In [1]:
```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```python
df = pd.read_csv("C:\\Users\\HP\\Downloads\\cars.csv")
df.head()
```

Out[2]:

	symboling	normalized-losses	make	fuel-type	body-style	drive-wheels	engine-location	width	height	engine-type	engine-size	horsepower	city-mpg	hig
0	3	?	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111	21	
1	3	?	alfa-romero	gas	convertible	rwd	front	64.1	48.8	dohc	130	111	21	
2	1	?	alfa-romero	gas	hatchback	rwd	front	65.5	52.4	ohcv	152	154	19	
3	2	164	audi	gas	sedan	fwd	front	66.2	54.3	ohc	109	102	24	
4	2	164	audi	gas	sedan	4wd	front	66.4	54.3	ohc	136	115	18	

In [3]:
```python
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 [4]:
```python
df['normalized-losses'].value_counts()
```

Out[4]:
```
?      41
161    11
91      8
150     7
128     6
134     6
104     6
102     5
168     5
74      5
95      5
103     5
65      5
94      5
85      5
122     4
93      4
106     4
148     4
118     4
154     3
83      3
115     3
125     3
137     3
101     3
113     2
119     2
197     2
81      2
194     2
129     2
192     2
87      2
158     2
108     2
188     2
145     2
153     2
164     2
89      2
110     2
107     1
256     1
90      1
231     1
142     1
77      1
98      1
121     1
78      1
186     1
Name: normalized-losses, dtype: int64
```

In [5]:
```python
df['horsepower'].value_counts()
```

Out[5]:
```
68     19
70     11
69     10
116     9
110     8
95      7
62      6
114     6
101     6
88      6
160     6
82      5
76      5
84      5
102     5
145     5
97      5
123     4
111     4
92      4
86      4
121     3
207     3
85      3
90      3
182     3
73      3
152     3
161     2
52      2
184     2
56      2
?       2
94      2
162     2
112     2
156     2
100     2
176     2
155     2
48      1
140     1
262     1
115     1
175     1
135     1
78      1
154     1
142     1
58      1
60      1
120     1
64      1
106     1
288     1
200     1
134     1
143     1
55      1
72      1
Name: horsepower, dtype: int64
```

In [6]:
```python
df['normalized-losses'].replace("?",np.nan,inplace=True)
df['horsepower'].replace("?",np.nan,inplace=True)

df['normalized-losses']=df['normalized-losses'].astype('float')
df['horsepower'] = df['horsepower'].astype('float')

nmean = df['normalized-losses'].mean()
hmean = df['horsepower'].mean()

df['normalized-losses'].fillna(nmean,inplace=True)
df['horsepower'].fillna(hmean,inplace=True)
```

In [7]:
```python
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
```

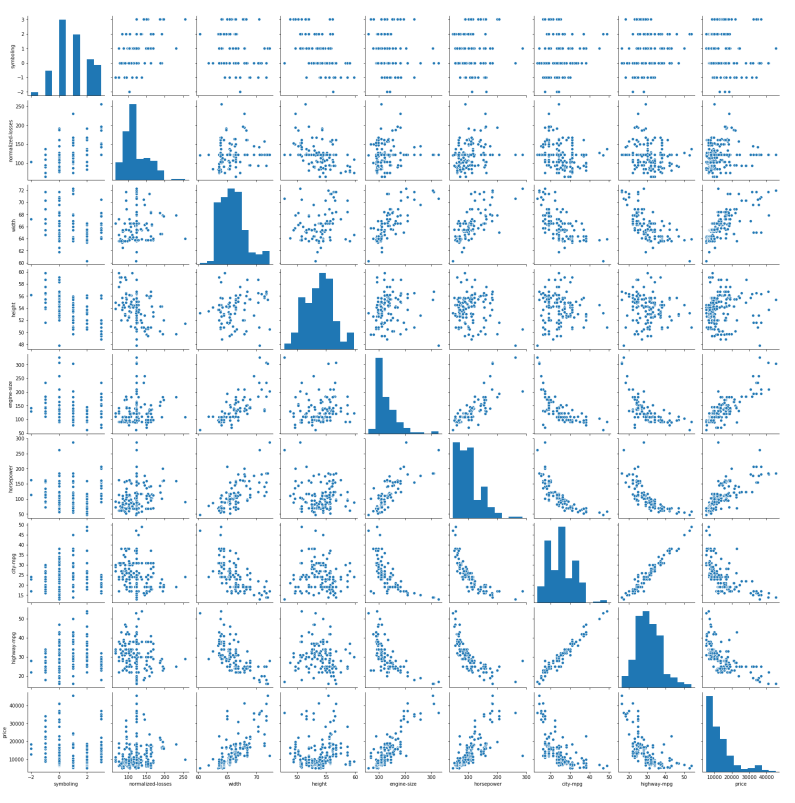
In [8]:
```python
df.describe()
```

Out[8]:

	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

In [9]:
```python
sns.pairplot(df)
```

Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x29d950ca0d0>
```



In [10]:
```python
df_num = df.select_dtypes(['int64','float64'])
df_cat = df.select_dtypes('object')
```

In [11]:
```python
df_cat
```

Out[11]:

	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

205 rows × 6 columns

In [12]:
```python
df_num
```

Out[12]:

	symboling	normalized-losses	width	height	engine-size	horsepower	city-mpg	highway-mpg	price
0	3	122.0	64.1	48.8	130	111.0	21	27	13495
1	3	122.0	64.1	48.8	130	111.0	21	27	16500
2	1	122.0	65.5	52.4	152	154.0	19	26	16500
3	2	164.0	66.2	54.3	109	102.0	24	30	13950
4	2	164.0	66.4	54.3	136	115.0	18	22	17450
...	...	...	...	...	...	...	...	...	...
200	-1	95.0	68.9	55.5	141	114.0	23	28	16845
201	-1	95.0	68.8	55.5	141	160.0	19	25	19045
202	-1	95.0	68.9	55.5	173	134.0	18	23	21485
203	-1	95.0	68.9	55.5	145	106.0	26	27	22470
204	-1	95.0	68.9	55.5	141	114.0	19	25	22625

205 rows × 9 columns

In [13]:
```python
from sklearn.preprocessing import LabelEncoder
for col in df_cat:
    le = LabelEncoder()
    df_cat[col] = le.fit_transform(df_cat[col])
```

In [14]:
```python
df_cat
```

Out[14]:

	make	fuel-type	body-style	drive-wheels	engine-location	engine-type
0	0	1	0	2	0	0
1	0	1	0	2	0	0
2	0	1	2	2	0	5
3	1	1	3	1	0	3
4	1	1	3	0	0	3
...	...	...	...	...	...	...
200	21	1	3	2	0	3
201	21	1	3	2	0	3
202	21	1	3	2	0	5
203	21	0	3	2	0	3
204	21	1	3	2	0	3

205 rows × 6 columns

In [15]:
```python
df_new = pd.concat([df_cat, df_num], axis=1)
```

In [16]:
```python
df_new.head()
```

Out[16]:

	make	fuel-type	body-style	drive-wheels	engine-location	engine-type	symboling	normalized-losses	width	height	engine-size	horsepower	city-mpg	highway-mpg
0	0	1	0	2	0	0	3	122.0	64.1	48.8	130	111.0	21	27
1	0	1	0	2	0	0	3	122.0	64.1	48.8	130	111.0	21	27
2	0	1	2	2	0	5	1	122.0	65.5	52.4	152	154.0	19	26
3	1	1	3	1	0	3	2	164.0	66.2	54.3	109	102.0	24	30
4	1	1	3	0	0	3	2	164.0	66.4	54.3	136	115.0	18	22

In [17]:
```python
x = df_new.iloc[:,:-1]
y = df_new.iloc[:,-1]
```

In [18]:
```python
from sklearn.model_selection import train_test_split

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

In [19]:
```python
from xgboost import XGBRegressor

xgb = XGBRegressor()

xgb.fit(xtrain, ytrain)

ypred = xgb.predict(xtest)
```

In [20]:
```python
from sklearn.metrics import mean_absolute_error as mae, mean_squared_error as mse, r2_score
print(f"MAE-: {mae(ytest, ypred)}")
print(f"MSE-: {mse(ytest, ypred)}")
print(f"RMSE-: {np.sqrt(mse(ytest, ypred))}")
print(f"R Squared-: {r2_score(ytest, ypred)}")
```

```
MAE-: 2087.5291472404233
MSE-: 11211227.00085248
RMSE-: 3348.3170400743834
R Squared-: 0.8139518932966816
```

In [24]:
```python
from sklearn.model_selection import GridSearchCV

xgb1 = XGBRegressor(random_state=1)
params= {
 'max_depth':range(3,10,2),
 'min_child_weight':range(1,6,2),
 'reg_alpha':[0, 0.001, 0.005, 0.01, 0.05],
 'learning_rate': [0.01, 0.1],
}
xgb_grid = GridSearchCV(xgb1,params,cv = 5,verbose=True)
xgb_grid.fit(xtrain,ytrain)
ypred=xgb_grid.predict(xtest)
```

```
Fitting 5 folds for each of 120 candidates, totalling 600 fits
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 600 out of 600 | elapsed:  1.1min finished
```

In [25]:
```python
print(f"MAE-: {mae(ytest, ypred)}")
print(f"MSE-: {mse(ytest, ypred)}")
print(f"RMSE-: {np.sqrt(mse(ytest, ypred))}")
print(f"R Squared-: {r2_score(ytest, ypred)}")
```

```
MAE-: 1942.9898878528227
MSE-: 8040316.062336145
RMSE-: 2835.545108499624
R Squared-: 0.8665725365671237
```