Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ __________Информатика и системы управления____________
КАФЕДРА _______Системы обработки информации и управления_________

Отчёт
по лабораторной работе № 2

«Изучение библиотек обработки данных»

по курсу «Технологии машинного обучения»

Выполнил:
Студент группы ИУ5-63 __________________ **__Волков А.С.__**
(Подпись, дата) (Фамилия И.О.)

Проверил:
________________ **__Гапанюк Ю.Е.__**
(Подпись, дата) (Фамилия И.О.)

Москва – 2020

Цель лабораторной работы:

Изучение библиотеки обработки данных Pandas.

Задание:

Выполните первое демонстрационное задание "demo assignment" под названием "Exploratory data analysis with Pandas" со страницы курса https://mlcourse.ai/assignments

Условие задания
- https://nbviewer.jupyter.org/github/Yorko/mlcourse_open/blob/master/jupyter_english/assignments_demo/assignment01_pandas_uci_adult.ipynb?flush_cache=true

Выполнение лабораторной работы:

Assignment #1 (demo)

Exploratory data analysis with Pandas

Same assignment as a [Kaggle Kernel (https://www.kaggle.com/kashnitsky/a1-demo-pandas-and-uci-adult-dataset)](https://www.kaggle.com/kashnitsky/a1-demo-pandas-and-uci-adult-dataset) + [solution (https://www.kaggle.com/kashnitsky/a1-demo-pandas-and-uci-adult-dataset-solution)](https://www.kaggle.com/kashnitsky/a1-demo-pandas-and-uci-adult-dataset-solution).

In this task you should use Pandas to answer a few questions about the [Adult (https://archive.ics.uci.edu/ml/datasets/Adult)](https://archive.ics.uci.edu/ml/datasets/Adult) dataset. (You don't have to download the data – it's already in the repository). Choose the answers in the [web-form (https://docs.google.com/forms/d/1uY7MpI2trKx6FLWZte0uVh3ULV4Cm_tDud0VDFGCOKg)](https://docs.google.com/forms/d/1uY7MpI2trKx6FLWZte0uVh3ULV4Cm_tDud0VDFGCOKg).

Unique values of all features (for more information, please see the links above):

- `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.
- `sex` : 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.
- `salary` : >50K,<=50K

In [2]:

```
import numpy as np
import pandas as pd
pd.set_option('display.max.columns', 100)
# to draw pictures in jupyter notebook
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
# we don't like warnings
# you can comment the following 2 lines if you'd like to
import warnings
warnings.filterwarnings('ignore')
```

In [3]:

```
data = pd.read_csv('../data/adult.data.csv')
data.head()
```

Out[3]:

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

1. How many men and women (*sex* feature) are represented in this dataset?

In [4]:

```
data['sex'].value_counts()
```

Out[4]:

```
Male      21790
Female    10771
Name: sex, dtype: int64
```

2. What is the average age (*age* feature) of women?

In [5]:

```
print(round(float(data.loc[data['sex']=='Female', ['age']].mean())))
```

```
37
```

3. What is the percentage of German citizens (*native-country* feature)?

In [6]:

```
print(round(float(data.loc[data['native-country']=='Germany', ['native-country']].count
()/data['native-country'].count()*100),2),'%')
```

```
0.42 %
```

4-5. What are the mean and standard deviation of age for those who earn more than 50K per year (*salary* feature) and those who earn less than 50K per year?

In [7]:

```
print('Standard deviation for those who earn <=50K :',float(data.loc[data['salary']=='<
=50K', ['age']].std())) #std <=50
print('Mean deviation for those who earn <=50K :',float(data.loc[data['salary']=='<=50
K', ['age']].mad())) #mad <=50
print('Standard deviation for those who earn >50K :',float(data.loc[data['salary']=='>5
0K', ['age']].std())) #std >50
print('Mean deviation for those who earn >50K :',float(data.loc[data['salary']=='>50K',
['age']].mad())) #mad >50
```

```
Standard deviation for those who earn <=50K : 14.020088490824813
Mean deviation for those who earn <=50K : 11.467855024821914
Standard deviation for those who earn >50K : 10.519027719851785
Mean deviation for those who earn >50K : 8.47674579194268
```

6. Is it true that people who earn more than 50K have at least high school education? (*education – Bachelors, Prof-school, Assoc-acdm, Assoc-voc, Masters* or *Doctorate* feature)

In [8]:

```
highEduList = ['Bachelors','Prof-school','Assoc-acdm','Assoc-voc','Masters','Doctorate'
]
dataList = list(data.loc[data['salary'] == '>50K', 'education'].unique())
fl = True;
for a in dataList:
    for b in highEduList:
        if a==b:
            fl = True;
            break;
        if a!=b:
            fl = False;
print(fl);
```

```
False
```

7. Display age statistics for each race (*race* feature) and each gender (*sex* feature). Use *groupby()* and *describe()*. Find the maximum age of men of *Amer-Indian-Eskimo* race.

Solution 1 without using groupby()

In [9]:

```
raceList = list(data['race'].unique())
genderList = list(data['sex'].unique())
for race in raceList:
    print(race,':',sep='')
    print(data.loc[data['race'] == race, ['age']].describe())
    print('--------------------')
for gender in genderList:
    print(gender,':',sep='')
    print(data.loc[data['sex'] == gender, ['age']].describe())
    print('--------------------')
```

```
White:
                age
count  27816.000000
mean      38.769881
std       13.782306
min       17.000000
25%       28.000000
50%       37.000000
75%       48.000000
max       90.000000
-------------------
Black:
               age
count  3124.000000
mean     37.767926
std      12.759290
min      17.000000
25%      28.000000
50%      36.000000
75%      46.000000
max      90.000000
-------------------
Asian-Pac-Islander:
               age
count  1039.000000
mean     37.746872
std      12.825133
min      17.000000
25%      28.000000
50%      36.000000
75%      45.000000
max      90.000000
-------------------
Amer-Indian-Eskimo:
              age
count  311.000000
mean    37.173633
std     12.447130
min     17.000000
25%     28.000000
50%     35.000000
75%     45.500000
max     82.000000
-------------------
Other:
              age
count  271.000000
mean    33.457565
std     11.538865
min     17.000000
25%     25.000000
50%     31.000000
75%     41.000000
max     77.000000
-------------------
Male:
                age
count  21790.000000
mean      39.433547
std       13.370630
min       17.000000
```

```
25%         29.000000
50%         38.000000
75%         48.000000
max         90.000000
-------------------
Female:
                  age
count   10771.000000
mean       36.858230
std        14.013697
min        17.000000
25%        25.000000
50%        35.000000
75%        46.000000
max        90.000000
-------------------
```

Solution2 using groupby()

In [10]:

```
data.groupby(['race', 'sex'])['age'].describe()
```

Out[10]:

race	sex	count	mean	std	min	25%	50%	75%	max
Amer-Indian-Eskimo	Female	119.0	37.117647	13.114991	17.0	27.0	36.0	46.00	80.0
	Male	192.0	37.208333	12.049563	17.0	28.0	35.0	45.00	82.0
Asian-Pac-Islander	Female	346.0	35.089595	12.300845	17.0	25.0	33.0	43.75	75.0
	Male	693.0	39.073593	12.883944	18.0	29.0	37.0	46.00	90.0
Black	Female	1555.0	37.854019	12.637197	17.0	28.0	37.0	46.00	90.0
	Male	1569.0	37.682600	12.882612	17.0	27.0	36.0	46.00	90.0
Other	Female	109.0	31.678899	11.631599	17.0	23.0	29.0	39.00	74.0
	Male	162.0	34.654321	11.355531	17.0	26.0	32.0	42.00	77.0
White	Female	8642.0	36.811618	14.329093	17.0	25.0	35.0	46.00	90.0
	Male	19174.0	39.652498	13.436029	17.0	29.0	38.0	49.00	90.0

The maximum age of men of Amer-Indian-Eskimo race:

In [11]:

```
raceAIEdata = data.loc[data['race'] == 'Amer-Indian-Eskimo']
print(int(raceAIEdata.loc[data['sex']=='Male', ['age']].max()))
```

```
82
```

8. Among whom is the proportion of those who earn a lot (>50K) greater: married or single men (*marital-status* feature)? Consider as married those who have a *marital-status* starting with *Married* (Married-civ-spouse, Married-spouse-absent or Married-AF-spouse), the rest are considered bachelors.

In [31]:

```
maleData = data.loc[data['sex'] == 'Male']
maleCount = maleData.count()

marriedMaleData = maleData.loc[maleData['marital-status'].str.contains('Married'), ['salary']]
marriedMaleCount = int(marriedMaleData.count())

marriedMale50KData = marriedMaleData.loc[marriedMaleData['salary'] == '>50K']
marriedMale50Kcount = int(marriedMale50KData.count())

singleMaleData = maleData.loc[~maleData['marital-status'].str.contains('Married'), ['salary']]
singleMaleCount = int(singleMaleData.count())

singleMale50KData = singleMaleData.loc[singleMaleData['salary'] == '>50K']
singleMale50Kcount = int(singleMale50KData.count())

singleMale50Kprop = singleMale50Kcount/singleMaleCount
marriedMale50Kprop = marriedMale50Kcount/marriedMaleCount
if singleMale50Kprop > marriedMale50Kprop:
    print('Among single men')
else:
    print('Among married men')
print('Proportion among single men:',singleMale50Kprop)
print('Proportion among married men:',marriedMale50Kprop)
```

```
Among married men
Proportion among single men: 0.08449509031397745
Proportion among married men: 0.4405139945351156
```

9. What is the maximum number of hours a person works per week (*hours-per-week* feature)? How many people work such a number of hours, and what is the percentage of those who earn a lot (>50K) among them?

In [47]:

```
maxHours = int(data['hours-per-week'].max())
print('The maximum number of hours a person works per week:', maxHours, '\n')

maxHoursSalaryData = data.loc[data['hours-per-week'] == maxHours, ['salary']]
print(int(maxHoursSalaryData.count()), 'people work such a number of hours\n')

maxHours50KData = maxHoursSalaryData.loc[maxHoursSalaryData['salary'] == '>50K']
print('The percentage of those who earn a lot (>50K) among them:',round(int(maxHours50KData.count())/int(maxHoursSalaryData.count())*100),'%')
```

```
The maximum number of hours a person works per week: 99

85 people work such a number of hours

The percentage of those who earn a lot (>50K) among them: 29 %
```

10. Count the average time of work (*hours-per-week*) for those who earn a little and a lot (*salary*) for each country (*native-country*). What will these be for Japan?

In [57]:

```
pd.set_option('display.max_rows', None)
print(data.groupby(['native-country', 'salary'])['hours-per-week'].mean(), '\n')

JapData = data.loc[data['native-country'] == 'Japan']

print('Japan:')
print('<=50K:', float(JapData.loc[JapData['salary'] == '<=50K', ['hours-per-week']].mea
n()))
print('>50K:', float(JapData.loc[JapData['salary'] == '>50K', ['hours-per-week']].mean
()))
```

```
native-country              salary
?                           <=50K     40.164760
                            >50K      45.547945
Cambodia                    <=50K     41.416667
                            >50K      40.000000
Canada                      <=50K     37.914634
                            >50K      45.641026
China                       <=50K     37.381818
                            >50K      38.900000
Columbia                    <=50K     38.684211
                            >50K      50.000000
Cuba                        <=50K     37.985714
                            >50K      42.440000
Dominican-Republic          <=50K     42.338235
                            >50K      47.000000
Ecuador                     <=50K     38.041667
                            >50K      48.750000
El-Salvador                 <=50K     36.030928
                            >50K      45.000000
England                     <=50K     40.483333
                            >50K      44.533333
France                      <=50K     41.058824
                            >50K      50.750000
Germany                     <=50K     39.139785
                            >50K      44.977273
Greece                      <=50K     41.809524
                            >50K      50.625000
Guatemala                   <=50K     39.360656
                            >50K      36.666667
Haiti                       <=50K     36.325000
                            >50K      42.750000
Holand-Netherlands          <=50K     40.000000
Honduras                    <=50K     34.333333
                            >50K      60.000000
Hong                        <=50K     39.142857
                            >50K      45.000000
Hungary                     <=50K     31.300000
                            >50K      50.000000
India                       <=50K     38.233333
                            >50K      46.475000
Iran                        <=50K     41.440000
                            >50K      47.500000
Ireland                     <=50K     40.947368
                            >50K      48.000000
Italy                       <=50K     39.625000
                            >50K      45.400000
Jamaica                     <=50K     38.239437
                            >50K      41.100000
Japan                       <=50K     41.000000
                            >50K      47.958333
Laos                        <=50K     40.375000
                            >50K      40.000000
Mexico                      <=50K     40.003279
                            >50K      46.575758
Nicaragua                   <=50K     36.093750
                            >50K      37.500000
Outlying-US(Guam-USVI-etc)  <=50K     41.857143
Peru                        <=50K     35.068966
                            >50K      40.000000
Philippines                 <=50K     38.065693
                            >50K      43.032787
```

```
Poland                        <=50K     38.166667
                              >50K      39.000000
Portugal                      <=50K     41.939394
                              >50K      41.500000
Puerto-Rico                   <=50K     38.470588
                              >50K      39.416667
Scotland                      <=50K     39.444444
                              >50K      46.666667
South                         <=50K     40.156250
                              >50K      51.437500
Taiwan                        <=50K     33.774194
                              >50K      46.800000
Thailand                      <=50K     42.866667
                              >50K      58.333333
Trinadad&Tobago               <=50K     37.058824
                              >50K      40.000000
United-States                 <=50K     38.799127
                              >50K      45.505369
Vietnam                       <=50K     37.193548
                              >50K      39.200000
Yugoslavia                    <=50K     41.600000
                              >50K      49.500000
Name: hours-per-week, dtype: float64

Japan:
<=50K: 41.0
>50K: 47.958333333333336
```

Other solution for Japan:

In [58]:

```
data.loc[data['native-country'] == 'Japan'].groupby(['native-country', 'salary'])['hour
s-per-week'].mean()
```

Out[58]:

```
native-country  salary
Japan           <=50K     41.000000
                >50K      47.958333
Name: hours-per-week, dtype: float64
```