Московский Государственный технический университет
имени Н. Э. Баумана

Лабораторная работа №2 по курсу:
«Технология машинного обучения»

Работу выполнил студент группы ИУ5-63
Федорова Антонина____________________

Работу проверил:
Гапанюк Ю.Е.______________________

Москва
2019

Задание:
Часть 1.
Выполните первое демонстрационное задание "demo assignment" под названием "Exploratory data analysis with Pandas" со страницы курса https://mlcourse.ai/assignments
Условие задания - https://nbviewer.jupyter.org/github/Yorko/mlcourse_open/blob/master/jupyter_english/assignments_demo/assignment01_pandas_uci_adult.ipynb?flush_cache=true
Набор данных можно скачать здесь - https://archive.ics.uci.edu/ml/datasets/Adult
Пример решения задания - https://www.kaggle.com/kashnitsky/a1-demo-pandas-and-uci-adult-dataset-solution

Часть 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

Текст программы с примерами выполнения программы:

```
! pip3 install pandasql
```

```
Collecting pandasql
  Downloading https://files.pythonhosted.org/packages/6b/c4/ee4096ffa2eeec
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-pack
Requirement already satisfied: pandas in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: sqlalchemy in /usr/local/lib/python3.6/dist
Requirement already satisfied: python-dateutil>=2 in /usr/local/lib/python
Requirement already satisfied: pytz>=2011k in /usr/local/lib/python3.6/dis
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-p
Building wheels for collected packages: pandasql
  Building wheel for pandasql (setup.py) ... done
  Stored in directory: /root/.cache/pip/wheels/53/6c/18/b87a2e5fa8a82e9c02
Successfully built pandasql
Installing collected packages: pandasql
Successfully installed pandasql-0.7.3
```

Часть 1

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

```
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')
from google.colab import drive
```

```
drive.mount("/content/gdrive", force_remount=True)
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?cli

Enter your authorization code:
··········
Mounted at /content/gdrive
```

```
#Загружаю данные с гугл диска
data = pd.read_csv('/content/gdrive/My Drive/adult.data.csv')
```

```
data.head()
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	rela
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	N
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	N
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	

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()
```

```
age    36.85823
dtype: float64
```

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

```
#Всего граждан из Германии
(data['native-country'] == 'Germany').sum()
```

```
137
```

```
(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 (salary feature) and those who earn less than 50K per year?

```
poor_mean = data.loc[data['salary'] == '<=50K', 'age'].mean()
poor_std = data.loc[data['salary'] == '<=50K', 'age'].std()
rich_mean = data.loc[data['salary'] == '>50K', 'age'].mean()
rich_std = data.loc[data['salary'] == '>50K', 'age'].std()
print('Средний возраст бедных = {0} +- {1}'.format(round(poor_mean), round(poor_
print('Средний возраст богатых = {0} +- {1}'.format(round(rich_mean), round(rich
```

```
Средний возраст бедных = 37 +- 14.0
Средний возраст богатых = 44 +- 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)

```
data.loc[data['salary'] == '>50K', 'education'].value_counts()
```

```
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
```

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.

```
for (race, sex), sub_data in data.groupby(['race', 'sex']):
    print("Race: {0}, sex: {1}".format(race, sex))
    print(sub_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
```

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.

```
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
```

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?

```
max_time = data['hours-per-week'].max()
print('Максимальное число часов работы: {0}'.format(max_time))
print('Число людей, работающих {0} часов в неделю и их зарплаты:'.format(max_tim
print(data.loc[data['hours-per-week'] == max_time, 'salary'].value_counts())
perc = ((data['hours-per-week'] == max_time) & (data['salary'] == '>50K')).sum()
print('Процент людей, которые работают по  {0} часов и получают зарплату более 5
```

```
Максимальное число часов работы: 99
Число людей, работающих 99 часов в неделю и их зарплаты:
<=50K    60
>50K     25
Name: salary, dtype: int64
Процент людей, которые работают по  99 часов и получают зарплату более 50k
```

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?

```
for (salary, native_country), sub_data in data.groupby(['salary', 'native-countr
  print("native-country: {0}, salary: {1}".format(native_country, salary))
  print(round(sub_data['hours-per-week'].mean(), 2))
```

```
native-country: ?, salary: <=50K
40.16
native-country: Cambodia, salary: <=50K
41.42
native-country: Canada, salary: <=50K
37.91
native-country: China, salary: <=50K
37.38
native-country: Columbia, salary: <=50K
38.68
native-country: Cuba, salary: <=50K
37.99
native-country: Dominican-Republic, salary: <=50K
42.34
native-country: Ecuador, salary: <=50K
38.04
native-country: El-Salvador, salary: <=50K
36.03
```

```
native-country: England, salary: <=50K
40.48
native-country: France, salary: <=50K
41.06
native-country: Germany, salary: <=50K
39.14
native-country: Greece, salary: <=50K
41.81
native-country: Guatemala, salary: <=50K
39.36
native-country: Haiti, salary: <=50K
36.33
native-country: Holand-Netherlands, salary: <=50K
40.0
native-country: Honduras, salary: <=50K
34.33
native-country: Hong, salary: <=50K
39.14
native-country: Hungary, salary: <=50K
31.3
native-country: India, salary: <=50K
38.23
native-country: Iran, salary: <=50K
41.44
native-country: Ireland, salary: <=50K
40.95
native-country: Italy, salary: <=50K
39.62
native-country: Jamaica, salary: <=50K
38.24
native-country: Japan, salary: <=50K
41.0
native-country: Laos, salary: <=50K
40.38
native-country: Mexico, salary: <=50K
40.0
native-country: Nicaragua, salary: <=50K
36.09
native-country: Outlying-US(Guam-USVI-etc), salary: <=50K
41.86
```

Часть 2

```
user_usage = pd.read_csv('/content/gdrive/My Drive/user_usage.csv')
android_devices = pd.read_csv('/content/gdrive/My Drive/android_devices.csv')
user_device = pd.read_csv('/content/gdrive/My Drive/user_device.csv')
```

```
user_usage.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

```
android_devices.head()
```

	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

```
user_device.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

Pandas

```
full1 = pd.merge(user_usage,
                 user_device,
                 on='use_id',
                 how = 'left')
```

```
full1.head()
```

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id	use
0	21.97	4.82	1557.33	22787	12
1	1710.08	136.88	7267.55	22788	28
2	1710.08	136.88	7267.55	22789	28
3	94.46	35.17	519.12	22790	29
4	71.59	79.26	1557.33	22792	28

```
full1.tail()
```

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id	u
235	260.66	68.44	896.96	25008	
236	97.12	36.50	2815.00	25040	
237	355.93	12.37	6828.09	25046	
238	632.06	120.46	1453.16	25058	
239	488.70	906.92	3089.85	25220	

Можно заметить, что появляются нулевые поля так как я использовала left join. То есть в итоговую таблицу помещаются те значения, которые ест в первой таблице, но их нет во второй таблице.

```
full1.agg(
    {'outgoing_mins_per_month' : ['sum', 'min', 'mean', 'max'],
     'outgoing_sms_per_month' : ['min', 'max', 'sum', 'mean'],
     'platform_version' : ['min', 'max', 'sum', 'mean']}
)
```

	outgoing_mins_per_month	outgoing_sms_per_month	platform_version
max	1816.630000	906.920000	10.100000
mean	274.559167	98.968292	5.554717
min	0.500000	0.250000	4.100000
sum	65894.200000	23752.390000	883.200000

Вывела для трех значений максимум, среднее, минимум, сумму (с помощью функции **агрегации**)

PandaSQL

```
import pandasql as ps
```

```
def example_query(full1):
    simple_query = '''
        SELECT
            outgoing_mins_per_month,
            monthly_mb,
            platform
        FROM full1
        where 1=1
        and monthly_mb >= 100
        --and platform = 'ios'
        LIMIT 15
        '''
    return ps.sqldf(simple_query, locals())
```

```
example_query(full1)
```

	outgoing_mins_per_month	monthly_mb	platform
0	21.97	1557.33	android
1	1710.08	7267.55	android
2	1710.08	7267.55	android
3	94.46	519.12	android
4	71.59	1557.33	android
5	71.59	1557.33	android
6	71.59	519.12	android
7	71.59	519.12	android
8	30.92	3114.67	android
9	69.80	25955.55	android
10	554.41	3114.67	android
11	189.10	519.12	android
12	283.30	15573.33	android
13	324.34	519.12	android
14	797.06	519.12	android

```
def aggr_query(full1):
    aggr_query = '''
        SELECT
            count(*),
            platform
        FROM full1
       group by platform
        LIMIT 15
        '''
    return ps.sqldf(aggr_query, locals())
```

```
aggr_query(full1)
```

	count(*)	platform
0	81	None
1	157	android
2	2	ios

```
def join_query(user_usage, user_device):
  join_query = '''
  select *
  from user_usage t0
  join user_device t1
  on t0.use_id = t1.use_id'''

  return ps.sqldf(join_query)
```

```
join_query(user_usage, user_device)
```

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id	u
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	
5	71.59	79.26	1557.33	22793	
6	71.59	79.26	519.12	22794	
7	71.59	79.26	519.12	22795	
8	30.92	22.77	3114.67	22799	
9	69.80	14.70	25955.55	22801	
10	554.41	150.06	3114.67	22804	
11	189.10	24.08	519.12	22805	
12	283.30	107.47	15573.33	22806	
13	324.34	92.52	519.12	22808	
14	797.06	7.67	519.12	22813	

15	797.06	7.67	15573.33	22814
16	797.06	7.67	15573.33	22815
17	797.06	7.67	15573.33	22816
18	797.06	7.67	15573.33	22817
19	78.80	327.33	10382.21	22819
20	78.80	327.33	15573.33	22820
21	78.80	327.33	15573.33	22822
22	164.10	192.64	3114.67	22823
23	208.26	91.76	5191.12	22824
24	681.44	47.35	1271.39	22829
25	324.27	91.50	519.12	22830