20bsit154-ass-2

March 30, 2023

[1]:
```
#1.         Binning using Python
#1.         Implement Binning using cut and qcut methods.
#2.         Also, transform the bins values.

import pandas as pd
na1 = ["n.a.", "not available"]

df = pd.read_csv("D:/sem6/dma/practical/Automobile_data.csv", na_values = na1)
```

[2]:
```
print(df)
```

```
    index      company   body-style  wheel-base  length engine-type  \
0       0  alfa-romero  convertible        88.6   168.8        dohc
1       1  alfa-romero  convertible        88.6   168.8        dohc
2       2  alfa-romero    hatchback        94.5   171.2        ohcv
3       3         audi        sedan        99.8   176.6         ohc
4       4         audi        sedan        99.4   176.6         ohc
..    …          …          …          …     …         …
56     81   volkswagen        sedan        97.3   171.7         ohc
57     82   volkswagen        sedan        97.3   171.7         ohc
58     86   volkswagen        sedan        97.3   171.7         ohc
59     87        volvo        sedan       104.3   188.8         ohc
60     88        volvo        wagon       104.3   188.8         ohc

   num-of-cylinders  horsepower  average-mileage    price
0              four         111               21  13495.0
1              four         111               21  16500.0
2               six         154               19  16500.0
3              four         102               24  13950.0
4              five         115               18  17450.0
..              …         …              …       …
56             four          85               27   7975.0
57             four          52               37   7995.0
58             four         100               26   9995.0
59             four         114               23  12940.0
60             four         114               23  13415.0

[61 rows x 10 columns]
```

[43]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9 entries, 0 to 8
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   index              9 non-null      int64
 1   company            9 non-null      object
 2   body-style         9 non-null      object
 3   wheel-base         9 non-null      float64
 4   length             9 non-null      float64
 5   engine-type        9 non-null      object
 6   num-of-cylinders   9 non-null      object
 7   horsepower         8 non-null      float64
 8   average-mileage    9 non-null      int64
 9   price              7 non-null      float64
dtypes: float64(4), int64(2), object(4)
memory usage: 848.0+ bytes
```

[4]:
```
df['wheel-base'].min()
```

[4]: 88.4

[6]:
```
df['wheel-base'].max()
```

[6]: 120.9

[11]:
```
#cut
bins = [88,100,130]
labels = ['small','medium']
```

[12]:
```
df['grade'] = pd.cut(df['wheel-base'], bins =bins, labels = labels,␣
 ↪include_lowest=True)
```

[13]:
```
print(df['grade'], df['wheel-base'])
```

```
0       small
1       small
2       small
3       small
4       small
        …
56      small
57      small
58      small
59     medium
60     medium
```

```
Name: grade, Length: 61, dtype: category
Categories (2, object): ['small' < 'medium'] 0      88.6
1      88.6
2      94.5
3      99.8
4      99.4
       …
56     97.3
57     97.3
58     97.3
59    104.3
60    104.3
Name: wheel-base, Length: 61, dtype: float64
```

[14]:
```
df['grade'].value_counts()
```

[14]:
```
small     42
medium    19
Name: grade, dtype: int64
```

[52]:
```
#qcut

df.describe()
```

[52]:
```
          index  wheel-base      length  horsepower  average-mileage  \
count  9.000000    9.000000    9.000000    8.000000         9.000000
mean   4.444444   97.655556  176.177778  113.125000        20.777778
std    3.431877    5.891331    7.150311   17.365298         2.166667
min    0.000000   88.600000  168.800000  101.000000        18.000000
25%    2.000000   94.500000  171.200000  101.750000        19.000000
50%    4.000000   99.800000  176.600000  110.500000        21.000000
75%    6.000000  101.200000  176.800000  112.000000        23.000000
max   10.000000  105.800000  192.700000  154.000000        24.000000

              price
count      7.000000
mean   16667.857143
std     1486.028985
min    13950.000000
25%    16465.000000
50%    16500.000000
75%    17187.500000
max    18920.000000
```

[15]:
```
df['grades'] = pd.qcut(df['wheel-base'], q=2)
```

[16]:
```
print(df['grades'])
```

```
0      (88.399, 96.3]
1      (88.399, 96.3]
2      (88.399, 96.3]
3       (96.3, 120.9]
4       (96.3, 120.9]
            …
56      (96.3, 120.9]
57      (96.3, 120.9]
58      (96.3, 120.9]
59      (96.3, 120.9]
60      (96.3, 120.9]
Name: grades, Length: 61, dtype: category
Categories (2, interval[float64, right]): [(88.399, 96.3] < (96.3, 120.9]]
```

[17]:
```python
df['grades'].value_counts()
```

[17]:
```
(88.399, 96.3]    31
(96.3, 120.9]     30
Name: grades, dtype: int64
```

[18]:
```python
labels = ['small','medium']
df['grades'] = pd.qcut(df['wheel-base'], q=2, labels = labels)
df['grades'].value_counts()
print(df['grades'], df['wheel-base'])
```

```
0       small
1       small
2       small
3      medium
4      medium
        …
56     medium
57     medium
58     medium
59     medium
60     medium
Name: grades, Length: 61, dtype: category
Categories (2, object): ['small' < 'medium'] 0       88.6
1       88.6
2       94.5
3       99.8
4       99.4
      …
56      97.3
57      97.3
58      97.3
59     104.3
60     104.3
```

```
Name: wheel-base, Length: 61, dtype: float64
```

[19]:
```python
df.groupby('grades')['wheel-base'].transform('mean')
```

[19]:
```
0      93.729032
1      93.729032
2      93.729032
3     103.393333
4     103.393333
         …
56    103.393333
57    103.393333
58    103.393333
59    103.393333
60    103.393333
Name: wheel-base, Length: 61, dtype: float64
```

[20]:
```python
df.groupby('grades')['wheel-base'].transform('median')
```

[20]:
```
0      94.5
1      94.5
2      94.5
3     101.6
4     101.6
      …
56    101.6
57    101.6
58    101.6
59    101.6
60    101.6
Name: wheel-base, Length: 61, dtype: float64
```

[21]:
```python
#2.         Outlier detection and removal
#1.         Detect the outlier using visualization method
#2.         Detect the outlier using statistical method
#3.         Treat the outliers

na1 = ["n.a.", "not available"]

df = pd.read_csv("D:/sem6/dma/practical/Automobile_data.csv", na_values = na1)
```

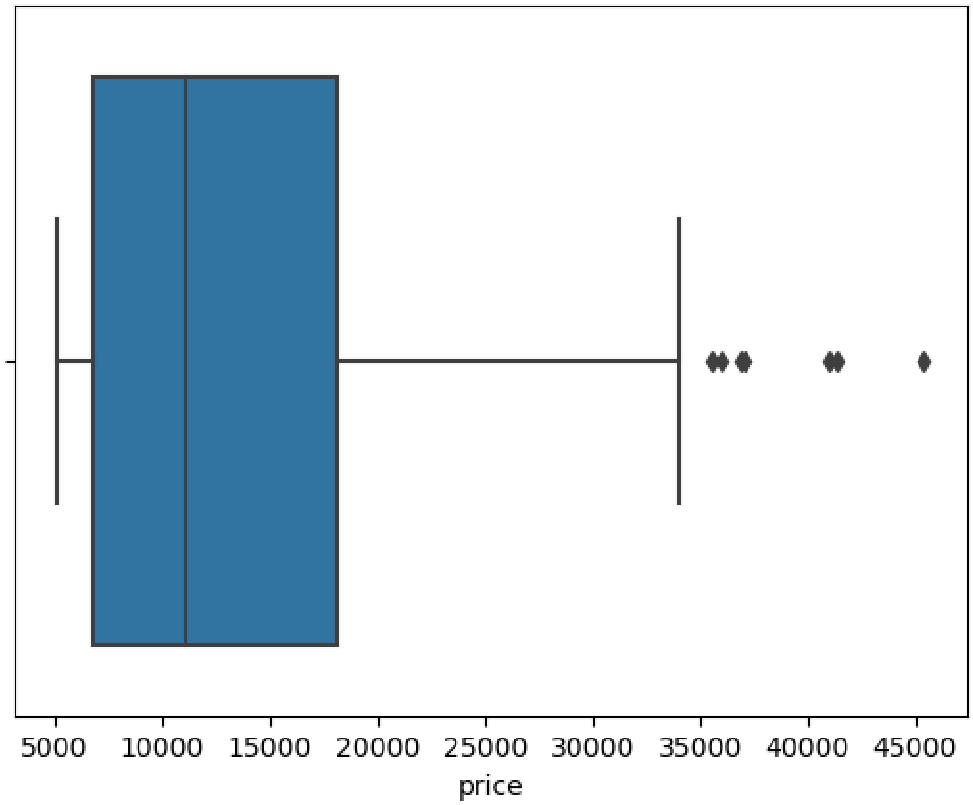
[22]:
```python
print(df)
```

```
    index      company   body-style  wheel-base  length engine-type  \
0       0  alfa-romero  convertible        88.6   168.8        dohc
1       1  alfa-romero  convertible        88.6   168.8        dohc
2       2  alfa-romero    hatchback        94.5   171.2        ohcv
3       3         audi        sedan        99.8   176.6         ohc
```

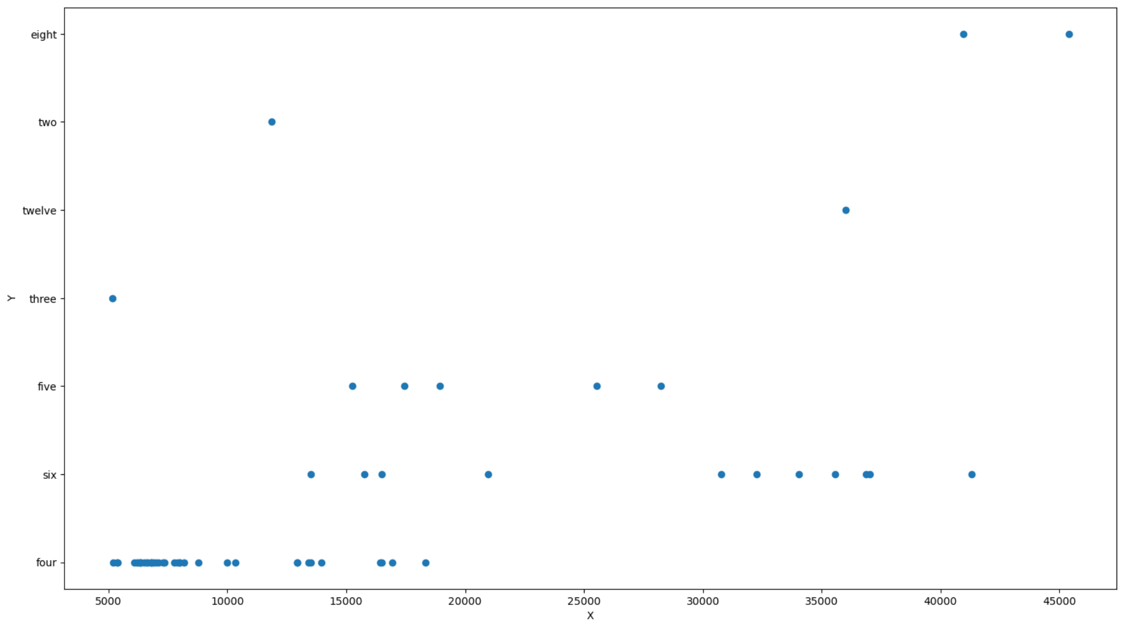
```
4       4         audi       sedan        99.4   176.6         ohc
..     ...          ...         ...        ...     ...         ...
56     81   volkswagen       sedan        97.3   171.7         ohc
57     82   volkswagen       sedan        97.3   171.7         ohc
58     86   volkswagen       sedan        97.3   171.7         ohc
59     87        volvo       sedan       104.3   188.8         ohc
60     88        volvo       wagon       104.3   188.8         ohc

   num-of-cylinders  horsepower  average-mileage    price
0              four         111               21  13495.0
1              four         111               21  16500.0
2               six         154               19  16500.0
3              four         102               24  13950.0
4              five         115               18  17450.0
..              ...         ...              ...      ...
56             four          85               27   7975.0
57             four          52               37   7995.0
58             four         100               26   9995.0
59             four         114               23  12940.0
60             four         114               23  13415.0

[61 rows x 10 columns]
```

[23]:
```
import seaborn as sns
sns.boxplot(x=df['price'])
```

[23]: <AxesSubplot:xlabel='price'>

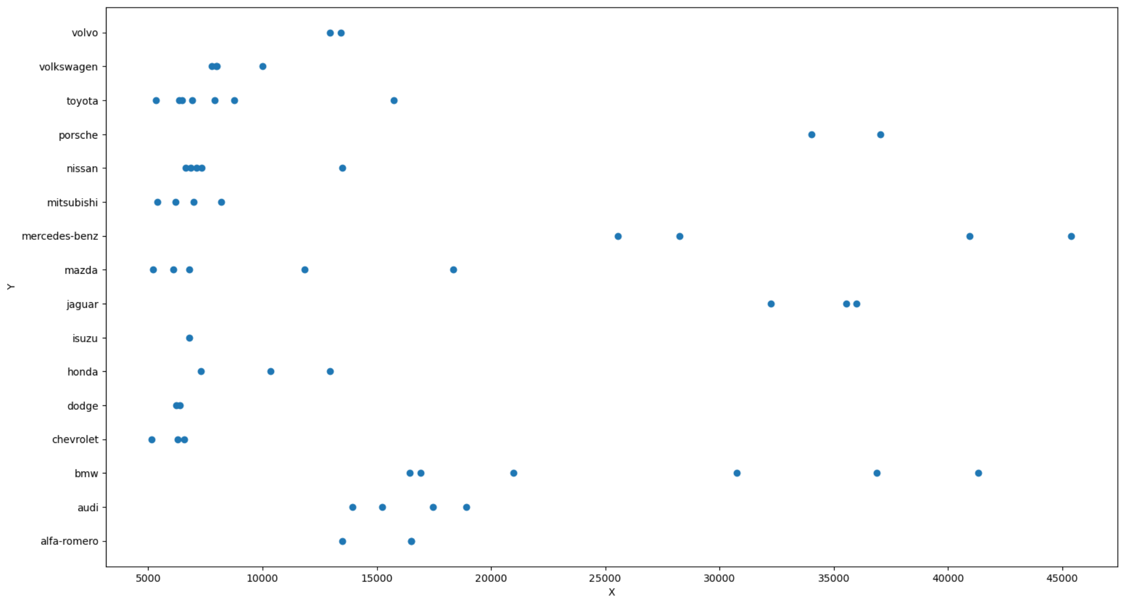


[24]:
```
# Scatter plot
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize = (18,10))
ax.scatter(df['price'], df['num-of-cylinders'])

# x-axis label
ax.set_xlabel('X')

# y-axis label
ax.set_ylabel('Y')
plt.show()
```

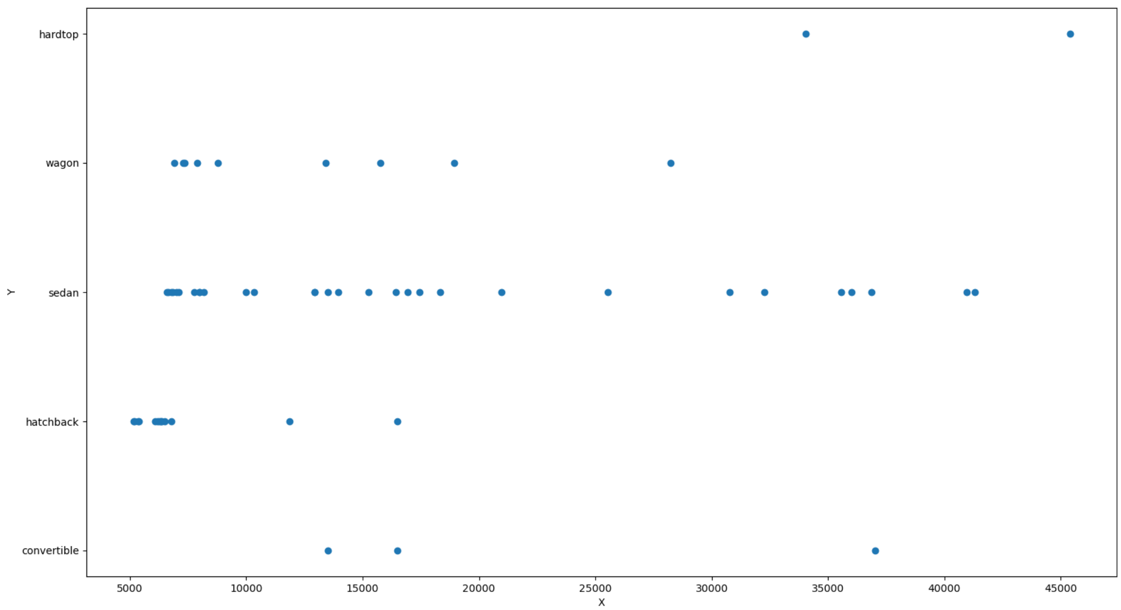


[25]:

```
# Scatter plot
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize = (18,10))
ax.scatter(df['price'], df['company'])

# x-axis label
ax.set_xlabel('X')

# y-axis label
ax.set_ylabel('Y')
plt.show()
```

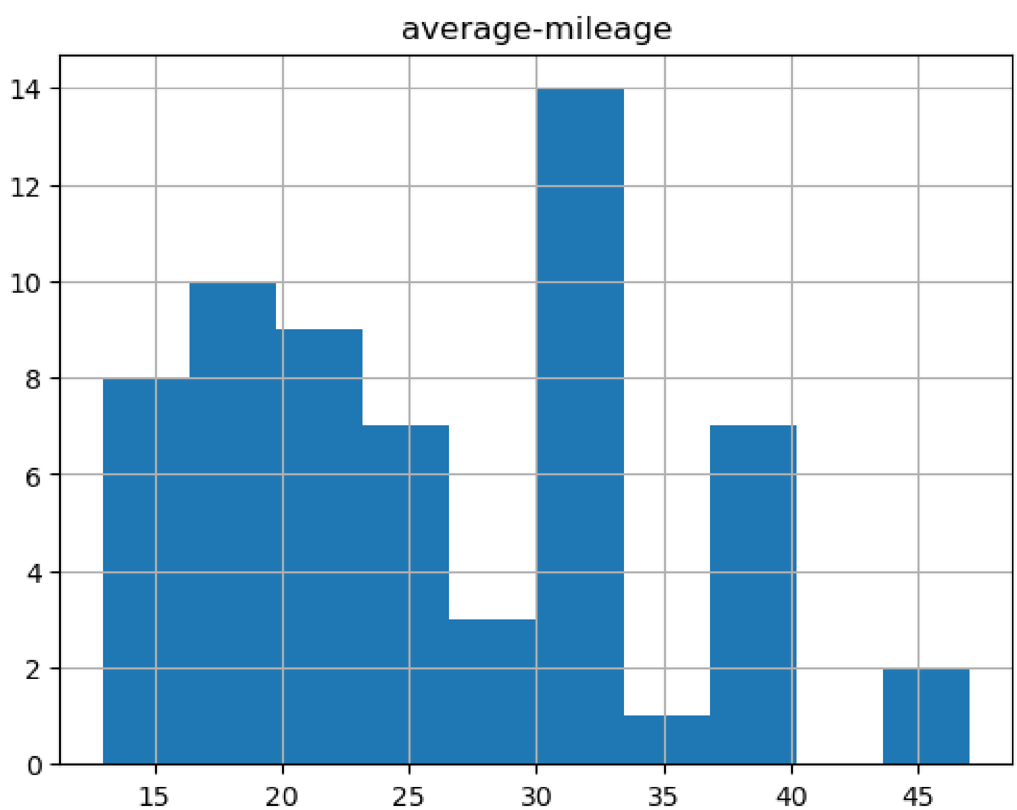


[26]:
```python
# Scatter plot
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize = (18,10))
ax.scatter(df['price'], df['body-style'])

# x-axis label
ax.set_xlabel('X')

# y-axis label
ax.set_ylabel('Y')
plt.show()
```



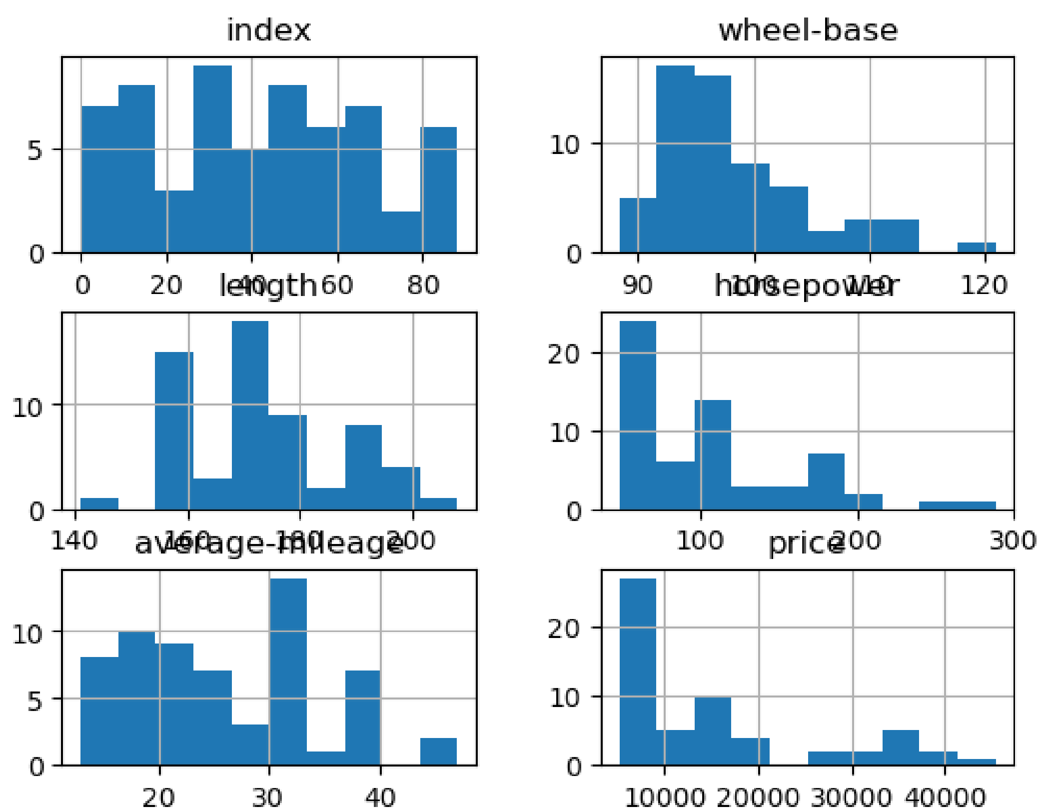
[27]:
```
df.hist(column='average-mileage', bins=10)
```

[27]: array([[<AxesSubplot:title={'center':'average-mileage'}>]], dtype=object)



[28]:
```
df.hist()
```

[28]:
```
array([[<AxesSubplot:title={'center':'index'}>,
        <AxesSubplot:title={'center':'wheel-base'}>],
       [<AxesSubplot:title={'center':'length'}>,
        <AxesSubplot:title={'center':'horsepower'}>],
       [<AxesSubplot:title={'center':'average-mileage'}>,
        <AxesSubplot:title={'center':'price'}>]], dtype=object)
```



[29]:
```
# Z score
from scipy import stats
import numpy as np

df['price z scores'] = stats.zscore(df['price'], nan_policy='omit')
print(df.head(40))
```

```
   index      company  body-style  wheel-base  length engine-type  \
0      0  alfa-romero  convertible        88.6   168.8        dohc
1      1  alfa-romero  convertible        88.6   168.8        dohc
```

2	2	alfa-romero	hatchback	94.5	171.2	ohcv
3	3	audi	sedan	99.8	176.6	ohc
4	4	audi	sedan	99.4	176.6	ohc
5	5	audi	sedan	99.8	177.3	ohc
6	6	audi	wagon	105.8	192.7	ohc
7	9	bmw	sedan	101.2	176.8	ohc
8	10	bmw	sedan	101.2	176.8	ohc
9	11	bmw	sedan	101.2	176.8	ohc
10	13	bmw	sedan	103.5	189.0	ohc
11	14	bmw	sedan	103.5	193.8	ohc
12	15	bmw	sedan	110.0	197.0	ohc
13	16	chevrolet	hatchback	88.4	141.1	l
14	17	chevrolet	hatchback	94.5	155.9	ohc
15	18	chevrolet	sedan	94.5	158.8	ohc
16	19	dodge	hatchback	93.7	157.3	ohc
17	20	dodge	hatchback	93.7	157.3	ohc
18	27	honda	wagon	96.5	157.1	ohc
19	28	honda	sedan	96.5	175.4	ohc
20	29	honda	sedan	96.5	169.1	ohc
21	30	isuzu	sedan	94.3	170.7	ohc
22	31	isuzu	sedan	94.5	155.9	ohc
23	32	isuzu	sedan	94.5	155.9	ohc
24	33	jaguar	sedan	113.0	199.6	dohc
25	34	jaguar	sedan	113.0	199.6	dohc
26	35	jaguar	sedan	102.0	191.7	ohcv
27	36	mazda	hatchback	93.1	159.1	ohc
28	37	mazda	hatchback	93.1	159.1	ohc
29	38	mazda	hatchback	93.1	159.1	ohc
30	39	mazda	hatchback	95.3	169.0	rotor
31	43	mazda	sedan	104.9	175.0	ohc
32	44	mercedes-benz	sedan	110.0	190.9	ohc
33	45	mercedes-benz	wagon	110.0	190.9	ohc
34	46	mercedes-benz	sedan	120.9	208.1	ohcv
35	47	mercedes-benz	hardtop	112.0	199.2	ohcv
36	49	mitsubishi	hatchback	93.7	157.3	ohc
37	50	mitsubishi	hatchback	93.7	157.3	ohc
38	51	mitsubishi	sedan	96.3	172.4	ohc
39	52	mitsubishi	sedan	96.3	172.4	ohc

	num-of-cylinders	horsepower	average-mileage	price	price z scores
0	four	111	21	13495.0	-0.168594
1	four	111	21	16500.0	0.099178
2	six	154	19	16500.0	0.099178
3	four	102	24	13950.0	-0.128049
4	five	115	18	17450.0	0.183831
5	five	110	19	15250.0	-0.012208
6	five	110	19	18920.0	0.314821
7	four	101	23	16430.0	0.092940

```
8          four   101     23  16925.0   0.137049
9           six   121     21  20970.0   0.497494
10          six   182     16  30760.0   1.369868
11          six   182     16  41315.0   2.310411
12          six   182     15  36880.0   1.915214
13        three    48     47   5151.0  -0.912117
14         four    70     38   6295.0  -0.810176
15         four    70     38   6575.0  -0.785226
16         four    68     31   6377.0  -0.802870
17         four    68     31   6229.0  -0.816058
18         four    76     30   7295.0  -0.721068
19         four   101     24  12945.0  -0.217603
20         four   100     25  10345.0  -0.449286
21         four    78     24   6785.0  -0.766513
22         four    70     38      NaN        NaN
23         four    70     38      NaN        NaN
24          six   176     15  32250.0   1.502640
25          six   176     15  35550.0   1.796699
26       twelve   262     13  36000.0   1.836798
27         four    68     30   5195.0  -0.908196
28         four    68     31   6095.0  -0.827998
29         four    68     31   6795.0  -0.765622
30          two   101     17  11845.0  -0.315623
31         four    72     31  18344.0   0.263494
32         five   123     22  25552.0   0.905790
33         five   123     22  28248.0   1.146027
34        eight   184     14  40960.0   2.278777
35        eight   184     14  45400.0   2.674420
36         four    68     37   5389.0  -0.890909
37         four    68     31   6189.0  -0.819622
38         four    88     25   6989.0  -0.748335
39         four    88     25   8189.0  -0.641405
```

[30]:
```
threshold = 3
# Position of the outlier
print(np.where(df['price z scores'] > 2))
```

```
(array([11, 34, 35], dtype=int64),)
```

[32]:
```
# IQR
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
index                 43.000000
wheel-base             6.700000
length                18.200000
```

```
horsepower              55.000000
average-mileage         12.000000
price                11312.000000
price z scores           1.007998
dtype: float64
```

[33]:
```
upper = Q3 + 1.5*IQR

lower = Q1 - 1.5*IQR

print(upper)

print(lower)
```

```
index                  125.500000
wheel-base             111.250000
length                 204.600000
horsepower             205.500000
average-mileage         49.000000
price                35088.500000
price z scores           1.755575
dtype: float64
index                  -46.500000
wheel-base              84.450000
length                 131.800000
horsepower             -14.500000
average-mileage          1.000000
price               -10159.500000
price z scores          -2.276416
dtype: float64
```

[34]:
```
Q1 = df['price'].quantile(0.25)
Q3 = df['price'].quantile(0.75)
IQR = Q3 - Q1


upper = Q3 + 1.5*IQR

lower = Q1 - 1.5*IQR

print(upper)
print(lower)
```

```
35088.5
-10159.5
```

[35]:
```
#print outliers
outliers = df['price'][((df['price']<(Q1-1.5*IQR)) | (df['price']>(Q3+1.
 ↪5*IQR)))]
print(outliers)
```

```
11    41315.0
12    36880.0
25    35550.0
26    36000.0
34    40960.0
35    45400.0
46    37028.0
Name: price, dtype: float64
```

[36]:
```
#replace some of the outlier values - approach 1

df['price'].replace([41315.0,36880.0],[111,111],inplace=True)

outliers = df['price'][((df['price']<(Q1-1.5*IQR)) | (df['price']>(Q3+1.
 ↪5*IQR)))]
print(outliers)
```

```
25    35550.0
26    36000.0
34    40960.0
35    45400.0
46    37028.0
Name: price, dtype: float64
```

[17]:
```
#drop the row that contains the outlier - approach 2
df.drop(index=34, inplace = True)

print("New Shape: ", df.shape)
```

```
New Shape:  (60, 11)
```

[37]:
```
outliers = df['price'][((df['price']<(Q1-1.5*IQR)) | (df['price']>(Q3+1.
 ↪5*IQR)))]
print(outliers)
```

```
25    35550.0
26    36000.0
34    40960.0
35    45400.0
46    37028.0
Name: price, dtype: float64
```

[ ]:

```

```