**PROJECT DOCUMENTATION**

**Table of Contents**

1. Background
2. Overview
3. Problem Statement
4. Project Objectives
5. Project Approach
6. Project Deliverables
7. Methodology
8. Data Gathering
9. Data Cleaning
10. Data Merging
11. Conclusion
12. Result
13. Limitations and Challenges.
14. **BACKGROUND**

**Overview**

Mappa is a company that aims to redefine the experience of finding a new home by using the power of data and technology. Their goal with this project is to create a master list of listed properties across the Greater London area (United Kingdom) in one dataset. The purpose of this dataset is to provide house hunters with a comprehensive and up-to-date list of available properties, enabling them to make informed decisions and secure the best value for their money.

**Project Objectives**

The main objective of this project is to extract and blend property datasets across the Greater London area in one dataset. This involves scraping data from various sources, cleaning and processing the data, and then combining it into a single, comprehensive dataset. The specific objectives of the project are:

1. To identify and scrape property data from various sources across the Greater London area.
2. To clean and pre-process the scraped data to ensure consistency and accuracy.
3. To combine the scraped data into a single, comprehensive dataset.
4. To make the final dataset available for use by house hunters.

**Project Approach**

To achieve the project objectives, we will be using web scraping techniques to extract property data from various sources, including property listing websites and real estate agents' websites. We will then use data cleaning and pre-processing techniques to ensure that the data is consistent and accurate. Once the data has been cleaned and pre-processed, we will merge the data into a single dataset using data blending techniques. Finally, we will make the dataset available for use by house hunters.

**Project Deliverables**

The main deliverable of this project is the comprehensive dataset of listed properties across the Greater London area. This dataset will be made available in a format that can be easily accessed and used by house hunters. In addition, we will also provide documentation on the data sources used, the data cleaning and pre-processing techniques used, and the data blending techniques used to create the final dataset.

1. **METHODOLOGY**

For this project, we followed a three-stage approach, which included:

1. Data Gathering
2. Data cleaning, and
3. Data merging.

**1. Data Gathering**

The scraping process began by utilizing the Beautifulsoup package for data gathering. However, the websites being scraped heavily depend on JavaScript, requiring the implementation of alternative strategies. To address this, the web scraping process was carried out using the Selenium library in Python. Selenium is an effective tool for extracting information from dynamic websites that heavily rely on JavaScript to generate content. By utilizing Selenium, interactions with the website, searches, and extraction of the desired data were made possible.

The first step was to download and Install Selenium drivers or Chrome (the driver is also available for other browsers). The link to download the driver is here.

For each of the three websites, the web scraping process was divided into two stages: Properties for sale (Sales) and properties for rent (Rent). In addition, a web scraping function was created for each stage for all websites, making it a total of six functions.

Taking the web scraping process for Right Move as a case study. The two functions for sales and rent are rightMove\_sales and rightMove\_rent, respectively. Both functions scrape property data from Rightmove for a given list of postcodes and transaction types and return the data as a pandas DataFrame. Here are the four arguments the function takes:

1. Postcodes -- a list of postcodes for which to scrape property data

2. Trans\_type -- the transaction type of the properties to be scrapped ('sales' or 'rent')

3. Website -- the name of the website being scraped (in this case, 'Rightmove')

4. df -- an empty panda DataFrame to store the scraped data

Inside the function, the ChromeDriver executable is set as the WebDriver, and the ChromeDriver is launched with the specified service. Then, the function navigates to the OnTheMarket website and interacts with the user interface to initiate a search for properties.

Using the Selenium WebDriver, the function locates and interacts with different elements on the webpage, such as canceling pop-up windows, maximizing the window, selecting the transaction type, entering the postcode in the search bar, and clicking the search button.

To ensure all relevant data is obtained, the function goes through multiple pages of search results. First, it scrapes property details such as an address, property type, number of bedrooms and bathrooms, price, description, listing date, agent details, property URL, and website source. These details are stored in individual lists.

After scraping the data from each page, the function scrolls down to load more properties and clicks on any pop-up windows that may appear. It then finds the next button to navigate to the next page of results, repeating this process until all pages have been scrapped.

This same technique is used for the other two websites. Finally, the scraped data for all postcodes are stored in a Pandas DataFrame. The scraped data has 13 different features, namely:

1. Unique Id: Consist of the postal code, the transaction type, the scrapping serial number, and the website name. E.g., For BR1S00001OM, BR1 - Postal code, S - Sales, 00001 - Serial number, OM - On the Market Website.

2. Location

3. Transaction type

4. Property type

5. Address

6. Number of bedrooms

7. Number of Bathrooms

8. Price

9. Description

10. Listing date

11. Agent name

12. Listing source

13. Property URL

14. Current date: The date the web scraping was performed (Applicable to only On the mar website)

The web scraping function was enhanced with informative print statements to monitor the progress for each website page and postal code. This feature allows for easy tracking and ensures that in case of network issues or interruptions, the scraping process can be resumed from the point where it left off. Figure 2.1 provides a visual representation of these print statements for reference. The Datasets obtained from the scrapping were then saved into a Comma-Separated-Value (CSV) file.

N.B.: When web scraping the Zoopla website, the installation of an undetected chrome driver was necessary to bypass the website security due to its protection against third-party applications using Cloudware.

Next, we will move on to the data cleaning stage to ensure the scraped data is consistent and accurate for further analysis.

**2. Data Cleaning**

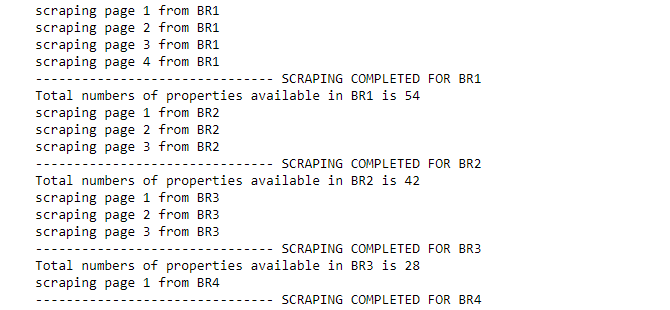


Figure 2.1: Web scraping function print statement references.