CAR PRICE PREDICTION(LINEAR REGRESSION)

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
import warnings
warnings.filterwarnings("ignore")
```

LOAD DATASET

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
0	3	?	alfa-romero	gas	convertible	rwd	front	64.1	48.8
1	3	?	alfa-romero	gas	convertible	rwd	front	64.1	48.8
2	1	?	alfa-romero	gas	hatchback	rwd	front	65.5	52.4
3	2	164	audi	gas	sedan	fwd	front	66.2	54.3
4	2	164	audi	gas	sedan	4wd	front	66.4	54.3

In [4]:

```
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    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 [5]:

```
df.describe()
```

Out[5]:

	symboling	width	height	engine-size	city-mpg	highway-mpg	
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.0
mean	0.834146	65.907805	53.724878	126.907317	25.219512	30.751220	1322
std	1.245307	2.145204	2.443522	41.642693	6.542142	6.886443	7902
min	-2.000000	60.300000	47.800000	61.000000	13.000000	16.000000	5118
25%	0.000000	64.100000	52.000000	97.000000	19.000000	25.000000	7788
50%	1.000000	65.500000	54.100000	120.000000	24.000000	30.000000	1034
75%	2.000000	66.900000	55.500000	141.000000	30.000000	34.000000	1650
max	3.000000	72.300000	59.800000	326.000000	49.000000	54.000000	4540

HANDALING THE MISSING VALUE

In [6]:
```
df.isna().sum()
```

Out[6]:
```
symboling           0
normalized-losses   0
make                0
fuel-type           0
body-style          0
drive-wheels        0
engine-location     0
width               0
height              0
engine-type         0
engine-size         0
horsepower          0
city-mpg            0
highway-mpg         0
price               0
dtype: int64
```

In [7]:
```
df["normalized-losses"].value_counts()
```

Out[7]:

?	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 [8]:

```
df["normalized-losses"].replace("?",np.nan,inplace=True)
```

In [9]:

```
df["horsepower"].replace("?",np.nan,inplace=True)
df["horsepower"]=df["horsepower"].astype("float64")
```

In [10]:

```
si=SimpleImputer(missing_values=np.nan,strategy="mean")
```

In [11]:

```
df[["normalized-losses"]]=si.fit_transform(df[["normalized-losses"]])
```

In [12]:

```
df[["horsepower"]]=si.fit_transform(df[["horsepower"]])
```

In [13]:

```
df.head()
```

Out[13]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height
0	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8
1	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8
2	1	122.0	alfa-romero	gas	hatchback	rwd	front	65.5	52.4
3	2	164.0	audi	gas	sedan	fwd	front	66.2	54.3
4	2	164.0	audi	gas	sedan	4wd	front	66.4	54.3

sperate columns on features and target

In [14]:

```
features=df.iloc[:,:-1]
```

In [15]:

```
features
```

Out[15]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	hei
0	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.
1	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.
2	1	122.0	alfa-romero	gas	hatchback	rwd	front	65.5	52.
3	2	164.0	audi	gas	sedan	fwd	front	66.2	54.
4	2	164.0	audi	gas	sedan	4wd	front	66.4	54.
...	...	...	...	...	...	...	...	...	...
200	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.
201	-1	95.0	volvo	gas	sedan	rwd	front	68.8	55.
202	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.
203	-1	95.0	volvo	diesel	sedan	rwd	front	68.9	55.
204	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.

205 rows × 14 columns

In [16]:

```
target=df.iloc[:,-1]
```

In [17]:

```
target
```

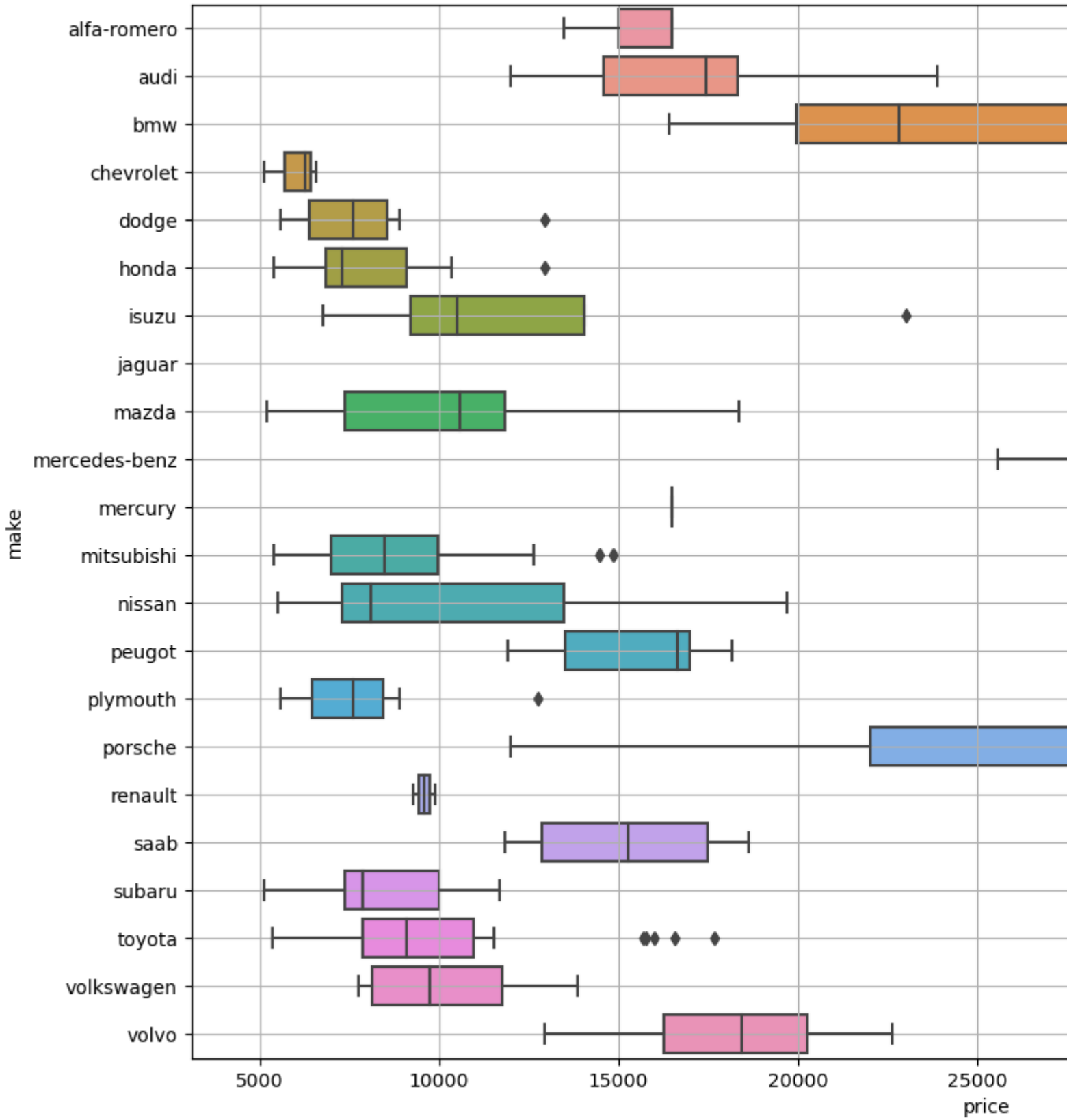
Out[17]:

```
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
```

OUTLIERS DETECTION

In [18]:

```
plt.figure(figsize=(15,10))
sns.boxplot(data=features,x=target,y="make");
plt.grid()
```



OUTLIERS HANDALING

In [19]:

```
features[(features.make=="plymouth")&(target>10000)]
```

Out[19]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	hei
124	3	122.0	plymouth	gas	hatchback	rwd	front	66.3	50.

In [20]:

```
features.drop(124,axis=0,inplace=True)
```

In [21]:

```
features[(features.make=="toyota")&(target>15000)]
```

Out[21]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	hei
172	2	134.0	toyota	gas	convertible	rwd	front	65.6	53.
178	3	197.0	toyota	gas	hatchback	rwd	front	67.7	52.
179	3	197.0	toyota	gas	hatchback	rwd	front	67.7	52.
180	-1	90.0	toyota	gas	sedan	rwd	front	66.5	54.
181	-1	122.0	toyota	gas	wagon	rwd	front	66.5	54.

In [22]:
```
features.drop([172,178,179,180,181],axis=0,inplace=True)
```
In [23]:
```
features[(features.make=="mitsubishi")&(target>12000)]
```
Out[23]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	he
82	3	122.0	mitsubishi	gas	hatchback	fwd	front	66.3	50
83	3	122.0	mitsubishi	gas	hatchback	fwd	front	66.3	50
84	3	122.0	mitsubishi	gas	hatchback	fwd	front	66.3	50

In [24]:
```
features.drop([83,84],axis=0,inplace=True)
```
In [25]:
```
features[(features.make=="honda")&(target>11000)]
```
Out[25]:

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

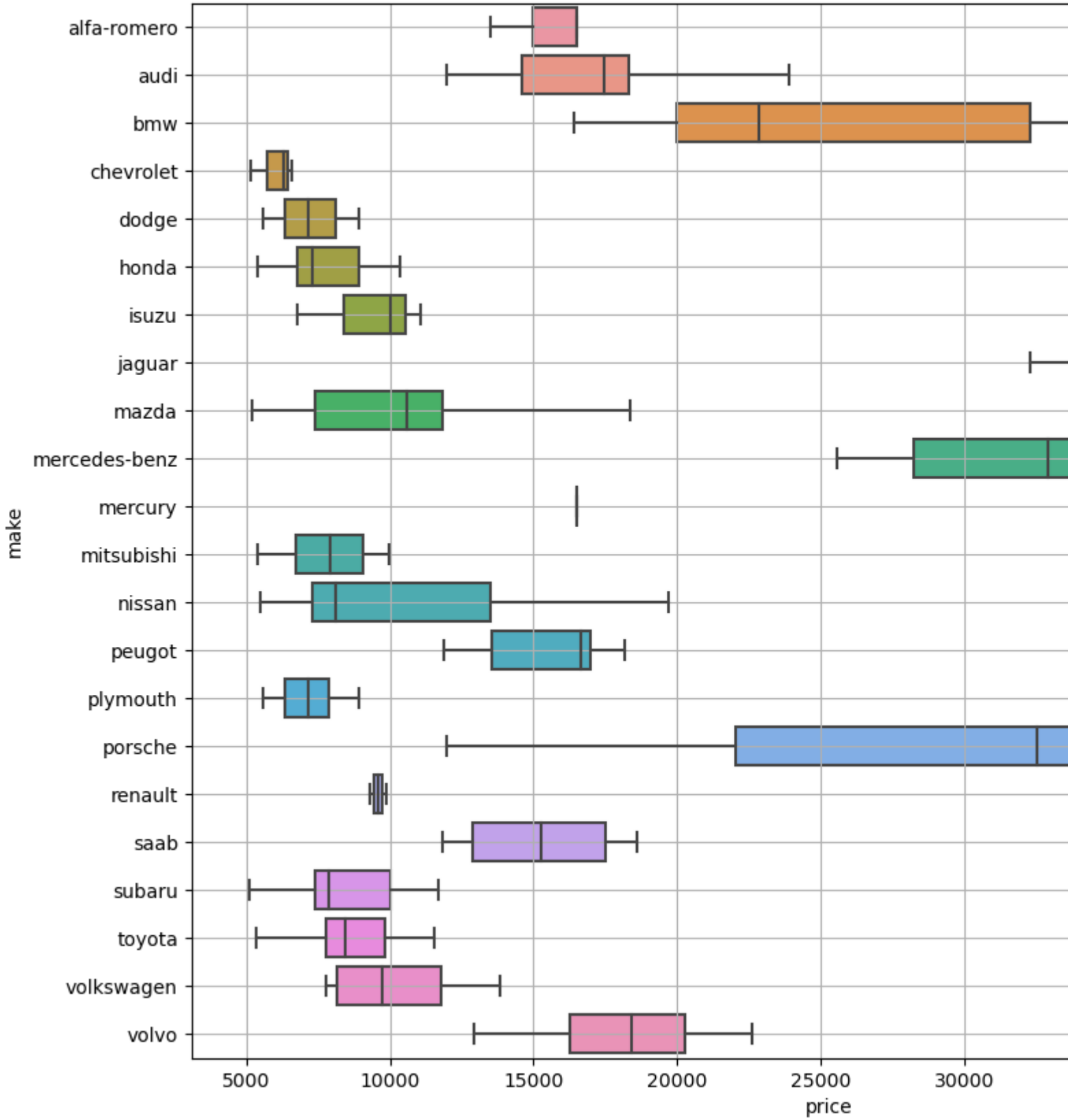
In [26]:
```
features.drop(41,axis=0,inplace=True)
```
In [27]:
```
features[(features.make=="dodge")&(target>11000)]
```
Out[27]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height
29	3	145.0	dodge	gas	hatchback	fwd	front	66.3	50.2

In [28]:
```
features.drop(29,axis=0,inplace=True)
```
In [29]:
```
features[(features.make=="isuzu")&(target>20000)]
```
Out[29]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engi t
45	0	122.0	isuzu	gas	sedan	fwd	front	63.6	52.0	ohc

In [30]:
```
target.drop([29,41,83,84,124,172,178,179,180,181,82],axis=0,inplace=True)
```
In [31]:
```
features.drop(45,axis=0,inplace=True)
```
In [32]:
```
plt.figure(figsize=(12,10))
sns.boxplot(data=features,x=target,y="make");
plt.grid()
```



SKEW DETECTION

In [33]:

```
colname=features.select_dtypes(["int64","float64"]).columns
```

In [34]:

```
colname
```

Out[34]:

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

In [35]:

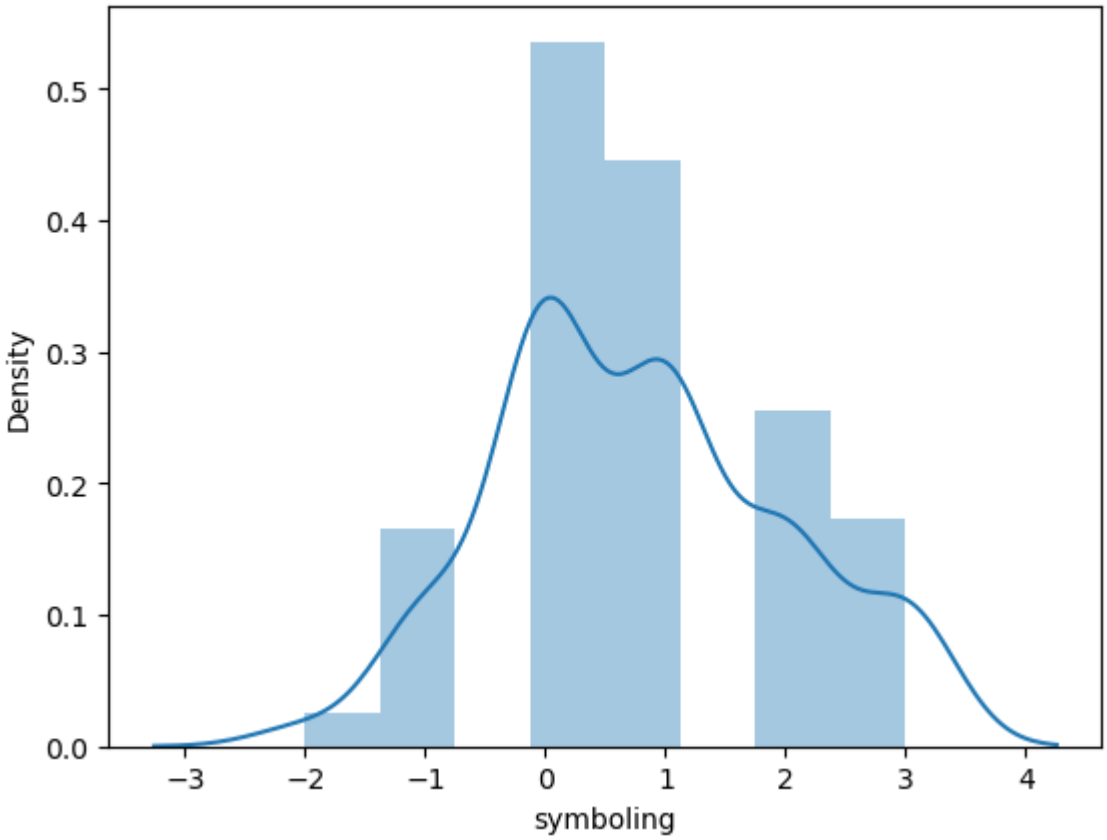
```
from scipy.stats import skew
skew(features["height"])
```

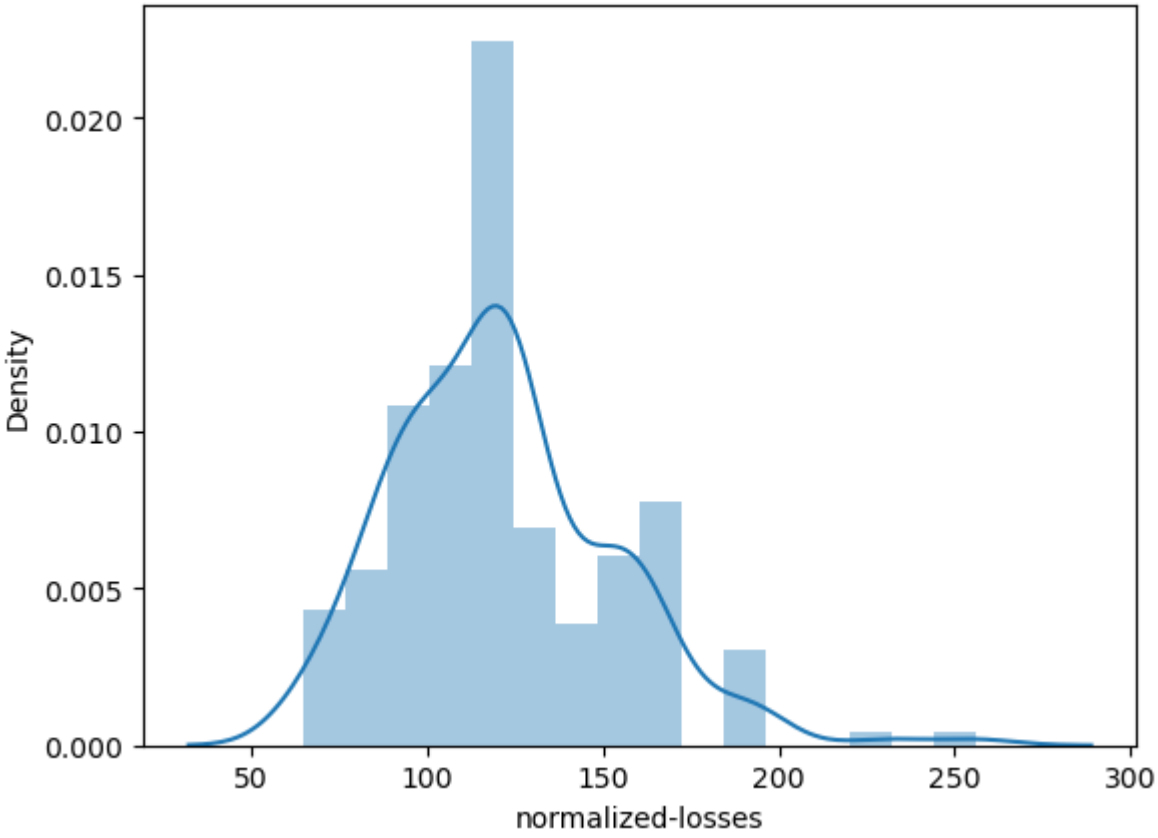
```
Out[35]:
        0.013839962443639326
```

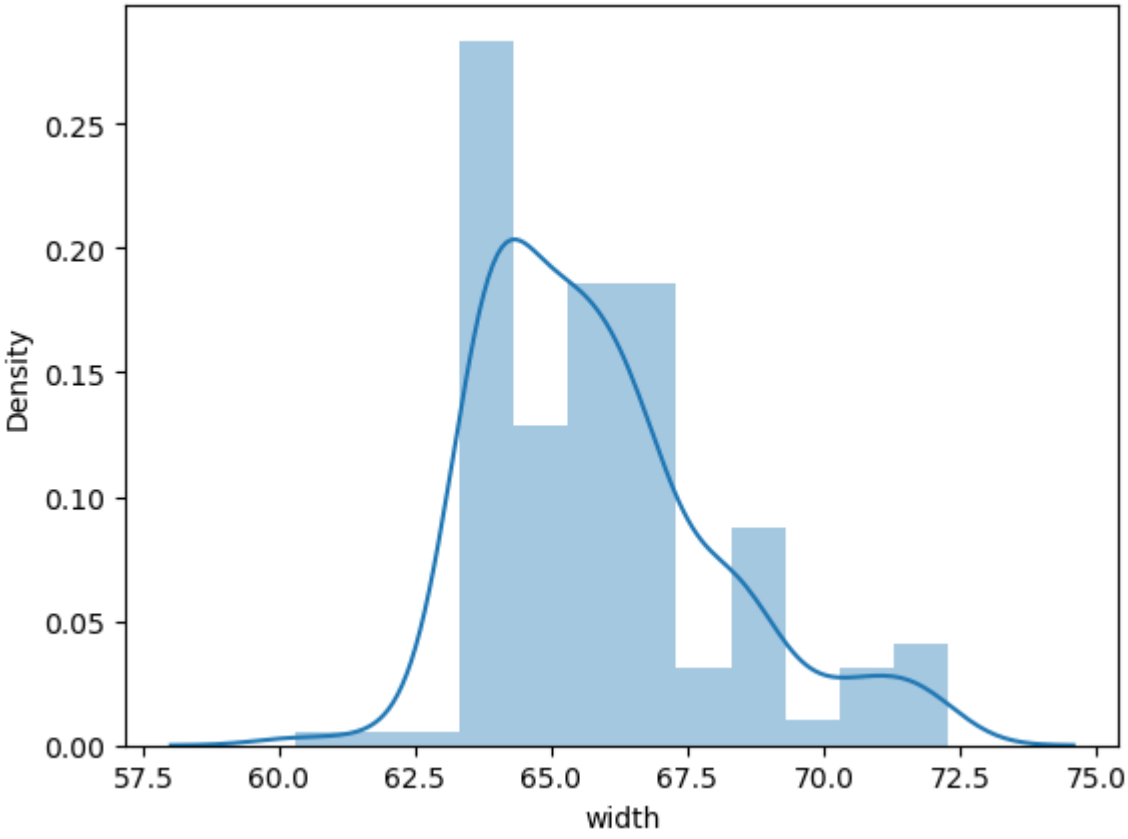
```
In [36]:
```

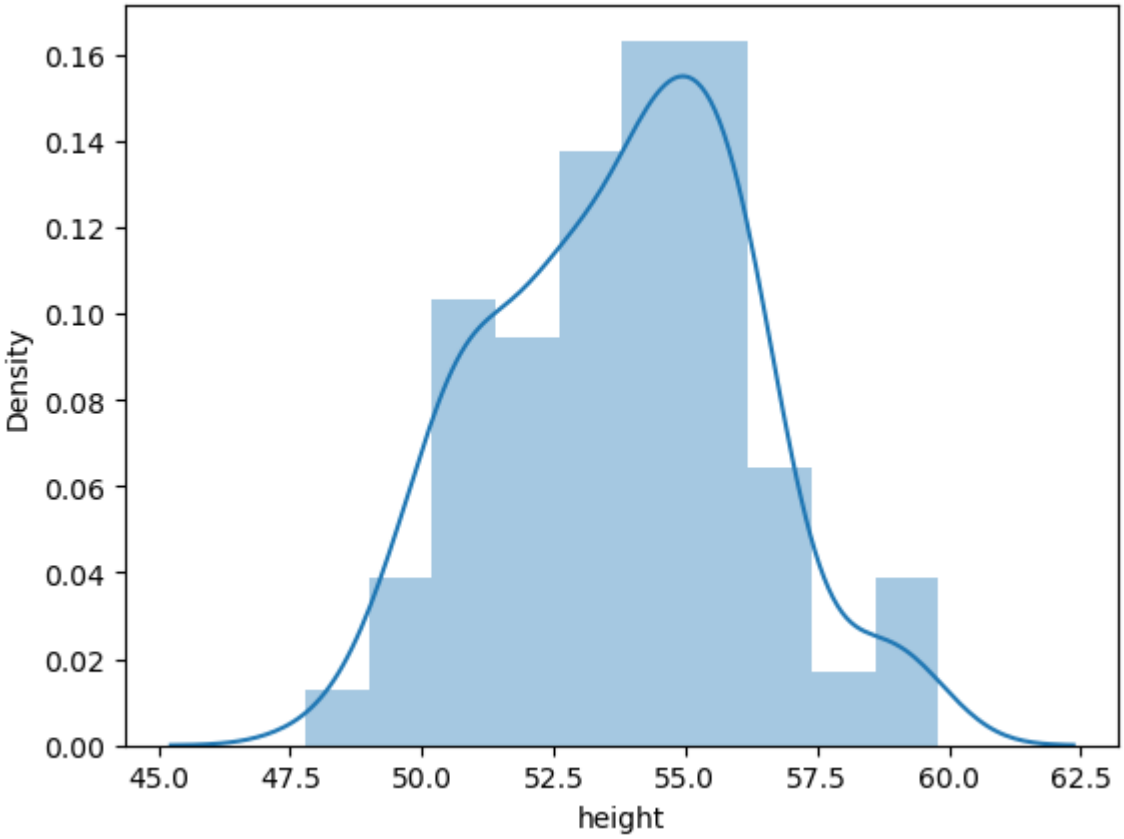
```
for i in features[colname]:
    print(i)
    print(skew(features[i]))
    plt.figure()
    sns.distplot(features[i]);
```

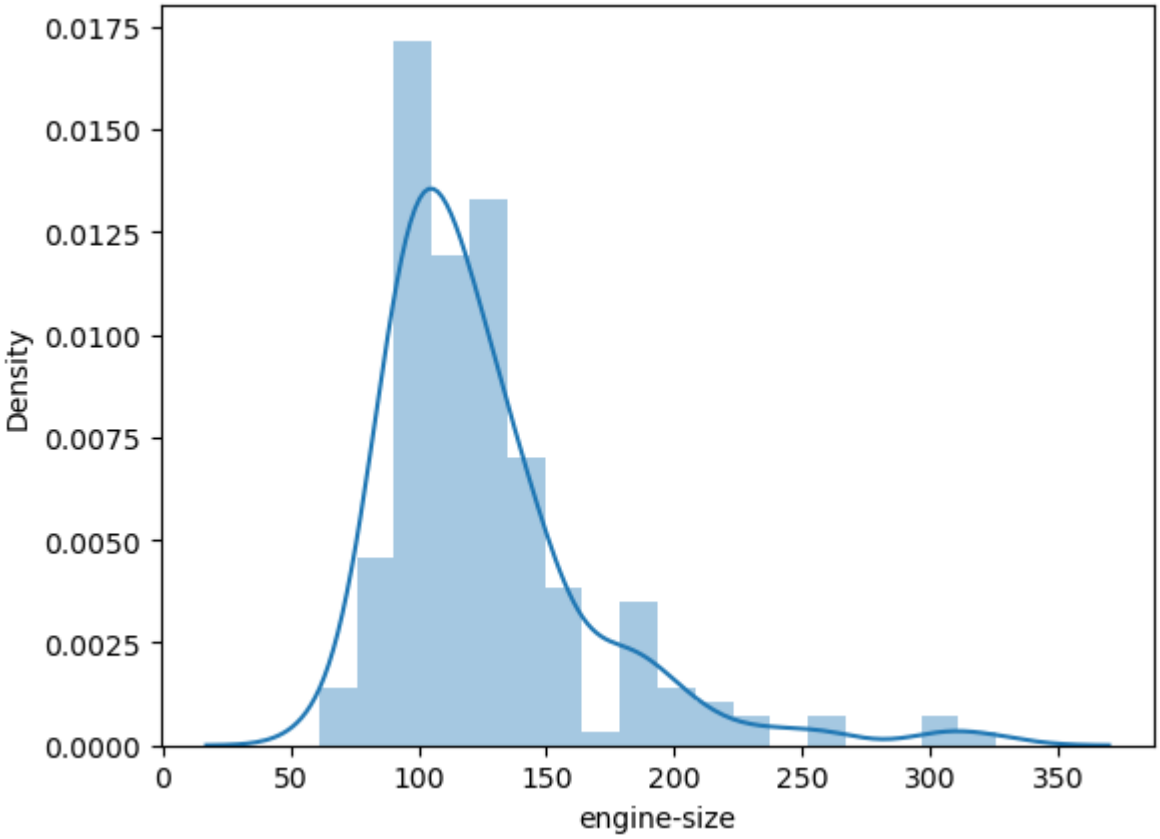
```
symboling
0.21386866184357742
normalized-losses
0.848205953606264
width
0.9140400320504322
height
0.013839962443639326
engine-size
2.0541257626466156
horsepower
1.5556576549504106
city-mpg
0.5999073033714895
highway-mpg
0.4760310091695327
```

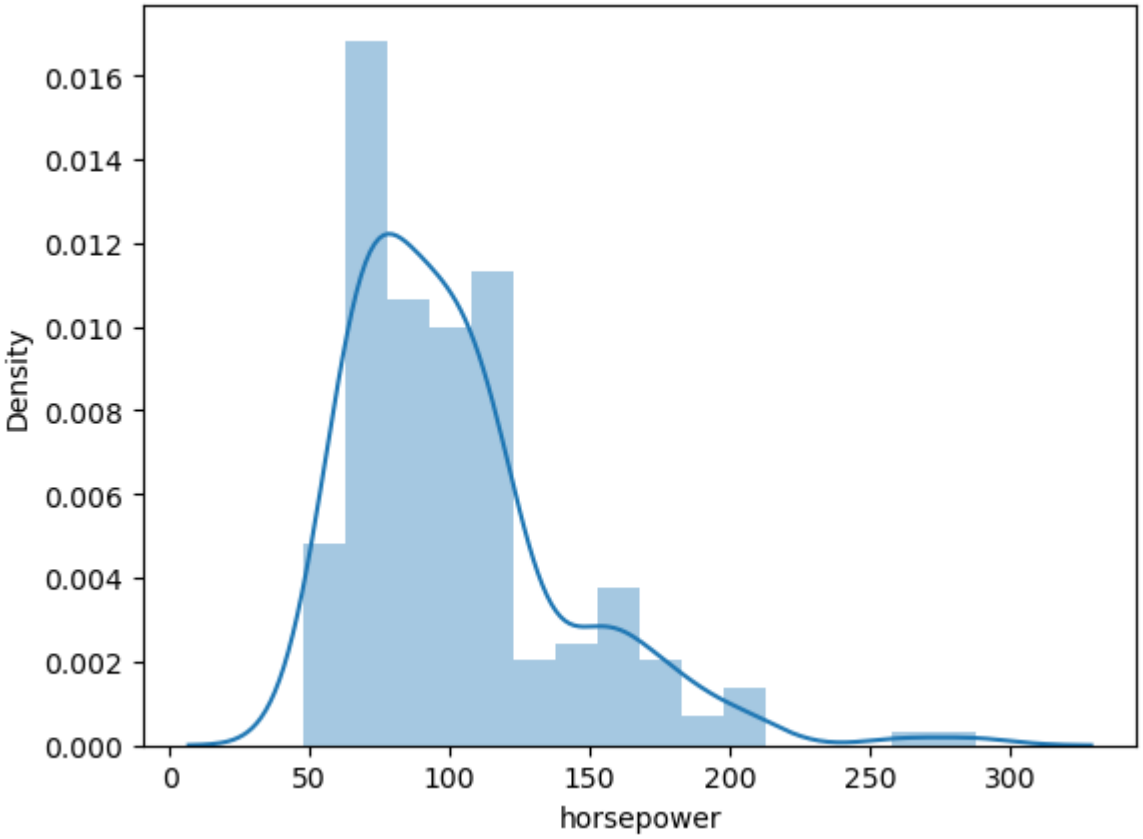


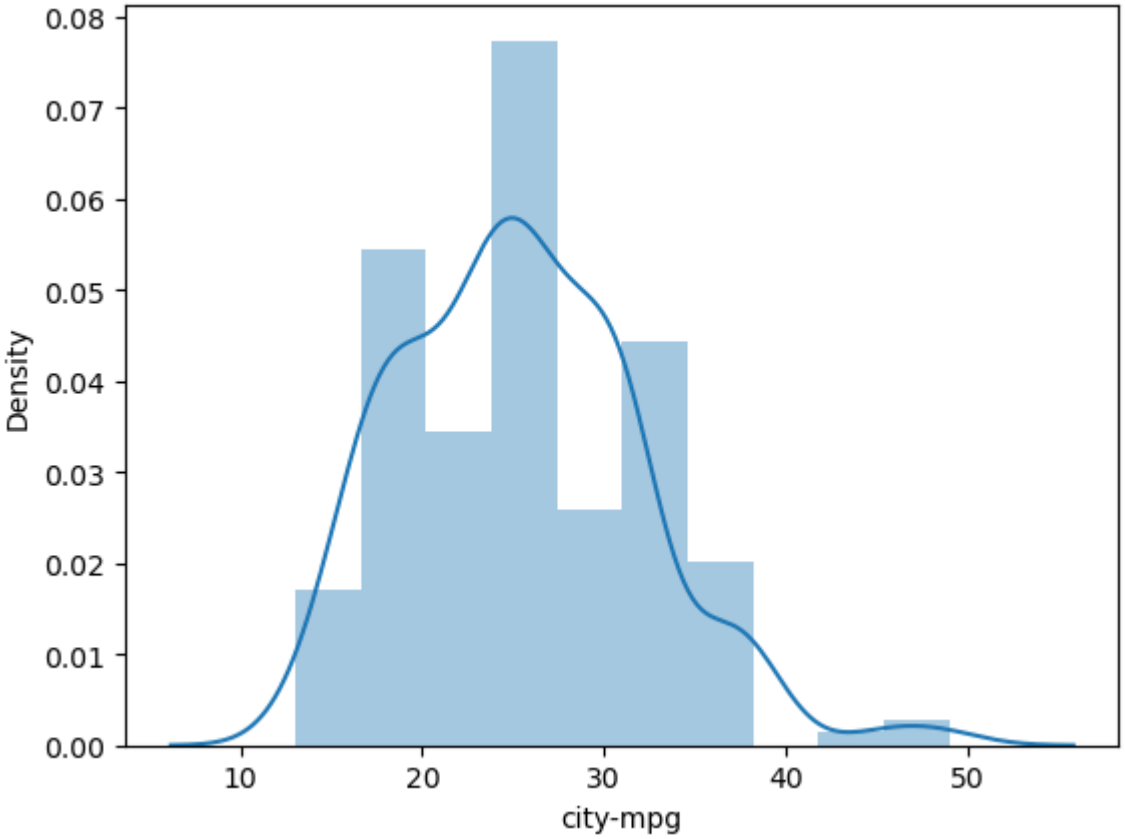

0.020
0.015
0.010
0.005
0.000
Density
50
100
150
200
250
300
normalized-losses

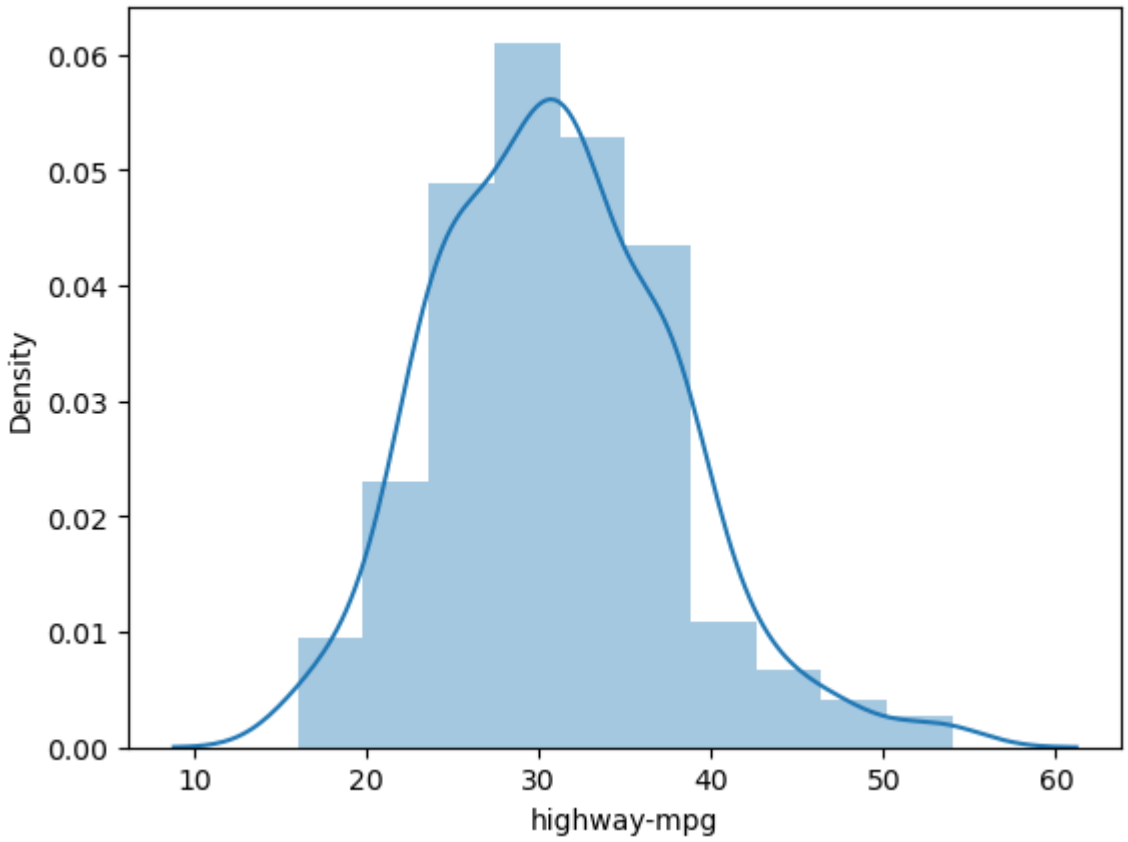

0.25
0.20
0.15
0.10
0.05
0.00
Density
57.5
60.0
62.5
65.0
67.5
70.0
72.5
75.0
width


0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
Density
45.0
47.5
50.0
52.5
55.0
57.5
60.0
62.5
height


0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
Density
0
50
100
150
200
250
300
350
engine-size


0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
Density
0
50
100
150
200
250
300
horsepower


0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
Density
10
20
30
40
50
city-mpg



CHECKING CORELATION OF TARGET

In [37]:

```
pd.concat([features,target],axis=1).corr().style.background_gradient()
```

Out[37]:

	symboling	normalized-losses	width	height	engine-size	horsepower	c
symboling	1.000000	0.447922	-0.272388	-0.521495	-0.153671	0.027074	0
normalized-losses	0.447922	1.000000	0.066622	-0.368540	0.090258	0.183385	-
width	-0.272388	0.066622	1.000000	0.296011	0.735112	0.643906	-
height	-0.521495	-0.368540	0.296011	1.000000	0.096041	-0.078245	-
engine-size	-0.153671	0.090258	0.735112	0.096041	1.000000	0.803956	-
horsepower	0.027074	0.183385	0.643906	-0.078245	0.803956	1.000000	-
city-mpg	0.007189	-0.212276	-0.641401	-0.078815	-0.642711	-0.797166	1
highway-mpg	0.084238	-0.168904	-0.677911	-0.142926	-0.667078	-0.761009	0
price	-0.096215	0.129980	0.730630	0.147417	0.871044	0.771608	-

In [38]:

```
pd.concat([features,target],axis=1).corr()["price"]
```

Out[38]:

```
symboling           -0.096215
normalized-losses    0.129980
width                0.730630
height               0.147417
engine-size          0.871044
horsepower           0.771608
city-mpg            -0.682415
highway-mpg         -0.707051
price                1.000000
Name: price, dtype: float64
```

In [39]:

```
features["height"].value_counts()
```

Out[39]:

```
50.8    14
55.7    12
54.5    10
55.5     9
52.0     9
54.3     8
56.7     8
52.6     7
56.1     7
54.1     7
51.6     7
54.9     6
52.8     6
53.0     5
50.6     5
53.7     5
55.1     5
49.6     4
53.3     4
58.7     4
52.5     3
59.1     3
56.2     3
49.7     3
57.5     3
53.5     3
54.4     2
53.9     2
56.3     2
50.5     2
59.8     2
56.5     2
54.7     2
48.8     2
50.2     2
49.4     2
51.4     2
51.0     1
54.8     1
55.4     1
56.0     1
55.2     1
53.2     1
47.8     1
55.9     1
52.4     1
55.6     1
53.1     1
58.3     1
Name: height, dtype: int64
```

In [40]:

```
features["height"]=np.log(features["height"])
```

In [41]:

```
skew(features["height"])
```

Out[41]:

```
-0.09062993710381134
```

ENCODING

ONE HOT ENCODING

In [42]:

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

In [43]:

```
one=OneHotEncoder()
one.fit_transform(features[["fuel-type"]]).toarray()
```

```
Out[43]:
         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.],
                [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.],
[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.],
[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.],
[1., 0.],
[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.]])
```

In [44]:

```
features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 194 entries, 0 to 204
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   symboling          194 non-null    int64  
 1   normalized-losses  194 non-null    float64
 2   make               194 non-null    object 
 3   fuel-type          194 non-null    object 
 4   body-style         194 non-null    object 
 5   drive-wheels       194 non-null    object 
 6   engine-location    194 non-null    object 
 7   width              194 non-null    float64
 8   height             194 non-null    float64
 9   engine-type        194 non-null    object 
 10  engine-size        194 non-null    int64  
 11  horsepower         194 non-null    float64
 12  city-mpg           194 non-null    int64  
 13  highway-mpg        194 non-null    int64  
dtypes: float64(4), int64(4), object(6)
memory usage: 26.8+ KB
```

ORDINAL ENCODING

In [45]:

```
from sklearn.preprocessing import OrdinalEncoder
```

In [46]:

```
oe=OrdinalEncoder()
oe.fit_transform(features[["make"]])
```

```
Out[46]:
        array([[ 0.],
               [ 0.],
               [ 0.],
               [ 1.],
               [ 1.],
               [ 1.],
               [ 1.],
               [ 1.],
               [ 1.],
               [ 1.],
               [ 2.],
               [ 2.],
               [ 2.],
               [ 2.],
               [ 2.],
               [ 2.],
               [ 2.],
               [ 2.],
               [ 3.],
               [ 3.],
               [ 3.],
               [ 4.],
               [ 4.],
               [ 4.],
               [ 4.],
               [ 4.],
               [ 4.],
               [ 4.],
               [ 4.],
               [ 5.],
               [ 5.],
               [ 5.],
               [ 5.],
               [ 5.],
               [ 5.],
               [ 5.],
               [ 5.],
               [ 5.],
               [ 5.],
               [ 5.],
               [ 5.],
               [ 6.],
               [ 6.],
               [ 6.],
               [ 7.],
               [ 7.],
               [ 7.],
               [ 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.],
[10.],
[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.],
[12.],
[13.],
[13.],
[13.],
[13.],
[13.],
[13.],
[13.],
[13.],
[13.],
[13.],
[13.],
[13.],
[13.],
[14.],
[14.],
[14.],
[14.],
[14.],
[14.],
[14.],
[15.],
[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.],
[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.],
[20.],
[20.],
[20.],
[20.],
[20.],
[20.],
[20.],
[20.],
[20.],
[20.],
[20.],
[20.],
[20.],
[20.],
[21.],
[21.],
[21.],
[21.],
[21.],
[21.],
[21.],
[21.],

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

In [47]:

```
oe.fit_transform(features[["body-style"]])
```

Out[47]:

```
array([[0.],
       [0.],
       [2.],
       [3.],
       [3.],
       [3.],
       [3.],
       [4.],
       [3.],
       [2.],
       [3.],
       [3.],
       [3.],
       [3.],
       [3.],
       [3.],
       [3.],
       [3.],
       [3.],
       [2.],
       [2.],
       [3.],
       [2.],
       [2.],
       [2.],
       [2.],
       [2.],
       [3.],
       [3.],
       [3.],
       [4.],
       [2.],
       [2.],
       [2.],
       [2.],
       [2.],
       [2.],
       [3.],
       [4.],
       [2.],
       [2.],
       [3.],
       [3.],
       [3.],
       [3.],
       [3.],
       [3.],
       [2.],
       [3.],
       [3.],
       [3.],
       [2.],
       [2.],
       [2.],
       [2.],
       [3.],
       [3.],
       [2.],
       [2.],
       [2.],
       [2.],
       [2.],
       [2.],
       [3.],
       [2.],
       [3.],
       [3.],
       [2.],
       [3.],
```

[3.],
[3.],
[4.],
[1.],
[3.],
[3.],
[0.],
[3.],
[1.],
[2.],
[2.],
[2.],
[2.],
[2.],
[2.],
[2.],
[2.],
[3.],
[3.],
[3.],
[3.],
[3.],
[3.],
[3.],
[3.],
[3.],
[4.],
[3.],
[2.],
[3.],
[4.],
[1.],
[2.],
[3.],
[3.],
[4.],
[3.],
[2.],
[2.],
[2.],
[3.],
[3.],
[4.],
[4.],
[3.],
[3.],
[4.],
[4.],
[3.],
[3.],
[3.],
[2.],
[2.],
[2.],
[3.],
[3.],
[4.],
[2.],
[1.],
[1.],
[0.],
[2.],
[4.],
[2.],
[2.],

[3.],
[2.],
[3.],
[2.],
[3.],
[2.],
[2.],
[2.],
[3.],
[3.],
[3.],
[3.],
[3.],
[4.],
[4.],
[4.],
[4.],
[2.],
[2.],
[2.],
[4.],
[4.],
[4.],
[3.],
[2.],
[3.],
[2.],
[3.],
[2.],
[3.],
[3.],
[2.],
[3.],
[2.],
[1.],
[1.],
[2.],
[1.],
[2.],
[3.],
[3.],
[2.],
[3.],
[2.],
[3.],
[3.],
[3.],
[3.],
[3.],
[3.],
[3.],
[0.],
[2.],
[3.],
[3.],
[4.],
[3.],
[4.],
[3.],
[4.],
[3.],
[4.],
[3.],
[3.],

```
               [3.],
               [3.],
               [3.]])
```

In [48]:
```
catcol=features.select_dtypes("object").columns
```
In [49]:
```
catcol
```
Out[49]:
```
Index(['make', 'fuel-type', 'body-style', 'drive-wheels', 'engine-location',
       'engine-type'],
      dtype='object')
```

In [50]:
```
features[catcol]=oe.fit_transform(features[catcol])
```

SCALING

#MINMAX SCALING

In [51]:
```
from sklearn.preprocessing import MinMaxScaler
```
In [52]:
```
sc=MinMaxScaler()
features=sc.fit_transform(features)
```
In [53]:
```
features=pd.DataFrame(features)
```
In [54]:
```
features
```
Out[54]:

	0	1	2	3	4	5	6	7	8	9	1
0	1.0	0.298429	0.000000	1.0	0.00	1.0	0.0	0.316667	0.092440	0.000000	0.26037
1	1.0	0.298429	0.000000	1.0	0.00	1.0	0.0	0.316667	0.092440	0.000000	0.26037
2	0.6	0.298429	0.000000	1.0	0.50	1.0	0.0	0.433333	0.410219	0.833333	0.34339
3	0.8	0.518325	0.047619	1.0	0.75	0.5	0.0	0.491667	0.569241	0.500000	0.18113
4	0.8	0.518325	0.047619	1.0	0.75	0.0	0.0	0.508333	0.569241	0.500000	0.28301
...	...	...	...	...	...	...	...	...	...	...	...
189	0.2	0.157068	1.000000	1.0	0.75	1.0	0.0	0.716667	0.666833	0.500000	0.30188
190	0.2	0.157068	1.000000	1.0	0.75	1.0	0.0	0.708333	0.666833	0.500000	0.30188
191	0.2	0.157068	1.000000	1.0	0.75	1.0	0.0	0.716667	0.666833	0.833333	0.42264
192	0.2	0.157068	1.000000	0.0	0.75	1.0	0.0	0.716667	0.666833	0.500000	0.31698
193	0.2	0.157068	1.000000	1.0	0.75	1.0	0.0	0.716667	0.666833	0.500000	0.30188

194 rows × 14 columns

STANDARD SCALING

In [55]:
```
from sklearn.preprocessing import StandardScaler
sd=StandardScaler()
features=sd.fit_transform(features)
```

In [56]:

```
features=pd.DataFrame(features)
```

In [57]:

```
features
```

Out[57]:

	0	1	2	3	4	5	6	
0	1.846173	0.019088	-1.934007	0.339032	-3.111634	1.234608	-0.125327	-0
1	1.846173	0.019088	-1.934007	0.339032	-3.111634	1.234608	-0.125327	-0
2	0.176441	0.019088	-1.934007	0.339032	-0.748984	1.234608	-0.125327	-0
3	1.011307	1.359859	-1.774620	0.339032	0.432341	-0.566249	-0.125327	0.
4	1.011307	1.359859	-1.774620	0.339032	0.432341	-2.367105	-0.125327	0.
...	...	...	...	...	...	...	...	..
189	-1.493292	-0.842836	1.413123	0.339032	0.432341	1.234608	-0.125327	1.
190	-1.493292	-0.842836	1.413123	0.339032	0.432341	1.234608	-0.125327	1.
191	-1.493292	-0.842836	1.413123	0.339032	0.432341	1.234608	-0.125327	1.
192	-1.493292	-0.842836	1.413123	-2.949576	0.432341	1.234608	-0.125327	1.
193	-1.493292	-0.842836	1.413123	0.339032	0.432341	1.234608	-0.125327	1.

194 rows × 14 columns

LINEAR REGRESSION

split train and test

In [58]:

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(features,target,test_size=0.2)
```

In [59]:

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(xtrain,ytrain)
```

Out[59]:

```
LinearRegression()
```

In [60]:

```
ypredict=lr.predict(xtest)
```

In [61]:

```
ypredict
```

Out[61]:

```
array([ 2645.08787283, 17133.73669145, 23586.65510901, 13273.90517756,
       13961.05716199, 18529.89394501, 11721.88426459,  5097.11613548,
        7785.20847432,  7013.90119517,  5009.91985575, 17360.41943718,
        9077.32796142,  5986.58434894,  4127.65151971,  2953.92104219,
        7808.90944729, 11185.07724303, 15393.4464199 , 20671.22430328,
       10240.72447238, 13467.98804116, 15054.74498827,  7568.03165973,
        8578.32244881,  1175.8107059 , 20373.16063022, 14542.72778456,
        9478.39620038,  9944.99979285, 22700.70078227, 18601.00238428,
       12994.65226522,  9045.92906686, 17234.53482065, 32424.95568543,
        7016.34183808, 19199.64331701, 23647.31481222])
```

In [ ]:

In [ ]: