```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.preprocessing import StandardScaler

z=pd.read_csv('D:/KGISL MICRO COLL/Domain class/Milestone 3/Car
details v3.csv')
z
```

```
                              name  year  selling_price  km_driven
fuel  \
0           Maruti Swift Dzire VDI  2014         450000     145500
Diesel
1     Skoda Rapid 1.5 TDI Ambition  2014         370000     120000
Diesel
2          Honda City 2017-2020 EXi  2006         158000     140000
Petrol
3        Hyundai i20 Sportz Diesel  2010         225000     127000
Diesel
4           Maruti Swift VXI BSIII  2007         130000     120000
Petrol
...                            ...   ...            ...        ...
...
8123            Hyundai i20 Magna  2013         320000     110000
Petrol
8124         Hyundai Verna CRDi SX  2007         135000     119000
Diesel
8125        Maruti Swift Dzire ZDi  2009         382000     120000
Diesel
8126               Tata Indigo CR4  2013         290000      25000
Diesel
8127               Tata Indigo CR4  2013         290000      25000
Diesel

     seller_type transmission                 owner     mileage
engine  \
0     Individual       Manual           First Owner   23.4 kmpl  1248
CC
1     Individual       Manual          Second Owner  21.14 kmpl  1498
CC
2     Individual       Manual           Third Owner   17.7 kmpl  1497
CC
3     Individual       Manual           First Owner   23.0 kmpl  1396
CC
4     Individual       Manual           First Owner   16.1 kmpl  1298
CC
...          ...          ...                   ...         ...      .
..
```

```
8123  Individual       Manual           First Owner   18.5 kmpl  1197
CC
8124  Individual       Manual  Fourth & Above Owner   16.8 kmpl  1493
CC
8125  Individual       Manual           First Owner   19.3 kmpl  1248
CC
8126  Individual       Manual           First Owner  23.57 kmpl  1396
CC
8127  Individual       Manual           First Owner  23.57 kmpl  1396
CC

       max_power                      torque  seats
0         74 bhp             190Nm@ 2000rpm    5.0
1     103.52 bhp         250Nm@ 1500-2500rpm    5.0
2         78 bhp       12.7@ 2,700(kgm@ rpm)    5.0
3         90 bhp    22.4 kgm at 1750-2750rpm    5.0
4       88.2 bhp       11.5@ 4,500(kgm@ rpm)    5.0
...          ...                         ...    ...
8123   82.85 bhp           113.7Nm@ 4000rpm    5.0
8124     110 bhp  24@ 1,900-2,750(kgm@ rpm)    5.0
8125    73.9 bhp             190Nm@ 2000rpm    5.0
8126      70 bhp         140Nm@ 1800-3000rpm    5.0
8127      70 bhp         140Nm@ 1800-3000rpm    5.0

[8128 rows x 13 columns]
```

```
z.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8128 entries, 0 to 8127
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   name           8128 non-null   object
 1   year           8128 non-null   int64
 2   selling_price  8128 non-null   int64
 3   km_driven      8128 non-null   int64
 4   fuel           8128 non-null   object
 5   seller_type    8128 non-null   object
 6   transmission   8128 non-null   object
 7   owner          8128 non-null   object
 8   mileage        7907 non-null   object
 9   engine         7907 non-null   object
 10  max_power      7913 non-null   object
 11  torque         7906 non-null   object
 12  seats          7907 non-null   float64
dtypes: float64(1), int64(3), object(9)
memory usage: 825.6+ KB
```

```
z['mileage'].unique()
```

```
array(['23.4 kmpl', '21.14 kmpl', '17.7 kmpl', '23.0 kmpl', '16.1
kmpl',
       '20.14 kmpl', '17.3 km/kg', '23.59 kmpl', '20.0 kmpl',
       '19.01 kmpl', '17.3 kmpl', '19.3 kmpl', nan, '18.9 kmpl',
       '18.15 kmpl', '24.52 kmpl', '19.7 kmpl', '22.54 kmpl', '21.0
kmpl',
       '25.5 kmpl', '26.59 kmpl', '21.5 kmpl', '20.3 kmpl', '21.4
kmpl',
       '24.7 kmpl', '18.2 kmpl', '16.8 kmpl', '24.3 kmpl', '14.0
kmpl',
       '18.6 kmpl', '33.44 km/kg', '23.95 kmpl', '17.0 kmpl',
       '20.63 kmpl', '13.93 kmpl', '16.0 kmpl', '17.8 kmpl', '18.5
kmpl',
       '12.55 kmpl', '12.99 kmpl', '14.8 kmpl', '13.5 kmpl', '26.0
kmpl',
       '20.65 kmpl', '27.3 kmpl', '11.36 kmpl', '17.68 kmpl',
       '14.28 kmpl', '18.53 kmpl', '14.84 kmpl', '21.12 kmpl',
       '20.36 kmpl', '21.27 kmpl', '18.16 kmpl', '22.0 kmpl', '25.1
kmpl',
       '20.51 kmpl', '21.66 kmpl', '25.2 kmpl', '22.9 kmpl', '16.02
kmpl',
       '20.54 kmpl', '22.77 kmpl', '15.71 kmpl', '23.1 kmpl',
       '19.02 kmpl', '19.81 kmpl', '26.2 km/kg', '16.47 kmpl',
       '15.04 kmpl', '19.1 kmpl', '21.79 kmpl', '18.8 kmpl', '21.21
kmpl',
       '15.37 kmpl', '11.79 kmpl', '19.0 kmpl', '14.3 kmpl', '15.8
kmpl',
       '15.1 kmpl', '19.09 kmpl', '22.32 kmpl', '21.9 kmpl', '14.53
kmpl',
       '21.63 kmpl', '20.85 kmpl', '20.45 kmpl', '19.67 kmpl',
       '23.01 kmpl', '20.77 kmpl', '17.92 kmpl', '17.01 kmpl',
       '22.37 kmpl', '19.33 kmpl', '9.5 kmpl', '12.83 kmpl', '22.48
kmpl',
       '16.78 kmpl', '14.67 kmpl', '15.0 kmpl', '13.96 kmpl', '18.0
kmpl',
       '12.07 kmpl', '26.21 kmpl', '10.8 kmpl', '16.3 kmpl', '13.6
kmpl',
       '14.74 kmpl', '15.6 kmpl', '19.56 kmpl', '22.69 kmpl',
       '19.16 kmpl', '18.12 kmpl', '12.1 kmpl', '17.5 kmpl', '42.0
kmpl',
       '20.4 kmpl', '21.1 kmpl', '19.44 kmpl', '13.0 kmpl', '21.43
kmpl',
       '22.95 kmpl', '16.2 kmpl', '15.3 kmpl', '28.09 kmpl', '17.4
kmpl',
       '19.4 kmpl', '26.6 km/kg', '17.6 kmpl', '28.4 kmpl', '14.1
kmpl',
       '25.17 kmpl', '22.74 kmpl', '17.57 kmpl', '16.95 kmpl',
       '19.49 kmpl', '17.21 kmpl', '13.2 kmpl', '14.2 kmpl', '26.8
kmpl',
       '25.4 kmpl', '11.5 kmpl', '27.28 kmpl', '17.97 kmpl', '12.8
```

```
kmpl',
       '16.55 kmpl', '12.05 kmpl', '14.07 kmpl', '21.02 kmpl',
       '11.57 kmpl', '17.9 kmpl', '15.96 kmpl', '17.1 kmpl', '17.19
kmpl',
       '21.01 kmpl', '24.0 kmpl', '25.6 kmpl', '21.38 kmpl', '23.84
kmpl',
       '23.08 kmpl', '14.24 kmpl', '20.71 kmpl', '15.64 kmpl',
       '14.5 kmpl', '16.34 kmpl', '27.39 kmpl', '11.1 kmpl', '13.9
kmpl',
       '20.88 km/kg', '20.92 kmpl', '23.8 kmpl', '24.4 kmpl',
       '15.29 kmpl', '21.19 kmpl', '22.5 kmpl', '19.6 kmpl', '23.65
kmpl',
       '25.32 kmpl', '23.5 kmpl', '16.6 kmpl', '23.9 kmpl', '20.8
kmpl',
       '27.62 kmpl', '12.9 kmpl', '25.44 kmpl', '17.88 kmpl', '22.7
kmpl',
       '17.2 kmpl', '15.42 kmpl', '19.68 kmpl', '18.7 kmpl', '15.4
kmpl',
       '19.34 kmpl', '22.71 kmpl', '25.8 kmpl', '13.7 kmpl', '12.2
kmpl',
       '18.49 kmpl', '9.0 kmpl', '0.0 kmpl', '13.58 kmpl', '10.1
kmpl',
       '20.5 kmpl', '25.0 kmpl', '10.5 kmpl', '22.07 kmpl', '22.3
kmpl',
       '15.26 kmpl', '20.62 kmpl', '27.4 kmpl', '23.2 kmpl', '14.4
kmpl',
       '18.4 kmpl', '30.46 km/kg', '14.02 kmpl', '11.0 kmpl', '20.6
kmpl',
       '22.05 kmpl', '20.2 kmpl', '18.1 kmpl', '22.1 kmpl', '19.87
kmpl',
       '13.01 kmpl', '18.06 kmpl', '26.1 kmpl', '16.52 kmpl',
       '13.55 kmpl', '24.2 kmpl', '25.83 kmpl', '11.2 kmpl', '17.09
kmpl',
       '21.03 kmpl', '17.45 kmpl', '21.64 kmpl', '21.94 km/kg',
       '13.87 kmpl', '19.98 kmpl', '20.52 kmpl', '23.57 kmpl',
       '11.7 kmpl', '17.43 kmpl', '18.88 kmpl', '13.68 kmpl',
       '11.18 kmpl', '20.89 kmpl', '11.8 kmpl', '19.62 kmpl', '21.7
kmpl',
       '14.9 kmpl', '19.5 kmpl', '10.91 kmpl', '15.7 kmpl', '20.73
kmpl',
       '15.85 kmpl', '20.7 kmpl', '14.23 kmpl', '16.5 kmpl', '17.36
kmpl',
       '12.6 kmpl', '16.36 kmpl', '14.95 kmpl', '16.9 kmpl', '19.2
kmpl',
       '16.96 kmpl', '22.15 kmpl', '18.78 kmpl', '19.61 kmpl',
       '17.71 kmpl', '18.3 kmpl', '19.12 kmpl', '19.72 kmpl', '12.0
kmpl',
       '11.4 kmpl', '23.03 kmpl', '11.07 kmpl', '15.9 kmpl', '17.67
kmpl',
       '20.46 kmpl', '13.1 kmpl', '13.45 km/kg', '24.8 kmpl',
```

```
       '15.73 kmpl', '15.11 kmpl', '12.7 kmpl', '21.2 kmpl', '20.38
kmpl',
       '21.56 kmpl', '13.22 kmpl', '14.49 kmpl', '15.05 kmpl',
       '23.26 kmpl', '15.41 kmpl', '13.8 kmpl', '22.27 kmpl',
       '32.52 km/kg', '14.66 kmpl', '12.12 kmpl', '16.84 kmpl',
       '14.09 kmpl', '14.7 kmpl', '13.4 kmpl', '15.5 kmpl', '13.49
kmpl',
       '11.88 km/kg', '14.6 kmpl', '10.75 kmpl', '24.5 kmpl',
       '11.74 kmpl', '16.07 kmpl', '15.63 kmpl', '26.3 km/kg',
       '23.7 km/kg', '25.47 kmpl', '17.05 kmpl', '23.3 kmpl', '11.9
kmpl',
       '13.38 kmpl', '20.86 kmpl', '19.2 km/kg', '10.9 kmpl',
       '18.25 kmpl', '15.2 kmpl', '20.37 kmpl', '17.8 km/kg', '21.8
kmpl',
       '11.96 kmpl', '24.04 kmpl', '19.69 kmpl', '13.73 kmpl',
       '21.04 kmpl', '25.01 kmpl', '10.93 kmpl', '10.9 km/kg',
       '24.29 kmpl', '13.44 kmpl', '20.07 kmpl', '21.1 km/kg',
       '19.08 kmpl', '20.34 kmpl', '11.68 kmpl', '12.5 kmpl', '12.3
kmpl',
       '23.87 kmpl', '16.38 kmpl', '17.42 kmpl', '10.0 kmpl',
       '18.24 kmpl', '10.71 kmpl', '19.59 kmpl', '16.7 kmpl',
       '19.83 kmpl', '21.76 kmpl', '16.05 kmpl', '20.28 kmpl',
       '16.25 kmpl', '16.73 kmpl', '18.48 kmpl', '13.2 km/kg',
       '21.4 km/kg', '14.99 kmpl', '18.76 kmpl', '16.4 kmpl',
       '19.64 kmpl', '14.94 kmpl', '16.6 km/kg', '16.0 km/kg',
       '17.11 kmpl', '22.8 km/kg', '32.26 km/kg', '33.0 km/kg',
       '12.4 kmpl', '18.44 kmpl', '16.09 kmpl', '19.0 km/kg',
       '12.62 kmpl', '21.13 kmpl', '15.17 kmpl', '21.73 kmpl',
       '21.72 kmpl', '12.85 kmpl', '14.81 kmpl', '13.24 kmpl',
       '14.4 km/kg', '21.49 kmpl', '14.62 kmpl', '26.83 km/kg',
       '11.45 kmpl', '12.08 kmpl', '15.74 kmpl', '11.3 kmpl',
       '15.1 km/kg', '14.21 kmpl', '11.72 kmpl', '16.51 kmpl'],
      dtype=object)
```

```
z['mileage'] = pd.to_numeric(z['mileage'].str.replace('kmpl',
'').str.replace('kg', ''), errors='coerce')
```

```
z['mileage'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 8128 entries, 0 to 8127
Series name: mileage
Non-Null Count  Dtype
--------------  -----
7819 non-null   float64
dtypes: float64(1)
memory usage: 63.6 KB
```

```
z
```

```
                              name  year  selling_price  km_driven
fuel  \
0           Maruti Swift Dzire VDI  2014         450000     145500
Diesel
1     Skoda Rapid 1.5 TDI Ambition  2014         370000     120000
Diesel
2          Honda City 2017-2020 EXi  2006         158000     140000
Petrol
3        Hyundai i20 Sportz Diesel  2010         225000     127000
Diesel
4           Maruti Swift VXI BSIII  2007         130000     120000
Petrol
...                            ...   ...            ...        ...
...
8123             Hyundai i20 Magna  2013         320000     110000
Petrol
8124         Hyundai Verna CRDi SX  2007         135000     119000
Diesel
8125        Maruti Swift Dzire ZDi  2009         382000     120000
Diesel
8126               Tata Indigo CR4  2013         290000      25000
Diesel
8127               Tata Indigo CR4  2013         290000      25000
Diesel

     seller_type transmission                 owner  mileage   engine
\
0     Individual       Manual           First Owner    23.40  1248 CC

1     Individual       Manual          Second Owner    21.14  1498 CC

2     Individual       Manual           Third Owner    17.70  1497 CC

3     Individual       Manual           First Owner    23.00  1396 CC

4     Individual       Manual           First Owner    16.10  1298 CC

...          ...          ...                   ...      ...      ...

8123  Individual       Manual           First Owner    18.50  1197 CC

8124  Individual       Manual  Fourth & Above Owner    16.80  1493 CC

8125  Individual       Manual           First Owner    19.30  1248 CC

8126  Individual       Manual           First Owner    23.57  1396 CC

8127  Individual       Manual           First Owner    23.57  1396 CC


       max_power                      torque  seats
```

```
0          74 bhp              190Nm@ 2000rpm    5.0
1     103.52 bhp        250Nm@ 1500-2500rpm    5.0
2          78 bhp      12.7@ 2,700(kgm@ rpm)    5.0
3          90 bhp   22.4 kgm at 1750-2750rpm    5.0
4       88.2 bhp      11.5@ 4,500(kgm@ rpm)    5.0
...           ...                         ...    ...
8123   82.85 bhp           113.7Nm@ 4000rpm    5.0
8124     110 bhp  24@ 1,900-2,750(kgm@ rpm)    5.0
8125    73.9 bhp              190Nm@ 2000rpm    5.0
8126      70 bhp        140Nm@ 1800-3000rpm    5.0
8127      70 bhp        140Nm@ 1800-3000rpm    5.0

[8128 rows x 13 columns]
```

```
z['torque'] =
pd.to_numeric(z['torque'].str.split().str[0].str.replace('Nm@', ''),
errors='coerce')

z['torque'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 8128 entries, 0 to 8127
Series name: torque
Non-Null Count  Dtype
--------------  -----
7151 non-null   float64
dtypes: float64(1)
memory usage: 63.6 KB
```

```
z['engine'] = z['engine'].str.replace('CC', '').astype(float)

z['max_power'] = pd.to_numeric(z['max_power'].str.replace('bhp',
'').str.strip(), errors='coerce')

z['engine'].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 8128 entries, 0 to 8127
Series name: engine
Non-Null Count  Dtype
--------------  -----
7907 non-null   float64
dtypes: float64(1)
memory usage: 63.6 KB

z['max_power'].info()

<class 'pandas.core.series.Series'>
RangeIndex: 8128 entries, 0 to 8127
Series name: max_power
Non-Null Count  Dtype
```

```
--------------  -----
7912 non-null   float64
dtypes: float64(1)
memory usage: 63.6 KB

z

                               name  year  selling_price  km_driven
fuel  \
0           Maruti Swift Dzire VDI  2014         450000     145500
Diesel
1     Skoda Rapid 1.5 TDI Ambition  2014         370000     120000
Diesel
2          Honda City 2017-2020 EXi  2006         158000     140000
Petrol
3        Hyundai i20 Sportz Diesel  2010         225000     127000
Diesel
4           Maruti Swift VXI BSIII  2007         130000     120000
Petrol
...                            ...   ...            ...        ...
...
8123             Hyundai i20 Magna  2013         320000     110000
Petrol
8124         Hyundai Verna CRDi SX  2007         135000     119000
Diesel
8125        Maruti Swift Dzire ZDi  2009         382000     120000
Diesel
8126               Tata Indigo CR4  2013         290000      25000
Diesel
8127               Tata Indigo CR4  2013         290000      25000
Diesel

     seller_type transmission                 owner  mileage
engine  \
0     Individual       Manual           First Owner    23.40  1248.0

1     Individual       Manual          Second Owner    21.14  1498.0

2     Individual       Manual           Third Owner    17.70  1497.0

3     Individual       Manual           First Owner    23.00  1396.0

4     Individual       Manual           First Owner    16.10  1298.0

...          ...          ...                   ...      ...     ...

8123  Individual       Manual           First Owner    18.50  1197.0

8124  Individual       Manual  Fourth & Above Owner    16.80  1493.0

8125  Individual       Manual           First Owner    19.30  1248.0
```

```
8126  Individual       Manual           First Owner    23.57  1396.0

8127  Individual       Manual           First Owner    23.57  1396.0
```

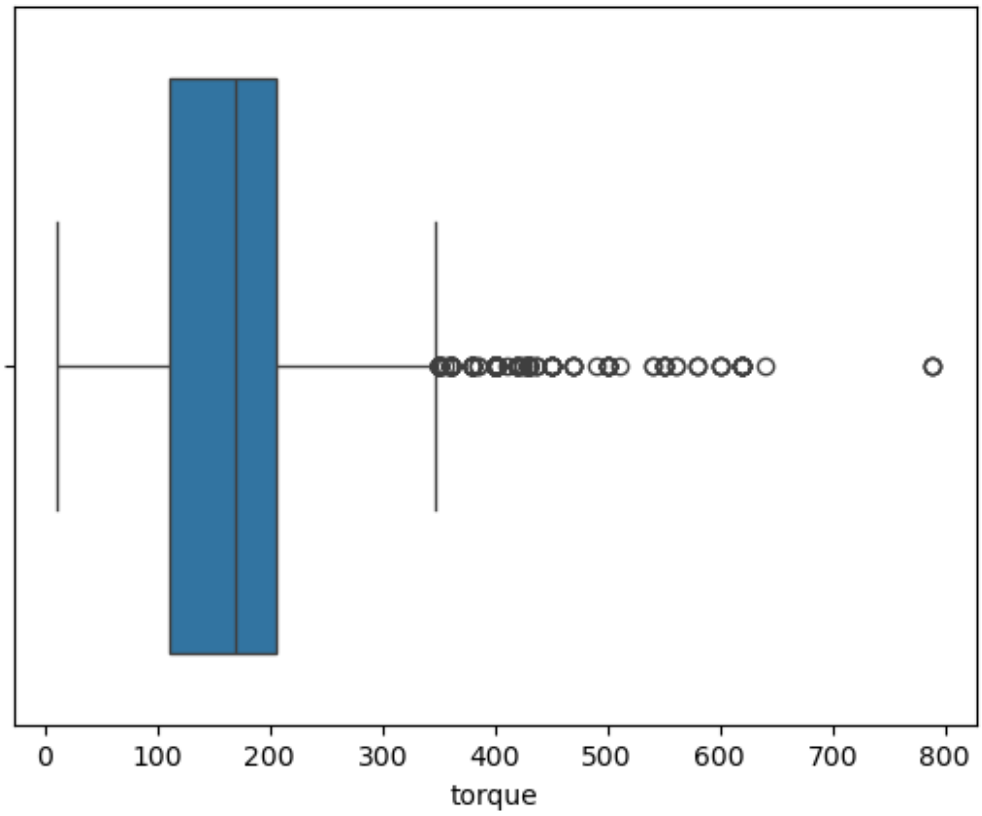
```
      max_power  torque  seats
0         74.00   190.0    5.0
1        103.52   250.0    5.0
2         78.00     NaN    5.0
3         90.00    22.4    5.0
4         88.20     NaN    5.0
...         ...     ...    ...
8123      82.85   113.7    5.0
8124     110.00     NaN    5.0
8125      73.90   190.0    5.0
8126      70.00   140.0    5.0
8127      70.00   140.0    5.0

[8128 rows x 13 columns]
```

```
z.isnull().sum()
```

```
name               0
year               0
selling_price      0
km_driven          0
fuel               0
seller_type        0
transmission       0
owner              0
mileage          309
engine           221
max_power        216
torque           977
seats            221
dtype: int64
```

```
sns.boxplot(x=z['torque'])
```

```
<Axes: xlabel='torque'>
```




```
z['torque'].sort_values(ascending=True)
```

```
226     11.4
2257    11.4
2320    11.4
6992    11.4
3469    11.4
        ...
8104     NaN
8105     NaN
8108     NaN
8113     NaN
8124     NaN
Name: torque, Length: 8128, dtype: float64
```

```
a=z.torque.median()
a
```

```
170.0
```

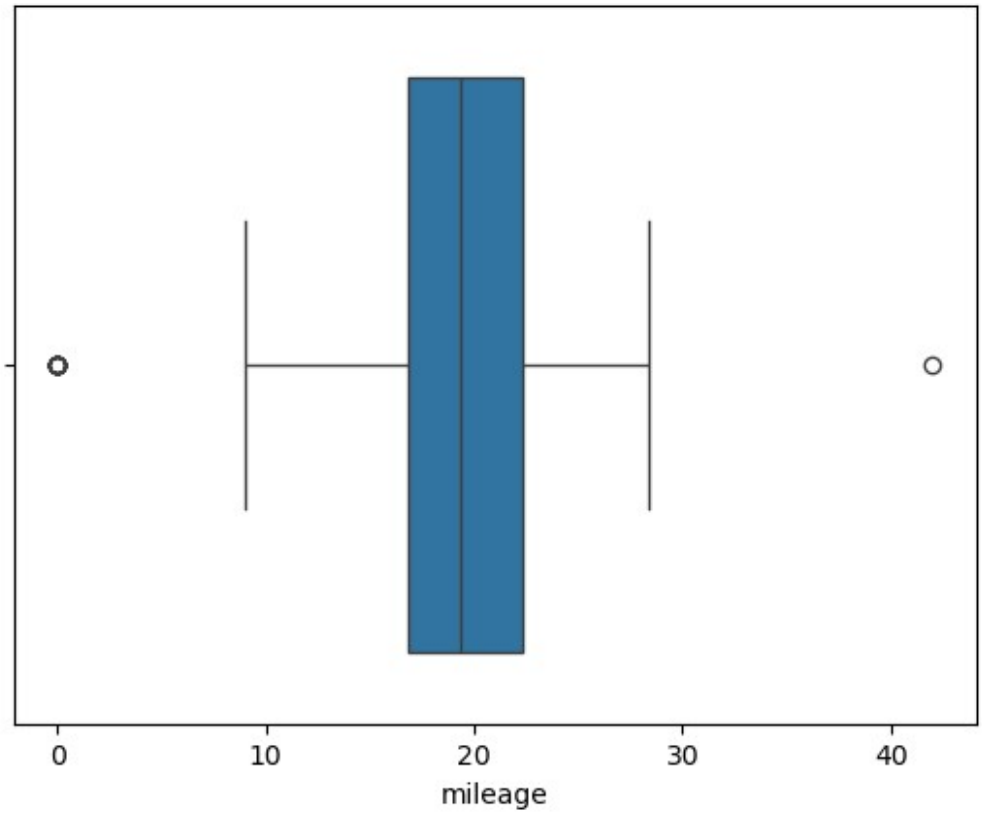
```
z.torque.fillna(a,inplace=True)
z.isnull().sum()
```

```
name                0
year                0
```

```
selling_price      0
km_driven          0
fuel               0
seller_type        0
transmission       0
owner              0
mileage          309
engine           221
max_power        216
torque             0
seats            221
dtype: int64
```

```
sns.boxplot(x=z['mileage'])
```

```
<Axes: xlabel='mileage'>
```




```
z['mileage'].sort_values(ascending=True)
```

```
4527    0.0
2725    0.0
6824    0.0
785     0.0
6629    0.0
```

```
       ...
7913    NaN
7996    NaN
8009    NaN
8068    NaN
8103    NaN
Name: mileage, Length: 8128, dtype: float64
```

```
a=z.mileage.median()
a
```

```
19.3
```

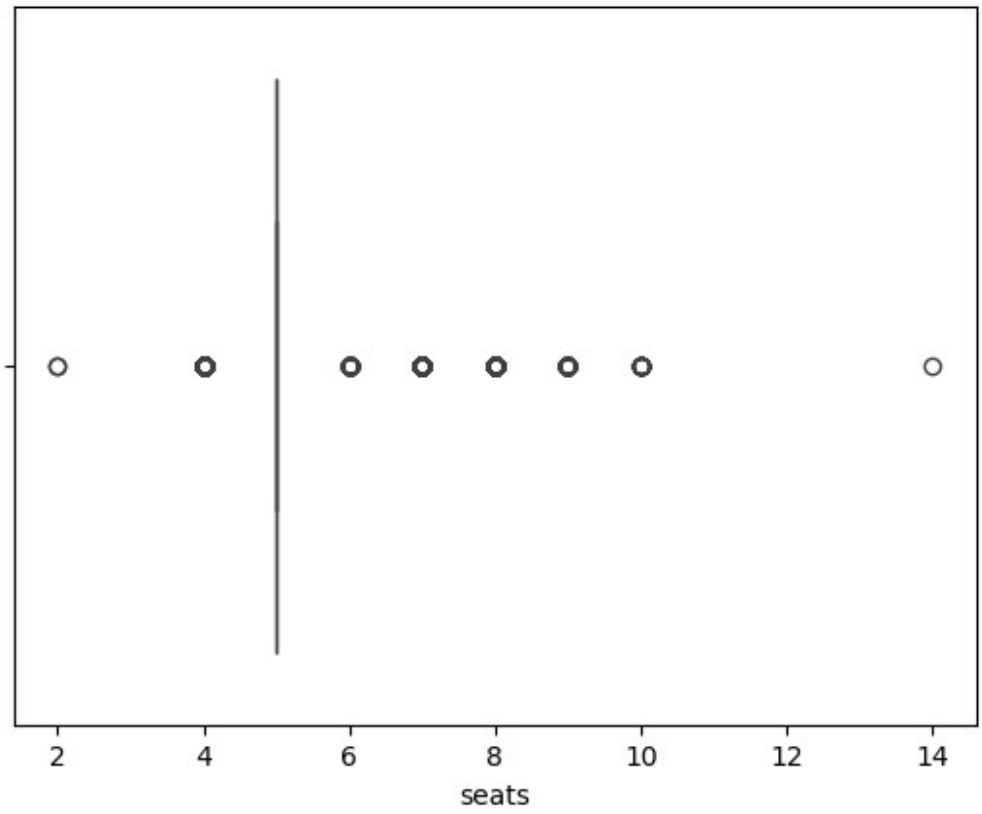
```
z.mileage.fillna(a,inplace=True)
```

```
z.isnull().sum()
```

```
name               0
year               0
selling_price      0
km_driven          0
fuel               0
seller_type        0
transmission       0
owner              0
mileage            0
engine           221
max_power        216
torque             0
seats            221
dtype: int64
```

```
sns.boxplot(x=z['seats'])
```

```
<Axes: xlabel='seats'>
```




```
z['seats'].sort_values(ascending=True)
```

```
5900    2.0
6629    2.0
6161    4.0
7624    4.0
4178    4.0
       ...
7846    NaN
7996    NaN
8009    NaN
8068    NaN
8103    NaN
Name: seats, Length: 8128, dtype: float64
```

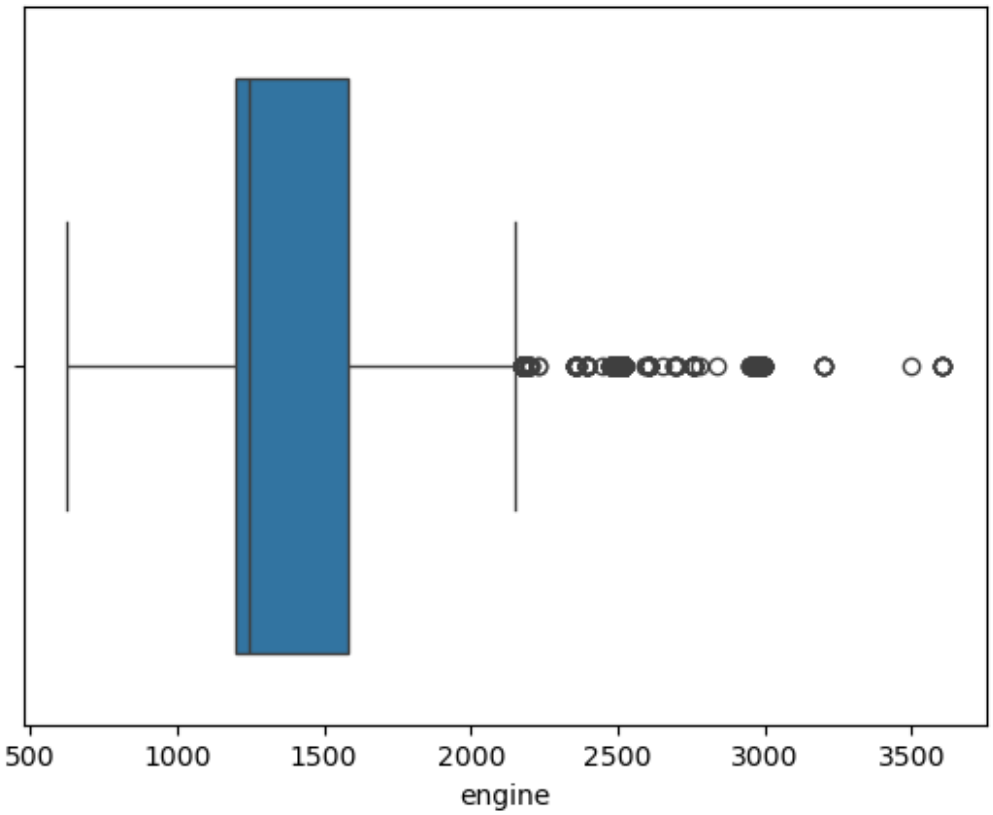
```
a=z.seats.median()
a
```

```
5.0
```

```
z.seats.fillna(a,inplace=True)
```

```
sns.boxplot(x=z['engine'])
```

```
<Axes: xlabel='engine'>
```




```
z['engine'].sort_values(ascending=True)
```

```
6738    624.0
1217    624.0
5878    624.0
3709    624.0
6817    624.0
        ...
7846      NaN
7996      NaN
8009      NaN
8068      NaN
8103      NaN
Name: engine, Length: 8128, dtype: float64
```

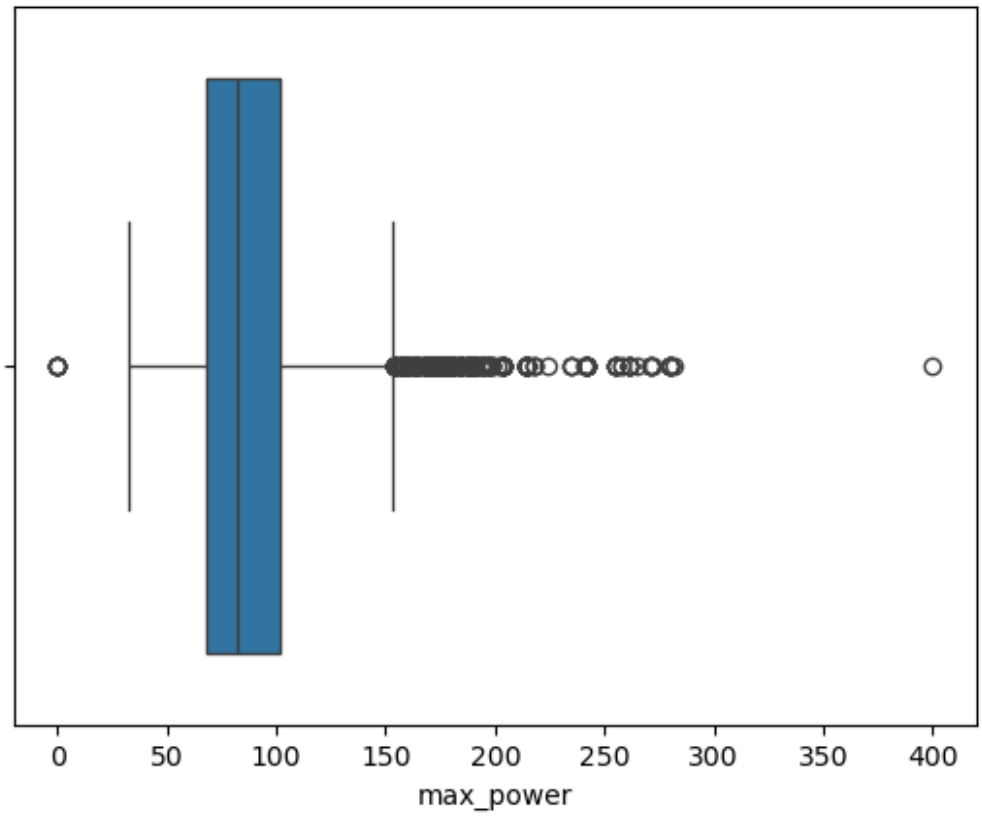
```
a=z.engine.median()
a
```

```
1248.0
```

```
z.engine.fillna(a,inplace=True)
```

```
sns.boxplot(x=z['max_power'])
```

```
<Axes: xlabel='max_power'>
```




```
z['max_power'].sort_values(ascending=True)
```

```
576     0.0
2550    0.0
2549    0.0
575     0.0
1443    0.0
       ...
7846    NaN
7996    NaN
8009    NaN
8068    NaN
8103    NaN
Name: max_power, Length: 8128, dtype: float64
```

```
a=z.max_power.median()
a
```

```
82.0
```

```
z.max_power.fillna(a,inplace=True)
```

```
z.isnull().sum()
```

```
name             0
year             0
selling_price    0
km_driven        0
fuel             0
seller_type      0
transmission     0
owner            0
mileage          0
engine           0
max_power        0
torque           0
seats            0
dtype: int64
```

```
df1=z.select_dtypes(exclude=['object'])
df1
```

```
      year  selling_price  km_driven  mileage  engine  max_power
torque  \
0     2014         450000     145500    23.40  1248.0      74.00
190.0
1     2014         370000     120000    21.14  1498.0     103.52
250.0
2     2006         158000     140000    17.70  1497.0      78.00
170.0
3     2010         225000     127000    23.00  1396.0      90.00
22.4
4     2007         130000     120000    16.10  1298.0      88.20
170.0
...    ...            ...        ...      ...     ...        ...     .
..
8123  2013         320000     110000    18.50  1197.0      82.85
113.7
8124  2007         135000     119000    16.80  1493.0     110.00
170.0
8125  2009         382000     120000    19.30  1248.0      73.90
190.0
8126  2013         290000      25000    23.57  1396.0      70.00
140.0
8127  2013         290000      25000    23.57  1396.0      70.00
140.0

      seats
0       5.0
1       5.0
2       5.0
3       5.0
4       5.0
...     ...
```

```
8123    5.0
8124    5.0
8125    5.0
8126    5.0
8127    5.0

[8128 rows x 8 columns]
```

```
q1=df1.quantile(0.25)
q3=df1.quantile(0.75)
q1
```

```
year               2011.0
selling_price    254999.0
km_driven         35000.0
mileage              16.8
engine             1197.0
max_power            68.1
torque              113.0
seats                 5.0
Name: 0.25, dtype: float64
```

```
q3
```

```
year               2017.00
selling_price    675000.00
km_driven         98000.00
mileage              22.07
engine             1582.00
max_power           101.25
torque              200.00
seats                 5.00
Name: 0.75, dtype: float64
```

```
iqr=q3-q1
iqr
```

```
year                  6.00
selling_price    420001.00
km_driven         63000.00
mileage               5.27
engine              385.00
max_power            33.15
torque               87.00
seats                 0.00
dtype: float64
```

```
b=(df1<(q1-1.5*iqr))|(df1>(q3+1.5*iqr))
b
```

```
       year  selling_price  km_driven  mileage  engine  max_power
torque  \
0     False          False      False    False   False      False
False
1     False          False      False    False   False      False
False
2     False          False      False    False   False      False
False
3     False          False      False    False   False      False
False
4     False          False      False    False   False      False
False
...     ...            ...        ...      ...     ...        ...
...
8123  False          False      False    False   False      False
False
8124  False          False      False    False   False      False
False
8125  False          False      False    False   False      False
False
8126  False          False      False    False   False      False
False
8127  False          False      False    False   False      False
False

      seats
0     False
1     False
2     False
3     False
4     False
...     ...
8123  False
8124  False
8125  False
8126  False
8127  False

[8128 rows x 8 columns]
```

```
df=z[~(b).any(axis=1)]
df
```

```
                               name  year  selling_price  km_driven
fuel  \
0           Maruti Swift Dzire VDI  2014         450000     145500
Diesel
1     Skoda Rapid 1.5 TDI Ambition  2014         370000     120000
Diesel
2         Honda City 2017-2020 EXi  2006         158000     140000
```

```
Petrol
3        Hyundai i20 Sportz Diesel  2010         225000     127000
Diesel
4           Maruti Swift VXI BSIII  2007         130000     120000
Petrol
...                            ...   ...            ...        ...
...
8123             Hyundai i20 Magna  2013         320000     110000
Petrol
8124         Hyundai Verna CRDi SX  2007         135000     119000
Diesel
8125        Maruti Swift Dzire ZDi  2009         382000     120000
Diesel
8126               Tata Indigo CR4  2013         290000      25000
Diesel
8127               Tata Indigo CR4  2013         290000      25000
Diesel

     seller_type transmission                 owner  mileage
engine  \
0     Individual       Manual           First Owner    23.40  1248.0

1     Individual       Manual          Second Owner    21.14  1498.0

2     Individual       Manual           Third Owner    17.70  1497.0

3     Individual       Manual           First Owner    23.00  1396.0

4     Individual       Manual           First Owner    16.10  1298.0

...          ...          ...                   ...      ...     ...

8123  Individual       Manual           First Owner    18.50  1197.0

8124  Individual       Manual  Fourth & Above Owner    16.80  1493.0

8125  Individual       Manual           First Owner    19.30  1248.0

8126  Individual       Manual           First Owner    23.57  1396.0

8127  Individual       Manual           First Owner    23.57  1396.0


      max_power  torque  seats
0         74.00   190.0    5.0
1        103.52   250.0    5.0
2         78.00   170.0    5.0
3         90.00    22.4    5.0
4         88.20   170.0    5.0
...         ...     ...    ...
```

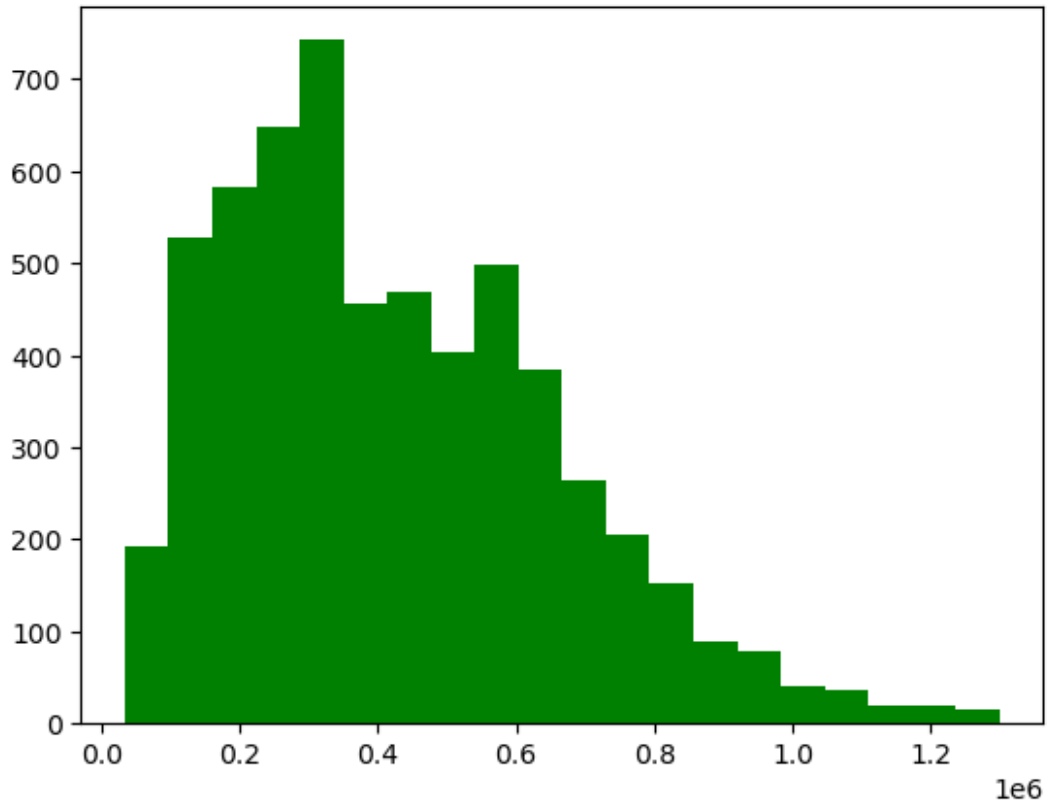
```
8123      82.85   113.7    5.0
8124     110.00   170.0    5.0
8125      73.90   190.0    5.0
8126      70.00   140.0    5.0
8127      70.00   140.0    5.0

[5826 rows x 13 columns]
```

```
z.shape,df.shape
```

```
((8128, 13), (5826, 13))
```

```
plt.hist(df['selling_price'],bins=20,color='green')
```

```
(array([192., 528., 583., 648., 743., 457., 469., 403., 499., 384.,
264.,
        206., 153.,  89.,  78.,  40.,  36.,  20.,  20.,  14.]),
 array([  33351.  ,   96683.45,  160015.9 ,  223348.35,  286680.8 ,
         350013.25,  413345.7 ,  476678.15,  540010.6 ,  603343.05,
         666675.5 ,  730007.95,  793340.4 ,  856672.85,  920005.3 ,
         983337.75, 1046670.2 , 1110002.65, 1173335.1 , 1236667.55,
        1300000.  ]),
 <BarContainer object of 20 artists>)
```



```
df.groupby(['owner']).count()
```

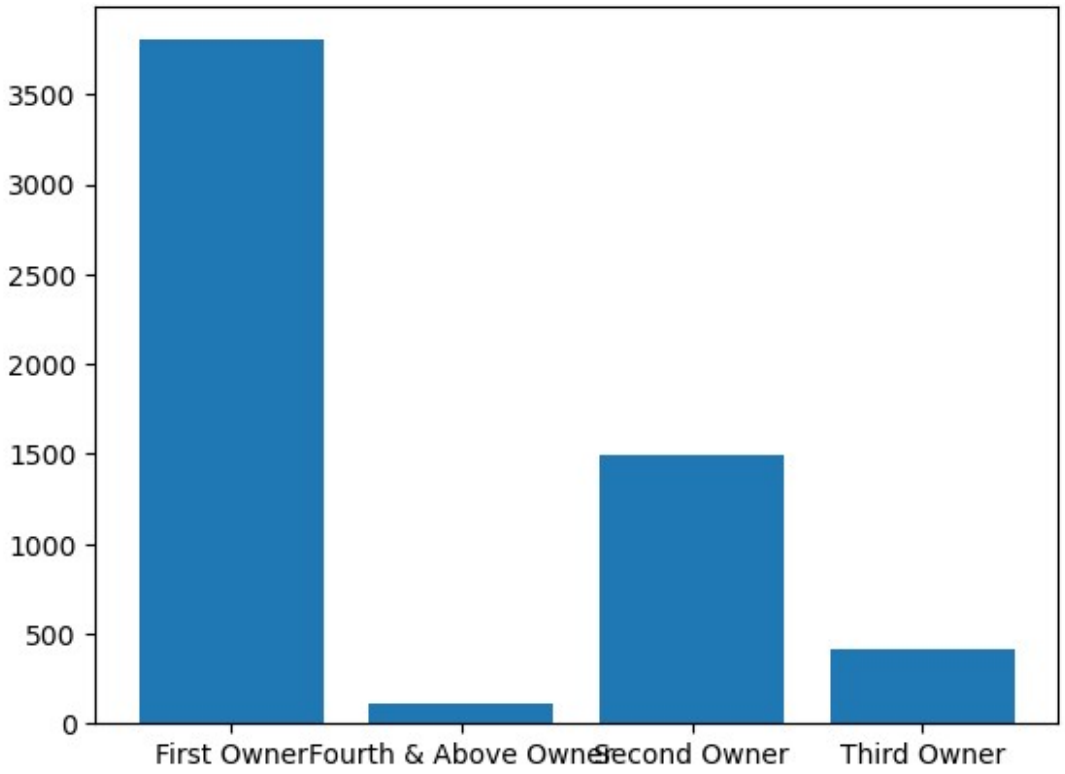
```
                      name  year  selling_price  km_driven  fuel
seller_type  \
owner

First Owner           3802  3802           3802       3802  3802
3802
Fourth & Above Owner   114   114            114        114   114
114
Second Owner          1496  1496           1496       1496  1496
1496
Third Owner            414   414            414        414   414
414

                      transmission  mileage  engine  max_power  torque
seats
owner

First Owner                   3802     3802    3802       3802    3802
3802
Fourth & Above Owner           114      114     114        114     114
114
Second Owner                  1496     1496    1496       1496    1496
1496
Third Owner                    414      414     414        414     414
414
```

```
a=df.groupby(['owner']).size().reset_index(name='count').rename(column
s={'owner':'Owner'})
a
```

```
                  Owner  count
0           First Owner   3802
1  Fourth & Above Owner    114
2          Second Owner   1496
3           Third Owner    414
```

```
plt.bar(a['Owner'],a['count'])
```

```
<BarContainer object of 4 artists>
```

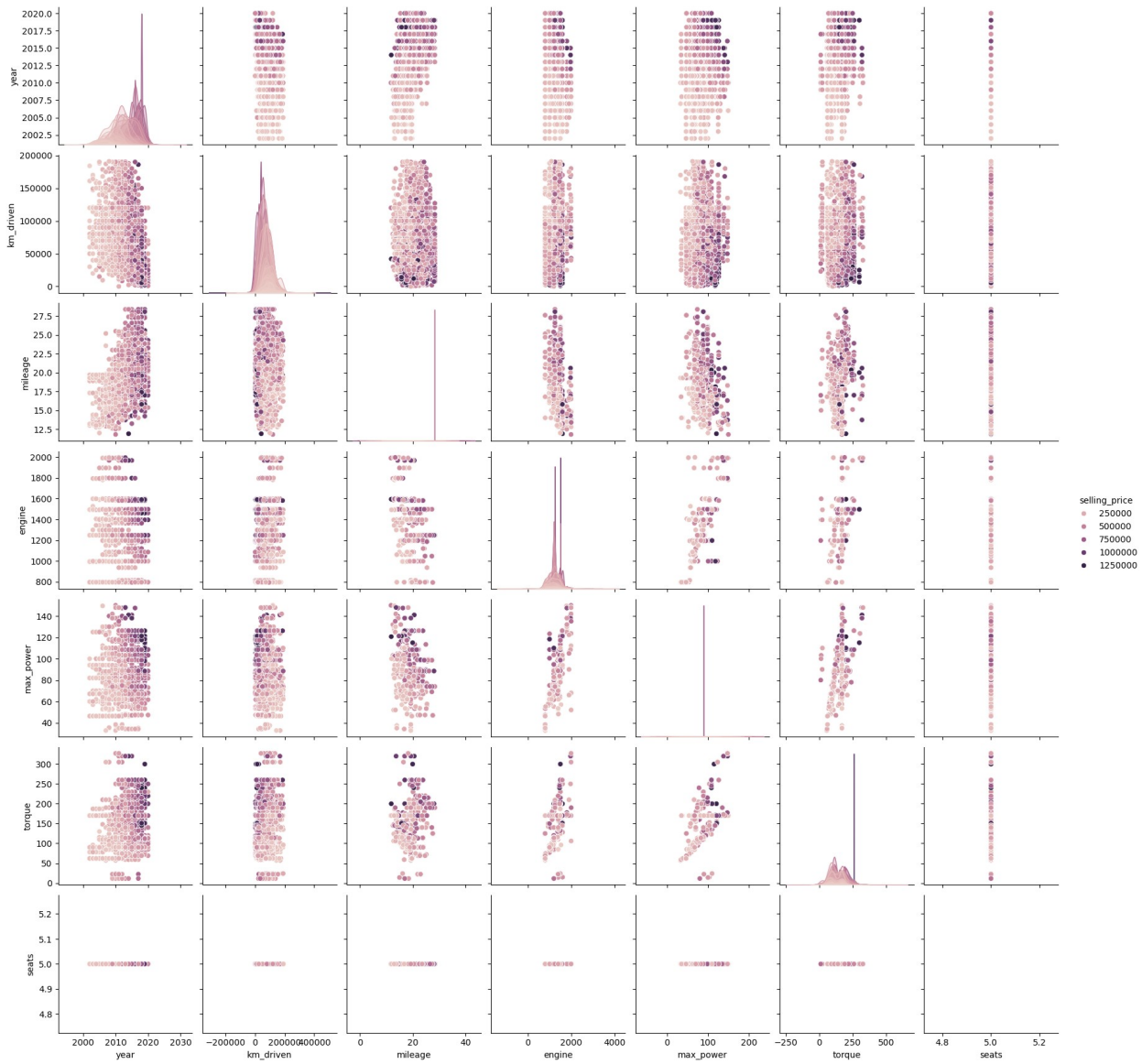



```
a['Percentage']=a['count']/sum(a['count'])*100
a
```

	Owner	count	Percentage
0	First Owner	3802	65.259183
1	Fourth & Above Owner	114	1.956746
2	Second Owner	1496	25.677995
3	Third Owner	414	7.106076

```
sns.pairplot(df,hue='selling_price')
```

```
<seaborn.axisgrid.PairGrid at 0x1909cb8ae10>
```




```
df1=df.select_dtypes(exclude=['object'])

df1
```

	year	selling_price	km_driven	mileage	engine	max_power	torque
0	2014	450000	145500	23.40	1248.0	74.00	190.0
1	2014	370000	120000	21.14	1498.0	103.52	250.0
2	2006	158000	140000	17.70	1497.0	78.00	170.0
3	2010	225000	127000	23.00	1396.0	90.00	22.4
4	2007	130000	120000	16.10	1298.0	88.20	

```
170.0
...    ...            ...        ...      ...     ...         ...     .
..
8123  2013         320000     110000    18.50  1197.0      82.85
113.7
8124  2007         135000     119000    16.80  1493.0     110.00
170.0
8125  2009         382000     120000    19.30  1248.0      73.90
190.0
8126  2013         290000      25000    23.57  1396.0      70.00
140.0
8127  2013         290000      25000    23.57  1396.0      70.00
140.0

      seats
0       5.0
1       5.0
2       5.0
3       5.0
4       5.0
...     ...
8123    5.0
8124    5.0
8125    5.0
8126    5.0
8127    5.0

[5826 rows x 8 columns]
```

```
df1=df1.drop(columns='seats')
```

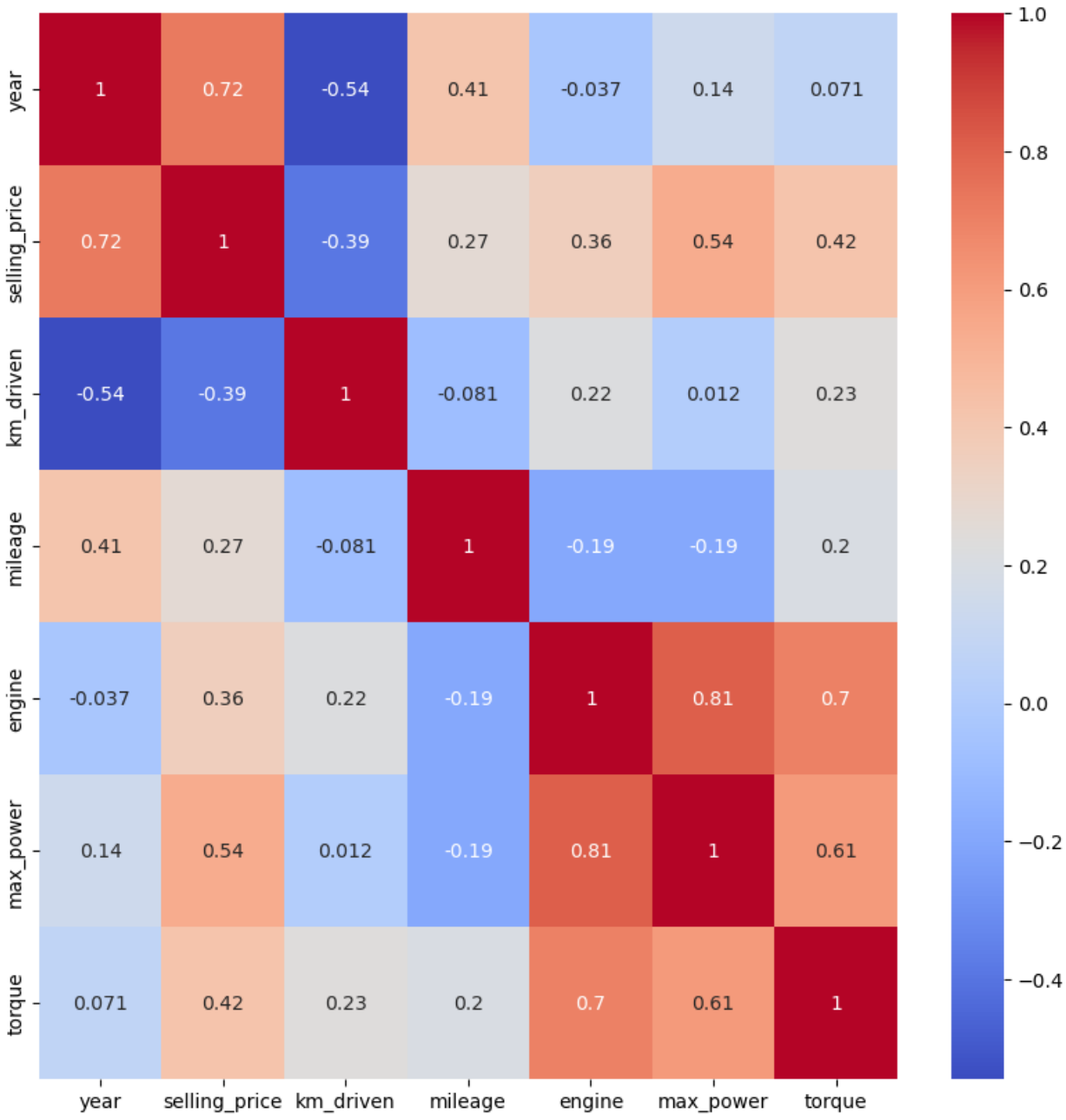
```
df1.corr()
```

```
                   year  selling_price  km_driven   mileage    engine
\
year           1.000000       0.723499  -0.544712  0.414836 -0.036754

selling_price  0.723499       1.000000  -0.389722  0.269387  0.359889

km_driven     -0.544712      -0.389722   1.000000 -0.080741  0.220696

mileage        0.414836       0.269387  -0.080741  1.000000 -0.190526

engine        -0.036754       0.359889   0.220696 -0.190526  1.000000

max_power      0.137883       0.537292   0.011957 -0.193613  0.812045

torque         0.070762       0.423824   0.234183  0.201636  0.698993


               max_power    torque
```

```
year             0.137883  0.070762
selling_price    0.537292  0.423824
km_driven        0.011957  0.234183
mileage         -0.193613  0.201636
engine           0.812045  0.698993
max_power        1.000000  0.608684
torque           0.608684  1.000000
```

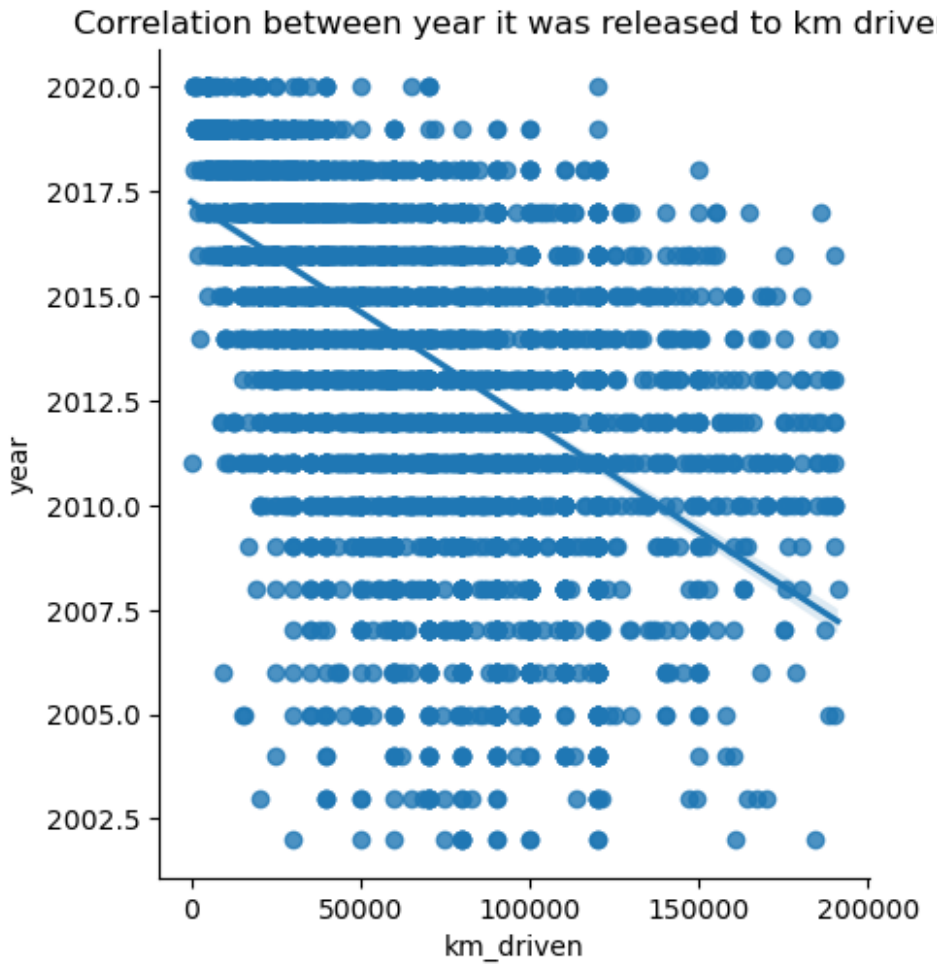
```
plt.figure(figsize=(10,10))
sns.heatmap(df1.corr(),annot=True,cmap='coolwarm')
```

```
<Axes: >
```



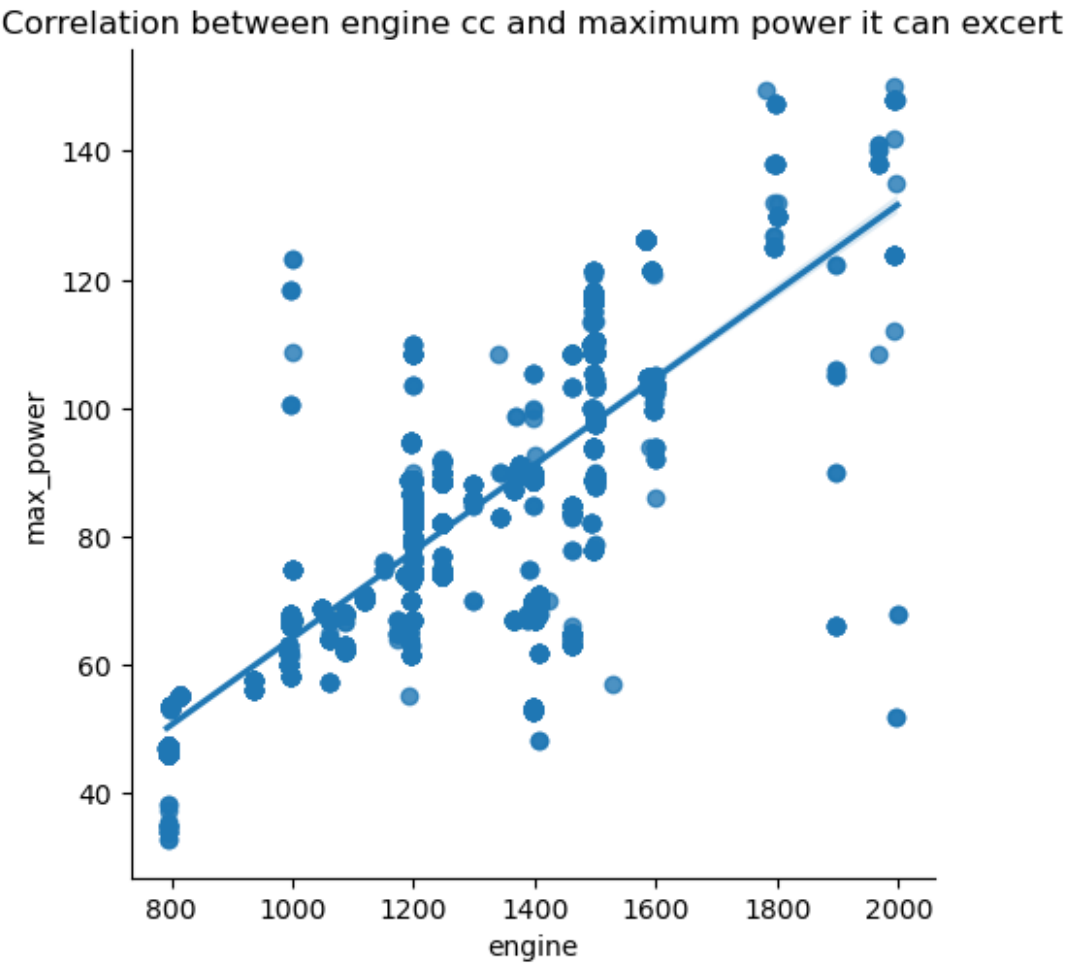

```
sns.lmplot(x='km_driven',y='year',data=df1)
plt.title('Correlation between year it was released to km driven')
```

```
Text(0.5, 1.0, 'Correlation between year it was released to km
driven')
```




```
sns.lmplot(x='engine',y='max_power',data=df1)
plt.title('Correlation between engine cc and maximum power it can
excert')
```

```
Text(0.5, 1.0, 'Correlation between engine cc and maximum power it can
excert')
```




```
df1
```

	year	selling_price	km_driven	mileage	engine	max_power	torque
0	2014	450000	145500	23.40	1248.0	74.00	190.0
1	2014	370000	120000	21.14	1498.0	103.52	250.0
2	2006	158000	140000	17.70	1497.0	78.00	170.0
3	2010	225000	127000	23.00	1396.0	90.00	22.4
4	2007	130000	120000	16.10	1298.0	88.20	170.0
...	...	...	...	...	...	...	...
8123	2013	320000	110000	18.50	1197.0	82.85	113.7
8124	2007	135000	119000	16.80	1493.0	110.00	170.0

```
8125  2009         382000     120000    19.30  1248.0      73.90
190.0
8126  2013         290000      25000    23.57  1396.0      70.00
140.0
8127  2013         290000      25000    23.57  1396.0      70.00
140.0

[5826 rows x 7 columns]
```

```
x=df1.drop(columns=['year','selling_price'])
y=df1['selling_price']
x
```

```
      km_driven  mileage  engine  max_power  torque
0        145500    23.40  1248.0      74.00   190.0
1        120000    21.14  1498.0     103.52   250.0
2        140000    17.70  1497.0      78.00   170.0
3        127000    23.00  1396.0      90.00    22.4
4        120000    16.10  1298.0      88.20   170.0
...         ...      ...     ...        ...     ...
8123     110000    18.50  1197.0      82.85   113.7
8124     119000    16.80  1493.0     110.00   170.0
8125     120000    19.30  1248.0      73.90   190.0
8126      25000    23.57  1396.0      70.00   140.0
8127      25000    23.57  1396.0      70.00   140.0

[5826 rows x 5 columns]
```

```
y
```

```
0       450000
1       370000
2       158000
3       225000
4       130000
        ...
8123    320000
8124    135000
8125    382000
8126    290000
8127    290000
Name: selling_price, Length: 5826, dtype: int64
```

```
from sklearn.feature_selection import f_classif
a=f_classif(x,y)
a
```

```
(array([3.82887898, 4.19194553, 3.21322149, 6.63005358, 4.13584628]),
 array([1.47843899e-127, 1.90094056e-148, 1.18314030e-092,
2.23819399e-287,
        3.26645125e-145]))
```

```
a=pd.Series(a[1])
a.index=x.columns
a

km_driven    1.478439e-127
mileage      1.900941e-148
engine        1.183140e-92
max_power    2.238194e-287
torque       3.266451e-145
dtype: float64




df

                              name  year  selling_price  km_driven
fuel  \
0           Maruti Swift Dzire VDI  2014         450000     145500
Diesel
1     Skoda Rapid 1.5 TDI Ambition  2014         370000     120000
Diesel
2          Honda City 2017-2020 EXi  2006         158000     140000
Petrol
3        Hyundai i20 Sportz Diesel  2010         225000     127000
Diesel
4           Maruti Swift VXI BSIII  2007         130000     120000
Petrol
...                            ...   ...            ...        ...
...
8123              Hyundai i20 Magna  2013         320000     110000
Petrol
8124          Hyundai Verna CRDi SX  2007         135000     119000
Diesel
8125        Maruti Swift Dzire ZDi  2009         382000     120000
Diesel
8126                Tata Indigo CR4  2013         290000      25000
Diesel
8127                Tata Indigo CR4  2013         290000      25000
Diesel

     seller_type transmission                 owner  mileage
engine  \
0     Individual       Manual           First Owner    23.40  1248.0

1     Individual       Manual          Second Owner    21.14  1498.0

2     Individual       Manual           Third Owner    17.70  1497.0

3     Individual       Manual           First Owner    23.00  1396.0
```

```
4     Individual       Manual           First Owner    16.10  1298.0

...          ...          ...                   ...      ...     ...

8123  Individual       Manual           First Owner    18.50  1197.0

8124  Individual       Manual  Fourth & Above Owner    16.80  1493.0

8125  Individual       Manual           First Owner    19.30  1248.0

8126  Individual       Manual           First Owner    23.57  1396.0

8127  Individual       Manual           First Owner    23.57  1396.0


      max_power  torque  seats
0         74.00   190.0    5.0
1        103.52   250.0    5.0
2         78.00   170.0    5.0
3         90.00    22.4    5.0
4         88.20   170.0    5.0
...         ...     ...    ...
8123      82.85   113.7    5.0
8124     110.00   170.0    5.0
8125      73.90   190.0    5.0
8126      70.00   140.0    5.0
8127      70.00   140.0    5.0

[5826 rows x 13 columns]

df.name.unique()

array(['Maruti Swift Dzire VDI', 'Skoda Rapid 1.5 TDI Ambition',
       'Honda City 2017-2020 EXi', ...,
       'Tata Manza Aura (ABS) Safire BS IV', 'Tata Nexon 1.5 Revotorq
XT',
       'Ford Freestyle Titanium Plus Diesel BSIV'], dtype=object)

df.fuel.unique()

array(['Diesel', 'Petrol', 'LPG', 'CNG'], dtype=object)

df.seller_type.unique()

array(['Individual', 'Dealer', 'Trustmark Dealer'], dtype=object)

df.transmission.unique()

array(['Manual', 'Automatic'], dtype=object)

df.owner.unique()
```

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

```
df.seats.unique()
```

```
array([5.])
```

```
from sklearn.preprocessing import LabelEncoder
l=LabelEncoder()
```

```
df['fuel']=l.fit_transform(df['fuel'])
df['seller_type']=l.fit_transform(df['seller_type'])
df['transmission']=l.fit_transform(df['transmission'])
df['owner']=l.fit_transform(df['owner'])
```

```
C:\Users\Praveen Kumar\AppData\Local\Temp\
ipykernel_2620\643564686.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df['fuel']=l.fit_transform(df['fuel'])
C:\Users\Praveen Kumar\AppData\Local\Temp\
ipykernel_2620\643564686.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df['seller_type']=l.fit_transform(df['seller_type'])
C:\Users\Praveen Kumar\AppData\Local\Temp\
ipykernel_2620\643564686.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df['transmission']=l.fit_transform(df['transmission'])
C:\Users\Praveen Kumar\AppData\Local\Temp\
ipykernel_2620\643564686.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df['owner']=l.fit_transform(df['owner'])
```

```
df

                              name  year  selling_price  km_driven
fuel  \
0            Maruti Swift Dzire VDI  2014         450000     145500
1
1     Skoda Rapid 1.5 TDI Ambition  2014         370000     120000
1
2          Honda City 2017-2020 EXi  2006         158000     140000
3
3        Hyundai i20 Sportz Diesel  2010         225000     127000
1
4            Maruti Swift VXI BSIII  2007         130000     120000
3
...                             ...   ...            ...        ...   .
..
8123             Hyundai i20 Magna  2013         320000     110000
3
8124          Hyundai Verna CRDi SX  2007         135000     119000
1
8125        Maruti Swift Dzire ZDi  2009         382000     120000
1
8126                Tata Indigo CR4  2013         290000      25000
1
8127                Tata Indigo CR4  2013         290000      25000
1

      seller_type  transmission  owner  mileage  engine  max_power
torque  \
0               1             1      0    23.40  1248.0      74.00
190.0
1               1             1      2    21.14  1498.0     103.52
250.0
2               1             1      3    17.70  1497.0      78.00
170.0
3               1             1      0    23.00  1396.0      90.00
22.4
4               1             1      0    16.10  1298.0      88.20
170.0
...           ...           ...    ...      ...     ...        ...
...
8123            1             1      0    18.50  1197.0      82.85
113.7
8124            1             1      1    16.80  1493.0     110.00
170.0
8125            1             1      0    19.30  1248.0      73.90
190.0
8126            1             1      0    23.57  1396.0      70.00
140.0
8127            1             1      0    23.57  1396.0      70.00
```

```
140.0
```

```
      seats
0       5.0
1       5.0
2       5.0
3       5.0
4       5.0
...     ...
8123    5.0
8124    5.0
8125    5.0
8126    5.0
8127    5.0

[5826 rows x 13 columns]
```

```
df.owner.unique()
```

```
array([0, 2, 3, 1])
```

```
df=df.drop(columns=['name','seats'])
df
```

```
      year  selling_price  km_driven  fuel  seller_type  transmission
owner  \
0     2014         450000     145500     1            1             1
0
1     2014         370000     120000     1            1             1
2
2     2006         158000     140000     3            1             1
3
3     2010         225000     127000     1            1             1
0
4     2007         130000     120000     3            1             1
0
...    ...            ...        ...   ...          ...           ...
...
8123  2013         320000     110000     3            1             1
0
8124  2007         135000     119000     1            1             1
1
8125  2009         382000     120000     1            1             1
0
8126  2013         290000      25000     1            1             1
0
8127  2013         290000      25000     1            1             1
0

      mileage  engine  max_power  torque
```

```
0       23.40  1248.0      74.00   190.0
1       21.14  1498.0     103.52   250.0
2       17.70  1497.0      78.00   170.0
3       23.00  1396.0      90.00    22.4
4       16.10  1298.0      88.20   170.0
...       ...     ...        ...     ...
8123    18.50  1197.0      82.85   113.7
8124    16.80  1493.0     110.00   170.0
8125    19.30  1248.0      73.90   190.0
8126    23.57  1396.0      70.00   140.0
8127    23.57  1396.0      70.00   140.0

[5826 rows x 11 columns]
```

```
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score

x=df.drop(columns=['selling_price'])
y=df['selling_price']
x
```

```
      year  km_driven  fuel  seller_type  transmission  owner  mileage
\
0     2014     145500     1            1             1      0    23.40

1     2014     120000     1            1             1      2    21.14

2     2006     140000     3            1             1      3    17.70

3     2010     127000     1            1             1      0    23.00

4     2007     120000     3            1             1      0    16.10

...    ...        ...   ...          ...           ...    ...      ...

8123  2013     110000     3            1             1      0    18.50

8124  2007     119000     1            1             1      1    16.80

8125  2009     120000     1            1             1      0    19.30

8126  2013      25000     1            1             1      0    23.57

8127  2013      25000     1            1             1      0    23.57


      engine  max_power  torque
0     1248.0      74.00   190.0
1     1498.0     103.52   250.0
2     1497.0      78.00   170.0
```

```
3     1396.0      90.00    22.4
4     1298.0      88.20   170.0
...      ...        ...     ...
8123  1197.0      82.85   113.7
8124  1493.0     110.00   170.0
8125  1248.0      73.90   190.0
8126  1396.0      70.00   140.0
8127  1396.0      70.00   140.0

[5826 rows x 10 columns]

y

0       450000
1       370000
2       158000
3       225000
4       130000
        ...
8123    320000
8124    135000
8125    382000
8126    290000
8127    290000
Name: selling_price, Length: 5826, dtype: int64
```

```
scaler=StandardScaler()
x=scaler.fit_transform(x)

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,rand
om_state=42)

rf=RandomForestRegressor()
param_grid = {
    'n_estimators': [500,1000,2000],
    'max_depth': [15,20,25],
    'min_samples_split': [5,7,8]
}

grid_search=GridSearchCV(estimator=rf,param_grid=param_grid,cv=5,scori
ng='neg_mean_squared_error')

grid_search.fit(x_train,y_train)
```

```
GridSearchCV(cv=5, estimator=RandomForestRegressor(),
             param_grid={'max_depth': [15, 20, 25],
                         'min_samples_split': [5, 7, 8],
                         'n_estimators': [500, 1000, 2000]},
             scoring='neg_mean_squared_error')
```

```
grid_search.best_params_
```

```
{'max_depth': 20, 'min_samples_split': 5, 'n_estimators': 2000}
```

```
grid_search.best_score_
```

```
-5191101754.742777
```

```
pr=grid_search.predict(x_test)
mean_squared_error(y_test,pr) ,r2_score(y_test,pr)
```

```
(5411710755.130806, 0.9053867145742261)
```

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

```
RandomForestRegressor()
```

```
pr=rf.predict(x_test)
mean_squared_error(y_test,pr), r2_score(y_test,pr)
```

```
(5581558013.544531, 0.9024172640869016)
```

```
from sklearn.tree import DecisionTreeRegressor
dt_regressor = DecisionTreeRegressor()
dt_regressor.fit(x_train, y_train)
```

```
DecisionTreeRegressor()
```

```
pr=dt_regressor.predict(x_test)
mean_squared_error(y_test,pr) , r2_score(y_test,pr)
```

```
(9839986128.41687, 0.8279668928589925)
```

```
from sklearn.neighbors import KNeighborsRegressor
knn=KNeighborsRegressor()
```

```
param={'n_neighbors' : [3,5,7,9],
       'weights' : ['uniform','distance'],
       'algorithm' : ['auto','ball_tree']}
```

```
knn1=GridSearchCV(knn,param,cv=5,scoring='neg_mean_squared_error')
knn1.fit(x_train,y_train)
```

```
GridSearchCV(cv=5, estimator=KNeighborsRegressor(),
             param_grid={'algorithm': ['auto', 'ball_tree'],
```

```
                          'n_neighbors': [3, 5, 7, 9],
                          'weights': ['uniform', 'distance']},
             scoring='neg_mean_squared_error')
```

```
knn1.best_params_
```

```
{'algorithm': 'ball_tree', 'n_neighbors': 9, 'weights': 'distance'}
```

```
knn1.best_score_
```

```
-7777108089.970784
```

```
pr=knn1.predict(x_test)
mean_squared_error(y_test,pr) , r2_score(y_test,pr)
```

```
(8096815730.99104, 0.858442852462163)
```