```
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.metrics import mean_squared_error, r2_score

df=pd.read_csv("C:/Users/user/Downloads/Car details v3.csv")

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   23.4 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]
```

```
df.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
```

```
df.isnull().sum()
```

```
name               0
year               0
selling_price      0
km_driven          0
fuel               0
seller_type        0
transmission       0
owner              0
mileage          221
engine           221
max_power        215
torque           222
seats            221
dtype: int64
```

```
df = df.dropna()
```

```
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   23.4 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

[7906 rows x 13 columns]
```

```
df.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
```

```
df['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', '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)
```

```
df.loc[:, 'mileage'] =
pd.to_numeric(df['mileage'].astype(str).str.replace('kmpl',
'').str.replace('kg', ''), errors='coerce')
```

```
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    23.4  1248 CC

1     Individual       Manual          Second Owner   21.14  1498 CC

2     Individual       Manual           Third Owner    17.7  1497 CC

3     Individual       Manual           First Owner    23.0  1396 CC

4     Individual       Manual           First Owner    16.1  1298 CC

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

8123  Individual       Manual           First Owner    18.5  1197 CC

8124  Individual       Manual  Fourth & Above Owner    16.8  1493 CC

8125  Individual       Manual           First Owner    19.3  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

[7906 rows x 13 columns]
```

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

```
<class 'pandas.core.series.Series'>
Index: 7906 entries, 0 to 8127
Series name: mileage
Non-Null Count  Dtype
--------------  -----
7819 non-null   object
dtypes: object(1)
memory usage: 123.5+ KB
```

```
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    23.4  1248 CC

1     Individual       Manual          Second Owner   21.14  1498 CC

2     Individual       Manual           Third Owner    17.7  1497 CC

3     Individual       Manual           First Owner    23.0  1396 CC

4     Individual       Manual           First Owner    16.1  1298 CC

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

8123  Individual       Manual           First Owner    18.5  1197 CC

8124  Individual       Manual  Fourth & Above Owner    16.8  1493 CC

8125  Individual       Manual           First Owner    19.3  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

[7906 rows x 13 columns]
```

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

df.torque.info()
```

```
<class 'pandas.core.series.Series'>
Index: 7906 entries, 0 to 8127
Series name: torque
Non-Null Count  Dtype
--------------  -----
7151 non-null   object
```

```
dtypes: object(1)
memory usage: 123.5+ KB
```

```
df.loc[:, 'max_power'] =
pd.to_numeric(df['max_power'].astype(str).str.replace('bhp',
'').str.strip(), errors='coerce')
```

```
df.max_power.info()
```

```
<class 'pandas.core.series.Series'>
Index: 7906 entries, 0 to 8127
Series name: max_power
Non-Null Count  Dtype
--------------  -----
7906 non-null   object
dtypes: object(1)
memory usage: 123.5+ KB
```

```
df.loc[:, 'engine'] = df['engine'].astype(str).str.replace('CC',
'').astype(float)
```

```
df.engine.info()
```

```
<class 'pandas.core.series.Series'>
Index: 7906 entries, 0 to 8127
Series name: engine
Non-Null Count  Dtype
--------------  -----
7906 non-null   object
dtypes: object(1)
memory usage: 123.5+ KB
```

```
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
```

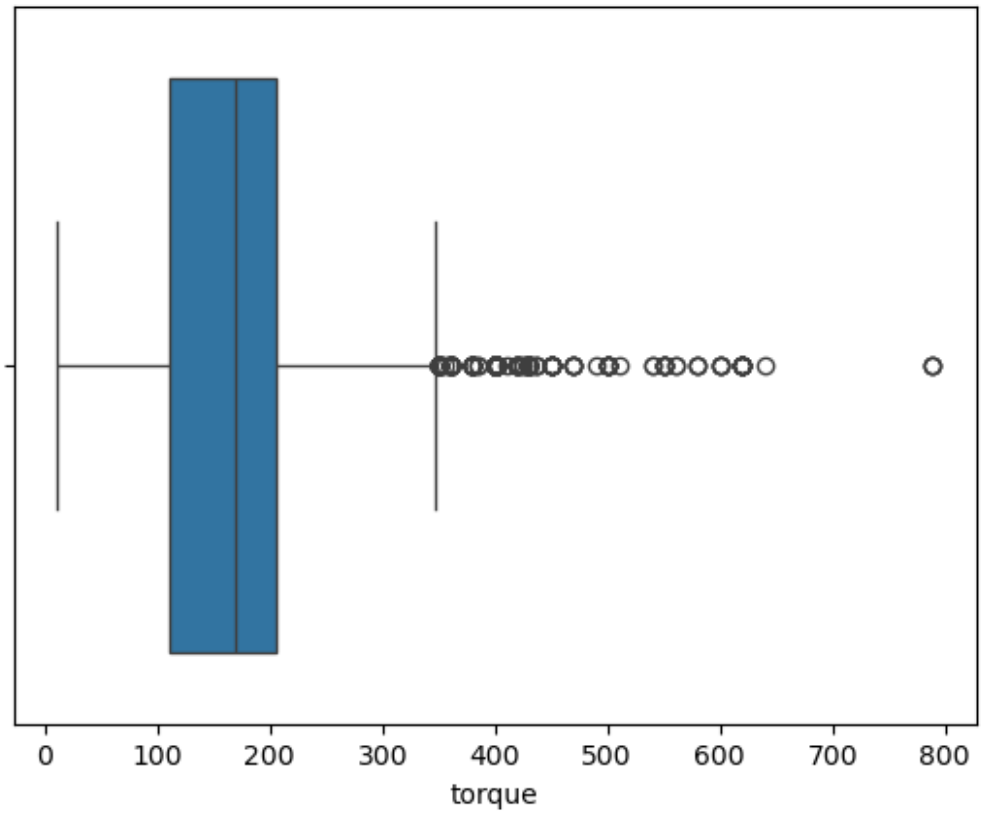
```
Diesel
8125        Maruti Swift Dzire ZDi  2009         382000     120000
Diesel
8126               Tata Indigo CR4  2013         290000      25000
Diesel
8127               Tata Indigo CR4  2013         290000      25000
Diesel

     seller_type transmission                 owner mileage  engine
max_power  \
0     Individual       Manual           First Owner    23.4  1248.0
74.0
1     Individual       Manual          Second Owner   21.14  1498.0
103.52
2     Individual       Manual           Third Owner    17.7  1497.0
78.0
3     Individual       Manual           First Owner    23.0  1396.0
90.0
4     Individual       Manual           First Owner    16.1  1298.0
88.2
...          ...          ...                   ...     ...     ...
...
8123  Individual       Manual           First Owner    18.5  1197.0
82.85
8124  Individual       Manual  Fourth & Above Owner    16.8  1493.0
110.0
8125  Individual       Manual           First Owner    19.3  1248.0
73.9
8126  Individual       Manual           First Owner   23.57  1396.0
70.0
8127  Individual       Manual           First Owner   23.57  1396.0
70.0

     torque  seats
0     190.0    5.0
1     250.0    5.0
2       NaN    5.0
3      22.4    5.0
4       NaN    5.0
...     ...    ...
8123  113.7    5.0
8124    NaN    5.0
8125  190.0    5.0
8126  140.0    5.0
8127  140.0    5.0

[7906 rows x 13 columns]
```

```
df.isnull().sum()
```

```
name                0
year                0
selling_price       0
km_driven           0
fuel                0
seller_type         0
transmission        0
owner               0
mileage            87
engine              0
max_power           0
torque            755
seats               0
dtype: int64
```

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

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




```
df['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: 7906, dtype: object
```

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

```
170.0
```

```
df.torque.fillna(a,inplace=True)
```

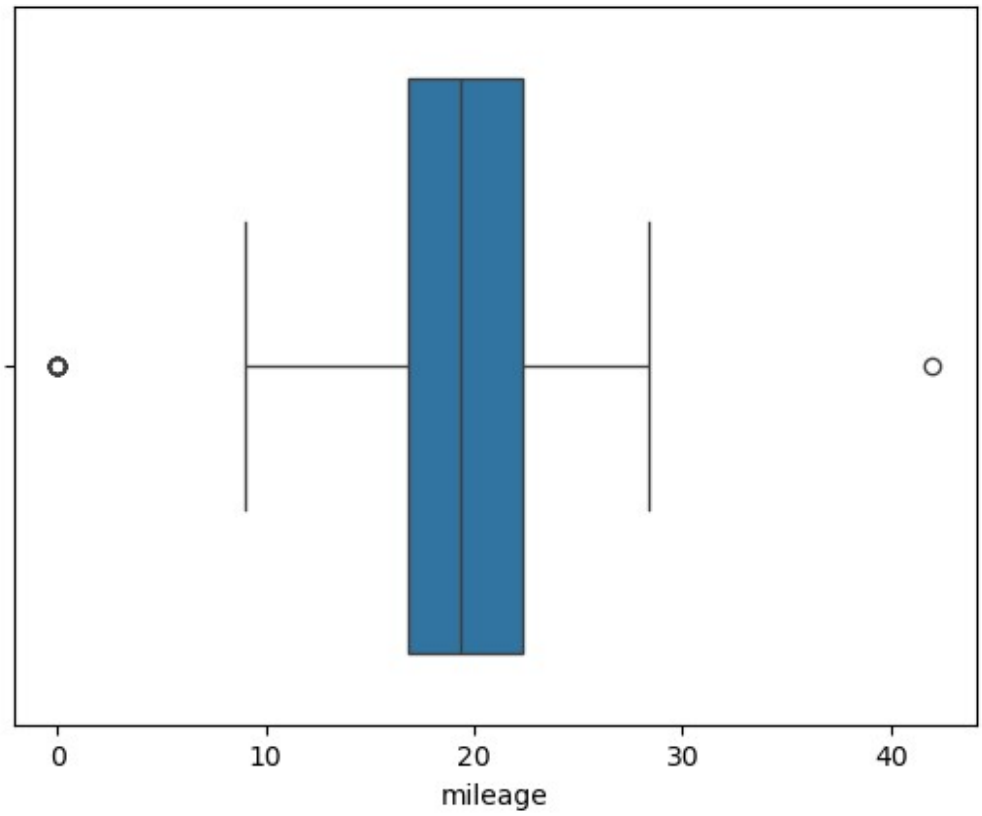
```
C:\Users\user\AppData\Local\Temp\ipykernel_38576\4170036967.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df.torque.fillna(a,inplace=True)
```

```
df.isnull().sum()
```

```
name              0
year              0
selling_price     0
km_driven         0
fuel              0
seller_type       0
transmission      0
owner             0
mileage          87
engine            0
max_power         0
torque            0
seats             0
dtype: int64
```

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

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




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

```
4527    0.0
2725    0.0
6824    0.0
785     0.0
6629    0.0
       ...
7308    NaN
7543    NaN
7642    NaN
7733    NaN
7913    NaN
Name: mileage, Length: 7906, dtype: object
```

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

```
19.3
```

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

```
C:\Users\user\AppData\Local\Temp\ipykernel_38576\1349450490.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame
```

```
See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df.mileage.fillna(a,inplace=True)
```

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

```
      year  selling_price  km_driven  mileage  torque  seats
0     2014         450000     145500    23.40   190.0    5.0
1     2014         370000     120000    21.14   250.0    5.0
2     2006         158000     140000    17.70   170.0    5.0
3     2010         225000     127000    23.00    22.4    5.0
4     2007         130000     120000    16.10   170.0    5.0
...    ...            ...        ...      ...     ...    ...
8123  2013         320000     110000    18.50   113.7    5.0
8124  2007         135000     119000    16.80   170.0    5.0
8125  2009         382000     120000    19.30   190.0    5.0
8126  2013         290000      25000    23.57   140.0    5.0
8127  2013         290000      25000    23.57   140.0    5.0

[7906 rows x 6 columns]
```

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

```
year              2012.00
selling_price    270000.00
km_driven         35000.00
mileage              16.78
torque              111.80
```

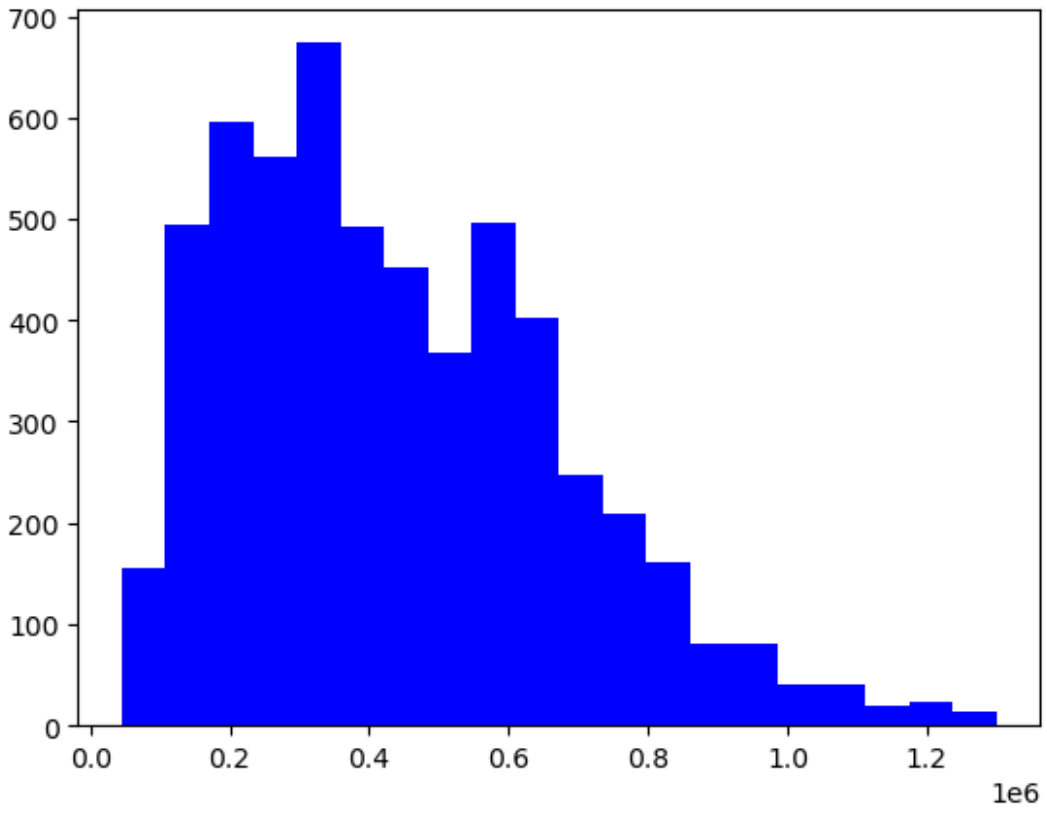
```
seats                  5.00
Name: 0.25, dtype: float64

q3

year                2017.00
selling_price    690000.00
km_driven         95425.00
mileage              22.32
torque              200.00
seats                 5.00
Name: 0.75, dtype: float64

iqr=q3-q1
iqr

year                   5.00
selling_price    420000.00
km_driven         60425.00
mileage               5.54
torque               88.20
seats                 0.00
dtype: float64

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

       year  selling_price  km_driven  mileage  torque  seats
0     False          False      False    False   False  False
1     False          False      False    False   False  False
2     False          False      False    False   False  False
3     False          False      False    False   False  False
4     False          False      False    False   False  False
...     ...            ...        ...      ...     ...    ...
8123  False          False      False    False   False  False
8124  False          False      False    False   False  False
8125  False          False      False    False   False  False
8126  False          False      False    False   False  False
8127  False          False      False    False   False  False

[7906 rows x 6 columns]

df2=df[~(b).any(axis=1)]
df2

                              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         74.0   190.0    5.0
1       103.52   250.0    5.0
2         78.0   170.0    5.0
3         90.0    22.4    5.0
4         88.2   170.0    5.0
```

```
...        ...     ...    ...
8123     82.85   113.7    5.0
8124     110.0   170.0    5.0
8125      73.9   190.0    5.0
8126      70.0   140.0    5.0
8127      70.0   140.0    5.0

[5610 rows x 13 columns]
```

```
plt.hist(df2['selling_price'],bins=20,color='blue')
```

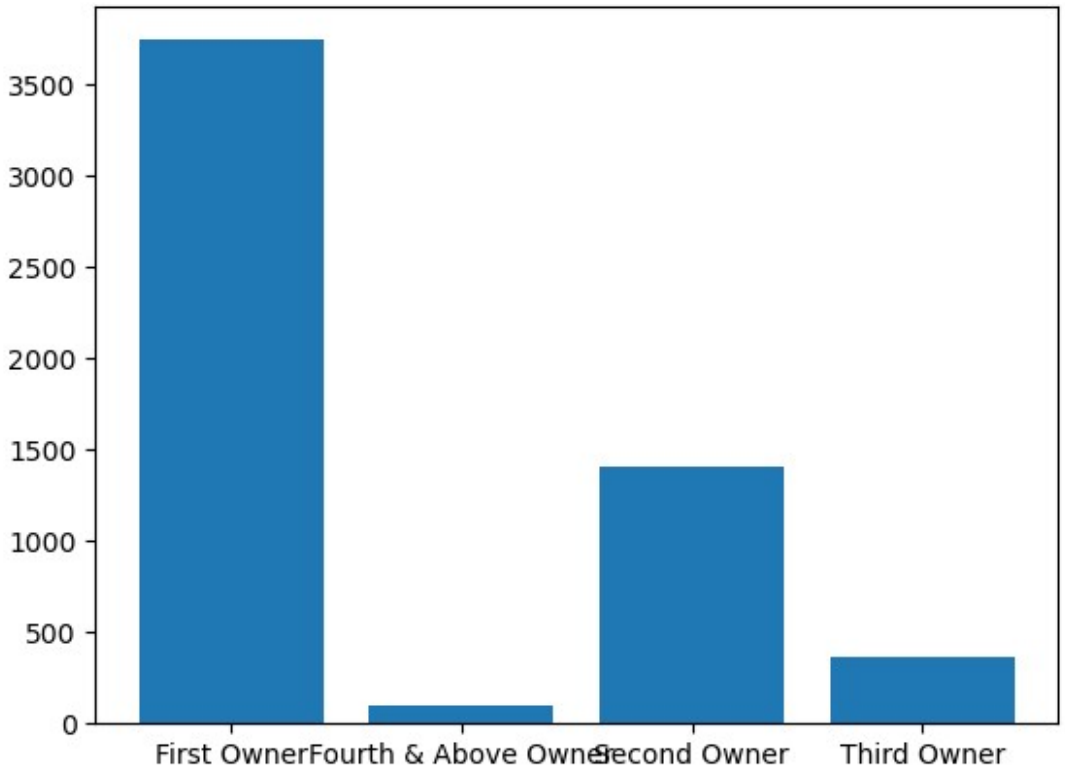
```
(array([156., 495., 596., 561., 674., 492., 453., 368., 496., 403.,
248.,
        209., 161.,  81.,  80.,  40.,  40.,  20.,  23.,  14.]),
 array([  45000.,  107750.,  170500.,  233250.,  296000.,  358750.,
         421500.,  484250.,  547000.,  609750.,  672500.,  735250.,
         798000.,  860750.,  923500.,  986250., 1049000., 1111750.,
        1174500., 1237250., 1300000.]),
 <BarContainer object of 20 artists>)
```




```
df2.groupby(['owner']).count()
```

```
                      name  year  selling_price  km_driven  fuel
seller_type  \
```

```
owner

First Owner            3746  3746            3746        3746  3746
3746
Fourth & Above Owner     97    97              97          97    97
97
Second Owner           1406  1406            1406        1406  1406
1406
Third Owner             361   361             361         361   361
361

                      transmission  mileage  engine  max_power  torque
seats
owner

First Owner                   3746     3746    3746        3746    3746
3746
Fourth & Above Owner            97       97      97          97      97
97
Second Owner                  1406     1406    1406        1406    1406
1406
Third Owner                    361      361     361         361     361
361

a=df2.groupby(['owner']).size().reset_index(name='count').rename(colum
ns={'owner':'Owner'})
a

                  Owner  count
0           First Owner   3746
1  Fourth & Above Owner     97
2          Second Owner   1406
3           Third Owner    361

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

<BarContainer object of 4 artists>
```

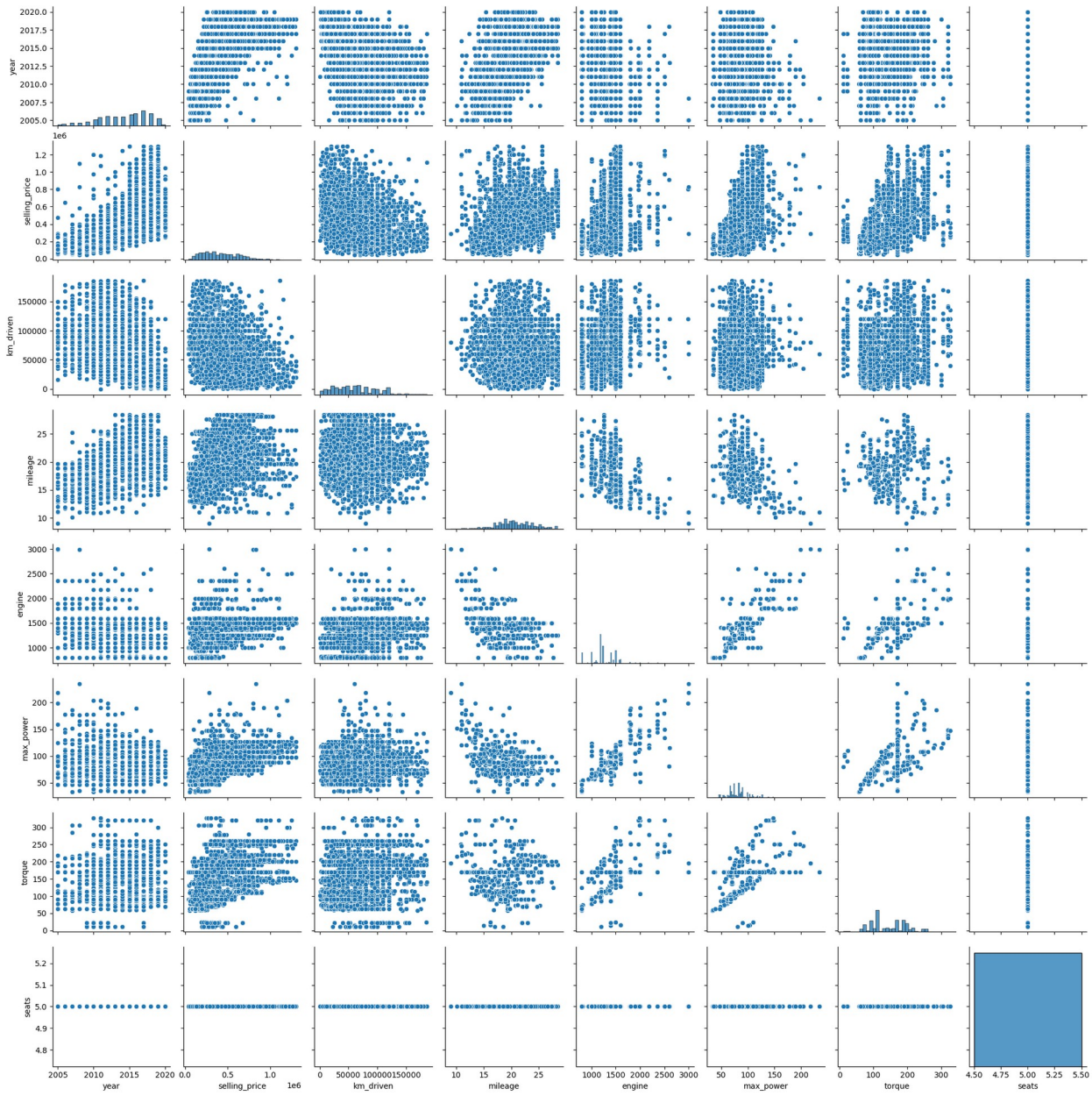



```
a['Percentage']=a['count']/sum(a['count'])*100
a
```

	Owner	count	Percentage
0	First Owner	3746	66.773619
1	Fourth & Above Owner	97	1.729055
2	Second Owner	1406	25.062389
3	Third Owner	361	6.434938

```
sns.pairplot(df2)
```

```
<seaborn.axisgrid.PairGrid at 0x24da931a190>
```




```
df1=df2.select_dtypes(exclude=['object'])

df1
```

	year	selling_price	km_driven	mileage	torque	seats
0	2014	450000	145500	23.40	190.0	5.0
1	2014	370000	120000	21.14	250.0	5.0
2	2006	158000	140000	17.70	170.0	5.0
3	2010	225000	127000	23.00	22.4	5.0
4	2007	130000	120000	16.10	170.0	5.0
...	...	...	...	...	...	...
8123	2013	320000	110000	18.50	113.7	5.0

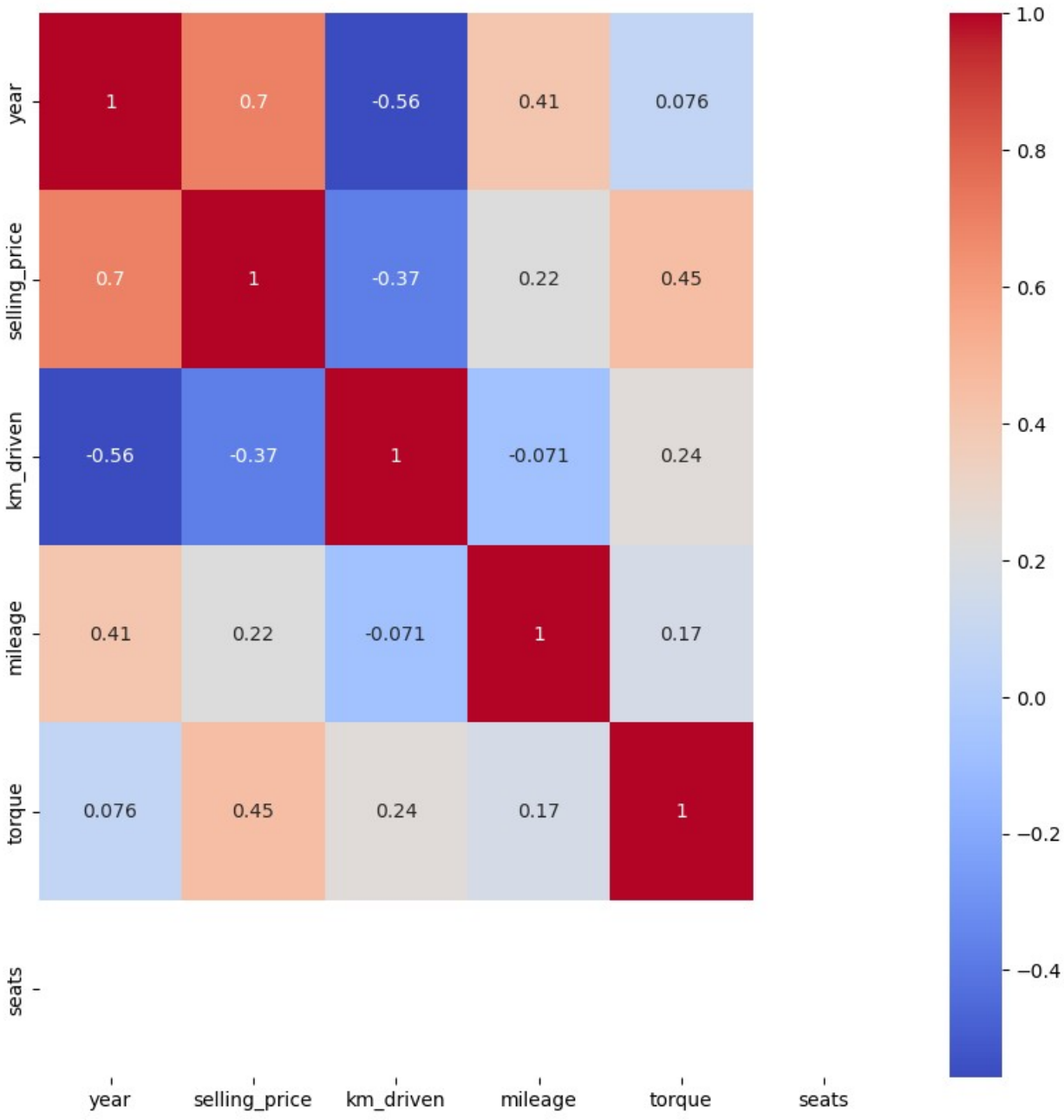
```
8124  2007         135000     119000    16.80   170.0    5.0
8125  2009         382000     120000    19.30   190.0    5.0
8126  2013         290000      25000    23.57   140.0    5.0
8127  2013         290000      25000    23.57   140.0    5.0

[5610 rows x 6 columns]
```

```
df1.corr()
```

```
                   year  selling_price  km_driven   mileage    torque
seats
year           1.000000       0.697896  -0.557120  0.406794  0.075960
NaN
selling_price  0.697896       1.000000  -0.371182  0.225000  0.447797
NaN
km_driven     -0.557120      -0.371182   1.000000 -0.071230  0.244400
NaN
mileage        0.406794       0.225000  -0.071230  1.000000  0.169434
NaN
torque         0.075960       0.447797   0.244400  0.169434  1.000000
NaN
seats               NaN            NaN        NaN       NaN       NaN
NaN
```

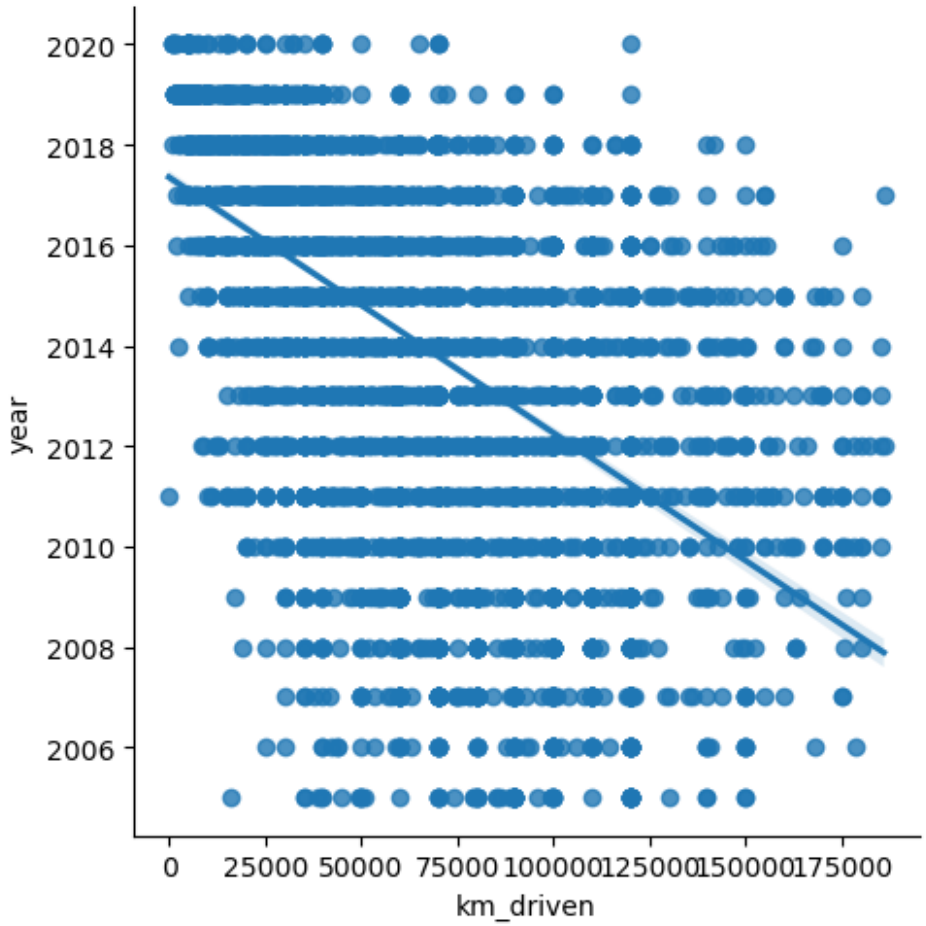
```
plt.figure(figsize=(10,10))
sns.heatmap(df1.corr(),annot=True,cmap='coolwarm')
```

```
<Axes: >
```




```
sns.lmplot(x='km_driven',y='year',data=df1)
```

```
<seaborn.axisgrid.FacetGrid at 0x24dae464810>
```

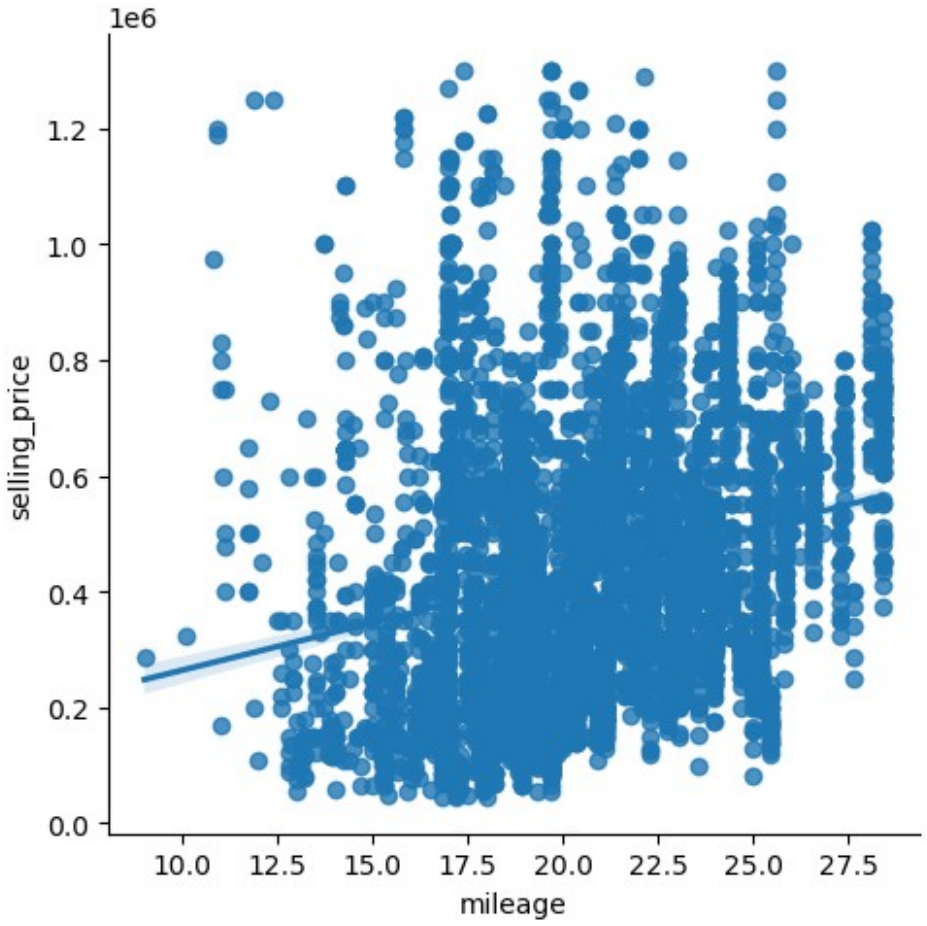



```
df1.columns
```

```
Index(['year', 'selling_price', 'km_driven', 'mileage', 'torque',
'seats'], dtype='object')
```

```
sns.lmplot(x='mileage', y='selling_price', data=df1)
```

```
<seaborn.axisgrid.FacetGrid at 0x24daf41ec90>
```




```
df1
```

	year	selling_price	km_driven	mileage	torque	seats
0	2014	450000	145500	23.40	190.0	5.0
1	2014	370000	120000	21.14	250.0	5.0
2	2006	158000	140000	17.70	170.0	5.0
3	2010	225000	127000	23.00	22.4	5.0
4	2007	130000	120000	16.10	170.0	5.0
...	...	...	...	...	...	...
8123	2013	320000	110000	18.50	113.7	5.0
8124	2007	135000	119000	16.80	170.0	5.0
8125	2009	382000	120000	19.30	190.0	5.0
8126	2013	290000	25000	23.57	140.0	5.0
8127	2013	290000	25000	23.57	140.0	5.0

[5610 rows x 6 columns]

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

```
      km_driven  mileage  torque  seats
0        145500    23.40   190.0    5.0
1        120000    21.14   250.0    5.0
2        140000    17.70   170.0    5.0
3        127000    23.00    22.4    5.0
4        120000    16.10   170.0    5.0
...         ...      ...     ...    ...
8123     110000    18.50   113.7    5.0
8124     119000    16.80   170.0    5.0
8125     120000    19.30   190.0    5.0
8126      25000    23.57   140.0    5.0
8127      25000    23.57   140.0    5.0

[5610 rows x 4 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: 5610, dtype: int64
```

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

```
C:\Users\user\anaconda3\Lib\site-packages\sklearn\feature_selection\
_univariate_selection.py:112: UserWarning: Features [3] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx,
UserWarning)
C:\Users\user\anaconda3\Lib\site-packages\sklearn\feature_selection\
_univariate_selection.py:113: RuntimeWarning: invalid value
encountered in divide
  f = msb / msw

(array([3.5024081 , 3.64473241, 4.48556631,        nan]),
 array([2.31715786e-106, 2.97795605e-114, 2.18187818e-161,
nan]))
```

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

```
km_driven    2.317158e-106
mileage      2.977956e-114
torque       2.181878e-161
seats                  NaN
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         74.0   190.0    5.0
1       103.52   250.0    5.0
2         78.0   170.0    5.0
3         90.0    22.4    5.0
4         88.2   170.0    5.0
...        ...     ...    ...
8123     82.85   113.7    5.0
8124     110.0   170.0    5.0
8125      73.9   190.0    5.0
8126      70.0   140.0    5.0
8127      70.0   140.0    5.0

[7906 rows x 13 columns]

df.fuel.unique()

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

df.name.unique()

array(['Maruti Swift Dzire VDI', 'Skoda Rapid 1.5 TDI Ambition',
       'Honda City 2017-2020 EXi', ..., 'Tata Nexon 1.5 Revotorq XT',
       'Ford Freestyle Titanium Plus Diesel BSIV',
       'Toyota Innova 2.5 GX (Diesel) 8 Seater BS IV'], dtype=object)

df.owner.unique()

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

df.transmission.unique()

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

df.seller_type.unique()

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

df.seats.unique()

array([ 5.,  4.,  7.,  8.,  6.,  9., 10., 14.,  2.])

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

```
df.loc[:, 'fuel'] = le.fit_transform(df['fuel'])
df.loc[:, 'seller_type'] = le.fit_transform(df['seller_type'])
df.loc[:, 'transmission'] = le.fit_transform(df['transmission'])
df.loc[:, 'owner'] = le.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
seats
0              1            1     0    23.40  1248.0      74.0   190.0
5.0
1              1            1     2    21.14  1498.0    103.52   250.0
5.0
2              1            1     4    17.70  1497.0      78.0   170.0
5.0
3              1            1     0    23.00  1396.0      90.0    22.4
5.0
4              1            1     0    16.10  1298.0      88.2   170.0
5.0
...          ...          ...   ...      ...     ...       ...     ...
...
8123           1            1     0    18.50  1197.0     82.85   113.7
5.0
8124           1            1     1    16.80  1493.0     110.0   170.0
5.0
```

```
8125           1            1     0    19.30  1248.0      73.9   190.0
5.0
8126           1            1     0    23.57  1396.0      70.0   140.0
5.0
8127           1            1     0    23.57  1396.0      70.0   140.0
5.0

[7906 rows x 13 columns]
```

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

```
array([0, 2, 4, 1, 3], dtype=object)
```

```
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
4
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      74.0   190.0
1       21.14  1498.0    103.52   250.0
2       17.70  1497.0      78.0   170.0
3       23.00  1396.0      90.0    22.4
4       16.10  1298.0      88.2   170.0
...       ...     ...       ...     ...
8123    18.50  1197.0     82.85   113.7
8124    16.80  1493.0     110.0   170.0
```

```
8125    19.30  1248.0      73.9   190.0
8126    23.57  1396.0      70.0   140.0
8127    23.57  1396.0      70.0   140.0

[7906 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
engine  \
0     2014     145500    1           1            1     0    23.40
1248.0
1     2014     120000    1           1            1     2    21.14
1498.0
2     2006     140000    3           1            1     4    17.70
1497.0
3     2010     127000    1           1            1     0    23.00
1396.0
4     2007     120000    3           1            1     0    16.10
1298.0
...    ...        ...  ...         ...          ...   ...      ...
...
8123  2013     110000    3           1            1     0    18.50
1197.0
8124  2007     119000    1           1            1     1    16.80
1493.0
8125  2009     120000    1           1            1     0    19.30
1248.0
8126  2013      25000    1           1            1     0    23.57
1396.0
8127  2013      25000    1           1            1     0    23.57
1396.0

     max_power  torque
0         74.0   190.0
1       103.52   250.0
2         78.0   170.0
3         90.0    22.4
4         88.2   170.0
...        ...     ...
8123     82.85   113.7
8124     110.0   170.0
8125      73.9   190.0
8126      70.0   140.0
```

```
8127      70.0   140.0

[7906 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: 7906, dtype: int64
```

```
from sklearn.preprocessing import StandardScaler
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)
```

```
rfr=RandomForestRegressor()
param_grid = {
    'n_estimators': [50, 300, 50],
    'max_depth': [15,20,25],
    'min_samples_split': [5,7,8]
}
```

```
grid_search=GridSearchCV(estimator=rfr,param_grid=param_grid,cv=5,scor
ing='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': [50, 300, 50]},
             scoring='neg_mean_squared_error')
```

```
grid_search.best_params_
```

```
{'max_depth': 20, 'min_samples_split': 5, 'n_estimators': 300}
```

```
grid_search.best_score_
```

```
-26737815091.785797
```

```
pr=grid_search.predict(x_test)
```

```
pr
```

```
array([539573.11422664, 531749.96695635, 167581.17739791, ...,
       593376.61110053, 394402.00414345, 101755.20071576])
```

```
mean_squared_error(y_test,pr)
```

```
18751994085.315895
```

```
r2_score(y_test,pr)
```

```
0.9719219152704778
```

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

```
RandomForestRegressor()
```

```
pr=rfr.predict(x_test)
```

```
pr
```

```
array([558454.96      , 514706.61666667, 168379.96      , ...,
       596529.04761905, 407589.99      , 105638.61      ])
```

```
mean_squared_error(y_test,pr)
```

```
18216651963.18847
```

```
 r2_score(y_test,pr)
```

```
0.9727235036986731
```

Decision Tree

```
from sklearn.tree import DecisionTreeRegressor
```

```
dt = DecisionTreeRegressor()
```

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

```
DecisionTreeRegressor()
```

```
pr=dt.predict(x_test)
```

```
pr
```

```
array([650000., 510000., 168000., ..., 650000., 350000., 140000.])
```

```
mean_squared_error(y_test,pr)
```

```
25662084253.10888
```

```
r2_score(y_test,pr)
```

```
0.9615751704743143
```

KNN Regressor

```
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': 3, 'weights': 'distance'}
```

```
knn1.best_score_
```

```
-41954849166.251816
```

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

```
31233827831.19562
```

```
 r2_score(y_test,pr)
```

```
0.9532323836984163
```