Московский государственный технический университет им. Н.Э. Баумана.

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Технологии машинного обучения»

Отчет по лабораторной работе №2:
«Изучение библиотек обработки данных»

Выполнил:
студент группы ИУ5-62
Андреев Артем

Подпись и дата:

Проверил:

Подпись и дата:

Москва, 2019 г.

mlcourse.ai (https://mlcourse.ai) – Open Machine Learning Course

Author:Yury Kashnitskiy (https://yorko.github.io). Translated and edited by Sergey Isaev (https://www.linkedin.com/in/isvforall/), Artem Trunov (https://www.linkedin.com/in/datamove/), Anastasia Manokhina (https://www.linkedin.com/in/anastasiamanokhina/), and Yuanyuan Pao (https://www.linkedin.com/in/yuanyuanpao/)

Assignment #1 (demo)

Exploratory data analysis with Pandas

In this task you should use Pandas to answer a few questions about the Adult (https://archive.ics.uci.edu/ml/datasets/Adult) dataset. (You don't have to download the data – it's already in the repository). Choose the answers in the web-form (https://docs.google.com/forms/d/1uY7MpI2trKx6FLWZte0uVh3ULV4Cm_tDud0VDFGCOKg).

Unique values of all features (for more information, please see the links above):

- `age` : continuous.
- `workclass` : Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
- `fnlwgt` : continuous.
- `education` : Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
- `education-num` : continuous.
- `marital-status` : Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
- `occupation` : Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
- `relationship` : Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
- `race` : White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
- `sex` : Female, Male.
- `capital-gain` : continuous.
- `capital-loss` : continuous.
- `hours-per-week` : continuous.
- `native-country` : United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.
- `salary` : >50K,<=50K

In [2]:

```
import numpy as np
import pandas as pd
pd.set_option('display.max.columns', 100)
# to draw pictures in jupyter notebook
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
# we don't like warnings
# you can comment the following 2 lines if you'd like to
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
data = pd.read_csv('data/adult.data.csv')
data.head()
```

Out[3]:

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

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

In [4]:
```
data['sex'].value_counts()
```

Out[4]:
```
 Male      21790
 Female    10771
Name: sex, dtype: int64
```

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

In [5]:
```
data[data['sex'] == ' Female']['age'].mean()
```

Out[5]:
```
36.85823043357163
```

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

In [6]:
```
float(data[data['native-country'] == ' Germany'].shape[0]) / (data.shape[0]) * 100
```

Out[6]:
```
0.42074874850281013
```

4-5. 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?

In [17]:
```
print('mean >50K:    ', data[data['salary'] == ' >50K']['age'].mean())
print('stddev >50K: ', data[data['salary'] == ' >50K']['age'].std())
print('mean <=50K:   ', data[data['salary'] == ' <=50K']['age'].mean())
print('stddev <=50K:', data[data['salary'] == ' <=50K']['age'].std())
```

```
mean >50K:     44.24984058155847
stddev >50K:  10.519027719851826
mean <=50K:   36.78373786407767
stddev <=50K: 14.02008849082488
```

6. 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)

In [36]:
```
all_gt_50K = data.loc[data['salary'] == ' >50K'].shape[0]
gt_50K_and_education = data.loc[(data['salary'] == ' >50K') & (data['education'].isin(['Bachelors', 'Prof-school', 'Assoc-acdm', 'Assoc-voc', 'Masters', 'Doctorate']))].shape[0]
print('yes, it\'s true' if all_gt_50K == gt_50K_and_education else 'no')
```

```
no
```

7. 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.

In [9]:
```
data.groupby('race')['age'].describe()
```

Out[9]:

race	count	mean	std	min	25%	50%	75%	max
Amer-Indian-Eskimo	311.0	37.173633	12.447130	17.0	28.0	35.0	45.5	82.0
Asian-Pac-Islander	1039.0	37.746872	12.825133	17.0	28.0	36.0	45.0	90.0
Black	3124.0	37.767926	12.759290	17.0	28.0	36.0	46.0	90.0
Other	271.0	33.457565	11.538865	17.0	25.0	31.0	41.0	77.0
White	27816.0	38.769881	13.782306	17.0	28.0	37.0	48.0	90.0

In [10]:
```
data.groupby('sex')['age'].describe()
```

Out[10]:

sex	count	mean	std	min	25%	50%	75%	max
Female	10771.0	36.858230	14.013697	17.0	25.0	35.0	46.0	90.0
Male	21790.0	39.433547	13.370630	17.0	29.0	38.0	48.0	90.0

In [11]:
```
data.groupby(['race', 'sex'])['age'].describe()
```

Out[11]:

race	sex	count	mean	std	min	25%	50%	75%	max
Amer-Indian-Eskimo	**Female**	119.0	37.117647	13.114991	17.0	27.0	36.0	46.00	80.0
	Male	192.0	37.208333	12.049563	17.0	28.0	35.0	45.00	82.0
Asian-Pac-Islander	**Female**	346.0	35.089595	12.300845	17.0	25.0	33.0	43.75	75.0
	Male	693.0	39.073593	12.883944	18.0	29.0	37.0	46.00	90.0
Black	**Female**	1555.0	37.854019	12.637197	17.0	28.0	37.0	46.00	90.0
	Male	1569.0	37.682600	12.882612	17.0	27.0	36.0	46.00	90.0
Other	**Female**	109.0	31.678899	11.631599	17.0	23.0	29.0	39.00	74.0
	Male	162.0	34.654321	11.355531	17.0	26.0	32.0	42.00	77.0
White	**Female**	8642.0	36.811618	14.329093	17.0	25.0	35.0	46.00	90.0
	Male	19174.0	39.652498	13.436029	17.0	29.0	38.0	49.00	90.0

In [12]:
```
data.groupby(['race', 'sex'])['age'].describe().loc[' Amer-Indian-Eskimo'].loc[' Male', 'max']
```

Out[12]: 82.0

8. 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.

In [13]:
```
all_gt_50k = data[(data['salary'] == ' >50K') & (data['sex'] == ' Male')].shape[0]
married_and_salary_gt_50k = data[(data['salary'] == ' >50K') &
                                 (data['sex'] == ' Male') &
                                 (data['marital-status'].isin([' Married-civ-spouse',
                                                              ' Married-spouse-absent',
                                                              ' Married-AF-spouse']))].shape[0]

bachelors = all_gt_50k - married_and_salary_gt_50k
print(all_gt_50k, married_and_salary_gt_50k, bachelors)
print('married men' if married_and_salary_gt_50k > bachelors else 'bachelors men' if married_and_salary_gt_50
k != bachelors else 'same')
```

```
6662 5965 697
married men
```

9. 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?

In [27]:
```
print("max hours-per-week:                    ", data['hours-per-week'].max())
max_hours_per_week_data = data[data['hours-per-week'] == 99]
print("people with max hours-per-week value:", max_hours_per_week_data.shape[0])
print("earn >50K among them:                  ", max_hours_per_week_data[data['salary'] == ' >50K'].shape[0])
```

```
max hours-per-week:                   99
people with max hours-per-week value: 85
earn >50K among them:                 25
```

10. 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?

In [32]:
```
data.groupby(['native-country', 'salary'])['hours-per-week'].describe()['mean']
```

```
Out[32]: native-country              salary
          ?                           <=50K     40.164760
                                      >50K      45.547945
          Cambodia                    <=50K     41.416667
                                      >50K      40.000000
          Canada                      <=50K     37.914634
                                      >50K      45.641026
          China                       <=50K     37.381818
                                      >50K      38.900000
          Columbia                    <=50K     38.684211
                                      >50K      50.000000
          Cuba                        <=50K     37.985714
                                      >50K      42.440000
          Dominican-Republic          <=50K     42.338235
                                      >50K      47.000000
          Ecuador                     <=50K     38.041667
                                      >50K      48.750000
          El-Salvador                 <=50K     36.030928
                                      >50K      45.000000
          England                     <=50K     40.483333
                                      >50K      44.533333
          France                      <=50K     41.058824
                                      >50K      50.750000
          Germany                     <=50K     39.139785
                                      >50K      44.977273
          Greece                      <=50K     41.809524
                                      >50K      50.625000
          Guatemala                   <=50K     39.360656
                                      >50K      36.666667
          Haiti                       <=50K     36.325000
                                      >50K      42.750000
                                                   ...
          Mexico                      >50K      46.575758
          Nicaragua                   <=50K     36.093750
                                      >50K      37.500000
          Outlying-US(Guam-USVI-etc)  <=50K     41.857143
          Peru                        <=50K     35.068966
                                      >50K      40.000000
          Philippines                 <=50K     38.065693
                                      >50K      43.032787
          Poland                      <=50K     38.166667
```

```
                                >50K      39.000000
 Portugal                       <=50K     41.939394
                                >50K      41.500000
 Puerto-Rico                    <=50K     38.470588
                                >50K      39.416667
 Scotland                       <=50K     39.444444
                                >50K      46.666667
 South                          <=50K     40.156250
                                >50K      51.437500
 Taiwan                         <=50K     33.774194
                                >50K      46.800000
 Thailand                       <=50K     42.866667
                                >50K      58.333333
 Trinadad&Tobago                <=50K     37.058824
                                >50K      40.000000
 United-States                  <=50K     38.799127
                                >50K      45.505369
 Vietnam                        <=50K     37.193548
                                >50K      39.200000
 Yugoslavia                     <=50K     41.600000
                                >50K      49.500000
Name: mean, Length: 82, dtype: float64
```

In [35]:
```
data.groupby(['native-country', 'salary'])['hours-per-week'].describe()['mean'].loc[' Japan']
```

Out[35]:
```
salary
 <=50K    41.000000
 >50K     47.958333
Name: mean, dtype: float64
```

In [ ]:

In [145]:
```
import numpy as np
import pandas as pd
pd.set_option('display.max.columns', 100)
pd.set_option('display.max_rows', 500)
import pandasql as pds
# to draw pictures in jupyter notebook
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
# we don't like warnings
# you can comment the following 2 lines if you'd like to
import warnings
warnings.filterwarnings('ignore')
```

In [146]:
```
user_usage = pd.read_csv("data/user_usage.csv")
user_device = pd.read_csv("data/user_device.csv")
android_devices = pd.read_csv("data/android_devices.csv")
```

In [147]:
```
user_usage.head()
```

Out[147]:

	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

In [148]:
```
user_device.head()
```
Out[148]:

	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

In [149]:
```
android_devices = android_devices.rename(index=str, columns={'Device': 'device'})
android_devices.head()
```
Out[149]:

	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

Merge using pandas

In [150]:
```python
usage_and_device = pd.merge(user_usage, user_device[['use_id', 'device']], on='use_id')
print('Total:', usage_and_device.shape[0])
usage_and_device.head()
```

```
Total: 159
```

Out[150]:

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

In [151]:
```python
usage_and_device_android = pd.merge(usage_and_device,
                                    android_devices[['Model', 'Retail Branding']],
                                    left_on='device', right_on='Model').drop_duplicates()
print('Total:', usage_and_device_android.shape[0])
usage_and_device_android.head()
```

```
Total: 150
```

Out[151]:

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id	device	Model	Retail Branding
0	21.97	4.82	1557.33	22787	GT-I9505	GT-I9505	Samsung
1	69.80	14.70	25955.55	22801	GT-I9505	GT-I9505	Samsung
2	249.26	253.22	1557.33	22875	GT-I9505	GT-I9505	Samsung
3	249.26	253.22	1557.33	22876	GT-I9505	GT-I9505	Samsung
4	83.46	114.06	3114.67	22880	GT-I9505	GT-I9505	Samsung

In [160]:
```
usage_and_device_android.groupby('Retail Branding').agg({
    "outgoing_mins_per_month": "mean",
    "outgoing_sms_per_month": "mean",
    "monthly_mb": "mean",
    "use_id": "count"
}).sort_values('use_id', ascending=False)
```

Out[160]:

	outgoing_mins_per_month	outgoing_sms_per_month	monthly_mb	use_id
Retail Branding				
Samsung	196.975556	93.815354	3725.970707	99
HTC	289.315789	97.678421	7080.200000	19
Sony	143.703846	39.114615	2715.352308	13
Motorola	96.780000	68.844000	4195.424000	5
OnePlus	308.740000	51.772500	8824.890000	4
Huawei	81.526667	9.500000	1561.226667	3
LGE	111.530000	12.760000	1557.330000	2
Lava	60.650000	261.900000	12458.670000	2
Lenovo	215.920000	12.930000	1557.330000	1
Vodafone	42.750000	46.830000	5191.120000	1
ZTE	42.750000	46.830000	5191.120000	1

Merge using pandasql

In [178]:
```
usage_and_device_sql = pds.sqldf("""
    SELECT uu.*, device FROM user_usage uu
    JOIN user_device USING (use_id)
""", {'user_usage': user_usage, 'user_device': user_device})
print('Total:', usage_and_device_sql.shape[0])
usage_and_device_sql.head()
```

```
Total: 159
```

Out[178]:

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

In [201]:
```
usage_and_device_android_sql = pds.sqldf("""
    SELECT DISTINCT ud.*, `Retail Branding` FROM usage_and_device ud
    JOIN android_devices ad ON ud.device = ad.Model
""", {'usage_and_device': usage_and_device_sql, 'android_devices': android_devices})
print('Total:', usage_and_device_android_sql.shape[0])
usage_and_device_android_sql.head()
```

```
Total: 150
```

Out[201]:

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

In [205]:
```python
pds.sqldf("""
    SELECT `Retail Branding`,
           AVG(outgoing_mins_per_month),
           AVG(outgoing_sms_per_month),
           AVG(monthly_mb),
           COUNT(use_id) use_id FROM usage_and_device_android
    GROUP BY `Retail Branding`
    ORDER BY use_id DESC
""", {'usage_and_device_android': usage_and_device_android_sql})
```

Out[205]:

	Retail Branding	AVG(outgoing_mins_per_month)	AVG(outgoing_sms_per_month)	AVG(monthly_mb)	use_id
0	Samsung	196.975556	93.815354	3725.970707	99
1	HTC	289.315789	97.678421	7080.200000	19
2	Sony	143.703846	39.114615	2715.352308	13
3	Motorola	96.780000	68.844000	4195.424000	5
4	OnePlus	308.740000	51.772500	8824.890000	4
5	Huawei	81.526667	9.500000	1561.226667	3
6	LGE	111.530000	12.760000	1557.330000	2
7	Lava	60.650000	261.900000	12458.670000	2
8	Lenovo	215.920000	12.930000	1557.330000	1
9	Vodafone	42.750000	46.830000	5191.120000	1
10	ZTE	42.750000	46.830000	5191.120000	1