

Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»



“Методы машинного обучения”

ЛАБОРАТОРНАЯ РАБОТА № 2. «Изучение библиотек обработки данных»

Студент группы ИУ5-24М

Петропавлов Д.М.

\_\_\_\_\_ Дата

\_\_\_\_\_ Подпись

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](https://nbviewer.jupyter.org/github/Yorko/mlcourse_open/blob/master/jupyter_english/assignments_demo/assignment01_pandas_uci_adult.ipynb?flush_cache=true)

Официальный датасет находится здесь, но данные и заголовки хранятся отдельно, что неудобно для анализа - <https://archive.ics.uci.edu/ml/datasets/Adult>

Поэтому готовый набор данных для лабораторной работы удобнее скачать здесь

- <https://raw.githubusercontent.com/Yorko/mlcourse.ai/master/data/adult.data.csv> (удобнее всего нажать на данной ссылке правую кнопку мыши и выбрать в контекстном меню пункт "сохранить ссылку", будет предложено сохранить файл в формате CSV)

Пример решения задания - <https://www.kaggle.com/kashnitsky/a1-demo-pandas-and-uci-adult-dataset-solution>

Объявление библиотек и настройка отображения:

In [1]:

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

Подключение предлагаемой выборки и проверка выводом первых 5ти:

In [2]:

```
data = pd.read_csv('C:/Dataset/adult.data.csv')
data.head()
```

Out[2]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

### Задача 1.

Сколько мужчин и женщин отображено в выборке

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

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

### Задача 2.

Найти средний возраст женщин

```
data_fem = data[data['sex']=='Female']
data_fem.head()
data_fem['age'].mean()
```

```
36.85823043357163
```

### Задача 3.

Каков процент жителей из Германии?

```
(data['native-country']=='Germany').sum()/data['age'].count()*100
```

```
0.42074874850281013
```

### Задачи 4-5.

Найти среднее и отклонение возраста для богатых (ЗП>50к) и бедных (ЗП<50к)

```
ages1 = data.loc[data['salary'] == '>50K', 'age']
ages2 = data.loc[data['salary'] == '<=50K', 'age']
print("rich: {0} +- {1} years, \npoor: {2} +- {3} years.".format(
    round(ages1.mean()), round(ages1.std(), 1),
    round(ages2.mean()), round(ages2.std(), 1)))
rich: 44.0 +- 10.5 years,
poor: 37.0 +- 14.0 years.
```

### Задача 6.

Проверить, является ли правдой утверждение что те, кто зарабатывает 50к+ имеют образование как минимум выше старшей школы?

In [7]:

```
data.loc[(data['salary'] == '>50K') &
(~data['education'].isin([' Bachelors', ' Prof-school', ' Assoc-acdm',
' Assoc-voc', ' Masters', ' Doctorate feature']))]
```

Out [7]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
7	52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	45	United-States	>50K
10	37	Private	280464	Some-college	10	Married-civ-spouse	Exec-managerial	Husband	Black	Male	0	0	80	United-States	>50K
20	40	Private	193524	Doctorate	16	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	0	60	United-States	>50K
27	54	?	180211	Some-college	10	Married-civ-spouse	?	Husband	Asian-Pac-Islander	Male	0	0	60	South	>50K
38	31	Private	84154	Some-college	10	Married-civ-spouse	Sales	Husband	White	Male	0	0	38	?	>50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32519	46	Private	364548	Some-college	10	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	48	United-States	>50K
32532	34	Private	204461	Doctorate	16	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	0	60	United-States	>50K
32539	71	?	287372	Doctorate	16	Married-civ-spouse	?	Husband	White	Male	0	0	10	United-States	>50K

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
32557	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-oper-inspect	Husband	White	Male	0	0	40	United-States	>50K
32560	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Executive-managerial	Wife	White	Female	15024	0	40	United-States	>50K

3612 rows × 15 columns

### Задача 7.

Отобразить возрастную статистику по каждой расе и полу. Использовать GROUPBY и DESCRIBE. Найти максимальный возраст мужчин Американско-Индийско-Эскимоской расы (?).

In [8]:

```
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

```

### Задача 8.

Найти кого больше среди тех кто получает 50к+: женатых или одиноких мужчин

In [9]:

```

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

```

Out[9]:

```

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

```

Такое же сравнение но среди тех кто имеет в статусе "Семейное положение" статус начинающийся с 'married'

In [10]:

```

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

```

Out[10]:

```

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

```

Общая статистика по "Семейному положению"

In [11]:

```

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

```

Out[11]:

```
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
```

### Задание 9.

Чему равно максимальное число отработанных в неделю часов? Сколько человек так живет и сколько из них получает много?

In [12]:

```
maxh = data['hours-per-week'].max()
num = data[data['hours-per-week'] == maxh].shape[0]
pers = float(data[(data['hours-per-week'] == maxh)
                  & (data['salary'] == '>50K')].shape[0]) / num
print("There are {a} men with {b} hpw and there is a percentage of rich among them {c}%"
      .format(a=num,b=maxh,c=int(100 * pers)))
There are 85 men with 99 hpw and there is a percentage of rich among them 29%
```

### Задание 10.

Найти среднее рабочее время относительно получающих больше/меньше 50к по каждой стране

In [13]:

```
for (country, salary), sub_df in data.groupby(['native-country', 'salary']):
    print(country, salary, round(sub_df['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
Trinidad&Tobago	<=50K	37.06
Trinidad&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. 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 Сравните время выполнения каждого запроса в Pandas и PandaSQL.

В качестве примеров можно использовать следующие статьи:

<https://www.shanelynn.ie/summarising-aggregation-and-grouping-data-in-python-pandas/> <https://www.shanelynn.ie/merge-join-dataframes-python-pandas-index-1/> (в разделе "Example data" данной статьи содержится рекомендуемый набор данных для проведения экспериментов). Пример сравнения Pandas и PandaSQL  
- [https://github.com/miptgirl/udacity\\_engagement\\_analysis/blob/master/pandasql\\_example.ipynb](https://github.com/miptgirl/udacity_engagement_analysis/blob/master/pandasql_example.ipynb)

Набор упражнений по Pandas с решениями - [https://github.com/guipsamora/pandas\\_exercises](https://github.com/guipsamora/pandas_exercises)  
Подключаем библиотеки:

In [1]:

```
import numpy as np
import pandas as pd
import pandasql as ps
from timeit import default_timer as timer
Датафреймы для двух выборок:
```

In [2]:

```
fd = pd.read_csv('C:/Dataset/user_device.csv')
sd = pd.read_csv('C:/Dataset/user_usage.csv')
Часть выборки User_device:
```

In [3]:

```
fd.head()
```

Out[3]:

	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

Часть выборки User\_usege:

In [4]:

```
sd.head()
```

Out[4]:

	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

Как видно выше, имеется общее поле use\_id. Реализуем соединение двух таблиц по этому полю с использованием PandaSQL:

In [5]:

```
def example1_pandasql(fd,sd):

    simple_query = '''
        SELECT *
        FROM fd JOIN sd
        WHERE fd.use_id==sd.use_id
    '''

    return ps.sqldf(simple_query, locals())
```

In [6]:

```
example1_pandasql(fd,sd)
```

Out[6]:

	use_id	user_id	platform	platform_version	device	use_type_id	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id
0	22787	12921	android	4.3	GT-I9505	1	21.97	4.82	1557.33	22787
1	22788	28714	android	6.0	SM-G930F	1	1710.08	136.88	7267.55	22788
2	22789	28714	android	6.0	SM-G930F	1	1710.08	136.88	7267.55	22789
3	22790	29592	android	5.1	D2303	1	94.46	35.17	519.12	22790
4	22792	28217	android	5.1	SM-G361F	1	71.59	79.26	1557.33	22792
...	...	...	...	...	...	...	...	...	...	...
154	23043	28953	android	6.0	SM-G900F	1	198.59	90.49	5191.12	23043
155	23044	28953	android	6.0	SM-G900F	1	198.59	90.49	3114.67	23044
156	23046	29454	android	6.0	Moto G (4)	1	106.65	82.13	5191.12	23046
157	23049	29725	android	6.0	SM-G900F	1	344.53	20.53	519.12	23049
158	23053	20257	android	5.1	Vodafone Smart ultra 6	1	42.75	46.83	5191.12	23053

159 rows × 10 columns  
И Pandas. Замерим время выполнения каждого:

In [7]:

```
t1 = timer()
example1_pandasql(fd,sd)
elapsed1 = timer() - t1
t2 = timer()
```

```
fsd = fd.merge(sd)
elapsed2 = timer() - t2
print("PandasSQL: {a} \nPandas: {b} ".format(a=elapsed1,b=elapsed2))
PandasSQL: 0.0464314
Pandas: 0.013121500000000001
```

Для полученной соединенной таблице попробуем провести агрегирование с группировкой и так же замерим время выполнения: PandaSQL:

In [8]:

```
def example2_pandasql(sd):
    aggr_query = """
        SELECT distinct
            device, avg(monthly_mb) as avg_mb
        FROM sd
        GROUP BY device
    """
    return ps.sqldf(aggr_query, locals())
```

In [9]:

```
example2_pandasql(fsd)
```

Out[9]:

	device	avg_mb
0	A0001	15573.330000
1	C6603	1557.330000
2	D2303	519.120000
3	D5503	1557.330000
4	D5803	1557.330000
5	D6603	7267.550000
6	E6653	5191.120000
7	EVA-L09	1557.330000
8	F3111	2076.450000
9	GT-I8190N	407.010000
10	GT-I9195	1211.260000
11	GT-I9300	464.185000
12	GT-I9505	5564.726364
13	GT-I9506	803.240000
14	GT-I9515	1557.330000
15	GT-N7100	11939.560000
16	HTC Desire 510	12562.488000
17	HTC Desire 530	1557.330000

	device	avg_mb
18	HTC Desire 620	74.400000
19	HTC Desire 626	519.120000
20	HTC Desire 825	5498.970000
21	HTC One M9	2362.070000
22	HTC One S	1038.210000
23	HTC One mini 2	13842.956667
24	HTC One_M8	6577.120000
25	HUAWEI CUN-L01	11.680000
26	HUAWEI VNS-L31	3114.670000
27	LG-H815	1557.330000
28	Lenovo K51c78	1557.330000
29	Moto G (4)	5191.120000
30	MotoE2(4G-LTE)	212.640000
31	Nexus 5X	1557.330000
32	ONE A2003	2076.450000
33	ONEPLUS A3003	3823.610000
34	SM-A300FU	1687.112500
35	SM-A310F	1557.330000
36	SM-A500FU	1557.330000
37	SM-G360F	1557.330000
38	SM-G361F	934.404000
39	SM-G531F	2076.450000
40	SM-G800F	1557.330000
41	SM-G900F	3841.427333
42	SM-G903F	1557.330000
43	SM-G920F	1985.168000
44	SM-G925F	3633.775000
45	SM-G930F	7959.700000
46	SM-G935F	4568.182000
47	SM-J320FN	830.574000

	device	avg_mb
48	SM-N9005	16611.550000
49	SM-N910F	8038.370000
50	VF-795	1557.330000
51	Vodafone Smart ultra 6	5191.120000
52	X11	12458.670000
53	iPhone6,2	650.920000
54	iPhone7,2	1271.390000

Аналогичная функция на Pandas и подсчет времени выполнения.

In [10]:

```
t1 = timer()
example2_pandasql(fsd)
elapsed1 = timer() - t1
t2 = timer()
fsd.groupby('device').monthly_mb.mean()
elapsed2 = timer() - t2
print("PandasSQL: {a} \nPandas: {b} ".format(a=elapsed1,b=elapsed2))
PandasSQL: 0.0259188999999998857
Pandas: 0.009907099999999502
```

Как можно видеть в обоих случаях Pandas превосходит PandaSQL почти на порядок. Несмотря на то что цифры в данном примере выглядят незначительными, для огромных датасетов параметр времени может стать значительным. Так же видно еще одно преимущество Pandas - размеры кода. То что реализовано на PandaSQL полноценной функцией-запросом пишется на Pandas в одну строчку. Таким образом за исключением совсем сложных SQL-запросов Pandas будет и быстрее, и короче.