Project Report

David García Morillo & Nils van Es Ostos

Spring 2024

Project description

We have chosen the ***Inside Airbnb*** dataset [9], which includes rental data for Airbnb apartments from cities all over the world. You can find the relevant code for this project in https://github.com/Nilsvanesostos/ML-Project/tree/main

Motivation & Previous Work

We are currently in the middle of a housing crisis [1], with the prices skyrocketed, specially in some areas. One of the believed reasons behind this, is the proliferation of Housing Marketplaces such as Airbnb [2]. This vacational housing market has been thoroughly studied in works such as [4] and [3], and with our project we plan to find distinct patterns over the data, such as studying the effect of the location/neighbourhood in the price, try to agglomerate lessors based on different features, predicting the price, etc.

About the dataset

The 4 cities with more accommodations in Spain according to *AirDNA*, a vacation rental data platform, are: Barcelona, Madrid, Málaga and Sevilla [5]. For this reason, datasets from these 4 cities have been chosen [9]. These datasets contain detailed information about the specifications of every apartment (host name, name of the accomodation, type of room, price, location, etc), calendar availability, and reviews.

Exploratory Data Analysis

Note: As the code is available in the repository attached, we will not go line by line stating every operation we did, but rather focus on the most important ones.

This dataset is composed of

- `listings.csv.gz`: Detailed Listings data
- `calendar.csv.gz`: Detailed Calendar Data
- `reviews.csv.gz`: Detailed Review Data
- `listings.csv`: Summary information and metrics for listings (good for visualisations)
- `reviews.csv`: Summary Review data and Listing ID (to facilitate time-based analytics and visualisations linked to a listing)
- `neighbourhoods.csv`: Neighbourhood list for geo filter. Sourced from city or open source GIS files
- `neighbourhoods.geojson`: GeoJSON file of neighbourhoods of the city

We will be focusing in `listings.csv.gz`, since it contains almost all of the useful information needed to predict the price accordingly. We will also use the `listings.csv` to fill some gaps from the other file, such as the neighbourhood. And finally, the `reviews.csv` and `reviews.geojson` to obtain the polygons for visualization purposes.

Let's start with `listings.csv.gz`, as with any dataset, we explore the different columns. In our case, out of the 75 columns available in [9], we selected the following 30:

- `host_response_time`: The average time it takes for the host to respond to inquiries.
- `host_response_rate`: The percentage of messages the host responds to.
- `host_acceptance_rate`: The percentage of booking requests that the host accepts.
- `host_is_superhost`: Whether the host is a superhost (1 for Yes, 0 for No).
- `host_neighbourhood`: The neighborhood where the host's listings are located.
- `host_listings_count`: The number of listings the host has in their profile.
- `host_total_listings_count`: The total number of listings the host manages.
- `host_verifications`: The types of verifications the host has completed (e.g., email, phone).

- `host_has_profile_pic`: Whether the host has a profile picture (1 for Yes, 0 for No).
- `host_identity_verified`: Whether the host's identity is verified (1 for Yes, 0 for No).
- `neighbourhood`: The neighborhood where the listing is located.
- `bathrooms`: The number of bathrooms in the listing.
- `bathrooms_text`: Text description of the number and type of bathrooms.
- `latitude`: The latitude of the listing's location.
- `longitude`: The longitude of the listing's location.
- `room_type`: The type of room being offered (e.g., entire home/apt, private room).
- `number_of_reviews`: The total number of reviews the listing has received.
- `accommodates`: The number of guests the listing can accommodate.
- `beds`: The number of beds available in the listing.
- `price`: The price per night for the listing.
- `has_availability`: Whether the listing has availability (1 for Yes, 0 for No).
- `review_scores_rating`: The overall rating score for the listing.
- `review_scores_accuracy`: The accuracy rating score for the listing.
- `review_scores_cleanliness`: The cleanliness rating score for the listing.
- `review_scores_checkin`: The check-in process rating score for the listing.
- `review_scores_communication`: The communication rating score for the listing.
- `review_scores_location`: The location rating score for the listing.
- `review_scores_value`: The value rating score for the listing.
- `license`: The license number for the listing, if applicable.
- `reviews_per_month`: The average number of reviews the listing receives per month.

Once we saw the different `null` values we had in every column to know what columns to drop/filter, we performed the following transformations

1. Filter the DataFrame:
    - Select rows where `number_of_reviews` is greater than 5.
    - Ensure `host_response_time` is not NaN.
    - Ensure `host_verifications` is not NaN.
2. Drop the `bathrooms` column.
3. Create the `bathrooms` column again, but now from `bathrooms_text` column:
    - Extract numeric values from `bathrooms_text`.
    - Replace NaN values with 0 and convert to integer.
4. Drop the `bathrooms_text` column.
5. Convert `price` to float:
    - Remove dollar signs and commas.
    - Replace NaN values with 0 and convert to float.
6. Replace NaN values in the `beds` column with 0.
7. Create a `dist` column for the distance to a place that we consider to be the center of the city, in the case of Madrid, Plaza del Sol:
    - Use the `haversine_distance` function with reference coordinates of the key-point.
8. Convert `host_response_rate` and `host_acceptance_rate` to integers:
    - Extract numeric values and replace NaN with 0.
9. Transform `host_verifications` into individual boolean columns:
    - `host_has_email`: Whether the host has email verification.
    - `host_has_phone`: Whether the host has phone verification.
    - `host_has_work_email`: Whether the host has work email verification.
10. Drop the `host_verifications` column.
11. Convert the following columns to boolean:
    - `host_has_profile_pic`: Whether the host has a profile picture.
    - `host_identity_verified`: Whether the host's identity is verified.
    - `host_is_superhost`: Whether the host is a superhost.
    - `has_availability`: Whether the listing has availability.

12. Create a `has_license` column to indicate the presence of a license:
    - Check if the `license` column is not NaN.
13. Drop the `license` column.
14. Enrich data with the neighbourhood from the other *listings* file
15. Perform label encoding in the following columns, mapping their values to an integer equivalent, respecting their inherent hierarchy:
    - `host_response_time`
    - `room_type_order`
16. Calculate the pairwise correlation between all the columns, to find what columns are redundant. This is important for the performance of our predictors since it will reduce the dimensionality of the data without removing insights from it. Using this method we dropped the following columns:
    - host_total_listings_count
    - review_scores_value
    - review_scores_accuracy
    - review_scores_communication
    - beds
    - review_scores_checkin
17. Use the `One Hot Encoding` technique to transform the rest of the categorical variables into columns with 1s and 0s.
18. We tried using *PCA (Principal Component Analysis)* to reduce the data dimensions even more, but it made the final bias to be way higher, so we decided to not use it.

Now we will focus on the most important plots we did.

We begin by checking the distribution of the `price` column, both as a whole and segregated by city, as you can see in figure 1. Note, the data used in the figures has been preprocessed to remove outliers for the sake of a better visualization.

As a conclusion from the figure, we can see that due to the bigger demand that Madrid has, since it's the biggest of the 4 cities, the prices are more reduced, at the same time that it has more outliers, in other words, more luxury places.

It would be wrong to think that just because the median of Madrid is lower than in the other 3 cities, it makes Madrid cheaper, since the most probable explanation is that

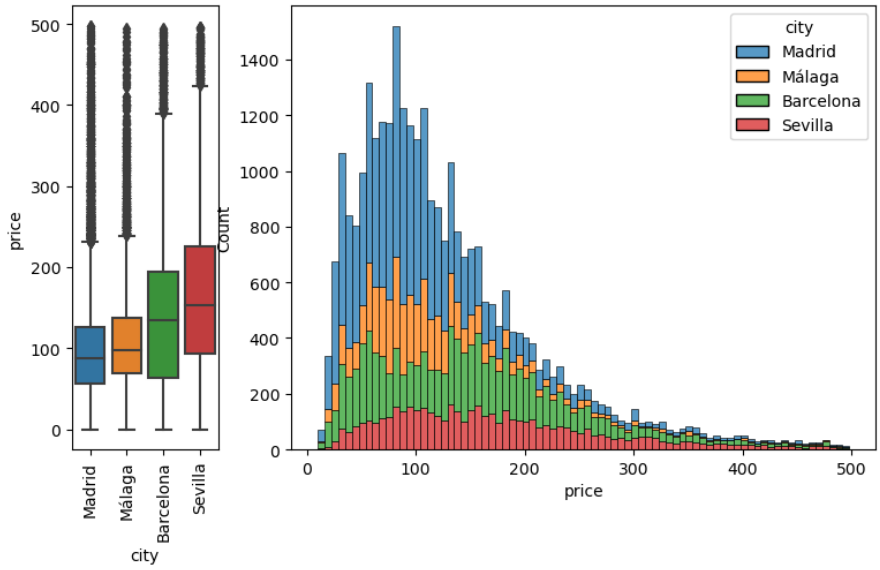

Figure 1: Explore price

there are many more AirBnbs further away from the center, while in the other smaller cities, most of the AirBnbs are closer to the center. So while it may be easier to find cheaper accommodation in Madrid, you'll have to stay further away from the center.

We are gonna test this hypothesis by plotting the same graph filtering only the places that are in the center, as seen in Figure 2

You can now see that is Barcelona the one with more cheap places overall, this said, we still see Madrid as having a low price compared to Seville, probably due to the higher supply.

We also visualized the correlation between columns as it can be seen in 3.

We can clearly see the price is correlated with the type of room (private room, apartment, etc.), also with the number of beds and with the distance to the center, these are all features that semantically also make sense to have relation to the price. Also that the different scores of the ratings are correlated between them, and that as one could expect, the score of the location is correlated to the distance to the center.

Then we started visualizing the average price and number of AirBnb per neighbourhood, first as a bar plot, like in 4 and 5, and then we used the information from `neighbours.geojson` to visualize it interactively using a cloropleth as it can be seen in 6 and 7.

Note that we do these plots on just one city, for obvious reasons

Little can be added to these plots as they are quite self-explanatory, most of the

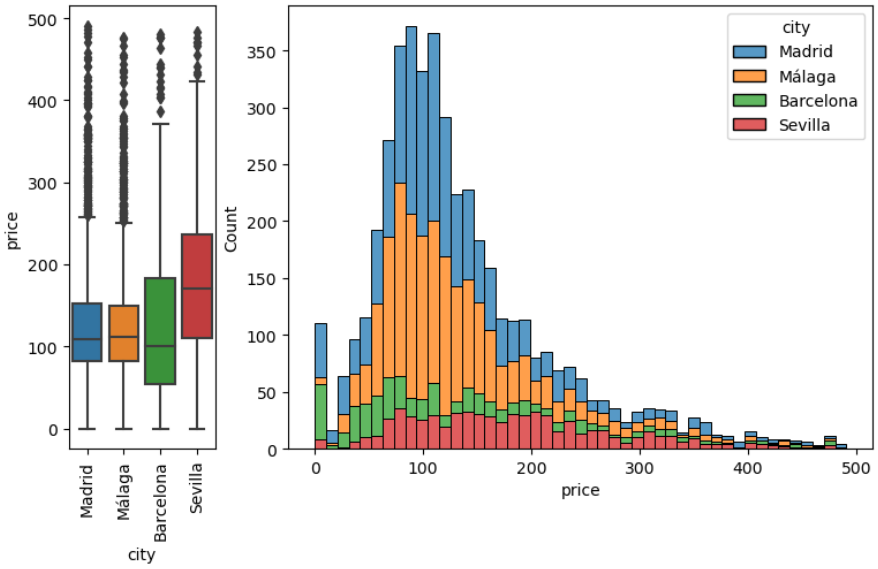

Figure 2: Distribution of price in places within the center of the city

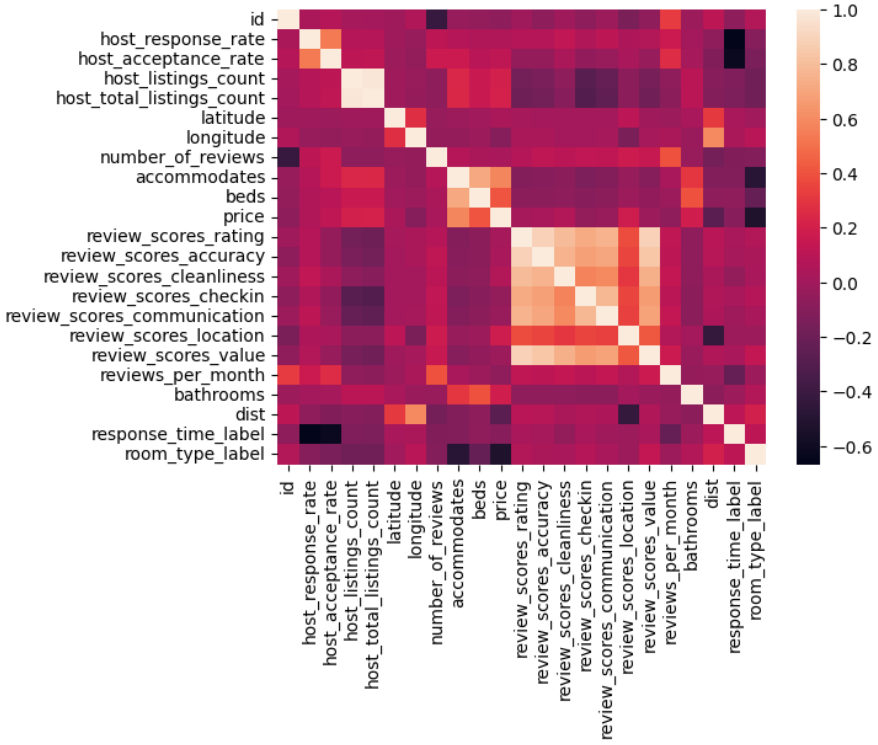

Figure 3: Correlation of columns

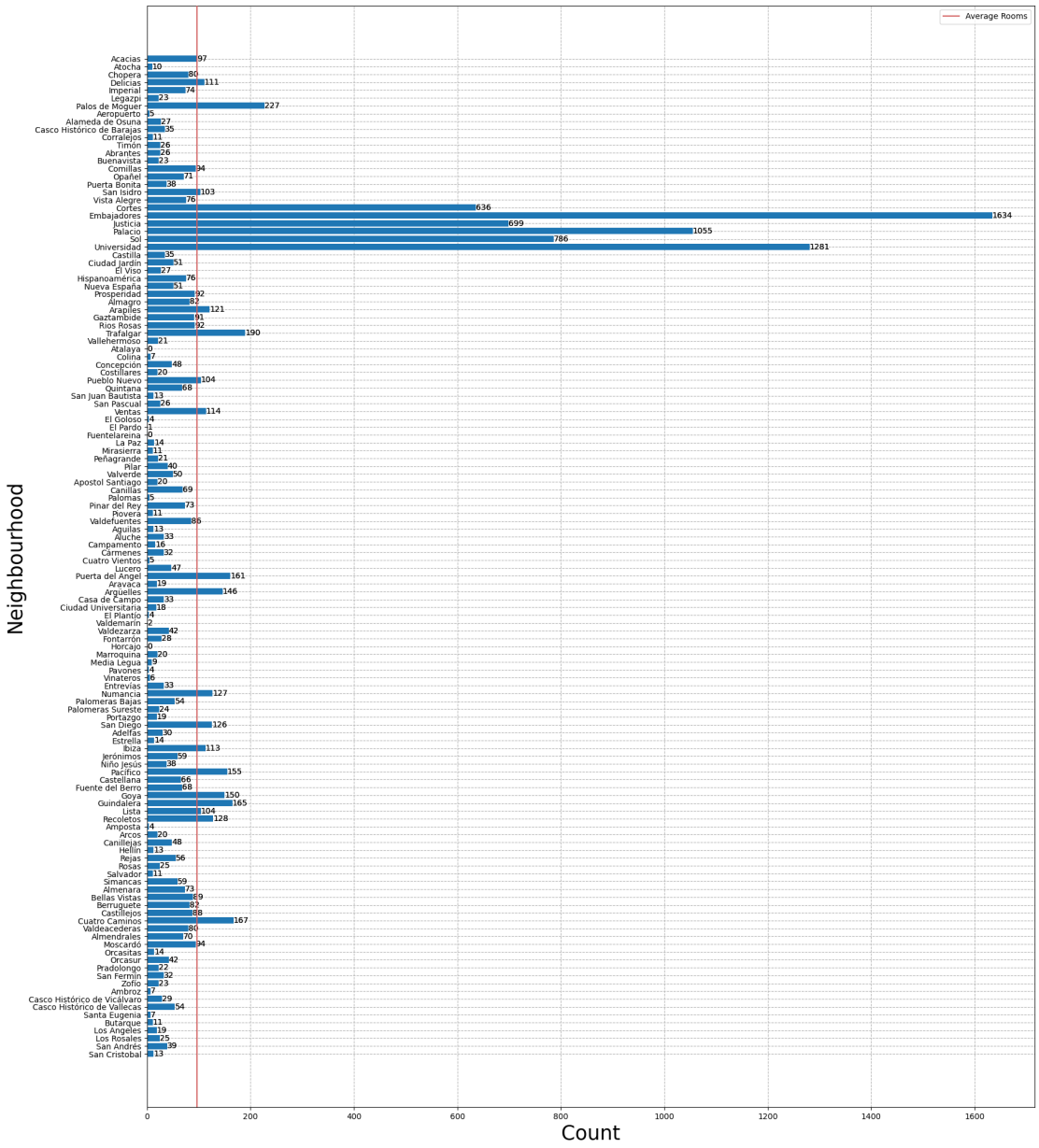

Figure 4: Count of places by neighbourhood

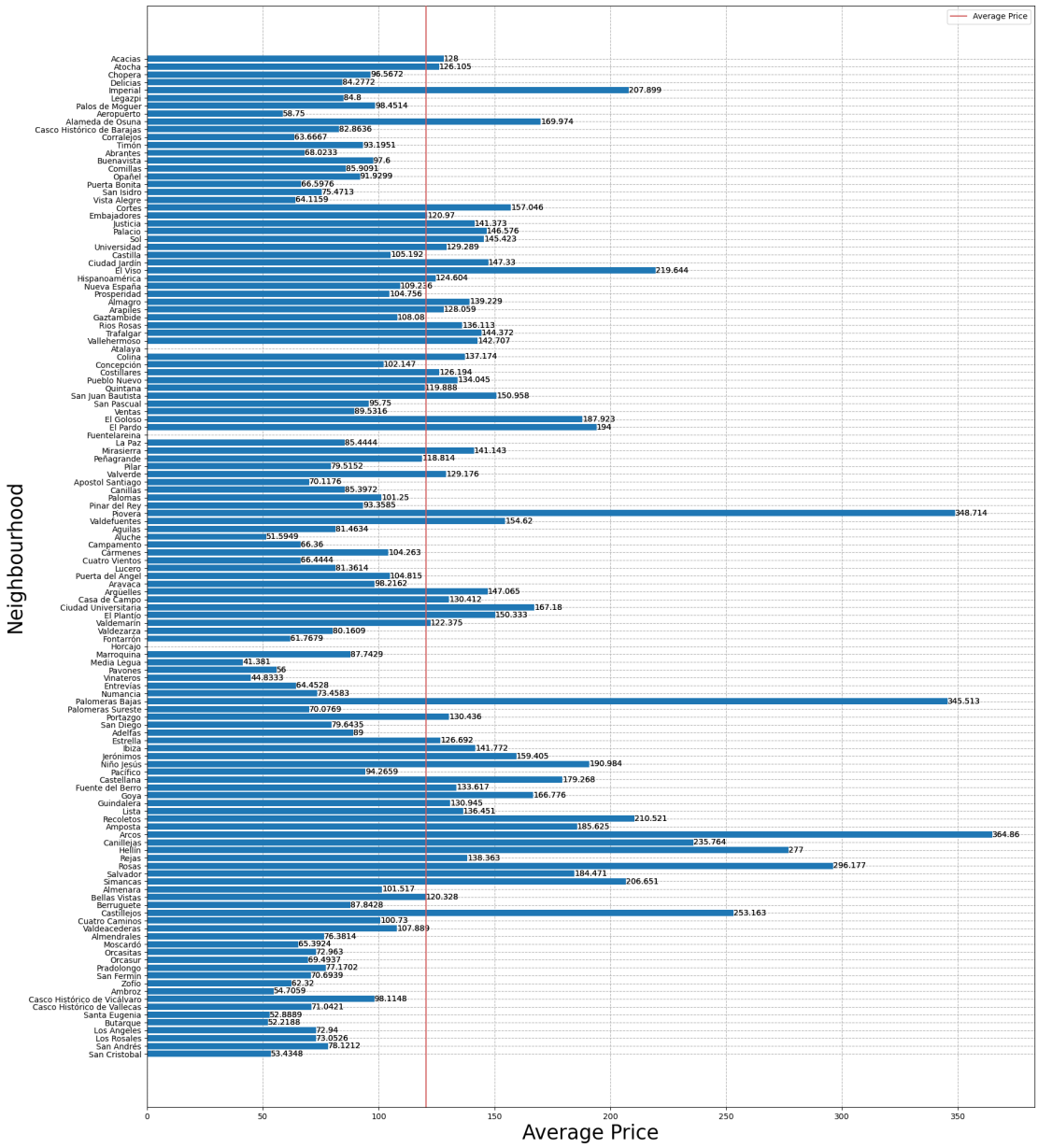

Figure 5: Average price by neighbourhood

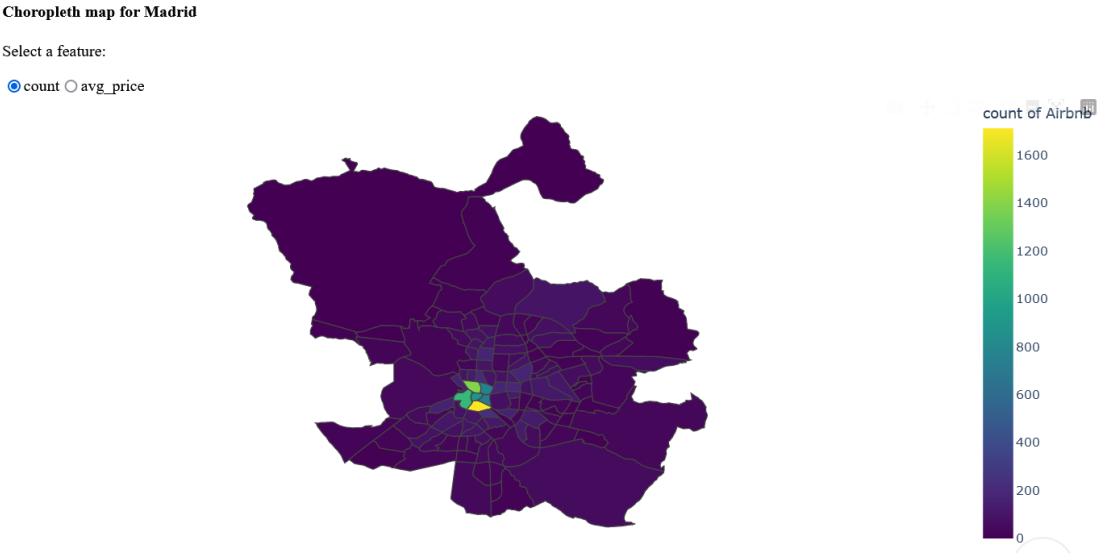

Figure 6: Count of places by neighbourhood

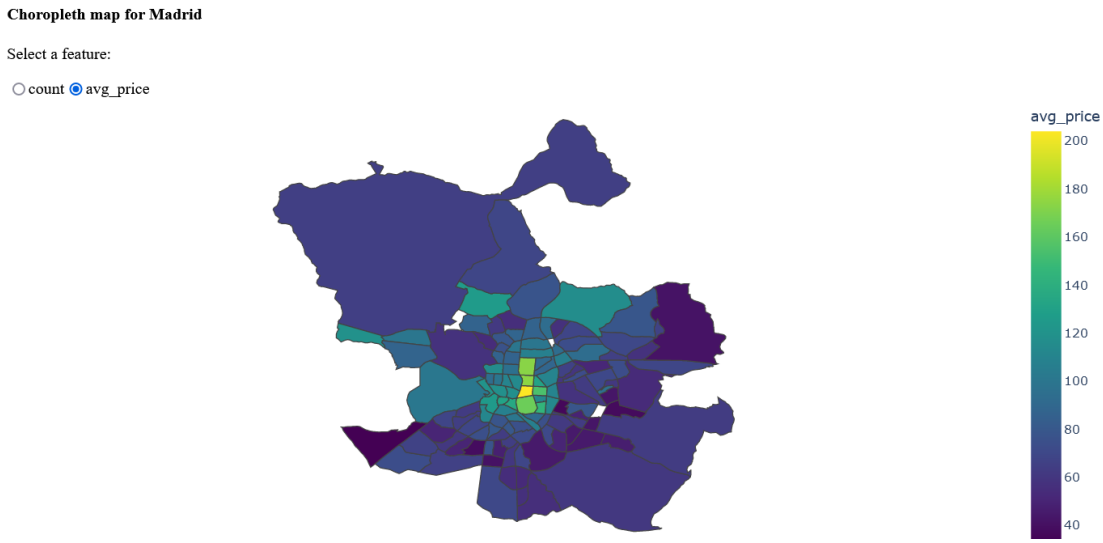

Figure 7: Average price by neighbourhood

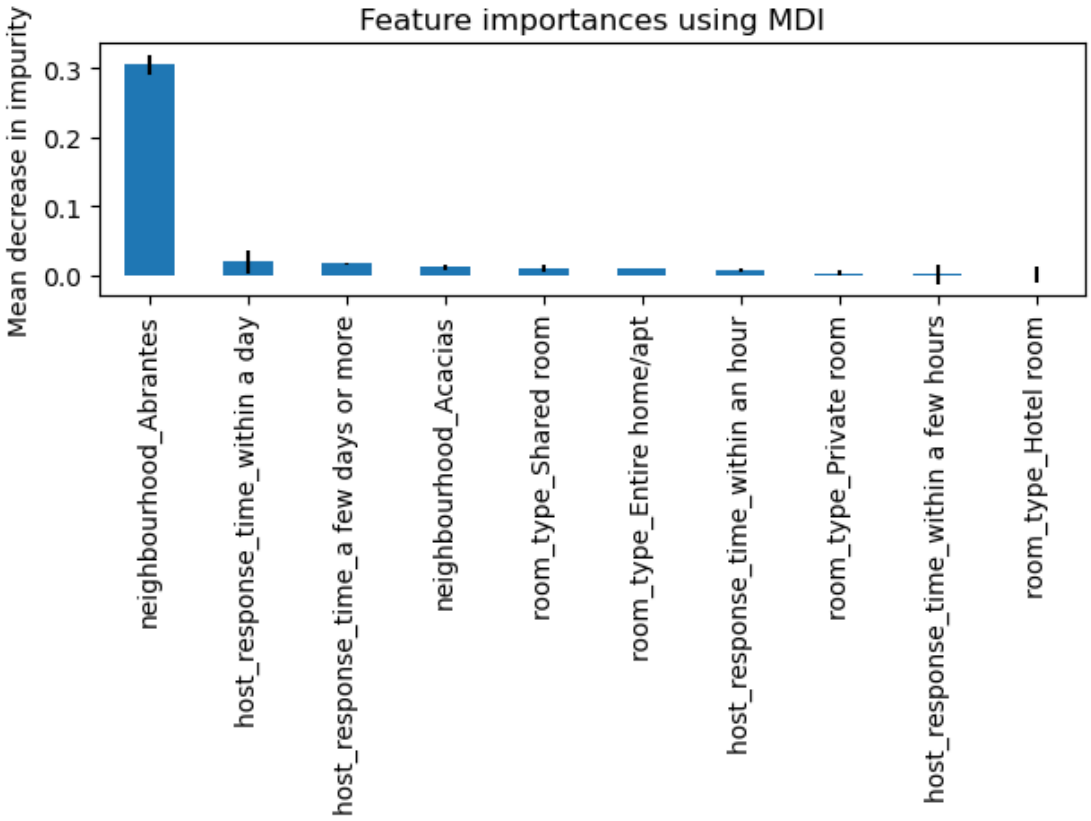

Figure 8: Feature importance

places are in the center, and the prices are also much more expensive there.

By training a *Random Forest Regressor*, we can also calculate the feature importance for that model. We can visualize the 10 most important features according to this metric in Figure 8

The reason why being in the *Abrantes* neighbourhood has so much importance is because it has one of the lowest standard deviation, which means it's very stable in price and it becomes a good feature to split the trees in. Also as we saw before, the kind of room place (apartment, room, etc.) matters a lot. Finally, how often does the host reply as well is an important feature according to this metric.

We didn't make use of clustering as we didn't see how to apply it in a meaningful way.

Selection of models & hyperparameters

As this is tabular data we went for classical ML methods instead of Neural Networks [6].

We wanted to explore methods with different levels of complexity, linear and non-linear, tree-based and non-tree based, so we ended up testing the following methods:

- **Linear Regression**: Arguably the simplest method, it estimates the linear relationship between a dependant (price), and some independent (all other features) variables. A computationally cheap and parallelizable method, but usually yields high bias since most of real-life datasets don't comply with the directly linear assumption.

- **Support Vector Machines**: Famous for its performance in Natural Language Processing [8] and in bioinformatics [7], but regarded as one of the most flexible and powerful classic ML methods overall. This method tries to fit an hyperplane to separate in the best way 2 different set of points, a non-linear function (kernel) can be applied to improve the performance in non-linear data. This can also be applied to regression by fitting the hyperplane to the data, trying to minimize the points outside of a threshold we define.

- **Random Forest**: This ensemble tree-method tries to reduce the variance of standard decision-tree method by training trees in different subset of the data and average their results.

- **Gradient Boosting Trees**: This ensemble tree-method tries to reduce the bias of standard decision-tree method by training trees by improving on the data points where the previous tree was not performant. It's famous to be good at tabular data [6].

The validation protocol we used is a *Cross Validation* of 5 sets, which means we will train each model 5 times, each one with a different train and test set.

Regarding hyperparameters, we have made use of *Grid* and *Random Search*. Along the several iterations of this project, several different hyperparameters have been tested, unfortunately, not every result from an iteration was recorded so we only have the results of the last batch of hyperparameter grid. This said, we list in Table 1 the ones we tested along the project.

Model	Hyperparameter	Description
Support Vector Regressor (svr)	kernel	Type of kernel used in the algorithm
	C	Regularization parameter
Random Forest Regressor (rfr)	n_estimators	Number of trees in the forest
	max_depth	Maximum depth of the trees
	min_samples_split	Minimum number of samples required to split an internal node
Gradient Boosting Regressor (gbt)	max_depth	Maximum depth of the individual trees
	max_features	Number of features to consider when looking for the best split
	min_samples_leaf	Minimum number of samples required to be at a leaf node
	n_estimators	Number of boosting stages to be run

Table 1: Hyperparameter Descriptions for Various Models

Results

To find the best set of hyperparameters, we used both a *Grid Search* and a *Random Search*, being the former a way to test using CV every possible set of combinations of a grid of parameters you specify, and the latter a way to just test a random subset of it.

You can find the results from each method summarized in this table The final

Model	R^2 Score
LinearRegression	0.00
SVR	0.452 96
RandomForestRegressor	0.517 41
GradientBoostingRegressor	0.3726

Table 2: R^2 Scores for Different Regression Models

model is the *Random Forest* since it's the one with highest R^2 score, and the estimation of its generalization performance is based on the fact that we used CV, so by averaging the 5 different Cross Validation scores we believe it's enough to estimate how good will the models generalize in unseen data.

The score is acceptable but far from being optimum, this means there are clearly some features that allow the different models to gain some predictive power, but due to hardware constraints we are unable to state whether a more extensive test of hyperparameters and/or more data could further improve the result.

Regarding hyperparameters, the one with an associated better result in *Random Forest* was a `min_samples_split` of 30, this means, that a bigger minimum number of samples required to split an internal node had associated a better result.

We also tried getting together the data from the 4 cities and remove the information regarding the neighbourhood, but the results were poorer, since the neighbourhood is a good predictor.

Conclusions

Overall, this study provides valuable insights into the factors influencing the Airbnb prices in four of the most visited cities in Spain, offering interesting information for both hosts and travellers. In addition, it has been proven the existence of dependencies between price and other features, and the importance of these features.

Furthermore, some advanced ML algorithms of different complexities where used to show the robustness of results and to compare the performance of the methods for the specific dataset used. Finally, let us notice that while the results are not outstanding, they still are satisfactory considering the hardware constrains.

Future work and limitations

Without the hardware limitations[1] we faced during this project, we would have been able to train against a more complete set of hyperparameters in the *Grid Search*.

It would also be interesting to see how merging the data available from more different cities affect the performance of the predictors. Along with testing with other kind of validation protocols, such as different methods of Cross Validation, like *Leave-One-Out CV*.

[1] We had to run every experiment in *Google Collaboratory* as our computers were not powerful enough to run a simple *Machine Learning* pipeline

Another future improvement is to use *Bayesian Hyperparameter Optimization*, to further optimise the best hyperparameters.

Finally, we could make further usage of ensemble methods, for example the *AdaBoost Regressor* or a *Voting Classifier*.

References

[1] OHCHR. (s. f.). UN expert urges action to end global affordable housing crisis. https://www.ohchr.org/en/press-releases/2023/10/un-expert-urges-action-end-global-affordable-housing-crisis.

[2] Barker, G. (2024, february 20). The Airbnb effect on housing and rent. Forbes. https://www.forbes.com/sites/garybarker/2020/02/21/the-airbnb-effect-on-housing-and-rent/

[3] Jain, S., Proserpio, D., Quattrone, G., & Quercia, D. (2021, january 15). Nowcasting gentrification using Airbnb data. arXiv.org. https://arxiv.org/abs/2101.05924

[4] Kanakaris, N., & Karacapilidis, N. (2023). Predicting prices of Airbnb listings via Graph Neural Networks and Document Embeddings: The case of the island of Santorini. Procedia Computer Science, 219, 705-712. https://doi.org/10.1016/j.procs.2023.01.342

[5] Moreno, G. (2019, january 22). Infografía: Las ciudades con más alojamientos de Airbnb. Statista Daily Data. https://es.statista.com/grafico/16721/anuncios-airbnb-espana/#:~:text=Tal%20y%20como%20muestra%20esta,anuncios%20en%20diciembre%20de%202018

[6] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci, "Deep Neural Networks and Tabular Data: A Survey," IEEE Transactions on Neural Networks and Learning Systems, vol. 35, no. 1, pp. 1-21, 2024, doi: 10.1109/TNNLS.2022.3229161.

[7] H. M. Muda, P. Saad, and R. M. Othman, "Remote protein homology detection and fold recognition using two-layer support vector machine classifiers," Comput Biol Med., vol. 41, no. 8, pp. 687-699, 2011, doi: 10.1016/j.compbiomed.2011.06.004.

[8] Y. Li, K. Bontcheva, and H. Cunningham, "SVM Based Learning System for Information Extraction," in Deterministic and Statistical Methods in Machine Learning. DSMML 2004, J. Winkler, M. Niranjan, and N. Lawrence, Eds., Lecture

Notes in Computer Science, vol. 3635, Springer, Berlin, Heidelberg, 2005, pp. 191-200, doi: 10.1007/11559887_19.

[9] Get the Data. (s. f.). http://insideairbnb.com/get-the-data/