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

In [2]:
```
df=pd.read_csv('cars.csv')
```

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

Out[3]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepow
0	3	?	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	1
1	3	?	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	1
2	1	?	alfa-romero	gas	hatchback	rwd	front	65.5	52.4	ohcv	152	1
3	2	164	audi	gas	sedan	fwd	front	66.2	54.3	ohc	109	1
4	2	164	audi	gas	sedan	4wd	front	66.4	54.3	ohc	136	1

In [4]:
```
df.info() #checking that any columns dtype is correct or different
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   symboling          205 non-null    int64  
 1   normalized-losses  205 non-null    object 
 2   make               205 non-null    object 
 3   fuel-type          205 non-null    object 
 4   body-style         205 non-null    object 
 5   drive-wheels       205 non-null    object 
 6   engine-location    205 non-null    object 
 7   width              205 non-null    float64
 8   height             205 non-null    float64
 9   engine-type        205 non-null    object 
 10  engine-size        205 non-null    int64  
 11  horsepower         205 non-null    object 
 12  city-mpg           205 non-null    int64  
 13  highway-mpg        205 non-null    int64  
 14  price              205 non-null    int64  
dtypes: float64(2), int64(5), object(8)
memory usage: 24.1+ KB
```

to know what are the values in a column

In [5]:
```
df['normalized-losses'].value_counts() #to see 'normalized-losses' column values because i
```

Out[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: normalized-losses, dtype: int64
```

to replace ? into nan

In [6]:
```
df['normalized-losses'].replace('?',np.nan,inplace=True) #replace '?' with np.nan,give inp
```

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 15 columns):
```

```
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          205 non-null    int64
 1   normalized-losses  164 non-null    object
 2   make               205 non-null    object
 3   fuel-type          205 non-null    object
 4   body-style         205 non-null    object
 5   drive-wheels       205 non-null    object
 6   engine-location    205 non-null    object
 7   width              205 non-null    float64
 8   height             205 non-null    float64
 9   engine-type        205 non-null    object
 10  engine-size        205 non-null    int64
 11  horsepower         205 non-null    object
 12  city-mpg           205 non-null    int64
 13  highway-mpg        205 non-null    int64
 14  price              205 non-null    int64
dtypes: float64(2), int64(5), object(8)
memory usage: 24.1+ KB
```

to change datatype (object into int or float)

In [8]:
```
df['normalized-losses']=df['normalized-losses'].astype('float64') #changing dtype from 'ob
```

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          205 non-null    int64
 1   normalized-losses  164 non-null    float64
 2   make               205 non-null    object
 3   fuel-type          205 non-null    object
 4   body-style         205 non-null    object
 5   drive-wheels       205 non-null    object
 6   engine-location    205 non-null    object
 7   width              205 non-null    float64
 8   height             205 non-null    float64
 9   engine-type        205 non-null    object
 10  engine-size        205 non-null    int64
 11  horsepower         205 non-null    object
 12  city-mpg           205 non-null    int64
 13  highway-mpg        205 non-null    int64
 14  price              205 non-null    int64
dtypes: float64(3), int64(5), object(7)
memory usage: 24.1+ KB
```

dropna/fillna

In [10]:
```
nmean=df['normalized-losses'].mean()
```

In [11]:
```
nmean
```

Out[11]:
```
122.0
```

In [12]:
```
df['normalized-losses'].fillna(nmean)
```

Out[12]:
```
0      122.0
1      122.0
2      122.0
3      164.0
4      164.0
       ...
200     95.0
201     95.0
202     95.0
203     95.0
204     95.0
Name: normalized-losses, Length: 205, dtype: float64
```

In [13]:
```
df['normalized-losses'].dropna()
```

Out[13]:
```
3      164.0
4      164.0
6      158.0
8      158.0
10     192.0
       ...
200     95.0
201     95.0
202     95.0
203     95.0
204     95.0
Name: normalized-losses, Length: 164, dtype: float64
```

filling using simple imputer

In [14]:
```
df['normalized-losses']
```

Out[14]:
```
0        NaN
1        NaN
2        NaN
3      164.0
4      164.0
       ...
200     95.0
201     95.0
202     95.0
203     95.0
204     95.0
Name: normalized-losses, Length: 205, dtype: float64
```

In [15]:
```
from sklearn.impute import SimpleImputer
```

In [16]:
```
si=SimpleImputer(missing_values=np.nan,strategy='mean')
```

In [17]:
```
df[['normalized-losses']]=si.fit_transform(df[['normalized-losses']]) #give mean value to
```

In [18]:
```
df[['normalized-losses']].value_counts()
```

Out[18]:
```
normalized-losses
122.0               45
```

```
161.0    11
91.0      8
150.0     7
128.0     6
104.0     6
134.0     6
95.0      5
94.0      5
74.0      5
65.0      5
103.0     5
85.0      5
168.0     5
102.0     5
148.0     4
106.0     4
118.0     4
93.0      4
101.0     3
154.0     3
115.0     3
83.0      3
125.0     3
137.0     3
87.0      2
188.0     2
158.0     2
153.0     2
81.0      2
145.0     2
192.0     2
89.0      2
129.0     2
194.0     2
197.0     2
119.0     2
113.0     2
110.0     2
108.0     2
164.0     2
186.0     1
231.0     1
142.0     1
77.0      1
78.0      1
98.0      1
90.0      1
121.0     1
107.0     1
256.0     1
dtype: int64
```

In [19]:

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

Out[19]:

```
horsepower
68            19
70            11
69            10
116            9
110            8
95             7
62             6
114            6
101            6
160            6
```

```
88              6
145             5
84              5
82              5
76              5
97              5
102             5
123             4
86              4
111             4
92              4
85              3
182             3
207             3
73              3
152             3
90              3
121             3
52              2
56              2
94              2
100             2
?               2
156             2
112             2
184             2
176             2
162             2
161             2
155             2
154             1
106             1
115             1
120             1
134             1
135             1
140             1
142             1
143             1
288             1
78              1
48              1
72              1
175             1
64              1
60              1
58              1
55              1
262             1
200             1
dtype: int64
```

In [20]:
```
df['horsepower'].replace('?',np.nan,inplace=True)
```

In [21]:
```
df['horsepower'].value_counts()
```

Out[21]:
```
68     19
70     11
69     10
116     9
110     8
95      7
114     6
```

```
160     6
101     6
62      6
88      6
145     5
76      5
97      5
82      5
84      5
102     5
92      4
111     4
123     4
86      4
207     3
182     3
90      3
121     3
152     3
85      3
73      3
161     2
94      2
56      2
112     2
184     2
155     2
156     2
52      2
100     2
162     2
176     2
140     1
115     1
134     1
78      1
48      1
288     1
143     1
142     1
200     1
58      1
55      1
60      1
175     1
154     1
72      1
120     1
64      1
135     1
262     1
106     1
Name: horsepower, dtype: int64
```

In [22]:
```
df['horsepower']=df['horsepower'].astype('float64') #don't have inplace so assign variable
```

In [23]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
```

```
 0   symboling          205 non-null    int64
 1   normalized-losses  205 non-null    float64
 2   make               205 non-null    object
 3   fuel-type          205 non-null    object
 4   body-style         205 non-null    object
 5   drive-wheels       205 non-null    object
 6   engine-location    205 non-null    object
 7   width              205 non-null    float64
 8   height             205 non-null    float64
 9   engine-type        205 non-null    object
 10  engine-size        205 non-null    int64
 11  horsepower         203 non-null    float64
 12  city-mpg           205 non-null    int64
 13  highway-mpg        205 non-null    int64
 14  price              205 non-null    int64
dtypes: float64(4), int64(5), object(6)
memory usage: 24.1+ KB
```

In [24]:
```
df['horsepower'].value_counts()
```

Out[24]:
```
68.0     19
70.0     11
69.0     10
116.0     9
110.0     8
95.0      7
114.0     6
160.0     6
101.0     6
62.0      6
88.0      6
145.0     5
76.0      5
97.0      5
82.0      5
84.0      5
102.0     5
92.0      4
111.0     4
123.0     4
86.0      4
207.0     3
182.0     3
90.0      3
121.0     3
152.0     3
85.0      3
73.0      3
161.0     2
94.0      2
56.0      2
112.0     2
184.0     2
155.0     2
156.0     2
52.0      2
100.0     2
162.0     2
176.0     2
140.0     1
115.0     1
134.0     1
78.0      1
48.0      1
288.0     1
```

```
143.0     1
142.0     1
200.0     1
58.0      1
55.0      1
60.0      1
175.0     1
154.0     1
72.0      1
120.0     1
64.0      1
135.0     1
262.0     1
106.0     1
Name: horsepower, dtype: int64
```

In [25]:
```
df[['horsepower']]=si.fit_transform(df[['horsepower']])
```

In [26]:
```
df['horsepower'].value_counts()
```

Out[26]:
```
68.000000     19
70.000000     11
69.000000     10
116.000000     9
110.000000     8
95.000000      7
88.000000      6
62.000000      6
101.000000     6
160.000000     6
114.000000     6
84.000000      5
97.000000      5
102.000000     5
145.000000     5
82.000000      5
76.000000      5
111.000000     4
92.000000      4
123.000000     4
86.000000      4
90.000000      3
73.000000      3
85.000000      3
207.000000     3
182.000000     3
121.000000     3
152.000000     3
112.000000     2
56.000000      2
161.000000     2
156.000000     2
94.000000      2
52.000000      2
104.256158     2
162.000000     2
155.000000     2
184.000000     2
100.000000     2
176.000000     2
55.000000      1
262.000000     1
134.000000     1
```

```
115.000000     1
140.000000     1
48.000000      1
58.000000      1
60.000000      1
78.000000      1
135.000000     1
200.000000     1
64.000000      1
120.000000     1
72.000000      1
154.000000     1
288.000000     1
143.000000     1
142.000000     1
175.000000     1
106.000000     1
Name: horsepower, dtype: int64
```

In [ ]:

handling missing value---practise

In [27]:
```
cf=pd.read_csv('cars.csv')
```

In [28]:
```
cf
```

Out[28]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsep
0	3	?	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	
1	3	?	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	
2	1	?	alfa-romero	gas	hatchback	rwd	front	65.5	52.4	ohcv	152	
3	2	164	audi	gas	sedan	fwd	front	66.2	54.3	ohc	109	
4	2	164	audi	gas	sedan	4wd	front	66.4	54.3	ohc	136	
...	...	...	...	...	...	...	...	...	...	...	...	...
200	-1	95	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	
201	-1	95	volvo	gas	sedan	rwd	front	68.8	55.5	ohc	141	
202	-1	95	volvo	gas	sedan	rwd	front	68.9	55.5	ohcv	173	
203	-1	95	volvo	diesel	sedan	rwd	front	68.9	55.5	ohc	145	
204	-1	95	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	

205 rows × 15 columns

In [29]:
```
cf.head(20)
```

Out[29]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsep
0	3	?	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	
1	3	?	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	
2	1	?	alfa-romero	gas	hatchback	rwd	front	65.5	52.4	ohcv	152	
3	2	164	audi	gas	sedan	fwd	front	66.2	54.3	ohc	109	
4	2	164	audi	gas	sedan	4wd	front	66.4	54.3	ohc	136	
5	2	?	audi	gas	sedan	fwd	front	66.3	53.1	ohc	136	
6	1	158	audi	gas	sedan	fwd	front	71.4	55.7	ohc	136	
7	1	?	audi	gas	wagon	fwd	front	71.4	55.7	ohc	136	
8	1	158	audi	gas	sedan	fwd	front	71.4	55.9	ohc	131	
9	0	?	audi	gas	hatchback	4wd	front	67.9	52.0	ohc	131	
10	2	192	bmw	gas	sedan	rwd	front	64.8	54.3	ohc	108	
11	0	192	bmw	gas	sedan	rwd	front	64.8	54.3	ohc	108	
12	0	188	bmw	gas	sedan	rwd	front	64.8	54.3	ohc	164	
13	0	188	bmw	gas	sedan	rwd	front	64.8	54.3	ohc	164	
14	1	?	bmw	gas	sedan	rwd	front	66.9	55.7	ohc	164	
15	0	?	bmw	gas	sedan	rwd	front	66.9	55.7	ohc	209	
16	0	?	bmw	gas	sedan	rwd	front	67.9	53.7	ohc	209	
17	0	?	bmw	gas	sedan	rwd	front	70.9	56.3	ohc	209	
18	2	121	chevrolet	gas	hatchback	fwd	front	60.3	53.2	l	61	
19	1	98	chevrolet	gas	hatchback	fwd	front	63.6	52.0	ohc	90	

In [30]:
```
cf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   symboling          205 non-null    int64  
 1   normalized-losses  205 non-null    object 
 2   make               205 non-null    object 
 3   fuel-type          205 non-null    object 
 4   body-style         205 non-null    object 
 5   drive-wheels       205 non-null    object 
 6   engine-location    205 non-null    object 
 7   width              205 non-null    float64
 8   height             205 non-null    float64
 9   engine-type        205 non-null    object 
 10  engine-size        205 non-null    int64  
 11  horsepower         205 non-null    object 
 12  city-mpg           205 non-null    int64  
 13  highway-mpg        205 non-null    int64  
 14  price              205 non-null    int64  
```

```
dtypes: float64(2), int64(5), object(8)
memory usage: 24.1+ KB
```

In [31]:
```
cf['normalized-losses'].value_counts()
```

Out[31]:
```
?      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: normalized-losses, dtype: int64
```

In [32]:
```
cf['normalized-losses'].replace('?',np.nan,inplace=True)
```

In [33]:
```
cf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          205 non-null    int64
 1   normalized-losses  164 non-null    object
 2   make               205 non-null    object
 3   fuel-type          205 non-null    object
 4   body-style         205 non-null    object
 5   drive-wheels       205 non-null    object
 6   engine-location    205 non-null    object
 7   width              205 non-null    float64
 8   height             205 non-null    float64
 9   engine-type        205 non-null    object
 10  engine-size        205 non-null    int64
 11  horsepower         205 non-null    object
 12  city-mpg           205 non-null    int64
 13  highway-mpg        205 non-null    int64
 14  price              205 non-null    int64
dtypes: float64(2), int64(5), object(8)
memory usage: 24.1+ KB
```

In [34]:
```
cf['normalized-losses']=cf['normalized-losses'].astype('float64')
```

In [35]:
```
cf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          205 non-null    int64
 1   normalized-losses  164 non-null    float64
 2   make               205 non-null    object
 3   fuel-type          205 non-null    object
 4   body-style         205 non-null    object
 5   drive-wheels       205 non-null    object
 6   engine-location    205 non-null    object
 7   width              205 non-null    float64
 8   height             205 non-null    float64
 9   engine-type        205 non-null    object
 10  engine-size        205 non-null    int64
 11  horsepower         205 non-null    object
 12  city-mpg           205 non-null    int64
 13  highway-mpg        205 non-null    int64
 14  price              205 non-null    int64
dtypes: float64(3), int64(5), object(7)
memory usage: 24.1+ KB
```

In [36]:
```
nmedian=cf['normalized-losses'].median()
```

In [37]:
```
nmedian
```

Out[37]:
```
115.0
```

In [38]:
```
cf['normalized-losses'].fillna(nmedian)
```

Out[38]:
```
0      115.0
1      115.0
```

```
2      115.0
3      164.0
4      164.0
       ...
200     95.0
201     95.0
202     95.0
203     95.0
204     95.0
Name: normalized-losses, Length: 205, dtype: float64
```

In [39]:
```
cf['normalized-losses'].dropna()
```

Out[39]:
```
3      164.0
4      164.0
6      158.0
8      158.0
10     192.0
       ...
200     95.0
201     95.0
202     95.0
203     95.0
204     95.0
Name: normalized-losses, Length: 164, dtype: float64
```

In [40]:
```
from sklearn.impute import SimpleImputer
```

In [41]:
```
si=SimpleImputer(missing_values=np.nan,strategy='median')
cf[['normalized-losses']]=si.fit_transform(cf[['normalized-losses']])
```

In [42]:
```
cf['normalized-losses'].value_counts()
```

Out[42]:
```
115.0    44
161.0    11
91.0      8
150.0     7
134.0     6
128.0     6
104.0     6
85.0      5
94.0      5
65.0      5
102.0     5
74.0      5
168.0     5
103.0     5
95.0      5
106.0     4
93.0      4
118.0     4
148.0     4
122.0     4
83.0      3
125.0     3
154.0     3
137.0     3
101.0     3
188.0     2
119.0     2
89.0      2
```

```
192.0     2
197.0     2
158.0     2
81.0      2
87.0      2
153.0     2
129.0     2
108.0     2
110.0     2
164.0     2
145.0     2
194.0     2
113.0     2
78.0      1
256.0     1
107.0     1
90.0      1
77.0      1
142.0     1
121.0     1
98.0      1
186.0     1
231.0     1
Name: normalized-losses, dtype: int64
```

OUTLIER

In [43]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   symboling          205 non-null    int64  
 1   normalized-losses  205 non-null    float64
 2   make               205 non-null    object 
 3   fuel-type          205 non-null    object 
 4   body-style         205 non-null    object 
 5   drive-wheels       205 non-null    object 
 6   engine-location    205 non-null    object 
 7   width              205 non-null    float64
 8   height             205 non-null    float64
 9   engine-type        205 non-null    object 
 10  engine-size        205 non-null    int64  
 11  horsepower         205 non-null    float64
 12  city-mpg           205 non-null    int64  
 13  highway-mpg        205 non-null    int64  
 14  price              205 non-null    int64  
dtypes: float64(4), int64(5), object(6)
memory usage: 24.1+ KB
```

split the table into input as feature and output as target

In [103…
```
feature=df.iloc[:,:-1]
target=df['price']                          #target=df.iloc[:,-1]
```

In [45]:
```
feature
```

Out[45]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsep
0	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	
1	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	
2	1	122.0	alfa-romero	gas	hatchback	rwd	front	65.5	52.4	ohcv	152	
3	2	164.0	audi	gas	sedan	fwd	front	66.2	54.3	ohc	109	
4	2	164.0	audi	gas	sedan	4wd	front	66.4	54.3	ohc	136	
...	...	...	...	...	...	...	...	...	...	...	...	
200	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	
201	-1	95.0	volvo	gas	sedan	rwd	front	68.8	55.5	ohc	141	
202	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohcv	173	
203	-1	95.0	volvo	diesel	sedan	rwd	front	68.9	55.5	ohc	145	
204	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	

205 rows × 14 columns

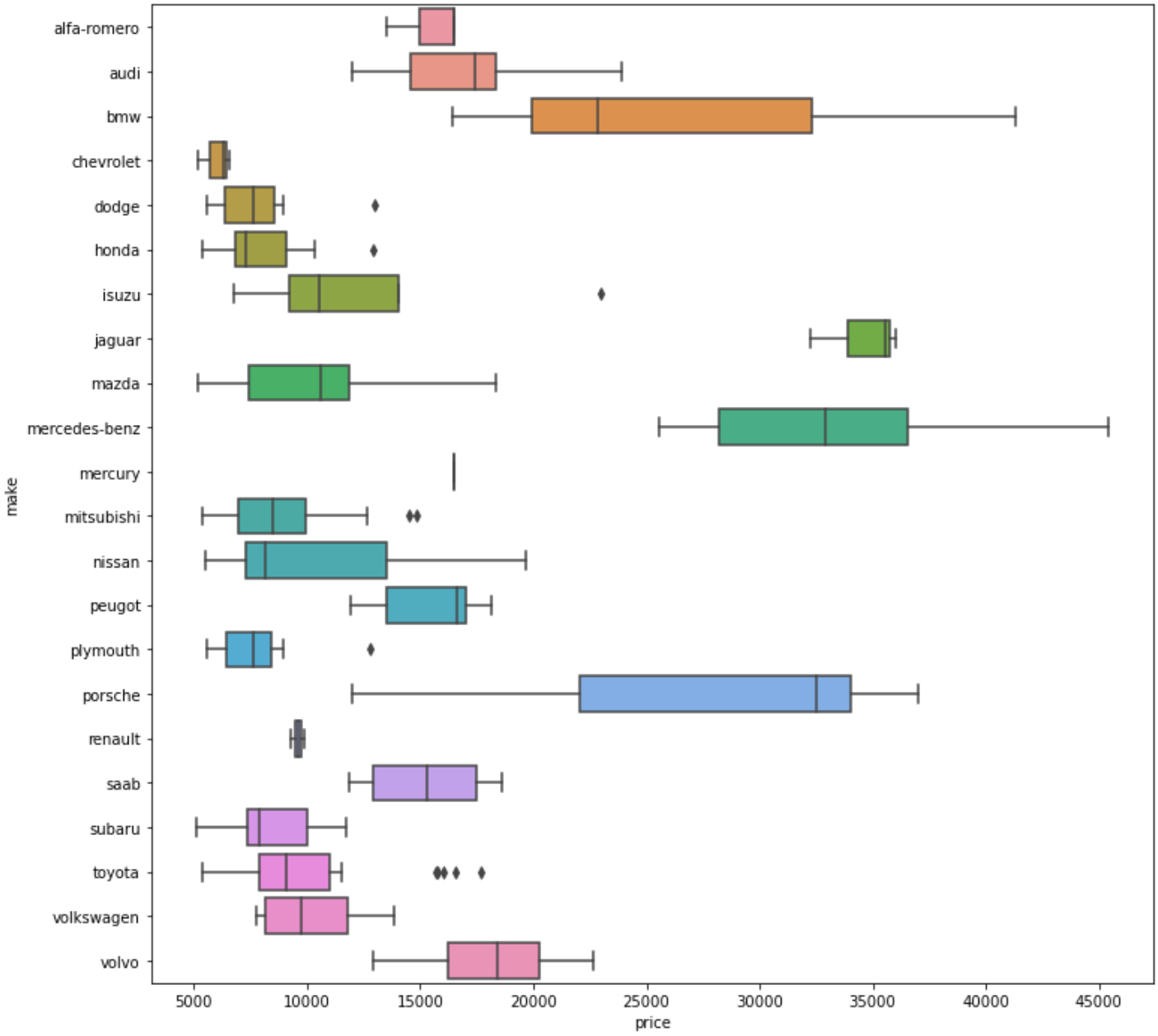
In [46]:
```
target
```

Out[46]:
```
0      13495
1      16500
2      16500
3      13950
4      17450
       ...
200    16845
201    19045
202    21485
203    22470
204    22625
Name: price, Length: 205, dtype: int64
```

finging outlier for each company(make--column)

In [47]:
```
plt.figure(figsize=(12,12))
sns.boxplot(data=feature,x=target,y='make')
```

Out[47]:
```
<AxesSubplot:xlabel='price', ylabel='make'>
```

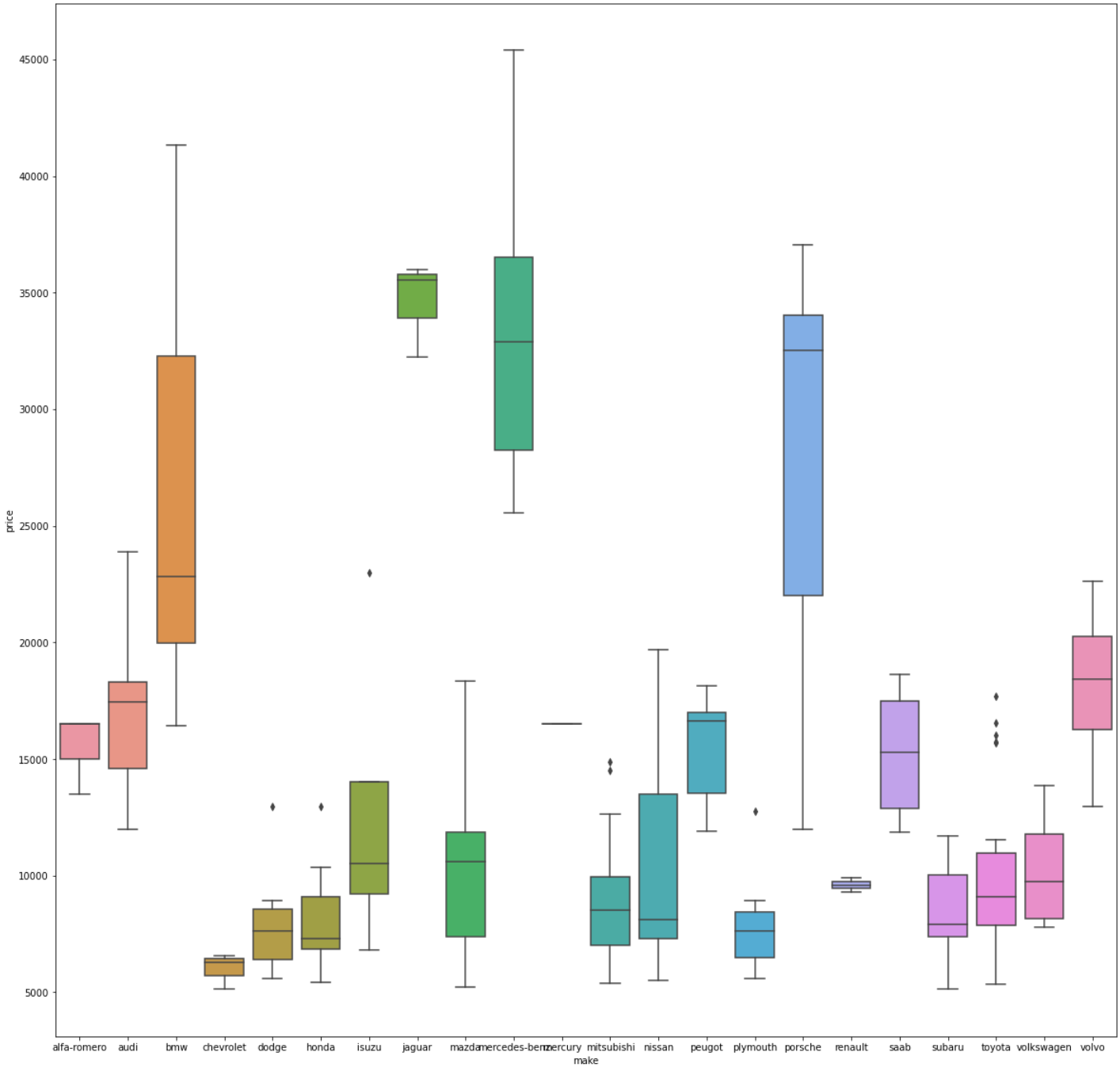


In [48]:

```
plt.figure(figsize=(20,20))
sns.boxplot(data=feature,y=target,x='make')
```

Out[48]:

```
<AxesSubplot:xlabel='make', ylabel='price'>
```



drop a outlier ------ from 'honda' that are greater than 12000

In [49]:
```
df.head()
```

Out[49]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepow
0	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111
1	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111
2	1	122.0	alfa-romero	gas	hatchback	rwd	front	65.5	52.4	ohcv	152	154
3	2	164.0	audi	gas	sedan	fwd	front	66.2	54.3	ohc	109	102

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepow
4	2	164.0	audi	gas	sedan	4wd	front	66.4	54.3	ohc	136	115

In [50]:
```
df[(df['make']=='honda')&(df['price']>12000)]
```

Out[50]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower
41	0	85.0	honda	gas	sedan	fwd	front	65.2	54.1	ohc	110	101.0

In [51]:
```
df.drop(41,axis=0,inplace=True) #drop a 41th row
```

In [52]:
```
df.head(50) #41 index was deleted
```

Out[52]:

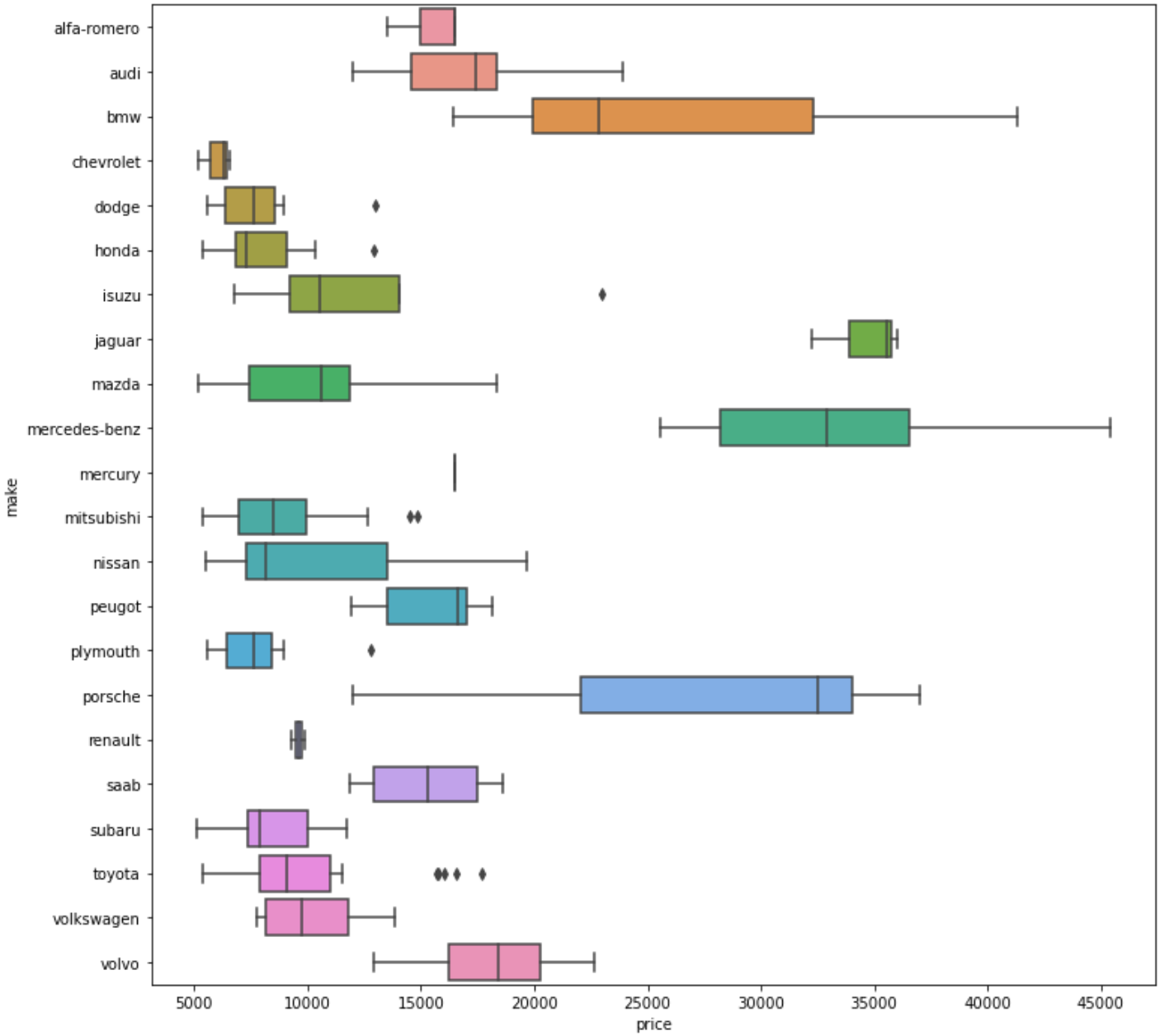
	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsep
0	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	
1	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	
2	1	122.0	alfa-romero	gas	hatchback	rwd	front	65.5	52.4	ohcv	152	
3	2	164.0	audi	gas	sedan	fwd	front	66.2	54.3	ohc	109	
4	2	164.0	audi	gas	sedan	4wd	front	66.4	54.3	ohc	136	
5	2	122.0	audi	gas	sedan	fwd	front	66.3	53.1	ohc	136	
6	1	158.0	audi	gas	sedan	fwd	front	71.4	55.7	ohc	136	
7	1	122.0	audi	gas	wagon	fwd	front	71.4	55.7	ohc	136	
8	1	158.0	audi	gas	sedan	fwd	front	71.4	55.9	ohc	131	
9	0	122.0	audi	gas	hatchback	4wd	front	67.9	52.0	ohc	131	
10	2	192.0	bmw	gas	sedan	rwd	front	64.8	54.3	ohc	108	
11	0	192.0	bmw	gas	sedan	rwd	front	64.8	54.3	ohc	108	
12	0	188.0	bmw	gas	sedan	rwd	front	64.8	54.3	ohc	164	
13	0	188.0	bmw	gas	sedan	rwd	front	64.8	54.3	ohc	164	
14	1	122.0	bmw	gas	sedan	rwd	front	66.9	55.7	ohc	164	
15	0	122.0	bmw	gas	sedan	rwd	front	66.9	55.7	ohc	209	
16	0	122.0	bmw	gas	sedan	rwd	front	67.9	53.7	ohc	209	
17	0	122.0	bmw	gas	sedan	rwd	front	70.9	56.3	ohc	209	
18	2	121.0	chevrolet	gas	hatchback	fwd	front	60.3	53.2	l	61	
19	1	98.0	chevrolet	gas	hatchback	fwd	front	63.6	52.0	ohc	90	
20	0	81.0	chevrolet	gas	sedan	fwd	front	63.6	52.0	ohc	90	
21	1	118.0	dodge	gas	hatchback	fwd	front	63.8	50.8	ohc	90	

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsep
22	1	118.0	dodge	gas	hatchback	fwd	front	63.8	50.8	ohc	90	
23	1	118.0	dodge	gas	hatchback	fwd	front	63.8	50.8	ohc	98	
24	1	148.0	dodge	gas	hatchback	fwd	front	63.8	50.6	ohc	90	
25	1	148.0	dodge	gas	sedan	fwd	front	63.8	50.6	ohc	90	
26	1	148.0	dodge	gas	sedan	fwd	front	63.8	50.6	ohc	90	
27	1	148.0	dodge	gas	sedan	fwd	front	63.8	50.6	ohc	98	
28	-1	110.0	dodge	gas	wagon	fwd	front	64.6	59.8	ohc	122	
29	3	145.0	dodge	gas	hatchback	fwd	front	66.3	50.2	ohc	156	
30	2	137.0	honda	gas	hatchback	fwd	front	63.9	50.8	ohc	92	
31	2	137.0	honda	gas	hatchback	fwd	front	63.9	50.8	ohc	92	
32	1	101.0	honda	gas	hatchback	fwd	front	64.0	52.6	ohc	79	
33	1	101.0	honda	gas	hatchback	fwd	front	64.0	52.6	ohc	92	
34	1	101.0	honda	gas	hatchback	fwd	front	64.0	52.6	ohc	92	
35	0	110.0	honda	gas	sedan	fwd	front	64.0	54.5	ohc	92	
36	0	78.0	honda	gas	wagon	fwd	front	63.9	58.3	ohc	92	
37	0	106.0	honda	gas	hatchback	fwd	front	65.2	53.3	ohc	110	
38	0	106.0	honda	gas	hatchback	fwd	front	65.2	53.3	ohc	110	
39	0	85.0	honda	gas	sedan	fwd	front	65.2	54.1	ohc	110	
40	0	85.0	honda	gas	sedan	fwd	front	62.5	54.1	ohc	110	
42	1	107.0	honda	gas	sedan	fwd	front	66.0	51.0	ohc	110	
43	0	122.0	isuzu	gas	sedan	rwd	front	61.8	53.5	ohc	111	
44	1	122.0	isuzu	gas	sedan	fwd	front	63.6	52.0	ohc	90	
45	0	122.0	isuzu	gas	sedan	fwd	front	63.6	52.0	ohc	90	
46	2	122.0	isuzu	gas	hatchback	rwd	front	65.2	51.4	ohc	119	
47	0	145.0	jaguar	gas	sedan	rwd	front	69.6	52.8	dohc	258	
48	0	122.0	jaguar	gas	sedan	rwd	front	69.6	52.8	dohc	258	
49	0	122.0	jaguar	gas	sedan	rwd	front	70.6	47.8	ohcv	326	
50	1	104.0	mazda	gas	hatchback	fwd	front	64.2	54.1	ohc	91	

In [ ]:

In [53]:
```
plt.figure(figsize=(12,12))
sns.boxplot(data=feature,x=target,y='make')
```

Out[53]:
```
<AxesSubplot:xlabel='price', ylabel='make'>
```

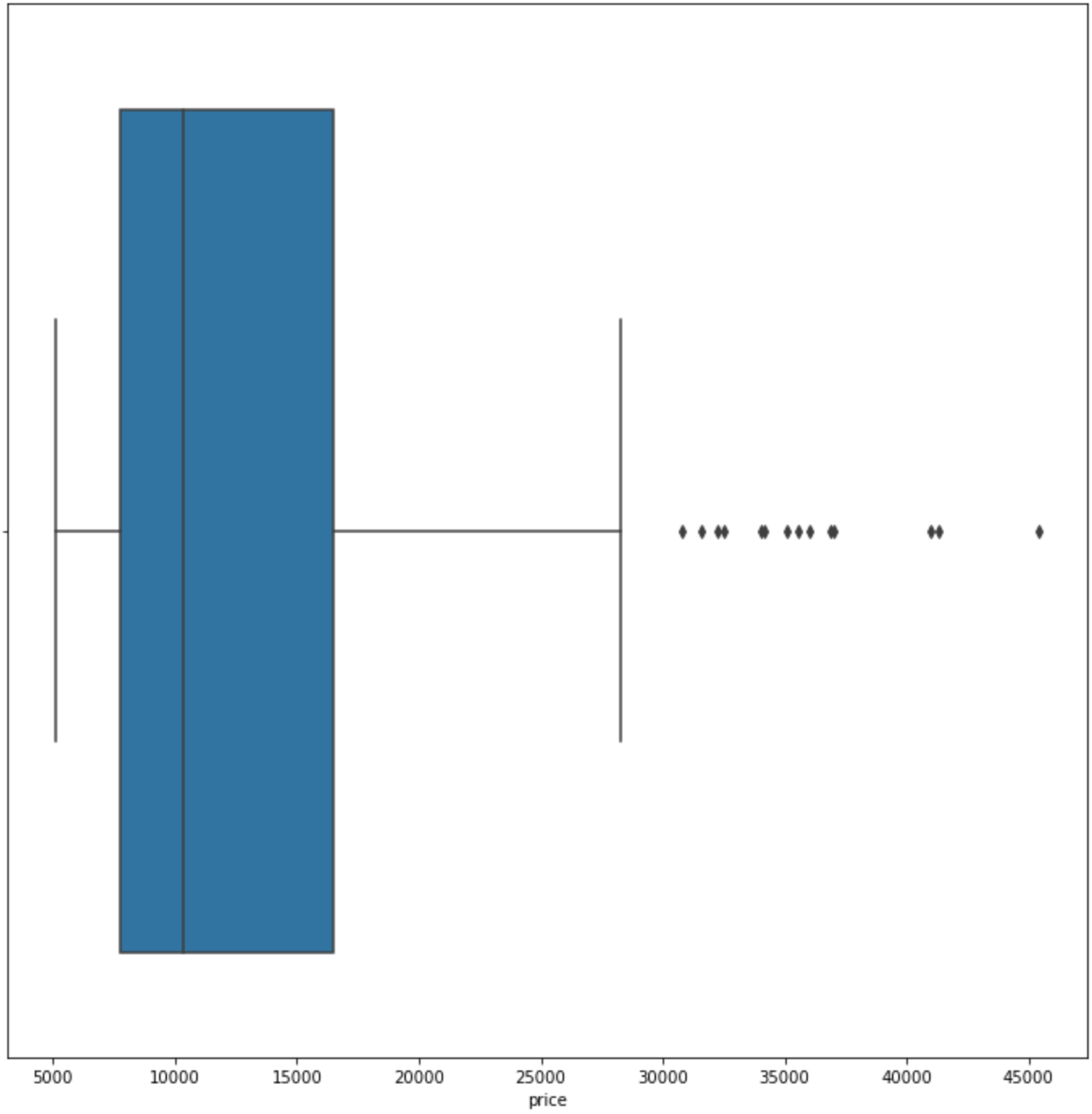


In [54]:

```
plt.figure(figsize=(12,12))
sns.boxplot(data=(feature['make']=='dodge'),x=target)
```

Out[54]:

```
<AxesSubplot:xlabel='price'>
```



to find null value in large dataset and drop

In [55]:
```
ll=pd.read_csv('hp.train.csv')
```

In [56]:
```
ll.head(20)
```

Out[56]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea
0	1	60.0	RL	65.0	8450.0	Pave	NaN	Reg	Lvl	AllPub	...	0
1	2	20.0	RL	80.0	9600.0	Pave	NaN	Reg	Lvl	AllPub	...	0
2	3	60.0	RL	68.0	11250.0	Pave	NaN	IR1	Lvl	AllPub	...	0
3	4	NaN	RL	NaN	NaN	Pave	NaN	IR1	Lvl	AllPub	...	0
4	5	60.0	RL	NaN	NaN	Pave	NaN	IR1	Lvl	AllPub	...	0

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea
5	6	50.0	RL	85.0	14115.0	Pave	NaN	IR1	Lvl	AllPub	...	0
6	7	20.0	RL	75.0	10084.0	Pave	NaN	Reg	Lvl	AllPub	...	0
7	8	60.0	RL	NaN	10382.0	Pave	NaN	IR1	Lvl	AllPub	...	0
8	9	50.0	RM	51.0	6120.0	Pave	NaN	Reg	Lvl	AllPub	...	0
9	10	NaN	RL	50.0	7420.0	Pave	NaN	Reg	Lvl	AllPub	...	0
10	11	20.0	RL	70.0	11200.0	Pave	NaN	Reg	Lvl	AllPub	...	0
11	12	60.0	RL	85.0	11924.0	Pave	NaN	IR1	Lvl	AllPub	...	0
12	13	20.0	RL	NaN	12968.0	Pave	NaN	IR2	Lvl	AllPub	...	0
13	14	20.0	RL	91.0	10652.0	Pave	NaN	IR1	Lvl	AllPub	...	0
14	15	20.0	RL	NaN	10920.0	Pave	NaN	IR1	Lvl	AllPub	...	0
15	16	45.0	RM	51.0	6120.0	Pave	NaN	Reg	Lvl	AllPub	...	0
16	17	20.0	RL	NaN	11241.0	Pave	NaN	IR1	Lvl	AllPub	...	0
17	18	90.0	RL	72.0	10791.0	Pave	NaN	Reg	Lvl	AllPub	...	0
18	19	20.0	RL	66.0	13695.0	Pave	NaN	Reg	Lvl	AllPub	...	0
19	20	20.0	RL	70.0	7560.0	Pave	NaN	Reg	Lvl	AllPub	...	0

20 rows × 81 columns

In [57]:
```
ll.isna().sum() #to find null value is there or not
```

Out[57]:
```
Id                0
MSSubClass        2
MSZoning          0
LotFrontage     261
LotArea           2
                 ...
MoSold            0
YrSold            0
SaleType          0
SaleCondition     0
SalePrice         0
Length: 81, dtype: int64
```

In [58]:
```
ll.dropna(how='all') #to drop a row if all columns are empty
```

Out[58]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	Poo
0	1	60.0	RL	65.0	8450.0	Pave	NaN	Reg	Lvl	AllPub	...	
1	2	20.0	RL	80.0	9600.0	Pave	NaN	Reg	Lvl	AllPub	...	
2	3	60.0	RL	68.0	11250.0	Pave	NaN	IR1	Lvl	AllPub	...	
3	4	NaN	RL	NaN	NaN	Pave	NaN	IR1	Lvl	AllPub	...	
4	5	60.0	RL	NaN	NaN	Pave	NaN	IR1	Lvl	AllPub	...	
...	...	...	...	...	...	...	...	...	...	...	...	...
1455	1456	60.0	RL	62.0	7917.0	Pave	NaN	Reg	Lvl	AllPub	...	

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	Poo
1456	1457	20.0	RL	85.0	13175.0	Pave	NaN	Reg	Lvl	AllPub	...	
1457	1458	70.0	RL	66.0	9042.0	Pave	NaN	Reg	Lvl	AllPub	...	
1458	1459	20.0	RL	68.0	9717.0	Pave	NaN	Reg	Lvl	AllPub	...	
1459	1460	20.0	RL	75.0	9937.0	Pave	NaN	Reg	Lvl	AllPub	...	

1460 rows × 81 columns

In [59]:
```
ll.dropna(how='all',subset=['MSSubClass']) #to drop a row if 'MSSubClass' is empty
```

Out[59]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	Poo
0	1	60.0	RL	65.0	8450.0	Pave	NaN	Reg	Lvl	AllPub	...	
1	2	20.0	RL	80.0	9600.0	Pave	NaN	Reg	Lvl	AllPub	...	
2	3	60.0	RL	68.0	11250.0	Pave	NaN	IR1	Lvl	AllPub	...	
4	5	60.0	RL	NaN	NaN	Pave	NaN	IR1	Lvl	AllPub	...	
5	6	50.0	RL	85.0	14115.0	Pave	NaN	IR1	Lvl	AllPub	...	
...	...	...	...	...	...	...	...	...	...	...	...	...
1455	1456	60.0	RL	62.0	7917.0	Pave	NaN	Reg	Lvl	AllPub	...	
1456	1457	20.0	RL	85.0	13175.0	Pave	NaN	Reg	Lvl	AllPub	...	
1457	1458	70.0	RL	66.0	9042.0	Pave	NaN	Reg	Lvl	AllPub	...	
1458	1459	20.0	RL	68.0	9717.0	Pave	NaN	Reg	Lvl	AllPub	...	
1459	1460	20.0	RL	75.0	9937.0	Pave	NaN	Reg	Lvl	AllPub	...	

1458 rows × 81 columns

to find skew

In [60]:
```
from scipy.stats import skew
```

In [61]:
```
colname=feature.select_dtypes(['int64','float64']).columns #give column name that have int
```

In [62]:
```
colname
```

Out[62]:
```
Index(['symboling', 'normalized-losses', 'width', 'height', 'engine-size',
       'horsepower', 'city-mpg', 'highway-mpg'],
      dtype='object')
```

In [63]:
```
feature[colname]
```

Out[63]:

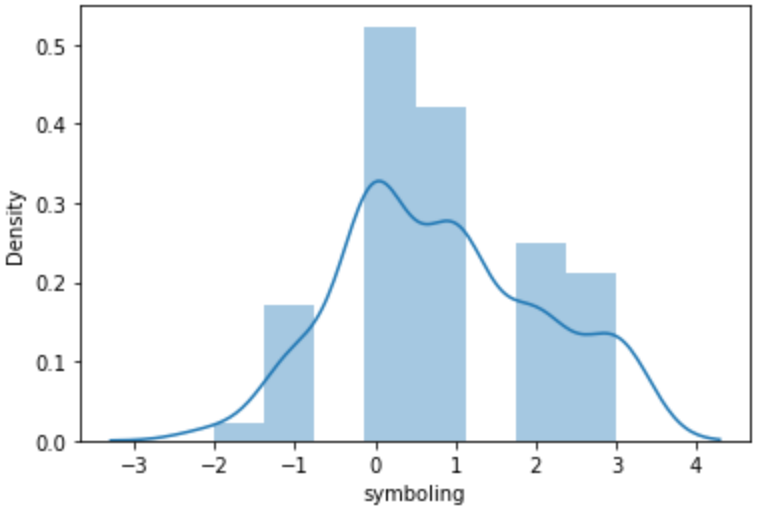
	symboling	normalized-losses	width	height	engine-size	horsepower	city-mpg	highway-mpg
0	3	122.0	64.1	48.8	130	111.0	21	27
1	3	122.0	64.1	48.8	130	111.0	21	27

	symboling	normalized-losses	width	height	engine-size	horsepower	city-mpg	highway-mpg
2	1	122.0	65.5	52.4	152	154.0	19	26
3	2	164.0	66.2	54.3	109	102.0	24	30
4	2	164.0	66.4	54.3	136	115.0	18	22
...	...	...	...	...	...	...	...	...
200	-1	95.0	68.9	55.5	141	114.0	23	28
201	-1	95.0	68.8	55.5	141	160.0	19	25
202	-1	95.0	68.9	55.5	173	134.0	18	23
203	-1	95.0	68.9	55.5	145	106.0	26	27
204	-1	95.0	68.9	55.5	141	114.0	19	25

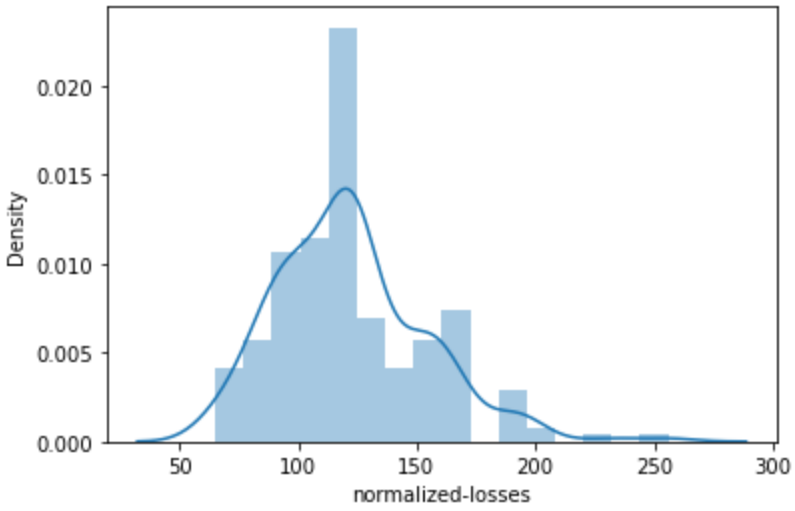
205 rows × 8 columns

In [64]:
```
for i in feature[colname]:
    print(i)
    print(skew(feature[i]))
    plt.figure()
    sns.distplot(feature[i]) #for display distplot to find have skew or not
    plt.show()
```

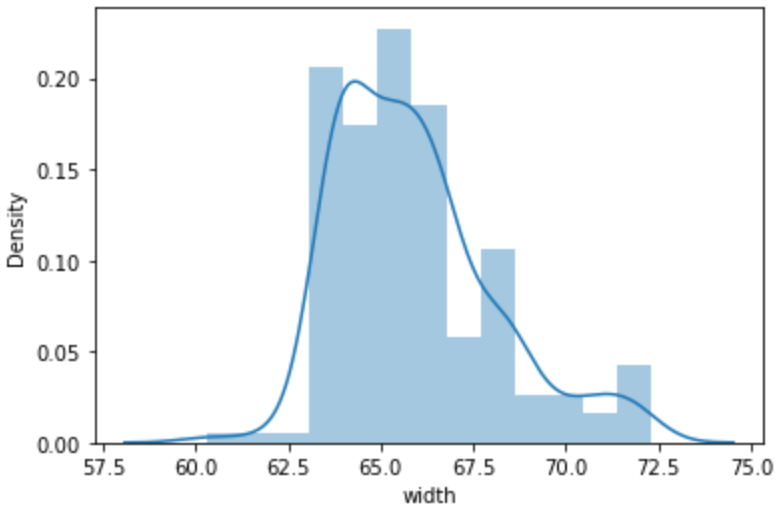
```
symboling
0.20952469094997359
```



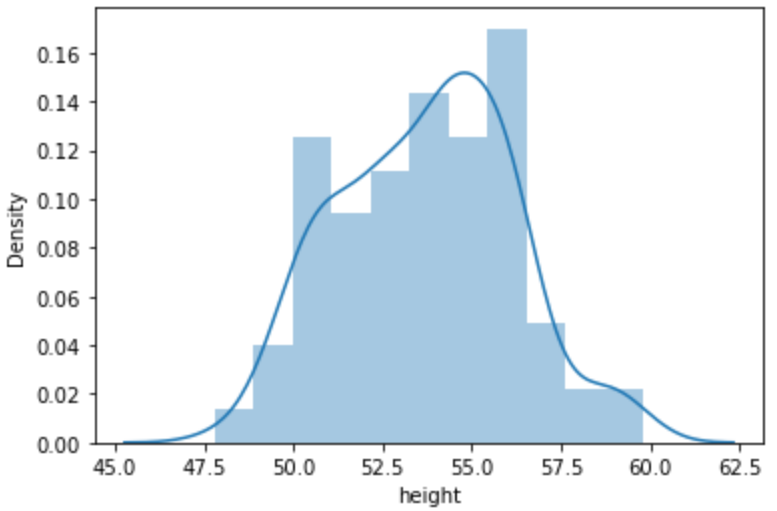

```
normalized-losses
0.8485348696008058
```



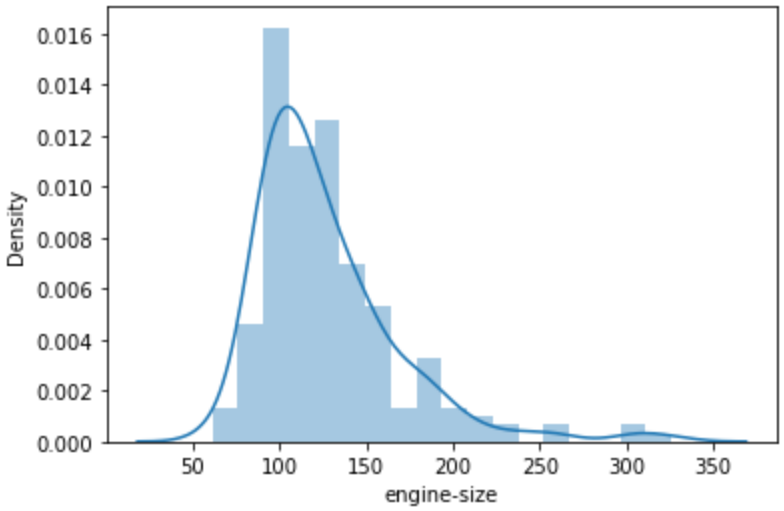

```
width
0.8973753485201392
```



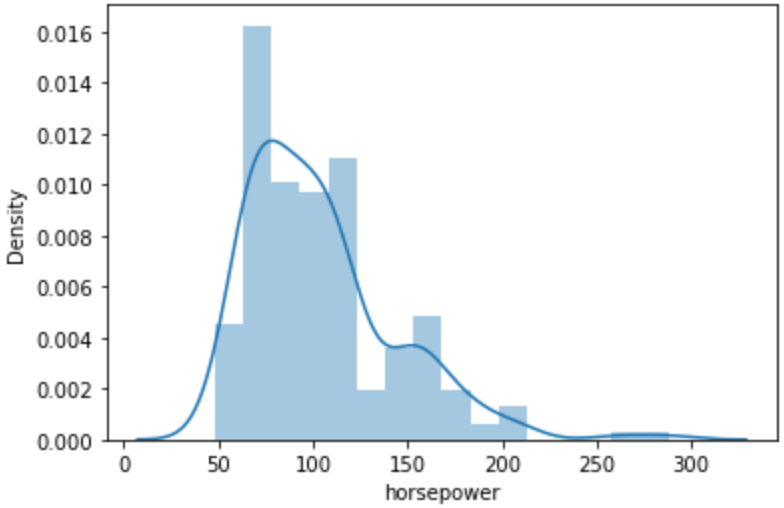

```
height
0.06265991683394276
```



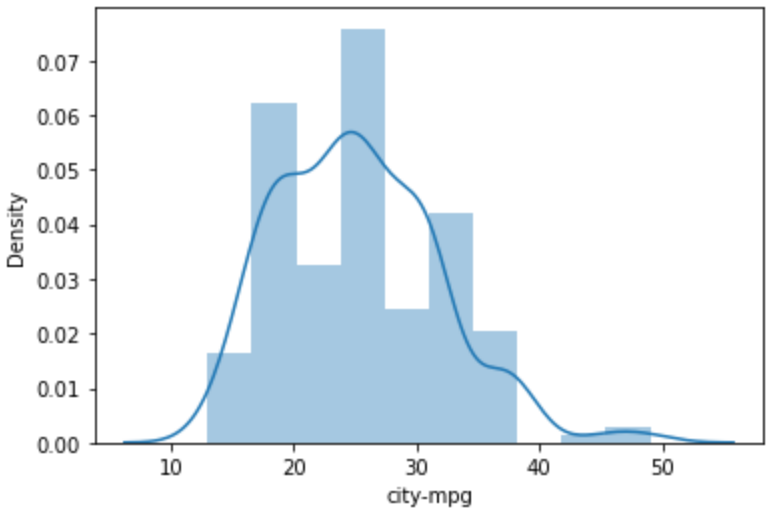

```
engine-size
1.9333748457840114
```



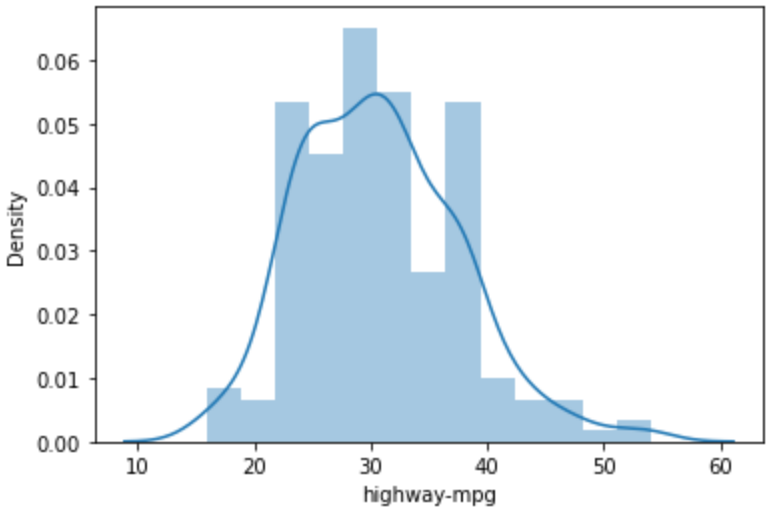

```
horsepower
1.3875147343096037
```




```
city-mpg
0.6588377533622138
```




```
highway-mpg
0.5360379305163596
```

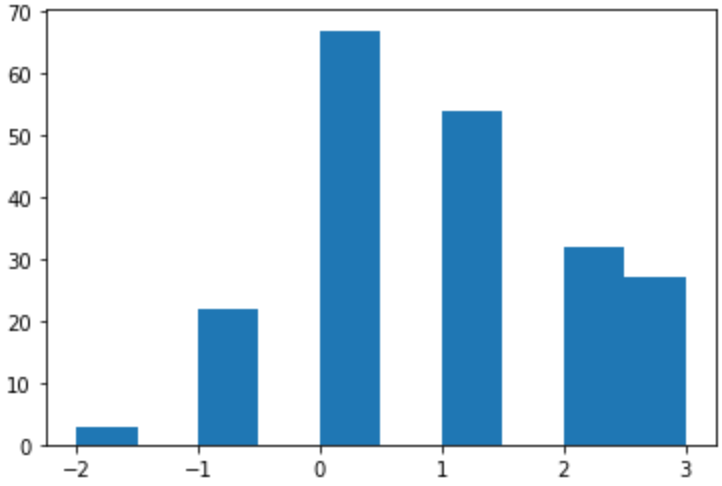


In [65]:

```
for i in feature[colname]:
    print(i)
    print(skew(feature[i]))
    plt.figure()
    plt.hist(feature[i],kde=True) #for display distplot to find have skew or not
    plt.show()
```

```
symboling
0.20952469094997359
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_4608/1405357852.py in <module>
      3     print(skew(feature[i]))
      4     plt.figure()
----> 5     plt.hist(feature[i],kde=True) #for display distplot to find have skew or not
      6     plt.show()

~\Anaconda3\lib\site-packages\matplotlib\pyplot.py in hist(x, bins, range, density, weights, cumulative, bottom, histtype, align, orientation, rwidth, log, color, label, stacked, data, **kwargs)
   2851         orientation='vertical', rwidth=None, log=False, color=None,
   2852         label=None, stacked=False, *, data=None, **kwargs):
-> 2853     return gca().hist(
   2854         x, bins=bins, range=range, density=density, weights=weights,
   2855         cumulative=cumulative, bottom=bottom, histtype=histtype,

~\Anaconda3\lib\site-packages\matplotlib\__init__.py in inner(ax, data, *args, **kwargs)
   1359     def inner(ax, *args, data=None, **kwargs):
   1360         if data is None:
-> 1361             return func(ax, *map(sanitize_sequence, args), **kwargs)
   1362
   1363         bound = new_sig.bind(ax, *args, **kwargs)

~\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py in hist(self, x, bins, range, density, weights, cumulative, bottom, histtype, align, orientation, rwidth, log, color, label, stacked, **kwargs)
   6909             if patch:
   6910                 p = patch[0]
-> 6911                 p.update(kwargs)
   6912                 if lbl is not None:
   6913                     p.set_label(lbl)

~\Anaconda3\lib\site-packages\matplotlib\artist.py in update(self, props)
   1060                     func = getattr(self, f"set_{k}", None)
   1061                     if not callable(func):
-> 1062                         raise AttributeError(f"{type(self).__name__!r} object "
   1063                                              f"has no property {k!r}")
```

```
1064                 ret.append(func(v))

AttributeError: 'Rectangle' object has no property 'kde'
```



encoding--convert categorical value into number

In [107…

```
feature
```

Out[107…

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsep
0	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	
1	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	
2	1	122.0	alfa-romero	gas	hatchback	rwd	front	65.5	52.4	ohcv	152	
3	2	164.0	audi	gas	sedan	fwd	front	66.2	54.3	ohc	109	
4	2	164.0	audi	gas	sedan	4wd	front	66.4	54.3	ohc	136	
…	…	…	…	…	…	…	…	…	…	…	…	
200	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	
201	-1	95.0	volvo	gas	sedan	rwd	front	68.8	55.5	ohc	141	
202	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohcv	173	
203	-1	95.0	volvo	diesel	sedan	rwd	front	68.9	55.5	ohc	145	
204	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	

204 rows × 14 columns

In [108…

```
target
```

Out[108…

```
0      13495
1      16500
2      16500
3      13950
4      17450
       ...
200    16845
```

```
201    19045
202    21485
203    22470
204    22625
Name: price, Length: 204, dtype: int64
```

onehotencoding

In [109…

```python
from sklearn.preprocessing import OneHotEncoder
```

In [110…

```python
one=OneHotEncoder()
```

In [111…

```python
one.fit_transform(feature[['fuel-type']]).toarray()
```

Out[111…

```
array([[0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
```

```
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
```

```
[0., 1.],
[1., 0.],
[0., 1.],
[1., 0.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[1., 0.],
[1., 0.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
[1., 0.],
[0., 1.],
[0., 1.],
[0., 1.],
[0., 1.],
```

```
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [0., 1.]])
```

label encoding---for output column

In [112…
```
from sklearn.preprocessing import LabelEncoder
```

In [113…
```
le=LabelEncoder()
```

In [114…
```
le.fit_transform(target)
```

Out[114…
```
array([118, 137, 137, 123, 148, 128, 150, 158, 169, 107, 136, 145, 162,
       163, 170, 174, 186, 183,   1,  11,  17,   7,  13,  49,  10,  20,
        41,  63,  66, 114,  14,  25,   5,  16,  32,  34,  34,  47,  69,
        65,  89,  90,  22,  86, 168,  96, 176, 181, 182,   2,   8,  23,
        21,  37,  95, 104, 120, 131,  65,  61,  91,  88,  93,  98, 153,
       154, 171, 173, 172, 175, 179, 180, 185, 187, 138,   4,   9,  19,
        42,  80,  62, 111, 126, 125,  28,  55,  72,  72,   6,  30,  18,
        24,  36,  35,  46,  39,  52,  58,  68,  77, 119, 124, 119, 147,
       161, 155, 106, 115, 110, 122, 130, 144, 142, 146, 141, 151, 152,
         7,  49,  10,  20,  41,  66, 112, 165, 177, 178, 184, 107,  73,
        79, 105, 108, 127, 129, 152, 157,   0,  29,  40,  31,  44,  81,
        70, 100,  38,  87,  53, 103,   3,  12,  15,  26,  48,  64,  27,
        33,  48,  45,  43,  59,  71,  54,  57,  74,  76,  60,  78,  84,
        97, 101, 149,  67,  92,  83,  94,  99, 140, 135, 132, 133,  44,
        50,  51,  56,  61,  75,  85, 102,  82, 116, 121, 109, 113, 117,
       134, 139, 156, 159, 143, 160, 164, 166, 167], dtype=int64)
```

ordinal encoding---- for input column

In [115…
```
from sklearn.preprocessing import OrdinalEncoder
```

In [116…
```
oe=OrdinalEncoder()
```

In [117…

```
oe.fit_transform(feature[['make']])
```

Out[117…

```
array([[ 0.],
       [ 0.],
       [ 0.],
       [ 1.],
       [ 1.],
       [ 1.],
       [ 1.],
       [ 1.],
       [ 1.],
       [ 1.],
       [ 2.],
       [ 2.],
       [ 2.],
       [ 2.],
       [ 2.],
       [ 2.],
       [ 2.],
       [ 2.],
       [ 2.],
       [ 3.],
       [ 3.],
       [ 3.],
       [ 4.],
       [ 4.],
       [ 4.],
       [ 4.],
       [ 4.],
       [ 4.],
       [ 4.],
       [ 4.],
       [ 4.],
       [ 4.],
       [ 5.],
       [ 5.],
       [ 5.],
       [ 5.],
       [ 5.],
       [ 5.],
       [ 5.],
       [ 5.],
       [ 5.],
       [ 5.],
       [ 5.],
       [ 5.],
       [ 5.],
       [ 5.],
       [ 6.],
       [ 6.],
       [ 6.],
       [ 6.],
       [ 7.],
       [ 7.],
       [ 7.],
       [ 8.],
       [ 8.],
       [ 8.],
       [ 8.],
       [ 8.],
       [ 8.],
       [ 8.],
       [ 8.],
       [ 8.],
       [ 8.],
       [ 8.],
       [ 8.],
       [ 8.],
       [ 8.],
```

[8.],
[8.],
[8.],
[8.],
[8.],
[9.],
[9.],
[9.],
[9.],
[9.],
[9.],
[9.],
[9.],
[9.],
[10.],
[11.],
[11.],
[11.],
[11.],
[11.],
[11.],
[11.],
[11.],
[11.],
[11.],
[11.],
[11.],
[11.],
[11.],
[12.],
[12.],
[12.],
[12.],
[12.],
[12.],
[12.],
[12.],
[12.],
[12.],
[12.],
[12.],
[12.],
[12.],
[12.],
[12.],
[12.],
[12.],
[12.],
[13.],
[13.],
[13.],
[13.],
[13.],
[13.],
[13.],
[13.],
[13.],
[13.],
[13.],
[13.],
[13.],
[14.],
[14.],
[14.],
[14.],
[14.],
[14.],
[14.],
[14.],
[15.],
[15.],
[15.],

[15.],
[15.],
[16.],
[16.],
[17.],
[17.],
[17.],
[17.],
[17.],
[17.],
[18.],
[18.],
[18.],
[18.],
[18.],
[18.],
[18.],
[18.],
[18.],
[18.],
[18.],
[18.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[19.],
[20.],
[20.],
[20.],
[20.],
[20.],
[20.],
[20.],
[20.],
[20.],
[20.],
[20.],
[20.],
[20.],

```
       [21.],
       [21.],
       [21.],
       [21.],
       [21.],
       [21.],
       [21.],
       [21.],
       [21.],
       [21.],
       [21.]])
```

we can able to encode 'categorical data' only

In [118…
```
catcol=feature.select_dtypes('object').columns #getting only categorical value
```

In [119…
```
catcol
```

Out[119…
```
Index(['make', 'fuel-type', 'body-style', 'drive-wheels', 'engine-location',
       'engine-type'],
      dtype='object')
```

In [120…
```
feature[catcol]
```

Out[120…

	make	fuel-type	body-style	drive-wheels	engine-location	engine-type
0	alfa-romero	gas	convertible	rwd	front	dohc
1	alfa-romero	gas	convertible	rwd	front	dohc
2	alfa-romero	gas	hatchback	rwd	front	ohcv
3	audi	gas	sedan	fwd	front	ohc
4	audi	gas	sedan	4wd	front	ohc
...	...	...	...	...	...	...
200	volvo	gas	sedan	rwd	front	ohc
201	volvo	gas	sedan	rwd	front	ohc
202	volvo	gas	sedan	rwd	front	ohcv
203	volvo	diesel	sedan	rwd	front	ohc
204	volvo	gas	sedan	rwd	front	ohc

204 rows × 6 columns

In [121…
```
feature[catcol]=oe.fit_transform(feature[catcol]) #do ordinal encoding for categorical co
```

In [122…
```
feature[catcol]
```

Out[122…

	make	fuel-type	body-style	drive-wheels	engine-location	engine-type
0	0.0	1.0	0.0	2.0	0.0	0.0
1	0.0	1.0	0.0	2.0	0.0	0.0

	make	fuel-type	body-style	drive-wheels	engine-location	engine-type
2	0.0	1.0	2.0	2.0	0.0	5.0
3	1.0	1.0	3.0	1.0	0.0	3.0
4	1.0	1.0	3.0	0.0	0.0	3.0
...	...	...	...	...	...	...
200	21.0	1.0	3.0	2.0	0.0	3.0
201	21.0	1.0	3.0	2.0	0.0	3.0
202	21.0	1.0	3.0	2.0	0.0	5.0
203	21.0	0.0	3.0	2.0	0.0	3.0
204	21.0	1.0	3.0	2.0	0.0	3.0

204 rows × 6 columns

scaling

In [123...]
```
from sklearn.preprocessing import MinMaxScaler
```

In [124...]
```
sc=MinMaxScaler()
```

In [125...]
```
feature=sc.fit_transform(feature)
```

In [126...]
```
feature    #it display array so we convert into dataframe
```

Out[126...]
```
array([[1.        , 0.29842932, 0.        , ..., 0.2625    , 0.22222222,
        0.28947368],
       [1.        , 0.29842932, 0.        , ..., 0.2625    , 0.22222222,
        0.28947368],
       [0.6       , 0.29842932, 0.        , ..., 0.44166667, 0.16666667,
        0.26315789],
       ...,
       [0.2       , 0.15706806, 1.        , ..., 0.35833333, 0.13888889,
        0.18421053],
       [0.2       , 0.15706806, 1.        , ..., 0.24166667, 0.36111111,
        0.28947368],
       [0.2       , 0.15706806, 1.        , ..., 0.275     , 0.16666667,
        0.23684211]])
```

In [ ]:
```
```

In [127...]
```
feature=pd.DataFrame(feature,columns=['symboling','normalized-losses','make','fuel-type',
```

In [128...]
```
feature
```

Out[128...]

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	hors
0	1.0	0.298429	0.000000	1.0	0.00	1.0	0.0	0.316667	0.083333	0.000000	0.260377	0

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	hors
1	1.0	0.298429	0.000000	1.0	0.00	1.0	0.0	0.316667	0.083333	0.000000	0.260377	0
2	0.6	0.298429	0.000000	1.0	0.50	1.0	0.0	0.433333	0.383333	0.833333	0.343396	0
3	0.8	0.518325	0.047619	1.0	0.75	0.5	0.0	0.491667	0.541667	0.500000	0.181132	0
4	0.8	0.518325	0.047619	1.0	0.75	0.0	0.0	0.508333	0.541667	0.500000	0.283019	0
...	...	...	...	...	...	...	...	...	...	...	...	
199	0.2	0.157068	1.000000	1.0	0.75	1.0	0.0	0.716667	0.641667	0.500000	0.301887	0
200	0.2	0.157068	1.000000	1.0	0.75	1.0	0.0	0.708333	0.641667	0.500000	0.301887	0
201	0.2	0.157068	1.000000	1.0	0.75	1.0	0.0	0.716667	0.641667	0.833333	0.422642	0
202	0.2	0.157068	1.000000	0.0	0.75	1.0	0.0	0.716667	0.641667	0.500000	0.316981	0
203	0.2	0.157068	1.000000	1.0	0.75	1.0	0.0	0.716667	0.641667	0.500000	0.301887	0

204 rows × 14 columns

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: