Chitramoy Mukherjee

DSC 540-T302

Week-3 and Week-4

Date : 09/22/2023

Activity-5

In [2]:

```
# Suppose you are working with the famous Boston housing price(from 1960)
# the machine learning community.Many regression problem can be formulated
# run on this dataset. You will do perform basic data wrangling activity(i
# dataset by reading it as a pandas dataframe.

# Load necessary libraries

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

In [3]:

```
# Read Boston housig dataset and read 10 records
import csv
file_path = 'C:\\Users\\14024\\OneDrive\\Desktop\\MS-DSC\\DSC-540\\Week-4\
file_path
# Open the CSV file
with open(file_path, 'r') as file:
    # Create a CSV reader
    csv_reader = csv.reader(file)

    # Skip the header row if it exists
    next(csv_reader, None)

    # Read and print the first 10 records
    for i, row in enumerate(csv_reader):
        if i < 10:
            print(row)
        else:
            break
```

```
['0.00632', '18', '2.31', '0', '0.538', '6.575', '65.2', '4.09', '1', '2
96', '15.3', '396.9', '4.98', '24']
['0.02731', '0', '7.07', '0', '0.469', '6.421', '78.9', '4.9671', '2',
'242', '17.8', '396.9', '9.14', '21.6']
['0.02729', '0', '7.07', '0', '0.469', '7.185', '61.1', '4.9671', '2',
'242', '17.8', '392.83', '4.03', '34.7']
['0.03237', '0', '2.18', '0', '0.458', '6.998', '45.8', '6.0622', '3',
'222', '18.7', '394.63', '2.94', '33.4']
['0.06905', '0', '2.18', '0', '0.458', '7.147', '54.2', '6.0622', '3',
'222', '18.7', '396.9', '5.33', '36.2']
['0.02985', '0', '2.18', '0', '0.458', '6.43', '58.7', '6.0622', '3', '2
22', '18.7', '394.12', '5.21', '28.7']
['0.08829', '12.5', '7.87', '0', '0.524', '6.012', '66.6', '5.5605',
'5', '311', '15.2', '395.6', '12.43', '22.9']
['0.14455', '12.5', '7.87', '0', '0.524', '6.172', '96.1', '5.9505',
'5', '311', '15.2', '396.9', '19.15', '27.1']
['0.21124', '12.5', '7.87', '0', '0.524', '5.631', '100', '6.0821', '5',
'311', '15.2', '386.63', '29.93', '16.5']
['0.17004', '12.5', '7.87', '0', '0.524', '6.004', '85.9', '6.5921',
'5', '311', '15.2', '386.71', '17.1', '18.9']
```

In [4]:

```
# Read the CSV file into a DataFrame
df = pd.read_csv(file_path)

# Get the total number of records
total_records = len(df)

print(f'Total number of records in the CSV file: {total_records}')
```

```
Total number of records in the CSV file: 506
```

In [5]:

```
# Create a smaller dataframe by dropping CHAS, NOX, B and LSTAT

print("\n 4 columns dropped by using df.drop() method\n")

# List of column names to drop
columns_to_drop = ['CHAS', 'NOX', 'B', 'LSTAT']

df = df.drop(columns=columns_to_drop)

print(df)
```

```
 4 columns dropped by using df.drop() method

        CRIM    ZN  INDUS     RM   AGE     DIS  RAD  TAX  PTRATIO  PRICE
0    0.00632  18.0   2.31  6.575  65.2  4.0900    1  296     15.3   24.0
1    0.02731   0.0   7.07  6.421  78.9  4.9671    2  242     17.8   21.6
2    0.02729   0.0   7.07  7.185  61.1  4.9671    2  242     17.8   34.7
3    0.03237   0.0   2.18  6.998  45.8  6.0622    3  222     18.7   33.4
4    0.06905   0.0   2.18  7.147  54.2  6.0622    3  222     18.7   36.2
..       ...   ...    ...    ...   ...     ...  ...  ...      ...    ...
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

[506 rows x 10 columns]
```

In [6]:

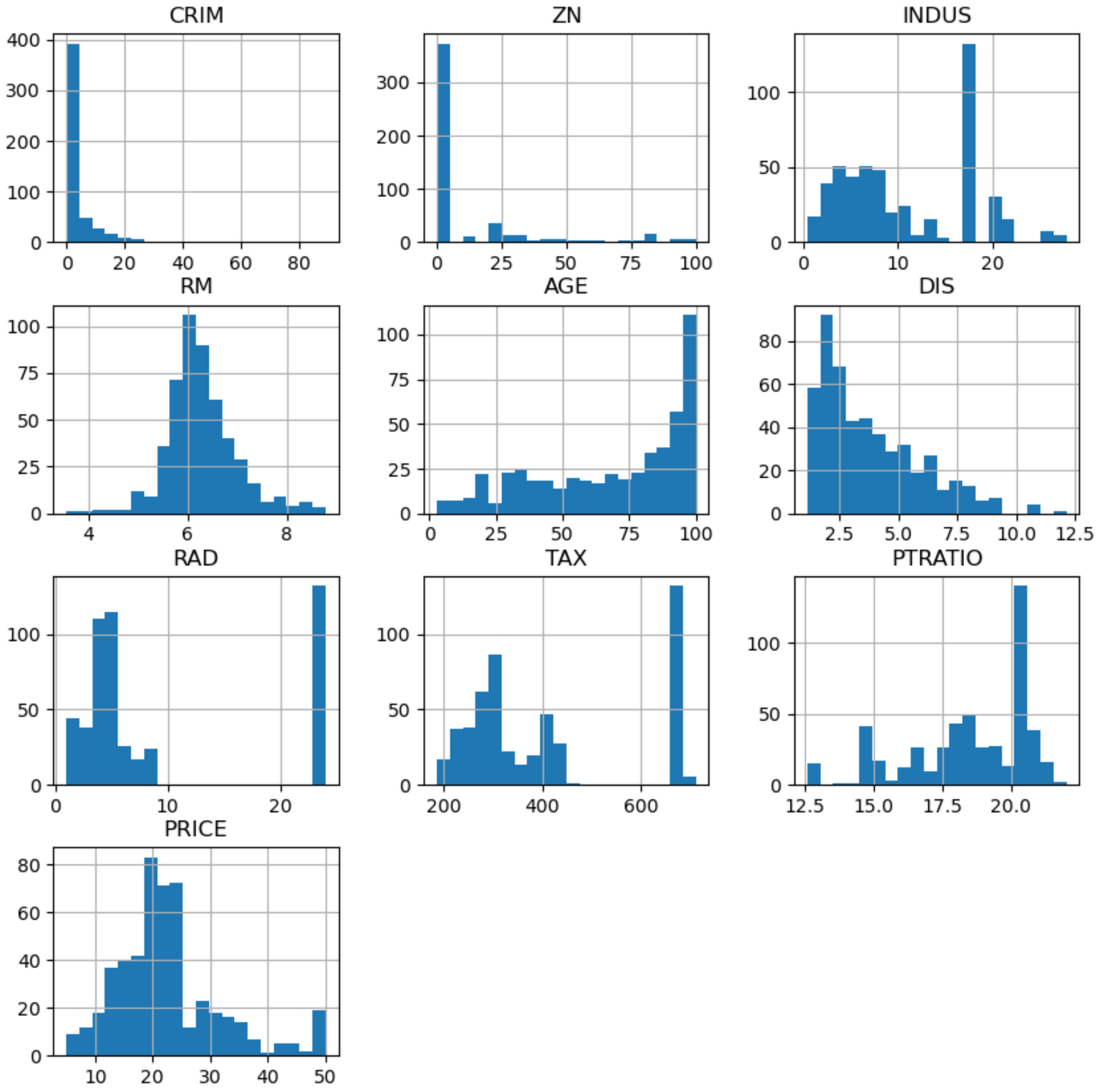
```
# Check the last seven records from the new Dataframe

df.tail(7)
```

Out[6]:

	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

In [7]:

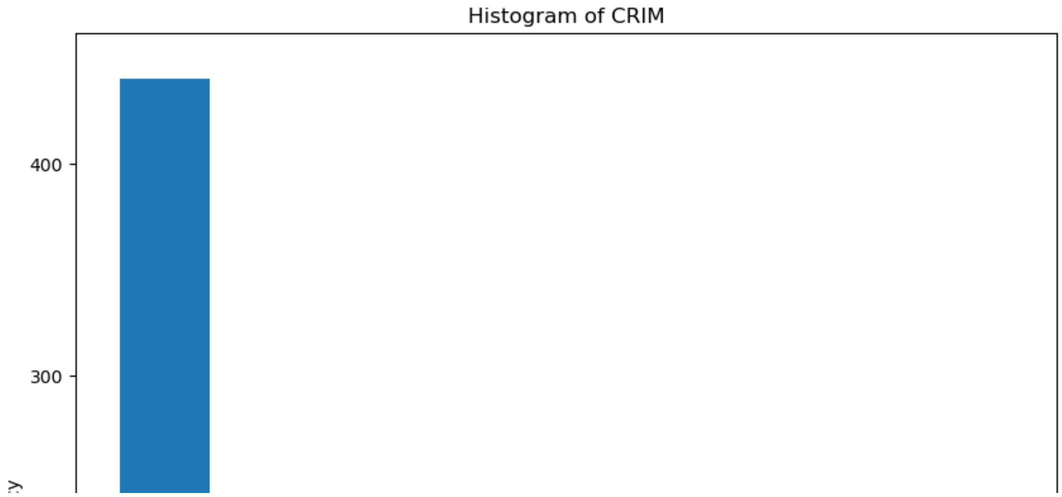
```
# Plot histograms for all columns
df.hist(bins=20, figsize=(10, 10))
plt.show()
```



In [97]:

```python
# Get the column names
columns = df.columns
print(columns)
# Loop through each column and plot a histogram
for column in columns:
    plt.figure(figsize=(10, 10))
    plt.hist(df[column], bins=10)
    plt.title(f'Histogram of {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.show()
```

```
Index(['CRIM', 'ZN', 'INDUS', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRAT
IO',
       'PRICE'],
      dtype='object')
```

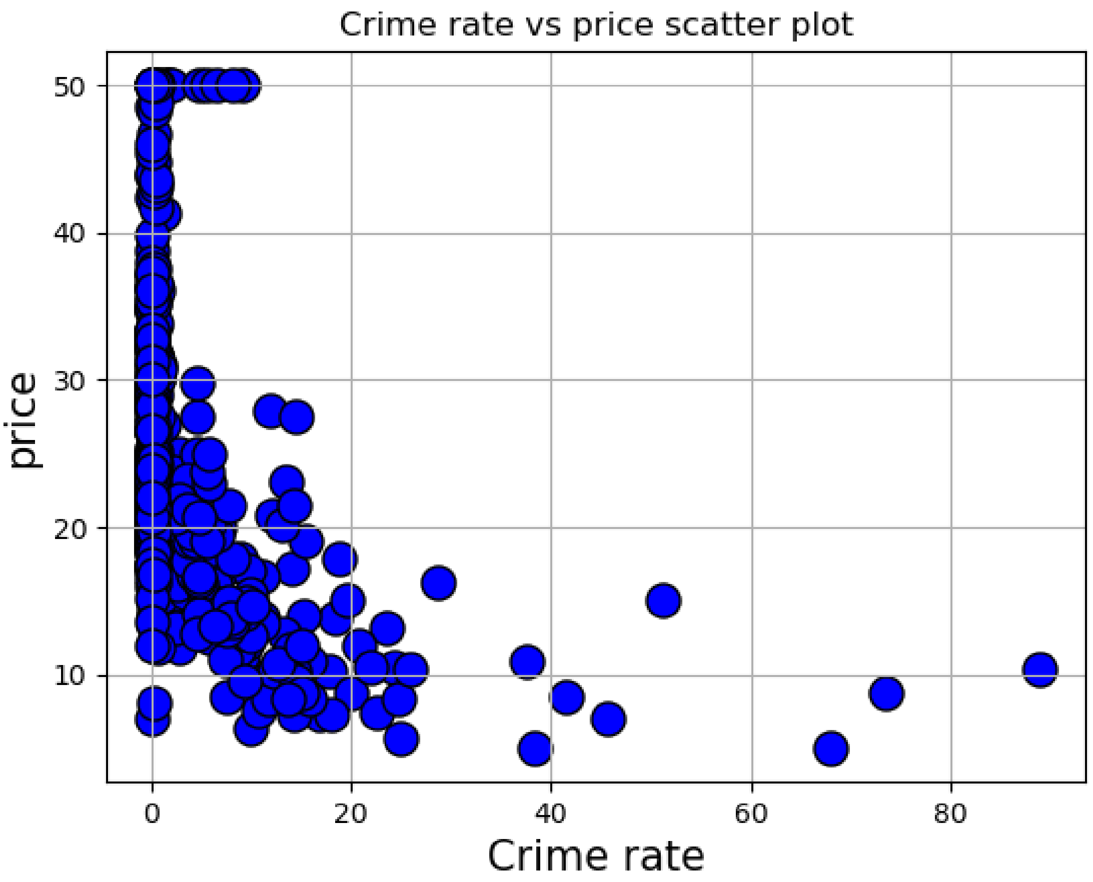


In [8]:

```
# Create a scatter plot of crime rate vs price

df.plot.scatter('CRIM','PRICE', s=150,c='blue',edgecolor='k',linewidths =
plt.grid(True)
plt.title('Crime rate vs price scatter plot')
plt.xlabel("Crime rate",fontsize=15)
plt.ylabel("price",fontsize=15)
plt.show
```

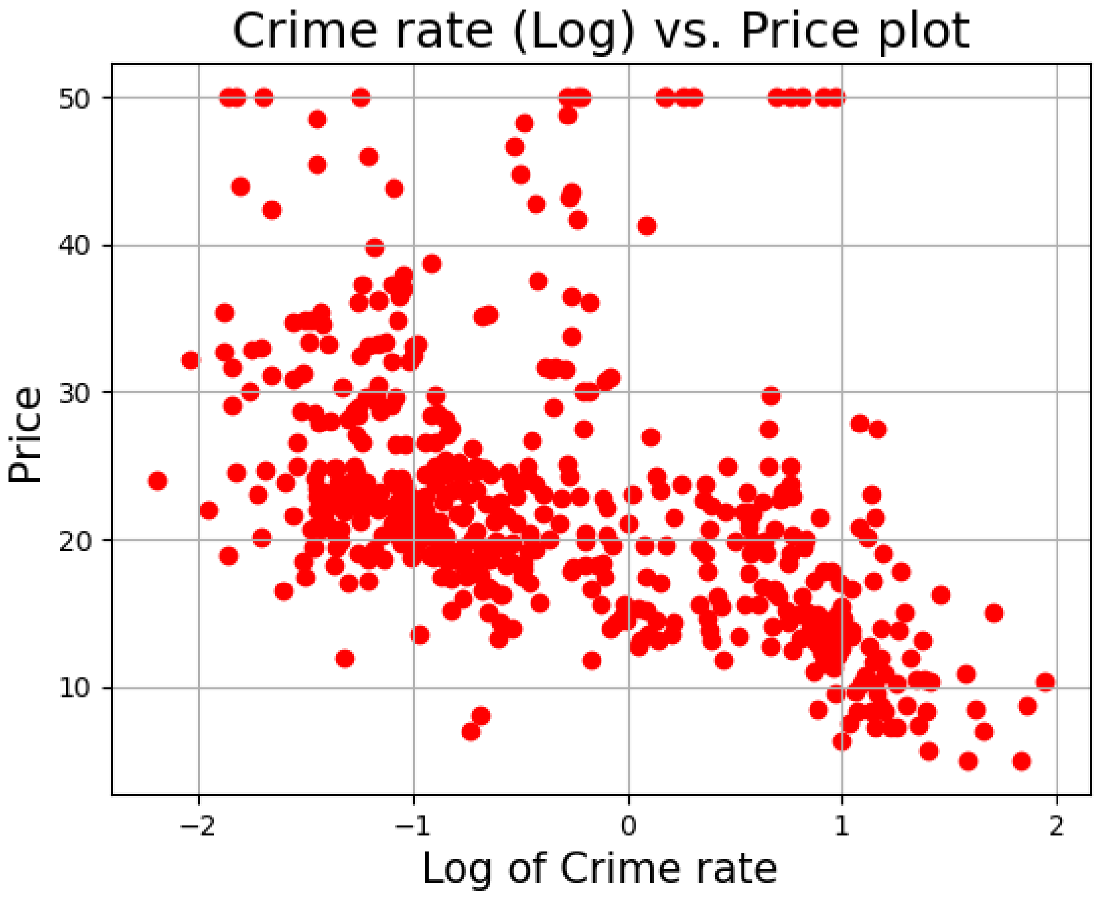
Out[8]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [9]:

```
# Create the plot
plt.scatter(np.log10(df['CRIM']),df['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()
```



In [10]:

```
#Mean rooms per dwelling
df['RM'].mean()
```

Out[10]:

```
6.284634387351787
```

In [106]:

```
# Median Age
median_age = df['AGE'].median()

print('Median of AGE: {median_age}')
```

```
Median of Column1: 77.5
```

In [19]:

```
# Mean distance to five Boston employment centers
df['DIS'].mean()
```

Out[19]:

```
3.795042687747034
```

In [23]:

```python
# Percentage of houses with a low price (<$20,000)

low_price_house=df['PRICE']<20
prcnt=low_price_house.mean()*100
print("\nPercentage of house with <20,000 price is: ",prcnt)
```

```

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

Activity 6

In [8]:

```python
#Read adult income dataset

df = pd.read_csv("C:\\Users\\14024\\OneDrive\\Desktop\\MS-DSC\\DSC-540\\We
df.head()
```

Out[8]:

	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	Male	2174	0	40
0	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	Male	0	0	13
1	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	Male	0	0	40
2	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Male	0	0	40
3	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Female	0	0	40
4	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	Female	0	0	40

In [13]:

```python
# Create a script to read a text file line by line.
names = []
with open('C:\\Users\\14024\\OneDrive\\Desktop\\MS-DSC\\DSC-540\\Week-4\\a
    for line in f:
        f.readline()
        var=line.split(":")[0]
        names.append(var)
```

In [14]:

```
names
```

Out[14]:

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

In [15]:

```
# Add a name of Income for the response variable to the dataset.

names.append('Income')
df = pd.read_csv("C:\\Users\\14024\\OneDrive\\Desktop\\MS-DSC\\DSC-540\\We
df.head()
```

Out[15]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	se
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	Ma
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	Ma
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	Ma
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Ma
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Fema

In [16]:

```
# Find the missing values

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

Out[16]:

```
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
```

In [18]:

```
# Create a dataframe with only age, education and occupation by subsetting

df1 = df[['age','education','occupation']]
df1.head()
```

Out[18]:

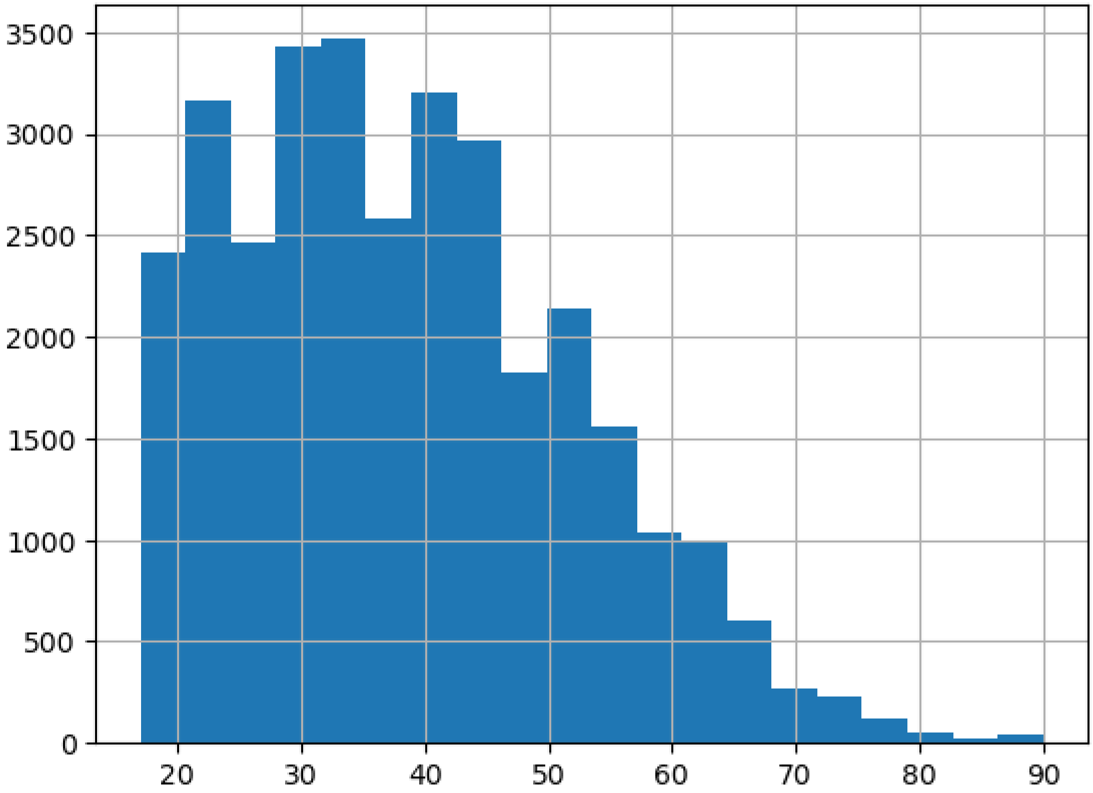
	age	education	occupation
0	39	Bachelors	Adm-clerical
1	50	Bachelors	Exec-managerial
2	38	HS-grad	Handlers-cleaners
3	53	11th	Handlers-cleaners
4	28	Bachelors	Prof-specialty

In [19]:

```
# Plot the histogram of age with bin size 20.

df1['age'].hist(bins=20)
```

Out[19]:

```
<AxesSubplot:>
```



In [22]:

```
# Create a function to strip the whitespace characters.Use apply method to
# Create a new column, copy the values from this new column to the old col

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

# Apply strip_whitespace function on  education column
df1['education_stripped']=df['education'].apply(strip_whitespace)
df1['education']=df1['education_stripped']
df1.drop(labels=['education_stripped'],axis=1,inplace=True)

# Apply strip_whitespace function on  occupation column
df1['occupation_stripped']=df['occupation'].apply(strip_whitespace)
df1['occupation']=df1['occupation_stripped']
df1.drop(labels=['occupation_stripped'],axis=1,inplace=True)
```

```
C:\Users\14024\AppData\Local\Temp\ipykernel_9460\2882816121.py:7: Settin
gWithCopyWarning:
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-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  df1['education_stripped']=df['education'].apply(strip_whitespace)
C:\Users\14024\AppData\Local\Temp\ipykernel_9460\2882816121.py:8: Settin
gWithCopyWarning:
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-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  df1['education']=df1['education_stripped']
C:\Users\14024\AppData\Local\Temp\ipykernel_9460\2882816121.py:9: Settin
gWithCopyWarning:
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-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  df1.drop(labels=['education_stripped'],axis=1,inplace=True)
C:\Users\14024\AppData\Local\Temp\ipykernel_9460\2882816121.py:12: Setti
ngWithCopyWarning:
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-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  df1['occupation_stripped']=df['occupation'].apply(strip_whitespace)
C:\Users\14024\AppData\Local\Temp\ipykernel_9460\2882816121.py:13: Setti
ngWithCopyWarning:
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-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  df1['occupation']=df1['occupation_stripped']
C:\Users\14024\AppData\Local\Temp\ipykernel_9460\2882816121.py:14: Setti
ngWithCopyWarning:
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-d
ocs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#return
ing-a-view-versus-a-copy)
  df1.drop(labels=['occupation_stripped'],axis=1,inplace=True)
```

In [26]:

```
# Find the number of people who are aged between 30 and 50.

df2=df1[(df1['age']>=30) & (df1['age']<=50)]
df2.head()
```

Out[26]:

	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

In [27]:

```
# Group the records based on age and education to find how the mean age is

df1.groupby(['age','education']).mean()
```

```
C:\Users\14024\AppData\Local\Temp\ipykernel_9460\867248816.py:3: FutureW
arning: Dropping invalid columns in DataFrameGroupBy.mean is deprecated.
In a future version, a TypeError will be raised. Before calling .mean, s
elect only columns which should be valid for the function.
  df1.groupby(['age','education']).mean()
```

Out[27]:

age	education
17	**10th**
	11th
	12th
	5th-6th
	7th-8th
...	**...**
90	**Bachelors**
	HS-grad
	Masters
	Prof-school
	Some-college

965 rows × 0 columns

In [28]:

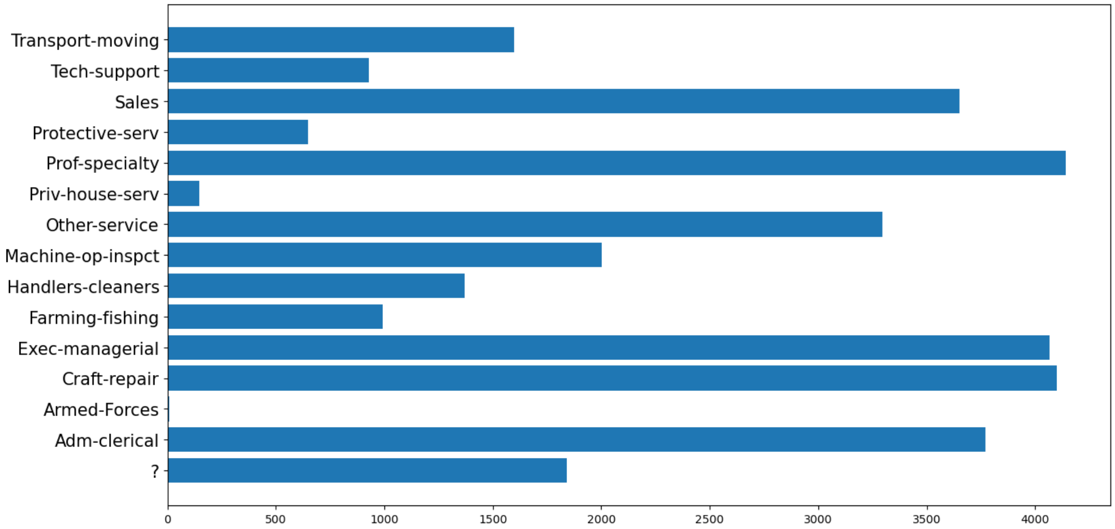
```
# Group by occupation and show the summary statistics of age.Find which pr
# has its largest share of the workforce above the 75th percentile.

df1.groupby('occupation').describe()['age']
```

Out[28]:

	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

In [31]:

```
# Use subset and group by to find outliers. Plot values in bar chart.

occupation_stat= df1.groupby('occupation').describe()['age']
plt.figure(figsize=(15,8))
plt.barh(y=occupation_stat.index,width=occupation_stat['count'])
plt.yticks(fontsize=15)
plt.show()
```



In [48]:

```
# Merge the data using column keys

df_random_1 = df[['age','workclass','education']].sample(10,random_state=1
df_random_1.head()

df_random_2 = df[['education','sex','occupation']].sample(10,random_state=
df_random_2.head()

df_merged = pd.merge(df_random_1,df_random_2,on='education',how='inner').c
df_merged
```

Out[48]:

	age	workclass	education	sex	occupation
0	27	?	12th	Male	?
1	24	Private	HS-grad	Male	Craft-repair
2	24	Private	HS-grad	Male	Tech-support
3	24	Private	HS-grad	Female	Exec-managerial
4	24	Private	HS-grad	Female	Other-service
5	30	Private	HS-grad	Male	Craft-repair
6	30	Private	HS-grad	Male	Tech-support
7	30	Private	HS-grad	Female	Exec-managerial
8	30	Private	HS-grad	Female	Other-service
9	51	Local-gov	HS-grad	Male	Craft-repair
10	51	Local-gov	HS-grad	Male	Tech-support
11	51	Local-gov	HS-grad	Female	Exec-managerial
12	51	Local-gov	HS-grad	Female	Other-service
13	23	Local-gov	HS-grad	Male	Craft-repair
14	23	Local-gov	HS-grad	Male	Tech-support
15	23	Local-gov	HS-grad	Female	Exec-managerial
16	23	Local-gov	HS-grad	Female	Other-service
17	20	Private	Some-college	Male	Craft-repair
18	20	Private	Some-college	Male	Transport-moving
19	54	Private	Some-college	Male	Craft-repair
20	54	Private	Some-college	Male	Transport-moving
21	52	Private	Bachelors	Female	Exec-managerial
22	52	Private	Bachelors	Male	Exec-managerial
23	51	Private	Bachelors	Female	Exec-managerial
24	51	Private	Bachelors	Male	Exec-managerial
25	45	Self-emp-inc	Doctorate	Male	Prof-specialty

In [43]:

```
# Create a series and practice basic arithmetic steps

data = [7.3, -2.5, 3.4, 1.5]
index = ['a', 'c', 'd', 'e']

series1 = pd.Series(data, index=index)
print(series1)

data = [-2.1, 3.6, -1.5, 4, 3.1]
index = [ 'a', 'c', 'e', 'f', 'g']

series2 = pd.Series(data, index=index)
print(series2)
```

```
a    7.3
c   -2.5
d    3.4
e    1.5
dtype: float64
a   -2.1
c    3.6
e   -1.5
f    4.0
g    3.1
dtype: float64
```

In [44]:

```
# Add Series 1 and Series 2

result = series1 + series2

print(result)
```

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

In [45]:

```
# Substract Series 1 and Series 2

result = series1 - series2

print(result)
```

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