МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Кафедра «Систем обработки информации и управления»

ОТЧЕТ

Лабораторная работа №2
по курсу «Методы машинного обучения»

Изучение библиотек обработки данных.

ИСПОЛНИТЕЛЬ: Семенова Е. В.

группа ИУ5-23М _____
подпись

"__"_____2019 г.

Москва - 2019

Задание:

Часть 1.

- Выполнить демонстрационное задание "demo assignment"

Часть 2.

- Выполнить следующие запросы с использованием двух различных библиотек - Pandas и PandaSQL:

1) один произвольный запрос на соединение двух наборов данных;

2) один произвольный запрос на группировку набора данных с использованием функций агрегирования.

- Сравнить время выполнения каждого запроса в Pandas и PandaSQL.

Часть 1

In [41]:

```
import numpy as np
import pandas as pd
pd.set_option('display.max.columns', 100)
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [42]:

```
data = pd.read_csv('datasets/adult.data.csv', sep=", ")
data.head()
```

Out[42]:

	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

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

In [43]:

```
print(f'Число мужчин: {data[data["sex"]=="Male"].count()["sex"]}')
print(f'Число женщин: {data[data["sex"]=="Female"].count()["sex"]}')
```

```
Число мужчин: 21790
Число женщин: 10771
```

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

In [44]:

```
print(f'Средний возраст женщин: {data.loc[data["sex"]=="Female", "age"].mean()}')
```

```
Средний возраст женщин: 36.85823043357163
```

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

In [45]:

```
print(f'Процент жителей Германии: {round((data[data["native-country"]=="Germany"].count()/data.count())["native-country"] * 100, 2)}%')
```

```
Процент жителей Германии: 0.42%
```

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 [46]:

```
less = data.loc[data["salary"]=="<=50K", "age"]
more = data.loc[data["salary"]==">50K", "age"]
print(f'Средний возраст (>50k) - {round(less.mean(), 1)} and (<=50k) - {round(more.
mean(), 1)}')
print(f'Среднее отклонение возраста (>50k) - {round(less.std(), 1)} and (<=50k) - {ro
und(more.std(), 1)}')
```

```
Средний возраст (>50k) - 36.8 and (<=50k) - 44.2
Среднее отклонение возраста (>50k) - 14.0 and (<=50k) - 10.5
```

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 [47]:

```
data[data["salary"]==">50K"]["education"].value_counts()
```

Out[47]:

```
Bachelors       2221
HS-grad         1675
Some-college    1387
Masters          959
Prof-school      423
Assoc-voc        361
Doctorate        306
Assoc-acdm       265
10th              62
11th              60
7th-8th           40
12th              33
9th               27
5th-6th           16
1st-4th            6
Name: education, dtype: int64
```

Вывод: нет, потому что есть ряды с более низким уровнем образования

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 [48]:

```
data.groupby(['race', 'sex'])
```

Out[48]:

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x12a97e0b8>
```

In [49]:

```
for (race, sex), group_data in data.groupby(['race', 'sex']):
    print(f'Race: {race}, sex: {sex}')
    print(group_data['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 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 [50]:

```
married_condition = data["marital-status"].str.startswith('Married')
salary_condition = data["salary"] == ">50K"
sex_condition = data["sex"] == "Male"
married = data.loc[married_condition & salary_condition & sex_condition, "age"].
count()
not_maried = data.loc[~married_condition & salary_condition & sex_condition, "ag
e"].count()
print(f'Среди женатых - {married}, среди неженатых {not_maried}')
```

```
Среди женатых - 5965, среди неженатых 697
```

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 [51]:

```
max_hours = data["hours-per-week"].max()
person_count = data.loc[data["hours-per-week"] == max_hours, "hours-per-week"].c
ount()
print(f'Max: {max_hours}')
print(f'Число людей: {person_count}')
print(f'Число людей с заработком >50K: {round((data.loc[(data["hours-per-week"] == m
ax_hours) & (data["salary"] == ">50K"), "hours-per-week"].count()) / person_coun
t * 100, 1)}%')
```

```
Max: 99
Число людей: 85
Число людей с заработком >50K: 29.4%
```

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 [52]:

```
for (country, salary), group_data in data.groupby(['native-country', 'salary']):
    print(country, salary, round(group_data['hours-per-week'].mean(), 2))
```

? <=50K 40.16
? >50K 45.55
Cambodia <=50K 41.42
Cambodia >50K 40.0
Canada <=50K 37.91
Canada >50K 45.64
China <=50K 37.38
China >50K 38.9
Columbia <=50K 38.68
Columbia >50K 50.0
Cuba <=50K 37.99
Cuba >50K 42.44
Dominican-Republic <=50K 42.34
Dominican-Republic >50K 47.0
Ecuador <=50K 38.04
Ecuador >50K 48.75
El-Salvador <=50K 36.03
El-Salvador >50K 45.0
England <=50K 40.48
England >50K 44.53
France <=50K 41.06
France >50K 50.75
Germany <=50K 39.14
Germany >50K 44.98
Greece <=50K 41.81
Greece >50K 50.62
Guatemala <=50K 39.36
Guatemala >50K 36.67
Haiti <=50K 36.33
Haiti >50K 42.75
Holand-Netherlands <=50K 40.0
Honduras <=50K 34.33
Honduras >50K 60.0
Hong <=50K 39.14
Hong >50K 45.0
Hungary <=50K 31.3
Hungary >50K 50.0
India <=50K 38.23
India >50K 46.48
Iran <=50K 41.44
Iran >50K 47.5
Ireland <=50K 40.95
Ireland >50K 48.0
Italy <=50K 39.62
Italy >50K 45.4
Jamaica <=50K 38.24
Jamaica >50K 41.1
Japan <=50K 41.0
Japan >50K 47.96
Laos <=50K 40.38
Laos >50K 40.0
Mexico <=50K 40.0
Mexico >50K 46.58
Nicaragua <=50K 36.09
Nicaragua >50K 37.5
Outlying-US(Guam-USVI-etc) <=50K 41.86
Peru <=50K 35.07
Peru >50K 40.0
Philippines <=50K 38.07
Philippines >50K 43.03
Poland <=50K 38.17

```
Poland >50K 39.0
Portugal <=50K 41.94
Portugal >50K 41.5
Puerto-Rico <=50K 38.47
Puerto-Rico >50K 39.42
Scotland <=50K 39.44
Scotland >50K 46.67
South <=50K 40.16
South >50K 51.44
Taiwan <=50K 33.77
Taiwan >50K 46.8
Thailand <=50K 42.87
Thailand >50K 58.33
Trinadad&Tobago <=50K 37.06
Trinadad&Tobago >50K 40.0
United-States <=50K 38.8
United-States >50K 45.51
Vietnam <=50K 37.19
Vietnam >50K 39.2
Yugoslavia <=50K 41.6
Yugoslavia >50K 49.5
```

Часть 2

Соединение двух наборов данных

In [53]:

```
import pandasql as ps
```

In [54]:

```
data_usage = pd.read_csv('datasets/user_usage.csv', sep=",")
data_usage.head()
```

Out[54]:

	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 [55]:

```
data_device = pd.read_csv('datasets/user_device.csv', sep=",")
data_device.head()
```

Out[55]:

	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 [56]:

```
import time

def count_mean_time(func, params, N =5):
    total_time = 0
    for i in range(N):
        time1 = time.time()
        if len(params) == 1:
            tmp_df = func(params[0])
        elif len(params) == 2:
            tmp_df = func(params[0], params[1])
        time2 = time.time()
        total_time += (time2 - time1)
    return total_time/N
```

In [57]:

```
def pandas_merge(data_usage, data_device):
    return pd.merge(data_usage, data_device, left_on='use_id', right_on='use_id'
, how='left')

result = pandas_merge(data_usage, data_device)
result.head()
```

Out[57]:

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id	user_id	platfo
0	21.97	4.82	1557.33	22787	12921.0	andr
1	1710.08	136.88	7267.55	22788	28714.0	andr
2	1710.08	136.88	7267.55	22789	28714.0	andr
3	94.46	35.17	519.12	22790	29592.0	andr
4	71.59	79.26	1557.33	22792	28217.0	andr

In [58]:

```
def pandasql_merge(data_usage, data_device):
    simple_query = '''
        SELECT *
        FROM data_usage
        LEFT JOIN data_device ON data_device.use_id = data_usage.use_id
        '''
    return ps.sqldf(simple_query, locals())
pandasql_merge(data_usage, data_device).head()
```

Out[58]:

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id	use_id	user_id
0	21.97	4.82	1557.33	22787	22787.0	12921.0
1	1710.08	136.88	7267.55	22788	22788.0	28714.0
2	1710.08	136.88	7267.55	22789	22789.0	28714.0
3	94.46	35.17	519.12	22790	22790.0	29592.0
4	71.59	79.26	1557.33	22792	22792.0	28217.0

In [59]:

```
print(f'pandas - {count_mean_time(pandas_merge, [data_usage, data_device])}')
print(f'pandasql - {count_mean_time(pandasql_merge, [data_usage, data_device])}'
)
```

```
pandas - 0.005152606964111328
pandasql - 0.017931175231933594
```

Группировка набора данных с использованием функций агрегирования

In [60]:

```
aggregations = {
    "outgoing_sms_per_month": {
        "sum": "sum",
        "mean": "mean"
    }
}

def pandas_agg(result):
    return result.groupby("platform").agg(aggregations)

pandas_agg(result).head()
```

Out[60]:

	outgoing_sms_per_month	
	sum	mean
platform		
android	13400.67	85.354586
ios	587.95	293.975000

In [61]:

```
def pandasql_agg(result):
    simple_query = '''
        SELECT platform, SUM(outgoing_sms_per_month), AVG(outgoing_sms_per_mont
h)
        FROM result
        GROUP BY platform
        '''
    return ps.sqldf(simple_query, locals())
pandasql_agg(result).head()
```

Out[61]:

	platform	SUM(outgoing_sms_per_month)	AVG(outgoing_sms_per_month)
0	None	9763.77	120.540370
1	android	13400.67	85.354586
2	ios	587.95	293.975000

In [62]:

```
print(f'pandas - {count_mean_time(pandas_agg, [result])}')
print(f'pandasql - {count_mean_time(pandasql_agg, [result])}')
```

```
pandas - 0.004696559906005859
pandasql - 0.01288299560546875
```