|  |  |
| --- | --- |
|  | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ**

Отчет по лабораторной работе № 2

по курсу “Введение в машинное обучение”

Исполнитель:

Студент группы ИУ5-43

Зинченко Я.И.

Москва, 2018

**Задание лабораторной работы**

Необходимо реализовать скрипт, выполняющий следующие действия:

1. Скачивание 1000 последних объявлений с hh.ru

Для выполнения этого пункта вам понадобится библиотека requests. С помощью этой библиотеки можно делать HTTP-запросы в API hh.ru. Что такое api. С API, предоставляемым hh.ru, можно ознакомиться здесь. Общая информация здесь

Вам понадобится этот метод. Чтобы тестировать запросы в api и смотреть, что они возвращают, можно использовать Postman. В качестве поискового запроса можно вводить ключевые слова, связанные с тематикой анализа данных: machine learning, data science, машинное обучение, big data, data analytics и тд. В ответе API будут интересующие нас поля: salary, area, name, employer

2. Получить медианное значение зарплат

Необходимо сделать обработку полученных на первом шаге данных и получить следующую структуру: Словарь, где ключом является название вакансии (как оно задано на hh.ru), а значением - медианное значение зарплаты по этой вакансии. То есть необходимо сгруппировать данные по имени вакансии. Также можно использовать другие варианты, например, сгруппировать по городу или любому другому интересному параметру из выдачи. В поле salary hh.ru отдает значения диапазона. Значением зарплаты считать среднее значение из диапазона, например, если зп от 100 до 150, то фиксировать значение 125.

3. Получить распределение зарплат по диапазонам

Необходимо выделить диапазоны зарплат, например:

до 80к, 80-120к, 120-150к, 150-200к, 200-300к, 300к+ Для каждого диапазона подсчитать количество предлагаемых вакансий.

\*. Построить графики по пунктам 2 и 3

**Текст программы**

**from** **functools** **import** reduce

**import** **requests**

**import** **matplotlib.pyplot** **as** **plt**

**from** **pylab** **import** \*

**from** **matplotlib** **import** rcParams

%matplotlib inline

#считывание 1000 вакансий с hh.ru

str\_ = []

req = []

**for** i **in** range(**10**):

i = 'https://api.hh.ru/vacancies?per\_page=100&page=' + str(i) + '&text=machine+learning+OR+big data'

str\_.append(i)

**for** j **in** str\_:

req.append(requests.get(j))

#обработка данных в список

js = []

**for** i **in** req:

js.append(i.json())

vac = []

**for** i **in** js:

**for** vacancy **in** i['items']:

vac.append(vacancy)

#получение значения зарплат

sal\_dict = {}

**print**('**\n**')

**for** sal **in** vac:

**if**(sal['salary'] != None):

**if** ((sal['salary']['to'] != None) **and** (sal['salary']['from'] != None)):

salary = (sal['salary']['to'] + sal['salary']['from'])/**2**

sal\_dict[sal['name']] = salary

**elif** ((sal['salary']['to'] == None) **and** (sal['salary']['from'] != None)):

salary = sal['salary']['from']

sal\_dict[sal['name']] = salary

**elif** ((sal['salary']['to'] != None) **and** (sal['salary']['from'] == None)):

salary = sal['salary']['to']/**2**

sal\_dict[sal['name']] = salary

#print(sal\_dict)

#получение медианного значения зарплат

sal\_data = []

sal\_prog = []

sal\_deve = []

sal\_othe = []

**for** val **in** sal\_dict.keys():

**if**('ata' **in** val):

sal\_data.append(sal\_dict[val])

**elif**('рограммист' **in** val):

sal\_prog.append(sal\_dict[val])

**elif**('азработчик' **in** val):

sal\_deve.append(sal\_dict[val])

**elif**('дминистратор' **in** val **or** 'енеджер' **in** val):

sal\_othe.append(sal\_dict[val])

sal\_data = sorted(sal\_data)

sal\_prog = sorted(sal\_prog)

sal\_deve = sorted(sal\_deve)

sal\_othe = sorted(sal\_othe)

med\_sdat = sal\_data[int(len(sal\_data)/**2**)]

med\_spro = sal\_prog[int(len(sal\_prog)/**2**)]

med\_sdev = sal\_deve[int(len(sal\_deve)/**2**)]

med\_soth = sal\_othe[int(len(sal\_othe)/**2**)]

**print**(med\_sdat, ' ', med\_spro, ' ', med\_sdev, ' ', med\_soth)

#получение распределения всех зарплат по диапазонам

val = sal\_dict.values()

s1 = reduce((**lambda** x, y: x + y), map((**lambda** x: **1** **if** x < **30000** **else** **0**), sal\_dict.values()))

s2 = reduce((**lambda** x, y: x + y), map((**lambda** x: **1** **if** **30000** < x < **50000** **else** **0**), sal\_dict.values()))

s3 = reduce((**lambda** x, y: x + y), map((**lambda** x: **1** **if** **50000** < x < **80000** **else** **0**), sal\_dict.values()))

s4 = reduce((**lambda** x, y: x + y), map((**lambda** x: **1** **if** **80000** < x < **100000** **else** **0**), sal\_dict.values()))

s5 = reduce((**lambda** x, y: x + y), map((**lambda** x: **1** **if** **100000** < x < **150000** **else** **0**), sal\_dict.values()))

s6 = reduce((**lambda** x, y: x + y), map((**lambda** x: **1** **if** **1500000** < x **else** **0**), sal\_dict.values()))

#график №1

bins = range(**15000**, **300000**, **10000**)

salary = []

**for** i **in** sal\_dict.values():

salary.append(int(i))

plt.title('all salary schedule')

plt.ylabel("value")

plt.xlabel("salary")

hist, bins = np.histogram(salary, bins = bins)

**print**(hist, bins)

plt.hist(salary, bins)

plt.show()

#график медиан

bins = range(**50000**, **250000**, **50000**)

med = [med\_sdat, med\_spro, med\_sdev, med\_soth]

plt.title('med salary schedule')

plt.ylabel("value")

plt.xlabel("salary")

hist, bins = np.histogram(med, bins = bins)

**print**(hist, bins)

plt.hist(med, bins)

plt.show()

#график диапазона

bins = range(**0**, **100**, **10**)

ran = [s1, s2, s3, s4, s5, s6]

plt.title('salary schedule')

plt.ylabel("value")

plt.xlabel("people")

hist, bins = np.histogram(ran, bins = bins)

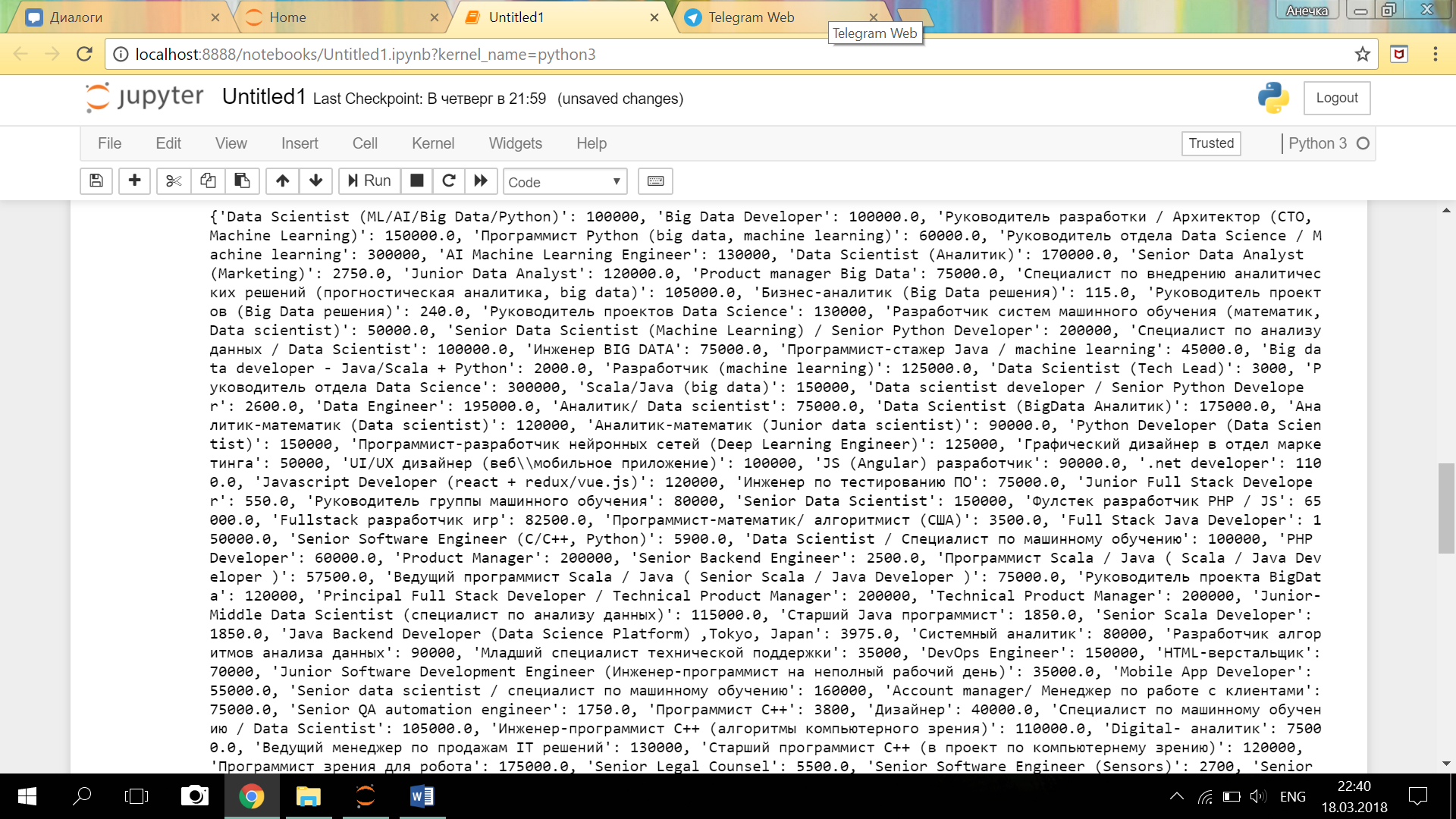
**print**(hist, bins)

plt.hist(ran, bins)

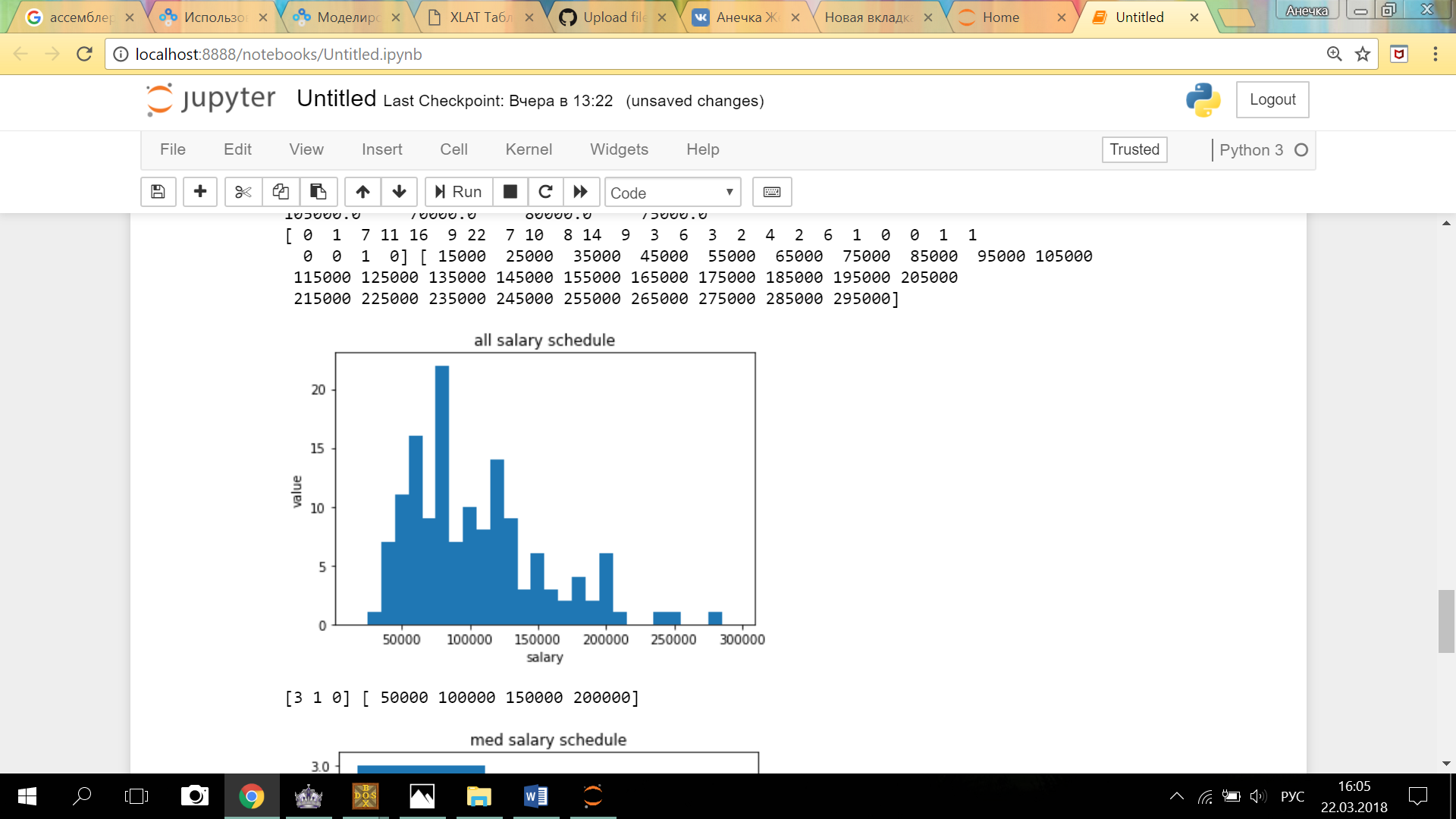
plt.show()

**Результат выполнения**

1.



2.



3.

