CAR PRICE PREDICTION USING LINEAR REGRESSION AND OTHERS ALGORITHMS

Importing the libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re
from sklearn.model_selection import train_test_split,
RandomizedSearchCV
from sklearn.metrics import r2_score
from sklearn.ensemble import RandomForestRegressor,
ExtraTreesRegressor
from sklearn.preprocessing import OrdinalEncoder
```

Problem Statement

Here we have the Car details along with its price. Based on the given data, we going to predict the price of the Car.

Defining the dataset

Link: https://drive.google.com/file/d/18xxy7kEGzp3vAwsA7QtrjyAwhffPiowo/view?usp=sharing

```
df = pd.read_csv('/content/CarPrice_Assignment (2).csv')
```

EDA - Exploratory Data Analysis (DE, DM, DC, DV)

```
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.shape
```

```
(205, 26)
```

```
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
```

```
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')
```

```
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.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
```

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

```
0
```

```
df.duplicated().sum()
```

```
0
```

```
df.head(2)
```

```
   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

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

1  convertible        rwd          front       88.6  ...         130


   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

   highwaympg    price
0          27  13495.0
1          27  16500.0

[2 rows x 26 columns]
```

```
df.drop('car_ID',axis=1,inplace=True) # car_ID is not going to help us
in prediction

df.head()
```

```
   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


       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  ...

   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

  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

[5 rows x 25 columns]
```

```
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)
```

```
df.CarName.nunique()
```

```
147
```

```
df.CarName[0]
```

{"type":"string"}

```
re.sub(' .*','',df.CarName[0])
```

{"type":"string"}

```
lis = []
for i in range(len(df)):
  brand_name = re.sub(' .*','',df.CarName[i])
  lis.append(brand_name)
```

```
print(lis)
```

```
['alfa-romero', 'alfa-romero', 'alfa-romero', 'audi', 'audi', 'audi',
'audi', 'audi', 'audi', 'audi', 'bmw', 'bmw', 'bmw', 'bmw', 'bmw',
'bmw', 'bmw', 'bmw', 'chevrolet', 'chevrolet', 'chevrolet', 'dodge',
'dodge', 'dodge', 'dodge', 'dodge', 'dodge', 'dodge', 'dodge',
'dodge', 'honda', 'honda', 'honda', 'honda', 'honda', 'honda',
'honda', 'honda', 'honda', 'honda', 'honda', 'honda', 'honda',
'isuzu', 'isuzu', 'isuzu', 'isuzu', 'jaguar', 'jaguar', 'jaguar',
'maxda', 'maxda', 'mazda', 'mazda', 'mazda', 'mazda', 'mazda',
'mazda', 'mazda', 'mazda', 'mazda', 'mazda', 'mazda', 'mazda',
'mazda', 'mazda', 'mazda', 'buick', 'buick', 'buick', 'buick',
'buick', 'buick', 'buick', 'buick', 'mercury', 'mitsubishi',
'mitsubishi', 'mitsubishi', 'mitsubishi', 'mitsubishi', 'mitsubishi',
'mitsubishi', 'mitsubishi', 'mitsubishi', 'mitsubishi', 'mitsubishi',
'mitsubishi', 'mitsubishi', 'Nissan', 'nissan', 'nissan', 'nissan',
'nissan', 'nissan', 'nissan', 'nissan', 'nissan', 'nissan', 'nissan',
'nissan', 'nissan', 'nissan', 'nissan', 'nissan', 'nissan', 'nissan',
'peugeot', 'peugeot', 'peugeot', 'peugeot', 'peugeot', 'peugeot',
'peugeot', 'peugeot', 'peugeot', 'peugeot', 'peugeot', 'plymouth',
'plymouth', 'plymouth', 'plymouth', 'plymouth', 'plymouth',
'plymouth', 'porsche', 'porcshce', 'porsche', 'porsche', 'porsche',
'renault', 'renault', 'saab', 'saab', 'saab', 'saab', 'saab', 'saab',
'subaru', 'subaru', 'subaru', 'subaru', 'subaru', 'subaru', 'subaru',
'subaru', 'subaru', 'subaru', 'subaru', 'subaru', 'toyota', 'toyota',
'toyota', 'toyota', 'toyota', 'toyota', 'toyota', 'toyota', 'toyota',
'toyota', 'toyota', 'toyota', 'toyota', 'toyota', 'toyota', 'toyota',
'toyota', 'toyota', 'toyota', 'toyota', 'toyota', 'toyota', 'toyota',
'toyota', 'toyota', 'toyota', 'toyota', 'toyota', 'toyota', 'toyota',
'toyota', 'toyouta', 'vokswagen', 'volkswagen', 'volkswagen',
'volkswagen', 'volkswagen', 'volkswagen', 'volkswagen', 'vw', 'vw',
'volkswagen', 'volkswagen', 'volkswagen', 'volvo', 'volvo', 'volvo',
'volvo', 'volvo', 'volvo', 'volvo', 'volvo', 'volvo', 'volvo',
'volvo']
```

```python
df['CarName'] = pd.DataFrame(lis)
```

```python
df.head(1)
```

```
   symboling      CarName fueltype aspiration doornumber      carbody
\
0          3  alfa-romero      gas        std        two  convertible


  drivewheel enginelocation  wheelbase  carlength  ...  enginesize  \
0        rwd          front       88.6      168.8  ...         130

   fuelsystem  boreratio stroke compressionratio  horsepower peakrpm
citympg  \
0        mpfi       3.47   2.68              9.0         111    5000
21
```

```
   highwaympg    price
0          27  13495.0

[1 rows x 25 columns]
```

```
df.CarName.nunique()
```

```
28
```

```
df.head()
```

```
   symboling      CarName fueltype aspiration doornumber      carbody
\
0          3  alfa-romero      gas        std        two  convertible

1          3  alfa-romero      gas        std        two  convertible

2          1  alfa-romero      gas        std        two    hatchback

3          2         audi      gas        std       four        sedan

4          2         audi      gas        std       four        sedan


  drivewheel enginelocation  wheelbase  carlength  ...  enginesize  \
0        rwd          front       88.6      168.8  ...         130
1        rwd          front       88.6      168.8  ...         130
2        rwd          front       94.5      171.2  ...         152
3        fwd          front       99.8      176.6  ...         109
4        4wd          front       99.4      176.6  ...         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 25 columns]
```

```
df.CarName.unique()
```

```
array(['alfa-romero', 'audi', 'bmw', 'chevrolet', 'dodge', 'honda',
       'isuzu', 'jaguar', 'maxda', 'mazda', 'buick', 'mercury',
       'mitsubishi', 'Nissan', 'nissan', 'peugeot', 'plymouth',
'porsche',
       'porcshce', 'renault', 'saab', 'subaru', 'toyota', 'toyouta',
       'vokswagen', 'volkswagen', 'vw', 'volvo'], dtype=object)
```

In the above values, Some values having spelling mistake so we going to correct it now

```
'''
'maxda' -> 'mazda'
'Nissan' -> 'nissan'
'porcshce' -> 'porsche'
"toyouta" -> "toyota"
"vokswagen" & "vw" -> "volkswagen"
'''
```

{"type":"string"}

Replacing the wrong name to correct names....

```
df.CarName.replace(['maxda', 'Nissan', 'porcshce', 'toyouta',
'vokswagen', 'vw'],
                   ['mazda', 'nissan', 'porsche', 'toyota',
'volkswagen', 'volkswagen'], inplace=True)
```

```
#Now check the value
```

```
df.CarName.nunique()
```

```
22
```

we reduce our values to 22.

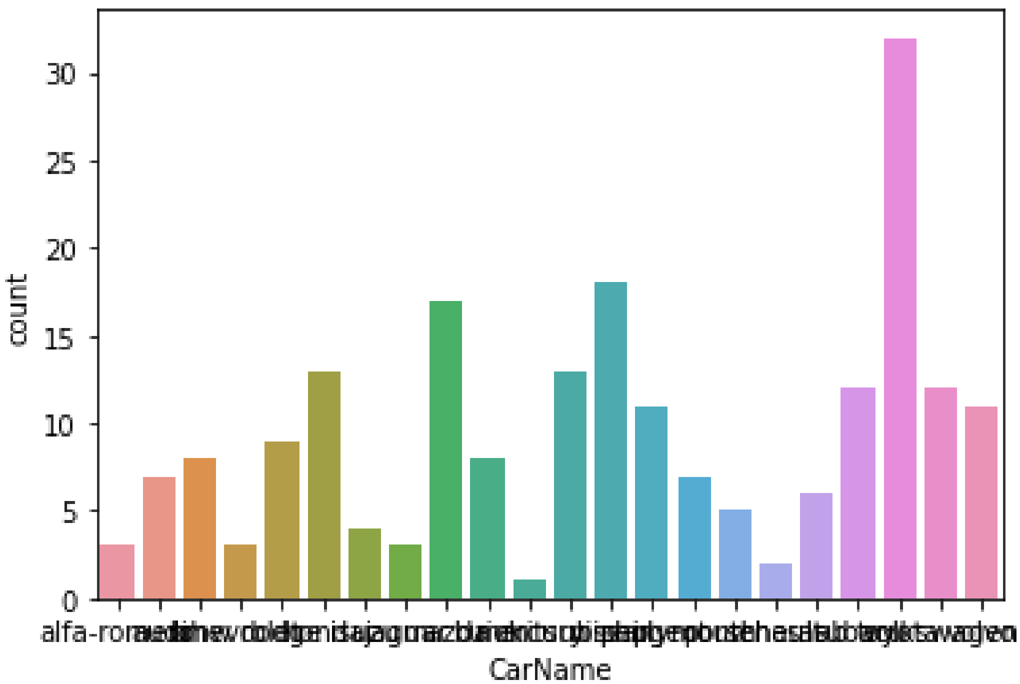
```
df.CarName.value_counts()
```

```
toyota        32
nissan        18
mazda         17
mitsubishi    13
honda         13
volkswagen    12
subaru        12
peugeot       11
volvo         11
dodge          9
buick          8
```

```
bmw             8
audi            7
plymouth        7
saab            6
porsche         5
isuzu           4
jaguar          3
chevrolet       3
alfa-romero     3
renault         2
mercury         1
Name: CarName, dtype: int64
```

```
sns.countplot(data = df, x='CarName')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa97276a690>
```



In the above column , from dodge to mercurcy are Luxury Cars or High Class Cars and having less than 10 number of datapoints , so we grouping them into a single category called "other"

```
df.CarName.value_counts()
```

```
toyota         32
nissan         18
mazda          17
mitsubishi     13
honda          13
```

```
volkswagen     12
subaru         12
peugeot        11
volvo          11
dodge           9
buick           8
bmw             8
audi            7
plymouth        7
saab            6
porsche         5
isuzu           4
jaguar          3
chevrolet       3
alfa-romero     3
renault         2
mercury         1
Name: CarName, dtype: int64
```

The above unique values in the column : CarName is reduced from 147 - 10 , So we can get more **ACCURACY** than before

```
# 147 -> 28 -> 22 -> 10
df.CarName.nunique()
```

```
22
```

#Performing (ENCODING :) ~ Changing values from string to intergers in required columns.

~We can also use LabelEncoding to change the values but here i am using replace functions.

```
le = OrdinalEncoder()
df['CarName']= le.fit_transform(df[['CarName']])
df['fueltype'] = le.fit_transform(df[['fueltype']])
df['aspiration'] = le.fit_transform(df[['aspiration']])
df['fueltype'] = le.fit_transform(df[['fueltype']])
df['doornumber'] = le.fit_transform(df[['doornumber']])
df['drivewheel'] = le.fit_transform(df[['drivewheel']])
df['enginelocation'] = le.fit_transform(df[['enginelocation']])
df['enginetype'] = le.fit_transform(df[['enginetype']])
df['cylindernumber'] = le.fit_transform(df[['cylindernumber']])
df['carbody'] = le.fit_transform(df[['carbody']])
df['fuelsystem'] = le.fit_transform(df[['fuelsystem']])
```

```
df.head()
```

```
   symboling  CarName  fueltype  aspiration  doornumber  carbody  
drivewheel  \
0          3      0.0       1.0         0.0         1.0      0.0   
2.0
```

```
1          3      0.0       1.0         0.0         1.0      0.0
2.0
2          1      0.0       1.0         0.0         1.0      2.0
2.0
3          2      1.0       1.0         0.0         0.0      3.0
1.0
4          2      1.0       1.0         0.0         0.0      3.0
0.0

   enginelocation  wheelbase  carlength  ...  enginesize
fuelsystem  \
0             0.0       88.6      168.8  ...         130         5.0

1             0.0       88.6      168.8  ...         130         5.0

2             0.0       94.5      171.2  ...         152         5.0

3             0.0       99.8      176.6  ...         109         5.0

4             0.0       99.4      176.6  ...         136         5.0


   boreratio  stroke  compressionratio  horsepower  peakrpm
citympg  \
0       3.47    2.68               9.0         111     5000        21

1       3.47    2.68               9.0         111     5000        21

2       2.68    3.47               9.0         154     5000        19

3       3.19    3.40              10.0         102     5500        24

4       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 25 columns]
```

ALL DATA ARE CLEANED

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   symboling         205 non-null    int64  
 1   CarName           205 non-null    float64
 2   fueltype          205 non-null    float64
 3   aspiration        205 non-null    float64
 4   doornumber        205 non-null    float64
 5   carbody           205 non-null    float64
 6   drivewheel        205 non-null    float64
 7   enginelocation    205 non-null    float64
 8   wheelbase         205 non-null    float64
 9   carlength         205 non-null    float64
 10  carwidth          205 non-null    float64
 11  carheight         205 non-null    float64
 12  curbweight        205 non-null    int64  
 13  enginetype        205 non-null    float64
 14  cylindernumber    205 non-null    float64
 15  enginesize        205 non-null    int64  
 16  fuelsystem        205 non-null    float64
 17  boreratio         205 non-null    float64
 18  stroke            205 non-null    float64
 19  compressionratio  205 non-null    float64
 20  horsepower        205 non-null    int64  
 21  peakrpm           205 non-null    int64  
 22  citympg           205 non-null    int64  
 23  highwaympg        205 non-null    int64  
 24  price             205 non-null    float64
dtypes: float64(18), int64(7)
memory usage: 40.2 KB
```

```
df.head(2)
```

```
   symboling  CarName  fueltype  aspiration  doornumber  carbody  
drivewheel  \
0          3      0.0       1.0         0.0         1.0      0.0         
2.0   
1          3      0.0       1.0         0.0         1.0      0.0         
2.0   

   enginelocation  wheelbase  carlength  ...  enginesize  
fuelsystem  \
0             0.0       88.6      168.8  ...         130         5.0   

1             0.0       88.6      168.8  ...         130         5.0   


   boreratio  stroke  compressionratio  horsepower  peakrpm
```

```
citympg  \
0       3.47    2.68               9.0         111     5000       21

1       3.47    2.68               9.0         111     5000       21


   highwaympg    price
0          27  13495.0
1          27  16500.0

[2 rows x 25 columns]
```

```
df.corr()
```

```
                  symboling   CarName  fueltype  aspiration
doornumber  \
symboling          1.000000 -0.092793  0.194311   -0.059866
0.664073
CarName           -0.092793  1.000000 -0.055049    0.011326   -
0.150465
fueltype           0.194311 -0.055049  1.000000   -0.401397
0.191491
aspiration        -0.059866  0.011326 -0.401397    1.000000   -
0.031792
doornumber         0.664073 -0.150465  0.191491   -0.031792
1.000000
carbody           -0.596135  0.101473 -0.147853    0.063028   -
0.680358
drivewheel        -0.041671 -0.056639 -0.132257    0.066465
0.098954
enginelocation     0.212471  0.054410  0.040070   -0.057191
0.137757
wheelbase         -0.531954 -0.013288 -0.308346    0.257611   -
0.447357
carlength         -0.357612  0.030733 -0.212679    0.234539   -
0.398568
carwidth          -0.232919 -0.103779 -0.233880    0.300567   -
0.207168
carheight         -0.541038  0.189507 -0.284631    0.087311   -
0.552208
curbweight        -0.227691 -0.077989 -0.217275    0.324902   -
0.197379
enginetype         0.050372 -0.081760  0.082695   -0.102963
0.062431
cylindernumber     0.197762  0.053106  0.110617   -0.133119
0.154322
enginesize        -0.105790 -0.174513 -0.069594    0.108217   -
0.020742
fuelsystem         0.091163  0.104030  0.041529    0.288086
0.015519
```

```
boreratio         -0.130051  0.191301 -0.054451    0.212614   -
0.119258
stroke            -0.008735 -0.210031 -0.241829    0.222982
0.011082
compressionratio  -0.178515  0.088972 -0.984356    0.295541   -
0.177888
horsepower         0.070873 -0.109686  0.163926    0.241685
0.126947
peakrpm            0.273606 -0.151830  0.476883   -0.183383
0.247668
citympg           -0.035823  0.108458 -0.255963   -0.202362
0.012417
highwaympg         0.034606  0.119023 -0.191392   -0.254416
0.036330
price             -0.079978 -0.262234 -0.105679    0.177926   -
0.031835

                   carbody  drivewheel  enginelocation  wheelbase
carlength  \
symboling        -0.596135   -0.041671        0.212471  -0.531954  -
0.357612
CarName           0.101473   -0.056639        0.054410  -0.013288
0.030733
fueltype         -0.147853   -0.132257        0.040070  -0.308346  -
0.212679
aspiration        0.063028    0.066465       -0.057191   0.257611
0.234539
doornumber       -0.680358    0.098954        0.137757  -0.447357  -
0.398568
carbody           1.000000   -0.155745       -0.277009   0.401362
0.334433
drivewheel       -0.155745    1.000000        0.147865   0.459745
0.485649
enginelocation   -0.277009    0.147865        1.000000  -0.187790  -
0.050989
wheelbase         0.401362    0.459745       -0.187790   1.000000
0.874587
carlength         0.334433    0.485649       -0.050989   0.874587
1.000000
carwidth          0.131710    0.470751       -0.051698   0.795144
0.841118
carheight         0.568534   -0.019719       -0.106234   0.589435
0.491029
curbweight        0.128467    0.575111        0.050468   0.776386
0.877728
enginetype       -0.037024   -0.116823        0.114127  -0.135577  -
0.113291
cylindernumber   -0.048408    0.223238        0.135541  -0.184596  -
0.109585
enginesize       -0.073352    0.524307        0.196826   0.569329
```

```
0.683360
fuelsystem       -0.065079    0.424686        0.105971   0.384601
0.557810
boreratio         0.010549    0.481827        0.185042   0.488750
0.606454
stroke           -0.015325    0.071591       -0.138455   0.160959
0.129533
compressionratio  0.136243    0.127479       -0.019762   0.249786
0.158414
horsepower       -0.153928    0.518686        0.317839   0.353294
0.552623
peakrpm          -0.109643   -0.039417        0.198461  -0.360469  -
0.287242
citympg           0.031697   -0.449581       -0.153487  -0.470414  -
0.670909
highwaympg       -0.007170   -0.452220       -0.102026  -0.544082  -
0.704662
price            -0.083976    0.577992        0.324973   0.577816
0.682920

                  ...  enginesize  fuelsystem  boreratio    stroke  \
symboling         ...   -0.105790    0.091163  -0.130051 -0.008735
CarName           ...   -0.174513    0.104030   0.191301 -0.210031
fueltype          ...   -0.069594    0.041529  -0.054451 -0.241829
aspiration        ...    0.108217    0.288086   0.212614  0.222982
doornumber        ...   -0.020742    0.015519  -0.119258  0.011082
carbody           ...   -0.073352   -0.065079   0.010549 -0.015325
drivewheel        ...    0.524307    0.424686   0.481827  0.071591
enginelocation    ...    0.196826    0.105971   0.185042 -0.138455
wheelbase         ...    0.569329    0.384601   0.488750  0.160959
carlength         ...    0.683360    0.557810   0.606454  0.129533
carwidth          ...    0.735433    0.521434   0.559150  0.182942
carheight         ...    0.067149    0.017046   0.171071 -0.055307
curbweight        ...    0.850594    0.611642   0.648480  0.168790
enginetype        ...    0.040766   -0.091787   0.029355 -0.141918
cylindernumber    ...   -0.085613    0.011970  -0.032844 -0.050088
enginesize        ...    1.000000    0.514070   0.583774  0.203129
fuelsystem        ...    0.514070    1.000000   0.475599  0.088153
boreratio         ...    0.583774    0.475599   1.000000 -0.055909
stroke            ...    0.203129    0.088153  -0.055909  1.000000
compressionratio  ...    0.028971   -0.100786   0.005197  0.186110
horsepower        ...    0.809769    0.655638   0.573677  0.080940
peakrpm           ...   -0.244660    0.014261  -0.254976 -0.067964
citympg           ...   -0.653658   -0.671581  -0.584532 -0.042145
highwaympg        ...   -0.677470   -0.645659  -0.587012 -0.043931
price             ...    0.874145    0.526823   0.553173  0.079443

                  compressionratio  horsepower   peakrpm   citympg  \
symboling                -0.178515    0.070873  0.273606 -0.035823
CarName                   0.088972   -0.109686 -0.151830  0.108458
```

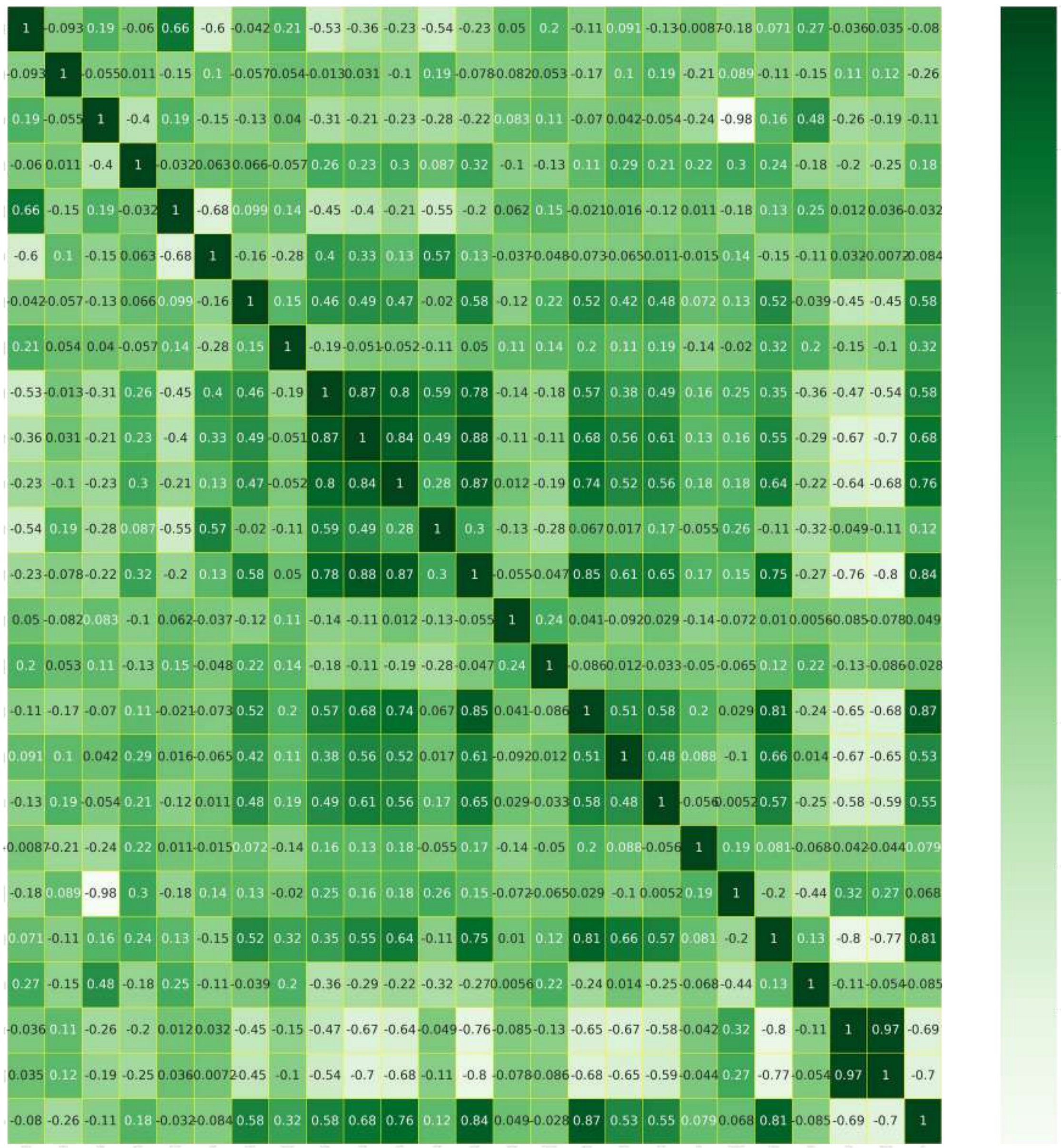
fueltype	-0.984356	0.163926	0.476883	-0.255963
aspiration	0.295541	0.241685	-0.183383	-0.202362
doornumber	-0.177888	0.126947	0.247668	0.012417
carbody	0.136243	-0.153928	-0.109643	0.031697
drivewheel	0.127479	0.518686	-0.039417	-0.449581
enginelocation	-0.019762	0.317839	0.198461	-0.153487
wheelbase	0.249786	0.353294	-0.360469	-0.470414
carlength	0.158414	0.552623	-0.287242	-0.670909
carwidth	0.181129	0.640732	-0.220012	-0.642704
carheight	0.261214	-0.108802	-0.320411	-0.048640
curbweight	0.151362	0.750739	-0.266243	-0.757414
enginetype	-0.071873	0.010301	0.005599	-0.085004
cylindernumber	-0.064701	0.115612	0.222731	-0.126422
enginesize	0.028971	0.809769	-0.244660	-0.653658
fuelsystem	-0.100786	0.655638	0.014261	-0.671581
boreratio	0.005197	0.573677	-0.254976	-0.584532
stroke	0.186110	0.080940	-0.067964	-0.042145
compressionratio	1.000000	-0.204326	-0.435741	0.324701
horsepower	-0.204326	1.000000	0.131073	-0.801456
peakrpm	-0.435741	0.131073	1.000000	-0.113544
citympg	0.324701	-0.801456	-0.113544	1.000000
highwaympg	0.265201	-0.770544	-0.054275	0.971337
price	0.067984	0.808139	-0.085267	-0.685751

	highwaympg	price
symboling	0.034606	-0.079978
CarName	0.119023	-0.262234
fueltype	-0.191392	-0.105679
aspiration	-0.254416	0.177926
doornumber	0.036330	-0.031835
carbody	-0.007170	-0.083976
drivewheel	-0.452220	0.577992
enginelocation	-0.102026	0.324973
wheelbase	-0.544082	0.577816
carlength	-0.704662	0.682920
carwidth	-0.677218	0.759325
carheight	-0.107358	0.119336
curbweight	-0.797465	0.835305
enginetype	-0.078456	0.049171
cylindernumber	-0.085897	-0.027628
enginesize	-0.677470	0.874145
fuelsystem	-0.645659	0.526823
boreratio	-0.587012	0.553173
stroke	-0.043931	0.079443
compressionratio	0.265201	0.067984
horsepower	-0.770544	0.808139
peakrpm	-0.054275	-0.085267
citympg	0.971337	-0.685751
highwaympg	1.000000	-0.697599
price	-0.697599	1.000000

```
[25 rows x 25 columns]
```

```
plt.figure(figsize=(100,100))
sns.heatmap(df.corr(), annot=True,linewidths = 2,linecolor = "yellow",
cmap='Greens',annot_kws={'size': 60})
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa97217ab90>
```



```
#sns.pairplot(df)
```

DATA VISUALIZATION

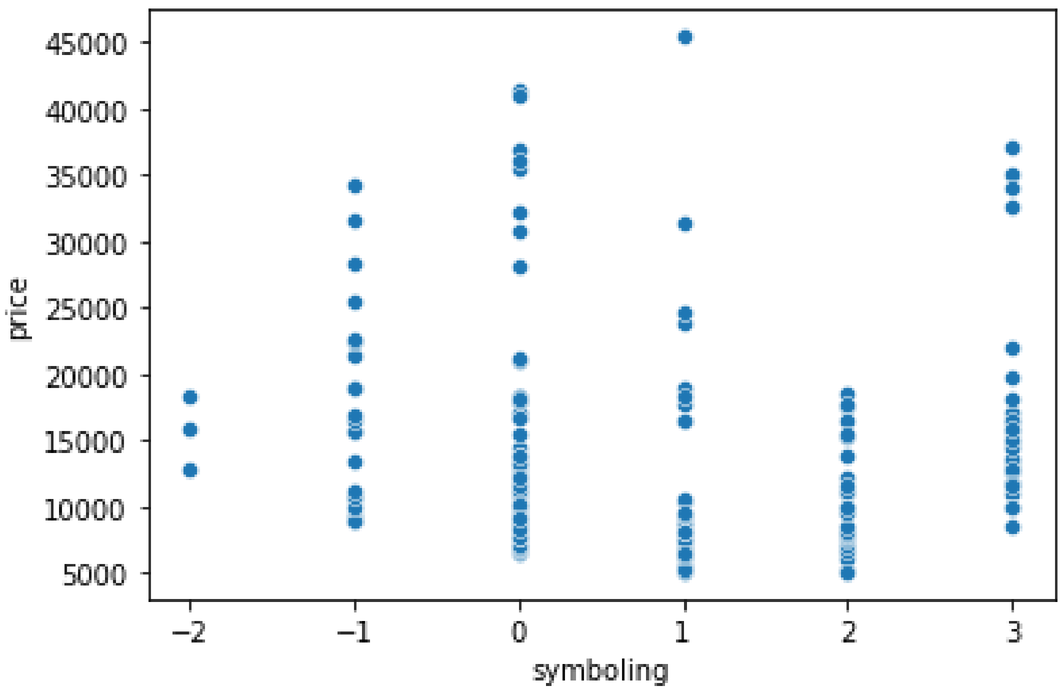
bold text

- Scatter Plot for features w.r.t target

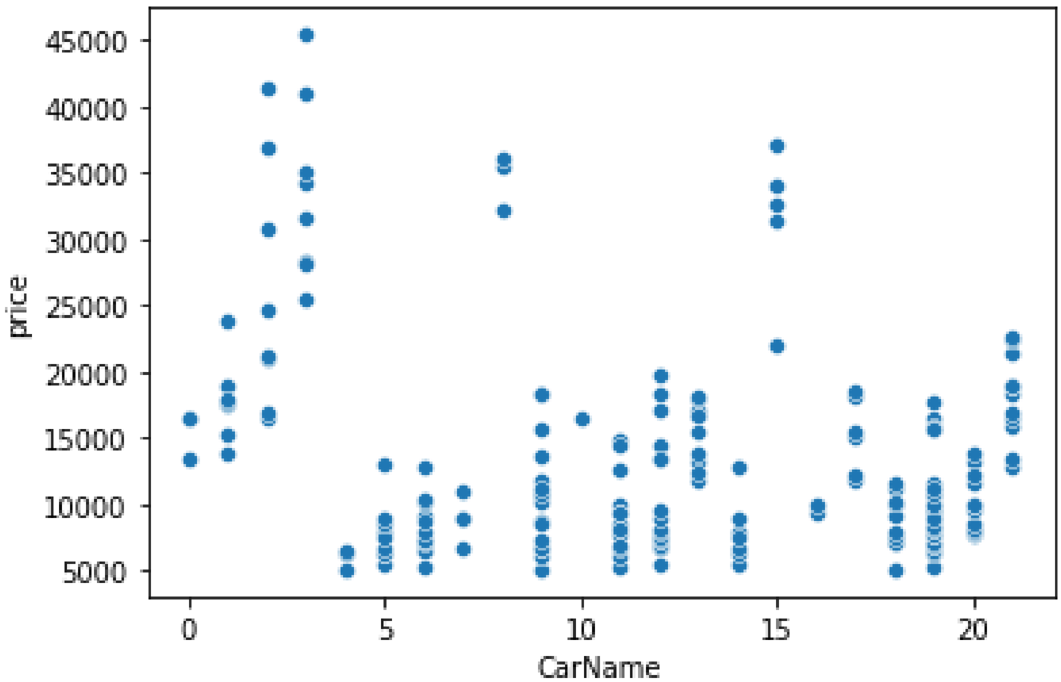
```
for i in df.drop('price',axis=1).columns:

    print("Scatter Plot for ",i)
    sns.scatterplot(x = df[i], y = df['price'])
    plt.show()
    print('-'*200)
```

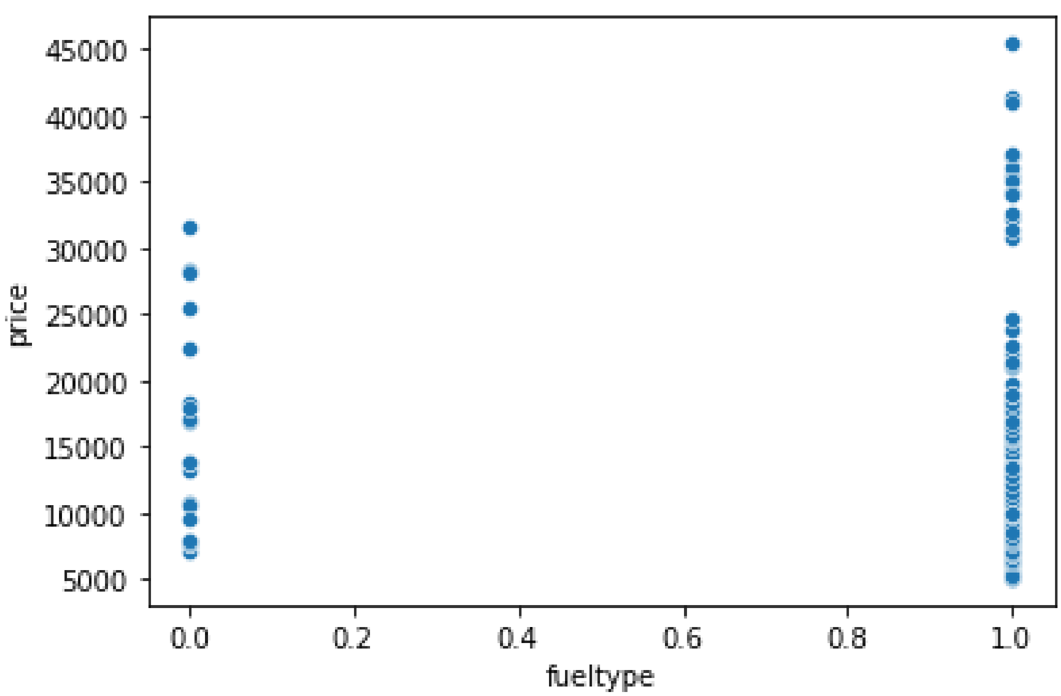
```
Scatter Plot for  symboling
```



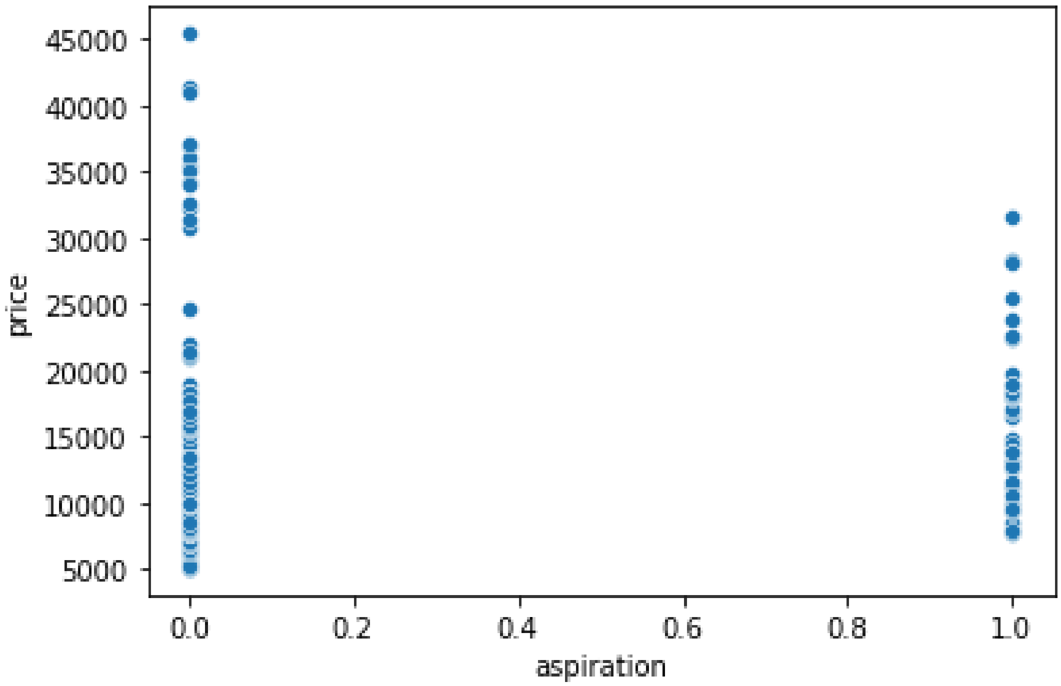

```
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
Scatter Plot for  CarName
```



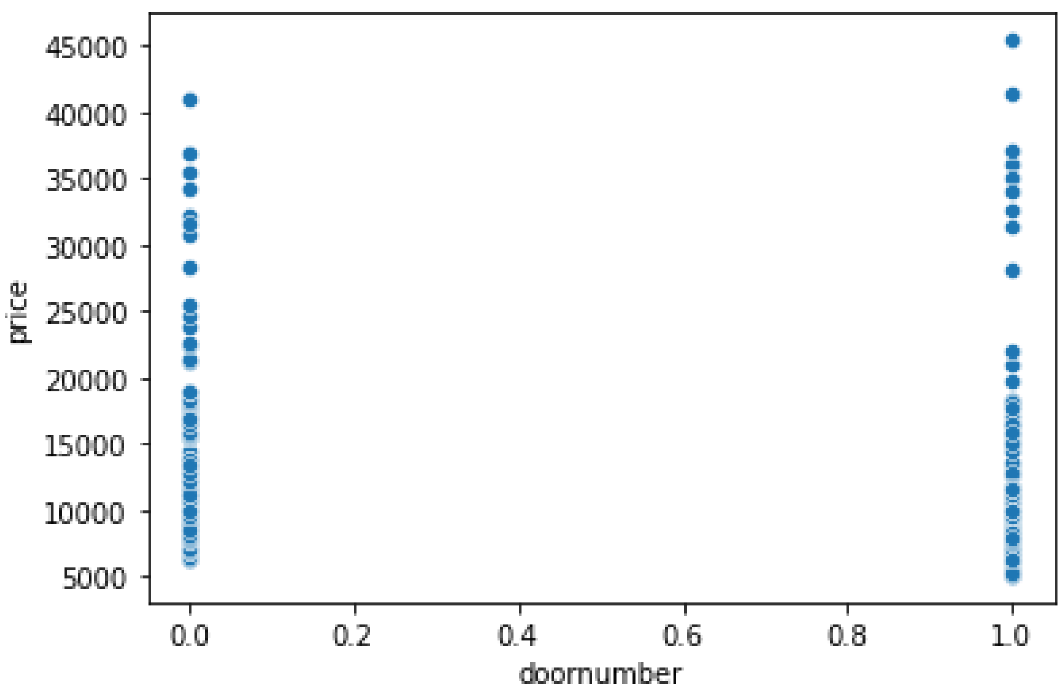
Scatter Plot for fueltype



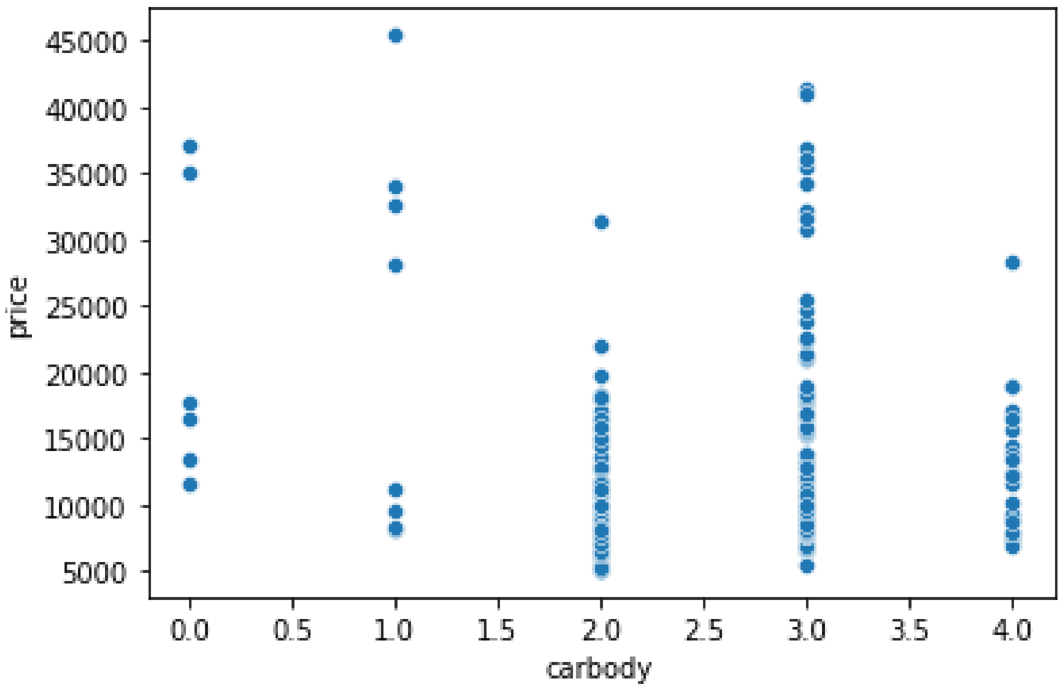
Scatter Plot for aspiration



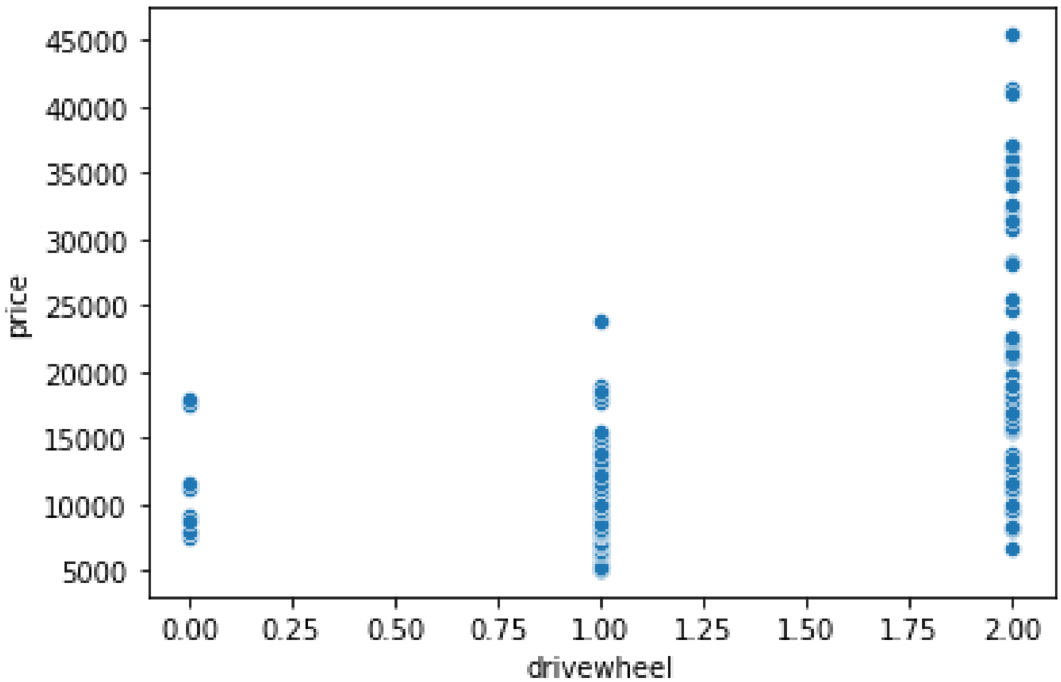
Scatter Plot for doornumber



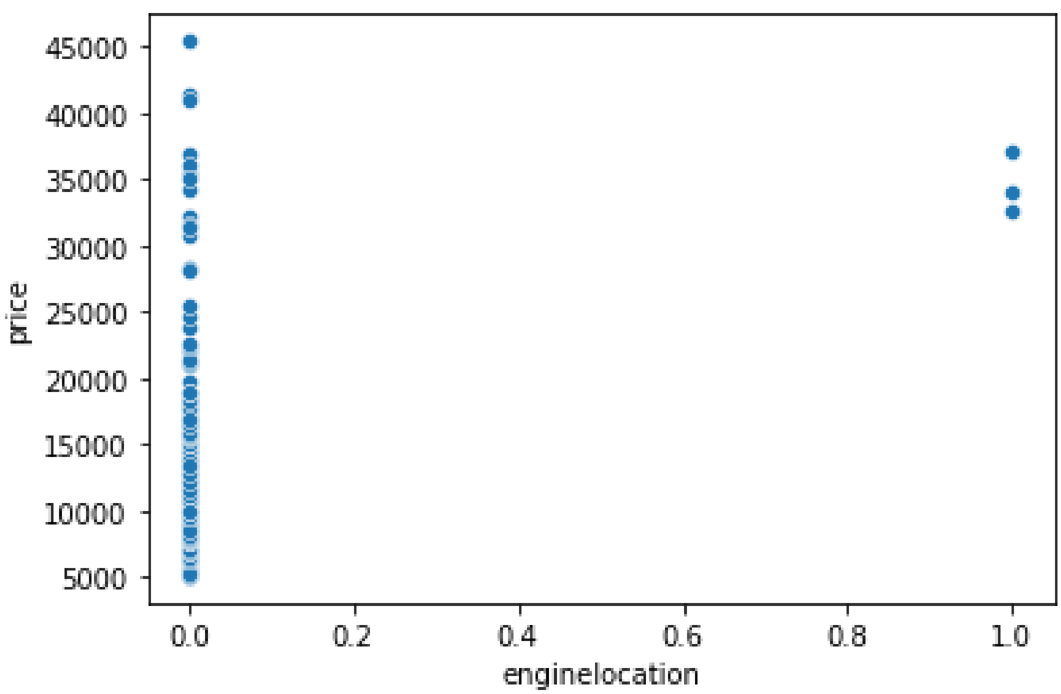
Scatter Plot for carbody



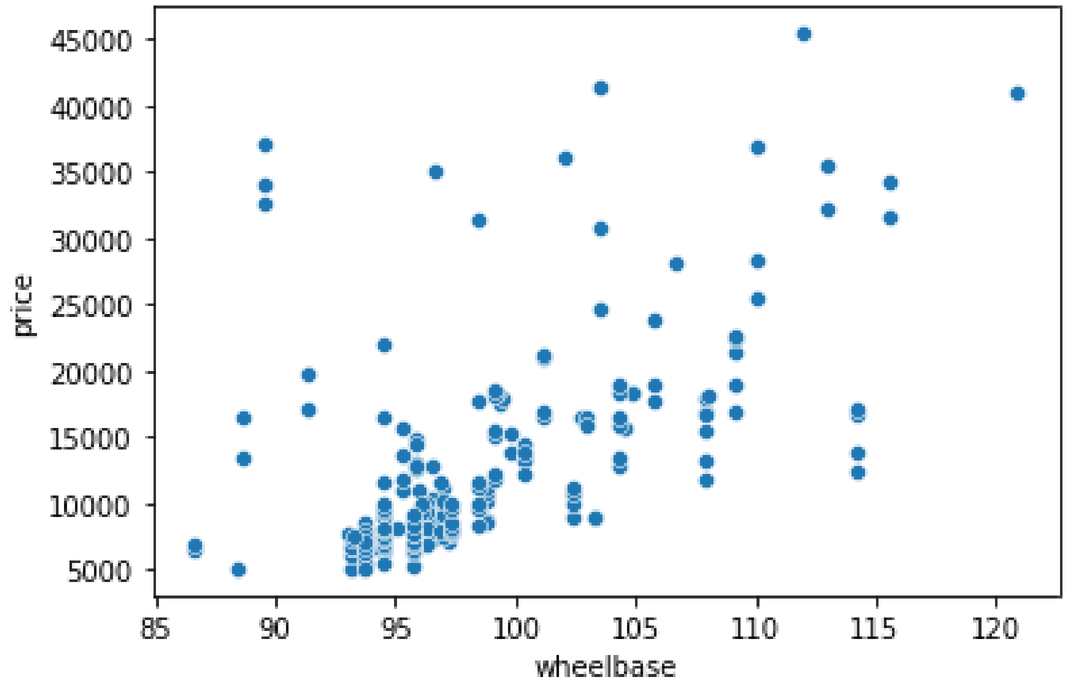
Scatter Plot for drivewheel



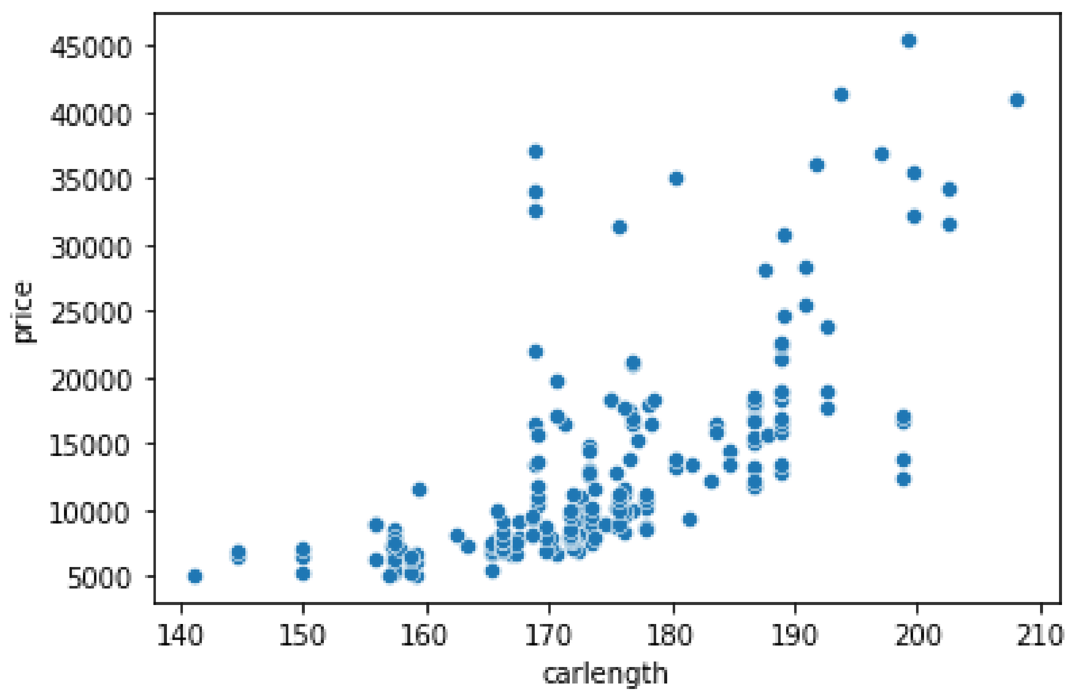
Scatter Plot for enginelocation



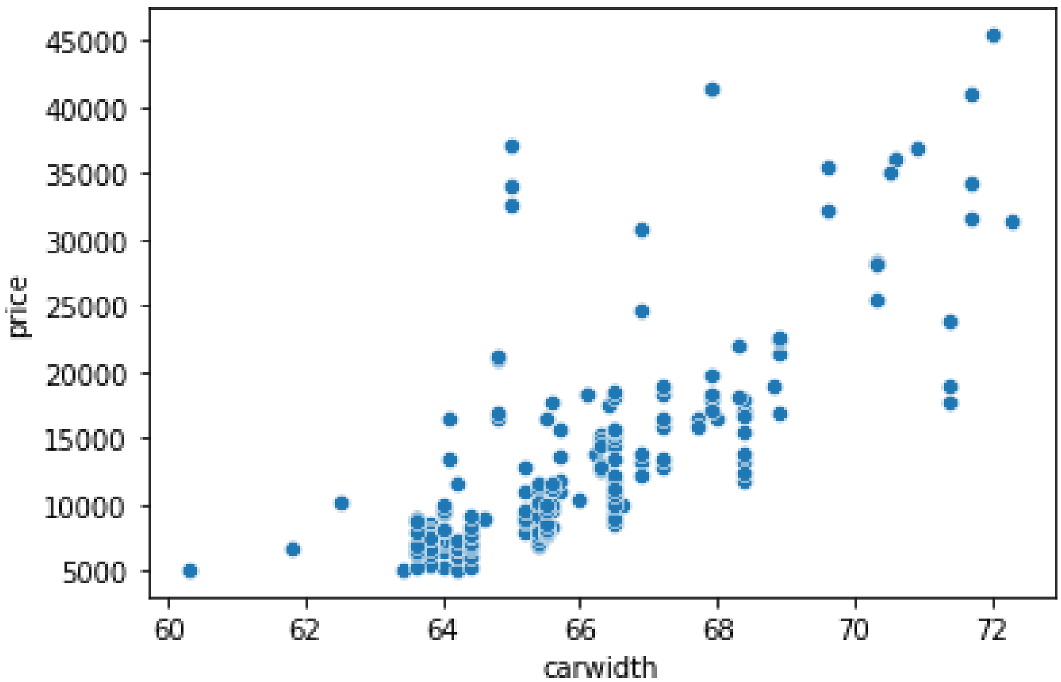
Scatter Plot for wheelbase



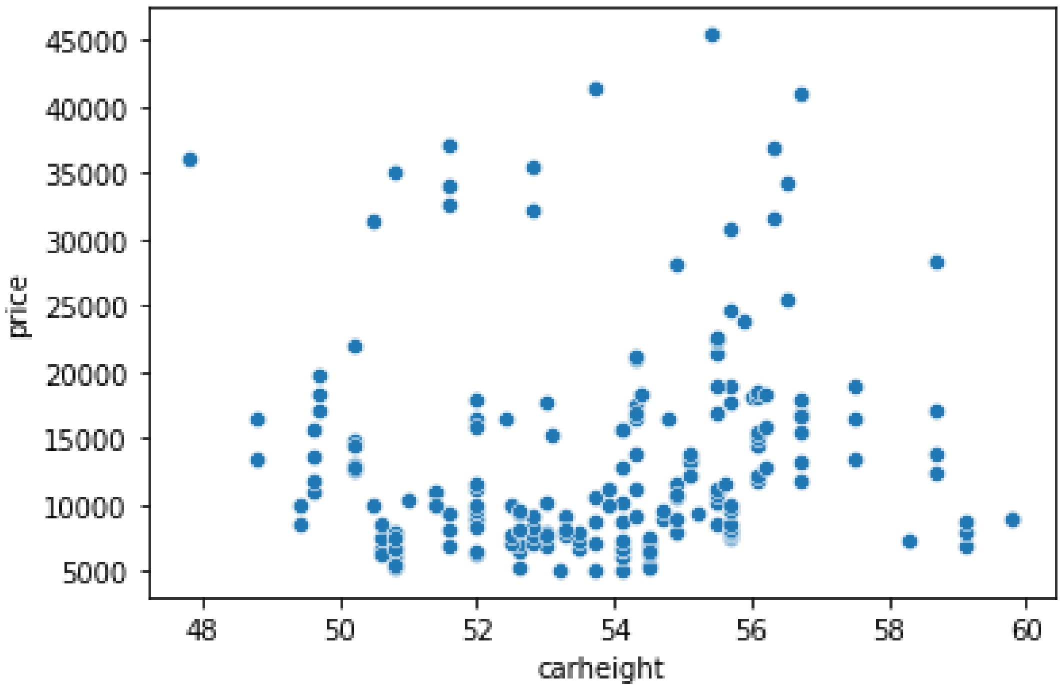
Scatter Plot for carlength



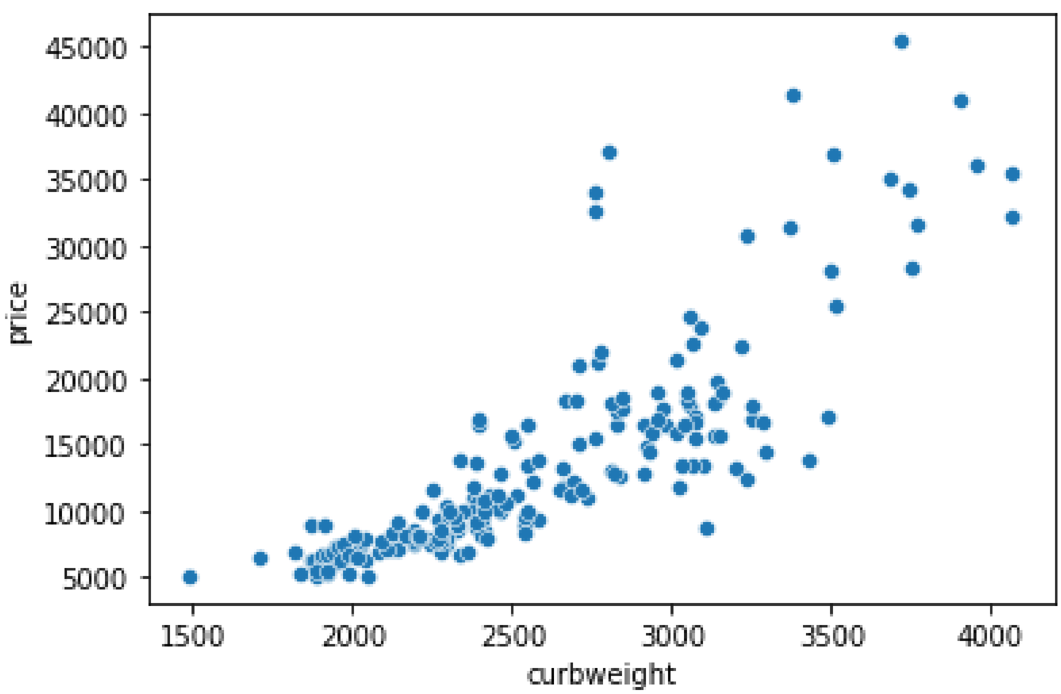
Scatter Plot for carwidth



Scatter Plot for carheight

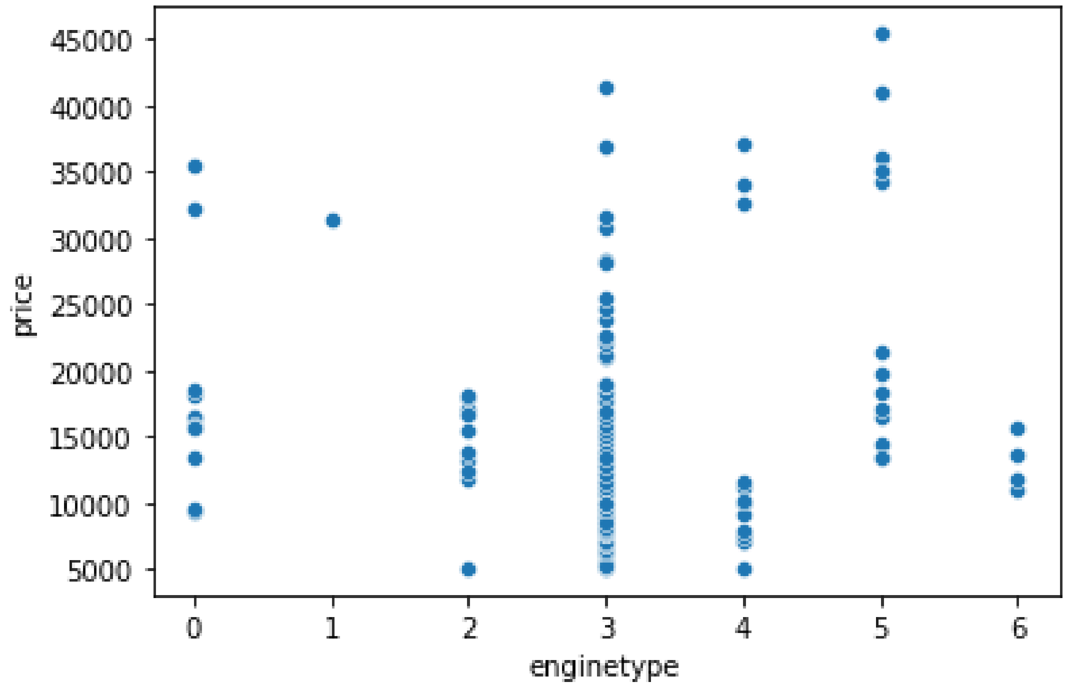



```
----------------------------------------------------------------------
----------------------------------------------------------------------
------------------------------------------------------------
Scatter Plot for  curbweight
```

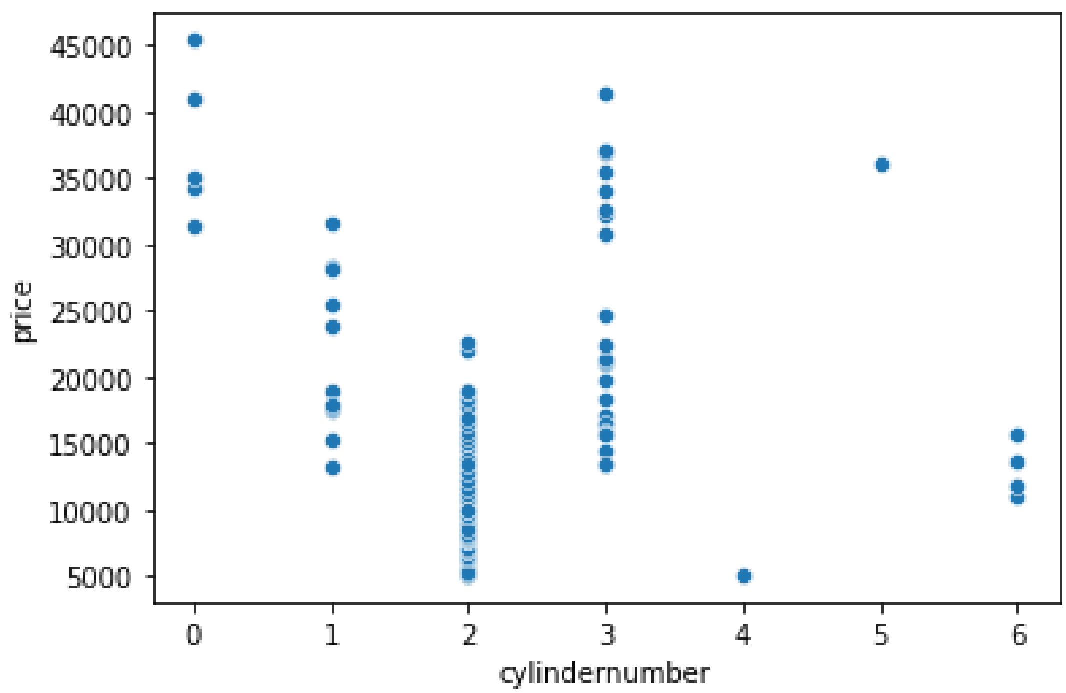



```
----------------------------------------------------------------------
----------------------------------------------------------------------
```

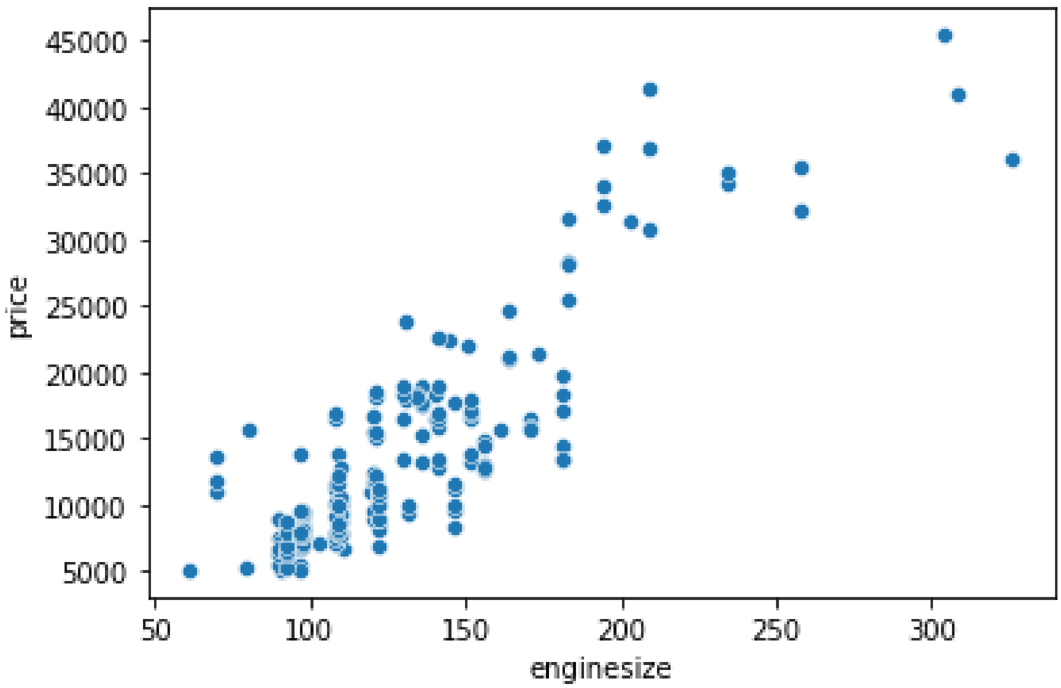
Scatter Plot for enginetype



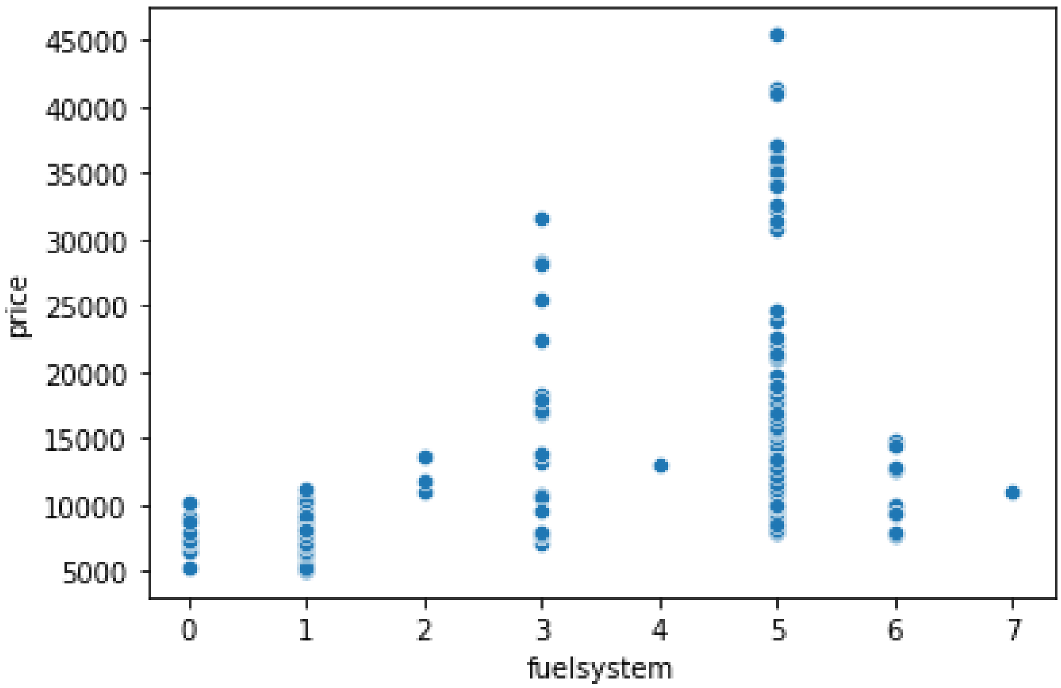
Scatter Plot for cylindernumber



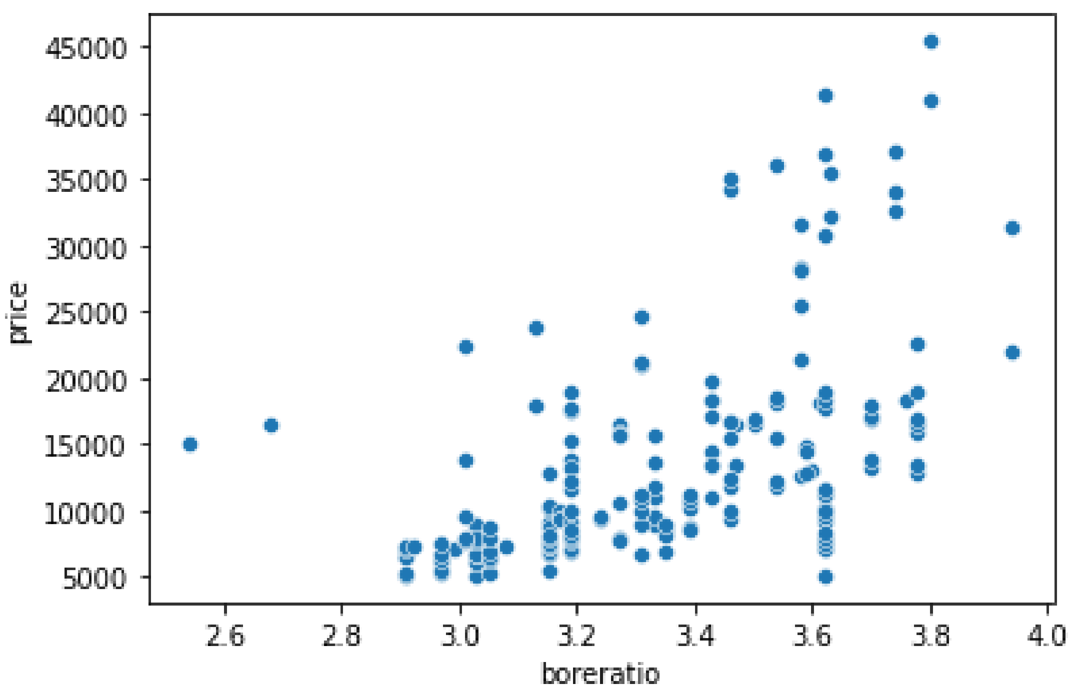
Scatter Plot for enginesize



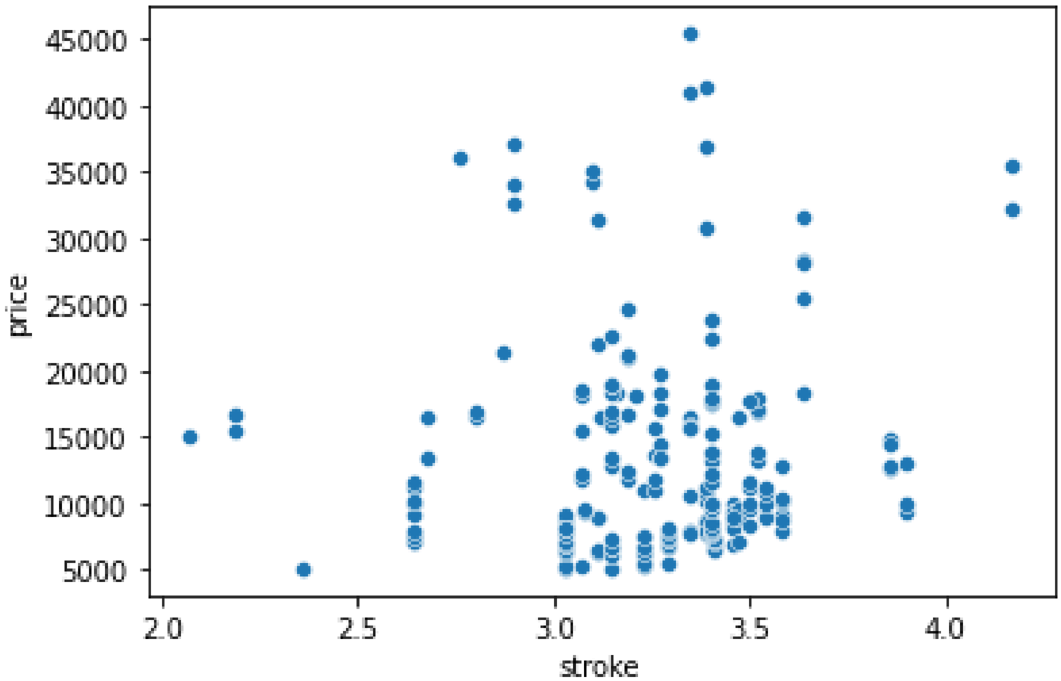
Scatter Plot for fuelsystem



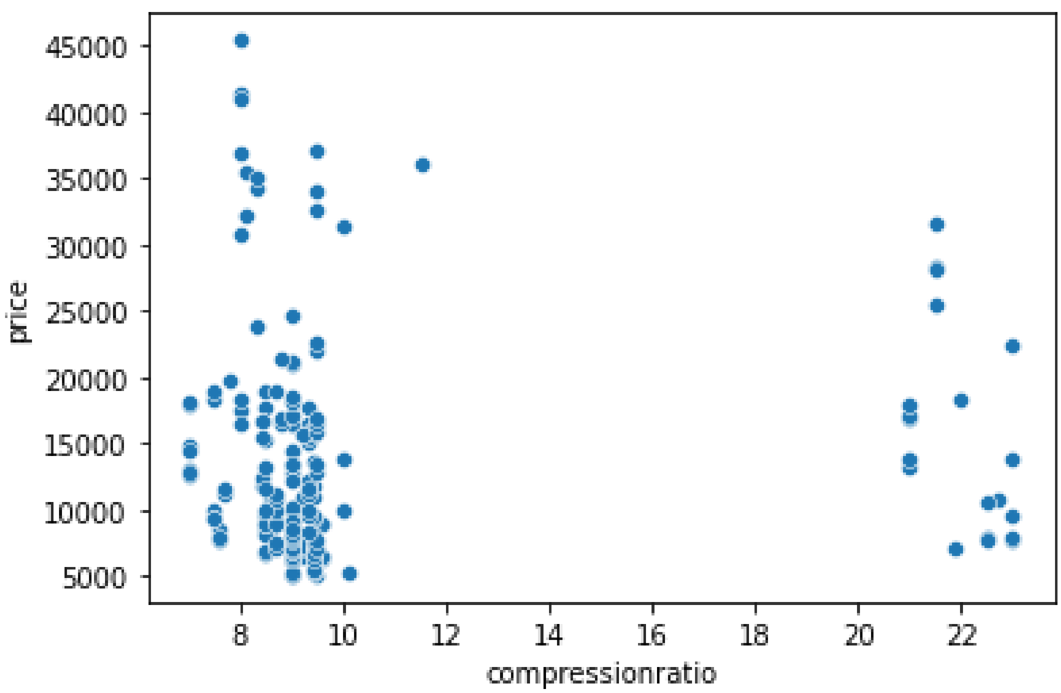
Scatter Plot for boreratio



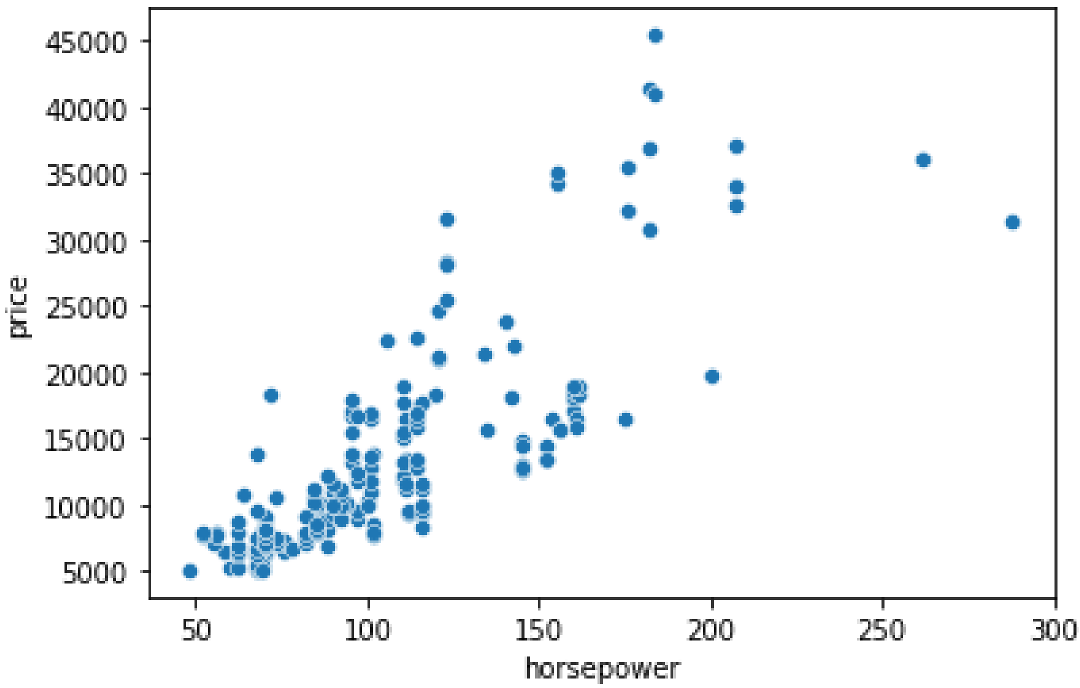
Scatter Plot for stroke



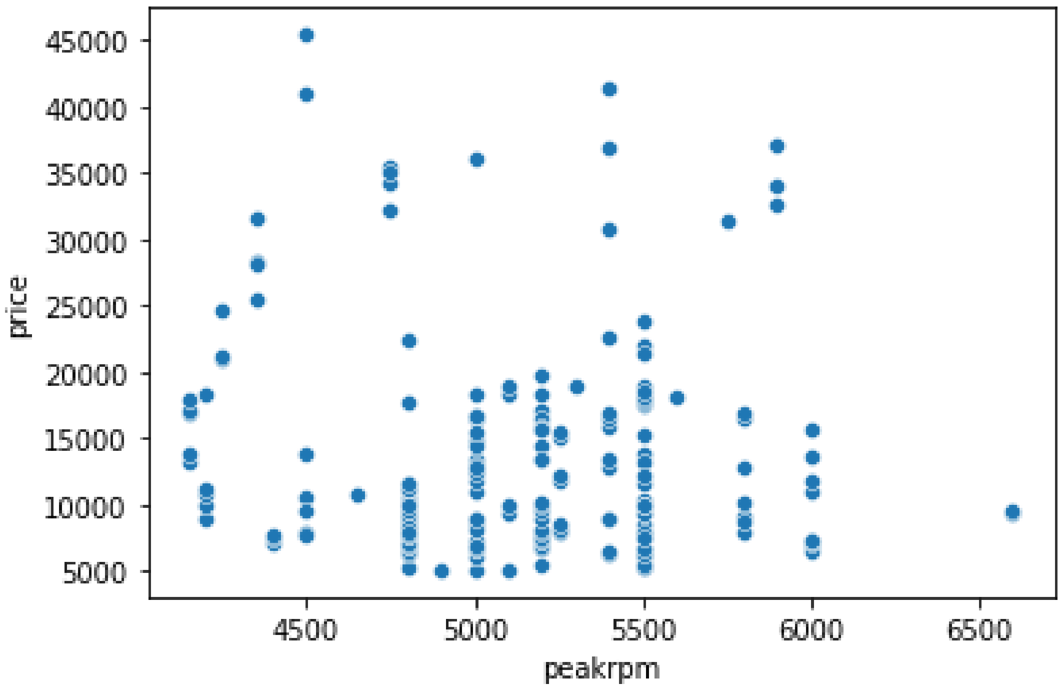
Scatter Plot for compressionratio



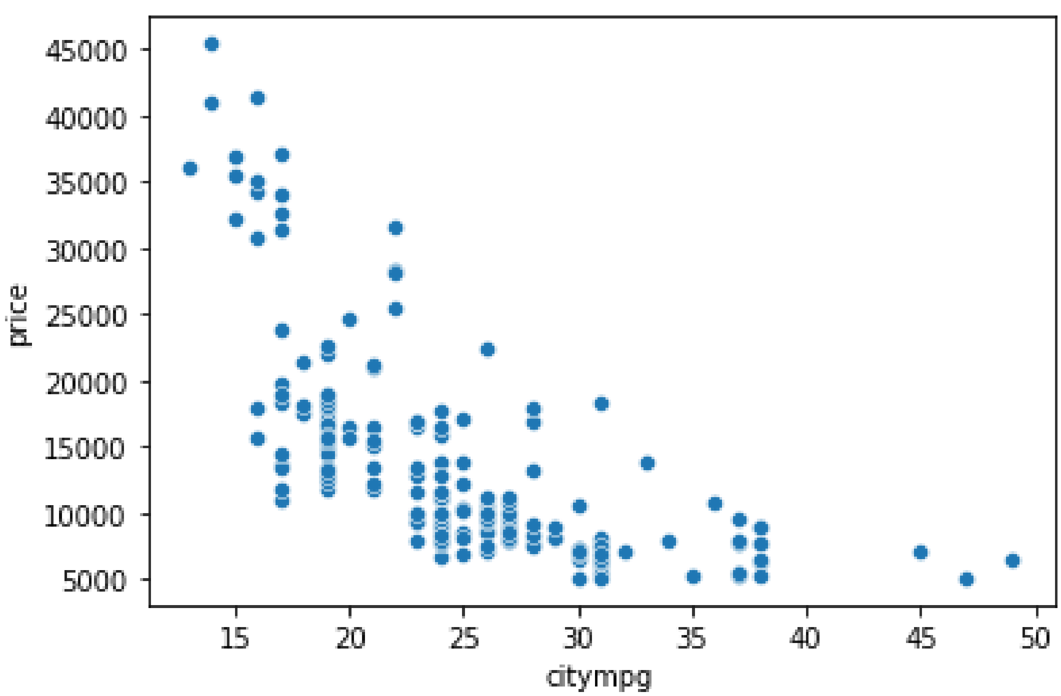
Scatter Plot for horsepower



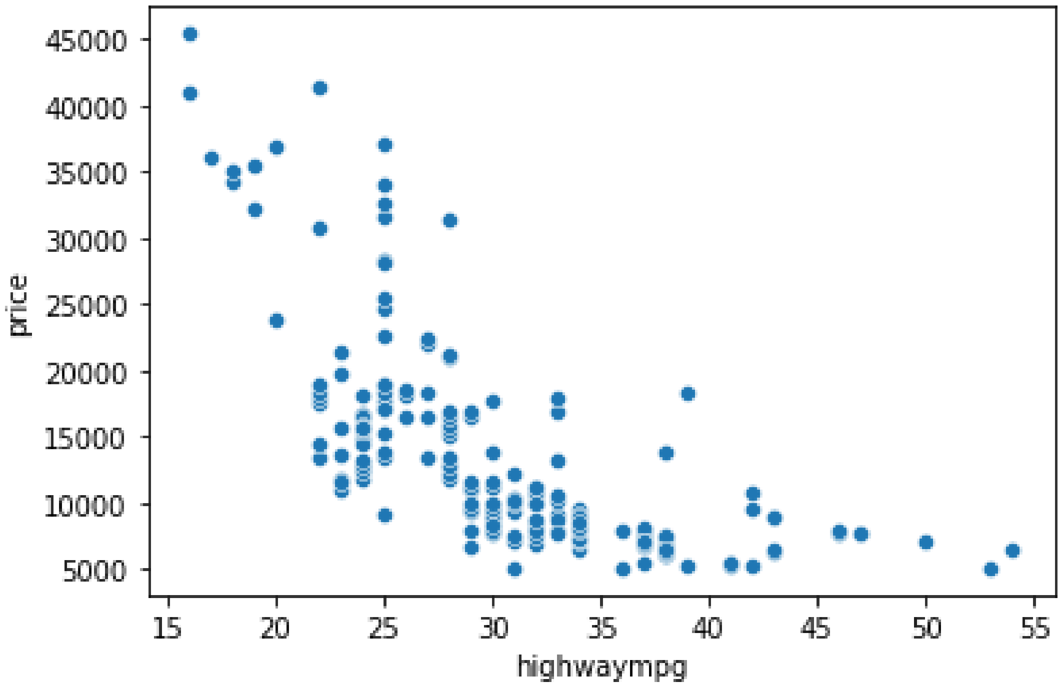
Scatter Plot for peakrpm



Scatter Plot for citympg

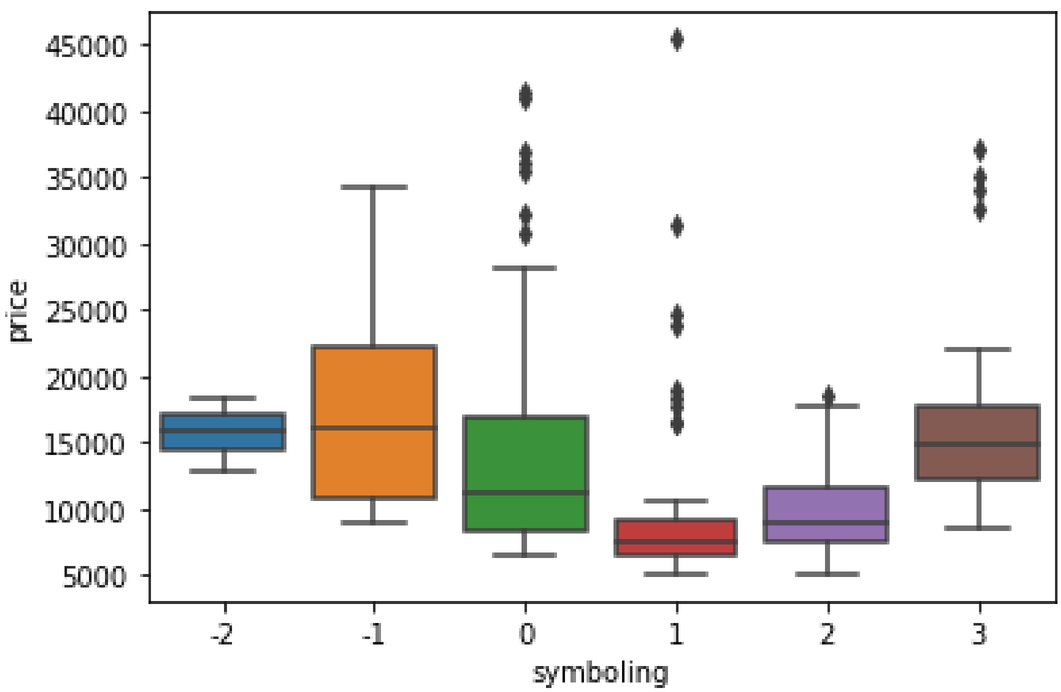


Scatter Plot for highwaympg

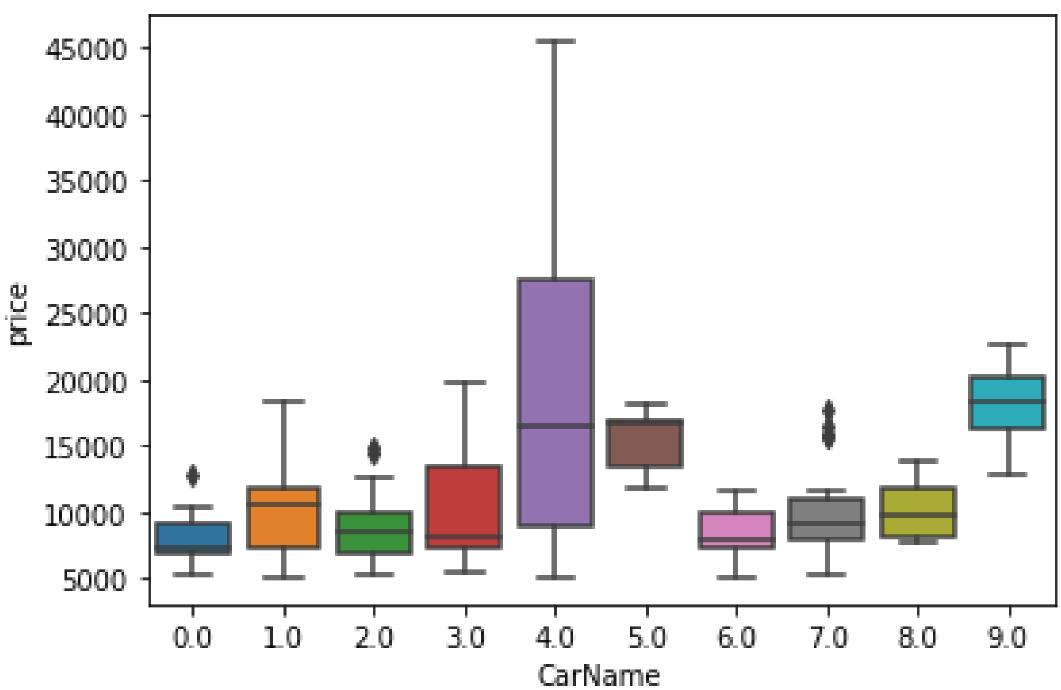



```
for i in df.drop('price',axis=1).columns:
    print("Box Plot for ",i)
    sns.boxplot(x = df[i], y = df['price'])
    plt.show()
    print('-'*100)
```

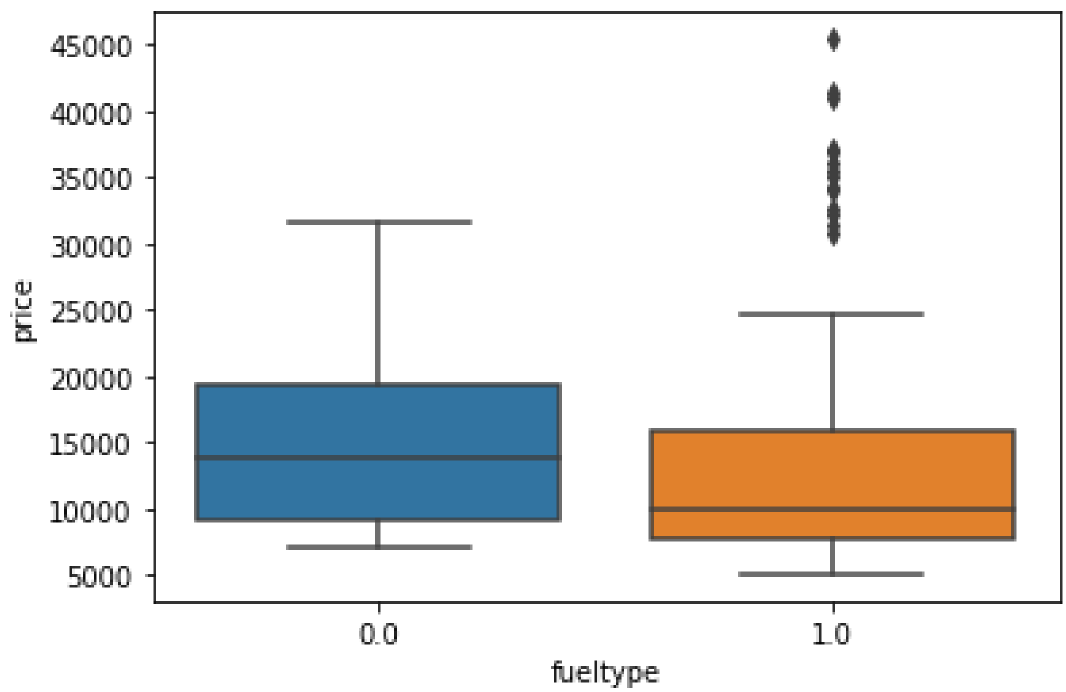
Box Plot for symboling



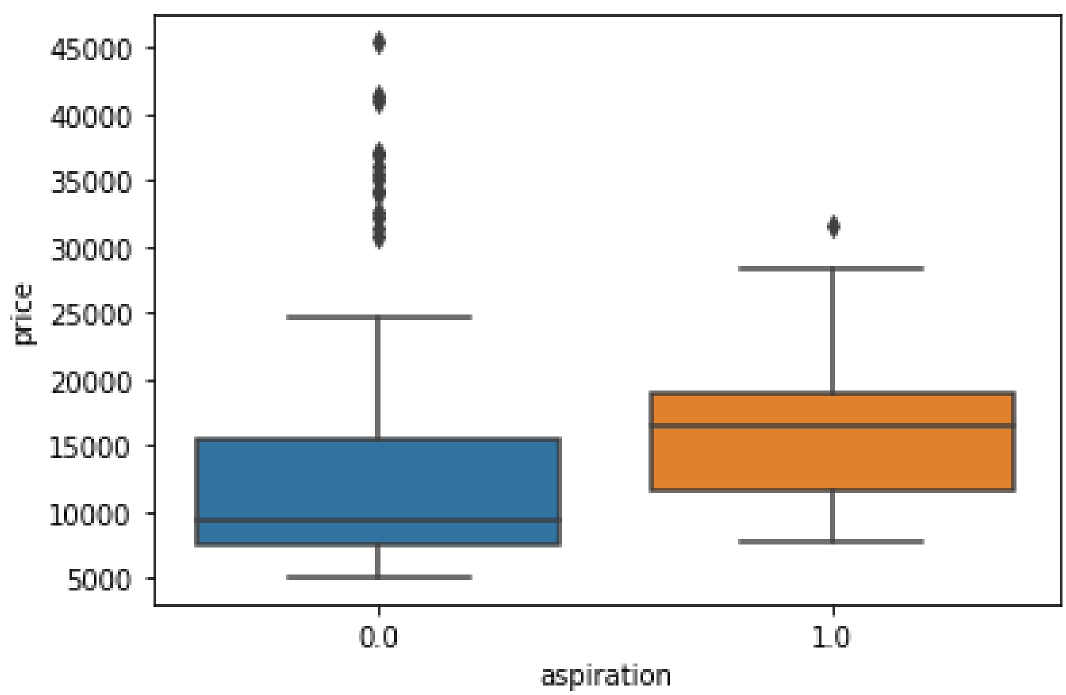
Box Plot for CarName



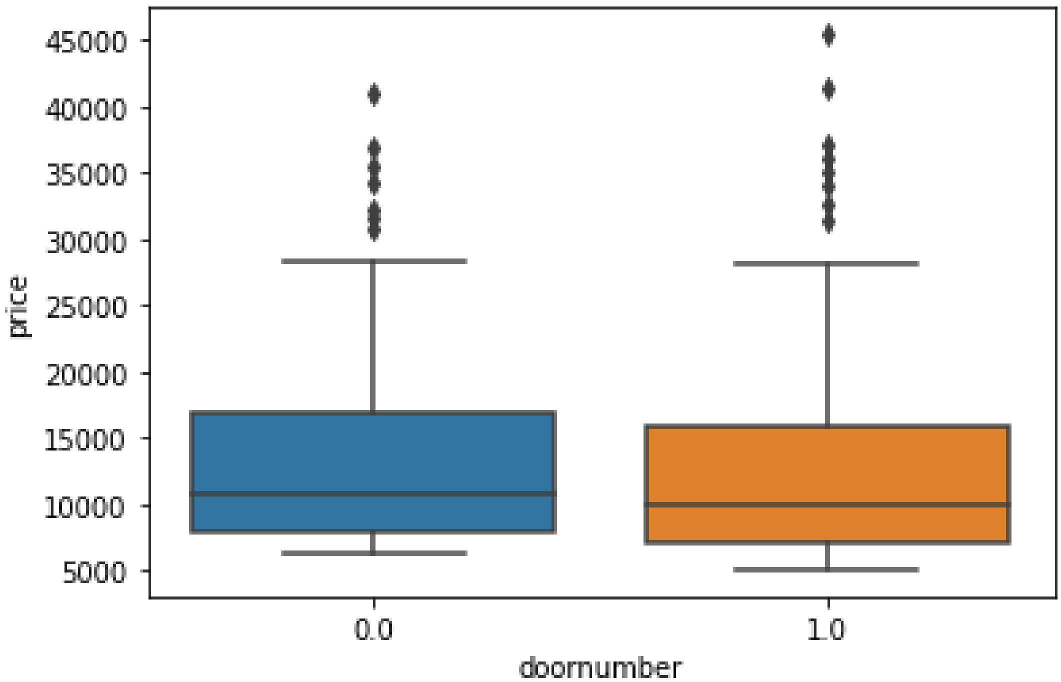
Box Plot for fueltype



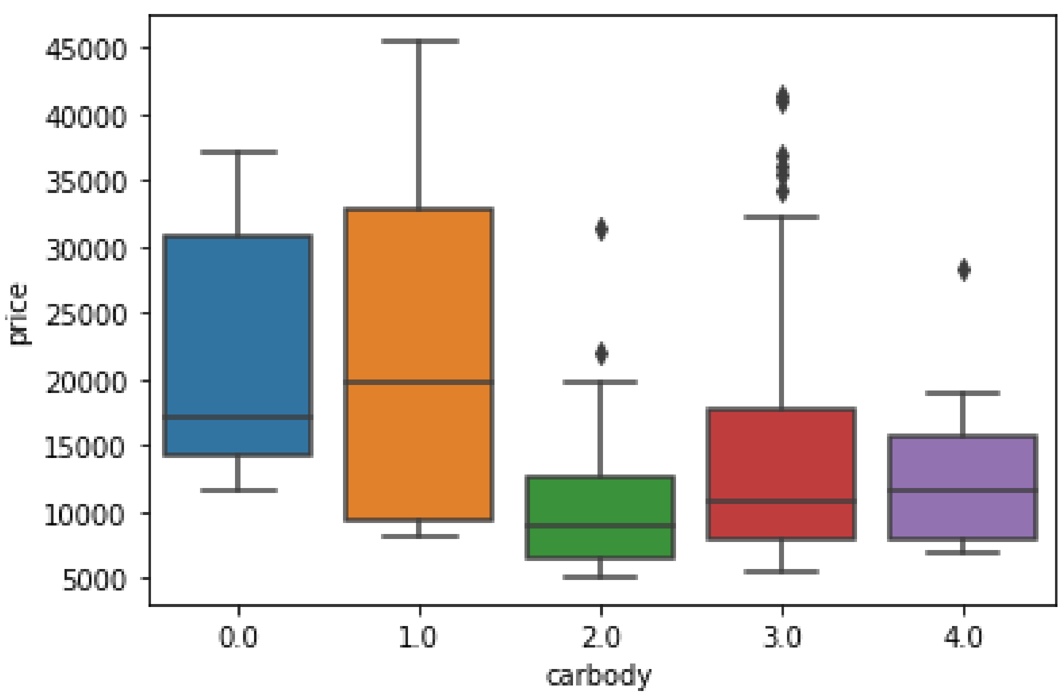

```
------------------------------
Box Plot for  aspiration
```



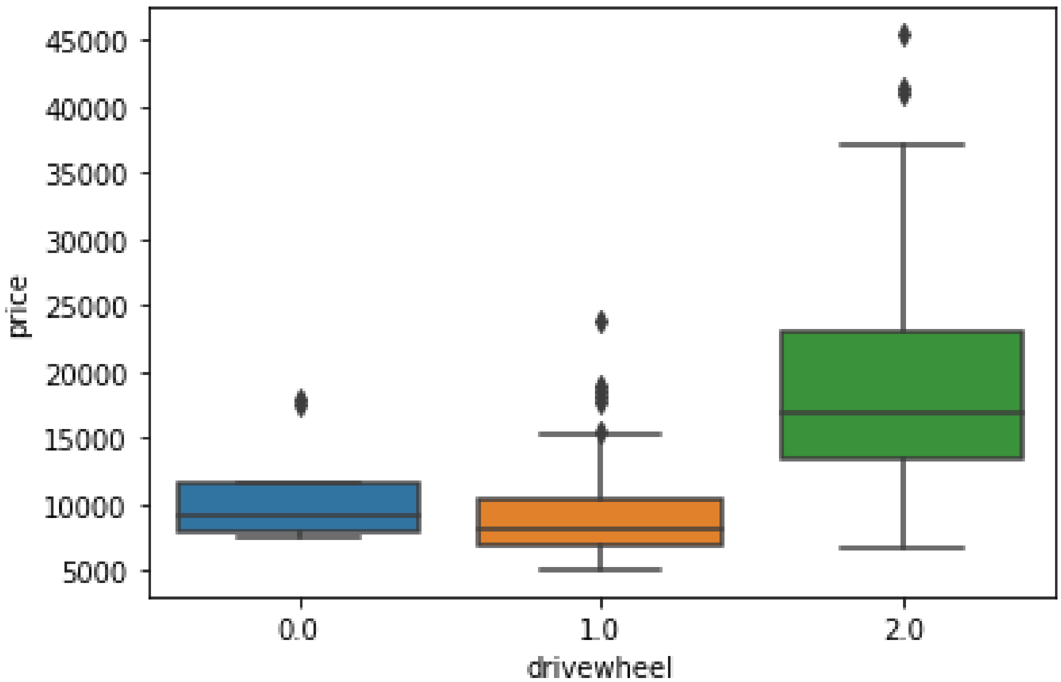

```
------------------------------
Box Plot for  doornumber
```



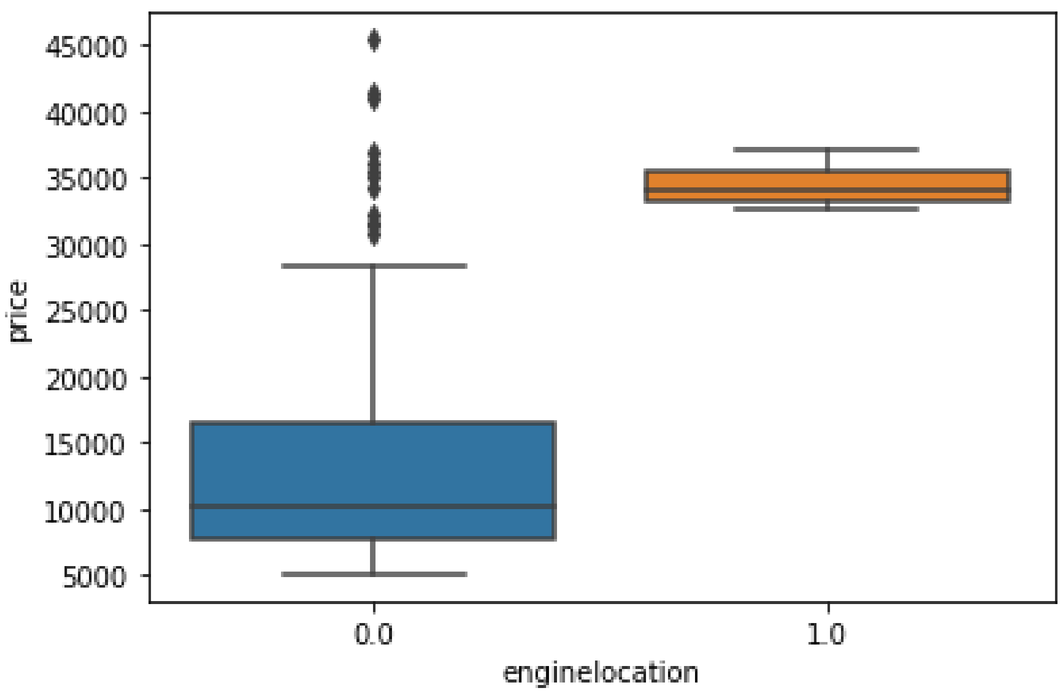
Box Plot for carbody



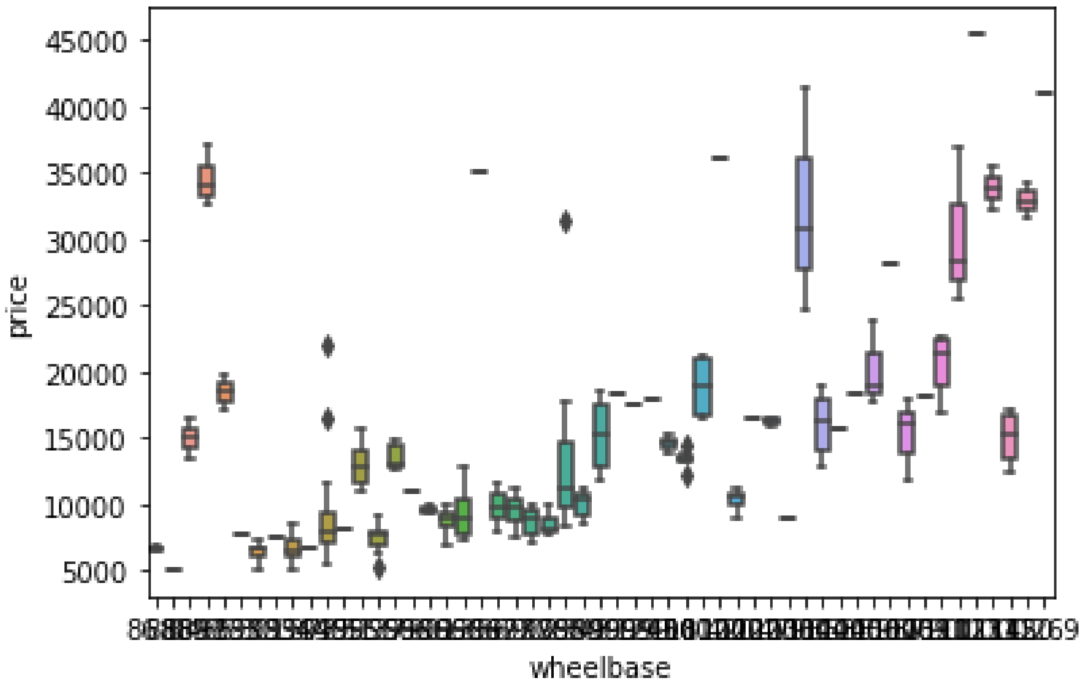
Box Plot for drivewheel



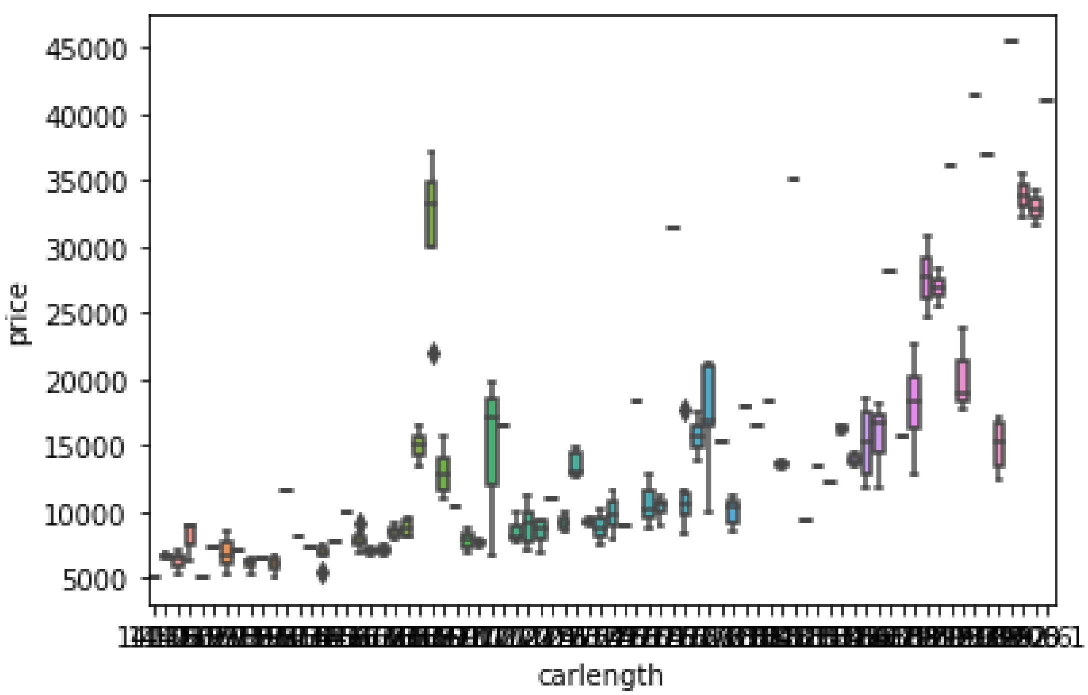

```
------------------------------
Box Plot for  enginelocation
```



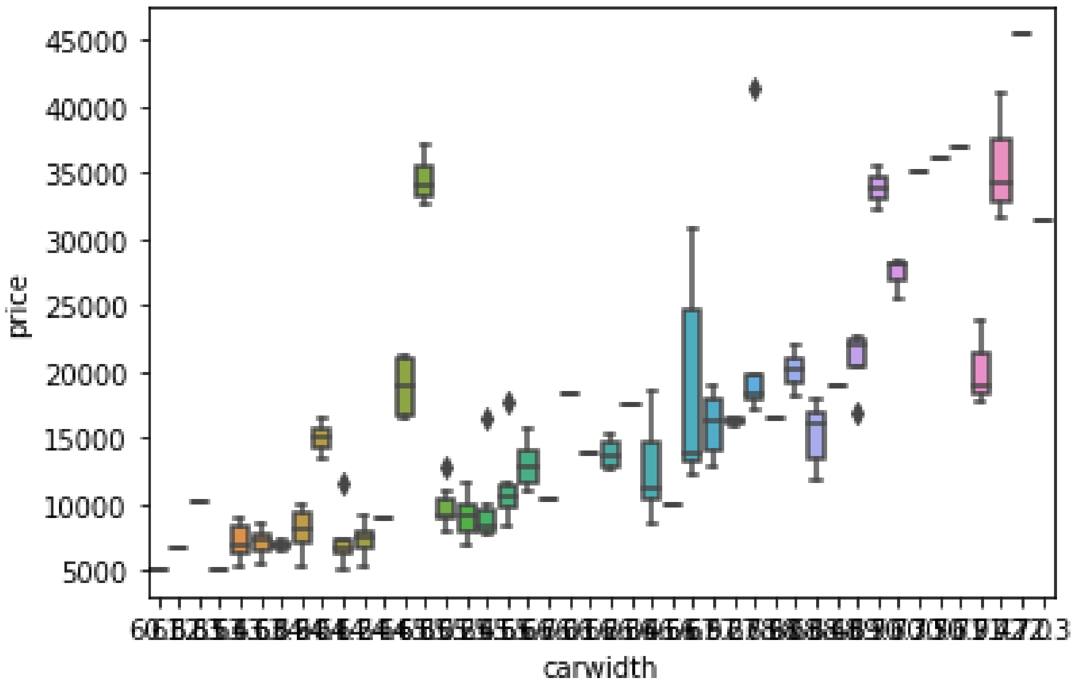

```
------------------------------
Box Plot for  wheelbase
```



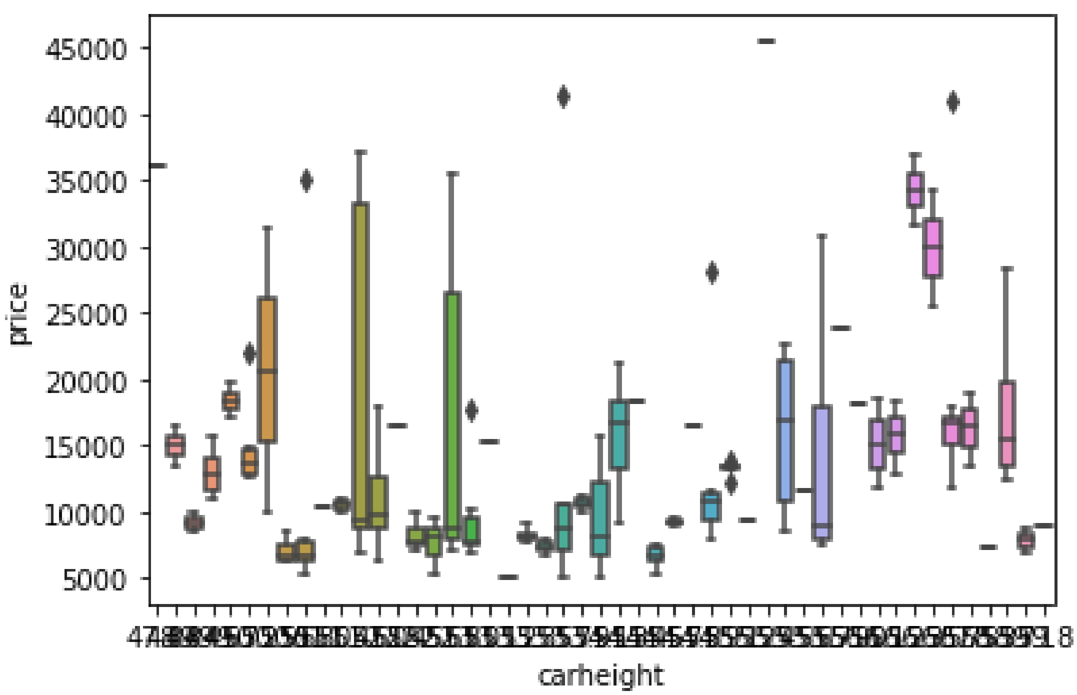
Box Plot for carlength



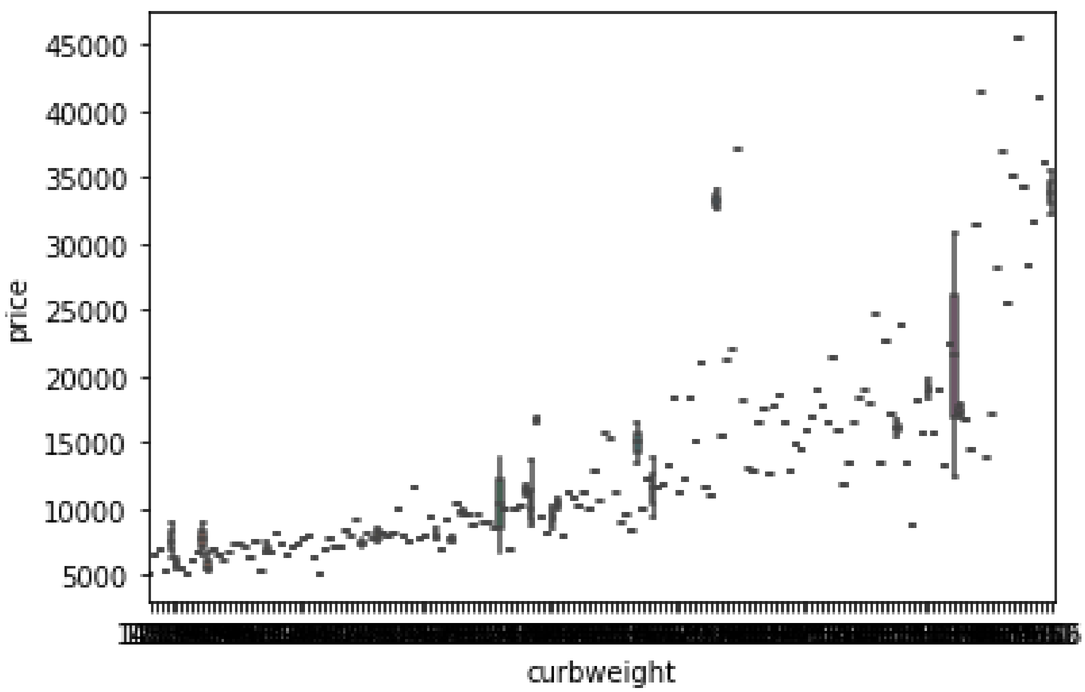
Box Plot for carwidth



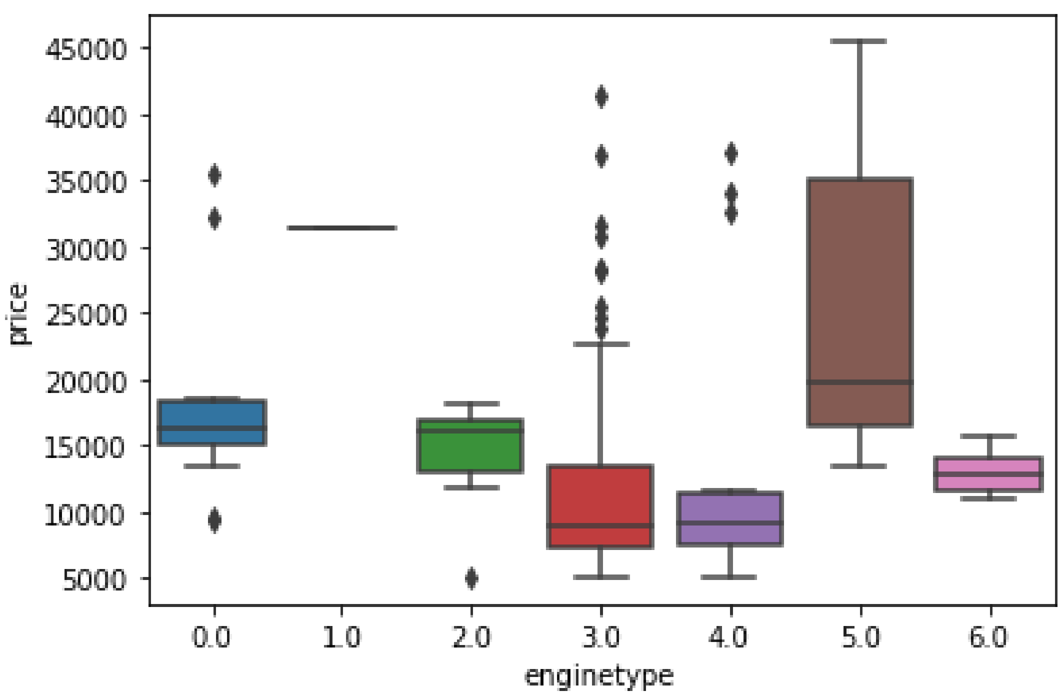
Box Plot for carheight



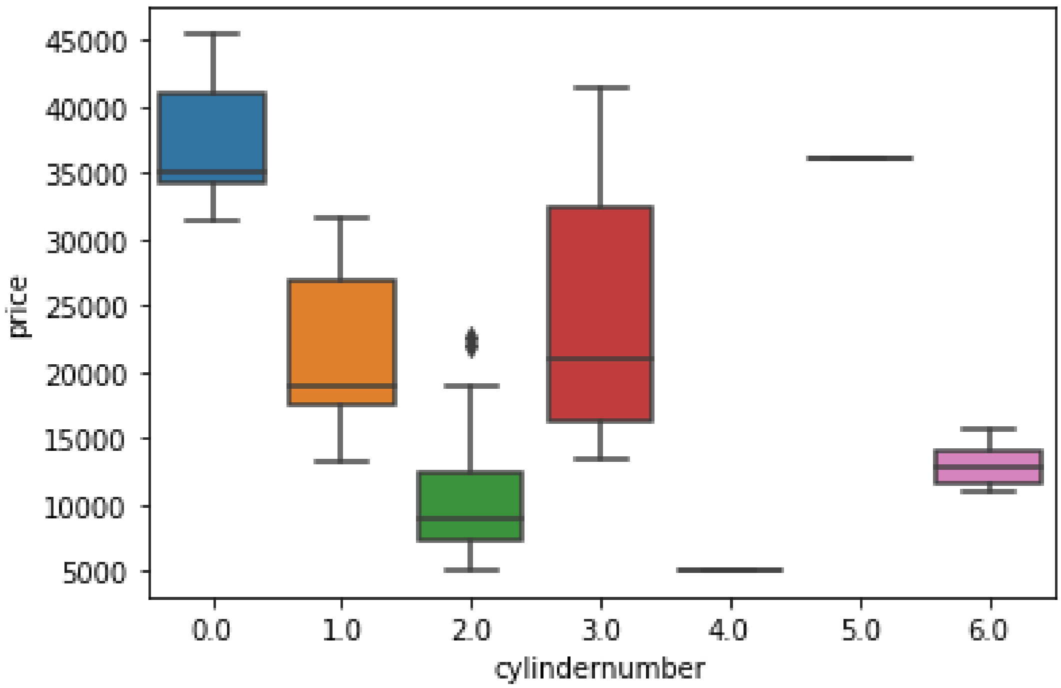
Box Plot for curbweight



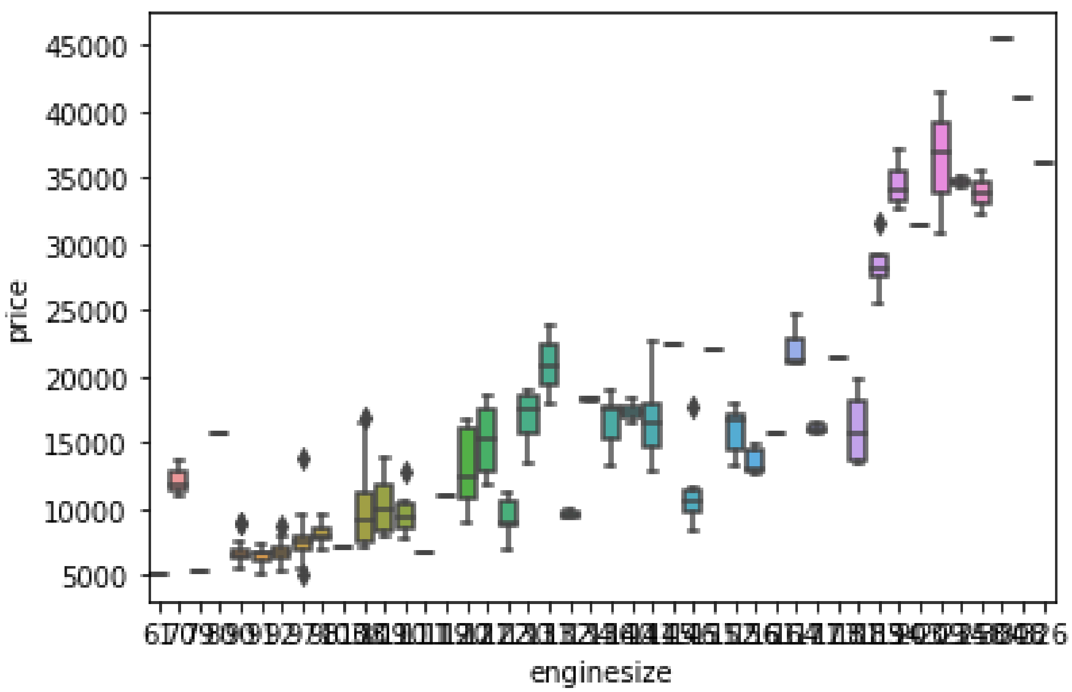
Box Plot for enginetype



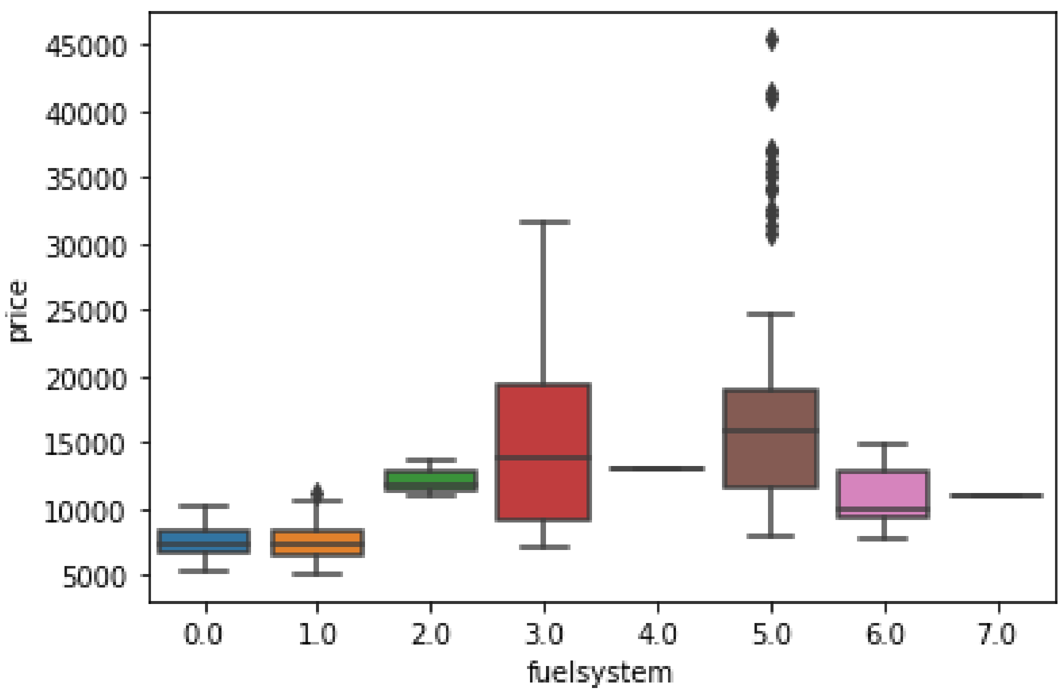
Box Plot for cylindernumber



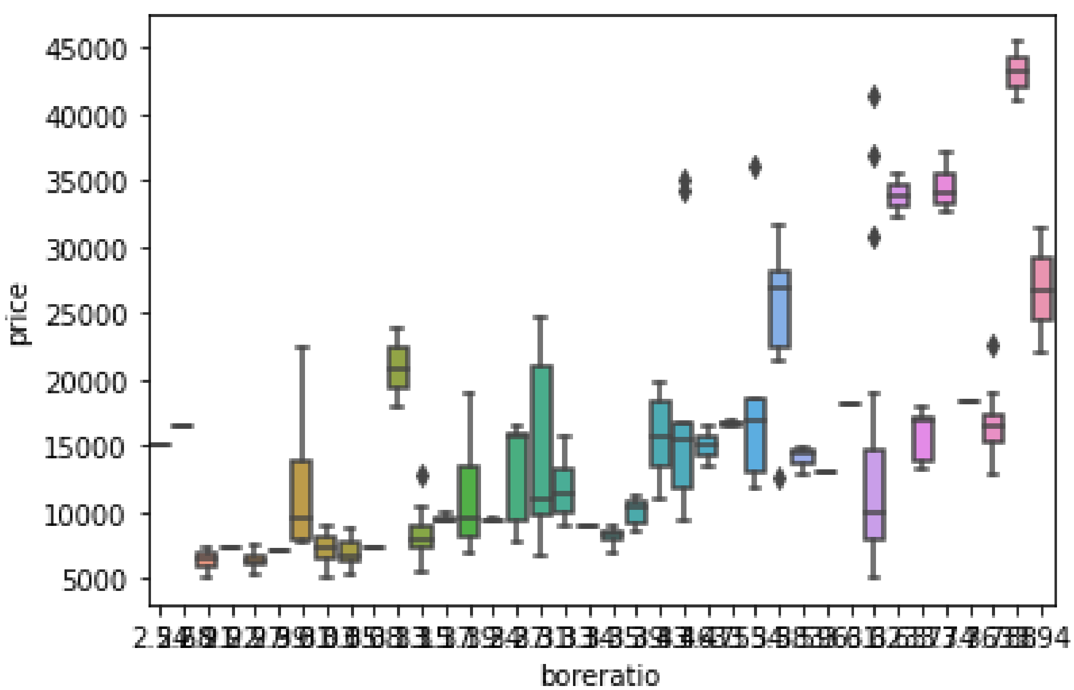
Box Plot for enginesize



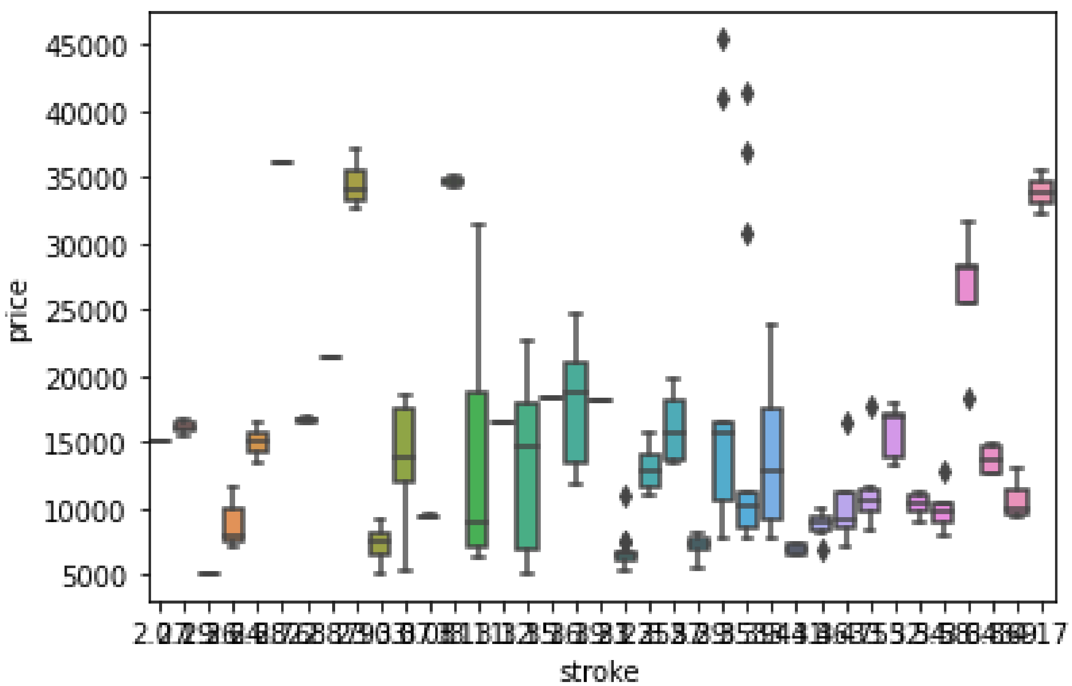
Box Plot for fuelsystem



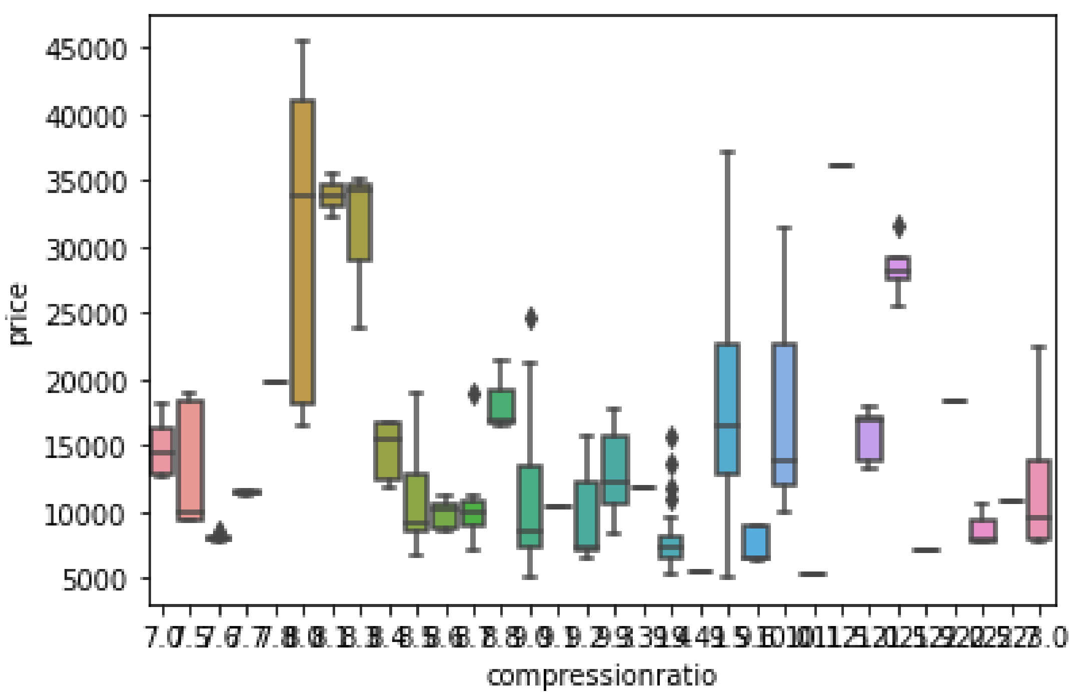

```
-----------------------------
Box Plot for  boreratio
```



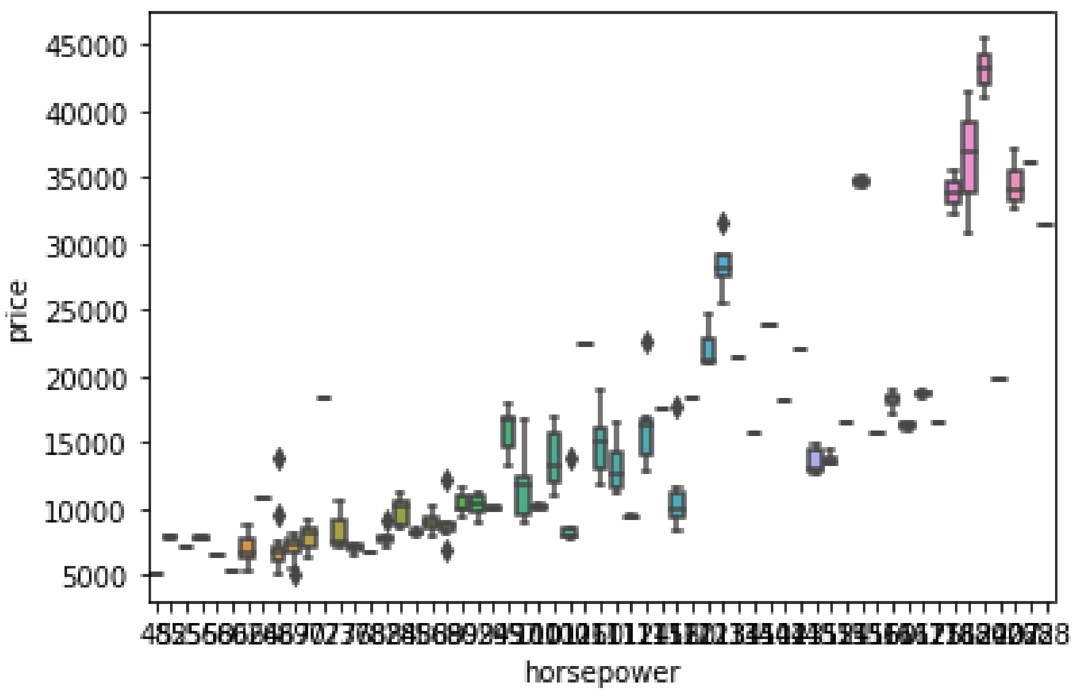

```
-----------------------------
Box Plot for  stroke
```



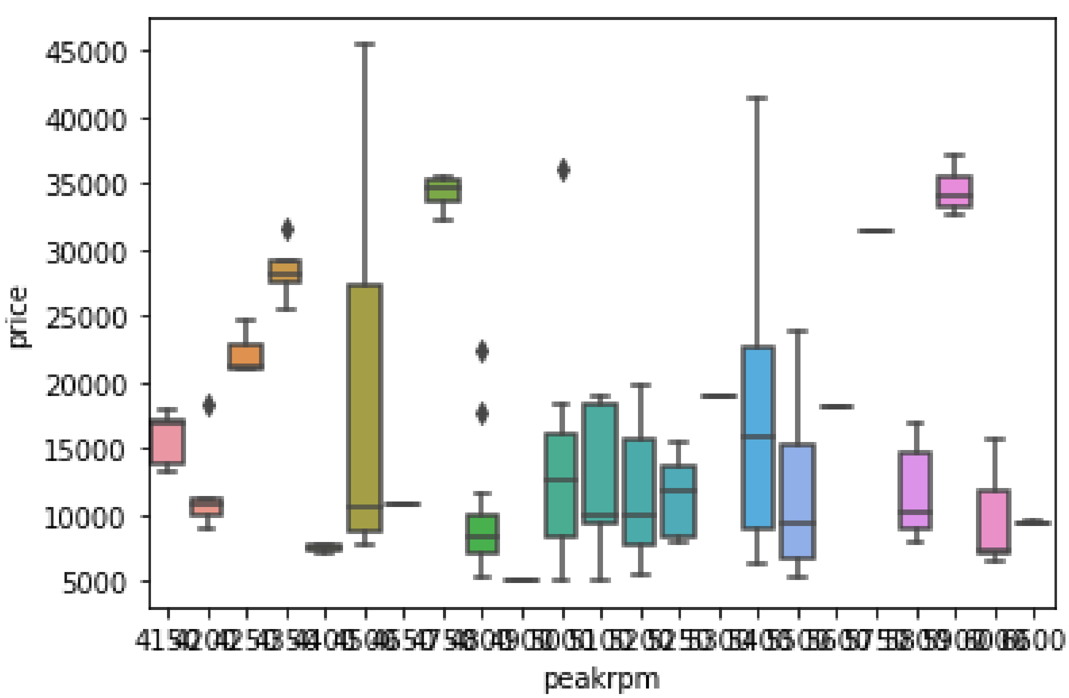
Box Plot for compressionratio



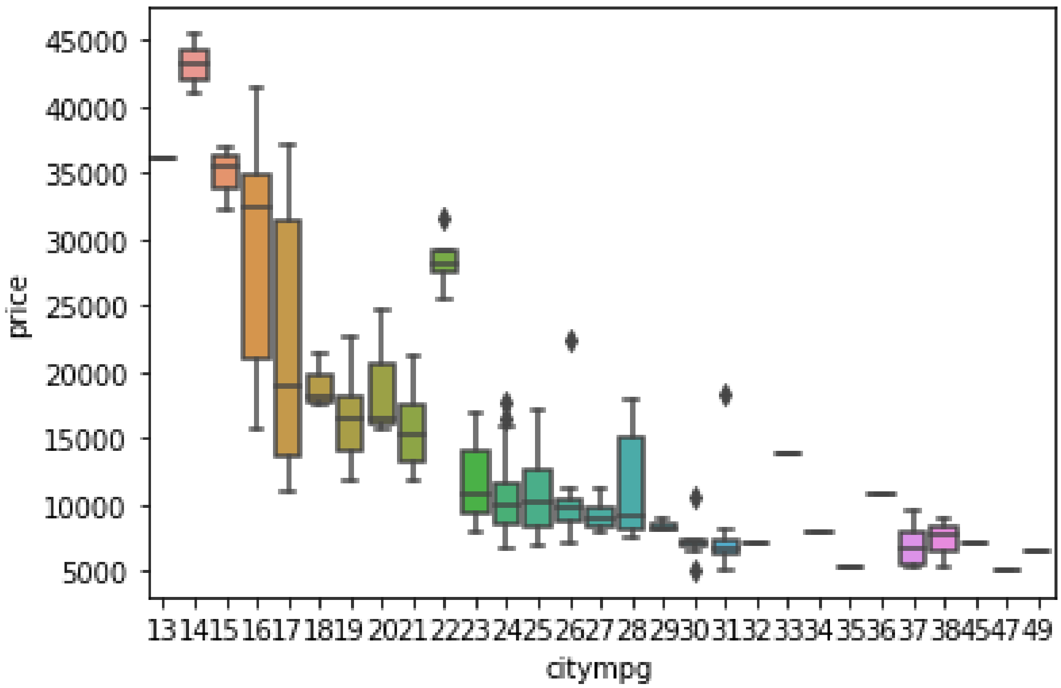
Box Plot for horsepower



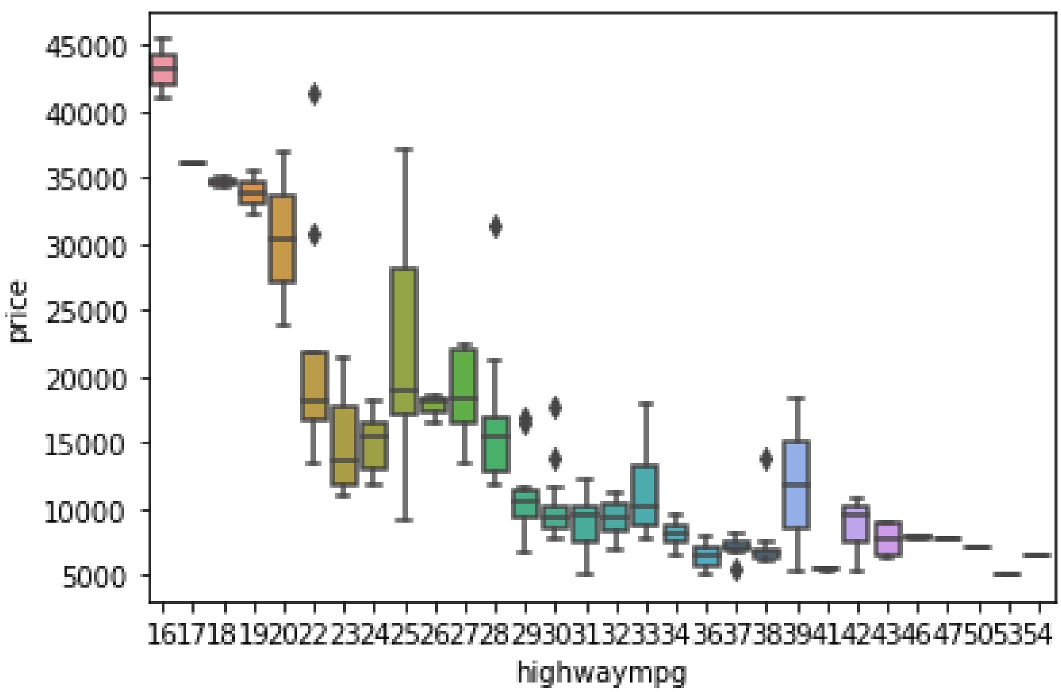
Box Plot for peakrpm



Box Plot for citympg



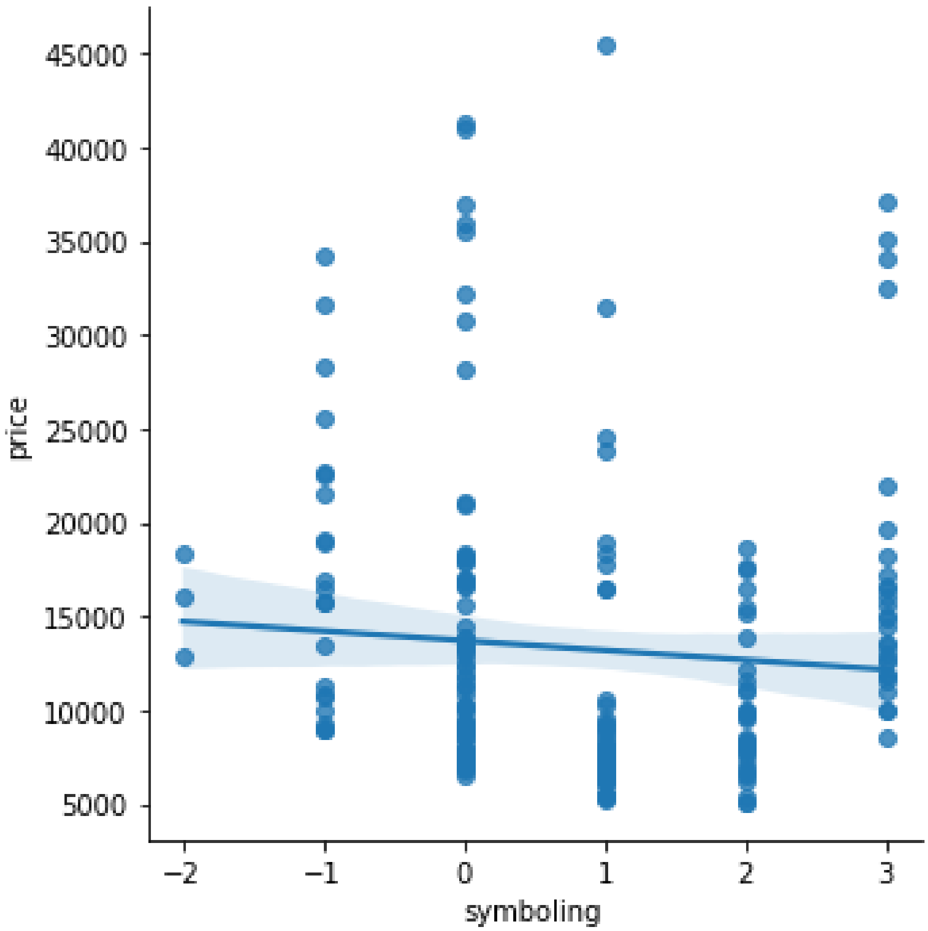

```
------------------------------
Box Plot for  highwaympg
```

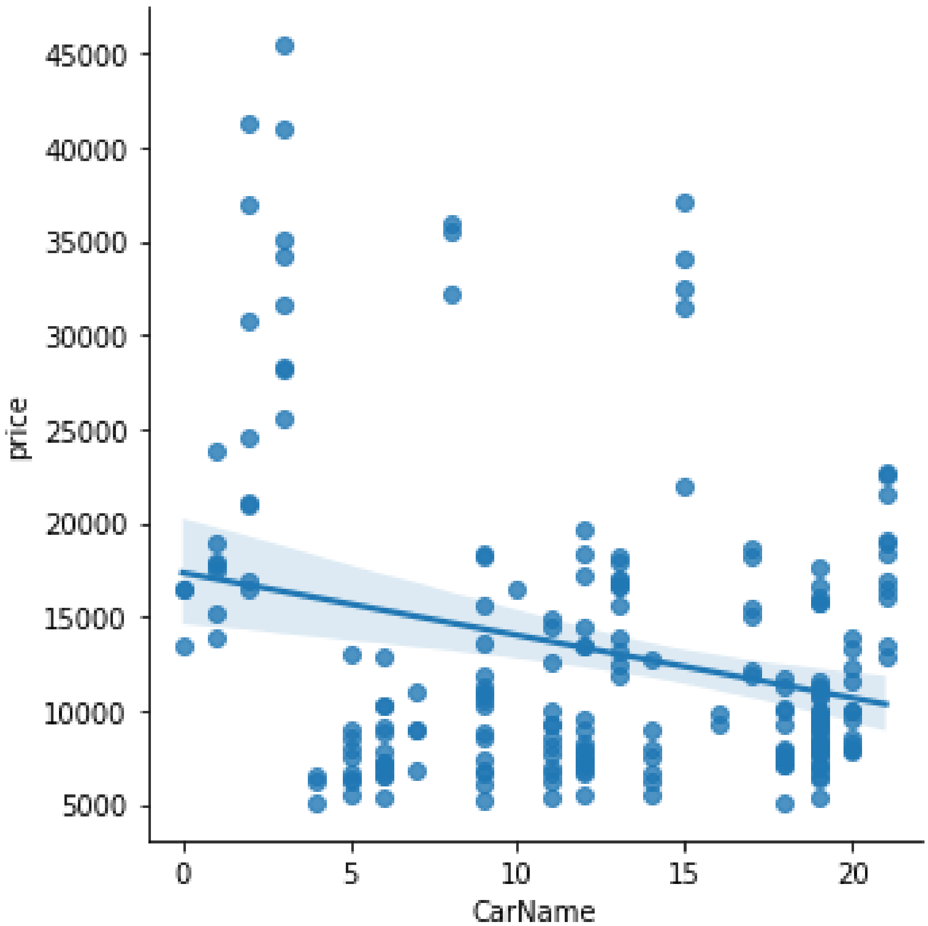



```
------------------------------
```

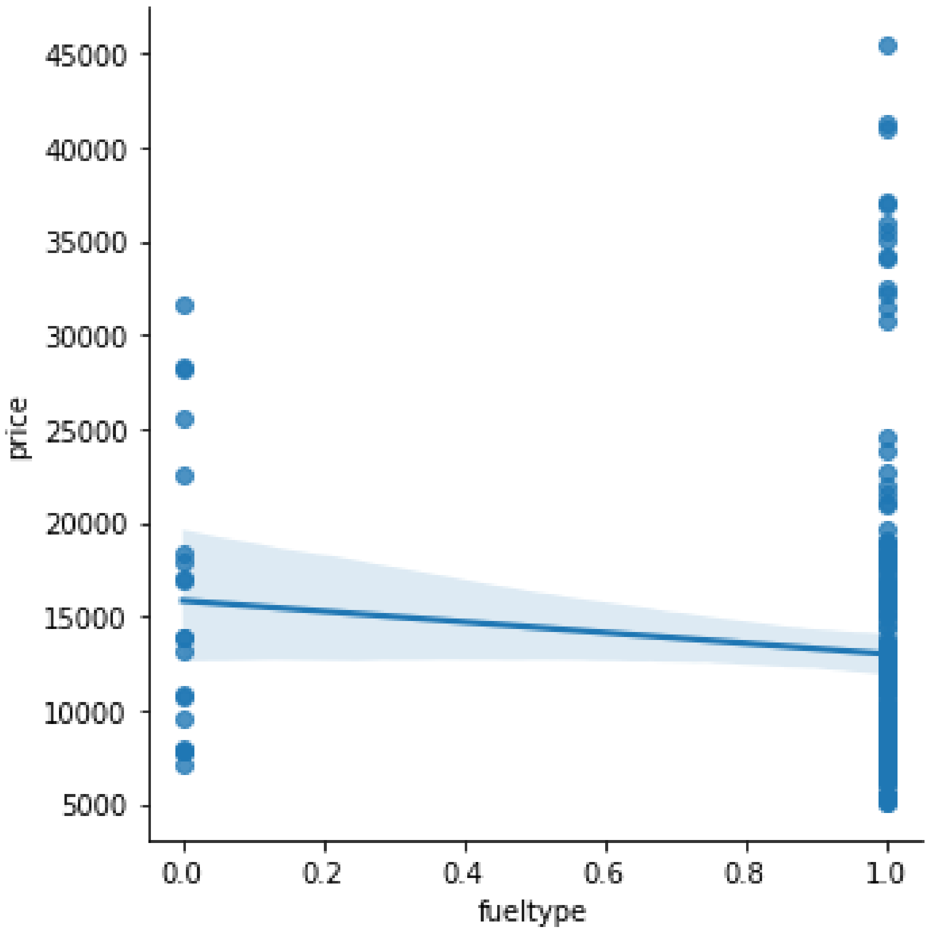
```python
for col in df.columns:
    if df[col].dtypes != 'object':
        sns.lmplot(data = df, x = col, y = 'price')
```

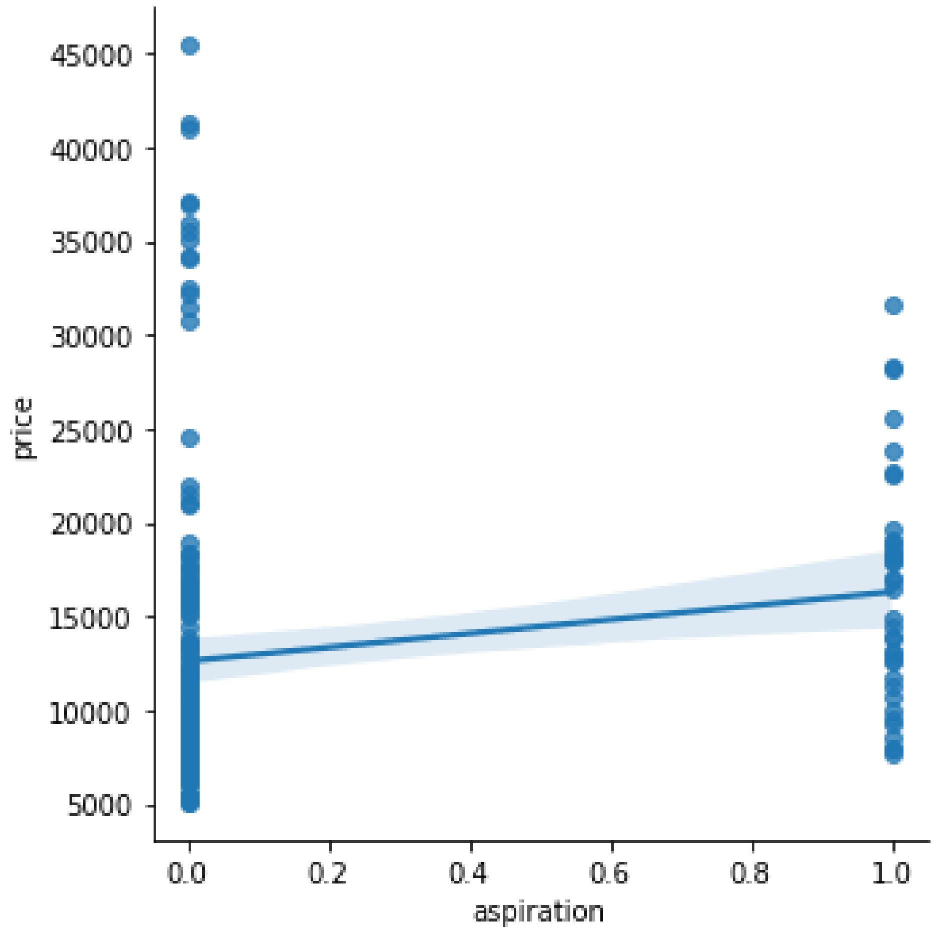
```
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:409:
RuntimeWarning: More than 20 figures have been opened. Figures created
through the pyplot interface (`matplotlib.pyplot.figure`) are retained
until explicitly closed and may consume too much memory. (To control
this warning, see the rcParam `figure.max_open_warning`).
  fig = plt.figure(figsize=figsize)
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:409:
RuntimeWarning: More than 20 figures have been opened. Figures created
through the pyplot interface (`matplotlib.pyplot.figure`) are retained
until explicitly closed and may consume too much memory. (To control
this warning, see the rcParam `figure.max_open_warning`).
  fig = plt.figure(figsize=figsize)
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:409:
RuntimeWarning: More than 20 figures have been opened. Figures created
through the pyplot interface (`matplotlib.pyplot.figure`) are retained
until explicitly closed and may consume too much memory. (To control
this warning, see the rcParam `figure.max_open_warning`).
  fig = plt.figure(figsize=figsize)
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:409:
RuntimeWarning: More than 20 figures have been opened. Figures created
through the pyplot interface (`matplotlib.pyplot.figure`) are retained
until explicitly closed and may consume too much memory. (To control
this warning, see the rcParam `figure.max_open_warning`).
  fig = plt.figure(figsize=figsize)
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:409:
RuntimeWarning: More than 20 figures have been opened. Figures created
through the pyplot interface (`matplotlib.pyplot.figure`) are retained
until explicitly closed and may consume too much memory. (To control
this warning, see the rcParam `figure.max_open_warning`).
  fig = plt.figure(figsize=figsize)
```


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

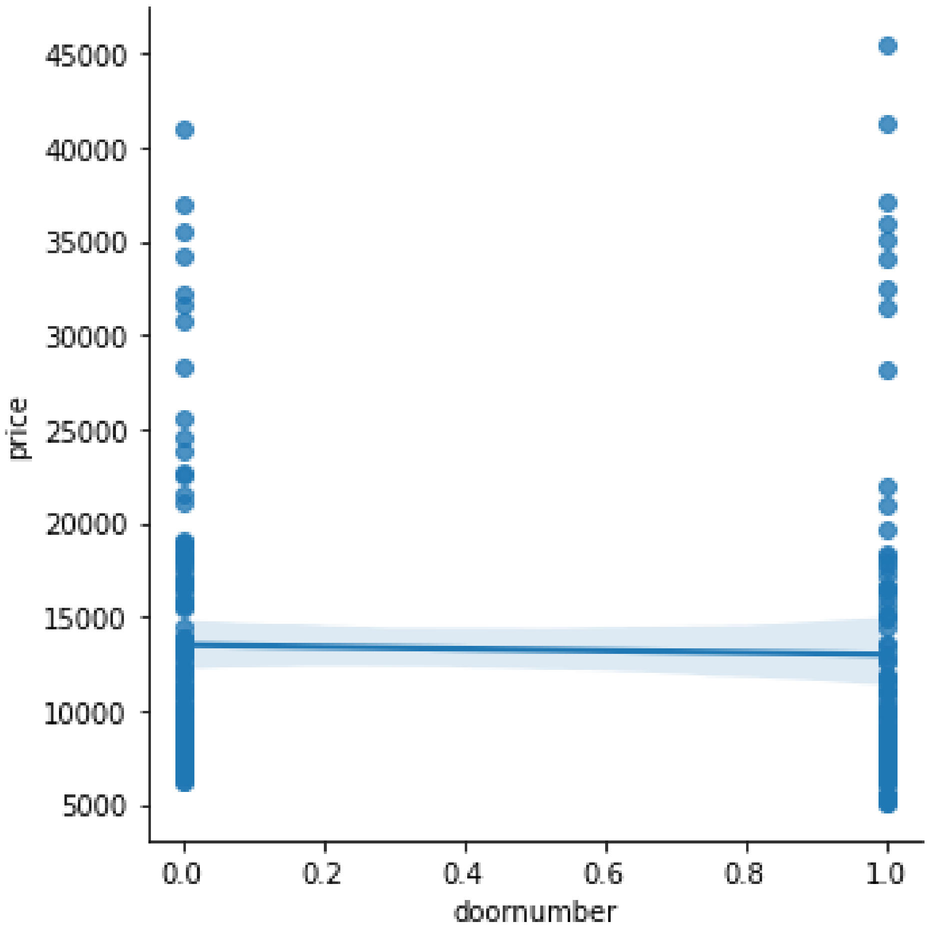


price
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
5
10
15
20
CarName

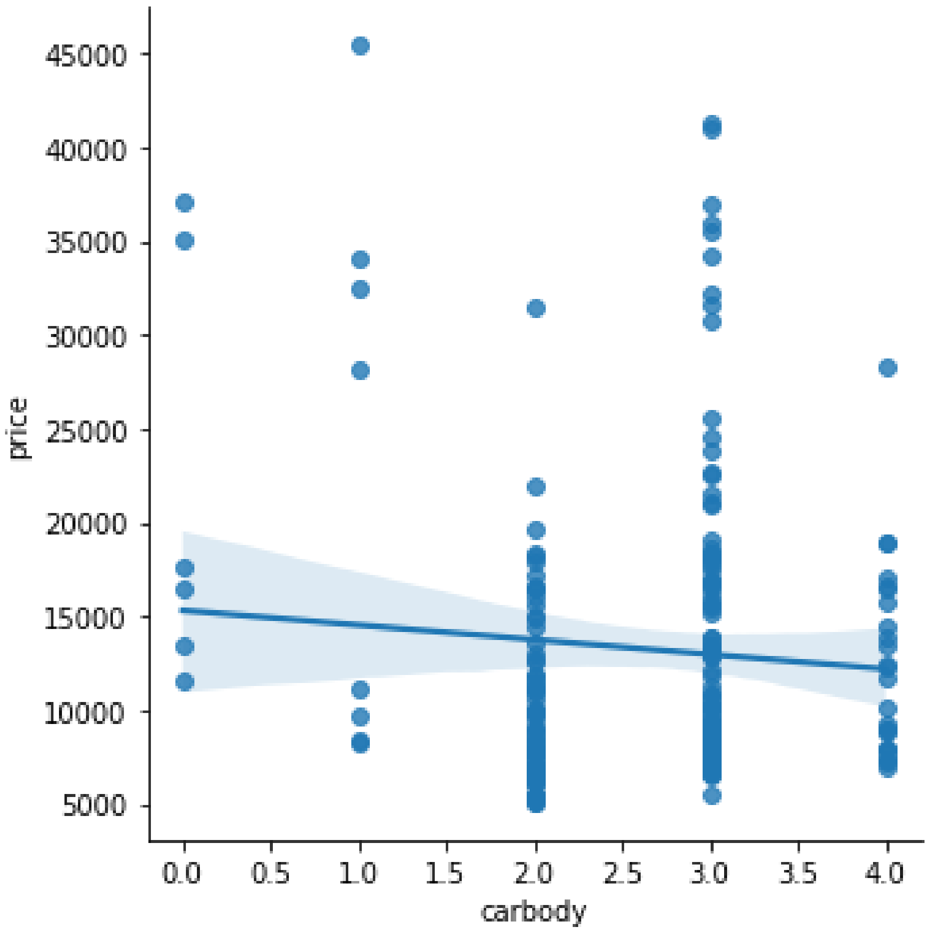

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
0.0
0.2
0.4
0.6
0.8
1.0
fueltype

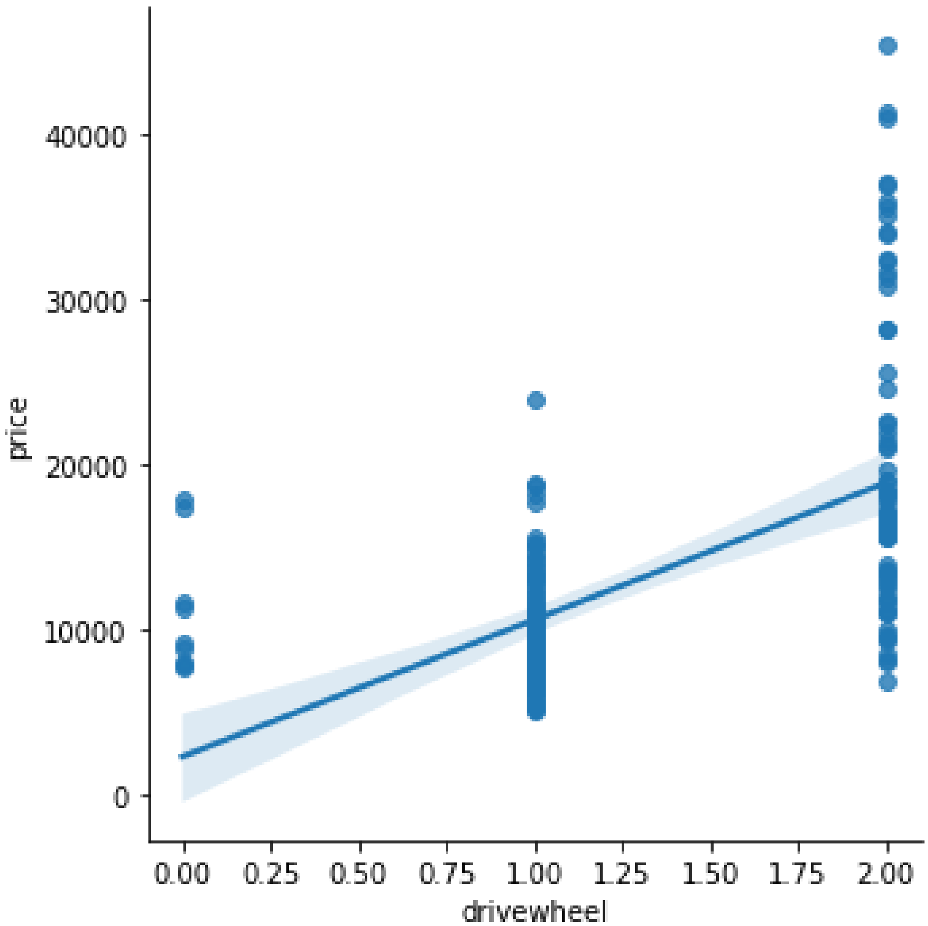


45000
40000
35000
30000
25000
20000
15000
10000
5000
price
0.0
0.2
0.4
0.6
0.8
1.0
aspiration

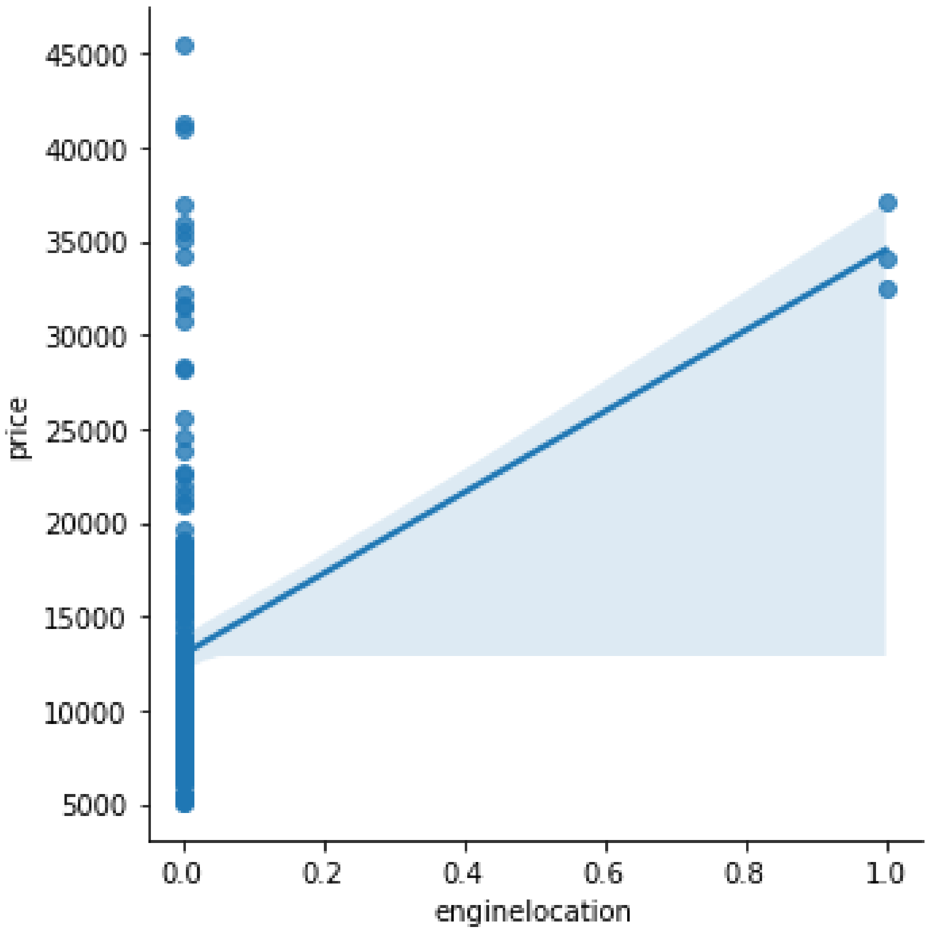


45000
40000
35000
30000
25000
20000
15000
10000
5000
price
0.0
0.2
0.4
0.6
0.8
1.0
doornumber

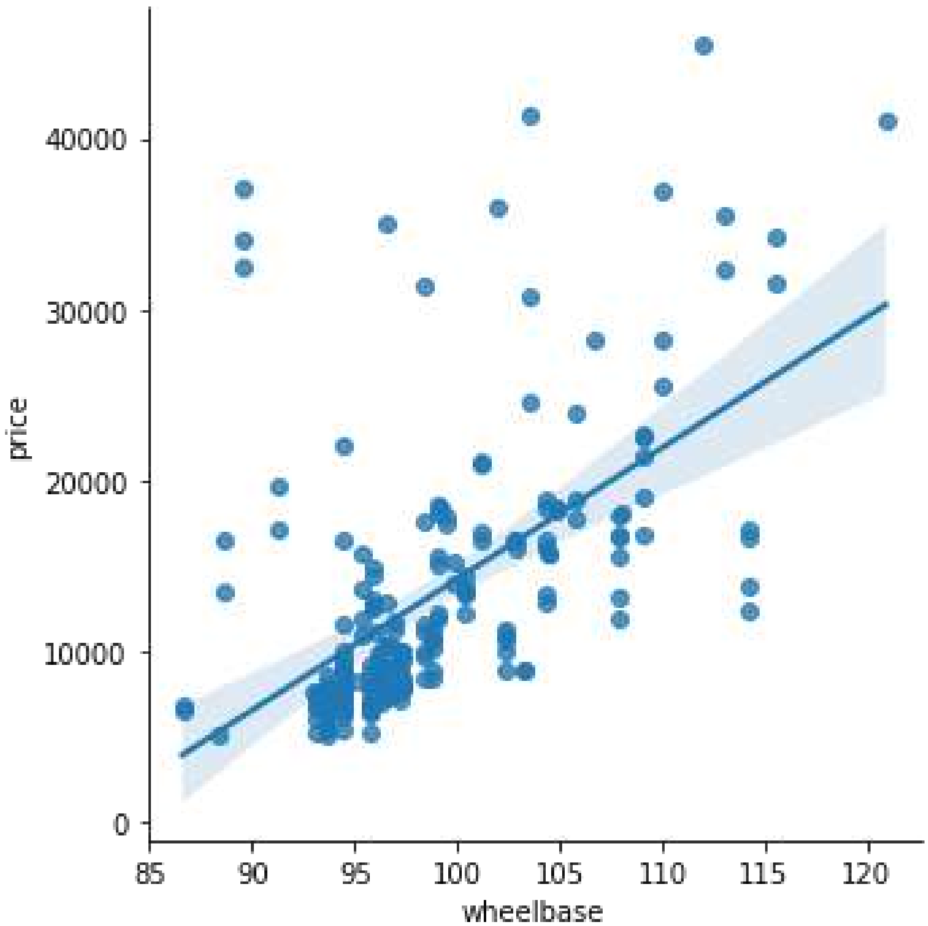

price
45000
40000
35000
30000
25000
20000
15000
10000
5000
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
carbody



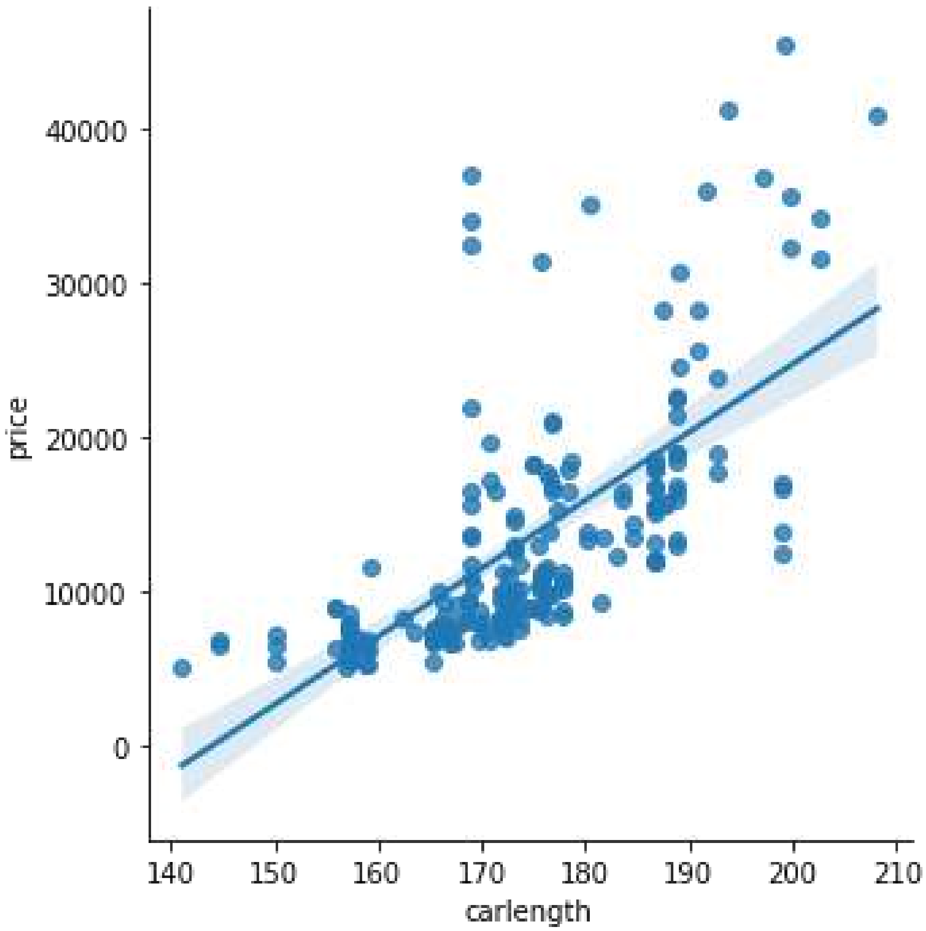
40000
30000
20000
10000
0
price
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
drivewheel



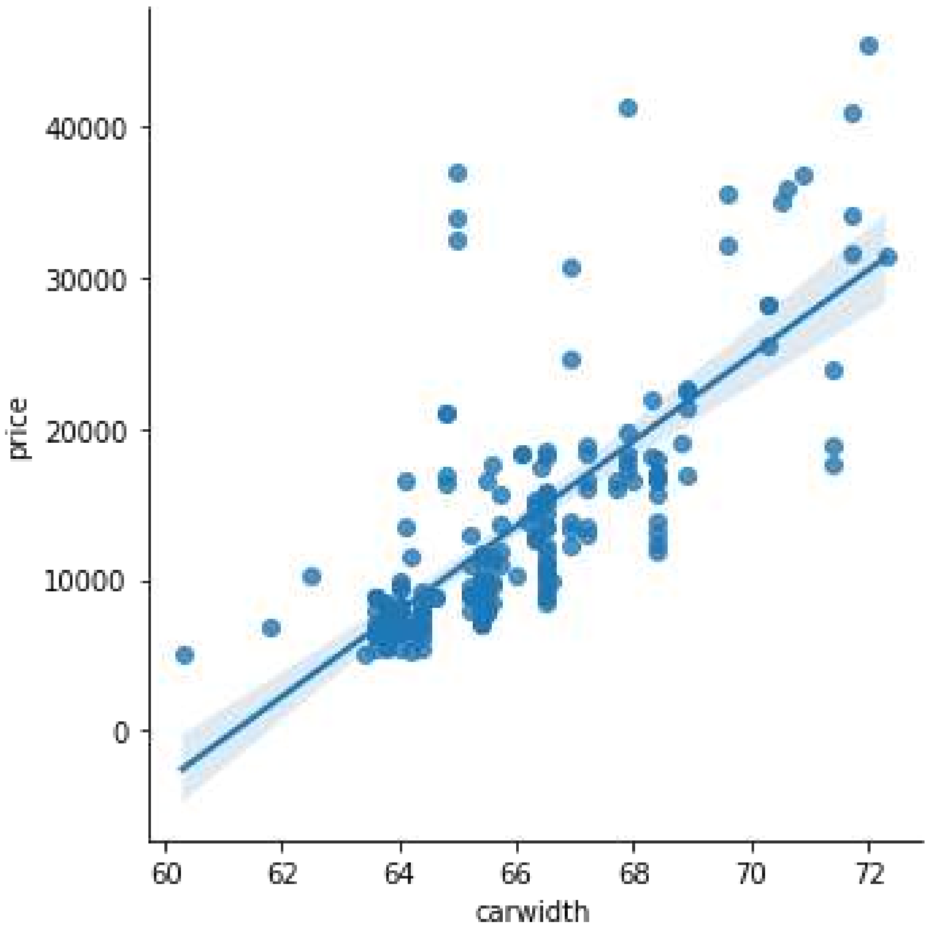
45000
40000
35000
30000
25000
20000
15000
10000
5000
price
0.0
0.2
0.4
0.6
0.8
1.0
enginelocation

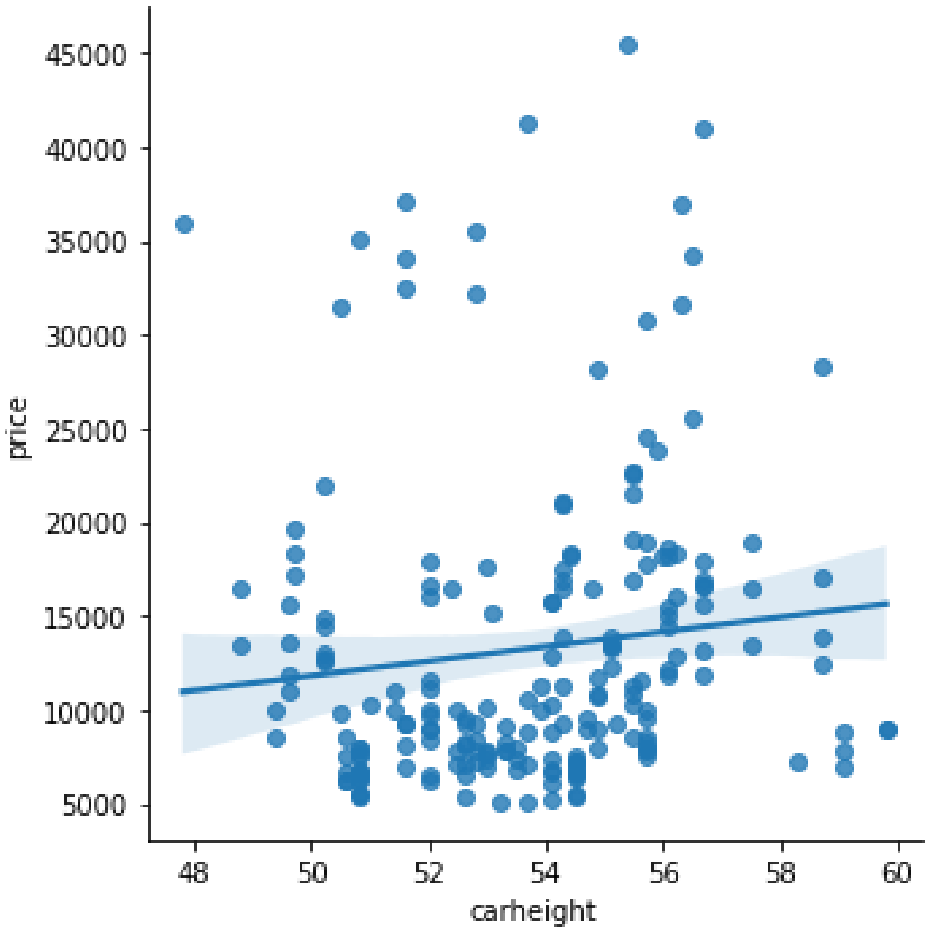


price
40000
30000
20000
10000
0
85
90
95
100
105
110
115
120
wheelbase

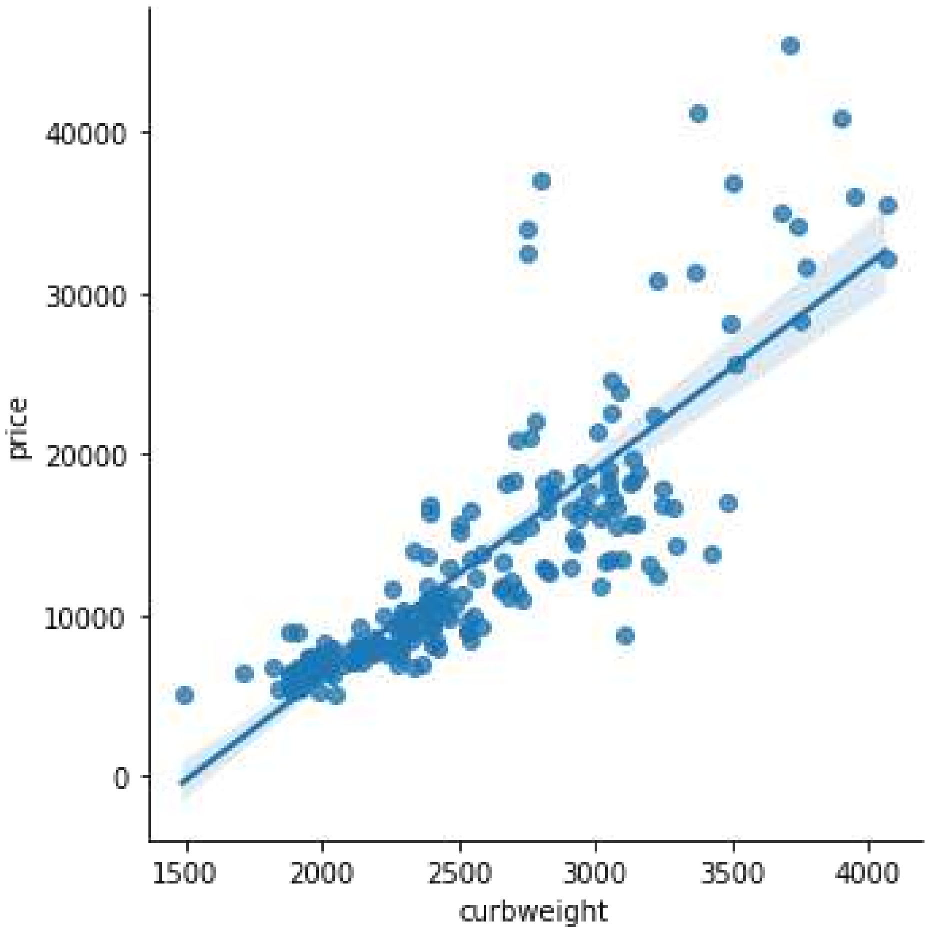


40000
30000
20000
10000
0
price
140
150
160
170
180
190
200
210
carlength

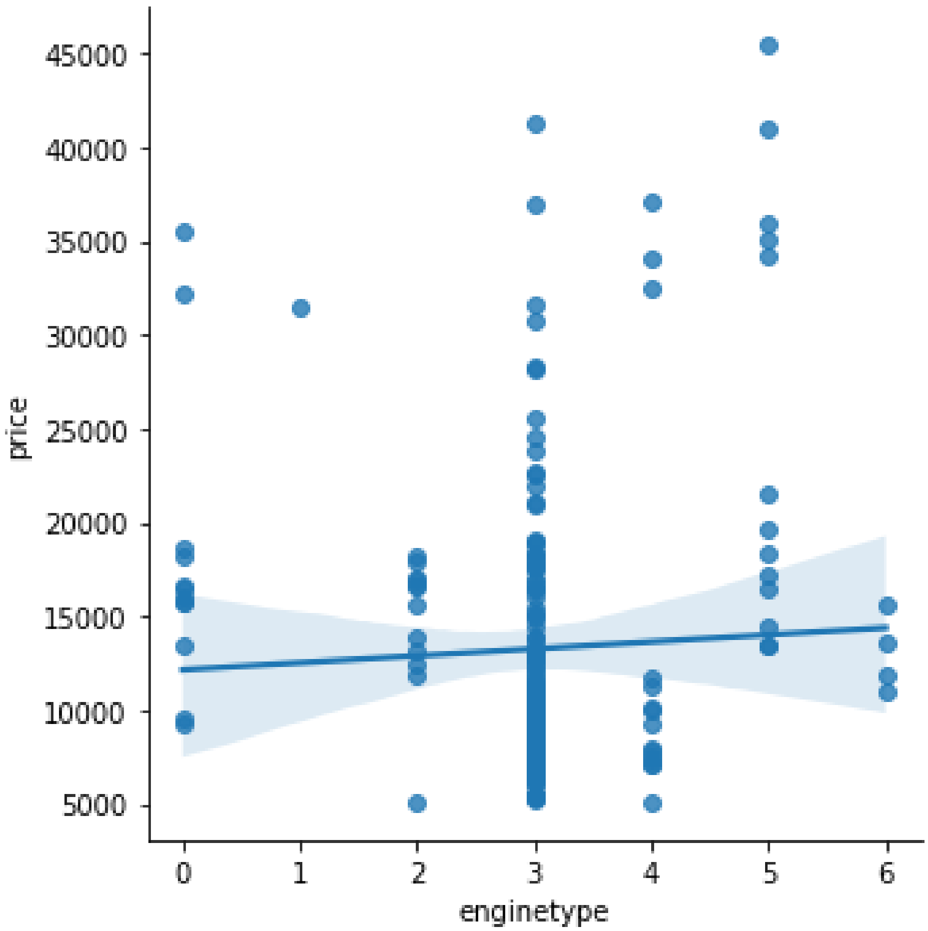

40000
30000
20000
10000
0
price
60
62
64
66
68
70
72
carwidth



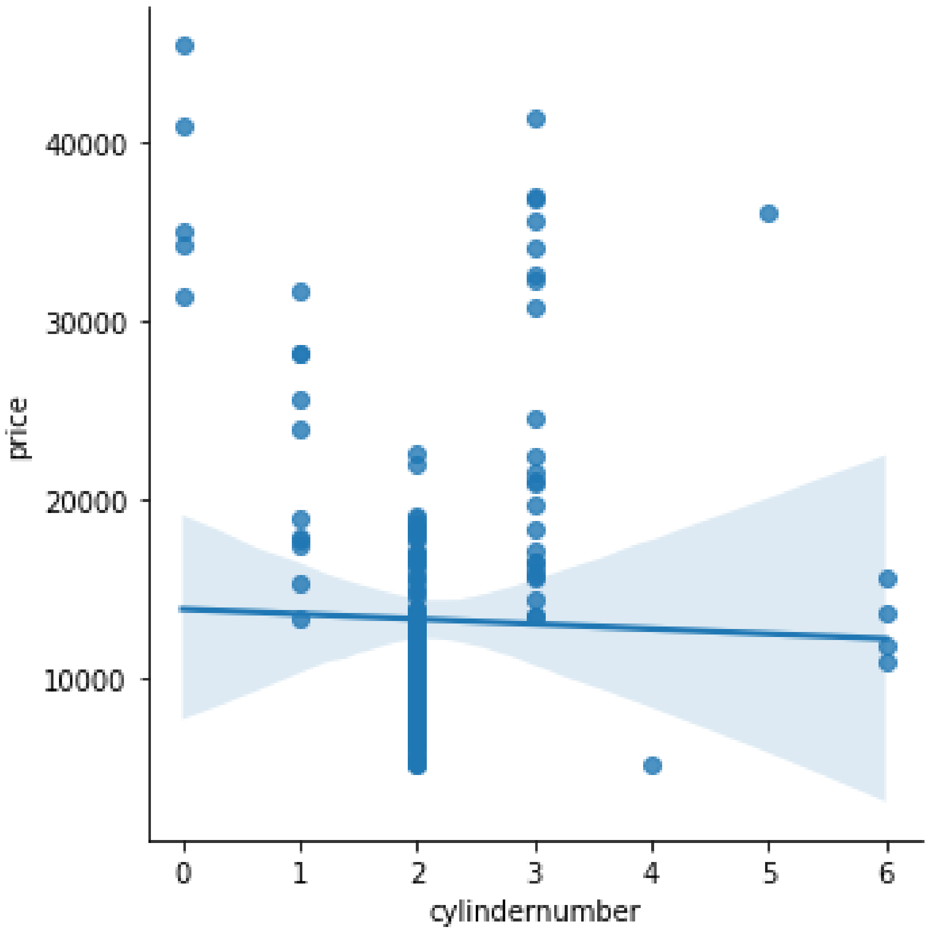
45000
40000
35000
30000
25000
20000
15000
10000
5000
price
48
50
52
54
56
58
60
carheight



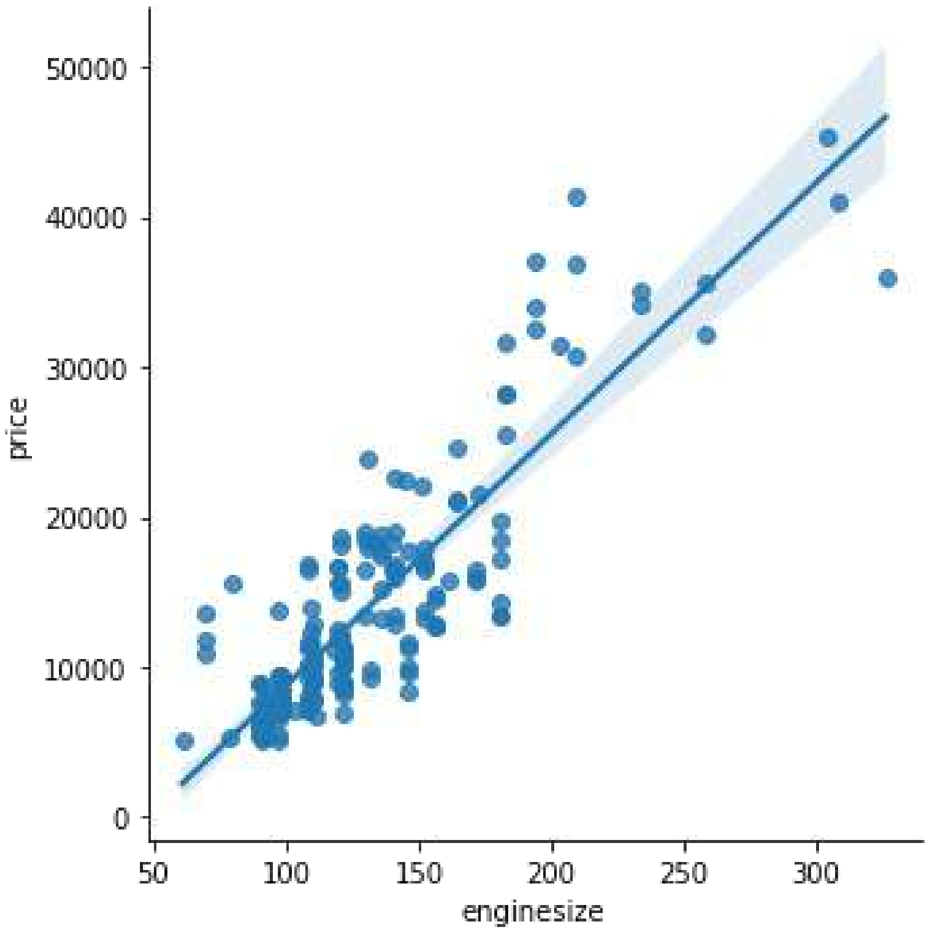
40000
30000
20000
10000
0
price
1500
2000
2500
3000
3500
4000
curbweight



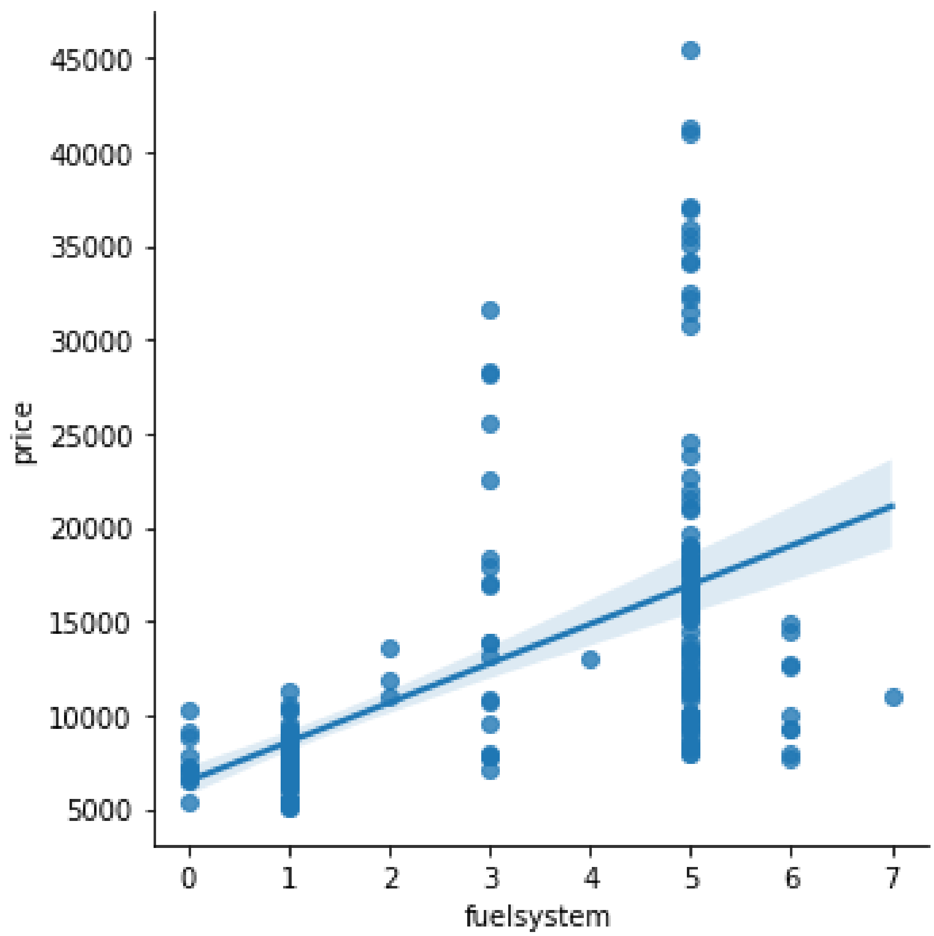
45000
40000
35000
30000
25000
20000
15000
10000
5000
price
0
1
2
3
4
5
6
enginetype



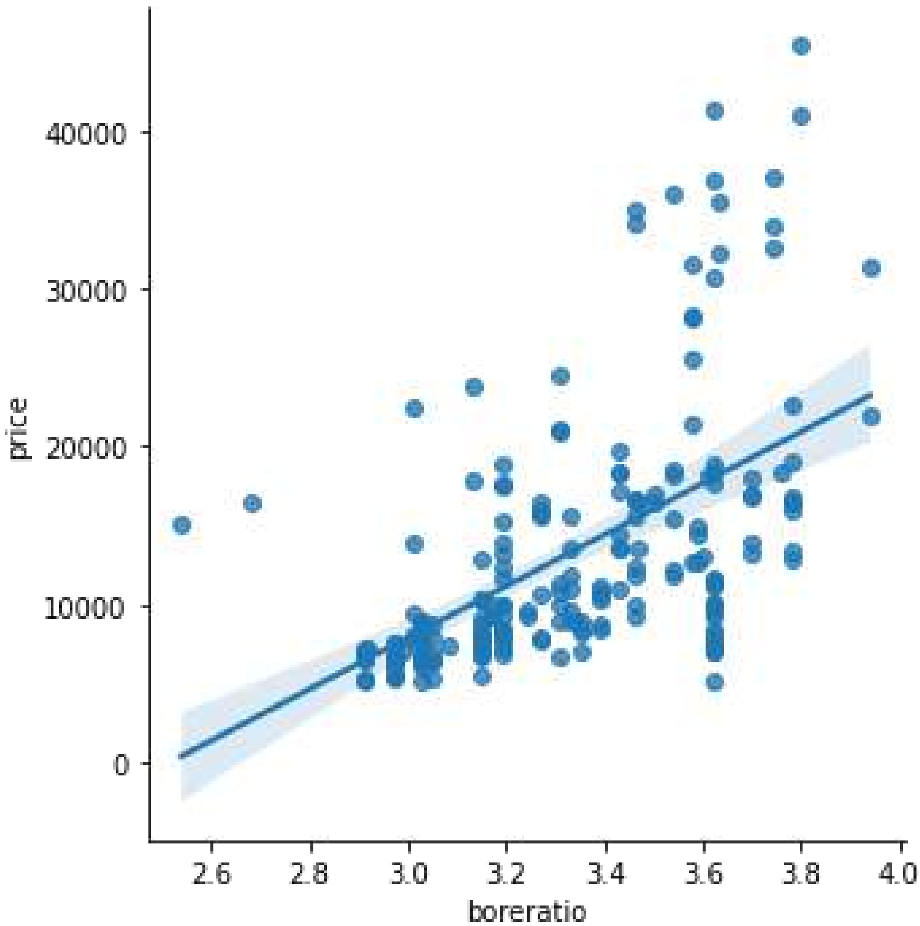
price
40000
30000
20000
10000
0
1
2
3
4
5
6
cylindernumber



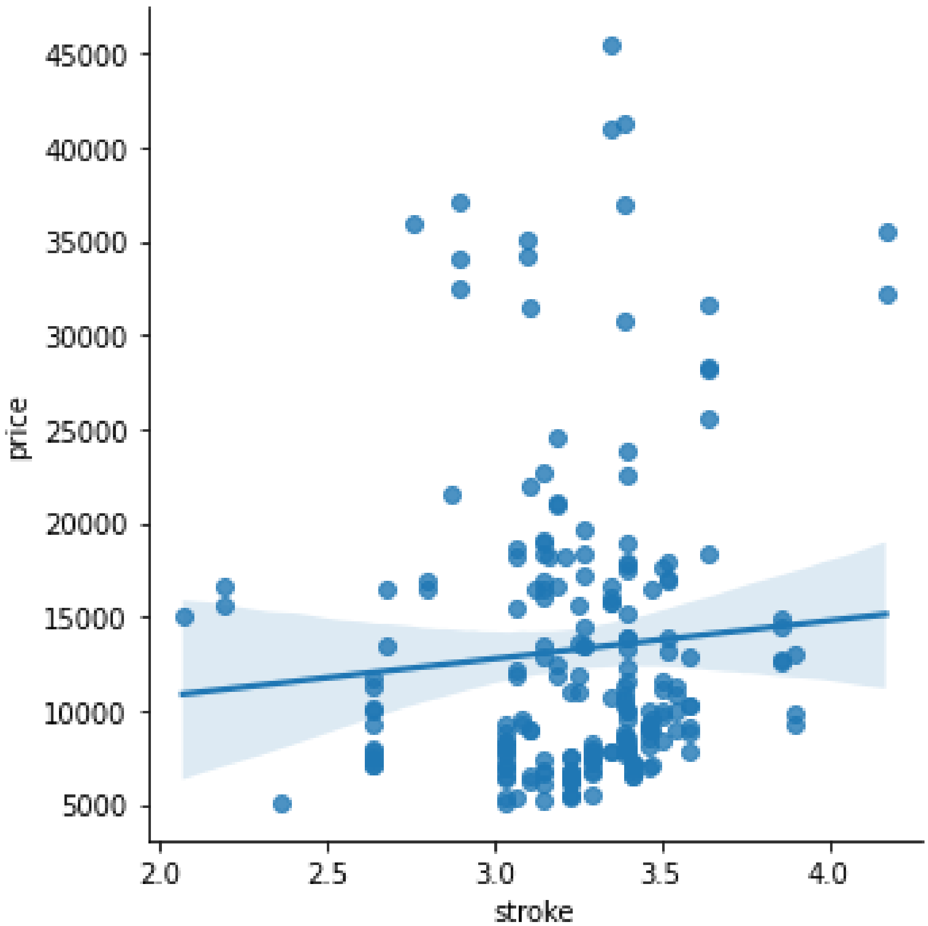
50000
40000
30000
20000
10000
0
price
50
100
150
200
250
300
enginesize

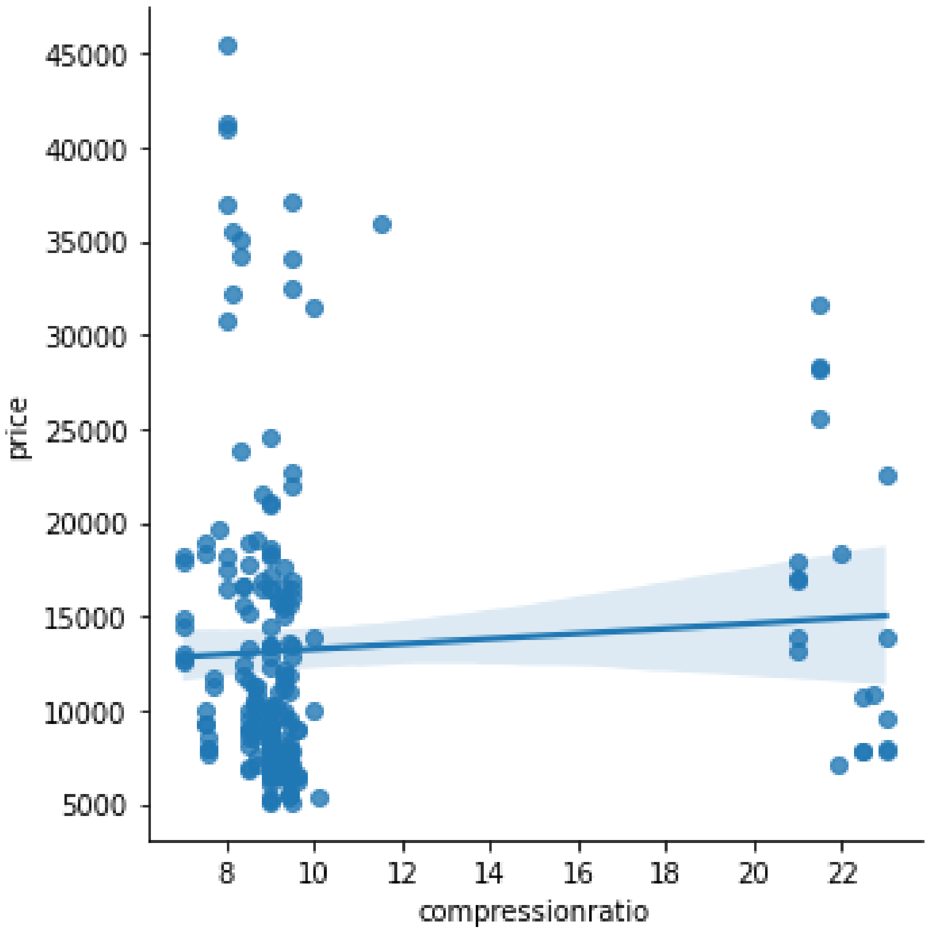


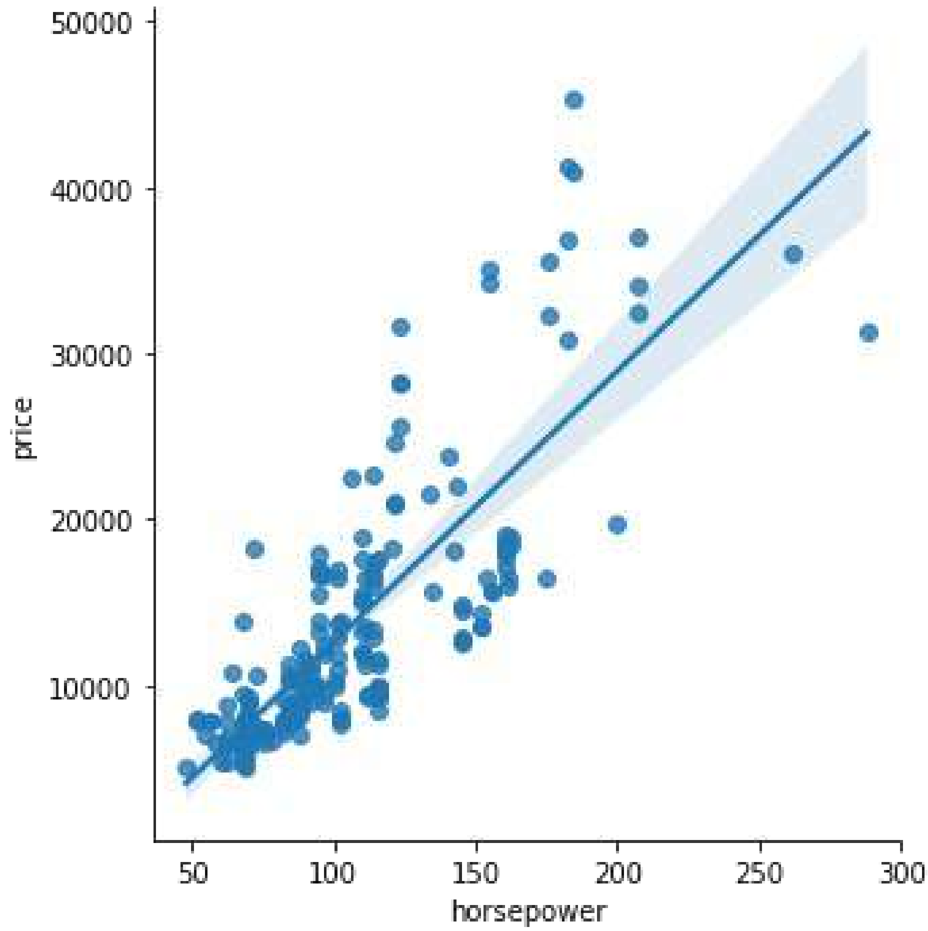
price
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
1
2
3
4
5
6
7
fuelsystem

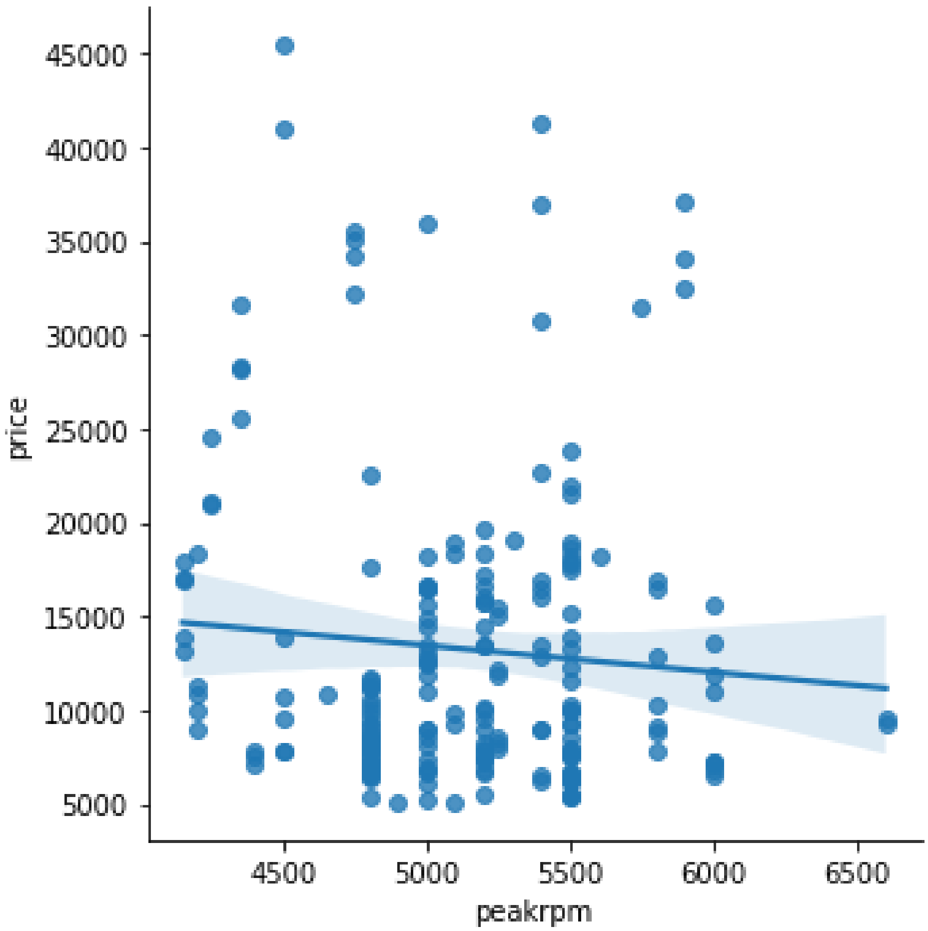


40000
30000
20000
10000
0
price
2.6
2.8
3.0
3.2
3.4
3.6
3.8
4.0
boreratio

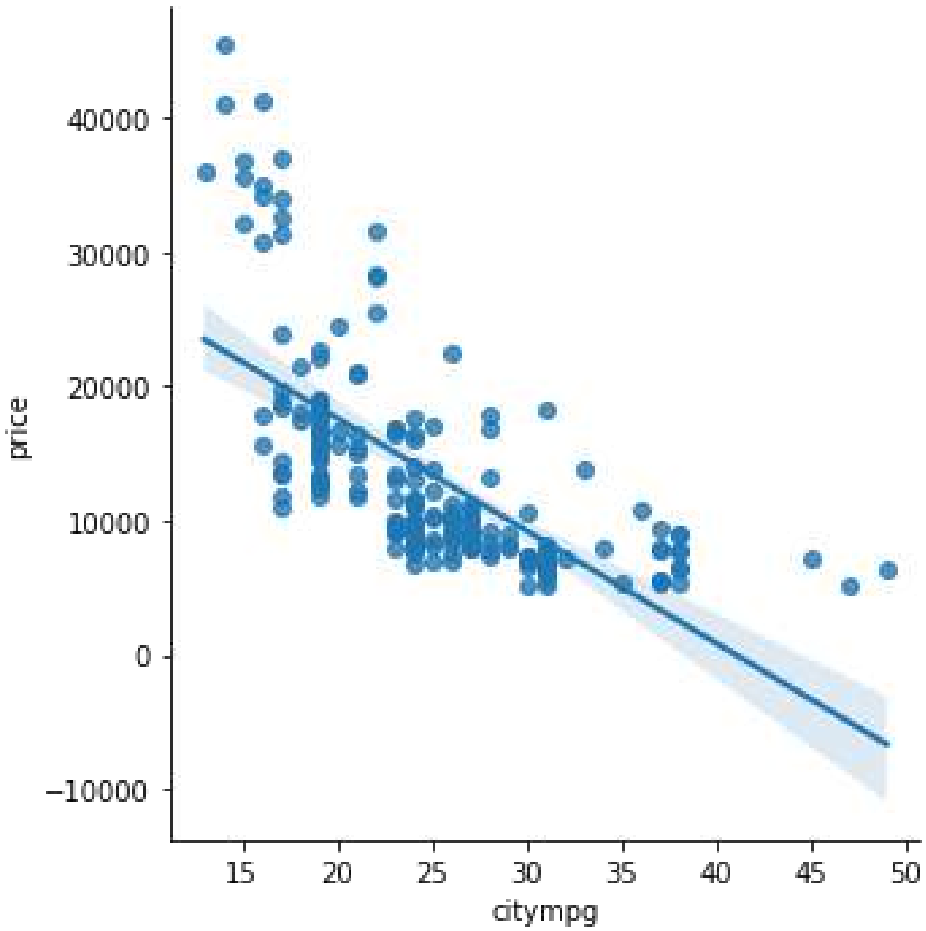

45000
40000
35000
30000
25000
20000
15000
10000
5000
price
2.0
2.5
3.0
3.5
4.0
stroke


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

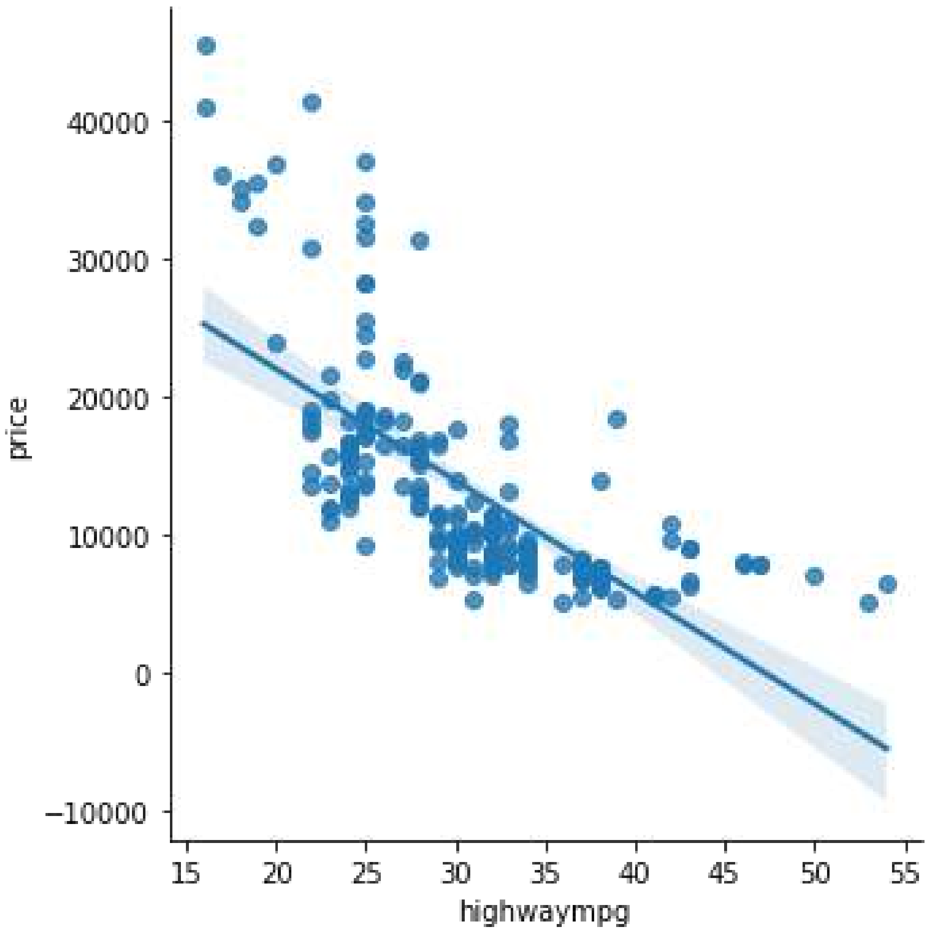

50000
40000
30000
20000
10000
price
50
100
150
200
250
300
horsepower

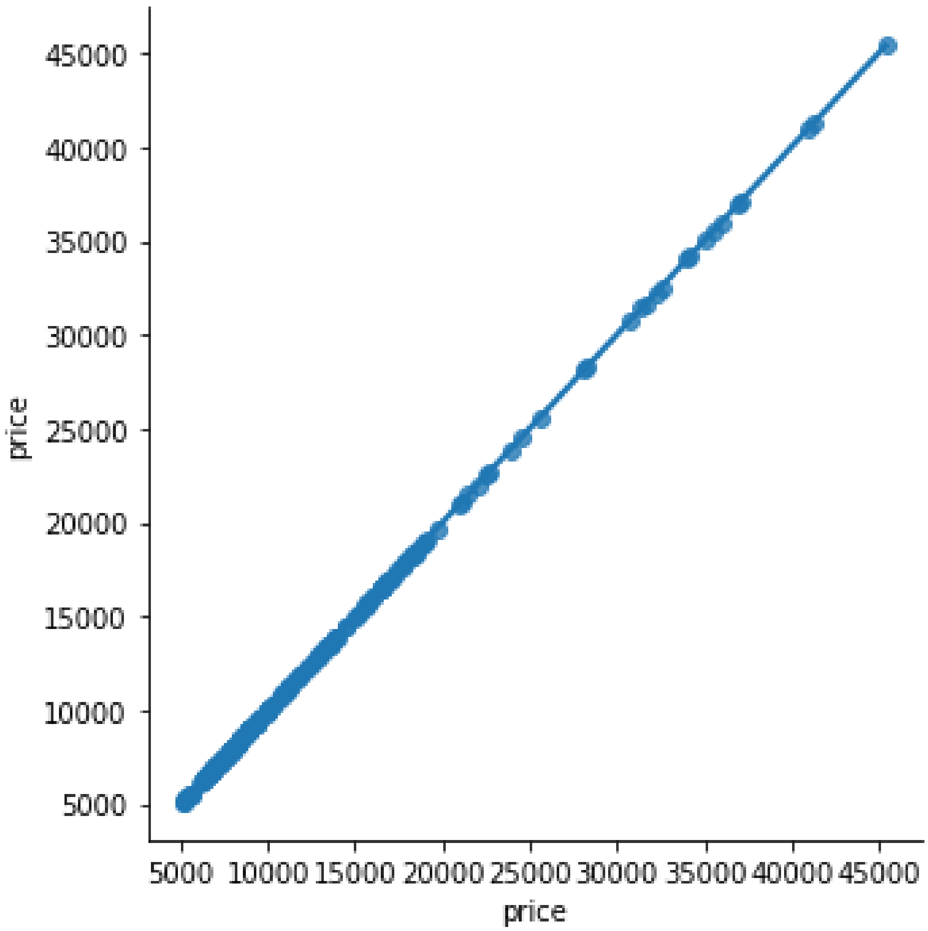


45000
40000
35000
30000
25000
20000
15000
10000
5000
price
4500
5000
5500
6000
6500
peakrpm



40000
30000
20000
10000
0
−10000
price
15
20
25
30
35
40
45
50
citympg


price
40000
30000
20000
10000
0
−10000
15
20
25
30
35
40
45
50
55
highwaympg




```
#To drop because uncorrelated to price:
to_drop = ['peakrpm', 'compressionratio', 'stroke', 'symboling']
df.drop(df[to_drop], axis = 1, inplace = True)
```

Seleting the Features and Target

```
df
df.to_csv('Cleaned_car_data.csv') # saving cleaned dataset for later
use

X = df.drop('price', axis=1)
y = df.price

X
```

```
     CarName  fueltype  aspiration  doornumber  carbody  drivewheel  \
0        0.0       1.0         0.0         1.0      0.0         2.0
1        0.0       1.0         0.0         1.0      0.0         2.0
2        0.0       1.0         0.0         1.0      2.0         2.0
3        1.0       1.0         0.0         0.0      3.0         1.0
```

```
4        1.0       1.0         0.0         0.0      3.0         0.0
..       ...       ...         ...         ...      ...         ...
200     21.0       1.0         0.0         0.0      3.0         2.0
201     21.0       1.0         1.0         0.0      3.0         2.0
202     21.0       1.0         0.0         0.0      3.0         2.0
203     21.0       0.0         1.0         0.0      3.0         2.0
204     21.0       1.0         1.0         0.0      3.0         2.0

     enginelocation  wheelbase  carlength  carwidth  carheight 
curbweight  \
0               0.0       88.6      168.8      64.1       48.8 
2548
1               0.0       88.6      168.8      64.1       48.8 
2548
2               0.0       94.5      171.2      65.5       52.4 
2823
3               0.0       99.8      176.6      66.2       54.3 
2337
4               0.0       99.4      176.6      66.4       54.3 
2824
..              ...        ...        ...       ...        ... 
...
200             0.0      109.1      188.8      68.9       55.5 
2952
201             0.0      109.1      188.8      68.8       55.5 
3049
202             0.0      109.1      188.8      68.9       55.5 
3012
203             0.0      109.1      188.8      68.9       55.5 
3217
204             0.0      109.1      188.8      68.9       55.5 
3062

     enginetype  cylindernumber  enginesize  fuelsystem  boreratio  \
0           0.0             2.0         130         5.0       3.47   
1           0.0             2.0         130         5.0       3.47   
2           5.0             3.0         152         5.0       2.68   
3           3.0             2.0         109         5.0       3.19   
4           3.0             1.0         136         5.0       3.19   
..          ...             ...         ...         ...        ...   
200         3.0             2.0         141         5.0       3.78   
201         3.0             2.0         141         5.0       3.78   
202         5.0             3.0         173         5.0       3.58   
203         3.0             3.0         145         3.0       3.01   
204         3.0             2.0         141         5.0       3.78   

     horsepower  citympg  highwaympg  
0           111       21          27  
1           111       21          27  
2           154       19          26  
```

```
3            102       24          30
4            115       18          22
..           ...      ...         ...
200          114       23          28
201          160       19          25
202          134       18          23
203          106       26          27
204          114       19          25

[205 rows x 20 columns]
```

```
df.columns
```

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

```
y
```

```
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
```

Feature Selection (Feature Importance)

```
model = ExtraTreesRegressor()
```

The **ExtraTreesRegressor** will going to check each features with this target variable and find out's the feature importance

~ To Know more about ExtraTeesRegressor refer thsi link : https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesRegressor.html#:~:text=An%20extra%2Dtrees%20regressor.,accuracy%20and%20control%20over%2Dfitting.

```
model.fit(X,y)
```

```
ExtraTreesRegressor()
```

```
model.feature_importances_
```

```
array([0.01677058, 0.00829443, 0.01486611, 0.00297802, 0.00713448,
       0.09081255, 0.02709801, 0.01907157, 0.0225882 , 0.05370716,
       0.00710102, 0.13519165, 0.00558668, 0.01161153, 0.24994807,
       0.01988773, 0.01239253, 0.12354184, 0.08600155, 0.08541631])
```

- Top 10 features w.r.t. the Target

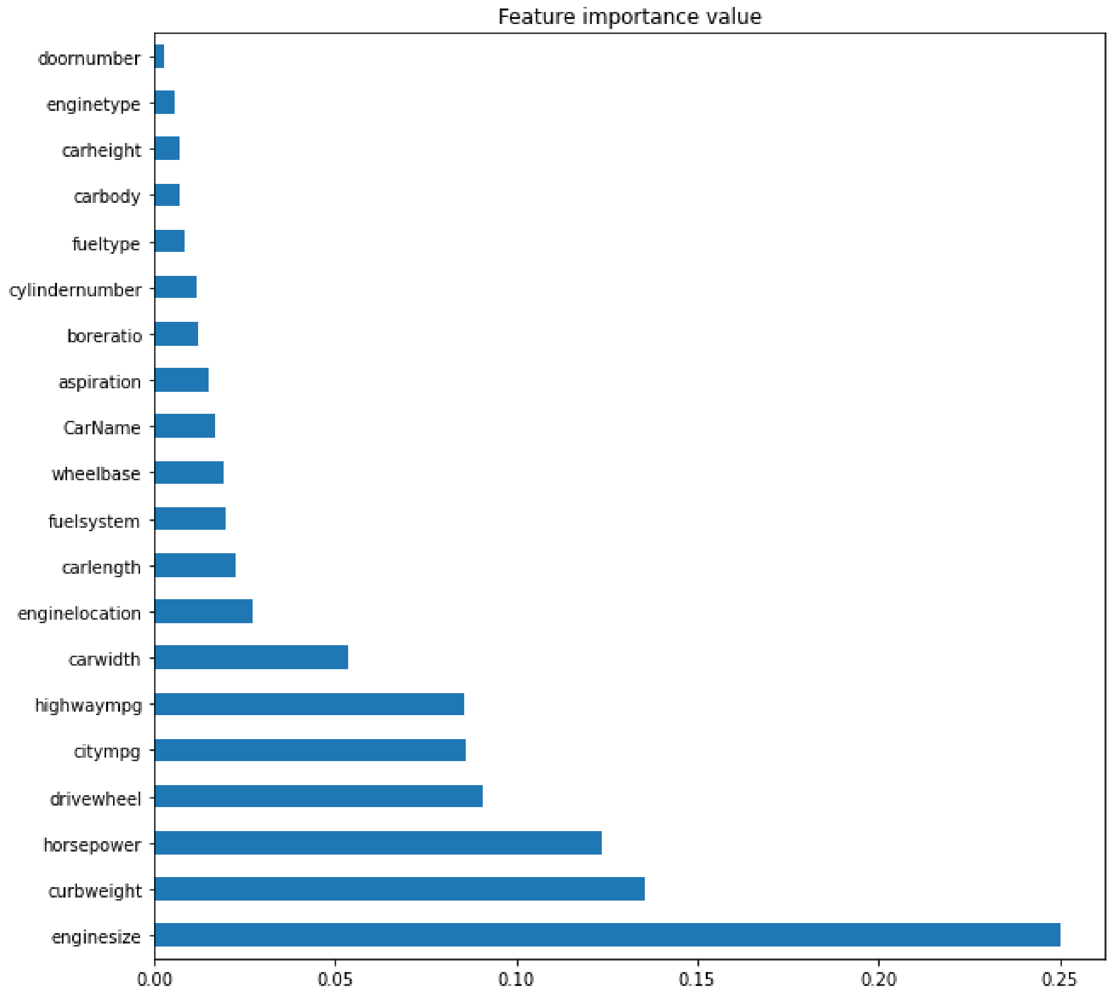
```
important_features = pd.Series(model.feature_importances_,
                               index = X.columns)
```

```
important_features
```

```
CarName           0.016771
fueltype          0.008294
aspiration        0.014866
doornumber        0.002978
carbody           0.007134
drivewheel        0.090813
enginelocation    0.027098
wheelbase         0.019072
carlength         0.022588
carwidth          0.053707
carheight         0.007101
curbweight        0.135192
enginetype        0.005587
cylindernumber    0.011612
enginesize        0.249948
fuelsystem        0.019888
boreratio         0.012393
horsepower        0.123542
citympg           0.086002
highwaympg        0.085416
dtype: float64
```

```
plt.figure(figsize=(10,10))
plt.title("Feature importance value")
important_features.nlargest(20).plot(kind='barh')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdfcbfe32d0>
```




```
important_features.nlargest(10)
```

```
enginesize        0.249948
curbweight        0.135192
horsepower        0.123542
drivewheel        0.090813
citympg           0.086002
highwaympg        0.085416
carwidth          0.053707
enginelocation    0.027098
carlength         0.022588
fuelsystem        0.019888
dtype: float64
```

```
important_features.nlargest(20).index
```

```
Index(['enginesize', 'curbweight', 'horsepower', 'drivewheel',
'citympg',
       'highwaympg', 'carwidth', 'enginelocation', 'carlength',
'fuelsystem',
       'wheelbase', 'CarName', 'aspiration', 'boreratio',
```

```
'cylindernumber',
       'fueltype', 'carbody', 'carheight', 'enginetype',
'doornumber'],
      dtype='object')
```

```
asd = important_features.nlargest(20).index
```

```
list(asd)
```

```
['enginesize',
 'curbweight',
 'horsepower',
 'drivewheel',
 'citympg',
 'highwaympg',
 'carwidth',
 'enginelocation',
 'carlength',
 'fuelsystem',
 'wheelbase',
 'CarName',
 'aspiration',
 'boreratio',
 'cylindernumber',
 'fueltype',
 'carbody',
 'carheight',
 'enginetype',
 'doornumber']
```

```
df[list(asd)]
```

```
     enginesize  curbweight  horsepower  drivewheel  citympg
highwaympg  \
0           130        2548         111         2.0       21
27
1           130        2548         111         2.0       21
27
2           152        2823         154         2.0       19
26
3           109        2337         102         1.0       24
30
4           136        2824         115         0.0       18
22
..          ...         ...         ...         ...      ...          .
..
200         141        2952         114         2.0       23
28
201         141        3049         160         2.0       19
25
202         173        3012         134         2.0       18
```

```
23
203          145        3217          106          2.0       26
27
204          141        3062          114          2.0       19
25

     carwidth  enginelocation  carlength  fuelsystem  wheelbase
CarName  \
0        64.1             0.0      168.8         5.0       88.6
0.0
1        64.1             0.0      168.8         5.0       88.6
0.0
2        65.5             0.0      171.2         5.0       94.5
0.0
3        66.2             0.0      176.6         5.0       99.8
1.0
4        66.4             0.0      176.6         5.0       99.4
1.0
..        ...             ...        ...         ...        ...      .
..
200      68.9             0.0      188.8         5.0      109.1
21.0
201      68.8             0.0      188.8         5.0      109.1
21.0
202      68.9             0.0      188.8         5.0      109.1
21.0
203      68.9             0.0      188.8         3.0      109.1
21.0
204      68.9             0.0      188.8         5.0      109.1
21.0

     aspiration  boreratio  cylindernumber  fueltype  carbody
carheight  \
0           0.0       3.47             2.0       1.0      0.0
48.8
1           0.0       3.47             2.0       1.0      0.0
48.8
2           0.0       2.68             3.0       1.0      2.0
52.4
3           0.0       3.19             2.0       1.0      3.0
54.3
4           0.0       3.19             1.0       1.0      3.0
54.3
..          ...        ...             ...       ...      ...        .
..
200         0.0       3.78             2.0       1.0      3.0
55.5
201         1.0       3.78             2.0       1.0      3.0
55.5
202         0.0       3.58             3.0       1.0      3.0
```

```
55.5
203          1.0       3.01              3.0       0.0      3.0
55.5
204          1.0       3.78              2.0       1.0      3.0
55.5

     enginetype  doornumber
0           0.0         1.0
1           0.0         1.0
2           5.0         1.0
3           3.0         0.0
4           3.0         0.0
..          ...         ...
200         3.0         0.0
201         3.0         0.0
202         5.0         0.0
203         3.0         0.0
204         3.0         0.0

[205 rows x 20 columns]
```

```
X_new = df[list(asd)]
```

```
len(X_new)
```

```
205
```

```
X_new.shape
```

```
(205, 20)
```

DATA SPLITTING

Spliting the dataset into Training & Testing sets

```
X_train, X_test, y_train, y_test = train_test_split(X_new, y,
test_size=0.20, random_state=62)
```

```
len(X_train)
```

```
164
```

```
len(X_test)
```

```
41
```

#**Training the model**

```
# These will be used later for Model Comparision
names, mses, rmses, r2s = [], [], [], []
```

```
from sklearn.metrics import mean_squared_error
```

LINEAR REGRESSION

```
# Linear Regression

from sklearn.linear_model import LinearRegression

lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
lr_pred = lr_model.predict(X_test)
#performance metrics
lr = r2_score(y_test, lr_pred)
lr_mse = mean_squared_error(y_test, lr_pred)
lr_rmse = np.sqrt(lr_mse)

print(f"r2_score : {lr}\n")

print(f"Mean Squared Error : {lr_mse}\n")

print(f"Root Mean Squared Error : {lr_rmse}\n")



# Model Comparision
names.append("Linear Regression")
mses.append(lr_mse)
rmses.append(lr_rmse)
r2s.append(lr)
```

```
r2_score : 0.8773629731348082

Mean Squared Error : 6767356.428731424

Root Mean Squared Error : 2601.4143131634037
```

DECISION TREE

```
# Decision Tree
from sklearn.tree import DecisionTreeRegressor

dt_model = DecisionTreeRegressor()
dt_model.fit(X_train, y_train)
```

```python
dt_pred = dt_model.predict(X_test)
dt_r2 = r2_score(y_test, dt_pred)
dt_mse = mean_squared_error(y_test, dt_pred)

dt_rmse = np.sqrt(dt_mse)

print(f"r2_score : {dt_r2}\n")

print(f"Mean Squared Error : {dt_mse}\n")

print(f"Root Mean Squared Error : {dt_rmse}\n")



# Model Comparision
names.append("Decision Tree Regression")
mses.append(dt_mse)
rmses.append(dt_rmse)
r2s.append(dt_r2)
```

```
r2_score : 0.9258650573153372

Mean Squared Error : 4090914.4146341463

Root Mean Squared Error : 2022.6009034493547
```

RANDOM FOREST REGRESSION

```python
model = RandomForestRegressor()
```

1. Performing the Hyper Parameter Tuning to get the best parametric value for the model using RandomizedSearchCV

~ To know more about RandomForest and Hyper Parametric Tunning refer this link : https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

```python
# No. of trees in a random forest
n_estimators = [int(i) for i in range(100, 1201, 100)] # - This is the parameter that defines how much decision tree we want to use

# The number of feautures to consider when looking for the best split
max_features = ['sqrt', 'auto']

# Maximun number of levels in a tree
max_depth = [i for i in np.linspace(start=5,stop=30,num=6)]

# Minimum number of samples required to split a node
```

```
min_samples_split = [2, 5, 10, 15, 100]

# Minimum number of samples required at each leaf node
min_samples_leaf = [1,2,5,10]

random_grid = {
    "n_estimators": n_estimators,
    "max_features": max_features,
    "max_depth": max_depth,
    "min_samples_split": min_samples_split,
    "min_samples_leaf": min_samples_leaf
}

random_grid
```

```
{'n_estimators': [100,
  200,
  300,
  400,
  500,
  600,
  700,
  800,
  900,
  1000,
  1100,
  1200],
 'max_features': ['sqrt', 'auto'],
 'max_depth': [5.0, 10.0, 15.0, 20.0, 25.0, 30.0],
 'min_samples_split': [2, 5, 10, 15, 100],
 'min_samples_leaf': [1, 2, 5, 10]}
```

RandomizedSearchCV is going to perform each possible outcome , i.e, it will try to find out the accuracy based on that data and it will tell you what are the suitable parameteric values that you can use to achieve the highest number of accuracy for this random forest regressor.

~ To know more about the RandomizedSearchCV refer this link : https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html

```
rf_random = RandomizedSearchCV(estimator = model,
                               param_distributions = random_grid,
                               scoring = 'neg_mean_squared_error',
                               n_jobs = 1,
                               random_state = 42,
                               verbose = 2,
                               cv = 5)

rf_random.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END max_depth=10.0, max_features=auto, min_samples_leaf=5,
min_samples_split=5, n_estimators=900; total time=   1.4s
[CV] END max_depth=10.0, max_features=auto, min_samples_leaf=5,
min_samples_split=5, n_estimators=900; total time=   1.4s
[CV] END max_depth=10.0, max_features=auto, min_samples_leaf=5,
min_samples_split=5, n_estimators=900; total time=   1.4s
[CV] END max_depth=10.0, max_features=auto, min_samples_leaf=5,
min_samples_split=5, n_estimators=900; total time=   1.4s
[CV] END max_depth=10.0, max_features=auto, min_samples_leaf=5,
min_samples_split=5, n_estimators=900; total time=   1.4s
[CV] END max_depth=15.0, max_features=auto, min_samples_leaf=2,
min_samples_split=10, n_estimators=1100; total time=   1.7s
[CV] END max_depth=15.0, max_features=auto, min_samples_leaf=2,
min_samples_split=10, n_estimators=1100; total time=   1.7s
[CV] END max_depth=15.0, max_features=auto, min_samples_leaf=2,
min_samples_split=10, n_estimators=1100; total time=   1.7s
[CV] END max_depth=15.0, max_features=auto, min_samples_leaf=2,
min_samples_split=10, n_estimators=1100; total time=   1.7s
[CV] END max_depth=15.0, max_features=auto, min_samples_leaf=2,
min_samples_split=10, n_estimators=1100; total time=   1.9s
[CV] END max_depth=15.0, max_features=sqrt, min_samples_leaf=5,
min_samples_split=100, n_estimators=300; total time=   0.6s
[CV] END max_depth=15.0, max_features=sqrt, min_samples_leaf=5,
min_samples_split=100, n_estimators=300; total time=   0.6s
[CV] END max_depth=15.0, max_features=sqrt, min_samples_leaf=5,
min_samples_split=100, n_estimators=300; total time=   0.6s
[CV] END max_depth=15.0, max_features=sqrt, min_samples_leaf=5,
min_samples_split=100, n_estimators=300; total time=   0.5s
[CV] END max_depth=15.0, max_features=sqrt, min_samples_leaf=5,
min_samples_split=100, n_estimators=300; total time=   0.4s
[CV] END max_depth=15.0, max_features=sqrt, min_samples_leaf=5,
min_samples_split=5, n_estimators=400; total time=   0.5s
[CV] END max_depth=15.0, max_features=sqrt, min_samples_leaf=5,
min_samples_split=5, n_estimators=400; total time=   0.5s
[CV] END max_depth=15.0, max_features=sqrt, min_samples_leaf=5,
min_samples_split=5, n_estimators=400; total time=   0.5s
[CV] END max_depth=15.0, max_features=sqrt, min_samples_leaf=5,
min_samples_split=5, n_estimators=400; total time=   0.5s
[CV] END max_depth=15.0, max_features=sqrt, min_samples_leaf=5,
min_samples_split=5, n_estimators=400; total time=   0.5s
[CV] END max_depth=20.0, max_features=sqrt, min_samples_leaf=10,
min_samples_split=5, n_estimators=700; total time=   0.9s
[CV] END max_depth=20.0, max_features=sqrt, min_samples_leaf=10,
min_samples_split=5, n_estimators=700; total time=   0.9s
[CV] END max_depth=20.0, max_features=sqrt, min_samples_leaf=10,
min_samples_split=5, n_estimators=700; total time=   0.9s
[CV] END max_depth=20.0, max_features=sqrt, min_samples_leaf=10,
min_samples_split=5, n_estimators=700; total time=   0.9s
[CV] END max_depth=20.0, max_features=sqrt, min_samples_leaf=10,
```

```
min_samples_split=5, n_estimators=700; total time=   0.9s
[CV] END max_depth=25.0, max_features=auto, min_samples_leaf=1,
min_samples_split=2, n_estimators=1000; total time=   1.9s
[CV] END max_depth=25.0, max_features=auto, min_samples_leaf=1,
min_samples_split=2, n_estimators=1000; total time=   1.9s
[CV] END max_depth=25.0, max_features=auto, min_samples_leaf=1,
min_samples_split=2, n_estimators=1000; total time=   1.9s
[CV] END max_depth=25.0, max_features=auto, min_samples_leaf=1,
min_samples_split=2, n_estimators=1000; total time=   1.8s
[CV] END max_depth=25.0, max_features=auto, min_samples_leaf=1,
min_samples_split=2, n_estimators=1000; total time=   1.8s
[CV] END max_depth=5.0, max_features=auto, min_samples_leaf=10,
min_samples_split=15, n_estimators=1100; total time=   1.5s
[CV] END max_depth=5.0, max_features=auto, min_samples_leaf=10,
min_samples_split=15, n_estimators=1100; total time=   1.6s
[CV] END max_depth=5.0, max_features=auto, min_samples_leaf=10,
min_samples_split=15, n_estimators=1100; total time=   1.5s
[CV] END max_depth=5.0, max_features=auto, min_samples_leaf=10,
min_samples_split=15, n_estimators=1100; total time=   1.5s
[CV] END max_depth=5.0, max_features=auto, min_samples_leaf=10,
min_samples_split=15, n_estimators=1100; total time=   1.5s
[CV] END max_depth=15.0, max_features=auto, min_samples_leaf=1,
min_samples_split=15, n_estimators=300; total time=   0.5s
[CV] END max_depth=15.0, max_features=auto, min_samples_leaf=1,
min_samples_split=15, n_estimators=300; total time=   0.4s
[CV] END max_depth=15.0, max_features=auto, min_samples_leaf=1,
min_samples_split=15, n_estimators=300; total time=   0.5s
[CV] END max_depth=15.0, max_features=auto, min_samples_leaf=1,
min_samples_split=15, n_estimators=300; total time=   0.4s
[CV] END max_depth=15.0, max_features=auto, min_samples_leaf=1,
min_samples_split=15, n_estimators=300; total time=   0.5s
[CV] END max_depth=5.0, max_features=auto, min_samples_leaf=2,
min_samples_split=10, n_estimators=700; total time=   1.1s
[CV] END max_depth=5.0, max_features=auto, min_samples_leaf=2,
min_samples_split=10, n_estimators=700; total time=   1.1s
[CV] END max_depth=5.0, max_features=auto, min_samples_leaf=2,
min_samples_split=10, n_estimators=700; total time=   1.1s
[CV] END max_depth=5.0, max_features=auto, min_samples_leaf=2,
min_samples_split=10, n_estimators=700; total time=   1.1s
[CV] END max_depth=5.0, max_features=auto, min_samples_leaf=2,
min_samples_split=10, n_estimators=700; total time=   1.1s
[CV] END max_depth=20.0, max_features=sqrt, min_samples_leaf=1,
min_samples_split=15, n_estimators=700; total time=   0.9s
[CV] END max_depth=20.0, max_features=sqrt, min_samples_leaf=1,
min_samples_split=15, n_estimators=700; total time=   0.9s
[CV] END max_depth=20.0, max_features=sqrt, min_samples_leaf=1,
min_samples_split=15, n_estimators=700; total time=   0.9s
[CV] END max_depth=20.0, max_features=sqrt, min_samples_leaf=1,
min_samples_split=15, n_estimators=700; total time=   0.9s
```

```
[CV] END max_depth=20.0, max_features=sqrt, min_samples_leaf=1,
min_samples_split=15, n_estimators=700; total time=   0.9s

RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(), n_jobs=1,
                   param_distributions={'max_depth': [5.0, 10.0, 15.0,
20.0,
                                                      25.0, 30.0],
                                        'max_features': ['sqrt',
'auto'],
                                        'min_samples_leaf': [1, 2, 5,
10],
                                        'min_samples_split': [2, 5,
10, 15,
                                                              100],
                                        'n_estimators': [100, 200,
300, 400,
                                                         500, 600,
700, 800,
                                                         900, 1000,
1100,
                                                         1200]},
                   random_state=42, scoring='neg_mean_squared_error',
                   verbose=2)
```

```
rf_random.best_params_ # this is the best parametric value
```

```
{'n_estimators': 1000,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth': 25.0}
```

- Testing Phase

```
y_pred = rf_random.predict(X_test)

final_df = pd.DataFrame({"Actual": y_test,
                         "Predicted": y_pred})

final_df
```

	Actual	Predicted
139	7053.0	7538.862833
66	18344.0	11850.921833
77	6189.0	6454.580500
167	8449.0	10399.293500
5	15250.0	14674.709938
25	6692.0	6978.290000
47	32250.0	37970.251000
37	7895.0	8323.105500
182	7775.0	7988.945500
60	8495.0	10261.061333

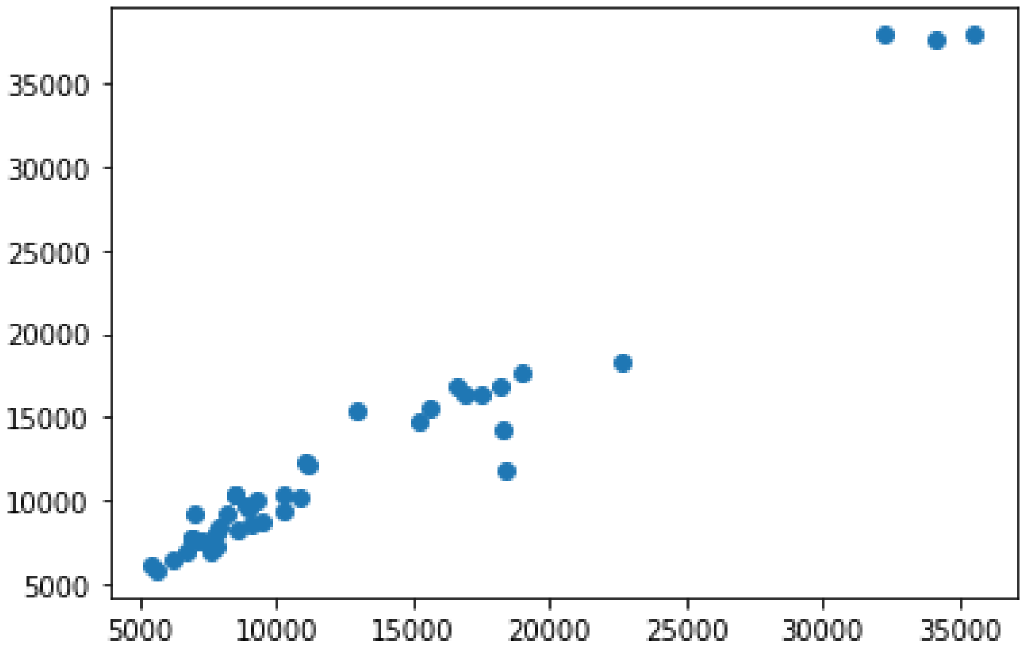
```
111  15580.0  15495.020167
170  11199.0  12050.256250
36    7295.0   7581.873000
123   8921.0   9747.965333
48   35550.0  37970.251000
176  10898.0  10126.736167
86    8189.0   9206.023667
204  22625.0  18344.655000
62   10245.0  10261.061333
40   10295.0   9319.768833
71   34184.0  37635.774000
4    17450.0  16368.447043
38    9095.0   8486.957167
112  16900.0  16325.002000
46   11048.0  12210.689238
32    5399.0   6075.882000
153   6918.0   7785.357000
187   9495.0   8697.014500
26    7609.0   6978.290000
85    6989.0   9141.971000
65   18280.0  14295.411083
118   5572.0   5779.295000
156   6938.0   7663.006500
178  16558.0  16870.698839
95    7799.0   7274.800000
199  18950.0  17605.935171
87    9279.0   9924.118833
99    8949.0   9542.954500
27    8558.0   8139.889000
194  12940.0  15339.058000
117  18150.0  16883.943670
```

```
final_df.corr()
```

```
             Actual  Predicted
Actual     1.000000   0.968626
Predicted  0.968626   1.000000
```

```
plt.scatter(y_test, y_pred)
```

```
<matplotlib.collections.PathCollection at 0x7fdfcb250210>
```



Random Forest - Performance Metrics

- USING HyperParameter Tuning

```
# With Hyperparameter tuning on 5 parameters
rf = r2_score(y_test, y_pred)
rft_mse = mean_squared_error(y_test, y_pred)
rf_rmse = np.sqrt(rft_mse)

print(f"r2_score : {rf}\n")
print(f"Mean Squared Error : {rft_mse}\n")
print(f"Root Mean Squared Error : {rf_rmse}\n")


# Model Comparision
names.append("RandomForest with HPT")
mses.append(rft_mse)
rmses.append(rf_rmse)
r2s.append(rf)
```

```
r2_score : 0.929300983350146

Mean Squared Error : 3901313.1438379395

Root Mean Squared Error : 1975.1742059468932
```

- USING without HyperParameter Tuning

```python
# Without Hyperparameter Tuning

model.fit(X_train, y_train)
rf_pred = model.predict(X_test)
rf_wht = r2_score(y_test, rf_pred)
rf_mse = mean_squared_error(y_test, rf_pred)
rfwt_rmse = np.sqrt(rf_mse)

print(f"r2_score : {rf_wht}\n")

print(f"Mean Squared Error : {rf_mse}\n")

print(f"Root Mean Squared Error : {rfwt_rmse}\n")


# Model Comparision
names.append("RandomForest without HPT")
mses.append(rf_mse)
rmses.append(rfwt_rmse)
r2s.append(rf_wht)
```

```
r2_score : 0.9262935859371868

Mean Squared Error : 4067267.348180411

Root Mean Squared Error : 2016.7467238551328
```

COMPARINING THE MODELS

```python
result_r2 = pd.DataFrame([rf,rf_wht, dt_r2, lr],['random forest with hyperparametric tuning','random forest without hyperparametric tuning', 'decision tree', 'linear regression'])

result_r2
```

	0
random forest with hyperparametric tuning	0.929301
random forest without hyperparametric tuning	0.926294
decision tree	0.925865
linear regression	0.877363

r2_score plot

```python
import plotly.express as px
fig = px.bar(x=names, y=r2s, color=names)
fig.update_layout({'title':{'text':"$R^2$ Score",'x':0.5}})
fig.show()
```

```
# MSE Plot
fig = px.bar(x=names, y=mses, color=names)
fig.update_layout({'title':{'text':"Mean Squared Error",'x':0.5}})
fig.show()

# RMSE Plot
fig = px.bar(x=names, y=rmses, color=names)
fig.update_layout({'title':{'text':"Root Mean Squared
Error",'x':0.5}})
fig.show()
```

Creating and Saving pickle file

```
import pickle
# open a file, where you ant to store the data
filename = 'LinearRegression.sav'

# dump information to that file
pickle.dump(lr_model, open(filename,'wb'))
loaded_model  = pickle.load(open('LinearRegression.sav','rb'))
```

All the process are done and we getting better accuracy

~ This project is done by a Team : ML Bots

ML BOTS MEMBERS		
S/NO	NAME	REGISTER NUMBER
1	R.SANJAY	40111136
2	S.SANJAY	40111137
3	P.VIKRAM	40111434