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 - и**зучение 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'**,**'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'**])  
  
#%%  
  
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  
SQLQuerly1=**'''  
 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(SQLQuerly1, 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:















