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

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

Выполнил:
студент группы ИУ5-63Б
Елизаров О. О.

_____

Москва — 2020 г.

1. Цель лабораторной работы

Изучение библиотеки обработки данных Pandas

2. Задание

Выполнить первое демонстрационное задание “demo assignment” под названием “Exploratory data analysis with Pandas” со страницы курса https://mlcourse.ai/assignments

1. How many men and women (sex feature) are represented in this dataset?
2. What is the average age (age feature) of women?
3. What is the percentage of German citizens (native-country feature)?
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?
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)
6. 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.
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.
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?
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?

А потом необходимо выполнить следующие задания: 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

3. Ход выполнения лабораторной работы

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

%config InlineBackend.figure_format = 'svg'
```

```
from pylab import rcParams
rcParams['figure.figsize'] = 8, 5
```

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

[2]:
```
'1.0.3'
```

[3]:
```
df = pd.DataFrame([[1, 2], [1, 5], [7, 8]],
     index=['A', 'B', 'C'],
     columns=['X', 'Y'])
df
```

[3]:

	X	Y
A	1	2
B	1	5
C	7	8

[4]:
```
df2 = pd.DataFrame([[1, 20], [4, 50], [70, 80]],
     index=['A', 'B', 'C'],
     columns=['X', 'Y'])
df2
```

[4]:

	X	Y
A	1	20
B	4	50
C	70	80

[5]:
```
def union_frames_sql(df,df2):
    simple_query = '''
        SELECT
            X,
            Y
        FROM df
        UNION
        SELECT
            X,
            Y
        FROM df2

        ORDER BY Y desc
        LIMIT 10
        '''
    return ps.sqldf(simple_query, locals())
```

[6]:
```
start_time = time.time()
print (union_frames_sql(df,df2))
print("--- %s seconds ---" % (time.time() - start_time))
```

```
    X   Y
0  70  80
1   4  50
```

```
2   1  20
3   7   8
4   1   5
5   1   2
--- 0.017925024032592773 seconds ---
```

[7]:
```
def union_frames_pandas(df,df2):
    df3=pd.concat([df,df2])
    return df3[['X', 'Y']].sort_values(by ='Y', ascending = False)[:10]
```

[8]:
```
start_time = time.time()
print(union_frames_pandas(df,df2))
print("--- %s seconds ---" % (time.time() - start_time))
```

```
    X   Y
C  70  80
B   4  50
A   1  20
C   7   8
B   1   5
A   1   2
--- 0.003989458084106445 seconds ---
```

[9]:
```
def join_frames_sql(df,df2):
    simple_query = '''
        SELECT
            *
        FROM df
        JOIN
        df2
        ON
        df.X=df2.X
        ORDER BY Y desc
        LIMIT 10
        '''
    return ps.sqldf(simple_query, locals())
```

[10]:
```
start_time = time.time()
print(join_frames_sql(df,df2))
print("--- %s seconds ---" % (time.time() - start_time))
```

```
   X  Y  X   Y
0  1  5  1  20
1  1  2  1  20
--- 0.009974241256713867 seconds ---
```

[11]:
```
def join_frames_pandas(df,df2):
    df3 = pd.merge(df, df2, on='X', how='inner')
```

```
    return df3.sort_values(by ='Y_x', ascending = False)[:10]
```

[12]:
```
start_time = time.time()
print(join_frames_pandas(df,df2))
print("--- %s seconds ---" % (time.time() - start_time))
```

```
   X  Y_x  Y_y
1  1    5   20
0  1    2   20
--- 0.0059833526611328125 seconds ---
```

[13]:
```
def aggr_frames_sql(df,func):
    simple_query = '''
        SELECT
            '''+func+'''(Y)
        FROM df
        GROUP BY X
        ORDER BY Y desc
        LIMIT 10
        '''
    return ps.sqldf(simple_query, locals())
```

[15]:
```
start_time = time.time()
print(aggr_frames_sql(df,'avg'))
print("--- %s seconds ---" % (time.time() - start_time))
```

```
   avg(Y)
0     8.0
1     3.5
--- 0.008976221084594727 seconds ---
```

[16]:
```
def avg_frame_pandas(df):
    return df.groupby('X').mean()
```

[18]:
```
start_time = time.time()
print(avg_frame_pandas(df))
print("--- %s seconds ---" % (time.time() - start_time))
```

```
     Y
X
1  3.5
7  8.0
--- 0.0039904117584228516 seconds ---
```