**Министерство образования и науки 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)**

# Методы машинного обучения

## *Отчёт по лабораторной работе № 2*

**«Изучение библиотек обработки данных.»**

Выполнил:
студент группы ИУ5 – 23М Крутов Т.Ю.

Преподаватель: Гапанюк Ю.Е.

*2020г.*

In [112]:

```
import numpy as np
import pandas as pd
import pandasql  as ps
```

In [113]:

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

Out[113]:

| | 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 | Fe |

In [42]:

```
data.describe()
```

Out[42]:

| | age | fnlwgt | education-num | capital-gain | capital-loss | hours-per-week |
|---|---|---|---|---|---|---|
| **count** | 32561.000000 | 3.256100e+04 | 32561.000000 | 32561.000000 | 32561.000000 | 32561.000000 |
| **mean** | 38.581647 | 1.897784e+05 | 10.080679 | 1077.648844 | 87.303830 | 40.437456 |
| **std** | 13.640433 | 1.055500e+05 | 2.572720 | 7385.292085 | 402.960219 | 12.347429 |
| **min** | 17.000000 | 1.228500e+04 | 1.000000 | 0.000000 | 0.000000 | 1.000000 |
| **25%** | 28.000000 | 1.178270e+05 | 9.000000 | 0.000000 | 0.000000 | 40.000000 |
| **50%** | 37.000000 | 1.783560e+05 | 10.000000 | 0.000000 | 0.000000 | 40.000000 |
| **75%** | 48.000000 | 2.370510e+05 | 12.000000 | 0.000000 | 0.000000 | 45.000000 |
| **max** | 90.000000 | 1.484705e+06 | 16.000000 | 99999.000000 | 4356.000000 | 99.000000 |

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

In [3]:

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

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

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

In [12]:

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

```
36.85823043357163
```

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

In [5]:

```
print((data['native-country'] == "Germany").value_counts()/9713)
```

```
False    3.338207
True     0.014105
Name: native-country, dtype: float64
```

## 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 [9]:

```
mean_less = data.loc[data['salary']=="<=50K", 'age'].mean()  # 4-5
mean_more = data.loc[data['salary']==">50K", 'age'].mean()
std_less = data.loc[data['salary']=="<=50K", 'age'].std()
std_more = data.loc[data['salary']==">50K", 'age'].std()
print('>50K - The average age of the rich: {}, \n standard deviation: {} \n'.format(mean_mo
print('<=50K - The average age of the poor: {}, \n standard deviation: {}'.format(mean_less
```

```
>50K - The average age of the rich: 44.2,
 standard deviation: 10.5

<=50K - The average age of the poor: 36.8,
 standard deviation: 14.0
```

## 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 [6]:

```
print(data.loc[data['salary']==">50K", 'education'].unique()) # 6
```

```
['HS-grad' 'Masters' 'Bachelors' 'Some-college' 'Assoc-voc' 'Doctorate'
 'Prof-school' 'Assoc-acdm' '7th-8th' '12th' '10th' '11th' '9th' '5th-6th'
 '1st-4th']
```

## 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 [10]:

```
group = data.groupby(['race','sex']) # 7
group.describe()
```

Out[10]:

| | | age | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | count | mean | std | min | 25% | 50% | 75% | max | count | |
| race | sex | | | | | | | | | | |
| Amer-Indian-Eskimo | Female | 119.0 | 37.117647 | 13.114991 | 17.0 | 27.0 | 36.0 | 46.00 | 80.0 | 119.0 | 112950 |
| | Male | 192.0 | 37.208333 | 12.049563 | 17.0 | 28.0 | 35.0 | 45.00 | 82.0 | 192.0 | 125715 |
| Asian-Pac-Islander | Female | 346.0 | 35.089595 | 12.300845 | 17.0 | 25.0 | 33.0 | 43.75 | 75.0 | 346.0 | 147452 |
| | Male | 693.0 | 39.073593 | 12.883944 | 18.0 | 29.0 | 37.0 | 46.00 | 90.0 | 693.0 | 166175 |
| Black | Female | 1555.0 | 37.854019 | 12.637197 | 17.0 | 28.0 | 37.0 | 46.00 | 90.0 | 1555.0 | 212971 |
| | Male | 1569.0 | 37.682600 | 12.882612 | 17.0 | 27.0 | 36.0 | 46.00 | 90.0 | 1569.0 | 242920 |
| Other | Female | 109.0 | 31.678899 | 11.631599 | 17.0 | 23.0 | 29.0 | 39.00 | 74.0 | 109.0 | 172519 |
| | Male | 162.0 | 34.654321 | 11.355531 | 17.0 | 26.0 | 32.0 | 42.00 | 77.0 | 162.0 | 213679 |
| White | Female | 8642.0 | 36.811618 | 14.329093 | 17.0 | 25.0 | 35.0 | 46.00 | 90.0 | 8642.0 | 183549 |
| | Male | 19174.0 | 39.652498 | 13.436029 | 17.0 | 29.0 | 38.0 | 49.00 | 90.0 | 19174.0 | 188987 |

10 rows × 48 columns

## 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 [65]:

```
sex= data['sex'] == 'Male'
mar = data['marital-status'].isin(['Married-civ-spouse', 'Married-spouse-absent' , 'Married
bach = data['marital-status'].isin(['Never-married', 'Separated', 'Divorced','Widowed'])
print(s)
```

```
0        False
1         True
2        False
3         True
4         True
         ...
32556     True
32557     True
32558    False
32559    False
32560     True
Name: marital-status, Length: 32561, dtype: bool
```

In [67]:

```
data.loc[sex & mar, 'salary'].value_counts()
```

Out[67]:

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

In [66]:

```
data.loc[sex & bach, 'salary'].value_counts()
```

Out[66]:

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

### 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 [68]:

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

```
99
```

In [77]:

```
(data['hours-per-week'] == 99).value_counts()
```

Out[77]:

```
False    32476
True        85
Name: hours-per-week, dtype: int64
```

In [79]:

```
data.loc[data['hours-per-week'] == 99, 'salary'].value_counts()
```

Out[79]:

```
<=50K    60
>50K     25
Name: salary, dtype: int64
```

In [ ]:

```
25/85*100
```

## 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 [96]:

```
group_sal = data.groupby(['native-country','salary'])
group_sal['hours-per-week'].mean().round(1)
```

Out[96]:

```
native-country  salary
?               <=50K     40.2
                >50K      45.5
Cambodia        <=50K     41.4
                >50K      40.0
Canada          <=50K     37.9
                          ...
United-States   >50K      45.5
Vietnam         <=50K     37.2
                >50K      39.2
Yugoslavia      <=50K     41.6
                >50K      49.5
Name: hours-per-week, Length: 82, dtype: float64
```

In [97]:

```
for (country, salary), avg in group_sal:
    print(country, salary, round(avg['hours-per-week'].mean(), 3))
```

```
? <=50K 40.165
? >50K 45.548
Cambodia <=50K 41.417
Cambodia >50K 40.0
Canada <=50K 37.915
Canada >50K 45.641
China <=50K 37.382
China >50K 38.9
Columbia <=50K 38.684
Columbia >50K 50.0
Cuba <=50K 37.986
Cuba >50K 42.44
Dominican-Republic <=50K 42.338
Dominican-Republic >50K 47.0
Ecuador <=50K 38.042
Ecuador >50K 48.75
El-Salvador <=50K 36.031
El-Salvador >50K 45.0
England <=50K 40.483
```

In [104]:

```
jap1 = group_sal.get_group(('Japan','<=50K'))
jap2 = group_sal.get_group(('Japan','>50K'))
hour1 = jap1['hours-per-week'].mean().round(1)
hour2 = jap2['hours-per-week'].mean().round(1)
print ( 'Japan - >50K - hours-per-week: {} \n Japan - <=50K - hours-per-week: {} '.format(h
```

```
Japan - >50K - hours-per-week: 41.0
 Japan - <=50K - hours-per-week: 48.0
```

# PANDASQL

In [105]:

```
user_usage = pd.read_csv('user_usage.csv')
user_device = pd.read_csv('user_device.csv')
android_devices = pd.read_csv('android_devices.csv')
```

In [106]:

```
user_usage.head()
```

Out[106]:

| | 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 [107]:

```
user_device.head()
```

Out[107]:

| | 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 [109]:

```
android_devices.head(10)
```

Out[109]:

| | 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 |
| **5** | 7Eleven | IN265 | IN265 | IN265 |
| **6** | A.O.I. ELECTRONICS FACTORY | A.O.I. | TR10CS1_11 | TR10CS1 |
| **7** | AG Mobile | AG BOOST 2 | BOOST2 | E4010 |
| **8** | AG Mobile | AG Flair | AG_Flair | Flair |
| **9** | AG Mobile | AG Go Tab Access 2 | AG_Go_Tab_Access_2 | AG_Go_Tab_Access_2 |

In [134]:

```
import time
```

## Произвольный запрос на соединение двух наборов данных

In [132]:

```
def resultpanda():
    result = pd.merge (user_usage, user_device
                [['use_id', 'platform', 'device']],
                on = 'use_id')
    return result
```

In [152]:

```
data_mobile = resultpanda()
data_mobile.head()
```

Out[152]:

| | outgoing_mins_per_month | outgoing_sms_per_month | monthly_mb | use_id | platform | device |
|---|---|---|---|---|---|---|
| **0** | 21.97 | 4.82 | 1557.33 | 22787 | android | GT-I9505 |
| **1** | 1710.08 | 136.88 | 7267.55 | 22788 | android | SM-G930F |
| **2** | 1710.08 | 136.88 | 7267.55 | 22789 | android | SM-G930F |
| **3** | 94.46 | 35.17 | 519.12 | 22790 | android | D2303 |
| **4** | 71.59 | 79.26 | 1557.33 | 22792 | android | SM-G361F |

In [135]:

```
start_time = time.time()
resultpanda()
print("--- %s seconds ---" % (time.time() - start_time))
```

```
--- 0.007946252822875977 seconds ---
```

In [127]:

```
def resultSQL (user_usage, user_device):
    join_query = '''
        SELECT
            usage.outgoing_mins_per_month,
            usage.outgoing_sms_per_month,
            usage.monthly_mb,
            usage.use_id,
            device.platform, device.device

        FROM user_usage AS usage
        JOIN user_device AS device ON (usage.use_id = device.use_id)


    '''
    return ps.sqldf(join_query, locals())
```

In [136]:

```
start_time = time.time()
resultSQL(user_usage,user_device)
print("--- %s seconds ---" % (time.time() - start_time))
```

```
--- 0.019946813583374023 seconds ---
```

## Произвольный запрос на группировку набора данных с использованием функций агрегирования

In [148]:

```
group = resultpanda().groupby('device')['outgoing_mins_per_month'].mean()
group
```

Out[148]:

```
device
A0001                    170.395000
C6603                     92.520000
D2303                     96.845000
D5503                    146.450000
D5803                    244.880000
D6603                    362.010000
E6653                    135.090000
EVA-L09                  115.260000
F3111                     46.262500
GT-I8190N                 85.970000
GT-I9195                 199.430000
GT-I9300                 167.560000
GT-I9505                 162.770909
GT-I9506                 119.800000
GT-I9515                 180.310000
GT-N7100                  16.340000
```

In [149]:

```
start_time = time.time()
group = resultpanda().groupby('device')['outgoing_mins_per_month'].mean()
print("--- %s seconds ---" % (time.time() - start_time))
```

```
--- 0.010970592498779297 seconds ---
```

In [162]:

```
query = '''
    SELECT
        device,
        avg('outgoing_mins_per_month')
    FROM data_mobile
    GROUP BY device
    '''
result = ps.sqldf(query, locals())
```

Out[162]:

| | device | avg('outgoing_mins_per_month') |
|---|---|---|
| **0** | A0001 | 0.0 |
| **1** | C6603 | 0.0 |
| **2** | D2303 | 0.0 |
| **3** | D5503 | 0.0 |
| **4** | D5803 | 0.0 |
| **5** | D6603 | 0.0 |
| **6** | E6653 | 0.0 |
| **7** | EVA-L09 | 0.0 |
| **8** | F3111 | 0.0 |
| **9** | GT-I8190N | 0.0 |

In [163]:

```
start_time = time.time()
result = ps.sqldf(query, locals())
print("--- %s seconds ---" % (time.time() - start_time))
```

```
--- 0.012966394424438477 seconds ---
```