HW1_coding

October 11, 2024

Pong Yui Yi Monica SID:20853295

1 Chapter 2

1.0.1 question 8

(a). Use the pd.read_csv() function to read the data into Python. Call the loaded data college. Make sure that you have the directory set to the correct location for the data.

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

[ ]:
```
college = pd.read_csv("College.csv")
college
```

[ ]:
```
                         Unnamed: 0 Private   Apps  Accept  Enroll  Top10perc  \
0      Abilene Christian University     Yes   1660    1232     721         23
1                Adelphi University     Yes   2186    1924     512         16
2                    Adrian College     Yes   1428    1097     336         22
3               Agnes Scott College     Yes    417     349     137         60
4         Alaska Pacific University     Yes    193     146      55         16
..                              …     …    …     …     …        …
772          Worcester State College      No   2197    1515     543          4
773               Xavier University     Yes   1959    1805     695         24
774  Xavier University of Louisiana     Yes   2097    1915     695         34
775                 Yale University     Yes  10705    2453    1317         95
776    York College of Pennsylvania     Yes   2989    1855     691         28

     Top25perc  F.Undergrad  P.Undergrad  Outstate  Room.Board  Books  \
0           52         2885          537      7440        3300    450
1           29         2683         1227     12280        6450    750
2           50         1036           99     11250        3750    400
3           89          510           63     12960        5450    450
4           44          249          869      7560        4120    800
..         …          …          …       …         …    …
772         26         3089         2029      6797        3900    500
```

```
773         47          2849          1107      11520          4960     600
774         61          2793           166       6900          4200     617
775         99          5217            83      19840          6510     630
776         63          2988          1726       4990          3560     500

     Personal  PhD  Terminal  S.F.Ratio  perc.alumni  Expend  Grad.Rate
0        2200   70        78       18.1           12    7041         60
1        1500   29        30       12.2           16   10527         56
2        1165   53        66       12.9           30    8735         54
3         875   92        97        7.7           37   19016         59
4        1500   76        72       11.9            2   10922         15
..        …   …       …        …          …      …        …
772      1200   60        60       21.0           14    4469         40
773      1250   73        75       13.3           31    9189         83
774       781   67        75       14.4           20    8323         49
775      2115   96        96        5.8           49   40386         99
776      1250   75        75       18.1           28    4509         99

[777 rows x 19 columns]
```

(b) Look at the data used in the notebook by creating and running a new cell with just the code college in it. You should notice that the first column is just the name of each university in a column named something like Unnamed: 0. We don't really want pandas to treat this as data. However, it may be handy to have these names for later. Try the following commands and similarly look at the resulting data frames:

[ ]:
```
college2 = pd.read_csv('College.csv', index_col=0)
college3 = college.rename({'Unnamed: 0': 'College'},axis=1)
college3 = college3.set_index('College')
college = college3
college
```

[ ]:
```
                               Private   Apps  Accept  Enroll  Top10perc  \
College
Abilene Christian University       Yes   1660    1232     721         23
Adelphi University                 Yes   2186    1924     512         16
Adrian College                     Yes   1428    1097     336         22
Agnes Scott College                Yes    417     349     137         60
Alaska Pacific University          Yes    193     146      55         16
…                                  …    …      …      …       …
Worcester State College             No   2197    1515     543          4
Xavier University                  Yes   1959    1805     695         24
Xavier University of Louisiana     Yes   2097    1915     695         34
Yale University                    Yes  10705    2453    1317         95
York College of Pennsylvania       Yes   2989    1855     691         28

                                Top25perc  F.Undergrad  P.Undergrad  Outstate  \
College
```

```
Abilene Christian University        52        2885        537     7440
Adelphi University                  29        2683       1227    12280
Adrian College                      50        1036         99    11250
Agnes Scott College                 89         510         63    12960
Alaska Pacific University           44         249        869     7560
…                                 …         …          …        …
Worcester State College             26        3089       2029     6797
Xavier University                   47        2849       1107    11520
Xavier University of Louisiana      61        2793        166     6900
Yale University                     99        5217         83    19840
York College of Pennsylvania        63        2988       1726     4990

                              Room.Board  Books  Personal  PhD  Terminal  \
College
Abilene Christian University        3300    450      2200   70        78
Adelphi University                  6450    750      1500   29        30
Adrian College                      3750    400      1165   53        66
Agnes Scott College                 5450    450       875   92        97
Alaska Pacific University           4120    800      1500   76        72
…                                    …     …        …  …        …
Worcester State College             3900    500      1200   60        60
Xavier University                   4960    600      1250   73        75
Xavier University of Louisiana      4200    617       781   67        75
Yale University                     6510    630      2115   96        96
York College of Pennsylvania        3560    500      1250   75        75

                              S.F.Ratio  perc.alumni  Expend  Grad.Rate
College
Abilene Christian University       18.1           12    7041         60
Adelphi University                 12.2           16   10527         56
Adrian College                     12.9           30    8735         54
Agnes Scott College                 7.7           37   19016         59
Alaska Pacific University          11.9            2   10922         15
…                                 …          …    …       …
Worcester State College            21.0           14    4469         40
Xavier University                  13.3           31    9189         83
Xavier University of Louisiana     14.4           20    8323         49
Yale University                     5.8           49   40386         99
York College of Pennsylvania       18.1           28    4509         99

[777 rows x 18 columns]
```

(c). Use the describe() method of to produce a numerical summary of the variables in the data set.

[ ]:
```python
college.describe(include='number')
```

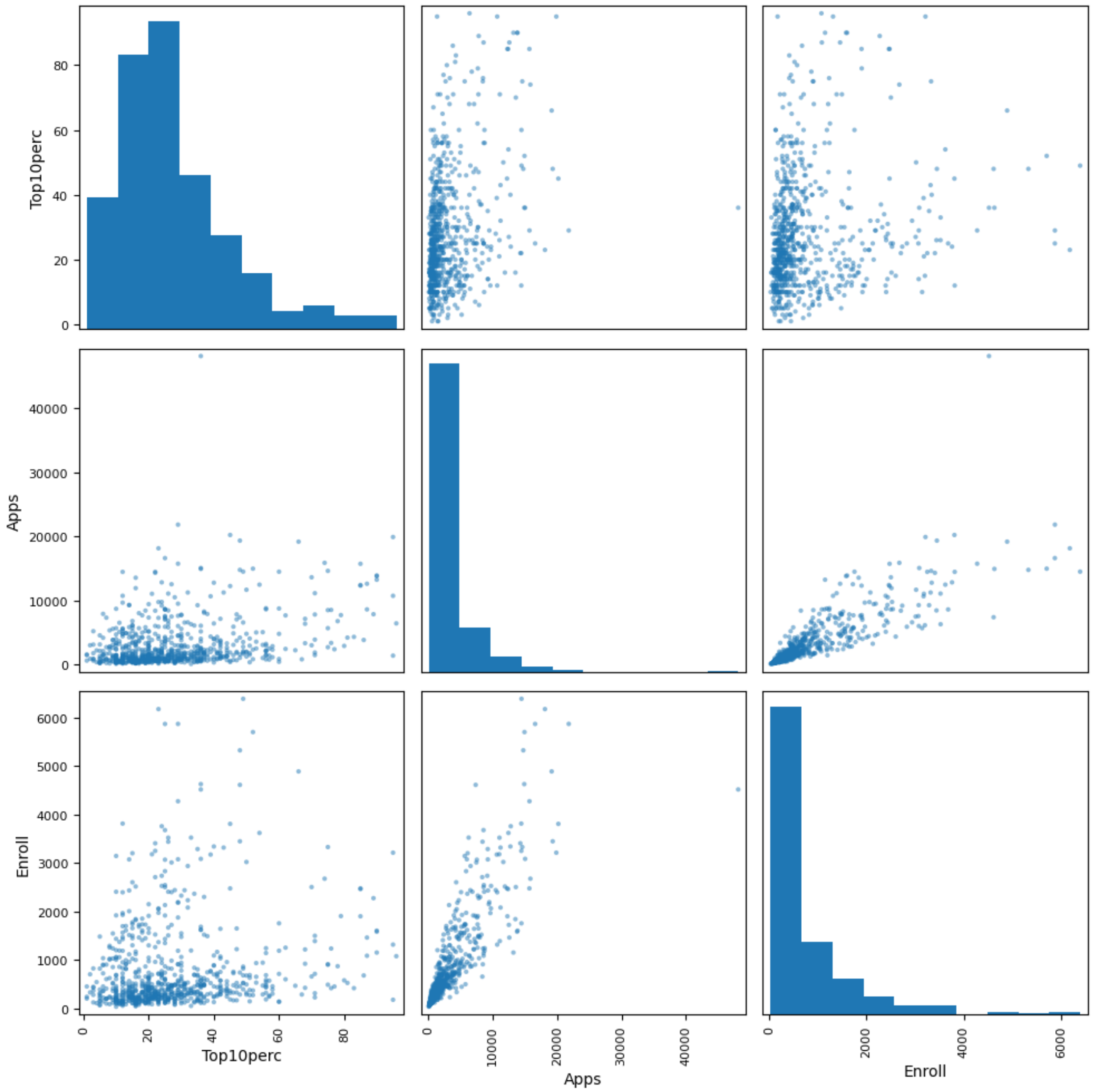
[ ]:

```
              Apps        Accept       Enroll    Top10perc    Top25perc  \
count    777.000000    777.000000   777.000000  777.000000   777.000000
mean    3001.638353   2018.804376   779.972973   27.558559    55.796654
std     3870.201484   2451.113971   929.176190   17.640364    19.804778
min       81.000000     72.000000    35.000000    1.000000     9.000000
25%      776.000000    604.000000   242.000000   15.000000    41.000000
50%     1558.000000   1110.000000   434.000000   23.000000    54.000000
75%     3624.000000   2424.000000   902.000000   35.000000    69.000000
max    48094.000000  26330.000000  6392.000000   96.000000   100.000000

        F.Undergrad   P.Undergrad      Outstate   Room.Board        Books  \
count    777.000000    777.000000    777.000000   777.000000   777.000000
mean    3699.907336    855.298584  10440.669241  4357.526384   549.380952
std     4850.420531   1522.431887   4023.016484  1096.696416   165.105360
min      139.000000      1.000000   2340.000000  1780.000000    96.000000
25%      992.000000     95.000000   7320.000000  3597.000000   470.000000
50%     1707.000000    353.000000   9990.000000  4200.000000   500.000000
75%     4005.000000    967.000000  12925.000000  5050.000000   600.000000
max    31643.000000  21836.000000  21700.000000  8124.000000  2340.000000

          Personal         PhD    Terminal   S.F.Ratio  perc.alumni  \
count   777.000000  777.000000  777.000000  777.000000   777.000000
mean   1340.642214   72.660232   79.702703   14.089704    22.743887
std     677.071454   16.328155   14.722359    3.958349    12.391801
min     250.000000    8.000000   24.000000    2.500000     0.000000
25%     850.000000   62.000000   71.000000   11.500000    13.000000
50%    1200.000000   75.000000   82.000000   13.600000    21.000000
75%    1700.000000   85.000000   92.000000   16.500000    31.000000
max    6800.000000  103.000000  100.000000   39.800000    64.000000

             Expend  Grad.Rate
count    777.000000  777.00000
mean    9660.171171   65.46332
std     5221.768440   17.17771
min     3186.000000   10.00000
25%     6751.000000   53.00000
50%     8377.000000   65.00000
75%    10830.000000   78.00000
max    56233.000000  118.00000
```

(d) Use the pd.plotting.scatter_matrix() function to produce a scatterplot matrix of the first columns [Top10perc, Apps, Enroll]. Recall that you can reference a list C of columns of a data frame A using A[C].

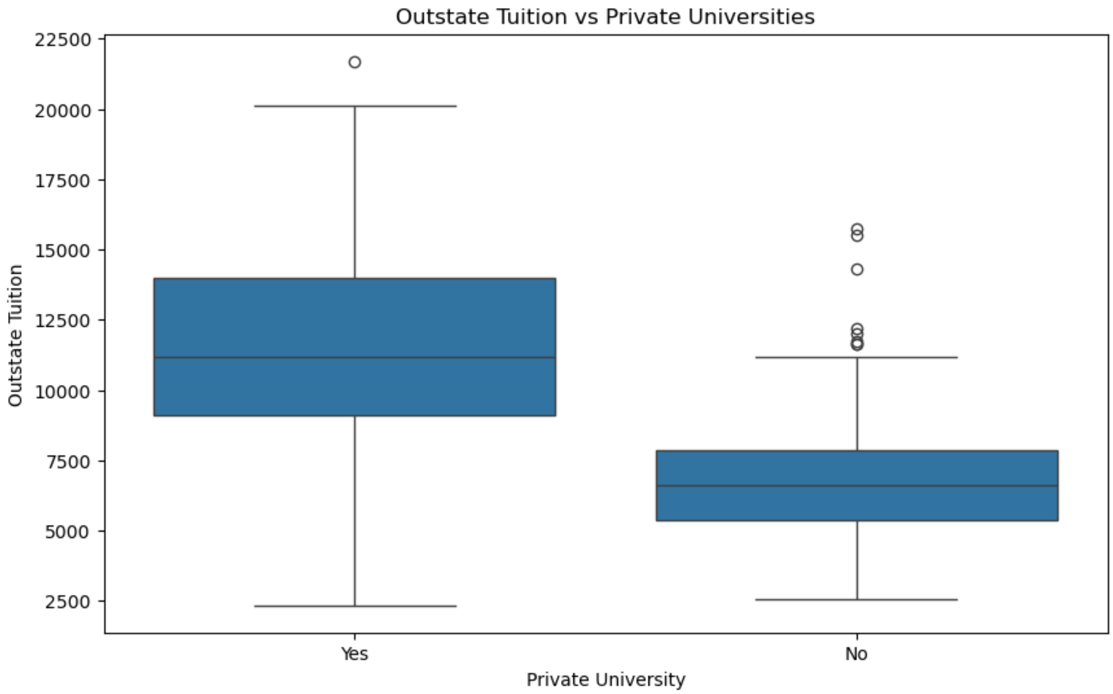
[ ]:

```
columns_to_plot = ['Top10perc', 'Apps', 'Enroll']
pd.plotting.scatter_matrix(college[columns_to_plot], figsize=(10, 10))
plt.tight_layout()
plt.show()
```



(e) Use the boxplot() method of college to produce side-by-side boxplots of Outstate versus Private.

[ ]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='Private', y='Outstate', data=college)
plt.title('Outstate Tuition vs Private Universities')
plt.xlabel('Private University')
plt.ylabel('Outstate Tuition')
plt.show()
```



(f) Create a new qualitative variable, called Elite, by binning the Top10perc variable into two groups based on whether or not the proportion of students coming from the top 10% of their high school classes exceeds 50%.

college[‘Elite’] = pd.cut(college[‘Top10perc’], [0,0.5,1], labels=[‘No’, ‘Yes’])

Use the value_counts() method of college[‘Elite’] to see how many elite universities there are. Finally, use the boxplot() method again to produce side-by-side boxplots of Outstate versus Elite.

[ ]:
```
college['Elite'] = pd.cut(college['Top10perc'],[0,0.5,1],labels=['No', 'Yes'])
college['Elite'] = college['Elite'].fillna('No')
college.head()
```

[ ]:
```
                             Private  Apps  Accept  Enroll  Top10perc  \
College
Abilene Christian University     Yes  1660    1232     721         23
Adelphi University               Yes  2186    1924     512         16
Adrian College                   Yes  1428    1097     336         22
Agnes Scott College              Yes   417     349     137         60
Alaska Pacific University        Yes   193     146      55         16

                              Top25perc  F.Undergrad  P.Undergrad  Outstate  \
College
Abilene Christian University         52         2885          537      7440
Adelphi University                   29         2683         1227     12280
Adrian College                       50         1036           99     11250
```

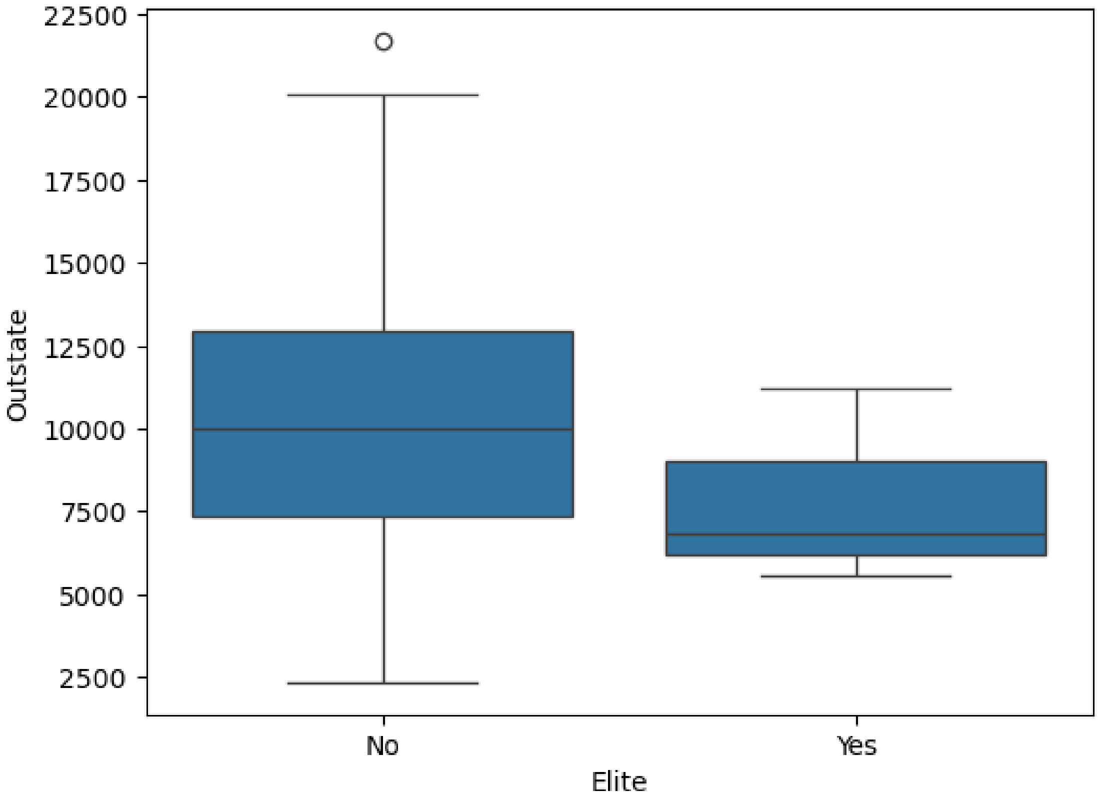
```
Agnes Scott College                    89          510           63     12960
Alaska Pacific University              44          249          869      7560

                              Room.Board  Books  Personal  PhD  Terminal  \
College
Abilene Christian University        3300    450      2200   70        78
Adelphi University                  6450    750      1500   29        30
Adrian College                      3750    400      1165   53        66
Agnes Scott College                 5450    450       875   92        97
Alaska Pacific University           4120    800      1500   76        72

                              S.F.Ratio  perc.alumni  Expend  Grad.Rate Elite
College
Abilene Christian University       18.1           12    7041         60    No
Adelphi University                 12.2           16   10527         56    No
Adrian College                     12.9           30    8735         54    No
Agnes Scott College                 7.7           37   19016         59    No
Alaska Pacific University          11.9            2   10922         15    No
```

[ ]:
```
college['Elite'].value_counts()
```

[ ]:
```
Elite
No     774
Yes      3
Name: count, dtype: int64
```

[ ]:
```
sns.boxplot(x=college['Elite'], y=college['Outstate'])
```

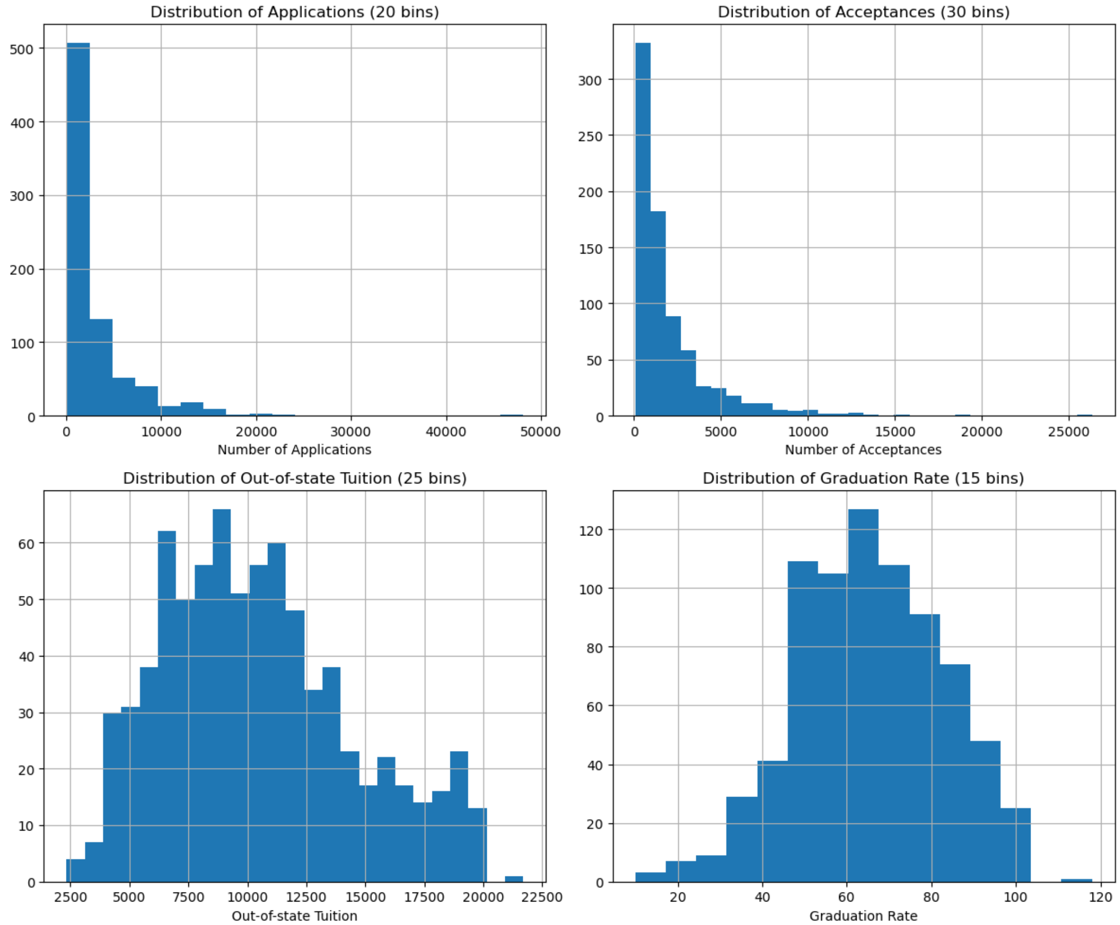
[ ]:
```
<Axes: xlabel='Elite', ylabel='Outstate'>
```



(g) Use the plot.hist() method of college to produce some histograms with differing numbers of bins for a few of the quantitative variables. The command plt.subplots(2, 2) may be useful: it will divide the plot window into four regions so that four plots can be made simultaneously. By changing the arguments you can divide the screen up in other combinations.

[ ]:
```
fig, axes = plt.subplots(2, 2, figsize=(12, 10))

college['Apps'].hist(bins=20, ax=axes[0, 0])
axes[0, 0].set_title('Distribution of Applications (20 bins)')
axes[0, 0].set_xlabel('Number of Applications')

college['Accept'].hist(bins=30, ax=axes[0, 1])
axes[0, 1].set_title('Distribution of Acceptances (30 bins)')
axes[0, 1].set_xlabel('Number of Acceptances')

college['Outstate'].hist(bins=25, ax=axes[1, 0])
axes[1, 0].set_title('Distribution of Out-of-state Tuition (25 bins)')
axes[1, 0].set_xlabel('Out-of-state Tuition')

college['Grad.Rate'].hist(bins=15, ax=axes[1, 1])
axes[1, 1].set_title('Distribution of Graduation Rate (15 bins)')
```

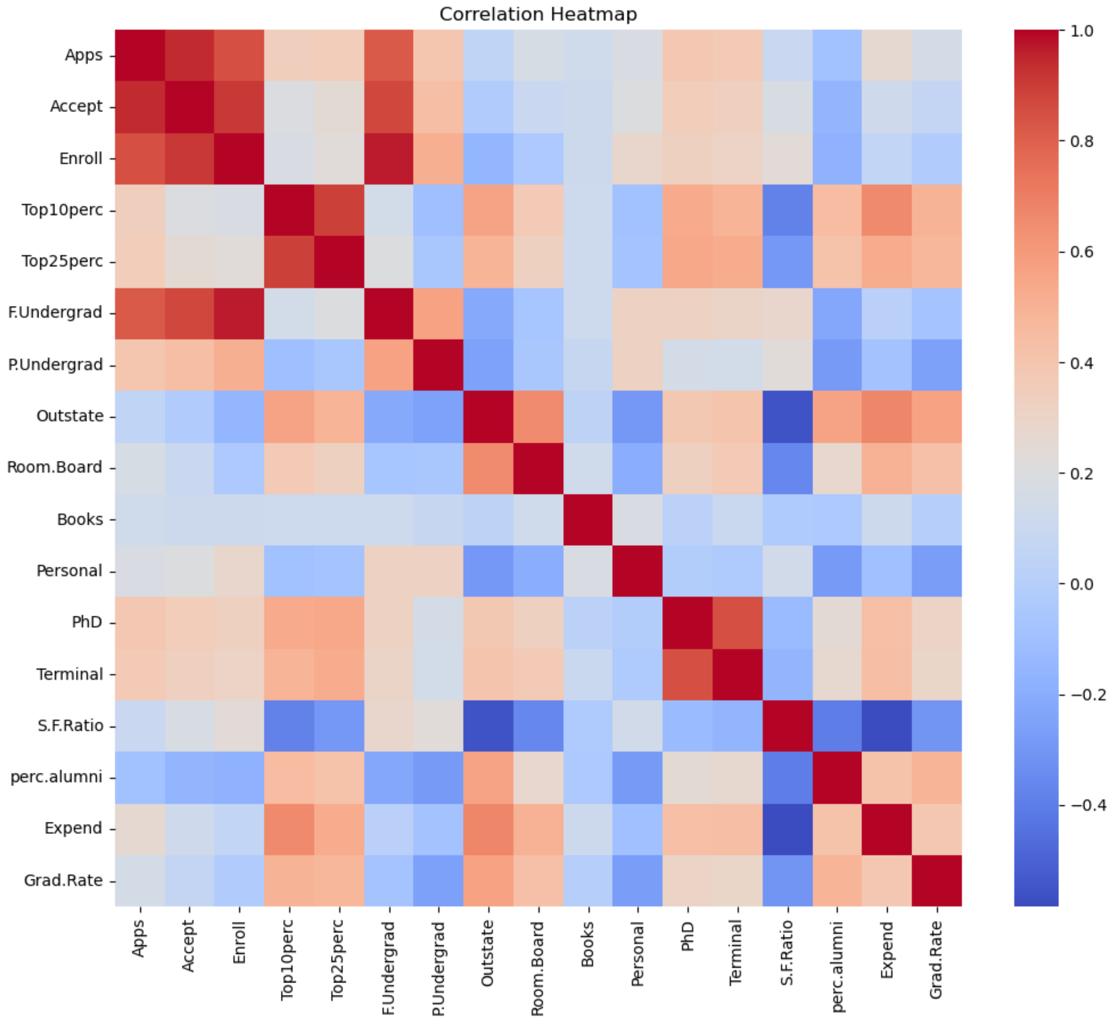
```
axes[1, 1].set_xlabel('Graduation Rate')

plt.tight_layout()
plt.show()
```



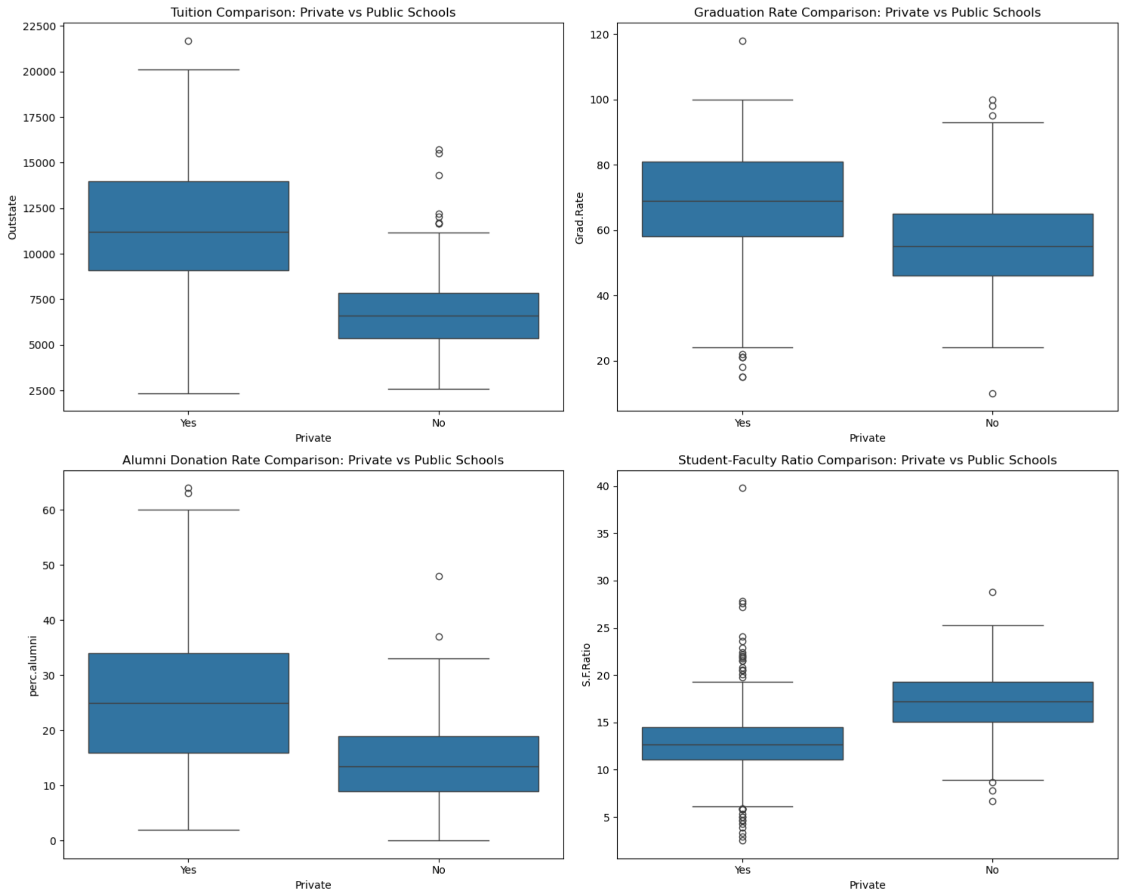
(h) Continue exploring the data, and provide a brief summary of what you discover.

[ ]:
```
# 1. Examine correlations between numerical variables
numeric_columns = college.select_dtypes(include=[np.number])
correlation_matrix = numeric_columns.corr()
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=False, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()

print("Some interesting correlations observed:")
print(correlation_matrix['Outstate'].sort_values(ascending=False).head())
print(correlation_matrix['Grad.Rate'].sort_values(ascending=False).head())
```

```python
# 2. Compare private and public schools
fig, axes = plt.subplots(2, 2, figsize=(15, 12))

sns.boxplot(x='Private', y='Outstate', data=college, ax=axes[0, 0])
axes[0, 0].set_title('Tuition Comparison: Private vs Public Schools')

sns.boxplot(x='Private', y='Grad.Rate', data=college, ax=axes[0, 1])
axes[0, 1].set_title('Graduation Rate Comparison: Private vs Public Schools')

sns.boxplot(x='Private', y='perc.alumni', data=college, ax=axes[1, 0])
axes[1, 0].set_title('Alumni Donation Rate Comparison: Private vs Public␣
↪Schools')

sns.boxplot(x='Private', y='S.F.Ratio', data=college, ax=axes[1, 1])
axes[1, 1].set_title('Student-Faculty Ratio Comparison: Private vs Public␣
↪Schools')

plt.tight_layout()
plt.show()

# 3. Explore potential outliers
print("\nPotential outliers:")
print(college[college['Grad.Rate'] > 100])
```




```
Some interesting correlations observed:
Outstate       1.000000
Expend         0.672779
Room.Board     0.654256
Grad.Rate      0.571290
perc.alumni    0.566262
Name: Outstate, dtype: float64
Grad.Rate      1.000000
Outstate       0.571290
Top10perc      0.494989
perc.alumni    0.490898
Top25perc      0.477281
Name: Grad.Rate, dtype: float64
```




```
Potential outliers:
                  Private  Apps  Accept  Enroll  Top10perc  Top25perc  \
College
Cazenovia College     Yes  3847    3433     527          9         35

                   F.Undergrad  P.Undergrad  Outstate  Room.Board  Books  \
College
Cazenovia College         1010           12      9384        4840    600

                   Personal  PhD  Terminal  S.F.Ratio  perc.alumni  Expend  \
College
Cazenovia College       500   22        47       14.3           20    7697

                   Grad.Rate Elite
College
Cazenovia College        118    No
```

Summary of data exploration: 1. Tuition fees are positively correlated with school reputation indicators (e.g., Top10perc, Top25perc). 2. Private schools generally have higher tuition fees but also higher graduation rates and alumni donation rates compared to public schools. 3. One school was found to have a graduation rate over 100%, which might be a data error. 4. There's a strong positive correlation between tuition fees and school expenditure, suggesting higher fees may reflect more educational resources. 5. Student-faculty ratio is negatively correlated with other quality indicators, implying that lower ratios might lead to better educational quality.

1.0.2 question 10

(a) To begin, load in the Boston data set, which is part of the ISLP library.

[ ]:
```python
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

[ ]:
```python
from ISLP import load_data
boston = load_data('Boston')
boston.head()
```

[ ]:
```
      crim    zn  indus  chas    nox     rm   age     dis  rad  tax  ptratio  \
0  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296     15.3
1  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242     17.8
2  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242     17.8
3  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222     18.7
4  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222     18.7

   lstat  medv
0   4.98  24.0
1   9.14  21.6
2   4.03  34.7
3   2.94  33.4
4   5.33  36.2
```

(b) How many rows are in this data set? How many columns? What do the rows and columns represent?

[ ]:
```python
num_rows, num_columns = boston.shape
print(f"The dataset has {num_rows} rows and {num_columns} columns.")
```

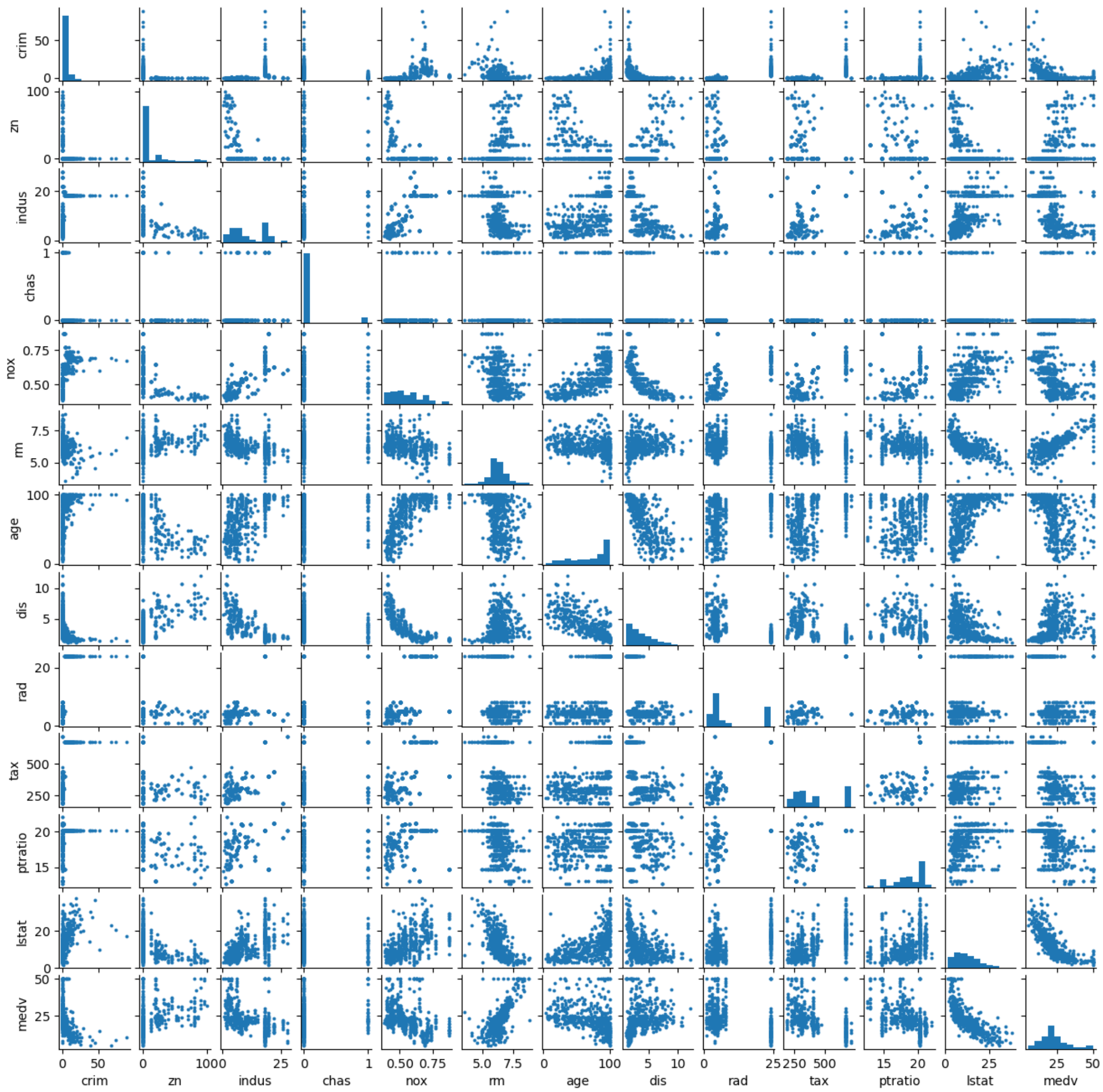
```
The dataset has 506 rows and 13 columns.
```

Each rows is town in Boston area. Columns are features that can influence house price: - crim - zn - indus - chas - nox - rm - age - dis - rad - tax - ptratio - lstat - medv

(c) Make some pairwise scatterplots of the predictors (columns) in this data set. Describe your findings.

[ ]:
```python
g = sns.PairGrid(boston)
g.map_upper(plt.scatter, s=3)
g.map_diag(plt.hist)
```

```python
g.map_lower(plt.scatter, s=3)
g.fig.set_size_inches(12, 12)
```



Based on the scatter plot matrix, we can observe the following:

1. Some variables show clear correlations, for example:

- rm (average number of rooms) is positively correlated with medv (median house value)
- lstat (percentage of lower status population) is negatively correlated with medv
- tax (property tax rate) is positively correlated with rad (accessibility to radial highways)

2. Some variables have skewed distributions, such as crim (per capita crime rate) and tax showing right-skewed distributions
3. Non-linear relationships exist between some variables, like dis (weighted distances to employment centers) and nox (nitric oxides concentration)

4. Some variables like chas (Charles River dummy variable) are binary
5. There are outliers present, especially in the crim and tax variables

(d) Are any of the predictors associated with per capita crime rate? If so, explain the relationship.

[ ]:
```
correlations = boston.corrwith(boston['crim']).sort_values()
correlations
```

[ ]:
```
medv      -0.388305
dis       -0.379670
rm        -0.219247
zn        -0.200469
chas      -0.055892
ptratio    0.289946
age        0.352734
indus      0.406583
nox        0.420972
lstat      0.455621
tax        0.582764
rad        0.625505
crim       1.000000
dtype: float64
```

Based on the correlation analysis results, we can observe the following predictors associated with the per capita crime rate (crim):

1. rad (index of accessibility to radial highways) shows a **strong positive correlation** (0.63) with crim, suggesting that areas with better highway access may have higher crime rates.
2. tax (property tax rate) also exhibits a **strong positive correlation** (0.58) with crim, which might reflect the relationship between socioeconomic conditions in high-tax areas and crime rates.
3. lstat (percentage of lower status population) shows a **moderate positive correlation** (0.46) with crim, indicating that areas with a higher proportion of lower-income residents may face higher crime rates.
4. nox (nitric oxides concentration) and indus (proportion of non-retail business acres) show **moderate positive correlations** (0.42 and 0.41 respectively) with crim, possibly reflecting the relationship between industrialization levels and crime rates.
5. age (proportion of owner-occupied units built prior to 1940) has a **weak positive correlation** (0.35) with crim, suggesting that older neighborhoods might face slightly higher crime rates.
6. dis (weighted distances to employment centers) shows a **weak negative correlation** (-0.38) with crim, indicating that areas farther from employment centers might have slightly lower crime rates.
7. rm (average number of rooms) has a **weak negative correlation** (-0.22) with crim, possibly reflecting that areas with better housing conditions might have slightly lower crime rates.

(e) Do any of the suburbs of Boston appear to have particularly high crime rates? Tax rates? Pupil-teacher ratios? Comment on the range of each predictor.

[ ]:
```
boston.loc[boston['crim'].nlargest(5).index]
```

```
[ ]:        crim   zn  indus  chas    nox     rm    age     dis  rad  tax  \
     380  88.9762  0.0   18.1     0  0.671  6.968   91.9  1.4165   24  666
     418  73.5341  0.0   18.1     0  0.679  5.957  100.0  1.8026   24  666
     405  67.9208  0.0   18.1     0  0.693  5.683  100.0  1.4254   24  666
     410  51.1358  0.0   18.1     0  0.597  5.757  100.0  1.4130   24  666
     414  45.7461  0.0   18.1     0  0.693  4.519  100.0  1.6582   24  666

          ptratio  lstat  medv
     380     20.2  17.21  10.4
     418     20.2  20.62   8.8
     405     20.2  22.98   5.0
     410     20.2  10.11  15.0
     414     20.2  36.98   7.0
```

```
[ ]: boston.loc[boston['tax'].nlargest(5).index]
```

```
[ ]:        crim   zn  indus  chas    nox     rm   age     dis  rad  tax  ptratio  \
     488  0.15086  0.0  27.74     0  0.609  5.454  92.7  1.8209    4  711     20.1
     489  0.18337  0.0  27.74     0  0.609  5.414  98.3  1.7554    4  711     20.1
     490  0.20746  0.0  27.74     0  0.609  5.093  98.0  1.8226    4  711     20.1
     491  0.10574  0.0  27.74     0  0.609  5.983  98.8  1.8681    4  711     20.1
     492  0.11132  0.0  27.74     0  0.609  5.983  83.5  2.1099    4  711     20.1

          lstat  medv
     488  18.06  15.2
     489  23.97   7.0
     490  29.68   8.1
     491  18.07  13.6
     492  13.35  20.1
```

```
[ ]: boston.loc[boston['ptratio'].nlargest(5).index]
```

```
[ ]:        crim    zn  indus  chas    nox     rm   age      dis  rad  tax  \
     354  0.04301  80.0   1.91     0  0.413  5.663  21.9  10.5857    4  334
     355  0.10659  80.0   1.91     0  0.413  5.936  19.5  10.5857    4  334
     127  0.25915   0.0  21.89     0  0.624  5.693  96.0   1.7883    4  437
     128  0.32543   0.0  21.89     0  0.624  6.431  98.8   1.8125    4  437
     129  0.88125   0.0  21.89     0  0.624  5.637  94.7   1.9799    4  437

          ptratio  lstat  medv
     354     22.0   8.05  18.2
     355     22.0   5.57  20.6
     127     21.2  17.19  16.2
     128     21.2  15.39  18.0
     129     21.2  18.34  14.3
```

```
[ ]: boston.describe()
```

```
[ ]:          crim          zn       indus        chas         nox          rm  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean     3.613524   11.363636   11.136779    0.069170    0.554695    6.284634
std      8.601545   23.322453    6.860353    0.253994    0.115878    0.702617
min      0.006320    0.000000    0.460000    0.000000    0.385000    3.561000
25%      0.082045    0.000000    5.190000    0.000000    0.449000    5.885500
50%      0.256510    0.000000    9.690000    0.000000    0.538000    6.208500
75%      3.677083   12.500000   18.100000    0.000000    0.624000    6.623500
max     88.976200  100.000000   27.740000    1.000000    0.871000    8.780000

              age         dis         rad         tax     ptratio       lstat  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean    68.574901    3.795043    9.549407  408.237154   18.455534   12.653063
std     28.148861    2.105710    8.707259  168.537116    2.164946    7.141062
min      2.900000    1.129600    1.000000  187.000000   12.600000    1.730000
25%     45.025000    2.100175    4.000000  279.000000   17.400000    6.950000
50%     77.500000    3.207450    5.000000  330.000000   19.050000   11.360000
75%     94.075000    5.188425   24.000000  666.000000   20.200000   16.955000
max    100.000000   12.126500   24.000000  711.000000   22.000000   37.970000

             medv
count  506.000000
mean    22.532806
std      9.197104
min      5.000000
25%     17.025000
50%     21.200000
75%     25.000000
max     50.000000
```

- Regarding **crime rates**, a few suburbs stand out with particularly high rates, with the highest reaching 88.9762. The crime rate distribution is highly uneven, ranging from 0.00632 to 88.9762, although most suburbs have relatively low crime rates, with a median of 0.25651.
- In terms of **tax rates**, some suburbs have noticeably higher rates, peaking at 711. The tax rate distribution is more uniform, spanning from 187 to 711, with a median of 330, indicating that most suburbs have moderate tax rates.
- **The pupil-teacher ratio** shows less variation, ranging from 12.6 to 22.0, with a median of 19.05, suggesting that most suburbs have relatively similar ratios without any particularly high outliers. Overall, the crime rate exhibits the most significant disparities among the suburbs, while the pupil-teacher ratio shows the least variation. These data reflect considerable socioeconomic differences across Boston's suburbs.

(f) How many of the suburbs in this data set bound the Charles river?

```
[ ]: boston['chas'].value_counts()[1]
```

```
[ ]: 35
```

(g) What is the median pupil-teacher ratio among the towns in this data set?

[ ]:
```
boston['ptratio'].median()
```

[ ]:
```
19.05
```

(h) Which suburb of Boston has lowest median value of owneroccupied homes? What are the values of the other predictors for that suburb, and how do those values compare to the overall ranges for those predictors? Comment on your findings.

[ ]:
```
boston['medv'].idxmin()
```

[ ]:
```
398
```

[ ]:
```
a = boston.describe()
a.loc['range'] = a.loc['max'] - a.loc['min']
a.loc[398] = boston.iloc[398]
a
```

[ ]:
```
             crim          zn       indus        chas         nox          rm  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean     3.613524   11.363636   11.136779    0.069170    0.554695    6.284634
std      8.601545   23.322453    6.860353    0.253994    0.115878    0.702617
min      0.006320    0.000000    0.460000    0.000000    0.385000    3.561000
25%      0.082045    0.000000    5.190000    0.000000    0.449000    5.885500
50%      0.256510    0.000000    9.690000    0.000000    0.538000    6.208500
75%      3.677083   12.500000   18.100000    0.000000    0.624000    6.623500
max     88.976200  100.000000   27.740000    1.000000    0.871000    8.780000
range   88.969880  100.000000   27.280000    1.000000    0.486000    5.219000
398     38.351800    0.000000   18.100000    0.000000    0.693000    5.453000

              age         dis         rad         tax     ptratio       lstat  \
count  506.000000  506.000000  506.000000  506.000000  506.000000  506.000000
mean    68.574901    3.795043    9.549407  408.237154   18.455534   12.653063
std     28.148861    2.105710    8.707259  168.537116    2.164946    7.141062
min      2.900000    1.129600    1.000000  187.000000   12.600000    1.730000
25%     45.025000    2.100175    4.000000  279.000000   17.400000    6.950000
50%     77.500000    3.207450    5.000000  330.000000   19.050000   11.360000
75%     94.075000    5.188425   24.000000  666.000000   20.200000   16.955000
max    100.000000   12.126500   24.000000  711.000000   22.000000   37.970000
range   97.100000   10.996900   23.000000  524.000000    9.400000   36.240000
398    100.000000    1.489600   24.000000  666.000000   20.200000   30.590000

             medv
count  506.000000
mean    22.532806
std      9.197104
min      5.000000
25%     17.025000
50%     21.200000
```

```
75%       25.000000
max       50.000000
range     45.000000
398        5.000000
```

The suburb with the lowest median value is 398. Relative to the other towns, this suburb has high crim, zn below quantile 75%, above mean indus, does not bound the chas, above mean nox, rm below quantile 25%, maximum age, dis near to the minimum value, maximum rad, tax in quantile 75%, ptratio as well and lstat above quantile 75%.

(i) In this data set, how many of the suburbs average more than seven rooms per dwelling? More than eight rooms per dwelling? Comment on the suburbs that average more than eight rooms per dwelling.

[ ]:
```python
# Calculate the number of suburbs with more than 7 rooms per dwelling on average
suburbs_over_7 = (boston['rm'] > 7).sum()

# Calculate the number of suburbs with more than 8 rooms per dwelling on average
suburbs_over_8 = (boston['rm'] > 8).sum()

print(f"Number of suburbs averaging more than 7 rooms per dwelling: 
 ↪{suburbs_over_7}")
print(f"Number of suburbs averaging more than 8 rooms per dwelling: 
 ↪{suburbs_over_8}")

# Get data for suburbs averaging more than 8 rooms per dwelling
suburbs_over_8_data = boston[boston['rm'] > 8]

print("\nCharacteristics of suburbs averaging more than 8 rooms per dwelling:")
print(suburbs_over_8_data.describe())
```

```
Number of suburbs averaging more than 7 rooms per dwelling: 64
Number of suburbs averaging more than 8 rooms per dwelling: 13

Characteristics of suburbs averaging more than 8 rooms per dwelling:
            crim         zn      indus       chas        nox         rm  \
count  13.000000  13.000000  13.000000  13.000000  13.000000  13.000000
mean    0.718795  13.615385   7.078462   0.153846   0.539238   8.348538
std     0.901640  26.298094   5.392767   0.375534   0.092352   0.251261
min     0.020090   0.000000   2.680000   0.000000   0.416100   8.034000
25%     0.331470   0.000000   3.970000   0.000000   0.504000   8.247000
50%     0.520140   0.000000   6.200000   0.000000   0.507000   8.297000
75%     0.578340  20.000000   6.200000   0.000000   0.605000   8.398000
max     3.474280  95.000000  19.580000   1.000000   0.718000   8.780000

             age        dis        rad         tax    ptratio      lstat  \
count  13.000000  13.000000  13.000000   13.000000  13.000000  13.000000
mean   71.538462   3.430192   7.461538  325.076923  16.361538   4.310000
std    24.608723   1.883955   5.332532  110.971063   2.410580   1.373566
```

```
min     8.400000   1.801000   2.000000  224.000000  13.000000   2.470000
25%    70.400000   2.288500   5.000000  264.000000  14.700000   3.320000
50%    78.300000   2.894400   7.000000  307.000000  17.400000   4.140000
75%    86.500000   3.651900   8.000000  307.000000  17.400000   5.120000
max    93.900000   8.906700  24.000000  666.000000  20.200000   7.440000

            medv
count  13.000000
mean   44.200000
std     8.092383
min    21.900000
25%    41.700000
50%    48.300000
75%    50.000000
max    50.000000
```

Based on the analysis results, we can make the following comments about the suburbs with an average of more than 8 rooms per dwelling:

1. Lower crime rate (crim), indicating better safety conditions in these areas.
2. Lower proportion of non-retail business acres (indus), suggesting these areas are primarily residential with less commercial activity.
3. Lower percentage of lower status population (lstat), implying that residents in these suburbs generally have better economic conditions.

2 Chapter 3

2.0.1 question 8

This question involves the use of simple linear regression on the Auto data set.

(a) Use the sm.OLS() function to perform a simple linear regression with mpg as the response and horsepower as the predictor. Use the summarize() function to print the results. Comment on the output. For example:

i. Is there a relationship between the predictor and the response?
ii. How strong is the relationship between the predictor and the response?
iii. Is the relationship between the predictor and the response positive or negative?
iv. What is the predicted mpg associated with a horsepower of 98? What are the associated 95 % confidence and prediction intervals?

[ ]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.formula.api as smf
import statsmodels.api as sm
```

[ ]:
```
# load data
df = pd.read_csv('auto.csv')
df.head()
```

[ ]:
```
    mpg  cylinders  displacement horsepower  weight  acceleration  year  \
0  18.0          8         307.0        130    3504          12.0    70
1  15.0          8         350.0        165    3693          11.5    70
2  18.0          8         318.0        150    3436          11.0    70
3  16.0          8         304.0        150    3433          12.0    70
4  17.0          8         302.0        140    3449          10.5    70

   origin                       name
0       1  chevrolet chevelle malibu
1       1          buick skylark 320
2       1         plymouth satellite
3       1              amc rebel sst
4       1                ford torino
```

[ ]:
```
# prepare data for modelling (training set)
X_train = df['horsepower']
y_train = df['mpg']
```

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

[ ]:
```
0    130
1    165
2    150
3    150
4    140
Name: horsepower, dtype: object
```

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

[ ]:
```
0    18.0
1    15.0
2    18.0
3    16.0
4    17.0
Name: mpg, dtype: float64
```

[ ]:
```
X_train.unique()
```

[ ]:
```
array(['130', '165', '150', '140', '198', '220', '215', '225', '190',
       '170', '160', '95', '97', '85', '88', '46', '87', '90', '113',
       '200', '210', '193', '?', '100', '105', '175', '153', '180', '110',
       '72', '86', '70', '76', '65', '69', '60', '80', '54', '208', '155',
       '112', '92', '145', '137', '158', '167', '94', '107', '230', '49',
       '75', '91', '122', '67', '83', '78', '52', '61', '93', '148',
```

```
       '129', '96', '71', '98', '115', '53', '81', '79', '120', '152',
       '102', '108', '68', '58', '149', '89', '63', '48', '66', '139',
       '103', '125', '133', '138', '135', '142', '77', '62', '132', '84',
       '64', '74', '116', '82'], dtype=object)
```

since there is a strange item – "?", we hope to delete it

```
[ ]: droplist = X_train[X_train == '?'].index
     X_train = X_train.drop(droplist)
     y_train = y_train.drop(droplist)
     X_train.unique()
```

```
[ ]: array(['130', '165', '150', '140', '198', '220', '215', '225', '190',
            '170', '160', '95', '97', '85', '88', '46', '87', '90', '113',
            '200', '210', '193', '100', '105', '175', '153', '180', '110',
            '72', '86', '70', '76', '65', '69', '60', '80', '54', '208', '155',
            '112', '92', '145', '137', '158', '167', '94', '107', '230', '49',
            '75', '91', '122', '67', '83', '78', '52', '61', '93', '148',
            '129', '96', '71', '98', '115', '53', '81', '79', '120', '152',
            '102', '108', '68', '58', '149', '89', '63', '48', '66', '139',
            '103', '125', '133', '138', '135', '142', '77', '62', '132', '84',
            '64', '74', '116', '82'], dtype=object)
```

it's fine now, then check for y

```
[ ]: y_train.unique()
```

```
[ ]: array([18. , 15. , 16. , 17. , 14. , 24. , 22. , 21. , 27. , 26. , 25. ,
            10. , 11. ,  9. , 28. , 19. , 12. , 13. , 23. , 30. , 31. , 35. ,
            20. , 29. , 32. , 33. , 17.5, 15.5, 14.5, 22.5, 24.5, 18.5, 29.5,
            26.5, 16.5, 31.5, 36. , 25.5, 33.5, 20.5, 30.5, 21.5, 43.1, 36.1,
            32.8, 39.4, 19.9, 19.4, 20.2, 19.2, 25.1, 20.6, 20.8, 18.6, 18.1,
            17.7, 27.5, 27.2, 30.9, 21.1, 23.2, 23.8, 23.9, 20.3, 21.6, 16.2,
            19.8, 22.3, 17.6, 18.2, 16.9, 31.9, 34.1, 35.7, 27.4, 25.4, 34.2,
            34.5, 31.8, 37.3, 28.4, 28.8, 26.8, 41.5, 38.1, 32.1, 37.2, 26.4,
            24.3, 19.1, 34.3, 29.8, 31.3, 37. , 32.2, 46.6, 27.9, 40.8, 44.3,
            43.4, 36.4, 44.6, 33.8, 32.7, 23.7, 32.4, 26.6, 25.8, 23.5, 39.1,
            39. , 35.1, 32.3, 37.7, 34.7, 34.4, 29.9, 33.7, 32.9, 31.6, 28.1,
            30.7, 24.2, 22.4, 34. , 38. , 44. ])
```

```
[ ]: d = {'horsepower':X_train.astype('float'), 'mpg':y_train}
     df = pd.DataFrame(data=d)
     df.head()
```

[ ]:

	horsepower	mpg
0	130.0	18.0
1	165.0	15.0
2	150.0	18.0

```
3        150.0  16.0
4        140.0  17.0
```

all fine now, than we use OLS to regress

```
[ ]: mod = smf.ols(formula='mpg ~ horsepower', data = df)
     res = mod.fit()
     print(res.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.606
Model:                            OLS   Adj. R-squared:                  0.605
Method:                 Least Squares   F-statistic:                     599.7
Date:                Fri, 11 Oct 2024   Prob (F-statistic):           7.03e-81
Time:                        00:53:43   Log-Likelihood:                -1178.7
No. Observations:                 392   AIC:                             2361.
Df Residuals:                     390   BIC:                             2369.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     39.9359      0.717     55.660      0.000      38.525      41.347
horsepower    -0.1578      0.006    -24.489      0.000      -0.171      -0.145
==============================================================================
Omnibus:                       16.432   Durbin-Watson:                   0.920
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               17.305
Skew:                           0.492   Prob(JB):                     0.000175
Kurtosis:                       3.299   Cond. No.                         322.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

Analysis of regression results:

i. There is a significant relationship between the predictor variable (horsepower) and the response variable (mpg). The p-value is far less than 0.05, indicating statistical significance.

ii. The R-squared value is 0.606, suggesting that horsepower explains 60.6% of the variation in mpg. This indicates a strong relationship, but other factors also influence mpg.

iii. The coefficient is negative (-0.1578), indicating an inverse relationship between horsepower and mpg. As horsepower increases, mpg decreases.

For iv:

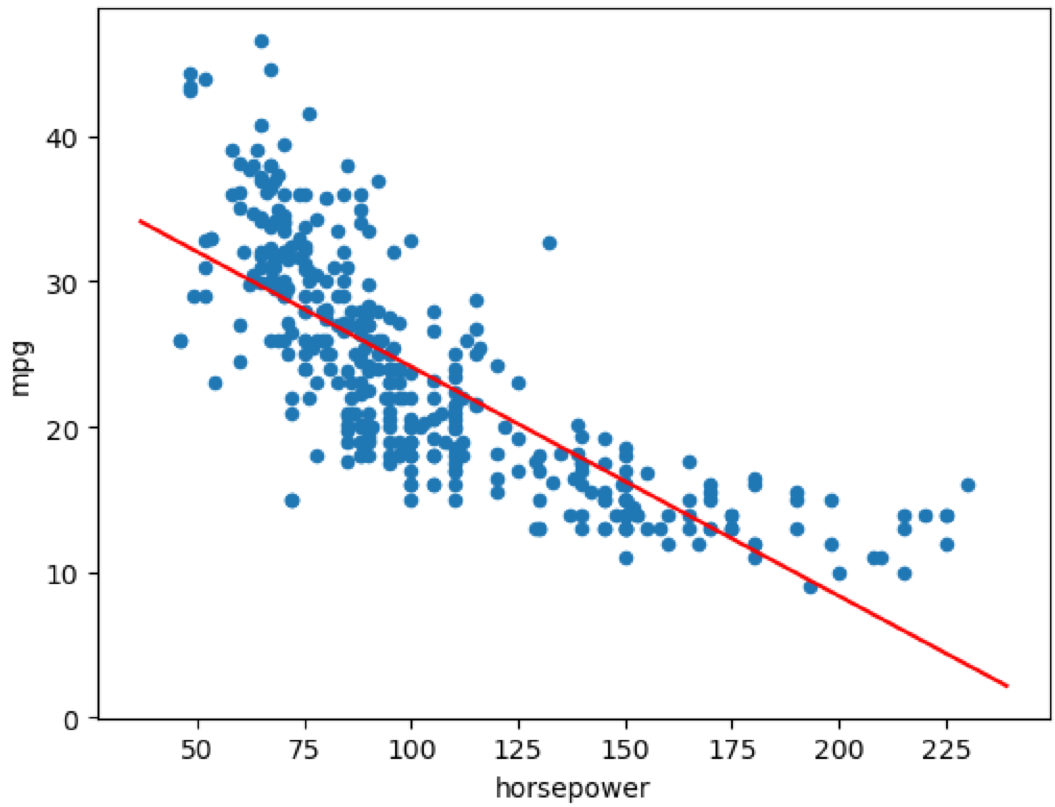
[ ]:
```python
from scipy.stats import t
from math import sqrt

def interval(x, y, x0,alpha = .05):
    n = np.size(x)
    x_bar = np.mean(x)
    y_bar = np.mean(y)
    S_xx = np.sum((x-x_bar)**2)         # page 541
    S_xy = np.sum((x-x_bar)*(y-y_bar))  # page 541
    b = S_xy/S_xx                       # page 542
    a = y_bar - b*x_bar                 # page 542
    S2 = np.sum((y-a-b*x)**2)/(n-2)     # page 552
    S = sqrt(S2)
    ts = t.ppf(1-alpha/2, n-2)
    w_conf = ts*S*sqrt(1/n + (x0-x_bar)**2/S_xx)      # page 558
    w_pred = ts*S*sqrt(1 + 1/n + (x0-x_bar)**2/S_xx)  # page 559
    print("                fit \t lwr \t  upr")
    print("confidence %3.5f %3.5f %3.5f" % (a+b*x0, a+b*x0 - w_conf, a+b*x0 +␣
 ↪w_conf))
    print("prediction %3.5f %3.5f %3.5f" % (a+b*x0, a+b*x0 - w_pred, a+b*x0 +␣
 ↪w_pred))

x = df['horsepower']
y = df['mpg']
x0 = 98

interval(x, y, x0)
```

```
                fit      lwr      upr
confidence 24.46708 23.97308 24.96108
prediction 24.46708 14.80940 34.12476
```

(b). Plot the response and the predictor in a new set of axes ax. Use the ax.axline() method or the abline() function defined in the lab to display the least squares regression line.

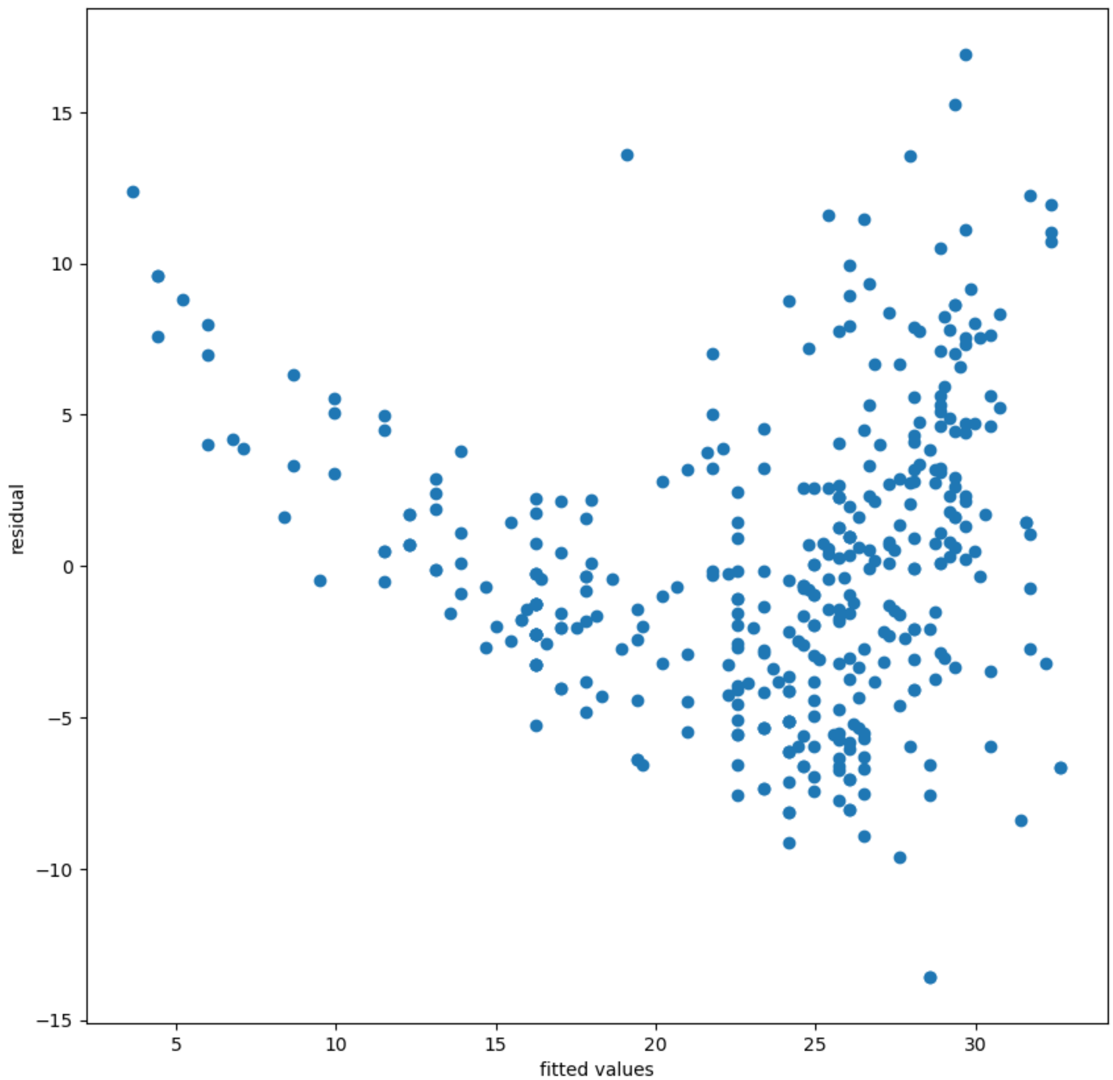
[ ]:
```python
def abline(ax, b, m, *args, **kwargs):
    "Add a line with slope m and intercept b to ax"
    xlim = ax.get_xlim()
    ylim = [m * xlim[0] + b, m * xlim[1] + b]
    ax.plot(xlim, ylim, *args, **kwargs)

ax = df.plot.scatter('horsepower', 'mpg')
abline(ax, res.params[0], res.params[1], 'r')
```

```
/var/folders/vj/7q7lmr4n4cnc5vshb5cgyl1c0000gn/T/ipykernel_19882/723198505.py:8:
FutureWarning: Series.__getitem__ treating keys as positions is deprecated. In a
future version, integer keys will always be treated as labels (consistent with
DataFrame behavior). To access a value by position, use `ser.iloc[pos]`
  abline(ax, res.params[0], res.params[1], 'r')
```



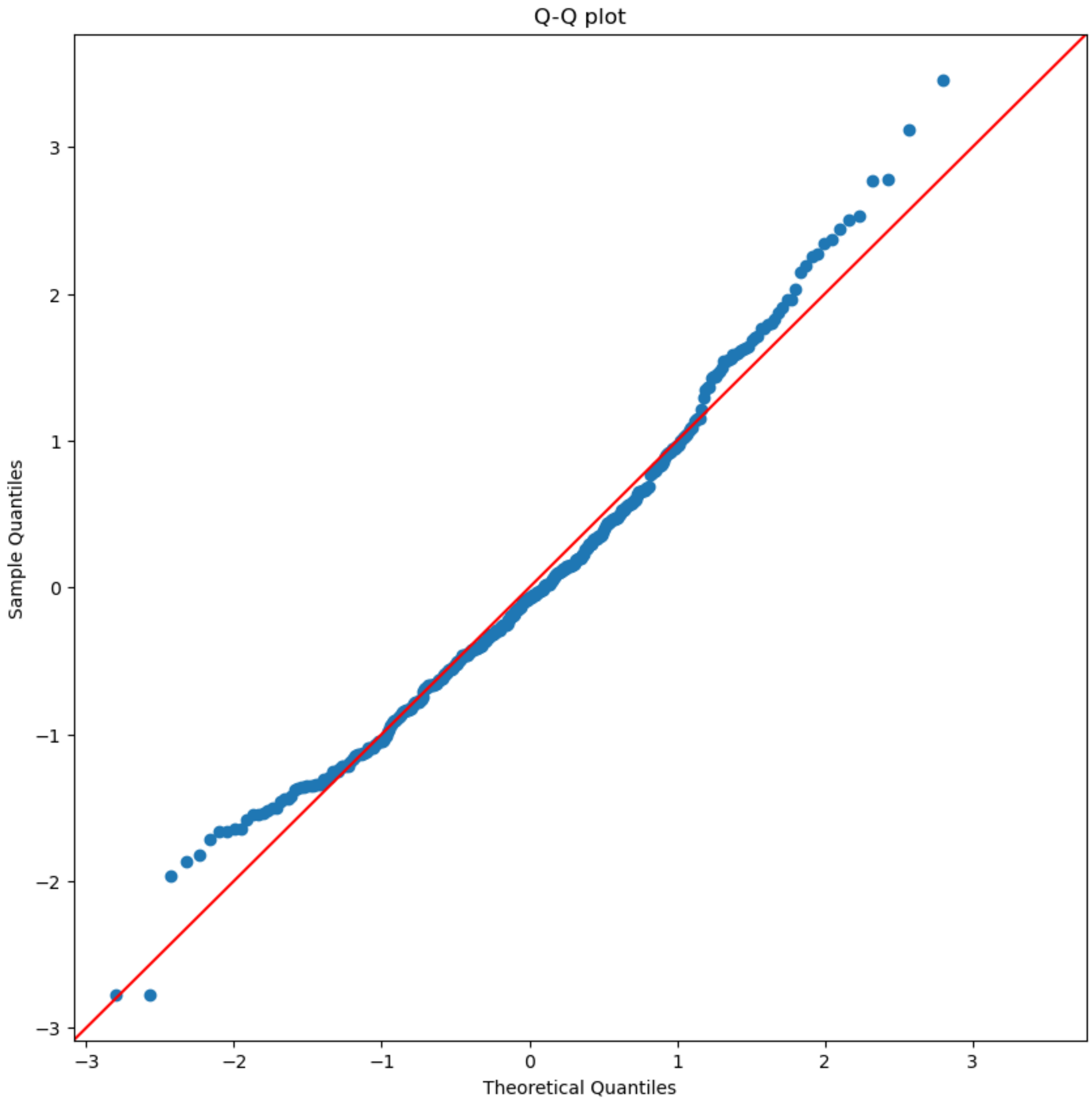
(c) Produce some of diagnostic plots of the least squares regression fit as described in the lab. Comment on any problems you see with the fit.

[ ]:
```
## 1. the scatter plot between fitted values and residual
fig, ax = plt.subplots(figsize=(10,10))
ax.scatter(res.fittedvalues, res.resid)
ax.set_xlabel("fitted values")
ax.set_ylabel("residual")
plt.show()
```



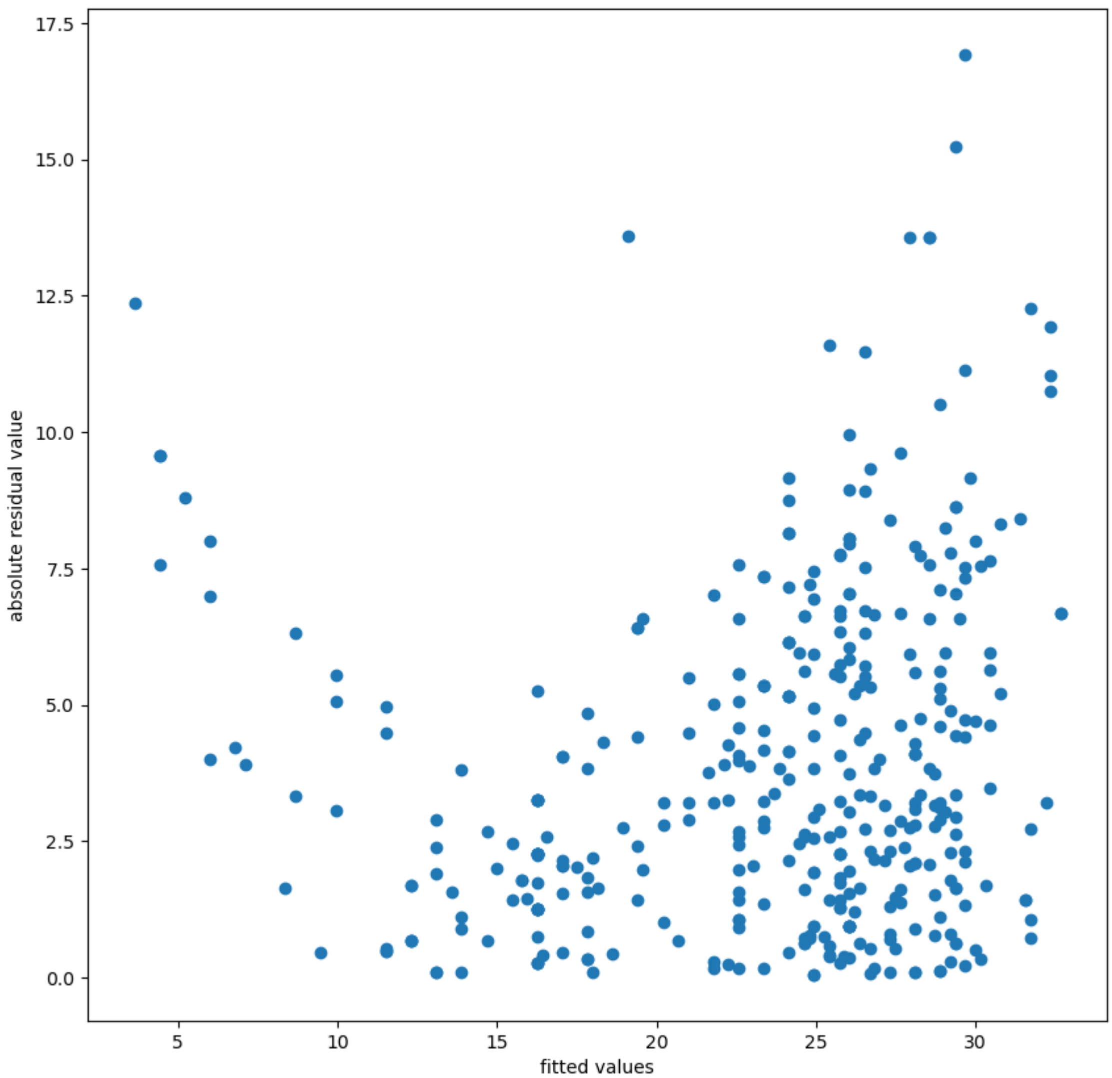
- We can clearly see that the drawn image (residuals vs fitted values) should be non-linear. Also, the funnel shape shown in the image indicates the presence of heteroscedasticity, since the variability of residuals increase with the increase of fitting values.

[ ]:
```
# 2. Q-Q plot
fig, ax = plt.subplots(figsize=(10,10))
sm.qqplot(res.resid, line='45', fit=True, ax=ax)
ax.set_title("Q-Q plot")
plt.show()
```



- The assumption of normality does hold, since the data greatly fit a straight line, although there seems to be a slight left skew.

[ ]:
```
# 3. Scale-Location plot of sqrt(|residuals|) against fitted values
fig, ax = plt.subplots(figsize=(10,10))
ax.scatter(res.fittedvalues, np.abs(res.resid))
ax.set_xlabel("fitted values")
ax.set_ylabel("absolute residual value")
plt.show()
```



- similar to the first graph, which do not take absolute value. It also proves that the assumption of homoscedasticity is not held.

2.0.2 question 9

This question involves the use of multiple linear regression on the Auto data set.

(a) Produce a scatterplot matrix which includes all of the variables in the data set.

[ ]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.formula.api as smf
import statsmodels.api as sm
```

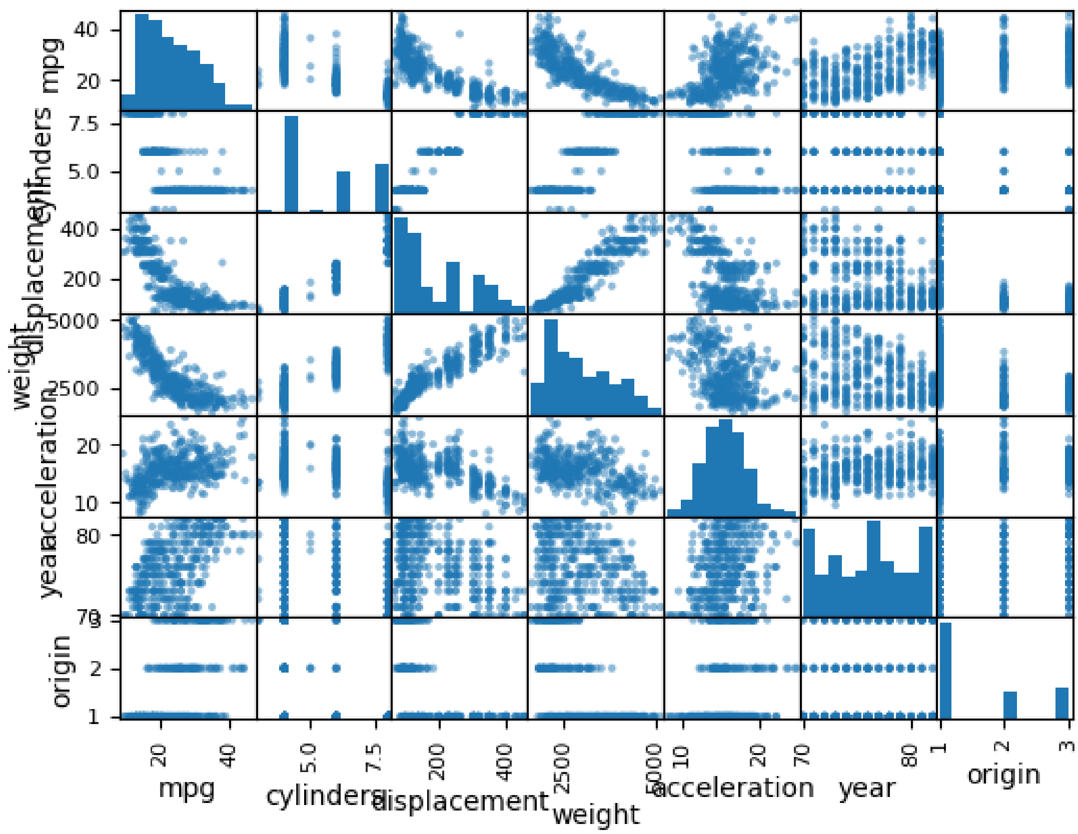
[ ]:
```python
df = pd.read_csv("Auto.csv")
df
```

[ ]:
```
      mpg  cylinders  displacement horsepower  weight  acceleration  year  \
0    18.0          8         307.0        130    3504          12.0    70
1    15.0          8         350.0        165    3693          11.5    70
2    18.0          8         318.0        150    3436          11.0    70
3    16.0          8         304.0        150    3433          12.0    70
4    17.0          8         302.0        140    3449          10.5    70
..    …         …           …         …      …           …    …
392  27.0          4         140.0         86    2790          15.6    82
393  44.0          4          97.0         52    2130          24.6    82
394  32.0          4         135.0         84    2295          11.6    82
395  28.0          4         120.0         79    2625          18.6    82
396  31.0          4         119.0         82    2720          19.4    82

     origin                       name
0         1  chevrolet chevelle malibu
1         1          buick skylark 320
2         1         plymouth satellite
3         1              amc rebel sst
4         1                ford torino
..      …                        …
392       1            ford mustang gl
393       2                  vw pickup
394       1              dodge rampage
395       1                ford ranger
396       1                chevy s-10

[397 rows x 9 columns]
```

[ ]:
```python
pd.plotting.scatter_matrix(df)
df.corr(numeric_only=True)
```

[ ]:
```
                   mpg  cylinders  displacement    weight  acceleration  \
mpg           1.000000  -0.776260     -0.804443 -0.831739      0.422297
cylinders    -0.776260   1.000000      0.950920  0.897017     -0.504061
displacement -0.804443   0.950920      1.000000  0.933104     -0.544162
weight       -0.831739   0.897017      0.933104  1.000000     -0.419502
acceleration  0.422297  -0.504061     -0.544162 -0.419502      1.000000
year          0.581469  -0.346717     -0.369804 -0.307900      0.282901
origin        0.563698  -0.564972     -0.610664 -0.581265      0.210084

                  year    origin
mpg           0.581469  0.563698
cylinders    -0.346717 -0.564972
displacement -0.369804 -0.610664
```

```
weight        -0.307900  -0.581265
acceleration   0.282901   0.210084
year           1.000000   0.184314
origin         0.184314   1.000000
```

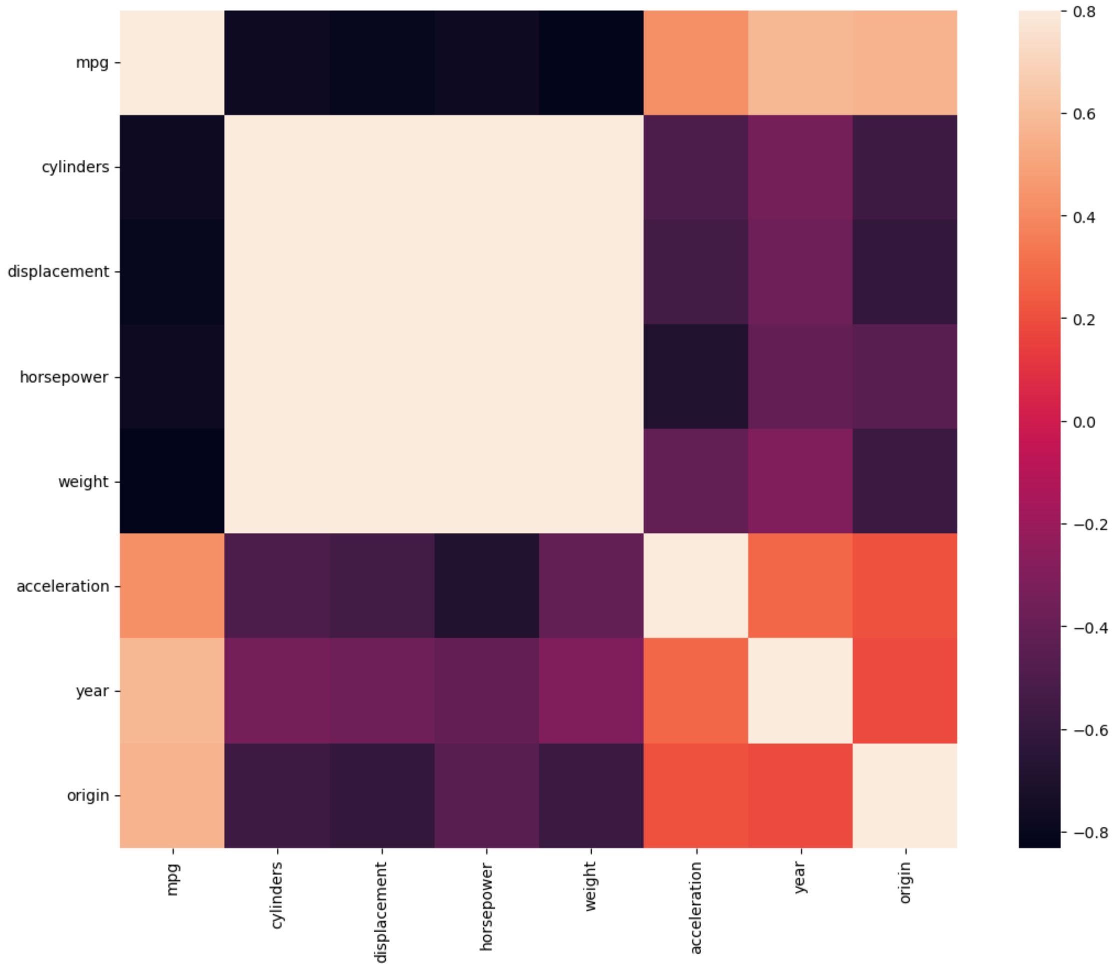


(b) Compute the matrix of correlations between the variables using the DataFrame.corr() method

[ ]:
```
df['horsepower'] = pd.to_numeric(df['horsepower'], errors='coerce')
numeric_columns = df.select_dtypes(include=[np.number]).columns
corrmat = df[numeric_columns].corr()
print(corrmat)
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, square=True)
f.tight_layout()
```

```
                   mpg  cylinders  displacement  horsepower    weight  \
mpg           1.000000  -0.776260     -0.804443   -0.778427 -0.831739
cylinders    -0.776260   1.000000      0.950920    0.842983  0.897017
displacement -0.804443   0.950920      1.000000    0.897257  0.933104
horsepower   -0.778427   0.842983      0.897257    1.000000  0.864538
weight       -0.831739   0.897017      0.933104    0.864538  1.000000
```

```
acceleration  0.422297  -0.504061     -0.544162   -0.689196 -0.419502
year          0.581469  -0.346717     -0.369804   -0.416361 -0.307900
origin        0.563698  -0.564972     -0.610664   -0.455171 -0.581265

              acceleration      year    origin
mpg               0.422297  0.581469  0.563698
cylinders        -0.504061 -0.346717 -0.564972
displacement     -0.544162 -0.369804 -0.610664
horsepower       -0.689196 -0.416361 -0.455171
weight           -0.419502 -0.307900 -0.581265
acceleration      1.000000  0.282901  0.210084
year              0.282901  1.000000  0.184314
origin            0.210084  0.184314  1.000000
```



(c) Use the sm.OLS() function to perform a multiple linear regression with mpg as the response and all other variables except name as the predictors. Use the summarize() function to print the results. Comment on the output. For instance:

i. Is there a relationship between the predictors and the response? Use the anova_lm() function from statsmodels to answer this question.
ii. Which predictors appear to have a statistically significant relationship to the response?
iii. What does the coefficient for the year variable suggest?

[ ]:
```
reg = smf.ols('mpg ~ cylinders + displacement + horsepower + weight +␣
↪acceleration + year + origin', df).fit()
reg.summary()
```

[ ]:

Dep. Variable:	mpg	**R-squared:**	0.821
Model:	OLS	**Adj. R-squared:**	0.818
Method:	Least Squares	**F-statistic:**	252.4
Date:	Fri, 11 Oct 2024	**Prob (F-statistic):**	2.04e-139
Time:	01:17:00	**Log-Likelihood:**	-1023.5
No. Observations:	392	**AIC:**	2063.
Df Residuals:	384	**BIC:**	2095.
Df Model:	7		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
Intercept	-17.2184	4.644	-3.707	0.000	-26.350	-8.087
cylinders	-0.4934	0.323	-1.526	0.128	-1.129	0.142
displacement	0.0199	0.008	2.647	0.008	0.005	0.035
horsepower	-0.0170	0.014	-1.230	0.220	-0.044	0.010
weight	-0.0065	0.001	-9.929	0.000	-0.008	-0.005
acceleration	0.0806	0.099	0.815	0.415	-0.114	0.275
year	0.7508	0.051	14.729	0.000	0.651	0.851
origin	1.4261	0.278	5.127	0.000	0.879	1.973

Omnibus:	31.906	**Durbin-Watson:**	1.309
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	53.100
Skew:	0.529	**Prob(JB):**	2.95e-12
Kurtosis:	4.460	**Cond. No.**	8.59e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 8.59e+04. This might indicate that there are strong multicollinearity or other numerical problems.

Based on the regression results:

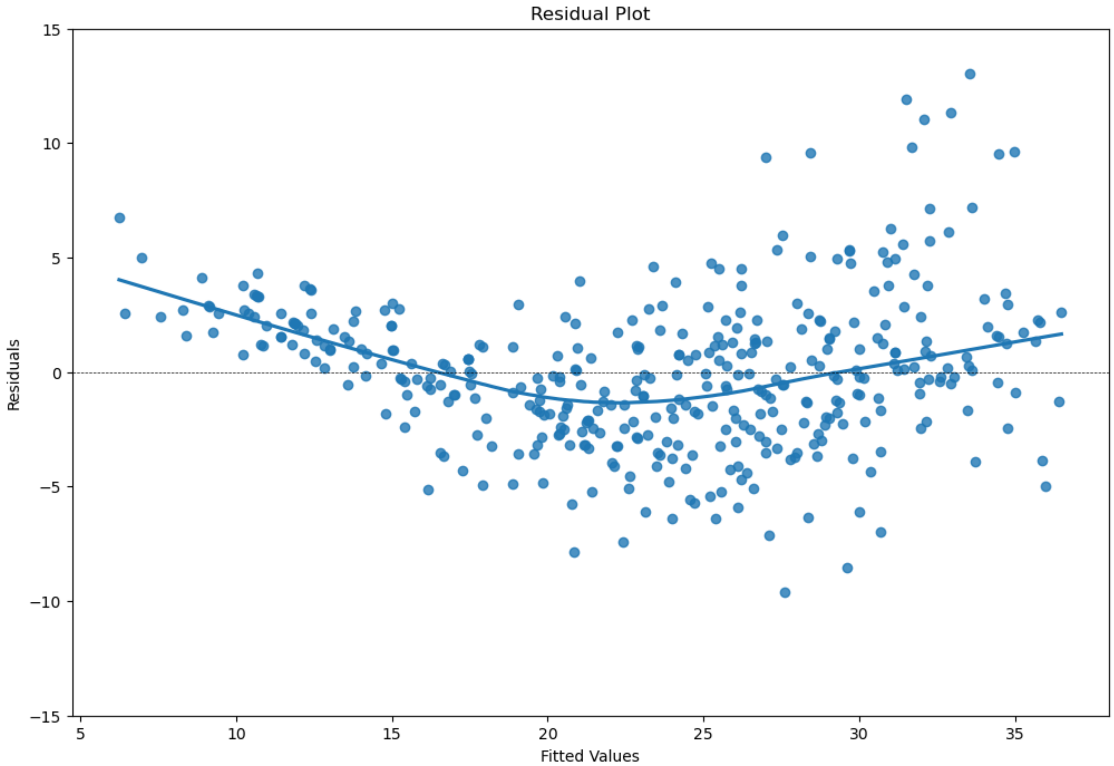
i. There appears to be a significant relationship between the predictors and the response (mpg). The F-statistic is large (252.4) with a very low p-value, indicating the model as a whole is statistically significant.

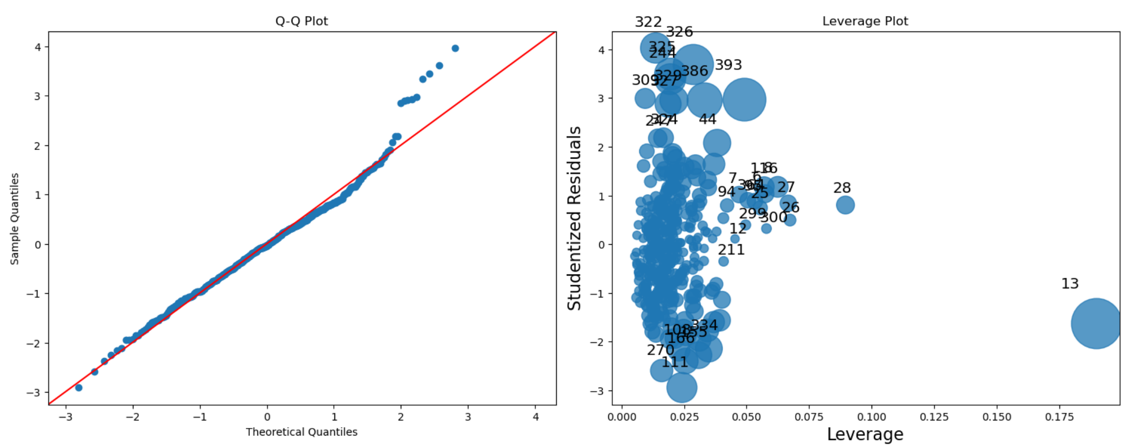
ii. Several predictors show statistically significant relationships with mpg, including weight, year, and origin (p-values < 0.05). Displacement also appears significant. Cylinders, horsepower, and acceleration do not show strong statistical significance in this model.

iii. The coefficient for the year variable (0.7508) suggests that, on average, for each year increase, the mpg increases by about 0.75, holding other variables constant. This indicates a trend of improving fuel efficiency over time.

(d) Produce some of diagnostic plots of the linear regression fit as described in the lab. Comment on any problems you see with the fit. Do the residual plots suggest any unusually large outliers? Does the leverage plot identify any observations with unusually high leverage?

[ ]:
```
# Diagnostic plots
# Distribution of residuals
plt.figure(figsize=(12, 8))
plt.ylim(-15, 15)
sns.regplot(x=reg.fittedvalues, y=reg.resid, lowess=True)
plt.axhline(y=0, linewidth=0.5, linestyle='dashed', color='black')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.title('Residual Plot')

# Add Q-Q plot
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 6))
sm.qqplot(reg.resid, line='45', fit=True, ax=ax1)
ax1.set_title('Q-Q Plot')

# Add leverage plot
sm.graphics.influence_plot(reg, ax=ax2, criterion="cooks")
ax2.set_title('Leverage Plot')

plt.tight_layout()
plt.show()
```





Based on the diagnostic plots, we can observe the following issues:

1. **Residual plot:**
   - The residuals are not completely randomly distributed around the zero line, showing some pattern. This suggests possible non-linear relationships not captured by the model.
   - The variance of residuals seems to increase with fitted values, potentially violating the homoscedasticity assumption.
   - Several points have residuals that deviate significantly from others, possibly indicating

outliers.
2. **Q-Q plot:**
   - Most points are close to the 45-degree line, but there are noticeable deviations at both ends. This indicates that the distribution of residuals may have some skewness or heavy tails, not fully conforming to the normality assumption.
3. **Leverage plot:**
   - A few observations have relatively large Cook's distances (>0.5), indicating they have substantial influence on the model fit.
   - Some points in the upper right corner have both high leverage values and large residuals, which could be influential outliers.

(e) Fit some models with interactions as described in the lab. Do any interactions appear to be statistically significant?

[ ]:
```
predictors = ' + '.join(df.columns.difference(['name', 'mpg']))
model = smf.ols(formula=f'mpg ~ {predictors} + horsepower*cylinders +␣
↪horsepower*year', data=df)
results = model.fit()
print(results.summary())

interactions = results.pvalues[['horsepower:cylinders', 'horsepower:year']]
significant_interactions = interactions[interactions < 0.05]
significant_interactions
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.870
Model:                            OLS   Adj. R-squared:                  0.867
Method:                 Least Squares   F-statistic:                     283.1
Date:                Fri, 11 Oct 2024   Prob (F-statistic):          5.43e-163
Time:                        01:26:01   Log-Likelihood:                -961.89
No. Observations:                 392   AIC:                             1944.
Df Residuals:                     382   BIC:                             1983.
Df Model:                           9
Covariance Type:            nonrobust
================================================================================
========
                           coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--------
Intercept              -40.8773     12.176     -3.357      0.001     -64.817
-16.937
acceleration            -0.1742      0.088     -1.985      0.048      -0.347
-0.002
cylinders               -3.0837      0.516     -5.971      0.000      -4.099
-2.068
displacement            -0.0048      0.007     -0.713      0.476      -0.018
```

```
0.008
horsepower              0.2260      0.119      1.897      0.059     -0.008
0.460
origin                  0.8834      0.243      3.634      0.000      0.405
1.361
weight                 -0.0038      0.001     -6.294      0.000     -0.005
-0.003
year                    1.3597      0.139      9.771      0.000      1.086
1.633
horsepower:cylinders    0.0308      0.004      7.336      0.000      0.023
0.039
horsepower:year        -0.0065      0.001     -4.696      0.000     -0.009
-0.004
==============================================================================
Omnibus:                       39.241   Durbin-Watson:                   1.647
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               72.217
Skew:                           0.596   Prob(JB):                     2.08e-16
Kurtosis:                       4.732   Cond. No.                     7.54e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 7.54e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[ ]:
```
horsepower:cylinders    1.331914e-12
horsepower:year         3.710540e-06
dtype: float64
```

Based on the results, we can observe that:

1. The interaction between horsepower and cylinders is statistically significant (p-value < 0.05).
2. The interaction between horsepower and year is also statistically significant (p-value < 0.05).

These significant interactions suggest that: 1. The effect of horsepower on mpg varies depending on the number of cylinders. 2. The effect of horsepower on mpg changes over different years.

(f) Try a few different transformations of the variables, such as log(X), 'X, X2. Comment on your findings

[ ]:
```
predictors = ' + '.join(df.columns.difference(['name','mpg']))
result = smf.ols('mpg ~ {} + horsepower*cylinders + np.power(horsepower,2)'.
 ↪format(predictors),data = df).fit()
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.863
Model:                            OLS   Adj. R-squared:                  0.860
```

```
Method:                 Least Squares   F-statistic:                     267.4
Date:                Fri, 11 Oct 2024   Prob (F-statistic):          6.47e-159
Time:                        01:29:28   Log-Likelihood:                -971.55
No. Observations:                 392   AIC:                             1963.
Df Residuals:                     382   BIC:                             2003.
Df Model:                           9
Covariance Type:            nonrobust
================================================================================
===========
                             coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-----------
Intercept                 10.7462      4.937      2.177      0.030       1.040
20.453
acceleration              -0.2361      0.099     -2.392      0.017      -0.430
-0.042
cylinders                 -3.2075      0.818     -3.920      0.000      -4.816
-1.599
displacement              -0.0048      0.007     -0.663      0.508      -0.019
0.009
horsepower                -0.3386      0.034    -10.048      0.000      -0.405
-0.272
origin                     0.8978      0.249      3.603      0.000       0.408
1.388
weight                    -0.0035      0.001     -5.310      0.000      -0.005
-0.002
year                       0.7373      0.045     16.459      0.000       0.649
0.825
horsepower:cylinders       0.0312      0.007      4.664      0.000       0.018
0.044
np.power(horsepower, 2)    0.0003      0.000      1.619      0.106   -6.39e-05
0.001
==============================================================================
Omnibus:                       39.733   Durbin-Watson:                   1.608
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               72.555
Skew:                           0.605   Prob(JB):                     1.76e-16
Kurtosis:                       4.725   Cond. No.                     5.38e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 5.38e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

We observed in the chapter that the scatterplot between horsepower and mpg suggested a non-linear relationship. By incorporating a new term, horsepower squared, we noticed an improvement in the R-squared value, which increased from 82.1% to 86.3%. This indicates that the quadratic transformation of horsepower provides a better fit for the data and explains more of the variance in mpg compared to the linear model.

Additionally, as mentioned in the instructions, we can explore other transformations such as log(X) or square root of X for different variables. These transformations might reveal further insights into the relationships between the predictors and mpg, potentially improving the model's performance and our understanding of the underlying patterns in the data.