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)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ**

Отчет по лабораторной работе № 2
«Изучение библиотек обработки данных»
по курсу “Технологии машинного обучения”

Исполнитель:
Студент группы ИУ5-63
Желанкина А.С.
___________ 28.02.2018

Москва, 2019

Задание лабораторной работы

<u>Часть 1.</u>

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

Набор данных можно скачать здесь - https://archive.ics.uci.edu/ml/datasets/Adult

<u>Часть 2.</u>

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.

Экранные формы с текстом программы и примерами её выполнения

<u>ЧАСТЬ 1</u>

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

In [2]:
```
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data'
data = pd.read_csv(url, sep=', ',header=None)
data.head()
```

```
C:\Anaconda\lib\site-packages\ipykernel_launcher.py:2: ParserWarning: Falling back to the 'python' en
gine because the 'c' engine does not support regex separators (separators > 1 char and different from
'\s+' are interpreted as regex); you can avoid this warning by specifying engine='python'.
```

Out[2]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
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

In [3]:
```
data.columns = ['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[3]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40

In [4]:
```
#How many men and women (sex feature) are represented in this dataset?

print('Total people', data['sex'].count())
print(data['sex'].value_counts())
```

```
Total people 32561
Male      21790
Female    10771
Name: sex, dtype: int64
```

In [5]:
```
#What is the average age (age feature) of women?

average_age = 0
counter = 0

for i in range(data['sex'].count()):
    if data['sex'][i] == 'Female':
        counter += 1
        average_age += data['age'][i]
print('The average age of women is ', average_age / counter)
```

```
The average age of women is  36.85823043357163
```

In [6]:
```
#What is the percentage of German citizens (native-country feature)?

counter_of_german = 0

for country in data['native-country']:
    if country == 'Germany':
        counter_of_german += 1

print('The percentage of German citizens is ',
      counter_of_german / data['native-country'].count() * 100, '%')
```

```
The percentage of German citizens is  0.42074874850281013 %
```

In [7]:
```
#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?

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

```
The average age of the rich: 44 +- 10.52 years.
The average age of the poor: 37 +- 14.02 years.
```

In [8]:
```
# 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)

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

Out[8]:
```
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 [10]:
```
#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.

amer = data.loc[(data['race'] == 'Amer-Indian-Eskimo') &
                (data['sex'] == 'Male'), 'age'].max()
print('The maximum age of men of Amer-Indian-Eskimo race is ', amer, '\n')

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

```
The maximum age of men of Amer-Indian-Eskimo race is  82

                                   capital-gain                              \
                                          count        mean          std  min
race                sex    age
Amer-Indian-Eskimo Female 17             2.0   527.500000   745.997654  0.0
                          18             2.0     0.000000     0.000000  0.0
                          19             5.0     0.000000     0.000000  0.0
                          21             4.0     0.000000     0.000000  0.0
                          22             4.0  4307.250000  6827.980442  0.0
                          23             4.0     0.000000     0.000000  0.0
                          24             3.0     0.000000     0.000000  0.0
                          25             4.0     0.000000     0.000000  0.0
                          27             4.0   831.250000  1662.500000  0.0
                          28             4.0     0.000000     0.000000  0.0
                          29             3.0     0.000000     0.000000  0.0
                          30             4.0     0.000000     0.000000  0.0
                          31             7.0     0.000000     0.000000  0.0
                          32             1.0     0.000000          NaN  0.0
                          33             2.0     0.000000     0.000000  0.0
```

click to unscroll output; double click to hide

In [11]:

```python
#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.

print('Married men:\n')
print(data.loc[(data['sex'] == 'Male') & (data['marital-status'].str.startswith('Married')),
         'salary'].value_counts())
print('\nSingle men:\n')
print(data.loc[(data['sex'] == 'Male') &
     (data['marital-status'].isin(['Never-married', 'Separated', 'Divorced', 'Widowed'])),
               'salary'].value_counts())
```

```
Married men:

<=50K    7576
>50K     5965
Name: salary, dtype: int64

Single men:

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

In [16]:

```python
#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 among them?

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

hard = data[data['hours-per-week'] == maximum].shape[0]
print(hard, 'people work such a number of hours')

rich = float(data[(data['hours-per-week'] == maximum) &
                  (data['salary'] == '>50K')].shape[0]) / hard
print('The percentage of those who earn a lot among them is', rich * 100, '%')
```

```
The maximum number of hours a person works per week is 99
85 people work such a number of hours
The percentage of those who earn a lot among them is 29.411764705882355 %
```

In [26]:

```python
#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?

little = []
lot = []
country_salary = data.groupby(['native-country', 'salary'])
for (country, salary), hours in country_salary:
    print(country, salary, int(hours['hours-per-week'].mean()))
```

```
? <=50K 40
? >50K 45
Cambodia <=50K 41
Cambodia >50K 40
Canada <=50K 37
Canada >50K 45
China <=50K 37
China >50K 38
Columbia <=50K 38
Columbia >50K 50
Cuba <=50K 37
Cuba >50K 42
Dominican-Republic <=50K 42
Dominican-Republic >50K 47
```

ЧАСТЬ 2

```
[ ]  !pip3 install pandasql
```

```
Collecting pandasql
  Downloading https://files.pythonhosted.org/packages/6b/c4/ee4096ffa2eeeca0c749b26f0371bd26aa5c8b611c43de99a4f86d
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from pandasql) (1.14.6)
Requirement already satisfied: pandas in /usr/local/lib/python3.6/dist-packages (from pandasql) (0.22.0)
Requirement already satisfied: sqlalchemy in /usr/local/lib/python3.6/dist-packages (from pandasql) (1.2.18)
Requirement already satisfied: python-dateutil>=2 in /usr/local/lib/python3.6/dist-packages (from pandas->pandasql
Requirement already satisfied: pytz>=2011k in /usr/local/lib/python3.6/dist-packages (from pandas->pandasql) (2018
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packages (from python-dateutil>=2->pandas
Building wheels for collected packages: pandasql
  Building wheel for pandasql (setup.py) ... done
  Stored in directory: /root/.cache/pip/wheels/53/6c/18/b87a2e5fa8a82e9c026311de56210b8d1c01846e18a9607fc9
Successfully built pandasql
Installing collected packages: pandasql
Successfully installed pandasql-0.7.3
```

```
[28]  import numpy as np
      import pandas as pd
      import pandasql as ps
      from time import time
```

```
[29]  url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wdbc.data'
      data_diagnostic = pd.read_csv(url, sep=',',header=None)
      data_diagnostic.columns = ['id'] + ['D' + str(i) for i in range(1, 32)]
      data_diagnostic.head()
```

	id	D1	D2	D3	D4	D5	D6	D7	D8	D9	...	D22	D23	D24	D25	D:
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...	25.38	17.33	184.60	2019.0	0.162
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...	24.99	23.41	158.80	1956.0	0.12:
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...	23.57	25.53	152.50	1709.0	0.14
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...	14.91	26.50	98.87	567.7	0.20
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...	22.54	16.67	152.20	1575.0	0.13

5 rows × 32 columns

```
[30]  url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/wpbc.data'
      data_prognostic = pd.read_csv(url, sep=',',header=None)
      data_prognostic.columns = ['id'] + ['P' + str(i) for i in range(1, 35)]
      data_prognostic.head()
```

	id	P1	P2	P3	P4	P5	P6	P7	P8	P9	...	P25	P26	P27	P28	P29	
0	119513	N	31	18.02	27.60	117.50	1013.0	0.09489	0.1036	0.1086	...	139.70	1436.0	0.1195	0.1926	0.3140	0
1	8423	N	61	17.99	10.38	122.80	1001.0	0.11840	0.2776	0.3001	...	184.60	2019.0	0.1622	0.6656	0.7119	0
2	842517	N	116	21.37	17.44	137.50	1373.0	0.08836	0.1189	0.1255	...	159.10	1949.0	0.1188	0.3449	0.3414	0
3	843483	N	123	11.42	20.38	77.58	386.1	0.14250	0.2839	0.2414	...	98.87	567.7	0.2098	0.8663	0.6869	0
4	843584	R	27	20.29	14.34	135.10	1297.0	0.10030	0.1328	0.1980	...	152.20	1575.0	0.1374	0.2050	0.4000	0

5 rows × 35 columns

```
[31]  # 1 - diagnosis, 2 - radius, 22 - worst radius
      t0_diagnostic = time()
      aggregations_diagnostic = {
          'D22': lambda x: max(x)
      }
      print(data_diagnostic.groupby('D1').agg(aggregations_diagnostic))
      t1_diagnostic = time()
      print('It takes: ', t1_diagnostic - t0_diagnostic)
```

```
      D22
D1
B   19.82
M   36.04
It takes:  0.005372047424316406
```

```
# 1 - diagnosis, 2 - radius, 22 - worst radius
t0_diagnostic = time()
aggr_query_diagnostic = '''
    SELECT
        D1,
        max(D22)
    FROM data_diagnostic
    GROUP BY D1
    '''
```

```
print(ps.sqldf(aggr_query_diagnostic, locals()))
t1_diagnostic = time()
print('It takes: ', t1_diagnostic - t0_diagnostic)
```

```
  D1  max(D22)
0  B     19.82
1  M     36.04
It (diagnostic) takes:  0.04991793632507324
```

[37]
```
t0 = time()
result = pd.merge(data_diagnostic[['id','D1', 'D2', 'D22']],
                  data_prognostic[['id', 'P1', 'P2']],
                  on='id')
t1 = time()
print(result)
print('It takes: ', t1 - t0)
```

```
         id D1     D2    D22 P1   P2
0    842517  M  20.57  24.99  N  116
1    843786  M  12.45  15.47  R   77
2    844359  M  18.25  22.88  N   60
3    844981  M  13.00  15.49  N  119
4    845636  M  16.02  19.19  N  123
```

[36]
```
t0 = time()
merge_query = '''
    SELECT
        d.id,
        d.D1,
        d.D2,
        d.D22,
        p.id,
        p.P1,
        p.P2
    FROM data_diagnostic as d JOIN data_prognostic as p ON (d.id = p.id)
    GROUP BY p.id
    '''
print(ps.sqldf(merge_query, locals()))
t1 = time()
print('It takes: ', t1 - t0)
```

```
         id D1     D2    D22        id P1   P2
0     85715  M  13.17  15.67     85715  N  111
1     86208  M  20.26  24.22     86208  R   10
2     86517  M  18.66  22.25     86517  N  108
3     87163  M  13.43  17.98     87163  N   84
4     87164  M  15.46  18.79     87164  N   98
5     87880  M  13.81  19.20     87880  N   38
6     89122  M  19.40  23.79     89122  N   17
```