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:	___Сметанкин К.И__
	ФИО
группа ИУ5-22М	____________________
	подпись
	"__"_________2020 г.
ПРЕПОДАВАТЕЛЬ:	___Гапанюк Ю.Е___
	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 (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

Exploratory data analysis with Pandas

Same assignment as a Kaggle Kernel + solution.

In this task you should use Pandas to answer a few questions about the Adult dataset. (You don't have to download the data – it's already in the repository). Choose the answers in the web-form.

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 [0]:

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

In [0]:

```
url = 'https://raw.githubusercontent.com/Yorko/mlcourse.ai/master/data/adult.dat
a.csv'
data = pd.read_csv(url, error_bad_lines=False)
```

In [230]:

```
data.head()
```

Out[230]:

	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 [231]:

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

Out[231]:

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

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

In [232]:

```
round(data[data['sex'] == 'Female']['age'].mean(), 0)
```

Out[232]:

```
37.0
```

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

In [233]:

```
round((data[data['native-country'] == 'Germany'].shape[0] / data.shape[0]) * 100
, 4)
```

Out[233]:

```
0.4207
```

1. (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 [234]:

```
# less than 50K per year
m = data[data['salary'] == '<=50K']['age'].mean()
s = data[data['salary'] == '<=50K']['age'].std()

print('<=50K: {} ± {}'.format(round(m, 0), round(s, 0)))
```

```
<=50K: 37.0 ± 14.0
```

In [235]:

```
# more than 50K per year
m = data[data['salary'] == '>50K']['age'].mean()
s = data[data['salary'] == '>50K']['age'].std()

print('>50K: {} ± {}'.format(round(m, 0), round(s, 0)))
```

```
>50K: 44.0 ± 11.0
```

1. 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 [236]:

```
education = ['Bachelors', 'Prof-school', 'Assoc-acdm', 'Assoc-voc', 'Masters',
'Doctorate']
data[data['salary'] == '>50K']['education']

c = 0
for h in data[data['salary'] == '>50K']['education']:
  if h in education:
    c += 1

res = (c / data[data['salary'] == '>50K'].shape[0]) * 100

print('educated percent: {}'.format(round(res, 2)))
```

```
educated percent: 57.84
```

In [237]:

```
with_salary = data[data['salary'] == '>50K']

educated = with_salary[data['education'].isin(education)]

res = (educated.shape[0] / with_salary.shape[0]) * 100

print('educated percent: {}'.format(round(res, 2)))
```

```
educated percent: 57.84
```

1. 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 [238]:

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

Out[238]:

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

1. 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 [239]:

```
statuses_married = ['Married-civ-spouse', 'Married-spouse-absent', 'Married-AF-spouse']

# maried group by salary
data[data['sex'] == 'Male'][data['marital-status'].isin(statuses_married)].groupby('salary')['salary'].describe()
```

Out[239]:

	count	unique	top	freq
salary				
<=50K	7576	1	<=50K	7576
>50K	5965	1	>50K	5965

In [240]:

```
# not maried
data[data['sex'] == 'Male'][~data['marital-status'].isin(statuses_married)].groupby('salary')['salary'].describe()
```

Out[240]:

	count	unique	top	freq
salary				
<=50K	7552	1	<=50K	7552
>50K	697	1	>50K	697

1. 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 [241]:

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

print('max hours per week: {}'.format(max_hours))
```

```
max hours per week: 99
```

In [242]:

```
workers = data[data['hours-per-week'] == max_hours]

print('workers with max hours: {}'.format(workers.shape[0]))
```

```
workers with max hours: 85
```

In [243]:

```
data[data['hours-per-week'] == max_hours].groupby('salary')['salary'].describe()
```

Out[243]:

	count	unique	top	freq
salary				
<=50K	60	1	<=50K	60
>50K	25	1	>50K	25

In [244]:

```
round(data[data['hours-per-week'] == max_hours][data['salary'] == '>50K'].shape[
0] / workers.shape[0], 2)
```

Out[244]:

```
0.29
```

1. 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 [245]:

```
data.groupby(['native-country', 'salary'])['hours-per-week'].describe().unstack
()[['mean']]
```

Out[245]:

	mean	
salary	<=50K	>50K
native-country		
?	40.164760	45.547945
Cambodia	41.416667	40.000000
Canada	37.914634	45.641026
China	37.381818	38.900000
Columbia	38.684211	50.000000
Cuba	37.985714	42.440000
Dominican-Republic	42.338235	47.000000
Ecuador	38.041667	48.750000
El-Salvador	36.030928	45.000000
England	40.483333	44.533333
France	41.058824	50.750000
Germany	39.139785	44.977273
Greece	41.809524	50.625000
Guatemala	39.360656	36.666667
Haiti	36.325000	42.750000
Holand-Netherlands	40.000000	NaN
Honduras	34.333333	60.000000
Hong	39.142857	45.000000
Hungary	31.300000	50.000000
India	38.233333	46.475000
Iran	41.440000	47.500000
Ireland	40.947368	48.000000
Italy	39.625000	45.400000
Jamaica	38.239437	41.100000
Japan	41.000000	47.958333
Laos	40.375000	40.000000
Mexico	40.003279	46.575758
Nicaragua	36.093750	37.500000
Outlying-US(Guam-USVI-etc)	41.857143	NaN
Peru	35.068966	40.000000
Philippines	38.065693	43.032787
Poland	38.166667	39.000000
Portugal	41.939394	41.500000
Puerto-Rico	38.470588	39.416667

	mean	
salary	**<=50K**	**>50K**
native-country		
Scotland	39.444444	46.666667
South	40.156250	51.437500
Taiwan	33.774194	46.800000
Thailand	42.866667	58.333333
Trinadad&Tobago	37.058824	40.000000
United-States	38.799127	45.505369
Vietnam	37.193548	39.200000
Yugoslavia	41.600000	49.500000

Часть 2

In [0]:

```
from pandasql import sqldf
pysqldf = lambda q: sqldf(q, globals())
```

In [247]:

```
url = 'https://raw.githubusercontent.com/shanealynn/Pandas-Merge-Tutorial/master/user_usage.csv'
user_usage = pd.read_csv(url, error_bad_lines=False)

user_usage.head()
```

Out[247]:

	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 [248]:

```
url = 'https://raw.githubusercontent.com/shanealynn/Pandas-Merge-Tutorial/maste
r/user_device.csv'
user_device = pd.read_csv(url, error_bad_lines=False)

user_device.head()
```

Out[248]:

	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 [249]:

```
url = 'https://raw.githubusercontent.com/shanealynn/Pandas-Merge-Tutorial/maste
r/android_devices.csv'
android_devices = pd.read_csv(url, error_bad_lines=False)

android_devices.head()
```

Out[249]:

	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

Pandas

Запрос на соединение двух наборов данных

In [250]:

```
user_device.merge(user_usage, how='inner', on='use_id')
```

Out[250]:

	use_id	user_id	platform	platform_version	device	use_type_id	outgoing_mins_per_mc
0	22787	12921	android	4.3	GT-I9505	1	2
1	22788	28714	android	6.0	SM-G930F	1	171
2	22789	28714	android	6.0	SM-G930F	1	171
3	22790	29592	android	5.1	D2303	1	9
4	22792	28217	android	5.1	SM-G361F	1	7
...	...	...	...	...	...	...	
154	23043	28953	android	6.0	SM-G900F	1	19
155	23044	28953	android	6.0	SM-G900F	1	19
156	23046	29454	android	6.0	Moto G (4)	1	10
157	23049	29725	android	6.0	SM-G900F	1	34
158	23053	20257	android	5.1	Vodafone Smart ultra 6	1	4

159 rows × 9 columns

Запрос на группировку набора данных с использованием функций агрегирования

In [251]:

```
user_device[user_device['platform'] == 'android'].groupby('device').count().reset_index()[['device', 'user_id']]
```

Out[251]:

	device	user_id
0	A0001	2
1	C6603	1
2	D2303	2
3	D5503	2
4	D5803	1
5	D6603	2
6	E6653	1
7	EVA-L09	2
8	F3111	4
9	GT-I8190N	1
10	GT-I9195	3
11	GT-I9300	3
12	GT-I9505	13
13	GT-I9506	1
14	GT-I9515	3
15	GT-N7100	2
16	HTC Desire 510	6
17	HTC Desire 530	1
18	HTC Desire 620	1
19	HTC Desire 626	2
20	HTC Desire 825	3
21	HTC One M9	1
22	HTC One S	2
23	HTC One mini 2	4
24	HTC One_M8	1
25	HUAWEI CUN-L01	1
26	HUAWEI VNS-L31	3
27	LG-H815	1
28	Lenovo K51c78	1
29	Moto G (4)	4
30	MotoE2(4G-LTE)	1
31	Nexus 5	1
32	Nexus 5X	1
33	ONE A2003	2
34	ONE E1003	3
35	ONEPLUS A3003	9

	device	user_id
36	SM-A300FU	5
37	SM-A310F	2
38	SM-A500FU	1
39	SM-G360F	2
40	SM-G361F	6
41	SM-G531F	1
42	SM-G800F	1
43	SM-G900F	32
44	SM-G903F	3
45	SM-G920F	8
46	SM-G925F	7
47	SM-G930F	3
48	SM-G935F	5
49	SM-J320FN	6
50	SM-N9005	1
51	SM-N910F	6
52	VF-795	1
53	Vodafone Smart ultra 6	1
54	X11	2

PandaSQL

In [252]:

```
!pip install pandasql
import pandasql as ps
```

```
Requirement already satisfied: pandasql in /usr/local/lib/python3.6/
dist-packages (0.7.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dis
t-packages (from pandasql) (1.18.3)
Requirement already satisfied: sqlalchemy in /usr/local/lib/python3.
6/dist-packages (from pandasql) (1.3.16)
Requirement already satisfied: pandas in /usr/local/lib/python3.6/di
st-packages (from pandasql) (1.0.3)
Requirement already satisfied: python-dateutil>=2.6.1 in /usr/local/
lib/python3.6/dist-packages (from pandas->pandasql) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python
3.6/dist-packages (from pandas->pandasql) (2018.9)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/
dist-packages (from python-dateutil>=2.6.1->pandas->pandasql) (1.12.
0)
```

Запрос на соединение двух наборов данных

In [253]:

```
query = '''
select *
from user_device ud
  join user_usage uu on ud.use_id = uu.use_id
'''

ps.sqldf(query, locals())
```

Out[253]:

	use_id	user_id	platform	platform_version	device	use_type_id	outgoing_mins_per_mo
0	22787	12921	android	4.3	GT-I9505	1	2
1	22788	28714	android	6.0	SM-G930F	1	171
2	22789	28714	android	6.0	SM-G930F	1	171
3	22790	29592	android	5.1	D2303	1	9
4	22792	28217	android	5.1	SM-G361F	1	7
...	...	...	...	...	...	...	
154	23043	28953	android	6.0	SM-G900F	1	19
155	23044	28953	android	6.0	SM-G900F	1	19
156	23046	29454	android	6.0	Moto G (4)	1	10
157	23049	29725	android	6.0	SM-G900F	1	34
158	23053	20257	android	5.1	Vodafone Smart ultra 6	1	4

159 rows × 10 columns

Запрос на группировку набора данных с использованием функций агрегирования

In [254]:

```
query = '''
select device, count(*) as user_id
from user_device ud
where ud.platform = 'android'
group by device
'''

ps.sqldf(query, locals())
```

Out[254]:

	device	user_id
0	A0001	2
1	C6603	1
2	D2303	2
3	D5503	2
4	D5803	1
5	D6603	2
6	E6653	1
7	EVA-L09	2
8	F3111	4
9	GT-I8190N	1
10	GT-I9195	3
11	GT-I9300	3
12	GT-I9505	13
13	GT-I9506	1
14	GT-I9515	3
15	GT-N7100	2
16	HTC Desire 510	6
17	HTC Desire 530	1
18	HTC Desire 620	1
19	HTC Desire 626	2
20	HTC Desire 825	3
21	HTC One M9	1
22	HTC One S	2
23	HTC One mini 2	4
24	HTC One_M8	1
25	HUAWEI CUN-L01	1
26	HUAWEI VNS-L31	3
27	LG-H815	1
28	Lenovo K51c78	1
29	Moto G (4)	4
30	MotoE2(4G-LTE)	1
31	Nexus 5	1
32	Nexus 5X	1
33	ONE A2003	2
34	ONE E1003	3
35	ONEPLUS A3003	9

	device	user_id
36	SM-A300FU	5
37	SM-A310F	2
38	SM-A500FU	1
39	SM-G360F	2
40	SM-G361F	6
41	SM-G531F	1
42	SM-G800F	1
43	SM-G900F	32
44	SM-G903F	3
45	SM-G920F	8
46	SM-G925F	7
47	SM-G930F	3
48	SM-G935F	5
49	SM-J320FN	6
50	SM-N9005	1
51	SM-N910F	6
52	VF-795	1
53	Vodafone Smart ultra 6	1
54	X11	2

Сравнение времени выполнения запросов библиотек Pandas и PandaSQL

join

In [0]:

```
query = '''
select *
from user_device ud
  join user_usage uu on ud.use_id = uu.use_id
'''
```

In [256]:

```
%%timeit
pysqldf(query)
```

```
100 loops, best of 3: 12.3 ms per loop
```

In [257]:

```
%%timeit
user_device.merge(user_usage, how='inner', on='use_id')
```

```
100 loops, best of 3: 3.22 ms per loop
```

sort + group + aggregate

In [0]:

```
query = '''
select device, count(*) as user_id
from user_device ud
where ud.platform = 'android'
group by device
'''
```

In [259]:

```
%%timeit
pysqldf(query)
```

```
100 loops, best of 3: 6.73 ms per loop
```

In [260]:

```
%%timeit
user_device[user_device['platform'] == 'android'].groupby('device').count().rese
t_index()[['device', 'user_id']]
```

```
100 loops, best of 3: 4.06 ms per loop
```

На основе полученных данных можно предположить, что функции сортировки, объединения и группировки работают быстрее в библиотеке Pandas