HANDLING MISSING VALUES

```
#after looking at the head of the dataset we have NaN and missing
values
#To find the of missing values in each column
#if present it shows true otherwise it shows false
data.isna().any()
```

```
dateCrawled            False
name                   False
seller                 False
offerType              False
price                  False
abtest                 False
vehicleType             True
yearOfRegistration     False
gearbox                 True
powerPS                False
model                   True
kilometer              False
monthOfRegistration    False
fuelType                True
brand                  False
notRepairedDamage       True
dateCreated            False
nrOfPictures           False
postalCode             False
lastSeen               False
dtype: bool
```

```
#To find the count of missing values each column using sum function
data.isnull().sum()
```

```
dateCrawled                0
name                       0
seller                     0
offerType                  0
price                      0
abtest                     0
vehicleType            37869
yearOfRegistration         0
gearbox                20209
powerPS                    0
model                  20484
kilometer                  0
monthOfRegistration        0
fuelType               33386
brand                      0
notRepairedDamage      72060
dateCreated                0
nrOfPictures               0
```

```
postalCode                0
lastSeen                  0
dtype: int64
```

```
#Finding the description of the dataset using describe function like
mean,median etc.,
data.describe()
```

```
              price  yearOfRegistration        powerPS      kilometer
\
count  3.715280e+05       371528.000000  371528.000000  371528.000000

mean   1.729514e+04         2004.577997     115.549477  125618.688228

std    3.587954e+06           92.866598     192.139578   40112.337051

min    0.000000e+00         1000.000000       0.000000    5000.000000

25%    1.150000e+03         1999.000000      70.000000  125000.000000

50%    2.950000e+03         2003.000000     105.000000  150000.000000

75%    7.200000e+03         2008.000000     150.000000  150000.000000

max    2.147484e+09         9999.000000   20000.000000  150000.000000


       monthOfRegistration  nrOfPictures    postalCode
count        371528.000000      371528.0  371528.00000
mean              5.734445           0.0   50820.66764
std               3.712412           0.0   25799.08247
min               0.000000           0.0    1067.00000
25%               3.000000           0.0   30459.00000
50%               6.000000           0.0   49610.00000
75%               9.000000           0.0   71546.00000
max              12.000000           0.0   99998.00000
```

```
#Finding the mode of vehicleType column using mode function
data['vehicleType'].mode()
```

```
0    limousine
dtype: object
```

```
#total value_counts in vehicleType column
data['vehicleType'].value_counts()
```

```
limousine     95894
kleinwagen    80023
kombi         67564
bus           30201
cabrio        22898
```

```
coupe         19015
suv           14707
andere         3357
Name: vehicleType, dtype: int64
```

```
#Replacing all NaN values in vehicleType column using mode
data['vehicleType'].fillna("limousine",inplace=True)
```

```
#Finding the mode of vehicleType column using mode function
data['gearbox'].mode()
```

```
0    manuell
dtype: object
```

```
#Replacing all NaN values in gearbox column using mode
data['gearbox'].fillna("manuell",inplace=True)


#Finding the mode of model column using mode function
data['model'].mode()
```

```
0    golf
dtype: object
```

```
#Replacing all NaN values in model column using mode
data['model'].fillna("golf",inplace=True)

#Finding the mode of fueltype column using mode function
data['fuelType'].mode()
```

```
0    benzin
dtype: object
```

```
#Replacing all NaN values in model column using mode
data['fuelType'].fillna("benzin",inplace=True)

#Finding the mode of notRepairedDamage column using mode function
data['notRepairedDamage'].mode()
```

```
0    nein
dtype: object
```

```
#Replacing all NaN values in notRepairedDamage column using mode
data['notRepairedDamage'].fillna("nein",inplace=True)

data.head()
```

```
           dateCrawled                              name  seller
offerType  \
0  2016-03-24 11:52:17                       Golf_3_1.6  privat
Angebot
1  2016-03-24 10:58:45             A5_Sportback_2.7_Tdi  privat
Angebot
```

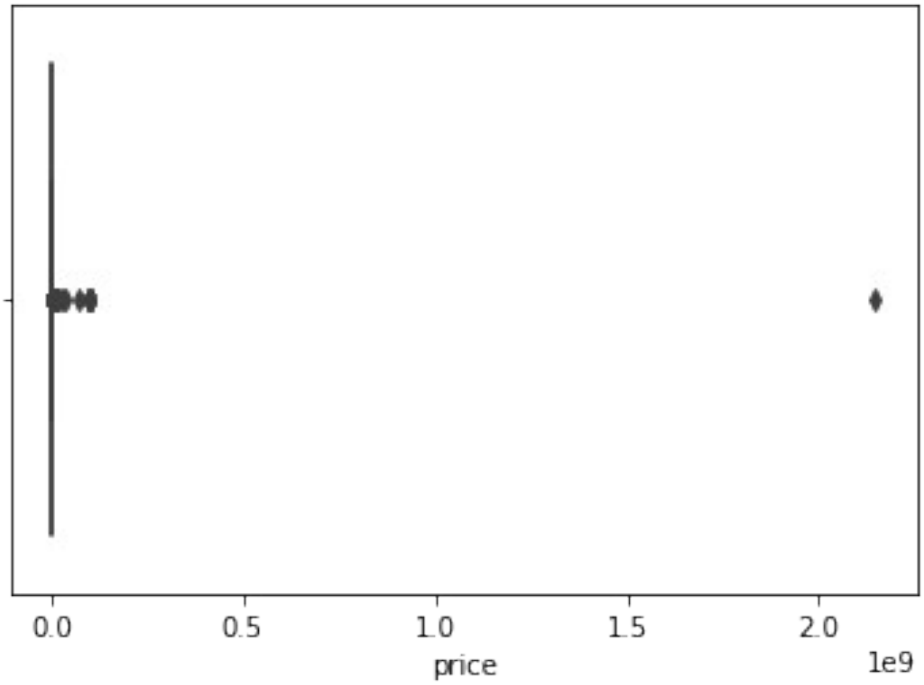
```
2  2016-03-14 12:52:21  Jeep_Grand_Cherokee_"Overland"  privat
Angebot
3  2016-03-17 16:54:04              GOLF_4_1_4__3TÜRER  privat
Angebot
4  2016-03-31 17:25:20  Skoda_Fabia_1.4_TDI_PD_Classic  privat
Angebot

   price abtest vehicleType  yearOfRegistration    gearbox  powerPS
model  \
0    480   test   limousine                1993    manuell        0
golf
1  18300   test       coupe                2011    manuell      190
golf
2   9800   test         suv                2004  automatik      163
grand
3   1500   test  kleinwagen                2001    manuell       75
golf
4   3600   test  kleinwagen                2008    manuell       69
fabia

   kilometer  monthOfRegistration fuelType       brand
notRepairedDamage  \
0     150000                    0   benzin  volkswagen
nein
1     125000                    5   diesel        audi
ja
2     125000                    8   diesel        jeep
nein
3     150000                    6   benzin  volkswagen
nein
4      90000                    7   diesel       skoda
nein

           dateCreated  nrOfPictures  postalCode             lastSeen

0  2016-03-24 00:00:00             0       70435  2016-04-07 03:16:57

1  2016-03-24 00:00:00             0       66954  2016-04-07 01:46:50

2  2016-03-14 00:00:00             0       90480  2016-04-05 12:47:46

3  2016-03-17 00:00:00             0       91074  2016-03-17 17:40:17

4  2016-03-31 00:00:00             0       60437  2016-04-06 10:17:21
```

****OUTLIERS DETECTION AND REPLACING OUTLIERS****

```
sns.boxplot(data['price'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

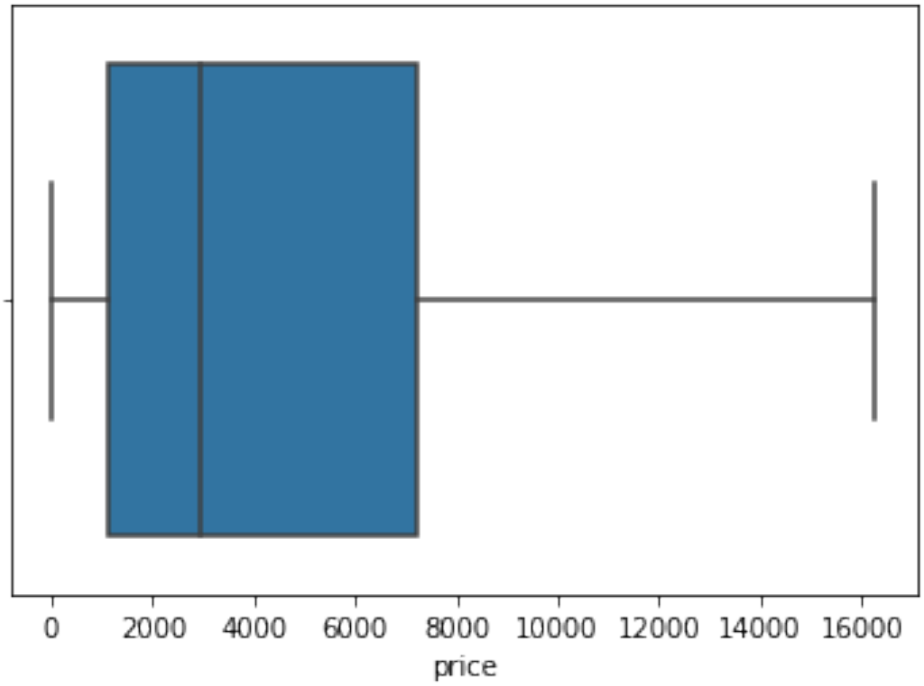
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7c68751490>
```




```
#finding the interquartilerange of price column
q1=data['price'].quantile(0.25)
q3=data['price'].quantile(0.75)
iqr=q3-q1
lower_bound=q1-1.5*iqr
upper_bound=q3+1.5*iqr

#replacing the outliers of price column with mean
data['price']=np.where(data['price']>upper_bound,upper_bound,np.where(
data['price']<lower_bound,upper_bound,data['price']))

#boxplot for price column
sns.boxplot(data['price'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

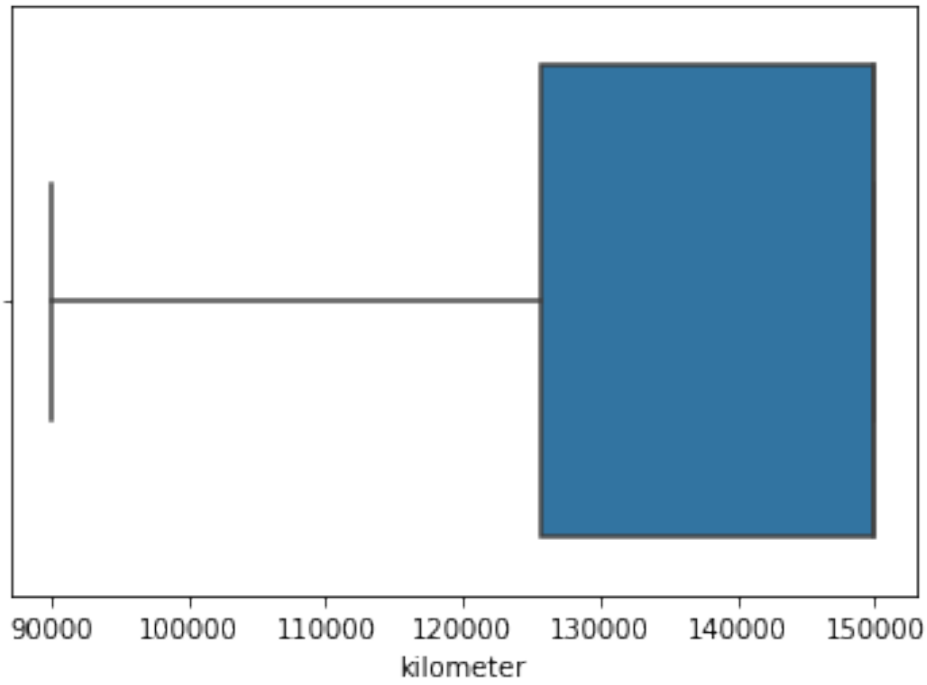
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7c68476750>
```




```
#finding the interquartilerange of kilometer column and replacing the
outliers with mean
q1=data['kilometer'].quantile(0.25)
q3=data['kilometer'].quantile(0.75)
iqr=q3-q1
lower_bound=q1-1.5*iqr
upper_bound=q3+1.5*iqr
data['kilometer']=np.where(data['kilometer']>upper_bound,data['kilomet
er'].mean(),np.where(data['kilometer']<lower_bound,data['kilometer'].m
ean(),data['kilometer']))

#boxplot for kilometer column
sns.boxplot(data['kilometer'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

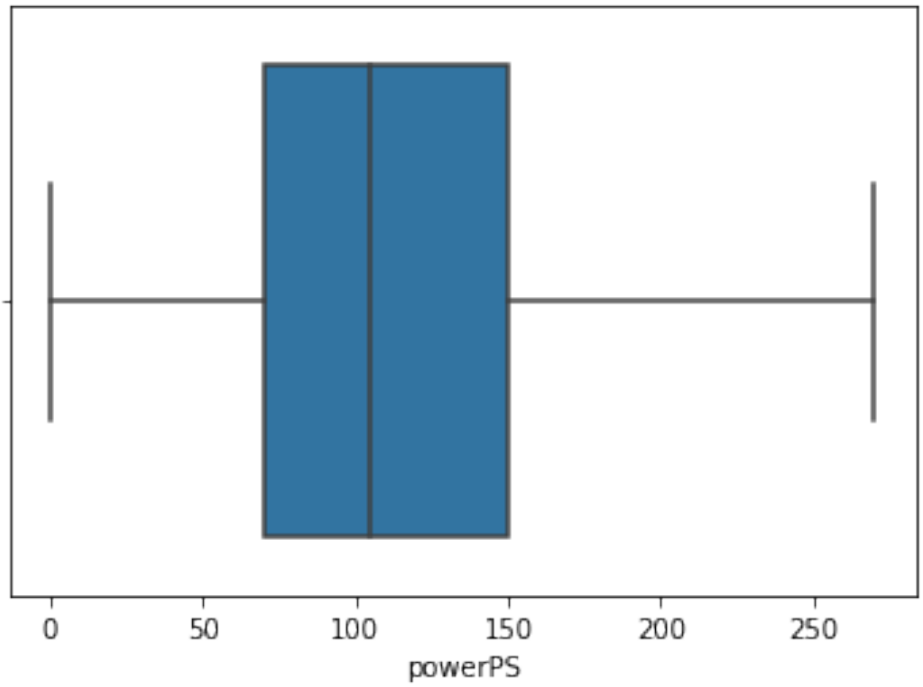
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7c686e26d0>
```




```
#finding the interquartilerange of powerPS column and replacing the
outliers with lower_bound,upper_bound
q1=data['powerPS'].quantile(0.25)
q3=data['powerPS'].quantile(0.75)
iqr=q3-q1
lower_bound=q1-1.5*iqr
upper_bound=q3+1.5*iqr
data['powerPS']=np.where(data['powerPS']>upper_bound,upper_bound,np.wh
ere(data['powerPS']<lower_bound,lower_bound,data['powerPS']))

#boxplot for powerPS column
sns.boxplot(data['powerPS'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

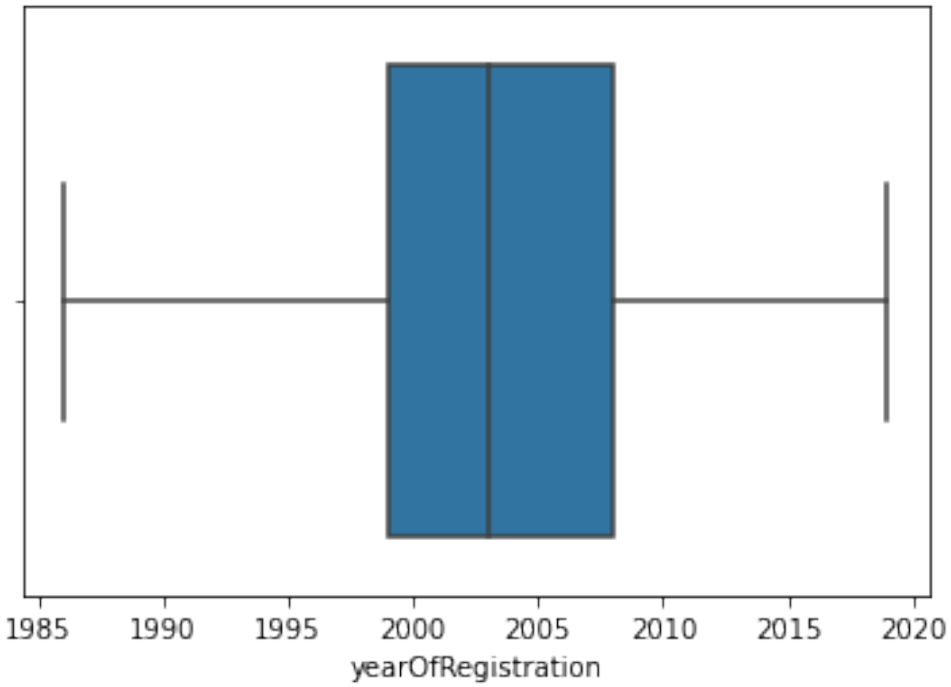
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7c67f31710>
```




```
#finding the interquartilerange of yearOfRegistration column and
replacing the outliers with mean
q1=data['yearOfRegistration'].quantile(0.25)
q3=data['yearOfRegistration'].quantile(0.75)
iqr=q3-q1
lower_bound=q1-1.5*iqr
upper_bound=q3+1.5*iqr
data['yearOfRegistration']=np.where(data['yearOfRegistration']>upper_b
ound,data['yearOfRegistration'].mode(),np.where(data['yearOfRegistrati
on']<lower_bound,data['yearOfRegistration'].mode(),data['yearOfRegistr
ation']))

#boxplot for yearOfRegistration column
sns.boxplot(data['yearOfRegistration'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

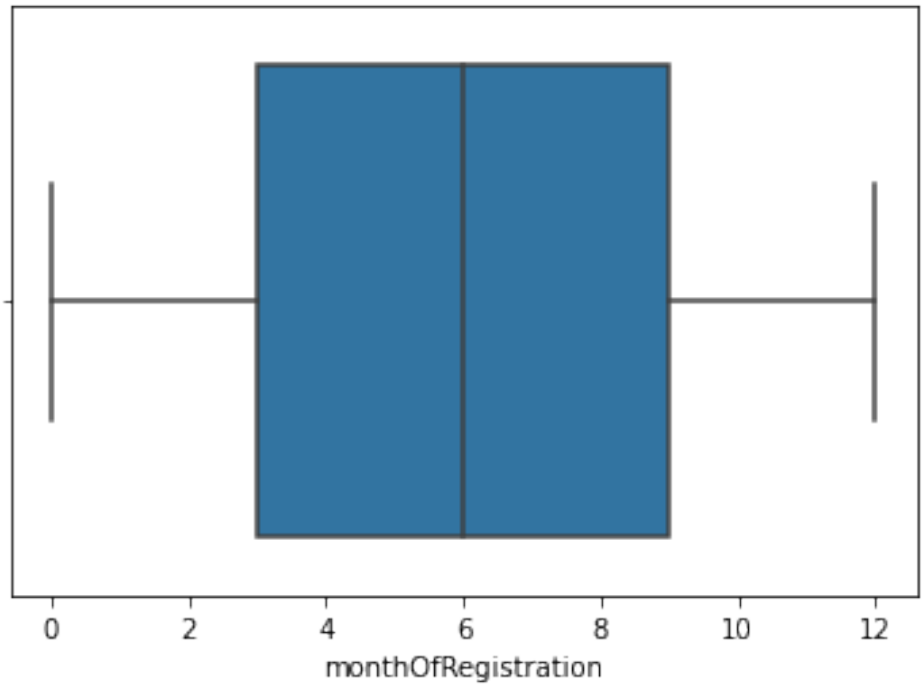
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7c68495850>
```




```
#boxplot for monthOfRegistation column
sns.boxplot(data['monthOfRegistration'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7c67e0dfd0>
```




```
#Reading the first five rows of cleaned dataset using head function
data.head()
```

```
           dateCrawled                              name  seller
offerType  \
0  2016-03-24 11:52:17                      Golf_3_1.6  privat
Angebot
1  2016-03-24 10:58:45            A5_Sportback_2.7_Tdi  privat
Angebot
2  2016-03-14 12:52:21  Jeep_Grand_Cherokee_"Overland"  privat
Angebot
3  2016-03-17 16:54:04              GOLF_4_1_4__3TÜRER  privat
Angebot
4  2016-03-31 17:25:20  Skoda_Fabia_1.4_TDI_PD_Classic  privat
Angebot

     price abtest vehicleType  yearOfRegistration    gearbox  powerPS
model  \
0    480.0   test   limousine                1993    manuell       0.0
golf
1  16275.0   test       coupe                2011    manuell     190.0
golf
2   9800.0   test         suv                2004  automatik     163.0
grand
3   1500.0   test  kleinwagen                2001    manuell      75.0
golf
4   3600.0   test  kleinwagen                2008    manuell      69.0
fabia
```

```
   kilometer  monthOfRegistration fuelType       brand
notRepairedDamage  \
0   150000.0                   0   benzin  volkswagen
nein
1   125000.0                   5   diesel        audi
ja
2   125000.0                   8   diesel        jeep
nein
3   150000.0                   6   benzin  volkswagen
nein
4    90000.0                   7   diesel       skoda
nein

           dateCreated  nrOfPictures  postalCode             lastSeen

0  2016-03-24 00:00:00             0       70435  2016-04-07 03:16:57

1  2016-03-24 00:00:00             0       66954  2016-04-07 01:46:50

2  2016-03-14 00:00:00             0       90480  2016-04-05 12:47:46

3  2016-03-17 00:00:00             0       91074  2016-03-17 17:40:17

4  2016-03-31 00:00:00             0       60437  2016-04-06 10:17:21
```

****EXPLORATORY DATA ANALYSIS****

Exploring Categorical Features

```
#list of all categorical columns
list(data.select_dtypes('object'))
```

```
['dateCrawled',
 'name',
 'seller',
 'offerType',
 'abtest',
 'vehicleType',
 'gearbox',
 'model',
 'fuelType',
 'brand',
 'notRepairedDamage',
 'dateCreated',
 'lastSeen']
```

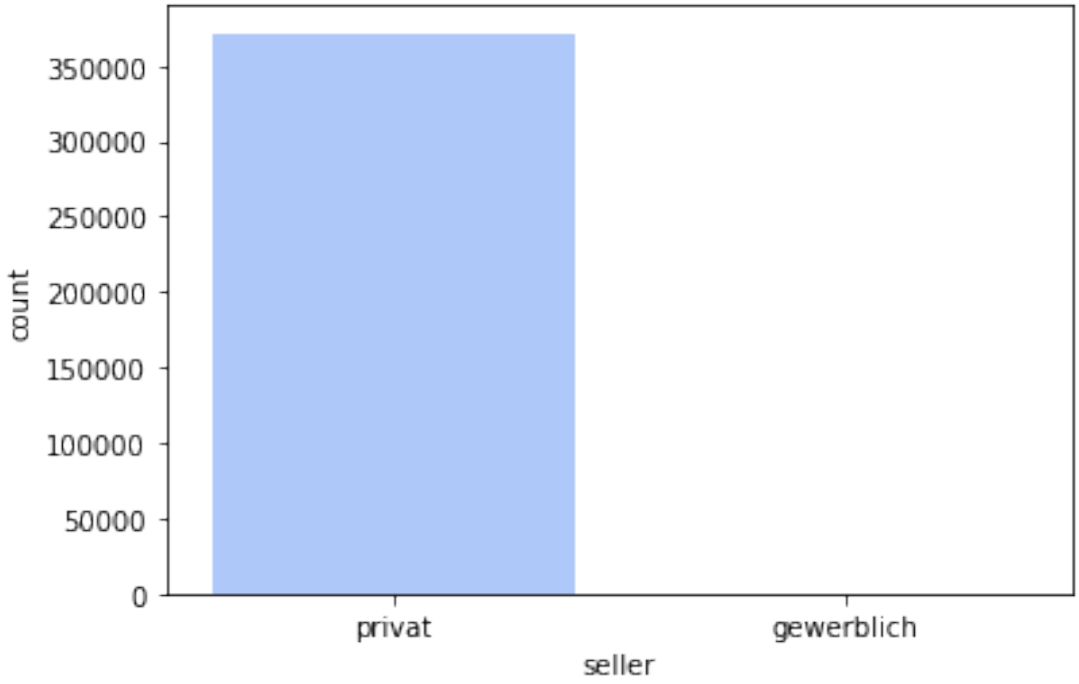
```
data['seller'].value_counts()
```

```
privat        371525
gewerblich         3
Name: seller, dtype: int64
```

```
#counting public and gewerblich types in seller column using countplot
sns.countplot(data['seller'],palette='coolwarm',saturation=0.9)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7c67e13a10>
```

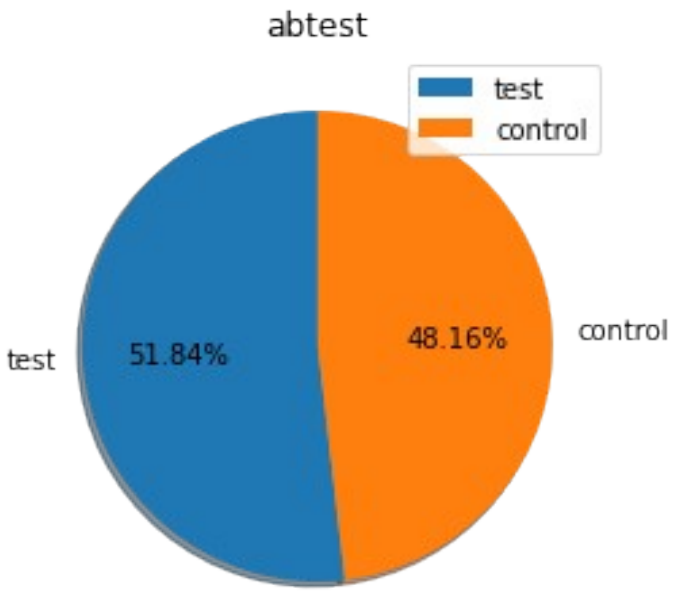



```
data['abtest'].value_counts()
```

```
test       192585
control    178943
Name: abtest, dtype: int64
```

```
#counting the percentage of different types in abtest column using pie
chart
plt.pie(data['abtest'].value_counts(),startangle=90,labels=['test','co
ntrol'],shadow=True,autopct='%1.2f%%')
plt.legend()
plt.title("abtest")
```

```
Text(0.5, 1.0, 'abtest')
```



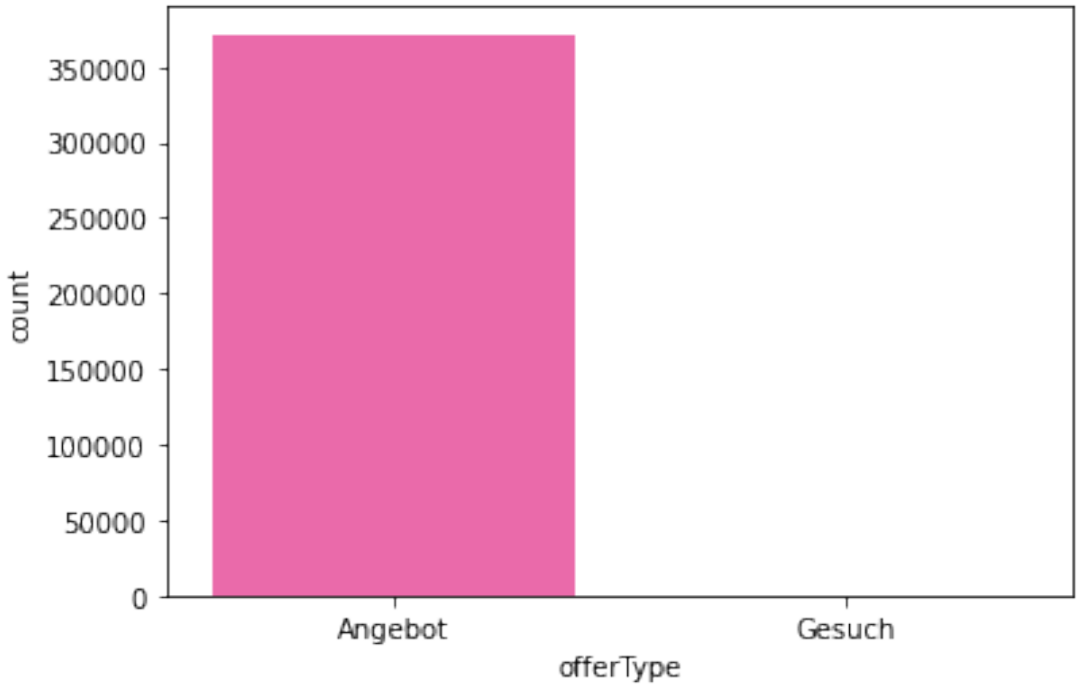

```
data['offerType'].value_counts()
```

```
Angebot    371516
Gesuch         12
Name: offerType, dtype: int64
```

```
#counting angebot and gesuch types in offerType column using countplot
sns.countplot(data['offerType'],palette='spring')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7c67cd3dd0>
```



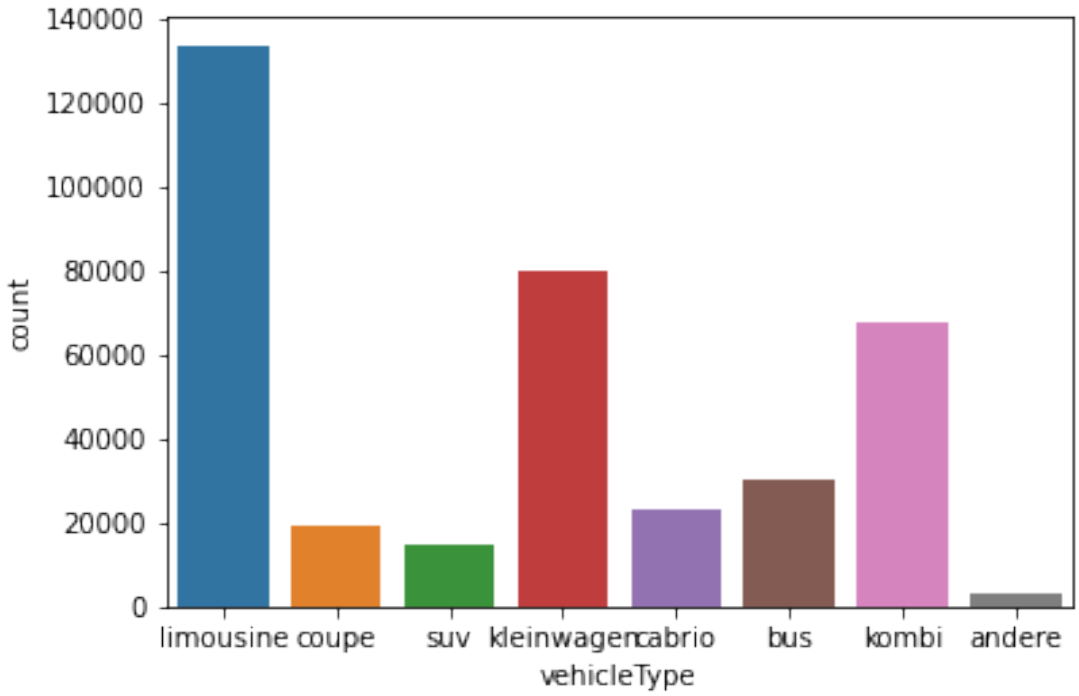

```
data['vehicleType'].value_counts()
```

```
limousine     133763
kleinwagen     80023
kombi          67564
bus            30201
cabrio         22898
coupe          19015
suv            14707
andere          3357
Name: vehicleType, dtype: int64
```

```
#count of each type in vehicleType column
sns.countplot(data['vehicleType'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

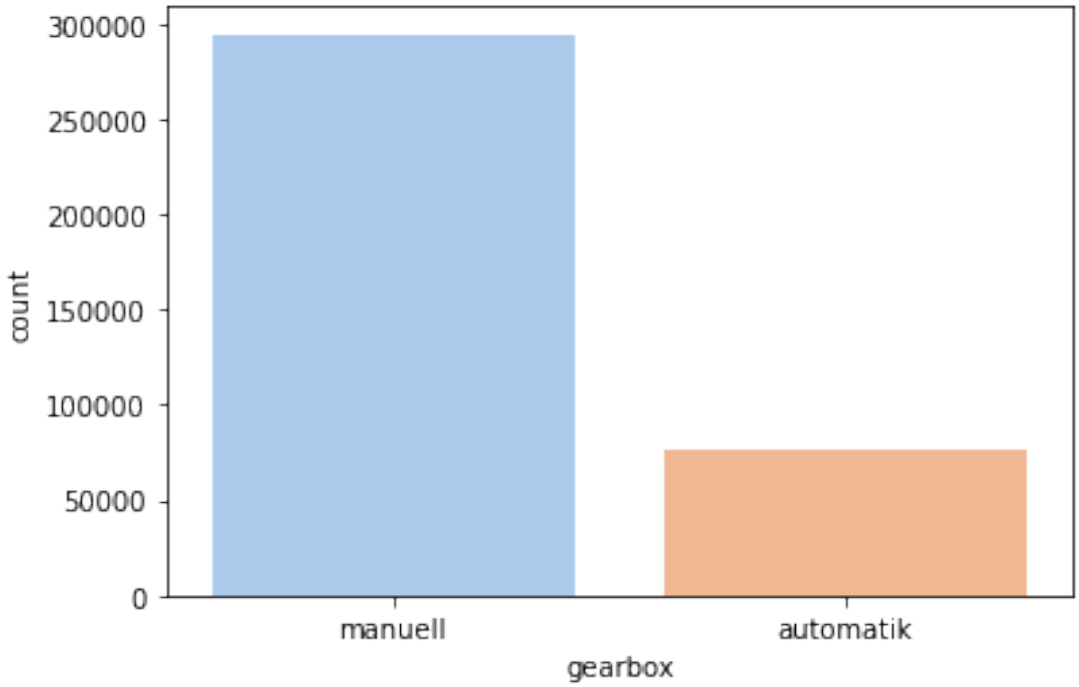
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7c67cbdc10>
```




```
#count of each type in gearbox column
sns.countplot(data['gearbox'],palette='pastel')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7c67c39ed0>
```



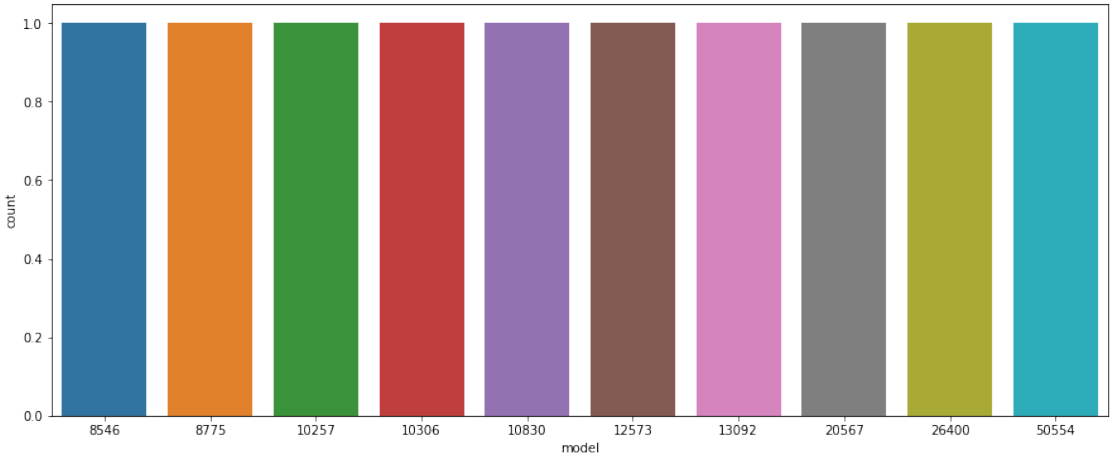

```
data['model'].value_counts()
```

```
golf                50554
andere              26400
3er                 20567
polo                13092
corsa               12573
                    ...
serie_2                 8
rangerover              6
serie_3                 4
serie_1                 2
discovery_sport         1
Name: model, Length: 251, dtype: int64
```

```
#top 10 models in model column
plt.figure(figsize =(15,6))
sns.countplot(data['model'].value_counts().head(10))
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7c67e0d450>
```

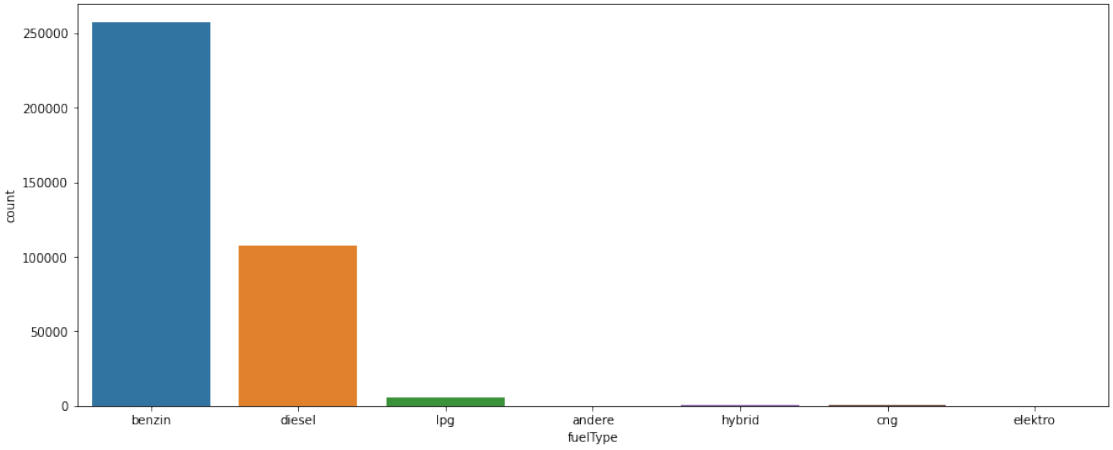



```
data['fuelType'].value_counts()
```

```
benzin     257243
diesel     107746
lpg          5378
cng           571
hybrid        278
andere        208
elektro       104
Name: fuelType, dtype: int64
```

```
plt.figure(figsize =(15,6))
sns.countplot(data['fuelType'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning

<matplotlib.axes._subplots.AxesSubplot at 0x7f7c67cb2e90>
```



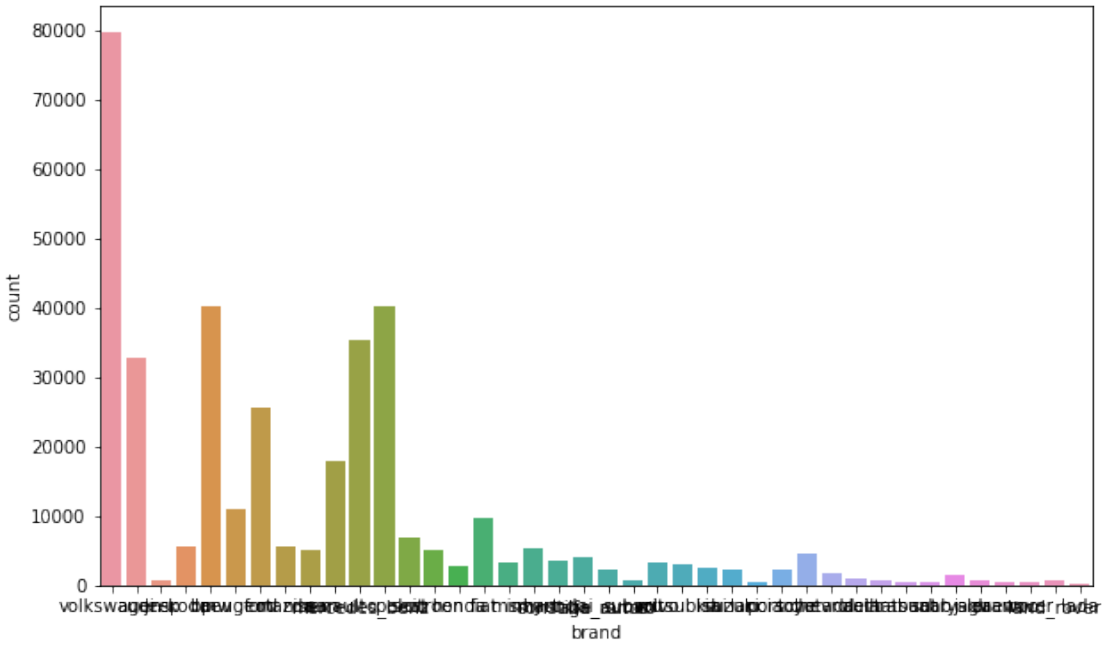
```
data['brand'].value_counts().head()
```

```
volkswagen       79640
bmw              40274
opel             40136
mercedes_benz    35309
audi             32873
Name: brand, dtype: int64
```

```
#count of eaach brand in brand column
plt.figure(figsize =(10,6))
sns.countplot(data['brand'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7c67af4fd0>
```




```
data['notRepairedDamage'].value_counts()
```

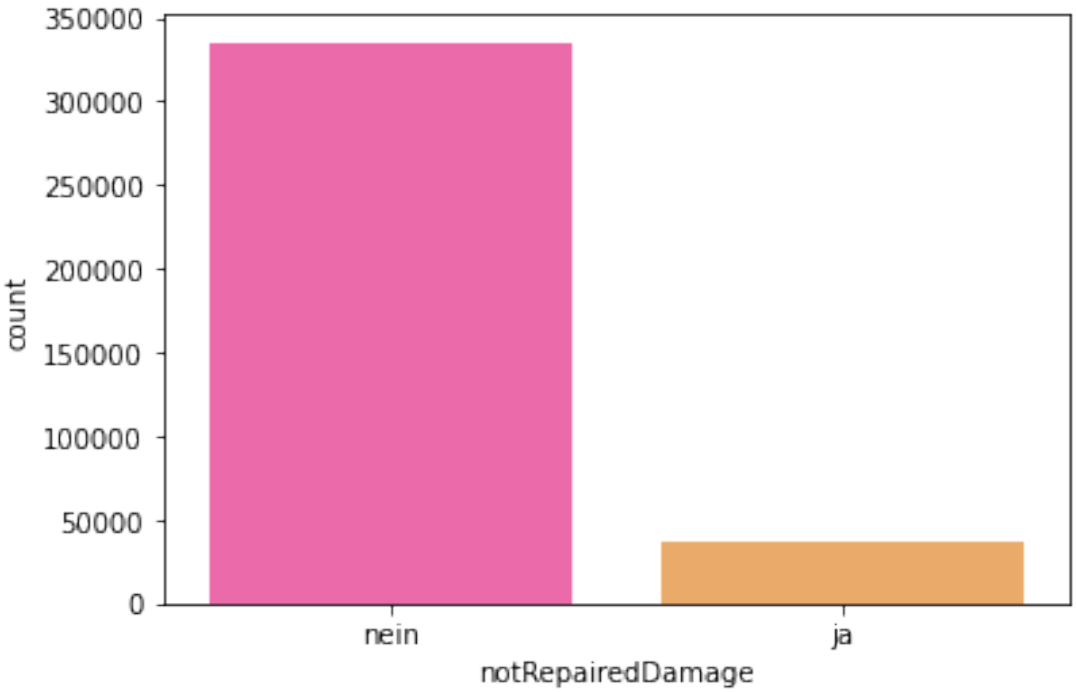
```
nein    335242
ja       36286
Name: notRepairedDamage, dtype: int64
```

```
sns.countplot(data['notRepairedDamage'],palette='spring')
```

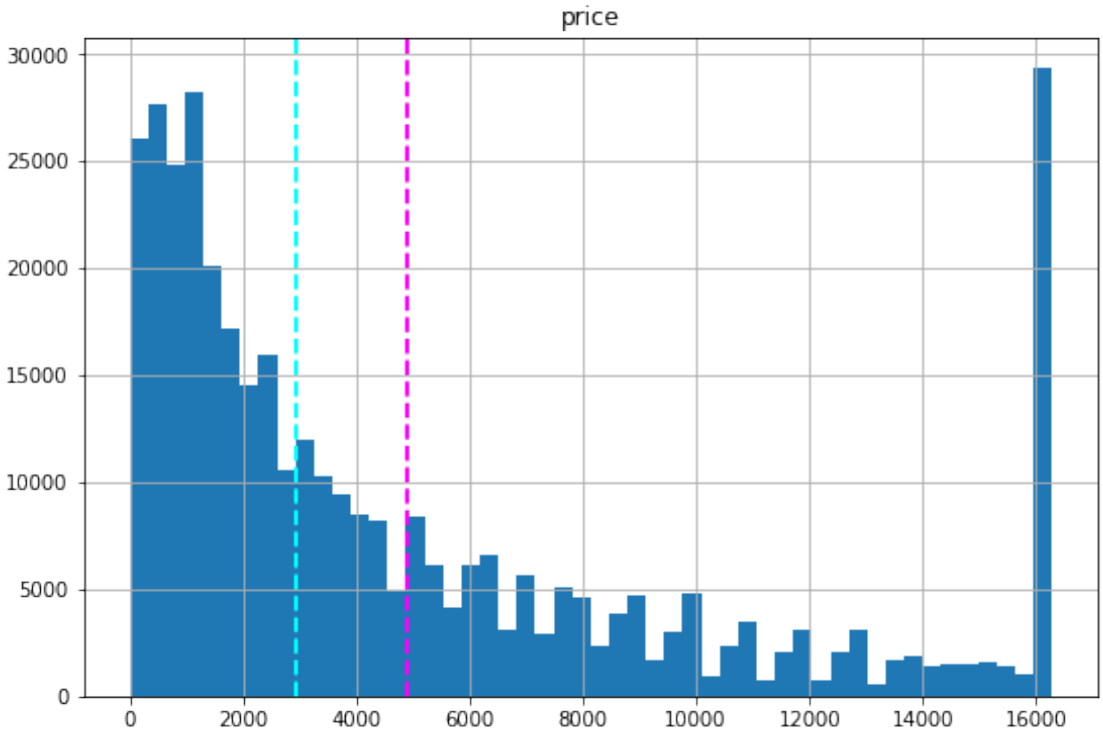
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variable as a keyword arg: x. From
```

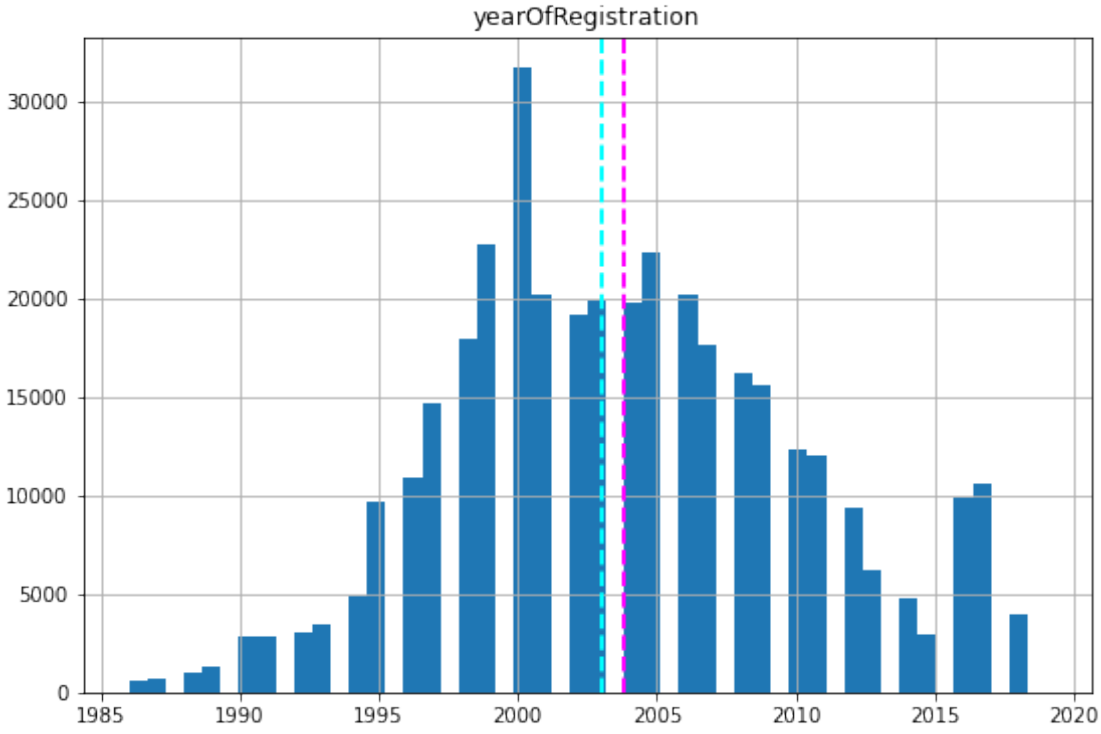
```
version 0.12, the only valid positional argument will be `data`, and
passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  FutureWarning
```

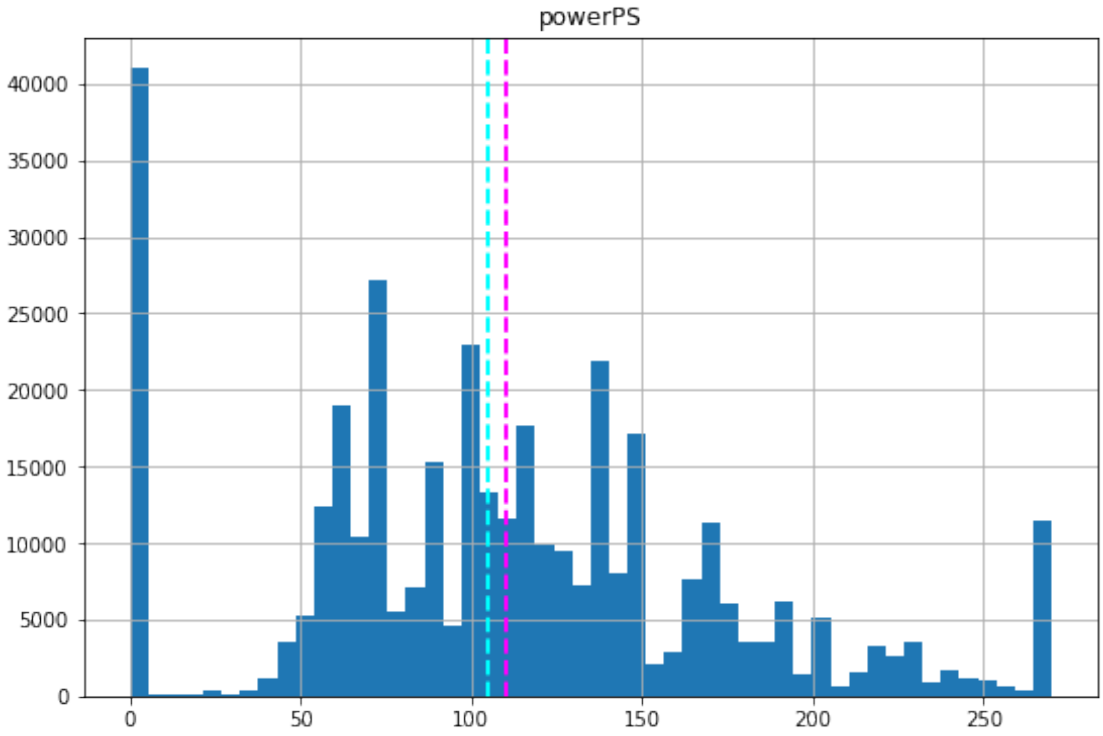
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7c67982f50>
```

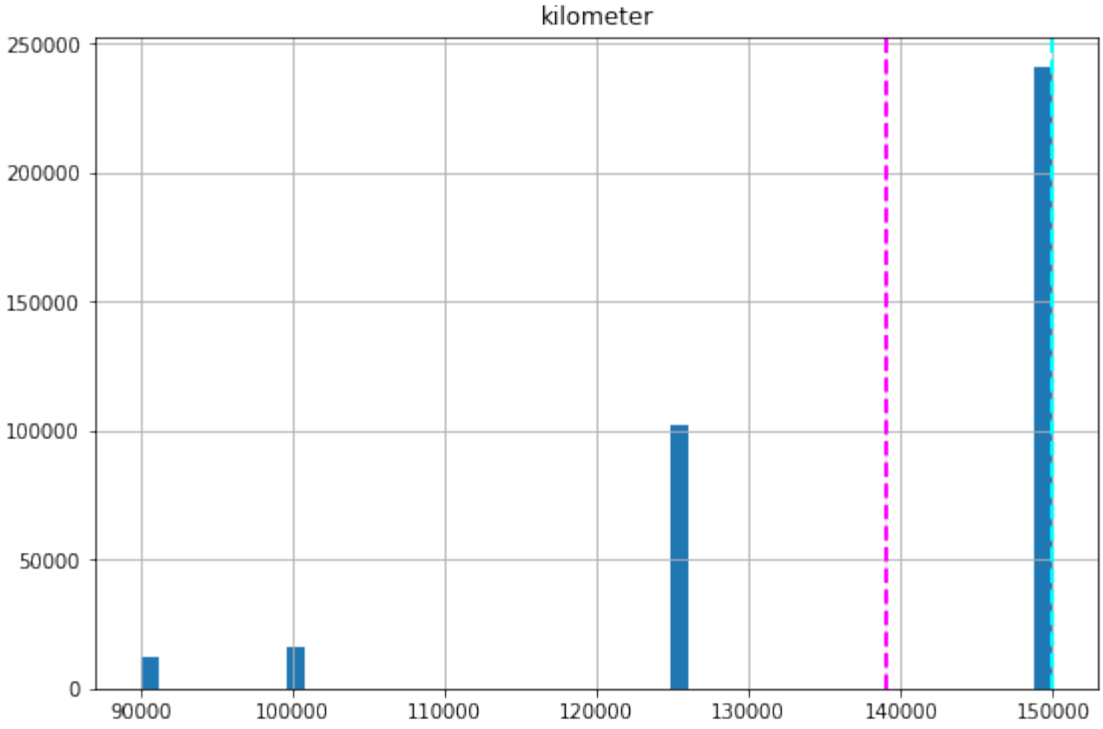


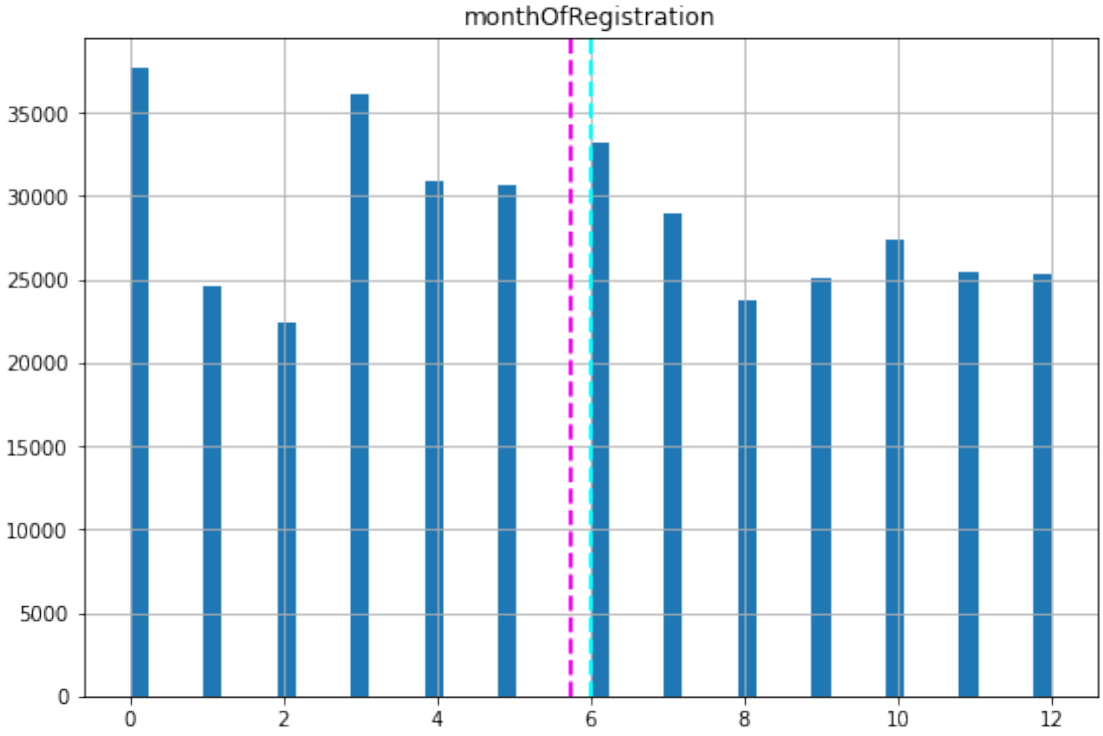

```
a=list(data.select_dtypes('number'))
for i in a:
    fig = plt.figure(figsize=(9, 6))
    ax = fig.gca()
    feature = data[i]
    feature.hist(bins=50, ax = ax)
    ax.axvline(feature.mean(), color='magenta', linestyle='dashed',
linewidth=2)
    ax.axvline(feature.median(), color='cyan', linestyle='dashed',
linewidth=2)
    ax.set_title(i)
plt.show()
```

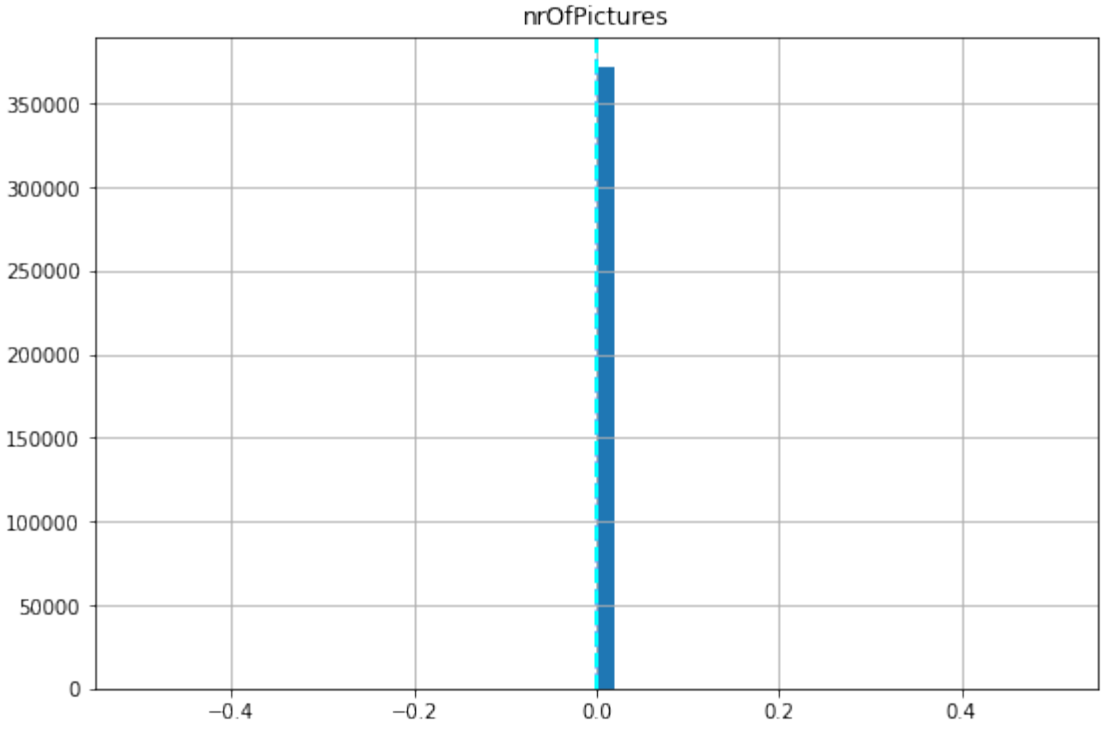

price
30000
25000
20000
15000
10000
5000
0
0
2000
4000
6000
8000
10000
12000
14000
16000

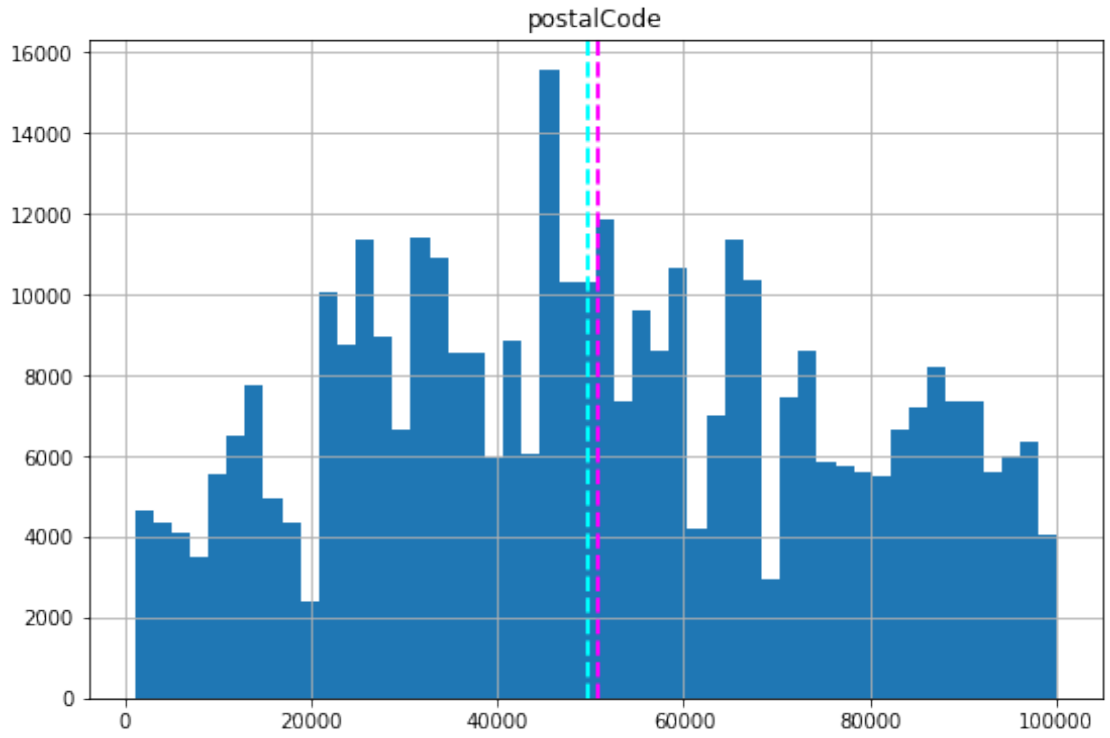

yearOfRegistration
30000
25000
20000
15000
10000
5000
0
1985
1990
1995
2000
2005
2010
2015
2020


powerPS
40000
35000
30000
25000
20000
15000
10000
5000
0
0
50
100
150
200
250


kilometer
250000
200000
150000
100000
50000
0
90000
100000
110000
120000
130000
140000
150000


monthOfRegistration
35000
30000
25000
20000
15000
10000
5000
0
0
2
4
6
8
10
12


nrOfPictures
350000
300000
250000
200000
150000
100000
50000
0
−0.4
−0.2
0.0
0.2
0.4



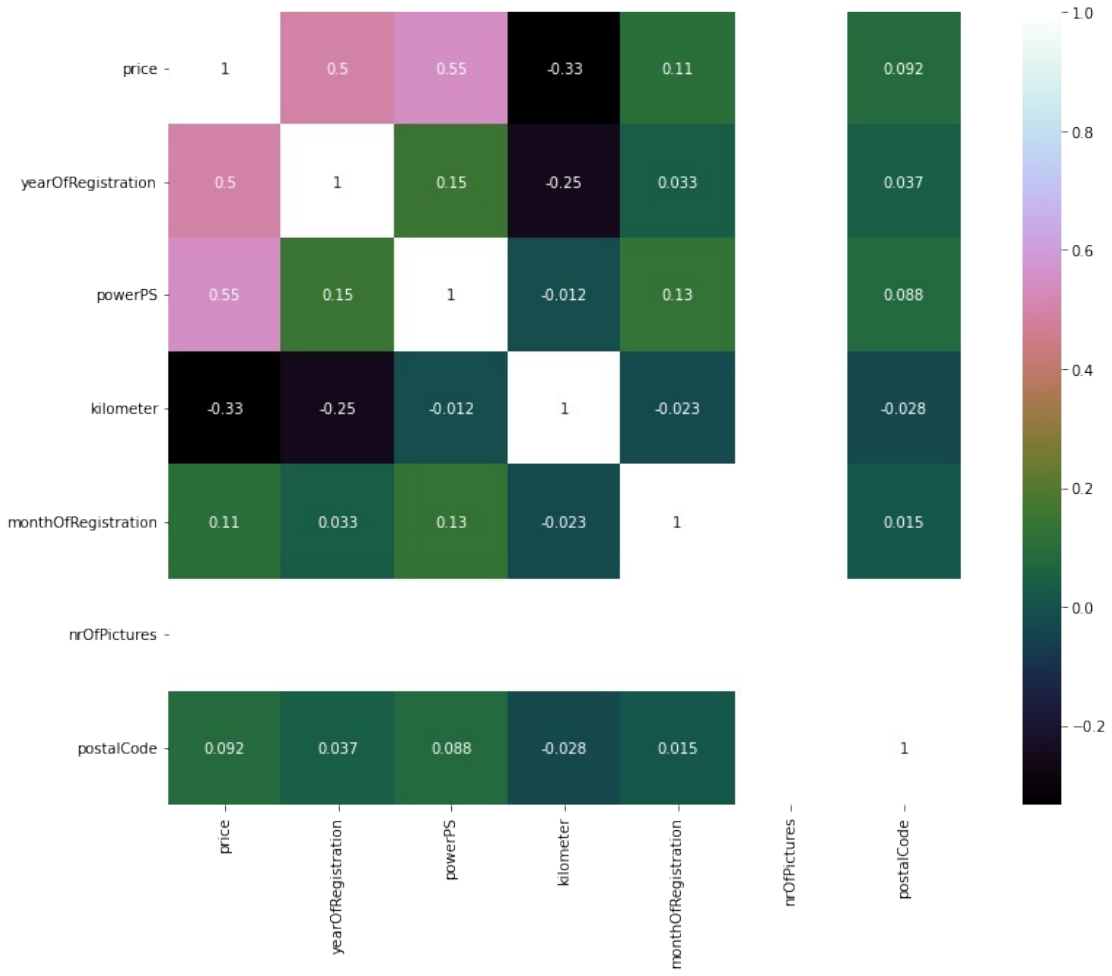

```
#correlation of  dataset using correaltion function
correlation=data.corr()
correlation
```

	price	yearOfRegistration	powerPS	kilometer	monthOfRegistration	nrOfPictures	postalCode
price	1.000000	0.498059	0.547702	-0.333261	0.107701	NaN	0.092355
yearOfRegistration	0.498059	1.000000	0.148963	-0.250738	0.032619	NaN	0.036769
powerPS	0.547702	0.148963	1.000000	-0.012199	0.133211	NaN	0.087730
kilometer	-0.333261	-0.250738	-0.012199	1.000000	-0.022828	NaN	-0.028500
monthOfRegistration	0.107701	0.032619	0.133211	-0.022828	1.000000	NaN	0.014963
nrOfPictures	NaN	NaN	NaN	NaN	NaN	NaN	NaN
postalCode	0.092355	0.036769	0.087730	-0.028500	0.014963	NaN	1.000000

```
nrOfPictures                      NaN          NaN          NaN
postalCode                   0.014963          NaN     1.000000
```

```
#exploring the correlation using heatmap
plt.figure(figsize=(15,10))
sns.heatmap(correlation, vmax=1,
square=True,annot=True,cmap='cubehelix')
```

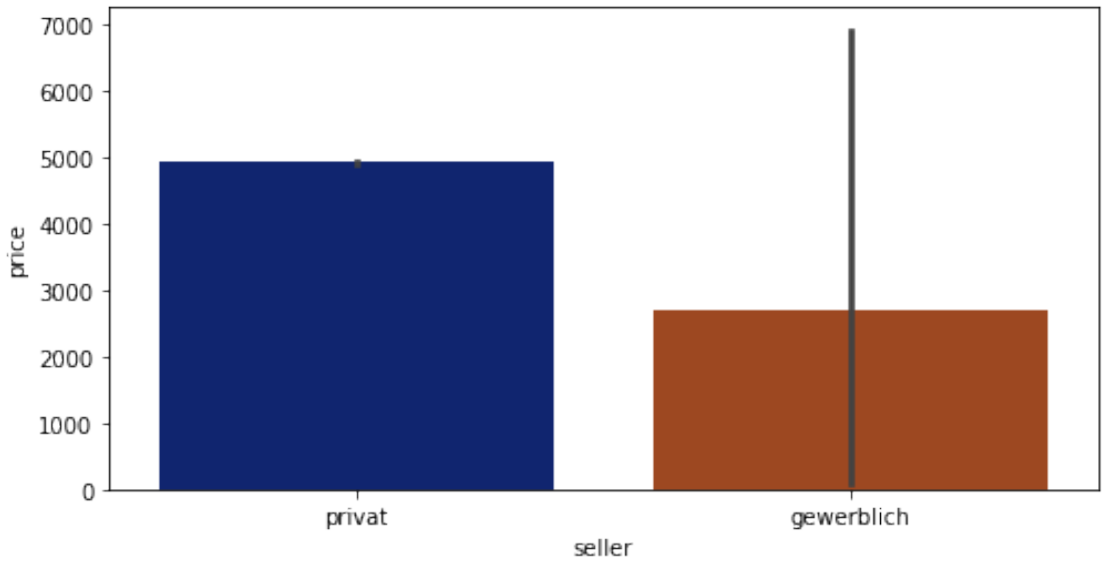
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7c649205d0>
```



1.SELLER VS PRICE

```
plt.figure(figsize=(8,4))
sns.barplot(x='seller',y='price',data=data,palette='dark')
```

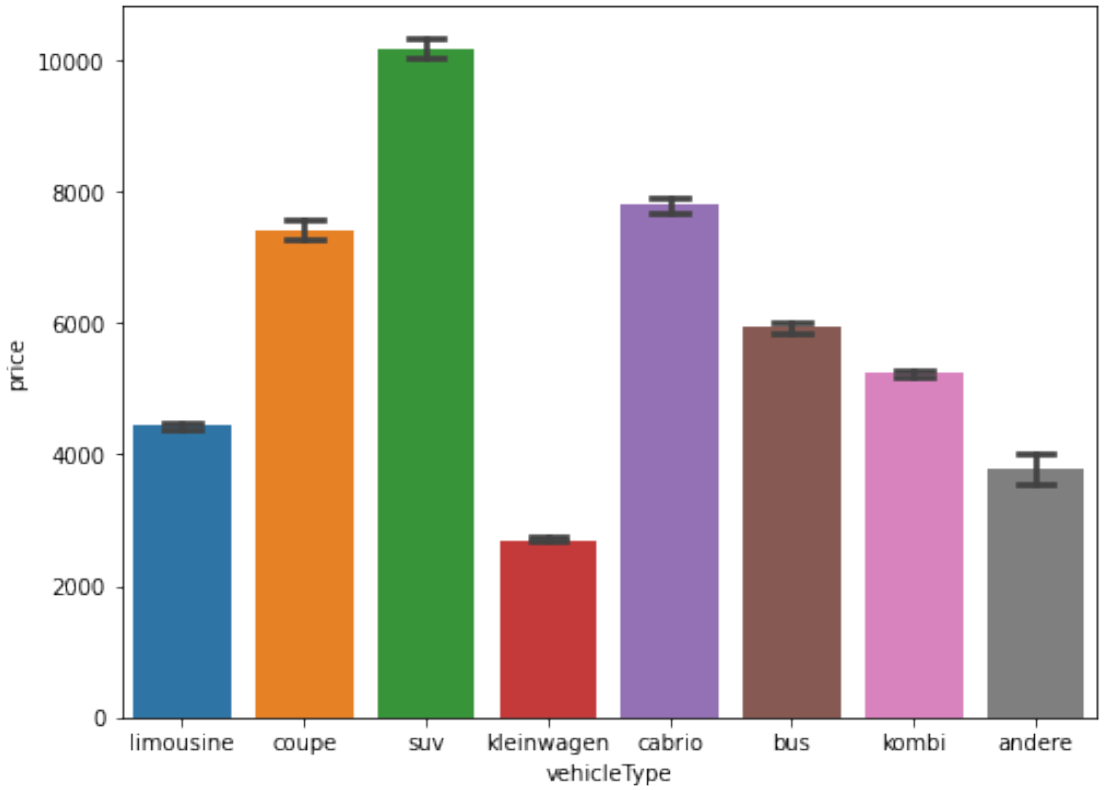
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7c678d8750>
```



2.VEHICLETYPE VS PRICE

```
plt.figure(figsize=(8,6))
sns.barplot(x='vehicleType',y='price',data=data,ci=100,capsize=0.3,sat
uration=0.8)
```

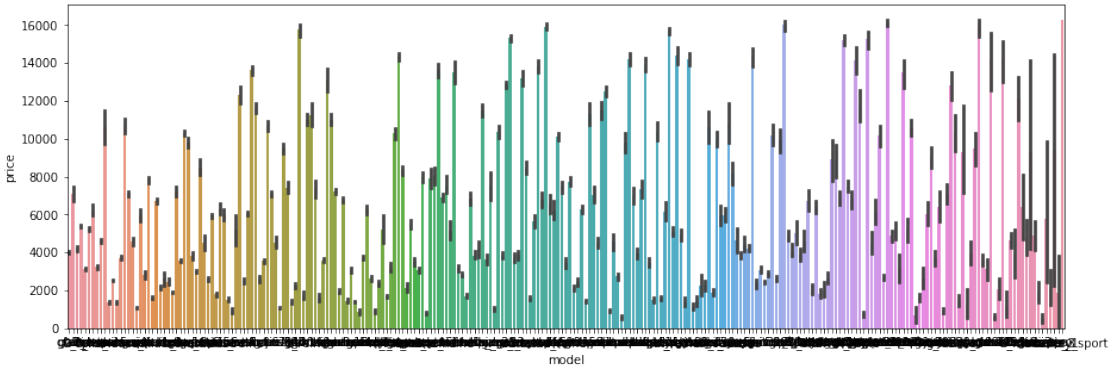
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7c67e5d4d0>
```



3.MODEL VS PRICE

```
plt.figure(figsize=(15,5))
sns.barplot(x='model',y='price',data=data)
```

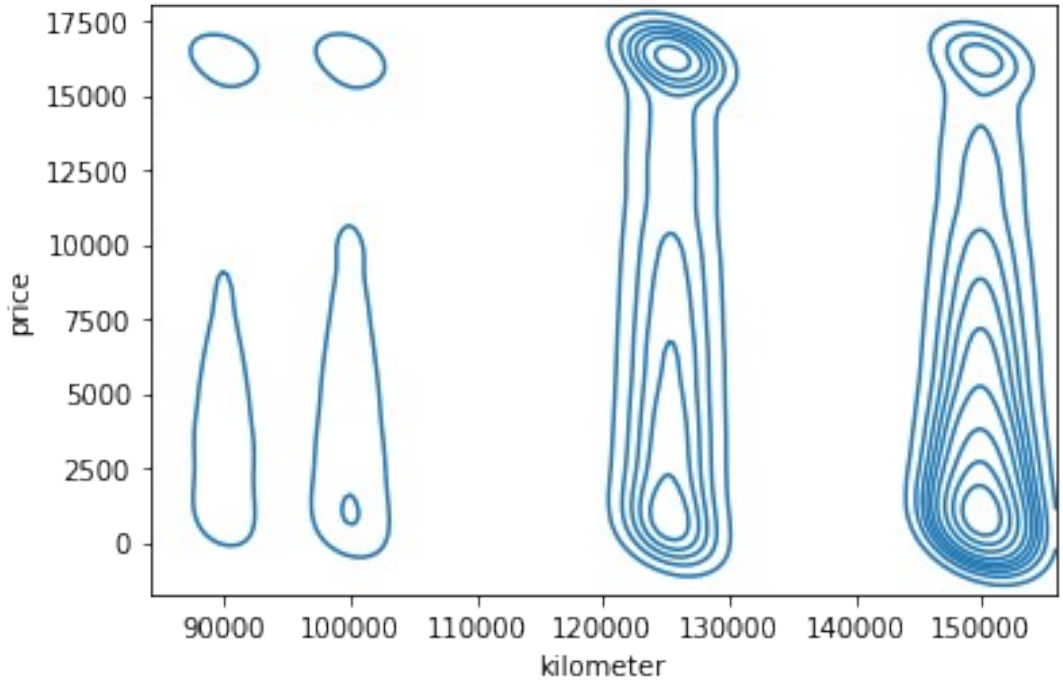
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7c67e64150>
```



4.KILOMETER VS PRICE

```
sns.kdeplot(x='kilometer',y='price',data=data,palette='husl')
```

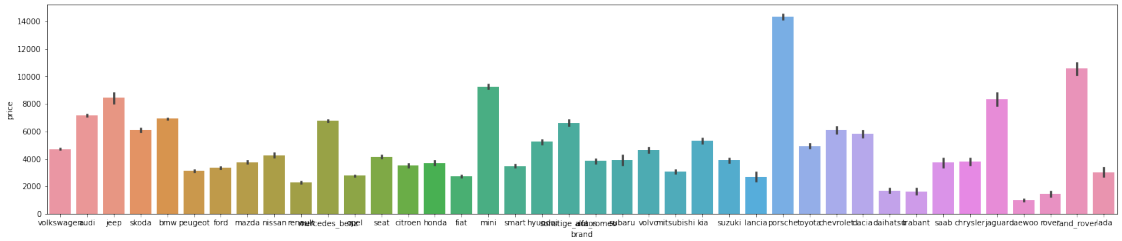
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7c6464bbd0>
```



5.BRAND VS PRICE

```
plt.figure(figsize=(25,5))
sns.barplot(x='brand',y='price',data=data)
```

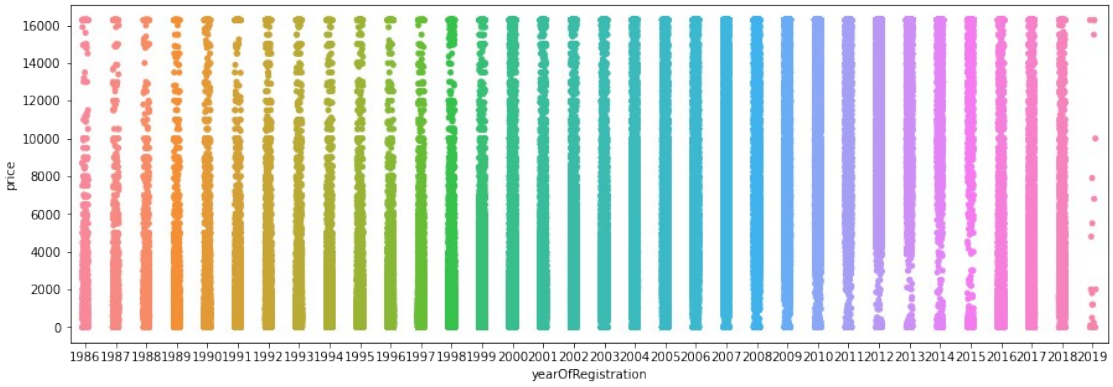
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7c64d00310>
```



6. YEAR OF REGISTRATION VS PRICE

```
plt.figure(figsize=(15,5))
sns.stripplot(x='yearOfRegistration',y='price',data=data)
```

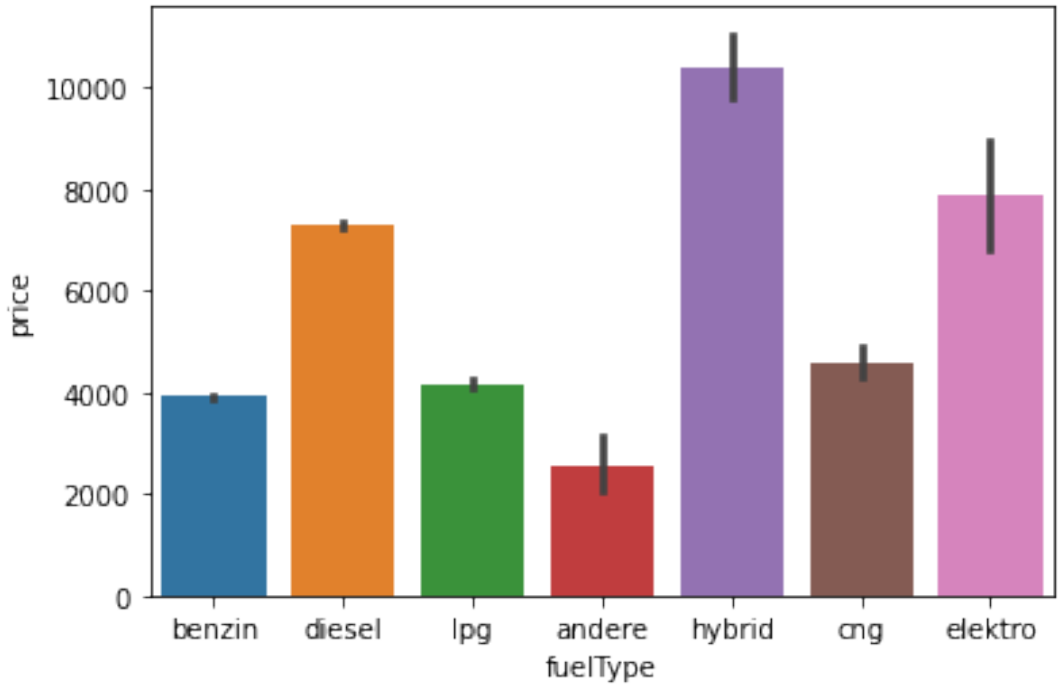
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7c6e546050>
```



7.FUEL TYPE VS PRICE

```
sns.barplot(x='fuelType',y='price',data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7c63db1250>
```



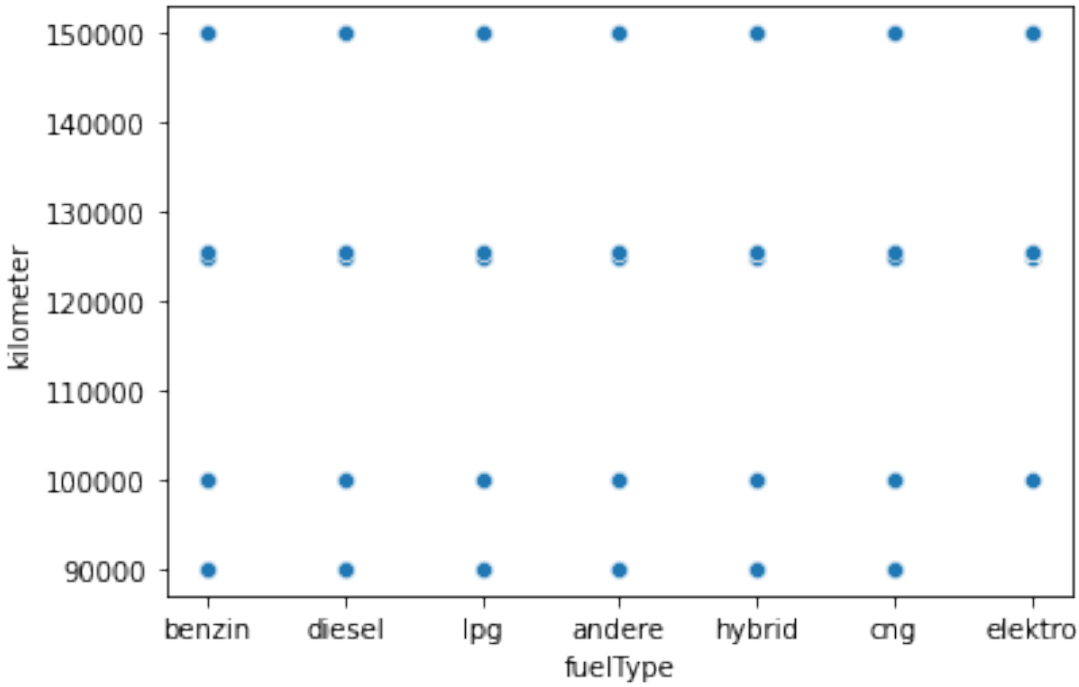
8.GEARBOX VS KILOMETER

```
sns.pointplot(x='gearbox',y='kilometer',hue='fuelType',data=data,ci=99
,saturation=0.8,capsize=0.3)
```

9.KILOMETER VS PRICE

```
sns.scatterplot(x='fuelType',y='kilometer',data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7c646a3cd0>
```

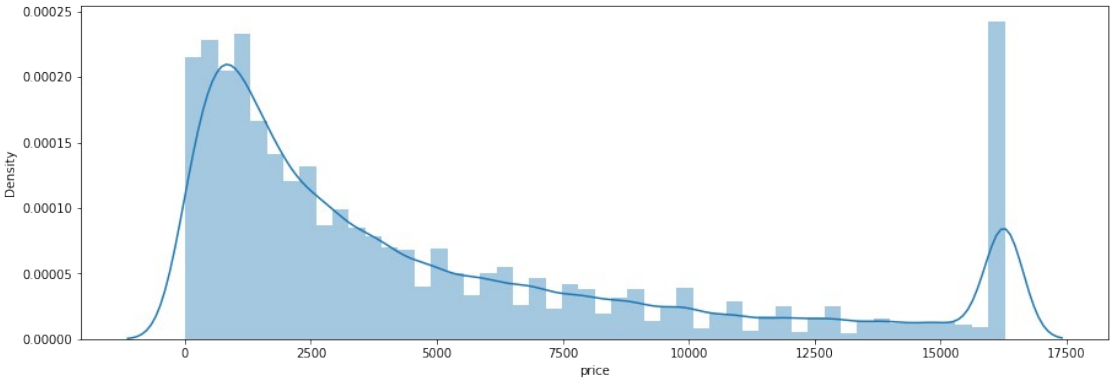


DISTRIBUTION PLOT

```
#examing the distribution of price column using distplot in seaborn library
plt.figure(figsize=(15,5))
sns.distplot(data['price'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2c0fdffb50>
```




```
parameters={'seller':{'privat':0,'gewerblich':1},
            'abtest':{'test':0,'control':1},
```

```
            'notRepairedDamage':{'nein':0,'ja':1},
            'vehicleType':
{'limousine':0,'kleinwagen':1,'kombi':2,'bus':3,'cabrio':4,'coupe':5,'
suv':6,'andere':7},
            'fuelType':
{'benzin':0,'diesel':1,'lpg':2,'cng':3 ,'hybrid':4,'andere':5,'elektro
':6}}
data_df=data.replace(parameters)
data_df.head()
```

```
           dateCrawled                              name  seller
offerType  \
0  2016-03-24 11:52:17                       Golf_3_1.6       0
Angebot
1  2016-03-24 10:58:45             A5_Sportback_2.7_Tdi       0
Angebot
2  2016-03-14 12:52:21  Jeep_Grand_Cherokee_"Overland"       0
Angebot
3  2016-03-17 16:54:04               GOLF_4_1_4__3TÜRER       0
Angebot
4  2016-03-31 17:25:20  Skoda_Fabia_1.4_TDI_PD_Classic       0
Angebot

     price  abtest  vehicleType  yearOfRegistration    gearbox
powerPS  \
0    480.0       0            0                1993    manuell
0.0
1  16275.0       0            5                2011    manuell
190.0
2   9800.0       0            6                2004  automatik
163.0
3   1500.0       0            1                2001    manuell
75.0
4   3600.0       0            1                2008    manuell
69.0

   model  kilometer  monthOfRegistration  fuelType       brand  \
0   golf   150000.0                    0         0  volkswagen
1   golf   125000.0                    5         1        audi
2  grand   125000.0                    8         1        jeep
3   golf   150000.0                    6         0  volkswagen
4  fabia    90000.0                    7         1       skoda

   notRepairedDamage          dateCreated  nrOfPictures  postalCode  \
0                  0  2016-03-24 00:00:00             0       70435
1                  1  2016-03-24 00:00:00             0       66954
2                  0  2016-03-14 00:00:00             0       90480
3                  0  2016-03-17 00:00:00             0       91074
4                  0  2016-03-31 00:00:00             0       60437
```

```
              lastSeen
0  2016-04-07 03:16:57
1  2016-04-07 01:46:50
2  2016-04-05 12:47:46
3  2016-03-17 17:40:17
4  2016-04-06 10:17:21
```

```
#converting all catogorical columns into numerical columns using
get_dummies function
Fe_df_cleaned=pd.get_dummies(data_df,columns=['offerType','gearbox'],d
rop_first=True)
Fe_df_cleaned.head()
```

```
           dateCrawled                              name  seller
price  \
0  2016-03-24 11:52:17                       Golf_3_1.6       0
480.0
1  2016-03-24 10:58:45             A5_Sportback_2.7_Tdi       0
16275.0
2  2016-03-14 12:52:21  Jeep_Grand_Cherokee_"Overland"       0
9800.0
3  2016-03-17 16:54:04               GOLF_4_1_4__3TÜRER       0
1500.0
4  2016-03-31 17:25:20  Skoda_Fabia_1.4_TDI_PD_Classic       0
3600.0

   abtest  vehicleType  yearOfRegistration  powerPS  model  kilometer
\
0       0            0                1993      0.0   golf   150000.0

1       0            5                2011    190.0   golf   125000.0

2       0            6                2004    163.0  grand   125000.0

3       0            1                2001     75.0   golf   150000.0

4       0            1                2008     69.0  fabia    90000.0


   monthOfRegistration  fuelType       brand  notRepairedDamage  \
0                    0         0  volkswagen                  0
1                    5         1        audi                  1
2                    8         1        jeep                  0
3                    6         0  volkswagen                  0
4                    7         1       skoda                  0

           dateCreated  nrOfPictures  postalCode             lastSeen
\
0  2016-03-24 00:00:00             0       70435  2016-04-07 03:16:57
```

```
1  2016-03-24 00:00:00             0       66954  2016-04-07 01:46:50

2  2016-03-14 00:00:00             0       90480  2016-04-05 12:47:46

3  2016-03-17 00:00:00             0       91074  2016-03-17 17:40:17

4  2016-03-31 00:00:00             0       60437  2016-04-06 10:17:21
```

```
   offerType_Gesuch  gearbox_manuell
0                 0                1
1                 0                1
2                 0                0
3                 0                1
4                 0                1
```

```
#shape of the dataset after label encoding
Fe_df_cleaned.shape
```

```
(371528, 20)
```

```
Fe_df_cleaned.columns
```

```
Index(['dateCrawled', 'name', 'seller', 'price', 'abtest',
'vehicleType',
       'yearOfRegistration', 'powerPS', 'model', 'kilometer',
       'monthOfRegistration', 'fuelType', 'brand',
'notRepairedDamage',
       'dateCreated', 'nrOfPictures', 'postalCode', 'lastSeen',
       'offerType_Gesuch', 'gearbox_manuell'],
      dtype='object')
```

```
#removing unncessary columns in the dataset
main_df=Fe_df_cleaned.drop(columns=['dateCrawled','dateCreated','name'
,'lastSeen','brand','model'],axis=1)
main_df.head()
```

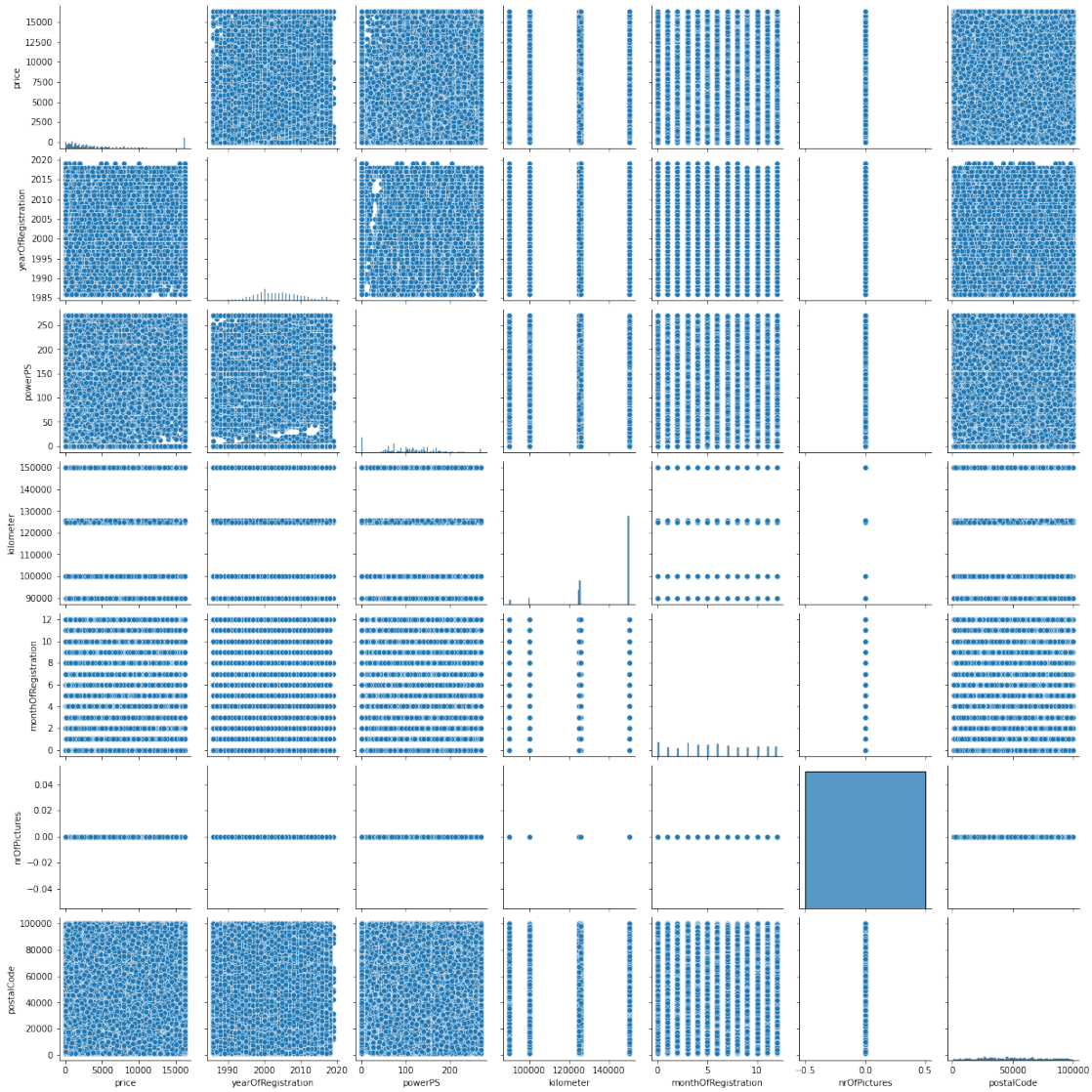
```
   seller    price  abtest  vehicleType  yearOfRegistration
powerPS  \
0       0    480.0       0            0                1993      0.0

1       0  16275.0       0            5                2011    190.0

2       0   9800.0       0            6                2004    163.0

3       0   1500.0       0            1                2001     75.0

4       0   3600.0       0            1                2008     69.0
```

```
   kilometer  monthOfRegistration  fuelType  notRepairedDamage
nrOfPictures  \
0   150000.0                    0         0                  0
0
1   125000.0                    5         1                  1
0
2   125000.0                    8         1                  0
0
3   150000.0                    6         0                  0
0
4    90000.0                    7         1                  0
0

   postalCode  offerType_Gesuch  gearbox_manuell
0       70435                 0                1
1       66954                 0                1
2       90480                 0                0
3       91074                 0                1
4       60437                 0                1
```

```
#multivariate analysis
plt.figure(figsize=(15,5))
sns.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x7f56e8088fd0>
```

```
<Figure size 1080x360 with 0 Axes>
```




```
#dividing the dataset into dependent and independent feature
Independent=main_df.drop(['price'],axis=1)
Dependent=main_df['price']
Independent.head()
```

	seller	abtest	vehicleType	yearOfRegistration	powerPS	kilometer
0	0	0	0	1993	0.0	150000.0
1	0	0	5	2011	190.0	125000.0
2	0	0	6	2004	163.0	125000.0
3	0	0	1	2001	75.0	150000.0
4	0	0	1	2008	69.0	90000.0

```
   monthOfRegistration  fuelType  notRepairedDamage  nrOfPictures
postalCode  \
0                    0         0                  0             0
70435
1                    5         1                  1             0
66954
2                    8         1                  0             0
90480
3                    6         0                  0             0
91074
4                    7         1                  0             0
60437

   offerType_Gesuch  gearbox_manuell
0                 0                1
1                 0                1
2                 0                0
3                 0                1
4                 0                1

Dependent.head()

0      480.0
1    16275.0
2     9800.0
3     1500.0
4     3600.0
Name: price, dtype: float64
```