Министерство науки и высшего образования РФ
Федеральное Государственное автономное образовательное учреждение высшего образования
«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ МАТЕМАТИКИ И МЕХАНИКИ ИМ. Н.И.ЛОБАЧЕВСКОГО
КАФЕДРА МАТЕМАТИЧЕСКОГО АНАЛИЗА
Направление: 02.03.01 – Математика и компьютерные науки
Профиль: Математическое и комьютерное моделирование

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Сбор данных с html-страниц

Работа завершена:

Студент гр. 05-503
«__» _____ 2019 г. _____ И. Р. Салимов

Работа допущена к защите:

Научный руководитель,
кандидат физ. -мат. наук
«__» _____ 2019 г. _____ А. А. Новиков

Заведующий кафедрой
математического анализа,
доктор физ. -мат. наук, профессор
«__» _____ 2019 г. _____ С. Р. Насыров

Казань — 2019

Содержание

Глоссарий

Библиотека языка — сборник подпрограмм и объектов языка программирования, используемых для разработки программного обеспечения.

Браузер — программное обеспечение для просмотра веб-страниц.

Интернет — глобальная система объединённых компьютерных сетей (и компьютеров) для хранения, обработки и передачи информации.

Прокси-сервер — сервер, являющийся посредником между узлами сети Интернет и осуществляющий обмен данными между ними.

Сайт — информационный ресурс сети Интернет, доступ к которому осуществляется с помощью браузера.

Сервер — специализированный компьютер или специализированное оборудование, предназначенное для выполнения на нём специализированного сервисного программного обеспечения.

Скрипт — последовательность действий в программировании, созданная для автоматического выполнения задачи.

CSV — широкоиспользуемый формат текстовых данных, предназначенный для представления таблиц.

Get-запрос — передача данных в сети Интернет с помощью ссылки (либо в ссылке).

Html-страница — написанный на языке разметки html документ или информационный ресурс сети Интернет, доступ к которому осуществляется с помощью браузера.

Url-адрес — система уникальных адресов электронных ресурсов (веб-страниц, документов) сети Интернет.

Ip-адрес — уникальный сетевой адрес узла в сети Интернет.

Введение

Текущий этап развития человечества характеризуется огромным ростом количества накопленной (и непрерывно растущей в количестве) информации. Такой большой объем данных порождает увеличение числа научных экспериментов. Так, в 2016 году, в виде открытых (доступных без каких-либо ограничений) данных были выложены сотни терабайт экспериментальных данных, полученных на большой адронном коллайдере. Работа многих технических систем также сопровождается сбором большого числа данных.

Однако, ввиду увеличения количества информации и её усложнения, задача сбора данных становится всё более нетривиальной и требует специальной обработки данных для получения практически полезной информации. Так, все данные, представленные в глобальной сети Интернет, в общем смысле неструктурированы, так как каждый ресурс в отдельности имеет индивидуальную и специфичную архитектуру. В основном, такие данные – это html-страницы, в которых хранится текстовая информация, содержащая полезную, и в гораздо большем объеме не имеющую практическую пользу (в плане возможного анализа) информацию. И объем данной информации растет каждый день. Всё это требует развитие технологий, которые позволяют собирать, обрабатывать и извлекать информацию, полезную в практическом смысле.

Данная выпускная квалификационная работа посвящена исследованию и применению методов сбора таких (полезных в практическом смысле) данных с открытых источников сети Интернет и их обработке для возможности дальнейшего анализа.

В качестве открытого источника информации взят html-сайт, являющийся площадкой по размещению объявлений недвижимости. Такой выбор сделан исходя из того, что в последнее десятилетие построение статистических моделей рынка недвижимости стало востребованным направлением как в научном, так и прикладном смысле. Оно оказывается всё более нетривиальной задачей анализа в виду роста рынка недвижимости.

Целью данной выпускной квалификационной работы является создание программы сбора практически полезных данных с html-страниц сайта объявлений о недвижимости. Для достижения поставленной цели необходимо решить ряд задач:

1) собрать данные о недвижимости с html-страниц (решению данной задачи посвящена глава 1);

2) реализовать алгоритм обработки данных для получения информации, имеющей практическую пользу (решению данной задачи посвящена глава 2);

3) отобразить практическую полезность собранных данных в наглядной форме (решению данной задачи посвящена глава 3).

Инструментом решения данной задачи выбран один из языков объектно-ориентированного программирования — высокоуровневый язык общего назначения Python.

Дипломная работа состоит из введения, 3 глав, заключения, списка использованных источников и 2 приложения. Работа содержит 19 страниц основного текста, 18 страниц приложения, 7 листингов и 2 рисунка. Список использованной литературы включает 8 наименований.

1 Сбор данных

Задача данной главы — описать алгоритм сбора данных с html-страниц. Алгоритм реализован в виде программы, которая собирает данные с площадки, размещающей объявления о продаже недвижимости в городе Казань.

Для решения поставленной задачи необходимо:

1) написать функцию скачивания html-страниц;
2) применить данную функцию на всех страницах с объявлениями;

Решению каждой подзадачи уделена отдельная подглава.

1.1 Скачивание html-страниц

Сначала нужно скачать html-страницу (далее — *страница*). В языке Python это делается с помощью функций широкоиспользуемой библиотеки requests созданием сессии через функцию session из той библиотеки. При её использовнии, мы посылаем сайту get-запрос с url-адресом необходимой страницы и заголовками браузера для имитации реального пользователя сети. Описание отправляемых на сайт подзаголовков есть в приложении. В ответ с сервера сайта приходит необходимая страница в виде массива кода на языке html, из которого мы в дальнейшем и будем извлекать полезные в практическом смысле данные.

Листинг 1.1 — Функция скачивания html-страницы

```
1  def get_html(url, curr_proxy, prox_headers):
2      global headers
3      code = ''
4      s = requests.session()
5      if curr_proxy == '0':
6          r = s.get(url, headers=headers)
7          code = r.text
8      else:
9          r = s.get(url, headers=prox_headers, proxies=curr_proxy)
10         code = r.text
11     return code
```

Все описываемые в этой подглаве действия происходят в написанной нами функции get_html. В аргументах данной функции кроме заголовков есть и переменная curr_proxy. Эта переменная отвечает за то, использовали ли мы ранее прокси-серер при подключении к серверу сайта. Если переменная curr_proxy не равна 0, подключение происходит через другие заголовки и через прокси-сервера, полученные через написанную нами функцию connect_to_good_proxy (код этой функции есть в приложение). Их получение происходит в случае, если описанные в следующей главе алгоритмы не могут собрать необходимые данные при парсинге странице. Возникновение таких ошибок говорит о блокировке нашего ip-адреса сервером сайта, для обхода которого и применяется механизм подключения к прокси-серверам.

1.2 Подсчет числа страниц

Так как страниц сайта с казанскими объявлениями множество, необходимо определить, сколько их всего и скачать каждую из них. На сайте множество типов недвижимости, но для наглядности нами выбраны только квартиры (далее, говоря о недвижимости, мы говорим о квартирах). Для определения количества страниц с квартиры нами была написана функция get_numb_of_pages, считающая число страниц посредством перебора параметра p= в url-адресе сайта. При достижении последней страницы сайта, последущее увеличение параметра возвращает первую страницу с объявлениями — это и будет сигналом к прекращению перебора. Так как такой перебор занимает достаточно долго время, нами был разработан такой алгоритм: каждая страница содержит ссылки на 3 следующие, поэтому ивзлекая из кода страницы ссылку на 3 страницу и переходя на неё, а не на последующую, можно ускорить подсчет страниц в 3 раза.

Функция get_numb_of_pages вызывается внутри функции get_from_area — функции, отвечающей за сбор данных в конкретном районе города (сбор данных разделен на сбор данных по районам города и вызов этой функции для нужного района осуществляется отпракой в неё ссылки с объявлениями соответствующего района). Цикл внутри

get_from_area перебирает страницы сайта с объявлениями по данному району, увеличивая значение параметра p в ссылке на страницу до максимального числа страниц, полученного из get_numb_of_pages. При получении ссылки на текущую страницу для получения html-кода страницы вызывается функция get_html, описанная ранее.

Последующая обработка данных производится в функции get_data_from_html, получающей полезные данные из html-кода страница. Описание этой функции дано в следующей главе.

Листинг 1.2 — Функция подсчета числа страниц с объявлениями

```
1  def get_numb_of_pages(url):
2      global curr_proxy
3      url_split = url.split(
4          'p=1')
5      pages = ''
6      n = int(61)
7      cure_page = 0
8      print("Counting...")
9      prox = '0'
10     headers = '0'
11     while True:
12         ***
13     print("All " + str(n) + " pages for this region")
14     return n
```

Здесь и далее «***» в листингах обозначает, что часть деталей кода скрыта. Весь код представлен в приложении.

2 Обработка данных

Задача данной главы — описать алгоритм обработки данных, полученных с html-страниц и реализовать его в виде функции, которая обрабатывает и сохраняет полезные данные, собранные со страниц объявлений для каждого района города Казань.

Для решения поставленной задачи необходимо:

1) написать функцию выделения полезных данных из всего массива html-кода страницы;

2) преобразовать полученные данные в формат, удобный для последующего анализа;

3) сохранить данные для возможности последующего анализа;

Решению каждой подзадачи уделена отдельная подглава.

2.1 Выделение полезных данных

Для извлечения практически полезной информации из html-страницы целесообразно применить процедуру *парсинга*. Парсинг — это анализ кода html-страницы и извлечение необходимых частей информации. Любая html-страница представляет из себя набор html-тегов — специальных кодовых слов, которые заключены в скобки из знаков большеменьше и содержат в себе информацию, которая отображается на вебстранице при просмотре через браузер. Парсинг позволяет извлечь эту информацию.

Производить процедуру парсинга в языке Python позволяет библиотека bs4. Она умеет получать любые части html-кода и текста из страницы, делая выборку на основе указанных селекторов — классов и id искомых тегов. С помощью библиотеки bs4 мы создается объект класса BeautifulSoup, который содержит в себе методы для парсинга.

Листинг 2.1 — Функция сбора данных

```
1 def get_data_from_html(html, area):
2     global number_of_ads
3
4     soup = BeautifulSoup(html, 'lxml')
```

```
5       ads = soup.find('div',
            class_='_93444fe79c-wrapper--1Z8Nz').find_all('div',
            class_='_93444fe79c-card--2Jgih')
6
7       temp_list = []
8       for i in range(0, len(ads)):
9               ***
```

Обработка данных была реализована нами в функции get_data_from_html. После создания объекта класса BeautifulSoup (мы назвали его soup) необходимо посмотреть на исходный код страницы с объявлениями, чтобы найти, какие блоки содержат в себе текст объявлений для возможности дальнейшей обработки их содержимого:

Рисунок 2.1 — Поиск тегов внутри исходного кода страницы

Внимательно рассмотрев код страницы, нами был найден фрагмент текста объявления. Вся необходимая информация (каждое объявление) находится в блоке <div class="_93444fe79c–card–_yguQ». Применим для её извлечения метод find, который есть у объекта soup. Метод

возвращает все блоки с информацией, которую мы ищем, в виде списка из строк (переменная ads). То есть возвращается список из всех объявлений, который кроме полезной иформации содержит также и технические данные html, необходимые для правильного отображения в браузере.

Далее для каждого объявления из списка необходимо произвести процедуру извлечения полезных данных. Объявления содержат в себе дату публикации объявления, цену за объект недвижимости, тип строения, количество комнат, цену за квадратный метр, комментарий владельца недвижимости и многое-многое. Многие из этих данных не представляют практической пользы. Для возможности демонстрации анализа данных (об этом в следующей главе) извлечем из объявления ссылку на него, площадь продаваемой недвижимости, цену, и дату публикации объявления.

С помощью цикла for пройдемся по всему списку объявлений (ads[i]). Объекты этого списка — фрагменты html-страницы, поэтому для выделения нужной информации воспользуемся всё той же функцией find. Внутри собранных блоков множество других блоков, и чтобы добраться до искомой информации из объявления, нужно запустить функцию find несколько раз, меняя её аргументы.

Для обычных и "премиум"(на сайте есть функция платного размещения объявлений) объявлений тег с площадью недвижимости отличается, поэтому первое извлечение данных будет разбито на два условия. Для обычных объявлений площадь это блок <div class="c6e8ba5398–single_title–22TGT», а у премимум объявлений — блок <div class="c6e8ba5398–title–2CW78». Результат работы метода find по данному тегу запишем в переменную info. Для остальных данных блоки в обоих типах объявлений имеют одинаковую структуру, поэтому дальнейший парсинг для них не будет отличаться. В строке с площадью есть и данные о типе недвижимости — квартира, апартаменты, студия или свободная планировка, поэтому необходимо также произвести разбитие строки с площадью на саму площадь и на тип недвижимости — для это в Python есть функция split. Для каждого типа недвижимости

мы прописали отдельное разбиение и сохранили площадь в переменную square:

Листинг 2.2 — Извлечение площади недвижимости

```
for i in range(0, len(ads)):
    minimass = []
    info = ads[i].find('div',
       class_='c6e8ba5398--single_title--22TGT')
    if info == None:
        info = ads[i].find('div',
           class_='c6e8ba5398--title--2CW78')
    try:
        square = str(info).split('flat., ')[1].split(' м б ')[0]
    except IndexError:
        try:
            square = str(info).split('apart., ')[1].split('
               м б ')[0]
        except IndexError:
            try:
                square = str(info).split('studio, ')[1].split('
                   м б ')[0]
            except IndexError:
                ***
```

Аналогичным образом извлекаем ссылку на недвижимость (ad_url), цену (price) и дату публикации объявления (time):

Листинг 2.3 — Извлечение остальных параметров недвижимости

```
info2 = ads[i].find('div', class_='undefined
   c6e8ba5398--main-info--1SXZr')
if info2 == None:
    info2 = ads[i].find('div',
       class_='c6e8ba5398--info-section--QBF61')
ad_url = str(info2).split('href="')[1].split('" target=')[0]
info3 = ads[i].find('div', class_='c6e8ba5398--header--1dF9r')
if info3 == None:
    info3 = ads[i].find('div',
       class_='c6e8ba5398--header--1df-X')
```

```
8  price =
       str(info3).split('rub</div>')[0].split('>')[1].replace(" ",
       "").replace("rub", "")
9  info4 = ads[i].find('div', class_='c6e8ba5398--absolute--9uFLj')
10 time = str(info4).split('</div>')[0].split('>')[1]
```

2.2 Преобразование данных

Дата пубиликации объявления имеет формат вида «вчера, 17:55», либо «12 марта, 17:55», что делает её неудобной для какого-либо анализа. Поэтому необходимо представить дату в числовом в виде и с разделителями, которые будет удобно использовать. Для этого нами была написана функция normalize_time, которая на входе получает «вчера, 17:55», а возвращает «13:06:17:55», что обозначает, что объявление было опубликовано 13 июня в 17:15. Алгоритм работает следующим образом: если в строке с датой есть слово «сегодня», то оно подменяется на текущий месяц, двоеточие, сегодняшний день и еще одно двоеточие. Аналогично, для «вчера». Если же в дате присутствует «июн», то после дня ставится двоеточие, «6», еще одно двоеточие и время. Аналогично, для остальных месяцев.

2.2.1 Срок продажи недвижимости

Время публикации объявления открывает еще одну возможность собрать данные. На большинстве сайтов с объявлениями срок продажи недвижимости скрыт от конечных пользователей — обычно объявление удаляется сразу после продажи. Но нами был реализован алгоритм, который может определить этот срок.

Для этого был написан скрипт, который ежедневно запускает программу и проводит все описанные до этого момента действия (еще и сохраняет их, но об этом в следующей подглаве) и проверяет, есть ли собранные в прошлый раз объявления на сайте. Есть ли какого-то объявления нет, следовательно недвижимость продали, а значит можно определить срок продажи недвижимости — вычтя из сегодняшней даты дату публикации этого объявления. Получившийся формат времени —

«дни:месяцы:часи:минуты». Реализован данный алгоритм был в функции get_from_area, когда после сбора всех данных и их сохранения каждое объявление проверяется на то, не было ли оно сохранено во время прошлого сбора данных. Если да, то оно помечается удаленным и сохраняется в отдельный файл, в котором собираются все такие данные (для данного района Казани).

2.3 Сохранение данных

Собрав ссылку на объявления, время продажи, цену и площадь, необходимо сохранить эту информацию в файл, формат которого позволяет его использовать другими программами. Нами был выбран формат CSV, удовлетворяющий этому требованию. CSV является одним из самых широкоиспользуемых стандартов для связи между собой всезможных разнородных систем, для передачи и обработки объемных данных с табличной структурой, и во многих скриптовых языках программирования есть встроенные средства для его разбора и генерации. Он хорошо понятен как программистам, так и рядовым пользователям, а проблемы с самими данными в нем хорошо обнаруживаются.

Нами был написана функция write_csv, позволяющая сохранить собранные данные в формате CSV. После сбора всех данных применяется эта функция. На вход ей подается список data, в котором последовательно находятся ссылка, площадь, цена и время продажи недвижимости. С помощью метода writerow класса writer библиотеки csv производится запись данных в файле с расширением .csv. Внутри файла будет расположена таблица с нашими данными:

Листинг 2.4 — Сохранение собранных данных

```
1 def write_csv(data, area):
2     with open('.\\Data\\' + area + '.csv', 'a', newline='') as
        f:
3         writer = csv.writer(f, delimiter=' ', quotechar='"',
            quoting=csv.QUOTE_MINIMAL)
4         writer.writerow([data['ad_url'],
5                          data['square'],
6                          data['price'],
```

```
7                                   data['time']])
```

Теперь собранные данные представлены в удобном формате и их уже возможно анализировать, что будет продемонстрировано в следующей главе.

3 Пример возможного анализа собранных данных

В данной главе продемонстрирован использование одного из методов анализа собранных данных, полученных с html-страниц. Этим методом будет метод множественной линейной регрессии, с помощью которого мы научились предсказывать время, за которое можно продать недвижимость по её установленной цене.

3.1 О методе множественной линейной регрессии

Метод множественной линейной регрессии (МЛР) является одним из самых распространённых средств решения задач быстрого прогнозирования непрерывной величины Y (изначально предполагается, что она непрерывна) по входным параметрам $X_1, ..., X_n$. В данном методе связь величины Y с параметрами $X_1, ..., X_n$ задаётся при помощи линейной модели $Y = \beta_0 + \beta_1 X_1 + ... + \beta_n X_n + \epsilon$, где $\beta_0, \beta_1, ..., \beta_n$ — действительные регрессионные коэфф-ты, ϵ - случайная величина, представляющая из себя ошибку прогнозирования. Регрессионные коэфф-ты ищут по обучающей выборке

$$\widetilde{S}_t = s_1 = (y_1, x_1), ..., s_m = (y_m, x_m),$$

где $x_j = (x_{j1}, ..., x_{jn})$ вектор значений входных данных $X_1, ..., X_n$ для данного объекта s_j.

3.2 Применение метода МЛР

В языке Python есть библиотека sklearn (scikit-learn), реализующая метод МЛР. Ею мы и воспользывались. Используя сохранненные в CSV-файлы собранные данные, мы написали скрипт, прогнозирующий на основе цены недвижимости время, за которое на неё найдется покупатель (в нашем случае это срок продажи квартиры, но нет никаких трудностей в применении этого алгоритма и на других типах недвижимости).

Разделив все собранные данные на два массива (цены и срок продажи квартиры), мы построили линейную регрессионную модель с помощью библиотеки sklearn, а именно метода LinearRegression:

Листинг 3.1 — Построение регрессионной модели

```
model = linear_model.LinearRegression()
model.fit(x, y)
y_pred = model.predict(x)
print('predicted response:', y_pred, sep='\n')
x_new = np.array(st).reshape((-1, 1))
print(x_new)
y_new = model.predict(x)
print(y_new)
```

Далее мы сделали предсказание на основе массива цен на квартиры с помощью метода predict, чтобы получить информацию о том, за какое время они продадутся. Полученные данные мы сверили с изначальным массивом времени продажи (которые собрали с сайта), чтобы измерить качество предсказания — используя корень из среднеквартичного отклонения (RMSE):

```
 [193.28779505]
 [202.10566585]
 [193.3490614 ]
 [186.60191835]
 [215.55437576]
 [233.48598898]
 [189.06788874]
 [190.00182693]
 [188.91845863]
 [190.00182693]
 [186.60191835]
 [186.45286181]
 [187.0483408 ]
 [193.3490614 ]]
RMSE:
12.687960998517426
```

Рисунок 3.1 — Корень из среднеквадратичной ошибки

Отклонение в 12.7 это хорошое значение, которое значит, что предсказание срока продажи не может отклоняться в пределах 12 дней. На основе собранных данных, мы видим, что квартиры продаются в сред-

нем за 3-4 месяца, и 12 дней — это не такое большое отклонение. Таким образом, мы продемонстрировали, что собранные данные это хорошая выборка для анализа.

Заключение

В результате проделанной выпускной квалификационной работы стало ясно, как производить сбор данных с html-страниц с помощью языка Python. В итоге была достигнута цель работы, а именно создание программы сбора практически полезных данных с html-страниц сайта объявлений о недвижимости. Нами была написана программа, которая собирает и сохраняет данные из объявлений с сайта недвижимости структурированно и автоматизированно, без помощи пользователя. Она создает из собранной информации базу данных в популярном формате, который понятен большому числу программистов и системам обработки данных.

Было продемонстрировано, что собранные данные имеют практическую пользу — с помощью написанной нами программы мы научились предсказывать время, за которое найдется покупатель на квартиру, с низким значением погрешности. Исследованные в работе принципы и собранные данные могут быть использованы для получения важных выводов при анализе одного из крупнейших рынков современности — рынка недвижимости.

Список литературы

1) Блохин А.В. *Теория эксперимента* / А.В. Блохин. — Минск, Республика Беларусь: БГУ, 2002. – 67 с. — ISBN 985-445-815-6.

2) Гафаров Ф.М *Искусственные нейронные сети и приложения* / А.М. Шихалёв. — Казань: Изд-во Казан. ун-та, 2018. – 121 с.

3) Поручиков М.А. *Анализ данных* / М.А. Поручиков. — Самара: Изд-во Самарского университета, 2016. – 88 с.

4) Шихалёв А.М. *Регрессионный анализ. Парная линейная регрессия* / А.М. Шихалёв. — Казань: Изд-во Казан. ун-та, 2015. – 88 с.

5) Holden S. *Python Web Programming* / Steve Holden. — Индианаполис, США: Sams Publishing, 2002. — 720 с. — ISBN 978-1887902991.

6) *Python 3.7.3 documentation* / [Электронный ресурс]. — URL: https://www.docs.python.org/3/

7) Ruvalcaba Z., Boehm A. *Murach's HTML5 and CSS3* / Zak Ruvalcaba, Anne Boehm. — Фресно, США: Mike Murach Associates Inc, 2011. — 711 с. — ISBN 978-1-943872-26-8.

8) Zelle J. *Python Programming: An Introduction to Computer Science.* / John Zelle. — Портланд, США: Franklin, Beedle Associates Inc, 2004. — 517 с. — ISBN 978-1887902991.

Приложение А Код программы из 1 и 2 главы

```
import requests
from bs4 import BeautifulSoup
import csv
from datetime import datetime as dt
from datetime import timedelta
import os
from urllib.request import Request, urlopen
import random
from multiprocessing import Pool, TimeoutError
import time as timef

'''
1) Выясняем число страниц
 ↪  ---DONE---
2) Сформировываем список ссылок на каждую страницу поиска
 ↪  ---DONE---
3) Собираем данные с каждой страницы
 ↪  ---DONE---
4) Настраиваем прокси, чтобы не быть заблокированными
 ↪  ---DONE---
'''

headers = {
    'Accept - Language': 'ru - RU, ru; q = 0.9, en - US; q =
     ↪  0.8, en; q = 0.7',
    'Host': 'kazan.???.ru',
    'Referer': 'https://???.ru',
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
     ↪  AppleWebKit/537.36 (KHTML, like Gecko)
     ↪  Chrome/72.0.3626.96 Safari/537.36'
}

```

```
26 number_of_ads = 0
27 proxy_check = 1
28 user_agents = [
29     'Mozilla/5.0 (en-us) AppleWebKit/534.14 (KHTML, like
       ↪ Gecko; Google Wireless Transcoder) Chrome/9.0.597
       ↪ Safari/534.14',
30     'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
       ↪ AppleWebKit/537.36 (KHTML, like Gecko)
       ↪ Chrome/72.0.3626.121 Safari/537.36',
31     'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
       ↪ AppleWebKit/537.36 (KHTML, like Gecko)
       ↪ Chrome/71.0.3578.80 Safari/537.36',
32     'Mozilla/5.0 (Windows NT 6.1; Win64; x64)
       ↪ AppleWebKit/537.36 (KHTML, like Gecko)
       ↪ Chrome/71.0.3578.80 Safari/537.36',
33     'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
       ↪ AppleWebKit/537.36 (KHTML, like Gecko)
       ↪ Chrome/71.0.3578.98 Safari/537.36',
34     'Mozilla/5.0 (Windows NT 6.1; Win64; x64)
       ↪ AppleWebKit/537.36 (KHTML, like Gecko)
       ↪ Chrome/71.0.3578.98 Safari/537.36',
35     'Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML,
       ↪ like Gecko) Chrome/71.0.3578.98 Safari/537.3',
36     'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
       ↪ AppleWebKit/537.36 (KHTML, like Gecko)
       ↪ Chrome/70.0.3538.102 Safari/537.36',
37     'Mozilla/5.0 (Windows NT 6.1; Win64; x64)
       ↪ AppleWebKit/537.36 (KHTML, like Gecko)
       ↪ Chrome/70.0.3538.102 Safari/537.36',
38     'Mozilla/5.0 (Linux; Android 6.0.1; SM-J700M)
       ↪ AppleWebKit/537.36 (KHTML, like Gecko)
       ↪ Chrome/70.0.3538.80 Mobile Safari/537.36',
```

```
39      'Mozilla/5.0 (Linux; Android 8.1.0; Moto G (5))
        ↪ AppleWebKit/537.36 (KHTML, like Gecko)
        ↪ Chrome/70.0.3538.80 Mobile Safari/537.36',
40      'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_1)
        ↪ AppleWebKit/537.36 (KHTML, like Gecko)
        ↪ Chrome/70.0.3538.102 Safari/537.36',
41      'Mozilla/5.0 (Windows NT 5.1; rv:7.0.1) Gecko/20100101
        ↪ Firefox/7.0.1',
42      'Mozilla/5.0 (Windows NT 6.1; WOW64; rv:18.0)
        ↪ Gecko/20100101 Firefox/18.0']
43  good_proxies = []
44
45
46  def get_from_area(area):
47      if area == 'Avia':
48          url = 'https://kazan.???.ru/cat.php?deal_type=sale' \
49
                    ↪ '&district%5B0%5D=258&engine_version=2&object_type'
                    ↪ \
50                  '%5B0%5D=1&offer_type=flat&p=1&room1=1&room2=1'
                    ↪ \
51
                    ↪ '&room3=1&room4=1&room5=1&room6=1&room7=1&room9=1&wp
52      if area == 'Vahit':
53          url = 'https://kazan.???.ru/cat.php?deal_type=sale' \
54
                    ↪ '&district%5B0%5D=259&engine_version=2&object_type'
                    ↪ \
55                  '%5B0%5D=1&offer_type=flat&p=1&room1=1&room2=1'
                    ↪ \
56
                    ↪ '&room3=1&room4=1&room5=1&room6=1&room7=1&room9=1&wp
57      if area == 'Kirov':
```

```
58          url = 'https://kazan.???.ru/cat.php?deal_type=sale' \
59
                  ↪  '&district%5B0%5D=260&engine_version=2&object_type'
                  ↪  \
60                '%5B0%5D=1&offer_type=flat&p=1&room1=1&room2=1'
                  ↪  \
61
                  ↪  '&room3=1&room4=1&room5=1&room6=1&room7=1&room9=1&wp
62      if area == 'Moscow':
63          url = 'https://kazan.???.ru/cat.php?deal_type=sale' \
64
                  ↪  '&district%5B0%5D=261&engine_version=2&object_type'
                  ↪  \
65                '%5B0%5D=1&offer_type=flat&p=1&room1=1&room2=1'
                  ↪  \
66
                  ↪  '&room3=1&room4=1&room5=1&room6=1&room7=1&room9=1&wp
67      if area == 'Novo':
68          url = 'https://kazan.???.ru/cat.php?deal_type=sale' \
69
                  ↪  '&district%5B0%5D=262&engine_version=2&object_type'
                  ↪  \
70                '%5B0%5D=1&offer_type=flat&p=1&room1=1&room2=1'
                  ↪  \
71
                  ↪  '&room3=1&room4=1&room5=1&room6=1&room7=1&room9=1&wp
72      if area == 'Privol':
73          url = 'https://kazan.???.ru/cat.php?deal_type=sale' \
74
                  ↪  '&district%5B0%5D=263&engine_version=2&object_type'
                  ↪  \
75                '%5B0%5D=1&offer_type=flat&p=1&room1=1&room2=1'
                  ↪  \
```

```
76
                ↪  '&room3=1&room4=1&room5=1&room6=1&room7=1&room9=1&wp
77      if area == 'Sovet':
78          url = 'https://kazan.???.ru/cat.php?deal_type=sale' \
79
                ↪  '&district%5B0%5D=264&engine_version=2&object_type'
                ↪  \
80              '%5B0%5D=1&offer_type=flat&p=1&room1=1&room2=1'
                ↪  \
81
                ↪  '&room3=1&room4=1&room5=1&room6=1&room7=1&room9=1&wp
82
83      numb_of_pages = get_numb_of_pages(url)
84      url_split = url.split('p=1')
85
86      # full_url =
         ↪  url_split[0]+'p='+str(numb_of_pages)+url_split[1]
87      prox = '0'
88      headers = '0'
89      for i in range(1, numb_of_pages + 1):  # numb_of_pages +
         ↪  1):
90          cure_url = url_split[0] + 'p=' + str(i) + url_split[1]
91          print("Страница " + str(i) + ':')
92          attemts = 0
93          timef.sleep(random.randint(10, 15))
94          while True:
95              try:
96                  html = get_html(cure_url, prox, headers)
97                  get_data_from_html(html, area)
98                  break
99              except:
100                 print("Ошибка в get_from_area (главный метод)
                     ↪  "
```

```
101                         "при получении содержимого страницы.
                             ↪ Меняем прокси")
102                 prox = connect_to_good_proxy()
103                 headers = {'User-Agent': random_ua()}
104                 # прокси меняется в get_html
105
106     old = []
107     new = []
108
109     with open('.\\Data\\Old\\old' + area + '.csv') as csvfile:
110         reader = csv.reader(csvfile, delimiter=' ',
             ↪ quotechar='"', quoting=csv.QUOTE_MINIMAL)
111         for data in reader:
112             old.append(data)
113         # print(old)
114     with open('.\\Data\\Cure\\' + area + '.csv') as csvfile:
115         reader = csv.reader(csvfile, delimiter=' ',
             ↪ quotechar='"', quoting=csv.QUOTE_MINIMAL)
116         for data in reader:
117             new.append(data)
118
119     temp = False
120     n = 0
121     for i in old:
122         temp = True
123         for j in new:
124             # print(i)
125             # print(j)
126             if i[0] == j[0]:
127                 temp = False
128                 break
129         if temp == True:
130             n = n + 1
```

```
131             print('Удаленное ' + str(n) + ': ' + str(i))
132             tSplit = i[3].split(':')
133             oldDate = dt(2019, int(tSplit[1]), int(tSplit[0]),
                ↪  int(tSplit[2]), int(tSplit[3]))
134             curDate = dt.now()
135             raznicaDate = str(curDate - oldDate)
136             temp = raznicaDate.split(' days, ')
137             if temp != '':
138                 temp2 = temp[1].split(':')
139                 time = temp[0] + ':' + temp2[0] + ':' +
                    ↪  temp2[1]
140             else:
141                 time = '0' + ':' + temp[0] + ':' + temp[1]
142
143             data = {
144                 'ad_url': i[0],
145                 'square': i[1],
146                 'price': i[2],
147                 'time': time
148             }
149             write_csv(data, 'Del\\' + area)
150
151     os.rename('.\\Data\\Old\\old' + area + '.csv',
152               '.\\Data\\Old\\Archive\\' + 'old' + area + '_' +
                  ↪  str(dt.now()).replace(":", "_") + '.csv')
153     os.rename('.\\Data\\Cure\\' + area + '.csv',
        ↪  '.\\Data\\Old\\' + 'old' + area + '.csv')
154     # SystemExit(1)
155
156
157 def random_ua():
158     global user_agents
```

```
159     return user_agents[random.randint(0, len(user_agents) -
        ↪ 1)]
160
161
162 def get_proxies():
163     proxies_req = Request('https://www.sslproxies.org/')
164     proxies_req.add_header('User-Agent', random_ua())
165     proxies_doc = urlopen(proxies_req).read().decode('utf8')
166
167     soup = BeautifulSoup(proxies_doc, 'html.parser')
168     proxies_table = soup.find(id='proxylisttable')
169
170     # Save proxies in the array
171     proxies=[]
172     for row in proxies_table.tbody.find_all('tr'):
173         proxies.append("https://" +
            ↪ row.find_all('td')[0].string + ':' +
            ↪ row.find_all('td')[1].string)
174     return proxies
175
176
177 def get_good_proxies():
178     proxy = ''
179     attemts = 0
180     proxies = get_proxies()
181     for current_proxy in proxies:
182         # ua = UserAgent()
183         headers = {'User-Agent': random_ua()}
184         s = requests.session()
185         delay_time = 15  # delay time in seconds
186         try:
187             r = s.post('https://???.ru/', proxies=proxy,
                ↪ headers=headers, timeout=15)
```

```
188             except:
189                 proxies.remove(current_proxy)
190         global good_proxies
191         good_proxies = proxies
192 
193 
194 def connect_to_good_proxy():
195         global good_proxies
196         if len(good_proxies) != 0:
197             prox = good_proxies[random.randint(0,
                 ↪ len(good_proxies) - 1)]
198             proxyDict = {
199                 "https": prox
200             }
201             return proxyDict
202         else:
203             get_good_proxies()
204             if len(good_proxies) != 0:
205                 prox = good_proxies[random.randint(0,
                     ↪ len(good_proxies) - 1)]
206                 proxyDict = {
207                     "https": prox
208                 }
209                 return proxyDict
210             else:
211                 raise ValueError("Нет прокси")
212 
213 
214 def get_html(url, curr_proxy, prox_headers):
215         global headers
216         code = ''
217         s = requests.session()
218         if curr_proxy == '0':
```

```
219             r = s.get(url, headers=headers)
220             code = r.text
221         else:
222             r = s.get(url, headers=prox_headers,
                ↪  proxies=curr_proxy)
223             code = r.text
224         return code
225 
226 
227 def get_numb_of_pages(url):  # перебираем ссылки от
    ↪  n-страницы, пока не дойдем до существубщей страницы
228     global curr_proxy
229     url_split = url.split(
230         'p=1')  # (если страница не существует, то нас
                ↪  перебрасывает на 1 страницу, что и проверяется в
                ↪  этом цикле)
231     pages = ''
232     n = int(61)
233     cure_page = 0
234     print("Подсчет числа страниц...")
235     prox = '0'
236     headers = '0'
237     while True:
238         try:
239             while n != cure_page:
240                 n = n - 1
241                 html = get_html(url_split[0] + 'p=' + str(n) +
                    ↪  url_split[1], prox, headers)
242                 soup = BeautifulSoup(html, 'lxml')
243 
244                 pages = soup.find('ul',
                    ↪  class_='_93444fe79c-list--35Suf').find('li',
```

```
245
246                     'span')  # .find_all('li',
                        ↪ class_='_93444fe79c-list-item--2QgXB')[-1]
247                 cure_page =
                    ↪ str(pages).split('<span>')[1].split('</span>')[0]
248                 cure_page = int(cure_page)
249                 n = int(n)
250                 print("Страниц меньше, чем " + str(n))
251                 timef.sleep(random.randint(10, 16))
252             break
253         except:
254             print("Капча при подсчета числа страниц, меняем
                ↪ прокси и пытаемся еще")
255             prox = connect_to_good_proxy()
256             headers = {'User-Agent': random_ua()}
257             n = n + 1
258     print("Всего " + str(n) + " страниц для данного района")
259     return n
260
261
262 def get_data_from_html(html, area):
263     global number_of_ads
264
265     soup = BeautifulSoup(html, 'lxml')
266     ads = soup.find('div',
        ↪ class_='_93444fe79c-wrapper--1Z8Nz').find_all('div',
        ↪ class_='_93444fe79c-card--2Jgih')
267
268     temp_list = []
269     for i in range(0, len(ads)):
270         minimass = []
```

```
271             info = ads[i].find('div',
                ↪  class_='c6e8ba5398--single_title--22TGT')
272             if info == None:
273                 info = ads[i].find('div',
                    ↪  class_='c6e8ba5398--title--2CW78')
274             try:
275                 square = str(info).split('кв., ')[1].split('
                    ↪  м²')[0]
276             except IndexError:
277                 try:
278                     square = str(info).split('апарт.,
                        ↪  ')[1].split(' м²')[0]
279                 except IndexError:
280                     try:
281                         square = str(info).split('Студия,
                            ↪  ')[1].split(' м²')[0]
282                     except IndexError:
283                         try:
284                             square = str(info).split('Своб.
                                ↪  планировка, ')[1].split(' м²')[0]
285                         except IndexError:
286                             info = ads[i].find('div',
                                ↪  class_='c6e8ba5398--subtitle--UTwbQ')
287                             try:
288                                 square = str(info).split('кв.,
                                    ↪  ')[1].split(' м²')[0]
289                             except IndexError:
290                                 try:
291                                     square =
                                        ↪  str(info).split('апарт.,
                                        ↪  ')[1].split(' м²')[0]
292                                 except IndexError:
293                                     try:
```

```
294                                square =
                                    ↪ str(info).split('Студия,
                                    ↪ ')[1].split(' м²')[0]
295                            except IndexError:
296                                try:
297                                    square =
                                        ↪ str(info).split('Своб.
                                        ↪ планировка,
                                        ↪ ')[1].split('
                                        ↪ м²')[0]
298                                except IndexError:
299                                    print(info)
300                                    print(ads[i])
301            info2 = ads[i].find('div', class_='undefined
                ↪ c6e8ba5398--main-info--1SXZr')
302            if info2 == None:
303                info2 = ads[i].find('div',
                    ↪ class_='c6e8ba5398--info-section--QBF61
                    ↪ c6e8ba5398--main-info--1SXZr')
304            ad_url = str(info2).split('href="')[1].split('"
                ↪ target=')[0]
305            info3 = ads[i].find('div',
                ↪ class_='c6e8ba5398--header--1dF9r')
306            if info3 == None:
307                info3 = ads[i].find('div',
                    ↪ class_='c6e8ba5398--header--1df-X')
308            price =
                ↪ str(info3).split('[U+20BD]</div>')[0].split('>')[1].replac
                ↪ ", "").replace("[U+20BD]", "")
309            info4 = ads[i].find('div',
                ↪ class_='c6e8ba5398--absolute--9uFLj')
310            time = str(info4).split('</div>')[0].split('>')[1]
311            time = normalize_time(time)
```

```
312             data = {
313                 'ad_url': ad_url,
314                 'square': square,
315                 'price': price,
316                 'time': time
317             }
318             print(data)
319             write_csv(data, 'Cure\\' + area)
320             number_of_ads = number_of_ads + 1
321             print(number_of_ads)
322
323
324     def write_csv(data, area):
325         with open('.\\Data\\' + area + '.csv', 'a', newline='') as
            ↪ f:
326             writer = csv.writer(f, delimiter=' ', quotechar='"',
                ↪ quoting=csv.QUOTE_MINIMAL)
327             writer.writerow([data['ad_url'],
328                              data['square'],
329                              data['price'],
330                              data['time']])
331
332
333     def normalize_time(time):
334         if time.find("сегодня") != -1:
335             time = time.replace("сегодня, ",
336                                 str(dt.now().day) + ":" +
                                    ↪ str(dt.now().month) + ":")
337         if time.find("вчера") != -1:
338             time = time.replace("вчера, ",
339                                 str(dt.now().day - 1) + ":" +
                                    ↪ str(dt.now().month) + ":")
340         if time.find("янв") != -1:
```

```
341             time = time.replace(" янв, ", ":1" + ":")
342         if time.find("фев") != -1:
343             time = time.replace(" фев, ", ":2" + ":")
344         if time.find("мар") != -1:
345             time = time.replace(" мар, ", ":3" + ":")
346         if time.find("апр") != -1:
347             time = time.replace(" апр, ", ":4" + ":")
348         if time.find("мая") != -1:
349             time = time.replace(" мая, ", ":5" + ":")
350         if time.find("июн") != -1:
351             time = time.replace(" июн, ", ":6" + ":")
352         if time.find("июл") != -1:
353             time = time.replace(" июл, ", ":7" + ":")
354         if time.find("авг") != -1:
355             time = time.replace(" авг, ", ":8" + ":")
356         if time.find("сен") != -1:
357             time = time.replace(" сен, ", ":9" + ":")
358         if time.find("окт") != -1:
359             time = time.replace(" окт, ", ":10" + ":")
360         if time.find("ноя") != -1:
361             time = time.replace(" ноя, ", ":11" + ":")
362         if time.find("дек") != -1:
363             time = time.replace(" дек, ", ":12" + ":")
364         return time
365
366
367 def everyday_process():
368     get_good_proxies()
369
370     params = ['Avia', 'Vahit', 'Kirov', 'Moscow', 'Novo',
        ↪  'Privol', 'Sovet']
371     pool = Pool(processes=7)
```

```
372     processes = [pool.apply_async(get_from_area, [i]) for i in
        ↪ params]
373     results = []
374     for process in processes:
375         results.append(process.get(timeout=3600))
376 
377 
378 def main():
379     pereriv = 0
380     while True:
381         x = 0
382         with open('.\\last_parse.csv') as csvfile:
383             reader = csv.reader(csvfile, delimiter=' ',
                ↪ quotechar='"', quoting=csv.QUOTE_MINIMAL)
384             for data in reader:
385                 x = data[0]
386         x = dt.strptime(x, "%Y-%m-%d %H:%M:%S.%f")
387         x = x + timedelta(hours=pereriv)
388         # y = x.replace(day=x.day, hour=x.hour+pereriv,
            ↪ minute=x.minute + 21, second=x.second,
            ↪ microsecond=x.microsecond)
389         if dt.now() > x:
390             try:
391                 everyday_process()
392                 print('hah')
393                 with open('.\\last_parse.csv', 'w+',
                    ↪ newline='') as f:
394                     writer = csv.writer(f, delimiter=' ',
                        ↪ quotechar='"')
395                     writer.writerow([dt.now()])
396                 pereriv = 5
397             except TimeoutError:
398                 pereriv = 1
```

```
399
400
401 if __name__ == '__main__':
402     # print(str(datetime.now().day))
403     try:
404         main()
405
406     except Exception as e:
407         print(e)
```

Приложение Б Код программы из 3 главы

```
import numpy as np
from sklearn import linear_model
import csv
from sklearn.metrics import mean_squared_error
from math import sqrt

del_home = []

with open('.\\Data\\Del\\' + "Vahit" + '.csv') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ', quotechar='"',
     ↪  quoting=csv.QUOTE_MINIMAL)
    for data in reader:
        del_home.append(data)
    print(del_home)

for r in del_home:
    del r[0]
    del r[0]
    r[1] = str(r[1])[:-3]
    print(r[1])
    a = str(r[1]).split(":")
    r[1] = int(a[0]) * 24 + int(a[1])
    r[0] = int(r[0])
    if r[0] < 0 or r[1] < 0:
        #r[0] = str(r[0])
        #r[1] = str(r[1])
        del_home.remove(r)

del_home.pop(149)
sold = [x[:] for x in del_home]

for r in del_home:
```

```
32      del r[1]
33
34  for r in sold:
35      del r[0]
36      '''if r[0] < 0:
37          index = sold.index(r)
38          del_home.pop(index)
39          sold.pop(index)
40      '''
41
42  x = np.array(del_home)
43  y = np.array(sold)
44
45  print(x)
46  print(y)
47  model = linear_model.LinearRegression()
48  model.fit(x, y)
49
50  y_pred = model.predict(x)
51  print('predicted response:', y_pred, sep='\n')
52
53  st = [1000000, 1500000, 2000000, 3100000, 5000000, 10000000]
54
55  x_new = np.array(st).reshape((-1, 1))
56  print(x_new)
57  y_new = model.predict(x)
58  print(y_new)
59
60  print(y)
61  print(y_new)
62  print("RMSE: ")
63  print(sqrt(mean_squared_error(y, y_new)))
```