Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»

Отчёт

“Методы машинного обучения”

Лабораторная работа № 2

“Изучение библиотек обработки данных”

ИСПОЛНИТЕЛЬ:

Студент группы ИУ5-21М

Гузилов А.В. _____

ПРЕПОДАВАТЕЛЬ:

Гапанюк Ю.Е._____

Москва – 2019

Цель работы

Изучить библиотеки обработки данных Pandas и PandaSQL.

Задание

1. Требуется выполнить первое демонстрационное задание под названием «Exploratory data analysis with Pandas» со страницы курса mlcourse.ai.
2. Требуетсявыполнитьследующиезапросысиспользованиемдвухразличн ыхбиблиотек — Pandas и PandaSQL:
   - один произвольный запрос на соединение двух наборов данных,
   - один произвольный запрос на группировку набора данных с использованием функций агрегирования. Также требуется сравнить время выполнения каждого запроса в Pandas и PandaSQL.

Ход выполнения работы

Часть 1

In [16]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
data = pd.read_csv('adult.data.csv')
data.head()
```

Out[16]:

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

In [17]:

```
data.dtypes
```

Out[17]:

```
age                int64
workclass         object
fnlwgt             int64
education         object
education-num      int64
marital-status    object
occupation        object
relationship      object
race              object
sex               object
capital-gain       int64
capital-loss       int64
hours-per-week     int64
native-country    object
salary            object
dtype: object
```

In [18]:

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

Out[18]:

```
Male      21790
Female    10771 Name:
```

sex, dtype: int64 In

[19]:

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

Out[19]:

36.85823043357163

In [20]: float((data['native-country'] == 'Germany').sum()) /

```
data.shape[0]
```

Out[20]:

0.004207487485028101

In [21]:

```
ages1 = data.loc[data['salary'] == '>50K', 'age'] ages2
= data.loc[data['salary'] == '<=50K', 'age']
print("The average age of the rich: {0} +- {1} years, poor - {2} +- {3} years.".format(
    round(ages1.mean()), round(ages1.std(), 1),
 round(ages2.mean()), round(ages2.std(), 1)))
```

```
The average age of the rich: 44.0 +- 10.5 years, poor - 37.0 +- 14.0 year
s.
```

In [22]: data.loc[data['salary'] == '>50K',

```
'education'].unique() # No
```

Out[22]:

```
array(['HS-grad', 'Masters', 'Bachelors', 'Some-college', 'Assoc-voc',
       'Doctorate', 'Prof-school', 'Assoc-acdm', '7th-8th', '12th',
```

```
       '10th', '11th', '9th', '5th-6th', '1st-4th'], dtype=object)
```

In [23]:

```
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
```

In [24]:

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

Out[24]:

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

In [25]:

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

Out[25]:

```
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
```

In [26]:

```
max_load = data['hours-per-week'].max() print("Max
time - {0} hours./week.".format(max_load))

num_workaholics = data[data['hours-per-week'] == max_load].shape[0] print("Total
number of such hard workers {0}".format(num_workaholics))

rich_share = float(data[(data['hours-per-week'] == max_load)
                 & (data['salary'] == '>50K')].shape[0]) / num_workaholics
print("Percentage of rich among them {0}%".format(int(100 * rich_share)))
```

```
Max time - 99 hours./week.
Total number of such hard workers 85
Percentage of rich among them 29%
```

In [27]:

```
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
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

In [28]:

```
pd.crosstab(data['native-country'], data['salary'],
            values=data['hours-per-week'], aggfunc=np.mean).T
```

Out[28]:

native-country	?	Cambodia	Canada	China	Columbia	Cuba	Dominican-Republic
salary							
<=50K	40.164760	41.416667	37.914634	37.381818	38.684211	37.985714	42.338235
>50K	45.547945	40.000000	45.641026	38.900000	50.000000	42.440000	47.000000

2 rows × 42 columns

Часть 2

In [127]:

```
import pandas as pd
import pandasql as ps


data = pd.read_csv('user_device.csv', sep=",")
data1 = pd.read_csv('user_usage.csv', sep=",")
data.head()
```

Out[127]:

	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 [128]:

```
data1.head()
```

Out[128]:

	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 [129]:

```
result = pd.merge(data1,
                  data[['use_id', 'platform', 'device']],
                  on='use_id')
result.head()
```

Out[129]:

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id	platform	dev
0	21.97	4.82	1557.33	22787	android	G I95

1	1710.08	136.88	7267.55	22788	android	S G93
2	1710.08	136.88	7267.55	22789	android	S G93
3	94.46	35.17	519.12	22790	android	D23
4	71.59	79.26	1557.33	22792	android	S G36

In [130]:

```
%timeit result.head()
```

401 µs ± 40.3 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)

In [131]:

```
join_query="""SELECT w.outgoing_mins_per_month, w.outgoing_sms_per_month, w.monthly_mb,
t.use_id, t.platform, t.device
FROM data1 AS w JOIN data AS t
ON w.use_id=t.use_id"""
result1 = ps.sqldf(join_query)
result1.head()
```

Out[131]:

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id	platform	d
0	21.97	4.82	1557.33	22787	android	
1	1710.08	136.88	7267.55	22788	android	G
2	1710.08	136.88	7267.55	22789	android	G
3	94.46	35.17	519.12	22790	android	D
4	71.59	79.26	1557.33	22792	android	G

In [132]:

```
%timeit result.head()
```

335 µs ± 16.2 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)

In [133]:

```
data.groupby(['platform',
'platform_version']).count().head(20)
```

Out[133]:

	use_id	user_id	device	use_type_id

platform	platform_version				
android	4.1	5	5	5	5
	4.2	1	1	1	1
	4.3	4	4	4	4
	4.4	21	21	21	21
	5.0	18	18	18	18
	5.1	27	27	27	27
	6.0	105	105	105	105
	7.0	2	2	2	2
	7.1	1	1	1	1 ios
7.1					
2	2	2	2		
	9.0	1	1	1	1
	9.2	1	1	1	1
	9.3	20	20	20	20
	10.0	4	4	4	4
	10.1	19	19	19	19
	10.2	41	41	41	41

In [134]:

```
%timeit data.groupby(['platform', 'platform_version']).count().head(20)
```

4.52 ms ± 142 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)

In [135]:

```
simple_query="""SELECT platform, platform_version, COUNT (*) as count
FROM data

GROUP BY platform, platform_version"""
result2 = ps.sqldf(simple_query)
result2.head(20)
```

Out[135]:

	platform	platform_version	count
0	android	4.1	5
1	android	4.2	1
2	android	4.3	4
3	android	4.4	21

0	android	5.0	18
1	android	5.1	27
2	android	6.0	105
3	android	7.0	2
4	android	7.1	1
5	ios	7.1	2
10	ios	9.0	1
11	ios	9.2	1
12	ios	9.3	20
13	ios	10.0	4
14	ios	10.1	19
15	ios	10.2	41

In [136]:

```
%timeit result2.head()
```

223 µs ± 10.2 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each) In

[ ]:

```

```

In [ ]: