CAR PRICE PREDICTION

Here I required to model the price of cars with the available independent variables. It will be used 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 management to understand the pricing dynamics of a new market.

Here I used 'Car Data' which I got from kaggle.

The independent and dependent variables found in the dataset I used to create this model are displayed in the table below.

DATA DICTONARY		
1	**Car_ID**	Unique id of each observation (Interger)
2	**Symboling**	Its assigned insurance risk rating, A value of +3 indicates that the auto is risky, -3 that it is probably pretty safe.(Categorical)
3	**carCompany**	Name of car company (Categorical)
4	**fueltype**	Car fuel type i.e gas or diesel (Categorical)
5	**aspiration**	Aspiration used in a car (Categorical)
6	**doornumber**	Number of doors in a car (Categorical)
7	**carbody**	body of car (Categorical)
8	**drivewheel**	type of drive wheel (Categorical)
9	**enginelocation**	Location of car engine (Categorical)
10	**wheelbase**	Weelbase of car (Numeric)
11	**carlength**	Length of car (Numeric)
12	**carwidth**	Width of car (Numeric)
13	**carheight**	height of car (Numeric)
14	**curbweight**	The weight of a car without occupants or baggage. (Numeric)
15	**enginetype**	Type of engine. (Categorical)
16	**cylindernumber**	cylinder placed in the car (Categorical)
17	**enginesize**	Size of car (Numeric)
18	**fuelsystem**	Fuel system of car (Categorical)
19	**boreratio**	Boreratio of car (Numeric)
20	**stroke**	Stroke or volume inside the engine (Numeric)
21	**compressionratio**	compression ratio of car (Numeric)
22	**horsepower**	Horsepower (Numeric)
23	**peakrpm**	car peak rpm (Numeric)
24	**citympg**	Mileage in city (Numeric)
25	**highwaympg**	Mileage on highway (Numeric)
26	**rice(Dependent variable**	Price of car (Numeric)

Data cleaning and preprocessing

Import Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Read datast

```
df = pd.read_csv("/content/CarPrice_Assignment.csv")
df
```

```
     car_ID  symboling                   CarName fueltype
aspiration  \
0         1          3        alfa-romero giulia      gas
std
1         2          3       alfa-romero stelvio      gas
std
2         3          1  alfa-romero Quadrifoglio      gas
std
3         4          2               audi 100 ls      gas
std
4         5          2                audi 100ls      gas
std
..      ...        ...                       ...      ...
...
200     201         -1           volvo 145e (sw)      gas
std
201     202         -1               volvo 144ea      gas
turbo
202     203         -1               volvo 244dl      gas
std
203     204         -1                 volvo 246   diesel
turbo
204     205         -1               volvo 264gl      gas
turbo

    doornumber      carbody drivewheel enginelocation  wheelbase  ...
\
0          two  convertible        rwd          front       88.6  ...
1          two  convertible        rwd          front       88.6  ...
2          two    hatchback        rwd          front       94.5  ...
3         four        sedan        fwd          front       99.8  ...
4         four        sedan        4wd          front       99.4  ...
```

```
..          ...           ...         ...             ...         ...  ...

200       four         sedan         rwd           front       109.1  ...

201       four         sedan         rwd           front       109.1  ...

202       four         sedan         rwd           front       109.1  ...

203       four         sedan         rwd           front       109.1  ...

204       four         sedan         rwd           front       109.1  ...


     enginesize  fuelsystem  boreratio  stroke compressionratio
horsepower  \
0           130        mpfi       3.47    2.68              9.0
111
1           130        mpfi       3.47    2.68              9.0
111
2           152        mpfi       2.68    3.47              9.0
154
3           109        mpfi       3.19    3.40             10.0
102
4           136        mpfi       3.19    3.40              8.0
115
..          ...         ...        ...     ...              ...
...
200         141        mpfi       3.78    3.15              9.5
114
201         141        mpfi       3.78    3.15              8.7
160
202         173        mpfi       3.58    2.87              8.8
134
203         145         idi       3.01    3.40             23.0
106
204         141        mpfi       3.78    3.15              9.5
114

     peakrpm citympg  highwaympg    price
0       5000      21          27  13495.0
1       5000      21          27  16500.0
2       5000      19          26  16500.0
3       5500      24          30  13950.0
4       5500      18          22  17450.0
..       ...     ...         ...      ...
200     5400      23          28  16845.0
201     5300      19          25  19045.0
202     5500      18          23  21485.0
203     4800      26          27  22470.0
```

```
204     5400      19           25  22625.0

[205 rows x 26 columns]
```

Current dataset consist of 205 rows and 26 columns

```
df.head()

   car_ID  symboling                   CarName fueltype aspiration
doornumber  \
0       1          3        alfa-romero giulia      gas        std
two
1       2          3       alfa-romero stelvio      gas        std
two
2       3          1  alfa-romero Quadrifoglio      gas        std
two
3       4          2               audi 100 ls      gas        std
four
4       5          2                audi 100ls      gas        std
four

       carbody drivewheel enginelocation  wheelbase  ...
enginesize  \
0  convertible        rwd          front       88.6  ...          130

1  convertible        rwd          front       88.6  ...          130

2    hatchback        rwd          front       94.5  ...          152

3        sedan        fwd          front       99.8  ...          109

4        sedan        4wd          front       99.4  ...          136


   fuelsystem  boreratio  stroke compressionratio horsepower  peakrpm
citympg  \
0        mpfi       3.47    2.68              9.0        111     5000
21
1        mpfi       3.47    2.68              9.0        111     5000
21
2        mpfi       2.68    3.47              9.0        154     5000
19
3        mpfi       3.19    3.40             10.0        102     5500
24
4        mpfi       3.19    3.40              8.0        115     5500
18

   highwaympg    price
0          27  13495.0
1          27  16500.0
```

```
2          26  16500.0
3          30  13950.0
4          22  17450.0

[5 rows x 26 columns]
```

```
df.tail()
```

```
     car_ID  symboling          CarName fueltype aspiration doornumber
\
200     201         -1  volvo 145e (sw)      gas        std       four

201     202         -1      volvo 144ea      gas      turbo       four

202     203         -1      volvo 244dl      gas        std       four

203     204         -1        volvo 246   diesel      turbo       four

204     205         -1      volvo 264gl      gas      turbo       four


    carbody drivewheel enginelocation  wheelbase  ...  enginesize
fuelsystem  \
200   sedan        rwd          front      109.1  ...         141
mpfi
201   sedan        rwd          front      109.1  ...         141
mpfi
202   sedan        rwd          front      109.1  ...         173
mpfi
203   sedan        rwd          front      109.1  ...         145
idi
204   sedan        rwd          front      109.1  ...         141
mpfi

     boreratio  stroke compressionratio horsepower  peakrpm citympg  \
200       3.78    3.15              9.5        114     5400      23
201       3.78    3.15              8.7        160     5300      19
202       3.58    2.87              8.8        134     5500      18
203       3.01    3.40             23.0        106     4800      26
204       3.78    3.15              9.5        114     5400      19

     highwaympg    price
200          28  16845.0
201          25  19045.0
202          23  21485.0
203          27  22470.0
204          25  22625.0

[5 rows x 26 columns]
```

```
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

df.describe()

           car_ID   symboling   wheelbase   carlength    carwidth
carheight  \
count  205.000000  205.000000  205.000000  205.000000  205.000000
205.000000
mean   103.000000    0.834146   98.756585  174.049268   65.907805
53.724878
std     59.322565    1.245307    6.021776   12.337289    2.145204
2.443522
min      1.000000   -2.000000   86.600000  141.100000   60.300000
47.800000
25%     52.000000    0.000000   94.500000  166.300000   64.100000
52.000000
50%    103.000000    1.000000   97.000000  173.200000   65.500000
54.100000
```

```
75%    154.000000    2.000000  102.400000  183.100000   66.900000
55.500000
max    205.000000    3.000000  120.900000  208.100000   72.300000
59.800000

        curbweight  enginesize   boreratio      stroke
compressionratio  \
count   205.000000  205.000000  205.000000  205.000000
205.000000
mean   2555.565854  126.907317    3.329756    3.255415
10.142537
std     520.680204   41.642693    0.270844    0.313597
3.972040
min    1488.000000   61.000000    2.540000    2.070000
7.000000
25%    2145.000000   97.000000    3.150000    3.110000
8.600000
50%    2414.000000  120.000000    3.310000    3.290000
9.000000
75%    2935.000000  141.000000    3.580000    3.410000
9.400000
max    4066.000000  326.000000    3.940000    4.170000
23.000000

       horsepower      peakrpm     citympg  highwaympg         price
count  205.000000   205.000000  205.000000  205.000000    205.000000
mean   104.117073  5125.121951   25.219512   30.751220  13276.710571
std     39.544167   476.985643    6.542142    6.886443   7988.852332
min     48.000000  4150.000000   13.000000   16.000000   5118.000000
25%     70.000000  4800.000000   19.000000   25.000000   7788.000000
50%     95.000000  5200.000000   24.000000   30.000000  10295.000000
75%    116.000000  5500.000000   30.000000   34.000000  16503.000000
max    288.000000  6600.000000   49.000000   54.000000  45400.000000

df.dtypes

car_ID              int64
symboling           int64
CarName            object
fueltype           object
aspiration         object
doornumber         object
carbody            object
drivewheel         object
enginelocation     object
wheelbase         float64
carlength         float64
carwidth          float64
carheight         float64
curbweight          int64
```

```
enginetype           object
cylindernumber       object
enginesize            int64
fuelsystem           object
boreratio           float64
stroke              float64
compressionratio    float64
horsepower            int64
peakrpm               int64
citympg               int64
highwaympg            int64
price               float64
dtype: object
```

8 columns are float datatype, 8 columns are integer datatype and 10 columns are object datatype.

```
df.columns
```

```
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')
```

Dropping duplicate rows

```
df.drop_duplicates()
```

```
     car_ID  symboling                   CarName fueltype
aspiration  \
0         1          3        alfa-romero giulia      gas        std

1         2          3       alfa-romero stelvio      gas        std

2         3          1  alfa-romero Quadrifoglio      gas        std

3         4          2               audi 100 ls      gas        std

4         5          2                audi 100ls      gas        std

..      ...        ...                       ...      ...        ...

200     201         -1           volvo 145e (sw)      gas        std
```

```
201     202         -1               volvo 144ea      gas      turbo
202     203         -1               volvo 244dl      gas        std
203     204         -1                 volvo 246   diesel      turbo
204     205         -1               volvo 264gl      gas      turbo

    doornumber      carbody drivewheel enginelocation  wheelbase  ...
\
0          two  convertible        rwd          front       88.6  ...
1          two  convertible        rwd          front       88.6  ...
2          two    hatchback        rwd          front       94.5  ...
3         four        sedan        fwd          front       99.8  ...
4         four        sedan        4wd          front       99.4  ...
..         ...          ...        ...            ...        ...  ...
200       four        sedan        rwd          front      109.1  ...
201       four        sedan        rwd          front      109.1  ...
202       four        sedan        rwd          front      109.1  ...
203       four        sedan        rwd          front      109.1  ...
204       four        sedan        rwd          front      109.1  ...

     enginesize  fuelsystem  boreratio  stroke compressionratio
horsepower  \
0           130        mpfi       3.47    2.68              9.0
111
1           130        mpfi       3.47    2.68              9.0
111
2           152        mpfi       2.68    3.47              9.0
154
3           109        mpfi       3.19    3.40             10.0
102
4           136        mpfi       3.19    3.40              8.0
115
..          ...         ...        ...     ...              ...
...
200         141        mpfi       3.78    3.15              9.5
114
```

```
201          141         mpfi       3.78    3.15                8.7
160
202          173         mpfi       3.58    2.87                8.8
134
203          145          idi       3.01    3.40               23.0
106
204          141         mpfi       3.78    3.15                9.5
114

     peakrpm citympg  highwaympg    price
0       5000      21          27  13495.0
1       5000      21          27  16500.0
2       5000      19          26  16500.0
3       5500      24          30  13950.0
4       5500      18          22  17450.0
..       ...     ...         ...      ...
200     5400      23          28  16845.0
201     5300      19          25  19045.0
202     5500      18          23  21485.0
203     4800      26          27  22470.0
204     5400      19          25  22625.0

[205 rows x 26 columns]
```

Check missing values

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

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
```

There are no missing values in the dataset.

Check the 'CarName' feature

```
df['CarName'].unique()
```

```
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)
```

Check 'fueltype' feature

```
df['fueltype'].unique()
```

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

Check 'aspiration' feature

```
df['aspiration'].unique()
```

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

Check 'doornumber' feature

```
df['doornumber'].unique()
```

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

Check 'carbody' feature

```
df['carbody'].unique()
```

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

Check 'drivewheel' feature

```
df['drivewheel'].unique()
```

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

Check 'enginelocation' feature

```
df['enginelocation'].unique()
```

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

Check 'enginetype' feature

```
df['enginetype'].unique()
```

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

Check 'cylindernumber' feature

```
df['cylindernumber'].unique()
```

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

Check 'fuelsystem' feature

```
df['fuelsystem'].unique()
```

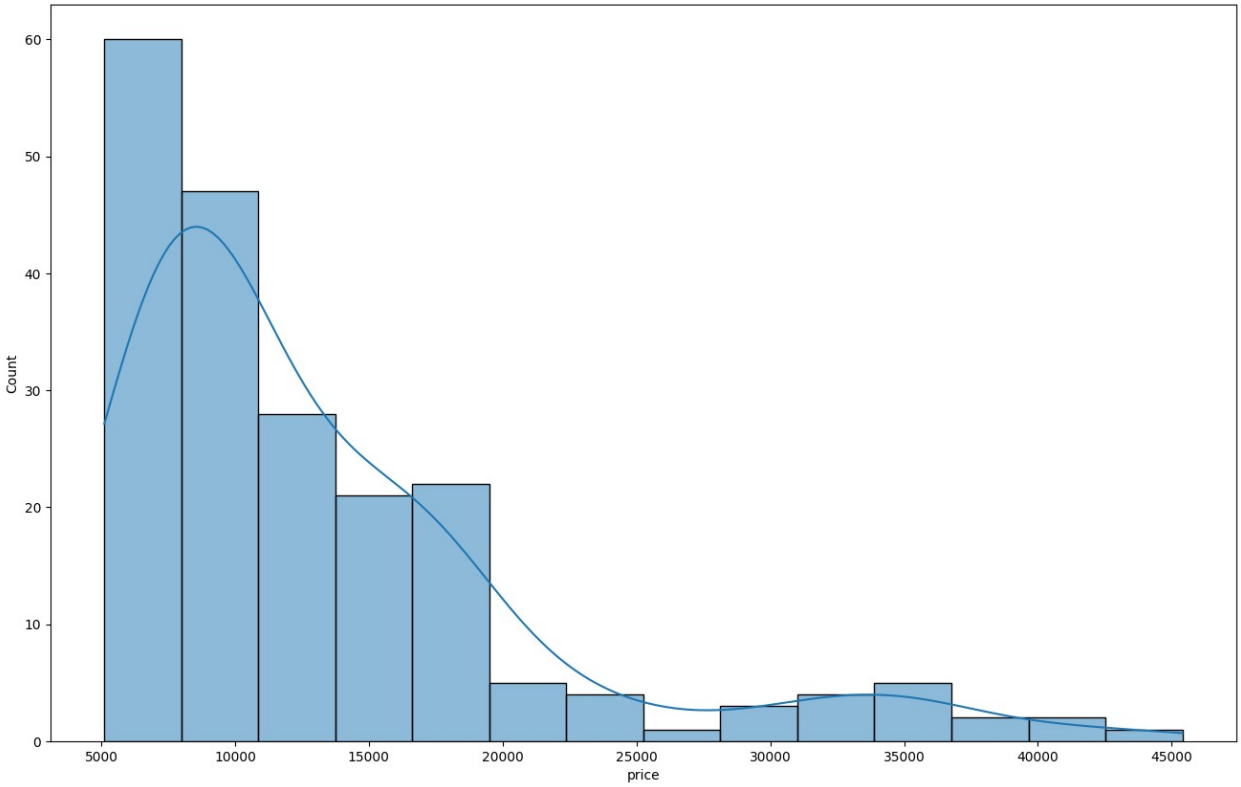
```
array(['mpfi', '2bbl', 'mfi', '1bbl', 'spfi', '4bbl', 'idi', 'spdi'],
      dtype=object)
```

Data visualizations

Check the distribution of car price

```
plt.figure(figsize=(16, 10))
sns.histplot(data=df, x='price', kde=True)
```

```
<Axes: xlabel='price', ylabel='Count'>
```




```
df['price'].describe()
```

```
count      205.000000
mean     13276.710571
std       7988.852332
min       5118.000000
25%       7788.000000
50%      10295.000000
75%      16503.000000
max      45400.000000
Name: price, dtype: float64
```

The plot seemed to be right-skewed, meaning that the most prices in the dataset are low(Below 15,000).

There is a significant difference between the mean and the median of the price distribution.

Minimum price is $5118

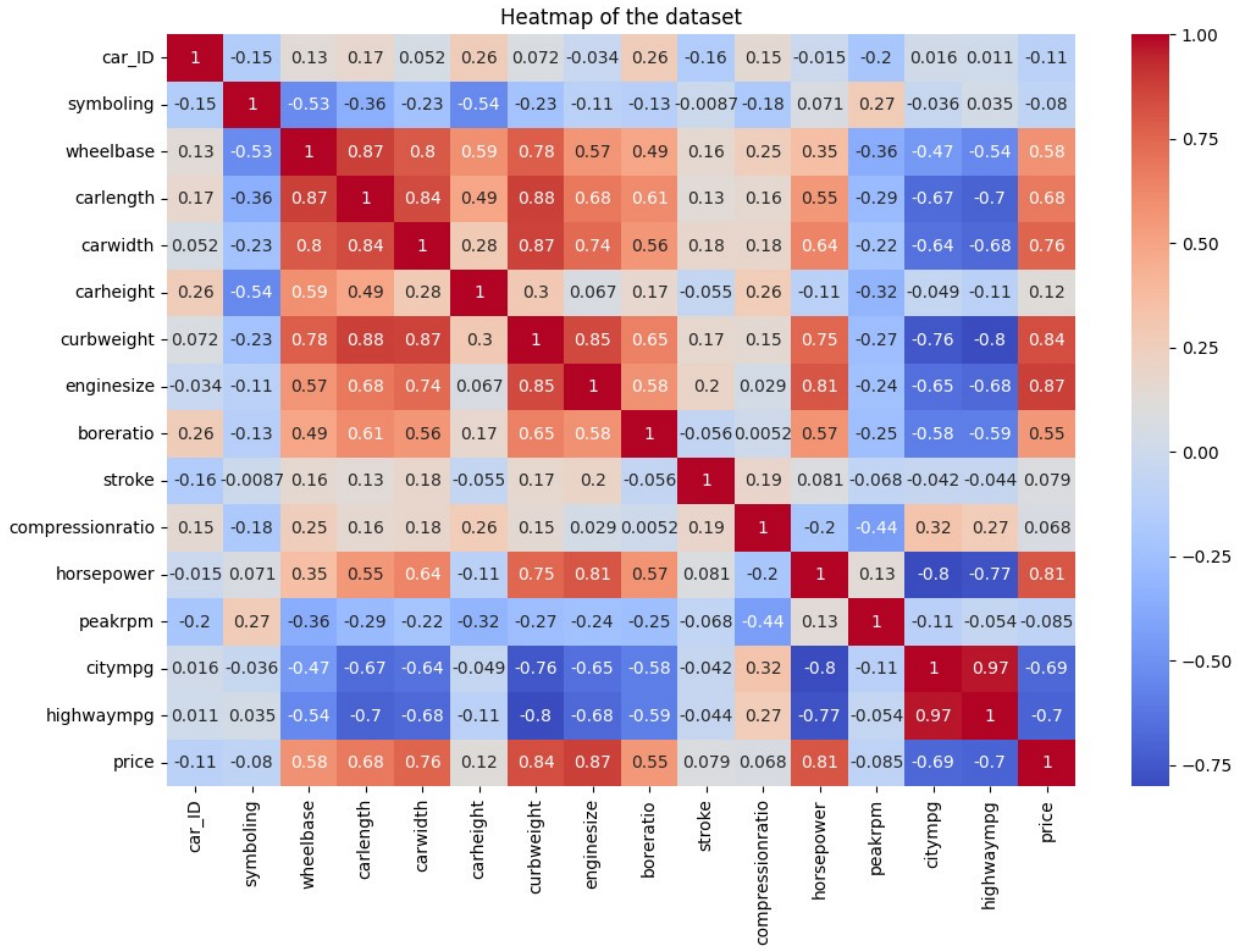
Maximum price is $5400

Data correlation using Heatmap

Multivariate analysis

```
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(),annot=True,cmap='coolwarm')
plt.title('Heatmap of the dataset')
```

```
<ipython-input-154-5daefb561f69>:2: FutureWarning: The default value
of numeric_only in DataFrame.corr is deprecated. In a future version,
it will default to False. Select only valid columns or specify the
value of numeric_only to silence this warning.
  sns.heatmap(df.corr(),annot=True,cmap='coolwarm')

Text(0.5, 1.0, 'Heatmap of the dataset')
```



Highly correlated variables to price are - carwidth, curbweght, enginesize and horsepower.

Visualising Categorical Data¶

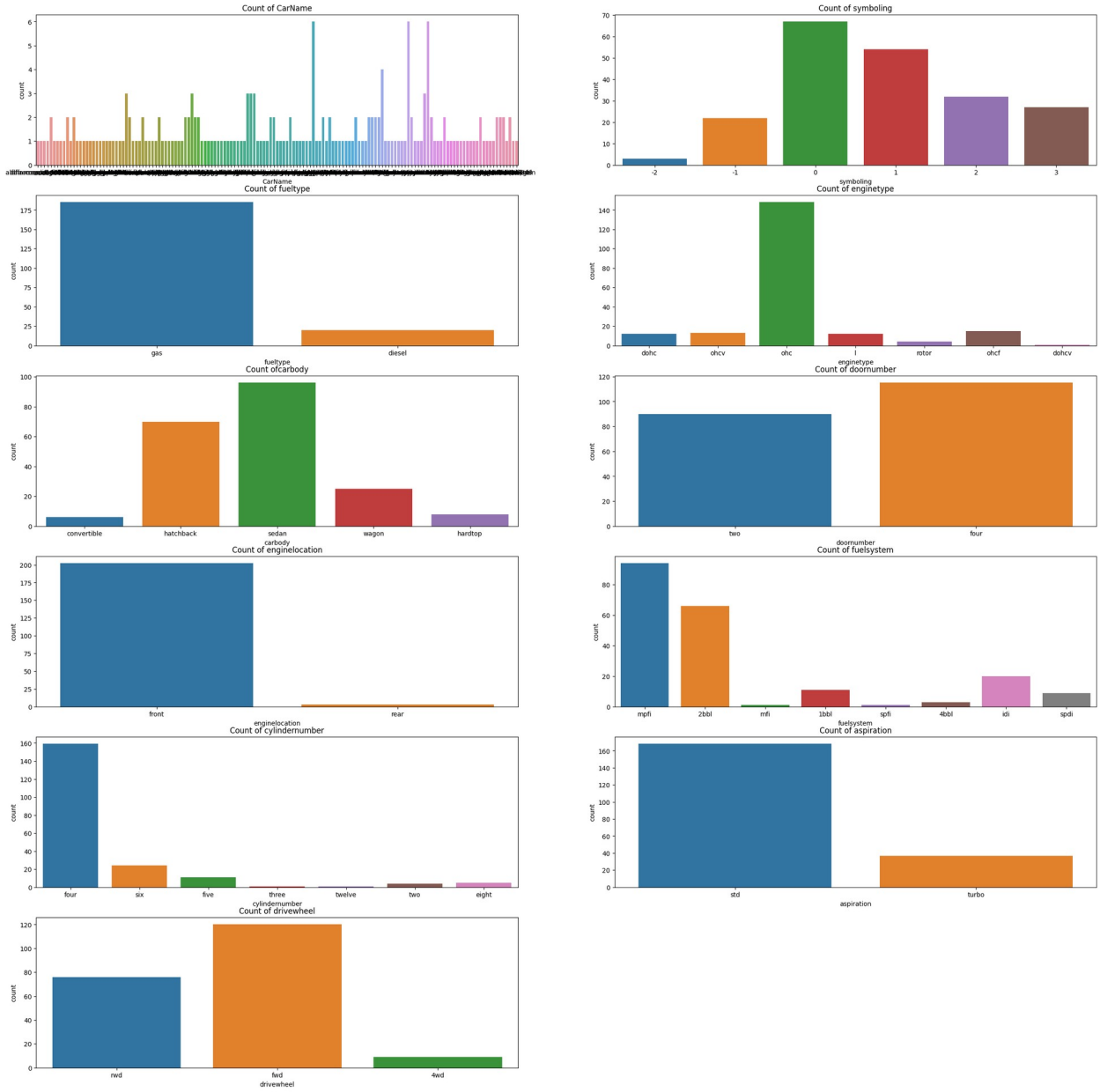
- CompanyName
- fueltype
- enginetype
- carbody
- doornumber

- enginelocation
- fuelsystem
- cylindernumber
- aspiration
- drivewheel

Visualising categorical data

```
plt.figure(figsize=(30,30))

plt.subplot(6,2,1)
sns.countplot(x='CarName',data=df)
plt.title('Count of CarName')

plt.subplot(6,2,2)
sns.countplot(x='symboling',data=df)
plt.title('Count of symboling')

plt.subplot(6,2,3)
sns.countplot(x='fueltype',data=df)
plt.title('Count of fueltype')

plt.subplot(6,2,4)
sns.countplot(x='enginetype',data=df)
plt.title('Count of enginetype')

plt.subplot(6,2,5)
sns.countplot(x='carbody',data=df)
plt.title('Count ofcarbody')

plt.subplot(6,2,6)
sns.countplot(x='doornumber',data=df)
plt.title('Count of doornumber')

plt.subplot(6,2,7)
sns.countplot(x='enginelocation',data=df)
plt.title('Count of enginelocation')

plt.subplot(6,2,8)
sns.countplot(x='fuelsystem',data=df)
plt.title('Count of fuelsystem')

plt.subplot(6,2,9)
sns.countplot(x='cylindernumber',data=df)
plt.title('Count of cylindernumber')

plt.subplot(6,2,10)
sns.countplot(x='aspiration',data=df)
plt.title('Count of aspiration')
```

```
plt.subplot(6,2,11)
sns.countplot(x='drivewheel',data=df)
plt.title('Count of drivewheel')

Text(0.5, 1.0, 'Count of drivewheel')
```



Number of gas fueled cars are more than diesel.

sedan is the top car type prefered.

It seems that the symboling with 0 and 1 values have high number of rows (i.e. They are most sold.)

ohc Engine type seems to be most favored type.

We can see most of the cars use gas as fuel.

FWD is the most favored, and the second place is RWD followed by 4WD.

Most of the cars' engine location are front.

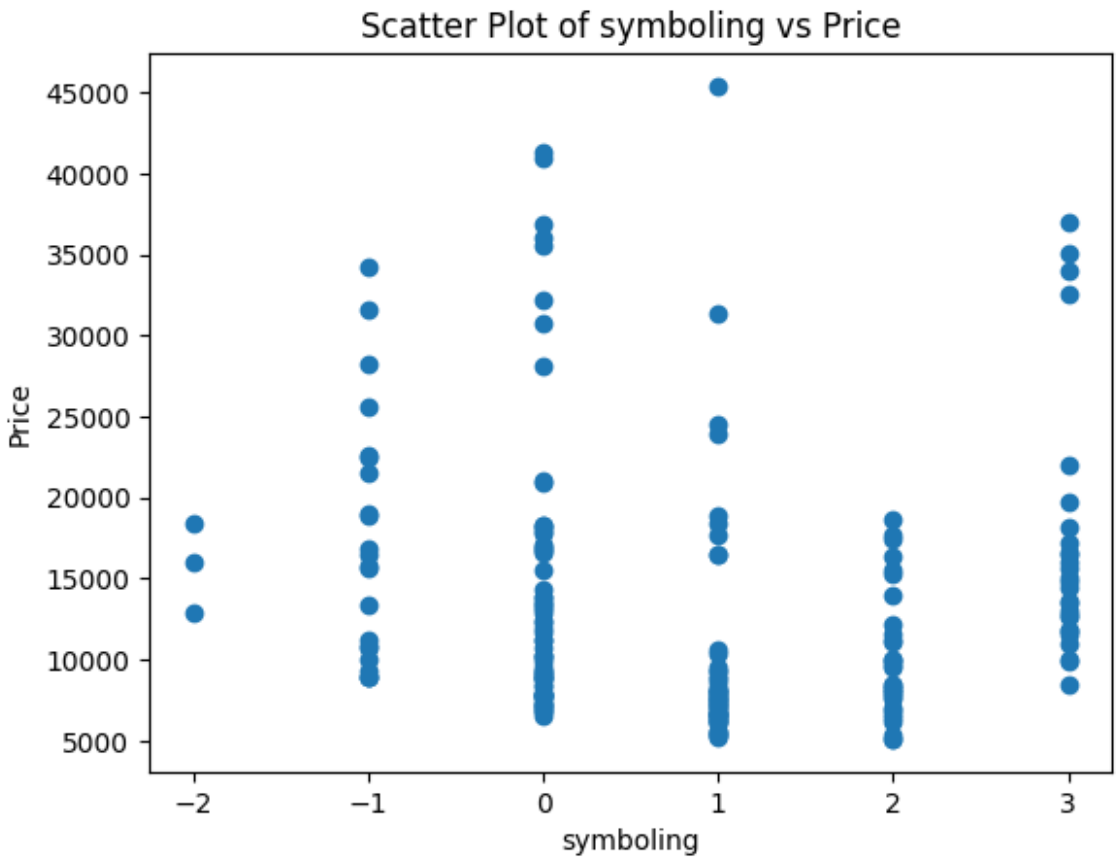
Most of the cars are four-cylinder.

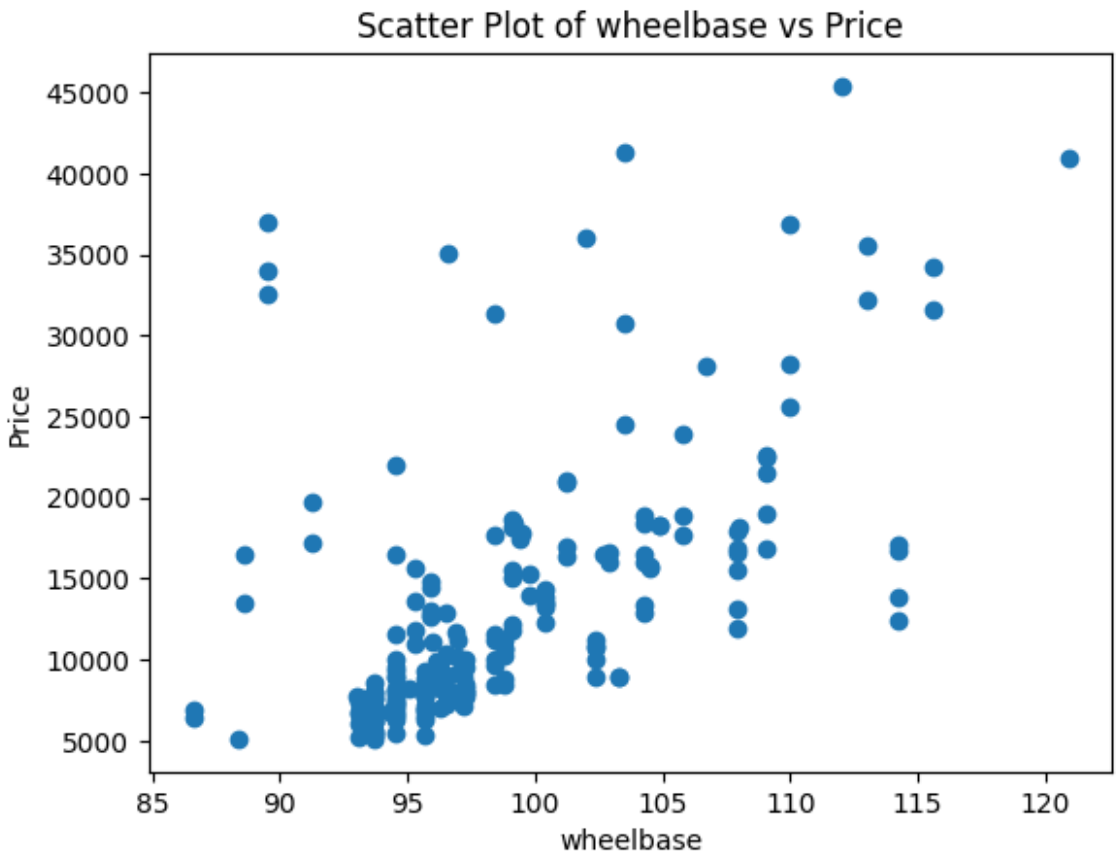
MPFI is the favored type of fuel system.

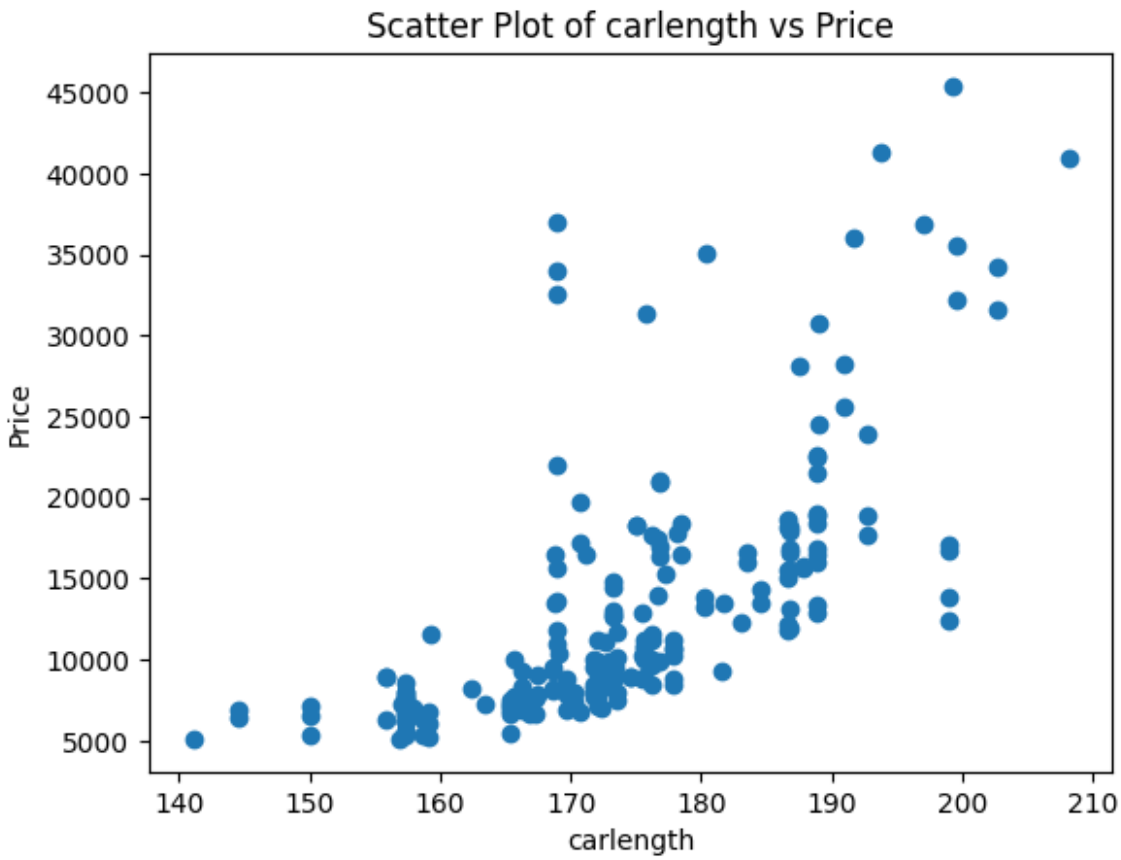
Visualising numerical variable

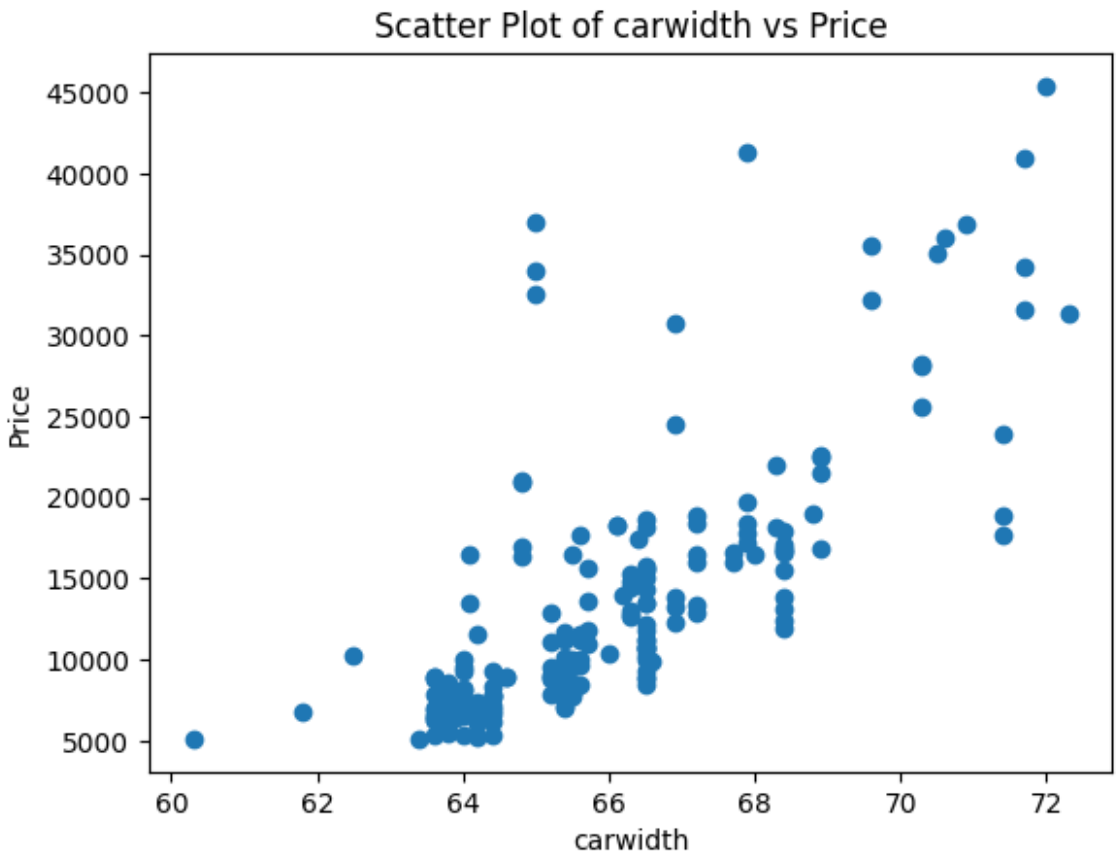
symboling
wheelbase
carlength
carwidth
carheight
curbweight
enginesize
boreratio
stroke
compressionratio
horsepower
peakrpm
citympg
highwaympg

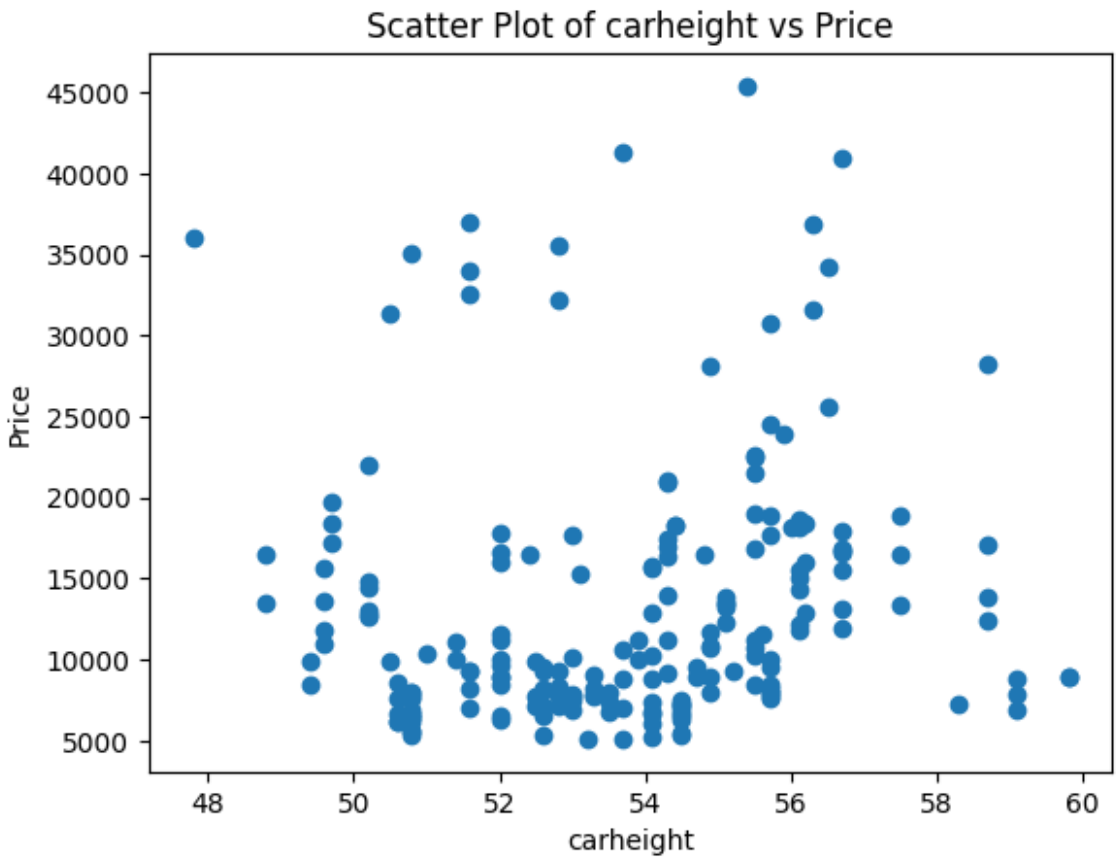
```
x=['symboling','wheelbase','carlength','carwidth','carheight','curbwei
ght','enginesize','boreratio','stroke','compressionratio','horsepower'
,'peakrpm','citympg','highwaympg']
for variable in x:
    plt.scatter(df[variable], df['price'])
    plt.xlabel(variable)
    plt.ylabel('Price')
    plt.title(f'Scatter Plot of {variable} vs Price')
    plt.show()
```

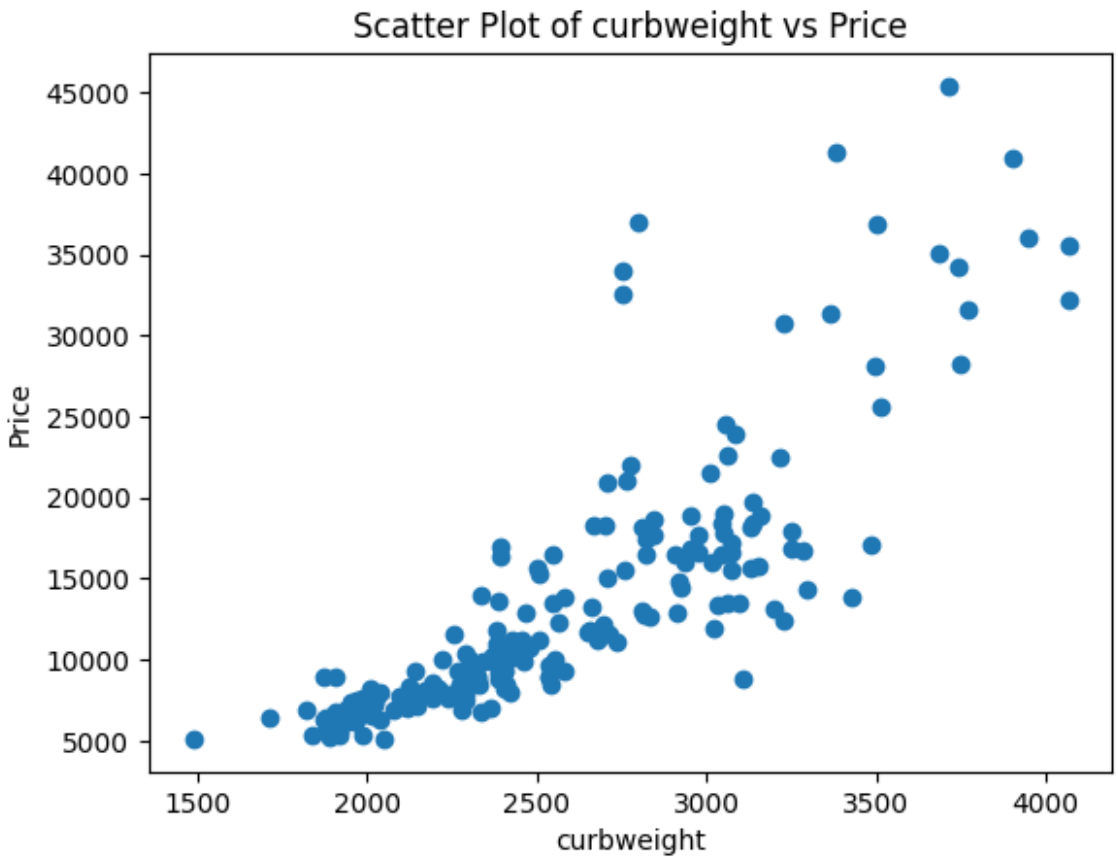

Scatter Plot of symboling vs Price
Price
45000
40000
35000
30000
25000
20000
15000
10000
5000
−2
−1
0
1
2
3
symboling

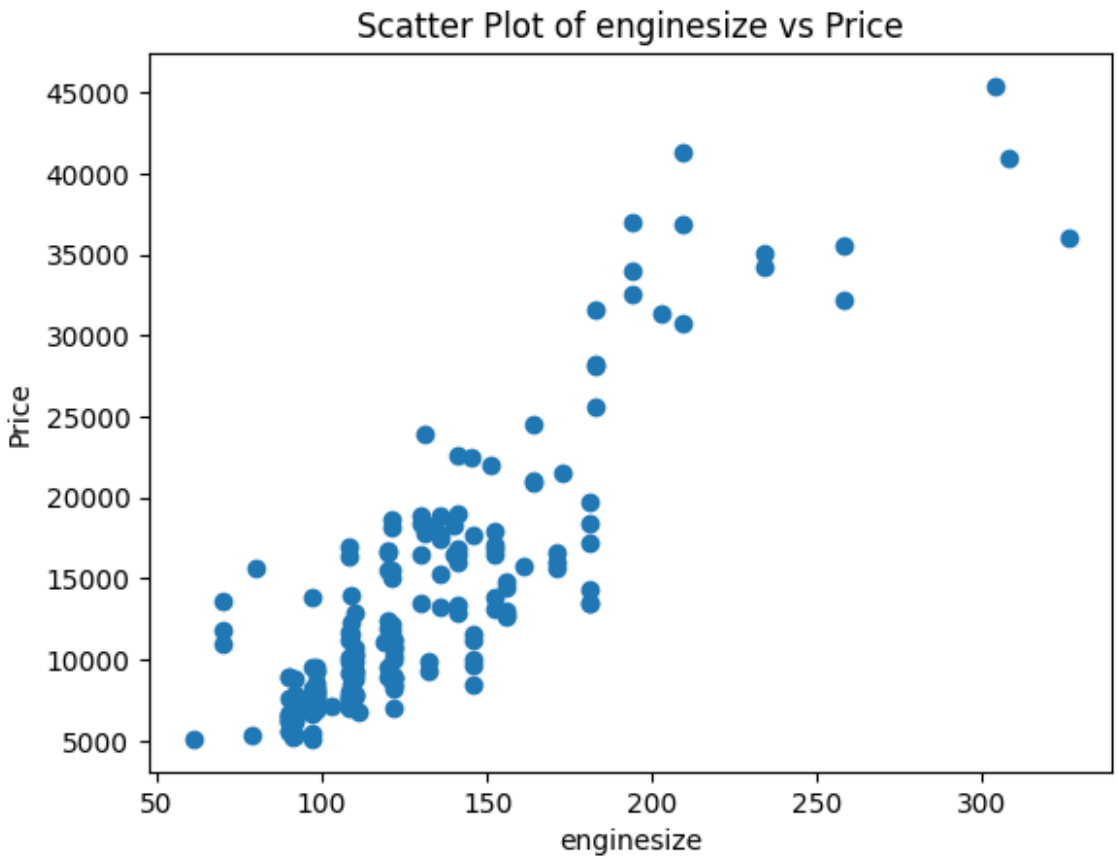

Scatter Plot of wheelbase vs Price
Price
wheelbase
45000
40000
35000
30000
25000
20000
15000
10000
5000
85
90
95
100
105
110
115
120

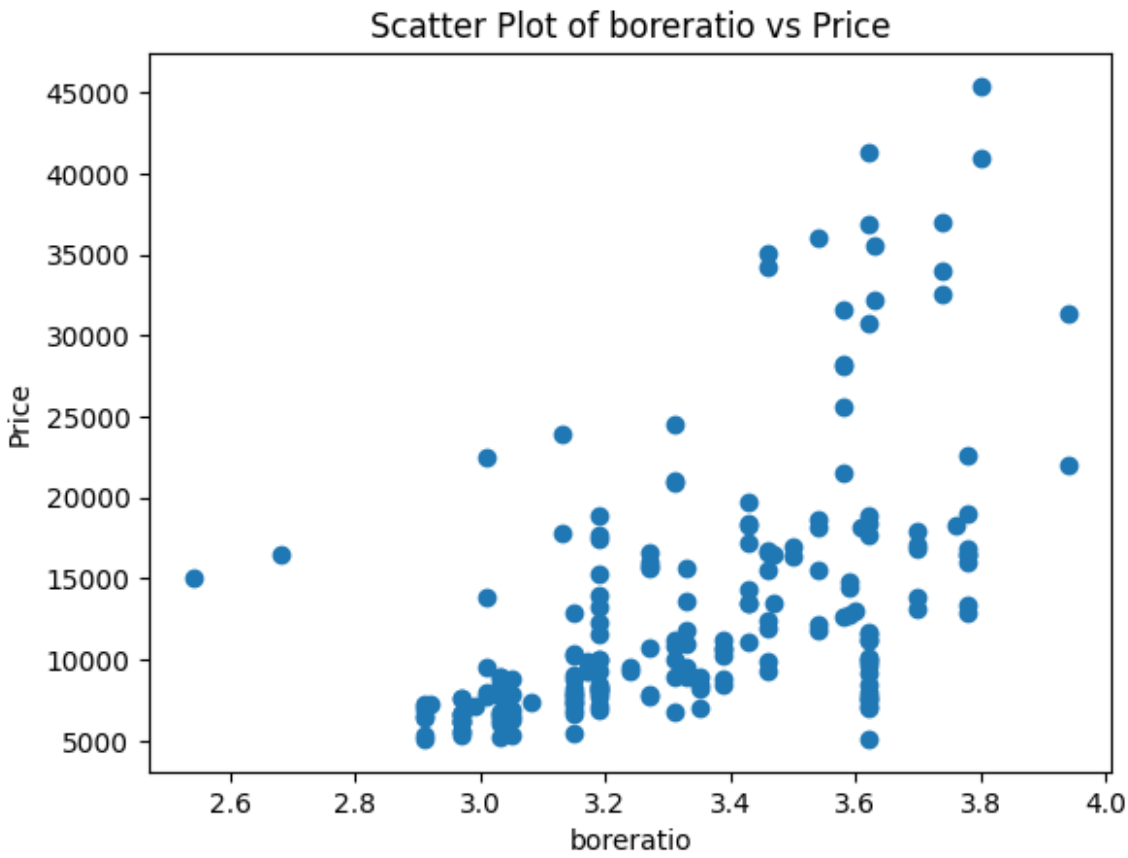

Scatter Plot of carlength vs Price
Price
carlength
45000
40000
35000
30000
25000
20000
15000
10000
5000
140
150
160
170
180
190
200
210

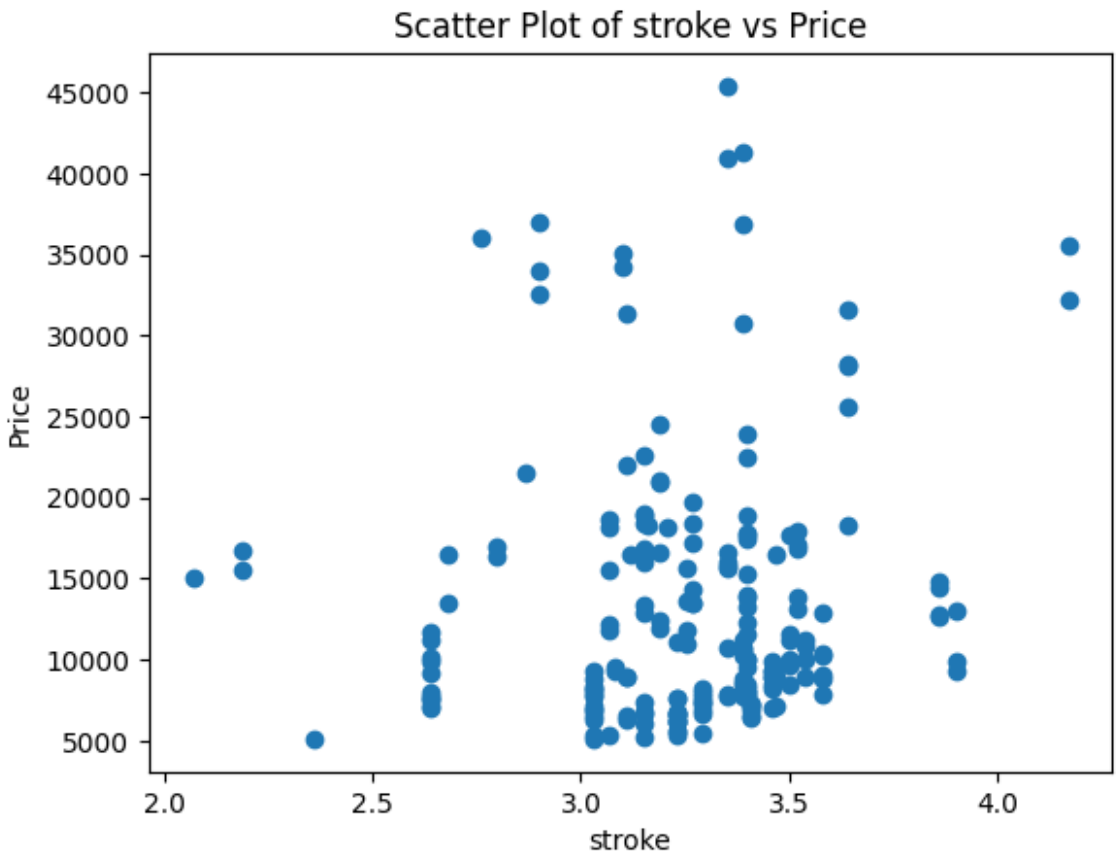

Scatter Plot of carwidth vs Price
Price
carwidth
45000
40000
35000
30000
25000
20000
15000
10000
5000
60
62
64
66
68
70
72

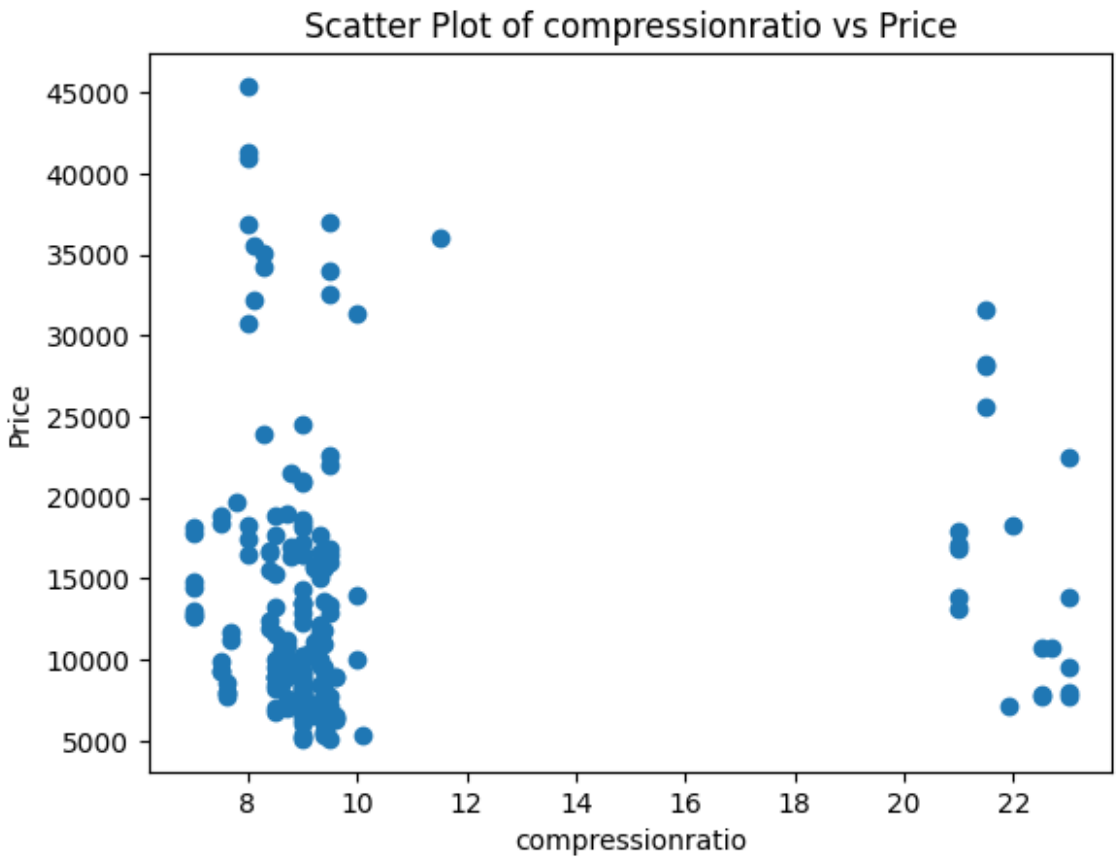

Scatter Plot of carheight vs Price
Price
carheight
45000
40000
35000
30000
25000
20000
15000
10000
5000
48
50
52
54
56
58
60

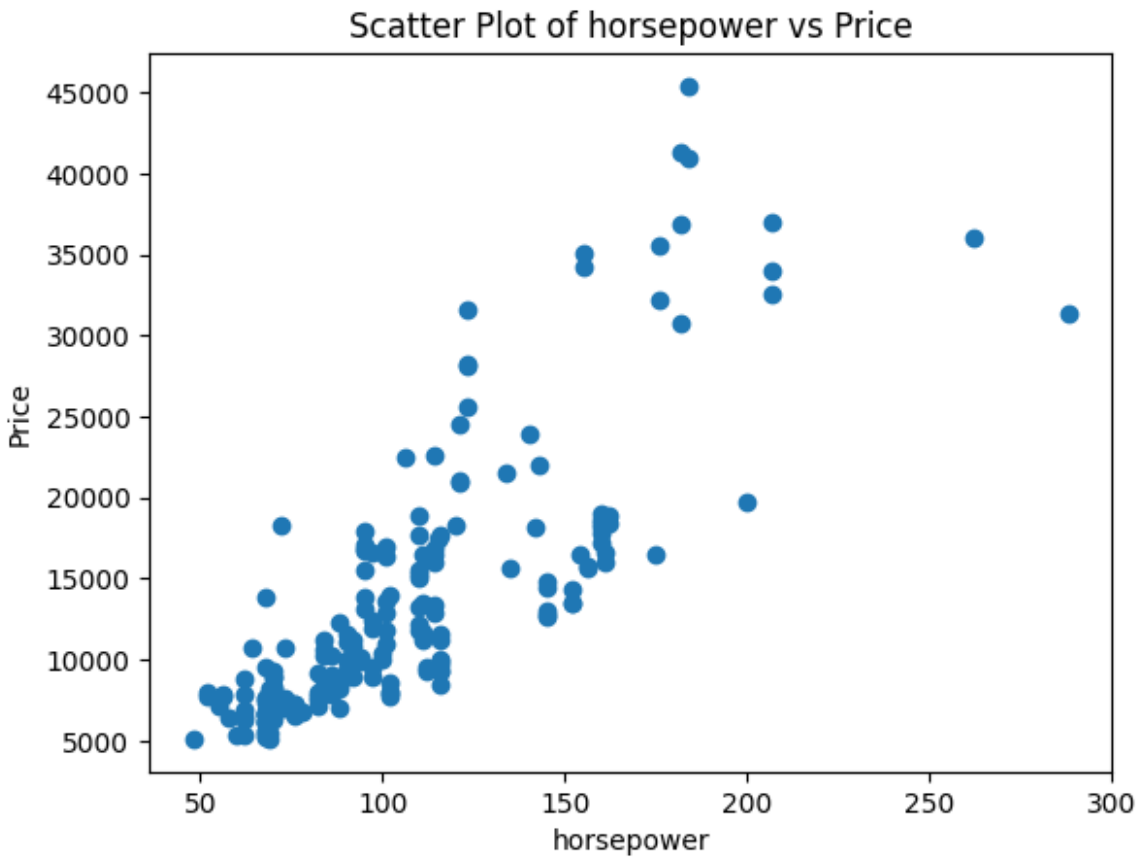

Scatter Plot of curbweight vs Price
Price
curbweight
45000
40000
35000
30000
25000
20000
15000
10000
5000
1500
2000
2500
3000
3500
4000

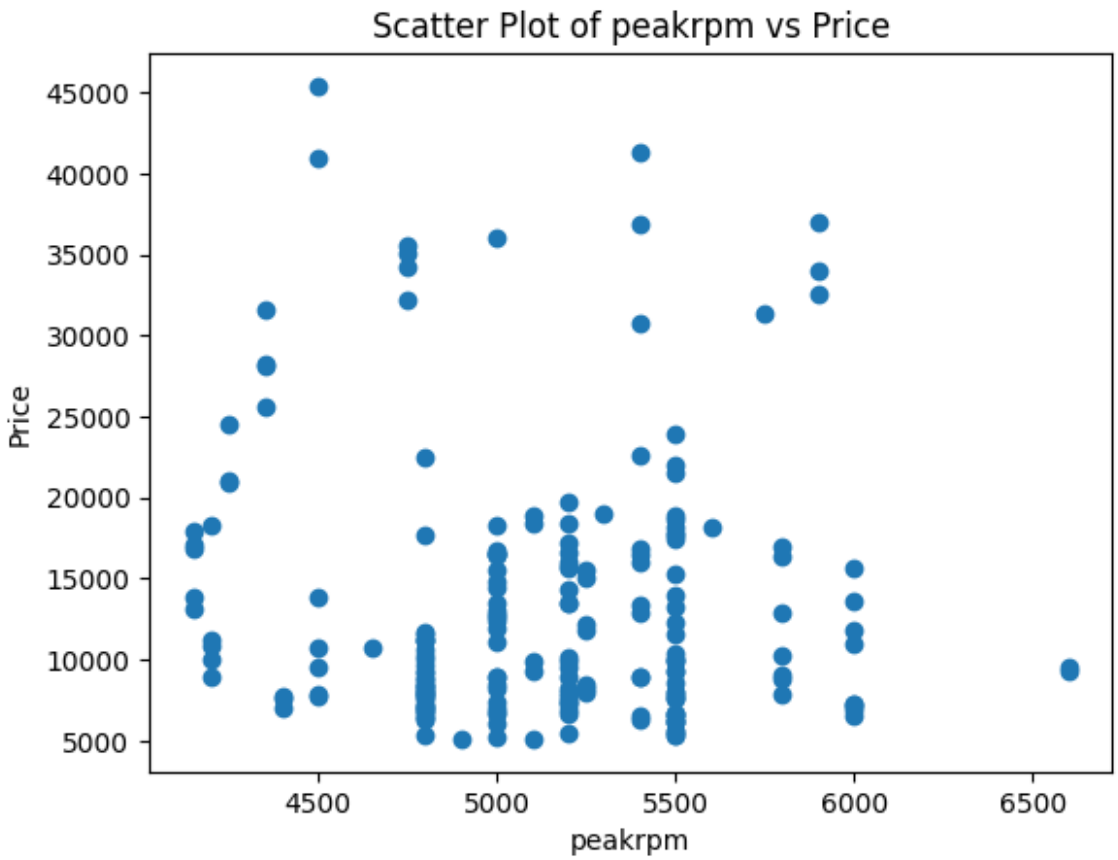

Scatter Plot of enginesize vs Price
Price
enginesize
45000
40000
35000
30000
25000
20000
15000
10000
5000
50
100
150
200
250
300

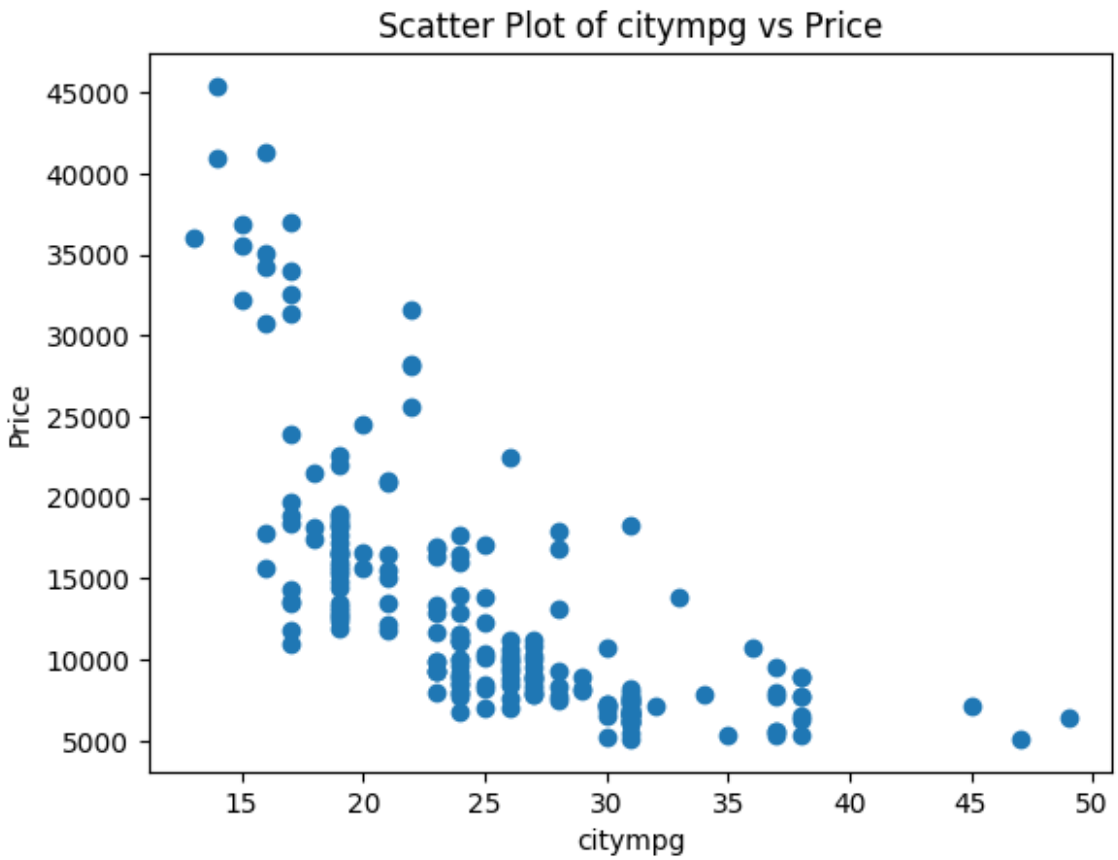

Scatter Plot of boreratio vs Price
Price
boreratio
45000
40000
35000
30000
25000
20000
15000
10000
5000
2.6
2.8
3.0
3.2
3.4
3.6
3.8
4.0

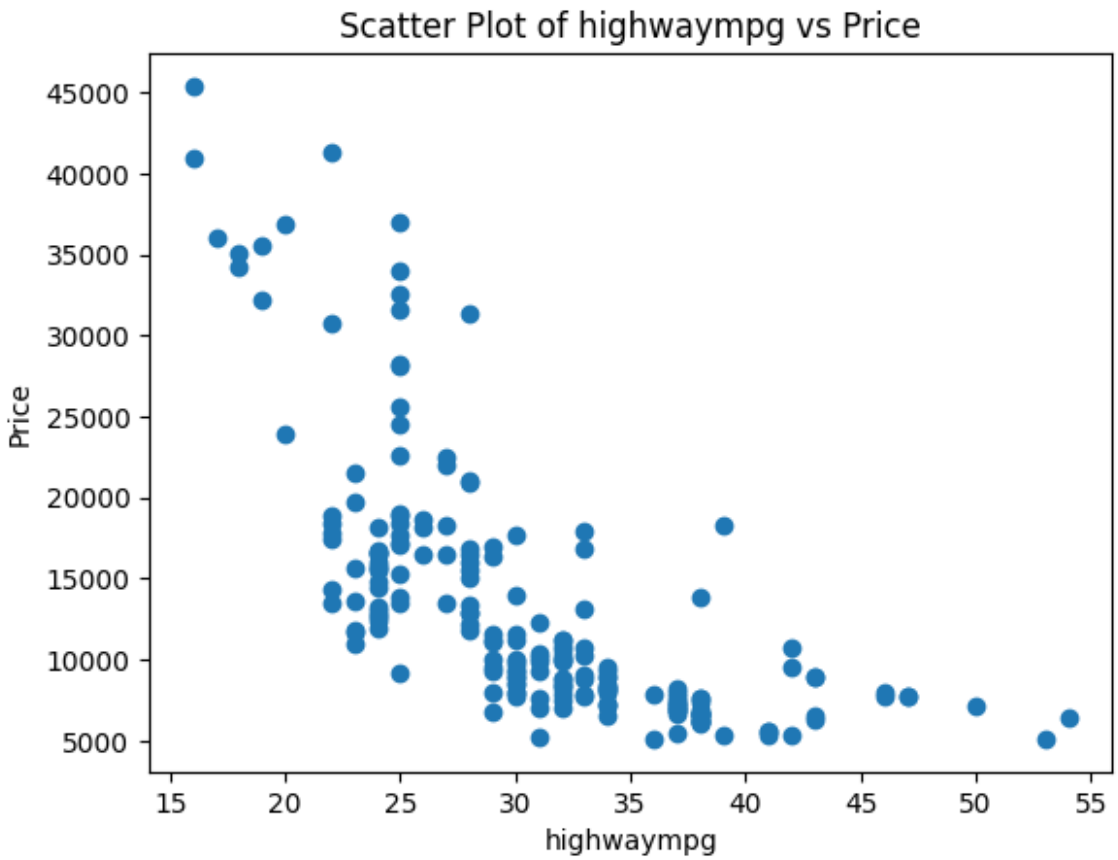

Scatter Plot of stroke vs Price
Price
stroke
45000
40000
35000
30000
25000
20000
15000
10000
5000
2.0
2.5
3.0
3.5
4.0


Scatter Plot of compressionratio vs Price
Price
45000
40000
35000
30000
25000
20000
15000
10000
5000
8
10
12
14
16
18
20
22
compressionratio


Scatter Plot of horsepower vs Price
Price
horsepower
45000
40000
35000
30000
25000
20000
15000
10000
5000
50
100
150
200
250
300


Scatter Plot of peakrpm vs Price
Price
peakrpm
45000
40000
35000
30000
25000
20000
15000
10000
5000
4500
5000
5500
6000
6500


Scatter Plot of citympg vs Price
Price
citympg
45000
40000
35000
30000
25000
20000
15000
10000
5000
15
20
25
30
35
40
45
50



carwidth, carlength and curbweight seems to have a poitive correlation with price.

carheight doesn't show any significant trend with price.

enginesize, boreratio, horsepower, wheelbase - seem to have a significant positive correlation with price.

citympg, highwaympg - seem to have a significant negative correlation with price.

Dropping unwanted column

```
df.drop(['car_ID'],axis=1,inplace=True)
df
```

```
     symboling                   CarName fueltype aspiration
doornumber  \
0            3        alfa-romero giulia      gas        std
two
1            3       alfa-romero stelvio      gas        std
two
2            1  alfa-romero Quadrifoglio      gas        std
two
3            2               audi 100 ls      gas        std
four
```

```
4            2                 audi 100ls      gas        std
four
..         ...                        ...      ...        ...        ..
.
200         -1           volvo 145e (sw)      gas        std
four
201         -1               volvo 144ea      gas      turbo
four
202         -1               volvo 244dl      gas        std
four
203         -1                 volvo 246   diesel      turbo
four
204         -1               volvo 264gl      gas      turbo
four

         carbody drivewheel enginelocation  wheelbase  carlength  ...
\
0    convertible        rwd           front       88.6      168.8  ...

1    convertible        rwd           front       88.6      168.8  ...

2      hatchback        rwd           front       94.5      171.2  ...

3          sedan        fwd           front       99.8      176.6  ...

4          sedan        4wd           front       99.4      176.6  ...

..           ...        ...             ...        ...        ...  ...

200        sedan        rwd           front      109.1      188.8  ...

201        sedan        rwd           front      109.1      188.8  ...

202        sedan        rwd           front      109.1      188.8  ...

203        sedan        rwd           front      109.1      188.8  ...

204        sedan        rwd           front      109.1      188.8  ...


     enginesize  fuelsystem  boreratio stroke compressionratio
horsepower  \
0           130        mpfi       3.47   2.68              9.0
111
1           130        mpfi       3.47   2.68              9.0
111
2           152        mpfi       2.68   3.47              9.0
154
3           109        mpfi       3.19   3.40             10.0
102
```

```
4           136        mpfi       3.19   3.40               8.0
115
..          ...         ...        ...    ...               ...
...
200         141        mpfi       3.78   3.15               9.5
114
201         141        mpfi       3.78   3.15               8.7
160
202         173        mpfi       3.58   2.87               8.8
134
203         145         idi       3.01   3.40              23.0
106
204         141        mpfi       3.78   3.15               9.5
114

    peakrpm  citympg  highwaympg    price
0      5000       21          27  13495.0
1      5000       21          27  16500.0
2      5000       19          26  16500.0
3      5500       24          30  13950.0
4      5500       18          22  17450.0
..      ...      ...         ...      ...
200    5400       23          28  16845.0
201    5300       19          25  19045.0
202    5500       18          23  21485.0
203    4800       26          27  22470.0
204    5400       19          25  22625.0

[205 rows x 25 columns]
```

Encoding using LabelEncoder

```
from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()
df['CarName']=encoder.fit_transform(df['CarName'])
df['fueltype']=encoder.fit_transform(df['fueltype'])
df['aspiration']=encoder.fit_transform(df['aspiration'])
df['doornumber']=encoder.fit_transform(df['doornumber'])
df['carbody']=encoder.fit_transform(df['carbody'])
df['drivewheel']=encoder.fit_transform(df['drivewheel'])
df['enginelocation']=encoder.fit_transform(df['enginelocation'])
df['enginetype']=encoder.fit_transform(df['enginetype'])
df['cylindernumber']=encoder.fit_transform(df['cylindernumber'])
df['fuelsystem']=encoder.fit_transform(df['fuelsystem'])

df.dtypes
```

```
symboling             int64
CarName               int64
fueltype              int64
```

```
aspiration            int64
doornumber            int64
carbody               int64
drivewheel            int64
enginelocation        int64
wheelbase           float64
carlength           float64
carwidth            float64
carheight           float64
curbweight            int64
enginetype            int64
cylindernumber        int64
enginesize            int64
fuelsystem            int64
boreratio           float64
stroke              float64
compressionratio    float64
horsepower            int64
peakrpm               int64
citympg               int64
highwaympg            int64
price               float64
dtype: object
```

df

	symboling	CarName	fueltype	aspiration	doornumber	carbody	\
0	3	2	1	0	1	0	
1	3	3	1	0	1	0	
2	1	1	1	0	1	2	
3	2	4	1	0	0	3	
4	2	5	1	0	0	3	
..	...	...	...	...	...	...	
200	-1	139	1	0	0	3	
201	-1	138	1	1	0	3	
202	-1	140	1	0	0	3	
203	-1	142	0	1	0	3	
204	-1	143	1	1	0	3	

	drivewheel	enginelocation	wheelbase	carlength	...	enginesize
\						
0	2	0	88.6	168.8	...	130
1	2	0	88.6	168.8	...	130
2	2	0	94.5	171.2	...	152
3	1	0	99.8	176.6	...	109
4	0	0	99.4	176.6	...	136

```
..          ...              ...        ...        ...  ...         ...

200           2                0      109.1      188.8  ...         141

201           2                0      109.1      188.8  ...         141

202           2                0      109.1      188.8  ...         173

203           2                0      109.1      188.8  ...         145

204           2                0      109.1      188.8  ...         141


     fuelsystem  boreratio  stroke  compressionratio  horsepower
peakrpm  \
0             5       3.47    2.68               9.0         111
5000
1             5       3.47    2.68               9.0         111
5000
2             5       2.68    3.47               9.0         154
5000
3             5       3.19    3.40              10.0         102
5500
4             5       3.19    3.40               8.0         115
5500
..          ...        ...     ...               ...         ...
...
200           5       3.78    3.15               9.5         114
5400
201           5       3.78    3.15               8.7         160
5300
202           5       3.58    2.87               8.8         134
5500
203           3       3.01    3.40              23.0         106
4800
204           5       3.78    3.15               9.5         114
5400

     citympg  highwaympg    price
0         21          27  13495.0
1         21          27  16500.0
2         19          26  16500.0
3         24          30  13950.0
4         18          22  17450.0
..       ...         ...      ...
200       23          28  16845.0
201       19          25  19045.0
202       18          23  21485.0
203       26          27  22470.0
```

```
204       19          25  22625.0

[205 rows x 25 columns]
```

Separate the data into input and output data

```
x=df.iloc[:,:-1].values
y=df.iloc[:,-1].values
y
```

```
array([13495.   , 16500.   , 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.
,
        7295.   ,  7895.   ,  9095.   ,  8845.   , 10295.   , 12945.
,
       10345.   ,  6785.   ,  8916.5  ,  8916.5  , 11048.   , 32250.
,
       35550.   , 36000.   ,  5195.   ,  6095.   ,  6795.   ,  6695.
,
        7395.   , 10945.   , 11845.   , 13645.   , 15645.   ,  8845.
,
        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.   ,  9279.   ,  5499.
,
        7099.   ,  6649.   ,  6849.   ,  7349.   ,  7299.   ,  7799.
,
        7499.   ,  7999.   ,  8249.   ,  8949.   ,  9549.   , 13499.
,
       14399.   , 13499.   , 17199.   , 19699.   , 18399.   , 11900.
,
       13200.   , 12440.   , 13860.   , 15580.   , 16900.   , 16695.
,
       17075.   , 16630.   , 17950.   , 18150.   ,  5572.   ,  7957.
```

```
,
       6229.   ,  6692.   ,  7609.   ,  8921.   , 12764.   , 22018.
,
      32528.   , 34028.   , 37028.   , 31400.5  ,  9295.   ,  9895.
,
      11850.   , 12170.   , 15040.   , 15510.   , 18150.   , 18620.
,
       5118.   ,  7053.   ,  7603.   ,  7126.   ,  7775.   ,  9960.
,
       9233.   , 11259.   ,  7463.   , 10198.   ,  8013.   , 11694.
,
       5348.   ,  6338.   ,  6488.   ,  6918.   ,  7898.   ,  8778.
,
       6938.   ,  7198.   ,  7898.   ,  7788.   ,  7738.   ,  8358.
,
       9258.   ,  8058.   ,  8238.   ,  9298.   ,  9538.   ,  8449.
,
       9639.   ,  9989.   , 11199.   , 11549.   , 17669.   ,  8948.
,
      10698.   ,  9988.   , 10898.   , 11248.   , 16558.   , 15998.
,
      15690.   , 15750.   ,  7775.   ,  7975.   ,  7995.   ,  8195.
,
       8495.   ,  9495.   ,  9995.   , 11595.   ,  9980.   , 13295.
,
      13845.   , 12290.   , 12940.   , 13415.   , 15985.   , 16515.
,
      18420.   , 18950.   , 16845.   , 19045.   , 21485.   , 22470.
,
      22625.   ])
```

Split the data into training and testing data

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,rando
m_state=42)
```

Scaling using StandardScaler

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(x_train)
x_train=scaler.transform(x_train)
x_test=scaler.transform(x_test)
```

Model Creation

```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
y_pred
```

```
array([25263.71155727, 17510.60100873,  9440.29961597, 13532.18496334,
       26109.69474918,  6312.04809043,  7584.6572441 ,  5893.76273471,
        9242.62222642,  5821.86589929, 13857.61067278,  6162.24167276,
       16482.82264907, 10706.77526758, 40393.49162707,  6861.31339325,
        -151.21120815, 14789.44762836,  9670.36006267, 10331.20595444,
       11095.07488034, 20819.20389829,  8195.27918247,  3546.04281395,
        7755.9093543 , 23953.59319483, 14433.67089985, 15762.62290799,
        5067.19507839, 16242.84673396, 26307.25262831,  7381.45646566,
        4235.16070955, 22127.7853912 ,  8452.86521883, 26796.96656169,
       10151.62813053,  9736.48156002,  6845.13061313, 15091.05428725,
        7338.31882785, 13461.96663554, 18879.25403184,  4724.11656838,
        6735.85164657, 10078.5772291 ,  8866.25914506,  6925.7671163 ,
       18238.42401821, 15330.17170675,  7106.81136173, 19425.30940642,
        3305.86756574,  9293.16030173,  4902.96088326, 14725.62603104,
       14265.61948852,  9710.88151382, 34700.30707048,  6428.57784804,
        8599.07387595, 20697.64419021])
```

```
df1=pd.DataFrame({"Actual_value":y_test,"Predicted_value":y_pred,'Diff
erence':y_test-y_pred})
df1
```

	Actual_value	Predicted_value	Difference
0	30760.000	25263.711557	5496.288443
1	17859.167	17510.601009	348.565991
2	9549.000	9440.299616	108.700384
3	11850.000	13532.184963	-1682.184963
4	28248.000	26109.694749	2138.305251
..	...	...	...
57	11845.000	9710.881514	2134.118486
58	37028.000	34700.307070	2327.692930
59	5389.000	6428.577848	-1039.577848
60	9233.000	8599.073876	633.926124
61	17199.000	20697.644190	-3498.644190

[62 rows x 3 columns]

Performance Evaluation

```
# 1. MAE
from sklearn.metrics import mean_absolute_error
print('MAE is ',mean_absolute_error(y_test,y_pred))
```

```
MAE is  2196.0531111239407
```

```
# 2. MAPE
from sklearn.metrics import mean_absolute_percentage_error
print('MAE is ',mean_absolute_percentage_error(y_test,y_pred))
```

```
MAE is  0.19661972274847273
```

```
# 3. MSE
from sklearn.metrics import mean_squared_error
print('MAE is ',mean_squared_error(y_test,y_pred))
```

```
MAE is  11091618.680797806
```

```
# 4. RMSE
from sklearn.metrics import mean_squared_error
mse=mean_squared_error(y_test,y_pred)
rmse=np.sqrt(mse)
rmse
```

```
3330.4081853126963
```

```
# 5. R2_score
from sklearn.metrics import r2_score
print(r2_score(y_test,y_pred))
```

```
0.8399116957802601
```