**Московский государственный технический университет им. Н.Э. Баумана**
**Кафедра «Системы обработки информации и управления»**

Лабораторная работа №2
по дисциплине
# «Методы машинного обучения»
на тему
# «Изучение библиотек обработки данных»

Выполнил:
студент группы ИУ5-21М
Андреев К.А.

______________________________

Москва — 2020 г.

# Цель лабораторной работы

Изучение библиотек обработки данных Pandas и PandaSQL.

# Задание

Часть 1: Выполните первое демонстрационное задание "demo assignment" под названием "Exploratory data analysis with Pandas" со страницы курса https://mlcourse.ai/assignments (https://mlcourse.ai/assignments)

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

# Ход выполнения работы

In [0]:

```
!pip install -U pandasql
import pandas as pd
import pandasql as pdsql
import numpy as np
from google.colab import files
import os
import time
```

In [0]:

```
uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))
```

Choose Files No file selected

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving adult.data.csv to adult.data.csv
Saving user_device.csv to user_device.csv
Saving user_usage.csv to user_usage.csv
User uploaded file "adult.data.csv" with length 3518607 bytes
User uploaded file "user_device.csv" with length 9437 bytes
User uploaded file "user_usage.csv" with length 6432 bytes
```

In [0]:

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

Out[0]:

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

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

In [0]:

```
data[data.sex == 'Female'].sex.count()
```

Out[0]:

```
10771
```

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

In [0]:

```
data[data.sex == 'Female'].age.mean()
```

Out[0]:

```
36.85823043357163
```

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

In [0]:

```
data[data['native-country'] == 'Germany'].age.count()/data['native-country'].count()
```

Out[0]:

```
0.004207487485028101
```

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 [0]:

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

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

1. 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 [0]:

```
ar1 = data.loc[data['salary'] == '>50K', 'education'].unique()
ar2 = ['Bachelors', 'Prof-school', 'Assoc-acdm', 'Assoc-voc', 'Masters', 'Doctorate']
if((set(ar1) | set(ar2)) == set(ar2)):
  print('Yes')
else:
  print('No')
```

No

1. 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.

In [0]:

```
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
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
```

1. 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 [0]:

```
print("Not-married: %.2f%%" % (data.loc[(data['sex'] == 'Male') & (~data['marital-status'].str.startswith('Married')) & (data['salary'] == '>50K'), 'salary'].count()/
      data.loc[(data['sex'] == 'Male') & (~data['marital-status'].str.startswith('Married')), 'salary'].count()*100))
print("Married: %.2f%%" % (data.loc[(data['sex'] == 'Male') & (data['marital-status'].str.startswith('Married')) & (data['salary'] == '>50K'), 'salary'].count()/
      data.loc[(data['sex'] == 'Male') & (~data['marital-status'].str.startswith('Married')), 'salary'].count()*100))
```

```
Not-married: 8.45%
Married: 72.31%
```

1. 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 [0]:

```
max_load = data['hours-per-week'].max()
print("Max H/w: %.0f" % max_load)

crazy=data[data['hours-per-week']==max_load].shape[0]
print("Mad people count: %.0f" % crazy)

print("Crazy billioners: %.2f%%" % (data[(data['hours-per-week'] == max_load) & (data['salary']== '>50K')].shape[0]/crazy*100))
print ("so sad(")
```

```
Max H/w: 99
Mad people count: 85
Crazy billioners: 29.41%
so sad(
```

1. 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 [0]:

```
start_time = time.time()
sss = pd.crosstab(data['native-country'], data['salary'], values=data['hours-per-week'], aggfunc=np.mean).T
print("--- %s seconds ---" % (time.time() - start_time))
sss
```

```
--- 0.023040294647216797 seconds ---
```

Out[0]:

| native-country | ? | Cambodia | Canada | China | Columbia | Cuba | Dominican-Republic | Ecuador | El-Salvador | England | France |
|---|---|---|---|---|---|---|---|---|---|---|---|
| salary | | | | | | | | | | | |
| <=50K | 40.164760 | 41.416667 | 37.914634 | 37.381818 | 38.684211 | 37.985714 | 42.338235 | 38.041667 | 36.030928 | 40.483333 | 41.058824 |
| >50K | 45.547945 | 40.000000 | 45.641026 | 38.900000 | 50.000000 | 42.440000 | 47.000000 | 48.750000 | 45.000000 | 44.533333 | 50.750000 |

# PandaSQL

*Время выполнение запроса выведено в последнем пункте 1 задания и в пункте, следующем далее.* Вывод: SQL-запрос по датасету выполнялся на порядок дольше.

Группировка

In [0]:

```
start_time = time.time()
print(pdsql.sqldf('select "native-country", avg("hours-per-week") from data where salary=">50K" group by "native-
country"').head())
print("--- %s seconds ---" % (time.time() - start_time))
```

```
  native-country  avg("hours-per-week")
0              ?              45.547945
1       Cambodia              40.000000
2         Canada              45.641026
3          China              38.900000
4       Columbia              50.000000
--- 0.5527682304382324 seconds ---
```

Соединение

In [0]:

```
print(pdsql.sqldf('select avg(u1.monthly_mb), u2.user_id from user_usage u1 join user_device u2 on u1.use_id = u2
.use_id group by u2.user_id'))
```

```
     avg(u1.monthly_mb)  user_id
0               1557.33     2873
1               3114.67     3191
2               1557.33     6356
3                407.01     6541
4               9005.49    10563
..                  ...      ...
102             6577.12    29717
103             1557.33    29719
104             2076.45    29721
105               74.40    29723
106              519.12    29725

[107 rows x 2 columns]
```