Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Лабораторная работа №2
по дисциплине
«Методы машинного обучения»
на тему
«Изучение библиотек обработки данных»

Выполнил:
студент группы ИУ5-22М
Смирнов А. И.

Москва — 2019 г.

1. Цель лабораторной работы

Изучить библиотеки обработки данных Pandas и PandaSQL [1].

2. Задание

Задание состоит из двух частей [1].

2.1. Часть 1

Требуется выполнить первое демонстрационное задание под названием «Exploratory data analysis with Pandas» со страницы курса mlcourse.ai [2].

2.2. Часть 2

Требуется выполнить следующие запросы с использованием двух различных библиотек — Pandas и PandaSQL [3, 4]:

- один произвольный запрос на соединение двух наборов данных,
- один произвольный запрос на группировку набора данных с использованием функций агрегирования.

Также требуется сравнить время выполнения каждого запроса в Pandas и PandaSQL.

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

[2]:
```
data = pd.read_csv('adult.data.csv')
data.head()
```

[2]:
```
   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
```

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

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

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

[4]:
```
36.85823043357163
```

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

[5]:
```
0.004207487485028101
```

[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.
```

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

[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)
```

[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
Race: Asian-Pac-Islander, sex: Male
count    693.000000
mean      39.073593
std       12.883944
min       18.000000
25%       29.000000
50%       37.000000
75%       46.000000
max       90.000000
Name: age, dtype: float64
Race: Black, sex: Female
count    1555.000000
mean       37.854019
std        12.637197
min        17.000000
25%        28.000000
50%        37.000000
75%        46.000000
max        90.000000
Name: age, dtype: float64
Race: Black, sex: Male
count    1569.000000
mean       37.682600
std        12.882612
min        17.000000
25%        27.000000
50%        36.000000
75%        46.000000
max        90.000000
Name: age, dtype: float64
Race: Other, sex: Female
count    109.000000
mean      31.678899
std       11.631599
min       17.000000
25%       23.000000
50%       29.000000
75%       39.000000
max       74.000000
Name: age, dtype: float64
```

```
Race: Other, sex: Male
count    162.000000
mean      34.654321
std       11.355531
min       17.000000
25%       26.000000
50%       32.000000
75%       42.000000
max       77.000000
Name: age, dtype: float64
Race: White, sex: Female
count    8642.000000
mean       36.811618
std        14.329093
min        17.000000
25%        25.000000
50%        35.000000
75%        46.000000
max        90.000000
Name: age, dtype: float64
Race: White, sex: Male
count    19174.000000
mean        39.652498
std         13.436029
min         17.000000
25%         29.000000
50%         38.000000
75%         49.000000
max         90.000000
Name: age, dtype: float64
```

[9]:

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

[9]:

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

[10]:

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

[10]:

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

[11]:

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

[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
```

[12]:
```python
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%
```

[13]:
```python
for (country, salary), sub_df in data.groupby(['native-country', 'salary']):
    print(country, salary, round(sub_df['hours-per-week'].mean(), 2))
```

```
? <=50K 40.16
? >50K 45.55
Cambodia <=50K 41.42
Cambodia >50K 40.0
Canada <=50K 37.91
Canada >50K 45.64
China <=50K 37.38
China >50K 38.9
Columbia <=50K 38.68
Columbia >50K 50.0
Cuba <=50K 37.99
Cuba >50K 42.44
Dominican-Republic <=50K 42.34
Dominican-Republic >50K 47.0
Ecuador <=50K 38.04
Ecuador >50K 48.75
El-Salvador <=50K 36.03
El-Salvador >50K 45.0
England <=50K 40.48
England >50K 44.53
France <=50K 41.06
France >50K 50.75
Germany <=50K 39.14
```

Germany >50K 44.98
Greece <=50K 41.81
Greece >50K 50.62
Guatemala <=50K 39.36
Guatemala >50K 36.67
Haiti <=50K 36.33
Haiti >50K 42.75
Holand-Netherlands <=50K 40.0
Honduras <=50K 34.33
Honduras >50K 60.0
Hong <=50K 39.14
Hong >50K 45.0
Hungary <=50K 31.3
Hungary >50K 50.0
India <=50K 38.23
India >50K 46.48
Iran <=50K 41.44
Iran >50K 47.5
Ireland <=50K 40.95
Ireland >50K 48.0
Italy <=50K 39.62
Italy >50K 45.4
Jamaica <=50K 38.24
Jamaica >50K 41.1
Japan <=50K 41.0
Japan >50K 47.96
Laos <=50K 40.38
Laos >50K 40.0
Mexico <=50K 40.0
Mexico >50K 46.58
Nicaragua <=50K 36.09
Nicaragua >50K 37.5
Outlying-US(Guam-USVI-etc) <=50K 41.86
Peru <=50K 35.07
Peru >50K 40.0
Philippines <=50K 38.07
Philippines >50K 43.03
Poland <=50K 38.17
Poland >50K 39.0
Portugal <=50K 41.94
Portugal >50K 41.5
Puerto-Rico <=50K 38.47
Puerto-Rico >50K 39.42
Scotland <=50K 39.44
Scotland >50K 46.67
South <=50K 40.16
South >50K 51.44
Taiwan <=50K 33.77
Taiwan >50K 46.8
Thailand <=50K 42.87

```
Thailand >50K 58.33
Trinadad&Tobago <=50K 37.06
Trinadad&Tobago >50K 40.0
United-States <=50K 38.8
United-States >50K 45.51
Vietnam <=50K 37.19
Vietnam >50K 39.2
Yugoslavia <=50K 41.6
Yugoslavia >50K 49.5
```

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

[14]:
```
native-country          ?   Cambodia     Canada      China   Columbia  \
salary
<=50K           40.164760  41.416667  37.914634  37.381818  38.684211
>50K            45.547945  40.000000  45.641026  38.900000  50.000000

native-country       Cuba  Dominican-Republic    Ecuador  El-Salvador  \
salary
<=50K           37.985714           42.338235  38.041667    36.030928
>50K            42.440000           47.000000  48.750000    45.000000

native-country    England  …   Portugal  Puerto-Rico   Scotland     South  \
salary                     …
<=50K           40.483333  …  41.939394    38.470588  39.444444  40.15625
>50K            44.533333  …  41.500000    39.416667  46.666667  51.43750

native-country     Taiwan   Thailand  Trinadad&Tobago  United-States  \
salary
<=50K           33.774194  42.866667        37.058824      38.799127
>50K            46.800000  58.333333        40.000000      45.505369

native-country    Vietnam  Yugoslavia
salary
<=50K           37.193548        41.6
>50K            39.200000        49.5

[2 rows x 42 columns]
```

[15]:
```python
user_usage = pd.read_csv('user_usage.csv')
user_device = pd.read_csv('user_device.csv')
```

[16]:
```python
user_usage.head()
```

[16]:
```
   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
```

[17]:
```
user_device.head()
```

[17]:
```
   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
```

[18]:
```
result = pd.merge(user_usage,
                  user_device[['use_id', 'platform', 'device']],
                  on='use_id')
result.head()
```

[18]:
```
   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
```

[19]:
```
result = pd.merge(user_usage,
                  user_device[['use_id', 'platform', 'device']],
                  on='use_id',
                  how='outer',
                  indicator=True)

result.iloc[[0, 1, 200,201,350,351]]
```

[19]:
```
     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
200                    28.79                   29.42     3114.67   23988
201                   616.56                   99.85     5414.14   24006
350                      NaN                     NaN         NaN   23050
351                      NaN                     NaN         NaN   23051

    platform     device      _merge
0    android   GT-I9505        both
1    android   SM-G930F        both
200      NaN        NaN   left_only
201      NaN        NaN   left_only
350      ios  iPhone7,2  right_only
351      ios  iPhone7,2  right_only
```

Список литературы

[1] Гапанюк Ю. Е. Лабораторная работа «Изучение библиотек обработки данных» [Электронный ресурс] // GitHub. — 2019. — Режим доступа: https://github.com/ugapanyuk/ml_course/wiki/LAB_PANDAS (дата обращения: 20.02.2019).

[2] Team The IPython Development. IPython 7.3.0 Documentation [Electronic resource] // Read the Docs. — 2019. — Access mode: https://ipython.readthedocs.io/en/stable/ (online; accessed: 20.02.2019).

[3] pandas 0.24.1 documentation [Electronic resource] // PyData. — 2019. — Access mode: http://pandas.pydata.org/pandas-docs/stable/ (online; accessed: 20.02.2019).

[4] yhat/pandasql: sqldf for pandas [Electronic resource] // GitHub. — 2017. — Access mode: https://github.com/yhat/pandasql (online; accessed: 22.02.2019).