hw1

September 22, 2023

1 Applied Problems

Jack Krebsbach Stats 313 9/16/23

1.0.1 #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 numpy as np
import pandas as pd
college = pd.read_csv("./data/College.csv")
```

[ ]:
```
# The first column is not named.
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) We don't need the name of the college to be treated as data, but we want to have that for later, so we take it as the index.

[ ]:
```python
college2 = pd.read_csv('./data/College.csv', index_col=0)
college3 = college.rename(columns = {'Unnamed: 0': 'College'})
college3
```

[ ]:
```
                          College 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]
```

[ ]:
```python
# We keep the original data
college = college3
college
```

[ ]:
```
                            College 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]
```

(c) We can look at the data by using describe.

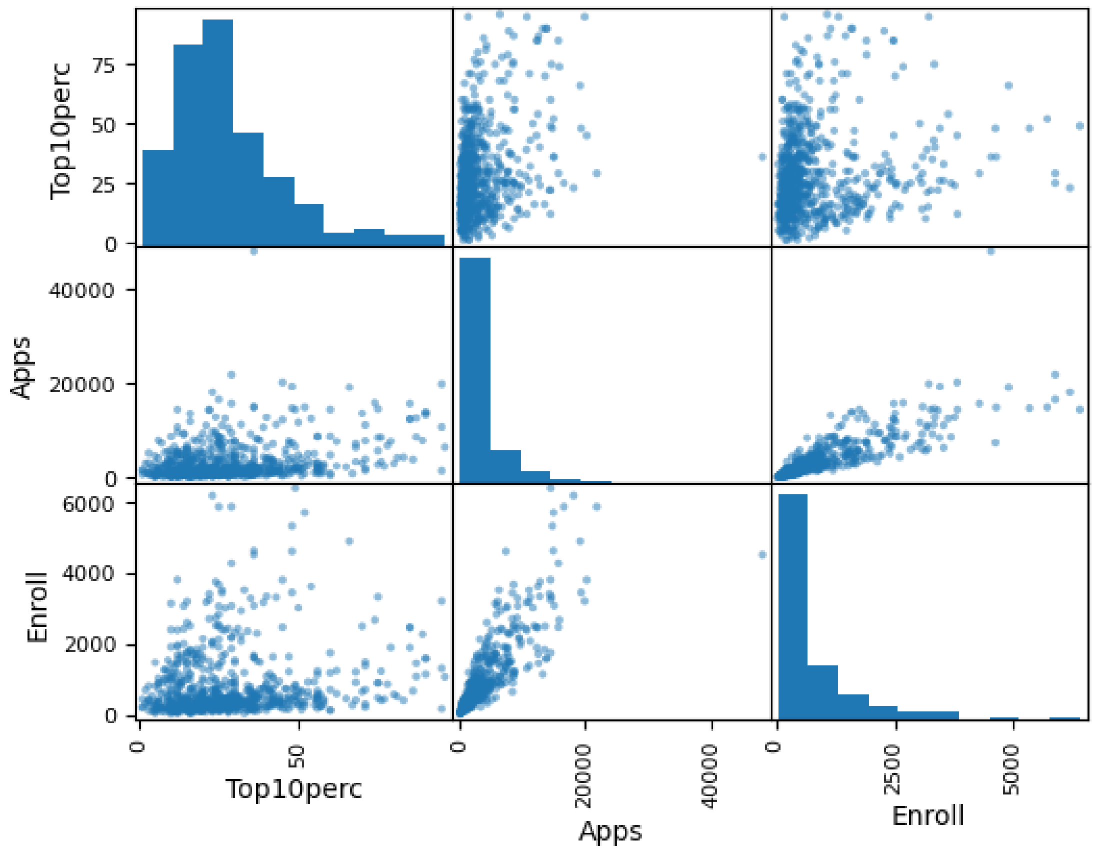
[ ]:
```python
college.describe()
```

[ ]:
```
               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
```

[ ]:
```python
#Compare the data to each other using scatter matrix
pd.plotting.scatter_matrix(college[['Top10perc', 'Apps', 'Enroll']])
```

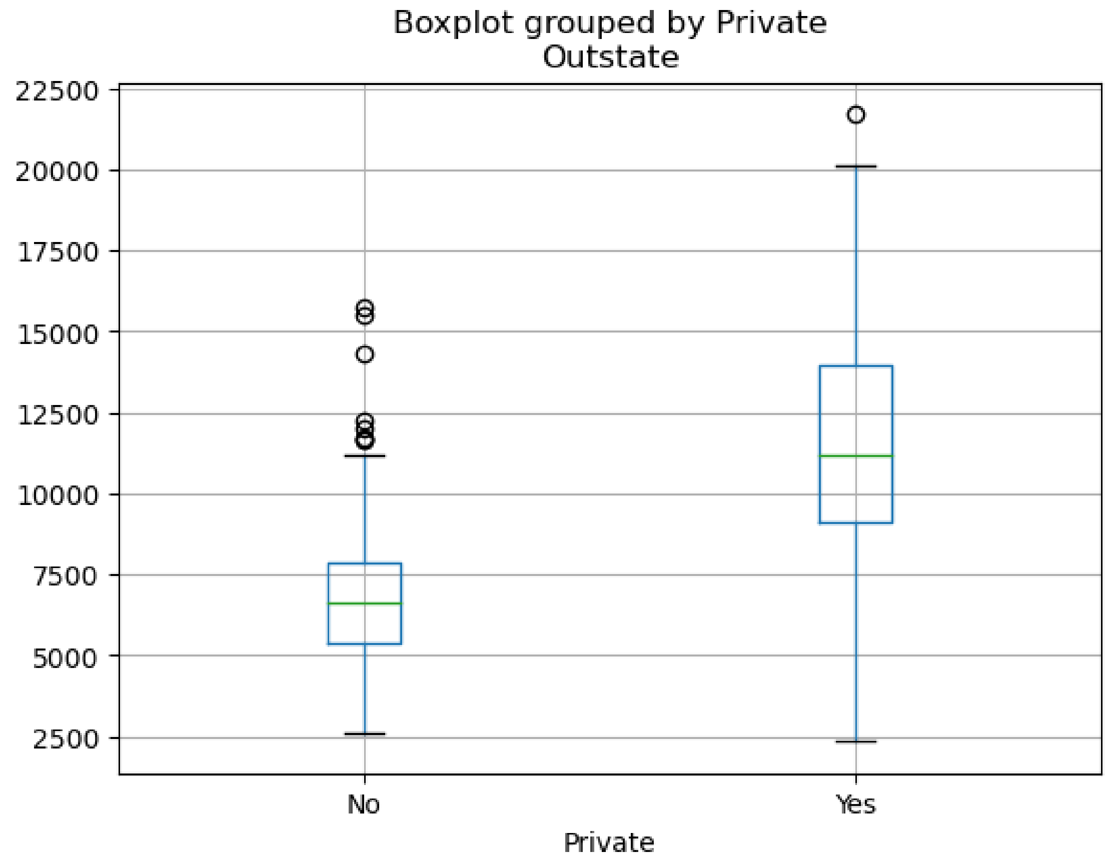
[ ]:
```
array([[<Axes: xlabel='Top10perc', ylabel='Top10perc'>,
        <Axes: xlabel='Apps', ylabel='Top10perc'>,
        <Axes: xlabel='Enroll', ylabel='Top10perc'>],
       [<Axes: xlabel='Top10perc', ylabel='Apps'>,
        <Axes: xlabel='Apps', ylabel='Apps'>,
        <Axes: xlabel='Enroll', ylabel='Apps'>],
       [<Axes: xlabel='Top10perc', ylabel='Enroll'>,
        <Axes: xlabel='Apps', ylabel='Enroll'>,
        <Axes: xlabel='Enroll', ylabel='Enroll'>]], dtype=object)
```



1.0.2 (e)

Private colleges appear to have a higher median number of out-of-state students. They also have a higher spread of number of students out of state compared to non-private schools

[ ]:
```
# Create a boxplot of Outstate vs Private.
college.boxplot('Outstate', by='Private');
```



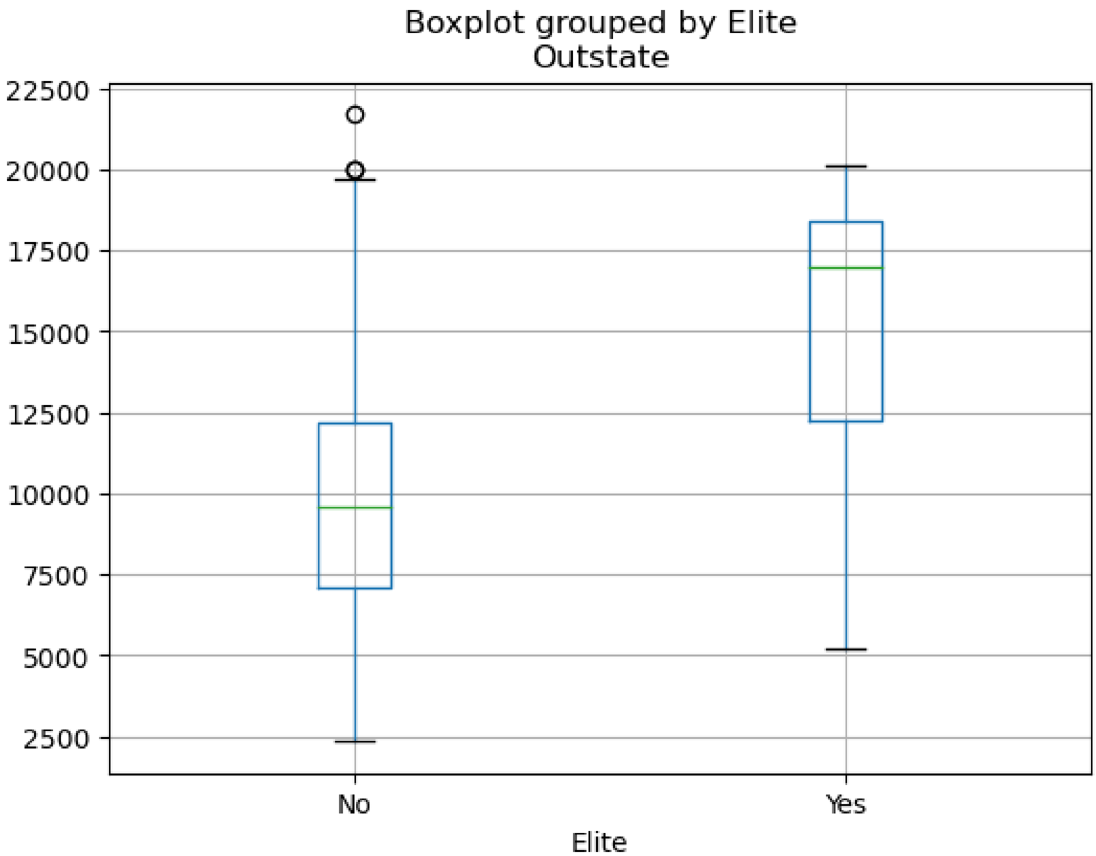
(f) There are 78 elite universities

```
[ ]: college['Elite'] = pd.cut(college['Top10perc'], [0,50,100], labels=['No',␣
       ↪'Yes'])
     print(college.Elite[college.Elite == 'Yes'].shape)
```

```
(78,)
```

```
[ ]: college.boxplot('Outstate', by='Elite')
```

```
[ ]: <Axes: title={'center': 'Outstate'}, xlabel='Elite'>
```

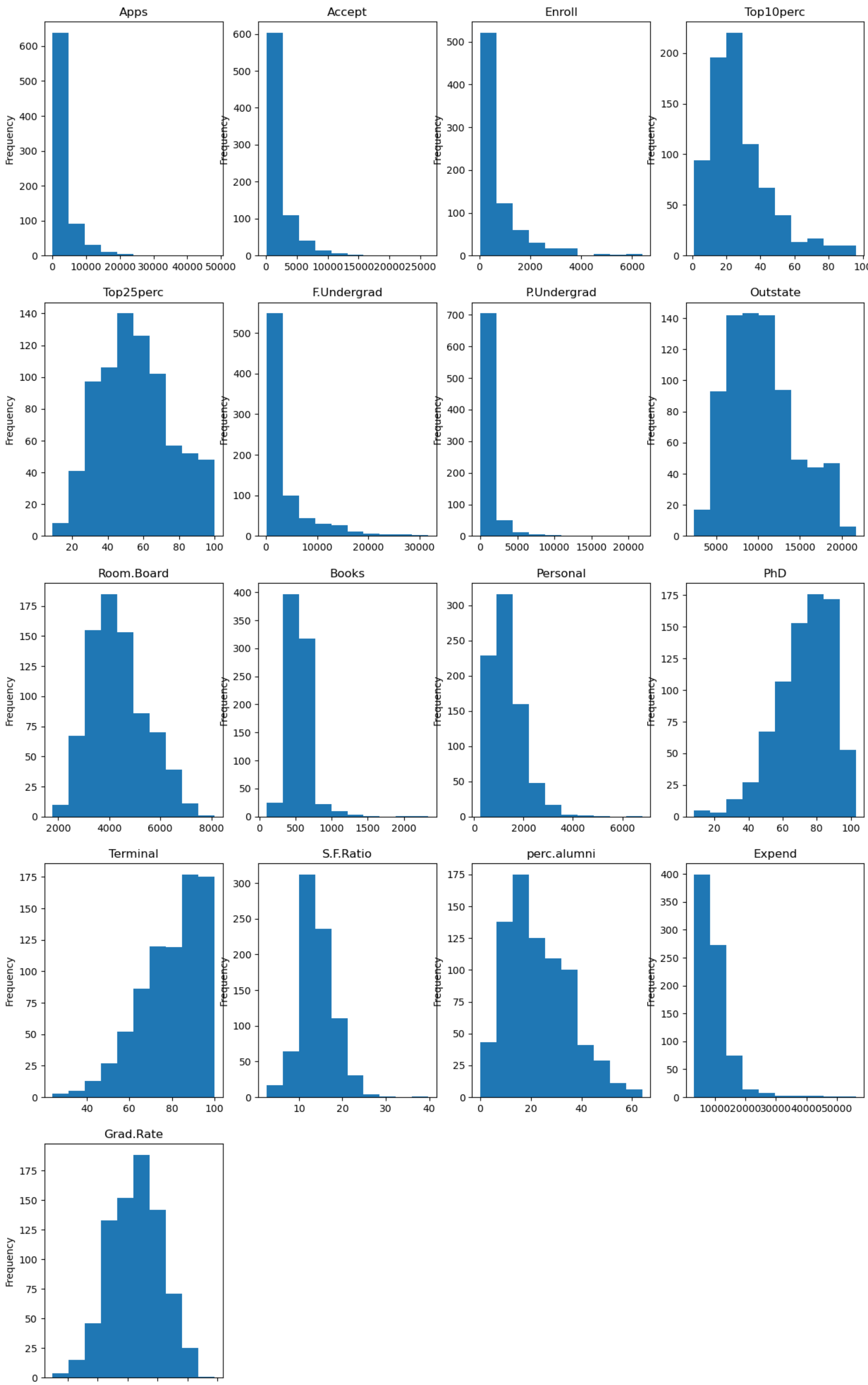


1.0.3 (g)

Exploring the histograms of different features of the data. If we are plotting histograms we want numerical data so we can bin each feature and look at the frequency counts.

```
[ ]: from matplotlib.pyplot import subplots
     # Let's make histograms of all the numerical columns
     column_names = college.select_dtypes(include='number').columns
     # Number of histograms we want
     number_of_histograms = column_names.size
     # Number of rows
     num_rows = number_of_histograms // 4 + (number_of_histograms % 4 > 0)  # Round␣
      ↪up to the nearest integer

     # Make the subplots
     fig, axes = subplots(nrows=num_rows, ncols=4, figsize=(15, 5 * num_rows))

     # Append the histograms to subplots
     for i, column_name in enumerate(column_names):
         ax = axes[i // 4, i % 4]  # Select the subplot
```

```python
    college[column_name].plot(kind='hist', ax=ax)
    ax.set_title(column_name)

# Remove any unused subplots
for i in range(number_of_histograms, num_rows * 4):
    fig.delaxes(axes[i // 4, i % 4])
```


Apps
Accept
Enroll
Top10perc
Top25perc
F.Undergrad
P.Undergrad
Outstate
Room.Board
Books
Personal
PhD
Terminal
S.F.Ratio
perc.alumni
Expend
Grad.Rate
Frequency

1.0.4 (h)

Continue to explore the data. We see that the graduation rate is normally distributed across colleges. Peculiarly it seems that some colleges have above 100% graduation rate which does not make sense.

Most faculty members at these colleges have a Phd, we see the Phd histogram is skewed to the left. The percentage of students who graduated in the top 25% percent of their class also normally distributed. The histograms for the number of applicants received, rejected, and enrolled are consistent and skewed to the right.

1.0.5 #10

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

[ ]:
```
        crim    zn  indus  chas    nox     rm   age     dis  rad  tax  \
0    0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296
1    0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242
2    0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242
3    0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222
4    0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222
..       …     …     …     …     …     …    …    …     …
501  0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273
502  0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273
503  0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273
504  0.10959   0.0  11.93     0  0.573  6.794  89.3  2.3889    1  273
505  0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273

     ptratio  lstat  medv
0       15.3   4.98  24.0
1       17.8   9.14  21.6
2       17.8   4.03  34.7
3       18.7   2.94  33.4
4       18.7   5.33  36.2
..       …      …    …
501     21.0   9.67  22.4
502     21.0   9.08  20.6
503     21.0   5.64  23.9
504     21.0   6.48  22.0
505     21.0   7.88  11.9

[506 rows x 13 columns]
```

1.0.6 (b)

There are 506 rows and 13 columns. The rows represent housing values of suburbs in boston, and the columns represent various characteristics about the suburb. For example, we have the average number of rooms per dwelling, the zone within the city, and crime rate per capita.

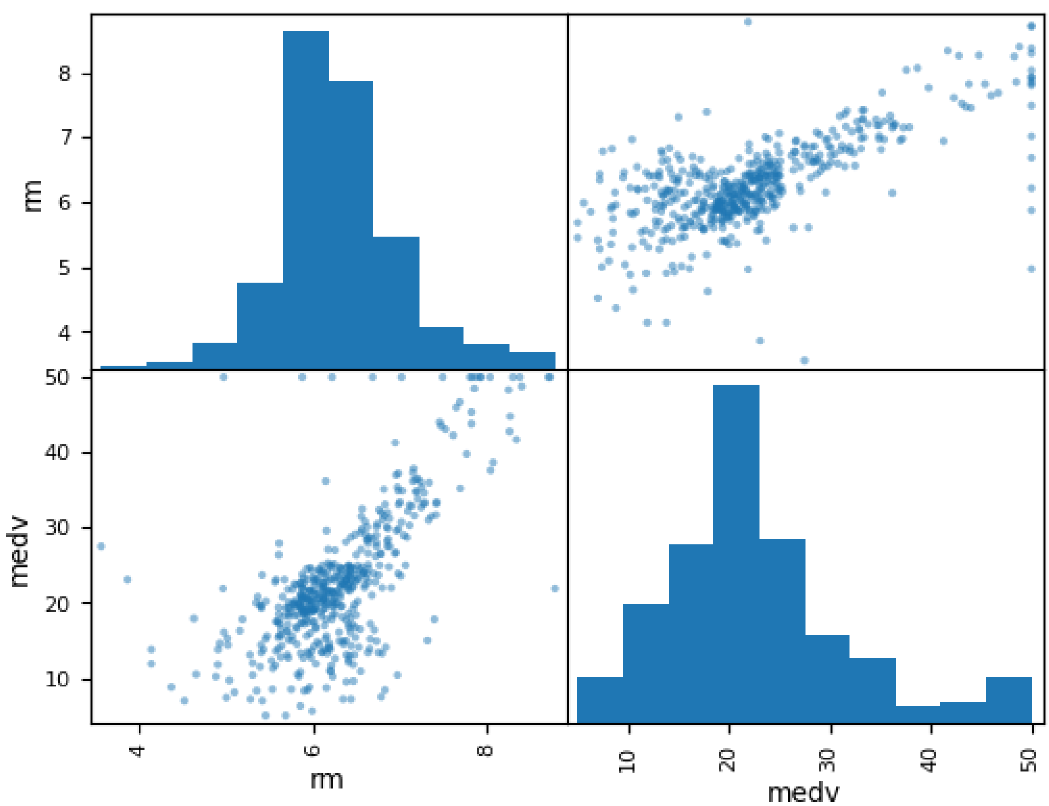
1.0.7 (c)

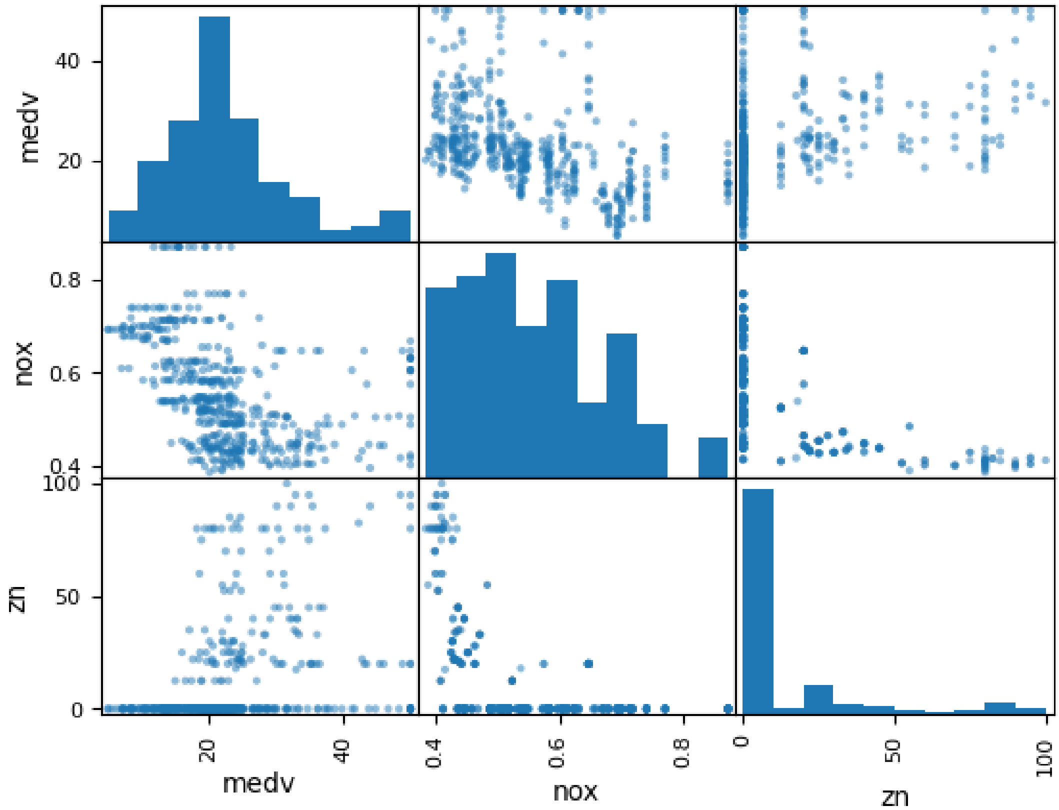
We see that there is a strong correlation between Zone and median value of home in $1000s. This makes sense as the more rooms you have the higher the home is likely to be in value. However, it is peculiar that the nitrgoen oxygen concentrate is corellated with median value of home.

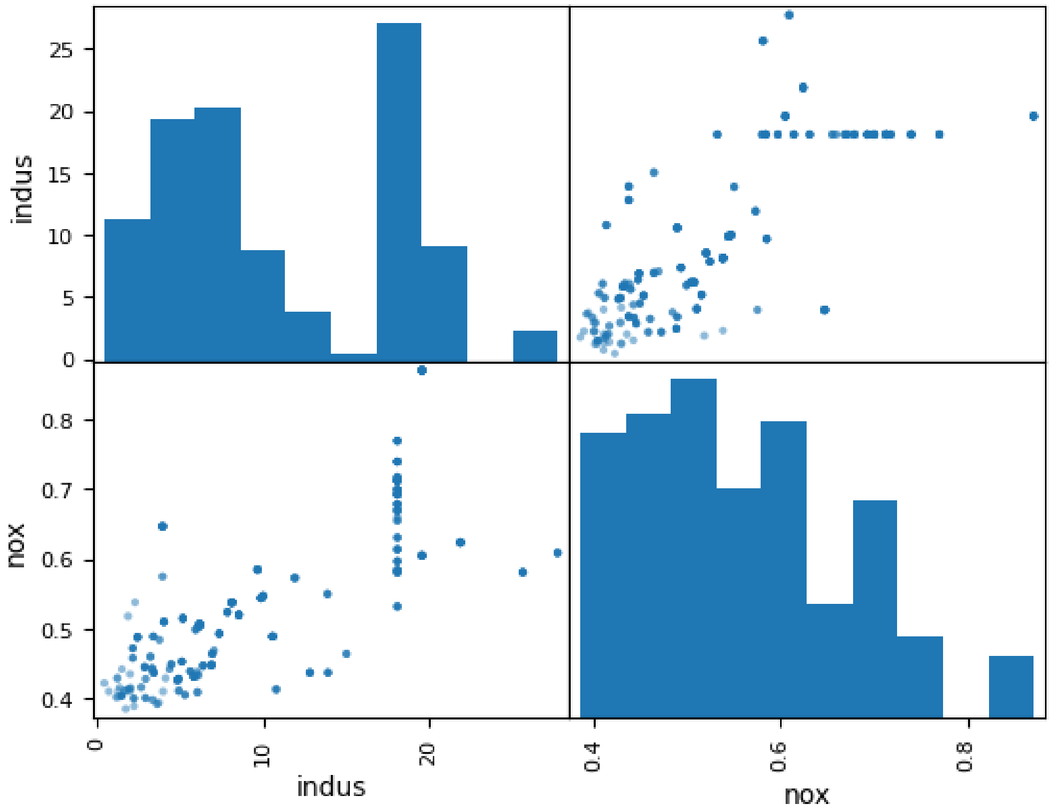
We can explain this by the `indus` variable. Suburbs that have a high proportion of non-retail business acres have a high conventration of nitrogen dioxide.

[ ]:
```
names = Boston.columns
pd.plotting.scatter_matrix(Boston[['rm', 'medv']])
pd.plotting.scatter_matrix(Boston[['medv', 'nox', 'zn']])
pd.plotting.scatter_matrix(Boston[['indus', 'nox']])
```

[ ]:
```
array([[<Axes: xlabel='indus', ylabel='indus'>,
        <Axes: xlabel='nox', ylabel='indus'>],
       [<Axes: xlabel='indus', ylabel='nox'>,
        <Axes: xlabel='nox', ylabel='nox'>]], dtype=object)
```




medv
nox
zn
40
20
0.8
0.6
0.4
100
50
0
20
40
0.4
0.6
0.8
0
50
100
medv
nox
zn



[ ]:
```
# Calculate the covariance matrix of the data to see where there is strong␣
↪correlation.
data = Boston
variable_labels = data.columns
cov_matrix = np.corrcoef(data, rowvar=False)
cov_matrix = np.round(cov_matrix, 2)
correlation_df = pd.DataFrame(cov_matrix, columns=variable_labels,␣
↪index=variable_labels)
correlation_df
```

[ ]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio \
crim	1.00	-0.20	0.41	-0.06	0.42	-0.22	0.35	-0.38	0.63	0.58	0.29
zn	-0.20	1.00	-0.53	-0.04	-0.52	0.31	-0.57	0.66	-0.31	-0.31	-0.39
indus	0.41	-0.53	1.00	0.06	0.76	-0.39	0.64	-0.71	0.60	0.72	0.38
chas	-0.06	-0.04	0.06	1.00	0.09	0.09	0.09	-0.10	-0.01	-0.04	-0.12
nox	0.42	-0.52	0.76	0.09	1.00	-0.30	0.73	-0.77	0.61	0.67	0.19
rm	-0.22	0.31	-0.39	0.09	-0.30	1.00	-0.24	0.21	-0.21	-0.29	-0.36
age	0.35	-0.57	0.64	0.09	0.73	-0.24	1.00	-0.75	0.46	0.51	0.26
dis	-0.38	0.66	-0.71	-0.10	-0.77	0.21	-0.75	1.00	-0.49	-0.53	-0.23
rad	0.63	-0.31	0.60	-0.01	0.61	-0.21	0.46	-0.49	1.00	0.91	0.46
tax	0.58	-0.31	0.72	-0.04	0.67	-0.29	0.51	-0.53	0.91	1.00	0.46

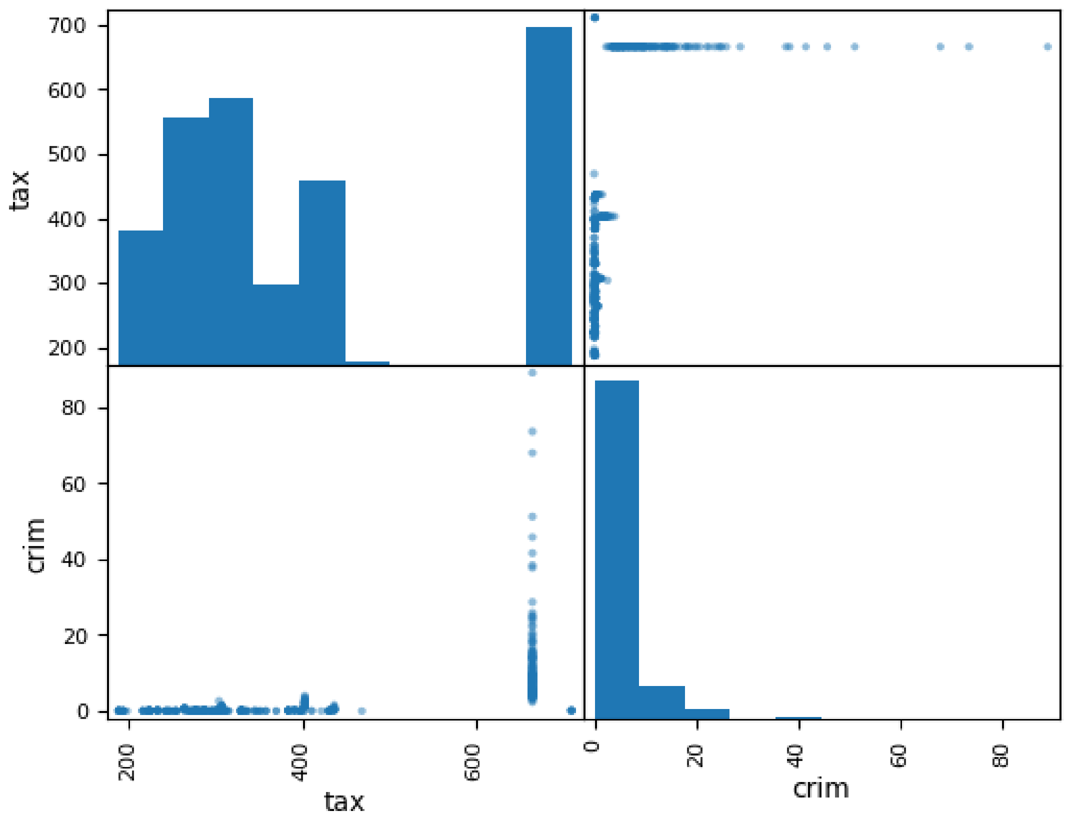
```
ptratio  0.29 -0.39   0.38 -0.12  0.19 -0.36  0.26 -0.23  0.46  0.46     1.00
lstat    0.46 -0.41   0.60 -0.05  0.59 -0.61  0.60 -0.50  0.49  0.54     0.37
medv    -0.39  0.36  -0.48  0.18 -0.43  0.70 -0.38  0.25 -0.38 -0.47    -0.51

         lstat  medv
crim      0.46 -0.39
zn       -0.41  0.36
indus     0.60 -0.48
chas     -0.05  0.18
nox       0.59 -0.43
rm       -0.61  0.70
age       0.60 -0.38
dis      -0.50  0.25
rad       0.49 -0.38
tax       0.54 -0.47
ptratio   0.37 -0.51
lstat     1.00 -0.74
medv     -0.74  1.00
```

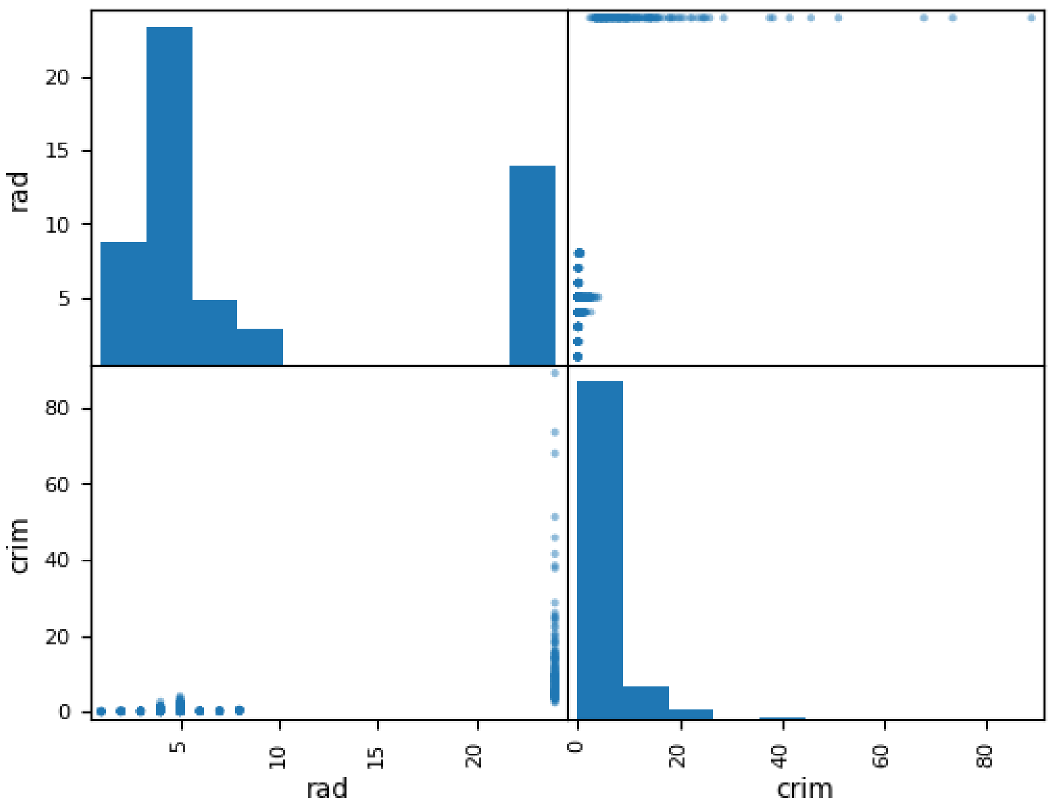
1.0.8 (d)

Factors per capita crime rate are the index of accessibility to radial highways and property tax rate per $10,0000. However, these are not very strong relationships, the correlation coefficient between `tax` and `crime` is 0.58 and the correlation coefficient betwee `rad` and `crime` is 0.63. They both have a positive relationship, but neither are very strong.

[ ]:
```python
pd.plotting.scatter_matrix(Boston[['tax','crim']])
pd.plotting.scatter_matrix(Boston[['rad', 'crim']])
```

[ ]:
```
array([[<Axes: xlabel='rad', ylabel='rad'>,
        <Axes: xlabel='crim', ylabel='rad'>],
       [<Axes: xlabel='rad', ylabel='crim'>,
        <Axes: xlabel='crim', ylabel='crim'>]], dtype=object)
```


700
600
500
400
300
200
tax
80
60
40
20
0
crim
200
400
600
tax
0
20
40
60
80
crim



(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.

Crime: There are 14 suburbs with a crime rate per capita between 80 and 89. Overall suburbs had a minimum of 0.006 crime rate and a maximum of 88.97.

Tax Rates: There are 85 suburbs in the top percentile with a tax property rate between 705.76 and 711.0 per $10,000 dollars. The full range of tax rates of the suburbs are 187 and 711.

Pupil-teacher ratio: There are 34 suburbs with in the lowest 1 percent of pupil teach ratio ranging from 21.9 to 22. The full range of pupil teacher ratio across suburbs is 12 to 22.

[ ]:
```
cut = pd.cut(Boston.crim, bins = 10)
Boston['crime_bins']= cut

print('Min:', np.min(Boston.crim))
print('Max:', np.max(Boston.crim))

top_10th_percentile_interval = cut.cat.categories[-1]
print("Top 10 percent of crime:", top_10th_percentile_interval)
Boston['crime_bins']= cut
high_crime = Boston[Boston.crime_bins == top_10th_percentile_interval]
```

```
print('Number in top 10 percentile:', high_crime.size)
Boston.crim.describe()
```

```
Min: 0.00632
Max: 88.9762
Top 10 percent of crime: (80.079, 88.976]
Number in top 10 percentile: 14
```

[ ]:
```
count    506.000000
mean       3.613524
std        8.601545
min        0.006320
25%        0.082045
50%        0.256510
75%        3.677083
max       88.976200
Name: crim, dtype: float64
```

[ ]:
```
cut = pd.cut(Boston.tax, bins = 100)
Boston['tax_bins']= cut

print('Min:', np.min(Boston.tax))
print('Max:', np.max(Boston.tax)    )

top_1_percentile_interval = cut.cat.categories[-1]
print("Top 1 percent of tax:", top_1_percentile_interval)
Boston['tax_bins']= cut
high_tax = Boston[Boston.tax_bins == top_1_percentile_interval]
print('Number in top percentile:', high_tax.size)
Boston.tax.describe()
```

```
Min: 187
Max: 711
Top 1 percent of tax: (705.76, 711.0]
Number in top percentile: 75
```

[ ]:
```
count    506.000000
mean     408.237154
std      168.537116
min      187.000000
25%      279.000000
50%      330.000000
75%      666.000000
max      711.000000
Name: tax, dtype: float64
```

[ ]:
```
cut = pd.cut(Boston.ptratio, bins=100)
Boston['ptratio_bins'] = cut

print('Min:', np.min(Boston.ptratio))
print('Max:', np.max(Boston.ptratio))

top_1_percentile_interval = cut.cat.categories[-1]
print("Top 1 percent of parent teach ratio:", top_1_percentile_interval)
Boston['ptratio_cut'] = cut
high_ptratio = Boston[Boston.ptratio_bins == top_1_percentile_interval]
print('Number in top percentile:', high_ptratio.size)
Boston.ptratio.describe()
```

```
Min: 12.6
Max: 22.0
Top 1 percent of parent teach ratio: (21.906, 22.0]
Number in top percentile: 34
```

[ ]:
```
count    506.000000
mean      18.455534
std        2.164946
min       12.600000
25%       17.400000
50%       19.050000
75%       20.200000
max       22.000000
Name: ptratio, dtype: float64
```

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

35 suburbs

[ ]:
```
bound_charles_river = Boston[Boston.chas == 1]
bound_charles_river.shape[0]
```

[ ]:
```
35
```

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

[ ]:
```
ptratio = Boston['ptratio']
medium_ptratio = np.median(ptratio)
medium_ptratio
```

[ ]:
```
19.05
```

The median pupil-teacher ratio is: 19.05

(h) Which suburb of Boston has lowest median value of owner- occupied 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.

Suburbs `398` and `405` have the lowest median value of owner occupied homes. They have a much greater incident of crime with crime indices in the greater 75th percentile with indexes if `38.3` and `67.92` respectively.

[ ]:
```
min_value_home = min(Boston.medv)
suburbs = Boston[Boston.medv == min_value_home]

print(Boston.crim.describe())
print(suburbs)
```

```
count    506.000000
mean       3.613524
std        8.601545
min        0.006320
25%        0.082045
50%        0.256510
75%        3.677083
max       88.976200
Name: crim, dtype: float64
        crim   zn  indus  chas    nox     rm    age     dis  rad  tax  \
398  38.3518  0.0   18.1     0  0.693  5.453  100.0  1.4896   24  666
405  67.9208  0.0   18.1     0  0.693  5.683  100.0  1.4254   24  666

     ptratio  lstat  medv         crime_bins          tax_bins  \
398     20.2  30.59   5.0  (35.594, 44.491]  (663.84, 669.08]
405     20.2  22.98   5.0  (62.285, 71.182]  (663.84, 669.08]

        ptratio_bins      ptratio_cut
398  (20.12, 20.214]  (20.12, 20.214]
405  (20.12, 20.214]  (20.12, 20.214]
```

(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.

More than seven rooms per dwelling: 64 More than eight rooms per dwelling: 13

The suburbs with that average 8 rooms per dwelling have a low mean crime index. The 75th percentile of Boston overall has an average index of 3.67 while the suburbs averaging 8 rooms or more has an index of 0.71. Some other extreme values (>75% or <25% percentiles) are accessibility to radial highways and lower status of the population. The targeted suburbs have a mean of 4.31% of the population having lower status.

[ ]:
```
stats = Boston.describe()
eight = Boston[Boston.rm > 8].describe()
```

```
seventy_fifth = stats.loc['75%']
twenty_fifth = stats.loc['25%']

extreme_columns = eight.columns[(eight.iloc[1] > seventy_fifth) | (eight.
 ↪iloc[0] < twenty_fifth)].tolist()
filtered_eight = eight[extreme_columns]
filtered_eight.iloc[1]
```

[ ]:
```
zn          13.615385
chas         0.153846
rm           8.348538
age         71.538462
tax        325.076923
ptratio     16.361538
medv        44.200000
Name: mean, dtype: float64
```

[ ]:
```
# For reference
Boston.describe()[extreme_columns].iloc[1:]
```

[ ]:
```
              zn      chas        rm         age         tax    ptratio  \
mean   11.363636  0.069170  6.284634   68.574901  408.237154  18.455534
std    23.322453  0.253994  0.702617   28.148861  168.537116   2.164946
min     0.000000  0.000000  3.561000    2.900000  187.000000  12.600000
25%     0.000000  0.000000  5.885500   45.025000  279.000000  17.400000
50%     0.000000  0.000000  6.208500   77.500000  330.000000  19.050000
75%    12.500000  0.000000  6.623500   94.075000  666.000000  20.200000
max   100.000000  1.000000  8.780000  100.000000  711.000000  22.000000

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

[ ]:
```
eight = Boston[Boston.rm > 8]
print('Number of suburbs with greater than eight:', eight.shape[0])
eight.describe()
```

```
Number of suburbs with greater than eight: 13
```

[ ]:
```
            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
```