МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Кафедра «Систем обработки информации и управления»

ОТЧЕТ

Лабораторная работа №2
по курсу «Методы машинного обучения»

Тема: «Изучение библиотек обработки данных»

ИСПОЛНИТЕЛЬ: _Бабин В.Е._____
ФИО
группа ИУ5-22 _____
подпись
"__"_____2020 г.

ПРЕПОДАВАТЕЛЬ: _____
ФИО
_____
подпись
"__"_____2020 г.

Москва - 2020

[ ]: Part 1

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-spouse-absent, Married-AF-spouse.

occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, 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(Guam-USVI-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

[2]:
```
import numpy as np
import pandas as pd
```

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

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

[4]:
```
print('{} <-- all'.format(data['sex'].shape[0]))
print('{} <-- male'.format(data[data['sex'] == 'Male'].shape[0]))
print('{} <-- female'.format(data[data['sex'] == 'Female'].shape[0]))
```

```
32561 <-- all
21790 <-- male
10771 <-- female
```

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

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

[5]:
```
36.85823043357163
```

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

[6]:
```
data[data['native-country'] == 'Germany'].shape[0] / data.shape[0] * 100
```

[6]:
```
0.4207487485028101 3
```

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?

[7]:
```
more_50_salary_aged = data[data['salary'] == '>50K']['age']
less_50_salary_aged = data[data['salary'] == '<=50K']['age']
```

```
print('More 50K: mean = {}; std =
 {}'.format(more_50_salary_aged.mean(), ␣ ↪more_50_salary_aged.std()))
print('Less 50K: mean = {}; std =
 {}'.format(less_50_salary_aged.mean(), ␣ ↪less_50_salary_aged.std()))
```

```
More 50K: mean = 44.24984058155847; std = 10.519027719851826
Less 50K: mean = 36.78373786407767; std = 14.02008849082488
```

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)

[8]:
```
high_school = ['Bachelors', 'Prof-school', 'Assoc-acdm',
 'Assoc-voc', ␣ ↪'Masters', 'Doctorate']
actual_education = data[data['salary'] == '>50K']['education'].unique()
not_high = [edu for edu in actual_education if edu not in high_school]
print('only with high education? {}'.format(not_high == []))
```

```
only with high education? False
```

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.

[9]:
```
data.groupby('sex')['age'].describe()
```

[9]:

sex	count	mean	std	min	25%	50%	75%	max
Female	10771.0	36.858230	14.013697	17.0	25.0	35.0	46.0	90.0
Male	21790.0	39.433547	13.370630	17.0	29.0	38.0	48.0	90.0

[10]:
```
data.groupby('race')['age'].describe()
```

[10]:

race	count	mean	std	min	25%	50%	75%	max
Amer-Indian-Eskimo	311.0	37.173633	12.447130	17.0	28.0	35.0	45.5	82.0
Asian-Pac-Islander	1039.0	37.746872	12.825133	17.0	28.0	36.0	45.0	90.0
Black	3124.0	37.767926	12.759290	17.0	28.0	36.0	46.0	90.0
Other	271.0	33.457565	11.538865	17.0	25.0	31.0	41.0	77.0
White	27816.0	38.769881	13.782306	17.0	28.0	37.0	48.0	90.0

[30]:
```
data.loc[(data['race'] == 'Amer-Indian-Eskimo') &
 (data['sex'] ==␣ ↪'Male')]['age'].max()
```

[30]:
```
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.

[34]:
```
married_status = ['Married-civ-spouse', 'Married-
 spouse-absent',␣ ↪'Married-AF-spouse']
more_50_men = data.loc[(data['salary'] == '>50K') & (data['sex'] == 'Male')]
married = more_50_men.loc[(more_50_men['marital-status'].isin(married_status))].
 ↪shape[0]
not_married = more_50_men.shape[0] - married
print('married amount = {}; not married = {}'.format(married, not_married))
print('married / not married = {}'.format(married/not_married))
```

```
married amount = 5965; not married = 697
married / not married = 8.558106169296988
```

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?

[50]:
```
 max_hours_per_week = data['hours-per-week'].max()
people_with_max = data[data['hours-per-week'] == max_hours_per_week]
percent = people_with_max[people_with_max['salary'] ==
 '>50K'].shape[0] /␣ ↪people_with_max.shape[0]* 100
print('max hours per week = {}'.format(max_hours_per_week))
print('people with max hours per week = {}'.format(people_with_max.shape[0]))
print('% = {}'.format(percent))
```

```
max hours per week = 99
people with max hours per week = 85
% = 29.411764705882355
```

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?

[68]:
```
less_50 = data[data['salary'] == '>50K']
more_50 = data[data['salary'] == '<=50K']

less_50_group = less_50.groupby('native-country')['hours-per-week'].mean()
more_50_group = more_50.groupby('native-country')['hours-per-week'].mean()

for i in gg.keys():
```

```
    print('country: {}; hours with little salary: {}; hours with big
 salary {}'. ↪format(i, less_50_group[i], more_50_group[i]))
```

```
country: ?; hours with little salary: 45.54794520547945; hours with
big salary 40.16475972540046
country: Cambodia; hours with little salary: 40.0; hours with
big salary 41.416666666666664
country: Canada; hours with little salary: 45.64102564102564;
hours with big salary 37.91463414634146
country: China; hours with little salary: 38.9; hours with
big salary 37.38181818181818
country: Columbia; hours with little salary: 50.0; hours with
big salary 38.68421052631579
country: Cuba; hours with little salary: 42.44; hours with
big salary 37.98571428571429
country: Dominican-Republic; hours with little salary: 47.0;
hours with big salary 42.338235294117645
country: Ecuador; hours with little salary: 48.75; hours with
big salary 38.041666666666664
country: El-Salvador; hours with little salary: 45.0; hours with
big salary 36.03092783505155
country: England; hours with little salary: 44.53333333333333;
hours with big salary 40.483333333333334
country: France; hours with little salary: 50.75; hours with
big salary 41.05882352941177
country: Germany; hours with little salary: 44.97727272727273;
hours with big salary 39.13978494623656
country: Greece; hours with little salary: 50.625; hours with
big salary 41.80952380952381
country: Guatemala; hours with little salary: 36.666666666666664;
hours with big salary 39.36065573770492
country: Haiti; hours with little salary: 42.75; hours with big salary 36.325
country: Honduras; hours with little salary: 60.0; hours with
big salary 34.333333333333336
country: Hong; hours with little salary: 45.0; hours with
big salary 39.142857142857146
country: Hungary; hours with little salary: 50.0; hours with big salary 31.3
country: India; hours with little salary: 46.475; hours with
big salary 38.233333333333334
country: Iran; hours with little salary: 47.5; hours with big salary 41.44
country: Ireland; hours with little salary: 48.0; hours with
big salary 40.94736842105263
country: Italy; hours with little salary: 45.4; hours with big salary 39.625
country: Jamaica; hours with little salary: 41.1; hours with
big salary 38.23943661971831
country: Japan; hours with little salary: 47.958333333333336;
hours with big salary 41.0
```

```
country: Laos; hours with little salary: 40.0; hours with big
salary 40.375 country: Mexico; hours with little salary:
46.57575757575758; hours with big salary 40.00327868852459
country: Nicaragua; hours with little salary: 37.5; hours with
big salary 36.09375
country: Peru; hours with little salary: 40.0; hours with
big salary 35.06896551724138
country: Philippines; hours with little salary: 43.032786885245905;
hours with big salary 38.065693430656935
country: Poland; hours with little salary: 39.0; hours with
big salary 38.166666666666664
country: Portugal; hours with little salary: 41.5; hours with
big salary 41.93939393939394
country: Puerto-Rico; hours with little salary: 39.416666666666664;
hours with big salary 38.470588235294116
country: Scotland; hours with little salary: 46.666666666666664;
hours with big salary 39.44444444444444
country: South; hours with little salary: 51.4375; hours with
big salary 40.15625
country: Taiwan; hours with little salary: 46.8; hours with
big salary 33.774193548387096
country: Thailand; hours with little salary: 58.333333333333336;
hours with big salary 42.86666666666667
country: Trinadad&Tobago; hours with little salary: 40.0; hours with
big salary 37.05882352941177
country: United-States; hours with little salary: 45.50536884674383;
hours with big salary 38.79912723305605
country: Vietnam; hours with little salary: 39.2; hours with
big salary 37.193548387096776
country: Yugoslavia; hours with little salary: 49.5; hours with big salary 41.6
```

[69]:
```
print('for Japan: hours with little salary: {}; hours with big
 salary {}'. ↵format(less_50_group['Japan'], more_50_group['Japan']))
```

```
for Japan: hours with little salary: 47.958333333333336; hours with
big salary 41.0
```

Part 2

[1]:
```
 import pandas as pd
import pandasql as ps
from datetime import datetime
```

[2]:
```
 android_devices = pd.read_csv('android_devices.csv')
user_device = pd.read_csv('user_device.csv')
user_usage = pd.read_csv('user_usage.csv')
print(android_devices.head())
print()
```

```
print(user_device.head())
print()
print(user_usage.head())
```

```
  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

   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

   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
```

[3]:
```
def query_duration_decorator(function_to_decorate):
    def qeury_duration(user_usage=None, user_device=None):
        start_time = datetime.now()

        if (user_usage is not None and user_device is not None):
            print(function_to_decorate(user_usage, user_device).head())
        else:
            print(function_to_decorate().head())

        end_time = datetime.now()
        print('Duration = {}'.format(end_time - start_time))
    return qeury_duration

#join pandasql
@query_duration_decorator
def pandasql_join(user_usage, user_device):
    join_query_ps = '''
            SELECT
                *
            FROM user_device
  INNER JOIN user_usage on user_usage.use_id = user_device.use_id
            '''
```

```
    return ps.sqldf(join_query_ps, locals())

#group pandasql
@query_duration_decorator
def pandasql_group(user_usage, user_device):
    group_query_ps = '''
            SELECT
                user_device.device,
                AVG(monthly_mb)
            FROM user_device
            JOIN user_usage on user_usage.use_id = user_device.use_id
            GROUP BY user_device.device
            '''
    return ps.sqldf(group_query_ps, locals())

#join pandas
@query_duration_decorator
def pandas_join():
    return user_device.merge(user_usage, how='inner', on='use_id')

#group pandas
@query_duration_decorator
def pandas_group():
    return user_device.merge(user_usage, how='inner', on='use_id').
 ↪groupby('device').monthly_mb.mean()
```

[4]:
```
pandasql_join(user_usage, user_device)
```

```
   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
Duration = 0:00:00.045000
```

[5]:
```
pandasql_group(user_usage, user_device)
```

```
   device AVG(monthly_mb)
0  A0001       15573.33
```

```
1 C6603        1557.33
2 D2303         519.12
3 D5503        1557.33
4 D5803        1557.33
Duration = 0:00:00.022035
```

[6]:
```
pandas_join()
```

```
   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
Duration = 0:00:00.018010
```

[7]:
```
pandas_group()
```

```
device
A0001   15573.33
C6603    1557.33
D2303     519.12
D5503    1557.33
D5803    1557.33
Name: monthly_mb, dtype: float64
Duration = 0:00:00.010999
```

As we can see queries with pandas faster than with pandasql

[ ]:
```
```