НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ УКРАИНЫ
“КИЕВСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ”
ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
КАФЕДРА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

ЛАБОРАТОРНАЯ РАБОТА №3

Дисциплина: «Специальные разделы программирования»

Тема: «Структуры для работы с большими объёмами данных в Python»

Выполнила

студентка 2 курса группы ФИ-41

Лавягина Ольга Алексеевна

Проверил

Колотий Андрей Всеволодович

Киев — 2016 года

1 ДАННЫЕ

В лабораторной работе используется набор данных Individual household comsumption Data Set, который можно загрузить из UCI-репозитория [1].

Этот архив содержит 2075259 измерений собранных с декабря 2006 года до ноября 2010 года (47 месяцев). Набор данных содержит недостающие значения в измерениях (около 1,25% строк). Очищенный набор данных содержит 2049281 строку (измерение).

Обозначения:

- date: дата измерения в формате dd/mm/yyyy;
- global_active_power: активная мощность, которую употребляет домохозяйство в минуту (усреднено) [кВт];
- global_reactive_power: реактивная мощность, которую употребляет домохозяйство в минуту (усреднено) [кВт];
- voltage: напряжение, усреднённое за минуту наблюдения [В];
- global_intensity: усреднённая сила тока для домохозяйства [А];
- sub_metering_1: набор употредителей энергии №1 [Вт-час активной энергии], отвечает кухне, на которой машина для мытья посуды и микроволновка (электрической плиты нет, используется газовая);
- sub_metering_2: набор употредителей энергии №2 [Вт-час активной энергии], отвечает прачечной, в которой работает стиральная машина, сушилка, холодильник и включен свет;
- sub_metering_3: набор употредителей энергии №3 [Вт-час активной энергии], отвечает бойлеру и кондиционеру.

2 ЗАДАНИЕ

Выполнить все задания, используя как numpy array, так и dataframe, проанализовав часовые траты на выполнение процедур (профилировка времени выполнения), сделав выводы по поводу ситуаций, в которых имеет смысл отдать преимущество той или иной структуре данных. Выводы оформить отчётом с указанным временем выполнения и оценкой по 5-бальной шкале удобности выполнения операций отбора.

Также необходимо оставить только те наблюдения, в которых нет пустых наблюдений (пустые значения — пустые поля между разделителем — ? — 28.04.2007, как пример).

Задания:

1) выбрать все домохозяйства, в которых общая активная употребляемая мощность превышает 5 кВт;
2) выбрать все домохозяйства, в которых вольтаж превышает 235 В;
3) выбрать все домохозяйства, в которых сила тока лежит в пределах 19-20 А, для них обнаружить те, в которых стиральная машина и холодильник употребляют больше, чем бойлер и кондицинер;
4) выбрать случайным образом 500000 домохозяйств (без повторов элементов выборки), для них вычислить средние величины всех 3-х групп употребления электрической энергии;
5) выбрать те домохозяйства, которые после 18-00 употребляют больше 6 кВт за минуту в среднем, среди отобранных определить те, у которых основное употребление электроэнергии в указанный промежуток времени приходится на стиральную машину, сушилку, холодильник и освещение (группа 2 является самой большой), а потом выбрать каждый второй результат из первой половины и каждый четвёртый результат из второй половины.

3 ЛИСТИНГ КОДА

Листинг кода с использованием dataframe

```
import csv
import pandas as pd
from cProfile import Profile
from pstats import Stats

profile = Profile()

def clean_data():

    raw_data_path = './household_power_consumption.csv'
    clean_data_path = './household_power_consumption_clean.csv'

    with open(raw_data_path, 'rb') as old_file:
        reader = csv.reader(old_file, delimiter=';')

        with open(clean_data_path, 'wb') as new_file:
            f = csv.writer(new_file, delimiter=';')

            for line in reader:
                if '?' in line:
                    continue
                f.writerow(line)

profile.enable()
clean_data()
profile.disable()
Stats(profile).sort_stats('time').print_stats()

def read_frame():
    path = './household_power_consumption_clean.csv'
    df = pd.read_csv(path, index_col=False, header=8, delimiter=';',
                     names=['Date', 'Time', 'Global_active_power', 'Global_reactive_power', 'Voltage',
                     'Global_intensity', 'Sub_metering_1', 'Sub_metering_2', 'Sub_metering_3'])
    return df

def active_power():
    df = read_frame()
    print 'Households with Global_active_power more than 5 kW'
    df1 = df[df['Global_active_power'] > 5]
    print df1[:5]

active_power()

def voltage():
```

```
    df = read_frame()
    print 'Households with Voltage more than 235 V'
    df1 = df[df['Voltage'] > 235]
    print df1[:5]

voltage()


def intensity():
    df = read_frame()
    print 'Households with Global_intensity in range from 19 to 20 A and where washer and fridge comsump' +
    'more than boiler and the conditioner'
    df1 = df[(df['Global_intensity'] > 19) & (df['Global_intensity'] < 20) &
    (df['Sub_metering_2'] > df['Sub_metering_3'])]
    print df1[:5]


intensity()


def average_consumption():
    df = read_frame()
    print '500000 random unique households with average consumption found'
    df = df.drop_duplicates(keep=False)
    df = df.sample(n=500000)
    df['Average'] = (df['Sub_metering_1'] + df['Sub_metering_2'] + df['Sub_metering_3'])/3
    print df[:5]

average_consumption()


def after_18():
    df = read_frame()
    print 'Households whete after 18:00 Global_active_power per minute is more than 5 kW, Sub_metering_2' +
    'is more than others, choosen every second result from first part and every fourth from second part'
    df = df[(df['Time'] > '18:00:00') & (df['Global_active_power'] > 5) &
    (df['Sub_metering_2'] > df['Sub_metering_1']) & (df['Sub_metering_2'] > df['Sub_metering_3'])]
    df1 = df[:len(df.index)/2:2]
    df2 = df[len(df.index)/2::4]
    frames = [df1, df2]
    result = pd.concat(frames)
    print result[:5]

after_18()
```

Листинг кода с использованием numpy array

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

```
import random
from cProfile import Profile
from pstats import Stats

profile = Profile()

def read_array():
    path = './household_power_consumption_clean.csv'
    df = pd.read_csv(path, index_col=False, header=8, delimiter=';',
                     names=['Date', 'Time', 'Global_active_power', 'Global_reactive_power', 'Voltage',
                     'Global_intensity', 'Sub_metering_1', 'Sub_metering_2', 'Sub_metering_3'])
    my_data = df.values
    return my_data

def active_power():
    array = read_array()
    print 'Households with Global_active_power more than 5 kW'
    array = array[array[:,2] > 5]
    np.set_printoptions(edgeitems=9)
    print array

profile.enable()
active_power()
profile.disable()
Stats(profile).sort_stats('time').print_stats()

def voltage():
    array = read_array()
    print 'Households with Voltage more than 235 V'
    array = array[array[:,4] > 235]
    np.set_printoptions(edgeitems=9)
    print array

voltage()

def intensity():
    array = read_array()
    print 'Households with Global_intensity in range from 19 to 20 A and where washer and fridge comsump' +
    'more than boiler and the conditioner'
    array = array[(array[:,5] > 19) & (array[:,5] < 20) & (array[:,7] > array[:,8])]
    np.set_printoptions(edgeitems=9)
    print array

intensity()

def average_consumption():
    array = read_array()
    print '500000 random households with average consumption found'
    array = array[np.random.randint(array.shape[0],size=50000), :]
```

```
    average = np.array((array[:,6] + array[:,7]  + array[:,8])/3)
    average = average.reshape((50000,1))
    array = np.concatenate((array, average), axis=1)
    np.set_printoptions(edgeitems=9)
    print array

average_consumption()

def after_18():
    array = read_array()
    print 'Households whete after 18:00 Global_active_power per minute is more than 5 kW, Sub_metering_2' +
    'is more than others, choosen every second result from first part and every fourth from second part'
    array = array[(array[:,1] > '18:00:00') & (array[:,2] > 5) & (array[:,7] > array[:,6]) &
    (array[:,7] > array[:,8])]

    array1 = array[:len(array)/2:2]
    array2 = array[len(array)/2::4]
    array = np.concatenate((array1, array2), axis=0)
    np.set_printoptions(edgeitems=9)
    print array

after_18()
```

4 АНАЛИЗ

На 4.1 изображено время выполнения функций с использованием dataframe. Названия строк — функции, используемые для выполнения задания. «Чтение файла» – время, за которое из csv-файла считываются данные и заносятся в dataframe или numpy array. «Функция» — время, за которое выполняется функция. Так как в каждой функции используется функция считывания данных из файла, то есть смысл показать, сколько времени выполняется функция без создания dataframe или numpy array из csv-файла — «Чтение файла – Функция». Всё время приведено в секундах. В последней строке находится только одно значение — это среднее время считывания данных из файла и создания из него структуры данных (среднее время «Чтение файла»)

Время	Чтение файла	Функция	Чтение файла – Функция
active_power	2.215	2.468	0.253
voltage	2.066	2.438	0.372
intensity	2.061	2.227	0.166
average_consumption	2.081	4.381	2.3
after_18	2.107	2.314	0.207
	2.106		

Таблиця 4.1 — Время для dataframe

На 4.2 изображено время выполнения функций с использованием numpy array (обозначения аналогичны 4.1).

Время	Чтение файла	Функция	Чтение файла - Функция
active_power	1.984	3.114	1.13
voltage	2.009	3.019	1.01
intensity	1.977	2.626	0.649
average_consumption	1.973	2.537	0.564
after_18	1.97	2.608	0.638
	1.9826		

Таблиця 4.2 — Время для numpy array

ВЫВОДЫ

В csv-файле, используемом в лабораторной работе, содержатся данные разных типов (числа с плавающей точком, дата, время, строка), поэтому при считывании из него в numpy array возникли трудности. Поэтому работа с dataframe оказалась удобнее. Время, которое тратится на создание numpy array из csv-файла меньше чем время, необходимое на создание dataframe из того же файла. Время выполнения функций получилось больше при использовании numpy array. Стандартные функции для создания выборок у двух рассматриваемых структур данных похожи. Обращение к элементам выборки в dataframe используется с использованием названия колонок, а в numpy array с использованием индексов. Pandas — это инструмент, который обеспечивает более удобный и рациональный способ работы с табличными данными в Python. Я бы оценила работу с dataframe в 5 баллов, а с numpy array в 4 балла из 5.

СПИСОК ЛИТЕРАТУРЫ

1. Lichman, M. UCI Machine Learning Repository. — 2013. http://archive.ics.uci.edu/ml.