Московский государственный технический университет имени Н.Э.Баумана

Кафедра «Системы обработки информации и управления»

ОТЧЕТ

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

Выполнил:

Студент ИУ5-24М

Гаврилюк А.Г.

Москва, 2020

In [30]:

```
import numpy as np
import pandas as pd
pd.set_option('display.max.columns', 100)
import matplotlib.pyplot as plt
import seaborn as sns
```

In [31]:

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

Out[31]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba

In [33]:

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

Out[33]:

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

In [34]:

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

Out[34]:

```
36.85823043357163
```

In [37]:

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

Out[37]:

```
0.004207487485028101
```

In [38]:

```
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.
```

In [39]:

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

Out[39]:

```
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)
```

In [40]:

```
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
```

In [41]:

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

Out[41]:

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

In [42]:

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

Out[42]:

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

In [43]:

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

Out[43]:

```
Married-civ-spouse       14976
Never-married            10683
Divorced                  4443
Separated                 1025
Separated                1025
Widowed                   993
Married-spouse-absent     418
Married-AF-spouse          23
Name: marital-status, dtype: int64
```

In [44]:

```
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%
```

In [46]:

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

Out[46]:

native-country	?	Cambodia	Canada	China	Columbia	Cuba	Dominican-Republic	Ecuador	El-Salvador	England	France
salary											
<=50K	40.164760	41.416667	37.914634	37.381818	38.684211	37.985714	42.338235	38.041667	36.030928	40.483333	41.058824
>50K	45.547945	40.000000	45.641026	38.900000	50.000000	42.440000	47.000000	48.750000	45.000000	44.533333	50.750000

In [48]:

```
user_usage = pd.read_csv('data/user_usage.csv')
user_device = pd.read_csv('data/user_device.csv')
devices = pd.read_csv('data/android_devices.csv')
```

In [49]:

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

Out[49]:

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id	platform	device
0	21.97	4.82	1557.33	22787	android	GT-I9505
1	1710.08	136.88	7267.55	22788	android	SM-G930F
2	1710.08	136.88	7267.55	22789	android	SM-G930F
3	94.46	35.17	519.12	22790	android	D2303
4	71.59	79.26	1557.33	22792	android	SM-G361F

In [51]:

```
import pandasql as ps
from pandasql import sqldf
from datetime import datetime
import time
```

In [52]:

```
tic = time.perf_counter()
tutorial = pd.merge(user_usage,
                    user_device[['use_id', 'platform', 'device']],
                    on='use_id')
toc = time.perf_counter()
print(f"Смержено за: {toc - tic:0.4f} seconds")
```

```
Смержено за: 0.0053 seconds
```

In [53]:

```
pysqldf = lambda q: sqldf(q, globals())
q = """
SELECT * FROM user_usage, user_device
WHERE user_usage.use_id = user_device.use_id;
"""
tic = time.perf_counter()
joined = pysqldf(q)
toc = time.perf_counter()
print(f"Смержено за: {toc - tic:0.4f} seconds")
```

```
Смержено за: 0.0293 seconds
```

In [54]:

```
joined.head()
```

Out[54]:

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id	use_id	user_id	platform	platform_version	device	use_ty
0	21.97	4.82	1557.33	22787	22787	12921	android	4.3	GT-I9505	
1	1710.08	136.88	7267.55	22788	22788	28714	android	6.0	SM-G930F	
2	1710.08	136.88	7267.55	22789	22789	28714	android	6.0	SM-G930F	
3	94.46	35.17	519.12	22790	22790	29592	android	5.1	D2303	
4	71.59	79.26	1557.33	22792	22792	28217	android	5.1	SM-G361F	

In [55]:

```
joined.describe()
```

Out[55]:

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id	use_id	user_id	platform_version
count	159.000000	159.000000	159.000000	159.000000	159.000000	159.000000	159.000000
mean	203.331509	87.978742	4180.378616	22922.327044	22922.327044	25960.918239	5.554717
std	248.660581	92.386434	5216.463795	76.511974	76.511974	6275.640431	0.828656
min	0.500000	0.250000	0.000000	22787.000000	22787.000000	2873.000000	4.100000
25%	70.070000	22.855000	1557.330000	22861.500000	22861.500000	24683.500000	5.000000
50%	137.060000	62.850000	2076.450000	22931.000000	22931.000000	29366.000000	6.000000
75%	241.035000	119.675000	5191.120000	22986.500000	22986.500000	29673.000000	6.000000
max	1710.080000	540.600000	31146.670000	23053.000000	23053.000000	29725.000000	10.100000

In [56]:

```
joined.groupby("platform_version")["outgoing_sms_per_month"].describe()
```

Out[56]:

platform_version	count	mean	std	min	25%	50%	75%	max
4.1	5.0	102.328000	51.393475	26.94	91.7600	91.760	150.5900	150.59
4.2	1.0	24.080000	NaN	24.08	24.0800	24.080	24.0800	24.08
4.3	3.0	66.366667	82.035137	4.82	19.8000	34.780	97.1400	159.50
4.4	17.0	108.699412	131.771975	7.67	7.6700	22.360	261.3300	327.33
5.0	17.0	99.321176	83.228036	5.83	60.8300	69.200	114.0600	273.75
5.1	23.0	63.606957	38.369532	4.64	41.2050	52.470	79.2600	162.39
6.0	88.0	86.057841	86.776242	0.25	22.2100	72.485	136.8800	435.29
7.0	2.0	39.035000	42.659752	8.87	23.9525	39.035	54.1175	69.20
7.1	1.0	15.380000	NaN	15.38	15.3800	15.380	15.3800	15.38
9.3	1.0	540.600000	NaN	540.60	540.6000	540.600	540.6000	540.60
10.1	1.0	47.350000	NaN	47.35	47.3500	47.350	47.3500	47.35

In [ ]: