Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

МЕТОДЫ МАШИННОГО ОБУЧЕНИЯ

Отчёт по лабораторной работе № 2

«Изучение библиотек обработки данных»

Выполнила:
студентка группы ИУ5 – 23М Галичий Д. А.

Преподаватель: Гапанюк Ю. Е.

2020г.

Демонстрационное задание "demo assignment"

под названием "Exploratory data analysis with Pandas" со страницы курса https://mlcourse.ai/assignments (https://mlcourse.ai/assignments)

Adult dataset contains following unique values:

- 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

Импорт библиотек

In [7]:

```
import numpy as np
import pandasql as ps
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
# to avoid warnings
import warnings
warnings.filterwarnings('ignore')
from time import time
import timeit
```

Загрузка данных

In [9]:

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

Out[9]:

	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 [39]:

```
male_count = sum(data['sex'].isin(['Male']))
female_count = sum(data['sex'] == 'Female')

print('Всего мужчин: {}'.format(male_count))
print('Всего женщин: {}'.format(female_count))
```

```
Всего мужчин: 21790
Всего женщин: 10771
```

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

In [51]:

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

```
Средний возраст женщин: 36.85823043357163
```

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

In [70]:

```
grp = data.groupby('native-country')
len(grp.get_group('Germany')) / len(data)
```

Out[70]:

```
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?

In [89]:

```
grp = data.groupby('salary')

avg1 = (grp.get_group('>50K')).age.mean().round(3)
std1 = (grp.get_group('>50K')).age.std().round(3)

avg2 = (grp.get_group('<=50K')).age.mean().round(3)
std2 = (grp.get_group('<=50K')).age.std().round(3)

print('>50K - the mean of age: {}, \n standard deviation: {} \n'.format(avg1, std1))
print('<=50K - the mean of age: {}, \n standard deviation: {}'.format(avg2, std2))
```

```
>50K - the mean of age: 44.25,
 standard deviation: 10.519

<=50K - the mean of age: 36.784,
 standard deviation: 14.02
```

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 [119]:

```
grp = data.groupby('salary')
grp1 = grp.get_group('>50K')

count1 = len(grp1)

count2 = sum(grp1['education'].isin(['Bachelors', 'Prof-school', 'Assoc-acdm', 'Assoc-voc', 'Masters', 'Doctorate']))

if (count1 == count2):
    print ('It is true')
else:
    print ('It is false')
```

```
It is false
```

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 [136]:

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

Out[136]:

		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

In [142]:

```
grp.get_group(('Amer-Indian-Eskimo', 'Male')).age.max()
```

Out[142]:

82

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 [155]:

```
grp = data.groupby(['sex', 'salary'])
grp1 = grp.get_group(('Male', '>50K'))
count_married = sum(grp1['marital-status'].isin(['Married-civ-spouse', 'Married-spouse-absent', 'Married-AF-spouse']))
count_bachelors = sum(grp1['marital-status'].isin(['Divorced', 'Never-married', 'Separated', 'Widowed']))

print(count_married, count_bachelors)
if (count_married > count_bachelors):
    print('The proportion of those who earn a lot (>50K) is greater among married men.')
elif (count_married < count_bachelors):
    print('The proportion of those who earn a lot (>50K) is greater among single men.')
else:
    print('The proportion of those who earn a lot (>50K) is equal  between married and single men.')
```

```
5965 697
The proportion of those who earn a lot (>50K) is greater among married me
n.
```

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 [157]:

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

Out[157]:

```
99
```

In [158]:

```
people_count = sum(data['hours-per-week'] == 99)
print(people_count)
```

```
85
```

In [160]:

```
percentage = sum((data['hours-per-week'] == 99) & (data['salary'] == '>50K')) * 100 / 85
print(percentage)
```

```
29.41176470588235
```

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 [168]:

```
grp = data.groupby(['native-country', 'salary'])
grp['hours-per-week'].mean().round(3)
```

Out[168]:

```
native-country  salary
?               <=50K     40.165
                >50K      45.548
Cambodia        <=50K     41.417
                >50K      40.000
Canada          <=50K     37.915
                           ...
United-States   >50K      45.505
Vietnam         <=50K     37.194
                >50K      39.200
Yugoslavia      <=50K     41.600
                >50K      49.500
Name: hours-per-week, Length: 82, dtype: float64
```

In [174]:

```
for (country, salary), avg in data.groupby(['native-country', 'salary']):
    print(country, salary, round(avg['hours-per-week'].mean(), 3))
```

? <=50K 40.165
? >50K 45.548
Cambodia <=50K 41.417
Cambodia >50K 40.0
Canada <=50K 37.915
Canada >50K 45.641
China <=50K 37.382
China >50K 38.9
Columbia <=50K 38.684
Columbia >50K 50.0
Cuba <=50K 37.986
Cuba >50K 42.44
Dominican-Republic <=50K 42.338
Dominican-Republic >50K 47.0
Ecuador <=50K 38.042
Ecuador >50K 48.75
El-Salvador <=50K 36.031
El-Salvador >50K 45.0
England <=50K 40.483
England >50K 44.533
France <=50K 41.059
France >50K 50.75
Germany <=50K 39.14
Germany >50K 44.977
Greece <=50K 41.81
Greece >50K 50.625
Guatemala <=50K 39.361
Guatemala >50K 36.667
Haiti <=50K 36.325
Haiti >50K 42.75
Holand-Netherlands <=50K 40.0
Honduras <=50K 34.333
Honduras >50K 60.0
Hong <=50K 39.143
Hong >50K 45.0
Hungary <=50K 31.3
Hungary >50K 50.0
India <=50K 38.233
India >50K 46.475
Iran <=50K 41.44
Iran >50K 47.5
Ireland <=50K 40.947
Ireland >50K 48.0
Italy <=50K 39.625
Italy >50K 45.4
Jamaica <=50K 38.239
Jamaica >50K 41.1
Japan <=50K 41.0
Japan >50K 47.958
Laos <=50K 40.375
Laos >50K 40.0
Mexico <=50K 40.003
Mexico >50K 46.576
Nicaragua <=50K 36.094
Nicaragua >50K 37.5
Outlying-US(Guam-USVI-etc) <=50K 41.857
Peru <=50K 35.069
Peru >50K 40.0
Philippines <=50K 38.066
Philippines >50K 43.033
Poland <=50K 38.167

```
Poland >50K 39.0
Portugal <=50K 41.939
Portugal >50K 41.5
Puerto-Rico <=50K 38.471
Puerto-Rico >50K 39.417
Scotland <=50K 39.444
Scotland >50K 46.667
South <=50K 40.156
South >50K 51.438
Taiwan <=50K 33.774
Taiwan >50K 46.8
Thailand <=50K 42.867
Thailand >50K 58.333
Trinadad&Tobago <=50K 37.059
Trinadad&Tobago >50K 40.0
United-States <=50K 38.799
United-States >50K 45.505
Vietnam <=50K 37.194
Vietnam >50K 39.2
Yugoslavia <=50K 41.6
Yugoslavia >50K 49.5
```

In [175]:

```
grp1 = grp.get_group(('Japan', '>50K'))
grp2 = grp.get_group(('Japan', '<=50K'))
avg1 = grp1['hours-per-week'].mean().round(3)
avg2 = grp2['hours-per-week'].mean().round(3)
print('Japan - >50K - hours-per-week: {}'.format(avg1))
print('Japan - <=50K - hours-per-week: {}'.format(avg2))
```

```
Japan - >50K - hours-per-week: 47.958
Japan - <=50K - hours-per-week: 41.0
```

Запросы с использованием двух различных библиотек - Pandas и PandaSQL

In [31]:

```
user_usage = pd.read_csv('data/user_usage.csv')
user_device = pd.read_csv('data/user_device.csv')
devices = pd.read_csv('data/android_devices.csv')
```

In [32]:

```
user_usage.head()
```

Out[32]:

	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 [66]:

```
user_device.head(20)
```

Out[66]:

	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
5	22787	12921	android	4.3	GT-I9505	1
6	22788	28714	android	6.0	SM-G930F	1
7	22789	28714	android	6.0	SM-G930F	1
8	22790	29592	android	5.1	D2303	1
9	22791	28775	ios	10.2	iPhone6,2	3
10	22792	28217	android	5.1	SM-G361F	1
11	22793	28217	android	5.1	SM-G361F	1
12	22794	28217	android	5.1	SM-G361F	1
13	22795	28217	android	5.1	SM-G361F	1
14	22796	29641	ios	10.2	iPhone7,2	3
15	22797	29642	ios	10.1	iPhone5,2	3
16	22798	29642	ios	10.1	iPhone5,2	3
17	22799	29643	android	6.0	ONEPLUS A3003	1
18	22800	14785	android	4.4	SM-A300FU	1
19	22801	10976	android	4.4	GT-I9505	1

In [34]:

```
devices.head(10)
```

Out[34]:

	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
5	7Eleven	IN265	IN265	IN265
6	A.O.I. ELECTRONICS FACTORY	A.O.I.	TR10CS1_11	TR10CS1
7	AG Mobile	AG BOOST 2	BOOST2	E4010
8	AG Mobile	AG Flair	AG_Flair	Flair
9	AG Mobile	AG Go Tab Access 2	AG_Go_Tab_Access_2	AG_Go_Tab_Access_2

Функции соединения двух наборов данных

In [45]:

```
# pandasql code
def merge_pandasql():
    user_usage = pd.read_csv('data/user_usage.csv')
    user_device = pd.read_csv('data/user_device.csv')
    join_query = '''
        SELECT
        uu.use_id,
        uu.outgoing_mins_per_month,
        uu.outgoing_sms_per_month,
        uu.monthly_mb,
        ud.platform,
        ud.device
        FROM user_usage AS uu JOIN user_device AS ud ON (uu.use_id = ud.use_id)
        '''
    return ps.sqldf(join_query, locals())
```

In [37]:

```
# pandas code
def merge_pandas():
    result = pd.merge(user_usage,
                      user_device[['use_id', 'platform', 'device']],
                      on='use_id')
    return result
```

Результаты выполнения функций

In [38]:

```
merge_pandas()
```

Out[38]:

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id	platform	c
0	21.97	4.82	1557.33	22787	android	GT
1	1710.08	136.88	7267.55	22788	android	(
2	1710.08	136.88	7267.55	22789	android	(
3	94.46	35.17	519.12	22790	android	I
4	71.59	79.26	1557.33	22792	android	(
...	...	...	...	...	...	
154	198.59	90.49	5191.12	23043	android	(
155	198.59	90.49	3114.67	23044	android	(
156	106.65	82.13	5191.12	23046	android	M
157	344.53	20.53	519.12	23049	android	(
158	42.75	46.83	5191.12	23053	android	Voc

159 rows × 6 columns

In [46]:

```
merge_pandasql()
```

Out[46]:

	use_id	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	platform	c
0	22787	21.97	4.82	1557.33	android	GT
1	22788	1710.08	136.88	7267.55	android	(
2	22789	1710.08	136.88	7267.55	android	(
3	22790	94.46	35.17	519.12	android	I
4	22792	71.59	79.26	1557.33	android	(
...	...	...	...	...	...	
154	23043	198.59	90.49	5191.12	android	(
155	23044	198.59	90.49	3114.67	android	(
156	23046	106.65	82.13	5191.12	android	M
157	23049	344.53	20.53	519.12	android	(
158	23053	42.75	46.83	5191.12	android	Voc

159 rows × 6 columns

Подсчёт времени выполнения функций

In [64]:

```
print(timeit.timeit("merge_pandas()", setup="from __main__ import merge_pandas", number
=1))
```

```
0.006954299999989644
```

In [60]:

```
print(timeit.timeit("merge_pandasql()", setup="from __main__ import merge_pandasql", nu
mber=1))
```

```
0.03120169999999689
```

Как видно из результатов подсчёта времени выполнения функций, pandasql работает медленнее, чем pandas.

Группировка набора данных с использованием функции агрегирования

In [103]:

```
start = time()
grp = data.groupby('occupation')['hours-per-week'].mean()
stop = time()
grp
```

Out[103]:

```
occupation
?                   31.906131
Adm-clerical        37.558355
Armed-Forces        40.666667
Craft-repair        42.304221
Exec-managerial     44.987703
Farming-fishing     46.989940
Handlers-cleaners   37.947445
Machine-op-inspct   40.755744
Other-service       34.701669
Priv-house-serv     32.885906
Prof-specialty      42.386715
Protective-serv     42.870570
Sales               40.781096
Tech-support        39.432112
Transport-moving    44.656230
Name: hours-per-week, dtype: float64
```

In [121]:

```
print(stop - start)
```

```
0.0059986114501953125
```

In [10]:

```
query = '''
    SELECT
        data.occupation,
        avg(data.'hours-per-week')
    FROM data
    GROUP BY data.occupation
    '''
sta = time()
res = ps.sqldf(query, locals())
sto = time()
res
```

Out[10]:

	occupation	avg(data.'hours-per-week')
0	?	31.906131
1	Adm-clerical	37.558355
2	Armed-Forces	40.666667
3	Craft-repair	42.304221
4	Exec-managerial	44.987703
5	Farming-fishing	46.989940
6	Handlers-cleaners	37.947445
7	Machine-op-inspct	40.755744
8	Other-service	34.701669
9	Priv-house-serv	32.885906
10	Prof-specialty	42.386715
11	Protective-serv	42.870570
12	Sales	40.781096
13	Tech-support	39.432112
14	Transport-moving	44.656230

In [11]:

```
print(sto - sta)
```

```
0.6427726745605469
```

Как видно из результатов подсчёта времени выполнения функций, pandasql работает медленнее, чем pandas.