МГТУ им. Н. Э. Баумана, кафедра ИУ5

курс “Технология машинного обучения”

Лабораторная работа №2

«Изучение библиотек обработки данных»

ВЫПОЛНИЛ:

Матюнин да Вейга Р.А.

Группа: ИУ5-61Б

ПРОВЕРИЛ:

Гапанюк Ю.Е.

Москва 2020

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

Задание:

Часть 1.

Условие задания

- **In this task you should use Pandas to answer a few questions about the Adult dataset.** https://nbviewer.jupyter.org/github/Yorko/mlcourse_open/blob/master/jupyter_english/assignments_demo/assignment01_pandas_uci_adult.ipynb?flush_cache=true

Набор данных

- https://archive.ics.uci.edu/ml/datasets/Adult

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.

Выполненная работа

In [1]:
```
import numpy as np
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
# we don't like warnings
# you can comment the following 2 lines if you'd like to
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
data = pd.read_csv('data/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

Часть 1.

1. **How many men and women (*sex* feature) are represented in this dataset?**

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

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

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

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

Out[4]: 36.85823043357163
```

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

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

Out[5]: 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)
```

4. **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 [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}
            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.
```

5. **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 [7]: data.loc[data['salary'] == '>50K', 'education'].unique() # No

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)
```

6. **Display statistics of age for each race (*race* feature) and each gender. Use *groupby()* and *describe()*. Find the maximum age of men of *Amer-Indian-Eskimo* race.**

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
```

...

7. **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 [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
```

In [10]:
```
data.loc[(data['sex'] == 'Male') &
     (data['marital-status'].str.startswith('Married')), 'salary'].valu
```

Out[10]:
```
<=50K    7576
>50K     5965
Name: salary, dtype: int64
```

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
```

8. **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 [12]: 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_workahol
         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%
```

9. **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 [13]: pd.crosstab(data['native-country'], data['salary'],
                     values=data['hours-per-week'], aggfunc=np.mean).T
```

Out[13]:

s	Honduras	Hong	Hungary	India	Iran	Ireland	Italy	Jamaica	Japan	Laos	Mexico	Nicaragua	Outlying-US(Guam-USVI-etc)	Peru	Philippines
0	34.333333	39.142857	31.3	38.233333	41.44	40.947368	39.625	38.239437	41.000000	40.375	40.003279	36.09375	41.857143	35.068966	38.065693
N	60.000000	45.000000	50.0	46.475000	47.50	48.000000	45.400	41.100000	47.958333	40.000	46.575758	37.50000	NaN	40.000000	43.032787

Часть 2.

```
In [1]: %matplotlib inline
        import pandas as pd
        import pandasql as ps
        from datetime import datetime
        import seaborn
        import matplotlib.pyplot as plt

        %config InlineBackend.figure_format = 'svg'
        from pylab import rcParams
        rcParams['figure.figsize'] = 8, 5
```

```
In [2]: pd.__version__
```

```
Out[2]: '1.0.1'
```

```
In [3]: project_submissions = pd.read_csv('./data/project_submissions.csv')
        daily_engagements = pd.read_csv('./data/daily_engagement.csv')
        enrollments = pd.read_csv('./data/enrollments.csv')
```

1. Один произвольный запрос на соединение двух наборов данных:

In [4]:
```
# pandasql code
def example1_pandasql(daily_engagements):
    simple_query = '''
        SELECT
            acct,
            total_minutes_visited,
            utc_date
        FROM daily_engagements
        ORDER BY total_minutes_visited desc
        LIMIT 10
        '''
    return ps.sqldf(simple_query, locals())

example1_pandasql(daily_engagements)
```

Out[4]:

	acct	total_minutes_visited	utc_date
0	317	1030.883197	2015-07-11
1	328	945.538914	2015-07-09
2	198	876.512846	2014-12-30
3	163	872.633923	2015-07-10
4	573	866.405226	2015-07-11
5	303	856.634726	2015-05-14
6	619	853.253236	2015-07-10
7	163	850.519340	2015-07-09
8	108	820.879483	2015-02-20
9	278	816.895443	2015-07-09

In [5]:
```
# pandas code
def example1_pandas(daily_engagements):
    return daily_engagements[['acct', 'total_minutes_visited', 'utc_dat

example1_pandas(daily_engagements)
```

Out[5]:

	acct	total_minutes_visited	utc_date
54536	317	1030.883197	2015-07-11
56403	328	945.538914	2015-07-09
33728	198	876.512846	2014-12-30
27699	163	872.633923	2015-07-10
97492	573	866.405226	2015-07-11
51779	303	856.634726	2015-05-14
105968	619	853.253236	2015-07-10
27698	163	850.519340	2015-07-09
18394	108	820.879483	2015-02-20
47372	278	816.895443	2015-07-09

2. Один произвольный запрос на группировку набора данных с использованием функций агрегирования:

In [9]:
```
# pandasql code
def example2_pandasql(daily_engagements):
    aggr_query = '''
        SELECT
            avg(total_minutes_visited) as total_minutes_visited,
            weekday
        FROM daily_engagements
        GROUP BY weekday
        '''
    return ps.sqldf(aggr_query, locals()).set_index('weekday')

# pandas code
def example2_pandas(daily_engagements):
    return pd.DataFrame(daily_engagements.groupby('weekday').total_
minutes_visited.mean())
```

In [10]:
```
weekday_engagement = example2_pandasql(daily_engagements)
weekday_engagement
```

Out[10]:

	total_minutes_visited
weekday	
Friday	23.156233
Monday	26.418982
Saturday	21.725677
Sunday	23.539406
Thursday	24.685176
Tuesday	26.857676
Wednesday	25.362789

In [11]:
```
example2_pandas(daily_engagements)
```

Out[11]:

	total_minutes_visited
weekday	
Friday	23.156233
Monday	26.418982
Saturday	21.725677
Sunday	23.539406
Thursday	24.685176
Tuesday	26.857676
Wednesday	25.362789

Время выполнения запроса в Pandas было быстрее чем в PandaSQL

- Ноутбук с выполненной работой и отчет размещены в репозитории на github: https://github.com/Yorati/TMO