

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



Отчет
Лабораторная работа № 2
По курсу «Технологии машинного обучения»
«Изучение библиотек обработки данных»

ИСПОЛНИТЕЛЬ:

Сергеев И.В.
Группа ИУ5-64Б

"__" _____ 2020 г.

ПРЕПОДАВАТЕЛЬ:

Гапанюк Ю.Е.

"__" _____ 2020 г.

Москва 2020

Описание задания

Цель лабораторной работы - изучение библиотеки обработки данных Pandas.

Задание - Выполните первое демонстрационное задание "demo assignment" под названием "Exploratory data analysis with Pandas" со страницы курса <https://mlcourse.ai/assignments>

Текст программы

Программа разрабатывалась в IDE PyCharm. Ниже приведён полный листинг программы:

```
###

import numpy
import pandas

###

pandas.set_option('display.max.columns', 100)
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns

###

# Firstly importing data from dataset 'adult.csv' from task
data = pandas.read_csv('adult.csv')
data.head()

###

data['gender'].value_counts()

###

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

###

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

###

ages1 = data.loc[data['income'] == '>50K', 'age']
ages2 = data.loc[data['income'] == '<=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)))

###

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

```

#%%

for (race, sex), sub_df in data.groupby(['race', 'gender']):
    print("Race: {0}, gender: {1}".format(race, sex))
    print(sub_df['age'].describe())

#%%

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

#%%

data.loc[(data['gender'] == 'Male') &
         (data['marital-status'].str.startswith('Married'))], 'income'].value_counts()

#%%

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

#%%

max_load = data['hours-per-week'].max()
print("Max working time per week: {0}".format(max_load))
num_workaholics = data[data['hours-per-week']==max_load].shape[0]
print("Number of workers satisfying it: {0}".format(num_workaholics))
rich_share = float(data[(data['hours-per-week']==max_load)& (data['income'] ==
'>50K')].shape[0])/num_workaholics
print("Rich workers percent: {0}%".format(int(100 * rich_share)))

#%%

pandas.crosstab(data['native-country'], data['income'],
                values=data['hours-per-week'], aggfunc=numpy.mean).T

#%%

dataframe1 = {
    'UserID': [0,1,2,3,4,5,6,7,8,9],

    'UserName': ['Petya', 'Vasya', 'Kolya', 'Semen', 'Masha', 'Nikita', 'Dmitriy', 'Vasiliy', 'Joh
n', 'Anya'],
    'age': [10,12,15,22,23,60,11,34,21,10],
    'socialID': [0,1,2,3,2,1,0,3,3,1]
}
dataframe2 = {
    'socialID': [0,1,2,3],
    'socialNet': ['Facebook', 'Inst', 'Telegram', 'WhatsApp']
}
Users = pandas.DataFrame(dataframe1, columns = ['UserID', 'UserName', 'age', 'socialID'])
Soc = pandas.DataFrame(dataframe2, columns = ['socialID', 'socialNet'])

#%%

import time
time1 = time.time()
result = pandas.merge(Users,
                      Soc,
                      on='socialID',

```

```

        how='right')
time2 = time.time()
time_rez=time2-time1
print("Result : \n {0}".format(result))
print("Time : {0}".format(time_rez))

#%%%

import pandasql as ps
simple_query = '''SELECT * FROM Users as d1 JOIN Soc as d2 ON
d1.socialID=d2.socialID'''
time1 = time.time()
ps.sqldf(simple_query, locals())
time2 = time.time()
time_rez=time2-time1
print("Время : {0}".format(time_rez))

#%%%

data2 = pandas.read_csv('adult.csv')
data.head()

#%%%

data_fires = pandas.read_csv('forestfires.csv', sep = ',')
data_fires.head()

#%%%

time1 = time.time()
aggregations = {
    'area':{
        'min_area': 'min',
        'max_area': 'max'
    },
    'rain':{
        'min_rain': 'min',
        'max_rain': 'max'
    },
    'temp':{
        'min_temp': 'min',
        'max_temp': 'max',
        'average_temp': 'mean'
    },
    'wind':{
        'min_wind_speed': 'min',
        'max_wind_speed': 'max',
        'average_wind_speed': 'mean'
    },
    'day':{
        'sumarry_days': 'count'
    }
}

result = data_fires.groupby('month').agg(aggregations)
time2 = time.time()
time_rez=time2-time1
print("Результат : \n {0}".format(result))
print("Время : {0}".format(time_rez))

#%%%

```

```

time1 = time.time()
# Запрос с агрегацией в PandaSQL
SQLQuery1='''
SELECT
    month,
    min(area) as min_area,
    max(area) as max_area,
    min(rain) as min_rain,
    max(rain) as max_rain,
    min(temp) as min_temp,
    max(temp) as max_temp,
    avg(temp) as average_temp,
    min(wind) as min_wind_speed,
    max(wind) as max_wind_speed,
    avg(wind) as average_wind_speed,
    count(day) as summary_days

FROM data_fires
GROUP BY month
ORDER BY month
'''

result = ps.sqldf(SQLQuery1, locals())
time2 = time.time()
time_rez=time2-time1
print("Результат : \n {0}".format(result))
print("Время : {0}".format(time_rez))

```

Примеры выполнения программы

Выполнение программы, а также наглядная демонстрация входных и выходных данных (таблиц, графиков и тд) осуществлялась на базе Jupyter Notebook, сервер которого запускался из-под PyCharm. Ниже приведены скриншоты, отражающие работу программы:

The screenshot shows a Jupyter Notebook interface with the following content:

In [1]:

```
import numpy
import pandas
```

In [2]:

```
pandas.set_option('display.max.columns', 100)
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:

```
# Firstly importing data from dataset 'adult.csv' from task
data = pandas.read_csv('adult.csv')
data.head()
```

Out[3]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States	<=50K
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States	<=50K
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States	>50K
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States	>50K
4	18	?	103497	Some-college	10	Never-married	?	Own-child	White	Female	0	0	30	United-States	<=50K

In [4]:

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



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

```
Out[4]: Male      32650
        Female    16192
        Name: gender, dtype: int64
```

```
In [5]: data.loc[data['gender'] == 'Female', 'age'].mean()
```

```
Out[5]: 36.92798913043478
```

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

```
Out[6]: 0.00421768150362393
```

```
In [7]: ages1 = data.loc[data['income'] == '>50K', 'age']
        ages2 = data.loc[data['income'] == '<=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)))
```

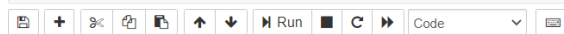
The average age of the rich: 44.0 +/- 10.6 years, poor - 37.0 +/- 14.1 years.

```
In [8]: data.loc[data['income'] == '>50K', 'education'].unique()
```

```
Out[8]: array(['Assoc-acdm', 'Some-college', 'Prof-school', 'HS-grad', 'Masters',
               'Doctorate', 'Bachelors', 'Assoc-voc', '9th', '10th', '7th-8th',
               '11th', '5th-6th', '1st-4th', '12th', 'Preschool'], dtype=object)
```

```
In [9]: for (race, sex), sub_df in data.groupby(['race', 'gender']):
        print("Race: {0}, gender: {1}".format(race, sex))
        print(sub_df['age'].describe())
```

```
Race: Amer-Indian-Eskimo, gender: Female
count    185.000000
mean      36.237838
```



```
In [9]: for (race, sex), sub_df in data.groupby(['race', 'gender']):
        print("Race: {0}, gender: {1}".format(race, sex))
        print(sub_df['age'].describe())
```

```
Race: Amer-Indian-Eskimo, gender: Female
count    185.000000
mean      36.237838
std       12.840056
min       17.000000
25%       26.000000
50%       35.000000
75%       46.000000
max       80.000000
Name: age, dtype: float64
Race: Amer-Indian-Eskimo, gender: Male
count    285.000000
mean      36.989474
std       11.703943
min       17.000000
25%       29.000000
50%       35.000000
75%       44.000000
max       82.000000
Name: age, dtype: float64
Race: Asian-Pac-Islander, gender: Female
count    517.000000
mean      35.657640
std       12.637799
min       17.000000
25%       25.000000
50%       34.000000
75%       44.000000
max       81.000000
Name: age, dtype: float64
Race: Asian-Pac-Islander, gender: Male
count    1000.000000
```

jupyter lab2 Last Checkpoint: 3 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```

25%      28.000000
50%      36.000000
75%      46.000000
max       90.000000
Name: age, dtype: float64
Race: Black, gender: Male
count    2377.000000
mean      37.922592
std       13.036981
min       17.000000
25%      28.000000
50%      36.000000
75%      46.000000
max       90.000000
Name: age, dtype: float64
Race: Other, gender: Female
count     155.000000
mean      31.212903
std       11.233061
min       17.000000
25%      23.000000
50%      29.000000
75%      37.000000
max       74.000000
Name: age, dtype: float64
Race: Other, gender: Male
count     251.000000
mean      35.167331
std       11.808297
min       17.000000
25%      26.000000
50%      32.000000
75%      42.500000
max       77.000000
Name: age, dtype: float64
Race: White, gender: Female

```

jupyter lab2 Last Checkpoint: 3 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```

mean      36.882935
std       14.454178
min       17.000000
25%      25.000000
50%      35.000000
75%      47.000000
max       90.000000
Name: age, dtype: float64
Race: white, gender: Male
count    28735.000000
mean      39.704507
std       13.475250
min       17.000000
25%      29.000000
50%      38.000000
75%      49.000000
max       90.000000
Name: age, dtype: float64

```

In [10]: `data.loc[(data['gender'] == 'Male') & (data['marital-status'].isin(['Never-married', 'Separated', 'Divorced', 'Widowed']))], 'income'].value_counts()`

```

Out[10]: <=50K    11414
         >50K     1001
         Name: income, dtype: int64



```

In [11]: `data.loc[(data['gender'] == 'Male') & (data['marital-status'].str.startswith('Married'))], 'income'].value_counts()`

```

Out[11]: <=50K    11318
         >50K     8917
         Name: income, dtype: int64

```


jupyter lab2
Last Checkpoint: 3 minutes ago (autosaved)
 Logout

File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3

In [12]:

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

Out[12]:

Married-civ-spouse 22379
Never-married 16117
Divorced 6633
Separated 1530
Widowed 1518
Married-spouse-absent 628
Married-AF-spouse 37
Name: marital-status, dtype: int64

In [13]:

max_load = data['hours-per-week'].max()
print("Max working time per week: {}".format(max_load))
num_workaholics = data[data['hours-per-week']==max_load].shape[0]
print("Number of workers satisfying it: {}".format(num_workaholics))
rich_share = float(data[(data['hours-per-week']==max_load)&(data['income'] == '>50K')].shape[0])/num_workaholics
print("Rich workers percent: {}%".format(int(100 * rich_share)))



Max working time per week: 99
Number of workers satisfying it: 137
Rich workers percent: 29%

In [14]:

pandas.crosstab(data['native-country'], data['income'],
values=data['hours-per-week'], aggfunc=numpy.mean).T

Out[14]:

	native-country	?	Cambodia	Canada	China	Columbia	Cuba	Dominican-Republic	Ecuador	El-Salvador	England	France	Germany	Greece	Gua
income															
<=50K	39.552590	41.157895	37.378151	36.686047	39.123457	39.201923	41.561224	37.948718	35.819444	39.375000	40.090909	38.898649	41.870968	38.	
>50K	45.318182	43.888889	46.126984	42.027778	56.250000	42.852941	42.800000	47.833333	43.454545	46.297872	46.500000	45.706897	55.555556	36.	


jupyter lab2
Last Checkpoint: 4 minutes ago (autosaved)
 Logout

File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3

In [15]:

dataframe1 = {
'UserID': [0,1,2,3,4,5,6,7,8,9],
'UserName': ['Petya', 'Vasya', 'Kolya', 'Semen', 'Masha', 'Nikita', 'Dmitriy', 'Vasiliy', 'John', 'Anya'],
'age': [10,12,15,22,23,60,11,34,21,10],
'socialID': [0,1,2,3,2,1,0,3,3,1]
}
dataframe2 = {
'socialID': [0,1,2,3],
'socialNet': ['Facebook', 'Inst', 'Telegram', 'WhatsApp']
}
Users = pandas.DataFrame(dataframe1, columns = ['UserID', 'UserName', 'age', 'socialID'])
Soc = pandas.DataFrame(dataframe2, columns = ['socialID', 'socialNet'])

In [16]:

import time
time1 = time.time()
result = pandas.merge(Users,
Soc,
on='socialID',
how='right')
time2 = time.time()
time_rez=time2-time1
print("Result : \n {}".format(result))
print("Time : {}".format(time_rez))

Result :

	UserID	UserName	age	socialID	socialNet
0	0	Petya	10	0	Facebook
1	6	Dmitriy	11	0	Facebook
2	1	Vasya	12	1	Inst
3	5	Nikita	60	1	Inst
4	9	Anya	10	1	Inst
5	2	Kolya	15	2	Telegram
6	4	Masha	23	2	Telegram
7	3	Semen	22	3	WhatsApp
8	7	Vasiliy	34	3	WhatsApp
9	8	John	21	3	WhatsApp


```
In [17]: import pandasql as ps
simple_query = '''SELECT * FROM Users as d1 JOIN Soc as d2 ON d1.socialID=d2.socialID'''
time1 = time.time()
ps.sqldf(simple_query, locals())
time2 = time.time()
time_rez=time2-time1
print("Время : {}".format(time_rez))
```

Время : 0.021942615509033203

```
In [18]: data2 = pandas.read_csv('adult.csv')
data.head()
```

Out[18]:

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relationship	race	gender	capital-gain	capital-loss	hours-per-week	native-country	income
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States	<=50K
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States	<=50K
2	28	Local-gov	336951	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States	>50K
3	44	Private	160323	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States	>50K
4	18	?	103497	Some-college	10	Never-married	?	Own-child	White	Female	0	0	30	United-States	<=50K

```
In [19]: data_fires = pandas.read_csv('forestfires.csv', sep = ',')
data_fires.head()
```

Out[19]:

	X	Y	month	day	FFMC	DMC	DC	ISI	temp	RH	wind	rain	area
0	7	5	mar	fri	86.2	26.2	94.3	5.1	8.2	51	6.7	0.0	0.0