Московский государственный технический университет имени Н. Э. Баумана

Кафедра «Системы обработки информации и управления»

Лабораторная работа №2

по курсу

«Методы машинного обучения»

на тему:

«Изучение библиотек обработки данных»

Выполнил:

Студент ИУ5-24М

Черната Н. С.

Москва, 2020

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

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.

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
data = pd.read_csv('MMO/lab2/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
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

In [3]:
```
data['sex'].value_counts()
```

Out[3]:
```
Male      21790
Female    10771
Name: sex, dtype: int64
```

In [4]:
```
data.loc[data['sex'] == 'Female', 'age'].mean()
```

Out[4]:
```
36.85823043357163
```

In [5]:
```
float((data['native-country'] == 'Germany').sum()) / data.shape[0]
```

Out[5]:
```
0.004207487485028101
```

In [6]:
```
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 years.
```

In [7]:
```
data.loc[data['salary'] == '>50K', 'education'].unique()
```

Out[7]:
```
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 [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
```

In [10]:
```
data.loc[(data['sex'] == 'Male') &
     (data['marital-status'].isin(['Never-married',
                                   'Separated',
                                   'Divorced',
                                   'Widowed'])), 'salary'].value_counts()
```

Out[10]:
```
<=50K    7552
>50K      697
Name: salary, dtype: int64
```

In [11]:
```
data.loc[(data['sex'] == 'Male') &
     (data['marital-status'].str.startswith('Married')), 'salary'].value_counts()
```

Out[11]:
```
<=50K    7576
>50K     5965
Name: salary, dtype: int64
```

In [12]:
```
data['marital-status'].value_counts()
```

Out[12]:
```
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 [13]:
```
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 [14]:
```
pd.crosstab(data['native-country'], data['salary'],
            values=data['hours-per-week'], aggfunc=np.mean).T
```

Out[14]:

native-country	?	Cambodia	Canada	China	Columbia	Cuba	Dominican-Republic	Ecuador	El-Salvador	England	...	Portugal	Puerto-Rico	Scotland
salary														
<=50K	40.164760	41.416667	37.914634	37.381818	38.684211	37.985714	42.338235	38.041667	36.030928	40.483333	...	41.939394	38.470588	39.444444
>50K	45.547945	40.000000	45.641026	38.900000	50.000000	42.440000	47.000000	48.750000	45.000000	44.533333	...	41.500000	39.416667	46.666667

2 rows × 42 columns

Часть 2.

In [15]:
```
user_usage = pd.read_csv('MMO/lab2/user_usage.csv')
user_device = pd.read_csv('MMO/lab2/user_device.csv')
devices = pd.read_csv('MMO/lab2/android_devices.csv')
```

In [16]:
```
result = pd.merge(user_usage,
                  user_device[['use_id', 'platform', 'device']],
                  on='use_id')
result.head()
```

Out[16]:

	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

In [17]:
```
import pandasql as ps
from pandasql import sqldf
from datetime import datetime
import time
```

In [18]:
```
tic = time.perf_counter()
tutorial = pd.merge(user_usage,
                    user_device[['use_id', 'platform', 'device']],
                    on='use_id')
toc = time.perf_counter()
print(f"Смержено за: {toc - tic:0.4f} seconds")
```

```
Смержено за: 0.0071 seconds
```

In [19]:
```
pysqldf = lambda q: sqldf(q, globals())
q = """
SELECT * FROM user_usage, user_device WHERE user_usage.use_id = user_device.use_id;
"""
tic = time.perf_counter()
joined = pysqldf(q)
toc = time.perf_counter()
print(f"Смержено за: {toc - tic:0.4f} seconds")
```

```
Смержено за: 0.0327 seconds
```

In [20]:
```
joined.head()
```

Out[20]:

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id	use_id	user_id	platform	platform_version	device	use_type_id
0	21.97	4.82	1557.33	22787	22787	12921	android	4.3	GT-I9505	1
1	1710.08	136.88	7267.55	22788	22788	28714	android	6.0	SM-G930F	1
2	1710.08	136.88	7267.55	22789	22789	28714	android	6.0	SM-G930F	1
3	94.46	35.17	519.12	22790	22790	29592	android	5.1	D2303	1
4	71.59	79.26	1557.33	22792	22792	28217	android	5.1	SM-G361F	1

In [21]:
```
joined.describe()
```

Out[21]:

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id	use_id	user_id	platform_version	use_type_id
count	159.000000	159.000000	159.000000	159.000000	159.000000	159.000000	159.000000	159.000000
mean	203.331509	87.978742	4180.378616	22922.327044	22922.327044	25960.918239	5.554717	1.012579
std	248.660581	92.386434	5216.463795	76.511974	76.511974	6275.640431	0.828656	0.111799
min	0.500000	0.250000	0.000000	22787.000000	22787.000000	2873.000000	4.100000	1.000000
25%	70.070000	22.855000	1557.330000	22861.500000	22861.500000	24683.500000	5.000000	1.000000
50%	137.060000	62.850000	2076.450000	22931.000000	22931.000000	29366.000000	6.000000	1.000000
75%	241.035000	119.675000	5191.120000	22986.500000	22986.500000	29673.000000	6.000000	1.000000
max	1710.080000	540.600000	31146.670000	23053.000000	23053.000000	29725.000000	10.100000	2.000000

In [22]:
```
joined.groupby("platform_version")["outgoing_mins_per_month"].describe()
```

Out[22]:

	count	mean	std	min	25%	50%	75%	max
platform_version								
4.1	5.0	53.566000	34.298639	16.34	16.340	74.59	74.5900	85.97
4.2	1.0	189.100000	NaN	189.10	189.100	189.10	189.1000	189.10
4.3	3.0	119.030000	161.601371	21.97	25.755	29.54	167.5600	305.58
4.4	17.0	313.242353	335.849533	12.85	61.220	78.80	797.0600	797.06
5.0	17.0	179.453529	81.117511	61.43	123.200	143.81	249.2600	360.86
5.1	23.0	131.250000	70.153911	42.75	71.590	109.32	198.0500	248.95
6.0	88.0	203.891591	260.770789	0.50	67.150	145.55	244.8925	1710.08
7.0	2.0	689.590000	752.729311	157.33	423.460	689.59	955.7200	1221.85