Introduction to Programming
Master Program in Data Science
and Advanced Analytics 2018/19

Price Prediction on Uniplaces.com

Project Group #3: Ekaterina Mylnikova[1], Marius Löwe[2], Bryan Balzuweit[3] & Joana Lorenz[4]

[1]M20180207, ekaterina.mylnikova@gmail.com

[2]M20180410, marius.loewe@whu.edu

[3]M20180204, bryan.balzuweit@gmail.com

[4]M20180412, joana.lorenz@outlook.de

Abstract: The aim of our project was to provide a decision-support tool for students seeking accommodation in Lisbon by predicting the competitive price of a housing offer, based on the features desired by the user. Similar existing projects had not focused on creating a user-friendly price prediction tool and were not targeted specifically towards Lisbon students.

We based our analysis on data from one of the largest accommodation search-sites for students coming to Lisbon: uniplaces.com. The data from uniplaces were scraped using the query language GraphQL. After enrichening our analysis with data from additional sources, several regression models were implemented and tested. The most accurate regression model was then used as a base to create a prototype interactive user-interface for the price prediction.

Overall, the model received a good R^2 on the test set and seems to give a good price indication for new user input. Some unexpected behavior that was observed in some cases could be due to the comparably small amount of data points, the influence of extreme values and the small proportion of continuous variables in the model.

Overall, we were able to broaden the existing research by providing a front-to-end approach for a user-facing price prediction tool based on uniplaces data. The resulting approach can be used as a base for a more elaborate user-facing model with a HTML web-implementation or to create further pricing analyses. Furthermore, the comprehensive scraping approach for uniplaces listings can be reused for diverse future projects that require information from the website.

Keywords: web-scraping; price-prediction

Statement of Contribution: All group members contributed to the project through discussions and partial participation in all project steps. Nevertheless, main contributions to the project were as follows:

Ekaterina Mylnikova:
Regression analysis | price prediction & visualization

Bryan Balzuweit:
Geopy scraping (zip-codes) | regression analysis

Marius Löwe:
Data cleaning | data exploration | regression analysis | report

Joana Lorenz:
Scraping of the uniplaces website | data cleaning | web-scraping of real Lisbon zip codes | Google Maps scraping (subways, attractions, universities) | regression analysis | price prediction & visualization | report

Introduction to Programming

Master Program in Data Science
and Advanced Analytics 2018/19

I. Introduction

As international students moving to Lisbon, all of us experienced problems finding a place to stay in Lisbon. Beside websites hosted by private renting companies, such as erasmusu.com, platforms offering short-term accommodation, such as airbnb.com, and marketplace websites like Facebook and Bquarto, one platform stands out: Uniplaces.com. Uniplaces is an international marketplace for apartments, rooms and shared rooms, providing comprehensive accommodation information provided by the landlord. The website is fully hosted in several languages, accepts direct bookings with international credit cards and is thus one of the major destinations for incoming international student looking for a room or apartment in Lisbon.

As an international student, it is difficult to understand the price structure of housing offers in Lisbon. The uniplaces website does not offer any tools that would easily enable a user to evaluate how much an accommodation with certain specifications should cost, considering all available offers. Thus, as a step to support decision making for international student in the uniplaces environment, we created a price prediction tool that predicts the expected monthly price of a housing offer in Lisbon, based on a number of desired housing-characteristics specified by the student.

First, we scraped the uniplaces platform to access housing data for Lisbon as a baseline for our analysis. Then we thoroughly cleaned the scraped data. Following this, we enriched our data with further information regarding the housing location which we scraped from Google Maps. We applied several regression models to our data and compared their performance. We then developed the price prediction based on the best model and created a prototype interactive user-interface.

II. Data

a. Extraction

Scraping offer information from uniplaces.com

We extracted data directly from uniplaces, using web-scraping techniques. First, we had to access the unique offer ID's of each listing in Lisbon in order to scrape the specific characteristics of each offer. Upon inspection of the website, we were not able to find the listing URL's or ID's within the HTML code of the uniplaces main search site. However, when opening the website using Chrome Developer Tools[1] we could observe an entry within the Network > XHR tab that seemed to represent the response to an API request[2] with specific parameters, returning the information we were looking for (see Appendix A). Thus, we requested the response of this API with parameters specified to

[1] Google Chrome Developer Tools documentation. Available: https://developers.google.com/web/tools/chrome-devtools/
[2] Google Chrome: "Cross-Origin XMLHttpRequest". Available: https://developer.chrome.com/apps/xhr

English language and geolocation boundaries that we obtained from the URL when zooming in on an appropriate part of the map on the uniplaces website. Uniplaces showed 127 pages for Lisbon offers. In order to not miss any listings, we looped over 140 pages - since we were going to exclude duplicates later on. From this API request we not only received the unique offer ID's (offer_id), which we wanted to use to find the separate listings, but also the listing's titles and rental types - which we saved, in case we needed them later. With the offer_id's we were now able to access all listings by adding the unique offer_id to the base URL https://www.uniplaces.com/accommodation/lisbon/.

Next, we needed to scrape each unique offer independently in order to receive specifications of the accommodation. However, uniplaces does not hold the listings information within the site's HTML code either, which is why conventional web-scraping libraries that are based on the HTML content of a website, such as Beautiful Soup, failed to work again. However, using the Google Chrome Developer Tool > Network > Other tab, we could observe a 'graphql' document that held all listing information that we wanted to access (see Appendix B).

As we found out, uniplaces operates via the query language GraphQL. GraphQL delivers a smart answer in JSON format based on client specified fields[3]. The GraphQL introspection query[4] as well as the response visible directly within the developer tool served to understand the specific schema, i.e. the field structure of the uniplaces GraphQL service. We specified the fields to be queried for each listing based on the content that is visible to the user on the uniplaces website. This means that a large amount of detailed information for each listing was queried and the resulting dataset contained many columns. We chose this approach in order to start our analysis from a complete dataset, being able to exclude less relevant information if not required, rather than initiating our project on an incomplete data set.

As expected, adjustments had to be made to the received JSON responses in order to alter them to a format we could work with. For example, information on features of the different rooms in one apartment had to be extracted from within a list of dictionaries and assigned to appropriate fields.

Geopy scraping to obtain true listing zip codes

By taking a look at the scraped zip codes, we noticed that not all zip codes were clean, as some contained strings or the code seemed completely invalid. Checking geo-locations (which we assumed to be reliable for all listings) on Google Maps, we noticed that not all listings were truly located in Lisbon. Thus, in order to verify our listings to only be in Lisbon, we decided to first obtain their real zip codes from the geo locations and then match the acquired zip codes with a full list of all Lisbon zip codes in order to exclude all offers that did not match.

[3] GraphQL documentation. Available: https://graphql.org/
[4] Craigbeck: Introspection query for GraphQL. Available: https://gist.github.com/craigbeck/b90915d49fda19d5b2b17ead14dcd6da

In order to get the real zip codes we used the python package Geopy. For each data point we used the reverse method of Geopy to acquire the related zip code. Since Geopy operates using a Google Maps API and we did not have a commercial license to use this specific API, we only had a limited number of requests per day for an IP address, therefore scraped data partially over several days.

After obtaining the offer zip codes, we scraped a complete list of Lisbon zip codes from worldpostalcode.com[5] in order to match them with our dataset. Where Geopy scraping was successful, we used this zip code for comparison. Where Geopy scraping returned NaN, we used the original zip code returned from uniplaces for comparison. By matching each listing's zip code with the complete list of Lisbon zip codes. Finally, we only kept entries in our data set which were located in Lisbon.

Google Maps scraping to enrich our model

In order to create a more comprehensive regression model, we enriched our analysis with further data. This step was done after data cleaning. Since we had already acquired all information related to specific housing offers from uniplaces, we decided to take three further variables connected to a listing's location into consideration: closeness to universities, subway stations and tourist attractions. For the first two, our hypothesis was that housing offers that are closer to subway stations and universities would be more attractive for students, who will naturally have to move through the city and especially to the university campus. The third variable was chosen since we assumed that the adjacency to tourist attractions could provide an estimate for the attractiveness of an offer in terms of a central location.

Therefore, we took strings of the relevant information from Google[6], Wikipedia[7] and Unipage.net[8]. For all three variables it was more efficient to copy-paste from the relevant web pages instead of using web-scraping, since the quantities of information were relatively small. For subway station information we could be sure it would be complete. For universities and tourist attractions the choice was more cumbersome. We selected the largest and most popular universities and tourist attractions. However, since there are several sources of information, especially for tourist attractions, we were aware that this information is subject to interpretation and should be handled carefully.

[5] World Postal Codes: "Lisboa Postcodes".
Available: https://worldpostalcode.com/portugal/lisboa/.
[6] Google Travel: Things to see. Available: https://www.google.ru/travel/things-to-do/see-all?g2lb=4181926%2C4208993%2C4209588%2C4224901%2C4225813%2C4207631%2C4215556%2C4220469&hl=en&gl=pt&un=1&dest_src=tau&dest_mid=%2Fm%2F04llb&sa=X&ved=2ahUKEwjBquzm1Y3fAhXMB8AKHSNUDDgQzO0BKAh6BAgMEBg
[7] Wikipedia: List of Lisbon Metro Stations. Available: https://en.wikipedia.org/wiki/List_of_Lisbon_metro_stations
[8] Unipage: Universities in Lisbon. Available: https://www.unipage.net/en/universities/lisbon

We then obtained the geo-location of all places via a Google Maps API call using the Python package urllib. After calculating the distance of each housing offer to each of the points of interest, we aggregated the information to six new fields: 'subways within 0.5 km', 'universities within 0.5 km', 'attractions within 0.5 km', 'closest subway', 'closest attraction', 'closest university'.

b. Transformation

As the data was scraped entirely from the website without regards to formatting preferences, we expected that the data cleaning needed to be done thoroughly. First, we renamed all columns to workable titles and dropped columns that were of no use, such as those indicating the currency (all values were in Euros) and duplicate columns. Afterwards we removed all the rows with duplicate offer ID's in the dataset to erase possible errors occurred in the scraping process.

We adjusted data formats, such as monetary values and dates. Where we thought necessary, we aggregated variables (e.g. 'cleaning_period' which seemed very specific) to a higher level, or created comparable variables. Examples for the latter would be the total commission and deposit amount as well as a logarithmic price variable. With the logarithmic price we expected to reduce the bias of higher-priced offers in our later analysis. Categorical values (e.g. the contract-, the property- and rental-type, etc.) were encoded in order to facilitate their handling during data exploration. To be able to access the meaning of the encoded variables later, we saved them in dictionaries.

We detected a lot of null values in the variables for certain amenities, which made sense to us, as we assumed landlords were not patient enough to specify all of these features. The null value treatment was one of the most challenging parts of the cleaning process. Since we had to reduce the dimensionality of our data in any case and we did not find the imputing of variables with a very high amount of null values purposeful, we came to the conclusion to drop all columns with more than 30% of null values. Furthermore, most of the values with such high percentages of null values were specific amenities in specific rooms, many of which we believed not to be crucial to our later regression model.

c. Description

The original data was scraped from uniplaces.com on November 23rd 2018 with a total of 103 columns and 2938 rows, where each row stands for a unique listing. During data cleaning we removed 20% of rows and 56% of columns from the original data frame. Next, we added information through Google Maps scraping and neighborhood categorization. After these steps, our data set had a total of 2350 unique data points and 51 columns. Each data point represents a listing on the website and is identified by a unique 'offer_id'. All columns are listed in Figure 1.

51 variables after data cleaning		
commission_rate	number_of_bathrooms	bedroom_window
deposit_amount	number_of_bedrooms	rental_type
contract_type	property_type	title
internet_included	property_verified	zip_code_scraped
offer_id	kitchen_window	cleaning
offer_price	pets_allowed	commission_amount
cancellation_policy	smoking_allowed	landlord_response_rate
max_guests	overnight_guests_allowed	double_bed
minimum_no_night	bedroom_balcony	log_offer_price
creation_date	single_beds	all_expenses_included
landlord_prev_confirmed	double_beds	bathrooms_per_person
neighborhood_id	kitchen_freezer	subways within 0.5 km
neighborhood	kitchen_fridge	universities within 0.5 km
property_id	kitchen_microwave	attractions within 0.5 km
latitude	kitchen_oven	closest subway
longitude	kitchen_stove	closest attraction
accomodation_type	kitchen_washing-machine	closest university

Figure 1: Data set columns after data cleaning

The data set was still very large, but since we could not be sure of the most relevant variables yet, we decided to analyze several aspects of the data set during data exploration in order to narrow our choice of variables down for the regression model. Furthermore, we wanted to leave the option of building a landlord-centered model, which might include different variables than a tenant-centered model. This is because the landlord has much more specific information at hand that could serve as a base for the price prediction, which the tenant would not be able to enter (e.g. closeness to subway stations).

Data Exploration

The goal of the data exploration was to get a reasonable understanding of the housing situation in Lisbon and the drivers of housing prices. Furthermore we needed to detect (and potentially treat) outliers to know about (or mitigate) their possible effect on our prediction model. Therefore, we first asked ourselves which factors had the highest impact on us when we were looking for a room.

Among others, some of our top criteria included the neighborhood and location, the size of the room, the amount of rooms and bathrooms and the pictures. As room size was rarely indicated by landlords on uniplaces, we accepted that we would not be able to include this in our analysis, despite being a major factor.

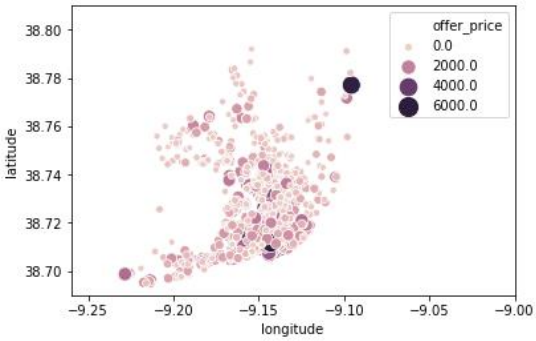


Figure 2: Mapped offer distribution

For the location, the distance to universities, subway stations and attractions were already included in our model, but this did not give us enough understanding about the popularity of certain neighborhoods which, in our opinion, could play a very important role in pricing. Therefore, we mapped all offers with the sizes and colors of the dots according to their price to detect certain areas with higher priced offers (see Figure 2).

As this did not yield interpretable results, we decided to categorize the neighborhoods in accordance with an article on a similar analysis[9]. We grouped all offers by their neighborhood and categorized neighborhoods based on their mean values for price and frequency into four different groups. This categorization (see Figure 3) yielded results close to what we expected, so we decided to use it in our prediction model.

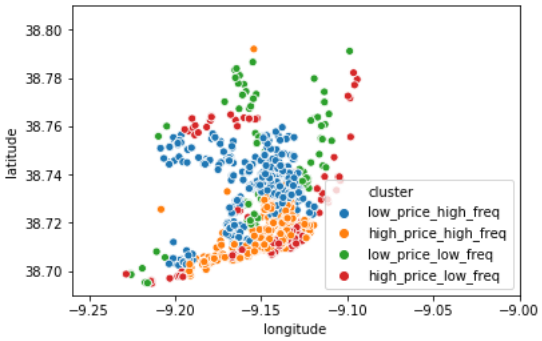


Figure 3: Offer distribution after neighborhood categorization

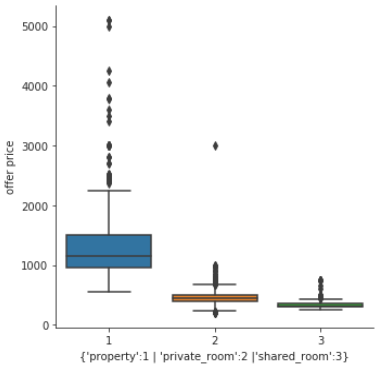


Figure 4: Price distribution for variable 'rental_type'

Additionally we created various boxplots (e.g. see Figure 4) to detect outliers and obtain an understanding of the variable's impact on the price as well as their distribution. Since we wanted our model to work on realistic user input of all sorts, we did not remove extreme offer prices at this point. This choice was evaluated again during the regression analysis.

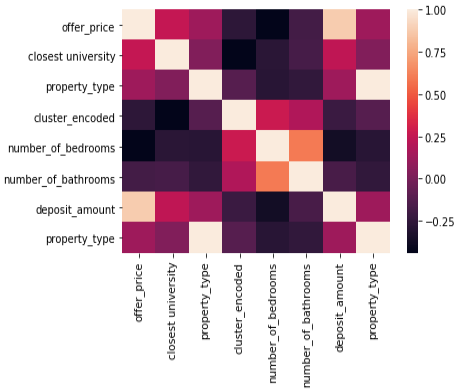


Figure 5: Correlation heat-map on a set of variables

Finally, we inspected correlations using heat-maps (e.g. see Figure 5) and pair-plots (see Appendix C) between different sets of variables and the price/logarithmic price. Here, the logarithmic price already showed some differences to the unadjusted price variable.

[9] S. Li: "Linear regression in Python – Predict the Bay Area's Home Prices." (2017). Data Science Plus. Available: https://datascienceplus.com/linear-regression-in-python-predict-the-bay-areas-home-prices/

d. Analysis

Before applying the prediction models we established an environment to run comparable tests.
Therefore we divided the data set into a training (80%) and a test (20%) data set. Although we ideally would have liked to have a larger test sample, we had to limit it to 20% of the total data, as we feared to compromise on the accuracy of the prediction.

Furthermore, we had to prepare the data for the different prediction models. First, we now had to fill the null values, since the regression models cannot process them. Continuous variables were imputed with the mean over all observations whereas categorical variables where filled using the median. Also, we now created one-hot-encoded variables from our categorical values, since the regression models would not work appropriately with number-encoding.

In order to select appropriate input variables for our model we mainly considered three factors:

- the ability of the end-user of the price prediction tool (i.e. the tenant/student) to determine the variable
- the relationship between the variable and the offer price (correlation)
- the behavior of the model's accuracy when adding or removing the variable

After looking at the correlation of variables with the price, we had a closer look at the most strongly correlated variables which were also possible to be determined by the user. The latter criterion e.g. disqualified closeness to subway stations or attractions, since this information could not possibly be determined by a tenant looking for a flat.

We ran several regressions, adding features (also the ones that did not show a strong correlation) successively to see whether R^2 improved by adding the variable to the model. We achieved the best results using the following variables: 'rental_type', 'bathrooms_per_person', 'number_of_bedrooms', 'double_bed', 'cluster', 'internet_included' and 'bedroom_window'.

The first price prediction model we applied was a simple linear multivariate regression. Knowing that this would most likely not be elaborate enough to a problem with that many dimensions, we chose it as a method that would provide a comparable benchmark for further analysis. After optimizing it, we obtained an R^2 value of 0.78 for this model. One observation was that the logged price was performing much better compared to the nominal price as the higher priced offers would drive the R^2 value, therefore we kept the log price for the further prediction models.

The second method we used is the Random Forest Regressor. This regressor allows for more granularity in the prediction as a predefined number of sub-samples from decision-trees is used to improve accuracy by lowering the

variance. After trying different variables the best R^2 value of the Random Forest Regressor was 0.83. When comparing the real prices to the predicted prices we still realized that the largest variance on the higher priced offers, but none of the attempts with other variables mitigated this problem.

We applied a third method, the Gradient Boosting Algorithm. Boosting methods try to reduce variance and bias by using an ensemble. However, different to the Random Forest where the predictors are made independently, the gradient boosting makes them sequentially. The gradient boosting resulted in an R^2 of 0.81.

The last regression method was the Extreme Gradient Boosting. One of the reasons to use this algorithm was that a lot of winning teams on Kaggle.com had used this regression method successfully on similar prediction problems. This alteration of the Gradient Boosting Algorithm is especially designed for structured classification and regression predictive modeling problems. With the same variables we achieved an R^2 of 0.85.

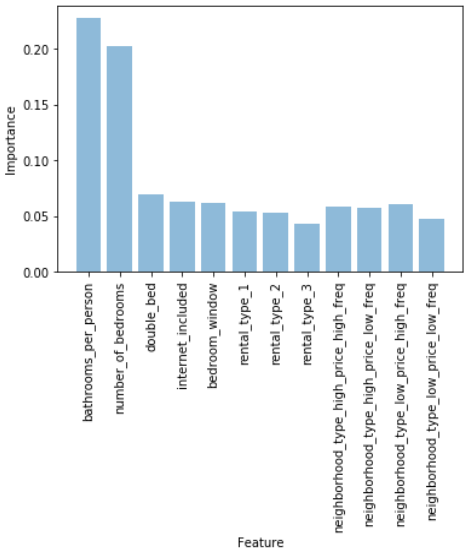


Figure 6: Feature importance in the final model

Since the XG Boost regressor achieved the greatest R^2, we used this model to build our price prediction tool. Feature importance (see Figure 6) was relatively equally distributed, with the most important features being the number of bathrooms per person, the number of bedrooms, the rental type and the neighborhood category. Using Python libraries, we were then able to create an interactive user-interface for the price prediction model (see Figure 7)

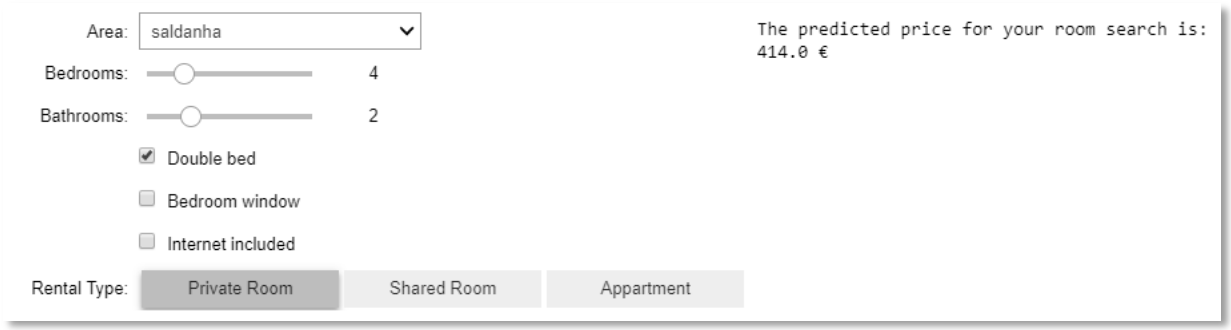


Figure 7: The final interactive user-interface for the price prediction model

III. Results and Discussion

Overall, we were able to combine several data sources to create a regression analysis on Lisbon housing offers with a good R^2 of 0.85 on the test set. Also, when given new user-input, the model seems to give good approximations on offer prices. Some unexpected behavior we observed was that in some cases the price decreases if a double-bed or a bedroom window is requested by the user or the price increases too strongly if certain variables are changed.

This could possibly be due to the relatively small amount of data points, to the lack of continuous variables in the model or the influence of a few high-priced offers on the model. We decided to keep high-priced offers in the data set, since we could not detect any outliers, but rather extreme values that seemed to help to predict very exquisite housing offers and increased R^2 of our models. The lack of continuous variables was due to the fact that most available data on uniplaces are categorical and that the Google Maps scraped data were in the end irrelevant for a tenant-based model. When looking at feature importance, it seems that especially the continuous variables added to the model's accuracy which is why we believe that having more of such variables could improve the model.

Looking at our final model and our process, many data points we acquired from uniplaces as well as from the Google Maps scraping were not used in our final regression model. This is because when collecting data for our model, we were mainly looking for a well-fitted regression. A lot of these variables could, however, not be included in our price prediction model, since they cannot be determined by a user.

IV. Conclusions

The created price prediction tool is able to interactively give a good approximation of a housing rental price in Lisbon based on user input. Some unexpected behavior could be observed and possible measures in order to improve the predictions have been outlined in the previous discussion. Especially the availability of a larger data set should be useful to increase the model's performance. Future work building on this project could include the elaboration of the model to a HTML based website, the building of a landlord-facing tool. Also, in case the uniplaces platform keeps growing, more housing offers are added or information on the website is more complete, the prediction could easily be re-modeled.

Beside the price prediction model itself, we were able to create a scraping-framework for uniplaces that can readily be reused to explore other Data Science questions. Furthermore, we obtained multiple variables that were not yet used in our prediction model but could serve as a base for a landlord-facing price prediction model and other analyses based on uniplaces data - e.g. for other cities that are covered on uniplaces.com or for specific offer types.

V. Bibliography

Brownlee, Jason: "A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning" (2016).
Available:https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/

Craigbeck: Introspection query for GraphQL.
Available: https://gist.github.com/craigbeck/b90915d49fda19d5b2b17ead14dcd6da

Das, Sagarnil: "Predicting Boston House Prices" (2017).
Available: https://www.kaggle.com/sagarnildass/predicting-boston-house-prices

GeoPy Documentation.
Available: https://geopy.readthedocs.io/en/stable/

Google Chrome: Cross-Origin XMLHttpRequest.
Available: https://developer.chrome.com/apps/xhr

Google Chrome Developer Tools documentation.
Available: https://developers.google.com/web/tools/chrome-devtools/

Google Maps Platform documentation.
Available: https://cloud.google.com/maps-platform/

Google Travel: Things to see.
Available: https://www.google.ru/travel/things-to-do/see-all?g2lb=4181926%2C4208993%2C4209588%2C4224901%2C4225813%2C4207631%2C4215556%2C4220469&hl=en&gl=pt&un=1&dest_src=tau&dest_mid=%2Fm%2F04llb&sa=X&ved=2ahUKEwjBquzm1Y3fAhXMB8AKHSNUDDgQzO0BKAh6BAgMEBg

GraphQL documentation.
Available: https://graphql.org/

Li, Susan: "Linear regression in Python - Predict the Bay Area's Home Prices." (2017). *Data Science Plus*.
Available: https://datascienceplus.com/linear-regression-in-python-predict-the-bay-areas-home-prices/

Marcelino, Pedro: "Comprehensive data exploration with Python" (2017).
Available: https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python

Mohun, Philip: "Making Models | Airbnb Price Prediction: Data Analysis" (2018). *Medium.*
Available: https://medium.com/@philmohun/making-models-airbnb-price-prediction-data-analysis-15b9af87c9d8

Pathak, Manish: "Using XGBoost in Python" (2018).
Available: https://www.datacamp.com/community/tutoials/xgboost-in-python

Python API Documentation.
Available: https://xgboost.readthedocs.io/en/latest/python/python_api.html

Sarkar, Tirthajyoti: "A very simple demo of interactive controls on Jupyter notebook" (2017). *Towards Data Science.*
Available:https://towardsdatascience.com/a-very-simple-demo-of-interactive-controls-on-jupyter-notebook-4429cf46aabd

Scikit-learn: gradient boost regression documentation.
Available:https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html

Scikit-learn: linear regression documentation.
Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

Scikit-learn: random forest regression documentation.
Available:https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

Stackoverflow.
Available: https://stackoverflow.com/

Unipage: Universities in Lisbon.
Available: https://www.unipage.net/en/universities/lisbon

Uniplaces Housing Offers in Lisbon.
Available: https://www.uniplaces.com/en/accommodation/lisbon

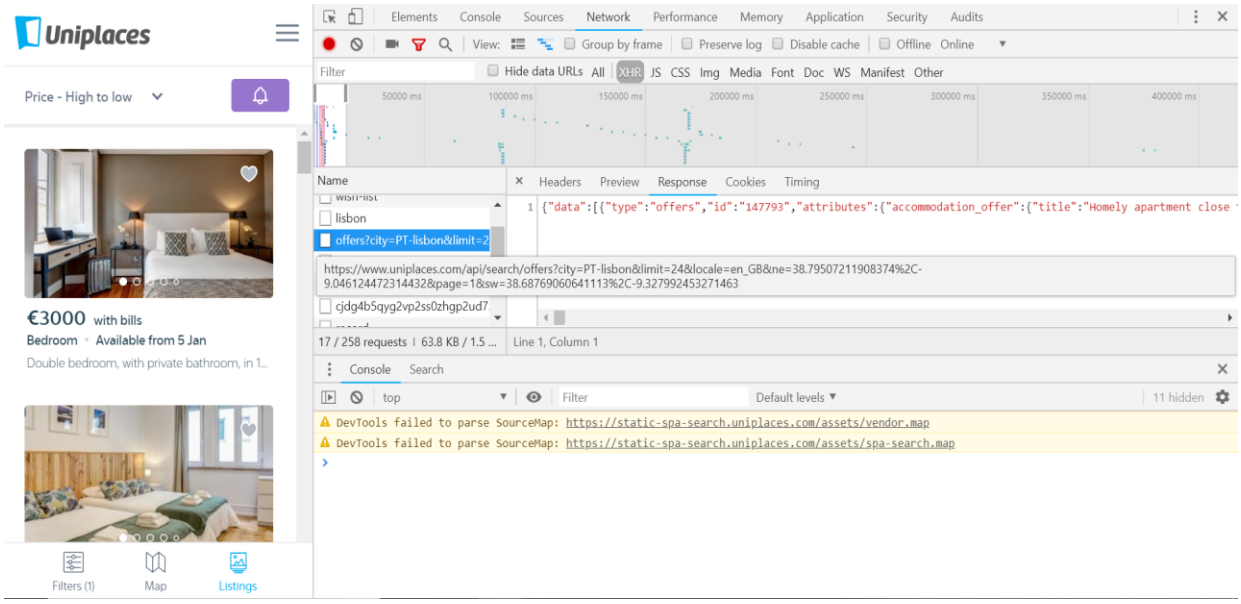
Urllib documentation.
Available: https://docs.python.org/3/library/urllib.html

Wikipedia: “List of Lisbon Metro Stations”.
Available: https://en.wikipedia.org/wiki/List_of_Lisbon_metro_stations

World Postal Codes: “Lisboa Postcodes".
Available: https://worldpostalcode.com/portugal/lisboa/

VI. Appendix

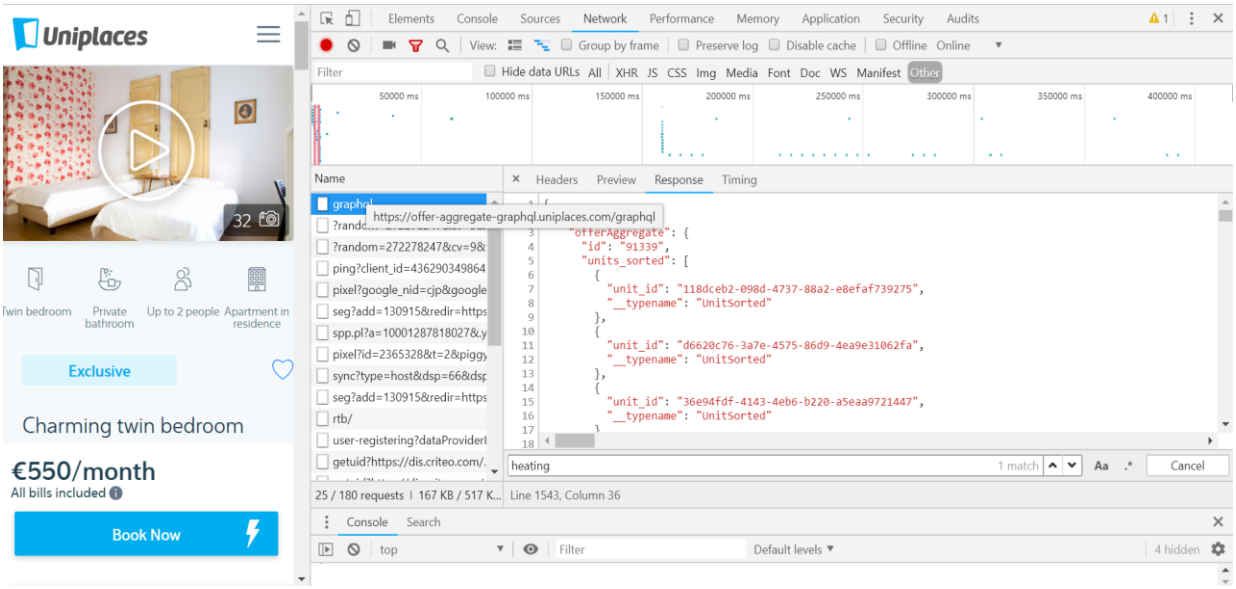
Appendix A: Extract from the developer tool view to identify the uniplaces API address



Introduction to Programming

Master Program in Data Science
and Advanced Analytics 2018/19

Appendix B: Extract from the developer tool view to identify the GraphQL query



Appendix C: Pair-plot of a subset of variables to identify correlations