Exploracion

August 26, 2023

1 Técnicas de procesamiento de datos para el análisis estadístico y para la construcción de modelos

Mario Javier Soriano Aguilera A01384282

[ ]:
```
from google.colab import drive
drive.mount("/content/gdrive")
!pwd
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call
drive.mount("/content/gdrive", force_remount=True).
/content
```

[29]:
```
%cd "/content/gdrive/MyDrive/InteligenciaArtificial/ModuloUnoEvidencia/"
!ls #List files located in defined folder
```

```
/content/gdrive/MyDrive/InteligenciaArtificial/ModuloUnoEvidencia
Exploracion  precios_autos.csv
```

[ ]:
```

```

[ ]:
```

```

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import kurtosis, skew, shapiro
import seaborn as sns
import numpy as np
import re

sns.set()
```

[ ]:
```
df = pd.read_csv('/content/gdrive/MyDrive/InteligenciaArtificial/
↪Modulo1Evidencia/precios_autos.csv')
```

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

[ ]:
```
   symboling                   CarName fueltype      carbody drivewheel  \
0          3        alfa-romero giulia      gas  convertible        rwd
1          3       alfa-romero stelvio      gas  convertible        rwd
2          1  alfa-romero Quadrifoglio      gas    hatchback        rwd
3          2               audi 100 ls      gas        sedan        fwd
4          2                audi 100ls      gas        sedan        4wd

  enginelocation  wheelbase  carlength  carwidth  carheight  …  enginetype  \
0          front       88.6      168.8      64.1       48.8  …        dohc
1          front       88.6      168.8      64.1       48.8  …        dohc
2          front       94.5      171.2      65.5       52.4  …        ohcv
3          front       99.8      176.6      66.2       54.3  …         ohc
4          front       99.4      176.6      66.4       54.3  …         ohc

  cylindernumber enginesize  stroke  compressionratio  horsepower  peakrpm  \
0           four        130    2.68               9.0         111     5000
1           four        130    2.68               9.0         111     5000
2            six        152    3.47               9.0         154     5000
3           four        109    3.40              10.0         102     5500
4           five        136    3.40               8.0         115     5500

   citympg  highwaympg    price
0       21          27  13495.0
1       21          27  16500.0
2       19          26  16500.0
3       24          30  13950.0
4       18          22  17450.0

[5 rows x 21 columns]
```

[ ]:

##Variables cuantitativas

[ ]:
```
df.describe().T
```

[ ]:
```
                  count         mean          std      min      25%  \
symboling         205.0     0.834146     1.245307    -2.00     0.00
wheelbase         205.0    98.756585     6.021776    86.60    94.50
carlength         205.0   174.049268    12.337289   141.10   166.30
carwidth          205.0    65.907805     2.145204    60.30    64.10
carheight         205.0    53.724878     2.443522    47.80    52.00
curbweight        205.0  2555.565854   520.680204  1488.00  2145.00
enginesize        205.0   126.907317    41.642693    61.00    97.00
stroke            205.0     3.255415     0.313597     2.07     3.11
compressionratio  205.0    10.142537     3.972040     7.00     8.60
horsepower        205.0   104.117073    39.544167    48.00    70.00
```

```
peakrpm           205.0   5125.121951   476.985643  4150.00  4800.00
citympg           205.0     25.219512     6.542142    13.00    19.00
highwaympg        205.0     30.751220     6.886443    16.00    25.00
price             205.0  13276.710571  7988.852332  5118.00  7788.00

                     50%       75%       max
symboling            1.00      2.00      3.00
wheelbase           97.00    102.40    120.90
carlength          173.20    183.10    208.10
carwidth            65.50     66.90     72.30
carheight           54.10     55.50     59.80
curbweight        2414.00   2935.00   4066.00
enginesize         120.00    141.00    326.00
stroke               3.29      3.41      4.17
compressionratio     9.00      9.40     23.00
horsepower          95.00    116.00    288.00
peakrpm           5200.00   5500.00   6600.00
citympg             24.00     30.00     49.00
highwaympg          30.00     34.00     54.00
price            10295.00  16503.00  45400.00
```

[ ]:
```
moda = df.mode()
print(moda[0:1])
```

```
   symboling      CarName fueltype carbody drivewheel enginelocation  \
0        0.0  peugeot 504      gas   sedan        fwd          front

   wheelbase  carlength  carwidth  carheight  …  enginetype cylindernumber  \
0       94.5      157.3      63.8       50.8  …         ohc            four

  enginesize  stroke  compressionratio  horsepower  peakrpm  citympg  \
0       92.0     3.4               9.0        68.0   5500.0     31.0

   highwaympg   price
0        25.0  5572.0

[1 rows x 21 columns]
```

[ ]:
```
df.var()
```

```
<ipython-input-8-28ded241fd7c>:1: FutureWarning: The default value of
numeric_only in DataFrame.var is deprecated. In a future version, it will
default to False. In addition, specifying 'numeric_only=None' is deprecated.
Select only valid columns or specify the value of numeric_only to silence this
warning.
  df.var()
```

[ ]:
```
symboling           1.550789e+00
wheelbase           3.626178e+01
carlength           1.522087e+02
carwidth            4.601900e+00
carheight           5.970800e+00
curbweight          2.711079e+05
enginesize          1.734114e+03
stroke              9.834309e-02
compressionratio    1.577710e+01
horsepower          1.563741e+03
peakrpm             2.275153e+05
citympg             4.279962e+01
highwaympg          4.742310e+01
price               6.382176e+07
dtype: float64
```

[ ]:
```
df.skew()
```

```
<ipython-input-9-9e0b1e29546f>:1: FutureWarning: The default value of
numeric_only in DataFrame.skew is deprecated. In a future version, it will
default to False. In addition, specifying 'numeric_only=None' is deprecated.
Select only valid columns or specify the value of numeric_only to silence this
warning.
  df.skew()
```

[ ]:
```
symboling           0.211072
wheelbase           1.050214
carlength           0.155954
carwidth            0.904003
carheight           0.063123
curbweight          0.681398
enginesize          1.947655
stroke             -0.689705
compressionratio    2.610862
horsepower          1.405310
peakrpm             0.075159
citympg             0.663704
highwaympg          0.539997
price               1.777678
dtype: float64
```

[ ]:
```
df.kurtosis()
```

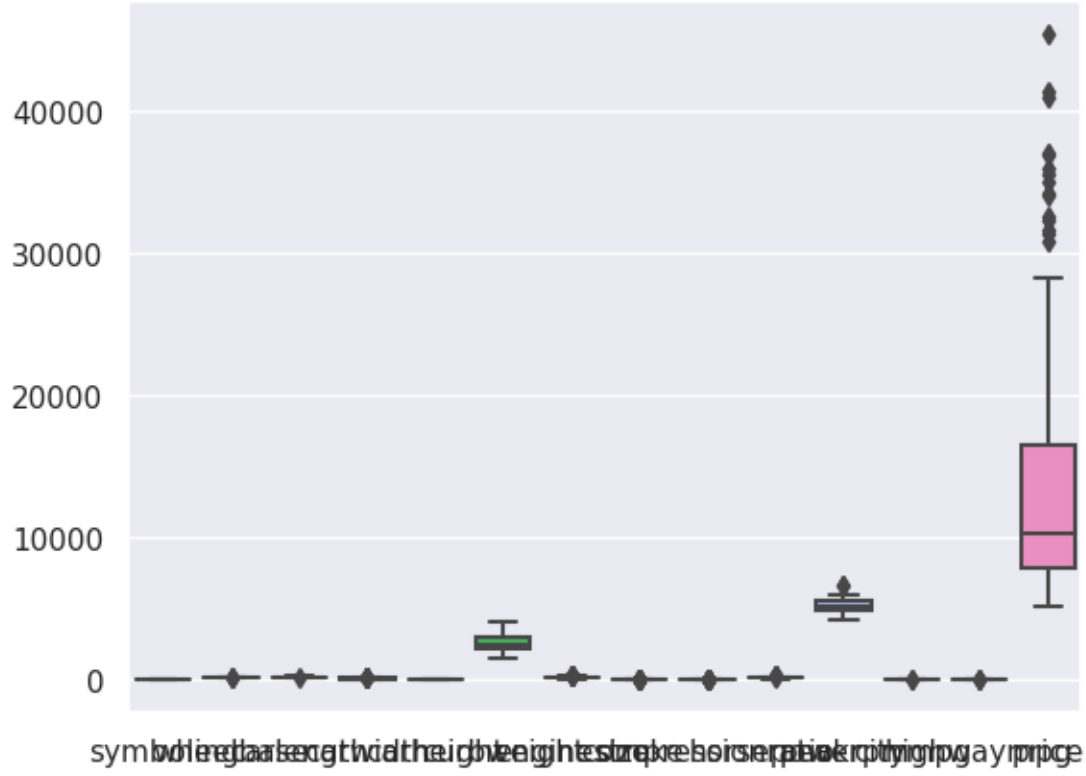
```
<ipython-input-10-c7edf97eb14c>:1: FutureWarning: The default value of
numeric_only in DataFrame.kurt is deprecated. In a future version, it will
default to False. In addition, specifying 'numeric_only=None' is deprecated.
Select only valid columns or specify the value of numeric_only to silence this
warning.
```

```
df.kurtosis()
```

```
symboling          -0.676271
wheelbase           1.017039
carlength          -0.082895
carwidth            0.702764
carheight          -0.443812
curbweight         -0.042854
enginesize          5.305682
stroke              2.174396
compressionratio    5.233054
horsepower          2.684006
peakrpm             0.086756
citympg             0.578648
highwaympg          0.440070
price               3.051648
dtype: float64
```

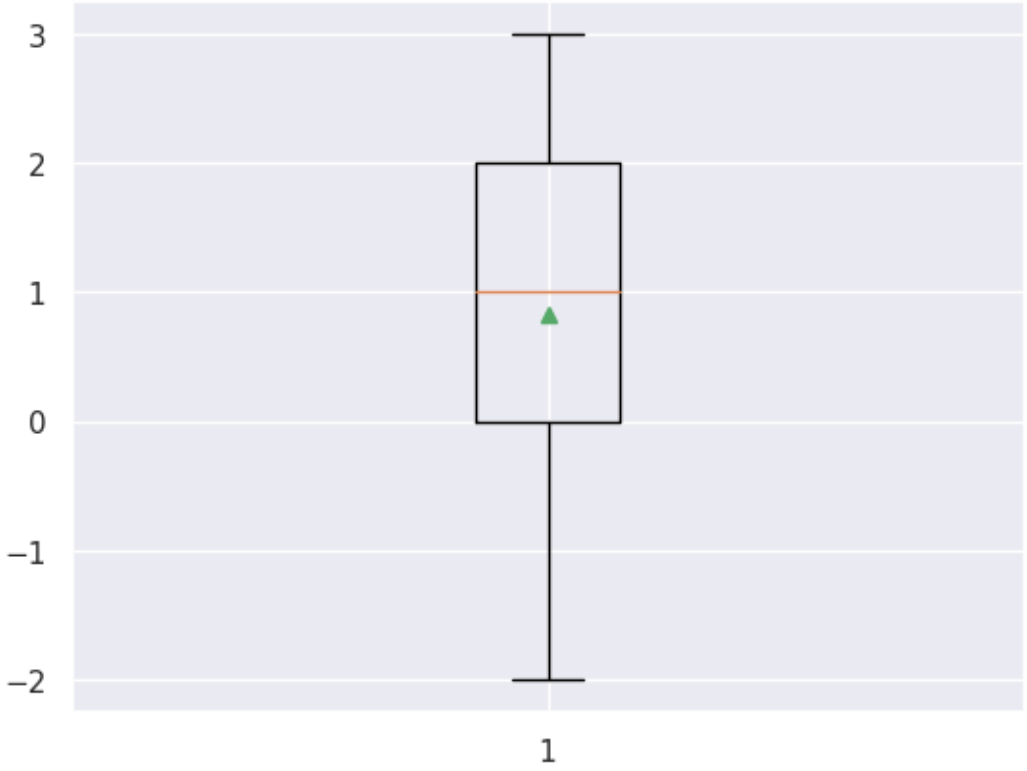
```
sns.boxplot(df)
```

```
<Axes: >
```



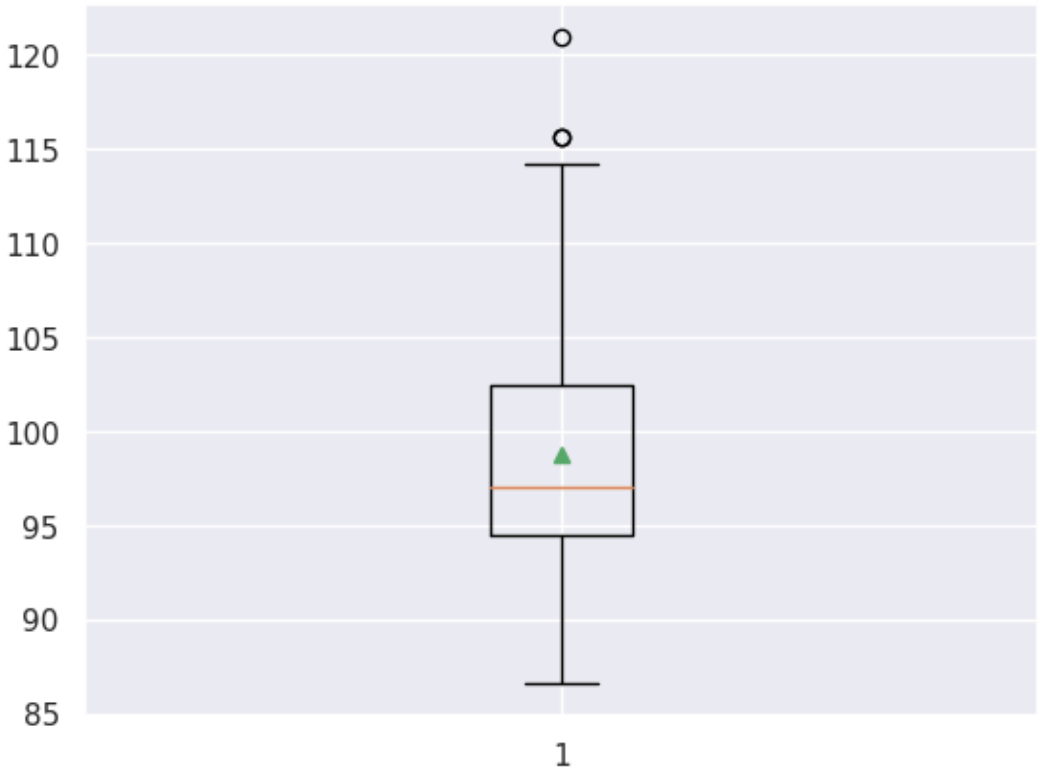
[ ]:
```
plt.boxplot(df.symboling, showmeans=True)
```

[ ]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7b1184c0c490>,
  <matplotlib.lines.Line2D at 0x7b1184c0c610>],
 'caps': [<matplotlib.lines.Line2D at 0x7b1184c0c8b0>,
  <matplotlib.lines.Line2D at 0x7b1184c0cb50>],
 'boxes': [<matplotlib.lines.Line2D at 0x7b1184c0c190>],
 'medians': [<matplotlib.lines.Line2D at 0x7b1184c0cdf0>],
 'fliers': [<matplotlib.lines.Line2D at 0x7b1184c0d330>],
 'means': [<matplotlib.lines.Line2D at 0x7b1184c0d090>]}
```



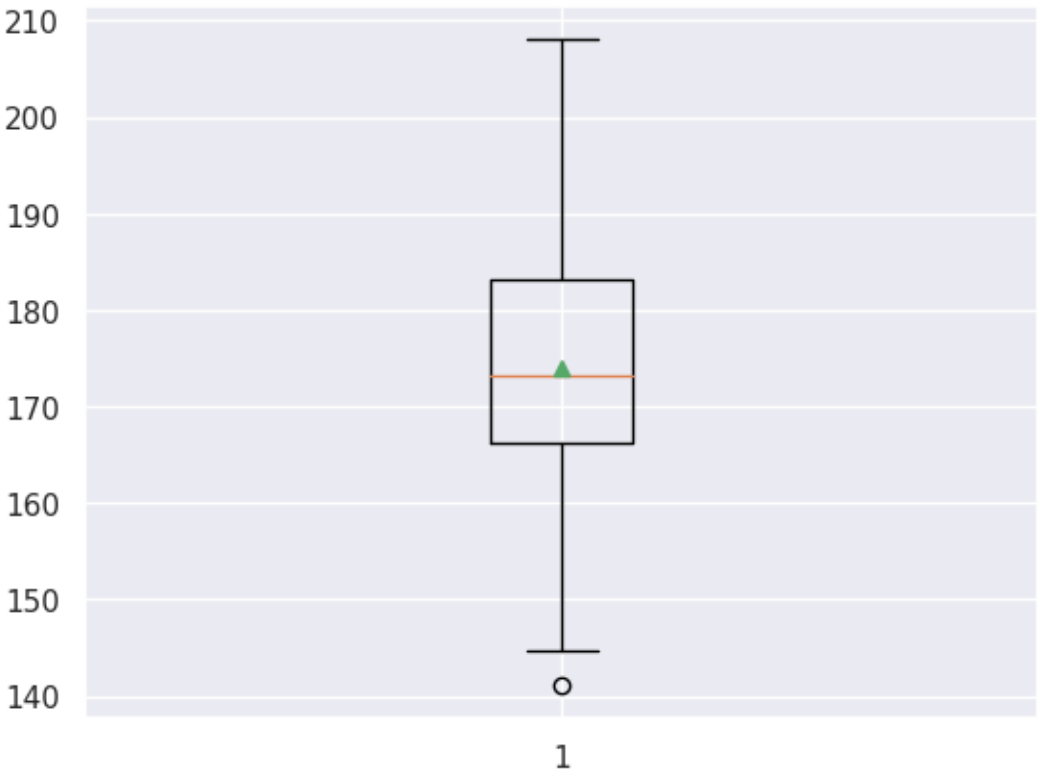
[ ]:
```
plt.boxplot(df.wheelbase, showmeans=True)
```

[ ]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7b1184c57580>,
  <matplotlib.lines.Line2D at 0x7b1184c57820>],
 'caps': [<matplotlib.lines.Line2D at 0x7b1184c57ac0>,
  <matplotlib.lines.Line2D at 0x7b1184c57d60>],
 'boxes': [<matplotlib.lines.Line2D at 0x7b1184c57400>],
 'medians': [<matplotlib.lines.Line2D at 0x7b1184c98040>],
 'fliers': [<matplotlib.lines.Line2D at 0x7b1184c98580>],
 'means': [<matplotlib.lines.Line2D at 0x7b1184c982e0>]}
```



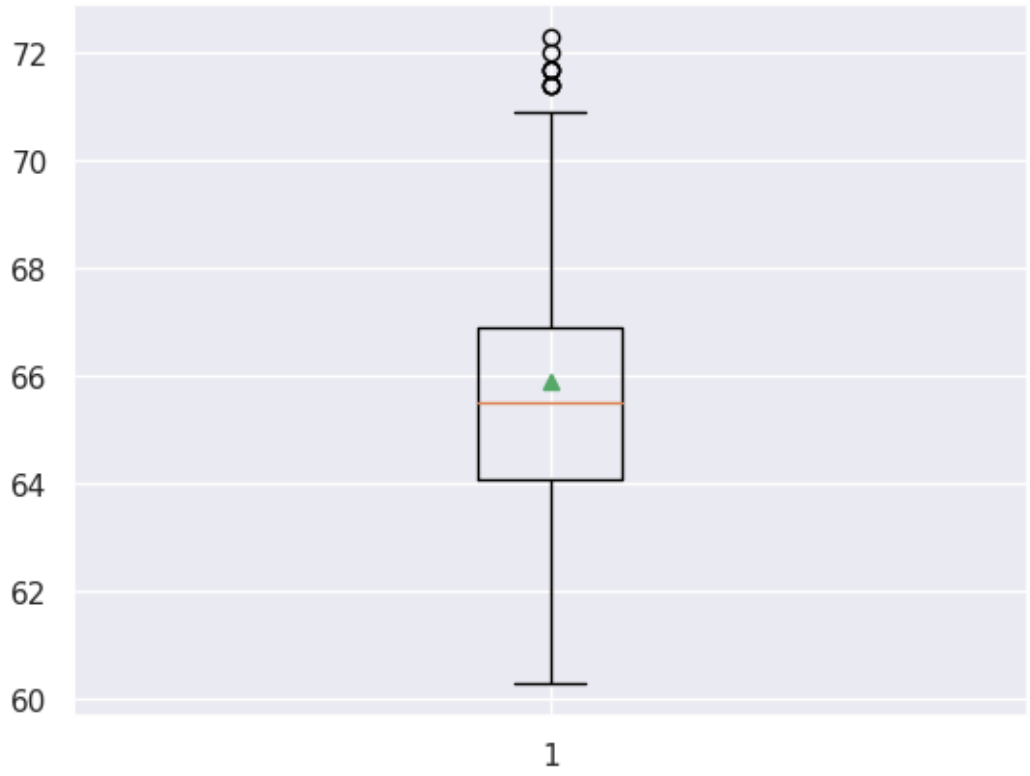
[ ]:
```
plt.boxplot(df.carlength, showmeans=True)
```

[ ]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7b1184cd7670>,
  <matplotlib.lines.Line2D at 0x7b1184cd7910>],
 'caps': [<matplotlib.lines.Line2D at 0x7b1184cd7bb0>,
  <matplotlib.lines.Line2D at 0x7b1184cd7e50>],
 'boxes': [<matplotlib.lines.Line2D at 0x7b1184cd73d0>],
 'medians': [<matplotlib.lines.Line2D at 0x7b1184b0c130>],
 'fliers': [<matplotlib.lines.Line2D at 0x7b1184b0c670>],
 'means': [<matplotlib.lines.Line2D at 0x7b1184b0c3d0>]}
```



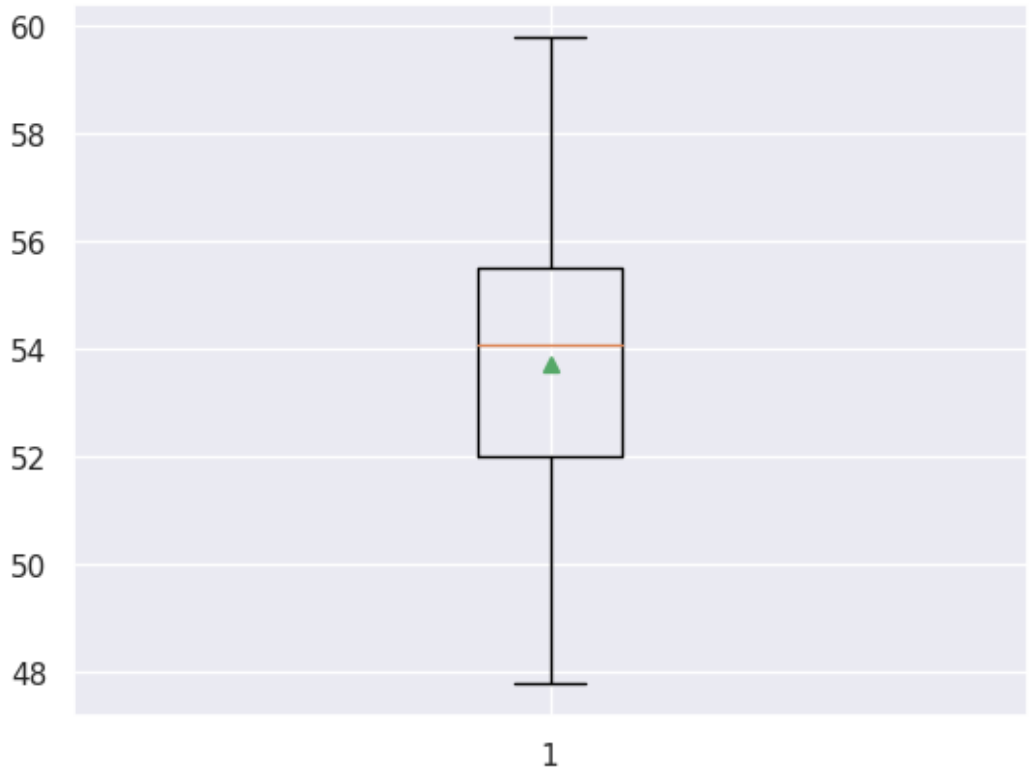
[ ]:
```
plt.boxplot(df.carwidth, showmeans=True)
```

[ ]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7b1184b47730>,
  <matplotlib.lines.Line2D at 0x7b1184b47a60>],
 'caps': [<matplotlib.lines.Line2D at 0x7b1184b47be0>,
  <matplotlib.lines.Line2D at 0x7b1184b47e80>],
 'boxes': [<matplotlib.lines.Line2D at 0x7b1184b47490>],
 'medians': [<matplotlib.lines.Line2D at 0x7b1184b7c160>],
 'fliers': [<matplotlib.lines.Line2D at 0x7b1184b7c640>],
 'means': [<matplotlib.lines.Line2D at 0x7b1184b7c3a0>]}
```



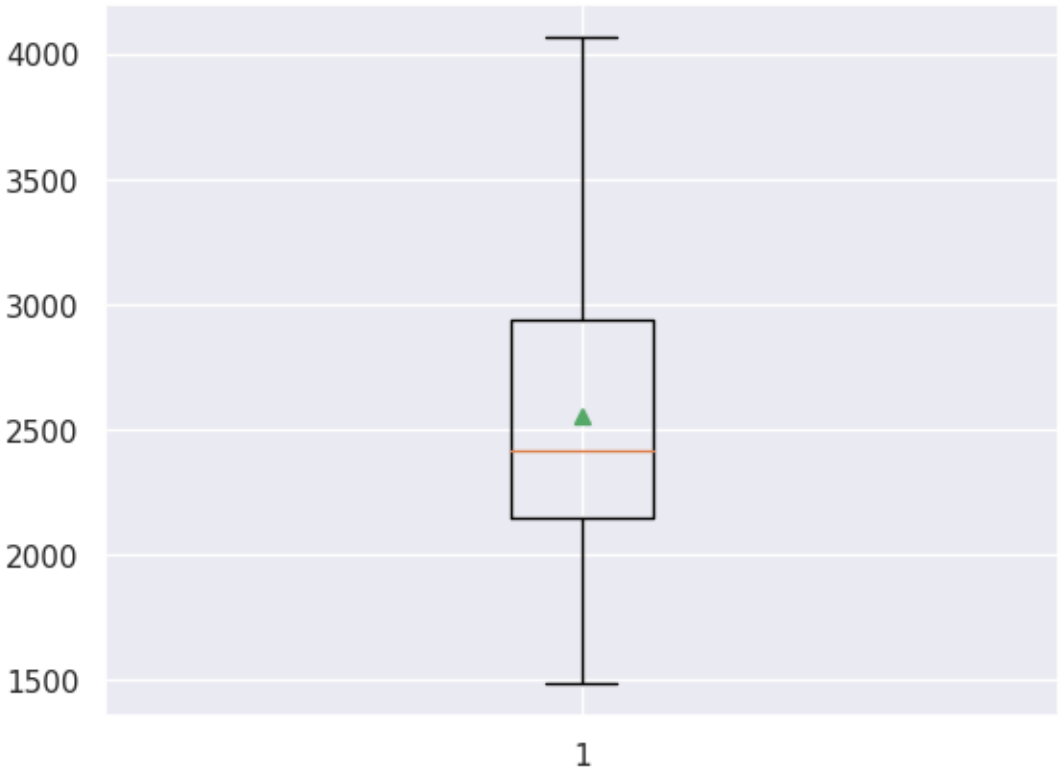
[ ]:
```
plt.boxplot(df.carheight, showmeans=True)
```

[ ]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7b1184bbf160>,
  <matplotlib.lines.Line2D at 0x7b1184bbf2e0>],
 'caps': [<matplotlib.lines.Line2D at 0x7b1184bbf580>,
  <matplotlib.lines.Line2D at 0x7b1184bbf820>],
 'boxes': [<matplotlib.lines.Line2D at 0x7b1184bbeec0>],
 'medians': [<matplotlib.lines.Line2D at 0x7b1184bbfac0>],
 'fliers': [<matplotlib.lines.Line2D at 0x7b11849fc040>],
 'means': [<matplotlib.lines.Line2D at 0x7b1184bbfd60>]}
```



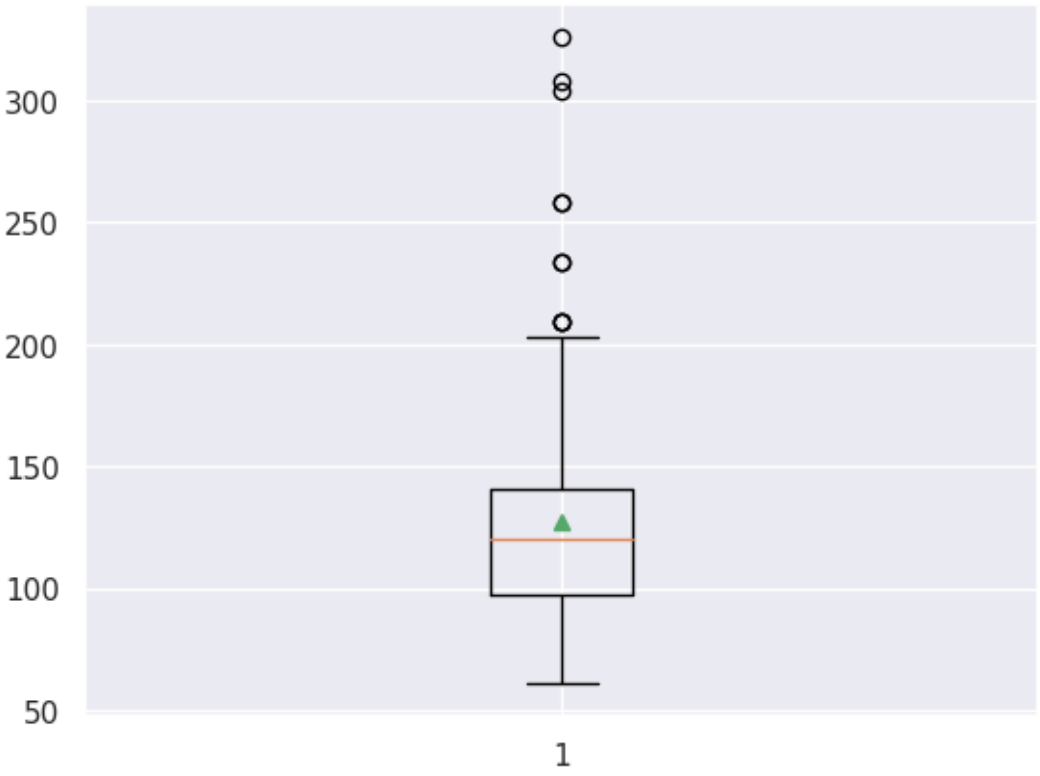
[ ]:
```
plt.boxplot(df.curbweight, showmeans=True)
```

[ ]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7b1184a423e0>,
  <matplotlib.lines.Line2D at 0x7b1184a42560>],
 'caps': [<matplotlib.lines.Line2D at 0x7b1184a42800>,
  <matplotlib.lines.Line2D at 0x7b1184a42aa0>],
 'boxes': [<matplotlib.lines.Line2D at 0x7b1184a42140>],
 'medians': [<matplotlib.lines.Line2D at 0x7b1184a42d40>],
 'fliers': [<matplotlib.lines.Line2D at 0x7b1184a43280>],
 'means': [<matplotlib.lines.Line2D at 0x7b1184a42fe0>]}
```



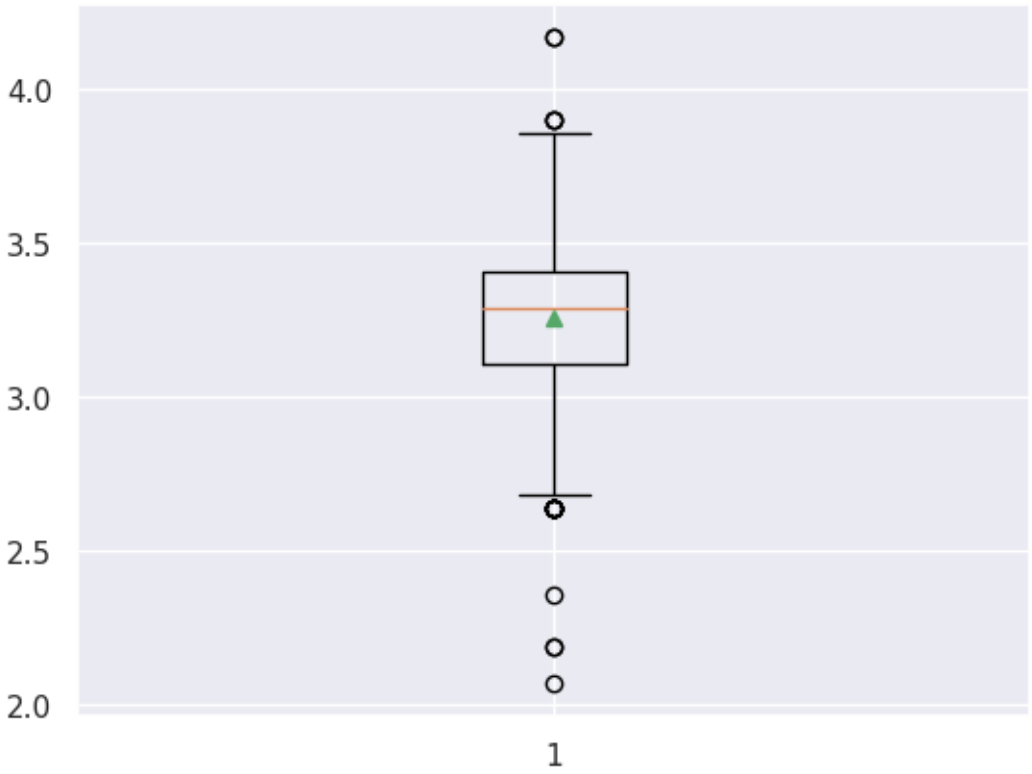
[ ]:
```
plt.boxplot(df.enginesize, showmeans=True)
```

[ ]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7b1184ac14b0>,
  <matplotlib.lines.Line2D at 0x7b1184ac1750>],
 'caps': [<matplotlib.lines.Line2D at 0x7b1184ac1960>,
  <matplotlib.lines.Line2D at 0x7b1184ac1c00>],
 'boxes': [<matplotlib.lines.Line2D at 0x7b1184ac1210>],
 'medians': [<matplotlib.lines.Line2D at 0x7b1184ac1ea0>],
 'fliers': [<matplotlib.lines.Line2D at 0x7b1184ac23e0>],
 'means': [<matplotlib.lines.Line2D at 0x7b1184ac2140>]}
```



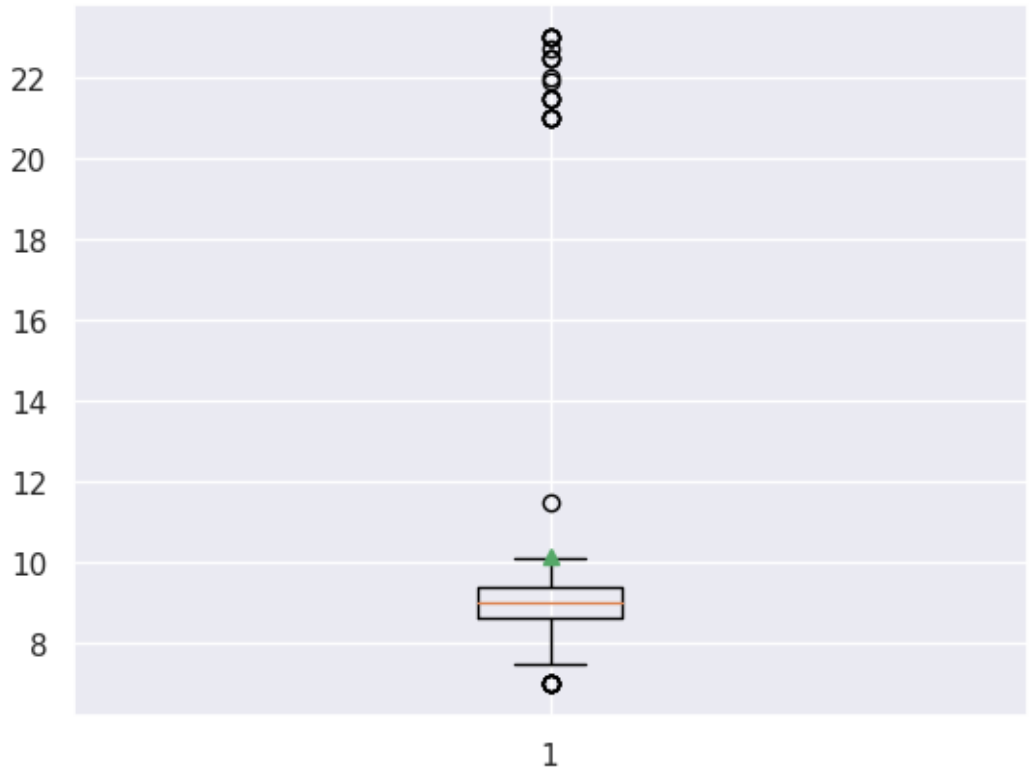
[ ]:
```
plt.boxplot(df.stroke, showmeans=True)
```

[ ]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7b1184950190>,
  <matplotlib.lines.Line2D at 0x7b1184950310>],
 'caps': [<matplotlib.lines.Line2D at 0x7b11849505b0>,
  <matplotlib.lines.Line2D at 0x7b1184950850>],
 'boxes': [<matplotlib.lines.Line2D at 0x7b1184927eb0>],
 'medians': [<matplotlib.lines.Line2D at 0x7b1184950af0>],
 'fliers': [<matplotlib.lines.Line2D at 0x7b1184951060>],
 'means': [<matplotlib.lines.Line2D at 0x7b1184950d90>]}
```



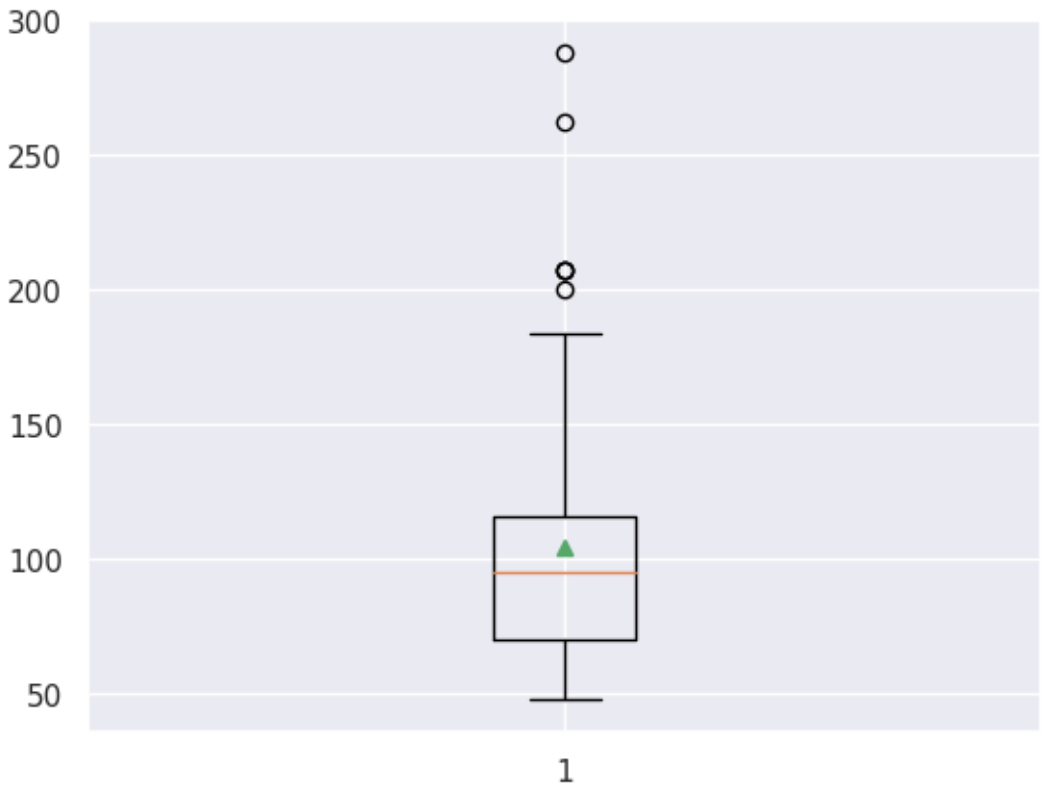
[ ]:
```
plt.boxplot(df.compressionratio, showmeans=True)
```

[ ]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7b11849965c0>,
  <matplotlib.lines.Line2D at 0x7b1184996860>],
 'caps': [<matplotlib.lines.Line2D at 0x7b1184996b00>,
  <matplotlib.lines.Line2D at 0x7b1184996da0>],
 'boxes': [<matplotlib.lines.Line2D at 0x7b1184996320>],
 'medians': [<matplotlib.lines.Line2D at 0x7b1184997040>],
 'fliers': [<matplotlib.lines.Line2D at 0x7b1184997580>],
 'means': [<matplotlib.lines.Line2D at 0x7b11849972e0>]}
```



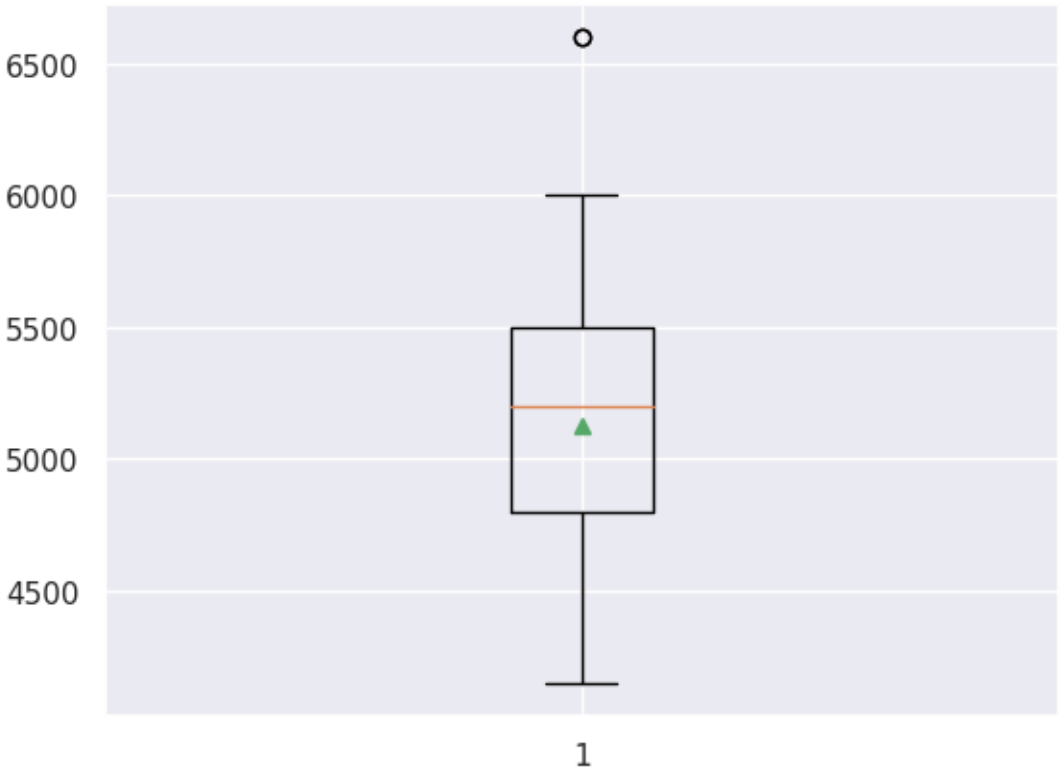
[ ]:
```
plt.boxplot(df.horsepower, showmeans=True)
```

[ ]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7b11848124d0>,
  <matplotlib.lines.Line2D at 0x7b1184812770>],
 'caps': [<matplotlib.lines.Line2D at 0x7b1184810340>,
  <matplotlib.lines.Line2D at 0x7b1184812b90>],
 'boxes': [<matplotlib.lines.Line2D at 0x7b1184812350>],
 'medians': [<matplotlib.lines.Line2D at 0x7b1184812e30>],
 'fliers': [<matplotlib.lines.Line2D at 0x7b1184813370>],
 'means': [<matplotlib.lines.Line2D at 0x7b11848130d0>]}
```



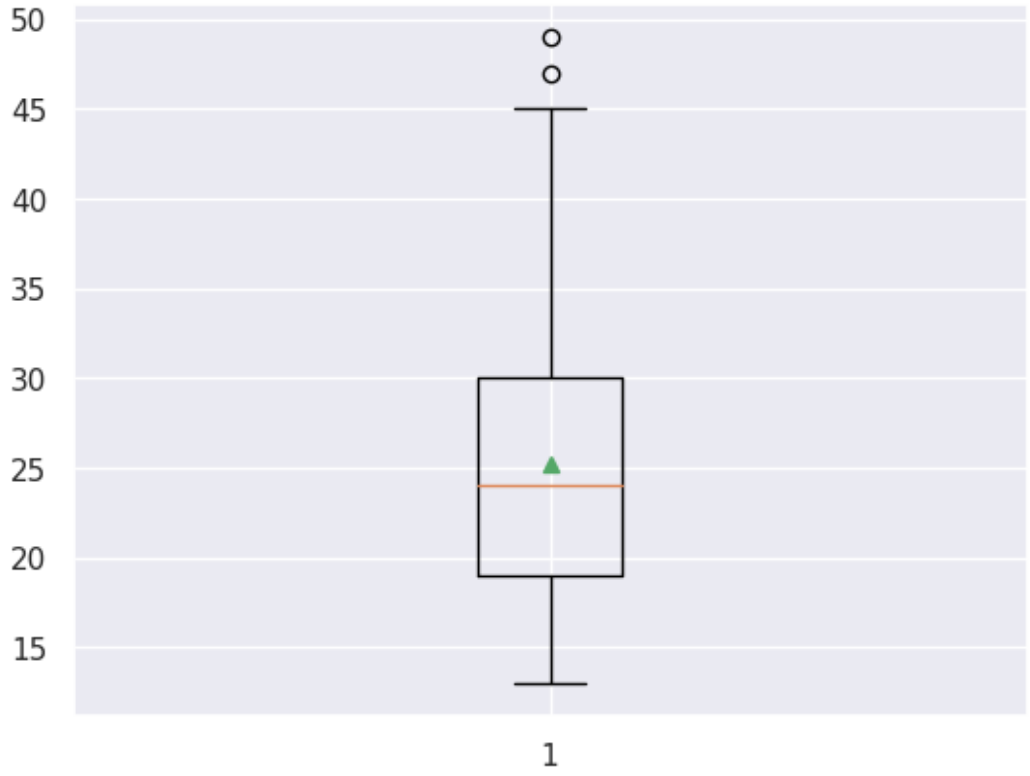
[ ]:
```
plt.boxplot(df.peakrpm, showmeans=True)
```

[ ]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7b118487fc10>,
  <matplotlib.lines.Line2D at 0x7b118487fd60>],
 'caps': [<matplotlib.lines.Line2D at 0x7b11848b4070>,
  <matplotlib.lines.Line2D at 0x7b11848b4310>],
 'boxes': [<matplotlib.lines.Line2D at 0x7b118487fa60>],
 'medians': [<matplotlib.lines.Line2D at 0x7b11848b45b0>],
 'fliers': [<matplotlib.lines.Line2D at 0x7b11848b4af0>],
 'means': [<matplotlib.lines.Line2D at 0x7b11848b4850>]}
```



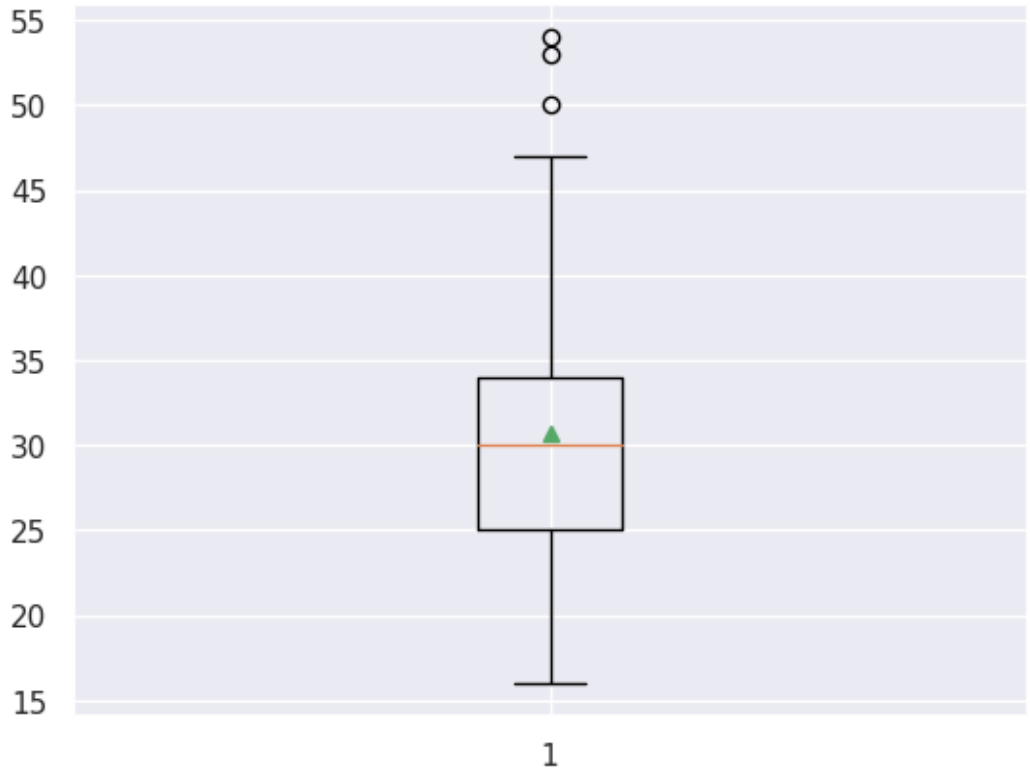
[ ]:
```
plt.boxplot(df.citympg, showmeans=True)
```

[ ]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7b11846f9cc0>,
  <matplotlib.lines.Line2D at 0x7b11846f9f60>],
 'caps': [<matplotlib.lines.Line2D at 0x7b11846fa200>,
  <matplotlib.lines.Line2D at 0x7b11846fa4a0>],
 'boxes': [<matplotlib.lines.Line2D at 0x7b11846f9a20>],
 'medians': [<matplotlib.lines.Line2D at 0x7b11846fa740>],
 'fliers': [<matplotlib.lines.Line2D at 0x7b11846fac80>],
 'means': [<matplotlib.lines.Line2D at 0x7b11846fa9e0>]}
```



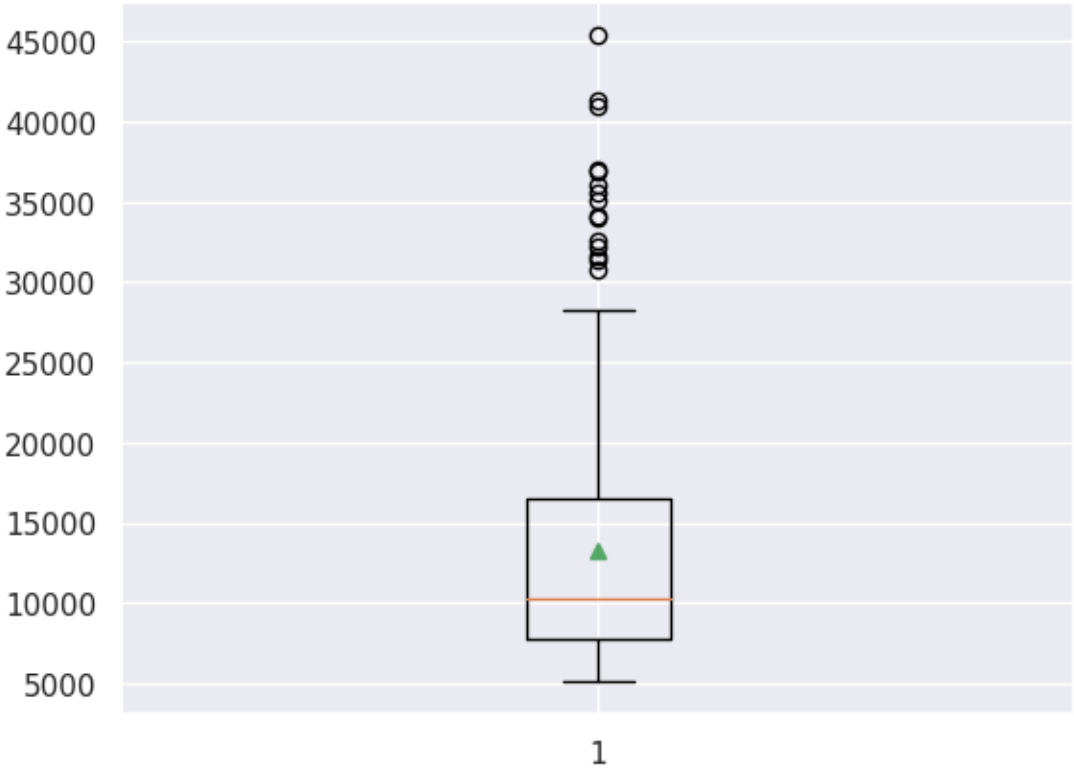
[ ]:
```
plt.boxplot(df.highwaympg, showmeans=True)
```

[ ]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7b11847761a0>,
  <matplotlib.lines.Line2D at 0x7b1184776440>],
 'caps': [<matplotlib.lines.Line2D at 0x7b1184776650>,
  <matplotlib.lines.Line2D at 0x7b11847768f0>],
 'boxes': [<matplotlib.lines.Line2D at 0x7b1184775f00>],
 'medians': [<matplotlib.lines.Line2D at 0x7b1184776b90>],
 'fliers': [<matplotlib.lines.Line2D at 0x7b11847770d0>],
 'means': [<matplotlib.lines.Line2D at 0x7b1184776e30>]}
```



[ ]:
```
plt.boxplot(df.price, showmeans=True)
```

[ ]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7b11845f6950>,
  <matplotlib.lines.Line2D at 0x7b11845f6bf0>],
 'caps': [<matplotlib.lines.Line2D at 0x7b11845f6e90>,
  <matplotlib.lines.Line2D at 0x7b11845f7130>],
 'boxes': [<matplotlib.lines.Line2D at 0x7b11845f66b0>],
 'medians': [<matplotlib.lines.Line2D at 0x7b11845f73d0>],
 'fliers': [<matplotlib.lines.Line2D at 0x7b11845f7910>],
 'means': [<matplotlib.lines.Line2D at 0x7b11845f7670>]}
```



[ ]:
```
sns.set(style="ticks", color_codes=True)
sns.pairplot(df,vars = [ 'symboling','wheelbase','carlength', 'carwidth',␣
 ↪'carheight','curbweight', 'enginesize', 'stroke', 'compressionratio',␣
 ↪'horsepower', 'peakrpm', 'citympg', 'highwaympg', 'price'], kind = "reg")
plt.show()
```

Output hidden; open in https://colab.research.google.com to view.

[ ]:
```
Tcorrelation = df.corr(method='pearson')
Tcorrelation
```

```
<ipython-input-27-6004b698a5e9>:1: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  Tcorrelation = df.corr(method='pearson')
```

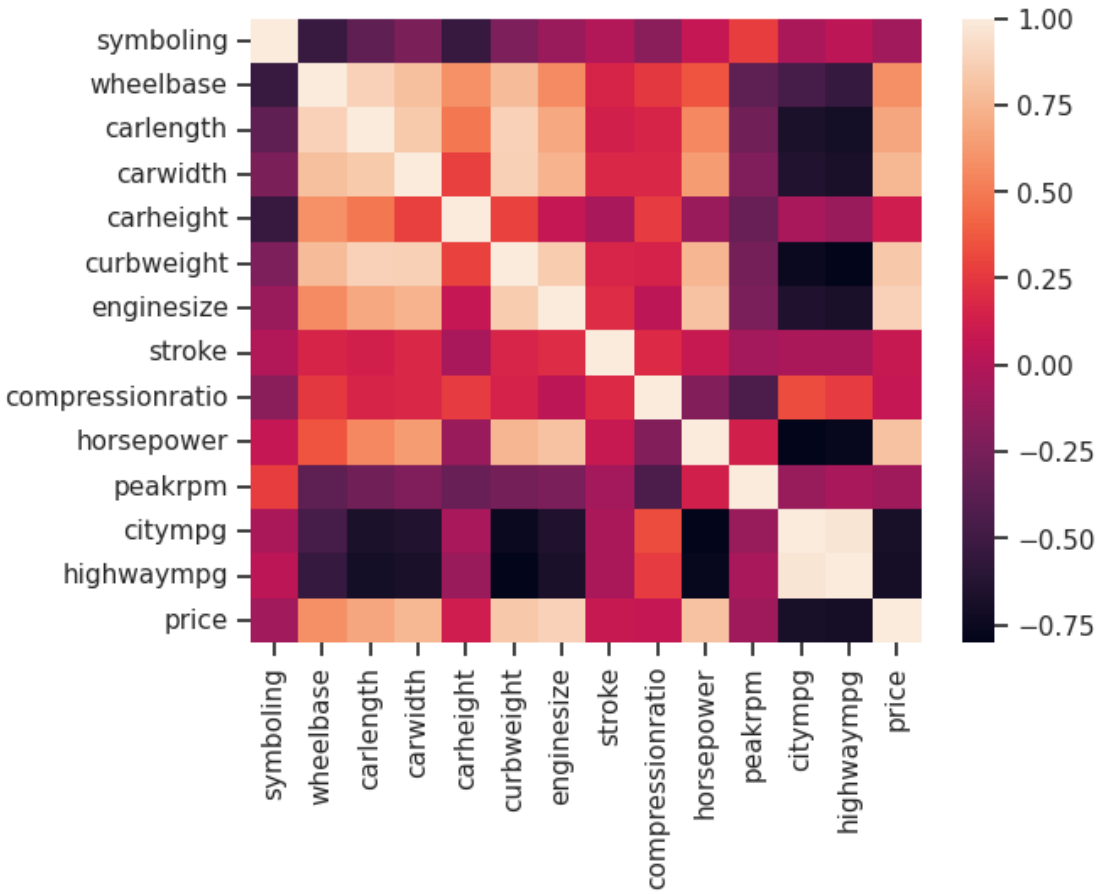
[ ]:

	symboling	wheelbase	carlength	carwidth	carheight	\
symboling	1.000000	-0.531954	-0.357612	-0.232919	-0.541038	
wheelbase	-0.531954	1.000000	0.874587	0.795144	0.589435	
carlength	-0.357612	0.874587	1.000000	0.841118	0.491029	
carwidth	-0.232919	0.795144	0.841118	1.000000	0.279210	

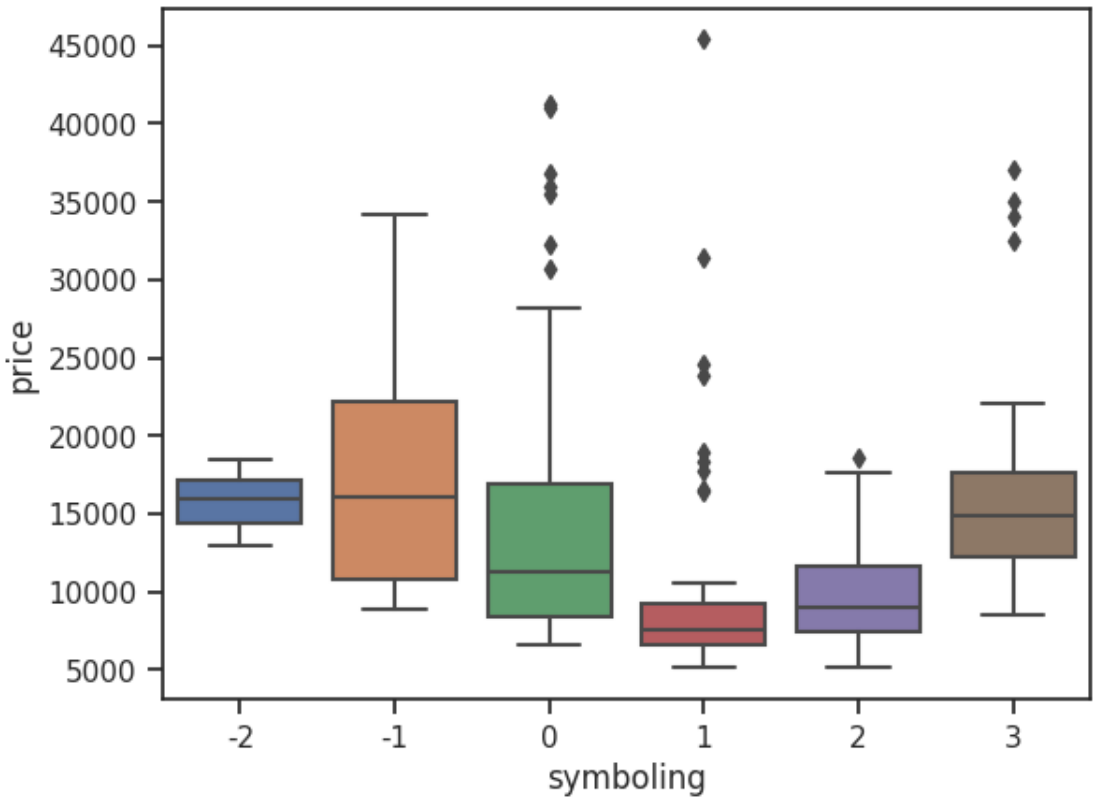
```
carheight          -0.541038   0.589435   0.491029  0.279210   1.000000
curbweight         -0.227691   0.776386   0.877728  0.867032   0.295572
enginesize         -0.105790   0.569329   0.683360  0.735433   0.067149
stroke             -0.008735   0.160959   0.129533  0.182942  -0.055307
compressionratio   -0.178515   0.249786   0.158414  0.181129   0.261214
horsepower          0.070873   0.353294   0.552623  0.640732  -0.108802
peakrpm             0.273606  -0.360469  -0.287242 -0.220012  -0.320411
citympg            -0.035823  -0.470414  -0.670909 -0.642704  -0.048640
highwaympg          0.034606  -0.544082  -0.704662 -0.677218  -0.107358
price              -0.079978   0.577816   0.682920  0.759325   0.119336

                   curbweight  enginesize    stroke  compressionratio  \
symboling          -0.227691   -0.105790 -0.008735         -0.178515
wheelbase           0.776386    0.569329  0.160959          0.249786
carlength           0.877728    0.683360  0.129533          0.158414
carwidth            0.867032    0.735433  0.182942          0.181129
carheight           0.295572    0.067149 -0.055307          0.261214
curbweight          1.000000    0.850594  0.168790          0.151362
enginesize          0.850594    1.000000  0.203129          0.028971
stroke              0.168790    0.203129  1.000000          0.186110
compressionratio    0.151362    0.028971  0.186110          1.000000
horsepower          0.750739    0.809769  0.080940         -0.204326
peakrpm            -0.266243   -0.244660 -0.067964         -0.435741
citympg            -0.757414   -0.653658 -0.042145          0.324701
highwaympg         -0.797465   -0.677470 -0.043931          0.265201
price               0.835305    0.874145  0.079443          0.067984

                   horsepower   peakrpm   citympg  highwaympg     price
symboling            0.070873  0.273606 -0.035823    0.034606 -0.079978
wheelbase            0.353294 -0.360469 -0.470414   -0.544082  0.577816
carlength            0.552623 -0.287242 -0.670909   -0.704662  0.682920
carwidth             0.640732 -0.220012 -0.642704   -0.677218  0.759325
carheight           -0.108802 -0.320411 -0.048640   -0.107358  0.119336
curbweight           0.750739 -0.266243 -0.757414   -0.797465  0.835305
enginesize           0.809769 -0.244660 -0.653658   -0.677470  0.874145
stroke               0.080940 -0.067964 -0.042145   -0.043931  0.079443
compressionratio    -0.204326 -0.435741  0.324701    0.265201  0.067984
horsepower           1.000000  0.131073 -0.801456   -0.770544  0.808139
peakrpm              0.131073  1.000000 -0.113544   -0.054275 -0.085267
citympg             -0.801456 -0.113544  1.000000    0.971337 -0.685751
highwaympg          -0.770544 -0.054275  0.971337    1.000000 -0.697599
price                0.808139 -0.085267 -0.685751   -0.697599  1.000000
```

[ ]:
```
sns.heatmap(Tcorrelation,xticklabels=Tcorrelation.columns,
 yticklabels=Tcorrelation.columns)
```

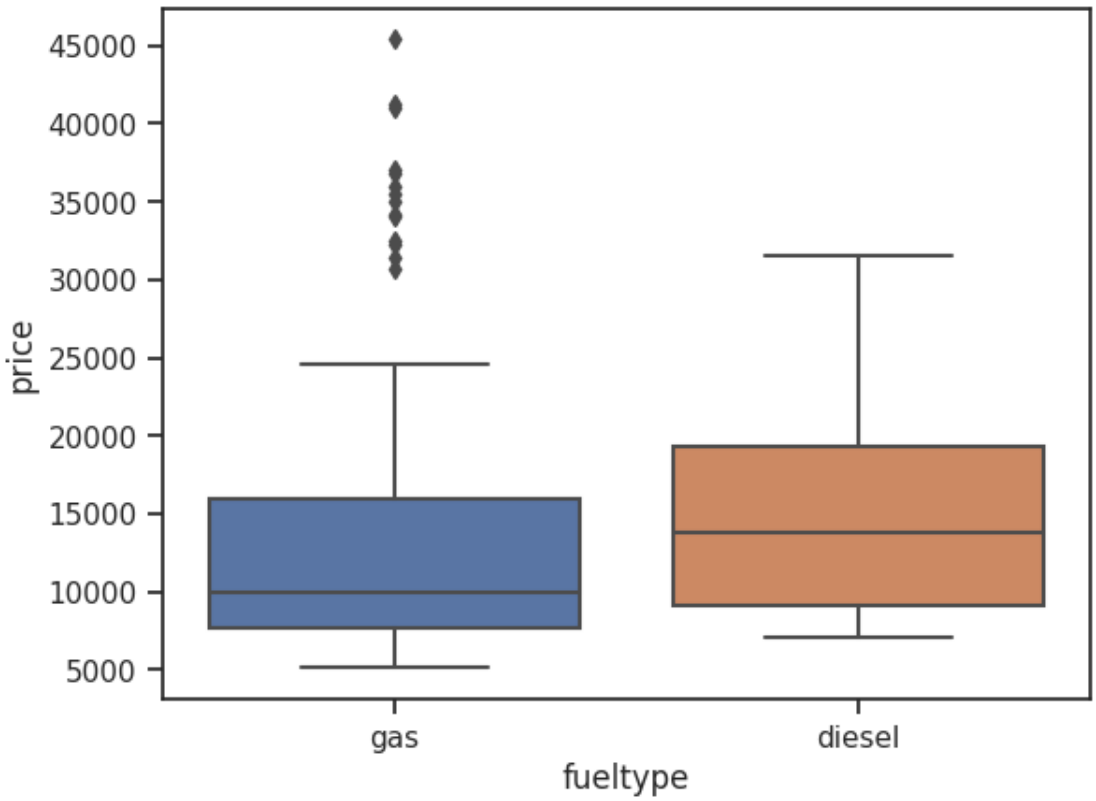
[ ]:
```
<Axes: >
```



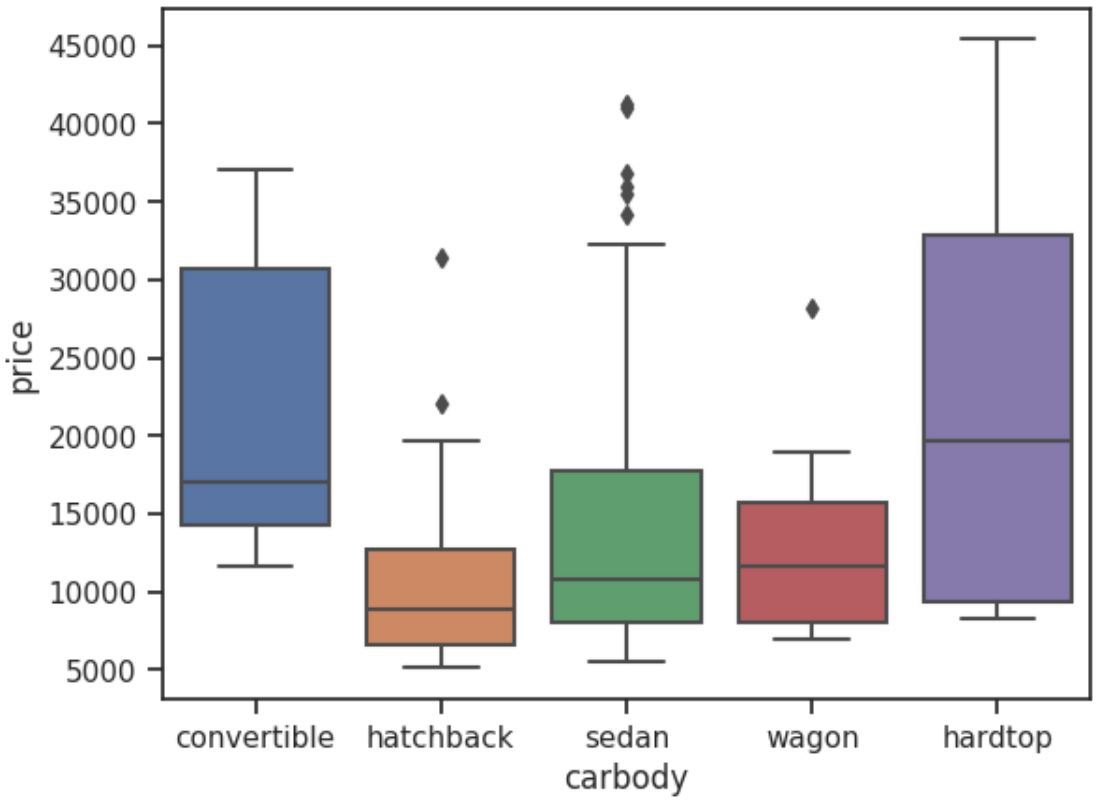
[ ]:
```
#boxplot
sns.boxplot(x="symboling",y="price",data=df)
plt.show()
```



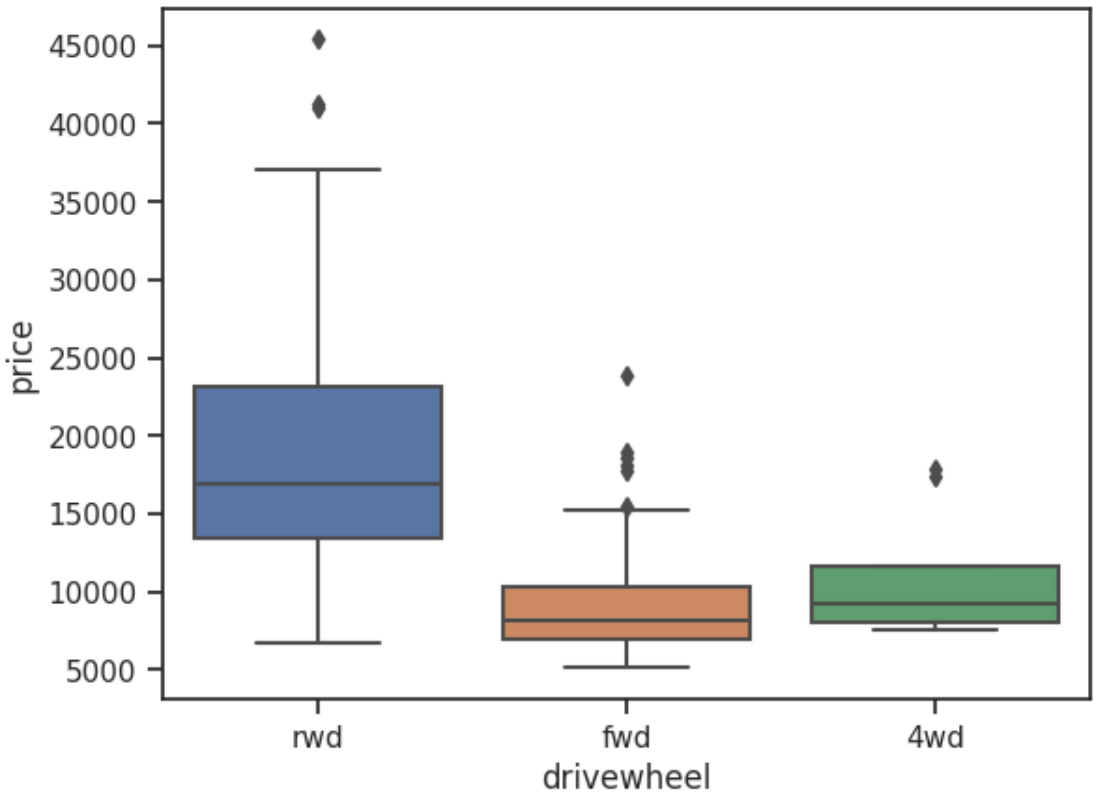
[ ]:
```
sns.boxplot(x="fueltype",y="price",data=df)
plt.show()
```



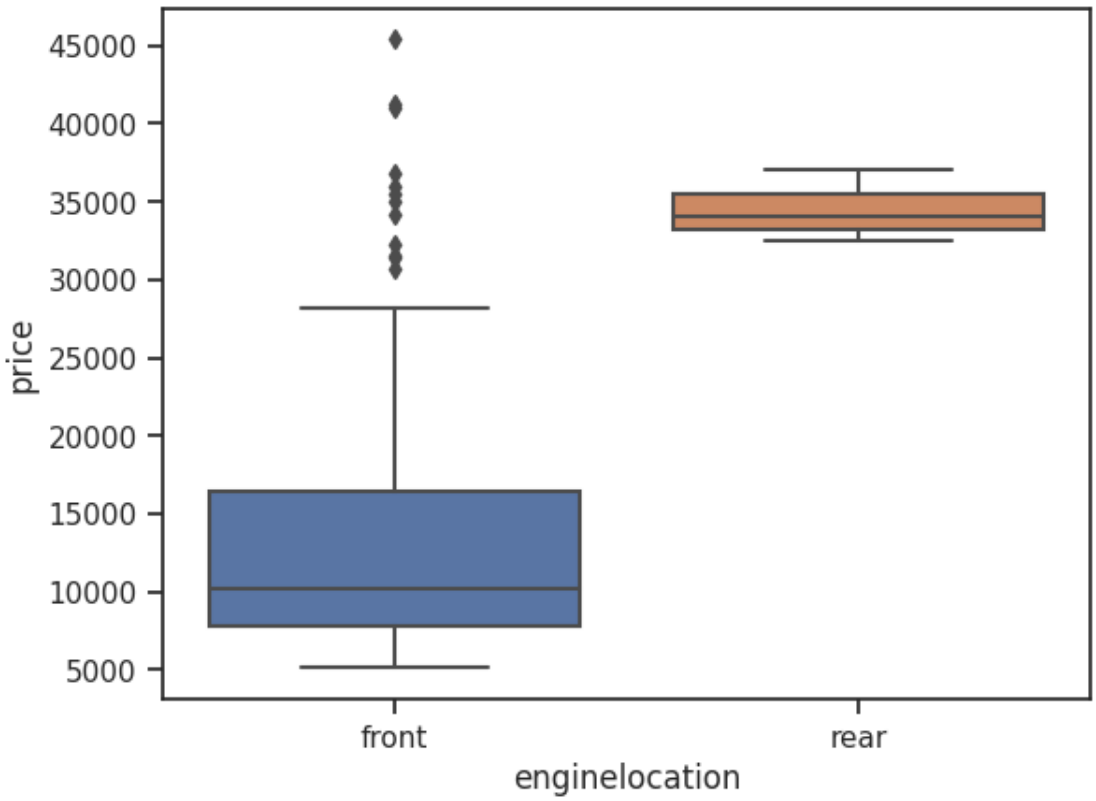
[ ]:
```
sns.boxplot(x="carbody",y="price",data=df)
plt.show()
```



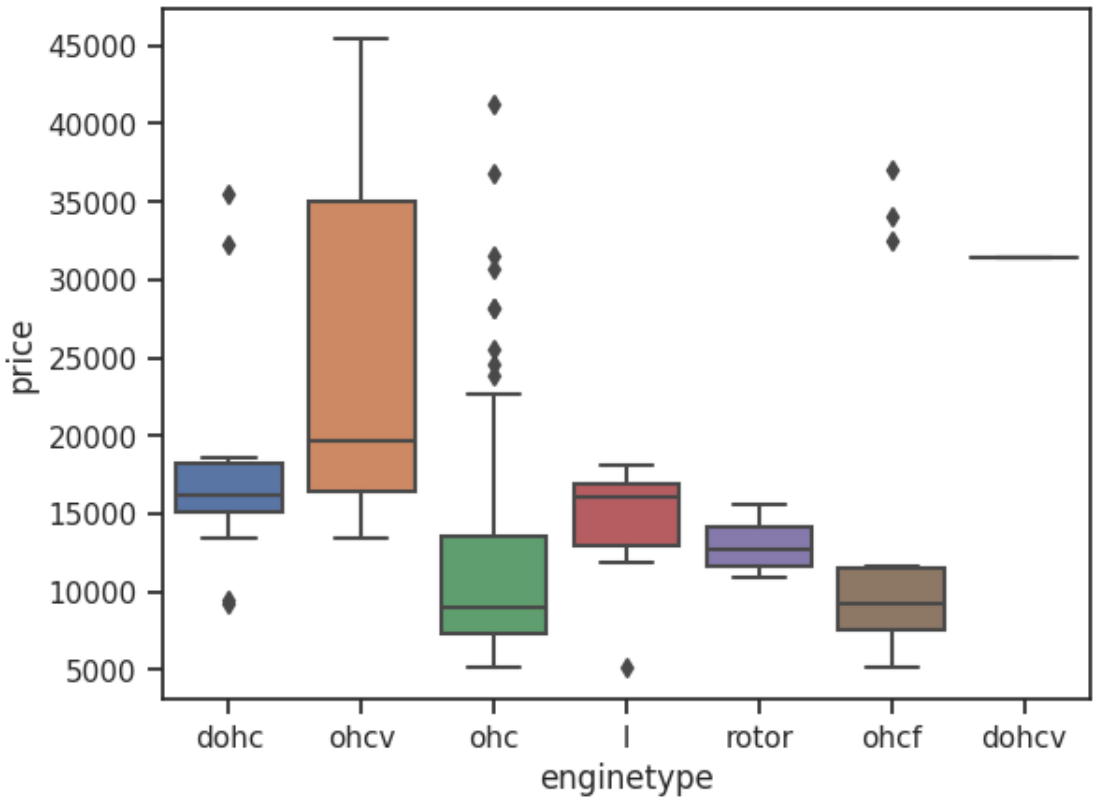
[ ]:
```
sns.boxplot(x="drivewheel",y="price",data=df)
plt.show()
```



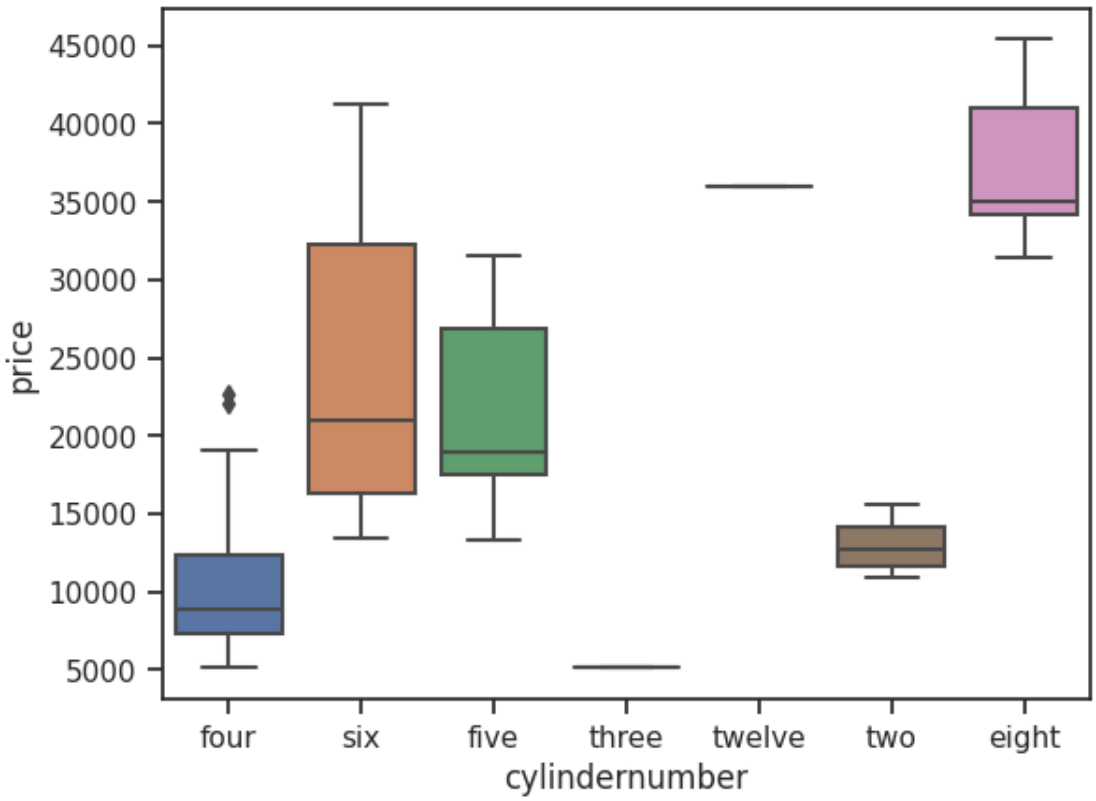
[ ]:
```
sns.boxplot(x="enginelocation",y="price",data=df)
plt.show()
```



[ ]:
```
sns.boxplot(x="enginetype",y="price",data=df)
plt.show()
```

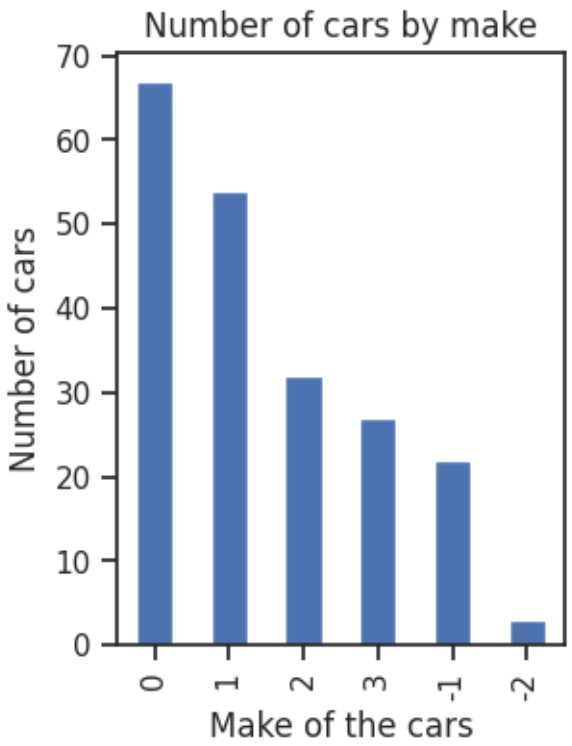


[ ]:
```
sns.boxplot(x="cylindernumber",y="price",data=df)
plt.show()
```



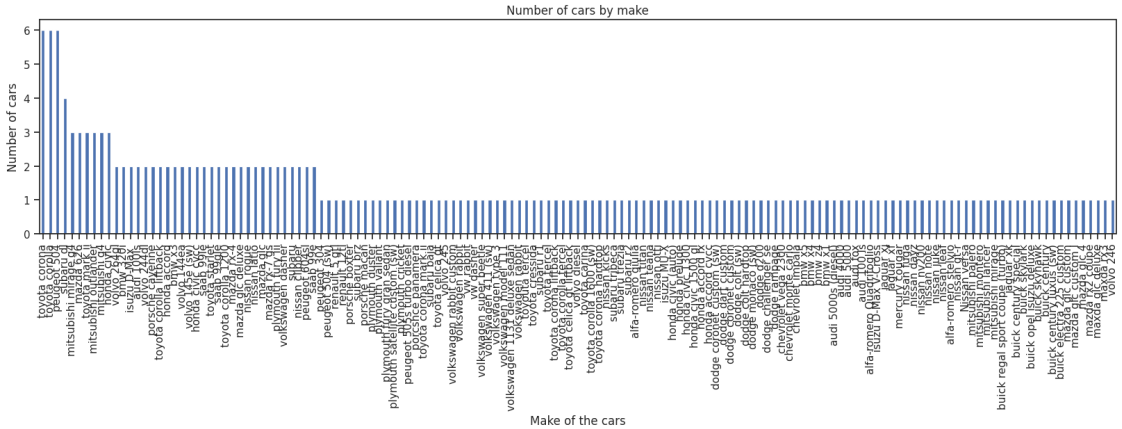
[ ]:
```
df.symboling.value_counts().plot(kind='bar', figsize=(3,4))
plt.title("Number of cars by make")
plt.ylabel('Number of cars')
plt.xlabel('Make of the cars')
```

[ ]:
```
Text(0.5, 0, 'Make of the cars')
```



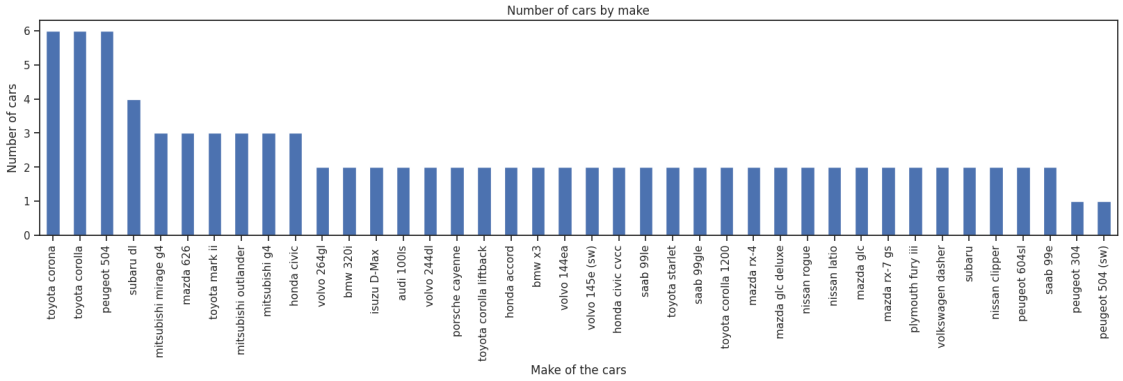
[ ]:
```
df.CarName.value_counts().plot(kind='bar', figsize=(20,4))
plt.title("Number of cars by make")
plt.ylabel('Number of cars')
plt.xlabel('Make of the cars')
```

[ ]:
```
Text(0.5, 0, 'Make of the cars')
```



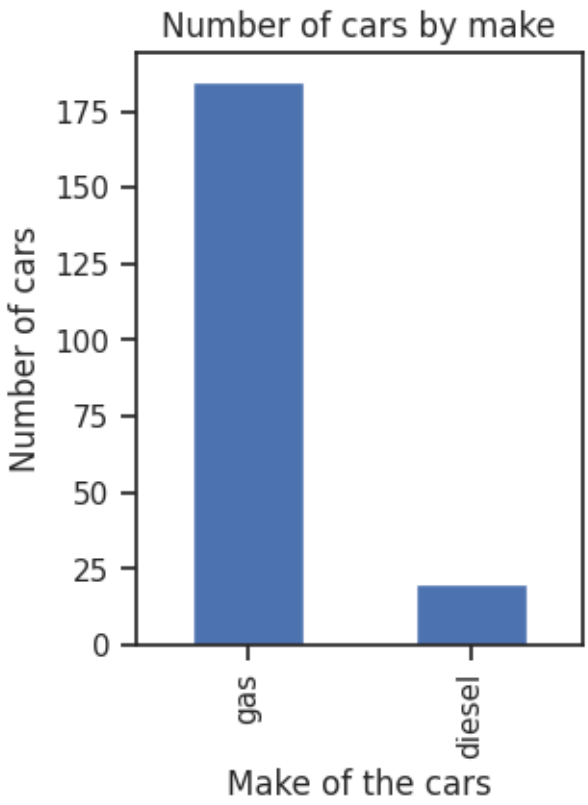
[ ]:
```
df.CarName.value_counts().nlargest(40).plot(kind='bar', figsize=(20,4))
plt.title("Number of cars by make")
plt.ylabel('Number of cars')
plt.xlabel('Make of the cars')
```

[ ]: Text(0.5, 0, 'Make of the cars')



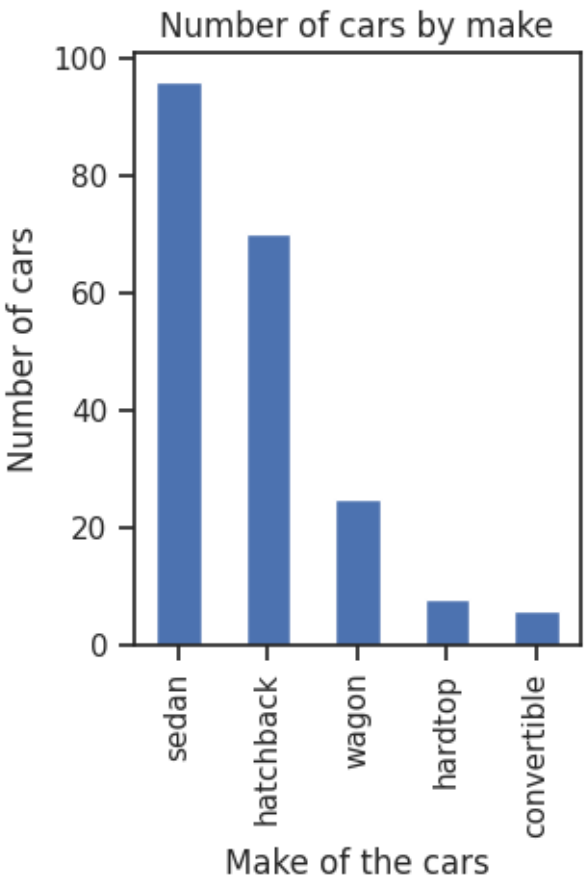
[ ]:
```
df.fueltype.value_counts().plot(kind='bar', figsize=(3,4))
plt.title("Number of cars by make")
plt.ylabel('Number of cars')
plt.xlabel('Make of the cars')
```

[ ]: Text(0.5, 0, 'Make of the cars')



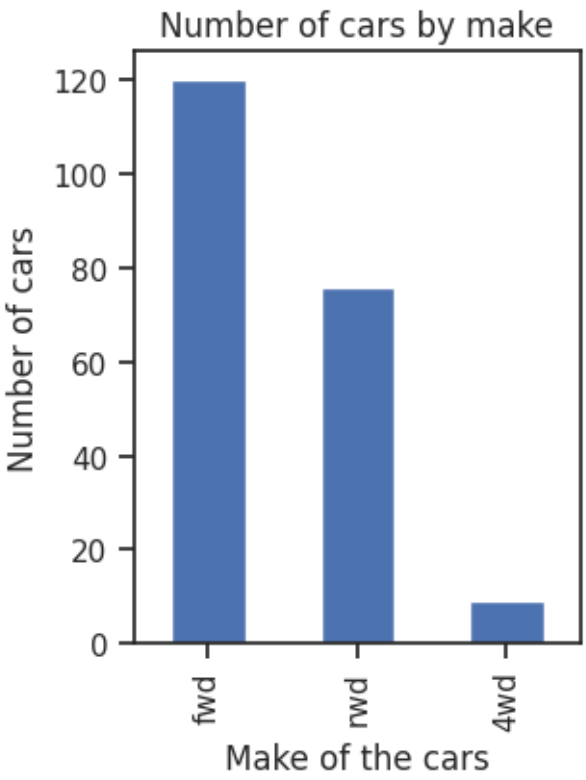
[ ]:
```
df.carbody.value_counts().plot(kind='bar', figsize=(3,4))
plt.title("Number of cars by make")
plt.ylabel('Number of cars')
plt.xlabel('Make of the cars')
```

[ ]:
```
Text(0.5, 0, 'Make of the cars')
```



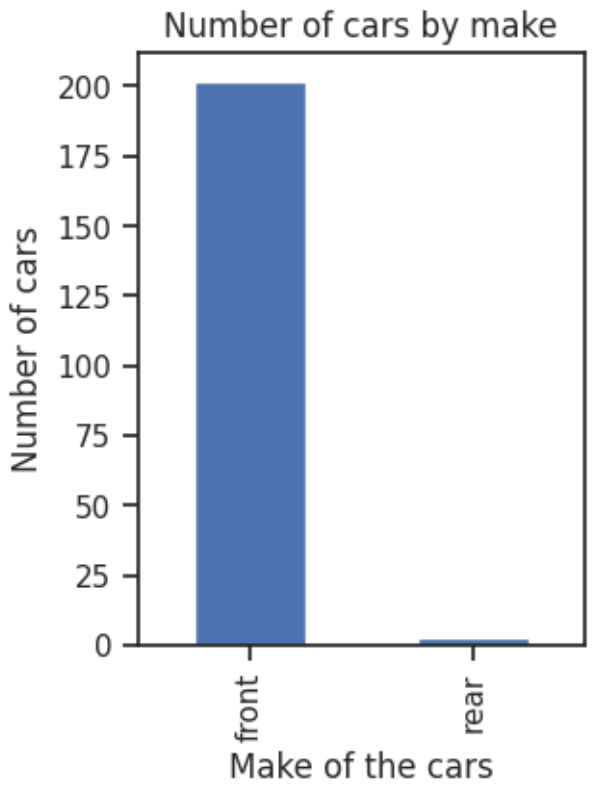
[ ]:
```
df.drivewheel.value_counts().plot(kind='bar', figsize=(3,4))
plt.title("Number of cars by make")
plt.ylabel('Number of cars')
plt.xlabel('Make of the cars')
```

[ ]:
```
Text(0.5, 0, 'Make of the cars')
```



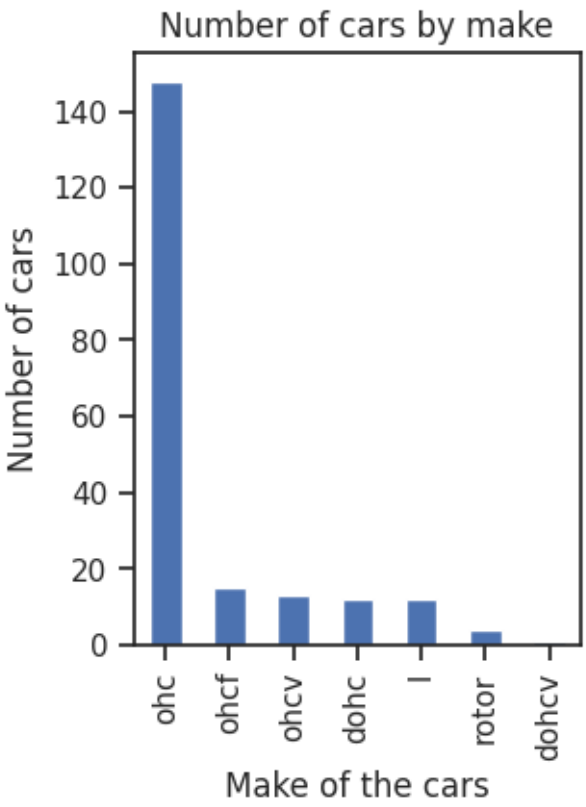
[ ]:
```
df.enginelocation.value_counts().plot(kind='bar', figsize=(3,4))
plt.title("Number of cars by make")
plt.ylabel('Number of cars')
plt.xlabel('Make of the cars')
```

[ ]:
```
Text(0.5, 0, 'Make of the cars')
```



[ ]:
```
df.enginetype.value_counts().plot(kind='bar', figsize=(3,4))
plt.title("Number of cars by make")
plt.ylabel('Number of cars')
plt.xlabel('Make of the cars')
```

[ ]:
```
Text(0.5, 0, 'Make of the cars')
```



Con respecto a todo el analisis descriptivo que se llevo a cabo, las variables numericas con las que nos quedaremos fueron, enginesize, highwaying, horsepower, curbweight y carlenght debido a que tienen correlacion con muchas de las otras variables, incluso categoricas, osea pueden describir algunas de las variables faltantes y tambien tienen alta correlacion con la variable precio.

Con respecto a las categoricas se escogio una, enginelocation, la parte mala es que tiene pocos datos pero sus distribuciones con respecto al precio estan muy alejadas y eso puede ser bueno para el modelo para disernir si su precio es alto o bajo

[ ]:
```
shapiro(df.carlength)
```

[ ]:
```
ShapiroResult(statistic=0.9820948839187622, pvalue=0.01036082860082388)
```

el resultado del ajuste indica que la distribución no se ajusta a los datos. Por lo tanto, se puede decir que los datos no tienen una distribución normal.

[ ]:
```
shapiro(df.highwaympg)
```

[ ]:
```
ShapiroResult(statistic=0.9735105633735657, pvalue=0.000651571957860142)
```

el resultado del ajuste indica que la distribución no se ajusta a los datos. Por lo tanto, se puede decir que los datos no tienen una distribución normal.

[ ]:
```
shapiro(df.enginesize)
```

[ ]:
```
ShapiroResult(statistic=0.8294388651847839, pvalue=3.0569154792128156e-14)
```

el resultado del ajuste indica que la distribución no se ajusta a los datos. Por lo tanto, se puede decir que los datos no tienen una distribución normal.

[ ]:
```
shapiro(df.horsepower)
```

[ ]:
```
ShapiroResult(statistic=0.8836246728897095, pvalue=1.736599579416076e-11)
```

el resultado del ajuste indica que la distribución no se ajusta a los datos. Por lo tanto, se puede decir que los datos no tienen una distribución normal.

[ ]:
```
shapiro(df.curbweight)
```

[ ]:
```
ShapiroResult(statistic=0.9530424475669861, pvalue=2.8916113024024526e-06)
```

el resultado del ajuste indica que la distribución no se ajusta a los datos. Por lo tanto, se puede decir que los datos no tienen una distribución normal.