Московский государственный технический университет им. Н.Э. Баумана
Факультет «Информатика и системы управления»
Кафедра «Автоматизированные системы обработки информации и управления»

Отчет по лабораторной работе № 2
"Изучение библиотек обработки данных"

По курсу
«Методы машинного обучения»

Выполнила:
Шаххуд Ф.М.
Студентка группы ИУ5И-12М

Москва, 2020

Часть 1.

```
import numpy as np
import pandas as pd

data = pd.read_csv('adult.data.csv')
data.head()
```

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relatio
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Hu
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Hu
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	

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

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

```
Male      21790
Female    10771
Name: sex, dtype: int64
```

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

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

```
36.85823043357163
```

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

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

```
0.004207487485028101
```

4. What are the mean and standard deviation of age for those who earn more than 50K per yea
5. and those who earn less than 50K per year?

```
print(data.loc[data['salary']=='>50K','age'].mean())
print(data.loc[data['salary']=='>50K','age'].std())
print(data.loc[data['salary']=='<=50K','age'].mean())
print(data.loc[data['salary']=='<=50K','age'].std())
```

```
44.24984058155847
10.51902771985177
36.78373786407767
14.020088490824813
```

6. Is it true that people who earn more than 50K have at least high school education? (educatio
   school, Assoc-acdm, Assoc-voc, Masters or Doctorate feature)

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

```
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)
```

7. Display statistics of age for each race (race feature) and each gender. Use groupby() and des
   maximum age of men of Amer-Indian-Eskimo race.

```
for element in data.groupby(['race','sex']):
  print(element[0])
  print(element[1]['age'].describe())
```

8. Among whom the proportion of those who earn a lot(>50K) is more: among married or single
   feature)? Consider married those who have a marital-status starting with Married (Married-c
   spouse-absent or Married-AF-spouse), the rest are considered bachelors.

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

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

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

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

9. What is the maximum number of hours a person works per week (hours-per-week feature)? H

```
maximum=data['hours-per-week'].max()
working=data[data['hours-per-week']==maximum].shape[0]
high=data[(data['hours-per-week']==maximum) & (data['salary']=='>50K')].shape[0]
print(maximum)
print(working)
print(high/working)
```

```
99
85
0.29411764705882354
```

10. Count the average time of work (hours-per-week) for those who earn a little and a lot (salary) (native-country). What will these be for Japan?

```
for element in data.groupby(['native-country','salary']):
  print (element[0])
  print(element[1]['hours-per-week'].mean())
```

Часть 2.

```
!pip install pandasql
```

```
Collecting pandasql
  Downloading https://files.pythonhosted.org/packages/6b/c4/ee4096ffa2eeeca0c7
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: pandas in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: sqlalchemy in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-p
Requirement already satisfied: python-dateutil>=2.6.1 in /usr/local/lib/python
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-packa
Building wheels for collected packages: pandasql
  Building wheel for pandasql (setup.py) ... done
  Created wheel for pandasql: filename=pandasql-0.7.3-cp36-none-any.whl size=2
  Stored in directory: /root/.cache/pip/wheels/53/6c/18/b87a2e5fa8a82e9c026311
Successfully built pandasql
Installing collected packages: pandasql
Successfully installed pandasql-0.7.3
```

```
%matplotlib inline
import pandas as pd
import pandasql as ps
from datetime import datetime
import seaborn
import matplotlib.pyplot as plt
```

```
%config InlineBackend.figure_format = 'svg'
from pylab import rcParams
```

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

user_usage_data=pd.read_csv("user_usage.csv")
user_device_data=pd.read_csv("user_device.csv")
device_data=pd.read_csv("android_devices.csv")

user_usage_data.head()
```

	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

```
user_device_data.head()
```

	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

```
device_data.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

```
device_data.rename(columns={"Retail Branding": "manufacturer"}, inplace=True)

def connect_pandasql(user_usage_data):
  query="""SELECT user_usage.* , user_device.platform,
  user_device.device FROM user_usage_data as user_usage
   inner join user_device_data as user_device on
    user_usage.use_id=user_device.use_id;"""
  result pandasql=ps.sqldf(query)
```

```
  query="""SELECT result_pandasql.* , device.'manufacturer',
  device.Model FROM result_pandasql as result_pandasql
   inner join device_data as device on
    result_pandasql.device=device.Model;"""
  return ps.sqldf(query)

result_pandasql=connect_pandasql(user_usage_data)
result_pandasql.head()
```

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id	platfo
0	21.97	4.82	1557.33	22787	andro
1	1710.08	136.88	7267.55	22788	andro
2	1710.08	136.88	7267.55	22789	andro
3	94.46	35.17	519.12	22790	andro
4	71.59	79.26	1557.33	22792	andro

```
def connect_pandas(user_usage_data):
  result_pandas = pd.merge(user_usage_data,
                 user_device_data[['use_id', 'platform', 'device']],
                 on='use_id')
  return  pd.merge(result_pandas,
                   device_data[['manufacturer', 'Model']],
                  left_on='device',
                  right_on='Model',
                  how='inner')

result_pandas=connect_pandas(user_usage_data)
result_pandas.head()
```

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id	platfo
0	21.97	4.82	1557.33	22787	andro
1	69.80	14.70	25955.55	22801	andro
2	249.26	253.22	1557.33	22875	andro
3	249.26	253.22	1557.33	22876	andro
4	83.46	114.06	3114.67	22880	andro

```
def group_pandasql(result_pandasql):
  query="""SELECT manufacturer,AVG(outgoing_mins_per_month),AVG(outgoing_sms_per_mo
      AVG(monthly_mb),Count(use_id) FROM result_pandasql GROUP BY manufacturer;
      """
  return ps.sqldf(query)


aggre_pandasql=group_pandasql(result_pandasql);
aggre_pandasql
```

	manufacturer	AVG(outgoing_mins_per_month)	AVG(outgoing_sms_per_month)	A
0	HTC	299.842955	93.059318	
1	Huawei	81.526667	9.500000	
2	LGE	111.530000	12.760000	
3	Lava	60.650000	261.900000	
4	Lenovo	215.920000	12.930000	
5	Motorola	95.127500	65.666250	
6	OnePlus	354.855000	48.330000	
7	Samsung	191.010093	92.390463	
8	Sony	177.315625	40.176250	
9	Vodafone	42.750000	46.830000	
10	ZTE	42.750000	46.830000	

```
def group_pandas(result_pandas):
  return result_pandas.groupby("manufacturer").agg({
        "outgoing_mins_per_month": "mean",
        "outgoing_sms_per_month": "mean",
        "monthly_mb": "mean",
        "use_id": "count"
    })

aggre_pandas=group_pandas(result_pandas)
aggre_pandas
```

manufacturer	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use
HTC	299.842955	93.059318	5144.077955	
Huawei	81.526667	9.500000	1561.226667	
LGE	111.530000	12.760000	1557.330000	
Lava	60.650000	261.900000	12458.670000	
Lenovo	215.920000	12.930000	1557.330000	
Motorola	95.127500	65.666250	3946.500000	
OnePlus	354.855000	48.330000	6575.410000	
Samsung	191.010093	92.390463	4017.318889	
Sony	177.315625	40.176250	3212.000625	
Vodafone	42.750000	46.830000	5191.120000	
ZTE	42.750000	46.830000	5191.120000	

```
import time

def count_mean_time(func, params, N =10):
    total_time = 0
    for i in range(N):
        time1 = time.time()
        tmp=func(params)
        time2 = time.time()
        total_time += (time2 - time1)
    return total_time/N


pandasql_time = count_mean_time(connect_pandasql,user_usage_data)
pandas_time = count_mean_time(connect_pandas,user_usage_data)
print("Mean time for executing connecting tables using:")
print("pandasql is {} - pandas is {}".format('%.4f'%pandasql_time,'%.4f'%pandas_tir
```

```
Mean time for executing connecting tables using:
pandasql is 0.1721 - pandas is 0.0093
```

```
pandasql_time = count_mean_time(group_pandasql,result_pandasql)
pandas_time = count_mean_time(group_pandas,result_pandas)
print("Mean time for executing aggregation of tables using:")
print("pandasql is {} - pandas is {}".format('%.4f'%pandasql_time,'%.4f'%pandas_tir
```

```
Mean time for executing aggregation of tables using:
pandasql is 0.0227 - pandas is 0.0026
```

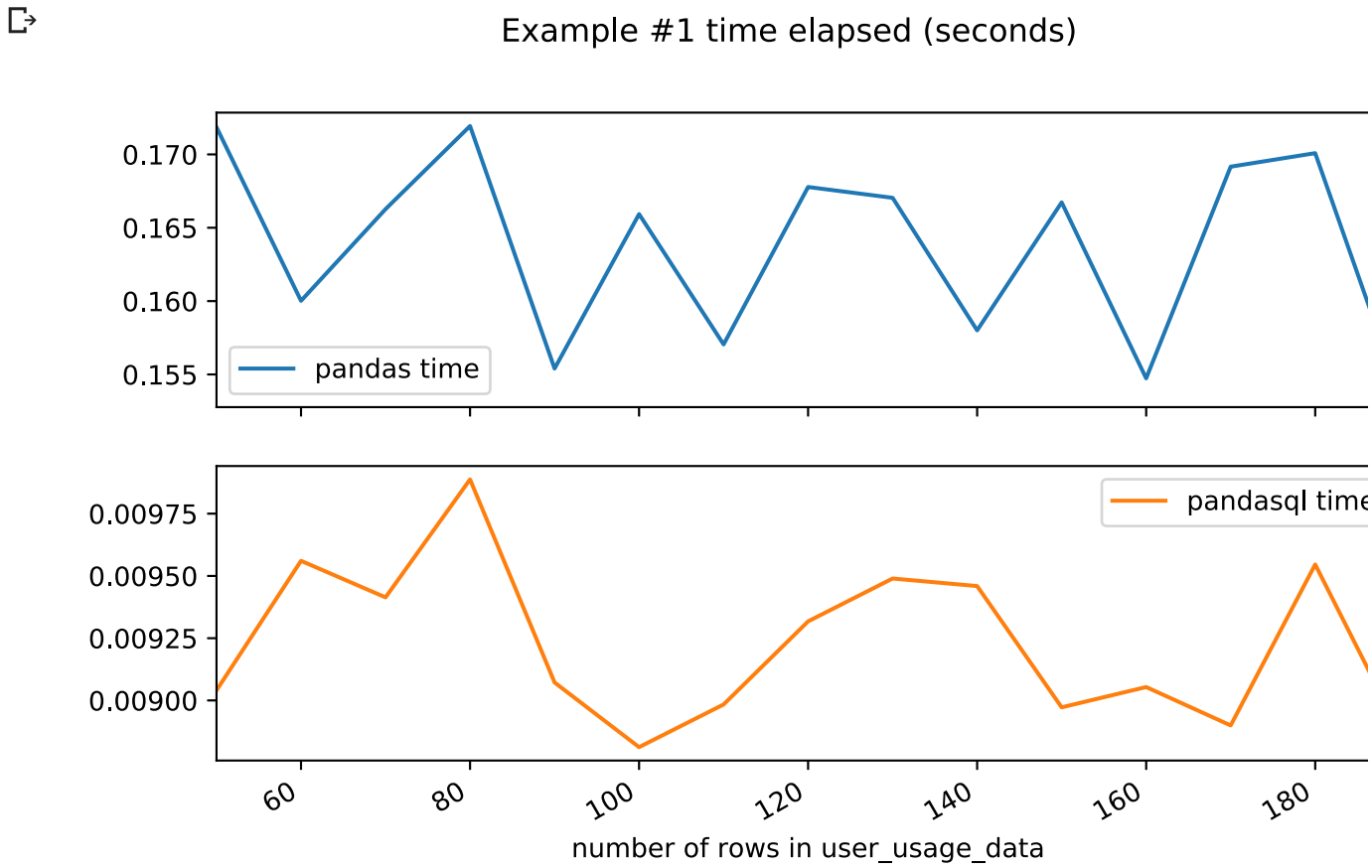
```
def count_mean_time2(func, params, N =5):
    total_time = 0
    for i in range(N):
```

```
        time1 = time.time()
        tmp_df = func(params)
        time2 = time.time()
        total_time += (time2 - time1)
    return total_time/N


ex1_times = []
for count in range(50, 200, 10):
    pandasql_time = count_mean_time2(connect_pandasql, user_usage_data[:count])
    pandas_time = count_mean_time2(connect_pandas, user_usage_data[:count])
    ex1_times.append({'count': count, 'pandasql_time': pandasql_time, 'pandas_time

ex1_times_df = pd.DataFrame(ex1_times)
ex1_times_df.columns = ['number of rows in user_usage_data', 'pandas time', 'pandas
ex1_times_df = ex1_times_df.set_index('number of rows in user_usage_data')

ax = ex1_times_df.plot(title = 'Example #1 time elapsed (seconds)', subplots = True
```




```
ex1_times = []
for count in range(50, 200, 10):
    pandasql_time = count_mean_time2(group_pandasql, result_pandasql[:count])
    pandas_time = count_mean_time2(group_pandas, result_pandas[:count])
    ex1_times.append({'count': count, 'pandasql_time': pandasql_time, 'pandas_time

ex1_times_df = pd.DataFrame(ex1_times)
ex1_times_df.columns = ['number of rows in user_usage_data', 'pandas time', 'pandas
ex1_times_df = ex1_times_df.set_index('number of rows in user_usage_data')
```

```
ax = ex1_times_df.plot(title = 'Example #1 time elapsed (seconds)', subplots = True
```