Mini Project I: Where do I book an accommodation?

Mudita Shakya

1. Introduction

Given a destination, check in date, and check out date, we want to be able to get the best possible accommodation deal for a trip. Websites such as Booking.com, Hotels.com, Expedia, and others offer the chance to look into the different hotels available near the destination with the information about the price, location, along with user review data. However, it would be better if we had a process to compare the accommodation deals offered by the different sites.

To solve this issue, we can get the data from these websites and apply data processing steps to get the results we want. The data from these websites will be obtained through web scraping, which is when we extract the data from a website using either the HTML structures of the website or through the APIs used by the website. Through this, we can get all the data in one place to compare and get the best deal possible.

2. Data collection

I collected the accommodations data from three websites namely:

i. Booking.com
ii. Expedia.fi
iii. OneTravel.com

First, I set the following the predefined values for the process.

Destination: London

Check in date: 10th November 2022

Check out date: 11th November 2022

Using these values, I scraped the available accommodation data from the three websites with main details such as: **name of the hotel**, **address**, **distance from city center**, **number of stars**, **review score**, **average price**, **hotel description** and **images**. These details were collected for **50** hotels presented in the search results for each website.

The tools I used for scraping the data were

- BeautifulSoup
- Selenium

Before starting the scraping process, it was important to check for the terms of use for the website. The websites: Booking.com and OneTravel.com do not allow for scraping to be done for commercial purposes. Expedia does not allow does not allow scraping of any kind. I checked this for other websites as well. The same was the case for the other websites such as: Agoda and Hotels.com.

For getting the necessary information, I checked the class names and CSS selectors for each required element. It was also necessary to change the currency to get the prices in euros. To get some of the additional details, it was also necessary to scrape the linked page for the specific hotel. For example: in all three websites that were scraped, the hotel description and images were not available with the main search results. For this, I had to also scrape the data from the page for the hotel.

The data collected were saved as separate pandas data frames for each website.

3. Data Processing

The collected data had to be cleaned before it could be used. The three data frames prepared were concatenated. Since we had to compare the data from the three websites, it was important that the data would be on the same scale. For instance, it was necessary to make sure that the price in the data had the same unit of currency. This was handled during the data collection phase. For the hotel rating score, Booking.com and OneTravel.com had the scale from 0-10. However, the scale for Expedia was from 0-5. This was corrected by changing the values from the data from Expedia to be on the same scale as that of the other websites.

I checked the dataset for missing values and checked the data types for all the columns.

I updated the price, distance from center, and review score columns so that only numeric values were maintained. The hotel name column values were preprocessed by removing leading and trailing whitespace characters and changing it to lowercase. Then I removed the duplicates based on hotel name from the dataset. This reduced the size of the dataset from 150 to 122.

4. Data Analysis

For data analysis, I first investigated the description of the numerical data. The mean price of the accommodation was found to be 225.53 € with a minimum of 16 € and a maximum of 1298 €. On average, the hotels were 2.04 miles from the center and the mean review score was 7.85. I then looked at the distribution of the numerical values by plotting histograms for price, number of stars, review score and distance from center.

Looking at the price distribution, it can be concluded that most of the accommodations are priced below 100 €.

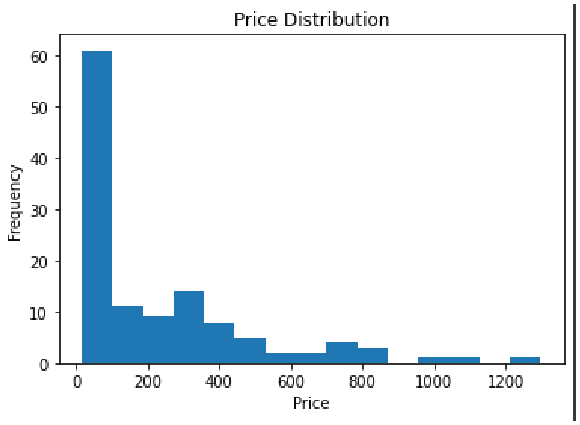

Fig 1. Histogram to show frequency distribution of price

From the distribution for number of stars, it can be gathered that the number of stars data was missing for a lot of the search results.

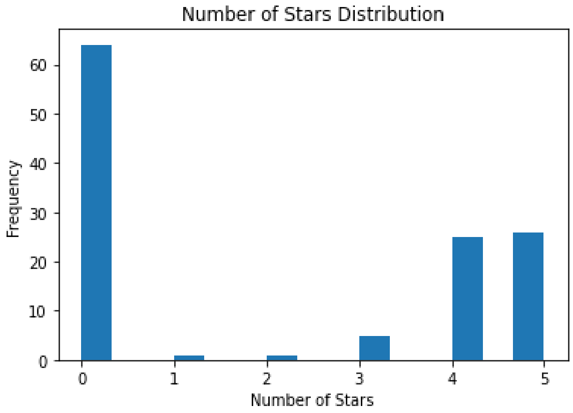

Fig 2. Histogram to show frequency distribution of number of stars

I used heatmaps to plot the correlation matrix between the following pairs of data:

i. Review score and Price
ii. Number of stars and Price
iii. Distance from center and Price

Review score and price have a moderately positive correlation.

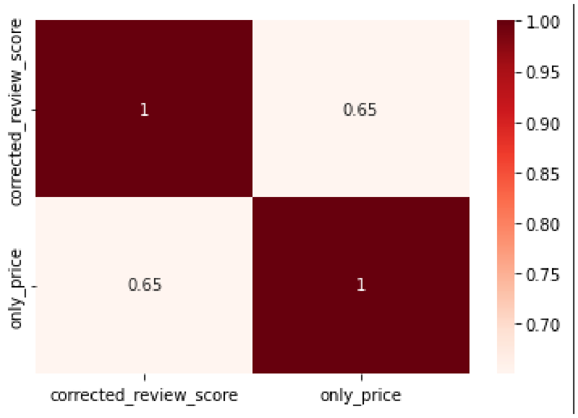

Fig 3. Heat map for correlation matrix of review score and price

Number of stars and price have a strong positive correlation.

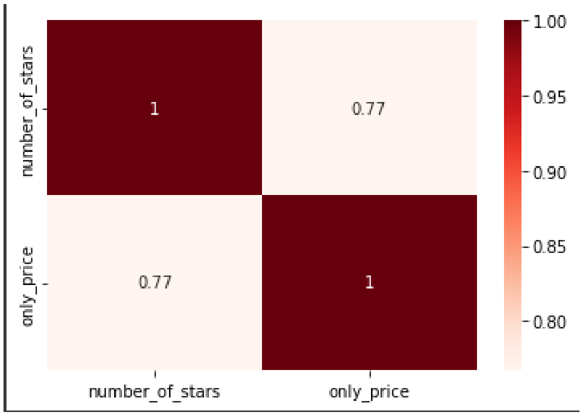

Fig 4. Heat map for correlation matrix of number of stars and price

As expected, distance from center and price have negative correlation which means as the distance from center decreases, the price of the accommodation increases.

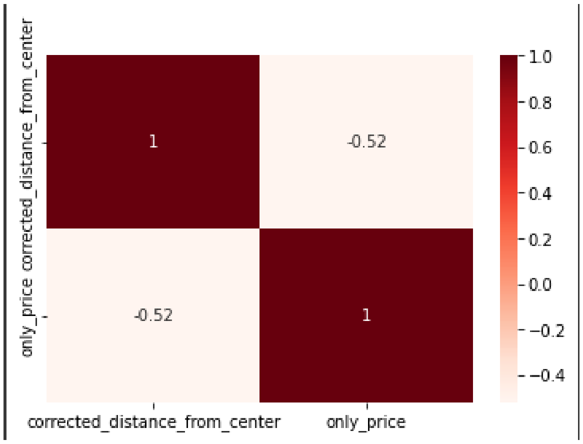

Fig 5. Heat map for correlation matrix of distance from center and price

5. Conclusion

Web scraping helps to extract useful data from different sites. However, as data is a valuable resource and companies are aware of web scraping practices, websites these days enforce measures to stop web scraping practices.

The above-mentioned websites that I scraped are dynamic websites which require user input to get the required results for example: the search results for the destination on the required date range. It is more difficult to get the required data items from these websites. I solved this by looking into the request URL for the website and using the request URL prepared after applying the search terms for the scraping.

I also tried scraping websites other than those listed above. Some websites like Hotellook.com returns forbidden request (403) error code when trying to access the website through Python requests library and through Selenium.

Some websites use lazy loading for getting the data from the backend which means that there is a delay so, when you first try getting the source of the website without waiting for all the content to be loaded, you don’t find all the required elements. This can be solved by applying an explicit wait using Selenium to check and wait until the required element has loaded before proceeding.

Through this project, I was able to come across different issues that one can face during web scraping and how to overcome them and practice the data processing, exploratory data analysis (EDA) and data visualization steps on real world data. I was able to extract the data to compare the accommodation deals offered.