Perfrom EDA and Preprocessing

In [ ]:

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

In [2]:

```
#load the dataset
df=pd.read_csv("cars.csv")
```

In [35]:

```
df.head()
```

Out[35]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg	price
0	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111.0	21	27	13495
1	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111.0	21	27	16500
2	1	122.0	alfa-romero	gas	hatchback	rwd	front	65.5	52.4	ohcv	152	154.0	19	26	16500
3	2	164.0	audi	gas	sedan	fwd	front	66.2	54.3	ohc	109	102.0	24	30	13950
4	2	164.0	audi	gas	sedan	4wd	front	66.4	54.3	ohc	136	115.0	18	22	17450

handling missing values

In [36]:

```
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
```

In [37]:

```
df.isna().sum()
```

Out[37]:

```
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 [38]:

```
df.dropna()
```

Out[38]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg	price
0	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111.0	21	27	13495
1	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111.0	21	27	16500
2	1	122.0	alfa-romero	gas	hatchback	rwd	front	65.5	52.4	ohcv	152	154.0	19	26	16500
3	2	164.0	audi	gas	sedan	fwd	front	66.2	54.3	ohc	109	102.0	24	30	13950
4	2	164.0	audi	gas	sedan	4wd	front	66.4	54.3	ohc	136	115.0	18	22	17450
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
200	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	114.0	23	28	16845
201	-1	95.0	volvo	gas	sedan	rwd	front	68.8	55.5	ohc	141	160.0	19	25	19045
202	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohcv	173	134.0	18	23	21485
203	-1	95.0	volvo	diesel	sedan	rwd	front	68.9	55.5	ohc	145	106.0	26	27	22470
204	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	114.0	19	25	22625

205 rows × 15 columns

In [39]:

```
df.dropna(how="all",subset=["normalized-losses","symboling"])
```

Out[39]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg	price
0	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111.0	21	27	13495
1	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111.0	21	27	16500
2	1	122.0	alfa-romero	gas	hatchback	rwd	front	65.5	52.4	ohcv	152	154.0	19	26	16500
3	2	164.0	audi	gas	sedan	fwd	front	66.2	54.3	ohc	109	102.0	24	30	13950
4	2	164.0	audi	gas	sedan	4wd	front	66.4	54.3	ohc	136	115.0	18	22	17450
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
200	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	114.0	23	28	16845
201	-1	95.0	volvo	gas	sedan	rwd	front	68.8	55.5	ohc	141	160.0	19	25	19045
202	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohcv	173	134.0	18	23	21485
203	-1	95.0	volvo	diesel	sedan	rwd	front	68.9	55.5	ohc	145	106.0	26	27	22470
204	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	114.0	19	25	22625

205 rows × 15 columns

In [40]:

```
df.fillna(10)
```

Out[40]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg	price
0	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111.0	21	27	13495
1	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111.0	21	27	16500
2	1	122.0	alfa-romero	gas	hatchback	rwd	front	65.5	52.4	ohcv	152	154.0	19	26	16500
3	2	164.0	audi	gas	sedan	fwd	front	66.2	54.3	ohc	109	102.0	24	30	13950
4	2	164.0	audi	gas	sedan	4wd	front	66.4	54.3	ohc	136	115.0	18	22	17450
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..
200	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	114.0	23	28	16845
201	-1	95.0	volvo	gas	sedan	rwd	front	68.8	55.5	ohc	141	160.0	19	25	19045
202	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohcv	173	134.0	18	23	21485
203	-1	95.0	volvo	diesel	sedan	rwd	front	68.9	55.5	ohc	145	106.0	26	27	22470
204	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	114.0	19	25	22625

205 rows × 15 columns

In [41]:

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

```
Out[41]: 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 [42]: df["normalized-losses"].value_counts()
```

Out[42]:

```
122.0    45
161.0    11
91.0      8
150.0     7
128.0     6
134.0     6
104.0     6
95.0      5
102.0     5
103.0     5
74.0      5
85.0      5
65.0      5
94.0      5
168.0     5
106.0     4
148.0     4
118.0     4
93.0      4
83.0      3
101.0     3
115.0     3
154.0     3
125.0     3
137.0     3
108.0     2
87.0      2
119.0     2
194.0     2
197.0     2
89.0      2
158.0     2
192.0     2
113.0     2
188.0     2
81.0      2
110.0     2
145.0     2
129.0     2
164.0     2
153.0     2
186.0     1
107.0     1
78.0      1
231.0     1
77.0      1
142.0     1
98.0      1
121.0     1
90.0      1
256.0     1
Name: normalized-losses, dtype: int64
```

In [43]:

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

In [44]:

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

In [45]:

```
df["normalized-losses"]=df["normalized-losses"].astype("float64")
```

In [46]:

```
Nmean=df["normalized-losses"].mean()
Nmean
```

Out[46]:

```
122.0
```

In [47]:

```
df["normalized-losses"].fillna(Nmean,inplace=True)
df
```

Out[47]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg	price
0	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111.0	21	27	13495
1	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111.0	21	27	16500
2	1	122.0	alfa-romero	gas	hatchback	rwd	front	65.5	52.4	ohcv	152	154.0	19	26	16500
3	2	164.0	audi	gas	sedan	fwd	front	66.2	54.3	ohc	109	102.0	24	30	13950
4	2	164.0	audi	gas	sedan	4wd	front	66.4	54.3	ohc	136	115.0	18	22	17450
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	114.0	23	28	16845
201	-1	95.0	volvo	gas	sedan	rwd	front	68.8	55.5	ohc	141	160.0	19	25	19045
202	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohcv	173	134.0	18	23	21485
203	-1	95.0	volvo	diesel	sedan	rwd	front	68.9	55.5	ohc	145	106.0	26	27	22470
204	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	114.0	19	25	22625

205 rows × 15 columns

In [48]:

```python
from sklearn.impute import SimpleImputer
```

In [90]:

```python
si=SimpleImputer(missing_values=np.nan,strategy="mean")
df[["normalized-losses","horsepower"]]=si.fit_transform(df[["normalized-losses","horsepower"]])
```

In [91]:

```python
#split the dateset
feature=df.iloc[:,:-1]
target=df.iloc[:,-1]
```

In [51]:

```python
feature
```

Out[51]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg
0	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111.0	21	27
1	3	122.0	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111.0	21	27
2	1	122.0	alfa-romero	gas	hatchback	rwd	front	65.5	52.4	ohcv	152	154.0	19	26
3	2	164.0	audi	gas	sedan	fwd	front	66.2	54.3	ohc	109	102.0	24	30
4	2	164.0	audi	gas	sedan	4wd	front	66.4	54.3	ohc	136	115.0	18	22
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	114.0	23	28
201	-1	95.0	volvo	gas	sedan	rwd	front	68.8	55.5	ohc	141	160.0	19	25
202	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohcv	173	134.0	18	23
203	-1	95.0	volvo	diesel	sedan	rwd	front	68.9	55.5	ohc	145	106.0	26	27
204	-1	95.0	volvo	gas	sedan	rwd	front	68.9	55.5	ohc	141	114.0	19	25

205 rows × 14 columns

In [52]:

```python
target
```

Out[52]:

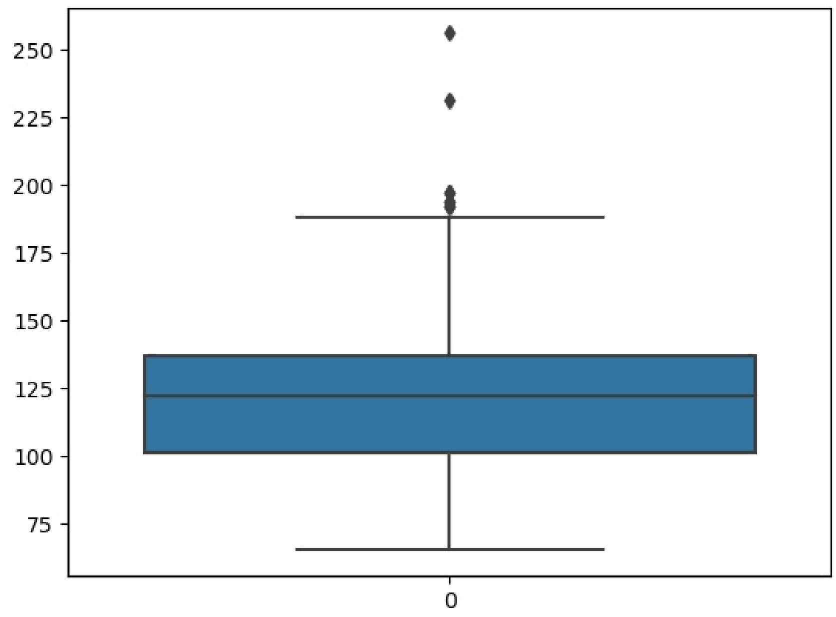
```
0      13495
1      16500
2      16500
3      13950
4      17450
       ...  
200    16845
201    19045
202    21485
203    22470
204    22625
Name: price, Length: 205, dtype: int64
```

handling outliers

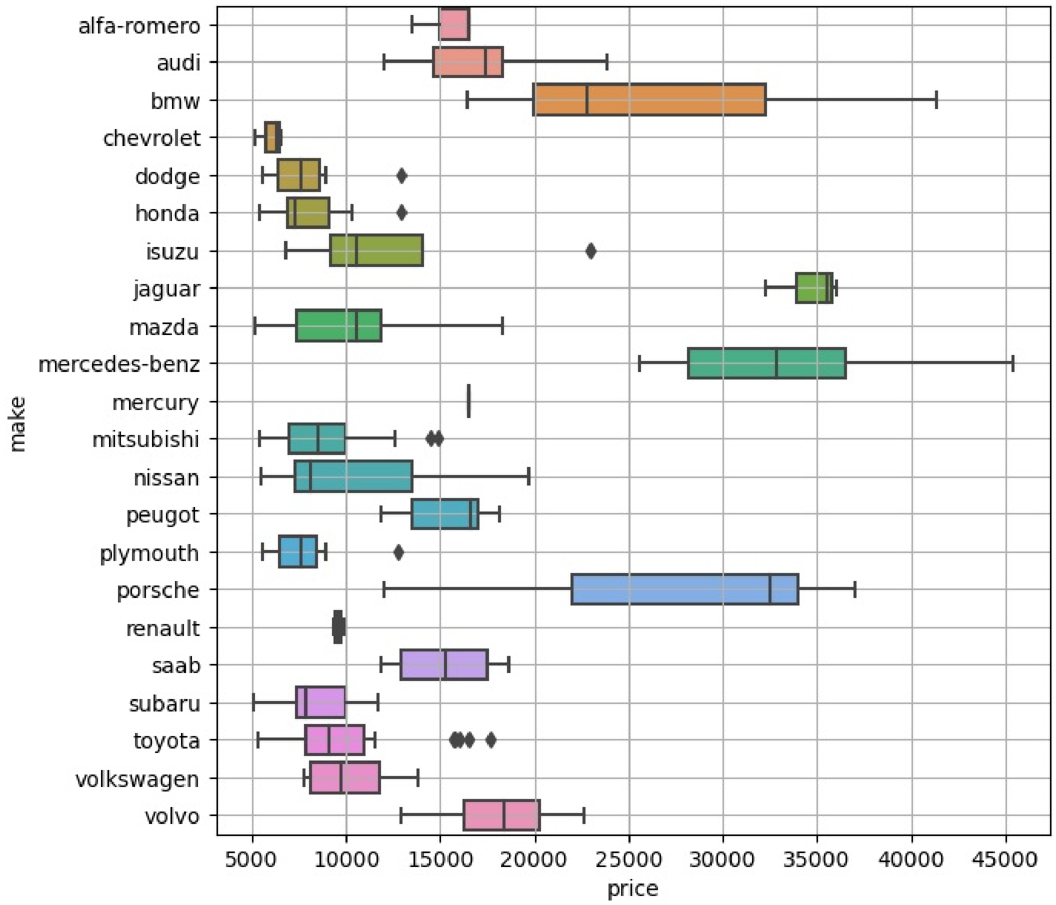
In [53]:

```python
sns.boxplot(data=feature["normalized-losses"])
```

Out[53]: `<AxesSubplot:>`



In [54]:
```
plt.figure(figsize=(7,7))
sns.boxplot(data=df,x=target,y="make")
plt.grid()
```



In [55]:
```
feature[(feature.make=="plymouth")&(target>10000)]
```

Out[55]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg
124	3	122.0	plymouth	gas	hatchback	rwd	front	66.3	50.2	ohc	156	145.0	19	24

In [92]:
```
feature[(feature.make=="isuzu")&(target>20000)]
```

Out[92]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg
45	0	122.0	isuzu	gas	sedan	fwd	front	63.6	52.0	ohc	90	70.0	38	43

In [56]:
```
feature.drop(124,axis=0,inplace=True)
```

In [93]:
```
feature.drop(45,axis=0,inplace=True)
```

In [94]:
```
df.describe()
```

Out[94]:

	symboling	normalized-losses	width	height	engine-size	horsepower	city-mpg	highway-mpg	price
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	0.834146	122.000000	65.907805	53.724878	126.907317	104.256158	25.219512	30.751220	13227.478049
std	1.245307	31.681008	2.145204	2.443522	41.642693	39.519211	6.542142	6.886443	7902.651615
min	-2.000000	65.000000	60.300000	47.800000	61.000000	48.000000	13.000000	16.000000	5118.000000
25%	0.000000	101.000000	64.100000	52.000000	97.000000	70.000000	19.000000	25.000000	7788.000000
50%	1.000000	122.000000	65.500000	54.100000	120.000000	95.000000	24.000000	30.000000	10345.000000
75%	2.000000	137.000000	66.900000	55.500000	141.000000	116.000000	30.000000	34.000000	16500.000000
max	3.000000	256.000000	72.300000	59.800000	326.000000	288.000000	49.000000	54.000000	45400.000000

handling skew

In [95]:
```
colname=feature.select_dtypes(["int64","float64"]).columns
```

In [96]:
```
colname
```

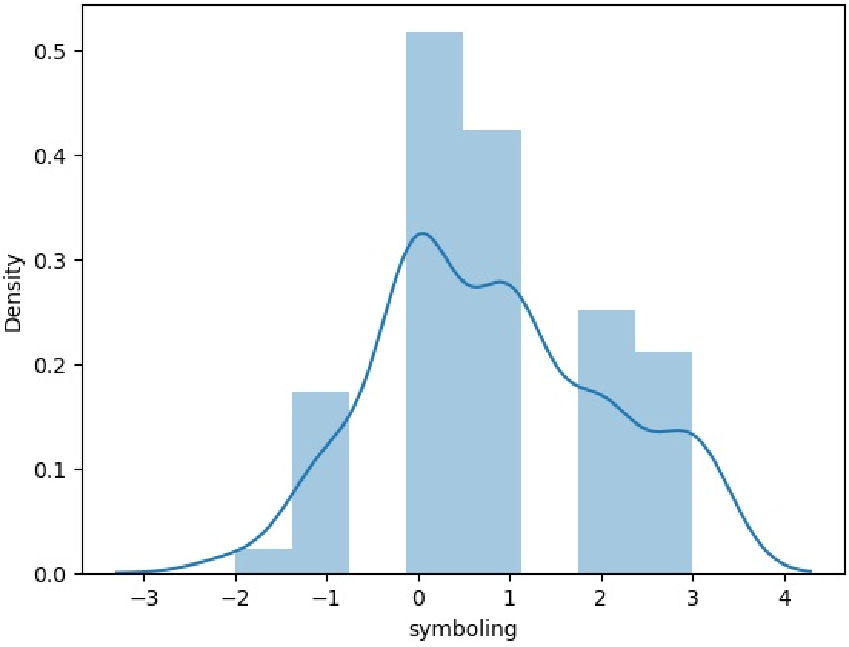
Out[96]:
```
Index(['symboling', 'normalized-losses', 'width', 'height', 'engine-size',
       'horsepower', 'city-mpg', 'highway-mpg'],
      dtype='object')
```

In [97]:
```
from scipy.stats import skew
skew(feature["normalized-losses"])
```

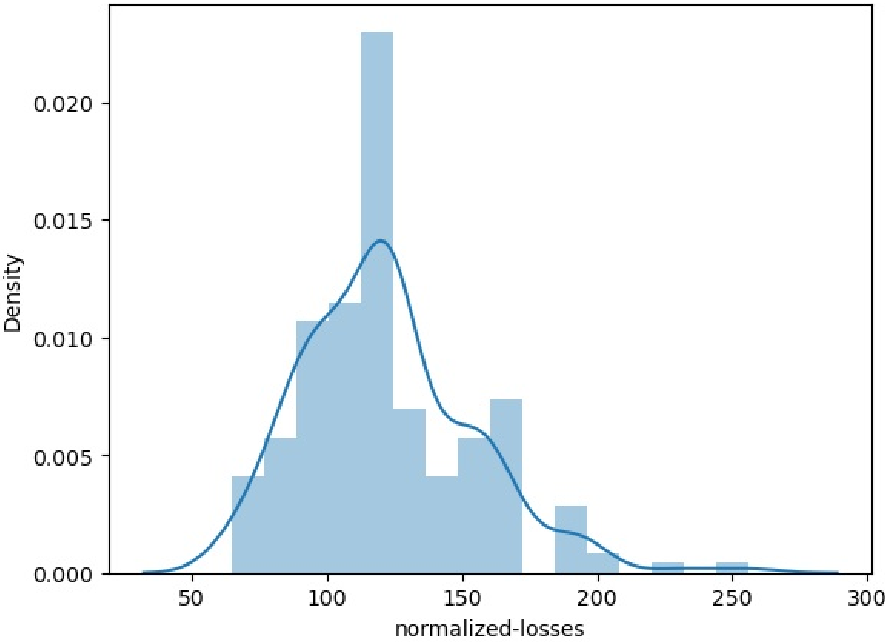
Out[97]:
```
0.8464627422841168
```

In [98]:
```
for i in feature[colname]:
    print(i)
    print(skew(feature[i]))
    plt.figure()
    sns.distplot(feature[i])
    plt.show()
```

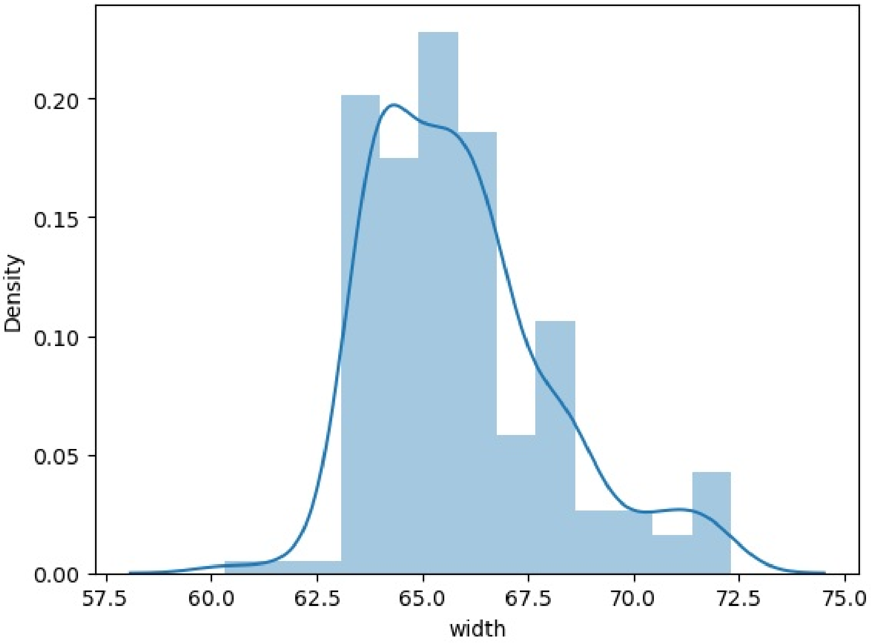
```
symboling
0.20132458820676538
```



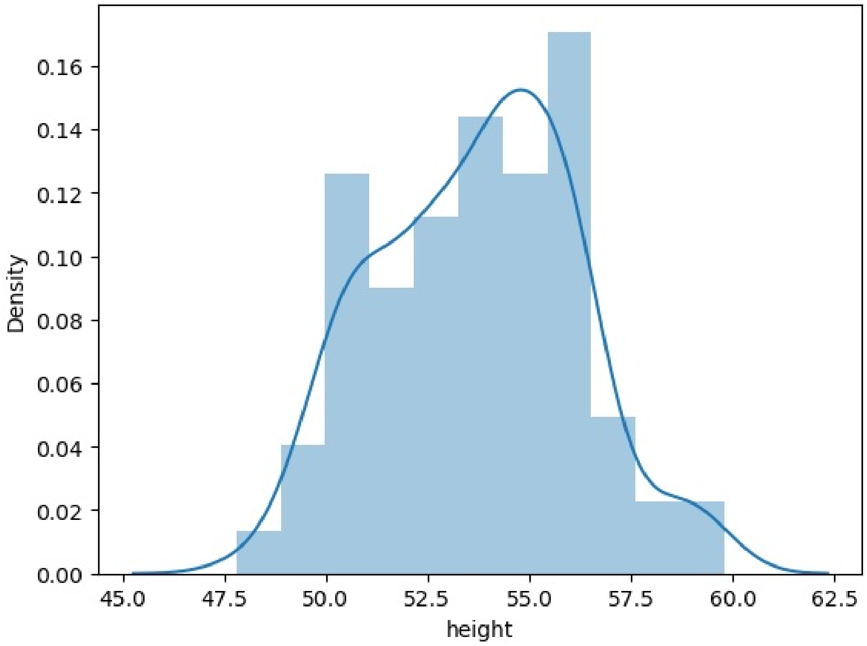
```
normalized-losses
0.8464627422841168
```



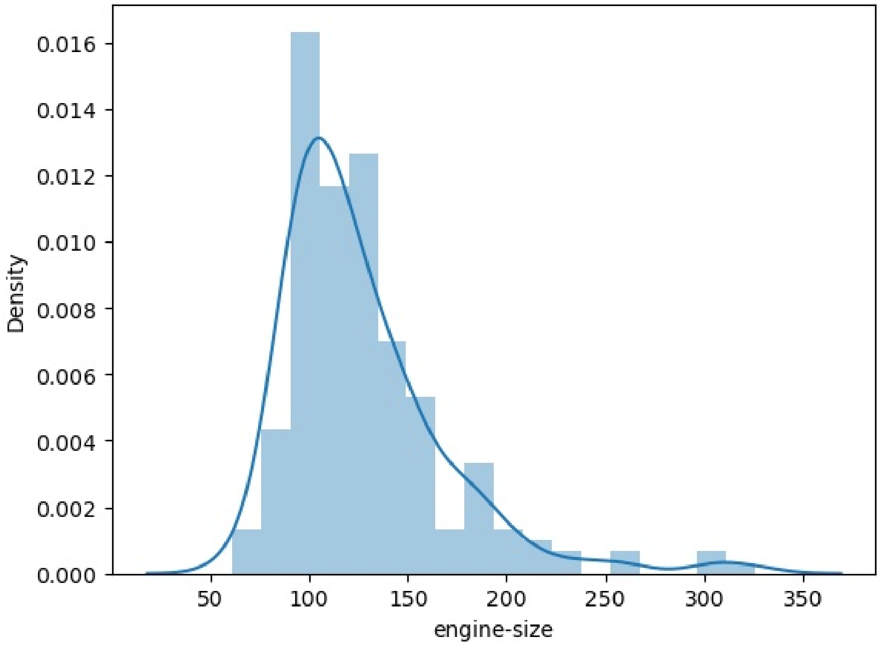
```
width
0.8931839293113768
```



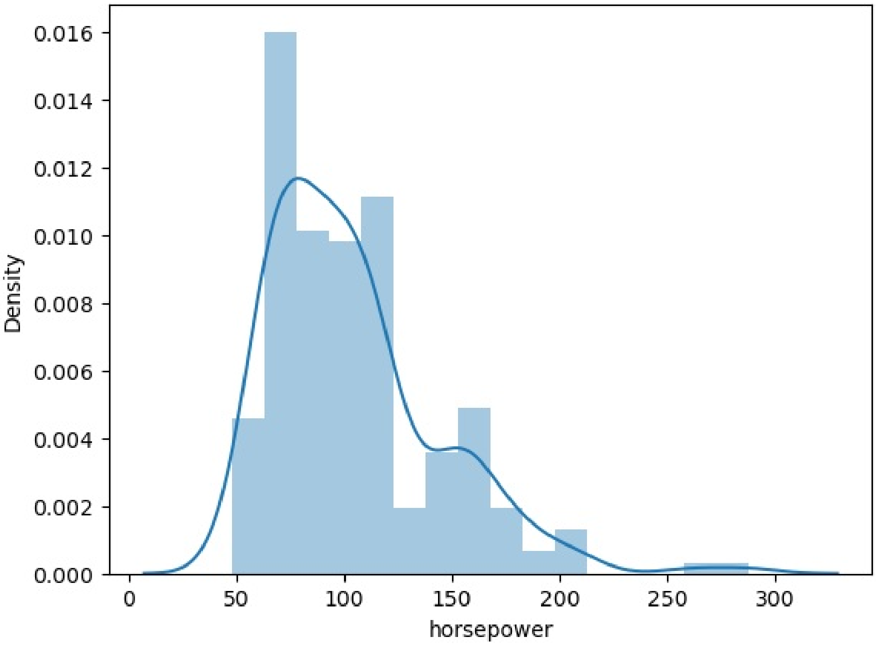
```
height
0.054074690888100505
```



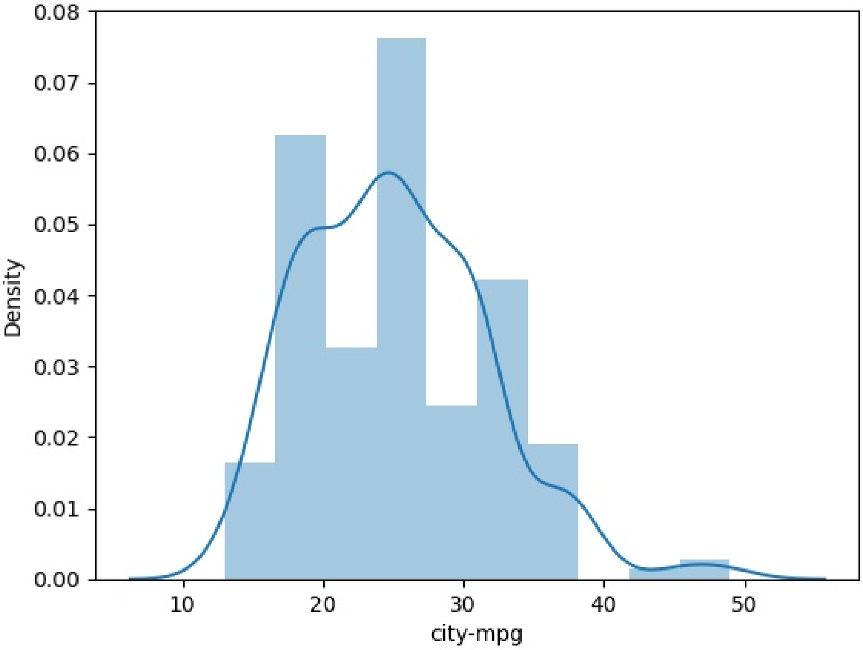
```
engine-size
1.93027551039049
```



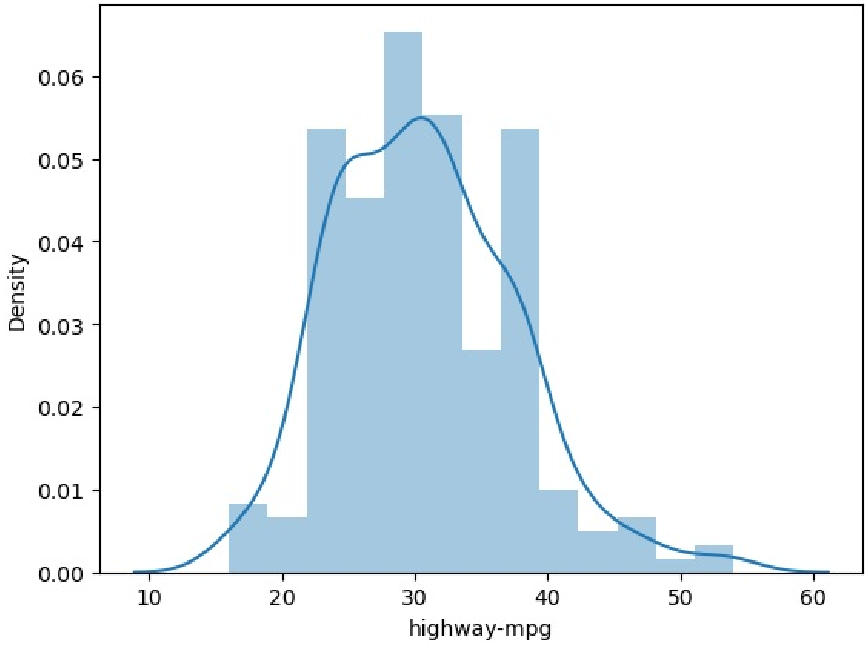
```
horsepower
1.3822861232508927
```



```
city-mpg
0.6676088844303997
```



```
highway-mpg
0.5455990153077631
```



In [99]:

```
pd.concat([feature,target],axis=1).corr()
```

Out[99]:

	symboling	normalized-losses	width	height	engine-size	horsepower	city-mpg	highway-mpg	price
symboling	1.000000	0.465705	-0.237408	-0.544638	-0.109046	0.068732	-0.029690	0.040839	-0.082101
normalized-losses	0.465705	1.000000	0.084436	-0.371162	0.111212	0.203811	-0.220834	-0.179626	0.133930
width	-0.237408	0.084436	1.000000	0.276598	0.734251	0.640614	-0.640208	-0.674984	0.729635
height	-0.544638	-0.371162	0.276598	1.000000	0.064270	-0.113501	-0.042298	-0.102096	0.137431
engine-size	-0.109046	0.111212	0.734251	0.064270	1.000000	0.809994	-0.652544	-0.676294	0.863317
horsepower	0.068732	0.203811	0.640614	-0.113501	0.809994	1.000000	-0.803888	-0.770754	0.756118
city-mpg	-0.029690	-0.220834	-0.640208	-0.042298	-0.652544	-0.803888	1.000000	0.970914	-0.675415
highway-mpg	0.040839	-0.179626	-0.674984	-0.102096	-0.676294	-0.770754	0.970914	1.000000	-0.697956
price	-0.082101	0.133930	0.729635	0.137431	0.863317	0.756118	-0.675415	-0.697956	1.000000

In [100…

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

Out[100]:

	symboling	normalized-losses	width	height	engine-size	horsepower	city-mpg	highway-mpg	price
symboling	1.000000	0.465705	-0.237408	-0.544638	-0.109046	0.068732	-0.029690	0.040839	-0.082101
normalized-losses	0.465705	1.000000	0.084436	-0.371162	0.111212	0.203811	-0.220834	-0.179626	0.133930
width	-0.237408	0.084436	1.000000	0.276598	0.734251	0.640614	-0.640208	-0.674984	0.729635
height	-0.544638	-0.371162	0.276598	1.000000	0.064270	-0.113501	-0.042298	-0.102096	0.137431
engine-size	-0.109046	0.111212	0.734251	0.064270	1.000000	0.809994	-0.652544	-0.676294	0.863317
horsepower	0.068732	0.203811	0.640614	-0.113501	0.809994	1.000000	-0.803888	-0.770754	0.756118
city-mpg	-0.029690	-0.220834	-0.640208	-0.042298	-0.652544	-0.803888	1.000000	0.970914	-0.675415
highway-mpg	0.040839	-0.179626	-0.674984	-0.102096	-0.676294	-0.770754	0.970914	1.000000	-0.697956
price	-0.082101	0.133930	0.729635	0.137431	0.863317	0.756118	-0.675415	-0.697956	1.000000

In [101…

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

Out[101]:

```
symboling           -0.082101
normalized-losses    0.133930
width                0.729635
height               0.137431
engine-size          0.863317
horsepower           0.756118
city-mpg            -0.675415
highway-mpg         -0.697956
price                1.000000
Name: price, dtype: float64
```

Encoding

OneHotEncoder

In [102…

```
from sklearn.preprocessing import OneHotEncoder
one=OneHotEncoder()
one.fit_transform(feature[["fuel-type"]]).toarray()
```

Out[102]:

```
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.]])
```

LabelEncoder

In [103…

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
le.fit_transform(target[])
```

```
  File "C:\Users\win10\AppData\Local\Temp\ipykernel_8732\3047889515.py", line 3
    le.fit_transform(target[])
                            ^
SyntaxError: invalid syntax
```

OrdinalEncoder

In [104…

```
from sklearn.preprocessing import OrdinalEncoder
on=OrdinalEncoder()
on.fit_transform(feature[["make"]])
```

Out[104]:

```
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.],
       [ 4.],
       [ 5.],
       [ 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.],
       [11.],
       [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.],
                [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.],
                [20.],
                [20.],
                [20.],
                [20.],
                [20.],
                [20.],
                [20.],
                [20.],
                [20.],
                [20.],
                [20.],
                [20.],
                [21.],
          [21.],
          [21.],
          [21.],
          [21.],
          [21.],
          [21.],
          [21.],
          [21.],
          [21.],
          [21.]])
```

In [105…

```
catcol=feature.select_dtypes("object").columns
```

In [106…

```
catcol
```

Out[106]:

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

In [107…

```
feature[catcol]=on.fit_transform(feature[catcol])
```

In [108…

```
feature[catcol]
```

Out[108]:

	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
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

In [109…

```
feature
```

Out[109]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg
0	3	122.0	0.0	1.0	0.0	2.0	0.0	64.1	48.8	0.0	130	111.0	21	27
1	3	122.0	0.0	1.0	0.0	2.0	0.0	64.1	48.8	0.0	130	111.0	21	27
2	1	122.0	0.0	1.0	2.0	2.0	0.0	65.5	52.4	5.0	152	154.0	19	26
3	2	164.0	1.0	1.0	3.0	1.0	0.0	66.2	54.3	3.0	109	102.0	24	30
4	2	164.0	1.0	1.0	3.0	0.0	0.0	66.4	54.3	3.0	136	115.0	18	22
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200	-1	95.0	21.0	1.0	3.0	2.0	0.0	68.9	55.5	3.0	141	114.0	23	28
201	-1	95.0	21.0	1.0	3.0	2.0	0.0	68.8	55.5	3.0	141	160.0	19	25
202	-1	95.0	21.0	1.0	3.0	2.0	0.0	68.9	55.5	5.0	173	134.0	18	23
203	-1	95.0	21.0	0.0	3.0	2.0	0.0	68.9	55.5	3.0	145	106.0	26	27
204	-1	95.0	21.0	1.0	3.0	2.0	0.0	68.9	55.5	3.0	141	114.0	19	25

204 rows × 14 columns

Scaling

MinMaxScaler

In [110…

```
from sklearn.preprocessing import MinMaxScaler
ms=MinMaxScaler()
features=pd.DataFrame(feature)
```

In [111…

```
features
```

Out[111]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg
0	3	122.0	0.0	1.0	0.0	2.0	0.0	64.1	48.8	0.0	130	111.0	21	27
1	3	122.0	0.0	1.0	0.0	2.0	0.0	64.1	48.8	0.0	130	111.0	21	27
2	1	122.0	0.0	1.0	2.0	2.0	0.0	65.5	52.4	5.0	152	154.0	19	26
3	2	164.0	1.0	1.0	3.0	1.0	0.0	66.2	54.3	3.0	109	102.0	24	30
4	2	164.0	1.0	1.0	3.0	0.0	0.0	66.4	54.3	3.0	136	115.0	18	22
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200	-1	95.0	21.0	1.0	3.0	2.0	0.0	68.9	55.5	3.0	141	114.0	23	28
201	-1	95.0	21.0	1.0	3.0	2.0	0.0	68.8	55.5	3.0	141	160.0	19	25
202	-1	95.0	21.0	1.0	3.0	2.0	0.0	68.9	55.5	5.0	173	134.0	18	23
203	-1	95.0	21.0	0.0	3.0	2.0	0.0	68.9	55.5	3.0	145	106.0	26	27
204	-1	95.0	21.0	1.0	3.0	2.0	0.0	68.9	55.5	3.0	141	114.0	19	25

204 rows × 14 columns

StandardScaler

In [112…

```
#StandardScaler
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
features=pd.DataFrame(feature)
features
```

Out[112]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg
0	3	122.0	0.0	1.0	0.0	2.0	0.0	64.1	48.8	0.0	130	111.0	21	27
1	3	122.0	0.0	1.0	0.0	2.0	0.0	64.1	48.8	0.0	130	111.0	21	27
2	1	122.0	0.0	1.0	2.0	2.0	0.0	65.5	52.4	5.0	152	154.0	19	26
3	2	164.0	1.0	1.0	3.0	1.0	0.0	66.2	54.3	3.0	109	102.0	24	30
4	2	164.0	1.0	1.0	3.0	0.0	0.0	66.4	54.3	3.0	136	115.0	18	22
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
200	-1	95.0	21.0	1.0	3.0	2.0	0.0	68.9	55.5	3.0	141	114.0	23	28
201	-1	95.0	21.0	1.0	3.0	2.0	0.0	68.8	55.5	3.0	141	160.0	19	25
202	-1	95.0	21.0	1.0	3.0	2.0	0.0	68.9	55.5	5.0	173	134.0	18	23
203	-1	95.0	21.0	0.0	3.0	2.0	0.0	68.9	55.5	3.0	145	106.0	26	27
204	-1	95.0	21.0	1.0	3.0	2.0	0.0	68.9	55.5	3.0	141	114.0	19	25

204 rows × 14 columns

In [113…

```
features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 204 entries, 0 to 204
Data columns (total 14 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   symboling          204 non-null    int64  
 1   normalized-losses  204 non-null    float64
 2   make               204 non-null    float64
 3   fuel-type          204 non-null    float64
 4   body-style         204 non-null    float64
 5   drive-wheels       204 non-null    float64
 6   engine-location    204 non-null    float64
 7   width              204 non-null    float64
 8   height             204 non-null    float64
 9   engine-type        204 non-null    float64
 10  engine-size        204 non-null    int64  
 11  horsepower         204 non-null    float64
 12  city-mpg           204 non-null    int64  
 13  highway-mpg        204 non-null    int64  
dtypes: float64(10), int64(4)
memory usage: 32.0 KB
```

In [114…

```
target.info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 205 entries, 0 to 204
Series name: price
Non-Null Count  Dtype
--------------  -----
205 non-null    int64
dtypes: int64(1)
memory usage: 1.7 KB
```

In [115…

```
target.drop(41,inplace=True,axis=0)
```

LinearRegression

In [116…

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(feature,target,test_size=0.3,random_state=1)
```

In [117…

```
xtrain
```

Out[117]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	highway-mpg
184	2	94.0	20.0	0.0	3.0	1.0	0.0	65.5	55.7	3.0	97	52.0	37	46
162	0	91.0	19.0	1.0	3.0	1.0	0.0	64.4	52.8	3.0	98	70.0	28	34
196	-2	103.0	21.0	1.0	3.0	2.0	0.0	67.2	56.2	3.0	141	114.0	24	28
185	2	94.0	20.0	1.0	3.0	1.0	0.0	65.5	55.7	3.0	109	85.0	27	34
74	1	122.0	9.0	1.0	1.0	2.0	0.0	72.0	55.4	5.0	304	184.0	14	16
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
134	3	150.0	17.0	1.0	2.0	1.0	0.0	66.5	56.1	3.0	121	110.0	21	28
138	2	83.0	18.0	1.0	2.0	1.0	0.0	63.4	53.7	4.0	97	69.0	31	36
73	0	122.0	9.0	1.0	3.0	2.0	0.0	71.7	56.7	5.0	308	184.0	14	16
141	0	102.0	18.0	1.0	3.0	1.0	0.0	65.4	52.5	4.0	108	82.0	32	37
37	0	106.0	5.0	1.0	2.0	1.0	0.0	65.2	53.3	3.0	110	86.0	27	33

142 rows × 14 columns

In [118…

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

In [88]:

```
ypred
```

Out[88]:

```
array([33723.79199967, 17813.72484582,  1111.7034949 , 10493.54241751,
       15598.10117847,  7942.04111118, 15527.68489101,  9768.70520706,
        7986.58857529, 11858.54250341,  7654.61642141,  5256.75556855,
        9517.98032949, 26299.57029173, 14611.14389038,  7416.59532149,
        9814.59800974,  7178.91988938, 14949.02045528, 17836.16195965,
        9416.22816417, 10995.02099718,  8199.20140486, 18327.80611898,
        3830.49400421, 14881.89927688, 11699.1736512 , 22556.39027694,
        6662.56216259, 23008.33399544, 15279.24567449, 14012.78553056,
        9333.71152339,  8365.12223517,  9657.04052712,  7206.69326333,
        8531.30729709, 16098.96083142, 16224.37658657, 14463.53706273,
       13566.99353089,  6889.45568261, 14478.63307949, 26838.30636138,
        7289.08532519,  7057.91845822, 13977.02024501,  5970.8357151 ,
       22287.13404067, 10390.66372614, 12632.67130563,  6617.32901998,
        8199.20140486, 12029.64660674, 20818.41140493, 11390.28738018,
        5562.05605774, 13416.87610196,  7187.15884191, 25847.96210417,
        8995.91995118,  9333.71152339])
```

In [ ]:

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