МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Кафедра «Систем обработки информации и управления»

# ОТЧЕТ

**Лабораторная работа № 2**
по курсу «Методы машинного обучения»

Тема: «Изучение библиотек обработки данных»

ИСПОЛНИТЕЛЬ: Паршева А. М.

группа ИУ5-22М _______________
подпись

"__"__________2020г.

ПРЕПОДАВАТЕЛЬ: _____________________
ФИО

_____________________
подпись

"__"__________2020 г.

Москва - 2020

# Лабораторная работа №2. Изучение библиотек обработки данных.

## Задание

1. Выполните первое демонстрационное задание "demo assignment" под названием "Exploratory data analysis with Pandas" со страницы курса https://mlcourse.ai/assignments
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
3. Сравните время выполнения каждого запроса в Pandas и PandaSQL.

## Часть 1

#### Описание данных

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 [64]:

```
import pandas as pd
```

In [65]:

```
data = pd.read_csv('adult.data',header=None,names=['age','work
class','fnlwgt','education',
                                                  'education-
num','marital-status','occupation',
                                                  'relationsh
ip','race','sex','capital-gain','capital-loss',
                                                  'hours-per-
week','native-country','salary'])
data.head()
```

Out[65]:

| | age | workclass | fnlwgt | education | education-num | marital-status | occupat |
|---|---|---|---|---|---|---|---|
| 0 | 39 | State-gov | 77516 | Bachelors | 13 | Never-married | Ad cler |
| 1 | 50 | Self-emp-not-inc | 83311 | Bachelors | 13 | Married-civ-spouse | Ex manage |
| 2 | 38 | Private | 215646 | HS-grad | 9 | Divorced | Handle clear |
| 3 | 53 | Private | 234721 | 11th | 7 | Married-civ-spouse | Handle clear |
| 4 | 28 | Private | 338409 | Bachelors | 13 | Married-civ-spouse | P speci |

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

In [66]:

```
data.groupby('sex').count().reset_index()[['sex','age']]
```

Out[66]:

| | sex | age |
|---|---|---|
| 0 | Female | 10771 |
| 1 | Male | 21790 |

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

In [67]:

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

Out[67]:

```
36.85823043357163
```

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

In [68]:

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

Out[68]:

```
0.42074874850281013
```

**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 [69]:

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

Out[69]:

```
36.78373786407767
```

In [70]:

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

Out[70]:

```
44.24984058155847
```

In [71]:

```
data[data['salary'] == ' <=50K']['age'].std()
```

Out[71]:

```
14.02008849082488
```

In [72]:

```
data[data['salary'] == ' >50K']['age'].std()
```

Out[72]:

```
10.519027719851826
```

**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 [73]:

```
# You code here
education_data = data[data['salary'] == ' >50K'].groupby('educ
ation').count().reset_index()[['education','age']]
educated_people = education_data[education_data['education'].i
sin([' Bachelors', ' Prof-school', ' Assoc-acdm',
                                                    ' Assoc-voc',
' Masters',' Doctorate'])]['age'].sum()
```

In [74]:

```
educated_percent = (educated_people/data[data['salary'] == ' >
50K'].shape[0]) * 100
```

In [75]:

```
if educated_percent > 50:
    print(True)
else:
    print(False)
```

```
True
```

**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.**

In [76]:

```
# You code here
data.groupby('sex')['age'].describe().reset_index()
```

Out[76]:

| | sex | count | mean | std | min | 25% | 50% | 75% | ma |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Female | 10771.0 | 36.858230 | 14.013697 | 17.0 | 25.0 | 35.0 | 46.0 | 90 |
| **1** | Male | 21790.0 | 39.433547 | 13.370630 | 17.0 | 29.0 | 38.0 | 48.0 | 90 |

In [77]:

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

Out[77]:

| | race | count | mean | std | min | 25% | 50% | 75% | m |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Amer-Indian-Eskimo | 311.0 | 37.173633 | 12.447130 | 17.0 | 28.0 | 35.0 | 45.5 | 8 |
| **1** | Asian-Pac-Islander | 1039.0 | 37.746872 | 12.825133 | 17.0 | 28.0 | 36.0 | 45.0 | 9 |
| **2** | Black | 3124.0 | 37.767926 | 12.759290 | 17.0 | 28.0 | 36.0 | 46.0 | 9 |
| **3** | Other | 271.0 | 33.457565 | 11.538865 | 17.0 | 25.0 | 31.0 | 41.0 | 7 |
| **4** | White | 27816.0 | 38.769881 | 13.782306 | 17.0 | 28.0 | 37.0 | 48.0 | 9 |

In [78]:

```
descride_data = data.groupby('race')['age'].describe().reset_index()
descride_data[descride_data['race'] == ' Amer-Indian-Eskimo']['max']
```

Out[78]:

```
0    82.0
Name: max, dtype: float64
```

In [79]:

```
data[data['race'] == ' Amer-Indian-Eskimo']['age'].max()
```

Out[79]:

```
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 [80]:

```
# You code here
marital_status_data = data[data['salary'] == ' >50K'].groupby(
'marital-status').count().reset_index()[['marital-status','age
']]
marital_status_data
```

Out[80]:

| | marital-status | age |
|---|---|---|
| **0** | Divorced | 463 |
| **1** | Married-AF-spouse | 10 |
| **2** | Married-civ-spouse | 6692 |
| **3** | Married-spouse-absent | 34 |
| **4** | Never-married | 491 |
| **5** | Separated | 66 |
| **6** | Widowed | 85 |

In [81]:

```
married = marital_status_data[marital_status_data['marital-sta
tus'].str.startswith(' Married')].sum()[1]
```

In [82]:

```
single = marital_status_data[~(marital_status_data['marital-st
atus'].str.startswith(' Married'))].sum()[1]
```

In [83]:

```
if married > single:
    print('married')
else:
    print('single')
```

married

**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 [84]:

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

Out[84]:

```
99
```

In [85]:

```
people_who_work_max_hours_per_week = data[data['hours-per-week
'] == max_hours_per_week].shape[0]
people_who_work_max_hours_per_week
```

Out[85]:

```
85
```

In [86]:

```
salary_df = data[data['hours-per-week'] == max_hours_per_week]
.groupby('salary').count().reset_index()[['salary', 'age']]
salary_df['percent'] = (salary_df['age'] / people_who_work_max
_hours_per_week) * 100
salary_df[salary_df['salary'] == ' >50K']['percent']
```

Out[86]:

```
1    29.411765
Name: percent, dtype: float64
```

**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 [87]:

```
# You code here
salary_hours_data = data.groupby(['native-country','salary'])[
'hours-per-week']\
.describe().reset_index()[['salary','native-country','mean']]

salary_hours_data[salary_hours_data['native-country'] == ' Jap
an']
```

Out[87]:

| | salary | native-country | mean |
|---|---|---|---|
| 47 | <=50K | Japan | 41.000000 |
| 48 | >50K | Japan | 47.958333 |

# Часть 2

In [88]:

```
android_devices = pd.read_csv('part_2_data/android_devices.csv
')
android_devices.head()
```

Out[88]:

| | 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 |
| 4 | 3Q | OC1020A | OC1020A | OC1020A |

In [89]:

```
user_device = pd.read_csv('part_2_data/user_device.csv')
user_device.head()
```

Out[89]:

| | 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 [90]:

```
user_usage = pd.read_csv('part_2_data/user_usage.csv')
user_usage.head()
```

Out[90]:

| | outgoing_mins_per_month | outgoing_sms_per_month | monthly_mb |
|---|---|---|---|
| **0** | 21.97 | 4.82 | 1557.33 |
| **1** | 1710.08 | 136.88 | 7267.55 |
| **2** | 1710.08 | 136.88 | 7267.55 |
| **3** | 94.46 | 35.17 | 519.12 |
| **4** | 71.59 | 79.26 | 1557.33 |

# Pandas

### Запрос на соединение двух наборов данных

In [91]:

```
def group_pandas():
    return user_device.merge(user_usage,how='inner',on='use_id
')
group_pandas().head()
```

Out[91]:

| | use_id | user_id | platform | platform_version | device | use_type_id | c |
|---|---|---|---|---|---|---|---|
| **0** | 22787 | 12921 | android | 4.3 | GT-I9505 | 1 | |
| **1** | 22788 | 28714 | android | 6.0 | SM-G930F | 1 | |
| **2** | 22789 | 28714 | android | 6.0 | SM-G930F | 1 | |
| **3** | 22790 | 29592 | android | 5.1 | D2303 | 1 | |
| **4** | 22792 | 28217 | android | 5.1 | SM-G361F | 1 | |

### Запрос на группировку набора данных с использованием функций агрегирования

In [92]:

```
def join_pandas():
    return user_device.groupby('platform').count().reset_index
()[['platform','user_id']]
join_pandas()
```

Out[92]:

| | platform | user_id |
|---|---|---|
| **0** | android | 184 |
| **1** | ios | 88 |

## PandaSQL

In [93]:

```
import pandasql as ps

ps.sqldf('select * from user_device limit 10', locals())
```

Out[93]:

| | 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 |
| **5** | 22787 | 12921 | android | 4.3 | GT-I9505 | 1 |
| **6** | 22788 | 28714 | android | 6.0 | SM-G930F | 1 |
| **7** | 22789 | 28714 | android | 6.0 | SM-G930F | 1 |
| **8** | 22790 | 29592 | android | 5.1 | D2303 | 1 |
| **9** | 22791 | 28775 | ios | 10.2 | iPhone6,2 | 3 |

**Запрос на соединение двух наборов данных**

In [94]:

```
def join_pandasql(user_device,user_usage):
    return ps.sqldf('select * from user_device join user_usage
on user_device.use_id = user_usage.use_id', locals())

join_pandasql(user_device,user_usage).head()
```

Out[94]:

| | use_id | user_id | platform | platform_version | device | use_type_id | c |
|---|---|---|---|---|---|---|---|
| **0** | 22787 | 12921 | android | 4.3 | GT-I9505 | 1 | |
| **1** | 22788 | 28714 | android | 6.0 | SM-G930F | 1 | |
| **2** | 22789 | 28714 | android | 6.0 | SM-G930F | 1 | |
| **3** | 22790 | 29592 | android | 5.1 | D2303 | 1 | |
| **4** | 22792 | 28217 | android | 5.1 | SM-G361F | 1 | |

**Запрос на группировку набора данных с использованием функций агрегирования**

In [95]:

```
def group_pandasql(user_device):
    return ps.sqldf('select platform, count(user_id) from user
_device group by platform', locals())

group_pandasql(user_device)
```

Out[95]:

| | platform | count(user_id) |
|---|---|---|
| **0** | android | 184 |
| **1** | ios | 88 |

**Сравнение времени выполнения запросов библиотек Pandas и PandaSQL**

In [96]:

```
import timeit

timeit.timeit("group_pandasql(user_device)", setup="from __mai
n__ import group_pandasql, user_device", number=1)
```

Out[96]:

```
0.011258081000050879
```

In [97]:

```
timeit.timeit("join_pandasql(user_device,user_usage)", setup="
from __main__ import join_pandasql,user_device,user_usage", nu
mber=1)
```

Out[97]:

```
0.022365900000067995
```

In [98]:

```
timeit.timeit("join_pandas", setup="from __main__ import join_
pandas", number=1)
```

Out[98]:

```
6.099999154685065e-07
```

In [99]:

```
timeit.timeit("group_pandas", setup="from __main__ import grou
p_pandas", number=1)
```

Out[99]:

```
7.779999577905983e-07
```

На основе полученных данных можно сделать вывод о том, что функции объединения и группировки работают быстрее в библиотеке Pandas