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

Кафедра «Автоматизированные системы обработки информации и управления»

«Методы машинного обучения»

Отчет по Лабораторной работе №2

Изучение библиотек обработки данных

Выполнила:
студент группы ИУ5-22М
Петрова Ирина
Проверил: доцент, к.т.н.
Гапанюк Ю. Е.

Москва, 2020

Лабораторная работа №2. Изучение библиотек обработки данных.

Цель лабораторной работы: изучение библиотек обработки данных Pandas и PandaSQL.

Требования к отчету: отчет по лабораторной работе должен содержать:

- титульный лист; описание задания; текст программы;
- экранные формы с примерами выполнения программы.
-
-

Задание:

Часть 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)

Условие задания
https://nbviewer.jupyter.org/github/Yorko/mlcourse_open/blob/master/jupyter_english/assignments_demo/assig flush_cache=true
(https://nbviewer.jupyter.org/github/Yorko/mlcourse_open/blob/master/jupyter_english/assignments_demo/assig flush_cache=true)

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.

Часть 1

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-spouseabsent, Married-AF-spouse.
- occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlerscleaners, 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(GuamUSVI-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 [5]:

```
import numpy as np
import pandas as pd
pd.set_option('display.max.columns', 100)
import seaborn as sns
%matplotlib inline
import matplotlib.pyplot as plt
sns.set(style="ticks")
import warnings
warnings.filterwarnings('ignore')
```

In [6]:

```
data = pd.read_csv('D:/Загрузки/adult.data', header=None, names=['age','workclass','fnlwgt','education',
                                                                   'education-num','marital-status','occupation',
                                                                   'relationship','race','sex','capital-gain','capital-loss',
                                                                   'hours-per-week','native-country','salary'])
data.head()
```

Out[6]:

	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 [29]:

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

Out[29]:

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

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

In [41]:

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

Out[41]:

```
36.85823043357163
```

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

In [54]:

```
data.loc[data['native-country'] == ' Germany', 'native-country'].value_counts()/data['n
ative-country'].count()*100
```

Out[54]:

```
 Germany    0.420749
Name: native-country, dtype: float64
```

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 [68]:

```
# mean and standard deviation of age for those who earn more than 50K
print("The average age is: {0} +- {1} years".format(
    round(data.loc[data['salary'] == ' >50K', 'age'].mean()),
 round(data.loc[data['salary'] == ' >50K', 'age'].std(), 1)))
```

```
The average age is: 44.0 +- 10.5 years
```

In [71]:

```
# mean and standard deviation of age for those who earn less than 50K
print("The average age is: {0} +- {1} years".format(
    round(data.loc[data['salary'] == ' <=50K', 'age'].mean()),
 round(data.loc[data['salary'] == ' <=50K', 'age'].std())))
```

```
The average age is: 37.0 +- 14.0 years
```

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 [81]:

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

Out[81]:

```
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 [83]:
```
print("Not all the people who earn more than 50K have at least high school education")
```

```
Not all the people who earn more than 50K have at least high school educat
ion
```

7. Display age statistics for each race (race feature) and each gender (sex feature). Use groupby() anddescribe(). Find the maximum age of men of Amer-Indian-Eskimo race.

In [90]:

```
data1 = data.groupby(['race', 'sex'])
data1['age'].describe()
```

Out[90]:

race	sex	count	mean	std	min	25%	50%	75%	max
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 [92]:
```
data.loc[data['race'] == ' Amer-Indian-Eskimo',
'age'].max()
```

Out[92]:

```
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 [120]:

```
data1 = data[data['sex'] == ' Male']
```

In [121]:

```
data2 = data1[data1['salary'] == ' >50K'].groupby('marital-status').count().reset_index
()[['marital-status', 'salary']]
data2
```

Out[121]:

	marital-status	salary
0	Divorced	284
1	Married-AF-spouse	4

2	Married-civ-spouse	5938
3	Married-spouse-absent	23
4	Never-married	325
5	Separated	49
6	Widowed	39

In [123]:

```
married_prop = data2[data2['marital-status'].str.startswith('
Married')].sum()[1] married_prop/data['marital-status'].count()*100 Out[123]:
```

18.319461932987316

In [124]:

```
single_prop = data2[~data2['marital-status'].str.startswith('
Married')].sum()[1] single_prop/data['marital-status'].count()*100 Out[124]:
```

2.140597647492399

In [125]:

```
if married_prop > single_prop:
    print('The proportion of those who earn a lot (>50K) is greater among married men')
else:     print('The proportion of those who earn a lot (>50K) is greater among single
 men')
```

The proportion of those who earn a lot (>50K) is greater among married men

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 [129]:

```
max_hours = data['hours-per-week'].max() print("The maximum number of hours
a person works per week is", max_hours)
```

The maximum number of hours a person works per week is 99

In [135]:

```
data1 = data.loc[data['hours-per-week'] == max_hours, 'salary'].count() print(data1,
"people work such a number of hours")
```

85 people work such a number of hours

In [152]:

```
percent = float(data[(data['hours-per-week'] == max_hours) & (data['salary'] == ' >50K'
)].shape[0])/data1*100 print("The
percentage is", round(percent))
```

The percentage is 29.0

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 [16]:

```
data3 = data.groupby(['native-country', 'salary']) pd.crosstab(data3['hours-per-
week'].describe().reset_index()[['native-country','salary' ,'mean']]).T
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las t)
<ipython-input-16-0b282dafc892> in <module>
      1 data3 = data.groupby(['native-country', 'salary'])
----> 2 pd.crosstab(data3['hours-per-week'].describe().reset_index()[['nat
ivecountry','salary','mean']]).T

TypeError: crosstab() missing 1 required positional argument: 'columns'
```

In [18]:

```
pd.crosstab(data['native-country'], data['salary'],          values=data['hours-per-
week'], aggfunc=np.mean).T Out[18]:
```

native-country	?	Cambodia	Canada	China	Columbia	Cuba	Ecu country	Dominican-Republic
salary								
<=50K	40.164760	41.416667	37.914634	37.381818	38.684211	37.985714	42.338235	38.04
>50K	45.547945	40.000000	45.641026	38.900000	50.000000	42.440000	47.000000	48.75

In [43]:

```
pd.crosstab(data.loc[data['native-country'] == ' Japan', 'native-country'], data['salar
y'],          values=data['hours-per-week'], aggfunc=np.mean).T Out[43]:
```

native-country	Japan
salary	
<=50K	41.000000
>50K	47.958333

Часть 2

In [45]:

Out[45]:

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

In [47]:

```
android_devices = pd.read_csv('D:/Загрузки/Pandas-Merge-Tutorial-master/android_device
s.csv')
android_devices.head()
```

```
user_device = pd.read_csv('D:/Загрузки/Pandas-Merge-Tutorial-master/user_device.csv')
user_device.head()
```

Out[47]:

	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 [49]:

```
user_usage = pd.read_csv('D:/Загрузки/Pandas-Merge-Tutorial-master/user_usage.csv')
user_usage.head()
```

Out[49]:

	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

Использование Pandas

Запрос на соединение двух наборов данных

In [52]:

```
join_pd = pd.merge(user_usage,
                   user_device[['use_id', 'platform', 'device']],
                   on='use_id')
join_pd.head()
```

Out[52]:

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id	platform	device
0	21.97	4.82	1557.33	22787	android	GT I9505
1	1710.08	136.88	7267.55	22788	android	SM G930F
2	1710.08	136.88	7267.55	22789	android	SM G930F
3	94.46	35.17	519.12	22790	android	D2303
4	71.59	79.26	1557.33	22792	android	SM G361F

Запрос на группировку набора данных с использованием функций агрегирования In [59]:

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

Out[59]:

	platform	device
0	android	184
1	ios	88

Использование PandaSQL

In [61]:

```
import pandasql as ps

ps.sqldf('select * from user_device limit 5', locals())
```

Out[61]:

	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

Запрос на соединение двух наборов данных

In [64]:

In [66]:

```
group_ps = ps.sqldf('select platform, count(device) from user_device group by platform'
, locals())
group_ps
```

Out[66]:
join_ps = ps.sqldf('select * from user_usage join user_device on user_device.use_id = u

ser_usage.use_id', locals()) join_ps.head() Out[64]: **outgoing_mins_per_month**

outgoing_sms_per_month monthly_mb use_id use_id user_id

0	21.97	4.82	1557.33	22787	22787	12921
1	1710.08	136.88	7267.55	22788	22788	28714
2	1710.08	136.88	7267.55	22789	22789	28714
3	94.46	35.17	519.12	22790	22790	29592
4	71.59	79.26	1557.33	22792	22792	28217

	platform	count(device)
0	android	184
1	ios	88

Сравнение времени выполнения запросов библиотек Pandas и PandaSQL

In [86]:

```
import timeit

time_group_ps = timeit.timeit("group_ps", setup="from __main__ import group_ps", number
=1)
time_group_ps # 0.0000006999999868639861
```

Out[86]: 6.999999868639861e-

07

In [88]:

```
time_join_ps = timeit.timeit("join_ps", setup="from __main__ import join_ps", number=1)
time_join_ps # 0.0000003999999997625272954
```

Out[88]: 3.9999997625272954e-

07

In [113]:

```
time_group_pd = timeit.timeit("group_pd", setup="from __main__ import group_pd", number
=1) time_group_pd #
0.0000005000000555810402
```

Out[113]: 5.000000555810402e-

07

In [104]:

```
time_join_pd = timeit.timeit("join_pd", setup="from __main__ import join_pd", number=1)
time_join_pd # 0.00000030000000106112566
```

Out[104]:

```
3.000000106112566e-07
```

In [114]:

```
if (time_group_ps > time_group_pd) & (time_join_ps > time_join_pd):
    print("Pandas is better")
else:
    print("PandaSQL is better")
```

```
Pandas is better
```