car-price-prediction-1

September 3, 2023

[417]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
sns.set()
%matplotlib inline

car=pd.read_csv(r"C:\Users\shilp\Downloads\archive\CarPrice_Assignment.csv")

car.shape
import warnings
warnings.filterwarnings('ignore')

car.head()
car.shape
```

[417]:
```
(205, 26)
```

[418]:
```
car.describe()
```

[418]:

	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
```

[419]:
```
car.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
```

[420]:
```
#Checking for null values in data

car.isnull().sum()
```

[420]:
```
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
```

[421]:
```
car.columns
#Drop Car Id as it is not significant

car.drop('car_ID',axis=1,inplace=True)
```

[422]:
```
#Separate Car Name from model

CompanyName = car['CarName'].apply(lambda x : x.split(' ')[0])
car.insert(3,"CompanyName",CompanyName)
car.drop(['CarName'],axis=1,inplace=True)
car.head()
```

[422]:
```
   symboling fueltype  CompanyName aspiration doornumber      carbody  \
0          3      gas  alfa-romero        std        two  convertible
1          3      gas  alfa-romero        std        two  convertible
2          1      gas  alfa-romero        std        two    hatchback
3          2      gas         audi        std       four        sedan
4          2      gas         audi        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]
```

[423]:
```
car['CompanyName'].value_counts()
```

[423]:
```
toyota         31
nissan         17
mazda          15
honda          13
mitsubishi     13
subaru         12
peugeot        11
volvo          11
volkswagen      9
dodge           9
buick           8
bmw             8
audi            7
plymouth        7
saab            6
```

```
isuzu           4
porsche         4
alfa-romero     3
chevrolet       3
jaguar          3
vw              2
maxda           2
renault         2
toyouta         1
vokswagen       1
Nissan          1
mercury         1
porcshce        1
Name: CompanyName, dtype: int64
```

[424]:
```
#Drop the duplicate values
car.drop_duplicates(inplace=True)
car.count() # No Duplicates
car.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 205 entries, 0 to 204
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   symboling         205 non-null    int64
 1   fueltype          205 non-null    object
 2   CompanyName       205 non-null    object
 3   aspiration        205 non-null    object
 4   doornumber        205 non-null    object
 5   carbody           205 non-null    object
 6   drivewheel        205 non-null    object
 7   enginelocation    205 non-null    object
 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    object
 14  cylindernumber    205 non-null    object
 15  enginesize        205 non-null    int64
 16  fuelsystem        205 non-null    object
 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(8), int64(7), object(10)
memory usage: 41.6+ KB
```

[425]:
```
car.columns
```

[425]:
```
Index(['symboling', 'fueltype', 'CompanyName', 'aspiration', 'doornumber',
       'carbody', 'drivewheel', 'enginelocation', 'wheelbase', 'carlength',
       'carwidth', 'carheight', 'curbweight', 'enginetype', 'cylindernumber',
       'enginesize', 'fuelsystem', 'boreratio', 'stroke', 'compressionratio',
       'horsepower', 'peakrpm', 'citympg', 'highwaympg', 'price'],
      dtype='object')
```

[426]:
```
#Use ANOVA Testing to check significance of variable


import statsmodels.api as sm
from statsmodels.formula.api import ols
mymod_id = ols('price ~ symboling', data = car).fit()
# performing type 2 anova test
aovtable = sm.stats.anova_lm(mymod_id, typ = 1)
print(aovtable)  #PR=6.414987e-16(<0.05) a significant variable cannot be␣
 ↪dropped

import statsmodels.api as sm
from statsmodels.formula.api import ols
mymod_id = ols('price ~ CompanyName', data = car).fit()
# performing type 2 anova test
aovtable = sm.stats.anova_lm(mymod_id, typ = 1)
print(aovtable)  #PR=6.414987e-16(<0.05) a significant variable cannot be␣
 ↪dropped

import statsmodels.api as sm
from statsmodels.formula.api import ols
mymod_id = ols('price ~ wheelbase', data = car).fit()
# performing type 2 anova test
aovtable = sm.stats.anova_lm(mymod_id, typ = 1)
print(aovtable)  #PR=6.414987e-16(<0.05) a significant variable cannot be␣
 ↪dropped

import statsmodels.api as sm
from statsmodels.formula.api import ols
mymod_id = ols('price ~ carlength', data = car).fit()
# performing type 2 anova test
aovtable = sm.stats.anova_lm(mymod_id, typ = 1)
```

```
print(aovtable)  #PR=6.414987e-16(<0.05) a significant variable cannot be␣
 ↪dropped

import statsmodels.api as sm
from statsmodels.formula.api import ols
mymod_id = ols('price ~ carwidth', data = car).fit()
# performing type 2 anova test
aovtable = sm.stats.anova_lm(mymod_id, typ = 1)
print(aovtable)  #PR=6.414987e-16(<0.05) a significant variable cannot be␣
 ↪dropped

import statsmodels.api as sm
from statsmodels.formula.api import ols
mymod_id = ols('price ~ carheight', data = car).fit()
# performing type 2 anova test
aovtable = sm.stats.anova_lm(mymod_id, typ = 1)
print(aovtable)  #PR=6.414987e-16(<0.05) a significant variable cannot be␣
 ↪dropped


import statsmodels.api as sm
from statsmodels.formula.api import ols
mymod_id = ols('price ~ curbweight', data = car).fit()
# performing type 2 anova test
aovtable = sm.stats.anova_lm(mymod_id, typ = 1)
print(aovtable)  #PR=6.414987e-16(<0.05) a significant variable cannot be␣
 ↪dropped



mymod_id = ols('price ~ fueltype', data = car).fit()
# performing type 2 anova test
aovtable = sm.stats.anova_lm(mymod_id, typ = 1)
print(aovtable)  #PR=0.131536(>0.05) not a significant variable can be dropped



mymod_id = ols('price ~ aspiration', data = car).fit()
# performing type 2 anova test
aovtable = sm.stats.anova_lm(mymod_id, typ = 1)
print(aovtable)  #PR=0.0.0107(<0.05) a significant variable cannot be dropped


mymod_id = ols('price ~ doornumber', data = car).fit()
# performing type 2 anova test
aovtable = sm.stats.anova_lm(mymod_id, typ = 1)
print(aovtable)  #PR=0.650448(>0.05) not a significant variable can be dropped
```

```
mymod_id = ols('price ~ carbody', data = car).fit()
# performing type 2 anova test
aovtable = sm.stats.anova_lm(mymod_id, typ = 1)
print(aovtable)  #PR=0.000005(>0.05) a significant variable cannot be dropped



mymod_id = ols('price ~ drivewheel', data = car).fit()
# performing type 2 anova test
aovtable = sm.stats.anova_lm(mymod_id, typ = 1)
print(aovtable)  #PR=6.632887e-24(>0.05) not a significant variable can be␣
 ↪dropped


mymod_id = ols('price ~ enginelocation', data = car).fit()
# performing type 2 anova test
aovtable = sm.stats.anova_lm(mymod_id, typ = 1)
print(aovtable) #PR=0.000002(<0.05) a significant variable cannot be dropped


mymod_id = ols('price ~ enginetype', data = car).fit()
# performing type 2 anova test
aovtable = sm.stats.anova_lm(mymod_id, typ = 1)
print(aovtable) #PR=4.692665e-09(>0.05) not a significant variable can be␣
 ↪dropped


mymod_id = ols('price ~ cylindernumber', data = car).fit()
# performing type 2 anova test
aovtable = sm.stats.anova_lm(mymod_id, typ = 1)
print(aovtable) #PR=8.065780e-41(>0.05) not a significant variable can be␣
 ↪dropped


mymod_id = ols('price ~ fuelsystem', data = car).fit()
# performing type 2 anova test
aovtable = sm.stats.anova_lm(mymod_id, typ = 1)
print(aovtable) #PR=2.990386e-16(>0.05) not a significant variable can be␣
 ↪dropped

mymod_id = ols('price ~ boreratio', data = car).fit()
# performing type 2 anova test
aovtable = sm.stats.anova_lm(mymod_id, typ = 1)
print(aovtable) #PR=2.990386e-16(>0.05) not a significant variable can be␣
 ↪dropped
```

```
mymod_id = ols('price ~ stroke', data = car).fit()
# performing type 2 anova test
aovtable = sm.stats.anova_lm(mymod_id, typ = 1)
print(aovtable) #PR=2.990386e-16(>0.05) not a significant variable can be␣
 ↪dropped


mymod_id = ols('price ~ compressionratio', data = car).fit()
# performing type 2 anova test
aovtable = sm.stats.anova_lm(mymod_id, typ = 1)
print(aovtable) #PR=2.990386e-16(>0.05) not a significant variable can be␣
 ↪dropped

mymod_id = ols('price ~ horsepower', data = car).fit()
# performing type 2 anova test
aovtable = sm.stats.anova_lm(mymod_id, typ = 1)
print(aovtable) #PR=2.990386e-16(>0.05) not a significant variable can be␣
 ↪dropped

mymod_id = ols('price ~ peakrpm', data = car).fit()
# performing type 2 anova test
aovtable = sm.stats.anova_lm(mymod_id, typ = 1)
print(aovtable) #PR=2.990386e-16(>0.05) not a significant variable can be␣
 ↪dropped

mymod_id = ols('price ~ citympg', data = car).fit()
# performing type 2 anova test
aovtable = sm.stats.anova_lm(mymod_id, typ = 1)
print(aovtable) #PR=2.990386e-16(>0.05) not a significant variable can be␣
 ↪dropped

mymod_id = ols('price ~ highwaympg', data = car).fit()
# performing type 2 anova test
aovtable = sm.stats.anova_lm(mymod_id, typ = 1)
print(aovtable) #PR=2.990386e-16(>0.05) not a significant variable can be␣
 ↪dropped
```

```
              df        sum_sq       mean_sq          F    PR(>F)
symboling    1.0  8.328034e+07  8.328034e+07   1.306852  0.254312
Residual   203.0  1.293636e+10  6.372591e+07        NaN       NaN
                df        sum_sq       mean_sq          F        PR(>F)
CompanyName   27.0  1.056711e+10  3.913744e+08  28.245639  1.655369e-50
Residual     177.0  2.452530e+09  1.385610e+07        NaN           NaN
              df        sum_sq       mean_sq           F        PR(>F)
wheelbase    1.0  4.346878e+09  4.346878e+09  101.745716  1.182820e-19
Residual   203.0  8.672761e+09  4.272296e+07         NaN           NaN
```

```
                df         sum_sq       mean_sq           F        PR(>F)
carlength      1.0   6.072096e+09  6.072096e+09  177.420344  1.678707e-29
Residual     203.0   6.947543e+09  3.422435e+07         NaN           NaN
               df         sum_sq       mean_sq           F        PR(>F)
carwidth      1.0   7.506797e+09  7.506797e+09  276.423646  9.627438e-40
Residual    203.0   5.512842e+09  2.715686e+07         NaN           NaN
               df         sum_sq       mean_sq         F    PR(>F)
carheight     1.0   1.854144e+08  1.854144e+08  2.932716  0.088328
Residual    203.0   1.283422e+10  6.322278e+07       NaN       NaN
                df         sum_sq       mean_sq           F        PR(>F)
curbweight     1.0   9.084248e+09  9.084248e+09  468.594431  1.214445e-54
Residual     203.0   3.935391e+09  1.938616e+07         NaN           NaN
               df         sum_sq       mean_sq         F    PR(>F)
fueltype      1.0   1.454053e+08  1.454053e+08  2.292741  0.131536
Residual    203.0   1.287423e+10  6.341987e+07       NaN       NaN
                df         sum_sq       mean_sq         F  PR(>F)
aspiration     1.0   4.121724e+08  4.121724e+08  6.636622  0.0107
Residual     203.0   1.260747e+10  6.210575e+07       NaN     NaN
                df         sum_sq       mean_sq         F    PR(>F)
doornumber     1.0   1.319520e+07  1.319520e+07  0.205946  0.650448
Residual     203.0   1.300644e+10  6.407115e+07       NaN       NaN
               df         sum_sq       mean_sq         F    PR(>F)
carbody       4.0   1.801997e+09  4.504992e+08  8.031976  0.000005
Residual    200.0   1.121764e+10  5.608821e+07       NaN       NaN
                df         sum_sq       mean_sq           F        PR(>F)
drivewheel     2.0   5.344065e+09  2.672033e+09  70.320553  6.632887e-24
Residual     202.0   7.675574e+09  3.799789e+07        NaN           NaN
                   df         sum_sq       mean_sq         F    PR(>F)
enginelocation    1.0   1.374973e+09  1.374973e+09  23.96974  0.000002
Residual        203.0   1.164467e+10  5.736289e+07       NaN       NaN
               df         sum_sq       mean_sq        F        PR(>F)
enginetype     6.0   2.880743e+09  4.801239e+08  9.37622  4.692665e-09
Residual     198.0   1.013890e+10  5.120655e+07      NaN           NaN
                   df         sum_sq       mean_sq          F        PR(>F)
cylindernumber    6.0   8.275757e+09  1.379293e+09  57.568881  8.065780e-41
Residual        198.0   4.743882e+09  2.395900e+07        NaN           NaN
               df         sum_sq       mean_sq           F        PR(>F)
fuelsystem     7.0   4.651199e+09  6.644569e+08  15.641865  2.990386e-16
Residual     197.0   8.368441e+09  4.247939e+07        NaN           NaN
               df         sum_sq       mean_sq         F        PR(>F)
boreratio     1.0   3.984018e+09  3.984018e+09  89.50747  7.907922e-18
Residual    203.0   9.035622e+09  4.451045e+07       NaN           NaN
               df         sum_sq       mean_sq         F    PR(>F)
stroke        1.0   8.216959e+07  8.216959e+07  1.289311  0.257514
Residual    203.0   1.293747e+10  6.373138e+07       NaN       NaN
                     df         sum_sq       mean_sq         F    PR(>F)
compressionratio    1.0   6.017361e+07  6.017361e+07  0.942573  0.332772
Residual          203.0   1.295947e+10  6.383973e+07       NaN       NaN
```

```
                df        sum_sq       mean_sq           F        PR(>F)
horsepower     1.0  8.502975e+09  8.502975e+09  382.163409  1.483437e-48
Residual     203.0  4.516664e+09  2.224958e+07         NaN           NaN
                df        sum_sq       mean_sq         F    PR(>F)
peakrpm        1.0  9.465912e+07  9.465912e+07  1.486718  0.224141
Residual     203.0  1.292498e+10  6.366985e+07       NaN       NaN
                df        sum_sq       mean_sq           F        PR(>F)
citympg        1.0  6.122549e+09  6.122549e+09  180.203163  7.978684e-30
Residual     203.0  6.897090e+09  3.397581e+07         NaN           NaN
                df        sum_sq       mean_sq           F        PR(>F)
highwaympg     1.0  6.335936e+09  6.335936e+09  192.437464  3.230681e-31
Residual     203.0  6.683704e+09  3.292465e+07         NaN           NaN
```

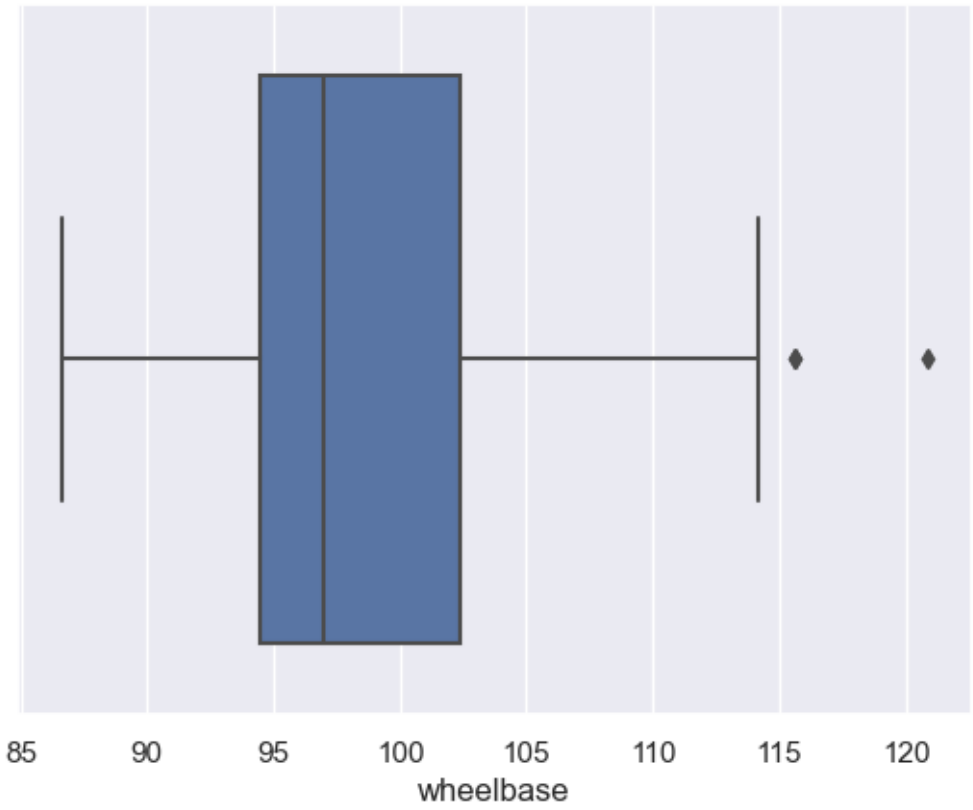
[427]:
```
#Drop insignificant variables

#Symboling, carheight, fueltype, doornumber, stroke, compressionratio based on␣
 ↪1-way Anova Testing

car.drop(['symboling','carheight', 'fueltype','doornumber','stroke',␣
 ↪'compressionratio'],axis=1,inplace=True)
```

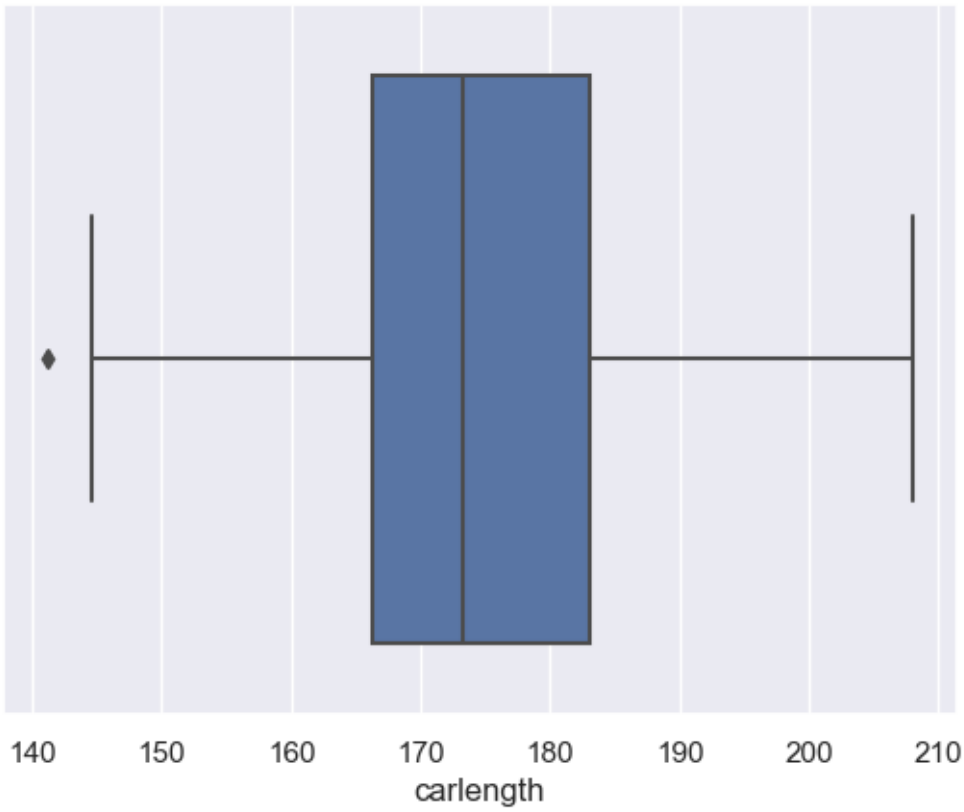
[428]:
```
#Check for remaining data

car.head()
```

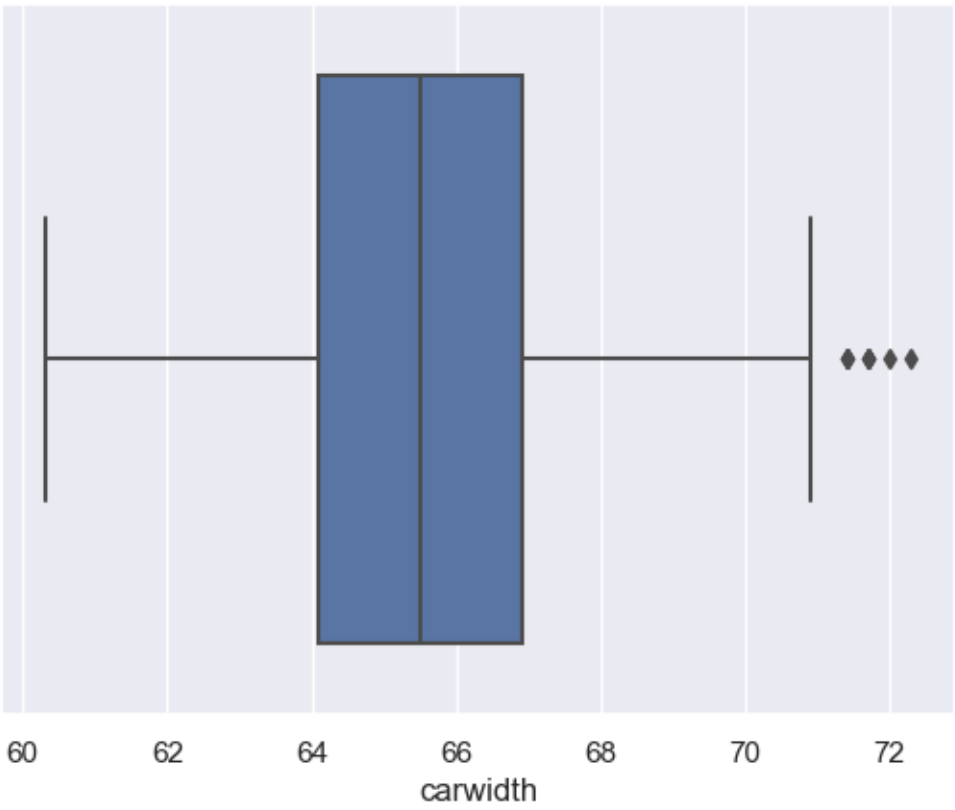
[428]:

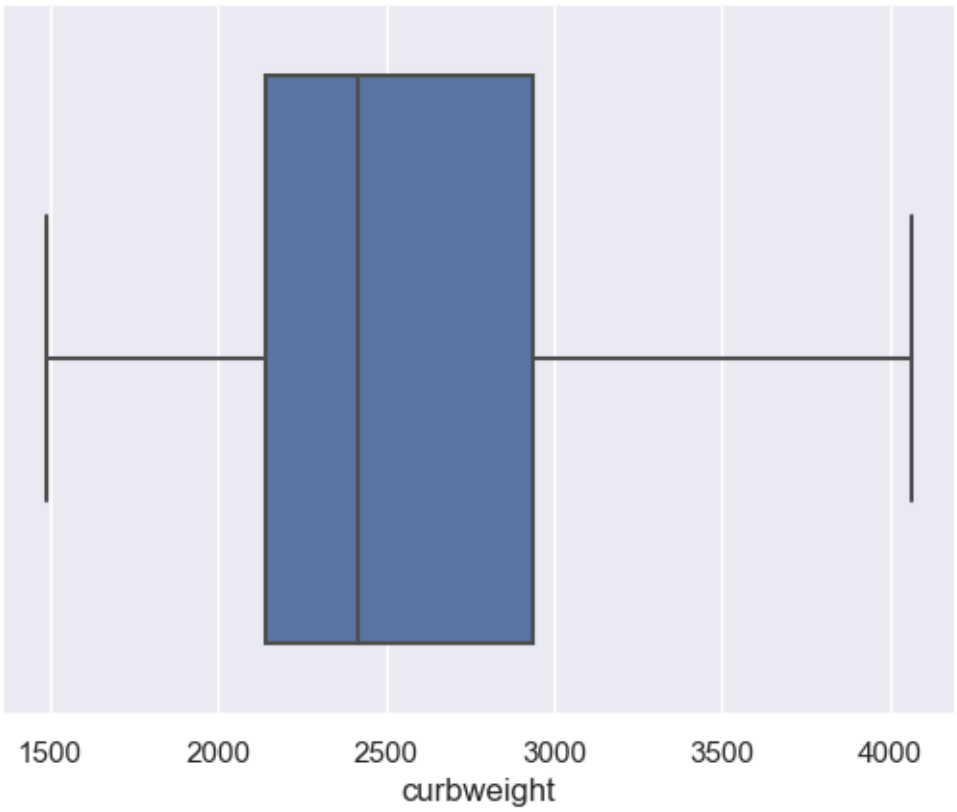
	CompanyName	aspiration	carbody	drivewheel	enginelocation	wheelbase	carlength	carwidth	curbweight	enginetype	cylindernumber	enginesize	fuelsystem	boreratio	horsepower	peakrpm	citympg	highwaympg	price
0	alfa-romero	std	convertible	rwd	front	88.6	168.8	64.1	2548	dohc	four	130	mpfi	3.47	111	5000	21	27	13495.0
1	alfa-romero	std	convertible	rwd	front	88.6	168.8	64.1	2548	dohc	four	130	mpfi	3.47	111	5000	21	27	16500.0
2	alfa-romero	std	hatchback	rwd	front	94.5	171.2	65.5	2823	ohcv	six	152	mpfi	2.68	154	5000	19	26	16500.0
3	audi	std	sedan	fwd	front	99.8	176.6	66.2	2337	ohc	four	109	mpfi	3.19	102	5500	24	30	13950.0
4	audi	std	sedan	4wd	front	99.4	176.6	66.4	2824	ohc	five	136	mpfi	3.19	115	5500	18	22	17450.0

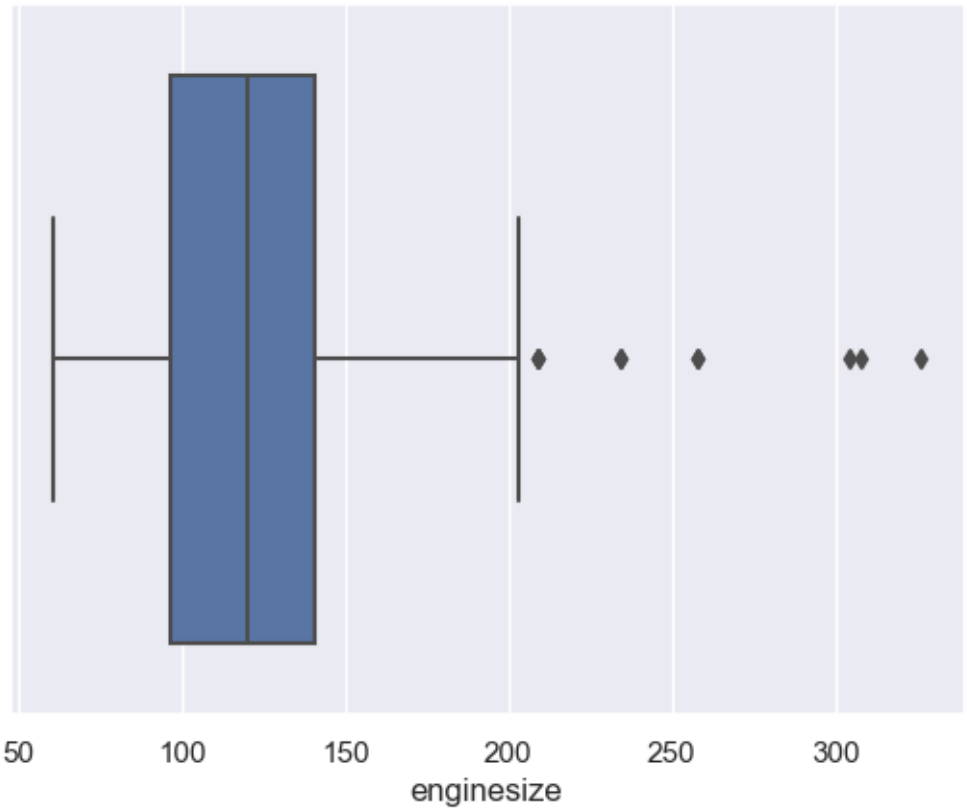
[429]:
```
#Check for outliers

car.describe()

sns.boxplot(x=car['wheelbase'])
plt.show()
sns.boxplot(x=car['carlength'])
plt.show()

sns.boxplot(x=car['carwidth'])
plt.show()

sns.boxplot(x=car['curbweight'])
plt.show()
sns.boxplot(x=car['enginesize'])
plt.show()

sns.boxplot(x=car['boreratio'])
plt.show()

sns.boxplot(x=car['peakrpm'])
plt.show()
sns.boxplot(x=car['citympg'])
plt.show()
sns.boxplot(x=car['highwaympg'])
plt.show()

#Check for outliers

car.describe()

sns.boxplot(x=car['boreratio'])
plt.show()
sns.boxplot(x=car['horsepower'])
plt.show()

sns.boxplot(x=car['peakrpm'])
plt.show()
sns.boxplot(x=car['citympg'])
plt.show()
sns.boxplot(x=car['highwaympg'])
plt.show()

sns.boxplot(x=car['price'])
plt.show()
```

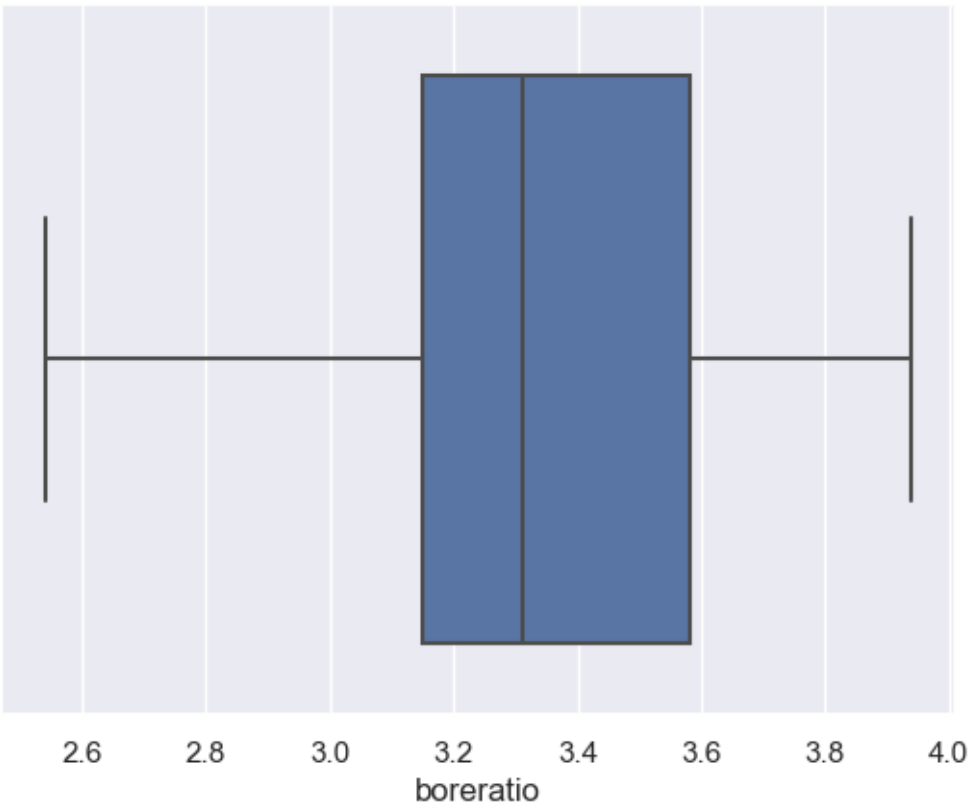

85
90
95
100
105
110
115
120
wheelbase

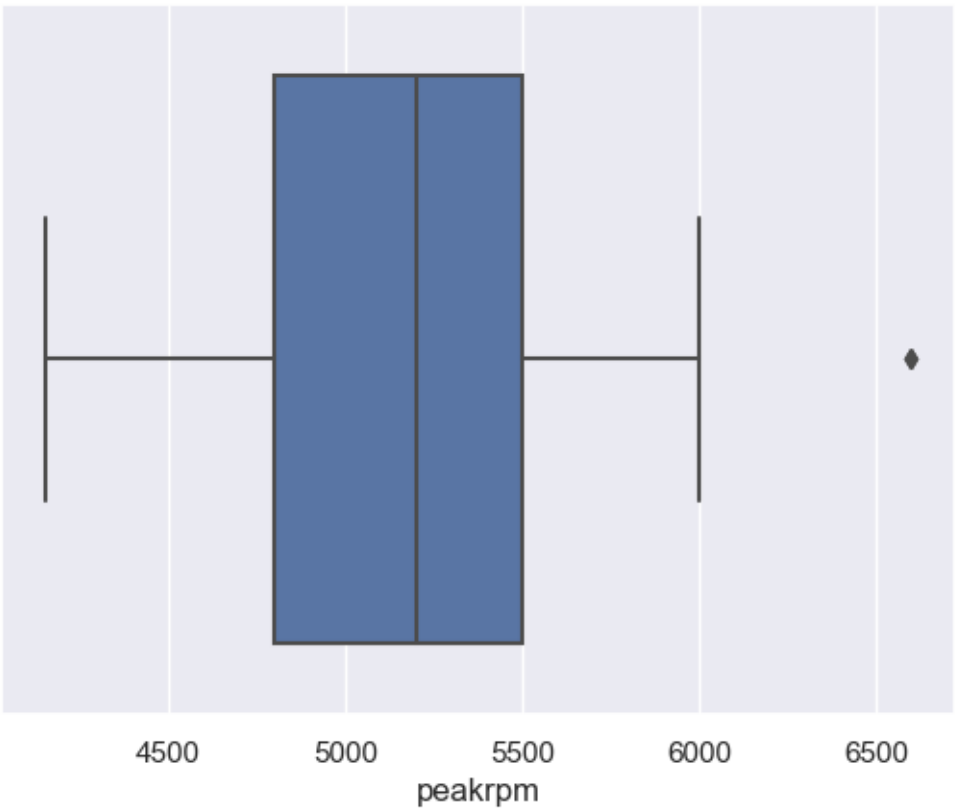

140
150
160
170
180
190
200
210
carlength

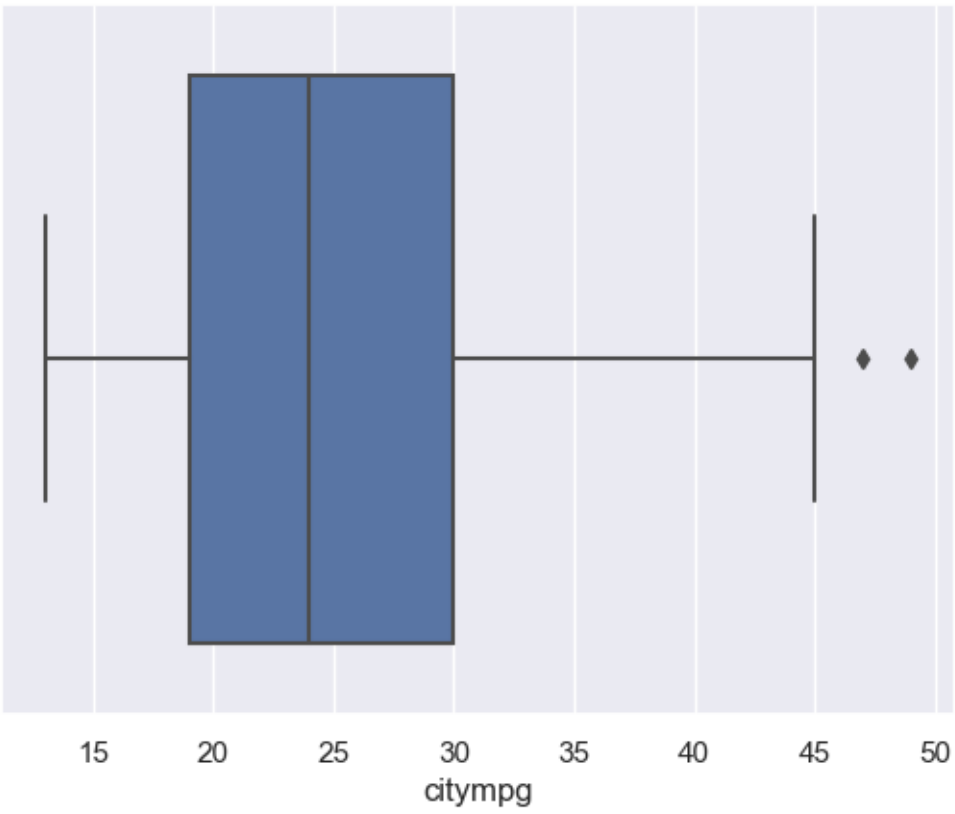

60
62
64
66
68
70
72
carwidth

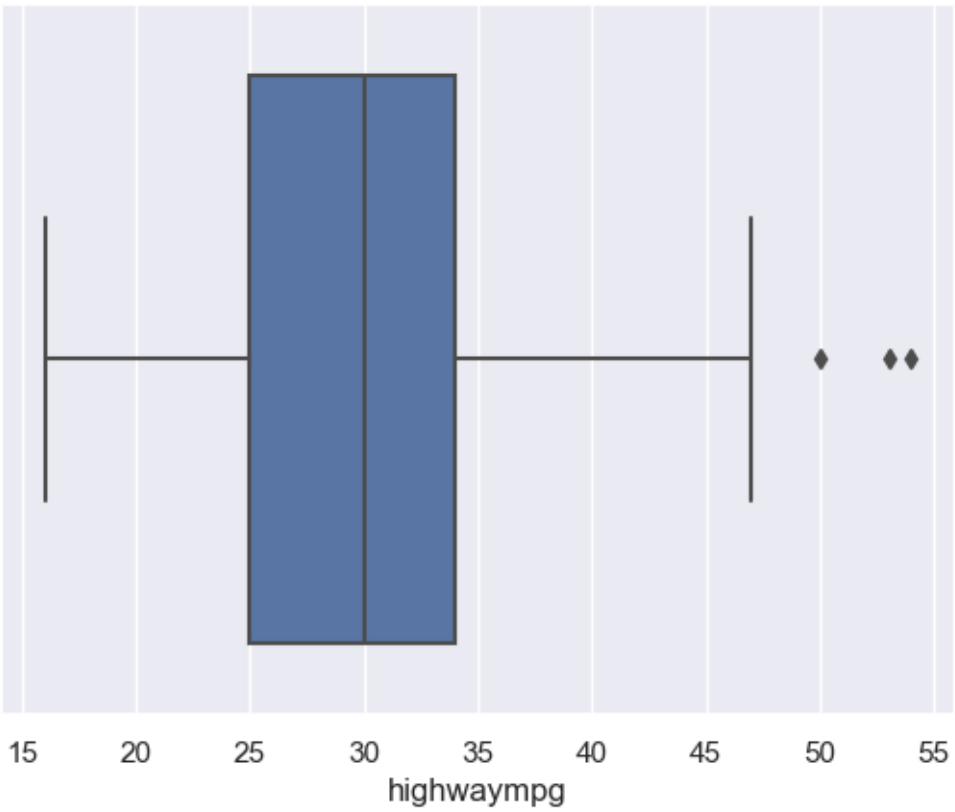

1500
2000
2500
3000
3500
4000
curbweight

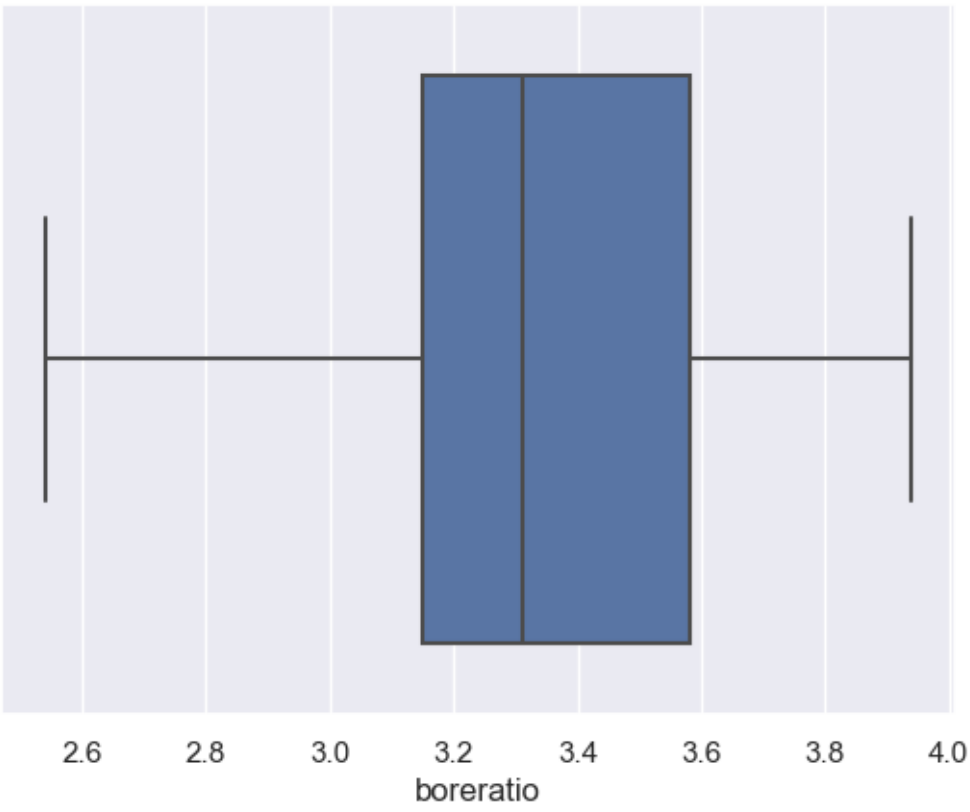

50
100
150
200
250
300
enginesize

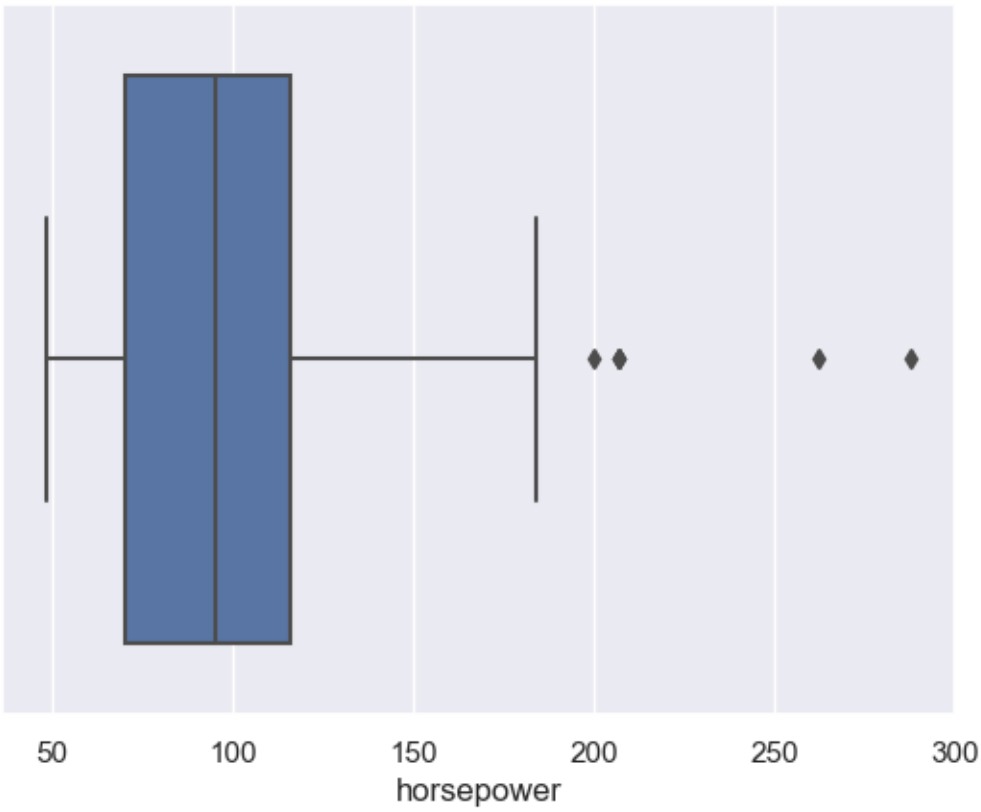

2.6
2.8
3.0
3.2
3.4
3.6
3.8
4.0
boreratio

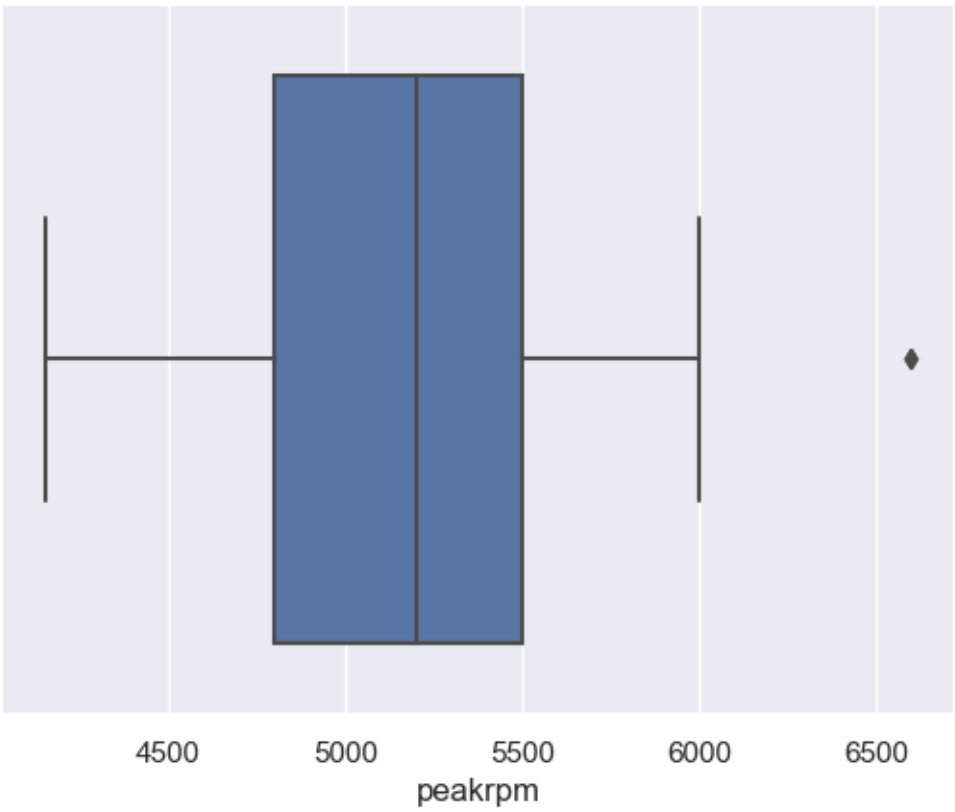

4500
5000
5500
6000
6500
peakrpm

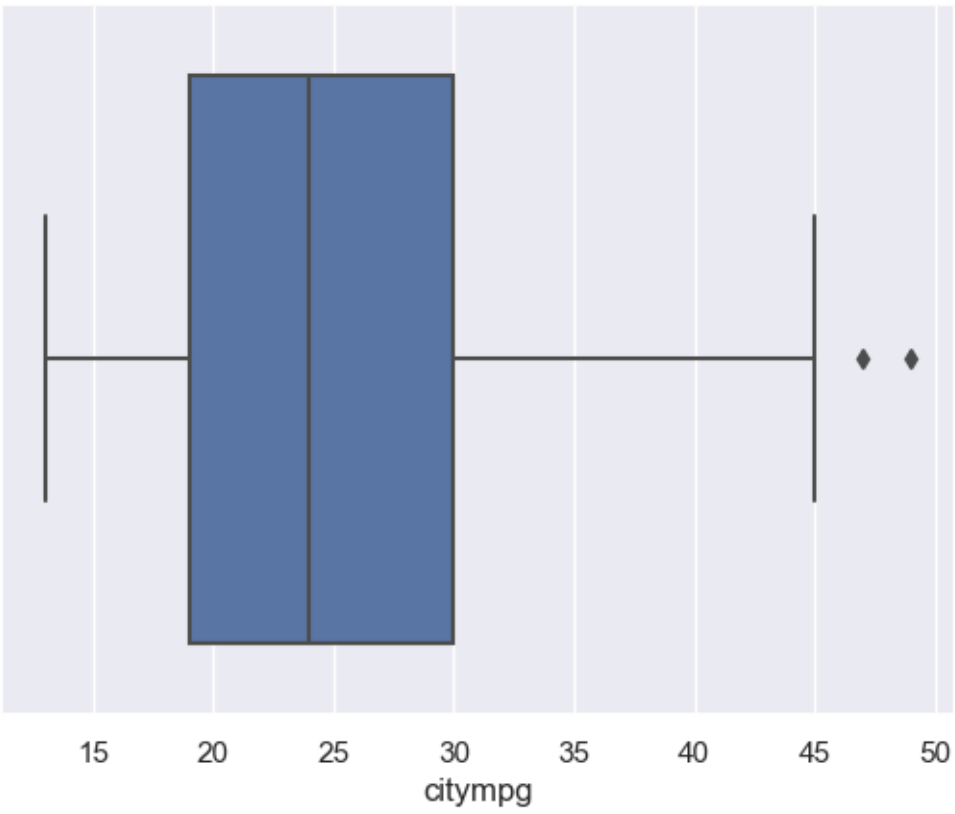

15
20
25
30
35
40
45
50
citympg

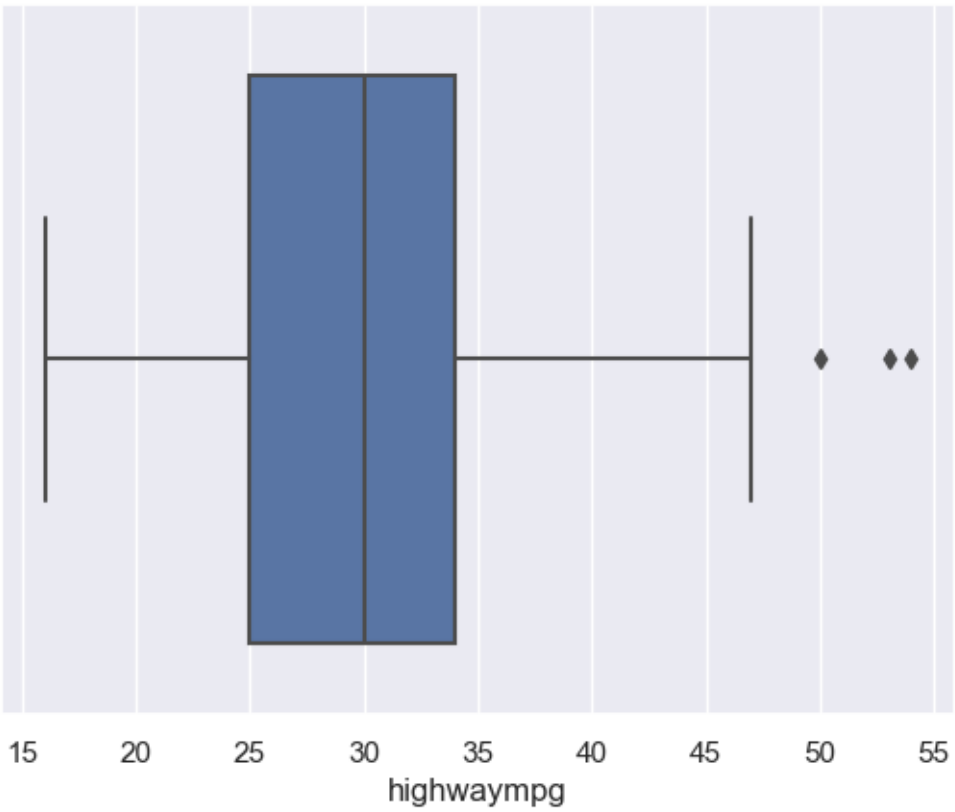

15
20
25
30
35
40
45
50
55
highwaympg

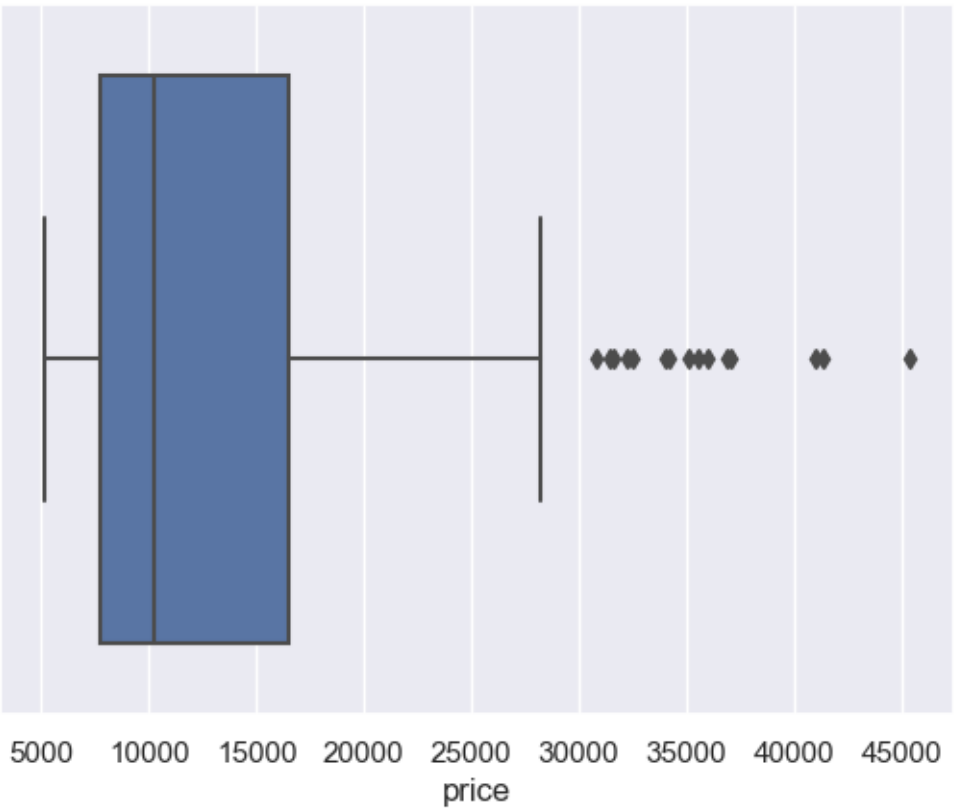

2.6
2.8
3.0
3.2
3.4
3.6
3.8
4.0
boreratio


50
100
150
200
250
300
horsepower


4500
5000
5500
6000
6500
peakrpm


15
20
25
30
35
40
45
50
citympg


15
20
25
30
35
40
45
50
55
highwaympg



[430]:
```
#Handling Outliers

# capping method required

Q1 = car.quantile(0.25)
Q3 = car.quantile(0.75)
IQR = Q3 - Q1

pos_outlier = Q3 + 1.5 * IQR

neg_outlier = Q1 - 1.5 * IQR
```

[432]:
```
print(Q1)
print("**************"*5)
print(Q3)
print("**************"*5)
print(IQR)
print("**************"*5)
print(pos_outlier)
print("**************"*5)
print(neg_outlier)
```

```
print("**************"*5)
```

```
wheelbase       94.50
carlength      166.30
carwidth        64.10
curbweight    2145.00
enginesize      97.00
boreratio        3.15
horsepower      70.00
peakrpm       4800.00
citympg         19.00
highwaympg      25.00
price         7788.00
Name: 0.25, dtype: float64
**********************************************************************
wheelbase       102.40
carlength       183.10
carwidth         66.90
curbweight     2935.00
enginesize      141.00
boreratio         3.58
horsepower      116.00
peakrpm        5500.00
citympg          30.00
highwaympg       34.00
price         16503.00
Name: 0.75, dtype: float64
**********************************************************************
wheelbase        7.90
carlength       16.80
carwidth         2.80
curbweight     790.00
enginesize      44.00
boreratio        0.43
horsepower      46.00
peakrpm        700.00
citympg         11.00
highwaympg       9.00
price         8715.00
dtype: float64
**********************************************************************
wheelbase       114.250
carlength       208.300
carwidth         71.100
curbweight     4120.000
enginesize      207.000
boreratio         4.225
```

```
horsepower      185.000
peakrpm        6550.000
citympg          46.500
highwaympg       47.500
price         29575.500
dtype: float64
**********************************************************************
wheelbase       82.650
carlength      141.100
carwidth        59.900
curbweight     960.000
enginesize      31.000
boreratio        2.505
horsepower       1.000
peakrpm       3750.000
citympg          2.500
highwaympg      11.500
price        -5284.500
dtype: float64
**********************************************************************
```

[433]:
```
income_q1 = car['wheelbase'].quantile(0.25)
income_q3 = car['wheelbase'].quantile(0.75)
income_iqr = income_q3 -  income_q1
income_upper = income_q3 + 1.5 * income_iqr
income_lower = income_q1 - 1.5 * income_iqr
```

[434]:
```
car['wheelbase'] = np.where(car['wheelbase'] > income_upper, income_upper,
                                    np.where(car['wheelbase'] < income_lower,␣
 ↪income_lower,
                                             car['wheelbase']) )
```

[435]:
```
income_q1 = car['carlength'].quantile(0.25)
income_q3 = car['carlength'].quantile(0.75)
income_iqr = income_q3 -  income_q1
income_upper = income_q3 + 1.5 * income_iqr
income_lower = income_q1 - 1.5 * income_iqr
```

[436]:
```
car['carlength'] = np.where(car['carlength'] > income_upper, income_upper,
                                    np.where(car['carlength'] < income_lower,␣
 ↪income_lower,
                                             car['carlength']) )
```

[437]:
```
income_q1 = car['carwidth'].quantile(0.25)
income_q3 = car['carwidth'].quantile(0.75)
income_iqr = income_q3 -  income_q1
income_upper = income_q3 + 1.5 * income_iqr
```

```
income_lower = income_q1 - 1.5 * income_iqr

car['carwidth'] = np.where(car['carwidth'] > income_upper, income_upper,
                                    np.where(car['carwidth'] < income_lower,␣
 ↪income_lower,
                                             car['carwidth']) )
```

[438]:
```
income_q1 = car['enginesize'].quantile(0.25)
income_q3 = car['enginesize'].quantile(0.75)
income_iqr = income_q3 -  income_q1
income_upper = income_q3 + 1.5 * income_iqr
income_lower = income_q1 - 1.5 * income_iqr

car['enginesize'] = np.where(car['enginesize'] > income_upper, income_upper,
                                    np.where(car['enginesize'] < income_lower,␣
 ↪income_lower,
                                             car['enginesize']) )
```

[439]:
```
income_q1 = car['curbweight'].quantile(0.25)
income_q3 = car['curbweight'].quantile(0.75)
income_iqr = income_q3 -  income_q1
income_upper = income_q3 + 1.5 * income_iqr
income_lower = income_q1 - 1.5 * income_iqr

car['curbweight'] = np.where(car['curbweight'] > income_upper, income_upper,
                                    np.where(car['curbweight'] < income_lower,␣
 ↪income_lower,
                                             car['curbweight']) )
```

[440]:
```
income_q1 = car['horsepower'].quantile(0.25)
income_q3 = car['horsepower'].quantile(0.75)
income_iqr = income_q3 -  income_q1
income_upper = income_q3 + 1.5 * income_iqr
income_lower = income_q1 - 1.5 * income_iqr

car['horsepower'] = np.where(car['horsepower'] > income_upper, income_upper,
                                    np.where(car['horsepower'] < income_lower,␣
 ↪income_lower,
                                             car['horsepower']) )
```

[441]:
```
income_q1 = car['boreratio'].quantile(0.25)
income_q3 = car['boreratio'].quantile(0.75)
income_iqr = income_q3 -  income_q1
income_upper = income_q3 + 1.5 * income_iqr
income_lower = income_q1 - 1.5 * income_iqr

car['boreratio'] = np.where(car['boreratio'] > income_upper, income_upper,
```

```
                                        np.where(car['boreratio'] < income_lower,␣
 ↪income_lower,
                                                 car['boreratio']) )
```

[442]:
```
income_q1 = car['peakrpm'].quantile(0.25)
income_q3 = car['peakrpm'].quantile(0.75)
income_iqr = income_q3 -  income_q1
income_upper = income_q3 + 1.5 * income_iqr
income_lower = income_q1 - 1.5 * income_iqr

car['peakrpm'] = np.where(car['peakrpm'] > income_upper, income_upper,
                                        np.where(car['peakrpm'] < income_lower,␣
 ↪income_lower,
                                                 car['peakrpm']) )
```

[443]:
```
income_q1 = car['citympg'].quantile(0.25)
income_q3 = car['citympg'].quantile(0.75)
income_iqr = income_q3 -  income_q1
income_upper = income_q3 + 1.5 * income_iqr
income_lower = income_q1 - 1.5 * income_iqr

car['citympg'] = np.where(car['citympg'] > income_upper, income_upper,
                                        np.where(car['citympg'] < income_lower,␣
 ↪income_lower,
                                                 car['citympg']) )
```

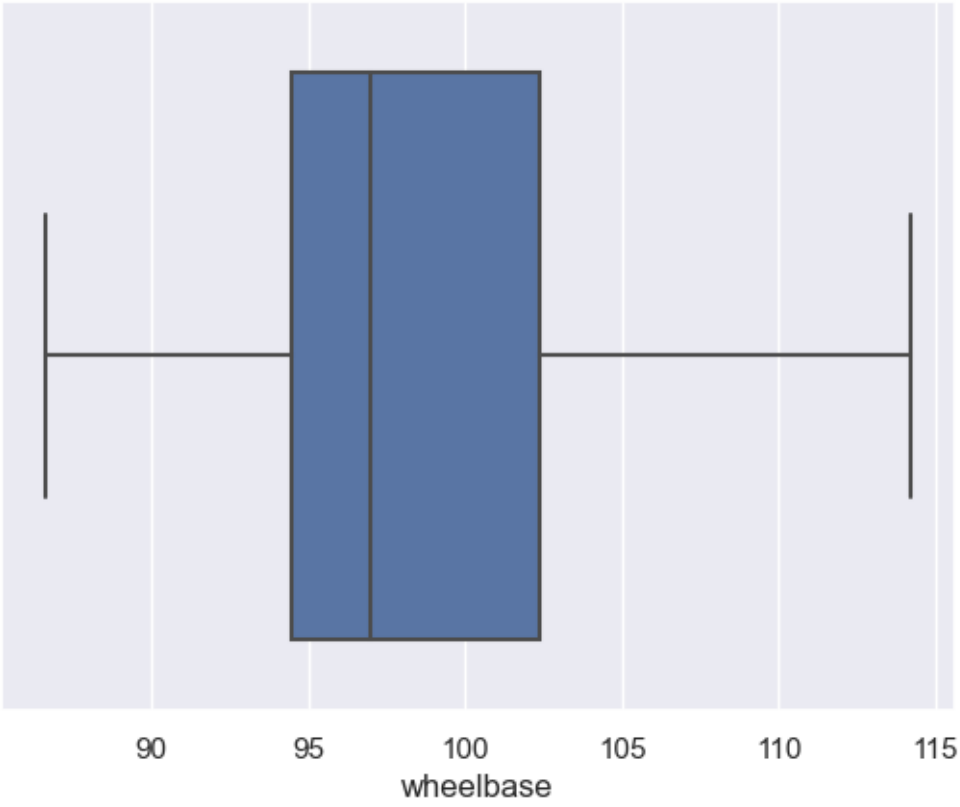
[444]:
```
income_q1 = car['highwaympg'].quantile(0.25)
income_q3 = car['highwaympg'].quantile(0.75)
income_iqr = income_q3 -  income_q1
income_upper = income_q3 + 1.5 * income_iqr
income_lower = income_q1 - 1.5 * income_iqr

car['highwaympg'] = np.where(car['highwaympg'] > income_upper, income_upper,
                                        np.where(car['highwaympg'] < income_lower,␣
 ↪income_lower,
                                                 car['highwaympg']) )
```

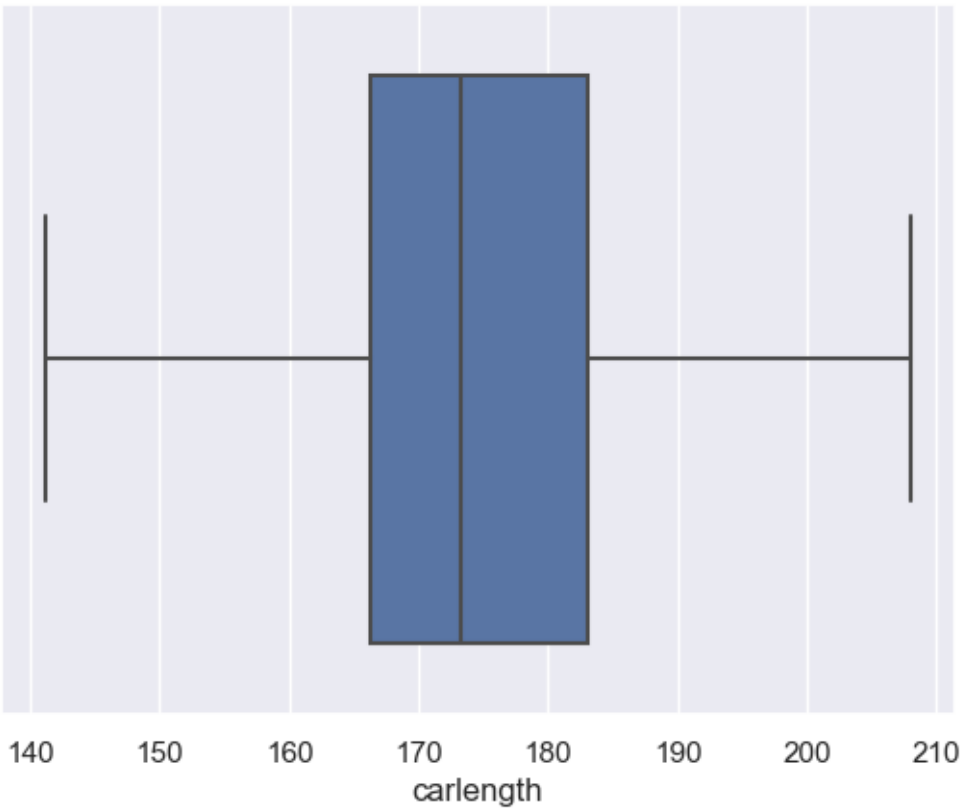
[445]:
```
income_q1 = car['price'].quantile(0.25)
income_q3 = car['price'].quantile(0.75)
income_iqr = income_q3 -  income_q1
income_upper = income_q3 + 1.5 * income_iqr
income_lower = income_q1 - 1.5 * income_iqr

car['price'] = np.where(car['price'] > income_upper, income_upper,
                                        np.where(car['price'] < income_lower,␣
 ↪income_lower,
```

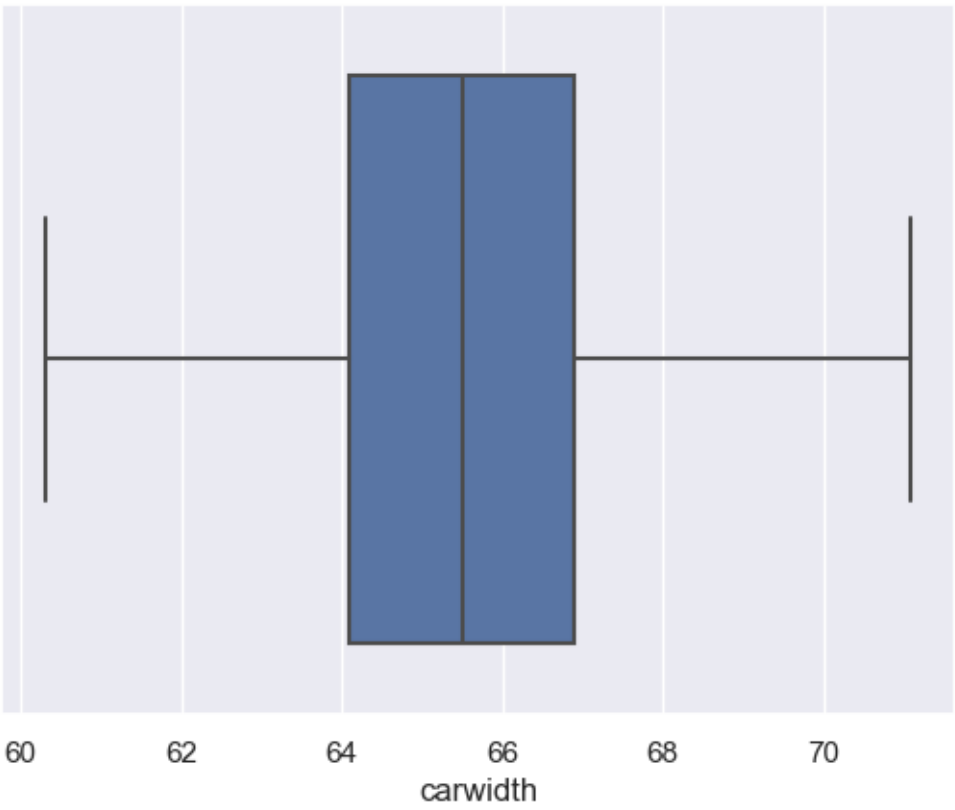
```
                                                car['price']) )
```

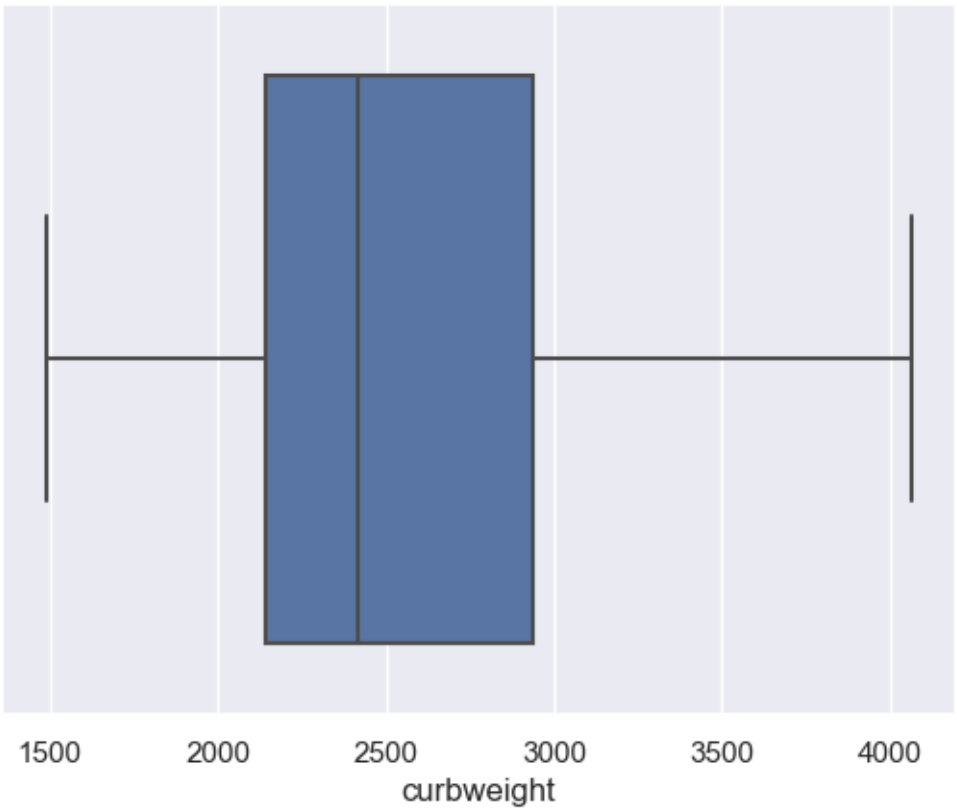
[446]:
```
#User Box plot to check if any outliers are there

#Check for outliers

car.describe()

sns.boxplot(x=car['wheelbase'])
plt.show()
sns.boxplot(x=car['carlength'])
plt.show()

sns.boxplot(x=car['carwidth'])
plt.show()

sns.boxplot(x=car['curbweight'])
plt.show()
sns.boxplot(x=car['enginesize'])
plt.show()

sns.boxplot(x=car['boreratio'])
plt.show()

sns.boxplot(x=car['peakrpm'])
plt.show()
sns.boxplot(x=car['citympg'])
plt.show()
sns.boxplot(x=car['highwaympg'])
plt.show()

#Check for outliers

car.describe()

sns.boxplot(x=car['boreratio'])
plt.show()
sns.boxplot(x=car['horsepower'])
plt.show()

sns.boxplot(x=car['peakrpm'])
plt.show()
sns.boxplot(x=car['citympg'])
plt.show()
sns.boxplot(x=car['highwaympg'])
plt.show()
```

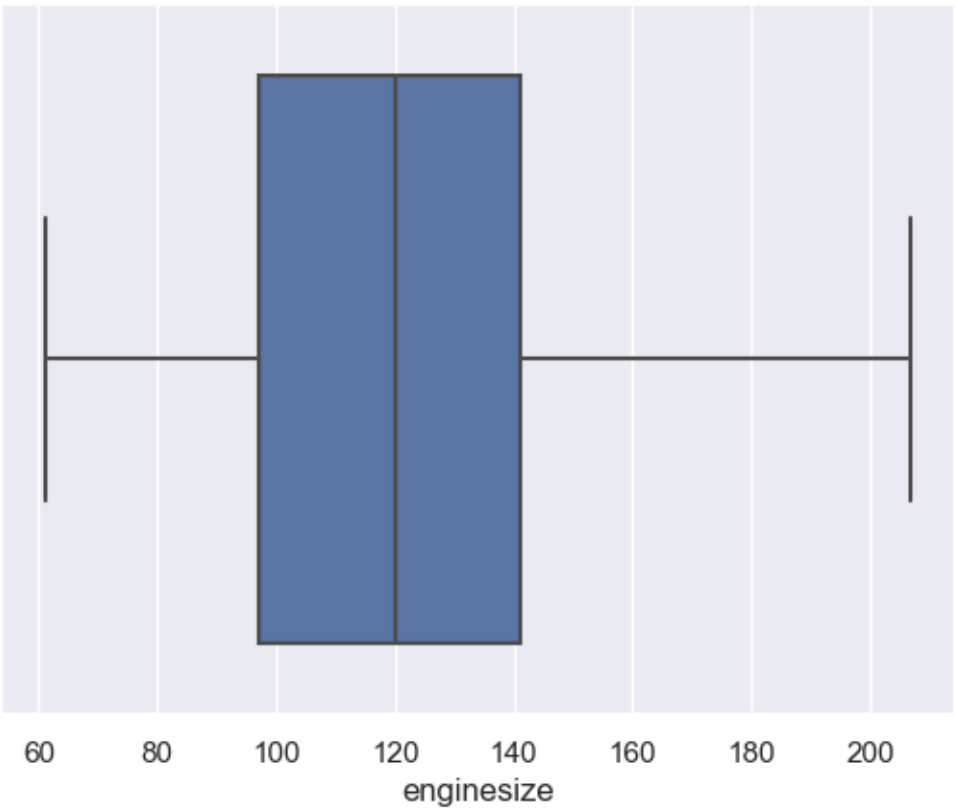
```
sns.boxplot(x=car['price'])
plt.show()
```

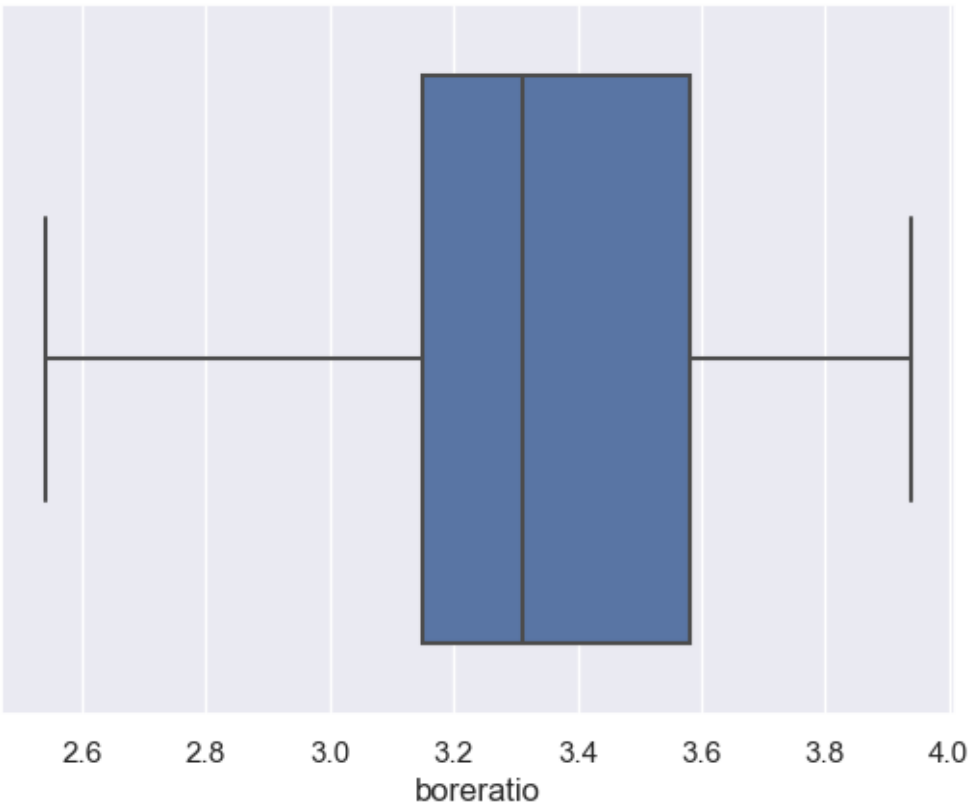


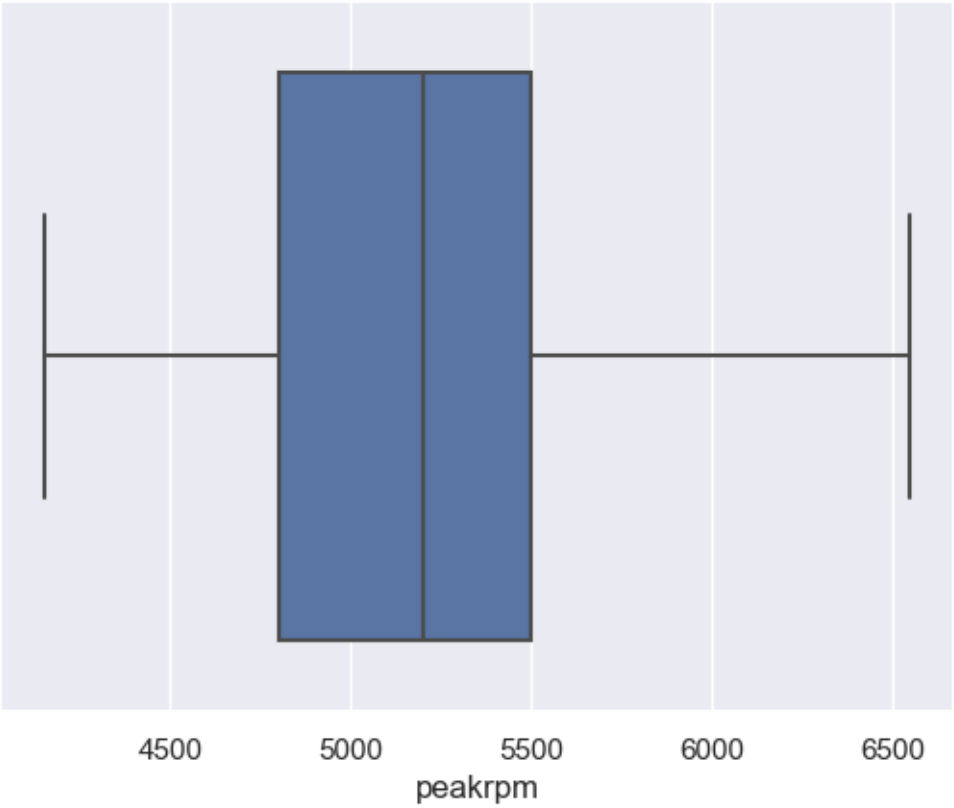

140
150
160
170
180
190
200
210
carlength

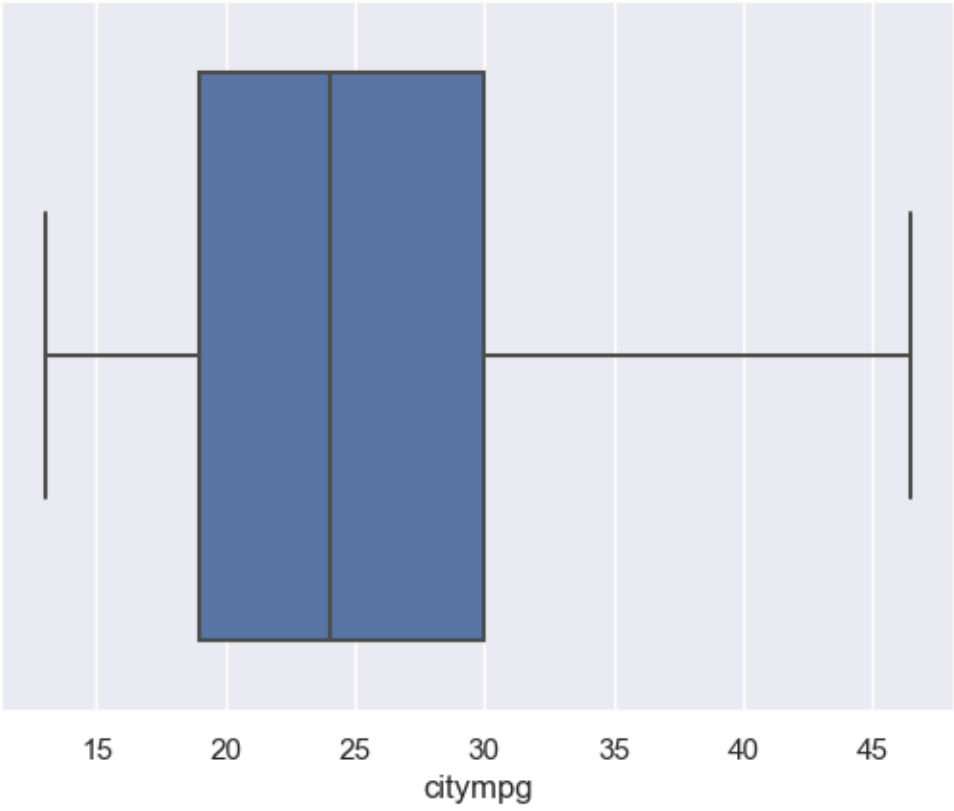

60
62
64
66
68
70
carwidth

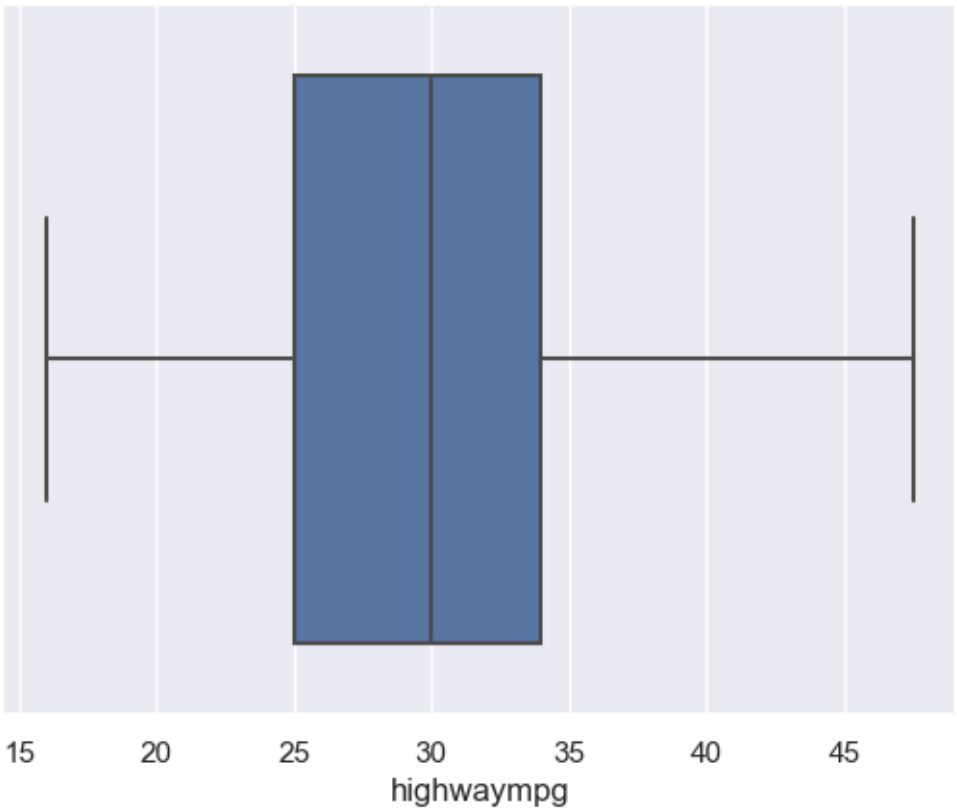

1500
2000
2500
3000
3500
4000
curbweight

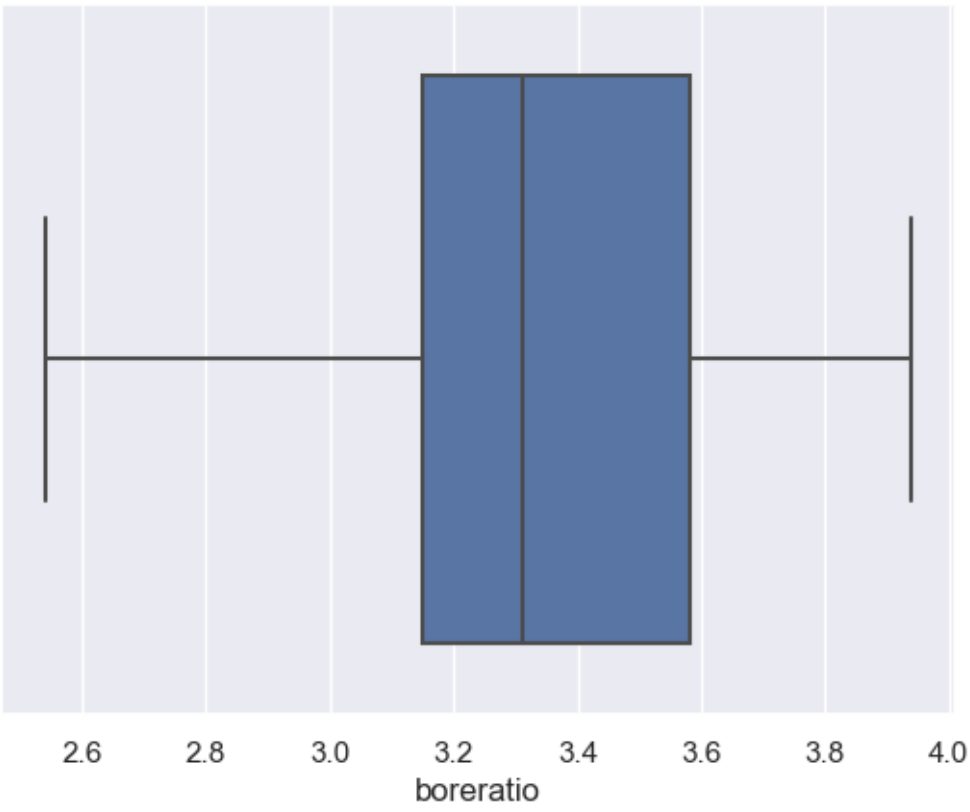

60
80
100
120
140
160
180
200
enginesize

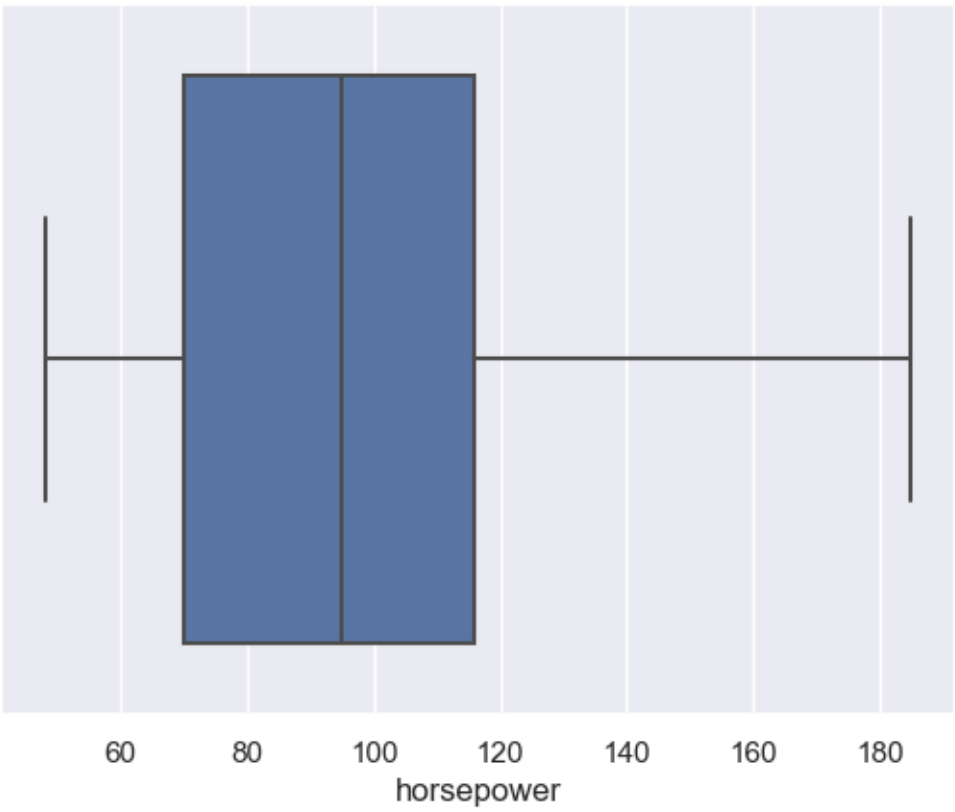

2.6
2.8
3.0
3.2
3.4
3.6
3.8
4.0
boreratio

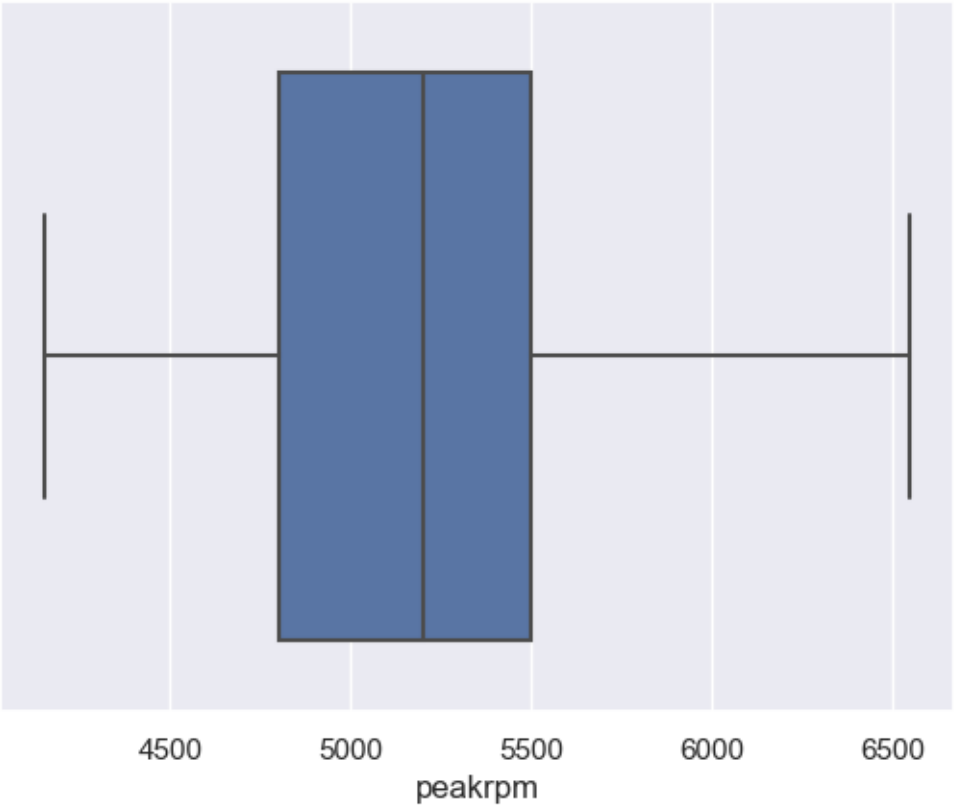

4500
5000
5500
6000
6500
peakrpm

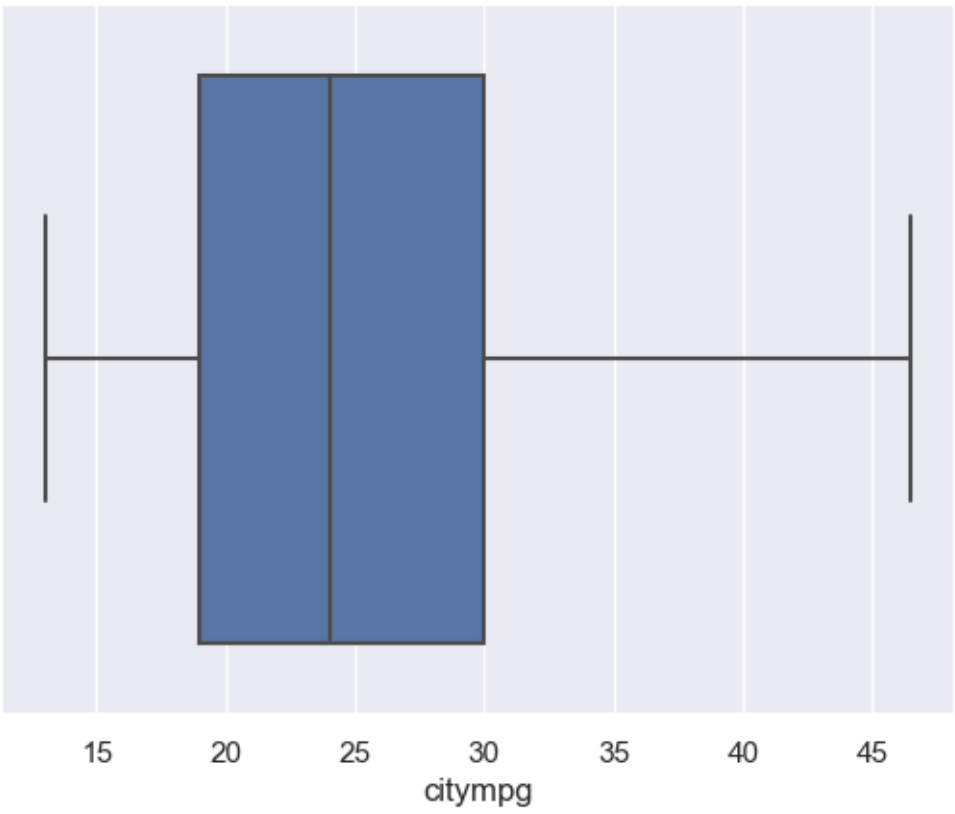

15
20
25
30
35
40
45
citympg

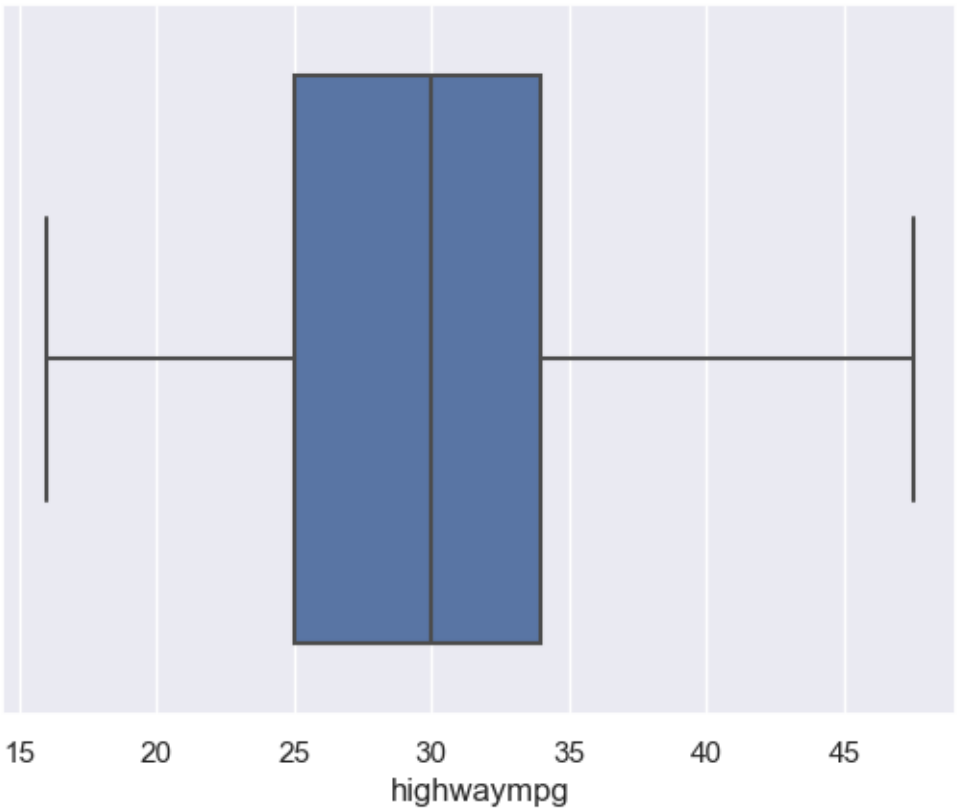

15
20
25
30
35
40
45
highwaympg

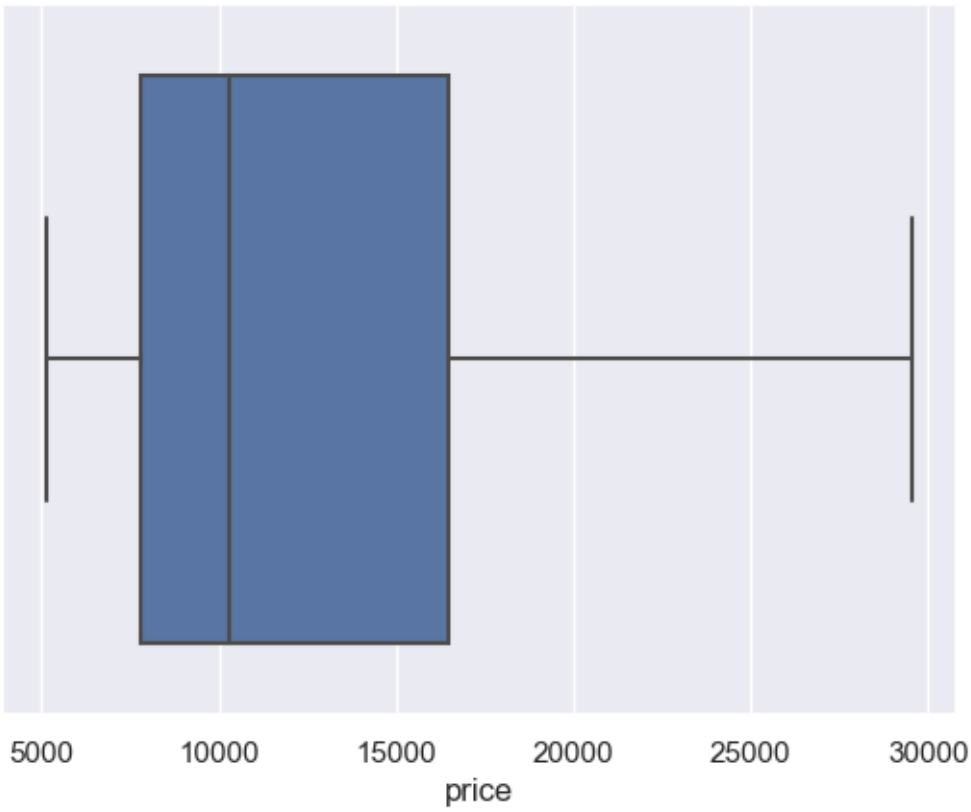

2.6
2.8
3.0
3.2
3.4
3.6
3.8
4.0
boreratio


60
80
100
120
140
160
180
horsepower


4500
5000
5500
6000
6500
peakrpm


15
20
25
30
35
40
45
citympg


15
20
25
30
35
40
45
highwaympg



[447]:
```
#Outliers Handled

#Encoding concept on object data
car.info()
# CarName, aspiration, carbody, drivewheel, enginelocation, enginetype,␣
 ↪cylindernumber are the object data

car = pd.get_dummies(car, columns=['CompanyName'])

car = pd.get_dummies(car, columns=['drivewheel'])
car['aspiration'].value_counts()

car['aspiration']=car['aspiration'].astype('category')
car['aspiration']=car['aspiration'].cat.codes

car['enginelocation'].value_counts()# 2 categories-use label encoding

car['enginelocation']=car['enginelocation'].astype('category')
car['enginelocation']=car['enginelocation'].cat.codes

car['carbody'].value_counts()# 5 categories-use one hot encoding
```

```
car = pd.get_dummies(car, columns=['carbody'])



car = pd.get_dummies(car, columns=['enginetype'])
car = pd.get_dummies(car, columns=['cylindernumber'])

car['fuelsystem'].value_counts()
car = pd.get_dummies(car, columns=['fuelsystem'])
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 205 entries, 0 to 204
Data columns (total 19 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   CompanyName     205 non-null    object
 1   aspiration      205 non-null    object
 2   carbody         205 non-null    object
 3   drivewheel      205 non-null    object
 4   enginelocation  205 non-null    object
 5   wheelbase       205 non-null    float64
 6   carlength       205 non-null    float64
 7   carwidth        205 non-null    float64
 8   curbweight      205 non-null    float64
 9   enginetype      205 non-null    object
 10  cylindernumber  205 non-null    object
 11  enginesize      205 non-null    float64
 12  fuelsystem      205 non-null    object
 13  boreratio       205 non-null    float64
 14  horsepower      205 non-null    float64
 15  peakrpm         205 non-null    float64
 16  citympg         205 non-null    float64
 17  highwaympg      205 non-null    float64
 18  price           205 non-null    float64
dtypes: float64(11), object(8)
memory usage: 32.0+ KB
```

[448]:
```
car.head()
#Drop carbody_wagon as we can drop one when used one hot encoding
car.drop(['carbody_wagon'],axis=1,inplace=True)
```

[449]:
```
car.head()
```

[449]:

	aspiration	enginelocation	wheelbase	carlength	carwidth	curbweight	\
0	0	0	88.6	168.8	64.1	2548.0	
1	0	0	88.6	168.8	64.1	2548.0	
2	0	0	94.5	171.2	65.5	2823.0	

```
3           0              0       99.8      176.6      66.2      2337.0
4           0              0       99.4      176.6      66.4      2824.0

   enginesize  boreratio  horsepower  peakrpm  …  cylindernumber_twelve  \
0       130.0       3.47       111.0   5000.0  …                      0
1       130.0       3.47       111.0   5000.0  …                      0
2       152.0       2.68       154.0   5000.0  …                      0
3       109.0       3.19       102.0   5500.0  …                      0
4       136.0       3.19       115.0   5500.0  …                      0

   cylindernumber_two  fuelsystem_1bbl  fuelsystem_2bbl  fuelsystem_4bbl  \
0                   0                0                0                0
1                   0                0                0                0
2                   0                0                0                0
3                   0                0                0                0
4                   0                0                0                0

   fuelsystem_idi  fuelsystem_mfi  fuelsystem_mpfi  fuelsystem_spdi  \
0               0               0                1                0
1               0               0                1                0
2               0               0                1                0
3               0               0                1                0
4               0               0                1                0

   fuelsystem_spfi
0                0
1                0
2                0
3                0
4                0

[5 rows x 70 columns]
```

[ ]:

[450]:
```
#creating duplicate data

car_copy=car.copy()
car_copy.head()
```

[450]:
```
   aspiration  enginelocation  wheelbase  carlength  carwidth  curbweight  \
0           0               0       88.6      168.8      64.1      2548.0
1           0               0       88.6      168.8      64.1      2548.0
2           0               0       94.5      171.2      65.5      2823.0
3           0               0       99.8      176.6      66.2      2337.0
4           0               0       99.4      176.6      66.4      2824.0
```

```
   enginesize  boreratio  horsepower  peakrpm  …  cylindernumber_twelve  \
0       130.0       3.47       111.0   5000.0  …                      0
1       130.0       3.47       111.0   5000.0  …                      0
2       152.0       2.68       154.0   5000.0  …                      0
3       109.0       3.19       102.0   5500.0  …                      0
4       136.0       3.19       115.0   5500.0  …                      0

   cylindernumber_two  fuelsystem_1bbl  fuelsystem_2bbl  fuelsystem_4bbl  \
0                   0                0                0                0
1                   0                0                0                0
2                   0                0                0                0
3                   0                0                0                0
4                   0                0                0                0

   fuelsystem_idi  fuelsystem_mfi  fuelsystem_mpfi  fuelsystem_spdi  \
0               0               0                1                0
1               0               0                1                0
2               0               0                1                0
3               0               0                1                0
4               0               0                1                0

   fuelsystem_spfi
0                0
1                0
2                0
3                0
4                0

[5 rows x 70 columns]
```

[451]:
```
#Splitting the data into dependant and independant variables
x=car_copy.loc[:, car_copy.columns != 'price']
y=car_copy['price']
```

[452]:
```
x.head()
```

[452]:
```
   aspiration  enginelocation  wheelbase  carlength  carwidth  curbweight  \
0           0               0       88.6      168.8      64.1      2548.0
1           0               0       88.6      168.8      64.1      2548.0
2           0               0       94.5      171.2      65.5      2823.0
3           0               0       99.8      176.6      66.2      2337.0
4           0               0       99.4      176.6      66.4      2824.0

   enginesize  boreratio  horsepower  peakrpm  …  cylindernumber_twelve  \
0       130.0       3.47       111.0   5000.0  …                      0
1       130.0       3.47       111.0   5000.0  …                      0
2       152.0       2.68       154.0   5000.0  …                      0
```

```
3       109.0       3.19       102.0   5500.0  …                     0
4       136.0       3.19       115.0   5500.0  …                     0

   cylindernumber_two  fuelsystem_1bbl  fuelsystem_2bbl  fuelsystem_4bbl  \
0                   0                0                0                0
1                   0                0                0                0
2                   0                0                0                0
3                   0                0                0                0
4                   0                0                0                0

   fuelsystem_idi  fuelsystem_mfi  fuelsystem_mpfi  fuelsystem_spdi  \
0               0               0                1                0
1               0               0                1                0
2               0               0                1                0
3               0               0                1                0
4               0               0                1                0

   fuelsystem_spfi
0                0
1                0
2                0
3                0
4                0

[5 rows x 69 columns]
```

[453]:
```
y.head()
```

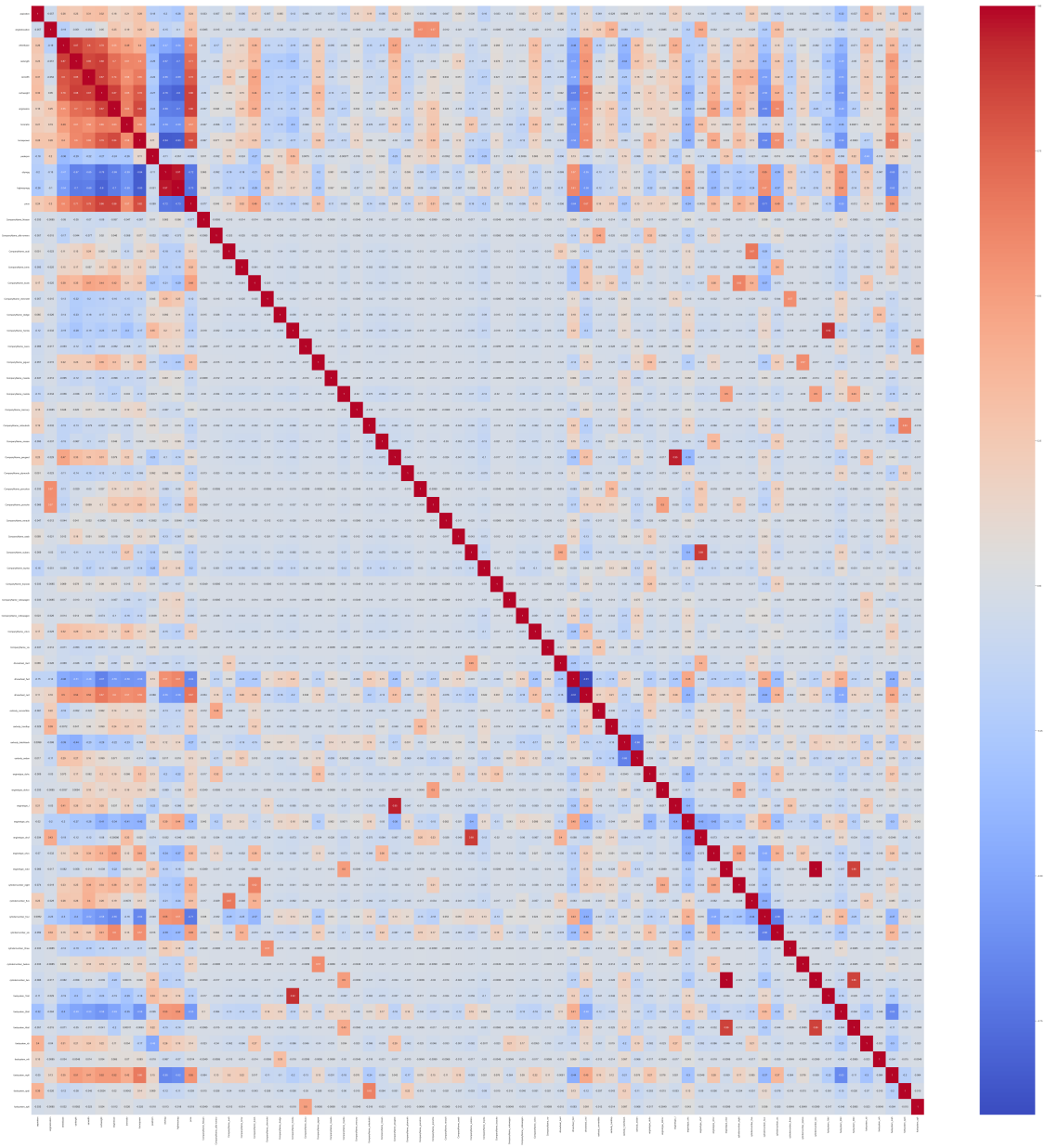
[453]:
```
0    13495.0
1    16500.0
2    16500.0
3    13950.0
4    17450.0
Name: price, dtype: float64
```

[454]:
```
#Feature Scaling

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
sc_x = sc.fit_transform(x)
pd.DataFrame(sc_x)
```

[454]:
```
          0         1         2         3         4         5         6   \
0  -0.469295 -0.121867 -1.723005 -0.426521 -0.858695 -0.014566  0.160196
1  -0.469295 -0.121867 -1.723005 -0.426521 -0.858695 -0.014566  0.160196
2  -0.469295 -0.121867 -0.717590 -0.231513 -0.184978  0.514882  0.809329
3  -0.469295 -0.121867  0.185580  0.207256  0.151880 -0.420797 -0.459430
```

```
4   -0.469295 -0.121867  0.117416  0.207256  0.248125  0.516807  0.337232
..        …       …        …         …         …         …
200 -0.469295 -0.121867  1.770387  1.198549  1.451189  0.763241  0.484762
201  2.130854 -0.121867  1.770387  1.198549  1.403066  0.949992  0.484762
202 -0.469295 -0.121867  1.770387  1.198549  1.451189  0.878757  1.428955
203  2.130854 -0.121867  1.770387  1.198549  1.451189  1.273437  0.602787
204  2.130854 -0.121867  1.770387  1.198549  1.451189  0.975021  0.484762

           7         8         9   …        59        60       61        62  \
0    0.519071  0.229801 -0.262757  … -0.070014 -0.141069 -0.23812 -0.689072
1    0.519071  0.229801 -0.262757  … -0.070014 -0.141069 -0.23812 -0.689072
2   -2.404880  1.441341 -0.262757  … -0.070014 -0.141069 -0.23812 -0.689072
3   -0.517266 -0.023777  0.791357  … -0.070014 -0.141069 -0.23812 -0.689072
4   -0.517266  0.342502  0.791357  … -0.070014 -0.141069 -0.23812 -0.689072
..        …       …        …  …       …        …       …        …
200  1.666445  0.314327  0.580534  … -0.070014 -0.141069 -0.23812 -0.689072
201  1.666445  1.610393  0.369711  … -0.070014 -0.141069 -0.23812 -0.689072
202  0.926204  0.877834  0.791357  … -0.070014 -0.141069 -0.23812 -0.689072
203 -1.183483  0.088924 -0.684403  … -0.070014 -0.141069 -0.23812 -0.689072
204  1.666445  0.314327  0.580534  … -0.070014 -0.141069 -0.23812 -0.689072

           63        64        65        66        67        68
0   -0.121867 -0.328798 -0.070014  1.086670 -0.214286 -0.070014
1   -0.121867 -0.328798 -0.070014  1.086670 -0.214286 -0.070014
2   -0.121867 -0.328798 -0.070014  1.086670 -0.214286 -0.070014
3   -0.121867 -0.328798 -0.070014  1.086670 -0.214286 -0.070014
4   -0.121867 -0.328798 -0.070014  1.086670 -0.214286 -0.070014
..        …       …        …         …         …         …
200 -0.121867 -0.328798 -0.070014  1.086670 -0.214286 -0.070014
201 -0.121867 -0.328798 -0.070014  1.086670 -0.214286 -0.070014
202 -0.121867 -0.328798 -0.070014  1.086670 -0.214286 -0.070014
203 -0.121867  3.041381 -0.070014 -0.920243 -0.214286 -0.070014
204 -0.121867 -0.328798 -0.070014  1.086670 -0.214286 -0.070014

[205 rows x 69 columns]
```

[455]:
```
# Finding correlation
plt.figure(figsize=(100,100))
corr = car_copy.corr()
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.show()
```



[456]:
```
#varience Inflation Factor

variable = sc_x
variable.shape

from statsmodels.stats.outliers_influence import variance_inflation_factor
variable = sc_x

vif = pd.DataFrame()
```

```
vif['Variance Inflation Factor'] = [variance_inflation_factor(variable, i ) for
 ↪i in range(variable.shape[1])]

vif['Features'] = x.columns


with pd.option_context('display.max_rows', None,):
    print(vif)
```

```
    Variance Inflation Factor                Features
0                    6.010858              aspiration
1                         inf           enginelocation
2                   15.179397               wheelbase
3                   23.359546               carlength
4                   14.328416                carwidth
5                   50.367924              curbweight
6                   56.547055              enginesize
7                   14.251309               boreratio
8                   36.682381              horsepower
9                    6.015534                 peakrpm
10                  42.952847                 citympg
11                  39.579347              highwaympg
12                        inf       CompanyName_Nissan
13                        inf  CompanyName_alfa-romero
14                        inf         CompanyName_audi
15                        inf          CompanyName_bmw
16                        inf        CompanyName_buick
17                        inf    CompanyName_chevrolet
18                        inf        CompanyName_dodge
19                        inf        CompanyName_honda
20                        inf        CompanyName_isuzu
21                        inf       CompanyName_jaguar
22                        inf        CompanyName_maxda
23                        inf        CompanyName_mazda
24                        inf      CompanyName_mercury
25                        inf   CompanyName_mitsubishi
26                        inf       CompanyName_nissan
27                        inf      CompanyName_peugeot
28                        inf     CompanyName_plymouth
29                        inf     CompanyName_porcshce
30                        inf      CompanyName_porsche
31                        inf      CompanyName_renault
32                        inf         CompanyName_saab
33                        inf       CompanyName_subaru
34                        inf       CompanyName_toyota
35                        inf      CompanyName_toyouta
36                        inf    CompanyName_vokswagen
```

```
37                      inf   CompanyName_volkswagen
38                      inf        CompanyName_volvo
39                      inf           CompanyName_vw
40                      inf           drivewheel_4wd
41                      inf           drivewheel_fwd
42                      inf           drivewheel_rwd
43                 3.153464      carbody_convertible
44                 2.447291          carbody_hardtop
45                 5.720502        carbody_hatchback
46                 4.331157            carbody_sedan
47                      inf          enginetype_dohc
48                      inf         enginetype_dohcv
49                      inf             enginetype_l
50                      inf           enginetype_ohc
51                      inf          enginetype_ohcf
52                      inf          enginetype_ohcv
53                      inf         enginetype_rotor
54                      inf     cylindernumber_eight
55                      inf      cylindernumber_five
56                      inf      cylindernumber_four
57                      inf       cylindernumber_six
58                      inf     cylindernumber_three
59                      inf    cylindernumber_twelve
60                      inf       cylindernumber_two
61                      inf          fuelsystem_1bbl
62                      inf          fuelsystem_2bbl
63                      inf          fuelsystem_4bbl
64                      inf           fuelsystem_idi
65                      inf           fuelsystem_mfi
66                      inf          fuelsystem_mpfi
67                      inf          fuelsystem_spdi
68                      inf          fuelsystem_spfi
```

[458]:
```
#drop the columns which has huge multi collinearity

x.drop(['aspiration','wheelbase','carwidth','highwaympg',
        ␣
 ↪'drivewheel_rwd','enginetype_rotor','fuelsystem_spfi'],axis=1,inplace=True)
```

[ ]:
```

```

[459]:
```
#Variance Inflation factor

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
sc_x = sc.fit_transform(x)
variable = sc_x
```

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif = pd.DataFrame()
vif['Variance Inflation Factor'] = [variance_inflation_factor(variable, i ) for␣
 ↪i in range(variable.shape[1])]
vif['Features'] = x.columns
```

[461]:
```
with pd.option_context('display.max_rows', None,):
    print(vif)
```

```
    Variance Inflation Factor                Features
0                         inf           enginelocation
1                   15.666647               carlength
2                   48.885647              curbweight
3                   45.781063              enginesize
4                   14.077264               boreratio
5                   19.959485              horsepower
6                    4.922656                 peakrpm
7                   12.409658                 citympg
8                         inf       CompanyName_Nissan
9                         inf  CompanyName_alfa-romero
10                        inf         CompanyName_audi
11                        inf          CompanyName_bmw
12                        inf        CompanyName_buick
13                        inf    CompanyName_chevrolet
14                        inf        CompanyName_dodge
15                        inf        CompanyName_honda
16                        inf        CompanyName_isuzu
17                        inf       CompanyName_jaguar
18                        inf        CompanyName_maxda
19                        inf        CompanyName_mazda
20                        inf      CompanyName_mercury
21                        inf   CompanyName_mitsubishi
22                        inf       CompanyName_nissan
23                        inf      CompanyName_peugeot
24                        inf     CompanyName_plymouth
25                        inf     CompanyName_porcshce
26                        inf      CompanyName_porsche
27                        inf      CompanyName_renault
28                        inf         CompanyName_saab
29                        inf       CompanyName_subaru
30                        inf       CompanyName_toyota
31                        inf      CompanyName_toyouta
32                        inf    CompanyName_vokswagen
33                        inf   CompanyName_volkswagen
34                        inf        CompanyName_volvo
35                        inf           CompanyName_vw
36                   2.864001           drivewheel_4wd
```

```
37                   6.742582           drivewheel_fwd
38                   2.889113      carbody_convertible
39                   2.387878          carbody_hardtop
40                   5.280902        carbody_hatchback
41                   4.117458            carbody_sedan
42                        inf          enginetype_dohc
43                        inf         enginetype_dohcv
44                        inf             enginetype_l
45                        inf           enginetype_ohc
46                        inf          enginetype_ohcf
47                        inf          enginetype_ohcv
48                        inf     cylindernumber_eight
49                        inf      cylindernumber_five
50                        inf      cylindernumber_four
51                        inf       cylindernumber_six
52                        inf     cylindernumber_three
53                        inf    cylindernumber_twelve
54                        inf       cylindernumber_two
55                  23.280211          fuelsystem_1bbl
56                  69.772633          fuelsystem_2bbl
57                   8.539725          fuelsystem_4bbl
58                  32.123953           fuelsystem_idi
59                   2.766519           fuelsystem_mfi
60                  82.740784          fuelsystem_mpfi
61                  15.479965          fuelsystem_spdi
```

[469]:
```
x.drop(['enginetype_ohc','CompanyName_subaru'],axis=1,inplace=True)
```

[470]:
```
#Variance Inflation factor

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
sc_x = sc.fit_transform(x)
variable = sc_x
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif = pd.DataFrame()
vif['Variance Inflation Factor'] = [variance_inflation_factor(variable, i ) for␣
 ↪i in range(variable.shape[1])]
vif['Features'] = x.columns

with pd.option_context('display.max_rows', None,):
    print(vif)
```

```
    Variance Inflation Factor                Features
0                    4.005195              boreratio
1                    2.770900                peakrpm
2                    4.024624                citympg
3                    1.067480      CompanyName_Nissan
```

```
4                    1.735435  CompanyName_alfa-romero
5                    1.785345        CompanyName_audi
6                    1.658280         CompanyName_bmw
7                    2.512522       CompanyName_buick
8                    1.777554    CompanyName_chevrolet
9                    1.692279       CompanyName_dodge
10                   2.394757       CompanyName_honda
11                   1.200455       CompanyName_isuzu
12                   1.436031      CompanyName_jaguar
13                   1.120749       CompanyName_maxda
14                   1.802784       CompanyName_mazda
15                   1.109786     CompanyName_mercury
16                   1.664874   CompanyName_mitsubishi
17                   2.012533      CompanyName_nissan
18                   1.389470    CompanyName_plymouth
19                   1.328371    CompanyName_porcshce
20                   1.998111     CompanyName_porsche
21                   1.138341     CompanyName_renault
22                   1.330938        CompanyName_saab
23                   1.213816     CompanyName_toyouta
24                   1.096893    CompanyName_vokswagen
25                   1.620950   CompanyName_volkswagen
26                   2.023741       CompanyName_volvo
27                   1.297563          CompanyName_vw
28                   1.478034          drivewheel_4wd
29                   2.073031     carbody_convertible
30                   1.745205         carbody_hardtop
31                   3.457466       carbody_hatchback
32                   3.149755           carbody_sedan
33                   2.238714         enginetype_dohc
34                   2.324809        enginetype_dohcv
35                   1.885751            enginetype_l
36                   2.230338         enginetype_ohcf
37                   2.841044         enginetype_ohcv
38                   3.617330    cylindernumber_eight
39                   1.736958    cylindernumber_three
40                   2.465294         fuelsystem_2bbl
41                   1.557707         fuelsystem_4bbl
42                   1.207708          fuelsystem_mfi
```

[528]:
```
#Split the data into train and test

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25,␣
 ↪random_state=101)
```

```
print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)
```

```
(153, 43) (52, 43) (153,) (52,)
```

[529]:
```
#OLS Method for Linear regression - training data

from statsmodels.regression.linear_model import OLS
import statsmodels.regression.linear_model as smf

reg_model = smf.OLS(endog = y_train, exog=x_train).fit()

reg_model.summary()
```

[529]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:                   price   R-squared (uncentered):
0.982
Model:                             OLS   Adj. R-squared (uncentered):
0.975
Method:                  Least Squares   F-statistic:
148.8
Date:                 Sun, 03 Sep 2023   Prob (F-statistic):
8.59e-81
Time:                         15:43:27   Log-Likelihood:
-1380.4
No. Observations:                 153   AIC:
2843.
Df Residuals:                     112   BIC:
2967.
Df Model:                          41
Covariance Type:            nonrobust
================================================================================
===========
                               coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-----------
boreratio                4449.9383    728.766      6.106      0.000    3005.982
5893.895
peakrpm                    -0.3895      0.494     -0.788      0.432      -1.369
0.590
citympg                  -131.1517     45.597     -2.876      0.005    -221.496
-40.807
CompanyName_Nissan     -1.232e-10   5.36e-11     -2.298      0.023    -2.3e-10
```

```
-1.7e-11
CompanyName_alfa-romero -3090.3824   2494.219     -1.239      0.218   -8032.357
1851.592
CompanyName_audi         8022.1881   1304.180      6.151      0.000    5438.123
1.06e+04
CompanyName_bmw          1.181e+04   1259.786      9.373      0.000    9311.727
1.43e+04
CompanyName_buick         1.61e+04   1399.595     11.503      0.000    1.33e+04
1.89e+04
CompanyName_chevrolet    1353.4051   1845.230      0.733      0.465   -2302.681
5009.491
CompanyName_dodge         568.6151   1197.723      0.475      0.636   -1804.520
2941.751
CompanyName_honda         653.7415   1387.903      0.471      0.639   -2096.210
3403.693
CompanyName_isuzu        1645.9267   1324.729      1.242      0.217    -978.854
4270.708
CompanyName_jaguar        1.36e+04   1981.317      6.862      0.000    9670.827
1.75e+04
CompanyName_maxda         -42.7009   2446.889     -0.017      0.986   -4890.899
4805.497
CompanyName_mazda        1614.1385   1011.511      1.596      0.113    -390.041
3618.318
CompanyName_mercury      3821.3398   2478.476      1.542      0.126   -1089.443
8732.122
CompanyName_mitsubishi    671.7316    962.816      0.698      0.487   -1235.966
2579.429
CompanyName_nissan       1022.2055   1019.107      1.003      0.318    -997.024
3041.435
CompanyName_plymouth     1136.8628   1345.680      0.845      0.400   -1529.430
3803.156
CompanyName_porcshce     1.764e+04   2800.673      6.299      0.000    1.21e+04
2.32e+04
CompanyName_porsche      1.364e+04   1699.045      8.030      0.000    1.03e+04
1.7e+04
CompanyName_renault      -948.9707   1805.816     -0.526      0.600   -4526.963
2629.021
CompanyName_saab         3129.3929   1394.553      2.244      0.027     366.264
5892.522
CompanyName_toyouta      2618.2329   2679.162      0.977      0.331   -2690.183
7926.649
CompanyName_vokswagen     767.1058   2503.086      0.306      0.760   -4192.438
5726.650
CompanyName_volkswagen   2153.6524   1161.543      1.854      0.066    -147.797
4455.102
CompanyName_volvo        5592.7681   1129.523      4.951      0.000    3354.763
7830.773
```

```
CompanyName_vw          -393.2985   1935.297     -0.203      0.839   -4227.842
3441.245
drivewheel_4wd           177.1520   1320.724      0.134      0.894   -2439.693
2793.997
carbody_convertible     4250.5300   1546.059      2.749      0.007    1187.212
7313.848
carbody_hardtop         1078.1247   1288.424      0.837      0.404   -1474.722
3630.971
carbody_hatchback        300.3036    731.922      0.410      0.682   -1149.906
1750.513
carbody_sedan            335.8202    673.224      0.499      0.619    -998.087
1669.727
enginetype_dohc         3097.7801   1324.884      2.338      0.021     472.692
5722.868
enginetype_dohcv        6557.0274   3687.727      1.778      0.078    -749.731
1.39e+04
enginetype_l            4357.0149   1061.142      4.106      0.000    2254.497
6459.532
enginetype_ohcf        -1260.3628    992.021     -1.271      0.207   -3225.925
705.200
enginetype_ohcv         3752.3456   1299.620      2.887      0.005    1177.315
6327.376
cylindernumber_eight   -3989.3381   2219.259     -1.798      0.075   -8386.515
407.839
cylindernumber_three    3.377e-13   3.99e-13      0.845      0.400   -4.54e-13
1.13e-12
fuelsystem_2bbl        -1632.6847    666.125     -2.451      0.016   -2952.527
-312.843
fuelsystem_4bbl          578.8754   2041.193      0.284      0.777   -3465.487
4623.238
fuelsystem_mfi           514.7135   2641.698      0.195      0.846   -4719.473
5748.900
==============================================================================
Omnibus:                       17.719   Durbin-Watson:                   2.076
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               47.098
Skew:                           0.384   Prob(JB):                     5.93e-11
Kurtosis:                       5.608   Cond. No.                     1.01e+16
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[3] The smallest eigenvalue is 3.96e-23. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
"""
```

[530]:
```
#OLS Method for Linear regression - test data

from statsmodels.regression.linear_model import OLS
import statsmodels.regression.linear_model as smf

reg_model = smf.OLS(endog = y_test, exog=x_test).fit()

reg_model.summary()
```

[530]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:                   price   R-squared (uncentered):
0.987
Model:                             OLS   Adj. R-squared (uncentered):
0.970
Method:                  Least Squares   F-statistic:
58.46
Date:                 Sun, 03 Sep 2023   Prob (F-statistic):
1.19e-15
Time:                         15:43:33   Log-Likelihood:
-454.42
No. Observations:                   52   AIC:
966.8
Df Residuals:                       23   BIC:
1023.
Df Model:                           29
Covariance Type:            nonrobust
================================================================================
===========
                               coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-----------
boreratio                 2392.7423   2026.522      1.181      0.250   -1799.437
6584.922
peakrpm                      1.5065      1.437      1.048      0.305      -1.467
4.480
citympg                   -208.0555     94.726     -2.196      0.038    -404.011
-12.100
CompanyName_Nissan      -2291.5675   2844.692     -0.806      0.429   -8176.261
3593.126
CompanyName_alfa-romero  1039.5509   4218.625      0.246      0.808   -7687.341
9766.442
CompanyName_audi        -2.112e-11   1.67e-11     -1.265      0.218   -5.56e-11
```

```
1.34e-11
CompanyName_bmw          1.237e+04   2425.642      5.100      0.000    7353.509
1.74e+04
CompanyName_buick        6951.1313   2198.671      3.162      0.004    2402.834
1.15e+04
CompanyName_chevrolet     807.7435   2275.724      0.355      0.726   -3899.949
5515.436
CompanyName_dodge        -833.7136   2574.753     -0.324      0.749   -6159.995
4492.568
CompanyName_honda       -1846.3214   2532.604     -0.729      0.473   -7085.412
3392.770
CompanyName_isuzu       -3.264e-12   1.71e-11     -0.191      0.850   -3.86e-11
3.21e-11
CompanyName_jaguar       1.217e+04   3020.389      4.029      0.001    5920.021
1.84e+04
CompanyName_maxda       -1328.0109   2769.888     -0.479      0.636   -7057.961
4401.939
CompanyName_mazda        4083.6404   1761.950      2.318      0.030     438.768
7728.512
CompanyName_mercury      3.347e-12   5.96e-12      0.562      0.580   -8.98e-12
1.57e-11
CompanyName_mitsubishi  -1241.3879   2308.555     -0.538      0.596   -6016.997
3534.222
CompanyName_nissan       -357.2964   2029.059     -0.176      0.862   -4554.725
3840.132
CompanyName_plymouth    -1229.6116   2326.960     -0.528      0.602   -6043.294
3584.071
CompanyName_porcshce    -1.255e-12      6e-13     -2.092      0.048    -2.5e-12
-1.43e-14
CompanyName_porsche     -1.189e-13   9.78e-13     -0.122      0.904   -2.14e-12
1.9e-12
CompanyName_renault      1.672e-12   6.14e-13      2.722      0.012    4.01e-13
2.94e-12
CompanyName_saab         2043.9898   2664.681      0.767      0.451   -3468.323
7556.303
CompanyName_toyouta      3.041e-12   9.32e-13      3.264      0.003    1.11e-12
4.97e-12
CompanyName_vokswagen   -6.823e-13    1.2e-12     -0.569      0.575   -3.16e-12
1.8e-12
CompanyName_volkswagen  -1425.6085   2374.738     -0.600      0.554   -6338.129
3486.912
CompanyName_volvo        5771.6455   2385.803      2.419      0.024     836.236
1.07e+04
CompanyName_vw          -2.473e-13   1.71e-13     -1.449      0.161   -6.01e-13
1.06e-13
drivewheel_4wd           1308.0235   2833.839      0.462      0.649   -4554.219
7170.266
```

```
carbody_convertible              0          0        nan        nan           0
0
carbody_hardtop          -2450.3689   3047.609     -0.804      0.430   -8754.829
3854.091
carbody_hatchback          303.7653   1751.277      0.173      0.864   -3319.027
3926.558
carbody_sedan             1190.9577   1493.489      0.797      0.433   -1898.560
4280.475
enginetype_dohc           3495.8985   2580.423      1.355      0.189   -1842.113
8833.910
enginetype_dohcv                 0          0        nan        nan           0
0
enginetype_l               581.9017   3152.317      0.185      0.855   -5939.163
7102.967
enginetype_ohcf          -1696.4676   2840.418     -0.597      0.556   -7572.320
4179.385
enginetype_ohcv           5164.9246   3328.702      1.552      0.134   -1721.020
1.21e+04
cylindernumber_eight      6951.1313   2198.671      3.162      0.004    2402.834
1.15e+04
cylindernumber_three       807.7435   2275.724      0.355      0.726   -3899.949
5515.436
fuelsystem_2bbl          -2321.3628   1128.356     -2.057      0.051   -4655.546
12.820
fuelsystem_4bbl          -6912.0113   3468.731     -1.993      0.058   -1.41e+04
263.605
fuelsystem_mfi                   0          0        nan        nan           0
0
==============================================================================
Omnibus:                        7.711   Durbin-Watson:                   1.661
Prob(Omnibus):                  0.021   Jarque-Bera (JB):                9.139
Skew:                           0.502   Prob(JB):                       0.0104
Kurtosis:                       4.792   Cond. No.                     1.01e+16
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[3] The smallest eigenvalue is 1.36e-23. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
"""
```

[531]:
```
#Normal Linear Regression

from sklearn.linear_model import LinearRegression
```

```
lm=LinearRegression()
lm.fit(x_train,y_train)

# Predict price by using lm model with test dataset

y_pred_price = lm.predict(x_test)
y_pred_price_train = lm.predict(x_train)

# Validate the actual price of the test data and predicted price

from sklearn.metrics import r2_score
r2_score(y_test, y_pred_price)
```

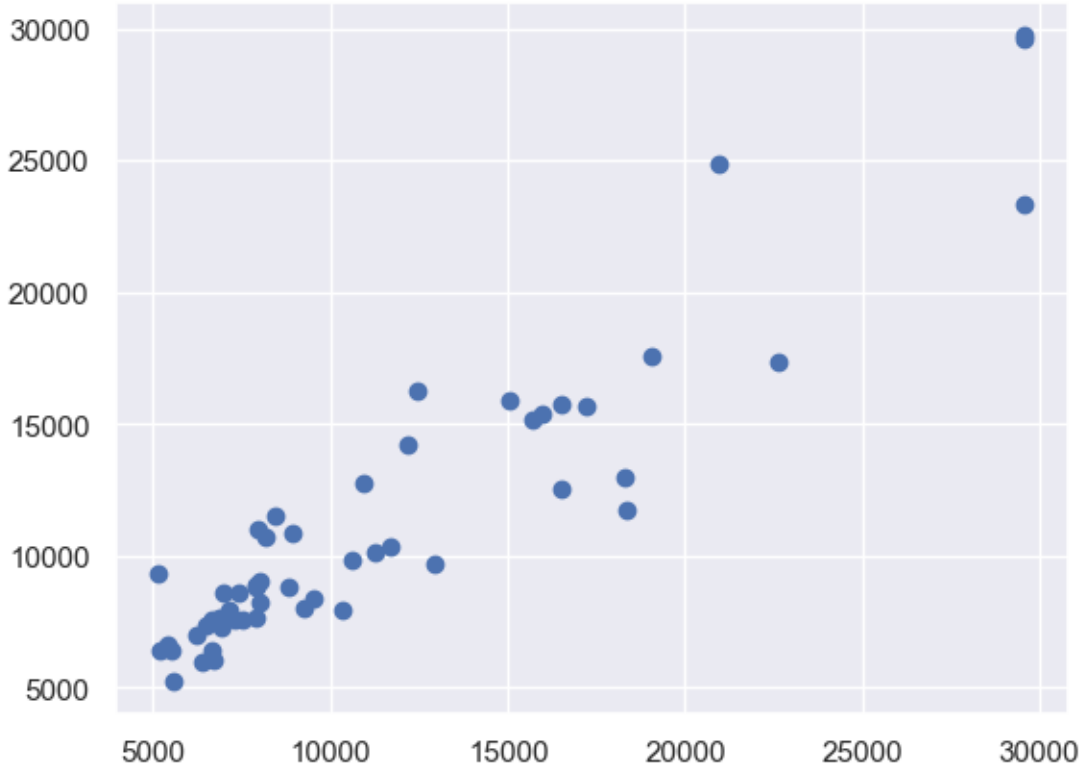
[531]: 0.8640547153572377

```
[532]: r2_score(y_train, y_pred_price_train)
```

[532]: 0.9265368189308657

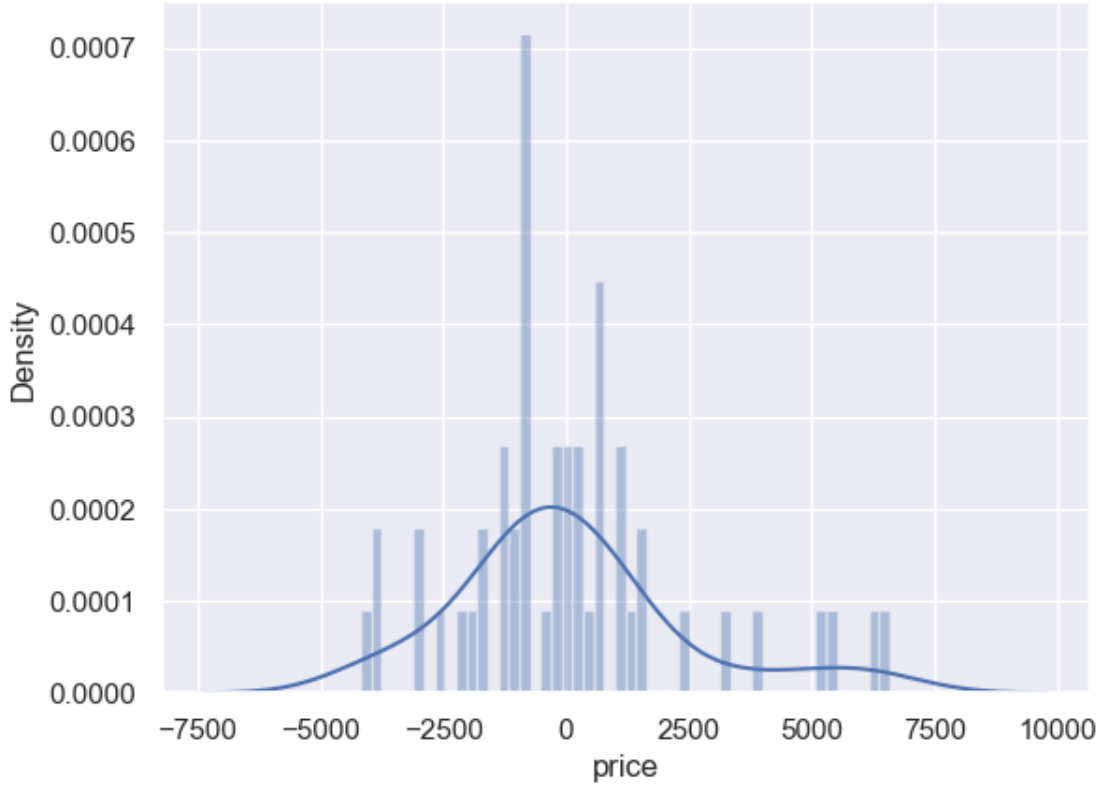
```
[533]: #Check linearity

plt.scatter(y_test, y_pred_price)
```

[533]: <matplotlib.collections.PathCollection at 0x24fd30cbfa0>



[508]:
```
reg_model = smf.OLS(endog = y_train, exog=x_train).fit()
```

[534]:
```
# Normality of Residual

sns.distplot((y_test - y_pred_price), bins=50)
plt.show()
```



[535]:
```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor()
rf.fit(x_train, y_train)
```

[535]:
```
RandomForestRegressor()
```

[536]:
```
y_pred_train = rf.predict(x_train)
y_pred_test = rf.predict(x_test)
```

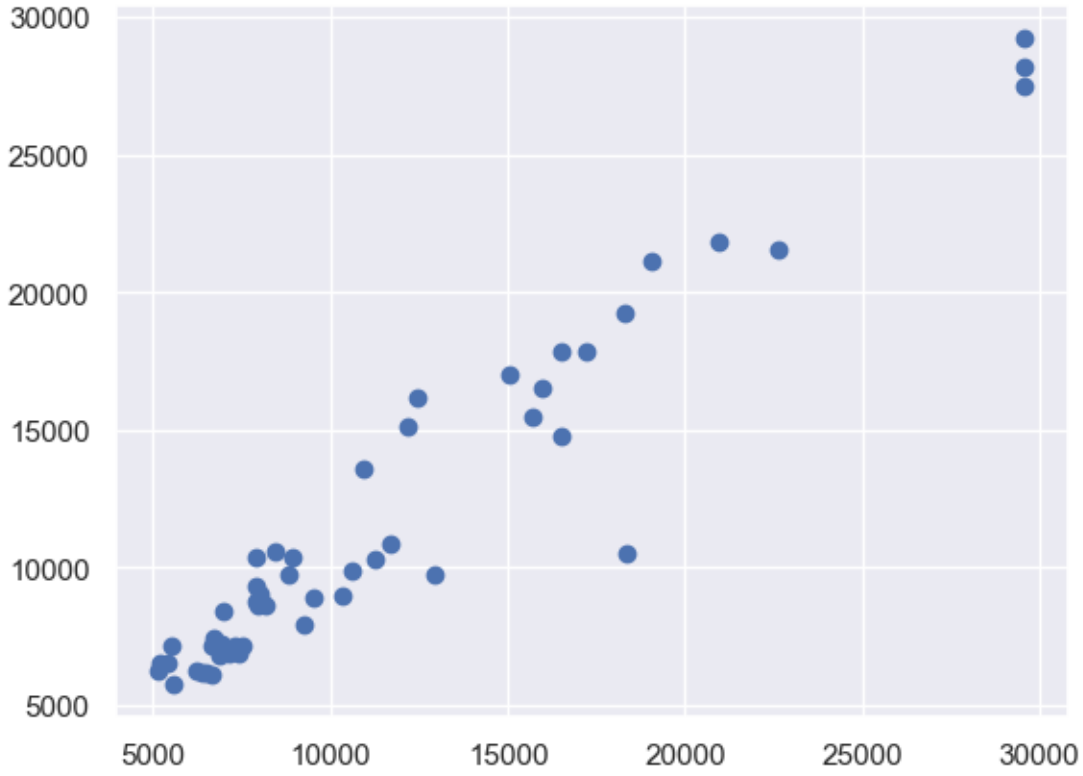
[537]:
```
from sklearn.metrics import r2_score
print(r2_score(y_train, y_pred_train))
print()
```

```
print(r2_score(y_test, y_pred_test))
```

```
0.9709609621988796
```

```
0.9225804061340487
```

[538]:
```
plt.scatter(y_test,y_pred_test)
plt.show()
```



[539]:
```
from sklearn.tree import DecisionTreeRegressor
dtree = DecisionTreeRegressor()
dtree.fit(x_train, y_train)
```

[539]:
```
DecisionTreeRegressor()
```

[540]:
```
y_pred_train = dtree.predict(x_train)
y_pred_test = dtree.predict(x_test)
```

[541]:
```
# Evaluate the model
from sklearn.metrics import r2_score
print(r2_score(y_train, y_pred_train))
print()
```

```
print(r2_score(y_test, y_pred_test))
```

```
0.9909306794205561
```

```
0.9185335479049056
```

[542]:
```
#Boosting techniques
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error
gdb = GradientBoostingRegressor()
gd = gdb.fit(x_train, y_train)
y_pred_train = gd.predict(x_train)
y_pred_test = gd.predict(x_test)


# Evaluate the model
from sklearn.metrics import r2_score
print(r2_score(y_train, y_pred_train))
print()
print(r2_score(y_test, y_pred_test))
```

```
0.973393656904805
```

```
0.9318834249340793
```

[543]:
```
from sklearn.ensemble import VotingRegressor
from sklearn.model_selection import cross_val_score
estimators=[('lm',lm),('rf',rf),('dtree',dtree),('gdb',gdb)]
vr=VotingRegressor(estimators)
scores=cross_val_score(vr,x,y,scoring='r2',cv=10)
print(np.round(np.mean(scores),2))
print(np.round(np.max(scores),2))
```

```
0.41
0.79
```

[ ]:

[ ]: