Project Title : Car Price Prediction

In [161…

```
from IPython.display import Image
Image(filename='car.jpg')
```

Out[161…

Project Overview :

With the rise in the variety of cars with differentiated capabilities and features such as model, production year, category, brand, fuel type, engine volume, mileage, cylinders, colour, airbags and many more, we are bringing a car price prediction challenge for all. We all aspire to own a car within budget with the best features available..

Import Libraries

In [165…

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

EDA

1. Initial Data Understanding

- Data loading and Inspection
- Data Types
- Missing Values
- Duplicates

In [169…

```
df = pd.read_csv('car_price.csv')
```

In [170…

```
df.head()
```

Out[170…

	ID	Price	Levy	Manufacturer	Model	Prod_year	Category	Leather interior	Fuel type	Engine volume	Mileage	Cylinders	Gear box type	w
0	45654403	13328	1399	LEXUS	RX 450	2010	Jeep	Yes	Hybrid	3.5	186005 km	6	Automatic	
1	44731507	16621	1018	CHEVROLET	Equinox	2011	Jeep	No	Petrol	3	192000 km	6	Tiptronic	
2	45774419	8467	-	HONDA	FIT	2006	Hatchback	No	Petrol	1.3	200000 km	4	Variator	
3	45769185	3607	862	FORD	Escape	2011	Jeep	Yes	Hybrid	2.5	168966 km	4	Automatic	
4	45809263	11726	446	HONDA	FIT	2014	Hatchback	Yes	Petrol	1.3	91901 km	4	Automatic	

In [171…

```python
df.sample(10)
```

Out[171…

	ID	Price	Levy	Manufacturer	Model	Prod_year	Category	Leather interior	Fuel type	Engine volume	Mileage	Cylinders	Gear b ty
7305	45813055	392	761	TOYOTA	Prius	2010	Hatchback	Yes	Hybrid	1.8	454699 km	4	Autom
1298	45768950	314	1172	MERCEDES-BENZ	E 350	2011	Sedan	Yes	Diesel	3.5	132630 km	6	Autom
12689	45430408	11290	-	AUDI	Allroad	2001	Jeep	Yes	Petrol	2.7	210000 km	6	Mar
5087	45644670	251	607	TOYOTA	Camry	2019	Sedan	Yes	Hybrid	2.5	39552 km	4	Autom
14869	45771036	2430	1598	MERCEDES-BENZ	E 350	2008	Sedan	Yes	Diesel	3	175614 km	6	Autom
18843	45773181	470	934	SUBARU	Forester	2015	Jeep	Yes	Petrol	2.5	87141 km	4	Autom
14545	45529910	21012	308	TOYOTA	Prius	2014	Hatchback	No	Plug-in Hybrid	1.8	112000 km	4	Autom
17547	44371531	21326	-	SUBARU	Forester	2013	Jeep	No	Petrol	2.5	116800 km	4	Varia
373	45658776	941	1058	LEXUS	RX 450	2012	Jeep	Yes	Hybrid	3.5	232357 km	6	Autom
241	45754449	5331	-	VOLKSWAGEN	Passat	2002	Sedan	Yes	Petrol	2.8	175000 km	6	Mar

In [172…

```python
df.tail()
```

Out[172…

	ID	Price	Levy	Manufacturer	Model	Prod_year	Category	Leather interior	Fuel type	Engine volume	Mileage	Cylinders	Gear box type
19232	45798355	8467	-	MERCEDES-BENZ	CLK 200	1999	Coupe	Yes	CNG	2.0 Turbo	300000 km	4	Manual
19233	45778856	15681	831	HYUNDAI	Sonata	2011	Sedan	Yes	Petrol	2.4	161600 km	4	Tiptronic
19234	45804997	26108	836	HYUNDAI	Tucson	2010	Jeep	Yes	Diesel	2	116365 km	4	Automatic
19235	45793526	5331	1288	CHEVROLET	Captiva	2007	Jeep	Yes	Diesel	2	51258 km	4	Automatic
19236	45813273	470	753	HYUNDAI	Sonata	2012	Sedan	Yes	Hybrid	2.4	186923 km	4	Automatic

In [173…

```python
df.shape
```

Out[173…

```
(19237, 18)
```

In [174…

```python
df.columns
```

Out[174…

```
Index(['ID', 'Price', 'Levy', 'Manufacturer', 'Model', 'Prod_year', 'Category',
       'Leather interior', 'Fuel type', 'Engine volume', 'Mileage',
       'Cylinders', 'Gear box type', 'Drive wheels', 'Doors', 'Wheel', 'Color',
       'Airbags'],
      dtype='object')
```

In [175…

```
df = df.rename(columns={'Engine volume':'Engine_volume','Fuel type':'Fuel_type','Leather interior':'Leather_int
```

In [176…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19237 entries, 0 to 19236
Data columns (total 18 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   ID                19237 non-null  int64 
 1   Price             19237 non-null  int64 
 2   Levy              19237 non-null  object
 3   Manufacturer      19237 non-null  object
 4   Model             19237 non-null  object
 5   Prod_year         19237 non-null  int64 
 6   Category          19237 non-null  object
 7   Leather_interior  19237 non-null  object
 8   Fuel_type         19237 non-null  object
 9   Engine_volume     19237 non-null  object
 10  Mileage           19237 non-null  object
 11  Cylinders         19237 non-null  int64 
 12  Gear_box_type     19237 non-null  object
 13  Drive_wheels      19237 non-null  object
 14  Doors             19237 non-null  object
 15  Wheel             19237 non-null  object
 16  Color             19237 non-null  object
 17  Airbags           19237 non-null  int64 
dtypes: int64(5), object(13)
memory usage: 2.6+ MB
```

In [177…

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

Out[177…

```
ID                  0
Price               0
Levy                0
Manufacturer        0
Model               0
Prod_year           0
Category            0
Leather_interior    0
Fuel_type           0
Engine_volume       0
Mileage             0
Cylinders           0
Gear_box_type       0
Drive_wheels        0
Doors               0
Wheel               0
Color               0
Airbags             0
dtype: int64
```

In [178…

```
df.duplicated().sum()
```

Out[178…

```
313
```

2. Basic Statistical Overview

- Summary Statistical : **describe()**

In [181…

```
df.describe()
```

Out[181…

	ID	Price	Prod_year	Cylinders	Airbags
count	1.923700e+04	1.923700e+04	19237.000000	19237.000000	19237.000000
mean	4.557654e+07	1.855593e+04	2010.912824	4.582991	6.582627
std	9.365914e+05	1.905813e+05	5.668673	1.199933	4.320168
min	2.074688e+07	1.000000e+00	1939.000000	1.000000	0.000000
25%	4.569837e+07	5.331000e+03	2009.000000	4.000000	4.000000
50%	4.577231e+07	1.317200e+04	2012.000000	4.000000	6.000000
75%	4.580204e+07	2.207500e+04	2015.000000	4.000000	12.000000
max	4.581665e+07	2.630750e+07	2020.000000	16.000000	16.000000

In [182…

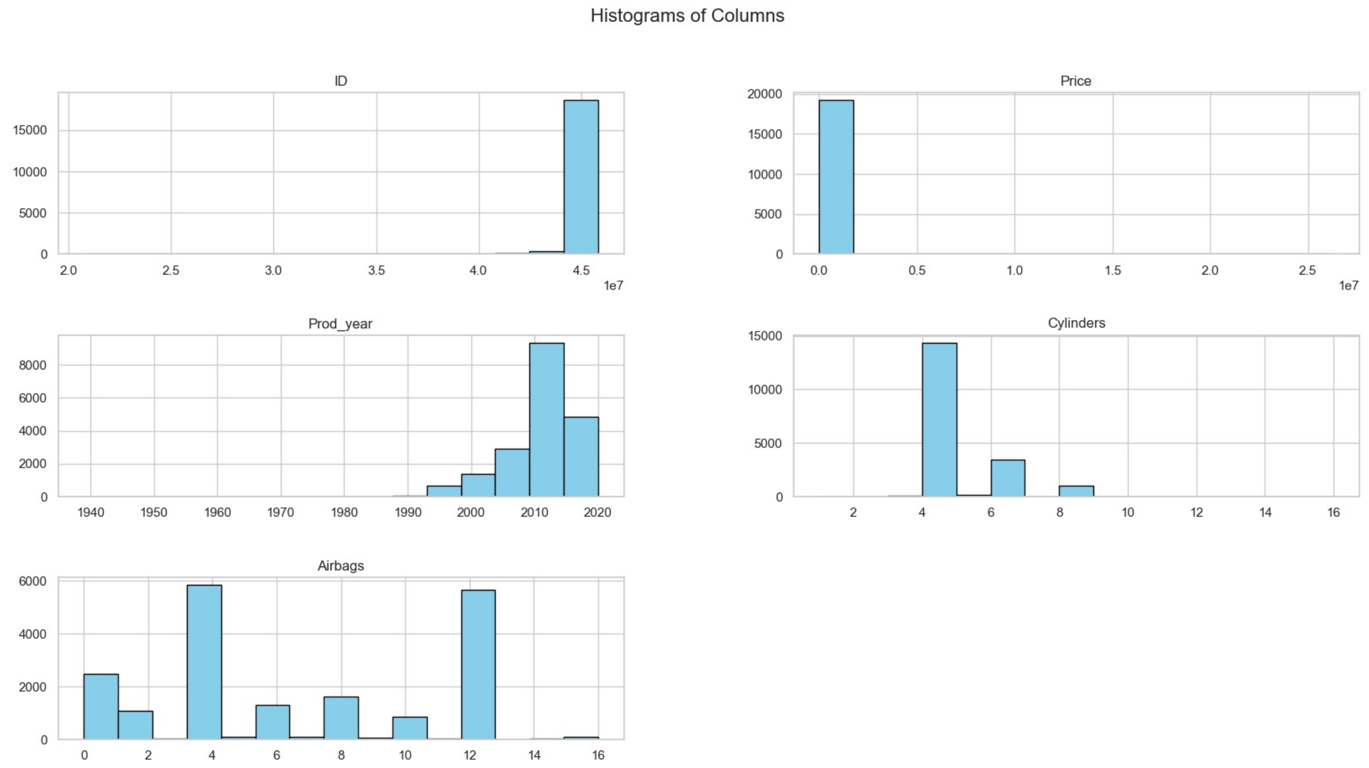
```
df.select_dtypes(include='object').describe()
```

Out[182…

	Levy	Manufacturer	Model	Category	Leather_interior	Fuel_type	Engine_volume	Mileage	Gear_box_type	Drive_wheels
count	19237	19237	19237	19237	19237	19237	19237	19237	19237	19237
unique	559	65	1590	11	2	7	107	7687	4	3
top	-	HYUNDAI	Prius	Sedan	Yes	Petrol	2	0 km	Automatic	Front
freq	5819	3769	1083	8736	13954	10150	3916	721	13514	12874

In [183…

```python
df.hist(bins=15, figsize=(20, 10), color='skyblue', edgecolor='black')

plt.suptitle('Histograms of Columns', fontsize=16)
plt.subplots_adjust(hspace=0.5)
plt.show()
```



- Summary Statistical : **Value_counts()**

In [185…

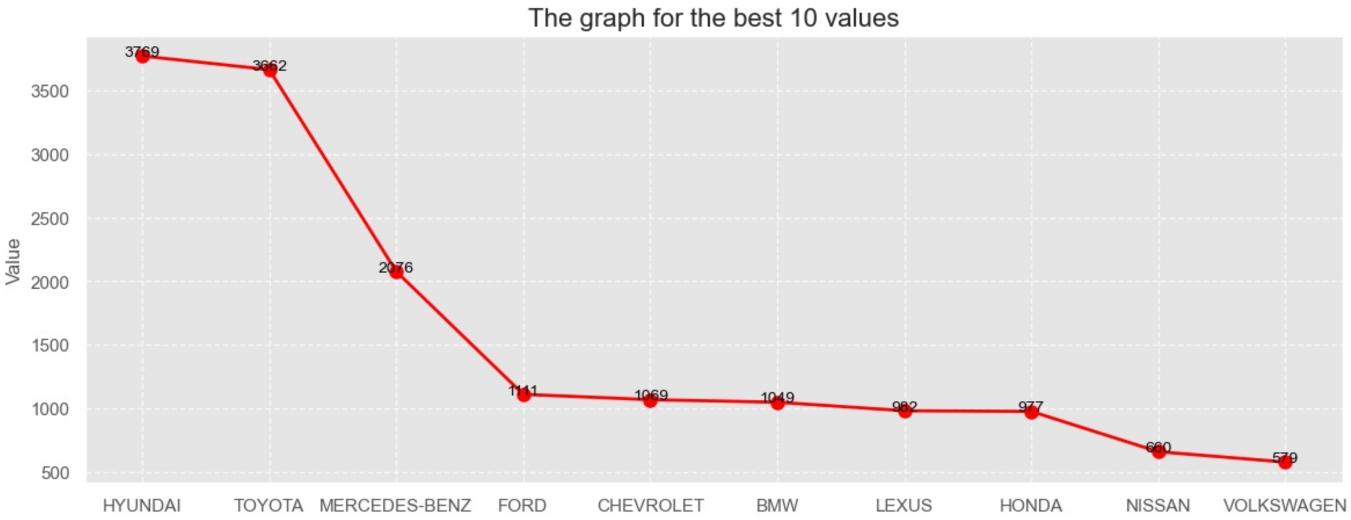
```python
top10=df['Manufacturer'].value_counts().sort_values(ascending=False)[:10]
top10
```

Out[185…

```
Manufacturer
HYUNDAI          3769
TOYOTA           3662
MERCEDES-BENZ    2076
FORD             1111
CHEVROLET        1069
BMW              1049
LEXUS             982
HONDA             977
NISSAN            660
VOLKSWAGEN        579
Name: count, dtype: int64
```

In [186…

```python
plt.style.use('ggplot')
plt.figure(figsize=(14, 5))

plt.plot(top10, marker='o', color='red', linestyle='-', linewidth=2, markersize=8)
plt.title('The graph for the best 10 values', fontsize=16)
plt.ylabel('Value', fontsize=12)

for i, value in enumerate(top10):
    plt.text(i, value + 0.5, str(value), ha='center', fontsize=10, color='black')

plt.grid(True, linestyle='--', alpha=0.7)

plt.show()
```



In [187…

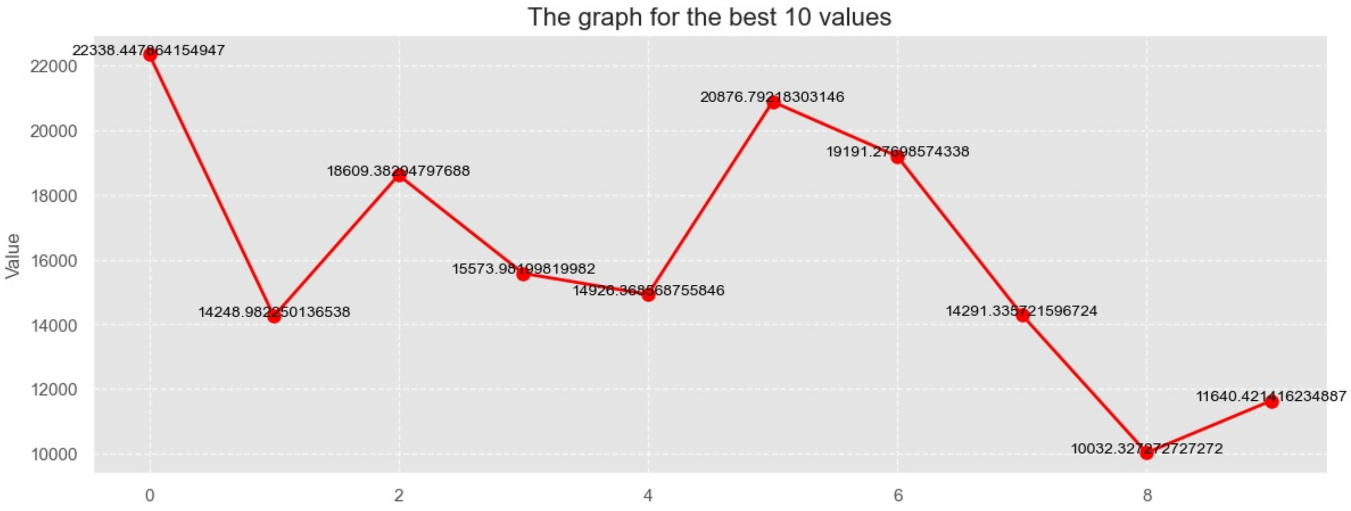
```python
top10MeanPrices=[df[df['Manufacturer']==i]['Price'].mean() for i in list(top10.index)]
top10MeanPrices
```

Out[187…

```
[22338.447864154947,
 14248.982250136538,
 18609.38294797688,
 15573.98199819982,
 14926.368568755846,
 20876.79218303146,
 19191.27698574338,
 14291.335721596724,
 10032.327272727272,
 11640.421416234887]
```

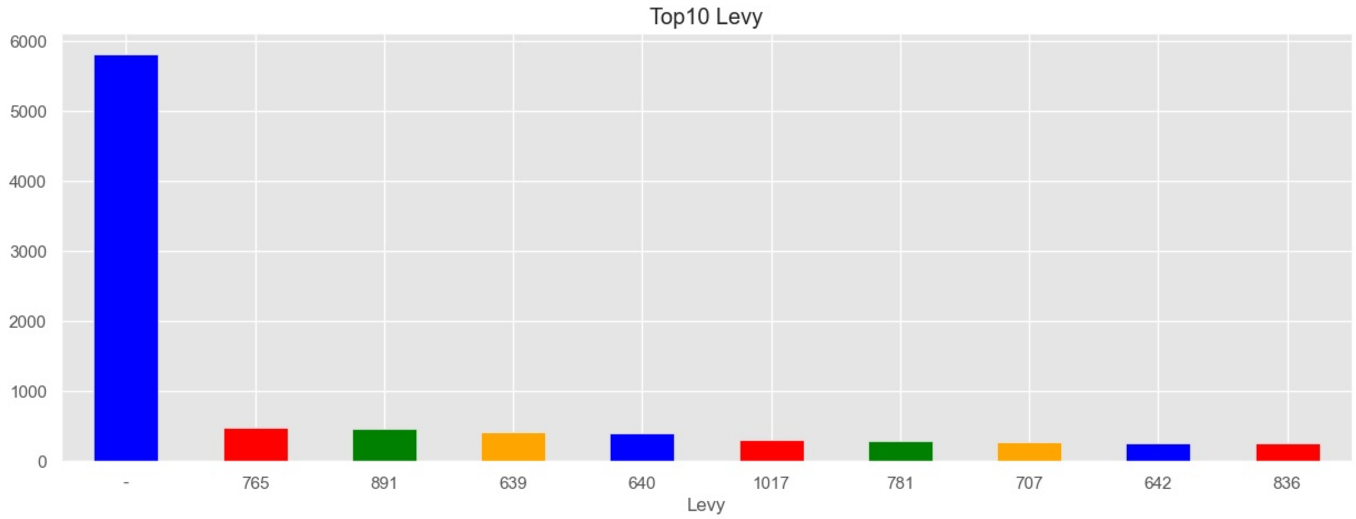
In [188…

```python
plt.style.use('ggplot')
plt.figure(figsize=(14, 5))

plt.plot(top10MeanPrices, marker='o', color='red', linestyle='-', linewidth=2, markersize=8)
plt.title('The graph for the best 10 values', fontsize=16)
plt.ylabel('Value', fontsize=12)

for i, value in enumerate(top10MeanPrices):
    plt.text(i, value + 0.5, str(value), ha='center', fontsize=10, color='black')

plt.grid(True, linestyle='--', alpha=0.7)

plt.show()
```

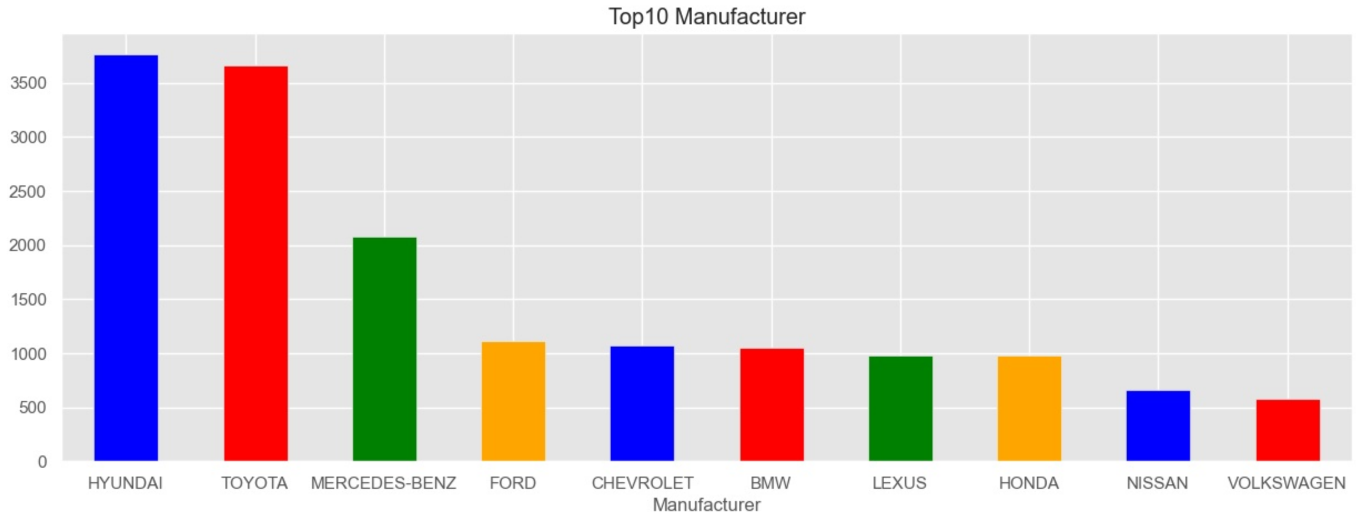


In [189…

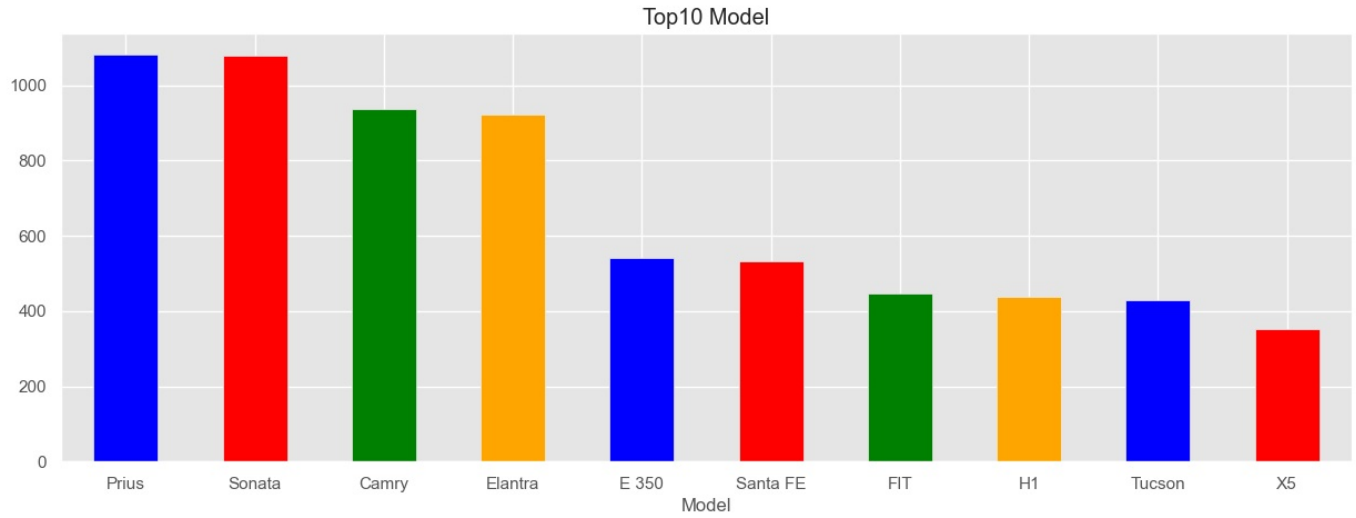
```python
object_data = df.select_dtypes(include='object')

for col in object_data:
    plt.style.use('ggplot')
    plt.figure(figsize=(15,5))
    Top10=df[col].value_counts()[:10]
    colors=['blue','red','green','orange']
    Top10.plot(kind='bar',color=colors)
    plt.xticks(rotation='horizontal' )
    plt.title('Top10'+' '+col)
    plt.show()
```



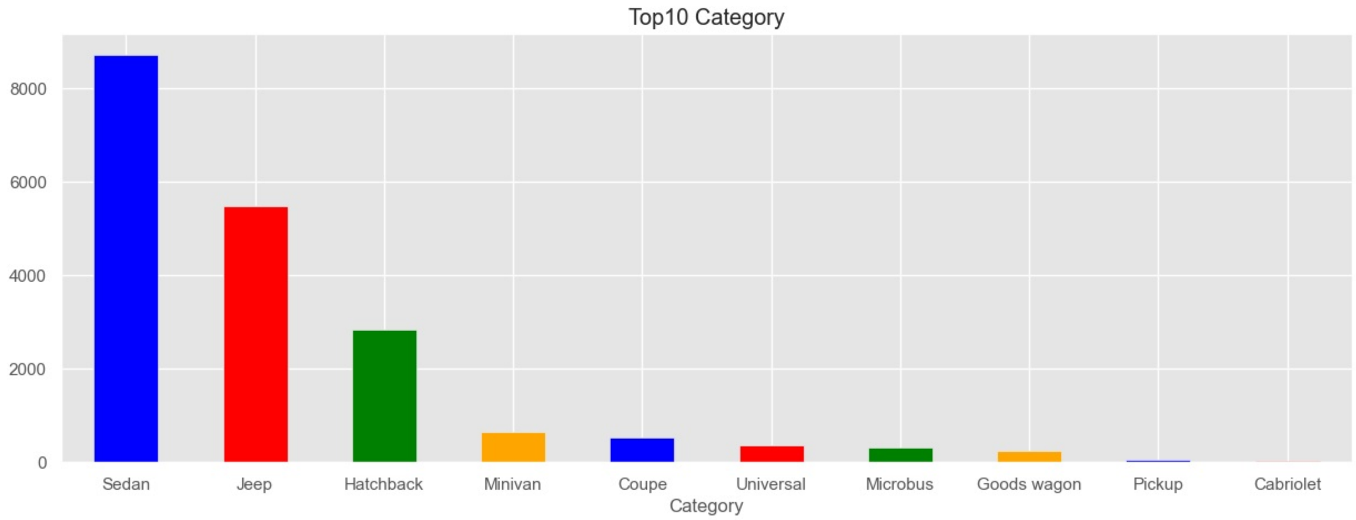
Top10 Levy



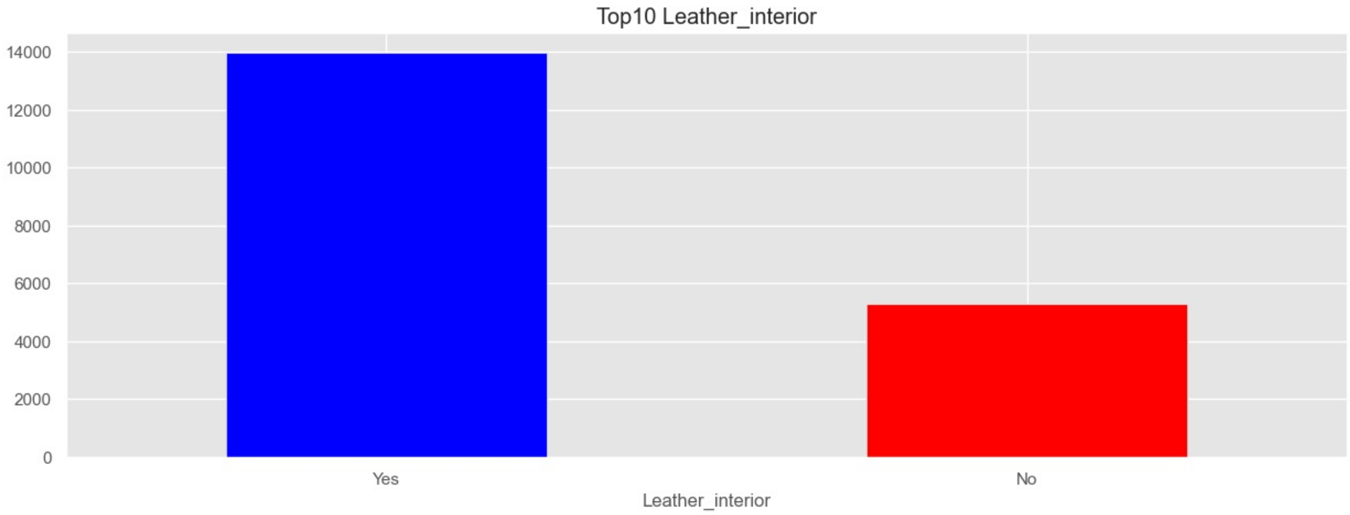
Top10 Manufacturer

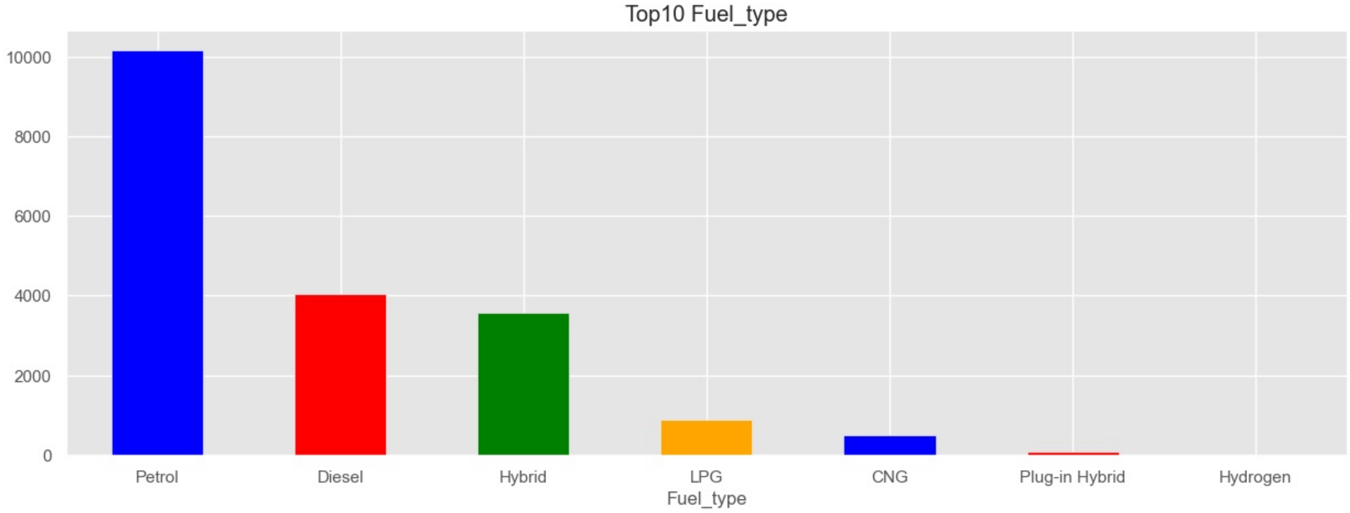


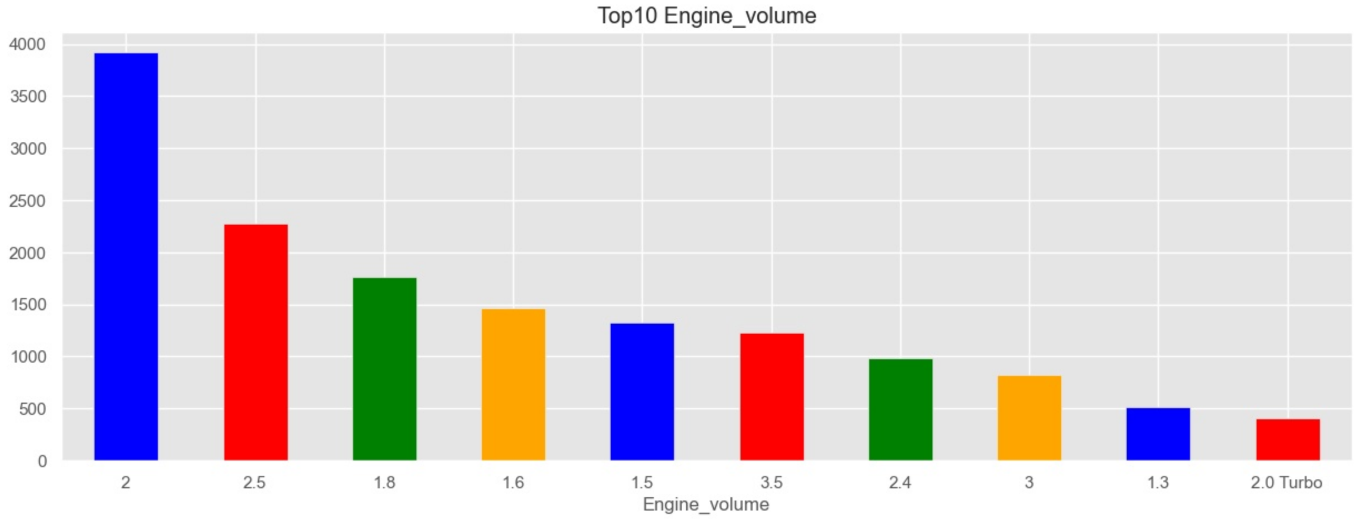
Top10 Model

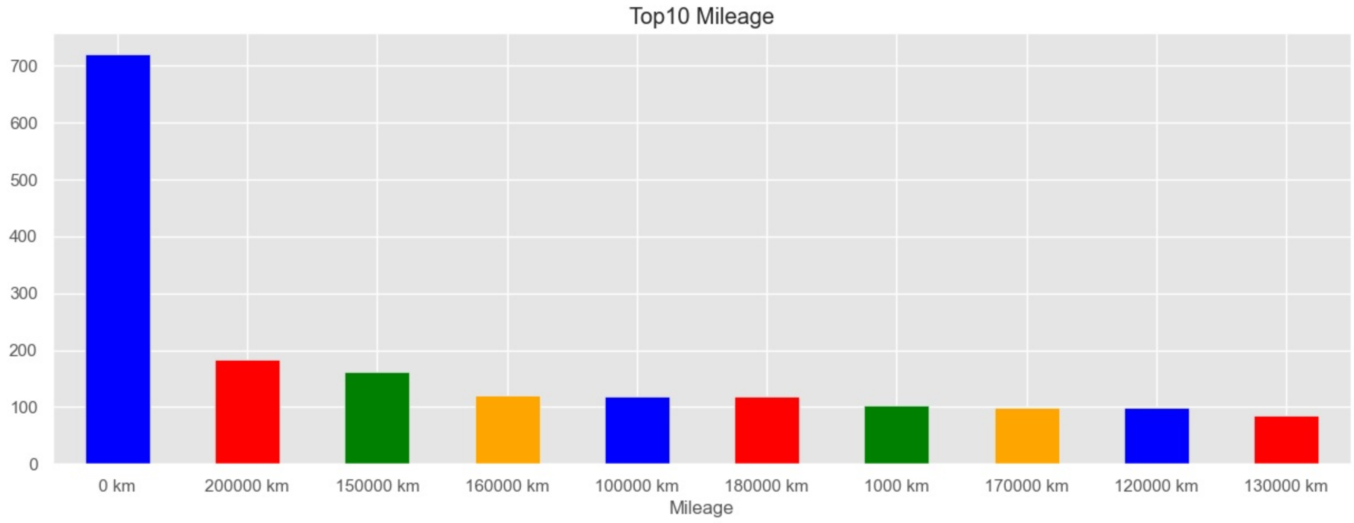


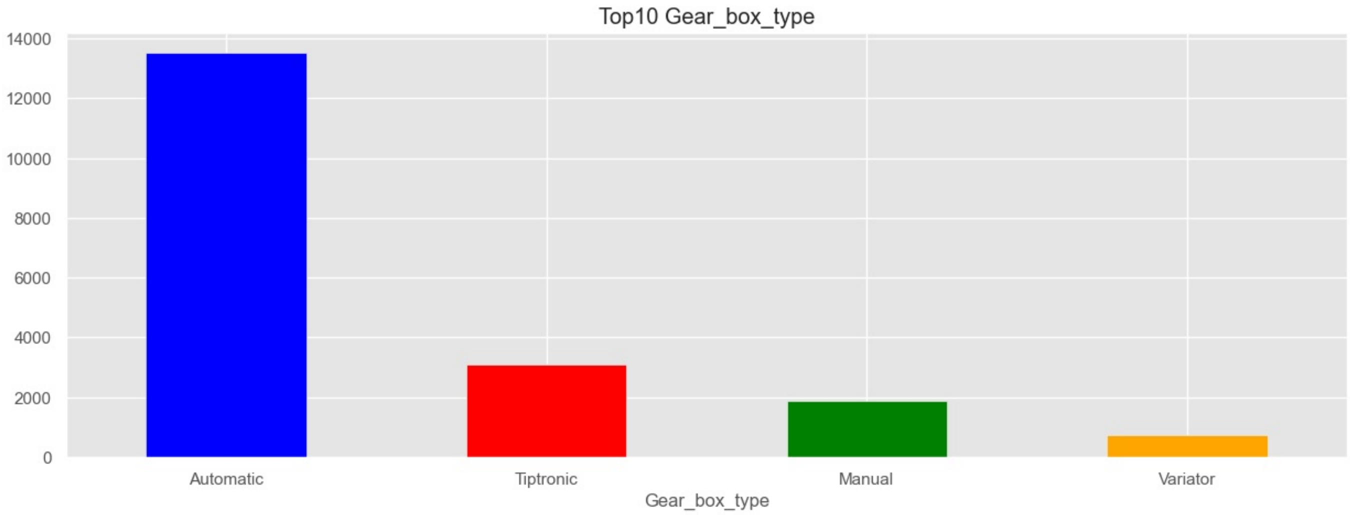
Top10 Category

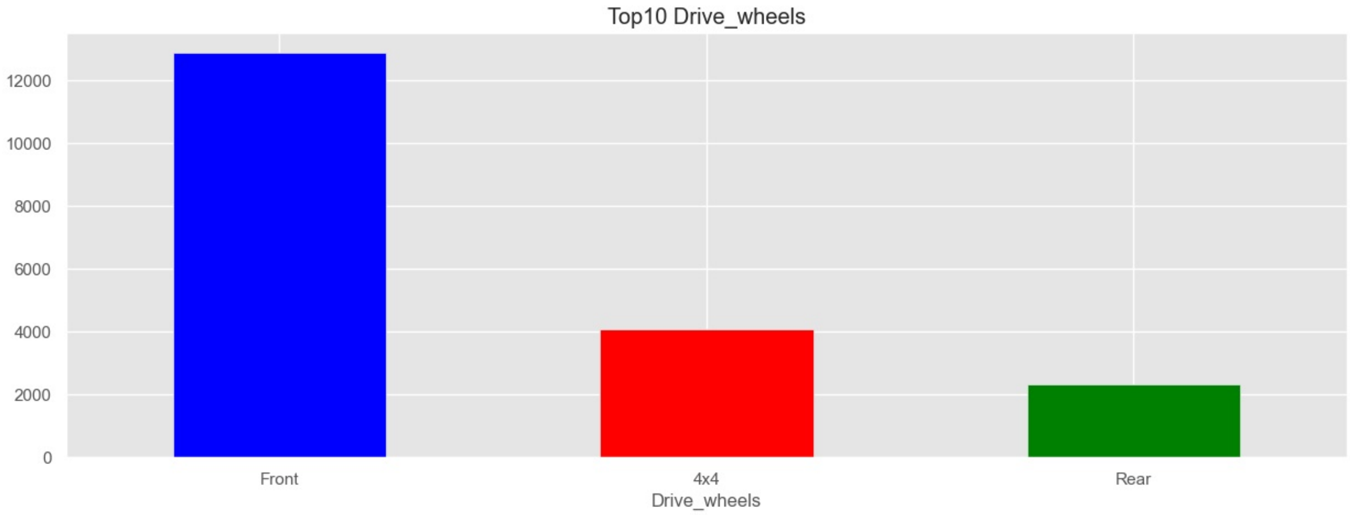

Top10 Leather_interior
14000
12000
10000
8000
6000
4000
2000
0
Yes
No
Leather_interior

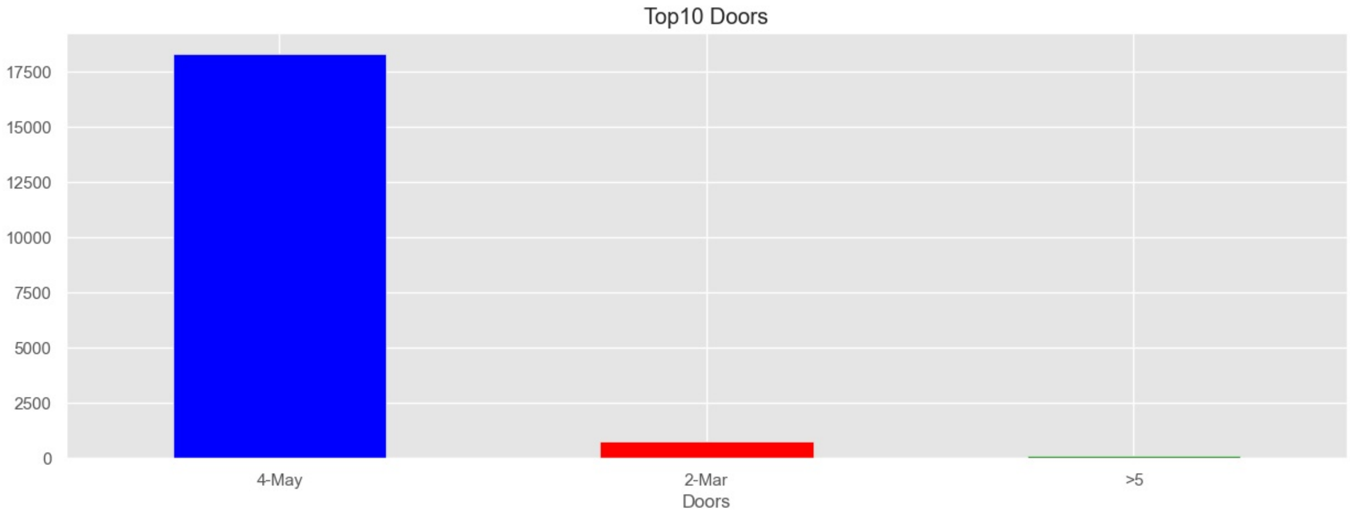

Top10 Fuel_type
10000
8000
6000
4000
2000
0
Petrol
Diesel
Hybrid
LPG
CNG
Plug-in Hybrid
Hydrogen
Fuel_type

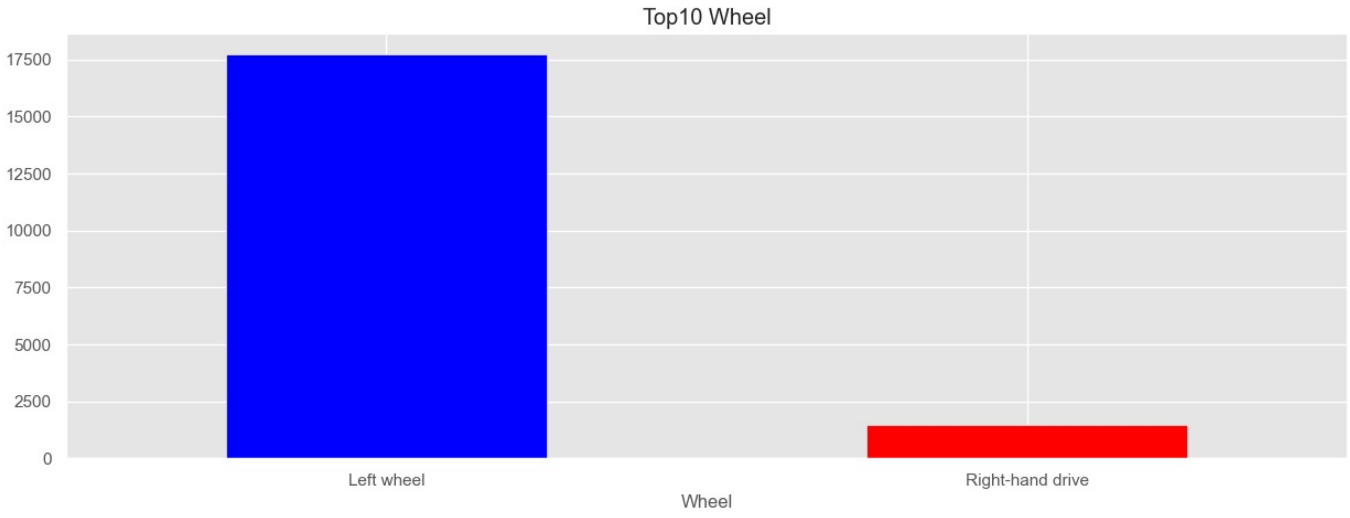

Top10 Engine_volume
4000
3500
3000
2500
2000
1500
1000
500
0
2
2.5
1.8
1.6
1.5
3.5
2.4
3
1.3
2.0 Turbo
Engine_volume

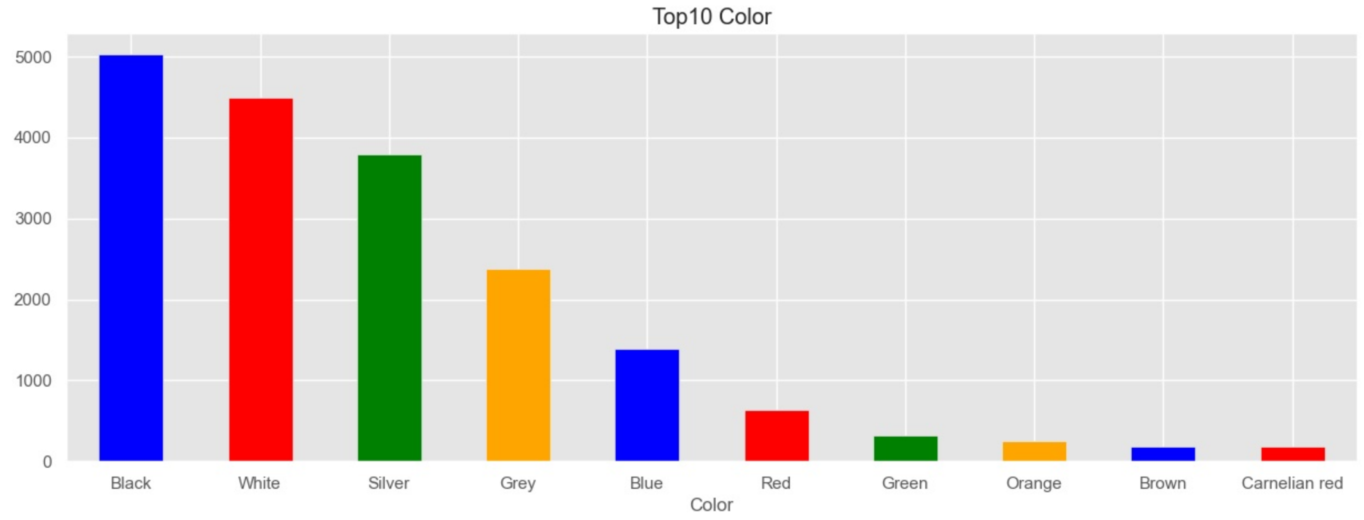

Top10 Mileage
700
600
500
400
300
200
100
0
0 km
200000 km
150000 km
160000 km
100000 km
180000 km
1000 km
170000 km
120000 km
130000 km
Mileage


Top10 Gear_box_type
14000
12000
10000
8000
6000
4000
2000
0
Automatic
Tiptronic
Manual
Variator
Gear_box_type


Top10 Drive_wheels
12000
10000
8000
6000
4000
2000
0
Front
4x4
Rear
Drive_wheels


Top10 Doors
17500
15000
12500
10000
7500
5000
2500
0
4-May
2-Mar
>5
Doors


Top10 Wheel
17500
15000
12500
10000
7500
5000
2500
0
Left wheel
Right-hand drive
Wheel



In [190…
```
df['Levy'].value_counts()
```

Out[190…
```
Levy
-       5819
765      486
891      461
639      410
640      405
        ... 
3156       1
2908       1
1279       1
1719       1
1901       1
Name: count, Length: 559, dtype: int64
```

In [191…
```
df['Engine_volume'].value_counts()
```

Out[191…
```
Engine_volume
2            3916
2.5          2277
1.8          1760
1.6          1462
1.5          1321
             ... 
6.8             1
6.7             1
3.1             1
0.8 Turbo       1
1.1 Turbo       1
Name: count, Length: 107, dtype: int64
```

In [192…
```
df['Mileage'].value_counts()
```

Out[192…
```
Mileage
0 km         721
200000 km    183
150000 km    161
160000 km    120
100000 km    119
            ... 
63083 km       1
28750 km       1
25077 km       1
77452 km       1
186923 km      1
Name: count, Length: 7687, dtype: int64
```

Data Cleaning

In [195…
```
df.drop_duplicates(inplace=True)
```

In [196…
```
df.shape
```

Out[196…
```
(18924, 18)
```

```
In [198… df['Levy'].unique()
```

```
Out[198… array(['1399', '1018', '-', '862', '446', '891', '761', '751', '394',
                '1053', '1055', '1079', '810', '2386', '1850', '531', '586',
                '1249', '2455', '583', '1537', '1288', '915', '1750', '707',
                '1077', '1486', '1091', '650', '382', '1436', '1194', '503',
                '1017', '1104', '639', '629', '919', '781', '530', '640', '765',
                '777', '779', '934', '769', '645', '1185', '1324', '830', '1187',
                '1111', '760', '642', '1604', '1095', '966', '473', '1138', '1811',
                '988', '917', '1156', '687', '11714', '836', '1347', '2866',
                '1646', '259', '609', '697', '585', '475', '690', '308', '1823',
                '1361', '1273', '924', '584', '2078', '831', '1172', '893', '1872',
                '1885', '1266', '447', '2148', '1730', '730', '289', '502', '333',
                '1325', '247', '879', '1342', '1327', '1598', '1514', '1058',
                '738', '1935', '481', '1522', '1282', '456', '880', '900', '798',
                '1277', '442', '1051', '790', '1292', '1047', '528', '1211',
                '1493', '1793', '574', '930', '1998', '271', '706', '1481', '1677',
                '1661', '1286', '1408', '1090', '595', '1451', '1267', '993',
                '1714', '878', '641', '749', '1511', '603', '353', '877', '1236',
                '1141', '397', '784', '1024', '1357', '1301', '770', '922', '1438',
                '753', '607', '1363', '638', '490', '431', '565', '517', '833',
                '489', '1760', '986', '1841', '1620', '1360', '474', '1099', '978',
                '1624', '1946', '1268', '1307', '696', '649', '666', '2151', '551',
                '800', '971', '1323', '2377', '1845', '1083', '694', '463', '419',
                '345', '1515', '1505', '2056', '1203', '729', '460', '1356', '876',
                '911', '1190', '780', '448', '2410', '1848', '1148', '834', '1275',
                '1028', '1197', '724', '890', '1705', '505', '789', '2959', '518',
                '461', '1719', '2858', '3156', '2225', '2177', '1968', '1888',
                '1308', '2736', '1103', '557', '2195', '843', '1664', '723',
                '4508', '562', '501', '2018', '1076', '1202', '3301', '691',
                '1440', '1869', '1178', '418', '1820', '1413', '488', '1304',
                '363', '2108', '521', '1659', '87', '1411', '1528', '3292', '7058',
                '1578', '627', '874', '1996', '1488', '5679', '1234', '5603',
                '400', '889', '3268', '875', '949', '2265', '441', '742', '425',
                '2476', '2971', '614', '1816', '1375', '1405', '2297', '1062',
                '1113', '420', '2469', '658', '1951', '2670', '2578', '1995',
                '1032', '994', '1011', '2421', '1296', '155', '494', '426', '1086',
                '961', '2236', '1829', '764', '1834', '1054', '617', '1529',
                '2266', '637', '626', '1832', '1016', '2002', '1756', '746',
                '1285', '2690', '1118', '5332', '980', '1807', '970', '1228',
                '1195', '1132', '1768', '1384', '1080', '7063', '1817', '1452',
                '1975', '1368', '702', '1974', '1781', '1036', '944', '663', '364',
                '1539', '1345', '1680', '2209', '741', '1575', '695', '1317',
                '294', '1525', '424', '997', '1473', '1552', '2819', '2188',
                '1668', '3057', '799', '1502', '2606', '552', '1694', '1759',
                '1110', '399', '1470', '1174', '5877', '1474', '1688', '526',
                '686', '5908', '1107', '2070', '1468', '1246', '1685', '556',
                '1533', '1917', '1346', '732', '692', '579', '421', '362', '3505',
                '1855', '2711', '1586', '3739', '681', '1708', '2278', '1701',
                '722', '1482', '928', '827', '832', '527', '604', '173', '1341',
                '3329', '1553', '859', '167', '916', '828', '2082', '1176', '1108',
                '975', '3008', '1516', '2269', '1699', '2073', '1031', '1503',
                '2364', '1030', '1442', '5666', '2715', '1437', '2067', '1426',
                '2908', '1279', '866', '4283', '279', '2658', '3015', '2004',
                '1391', '4736', '748', '1466', '644', '683', '2705', '1297', '731',
                '1252', '2216', '3141', '3273', '1518', '1723', '1588', '972',
                '682', '1094', '668', '175', '967', '402', '3894', '1960', '1599',
                '2000', '2084', '1621', '714', '1109', '3989', '873', '1572',
                '1163', '1991', '1716', '1673', '2562', '2874', '965', '462',
                '605', '1948', '1736', '3518', '2054', '2467', '1681', '1272',
                '1205', '750', '2156', '2566', '115', '524', '3184', '676', '1678',
                '612', '328', '955', '1441', '1675', '3965', '2909', '623', '822',
                '867', '3025', '1993', '792', '636', '4057', '3743', '2337',
                '2570', '2418', '2472', '3910', '1662', '2123', '2628', '3208',
                '2080', '3699', '2913', '864', '2505', '870', '7536', '1924',
                '1671', '1064', '1836', '1866', '4741', '841', '1369', '5681',
                '3112', '1366', '2223', '1198', '1039', '3811', '3571', '1387',
                '1171', '1365', '1531', '1590', '11706', '2308', '4860', '1641',
                '1045', '1901'], dtype=object)
```

```
In [199… # replace (-) by (0) in Levy column
         df['Levy'].replace({'-':0},inplace=True)
         df['Levy']=df['Levy'].astype(float)
```

```
C:\Users\RPC\AppData\Local\Temp\ipykernel_11504\272051335.py:2: FutureWarning: A value is trying to be set on a
copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on w
hich we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)'
or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  df['Levy'].replace({'-':0},inplace=True)
```

In [200…
```
df['Levy'].unique()
```

Out[200…
```
array([ 1399.,  1018.,     0.,   862.,   446.,   891.,   761.,   751.,
         394.,  1053.,  1055.,  1079.,   810.,  2386.,  1850.,   531.,
         586.,  1249.,  2455.,   583.,  1537.,  1288.,   915.,  1750.,
         707.,  1077.,  1486.,  1091.,   650.,   382.,  1436.,  1194.,
         503.,  1017.,  1104.,   639.,   629.,   919.,   781.,   530.,
         640.,   765.,   777.,   779.,   934.,   769.,   645.,  1185.,
        1324.,   830.,  1187.,  1111.,   760.,   642.,  1604.,  1095.,
         966.,   473.,  1138.,  1811.,   988.,   917.,  1156.,   687.,
       11714.,   836.,  1347.,  2866.,  1646.,   259.,   609.,   697.,
         585.,   475.,   690.,   308.,  1823.,  1361.,  1273.,   924.,
         584.,  2078.,   831.,  1172.,   893.,  1872.,  1885.,  1266.,
         447.,  2148.,  1730.,   730.,   289.,   502.,   333.,  1325.,
         247.,   879.,  1342.,  1327.,  1598.,  1514.,  1058.,   738.,
        1935.,   481.,  1522.,  1282.,   456.,   880.,   900.,   798.,
        1277.,   442.,  1051.,   790.,  1292.,  1047.,   528.,  1211.,
        1493.,  1793.,   574.,   930.,  1998.,   271.,   706.,  1481.,
        1677.,  1661.,  1286.,  1408.,  1090.,   595.,  1451.,  1267.,
         993.,  1714.,   878.,   641.,   749.,  1511.,   603.,   353.,
         877.,  1236.,  1141.,   397.,   784.,  1024.,  1357.,  1301.,
         770.,   922.,  1438.,   753.,   607.,  1363.,   638.,   490.,
         431.,   565.,   517.,   833.,   489.,  1760.,   986.,  1841.,
        1620.,  1360.,   474.,  1099.,   978.,  1624.,  1946.,  1268.,
        1307.,   696.,   649.,   666.,  2151.,   551.,   800.,   971.,
        1323.,  2377.,  1845.,  1083.,   694.,   463.,   419.,   345.,
        1515.,  1505.,  2056.,  1203.,   729.,   460.,  1356.,   876.,
         911.,  1190.,   780.,   448.,  2410.,  1848.,  1148.,   834.,
        1275.,  1028.,  1197.,   724.,   890.,  1705.,   505.,   789.,
        2959.,   518.,   461.,  1719.,  2858.,  3156.,  2225.,  2177.,
        1968.,  1888.,  1308.,  2736.,  1103.,   557.,  2195.,   843.,
        1664.,   723.,  4508.,   562.,   501.,  2018.,  1076.,  1202.,
        3301.,   691.,  1440.,  1869.,  1178.,   418.,  1820.,  1413.,
         488.,  1304.,   363.,  2108.,   521.,  1659.,    87.,  1411.,
        1528.,  3292.,  7058.,  1578.,   627.,   874.,  1996.,  1488.,
        5679.,  1234.,  5603.,   400.,   889.,  3268.,   875.,   949.,
        2265.,   441.,   742.,   425.,  2476.,  2971.,   614.,  1816.,
        1375.,  1405.,  2297.,  1062.,  1113.,   420.,  2469.,   658.,
        1951.,  2670.,  2578.,  1995.,  1032.,   994.,  1011.,  2421.,
        1296.,   155.,   494.,   426.,  1086.,   961.,  2236.,  1829.,
         764.,  1834.,  1054.,   617.,  1529.,  2266.,   637.,   626.,
        1832.,  1016.,  2002.,  1756.,   746.,  1285.,  2690.,  1118.,
        5332.,   980.,  1807.,   970.,  1228.,  1195.,  1132.,  1768.,
        1384.,  1080.,  7063.,  1817.,  1452.,  1975.,  1368.,   702.,
        1974.,  1781.,  1036.,   944.,   663.,   364.,  1539.,  1345.,
        1680.,  2209.,   741.,  1575.,   695.,  1317.,   294.,  1525.,
         424.,   997.,  1473.,  1552.,  2819.,  2188.,  1668.,  3057.,
         799.,  1502.,  2606.,   552.,  1694.,  1759.,  1110.,   399.,
        1470.,  1174.,  5877.,  1474.,  1688.,   526.,   686.,  5908.,
        1107.,  2070.,  1468.,  1246.,  1685.,   556.,  1533.,  1917.,
        1346.,   732.,   692.,   579.,   421.,   362.,  3505.,  1855.,
        2711.,  1586.,  3739.,   681.,  1708.,  2278.,  1701.,   722.,
        1482.,   928.,   827.,   832.,   527.,   604.,   173.,  1341.,
        3329.,  1553.,   859.,   167.,   916.,   828.,  2082.,  1176.,
        1108.,   975.,  3008.,  1516.,  2269.,  1699.,  2073.,  1031.,
        1503.,  2364.,  1030.,  1442.,  5666.,  2715.,  1437.,  2067.,
        1426.,  2908.,  1279.,   866.,  4283.,   279.,  2658.,  3015.,
        2004.,  1391.,  4736.,   748.,  1466.,   644.,   683.,  2705.,
        1297.,   731.,  1252.,  2216.,  3141.,  3273.,  1518.,  1723.,
        1588.,   972.,   682.,  1094.,   668.,   175.,   967.,   402.,
        3894.,  1960.,  1599.,  2000.,  2084.,  1621.,   714.,  1109.,
        3989.,   873.,  1572.,  1163.,  1991.,  1716.,  1673.,  2562.,
        2874.,   965.,   462.,   605.,  1948.,  1736.,  3518.,  2054.,
        2467.,  1681.,  1272.,  1205.,   750.,  2156.,  2566.,   115.,
         524.,  3184.,   676.,  1678.,   612.,   328.,   955.,  1441.,
        1675.,  3965.,  2909.,   623.,   822.,   867.,  3025.,  1993.,
         792.,   636.,  4057.,  3743.,  2337.,  2570.,  2418.,  2472.,
        3910.,  1662.,  2123.,  2628.,  3208.,  2080.,  3699.,  2913.,
         864.,  2505.,   870.,  7536.,  1924.,  1671.,  1064.,  1836.,
        1866.,  4741.,   841.,  1369.,  5681.,  3112.,  1366.,  2223.,
        1198.,  1039.,  3811.,  3571.,  1387.,  1171.,  1365.,  1531.,
        1590., 11706.,  2308.,  4860.,  1641.,  1045.,  1901.])
```

In [201…
```
df['Levy'].mean()
```

Out[201…
```
632.8864933417882
```

In [202…
```
df['Levy'].replace({0:np.nan},inplace=True)

m=df['Levy'].mean()
df['Levy'].fillna(m,inplace=True)
```

```
C:\Users\RPC\AppData\Local\Temp\ipykernel_11504\1171977336.py:1: FutureWarning: A value is trying to be set on a
copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on w
hich we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)'
or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  df['Levy'].replace({0:np.nan},inplace=True)
C:\Users\RPC\AppData\Local\Temp\ipykernel_11504\1171977336.py:4: FutureWarning: A value is trying to be set on a
copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on w
hich we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)'
or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  df['Levy'].fillna(m,inplace=True)
```

In [203…
```
df['Levy'].value_counts()
```

Out[203…
```
Levy
906.299205     5709
765.000000      482
891.000000      453
639.000000      403
640.000000      398
               ... 
3156.000000       1
2908.000000       1
1279.000000       1
1719.000000       1
1901.000000       1
Name: count, Length: 559, dtype: int64
```

In [205…
```
df['Engine_volume'].unique()
```

Out[205…
```
array(['3.5', '3', '1.3', '2.5', '2', '1.8', '2.4', '4', '1.6', '3.3',
       '2.0 Turbo', '2.2 Turbo', '4.7', '1.5', '4.4', '3.0 Turbo',
       '1.4 Turbo', '3.6', '2.3', '1.5 Turbo', '1.6 Turbo', '2.2',
       '2.3 Turbo', '1.4', '5.5', '2.8 Turbo', '3.2', '3.8', '4.6', '1.2',
       '5', '1.7', '2.9', '0.5', '1.8 Turbo', '2.4 Turbo', '3.5 Turbo',
       '1.9', '2.7', '4.8', '5.3', '0.4', '2.8', '3.2 Turbo', '1.1',
       '2.1', '0.7', '5.4', '1.3 Turbo', '3.7', '1', '2.5 Turbo', '2.6',
       '1.9 Turbo', '4.4 Turbo', '4.7 Turbo', '0.8', '0.2 Turbo', '5.7',
       '4.8 Turbo', '4.6 Turbo', '6.7', '6.2', '1.2 Turbo', '3.4',
       '1.7 Turbo', '6.3 Turbo', '2.7 Turbo', '4.3', '4.2', '2.9 Turbo',
       '0', '4.0 Turbo', '20', '3.6 Turbo', '0.3', '3.7 Turbo', '5.9',
       '5.5 Turbo', '0.2', '2.1 Turbo', '5.6', '6', '0.7 Turbo',
       '0.6 Turbo', '6.8', '4.5', '0.6', '7.3', '0.1', '1.0 Turbo', '6.3',
       '4.5 Turbo', '0.8 Turbo', '4.2 Turbo', '3.1', '5.0 Turbo', '6.4',
       '3.9', '5.7 Turbo', '0.9', '0.4 Turbo', '5.4 Turbo', '0.3 Turbo',
       '5.2', '5.8', '1.1 Turbo'], dtype=object)
```

In [206…
```
# replace (Turbo) by ('') in Engine volume column
df['Engine_volume']=df['Engine_volume'].str.replace('Turbo','')
df['Engine_volume'] = pd.to_numeric(df['Engine_volume'])
```

In [207…
```
df['Engine_volume'].unique()
```

Out[207…
```
array([ 3.5,  3. ,  1.3,  2.5,  2. ,  1.8,  2.4,  4. ,  1.6,  3.3,  2.2,
        4.7,  1.5,  4.4,  1.4,  3.6,  2.3,  5.5,  2.8,  3.2,  3.8,  4.6,
        1.2,  5. ,  1.7,  2.9,  0.5,  1.9,  2.7,  4.8,  5.3,  0.4,  1.1,
        2.1,  0.7,  5.4,  3.7,  1. ,  2.6,  0.8,  0.2,  5.7,  6.7,  6.2,
        3.4,  6.3,  4.3,  4.2,  0. , 20. ,  0.3,  5.9,  5.6,  6. ,  0.6,
        6.8,  4.5,  7.3,  0.1,  3.1,  6.4,  3.9,  0.9,  5.2,  5.8])
```

In [209…
```
df['Mileage'].unique()
```

```
Out[209… array(['186005 km', '192000 km', '200000 km', ..., '140607 km',
                '307325 km', '186923 km'], dtype=object)
```

```
In [210… # replace (km) by ('') in Mileage column
         df['Mileage']=df['Mileage'].str.replace('km','')
         df['Mileage']=pd.to_numeric(df['Mileage'])
```

```
In [211… df['Mileage'].unique()
```

```
Out[211… array([186005, 192000, 200000, ..., 140607, 307325, 186923], dtype=int64)
```

```
In [213… df=df.drop(['ID','Doors'],axis=1)
```

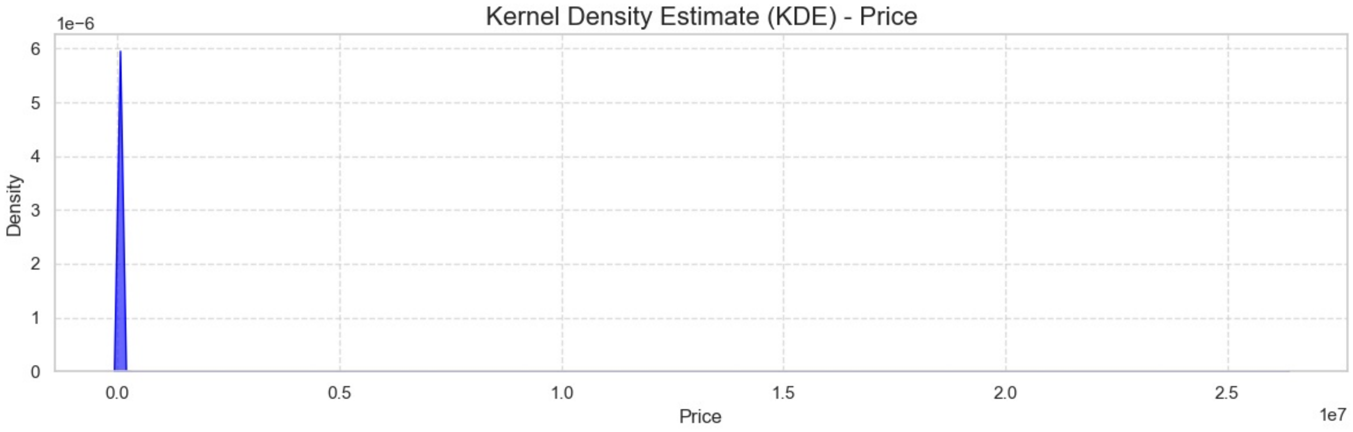
```
In [214… df.info()
```

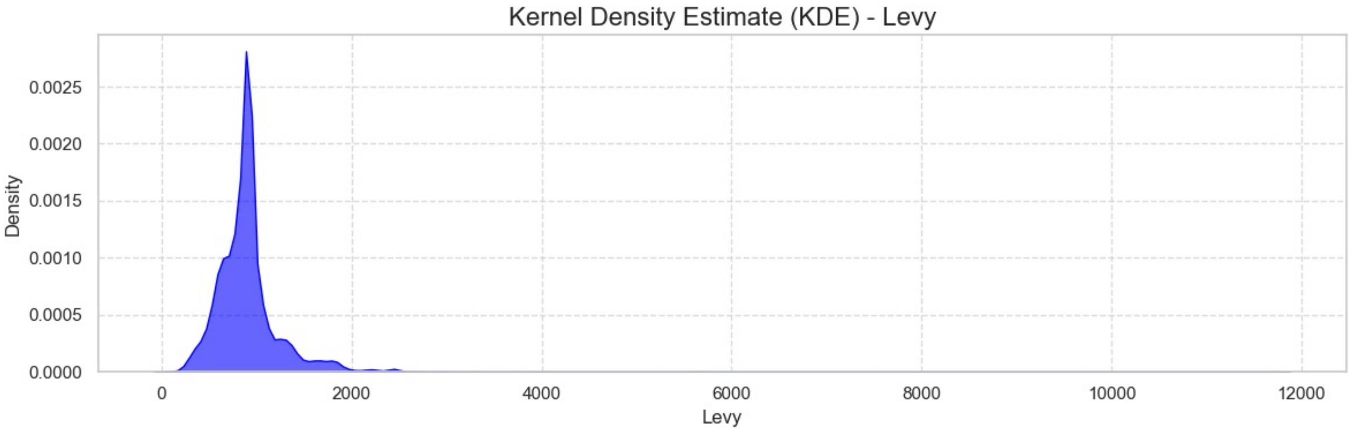
```
<class 'pandas.core.frame.DataFrame'>
Index: 18924 entries, 0 to 19236
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Price             18924 non-null  int64  
 1   Levy              18924 non-null  float64
 2   Manufacturer      18924 non-null  object 
 3   Model             18924 non-null  object 
 4   Prod_year         18924 non-null  int64  
 5   Category          18924 non-null  object 
 6   Leather_interior  18924 non-null  object 
 7   Fuel_type         18924 non-null  object 
 8   Engine_volume     18924 non-null  float64
 9   Mileage           18924 non-null  int64  
 10  Cylinders         18924 non-null  int64  
 11  Gear_box_type     18924 non-null  object 
 12  Drive_wheels      18924 non-null  object 
 13  Wheel             18924 non-null  object 
 14  Color             18924 non-null  object 
 15  Airbags           18924 non-null  int64  
dtypes: float64(2), int64(5), object(9)
memory usage: 2.5+ MB
```

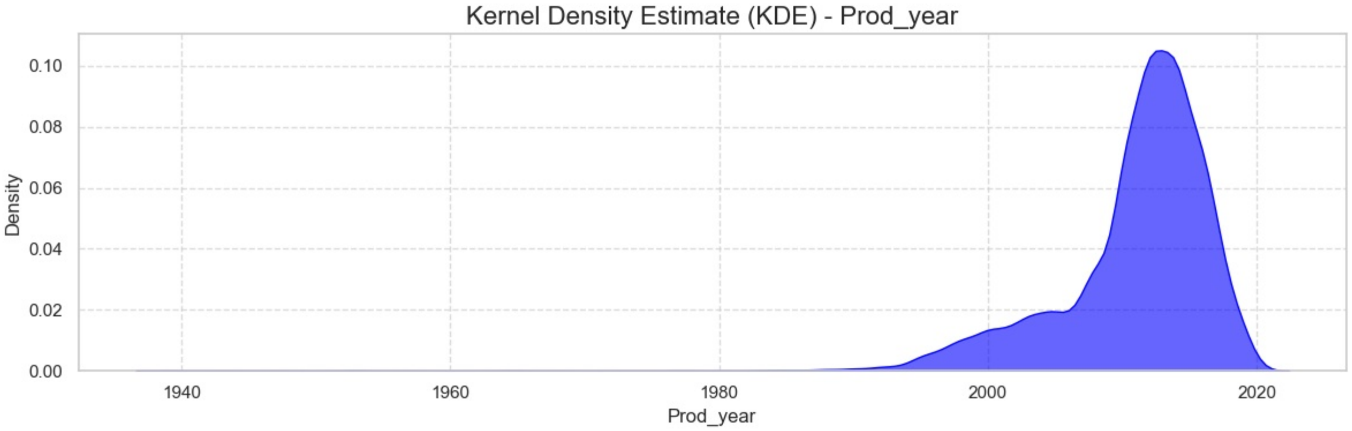
Distribution of Variables

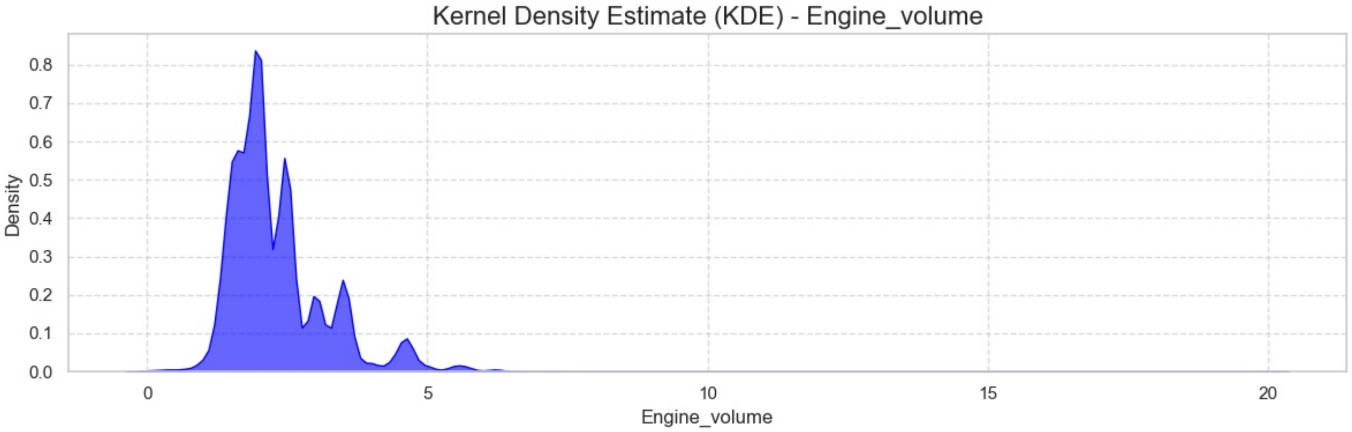
- Numerical Features (KDE)

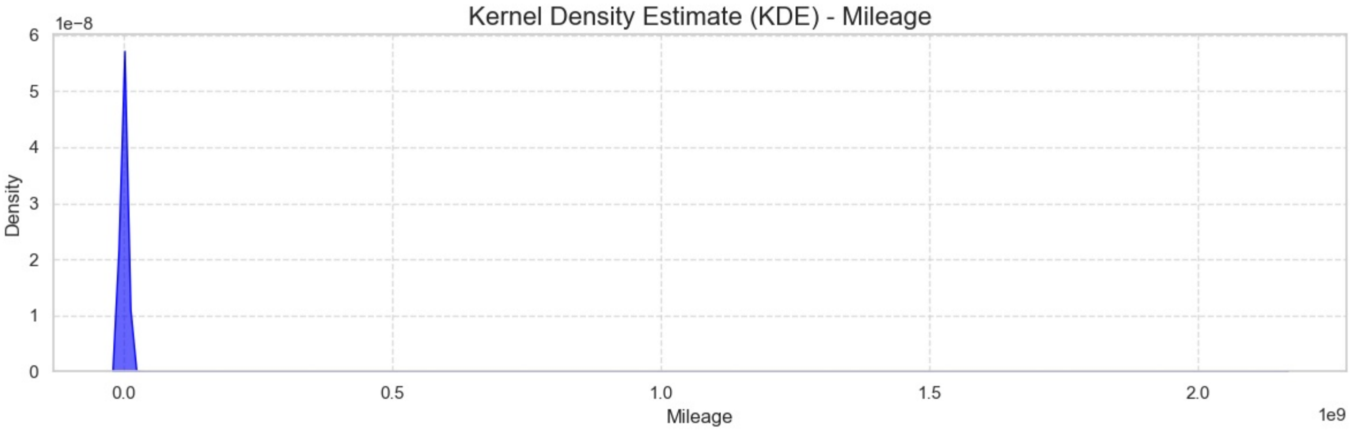
```
In [218… sns.set(style="whitegrid")

         for col in df.select_dtypes('number').columns:
             plt.figure(figsize=(12, 4))
             sns.kdeplot(df[col], fill=True, color='blue', alpha=0.6)
             plt.title(f'Kernel Density Estimate (KDE) - {col}', fontsize=16)
             plt.xlabel(col, fontsize=12)
             plt.ylabel('Density', fontsize=12)
             plt.grid(True, linestyle='--', alpha=0.7)
             plt.tight_layout()
             plt.show()
```

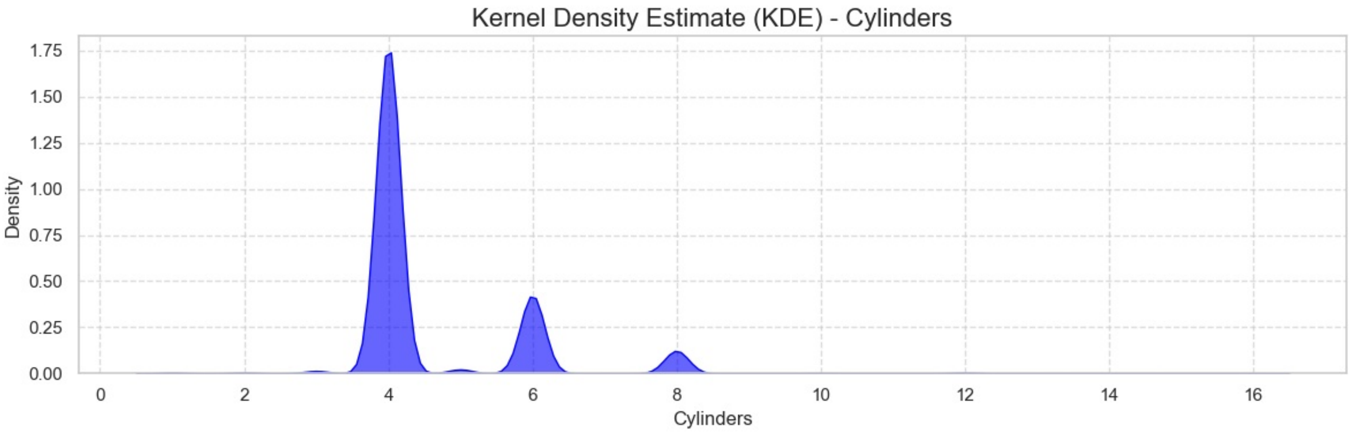


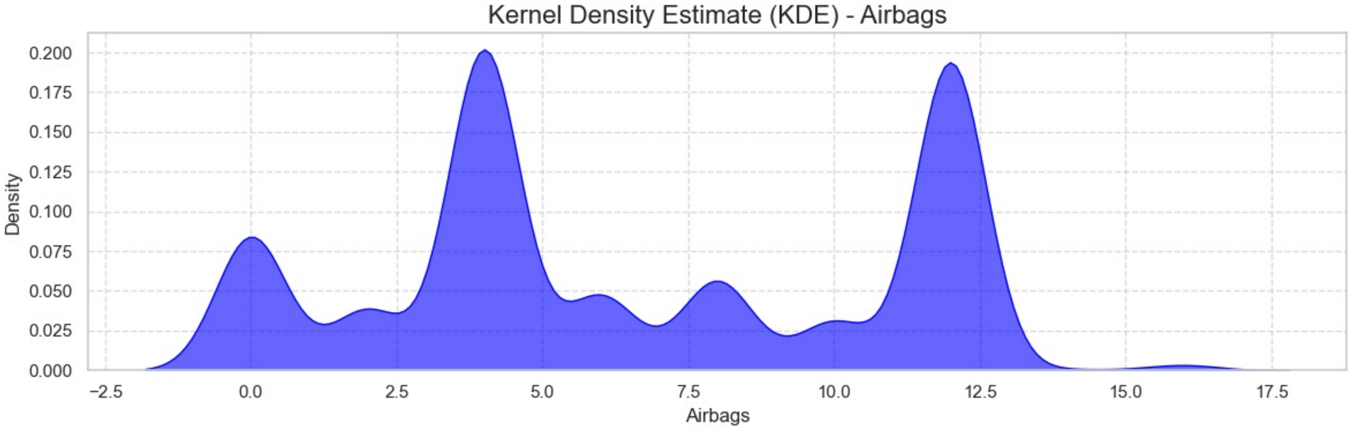

Kernel Density Estimate (KDE) - Levy
Density
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
2000
4000
6000
8000
10000
12000
Levy


Kernel Density Estimate (KDE) - Prod_year
Density
0.10
0.08
0.06
0.04
0.02
0.00
1940
1960
1980
2000
2020
Prod_year


Kernel Density Estimate (KDE) - Engine_volume
Density
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
5
10
15
20
Engine_volume

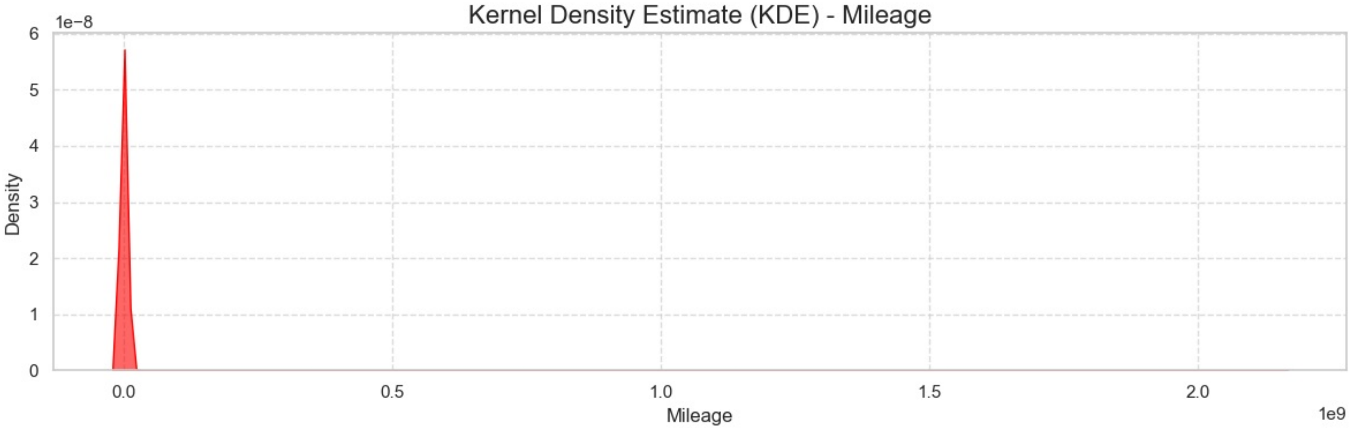

Kernel Density Estimate (KDE) - Mileage
1e−8
Density
6
5
4
3
2
1
0
0.0
0.5
1.0
1.5
2.0
1e9
Mileage





In [220…

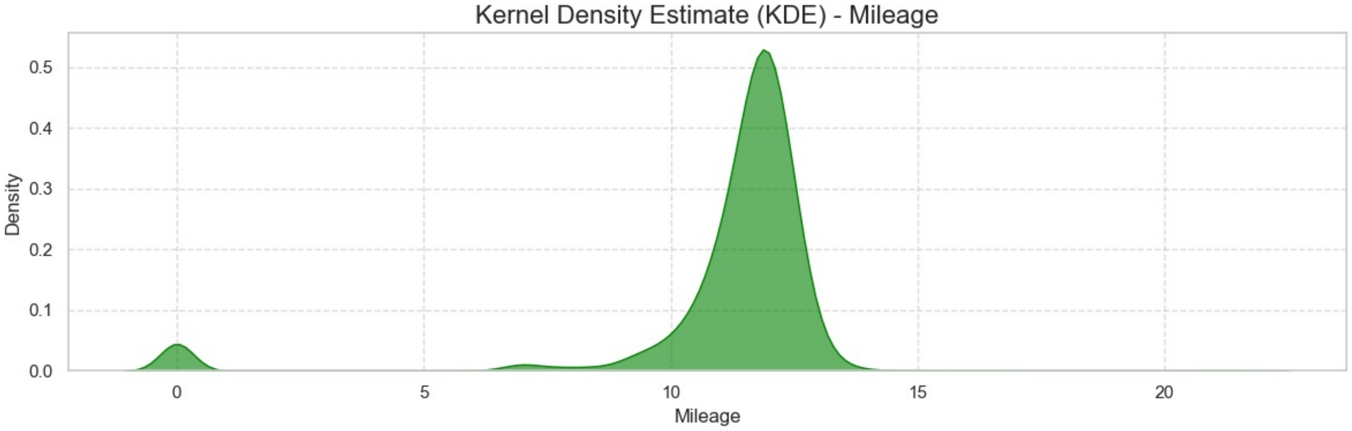
```
sns.set(style="whitegrid")
plt.figure(figsize=(12, 4))
sns.kdeplot(df['Mileage'], fill=True, color='red', alpha=0.6)
plt.title(f'Kernel Density Estimate (KDE) - {'Mileage'}', fontsize=16)
plt.xlabel('Mileage', fontsize=12)
plt.ylabel('Density', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



In [221…

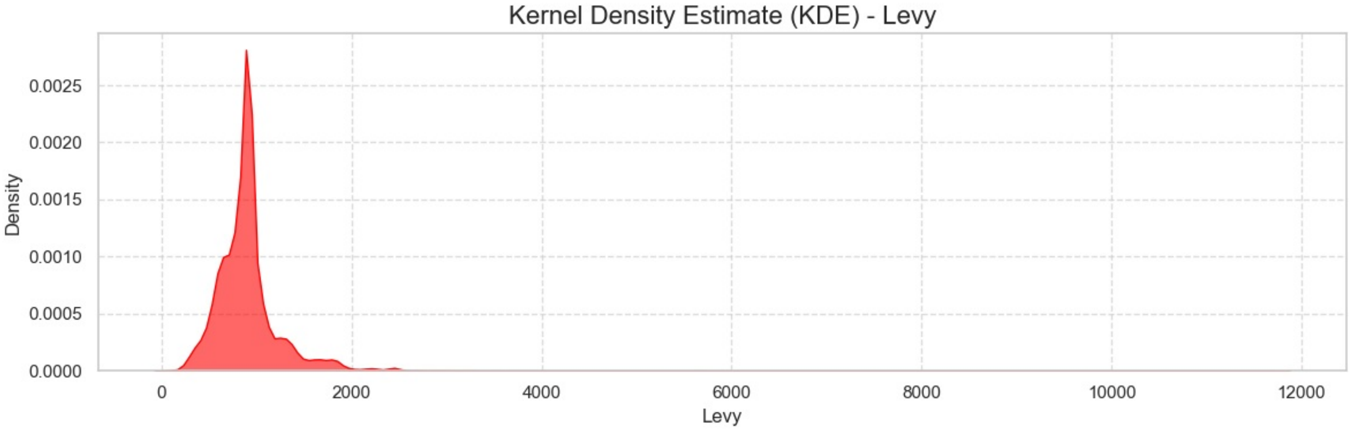
```
# log transformation

sns.set(style="whitegrid")
plt.figure(figsize=(12, 4))
sns.kdeplot(np.log(df['Mileage']).replace(-np.inf,1e-6), fill=True, color='green', alpha=0.6)
plt.title(f'Kernel Density Estimate (KDE) - {'Mileage'}', fontsize=16)
plt.xlabel('Mileage', fontsize=12)
plt.ylabel('Density', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

```
C:\Users\RPC\anaconda3\Lib\site-packages\pandas\core\arraylike.py:399: RuntimeWarning: divide by zero encountere
d in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```



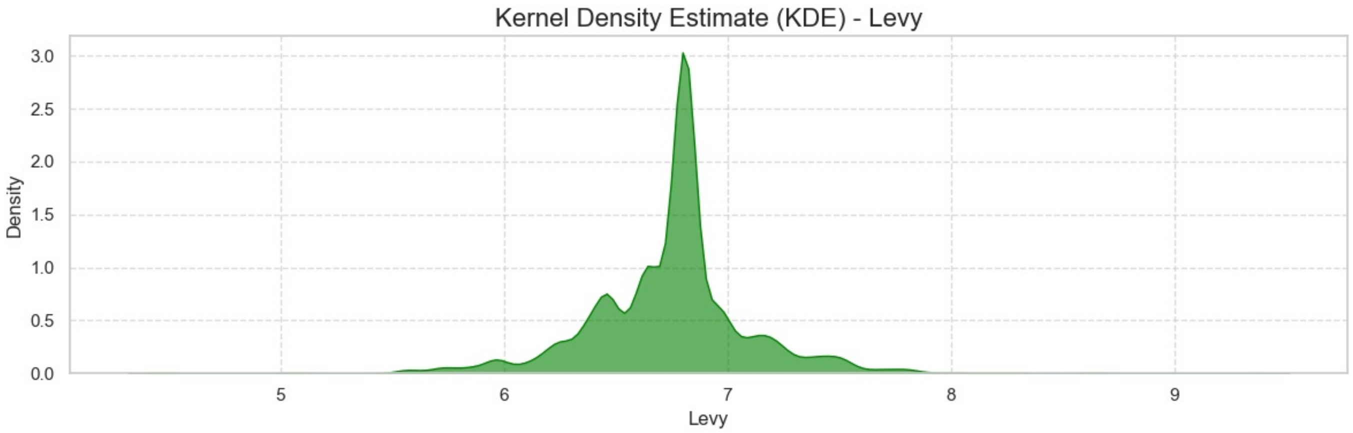
In [223…

```
sns.set(style="whitegrid")
plt.figure(figsize=(12, 4))
sns.kdeplot(df['Levy'], fill=True, color='red', alpha=0.6)
plt.title(f'Kernel Density Estimate (KDE) - {'Levy'}', fontsize=16)
plt.xlabel('Levy', fontsize=12)
plt.ylabel('Density', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



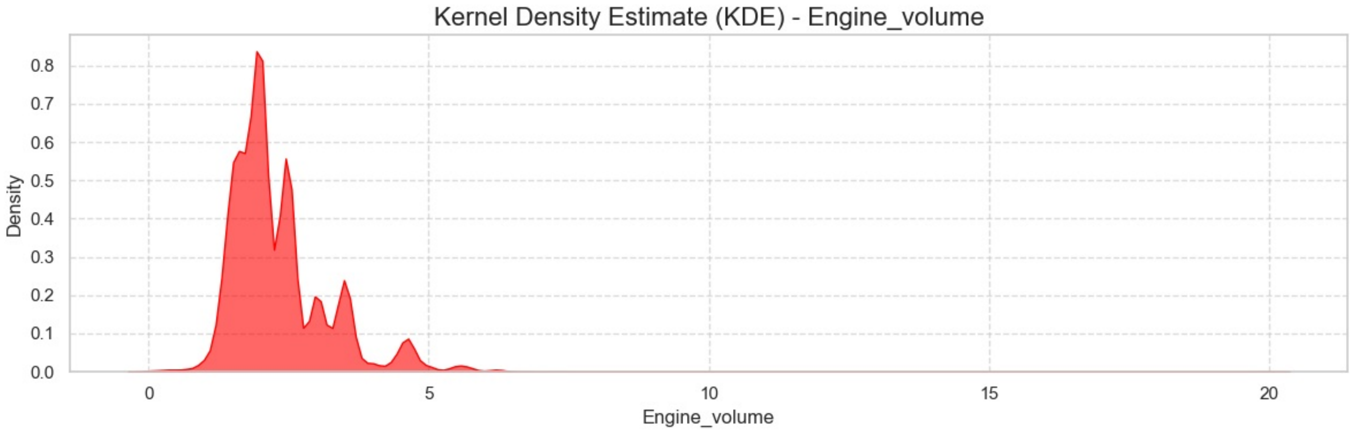
In [224…

```
# log transformation

sns.set(style="whitegrid")
plt.figure(figsize=(12, 4))
sns.kdeplot(np.log(df['Levy']).replace(-np.inf,1e-6), fill=True, color='green', alpha=0.6)
plt.title(f'Kernel Density Estimate (KDE) - {'Levy'}', fontsize=16)
plt.xlabel('Levy', fontsize=12)
plt.ylabel('Density', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



In [226…

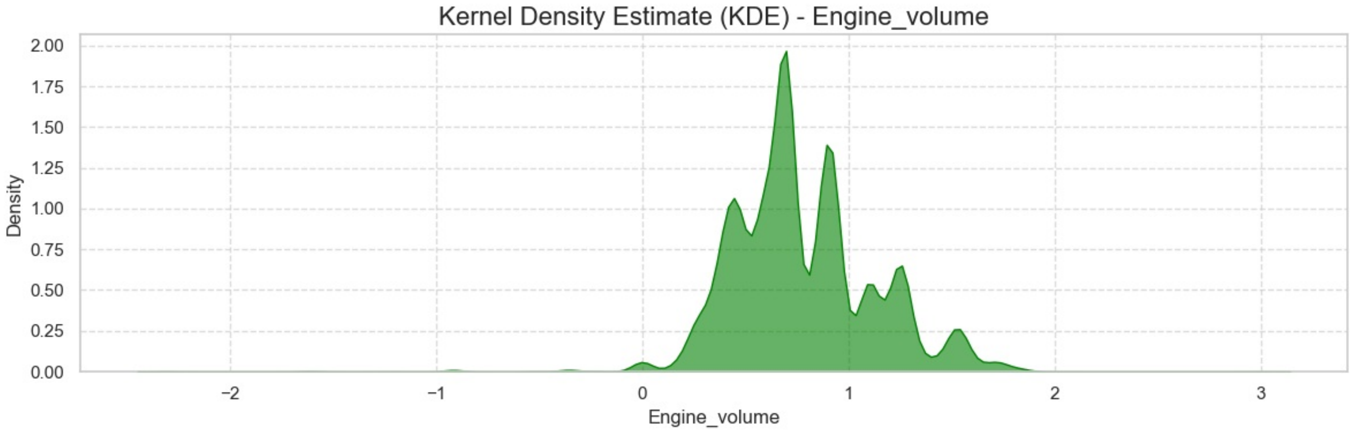
```
sns.set(style="whitegrid")
plt.figure(figsize=(12, 4))
sns.kdeplot(df['Engine_volume'], fill=True, color='red', alpha=0.6)
plt.title(f'Kernel Density Estimate (KDE) - {'Engine_volume'}', fontsize=16)
plt.xlabel('Engine_volume', fontsize=12)
plt.ylabel('Density', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



In [227…

```
# log transformation

sns.set(style="whitegrid")
plt.figure(figsize=(12, 4))
sns.kdeplot(np.log(df['Engine_volume']).replace(-np.inf,1e-6), fill=True, color='green', alpha=0.6)
plt.title(f'Kernel Density Estimate (KDE) - {'Engine_volume'}', fontsize=16)
plt.xlabel('Engine_volume', fontsize=12)
plt.ylabel('Density', fontsize=12)
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

```
C:\Users\RPC\anaconda3\Lib\site-packages\pandas\core\arraylike.py:399: RuntimeWarning: divide by zero encountere
d in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

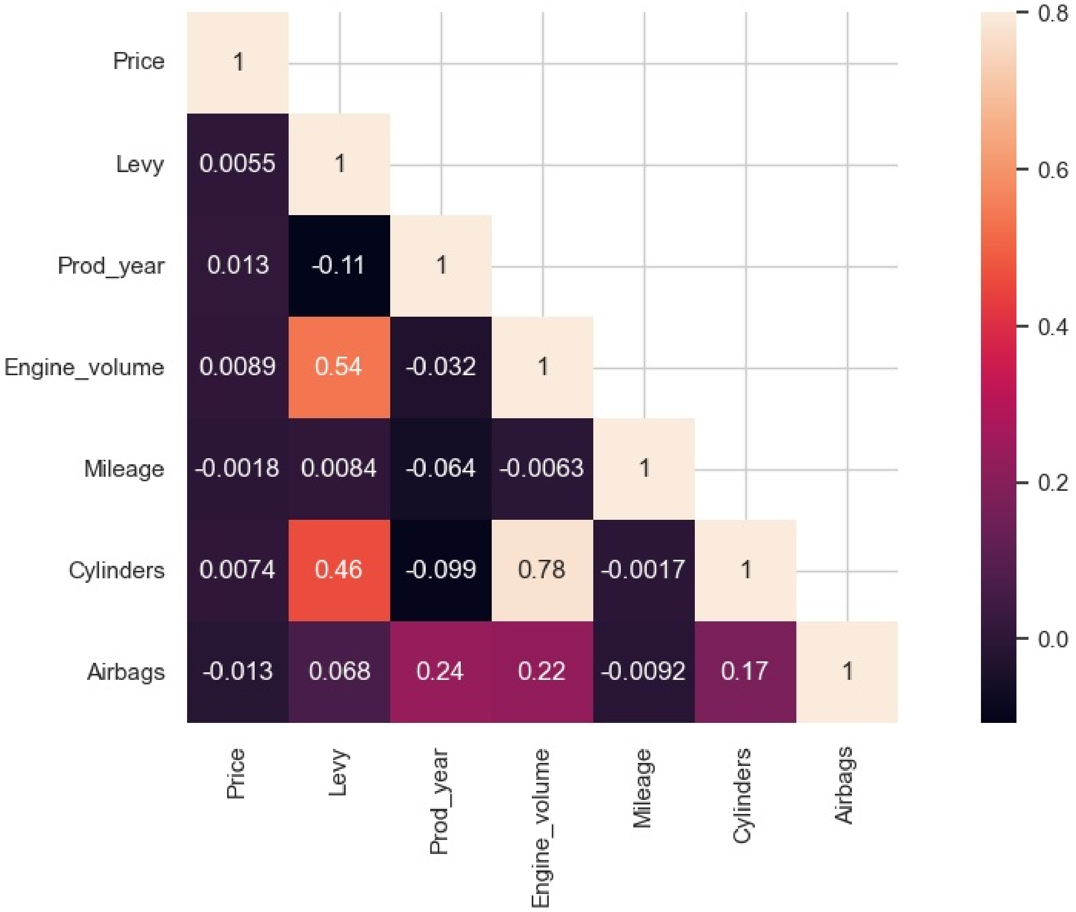


In [ ]:

Correlation

In [230…

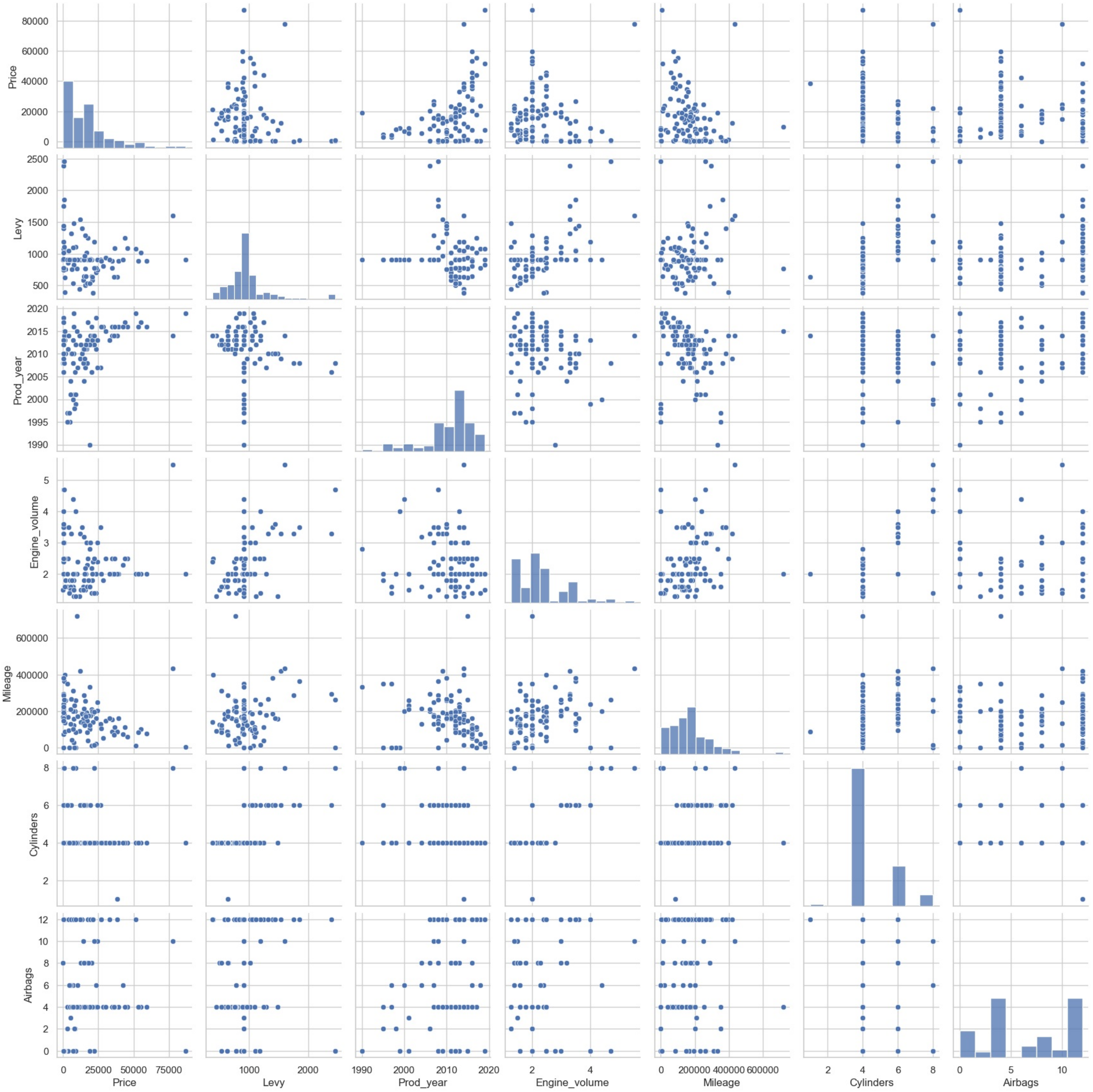
```
# sns.heatmap(df[['Price','Levy','Prod_year','Engine volume','Mileage','Cylinders','Airbags']].corr(),annot=Tru
corrMatt = df[['Price','Levy','Prod_year','Engine_volume','Mileage','Cylinders','Airbags']].corr()
mask = np.array(corrMatt)
mask[np.tril_indices_from(mask)] = False
fig,ax= plt.subplots()
fig.set_size_inches(14,6)
sns.heatmap(corrMatt, mask=mask,vmax=.8, square=True,annot=True)
plt.show()
```



Feature Interactions

In [232…

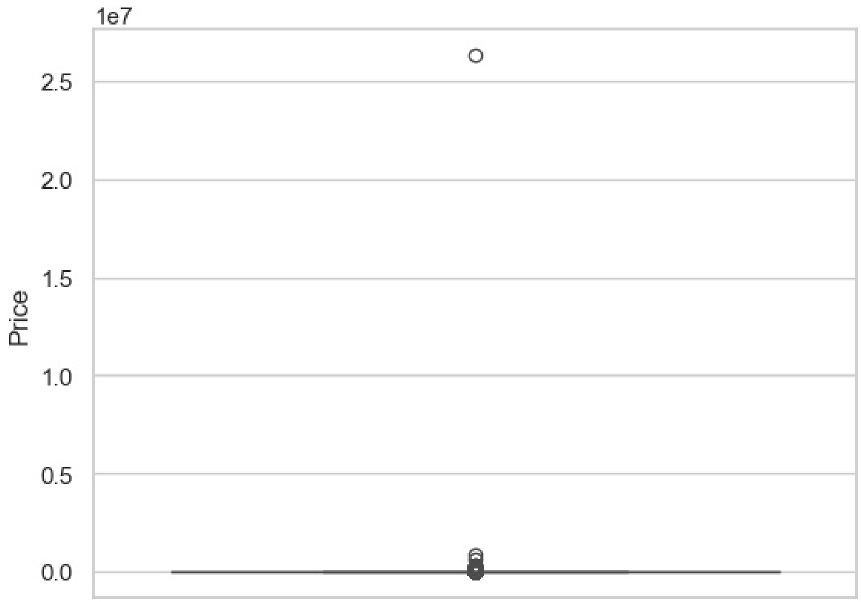
```
sns.pairplot(df[:100])
plt.show()
```



Detect Outliers

In [235…

```
sns.boxplot(df['Price'])
plt.show()
```



In [236…

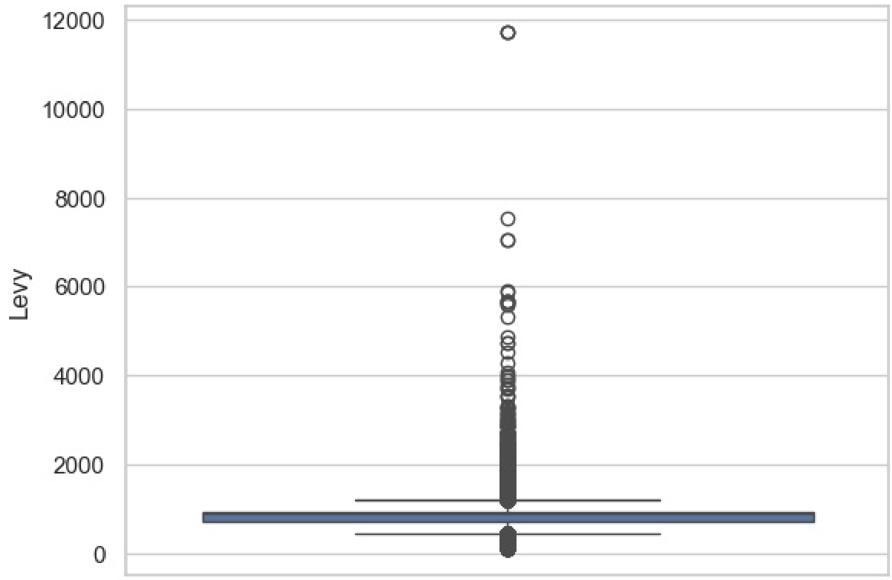
```
df[df['Price']> 5e5]
```

Out[236…

	Price	Levy	Manufacturer	Model	Prod_year	Category	Leather_interior	Fuel_type	Engine_volume	Mileage
1225	627220	906.299205	MERCEDES-BENZ	G 65 AMG 63AMG	2020	Jeep	Yes	Petrol	6.3	0
8541	872946	2067.000000	LAMBORGHINI	Urus	2019	Universal	Yes	Petrol	4.0	2531
16983	26307500	906.299205	OPEL	Combo	1999	Goods wagon	No	Diesel	1.7	99999

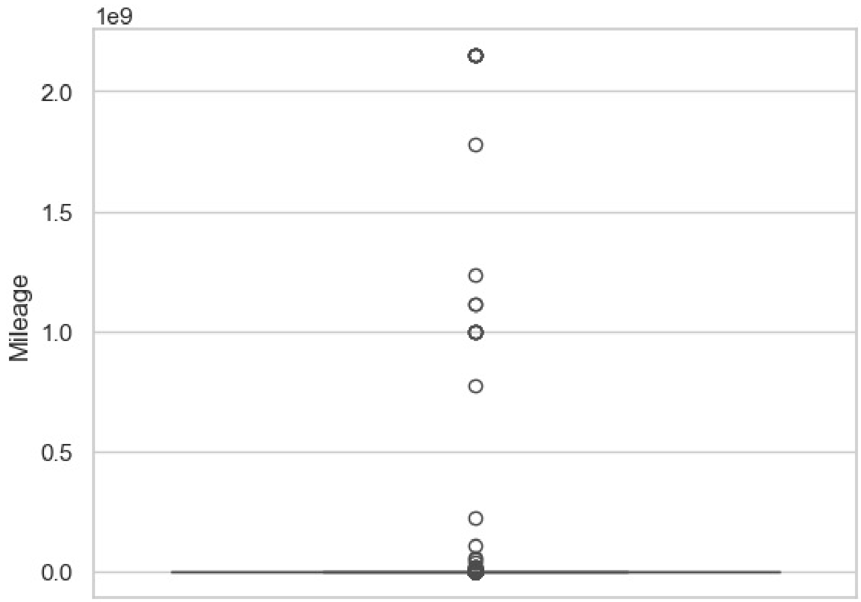
In [237…

```
sns.boxplot(df['Levy'])
plt.show()
```



In [238…

```
sns.boxplot(df['Mileage'])
plt.show()
```



In [239…

```
numerical_data=df[['Price','Levy','Engine_volume','Mileage','Cylinders','Airbags']]
for column in numerical_data.columns:
    Q1=numerical_data[column].quantile(0.25)
    Q3=numerical_data[column].quantile(0.75)
    IQR = Q3-Q1

    Lower_bound = Q1 - 1.5*IQR
    Upper_bound = Q3 + 1.5*IQR

    outliers = ((numerical_data[column]>Upper_bound)|(numerical_data[column]<Lower_bound)).sum()
    Total = numerical_data[column].shape[0]
    print(f'Total of outliers in {column} are   :   {outliers}--{round(100*(outliers)/Total,2)}%')

    if outliers > 0:
        df=df.loc[(df[column] <= Upper_bound) & (df[column] >= Lower_bound)]
```

```
Total of outliers in Price are   :   1055--5.57%
Total of outliers in Levy are   :   3103--16.4%
Total of outliers in Engine_volume are   :   1358--7.18%
Total of outliers in Mileage are   :   635--3.36%
Total of outliers in Cylinders are   :   4765--25.18%
Total of outliers in Airbags are   :   0--0.0%
```

In [240…

```
# def outliers(df, col):
#     Q1 = df[col].quantile(0.25)
#     Q3 = df[col].quantile(0.75)
#     IQR = Q3-Q1

#     lower_bound= Q1-1.5 * IQR
#     upper_bound= Q1+1.5 * IQR

#     df_no_outliers = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
#     return df_no_outliers

# df= outliers(df,'Price')
# df= outliers(df,'Mileage')
# df= outliers(df,'Levy')
# df= outliers(df,'Engine_volume')
# df= outliers(df,'Cylinders')
# df= outliers(df,'Airbags')
```

In [241…

```
df.shape
```

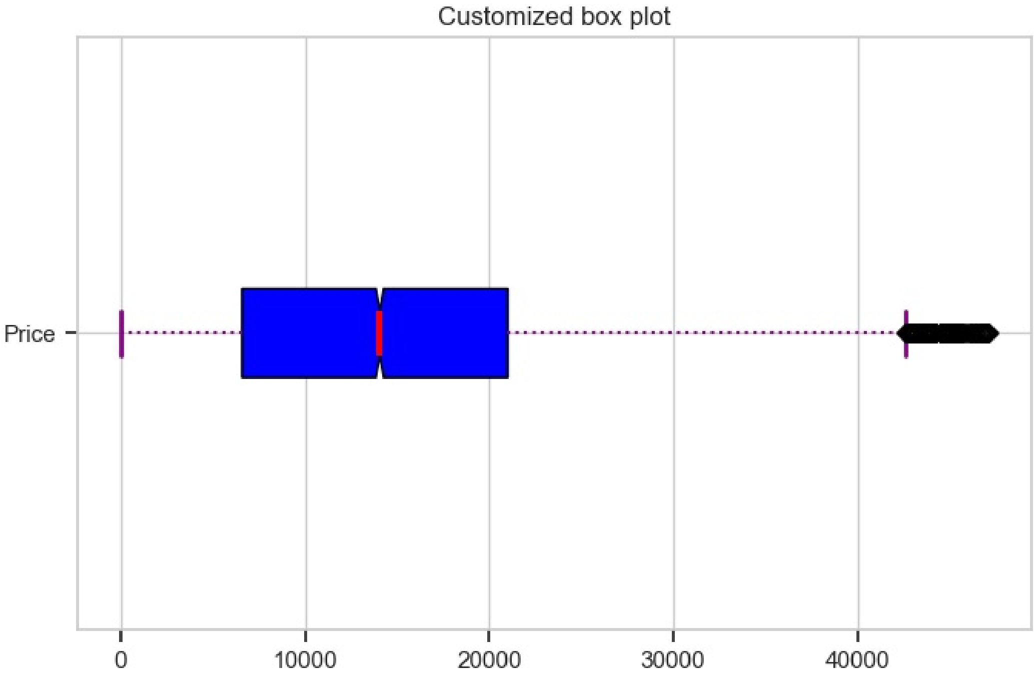
Out[241…

```
(11520, 16)
```

In [242…

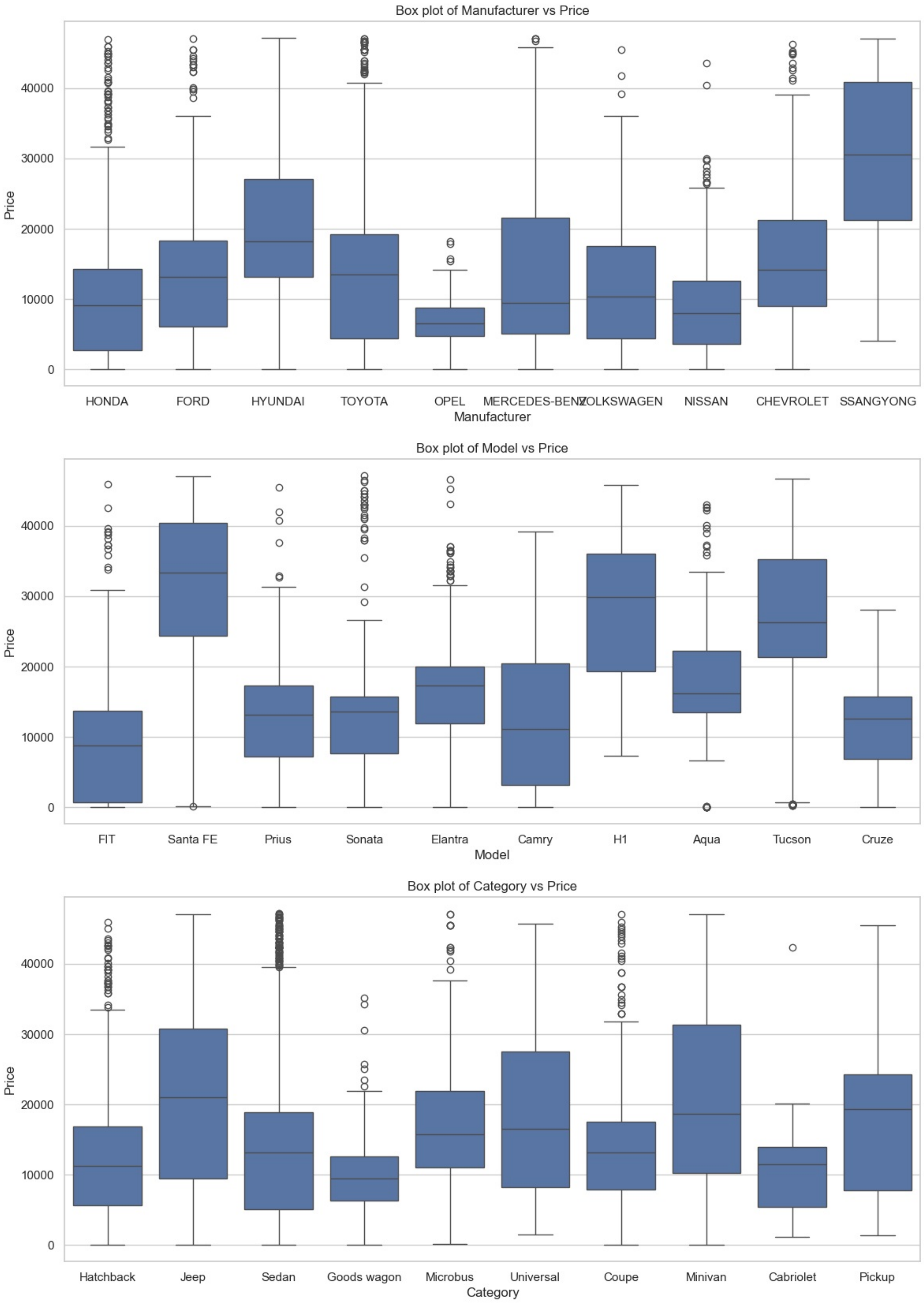
```
# Import libraries
import matplotlib.pyplot as plt
import numpy as np

# Creating dataset
data = df['Price']

fig = plt.figure(figsize =(8, 5))
ax = fig.add_subplot(111)

# Creating axes instance
bp = ax.boxplot(data, patch_artist = True,
                notch ='True', vert = 0)
```

```
colors = ['#0000FF']

for patch, color in zip(bp['boxes'], colors):
    patch.set_facecolor(color)

# changing color and linewidth of
# whiskers
for whisker in bp['whiskers']:
    whisker.set(color ='#8B008B',
                linewidth = 1.5,
                linestyle =":")

# changing color and linewidth of
# caps
for cap in bp['caps']:
    cap.set(color ='#8B008B',
            linewidth = 2)

# changing color and linewidth of
# medians
for median in bp['medians']:
    median.set(color ='red',
               linewidth = 3)

# changing style of fliers
for flier in bp['fliers']:
    flier.set(marker ='D',
              color ='#e7298a',
              alpha = 0.5)

# x-axis labels
ax.set_yticklabels(['Price'])

# Adding title
plt.title("Customized box plot")

# Removing top axes and right axes
# ticks
ax.get_xaxis().tick_bottom()
ax.get_yaxis().tick_left()

# show plot
plt.show()
```



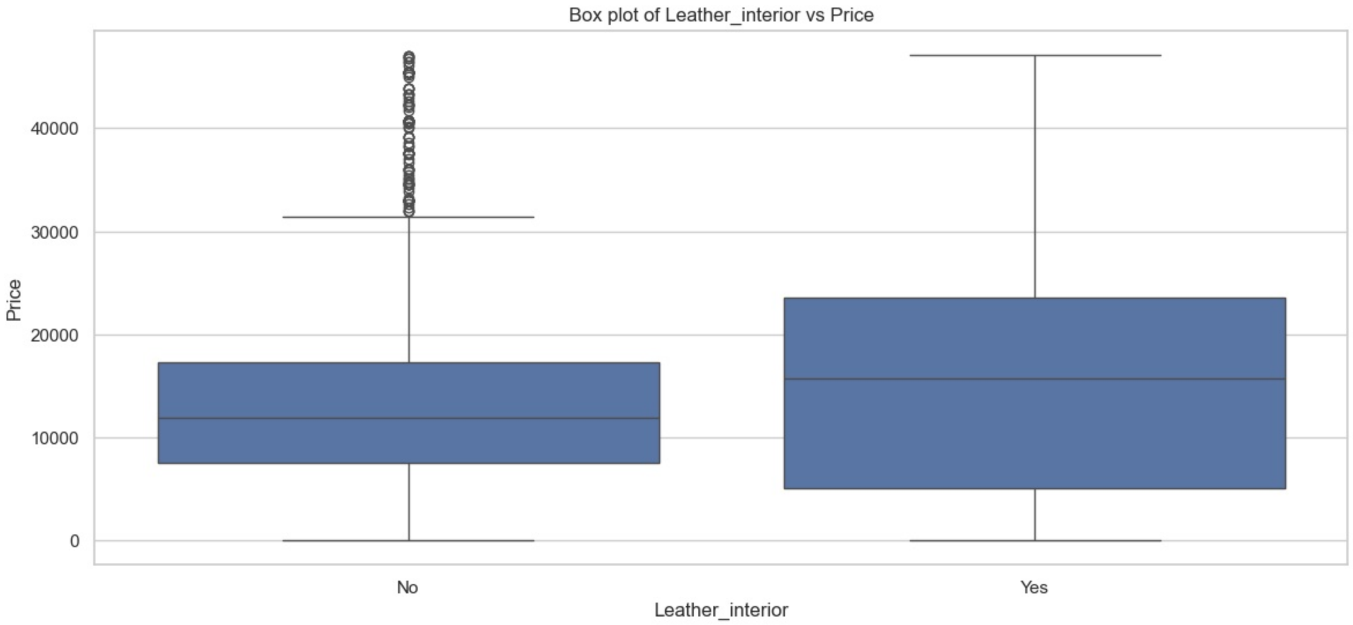
In [243…

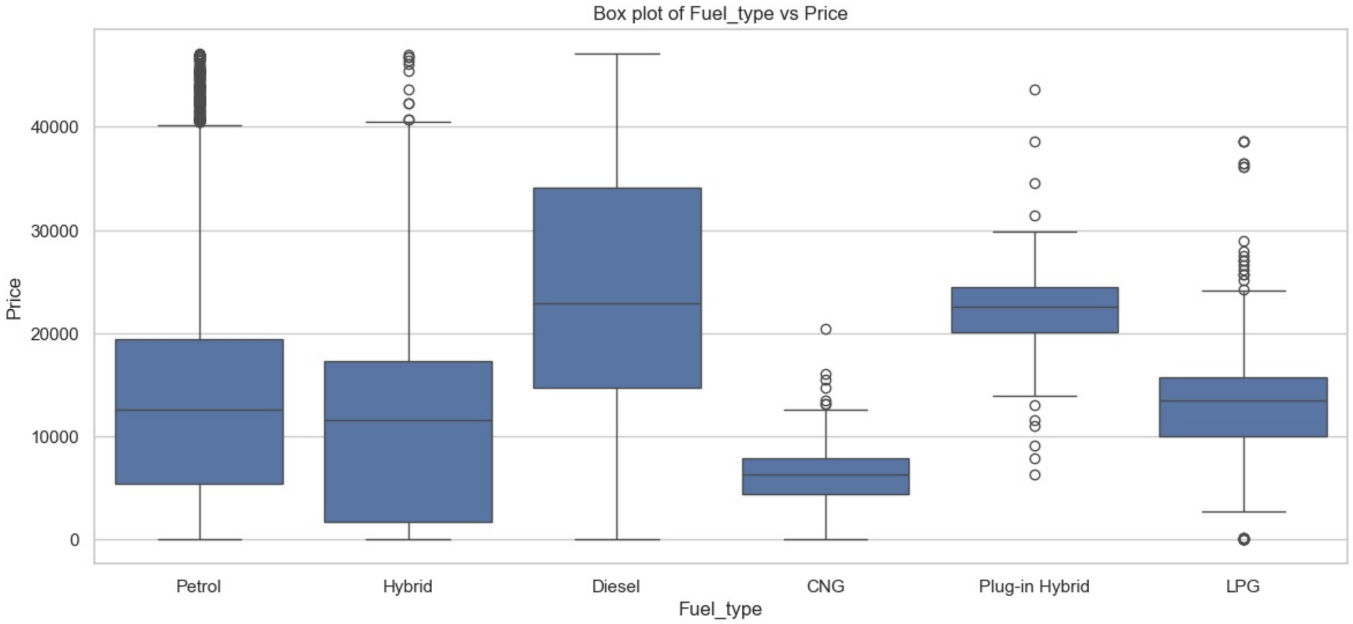
```
# Target Variable Analysis
# Relationship with Predictors (scatter plots, box Plots against the target)

for col in df.select_dtypes('object'):
    top_10_categs = df[col].value_counts().index[:10]
    filtered_df = df[df[col].isin(top_10_categs)]

    plt.figure(figsize=(14,6))
    sns.boxplot(x=filtered_df[col], y=filtered_df['Price'])
    plt.title(f'Box plot of {col} vs Price')
    plt.show()
```

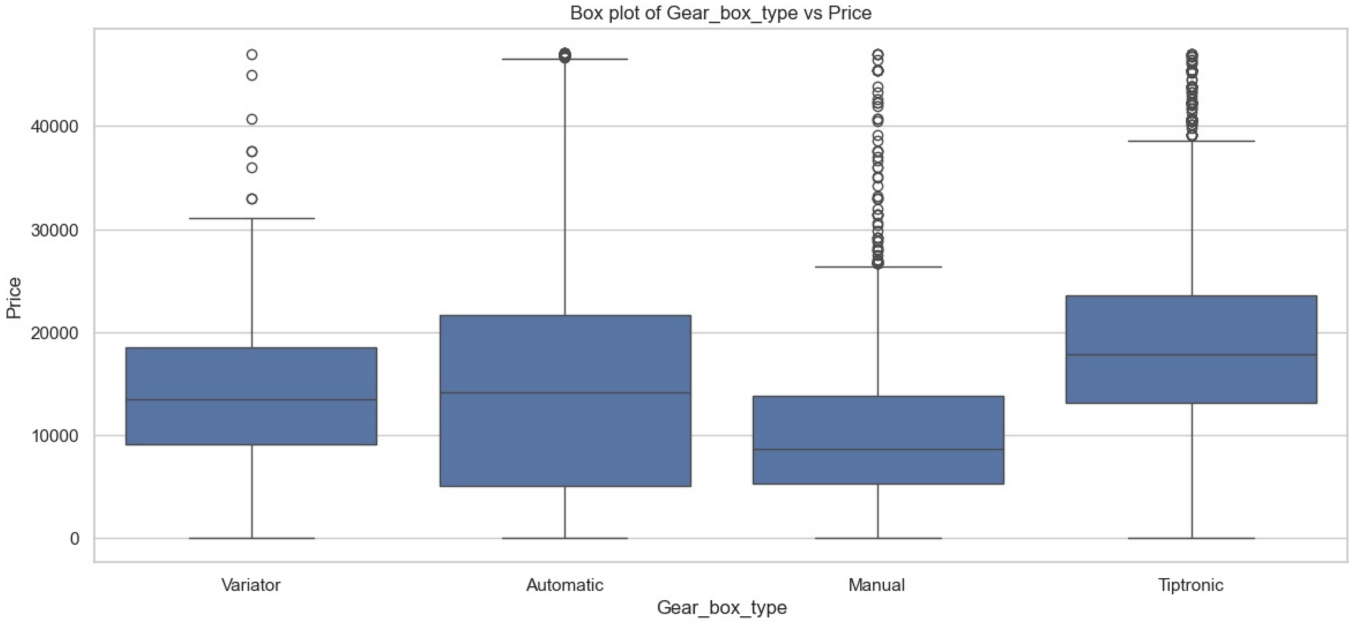

Box plot of Manufacturer vs Price

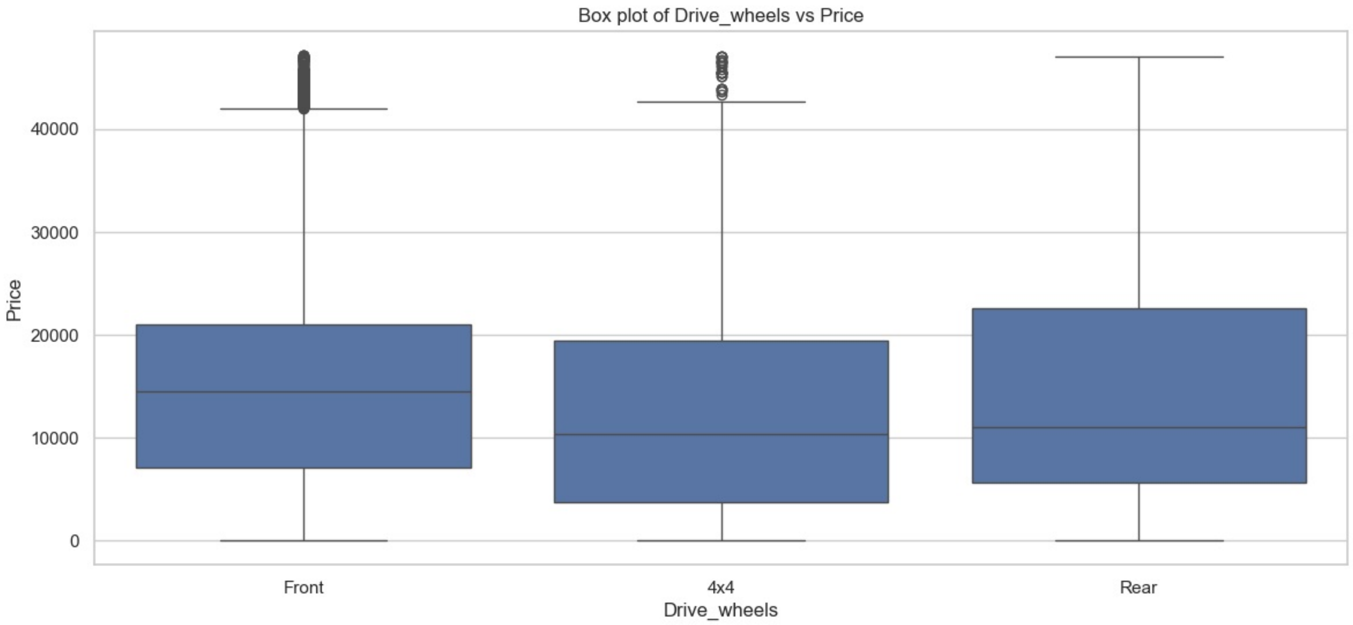
Box plot of Model vs Price

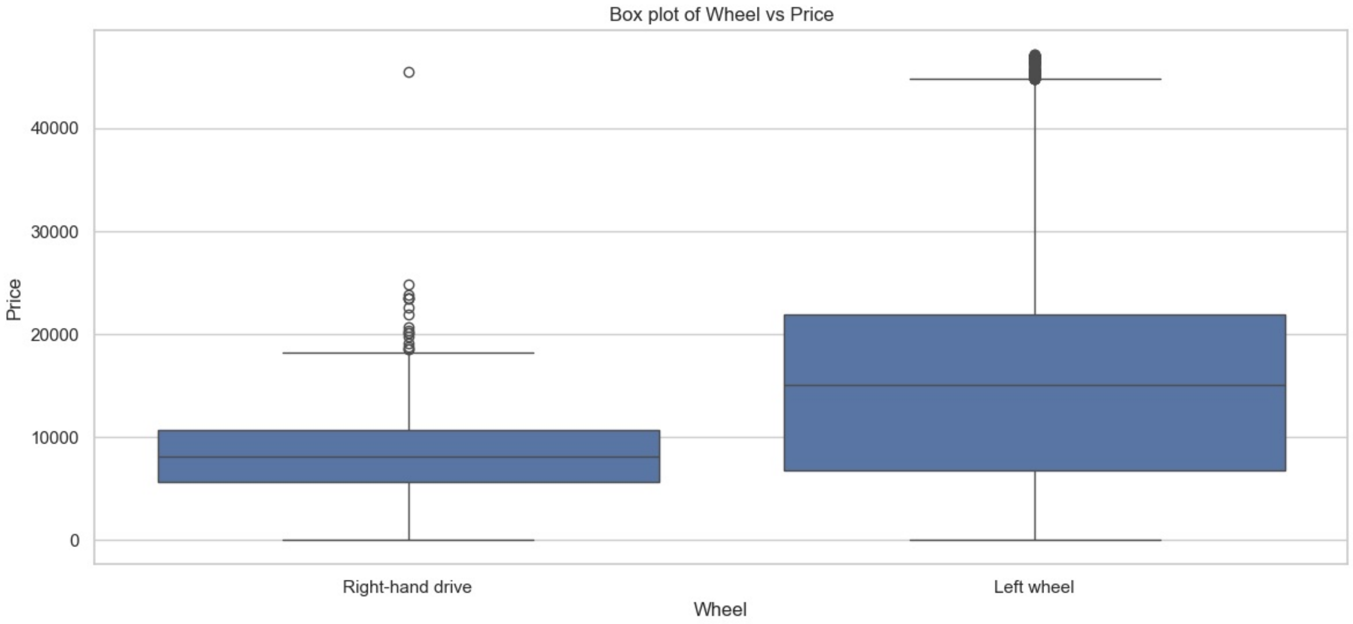
Box plot of Category vs Price

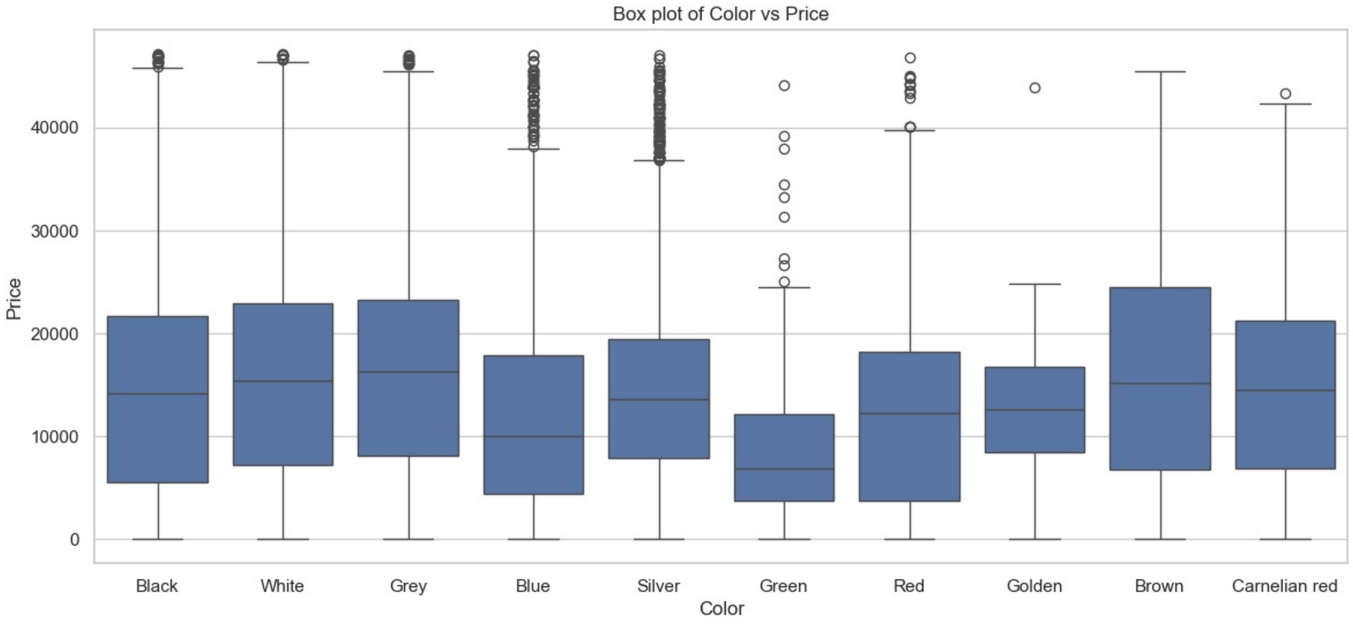

Box plot of Leather_interior vs Price
Price
0
10000
20000
30000
40000
No
Yes
Leather_interior


Box plot of Fuel_type vs Price
Price
0
10000
20000
30000
40000
Petrol
Hybrid
Diesel
CNG
Plug-in Hybrid
LPG
Fuel_type


Box plot of Gear_box_type vs Price
Price
0
10000
20000
30000
40000
Variator
Automatic
Manual
Tiptronic
Gear_box_type

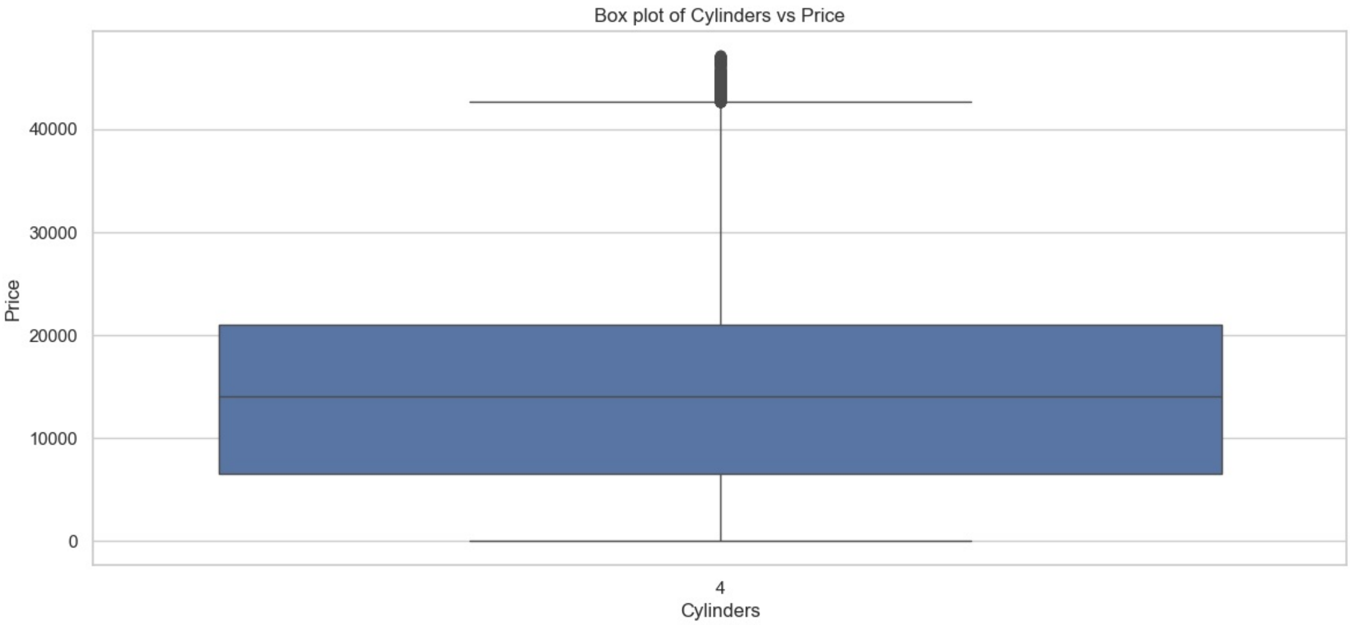




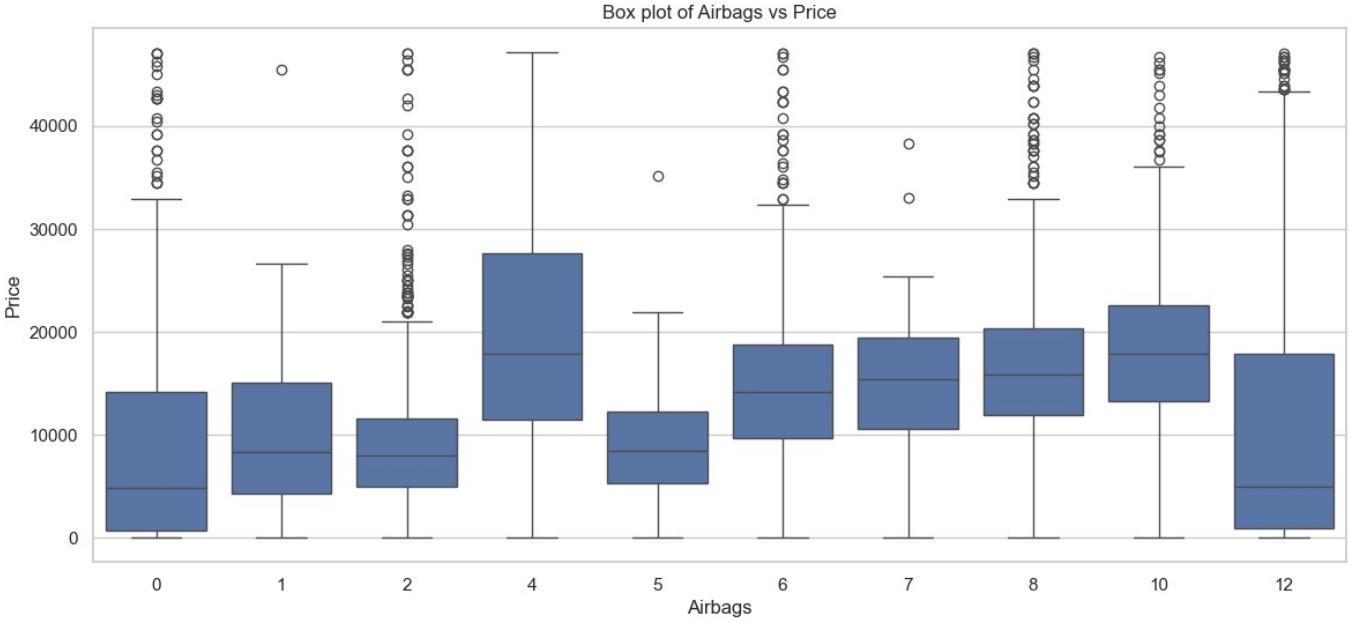


In [244…

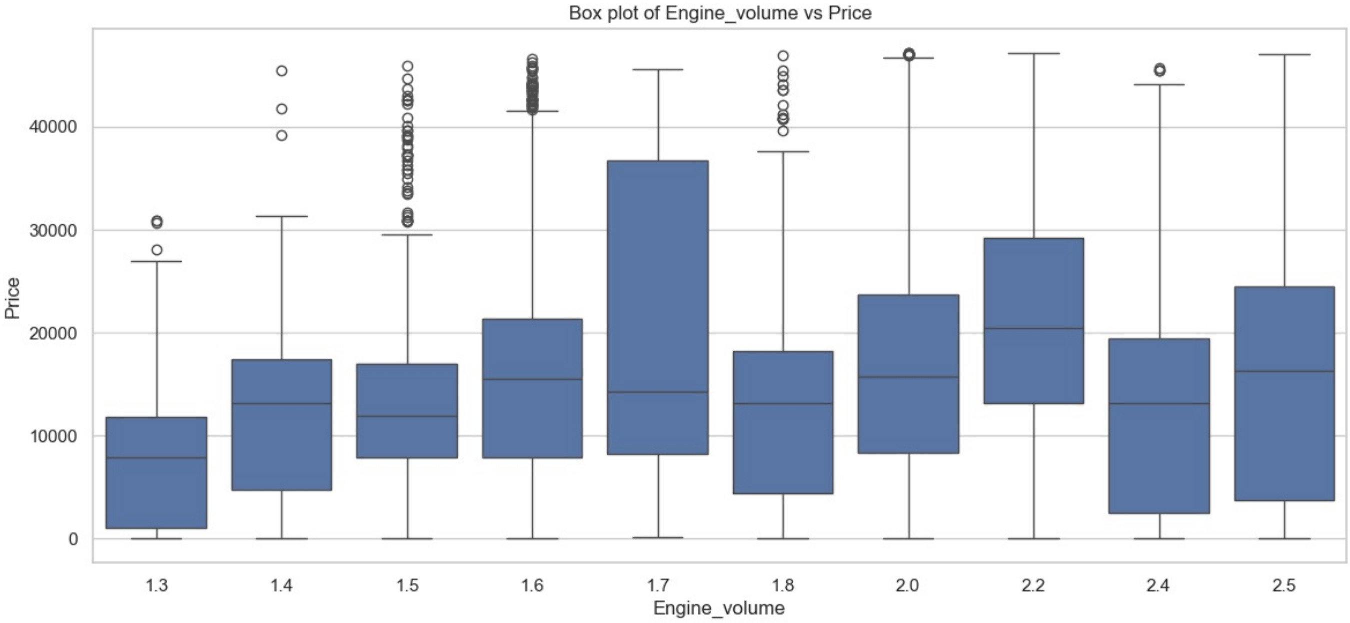
```
for col in ['Cylinders','Airbags','Engine_volume']:
    top_10_categs = df[col].value_counts().index[:10]
    filtered_df = df[df[col].isin(top_10_categs)]

    plt.figure(figsize=(14,6))
    sns.boxplot(x=filtered_df[col], y=filtered_df['Price'])
    plt.title(f'Box plot of {col} vs Price')
    plt.show()
```



Box plot of Cylinders vs Price



Box plot of Airbags vs Price



Box plot of Engine_volume vs Price

Feature Extraction

In [247…

```
# Date
from datetime import datetime
dtime=datetime.now()
```

```
# calcul age of cars
df['Age_of_Car']=dtime.year-df['Prod_year']
```

In [248…

```
# df = df.drop(columns=['Prod_year'],axis=1)
```

In [249…

```
df[['Age_of_Car','Prod_year']]
```

Out[249…

	Age_of_Car	Prod_year
2	19	2006
3	14	2011
5	9	2016
6	15	2010
7	12	2013
...	...	...
19230	14	2011
19232	26	1999
19233	14	2011
19234	15	2010
19236	13	2012

11520 rows × 2 columns

Transform Data

In [252…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 11520 entries, 2 to 19236
Data columns (total 17 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Price             11520 non-null  int64  
 1   Levy              11520 non-null  float64
 2   Manufacturer      11520 non-null  object 
 3   Model             11520 non-null  object 
 4   Prod_year         11520 non-null  int64  
 5   Category          11520 non-null  object 
 6   Leather_interior  11520 non-null  object 
 7   Fuel_type         11520 non-null  object 
 8   Engine_volume     11520 non-null  float64
 9   Mileage           11520 non-null  int64  
 10  Cylinders         11520 non-null  int64  
 11  Gear_box_type     11520 non-null  object 
 12  Drive_wheels      11520 non-null  object 
 13  Wheel             11520 non-null  object 
 14  Color             11520 non-null  object 
 15  Airbags           11520 non-null  int64  
 16  Age_of_Car        11520 non-null  int64  
dtypes: float64(2), int64(6), object(9)
memory usage: 1.6+ MB
```

In [253…

```
df_object = df.select_dtypes('object')
df_non_object = df.select_dtypes('number')
```

In [254…

```
def number_unique_columns(data):
    for i in data.columns:
        print(f'{i} : {data[i].nunique()}')
```

In [255…

```
number_unique_columns(df_object)
```

```
Manufacturer : 55
Model : 953
Category : 11
Leather_interior : 2
Fuel_type : 6
Gear_box_type : 4
Drive_wheels : 3
Wheel : 2
Color : 16
```

In [257…

```
# for label encoding
from sklearn.preprocessing import LabelEncoder

df_object_for_LB = df_object[['Manufacturer','Model','Category','Fuel_type','Color','Leather_interior','Wheel']]

LabelEncoders = {}
for col in df_object_for_LB:
    label = LabelEncoder()
    df_object_for_LB[col]=label.fit_transform(df_object_for_LB[col])
    LabelEncoders[col] = label
```

```
C:\Users\RPC\AppData\Local\Temp\ipykernel_11504\2837316513.py:9: 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#retu
rning-a-view-versus-a-copy
  df_object_for_LB[col]=label.fit_transform(df_object_for_LB[col])
C:\Users\RPC\AppData\Local\Temp\ipykernel_11504\2837316513.py:9: 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#retu
rning-a-view-versus-a-copy
  df_object_for_LB[col]=label.fit_transform(df_object_for_LB[col])
C:\Users\RPC\AppData\Local\Temp\ipykernel_11504\2837316513.py:9: 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#retu
rning-a-view-versus-a-copy
  df_object_for_LB[col]=label.fit_transform(df_object_for_LB[col])
C:\Users\RPC\AppData\Local\Temp\ipykernel_11504\2837316513.py:9: 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#retu
rning-a-view-versus-a-copy
  df_object_for_LB[col]=label.fit_transform(df_object_for_LB[col])
C:\Users\RPC\AppData\Local\Temp\ipykernel_11504\2837316513.py:9: 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#retu
rning-a-view-versus-a-copy
  df_object_for_LB[col]=label.fit_transform(df_object_for_LB[col])
C:\Users\RPC\AppData\Local\Temp\ipykernel_11504\2837316513.py:9: 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#retu
rning-a-view-versus-a-copy
  df_object_for_LB[col]=label.fit_transform(df_object_for_LB[col])
C:\Users\RPC\AppData\Local\Temp\ipykernel_11504\2837316513.py:9: 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#retu
rning-a-view-versus-a-copy
  df_object_for_LB[col]=label.fit_transform(df_object_for_LB[col])
```

In [258…

```
LabelEncoders
```

Out[258…

```
{'Manufacturer': LabelEncoder(),
 'Model': LabelEncoder(),
 'Category': LabelEncoder(),
 'Fuel_type': LabelEncoder(),
 'Color': LabelEncoder(),
 'Leather_interior': LabelEncoder(),
 'Wheel': LabelEncoder()}
```

In [259…

```
# mapping
mapping = {category : index for index, category in enumerate(LabelEncoders['Category'].classes_)}
print(mapping)
```

```
{'Cabriolet': 0, 'Coupe': 1, 'Goods wagon': 2, 'Hatchback': 3, 'Jeep': 4, 'Limousine': 5, 'Microbus': 6, 'Miniva
n': 7, 'Pickup': 8, 'Sedan': 9, 'Universal': 10}
```

In [260…

```
# Save Label encoder for using
import pickle
with open('label_encoders.pkl','wb') as f :
    pickle.dump(LabelEncoders, f)
```

In [262…

```
# for one hot encoding

from sklearn.preprocessing import OneHotEncoder

categorical_cols = df_object[['Gear_box_type', 'Drive_wheels']].columns

ohe = OneHotEncoder(sparse_output=False, handle_unknown='ignore')
one_hot_encoded = ohe.fit_transform(df_object[categorical_cols])
one_hot_columns = ohe.get_feature_names_out(categorical_cols)
df_ohe = pd.DataFrame(one_hot_encoded, columns=one_hot_columns, index=df.index)
df_for_ohe = df_object.drop(columns=categorical_cols).join(df_ohe)

df_for_ohe = df_for_ohe.drop(['Manufacturer','Model','Category','Fuel_type','Color','Leather_interior', 'Wheel'
```

In [263…

```
df_for_ohe.head()
```

Out[263…

	Gear_box_type_Automatic	Gear_box_type_Manual	Gear_box_type_Tiptronic	Gear_box_type_Variator	Drive_wheels_4x4	Drive_wh
2	0.0	0.0	0.0	1.0	0.0	
3	1.0	0.0	0.0	0.0	1.0	
5	1.0	0.0	0.0	0.0	0.0	
6	1.0	0.0	0.0	0.0	0.0	
7	1.0	0.0	0.0	0.0	0.0	

In [264…

```
df_for_ohe.shape
```

Out[264…

```
(11520, 7)
```

In [265…

```
# save one hot encoder

import pickle
with open('One_Hot_Encoder.pkl', 'wb') as f:
    pickle.dump(ohe,f)
```

In [267…

```
df = pd.concat([df_non_object, df_object_for_LB, df_for_ohe],axis=1)
```

In [268…

```
df.head()
```

Out[268…

	Price	Levy	Prod_year	Engine_volume	Mileage	Cylinders	Airbags	Age_of_Car	Manufacturer	Model	...	Color	Leath
2	8467	906.299205	2006	1.3	200000	4	2	19	17	412	...	1	
3	3607	862.000000	2011	2.5	168966	4	0	14	13	397	...	14	
5	39493	891.000000	2016	2.0	160931	4	4	9	18	761	...	14	
6	1803	761.000000	2010	1.8	258909	4	12	15	48	694	...	14	
7	549	751.000000	2013	2.4	216118	4	12	12	18	782	...	7	

5 rows × 22 columns

In [269…

```
df.shape
```

Out[269…

```
(11520, 22)
```

In [270…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 11520 entries, 2 to 19236
Data columns (total 22 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Price                   11520 non-null  int64  
 1   Levy                    11520 non-null  float64
 2   Prod_year               11520 non-null  int64  
 3   Engine_volume           11520 non-null  float64
 4   Mileage                 11520 non-null  int64  
 5   Cylinders               11520 non-null  int64  
 6   Airbags                 11520 non-null  int64  
 7   Age_of_Car              11520 non-null  int64  
 8   Manufacturer            11520 non-null  int32  
 9   Model                   11520 non-null  int32  
 10  Category                11520 non-null  int32  
 11  Fuel_type               11520 non-null  int32  
 12  Color                   11520 non-null  int32  
 13  Leather_interior        11520 non-null  int32  
 14  Wheel                   11520 non-null  int32  
 15  Gear_box_type_Automatic  11520 non-null  float64
 16  Gear_box_type_Manual     11520 non-null  float64
 17  Gear_box_type_Tiptronic  11520 non-null  float64
 18  Gear_box_type_Variator   11520 non-null  float64
 19  Drive_wheels_4x4         11520 non-null  float64
 20  Drive_wheels_Front       11520 non-null  float64
 21  Drive_wheels_Rear        11520 non-null  float64
dtypes: float64(9), int32(7), int64(6)
memory usage: 1.7 MB
```

Model

Spliting Data

In [275…

```python
from sklearn.model_selection import train_test_split

x = df.drop('Price',axis=1)
y = df['Price']

x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.15,random_state=1234)
```

In [276…

```python
print(f'x_train : {x_train.shape}')
print(f'x_test : {x_test.shape}')
print('--------------------------')
print(f'y_train : {y_train.shape}')
print(f'y_test : {y_test.shape}')
```

```
x_train : (9792, 21)
x_test : (1728, 21)
--------------------------
y_train : (9792,)
y_test : (1728,)
```

In [277…

```python
x_train.columns
```

Out[277…

```
Index(['Levy', 'Prod_year', 'Engine_volume', 'Mileage', 'Cylinders', 'Airbags',
       'Age_of_Car', 'Manufacturer', 'Model', 'Category', 'Fuel_type', 'Color',
       'Leather_interior', 'Wheel', 'Gear_box_type_Automatic',
       'Gear_box_type_Manual', 'Gear_box_type_Tiptronic',
       'Gear_box_type_Variator', 'Drive_wheels_4x4', 'Drive_wheels_Front',
       'Drive_wheels_Rear'],
      dtype='object')
```

Standard Scaling

In [280…

```python
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

x_train[['Levy','Engine_volume','Mileage','Age_of_Car']] = scaler.fit_transform(x_train[['Levy','Engine_volume'
x_test[['Levy','Engine_volume','Mileage','Age_of_Car']] = scaler.fit_transform(x_test[['Levy','Engine_volume','
```

In [281…

```python
# saving scaling
```

```
import pickle
with open('scaler.pkl' , 'wb') as file :
    pickle.dump(scaler, file)
```

Creating Model

In [284…

```
from xgboost import XGBRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor

from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
import math

r_2=[]
rmse=[]
mae=[]

def reg(model):
    model.fit(x_train,y_train)
    pred = model.predict(x_test)

    R2 = r2_score(y_test,pred)
    RMSE = math.sqrt(mean_squared_error(y_test,pred))
    MAE = mean_absolute_error(y_test,pred)

    r_2.append(R2)
    rmse.append(RMSE)
    mae.append(MAE)
```

In [285…

```
XGBRegressor_model = XGBRegressor()
RandomForestRegressor_model = RandomForestRegressor()
DecisionTreeRegressor_model = DecisionTreeRegressor()
GradientBoostingRegressor_model = GradientBoostingRegressor()
```

In [286…

```
reg(XGBRegressor_model)
reg(RandomForestRegressor_model)
reg(DecisionTreeRegressor_model)
reg(GradientBoostingRegressor_model)
```

In [287…

```
Algorithms = ['RandomForestRegressor','XGBRegressor','DecisionTreeRegressor','GradientBoostingRegressor']
```

In [288…

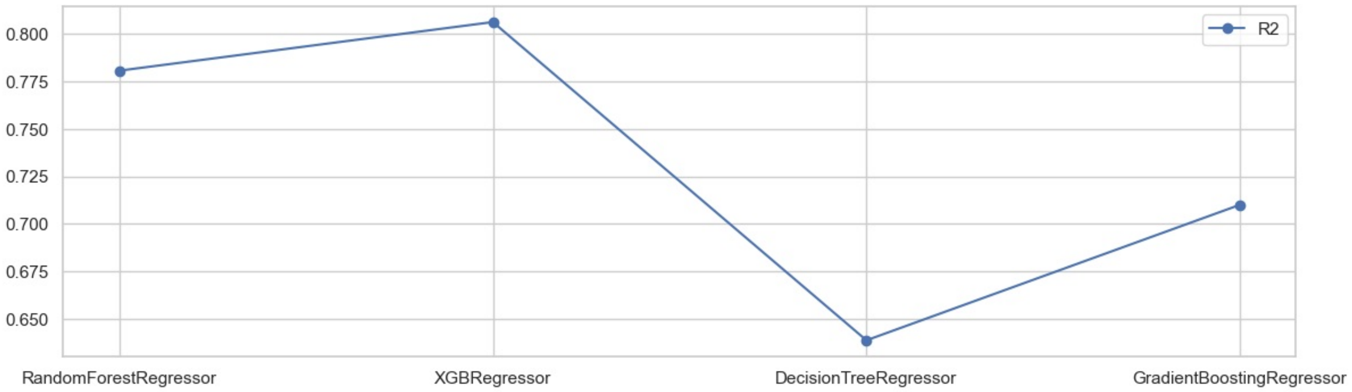
```
result=pd.DataFrame({'Algorithms':Algorithms,'R2':r_2,'rmse':rmse,'mae':mae})
result
```

Out[288…

	Algorithms	R2	rmse	mae
0	RandomForestRegressor	0.780724	5288.389925	3628.887207
1	XGBRegressor	0.806331	4970.024696	3242.808946
2	DecisionTreeRegressor	0.638656	6788.733567	4079.007207
3	GradientBoostingRegressor	0.710062	6081.073712	4418.135430

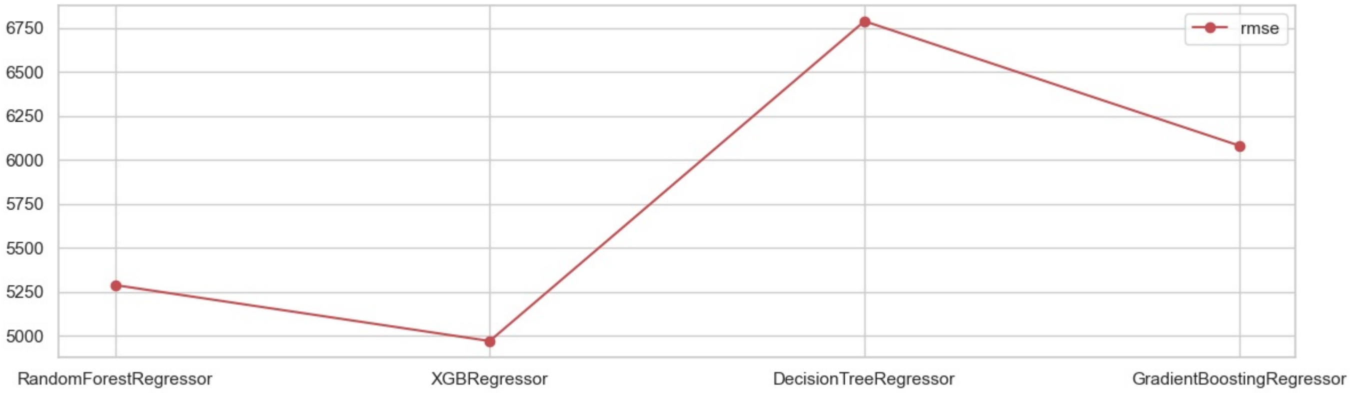
In [289…

```
fig,sx=plt.subplots(figsize=(14,4))
plt.plot(result.Algorithms,result.R2,label='R2',c='b',marker='o')
plt.legend()
plt.show()
```



In [290…

```
fig,sx=plt.subplots(figsize=(14,4))
plt.plot(result.Algorithms,result.rmse,label='rmse',c='r',marker='o')
plt.legend()
plt.show()
```



In [848…

```
# saving model
import pickle
with open('XGBRegressor_model.pkl' , 'wb') as file3 :
    pickle.dump(XGBRegressor_model, file3)
```

In [849…

```
import pickle

with open("XGBRegressor_model.pkl", "rb") as file:
   XGBRegressor_model = pickle.load(file)

print(XGBRegressor_model)  # May contain version info in metadata
```

```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, ...)
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js