FACULTAD DE ESTUDIOS ESTADÍSTICOS

MÁSTER EN MINERÍA DE DATOS E INTELIGENCIA DE NEGOCIOS

Curso 2018/2019

Trabajo de Fin de Máster

TITULO: Rentalbility - Predicting the profitability of rental of properties in Madrid, a kick-off for a tool to help small investors.

Alumna: Priscilla Toscano Pinel

Tutor: Aida Calviño Martínez

Septiembre de 2019

UNIVERSIDAD COMPLUTENSE
MADRID

Table of Contents

Table of Figures

1. INTRODUCTION

Housing is one of the primary axes of needs and welfare in any society, and at the same time, one of the safest and most rentable way of investment, at least in Spain. According to a study carried out by Idealista, one of Spain's strongest online classifieds, real estate investment in Spain offers profitability rates that almost triple in the worst case those of the 10-year Spanish State Bonds, (Idealista, 2019a). The yield of ten-year Treasury bonds is 1.7%, according to the latest data from the Bank of Spain (Banco de España, 2019). Meanwhile, the gross rentability offered by the investment in housing and renting was 7.5%, during the first quarter of 2019 (Idealista, 2019a).

According to a similar study (Idealista, 2018a), which relates the purchase and rental prices of different real estate products to calculate their gross profitability among the Spanish capitals, Las Palmas de Gran Canaria is the most profitable with 7.1%. Meanwhile, in the most populated capitals, the profitability in Barcelona is 4.7%, lower than that of Madrid (5.2%) and Valencia (5.8%) (Figure 1).

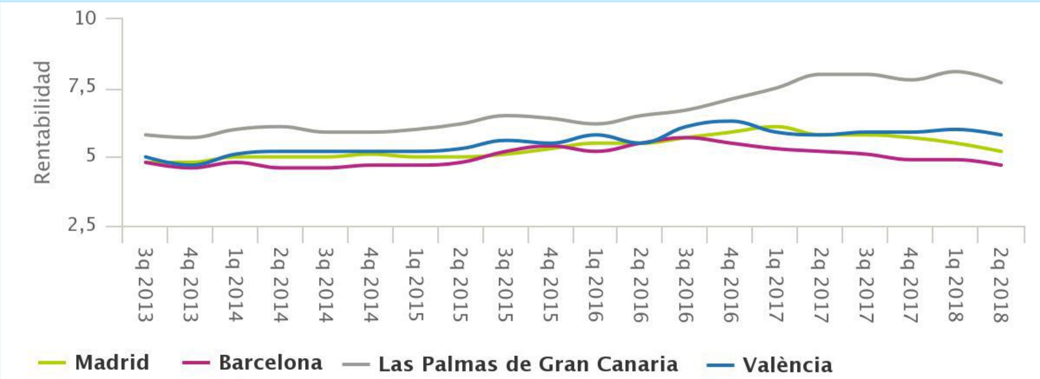


Figure 1: Evolution of the rentability of housing in Spain
Source: (Idealista, 2018a)

Moreover, Madrid's actual returning on buying a house is the lowest figure since the third quarter of 2015 (5.1%, in Figure 2). Another study from Idealista, (Idealista, 2018b) explains that the drop on the rentability reflects that sales price in Madrid is increasing extraordinarily higher, while the rental price does not rise at the same rate, as it can be observed in Figure 2.

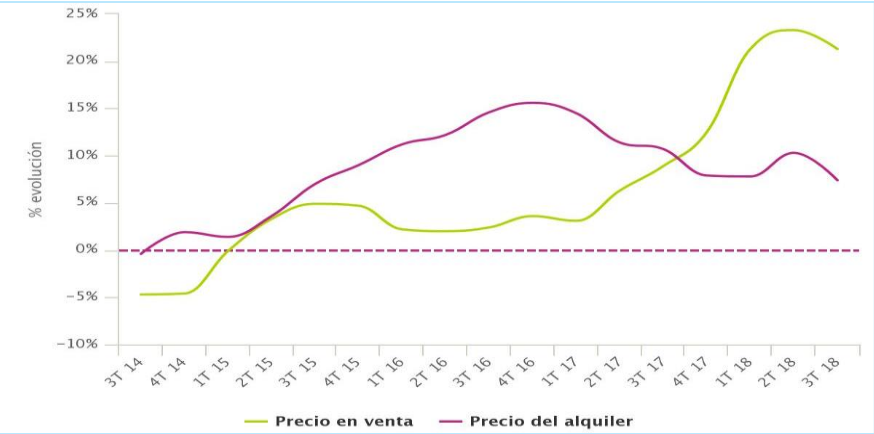


Figure 2: Evolution of housing prices in Madrid (Purchase vs. Rent)
Source: (Idealista, 2018b)

On the other hand, there is the fast-growing market for vacation rentals. These refer to lodging, and in this specific case homes and apartments, which are offered to be rented on online market places for a short period. Usually, the companies which offer these services, such as Airbnb, HomeAway, and Rentalia, do not own the properties; they act as a broker. In most of the cases, the host is the owner of the house, a tenant, which use them as a source for extra income or a third party property management corporations, which have the short-term rental as their business model.

Airbnb is the biggest "people-to-people" (peer-to-peer) vacation rental platform, with more than 6 million listings in 191 countries (Airbnb, 2019a). Airbnb was born in 2008 as AirBed and Breakfast, and soon became a "unicorn"[1] of the sharing economy[2] in Silicon Valley (Biz, 2016). However, together with the fast expansion of this business model came the regulation battles with local governments to control and legalize the vacation rentals.

The discontentment of locals and the legislative wars also impact Airbnb in Madrid, where more than 60% of the bookings happen in the Center district (Colliers International, 2018). The platform has 17300 listings (64.7% are entire apartments) and 10700 active users only in Madrid, which makes this Capital the largest Market of Spain (Airbnb, 2019b). In Figure 3, we can see the Airbnb distribution within the city center. In April 2019, the government of the Community of Madrid approved the local regulation for controlling the tourist activity and avoiding agglomerations in the lodging. The regulation establishes: as lodge one house that is rented 90 days a year or more; as requirement owning a license to operate; a maximum ratio of guests based on the number of useful square meters of the house (Comunidad de Madrid, 2019).

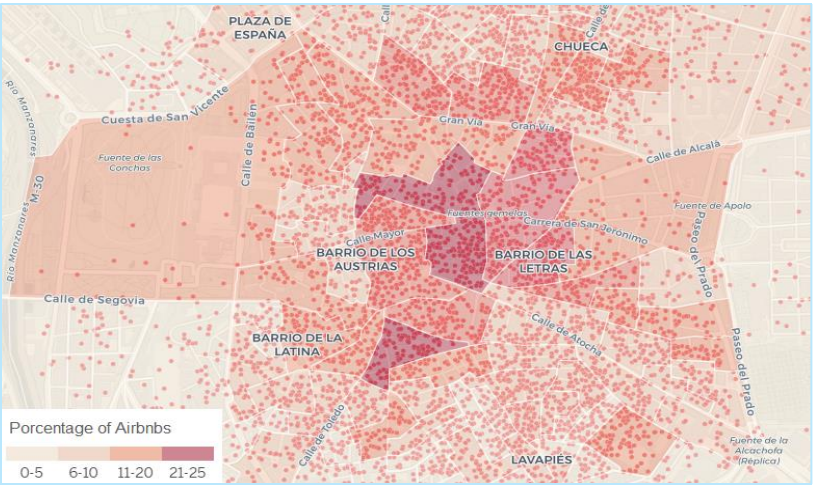


Figure 3: Geographic concentration of Airbnb accommodations in Madrid
Source: El País (2019)

---

[1] "A start-up (= new business) whose value is considered to be over $1 billion". ("UNICORN | meaning in the Cambridge English Dictionary," 2019)

[2] "An economic system that is based on people sharing possessions and services, either for free or for payment, usually using the internet to organize this." ("SHARING ECONOMY | meaning in the Cambridge English Dictionary," 2019)

On the 2018 Madrid Activity Report, Airbnb affirmed the local hosts earned 132 million euro and the mean annual income for a typical host is 5022€ in this year (Airbnb, 2019b). Nevertheless, the company advertises in their homepage for hosts that they could earn around 18000€ yearly by renting a whole place at least 15 nights on a month (Airbnb, 2019c). Despite the regulation, the increasing number of Airbnbs and its profitable earnings suggests to us that this trend that came to stay.

When talking about property rentals, we see the two sides of a coin with vacation and traditional rentals. We have the same product, addressed for two different markets and target, but with a single goal, take the profitability out of it. Therefore, understanding which factors affects the real estate market and nuances of each rental strategy is not so easy as it may seem. Besides the natural fluctuation of both markets, there are several factors which make every property conditions unique. Deciding whether to buy or not a property to invest in markets so diverse could be a big challenge for an individual since the optimal pricing or the proper channel to publish it may not follow the common sense. Thus, we see an opportunity to develop a tool to help this investor.

With this tool, we wish to help in this decision-making challenge of real estate investors. The following work attempts to provide a tool to predict the returned investment for a rental property taking into consideration a multichannel strategy for both the short-term and long term. The Return on Investment (ROI) is a very well known profitability measurement used in most investment decisions. It is a ratio of the total benefits of an investment divided by its costs. Furthermore, we wish to provide investors with a platform containing a data analytics package assisting them to comprehend each rental model and outperform in their investment.

The platform we aim to develop with the help of this study has already two similar applications in the United States. The first one is Mashvisor, a property search engine which calculates the ROI for houses in both rental channels, the traditional and Airbnb, however, it is only available in the US (Mashvisor, 2019). The second one is AirDNA, which is already available in Spain and provides an in-depth market analysis for vacation rentals properties across two market-places, HomeAway and Airbnb, including a tool which predicts the annual revenue, occupancy rate, and average daily price. We see the future of our platform as the combination of both of them but specialized in the European market. On this work, we used both tools as a guide and a benchmark comparison.

We decided to focus and limit this research on Madrid's market for the first instance, because besides being the capital of Spain and the most populated city, Madrid sets the tendency of Spanish real estate market. Fernando Encinar, Idealista's Co-founder, said to the Spanish newspaper elEconomista: "The experience tells us that the Real Estate market from Barcelona and Madrid set the tendency and give us an idea towards which direction the other markets will go on the mid-term"(elEconomista.es, 2018). Madrid is also the city in Spain with more Airbnb users, accordingly with (Airbnb, 2019b).

In order to successfully develop a reliable tool, we deployed several data mining techniques in a different dataset for each rental case, Idealista, and Airbnb. First, we needed to be able to predict the rental income for each case, for secondly, be able to apply these models on a third dataset of properties on sales in Madrid, for thirdly, be able to calculate the Return on Investment based on the ratio of them. We calculated the ROI, for payments in cash and the ROIM for properties financed with a mortgage. With these data, we performed data analysis to better understand the market and evaluate the viability of the tool. In the end, we also did a short study with the Airbnb dataset, but with a different target variable to predict whether a house would be frequently occupied or not based on their occupancy rate. It is indispensable to emphasize that in this dissertation, we limited our work to the model's evaluation and analytics for only Idealista and Airbnb as a feasibility study for the future development of the application.

1.1. Project Justification

Deciding where, when and how to invest it is not elementary, there are several variables that one should investigate and take into consideration, even more, when we talk about a decision which requires a significant amount of capital. Consequently, the more information about it, the better. However, nowadays, it is still a big problem. The amount of information is so overwhelming that instead of clarifying our minds it makes it blur.

The decision should be assertive and maximize the profit, therefore, a methodology that calculates the Return on Investment, which already takes into consideration the relevant variables could be an excellent focused guide for the small investor, who aims to become a landlord without much knowledge about it.

Furthermore, having the rental price of a house given by a predictive algorithm is more reliable than when given by a subjective assumption of a person. Since the model follows a methodology based on the relationship between several variables and uses real-time market data. Thus, it provides a fair price for both landlord and tenant, creating a trustworthy relationship between both parts.

1.2. Project Goals

The main goal of this project is to propose a methodology to calculate the Return on Investment (ROI) of a property by the renting in the short term, as for vacation rentals, and long term, for one year, in Madrid. The secondary purpose we have in this study is to be a kick-off for the development of a platform, in the form of web page or APP, to help small individual proprietaries on their decision-making process to buy a house and how to invest in it.

Furthermore, we aim to understand which variables influence the rental prices of properties in Madrid. Finally, with this research, we would like to provide Madrid's public entities with another study to understand the fast-growing housing rental market and possibly assist the development of solutions with a positive social impact on the public policies level and promote the social cohesion of the different stakeholders in the city.

2. METHODOLOGY

This project had a duration of 7 months, starting on February 2019 and finishing in September 2019, and it was accomplished in Madrid, Spain. During the study, we made use of several software programs and languages. We started with Spyder (3.7

Python) to extract the data from Idealista's API, SAS Enterprise Miner 14.1 to perform the data exploration and cleaning, R Studio Version 1.1.463 for the development of models (with Caret library), and Microsoft Power BI for the final analytical study. We also used SAS Base 9.4 for the occupancy rate study and Excel as complementary tools for specific needs. The steps we followed for each dataset in this study are described below:

1) Data Acquisition, one for each rental channel (Idealista and Airbnb);
2) Data Exploration and Statistical Analysis;
3) Models Development and Evaluation;
4) Rent Prediction (with a dataset for houses on sale in Idealista);
5) Return on Investment Calculation (ROI and ROIM);
6) Data Analytics;

For the occupancy rate models, we used the same Airbnb dataset after the data modifications, executed the models the using SAS Base.

During this dissertation, we refer to long-term renting the cases when the contract established a minimum period of one-year rent. Thus, we consider Idealista the most appropriate data and information source since it is the most significant online platform available on the market. Nevertheless, for short-term renting, we consider vacation rentals, hence the length is smaller than one year, and the most suitable source of data is Airbnb since around 83% of vacation rentals in Madrid are listed on this platform (AirDNA, 2019).

2.1. ROI Calculation

Since the primary goal of our project is to develop a tool capable of calculating the ROI, an explanation of this common finance index is necessary. The ROI (Return on Investment) is also often called ROA (Return on Capital) (Brealey et al., 2011, p. 299). According to Gitman (2004), it measures the overall effectiveness of management in generating profits with its available assets (Gitman, 2004, p. 65). In easy words, it is a performance index to evaluate the rentability of an investment.

Gitman (2004) describes the formula as:

$$Return\ on\ total\ assets\ = \frac{Earnings\ available\ for\ common\ Stockholders}{Total\ Assets}$$

This formula can be easily translated as total earnings (or benefits) of an investment divided by its costs. Therefore, the result of this formula is a percentage (or ratio), which allows this measure to be easily comparable and interpretable. The higher the ratio, the higher the return on this investment.

Another advantage of this measurement system is that it is quite flexible to the kind of investment we want to evaluate, which is, in our case, the ROI on rental properties. When purchasing a property, besides other possibilities, it is possible to realize the transaction by cash or via financed transactions. On this project, we will focus on the return for both possibilities.

In the specific case of an investor buying a house and paying it by cash, the ROIC (Return on Investment in Cash) formula is straightforward (Folger, 2019):

$$ROIC = \frac{Yearly\ Rental\ Income}{Property\ purchase\ price}$$

Moreover, if we have an investor, who is buying a property with a mortgage, we would have to use another formula, which should take into consideration the interest paid over the years and the downpayment. The ROIM (Return on Investment with Mortgage) should be the yearly rental income divided by the total investment. The total investment is the purchase price of the property (*p*) plus the interest (*i*) paid over the financed period (*t*) minus the percentage of the downpayment needed for the mortgage (*d*) (Folger, 2019).

$$ROIM = \frac{Yealy\ Rental\ Income}{p + (p * i * t) - (p * d)}$$

As an illustration, let us imagine we have a house which purchase price (*p*) is 100000€, and the rental price is 1000€ per month. If the investor pays the house in cash, the ROI would be 12%.

$$ROIC = \frac{Yearly\ Rental\ Income}{Property\ purchase\ price} = \frac{1000 * 12}{100000} = 12\%$$

However, if the investor decides to take a mortgage with 2,25% fixed interests, over a 30-years loan and a downpayment of 20% of the purchase price, the ROIM would be 8,13%.

$$ROIM = \frac{Yealy\ Rental\ Income}{p + (p * i * t) - (p * d)} = \frac{1000 * 12}{100000 + (100000 * 0.0225 * 30) - (0.2 * 100000)} = 8{,}13\%$$

It is important to emphasize that in this methodology, we use only the sale and rent prices to calculate its gross profitability. In order to obtain the net income that offers a real estate investment, the investor must count on the additional expenses for each rental cases. The only expenses we take into consideration are the utility, internet costs, and the service fee for the Airbnb case, we describe the calculation of them on section 4.1.

2.2. Data Mining Guideline

The previously mentioned steps we followed on the development of this work are based on a methodology developed by SAS Enterprise called SEMMA, which is an acronym for Sample, Explore, Modify, Model, and Assess (SAS, 2018). It guides the deployment of data mining projects. According to SAS, the process of this methodology consists in:

- **Sample** — Identify and set up roles for variables;
- **Explore** — Explore data sets statistically and graphically, obtain descriptive statistics, identify relevant variables and perform association analysis, among other tasks;
- **Modify** — Prepare the data for analysis: transform existing variables, identify outliers, replace missing values, perform cluster analysis ;

- **Model** — Develop a predictive model for a target variable. This is the most critical phase of this project since we need to develop models that will adjust to data that is continuously updating once the platform runs. Therefore we train seven (neural network, random forest, gradient boosting, extreme gradient boosting, support vector machine different machine learning models and exhaustively search for the best configuration for each;
- **Assess** — Compare competing predictive models. To evaluate the best model we use repeated cross-validation. The best model selected in this phase, for each rental channel, would be the model running on the behind in our application, and their predictions used to calculate the ROI.

We aggregate the Sample, Explore and Modify phase together on the Data Exploration in SAS Enterprise Miner Section.

2.3. Data Mining Techniques and Methodology

In order to accomplish the prediction of the rental income for a property, we need to make use of several supervised machine learning techniques. Since our target variable is continuous, the yearly income, our models are regressions, and they aim to predict more than explain. Below, we present a brief description of each algorithm we used on the project. All explanations are based on class notes and manuscripts of Portela (2019) during the Machine Learning lessons at Complutense University of Madrid. On this project, we tuned and trained every model in order to obtain the most suitable architecture and parameters for each model. Although some parameters can have different names in SAS and R, the concepts are the same. Hence, we focus on explaining the parameters for R because this was the main software on the modelization. Further information regarding these specific configurations will be described during the data analysis section.

2.3.1. Machine Learning Algorithms

- **Neural Network** (NN) - Neural Networks are composed by interconnected nodes (or units) forming multi-layer networks. They have an input layer, which is connected with one (or more) hidden layers, and this to the output layers (the predictions). Neural Networks consist of a functional approach to the relationship between input and output variables. It imitates the operation of brain neurons, where each neuron processes and combines different stimuli from the other neurons with which they are connected.

  In general, the number of units for a Neural Network with one layer and one variable output is given by the formula: $h\,(k + 1) + h + 1$, where h = number of hidden nodes, k = number of input nodes. The ideal is having between 10 or 25 observations per parameter.

  Like their human analogy, they are designed to recognize patterns and learn from them. Each of these connections carries a weight that adjusts as learning proceeds. Neural Networks work better than the usual statistic models if the relationships are nonlinear or complex. Therefore they require activation functions to introduce non-linearity to solve complex problems. The optimization function helps neural networks to improve their accuracy by estimating the parameters in order to minimize an error function.

Within the Caret library in R, there are two different functions to train Neural Networks, nnet and avNNet. The difference between then is that meanwhile the nnet is a single-hidden-layer neural network, the avNNet, aggregates several neural network models. They both have the same parameters to tune:

- *Size* = number of units in the hidden layer
- *Decay* = weight decay (= learning rate)

With SAS base we can also define an activation and optimization function. The activation function can be Tahn or Softmax. The optimization functions can be Levenberg-Marquardt (LEVMAR) or Back Propagation (Bprop).

- **Random Forest (RF) and Bagging** – Random Forest, Bagging, Gradient Boosting and Extreme Gradient Boosting are tree-based methods. They combine the output of several trees by averaging them. They aim to improve the stability and predictions of the models, besides reducing its variance and avoid overfitting.

  The Bootstrap Aggregating (Bagging) was the first of these methods, where we build various trees, with different sets of observations, and then the average of the predictions is obtained. Random Forest algorithms differ from the previous one on the introduction of random samples of variables (mtry) during tree construction.

  The Caret library has the following parameters to be tuned:

  - *Mtry* = Number of random variables used in each tree
  - *Ntree* = Number of trees used in the forest
  - *Sampsize* = percentage of the observations used in each tree
  - *Nodesize* = minimum number of observations in each terminal node

- **Gradient Boosting** (GBM) – Gradient boosting algorithm consists of the same process of the previous tree-based methods, but with a slight modification of the predictions by using a regularization factor (shrinkage), which tries to minimize the residuals. Since this model constructs different trees each time and adjusts the predictions by minimizing the errors, some trees correct others. The flexibility and adaptation of the method improve the construction of a single tree. This process has to be monitored in principle by early stopping to determine the number of iterations.

  - *Shrinkage* – parameter of regularization, it reduces the influence of each individual trees and sets the speed of adjustment: when it is lower, it is slower and needs more iterations, but the adjustment is more precise.
  - *n.minobsinnode* - the maximum size of end nodes
  - *n.trees* - the number of iterations (trees)
  - *interaction.depth* - number of splits it has to perform on a tree
  - *bag.fraction* - the percentage of the observations used in each tree

- **Extreme Gradient Boosting** (XGBM) – XGBoost is one implementation of Gradient Boosting framework, with more regularization factors (gamma, lambda, alfa) to control over-fitting, which gives it better performance and speed. XGBoost algorithm was developed as a research project at the University of Washington by Tianqi Chen and Carlos Guestrin in 2016 (Morde, 2019). XGBoost optimizes standard GBM algorithm through systems optimization and algorithmic enhancements. The key system optimization are: Parallelization, it uses parallelized implementation to lead the process of sequential tree building using, and Tree Pruning, it uses 'max_depth' parameter to prune trees backward. Within the algorithmic enhancements, we can highlight its regularization, which penalizes more complex models through both LASSO (alpha) and Ridge regularization (lambda) to prevent overfitting, and the built-in cross-validation method (Morde, 2019).

- *nrounds* - the maximum number of Boosting Iterations
- *max*_depth - Maximum Tree Depth
- *eta* (= Shrinkage) – regularization parameter, controls the learning rate and it is used to prevent overfitting by making the boosting process more conservative
- *gamma* - Minimum Loss Reduction
- *colsample_bytree* - Subsample Ratio of Columns
- *min_child_weight* - Minimum Sum of Instance Weight needed in a child
- *subsample* - the percentage of the observations used in each tree

To better understand this "family" of algorithms, we can see in figure 4 the summary of the tree-based algorithms, their evolutions and differences.

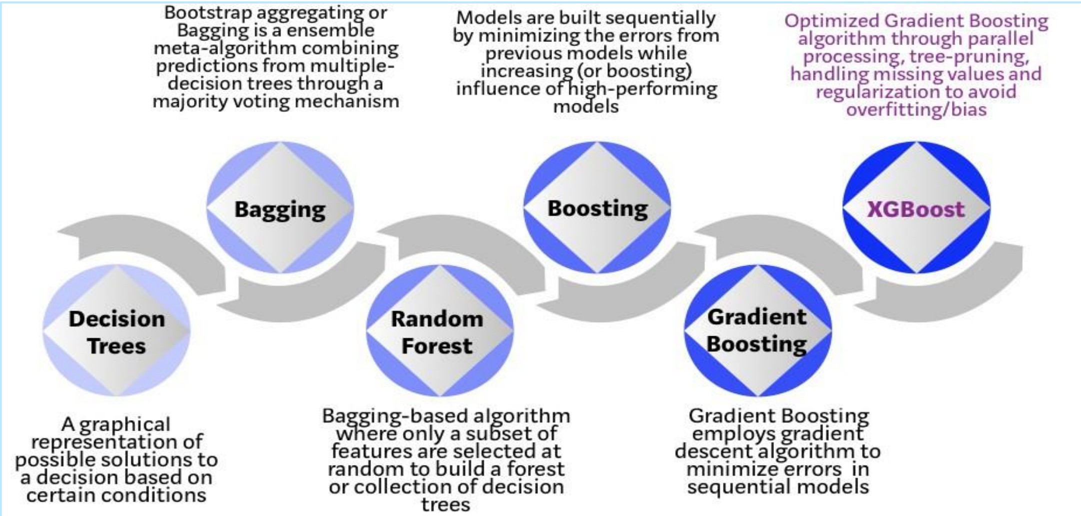


Figure 4: Evolution of Tree-Based Algorithms
Source: (Morde, 2019)

The **Out-of-bag** (OOB) error, is the measurement method we used to evaluate the prediction error of tree-based algorithms in this thesis.

- **Support Vector Machine** (SVM) – This algorithm aims at finding the optimal hyperplane which maximizes the margin between classes with algebraic methods. It treats non-linear separable data with a kernel function, which transforms the data into a higher dimensional one to make it possible to perform the linear separation. There are four types of kernel functions, but in this thesis, we are only using the linear and the radial basis (RBF).
    - *C* - the cost associated with misclassification (for both linear and RBF)
    - *Sigma* (= RBF factor) – it is the regularization parameter for the RBF function, increasing sigma in the RBF function implies less bias and greater overfitting.

- **K-nearest neighbor** (K-NN) - In classification, this algorithm tries to assign to each observation the most frequent value of the k observations closest to it, in other words, this algorithm uses the similarity of the training data to classify the new cases. The only parameter we can tune with this model is K, which is the number of most similar cases (neighbors).

- **Ensemble** - This method consists of the construction of predictions from the combination of the results from other models. The objective of this technique is to improve the prediction of the models since one model correct the others. Another advantage of this process is the reduction of the prediction's variance. On the downside, these models are much more complex and hard to interpret.

2.3.1. Data Mining evaluation and optimization techniques

- **Early stopping** (ES) – This is an optimization technique to avoid overfitting of the models to the training data. Early Stopping consists of dividing the data into training and validation, and stop the estimation process when the error in the validation data begins to increase.

- **Cross-Validation** (CV) - We use this technique to evaluate the fit of the models to other data set. It reduces the dependence between the data partition of the sample used for the training of the model and the data partition used for the validation. The CV divides the data randomly in k groups, separating i set and building the model with the rest of groups (k-i), the error is estimated with set i.

- **Repeated Training-Test** - It is another technique for evaluating the predictive capacity of the model. It consists of splitting the data into train and test partitions and running the model repeated times with random the seeds. Cross-Validation is more effective than this technique but demands more time to execute.

- **Root Mean Square Error (RSME)** –is the standard deviation of the residuals (prediction errors), it helps to evaluate how concentrated the data is around the line of best fit.

- **Coefficient of Determination (R^2)** – it is one statistic measurement for the effectiveness, the capacity of explanation, of the model. The R^2 formula is:
  $$R^2 = \frac{SSM}{SST} = 1 - \frac{SSE}{SST},$$
  which is the **S**um of **S**quare of the **M**odel (which measures the information contained in the model) or **S**um of **S**quare **E**rrors (which measures the error made), by the **S**um of **S**quare **T**otal (which measures the error which incurred if there is no model). It takes values close to 1 when the model is more accurate.

On the following two chapters, we explain how we used each of these machine learning models and applied the previously explained techniques to build and evaluate the best model for the *Rentalbility* platform.

3. IDEALISTA DATA ANALYSIS

On this chapter, we explain every step we took in the whole data analysis process for properties available on Idealista, from the data acquisition until the construction and evaluation of the model. The outcome of this process is the model which we will use to calculate the rental predictions for the properties in traditional renting.

3.1. Data Source

The Idealista dataset was obtained between March and April 2019 using the Idealista API. The original dataset had 40 variables and 7139 observations. On this dataset, we found location-related variables, such as latitude, longitude, neighborhood and district; property characteristics, as the number of room and bathroom, size, parking and lift presence; and price-related variables, as monthly rent and price per square meter. A summary of all original variables and its explanation can be found in Appendix A.

Idealista facilitates access to its data via API upon request on their webpage[3]. They provide an API key and a secret to access the data via OAuth authentication. The API is limited to 100 requests per months and one request per sec. However, it was possible to request an extension on the number of requests, which allowed us to get 1000 requests per month. The outcome of these requests was in JSON format.

Idealista does not provide any further information on the coding to access the data. Therefore we needed to develop a code in python. We got the base on Stackflow (Manelmc, 2016). However, it was obsolete, and it was required to adapt it to our needs. The final and complete is available in Appendix B.

We used the libraries *panda*, *JSON*, *urllib*, *requests* and *base64*. The code is composed of three parts. The first is the GET function, where we establish the session with Idealista host and insert the OAuth2 credentials.

```
def get_oauth_token():
    url = "https://api.idealista.com/oauth/token"
    apikey= 'wgl3146s53142x7wwos95kilvznto8yl' #sent by idealista
    secret= 'hD46ajexp35t'  #sent by idealista
    auth = base64.b64encode(bytes(apikey + ':' + secret, "utf-8"))
    headers = {'Content-Type': 'application/x-www-form-urlencoded;charset=UTF-8' ,'Authorization' : 'Basic ' + auth.decode("utf-8")}
    params = urllib.parse.urlencode({'grant_type':'client_credentials'})
    content = rq.post(url,headers = headers, params=params)
    bearer_token = json.loads(content.text)['access_token']
    return bearer_token
```

The second part is the search query, which results in the JSON file.

```
def search_api(token, url):
    headers = {'Content-Type': 'Content-Type: multipart/form-data;', 'Authorization' : 'Bearer ' + token}
    content = rq.post(url, headers = headers)
    result = json.loads(content.text)
    return result
```

[3] http://developers.idealista.com/access-request

Finally, the third part is composed of the filters we want to apply to our search. The original code had ten options of filtering, such as country, publication date, property type, operation type, or location. Notwithstanding, we considered the presence of some special items (such as swimming pool, balcony, air conditioning, and complete furniture or only kitchen's furniture) could play an essential role in the rent price. Unfortunately, these characteristics were only available as additional filters and not as variables. Hence, we had to force the API to provide this information by using a loop *for* in the filters, in a way that it would return "true" if the filter for each of that elements was activated and "false" otherwise. Subsequently, we had three boolean variables for swimming pool, balcony and air conditioning, and one binary for *furnishedkitchen* or furnished.

```
country = 'es' #values: es, it, pt
locale = 'es' #values: es, it, pt, en, ca
language = 'es' #
max_items = '50'
operation = 'sale' # sale or rent
property_type = 'homes'
order = 'publicationDate'
center = '40.4167,-3.70325'
distance = '15000'
sort = 'desc'
bankOffer = 'false'

df_tot = pd.DataFrame()
limit = 2

for i1 in ('true','false'):
    for i2 in ('true','false'):
        for i3 in ('true','false'):
            for i4 in ('furnished','furnishedKitchen'):
                for i in range(1,limit):
                    url = ('https://api.idealista.com/3.5/'+country+'/search?operation='+operation+#"&locale="+locale+
                           '&maxItems='+max_items+
                           '&order='+order+
                           '&center='+center+
                           '&distance='+distance+
                           '&propertyType='+property_type+
                           '&sort='+sort+
                           '&numPage=%s'+
                           '&airConditioning=%s'+
                           '&swimmingPool=%s'+
                           '&terrance=%s'+
                           '&furnished=%s'+
                           '&language='+language) %(i,i1,i2,i3,i4)
                    a = search_api(get_oauth_token(), url)
                    df = pd.DataFrame.from_dict(a['elementList'])
                    df['AC'] = i1
                    df['Piscina']=i2
                    df['Terraza']=i3
                    df['Amueblado']=i4
                    df_tot = pd.concat([df_tot,df])
```

This maneuver had several challenging consequences. The first one involved the boolean filters. The "true" filter only selected houses with the characteristic we defined, the "false" filter meant no filters were applied, instead of houses without it and therefore this filter could not be fully trusted. The second issue was that this maneuver gave us 16 possibilities (four loops raised by two filters possibilities) of filters and the loop needed to go through the data 16 times. The third problem was that each loop was counted as one request and should be multiplied by the number of pages to get the total of requests in the extraction. That meant that one-page extraction was equivalent to 16 requests, which made us quickly exhaust the 100 requests limit of requests per month in a couple of extractions. To overcome this, we asked for an extension of requests to 1000. This limitation, together with the one request/sec precluded us from extracting all information at once. Consequently, it was needed to download only one dataset per day. Each dataset was composed by approximated 800 observations (maximum items per page (50) by 16 requests).

These three issues together entailed in a fourth issue, which caused the query to download the same houses several times but with different "false" filters. To overcome this problem, we had to deeply analyze the "false" values to understand its behavior. The conclusion was to convert the TRUE into "1" and FALSE into "0". Since the TRUE was always the truth, but the FALSE was not always trustworthy, if we summed the zeros and ones, the higher sum would be the right one. After that, we organized the URL alphabetically, the SUM from higher to small and created a rule to define the right observation to keep. The rule defined as the right property(rule = 1) the one with a higher sum or with property ID unique. In Figure 5, we can see a sample data and the formula we used to determine the right observation.

AJ	AK	AL	AM	AN	AO	AP	AQ	AR	AS	AT	AU
Property ID	AC	Piscina	Terraza	Amueblado	Extraction_Date	AC_B	PISCINA_B	TERRAZA_B	SUM	Count if	Rule
1067532	TRUE	TRUE	FALSE	furnishedKitchen	22/03/2019	1	1	0	2	2	=IF([@[Count if]]=1;1;IF(AND(AJ2=AJ3;AJ2<>AJ1);1;0))
1067532	FALSE	TRUE	FALSE	furnishedKitchen	22/03/2019	0	1	0	1	2	0
1166175	TRUE	TRUE	FALSE	furnishedKitchen	24/03/2019	1	1	0	2	3	1
1166175	TRUE	TRUE	FALSE	furnishedKitchen	23/03/2019	1	1	0	2	3	0
1166175	FALSE	TRUE	FALSE	furnishedKitchen	23/03/2019	0	1	0	1	3	0
1292763	TRUE	FALSE	FALSE	furnishedKitchen	14/03/2019_2	1	0	0	1	1	1
1316270	TRUE	TRUE	FALSE	furnished	01/04/2019	1	1	0	2	3	1
1316270	TRUE	TRUE	FALSE	furnished	31/03/2019	1	1	0	2	3	0
1316270	FALSE	TRUE	FALSE	furnished	31/03/2019	0	1	0	1	3	0
1417019	FALSE	TRUE	TRUE	furnished	28/03/2019	0	1	1	2	9	1
1417019	FALSE	TRUE	TRUE	furnished	27/03/2019	0	1	1	2	9	0
1417019	FALSE	TRUE	TRUE	furnished	25/03/2019	0	1	1	2	9	0

Figure 5: Idealista sample data after modifications

We also faced some issues with the encoding, but we overcame them by converting to UTF 8 with excel query.

3.1.1. Prework

Before we could start the data exploration with SAS Enterprise Miner, we executed some manual adjustments with excel beside the one already mentioned above.

- Filter Municipality – Madrid only
- Creation of three variables from the text variable *parkingSpace - Has_Parking* (Boolean), *Parking_Price_Included* (Boolean), *Parking* (the combination of these previous variables with three levels: *"Yes, Price included", "No Parking, Yes", "Price NOT included")*
- Calculation of Yearly_Price (target variable): $Yearly\ Price = rent\ price * 12$

3.2. Data Exploration in SAS Enterprise Miner

After data pre-processing in Excel, we started our mining work using SAS Enterprise Miner with 7139 observations and 25 variables.

We assigned the roles accordingly with the functions and types of each variable. On the table below (Figure 6), we can find a summary of the variables we with worked.

Variable Name	Role ▲	Measurement Level
propertyCode	ID	Interval
AC	Input	Binary
Amueblado	Input	Binary
Has_Parking	Input	Binary
Parking	Input	Nominal
Parking_Price_Included	Input	Interval
Piscina	Input	Binary
SUM	Input	Interval
Terraza	Input	Binary
bathrooms	Input	Nominal
distance	Input	Interval
district	Input	Nominal
exterior	Input	Binary
floor	Input	Nominal
hasLift	Input	Nominal
hasPlan	Input	Binary
hasVideo	Input	Binary
latitude	Input	Interval
longitude	Input	Interval
neighborhood	Input	Nominal
numPhotos	Input	Interval
propertyType	Input	Nominal
rooms	Input	Nominal
showAddress	Input	Binary
size	Input	Interval
_Yearly_Price	Target	Interval

Figure 6: Idealista Variables Role and Levels

We executed an analysis of the data to detect missing data, anomalies and trends. In addition, we did a descriptive analysis to understand the relationship between variables and observations. We also created a random variable, intending to help us define the least valuable variables.

3.2.1. Interval Variables Statistical Analysis

As we can see in the interval observations statistical values below (Figure 7) the dataset did not have any missing. The minimal and maximum were within the supposed limits. Nonetheless, the target variable, *Yearly_Target*, reached a maximum amount of 240.000 € (which means a monthly rental of 20.000€), for this reason, we decided to filter them out, by setting a limit for the annual rent of 50.000€. We also set up a limit of 220m^2 for the *size* (of the property) variable. Most of our variables were symmetrical or not much asymmetrical (the skewness of *size* and *yearly_price* reduced once we set the filter).

Variable	Role	Mean	Standard Deviation	Non Missing	Missing	Minimum	Median	Maximum	Skewness	Kurtosis
Parking_Price_Included	INPUT	0.304945	0.460416	7139	0	0	0	1	0.847537	-1.28204
Random	INPUT	0.500893	0.287387	7139	0	0.000265	0.500591	0.999875	-0.00434	-1.19155
SUM	INPUT	1.55526	0.842696	7139	0	0	2	3	-0.01871	-0.60014
distance	INPUT	4447.351	2923.035	7139	0	15	3757	14409	0.639064	-0.50727
latitude	INPUT	4.0436E8	312394.5	7139	0	4.0334E8	4.0434E8	4.0533E8	0.019794	-0.03342
longitude	INPUT	-3.686E7	364212.1	7139	0	-3.832E7	-3.689E7	-3.542E7	0.135576	1.697315
numPhotos	INPUT	19.44096	11.26379	7139	0	0	18	103	1.320214	3.984316
size	INPUT	100.6175	75.4546	7139	0	13	80	2000	5.504909	75.21424
_Yearly_Price	TARGET	18076.75	11816.11	7139	0	4200	14400	240000	4.019985	35.12899

Figure 7: Idealista Interval Variable Summary Statistics before changes

Figure 8 shows the final results after the changes we executed. All the observations were within limits and the skewness controlled.

Variable	Role	Mean	Standard Deviation	Non Missing	Missing	Minimum	Median	Maximum	Skewness	Kurtosis
Parking_Price_Included	INPUT	0.295718	0.456398	6983	0	0	0	1	0.895449	-1.19851
REP_size	INPUT	88.24712	40.49365	6782	201	13	80	220	1.038449	0.687021
Random	INPUT	0.501496	0.28771	6983	0	0.000265	0.501439	0.999875	-0.00567	-1.19543
SUM	INPUT	1.542747	0.838335	6983	0	0	2	3	-0.01464	-0.58439
distance	INPUT	4449.374	2930.298	6983	0	15	3757	14409	0.640463	-0.50939
latitude	INPUT	4.0436E8	314526.9	6983	0	4.0334E8	4.0434E8	4.0533E8	0.029304	-0.03968
longitude	INPUT	-3.686E7	364110.8	6983	0	-3.832E7	-3.69E7	-3.542E7	0.1692	1.675658
numPhotos	INPUT	19.16526	10.98712	6983	0	0	18	103	1.326414	4.270372
_Yearly_Price	TARGET	16927.64	8304.102	6983	0	4200	14400	49200	1.497845	1.961963

Figure 8: Idealista Interval Variable Summary Statistics after changes

3.2.2. Class Variables Statistical Analysis

During our class variables analysis, we identified that the main problem was related to a lack of representations within the classes.

Data Role	Variable Name	Role	Number of Levels	Missing	Mode	Mode Percentage	Mode2	Mode2 Percentage
TRAIN	AC	INPUT	2	0	1	74.46	0	25.54
TRAIN	Amueblado	INPUT	2	0	furnished	54.07	furnishedKitchen	45.93
TRAIN	Has_Parking	INPUT	2	0	0	63.96	1	36.04
TRAIN	Parking	INPUT	3	0	No Parking	63.96	Yes, Price included	30.49
TRAIN	Piscina	INPUT	2	0	0	66.56	1	33.44
TRAIN	Terraza	INPUT	2	0	0	52.37	1	47.63
TRAIN	bathrooms	INPUT	8	0	1	55.57	2	32.46
TRAIN	district	INPUT	21	0	Centro	13.95	Salamanca	11.91
TRAIN	exterior	INPUT	2	0	1	84.94	0	15.06
TRAIN	floor	INPUT	27	173	1	16.68	2	16.26
TRAIN	hasLift	INPUT	3	187	1	86.97	0	10.41
TRAIN	hasPlan	INPUT	2	0	0	80.10	1	19.90
TRAIN	hasVideo	INPUT	2	0	0	83.36	1	16.64
TRAIN	neighborhood	INPUT	132	0	Lavapiés-Embajadores	3.19	Chueca-Justicia	3.08
TRAIN	propertyType	INPUT	5	0	flat	81.93	penthouse	6.64
TRAIN	rooms	INPUT	9	0	2	31.26	1	28.10
TRAIN	showAddress	INPUT	2	0	0	74.84	1	25.16

Figure 9: Idealista Class Variable Summary Statistics before changes

As we can see in Figure 9, *floor* and *haslift* had some missings, and after some investigation, we came to the conclusions that all *floor* missing were related to typology *chalet*. Almost the same houses were also missing for *haslift*. In this case, we imputed the missing value with *chalet*. *Floor*, *Bathrooms*, *District* and *Rooms* presented many levels which were not enough represented (more than 1%). Therefore, we had to regroup them. For *floor*, we limited to *9+* the maximum of floors that were above 9, and grouped all the observations below the ground floor to *ss*. The number of *bathrooms* we replaced those above 3 to *3+* and the number of *rooms* we replaced those above 4 to *4+.* Below we can check a summary of the modifications we executed with the *Replacement Node* (Figure 10).

Variable	Formatted Value	Type	Character Unformatted Value	Numeric Value	Replacement Value	Label
bathrooms	3	N		3	+3	bathrooms
bathrooms	4	N		4	+3	bathrooms
bathrooms	5	N		5	+3	bathrooms
district	Usera	C	Usera	.	Puente de Vallecas	district
district	Barajas	C	Barajas	.	San Blas	district
district	Moratalaz	C	Moratalaz	.	Villa de Vallecas	district
district	Villaverde	C	Villaverde	.	Villa de Vallecas	district
district	Vicálvaro	C	Vicálvaro	.	Villa de Vallecas	district
floor		C		.	chalet	floor
floor	9	C	9	.	+9	floor
floor	en	C	en	.	ss	floor
floor	11	C	11	.	+9	floor
floor	10	C	10	.	+9	floor
floor	12	C	12	.	+9	floor
floor	17	C	17	.	+9	floor
floor	st	C	st	.	ss	floor
floor	14	C	14	.	+9	floor
floor	13	C	13	.	+9	floor
floor	15	C	15	.	+9	floor
floor	16	C	16	.	+9	floor
floor	-1	C	-1	.	ss	floor
floor	18	C	18	.	+9	floor
floor	20	C	20	.	+9	floor
floor	-2	C	-2	.	ss	floor
floor	19	C	19	.	+9	floor
rooms	4	N		4	+4	rooms
rooms	5	N		5	+4	rooms
rooms	6	N		6	+4	rooms
rooms	7	N		7	+4	rooms

Figure 10: Idealista Replacement Values for Class Variable

In Figure 11, we can observe the class variable summary after these changes, without missing values and with the pertinent changes done. We tried to reduce the value of the mode percentage of the relevant variables to have a more homogenous level distribution.

Data Role	Variable Name	Role	Number of Levels	Missing	Mode	Mode Percentage	Mode2	Mode2 Percentage
TRAIN	AC	INPUT	2	0	1	74.05	0	25.95
TRAIN	Amueblado	INPUT	2	0	furnished	54.92	furnishedKitchen	45.08
TRAIN	Has_Parking	INPUT	2	0	0	64.87	1	35.13
TRAIN	IMP_hasLift	INPUT	3	0	1	87.53	0	10.64
TRAIN	Parking	INPUT	3	0	No Parking	64.87	Yes, Price included	29.57
TRAIN	Piscina	INPUT	2	0	0	66.91	1	33.09
TRAIN	REP_G_neighborhood	INPUT	8	0	6	15.15	2	14.09
TRAIN	REP_bathrooms	INPUT	4	0	1	56.81	2	33.12
TRAIN	REP_floor	INPUT	12	0	1	16.88	2	16.31
TRAIN	REP_rooms	INPUT	5	0	2	31.88	1	28.73
TRAIN	Terraza	INPUT	2	0	0	52.87	1	47.13
TRAIN	exterior	INPUT	2	0	1	85.49	0	14.51
TRAIN	propertyType	INPUT	5	0	flat	82.57	penthouse	6.70

Figure 11: Idealista Class Variable Summary Statistics after changes

District and *neighborhood* are collinear variables. Therefore we could not keep both. Since *neighborhood* has higher relative importance than *district*, as we can see in Figure 12, we decided to keep it and group it into smaller groups. We used the *Variable Selection Node*, which groups the class levels accordingly with the relationship with the target variable (higher R^2). We further justify this decision in the section "3.2.4. Variables Selection". The table with the *neighborhood* and grouped level relation is in Appendix C.

3.2.3. Variables Importance and Correlation

Analyzing Variable Worth, we can see that *size* is the most important variable (Figure 12) and it has as well the highest correlation with the target variable (Figure 13). That was already expected, since the basis for rent price is usually calculated on average square meter price in the neighborhood. Then we see the number of *bathrooms* and *rooms*, which are also related to the previous explanation. The property's location (*neighborhood)* is also crucial. These four variables are the most relevant aspects of impact on the rental price and they are according to common sense. Notwithstanding, it is also interesting to see that the random variable is the least important, which reinforces our initial theory of multiple factors affecting the pricing. It also is curious to observe the strong influence of advertising related variables, such as the number of photos and video on the rental price, and also what seems to be a high correlation between the number of photos on the ad, and the rental price (Figure 13). As we can see in Figure 12, after *neighborhood* the importance of the remaining variables is almost flat.

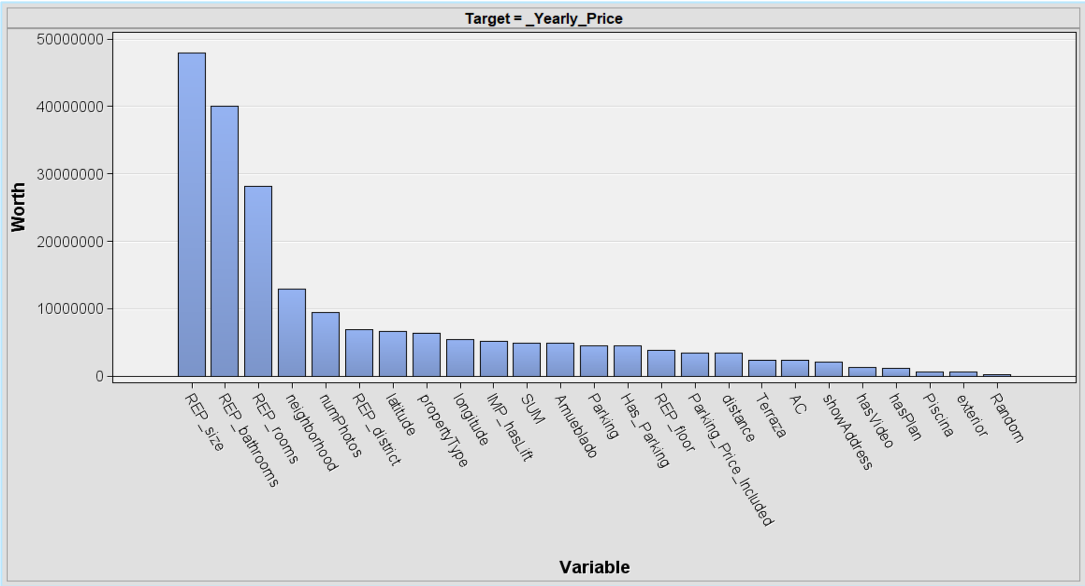


Figure 12: Idealista Variables Worth

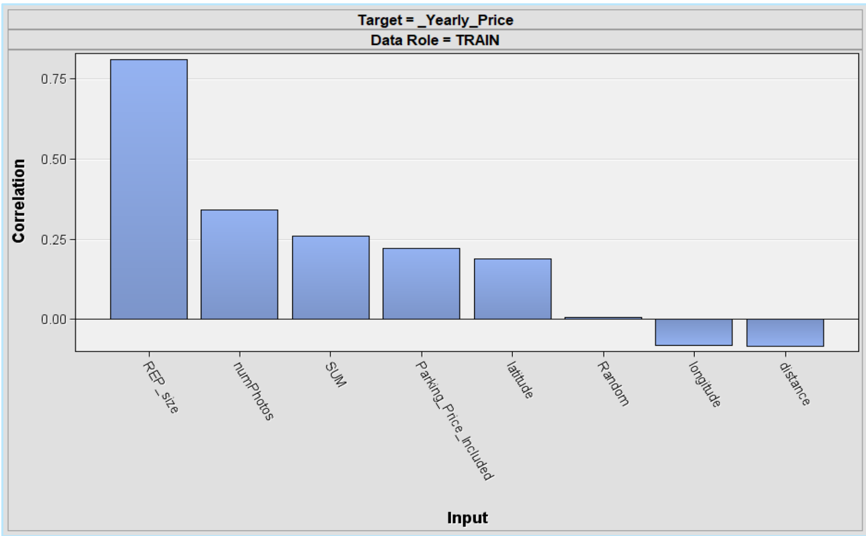


Figure 13: Idealista Variables Correlation

3.3.4. Variables Selection

The variables selection were a critical step in the development of our models because it defined which parameters we would take to the modeling phase and later on the elements the clients could filter or personalize when using our application.

In order to implement an effective variables selection, we built six different models in Miner with eight different transformation and variables selection options (in total 42 models), as we can see in the in Figure 14.

With this strategy, we sought the models to reveal which transformations suited them better, instead of predefining this beforehand. Thus we ensured the selected variables fitted well to the data maximizing the prediction capacity of the future models. These models were only of assistance to help us defining which variables are better when modeling. Therefore, we were not interested in their final results.

0 - Pure / control path, without any modification and all variables;
1 - Group levels for District / Neighborhood;

The following models consider with District / Neighborhood grouped:

2 - Transform Variables;
3 -Transform Variables + Variables Selection;
4 – Variable Selection;
5 – Variable Clustering;
6 – Decision Tree;
7 – Transform Variables + Decision Tree.

The *Transform Variables Node* consists of performing some statistical transformation in the observations so that the prediction models express its true relationship with the target variable. In this case, since the target variable is continuous we applied the method of maximum correlation that maximizes the linear correlation coefficient with the target variable.

The *Variables Selection Node* allows selecting the most significant variables based on their R^2. Those variables that do not reach a minimum value of this statistic are rejected and not used in the modeling phase.

The *Variable Clustering* also allows a variable selection but, instead of taking into consideration the relationship between the target variable, as in the previous case, it uses the relationship between input variables to create a hierarchical cluster. We defined correlation as the clustering source and after the node created the clusters, we evaluated which variables would be part of it based on their R^2.

The *Decision Tree* is a prediction model, but it can also be used as in input to another model. We set the leaf role as input and variance as the splitting criterion, with 0.2 of Significance Level and 200 leaf size.

In Appendix D, we have the results of the nodes with the most relevant impact on the models.

We decided not to take *hasplan*, *hasvideo*, *numphotos*, *showaddress* to the modeling phase since these variables only affect the ads and not the rental price.

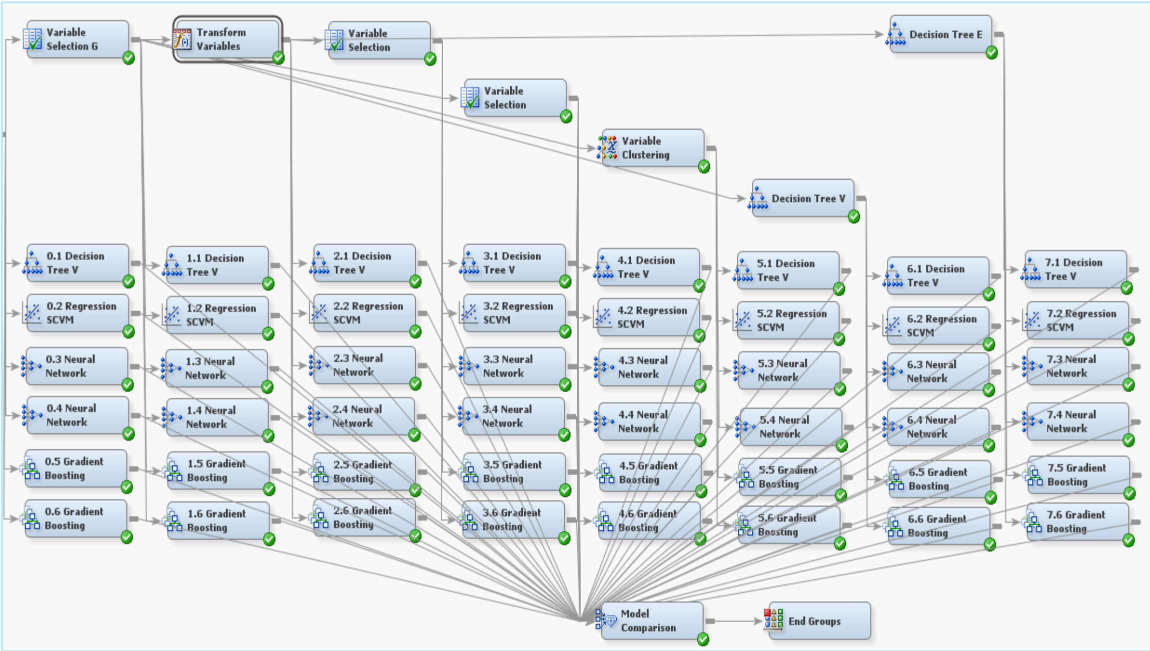


Figure 14: Idealista Miner Models

After running the *Model Comparison Node*, we observed that the best models for our dataset were the gradient boosting coming from the pure (branch 0) or with the grouping for *neighborhood* (branch 1) (Figure 15).

Afterward, we selected the ten better gradient boosting models from the different branches (highlighted in green in Figure 15) and run the Repeated Training-Test (10 repetitions) to ensure we select the best models and, thus the best variable selection. The best model seems to be the one coming from model 1.6, which has no transformations, besides grouping the neighborhood (Figure 16).

We analyzed and compared the variables selection of the four better models, as we can see in Figure 17. They all select almost the same variables and with similar importance ratios.

We did not see any clear drop on the importance of variables to define a cut point (Figure 17). Therefore, we decided to keep the variables selected by model 1.6, but excluding *G_district*, *parking_price_included*, and *has_parking* because these variables are already being explained in the with *G_neighborhood* and *parking*, respectively.

Model Node	Model Description	Test: Root Average Squared Error ▲
Boost14	0.6 Gradient Boosting	2839.806
Boost10	1.6 Gradient Boosting	2881.689
Boost24	2.6 Gradient Boosting	2881.689
Boost13	0.5 Gradient Boosting	2897.09
Neural7	1.3 Neural Network	2929.271
Neural8	0.4 Neural Network	2931.807
Neural9	0.3 Neural Network	2942.17
Boost23	2.5 Gradient Boosting	2953.101
Boost9	1.5 Gradient Boosting	2953.101
Boost26	3.6 Gradient Boosting	2957.473
Neural3	2.4 Neural Network	2985.631
Boost15	4.6 Gradient Boosting	2989.538
Neural22	3.4 Neural Network	3008.133
Neural5	2.3 Neural Network	3016.581
Neural6	1.4 Neural Network	3016.762
Neural4	3.3 Neural Network	3041.618
Boost25	3.5 Gradient Boosting	3047.752
Boost4	4.5 Gradient Boosting	3084.377
Neural21	4.3 Neural Network	3090.978
Neural20	4.4 Neural Network	3092.355
Neural12	6.3 Neural Network	3171.932
Neural11	7.3 Neural Network	3193.418
Boost11	7.6 Gradient Boosting	3205.919
Boost8	6.6 Gradient Boosting	3205.919
Neural10	6.4 Neural Network	3228.384
Neural2	7.4 Neural Network	3258.27
Boost3	7.5 Gradient Boosting	3263.951
Boost7	6.5 Gradient Boosting	3263.951
Reg8	0.2 Regression SCVM	3401.905
Reg	2.2 Regression SCVM	3458.556
Boost6	5.6 Gradient Boosting	3488.006
Reg3	3.2 Regression SCVM	3504.908
Reg7	7.2 Regression SCVM	3525.496
Boost5	5.5 Gradient Boosting	3580.984
Reg2	1.2 Regression SCVM	3622.976
Reg6	6.2 Regression SCVM	3637.044
Reg4	4.2 Regression SCVM	3642.007
Tree2	0.1 Decision Tree V	3672.093
Tree10	7.1 Decision Tree V	3706.242
Tree9	6.1 Decision Tree V	3706.242
Neural13	5.4 Neural Network	3759.753
Neural19	5.3 Neural Network	3767.288
Tree	2.1 Decision Tree V	3862.842
Tree4	1.1 Decision Tree V	3862.842
Tree6	3.1 Decision Tree V	3873.631
Tree7	4.1 Decision Tree V	3873.631
Reg5	5.2 Regression SCVM	4309.776

Figure 15: Idealista Model Comparison Results

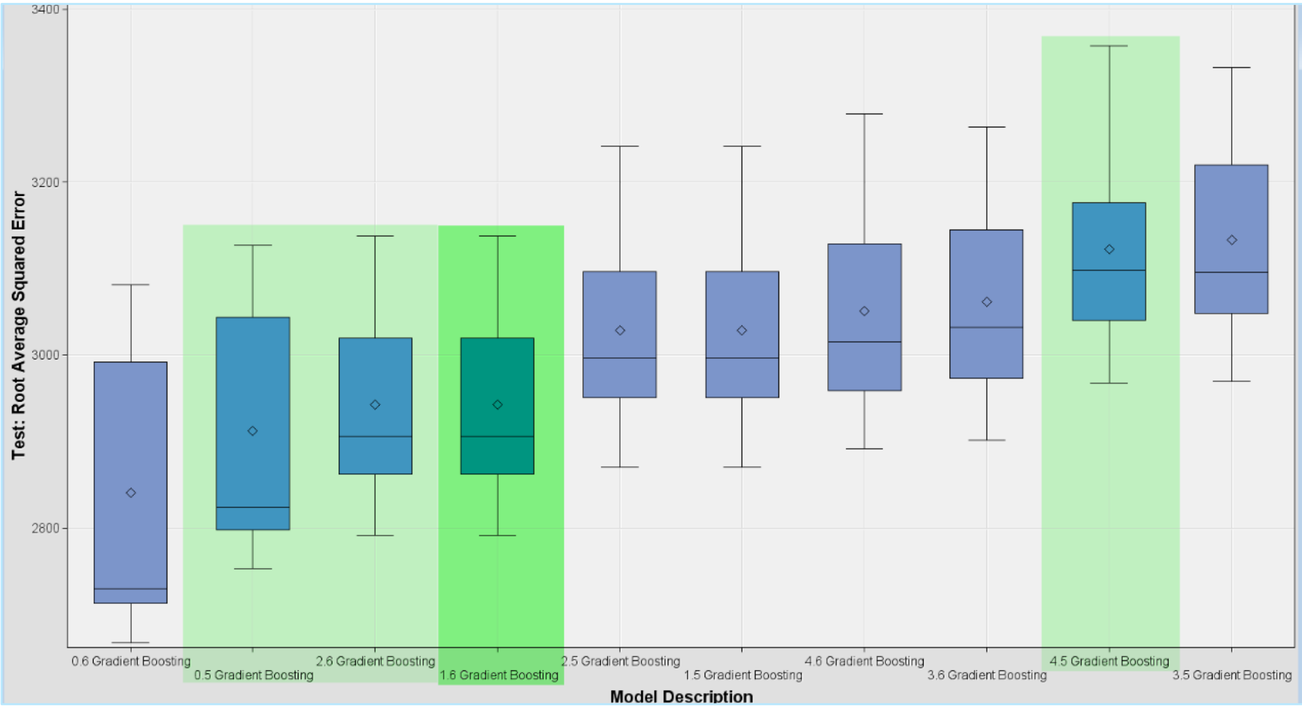


Figure 16: Idealista Box-Plot for Repeated Training-Test

In Figure 17, we can see in green the variables we took to the modeling phase in R, and in red the variables we excluded.

IDEALISTA					
Variables	VI 1.6	VI 2.6	VI 0.5	VI 4.5	Mean
size	100%	100%	100%	100%	100%
bathrooms	47%	47%	46%	46%	47%
neighborhood	30%	30%	41%	0%	25%
distance	29%	29%	12%	36%	27%
rooms	16%	16%	18%	13%	16%
latitude	15%	15%	6%	21%	14%
longitude	12%	12%	2%	14%	10%
district	11%	11%	21%	0%	11%
floor	11%	11%	10%	7%	10%
AC	6%	6%	6%	6%	6%
Parking_Price_Included	5%	5%	3%	0%	3%
Parking	5%	5%	3%	6%	5%
Has_Parking	5%	5%	3%	6%	5%
SUM	5%	5%	4%	5%	5%
IMP_hasLift	4%	4%	4%	6%	5%
Piscina	4%	4%	2%	0%	3%
Amueblado	3%	3%	3%	0%	2%
Terraza	3%	3%	2%	0%	2%
exterior	0%	0%	2%	2%	1%
propertyType	-	-	-	0,0	0,0

Figure 17: Idealista Variable Selection Analysis

3.3. Modeling in R

Once we had the 15 variables selected and transformed to better fit to the upcoming models, we started the modeling phase for the Idealista data for rental properties using the R Studio. By training several models, we sought to find the model with higher R^2 and lower RMSE to be the model we would use on Rentalbility platform.

3.3.1. Neural Network

For the Neural Network models, we used both AvNNet and NNet function from the Caret library, but we only kept the kept one model, the one with better results.

We started tuning our Neural Network with the NNET function of caret, with five repetitions. For the architecture selection, we used 8,10,12,15,18,20,25 units per hidden layer (since we have 6983 observations and 15 variables to obtain 20 obs/parameters we would need 20 units: $h\,(15\,+\,1) + h\,+\,1 = 6980/20$, thus we kept a range of 45 and 13 obs/parameters). We used weight decay of 0.001, 0.01, 0.1.

```
nnetgrid <-  expand.grid(size=c(8,10,12,15,18,20,25),decay=c(0.01,0.1,0.001))

rednnet<- train(Yearly_Price~.,data=idealistabis,
                method="nnet",linout = TRUE,maxit=100,trControl=control,tuneGrid=nnetgrid)
```

With this function, after the model run all combinations between size and decay, we obtained the result shown in Figure 18, where we can see the best model had 15 hidden layers, a learning rate of 0.1, R^2 of 0.552.

```
> rednnet
Neural Network

6983 samples
  16 predictor

No pre-processing
Resampling: Cross-Validated (4 fold, repeated 5 times)
Summary of sample sizes: 5237, 5237, 5238, 5237, 5238, 5237, ...
Resampling results across tuning parameters:

  size  decay  RMSE      Rsquared   MAE     
   8    0.001  8178.471  0.1293112  6134.521
   8    0.010  7370.647  0.4759739  5537.012
   8    0.100  6440.458  0.5414265  4798.096
  10    0.001  8099.301  0.2859282  6098.920
  10    0.010  7299.789  0.4446132  5466.856
  10    0.100  6646.093  0.5910395  4970.530
  12    0.001  7862.366  0.4357961  5919.810
  12    0.010  7479.026  0.5316767  5626.451
  12    0.100  6622.876  0.5660729  4917.405
  15    0.001  7024.789  0.5459183  5263.092
  15    0.010  6986.960  0.5209294  5217.637
  15    0.100  6075.136  0.5521130  4526.785
  18    0.001  7083.860  0.4947689  5304.613
  18    0.010  6441.546  0.4961324  4786.013
  18    0.100  6306.371  0.5672625  4709.899
  20    0.001  6880.136  0.4693379  5138.019
  20    0.010  6433.525  0.4673731  4787.255
  20    0.100  7041.721  0.4784871  5232.883
  25    0.001  6598.454  0.4855105  4906.860
  25    0.010  6803.620  0.4950351  5101.356
  25    0.100  6524.274  0.5107054  4866.953

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were size = 15 and decay = 0.1.
```

Figure 18: Idealista NNet results

Then we proceeded with the training of avNNet function. We also executed cross-validation with different seeds and randomization. This package uses the same algorithm and activation function. For the configuration of the neural networks models, we reduced the size possibilities to 8,10,12,15,18 and tested with the decay of 0.001, 0.01, 0.1.

```
avnnetgrid <-expand.grid(size=c(8,10,12,15,18),decay=c(0.01,0.1,0.001),bag=FALSE)

redavnnet<- train(Yearly_Price~.,data=idealistabis,
                   method="avNNet",linout = TRUE,maxit=100,trControl=control,repeats=5,tuneGrid=avnnetgrid)
```

Not surprisingly, with this function, we had different results. Our best model had 10 10 units, a decay of 0.1, and R^2 equals to 0,703.

```
> redavnnet
Model Averaged Neural Network

6983 samples
  16 predictor

No pre-processing
Resampling: Cross-Validated (4 fold, repeated 5 times)
Summary of sample sizes: 5238, 5237, 5238, 5236, 5236, 5238, ...
Resampling results across tuning parameters:

  size  decay  RMSE      Rsquared   MAE
   8    0.001  7438.161  0.5413641  5564.243
   8    0.010  7231.585  0.6466164  5394.499
   8    0.100  5801.626  0.6941865  4234.542
  10    0.001  7319.656  0.5177971  5455.200
  10    0.010  6974.716  0.5314101  5210.174
  10    0.100  5644.382  0.7035346  4139.805
  12    0.001  6918.192  0.5684549  5150.043
  12    0.010  6492.307  0.6591512  4817.082
  12    0.100  6287.310  0.6529577  4651.384
  15    0.001  6397.485  0.6577086  4746.080
  15    0.010  6321.948  0.6499139  4678.728
  15    0.100  5957.116  0.6659754  4406.763
  18    0.001  6467.483  0.6462714  4764.372
  18    0.010  6086.913  0.6742686  4464.885
  18    0.100  6245.670  0.6151305  4618.915

Tuning parameter 'bag' was held constant at a value of FALSE
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were size = 10, decay = 0.1 and bag = FALSE.
```

Figure 19: Idealista avNNet results

Since the Avnnet had better results, we selected this model for the later competition with other models.

3.3.2. Random Forest and Bagging

In search of the best Random Forest, we started by finding the best number of variables for each tree (mtry). Therefore we used 3,5,8,10,12 and 15 (which is the bagging model). On this first try, we did not sample the observations. We set 1000 trees and a minimum of 20 obs per node. We also used cross-validation with 5 repetitions.

```
rfgrid<-expand.grid(mtry=c(3,5,8,10,12,15))

rf<- train(Yearly_Price~.,data=idealistabis,
           method="rf",trControl=control,tuneGrid=rfgrid,
           linout = TRUE,ntree=1000, nodesize=20,replace=TRUE,
           importance=TRUE)
```

With this tuning, the optimal selected model (Figure 20), was with 10 variables and R^2 of 0,90.

```
> rf
Random Forest

6983 samples
  15 predictor

No pre-processing
Resampling: Cross-Validated (4 fold, repeated 5 times)
Summary of sample sizes: 5237, 5237, 5237, 5238, 5237, 5237, ...
Resampling results across tuning parameters:

  mtry  RMSE      Rsquared   MAE
   3    2854.860  0.8903696  1860.980
   5    2674.157  0.8992436  1692.419
   8    2629.370  0.9010053  1636.459
  10    2629.190  0.9006143  1627.319
  12    2636.373  0.8998613  1625.312
  15    2650.680  0.8985988  1628.478
  18    2666.924  0.8972438  1633.579
  20    2679.658  0.8962027  1639.845

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 10.
```

Figure 20: Idealista RF results

We rerun the tuning with the 10 variables, but now sampling the observations, at 60% of the total observations.

```
rfgrid1<-expand.grid(mtry=c(10))

rf1<- train(Yearly_Price~.,data=idealistabis,
           method="rf",trControl=control,tuneGrid=rfgrid1,
           linout = TRUE,ntree=1000, sampsize=5000, nodesize=20,replace=TRUE,
           importance=TRUE)
```

The sampling increased the RSME (Figure 21). Therefore, we keep it without sampling.

```
> rf1$results
  mtry     RMSE  Rsquared      MAE  RMSESD RsquaredSD    MAESD
1   10 2630.386 0.9004071 1634.936 101.517 0.00800231 39.22001
```

Figure 21: Idealista RF1 results

We evaluated the need for early stopping. As we can see in Figure 22, the Out of Bag Error (OBB) got flat before 500 iterations. Hence, we reduced the iterations to understand what worked better with the data. Between 300 and 400 iterations should be enough, as we can observe in green in Figure 22.

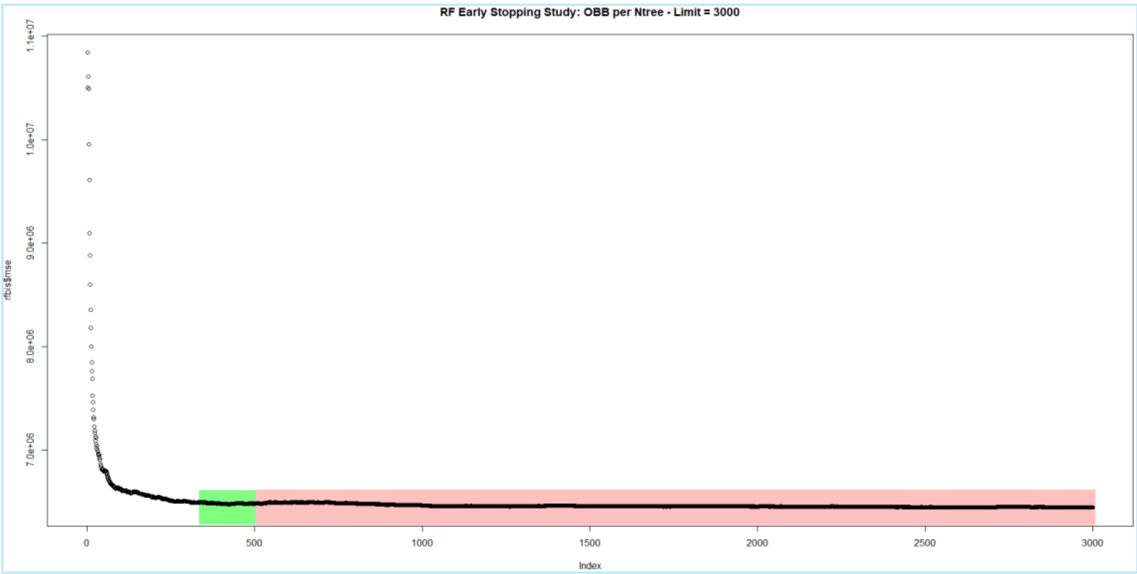


Figure 22: Idealista RF1 Early Stopping Study

We run again the tunning, but changing the set up of the Random Forest. We kept 10 variables without sampling observations but testing both ntree= 400 and 300. With 300 trees (rf2) we got as results R^2=0.9016 and RSME=2615.7. With ntree=400 (rf3), we obtained an R^2 of 0.9003 and RSME of 2629.89. With this last test, we found the best Random Forest, the rf2 model. The final results are in Figure 23.

```
> rf2$results
  mtry     RMSE  Rsquared      MAE   RMSESD  RsquaredSD    MAESD
1   10 2615.708 0.9016487 1621.735 101.0806 0.005792454 47.55379
```

Figure 23: Idealista RF2 results

In Figure 24, we can see the variables importance for RF2. As we can see, *size*, *bathrooms*, *neighborhood*, *rooms*, *distance*, *latitude,* and *longitude* are the most important variables on this model.

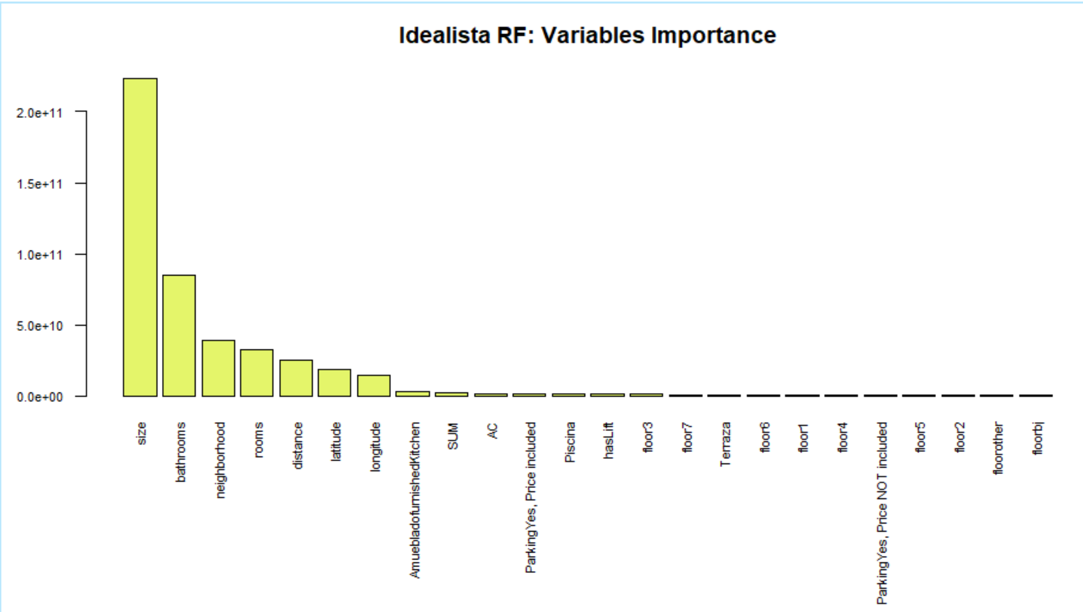


Figure 24: Idealista RF2 Variable Importance

3.3.3. Gradient Boosting

For tuning the Gradient Boosting, we worked with two approaches, one aggressive and other more conservative, to avoid overfitting. Therefore, we have a big range of shrinkage values, from very small 0.001 until 0.2. In the minimum number of observations per parameters, we set it to only a few, from 5 and up to 30. We set the number of trees from 100 until 5000.

```
gbmgrid<-expand.grid(shrinkage=c(0.1,0.05,0.03,0.01,0.001,0.2),
                     n.minobsinnode=c(5,10,20,30),
                     n.trees=c(100,300,500,1000,2000,5000),
                     interaction.depth=c(2))

gbm<- train(Yearly_Price~.,data=idealistabis,
            method="gbm",trControl=control,tuneGrid=gbmgrid,
            distribution="gaussian", bag.fraction=1,verbose=FALSE)
```

After running 144 possible models, we can see some of them in Figure 25, R recommended the model with 5000 trees, shrinkage of 0.1 and 30 observations per tree. Due to the high shrinkage parameter, we can say that this model is more aggressive, but at the same time, it is balanced by the high number of trees and the number of nodes.

```
> gbm
Stochastic Gradient Boosting

6983 samples
  15 predictor

No pre-processing
Resampling: Cross-Validated (4 fold, repeated 5 times)
Summary of sample sizes: 5237, 5238, 5237, 5237, 5236, 5238, ...
Resampling results across tuning parameters:

  shrinkage  n.minobsinnode  n.trees  RMSE      Rsquared   MAE
  0.001       5               100     7816.945  0.6410092  5864.114
  0.001       5               300     7007.920  0.6748544  5243.385
  0.001       5               500     6372.057  0.6915925  4763.439
  0.001       5              1000     5299.040  0.7319576  3908.358
  0.001       5              2000     4238.062  0.7871146  3056.035
  0.001       5              5000     3424.576  0.8366055  2360.364
  0.001      10               100     7816.945  0.6410092  5864.114
  0.001      10               300     7007.920  0.6748544  5243.385
  0.001      10               500     6372.057  0.6915925  4763.439
  0.001      10              1000     5299.040  0.7319576  3908.358
  0.001      10              2000     4238.062  0.7871146  3056.035
  0.001      10              5000     3424.576  0.8366055  2360.364
  0.001      20               100     7816.945  0.6410092  5864.114
  0.001      20               300     7007.920  0.6748544  5243.385
  0.001      20               500     6372.057  0.6915925  4763.439
  0.001      20              1000     5299.040  0.7319576  3908.358
  0.001      20              2000     4238.062  0.7871146  3056.035
  0.001      20              5000     3424.578  0.8366045  2360.363
  0.001      30               100     7816.945  0.6410092  5864.114
  0.001      30               300     7007.920  0.6748544  5243.385
  0.001      30               500     6372.057  0.6915925  4763.439
  0.001      30              1000     5299.040  0.7319576  3908.358
  0.001      30              2000     4238.062  0.7871146  3056.035
  0.001      30              5000     3424.402  0.8366103  2360.129
  0.010       5               100     5292.736  0.7317970  3903.241
  0.010       5               300     3779.485  0.8137887  2668.892
  0.010       5               500     3423.075  0.8366426  2358.874
  0.010       5              1000     3199.698  0.8524762  2187.660
  0.010       5              2000     3095.186  0.8614162  2117.006
  0.010       5              5000     2972.569  0.8720998  2023.340
  0.010      10               100     5292.736  0.7317970  3903.241
  0.010      10               300     3779.485  0.8137887  2668.892
  0.010      10               500     3423.075  0.8366426  2358.874
  0.010      10              1000     3199.321  0.8525072  2187.416
  0.010      10              2000     3092.573  0.8616426  2115.720
  0.010      10              5000     2972.065  0.8721365  2023.559
  0.010      20               100     5292.736  0.7317970  3903.241
  0.010      20               300     3779.485  0.8137887  2668.892
  0.010      20               500     3423.075  0.8366426  2358.874
  0.010      20              1000     3197.549  0.8526722  2186.626
  0.010      20              2000     3088.537  0.8620017  2112.695
  0.010      20              5000     2969.510  0.8723299  2021.922
  0.010      30               100     5292.736  0.7317970  3903.241
  0.010      30               300     3779.485  0.8137887  2668.892
  0.010      30               500     3422.801  0.8366613  2358.520
  0.050      30              1000     2967.436  0.8725068  2018.363
  0.050      30              2000     2896.109  0.8785051  1959.934
  0.050      30              5000     2839.802  0.8831400  1909.476
  0.100       5               100     3202.339  0.8519601  2187.231
  0.100       5               300     3035.657  0.8666504  2068.976
  0.100       5               500     2964.340  0.8727884  2015.654
  0.100       5              1000     2890.568  0.8789867  1956.797
  0.100       5              2000     2849.238  0.8823897  1916.366
  0.100       5              5000     2840.004  0.8832227  1902.343
  0.100      10               100     3201.447  0.8520548  2186.851
  0.100      10               300     3033.096  0.8668599  2068.539
  0.100      10               500     2965.093  0.8727262  2016.369
  0.100      10              1000     2889.743  0.8790720  1955.380
  0.100      10              2000     2840.056  0.8831584  1911.889
  0.100      10              5000     2823.644  0.8845490  1888.765
  0.100      20               100     3199.738  0.8522270  2186.075
  0.100      20               300     3029.919  0.8671260  2065.554
  0.100      20               500     2962.916  0.8728899  2014.724
  0.100      20              1000     2892.119  0.8788350  1956.127
  0.100      20              2000     2849.529  0.8823321  1916.048
  0.100      20              5000     2826.199  0.8842996  1884.876
  0.100      30               100     3200.815  0.8521230  2187.374
  0.100      30               300     3030.036  0.8671175  2064.941
  0.100      30               500     2963.899  0.8728009  2014.347
  0.100      30              1000     2894.365  0.8786520  1958.777
  0.100      30              2000     2849.459  0.8823535  1919.935
  0.100      30              5000     2816.100  0.8851162  1884.352
  0.200       5               100     3100.623  0.8608723  2115.219
  0.200       5               300     2950.628  0.8739304  2002.054
  0.200       5               500     2901.300  0.8780948  1961.090
  0.200       5              1000     2865.125  0.8810868  1925.704
  0.200       5              2000     2851.937  0.8822480  1909.186
  0.200       5              5000     2909.072  0.8777800  1959.126
  0.200      10               100     3100.104  0.8608892  2114.862
  0.200      10               300     2951.691  0.8738375  2003.380
  0.200      10               500     2899.296  0.8782551  1961.439
  0.200      10              1000     2850.652  0.8822881  1917.047
  0.200      10              2000     2834.938  0.8836265  1899.221
  0.200      10              5000     2873.704  0.8806603  1931.855
  0.200      20               100     3092.271  0.8615917  2109.248
  0.200      20               300     2947.138  0.8742050  1999.723
  0.200      20               500     2901.609  0.8780242  1961.670
  0.200      20              1000     2856.601  0.8817442  1920.091
  0.200      20              2000     2837.404  0.8833594  1898.470
  0.200      20              5000     2858.114  0.8818223  1912.167
  0.200      30               100     3094.535  0.8614008  2109.485
  0.200      30               300     2949.542  0.8739985  1999.841
  0.200      30               500     2903.347  0.8778792  1963.931
  0.200      30              1000     2855.278  0.8818799  1922.637
  0.200      30              2000     2829.091  0.8840613  1896.982
  0.200      30              5000     2834.135  0.8837645  1897.280

Tuning parameter 'interaction.depth' was held constant at a value of 2
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were n.trees = 5000, interaction.depth =
 2, shrinkage = 0.1 and n.minobsinnode = 30.
```

Figure 25: Idealista GBM results

We also studied the possibility of sampling the observations (GBMR model) by keeping all the parameters constant and changing the bag fraction to 0.6 (4189 observations). The RSME decreased a little bit, as in Figure 26, so we decided to keep the sampling.

```
> gbmr$results
  shrinkage n.minobsinnode n.trees interaction.depth     RMSE  Rsquared      MAE   RMSESD RsquaredSD     MAESD
1       0.1             30    5000                 2 2811.426 0.8854568 1879.755 101.8203 0.00828378 45.46015
```

Figure 26: Idealista GBMr results

Due to the high amount of trees, we needed to check if we could stop earlier or if it was needed to add more trees. We rerun the model, with 1.000, 5.000, 8.000, 10.000 trees. As we can see in the chart below (Figure 27) from 1.000 to 5.000 there is a big drop on the OBB. After 5.000 the OBB decreases at a lower rate, but since this drop is not significant, and the more iterations more the complex the model is, we decided to keep 5000 trees.

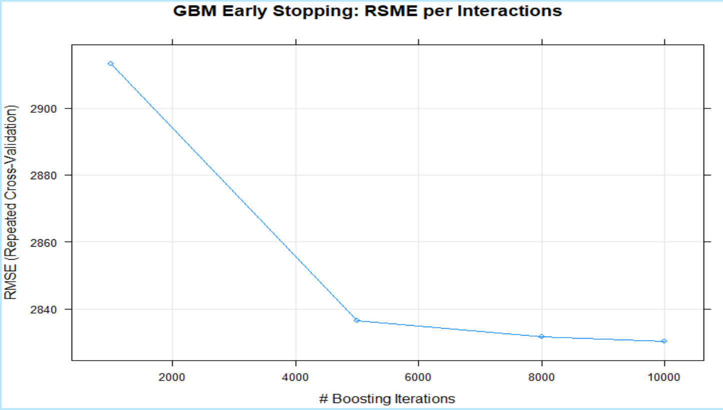


Figure 27: Idealista GBMr Early Stopping

Finally, we examined the variable importance of our model, as we can see in Figure 28, again *size*, *distance*, *neighborhood*, *bathroom* play an important role. As a matter of fact, the top 5 variables are the same from the random forest, but with different rates.

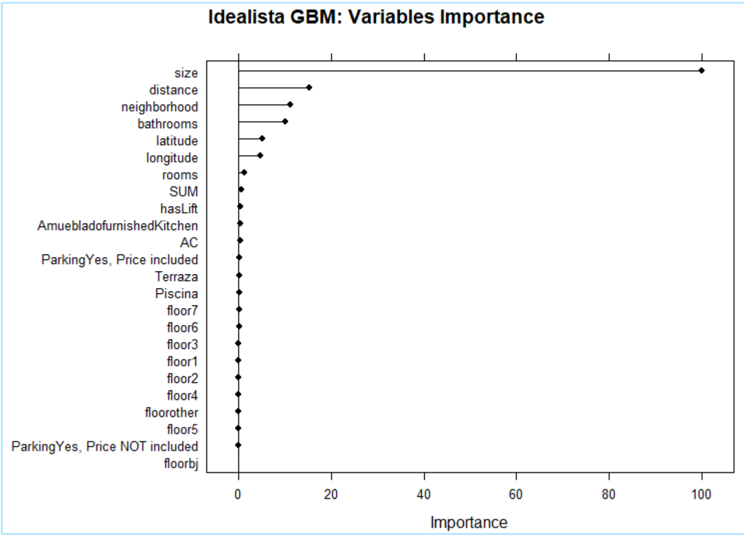


Figure 28: Idealista GBMr variable importance

3.3.4. Extreme Gradient Boosting

Following the Gradient boosting approaches, we built a wide grid for the XGboost model. We set a minimum number of instances in each child tree of 20, the shrinkage between 0.01 and 0.3, the number of iterations from 100 until 5000. We decided not to train the coefficient of regularization gamma (0 = without penalization).

```
xgbmgrid <-expand.grid( min_child_weight=20,
                        eta=c(0.1,0.05,0.03,0.01,0.001,0.2,0.3),
                        nrounds=c(100,300,500,1000,2000,4000,5000),
                        max_depth=6,
                        gamma=0,
                        colsample_bytree=1,
                        subsample=1)

xgbm<- train(Yearly_Price~.,data=idealistabis,
             method="xgbTree",trControl=control,
             tuneGrid=xgbmgrid,objective = "reg:linear",verbose=FALSE,
             alpha=1,lambda=0)
```

After running 49 model possibilities, the final values selected for the model xgbm were nrounds=2000, max_depth=6, eta=0.03, gamma=0, colsample_bytree=1, min_child_weight=20 and subsample=1, as in Figure 29.

```
> xgbm
eXtreme Gradient Boosting

6983 samples
  15 predictor

No pre-processing
Resampling: Cross-Validated (4 fold, repeated 5 times)
Summary of sample sizes: 5238, 5236, 5238, 5237, 5238, 5237, ...
Resampling results across tuning parameters:

  eta    nrounds  RMSE       Rsquared   MAE
  0.001   100     17119.489  0.8297340  15315.309
  0.001   300     14147.011  0.8397411  12537.730
  0.001   500     11733.704  0.8472906  10263.910
  0.001  1000      7546.744  0.8586970   6244.330
  0.001  2000      3925.631  0.8720671   2705.890
  0.001  4000      2847.081  0.8844806   1835.044
  0.001  5000      2782.529  0.8879752   1795.435
  0.010   100      7518.372  0.8586466   6216.339
  0.010   300      3036.309  0.8797124   1960.483
  0.010   500      2782.462  0.8879567   1794.911
  0.010  1000      2712.694  0.8931510   1740.273
  0.010  2000      2681.705  0.8955617   1707.712
  0.010  4000      2665.751  0.8968179   1687.332
  0.010  5000      2662.188  0.8970935   1682.390
  0.030   100      3028.489  0.8796585   1955.266
  0.030   300      2719.026  0.8926534   1747.577
  0.030   500      2688.839  0.8949975   1718.344
  0.030  1000      2668.387  0.8965767   1693.370
  0.030  2000      2660.699  0.8971928   1679.278
  0.030  4000      2663.226  0.8970609   1678.716
  0.030  5000      2667.473  0.8967564   1681.359
  0.050   100      2782.367  0.8878857   1796.024
  0.050   300      2690.287  0.8948659   1719.568
  0.050   500      2675.078  0.8960361   1702.173
  0.050  1000      2661.558  0.8971067   1682.953
  0.050  2000      2663.533  0.8970198   1680.989
  0.050  4000      2679.874  0.8958485   1692.597
  0.050  5000      2687.806  0.8952671   1698.085
  0.100   100      2719.944  0.8925675   1747.817
  0.100   300      2676.973  0.8959149   1699.609
  0.100   500      2667.007  0.8967243   1687.179
  0.100  1000      2670.951  0.8964917   1686.046
  0.100  2000      2687.137  0.8953293   1696.380
  0.100  4000      2711.731  0.8935146   1715.432
  0.100  5000      2718.203  0.8930341   1720.438
  0.200   100      2703.225  0.8938439   1730.030
  0.200   300      2689.742  0.8949714   1707.445
  0.200   500      2691.568  0.8948783   1706.177
  0.200  1000      2710.141  0.8935364   1719.175
  0.200  2000      2731.064  0.8919942   1734.437
  0.200  4000      2746.330  0.8908456   1745.042
  0.200  5000      2749.123  0.8906322   1746.868
  0.300   100      2723.244  0.8923580   1745.024
  0.300   300      2720.029  0.8927974   1737.071
  0.300   500      2733.626  0.8918145   1745.509
  0.300  1000      2751.723  0.8905128   1755.599
  0.300  2000      2769.137  0.8892143   1767.842
  0.300  4000      2777.217  0.8885992   1773.030
  0.300  5000      2778.371  0.8885099   1773.653

Tuning parameter 'max_depth' was held constant at a value of 6
Tuning parameter 'gamma' was
 1
Tuning parameter 'min_child_weight' was held constant at a value of 20
Tuning parameter
 'subsample' was held constant at a value of 1
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were nrounds = 2000, max_depth = 6, eta = 0.03, gamma =
 0, colsample_bytree = 1, min_child_weight = 20 and subsample = 1.
```

Figure 29: Idealista XGBM Results

As we needed to evaluate the penalization factor gamma, we rerun the model, but we kept all the parameters from our winner model and set up a grid of gammas between 0 and 1. R suggested maintaining gamma as 0. Hence, we keep the previous xgbm model.

```
> xgbmg
eXtreme Gradient Boosting

6983 samples
  15 predictor

No pre-processing
Resampling: Cross-Validated (4 fold, repeated 5 times)
Summary of sample sizes: 5238, 5237, 5236, 5238, 5237, 5237, ...
Resampling results across tuning parameters:

  gamma  RMSE      Rsquared  MAE
  0.000  2631.166  0.899583  1668.106
  0.001  2631.166  0.899583  1668.106
  0.010  2631.166  0.899583  1668.106
  0.100  2631.166  0.899583  1668.106
  1.000  2631.166  0.899583  1668.106

Tuning parameter 'nrounds' was held constant at a value of 2000
Tuning parameter 'max_depth' was
 parameter 'min_child_weight' was held constant at a value of 20
Tuning parameter 'subsample' was
 held constant at a value of 1
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were nrounds = 2000, max_depth = 6, eta = 0.03, gamma =
 0, colsample_bytree = 1, min_child_weight = 20 and subsample = 1.
```

Figure 30: Idealista XGBMg Results

Since we trained a vast number of trees, we did not need to study the early stopping in this case.

Moreover, below in Figure 31, we see the variables importance for the XGBM model. *Size*, *neighborhood*, *bathrooms*, *distance* are again the most important variables.

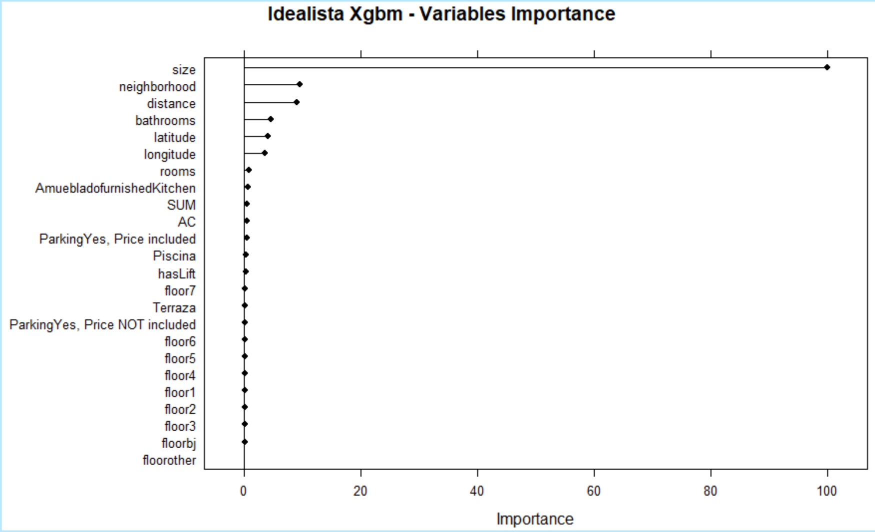


Figure 31: Idealista XGBM variable importance

3.3.5. Support Vector Machine

We trained Linear and Radial Support Vector Machine models.

3.3.5.1 Linear

We tuned the linear SVM model by varying the penalty factor C between 0.01 and 10.

```
SVMgridl<-expand.grid(C=c(0.01,0.05,0.1,0.2,0.5,1,2,5,10))

SVMl<- train(data=idealistabis,Yearly_Price~.,
             method="svmLinear",trControl=control,
             tuneGrid=SVMgridl,verbose=FALSE)
```

The best, in Figure 32 model had C = 0.01 and R^2 = 0.755.

```
> SVMl
Support Vector Machines with Linear Kernel

6983 samples
  15 predictor

No pre-processing
Resampling: Cross-Validated (4 fold)
Summary of sample sizes: 5237, 5237, 5238, 5237
Resampling results across tuning parameters:

  C      RMSE      Rsquared   MAE
   0.01  4179.019  0.7552518  2485.047
   0.05  4214.249  0.7521859  2482.975
   0.10  4219.424  0.7516826  2482.769
   0.20  4220.188  0.7516091  2482.594
   0.50  4221.821  0.7514983  2482.718
   1.00  4222.403  0.7514442  2482.610
   2.00  4221.502  0.7515064  2482.613
   5.00  4222.151  0.7514393  2482.615
  10.00  4222.544  0.7514359  2482.715

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was C = 0.01.
```

Figure 32: Idealista SVML results

3.3.5.2. Radial

For the Radial SVM, we kept the same range of penalty parameters, from 0.01 until 10 and varied the sigma from 0.1 to 5.

```
SVMrgrid<-expand.grid(C=c(0.01,0.05,0.1,0.2,0.5,1,2,5),
                      sigma=c(0.01,0.05,0.1,0.2,0.5,1,2,5))

SVMr<- train(data=idealistabis,Yearly_Price~.,
             method="svmRadial",trControl=control,
             tuneGrid=SVMrgrid,verbose=FALSE)
```

After running the 64 models possibilities, the best model selected presented C=5 and sigma = 0.05. Its R^2 is 0.875, as we can observe in Figure 33.

```
> SVMr
Support Vector Machines with Radial Basis Function Kernel

6983 samples
  15 predictor

No pre-processing
Resampling: Cross-Validated (4 fold)
Summary of sample sizes: 5238, 5237, 5238, 5236
Resampling results across tuning parameters:

  C     sigma  RMSE      Rsquared   MAE
  0.01  0.01   5008.688  0.7706384  3150.944
  0.01  0.05   5565.589  0.7009998  3395.677
  0.01  0.10   6858.823  0.5891638  4247.453
  0.01  0.20   7971.193  0.4429138  5192.419
  0.01  0.50   8428.142  0.2808598  5639.163
  0.01  1.00   8506.059  0.2235511  5720.001
  0.01  2.00   8532.865  0.1862369  5753.589
  0.01  5.00   8544.329  0.1573928  5774.845
  0.05  0.01   3715.926  0.8245629  2421.169
  0.05  0.05   3983.862  0.8028866  2454.854
  0.05  0.10   4918.906  0.7239003  2924.053
  0.05  0.20   6526.789  0.5726335  3921.044
  0.05  0.50   7793.587  0.3422156  4954.517
  0.05  1.00   8077.086  0.2642530  5234.790
  0.05  2.00   8211.371  0.2218454  5366.935
  0.05  5.00   8292.030  0.1900913  5451.903
  0.10  0.01   3492.446  0.8369227  2275.886
  0.10  0.05   3575.340  0.8334759  2229.109
  0.10  0.10   4222.504  0.7810619  2503.099
  0.10  0.20   5644.546  0.6474110  3281.557
  0.10  0.50   7256.664  0.4015811  4422.351
  0.10  1.00   7671.986  0.3067136  4801.784
  0.10  2.00   7869.001  0.2540913  4996.254
  0.10  5.00   7999.722  0.2168234  5135.368
  0.20  0.01   3341.175  0.8455844  2180.025
  0.20  0.05   3277.462  0.8535798  2069.146
  0.20  0.10   3695.745  0.8222523  2206.156
  0.20  0.20   4819.864  0.7180010  2747.348
  0.20  0.50   6556.214  0.4865781  3833.350
  0.20  1.00   7108.108  0.3831659  4279.184
  0.20  2.00   7385.738  0.3253050  4532.416
  0.20  5.00   7582.839  0.2787313  4719.079
  0.50  0.01   3213.688  0.8538953  2101.581
  0.50  0.05   3054.519  0.8682032  1930.634
  0.50  0.10   3248.752  0.8534659  1962.160
  0.50  0.20   3990.276  0.7886397  2284.813
  0.50  0.50   5567.191  0.5984110  3155.294
  0.50  1.00   6221.986  0.4897417  3600.378
  0.50  2.00   6570.797  0.4257834  3878.348
  0.50  5.00   6847.238  0.3727426  4113.835
  1.00  0.01   3143.626  0.8593035  2053.920
  1.00  0.05   2966.735  0.8740845  1865.417
  1.00  0.10   3085.068  0.8646106  1881.311
  1.00  0.20   3618.170  0.8188491  2123.686
  1.00  0.50   4970.017  0.6629022  2859.069
  1.00  1.00   5647.715  0.5593159  3295.500
  1.00  2.00   6024.208  0.4953939  3580.542
  1.00  5.00   6332.363  0.4390312  3830.418
  2.00  0.01   3093.378  0.8632167  2012.657
  2.00  0.05   2935.831  0.8759045  1836.994
  2.00  0.10   3035.920  0.8673062  1861.639
  2.00  0.20   3460.274  0.8303098  2072.067
  2.00  0.50   4674.391  0.6934595  2751.044
  2.00  1.00   5332.531  0.5981768  3185.690
  2.00  2.00   5714.667  0.5373615  3480.164
  2.00  5.00   6037.587  0.4823320  3749.744
  5.00  0.01   3039.084  0.8676923  1964.190
  5.00  0.05   2934.777  0.8755419  1827.347
  5.00  0.10   3073.919  0.8633326  1888.041
  5.00  0.20   3440.013  0.8304778  2068.089
  5.00  0.50   4626.247  0.6959571  2734.007
  5.00  1.00   5275.959  0.6028319  3167.177
  5.00  2.00   5647.532  0.5445454  3450.583
  5.00  5.00   5962.591  0.4918289  3720.405

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were sigma = 0.05 and C = 5.
```

Figure 33: Idealista SVMR results

3.3.6. Models Assessment

With all the seven models tunned, we could finally run the final competition between them with cross-validation and 20 repetitions to know which one model is the best for the Idealista data. As we can in the boxplot (Figure 34), the Xgbm is the best model because it has the lower RSME of 2641.337 and the higher R^2 of 0.898808. Besides the XGBoost, the Random Forest model also performed very well, with RSME of 2668.869 and R^2 of 0.8985962. Both models also have very low variability, which is another positive aspect for both models.

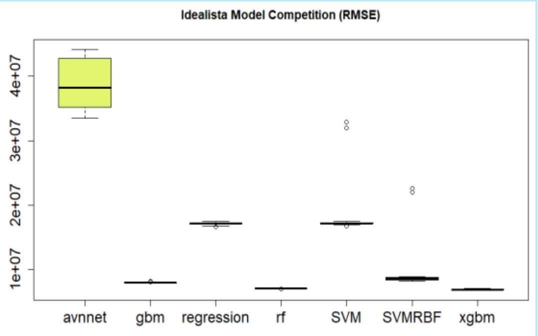


Figure 34: Idealista Model Assesment

3.3.7. Ensemble

With those models with lower RSME, we made ensembled models by calculating the mean predictions of each of them. We had saved all the predictions from each model before and took the mean from this combination, as we can see below. We made the ensemble models by combining the 4 best models in Figure 34 (Xgbm>rf>gbm>SVMRadial) in different groups.

```
unipredi$predi10<-(unipredi$rf+unipredi$xgbm)/2
unipredi$predi11<-(unipredi$rf+unipredi$gbm+unipredi$xgbm)/3
unipredi$predi12<-(unipredi$gbm+unipredi$rf+unipredi$xgbm+unipredi$SVMRBF)/4
```

In Figure 35, the results of the ensemble models are compared with the originals. In all ensemble models, the RSME rate decreased. Therefore, the best model became *Predi12*, which is the combination of Xgbm, rf, gbm, and SVMRBF. It has RSME= 6596163 and R^2= 0.9010966.

	modelo	r2	error
1	gbm	0.8840260	7997348
2	predi10	0.9035712	6649548
3	predi11	0.9010966	6820189
4	predi12	0.9043454	6596163
5	rf	0.8965789	7131721
6	SVMRBF	0.8553746	9973093
7	xgbm	0.8986906	6986107

Figure 35: Idealista Final Model Assesment (R^2 and RSME)

With the boxplot (Figure 36), we can see that with that by applying the ensemble models the variance got even smaller.

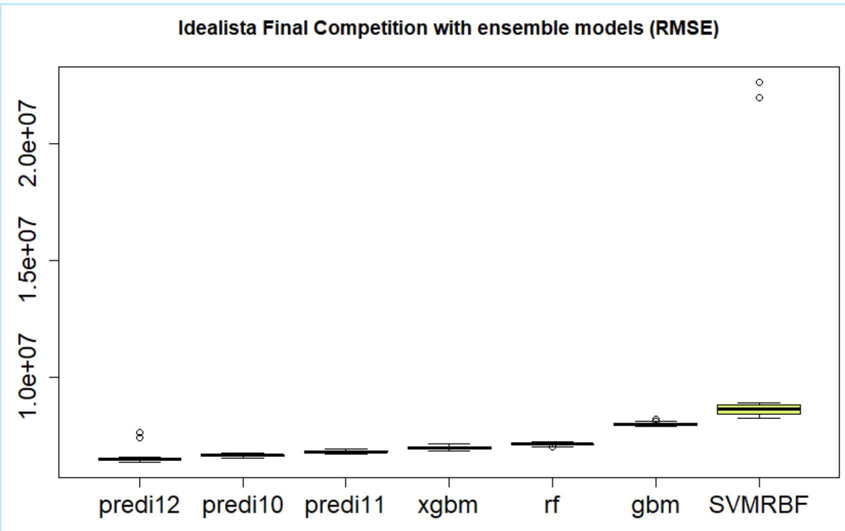


Figure 36: Idealista Final Model Assesment (Boxplot)

With all these studies, we are finally able to conclude the modeling phase of the Idealista Rental Model. Later in chapter 5, we will use the *Predi12* to predict the traditional rental yearly income of properties on sale, as a simulation of what would run in the back-end of Rentalbiliy application.

4. AIRBNB DATA ANALYSIS

On this chapter, we carry out the complete data analysis process for Airbnb rental data, following the same methodology from the previous model. The final result from this section is the model, which we will use to predict the rent for the properties in the short-term. We will also make use of the cleaned dataset to develop the occupancy rate study in chapter 6.

4.1. Data Source

We started our search for Airbnb data on its official webpage. Unfortunately, Airbnb does not offer an open data tool for developers to access their data, such as datasets or APIs, although it is possible to scrape it. A website called Inside Airbnb provides the scraped datasets monthly from several cities around the world where Airbnb is available. We collected the dataset (called *listings)* with 106 variables and more than 17000 observations, which contains all the properties available on the Airbnb platform in March 2019 on Madrid.

The variables in this dataset contained the same information available on the listing webpage. It had ad-related variables, such as ad title, house description and link to the photos; location-related variables, as latitude, longitude, neighborhood and district; listing's characteristics, as the number of room, bathroom, accommodation capacity, amenities, square feet; price-related variables, daily price, cleaning fee, security deposit; host-related variables, as host id, name, identity verification, total listings; and other relevant information as reviews-related variables. In Appendix E, a summary can be found with all the original variables and the additional ones we created.

4.1.1. Prework

Eventually, some pre-processing of the data in Excel was needed before we could start analyzing our data. Some of the essential information for quantifying Airbnb's revenue, such as the occupancy and booking rate, were not accessible to be scraped and thus, were not available on our dataset. Therefore, we had to develop several calculations in order to estimate an accurate yearly revenue. We followed the "San Francisco Model" methodology, which is also recommended by Inside Airbnb (Cox, 2019).

The San Francisco Model refers to a method created by Alex Marqusee for the San Francisco Planning Department (Brousseau, 2015) and the Budget and Legislative Analyst's Office (Rodgers, 2015) to quantify the impact of Airbnb in this city. These institutions used the method to develop the vacation's rentals regulations in the city. In 2017, Madrid's Municipal Board of the Center District also used the method in an analysis of the impact of vacation lodging in the city (Junta Municipal Distrito Centro and RED2RED, 2017). This method uses the review data available on Airbnb to estimate the listing's bookings, occupancy rate and revenue.

We used the "San Francisco Model" as a base for our calculations, but with a few adaptations to Madrid's case. Below we have the list of variables and formulas we need to create:

- **Days on Airbnb** $= last_{review} - first_{review}$

- **Minimum Booking in YEAR** = $IFERROR(number\_of\_reviews/(Days\ on\ Airbnb/365); reviews\_per\_month * 12)$

  According to (Marqusee, 2015), the minimum number of bookings a listing could have in Airbnb is the number of reviews this listing received, assuming that each review relates to a guest's booking. Therefore, the average minimum number of bookings in a year would be total bookings of the lodge by the number of years this property is in Airbnb.

- **Estimated Bookings in Year** = $MIN\_Booking\_YEAR/50\%$
  In order to determinate the *estimated booking* of a listing (Marqusee, 2015) uses review rate, which is an assumption of the percentage of guests who leaves a review after the stay. The "San Francisco Model" actually uses two review rates: one of 72%, and another of 30.5%. However, Inside Airbnb (Cox, 2019) considers the first unverifiable (since it was attributed to the speech of Airbnb's CEO and co-founder Brian Chesky); and the second one "not conservative enough" (it does not take into consideration missing reviews due to deleted listings). Therefore Cox (2019) suggests a 50% review rate, as it sits almost between 72% and 30.5%. We also assumed the Review Rate of 50%, which implies that at least 50% of the people who booked a property left a review, and subsequently, the number of bookings of a listing should be the double of the reviews.

- **Nights Per YEAR CAP** =
  $$IF(EST\_Bookings\_YEAR * IF(minimum\_nights\_avg\_ntm > 2; minimum\_nights\_avg\_ntm; 3{,}7) > 255; 255; EST\_Bookings\_YEAR * IF(minimum\_nights\_avg\_ntm > 3{,}7; minimum\_nights\_avg\_ntm; 3{,}7))$$
  For the booked *Nights per year,* we considered the average length of stay of 3.7 nights, as declared in the Airbnb Economic Activity Report in the City of Madrid (Airbnb, 2019b). Though, if a listing has a higher minimum night than the average length of stay, the minimum nights was used instead. We set a limit of 255 nights (70%) per year since the Statistical Institute of Madrid points at that the average occupancy rate for touristic flats in Madrid was 70% in 2017 (Instituto de Estadística de la Comunidad de Madrid, 2019).

- **Occupancy Rate** = $Nights\_Per\_YEAR\_CAP/365$
  The *occupancy rate* is computed as *nights per year* divided by 365.

- **Yearly Revenue** = $price * Occupancy\ Rate * 365$

- **Utilities Cost** = $81{,}144648585 + ((guests\_included - 1) * 17{,}501774895)$
  The utility cost was estimated based on basic costs (electricity, heating, cooling, water, garbage) for a house of one person (the equivalent of 81.14€) plus the capacity of the house by 17.5, which is the progression in which this cost increase. These numbers come from a formula used on the website Numbeo (2019) and were calculated in March 2019.

- **Cost Year** =
  $$(Yearly\ Revenue * 3\%) + 42{,}73 + ((Utilities\ Cost) * 12 * Occupancy\ Rate)$$

The 3% of the revenue is the Airbnb's service fee (Airbnb, 2019c) and the 42,73 is an approximated price for internet in Madrid, also based on Numbeo (2019) figures. We used the occupancy rate to adjust the costs to the listings occupation.

- **Yearly Profit** $= Yearly\ Revenue - Cost\ per\ Year$

Finally, we limited the *room_type* to "entire home/apt", since we want to evaluate the investment of buying a house and renting it entirely, besides all properties in Idealista are entire houses and the rental income would not be comparable to a shared room. We also filter the *host_verifications* for "government_id" to ensure we were using real houses with the host identity verified.

4.2. Data Exploration in SAS Enterprise Miner

After the pre-processing of the data in Excel, we started our mining work using SAS Enterprise Miner with 6656 observations and 52 variables. We assigned the roles accordingly with the functions and types of each variable. In Figure 37, we can find a summary of the variables we worked with.

Variable Name	Role	Measurement Level ▲
Air conditioning	Input	Binary
Bathtub	Input	Binary
Breakfast	Input	Binary
Coffee maker	Input	Binary
Cooking basics	Input	Binary
Free street parking	Input	Binary
Has License	Input	Binary
Host greets you	Input	Binary
Hot water	Input	Binary
Internet	Input	Binary
Laptop friendly workspace	Input	Binary
Long term stays allowed	Input	Binary
Microwave	Input	Binary
Patio or balcony	Input	Binary
Pool	Input	Binary
Refrigerator	Input	Binary
Shampoo	Input	Binary
24 hour check in	Input	Binary
host identity verified	Input	Binary
host is superhost	Input	Binary
instant bookable	Input	Binary
is location exact	Input	Binary
Yearly Profit	Target	Interval
availability rate	Input	Interval
calculated host listings count e	Input	Interval
cleaning fee	Input	Interval
extra people	Input	Interval
host response rate	Input	Interval
host since days	Input	Interval
latitude	Input	Interval
longitude	Input	Interval
maximum nights	Input	Interval
minimum nights	Input	Interval
number of reviews	Input	Interval
number of reviews ltm	Input	Interval
review scores accuracy	Input	Interval
review scores checkin	Input	Interval
review scores cleanliness	Input	Interval
review scores communication	Input	Interval
review scores location	Input	Interval
review scores rating	Input	Interval
review scores value	Input	Interval
security deposit	Input	Interval
square feet	Rejected	Interval
accommodates	Input	Nominal
bathrooms	Input	Nominal
bed type	Rejected	Nominal
bedrooms	Input	Nominal
beds	Input	Nominal
cancellation policy	Input	Nominal
host response time	Input	Nominal
id	ID	Nominal
neighbourhood cleansed	Input	Nominal
neighbourhood group cleansed	Input	Nominal
zipcode	Rejected	Nominal

Figure 37: Airbnb Variables Roles and Levels

4.2.1. Interval Variables Statistical Analysis

In Figure 38, we can see the statistical analysis of the interval observations before any modification.

Variable	Role	Mean	Standard Deviation	Non Missing	Missing	Minimum	Median	Maximum	Skewness	Kurtosis
Random	INPUT	0.500322	0.28835	6656	0	0.00005	0.499261	0.999986	0.004736	-1.18915
availability_rate	INPUT	0.327053	0.264081	6656	0	0	0.29589	0.986301	0.500274	-0.66043
calculated_host_listings_count_e	INPUT	10.14498	17.35327	6656	0	1	2	92	2.648994	7.16893
cleaning_fee	INPUT	34.83093	24.56723	6175	481	0	30	600	3.88721	54.60269
extra_people	INPUT	10.46499	11.22611	6656	0	0	10	240	4.318902	63.83647
host_response_rate	INPUT	97.78832	8.037534	6368	288	0	100	100	-6.70561	59.57874
host_since_days	INPUT	1438.055	707.3432	6656	0	41	1372	3769	0.200165	-0.83032
latitude	INPUT	40.41951	0.01583	6656	0	40.33249	40.41707	40.51085	1.073289	5.770316
longitude	INPUT	-3.69887	0.017778	6656	0	-3.8355	-3.70224	-3.57699	1.785318	10.34552
maximum_nights	INPUT	817.773	1386.767	6656	0	1	1125	100000	56.65622	3963.722
minimum_nights	INPUT	2.877404	5.18447	6656	0	1	2	120	11.25344	168.376
number_of_reviews	INPUT	59.45388	69.26427	6656	0	1	35	555	2.114138	5.898728
number_of_reviews_ltm	INPUT	25.22596	23.44649	6656	0	1	18	155	1.211355	1.312383
review_scores_accuracy	INPUT	9.560085	0.777737	6624	32	2	10	10	-3.67833	24.57405
review_scores_checkin	INPUT	9.670592	0.723751	6624	32	2	10	10	-4.45753	33.4807
review_scores_cleanliness	INPUT	9.448219	0.798065	6624	32	2	10	10	-2.72583	15.02416
review_scores_communication	INPUT	9.709239	0.690086	6624	32	2	10	10	-4.77694	38.27604
review_scores_location	INPUT	9.713768	0.603641	6624	32	2	10	10	-3.64091	26.64957
review_scores_rating	INPUT	92.84209	7.315508	6624	32	20	94	100	-3.35469	21.76922
review_scores_value	INPUT	9.208937	0.824364	6624	32	2	9	10	-2.28825	12.67461
security_deposit	INPUT	156.616	179.9573	5789	867	0	150	4000	7.216986	112.9435
Yearly_Profit	TARGET	11883.07	9643.467	6656	0	88.28587	10192.86	122636.5	2.522119	13.4362

Figure 38: Airbnb Interval Variable Summary Statistics before changes

We observed a few missing values in some variables, like *Security deposit* (13%) and *cleaning fee* (7%). For these variables, we decided to impute the observations using the tree method (it estimates each value to be imputed based on the other input variables, thus it is more accurate than using the mean or median of the variable to replace the missing values). Regarding the Review-related variables, we observed 0,05% of missing, thus we decided to delete these 32 observations.

We decided to reject *Square feet* after an analysis of missing (6483 missing), *Zipcode* since it is a nominal variable with many levels and we already have similar information with the neighborhood and *bed_type* due to unbalanced levels (6563 "Real Bed").

Regarding the maximum and minimum observation limits, the only anomaly was *maximum nights,* where we replaced all the observation higher than 365 nights with 365 (according to our definition of short-term rental as less than a year), its skewness decreased to -1.17 after this adjustment. We performed an analysis to detect outliers using the Mean Absolute Deviation, Standard deviation, and Interquartile Range methods, but the results were very similar to the original limits. Therefore we kept them as they were. Figure 39 shows the interval variables after changes.

Variable	Role	Mean	Standard Deviation	Non Missing	Missing	Minimum	Median	Maximum	Skewness	Kurtosis
IMP_REP_cleaning_fee	INPUT	34.22936	22.13175	6624	0	0	30	180	1.874363	7.535715
IMP_host_response_rate	INPUT	97.87949	7.711398	6624	0	0	100	100	-6.90606	63.67542
IMP_security_deposit	INPUT	156.1314	169.1916	6624	0	0	150	4000	7.584762	126.6702
REP_maximum_nights	INPUT	283.2745	138.5545	6624	0	1	365	365	-1.17504	-0.52524
REP_review_scores_accuracy	INPUT	9.568086	0.714942	6624	0	5	10	10	-2.4669	9.442694
REP_review_scores_checkin	INPUT	9.677989	0.659219	6624	0	5	10	10	-3.0565	13.56028
REP_review_scores_cleanliness	INPUT	9.453955	0.756666	6624	0	5	10	10	-1.94284	6.131642
REP_review_scores_communication	INPUT	9.716184	0.626124	6624	0	5	10	10	-3.24751	15.25738
REP_review_scores_location	INPUT	9.716184	0.578494	6624	0	5	10	10	-2.70353	10.86736
REP_review_scores_rating	INPUT	92.90399	6.82924	6624	0	50	94	100	-2.30896	8.744871
REP_review_scores_value	INPUT	9.21558	0.78007	6624	0	5	9	10	-1.49572	4.768538
Random	INPUT	0.500945	0.28829	6624	0	0.00005	0.500027	0.999986	0.002529	-1.18882
availability_rate	INPUT	0.326156	0.262978	6624	0	0	0.29589	0.986301	0.498609	-0.65945
calculated_host_listings_count_e	INPUT	10.12711	17.3352	6624	0	1	2	92	2.653189	7.194235
extra_people	INPUT	10.48188	11.22507	6624	0	0	10	240	4.336031	64.1511
host_since_days	INPUT	1438.509	706.8497	6624	0	41	1370	3769	0.201143	-0.8306
latitude	INPUT	40.41947	0.015759	6624	0	40.33249	40.41704	40.50777	1.052171	5.524104
longitude	INPUT	-3.69888	0.017747	6624	0	-3.8355	-3.70224	-3.57699	1.779937	10.3932
minimum_nights	INPUT	2.867905	5.170992	6624	0	1	2	120	11.35451	170.8508
numMissing	INPUT	0.288043	0.673971	6624	0	0	0	4	2.534833	6.513694
number_of_reviews	INPUT	59.73536	69.31259	6624	0	1	36	555	2.110601	5.88003
number_of_reviews_ltm	INPUT	25.34254	23.44282	6624	0	1	18	155	1.208133	1.306317
Yearly_Profit	TARGET	11917.08	9646.363	6624	0	181.0599	10218.42	122636.5	2.52525	13.45561

Figure 39: Airbnb Interval Variable Summary Statistics after changes

4.2.2. Class Variables Statistical Analysis

In Figure 40, we have the class variable analysis before the changes. Similar to Idealista's data, we also faced some issues with lack of representations within the classes.

Data Role	Variable Name	Role	Number of Levels	Missing	Mode	Mode Percentage	Mode2	Mode2 Percentage
TRAIN	Air_conditioning	INPUT	2	0	1	82.14	0	17.86
TRAIN	Bathtub	INPUT	2	0	0	92.05	1	7.95
TRAIN	Breakfast	INPUT	2	0	0	91.17	1	8.83
TRAIN	Coffee_maker	INPUT	2	0	1	53.64	0	46.36
TRAIN	Cooking_basics	INPUT	2	0	1	51.31	0	48.69
TRAIN	Free_street_parking	INPUT	2	0	0	92.37	1	7.63
TRAIN	Has_License	INPUT	2	0	0	63.69	1	36.31
TRAIN	Host_greets_you	INPUT	2	0	1	54.37	0	45.63
TRAIN	Hot_water	INPUT	2	0	1	76.46	0	23.54
TRAIN	Internet	INPUT	2	0	0	68.39	1	31.61
TRAIN	Laptop_friendly_workspace	INPUT	2	0	1	74.85	0	25.15
TRAIN	Long_term_stays_allowed	INPUT	2	0	0	51.26	1	48.74
TRAIN	Microwave	INPUT	2	0	1	54.54	0	45.46
TRAIN	Patio_or_balcony	INPUT	2	0	0	85.16	1	14.84
TRAIN	Pool	INPUT	2	0	0	96.86	1	3.14
TRAIN	Refrigerator	INPUT	2	0	1	57.14	0	42.86
TRAIN	Shampoo	INPUT	2	0	1	80.89	0	19.11
TRAIN	_24_hour_check_in	INPUT	2	0	0	86.60	1	13.40
TRAIN	accommodates	INPUT	16	0	4	37.47	2	18.58
TRAIN	bathrooms	INPUT	15	2	1,0	71.54	2,0	18.55
TRAIN	bedrooms	INPUT	10	2	1	46.03	2	30.71
TRAIN	beds	INPUT	19	1	2	33.91	1	26.40
TRAIN	cancellation_policy	INPUT	5	0	strict_14_with_grace_period	42.52	moderate	39.78
TRAIN	host_identity_verified	INPUT	2	0	f	53.05	t	46.95
TRAIN	host_is_superhost	INPUT	2	0	f	69.31	t	30.69
TRAIN	host_response_time	INPUT	5	289	within an hour	81.13	within a few hours	9.38
TRAIN	instant_bookable	INPUT	2	0	t	69.89	f	30.11
TRAIN	is_location_exact	INPUT	2	0	t	73.36	f	26.64
TRAIN	neighbourhood_cleansed	INPUT	117	0	Embajadores	18.15	Universidad	13.25
TRAIN	neighbourhood_group_cleansed	INPUT	21	0	Centro	64.77	Salamanca	6.70

Figure 40: Airbnb Class Variable Summary Statistics before changes

In this case, we had to merge several categories due to their low frequency. For example, in *Accommodates* we had to merge together the categories with more than 7, in *Bathrooms* we combined the 1+1,5; 2+2,5 and 3+, in *Bedrooms* and *beds* we unified all the categories with more than 4 and 7, respectively (for bedroom and bathroom we kept the same structure of the Idealista dataset). In Appendix F, there is a detailed explanation of all the modifications applied to the class variables.

We also identified a few missings in the variables *bathrooms*, *rooms*, *beds*, *host response rate.* We decided to impute these observations with the Tree method.

Data Role	Variable Name	Role	Number of Levels	Missing	Mode	Mode Percentage	Mode2	Mode2 Percentage
TRAIN	Air_conditioning	INPUT	2	0	1	82.17	0	17.83
TRAIN	Bathtub	INPUT	2	0	0	92.03	1	7.97
TRAIN	Breakfast	INPUT	2	0	0	91.21	1	8.79
TRAIN	Coffee_maker	INPUT	2	0	1	53.79	0	46.21
TRAIN	Cooking_basics	INPUT	2	0	1	51.42	0	48.58
TRAIN	Free_street_parking	INPUT	2	0	0	92.36	1	7.64
TRAIN	Has_License	INPUT	2	0	0	63.56	1	36.44
TRAIN	Host_greets_you	INPUT	2	0	1	54.51	0	45.49
TRAIN	Hot_water	INPUT	2	0	1	76.68	0	23.32
TRAIN	IMP_REP_bathrooms	INPUT	3	0	1	77.29	2	19.57
TRAIN	IMP_REP_bedrooms	INPUT	5	0	1	46.09	2	30.75
TRAIN	IMP_REP_beds	INPUT	8	0	2	33.95	1	26.36
TRAIN	IMP_REP_host_response_time	INPUT	2	0	within an hour	83.76	more than a hour	16.24
TRAIN	Internet	INPUT	2	0	0	68.30	1	31.70
TRAIN	Laptop_friendly_workspace	INPUT	2	0	1	74.98	0	25.02
TRAIN	Long_term_stays_allowed	INPUT	2	0	0	51.07	1	48.93
TRAIN	M_Variable	INPUT	5	0	0	81.52	1	9.80
TRAIN	Microwave	INPUT	2	0	1	54.66	0	45.34
TRAIN	Patio_or_balcony	INPUT	2	0	0	85.11	1	14.89
TRAIN	Pool	INPUT	2	0	0	96.88	1	3.13
TRAIN	REP_accommodates	INPUT	6	0	4	37.53	2	18.77
TRAIN	REP_cancellation_policy	INPUT	3	0	Strict	43.33	moderate	39.86
TRAIN	Refrigerator	INPUT	2	0	1	57.28	0	42.72
TRAIN	Shampoo	INPUT	2	0	1	80.95	0	19.05
TRAIN	_24_hour_check_in	INPUT	2	0	0	86.53	1	13.47
TRAIN	host_identity_verified	INPUT	2	0	f	53.03	t	46.97
TRAIN	host_is_superhost	INPUT	2	0	f	69.16	t	30.84
TRAIN	instant_bookable	INPUT	2	0	t	70.00	f	30.00
TRAIN	is_location_exact	INPUT	2	0	t	73.37	f	26.63
TRAIN	neighbourhood_cleansed	INPUT	117	0	Embajadores	18.18	Universidad	13.24
TRAIN	neighbourhood_group_cleansed	INPUT	21	0	Centro	64.87	Salamanca	6.72

Figure 41: Airbnb Class Variable Summary Statistics after changes

The summary of these variables after changes is presented in Figure 41. As we can see above, with the variables *neighborhood_cleansed* and *neighborhood_cleansed_group* (districts) we faced the same collinearity and overly levels issue we had with Idealista data. The approach for overcoming this problem was the same: group *neighborhood_cleansed into* smaller groups (according to its relation with the target variable) using the *Variable Selection Node*. A table with the relation between the neighborhoods and its groups is in Appendix G.

4.2.3. Variables Importance and Correlation

The most important variables (Figure 42) are related to the capacity of accommodation, like the number of bathrooms, beds and bedrooms, which is evident since they affect the price of the stay, and consequently, the yearly profit. Subsequently, we see the review-related variables. Once more, it is understandable, as the reviews directly affect the occupancy rate, since people rely on reviews on their decision-making process. Moreover, we see the *number of reviews ltm* (last twelve months) as the most important variable, again it is closely related with the profit, the more reviews a listing can get, the more probable it will be booked often. Finally, we can highlight the importance of other interesting variables, such as the *latitude*, *neighborhood* and the presence of a *coffeemaker*, *microwave*, *air conditioning*, *refrigerator*, *patio/balcony*, *laptop-friendly*, *shampoo*, *bathtub* which are amenities and location-related variables, these indicate what a customers take into consideration before making booking decisions. We also added a random variable to define which variables are not important, among which we can see *internet*, *cancelation policy* and *pool*.

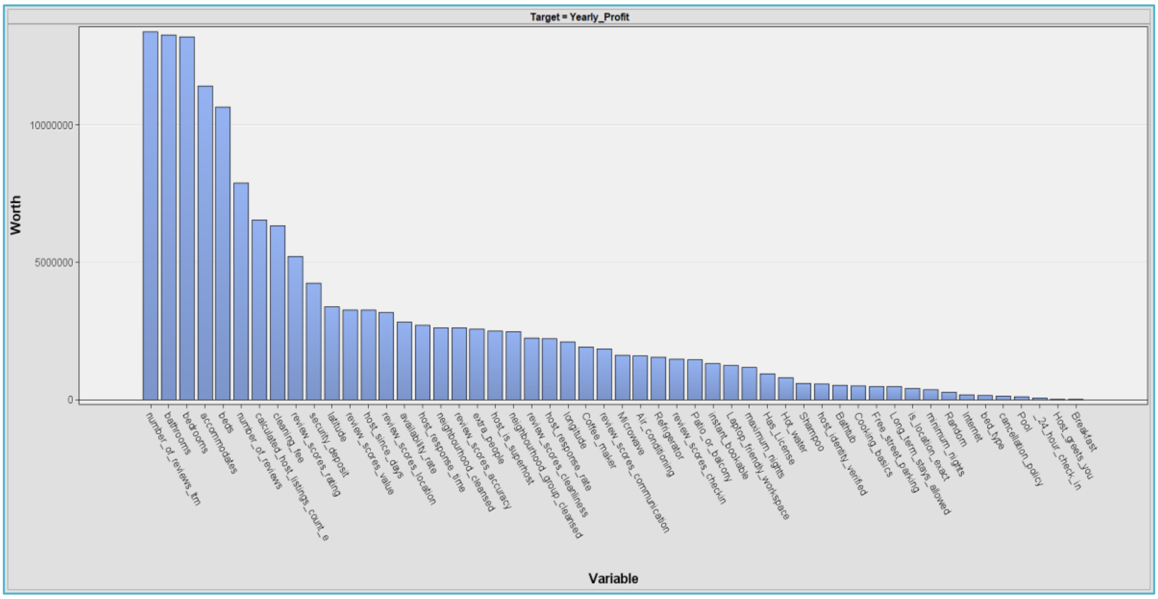


Figure 42: Airbnb Variables Worth

Furthermore, we can see again that the reviews-related variables are also more correlated with the target variable (Figure 43).

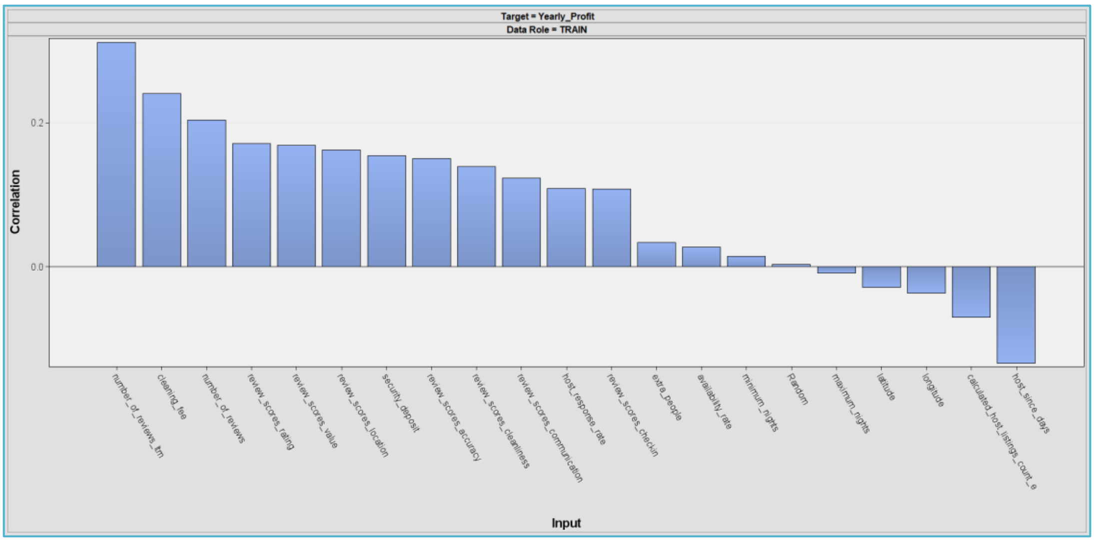


Figure 43: Airbnb Variables Correlation

4.2.4. Variables Selection

On the Airbnb's variables selection phase, we followed the same strategy we implemented in the Idealista dataset. Again, we run six different models from seven different branches of transformations and variable selection approaches (we used the same configuration described in section 3.3.4). In Appendix H, we have the results of the nodes with the most relevant impact on the models.

Starting from the same Idealista perspective, we decided not to use *calculated host listings count e, host is superhost, host since days, number of reviews ltm*, and any review-related variable to the modeling phase, since the investor would not have any power over these variables before buying the property, hence they would not affect the ROI.

After running the model comparison node (see Figure 44), we concluded that this dataset behaves very similar to Idealista's. The best models were also the gradient boosting ones (which was predictable), departing from branch 1 or 2 (grouping for the neighborhood and grouping together with transformation). Subsequently, we run a Repeated Training-Test (10 repetitions) with the selected ten best gradient boosting models (highlighted in green in Figure 44) from the different branches. The final boxplot is shown in Figure 45.

The best variable selection again is the one coming from branch 1, which had no transformations, besides grouping *neighborhood*. We compared this selection with other models (highlighted in green in Figure 45). They all selected almost the same variables and with a similar importance ratio (Figure 46).

Model Node	Model Description	Test: Root Average Squared Error ▲
Boost16	1.6 Gradient Boosting	7422.296
Boost7	2.6 Gradient Boosting	7422.706
Boost4	0.6 Gradient Boosting	7430.429
Boost3	0.5 Gradient Boosting	7491.614
Boost6	2.5 Gradient Boosting	7493.912
Boost15	1.5 Gradient Boosting	7494.04
Boost12	5.6 Gradient Boosting	7567.131
Boost11	5.5 Gradient Boosting	7620.226
Neural12	2.3 Neural Network	7767.86
Boost9	3.6 Gradient Boosting	7831.499
Boost8	3.5 Gradient Boosting	7844.196
Neural5	5.4 Neural Network	7866.873
Neural10	2.4 Neural Network	7868.102
Boost10	4.5 Gradient Boosting	7901.546
Neural14	1.3 Neural Network	7907.54
Boost5	4.6 Gradient Boosting	7908.006
Neural15	0.4 Neural Network	7909.1
Neural16	0.3 Neural Network	7918.467
Neural6	5.3 Neural Network	7928.476
Boost2	7.6 Gradient Boosting	7947.05
Boost	7.5 Gradient Boosting	7951.263
Neural13	1.4 Neural Network	7963.72
Boost14	6.6 Gradient Boosting	8015.313
Boost13	6.5 Gradient Boosting	8015.444
Reg8	0.2 Regression SCVM	8019.26
Neural9	3.4 Neural Network	8038.613
Reg5	5.2 Regression SCVM	8066.084
Neural11	3.3 Neural Network	8069.729
Reg2	1.2 Regression SCVM	8098.686
Neural8	4.3 Neural Network	8131.862
Reg	2.2 Regression SCVM	8142.796
Reg3	3.2 Regression SCVM	8149.557
Reg4	4.2 Regression SCVM	8172.775
Neural4	6.3 Neural Network	8187.118
Neural	7.4 Neural Network	8189.422
Neural3	7.3 Neural Network	8241.017
Neural7	4.4 Neural Network	8245.298
Reg7	7.2 Regression SCVM	8246.26
Neural2	6.4 Neural Network	8266.809
Reg6	6.2 Regression SCVM	8268.872
Tree8	6.1 Decision Tree V	8279.169
Tree6	4.1 Decision Tree V	8299.201
Tree	2.1 Decision Tree V	8308.319
Tree3	1.1 Decision Tree V	8308.319
Tree5	3.1 Decision Tree V	8310.479
Tree7	5.1 Decision Tree V	8323.928
Tree9	7.1 Decision Tree V	8337.693
Tree10	0.1 Decision Tree V	8382.529

Figure 44: Airbnb Model Comparison Results

Since we do not see any significative drop on the importance ratio in Figure 46 in order to define a cut point, we decided to keep the variables selected by model 1.5, with relative importance strictly higher than 0%.

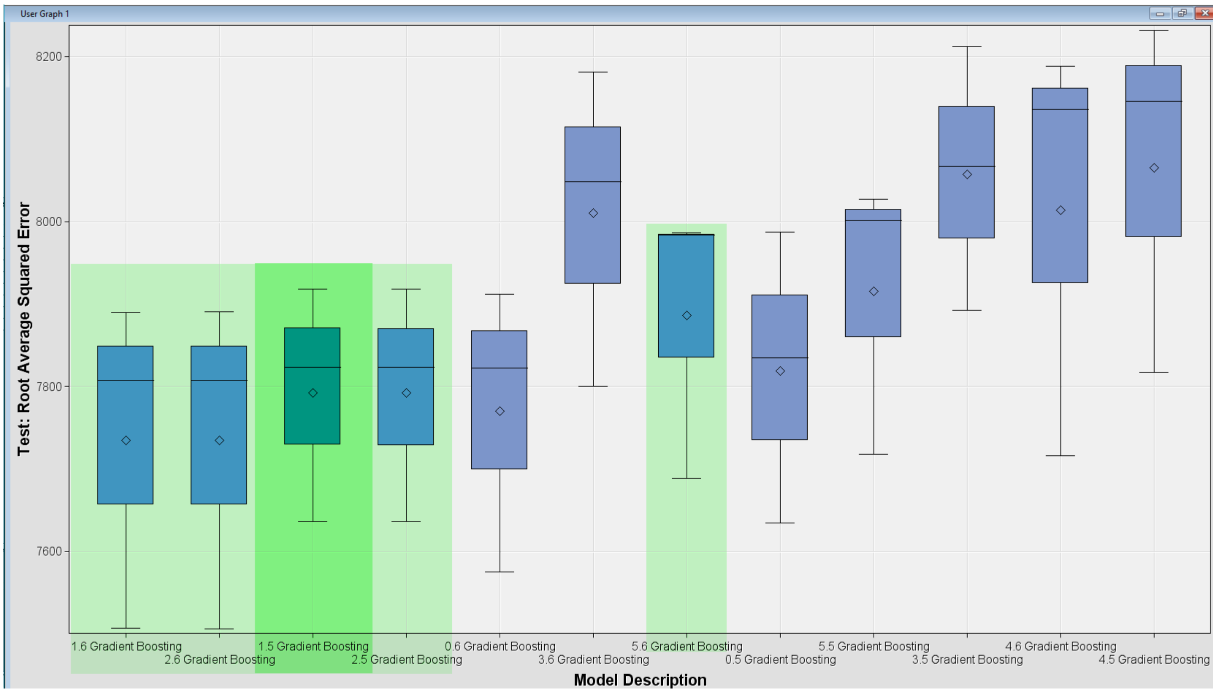


Figure 45: Airbnb Box-Plot for Repeated Training-Test

In Figure 46, we can see in green all 32 variables we use in the modeling phase in R and the occupancy rate study.

Variables	VI 1.5	VI 2.5	VI 1.6	VI 2.6	Mean
IMP_REP_bathrooms	100%	100%	100%	100%	100%
IMP_REP_bedrooms	67%	67%	70%	70%	68%
G_neighbourhood_cleansed	62%	62%	64%	64%	63%
IMP_REP_cleaning_fee	60%	60%	64%	64%	62%
IMP_security_deposit	57%	57%	64%	64%	60%
availability_rate	56%	56%	61%	61%	58%
REP_accommodates	54%	54%	57%	57%	55%
IMP_REP_beds	47%	47%	55%	55%	51%
longitude	39%	39%	46%	46%	42%
Air_conditioning	34%	34%	33%	33%	33%
latitude	34%	34%	41%	41%	37%
Shampoo	33%	33%	34%	34%	34%
REP_cancellation_policy	33%	33%	37%	37%	35%
IMP_REP_host_response_time	32%	32%	31%	31%	31%
Laptop_friendly_workspace	31%	31%	29%	29%	30%
host_identity_verified	28%	28%	23%	23%	25%
REP_maximum_nights	28%	28%	33%	33%	30%
Refrigerator	25%	25%	29%	29%	27%
minimum_nights	28%	28%	33%	33%	30%
extra_people	24%	24%	37%	37%	30%
Host_greets_you	22%	22%	24%	24%	23%
Has_License	20%	20%	24%	24%	22%
Coffee_maker	16%	16%	18%	18%	17%
Long_term_stays_allowed	16%	16%	14%	14%	15%
is_location_exact	14%	14%	13%	13%	14%
instant_bookable	13%	13%	8%	8%	11%
Hot_water	12%	12%	12%	12%	12%
Internet	10%	10%	10%	10%	10%
Cooking_basics	9%	9%	9%	9%	9%
Patio_or_balcony	4%	4%	5%	5%	5%
_24_hour_check_in	3%	3%	1%	1%	2%
Microwave	3%	3%	3%	3%	3%
Pool	0%	0%	0%	0%	0%
Breakfast	0%	0%	0%	0%	0%
Bathtub	0%	0%	0%	0%	0%
t_parking	0%	0%	0%	0%	0%

Figure 46: Airbnb Variable Selection Analysis

4.3. Modeling in R

With the Airbnb data prepared, we started the modeling phase for the short-term rentals.

4.3.1. Neural Network

We tunned the Neural Network with both NNET and avNNet functions from caret package with repeated cross-validation with 5 repetitions and random seeds. For the architecture selection of the NNET, we used 2,6,8,10,13,15,20 units per hidden layer, (since we have 6624 observations and 32 variables to get 20 obs/parameters we would need 10 hidden layers: h (34 + 1)+ h + 1=6624 /20, thus we set that range which implies from 100 to 12 obs/parameter). We set decays of 0.01,0.1,0.001,0.2,0.05.

```r
nnetgrid <-  expand.grid(size=c(2,6,8,10,13,15,20),decay=c(0.01,0.1,0.001,0.2,0.05))

rednnet<- train(Yearly_Profit~.,data=airbnbbis,
                method="nnet",linout = TRUE,maxit=100,trControl=control,tuneGrid=nnetgrid)
```

With this function, we obtained the following result in Figure 47, where the best model is found to have 20 hidden layers, a weight decay of 0.2, R^2 of 0.1041. For sure we could try to improve this model, but we decided to it with the avNNet function.

```
6624 samples
  32 predictor

No pre-processing
Resampling: Cross-Validated (4 fold, repeated 5 times)
Summary of sample sizes: 4968, 4968, 4968, 4968, 4968, 4968, ...
Resampling results across tuning parameters:

  size  decay  RMSE      Rsquared    MAE
   2    0.001  9640.007         NaN  6694.695
   2    0.010  9578.222  0.08285140  6665.454
   2    0.050  9598.926  0.09634876  6667.778
   2    0.100  9551.682  0.06370993  6643.767
   2    0.200  9441.192  0.07625946  6588.865
   6    0.001  9579.124  0.07569013  6666.929
   6    0.010  9583.150  0.09320555  6647.859
   6    0.050  9497.029  0.08473427  6615.949
   6    0.100  9529.437  0.07396895  6634.195
   6    0.200  9325.673  0.09316868  6542.431
   8    0.001  9515.573  0.07469302  6633.392
   8    0.010  9583.537  0.08786472  6669.741
   8    0.050  9572.378  0.05371294  6658.149
   8    0.100  9532.702  0.06334650  6636.823
   8    0.200  9388.561  0.07457173  6559.304
  10    0.001  9463.169  0.12063214  6592.380
  10    0.010  9528.598  0.07495758  6634.742
  10    0.050  9509.490  0.06550313  6641.696
  10    0.100  9384.022  0.09780158  6577.296
  10    0.200  9340.201  0.08883527  6551.629
  13    0.001  9462.615  0.06224413  6602.172
  13    0.010  9414.375  0.09493766  6549.682
  13    0.050  9416.190  0.07630090  6592.543
  13    0.100  9512.731  0.05924000  6635.186
  13    0.200  9300.214  0.09504641  6513.488
  15    0.001  9409.094  0.08568715  6587.123
  15    0.010  9488.043  0.07907396  6613.157
  15    0.050  9293.023  0.10181849  6513.743
  15    0.100  9336.493  0.09070534  6523.113
  15    0.200  9291.719  0.08978167  6514.361
  20    0.001  9361.550  0.07243122  6544.376
  20    0.010  9374.605  0.09220127  6555.679
  20    0.050  9348.041  0.07789179  6539.910
  20    0.100  9365.873  0.10182066  6554.909
  20    0.200  9216.811  0.10413366  6456.731

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were size = 20 and decay = 0.2.
```

Figure 47: Airbnb NNET results

For the configuration of the neural networks models with avNNet, we reduced the grid for the hidden layers: 8, 10, 12, 15, 18, 20, 22. We tested with the same learning rates from the previous function.

```
avnnetgrid <-expand.grid(size=c(8,10,12,15,18,20,22),decay=c(0.01,0.1,0.001,0.2,0.05),bag=FALSE)

redavnnet<- train(Yearly_Profit~.,data=airbnbbis,
                  method="avNNet",linout = TRUE,maxit=100,trControl=control,repeats=5,tuneGrid=avnnetgrid)
```

With this function, we got some improvements. Our best model also had 20 hidden layers, a weight decay of 0.1, and 0,1558 equals to R^2, as shown in Figure 48.

```
> redavnnet
Model Averaged Neural Network

6624 samples
  32 predictor

No pre-processing
Resampling: Cross-Validated (4 fold, repeated 5 times)
Summary of sample sizes: 4968, 4968, 4968, 4968, 4968, 4968, ...
Resampling results across tuning parameters:

  size  decay  RMSE      Rsquared    MAE
   8    0.001  9552.841  0.05719948  6635.185
   8    0.010  9537.497  0.08163962  6625.296
   8    0.050  9389.692  0.10686208  6513.369
   8    0.100  9368.756  0.11969489  6506.606
   8    0.200  9185.905  0.13337294  6383.739
  10    0.001  9422.457  0.09881381  6544.410
  10    0.010  9344.512  0.11763727  6498.217
  10    0.050  9380.943  0.10873519  6522.509
  10    0.100  9255.989  0.13837425  6436.925
  10    0.200  9112.533  0.15371454  6348.450
  12    0.001  9404.401  0.09937678  6528.474
  12    0.010  9357.273  0.11317729  6504.623
  12    0.050  9316.494  0.12019508  6470.767
  12    0.100  9294.591  0.11423172  6466.829
  12    0.200  9142.255  0.14672101  6359.637
  15    0.001  9353.872  0.10706185  6499.037
  15    0.010  9297.389  0.10819618  6460.703
  15    0.050  9214.850  0.13787207  6401.949
  15    0.100  9150.642  0.14679756  6372.841
  15    0.200  9112.439  0.14528737  6347.000
  18    0.001  9305.940  0.11674454  6475.931
  18    0.010  9245.655  0.13290568  6430.530
  18    0.050  9160.993  0.14384134  6372.551
  18    0.100  9142.778  0.15052338  6372.078
  18    0.200  9160.193  0.13807217  6372.231
  20    0.001  9273.330  0.12050608  6457.500
  20    0.010  9113.738  0.14164452  6355.213
  20    0.050  9133.339  0.14803897  6364.348
  20    0.100  9069.343  0.15580387  6327.109
  20    0.200  9153.267  0.13747520  6370.139
  22    0.001       NaN         NaN       NaN
  22    0.010       NaN         NaN       NaN
  22    0.050       NaN         NaN       NaN
  22    0.100       NaN         NaN       NaN
  22    0.200       NaN         NaN       NaN

Tuning parameter 'bag' was held constant at a value of FALSE
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were size = 20, decay = 0.1 and bag = FALSE.
```

Figure 48: Airbnb avNNet results

4.3.2. Random Forest and Bagging

In the Random Forest tunning, we also started by searching the finest number of variables for each tree (mtry). We tested 5,8,10,12,15,18,20,25,30 and 32 (which is the bagging model) variables. On this first try, we did not sample the observations. We set 1000 trees and a minimum of 20 obs per node. We also used cross-validation with 5 repetitions.

```
rfgrid<-expand.grid(mtry=c(5,8,10,12,15,18,20,25,30,32))

rf<- train(Yearly_Profit~.,data=airbnbbis,
           method="rf",trControl=control,tuneGrid=rfgrid,
           linout = TRUE,ntree=1000, nodesize=20,replace=TRUE,
           importance=TRUE)
```

With this tuning, the optimal selected model (lowest RSME) had with 18 variables and R^2 of 0.40 (see Figure 49).

```
> rf
Random Forest

6624 samples
  32 predictor

No pre-processing
Resampling: Cross-Validated (4 fold, repeated 5 times)
Summary of sample sizes: 4968, 4968, 4968, 4968, 4968, 4968, ...
Resampling results across tuning parameters:

  mtry  RMSE      Rsquared   MAE
   5    7645.175  0.3941460  5316.765
   8    7566.748  0.4006557  5251.587
  10    7548.447  0.4008249  5234.756
  12    7535.279  0.4013651  5223.156
  15    7527.679  0.4005536  5213.576
  18    7522.438  0.4000053  5206.107
  20    7524.095  0.3989346  5206.050
  25    7528.189  0.3966855  5204.611
  30    7536.183  0.3942022  5205.753
  32    7541.414  0.3929346  5206.311

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 18.
```

Figure 49: Airbnb RF results

Afterward, we tested the need for sampling observations (model rf2). Therefore we run the tuning with the 18 variables, sampling 4000 observations. The RSME increases to 7568.638, and the R^2 decrease to 0.394712. Therefore we kept the previous setting without sampling.

```
> rf2$results
  mtry     RMSE Rsquared      MAE   RMSESD RsquaredSD    MAESD
1   18 7568.638 0.394712 5245.029 365.8413 0.03185049 95.87249
```

Figure 50: Airbnb RF2 results

We studied the need for early stopping. As we can see from the chart below (Figure 51), the Out of Bag Error (OBB) got stable before 500 iterations. This confirmed the need for early stopping.

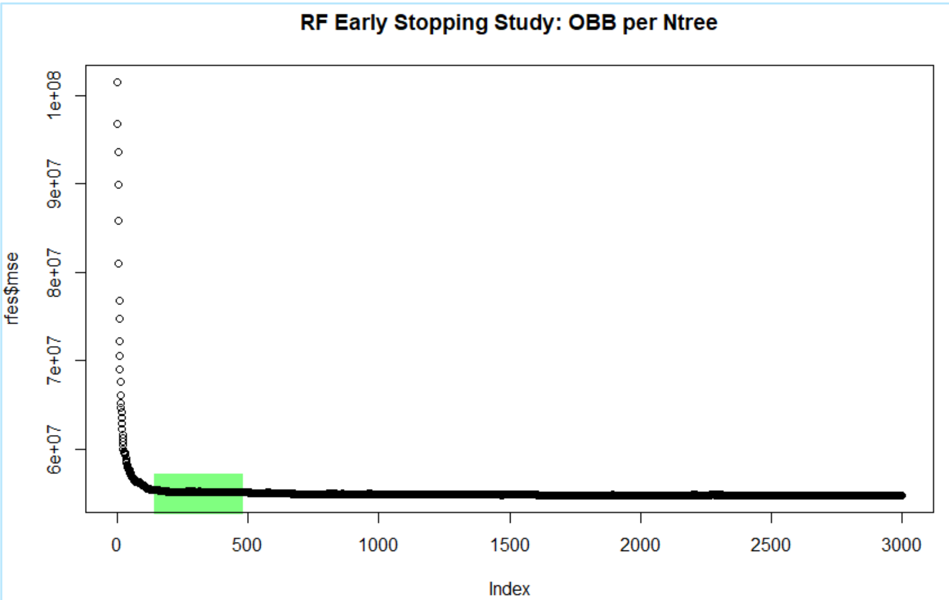


Figure 51: Airbnb RF Early Stopping Study

We changed the set up of the previous (RF) model, keeping 18 variables, 400 trees and without sampling observations. The R^2 increased to 0.431 (further information in Figure 52). Therefore we kept RF3 as our final Random Forest model.

```
> rf3$results
  mtry     RMSE  Rsquared     MAE   RMSESD RsquaredSD    MAESD
1   18 7505.147 0.4031016 5204.11 365.5542 0.03302365 84.77704
```

Figure 52: Airbnb RF3 results

Below we can see the variables importance ranking for this model. As we can see, the security deposit, cleaning fee, accommodates7+, and latitude play an essential role in this model.

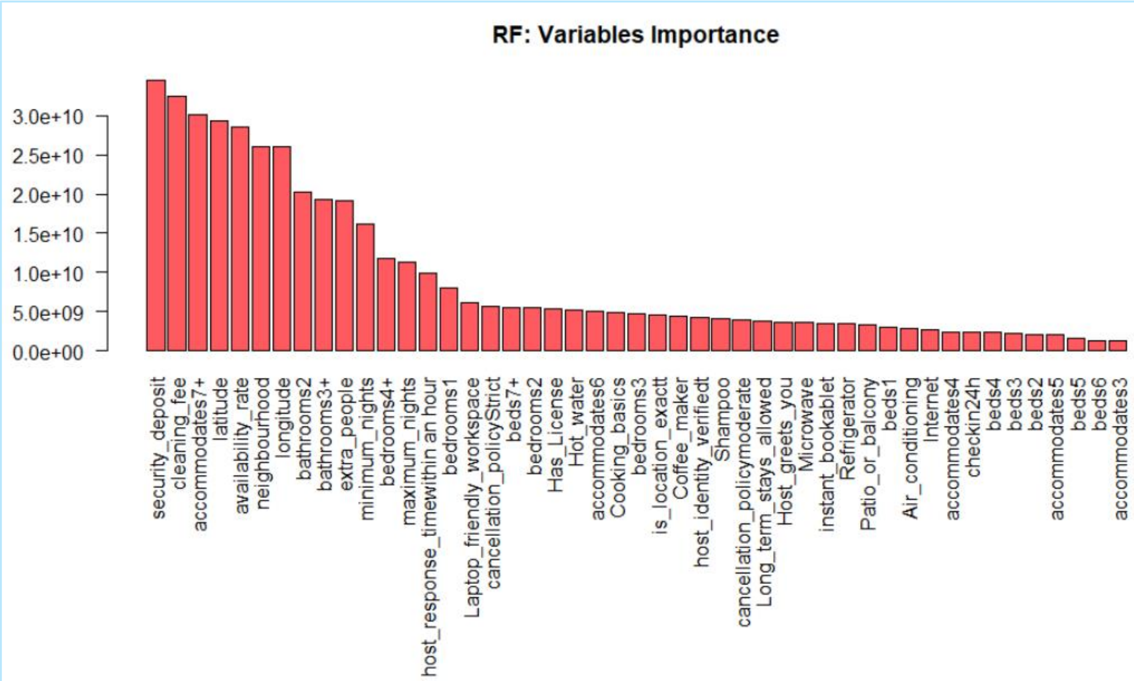


Figure 53: Airbnb RF3 Variable Importance

4.3.3. Gradient Boosting

For the tuning of the Gradient Boosting, we used the same strategy of the Idealista model, preparing a tuning grid with a wide range of parameters, from more aggressive to more conservative. Therefore, we set the range of shrinkage, from 0.001 to 0.2. The minimum number of observations per parameters is set from 5 and to 30. The number of trees from 100 to 5000.

```
gbmgrid<-expand.grid(shrinkage=c(0.1,0.05,0.03,0.01,0.001,0.2),
                     n.minobsinnode=c(5,10,20,30),
                     n.trees=c(100,300,500,1000,2000,5000),
                     interaction.depth=c(2))


gbm<- train(Yearly_Profit~.,data=airbnbbis,
            method="gbm",trControl=control,tuneGrid=gbmgrid,
            distribution="gaussian", bag.fraction=1,verbose=FALSE)
```

After running the 144 possibilities, R recommended us the model with 2000 trees, shrinkage of 0.1 and 30 observations per node. Due to the high shrinkage, we could say this model is aggressive, but at the same time, it is balanced by the high number of trees and nodes.

```
> gbm
Stochastic Gradient Boosting

6624 samples
  32 predictor

No pre-processing
Resampling: Cross-Validated (4 fold, repeated 5 times)
Summary of sample sizes: 4968, 4968, 4968, 4968, 4968, 4968, ...
Resampling results across tuning parameters:
```

shrinkage	n.minobsinnode	n.trees	RMSE	Rsquared	MAE
0.001	5	100	9529.445	0.1236142	6629.627
0.001	5	300	9352.709	0.1482367	6528.957
0.001	5	500	9206.586	0.1654877	6446.533
0.001	5	1000	8961.030	0.1895061	6299.681
0.001	5	2000	8682.947	0.2286164	6110.218
0.001	5	5000	8333.163	0.2713387	5836.339
0.001	10	100	9529.445	0.1236142	6629.627
0.001	10	300	9352.709	0.1482367	6528.957
0.001	10	500	9206.586	0.1654877	6446.533
0.001	10	1000	8960.891	0.1895394	6299.534
0.001	10	2000	8681.753	0.2288005	6109.329
0.001	10	5000	8332.740	0.2713870	5835.194
0.001	20	100	9529.445	0.1236142	6629.627
0.001	20	300	9352.531	0.1483079	6528.833
0.001	20	500	9206.195	0.1656307	6446.134
0.001	20	1000	8960.006	0.1897146	6298.691
0.001	20	2000	8680.886	0.2289935	6109.009
0.001	20	5000	8333.333	0.2708810	5833.774
0.001	30	100	9529.445	0.1236142	6629.627
0.001	30	300	9352.531	0.1483079	6528.833
0.001	30	500	9206.195	0.1656307	6446.134
0.001	30	1000	8960.006	0.1897146	6298.691
0.001	30	2000	8680.886	0.2289935	6109.009
0.001	30	5000	8333.339	0.2707859	5833.531
0.010	5	100	8959.605	0.1895892	6298.828
0.010	5	300	8516.483	0.2489640	5982.852
0.010	5	500	8331.945	0.2713471	5835.411
0.010	5	1000	8106.039	0.3037520	5645.869
0.010	5	2000	7915.436	0.3316048	5492.453
0.010	5	5000	7735.810	0.3577522	5370.295
0.010	10	100	8959.499	0.1895761	6298.750
0.010	10	300	8514.879	0.2492705	5981.752
0.010	10	500	8330.987	0.2715015	5834.053
0.010	10	1000	8106.096	0.3036840	5644.009
0.010	10	2000	7914.114	0.3316886	5489.033
0.010	10	5000	7736.495	0.3575777	5367.731
0.010	20	100	8958.684	0.1897093	6297.946
0.010	20	300	8513.879	0.2492947	5980.665
0.010	20	500	8332.163	0.2709169	5832.613
0.010	20	1000	8110.056	0.3026972	5643.539
0.010	20	2000	7918.019	0.3308901	5489.428
0.010	20	5000	7739.725	0.3570935	5365.495
0.010	30	100	8958.684	0.1897093	6297.946
0.010	30	300	8513.852	0.2492902	5980.636
0.010	30	500	8332.530	0.2707507	5832.727
0.010	30	1000	8109.668	0.3026150	5643.450
0.010	30	2000	7915.294	0.3312758	5487.686
0.050	30	100	8321.825	0.2722221	5827.505
0.050	30	300	7983.137	0.3214295	5541.112
0.050	30	500	7850.792	0.3408210	5444.811
0.050	30	1000	7721.265	0.3601266	5355.696
0.050	30	2000	7653.793	0.3709900	5303.243
0.050	30	5000	7643.429	0.3745233	5293.144
0.100	5	100	8089.334	0.3060268	5636.969
0.100	5	300	7806.217	0.3474417	5421.903
0.100	5	500	7722.802	0.3597507	5361.310
0.100	5	1000	7663.301	0.3692614	5313.903
0.100	5	2000	7658.024	0.3724112	5305.310
0.100	5	5000	7844.354	0.3537070	5418.638
0.100	10	100	8093.758	0.3049997	5636.462
0.100	10	300	7812.944	0.3460654	5422.475
0.100	10	500	7727.386	0.3589630	5362.465
0.100	10	1000	7666.708	0.3686712	5316.701
0.100	10	2000	7656.776	0.3720643	5305.924
0.100	10	5000	7811.140	0.3565903	5390.709
0.100	20	100	8092.969	0.3049064	5635.134
0.100	20	300	7811.672	0.3462694	5419.690
0.100	20	500	7730.892	0.3584786	5360.459
0.100	20	1000	7664.742	0.3692307	5308.435
0.100	20	2000	7643.848	0.3739250	5292.821
0.100	20	5000	7769.425	0.3617215	5361.403
0.100	30	100	8089.382	0.3057131	5633.463
0.100	30	300	7805.094	0.3473701	5415.518
0.100	30	500	7718.089	0.3605397	5353.485
0.100	30	1000	7651.676	0.3713361	5301.954
0.100	30	2000	7639.103	0.3746131	5293.577
0.100	30	5000	7730.888	0.3664741	5346.530
0.200	5	100	7892.440	0.3342757	5483.830
0.200	5	300	7704.988	0.3623794	5349.606
0.200	5	500	7670.012	0.3684730	5320.574
0.200	5	1000	7666.000	0.3717648	5310.961
0.200	5	2000	7793.274	0.3593134	5390.509
0.200	5	5000	8141.388	0.3270476	5630.895
0.200	10	100	7898.476	0.3329595	5483.004
0.200	10	300	7703.365	0.3626266	5348.277
0.200	10	500	7671.403	0.3680317	5321.438
0.200	10	1000	7662.259	0.3715318	5312.014
0.200	10	2000	7763.158	0.3617549	5372.254
0.200	10	5000	8090.991	0.3305616	5592.841
0.200	20	100	7897.863	0.3328357	5483.139
0.200	20	300	7705.761	0.3623577	5345.022
0.200	20	500	7667.390	0.3688735	5311.769
0.200	20	1000	7648.709	0.3735222	5296.759
0.200	20	2000	7739.110	0.3645691	5348.419
0.200	20	5000	8012.810	0.3381989	5532.205
0.200	30	100	7894.928	0.3332676	5479.728
0.200	30	300	7696.963	0.3638385	5339.223
0.200	30	500	7654.940	0.3708961	5304.627
0.200	30	1000	7641.711	0.3746141	5296.286
0.200	30	2000	7708.235	0.3687317	5332.795
0.200	30	5000	7941.391	0.3460612	5491.229

```
Tuning parameter 'interaction.depth' was held constant at a value of 2
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were n.trees = 2000, interaction.depth = 2, shrinkage = 0.1
 and n.minobsinnode = 30.
```

Figure 54: Airbnb GBM results

On the early stopping chart (Figure 55), we can see how stopping at 2000 is optimal since it is the lowerest RSME point. Therefore, we kept the GBM model.

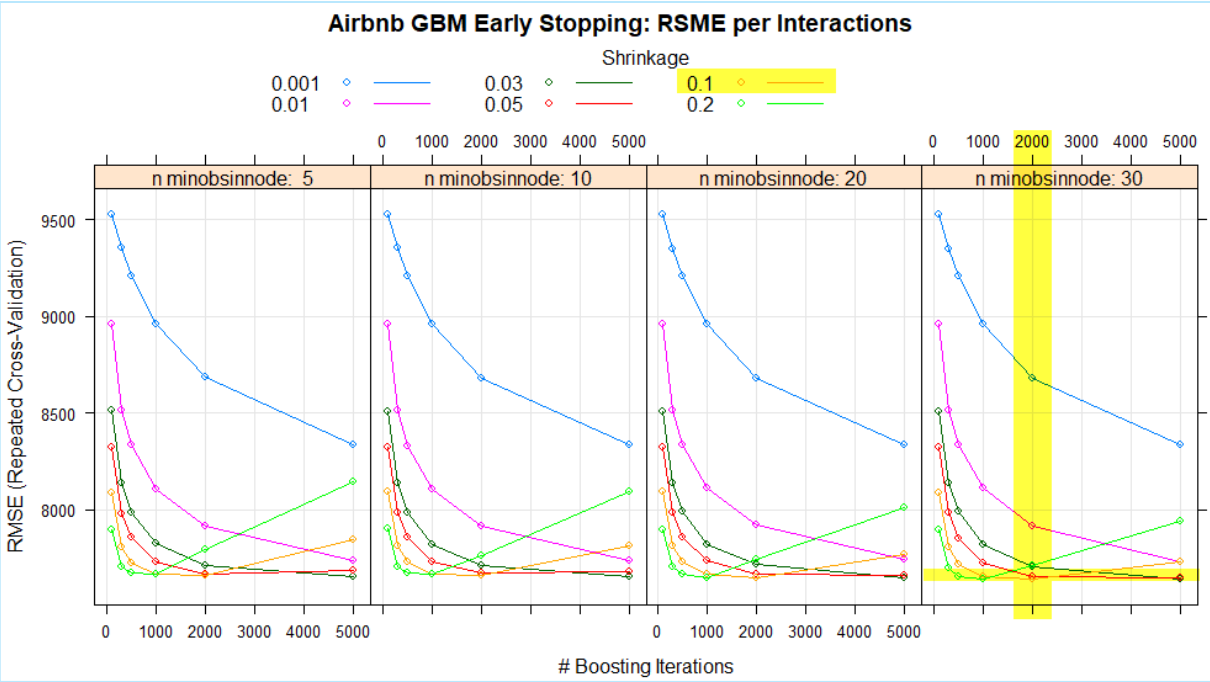


Figure 55: Airbnb GBM Early stopping

We also studied the possibility of sampling the observations (GBMr) by keeping all previous parameters constant and changing the bag fraction to 0.6. The R^2 increased to 0.38 (Figure 56), thus we selected GBMr as the final Gradient Boosting model.

```
> gbmr$results
  shrinkage n.minobsinnode n.trees interaction.depth     RMSE  Rsquared      MAE   RMSESD RsquaredSD
1       0.1             30    2000                 2 7604.901 0.3823069 5270.254 245.8813  0.0230782
     MAESD
1 39.03997
```

Figure 56: Airbnb GBMr results

Finally, we examine the variables importance of GBMr, as we can see in Figure 57, again, security deposit, cleaning fee, accommodates7+, and longitude play an important role. The top 5 variables are very similar to the Random Forest model.

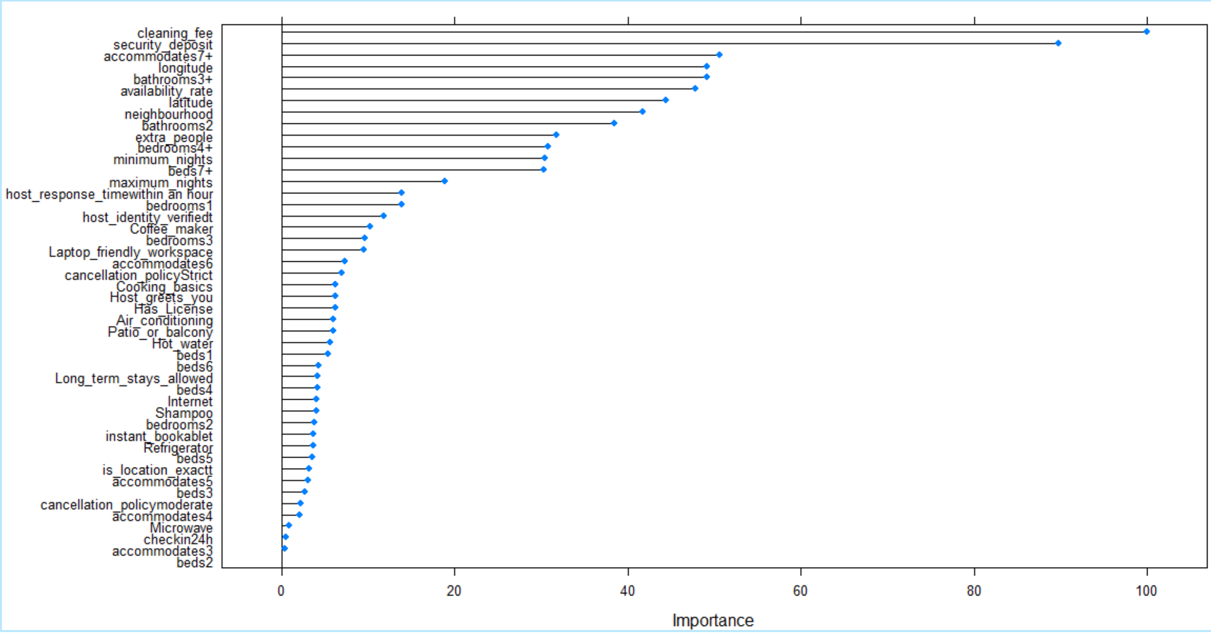


Figure 57: Airbnb GBMr Variables Importance

4.3.4. Extreme Gradient Boosting

The tune grid of the Extreme Gradient Boost was also wide, with aggressive and moderate parameters. We set the learning rate (eta) between 0.001 and 0.5, the number of iterations from 100 to 5000, the coefficient of regularization, gamma, from 0 to 1. We decided not to sample variables and observation at first and to keep the alpha and lambda (both additional penalization parameters) as 1 and 0, respectively because in this grid there was already 1120 model to run.

```
xgbmgrid <-expand.grid(min_child_weight=20,
                       eta=c(0.001,0.01,0.03,0.05,0.1,0.2,0.3,0.5),
                       nrounds=c(100,300,500,1000,2000,4000,5000),
                       max_depth=c(1,2,4,6),
                       gamma=c(0,0.001,0.01,0.1,1),
                       colsample_bytree=1,
                       subsample=1)

xgbm1<- train(Yearly_Profit~.,data=airbnbbis,
              method="xgbTree",trControl=control,
              tuneGrid=xgbmgrid,objective = "reg:linear",verbose=FALSE,
              alpha=1,lambda=0)
```

After running the 1120 models possibility, the optimal values used for the XGBM model were nrounds=100, max_depth=6, eta=0.1, gamma=0, colsample_bytree=1, min_child_weight=20, leading to an R^2 of 0,4083, we can see a sample of the models and further results in Figure 58.

```
0.100  4  1.000  2000  7814.870  0.36464098  5387.296
0.100  4  1.000  4000  7969.456  0.35118131  5526.809
0.100  4  1.000  5000  8016.716  0.34724006  5572.031
0.100  6  0.000   100  7407.593  0.40830031  5090.429
0.100  6  0.000   300  7470.719  0.40296010  5099.870
0.100  6  0.000   500  7534.156  0.39642302  5151.397
0.100  6  0.000  1000  7660.559  0.38314495  5251.739
0.100  6  0.000  2000  7755.495  0.37351357  5343.506
0.100  6  0.000  4000  7792.001  0.37010477  5384.026
0.100  6  0.000  5000  7797.066  0.36961561  5389.502
0.100  6  0.001   100  7407.593  0.40830031  5090.429
0.100  6  0.001   300  7470.719  0.40296010  5099.870
0.100  6  0.001   500  7534.156  0.39642302  5151.397
0.100  6  0.001  1000  7660.559  0.38314495  5251.739
0.100  6  0.001  2000  7755.495  0.37351357  5343.506
0.100  6  0.001  4000  7792.001  0.37010477  5384.026
0.100  6  0.001  5000  7797.066  0.36961561  5389.502
0.100  6  0.010   100  7407.593  0.40830031  5090.429
0.100  6  0.010   300  7470.719  0.40296010  5099.870
0.100  6  0.010   500  7534.156  0.39642302  5151.397
0.100  6  0.010  1000  7660.559  0.38314495  5251.739
0.100  6  0.010  2000  7755.495  0.37351357  5343.506
0.100  6  0.010  4000  7792.001  0.37010477  5384.026
0.100  6  0.010  5000  7797.066  0.36961561  5389.502
0.100  6  0.100   100  7407.593  0.40830031  5090.429
0.100  6  0.100   300  7470.719  0.40296010  5099.870
0.100  6  0.100   500  7534.156  0.39642302  5151.397
0.100  6  0.100  1000  7660.559  0.38314495  5251.739
0.100  6  0.100  2000  7755.495  0.37351357  5343.506
0.100  6  0.100  4000  7792.001  0.37010477  5384.026
0.100  6  0.100  5000  7797.066  0.36961561  5389.502
0.100  6  1.000   100  7407.593  0.40830031  5090.429
0.100  6  1.000   300  7470.719  0.40296010  5099.870
0.100  6  1.000   500  7534.156  0.39642302  5151.397
0.100  6  1.000  1000  7660.559  0.38314495  5251.739
0.100  6  1.000  2000  7755.495  0.37351357  5343.506
0.100  6  1.000  4000  7792.001  0.37010477  5384.026
0.100  6  1.000  5000  7797.066  0.36961561  5389.502
0.200  1  0.000   100  8168.389  0.28663777  5681.338
0.200  1  0.000   300  7989.762  0.31284820  5564.505
0.200  1  0.000   500  7935.612  0.32144393  5542.938
0.200  1  0.000  1000  7886.806  0.32959963  5518.627
0.200  1  0.000  2000  7860.222  0.33434244  5502.896
0.200  1  0.000  4000  7851.005  0.33617195  5495.425
0.200  1  0.000  5000  7851.700  0.33627145  5493.767
0.200  1  0.001   100  8168.389  0.28663777  5681.338
0.200  1  0.001   300  7989.762  0.31284820  5564.505
0.200  1  0.001   500  7935.612  0.32144393  5542.938
0.200  1  0.001  1000  7886.806  0.32959963  5518.627
0.200  1  0.001  2000  7860.222  0.33434244  5502.896
0.200  1  0.001  4000  7851.005  0.33617195  5495.425
0.200  1  0.001  5000  7851.700  0.33627145  5493.767
0.200  1  0.010   100  8168.389  0.28663777  5681.338
0.200  1  0.010   300  7989.762  0.31284820  5564.505
0.200  1  0.010   500  7935.612  0.32144393  5542.938
0.200  1  0.010  1000  7886.806  0.32959963  5518.627
0.200  1  0.010  2000  7860.222  0.33434244  5502.896
0.200  1  0.010  4000  7851.005  0.33617195  5495.425
0.200  1  0.010  5000  7851.700  0.33627145  5493.767
```

```
0.500  4  0.100   100  7959.153  0.34207614  5490.969
0.500  4  0.100   300  8234.911  0.32176491  5716.431
0.500  4  0.100   500  8340.870  0.31563417  5803.400
0.500  4  0.100  1000  8452.779  0.30814371  5899.823
0.500  4  0.100  2000  8529.402  0.30315022  5972.828
0.500  4  0.100  4000  8547.252  0.30212332  5990.145
0.500  4  0.100  5000  8548.282  0.30206433  5991.262
0.500  4  1.000   100  7959.153  0.34207614  5490.969
0.500  4  1.000   300  8234.911  0.32176491  5716.431
0.500  4  1.000   500  8340.870  0.31563417  5803.400
0.500  4  1.000  1000  8452.779  0.30814371  5899.823
0.500  4  1.000  2000  8529.402  0.30315022  5972.828
0.500  4  1.000  4000  8547.252  0.30212332  5990.145
0.500  4  1.000  5000  8548.282  0.30206433  5991.262
0.500  6  0.000   100  8144.627  0.32889368  5639.352
0.500  6  0.000   300  8345.694  0.31488128  5828.064
0.500  6  0.000   500  8375.173  0.31288928  5852.088
0.500  6  0.000  1000  8386.806  0.31229084  5865.289
0.500  6  0.000  2000  8387.849  0.31220883  5866.492
0.500  6  0.000  4000  8387.881  0.31220601  5866.531
0.500  6  0.000  5000  8387.881  0.31220601  5866.531
0.500  6  0.001   100  8144.627  0.32889368  5639.352
0.500  6  0.001   300  8345.694  0.31488128  5828.064
0.500  6  0.001   500  8375.173  0.31288928  5852.088
0.500  6  0.001  1000  8386.806  0.31229084  5865.289
0.500  6  0.001  2000  8387.849  0.31220883  5866.492
0.500  6  0.001  4000  8387.884  0.31220579  5866.534
0.500  6  0.001  5000  8387.884  0.31220579  5866.534
0.500  6  0.010   100  8144.627  0.32889368  5639.352
0.500  6  0.010   300  8345.694  0.31488128  5828.064
0.500  6  0.010   500  8375.173  0.31288928  5852.088
0.500  6  0.010  1000  8386.806  0.31229084  5865.289
0.500  6  0.010  2000  8387.848  0.31220878  5866.490
0.500  6  0.010  4000  8387.881  0.31220539  5866.533
0.500  6  0.010  5000  8387.881  0.31220539  5866.533
0.500  6  0.100   100  8144.627  0.32889368  5639.352
0.500  6  0.100   300  8345.694  0.31488128  5828.064
0.500  6  0.100   500  8375.173  0.31288928  5852.088
0.500  6  0.100  1000  8386.806  0.31229084  5865.289
0.500  6  0.100  2000  8387.802  0.31221498  5866.460
0.500  6  0.100  4000  8387.833  0.31221314  5866.501
0.500  6  0.100  5000  8387.833  0.31221314  5866.501
0.500  6  1.000   100  8144.627  0.32889368  5639.352
0.500  6  1.000   300  8345.694  0.31488128  5828.064
0.500  6  1.000   500  8375.173  0.31288928  5852.088
0.500  6  1.000  1000  8386.806  0.31229084  5865.289
0.500  6  1.000  2000  8387.844  0.31220890  5866.605
0.500  6  1.000  4000  8387.854  0.31220862  5866.615
0.500  6  1.000  5000  8387.854  0.31220862  5866.615

Tuning parameter 'colsample_bytree' was held constant at a value of 1
Tuning
 parameter 'min_child_weight' was held constant at a value of 20
Tuning parameter 'subsample'
 was held constant at a value of 1
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were nrounds = 100, max_depth = 6, eta = 0.1, gamma =
 0, colsample_bytree = 1, min_child_weight = 20 and subsample = 1.
```

Figure 58: Airbnb XGBM results

Regarding the variable importance for the XGBM, we can see in Figure 59 that the top 5 variables are similar to the previous models, but different relative importance.

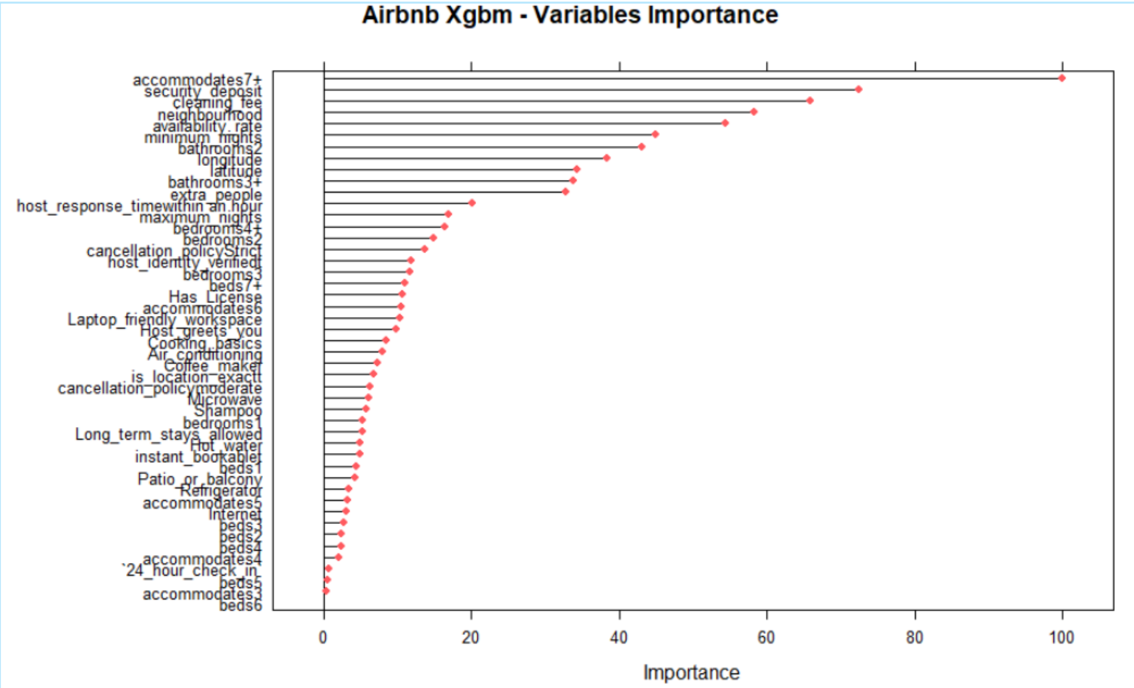


Figure 59: Airbnb XGBM Variables Importance

4.3.5. Support Vector Machine

Finally, we came to our last model, the Support Vector Machine. We trained again linear and radial models.

4.3.5.1. Linear

We tuned the linear SVM model by varying the penalty factor C between 0.01 to 10.

```
SVMgridl<-expand.grid(C=c(0.01,0.05,0.1,0.2,0.5,1,2,5,10))

SVMl<- train(data=airbnbbis,Yearly_Profit~.,
             method="svmLinear",trControl=control,
             tuneGrid=SVMgridl,verbose=FALSE)
```

The best model, in Figure 60, had C parameter = 2 and R^2 = 0.28.

```
> SVMl
Support Vector Machines with Linear Kernel

6624 samples
  32 predictor

No pre-processing
Resampling: Cross-Validated (4 fold, repeated 5 times)
Summary of sample sizes: 4968, 4968, 4968, 4968, 4968, 4968, ...
Resampling results across tuning parameters:

  C      RMSE      Rsquared   MAE
   0.01  8191.242  0.2873267  5575.943
   0.05  8187.790  0.2872191  5577.811
   0.10  8187.411  0.2872136  5578.238
   0.20  8187.392  0.2871677  5578.571
   0.50  8187.465  0.2871142  5578.832
   1.00  8187.333  0.2871227  5578.800
   2.00  8187.234  0.2871540  5578.763
   5.00  8187.274  0.2871337  5578.866
  10.00  8187.473  0.2870913  5578.955

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was C = 2.
```

Figure 60: Airbnb SVML results

4.3.5.2. Radial

For the Radial SVM, we kept the same range of penalty parameters, from 0.01 to 10 and varied the sigma from 0.1 to 5.

```
SVMrgrid<-expand.grid(C=c(0.01,0.05,0.1,0.2,0.5,1,2,5),
                      sigma=c(0.01,0.05,0.1,0.2,0.5,1,2,5))


SVMr<- train(data=idealistabis,Yearly_Price~.,
             method="svmRadial",trControl=control,
             tuneGrid=SVMrgrid,verbose=FALSE)
```

Our best model also had C=2 and sigma of 0.01. This model had an R^2 of 0.35, as we can see in Figure 61.

```
> SVMr
Support Vector Machines with Radial Basis Function Kernel

6624 samples
  32 predictor

No pre-processing
Resampling: Cross-Validated (4 fold, repeated 5 times)
Summary of sample sizes: 4968, 4968, 4968, 4968, 4968, 4968, ...
Resampling results across tuning parameters:
```

C	sigma	RMSE	Rsquared	MAE
0.01	0.01	8985.125	0.22956130	5930.480
0.01	0.05	9497.789	0.13674090	6253.370
0.01	0.10	9722.236	0.08796462	6463.504
0.01	0.20	9774.498	0.04806663	6522.008
0.01	0.50	9780.841	0.02077725	6531.272
0.01	1.00	9782.277	0.01702091	6532.877
0.01	2.00	9783.130	0.01610856	6533.924
0.01	5.00	9784.023	0.01484176	6534.936
0.05	0.01	8442.193	0.27812325	5594.518
0.05	0.05	9046.372	0.19506468	5858.276
0.05	0.10	9537.422	0.12384802	6261.955
0.05	0.20	9731.697	0.05485275	6468.407
0.05	0.50	9766.482	0.02293180	6510.620
0.05	1.00	9773.063	0.01848547	6518.081
0.05	2.00	9777.147	0.01732049	6522.778
0.05	5.00	9781.898	0.01572333	6527.430
0.10	0.01	8241.375	0.30090853	5481.352
0.10	0.05	8786.475	0.23263164	5661.821
0.10	0.10	9370.245	0.14665853	6101.633
0.10	0.20	9681.476	0.06503475	6413.862
0.10	0.50	9743.939	0.02637869	6489.321
0.10	1.00	9755.692	0.02051090	6502.473
0.10	2.00	9762.790	0.01884453	6510.818
0.10	5.00	9771.049	0.01665768	6519.045
0.20	0.01	8074.526	0.32008100	5380.858
0.20	0.05	8501.227	0.27170475	5473.423
0.20	0.10	9143.874	0.17724637	5906.010
0.20	0.20	9594.197	0.08179568	6332.897
0.20	0.50	9705.284	0.03442907	6459.383
0.20	1.00	9725.757	0.02598883	6479.918
0.20	2.00	9737.846	0.02253468	6492.506
0.20	5.00	9752.031	0.01876168	6505.490
0.50	0.01	7909.728	0.33959662	5282.774
0.50	0.05	8129.129	0.31511790	5286.906
0.50	0.10	8785.870	0.21775606	5656.682
0.50	0.20	9405.364	0.10343006	6182.930
0.50	0.50	9619.867	0.04477423	6401.943
0.50	1.00	9659.182	0.03349745	6438.393
0.50	2.00	9680.187	0.02823974	6458.125
0.50	5.00	9704.917	0.02280885	6479.337
1.00	0.01	7832.762	0.34828591	5239.681
1.00	0.05	7913.816	0.33658431	5224.760
1.00	0.10	8512.908	0.24650543	5575.192
1.00	0.20	9197.293	0.11837518	6111.624
1.00	0.50	9482.188	0.05051886	6370.925
1.00	1.00	9535.421	0.03817515	6417.345
1.00	2.00	9562.580	0.03226280	6443.747
1.00	5.00	9592.837	0.02645198	6471.399
2.00	0.01	7811.294	0.35001533	5231.136
2.00	0.05	7833.022	0.34182281	5260.687
2.00	0.10	8321.985	0.26705509	5580.811
2.00	0.20	9057.460	0.13014891	6148.183
2.00	0.50	9387.495	0.05499308	6430.874
2.00	1.00	9445.161	0.04247064	6479.922
2.00	2.00	9472.908	0.03681231	6506.283
2.00	5.00	9505.686	0.03051084	6536.200
5.00	0.01	7864.545	0.34414228	5274.847
5.00	0.05	7870.879	0.33645732	5377.738
5.00	0.10	8262.285	0.27191657	5627.449
5.00	0.20	9021.522	0.13231967	6205.721
5.00	0.50	9367.880	0.05688324	6497.968
5.00	1.00	9426.246	0.04478417	6546.139
5.00	2.00	9451.677	0.03968678	6571.592
5.00	5.00	9486.158	0.03297920	6603.926

```
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were sigma = 0.01 and C = 2.
```

Figure 61: Airbnb SVMR results

4.3.6. Models Assessment

With the 6 winning models prepared, we run a model competition with cross-validation of 4 groups and 20 repetitions. As we can see in the box-plot (Figure 62), the XGBM model has the lowest RSME and the highest R^2 of 0.40. Besides the XGboosting, the Random Forest also performed well with an R^2 of 0.39 and a smaller variability than the XGBM model.

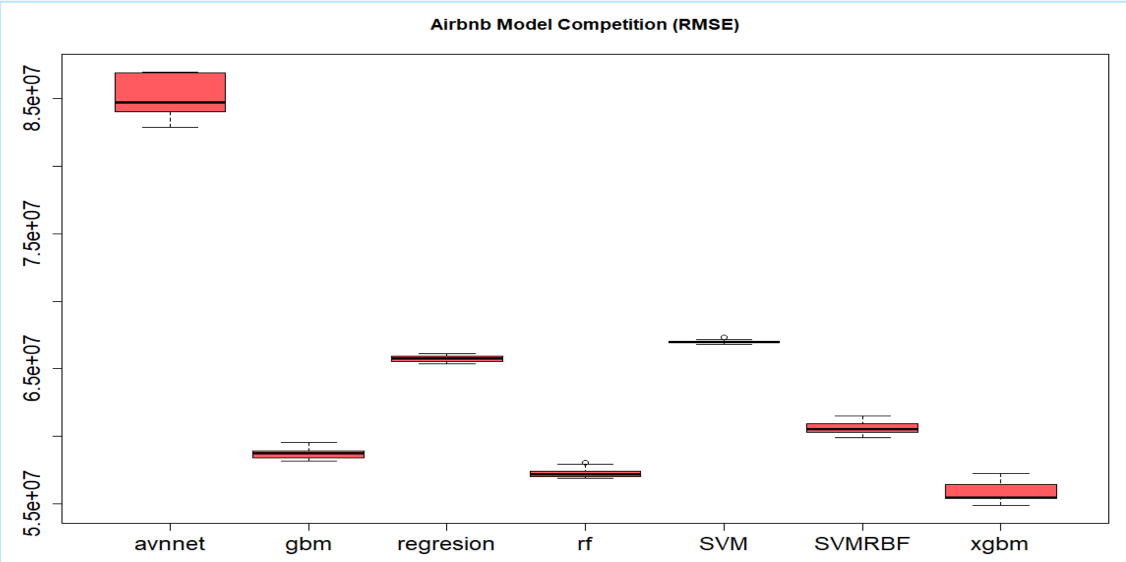


Figure 62: Airbnb Model Assesment

4.3.7. Ensemble

With the four best models in order to attempt to reduce the variability of our models and an increase in R^2. With the previously saved predictions from each model, we combined and took the mean of them. As we can see below, we made 3 ensembled models by combining our four better models (Xgbm> rf >gbm>SVMRadial) in different groups.

```
unipredi$predi10<-(unipredi$rf+unipredi$xgbm)/2
unipredi$predi11<-(unipredi$rf+unipredi$gbm+unipredi$xgbm)/3
unipredi$predi12<-(unipredi$gbm+unipredi$rf+unipredi$xgbm+unipredi$SVMRBF)/4
```

In Figure 63, we can see the results of the ensemble models compared with the original models. In all ensemble models, we got better results, with lower RSME and variability (Figure 64). The best ensemble model is *predi12*, with RMSE of 54684826 and R^2 of 0.4123217.

	modelo	r2	error
1	gbm	0.3692390	58693774
2	predi10	0.4070643	55174043
3	predi11	0.4093477	54961566
4	predi12	0.4123217	54684826
5	rf	0.3844497	57278383
6	SVMRBF	0.3485240	60621349
7	xgbm	0.4005581	55779458

Figure 63: Airbnb Idealista Final Model Assesment (R2 and RSME)

Although we got better results with all ensemble models, in both RSME and variability (Figure 64), we could not get a relevant increase on the R^2 after exhausting all attempts models and tuning possibilities. We believe the low 0.41 R^2 in this model compared to the 0.9 of Idealista is related to the several estimations we need to perform to calculate the target variable, which could affect the relationship between input and the target variable. Another possibility that could justify such difference is the volatility of the short-term rentals compared to the long-term. Despite the low indexes, we use *predi12* to predict vacation rentals in the next chapter.

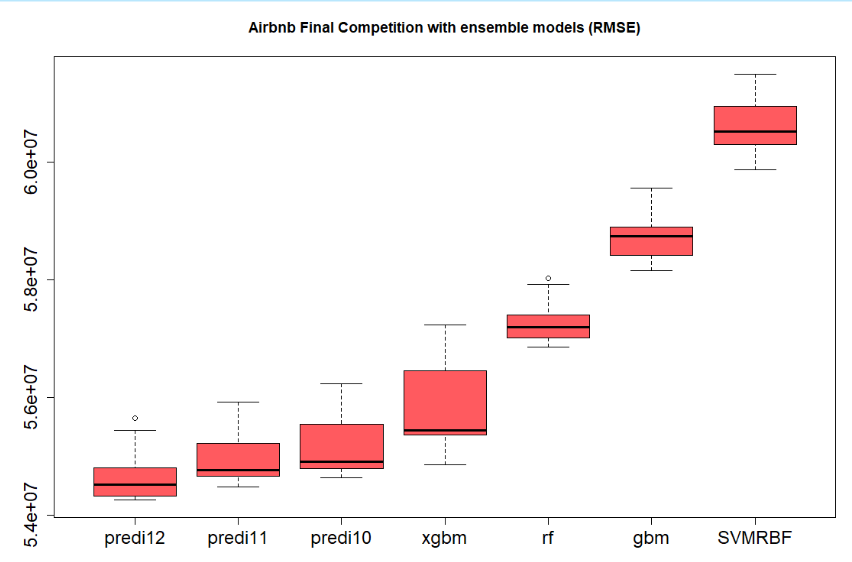


Figure 64: Airbnb Final Model Assesment (Boxplot)

5. MULTICHANNEL RENT PREDICTION & ROI CALCULATION

Lastly, with our winner model for both rental channels, *predi12*, we were able to calculate the rent predictions for the dataset of houses on sale in Idealista. To get this data, we used the same python code and credentials for the rent data, the only change in the code was on the field operation, which we changed to 'sale'.

In order to run the predictions, beforehand we needed to execute the same modifications we implemented on the two train model using Enterprise Miner. We performed the level aggrupation to the class variables *rooms*, *bedrooms*, *floors*, and *neighborhood*. Then, we applied a filter for *size* (>220m^2) and *sale price* (>600.000€), we decided to apply these filters to be more aligned to the reality of a small/medium investor. After this filter, we ended with 296 observations.

However, since the test dataset was from Idealista, there were some variables specific from Airbnb model missing. Therefore, we needed to manually add these Airbnb variables that did not exist on the Idealista dataset.

Once we have a running application, these variables should be inserted by the user, which means they are personalizable and adjustable. They relate to the amenities the host could offer to the visitor, the fees they could charge, the availability and other "house rules". Since not all future host will know all these details beforehand, we also set some default values, which are the ones we are using on the prediction (Figure 65). For this case, we imagined a "flexible" and "available" investor profile:

- For the fees, we used a random sample of the original Airbnb database. We sought to have variability and a realistic database and not only the same value for everyone.
- For the minimum, maximum nights and availability rate, we made a random range, but limited, looking for the variability, but within the limit of being "flexible". For the minimum of nights, the limit varies from 1 to 3 nights; for the maximum of nights, we set the limit between 28 and 30 days (considering these are for a vacation lodge); for the availability rate, we assumed the investor would be almost always available, since their main goal by investing is increasing profitability, thus an availability rate above 90%.
- The remaining binary variables of amenities, we defined them if the house had all of them.

Figure 65 shows all variables we added to the Airbnb rent prediction model.

```
#3) Add Variables
abpredi$extra_people<-sample(airbnb$extra_people,296)
abpredi$cleaning_fee<-sample(airbnb$cleaning_fee,296)
abpredi$security_deposit<-sample(airbnb$security_deposit,296)
abpredi$minimum_nights<-sample(1:3,replace = TRUE,296)
abpredi$maximum_nights<-sample(28:31,replace = TRUE,296)
abpredi$availability_rate<-runif(296,0.9,0.98)
abpredi$cancellation_policy<-"flexible"
abpredi$host_response_time<-"within an hour"
abpredi$host_identity_verified<-"t"
abpredi$instant_bookable<-"t"
abpredi$is_location_exact<-"t"
abpredi$Hot_water<-1
abpredi$Internet<-1
abpredi$checkin24h<-1
abpredi$Coffee_maker<-1
abpredi$Host_greets_you<-1
abpredi$Has_License<-1
abpredi$Shampoo<-1
abpredi$Laptop_friendly_workspace<-1
abpredi$Cooking_basics<-1
abpredi$Microwave<-1
abpredi$Refrigerator<-1
abpredi$"24_hour_check_in"<-1
abpredi$Long_term_stays_allowed<-0
```

Figure 65: Airbnb Prediction Added Variables to the Airbnb Prediction Model

With both data ready to execute the predictions in R, using the function *predict* we individually run the predictions for XGBM, GBM, RF, and SVMR, and calculate the mean of them to get the predictions of *predi12*.

```
####PREDICTIONS########

#unipredi$predi12<-(unipredi$gbm+unipredi$rf+unipredi$xgbm+unipredi$SVMRBF)/4

predictideal<-predict(xgbm,idpredibis)
predictideal1<-predict(gbm,idpredibis)
predictideal2<-predict(rf,idpredibis)
predictideal3<-predict(SVMr,idpredibis)

predi12ideal<-(predictideal+predictideal1+predictideal2+predictideal3)/4

idpredi$rentpredictions<-predi12ideal
```

At long last, with that last procedure, we had the actual sales price and the predictions for both vacation and traditional rents. Therefore, we were able to calculate the ROI and ROIM for both investments strategy. Below we can see the formula of both indicators which we applied to the properties on sale dataset in R. Although ROI is a fixed formula, the idea behind the ROIM is to give the investor the possibility to personalize the interest (*i*), downpayment (*dp*) and installments (*t*) according to their financing capacities. Therefore we inserted the amounts below as default amounts.

```
#########ROI###############

#ROI in cash

idpredi$ROIC<-(idpredi$rentpredictions/idpredi$price)

#ROI in Mortage

i<-0.0225
dp<-0.2
pr<-idpredi$price
t<-30

totalinvest<-(pr+(pr*t*i)-(pr*dp))

idpredi$ROIM<-(idpredi$rentpredictions/(totalinvest))
```

Once we develop the final Rentalbility platform, all these calculations would be running on its background. The client view would be a colored map which indicates the rental channel where the ROI and ROIC are the best, their values and more specific analytics. In Chapter 7, we will provide an illustrated example of all those data and predictions applied in a data visualization tool.

6. OCCUPANCY RATE STUDY

As a way to increment and support the Airbnb model, we made a short study with the same dataset to understand Airbnb's occupancy rate behavior. We sought to predict if a house would be often occupied or not. The target variable, *occu_bi,* is a boolean variable, which takes 1 for highly occupied houses and 0 otherwise. This variable was calculated based on the occupancy rate of each property, it goes from 1% to 70%. The highly occupied houses present more than 50% occupancy rate on a year. We defined this range to obtain an equal and meaningful frequency for both categories. We trained four different types of models using SAS 9.4 base for this study. We selected the 20 most important variables from the 30 of the previous model.

6.1. Neural Networks

We trained models with 5,10,15,20 and 25 units for both *Levmar* and *Backpropagation* algorithms. For this purpose, we used the macros provided in the Machine Learning classes by Portela (2019) *Variar* and *neuralbinariabasica,* which only uses train data.

```
%macro variar(seminicio=,semifin=,inicionodos=,finalnodos=,increnodos=);
title '';
data union;run;
%do semilla=&seminicio %to &semifin;
%do nodos=&inicionodos %to &finalnodos %by &increnodos;
   %neuralbinariabasica(archivo=airbnb,
   listconti=extra_people minimum_nights,
   listclass=Has_License Shampoo Host_greets_you host_response_til cleaning_fee2
             cleaning_fee4 security_deposit1 maximum_nights2 availability_rate2
             availability_rate4 latitude2 latitude4 longitude2
             minimum_nights2 cancellation_policy2 neighbourhood_cleansed46
             neighbourhood_cleansed107 neighbourhood_group_clea10,vardep=Occu_BI,nodos=&nodos,corte=50,semilla=&semilla,porcen=0.80,algo=levmar);
   data estadisticos;set estadisticos;nodos=&nodos;semilla=&semilla;run;
   data union;set union estadisticos;run;
%end;
%end;
proc sort data=union;by nodos;run;
proc boxplot data=union;plot (porcenVN porcenFN porcenVP porcenFP
sensi especif tasafallos tasaciertos precision F_M)*nodos;run;
%mend;

%variar(seminicio=12345,semifin=12355,inicionodos=5,finalnodos=25,increnodos=5);
```

In the boxplot (Figure 66) we can see the accuracy rate of each of the Levmar mode. The best one had 10 hidden layers.

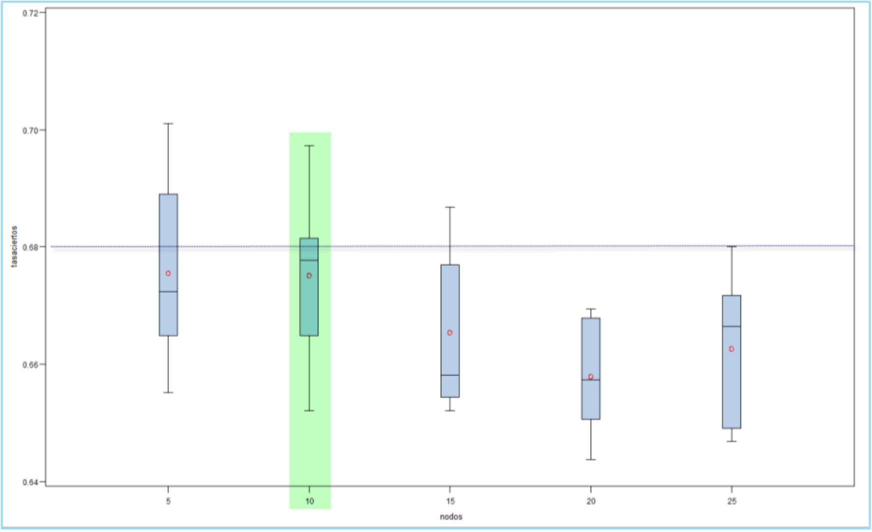


Figure 66: Occupancy Rate NN Levmar (Accuracy Rate boxplot)

Then trained the model with the Backprop optimization, with momentum = 0.2, learning rate = 0.1 and Tanh function.

```
%macro variar(seminicio=,semifin=,inicionodos=,finalnodos=,increnodos=);
title '';
data union;run;
%do semilla=&seminicio %to &semifin;
%do nodos=&inicionodos %to &finalnodos %by &increnodos;
   %neuralbinariabasica(archivo=airbnb,
   listconti=extra_people minimum_nights,
   listclass=Has_License Shampoo Host_greets_you host_response_til cleaning_fee2
               cleaning_fee4 security_deposit1 maximum_nights2 availability_rate2
               availability_rate4 latitude2 latitude4 longitude2
               minimum_nights2 cancellation_policy2 neighbourhood_cleansed46
               neighbourhood_cleansed107 neighbourhood_group_clea10,vardep=Occu_BI,nodos=&nodos,corte=50,semilla=&semilla,porcen=0.80,algo=bprop mom=0.2 learn=0.1);
   data estadisticos;set estadisticos;nodos=&nodos;semilla=&semilla;run;
   data union;set union estadisticos;run;
%end;
%end;
proc sort data=union;by nodos;run;
proc boxplot data=union;plot (porcenVN porcenFN porcenVP porcenFP
sensi especif tasafallos tasaciertos precision F_M)*nodos;run;
%mend;

%variar(seminicio=12345,semifin=12355,inicionodos=5,finalnodos=25,increnodos=5);
```

These models with Backprop algorithm and 10 and 15 hidden layers performed better than the previous optimization algorithm as we can see in Figure 67.

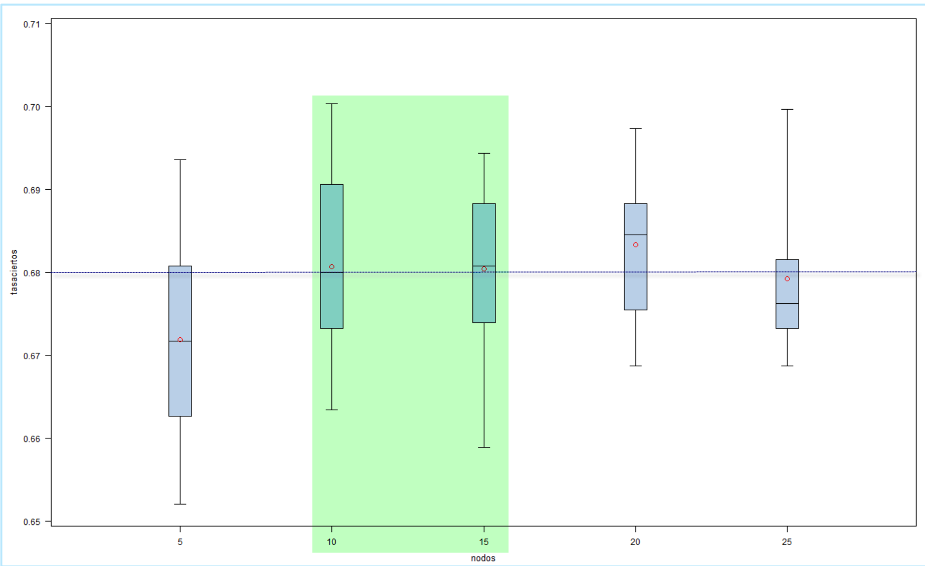


Figure 67: Occupancy Rate NN Models Backprop (Accuracy Rate boxplot)

We decided to keep the backpropagation models and observe the need for Early Stopping for this model with the macro *redneuralbinaria*.

```
%redneuronalbinaria(archivo=airbnb,listclass=Has_License Shampoo Host_greets_you host_response_til cleaning_fee2
              cleaning_fee4 security_deposit1 maximum_nights2 availability_rate2
              availability_rate4 latitude2 latitude4 longitude2
              minimum_nights2 cancellation_policy2 neighbourhood_cleansed46
              neighbourhood_cleansed107 neighbourhood_group_clea10,
listconti=extra_people minimum_nights,
vardep=Occu_BI,porcen=0.80,semilla=442711,ocultos=10,meto=bprop mom=0.2 learn=0.1,acti=TANH);

%redneuronalbinaria(archivo=airbnb,listclass=Has_License Shampoo Host_greets_you host_response_til cleaning_fee2
              cleaning_fee4 security_deposit1 maximum_nights2 availability_rate2
              availability_rate4 latitude2 latitude4 longitude2
              minimum_nights2 cancellation_policy2 neighbourhood_cleansed46
              neighbourhood_cleansed107 neighbourhood_group_clea10,
listconti=extra_people minimum_nights,
vardep=Occu_BI,porcen=0.80,semilla=442711,ocultos=15,meto=bprop mom=0.2 learn=0.1,acti=TANH);
```

In the case of the model with 10 hidden units (Figure 68), the macro recommended stopping at 30, meanwhile, for 15 hidden units (Figure 69) it recommended stopping at 32. However, when we look at both charts, it seems that in none of the cases the Early Stopping is not needed.

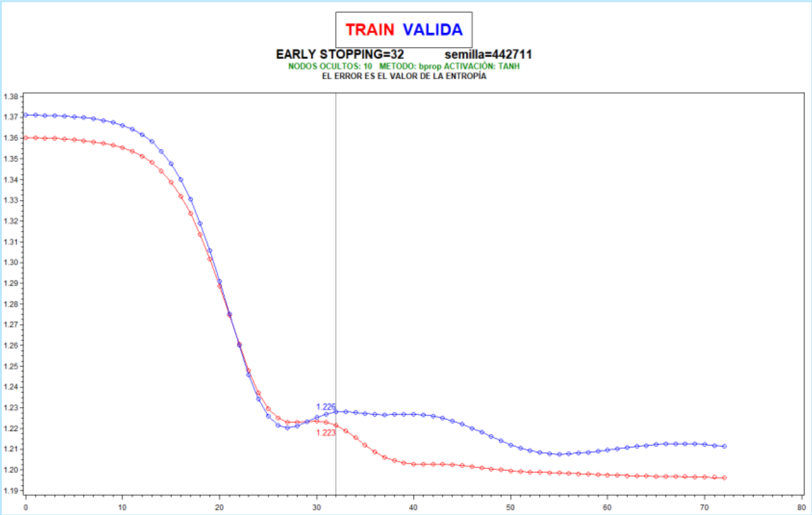


Figure 68: Occupancy Rate NN 10 hidden units Early Stopping

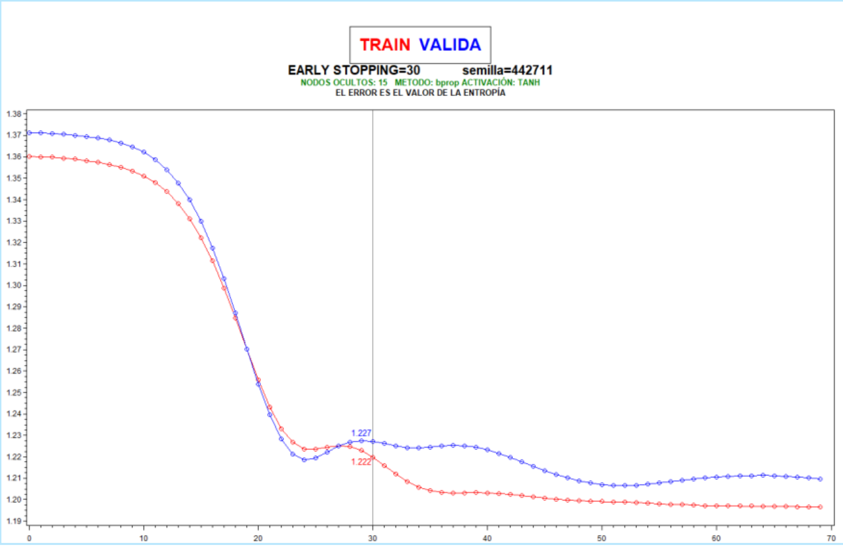


Figure 69: Occupancy Rate NN 15 hidden units Early Stopping

Nevertheless, we decided to take a closer look at it by taking these models to a cross-validation test with 10 different seeds and 4 groups.

```
%cruzadabinarianeural(archivo=airbnb,vardepen=Occu_BI,
conti=extra_people minimum_nights ,
categor=Has_License Shampoo Host_greets_you host_response_til cleaning_fee2
               cleaning_fee4 security_deposit1 maximum_nights2 availability_rate2
               availability_rate4 latitude2 latitude4 longitude2
               minimum_nights2 cancellation_policy2 neighbourhood_cleansed46
               neighbourhood_cleansed107 neighbourhood_group_clea10,
ngrupos=4,sinicio=12345,sfinal=12350,nodos=15,algo=bprop mom=0.8 learn=0.1);
data final101;set final;modelo=101;


%cruzadabinarianeural(archivo=airbnb,vardepen=Occu_BI,
conti=extra_people minimum_nights ,
categor=Has_License Shampoo Host_greets_you host_response_til cleaning_fee2
               cleaning_fee4 security_deposit1 maximum_nights2 availability_rate2
               availability_rate4 latitude2 latitude4 longitude2
               minimum_nights2 cancellation_policy2 neighbourhood_cleansed46
               neighbourhood_cleansed107 neighbourhood_group_clea10,
ngrupos=4,sinicio=12345,sfinal=12350,nodos=10,algo=bprop mom=0.2 learn=0.1);
data final102;set final;modelo=102;

%cruzadabinarianeural(archivo=airbnb,vardepen=Occu_BI,
conti=extra_people minimum_nights ,
categor=Has_License Shampoo Host_greets_you host_response_til cleaning_fee2
               cleaning_fee4 security_deposit1 maximum_nights2 availability_rate2
               availability_rate4 latitude2 latitude4 longitude2
               minimum_nights2 cancellation_policy2 neighbourhood_cleansed46
               neighbourhood_cleansed107 neighbourhood_group_clea10,
ngrupos=4,sinicio=12345,sfinal=12350,nodos=15,early=30,algo=bprop mom=0.8 learn=0.1);
data final103;set final;modelo=103;


%cruzadabinarianeural(archivo=airbnb,vardepen=Occu_BI,
conti=extra_people minimum_nights ,
categor=Has_License Shampoo Host_greets_you host_response_til cleaning_fee2
               cleaning_fee4 security_deposit1 maximum_nights2 availability_rate2
               availability_rate4 latitude2 latitude4 longitude2
               minimum_nights2 cancellation_policy2 neighbourhood_cleansed46
               neighbourhood_cleansed107 neighbourhood_group_clea10,
ngrupos=4,sinicio=12345,sfinal=12350,nodos=10,early=32,algo=bprop mom=0.8 learn=0.2);
data final104;set final;modelo=104;
```

Figure 70 shows the boxplot for these Neural Network models. With the model 102 (back prop, 10 units, mom=0.8 learn=0.1) we got a misclassification rate below 0.34.

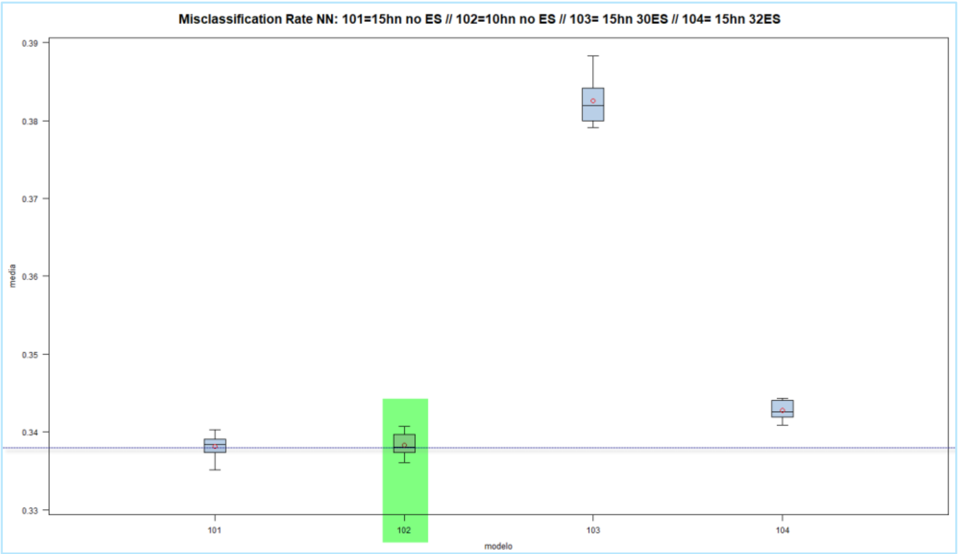


Figure 70: Occupancy Rate NN Models Backprop (Misclassification Rate boxplot)

6.2. Random Forest and Bagging

Proceeding with Random Forest models, we trained 6 models with the configuration in Figure 71:

RANDOM FOREST / BAGGING CONFIGURATION								
TREE	# Max Trees	Seed	% obs/sample	Max Depth	# Variables per branch	Significance Level	min. obs/node	Model
201	100	12345	0,6	10	15	0,1	30	Random Forest
202	1000	12346	1	10	5	0,1	20	Random Forest
203	1000	12347	1	10	5	0,05	20	Random Forest
204	200	12348	0,6	10	40	0,05	30	Random Forest
205	100	12345	0,6	10	20	0,1	30	Bagging
206	1000	12346	1	10	20	0,1	20	Bagging

Figure 71: Occupancy Rate RF and Bagging set up

The code for this section can be found in Appendix B.

In the boxplot (Figure 72), we can see the average accuracy rate of each model. Most of them were around 0.55 and 0.57.

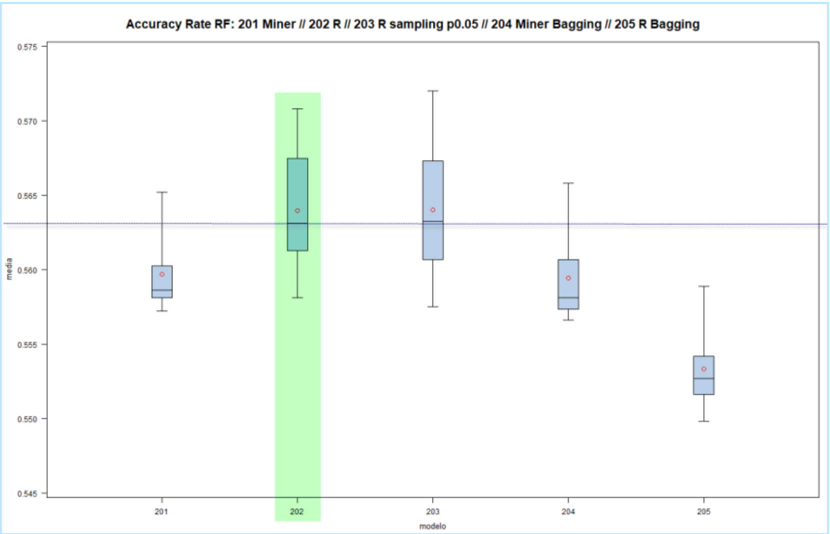


Figure 72: Occupancy Rate RF initial Models (Accuracy Rate boxplot)

In order to improve them (by reducing its variance), we increased the number of observations per leaf, decreased the *max depth* and the *p-value*, as we can see below.

```
%cruzadarandomforestbin(
archivo=airbnb,vardep=Occu_BI,
conti=extra_people minimum_nights ,
categor=Has_License Shampoo Host_greets_you host_response_til cleaning_fee2
               cleaning_fee4 security_deposit1 maximum_nights2 availability_rate2
               availability_rate4 latitude2 latitude4 longitude2
               minimum_nights2 cancellation_policy2 neighbourhood_cleansed46
               neighbourhood_cleansed107 neighbourhood_group_cleal0,
maxtrees=1000,variables=5,porcenbag=1,maxbranch=4,tamhoja=30,maxdepth=5,pvalor=0.1,
ngrupos=4,sinicio=13345,sfinal=13345,objetivo=tasafallos);
data final206;set final;modelo=206;
```

These parameters reduced the variance and increased accuracy rate. Therefore, we considered this one the best RF model.

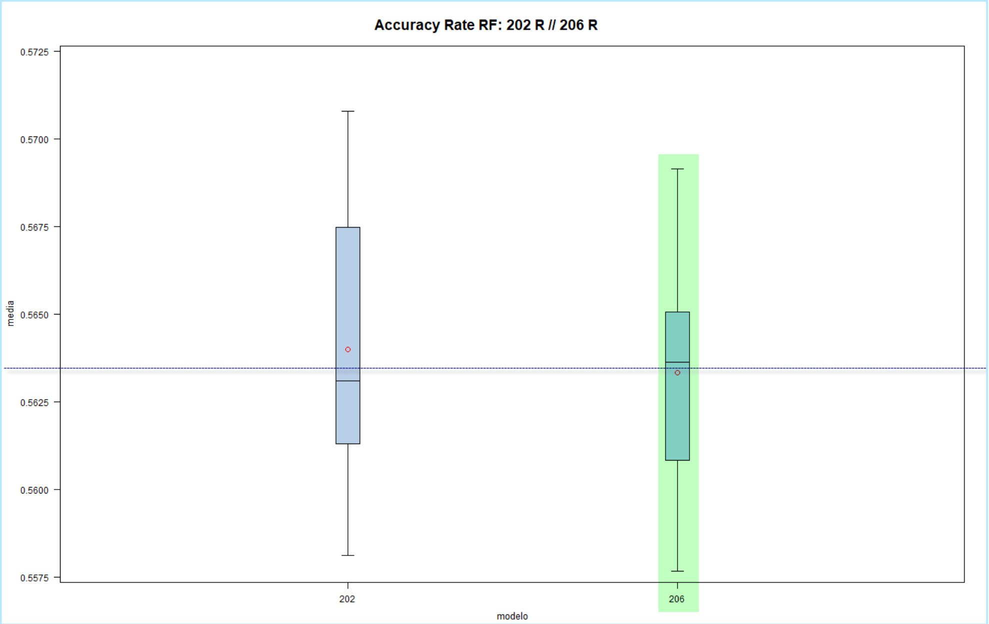


Figure 73: Occupancy Rate RF final Models (Accuracy Rate boxplot)

6.3. Gradient Boosting

For the Gradient Boosting models, we first trained 3 models (301, 302, and 303). The parameters set for each of them are described in the code lines that follow:

```
%cruzadatreeboostbin(archivo=airbnb,vardepen=Occu_BI,
conti=extra_people minimum_nights ,
categor=Has_License Shampoo Host_greets_you host_response_til cleaning_fee2
              cleaning_fee4 security_deposit1 maximum_nights2 availability_rate2
              availability_rate4 latitude2 latitude4 longitude2
              minimum_nights2 cancellation_policy2 neighbourhood_cleansed46
              neighbourhood_cleansed107 neighbourhood_group_clea10,
leafsize=20,iteraciones=300,shrink=0.1,maxbranch=2,maxdepth=5,mincatsize=20,minobs=20,
ngrupos=4,sinicio=13345,sfinal=13350,objetivo=tasaciertos);
data final301;set final;modelo=301;

%cruzadatreeboostbin(archivo=airbnb,vardepen=Occu_BI,
conti=extra_people minimum_nights ,
categor=Has_License Shampoo Host_greets_you host_response_til cleaning_fee2
              cleaning_fee4 security_deposit1 maximum_nights2 availability_rate2
              availability_rate4 latitude2 latitude4 longitude2
              minimum_nights2 cancellation_policy2 neighbourhood_cleansed46
              neighbourhood_cleansed107 neighbourhood_group_clea10,
leafsize=20,iteraciones=2000,shrink=0.2,maxbranch=2,maxdepth=2,mincatsize=20,minobs=30,
ngrupos=4,sinicio=13345,sfinal=13350,objetivo=tasaciertos);
data final302;set final;modelo=302;

%cruzadatreeboostbin(archivo=airbnb,vardepen=Occu_BI,
conti=extra_people minimum_nights ,
categor=Has_License Shampoo Host_greets_you host_response_til cleaning_fee2
              cleaning_fee4 security_deposit1 maximum_nights2 availability_rate2
              availability_rate4 latitude2 latitude4 longitude2
              minimum_nights2 cancellation_policy2 neighbourhood_cleansed46
              neighbourhood_cleansed107 neighbourhood_group_clea10,
leafsize=20,iteraciones=2000,shrink=0.2,maxbranch=2,maxdepth=10,mincatsize=20,minobs=30,
ngrupos=4,sinicio=13345,sfinal=13350,objetivo=tasaciertos);
data final303;set final;modelo=303;
```

With this algorithm, we got better results as we can see in the boxplot shown in Figure 74. The winner model was the 301 with a misclassification rate of 0.302 and accuracy of 0.69. The 302 had an accuracy rate of 0.68.

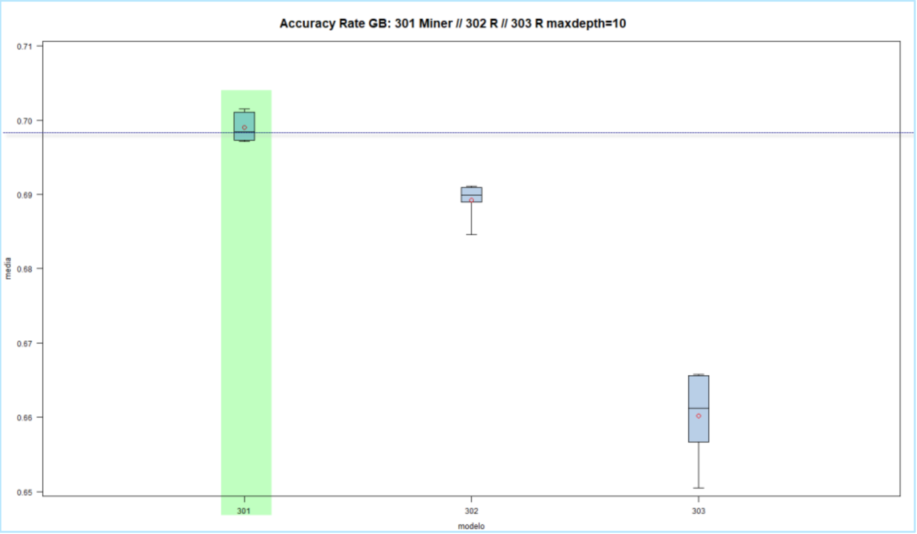


Figure 74: Occupancy Rate GBM initial Models (Accuracy Rate boxplot)

We tried to improve this model by manipulating the *shrinkage* and the *max depth* in two different models, 304 (shrink=0.05, leafsize/mincatsize/minobs = 30) and 305 (shrink=0.1, leafsize/mincatsize/minobs = 30).

```
%cruzadatreeboostbin(archivo=airbnb,vardepen=Occu_BI,
conti=extra_people minimum_nights ,
categor=Has_License Shampoo Host_greets_you host_response_til cleaning_fee2
               cleaning_fee4 security_deposit1 maximum_nights2 availability_rate2
               availability_rate4 latitude2 latitude4 longitude2
               minimum_nights2 cancellation_policy2 neighbourhood_cleansed46
               neighbourhood_cleansed107 neighbourhood_group_clea10,
leafsize=30,iteraciones=300,shrink=0.05,maxbranch=2,maxdepth=5,mincatsize=30,minobs=30,
ngrupos=4,sinicio=13345,sfinal=13350,objetivo=tasaciertos);
data final304;set final;modelo=304;

%cruzadatreeboostbin(archivo=airbnb,vardepen=Occu_BI,
conti=extra_people minimum_nights ,
categor=Has_License Shampoo Host_greets_you host_response_til cleaning_fee2
               cleaning_fee4 security_deposit1 maximum_nights2 availability_rate2
               availability_rate4 latitude2 latitude4 longitude2
               minimum_nights2 cancellation_policy2 neighbourhood_cleansed46
               neighbourhood_cleansed107 neighbourhood_group_clea10,
leafsize=30,iteraciones=300,shrink=0.1,maxbranch=2,maxdepth=5,mincatsize=30,minobs=30,
ngrupos=4,sinicio=13345,sfinal=13350,objetivo=tasaciertos);
data final305;set final;modelo=305;
```

In fact, we got even better results. All accuracy rates were greater than 0.7. The best model was the 304 with the p-value set to 0.05.

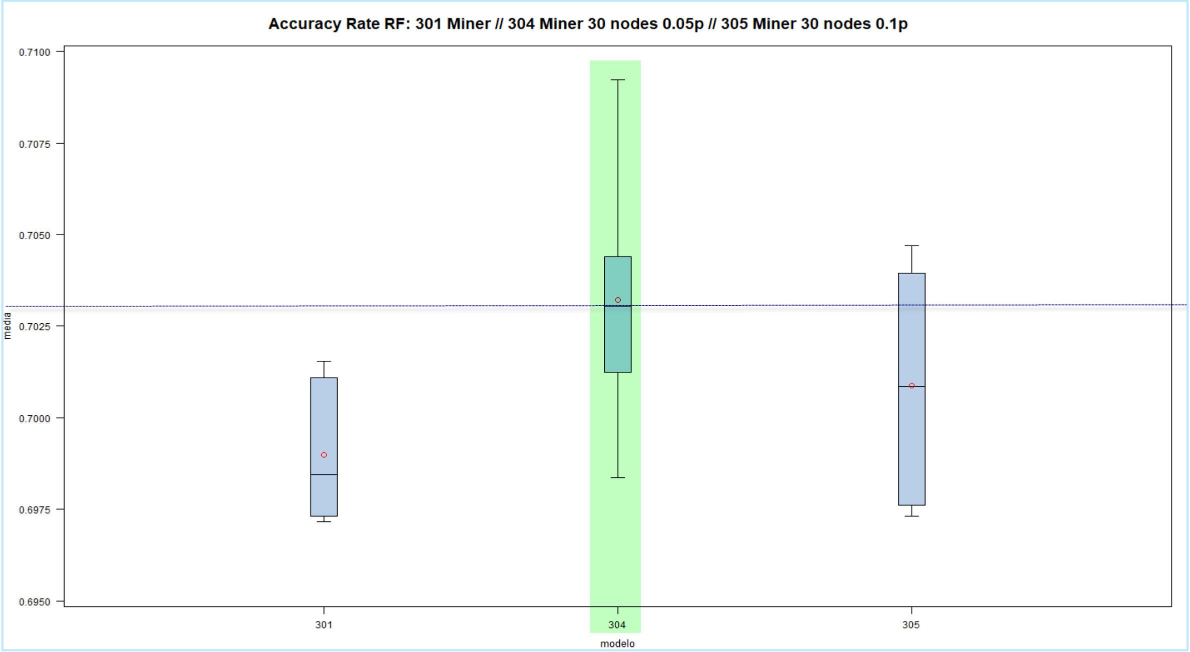


Figure 75: Occupancy Rate GBM final Models (Accuracy Rate boxplot)

6.4. K-Nearest Neighbor

To end this modeling section, we trained a new algorithm, the K-nearest neighbor (K-NN), we varied the K from 1 to 4.

```
%cruzadakNNbin(archivo=airbnb,vardepen=Occu_BI,listconti=extra_people minimum_nights Has_License Shampoo Host_greets_you host_response_til cleaning_fee2
               cleaning_fee4 security_deposit1 maximum_nights2 availability_rate2,ngrupos=4,seminicio=12345,semifinal=12350,k=1);
data final501;set final;modelo=501;

%cruzadakNNbin(archivo=airbnb,vardepen=Occu_BI,listconti=extra_people minimum_nights Has_License Shampoo Host_greets_you host_response_til cleaning_fee2
               cleaning_fee4 security_deposit1 maximum_nights2 availability_rate2,ngrupos=4,seminicio=12345,semifinal=12350,k=2);
data final502;set final;modelo=502;

%cruzadakNNbin(archivo=airbnb,vardepen=Occu_BI,listconti=extra_people minimum_nights Has_License Shampoo Host_greets_you host_response_til cleaning_fee2
               cleaning_fee4 security_deposit1 maximum_nights2 availability_rate2,ngrupos=4,seminicio=12345,semifinal=12350,k=3);
data final503;set final;modelo=503;

%cruzadakNNbin(archivo=airbnb,vardepen=Occu_BI,listconti=extra_people minimum_nights Has_License Shampoo Host_greets_you host_response_til cleaning_fee2
               cleaning_fee4 security_deposit1 maximum_nights2 availability_rate2,ngrupos=4,seminicio=12345,semifinal=12350,k=4);
data final504;set final;modelo=504;
```

In Figure 76, we can see the results for each of them. The model best model had K=3 and it got a 0.34 misclassification rate.

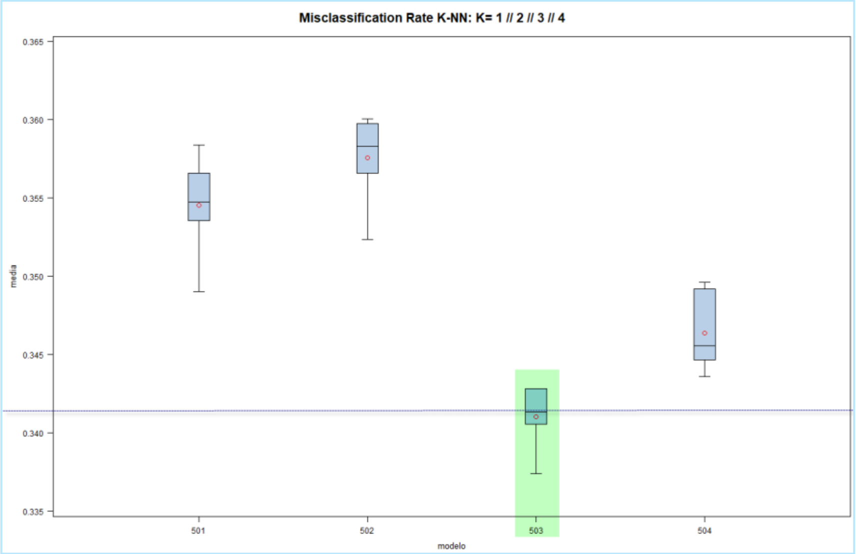


Figure 76: Occupancy Rate K-NN final Models (Misclassification Rate boxplot)

6.5. Models Assessment

Finally, we run the Repeated Cross-Validation Test with our 5 winners. To do so, we used 11 seeds and 4 CV groups.

In Figure 77 we have the accuracy rate boxplot. Cleary, the winner model is the Gradient Boosting, being the only one with an accuracy rate above 0,7.

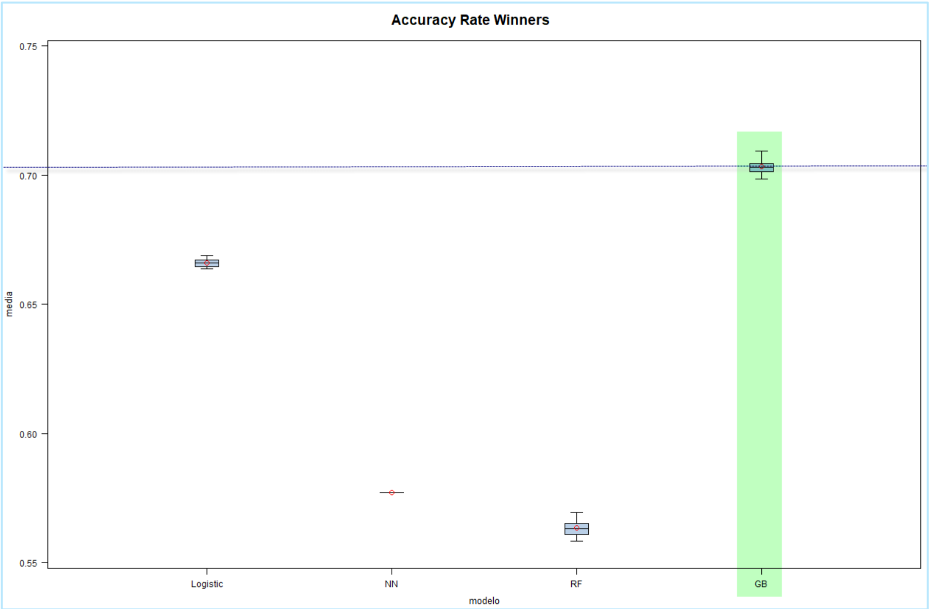


Figure 77: Occupancy Rate Models Assessment (boxplot)

To conclude our short study on the occupancy rate, we took the Gradient Boosting model and applied to in the same data of the predictions we did with Airbnb and Idealista. We analyzed all these data together in the following section.

7. DATA VISUALIZATION AND ANALYTICS

As a final stage of our project, we built a dashboard in Microsoft Power Bi, with the three predictions integrated in a dataset of the 296 properties on sale in Madrid in July 2019. With this dashboard, we wish to simulate the possible analytics and build the data visualization draft we could have in our app.

In Figure 78, we can see the Rentalbility Analytics Model. It is composed on the left edge by filters where the user could configure the aspects of their property search. On the middle, we have the Rental Index, with the estimated predictions from our models, the ROIC (Return on Investment in Cash) and ROIM (Return on Investment with Mortgage), the average property price and the Airbnb demand. We calculated the Airbnb demand using the predictions of the Occupancy Rate study, where the houses predicted with high occupancy rate were considered highly demanded (above 50%) and houses with an occupancy rate below 50% were considered low demand. On the right side, we have a bubbles map that shows us which rental channel is more profitable for this house. The colors refer to rental channel: green refers to Idealista and pink to Airbnb. The size of the bubble represents the ROI percentage, the biggest the bubble the higher is the rentability. In the lower part, we have a bar chart which analyses the ROI by district.

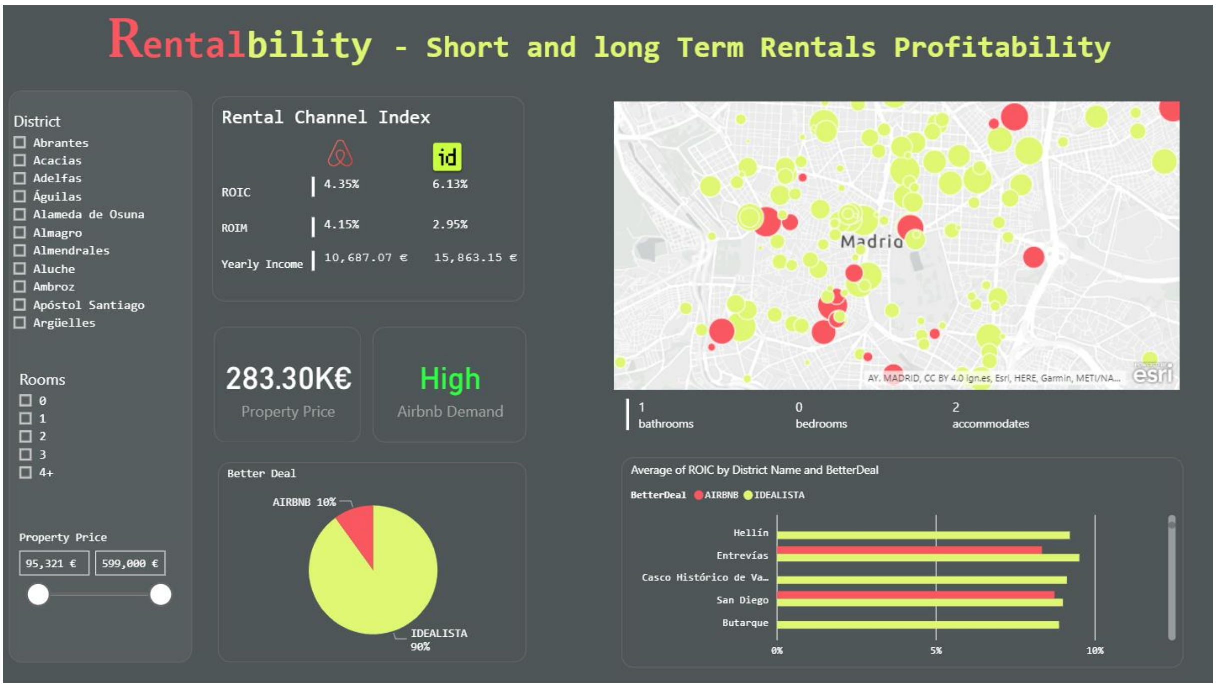


Figure 78: Rentalbility Analytics Model

The values appearing in Figure 78 could be considered an average of Madrid's market. Our average Idealista ROI in Madrid is 6%, which is close to 5.1% on reported by Idealista (Idealista, 2019b). This difference between Idealista's and Retalbilty could due to the expenses and calculations methodology. This similarity between both platforms reinforces the reliability of our model.

Madrid's average yearly income we have with Airbnb rental model is 10.6K. In Airbnb home page they suggest that in Madrid a host could earn about 12k – 18k euros with the platform (Airbnb, 2019c), depending on the house, however, they do not explain what is inside of their formula.

As we can see in, only 10% of the properties have higher ROI with short term rental than with long term. At first sight, it may seem odd, however, it was the expected. According to (elEconomista.es, 2018), on average, the vacation rental is only more profitable than the traditional when the occupancy rate exceeds 70%, which is, at the same time hard to achieve, since they do not offer a hospitality service. The explanation for this issue is that from Friday to Monday the occupation rate is hight, but from Tuesday to Thursday, it is significantly lower, since in those days the client profile is different and prefers a hotel that provides services (elEconomista.es, 2018). The fact that we limited our occupancy rate estimations to 70% to get a more conservative model could also be affecting this low percentage of better deals with the vacation rental.

When we analyze the bubbles chart in Figure 78, on the contrary to what we saw in Figure 3, the vacation lodging is not only concentrated in the city center. That have two implications: one is that this type of rental is not exclusively profitable in the city center of Madrid, and the second is that it may have some properties that are being used as vacation lodging that could be earning more profitability with traditional rental.

In Figure 79, we simulated an example of a search of a house located in Goya. We would have an average ROIC with Airbnb of almost 3% and with Idealista 4%. However, the best deal would depend on the aspects of the property. Out of the five properties we have on our database in Goya, four have a higher ROIC with long term rental model.

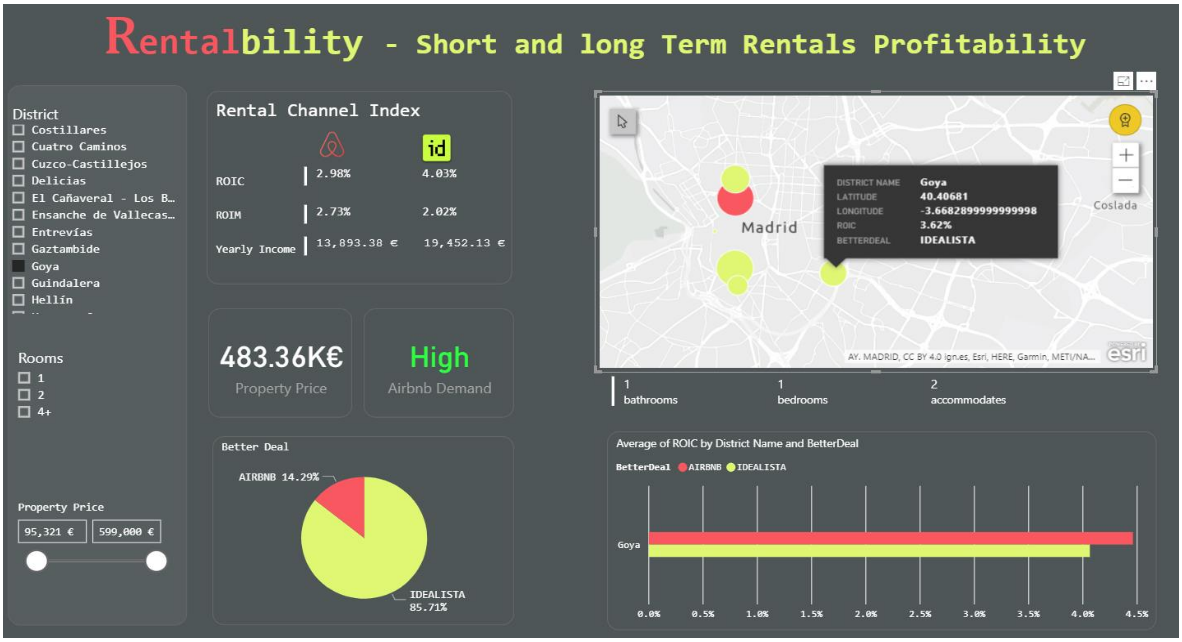


Figure 79: Rentalbility Model: Goya Example

Finally, from these analyses, we can conclude that when it comes to property investment, the decision of which rent strategy to chose is not all black and white, therefore, Rentalbility can be a really useful tool these investors.

8. CONCLUSION

The main goal of our project was to study and propose a methodology to calculate the Return on Investment of a rental property in Madrid, in the short and long term, using machine learning techniques. After this lengthy study, we believe we accomplished this goal. Nevertheless, it is necessary to review the methodology before the development and implementation of the tool.

While, the Idealista model works faultlessly, with an R^2 of 0.9, the Airbnb model presents a very low R^2 of 0.41. The main reason could be the lack of official and trustful data provided by Airbnb regarding the income and occupancy rate of its users. That increased the difficulty of developing a model for future listings, without using the most influential aspects of the house already listed and only using the attributes of the property. Thus, we believe the low R^2 is due to the calculations and estimations we needed to develop for the target variable. One possible solution for this issue would be to rerun the model, but with the original daily price variable as the target, and only apply the calculation to obtain the yearly profit afterward. Another possible solution would be to compare our estimations with other companies which provide Airbnb market research. Another aspect to highlight in the comparison between our two models is that Airbnb market is more volatile than Idealista. That is due to the short term relation of Airbnb business model.

Our secondary purpose was to understand how and which variables influence the rental prices of properties in Madrid. In fact, when we analyzed the variable importance graphics on the modeling phase, we could see clearly which one were the most relevant because they tend to repeat in every model. However, when we were analyzing the data, on Power BI, and investigating the behavior of these variables in order to find a pattern, we could not find any clear relationship between the data and the fact that the better deal was Airbnb or Idealista. That, together with the fact that our best models involved the combination of complex models, proofs there is a necessity for a tool to support individual investors on the decision-making process of which rental model is the most appropriate for each house.

With this research, we also sought to provide Madrid's public entities with a study to understand the fast-growing housing rental market and possibly assist the development of solutions with a positive social impact. Considering this, we see another application for our study, where we could compare our model's predictions with the actual Airbnb properties. As we saw in our analysis, only 10% of the houses available had a better deal with Airbnb. That leads us to the assumption that the owners of properties in Airbnb, could be earning more by coming back to the traditional rental model. With this information, Madrid's policymakers could create, for example, incentives to Airbnb property owners in areas of housing issues due to Airbnb excessive rentals, as Malasaña and Lavapies, to move back to traditional rentals.

To conclude, as next steps and future work for this project, after reviewing the Airbnb model, we would deploy the models using the R application Shiny. We also want to add real-time data from more portals, as Fotocasa and Homeaway to feed our database and provide more accurate information to our users.

9. BIBLIOGRAPHY

Airbnb, 2019a. About Us. Airbnb Newsroom. URL https://press.airbnb.com/en-us/about-us/ (accessed 9.11.19).

Airbnb, 2019b. Informe de actividad económica de Airbnb en la ciudad de Madrid [WWW Document]. URL https://press.airbnb.com/es/los-anfitriones-y-huespedes-en-airbnb-generaron-780-millones-e-en-madrid/ (accessed 5.9.19).

Airbnb, 2019c. Rent out your house, apartment or room on Airbnb [WWW Document]. Airbnb. URL https://www.airbnb.com/host/homes (accessed 9.5.19).

AirDNA, 2019. Analyze 20,664 Vacation Rentals in Madrid | MarketMinder [WWW Document]. AirDNA - Airbnb HomeAway Data. URL https://www.airdna.co/vacation-rental-data/app/es/madrid/madrid/overview (accessed 9.2.19).

Banco de España, 2019. Rentabilidad bonos (INDICADORES FINANCIEROS No. 1.2), SERIES DIARIAS.

Biz, C., 2016. How Airbnb was founded: a visual history - Business Insider [WWW Document]. URL https://www.businessinsider.com/how-airbnb-was-founded-a-visual-history-2016-2?IR=T#thats-when-the-company-hit-the-accelerator-on-growth-and-learned-a-bunch-about-their-business-chesky-famously-lived-exclusively-in-airbnbs-for-a-few-months-in-2010-when-their-employees-crowded-out-the-bedroom-space-left-in-their-apartment-15 (accessed 9.12.19).

Brealey, R.A., Myers, S.C., Allen, F., 2011. Principles of corporate finance, 10th ed. ed, The McGraw-Hill/Irwin series in finance, insurance, and real estate. McGraw-Hill/Irwin, New York.

Brousseau, F., 2015. Analysis of the impact of short-term rentals on housing [WWW Document]. CITY Cty. San Franc. BOARD Superv. - Budg. Legis. Anal. URL https://sfbos.org/sites/default/files/FileCenter/Documents/52601-BLA.ShortTermRentals.051315.pdf

Colliers International, 2018. Airbnb in Europe [WWW Document]. URL https://www.colliers.com/-/media/files/emea/emea/research/hotels/airbnb_spain_2018_v7.pdf?la=en-gb

Comunidad de Madrid, 2019. Boletín Oficial de la Comunidad de Madrid, Decreto 79/2014.

Cox, M., 2019. Inside Airbnb. Adding data to the debate. [WWW Document]. Airbnb. URL http://insideairbnb.com (accessed 9.5.19).

elEconomista.es, 2018. ¿En qué ciudad puede ganar más con el alquiler de su vivienda? La rentabilidad toca máximos - elEconomista.es [WWW Document]. URL https://www.eleconomista.es/construccion-inmobiliario/noticias/9086516/04/18/En-que-ciudad-puede-ganar-mas-con-la-compra-de-vivienda-para-alquilar-La-rentabilidad-toca-maximos.html (accessed 1.13.19).

elEconomista.es, 2018. El alquiler turístico solo es más rentable que el residencial si la ocupación supera el 70% - elEconomista.es [WWW Document]. URL https://www.eleconomista.es/empresas-finanzas/inmobiliaria/noticias/9187415/06/18/El-alquiler-turistico-solo-gana-al-residencial-si-se-ocupa-al-70.html (accessed 9.16.19).

Folger, J., 2019. How to Calculate the ROI on a Rental Property [WWW Document]. Investopedia. URL https://www.investopedia.com/articles/investing/062215/how-calculate-roi-rental-property.asp (accessed 9.3.19).

Gitman, L., J., 2004. Principles of Managerial Finance, 10th ed. Pearson Education.

Idealista, 2019a. La rentabilidad de la inversión en vivienda se sitúa en 7,5% en el primer trimestre [WWW Document]. idealista/news. URL https://www.idealista.com/news/inmobiliario/vivienda/2019/04/25/772879-la-rentabilidad-de-la-inversion-en-vivienda-se-situa-en-7-5-en-el-primer-trimestre (accessed 9.9.19).

Idealista, 2019b. Informes de precios. Guías y contratos. Evolución precio vivienda — idealista [WWW Document]. URL https://www.idealista.com/informes-precio-vivienda (accessed 1.13.19).

Idealista, 2018a. Descubre cómo ha evolucionado la rentabilidad de la vivienda en las principales ciudades desde 2013 [WWW Document]. idealista/news. URL https://www.idealista.com/news/inmobiliario/vivienda/2018/08/01/766861-la-evolucion-de-la-rentabilidad-de-la-vivienda-en-las-capitales-de-provincia-desde (accessed 1.7.19).

Idealista, 2018b. El precio del alquiler crece más rápido que el de venta: cómo han cambiado en los últimos años [WWW Document]. idealista/news. URL https://www.idealista.com/news/inmobiliario/vivienda/2018/11/07/769434-los-alquileres-de-vivienda-crecen-mas-rapido-que-los-precios-de-venta-con-madrid-y (accessed 1.7.19).

Instituto de Estadística de la Comunidad de Madrid, 2019. Viajeros, pernoctaciones, grado de ocupación y estancia media en apartamentos turísticos [WWW Document]. Anu. Estad. Comunidad Madr. 1985-2019. URL http://www.madrid.org/iestadis/fijas/estructu/general/anuario/ianucap13.htm

Junta Municipal Distrito Centro, RED2RED, 2017. Análisis del impacto de las viviendas de uso turístico en el distrito Centro [WWW Document]. URL https://diario.madrid.es/centro/2017/05/08/presentado-el-analisis-del-impacto-de-las-viviendas-de-uso-turistico-en-el-distrito-centro/

Manelmc, 2016. python - How to get real estate data with Idealista API? [WWW Document]. Stack Overflow. URL https://stackoverflow.com/questions/40023931/how-to-get-real-estate-data-with-idealista-api (accessed 9.4.19).

Marqusee, A., 2015. Airbnb and San Francisco: Descriptive Statistics and Academic Research.

Mashvisor, 2019. Traditional and Airbnb Investment Property [WWW Document]. Mashvisor. URL https://www.mashvisor.com/ (accessed 9.13.19).

Morde, V., 2019. XGBoost Algorithm: Long May She Reign! [WWW Document]. Medium. URL https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d (accessed 9.14.19).

Numbeo, 2019. Cost of Living in Madrid [WWW Document]. URL https://www.numbeo.com/cost-of-living/in/Madrid (accessed 9.5.19).

Portela, J., 2019. Machine Learning Notes.

Rodgers, A., 2015. Executive Summary of Amendments Relating to Short-Term Rentals [WWW Document]. San Franc. Plan. Dep. URL http://commissions.sfplanning.org/cpcpackets/2014-001033PCA.pdf

SAS, 2018. Data Mining and SEMMA :: Data Mining Using SAS(R) Enterprise Miner(TM): A Case Study Approach, Third Edition [WWW Document]. URL http://support.sas.com/documentation/cdl/en/emcs/66392/HTML/default/viewer.htm#n0pejm83csbja4n1xueveo2uoujy.htm (accessed 1.14.19).

SHARING ECONOMY | meaning in the Cambridge English Dictionary [WWW Document], 2019. . Camb. Adv. Learn. Dict. Thesaurus. URL https://dictionary.cambridge.org/dictionary/english/sharing-economy (accessed 9.11.19).

UNICORN | meaning in the Cambridge English Dictionary [WWW Document], 2019. URL https://dictionary.cambridge.org/dictionary/english/unicorn (accessed 9.12.19).

10. APPENDIX

Appendix A: Idealista Variables Description

Variable	Description	Variable Type	Role	Modeling	Comments	#
Column1	column id	ID number	Rejected			1
index	index	ID number	Rejected			2
address	address	text	Rejected			3
bathrooms	number of bathrooms	Interval	Input			4
country	country	"es"	Rejected			5
detailedType	Type and subtype of property	Text	Rejected			6
distance	distance from Center (Sol)	Interval	Input			7
district	district	Categorical Text	Input			8
exterior	is a exterior	boolean	Input			9
externalReference	externalReference	Text	Rejected			10
floor	floor	Nominal	Input			11
has360	has360	boolean	Rejected			12
has3DTour	has3DTour	boolean	Rejected			13
hasLift	hasLift	boolean	Input			14
hasPlan	hasPlan	boolean	Input			15
hasVideo	hasVideo	boolean	Input			16
latitude_bad	latitude_bad	Interval	Rejected			17
latitude	latitude	Interval	Input			18
longitude_bad	longitude_bad	Interval	Rejected			19
longitude	longitude	Interval	Input			20
municipality	municipality	"Madrid"	Rejected		Filtered for Mardrid	21
neighborhood	neighborhood	Nominal	Input			22
newDevelopment	newDevelopment	boolean	Rejected			23
numPhotos	numPhotos	Interval	Input			24
operation	Sale or Rent	"rent"	Rejected			25
parkingSpace	parkingSpace	text	Rejected		created new variables	26
price	Rental price	Interval	Rejected			27
priceByArea	price per m2	Interval	Rejected			28
propertyCode	property Id Code	ID	Input			29
propertyType	propertyType	Nominal	Input			30
province	province	Nominal	Rejected			31
rooms	rooms	Nominal	Input			32
showAddress	showAddress	boolean	Input			33
size	size	Interval	Input			34
status	status	Nominal	Rejected			35
suggestedTexts	suggested tittle	text	Rejected			36
thumbnail	thumbnail	text	Rejected			37
url	url	text	Rejected			38
AC	AC	boolean	Input			39
Piscina	Piscina	boolean	Input			40
Terraza	Terraza	boolean	Input			41
Amueblado	Amueblado	Nominal	Input			42
SUM	SUM	Nominal	Input			43
Count if	Count if	Interval	Rejected			44
Rule	Rule	boolean	Rejected			45
Has_Parking	Has_Parking	boolean	Input			46
Parking_Price_Included	Parking_Price_Included	boolean	Input			47
Parking_Price	Parking_Price	Interval	Input			48
Yearly_Price	Yearly_Price	Interval	Target			49
Parking	combination of Parking	Nominal	Input			50

Appendix B: Access to Codes Repository

With the following link, it is possible to access a repository on GitHub with all codes (in R, Python and SAS) used in this dissertation.

Scan me

or

https://github.com/pri-nel/TFM_Rental-Predictions

Appendix C: Idealista Neighborhood and Group levels

LEVEL	GROUP
12 DE OCTUBRE-ORCASUR	0
ABRANTES	0
AEROPUERTO	0
ALUCHE	0
AMBROZ	0
AMPOSTA	0
BERRUGUETE	0
BUENA VISTA	0
BUTARQUE	0
CAMPAMENTO	0
CANILLEJAS	0
CASCO HISTÓRICO DE BARAJAS	0
CASCO HISTÓRICO DE VALLECAS	0
EL CAÑAVERAL - LOS BERROCALES	0
ENSANCHE DE VALLECAS - LA GAVIA	0
ENTREVÍAS	0
FONTARRÓN	0
HORCAJO	0
LOS ÁNGELES	0
NUMANCIA	0
OPAÑEL	0
ORCASITAS	0
PALOMERAS BAJAS	0
PALOMERAS SURESTE	0
PAVONES	0
PORTAZGO	0
PRADOLONGO	0
PUERTA BONITA	0
SAN ANDRÉS	0
SAN DIEGO	0
SAN FERMÍN	0
TIMÓN	0
VALDEACEDERAS	0
VINATEROS	0
VISTA ALEGRE	0
ÁGUILAS	0
ALMENDRALES	1
ARCOS	1
BELLAS VISTAS	1
CASCO HISTÓRICO DE VICÁLVARO	1
COMILLAS	1
IMPERIAL	1
LOS CÁRMENES	1
LOS ROSALES	1
LUCERO	1
MARROQUINA	1
MOSCARDÓ	1
PALOS DE MOGUER	1
PAU DE CARABANCHEL	1
PILAR	1
PUEBLO NUEVO	1
PUERTA DEL ÁNGEL	1
QUINTANA	1
REJAS	1
SAN ISIDRO	1
SANTA EUGENIA	1
SIMANCAS	1
TRES OLIVOS - VALVERDE	1
VALDEBERNARDO - VALDERRIBAS	1
VENTAS	1
ZOFÍO	1
ACACIAS	2
ADELFAS	2
CANILLAS	2
CASA DE CAMPO	2
CHOPERA	2

LEVEL	GROUP
COLINA	2
CONCEPCIÓN	2
CUATRO CAMINOS	2
EL PARDO	2
LAVAPIÉS-EMBAJADORES	2
LEGAZPI	2
MEDIA LEGUA	2
PINAR DEL REY	2
PROSPERIDAD	2
ROSAS	2
VALDEZARZA	2
VENTILLA-ALMENARA	2
VIRGEN DEL CORTIJO - MANOTERAS	2
APÓSTOL SANTIAGO	3
ARROYO DEL FRESNO	3
CAMPO DE LAS NACIONES-CORRALEJOS	3
CUZCO-CASTILLEJOS	3
DELICIAS	3
FUENTE DEL BERRO	3
FUENTELARREINA	3
GUINDALERA	3
PACÍFICO	3
PALACIO	3
PEÑAGRANDE	3
SAN PASCUAL	3
SANCHINARRO	3
SOL	3
CIUDAD JARDÍN	4
CONDE ORGAZ-PIOVERA	4
ESTRELLA	4
GAZTAMBIDE	4
HUERTAS-CORTES	4
IBIZA	4
LAS TABLAS	4
MALASAÑA-UNIVERSIDAD	4
MONTECARMELO	4
ALAMEDA DE OSUNA	5
ARGÜELLES	5
CHUECA-JUSTICIA	5
CIUDAD UNIVERSITARIA	5
COSTILLARES	5
GOYA	5
NUEVOS MINISTERIOS-RÍOS ROSAS	5
SAN JUAN BAUTISTA	5
VALLEHERMOSO	5
ARAPILES	6
ARAVACA	6
ATALAYA	6
BERNABÉU-HISPANOAMÉRICA	6
EL VISO	6
LA PAZ	6
LISTA	6
NUEVA ESPAÑA	6
TRAFALGAR	6
VALDEBEBAS - VALDEFUENTES	6
ALMAGRO	7
CASTELLANA	7
CASTILLA	7
EL PLANTÍO	7
JERÓNIMOS	7
MIRASIERRA	7
NIÑO JESÚS	7
PALOMAS	7
RECOLETOS	7
SALVADOR	7
VALDEMARÍN	7

Appendix D: Idealista Variables Selection & Transformations Results

Idealista Variables Transformation Node

```
Computed Transformations
(maximum 500 observations printed)

                                  Input
Input Name                Role    Level     Name                         Level       Formula

Parking_Price_Included    INPUT   INTERVAL  SQR_Parking_Price_Included   INTERVAL    (max(Parking_Price_Included-0, 0.0))**2
SUM                       INPUT   INTERVAL  EXP_SUM                      INTERVAL    exp(max(SUM-0, 0.0)/3)
distance                  INPUT   INTERVAL  SQRT_distance                INTERVAL    sqrt(max(distance-15, 0.0)/14394)
latitude                  INPUT   INTERVAL  LOG_latitude                 INTERVAL    log(max(latitude-403343502, 0.0)/1982194  + 1)
longitude                 INPUT   INTERVAL  PWR_longitude                INTERVAL    (max(longitude--38318927, 0.0)/2895753)**4
size                      INPUT   INTERVAL  PWR_size                     INTERVAL    (max(size-15, 0.0)/1985)**0.25
```

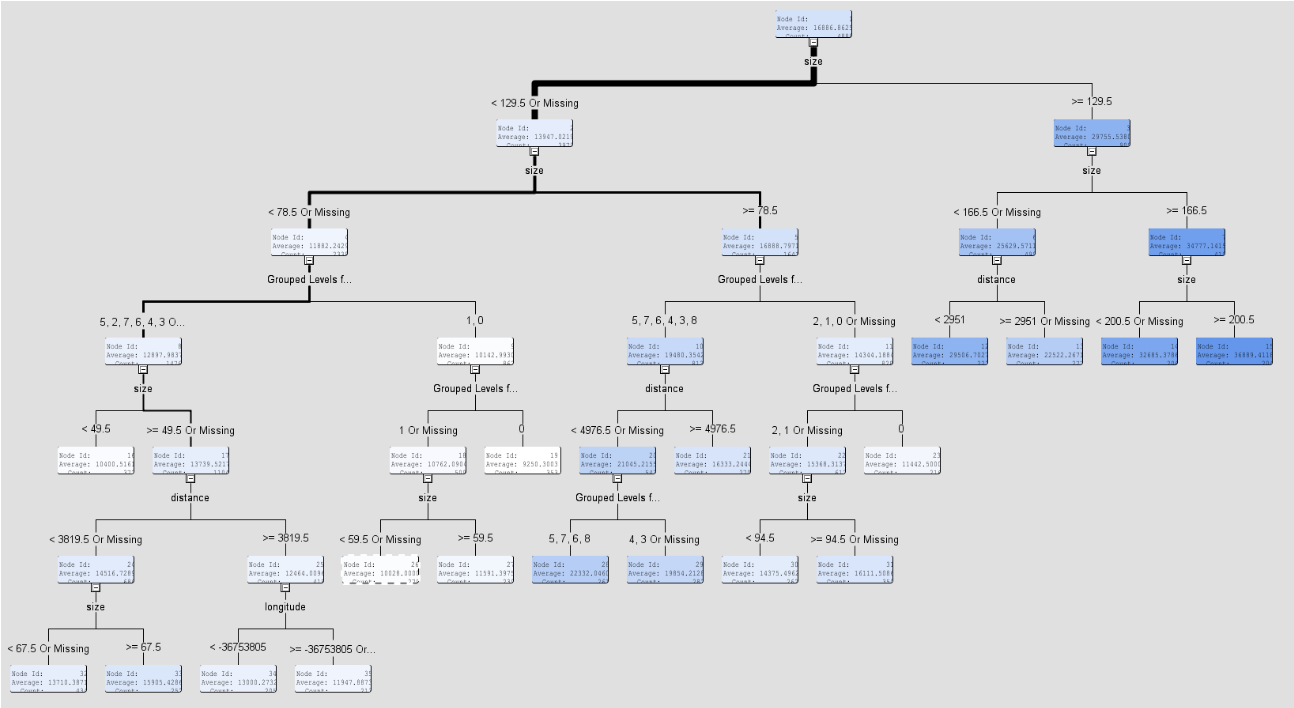
Idealista Variables Selection Node

Label	Role ▲	Measurement Level	Type	Reasons for Rejection
AC	Input	Binary	Numeric	
Grouped Levels for G_neighborhood	Input	Nominal	Numeric	
Grouped Levels for REP_bathrooms	Input	Nominal	Numeric	
Grouped Levels for REP_district	Input	Nominal	Numeric	
Grouped Levels for REP_floor	Input	Nominal	Numeric	
Grouped Levels for REP_rooms	Input	Nominal	Numeric	
Has_Parking	Input	Binary	Numeric	
Transformed: latitude	Input	Interval	Numeric	
Transformed: longitude	Input	Interval	Numeric	
Transformed: size	Input	Interval	Numeric	
Transformed: distance	Input	Interval	Numeric	
Amueblado	Rejected	Binary	Character	Varsel:Small R-square value
Transformed: SUM	Rejected	Interval	Numeric	Varsel:Small R-square value
Grouped Levels for neighborhood	Rejected	Nominal	Numeric	Varsel:Small R-square value, Group variable preferred
Imputed: hasLift	Rejected	Nominal	Numeric	Varsel:Small R-square value
Parking	Rejected	Nominal	Character	Varsel:Small R-square value
Piscina	Rejected	Binary	Numeric	Varsel:Small R-square value
Replacement: bathrooms	Rejected	Nominal	Numeric	Varsel:Small R-square value, Group variable preferred
Replacement: floor	Rejected	Nominal	Character	Varsel:Small R-square value, Group variable preferred
Replacement: rooms	Rejected	Nominal	Numeric	Varsel:Small R-square value, Group variable preferred
Transformed: Parking_Price_Included	Rejected	Interval	Numeric	Varsel:Small R-square value
Terraza	Rejected	Binary	Numeric	Varsel:Small R-square value
exterior	Rejected	Binary	Numeric	Varsel:Small R-square value
propertyType	Rejected	Nominal	Character	Varsel:Small R-square value

Idealista Clustering Node

Cluster	Label	R-Square With Own Cluster Component	Next Closest Cluster	R-Square with Next Cluster Component	Type	1-R2 Ratio	Variable Selected
CLUS1	Cluster 1	1	CLUS2	0.056222	ClusterComp	0	YES
CLUS1	Parking_Price_Included	0.59801	CLUS2	0.028506	Variable	0.413786	NO
CLUS1	distance	0.639218	CLUS2	0.150283	Variable	0.424591	NO
CLUS1	SUM	0.491393	CLUS2	0.012022	Variable	0.514796	NO
CLUS1	latitude	0.388228	CLUS2	0.00771	Variable	0.616525	NO
CLUS1	size	0.236639	CLUS2	.0008324	Variable	0.763997	NO
CLUS2	Cluster 2	1	CLUS1	0.056222	ClusterComp	0	YES
CLUS2	longitude	1	CLUS1	0.056222	Variable	0	NO

Idealista Decision Tree Node



Appendix E: Airbnb Variables Description

Variable	Description	Variable Type	Role	Comments
id	Ad/room unique Identification	ID Number	ID	
listing_url	Link to the room ad	URL	reject	
scrape_id	"Inside Airbnb" scrape Id	ID Number	reject	
last_scraped	Scrape date	Date	reject	
name	Title of the the Ad	Text	reject	
summary	Short description of the house	Text	reject	
space	Description of The space	Text	reject	
description	Full description of the house	Text	reject	
experiences_offered	If the owner offer a Airbnb Expirience (all ar	"None"	reject	
neighborhood_overview	Description of the neighborhood	Text	reject	
notes	Other things to note	Text	reject	
transit	Explanations of how to get to the house	Text	reject	
access	Description of Guest access	Text	reject	
interaction	Description of the kink of interaction with gu	Text	reject	
house_rules	Description of House Rules	Text	reject	
thumbnail_url	Empty field	Blank	reject	
medium_url	Empty field	Blank	reject	
picture_url	Link to the cover picture	URL	reject	
xl_picture_url	Empty field	Blank	reject	
host_id	Host unique Identification	ID Number	reject	
host_url	Link to the host profile	URL	reject	
host_name	Host name	Text ID	reject	
host_since	Date from host sign up	Date	reject	extracted days
Host since days	Self Calculated for Host since days	Numerical	Input	
host_location	Host location (City, State, Contry)	Text	reject	
host_about	Short description of the host	Text	reject	
host_response_time	Time host takes to reply a message	Categorical Text	Input	
host_response_rate	How many messages the host replies	Porcentage	Input	
host_acceptance_rate	Host acceptance Rate	"N/A"	reject	
host_is_superhost	If host is a Super Host	Boolean	Input	
host_thumbnail_url	Host Thumbnail	URL	reject	
host_picture_url	Host Picture	URL	reject	
host_neighbourhood	House Neighbourhood	Text	reject	
host_listings_count	How many houses/rooms the host hast in Ai	Numerical	reject	
host_total_listings_count	How many houses/rooms the host hast in Ai	Numerical	reject	
host_verifications	How Host was verified	Text	reject	Filtered for contains "government_id"
host_has_profile_pic	If Host Has Profile Picture	Boolean	Input	
host_identity_verified	If Host Identity was Verified	Boolean	Input	50% False
street	House location (City, State, Contry)	Text	reject	Not accurate neither relieable
neighbourhood	District	Categorical Text	reject	
neighbourhood_cleansed	District	Categorical Text	Input	
neighbourhood_group_cleansed	Neighbourhood Group Zone	Categorical Text	Input	
city	City	Categorical Text	reject	Not accurate neither relieable
state	State	Categorical Text	reject	Not accurate neither relieable
zipcode	Zipcode	Numerical	Input	
market	Market	Categorical Text	reject	Not accurate neither relieable
smart_location	Smart Location	Categorical Text	reject	Not accurate neither relieable
country_code	Country Code	"ES"	reject	
country	Country	"Spain"	reject	Not accurate neither relieable
latitude	Latitude	Numerical	Input	
longitude	Longitude	Numerical	Input	
is_location_exact	If the Location is Exact	Boolean	Input	
property_type	Property Type	Categorical Text	Input	
room_type	Room Type	Categorical Text	reject	Filtered for only "Entire home/apt"
accommodates	How many guests can accommodates de ho	Numerical Category	Input	
bathrooms	Amount of bathrooms	Numerical Category	Input	0,5 means toilette only
bedrooms	Amount of Bedrooms	Numerical Category	Input	
beds	Amount of Beds	Numerical	Input	
bed_type	Bed Type	Categorical Text	Input	
amenities	Which Amenities the house has	Text	reject	Created new boolean variable for each
square_feet	Square Feet of the house	Numerical	Input	
price	Price per Night	Numerical	reject	filtered less than 900
weekly_price	Weekly Price	Numerical	reject	Too many missings
monthly_price	Monthly Price	Numerical	reject	Too many missings
security_deposit	Security Deposit	Numerical	Input	
cleaning_fee	Cleaning Fee	Numerical	Input	
guests_included	Amount of Guests Included on night price	Numerical	reject	
extra_people	Fee per Extra People	Numerical	reject	
minimum_nights	Minimum Nights of stay	Numerical	Input	

Variable	Description	Variable Type	Role	Comments
maximum_nights	Maximum Nights of stay	Numerical	Input	
minimum_minimum_nights	Minimum Minimum Nights	Numerical	reject	
maximum_minimum_nights	Maximum Minimum Nights	Numerical	reject	
minimum_maximum_nights	Minimum Maximum Nights	Numerical	reject	
maximum_maximum_nights	Maximum Maximum Nights	Numerical	reject	
minimum_nights_avg_ntm	Minimum Nights in Avg from last Twelve Mo	Numerical	reject	Filtered for less than 300 days
maximum_nights_avg_ntm	Maximum Nights in Avg from last Twelve M	Numerical	reject	
calendar_updated	Last time Calendar was Updated	Categorical Text	reject	
has_availability	Has Availability	"t"	reject	
availability_30	Availability in 30 days	Numerical	reject	
availability_60	Availability in 60 days	Numerical	reject	
availability_90	Availability in 90 days	Numerical	reject	
availability_365	Availability in 365 days	Numerical	reject	
calendar_last_scraped	Calendar Last Scraped	Date	reject	
number_of_reviews	Number Of Reviews	Numerical	Input	
number_of_reviews_ltm	Number Of Reviews Last Twelve Months	Numerical	Input	Filtered for more than 0
first_review	First Review	Date	reject	
last_review	Last Review	Date	reject	
review_scores_rating	Review Scores Rating	Numerical Category	Input	From 1 to 10
review_scores_accuracy	Review Scores Accuracy	Numerical Category	Input	From 1 to 10
review_scores_cleanliness	Review Scores Cleanliness	Numerical Category	Input	From 1 to 10
review_scores_checkin	Review Scores Checkin	Numerical Category	Input	From 1 to 10
review_scores_communication	Review Scores Communication	Numerical Category	Input	From 1 to 10
review_scores_location	Review Scores Location	Numerical Category	Input	From 1 to 10
review_scores_value	Review Scores Value	Numerical Category	Input	From 1 to 10
requires_license	Requires License	"t"	reject	
license	License	Text	reject	Modified to Has License?
Has_License	Self Calculated for Has License	Boolean	Input	
jurisdiction_names	Jurisdiction Names	Blank	reject	
instant_bookable	If it is Instant Bookable	Boolean	Input	
is_business_travel_ready	If it is Business Travel Ready	Boolean	Input	
cancellation_policy	Cancellation Policy	Categorical Text	Input	Has 6 categories
require_guest_profile_picture	If requires Guest Profile Picture	Boolean	reject	
require_guest_phone_verificatio	If require Guest Phone Verification	Boolean	reject	
calculated_host_listings_count	Calculated Host Listings Count	Numerical	reject	
calculated_host_listings_count_	Calculated Host Listings Count Entire Home	Numerical	Input	
calculated_host_listings_count_	Calculated Host Listings Count Private Roor	Numerical	reject	
calculated_host_listings_count_	Calculated Host Listings Count Shared Roor	Numerical	reject	
reviews_per_month	Average number of reviews Per Month	Numerical	no	[number_of_reviews]/[calendar_last_scraped]-[first_review])/30)
Air conditioning	Self Calculated for Has Air conditioning	Boolean	Input	
Internet	Self Calculated for Has Internet	Boolean	Input	
Pool	Self Calculated for Has Pool	Boolean	Input	
Breakfast	Self Calculated for Has Breakfast	Boolean	Input	
Free street parking	Self Calculated for Has Free street parking	Boolean	Input	
Shampoo	Self Calculated for Has Shampoo	Boolean	Input	
24-hour check-in	Self Calculated for Has 24-hour check-in	Boolean	Input	
Laptop friendly workspace	Self Calculated for Has Laptop friendly worl	Boolean	Input	
Bathtub	Self Calculated for Has Bathtub	Boolean	Input	
Hot water	Self Calculated for Has Hot water	Boolean	Input	
Microwave	Self Calculated for Has Microwave	Boolean	Input	
Coffee maker	Self Calculated for Has Coffee maker	Boolean	Input	
Refrigerator	Self Calculated for Has Refrigerator	Boolean	Input	
Cooking basics	Self Calculated for Has Cooking basics	Boolean	Input	
Patio or balcony	Self Calculated for Has Patio or balcony	Boolean	Input	
Long term stays allowed	Self Calculated for Is Long term stays allowe	Boolean	Input	
Host greets you	Self Calculated for Host greets you	Boolean	Input	
Days on Airbnb	Self Calculated for Days on Airbnb	Numerical	reject	[last_review]-[first_review]
MIN_Booking_YEAR	Self Calculated for MIN Booking YEAR	Numerical	reject	IFERROR([number_of_reviews]/([Days on Airbnb]/365);[reviews_per_month]*12)
EST_Bookings_YEAR	Self Calculated for EST Bookings YEAR	Numerical	reject	[MIN_Booking_YEAR]/50%
Nights_Per_YEAR_CAP	Self Calculated for Nights Per YEAR CAP	Numerical	reject	IF([EST_Bookings_YEAR]*IF([minimum_nights_avg_ntm]>2;[minimum_nights_avg_ntm];2)>255;255;[EST_Bookings_YEAR]*IF([minimum_nights_avg_ntm]>2;[minimum_nights_avg_ntm];2))
Occupancy Rate	Self Calculated for Occupancy Rate	Numerical	reject	[Nights_Per_YEAR_CAP]/365
Yearly Revenue	Self Calculated for Yearly Revenue	Numerical	reject	[price]*[Occupancy Rate]*365
Est_NPY_IA	Estimated from Inside Airbnb for Est NPY IA	Numerical	reject	IF((([reviews_per_month]*12/50%)*IF([minimum_nights_avg_ntm]>2;[minimum_nights_avg_ntm];2))>255;255;(([reviews_per_month]*12/50%)*IF([minimum_nights_avg_ntm]>2;[minimum_nights_avg_ntm];2)))
Occupancy Rate IA	Estimated from Inside Airbnb for	Numerical	reject	[Est_NPY_IA]/365
Month Income IA	Estimated from Inside Airbnb for Month	Numerical	reject	[Occupancy Rate IA]*[price]*30
Year IA	Estimated from Inside Airbnb for Year	Numerical	reject	[Month Income IA]*12
Utilities Cost	Self Calculated Basic Utilities cost	Numerical	reject	81,144648585+(([guests_included]-1)*17,501774895)
Cost Year	Self Calculated for Cost per Night	Numerical	reject	([Yearly Revenue]*3%)+((42,73+[Utilities Cost])*12*[Occupancy Rate])
Yearly Profit	Self Calculated for Profit	Numerical	TARGET	[Yearly Revenue]-[Cost per Night]
availability_rate	availability_rate			([availability_30]*12)/365

Appendix F: Airbnb Replacement Values for Class Variable

Variable	Formatted Value	Type	Character Unformatted Value	Numeric Value	Replacement Value	Label
accommodates	8	C	8	.	7+	accommodates
accommodates	7	C	7	.	7+	accommodates
accommodates	10	C	10	.	7+	accommodates
accommodates	12	C	12	.	7+	accommodates
accommodates	9	C	9	.	7+	accommodates
accommodates	16	C	16	.	7+	accommodates
accommodates	1	C	1	.	2	accommodates
accommodates	11	C	11	.	7+	accommodates
accommodates	14	C	14	.	7+	accommodates
accommodates	13	C	13	.	7+	accommodates
accommodates	15	C	15	.	7+	accommodates
bathrooms	1,0	C	1,0	.	1	bathrooms
bathrooms	2,0	C	2,0	.	2	bathrooms
bathrooms	1,5	C	1,5	.	1	bathrooms
bathrooms	3,0	C	3,0	.	3+	bathrooms
bathrooms	2,5	C	2,5	.	2	bathrooms
bathrooms	4,0	C	4,0	.	3+	bathrooms
bathrooms	3,5	C	3,5	.	3+	bathrooms
bathrooms	5,0	C	5,0	.	3+	bathrooms
bathrooms	4,5	C	4,5	.	3+	bathrooms
bathrooms	0,5	C	0,5	.	3+	bathrooms
bathrooms		C		.	_blank_	bathrooms
bathrooms	0,0	C	0,0	.	_blank_	bathrooms
bathrooms	5,5	C	5,5	.	3+	bathrooms
bathrooms	6,0	C	6,0	.	3+	bathrooms
bathrooms	6,5	C	6,5	.	3+	bathrooms
bedrooms	4	C	4	.	4+	bedrooms
bedrooms	5	C	5	.	4+	bedrooms
bedrooms	6	C	6	.	4+	bedrooms
bedrooms	7	C	7	.	4+	bedrooms
bedrooms		C		.	_blank_	bedrooms
bedrooms	8	C	8	.	4+	bedrooms
beds	7	C	7	.	7+	beds
beds	8	C	8	.	7+	beds
beds	9	C	9	.	7+	beds
beds	10	C	10	.	7+	beds
beds	12	C	12	.	7+	beds
beds	11	C	11	.	7+	beds
beds	14	C	14	.	7+	beds
beds	16	C	16	.	7+	beds
beds	17	C	17	.	7+	beds
beds		C		.	_blank_	beds
beds	13	C	13	.	7+	beds
beds	15	C	15	.	7+	beds
cancellation_policy	super_strict_30	C	super_strict_30	.	super_strict	cancellation_policy
cancellation_policy	super_strict_60	C	super_strict_60	.	super_strict	cancellation_policy
host_response_time	within a few hours	C	within a few hours	.	more than a hour	host_response_time
host_response_time	within a day	C	within a day	.	more than a hour	host_response_time
host_response_time	a few days or more	C	a few days or more	.	more than a hour	host_response_time
property_type	Guest suite	C	Guest suite	.	Other	property_type
property_type	Other	C	Other	.	Other	property_type
property_type	Guesthouse	C	Guesthouse	.	Other	property_type
property_type	Casa particular (Cuba)	C	Casa particular (Cuba)	.	Other	property_type
property_type	Townhouse	C	Townhouse	.	Other	property_type
property_type	Chalet	C	Chalet	.	Other	property_type
property_type	Camper/RV	C	Camper/RV	.	Other	property_type
property_type	Bed and breakfast	C	Bed and breakfast	.	Other	property_type
property_type	Hut	C	Hut	.	Other	property_type
property_type	Tiny house	C	Tiny house	.	Other	property_type
property_type	Villa	C	Villa	.	Other	property_type
zipcode		C		.	_blank_	zipcode
zipcode						

Appendix G: Airbnb Neighborhood and Group levels

LEVEL	GROUP
ABRANTES	0
ALMENDRALES	0
APOSTOL SANTIAGO	0
ARCOS	0
BUTARQUE	0
CANILLEJAS	0
CIUDAD UNIVERSITARIA	0
EL PLANTÍO	0
HELLÍN	0
LOS ROSALES	0
PUERTA BONITA	0
SAN ANDRÉS	0
SAN FERMÍN	0
VISTA ALEGRE	0
ZOFÍO	0
Los Ángeles	0
Pau de Carabanchel	0
Buena Vista	0
ADELFAS	1
AEROPUERTO	1
AGUILAS	1
ALMENARA	1
ALUCHE	1
ARAVACA	1
BELLAS VISTAS	1
CASCO HISTÓRICO DE VALLECAS	1
CASCO HISTÓRICO DE VICÁLVARO	1
CHOPERA	1
CIUDAD JARDÍN	1
COMILLAS	1
CONCEPCIÓN	1
CÁRMENES	1
EL GOLOSO	1
ENTREVÍAS	1
FONTARRÓN	1
LUCERO	1
MIRASIERRA	1
NUMANCIA	1
PILAR	1
PINAR DEL REY	1
PRADOLONGO	1
PROSPERIDAD	1
PUERTA DEL ANGEL	1
QUINTANA	1
REJAS	1
SIMANCAS	1
VALDEACEDERAS	1
VALDEZARZA	1
VENTAS	1
VINATEROS	1
Ventilla-Almenara	1
Águilas	1
Orcasitas	1
Ensanche de Vallecas - La Gavia	1
Valdemarín	1
Apóstol Santiago	1
Puerta del Ángel	1
Fuentelarreina	1
Arroyo del Fresno	1
ACACIAS	2
ARAPILES	2
BERRUGUETE	2
BUENAVISTA	2
CASA DE CAMPO	2
CASTILLEJOS	2
COLINA	2
COSTILLARES	2
DELICIAS	2
FUENTE DEL BERRO	2
GUINDALERA	2
IMPERIAL	2
MOSCARDÓ	2
PALOMERAS BAJAS	2

LEVEL	GROUP
PEÑAGRANDE	2
PIOVERA	2
PUEBLO NUEVO	2
SAN DIEGO	2
SAN ISIDRO	2
SAN JUAN BAUTISTA	2
SANTA EUGENIA	2
TRAFALGAR	2
VALDEFUENTES	2
VALLEHERMOSO	2
Conde Orgaz-Piovera	2
Cuzco-Castillejos	2
Valdebebas - Valdefuentes	2
El Cañaveral - Los Berrocales	2
Atalaya	2
Ambroz	2
Virgen del Cortijo - Manoteras	2
Sanchinarro	2
ALMAGRO	3
ARGÜELLES	3
ATOCHA	3
CANILLAS	3
CASTILLA	3
CUATRO CAMINOS	3
GAZTAMBIDE	3
HISPANOAMÉRICA	3
LA PAZ	3
LEGAZPI	3
LOS ANGELES	3
NUEVA ESPAÑA	3
OPAÑEL	3
ORCASUR	3
PACÍFICO	3
PALOMERAS SURESTE	3
PORTAZGO	3
RIOS ROSAS	3
VALVERDE	3
Nuevos Ministerios-Ríos Rosas	3
Tres Olivos - Valverde	3
Bernabéu-Hispanoamérica	3
Montecarmelo	3
Las Tablas	3
EL VISO	4
EMBAJADORES	4
GOYA	4
JERÓNIMOS	4
LISTA	4
PALOS DE MOGUER	4
SAN CRISTOBAL	4
SAN PASCUAL	4
TIMÓN	4
UNIVERSIDAD	4
Malasaña-Universidad	4
Lavapiés-Embajadores	4
ALAMEDA DE OSUNA	5
CASCO HISTÓRICO DE BARAJAS	5
CASTELLANA	5
IBIZA	5
JUSTICIA	5
NIÑO JESÚS	5
PALACIO	5
Chueca-Justicia	5
CAMPAMENTO	6
CORTES	6
PALOMAS	6
SOL	6
Huertas-Cortes	6
CORRALEJOS	7
ESTRELLA	7
MEDIA LEGUA	7
RECOLETOS	7
ROSAS	7
SALVADOR	7
Campo de las Naciones-Corralejos	7

Appendix H: Airbnb Variables Selection & Transformations Results

Airbnb Variables Transformation Node

```
Computed Transformations
(maximum 500 observations printed)

                                   Input
Input Name                 Role    Level      Name                        Level                     Formula

IMP_REP_cleaning_fee       INPUT   INTERVAL   SQRT_IMP_REP_cleaning_fee   INTERVAL   sqrt(max(IMP_REP_cleaning_fee-0, 0.0)/180)
IMP_security_deposit       INPUT   INTERVAL   LOG_IMP_security_deposit    INTERVAL   log(max(IMP_security_deposit-0, 0.0)/4000  + 1)
REP_maximum_nights         INPUT   INTERVAL   PWR_REP_maximum_nights      INTERVAL   (max(REP_maximum_nights-1, 0.0)/364)**4
availability_rate          INPUT   INTERVAL   PWR_availability_rate       INTERVAL   (max(availability_rate-0, 0.0)/0.9863013699)**0.25
extra_people               INPUT   INTERVAL   LOG_extra_people            INTERVAL   log(max(extra_people-0, 0.0)/240  + 1)
latitude                   INPUT   INTERVAL   PWR_latitude                INTERVAL   (max(latitude-40.33249, 0.0)/0.17528)**4
longitude                  INPUT   INTERVAL   PWR_longitude               INTERVAL   (max(longitude--3.8355, 0.0)/0.2536)**4
minimum_nights             INPUT   INTERVAL   PWR_minimum_nights          INTERVAL   (max(minimum_nights-1, 0.0)/90)**4
```

Airbnb Variables Selection Node

Label	Role	Measurement Level	Type	Reasons for Rejection ▲
	Input	Binary	Numeric	
Grouped Levels for G_neighbourhood_cleansed	Input	Nominal	Numeric	
Grouped Levels for IMP_REP_bedrooms	Input	Nominal	Numeric	
Grouped Levels for IMP_REP_beds	Input	Nominal	Numeric	
Grouped Levels for REP_accommodates	Input	Nominal	Numeric	
	Input	Binary	Numeric	
Imputed: Replacement: bathrooms	Input	Nominal	Character	
Imputed: Replacement: host_response_time	Input	Nominal	Character	
Transformed: Imputed: security_deposit	Input	Interval	Numeric	
	Input	Binary	Numeric	
	Input	Binary	Numeric	
Transformed: availability_rate	Input	Interval	Numeric	
	Input	Binary	Numeric	
Refrigerator	Input	Binary	Numeric	
Transformed: Imputed: Replacement: cleaning_fee	Input	Interval	Numeric	
Shampoo	Input	Binary	Numeric	
host_identity_verified	Input	Binary	Character	
instant_bookable	Input	Binary	Character	
Bathtub	Rejected	Binary	Numeric	Varsel:Small R-square value
Breakfast	Rejected	Binary	Numeric	Varsel:Small R-square value
	Rejected	Binary	Numeric	Varsel:Small R-square value
	Rejected	Binary	Numeric	Varsel:Small R-square value
	Rejected	Binary	Numeric	Varsel:Small R-square value
Has_License	Rejected	Binary	Numeric	Varsel:Small R-square value
	Rejected	Binary	Numeric	Varsel:Small R-square value
Internet	Rejected	Binary	Numeric	Varsel:Small R-square value
Transformed: extra_people	Rejected	Interval	Numeric	Varsel:Small R-square value
Microwave	Rejected	Binary	Numeric	Varsel:Small R-square value
Transformed: Replacement: maximum_nights	Rejected	Interval	Numeric	Varsel:Small R-square value
Transformed: latitude	Rejected	Interval	Numeric	Varsel:Small R-square value
Transformed: longitude	Rejected	Interval	Numeric	Varsel:Small R-square value
Transformed: minimum_nights	Rejected	Interval	Numeric	Varsel:Small R-square value
Pool	Rejected	Binary	Numeric	Varsel:Small R-square value
Replacement: cancellation_policy	Rejected	Nominal	Character	Varsel:Small R-square value
	Rejected	Binary	Numeric	Varsel:Small R-square value
is_location_exact	Rejected	Binary	Character	Varsel:Small R-square value
Grouped Levels for neighbourhood_cleansed	Rejected	Nominal	Numeric	Varsel:Small R-square value, Group variable preferred
Imputed: Replacement: bedrooms	Rejected	Nominal	Character	Varsel:Small R-square value, Group variable preferred
Imputed: Replacement: beds	Rejected	Nominal	Character	Varsel:Small R-square value, Group variable preferred
Replacement: accommodates	Rejected	Nominal	Character	Varsel:Small R-square value, Group variable preferred

Airbnb Clustering Node

Cluster	Label	R-Square With Own Cluster Component	Next Closest Cluster	R-Square with Next Cluster Component	Type	1-R2 Ratio	Variable Selected
CLUS1	Cluster 1	1	CLUS3	0.010873	ClusterComp	0	YES
CLUS1	Imputed: security_deposit	0.568678	CLUS2	0.00352	Variable	0.432846	NO
CLUS1	Imputed: Replacement: cle...	0.57041	CLUS3	0.015054	Variable	0.436155	NO
CLUS1	extra_people	0.170284	CLUS3	0.00374	Variable	0.832831	YES
CLUS2	Cluster 2	1	CLUS1	0.002974	ClusterComp	0	YES
CLUS2	longitude	0.611736	CLUS3	.0006406	Variable	0.388513	NO
CLUS2	latitude	0.592834	CLUS1	0.003896	Variable	0.408759	NO
CLUS2	minimum_nights	0.0252	CLUS1	0.002186	Variable	0.976935	YES
CLUS3	Cluster 3	1	CLUS1	0.010873	ClusterComp	0	YES
CLUS3	Replacement: maximum_n...	0.529051	CLUS1	0.00374	Variable	0.472717	NO
CLUS3	availability_rate	0.529051	CLUS1	0.008196	Variable	0.474841	NO

Airbnb Decision Tree Node