



**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 ИУ5 «Системы обработки информации и управления»**

Лабораторная работа №2  
по дисциплине «Технология машинного обучения» на тему:

Изучение библиотек обработки данных.

---

Выполнил:  
студент группы № ИУ5-62  
Морозенков О.Н.  
подпись, дата

Проверил:  
Ю.Е. Гапанюк  
подпись, дата

2020 г.

## Задание:

### Часть 1.

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](https://nbviewer.jupyter.org/github/Yorko/mlcourse_open/blob/master/jupyter_english/assignments_demo/assignment01_pandas_uci_adult.ipynb?flush_cache=true)

Набор данных можно скачать здесь - <https://archive.ics.uci.edu/ml/datasets/Adult>

Пример решения задания - <https://www.kaggle.com/kashnitsky/a1-demo-pandas-and-uci-adult-dataset-solution>

### Часть 2.

Выполните следующие запросы с использованием двух различных библиотек - **Pandas** и **PandaSQL**:

- один произвольный запрос на соединение двух наборов данных
- один произвольный запрос на группировку набора данных с использованием функций агрегирования

Сравните время выполнения каждого запроса в Pandas и PandaSQL.

In [3]:

```
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 [4]:

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

Out[4]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

In [6]:

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

Out[6]:

Male 21790
Female 10771
Name: sex, dtype: int64

In [19]:

```
#df.loc['2012-Feb', 'Close'].mean()
#data.loc['age'].mean()
data.loc[data['sex'] == 'Female', 'age'].mean()
#data.loc[data['sex'] == 'Female'].mean()
```

Out[19]:

36.85823043357163

In [25]:

```
data.loc[data['native-country'] == 'Germany'].count() / data.shape[0] * 100
```

Out[25]:

age 0.420749
workclass 0.420749
fnlwgt 0.420749
education 0.420749
education-num 0.420749
marital-status 0.420749
occupation 0.420749
relationship 0.420749

```
age          0.420749
sex          0.420749
capital-gain 0.420749
capital-loss 0.420749
hours-per-week 0.420749
native-country 0.420749
salary       0.420749
dtype: float64
```

In [27]:

```
data.loc[data['salary'] == '<=50K', 'age'].mean()
```

Out[27]:

```
36.78373786407767
```

In [28]:

```
data.loc[data['salary'] == '>50K', 'age'].mean()
```

Out[28]:

```
44.24984058155847
```

In [29]:

```
ages1 = data.loc[data['salary'] == '>50K', 'age']
ages2 = data.loc[data['salary'] == '<=50K', 'age']
print("The average age of the rich: {0} +- {1} years, poor - {2} +- {3} years.".format(
    round(ages1.mean()), round(ages1.std(), 1),
    round(ages2.mean()), round(ages2.std(), 1)))
```

The average age of the rich: 44 +- 10.5 years, poor - 37 +- 14.0 years.

In [32]:

```
data.loc[data['salary'] == '>50K', 'education'].value_counts()
```

Out[32]:

```
Bachelors    2221
HS-grad      1675
Some-college 1387
Masters       959
Prof-school   423
Assoc-voc     361
Doctorate     306
Assoc-acdm    265
10th          62
11th          60
7th-8th       40
12th          33
9th           27
5th-6th       16
1st-4th        6
Name: education, dtype: int64
```

In [34]:

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

```
sex  race
Female Amer-Indian-Eskimo  37.117647
      Asian-Pac-Islander   35.089595
      Black                37.854019
      Other                31.678899
      White                36.811618
Male   Amer-Indian-Eskimo  37.208333
      Asian-Pac-Islander   39.073593
      Black                37.682600
      Other                34.654321
      White                39.652498
Name: age, dtype: float64
```

In [37]:

```
for (race, sex), sub_df in data.groupby(['race', 'sex']):  
    print("Race: {0}, sex: {1}".format(race, sex))  
    print(sub_df['age'].describe())
```

Race: Amer-Indian-Eskimo, sex: Female

count 119.000000  
mean 37.117647  
std 13.114991  
min 17.000000  
25% 27.000000  
50% 36.000000  
75% 46.000000  
max 80.000000

Name: age, dtype: float64

Race: Amer-Indian-Eskimo, sex: Male

count 192.000000  
mean 37.208333  
std 12.049563  
min 17.000000  
25% 28.000000  
50% 35.000000  
75% 45.000000  
max 82.000000

Name: age, dtype: float64

Race: Asian-Pac-Islander, sex: Female

count 346.000000  
mean 35.089595  
std 12.300845  
min 17.000000  
25% 25.000000  
50% 33.000000  
75% 43.750000  
max 75.000000

Name: age, dtype: float64

Race: Asian-Pac-Islander, sex: Male

count 693.000000  
mean 39.073593  
std 12.883944  
min 18.000000  
25% 29.000000  
50% 37.000000  
75% 46.000000  
max 90.000000

Name: age, dtype: float64

Race: Black, sex: Female

count 1555.000000  
mean 37.854019  
std 12.637197  
min 17.000000  
25% 28.000000  
50% 37.000000  
75% 46.000000  
max 90.000000

Name: age, dtype: float64

Race: Black, sex: Male

count 1569.000000  
mean 37.682600  
std 12.882612  
min 17.000000  
25% 27.000000  
50% 36.000000  
75% 46.000000  
max 90.000000

Name: age, dtype: float64

Race: Other, sex: Female

count 109.000000  
mean 31.678899  
std 11.631599  
min 17.000000  
25% 23.000000  
50% 29.000000  
75% 39.000000  
max 74.000000

Name: age, dtype: float64

Race: Other, sex: Male

count 162.000000  
mean 34.654321  
std 11.355531  
min 17.000000  
25% 26.000000

```
25%    26.000000
50%    32.000000
75%    42.000000
max     77.000000
Name: age, dtype: float64
Race: White, sex: Female
count   8642.000000
mean    36.811618
std     14.329093
min     17.000000
25%     25.000000
50%     35.000000
75%     46.000000
max     90.000000
Name: age, dtype: float64
Race: White, sex: Male
count  19174.000000
mean    39.652498
std     13.436029
min     17.000000
25%     29.000000
50%     38.000000
75%     49.000000
max     90.000000
Name: age, dtype: float64
```

In [38]:

```
data.loc[data['salary'] == '>50K', 'marital-status'].value_counts()
```

Out[38]:

```
Married-civ-spouse    6692
Never-married         491
Divorced              463
Widowed              85
Separated             66
Married-spouse-absent  34
Married-AF-spouse     10
Name: marital-status, dtype: int64
```

In [39]:

```
data.loc[(data['sex'] == 'Male') &
         (data['marital-status'].isin(['Never-married',
                                       'Separated',
                                       'Divorced',
                                       'Widowed']))], 'salary'].value_counts()
```

Out[39]:

```
<=50K    7552
>50K      697
Name: salary, dtype: int64
```

In [43]:

```
data.loc[(data['sex'] == 'Male') &
         (data['marital-status'].str.startswith('Married'))], 'salary'].value_counts()
```

Out[43]:

```
<=50K    7576
>50K     5965
Name: salary, dtype: int64
```

In [44]:

```
data['hours-per-week'].max()
```

Out[44]:

99

In [47]:

```
data.loc[data['hours-per-week'] == data['hours-per-week'].max()].count()
```

Out[47]:

```
age      85
workclass 85
fnlwgt   85
education 85
education-num 85
marital-status 85
occupation 85
relationship 85
race      85
sex       85
capital-gain 85
capital-loss 85
hours-per-week 85
native-country 85
salary    85
dtype: int64
```

In [62]:

```
data.loc[(data['hours-per-week'] == data['hours-per-week'].max()) &
         (data['salary'] == '>50K')].count() / data.loc[data['salary']=='>50K'].count() *100
```

Out[62]:

```
age      0.318837
workclass 0.318837
fnlwgt   0.318837
education 0.318837
education-num 0.318837
marital-status 0.318837
occupation 0.318837
relationship 0.318837
race      0.318837
sex       0.318837
capital-gain 0.318837
capital-loss 0.318837
hours-per-week 0.318837
native-country 0.318837
salary    0.318837
dtype: float64
```

In [64]:

```
print(data.groupby(['native-country', 'salary'])['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

```
...
Mexico          >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
Trinidad&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, Length: 82, dtype: float64
```

In []:



In [8]:

```
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 [9]:

```
import pandas as ps
from pandasql import sqldf
pysqldf = lambda q: sqldf(q, globals())
```

In [10]:

```
data_user_device = pd.read_csv('user_device.csv')
data_user_device.head()
```

Out[10]:

	use_id	user_id	platform	platform_version	device	use_type_id
0	22782	26980	ios	10.2	iPhone7,2	2
1	22783	29628	android	6.0	Nexus 5	3
2	22784	28473	android	5.1	SM-G903F	1
3	22785	15200	ios	10.2	iPhone7,2	3
4	22786	28239	android	6.0	ONE E1003	1

In [11]:

```
data_user_usage = pd.read_csv('user_usage.csv')
data_user_usage.head()
```

Out[11]:

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id
0	21.97	4.82	1557.33	22787
1	1710.08	136.88	7267.55	22788
2	1710.08	136.88	7267.55	22789
3	94.46	35.17	519.12	22790
4	71.59	79.26	1557.33	22792

In [12]:

```
data_android_devices = pd.read_csv('android_devices.csv')
data_android_devices.head()
```

Out[12]:

	Retail Branding	Marketing Name	Device	Model
0	NaN	NaN	AD681H	Smartfren Andromax AD681H
1	NaN	NaN	FJL21	FJL21
2	NaN	NaN	T31	Panasonic T31
3	NaN	NaN	hws7721g	MediaPad 7 Youth 2

In [16]:

```
pysqldf("""select outgoing_mins_per_month, monthly_mb from data_user_usage order by monthly_mb limit 10""")
```

Out[16]:

	outgoing_mins_per_month	monthly_mb
0	227.13	0.00
1	124.70	11.68
2	29.54	33.79
3	12.85	74.40
4	70.34	212.64
5	341.85	265.81
6	463.05	362.02
7	190.08	369.84
8	85.97	407.01
9	436.37	415.10

In [19]:

```
%%timeit  
pysqldf(""" select * from data_user_usage as u join data_user_device as d on u.use_id = d.use_id """).head()
```

72.6 ms ± 975 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)

In []: