Министерство науки и высшего образования 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 ИУ5 «Системы обработки информации и управления»

Отчет

по дисциплине «Технология Машинного обучения»

Выполнил:
студент группы ИУ5-62
Миронов Святослав
подпись, дата

2019 г.

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

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

Пример решения задания - https://www.kaggle.com/kashnitsky/a1-demo-pandas-and-uci-adult-dataset-solution

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.

В качестве примеров можно использовать следующие статьи:

- https://www.shanelynn.ie/summarising-aggregation-and-grouping-data-in-python-pandas/
- https://www.shanelynn.ie/merge-join-dataframes-python-pandas-index-1/ (в разделе "Example data" данной статьи содержится рекомендуемый набор данных для проведения экспериментов).

Lab2

June 3, 2019

In [1]:
```
!pip install seaborn
```

```
Requirement already satisfied: seaborn in /srv/conda/lib/python3.6/site-packages (0.9.0)
Requirement already satisfied: scipy>=0.14.0 in /srv/conda/lib/python3.6/site-packages (from se
Requirement already satisfied: pandas>=0.15.2 in /srv/conda/lib/python3.6/site-packages (from s
Requirement already satisfied: matplotlib>=1.4.3 in /srv/conda/lib/python3.6/site-packages (fro
Requirement already satisfied: numpy>=1.9.3 in /srv/conda/lib/python3.6/site-packages (from sea
Requirement already satisfied: python-dateutil>=2.5.0 in /srv/conda/lib/python3.6/site-packages
Requirement already satisfied: pytz>=2011k in /srv/conda/lib/python3.6/site-packages (from pand
Requirement already satisfied: cycler>=0.10 in /srv/conda/lib/python3.6/site-packages (from mat
Requirement already satisfied: kiwisolver>=1.0.1 in /srv/conda/lib/python3.6/site-packages (fro
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /srv/conda/lib/pytho
Requirement already satisfied: six>=1.5 in /srv/conda/lib/python3.6/site-packages (from python-
Requirement already satisfied: setuptools in /srv/conda/lib/python3.6/site-packages (from kiwis
```

In [2]:
```
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
data = pd.read_csv('adult.csv')
data.head()
```

Out[3]:

	age	workclass	fnlwgt	education	education-num	\
0	39	State-gov	77516	Bachelors	13	
1	50	Self-emp-not-inc	83311	Bachelors	13	
2	38	Private	215646	HS-grad	9	
3	53	Private	234721	11th	7	
4	28	Private	338409	Bachelors	13	

	marital-status	occupation	relationship	race	sex	\
0	Never-married	Adm-clerical	Not-in-family	White	Male	
1	Married-civ-spouse	Exec-managerial	Husband	White	Male	
2	Divorced	Handlers-cleaners	Not-in-family	White	Male	
3	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	
4	Married-civ-spouse	Prof-specialty	Wife	Black	Female	

```
   capital-gain  capital-loss  hours-per-week native-country salary
0          2174             0              40  United-States  <=50K
1             0             0              13  United-States  <=50K
2             0             0              40  United-States  <=50K
3             0             0              40  United-States  <=50K
4             0             0              40           Cuba  <=50K
```

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

```
In [4]: data["sex"].value_counts()
```

```
Out[4]: Male      21790
        Female    10771
        Name: sex, dtype: int64
```

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

```
In [5]: data[data["sex"] == "Female"]["age"].mean()
```

```
Out[5]: 36.85823043357163
```

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

```
In [6]: print("{0:%}".format(data[data["native-country"] == "Germany"].shape[0] / data.shape[0]
```

```
0.420749%
```

4 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 [7]: ages1 = data[data["salary"] == "<=50K"]["age"]
        ages2 = data[data["salary"] ==  ">50K"]["age"]
        print("<=50K: = {0} ś {1} years".format(ages1.mean(), ages1.std()))
        print(" >50K: = {0} ś {1} years".format(ages2.mean(), ages2.std()))
```

```
<=50K: = 36.78373786407767 ś 14.02008849082488 years
 >50K: = 44.24984058155847 ś 10.519027719851826 years
```

5 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 [8]: high_educations = set(["Bachelors", "Prof-school", "Assoc-acdm",
                               "Assoc-voc", "Masters", "Doctorate"])
        def high_educated(e):
            return e in high_educations

        data[data["salary"] == ">50K"]["education"].map(high_educated).all()
```

```
Out[8]: False
```

6 7. 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.

```
In [9]: data.groupby(["race", "sex"])["age"].describe()
```

```
Out[9]:                                count       mean        std   min   25%   50%  \
        race                 sex
        Amer-Indian-Eskimo   Female    119.0  37.117647  13.114991  17.0  27.0  36.0
                             Male      192.0  37.208333  12.049563  17.0  28.0  35.0
        Asian-Pac-Islander   Female    346.0  35.089595  12.300845  17.0  25.0  33.0
                             Male      693.0  39.073593  12.883944  18.0  29.0  37.0
        Black                Female   1555.0  37.854019  12.637197  17.0  28.0  37.0
                             Male     1569.0  37.682600  12.882612  17.0  27.0  36.0
        Other                Female    109.0  31.678899  11.631599  17.0  23.0  29.0
                             Male      162.0  34.654321  11.355531  17.0  26.0  32.0
        White                Female   8642.0  36.811618  14.329093  17.0  25.0  35.0
                             Male    19174.0  39.652498  13.436029  17.0  29.0  38.0

                                       75%   max
        race                 sex
        Amer-Indian-Eskimo   Female  46.00  80.0
                             Male    45.00  82.0
        Asian-Pac-Islander   Female  43.75  75.0
                             Male    46.00  90.0
        Black                Female  46.00  90.0
                             Male    46.00  90.0
        Other                Female  39.00  74.0
                             Male    42.00  77.0
        White                Female  46.00  90.0
                             Male    49.00  90.0
```

7 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 [10]: def is_married(m):
             return m.startswith("Married")

         data["married"] = data["marital-status"].map(is_married)
         (data[(data["sex"] == "Male") & (data["salary"] == ">50K")]
              ["married"].value_counts())
```

```
Out[10]: True     5965
         False     697
         Name: married, dtype: int64
```

8 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 [11]: m = data["hours-per-week"].max()
         print("Maximum is {} hours/week.".format(m))

         people = data[data["hours-per-week"] == m]
         c = people.shape[0]
         print("{} people work this time at week.".format(c))

         s = people[people["salary"] == ">50K"].shape[0]
         print("{0:%} get >50K salary.".format(s / c))
```

```
Maximum is 99 hours/week.
85 people work this time at week.
29.411765% get >50K salary.
```

9 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 [12]: p = pd.crosstab(data["native-country"], data["salary"],
                         values=data['hours-per-week'], aggfunc="mean")
         p
```

Out[12]:

native-country	<=50K	>50K
?	40.164760	45.547945
Cambodia	41.416667	40.000000
Canada	37.914634	45.641026
China	37.381818	38.900000
Columbia	38.684211	50.000000
Cuba	37.985714	42.440000
Dominican-Republic	42.338235	47.000000
Ecuador	38.041667	48.750000
El-Salvador	36.030928	45.000000
England	40.483333	44.533333
France	41.058824	50.750000
Germany	39.139785	44.977273
Greece	41.809524	50.625000
Guatemala	39.360656	36.666667
Haiti	36.325000	42.750000
Holand-Netherlands	40.000000	NaN
Honduras	34.333333	60.000000
Hong	39.142857	45.000000
Hungary	31.300000	50.000000
India	38.233333	46.475000
Iran	41.440000	47.500000
Ireland	40.947368	48.000000
Italy	39.625000	45.400000
Jamaica	38.239437	41.100000
Japan	41.000000	47.958333
Laos	40.375000	40.000000
Mexico	40.003279	46.575758
Nicaragua	36.093750	37.500000
Outlying-US(Guam-USVI-etc)	41.857143	NaN
Peru	35.068966	40.000000
Philippines	38.065693	43.032787
Poland	38.166667	39.000000
Portugal	41.939394	41.500000
Puerto-Rico	38.470588	39.416667
Scotland	39.444444	46.666667
South	40.156250	51.437500
Taiwan	33.774194	46.800000
Thailand	42.866667	58.333333
Trinadad&Tobago	37.058824	40.000000
United-States	38.799127	45.505369
Vietnam	37.193548	39.200000
Yugoslavia	41.600000	49.500000

In [13]:
```
devices = pd.read_csv('user_device.csv')
usage = pd.read_csv('user_usage.csv')
android = pd.read_csv('android_devices.csv')
```

```
devices.head()
```

Out[13]:

	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 [14]: usage.head()
```

Out[14]:

	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 [15]: android.head()
```

Out[15]:

	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 [16]: !pip install pandasql

Collecting pandasql
  Downloading https://files.pythonhosted.org/packages/6b/c4/ee4096ffa2eeeca0c749b26f0371bd26aa5
Requirement already satisfied: numpy in /srv/conda/lib/python3.6/site-packages (from pandasql)
Requirement already satisfied: pandas in /srv/conda/lib/python3.6/site-packages (from pandasql)
Requirement already satisfied: sqlalchemy in /srv/conda/lib/python3.6/site-packages (from panda
Requirement already satisfied: python-dateutil>=2.5.0 in /srv/conda/lib/python3.6/site-packages
Requirement already satisfied: pytz>=2011k in /srv/conda/lib/python3.6/site-packages (from pand
Requirement already satisfied: six>=1.5 in /srv/conda/lib/python3.6/site-packages (from python-
Building wheels for collected packages: pandasql
  Running setup.py bdist_wheel for pandasql ... done
  Stored in directory: /home/jovyan/.cache/pip/wheels/53/6c/18/b87a2e5fa8a82e9c026311de56210b8c
Successfully built pandasql
Installing collected packages: pandasql
Successfully installed pandasql-0.7.3
```

```
In [18]: res=pd.merge(devices,usage, left_on='use_id', right_on='use_id', how='inner')
         res.head()
```

Out[18]:

	use_id	user_id	platform	platform_version	device	use_type_id	\
0	22787	12921	android	4.3	GT-I9505	1	

```
1   22788    28714  android              6.0  SM-G930F           1
2   22789    28714  android              6.0  SM-G930F           1
3   22790    29592  android              5.1     D2303           1
4   22792    28217  android              5.1  SM-G361F           1

   outgoing_mins_per_month  outgoing_sms_per_month  monthly_mb
0                    21.97                    4.82     1557.33
1                  1710.08                  136.88     7267.55
2                  1710.08                  136.88     7267.55
3                    94.46                   35.17      519.12
4                    71.59                   79.26     1557.33
```

```
In [19]: %%timeit
         res=pd.merge(devices,usage, left_on='use_id', right_on='use_id', how='inner')
```

```
3.02 ms ś 128 ţs per loop (mean ś std. dev. of 7 runs, 100 loops each)
```

,

```
In [21]: result = pd.merge(usage,devices[['use_id', 'platform', 'device']],on='use_id')
         result.head()
```

```
Out[21]:    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

  platform    device
0  android  GT-I9505
1  android  SM-G930F
2  android  SM-G930F
3  android     D2303
4  android  SM-G361F
```

```
In [23]: from pandasql import sqldf
         pysqldf = lambda q: sqldf(q, globals())
```

```
In [24]: pysqldf("""SELECT *
                    FROM devices AS d JOIN usage AS u
                    ON d.use_id = u.use_id
                 """).head()
```

```
Out[24]:    use_id  user_id platform  platform_version    device  use_type_id  \
0   22787    12921  android              4.3  GT-I9505           1
1   22788    28714  android              6.0  SM-G930F           1
2   22789    28714  android              6.0  SM-G930F           1
```

```
3   22790    29592  android               5.1     D2303            1
4   22792    28217  android               5.1  SM-G361F            1

   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 [25]: %%timeit
         pysqldf("""SELECT *
                    FROM devices AS d JOIN usage AS u
                    ON d.use_id = u.use_id
                 """).head()
```

```
17.6 ms ś 1.37 ms per loop (mean ś std. dev. of 7 runs, 100 loops each)
```

10 , pdqsl 6

```
In [27]: result.groupby("platform")["monthly_mb"].mean().head()
```

```
Out[27]: platform
         android    4221.387834
         ios         961.155000
         Name: monthly_mb, dtype: float64
```

```
In [28]: %%timeit
         result.groupby("platform")["monthly_mb"].mean().head()
```

```
870 ţs ś 35.7 ţs per loop (mean ś std. dev. of 7 runs, 1000 loops each)
```

```
In [29]: pysqldf("""SELECT platform, AVG(monthly_mb)
                    FROM result
                    GROUP BY platform
                 """).head()
```

```
Out[29]:   platform  AVG(monthly_mb)
         0  android      4221.387834
         1      ios       961.155000
```

```
In [30]: %%timeit
         pysqldf("""SELECT platform, AVG(monthly_mb)
                    FROM result
                    GROUP BY platform
                 """).head()
```

```
8.54 ms ś 1.03 ms per loop (mean ś std. dev. of 7 runs, 100 loops each)
```

11 , pdsql 10

In [0]: