R for Data Science Project 1

Data Cleaning and Exploratory Analysis: Airbnb Data from Rio de Janeiro, Brazil

Aims

The aim of this project is to use supervised learning to develop and train a model to predict the review score of an Airbnb listing (*Airbnb: Holiday Rentals, Cabins, Beach Houses, Unique Homes & Experiences*, no date) based on attributes of that listing. Depending on the success of the model, an alternative aim is to use unsupervised learning to identify clusters of Airbnb listings based on those same attributes.

Having an Airbnb listing can be extremely profitable. Pofahl (2017) cites a study by the Economic Research Institute Foundation which shows that Airbnb rentals and related tourism added 2.5 billion BRL to Brazil's GDP in 2016. A model that can predict review scores, or identify the shared attributes of successful listings, could be very useful for current and future hosts.

In order to achieve this aim, a data set consisting of Airbnb listings in Rio de Janeiro, Brazil, has been gathered and cleaned. The data set has a large number of listings as well as a significant number of attributes (features) for each listing. Many of the attributes, intuitively, seem likely to have a strong bearing on review score, such as price, location, host response rate, etc. Review score (the target variable) is included as an attribute, which is essential for supervised learning. For these reasons, the data set is considered appropriate for the proposed modelling task.

Data Gathering

The data set was taken from http://insideairbnb.com/. According to the site, "The data utilizes public information compiled from the Airbnb web-site" and "No "private" information is being used." (*Inside Airbnb. Adding data to the debate.*, no date) Furthermore, the data is made available under a Creative Commons CC0 1.0 Universal (CC0 1.0) "Public Domain Dedication" (*Creative Commons — CC0 1.0 Universal*, no date), and therefore is freely available for use.

Given that the data is available from the Airbnb website, it would have been possible to obtain the data directly from source. However, this would have involved significantly more work in scraping the website, aggregating and filtering the data. Furthermore, the Inside Airbnb website hosts the data in (zipped) CSV format, making it ideal for use in this project.

The zipped file was downloaded and unzipped manually to the local environment before being read into RStudio as a data frame object.

Data Checking, Cleaning and Manipulation

The R Script begins by installing and loading the necessary packages and libraries. Loading the tidyverse package provides access to the dplyr and ggplot2 packages, both of which are used in the data cleaning and analysis. gridExtra enables the plotting of multiple graphs in a grid in the same window. Finally, the 'here' library enables the loading and saving of data without any of the problems associated with setwd() (Bryan, 2017).

After reading in the data set, a number of unnecessary columns are dropped. The code first displays the column names in to aid identification of the column indexes. The following columns are unlikely to be useful for the analysis/modelling, and so are dropped:

- listing_url
- scrape_id
- last_scraped
- picture_url
- host_id
- host_url
- host_name
- host_since
- host_location
- host_about
- host_thumbnail_url
- host_picture_url
- host_neighbourhood
- host_listings_count
- host_total_listings_count
- minimum_nights_avg_ntm
- maximum_nights_avg_ntm
- calendar_updated
- has_availability

- availability_30
- availability_60
- availability_90
- availability_365
- calendar_last_scraped
- number_of_reviews_ltm
- number_of_reviews_l30d
- first_review
- last_review
- license
- calculated_host_listings_count
- calculated_host_listings_count_entire_homes
- calculated_host_listings_count_private_rooms
- calculated_host_listings_count_shared_rooms
- reviews_per_month

The following columns are dopped because the information is available in other columns, often in a more appropriate format:

- name
- description
- neighborhood_overview
- neighbourhood
- neighbourhood_group_cleansed
- property_type
- bathrooms
- minimum_minimum_nights
- maximum_minimum_nights
- minimum_maximum_nights
- maximum_maximum_nights

This leaves 29 variables (columns) and 26,628 observations (rows).

The code then performs an initial check for missing values. The main concern at this point is the review_scores_rating column. This variable is essential to the final model, and so all rows with a missing value are dropped, leaving 16,155 observations.

Two of the columns, host_verifications and amenities, contain multiple entries per cell. For example, host_verifications contains different methods used by hosts to verify a booking. Data like this can be made useful for machine learning by employing one-hot encoding. This involves creating a separate column for each category with a 1 indicating that the category applies, and a 0 indicating it does not.

The host_verification and amenities character strings are cleaned and separated, and unique values are collected in two separate vectors. While host_verifications has 16 distinct values, amenities has 899. This is too many to feasibly turn into distinct columns. A quick examination of the first 25 amenities shows that one option would be to create higher level categories. For example, “coffee maker”, “oven” and “microwave” could be combined into the single category “cooking equipment”. However, this would be very time-consuming and so is considered beyond this project. Consequently, the amenities column is dropped (making use of Tidyverse’s pipe operator; the pipe is used often throughout this script to avoid having to create intermediary data frames).

Returning to host_verifications, for each row in the data frame the grepl method is used to return a logical value based on whether a verification method is present in that particular row. These logical values are converted to 1s or 0s and placed in a new column, using the verification method as the column name. The original host_verifications column is then dropped.

The earlier str() call revealed that a number of columns are not in the correct format for the analysis. Host_response_time contains just four distinct options and so is converted from character to ordered factors. Host_acceptance_rate, host_response_rate, and price should be converted to numeric values. To do so, the % and $ symbols must be removed. A function is created to do this to avoid repetitive code. The host_acceptance_rate and host_response_rate values are divided by 100 in order to get values between 0 and 1.

The next column to handle is the bathroom column. The sub method corrects any spelling/formatting differences, reducing the number of distinct entries from 44 to 42. The column is then converted to factor format. Although there is a natural ordering between the factors based on numbers (i.e. 1 bath < 1.5 baths), the ordering between shared and private baths is more ambiguous, and so the factors are left unordered.

Four of the columns in the data frame have entries “t” or “f” to represent True or False. Similarly to the one-hot encoding performed earlier, these must be converted to 1 or 0 for machine learning. While it would be possible to do this with a for loop, to save on computational expense, R’s lapply method is used instead.

Finally, the neighbourhood_cleansed and room_type columns are converted to factors, again using lapply. Another call to str() shows that all of the columns are now in the appropriate format.

The script now moves on to address the remaining missing values. While this could have been done before cleaning the columns, some of the operations introduced NAs by coercion and so it is better to deal with all missing values at this stage.

There is no logical way to impute any of the missing values. Using the mean to fill missing numbers of bedrooms or bathrooms, for example, would not make sense. Intuitively, price is likely to be one of the more important factors affecting review scores. Since only 915 observations are missing prices (approximately 5% of the data), these observations are dropped.

For the other features, recall that some columns of the original data set were dropped at the beginning of the script due to repetition of information. It may be the case that these columns can help to fix some missing values. A temporary data frame (temp_df) is created, consisting of the id, name, description and host_picture_url columns. This data frame is then joined with the main data frame. A left join is used with the main data frame as the left data frame. This results in the temporary data frame dropping the rows that were already dropped from the main data frame (such as where ratings were missing).

From this temporary data frame, a smaller data frame is created (profile_pic) in order to look at the missing values in the host_has_profile_pic column. The justification is that, if a host has a URL for a profile picture, then they must have a profile picture. However, no URLS are found for the missing values. It is therefore reasonable to assume that these hosts do not have a profile picture, and so the missing values can be replaced with zeros (false). The profile_pic data frame is no longer needed so it is removed from the RStudio environment to save on memory.

A similar process is carried out for bathrooms_text, bedrooms, and beds. The grepl function is used to search through the name and description columns, trying to match either the English or Portuguese words for bath(room) and bed(room). For bathrooms, four matches are found in the description column. The logical vector created by grepl is used to print the descriptions to the console. The number of bathrooms and whether they are private or shared is somewhat ambiguous in the first match. However, the second and third matches both state one bathroom, while the fourth has one shared bathroom. The logical vector is used again, this time to recover the id of these observations, before replacing the missing value in the original data frame with the correct values.

For bedrooms, 58 observations are found containing one of "bed", "quarto", "cama", or "studio" in the name column. Of those, six have one bedroom, 50 are studios. These are also corrected in the main data frame (ignoring the two matches that are unclear). Searching the description column yields 708 matches. Manually extracting information from 705 descriptions is considered beyond the scope of this project and so these matches are ignored. Lastly, 17 missing values in the beds column are corrected using the same approach.

Although some missing values remain in the data set, there is no logical way of imputing them. For example, using the mean number of bedrooms to fill in the missing bedroom values would be inappropriate and inaccurate. However, to drop the observations could lead to the loss of valuable information in other columns. If required, these rows could be dropped when it comes to the modelling stage. However, for now these observations are retained, complete with missing values (as NAs). This leaves 15,240 observations

Exploratory Analysis

The exploratory analysis begins by looking at a summary of the data frame. Immediately two things stand out: the maximum values for minimum_nights (1,000) and maximum_nights (9,999,999,999). A box plot and histogram show that the bulk of observations have a relatively low value for minimum_nights, but a small number have very high values. A common measure of outliers is 1.5 times the interquartile range above the third quartile. However, using this measurement here would classify 2,140 values as outliers. Rather than dropping all of these observations, the decision is made to drop observations with a minimum night stay greater than 14 days (304 observations in total). This is justified on the basis that a minimum stay greater than two weeks is more of a long-term rental than a holiday let. Such rentals are likely to be rated on different factors, so it seems appropriate to remove them from the data set.

For maximum nights, the maximum value is clearly an error in the data. When examined, it seems that a large number of observations have a maximum of 1,125 nights, with just five observations displaying a higher value. This suggests that 1,125 may be a default/cut-off value for Airbnb. The five values are therefore changed to 1,125. The histograms for minimum and maximum nights now appear more reasonable, albeit with an uneven distribution for maximum nights.

Examining the distribution of review scores shows a heavy skew to the left. After zooming in on the higher scores and setting the bin width to one, it is revealed that approximately 6,000 of the observations received the highest possible rating. This is a large proportion of the data, which could have a negative impact on the predictive capabilities of the final model, though it does still leave a relatively large number of observations with a different score.

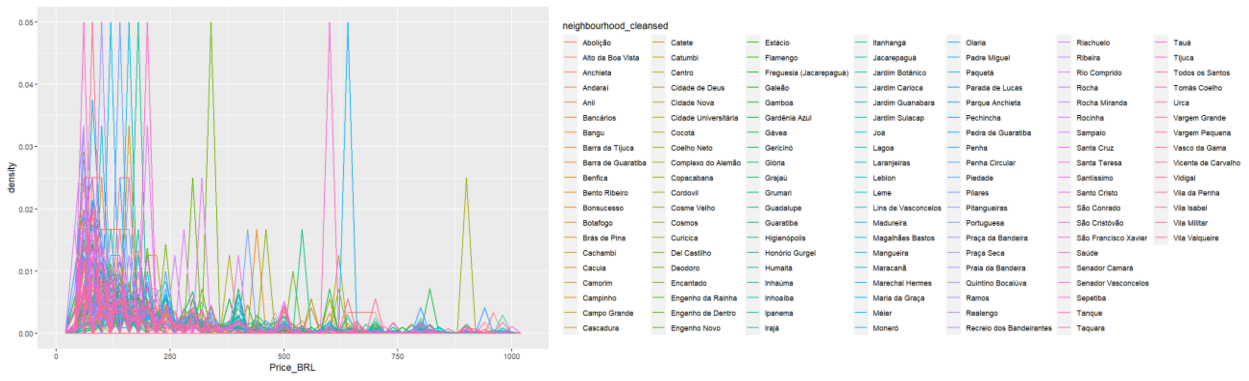
The grid.arrange function is then used to easily display the distribution of the specific factor review scores (accuracy, cleanliness, check-in, communication, location and value). Again, the distributions all skew left.

The price variable, in contrast, skews right. A scatter plot of price against rating shows a vague upward trend in the data. In particular, there are few expensive properties with scores lower than 80. However, there is a large number of low-priced properties with a high score, so it is not a strong correlation.

Plotting neighbourhood against rating does not reveal any strong trends. Rather than continuing to plot all combinations in this way, a correlation table of the numeric variables is created. From this, a subset of the variables is selected (rating and price) to determine if there is a strong correlation between either of these variables and any of the other variables in the table.

When sorted by correlation with rating, it is unsurprising to see that the specific factor review scores have the highest correlation with overall rating. Beyond that, the highest correlation is 0.20 – the correlation between rating and whether or not the host is a superhost. When sorted by price, the highest correlations are with attributes relating to size (bedrooms, accommodates and beds). A pairwise plot is used to better visualise these correlations.

The final visualisation is a density plot of the price by neighbourhood. Due to the number of different neighbourhoods, the legend completely covers the plot if viewed in RStudio. Therefore, the ggsave function is used to save the image to disk.



While this does indicate the existence of three neighbourhoods with almost exclusively high-priced listings, it is difficult to identify them due to the similar colours used in the legend. Instead, a summary table is created showing the average price and number of listings per neighbourhood. The high-priced listings appear to be one-offs.

Conclusion

The main conclusion from the exploratory analysis is that there does not appear to be any strong correlations between rating and any other variable. While this is disappointing it is not necessarily fatal. The strength of machine learning often lies in its ability to make predictions from variables which independently are weak, but when combined possess significant predictive power.

While the data cleaning did involve dropping just under half of the total observations, the final data frame still contains 13,768 rows. This should be sufficient for the modelling task. As suggested above, future work could reduce the number of observations dropped by examining the name and description columns in more detail in order to fill missing values. Furthermore, time spent categorising the amenities available could lead to a more powerful data set for prediction purposes.

Bibliography

Airbnb: Holiday Rentals, Cabins, Beach Houses, Unique Homes & Experiences (no date) *Airbnb*. Available at: https://www.airbnb.co.uk (Accessed: 25 June 2021).

Bryan, J. (2017) *Project-oriented workflow*, *Tidyverse*. Available at: https://www.tidyverse.org/blog/2017/12/workflow-vs-script/ (Accessed: 26 June 2021).

Creative Commons — CC0 1.0 Universal (no date). Available at: https://creativecommons.org/publicdomain/zero/1.0/ (Accessed: 25 June 2021).

Inside Airbnb. Adding data to the debate. (no date) *Inside Airbnb*. Available at: http://insideairbnb.com (Accessed: 25 June 2021).

Pofahl, A. (2017) *How Airbnb added BRL 2,5 billion to Brazil's GDP*, *LABS English*. Available at: https://labsnews.com/en/articles/business/how-airbnb-added-brl-25-billion-to-brazils-gdp/ (Accessed: 26 June 2021).

'R Draw Multiple ggplot2 Plots Side-by-Side (Example) | Plot on One Page' (no date) *Statistics Globe*. Available at: https://statisticsglobe.com/draw-multiple-ggplot-plots-side-by-side/ (Accessed: 26 June 2021).

Trader, T. R. (2018) 'R Code – Best practices | R-bloggers', 1 September. Available at: https://www.r-bloggers.com/2018/09/r-code-best-practices/ (Accessed: 26 June 2021).