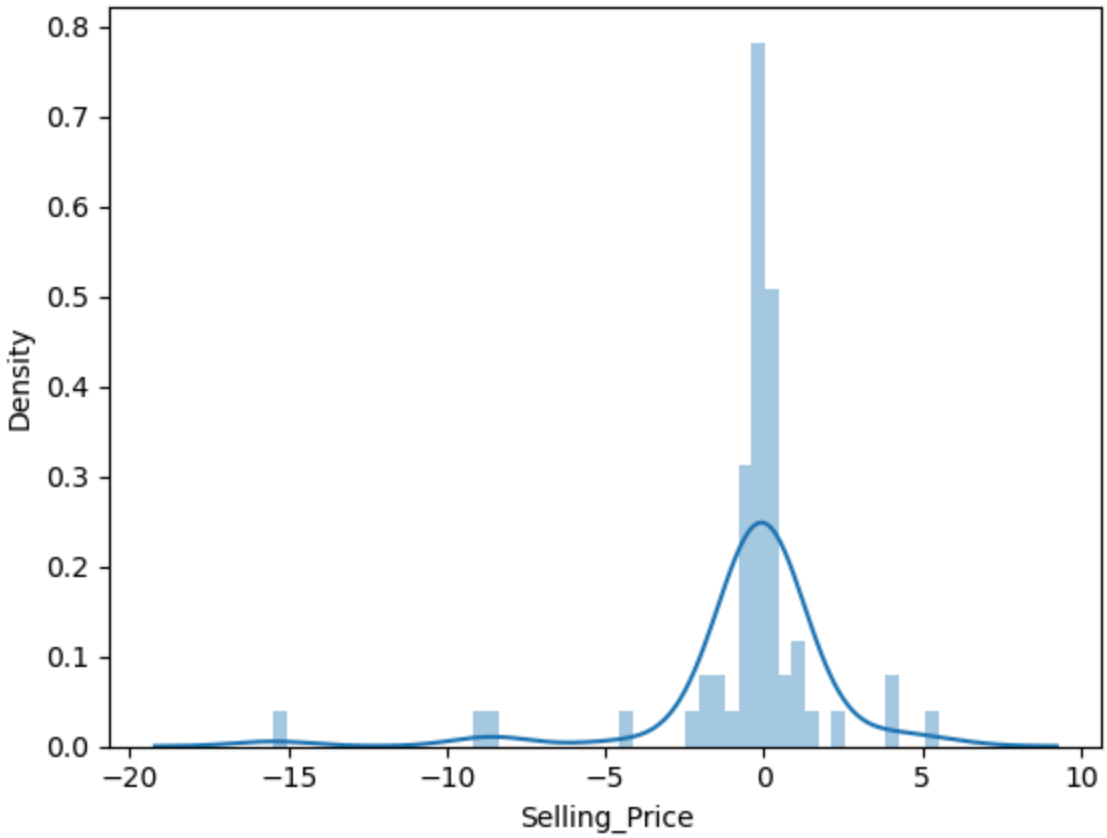
In [122…
```
y_pred
```

Out[122…
```
array([23.27699711,  3.69874845,  0.68444389,  0.2847245 ,  0.62459747,
        0.25029553,  0.41243903,  4.49915519,  3.8583009 ,  1.16288033,
       23.27699711, 23.28469344,  7.43330901,  2.77303057,  7.27768673,
        0.3060368 ,  5.28840354,  7.33407163,  4.93966888,  5.35142082,
        0.68476858,  5.28558101,  0.59793313,  1.17992504,  0.56746235,
       10.50297087,  2.97811358, 17.97085404,  8.75905635,  0.30876405,
        2.8108766 ,  1.1635106 ,  4.54067581,  6.20926818,  2.70491992,
        2.81115306,  0.57032066, 10.0128442 ,  0.36521691,  1.16133974,
        0.65264802,  7.47208922,  7.17726758,  5.67686153,  9.35655894,
        2.76377987,  4.34621969,  0.29295418,  1.97251475,  0.33698417,
        4.43117365,  0.32672353,  0.41374821,  4.73191599,  3.30874455,
        4.99163158, 11.50917179, 23.27699711,  2.99640056,  6.39369547,
       10.69921968])
```

In [123…
```
sb.distplot(y_test-y_pred)
```

```
E:\Anaconda\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

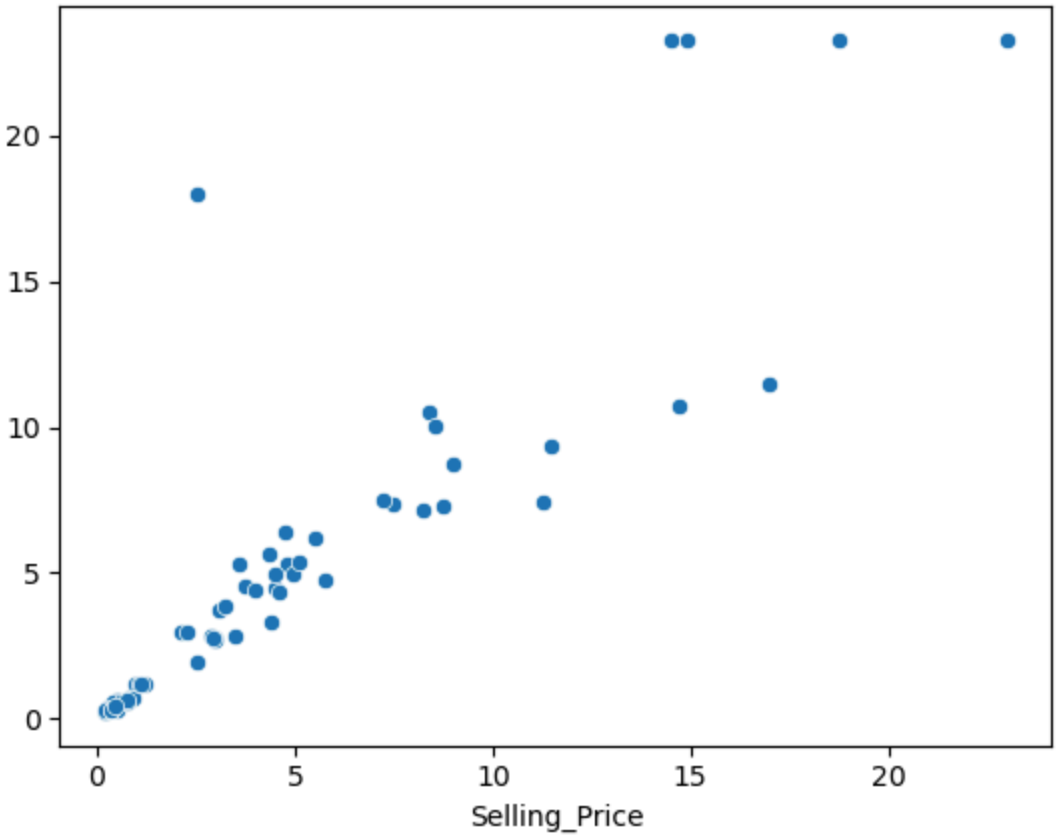
Out[123…
```
<AxesSubplot:xlabel='Selling_Price', ylabel='Density'>
```



In [124…
```
sb.scatterplot(y_test,y_pred)
```

```
E:\Anaconda\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass
the following variables as keyword args: x, y. From version 0.12, the only v
alid positional argument will be `data`, and passing other arguments without
an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[124… <AxesSubplot:xlabel='Selling_Price'>



In [125…
```
# import pickle

# file=open('car_model.pkl','wb')
# pickle.dump(rf,file)
```

In [126…
```
x_train.head()
```

Out[126…

	Present_Price	Kms_Driven	Owner	no_year	Fuel_Type_Diesel	Fuel_Type
288	13.60	34000	0	5	0	
223	9.40	61381	0	5	1	
245	9.40	71000	0	8	1	
239	4.43	23709	0	8	0	
173	0.51	1300	0	3	0	

In [127…
```
#Use pickle to save our model so that we can use it later

import pickle
```

```
pickle.dump(rf, open('car_rf_model.pkl','wb'))
```

In [129…
```
x_train['Owner'].nunique()
```

Out[129… 2

In [130…
```
df
```

Out[130…

	Selling_Price	Present_Price	Kms_Driven	Owner	no_year	Fuel_Type_Die
0	3.35	5.59	27000	0	6	
1	4.75	9.54	43000	0	7	
2	7.25	9.85	6900	0	3	
3	2.85	4.15	5200	0	9	
4	4.60	6.87	42450	0	6	
...	...	...	...	...	...	
296	9.50	11.60	33988	0	4	
297	4.00	5.90	60000	0	5	
298	3.35	11.00	87934	0	11	
299	11.50	12.50	9000	0	3	
300	5.30	5.90	5464	0	4	

301 rows × 9 columns

In [131…

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_8704\2250394493.py in <module>
----> 1 df.Seller_Type

E:\Anaconda\lib\site-packages\pandas\core\generic.py in __getattr__(self, na
me)
   5573         ):
   5574             return self[name]
-> 5575         return object.__getattribute__(self, name)
   5576
   5577     def __setattr__(self, name: str, value) -> None:

AttributeError: 'DataFrame' object has no attribute 'Seller_Type'
```

In [ ]:

In [ ]:

In [ ]: