predicting car price using linear regression algorithm

February 19, 2023

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
df=pd.read_table('https://archive.ics.uci.edu/ml/machine-learning-databases/
↪autos/imports-85.data',header=None,sep=',',names=['symboling','normalize␣
↪loss','make','fuel type','aspiriation','no.of.
↪doors','body-style','drive-wheels','engine-location','wheel-base','length','width','height'
↪of.
↪cylinders','engine-size','fuel-system','bore','stroke','compression-ratio','horsepower','pe
```

[3]:
```
df.head()
```

[3]:

	symboling	normalize loss	make	fuel type	aspiriation	no.of.doors	\
0	3	?	alfa-romero	gas	std	two	
1	3	?	alfa-romero	gas	std	two	
2	1	?	alfa-romero	gas	std	two	
3	2	164	audi	gas	std	four	
4	2	164	audi	gas	std	four	

	body-style	drive-wheels	engine-location	wheel-base	…	engine-size	\
0	convertible	rwd	front	88.6	…	130	
1	convertible	rwd	front	88.6	…	130	
2	hatchback	rwd	front	94.5	…	152	
3	sedan	fwd	front	99.8	…	109	
4	sedan	4wd	front	99.4	…	136	

	fuel-system	bore	stroke	compression-ratio	horsepower	peak-rpm	city-mpg	\
0	mpfi	3.47	2.68	9.0	111	5000	21	
1	mpfi	3.47	2.68	9.0	111	5000	21	
2	mpfi	2.68	3.47	9.0	154	5000	19	
3	mpfi	3.19	3.40	10.0	102	5500	24	
4	mpfi	3.19	3.40	8.0	115	5500	18	

	highway-mpg	price

```
0          27  13495
1          27  16500
2          26  16500
3          30  13950
4          22  17450

[5 rows x 26 columns]
```

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          205 non-null    int64
 1   normalize loss     205 non-null    object
 2   make               205 non-null    object
 3   fuel type          205 non-null    object
 4   aspiriation        205 non-null    object
 5   no.of.doors        205 non-null    object
 6   body-style         205 non-null    object
 7   drive-wheels       205 non-null    object
 8   engine-location    205 non-null    object
 9   wheel-base         205 non-null    float64
 10  length             205 non-null    float64
 11  width              205 non-null    float64
 12  height             205 non-null    float64
 13  weight             205 non-null    int64
 14  engine-type        205 non-null    object
 15  no.of.cylinders    205 non-null    object
 16  engine-size        205 non-null    int64
 17  fuel-system        205 non-null    object
 18  bore               205 non-null    object
 19  stroke             205 non-null    object
 20  compression-ratio  205 non-null    float64
 21  horsepower         205 non-null    object
 22  peak-rpm           205 non-null    object
 23  city-mpg           205 non-null    int64
 24  highway-mpg        205 non-null    int64
 25  price              205 non-null    object
dtypes: float64(5), int64(5), object(16)
memory usage: 41.8+ KB
```

1 handling missing value

[5]:
```
df['normalize loss'].value_counts()
```

[5]:
```
?      41
161    11
91      8
150     7
134     6
128     6
104     6
85      5
94      5
65      5
102     5
74      5
168     5
103     5
95      5
106     4
93      4
118     4
148     4
122     4
83      3
125     3
154     3
115     3
137     3
101     3
119     2
87      2
89      2
192     2
197     2
158     2
81      2
188     2
194     2
153     2
129     2
108     2
110     2
164     2
145     2
113     2
256     1
```

```
107     1
90      1
231     1
142     1
121     1
78      1
98      1
186     1
77      1
Name: normalize loss, dtype: int64
```

[6]:
```
df['normalize loss'].replace('?',np.nan,inplace=True)
```

[7]:
```
from sklearn.impute import SimpleImputer
```

[8]:
```
si=SimpleImputer()
```

[9]:
```
df[['normalize loss']]=si.fit_transform(df[['normalize loss']])
```

[10]:
```
df['normalize loss']=df['normalize loss'].astype('int64')
```

[11]:
```
df['normalize loss'].value_counts()
```

[11]:
```
122    45
161    11
91      8
150     7
128     6
134     6
104     6
95      5
102     5
103     5
74      5
85      5
65      5
94      5
168     5
106     4
148     4
118     4
93      4
83      3
101     3
115     3
154     3
125     3
```

```
137     3
108     2
87      2
119     2
194     2
197     2
89      2
158     2
192     2
113     2
188     2
81      2
110     2
145     2
129     2
164     2
153     2
186     1
107     1
78      1
231     1
77      1
142     1
98      1
121     1
90      1
256     1
Name: normalize loss, dtype: int64
```

[12]:
```
df['horsepower'].value_counts()
```

[12]:
```
68     19
70     11
69     10
116     9
110     8
95      7
88      6
62      6
101     6
160     6
114     6
84      5
97      5
102     5
145     5
82      5
```

```
76      5
111     4
92      4
123     4
86      4
90      3
73      3
85      3
207     3
182     3
121     3
152     3
112     2
56      2
161     2
156     2
94      2
52      2
?       2
162     2
155     2
184     2
100     2
176     2
55      1
262     1
134     1
115     1
140     1
48      1
58      1
60      1
78      1
135     1
200     1
64      1
120     1
72      1
154     1
288     1
143     1
142     1
175     1
106     1
Name: horsepower, dtype: int64
```

[13]:
```
df[df['horsepower']=='?']
```

[13]:
```
     symboling  normalize loss     make fuel type aspiriation no.of.doors  \
130          0             122  renault       gas         std        four
131          2             122  renault       gas         std         two

    body-style drive-wheels engine-location  wheel-base  …  engine-size  \
130      wagon          fwd           front        96.1  …          132
131  hatchback          fwd           front        96.1  …          132

     fuel-system  bore  stroke compression-ratio horsepower  peak-rpm  \
130         mpfi  3.46    3.90               8.7          ?         ?
131         mpfi  3.46    3.90               8.7          ?         ?

    city-mpg highway-mpg price
130       23          31  9295
131       23          31  9895

[2 rows x 26 columns]
```

[14]:
```
df.drop([130,131],axis=0,inplace=True)
```

[15]:
```
df['horsepower']=df['horsepower'].astype('int64')
```

[16]:
```
df['peak-rpm'].unique()
```

[16]:
```
array(['5000', '5500', '5800', '4250', '5400', '5100', '4800', '6000',
       '4750', '4650', '4200', '4350', '4500', '5200', '4150', '5600',
       '5900', '5750', '5250', '4900', '4400', '6600', '5300'],
      dtype=object)
```

[17]:
```
df['peak-rpm']=df['peak-rpm'].astype('int64')
```

[18]:
```
df['price'].value_counts()
```

[18]:
```
?        4
8921     2
18150    2
7898     2
7775     2
        ..
40960    1
45400    1
16503    1
5389     1
22625    1
Name: price, Length: 185, dtype: int64
```

[19]:
```
df[df['price']=='?']
```

[19]:
```
     symboling  normalize loss     make fuel type aspiriation no.of.doors  \
9            0             122     audi       gas       turbo         two
44           1             122    isuzu       gas         std         two
45           0             122    isuzu       gas         std        four
129          1             122  porsche       gas         std         two

    body-style drive-wheels engine-location  wheel-base  …  engine-size  \
9    hatchback          4wd           front        99.5  …          131
44       sedan          fwd           front        94.5  …           90
45       sedan          fwd           front        94.5  …           90
129  hatchback          rwd           front        98.4  …          203

     fuel-system  bore  stroke compression-ratio horsepower  peak-rpm  \
9           mpfi  3.13    3.40               7.0        160      5500
44          2bbl  3.03    3.11               9.6         70      5400
45          2bbl  3.03    3.11               9.6         70      5400
129         mpfi  3.94    3.11              10.0        288      5750

    city-mpg highway-mpg price
9         16          22     ?
44        38          43     ?
45        38          43     ?
129       17          28     ?

[4 rows x 26 columns]
```

[20]:
```
df.drop([9,44,45,129],axis=0,inplace=True)
```

[21]:
```
df['price']=df['price'].astype('int64')
```

[22]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 199 entries, 0 to 204
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          199 non-null    int64
 1   normalize loss     199 non-null    int64
 2   make               199 non-null    object
 3   fuel type          199 non-null    object
 4   aspiriation        199 non-null    object
 5   no.of.doors        199 non-null    object
 6   body-style         199 non-null    object
 7   drive-wheels       199 non-null    object
 8   engine-location    199 non-null    object
 9   wheel-base         199 non-null    float64
 10  length             199 non-null    float64
```

```
 11  width              199 non-null    float64
 12  height             199 non-null    float64
 13  weight             199 non-null    int64
 14  engine-type        199 non-null    object
 15  no.of.cylinders    199 non-null    object
 16  engine-size        199 non-null    int64
 17  fuel-system        199 non-null    object
 18  bore               199 non-null    object
 19  stroke             199 non-null    object
 20  compression-ratio  199 non-null    float64
 21  horsepower         199 non-null    int64
 22  peak-rpm           199 non-null    int64
 23  city-mpg           199 non-null    int64
 24  highway-mpg        199 non-null    int64
 25  price              199 non-null    int64
dtypes: float64(5), int64(9), object(12)
memory usage: 42.0+ KB
```

[23]:
```
df['bore'].value_counts()
```

[23]:
```
3.62    23
3.19    20
3.15    15
2.97    12
3.03    10
3.31     8
3.78     8
3.43     8
3.27     7
2.91     7
3.46     7
3.39     6
3.54     6
3.05     6
3.58     6
3.70     5
3.01     5
3.35     4
?        4
3.17     3
3.59     3
3.74     3
3.47     2
3.24     2
3.63     2
3.50     2
3.80     2
```

```
3.33     2
3.94     1
3.13     1
2.54     1
3.08     1
3.61     1
3.34     1
3.60     1
2.92     1
3.76     1
2.68     1
2.99     1
Name: bore, dtype: int64
```

[24]:
```
df[df['bore']=='?']
```

[24]:
```
    symboling  normalize loss   make fuel type aspiriation no.of.doors  \
55          3             150  mazda       gas         std         two
56          3             150  mazda       gas         std         two
57          3             150  mazda       gas         std         two
58          3             150  mazda       gas         std         two

   body-style drive-wheels engine-location  wheel-base  …  engine-size  \
55  hatchback          rwd           front        95.3  …           70
56  hatchback          rwd           front        95.3  …           70
57  hatchback          rwd           front        95.3  …           70
58  hatchback          rwd           front        95.3  …           80

    fuel-system  bore  stroke compression-ratio horsepower  peak-rpm city-mpg  \
55         4bbl     ?       ?               9.4        101      6000       17
56         4bbl     ?       ?               9.4        101      6000       17
57         4bbl     ?       ?               9.4        101      6000       17
58         mpfi     ?       ?               9.4        135      6000       16

   highway-mpg  price
55          23  10945
56          23  11845
57          23  13645
58          23  15645

[4 rows x 26 columns]
```

[25]:
```
df.drop([55,56,57,58],axis=0,inplace=True)
```

[26]:
```
df['bore']=df['bore'].astype('float64')
```

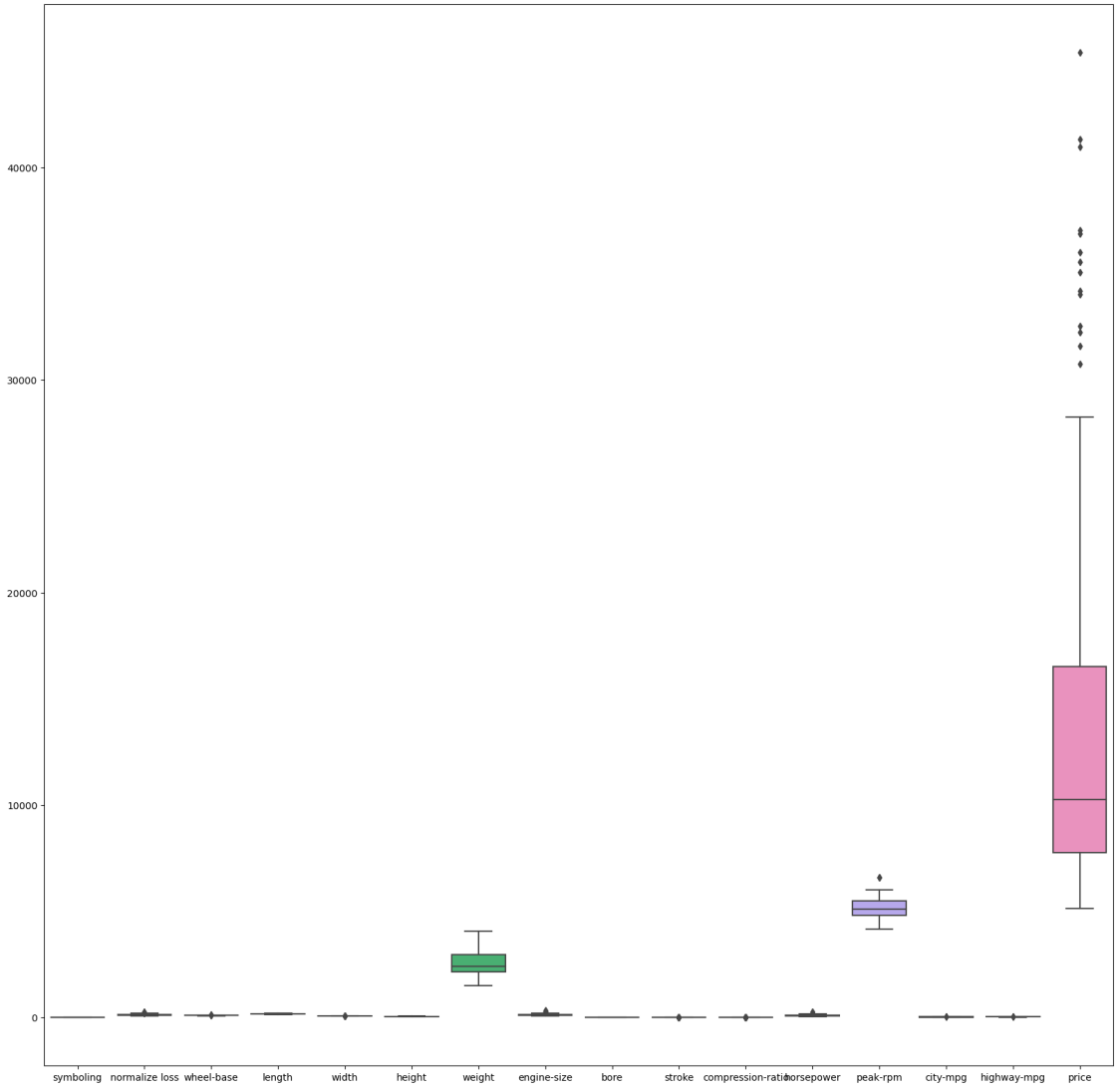
[27]:
```
df['stroke'].value_counts()
```

[27]:
```
3.40    19
3.15    14
3.03    14
3.23    14
3.39    13
2.64    11
3.29     9
3.35     9
3.46     8
3.27     6
3.58     6
3.07     6
3.41     6
3.50     6
3.19     6
3.52     5
3.64     5
3.47     4
3.54     4
3.86     4
3.11     3
2.90     3
3.08     2
2.19     2
2.68     2
3.10     2
4.17     2
2.80     2
3.12     1
3.21     1
3.16     1
2.07     1
2.36     1
2.76     1
3.90     1
2.87     1
Name: stroke, dtype: int64
```

[28]:
```
df['stroke']=df['stroke'].astype('float64')
```

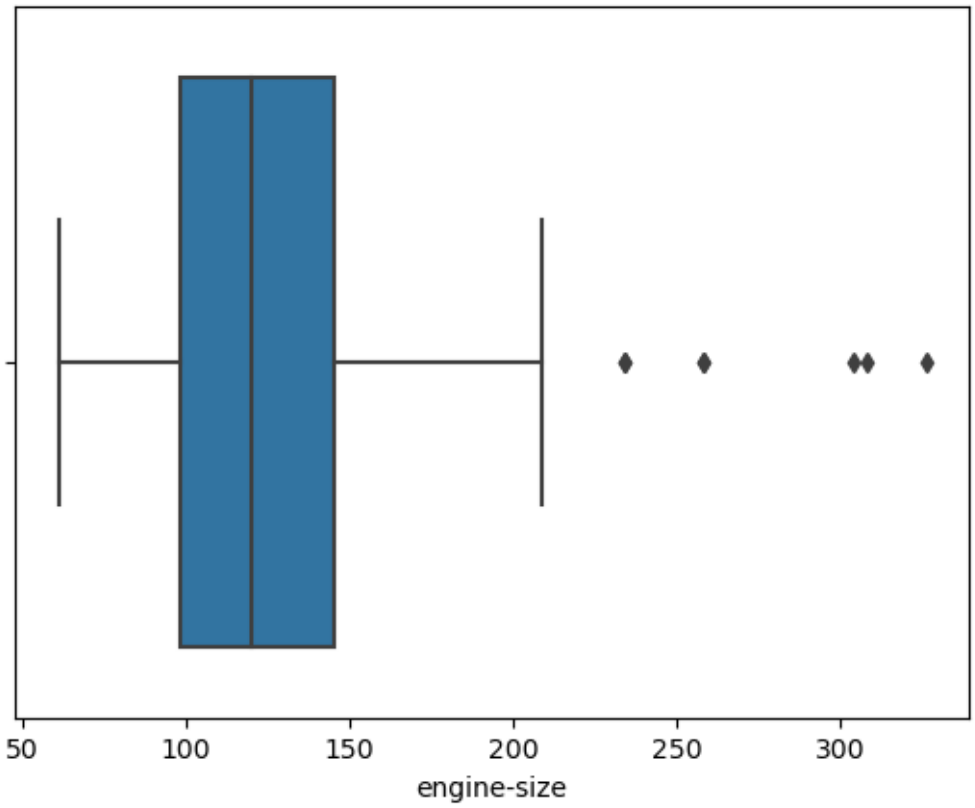
[29]:
```
plt.figure(figsize=(20,20))
sns.boxplot(data=df)
```

[29]:
```
<AxesSubplot:>
```



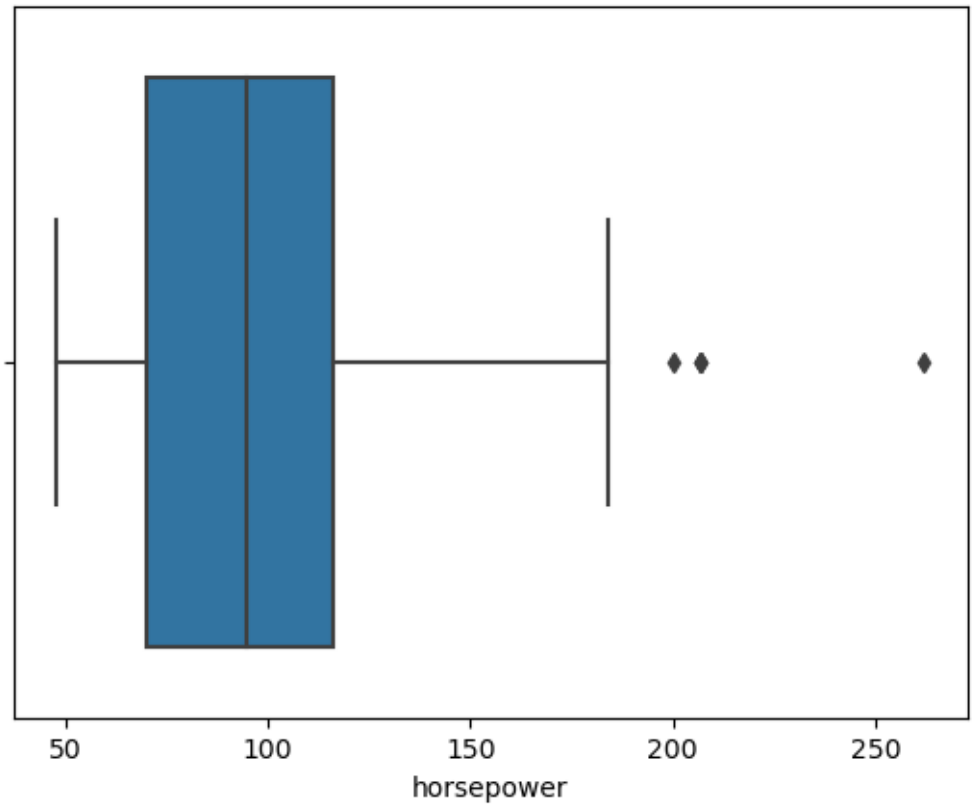
[59]:
```
sns.boxplot(x='engine-size',data=df)
```

[59]: <AxesSubplot:xlabel='engine-size'>



[61]:
```
sns.boxplot(x='horsepower',data=df)
```

[61]:
```
<AxesSubplot:xlabel='horsepower'>
```



[30]:
```
df.corr().style.background_gradient()
```

[30]:
```
<pandas.io.formats.style.Styler at 0x279d4600400>
```

[31]:
```
df.drop(['city-mpg'],axis=1,inplace=True)
```

[60]:
```
df.head()
```

[60]:

	symboling	normalize loss	make	fuel type	aspiriation	no.of.doors	\
0	3	122	0.0	1.0	0.0	2.0	
1	3	122	0.0	1.0	0.0	2.0	
2	1	122	0.0	1.0	0.0	2.0	
3	2	164	1.0	1.0	0.0	1.0	
4	2	164	1.0	1.0	0.0	1.0	

	body-style	drive-wheels	engine-location	wheel-base	…	\
0	0.0	2.0	0.0	88.6	…	
1	0.0	2.0	0.0	88.6	…	
2	2.0	2.0	0.0	94.5	…	
3	3.0	1.0	0.0	99.8	…	
4	3.0	0.0	0.0	99.4	…	

```
   no.of.cylinders  engine-size  fuel-system  bore  stroke  compression-ratio  \
0              2.0          130          4.0  3.47    2.68                9.0
1              2.0          130          4.0  3.47    2.68                9.0
2              3.0          152          4.0  2.68    3.47                9.0
3              2.0          109          4.0  3.19    3.40               10.0
4              1.0          136          4.0  3.19    3.40                8.0

   horsepower  peak-rpm  highway-mpg  price
0         111      5000           27  13495
1         111      5000           27  16500
2         154      5000           26  16500
3         102      5500           30  13950
4         115      5500           22  17450

[5 rows x 25 columns]
```

[33]:
```
catcol=df.select_dtypes('object').columns
```

[34]:
```
catcol
```

[34]:
```
Index(['make', 'fuel type', 'aspiriation', 'no.of.doors', 'body-style',
       'drive-wheels', 'engine-location', 'engine-type', 'no.of.cylinders',
       'fuel-system'],
      dtype='object')
```

[35]:
```
from sklearn.preprocessing import OrdinalEncoder
```

[36]:
```
oe=OrdinalEncoder()
```

[37]:
```
df[catcol]=oe.fit_transform(df[catcol])
```

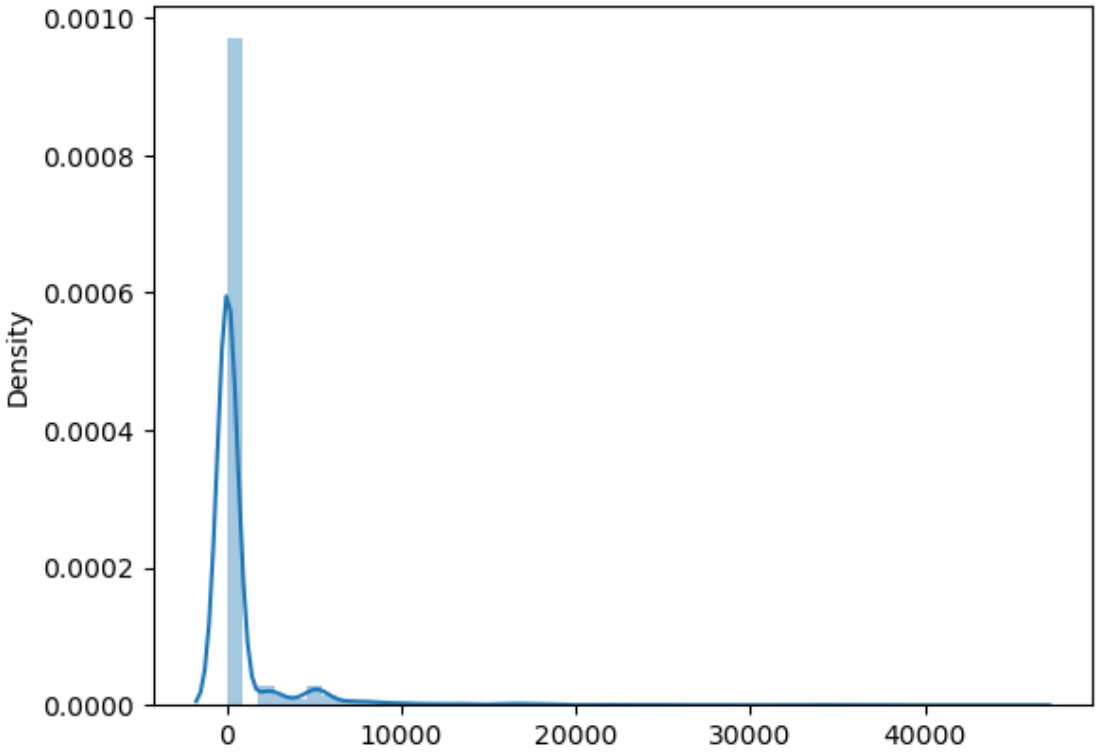
[38]:
```
df.head()
```

[38]:
```
   symboling  normalize loss  make  fuel type  aspiriation  no.of.doors  \
0          3             122   0.0        1.0          0.0          2.0
1          3             122   0.0        1.0          0.0          2.0
2          1             122   0.0        1.0          0.0          2.0
3          2             164   1.0        1.0          0.0          1.0
4          2             164   1.0        1.0          0.0          1.0

   body-style  drive-wheels  engine-location  wheel-base  …  \
0         0.0           2.0              0.0        88.6  …
1         0.0           2.0              0.0        88.6  …
2         2.0           2.0              0.0        94.5  …
3         3.0           1.0              0.0        99.8  …
4         3.0           0.0              0.0        99.4  …
```

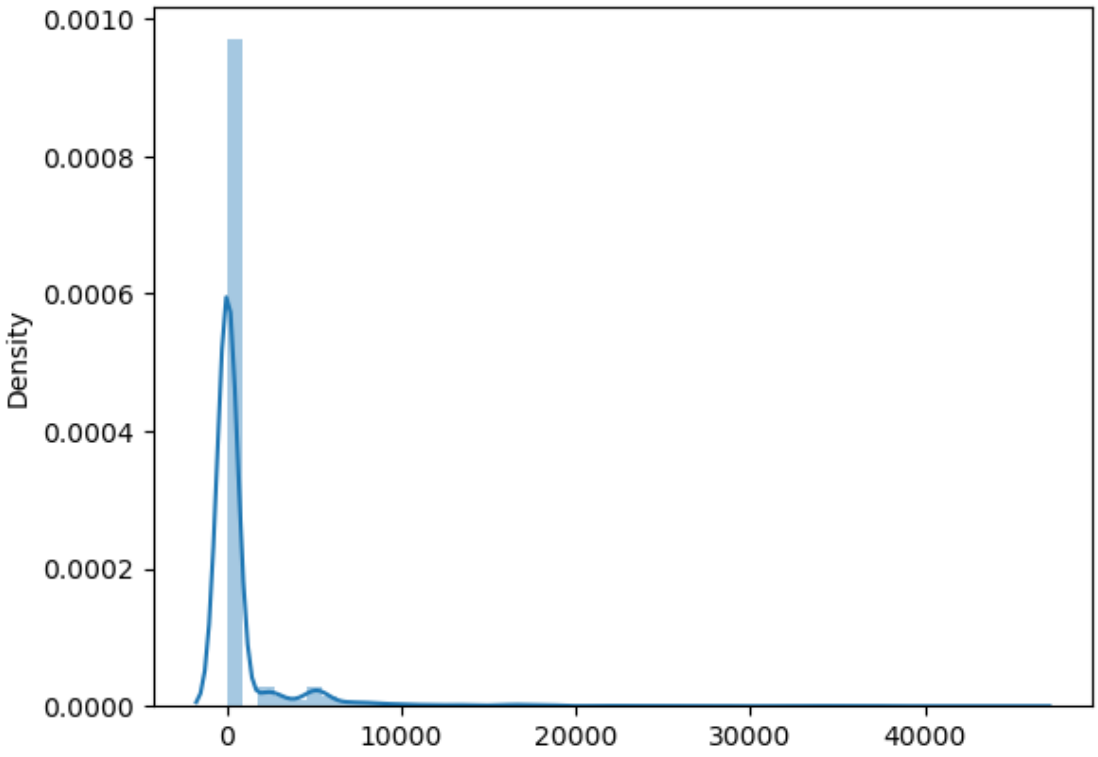
```
   no.of.cylinders  engine-size  fuel-system  bore  stroke  compression-ratio  \
0              2.0          130          4.0  3.47    2.68                9.0
1              2.0          130          4.0  3.47    2.68                9.0
2              3.0          152          4.0  2.68    3.47                9.0
3              2.0          109          4.0  3.19    3.40               10.0
4              1.0          136          4.0  3.19    3.40                8.0

   horsepower  peak-rpm  highway-mpg  price
0         111      5000           27  13495
1         111      5000           27  16500
2         154      5000           26  16500
3         102      5500           30  13950
4         115      5500           22  17450

[5 rows x 25 columns]
```

[57]:
```
from scipy.stats import skew
for i in df:
    print(i)
    sns.distplot(df)
    plt.show()
```

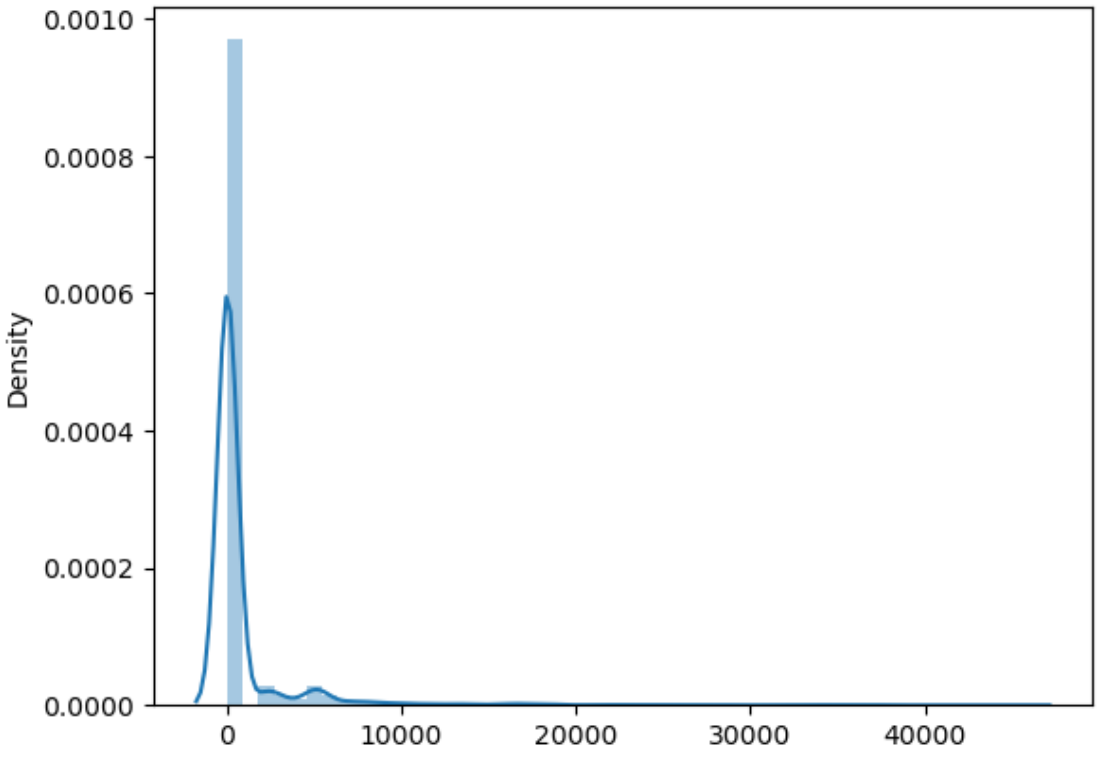
symboling



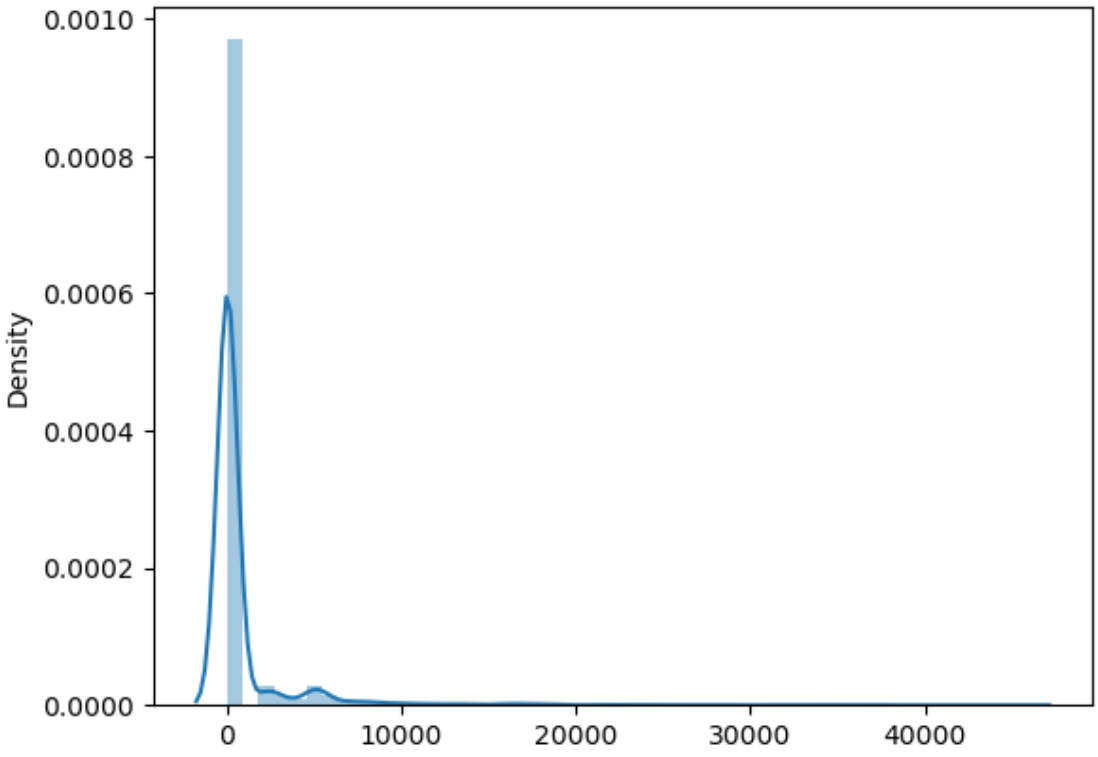
normalize loss



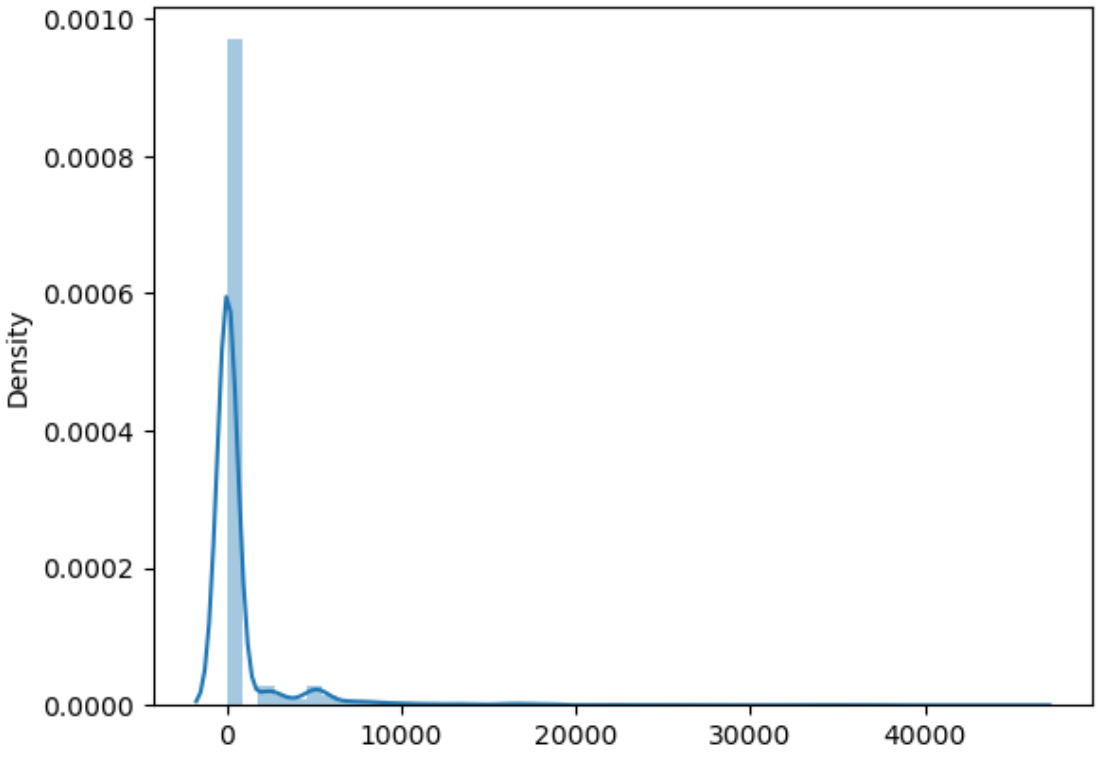
make


0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
Density
0
10000
20000
30000
40000

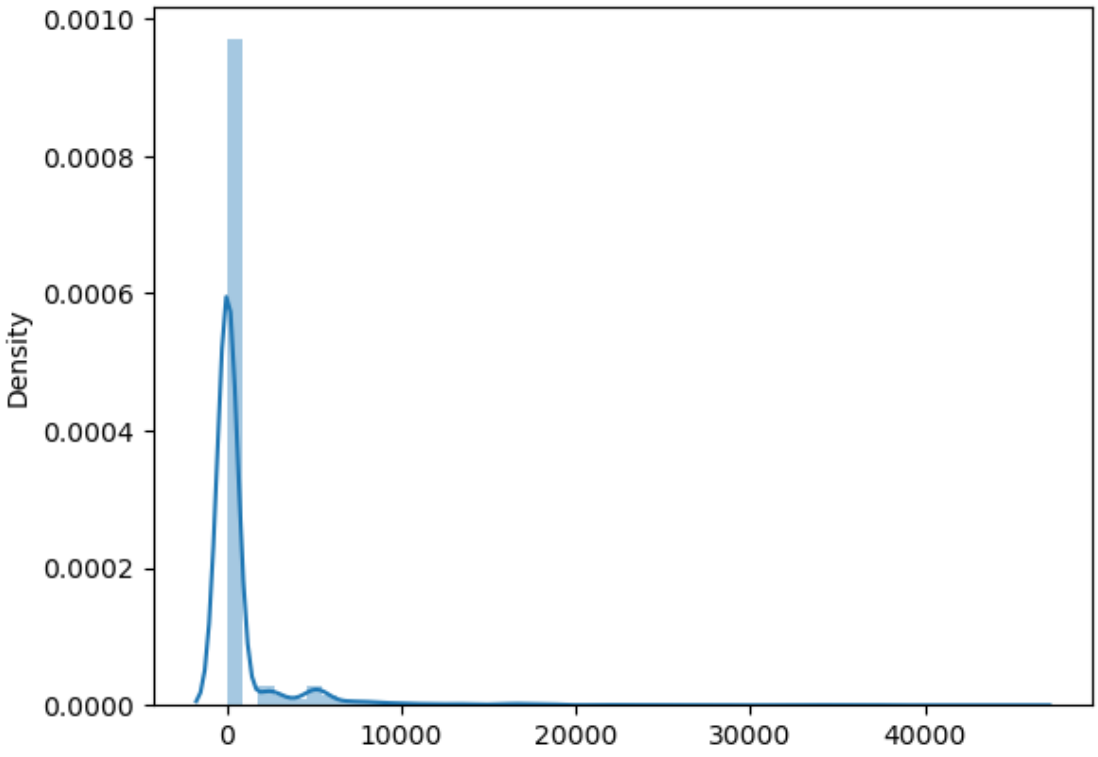
```
fuel type
```



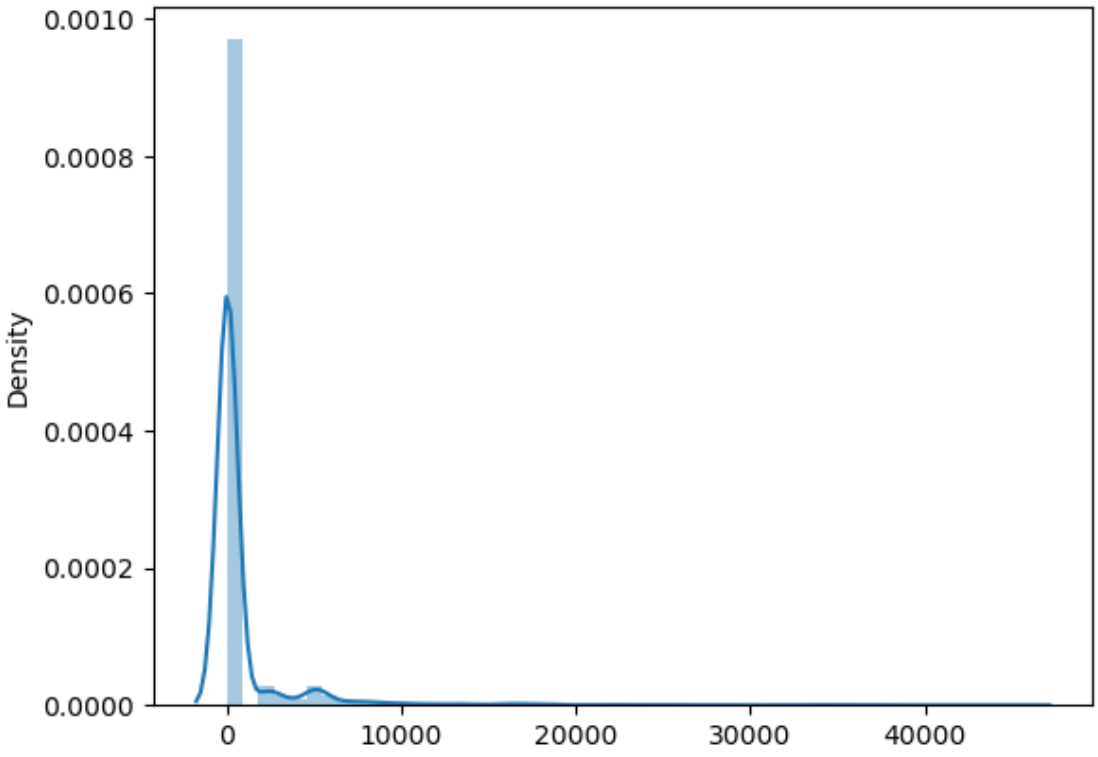

```
aspiriation
```


0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
Density
0
10000
20000
30000
40000

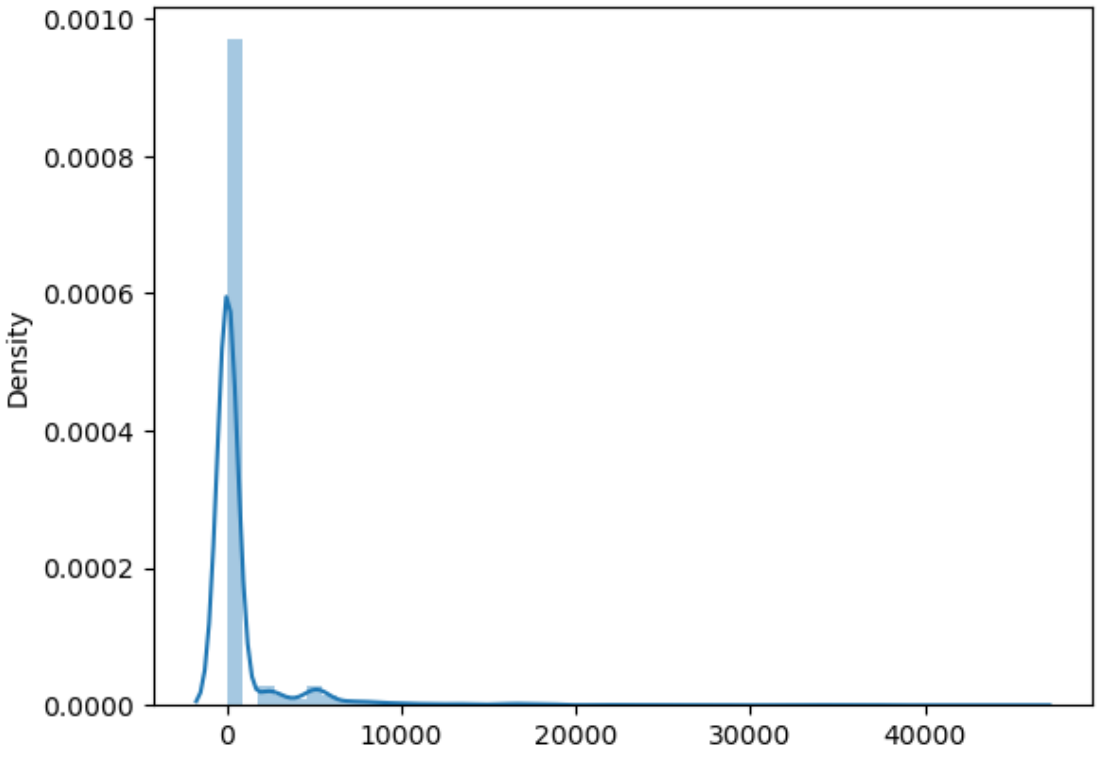
no.of.doors


0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
Density
0
10000
20000
30000
40000

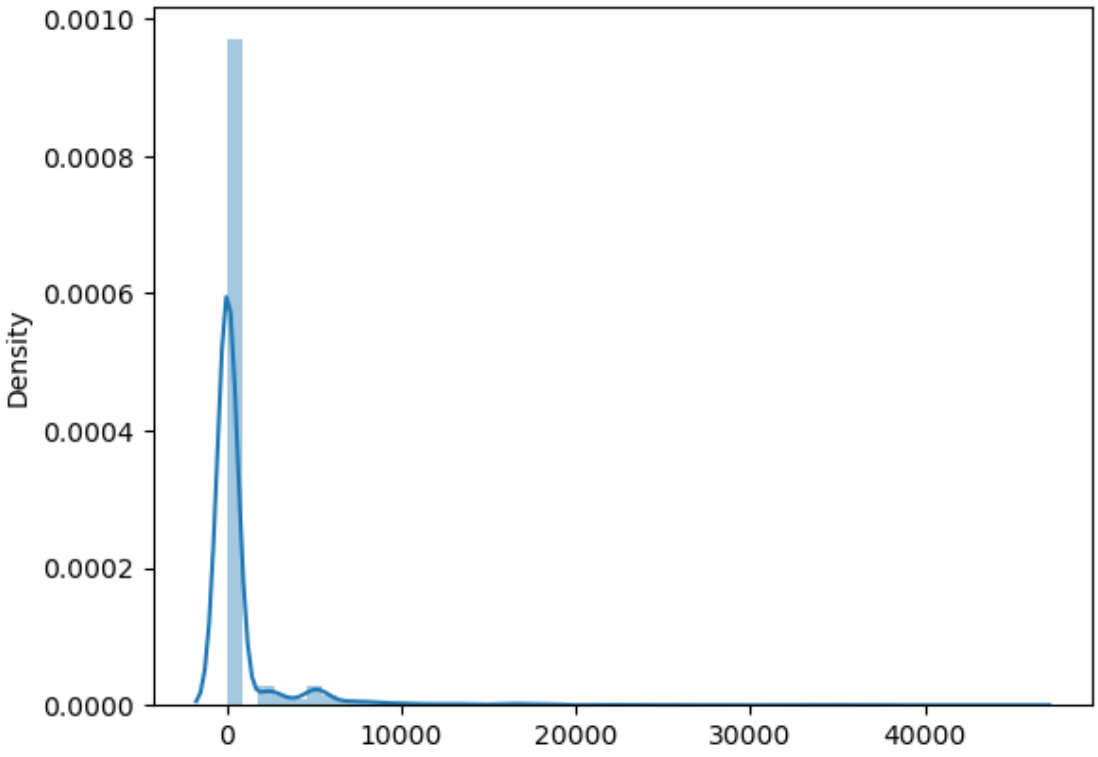
```
body-style
```



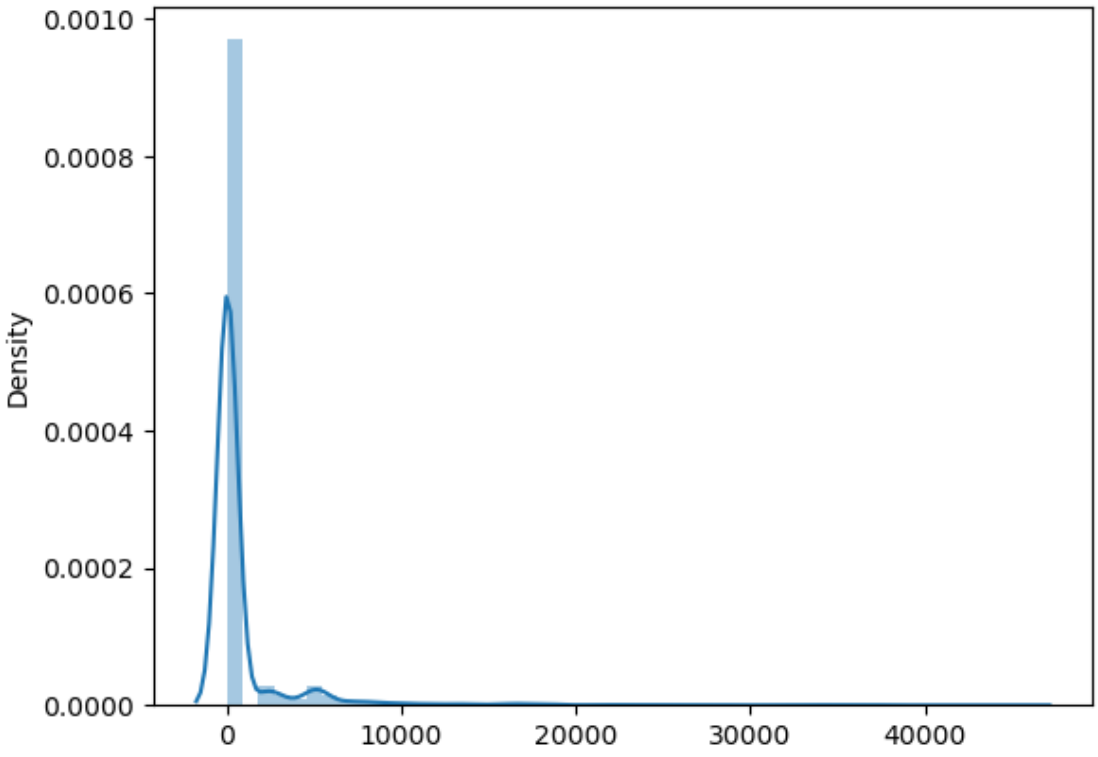

```
drive-wheels
```



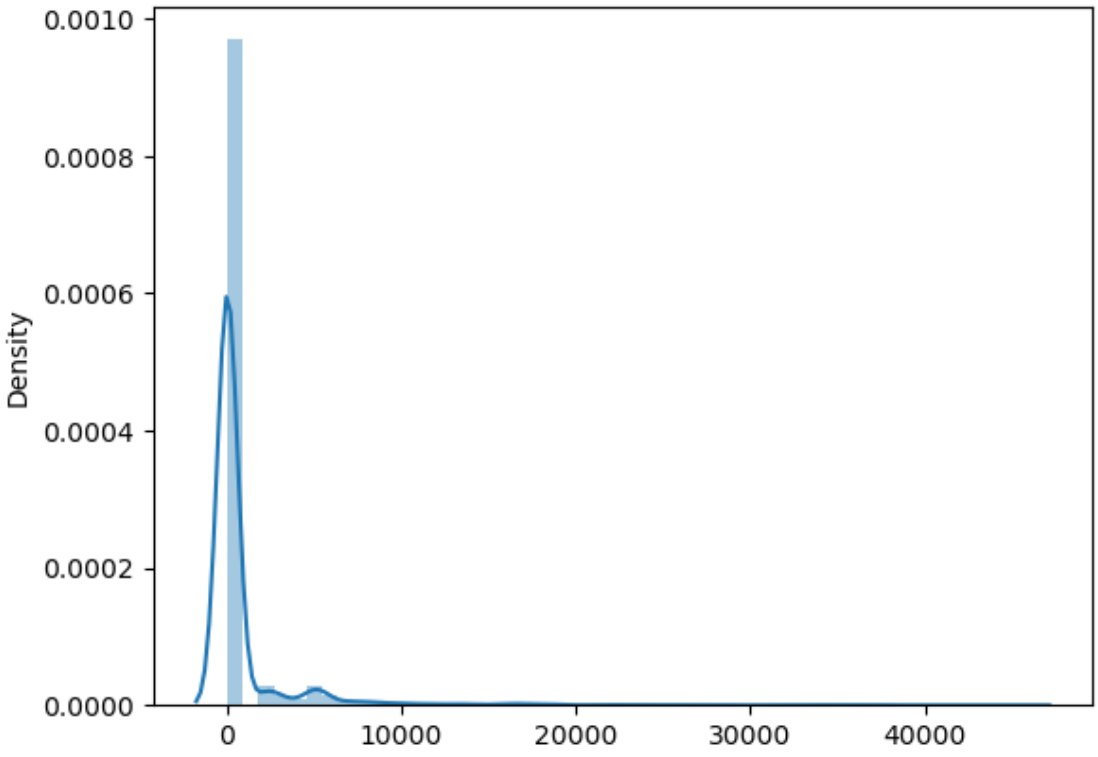
engine-location


0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
Density
0
10000
20000
30000
40000

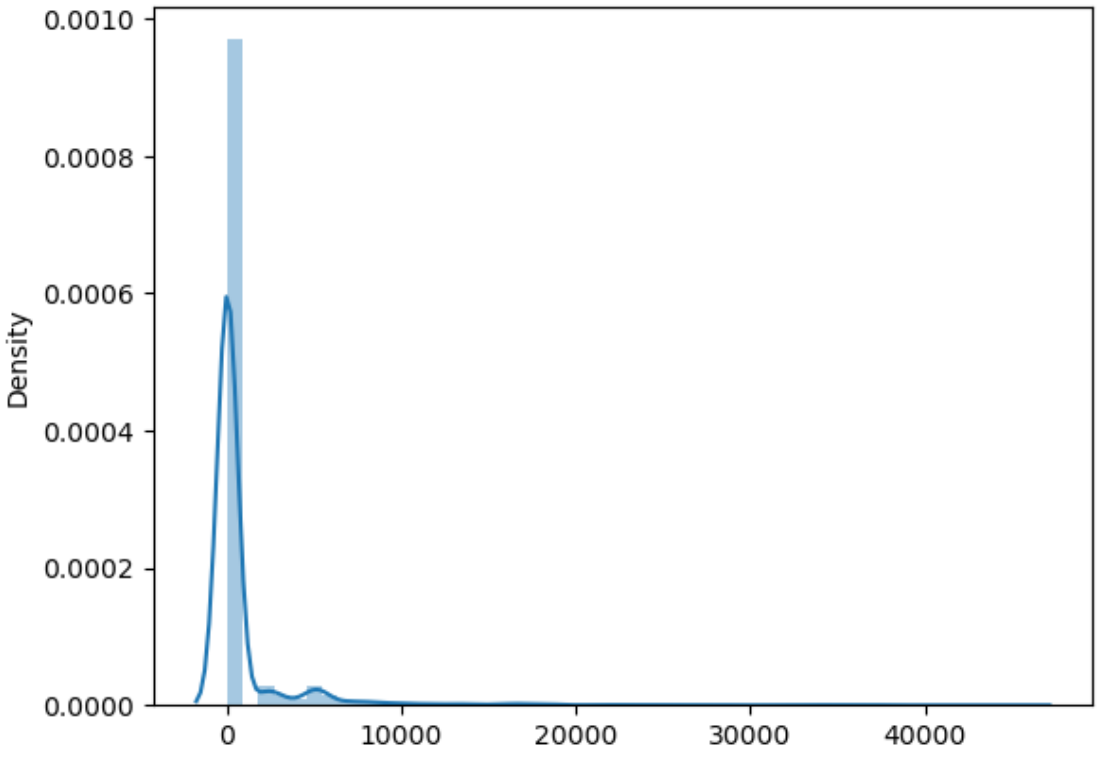
wheel-base


0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
Density
0
10000
20000
30000
40000

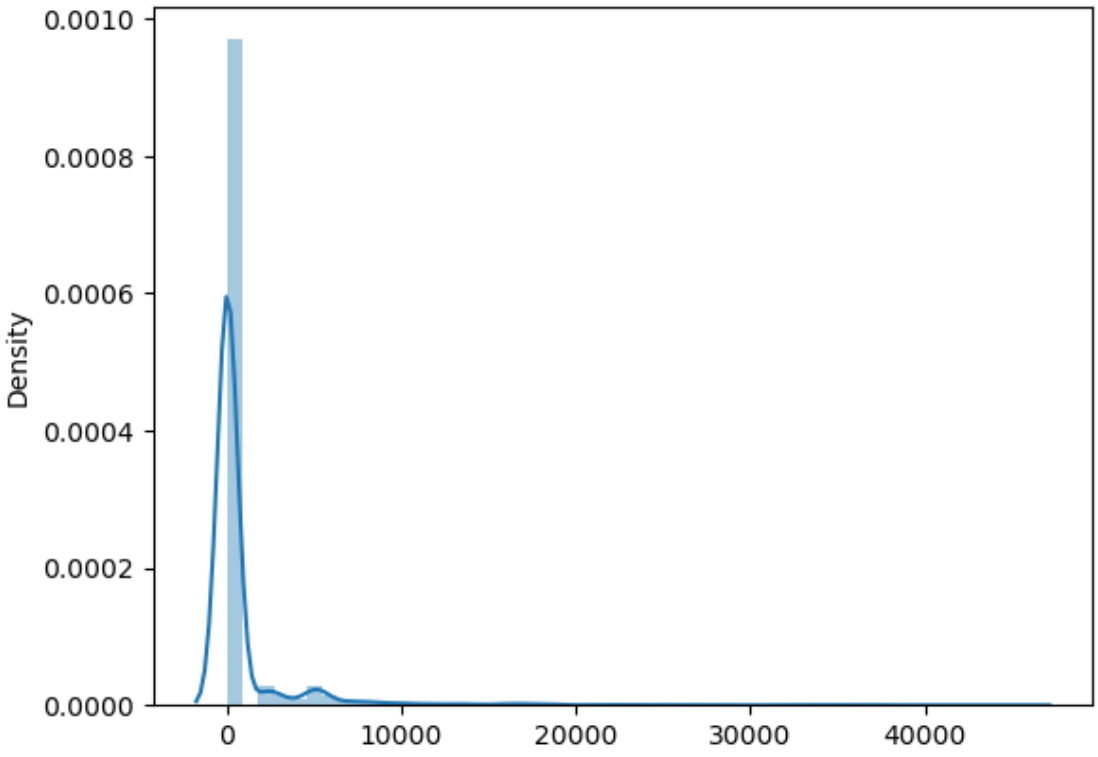
length


0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
Density
0
10000
20000
30000
40000

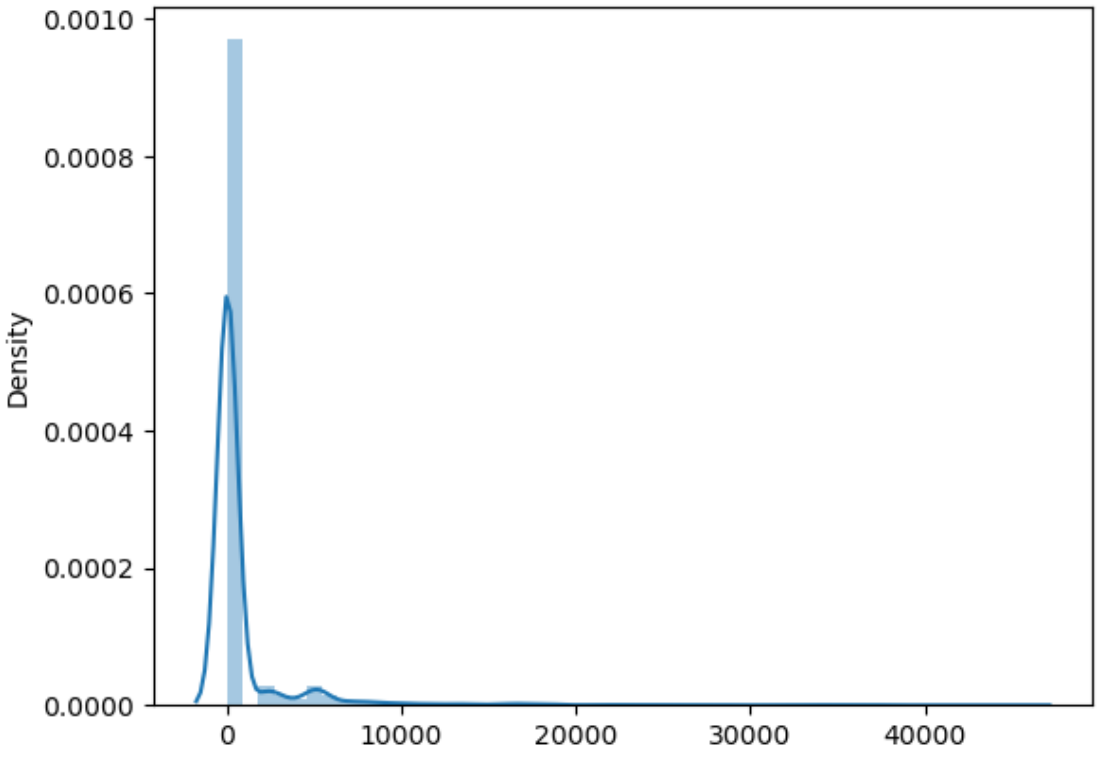
```
width
```


0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
Density
0
10000
20000
30000
40000

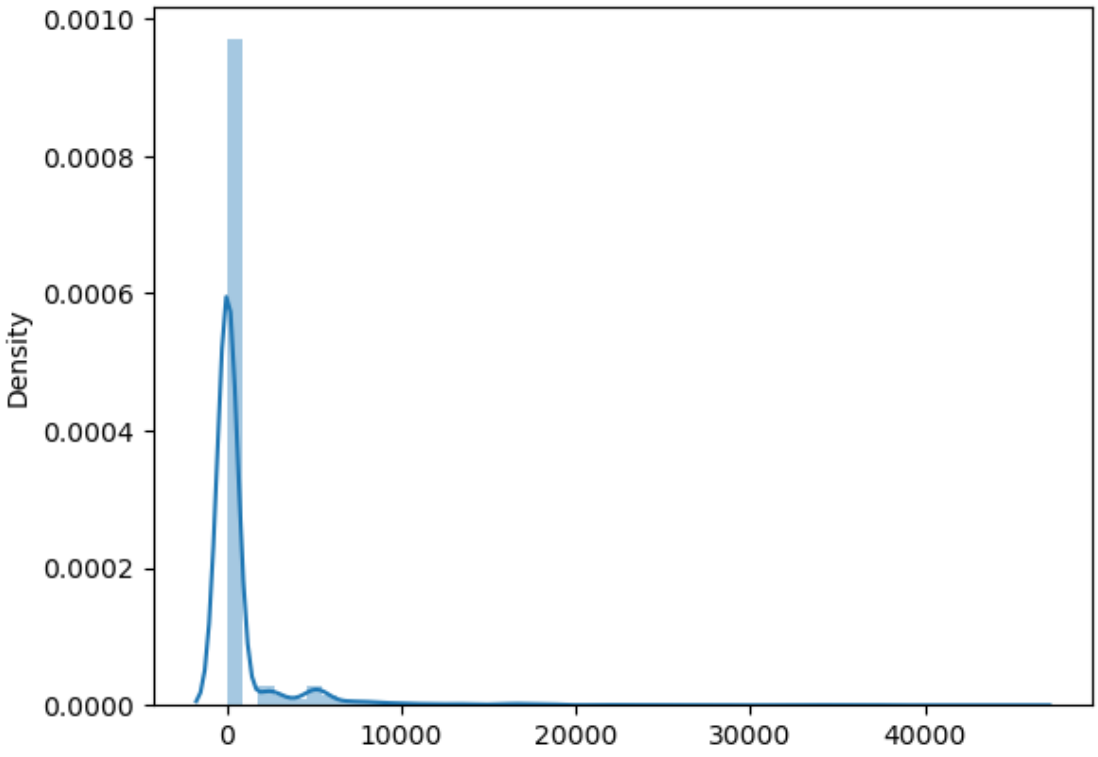
height


0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
Density
0
10000
20000
30000
40000

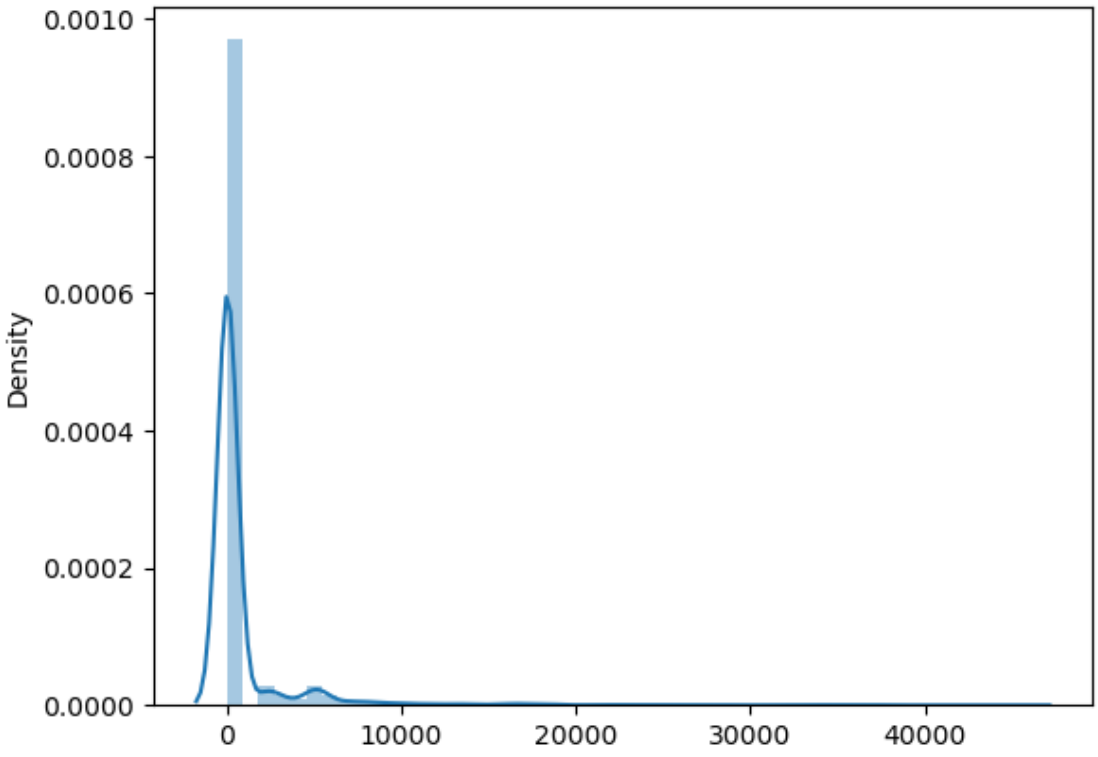
weight


0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
Density
0
10000
20000
30000
40000

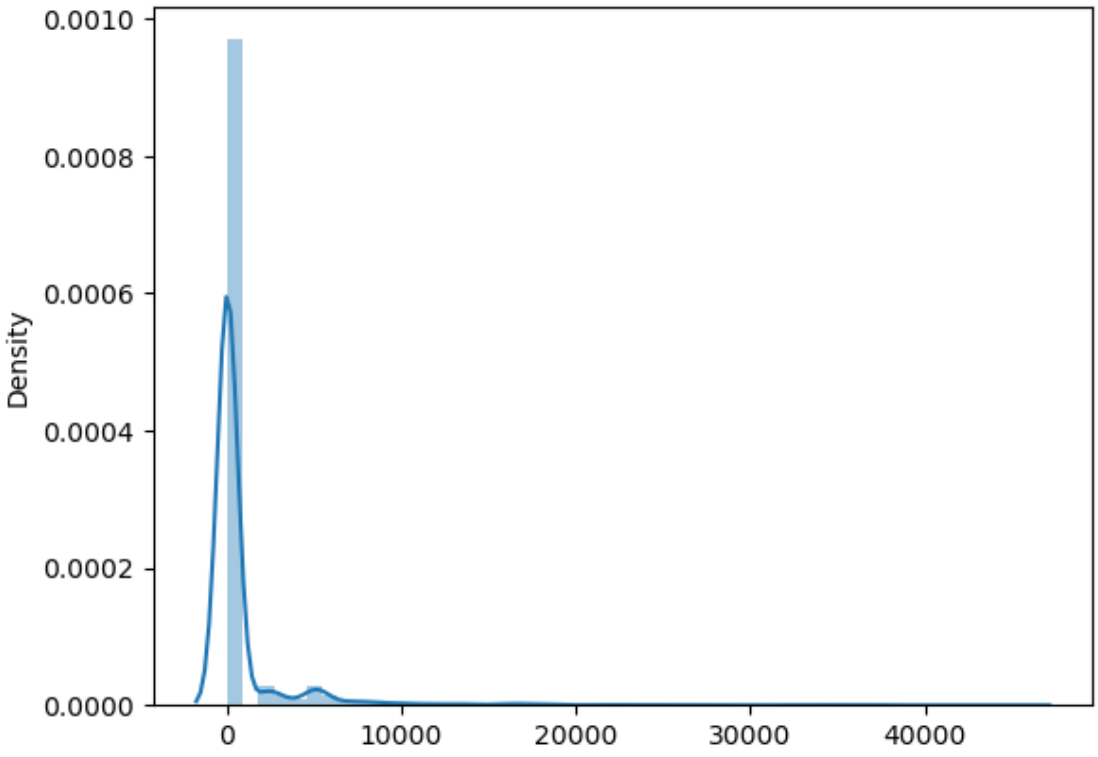
engine-type



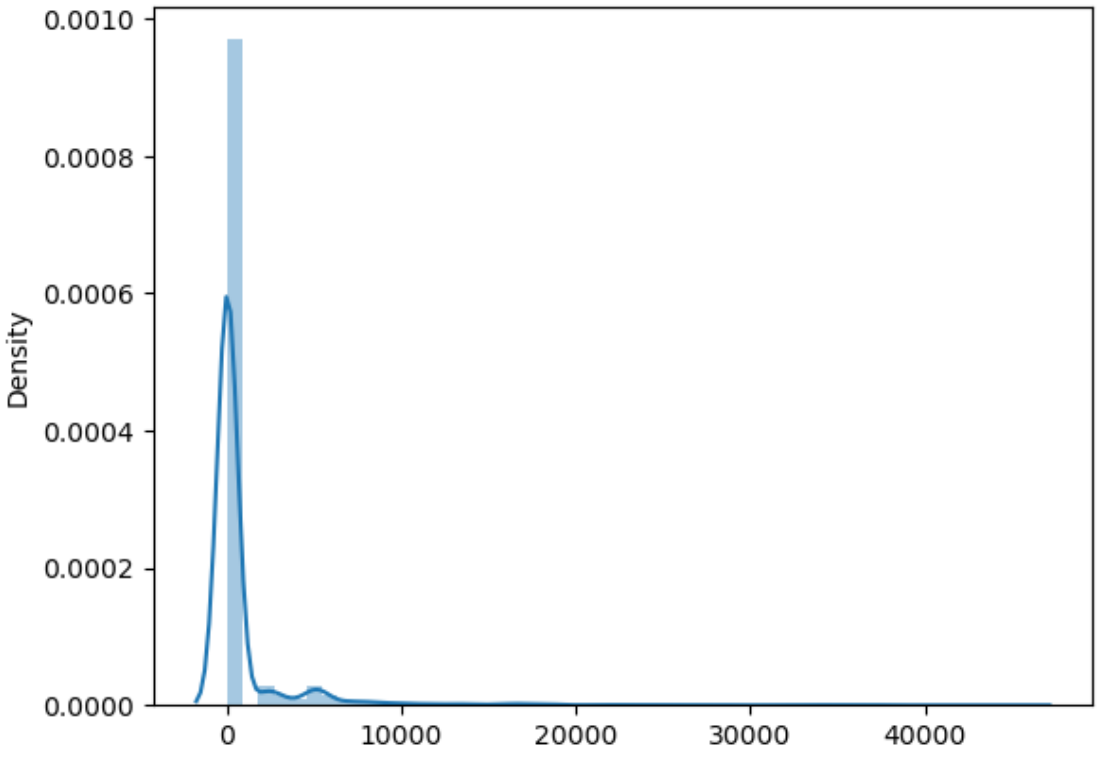
no.of.cylinders


0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
Density
0
10000
20000
30000
40000

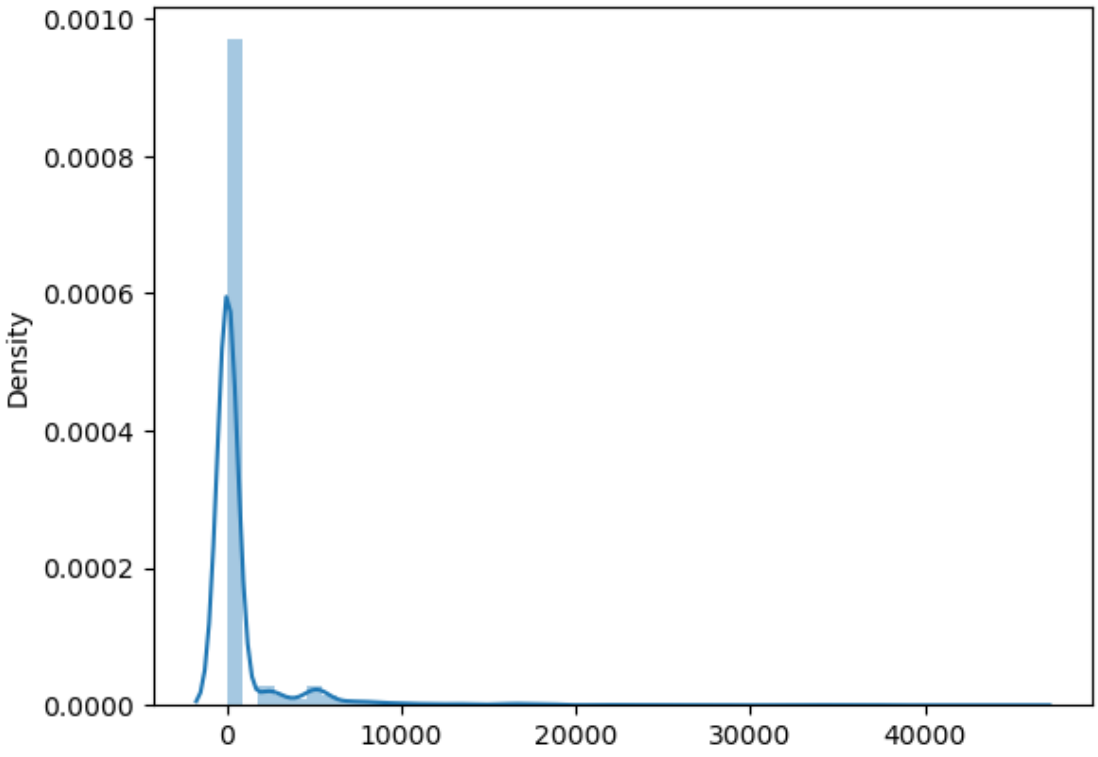
engine-size


0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
Density
0
10000
20000
30000
40000

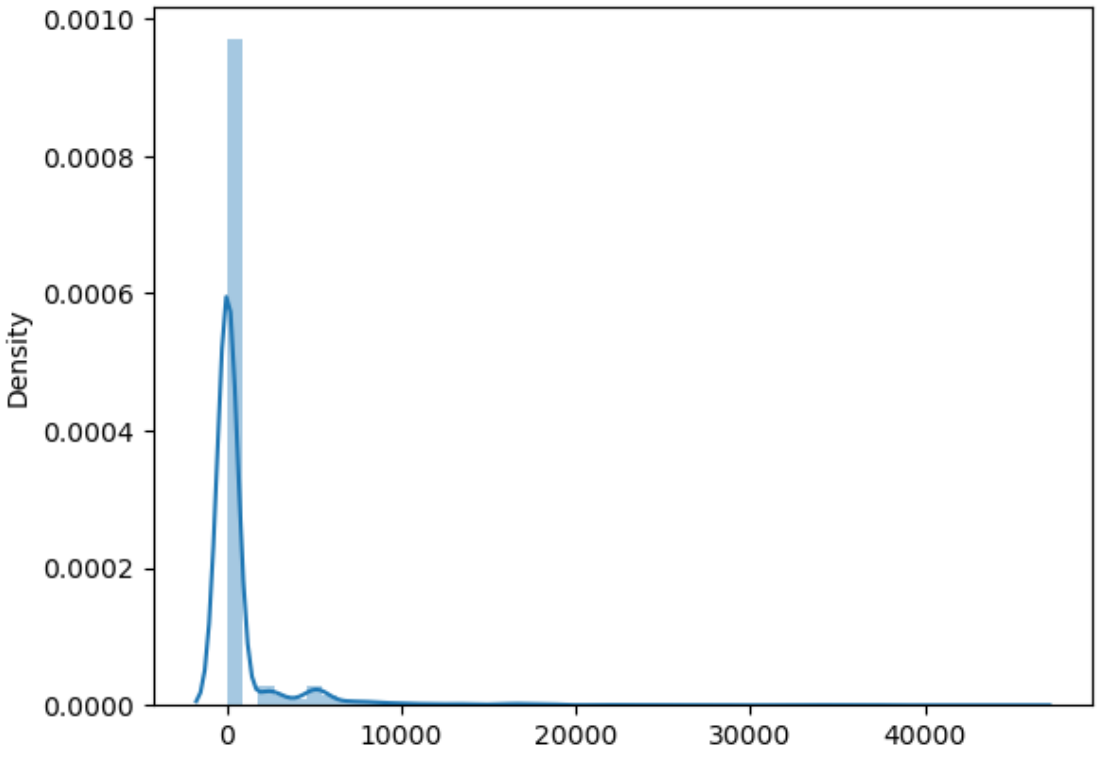
```
fuel-system
```


0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
Density
0
10000
20000
30000
40000

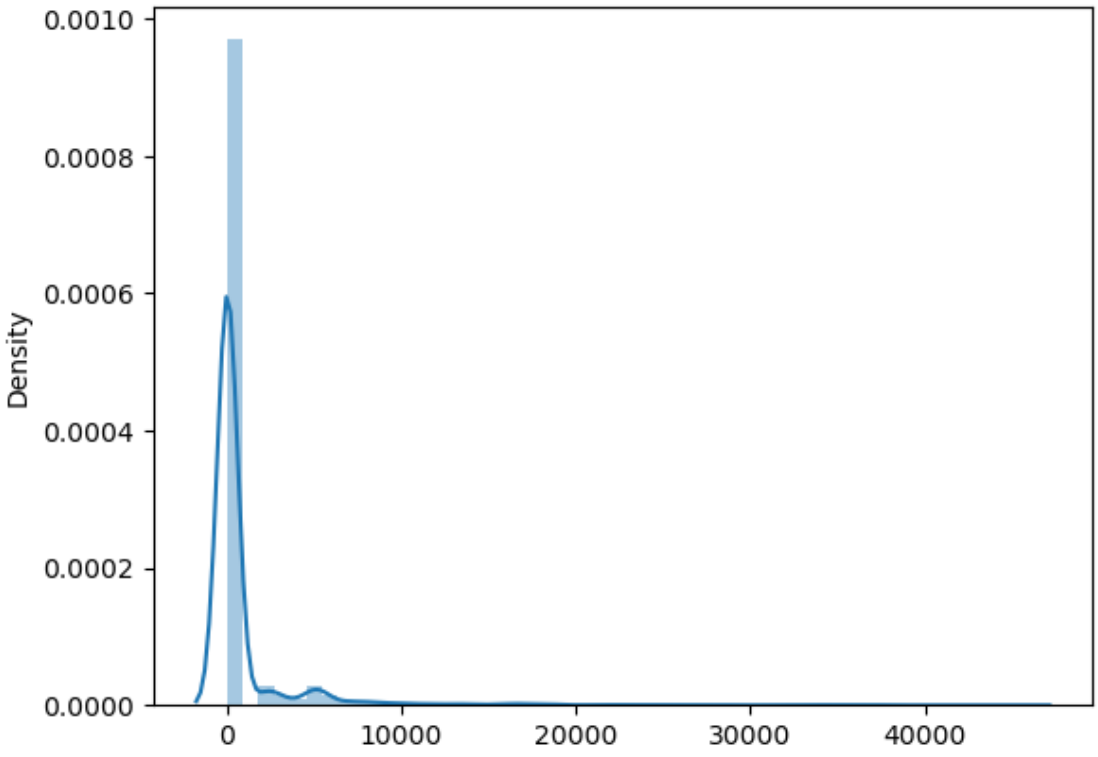
bore


0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
Density
0
10000
20000
30000
40000

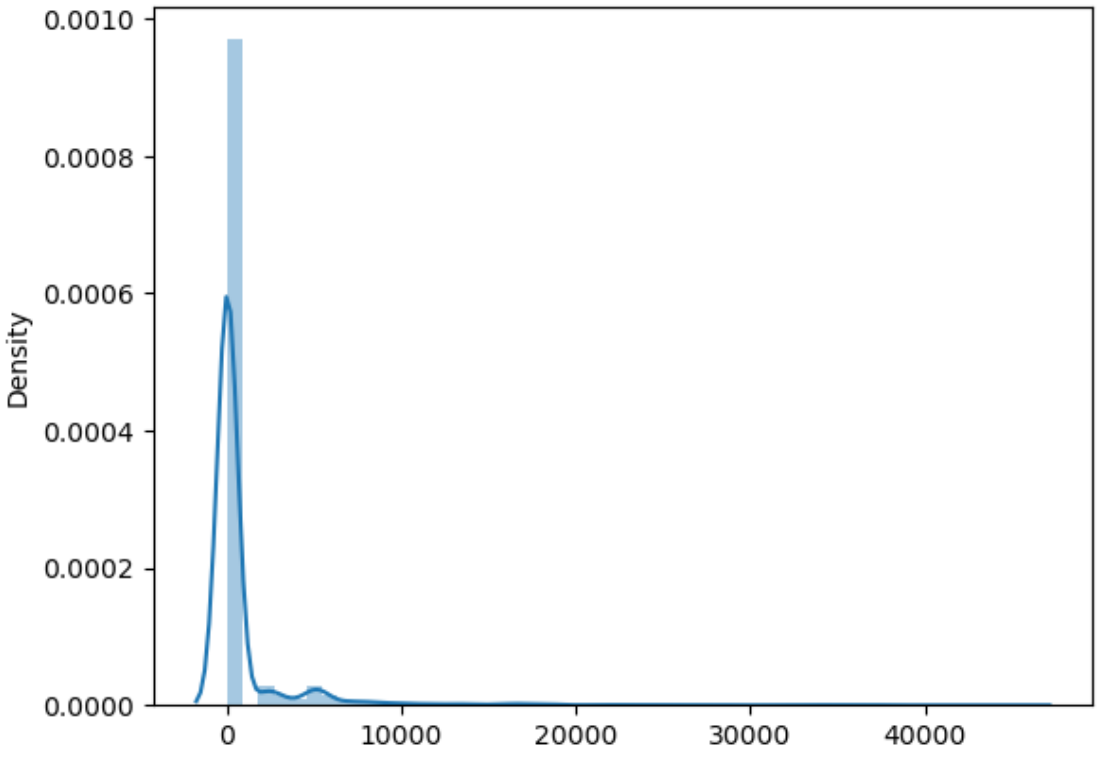
```
stroke
```



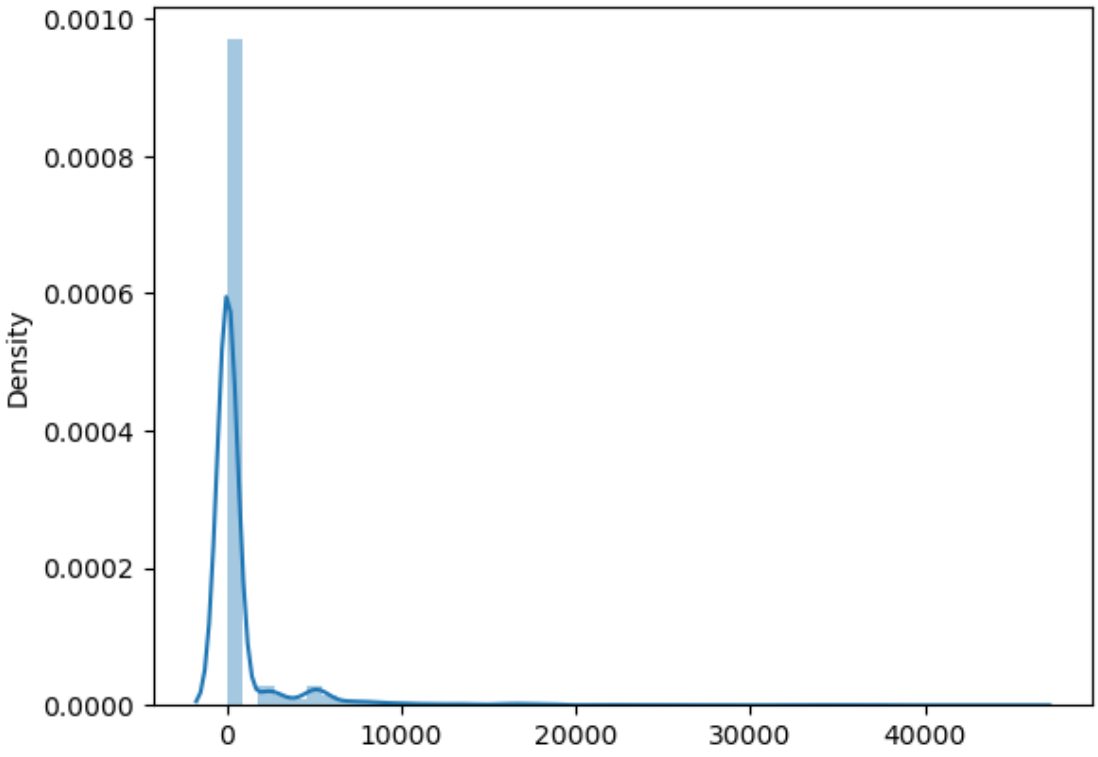
compression-ratio


0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
Density
0
10000
20000
30000
40000

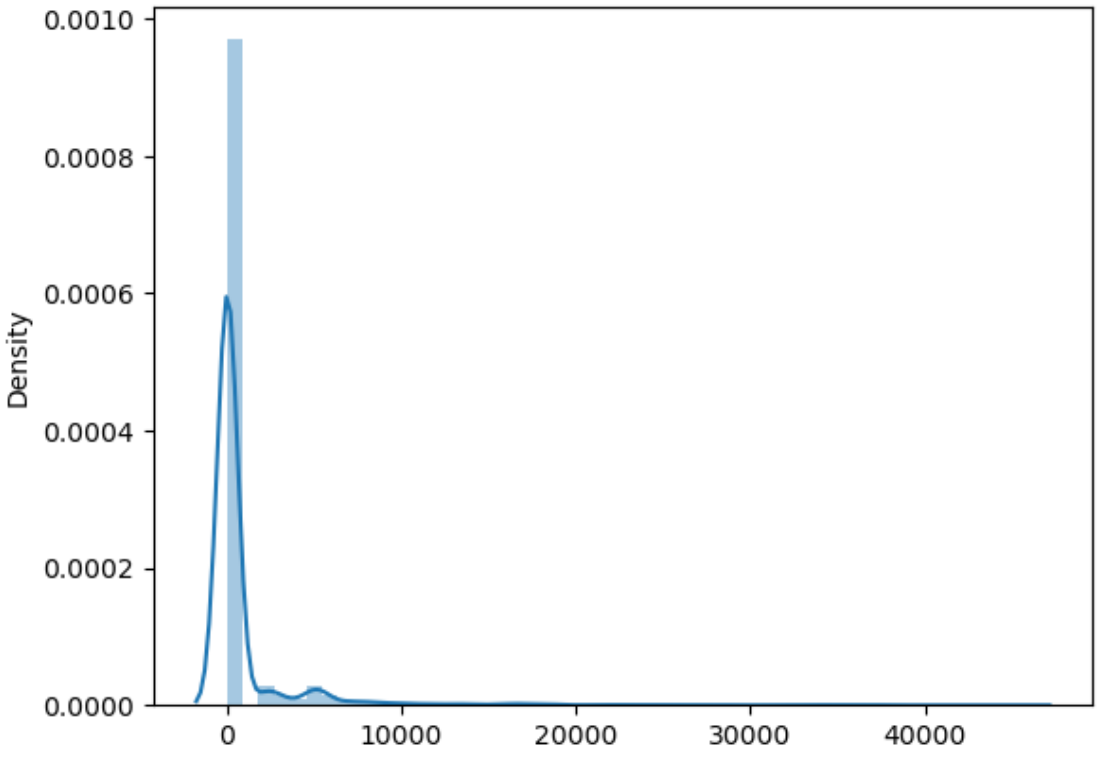
```
horsepower
```


0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
Density
0
10000
20000
30000
40000

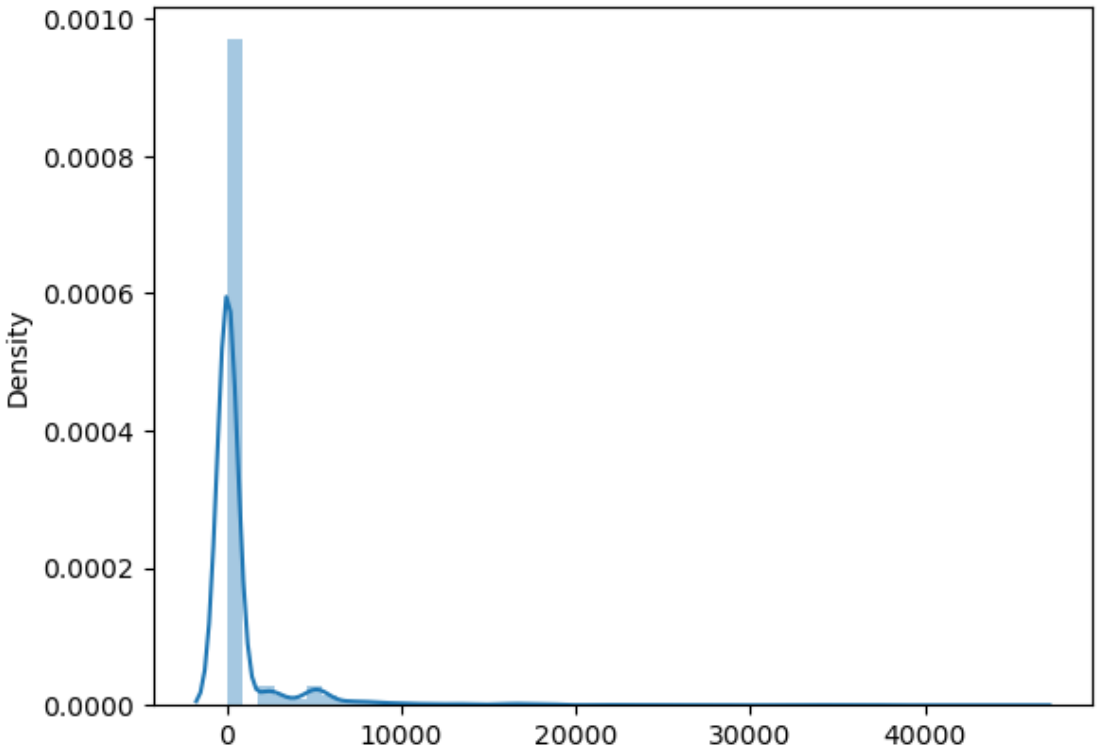
peak-rpm


0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
Density
0
10000
20000
30000
40000

highway-mpg


0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
Density
0
10000
20000
30000
40000

price



[39]:
```
x=df.iloc[:,:-1]
y=df.iloc[:,-1]
```

[40]:
```
x
```

[40]:

	symboling	normalize loss	make	fuel type	aspiriation	no.of.doors	\
0	3	122	0.0	1.0	0.0	2.0	
1	3	122	0.0	1.0	0.0	2.0	
2	1	122	0.0	1.0	0.0	2.0	
3	2	164	1.0	1.0	0.0	1.0	
4	2	164	1.0	1.0	0.0	1.0	
..	…	…	…	…	…	…	
200	-1	95	20.0	1.0	0.0	1.0	
201	-1	95	20.0	1.0	1.0	1.0	
202	-1	95	20.0	1.0	0.0	1.0	
203	-1	95	20.0	0.0	1.0	1.0	
204	-1	95	20.0	1.0	1.0	1.0	

	body-style	drive-wheels	engine-location	wheel-base	…	engine-type	\
0	0.0	2.0	0.0	88.6	…	0.0	
1	0.0	2.0	0.0	88.6	…	0.0	
2	2.0	2.0	0.0	94.5	…	4.0	

```
3          3.0          1.0              0.0        99.8  …          2.0
4          3.0          0.0              0.0        99.4  …          2.0
..         …            …                …          …  …            …
200        3.0          2.0              0.0       109.1  …          2.0
201        3.0          2.0              0.0       109.1  …          2.0
202        3.0          2.0              0.0       109.1  …          4.0
203        3.0          2.0              0.0       109.1  …          2.0
204        3.0          2.0              0.0       109.1  …          2.0

     no.of.cylinders  engine-size  fuel-system  bore  stroke  \
0                2.0          130          4.0  3.47    2.68
1                2.0          130          4.0  3.47    2.68
2                3.0          152          4.0  2.68    3.47
3                2.0          109          4.0  3.19    3.40
4                1.0          136          4.0  3.19    3.40
..               …            …            …    …       …
200              2.0          141          4.0  3.78    3.15
201              2.0          141          4.0  3.78    3.15
202              3.0          173          4.0  3.58    2.87
203              3.0          145          2.0  3.01    3.40
204              2.0          141          4.0  3.78    3.15

     compression-ratio  horsepower  peak-rpm  highway-mpg
0                  9.0         111      5000           27
1                  9.0         111      5000           27
2                  9.0         154      5000           26
3                 10.0         102      5500           30
4                  8.0         115      5500           22
..                 …           …        …              …
200                9.5         114      5400           28
201                8.7         160      5300           25
202                8.8         134      5500           23
203               23.0         106      4800           27
204                9.5         114      5400           25

[195 rows x 24 columns]
```

[41]:
```
y
```

[41]:
```
0      13495
1      16500
2      16500
3      13950
4      17450
       …
200    16845
201    19045
```

```
202    21485
203    22470
204    22625
Name: price, Length: 195, dtype: int64
```

[42]:
```
from sklearn.model_selection import train_test_split
```

[43]:
```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.3,random_state=1)
```

[44]:
```
from sklearn.linear_model import LinearRegression
```

[45]:
```
lr=LinearRegression()
```

[46]:
```
lr.fit(xtrain,ytrain)
```

[46]:
```
LinearRegression()
```

[47]:
```
ypred=lr.predict(xtest)
```

[58]:
```
ypred
```

[58]:
```
array([ 6698.99228135, 28494.65087125,  8576.10253855,  6733.38247687,
       10146.2958262 , 12784.19160876, 18535.69897237, 15118.79053745,
        8429.90108644, 14118.7225434 , 15633.43681051, 19692.11281882,
       13676.59023069, 12700.07179823, 19580.63244985,  6838.3147652 ,
       13819.35396337, 12315.34796741,  6964.87020145, 21792.53044686,
        9764.34777218, 19884.64194795,  6045.7660725 ,  6296.394601  ,
        7997.42191802, 19042.74115491, 13110.20355066,  6067.66512258,
       19416.11909065,  9585.18844337,  6136.09868382, 16019.89385047,
       41502.18474808, 15517.02678402, 10163.95693248,  7422.78657077,
       32562.24163716, 14828.6280675 , 16281.11468108, 11500.38229169,
       27372.04331068, 35124.71244045,  8570.57475996,  5843.1270034 ,
       11722.74237975, 20889.38106432, 13297.56720442,  6056.70251813,
       43413.56796661, 15139.14511896,  7558.80889856,  8081.67202498,
        4769.42145644, 16277.80293709, 16786.3178163 , 16236.89511772,
       16910.4746216 , 17629.14168999,  9599.4545431 ])
```

[48]:
```
from sklearn.metrics import r2_score
```

[49]:
```
r2=r2_score(ytest,ypred)
```

[50]:
```
print(f'accuracy:{r2}')
```

```
accuracy:0.8720413956618704
```

[51]:
```
train=lr.score(xtrain,ytrain)
test=lr.score(xtest,ytest)
print(f'training acc:{train}\ntesting acc:{test}')
```

```
training acc:0.9080102662669532
testing acc:0.8720413956618704
```

[52]:
```
from sklearn.linear_model import Ridge,Lasso
```

[53]:
```
for i in range(50,150):
    l1=Lasso(alpha=i)
    l1.fit(xtrain,ytrain)
    train=l1.score(xtrain,ytrain)
    test=l1.score(xtest,ytest)
    print(f'{i}\ntraining acc:{train}\ntesting acc:{test}')
```

```
50
training acc:0.8987790936321467
testing acc:0.8778577001557755
51
training acc:0.898557110224422
testing acc:0.8777202147690585
52
training acc:0.8983307376368072
testing acc:0.8775780485478457
53
training acc:0.8980999397649365
testing acc:0.8774324994436284
54
training acc:0.8978647162402443
testing acc:0.8772835689805384
55
training acc:0.897625033564214
testing acc:0.8771305106206092
56
training acc:0.8973808603045463
testing acc:0.8769702986244894
57
training acc:0.8971327310928292
testing acc:0.8767975479223011
58
training acc:0.8968835618673036
testing acc:0.8766597652785257
59
training acc:0.8966323134017502
testing acc:0.8765343372240502
60
training acc:0.8963765556882045
testing acc:0.8763980520043999
61
training acc:0.8961180875619552
testing acc:0.8762775812422859
```

```
62
training acc:0.8958717566058294
testing acc:0.8761991201687037
63
training acc:0.8956223692142481
testing acc:0.8760976928315385
64
training acc:0.8953690604931904
testing acc:0.8759934805316031
65
training acc:0.8951116903414951
testing acc:0.875885954503147
66
training acc:0.894850264558507
testing acc:0.8757750954327215
67
training acc:0.8945849180206136
testing acc:0.8756613908101869
68
training acc:0.8943155811777709
testing acc:0.8755445969825992
69
training acc:0.8940422561299346
testing acc:0.8754246945743662
70
training acc:0.8937648905612573
testing acc:0.8753014191542043
71
training acc:0.8934835902474028
testing acc:0.8751753117452096
72
training acc:0.8931982578489607
testing acc:0.8750458113721323
73
training acc:0.8929089815786517
testing acc:0.8749134857463867
74
training acc:0.8926156806488432
testing acc:0.8747777406420675
75
training acc:0.892318392695626
testing acc:0.8746388749561881
76
training acc:0.8920171479341998
testing acc:0.8744971921257451
77
training acc:0.8917118979200771
testing acc:0.8743520598005837
```

```
78
training acc:0.891402667452057
testing acc:0.8742037935991889
79
training acc:0.8910894538338299
testing acc:0.8740523908667099
80
training acc:0.8907722724490748
testing acc:0.8738981910569805
81
training acc:0.8904510808118251
testing acc:0.8737405458019998
82
training acc:0.8901255115005082
testing acc:0.873580785776405
83
training acc:0.8898027140302752
testing acc:0.8734269646255475
84
training acc:0.8894795147114092
testing acc:0.873274111881688
85
training acc:0.889152405359156
testing acc:0.87311736391601
86
training acc:0.8888214443826826
testing acc:0.8729580662354557
87
training acc:0.8884866122259811
testing acc:0.872795774161609
88
training acc:0.8881479089361318
testing acc:0.8726304727490404
89
training acc:0.8878053120679471
testing acc:0.8724616605118524
90
training acc:0.8874588667346873
testing acc:0.8722903201839566
91
training acc:0.8871085500694035
testing acc:0.8721159630553728
92
training acc:0.886754339060074
testing acc:0.8719380893611717
93
training acc:0.8863962789268096
testing acc:0.8717576797254354
```

```
94
training acc:0.8860343243201768
testing acc:0.8715737585183951
95
training acc:0.8856685208719778
testing acc:0.8713872920179119
96
training acc:0.8852988449544412
testing acc:0.8711978042603459
97
training acc:0.8849252782797233
testing acc:0.8710048034759224
98
training acc:0.884547858511548
testing acc:0.8708092539163037
99
training acc:0.8841665496284781
testing acc:0.8706101981832395
100
training acc:0.8837813685246889
testing acc:0.8704081421672637
101
training acc:0.883392334363818
testing acc:0.870203469904733
102
training acc:0.8829994113860238
testing acc:0.8699953497374595
103
training acc:0.8826026159048508
testing acc:0.8697842428401557
104
training acc:0.8822019675273163
testing acc:0.8695704427912875
105
training acc:0.881797430224399
testing acc:0.8693532827000909
106
training acc:0.8813890207234331
testing acc:0.8691331160631954
107
training acc:0.8809858198341969
testing acc:0.8689092859391159
108
training acc:0.8805798453943117
testing acc:0.8686825793038196
109
training acc:0.8801700544629933
testing acc:0.8684525935786163
```

```
110
training acc:0.8797565200862001
testing acc:0.8682196136094994
111
training acc:0.8793379205502796
testing acc:0.8679764037070097
112
training acc:0.8789113852518534
testing acc:0.867698288191229
113
training acc:0.8784814341246673
testing acc:0.8674156396484908
114
training acc:0.8780477785593802
testing acc:0.8671302676841788
115
training acc:0.8776103378657916
testing acc:0.8668419811955037
116
training acc:0.8771690999460224
testing acc:0.8665508419253491
117
training acc:0.8767240436793106
testing acc:0.8662565590269977
118
training acc:0.8762751680015037
testing acc:0.8659591556212478
119
training acc:0.8758224724272853
testing acc:0.8656586287547197
120
training acc:0.875365957502421
testing acc:0.8653549890035336
121
training acc:0.8749056233088367
testing acc:0.8650482458059642
122
training acc:0.87444146990298
testing acc:0.8647383967535078
123
training acc:0.8739735015885223
testing acc:0.8644254822351417
124
training acc:0.8735017136260151
testing acc:0.8641094498330872
125
training acc:0.8730261053271066
testing acc:0.8637902931454002
```

```
126
training acc:0.8725466756720397
testing acc:0.8634679964706843
127
training acc:0.8720634251115341
testing acc:0.863142566448589
128
training acc:0.8715763540886172
testing acc:0.862814009384133
129
training acc:0.871085462649268
testing acc:0.8624823250339148
130
training acc:0.8705939986917607
testing acc:0.8621489473135387
131
training acc:0.8704911638698728
testing acc:0.8619718884676962
132
training acc:0.8703875150483089
testing acc:0.8617941027648728
133
training acc:0.870283077359817
testing acc:0.8616152890818493
134
training acc:0.8701778508520689
testing acc:0.8614354455895822
135
training acc:0.8700718355792915
testing acc:0.8612545714209077
136
training acc:0.8699650520041686
testing acc:0.8610724161581467
137
training acc:0.8698574590125263
testing acc:0.8608894749013627
138
training acc:0.8697490774607693
testing acc:0.8607055023202856
139
training acc:0.8696399075681887
testing acc:0.8605204968013758
140
training acc:0.8695299492125976
testing acc:0.8603344610617513
141
training acc:0.8694192025433806
testing acc:0.8601473931789643
```

```
142
training acc:0.8693076675418653
testing acc:0.8599592943476293
143
training acc:0.8691953443195977
testing acc:0.8597701628829879
144
training acc:0.869082232904364
testing acc:0.8595799997573565
145
training acc:0.8689683333521415
testing acc:0.8593888051005959
146
training acc:0.8688536455978784
testing acc:0.8591965801721144
147
training acc:0.8687381697774824
testing acc:0.8590033239099069
148
training acc:0.8686219058546352
testing acc:0.8588090393797081
149
training acc:0.8685048537255315
testing acc:0.8586137291966242
```

[54]:
```
l1=Lasso(alpha=136)
l1.fit(xtrain,ytrain)
```

[54]:
```
Lasso(alpha=136)
```

[55]:
```
train=l1.score(xtrain,ytrain)
test=l1.score(xtest,ytest)
print(f'training acc:{train}\ntesting acc:{test}')
```

```
training acc:0.8699650520041686
testing acc:0.8610724161581467
```

[ ]:
```

```