Data Mining
Master in Management + Analytics, Business School,
Torcuato Di Tella University.

Contact Prediction Challenge

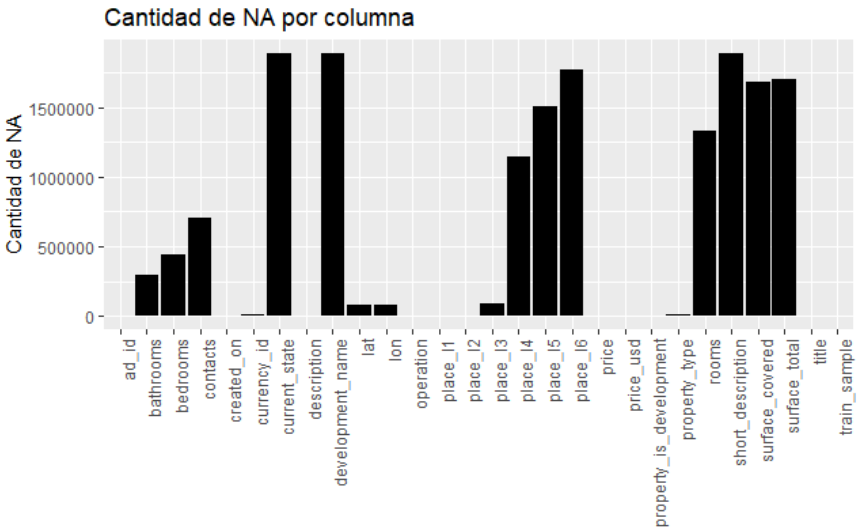
Report on the development of the prediction model for the publications of July, August, and September 2023.

Mauro Bertini and Lucas Veteikis

1. Exploratory Data Analysis

Null values

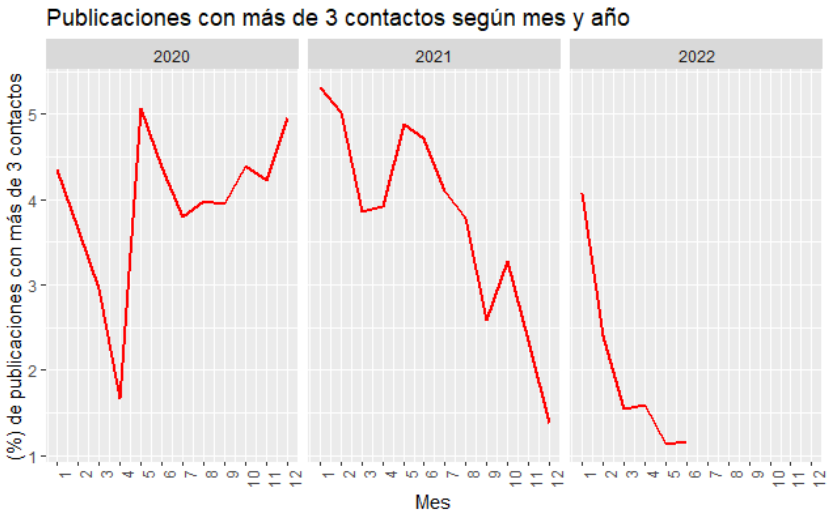
To start the exploratory data analysis, we began by analyzing the number of null values per column. This information is useful for excluding certain variables in machine learning models that do not accept null values, such as KMeans. It's also important to consider the quantity of null values when creating new variables because if we use variables for feature engineering that already have many null values, this characteristic will be replicated in the new variables and will affect their quality.



In this regard, the variables 'current_state,' 'development_name,' 'place_l6,' and 'short_description' have the highest number of null values. These variables should be avoided for creating new ones, or for some of them, a way to impute the missing values could be explored.

Time Factor

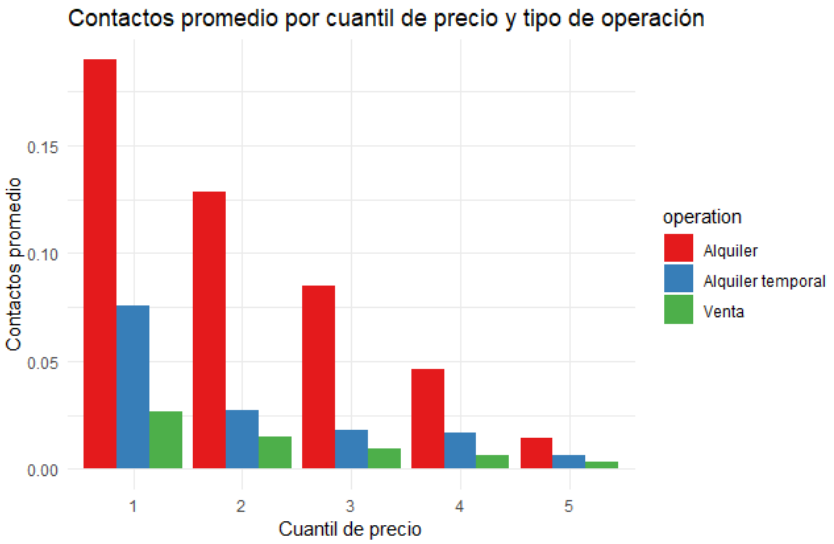
Due to the changes in the macroenvironment, we initially suspected that the temporal factor could have an influence. Therefore, the question we aimed to answer was, 'Is it worthwhile to consider all the data when modeling, or is it better to only take a portion?' We addressed this question by trying different models. Additionally, we conducted an exploratory analysis of the temporal factor and how it affected the response variable.



As observed, the proportions of postings with more than 3 contacts per month do not follow a clear trend over the years under evaluation. For instance, the month of May shows maximum values in 2020 and 2021 but hits a minimum in 2022. These discrepancies repeat across months, suggesting that data prior to a certain date could potentially impact the model adversely rather than improve it. This led us to consider the possibility of working with models using only the most recent months.

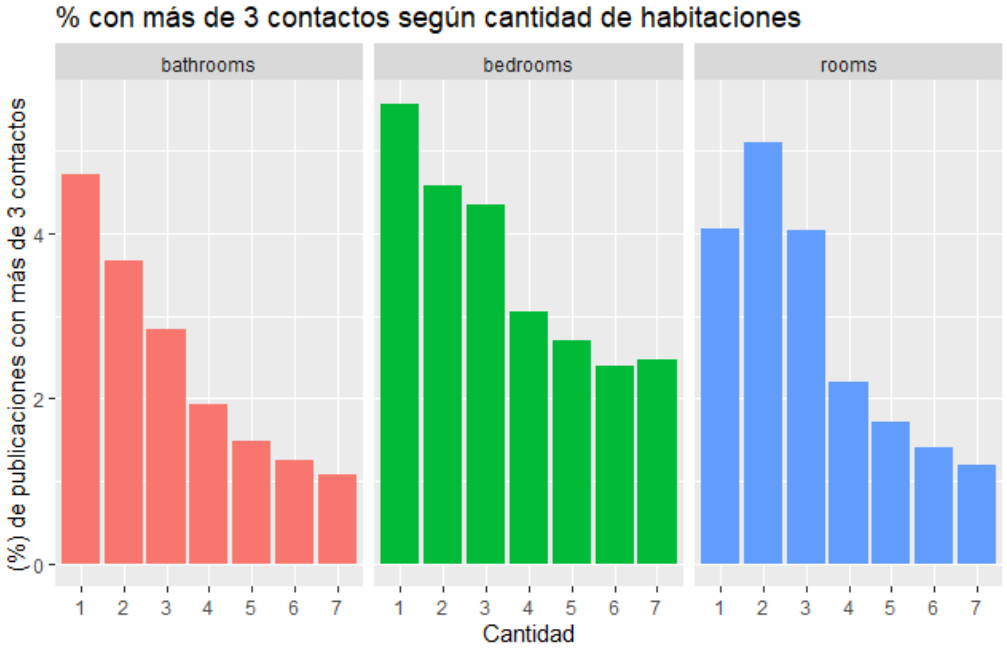
Variables which influence the response variable

- Price in USD



As observed in the graph, the price appears to be a determining factor in whether a posting will receive more than 3 contacts, with significantly more contacts for lower-priced postings. It's also interesting to note how the price difference between quantiles decreases according to the type of transaction, indicating that demand is more sensitive to price in rental listings and less so in sales.

- Amount of bedrooms and bathrooms



Similar distributions are evident across the three variables, where a higher number of rooms, bedrooms, or bathrooms corresponds to a lower percentage of postings with more than 3 contacts. This also suggests that these three variables could be good predictors of whether a posting will have more or less than 3 contacts..

2. Feature Engineering

Initially, all variables were selected for the exploratory data analysis conducted in the previous point. Based on this analysis, several decisions were made:

Modifications or removal of columns from the dataset

- The 'ad_id' column, which holds predictive power due to its temporal relationship (posts have higher ad_ids as time passes), was converted to numeric format and retained for the model. It was converted to numeric because using One-Hot Encoding (OHE) on this variable as a factor risks overfitting.
- The 'description' and 'title' columns, being unique to each post, couldn't be passed to OHE due to the risk of overfitting. For these columns, dummy variables were created that marked 1 if the description or title had a specific word and 0 if not. Subsequently, both variables were removed.
- The columns 'current_state,' 'development_name,' and 'short_description' were eliminated due to the high number of null values they contained. They were also analyzed for their potential to transform into other predictive variables, but their content didn't provide rich information for analysis.
- The variables 'place_l5' and 'place_l6' have more null values, possibly because their specificity refers to gated communities or more exclusive areas. Based on this premise, they could be interesting predictive variables, so they were modified to binary variables (1 and 0) to incorporate them into the model.

Created Variables

- New variables were created based on price, rooms, and areas, such as 'price_per_room,' 'price_per_surf_cov,' 'bathroom_proportion,' or 'bedroom_proportion,' which are ratios of variables already present in the dataset.
- A logarithmic transformation was applied to the price in dollars, leaving the original price variable in the dataset and including the new 'log_price_usd.'
- A variable related to price was created, representing the ratio between the price in USD and the average price of the geographic location (l1, l2, or l3). If this value is >1, it means the property is more expensive than the average in its location.
- Dummy variables were created based on the presence of certain words in the description or title of the post, such as 'barbecue', 'garden'' or 'beach.'

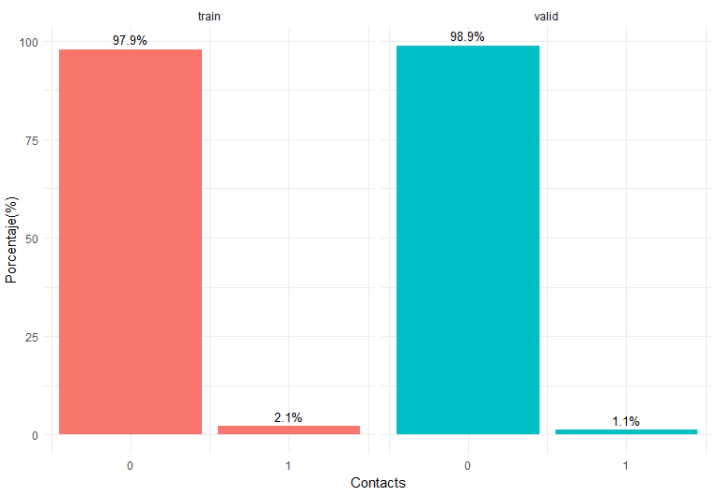
Created Variables not used

- In attempts where we used the complete database, we tried to add a dummy variable called 'pandemic', which would have a value of 1 for postings made during strict quarantine periods, such as March to May 2020. This variable didn't have the expected effect and was discarded once we decided to trim the database.

3. Models Validation

The decision to use AUC as the model performance measure was made because it performs better with imbalanced datasets (as is the case with this dataset). Decisions regarding model uploads to Kaggle were based on improvements in this measure during validation.
To construct an appropriate validation set, we chose to filter the dataset so that our validation set comprises the most recent months. We believe it's crucial to ensure that our model predicts well for the most recent postings, which are more similar to those the model will eventually be evaluated on in the evaluation set.
This concept arose due to the disparity between our model results and those displayed on Kaggle. We realized that having a metric indicating that a model performs well on old data, which is vastly different from the months in the evaluation set (July to September 2022), was not beneficial.
As a result, we identified errors in generating validation sets for previous models, leading to two decisions that, with further course progression, proved to be less effective:

- Performing a random sampling of our dataset to define the validation set: This introduced variability and included information from previous data (this point was more pronounced when we considered the dataset), resulting in a validation set that didn't provide a representative comparison with the subsequent months.
- Considering the improvement in AUC as a significant factor with that sampling: Our models that carried the error in the validation set consistently showed an improvement in AUC that wasn't reflected in the performance on Kaggle.



Another factor that could contribute to this difference was that Kaggle only sampled 30% of

the results to assign the score, but this was a factor beyond our control.

Based on this analysis, we concluded that the best way to construct the validation set was by taking the last two months of the dataset (May and June 2022). Following this validation set, the proportion of the validation class is similar to the model's training data.

4. Used Algorithms

The algorithm selection process was based on the progress of the classes, advancing in complexity as the course continued:

Naive Bayes

An attempt was made using Naive Bayes with an Argentina dataset from January 2020, dividing it into training and validation sets. A smoothing of 10 was used as a hyperparameter. Although this attempt was experimental and didn't aim to predict the original set effectively, it achieved a 75% accuracy.

Decision Trees

For the initial submission as a basic model, a sample of the previous three months was taken, assuming that more recent values would have greater predictive power. To construct the validation set, the k-fold cross-validation method was used, testing with 3, 5, and 10 folds, with the best performance achieved with 10 folds. The model was trained using the dataset's default variables, and according to the ROC metric, it resulted in a prediction score of 0.92 on Kaggle.

Xgboost

Multiple attempts were made to improve the performance of the XGBoost model. The initial attempts involved manually optimizing hyperparameters, adding variables, and adjusting data sampling. One attempt that showed promise with manually tuned hyperparameters resulted from using only data from May and June 2022.

Additional tests were conducted with different combinations of months and years. However, the conclusions indicated that more recent data yielded better results than earlier data, possibly due to the pandemic's influence on the economy during 2020 and 2021, or other external or business-related variables explaining the differences between the data from different years.

Finally, a random search was employed to optimize hyperparameters. The best-performing model resulted from XGBoost with this type of hyperparameter search. These latter models involved more robust feature engineering, which also contributed to their superiority. The approach to conducting an efficient random search was:

- First, we established a broad range of hyperparameters and trained 100 XGBoost models.
- Subsequently, we evaluated the model results and selected the minimum and maximum hyperparameter values that surpassed a certain performance threshold in

validation and were below a certain training performance threshold (to eliminate those prone to overfitting). This process involved using the "get_new_min_max" function, available in the functions file.

- With these new "optimal" minimum and maximum values for each hyperparameter, we trained another set of 20 models. From these 20 models, we selected the best-performing one and trained it with all the data for subsequent predictions.

Finally, the best model was tuned with the following hyperparameters: nround: 271, max_depth: 9, ETA: 0.1199, gamma: 4.5335, colsample_bytree: 0.3221, min_child_weight: 11.7117, subsample: 0.8261.