# ПРИЛОЖЕНИЕ А

**Функция для геокодирования**

**import time**

**from** selenium **import** webdriver

**from** selenium.webdriver.common.by **import** By

**import** pandas **as** pd

**import** requests

**import** warnings

warnings.simplefilter(action**=**'ignore', category**=**FutureWarning)

driver **=** webdriver.Chrome()

hrefs **=** []

**try**:

**for** page **in** range(1,50):

        driver.get(url**=f**'https://spb.cian.ru/cat.php?deal\_type=sale&engine\_version=2&offer\_type=offices&office\_type%5B0%5D=2&office\_type%5B1%5D=4&office\_type%5B2%5D=5&office\_type%5B3%5D=7&office\_type%5B4%5D=9&office\_type%5B5%5D=12&p={page}&region=2&sort=creation\_date\_desc')

        time.sleep(10)

        print('main base loaded')

*#   Постепенная прогрузка страницы*

        skrol **=** 0

**for** i **in** range(15):

            skrol **+=** 1000

            driver.execute\_script(**f**"window.scrollTo(0, {skrol});")

            time.sleep(1)

        time.sleep(10)

        print(**f**'page {page} loaded')

*#   Взятие href с каждой страницы*

**for** i **in** range(1, 34):

**try**:

                        item **=** driver.find\_element(By.XPATH, **f**'/html/body/div[1]/div/div[5]/div[{i}]/div/div[2]/div/div[1]/div[1]/div[1]/h3/div/a').get\_attribute('href')

                        hrefs.append(item)

                        time.sleep(1)

**except** Exception **as** inner\_ex:

                        print(**f**'Error in inner loop: {inner\_ex}')

**continue**

**except** Exception **as** ex:

    print(**f**'Error in outer loop: {ex}')

driver.close()

driver.quit()

# ПРИЛОЖЕНИЕ Б

**Парсер объявлений**

driver **=** webdriver.Chrome()

data\_spec\_all **=** {}

**try**:

**for** k **in** range(655,1349):

*#         Инициализация селениума*

        driver.get(url**=**hrefs[k])

        time.sleep(10)

*#         Название*

**try**:

            title **=** driver.find\_element(By.XPATH, '/html/body/div[2]/div/div[2]/div[2]/section/div/div/div[1]/div/h1').text

**except** Exception **as** title\_exception:

            title **=** **None**

*#         Цены*

**try**:

            price **=** driver.find\_element(By.XPATH,"/html/body/div[2]/div/div[2]/div[3]/div/div[1]/div[1]/div[3]/div/div[1]/span").text

**except** Exception **as** price\_exception:

            price **=** **None**

**try**:

            square\_price **=** driver.find\_element(By.XPATH,"/html/body/div[2]/div/div[2]/div[3]/div/div[1]/div[3]/div/div/div[1]/span[2]").text

**except** Exception **as** sq\_price\_exception:

            square\_price **=** **None**

**try**:

            squares **=** driver.find\_element(By.XPATH,"/html/body/div[2]/div/div[2]/div[2]/div[3]/div[1]/div[2]/span[2]").text

**except** Exception **as** sq\_exception:

            squares **=** **None**

**try**:

            floor **=** driver.find\_element(By.XPATH,"/html/body/div[2]/div/div[2]/div[2]/div[3]/div[2]/div[2]/span[2]").text

**except** Exception **as** floor\_exception:

            floor **=** **None**

**try**:

            free **=** driver.find\_element(By.XPATH,"/html/body/div[2]/div/div[2]/div[2]/div[3]/div[3]/div[2]/span[2]").text

**except** Exception **as** free\_exception:

            free **=** **None**

**try**:

            city **=** driver.find\_element(By.XPATH, '/html/body/div[2]/div/div[2]/div[2]/section/div/div/div[2]/address/div/div/a[1]').text

**except** Exception **as** city\_exception:

            city **=** **None**

**try**:

            district **=** driver.find\_element(By.XPATH, '/html/body/div[2]/div/div[2]/div[2]/section/div/div/div[2]/address/div/div/a[2]').text

**except** Exception **as** distr\_exception:

            district **=** **None**

**try**:

            okrug **=** driver.find\_element(By.XPATH, '/html/body/div[2]/div/div[2]/div[2]/section/div/div/div[2]/address/div/div/a[3]').text

**except** Exception **as** okrug\_exception:

            okrug **=** **None**

**try**:

            street **=** driver.find\_element(By.XPATH, '/html/body/div[2]/div/div[2]/div[2]/section/div/div/div[2]/address/div/div/a[4]').text

**except** Exception **as** street\_exception:

            street **=** **None**

**try**:

            house\_number **=** driver.find\_element(By.XPATH, '/html/body/div[2]/div/div[2]/div[2]/section/div/div/div[2]/address/div/div/a[5]').text

**except** Exception **as** house\_exception:

            house\_number **=** **None**

*#         Url продавца*

**try**:

            seller **=** driver.find\_element(By.XPATH, '/html/body/div[2]/div/div[2]/div[3]/div/div[3]/div/div/div[1]/div/div[2]/div/div/div[1]/a').get\_attribute('href')

**except** Exception **as** seller\_exception:

            seller **=** **None**

**try**:

            description **=** driver.find\_element(By.XPATH, '/html/body/div[2]/div/div[2]/div[2]/div[5]/div/div/div/div/span').text

**except** Exception **as** description\_exception:

            description **=** **None**

*#         Метро*

**try**:

            sub\_near **=** []

            sub\_list **=** driver.find\_element(By.XPATH, '/html/body/div[2]/div/div[2]/div[2]/section/div/div/div[2]/address/ul[1]')

*# Обработка данных списка, если он найден*

**for** li **in** sub\_list.find\_elements(By.TAG\_NAME, 'li'):

                sub **=** li.find\_element(By.TAG\_NAME, 'a').text

                time\_value **=** li.find\_element(By.XPATH, '/html/body/div[2]/div/div[2]/div[2]/section/div/div/div[2]/address/ul[1]/li[1]/span').text

                sub\_near.append([sub,time\_value])

**except** Exception **as** sub\_exception:

            sub\_near **=** **None**

*#         Шоссе*

**try**:

            highway\_near **=** []

            way\_list **=** driver.find\_element(By.XPATH, '/html/body/div[2]/div/div[2]/div[2]/section/div/div/div[2]/address/ul[2]')

*# Обработка данных списка, если он найден*

**for** li **in** way\_list.find\_elements(By.TAG\_NAME, 'li'):

                way **=** li.find\_element(By.TAG\_NAME, 'a').text

                dist\_value **=** li.find\_element(By.XPATH, '/html/body/div[2]/div/div[2]/div[2]/section/div/div/div[2]/address/ul[2]/li[2]/span').text

                highway\_near.append([way,dist\_value])

**except** Exception **as** highway\_exception:

            highway\_near **=** **None**

        new\_data **=** {'url': hrefs[k],'seller\_url':seller,'title':title,'squares':squares, 'full\_price': price,

                    'price\_per\_square': square\_price,'city':city, 'district':district, 'okrug': okrug, 'street': street ,'house\_number':house\_number,

                    'description': description,'sub\_near':sub\_near,'highway\_near':highway\_near,'floor':floor,'free':free}

        cian **=** cian.append(new\_data, ignore\_index**=True**)

        time.sleep(5)

**except** Exception **as** ex:

    print(ex)

**finally**:

    driver.close()

    driver.quit()

# ПРИЛОЖЕНИЕ В

**Функция для геокодирования**

**import** pandas **as** pd

**import** numpy **as** np

**import** requests

**def** get\_coordinates(address):

    api\_key **=** "\_\_\_"

    url **=** **f**"https://geocode-maps.yandex.ru/1.x/?apikey={api\_key}&geocode={address}&format=json"

    response **=** requests.get(url)

    data **=** response.json()

**if** response.status\_code **==** 200:

        found **=** int(data['response']['GeoObjectCollection']['metaDataProperty']['GeocoderResponseMetaData']['found'])

**if** found **>** 0:

            coords\_str **=** data['response']['GeoObjectCollection']['featureMember'][0]['GeoObject']['Point']['pos']

            coordinates **=** tuple(map(float, coords\_str.split()))

**return** coordinates

**else**:

            print("Объекты по заданному адресу не найдены.")

**return** **None**

**else**:

        print("Ошибка при получении координат.")

**return** **None**

**for** i **in** range(0,len(df)):

    address **=** df.iloc[i,**-**3]

    coordinates **=** get\_coordinates(address)

    df.at[i, 'latitude'], df.at[i, 'longitude'] **=** coordinates

# ПРИЛОЖЕНИЕ Г

**Функция введения фиктивных переменных по районам**

**import** pandas **as** pd

**from** shapely.geometry **import** Point, Polygon

**from** shapely.wkt **import** loads

rk **=** pd.read\_csv(**r**'C:\Users\pelik\python\diplomopis\spb\_geo\boundary-polygon-land-lvl5.csv')

rk **=** rk.rename(columns**=**{'WKT\tNAME\tNAME\_EN\tNAME\_RU\tADMIN\_LVL\tOSM\_TYPE\tOSM\_ID\tADMIN\_L1D\tADMIN\_L1\tADMIN\_L2D\tADMIN\_L2\tADMIN\_L3D\tADMIN\_L3\tADMIN\_L4D\tADMIN\_L4\tADMIN\_L5D\tADMIN\_L5\tADMIN\_L6D\tADMIN\_L6\tADMIN\_L7D\tADMIN\_L7\tADMIN\_L8D\tADMIN\_L8\tADMIN\_L9D\tADMIN\_L9\tADMIN\_L10D\tADMIN\_L10\toktmo\tokato':'polygon'})

districts **=** []

**for** i **in** range (18):

    string **=** rk.iat[i,0]

    english\_value **=** string.split('\t')[2]

    english\_value **=** english\_value.strip()

    english\_value **=** english\_value.replace(' District','')

    districts.append(english\_value)

rk['polygon'] **=** rk['polygon'].apply(loads)

**def** point\_in\_polygon(point):

**for** polygon **in** range(len(rk['polygon'])):

**if** rk['polygon'].iloc[polygon].contains(point):

            ror[districts[polygon]][i] **=** 1

**return** **False**

# ПРИЛОЖЕНИЕ Д

**Функция для геокодирования**

**def** extract\_floors(text):

**if** pd.isna(text):

**return** np.nan

    match **=** re.match(**r**'(\d**+**)\s**\***из\s**\***(\d**+**)**|**(\d**+**)', str(text))

**if** match:

        current\_floor **=** int(match.group(1) **or** match.group(3))

        max\_floor **=** int(match.group(2) **or** match.group(3))

**return** current\_floor, max\_floor

**return** np.nan

df[['current\_floor', 'max\_floor']] **=** pd.DataFrame(df['floor'].apply(extract\_floors).tolist(), index**=**df.index)

df