Machine Learning Project-02 Linear Regression

Car Price Predication

Problem Description

A Chinese automobile company aspires to enter the US market by setting up their manufacturing unit there and producing cars locally to give competition to their US and European counterparts. They have contracted an automobile consulting company to understand the factors on which the pricing of cars depends. Specifically, they want to understand the factors affecting the pricing of cars in the American market, since those may be very different from the Chinese market. Essentially, the company wants to know:

1. Which variables are significant in predicting the price of a car.
2. How well those variables describe the price of a car.

Based on various market surveys, the consulting firm has gathered a large dataset of different types of cars across the American market.

Business Goal

You are required to model the price of cars with the available independent variables. It will be used by the management to understand how exactly the prices vary with the independent variables. They can accordingly manipulate the design of the cars, the business strategy etc. to meet certain price levels. Further, the model will be a good way for the management to understand the pricing dynamics of a new market.

Dataset: https://drive.google.com/file/d/1FHmYNLs9v0Enc-UExEMpitOFGsWvB2dP/view

In [8]:
```
# import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [9]:
```
# import dataset
df = pd.read_csv('CarPrice_Assignment.csv')
```

1. Explore the data

In [10]:
```
df.shape
```

Out[10]:
```
(205, 26)
```

In [11]:
```
df.head()
```

Out[11]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fuelsy
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...	130	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...	130	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...	152	
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...	109	
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...	136	

5 rows × 26 columns

In [12]:
```
df.columns
```

Out[12]:
```
Index(['car_ID', 'symboling', 'CarName', 'fueltype', 'aspiration',
       'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'wheelbase',
       'carlength', 'carwidth', 'carheight', 'curbweight', 'enginetype',
       'cylindernumber', 'enginesize', 'fuelsystem', 'boreratio', 'stroke',
       'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'highwaympg',
       'price'],
      dtype='object')
```

In [13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   car_ID            205 non-null    int64  
 1   symboling         205 non-null    int64  
 2   CarName           205 non-null    object 
 3   fueltype          205 non-null    object 
 4   aspiration        205 non-null    object 
 5   doornumber        205 non-null    object 
 6   carbody           205 non-null    object 
 7   drivewheel        205 non-null    object 
 8   enginelocation    205 non-null    object 
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64  
 14  enginetype        205 non-null    object 
 15  cylindernumber    205 non-null    object 
 16  enginesize        205 non-null    int64  
 17  fuelsystem        205 non-null    object 
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64  
 22  peakrpm           205 non-null    int64  
 23  citympg           205 non-null    int64  
 24  highwaympg        205 non-null    int64  
 25  price             205 non-null    float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB
```

In [14]:
```
df['CarName'].unique()
```

Out[14]:
```
array(['alfa-romero giulia', 'alfa-romero stelvio',
       'alfa-romero Quadrifoglio', 'audi 100 ls', 'audi 100ls',
       'audi fox', 'audi 5000', 'audi 4000', 'audi 5000s (diesel)',
       'bmw 320i', 'bmw x1', 'bmw x3', 'bmw z4', 'bmw x4', 'bmw x5',
       'chevrolet impala', 'chevrolet monte carlo', 'chevrolet vega 2300',
       'dodge rampage', 'dodge challenger se', 'dodge d200',
       'dodge monaco (sw)', 'dodge colt hardtop', 'dodge colt (sw)',
       'dodge coronet custom', 'dodge dart custom',
       'dodge coronet custom (sw)', 'honda civic', 'honda civic cvcc',
       'honda accord cvcc', 'honda accord lx', 'honda civic 1500 gl',
       'honda accord', 'honda civic 1300', 'honda prelude',
       'honda civic (auto)', 'isuzu MU-X', 'isuzu D-Max ',
       'isuzu D-Max V-Cross', 'jaguar xj', 'jaguar xf', 'jaguar xk',
       'maxda rx3', 'maxda glc deluxe', 'mazda rx2 coupe', 'mazda rx-4',
       'mazda glc deluxe', 'mazda 626', 'mazda glc', 'mazda rx-7 gs',
       'mazda glc 4', 'mazda glc custom l', 'mazda glc custom',
       'buick electra 225 custom', 'buick century luxus (sw)',
       'buick century', 'buick skyhawk', 'buick opel isuzu deluxe',
       'buick skylark', 'buick century special',
       'buick regal sport coupe (turbo)', 'mercury cougar',
       'mitsubishi mirage', 'mitsubishi lancer', 'mitsubishi outlander',
       'mitsubishi g4', 'mitsubishi mirage g4', 'mitsubishi montero',
       'mitsubishi pajero', 'Nissan versa', 'nissan gt-r', 'nissan rogue',
       'nissan latio', 'nissan titan', 'nissan leaf', 'nissan juke',
       'nissan note', 'nissan clipper', 'nissan nv200', 'nissan dayz',
       'nissan fuga', 'nissan otti', 'nissan teana', 'nissan kicks',
       'peugeot 504', 'peugeot 304', 'peugeot 504 (sw)', 'peugeot 604sl',
       'peugeot 505s turbo diesel', 'plymouth fury iii',
       'plymouth cricket', 'plymouth satellite custom (sw)',
       'plymouth fury gran sedan', 'plymouth valiant', 'plymouth duster',
       'porsche macan', 'porcshce panamera', 'porsche cayenne',
       'porsche boxter', 'renault 12tl', 'renault 5 gtl', 'saab 99e',
       'saab 99le', 'saab 99gle', 'subaru', 'subaru dl', 'subaru brz',
       'subaru baja', 'subaru r1', 'subaru r2', 'subaru trezia',
       'subaru tribeca', 'toyota corona mark ii', 'toyota corona',
       'toyota corolla 1200', 'toyota corona hardtop',
       'toyota corolla 1600 (sw)', 'toyota carina', 'toyota mark ii',
       'toyota corolla', 'toyota corolla liftback',
       'toyota celica gt liftback', 'toyota corolla tercel',
       'toyota corona liftback', 'toyota starlet', 'toyota tercel',
       'toyota cressida', 'toyota celica gt', 'toyouta tercel',
       'vokswagen rabbit', 'volkswagen 1131 deluxe sedan',
       'volkswagen model 111', 'volkswagen type 3', 'volkswagen 411 (sw)',
       'volkswagen super beetle', 'volkswagen dasher', 'vw dasher',
       'vw rabbit', 'volkswagen rabbit', 'volkswagen rabbit custom',
```

```
       'volvo 145e (sw)', 'volvo 144ea', 'volvo 244dl', 'volvo 245',
       'volvo 264gl', 'volvo diesel', 'volvo 246'], dtype=object)
```

In [15]:
```
df['fueltype'].unique()
```

Out[15]:
```
array(['gas', 'diesel'], dtype=object)
```

In [16]:
```
df['aspiration'].unique()
```

Out[16]:
```
array(['std', 'turbo'], dtype=object)
```

In [17]:
```
df['doornumber'].unique()
```

Out[17]:
```
array(['two', 'four'], dtype=object)
```

In [18]:
```
#replace the text values with numbers
df['doornumber'].replace('two',2,inplace=True)
df['doornumber'].replace('four',4,inplace=True)
df['doornumber'].unique()
```

Out[18]:
```
array([2, 4], dtype=int64)
```

In [19]:
```
df['carbody'].unique()
```

Out[19]:
```
array(['convertible', 'hatchback', 'sedan', 'wagon', 'hardtop'],
      dtype=object)
```

In [20]:
```
df['drivewheel'].unique()
```

Out[20]:
```
array(['rwd', 'fwd', '4wd'], dtype=object)
```

In [21]:
```
df['enginelocation'].unique()
```

Out[21]:
```
array(['front', 'rear'], dtype=object)
```

In [22]:
```
df['wheelbase'].unique()
```

Out[22]:
```
array([ 88.6,  94.5,  99.8,  99.4, 105.8,  99.5, 101.2, 103.5, 110. ,
        88.4,  93.7, 103.3,  95.9,  86.6,  96.5,  94.3,  96. , 113. ,
       102. ,  93.1,  95.3,  98.8, 104.9, 106.7, 115.6,  96.6, 120.9,
       112. , 102.7,  93. ,  96.3,  95.1,  97.2, 100.4,  91.3,  99.2,
       107.9, 114.2, 108. ,  89.5,  98.4,  96.1,  99.1,  93.3,  97. ,
        96.9,  95.7, 102.4, 102.9, 104.5,  97.3, 104.3, 109.1])
```

In [23]:
```
df['carlength'].unique()
```

Out[23]:
```
array([168.8, 171.2, 176.6, 177.3, 192.7, 178.2, 176.8, 189. , 193.8,
       197. , 141.1, 155.9, 158.8, 157.3, 174.6, 173.2, 144.6, 150. ,
       163.4, 157.1, 167.5, 175.4, 169.1, 170.7, 172.6, 199.6, 191.7,
       159.1, 166.8, 169. , 177.8, 175. , 190.9, 187.5, 202.6, 180.3,
       208.1, 199.2, 178.4, 173. , 172.4, 165.3, 170.2, 165.6, 162.4,
       173.4, 181.7, 184.6, 178.5, 186.7, 198.9, 167.3, 168.9, 175.7,
       181.5, 186.6, 156.9, 157.9, 172. , 173.5, 173.6, 158.7, 169.7,
       166.3, 168.7, 176.2, 175.6, 183.5, 187.8, 171.7, 159.3, 165.7,
       180.2, 183.1, 188.8])
```

In [24]:
```
df['carwidth'].unique()
```

Out[24]:
```
array([64.1, 65.5, 66.2, 66.4, 66.3, 71.4, 67.9, 64.8, 66.9, 70.9, 60.3,
       63.6, 63.8, 64.6, 63.9, 64. , 65.2, 62.5, 66. , 61.8, 69.6, 70.6,
       64.2, 65.7, 66.5, 66.1, 70.3, 71.7, 70.5, 72. , 68. , 64.4, 65.4,
       68.4, 68.3, 65. , 72.3, 66.6, 63.4, 65.6, 67.7, 67.2, 68.9, 68.8])
```

In [25]:
```
df['carheight'].unique()
```

Out[25]:
```
array([48.8, 52.4, 54.3, 53.1, 55.7, 55.9, 52. , 53.7, 56.3, 53.2, 50.8,
       50.6, 59.8, 50.2, 52.6, 54.5, 58.3, 53.3, 54.1, 51. , 53.5, 51.4,
       52.8, 47.8, 49.6, 55.5, 54.4, 56.5, 58.7, 54.9, 56.7, 55.4, 54.8,
       49.4, 51.6, 54.7, 55.1, 56.1, 49.7, 56. , 50.5, 55.2, 52.5, 53. ,
       59.1, 53.9, 55.6, 56.2, 57.5])
```

In [26]:
```
# create a function for ease of handle
def unique(x):
    return df[x].unique()
```

In [27]:
```
unique('curbweight')
```

Out[27]:
```
array([2548, 2823, 2337, 2824, 2507, 2844, 2954, 3086, 3053, 2395, 2710,
       2765, 3055, 3230, 3380, 3505, 1488, 1874, 1909, 1876, 2128, 1967,
       1989, 2191, 2535, 2811, 1713, 1819, 1837, 1940, 1956, 2010, 2024,
       2236, 2289, 2304, 2372, 2465, 2293, 2734, 4066, 3950, 1890, 1900,
       1905, 1945, 1950, 2380, 2385, 2500, 2410, 2443, 2425, 2670, 2700,
       3515, 3750, 3495, 3770, 3740, 3685, 3900, 3715, 2910, 1918, 1944,
       2004, 2145, 2370, 2328, 2833, 2921, 2926, 2365, 2405, 2403, 1889,
       2017, 1938, 1951, 2028, 1971, 2037, 2008, 2324, 2302, 3095, 3296,
       3060, 3071, 3139, 3020, 3197, 3430, 3075, 3252, 3285, 3485, 3130,
       2818, 2778, 2756, 2800, 3366, 2579, 2460, 2658, 2695, 2707, 2758,
       2808, 2847, 2050, 2120, 2240, 2190, 2340, 2510, 2290, 2455, 2420,
       2650, 1985, 2040, 2015, 2280, 3110, 2081, 2109, 2275, 2094, 2122,
       2140, 2169, 2204, 2265, 2300, 2540, 2536, 2551, 2679, 2714, 2975,
       2326, 2480, 2414, 2458, 2976, 3016, 3131, 3151, 2261, 2209, 2264,
       2212, 2319, 2254, 2221, 2661, 2563, 2912, 3034, 2935, 3042, 3045,
       3157, 2952, 3049, 3012, 3217, 3062], dtype=int64)
```

In [28]:
```
unique('enginetype')
```

Out[28]:
```
array(['dohc', 'ohcv', 'ohc', 'l', 'rotor', 'ohcf', 'dohcv'], dtype=object)
```

In [29]:
```
unique('cylindernumber')
```

Out[29]:
```
array(['four', 'six', 'five', 'three', 'twelve', 'two', 'eight'],
      dtype=object)
```

In [30]:
```
#replace the text values with numbers
df['cylindernumber'].replace('four',4,inplace=True)
df['cylindernumber'].replace('six',6,inplace=True)
df['cylindernumber'].replace('five',5,inplace=True)
df['cylindernumber'].replace('three',3,inplace=True)
df['cylindernumber'].replace('twelve',12,inplace=True)
df['cylindernumber'].replace('two',2,inplace=True)
df['cylindernumber'].replace('eight',8,inplace=True)
df['cylindernumber'].unique()
```

Out[30]:
```
array([ 4,  6,  5,  3, 12,  2,  8], dtype=int64)
```

In [31]:
```
unique('enginesize')
```

Out[31]:
```
array([130, 152, 109, 136, 131, 108, 164, 209,  61,  90,  98, 122, 156,
        92,  79, 110, 111, 119, 258, 326,  91,  70,  80, 140, 134, 183,
       234, 308, 304,  97, 103, 120, 181, 151, 194, 203, 132, 121, 146,
       171, 161, 141, 173, 145], dtype=int64)
```

In [32]:
```
unique('fuelsystem')
```

Out[32]:
```
array(['mpfi', '2bbl', 'mfi', '1bbl', 'spfi', '4bbl', 'idi', 'spdi'],
      dtype=object)
```

In [33]:
```
unique('boreratio')
```

Out[33]:
```
array([3.47, 2.68, 3.19, 3.13, 3.5 , 3.31, 3.62, 2.91, 3.03, 2.97, 3.34,
       3.6 , 2.92, 3.15, 3.43, 3.63, 3.54, 3.08, 3.33, 3.39, 3.76, 3.58,
       3.46, 3.8 , 3.78, 3.17, 3.35, 3.59, 2.99, 3.7 , 3.61, 3.94, 3.74,
       2.54, 3.05, 3.27, 3.24, 3.01])
```

In [34]:
```
unique('stroke')
```

Out[34]:
```
array([2.68 , 3.47 , 3.4  , 2.8  , 3.19 , 3.39 , 3.03 , 3.11 , 3.23 ,
       3.46 , 3.9  , 3.41 , 3.07 , 3.58 , 4.17 , 2.76 , 3.15 , 3.255,
       3.16 , 3.64 , 3.1  , 3.35 , 3.12 , 3.86 , 3.29 , 3.27 , 3.52 ,
       2.19 , 3.21 , 2.9  , 2.07 , 2.36 , 2.64 , 3.08 , 3.5  , 3.54 ,
       2.87 ])
```

In [35]:
```
unique('compressionratio')
```

Out[35]:
```
array([ 9.  , 10.  ,  8.  ,  8.5 ,  8.3 ,  7.  ,  8.8 ,  9.5 ,  9.6 ,
        9.41,  9.4 ,  7.6 ,  9.2 , 10.1 ,  9.1 ,  8.1 , 11.5 ,  8.6 ,
       22.7 , 22.  , 21.5 ,  7.5 , 21.9 ,  7.8 ,  8.4 , 21.  ,  8.7 ,
        9.31,  9.3 ,  7.7 , 22.5 , 23.  ])
```

In [36]:
```
unique('horsepower')
```

Out[36]:
```
array([111, 154, 102, 115, 110, 140, 160, 101, 121, 182,  48,  70,  68,
        88, 145,  58,  76,  60,  86, 100,  78,  90, 176, 262, 135,  84,
        64, 120,  72, 123, 155, 184, 175, 116,  69,  55,  97, 152, 200,
        95, 142, 143, 207, 288,  73,  82,  94,  62,  56, 112,  92, 161,
       156,  52,  85, 114, 162, 134, 106], dtype=int64)
```

In [37]:
```
unique('peakrpm')
```

Out[37]:
```
array([5000, 5500, 5800, 4250, 5400, 5100, 4800, 6000, 4750, 4650, 4200,
       4350, 4500, 5200, 4150, 5600, 5900, 5750, 5250, 4900, 4400, 6600,
       5300], dtype=int64)
```

In [38]:
```
unique('citympg')
```

Out[38]:
```
array([21, 19, 24, 18, 17, 16, 23, 20, 15, 47, 38, 37, 31, 49, 30, 27, 25,
       13, 26, 36, 22, 14, 45, 28, 32, 35, 34, 29, 33], dtype=int64)
```

In [39]:
```
unique('highwaympg')
```

Out[39]:
```
array([27, 26, 30, 22, 25, 20, 29, 28, 53, 43, 41, 38, 24, 54, 42, 34, 33,
       31, 19, 17, 23, 32, 39, 18, 16, 37, 50, 36, 47, 46], dtype=int64)
```

In [40]:
```
unique('price')
```

Out[40]:
```
array([13495.   , 16500.   , 13950.   , 17450.   , 15250.   , 17710.   ,
       18920.   , 23875.   , 17859.167, 16430.   , 16925.   , 20970.   ,
       21105.   , 24565.   , 30760.   , 41315.   , 36880.   ,  5151.   ,
        6295.   ,  6575.   ,  5572.   ,  6377.   ,  7957.   ,  6229.   ,
        6692.   ,  7609.   ,  8558.   ,  8921.   , 12964.   ,  6479.   ,
        6855.   ,  5399.   ,  6529.   ,  7129.   ,  7295.   ,  7895.   ,
        9095.   ,  8845.   , 10295.   , 12945.   , 10345.   ,  6785.   ,
        8916.5  , 11048.   , 32250.   , 35550.   , 36000.   ,  5195.   ,
        6095.   ,  6795.   ,  6695.   ,  7395.   , 10945.   , 11845.   ,
       13645.   , 15645.   ,  8495.   , 10595.   , 10245.   , 10795.   ,
       11245.   , 18280.   , 18344.   , 25552.   , 28248.   , 28176.   ,
       31600.   , 34184.   , 35056.   , 40960.   , 45400.   , 16503.   ,
        5389.   ,  6189.   ,  6669.   ,  7689.   ,  9959.   ,  8499.   ,
       12629.   , 14869.   , 14489.   ,  6989.   ,  8189.   ,  9279.   ,
        5499.   ,  7099.   ,  6649.   ,  6849.   ,  7349.   ,  7299.   ,
        7799.   ,  7499.   ,  7999.   ,  8249.   ,  8949.   ,  9549.   ,
       13499.   , 14399.   , 17199.   , 19699.   , 18399.   , 11900.   ,
       13200.   , 12440.   , 13860.   , 15580.   , 16900.   , 16695.   ,
       17075.   , 16630.   , 17950.   , 18150.   , 12764.   , 22018.   ,
       32528.   , 34028.   , 37028.   , 31400.5  ,  9295.   ,  9895.   ,
       11850.   , 12170.   , 15040.   , 15510.   , 18620.   ,  5118.   ,
        7053.   ,  7603.   ,  7126.   ,  7775.   ,  9960.   ,  9233.   ,
       11259.   ,  7463.   , 10198.   ,  8013.   , 11694.   ,  5348.   ,
        6338.   ,  6488.   ,  6918.   ,  7898.   ,  8778.   ,  6938.   ,
        7198.   ,  7788.   ,  7738.   ,  8358.   ,  9258.   ,  8058.   ,
        8238.   ,  9298.   ,  9538.   ,  8449.   ,  9639.   ,  9989.   ,
       11199.   , 11549.   , 17669.   ,  8948.   , 10698.   ,  9988.   ,
       10898.   , 11248.   , 16558.   , 15998.   , 15690.   , 15750.   ,
        7975.   ,  7995.   ,  8195.   ,  9495.   ,  9995.   , 11595.   ,
        9980.   , 13295.   , 13845.   , 12290.   , 12940.   , 13415.   ,
       15985.   , 16515.   , 18420.   , 18950.   , 16845.   , 19045.   ,
       21485.   , 22470.   , 22625.   ])
```

2. Data Preprocessing.

In [41]:
```
# check the data type
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   car_ID            205 non-null    int64  
 1   symboling         205 non-null    int64  
 2   CarName           205 non-null    object 
 3   fueltype          205 non-null    object 
 4   aspiration        205 non-null    object 
 5   doornumber        205 non-null    int64  
 6   carbody           205 non-null    object 
 7   drivewheel        205 non-null    object 
 8   enginelocation    205 non-null    object 
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64  
 14  enginetype        205 non-null    object 
 15  cylindernumber    205 non-null    int64  
 16  enginesize        205 non-null    int64  
 17  fuelsystem        205 non-null    object 
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64  
 22  peakrpm           205 non-null    int64  
 23  citympg           205 non-null    int64  
 24  highwaympg        205 non-null    int64  
 25  price             205 non-null    float64
dtypes: float64(8), int64(10), object(8)
memory usage: 41.8+ KB
```

In [42]:
```
# check for null values
df.isna().sum()
```

Out[42]:
```
car_ID              0
symboling           0
CarName             0
fueltype            0
aspiration          0
doornumber          0
carbody             0
drivewheel          0
enginelocation      0
wheelbase           0
carlength           0
carwidth            0
carheight           0
curbweight          0
enginetype          0
cylindernumber      0
enginesize          0
fuelsystem          0
boreratio           0
stroke              0
compressionratio    0
horsepower          0
peakrpm             0
citympg             0
highwaympg          0
price               0
dtype: int64
```

In [43]:
```
# check for duplicates
df.duplicated().sum()
```

Out[43]:
```
0
```

In [44]:
```
df.columns
```

Out[44]:
```
Index(['car_ID', 'symboling', 'CarName', 'fueltype', 'aspiration',
       'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'wheelbase',
       'carlength', 'carwidth', 'carheight', 'curbweight', 'enginetype',
       'cylindernumber', 'enginesize', 'fuelsystem', 'boreratio', 'stroke',
       'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'highwaympg',
       'price'],
      dtype='object')
```

In [45]:
```
# statistical analysis
```

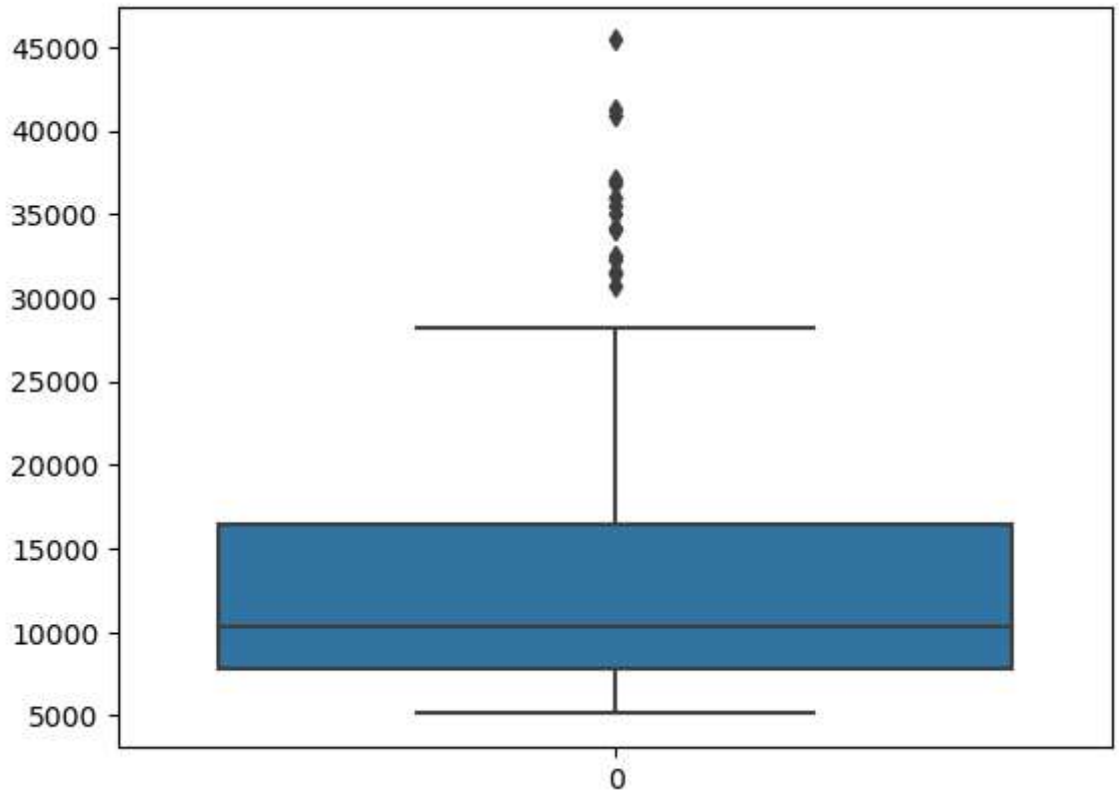
```
df.describe()
```

Out[45]:

	car_ID	symboling	doornumber	wheelbase	carlength	carwidth	carheight	curbweight	cylindernumber	enginesize	boreratio
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	103.000000	0.834146	3.121951	98.756585	174.049268	65.907805	53.724878	2555.565854	4.380488	126.907317	3.329756
std	59.322565	1.245307	0.994966	6.021776	12.337289	2.145204	2.443522	520.680204	1.080854	41.642693	0.270844
min	1.000000	-2.000000	2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	2.000000	61.000000	2.540000
25%	52.000000	0.000000	2.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	4.000000	97.000000	3.150000
50%	103.000000	1.000000	4.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	4.000000	120.000000	3.310000
75%	154.000000	2.000000	4.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	4.000000	141.000000	3.580000
max	205.000000	3.000000	4.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	12.000000	326.000000	3.940000

In [46]:
```
#check for outliers
sns.boxplot(data=df['price'])
# ouliers are there, but not removing it.
```

Out[46]:
```
<AxesSubplot:>
```



In [47]:
```
df.columns
```

Out[47]:
```
Index(['car_ID', 'symboling', 'CarName', 'fueltype', 'aspiration',
       'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'wheelbase',
       'carlength', 'carwidth', 'carheight', 'curbweight', 'enginetype',
       'cylindernumber', 'enginesize', 'fuelsystem', 'boreratio', 'stroke',
       'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'highwaympg',
       'price'],
      dtype='object')
```

In [48]:
```
df.head()
```

Out[48]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fuelsy
0	1	3	alfa-romero giulia	gas	std	2	convertible	rwd	front	88.6	...	130	
1	2	3	alfa-romero stelvio	gas	std	2	convertible	rwd	front	88.6	...	130	
2	3	1	alfa-romero Quadrifoglio	gas	std	2	hatchback	rwd	front	94.5	...	152	
3	4	2	audi 100 ls	gas	std	4	sedan	fwd	front	99.8	...	109	
4	5	2	audi 100ls	gas	std	4	sedan	4wd	front	99.4	...	136	

5 rows × 26 columns

In [49]:
```
#Encode the categorical datas from the dataset. For Ml Model generation
```

In [50]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   car_ID            205 non-null    int64  
 1   symboling         205 non-null    int64  
 2   CarName           205 non-null    object 
 3   fueltype          205 non-null    object 
 4   aspiration        205 non-null    object 
 5   doornumber        205 non-null    int64  
 6   carbody           205 non-null    object 
 7   drivewheel        205 non-null    object 
 8   enginelocation    205 non-null    object 
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64  
 14  enginetype        205 non-null    object 
 15  cylindernumber    205 non-null    int64  
 16  enginesize        205 non-null    int64  
 17  fuelsystem        205 non-null    object 
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64  
 22  peakrpm           205 non-null    int64  
 23  citympg           205 non-null    int64  
 24  highwaympg        205 non-null    int64  
 25  price             205 non-null    float64
dtypes: float64(8), int64(10), object(8)
memory usage: 41.8+ KB
```

In [68]:
```
#Encoding using Label Encoder
from sklearn import preprocessing
label=preprocessing.LabelEncoder()
```

In [76]:
```
label.fit(df.fueltype)
df.fueltype=label.transform(df.fueltype)
```

In [69]:
```
label.fit(df.aspiration)
df.aspiration=label.transform(df.aspiration)
```

In [70]:
```
label.fit(df.carbody)
df.carbody=label.transform(df.carbody)
```

In [71]:
```
label.fit(df.drivewheel)
df.drivewheel=label.transform(df.drivewheel)
```

In [72]:
```
label.fit(df.enginelocation)
df.enginelocation=label.transform(df.enginelocation)
```

In [73]:
```
label.fit(df.enginetype)
df.enginetype=label.transform(df.enginetype)
```

In [74]:
```
label.fit(df.fuelsystem)
df.fuelsystem=label.transform(df.fuelsystem)
```

In [77]:
```
df.head()
```

Out[77]:

	car_ID	symboling	CarName	fueltype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fuelsyst
0	1	3	alfa-romero giulia	1	0	2	0	2	0	88.6	...	130	
1	2	3	alfa-romero stelvio	1	0	2	0	2	0	88.6	...	130	
2	3	1	alfa-romero Quadrifoglio	1	0	2	2	2	0	94.5	...	152	
3	4	2	audi 100 ls	1	0	4	3	1	0	99.8	...	109	
4	5	2	audi 100ls	1	0	4	3	0	0	99.4	...	136	

5 rows × 26 columns

In [84]:
```
df.info()
```

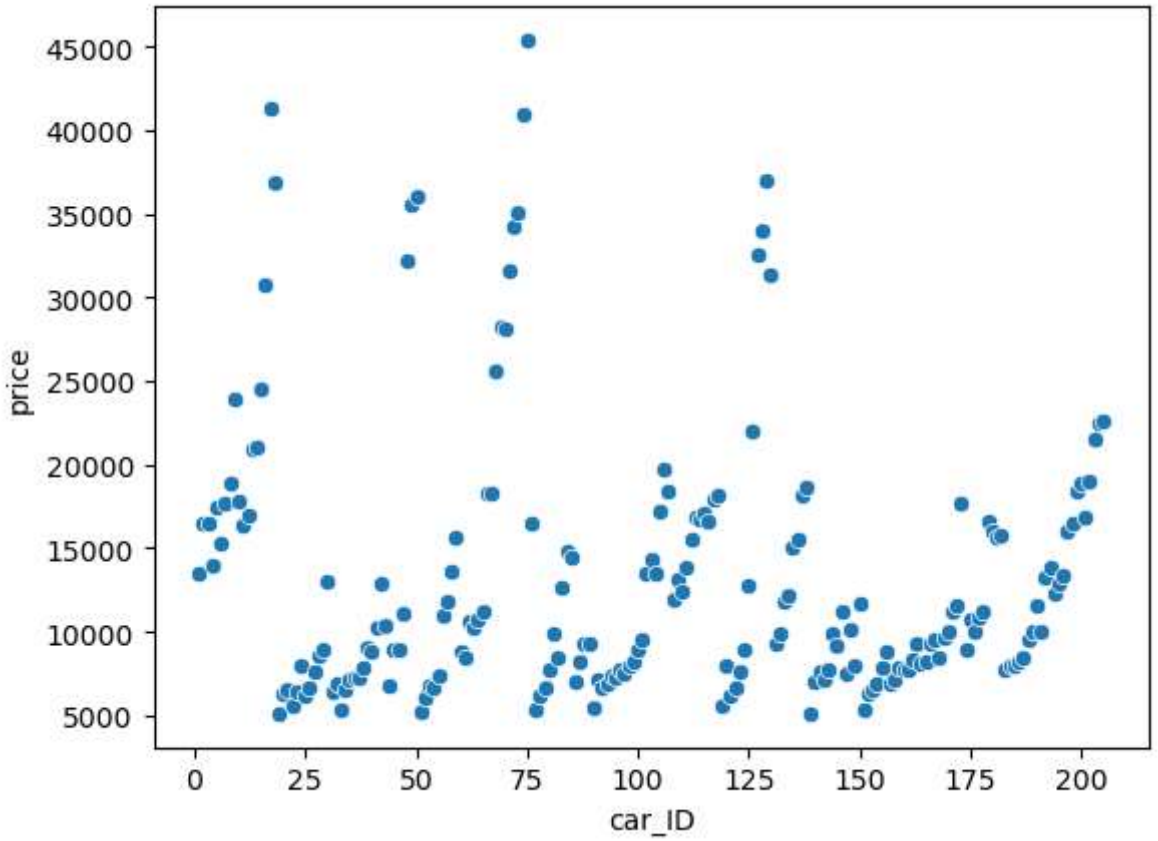
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   car_ID            205 non-null    int64  
 1   symboling         205 non-null    int64  
 2   CarName           205 non-null    object 
 3   fueltype          205 non-null    int32  
 4   aspiration        205 non-null    int64  
 5   doornumber        205 non-null    int64  
 6   carbody           205 non-null    int64  
 7   drivewheel        205 non-null    int64  
 8   enginelocation    205 non-null    int64  
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64  
 14  enginetype        205 non-null    int64  
 15  cylindernumber    205 non-null    int64  
 16  enginesize        205 non-null    int64  
 17  fuelsystem        205 non-null    int64  
 18  boreratio         205 non-null    float64
 19  stroke            205 non-null    float64
 20  compressionratio  205 non-null    float64
 21  horsepower        205 non-null    int64  
 22  peakrpm           205 non-null    int64  
 23  citympg           205 non-null    int64  
 24  highwaympg        205 non-null    int64  
 25  price             205 non-null    float64
dtypes: float64(8), int32(1), int64(16), object(1)
memory usage: 41.0+ KB
```

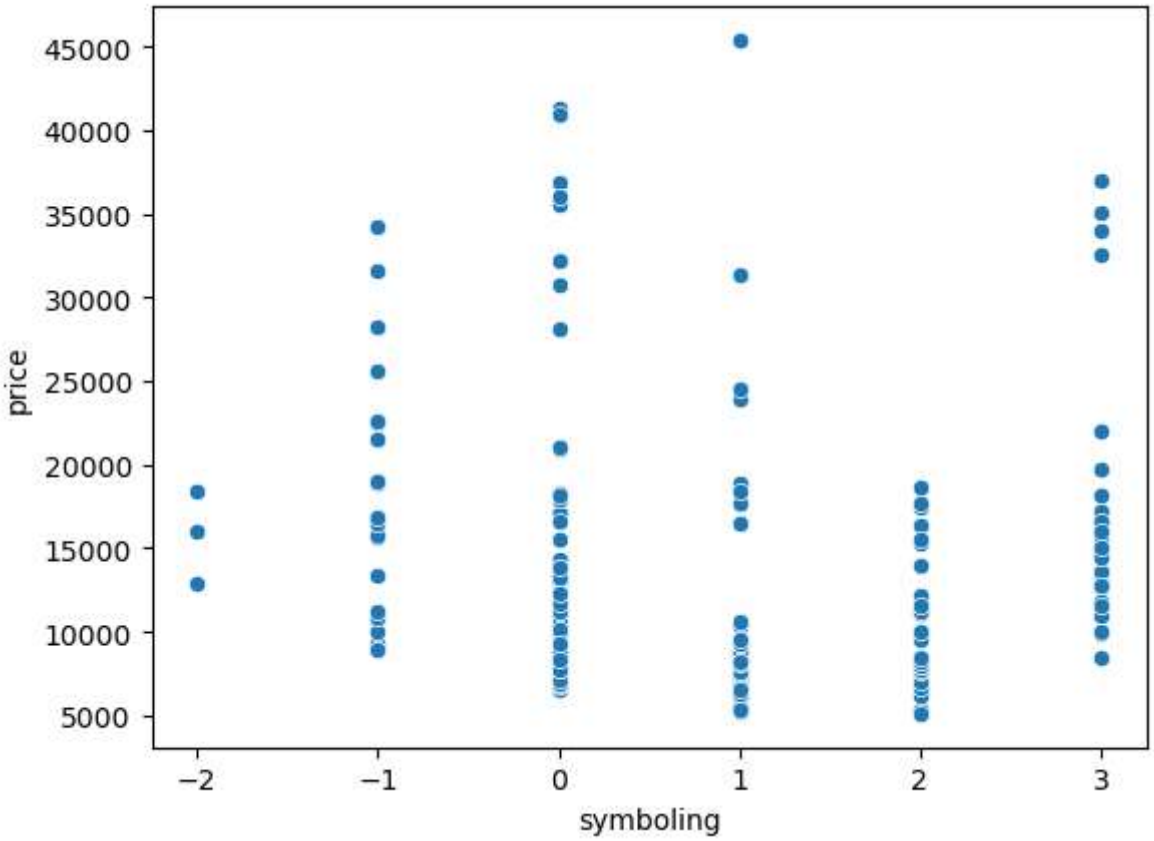
In [53]:
```
# check the correlation between 'price' and independent variables
```

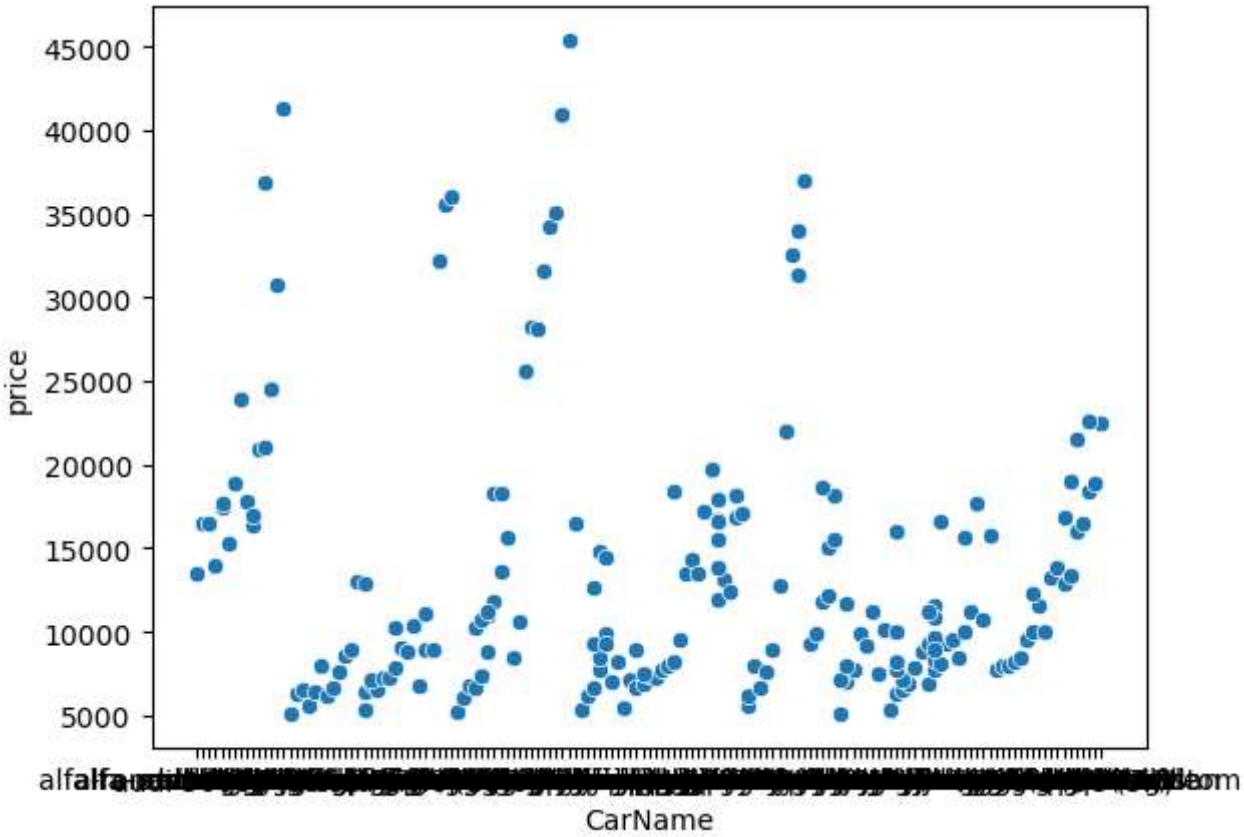
In [78]:
```
df.columns
```

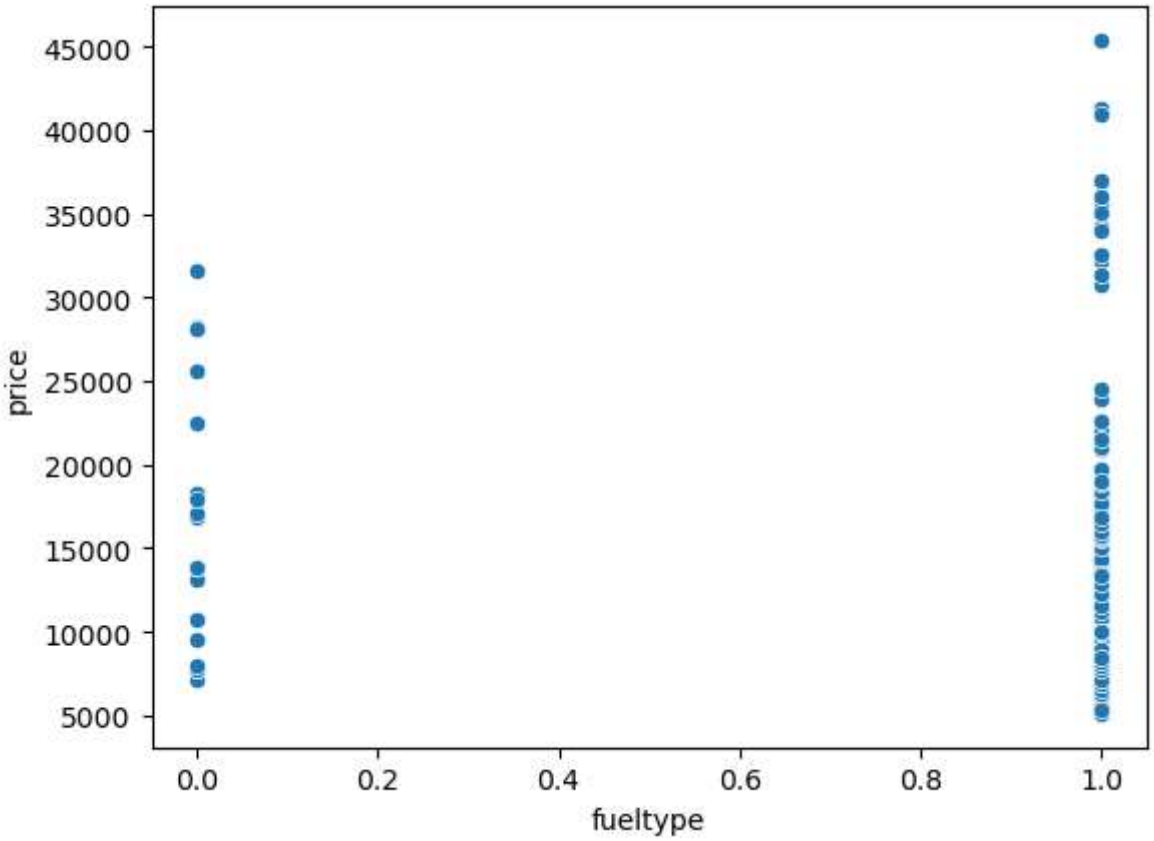
Out[78]:
```
Index(['car_ID', 'symboling', 'CarName', 'fueltype', 'aspiration',
       'doornumber', 'carbody', 'drivewheel', 'enginelocation', 'wheelbase',
       'carlength', 'carwidth', 'carheight', 'curbweight', 'enginetype',
       'cylindernumber', 'enginesize', 'fuelsystem', 'boreratio', 'stroke',
       'compressionratio', 'horsepower', 'peakrpm', 'citympg', 'highwaympg',
       'price'],
      dtype='object')
```

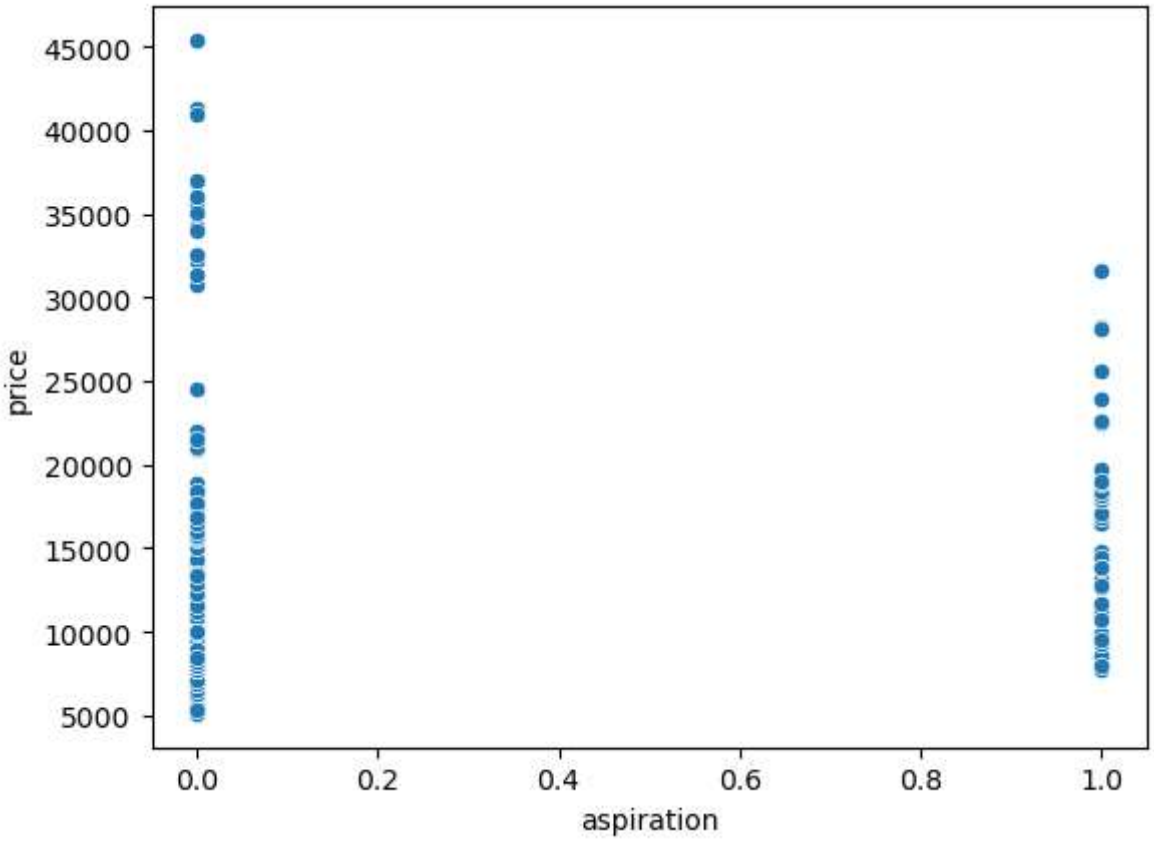
In [83]:
```
# Creating scatter plots for each numerical column against 'price'
for col in df:
    sns.scatterplot(data=df, x=col, y='price')
    plt.show()
```

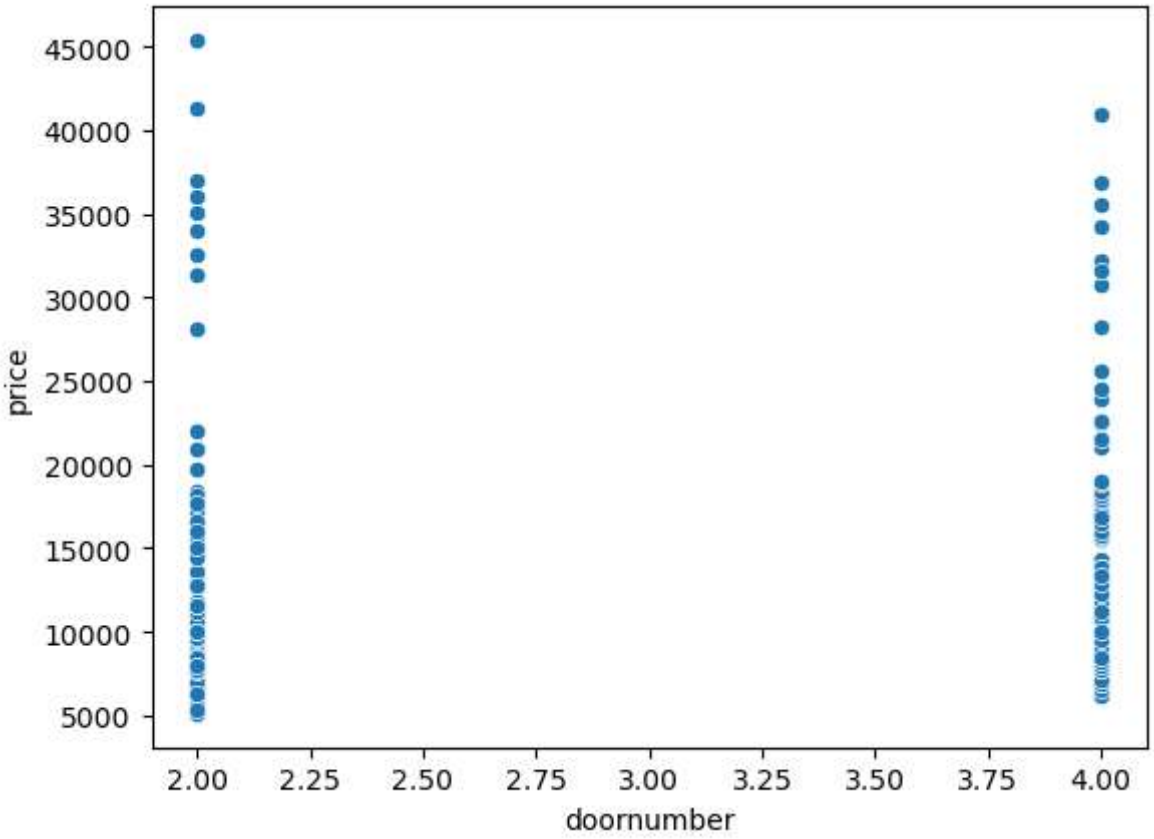


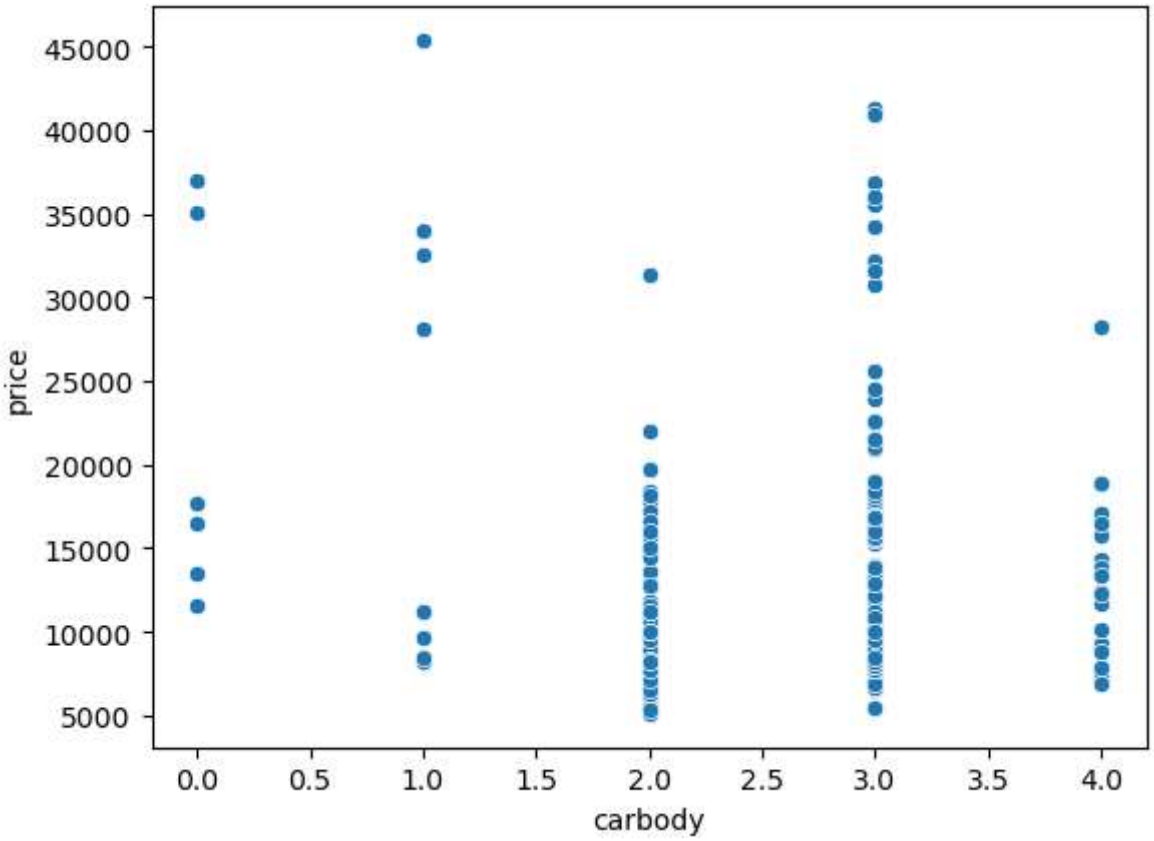

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
−2
−1
0
1
2
3
symboling

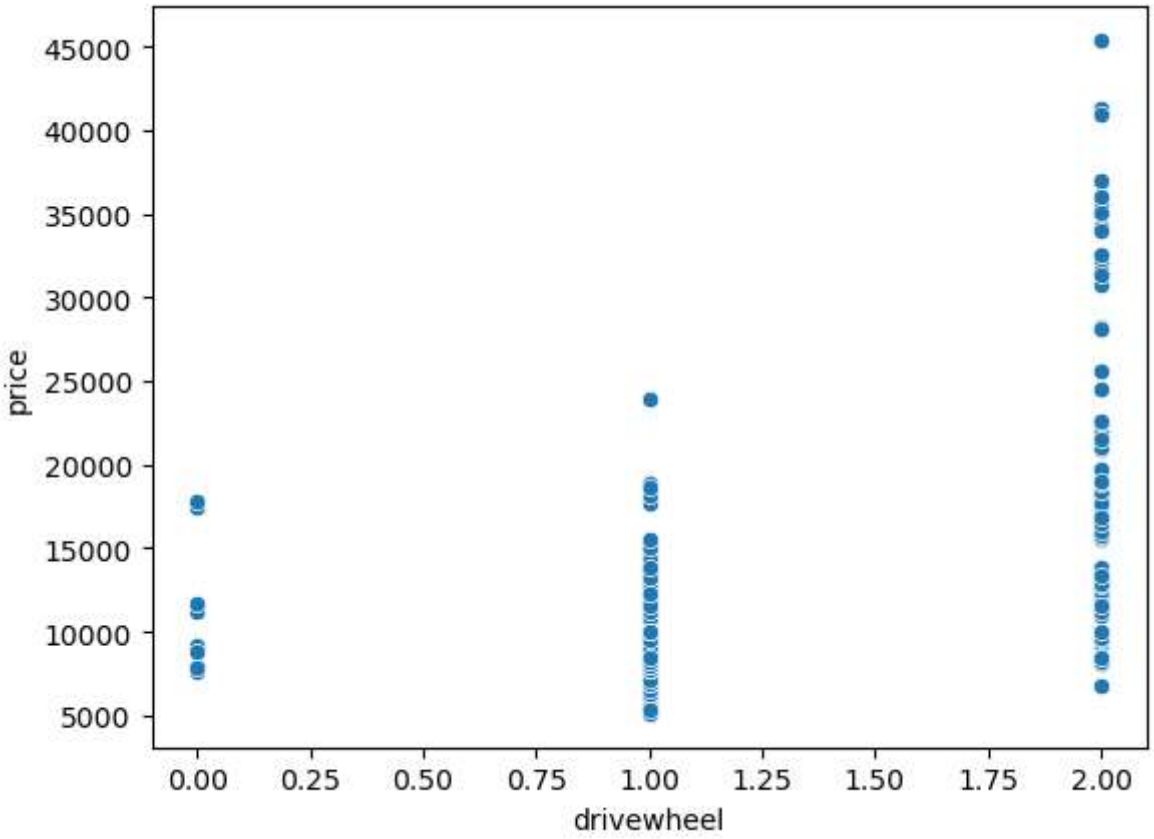

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
CarName

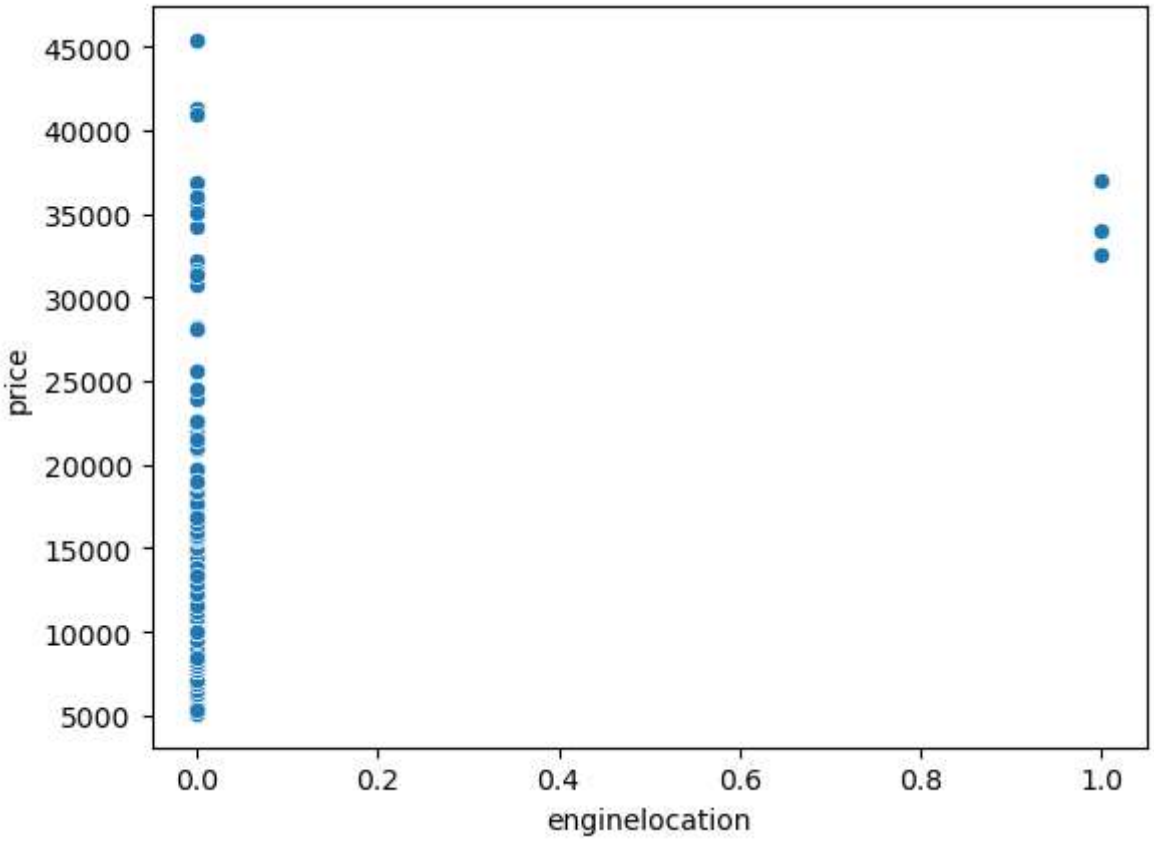

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
0.0
0.2
0.4
0.6
0.8
1.0
fueltype

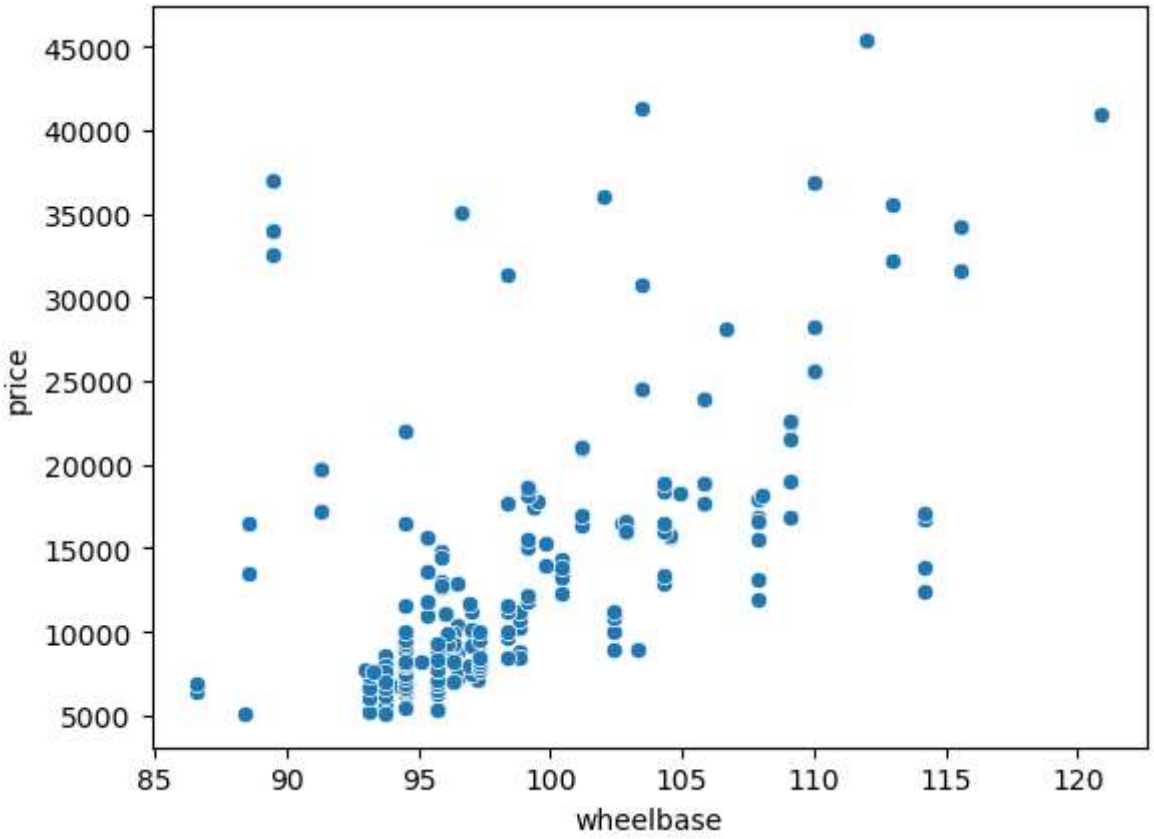

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
0.0
0.2
0.4
0.6
0.8
1.0
aspiration

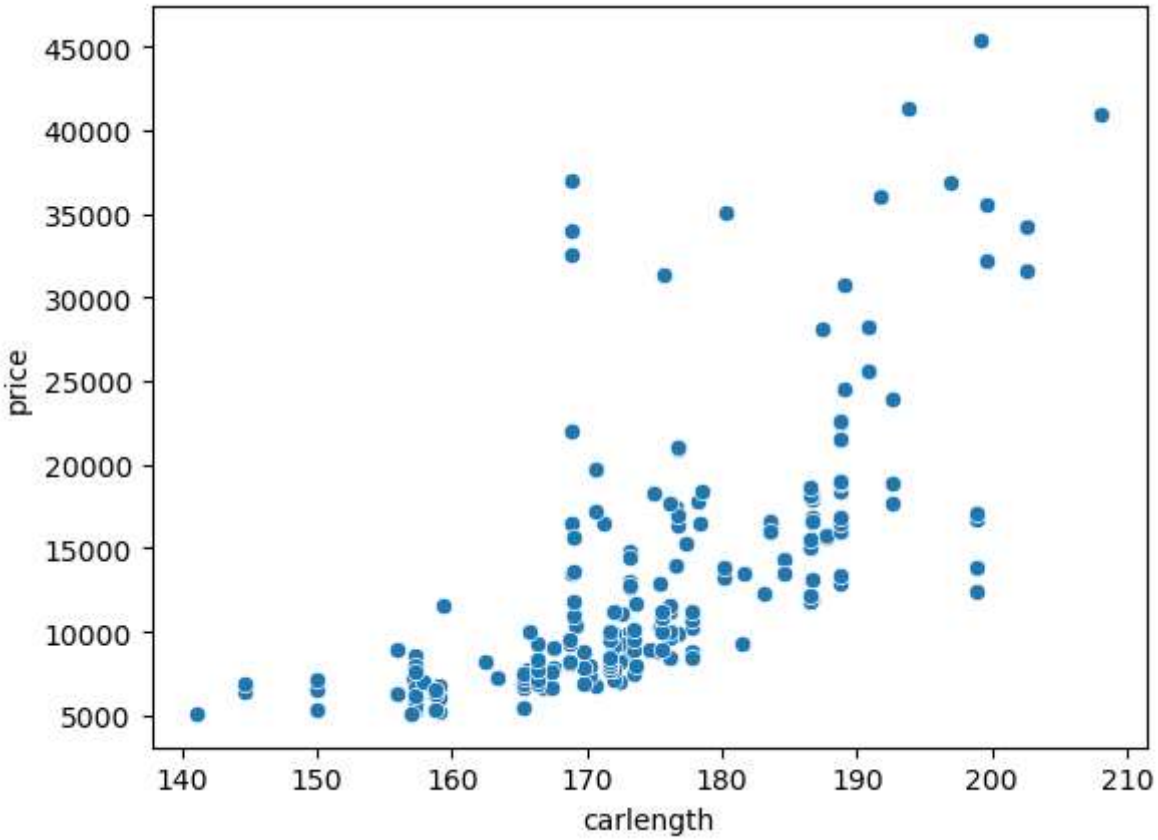

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
2.00
2.25
2.50
2.75
3.00
3.25
3.50
3.75
4.00
doornumber

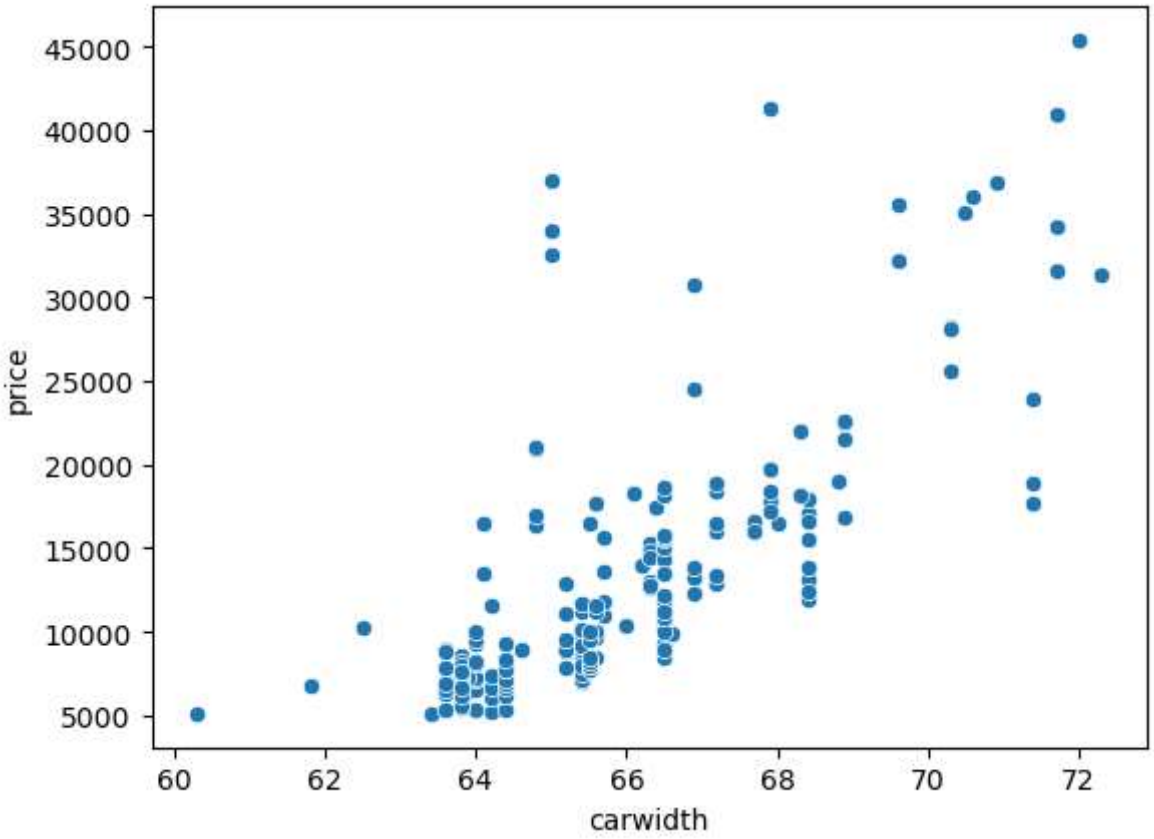

price
45000
40000
35000
30000
25000
20000
15000
10000
5000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
carbody

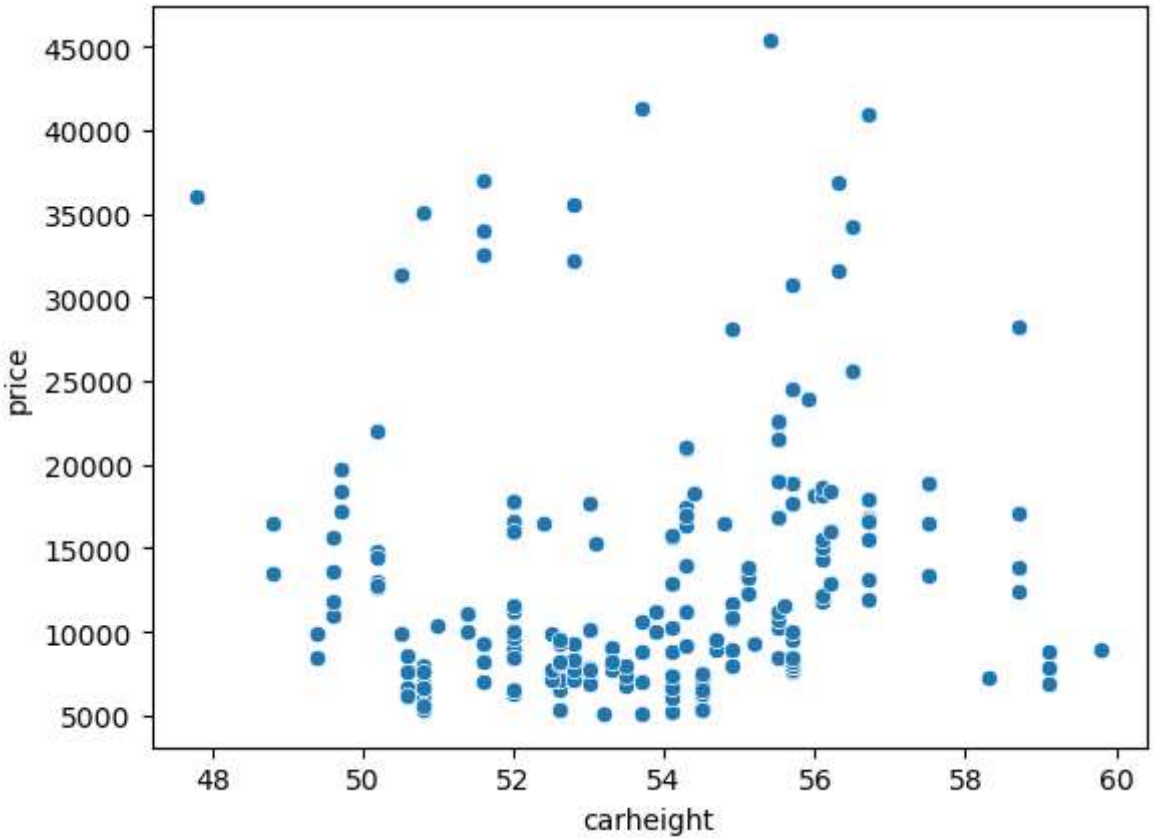

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
drivewheel

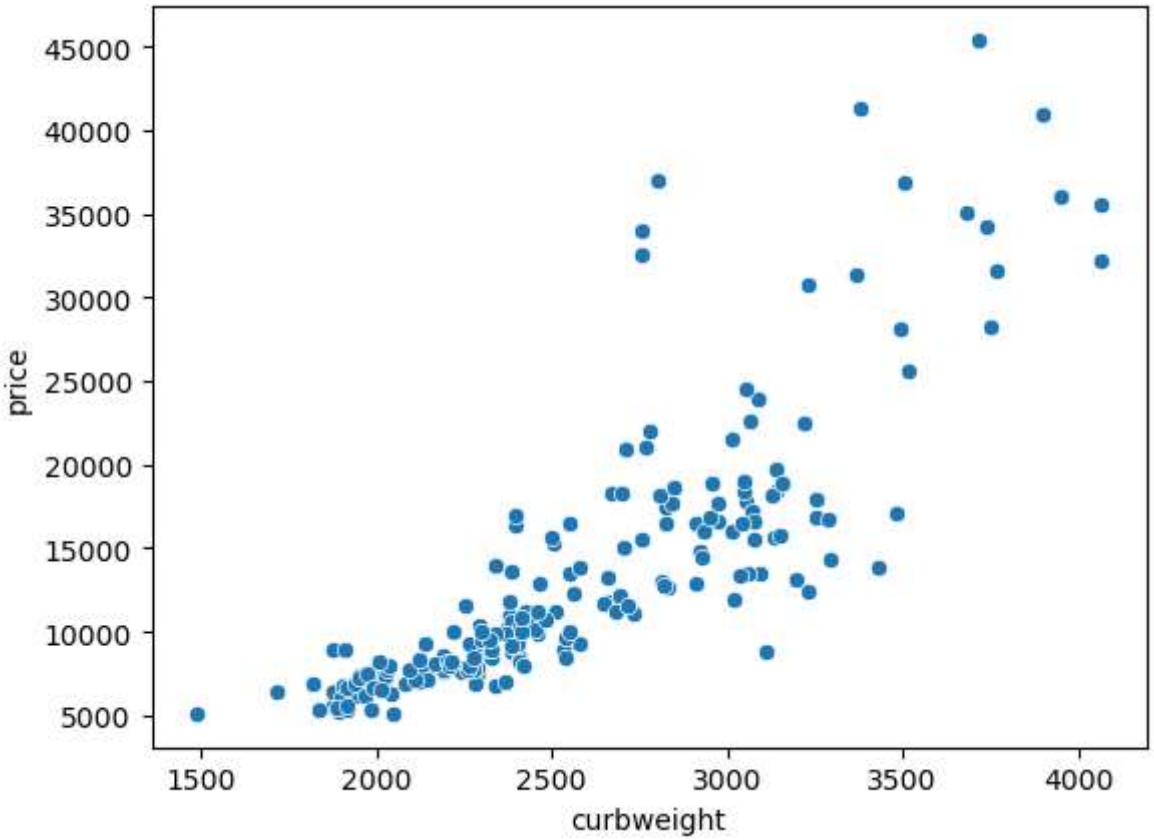

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
0.0
0.2
0.4
0.6
0.8
1.0
enginelocation

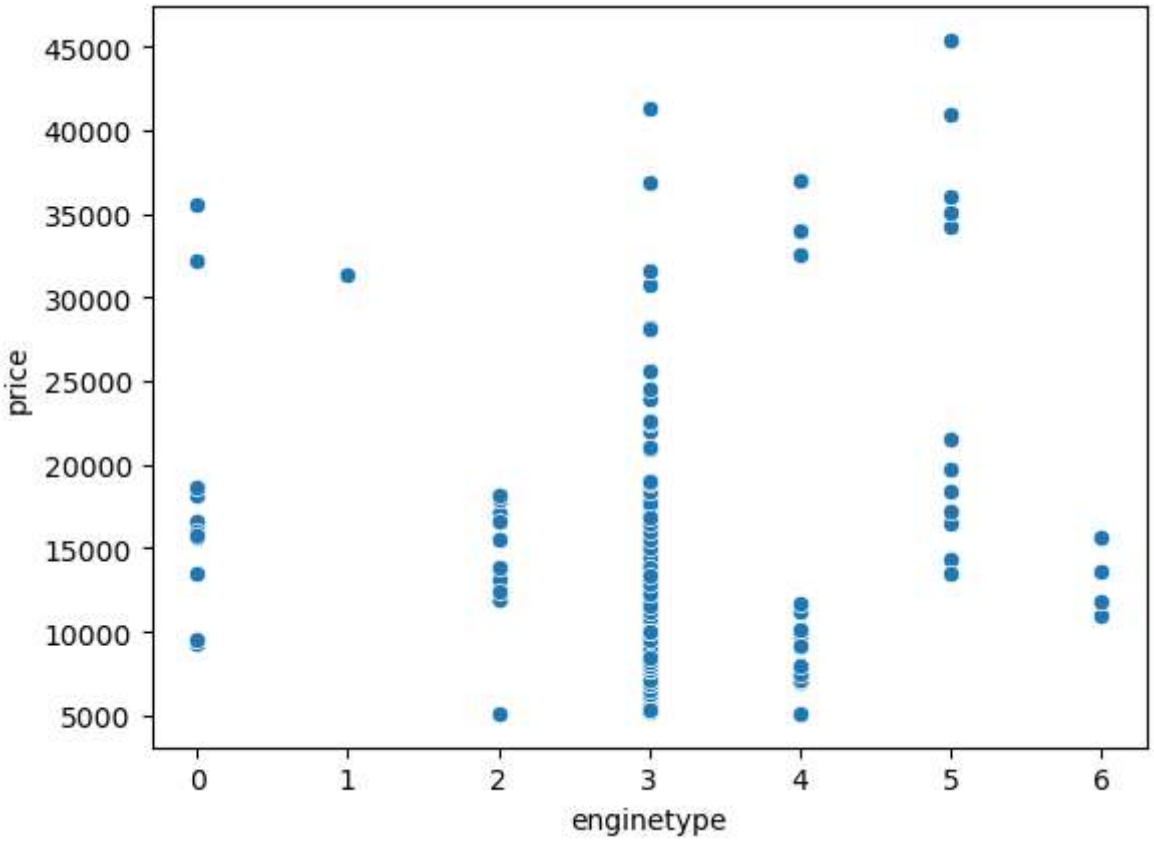

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
85
90
95
100
105
110
115
120
wheelbase

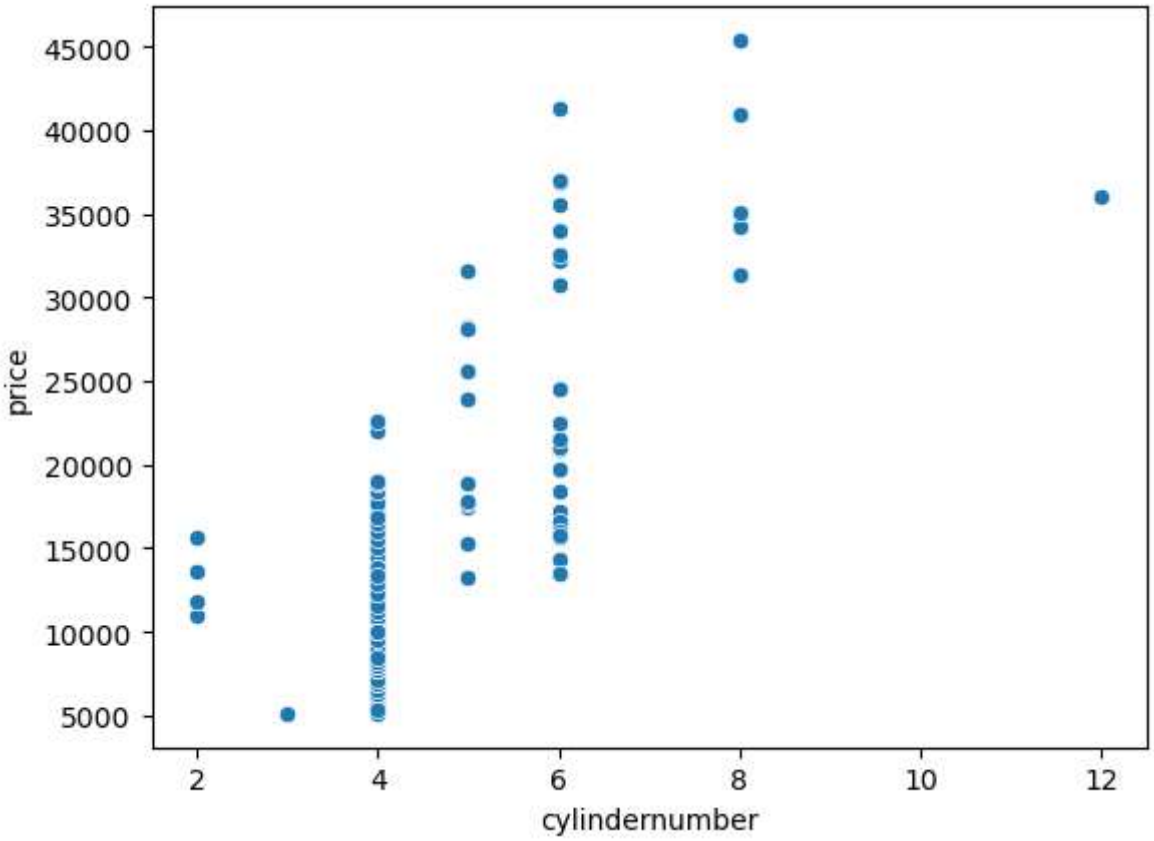

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
140
150
160
170
180
190
200
210
carlength

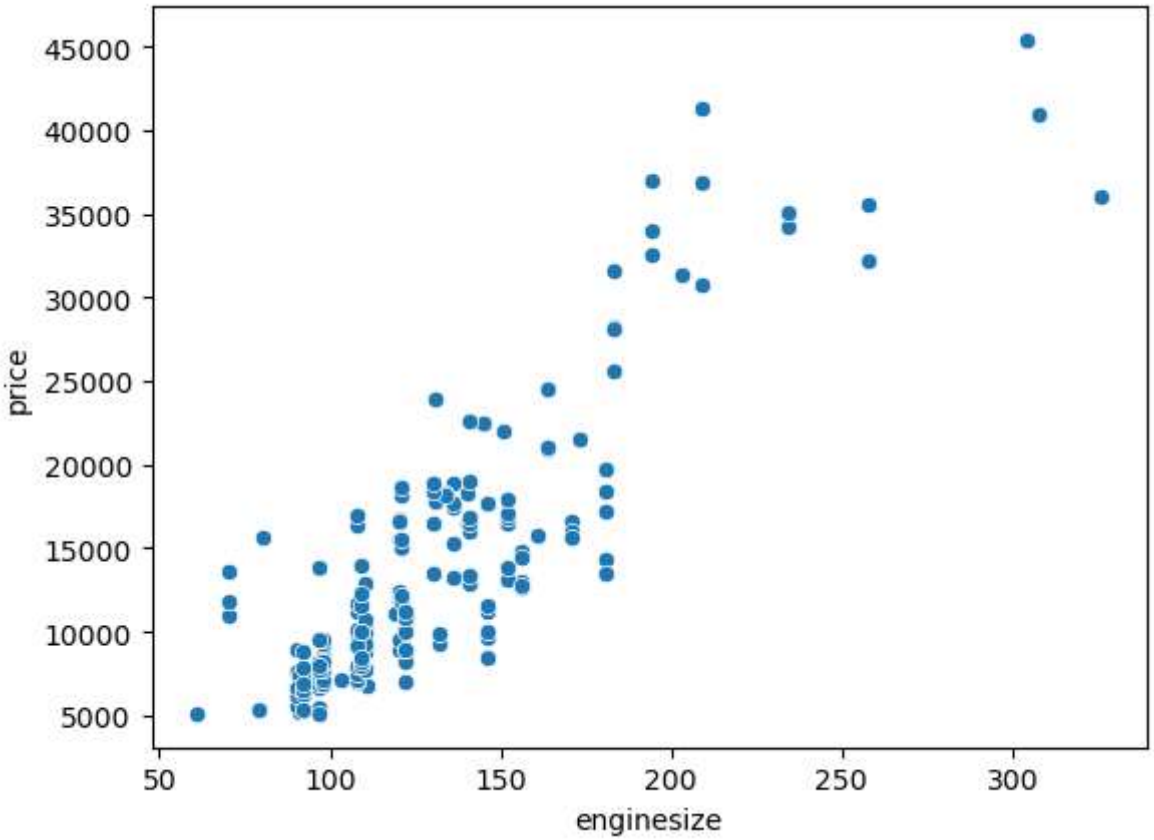

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
60
62
64
66
68
70
72
carwidth

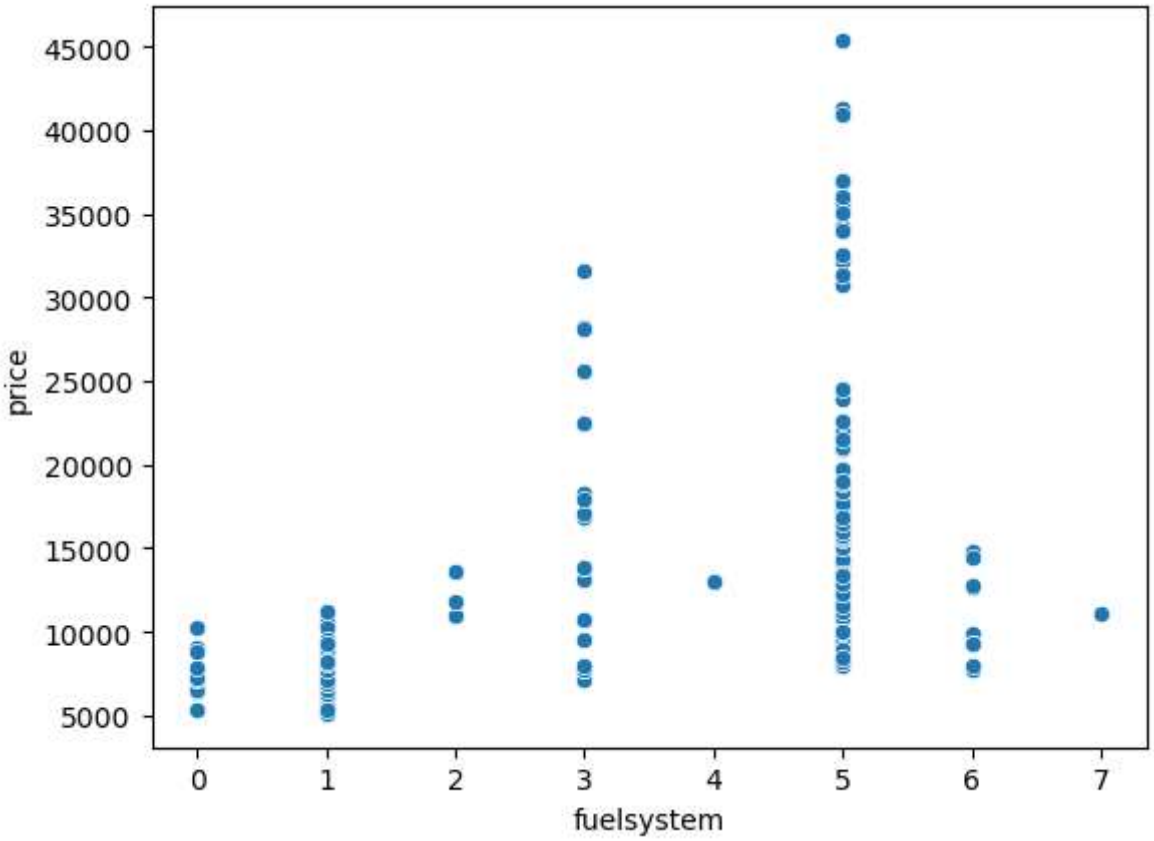

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
48
50
52
54
56
58
60
carheight

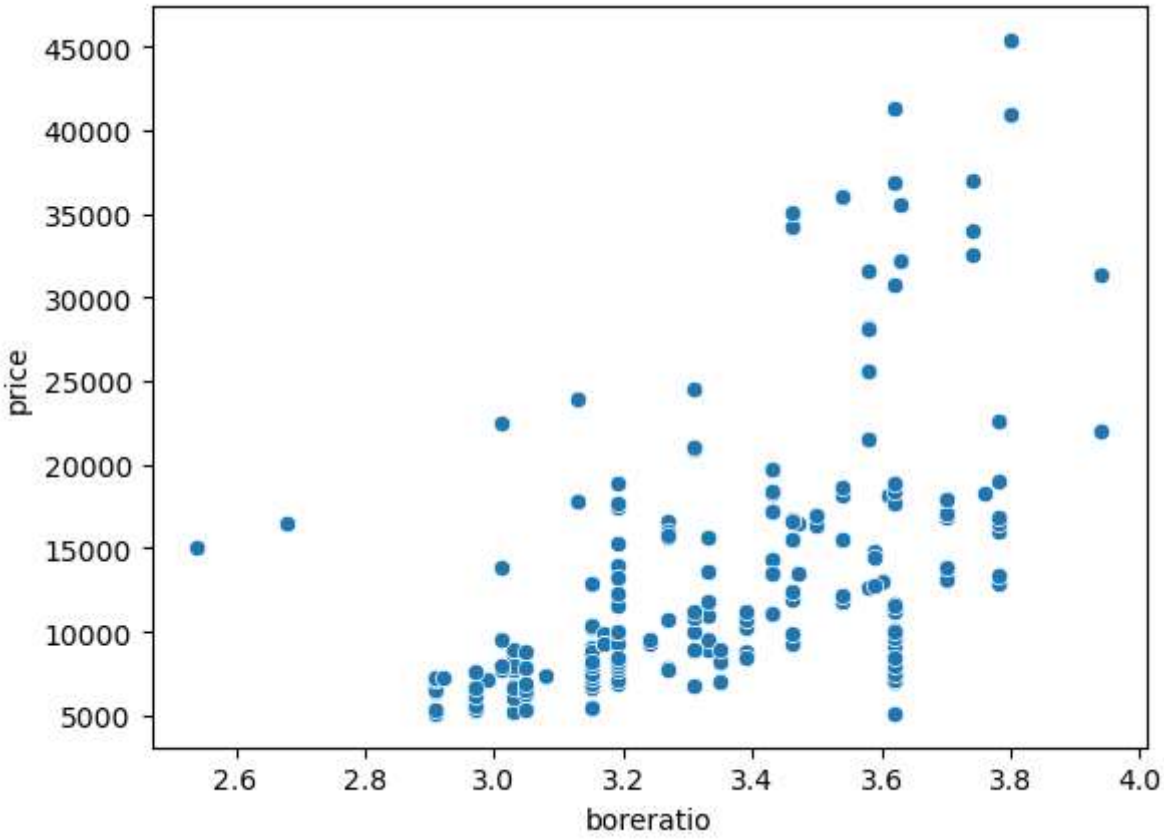

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
1500
2000
2500
3000
3500
4000
curbweight

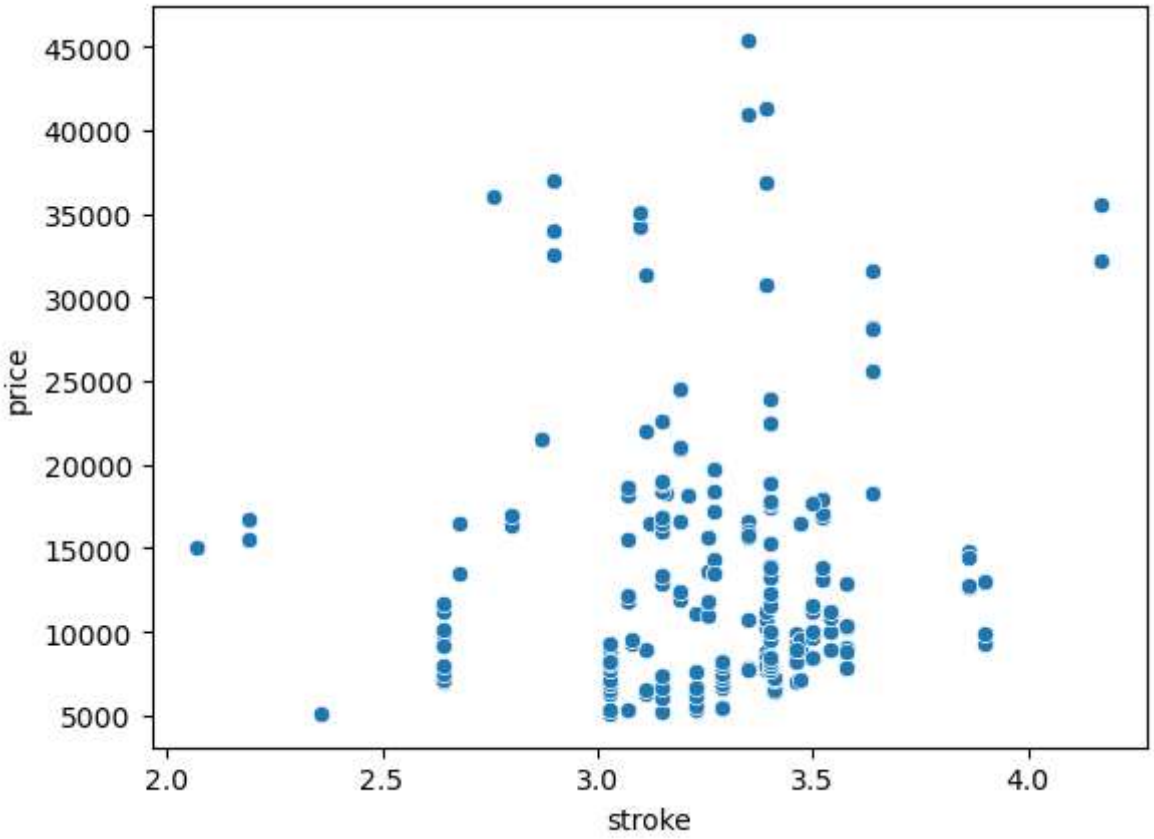

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
0
1
2
3
4
5
6
enginetype

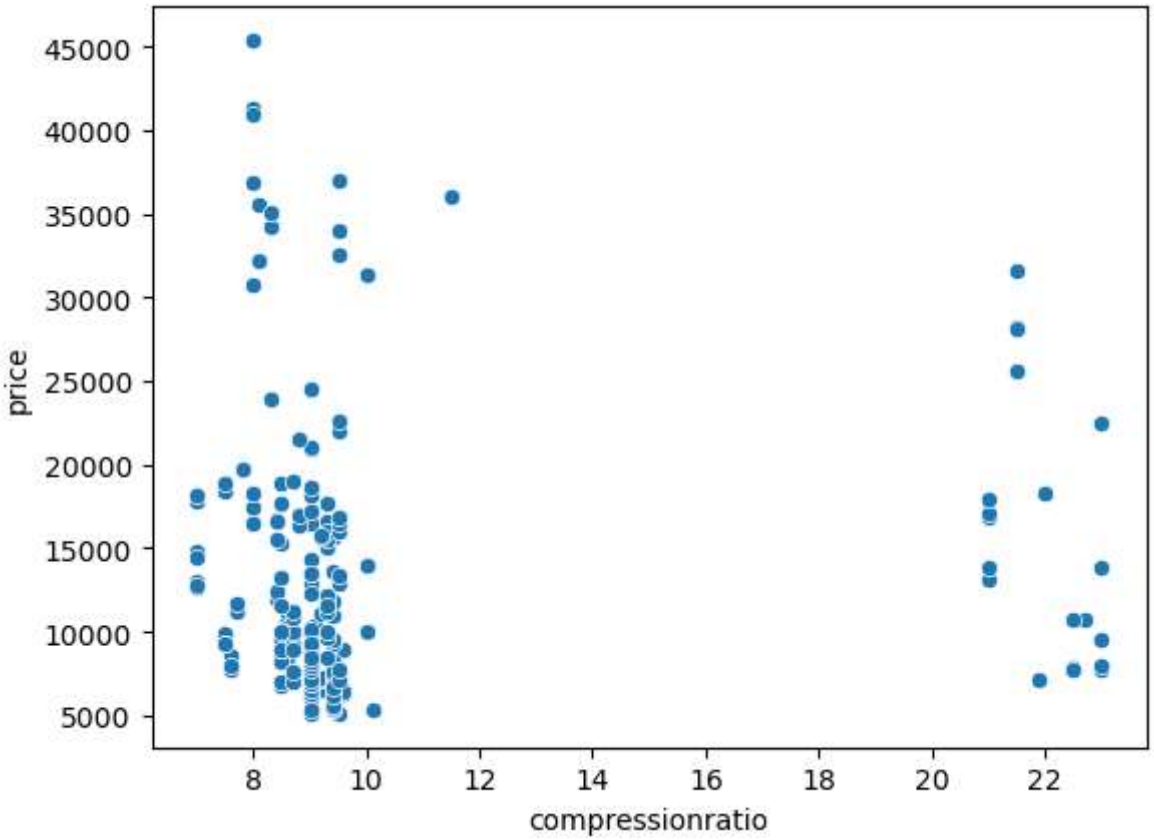

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
2
4
6
8
10
12
cylindernumber

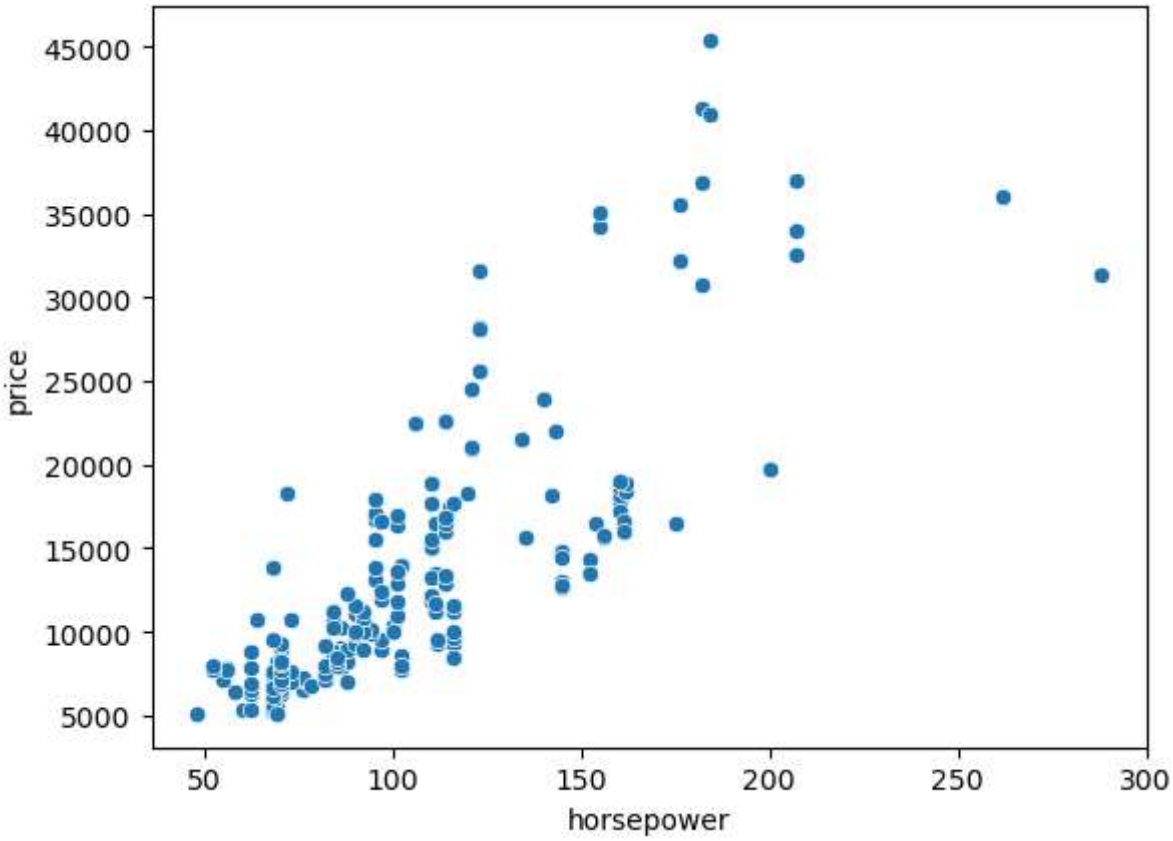

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
50
100
150
200
250
300
enginesize

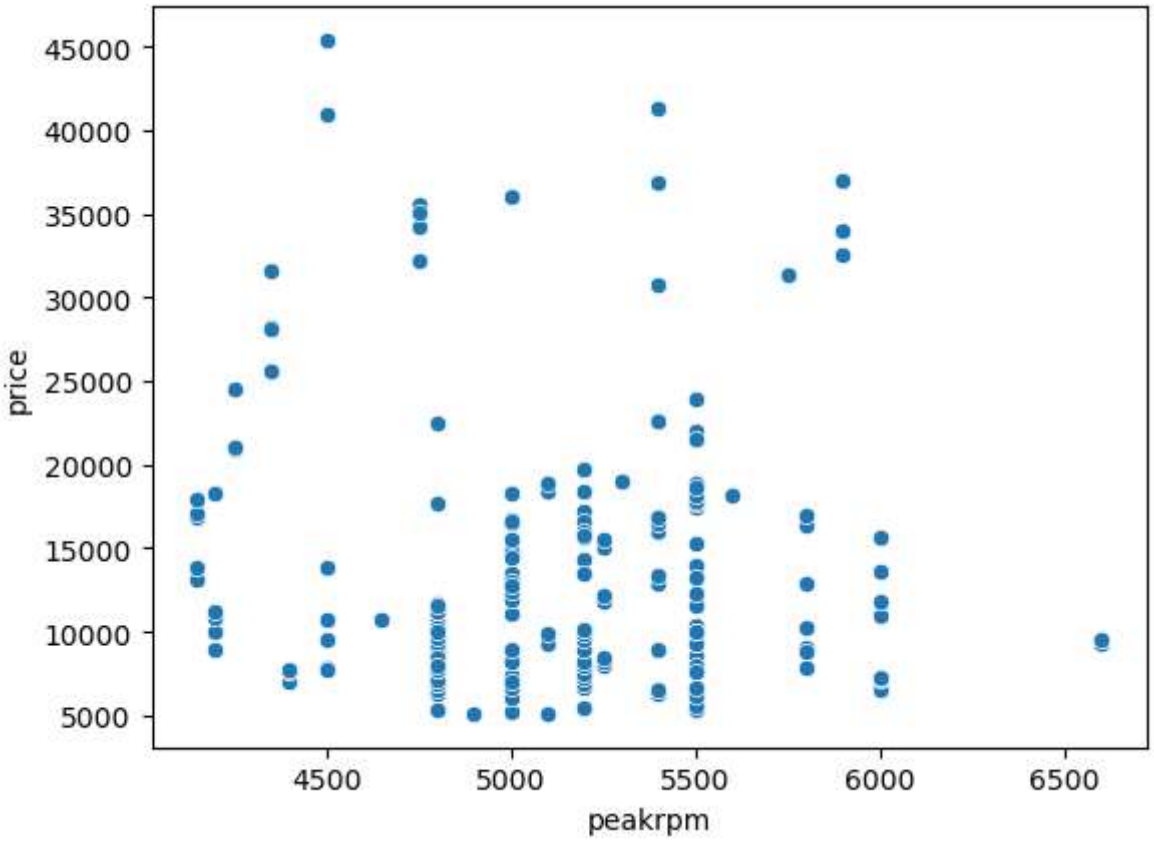

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
0
1
2
3
4
5
6
7
fuelsystem

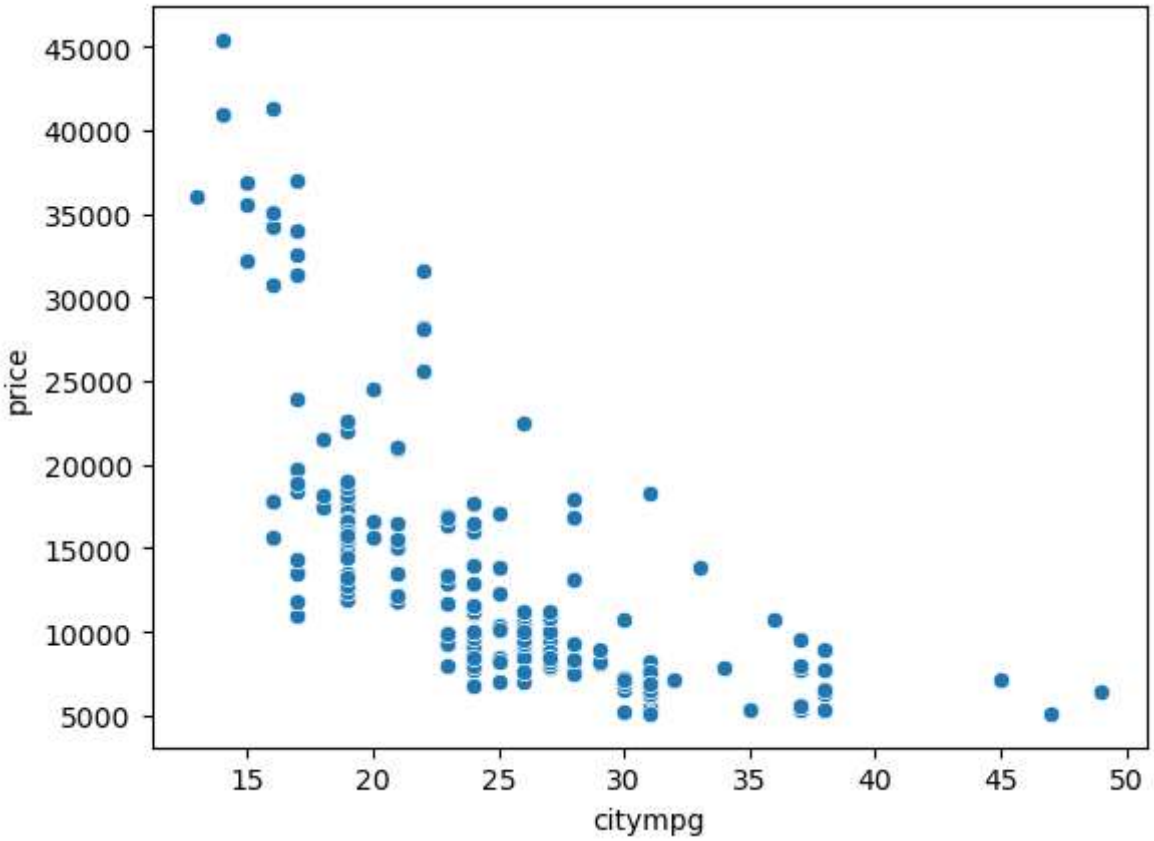

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
2.6
2.8
3.0
3.2
3.4
3.6
3.8
4.0
boreratio

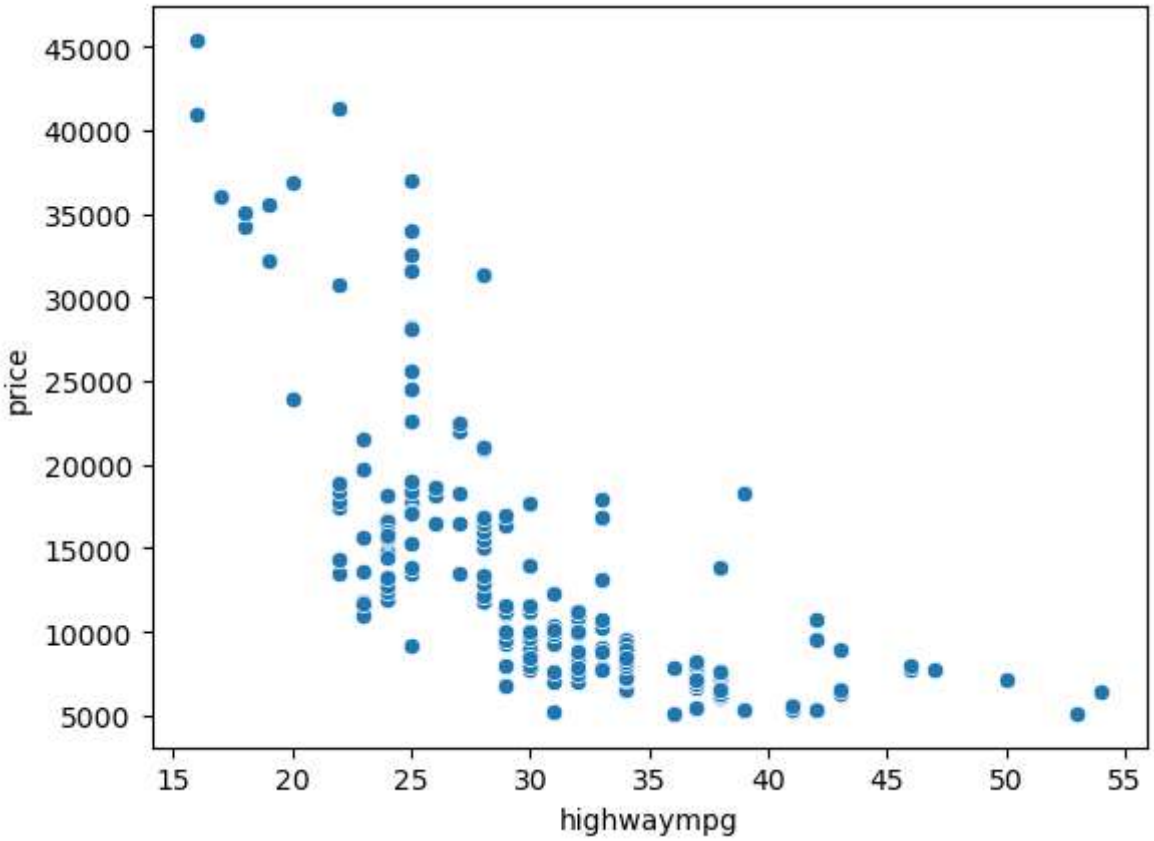

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
2.0
2.5
3.0
3.5
4.0
stroke

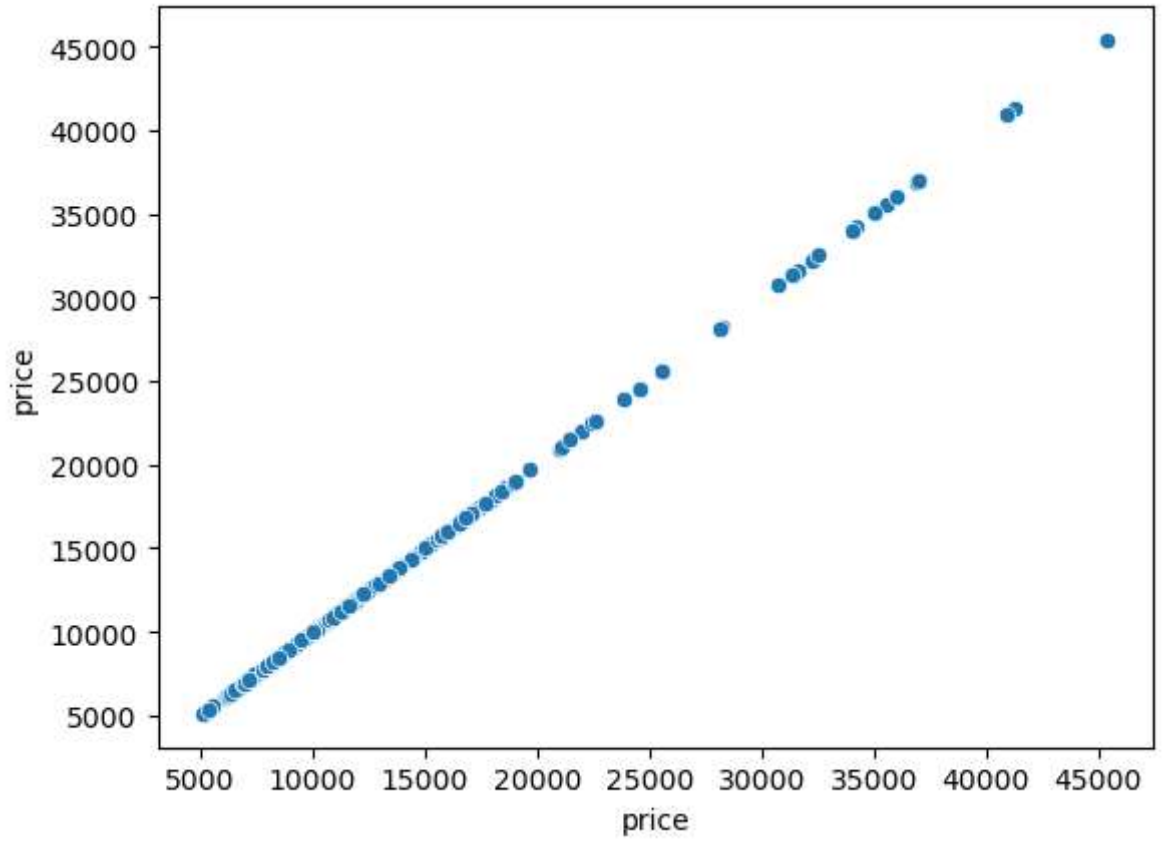

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
8
10
12
14
16
18
20
22
compressionratio


45000
40000
35000
30000
25000
20000
15000
10000
5000
price
50
100
150
200
250
300
horsepower


45000
40000
35000
30000
25000
20000
15000
10000
5000
price
4500
5000
5500
6000
6500
peakrpm


45000
40000
35000
30000
25000
20000
15000
10000
5000
price
15
20
25
30
35
40
45
50
citympg

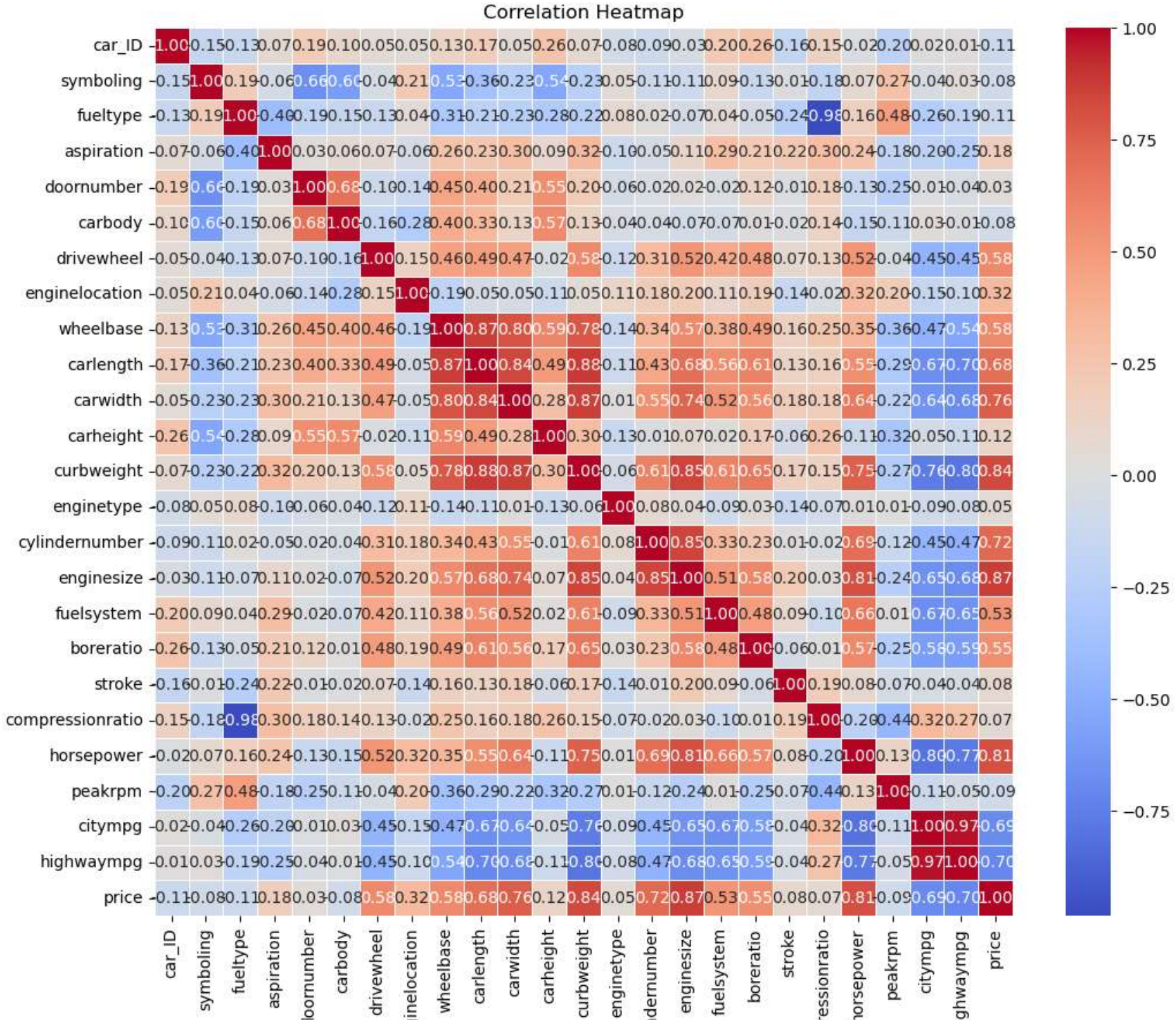

price
45000
40000
35000
30000
25000
20000
15000
10000
5000
15
20
25
30
35
40
45
50
55
highwaympg



In [ ]:
```
# plot a heatmap to show the correlation among numerical variables.
```

In [85]:
```
# Create a correlation matrix between numerical columns
correlation_matrix = df.corr()

plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```


Correlation Heatmap
car_ID
symboling
fueltype
aspiration
doornumber
carbody
drivewheel
enginelocation
wheelbase
carlength
carwidth
carheight
curbweight
enginetype
cylindernumber
enginesize
fuelsystem
boreratio
stroke
compressionratio
horsepower
peakrpm
citympg
highwaympg
price
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

d eng cylin compr hi

Observations from scatter plot and heatmap

The independent variables that affects the price of car are,

1. 'drivewheel'
2. 'wheelbase'
3. 'carlength'
4. 'carwidth'
5. 'curbweight'
6. 'cylindernumber'
7. 'enginesize'
8. 'fuelsystem'
9. 'boreratio'
10. 'horsepower'
11. 'citympg' (negative correlation)
12. 'highwaympg' (negative correlation)

In [86]:
```
#sort the dataset with the necessary features only.
columns_to_sort=['drivewheel','wheelbase','carlength','carwidth',
                 'curbweight','cylindernumber','enginesize','fuelsystem','boreratio','horsepower','citympg','highwaympg']
```

In [94]:
```
independent_data_for_model=df[columns_to_sort]
independent_data_for_model.head()
```

Out[94]:

	drivewheel	wheelbase	carlength	carwidth	curbweight	cylindernumber	enginesize	fuelsystem	boreratio	horsepower	citympg	highwa
0	2	88.6	168.8	64.1	2548	4	130	5	3.47	111	21	
1	2	88.6	168.8	64.1	2548	4	130	5	3.47	111	21	
2	2	94.5	171.2	65.5	2823	6	152	5	2.68	154	19	
3	1	99.8	176.6	66.2	2337	4	109	5	3.19	102	24	
4	0	99.4	176.6	66.4	2824	5	136	5	3.19	115	18	

Create a ML model for Car Price Prediction using Linear Regression-Multiple Variables

The independent variables that affects the price of car are,

1. 'drivewheel'
2. 'wheelbase'
3. 'carlength'
4. 'carwidth'
5. 'curbweight'
6. 'cylindernumber'
7. 'enginesize'
8. 'fuelsystem'
9. 'boreratio'
10. 'horsepower'
11. 'citympg' (negative correlation)
12. 'highwaympg' (negative correlation)

In [90]:
```
#Using Linear Regression Model
```

In [101]:
```
#spliting the dataset to x(independent variables) and y(dependent variable='Price')
x=df[['drivewheel','wheelbase','carlength','carwidth',
                'curbweight','cylindernumber','enginesize','fuelsystem','boreratio','horsepower','citympg','highwaympg']]
y=df.price
```

In [104]:
```
#independant variable,x
x
```

Out[104]:

	drivewheel	wheelbase	carlength	carwidth	curbweight	cylindernumber	enginesize	fuelsystem	boreratio	horsepower	citympg	highv
0	2	88.6	168.8	64.1	2548	4	130	5	3.47	111	21	
1	2	88.6	168.8	64.1	2548	4	130	5	3.47	111	21	
2	2	94.5	171.2	65.5	2823	6	152	5	2.68	154	19	
3	1	99.8	176.6	66.2	2337	4	109	5	3.19	102	24	
4	0	99.4	176.6	66.4	2824	5	136	5	3.19	115	18	
...	...	...	...	...	...	...	...	...	...	...	...	
200	2	109.1	188.8	68.9	2952	4	141	5	3.78	114	23	
201	2	109.1	188.8	68.8	3049	4	141	5	3.78	160	19	
202	2	109.1	188.8	68.9	3012	6	173	5	3.58	134	18	
203	2	109.1	188.8	68.9	3217	6	145	3	3.01	106	26	
204	2	109.1	188.8	68.9	3062	4	141	5	3.78	114	19	

205 rows × 12 columns

In [105]:
```
#dependant variable,y
y
```

Out[105]:
```
0      13495.0
1      16500.0
2      16500.0
3      13950.0
4      17450.0
        ...
200    16845.0
201    19045.0
202    21485.0
203    22470.0
204    22625.0
Name: price, Length: 205, dtype: float64
```

In [91]:
```
#import libraries
from sklearn.linear_model import LinearRegression
```

In [93]:
```
linre=LinearRegression()
linre
```

Out[93]:
```
LinearRegression()
```

In [106]:
```
linre.fit(x,y)
```

Out[106]:
```
LinearRegression()
```

In [118]:
```
#predict the price
value=[[2,88.6,168.8,64.1,2548,4,130,5,3.47,111,21,27]]
predicted=linre.predict(value)
print(predicted)
```

```
[13039.64047938]
C:\ProgramData\Anaconda\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

In [109]:
```
#check the Accuracy of created ML model
linre.score(x,y)
```

Out[109]:
```
0.8304847089055049
```

In [119]:
```
x.head(1)
```

Out[119]:

	drivewheel	wheelbase	carlength	carwidth	curbweight	cylindernumber	enginesize	fuelsystem	boreratio	horsepower	citympg	highwa
0	2	88.6	168.8	64.1	2548	4	130	5	3.47	111	21	

In [114]:
```
y.head(1)
```

Out[114]:
```
0    13495.0
Name: price, dtype: float64
```

Result analysis

1. The Score of created ML model is good.
2. The predicted value lies closer to the actual value, So The created ML model is working good.

In [ ]: