Student Details

When submitting, fill your name and ID in this cell. Note that this is a markdown cell.

Student Name and ID: MyLastName, MY_FirstName [My_StudentID] same for your partner

Programming Assignment 1 SPRING 2022:

Exploratory Analysis over census data for salaries from the year 1994

Assignment Details

In this assignment, you will conduct a guided exploration over the given dataset, census1994 Dataset.

You will prepare a report with the following outline for each one of the dataset. Look at the following Example.

Introduction

Retrieving the Data

Glimpse of Data

Check for missing data

Data Exploration

You will learn and use some of the most common exploration/aggregation/descriptive operations. This should also help you learn most of the key functionalities in Python/Pandas, Weka and R. DO Task 1, Task 2, Task 3, Task 4 using Python/Pandas, Weka, R Tasks are described for Pandas, Ask yourself how do I perform tis tasks in Weka in R. Task 1 is in Pandas/Weka/R Task 2 is in Pandas/Weka/R Task 3 in in Pandas/Weka/R

You will also learn how to use visualization libraries to identify patterns in data that will help in your further data analysis. You will also explore most popular chart types and how to use different libraries and styles to make your visualizations more attractive.

Dataset Details

In this assignment, you will work on census1994 dataset. This data was extracted from the census bureau database. Extraction was done by Barry Becker from the 1994 Census database. The objective of the survey was to determine whether a person makes over 50K a year or not.

census 1994 Dataset

>50K, <=50K.

age: continuous.

workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.

fnlwgt: continuous.

education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.

education-num: continuous.

marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.

occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.

relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.

race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.

gender: Female, Male.

capital-gain: continuous.

capital-loss: continuous.

hours-per-week: continuous.

native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

Required Python Packages

You will use the packages imported below in this assignment. You will not require any other packages.

```
# special IPython command to prepare the notebook for matplotlib
%matplotlib inline

#Array processing
import numpy as np
#Data analysis, wrangling and common exploratory operations
import pandas as pd
from pandas import Series, DataFrame

#For visualization. Matplotlib for basic viz and seaborn for more
stylish figures
import matplotlib.pyplot as plt
# import seaborn as sns

#For some of the date operations
#import datetime
```

Reading Dataset

The Python code below reads the census1994 dataset into a Pandas data frame with the name df_census. For this code to work, the file 'census1994.csv' must be in the same folder as the notebook.

```
#read the csv file into a Pandas data frame
df_census = pd.read_csv('census1994.csv')

#return the first 5 rows of the dataset
df_census.head()
```

```
        Date  Age         WorkClass  fnlwgt   education  education-
num  \
0  3/20/1994   39         State-gov   77516   Bachelors
13
1  1/14/1994   50  Self-emp-not-inc   83311   Bachelors
13
2  8/14/1994   38           Private  215646     HS-grad
9
3  3/17/1994   53           Private  234721        11th
7
4  9/20/1994   28           Private  338409   Bachelors
13

        marital-status          occupation    relationship    race
gender  \
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   class

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
```

Task 1: Statistical Exploratory Data Analysis

Let us start with getting know the dataset. Your first task will be to get some basic information by using Pandas features.

```
#For each task below, look for a Pandas function to do the task.
#Replace None in each task with your code.

#Before starting with the tasks in the assignment, we need to remove
the rows with missing values
#write the code for it here and save the new data frame with the same
name as df_census.
###################begin your code.
df = df_census.replace(to_replace= "\\?", value= np.nan,regex= True)
dfClean= df.dropna().reset_index()
# dfClean.columns = dfClean.columns.str.lstrip()
dfClean.columns = dfClean.columns.str.replace('^ +', '_')
###################end your code.


#Task 1-a: Print the details of the df_census data frame (information
such as number of rows,columns, name of columns, etc)
print("Task 1-a: Details of df_census data frame are: \n", None)
# print(dfClean.info(verbose=None, buf=None, max_cols=None,
memory_usage=None, null_counts=None))
```

```
#census1994.csv.info(verbose=None, buf=None, max_cols=None,
memory_usage=None, null_counts=None)

#Task 1-b: Find the number of rows and columns in the df_census data
frame.
num_rows = len (dfClean)
num_cols = len(dfClean.columns)
print("\n\nTask 1-b: Number of rows:%s and number of columns:%s" %
(num_rows, num_cols))


#Task 1-c: Print the descriptive details (min, max, quartiles etc) for
'Age' column of the df_census
print("\n\nTask 1-c: Descriptive details of age is \n",
Series(dfClean['Age']).describe())


#Task 1-d: Print the number of unique values for 'education_num' and
'hours-per-week' columns
num_uniq_1 = dfClean['education-num'].unique()
num_uniq_2 = dfClean['hours-per-week'].unique()
print("\n\nTask 1-d: The number of unique 1### :", num_uniq_1)
print("Task 1-d: The number of unique 2### :", num_uniq_2)
```

```
Task 1-a: Details of df_census data frame are:
 None


Task 1-b: Number of rows:30162 and number of columns:17


Task 1-c: Descriptive details of age is
 count    30162.000000
mean        38.437902
std         13.134665
min         17.000000
25%         28.000000
50%         37.000000
75%         47.000000
max         90.000000
Name: Age, dtype: float64


Task 1-d: The number of unique 1### : [13  9  7 14  5 10 12  4 16 11
15  3  6  1  8  2]
Task 1-d: The number of unique 2### : [40 13 16 45 50 80 30 35 60 20
52 44 15 25 43 38 55 48 58 32 70 22 56 41
 28 36 24 46  2 42 12 65  1 34 75 98 33 54 10  6 64 19 18 72  8  9 47
37
 21 26 14  5  7 99 53 39 62 59 57 78 90 66 11 49 84 17 68  3 27 85 31
51
 77 63 23  4 87 88 73 89 97 94 29 96 67 82 86 91 81 76 92 61 74 95]
```

```
C:\Users\16824\AppData\Local\Temp/ipykernel_29288/1388007489.py:10:
FutureWarning: The default value of regex will change from True to
False in a future version.
  dfClean.columns = dfClean.columns.str.replace('^ +', '_')
```

Task 2: Aggregation & Filtering & Rank

In this task, we will perform some very high level aggregation and filtering operations. Then, we will apply ranking on the results for some tasks. Pandas has a convenient and powerful syntax for aggregation, filtering, and ranking. DO NOT write a for loop. Pandas has built-in functions for all tasks.

```
#Task 2-a: Find out the sum of Captial Gain for people with education
level as Bachelors and HS-Grad.
sum_capital_gain_bachelors = dfClean.loc[dfClean['education'] == '
Bachelors','capital-gain'].sum()
sum_capital_gain_HS_Grad =  dfClean.loc[dfClean['education'] == ' HS-
grad','capital-gain'].sum()
print ("Task 2-a: The sum of capital gain for education level as
bachelors is %s and as HS-Grad is %s"
       % (sum_capital_gain_bachelors, sum_capital_gain_HS_Grad))


#Task 2-b: Find out the total number of people surveyed in months may,
october and december.
#Create a new column for 'Survey_Month' by using 'Date' column
#write the code for extracting the month from the date column here

###############begin your code here
months={1: 'January',2 : 'February',3 : 'March',4: 'April', 5: 'May',
6: 'June', 7: 'July', 8: 'August', 9: 'September',
10 :'October',11 :'November', 12: 'December'}
monthR = pd.DatetimeIndex(dfClean['Date']).month
dfClean['Month']  = monthR.map(months)
###############send you code here
num_surveys_may = dfClean.loc[dfClean['Month'] ==
'May','Date'].count()
num_surveys_october = dfClean.loc[dfClean['Month'] ==
'October','Date'].count()
num_surveys_december = dfClean.loc[dfClean['Month'] ==
'December','Date'].count()
print ("\n\nTask 2-b: The total number of surveys in may is %s, in
october is %s, and in december is %s"
       % (num_surveys_may, num_surveys_october, num_surveys_december))
```

```
#Task 2-c: Let us now use multiple filtering criteria
# Find out the total number of surveys in september and november with
workclass as private and age less than 50.
num_surveys_september = dfClean.loc[(dfClean['Month'] == 'September')
& (dfClean['WorkClass'] == ' Private') & (dfClean['Age']<
50),'WorkClass'].count()
num_surveys_november = dfClean.loc[(dfClean['Month'] == 'November') &
(dfClean['WorkClass'] == ' Private') & (dfClean['Age']<
50),'WorkClass'].count()
print ("\n\nTask 2-c: The total number of surveys that meet the given
conditions in september is %s and in november is %s"
       % (num_surveys_september, num_surveys_november))


#Task 2-d: Find out 3 least surveyed education categories, print their
names and corresponding number of surveys for periods January-June and
July-December.
maskJantoJune = (pd.DatetimeIndex(dfClean['Date']).month < 7)
dataset_June_to_Jan= dfClean.loc[maskJantoJune]
maskJultoDec = (pd.DatetimeIndex(dfClean['Date']).month > 6) &
(pd.DatetimeIndex(dfClean['Date']).month < 13)
dataset_Jul_to_Dec= dfClean.loc[maskJultoDec]
top3_least_surveyed_Jan_June =
dataset_June_to_Jan.groupby('education').size().sort_values(ascending=
True).head(3).to_frame(name = 'Survey Count').reset_index()
top3_least_surveyed_July_December =
dataset_Jul_to_Dec.groupby('education').size().sort_values(ascending=T
rue).head(3).to_frame(name = 'Survey Count').reset_index()
print ("\n\nTask 2-d: \nThe top 3 least surveyed education categories
in January-June: \n%s \n\nThe top 3 least surveyed education
categories in July-December: \n%s"
                     %
(top3_least_surveyed_Jan_June,top3_least_surveyed_July_December))


#Task 2-e: Find out top 5 native-countries besides United-States,
print their names and number of surveys belonging to each.
top5_most_surveyed_native_countries = dfClean.loc[(dfClean['native-
country'] != ' United-States')].groupby('native-
country').size().sort_values(ascending=False).head(5).to_frame(name =
'Survey Count').reset_index()
print ("\n\nTask 2-e: \nThe top 5 most surveyed native countries : \n
%s"
                     % (top5_most_surveyed_native_countries))


#Task 2-f: Find out Top-5 native-countries with the most number of
```

```
samples belonging to class >50K
top5_native_countries = dfClean.loc[(dfClean['class'] == '
>50K')].groupby('native-country').size().sort_values(ascending=
False).head().to_frame(name = 'Survey Count').reset_index()
print ("\n\nTask 2-f: \nThe top 5 native countries with the most
number of surveys with class >50K: \n%s"
                     % (top5_native_countries))
```

```
Task 2-a: The sum of capital gain for education level as bachelors is
8751485 and as HS-Grad is 5799557


Task 2-b: The total number of surveys in may is 2510, in october is
2510, and in december is 2602


Task 2-c: The total number of surveys that meet the given conditions
in september is 1465 and in november is 1507


Task 2-d:
The top 3 least surveyed education categories in January-June:
    education  Survey Count
0   Preschool            19
1     1st-4th            74
2     5th-6th           156

The top 3 least surveyed education categories in July-December:
    education  Survey Count
0   Preschool            26
1     1st-4th            77
2     5th-6th           132


Task 2-e:
The top 5 most surveyed native countries :
  native-country  Survey Count
0         Mexico           610
1    Philippines           188
2        Germany           128
3    Puerto-Rico           109
4         Canada           107


Task 2-f:
The top 5 native countries with the most number of surveys with class
>50K:
   native-country  Survey Count
0   United-States          6995
```

```
1     Philippines            60
2         Germany            44
3           India            40
4          Canada            36
```

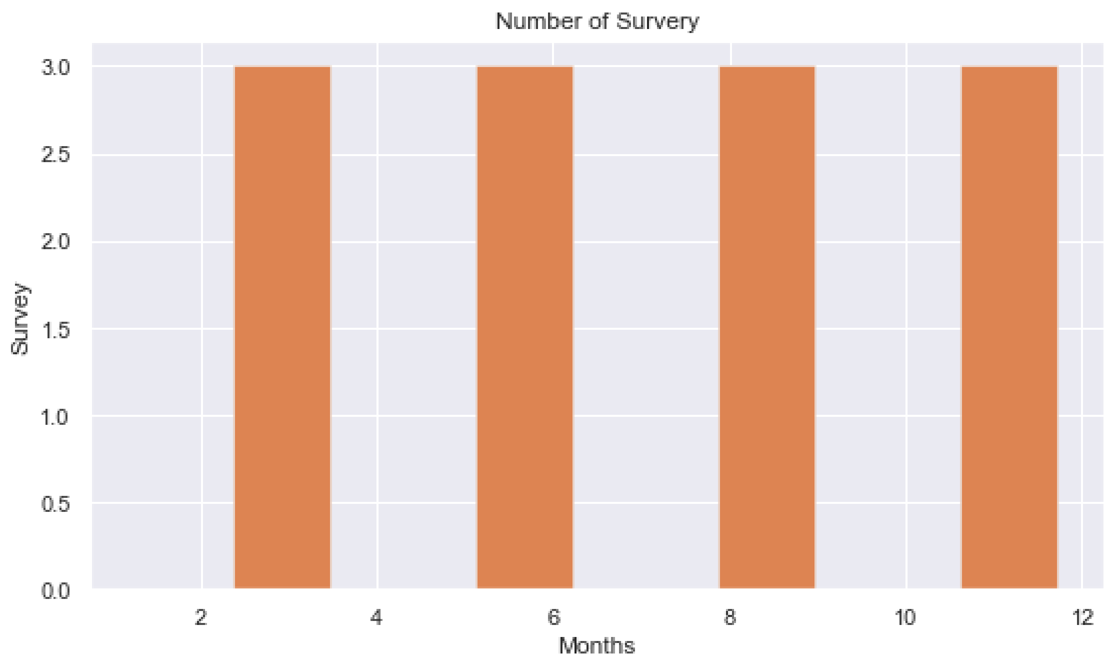
Task 3: Visualization

In this task, you will perform a number of visualization tasks to get some intuition about the data. Visualization is a key component of exploration. You can choose to use either Matplotlib or Seaborn for plotting. The default figures generated from Matplotlib might look a bit ugly. So you might want to try Seaborn to get better figures. Seaborn has a variety of styles. Feel free to experiment with them and choose the one you like. We have earmarked 10 points for the aesthetics of your visualizations.

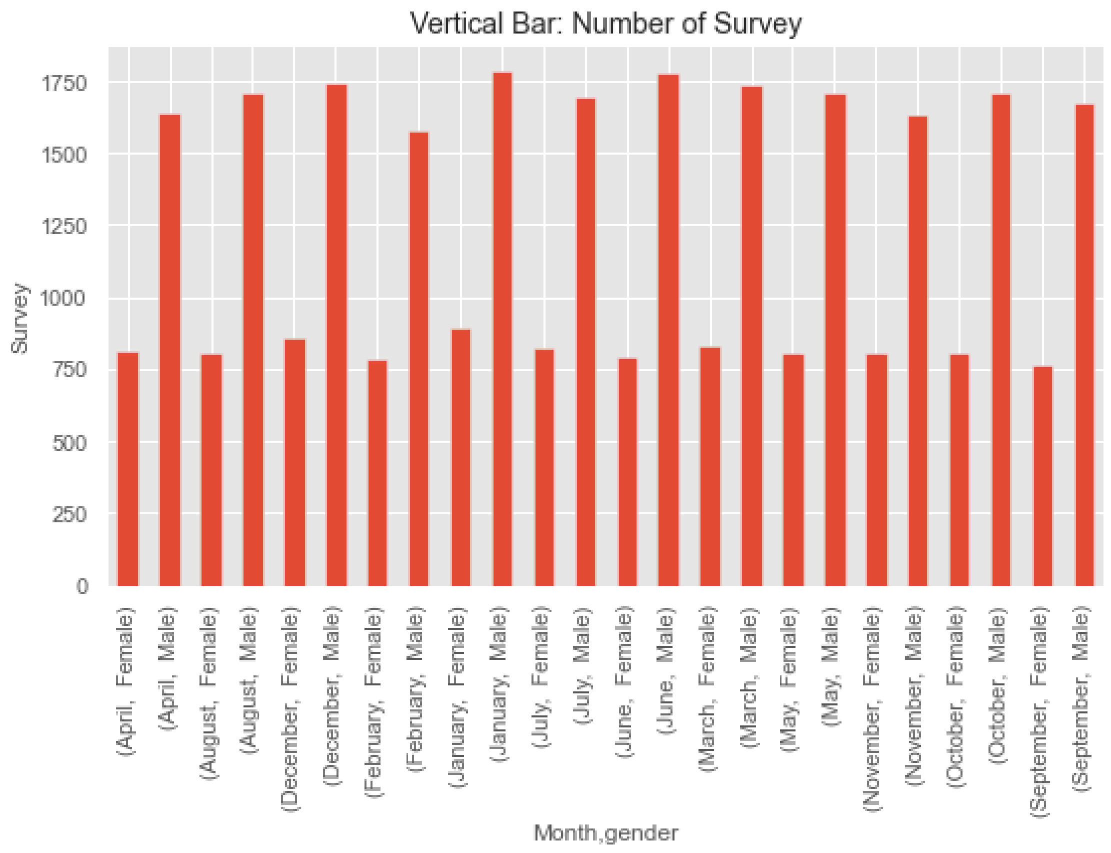
```
#Task 3-a: Draw a histogram for total number of surveys taken each
month. Dislpay months with their corresponding numbers(Eg: January is
1)
#########################begin code for Task 3-a
plt.rcParams["figure.figsize"]= (9,5)
values =
dfClean.groupby(pd.DatetimeIndex(dfClean['Date']).month).size().to_fra
me(name = 'SurveyCount').reset_index()
values = values.reindex(columns= ['SurveyCOunt', 'Date'])
plt.title('Number of Survery')
plt.xlabel('Months')
plt.ylabel('Survey')
plt.style.use('ggplot')
plt.hist(values, bins= 4)
plt.show()
#########################end code for Task 3-a

#Task 3-b: Draw a vertical bar chart for total number of surveys taken
for each gender for each month. Display months with their
corresponding names.
# Remember to make the bar chart into a vertical bar chart
#########################begin code for Task 3-b
VBC = dfClean.groupby(['Month','gender']).size()
plt.title('Vertical Bar: Number of Survey')
plt.ylabel('Survey')
VBC.plot(x = "Month", kind="bar")
#########################end code for Task 3-b
```

```
C:\Python310\lib\site-packages\matplotlib\axes\_axes.py:6565:
RuntimeWarning: All-NaN slice encountered
  xmin = min(xmin, np.nanmin(xi))
C:\Python310\lib\site-packages\matplotlib\axes\_axes.py:6566:
```

```
RuntimeWarning: All-NaN slice encountered
  xmax = max(xmax, np.nanmax(xi))
```

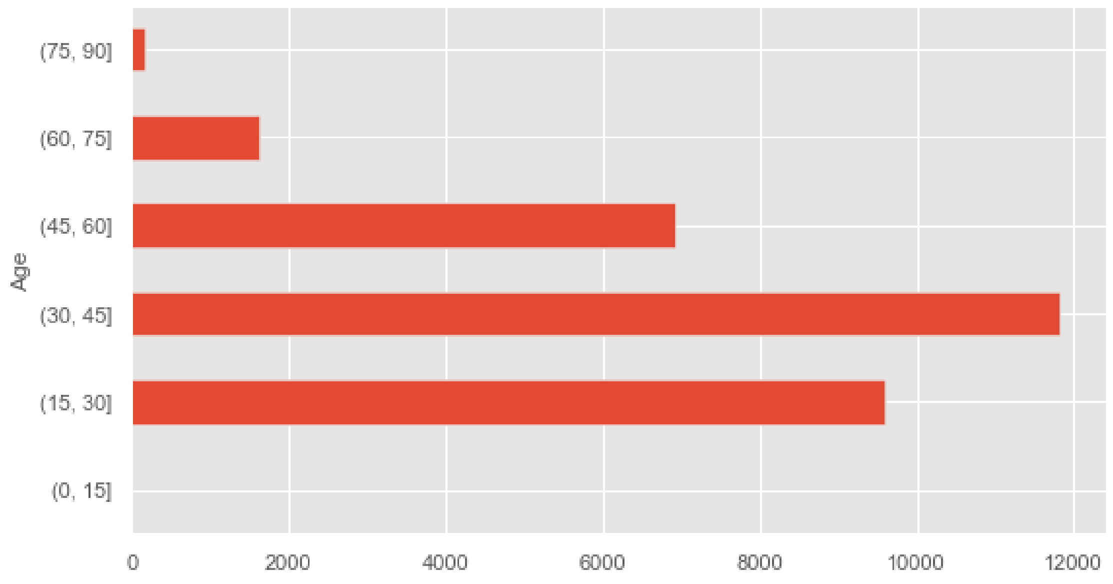


```
<AxesSubplot:title={'center':'Vertical Bar: Number of Survey'},
xlabel='Month,gender', ylabel='Survey'>
```



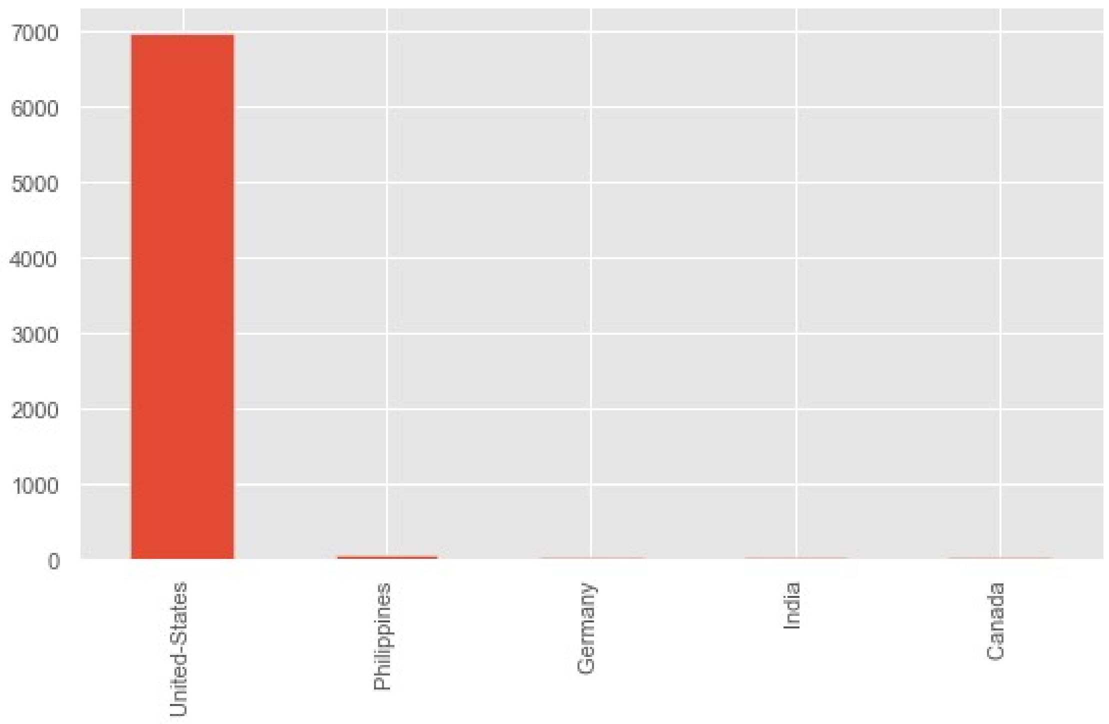
```python
#Task 3-c: Draw a horizontal bar chart for number of surveys taken
with respect to age feature keeping the age interval as 15.
# Remember to make the bar chart into a horizontal bar chart
#########################begin code for Task 3-c
dfCleans = dfClean.groupby(pd.cut(dfClean['Age'],
[0,15,30,45,60,75,90])).size()
dfCleans.plot.barh()
#########################end code for Task 3-c
```

```
<AxesSubplot:ylabel='Age'>
```



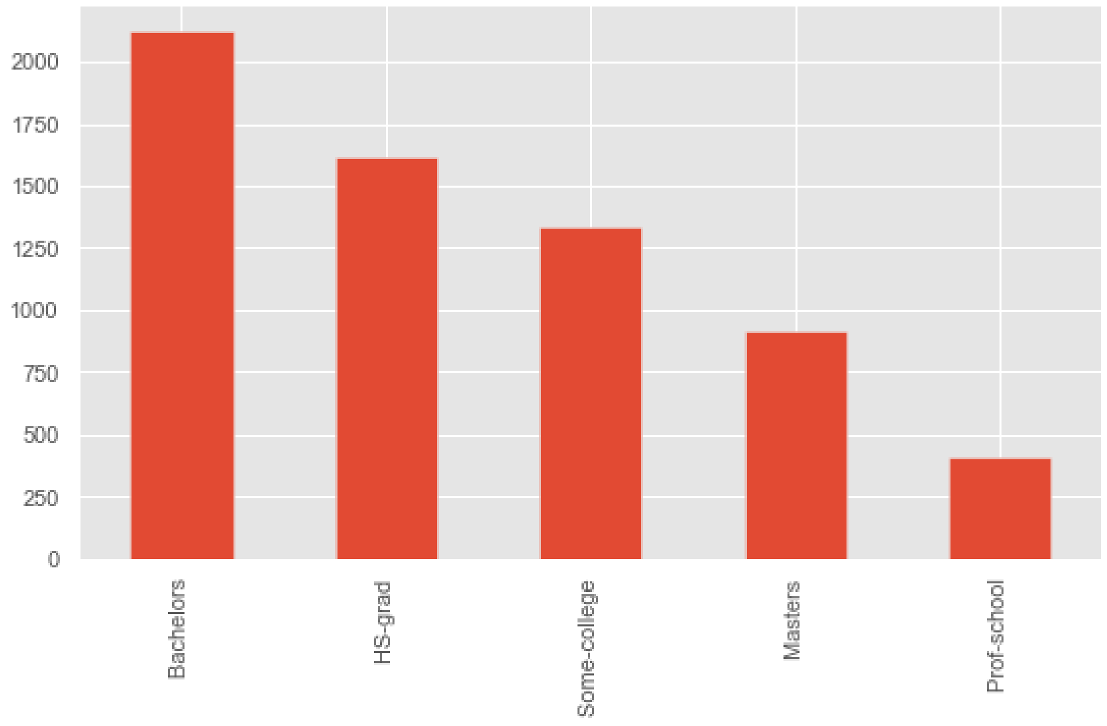
```
#Task 3-d: Draw a "vertical" bar chart that lists the top-5 native-
countries based on the number of samples with class >50K.
# Remember to make the bar chart into a vertical bar chart
#########################begin code for Task 3-d
dfClean.loc[(dfClean['class'] == ' >50K')]['native-
country'].value_counts().head(5).plot(x = 'month', kind="bar")
#########################end code for Task 3-d
```

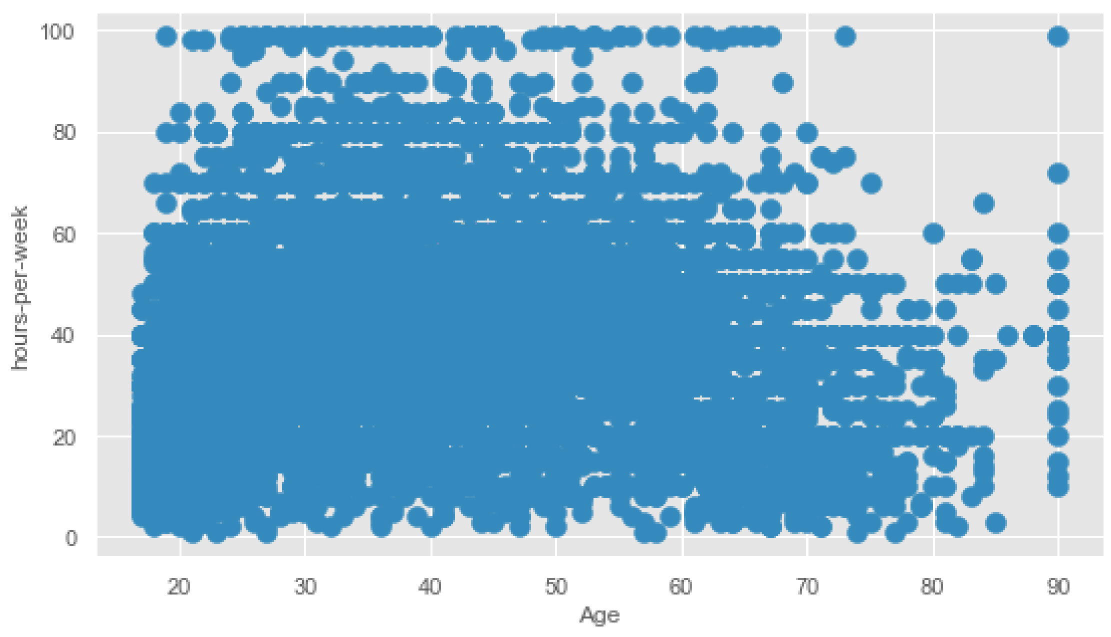
```
<AxesSubplot:>
```



```
#Task 3-e: Now repeat Task 3-d based on education (again top-5)
#########################begin code for Task 3-e
dfClean.loc[(dfClean['class'] == ' >50K')]
['education'].value_counts().head(5).plot(x = 'Month', kind="bar")
#########################end code for Task 3-e

#Task 3-f: Draw a scatter plot for age vs hours per week.
#########################begin code for Task 3-f
dfClean.plot.scatter(x = 'Age', y = 'hours-per-week', s = 100)
#########################end code for Task 3-f
```

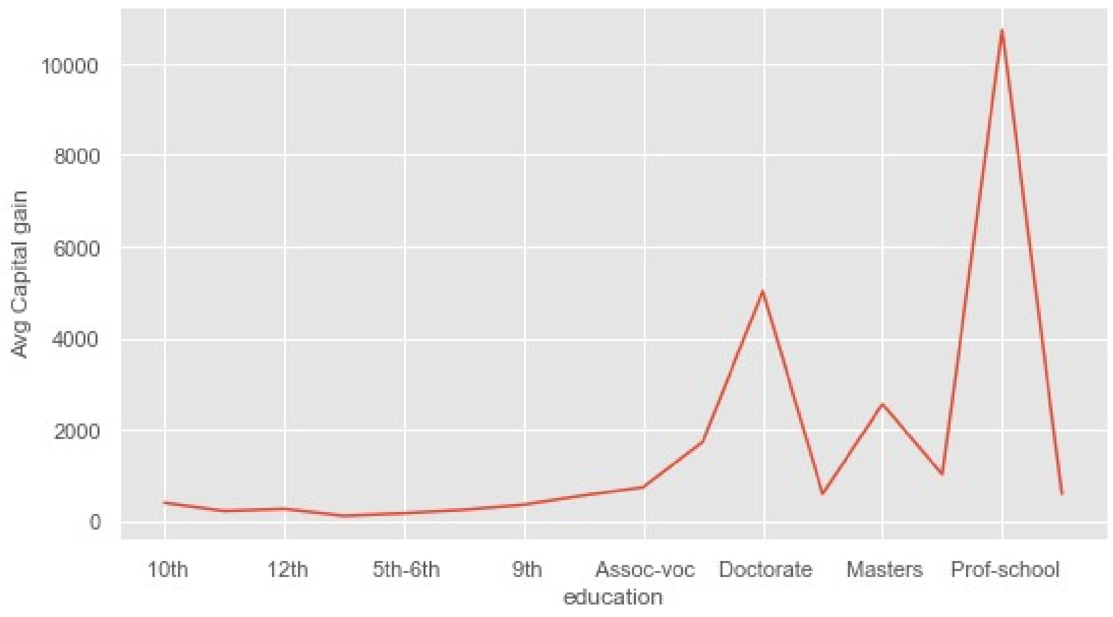
```
<AxesSubplot:xlabel='Age', ylabel='hours-per-week'>
```





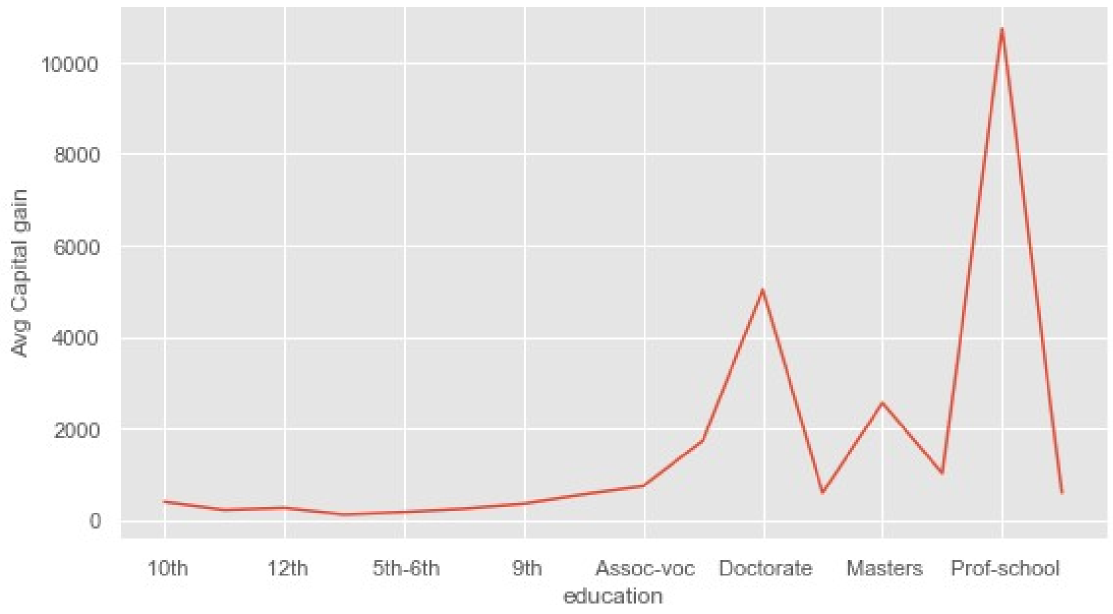
```
#Task 3-g: Draw a line chart showing average capital gain for each
education category.
# X-axis : education category, Y-axis : the avg capital gain
#########################begin code for Task 3-g
plt.xlabel('Education Category')
plt.ylabel('Avg Capital gain')
dfClean.groupby('education')["capital-gain"].mean().plot()
#########################end code for Task 3-g
```

```
<AxesSubplot:xlabel='education', ylabel='Avg Capital gain'>
```



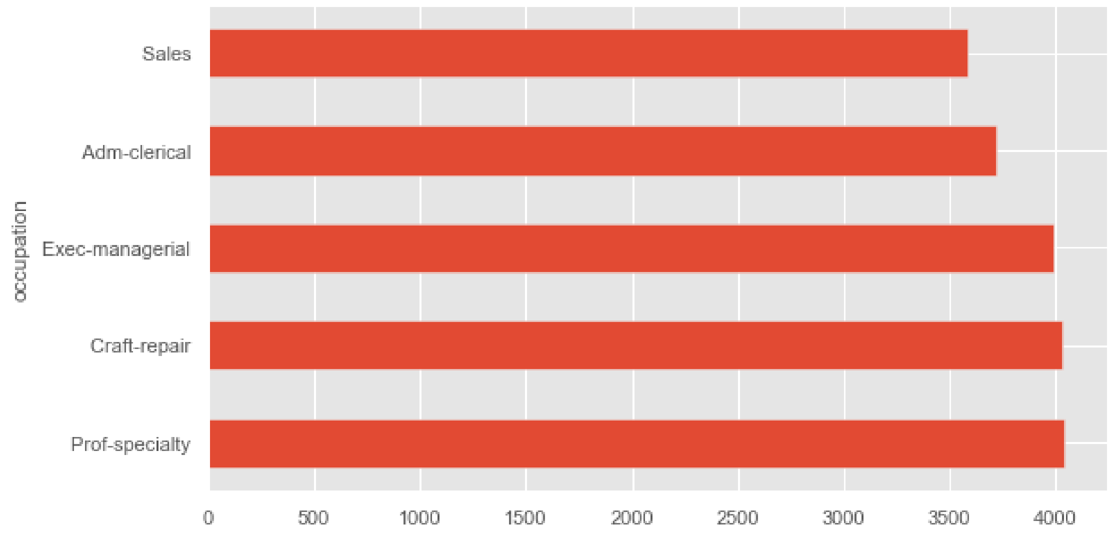
```
#Task 3-g: Draw a line chart showing average capital gain for each
education category.
# X-axis : education category, Y-axis : the avg capital gain
#########################begin code for Task 3-g
plt.xlabel('Education Category')
plt.ylabel('Avg Capital gain')
dfClean.groupby('education')["capital-gain"].mean().plot()
#########################end code for Task 3-g



#########################end code for Task 3-i
```

```
<AxesSubplot:xlabel='education', ylabel='Avg Capital gain'>
```



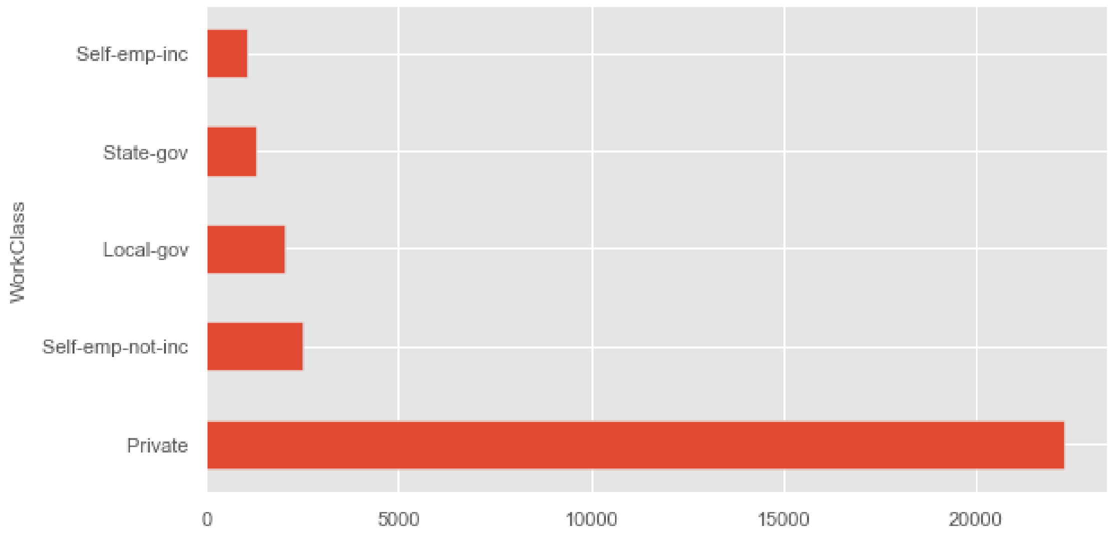
```
#Task 3-h: Draw a 'horizontal' bar chart for the top-5 most common
occupation.
#########################begin code for Task 3-h
dfClean.groupby('occupation').size().sort_values(ascending=False).head
(5).plot.barh()
#########################end code for Task 3-h
```

```
<AxesSubplot:ylabel='occupation'>
```



```
#Task 3-i: Draw a 'horizontal' bar chart for the top-5 most common
workclass.
#########################begin code for Task 3-i
dfClean.groupby('WorkClass').size().sort_values(ascending=False).head(
5).plot.barh()
```

```
<AxesSubplot:ylabel='WorkClass'>
```



Task 4:

Find out an interesting information from your census1994 dataset. Create a visualization for it. This task is worth 20 points. Your result will be judged based on the uniqueness and quality of your work (having a meaningful result and an aesthetic visulization).

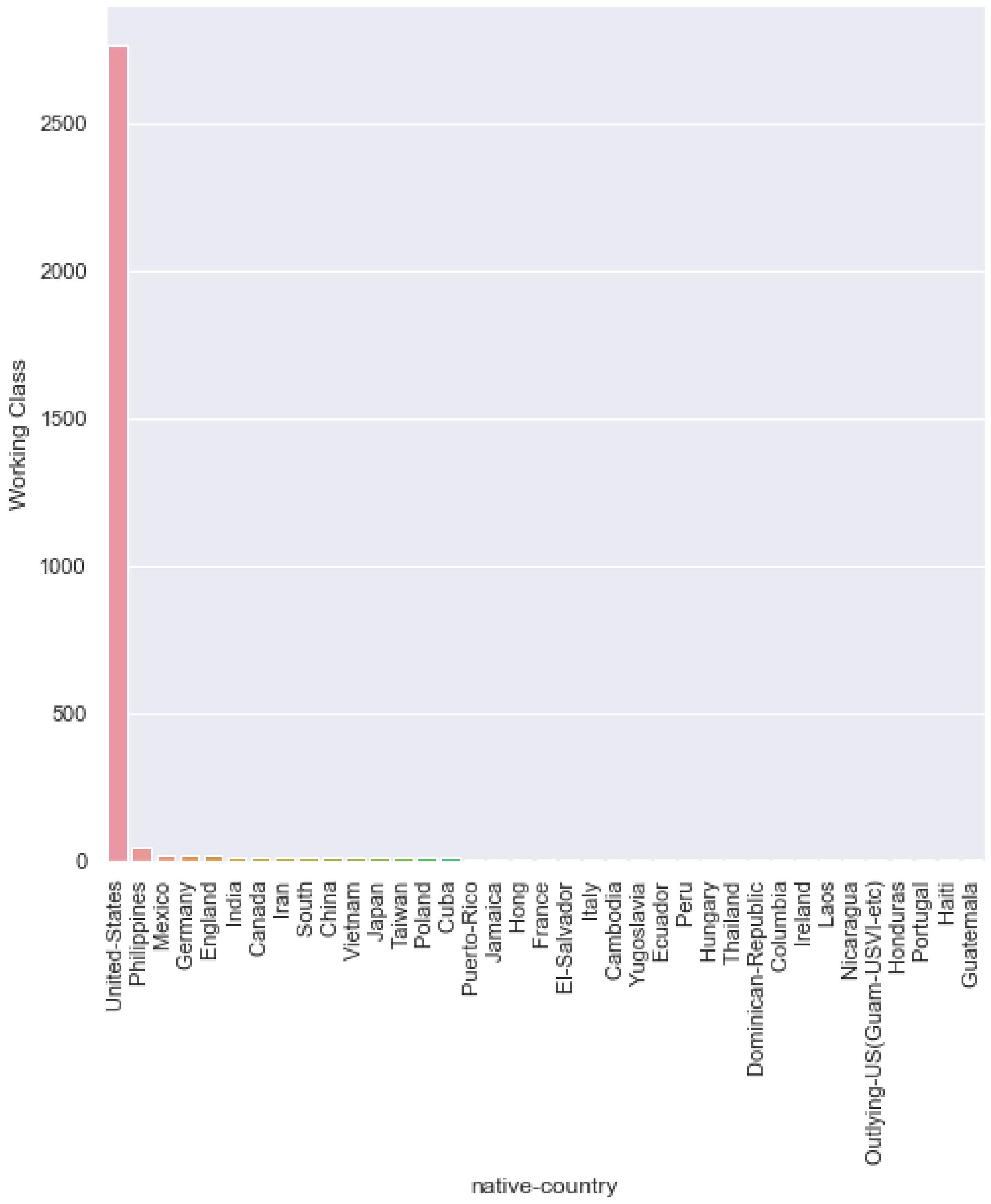
```
# The most number of bachelors are from United-States that from 20 t0
40 working class that will increase the productivity of country
```

```
#########################begin code for Task 4
import seaborn as sns
data_count = dfClean.loc[(dfClean['Age'] <= 40) & (dfClean['Age'] >
20) & (dfClean['education'] == ' Bachelors')].groupby('native-
country').size().sort_values(ascending= False).to_frame(name =
'Working Class').reset_index()
data_count
##########################end code for Task 4
```

```
                 native-country  Working Class
0                 United-States           2762
1                   Philippines             41
2                        Mexico             19
3                       Germany             16
4                       England             15
5                         India             13
6                        Canada             12
7                          Iran             12
8                         South             11
9                         China             10
10                      Vietnam             10
11                        Japan              9
12                       Taiwan              9
13                       Poland              8
14                         Cuba              7
15                  Puerto-Rico              6
16                      Jamaica              5
17                         Hong              4
18                       France              4
19                  El-Salvador              4
20                        Italy              4
21                     Cambodia              3
22                   Yugoslavia              3
23                      Ecuador              3
24                         Peru              2
25                      Hungary              2
26                     Thailand              2
27           Dominican-Republic              2
28                     Columbia              2
29                      Ireland              2
30                         Laos              1
31                    Nicaragua              1
32   Outlying-US(Guam-USVI-etc)              1
33                     Honduras              1
34                     Portugal              1
35                        Haiti              1
36                    Guatemala              1
```

```
sns.set(style="darkgrid")
plt.figure(figsize=(8, 8))
plt.xticks(rotation='vertical')
```

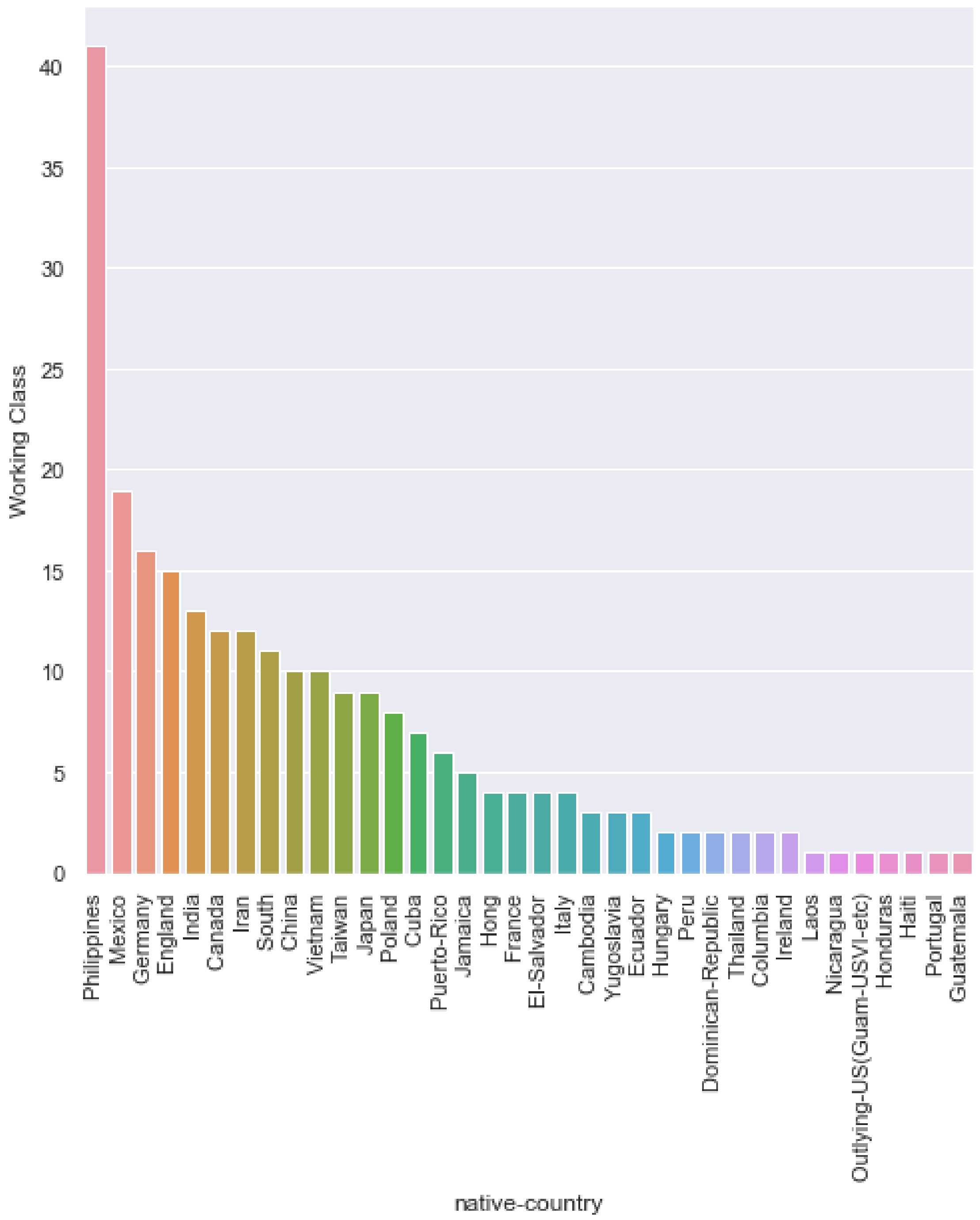
```
sns.barplot(x="native-country", y="Working Class", data=data_count,
ci=None);
```



```
data_countW = dfClean.loc[(dfClean['Age'] <= 40) & (dfClean['Age'] >
20) & (dfClean['native-country'] != ' United-States')
&(dfClean['education'] == ' Bachelors')].groupby('native-
country').size().sort_values(ascending= False).to_frame(name =
'Working Class').reset_index()
data_countW
```

```
                  native-country  Working Class
0                    Philippines             41
1                         Mexico             19
2                        Germany             16
3                        England             15
4                          India             13
5                         Canada             12
6                           Iran             12
7                          South             11
8                          China             10
9                        Vietnam             10
10                        Taiwan              9
11                         Japan              9
12                        Poland              8
13                          Cuba              7
14                   Puerto-Rico              6
15                       Jamaica              5
16                          Hong              4
17                        France              4
18                   El-Salvador              4
19                         Italy              4
20                      Cambodia              3
21                    Yugoslavia              3
22                       Ecuador              3
23                       Hungary              2
24                          Peru              2
25            Dominican-Republic              2
26                      Thailand              2
27                      Columbia              2
28                       Ireland              2
29                          Laos              1
30                     Nicaragua              1
31    Outlying-US(Guam-USVI-etc)              1
32                      Honduras              1
33                         Haiti              1
34                      Portugal              1
35                     Guatemala              1
```

```
sns.set(style="darkgrid")
plt.figure(figsize=(8, 8))
plt.xticks(rotation='vertical')
sns.barplot(x="native-country", y="Working Class", data=data_countW,
ci=None);
```



Grading

Report for Python Report Explanation Create a report in a pdf file explaining the answer for task 2, 3 and 4 Note: the more detailed explanation the better

Rubricks Task 1: 8 points Task 2: 40 points Task 3: 20 points Task 4: 20 points

Report : 12 points Total : 100 points