Article

Identifying Real Estate Opportunities using Machine Learning

Alejandro Baldominos [1,2]*, Antonio José Moreno [2], Rubén Iturrarte [2], Óscar Bernárdez [2] and Carlos Afonso [2]

[1] Computer Science Department, Universidad Carlos III de Madrid, Leganés, Spain
[2] Artificial Intelligence Group, Rentier Token, Madrid, Spain
* Correspondence: abaldomi@inf.uc3m.es; Tel.: +34-91-624-6258

Version October 15, 2018 submitted to Preprints

Abstract: The real estate market is exposed to many fluctuations in prices, because of existing correlations with many variables, some of which cannot be controlled or might even be unknown. Housing prices can increase rapidly (or in some cases, also drop very fast), yet the numerous listings available online where houses are sold or rented are not likely to be updated that often. In some cases, individuals interested in selling a house (or apartment) might include it in some online listing, and forget about updating the price. In other cases, some individuals might be interested in deliberately setting a price below the market price in order to sell the home faster, for various reasons. In this paper we aim at developing a machine learning application that identifies opportunities in the real estate market in real time, i.e., houses that are listed with a price substantially below the market price. This program can be useful for investors interested in the housing market. We have focused in a use case considering real estate assets located in the Salamanca district in Madrid (Spain) and listed in the most relevant Spanish online site for home sales and rentals. The application is formally implemented as a regression problem, that tries to estimate the market price of a house given features retrieved from public online listings. For building this application, we have performed a feature engineering stage in order to discover relevant features that allows attaining a high predictive performance. Several machine learning algorithms have been tested, including regression trees, *k*-nearest neighbors, support vector machines and neural networks, identifying advantages and handicaps of each of them.

Keywords: real estate; appraisal; investment; machine learning; artificial intelligence

---

1. Introduction

The real estate market is rapidly evolving. A recent report published by MSCI estimates the size of the professionally managed real estate investment market in $8.5 trillion in 2017, increasing a total of $1.1 trillion since the previous year [1]. Of course, the real market size is expected to be much larger when counting assets which are not professionally managed or that are not object of investment.

When looked from a macroeconomic perspective, there are many aspects that significantly drive the behavior of this market, such as demographics, interest rates, government regulation and, for short, global economic health.

However, looking at the market evolution from a global perspective turns out to be too simplistic. Although the market at a global scale is very tightly correlated, there are many aspects influencing the behavior of markets at a local scale, such as political instability or the emergence of highly demanded “hot spots" that can shift rapidly. Also, different market segments evolve at different paces, such as high-end luxury condos.

An example of such differences can be observed in Figure 1, which shows the evolution of the price (measured in euros per square meter) of resale houses in four different Spanish regions: Barcelona (blue), Madrid (yellow), Palma de Mallorca (red) and Lugo (green). Data for Barcelona and Madrid are available since 2002, whereas for Palma de Mallorca and Lugo are available after 2008.

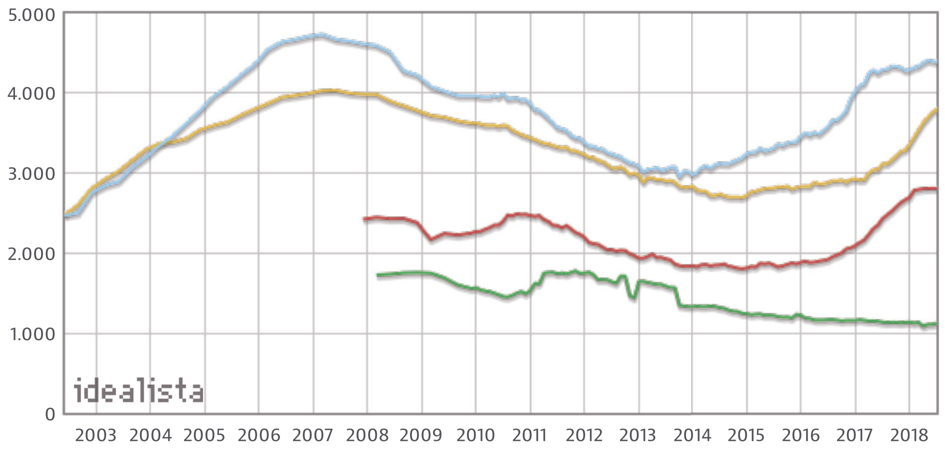

Figure 1. Evolution of the Spanish resale real estate market, focusing on four different regions: Barcelona (blue), Madrid (yellow), Palma de Mallorca (red) and Lugo (green). Source: Idealista [2].

Attending to the figure we can see some common patterns in the evolution of Barcelona's and Madrid's markets, which are the two largest metropolitan areas in Spain: a growth in the early 2000s with a maximum in 2007, followed by a fall after 2008 due to the global financial crisis, which lasted until 2015, a moment after which prices have started to recover almost reaching maximum values again as of 2018.

Meanwhile, in Palma de Mallorca, which is a remarkable vacational place, the increase in the last two years is more pronounced than in Madrid and Barcelona. On the other hand, in Lugo, which is occupied mostly by rural areas, prices increased slightly in 2011, but are steadily decreasing since then. The figure clearly shows how although some global patterns can be found in the evolution, each region still shows some specificities in the prices evolution.

Of course, higher variability can be found when looking at specific assets. Most important factors driving the value of a house are the size and the location, but there are many other variables that are often taken into account when determining its value: number of bedrooms, proximity to some form of public transport (buses, underground, etc.), number and quality of schools in the area, shopping opportunities, availability of an elevator (in apartments located in higher floors), availability of gardens or parks, etc. But certainly, the main reason ultimately determining the value of houses is demand.

With such a high variability and unpredictable factors (e.g., one neighborhood deemed better or more fashionable than others) it is likely that the price of some assets will deviate from its expected value. When the actual price is much less than the expected value, we could be dealing with an investment opportunity: an asset that could generate immediate profit if sold soon after its purchase.

There can be a number of reasons motivating the existence of these investment opportunities, from which two can be easily identified. The first one can occur when the seller publishes the advertisement in some online listing. In that case, it could occur that the seller ignored the actual value of the house. More likely, the house has remained published in the listing for some time, during which its value has risen (because of global or local trends) but the seller has not updated its value. In the second case, the asset price can be deliberately set lower than its expected value with the intention of it be sold fast, for example if the seller needs the money.

Whichever the case, investment opportunities in real estate exist and can be identified. In this paper, we aim at using machine learning techniques to identify such opportunities, by determining whether the price of an asset is smaller than its estimated value. In particular, we have considered a dataset of real estate assets located in the Salamanca district of Madrid, Spain, and listed in Idealista, the most relevant Spanish online site for home sales and rentals, during the second semester of 2017.

The remainder of this paper is structured as follows: Section 2 describes the state of the art in prediction of houses prices, and places this work in its context. Later, section 3 describes the dataset used to train the models, with the machine learning techniques being described in section 4. An evaluation of the system is performed and its setup and results are discussed in section 5. Finally, some conclusive remarks and future lines of work are provided in section 6.

2. State of the Art

Traditional models used for the appraisal or estimation of real estate assets have relied in hedonic regression, which breaks the asset apart into its constituent characteristics in order to establish a relationship between each of these and the price of the property. The model learned with hedonic regression can then be used to estimate the price of an asset whose characteristics are known in advance. In recent years, two works using this method have been presented by Jiang et al. [3,4] focusing on properties in Singapore with sale transactions between 1995 and 2014. The data used by the authors comprise a total of 315,000 transactions related to 216,000 dwellings in 4,820 buildings, from which some were resales. Properties considered by authors to generate the model are the location, specified as the postal district, the property type (apartment or condominium) and the ownership type (99 years, 999 years or freehold).

Some related work is concerned with trying to estimate the health of a real estate market using data obtained from search queries and news. In these cases, the housing index price (HPI) is often used as a proxy to determine the status of such market. As an example, Greenstein et al. [5] hypothesized that search indices are correlated with the underlying conditions of the US housing market, and relied on search data obtained from Google Trends as well as housing market indicators for testing that hypothesis. It is remarkable that authors state that traditional forecasting had often relied on statistics provided in annual reports, financial statements, government data, etc., which are often published with significant delay and do not provide enough data resolution. To validate their hypothesis, a seasonal autoregressive model was used to estimate the relationship. In the end, they are capable of predicting home sales using online search, and also the house price index and demand for home appliances. However, this does not allow the prediction of individual real estate assets. A similar work was also published the same year by Sun et al. [6] who base their research on the fact that 90% buyers and 92% sellers rely on the Internet for finding or posting information about properties. In their work, authors propose combining online daily news sentiments with search engine data from Baidu for predicting the house price index. Prediction models are trained using support vector regression, artificial neural networks trained with back-propagation and radial basis functions neural networks, and tested on data retrieved from Beijing, Shanghai, Chengdu and Hangzhou.

Other works rely on expert systems based on fuzzy logic, because of the similarity between this technique and the human approach to decision making [7]. For example, Guan et al. [8] have used adaptive neuro-fuzzy inference system (ANFIS) for real estate appraisal. To do so, they have used data from 20,192 sales records in a mid-western city of the US between 2003 and 2007, which were finally reduced to 16,523 after manual curation. The features comprised by this data include location, year of construction and sale, square footage in each floor (including basement and garage), number of baths, number of fireplaces, presence of central air, lot type, construction type, wall type and basement type. Their approach combines neural networks with fuzzy inference, and authors report mean average percentage errors for different age bands and different variables sets. It can be seen that ANFIS outperform multiple regression analysis (MRA), although in some cases using location only instead of all variables can improve results. In 2016, Sarip et al. [9] presented a work where they used fuzzy least-squares regression FLSR to build a model from 352 properties in the district of Petaling, Kuala Lumpur. For constructing the model, eight variables were considered: land area, built-up area, number of bedrooms, number of bathrooms, building age, repair condition, quality of furniture and location, along with the price in Malaysian ringgits (MYR). Those features which were categorial were first converted into numeric attributes to be manageable by the model. Authors reported a mean absolute error around 183,000 for FLSR, smaller than ANFIS and artificial neural networks. Finally, in 2017, del Giudice et al. [10] introduced an approach where fuzzy logic was used to evaluate the situations of a real estate market with imprecise information, showing a case study which focused in the purchase of one office building. Interestingly, del Giudice et al. state that while there is a wide theoretical background regarding real estate investment, empirical support for such theory is scarce in the literature.

A work by Rafiei et al. [11] presents an innovative approach where the focus is placed not in property appraisal but in decision making at the time of starting a new construction, i.e., whether "to build or not to build", in the words of the authors. Authors describe the use of a deep belief restricted Boltzmann machine for learning a model from 350 condos (3-9 stories) built in Tehran (Iran) between 1993 and 2008. From these condos, a total of 26 attributes are used to train the model. Seven attributes refer to physical and financial properties of the constructions, such as the ZIP code, the total floor area and the lot area, the estimated construction cost per square meter, the duration of construction and the price of the unit at the beginning of the project. Authors also retrieved additional economic variables, which included regulation in the area, prices index, financial status in the construction area, currency exchange rate, population and demographics, etc. When evaluating their model over a test set of another 10 condos they obtain a test error as low as 3.6%, a number better than that attained using neural networks trained with backpropagation.

Finally, some authors have relied on the use of machine learning techniques for estimating or predicting the price of individual real estate assets. It is the case of Park and Kwon Bae [12], who have analyzed housing data of 5,359 townhouses in Fairfax County, Virginia, combined from various databases, from 2004 and 2007. These assets had 76 attributes, from which 28 were eventually selected after filtering using a *t*-test and logistic regression. Sixteen of these features are physical features, referring to the number of bedrooms and bathrooms, number of fireplaces, total area, cooling and heating systems, parking type, etc. Besides, three variables refer to the ratings of elementary, middle and high schools in the area, and other eight refer to the mortgage contract rate, location and construction and sale date. The last attribute is the class, which the authors have converted in binary: either the closing price is larger than the listing price or the other way around. Therefore, the problem can be seen as a classification to decide whether an investment is worthy or not instead of a regression problem. For performing classification, authors compare different algorithms: decision trees (C4.5), RIPPER, Naive Bayes and AdaBoost. Best results are achieved using RIPPER (repeated incremental pruning to produce error reduction), a propositional rule learner, with an average error of about 25%. Authors do not provide more specific metrics such as F1 score which allows to determine the model goodness regardless of the class distribution.

Another work was presented by Manganelli et al. [13] in 2016, where the focus is put on appraisal based on 148 sales of residencial property in a city of the Campania region, Italy. Considered variables are the age of the property, date of sale, internal area, balconies area, connected area, number of services, number of views, maintenance status and floor level. Authors used linear programming to analyze the real estate data, learning the coefficients for the model, achieving an average percentage error of 7.13%.

Finally, two more works have been published by del Giudice et al. [14,15] in 2017. In the first of these works, authors use a genetic algorithm (GA) to try to establish a relationship between rental prices and the location of assets in a central urban of Naples divided into five subareas, considering also the commercial area, maintenance status and floor to be relevant parameters. The way in which the authors use the GA is towards learning weights for these parameters, effectively building a regression model which could be used for estimating rental prices. The obtained absolute average percentage error is equal to 10.62%. In the second work, authors rely on Markov chain hybrid Monte Carlo method (MCHMCM), yet they also tested neural networks, MRA and penalized spline semiparametric method (PSSM). The model is again used for predicting the price of real estate properties in the center of Naples, with the dataset comprising only 65 housing sales in twelve months. Besides the variables from the previous work, they also considered the number of bathrooms, outer surface, panoramic quality, occupancy status and distance from the funicular station. The MCHMCM model achieves an absolute average percentage error of 6.61%, a better rate that the tested alternatives.

As shown in this section, literature on the application of machine learning to the valuation of real estate assets is relatively scarce, at least in the way we are approaching the problem. While the few last works cited in this section adhere to the same approach than ours, those works are restricted to certain areas and only probes a small set of techniques. In this work, we will also focus on the snapshot of the real estate market in a small region during a period of six months, yet evaluating additional machine learning techniques, and approaching the problem as a regression task.

3. Data

In this paper, we will be focusing in a small segment of the market, comprising high-end real estate assets located in the Salamanca district in Madrid, Spain. The city of Madrid is the capital of Spain and also the largest city, with more than 3 million residents. Its location within Spain (and also framed within the context of the European continent) is shown in Figure 2. The city is divided into 21 administrative divisions called districts. The Salamanca district is located to the north-east of the historical center (see Figure 3) and is home to 150,000 people, being one of the wealthiest and more expensive places in Madrid and Spain.

3.1. Source and Description

In order to identify opportunities within the framework of a stable market, we have used data from all residences costing more than one million euros, listed in Idealista (the more relevant Spanish online platform for real estate sales and rentals) in the second semester of 2017 (from June 1st to December 31st).

Figure 2. Location of the city of Madrid within Spain.

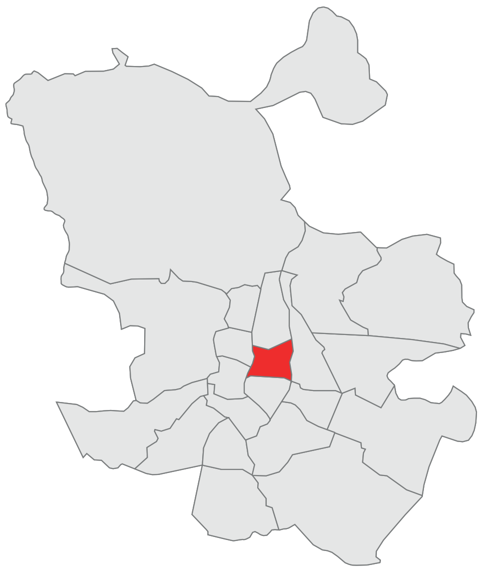

Figure 3. Location of the Salamanca district within the city of Madrid.

The raw data used for training the machine learning models comprise several attributes, which can be grouped in three categories: location, home characteristics and ad characteristics. In particular, the features involved in determining the location of the real estate asset are the following:

- **Zone**: division within the Salamanca district where the asset is located. This zone is determined by Idealista based on the asset location.
- **Postal code**: the postal code for the area where the asset is located.
- **Street name**: the name of the street where the asset is located.
- **Street number**: number within the street where the asset is located.
- **Floor number**: the number of the floor where the asset is located.

The features defining the asset characteristics are the following:

- **Type of asset**: whether the asset is an apartment (flat, condo...) or a villa (detached or semi-detached house).
- **Constructed area**: the total area of the asset indicated in square meters.
- **Floor area**: the floor area of the asset indicated in square meters, which will in some cases be smaller than the constructed area, because terraces, gardens, etc. are ignored in this feature.
- **Construction year**: construction year of the building.
- **Number of rooms**: the total number of rooms in the asset.
- **Number of baths**: the total number of bathrooms in the asset.
- **Is penthouse**: whether the asset is a penthouse.
- **Is duplex**: whether the asset is a duplex, i.e., a house of two floors connected by stairs.
- **Has lift**: whether the building where the asset is located has a lift.
- **Has box room**: whether the asset includes a box room.
- **Has swimming pool**: whether the asset has a swimming pool (either private or in the building).
- **Has garden**: whether the asset has a garden.
- **Has parking**: whether the asset has a parking lot included in the price.
- **Parking price**: if a parking lot is offered by an additional cost, then that price is specified in this feature.
- **Community costs**: the monthly fee for community costs.

Finally, the features defining the ad characteristics are the following:

- **Date of activation**: the date when the ad was published in the listing and displayed publicly.

Table 1. Data description, showing the range of values for each feature and the number of empty values, as well as mean and standard deviation in the case of numerical values.

Feature	Type	Range	Mean (Std. Dev.)	Empty values
Zone	Categorical	1, 2, 3, 4, 5, 6	–	–
Postal code	Categorical	28001, 28006, 28009, 28014, 28028, 28046	–	–
Street name	Categorical	65 values	–	1,453
Street number	Categorical	77 values	–	2,049
Floor number	Categorical	Basement, Floor, Mezz, 1–14	–	119
Type of asset	Categorical	Apartment, Villa	–	–
Constructed area	Numerical	50–2,041 sq.m.	288.76 (133.71)	–
Floor area	Numerical	93–1,700 sq.m.	257.63 (126.43)	1,673
Construction year	Numerical	1848–2018	1953.23 (31.35)	1,517
Number of rooms	Numerical	0–20	4.19 (1.35)	6
Number of baths	Numerical	0–10	3.53 (1.14)	5
Is penthouse	Boolean	T (169) / F (288)	–	1,809
Is duplex	Boolean	T (50) / F (260)	–	1,956
Has lift	Boolean	T (2,123) / F (20)	–	123
Has box room	Boolean	T (1,212) / F (269)	–	785
Has swimming pool	Boolean	T (127) / F (413)	–	1,726
Has garden	Boolean	T (155) / F (391)	–	1,720
Has parking	Boolean	T (687) / F (77)	–	1,502
Parking price	Numeric	115–750,000	52,359.50 (102,670)	2,209
Community costs	Numeric	0–3,000	353.71 (299.61)	1,536

- **Date of deactivation**: the date when the ad was removed from the listing, which could indicate that it was already sold or rented.
- **Price**: current price of the asset, or price when the ad was deactivated, which in most cases corresponds to the price at which the asset was sold or rented.

Some of the previous features might not be available. This will happen in most cases because the seller has not explicitly specified some information about the asset, for example, whether a lift is available, which are the monthly community costs, etc. Regarding the street name and number, it can be intentionally hidden by the seller, so potential buyers are not aware of the actual location of the building. Some information is always available, such as the zone, postal code, constructed area, number of rooms and number of bathrooms.

In the dataset, there is a total of 2,266 real estate assets, from which 2,174 are apartments and 92 are villas. The assets prices range between 1 and 90 million euros, with an average price of about 2.02 million and a median price of about 1.66 million. More information about the data and the range of values is available in Table 1.

3.2. Data Cleansing

Before starting to work with the data, we have proceeded to clean some of the data. Although the data is relatively clean, some information is particularly noisy because the way it is communicated by sellers in the listing.

An example is the street name, which is a free text field. For this reason, there was no consensus regarding some aspects such as capitalization, prepositions or accents. We have manually revised this field in order to standardize it according to the actual street names gathered in the streets guide.

The floor area is not specified in most of the cases. In this case, we have assumed it to be equal to the constructed area. Even when this is a rough estimate, it is often the case than the floor area is only slightly smaller than the constructed area, and therefore we have considered this estimation to be an appropriate proxy.

Finally, for all binary features, we have considered empty values as non-availability of the corresponding features. This seems reasonable within the scope of online listings, since it is more likely that the seller fills the field when the feature is available. For example, in the case of swimming pools, 127 ads specify that the house features a pool, whereas 413 explicitly state that the asset does

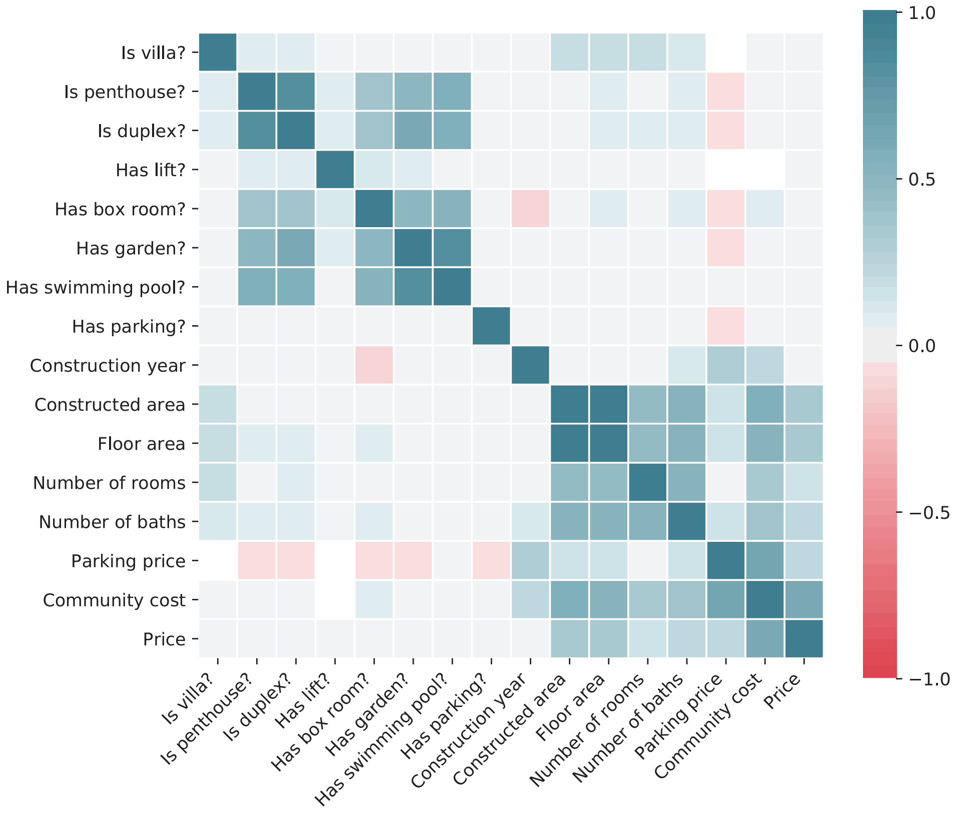

Figure 4. Matrix of correlations between each pair of variables.

not include a swimming pool. We will consider that for those ads where no information is provided about the pool (in this case, a total of 1,726), then such a pool is not available.

3.3. Exploratory Data Analysis

Before proceeding with the training of machine learning models, we are interested in getting some insights about the data at hand. In particular, it would be interesting to know how different variables affect the price of the real estate model, to understand their potential quality as predictors. This information is shown in Figure 4, where the correlation between each pair of variables (considering only binary and continuous variables) is shown. The Pearson correlation coefficient is computed for continuous variables, and the point biserial correlation coefficient is computed for binary variables.

From the correlation matrix we can see that the main variables affecting the asset price are those related to the house size, mainly the constructed area. Of course, all these variables, such as the area, number of bedrooms and number of bathrooms, are highly correlated among them. There is also a significant impact of the monthly community cost, but in this case it is likely that the causality points in the other direction: the cost grows for those houses which are more expensive. Interestingly, most of the binary features as well as the construction year do not seem to have either positive or negative correlation with the price.

Figure 5 shows the distribution of asset prices based on their location, more specifically, based on their zone attribute provided by Idealista. Although differences are not very noticeable, it can be seen how assets in zone 1 have a substantially larger mean and median price when compared to the other zones. Zones 2 and 3 have slightly lower prices than the rest of the zones.

To understand whether these variations can be explained because of the zone and not because of other factors, we have also plotted the distribution of constructed areas per each zone, which can be seen in Figure 6. In this figure we can see how there are not remarkable differences between different zones, with the exception of zone 4 where the median area is noticeably larger. When comparing the

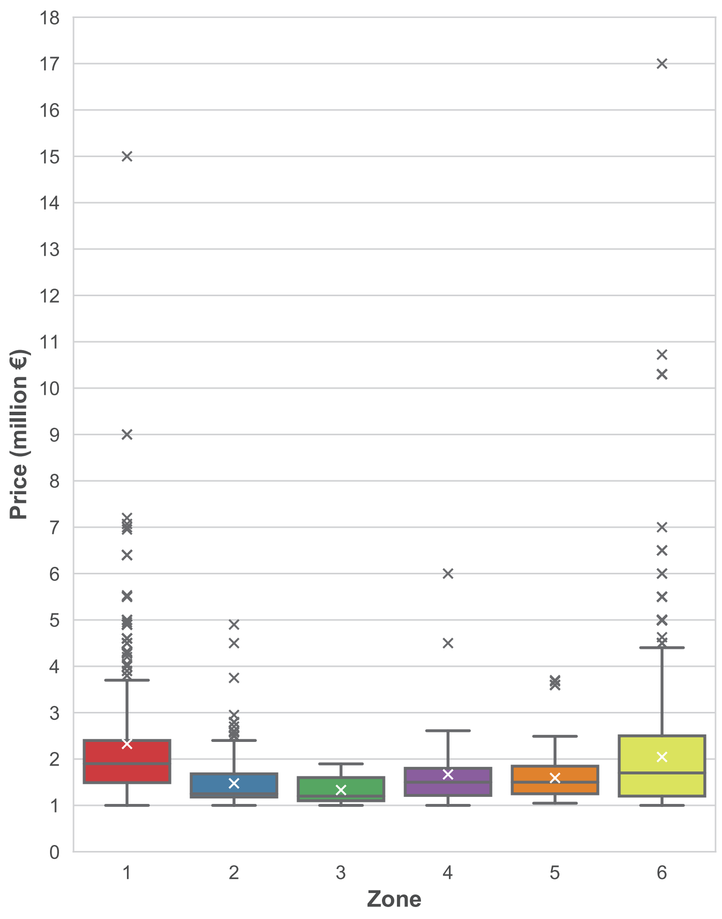

Figure 5. Boxplot of the distribution of prices based on the asset location.

distributions of areas with the distribution of prices, we can conclude that the asset location is also a factor when determining the price, beyond the constructed area itself. For example, it seems that zone 1 is more expensive in average, since prices are larger while areas follow a similar distribution than in other areas. The opposite happens with zone 4, where prices are roughly the same but constructed areas are larger.

Finally, we are interested in checking how a linear regression model can fit the prices based only on the constructed area. We plot the regression model on Figure 7, where two models are actually plotted, one per each type of asset. Assets with prices larger than 8 million have been removed from the figure to ease readability, since these assets are a minority. We can see how only the constructed area is insufficient to accurately fit the prices, and therefore a more complex model involving more predictions must be built for achieving successful prediction.

4. Machine Learning Proposal

In the previous section, we have seen how different variables correlate with the asset price, with the constructed area being the attribute with a larger influence on the price. We have also graphically seen how a linear regression model considering only the constructed area is insufficient for successfully predicting the price.

In this work, we will use machine learning to build and train more complex models for carrying out price prediction. This problem will be tackled as a multi-variate regression problem, with the price being the output, where features can be binary (such as whether there is a swimming pool or a lift), categorical (such as the asset location) or continuous (such as the constructed area). To be able to use many machine learning techniques, we will convert all categorical features using one-hot encoding.

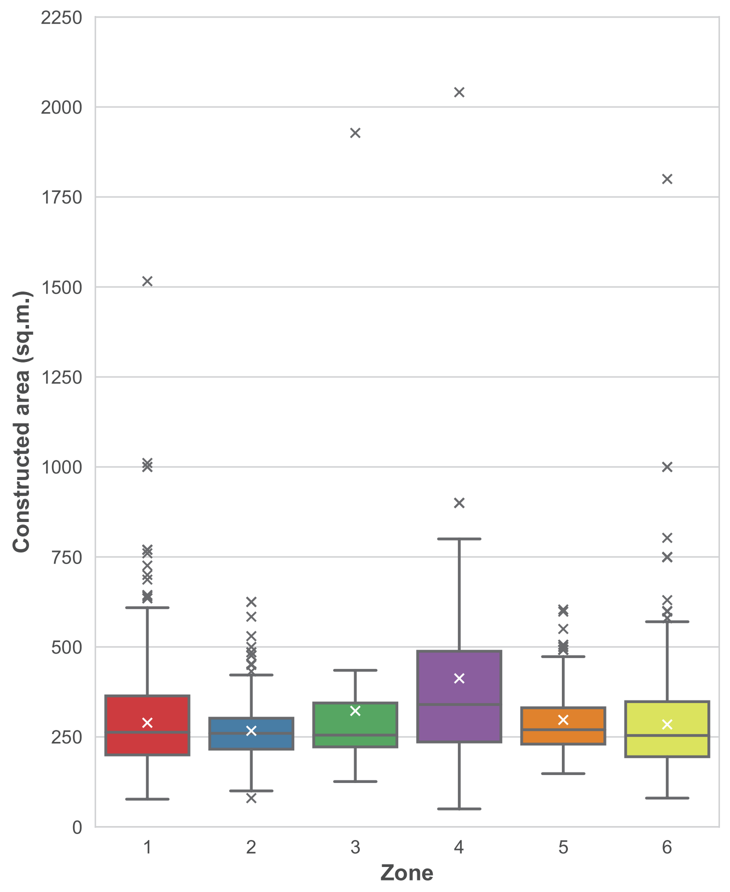

Figure 6. Boxplot of the distribution of constructed areas based on the asset location.

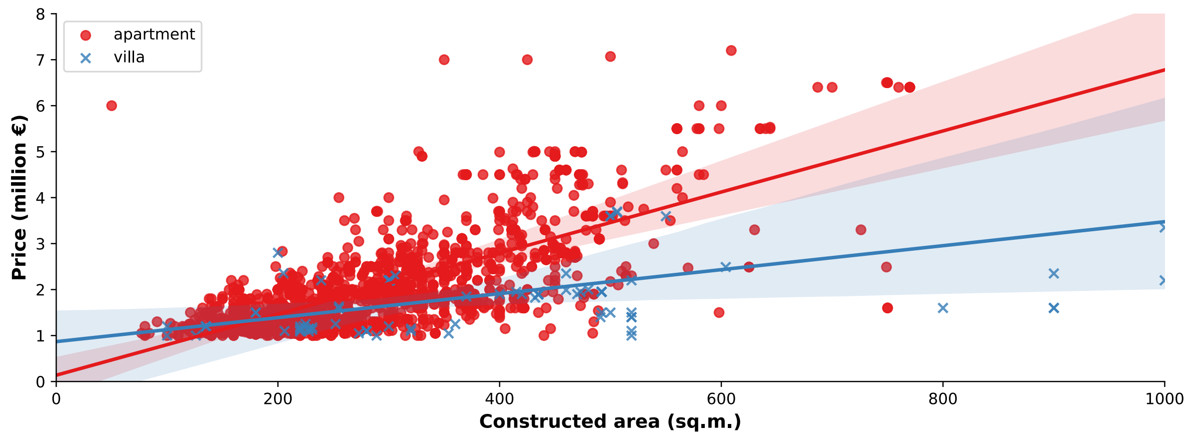

Figure 7. Linear regression between the constructed area and the asset price, based on the asset type.

This means that any categorical feature with N values will be transformed into a total of N binary features.

Also, we know that the values for some of the features remain unknown in some instances. This happens with the street name, the street number, the construction year, the parking price and the community costs. Since the first feature is categorical, the problem of null values is resolved after performing one-hot encoding (since all binary features can take the value “false"). As for the other variables, we have decided to remove them from the data. This is especially important for the street number, since it is not a relevant feature but could lead to overfitting (in some cases, the street name and number identifies an asset).

Bearing in mind with these constraints, we have finally considered the use of four different machine learning techniques:

- Support vector regression: this method is also known as a "kernel method", and constitutes an extension of classical support vector machine classifiers to support regression [16]. Kernel methods transform data into a higher dimensional state where data can be separable by a certain function, then learning such function to discriminate between instances. When applied to regression, a parameter epsilon (ϵ) is introduced, thus aiming that the learned function does not deviate from the real output by a value larger than epsilon for each instance.
- k-nearest neighbors: this is an example of a geometric technique to perform regression. In this technique, data is not really used to build a model, but is rather considered a "lazy" algorithm, since it needs to traverse the whole learning set in order to carry out prediction for a single instance. In particular, what k-nearest neighbors does is compute the distance of the instance to be predicted to all of the instances in the learning dataset, based on some distance function or metric (such as Euclidean or cosine distance). Once distances are computed, the k instances in the training set which are closest to the instance subject of prediction will be retrieved, and their known output will be aggregated to generate a predicted output, for example, by computing their average. This method is interesting since it will consider assets similar to the one we want to predict, but can be problematic when dealing with high-dimensional data with binary attributes.
- Ensembles of regression trees: regression trees are logical models that are able to learn a set of rules in order to compute the output of an instance given its features. A regression tree can be learned from a training dataset by choosing the most informative feature according to some criterion (such as entropy or statistical dispersion) and then dividing the dataset based on some condition over that feature. The process is repeated with each of the divided training subsets until a tree is completely formed, either because no more features are available to be selected or because fitting a regression model on the subset performs better than computing a regression sub-tree. In this paper we will use ensembles of trees, which means that several models will be combined in order to reduce regression bias. To build the ensemble, we will use the technique known as extremely randomized trees, introduced by Geurts et al. [17] as an improvement to random forests.
- Multi-layer perceptron: the multi-layer perceptron is an example of a connectionist model, more particularly an implementation of an artificial neural networks. This kind of models comprise an input layer that receives as input the values for the features of each instance and several hidden layers, each of which features several hidden units, or neurons. The model is fully connected, meaning that each neuron from one layer is connected to every single neuron in the following layer. Each connection has a corresponding floating number, called weight, which serves for aggregating the inputs to the neuron, and then a non-linear activation function is applied. During training, a gradient descent algorithm is used to fit the connections weights via a process known as backpropagation.

These algorithms can be configured based on different parameters, which can have an impact on performance. In the following section, we describe the experimental setup along with the different instantiations of parameters for testing each of the previously listed machine learning algorithms.

5. Evaluation

In this section, we will first introduce the experimental setup which has been applied for testing different machine learning regression models. Then, we will describe the results and discuss relevant findings.

5.1. Experimental Setup

As described in the previous section, we will test the performance of four different machine learning techniques. When assessing how these techniques perform, we must be aware that some

Table 2. Configuration in scikit-learn of the different machine learning algorithms that will be used for addressing the regression problem.

ML algorithm	Parameter	Values
Support vector regression `svm.SVR`	Kernel type (`kernel`)	Radial basis function kernel (`rbf`)
k-nearest neighbors `neighbors.KNeighborsRegressor`	Number of neighbors (`n_neighbors`)	5, 10, 20, 50
	Distance metric (`metric`)	Minkowski, cosine
	Weight function (`weights`)	Uniform, inverse to distance (`distance`)
Ensembles of regression trees `ensemble.ExtraTreesRegressor`	Number of trees in the forest (`n_estimators`)	10, 20, 50
	Criterion for split quality (`criterion`)	Mean absolute error (`mae`), mean squared error (`mse`)
	Whether bootstrap samples are used (`bootstrap`)	True, false
Multi-layer perceptron `neural_network.MLPRegressor`	Network architecture (`hidden_layer_sizes`)	1024, 256–128, 128–64–32

of them can be configured according to different parameters. In this paper, we have considered the following:

- Support vector regression: we will specify the kernel type.
- *k*-nearest neighbors: we will configure the number of neighbors to consider, the distance metric and the weight function used for prediction.
- Ensembles of regression trees: we will set up the number of trees that conform the forest, the criterion for determining the quality of a split and whether or not bootstrap samples are used when building trees.
- Multi-layer perceptron: we will consider different architectures, i.e., different configurations of how hidden units are distributed among layers.

For training machine learning models, we will use the scikit-learn package for Python [18], in its version 0.19.2 (the latest as of August 2018). Table 2 describes the different configurations considered for training the machine learning models. Also, the canonical names of the classes and the attributes in the scikit-learn package are shown to ease reproducibility of the experiments. In the case of the multi-layer perceptron, each number indicates the number of neurons in one layer, therefore the configuration “256–128” means 256 units in the first hidden layer and 128 units in the second hidden layer.

Additionally, some machine learning techniques perform better when data is normalized. In this case, we will test out all techniques both with data normalized in the range [0, 1] and not normalized. In order to normalize data, we have divided continuous features by the maximum value in the training set. Of course, it could happen that some values in the test are larger than one, but that will not constitute an issue. Normalizing data is important for some techniques to work properly: it is the case of neural networks, where big values can lead to the vanishing or exploding gradient problem.

From the four techniques described before, two of them have an stochastic behavior: it is the case of the ensembles of regression trees (where trees in the forest are built based on some random factors) and of the multi-layer perceptron, where weights are initialized randomly. In order to reduce bias and obtain significant results, we have run each of the experiments involving these techniques a total of 30 times.

Finally, in order to prevent biased results when sampling the dataset in order to build the train and test sets, we will use 5-folds cross validation. In this approach, the whole dataset is first randomly shuffled and then splitted into five equally sized folds. Five experiments will then be carried out: one for each different fold used as the test set. The training set will always be made out of the remaining four folds. When reporting results in the following section, all metrics will refer to the average of the cross validation, i.e., will be computed as the average of the five results obtained for each of the test sets (macro-average).

The total number of experiments that will be carried out is equal to 10 for the support vector regression, 160 for the *k*-nearest neighbors, 3,600 for the ensembles of regression trees and 900 for the multi-layer perceptron. It is worth noting that in all techniques, every combination of setups is run ten times: once per fold both with normalized and not normalized data. In the latter two techniques, each of these experiments is run 30 times. Therefore, the total number of experiments is 4,670.

5.2. Results and Findings

In this section, we will explore the performance achieved by different machine learning models. The following quality metrics for regression have been computed, which are provided by the scikit-learn API:

- Explained variance regression score, which measures the extent to which a model accounts for the variation of a dataset. Letting $\hat{y}$ be the predicted output and y the actual output, this metric is computed as follows:
$$E_{var}(y,\hat{y}) = 1 - \frac{Var\{y-\hat{y}\}}{Var\{y\}}$$
In this equation, Var is the variance of a distribution. The best possible score is 1.0, which would occur when $y = \hat{y}$.
- Mean absolute error, which computes the average of the error for all the instances, computed as follows:
$$MAE(y,\hat{y}) = \frac{1}{n}\sum_{i=0}^{n}|y_i - \hat{y}_i|$$
Since this is an error metric, the best possible value is 0.
- Median absolute error, similar to the previous score but computing the median of the distribution of differences between the expected and actual values:
$$MedAE(y,\hat{y}) = median(|y_1 - \hat{y}_1|, \ldots, |y_n - \hat{y}_n|)$$
Again, since this is an error metric, the best possible value is 0.
- Mean squared error, similar to MAE but with all errors squared, and therefore computed as follows:
$$MSE(y,\hat{y}) = \frac{1}{n}\sum_{i=0}^{n}|y_i - \hat{y}_i|^2$$
As it happened with MAE, since this is an error metric, the best possible value is 0.
- Coefficient of determination (R^2), which provides a measure of how well future samples are likely to be predicted. It is computed using the following equation:
$$R^2(y,\hat{y}) = 1 - \frac{\sum_{i=0}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n}(y_i - \bar{y})^2}$$
In the previous equation, $\bar{y}$ refers to the average of the real outputs. The maximum value for the coefficient of determination is 1.0, which would be obtained when the predicted output matches the real output for all instances. R^2 would be 0 if the model always predicts the average output, but it can also hold negative values, since the model can work arbitrarily worse than just predicting the estimated value.

In order to discuss the results, we will address relevant findings which can be extracted from the data.

Which model performs best?

When considering the mean absolute error for determining the best performers, we notice that ensembles of regression trees work systematically better than other techniques. In particular, the top

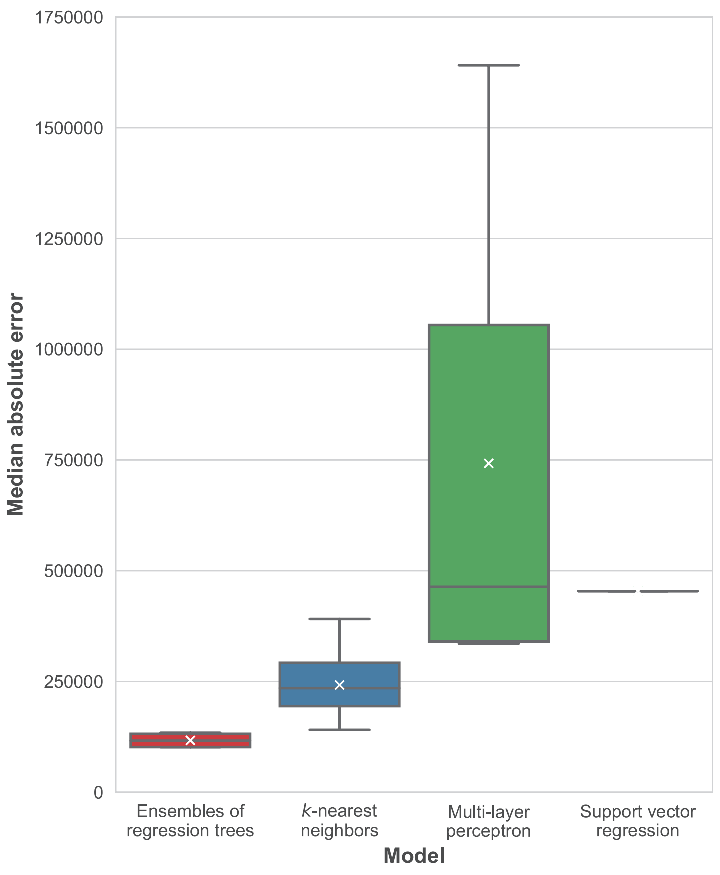

Figure 8. Boxplot of the distribution of median average errors per machine learning technique.

ten models are always ensembles, with either 10, 20 or 50 estimators. The split criterion, bootstrap instances and normalization neither seem to have a significant effect on performance. The smallest mean absolute error is 338,715 euros (a relative error of 16.80%). Consistent results are found when considering the median absolute error, although best performers are not sorted on a side-by-side basis (i.e., the model with the best MAE do not correspond to the model with the best MedAE). In this case, all top-ten models are also ensembles of regression trees. The best median absolute error is 94,850 euros (a relative error of 5.71%).

The distribution of best median average errors per machine learning technique is shown in Figure 8. In this figure, we can see how ensembles of regression trees significantly outperform all of the other techniques, followed by *k*-nearest neighbors, support vector regression and the multi-layer perceptron. In the case of support vector regression, only one configuration was tested and the model is deterministic, for this reason there is a single result instead of a distribution. As for the multi-layer perceptron, a very large variance can be seen in the distribution of results, along with a remarkable difference between the average and the median MedAE. Worst MAE and MedAE are always reported by the multilayer-perceptron with a single layer with 1024 units. Because of the relatively small availability of data, this model could be affected by overfitting, although further testing is required to confirm this diagnosis.

Regarding the explained variance regression score and the coefficient of determination, both metrics are highly correlated. The best model when assessed based on R^2 achieves a value of about 0.46 for both metrics.

How are models affected by setup?

The impact of parameters in the median absolute error for the different machine learning techniques is shown in Figures 9, 10 and 11 for the ensembles of regression trees, the *k*-nearest neighbors and the multi-layer perceptron respectively.

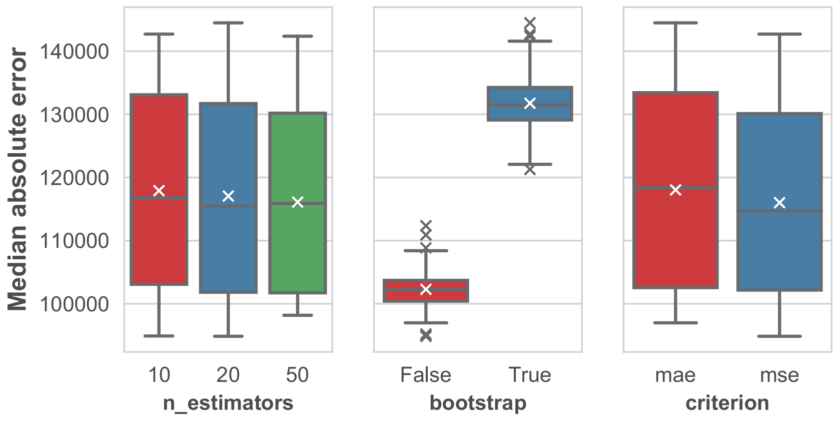

Figure 9. Distribution of the median average error based on the different parameters for the ensembles of regression trees.

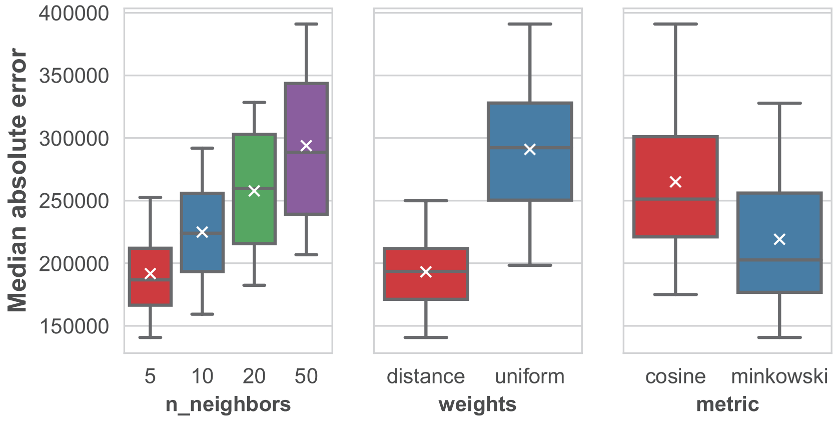

Figure 10. Distribution of the median average error based on the different parameters for the *k*-nearest neighbors.

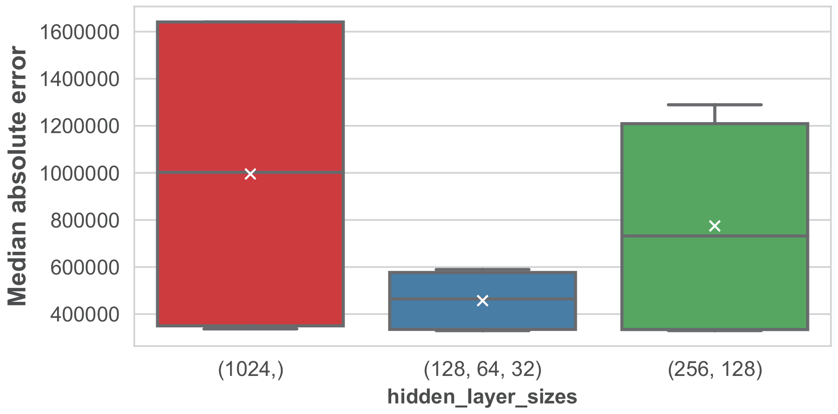

Figure 11. Distribution of the median average error based on the different parameters for the multi-layer perceptron.

As we can see, in the case of ensembles of regression trees the number of trees in the forest has little impact in the performance. It seems that the mean MedAE slightly decreases as the number of estimators grow, but the variation is very small. Variance also suffer a reduction as a result of introducing more trees, although again this effect is barely perceptible. In the case of the use of bootstrap sampling, it seems clear that its use negatively affects the model performance in a significant manner. As for the split criterion, a slight reduction in MedAE follows the use of the median squared error (MSE), but as it happened with the number of estimators the effect is small.

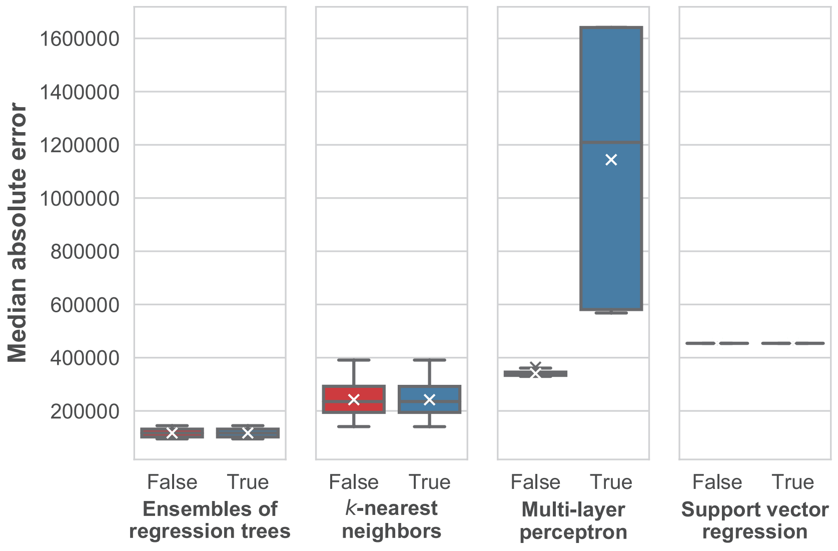

Figure 12. Distribution of the median average error based on the use of normalization for the different machine learning techniques.

When studying the number of neighbors, a clear pattern is shown where the error grows proportionally to the number of considered neighbors. Also, the variance in MedAE grows as well. This is indicating that considering more similar instances is not only increasing variability but also adding confusion to the regression. From the obtained results, it is left as future work to test this technique with a number of neighbors smaller than five (like one or two) to test whether the error keeps descending. Considering the weights of the different features when computing the nearest neighbors, it is clear that weights that are inversely proportional to the distance of the target instance to the neighbors work better than uniform weights. And regarding the distance metric, Minkowski has also reported significantly better results than a cosine distance.

Finally, in the case of the multi-layer perceptron, results show that performance is very sensitive to the architecture. It is interesting to see that the minimum MedAE is constant between different alternatives, but the mean and median MedAE is much smaller as more layers with less units each are introduced. As a result, variance is also affected, and we check that the configuration using three hidden layers with 128 units in the first one is the one reporting the best average MedAE. Nevertheless, as we concluded earlier, errors are always much larger than those obtained using ensembles of regression trees or *k*-nearest neighbors.

When does normalization provide an advantage?

The impact of the use of normalization in the median average error is shown in Figure 12 for the different machine learning techniques considered in this paper.

From this figure, we can see how normalization does not have an impact in the ensembles of regression trees, *k*-nearest neighbors and support vector regression. This occur due to the fact that normalization results in a linear transformation of data. Therefore, in the case of *k*-nearest neighbors, the distance metric is consistent between any pair of instance after the linear transformation of their features. The same thing happens when using support vector machines, and when using such features for building a regression tree.

The only case in which normalization makes a difference is when using a multi-layer perceptron. Interestingly, in this case the median average error increases when normalization is used. This is contrary to the common belief, where normalization is often considered a good practice to prevent numerical instability when training the neural network. In this paper, we will not study the causes of this issue in further detail, since it has been proved that the perceptron is not competitive when compared to other machine learning techniques in the problem of real estate price prediction. Therefore,

Table 3. Average (and standard deviation) of training and prediction times for different machine learning configurations.

ML algorithm	Parameter	Value	Train time (s)	Predict time (s)
Support vector regression	–	–	0.500 (0.022)	0.093 (0.0015)
Ensembles of regression trees	`n_estimators`	`10`	0.999 (1.068)	0.0011 (0.00006)
		`20`	2.037 (2.185)	0.0021 (0.00011)
		`50`	5.192 (5.587)	0.0049 (0.00024)
	`bootstrap`	`true`	1.752 (2.250)	0.0026 (0.0015)
		`false`	3.734 (4.905)	0.0028 (0.0017)
	`criterion`	`mae`	5.337 (4.195)	0.0027 (0.0016)
		`mse`	0.149 (0.103)	0.0027 (0.0016)
k-nearest neighbors	`n_neighbors`	`5`	0.0016 (0.0014)	0.013 (0.0038)
		`10`	0.0017 (0.0015)	0.014 (0.0036)
		`20`	0.0017 (0.0016)	0.015 (0.0031)
		`50`	0.0016 (0.0015)	0.018 (0.0016)
	`weights`	`distance`	0.0017 (0.0015)	0.015 (0.0035)
		`uniform`	0.0016 (0.0014)	0.015 (0.0037)
	`metric`	`cosine`	0.00027 (0.000012)	0.018 (0.0013)
		`minkowski`	0.0030 (0.00038)	0.012 (0.0031)
Multi-layer perceptron	`hidden_layer_sizes`	`(128, 64, 32)`	1.727 (1.104)	0.00076 (0.00013)
		`(256, 128)`	3.045 (1.487)	0.0011 (0.000061)
		`(1024)`	8.376 (0.094)	0.0021 (0.000061)

a more thorough study of the causes, involving how the loss value evolves with the training epochs is left for future work.

How much time do models require to train and run? Training and exploitation times can be a critical aspect of running a machine learning model in a production setting. For this reason, in this paper we have decided to introduce a study of how times vary depending on the machine learning technique and its specific setup. It is worth recalling that times reported in this section refer to training and testing within cross-validation setting, where training is done with 80% of the instances (1,813 assets) and testing with the remaining 20% (453 assets).

The average time, along with the standard deviation, for both training and prediction of the different machine learning models and configurations is shown in Table 3. In general, we can see how *k*-nearest neighbors is a lazy ML algorithm, where the training time is negligible when compared with the prediction time; whereas this effect does not happen in other techniques.

For such reason, *k*-nearest neighbors is the fastest algorithm to train, and no significant differences are found for different setups, with the only exception of the distance metric. The reason of Minkowski requiring more time than the cosine distance is that when Minkowski is used, some pre-processing can be done during training in order to accelerate prediction (such as building a KD-tree or a Ball-tree). As a result, training is faster when this pre-processing is not done (because it is not possible using a cosine distance) but prediction is slower.

Support vector regression's training is relatively fast when compared to the neural network models or ensembles of trees, but prediction time is about two orders of magnitude higher.

In the case of ensembles, it can be seen how the setup has a remarkable effect on both training and prediction time. These times scale linearly with the number of trees, which is an expected behavior. The use of bootstrap instances and the split criterion do not affect the prediction time, but impact significantly the cost of training. This cost is specially sensitive to the training split criterion, where computing the MAE requires about 50 times the time of computing the MSE. This is likely to be due to implementation details of the error computation.

Finally, in the case of the multi-layer perceptron, times are in a similar scale than those of the ensembles. Interestingly, the time in this case decreases as more layers are added, given that the number of units is smaller. This is likely to be due to the implementation, although a deeper study is not within the scope of this paper and is left for future work.

6. Conclusions and Future Work

The real estate market constitutes a good setting for investing, due to the many aspects governing the prices of real estate assets and the variances that can be found when looking at local markets and small-scale niches.

In this paper, we have explored the application of diverse machine learning techniques with the objective of identifying real estate opportunities for investment. In particular, we have focused first in the problem of predicting the price of a real estate asset whose features are known, and have modelled it as a regression problem.

We have performed a thorough cleansing and exploration of the input data, after which we have decided to build machine learning models using four different techniques: ensembles of regression trees, *k*-nearest neighbors, support vector machines for regression and multi-layer perceptrons. Cross-validation of five folds have been used in order to avoid biases resulting from the split in train and test subsets. Because we understand that the parameterization of the different techniques can drive significant variations in the performance, we have identified some of the potentially most influencing parameters and tested different setups for those. We have also reported results on the use of the algorithms after data normalization.

After training and evaluating the models, we have thoroughly studied the results, revealing findings on how different setups impact the performance. Results prove that outperforming models are always those consisting of ensembles of regression trees. In quantitative terms, we have found that the smallest mean absolute error is 338,715 euros, and the best median absolute error is 94,850 euros. These errors can be considered high within the scope of financial investment, but are relatively small under the fact that data comprise only assets with values over one million euros. In fact, when the mean and median absolute errors are compared with the mean and median of the distribution of prices, relative errors of 16.80% and 5.71% are obtained respectively.

In this sense, it is worth mentioning that the fact of the mean being much larger than the median can be explained due to the presence of outliers. For example, the most expensive asset in the dataset is priced 90 million dollars, and according to the description it is an apartment of 473 square meters with five bedrooms and five bathrooms. This price seems to be excessive for an apartment of such characteristics, meaning that either the price is a typo and the asset was sold at a much smaller price, or the description has been recorded incorrectly.

In either case, further analysis of prediction errors, even involving manual assessment of experts, is left for future work which can serve for improving the quality of the database and therefore of the trained machine learning models.

Also, the evaluation of a model built using *k*-nearest neighbors with a number of neighbors smaller than five was left for future work. Additionally, further analysis on the impact of normalization in the performance of the multi-layer perceptron can be of interest, since results reported in this paper seems to contradict the common belief that normalization helps prevent numerical instability and can ease faster convergence. Finally, the impact of different architectures in both regression error and time are worth exploring.

Another field for potential research involves the use of deep learning techniques for extracting relevant features from natural language descriptions. So far, these data have not been provided to us, but all adverts have a description introduced by the seller describing the home at sale. A feature vector extracted from this text, for example by using convolutional neural networks with temporal components, could add a remarkable value to the features already known.

Finally, in this paper we have approached the problem of identifying investment opportunities as a regression problem consisting on the estimation of the actual appraisal of the assets. However, if this valuation were done manually, then the problem could be tackled as a binary classification problem, where the objective would be to determine whether the asset itself is an investment opportunity; for example, if the sale price were smaller than the valuation price. Exploring this research line is also suggested as a future work.

Data Statement

This work has relied on data provided by Idealista, consisting on the list of real estate assets for sale in the platform in the Salamanca district (Madrid, Spain) during the second semester of 2017 (between July 1st and December 31st, both inclusive). Due to our non-disclosure agreement with Idealista, we are not allowed to publish or redistribute the dataset used for this work. For authors aiming at reproducing experiments in this paper, we suggest contacting Idealista using the form available in the following address: https://www.idealista.com/data/empresa/sobre-nosotros.

Conflicts of Interest

Some of the knowledge obtained from this paper or the machine learning models built in this work might be implemented and/or commercialized in the future by Rentier, a company to which all authors in the present work are affiliated. The authors declare no other conflicts of interest.

Acknowledgment

The authors would like to thank Beatriz Cámara and Alfonso Lozano from Idealista for their collaboration in providing access to the data used in this paper and support.

References

1. Teuben, B.; Bothra, H. Real Estate Market Size 2017 – Annual Update on the Size of the Professionally Managed Global Real Estate Investment Market. Technical report, MSCI, Inc., 2018. Available online: https://www.msci.com/documents/10199/6fdca931-3405-1073-e7fa-1672aa66f4c2.
2. Idealista. Índice Idealista 50: Evolución del Precio de la Vivienda de Segunda Mano en España, 2018. Available online: https://www.idealista.com/news/estadisticas/indicevivienda#precio.
3. Jiang, L.; Phillips, P.C.B.; Yu, J. A New Hedonic Regression for Real Estate Prices Applied to the Singapore Residential Market. Technical report, Cowles Foundation Discussion Paper No. 1969, 2014. Available at SSRN: https://ssrn.com/abstract=2533017.
4. Jiang, L.; Phillips, P.C.; Yu, J. New Methodology for Constructing Real Estate Price Indices Applied to the Singapore Residential Market. *Journal of Banking and Finance* **2015**, *61*, S121–S131.
5. Greenstein, S.M.; Tucker, C.E.; Wu, L.; Brynjolfsson, E. The Future of Prediction : How Google Searches Foreshadow Housing Prices and Sales The Future of Prediction How Google Searches Foreshadow Housing Prices. In *Economic Analysis of the Digital Economy*; The University of Chicago Press, 2015; pp. 89–118.
6. Sun, D.; Du, Y.; Xu, W.; Zuo, M.; Zhang, C.; Zhou, J. Combining Online News Articles and Web Search to Predict the Fluctuation of Real Estate Market in Big Data Context. *Pacific Asia Journal of the Association for Information Systems* **2015**, *6*, 19–37.
7. Zurada, J.; Levitan, A.; Juan, G. Non-Conventional Approaches to Property Vale Assessment. *Journal of Applied Business Research* **2016**, *22*, 1–14.
8. Guan, J.; Shi, D.; Zurada, J.M.; Levitan, A.S. Analyzing Massive Data Sets: An Adaptive Fuzzy Neural Approach for Prediction, with a Real Estate Illustration. *Journal of Organizational Computing and Electronic Commerce* **2014**, *24*, 94–112.
9. Sarip, A.G.; Hafez, M.B.; Nasir Daud, M. Application of Fuzzy Regression Model for Real Estate Price Prediction. *Malaysian Journal of Computer Science* **2016**, *29*, 15–27.
10. Del Giudice, V.; De Paola, P.; Cantisani, G. Valuation of Real Estate Investments through Fuzzy Logic. *Buildings* **2017**, *7*, 26.
11. Rafiei, M.H.; Adeli, H. A Novel Machine Learning Model for Estimation of Sale Prices of Real Estate Units. *Journal of Construction Engineering and Management* **2016**, *142*, 04015066.
12. Park, B.; Kwon Bae, J. Using Machine Learning Algorithms for Housing Price Prediction: The Case of Fairfax County, Virginia Housing Data. *Expert Systems with Applications* **2015**, *42*, 2928–2934.
13. Manganelli, B.; Paola, P.D.; Giudice, V.D. Linear Programming in a Multi-Criteria Model for Real Estate Appraisal. International Conference on Computational Science and its Applications, Part I, 2016, Vol. 9786, *Lecture Notes in Computer Science*, pp. 182–192.

14. Del Giudice, V.; De Paola, P.; Forte, F. Using Genetic Algorithms for Real Estate Appraisals. *Buildings* **2017**, *7*, 31.
15. Del Giudice, V.; De Paola, P.; Forte, F.; Manganelli, B. Real Estate appraisals with Bayesian approach and Markov Chain Hybrid Monte Carlo Method: An Application to a Central Urban Area of Naples. *Sustainability* **2017**, *9*, 2138.
16. Smola, A.J.; Schölkopf, B. A Tutorial on Support Vector Regression. *Statistics and Computing* **2004**, *14*, 199–222.
17. Geurts, P.; Ernst, D.; Wehenkel, L. Extremely Randomized Trees. *Machine Learning* **2006**, *63*, 3–42.
18. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **2011**, *12*, 2825–2830.