prerak-project-lr

October 1, 2023

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

[2]:
```
df=pd.read_excel(r"C:\Users\prera\OneDrive\Desktop\Imarticus\ML\datasets\CAR␣
↪DETAILS FROM CAR DEKHO.xlsx")
```

[3]:
```
df
```

[3]:
```
                                name    year  selling_price  km_driven  \
0                       Maruti 800 AC  2007.0        60000.0    70000.0
1            Maruti Wagon R LXI Minor  2007.0       135000.0    50000.0
2                Hyundai Verna 1.6 SX  2012.0       600000.0   100000.0
3              Datsun RediGO T Option  2017.0       250000.0    46000.0
4               Honda Amaze VX i-DTEC  2014.0       450000.0   141000.0
…                                   …     …             …          …
4335  Hyundai i20 Magna 1.4 CRDi (Diesel)  2014.0       409999.0    80000.0
4336           Hyundai i20 Magna 1.4 CRDi  2014.0       409999.0    80000.0
4337                  Maruti 800 AC BSIII  2009.0       110000.0    83000.0
4338     Hyundai Creta 1.6 CRDi SX Option  2016.0       865000.0        NaN
4339                     Renault KWID RXT  2016.0            NaN    40000.0

        fuel seller_type transmission         owner
0     Petrol  Individual       Manual   First Owner
1     Petrol  Individual       Manual   First Owner
2     Diesel  Individual       Manual   First Owner
3     Petrol  Individual       Manual   First Owner
4     Diesel  Individual       Manual  Second Owner
…        …          …           …            …
4335  Diesel  Individual       Manual  Second Owner
4336  Diesel  Individual       Manual  Second Owner
4337  Petrol  Individual       Manual  Second Owner
4338     NaN  Individual       Manual   First Owner
4339  Petrol  Individual       Manual   First Owner
```

```
[4340 rows x 8 columns]
```

[4]:
```python
df.shape
```

[4]:
```
(4340, 8)
```

[5]:
```python
df.head()
```

[5]:
```
                      name    year  selling_price  km_driven    fuel  \
0             Maruti 800 AC  2007.0        60000.0    70000.0  Petrol
1  Maruti Wagon R LXI Minor  2007.0       135000.0    50000.0  Petrol
2      Hyundai Verna 1.6 SX  2012.0       600000.0   100000.0  Diesel
3    Datsun RediGO T Option  2017.0       250000.0    46000.0  Petrol
4     Honda Amaze VX i-DTEC  2014.0       450000.0   141000.0  Diesel

  seller_type transmission         owner
0  Individual       Manual   First Owner
1  Individual       Manual   First Owner
2  Individual       Manual   First Owner
3  Individual       Manual   First Owner
4  Individual       Manual  Second Owner
```

[6]:
```python
df.tail()
```

[6]:
```
                                   name    year  selling_price  km_driven  \
4335  Hyundai i20 Magna 1.4 CRDi (Diesel)  2014.0       409999.0    80000.0
4336           Hyundai i20 Magna 1.4 CRDi  2014.0       409999.0    80000.0
4337                  Maruti 800 AC BSIII  2009.0       110000.0    83000.0
4338     Hyundai Creta 1.6 CRDi SX Option  2016.0       865000.0        NaN
4339                     Renault KWID RXT  2016.0            NaN    40000.0

        fuel seller_type transmission         owner
4335  Diesel  Individual       Manual  Second Owner
4336  Diesel  Individual       Manual  Second Owner
4337  Petrol  Individual       Manual  Second Owner
4338     NaN  Individual       Manual   First Owner
4339  Petrol  Individual       Manual   First Owner
```

[7]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4340 entries, 0 to 4339
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   name           4340 non-null   object
 1   year           4263 non-null   float64
```

```
 2   selling_price  3820 non-null   float64
 3   km_driven      3908 non-null   float64
 4   fuel           4287 non-null   object
 5   seller_type    4300 non-null   object
 6   transmission   4306 non-null   object
 7   owner          4309 non-null   object
dtypes: float64(3), object(5)
memory usage: 271.4+ KB
```

[8]:
```
df.describe(include='all')
```

[8]:
```
                            name         year  selling_price      km_driven  \
count                       4340  4263.000000   3.820000e+03    3908.000000
unique                      1491          NaN            NaN            NaN
top     Maruti Swift Dzire VDI          NaN            NaN            NaN
freq                          69          NaN            NaN            NaN
mean                         NaN  2013.084917   5.007083e+05   66261.846725
std                          NaN     4.220941   5.682613e+05   47093.358054
min                          NaN  1992.000000   2.200000e+04       1.000000
25%                          NaN  2011.000000   2.007492e+05   35000.000000
50%                          NaN  2014.000000   3.500000e+05   60000.000000
75%                          NaN  2016.000000   6.000000e+05   90000.000000
max                          NaN  2020.000000   8.150000e+06  806599.000000

          fuel seller_type transmission        owner
count     4287        4300         4306         4309
unique       5           3            2            5
top     Diesel  Individual       Manual  First Owner
freq      2129        3214         3863         2812
mean       NaN         NaN          NaN          NaN
std        NaN         NaN          NaN          NaN
min        NaN         NaN          NaN          NaN
25%        NaN         NaN          NaN          NaN
50%        NaN         NaN          NaN          NaN
75%        NaN         NaN          NaN          NaN
max        NaN         NaN          NaN          NaN
```

[9]:
```
df.nunique()
```

[9]:
```
name             1491
year               27
selling_price     420
km_driven         716
fuel                5
seller_type         3
transmission        2
owner               5
```

```
dtype: int64
```

[10]:
```python
df_c=df.copy(deep=True)
```

[11]:
```python
df.isnull().sum()
```

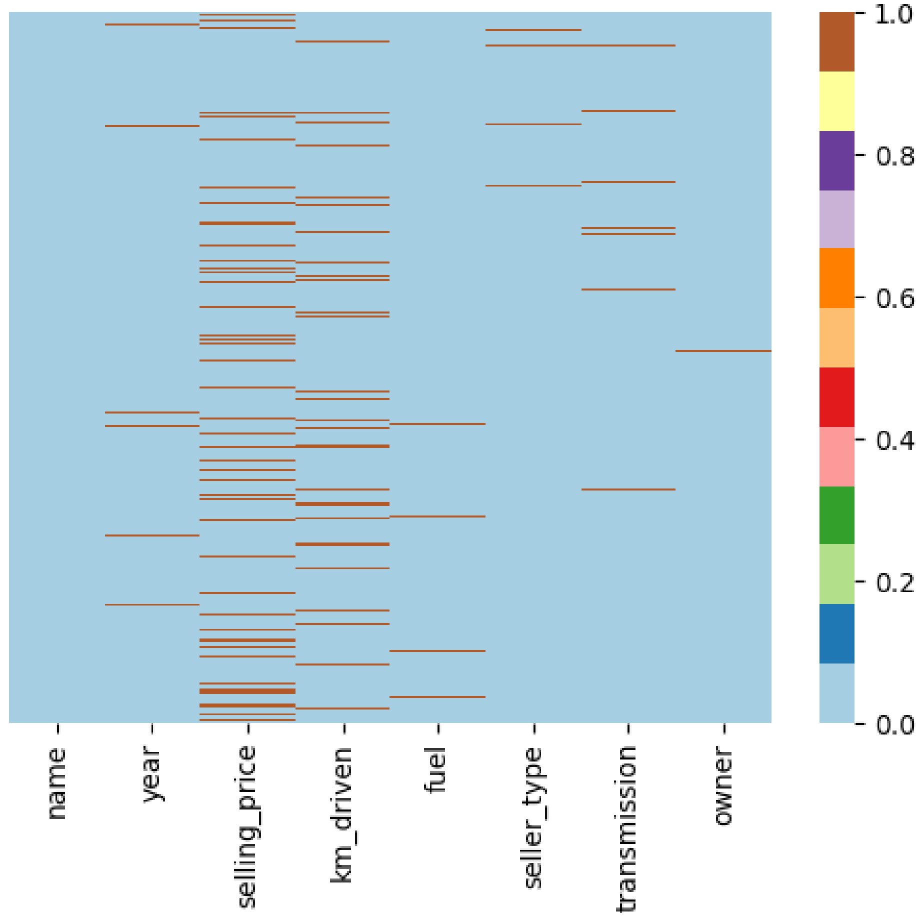
[11]:
```
name                0
year               77
selling_price     520
km_driven         432
fuel               53
seller_type        40
transmission       34
owner              31
dtype: int64
```

[12]:
```python
(df.isnull().sum()/(len(df)))*100 #percentage of missing values in each column
```

[12]:
```
name              0.000000
year              1.774194
selling_price    11.981567
km_driven         9.953917
fuel              1.221198
seller_type       0.921659
transmission      0.783410
owner             0.714286
dtype: float64
```

[13]:
```python
sns.heatmap(df.isnull(),yticklabels=False,cmap="Paired") #to check the missing␣
↪values on a heatmap
```

[13]: <Axes: >



[14]:
```
df['fuel'].value_counts()
```

[14]:
```
Diesel      2129
Petrol      2095
CNG           39
LPG           23
Electric       1
Name: fuel, dtype: int64
```

[15]:
```
df['seller_type'].value_counts()
```

[15]:
```
Individual          3214
Dealer               985
Trustmark Dealer     101
Name: seller_type, dtype: int64
```

[16]:
```
df['transmission'].value_counts()
```

[16]:
```
Manual       3863
Automatic     443
Name: transmission, dtype: int64
```

[17]:
```
df.dropna(subset=["seller_type"],inplace=True)
df.dropna(subset=["transmission"],inplace=True)
df.dropna(subset=["owner"],inplace=True)
```

[18]:
```
df["fuel"].fillna(df["fuel"].mode()[0],inplace=True)
df["year"].ffill(axis=0,inplace=True)
```

[19]:
```
df["selling_price"].fillna(df["selling_price"].median(),inplace=True)
df["km_driven"].fillna(df["km_driven"].median(),inplace=True)
```

[20]:
```
df.isnull().sum()
```

[20]:
```
name             0
year             0
selling_price    0
km_driven        0
fuel             0
seller_type      0
transmission     0
owner            0
dtype: int64
```

[21]:
```
df["Current year"]=2023
```

[22]:
```
df["Age"]=df["Current year"]-df["year"]
```

[23]:
```
df.drop(["Current year"],axis=1,inplace=True)
```

[24]:
```
df
```

[24]:
```
                                  name    year  selling_price  km_driven  \
0                        Maruti 800 AC  2007.0        60000.0    70000.0
1             Maruti Wagon R LXI Minor  2007.0       135000.0    50000.0
2                 Hyundai Verna 1.6 SX  2012.0       600000.0   100000.0
3               Datsun RediGO T Option  2017.0       250000.0    46000.0
4                Honda Amaze VX i-DTEC  2014.0       450000.0   141000.0
…                                    …       …              …          …
4335  Hyundai i20 Magna 1.4 CRDi (Diesel)  2014.0       409999.0    80000.0
4336           Hyundai i20 Magna 1.4 CRDi  2014.0       409999.0    80000.0
4337                  Maruti 800 AC BSIII  2009.0       110000.0    83000.0
4338     Hyundai Creta 1.6 CRDi SX Option  2016.0       865000.0    60000.0
4339                     Renault KWID RXT  2016.0       350000.0    40000.0
```

```
        fuel seller_type transmission         owner   Age
0      Petrol  Individual       Manual   First Owner  16.0
1      Petrol  Individual       Manual   First Owner  16.0
2      Diesel  Individual       Manual   First Owner  11.0
3      Petrol  Individual       Manual   First Owner   6.0
4      Diesel  Individual       Manual  Second Owner   9.0
…       …          …            …            …    …
4335   Diesel  Individual       Manual  Second Owner   9.0
4336   Diesel  Individual       Manual  Second Owner   9.0
4337   Petrol  Individual       Manual  Second Owner  14.0
4338   Diesel  Individual       Manual   First Owner   7.0
4339   Petrol  Individual       Manual   First Owner   7.0

[4239 rows x 9 columns]
```

[25]:
```python
df[df['Age']==df['Age'].min()].reset_index() #Newest cars in all the dataset
```

[25]:
```
    index                                      name    year  \
0     158                        Maruti Wagon R LXI  2020.0
1     289              Mahindra XUV500 W11 Option AWD  2020.0
2     694           Hyundai Grand i10 Nios Magna CRDi  2020.0
3     963                        Audi A5 Sportback  2020.0
4    1002              Hyundai Creta 1.4 EX Diesel  2020.0
5    1195                        Maruti Baleno Zeta  2020.0
6    1291                       Maruti Alto 800 VXI  2020.0
7    1324                         Maruti Swift VXI  2020.0
8    1409          Volkswagen Polo 1.0 TSI Highline Plus  2020.0
9    1428              Hyundai Grand i10 Nios Sportz  2020.0
10   1432           Hyundai Grand i10 Nios AMT Magna  2020.0
11   1516              Mahindra XUV500 W11 Option AWD  2020.0
12   1575                         Renault KWID RXL  2020.0
13   1595                        Maruti Alto K10 LX  2020.0
14   1689           Hyundai Elite i20 Magna Plus BSIV  2020.0
15   1714              Ford Freestyle Titanium Diesel  2020.0
16   1715                        Ford Figo Titanium  2020.0
17   1716           Ford Ecosport 1.5 Diesel Titanium  2020.0
18   1774              Ford Aspire Titanium BSIV  2020.0
19   1775  Ford EcoSport 1.5 Ti VCT MT Titanium BE BSIV  2020.0
20   1776                        Ford Figo Titanium  2020.0
21   1777              Ford Ecosport 1.5 Petrol Trend  2020.0
22   1778     Ford EcoSport 1.5 TDCi Titanium Plus BSIV  2020.0
23   1779                   Ford Freestyle Titanium  2020.0
24   1780        Ford Ecosport Thunder Edition Diesel  2020.0
25   1781              Ford Freestyle Titanium Plus  2020.0
26   1963               Hyundai Venue SX Opt Diesel  2020.0
27   2016                            Tata Altroz XZ  2020.0
28   2129                     Honda BR-V i-VTEC VX MT  2020.0
```

```
29   2137                    Maruti Ertiga 1.5 VDI  2020.0
30   2154              Ford Ecosport Sports Petrol  2020.0
31   2211              Hyundai Creta 1.4 EX Diesel  2020.0
32   2360       Renault KWID Climber 1.0 MT Opt BSIV  2020.0
33   2476                           Tata Altroz XE  2020.0
34   2481               Hyundai Santro Sportz BSIV  2020.0
35   2558                    Maruti Swift ZXI Plus  2020.0
36   2699                Mahindra Scorpio S5 BSIV  2020.0
37   3024                      Maruti Alto 800 LXI  2020.0
38   3050                      Maruti Alto 800 LXI  2020.0
39   3112           Maruti Eeco CNG 5 Seater AC BSIV  2020.0
40   3268                         Maruti Swift VXI  2020.0
41   3422                      Maruti Alto 800 VXI  2020.0
42   3431            Hyundai Venue SX Opt Turbo BSIV  2020.0
43   3486     Hyundai Grand i10 1.2 Kappa Magna BSIV  2020.0
44   3933          Ford Figo Aspire 1.5 TDCi Titanium  2020.0
45   4105                          Tata Harrier XE  2020.0
46   4278                 Honda Amaze S Petrol BSIV  2020.0

    selling_price  km_driven    fuel seller_type transmission  \
0        240000.0    60000.0  Petrol  Individual       Manual
1       1400000.0    25000.0  Diesel      Dealer       Manual
2        700000.0     1400.0  Diesel  Individual       Manual
3       4700000.0     1500.0  Diesel  Individual    Automatic
4       1050000.0    10000.0  Diesel  Individual       Manual
5        700000.0     1100.0  Petrol  Individual       Manual
6        350000.0     1000.0  Petrol  Individual       Manual
7        350000.0     1500.0  Petrol  Individual       Manual
8        802000.0     5000.0  Petrol  Individual       Manual
9        600000.0     5000.0  Petrol  Individual       Manual
10       640000.0     4000.0  Petrol  Individual    Automatic
11      1400000.0    25000.0  Diesel      Dealer       Manual
12       300000.0    20000.0  Petrol  Individual       Manual
13       250000.0     1100.0  Petrol  Individual       Manual
14       545000.0    60000.0  Petrol  Individual       Manual
15       350000.0      101.0  Diesel      Dealer       Manual
16       635000.0      101.0  Petrol      Dealer       Manual
17      1000000.0      101.0  Diesel      Dealer       Manual
18       828999.0     1010.0  Petrol      Dealer       Manual
19      1119000.0    60000.0  Petrol      Dealer       Manual
20       746000.0     1111.0  Petrol      Dealer       Manual
21      1030000.0     1010.0  Petrol      Dealer       Manual
22      1334000.0     1010.0  Diesel      Dealer       Manual
23       811999.0    60000.0  Petrol      Dealer       Manual
24      1331000.0     1010.0  Diesel      Dealer       Manual
25       852000.0     1010.0  Petrol      Dealer       Manual
26      1000000.0     5000.0  Diesel  Individual       Manual
```

27	830000.0	10000.0	Petrol	Individual	Manual
28	350000.0	1100.0	Petrol	Dealer	Manual
29	550000.0	60000.0	Diesel	Individual	Manual
30	350000.0	1000.0	Petrol	Individual	Manual
31	1050000.0	10000.0	Diesel	Individual	Manual
32	541000.0	1000.0	Petrol	Dealer	Manual
33	500000.0	5000.0	Petrol	Individual	Manual
34	350000.0	5000.0	Petrol	Individual	Manual
35	550000.0	5000.0	Petrol	Individual	Manual
36	350000.0	11000.0	Diesel	Individual	Manual
37	350000.0	5000.0	Petrol	Individual	Manual
38	310000.0	1700.0	Petrol	Individual	Manual
39	350000.0	7000.0	CNG	Individual	Manual
40	619000.0	1500.0	Petrol	Individual	Manual
41	350000.0	40000.0	Petrol	Individual	Manual
42	1050000.0	1100.0	Petrol	Individual	Manual
43	545000.0	5000.0	Petrol	Individual	Manual
44	530000.0	45000.0	Diesel	Dealer	Manual
45	426000.0	60000.0	Diesel	Individual	Manual
46	614000.0	1000.0	Petrol	Individual	Manual

	owner	Age
0	First Owner	3.0
1	First Owner	3.0
2	First Owner	3.0
3	First Owner	3.0
4	First Owner	3.0
5	First Owner	3.0
6	First Owner	3.0
7	First Owner	3.0
8	First Owner	3.0
9	First Owner	3.0
10	First Owner	3.0
11	First Owner	3.0
12	First Owner	3.0
13	Fourth & Above Owner	3.0
14	First Owner	3.0
15	Test Drive Car	3.0
16	Test Drive Car	3.0
17	Test Drive Car	3.0
18	Test Drive Car	3.0
19	Test Drive Car	3.0
20	Test Drive Car	3.0
21	Test Drive Car	3.0
22	Test Drive Car	3.0
23	Test Drive Car	3.0
24	Test Drive Car	3.0

```
25         Test Drive Car  3.0
26            First Owner  3.0
27            First Owner  3.0
28            First Owner  3.0
29            First Owner  3.0
30            First Owner  3.0
31            First Owner  3.0
32         Test Drive Car  3.0
33            First Owner  3.0
34            First Owner  3.0
35            First Owner  3.0
36            First Owner  3.0
37            First Owner  3.0
38            First Owner  3.0
39            First Owner  3.0
40            First Owner  3.0
41            First Owner  3.0
42            First Owner  3.0
43            First Owner  3.0
44            First Owner  3.0
45            First Owner  3.0
46            First Owner  3.0
```

[26]:
```
df[df['Age']==df['Age'].max()].reset_index() #Oldest car from all the dataset
```

[26]:
```
   index                name    year  selling_price  km_driven    fuel  \
0   3334  Maruti 800 AC BSII  1992.0        50000.0   100000.0  Petrol

  seller_type transmission                 owner   Age
0  Individual       Manual  Fourth & Above Owner  31.0
```

[27]:
```
cat_data=df.select_dtypes(include=object)
num_data=df.select_dtypes(exclude=object)
```

[28]:
```
cat_data
```

[28]:
```
                                     name    fuel seller_type transmission  \
0                           Maruti 800 AC  Petrol  Individual       Manual
1                Maruti Wagon R LXI Minor  Petrol  Individual       Manual
2                    Hyundai Verna 1.6 SX  Diesel  Individual       Manual
3                  Datsun RediGO T Option  Petrol  Individual       Manual
4                   Honda Amaze VX i-DTEC  Diesel  Individual       Manual
…                                       …       …           …            …
4335  Hyundai i20 Magna 1.4 CRDi (Diesel)  Diesel  Individual       Manual
4336           Hyundai i20 Magna 1.4 CRDi  Diesel  Individual       Manual
4337                  Maruti 800 AC BSIII  Petrol  Individual       Manual
4338     Hyundai Creta 1.6 CRDi SX Option  Diesel  Individual       Manual
```

```
4339                     Renault KWID RXT  Petrol  Individual       Manual

             owner
0      First Owner
1      First Owner
2      First Owner
3      First Owner
4     Second Owner
…            …
4335  Second Owner
4336  Second Owner
4337  Second Owner
4338   First Owner
4339   First Owner

[4239 rows x 5 columns]
```

[29]:
```
num_data
```

[29]:
```
        year  selling_price  km_driven   Age
0     2007.0        60000.0    70000.0  16.0
1     2007.0       135000.0    50000.0  16.0
2     2012.0       600000.0   100000.0  11.0
3     2017.0       250000.0    46000.0   6.0
4     2014.0       450000.0   141000.0   9.0
…       …             …          …    …
4335  2014.0       409999.0    80000.0   9.0
4336  2014.0       409999.0    80000.0   9.0
4337  2009.0       110000.0    83000.0  14.0
4338  2016.0       865000.0    60000.0   7.0
4339  2016.0       350000.0    40000.0   7.0

[4239 rows x 4 columns]
```

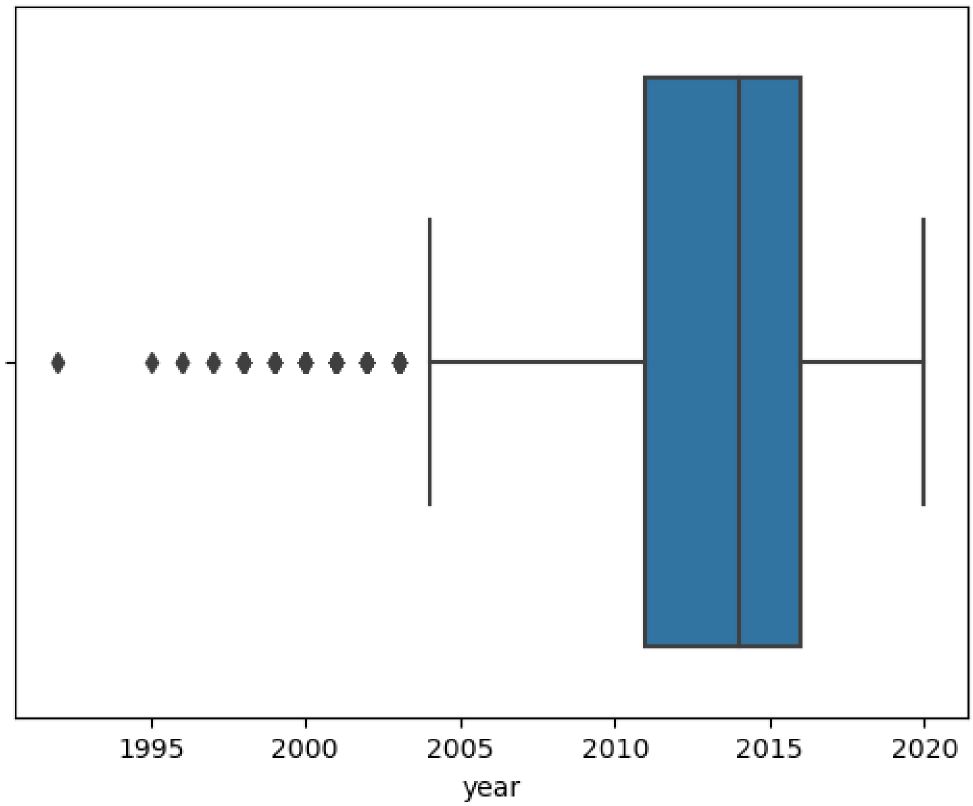
[30]:
```
cor = df.corr()
cor["selling_price"].sort_values(ascending=False)
```

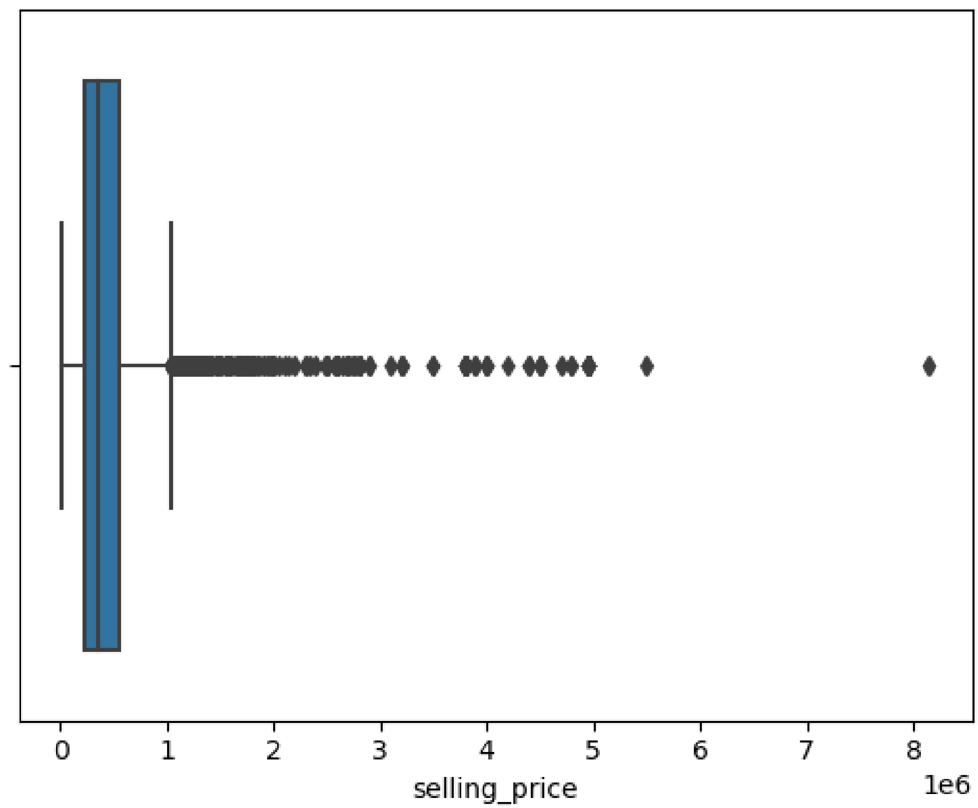
```
C:\Users\prera\AppData\Local\Temp\ipykernel_18460\1617938748.py:1:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only valid
columns or specify the value of numeric_only to silence this warning.
  cor = df.corr()
```

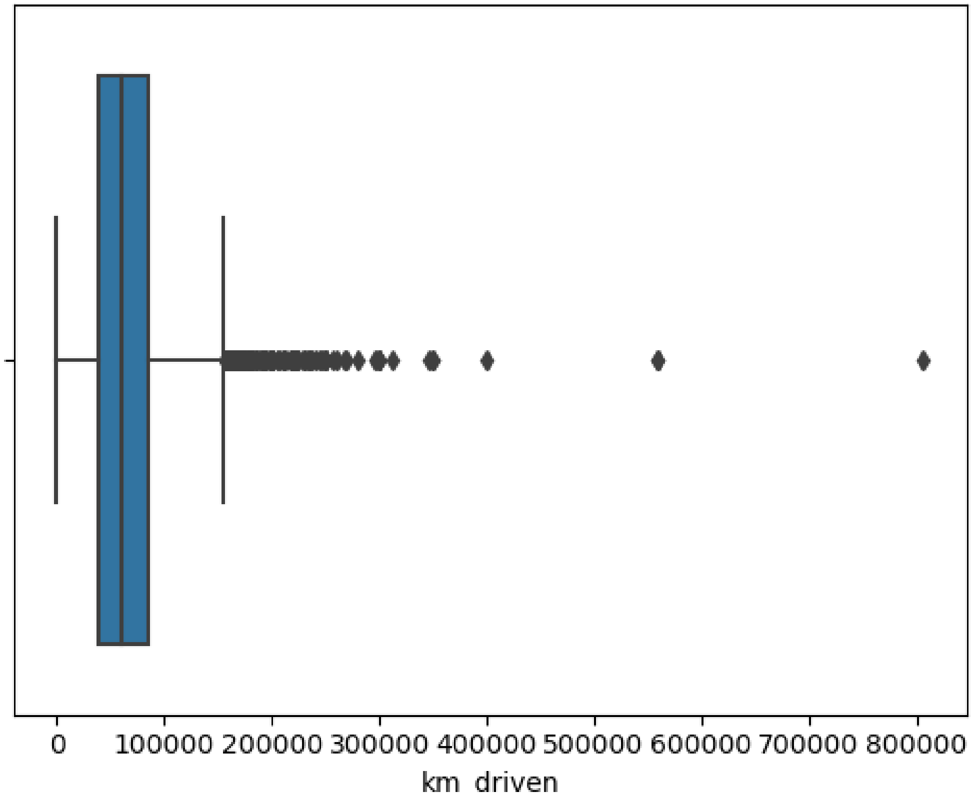
[30]:
```
selling_price    1.000000
year             0.378202
km_driven       -0.166386
Age             -0.378202
```

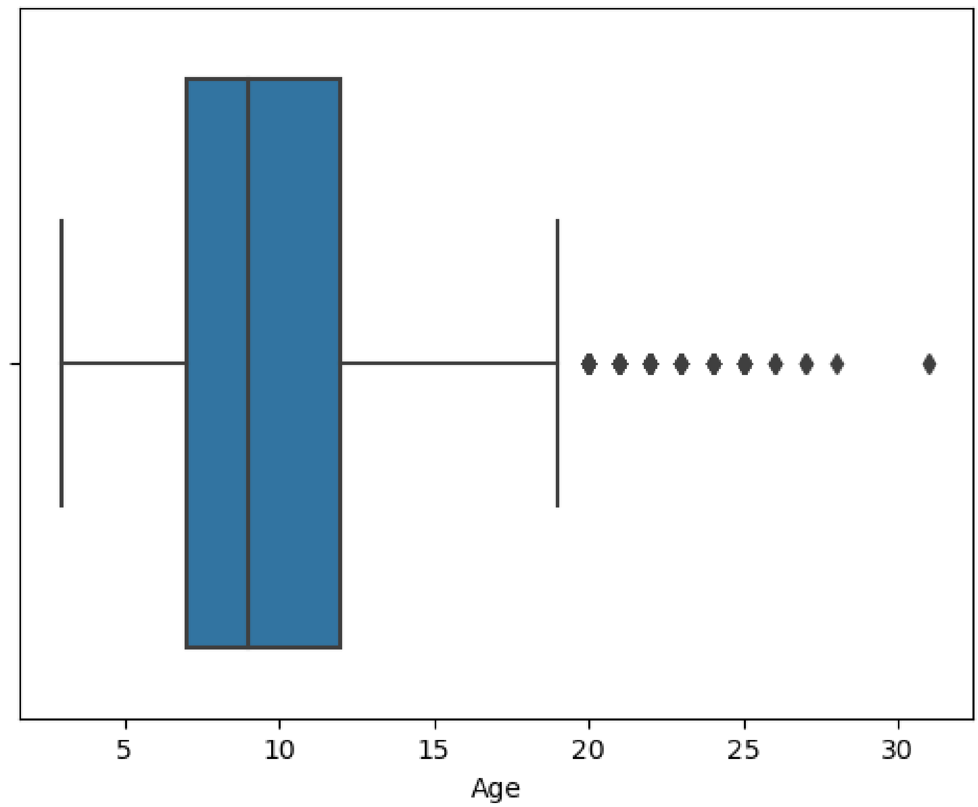
```
Name: selling_price, dtype: float64
```

[31]:
```
for i in num_data.columns:
    sns.boxplot(x=df[i])
    plt.show()
```



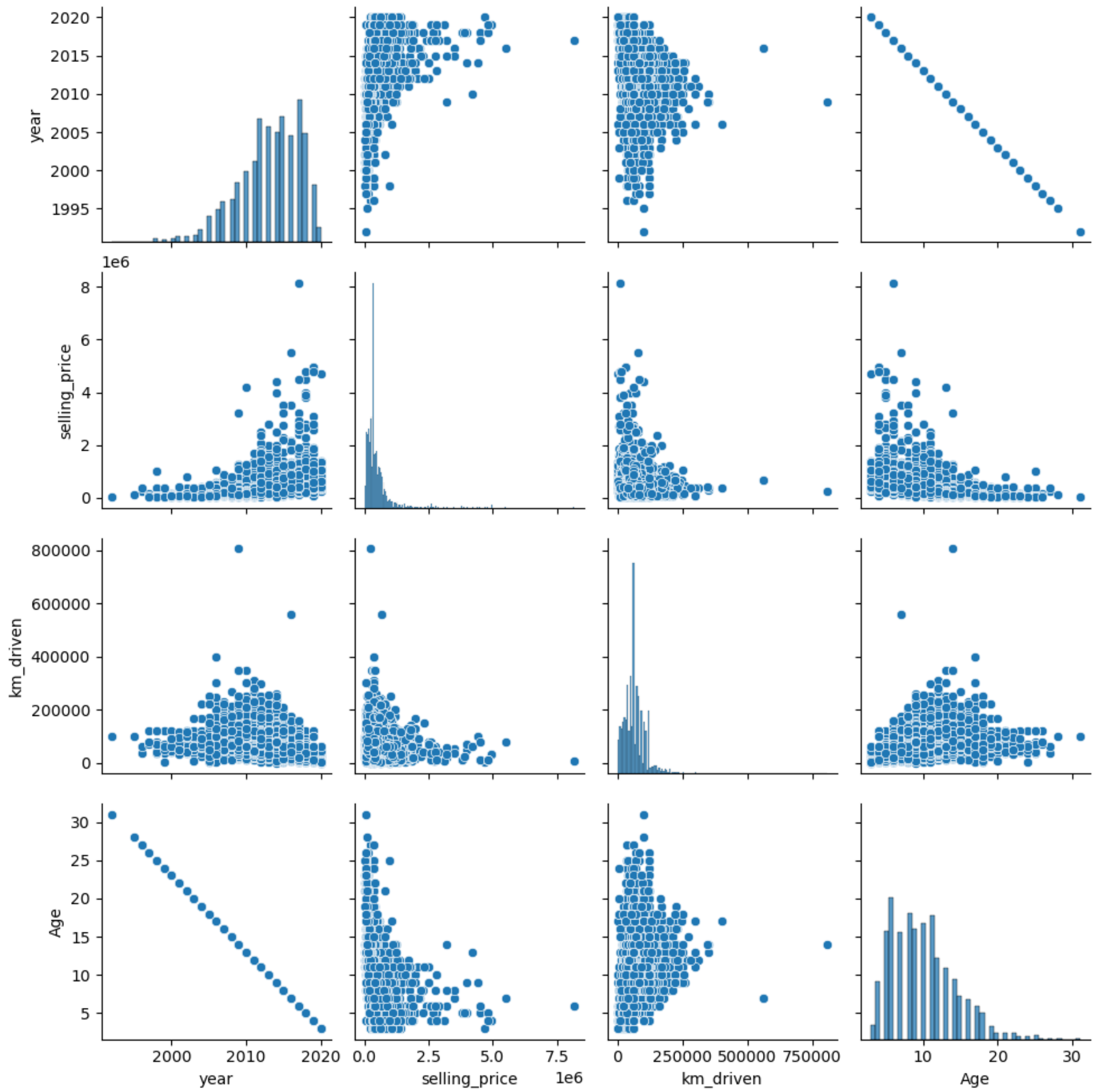

0
1
2
3
4
5
6
7
8
1e6
selling_price


0
100000
200000
300000
400000
500000
600000
700000
800000
km_driven



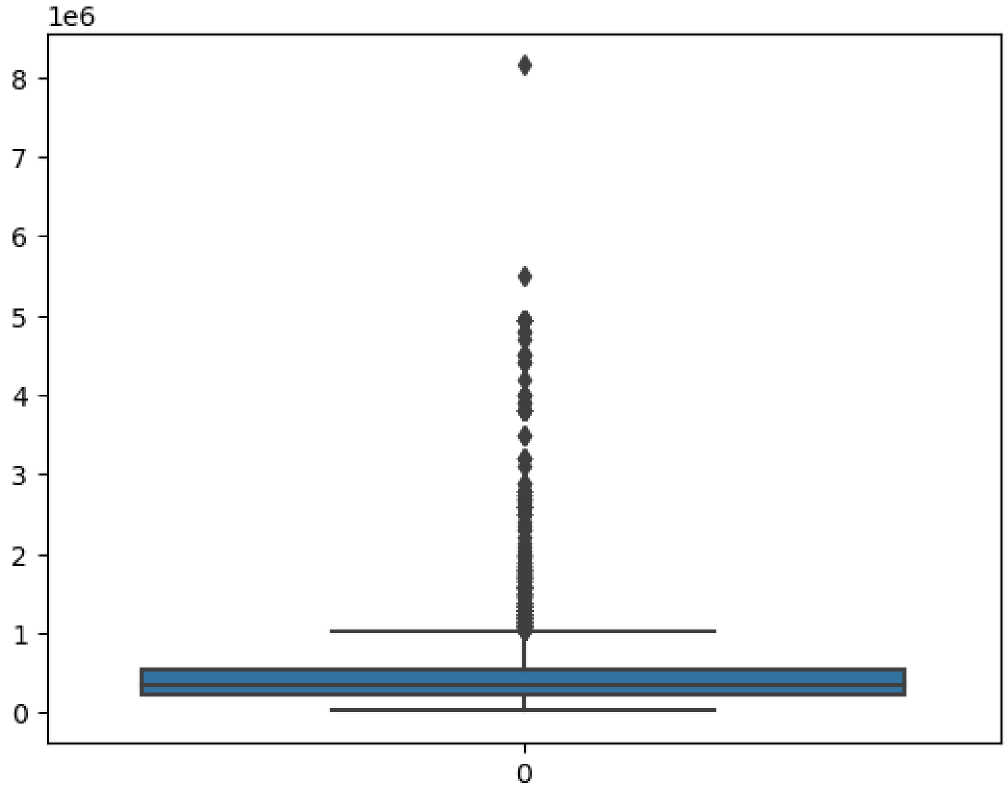
[32]:
```
sns.pairplot(df)
```

[32]:
```
<seaborn.axisgrid.PairGrid at 0x2189b5ddb90>
```



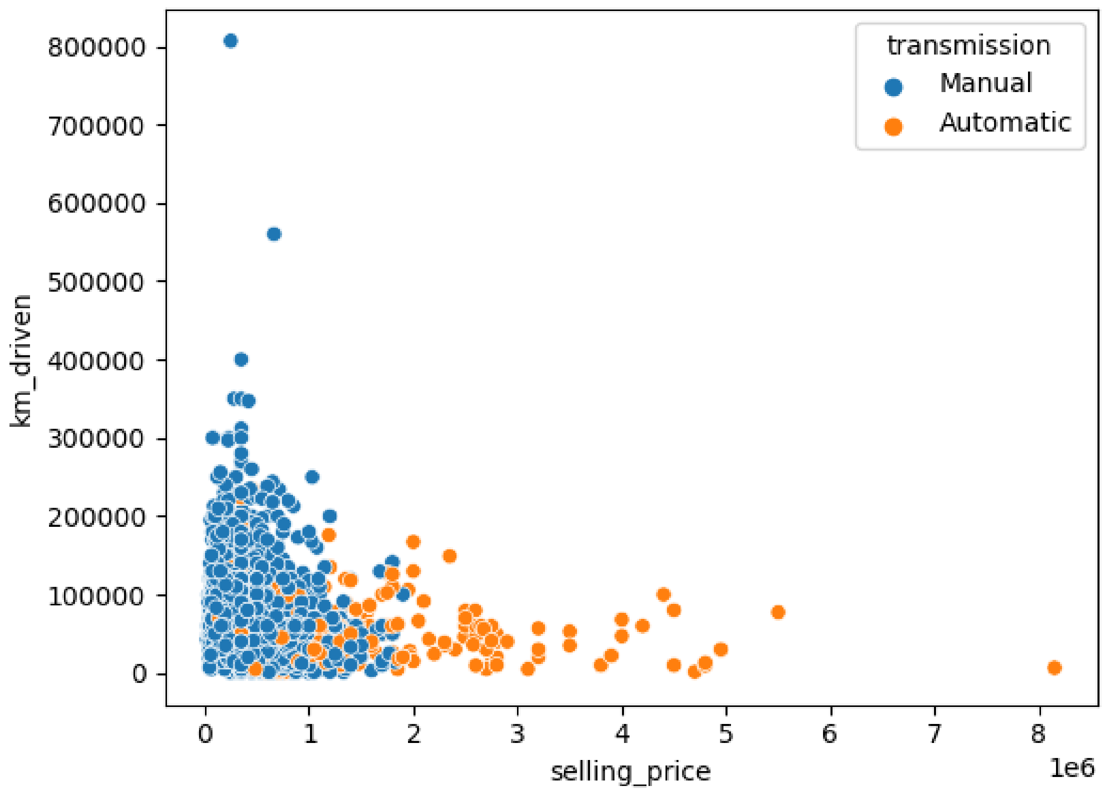
[33]:
```
sns.boxplot(df['selling_price'])
```

[33]: <Axes: >



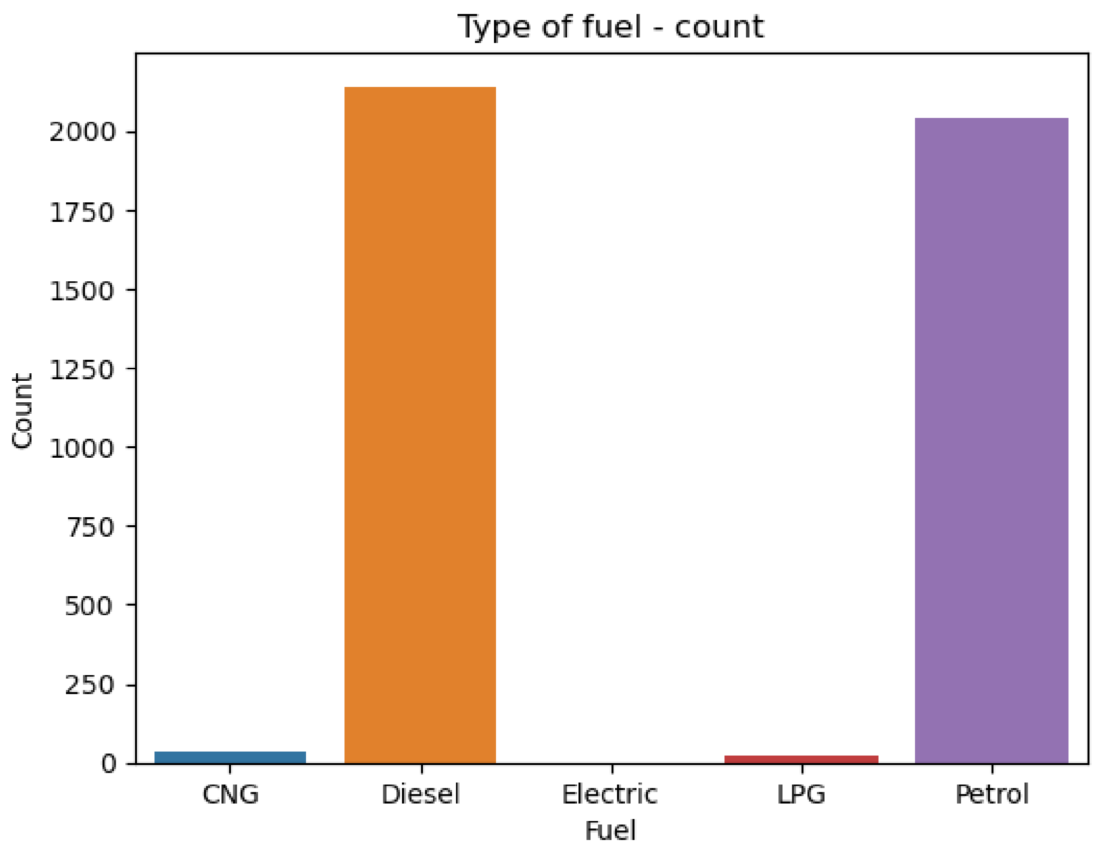
[34]:
```
sns.scatterplot(x='selling_price',y='km_driven',hue="transmission",data=df)
```

[34]: <Axes: xlabel='selling_price', ylabel='km_driven'>



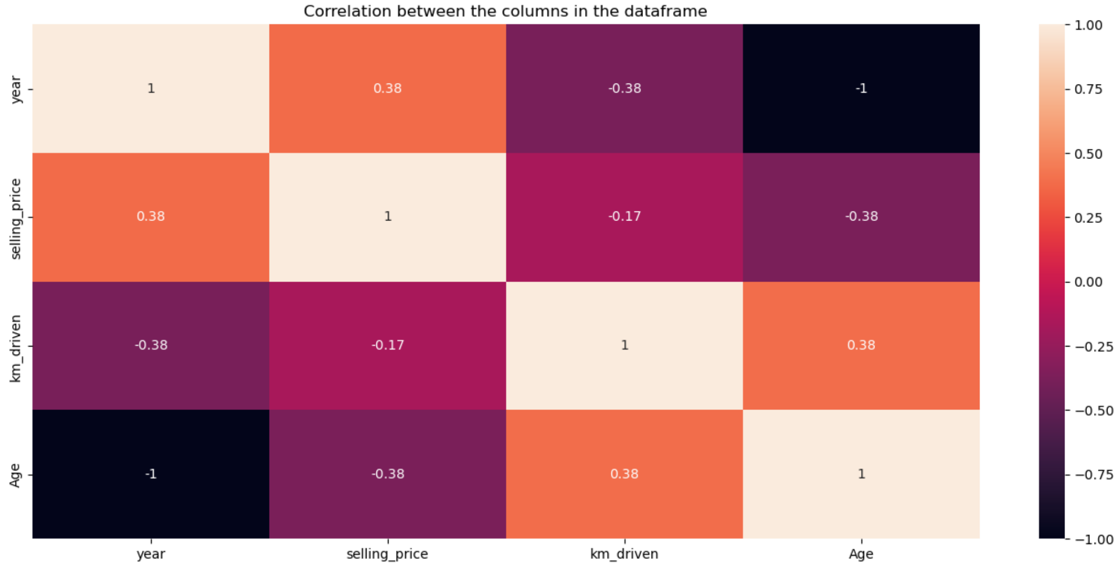
[35]:
```
a=df.groupby("fuel")["fuel"].count()
```

[36]:
```
sns.barplot(x=a.index,y=a.values)
plt.title("Type of fuel - count")
plt.xlabel("Fuel")
plt.ylabel("Count")
plt.show()
```



[37]:
```
plt.figure(figsize = (16,7))
sns.heatmap(df.corr(), annot = True)
plt.title('Correlation between the columns in the dataframe')
plt.show()
```

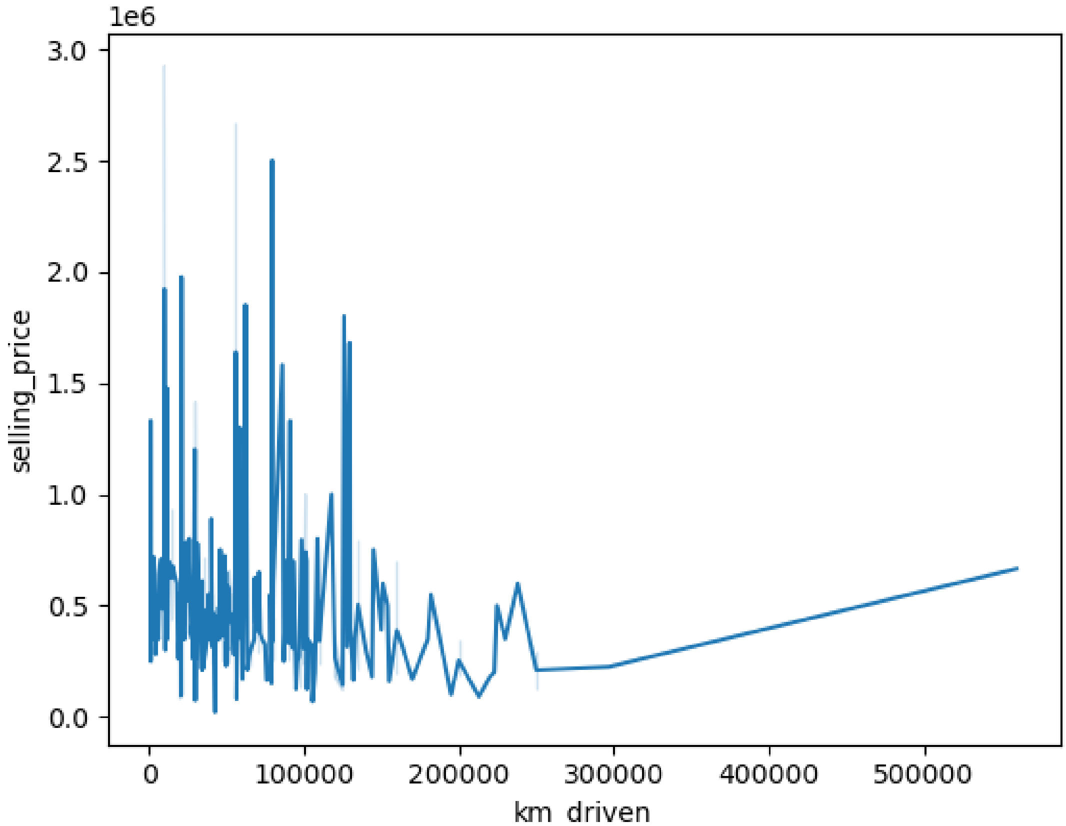
```
C:\Users\prera\AppData\Local\Temp\ipykernel_18460\1121753131.py:2:
FutureWarning: The default value of numeric_only in DataFrame.corr is
deprecated. In a future version, it will default to False. Select only valid
columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(df.corr(), annot = True)
```



[38]:
```
x=df.sample(500)
```

[39]:
```
sns.lineplot(x='km_driven',y="selling_price",data=x)
```

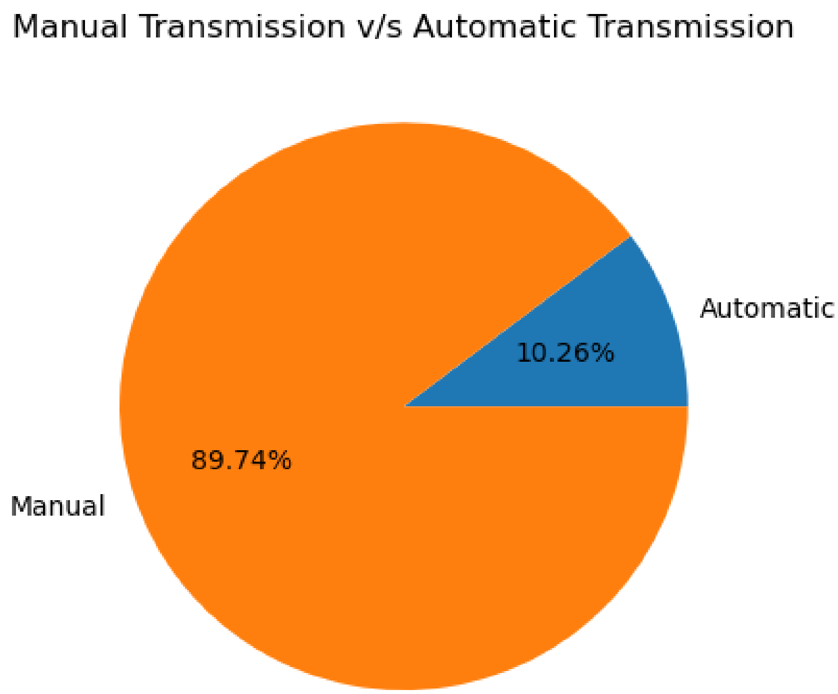
[39]: <Axes: xlabel='km_driven', ylabel='selling_price'>



[104]:
```
b=df.groupby("transmission")["transmission"].count()
```

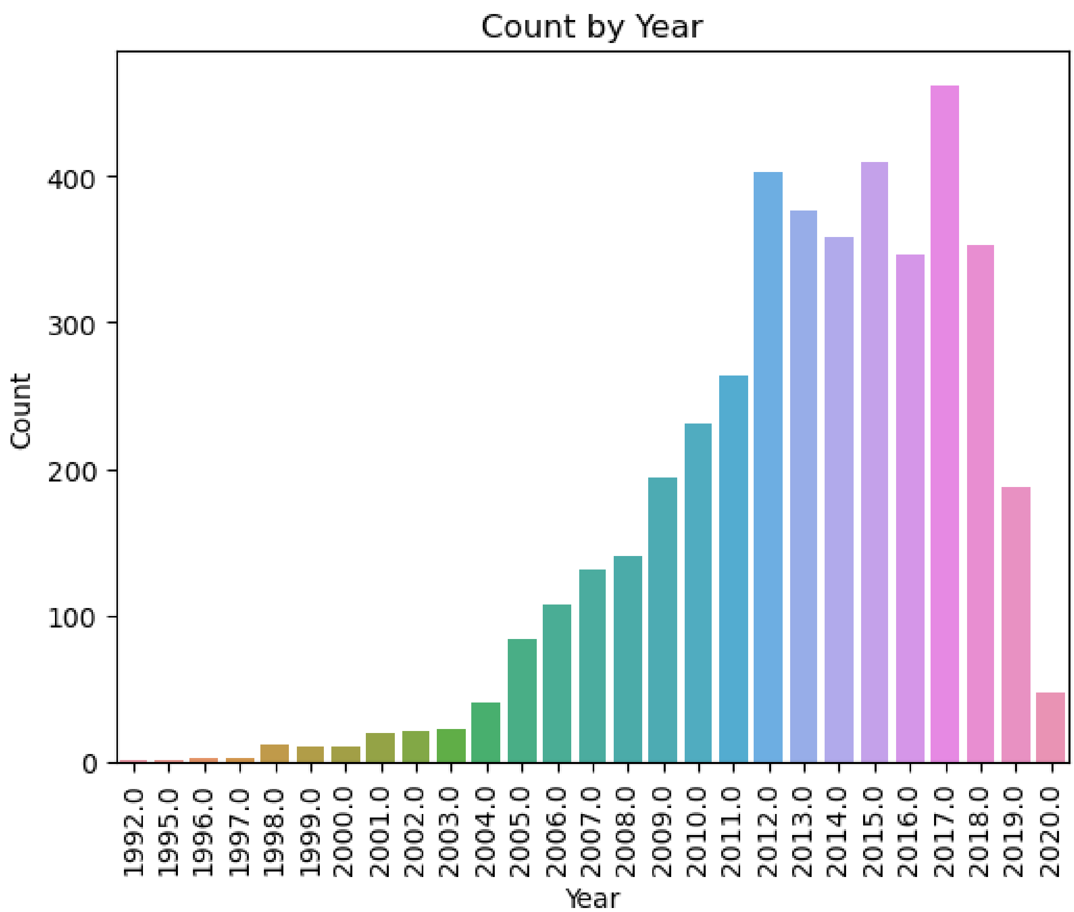
[105]:
```
plt.pie(b,labels=b.index,autopct="%.2f%%")
plt.title("Manual Transmission v/s Automatic Transmission")
```

[105]:
```
Text(0.5, 1.0, 'Manual Transmission v/s Automatic Transmission')
```



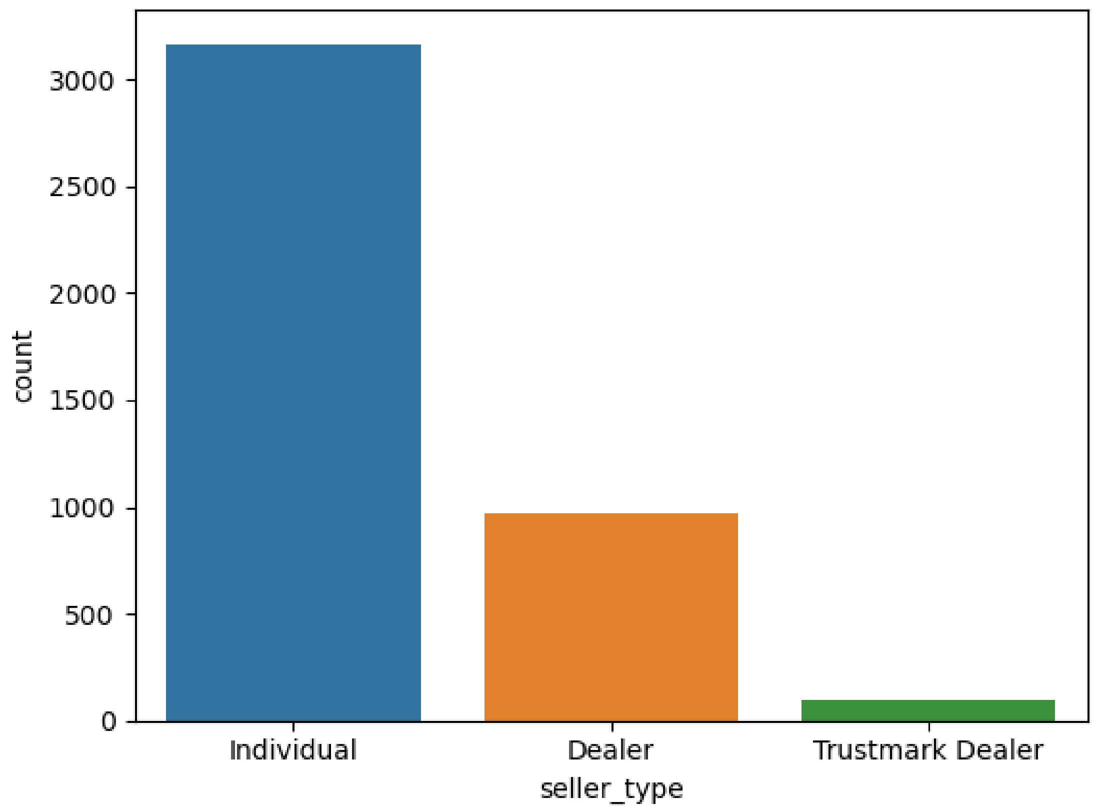
[42]:
```
c=df.groupby("year")["year"].count()
```

[43]:
```
sns.barplot(x=c.index,y=c.values)
plt.xticks(rotation=90)
plt.title("Count by Year")
plt.xlabel("Year")
plt.ylabel("Count")
plt.show()
```



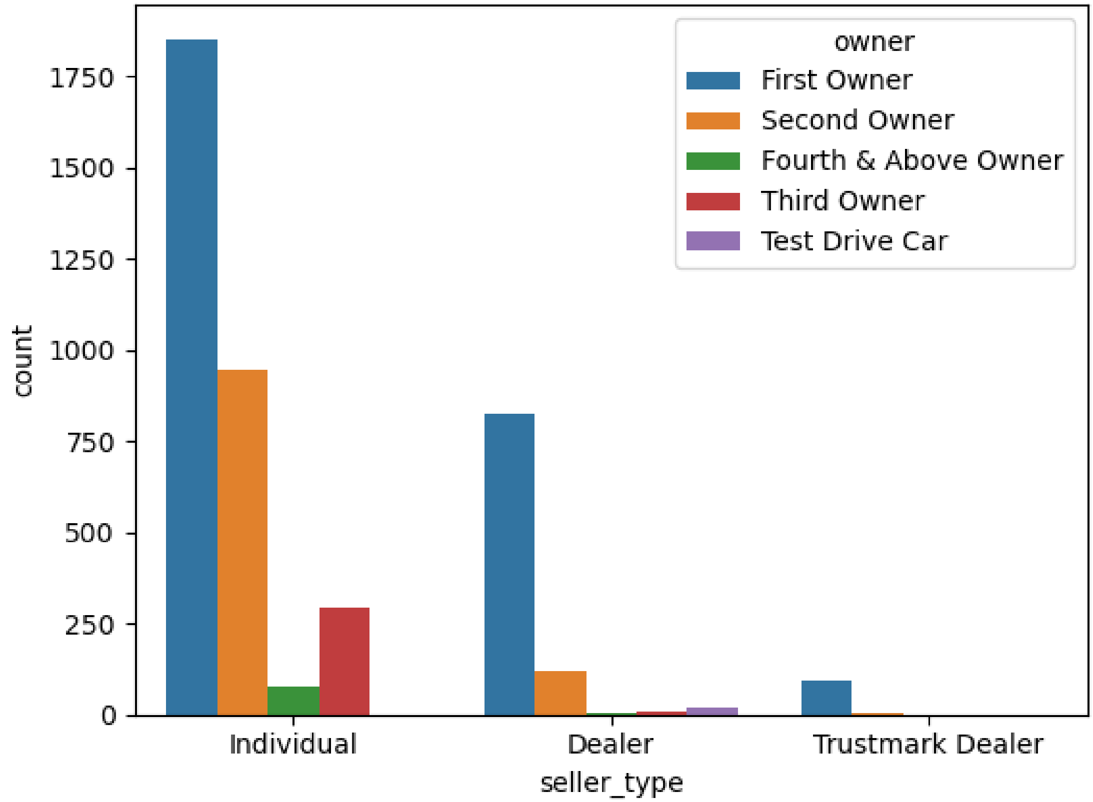
[44]:
```
sns.countplot(x='seller_type',data=df)
```

[44]: <Axes: xlabel='seller_type', ylabel='count'>



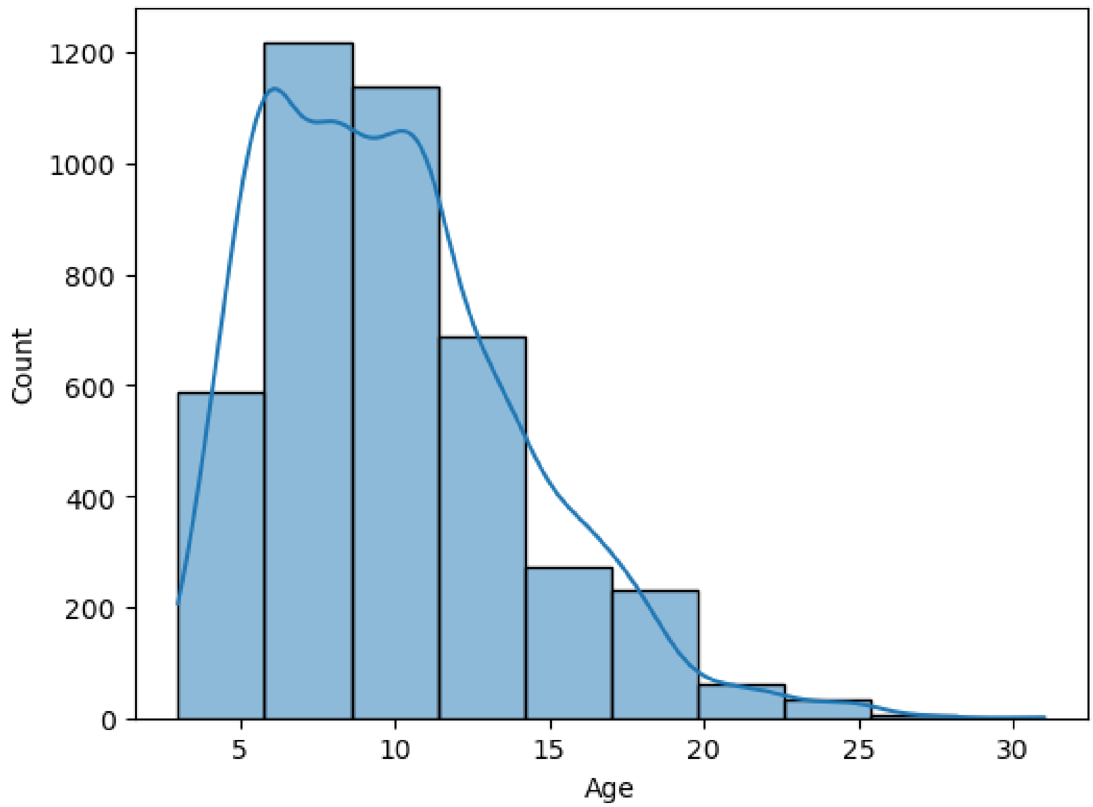
[45]:
```
sns.countplot(x='seller_type',hue='owner',data=df)
```

[45]: <Axes: xlabel='seller_type', ylabel='count'>



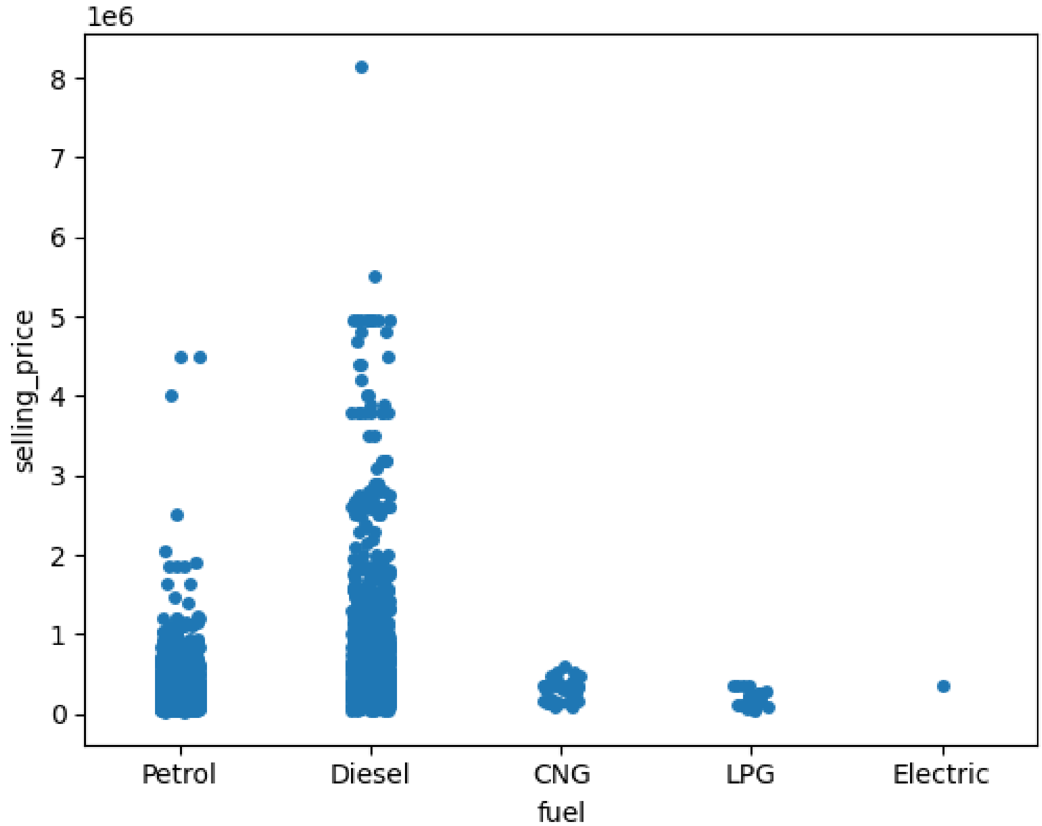
[46]:
```
sns.histplot(df["Age"],bins=10,kde=True)
```

[46]: <Axes: xlabel='Age', ylabel='Count'>



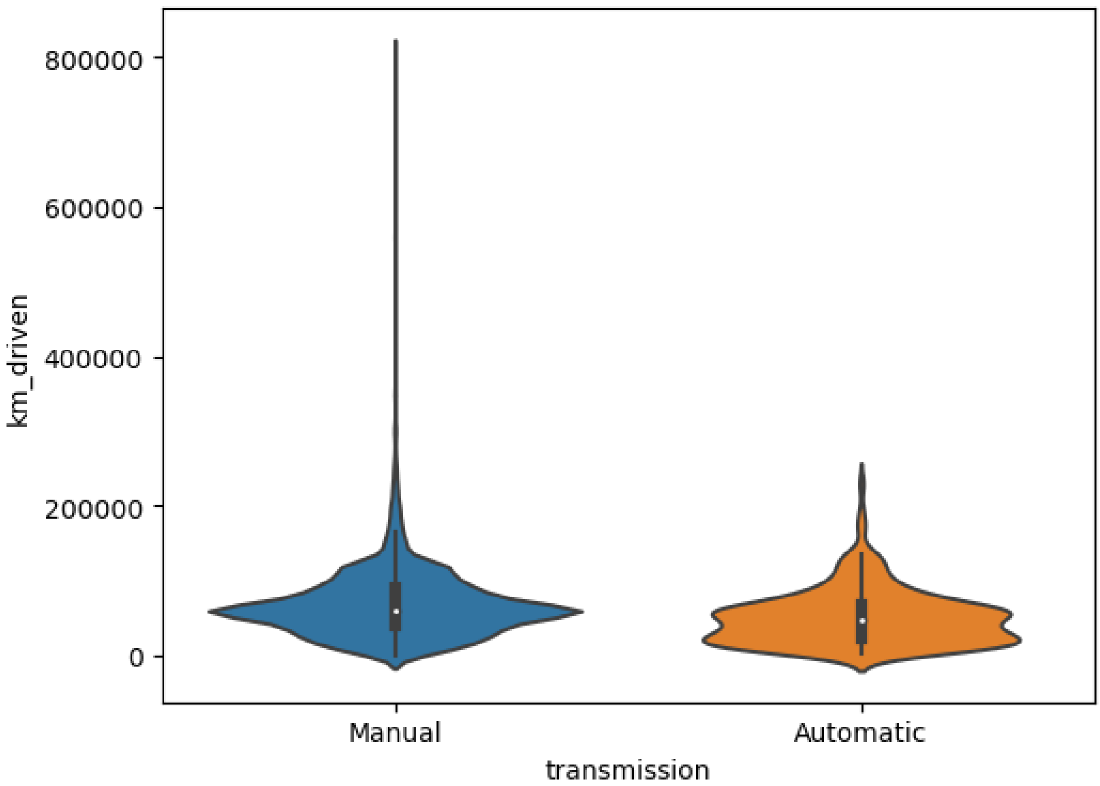
[47]:
```
sns.stripplot(x="fuel",y="selling_price",data=df)
```

[47]: <Axes: xlabel='fuel', ylabel='selling_price'>

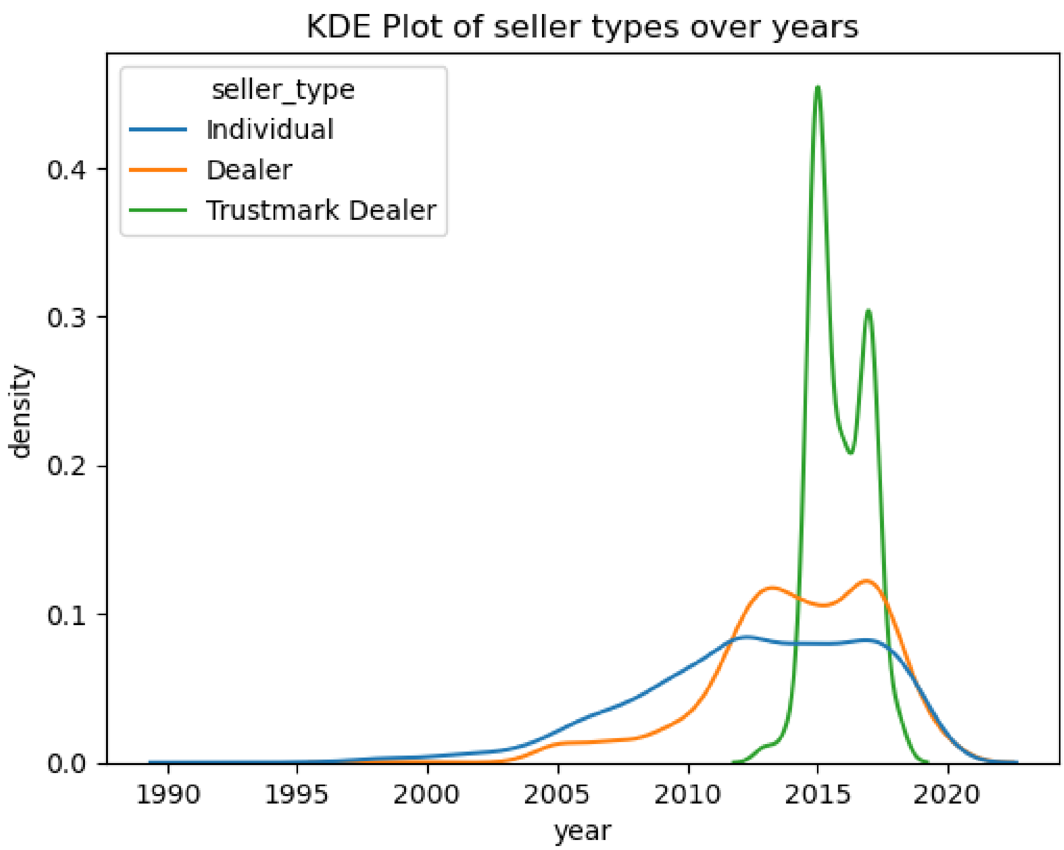



```
[48]: sns.violinplot(x="transmission",y="km_driven",data=df)
```

```
[48]: <Axes: xlabel='transmission', ylabel='km_driven'>
```



[49]:
```
sns.kdeplot(data=df, x='year', hue='seller_type',␣
 ↪common_norm=False,warn_singular=False)
plt.title('KDE Plot of seller types over years')
plt.xlabel('year')
plt.ylabel('density')
plt.show()
```



1 Outlier detection and removal

[50]:
```python
from scipy import stats
z_scores=stats.zscore(df["selling_price"])
z_score_outliers=(z_scores<-3)|(z_scores>3)
```

[51]:
```python
z_score_outlier_rows=df[z_score_outliers]
print("outliers detected by Z-score:",z_score_outlier_rows)
```

```
outliers detected by Z-score:
name    year  \
89    Mercedes-Benz S-Class S 350d Connoisseurs Edition  2017.0
96                                    Audi A8 4.2 TDI  2013.0
101          Mercedes-Benz E-Class Exclusive E 200 BSIV  2018.0
102                              BMW X1 sDrive 20d xLine  2017.0
105                                   BMW 7 Series 730Ld  2012.0
…                                                   …     …
4047                      Volvo XC 90 D5 Inscription BSIV  2017.0
4186                      Toyota Fortuner 2.8 4WD AT BSIV  2017.0
```

```
4224                        Toyota Fortuner 2.7 2WD AT  2014.0
4304                Audi Q5 3.0 TDI Quattro Technology  2018.0
4313                Ford Endeavour 2.2 Titanium AT 4X2  2019.0

      selling_price  km_driven    fuel seller_type transmission         owner  \
89        8150000.0     6500.0  Diesel      Dealer    Automatic   First Owner
96        2800000.0    49000.0  Diesel      Dealer    Automatic   First Owner
101       4500000.0     9800.0  Petrol      Dealer    Automatic   First Owner
102       2750000.0    13000.0  Diesel  Individual    Automatic   First Owner
105       2500000.0    60000.0  Diesel      Dealer    Automatic   First Owner
…               …        …      …         …          …            …
4047      4500000.0    80000.0  Diesel  Individual    Automatic   First Owner
4186      2750000.0    41000.0  Diesel  Individual    Automatic   First Owner
4224      2500000.0    70000.0  Petrol  Individual    Automatic  Second Owner
4304      3899000.0    22000.0  Diesel      Dealer    Automatic   First Owner
4313      2800000.0    10000.0  Diesel  Individual    Automatic   First Owner

       Age
89     6.0
96    10.0
101    5.0
102    6.0
105   11.0
…     …
4047   6.0
4186   6.0
4224   9.0
4304   5.0
4313   4.0

[81 rows x 9 columns]
```

[52]:
```python
df.shape
```

[52]: (4239, 9)

[60]:
```python
new_df.shape
```

[60]: (4158, 9)

[58]:
```python
x=(z_scores>-3)&(z_scores<3)
```

[59]:
```python
new_df=df[x]
```

[61]:
```python
print(new_df)
```

```
                                    name    year  selling_price  km_driven  \
0                          Maruti 800 AC  2007.0        60000.0    70000.0
```

```
1                 Maruti Wagon R LXI Minor  2007.0       135000.0    50000.0
2                     Hyundai Verna 1.6 SX  2012.0       600000.0   100000.0
3                   Datsun RediGO T Option  2017.0       250000.0    46000.0
4                    Honda Amaze VX i-DTEC  2014.0       450000.0   141000.0
…                                      …     …            …          …
4335  Hyundai i20 Magna 1.4 CRDi (Diesel)  2014.0       409999.0    80000.0
4336           Hyundai i20 Magna 1.4 CRDi  2014.0       409999.0    80000.0
4337                  Maruti 800 AC BSIII  2009.0       110000.0    83000.0
4338     Hyundai Creta 1.6 CRDi SX Option  2016.0       865000.0    60000.0
4339                     Renault KWID RXT  2016.0       350000.0    40000.0

        fuel seller_type transmission         owner   Age
0     Petrol  Individual       Manual   First Owner  16.0
1     Petrol  Individual       Manual   First Owner  16.0
2     Diesel  Individual       Manual   First Owner  11.0
3     Petrol  Individual       Manual   First Owner   6.0
4     Diesel  Individual       Manual  Second Owner   9.0
…        …           …          …            …    …
4335  Diesel  Individual       Manual  Second Owner   9.0
4336  Diesel  Individual       Manual  Second Owner   9.0
4337  Petrol  Individual       Manual  Second Owner  14.0
4338  Diesel  Individual       Manual   First Owner   7.0
4339  Petrol  Individual       Manual   First Owner   7.0

[4158 rows x 9 columns]
```

[62]:
```python
z_scores=stats.zscore(new_df["km_driven"])
z_score_outlier=(z_scores<-3)|(z_scores>3)
```

[63]:
```python
z_score_outlier_row=new_df[z_score_outlier]
print("outliers detected by Z-score:",z_score_outlier_row)
```

```
outliers detected by Z-score:
name    year  selling_price  \
69    Chevrolet Tavera Neo LS B3 - 7(C) seats BSIII  2010.0       280000.0
70                    Toyota Corolla Altis Diesel D4DG  2011.0       350000.0
197                                   Mahindra Xylo E4  2009.0       229999.0
225                      Mahindra Renault Logan 1.5 DLS  2008.0        89999.0
324                             Mahindra XUV500 W8 2WD  2012.0       850000.0
394                           Mahindra Scorpio REV 116  2006.0       220000.0
525                    Maruti SX4 S Cross DDiS 320 Delta  2016.0       665000.0
656                              Tata Safari Storme VX  2013.0       360000.0
821                             Hyundai EON Magna Plus  2013.0       125000.0
1101                                   Tata Indica DLS  2006.0        85000.0
1116                Toyota Innova 2.5 V Diesel 7-seater  2005.0       200000.0
1243                             Maruti Swift VXI BSIII  2009.0       250000.0
1253                         Toyota Corolla Altis D-4D J  2014.0       715000.0
1414                Skoda Superb Elegance 2.0 TDI CR AT  2011.0       450000.0
```

1426	Mahindra Scorpio VLX AT 2WD BSIII	2004.0	350000.0
1466	Mahindra Renault Logan 1.5 DLS	2008.0	89999.0
1659	Toyota Innova 2.5 G (Diesel) 8 Seater BS IV	2006.0	229999.0
1668	Toyota Innova 2.5 GX (Diesel) 7 Seater	2014.0	650000.0
1674	Volkswagen Jetta 2.0 TDI Comfortline	2011.0	350000.0
1923	Mahindra Bolero SLE BSIII	2007.0	185000.0
2278	Hyundai Accent CRDi	2006.0	170000.0
2394	Toyota Innova 2.5 V Diesel 8-seater	2009.0	350000.0
2401	Toyota Innova 2.5 E Diesel MS 7-seater	2011.0	350000.0
2402	Mahindra Scorpio 2.6 CRDe	2005.0	175000.0
2672	Maruti Swift Vdi BSIII	2009.0	180000.0
2760	Tata New Safari DICOR 2.2 EX 4x2	2010.0	300000.0
2855	Mahindra Scorpio 2.6 CRDe	2005.0	229999.0
2955	Toyota Innova 2.5 G4 Diesel 7-seater	2007.0	440000.0
2961	Mahindra Scorpio VLS 2.2 mHawk	2008.0	350000.0
2964	Maruti Swift VDI	2012.0	225000.0
3171	Maruti Swift Dzire VDI	2014.0	450000.0
3447	Mahindra Ingenio CRDe	2015.0	210000.0
3461	Toyota Innova 2.5 EV Diesel PS 7 Seater BSIII	2012.0	300000.0
3470	Mahindra Xylo Celebration Edition BSIV	2010.0	200000.0
3531	Ford Endeavour 2.5L 4X2	2011.0	500000.0
3572	Mahindra Scorpio VLX 2WD AIRBAG BSIV	2014.0	600000.0
3611	Hyundai Verna 1.6 SX	2012.0	434999.0
3679	Toyota Innova 2.5 G (Diesel) 7 Seater BS IV	2006.0	350000.0
3718	Toyota Innova 2.5 GX 8 STR BSIV	2009.0	420000.0
3734	Mahindra XUV500 W8 2WD	2013.0	550000.0
3787	Hyundai Santa Fe 4X4	2011.0	800000.0
3898	Tata Indica GLS BS IV	2010.0	350000.0
3979	Mahindra Verito 1.5 D2 BSIII	2011.0	350000.0
3981	Toyota Innova 2.5 VX (Diesel) 8 Seater	2014.0	1030000.0
3994	Tata Indica GLS BS IV	2010.0	75000.0
4088	Maruti 800 AC	2009.0	120000.0
4184	Maruti SX4 S Cross DDiS 320 Delta	2016.0	665000.0
4208	Toyota Qualis FS B3	2013.0	150000.0
4231	Toyota Innova 2.5 G (Diesel) 8 Seater BS IV	2011.0	350000.0
4255	Mahindra XUV500 W8 2WD	2014.0	650000.0
4275	Mahindra XUV500 W8 2WD	2014.0	650000.0
4286	Fiat Punto 1.3 Emotion	2010.0	130000.0

	km_driven	fuel	seller_type	transmission	owner	Age
69	350000.0	Diesel	Individual	Manual	Second Owner	13.0
70	230000.0	Diesel	Individual	Manual	First Owner	12.0
197	230000.0	Diesel	Individual	Manual	Third Owner	14.0
225	213000.0	Diesel	Individual	Manual	First Owner	15.0
324	212814.0	Diesel	Dealer	Manual	First Owner	11.0
394	220000.0	Petrol	Individual	Manual	Second Owner	17.0
525	560000.0	Diesel	Dealer	Manual	First Owner	7.0
656	206500.0	Diesel	Individual	Manual	First Owner	10.0

```
821     205000.0  Petrol  Individual       Manual           First Owner  10.0
1101    300000.0  Diesel  Individual       Manual          Second Owner  17.0
1116    223000.0  Diesel  Individual       Manual           First Owner  18.0
1243    806599.0  Petrol      Dealer       Manual           First Owner  14.0
1253    234000.0  Diesel  Individual       Manual           First Owner   9.0
1414    235000.0  Diesel  Individual    Automatic           First Owner  12.0
1426    223660.0  Diesel  Individual    Automatic           Third Owner  19.0
1466    213000.0  Diesel  Individual       Manual           First Owner  15.0
1659    300000.0  Diesel  Individual       Manual           First Owner  17.0
1668    244000.0  Diesel  Individual       Manual           First Owner   9.0
1674    312000.0  Diesel  Individual       Manual           Third Owner  12.0
1923    230000.0  Diesel  Individual       Manual          Second Owner  16.0
2278    245244.0  Diesel  Individual       Manual  Fourth & Above Owner  17.0
2394    350000.0  Diesel  Individual       Manual           First Owner  14.0
2401    267000.0  Diesel  Individual       Manual          Second Owner  12.0
2402    250000.0  Diesel  Individual       Manual          Second Owner  18.0
2672    220000.0  Diesel  Individual       Manual           First Owner  14.0
2760    250000.0  Diesel  Individual       Manual          Second Owner  13.0
2855    221000.0  Diesel  Individual       Manual           Third Owner  18.0
2955    223000.0  Diesel  Individual       Manual  Fourth & Above Owner  16.0
2961    270000.0  Diesel  Individual       Manual           Third Owner  15.0
2964    296823.0  Diesel  Individual       Manual           First Owner  11.0
3171    260000.0  Diesel  Individual       Manual          Second Owner   9.0
3447    210000.0  Diesel  Individual       Manual           First Owner   8.0
3461    250000.0  Diesel  Individual       Manual           First Owner  11.0
3470    240000.0  Diesel  Individual       Manual           Third Owner  13.0
3531    224642.0  Diesel      Dealer       Manual          Second Owner  12.0
3572    238000.0  Diesel  Individual       Manual           First Owner   9.0
3611    235000.0  Diesel  Individual       Manual          Second Owner  11.0
3679    400000.0  Diesel  Individual       Manual           Third Owner  17.0
3718    347089.0  Diesel      Dealer       Manual           First Owner  14.0
3734    222252.0  Diesel  Individual       Manual           First Owner  10.0
3787    220000.0  Diesel  Individual       Manual           First Owner  12.0
3898    300000.0  Petrol  Individual       Manual           Third Owner  13.0
3979    280000.0  Diesel  Individual       Manual           First Owner  12.0
3981    250000.0  Diesel  Individual       Manual          Second Owner   9.0
3994    300000.0  Petrol  Individual       Manual           Third Owner  13.0
4088    250000.0  Petrol  Individual       Manual          Second Owner  14.0
4184    560000.0  Diesel      Dealer       Manual           First Owner   7.0
4208    256000.0  Diesel      Dealer       Manual           First Owner  10.0
4231    230000.0  Diesel  Individual       Manual           First Owner  12.0
4255    218000.0  Diesel  Individual       Manual          Second Owner   9.0
4275    218000.0  Diesel  Individual       Manual          Second Owner   9.0
4286    210000.0  Diesel  Individual       Manual          Second Owner  13.0
```

[64]:
```
p=(z_scores>-3)&(z_scores<3)
df_new=new_df[p]
```

[65]:
```
df_new
```

[65]:
```
                                  name    year  selling_price  km_driven  \
0                        Maruti 800 AC  2007.0        60000.0    70000.0
1             Maruti Wagon R LXI Minor  2007.0       135000.0    50000.0
2                 Hyundai Verna 1.6 SX  2012.0       600000.0   100000.0
3               Datsun RediGO T Option  2017.0       250000.0    46000.0
4                Honda Amaze VX i-DTEC  2014.0       450000.0   141000.0
…                                  …     …            …          …
4335  Hyundai i20 Magna 1.4 CRDi (Diesel)  2014.0    409999.0    80000.0
4336           Hyundai i20 Magna 1.4 CRDi  2014.0    409999.0    80000.0
4337                  Maruti 800 AC BSIII  2009.0    110000.0    83000.0
4338     Hyundai Creta 1.6 CRDi SX Option  2016.0    865000.0    60000.0
4339                     Renault KWID RXT  2016.0    350000.0    40000.0

        fuel seller_type transmission         owner   Age
0     Petrol  Individual       Manual   First Owner  16.0
1     Petrol  Individual       Manual   First Owner  16.0
2     Diesel  Individual       Manual   First Owner  11.0
3     Petrol  Individual       Manual   First Owner   6.0
4     Diesel  Individual       Manual  Second Owner   9.0
…       …         …           …             …    …
4335  Diesel  Individual       Manual  Second Owner   9.0
4336  Diesel  Individual       Manual  Second Owner   9.0
4337  Petrol  Individual       Manual  Second Owner  14.0
4338  Diesel  Individual       Manual   First Owner   7.0
4339  Petrol  Individual       Manual   First Owner   7.0

[4106 rows x 9 columns]
```

2 Linear Regression

[94]:
```
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
from sklearn.preprocessing import OneHotEncoder,StandardScaler #male=0 female=1␣
  ↪it will remain same
from sklearn.compose import ColumnTransformer
import joblib #convert whole model into binary format and save it pkl
```

[67]:
```
categorical_cols=['fuel','seller_type','transmission','owner']
encoder=OneHotEncoder(drop='first',sparse=False)
encoder_cols=pd.DataFrame(encoder.
  ↪fit_transform(df_new[categorical_cols]),columns=encoder.
  ↪get_feature_names_out(categorical_cols))
numerical_cols=['year','km_driven','Age']
```

```
scaler=StandardScaler()
scaled_cols=pd.DataFrame(scaler.
 ↪fit_transform(df_new[numerical_cols]),columns=scaler.
 ↪get_feature_names_out(numerical_cols))
```

```
C:\Users\prera\anaconda3\Lib\site-
packages\sklearn\preprocessing\_encoders.py:868: FutureWarning: `sparse` was
renamed to `sparse_output` in version 1.2 and will be removed in 1.4.
`sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

[68]:

```
encoder_cols
```

[68]:

```
      fuel_Diesel  fuel_Electric  fuel_LPG  fuel_Petrol  \
0             0.0            0.0       0.0          1.0
1             0.0            0.0       0.0          1.0
2             1.0            0.0       0.0          0.0
3             0.0            0.0       0.0          1.0
4             1.0            0.0       0.0          0.0
…             …              …         …            …
4101          1.0            0.0       0.0          0.0
4102          1.0            0.0       0.0          0.0
4103          0.0            0.0       0.0          1.0
4104          1.0            0.0       0.0          0.0
4105          0.0            0.0       0.0          1.0

      seller_type_Individual  seller_type_Trustmark Dealer  \
0                        1.0                           0.0
1                        1.0                           0.0
2                        1.0                           0.0
3                        1.0                           0.0
4                        1.0                           0.0
…                        …                             …
4101                     1.0                           0.0
4102                     1.0                           0.0
4103                     1.0                           0.0
4104                     1.0                           0.0
4105                     1.0                           0.0

      transmission_Manual  owner_Fourth & Above Owner  owner_Second Owner  \
0                     1.0                         0.0                 0.0
1                     1.0                         0.0                 0.0
2                     1.0                         0.0                 0.0
3                     1.0                         0.0                 0.0
4                     1.0                         0.0                 1.0
…                     …                           …                   …
4101                  1.0                         0.0                 1.0
```

```
4102                  1.0                         0.0                   1.0
4103                  1.0                         0.0                   1.0
4104                  1.0                         0.0                   0.0
4105                  1.0                         0.0                   0.0

      owner_Test Drive Car  owner_Third Owner
0                      0.0                0.0
1                      0.0                0.0
2                      0.0                0.0
3                      0.0                0.0
4                      0.0                0.0
…                    …                 …
4101                   0.0                0.0
4102                   0.0                0.0
4103                   0.0                0.0
4104                   0.0                0.0
4105                   0.0                0.0

[4106 rows x 11 columns]
```

[69]:
```
scaled_cols
```

[69]:
```
          year  km_driven       Age
0    -1.434263   0.172104  1.434263
1    -1.434263  -0.369703  1.434263
2    -0.247618   0.984815  0.247618
3     0.939028  -0.478065 -0.939028
4     0.227041   2.095520 -0.227041
…          …          …        …
4101  0.227041   0.443008 -0.227041
4102  0.227041   0.443008 -0.227041
4103 -0.959605   0.524279  0.959605
4104  0.701699  -0.098800 -0.701699
4105  0.701699  -0.640607 -0.701699

[4106 rows x 3 columns]
```

[70]:
```
X=pd.concat([encoder_cols,scaled_cols],axis=1)
Y=df_new['selling_price']
```

[71]:
```
X_train,X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.
 ↪2,random_state=42)
```

[72]:
```
model=LinearRegression()
model.fit(X_train,Y_train)
y_pred=model.predict(X_test)
```

[73]:
```
print(model.intercept_) #y-intercept of the model
```

450630.40501939977

[74]:
```
print(model.coef_)
```

```
[ 1.87891430e+05 -1.89286477e+05  2.11278815e+03 -1.09207793e+04
 -2.57172544e+04  1.26652592e+05 -3.40613893e+05 -4.72746959e+04
 -2.79143810e+04  2.94865509e+05 -4.72866455e+04 -8.82644749e+18
 -2.70243750e+04 -8.82644749e+18]
```

[75]:
```
mae = mean_absolute_error(Y_test,y_pred)
mse= mean_squared_error(Y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(Y_test, y_pred)

print('Mean Absolute Error',mae)
print('Mean Squared Error',mse)
print('Root Mean Absolute Error',rmse)
print('R2 Score',r2)
```

```
Mean Absolute Error 162652.5674116384
Mean Squared Error 50534163119.10173
Root Mean Absolute Error 224798.0496336695
R2 Score 0.4320817327039559
```

[76]:
```
#adjusted_r2=1-[(1-r2)*(n-1)/(n-k-1)]
adjusted_r2=1-((1-0.43205)*(4106-1)/(4106-11-1))
print('adjusted r2 is :',adjusted_r2)
```

adjusted r2 is : 0.4305239985344407

[77]:
```
y_mean=np.mean(Y_test)
SSR = np.sum((y_pred - y_mean) ** 2)
SSR
```

[77]: 33701119074990.78

[78]:
```
SST = np.sum((Y_test - y_mean) ** 2)
SST
```

[78]: 73142711682221.97

[79]:
```
SSE=SST-SSR
SSE
```

[79]: 39441592607231.19

[80]:
```
b=pd.DataFrame({"Actual":Y_test,"Predicted":y_pred})
b
```

[80]:
```
        Actual      Predicted
2649  275000.0  387654.405019
621   750000.0  713798.405019
809   851000.0  581702.405019
1186  170000.0  500454.405019
3210  450000.0  490822.405019
…          …              …
2311  275000.0  348998.405019
274   650000.0  430150.405019
2962  275000.0  422470.405019
2473  270000.0  454214.405019
1679  350000.0  494150.405019

[822 rows x 2 columns]
```

[81]:
```
sns.scatterplot(x=Y_test,y=y_pred)
plt.xlabel('Actual Selling Price')
plt.ylabel('Predicted Selling Price')
plt.title('Actual Selling Price v/s Predicted Selling Price for Vehicle')
```

[81]:
```
Text(0.5, 1.0, 'Actual Selling Price v/s Predicted Selling Price for Vehicle')
```

[82]:
```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import Ridge,Lasso
```

[83]:
```
lr_model=LinearRegression()
lr_scores=cross_val_score(lr_model,X_train,Y_train,cv=5)
```

[84]:
```
lasso_model=Lasso(alpha=1.0)
lassso_scores=cross_val_score(lasso_model,X_train,Y_train,cv=5)
```

[85]:
```
ridge_model=Ridge(alpha=1.0)
ridge_scores=cross_val_score(ridge_model,X_train,Y_train,cv=5)
```

[86]:
```
lr_model.fit(X_train,Y_train)
lr_prediction =lr_model.predict(X_test)
lr_mae =mean_absolute_error(Y_test,lr_prediction)
lr_mse =mean_squared_error(Y_test,lr_prediction)
lr_rmse = np.sqrt(lr_mse)
lr_r2 = r2_score(Y_test,lr_prediction)
print('Linear mae',lr_mae)
```

```
print('Linear mse',lr_mse)
print('Linear rmse',lr_rmse)
print('Linear r2',lr_r2)
```

```
Linear mae 162652.5674116384
Linear mse 50534163119.10173
Linear rmse 224798.0496336695
Linear r2 0.4320817327039559
```

[87]:
```
lasso_model.fit(X_train,Y_train)
lasso_prediction =lasso_model.predict(X_test)
lasso_mae =mean_absolute_error(Y_test,lasso_prediction)
lasso_mse =mean_squared_error(Y_test,lasso_prediction)
lasso_rmse = np.sqrt(lasso_mse)
lasso_r2 = r2_score(Y_test,lr_prediction)
print('Lasso mae',lasso_mae)
print('Lasso mse',lasso_mse)
print('Lasso rmse',lasso_rmse)
print('Lasso r2',lasso_r2)
```

```
Lasso mae 162992.86828325025
Lasso mse 50535857645.50097
Lasso rmse 224801.8185991852
Lasso r2 0.4320817327039559
```

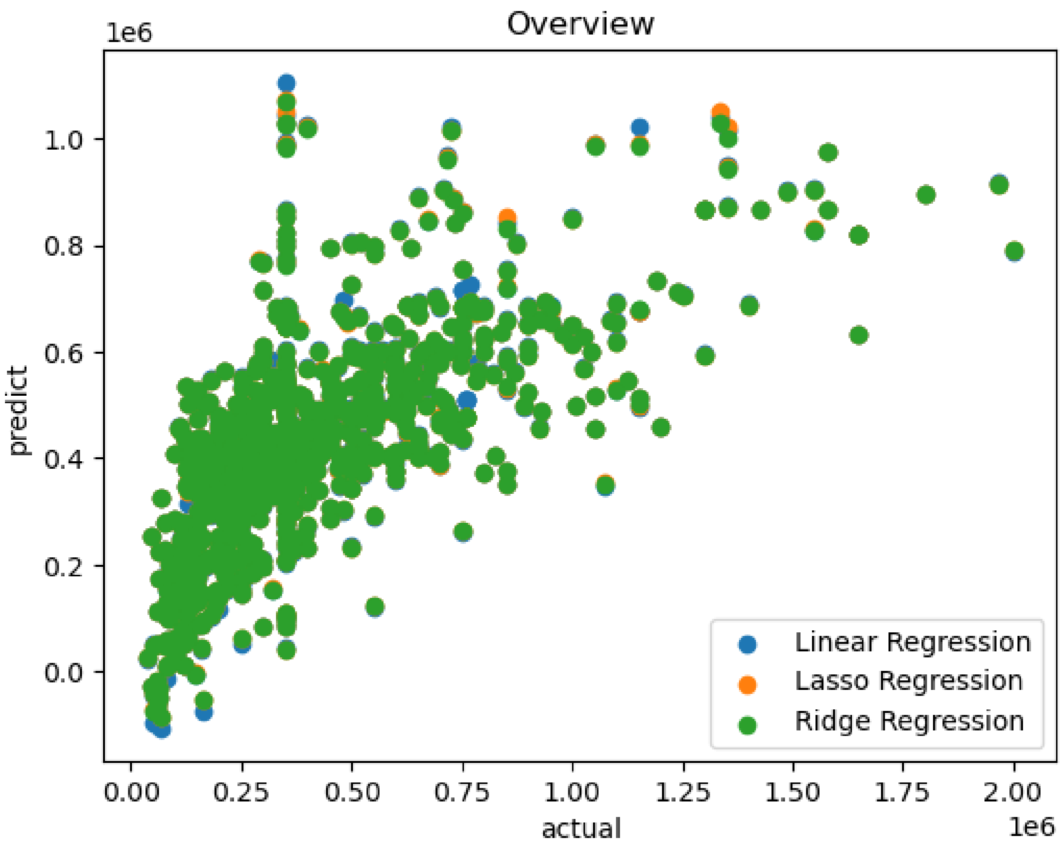
[88]:
```
ridge_model.fit(X_train,Y_train)
ridge_prediction =ridge_model.predict(X_test)
ridge_mae =mean_absolute_error(Y_test,ridge_prediction)
ridge_mse =mean_squared_error(Y_test,ridge_prediction)
ridge_rmse = np.sqrt(ridge_mse)
ridge_r2 = r2_score(Y_test,ridge_prediction)
print('ridge mae',ridge_mae)
print('ridge mse',ridge_mse)
print('ridge rmse',ridge_rmse)
print('ridge r2',ridge_r2)
```

```
ridge mae 163049.62129809914
ridge mse 50515182474.76968
ridge rmse 224755.82856684647
ridge r2 0.43229504294748
```

[89]:
```
plt.scatter(Y_test,lr_prediction,alpha=1.0,label='Linear Regression')
plt.scatter(Y_test,lasso_prediction,alpha=1.0,label='Lasso Regression')
plt.scatter(Y_test,ridge_prediction,alpha=1.0,label='Ridge Regression')
plt.xlabel('actual')
plt.ylabel('predict')
plt.title('Overview')
```

```
plt.legend()
```

[89]: <matplotlib.legend.Legend at 0x2189e5437d0>



3 Robust Techniques

[90]:
```
# MM estimator:huberregression
from sklearn.linear_model import HuberRegressor
X_scaled = scaler.fit_transform(X_test)
huber = HuberRegressor(epsilon=1.35)
huber.fit(X_scaled, Y_test)
huber_prediction = huber.predict(X_scaled)
huber_mae =mean_absolute_error(Y_test,huber_prediction)
huber_mse =mean_squared_error(Y_test,huber_prediction)
huber_rmse = np.sqrt(huber_mse)
huber_r2 = r2_score(Y_test,huber_prediction)
print('huber mae:',huber_mae)
print('huber mse:',huber_mse)
print('huber rmse:',huber_rmse)
```

```
print('huber r2:',huber_r2)
```

```
huber mae: 151509.51960494742
huber mse: 52306829051.87994
huber rmse: 228706.86271268717
huber r2: 0.41215997477030997
```

[91]:

```
# MM estimate: RANSAC regression
from sklearn.linear_model import RANSACRegressor
from sklearn.datasets import make_regression
ransac = RANSACRegressor()
mm= ransac.fit(X_test, Y_test)
mm_estimate_coeff = ransac.estimator_.coef_
mm_estimate_intercept = ransac.estimator_.intercept_
mm_prediction = ransac.predict(X_test)
print("MM Estimate Coefficients:", mm_estimate_coeff)
print("MM Estimate Intercept:", mm_estimate_intercept)
mm_mae =mean_absolute_error(Y_test,mm_prediction)
mm_mse =mean_squared_error(Y_test,mm_prediction)
mm_rmse = np.sqrt(mm_mse)
mm_r2 = r2_score(Y_test,huber_prediction)
print('mm mae:',mm_mae)
print('mm mse:',mm_mse)
print('mm rmse:',mm_rmse)
print('mm r2:',mm_r2)
```

```
MM Estimate Coefficients: [ 9.73663682e+04 -3.18323146e-11 -2.28897470e+04
1.02924682e+04
 -1.11536088e+05 -5.85164524e+04 -1.65343121e+05 -5.42858739e+03
 -1.15995306e+04  0.00000000e+00  6.98193723e+03  4.97283085e+04
 -2.19533011e+03 -4.97283085e+04]
MM Estimate Intercept: 506712.60891998676
mm mae: 163132.59291319893
mm mse: 63118785024.49366
mm rmse: 251234.52196004763
mm r2: 0.41215997477030997
```

[92]:

```
# lts estimate
from sklearn.linear_model import RANSACRegressor
ransac = RANSACRegressor()

ransac.fit(X_test, Y_test)

lts_estimate_coeff = ransac.estimator_.coef_
lts_estimate_intercept = ransac.estimator_.intercept_

print("LTS Estimate Coefficients:", lts_estimate_coeff)
```

```
print("LTS Estimate Intercept:", lts_estimate_intercept)

lts_prediction = ransac.predict(X_test)
lts_mae =mean_absolute_error(Y_test,lts_prediction)
lts_mse =mean_squared_error(Y_test,lts_prediction)
lts_rmse = np.sqrt(lts_mse)
lts_r2 = r2_score(Y_test,huber_prediction)
print('lts mae:',lts_mae)
print('lts mse:',lts_mse)
print('lts rmse:',lts_rmse)
print('lts r2:',lts_r2)
```

```
LTS Estimate Coefficients: [ 9.24554405e+04 -3.63797881e-11 -8.73447364e+04
-8.90569007e+03
 -3.36927436e+04  3.20507734e+05  1.51078643e+04  9.94653712e+03
 -2.75691832e+04  7.96513225e+05 -1.50626456e+04  4.98090846e+04
  2.27336327e+04 -4.98090846e+04]
LTS Estimate Intercept: 322497.542062135
lts mae: 160854.6360410159
lts mse: 66852662530.40172
lts rmse: 258558.81831877583
lts r2: 0.41215997477030997
```

[93]:
```
# theil sen regressor
from sklearn.linear_model import TheilSenRegressor

# Create a Theil-Sen estimator model
theil_sen = TheilSenRegressor()

# Fit the model to the data
theil_sen.fit(X_test, Y_test)

# Get the Theil-Sen estimate of the coefficients
theil_sen_estimate_intercept = theil_sen.intercept_
theil_sen_estimate_coefficient = theil_sen.coef_[0]
print("Theil-Sen Estimate Intercept:", theil_sen_estimate_intercept)
print("Theil-Sen Estimate Coefficient:", theil_sen_estimate_coefficient)

ts_prediction = theil_sen.predict(X_test)
ts_mae =mean_absolute_error(Y_test,ts_prediction)
ts_mse =mean_squared_error(Y_test,ts_prediction)
ts_rmse = np.sqrt(ts_mse)
ts_r2 = r2_score(Y_test,ts_prediction)
print('ts mae:',ts_mae)
print('ts mse:',ts_mse)
print('ts rmse:',ts_rmse)
print('ts r2:',ts_r2)
```

```
Theil-Sen Estimate Intercept: 358969.13287727424
Theil-Sen Estimate Coefficient: 250621.28636068667
ts mae: 156739.87098063715
ts mse: 53414070816.13874
ts rmse: 231114.84334879648
ts r2: 0.3997164556651527
```

[96]:
```
from sklearn.ensemble import GradientBoostingRegressor
```

[97]:
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[98]:
```
gb_param_grid = {'n_estimators': [100, 300, 500], 'max_depth': [3, 5, 7],␣
 ↪'learning_rate': [0.01, 0.1, 0.2]}
gb_model = GradientBoostingRegressor(random_state=42)
gb_grid_search = GridSearchCV(estimator=gb_model, param_grid=gb_param_grid,␣
 ↪scoring='neg_mean_squared_error', cv=5)
gb_grid_search.fit(X_train_scaled, Y_train)
best_gb_params = gb_grid_search.best_params_

gradient_boosting_model = GradientBoostingRegressor(**best_gb_params,␣
 ↪random_state=42)

gradient_boosting_model.fit(X_train_scaled, Y_train)
```

[98]:
```
GradientBoostingRegressor(learning_rate=0.01, n_estimators=500, random_state=42)
```

[99]:
```
gradboost_predictions = gradient_boosting_model.predict(X_test_scaled)
```

[101]:
```
gb_mae = mean_absolute_error(Y_test, gradboost_predictions)
gb_mse = mean_squared_error(Y_test, gradboost_predictions)
gb_rmse = np.sqrt(gb_mse)
gb_r2 = r2_score(Y_test, gradboost_predictions)

print('Gradient boosting mean absolute error:', gb_mae)
print('Gradient boosting mean squared error:', gb_mse)
print('Gradient boosting root mean squared error:', gb_rmse)
print('Gradient boosting R2 score:', gb_r2)
```

```
Gradient boosting mean absolute error: 150006.42071677113
Gradient boosting mean squared error: 45755254193.22717
Gradient boosting root mean squared error: 213904.77833191847
Gradient boosting R2 score: 0.485788562088895
```

[ ]:
```

```