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

«Библиотеки обработки данных для языка Python»

По курсу
“ Методы машинного обучения ”

Выполнил:
Али Диб А.Ж.
Студент группы ИУ5-22М

Москва, 2020

```
import numpy as np
import pandas as pd
```

```
data=pd.read_csv('adult.data.txt')
```

```
data.head()
```

	age	workclass	fnlwgt	education	education-num	marital-status	occu
0	39	State-gov	77516	Bachelors	13	Never-married	Adr
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-ma
2	38	Private	215646	HS-grad	9	Divorced	Handlers-
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-

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

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

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

```
36.85823043357163
```

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

```
36.85823043357163
```

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

```
0.004207487485028101
```

```
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.".for
    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.
```

```
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)
```

```
data.groupby(['race','sex']).first()
```

		age	workclass	fnlwgt	education	education-num	marital-status	occupa
race	sex							
Amer-Indian-Eskimo	Female	35	Private	153790	Some-college	10	Never-married	S
	Male	34	Private	245487	7th-8th	4	Married-civ-spouse	Trans mo
Asian-Pac-Islander	Female	30	Private	117747	HS-grad	9	Married-civ-spouse	S
	Male	30	State-gov	141297	Bachelors	13	Married-civ-spouse	Prof-spe
Black	Female	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-spe
	Male	53	Private	234721	11th	7	Married-civ-spouse	Hand clea

```
for element in data.groupby(['race','sex']):
    print("Race:{0},Sex:{1}".format(element[0][0],element[0][1]))
    print(element[1]['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
```

```
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
```

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

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

```
data['marital-status']
```

```
0             Never-married
1        Married-civ-spouse
2                  Divorced
3        Married-civ-spouse
4        Married-civ-spouse
                 ...
32556    Married-civ-spouse
32557    Married-civ-spouse
32558               Widowed
32559         Never-married
32560    Married-civ-spouse
Name: marital-status, Length: 32561, dtype: object
```

```
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=data[(data['hours-per-week']==max_load) & (data['salary']=='>50K')].shap
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%
```

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

```
Greece   <=50K : 41.80952380952381
Greece   >50K : 50.625
Guatemala   <=50K : 39.36065573770492
Guatemala   >50K : 36.666666666666664
Haiti   <=50K : 36.325
Haiti   >50K : 42.75
Holand-Netherlands   <=50K : 40.0
Honduras   <=50K : 34.333333333333336
Honduras   >50K : 60.0
Hong   <=50K : 39.142857142857146
Hong   >50K : 45.0
Hungary   <=50K : 31.3
Hungary   >50K : 50.0
India   <=50K : 38.233333333333334
India   >50K : 46.475
Iran   <=50K : 41.44
Iran   >50K : 47.5
Ireland   <=50K : 40.94736842105263
Ireland   >50K : 48.0
Italy   <=50K : 39.625
Italy   >50K : 45.4
Jamaica   <=50K : 38.23943661971831
Jamaica   >50K : 41.1
Japan   <=50K : 41.0
Japan   >50K : 47.958333333333336
Laos   <=50K : 40.375
Laos   >50K : 40.0
Mexico   <=50K : 40.00327868852459
Mexico   >50K : 46.57575757575758
Nicaragua   <=50K : 36.09375
Nicaragua   >50K : 37.5
Outlying-US(Guam-USVI-etc)   <=50K : 41.857142857142854
Peru   <=50K : 35.06896551724138
Peru   >50K : 40.0
Philippines   <=50K : 38.065693430656935
Philippines   >50K : 43.032786885245905
Poland   <=50K : 38.166666666666664
Poland   >50K : 39.0
Portugal   <=50K : 41.93939393939394
Portugal   >50K : 41.5
Puerto-Rico   <=50K : 38.470588235294116
Puerto-Rico   >50K : 39.416666666666664
Scotland   <=50K : 39.44444444444444
Scotland   >50K : 46.666666666666664
South   <=50K : 40.15625
South   >50K : 51.4375
Taiwan   <=50K : 33.774193548387096
Taiwan   >50K : 46.8
Thailand   <=50K : 42.86666666666667
Thailand   >50K : 58.333333333333336
Trinadad&Tobago   <=50K : 37.05882352941177
Trinadad&Tobago   >50K : 40.0
United-States   <=50K : 38.79912723305605
United-States   >50K : 45.50536884674383
Vietnam   <=50K : 37.193548387096776
Vietnam   >50K : 39.2
Yugoslavia   <=50K : 41.6
Yugoslavia   >50K : 49.5
```

```
import pandas as pd
import pandasql as psql

user_usage = pd.read_csv("user_usage.csv")
user_device = pd.read_csv("user_device.csv")
devices = pd.read_csv("android_devices.csv")
devices.rename(columns={"Retail Branding": "manufacturer"}, inplace=True)

user_usage.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.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

```
devices.head()
```

	manufacturer	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

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

def pd_merge(user_usage):
  result= pd.merge(user_usage , user_device[['use_id' , 'platform' , 'device']] , h
  return pd.merge(result
```

```
  return pd.merge(result,
                  devices[['manufacturer', 'Model']],
                  left_on='device',
                  right_on='Model',
                  how='inner')


result=pd_merge(user_usage)
result.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 pd_agg():
  return result.groupby("manufacturer").agg({
    "outgoing_mins_per_month":"mean",
    "outgoing_sms_per_month": "mean",
        "monthly_mb": "mean",
        "use_id": "count"})


pd_agg()
```

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	

```
def pd_sql_merge(user_usage):
  temp= psql.sqldf("select i.*, e.platform, e.device from  user_usage i inner join
  return psql.sqldf("select i.*, e.manufacturer, e.model from temp i inner join dev
```

```
res=pd_sql_merge(user_usage)
pd_sql_merge(user_usage).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 pd_sql_agg():
  return psql.sqldf("select manufacturer,avg(outgoing_mins_per_month), avg(outgoing
```

```
pd_sql_agg().head()
```

	manufacturer	avg(outgoing_mins_per_month)	avg(outgoing_sms_per_month)	av
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	

```
import time

def count_mean_time(func, params, N =5):
    total_time = 0
    for i in range(N):
        time1 = time.time()
        tmp_df = func(params[0])
        time2 = time.time()
        total_time += (time2 - time1)
    return total_time/N


ex2_times = []
for count in range(50, 200, 10):
    pandasql_time = count_mean_time(pd_sql_merge, [user_usage[:count]])
    pandas_time = count_mean_time(pd_merge, [user_usage[:count]])
    ex2_times.append({'count': count, 'pandasql_time': pandasql_time, 'pandas_time

ex2_times_df = pd.DataFrame(ex2_times)

ex2_times_df.columns = ['number of rows in user_usage', 'pandas time', 'pandasql ti
ex2_times_df = ex2_times_df.set_index('number of rows in user_usage')
```

```
ax = ex2_times_df.plot(title = 'Example #2 time elapsed (seconds)', subplots = True
```