AbhijitMandal_DSC540_Week3-4Ex

April 4, 2021

0.0.1 DSC 540 Week 3-4

Abhijit Mandal

0.0.2 Activity 5: Generating Statistics from a CSV File

- Load the necessary libraries.
- Read in the Boston housing dataset (given as a .csv file) from the local directory.
- Check the first 10 records. Find the total number of records.
- Create a smaller DataFrame with columns that do not include CHAS, NOX, B, and LSTAT.
- Check the last seven records of the new DataFrame you just created.
- Plot the histograms of all the variables (columns) in the new DataFrame.
- Plot them all at once using a for loop. Try to add a unique title to a plot.
- Create a scatter plot of crime rate versus price.
- Plot using log10(crime) versus price.
- Calculate some useful statistics, such as mean rooms per dwelling, median age, mean distances to five Boston employment centers, and the percentage of houses with a low price (< $20,000).

0.0.3 Load the necessary libraries.

[100]:

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

0.0.4 Read in the Boston housing dataset (given as a .csv file) from the local directory.

[101]:

```
bostonHousingDF=pd.read_csv("../Boston_housing.csv")
```

0.0.5 Check first 10 records

[102]:

```
bostonHousingDF.head(10)
```

[102]:

```
      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
```

```
5  0.02985   0.0   2.18     0  0.458  6.430   58.7  6.0622    3  222     18.7
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

        B  LSTAT  PRICE
0  396.90   4.98   24.0
1  396.90   9.14   21.6
2  392.83   4.03   34.7
3  394.63   2.94   33.4
4  396.90   5.33   36.2
5  394.12   5.21   28.7
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
```

0.0.6 Find the total number of records.

[103]:

```
bostonHousingDF.shape
```

[103]:

```
(506, 14)
```

0.0.7 Create a smaller DataFrame with columns which do not include 'CHAS', 'NOX', 'B', and 'LSTAT'

[104]:

```
bostonHousingSmallDF =␣
 ↪bostonHousingDF[['CRIM','ZN','INDUS','RM','AGE','DIS','RAD','TAX','PTRATIO','PRICE']]
```

0.0.8 Check the last 7 records of the new DataFrame you just created

[105]:

```
bostonHousingSmallDF.tail(7)
```

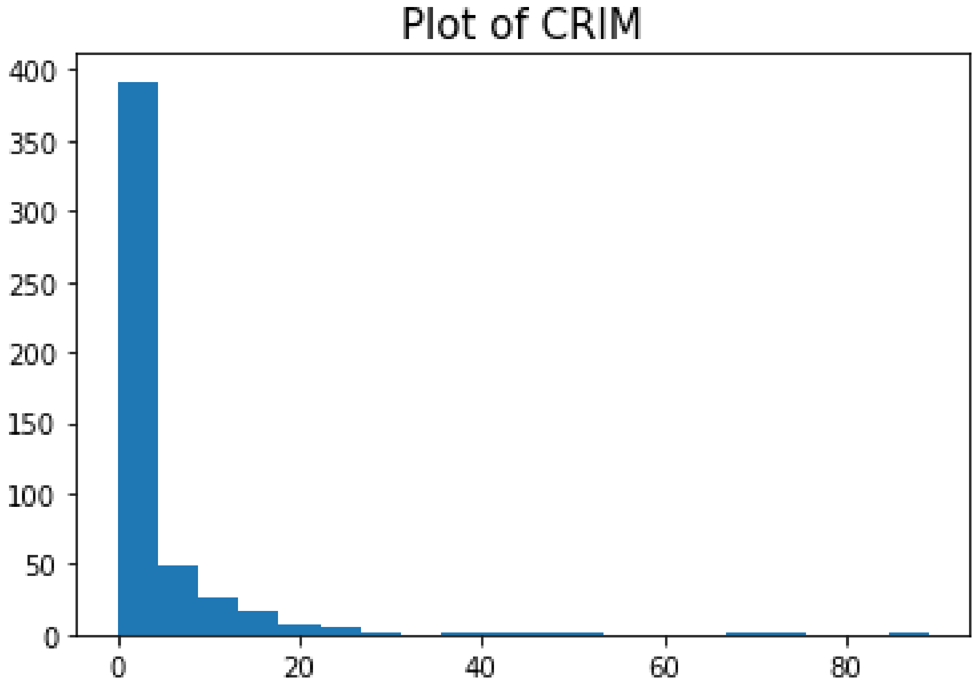
[105]:

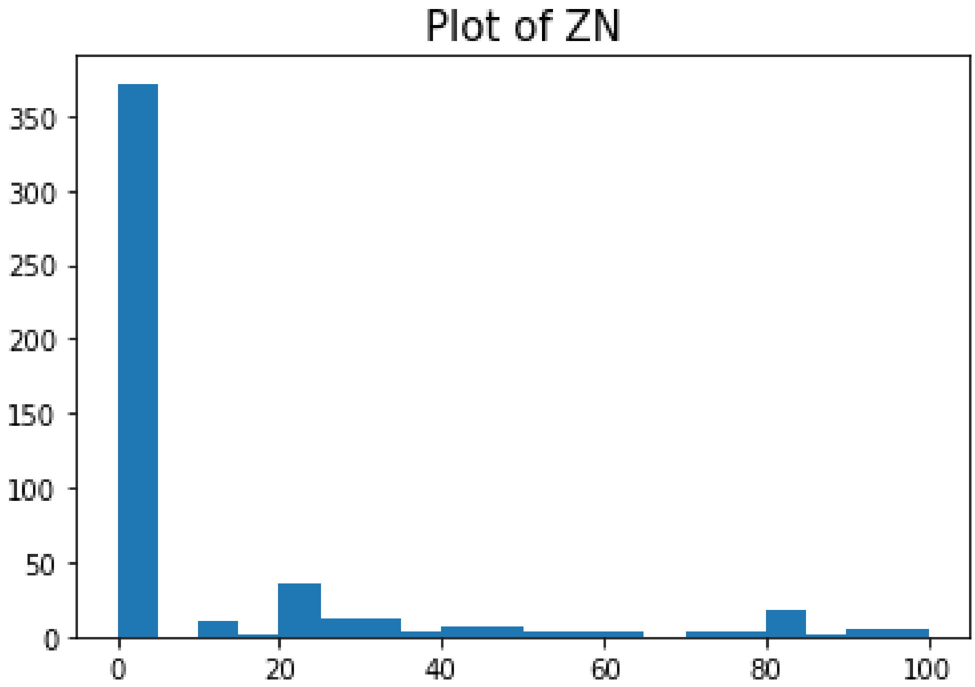
```
        CRIM   ZN  INDUS     RM   AGE     DIS  RAD  TAX  PTRATIO  PRICE
499  0.17783  0.0   9.69  5.569  73.5  2.3999    6  391     19.2   17.5
500  0.22438  0.0   9.69  6.027  79.7  2.4982    6  391     19.2   16.8
501  0.06263  0.0  11.93  6.593  69.1  2.4786    1  273     21.0   22.4
502  0.04527  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
```

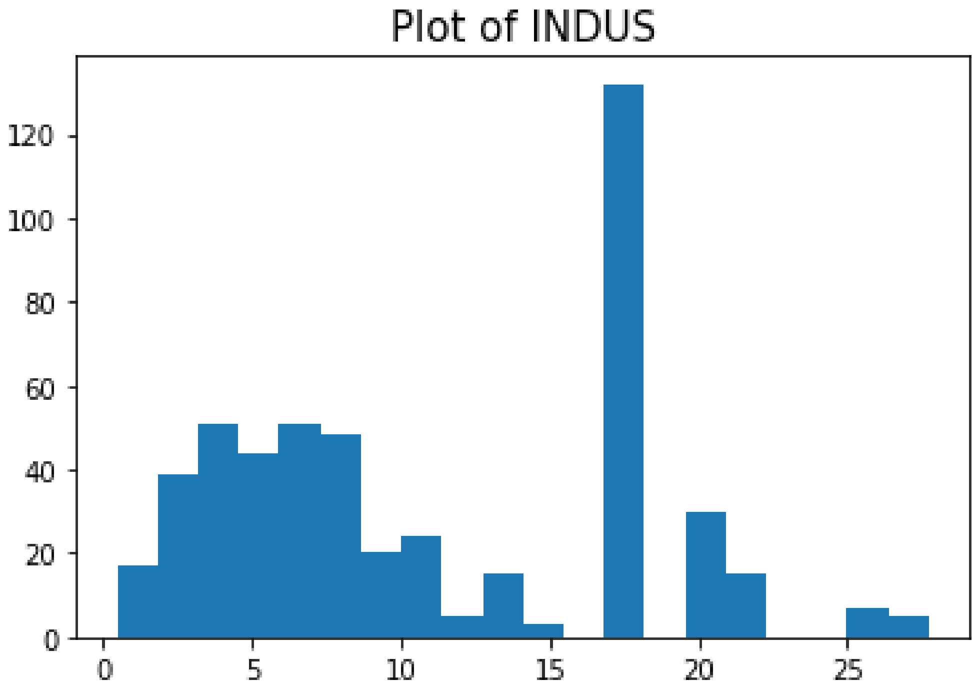
0.0.9 Can you plot histograms of all the variables (columns) in the new DataFrame?

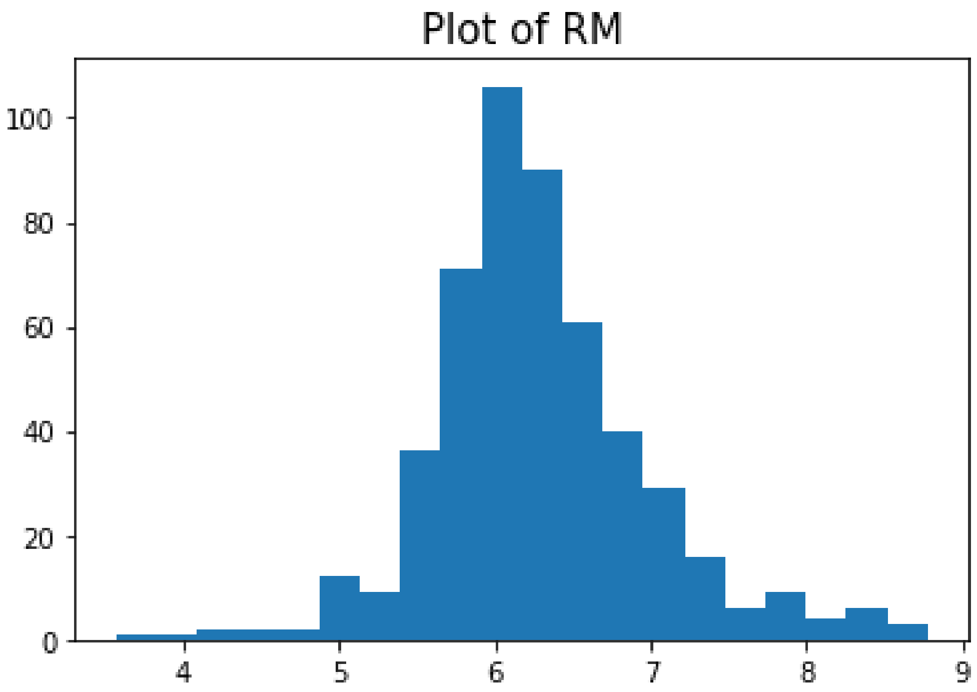
[106]:

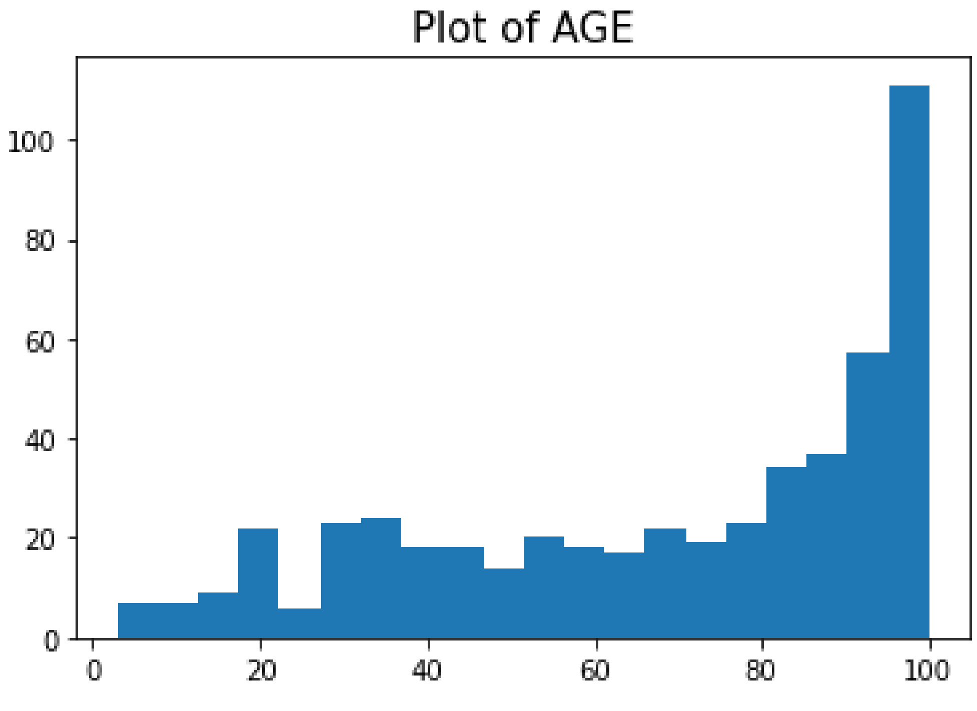
```
for c in bostonHousingSmallDF.columns:
    plt.title("Plot of "+c,fontsize=15)
    plt.hist(bostonHousingSmallDF[c],bins=20)
    plt.show()
```

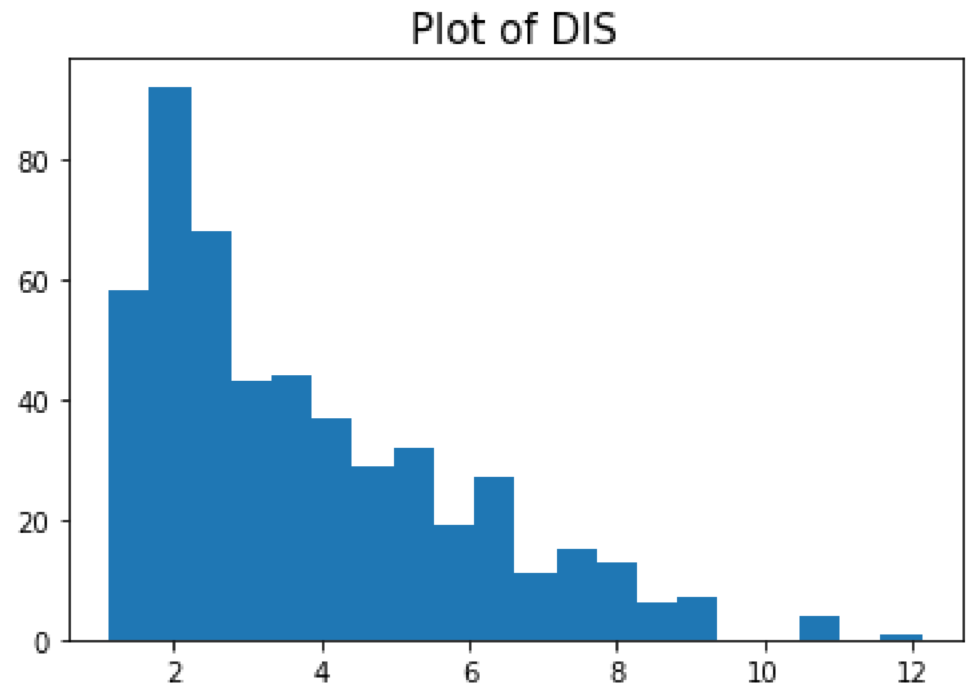


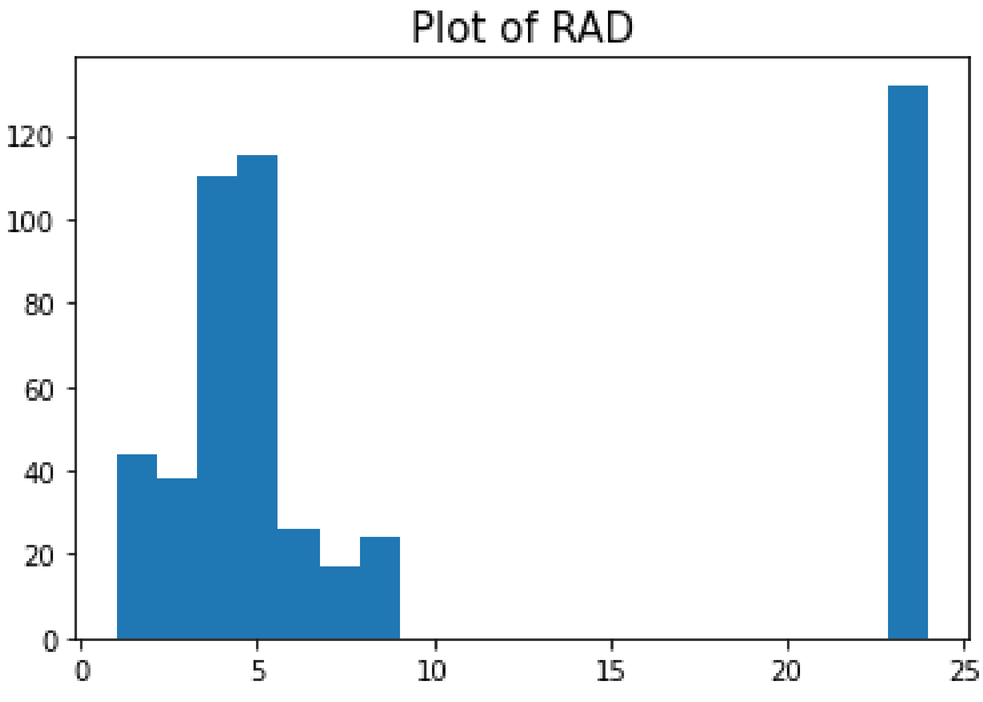

Plot of ZN
0
50
100
150
200
250
300
350
0
20
40
60
80
100

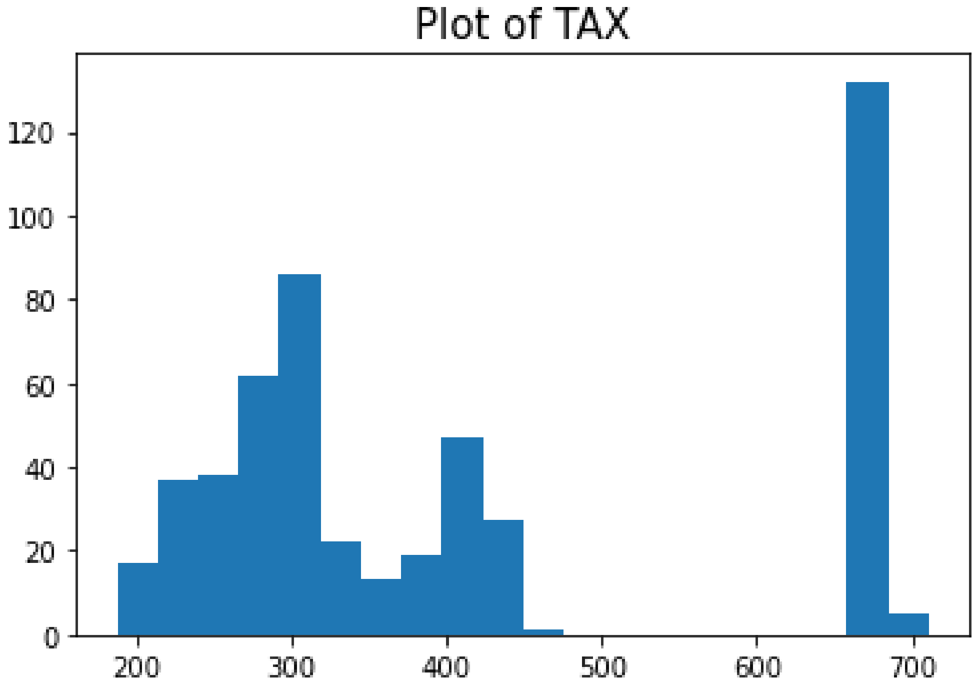

Plot of INDUS
0
20
40
60
80
100
120
0
5
10
15
20
25

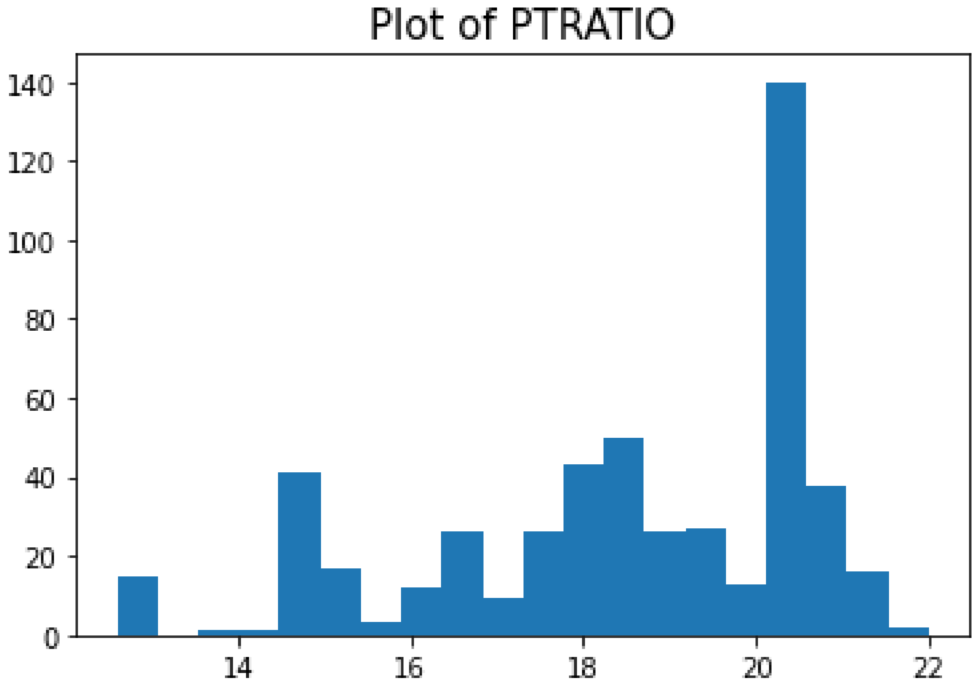

Plot of RM
100
80
60
40
20
0
4
5
6
7
8
9

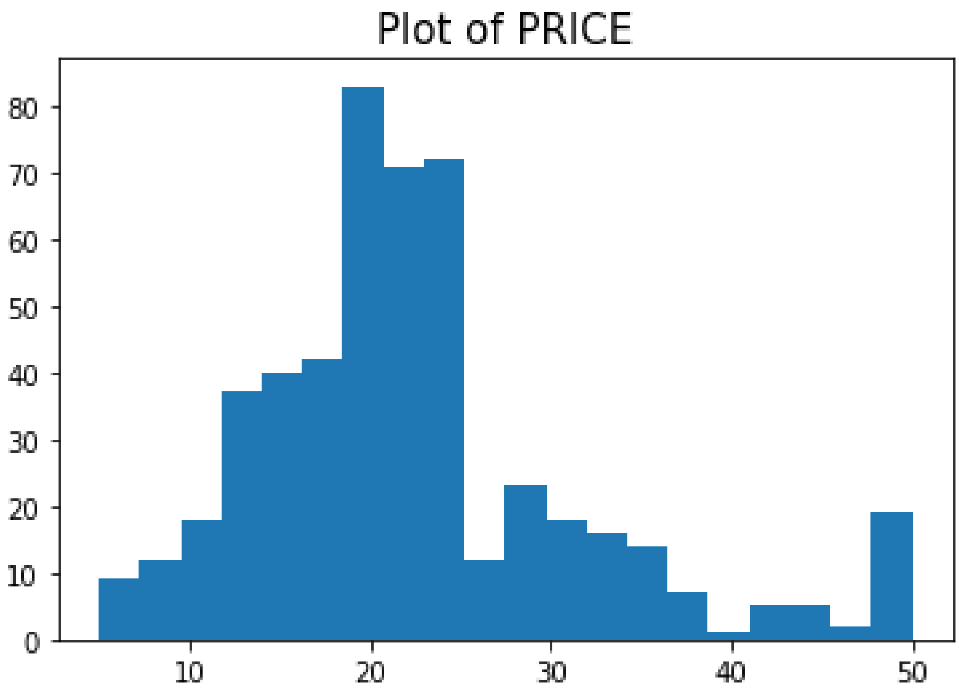

Plot of AGE
100
80
60
40
20
0
0
20
40
60
80
100


Plot of DIS
0
20
40
60
80
2
4
6
8
10
12


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


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

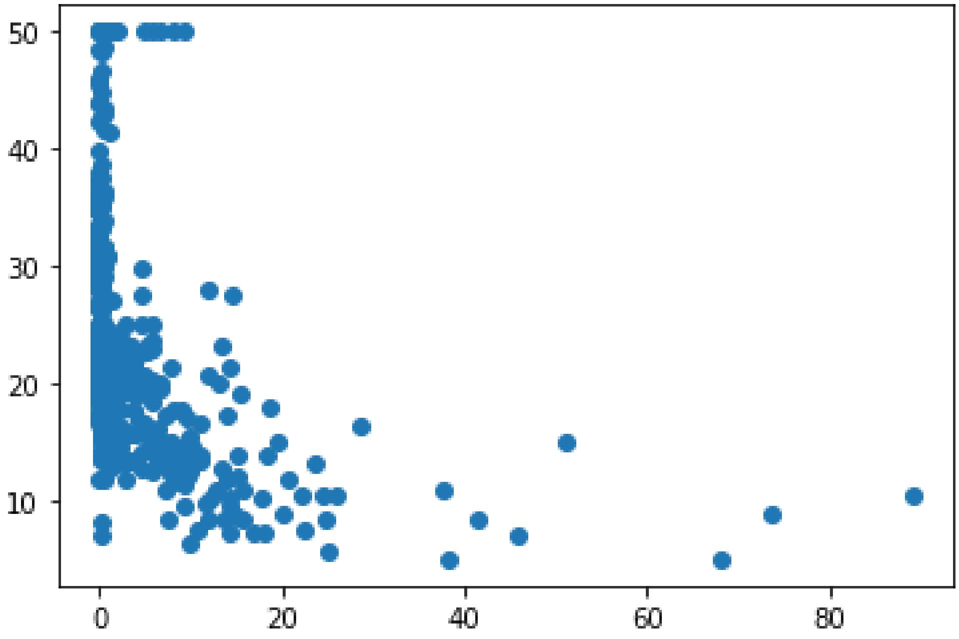

Plot of PTRATIO
0
20
40
60
80
100
120
140
14
16
18
20
22



0.0.10 Crime rate could be an indicator of house price (people don't want to live in high-crime areas). Create a scatter plot of crime rate vs. Price.

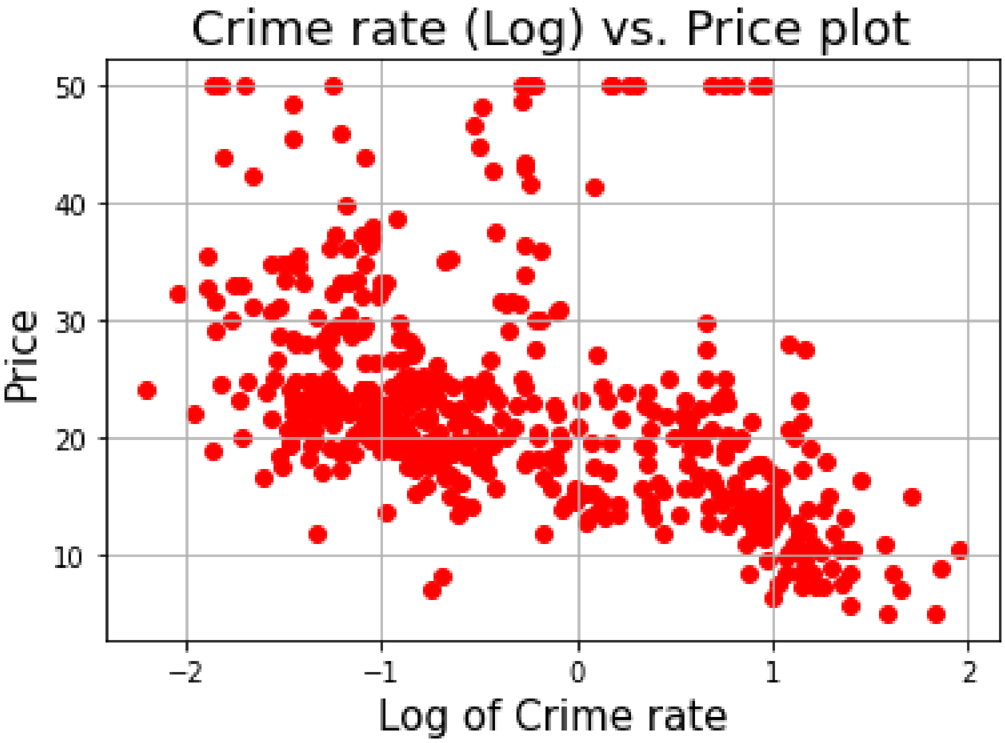
[107]:

```
plt.scatter(bostonHousingSmallDF['CRIM'],bostonHousingSmallDF['PRICE'])
plt.show()
```



0.0.11 We can understand the relationship better if we plot log10(crime) vs. Price

[108]:
```
plt.scatter(np.
 ↪log10(bostonHousingSmallDF['CRIM']),bostonHousingSmallDF['PRICE'],c='red')
plt.title("Crime rate (Log) vs. Price plot", fontsize=18)
plt.xlabel("Log of Crime rate",fontsize=15)
plt.ylabel("Price",fontsize=15)
plt.grid(True)
plt.show()
```



0.0.12 Calculate the mean rooms per dwelling?

[109]:
```
bostonHousingSmallDF['RM'].mean()
```

[109]:
```
6.284634387351788
```

0.0.13 Calculate median Age?

[110]:
```
bostonHousingSmallDF['AGE'].median()
```

[110]:
```
77.5
```

0.0.14 Calculate average (mean) distances to five Boston employment centres?

[111]:
```
bostonHousingSmallDF['DIS'].mean()
```

[111]:
```
3.795042687747034
```

0.0.15 calculate the percentage of houses with low price (< $20,000)?

[112]:
```
low_price=bostonHousingSmallDF['PRICE']<20
# This creates a Boolean array of True, False
print(low_price)
pcnt=low_price.mean()*100
print("\nPercentage of house with <20,000 price is: ",pcnt)
```

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

Percentage of house with <20,000 price is:  41.50197628458498
```

Activity 6: Working with the Adult Income Dataset (UCI)

- Load the necessary libraries.
- Read the adult income dataset from the following URL: https://github.com/TrainingByPackt/Data-Wrangling-with-Python/blob/master/Chapter04/Activity06/
- Create a script that will read a text file line by line.
- Add a name of Income for the response variable to the dataset.
- Find the missing values.
- Create a DataFrame with only age, education, and occupation by using subsetting.
- Plot a histogram of age with a bin size of 20.
- Create a function to strip the whitespace characters.

- Use the apply method to apply this function to all the columns with string values, create a new column, copy the values from this new column to the old column, and drop the new column.
- Find the number of people who are aged between 30 and 50.
- Group the records based on age and education to find how the mean age is distributed.
- Group by occupation and show the summary statistics of age. Find which profession has the oldest workers on average and which profession has its largest share of the workforce above the 75th percentile.
- Use subset and groupby to find outliers.
- Plot the values on a bar chart.
- Merge the data using common keys.

0.0.16 Read in the adult income data set (given as a .csv file) from the local directory and check first 5 records

[113]:

```
df = pd.read_csv("../adult_income_data.csv")
df.head()
```

[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

         Adm-clerical   Not-in-family   White     Male   2174   0   40  \
0     Exec-managerial         Husband   White     Male      0   0   13
1   Handlers-cleaners   Not-in-family   White     Male      0   0   40
2   Handlers-cleaners         Husband   Black     Male      0   0   40
3      Prof-specialty            Wife   Black   Female      0   0   40
4     Exec-managerial            Wife   White   Female      0   0   40

    United-States   <=50K
0   United-States   <=50K
1   United-States   <=50K
2   United-States   <=50K
3            Cuba   <=50K
4   United-States   <=50K
```

0.0.17 Create a script that will read a text file line by line and extracts the first line, which is the header of the .csv file:

[114]:

```
names = []
with open('../adult_income_names.txt','r') as f:
    for line in f:
        f.readline()
        var=line.split(":")[0]
        names.append(var)
names
```

[114]:
```
['age',
 'workclass',
 'fnlwgt',
 'education',
 'education-num',
 'marital-status',
 'occupation',
 'relationship',
 'sex',
 'race',
 'capital-gain',
 'capital-loss',
 'hours-per-week',
 'native-country']
```

0.0.18 Add a name ("Income") for the response variable (last column) to the dataset and read it again with the column names supplied

[115]:
```
names.append('Income')
```

[116]:
```
df = pd.read_csv("../adult_income_data.csv",names=names)
df.head()
```

[116]:
```
   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

        marital-status          occupation    relationship    sex     race  \
0        Never-married        Adm-clerical   Not-in-family  White     Male
1   Married-civ-spouse     Exec-managerial         Husband  White     Male
2             Divorced   Handlers-cleaners   Not-in-family  White     Male
3   Married-civ-spouse   Handlers-cleaners         Husband  Black     Male
4   Married-civ-spouse      Prof-specialty            Wife  Black   Female

   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
```

0.0.19 Show a statistical summary of the data set. Did you notice only a small number of columns are included?

[117]:
```
df.describe()
```

[117]:
```
                age        fnlwgt  education-num  capital-gain  capital-loss  \
count  32561.000000  3.256100e+04   32561.000000  32561.000000  32561.000000
mean      38.581647  1.897784e+05      10.080679   1077.648844     87.303830
std       13.640433  1.055500e+05       2.572720   7385.292085    402.960219
min       17.000000  1.228500e+04       1.000000      0.000000      0.000000
25%       28.000000  1.178270e+05       9.000000      0.000000      0.000000
50%       37.000000  1.783560e+05      10.000000      0.000000      0.000000
75%       48.000000  2.370510e+05      12.000000      0.000000      0.000000
max       90.000000  1.484705e+06      16.000000  99999.000000   4356.000000

       hours-per-week
count    32561.000000
mean        40.437456
std         12.347429
min          1.000000
25%         40.000000
50%         40.000000
75%         45.000000
max         99.000000
```

0.0.20 Many variables in the dataset have multiple factors or classes. Can you write a loop to count and print them?

[118]:
```
var_cls =␣
↪['workclass','education','marital-status','occupation','relationship','race','sex','native-
for v in var_cls:
    classes=df[v].unique()
    num_classes = df[v].nunique()
    print("There are {} classes in the \"{}\" column. They are: {}".
↪format(num_classes,v,classes))
    print("-"*100)
```

```
There are 9 classes in the "workclass" column. They are: [' State-gov' ' Self-
emp-not-inc' ' Private' ' Federal-gov' ' Local-gov'
 ' ?' ' Self-emp-inc' ' Without-pay' ' Never-worked']
--------------------------------------------------------------------------------
--------------------
There are 16 classes in the "education" column. They are: [' Bachelors' ' HS-
grad' ' 11th' ' Masters' ' 9th' ' Some-college'
 ' Assoc-acdm' ' Assoc-voc' ' 7th-8th' ' Doctorate' ' Prof-school'
 ' 5th-6th' ' 10th' ' 1st-4th' ' Preschool' ' 12th']
--------------------------------------------------------------------------------
```

```
--------------------
There are 7 classes in the "marital-status" column. They are: [' Never-married'
' Married-civ-spouse' ' Divorced'
 ' Married-spouse-absent' ' Separated' ' Married-AF-spouse' ' Widowed']
--------------------------------------------------------------------------------
--------------------
There are 15 classes in the "occupation" column. They are: [' Adm-clerical' '
Exec-managerial' ' Handlers-cleaners' ' Prof-specialty'
 ' Other-service' ' Sales' ' Craft-repair' ' Transport-moving'
 ' Farming-fishing' ' Machine-op-inspct' ' Tech-support' ' ?'
 ' Protective-serv' ' Armed-Forces' ' Priv-house-serv']
--------------------------------------------------------------------------------
--------------------
There are 6 classes in the "relationship" column. They are: [' Not-in-family' '
Husband' ' Wife' ' Own-child' ' Unmarried'
 ' Other-relative']
--------------------------------------------------------------------------------
--------------------
There are 2 classes in the "race" column. They are: [' Male' ' Female']
--------------------------------------------------------------------------------
--------------------
There are 5 classes in the "sex" column. They are: [' White' ' Black' ' Asian-
Pac-Islander' ' Amer-Indian-Eskimo' ' Other']
--------------------------------------------------------------------------------
--------------------
There are 42 classes in the "native-country" column. They are: [' United-States'
' Cuba' ' Jamaica' ' India' ' ?' ' Mexico' ' South'
 ' Puerto-Rico' ' Honduras' ' England' ' Canada' ' Germany' ' Iran'
 ' Philippines' ' Italy' ' Poland' ' Columbia' ' Cambodia' ' Thailand'
 ' Ecuador' ' Laos' ' Taiwan' ' Haiti' ' Portugal' ' Dominican-Republic'
 ' El-Salvador' ' France' ' Guatemala' ' China' ' Japan' ' Yugoslavia'
 ' Peru' ' Outlying-US(Guam-USVI-etc)' ' Scotland' ' Trinadad&Tobago'
 ' Greece' ' Nicaragua' ' Vietnam' ' Hong' ' Ireland' ' Hungary'
 ' Holand-Netherlands']
--------------------------------------------------------------------------------
--------------------
```

0.0.21 Is there any missing (NULL) data in the dataset? Write a single line of code to show this for all coumns

[119]:

```
df.isnull().sum()
```

[119]:

```
age               0
workclass         0
fnlwgt            0
education         0
education-num     0
```

```
marital-status    0
occupation        0
relationship      0
sex               0
race              0
capital-gain      0
capital-loss      0
hours-per-week    0
native-country    0
Income            0
dtype: int64
```

0.0.22 Create a DataFrame with only age, education, and occupation by using subsetting:

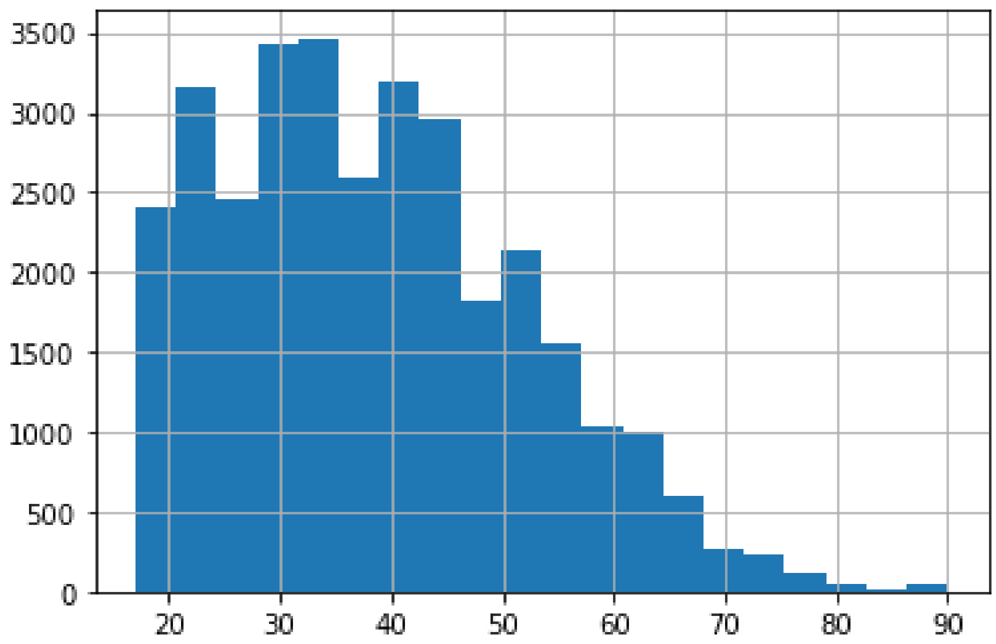
```
[124]: df_subset = df[['age','education','race','occupation']]
       df_subset.head()
```

```
[124]:    age  education    race         occupation
       0   39  Bachelors    Male       Adm-clerical
       1   50  Bachelors    Male    Exec-managerial
       2   38    HS-grad    Male  Handlers-cleaners
       3   53       11th    Male  Handlers-cleaners
       4   28  Bachelors  Female     Prof-specialty
```

0.0.23 Show the histogram of age with bin size = 20

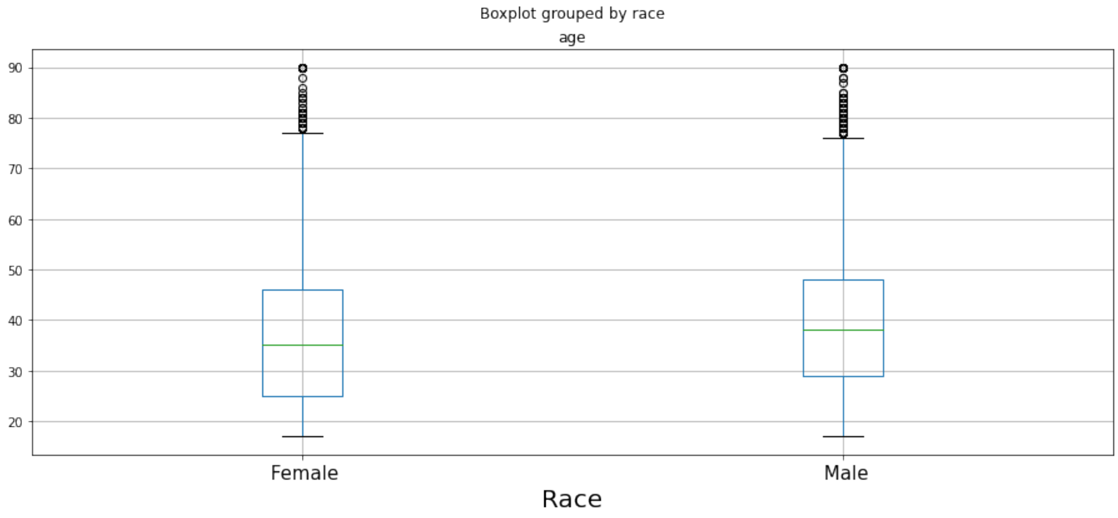
```
[125]: df_subset['age'].hist(bins=20)
```

```
[125]: <AxesSubplot:>
```



0.0.24 Show boxplots of age grouped by race (Use a long figure size 15x6 and make x ticks font size 15)

[126]:
```
df_subset.boxplot(column='age',by='race',figsize=(15,6))
plt.xticks(fontsize=15)
plt.xlabel("Race",fontsize=20)
plt.show()
```



0.0.25 Create a function to strip the whitespace characters

[127]:

```
def strip_whitespace(s):
    return s.strip()
```

0.0.26 Use the 'apply' method to apply this function to all the columns with string values, create a new column, copy the values from this new column to the old column, and drop the new column.

[128]:

```
df_subset['education_stripped']=df['education'].apply(strip_whitespace)
df_subset['education']=df_subset['education_stripped']
df_subset.drop(labels=['education_stripped'],axis=1,inplace=True)


df_subset['occupation_stripped']=df['occupation'].apply(strip_whitespace)
df_subset['occupation']=df_subset['occupation_stripped']
df_subset.drop(labels=['occupation_stripped'],axis=1,inplace=True)


df_subset['race_stripped']=df['race'].apply(strip_whitespace)
df_subset['race']=df_subset['race_stripped']
df_subset.drop(labels=['race_stripped'],axis=1,inplace=True)
```

```
<ipython-input-128-5e98db29da69>: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
  df_subset['education_stripped']=df['education'].apply(strip_whitespace)
<ipython-input-128-5e98db29da69>: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
  df_subset['education']=df_subset['education_stripped']
/Users/abhijitmandal/opt/anaconda3/lib/python3.8/site-
packages/pandas/core/frame.py:4163: 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
  return super().drop(
<ipython-input-128-5e98db29da69>: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
  df_subset['occupation_stripped']=df['occupation'].apply(strip_whitespace)
<ipython-input-128-5e98db29da69>:7: 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
  df_subset['occupation']=df_subset['occupation_stripped']
<ipython-input-128-5e98db29da69>:11: 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
  df_subset['race_stripped']=df['race'].apply(strip_whitespace)
<ipython-input-128-5e98db29da69>:12: 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
  df_subset['race']=df_subset['race_stripped']
```

0.0.27 Find the number of people who are aged between 30 and 50 (inclusive) by using the following command

[129]:
```
df_filtered=df_subset[(df_subset['age']>=30) & (df_subset['age']<=50)]
```

[130]:
```
df_filtered.head()
```

[130]:
```
   age  education    race         occupation
0   39  Bachelors    Male       Adm-clerical
1   50  Bachelors    Male    Exec-managerial
2   38    HS-grad    Male  Handlers-cleaners
5   37    Masters  Female    Exec-managerial
6   49        9th  Female      Other-service
```

0.0.28 Find the shape of the filtered DataFrame and specify the index of the tuple as 0 to return the first element

[131]:
```
data1=df_filtered.shape[0]
data1
```

[131]:
```
16390
```

0.0.29 Print the number of black people aged between 30 and 50 using the following command

[132]:

```
print("There are {} people of age between 30 and 50 in this dataset.".
 ↪format(data1))
```

```
There are 16390 people of age between 30 and 50 in this dataset.
```

0.0.30 Group the records based on occupation to find how the mean age is distributed:

[133]:

```
 df_subset.groupby('occupation').describe()['age']
```

[133]:

```
                     count       mean        std   min   25%   50%   75%   max
occupation
?                   1843.0  40.882800  20.336350  17.0  21.0  35.0  61.0  90.0
Adm-clerical        3770.0  36.964456  13.362998  17.0  26.0  35.0  46.0  90.0
Armed-Forces           9.0  30.222222   8.089774  23.0  24.0  29.0  34.0  46.0
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
Farming-fishing      994.0  41.211268  15.070283  17.0  29.0  39.0  52.0  90.0
Handlers-cleaners   1370.0  32.165693  12.372635  17.0  23.0  29.0  39.0  90.0
Machine-op-inspct   2002.0  37.715285  12.068266  17.0  28.0  36.0  46.0  90.0
Other-service       3295.0  34.949621  14.521508  17.0  22.0  32.0  45.0  90.0
Priv-house-serv      149.0  41.724832  18.633688  17.0  24.0  40.0  57.0  81.0
Prof-specialty      4140.0  40.517633  12.016676  17.0  31.0  40.0  48.0  90.0
Protective-serv      649.0  38.953775  12.822062  17.0  29.0  36.0  47.0  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
Transport-moving    1597.0  40.197871  12.450792  17.0  30.0  39.0  49.0  90.0
```

0.0.31 Group by occupation and show the summary statistics of age. Find which profession has the oldest workers on average and which profession has its largest share of workforce above the 75th percentile:

[134]:

```
df_subset.groupby('occupation').describe()['age']
```

[134]:

```
                     count       mean        std   min   25%   50%   75%   max
occupation
?                   1843.0  40.882800  20.336350  17.0  21.0  35.0  61.0  90.0
Adm-clerical        3770.0  36.964456  13.362998  17.0  26.0  35.0  46.0  90.0
Armed-Forces           9.0  30.222222   8.089774  23.0  24.0  29.0  34.0  46.0
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
Farming-fishing      994.0  41.211268  15.070283  17.0  29.0  39.0  52.0  90.0
Handlers-cleaners   1370.0  32.165693  12.372635  17.0  23.0  29.0  39.0  90.0
Machine-op-inspct   2002.0  37.715285  12.068266  17.0  28.0  36.0  46.0  90.0
Other-service       3295.0  34.949621  14.521508  17.0  22.0  32.0  45.0  90.0
```

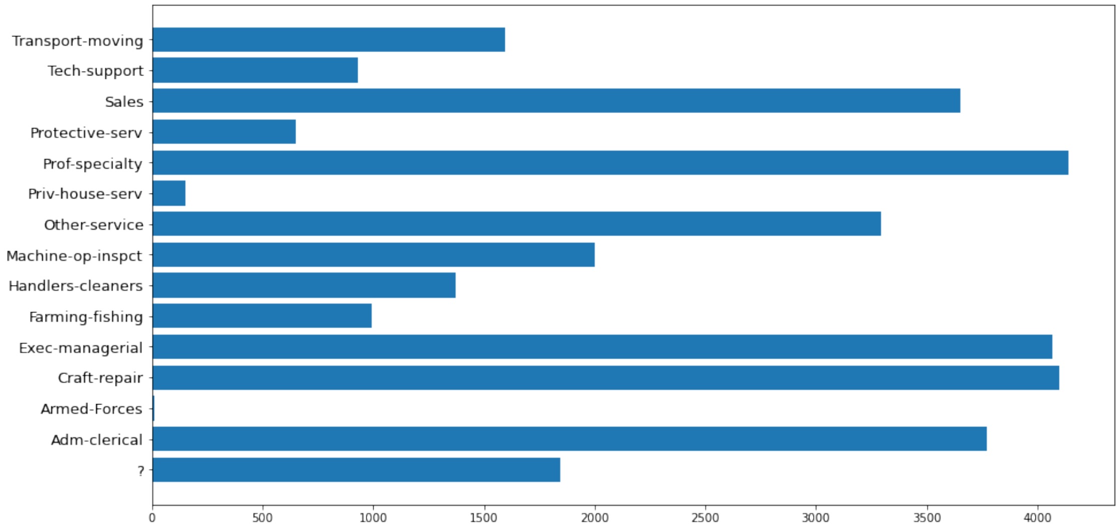
```
Priv-house-serv     149.0  41.724832  18.633688  17.0  24.0  40.0  57.0  81.0
Prof-specialty     4140.0  40.517633  12.016676  17.0  31.0  40.0  48.0  90.0
Protective-serv     649.0  38.953775  12.822062  17.0  29.0  36.0  47.0  90.0
Sales              3650.0  37.353973  14.186352  17.0  25.0  35.0  47.0  90.0
Tech-support        928.0  37.022629  11.316594  17.0  28.0  36.0  44.0  73.0
Transport-moving   1597.0  40.197871  12.450792  17.0  30.0  39.0  49.0  90.0
```

0.0.32 Use subset and groupby to find the outliers

[135]:
```
occupation_stats= df_subset.groupby( 'occupation').describe()['age']
```

0.0.33 Plot the values on a bar chart

[136]:
```
plt.figure(figsize=(15,8))
plt.barh(y=occupation_stats.index, width=occupation_stats['count'])
plt.yticks(fontsize=13)
plt.show()
```



0.0.34 Practice Merging by common keys: Suppose you are given two datasets where the common key is occupation. Can you merge them?¶

[137]:
```
df_1 = df[['age','workclass','occupation']].sample(5,random_state=101)
df_1.head()
```

[137]:
```
       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
```

[138]:

```python
df_2 = df[['education','race','occupation']].sample(5,random_state=101)
df_2.head()
```

[138]:

```
      education     race          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
```

[139]:

```python
df_merged = pd.merge(df_1,df_2,on='occupation',how='inner').drop_duplicates()
df_merged
```

[139]:

```
   age workclass          occupation education     race
0   51   Private   Machine-op-inspct   HS-grad   Female
1   19   Private               Sales      11th     Male
2   40   Private     Exec-managerial   HS-grad     Male
3   17   Private   Handlers-cleaners      10th     Male
4   61   Private        Craft-repair   7th-8th     Male
```

0.0.35 Create a series and practice basic arithmetic steps

- Series 1 = 7.3, -2.5, 3.4, 1.5
    i. Index = ‘a’, ‘c’, ‘d’, ‘e’
- Series 2 = -2.1, 3.6, -1.5, 4, 3.1
    i. Index = ‘a’, ‘c’, ‘e’, ‘f’, ‘g’
- Add Series 1 and Series 2 together and print the results
- Subtract Series 1 from Series 2 and print the results

[142]:

```python
# series 1
data1 = np.array([7.3, -2.5, 3.4, 1.5])

# providing an index
series1 = pd.Series(data1, index =['a', 'c', 'd', 'e'])
print(series1)
```

```
a    7.3
c   -2.5
d    3.4
e    1.5
dtype: float64
```

[143]:

```python
# series 2
data2 = np.array([-2.1, 3.6, -1.5, 4, 3.1])
```

```python
# providing an index
series2 = pd.Series(data2, index =['a', 'c', 'e', 'f','g'])
print(series2)
```

```
a   -2.1
c    3.6
e   -1.5
f    4.0
g    3.1
dtype: float64
```

[145]:

```python
# series 1 + series 2
series3 = series1.add(series2, fill_value=10);
print(series3)
```

```
a     5.2
c     1.1
d    13.4
e     0.0
f    14.0
g    13.1
dtype: float64
```

[147]:

```python
# series 1 - series 2
series4 = series1.subtract(series2, fill_value=10);
print(series4)
```

```
a    9.4
c   -6.1
d   -6.6
e    3.0
f    6.0
g    6.9
dtype: float64
```

[ ]:

```python

```