Brian_Reppeto540Week3_4

January 6, 2024

0.0.1 DSC 540 Week 2 Data Wrangling with Python:

0.0.2 Activity 5 Generating Stats from a csv file

0.0.3 Author: Brian Reppeto¶ 12/17/2023

[94]:
```
# import libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[95]:
```
#read the Boston housing dataset

bost_hs_df=pd.read_csv("Boston_housing.csv")
```

[96]:
```
# read the first 10 rows

bost_hs_df.head(10)
```

[96]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2
5	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
6	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2			
7	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2			
8	0.21124	12.5	7.87	0	0.524	5.631	100.0	6.0821	5	311	15.2			
9	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311	15.2			

```
6  395.60  12.43   22.9
7  396.90  19.15   27.1
8  386.63  29.93   16.5
9  386.71  17.10   18.9
```

[97]:
```
# Use Np shape array to find the rows and column count.  506 rows 14 columns

bost_hs_df.shape
```

[97]:
```
(506, 14)
```

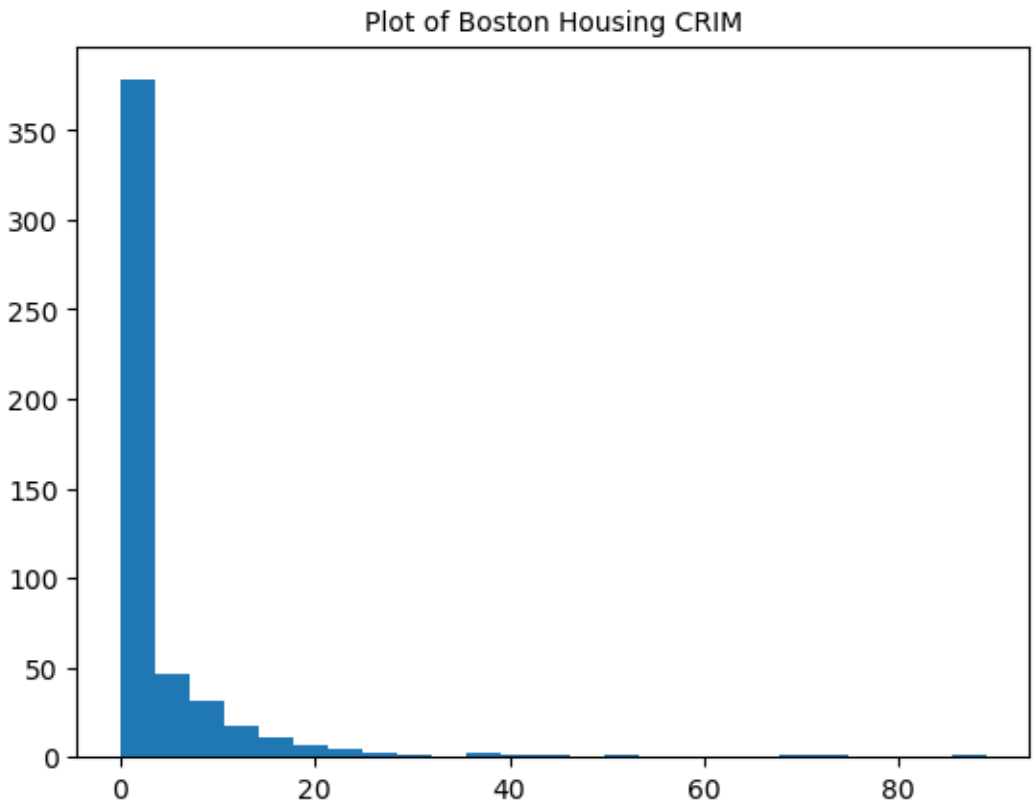
[98]:
```
# Smaller DF with selected columns from the original df

bost_hs_sm_df=bost_hs_df[['CRIM','ZN','INDUS','RM','AGE','DIS','RAD','TAX','PTRATIO','PRICE']]
```

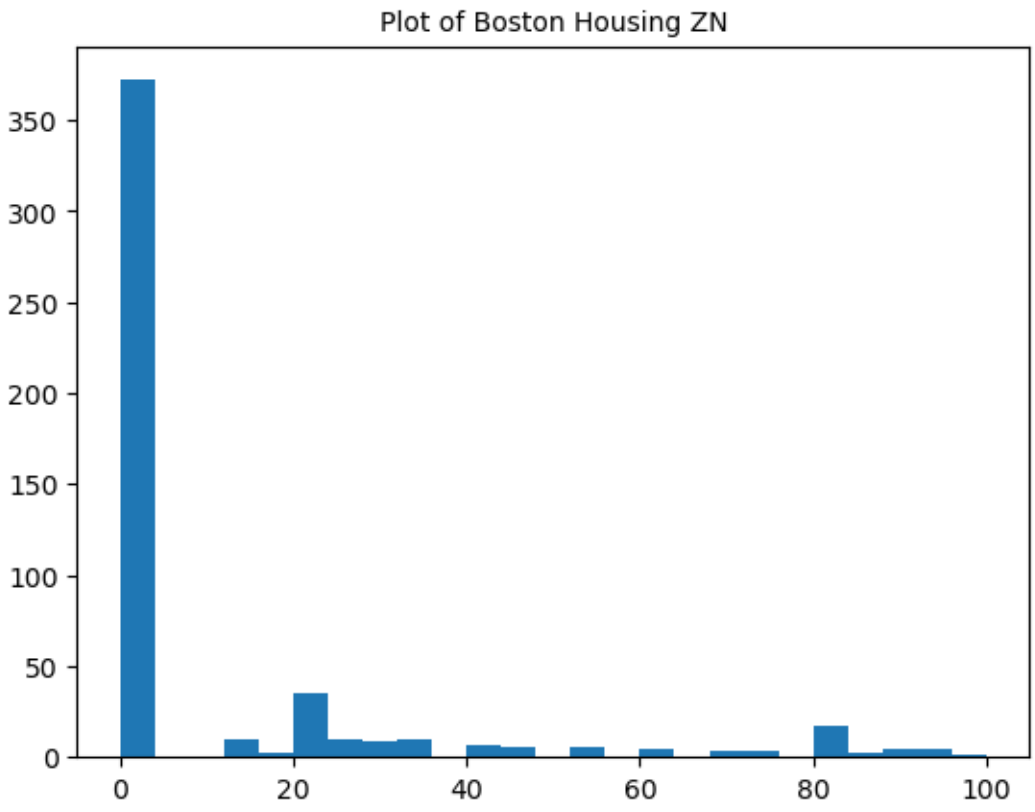
[99]:
```
# Check last 7 records using tail (7) to grab the last 7 rows

bost_hs_sm_df.tail(7)
```

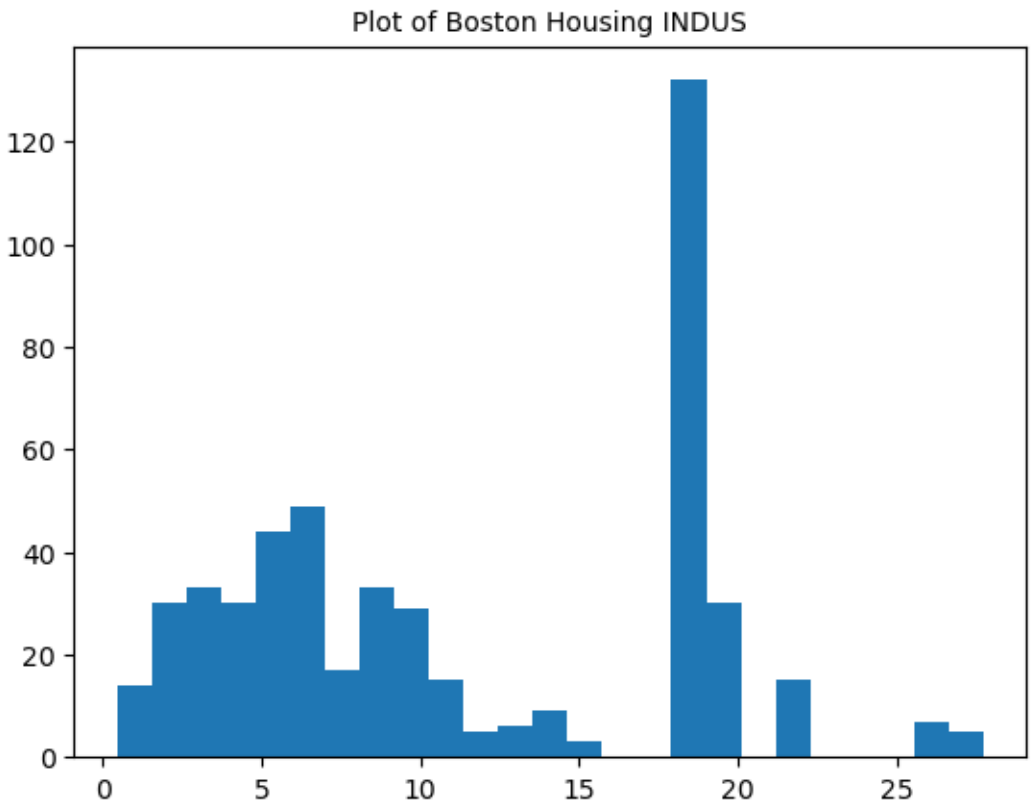
[99]:

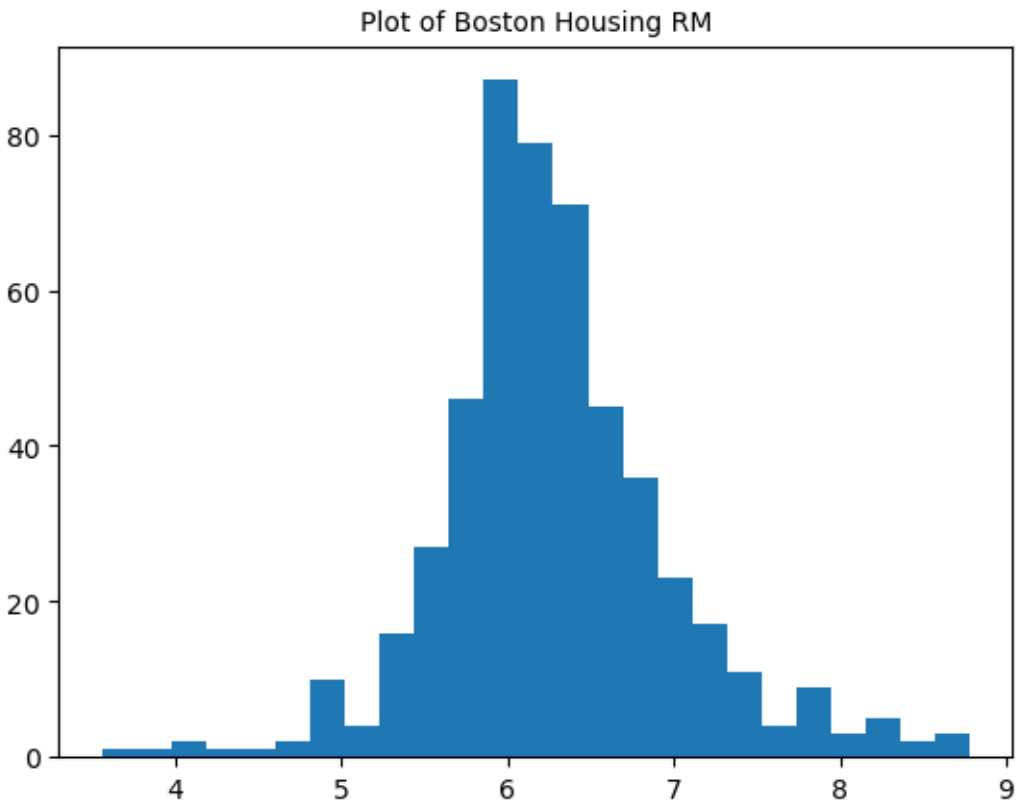
	CRIM	ZN	INDUS	RM	AGE	DIS	RAD	TAX	PTRATIO	PRICE
499	0.17783	0.0	9.69	5.569	73.5	2.3999	6	391	19.2	17.5
500	0.22438	0.0	9.69	6.027	79.7	2.4982	6	391	19.2	16.8
501	0.06263	0.0	11.93	6.593	69.1	2.4786	1	273	21.0	22.4
502	0.04527	0.0	11.93	6.120	76.7	2.2875	1	273	21.0	20.6
503	0.06076	0.0	11.93	6.976	91.0	2.1675	1	273	21.0	23.9
504	0.10959	0.0	11.93	6.794	89.3	2.3889	1	273	21.0	22.0
505	0.04741	0.0	11.93	6.030	80.8	2.5050	1	273	21.0	11.9

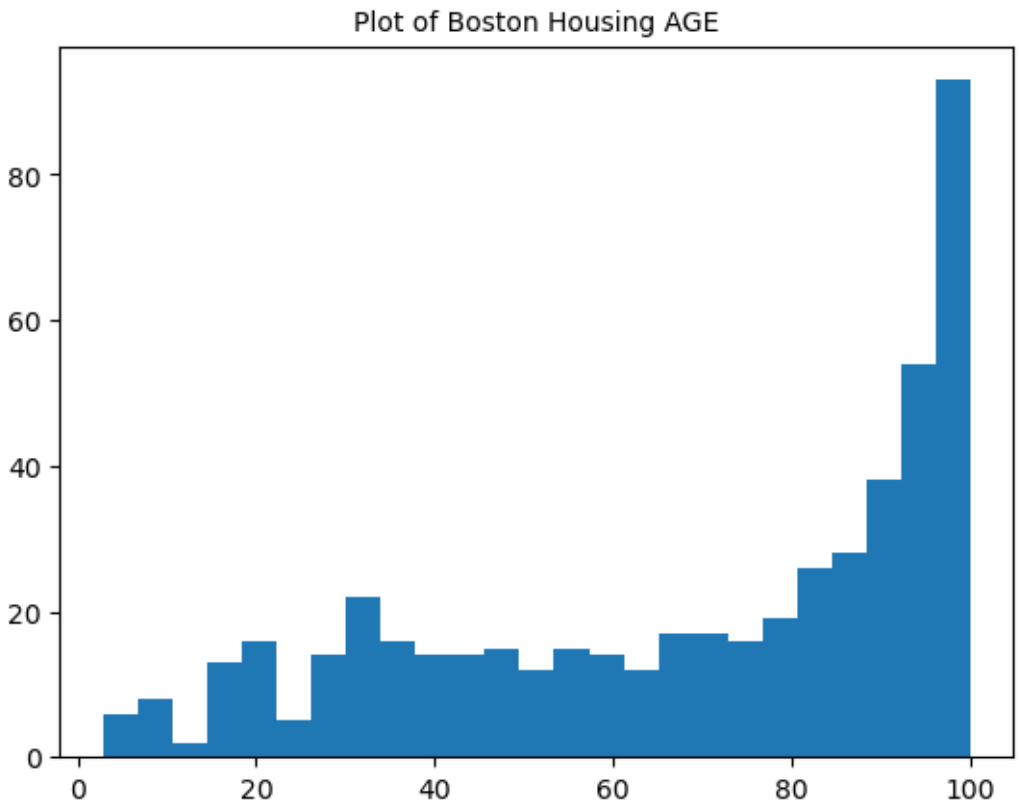
[100]:
```
# Using matplot and the bost_hs_sm df create histograms for each of the columns

# Iterate over columns

for i in bost_hs_sm_df.columns:
    plt.title("Plot of Boston Housing "+ i,fontsize=10)
    plt.hist(bost_hs_sm_df[i],bins=25)
    plt.show()
```

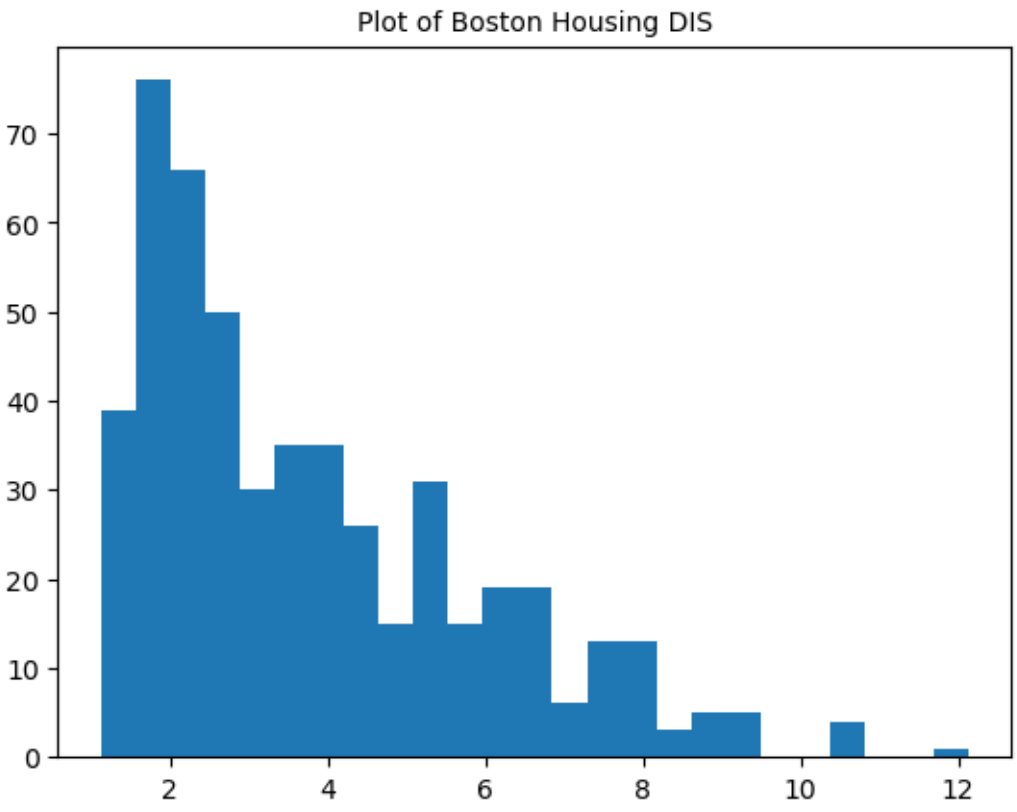

Plot of Boston Housing CRIM
350
300
250
200
150
100
50
0
0
20
40
60
80

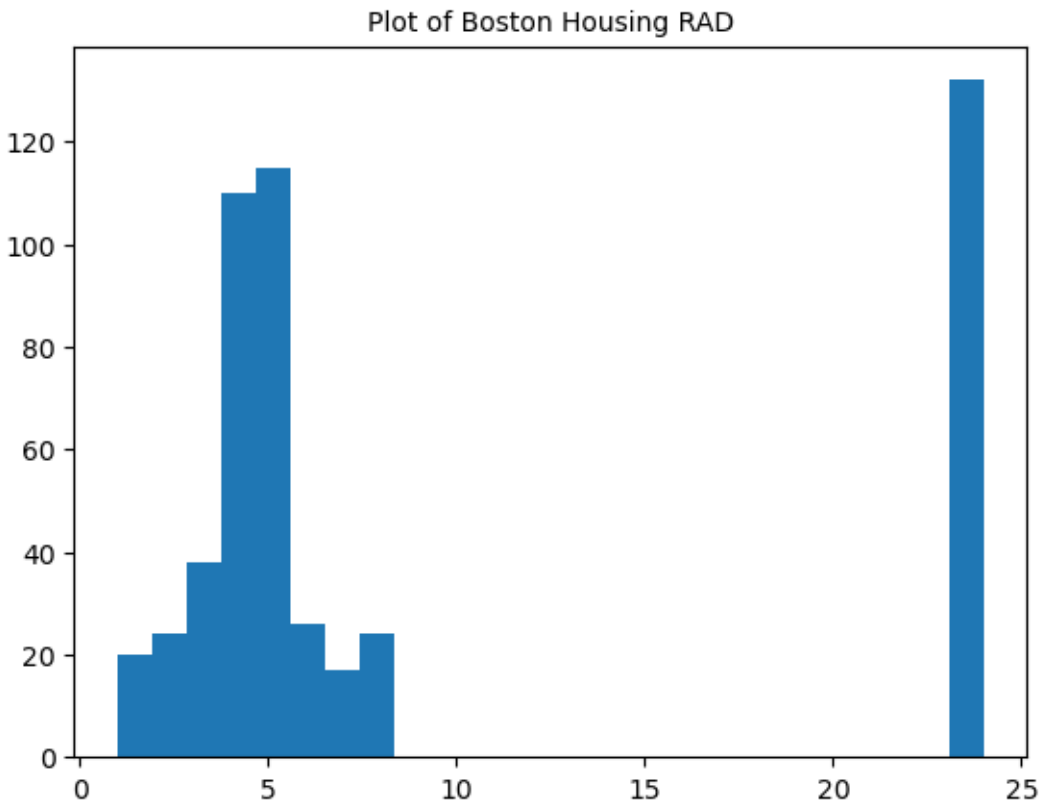

Plot of Boston Housing ZN
350
300
250
200
150
100
50
0
0
20
40
60
80
100

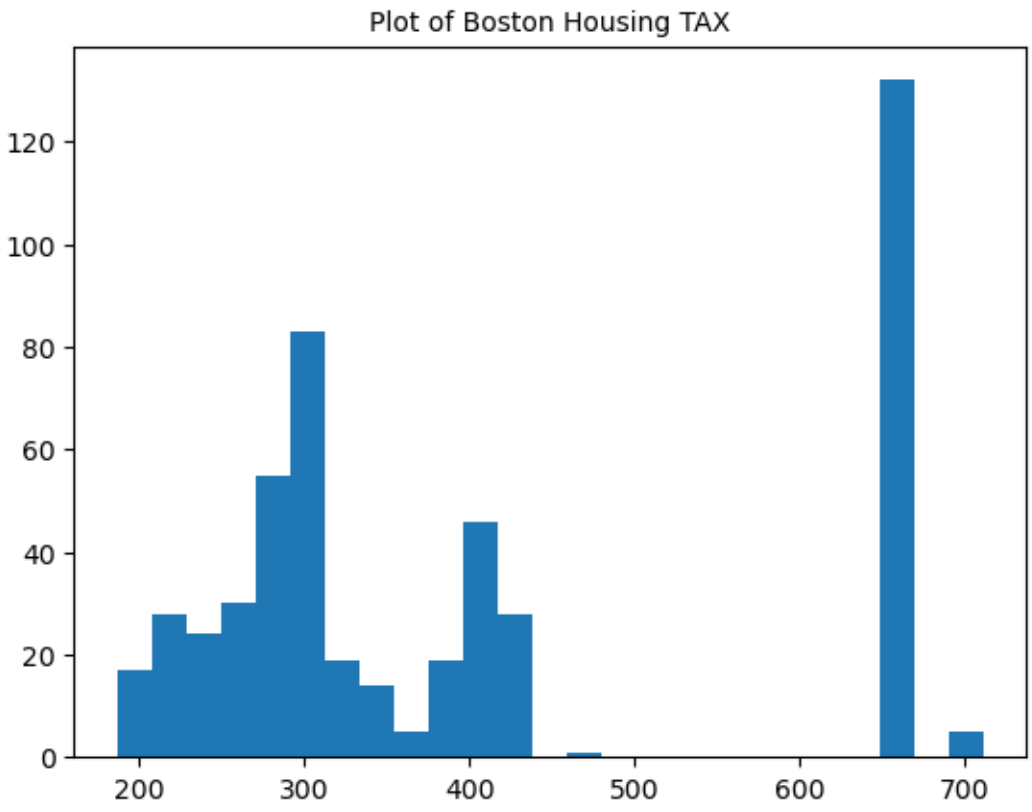

Plot of Boston Housing INDUS
120
100
80
60
40
20
0
0
5
10
15
20
25

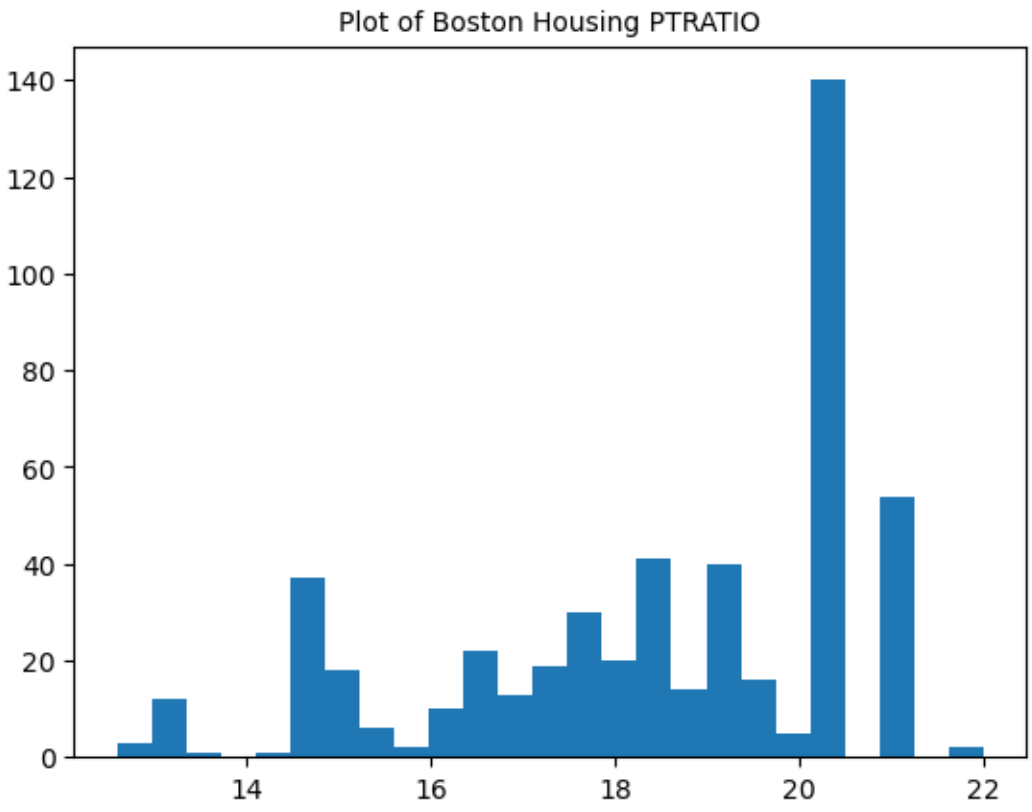

Plot of Boston Housing RM
80
60
40
20
0
4
5
6
7
8
9

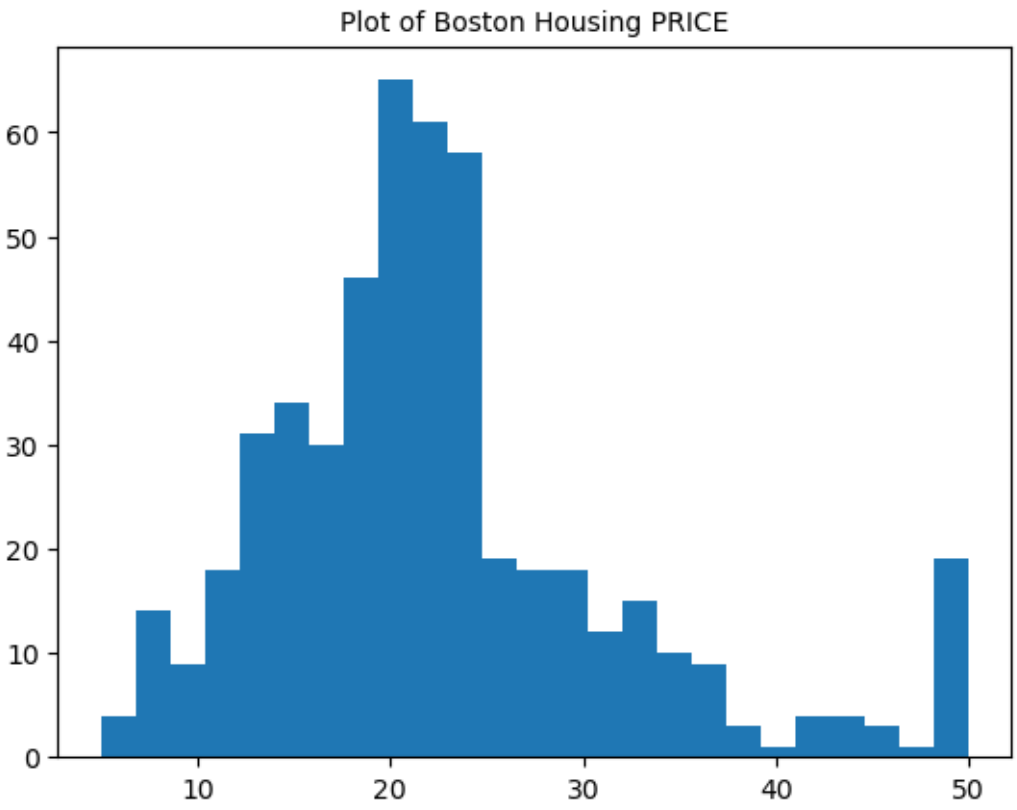

Plot of Boston Housing AGE
0
20
40
60
80
0
20
40
60
80
100


Plot of Boston Housing DIS
70
60
50
40
30
20
10
0
2
4
6
8
10
12

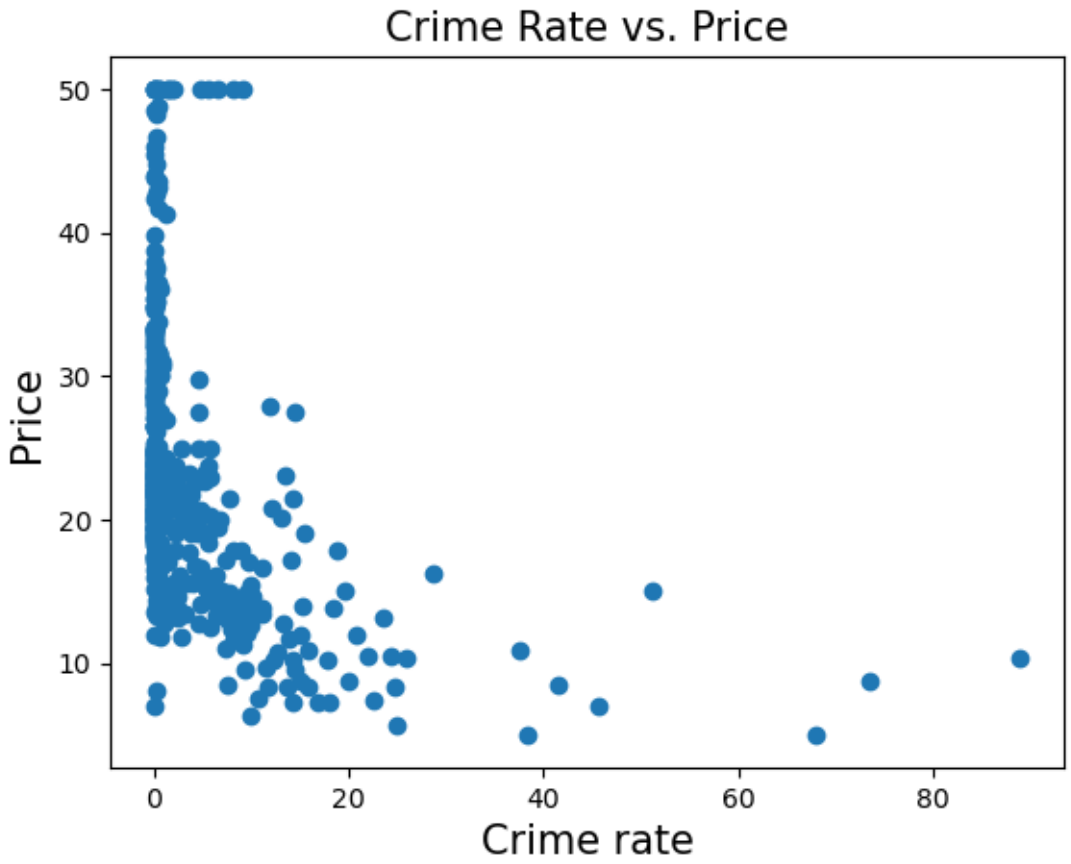

Plot of Boston Housing RAD
120
100
80
60
40
20
0
0
5
10
15
20
25


Plot of Boston Housing TAX
120
100
80
60
40
20
0
200
300
400
500
600
700

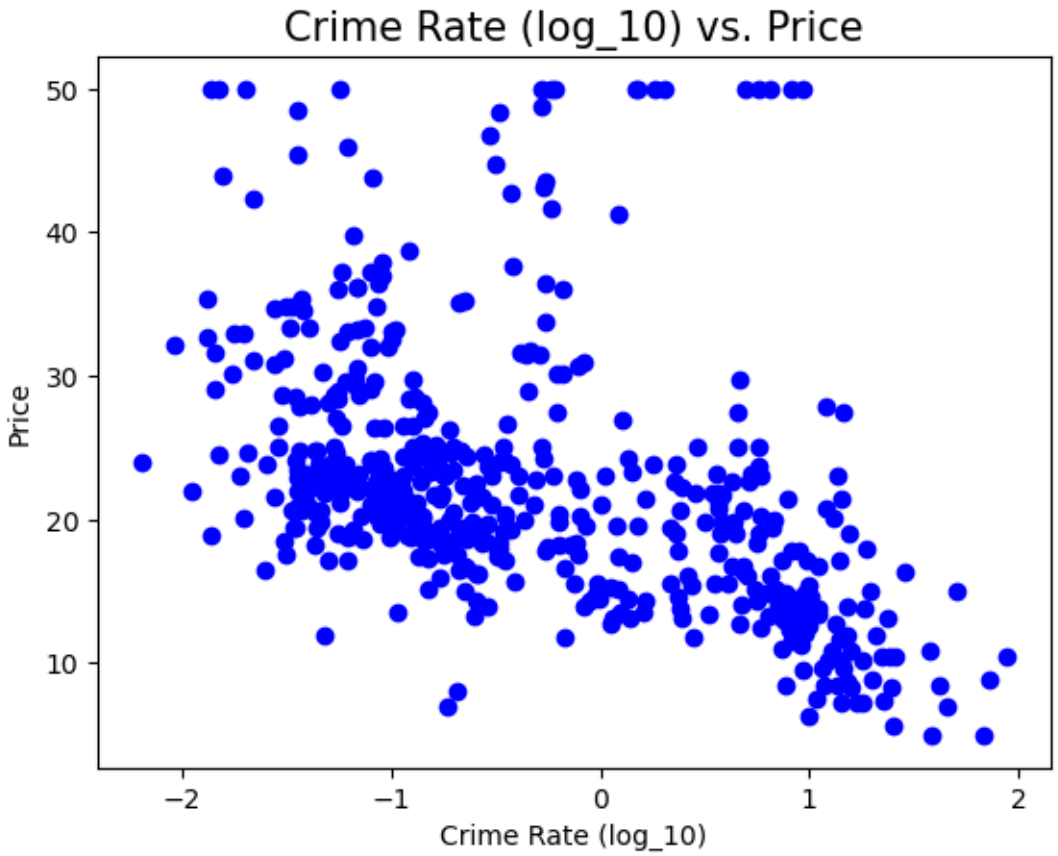

Plot of Boston Housing PTRATIO
140
120
100
80
60
40
20
0
14
16
18
20
22



[101]:
```
# Scatter plot of crime rate vs price using matplot

plt.scatter(bost_hs_sm_df['CRIM'],bost_hs_sm_df['PRICE'])
plt.title('Crime Rate vs. Price', fontsize=15)
plt.xlabel("Crime rate",fontsize=15)
plt.ylabel("Price",fontsize=15)
plt.show()
```



[102]:
```
# Using Matplot Using log(10)

# Create a scatter plot
plt.scatter(np.log10(bost_hs_sm_df['CRIM']), bost_hs_sm_df['PRICE'],␣
 ↪label='CRIM', color='blue')

# Add labels and title
plt.xlabel('Crime Rate (log_10)')
plt.ylabel('Price')
plt.title('Crime Rate (log_10) vs. Price', fontsize=15)

# Show the plot
plt.show()
```



[103]:
```
# Calculate the mean of rooms using the Boston housing df.

bost_hs_sm_df['RM'].mean()
```

[103]: 6.284634387351779

[104]:
```
# Calculate the median age using the Boston housing df.

bost_hs_sm_df['AGE'].median()
```

[104]: 77.5

[105]:
```
# Calculate the mean distance to five Boston employment centers using the␣
↪Boston housing df.

bost_hs_sm_df['DIS'].mean()
```

[105]: 3.795042687747036

[106]:
```
# create a boolean index to check if the values in the 'PRICE' column of a␣
↪DataFramewhere price is less than 20

house_less_20=bost_hs_sm_df['PRICE']<20
```

[107]:
```
# Print the boolean index where the new boolean is only not equal to false

print((house_less_20 != "False") )
```

```
0      True
1      True
2      True
3      True
4      True
       …
501    True
502    True
503    True
504    True
505    True
Name: PRICE, Length: 506, dtype: bool
```

[108]:
```
# Find the total number of rows

num_rows = len(house_less_20)

print("Total number of rows:", num_rows)
```

```
Total number of rows: 506
```

[109]:
```
# Calculate the mean of the house_less_20 variable and then expresses that mean␣
↪as a percentage
# by multiplying it by 100. The result is stored in the variable perc.

perc=house_less_20.mean()*100
```

[110]:
```
#  using an f-string print a message then the calcuation of the percent of␣
↪houses < 20,000.

print(f" % of houses with < 20,000: {perc} ")
```

```
 % of houses with < 20,000: 41.50197628458498
```

0.1 Activity 6 Working with the Adult Income Dataset

[111]:
```
# import libraries

import numpy as np
```

```
import pandas as pd
import matplotlib.pyplot as plt
```

[112]:
```
#read the adult_income_data. dataset

adult_df=pd.read_csv("adult_income_data.csv")
```

[113]:
```
# read the first 10 rows

adult_df.head(10)
```

[113]:
```
   39          State-gov   77516      Bachelors  13           Never-married  \
0  50   Self-emp-not-inc   83311      Bachelors  13      Married-civ-spouse
1  38            Private  215646        HS-grad   9                Divorced
2  53            Private  234721           11th   7      Married-civ-spouse
3  28            Private  338409      Bachelors  13      Married-civ-spouse
4  37            Private  284582        Masters  14      Married-civ-spouse
5  49            Private  160187            9th   5   Married-spouse-absent
6  52   Self-emp-not-inc  209642        HS-grad   9      Married-civ-spouse
7  31            Private   45781        Masters  14           Never-married
8  42            Private  159449      Bachelors  13      Married-civ-spouse
9  37            Private  280464   Some-college  10      Married-civ-spouse

          Adm-clerical   Not-in-family     Male   2174  0  40   United-States  \
0     Exec-managerial         Husband     Male      0  0  13   United-States
1   Handlers-cleaners   Not-in-family     Male      0  0  40   United-States
2   Handlers-cleaners         Husband     Male      0  0  40   United-States
3      Prof-specialty            Wife   Female      0  0  40            Cuba
4     Exec-managerial            Wife   Female      0  0  40   United-States
5       Other-service   Not-in-family   Female      0  0  16         Jamaica
6     Exec-managerial         Husband     Male      0  0  45   United-States
7      Prof-specialty   Not-in-family   Female  14084  0  50   United-States
8     Exec-managerial         Husband     Male   5178  0  40   United-States
9     Exec-managerial         Husband     Male      0  0  80   United-States

    <=50K
0   <=50K
1   <=50K
2   <=50K
3   <=50K
4   <=50K
5   <=50K
6    >50K
7    >50K
8    >50K
9    >50K
```

[114]:
```
# using the .txt file  create a empty list, open the .txt file loop thru each␣
 ↪line in .txt seperated by ':'.
# append to the headers list

headers = []  # initializes an empty list called headers to store the extracted␣
 ↪headers.
with open('adult_income_names.txt','r') as b:
    for line in b:
        b.readline()
        var=line.split(":")[0]
        headers.append(var)
```

[115]:
```
# Output the headers list values

headers
```

[115]:
```
['age',
 'workclass',
 'fnlwgt',
 'education',
 'education-num',
 'marital-status',
 'occupation',
 'relationship',
 'sex',
 'capital-gain',
 'capital-loss',
 'hours-per-week',
 'native-country']
```

[116]:
```
# append the 'Income' name to the response dataset

headers.append('Income')
```

[117]:
```
# update the adult_df with the new names

adult_df=pd.read_csv("adult_income_data.csv",names=headers)

# Output the updated file for the new column

adult_df.head(10)
```

[117]:

	age	workclass	fnlwgt	education	education-num	\
0	39	State-gov	77516	Bachelors	13	
1	50	Self-emp-not-inc	83311	Bachelors	13	
2	38	Private	215646	HS-grad	9	
3	53	Private	234721	11th	7	

```
4   28            Private  338409   Bachelors             13
5   37            Private  284582     Masters             14
6   49            Private  160187         9th              5
7   52   Self-emp-not-inc  209642     HS-grad              9
8   31            Private   45781     Masters             14
9   42            Private  159449   Bachelors             13

           marital-status         occupation    relationship      sex  \
0           Never-married       Adm-clerical   Not-in-family     Male
1      Married-civ-spouse    Exec-managerial         Husband     Male
2                Divorced  Handlers-cleaners   Not-in-family     Male
3      Married-civ-spouse  Handlers-cleaners         Husband     Male
4      Married-civ-spouse     Prof-specialty            Wife   Female
5      Married-civ-spouse    Exec-managerial            Wife   Female
6   Married-spouse-absent      Other-service   Not-in-family   Female
7      Married-civ-spouse    Exec-managerial         Husband     Male
8           Never-married     Prof-specialty   Not-in-family   Female
9      Married-civ-spouse    Exec-managerial         Husband     Male

   capital-gain  capital-loss  hours-per-week  native-country  Income
0          2174             0              40   United-States   <=50K
1             0             0              13   United-States   <=50K
2             0             0              40   United-States   <=50K
3             0             0              40   United-States   <=50K
4             0             0              40            Cuba   <=50K
5             0             0              40   United-States   <=50K
6             0             0              16         Jamaica   <=50K
7             0             0              45   United-States    >50K
8         14084             0              50   United-States    >50K
9          5178             0              40   United-States    >50K
```

[118]:
```
# Missing data for adult_income_data DF

adult_df.isnull().sum()
```

[118]:
```
age               0
workclass         0
fnlwgt            0
education         0
education-num     0
marital-status    0
occupation        0
relationship      0
sex               0
capital-gain      0
capital-loss      0
hours-per-week    0
```

```
native-country    0
Income            0
dtype: int64
```

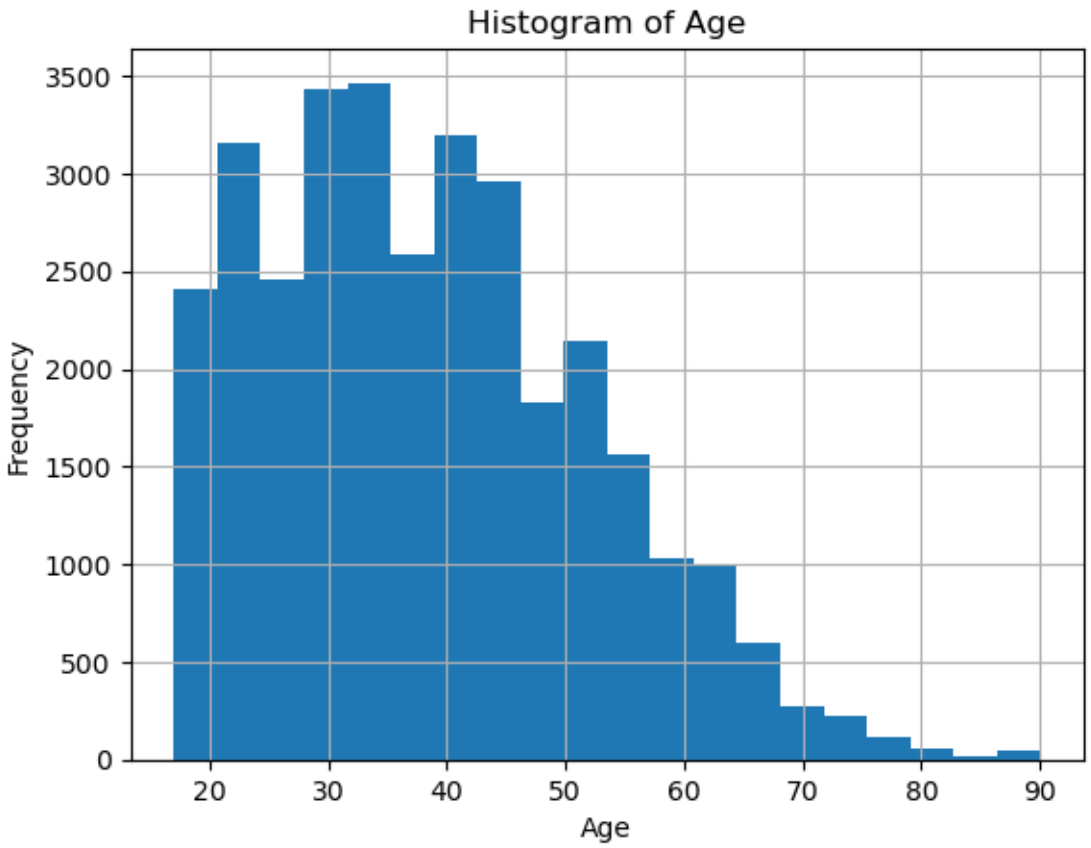
[119]:
```
# Create a subset df with only 'age', 'education', and 'occupation'.

adult_sub=adult_df[['age','education','occupation']]
```

[120]:
```
# Head the new subset df to see changes

adult_sub.head(10)
```

[120]:

	age	education	occupation
0	39	Bachelors	Adm-clerical
1	50	Bachelors	Exec-managerial
2	38	HS-grad	Handlers-cleaners
3	53	11th	Handlers-cleaners
4	28	Bachelors	Prof-specialty
5	37	Masters	Exec-managerial
6	49	9th	Other-service
7	52	HS-grad	Exec-managerial
8	31	Masters	Prof-specialty
9	42	Bachelors	Exec-managerial

[121]:
```
# Plot a histogram of age with a bin size of 20

adult_sub['age'].hist(bins=20)
plt.title('Histogram of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



[122]:
```
# Function to strip whitespace characters using the python strip method command

def stp_wht_spc(c):
    return c.strip()
```

[123]:
```
# Use the apply method to apply the stp_wht_spc function to the adult_sub df␣
 ↪string columns

# Education columns

adult_sub['ed_strip']=adult_df['education'].apply(stp_wht_spc)
adult_sub['education']=adult_sub['ed_strip']
adult_sub.drop(labels=['ed_strip'],axis=1,inplace=True)

# Occupation column

adult_sub['oc_strip']=adult_df['occupation'].apply(stp_wht_spc)
adult_sub['occupation']=adult_sub['oc_strip']
adult_sub.drop(labels=['oc_strip'],axis=1,inplace=True)
```

```
/var/folders/yd/8fns7t4j1n9gv77y0dvs5zdc0000gn/T/ipykernel_27766/1840534740.py:5
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  adult_sub['ed_strip']=adult_df['education'].apply(stp_wht_spc)
/var/folders/yd/8fns7t4j1n9gv77y0dvs5zdc0000gn/T/ipykernel_27766/1840534740.py:6
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  adult_sub['education']=adult_sub['ed_strip']
/var/folders/yd/8fns7t4j1n9gv77y0dvs5zdc0000gn/T/ipykernel_27766/1840534740.py:7
: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  adult_sub.drop(labels=['ed_strip'],axis=1,inplace=True)
/var/folders/yd/8fns7t4j1n9gv77y0dvs5zdc0000gn/T/ipykernel_27766/1840534740.py:1
1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  adult_sub['oc_strip']=adult_df['occupation'].apply(stp_wht_spc)
/var/folders/yd/8fns7t4j1n9gv77y0dvs5zdc0000gn/T/ipykernel_27766/1840534740.py:1
2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  adult_sub['occupation']=adult_sub['oc_strip']
/var/folders/yd/8fns7t4j1n9gv77y0dvs5zdc0000gn/T/ipykernel_27766/1840534740.py:1
3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  adult_sub.drop(labels=['oc_strip'],axis=1,inplace=True)
```

[124]:
```
# Head the subset df

adult_sub.head(10)
```

[124]:

	age	education	occupation
0	39	Bachelors	Adm-clerical
1	50	Bachelors	Exec-managerial
2	38	HS-grad	Handlers-cleaners
3	53	11th	Handlers-cleaners
4	28	Bachelors	Prof-specialty
5	37	Masters	Exec-managerial
6	49	9th	Other-service
7	52	HS-grad	Exec-managerial
8	31	Masters	Prof-specialty
9	42	Bachelors	Exec-managerial

[125]:
```
# Find the number of people who are aged between 30 and 50

adult_ages=adult_sub[(adult_sub['age']>=30) & (adult_sub['age']<=50)]
```

[126]:
```
# Head the filter to show the results

adult_ages.head(10)
```

[126]:

	age	education	occupation
0	39	Bachelors	Adm-clerical
1	50	Bachelors	Exec-managerial
2	38	HS-grad	Handlers-cleaners
5	37	Masters	Exec-managerial
6	49	9th	Other-service
8	31	Masters	Prof-specialty
9	42	Bachelors	Exec-managerial
10	37	Some-college	Exec-managerial
11	30	Bachelors	Prof-specialty
13	32	Assoc-acdm	Sales

[127]:
```
# Create a subset df with only 'age', 'education' to use for the group by below

adult_sub1=adult_sub[['education','age']]
```

[128]:
```
# Create a subset df people who are aged between 30 and 50

adult_sub2=adult_sub1[(adult_sub['age']>=30) & (adult_sub['age']<=50)]
```

[129]:
```
# head the new DF to see results

adult_sub2.head(10)
```

[129]:

	education	age
0	Bachelors	39
1	Bachelors	50
2	HS-grad	38
5	Masters	37
6	9th	49
8	Masters	31
9	Bachelors	42
10	Some-college	37
11	Bachelors	30
13	Assoc-acdm	32

[130]:
```
# Group the record based on age and education to find the mean age
# Since the find the number between 30 and 50 was the question before this I␣
 ↪assumed this was the ask
# If my assumption was wrong I can change, just let me know

adult_sub2.groupby(['education']).mean()
```

[130]:

education	age
10th	39.055921
11th	38.777188
12th	38.362319
1st-4th	39.384615
5th-6th	40.422222
7th-8th	40.617021
9th	38.894737
Assoc-acdm	38.825816
Assoc-voc	38.803851
Bachelors	39.179910
Doctorate	40.889831
HS-grad	39.049671
Masters	40.960360
Preschool	38.428571
Prof-school	40.492147
Some-college	39.224719

[131]:
```
# Group by occupation and show summary stats for age

result=adult_sub.groupby('occupation').describe()['age']


# Sort by mean age in descending order
result_sorted = result.sort_values(by='mean', ascending=False)

# Display the sorted result
```

```
print(result_sorted)
```

```
                   count       mean        std   min   25%   50%   75%   max
occupation
Exec-managerial   4066.0  42.169208  11.974548  17.0  33.0  41.0  50.0  90.0
Priv-house-serv    149.0  41.724832  18.633688  17.0  24.0  40.0  57.0  81.0
Farming-fishing    994.0  41.211268  15.070283  17.0  29.0  39.0  52.0  90.0
?                 1843.0  40.882800  20.336350  17.0  21.0  35.0  61.0  90.0
Prof-specialty    4140.0  40.517633  12.016676  17.0  31.0  40.0  48.0  90.0
Transport-moving  1597.0  40.197871  12.450792  17.0  30.0  39.0  49.0  90.0
Craft-repair      4099.0  39.031471  11.606436  17.0  30.0  38.0  47.0  90.0
Protective-serv    649.0  38.953775  12.822062  17.0  29.0  36.0  47.0  90.0
Machine-op-inspct 2002.0  37.715285  12.068266  17.0  28.0  36.0  46.0  90.0
Sales             3650.0  37.353973  14.186352  17.0  25.0  35.0  47.0  90.0
Tech-support       928.0  37.022629  11.316594  17.0  28.0  36.0  44.0  73.0
Adm-clerical      3770.0  36.964456  13.362998  17.0  26.0  35.0  46.0  90.0
Other-service     3295.0  34.949621  14.521508  17.0  22.0  32.0  45.0  90.0
Handlers-cleaners 1370.0  32.165693  12.372635  17.0  23.0  29.0  39.0  90.0
Armed-Forces         9.0  30.222222   8.089774  23.0  24.0  29.0  34.0  46.0
```

The profession with the oldest workers on average is “Exec-managerial”. The profession that has the largest share above 75% is the “?” profession.

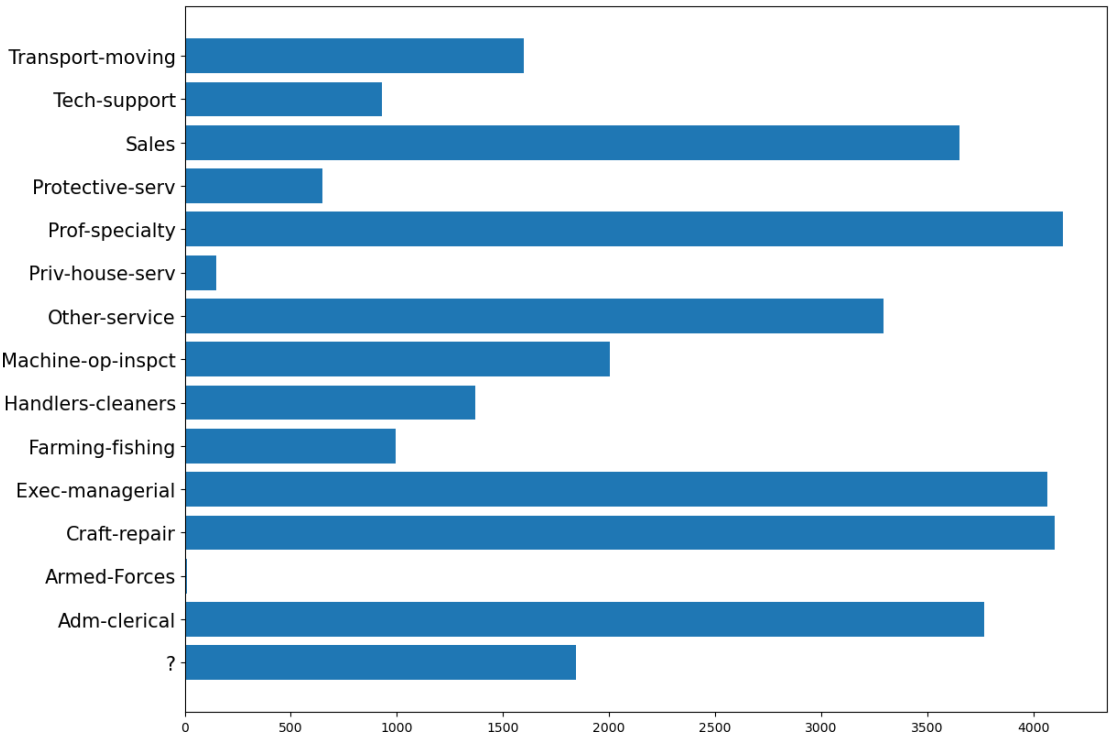
[132]:
```
# Use subset and groupby to find the outliers

out_stat=adult_sub.groupby('occupation').describe()['age']

# Sort by count in descending order
out_sorted = out_stat.sort_values(by='count', ascending=False)

# Display the sorted result
print(out_sorted)
```

```
                   count       mean        std   min   25%   50%   75%   max
occupation
Prof-specialty    4140.0  40.517633  12.016676  17.0  31.0  40.0  48.0  90.0
Craft-repair      4099.0  39.031471  11.606436  17.0  30.0  38.0  47.0  90.0
Exec-managerial   4066.0  42.169208  11.974548  17.0  33.0  41.0  50.0  90.0
Adm-clerical      3770.0  36.964456  13.362998  17.0  26.0  35.0  46.0  90.0
Sales             3650.0  37.353973  14.186352  17.0  25.0  35.0  47.0  90.0
Other-service     3295.0  34.949621  14.521508  17.0  22.0  32.0  45.0  90.0
Machine-op-inspct 2002.0  37.715285  12.068266  17.0  28.0  36.0  46.0  90.0
?                 1843.0  40.882800  20.336350  17.0  21.0  35.0  61.0  90.0
Transport-moving  1597.0  40.197871  12.450792  17.0  30.0  39.0  49.0  90.0
Handlers-cleaners 1370.0  32.165693  12.372635  17.0  23.0  29.0  39.0  90.0
Farming-fishing    994.0  41.211268  15.070283  17.0  29.0  39.0  52.0  90.0
Tech-support       928.0  37.022629  11.316594  17.0  28.0  36.0  44.0  73.0
Protective-serv    649.0  38.953775  12.822062  17.0  29.0  36.0  47.0  90.0
Priv-house-serv    149.0  41.724832  18.633688  17.0  24.0  40.0  57.0  81.0
```

```
Armed-Forces          9.0  30.222222   8.089774  23.0  24.0  29.0  34.0  46.0
```

The Armed-Forces occucation appears to be an outlier.

[133]:
```
# Plot the values of the bar chart

plt.figure(figsize=(13,10))
plt.barh(y=out_stat.index,width=out_stat['count'])
plt.yticks(fontsize=15)
plt.show()
```



[134]:
```
# Merge data using common keys.
# create new df by subset of columns from orig df then sample 10 rows and seed␣
 ↪the rand # gen

merge_adult=adult_df[['age','workclass', 'occupation']].
 ↪sample(10,random_state=101)
```

[135]:
```
# head the new df

merge_adult.head()
```

[135]:
```
       age workclass          occupation
22357   51   Private   Machine-op-inspct
```

```
26009   19   Private              Sales
20734   40   Private     Exec-managerial
17695   17   Private   Handlers-cleaners
27908   61   Private        Craft-repair
```

[136]:
```
# Merge data using common keys.
# create new df by subset of columns from orig df then sample 10 rows and seed the rand # gen

merge2_adult=adult_df[['education','sex','occupation']].sample(10,random_state=101)
```

[137]:
```
# head the new df

merge2_adult.head()
```

[137]:
```
       education      sex          occupation
22357   HS-grad   Female   Machine-op-inspct
26009      11th     Male               Sales
20734   HS-grad     Male     Exec-managerial
17695      10th     Male   Handlers-cleaners
27908   7th-8th     Male        Craft-repair
```

[138]:
```
# Merge data using common keys.
# create new df merge from two new dfs on occupation and an inner join droping the dups.

final_adult=pd.merge(merge_adult, merge2_adult, on ='occupation',how ='inner').drop_duplicates()
```

[139]:
```
# results

final_adult
```

[139]:
```
     age   workclass          occupation      education      sex
0     51     Private   Machine-op-inspct        HS-grad   Female
1     19     Private               Sales           11th     Male
2     19     Private               Sales        HS-grad     Male
3     22     Private               Sales           11th     Male
4     22     Private               Sales        HS-grad     Male
5     40     Private     Exec-managerial        HS-grad     Male
6     17     Private   Handlers-cleaners           10th     Male
7     61     Private        Craft-repair        7th-8th     Male
8     58           ?                   ?   Some-college   Female
10    26           ?                   ?   Some-college   Female
12    37   Local-gov       Other-service        HS-grad     Male
13    22     Private        Adm-clerical      Assoc-voc   Female
```

0.2 Exercise 3

Create a series and practice basic arithmetic steps

[140]:
```
# Create a series and practice basic arithmetic steps

import pandas as pd  #import library

# Creating Series 1
series1_data = [7.3, -2.5, 3.4, 1.5]
series1_index = ['a', 'c', 'd', 'e']
series1 = pd.Series(series1_data, index=series1_index)  # create the pandas␣
 ↪series

# Creating Series 2
series2_data = [-2.1, 3.6, -1.5, 4, 3.1]
series2_index = ['a', 'c', 'e', 'f', 'g']
series2 = pd.Series(series2_data, index=series2_index)  # create the pandas␣
 ↪series

# Adding Series 1 and Series 2
addition_result = series1 + series2
print("Addition Result:")
print(addition_result)

# Subtracting Series 1 from Series 2
subtraction_result = series2 - series1
print("\nSubtraction Result:")
print(subtraction_result)
```

```
Addition Result:
a    5.2
c    1.1
d    NaN
e    0.0
f    NaN
g    NaN
dtype: float64

Subtraction Result:
a   -9.4
c    6.1
d    NaN
e   -3.0
f    NaN
g    NaN
dtype: float64
```