МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра «Систем обработки информации и управления»

# ОТЧЕТ

**Лабораторная работа №__2__**
по дисциплине «Методики машинного обучения»

Тема: «Изучение библиотек обработки данных.»

ИСПОЛНИТЕЛЬ: Егоров С.А.
ФИО
группа ИУ5-22М _____
подпись
"__"_____2020 г.

ПРЕПОДАВАТЕЛЬ: Гапанюк Ю.Е.
ФИО
_____
подпись
"__"_____2020 г.

Москва - 2020

## Цель лабораторной работы

Изучить библиотеки обработки данных Pandas и PandaSQL, выполнив произвольный запрос на соединение двух наборов данных и выполнив один произвольный на группировку набора данных с использованием функций агрегирования. Сравнить выполнения каждого запроса в Pandas и PandaSQL.

## Реализация задания

### *Часть 1.*

In [2]:
```
import numpy as np
import pandas as pd
pd.set_option("display.width", 70)
```

In [2]:
```
data = pd.read_csv('adult.data.csv')
data.head()
```

Out[2]:

| | 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 [3]:
```
#Определим колличесво мужчин и женщин в наборе
data["sex"].value_counts()
```

Out[3]:
```
Male      21790
Female    10771
Name: sex, dtype: int64
```

In [5]:
```
#Определим средний возраст женщин
data.loc[data['sex'] == 'Female', 'age'].mean()
```

Out[5]:
```
36.85823043357163
```

In [8]:
```
#Какова доля немецкий граждан?
print("{0:%}".format(data[data['native-country'] == 'Germany'].shape[0]/data.shape[0]))
```

```
0.420749%
```

In [7]:
```
#Каковы среднее значение и стандартное отклонение возраста тех, кто получает более 50 тысяч в год,
#и тех, кто получает менее 50 тысяч в год?
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: 44.0 +- 10.5 years, poor - 37.0 +- 14.0 years.
```

In [9]:
```
#Какое образование получают люди с ЗП >50K
data.loc[data['salary'] == '>50K', 'education'].unique() # No
```

Out[9]:
```
array(['HS-grad', 'Masters', 'Bachelors', 'Some-college', 'Assoc-voc',
       'Doctorate', 'Prof-school', 'Assoc-acdm', '7th-8th', '12th',
       '10th', '11th', '9th', '5th-6th', '1st-4th'], dtype=object)
```

In [11]:
```
#Какой доход у неженившихся мужчин?
data.loc[(data['sex'] == 'Male') &
     (data['marital-status'].isin(['Never-married',
                                   'Separated',
                                   'Divorced',
                                   'Widowed'])), 'salary'].value_counts()
```

Out[11]:
```
<=50K    7552
>50K      697
Name: salary, dtype: int64
```

In [12]:
```
#Какой доход у женившихся мужчин?
data.loc[(data['sex'] == 'Male') &
     (data['marital-status'].str.startswith('Married')), 'salary'].value_counts()
```

Out[12]:
```
<=50K    7576
>50K     5965
Name: salary, dtype: int64
```

In [13]:
```
data['marital-status'].value_counts()
```

Out[13]:
```
Married-civ-spouse       14976
Never-married            10683
Divorced                  4443
Separated                 1025
Widowed                    993
Married-spouse-absent      418
Married-AF-spouse           23
Name: marital-status, dtype: int64
```

In [18]:
```
#Какое максимальное количество часов человек работает в неделю (функция "часы в неделю")?
max_load = data['hours-per-week'].max()
print("Max time - {0} hours./week.".format(max_load))

#Сколько людей работает такое количество часов?
num_workaholics = data[data['hours-per-week'] == max_load].shape[0]
print("Total number of such hard workers {0}".format(num_workaholics))

#Каков процент тех, кто много зарабатывает среди них?
rich_share = float(data[(data['hours-per-week'] == max_load)
                 & (data['salary'] == '>50K')].shape[0]) / num_workaholics
print("Percentage of rich among them {0}%".format(int(100 * rich_share)))
```

```
Max time - 99 hours./week.
Total number of such hard workers 85
Percentage of rich among them 29%
```

## *Часть 2.*

In [14]:
```
from pandasql import sqldf
pysqldf = lambda q: sqldf(q, globals())
```

In [5]:
```
user_usage = pd.read_csv("user_usage.csv")
user_device = pd.read_csv("user_device.csv")
devices = pd.read_csv("android_devices.csv")
```

In [27]:
```
user_usage.head()
```

Out[27]:

| | 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 [26]:
```
user_device.head()
```

Out[26]:

| | 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 [13]:
```
%%timeit
pd.merge(user_usage,user_device[['use_id', 'platform', 'device']],on='use_id')
```

```
2.49 ms ± 106 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

In [17]:
```
%%timeit
pysqldf("""SELECT uu.outgoing_mins_per_month, uu.outgoing_sms_per_month, uu.monthly_mb, uu.use_id,
                  ud.platform, ud.device
           FROM user_usage AS uu JOIN user_device AS ud
           ON uu.use_id = ud.use_id
        """)
```

```
9.18 ms ± 1.08 ms per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

In [22]:
```
%%timeit
user_usage.groupby("use_id")["outgoing_sms_per_month"].mean().head()
```

```
851 µs ± 56.8 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

In [21]:
```
%%timeit
pysqldf("""SELECT use_id, AVG(outgoing_sms_per_month)
           FROM user_usage
           GROUP BY use_id
        """).head()
```

```
4.51 ms ± 116 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

Опираясь на полученные данные, можно сказать, что для выполнения таких простых запросов лучше выбирать библиотеку Pandas.