МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет «Информатика и система управления»
Кафедра ИУ-5 «Системы обработки информации и управления»

ОТЧЕТ

Лабораторная работа №2
по курсу «Методы машинного обучения»

Исполнитель - студент группы ИУ5-21М:

Кауров Максим __________

Москва – 2020 год

Часть 1

```
import numpy as np
import pandas as pd
import time
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')
```

```
#Загружаем датасет
from google.colab import files
files.upload()
```

Выбрать файлы adult.data.csv

- **adult.data.csv**(application/vnd.ms-excel) - 3518607 bytes, last modified: 06.04.2020 - 100% done

```
Saving adult.data.csv to adult.data.csv
{'adult.data.csv': b'age,workclass,fnlwgt,education,education-num,marital-status,occu
```

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

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband

```
print("Список колонок с типами данных")
print(data.dtypes)
```

```
Список колонок с типами данных
age                int64
workclass         object
fnlwgt             int64
education         object
education-num      int64
marital-status    object
occupation        object
relationship      object
race              object
sex               object
capital-gain       int64
capital-loss       int64
hours-per-week     int64
native-country    object
salary            object
dtype: object
```

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

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

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

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

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

```
36.85823043357163
```

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

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

```
0.004207487485028101
```

4-5. What are the mean and standard deviation of age for those who earn more than 50K per year
50K per year?

```
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)))
```

6. Is it true that people who earn more than 50K have at least high school education? (educatio
   Assoc-voc, Masters or Doctorate feature)

```
data.loc[data['salary'] == '>50K', 'education'].unique() # Ответ: Нет
```

```
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)
```

7. Display age statistics for each race (race feature) and each gender (sex feature). Use groupb
   men of Amer-Indian-Eskimo race.

```
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
```

8. Among whom is the proportion of those who earn a lot (>50K) greater: married or single mer married those who have a marital-status starting with Married (Married-civ-spouse, Married-s rest are considered bachelors.

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

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

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

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

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

```
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
```

9. What is the maximum number of hours a person works per week (hours-per-week feature)? hours, and what is the percentage of those who earn a lot (>50K) among them?

```
max_load = data['hours-per-week'].max()
print("Max time - {0} hours./week.".format(max_load))

num_workaholics = data[data['hours-per-week'] == max_load].shape[0]
print("Total number of such hard workers {0}".format(num_workaholics))

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%
```

10. Count the average time of work (hours-per-week) for those who earn a little and a lot (salary) these be for Japan?

```
pd.crosstab(data['native-country'], data['salary'],
           values=data['hours-per-week'], aggfunc=np.mean).T
```

native-country	?	Cambodia	Canada	China	Columbia	Cuba	Dominican-Republic
salary							
<=50K	40.164760	41.416667	37.914634	37.381818	38.684211	37.985714	42.338235
>50K	45.547945	40.000000	45.641026	38.900000	50.000000	42.440000	47.000000

Часть 2

```
!pip  install -U  pandasql
import pandas as pd
import pandasql as ps
from datetime import datetime
import seaborn
```

```
Collecting pandasql
  Downloading https://files.pythonhosted.org/packages/6b/c4/ee4096ffa2eeeca0c749b26f0
Requirement already satisfied, skipping upgrade: numpy in /usr/local/lib/python3.6/di
Requirement already satisfied, skipping upgrade: pandas in /usr/local/lib/python3.6/d
Requirement already satisfied, skipping upgrade: sqlalchemy in /usr/local/lib/python3
Requirement already satisfied, skipping upgrade: pytz>=2017.2 in /usr/local/lib/pytho
Requirement already satisfied, skipping upgrade: python-dateutil>=2.6.1 in /usr/local
Requirement already satisfied, skipping upgrade: six>=1.5 in /usr/local/lib/python3.6
Building wheels for collected packages: pandasql
  Building wheel for pandasql (setup.py) ... done
  Created wheel for pandasql: filename=pandasql-0.7.3-cp36-none-any.whl size=26819 sh
  Stored in directory: /root/.cache/pip/wheels/53/6c/18/b87a2e5fa8a82e9c026311de56210
Successfully built pandasql
Installing collected packages: pandasql
Successfully installed pandasql-0.7.3
```

```
#Загружаем 2 датасета
from google.colab import files
files.upload()
```

Выбрать файлы user_device.csv

- **user_device.csv**(application/vnd.ms-excel) - 9437 bytes, last modified: 06.04.2020 - 100% done

```
Saving user_device.csv to user_device.csv
{'user_device.csv': b'use_id,user_id,platform,platform_version,device,use_type_id\n22
```

```
from google.colab import files
files.upload()
```

Выбрать файлы user_usage.csv

- **user_usage.csv**(application/vnd.ms-excel) - 6432 bytes, last modified: 06.04.2020 - 100% done

```
Saving user_usage.csv to user_usage.csv
{'user_usage.csv': b'outgoing_mins_per_month,outgoing_sms_per_month,monthly_mb,use_id
```

```
data_1 = pd.read_csv('user_device.csv')
data_2 = pd.read_csv('user_usage.csv')
```

```
data_1.head()
```

	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

```
data_2.head()
```

	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

```
#pandas code
start_time = time.time()
join_df = pd.merge(data_1,
                   data_2[data_1.use_type_id == 1],
                   how = 'inner',
                   right_on ='use_id',
                   left_on = 'use_id')
join_df = join_df[['use_id', 'user_id', 'platform', 'device', 'outgoing_mins_per_month', '
print(join_df)
print("---  %s  seconds ---"  % (time.time()  - start_time))
```

```
    use_id  user_id platform      device  outgoing_mins_per_month  monthly_mb
0    22789    28714  android    SM-G930F                  1710.08     7267.55
1    22792    28217  android    SM-G361F                    71.59     1557.33
2    22793    28217  android    SM-G361F                    71.59     1557.33
3    22794    28217  android    SM-G361F                    71.59      519.12
4    22795    28217  android    SM-G361F                    71.59      519.12
..     ...      ...      ...         ...                      ...         ...
88   23041    28953  android    SM-G900F                   198.59     5191.12
89   23043    28953  android    SM-G900F                   198.59     5191.12
90   23044    28953  android    SM-G900F                   198.59     3114.67
91   23046    29454  android  Moto G (4)                   106.65     5191.12
92   23049    29725  android    SM-G900F                   344.53      519.12

[93 rows x 6 columns]
---     0.021392345428466797    seconds ---
```

```
#pandaSQL code
start_time = time.time()
join_query = '''
    SELECT
        d1.use_id,
        user_id,
        platform,
        device,
        outgoing_mins_per_month,
        monthly_mb
    FROM data_1 as d1 JOIN data_2 as d2 ON (d1.use_id = d2.use_id)
        WHERE (use_type_id = 1)
'''
pandasqlcode = ps.sqldf(join_query)
print(pandasqlcode)
print("---  %s  seconds ---"  % (time.time()  - start_time))
```

```
     use_id  user_id platform                 device  \
0     22787    12921  android                GT-I9505
1     22788    28714  android                SM-G930F
2     22789    28714  android                SM-G930F
3     22790    29592  android                   D2303
4     22792    28217  android                SM-G361F
..      ...      ...      ...                    ...
152   23043    28953  android                SM-G900F
153   23044    28953  android                SM-G900F
154   23046    29454  android              Moto G (4)
155   23049    29725  android                SM-G900F
156   23053    20257  android  Vodafone Smart ultra 6

     outgoing_mins_per_month  monthly_mb
0                      21.97     1557.33
1                    1710.08     7267.55
2                    1710.08     7267.55
3                      94.46      519.12
4                      71.59     1557.33
..                       ...         ...
152                   198.59     5191.12
153                   198.59     3114.67
154                   106.65     5191.12
155                   344.53      519.12
156                    42.75     5191.12

[157 rows x 6 columns]
---     0.034714698791503906    seconds ---
```

Как можно заметить код PandaSQL выполняется на 50% дольше

Группировка

```
#Код Pandas
start_time  = time.time()
pandascode = pd.DataFrame(join_df.groupby('device').monthly_mb.mean())
pandascode = pandascode.sort_values('monthly_mb')
print(pandascode)
print("---  %s  seconds ---"  % (time.time()  - start_time))
```

```
                    monthly_mb
device
HUAWEI CUN-L01     11.680000
HTC Desire 620     74.400000
GT-I8190N         407.010000
D2303             519.120000
HTC Desire 626    519.120000
SM-J320FN         778.665000
GT-I9300          894.580000
SM-G361F          934.404000
HTC One S        1038.210000
GT-I9195         1211.260000
iPhone7,2        1271.390000
SM-G935F         1384.303333
SM-A310F         1557.330000
SM-A500FU        1557.330000
SM-G903F         1557.330000
SM-G800F         1557.330000
SM-G920F         1557.330000
VF-795           1557.330000
SM-G360F         1557.330000
LG-H815          1557.330000
Lenovo K51c78    1557.330000
HTC Desire 530   1557.330000
C6603            1557.330000
D5503            1557.330000
EVA-L09          1557.330000
SM-G531F         2076.450000
ONE A2003        2076.450000
SM-A300FU        2336.000000
HUAWEI VNS-L31   3114.670000
ONEPLUS A3003    3823.610000
GT-I9505         3924.252857
SM-G925F         4152.880000
HTC Desire 825   4513.560000
SM-G900F         4556.647778
Moto G (4)       5191.120000
SM-N910F         5437.576667
D6603            7267.550000
SM-G930F         8305.775000
HTC One mini 2  10382.210000
X11             12458.670000
A0001           15573.330000
HTC Desire 510  15573.330000
GT-N7100        20764.450000
---     0.010747194290161133    seconds ---
```

```
#Код PandaSQL
start_time  = time.time()
print(ps.sqldf('select  "device", avg("monthly_mb") as AVG_Traf from  pandasqlcode  group
print("---  %s  seconds ---"  % (time.time()  - start_time))
```

```
                    device      AVG_Traf
0           HUAWEI CUN-L01     11.680000
1           HTC Desire 620     74.400000
2           MotoE2(4G-LTE)    212.640000
3                GT-I8190N    407.010000
4                 GT-I9300    464.185000
5                    D2303    519.120000
6           HTC Desire 626    519.120000
7                 GT-I9506    803.240000
8                SM-J320FN    830.574000
9                 SM-G361F    934.404000
10               HTC One S   1038.210000
11                GT-I9195   1211.260000
12                   C6603   1557.330000
13                   D5503   1557.330000
14                   D5803   1557.330000
15                 EVA-L09   1557.330000
16                GT-I9515   1557.330000
17          HTC Desire 530   1557.330000
18                 LG-H815   1557.330000
19           Lenovo K51c78   1557.330000
20                Nexus 5X   1557.330000
21                SM-A310F   1557.330000
22               SM-A500FU   1557.330000
23                SM-G360F   1557.330000
24                SM-G800F   1557.330000
25                SM-G903F   1557.330000
26                  VF-795   1557.330000
27               SM-A300FU   1687.112500
28                SM-G920F   1985.168000
29                   F3111   2076.450000
30               ONE A2003   2076.450000
31                SM-G531F   2076.450000
32              HTC One M9   2362.070000
33          HUAWEI VNS-L31   3114.670000
34                SM-G925F   3633.775000
35           ONEPLUS A3003   3823.610000
36                SM-G900F   3841.427333
37                SM-G935F   4568.182000
38                   E6653   5191.120000
39              Moto G (4)   5191.120000
40  Vodafone Smart ultra 6   5191.120000
41          HTC Desire 825   5498.970000
42                GT-I9505   5564.726364
43              HTC One_M8   6577.120000
44                   D6603   7267.550000
45                SM-G930F   7959.700000
46                SM-N910F   8038.370000
47                GT-N7100  11939.560000
48                     X11  12458.670000
49          HTC Desire 510  12562.488000
50          HTC One mini 2  13842.956667
51                   A0001  15573.330000
52                SM-N9005  16611.550000
---     0.025293588638305664    seconds ---
```

Как можно заметить код PandaSQL выполняется в 2.5 раза больше, чем код Pandas