Universidad de San Andrés
Departamento de Economía
Maestría en Economía

Selection of optimal window size and ensemble methods for rental price estimation with XGBoost: Evidence from Buenos Aires

Juan Bautista SOSA

Supervisors: Paula MARGARETIC & Walter SOSA ESCUDERO

SELECTION OF OPTIMAL WINDOW SIZE AND ENSEMBLE METHODS FOR RENTAL PRICE ESTIMATION WITH XGBOOST: EVIDENCE FROM BUENOS AIRES

Abstract

In the context of forecasting rental prices of residential properties, the fact that the relationship between rents and property characteristics may be subject to significant changes over time can hinder the predictive performance of models that do not adjust to these changes. Previous studies have tested the accuracy of different statistical models in predicting rent prices, but none has focused on how the definition of the estimation window of observations used in the training stage can affect predictive performance. This paper explores the impact of selecting different window sizes in a forecasting experiment of rent prices in the City of Buenos Aires from 2020 to 2022 using the machine learning algorithm XGBoost. The results obtained for out-of-sample one-month-ahead forecasts indicate that the decision to use an expanding window or rolling windows is not trivial in terms of the achieved predictive performance. Among a set of rolling window sizes, the model with a size of 6 months yields the lowest error in terms of root mean square error and mean absolute error. Additionally, three methods that dynamically combine the predictions of models with different window sizes are tested. An ensemble method that outputs a weighted average of the predictions using time-varying weights based on the inverse of previous forecasting errors emerges as the superior strategy, producing a mean absolute percentage error of 19.7%. Overall, these results underscore the need to take into account the temporal dimension when selecting observations for training if the goal is to maximize out-of-sample predictive performance.

Keywords: House prices · Adaptive learning · XGBoost

I thank Máximo Ripani, Manuel Der and Sebastián Grill for their insightful comments.

1 Introduction

Accurately estimating the market rental price of residential properties is a relevant problem for many actors in an economy. It can inform the decisions of investors who look to maximize the returns of their investments in real estate, aid financial planning for home-buyers, and help local governments assess housing affordability in their jurisdictions. This paper investigates how considering the temporal dimension in the selection of the sample used to train a model to predict rent prices can impact its predictive accuracy.

There is an extensive literature on real estate valuation based on properties' characteristics and location, guided mainly by the theory of the hedonic pricing model (Rosen, 1974). This microeconomic model is based on the conception that the price of certain goods is equal to the sum of the prices of its characteristics. In the case of real estate, property prices can be conceived as the sum of the valuation of relevant amenities, such as the number of rooms of the property, and dis-amenities, such as the distance to the nearest green space.

A standard methodological approach to the problem of estimating the price of a property based on its characteristics consists of using linear regression models. In the real estate market, these models have commonly been used as a method of indirect valuation of desirable amenities such as sea view (Fleischer, 2012), energy efficiency (Fuerst et al., 2015), and green spaces (Trojanek et al., 2018), as well as to estimate the negative effect that variables such as noise pollution (Zheng et al., 2020) and air pollution (Amini et al., 2022) have on property prices. Some studies use a modified version of the linear regression model to account for the presence of spatial dependence in the dependent variable or the error term and ensure unbiased and efficient estimators (Anselin & Le Gallo, 2006; Bisello et al., 2020; Margaretic & Sosa, 2023).

The aforementioned methods are useful for making inference on the estimated parameters, but they are not characterised by having considerable predictive capacity over other methods. Numerous studies have shown that supervised machine learning models consistently outperform linear regression models in terms of predictive capacity in settings where the goal is to predict property prices (Abidoye et al., 2019; Antipov & Pokryshevskaya, 2012; Pérez-Rave et al., 2019; Rico-Juan & Taltavull de La Paz, 2021). The different machine learning models that have been used for this purpose include Artificial Neural Networks (Xu & Zhang, 2021), Support Vector Machines (Ho et al., 2021; Lahmiri et al., 2023), and tree-based methods such as Random Forest (Carranza et al., 2019; Levantesi & Piscopo, 2020) and gradient boosting algorithms (Baur et al., 2023; Rampini & Re Cecconi, 2022), among others. Their superior performance can be attributed to their ability to capture non-linearities in the relationship between property prices and the variables that explain them, as well as to exploit the bias-variance trade-off (Breiman, 2001). The present study uses an implementation of gradient boosting algorithms on decision trees called XGBoost (Chen & Guestrin, 2016), which is adapted to improve out-of-sample predictive performance by adding regularization to the objective function, to predict rent prices from a rich data set of 168,423 property listings published in an e-commerce platform in the City of Buenos Aires between 2020 and 2022.

In most of the existing literature regarding the prediction of real estate prices at the property-level, it is uncommon to frame the relationship between the price of a property and its characteristics as one that is affected intimately by the temporal dimension. After all, most of the characteristics of residential properties only change slowly (like the socio-economic conditions of neighbourhoods) or are even time-invariant (like many of the physical aspects of the buildings). When the interest lies on inference about parameters in a model that explains prices, it is reasonable to focus on the estimate of the average effect of one particular independent variable on a relatively large period of time, ignoring the possible fluctuations that may be observed in the short run. Also, when the interest lies on exploring the importance that certain variables have for predicting property prices, it is also usual in the literature to adopt an *ex-post* stance and make conclusions out of a relatively long period of time, ignoring the time-varying nature of the relationship. Hence, despite the proliferation of studies that predict property prices, very few of them explicitly incorporate the time-dimension in their estimations.

There are some studies that have employed traditional time-series methods or machine learning models to forecast the evolution of an aggregated index of property sale prices (Abidoye et al., 2019; Levantesi & Piscopo, 2020; Xu & Zhang, 2021) and rent prices (Stevenson, 2007) in different locations. Some other studies have used the forecasts of price indices as a tool to predict the prices of individual properties (Glennon et al., 2018). However, these aggregate measures cannot capture the rich time-varying relationship between the price of a property and the relevant characteristics that determine its price. The focus of the current study is to predict the rent prices of individual properties based on their characteristics and their surroundings, which allows exploiting the full extent of the information contained in the data set of property listings.

In situations where the focus is exclusively on the accurate prediction of the dependent variable, ignoring the time-varying nature of the data can be detrimental for various reasons. Firstly, in addition to the temporal fluctuations in the average price per square meter that are regularly observed across cities, there are reasons to believe that the way in which properties' amenities or dis-amenities are valued in the real estate market is also subject to change across time. For example, there is evidence that as a result of the 2020 COVID-19 pandemic and the consequent surge in remote work, the valuation of private yard space in residential buildings increased significantly (Malik et al., 2023). Similarly, by including year-dummies in an estimation of rent prices with machine learning, Lorenz et al. (2022) report estimates of the marginal effect of the variables "distance to the city center" and "distance to the nearest department store" that vary across the different years in their sample. In light of these results, it is possible that not considering the dynamic changes in the valuation of properties' characteristics can harm predictive performance. Although the current paper does not focus specifically on how the market valuation of specific property characteristics have changed over time, it uses an adaptive learning strategy that deals with this issue to improve predictive performance. By repeatedly retraining the model with more recent observations and discarding old ones, the parameters of the explanatory variables are updated constantly to adapt to these changes.

Secondly, even in a situation where the underlying valuation of properties' amenities and dis-amenities does not change with time, the sample distribution of the characteristics of the observations available to the analyst at the moment of estimation may change significantly from period to period, either by chance or due to an omitted variable that affects the data-gathering process. In either case, if the analyst's goal is to maximize predictive accuracy on the sample obtained at each upcoming period rather than having unbiased estimators of the population parameters (as it happens in many real-world forecasting applications), the methodological strategy adopted should take the changing characteristics of the available training and test samples into account.

This paper contributes to the established machine learning literature on adaptive learning strategies to deal with the issue of concept drift, which refers to a situation in which the relation between the dependent variable and the independent variable is subject to fluctuations across time. Gama et al. (2014) provides a survey of various methodologies used to adapt to concept drift in online supervised learning applications. One adaptive learning strategy used to deal with concept drift consists of shortening the predictive model's "memory" or window of estimation. This refers to the the length of the time-frame from which observations used to train a model are obtained. This window can be shortened through a forgetting mechanism: discarding old observations that were obtained before a certain amount of periods in the past from the target period.

The present study uses a forgetting mechanism which consists of defining a rolling window with a fixed size, so that, for the predictions for rent prices made in each new period, the property listings from the most recent past period are added to the training set, and the property listings from the oldest period are discarded. This strategy is compared to an estimation strategy that has no forgetting mechanism, meaning that all available observations before the target period are used to train the model, regardless of how far in the past they were observed. This is commonly refer to as using an expanding window, given that the amount of periods that are used to train the model increases with time. This paper provides evidence regarding the relative advantage of using a rolling window over an expanding window. It finds that, in the context of forecasting one-month-ahead rent prices in the City of Buenos Aires, a rolling window outperforms the expanding window in terms of root mean square error (RMSE) and mean absolute error (MAE), but not in terms of mean absolute percentage error (MAPE).

Similarly, there is a long standing literature in the field of time-series econometrics devoted to dealing with structural breaks. These refer to a change in the parameters of a model that occurs in a certain period of time. Much of this literature is focused on the precise estimation of the timing and size of the breaks (Chow, 1960; Hansen, 2001; Perron et al., 2006). The advantage of doing this is that, once the breaks have been identified, only post-break data can be used to train the model and make predictions. However, Pesaran and Timmermann (2007) argue that only using post-break data may not be optimal in situations when breaks are small and difficult to identify. The authors show that there can be large gains in terms of out-of-sample predictive performance by using pre-break data as well. In this context, the choice of the length of the estimation window becomes relevant, as there can be a trade-off in terms of predictive performance involved. On the one hand, a model with shorter estimation window can adapt more quickly to large and abrupt structural breaks in the data-generating process by only using the most recent observations for training. On the other hand, longer windows include a larger amount of observations in the training set, which reduces the variance of the estimators and can improve forecasting accuracy, specially in the absence of structural breaks, and even when these are sufficiently small or hard to identify.

The main contribution of this paper is providing empirical evidence to determine the optimal size of the rolling window used to forecast rent prices in the real estate market in an online learning scenario that mimics a real application. The predictive performance of 5 different rolling window sizes (1, 3, 6, 12 and 24 months) is compared according to three distinct error metrics (RMSE, MAE and MAPE). The choice of this set of window sizes is driven by several factors: a) having a sufficiently fine grid of possible candidates, b) with a considerably wide range of values, c) while being able to allocate a good amount of months to the out-of-sample evaluation of all the models, and d) incurring in reasonable computational costs. A period 6 months emerges as the preferred window size, significantly outperforming all the other sizes tested along the first two error metrics.

The only study found in the literature that tests the predictive accuracy of different rolling window sizes in the context of rent prediction is the one by Füss and Koller (2016), who use spatiotemporal autoregressive (STAR) models for one-day-ahead forecasts of rent offer prices in Zurich between 2002 and 2014. The authors try specifications with three different window sizes (400, 500, and 600 days) as a robustness test for their forecasting experiment, and they do not find significant variations between them in terms of RMSE. They do not provide details on how the performance of different window sizes varies across the periods in the sample, nor the carry out tests for the significance of the difference in performance between the estimations. Given that the present study has the explicit goal of determining the optimal rolling window size, that it tests a finer grid of window sizes, that it provides detailed results on how the models' performance change with time, and that statistical tests are carried out to support its conclusions, it provides novel results that can be considered as a benchmark in subsequent forecasting exercises that utilise rolling windows to predict rent prices.

Another important result that emerges from the current research is that, even though some models have better forecasting accuracy than others on average across the entire sample period, there is no single rolling window size that consistently outperforms the rest. Consequently, this study explores the extent to which three additional forecasting strategies that dynamically combine the forecasts of all the models estimated using different window sizes can reduce the forecast error in comparison to using the predictions of individual models. The first one is a cross-validation method which outputs the prediction of the model with the rolling window size that had the lowest forecasting error in the period immediately before the target period. The second one is an ensemble method that outputs a simple average of the predictions of the models with different rolling window sizes. The third one is an ensemble method that outputs a weighted average of the predictions of these models, using time-varying weights determined by the inverse of the error of each model in the previous period. These strategies are based on the work by Pesaran and Timmermann (2007), so this paper can be considered as an empirical application of their proposed methodologies.

The results obtained show that the weighted ensemble model is the best methodology among all the models tested, yielding significantly lower errors than all the individual models with different rolling window sizes and the model with an expanding window. These results are in line with the literature that highlights the diversification gains produced by forecast combination methods (Timmermann, 2006). In the presence of structural breaks, it may be unreasonable to expect that one single model outperforms all others in all the periods in the sample. In this context, finding the optimal model may be difficult or even unfeasible. Therefore, taking an average across various models can be thought of as an "insurance against selecting a poor model" (Sun et al., 2021). The results obtained here are also in line with the empirical literature on house price estimation that uses forecast combinations. For example, using data of individual property transactions from US counties, Glennon et al. (2018) find that combining forecasts based on house price indices with five different methods reduce the bias and the variance of the forecast error and significantly outperform single-model forecast in the vast majority of time periods. Similarly, Gupta et al. (2011) find that a combination of predictions from 10 time-series models yields better results in predicting the US house price index than the individual models.

A further contribution of this paper is to provide empirical evidence about the performance of using ensembles of XGBoost models in a real-time forecasting scenario regarding rent prices. The weighted ensemble of XGBoost models trained with different window sizes, which emerged as the superior strategy, produced an average MAPE value of 19.7% across the entire target sample. Studies for other cities have produced similar results with related methodologies. Only those studies that report MAPE values are comparable, given that this error metric is expressed in the same unit even across real estate markets with different currencies. Glennon et al. (2018) obtain an average MAPE of 22.59% predicting sale prices in US counties using a strategy that combines forecasts of different models with a weighting scheme based on the inverse of the mean square error (MSE) calculated once over a fixed holdout sample. Baur et al. (2023) obtain a lower MAPE of 9.17% when estimating rental offers in Berlin expressed in euros per square meter with gradient boosting and feature extraction from property descriptions. They also obtain a MAPE of 20.57% for house purchase offers in Los Angeles in dollars per square feet.

In a closer relation to the data used in this paper, some studies produce results for real estate markets in other Argentinian cities using machine learning models. Carranza et al. (2019) find a MAPE of 20% when estimating land values with Random Forest and kriging in City of San Francisco in the province of Córdoba, and Cerino et al. (2021) a MAPE of 18.57% using Quantile Regression Forest in the same province. Perhaps the paper that is most closely related to the current study is the one by Rapaport and Normand (2023), who obtain a MAPE of 13.9% using Random Forest to predict of asking prices of properties for sale in the City of Buenos Aires using data obtained from the same source used in the current study.

Apart from the fact that the current study focuses on rents rather than sales or land values, a significant departure from the previous studies conducted in Argentina is that, while this study pays particular attention to the temporal dimension of the observations at the time of defining the forecasting experiment, the other studies indiscriminately mix observations across time in the training and test sets. When the goal is to to maximize predictive performance in a forecasting exercise that mimics a real-world application, doing this amounts to making the methodological error known as data leakage. This involves making use of data that is only available to the analyst because she frames the estimation problem retrospectively, but that would otherwise not be available in all periods in a real application[1]. The results obtained by a strategy that suffers from data leakage in the training stage cannot be considered as valid evidence in favour of its forecasting ability because the prediction problem is ill-defined, as it would not be applicable to any real forecasting application. This is something that the current study is deliberately careful to avoid and therefore the performance metrics obtained are not strictly comparable to the other studies for Argentina mentioned. In this sense, the results obtained can be considered as novel evidence about the efficacy of using XGBoost and forecast combination methods to predict rent prices in Argentina.

The structure of the paper is the following: Section 2 describes the data set and the data cleaning procedure, Section 3 explains how the gradient boosted decision tree model as implemented by the XGBoost algorithm works, Section 4 provides details on the design of the forecasting exercise, Section 5 presents the results obtained, and Section 6 summarises the main conclusions and mentions possible extensions.

[1]One common example of data leakage is to split the observations into the training and testing samples in a random fashion, without taking into account the period in which each observation was obtained. This leads to the erroneous practice of training the model with observations that were observed after some of the observations in the test set. There are additional examples of data leakage, such as those that commit it in the processes of feature standardization, outliers deletion, or missing value imputation.

2 Data

The data set used for this analysis contains all active listings of residential properties for rent in the City of Buenos Aires, published in the biggest e-commerce platform in Latin America, called *Mercado Libre*, with a monthly frequency, spanning from January 2018 to December 2022. The choice of the sample period was determined by the availability of the data shared by the company. All periods available at the time of producing the results were used.

For each property listing, the data set contains information on the following property characteristics: whether it is a house or a flat, whether it is furnished or unfurnished, its total and covered area in square meters, the number of rooms, bathrooms and bedrooms, whether it has a pool, security, a heating system, air conditioning, parking, a fitness space, and/or a common space, the coordinates for its location, its asking price, and whether it is originally invoiced in US dollars or Argentine pesos. All prices in this analysis are expressed in real pesos using December 2022 as the base year and the mean monthly exchange rate from Argentine pesos to US dollars. Using the location for each property, dummy variables for each of the 15 communes of the city[2] and continuous variables for the distance to the nearest train or subway station and to the nearest green space are created.

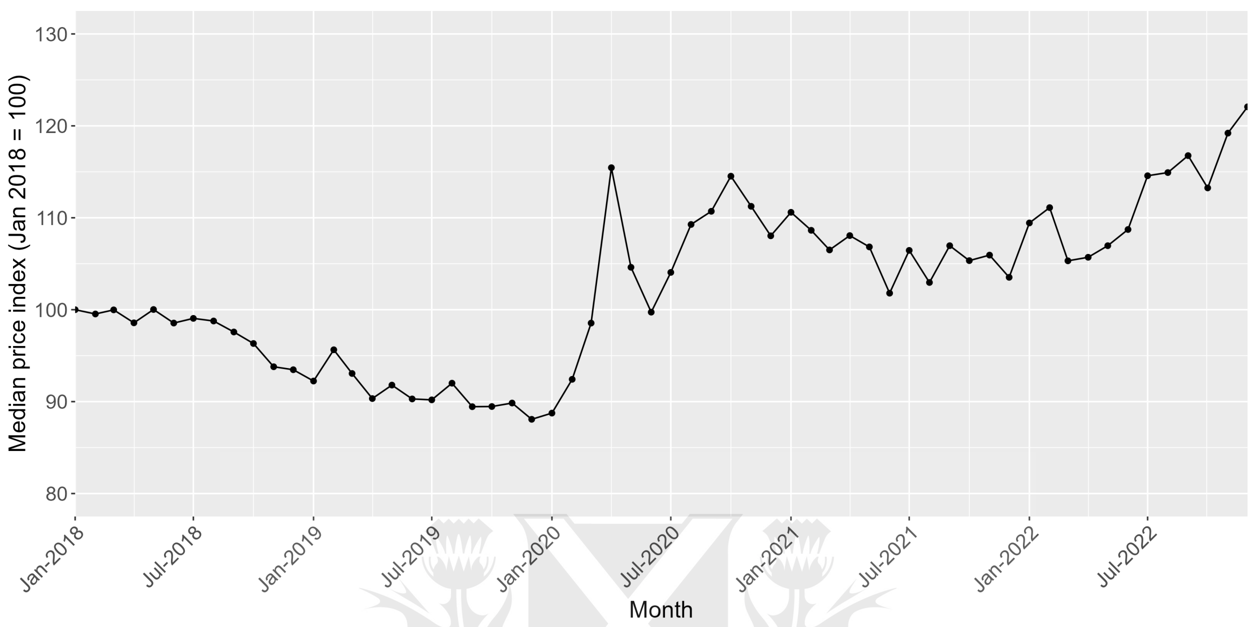
Each property listing is observed once during each month in which it was active on the web-page. Thus, listings that are published for a longer period appear as having more observations in the data set. It is reasonable to expect that the properties that were not taken down from the web-page for a long period of time were probably not sold during that period. This can be considered as a sign that their asking price is not in line with the market transaction price for similar properties. Given that this study uses asking prices as a proxy for the true variable of interest, which is the unobserved price of the transacted rent, this possible source of bias is removed by only using listing appearances with a maximum of 12 months since they were first published. On top of this, to avoid giving more weight to listings that appear for longer periods in the data set, only one monthly appearance for each listing is selected at random.

After going through a data cleaning procedure described in Appendix A, the resulting data set contains 168,423 monthly appearances of unique property listings active during the period of analysis. This large volume of data on property prices and characteristics that spans across 5 years provides sufficiently rich information to reserve a holdout sample for model calibration, and to thoroughly evaluate the performance of models' forecasts across several months, including models that use very small estimation windows (starting from 1 month). Table B.1 in Appendix B shows the description of the final features in the data set and Table C.1 in Appendix C the summary statistics for the relevant variables. The median rent price for the entire data is 94,907.12 pesos (330.75 US dollars) and the mean covered area is 69.86 square meters.

Figure 1 shows the evolution of an index of the monthly median price in pesos per covered square meters for the full sample, setting the first month to 100. The index shows substantial variation across the entire period, which motivates the presumption that it may not always be optimal to use the full sample for training a model to make one-period-ahead predictions.

[2]The communes are the official administrative divisions of the city.

Figure 1: Evolution of Median Price Index



Source: Own-elaboration using the cleaned data set from *Mercado Libre*

3 Model

The statistical model used for this prediction exercise is Extreme Gradient Boosting, implemented using the open-source package for the Python programming language called XGboost[3](Chen & Guestrin, 2016). XGBoost is a machine learning algorithm based on gradient boosted decision trees (GBDT) (Friedman, 2001, 2002). GBDT is a boosting algorithm for classification and regression trees (or CARTs). Appendix D contains a detailed explanation of how these models work.

4 Forecasting experiment

This study proposes a forecasting experiment that mimics the conditions of an out-of-sample forecasting application in real-time. One-month-ahead rent prices are predicted on the basis of the characteristics and the location of properties only using past data. Multiple XGBoost models are trained using different estimation windows, so that each model is trained on a distinct set of observations belonging to varying amounts of periods in the past. In this way, for all the observations in each period, several predictions are done.

The first goal is to test which is the length of the estimation window that minimizes the mean prediction error across all periods. Then, two different methods to define the estimation window are tested: an expanding window and a rolling window. Finally, three methods to combine the predictions of models with different rolling window sizes are used: a cross-validation approach which

[3]The documentation of the package is available at https://xgboost.readthedocs.io/en/stable/.

selects the prediction from the model with the lowest error in the period before the target period, an ensemble that combines all models with rolling windows using a simple mean, and an ensemble that uses a weighted mean based on past performance. These methods are explained below.

4.1 Expanding window

An expanding estimation window has a fixed starting point (usually the first available period in the data) and an end point that increases as the target period moves into the future. Formally, consider a time-series that starts from a period $t = 0$ and ends in period $t = T$. An expanding window uses all the data in periods $t = \{0, ..., T\}$ to train the model to make predictions in $T + 1$. This is equivalent to using the full historical data set available up to the target period to train the model. Therefore, the length of the expanding window is variable and increases with time.

In the absence of structural breaks in the predicted time-series, Pesaran and Timmermann (2007) show that using all the available information for training is optimal because it reduces the variance of the estimators. However, if breaks are present in the data, using the full series in the training stage will lead to biased forecast errors. Furthermore, the fact that the size of the data used for training increases with time means that this method becomes increasingly computationally expensive as time goes by. This is particularly relevant when the forecasting exercise is designed to mimic a real application in which computing resources are limited.

Considering that the estimation of this model is straightforward in terms of selecting the training sample since it always involves using all the available data up to each period without having to defined the size of a fixed window, it is considered as the benchmark specification in this study.

4.2 Rolling windows of different sizes

If it is possible to precisely estimate the timing and the size of the structural breaks that the time series is subject to, using only the historical data belonging to the post-break period may be the superior strategy. Yet, in many applications the timing and size of breaks are difficult to estimate. This study considers the timing and size of structural breaks as unknown. In this context, Pesaran and Timmermann (2007) show that it can be optimal to include pre-break data in the training stage. Thus, the question of how much of the pre-break data should be included in order to maximize forecasting accuracy becomes relevant. To answer this question, the second method used here to define the size of the estimation window consists of using rolling windows of different sizes.

A rolling window is characterised by having a fixed size and variable starting and ending points. If s is the window size, then, for the target period $T + 1$, the rolling window uses data in periods $t = \{T - s, ..., T\}$ for training. As time passes, the fixed-size window is moved one period ahead, therefore the oldest period is dropped from the training set, and the most recent period is included. The values for the window size s that are tested in this study are $S = \{1, 3, 6, 12, 24\}$, where the unit is 1 month. The forecasting exercise tests if one of these window sizes performs significantly better than the others across all periods.

Despite the fact that a rolling window may potentially discard useful data in the absence of breaks, it has the advantage of being computationally less expensive as the size of the training data is roughly fixed across time.

4.3 Cross-validation method

It may be the case that the size of the rolling window that yields the best performance is not the same one across all periods. If this is true, then a strategy that chooses a single window size across all periods (i.e. the one with the best average performance across all periods) may be sub-optimal.

A superior strategy may involve dynamically changing the window size by selecting the one that performs best in each period. However, knowing the performance of the models with different window sizes beforehand is unfeasible. Thus, a feasible strategy of this kind, which Pesaran and Timmermann (2007) denote "cross-validation approach", involves using performance in the most recent periods of the data as the basis to select the model that is used to make the prediction in the target period.

Formally, the cross-validation method consists of using the prediction error of each model that uses a rolling window on the observations in the last τ periods in the data as a measure to compare them with each other, and subsequently use the model with the lowest error across observations in the τ periods to make predictions in $T+1$. In this exercise, τ is set to 1 so that only the observations in the period immediately before the target period are used.

4.4 Ensemble methods

In addition to the cross-validation approach, a strategy that dynamically combines the prediction of the pre-trained models with different window sizes is considered. In this exercise, two ensemble methods based on models with different rolling window sizes are used: an ensemble that takes an average of the prediction from all models and a weighted ensemble based on the inverse of the forecast error in past periods. Formally, the prediction of the target variable by the ensemble methods for each observation i is the weighted average of the predictions of the models with different window sizes S:

$$\hat{y}_i^{ens} = \sum_{s \in S} \frac{w_s}{\sum_{s \in S} w_s} \hat{y}_i^s \tag{1}$$

where $\hat{y}_i^{ens}$ is the prediction for observation i done by the ensemble, $\hat{y}_i^s$ is the prediction of the model indexed by its rolling window size s, and w_s is the weight attributed to it. In the ensemble method with equal weights, w_s takes the same value for all window sizes. If this value is normalized to 1, $\sum_{s \in S} w_s$ is simply the amount of window sizes used $n(S)$. This method is less costly to compute that the weighted ensemble because it does not require calculating the predictive performance of the individual models in past periods.

In the weighted ensemble method, each weight is defined by the inverse of the prediction error calculated on the observations in last τ periods before $T+1$:

$$w_s(s|T+1,\tau) = \frac{1}{Err(s|T+1,\tau)} \tag{2}$$

where $Err(s|T+1,\tau)$ is the value of the error metric of the model that uses a rolling window size s to make a prediction for the observations in the last τ observations before the target period $T+1$. As in the cross-validation approach explained above, τ is set to 1, so that the models' predictions are weighted by the their performance on the last period before the target period. Also, it is easy to see that the cross-validation approach is equivalent to setting the weight in equation (1) of the model with the lowest error metric in each period equal to 1 and the rest of the weights equal to 0 .

4.5 Error metrics

Three error metrics are used to evaluate the performance of all the methods, as well as to create the weights for the ensemble methods: root mean square error (RMSE), mean absolute error (MAE) and mean absolute percentage error (MAPE). These are defined as follows[4]:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{3}$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{4}$$

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{5}$$

The main difference between RMSE and MAE is that, given that the RMSE squares the difference between the prediction and the target variable before taking its square root, it gives more weight to larger errors. The MAE metric, by weighting the errors equally regardless of their size, is less influenced by outlier errors. In addition, MAE is more easy to interpret because it is expressed in the same unit as the target variable. The MAPE metric, on the other hand, is expressed in percentage terms. Its scale-independence makes it directly comparable to prediction results from other studies. However, it has the disadvantage that it penalizes positive errors more than negative errors (for a further discussion, see Hyndman and Koehler (2006)). Using multiple error metric helps

[4]For simplicity of notation, the indices that define the specific time period for which the error metrics are calculated are omitted. The periods may correspond to the target period $T+1$ for model evaluation, or to the periods that go from $T-\tau+1$ to T used in the cross-validation and weighted ensemble approaches.

to provide a more complete picture of the models' performance and makes the results more robust if common patterns across are found.

4.6 Hyper-parameter tuning

Before estimation, it is necessary to set values for the hyper-parameters of the XGboost algorithm. This is not straightforward, as the choice of values can yield different results across different forecasting exercises. Hyper-parameter tuning refers to the strategy used to select the optimal values for the model's hyper-parameters.

One possible approach to this problem involves updating the hyper-parameters each time the model is fitted to a new sample. In this exercise, given that a large number of models is fitted according to the different methods tested and the time periods in the sample, this strategy is too computationally expensive. A simpler strategy is chosen: a single XGBoost model is calibrated once using only observations from the first year of the data and the selected values for the hyper-parameters are used in all the other models. The entire sample goes from January 2018 to December 2022, so the data used in the calibration stage goes from January 2018 to December 2018. The observations that were used for tuning are not included in the out-of-sample test set.

To select the best set of hyper-parameters, the Grid Search cross-validation method (Hastie et al., 2009) with 5 folds is implemented. The hyper-parameters of the XGBoost algorithm that are calibrated in this study are:

- Learning rate: a value between 0 and 1 that is multiplied with the prediction of each tree in each step of the algorithm.
- Number of estimators: the number of trees used in the XGboost ensemble.
- Minimum child weight: the minimum number of observations in each new leaf required to make a partition.
- Maximum tree depth: the maximum level of partitions in a tree.
- Minimum split loss: the minimum reduction in the loss function required to make a new partition.
- Column sample by tree: a value between 0 and 1 that denotes the proportion of randomly sampled features that are used to fit each tree.
- L1 regularization: value of the L1 regularization penalty used on the predicted values in each leaf.

To enable the definition a sufficiently fine grid for each parameter while keeping a reasonably short computation time, the parameters are divided into two sets, and the calibration is carried out in two stages. More details about the distribution of parameters into the two sets, the values chosen for the grids, and the final values selected can be found in Appendix E.

5 Results

The forecasting performance of the different methods explained above was evaluated on the basis of one-month-ahead predictions for all listing in the processed data set. Notwithstanding the fact that the available data set goes from January 2018 to December 2022, the evaluation sample selected goes from February 2020 to December 2022. This choice of time period is made to allow comparison between all methods in the same amount of periods. The method that uses the largest rolling window of 24 months can only be estimated starting from January 2020. Moreover, the cross-validation method and the two ensemble methods require one additional month so that the weights that they assign to the pre-trained models can be calculated for the initial period. The model with the expanding window used the largest window of all models, starting with a size of 25 months in the first target period and reaching a size of 59 months in the last.

The independent variables used to predict properties' rent prices in real Argentine pesos of December 2022 were: *covered_ area*, *uncovered_ area*, *bedrooms*, *bathrooms*, *pool*, *security*, *furnished*, *heating*, *air_ conditioning*, *parking*, *common_ space*, *fitness_ space*, *distance_ to_ transport*, *distance_ to_ greenspace* and 14 dummy variables for the city's communes.

The results of the evaluation of the model with an expanding window and the models with rolling windows for all periods are shown in Figures 2, 3, and 4, respectively. The models were evaluated using the RMSE, MAE, and MAPE metrics. The model that used a rolling window of 6 months showed the best performance in terms of RMSE and MAE, with mean values of 138,909.62 and 51,543.04, respectively. These results suggest that using all the available data at each period is not always the superior strategy. They may be indicative of the fact that the relationship between house prices and their characteristics is subject to concept drift.

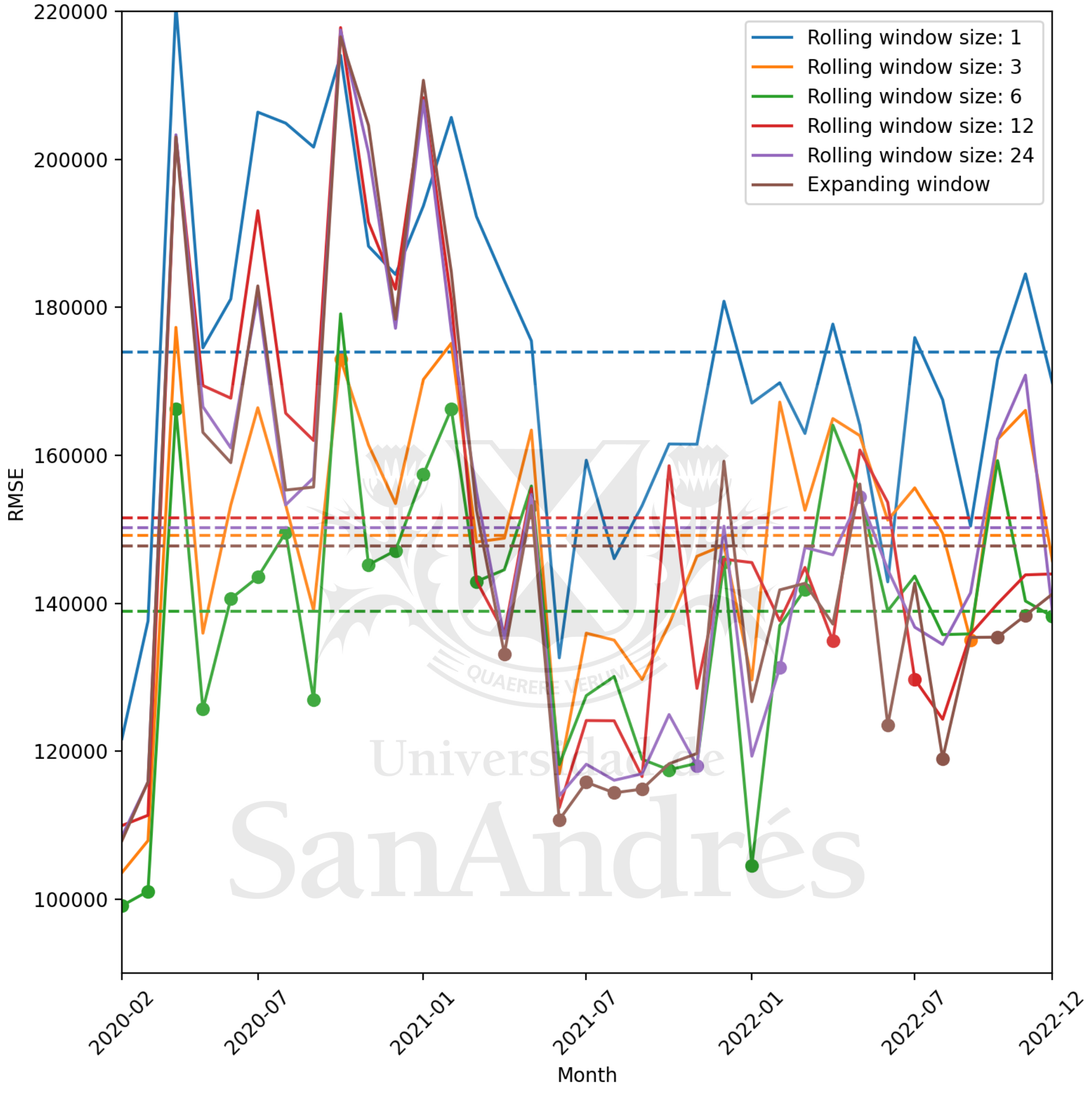
However, not all the error metrics led unanimously to this conclusion. The model with the lowest MAPE was the one that uses an expanding window, with a value of 20.1%. It was followed by the model with a window size of 24 months, with a value of 20.6%. Starting from March 2021, the model with the expanding window outperformed all the rolling window models in all periods in terms of MAPE. Interestingly, the average performance across all periods in terms of MAPE increased with the size of the estimation window.

Table 1: Difference in forecasting performance in comparison to model with rolling window of size 6

Model	RMSE	MAE	MAPE
Rolling window size 1	1.252***	1.203***	1.159***
Rolling window size 3	1.074***	1.048***	1.036***
Rolling window size 12	1.091**	1.076**	0.981
Rolling window size 24	1.082*	1.067**	0.974
Expanding window	1.063	1.044	0.949

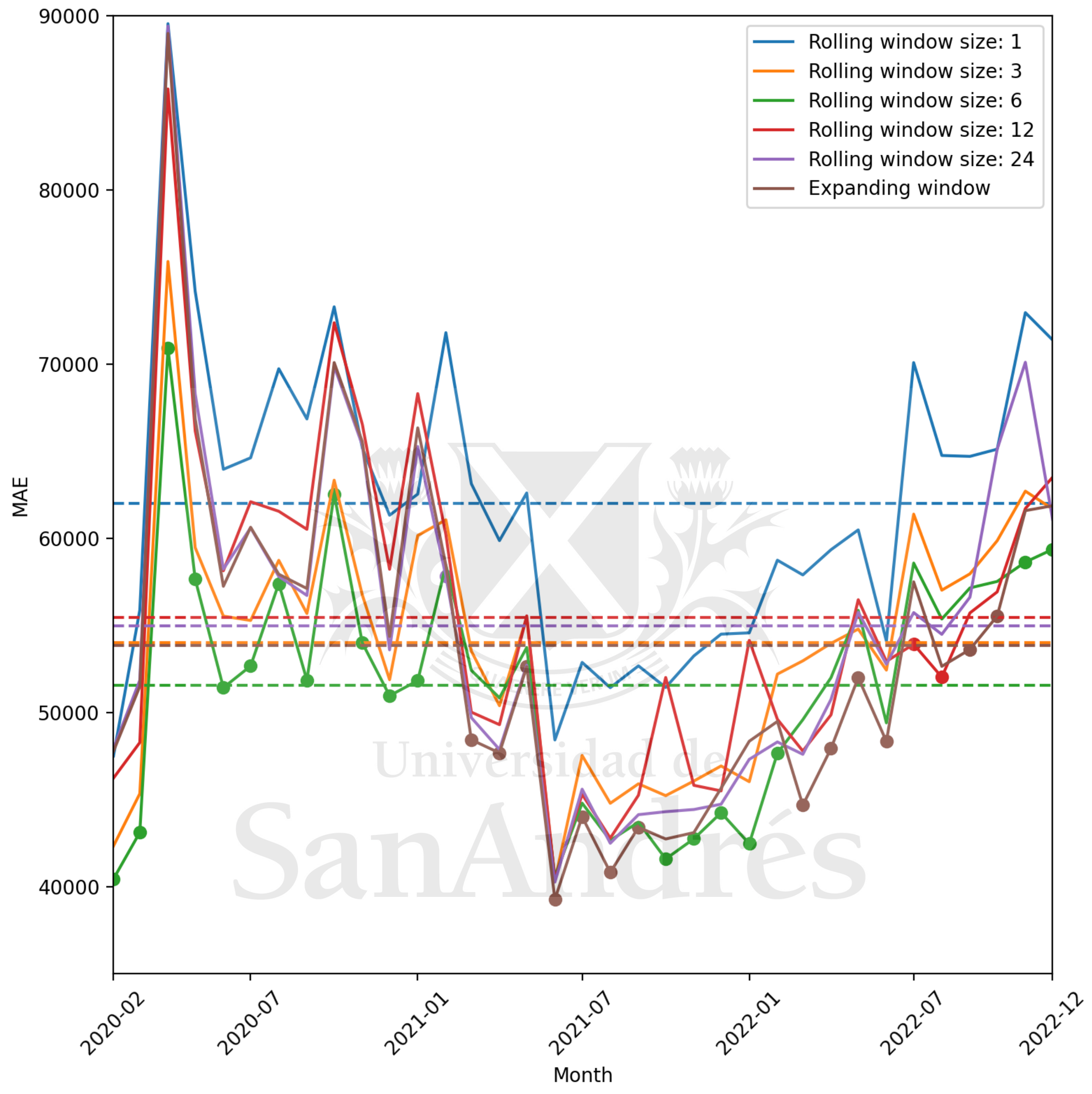
Notes: The values in the cells correspond to the ratio of the mean of the error metric across all periods for each model and the same measure for the model with a rolling window of size 6. The statistical significance of the difference in forecasting performance was calculated using the Diebold-Mariano test with Newey-West standard errors. *10%, **5%, ***1%.

Figure 2: Evolution of RMSE for models with rolling windows and model with expanding window



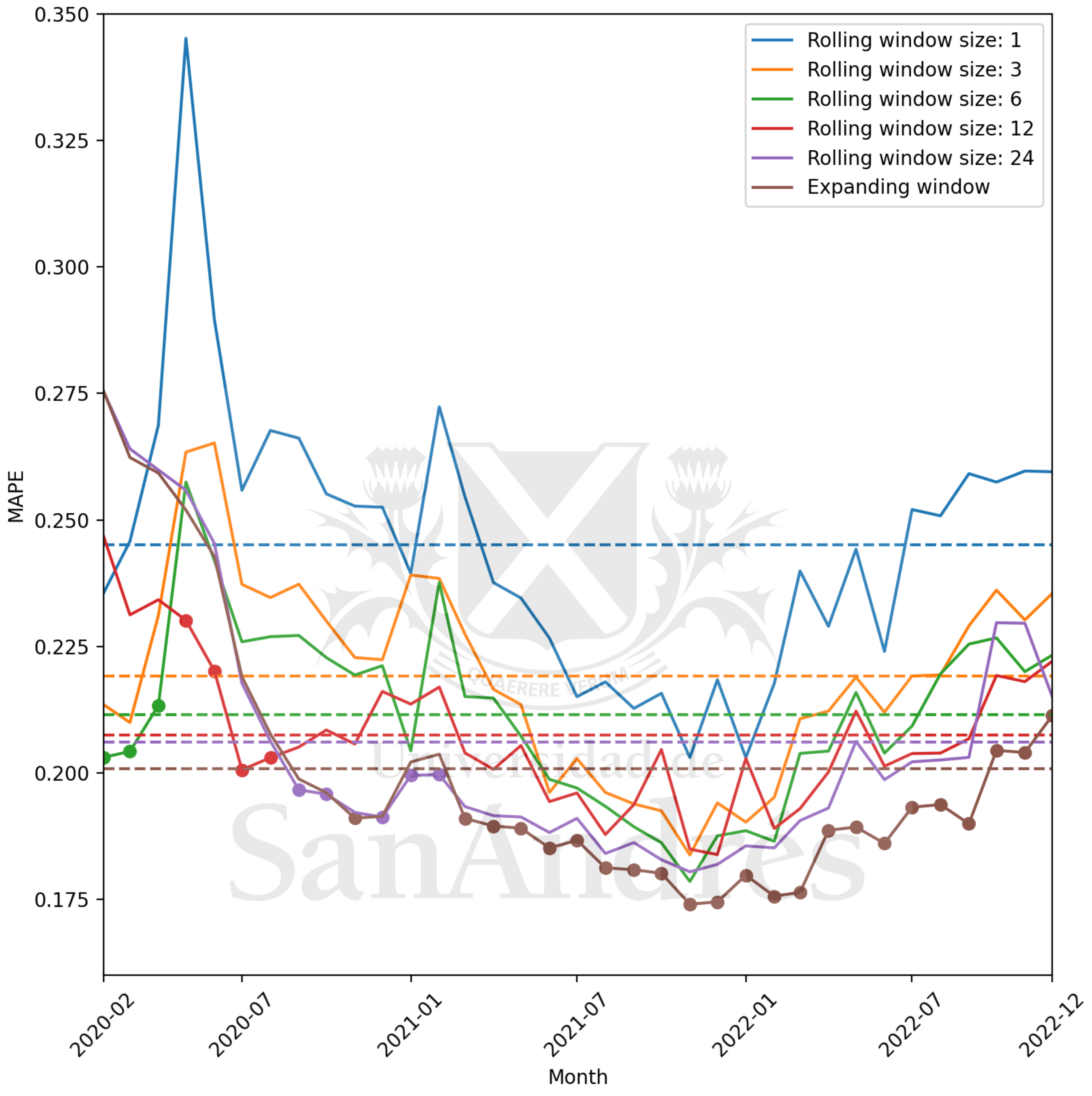
Notes: The dashed horizontal lines correspond to the mean value for each series across the entire sample period. The mean values for the models with rolling window size 1, 3, 6, 12, and 24 and the expanding window are, respectively, 173,904.75, 149,191.22, 138,908.62, 151,507.69, 150,259.45, and 147,708.97. Each dot represents the error value of the model that yielded the lowest error in each month.

Figure 3: Evolution of MAE for models with rolling windows and model with expanding window



Notes: The dashed horizontal lines correspond to the mean value for each series across the entire sample period. The mean values for the models with rolling window size 1, 3, 6, 12, and 24 and the expanding window are, respectively, 62,018.05, 54,014.45, 51,543.04, 55,453.06, 54,986.02, and 53,828.64. Each dot represents the error value of the model that yielded the lowest error in each month.

Figure 4: Evolution of MAPE for models with rolling windows and model with expanding window



Notes: The dashed horizontal lines correspond to the mean value for each series across the entire sample period. The mean values for the models with rolling window size 1, 3, 6, 12, and 24 and the expanding window are, respectively, 0.245, 0.219, 0.211, 0.207, 0.206, and 0.201. Each dot represents the error value of the model that yielded the lowest error in each month.

Table 1 shows the values of the ratio of the error metrics of the model with a rolling window of size 6 with respect to the other models mentioned so far. A value that is less than 1 in the table means that the corresponding model had a lower mean error metric across all periods in the evaluation set than the model with window size 6. Additionally, the table shows the results of the Diebold-Mariano test with Newey-West standard errors to establish the significance of the difference in forecasting performance. It can be seen that the model with window size 6 significantly outperformed the rest of the models with rolling windows according to RMSE and MAE, but the difference in these error metrics with the expanding window model was not significant. On the other hand, even though the model with window size 6 was outperformed in terms of MAPE by the models with window sizes of 12 and 24, as well as the one with an expanding window, the difference between their forecasting performance was not significant.

Table 2 provides a comprehensive summary of the figures by showing the amount of periods in which each model had the best performance among all models for each error metric. The first three columns compare the performance only for the models with rolling windows. The model with rolling window of size 6 was the best model in 57% of the 35 months according to the RMSE, 63% according to the MAE, and 17% according to the MAPE, while the same results for the model with rolling window of size 24 were 20%, 20% and 71%. In the subsequent three columns of the table, the model with an expanding window is also considered. The model with rolling window size 6 was outperformed by the model with an expanding window according to the RMSE in 2 of the months in which it was the best model in comparison to the rest of the models with rolling windows, in 3 months according to the MAE, and in 1 month according to the MAPE. The model with an expanding window was the best model in 29% of all months according to the RMSE, 37% according to the MAE, and 66% according to the MAPE. The model with the shortest window size of 1 month was not the best model in any period.

Table 2: Amount of months in which each model yielded the lowest error metric: rolling windows and expanding window

	Only rolling windows			Rolling windows and expanding window		
Model	RMSE	MAE	MAPE	RMSE	MAE	MAPE
Rolling window size 1	0	0	0	0	0	0
Rolling window size 3	2	1	0	2	0	0
Rolling window size 6	20	22	4	18	19	3
Rolling window size 12	6	5	6	2	2	4
Rolling window size 24	7	7	25	3	1	5
Expanding window	-	-	-	10	13	23

In light of these results, the conclusion about which size of estimation window is the preferred one for this forecasting application depends on which error metric is considered. What stands out from them is that no single model consistently outperformed the rest in all periods in the sample. In fact, there were models that showed the worst performance in some periods (such as window size 12 on January 2021 in terms of MAE) and the best performance in other periods (the same model and error metric on July and August 2022). This motivated the idea of using a strategy that aims to select the best-performing model in each period or a combination of predictions from more than one model. The next step involved testing whether a feasible strategy that dynamically chooses models with different rolling window sizes or takes an average of their predictions could yield better results.

Figures 5, 6 and 7 show the performance of the expanding window, cross-validation, unweighted ensemble, and weighted ensemble methods in terms of RMSE, MAE and MAPE across all periods in the sample. Overall, the expanding window method yielded the worse performance among this new set of models in terms of the mean of the three error metrics across all periods, though the difference in terms of MAPE with the second worst model, the cross-validation approach, was not large (20.1% vs 20.0%). The best-performing model across all error metrics was the weighted ensemble, with an RMSE of 134,739.82, a MAE of 49,949.48, and a MAPE of 19.7%. The ensemble with equal weights had a very similar performance, resulting as the second-best method across the mean of all three metrics. The dots in the figures represent the model that yielded the lowest error in each month among all the models considered so far, including the rolling window models. In the majority of periods (60% for RMSE, 66% for MAE, and 71% for MAPE), the best model was either the one with an expanding window or any of the forecast combination methods, and not a model with a fixed rolling window.

The first three columns of Table 3 show how the three methods that combine the forecasts of multiple models fared against each other and the expanding window model. The weighted ensemble method was the preferred model in most periods according to RMSE and MAE, and had the same number of winning months than the expanding window model according to the MAPE. The cross-validation method correctly selected the prediction of the best-performing model with a rolling window in 6 out of 35 months in terms of RMSE, 2 in terms of MAE, and 14 in terms of MAPE. For this to have happened, any of the rolling window models would have had to produce the lowest error metric in at least two consecutive months.

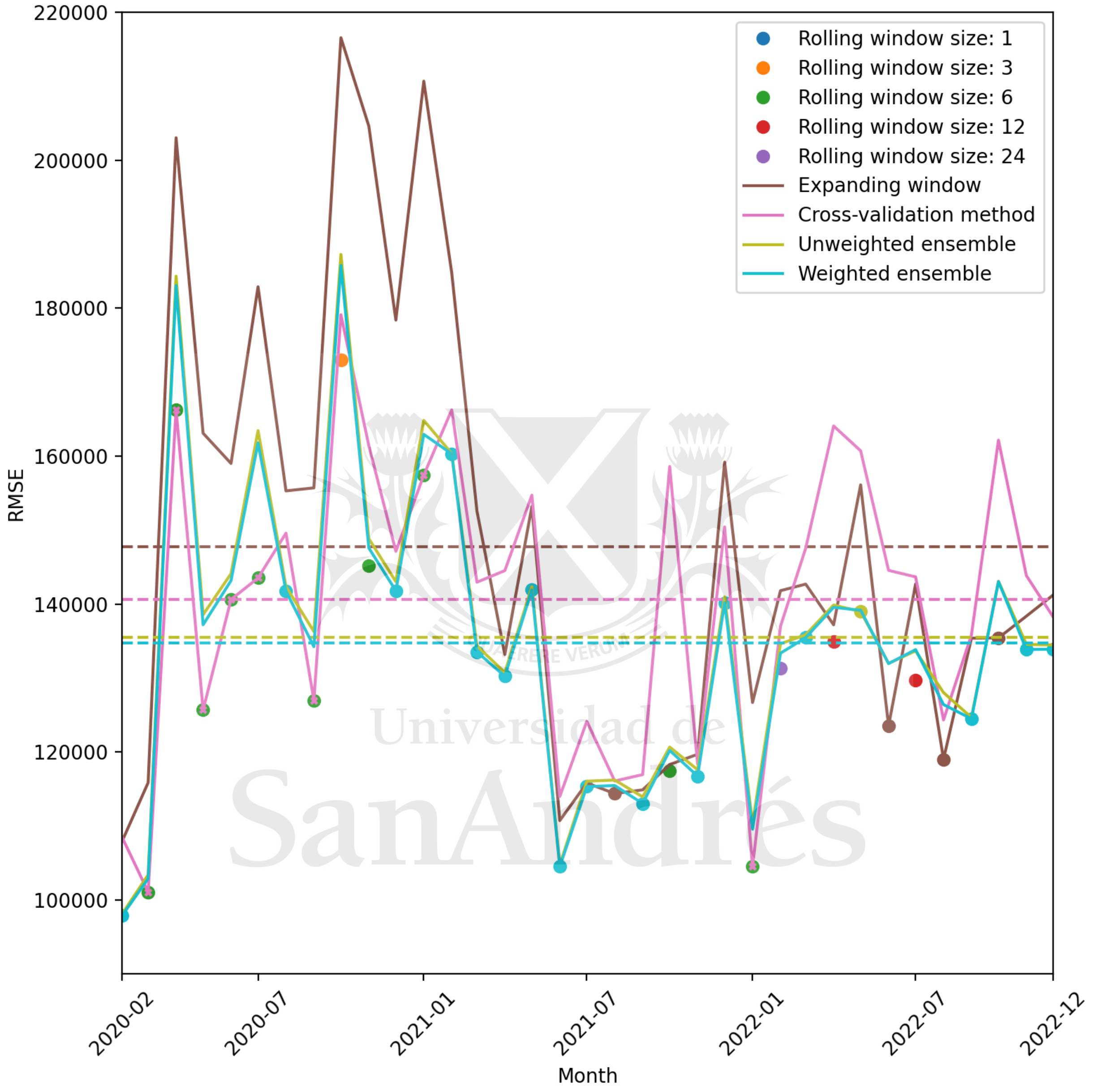
The last three columns of Table 3 show the number of months in which each model produced the best results, considering all models tested so far[5]. Of all the models, the weighted ensemble had the best performance in the greatest number of periods in terms of RMSE and MAE, and yielded similar results to the model with an expanding window in terms of MAPE (12 vs. 13 periods). The second best model with this criterion was the one with a rolling window of size 6 in terms of RMSE and MAE, but not MAPE.

Table 3: Amount of months in which each model yielded the lowest error metric: rolling windows, forecast combinations and expanding windows

	Forecast combinations and expanding window			All models except cross-validation		
Model	RMSE	MAE	MAPE	RMSE	MAE	MAPE
Rolling window size 1	-	-	-	0	0	0
Rolling window size 3	-	-	-	1	0	0
Rolling window size 6	-	-	-	10	11	3
Rolling window size 12	-	-	-	2	1	4
Rolling window size 24	-	-	-	1	0	3
Expanding window	6	2	14	4	2	13
Cross-validation	9	10	7	-	-	-
Unweighted ensemble	2	3	0	1	3	0
Weighted ensemble	18	20	14	16	18	12

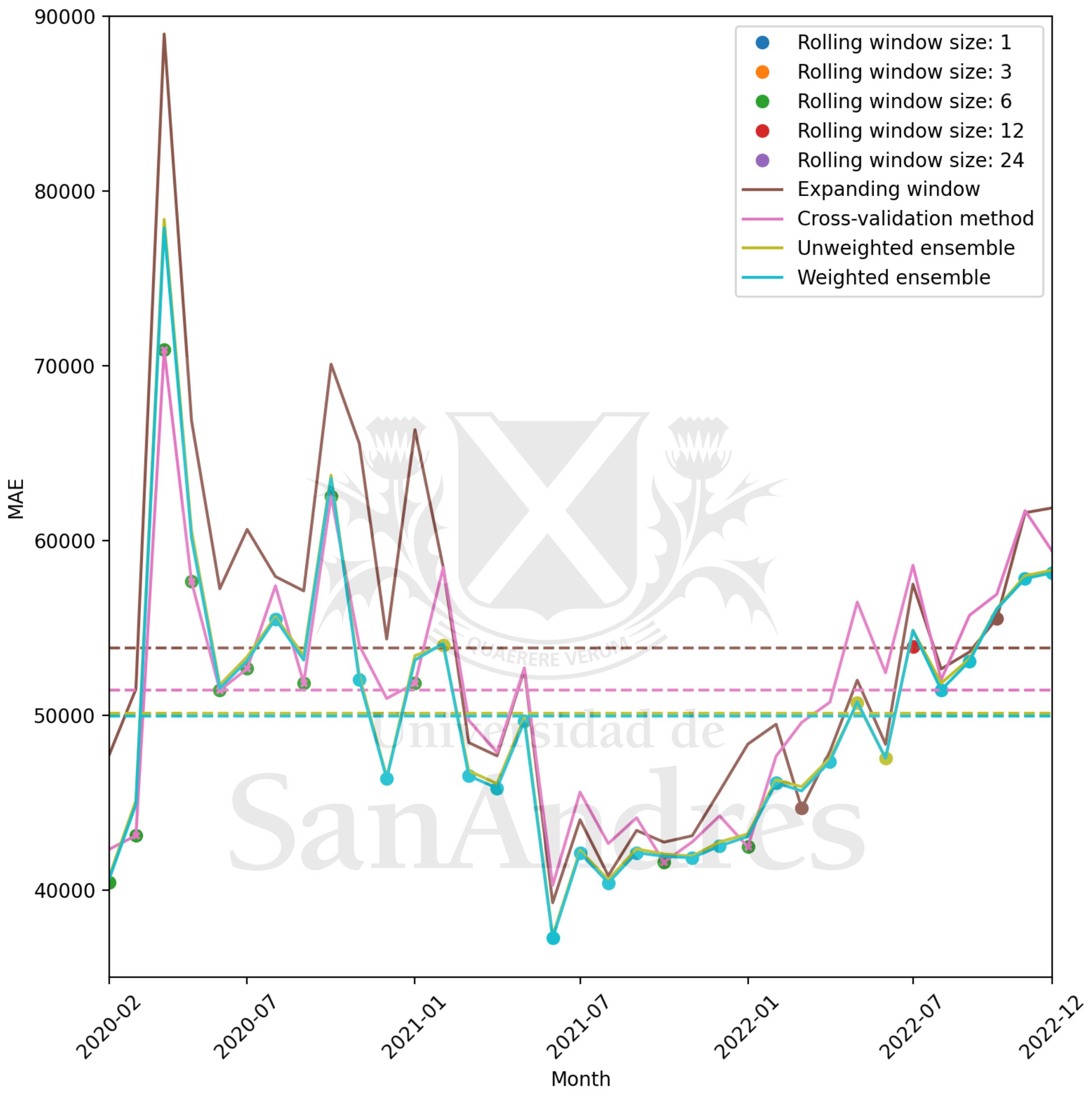
[5]The cross-validation method is excluded from the analysis in the last three columns because it only yields the best performance when any of the rolling window models do so. The latter are included in the summary, so adding the cross-validation method as well would be redundant.

Figure 5: Evolution of RMSE for models that use a cross-validation approach, ensemble methods, and an expanding window



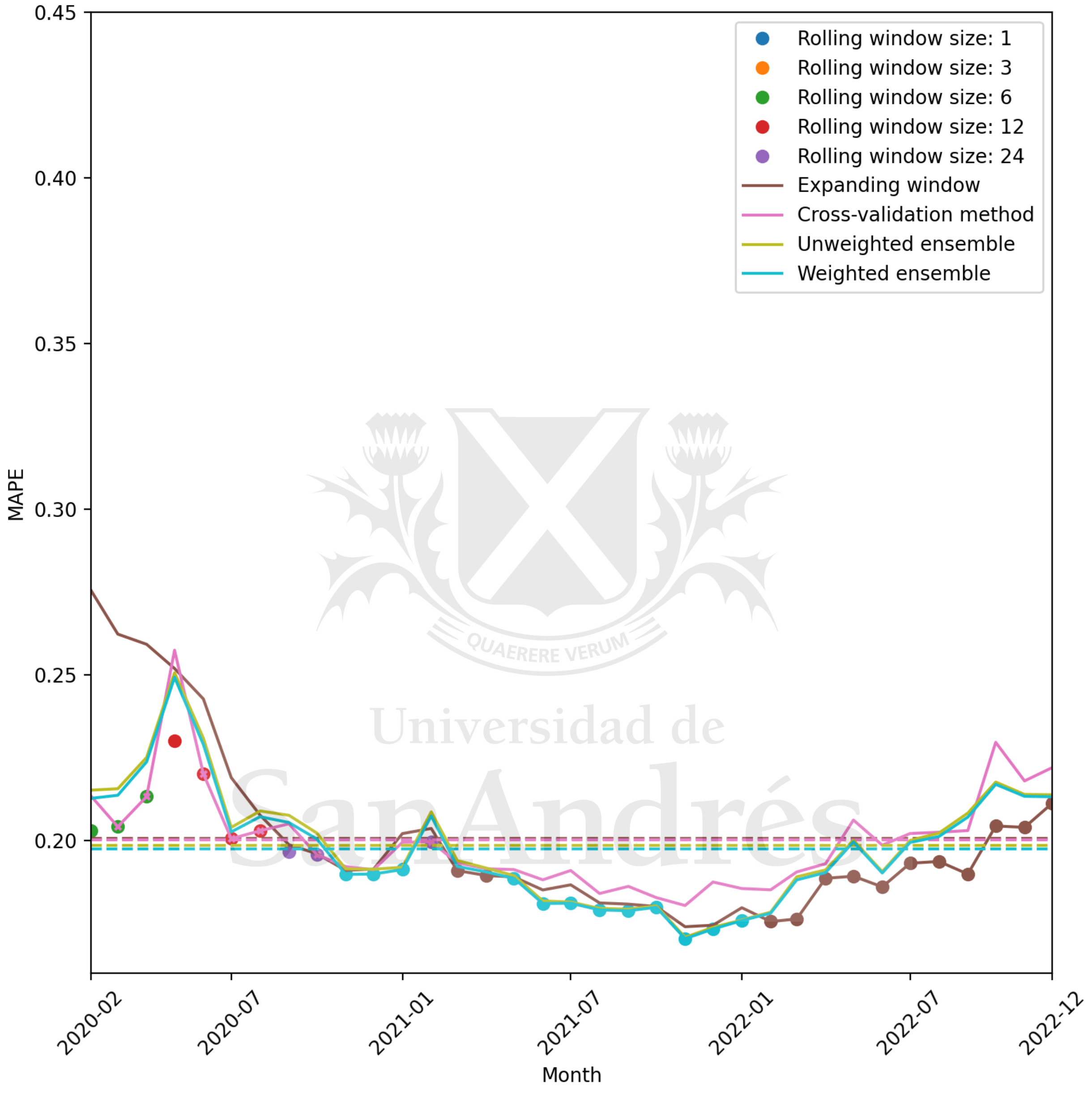
Notes: The dashed lines correspond to the mean value for each series across the entire sample period. The mean values for the model with an expanding window, the model that uses the cross-validation approach, the unweighted ensemble, and the weighted ensemble are, respectively, 147,708.97, 140,587.28, 135,498.11, and 134,739.83. The dots represent the lowest value of the error metric for all models in each month and the colour corresponds to the model that produced that value. In the periods in which the cross-validation method selected the rolling window model that yielded the lowest error, a small cross is added on top of the dot. The lines for the models with rolling windows were removed for ease of visualization.

Figure 6: Evolution of MAE for models that use a cross-validation approach, ensemble methods, and an expanding window



Notes: The dashed lines correspond to the mean value for each series across the entire sample period. The mean values for the model with an expanding window, the model that uses the cross-validation approach, the unweighted ensemble, and the weighted ensemble are, respectively, 53,828.64, 51,434.92, 50,122.31, and 49,949.48. The dots represent the lowest value of the error metric for all models in each month and the colour corresponds to the model that produced that value. In the periods in which the cross-validation method selected the rolling window model that yielded the lowest error, a small cross is added on top of the dot. The lines for the models with rolling windows were removed for ease of visualization.

Figure 7: Evolution of MAPE for models that use a cross-validation approach, ensemble methods, and an expanding window



Notes: The dashed lines correspond to the mean value for each series across the entire sample period. The mean values for the model with an expanding window, the model that uses the cross-validation approach, the unweighted ensemble, and the weighted ensemble are, respectively, 0.201, 0.200, 0.199, and 0.197. The dots represent the lowest value of the error metric for all models in each month and the colour corresponds to the model that produced that value. In the periods in which the cross-validation method selected the rolling window model that yielded the lowest error, a small cross is added on top of the dot. The lines for the models with rolling windows were removed for ease of visualization.

Finally, Table 4 provides a summary of how all the different models tested performed in comparison to the model that used an expanding window, which was considered as the benchmark strategy as it did not require defining the size of the window beforehand nor any computation of previous performance. The tables also includes the results of the tests for statistical significance in the difference in forecasting performance. The cross-validation method outperformed the expanding window model across all error metrics, but without statistical significance. The two ensemble methods, on the other hand, yielded significantly lower error metrics than the expanding window model, in terms of RMSE and MAE, and also in terms of MAPE but without statistical significance.

Table 4: Difference in forecasting performance between all models and the model that uses an expanding window

Model	RMSE	MAE	MAPE
Rolling window size 1	1.177***	1.152***	1.221***
Rolling window size 3	1.010	1.003	1.092***
Rolling window size 6	0.940	0.958	1.053
Rolling window size 12	1.026**	1.030***	1.033
Rolling window size 24	1.017	1.022**	1.026**
Cross-validation	0.952	0.956	0.998
Ensemble with equal weights	0.917***	0.931***	0.989
Weighted ensemble	0.912***	0.928***	0.984

Notes: The values in the cells correspond to the ratio of the mean of the error metric across all periods for each model and the same measure for the expanding window model. The statistical significance of the difference in forecasting performance was calculated using the Diebold-Mariano test with Newey-West standard errors. *10%, **5%, ***1%.

6 Conclusion

This study relied on an out-of-sample one-step-ahead forecasting experiment to find the optimal window size for predicting real estate rental prices in the City of Buenos Aires using the XGBoost algorithm. It meticulously examined models based on different window sizes, a cross-validation approach, and ensemble methods, leveraging a rich set of variables obtained from property listings published in an online platform.

Its findings provide evidence that a rolling window of a size of 6 months significantly outperforms windows of sizes 1, 3, 12 and 24 months in terms of predictive accuracy, as measured by both the root mean square error (RMSE) and the mean absolute error (MAE). This rolling window of size 6 also outperforms the traditional expanding window approach according to those two error metrics, albeit without statistical significance. Notably, an approach that combines predictions from models with distinct window sizes weighted according to their recent performance emerged as the superior strategy overall, yielding an average MAPE of 19.7% across the evaluated sample. It significantly outperformed all models that use a rolling window, as well as the model with an expanding window in terms of RMSE and MAE, and in terms of MAPE but without statistical significance.

These findings emphasize the benefit of leveraging recent historical data in a manner that adapts to the evolving dynamics of the real estate market to handle the issue of concept drift. They also challenge the conventional approach of utilizing all available data regardless of how far in the past it was obtained, suggesting that a more nuanced strategy of selective data inclusion can yield superior predictive accuracy. Furthermore, the findings add evidence to an established empirical literature that favours the use of forecast combination strategies over the prediction of individual models.

Overall, the methodology used in this study, which emphasizes the prevention of data leakage at all stages and designs a setup to compare the performance of different models, serves as a robust framework for developing models of house price prediction that carefully consider the temporal dimension of the problem. The methodological approach outlined ensures that the predictive strategy is applicable to real-world forecasting applications, providing valuable insights for investors, policymakers, and other stakeholders in the real estate market. The results obtained can serve as a benchmark for the efficacy of the machine learning models in predicting house prices in the City of Buenos Aires, as well as to inform interested actors about what the length of the relevant time-frame of pricing decisions in this real estate market is. Finally, even though this paper focuses solely on rent prices, considering that the forecasting horizon is only one-month ahead, its methodological approach can offer practical insights about how to incorporate the influence of time in an exercise of property valuation, which is a common policy challenge encountered by tax agencies.

Future research could explore the use of additional statistical models and apply them to different geographical areas and with a different data source to investigate if the results obtained here can be replicated in other contexts. Another valuable extension would be the inclusion additional predictive factors to improve predictive performance. Variables that may be particularly interesting for this purpose are macroeconomic indicators that are known to affect the real estate market with considerable frequency. Asides from improving performance, including them may help to develop an economic explanation of the determinants of the length of the time-horizon of agents' pricing behaviour, as well as to identify specific macroeconomic shocks that may significantly alter the dynamics of the market. Also, perhaps more accuracy can be obtained by defining dummy variables for neighbourhoods with a data-driven methodology (as it was done in Füss and Koller (2016)), instead of using their administrative definition. Lastly, if the findings of this paper were to be used to produce valuations for properties' rents for a practical application, it would be recommendable to incorporate methods of uncertainty quantification such as conformal prediction (Angelopoulos & Bates, 2021; Gammerman et al., 2013).

References

Abidoye, R. B., Chan, A. P., Abidoye, F. A., & Oshodi, O. S. (2019). Predicting property price index using artificial intelligence techniques: Evidence from Hong Kong. *International journal of housing markets and analysis*, *12*(6), 1072–1092.

Amini, A., Nafari, K., & Singh, R. (2022). Effect of air pollution on house prices: Evidence from sanctions on Iran. *Regional Science and Urban Economics*, *93*, 103720.

Angelopoulos, A. N., & Bates, S. (2021). A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv preprint arXiv:2107.07511*.

Anselin, L., & Le Gallo, J. (2006). Interpolation of air quality measures in hedonic house price models: Spatial aspects. *Spatial Economic Analysis*, *1*(1), 31–52.

Antipov, E. A., & Pokryshevskaya, E. B. (2012). Mass appraisal of residential apartments: An application of random forest for valuation and a CART-based approach for model diagnostics. *Expert Systems with Applications*, *39*(2), 1772–1778.

Baur, K., Rosenfelder, M., & Lutz, B. (2023). Automated real estate valuation with machine learning models using property descriptions. *Expert Systems with Applications*, *213*, 119147.

Bisello, A., Antoniucci, V., & Marella, G. (2020). Measuring the price premium of energy efficiency: A two-step analysis in the Italian housing market. *Energy and Buildings*, *208*, 109670.

Breiman, L. (2001). Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical science*, *16*(3), 199–231.

Carranza, J. P., Piumetto, M. A., Salomon, M. J., Monzani, F., Montenegro, M. G., & Córdoba, M. (2019). Valuación masiva de la tierra urbana mediante inteligencia artificial: El caso de la ciudad de San Francisco, Córdoba, Argentina.

Cerino, R. M., Carranza, J. P., Piumetto, M. A., Bullano, M. E., Donalisio, V. C., & Monzani, F. (2021). Propuesta para la valuacin masiva del suelo urbano. aplicacin espacial del algoritmo Quantile Regression Forest. *Revista Vivienda y Ciudad*, *8*, 261–274.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794.

Chow, G. C. (1960). Tests of equality between sets of coefficients in two linear regressions. *Econometrica: Journal of the Econometric Society*, 591–605.

Fleischer, A. (2012). A room with a view—a valuation of the mediterranean sea view. *Tourism Management*, *33*(3), 598–602.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of statistics*, 1189–1232.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational statistics & data analysis*, *38*(4), 367–378.

Fuerst, F., McAllister, P., Nanda, A., & Wyatt, P. (2015). Does energy efficiency matter to home-buyers? An investigation of EPC ratings and transaction prices in England. *Energy Economics*, *48*, 145–156.

Füss, R., & Koller, J. A. (2016). The role of spatial and temporal structure for residential rent predictions. *International Journal of Forecasting*, *32*(4), 1352–1368.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, *46*(4), 1–37.

Gammerman, A., Vovk, V., & Vapnik, V. (2013). Learning by transduction. *arXiv preprint arXiv:1301.7375*.

Glennon, D., Kiefer, H., & Mayock, T. (2018). Measurement error in residential property valuation: An application of forecast combination. *Journal of Housing Economics*, *41*, 1–29.

Gupta, R., Kabundi, A., & Miller, S. M. (2011). Forecasting the US real house price index: Structural and non-structural models with and without fundamentals. *Economic Modelling*, *28*(4), 2013–2021.

Hansen, B. E. (2001). The new econometrics of structural change: Dating breaks in US labor productivity. *Journal of Economic perspectives*, *15*(4), 117–128.

Hastie, T., Tibshirani, R., & Friedman, J. H. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (Vol. 2). Springer.

Ho, W. K., Tang, B.-S., & Wong, S. W. (2021). Predicting property prices with machine learning algorithms. *Journal of Property Research*, *38*(1), 48–70.

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International journal of forecasting*, *22*(4), 679–688.

Lahmiri, S., Bekiros, S., & Avdoulas, C. (2023). A comparative assessment of machine learning methods for predicting housing prices using Bayesian optimization. *Decision Analytics Journal*, *6*, 100166.

Levantesi, S., & Piscopo, G. (2020). The importance of economic variables on London real estate market: A random forest approach. *Risks*, *8*(4), 112.

Lorenz, F., Willwersch, J., Cajias, M., & Fuerst, F. (2022). Interpretable machine learning for real estate market analysis. *Real Estate Economics*.

Malik, K., Kim, S., & Cultice, B. J. (2023). The impact of remote work on green space values in regional housing markets.

Margaretic, P., & Sosa, J. B. (2023). How local is the crime effect on house prices? *The Journal of Real Estate Finance and Economics*, 1–40.

Pérez-Rave, J. I., Correa-Morales, J. C., & González-Echavarría, F. (2019). A machine learning approach to big data regression analysis of real estate prices for inferential and predictive purposes. *Journal of Property Research*, *36*(1), 59–96.

Perron, P., et al. (2006). Dealing with structural breaks. *Palgrave handbook of econometrics*, *1*(2), 278–352.

Pesaran, M. H., & Timmermann, A. (2007). Selection of estimation window in the presence of breaks. *Journal of Econometrics*, *137*(1), 134–161.

Rampini, L., & Re Cecconi, F. (2022). Artificial intelligence algorithms to predict Italian real estate market prices. *Journal of Property Investment & Finance*, *40*(6), 588–611.

Rapaport, M., & Normand, A. (2023). Predicción de precios de inmuebles en CABA mediante técnicas de aprendizaje automático. *Undergraduate thesis, Universidad Nacional de Luján.*

Rico-Juan, J. R., & Taltavull de La Paz, P. (2021). Machine learning with explainability or spatial hedonics tools? An analysis of the asking prices in the housing market in Alicante, Spain. *Expert Systems with Applications*, *171*, 114590.

Stevenson, S. (2007). A comparison of the forecasting ability of ARIMA models. *Journal of Property Investment & Finance*, *25*(3), 223–240.

Sun, Y., Hong, Y., Lee, T.-H., Wang, S., & Zhang, X. (2021). Time-varying model averaging. *Journal of Econometrics*, *222*(2), 974–992.

Timmermann, A. (2006). Forecast combinations. *Handbook of economic forecasting*, *1*, 135–196.

Trojanek, R., Gluszak, M., & Tanas, J. (2018). The effect of urban green spaces on house prices in Warsaw. *International Journal of Strategic Property Management*, *22*(5), 358–371.

Xu, X., & Zhang, Y. (2021). House price forecasting with neural networks. *Intelligent Systems with Applications*, *12*, 200052.

Zheng, X., Peng, W., & Hu, M. (2020). Airport noise and house prices: A quasi-experimental design study. *Land Use Policy*, *90*, 104287.

Appendix A Data cleaning process

Before any estimation, the property data is subject to a thorough cleaning procedure. Even though the online platform imposes some restrictions on the information contained in each property listing, users have some degree of freedom to complete it, leaving space for inconsistencies. Therefore, the cleaning procedure involves dropping observations with missing data in certain features, observations with incorrect feature values, and observations with outlying feature values, as well as doing missing value imputation and feature engineering.

The following criteria were used to remove observations with missing or incorrect information from the data set:

- Observations without location coordinates.
- With incorrect location attributes such as being located outside the boundaries of the city or with a value for the commune that does not coincide with the actual location according to the coordinates (except when the value corresponds to one of the adjacent communes to the actual one).
- With a negative listing price or with a price equal to 11,111,111, 1,111,111, 111,111, 11,111, or 1,111, for both currencies.
- With a negative value for total or covered area, or with a covered area greater than total area.
- With less rooms than bedrooms.
- Marked as invoiced in US dollars, but with a listing price that differs from the dollar-converted listing price which is given in the original data set.
- Marked as invoiced in Argentine pesos, but with a listing price that is equal to the dollar-converted listing price which is given in the original data set.
- Houses with duplicated values in all of these variables: location, property type, total area, and covered area, in the same month, with different listing ID (only one listing ID, chosen at random, is kept).
- Flats with duplicated values in all of these variables: location, property type, total area, covered area, bedrooms, bathrooms, and rooms, in the same month, with different listing ID (only one listing ID, chosen at random, is kept).
- Monthly listings appearances with more than one year after the month they were first published.

Additionally, some observations with extreme values were removed either because the values are highly implausible or because they were considered outliers. Listings that complied with any of the following criteria were removed:

- Flats with less than 25 square meters of covered area.
- Houses with less than 35 square meters of covered area
- Flats with more than 1,500 square meters of covered area.

- Houses with more than 3,000 square meters of covered area.
- A ratio of uncovered area to covered area of more than 3.
- Zero or more than 10 bedrooms.
- Zero or more than 10 bathrooms.
- Zero or more than 15 rooms.
- Property age greater than 100.

All prices in US dollars were converted to Argentine pesos using the mean monthly rate of the unofficial exchange rate (or "blue" dollar), which was the appropriate reference exchange rate for property transactions during the period. Next, all prices were converted to real pesos with December 2022 (the last month of the sample) as the base year using the national Consumer Price Index. Additional outliers were removed according to absolute values of the resulting rent price:

- Listing price in pesos lower than 8,000 (roughly USD 25).
- Listing price in pesos greater than 8,000,000 (roughly USD 25,000).
- Listing price in pesos per covered square meter lower than 350 (roughly USD 1).
- Listing price in pesos per covered square meter greater than 20,000 (roughly USD 60).

In addition to this, observations with a price per square meter smaller (greater) than the 1th (99th) percentile were removed. In order to avoid data leakage at the time of defining the cut-off values for outliers, percentiles for each month were calculated only using observations from that month and the previous 11 months. Furthermore, the percentiles were calculated by groups according to the property type (house or flat) and the neighbourhood. For groups that have less than 10 observations, broader groups were defined according to property type and commune, and if the problem persists, to commune only.

Following the cleaning procedure, most of the listings still appeared more than once in the data as they were published for more than 1 month. To ensure that the results are not skewed by the duration of the listings' presence in the website, the analysis is limited to a single entry of each listing, which is chosen at random.

Zero-imputation was done for the missing values in the following the binary variables related to property amenities: multi-purpose room, party room, playground, business center, parking, pool, security, gym, jogging track, grill area, tennis court, cinema, air conditioning, furnished, and heating.

The variable "common space" was defined as having at least one of the following amenities: multi-purpose room, party room, playground, business center, grill area, and cinema. The variable "fitness space" was defined as having a gym, tennis court and/or jogging track.

Data for train and subway stations and green spaces were downloaded from the Open Data portal of the Government of the City of Buenos Aires[6]. Green spaces were defined as all public

[6]https://data.buenosaires.gob.ar/dataset/

squares, gardens or parks larger than 10,000 square meters. Parks with semi-public access were not considered.

Appendix B Variable description

Table B.1: Variable description

Variable	Description
id	Identicication number for each unique listing
listing_month	Date of publication
listing_age	Number of days since start of publication
month	Month of the year of publication (1 to 12)
invoiced_in_usd	Binary equal to 1 if listing was published invoiced in US dollars
price_realpesos	Rent asking price in real pesos of December 2022
price_usd	Rent asking price in current US dollars
total_area	Total property area in square meters
covered_area	Covered property area in square meters
uncovered_area	Uncovered property area in square meters
price_realpesos_per_covered_sqm	Price in real pesos per covered square meter
price_usd_per_covered_sqm	Price in US dollars per covered square meter
house	Binary equal to 1 if property is a house, 0 if it is a flat
rooms	Number of rooms
bedrooms	Number of bedrooms
bathrooms	Number of bathrooms
furnished	Binary equal to 1 if property is offered as furnished
property_age	Property age in years
heating	Binary equal to 1 if property has a heating system
air_conditioning	Binary equal to 1 if property has air conditioning
parking	Binary equal to 1 if property or building has parking
security	Binary equal to 1 if property or building has private security
pool	Binary equal to 1 if property or building has swimming pool
common_space	Binary equal to 1 if property or building has at least one of the following ammenities: multi-purpose room, party room, playground, business center, grill area, cinema
fitness_space	Binary equal to 1 if property has at least one of the following ammenities: gym, tennis court, jogging track
latitude	Latitude in decimal degrees (EPSG 4326)
longitude	Longitude in decimal degrees (EPSG 4326)
commune	Official administrative area in which the property is located
neighbourhood	Unofficial neighbourhood in which the property is located
distance_to_transport	Distance to nearest subway or train station in meters
distance_to_greenspace	Distance to nearest green space in meters

Appendix C Descriptive statistics

Table C.1: Summary statistics

variable	min	Q0.25	median	mean	Q0.75	max	sd
listing_age	0.00	17.00	29.00	38.28	47.00	364.00	37.52
total_area	25.00	42.00	55.00	78.03	83.00	3,753.00	73.64
covered_area	25.00	40.00	50.00	69.86	75.00	1,480.00	59.42
uncovered_area	0.00	0.00	3.00	8.17	8.00	2,600.00	25.82
price_realpesos	19,933.03	70,645.54	94,907.12	189,500.71	150,020.24	7,655,372.07	316,440.62
price_usd	48.60	223.60	330.75	635.49	553.16	21,000.00	999.62
price_realpesos_per_covered_sqm	382.13	1,576.87	1,899.89	2,298.14	2,401.95	17,593.72	1,412.88
price_usd_per_covered_sqm	0.91	4.78	6.38	7.84	9.16	60.00	4.81
rooms	1.00	2.00	2.00	2.61	3.00	15.00	1.12
bedrooms	1.00	1.00	1.00	1.67	2.00	10.00	0.90
bathrooms	1.00	1.00	1.00	1.39	2.00	10.00	0.75
property_age	0.00	1.00	17.06	21.83	40.00	100.00	20.66
distance_to_transport	1.07	255.67	447.32	541.04	708.42	3,296.82	404.69
distance_to_greenspace	0.00	347.72	693.22	743.47	1,057.41	2,712.54	476.60
invoiced_in_usd	0.00	0.00	0.00	0.11	0.00	1.00	0.31
house	0.00	0.00	0.00	0.02	0.00	1.00	0.15
furnished	0.00	0.00	0.00	0.06	0.00	1.00	0.23
heating	0.00	0.00	0.00	0.25	1.00	1.00	0.43
air_conditioning	0.00	0.00	0.00	0.32	1.00	1.00	0.47
parking	0.00	0.00	0.00	0.24	0.00	1.00	0.43
security	0.00	0.00	0.00	0.08	0.00	1.00	0.27
pool	0.00	0.00	0.00	0.15	0.00	1.00	0.35
common_space	0.00	0.00	0.00	0.13	0.00	1.00	0.33
fitness_space	0.00	0.00	0.00	0.09	0.00	1.00	0.29

Appendix D Regression trees, GDBT and XGBoost

D.1 Regression tree

A regression tree is a model that splits the space of features into non-overlapping regions according to their values. Then, for each region, it predicts the mean value of the target variable for all the training observations in the region. Formally, given a training set $\{x_i, y_i\}_{i=1}^{n}$ of n observations, with a vector x of features and a continuous target variable y, a regression tree can be defined by the following function (Hastie et al., 2009):

$$\hat{f}(x) = \sum_{k=1}^{J} c_j \cdot I(x \in R_j) \tag{6}$$

where $\hat{f}(x)$ is the predicted value for the target variable using the features, c_j is the mean value of the target variable of the training observations within the region, and $I(x \in R_j)$ is an indicator function which equals 1 if x falls into the j^{th} region and 0 otherwise.

The first step in building a regression tree consists of finding a way to partition the data into J regions $R_1, R_2, \dots, R_J$ in order to minimize a loss function, which is typically the sum of squared residuals $\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$. This involves choosing the optimal combination of features and splitting-points used to segment the feature space. Trying all possible combinations of features and splitting-points is too computationally expensive. To avoid this, the regression tree relies on an *exact greedy* algorithm: it begins by finding the feature-splitting-point combination that divides the data into two half-planes such that the loss function is minimized, looking at all possible candidates according to the feature values of the observations in the training data. Then, it proceeds recursively by repeating this step for each of the two half-planes created by the previous binary split, without regarding future splits in each step.

This process of splitting the data could potentially go on until the tree has one terminal node (also called a leaf) for each distinct observation. However, in this scenario, the model would be overfitting the training data and thus its ability to generalize and predict accurately on new data would be hindered. For this reason, strategies to make the tree stop growing have been devised. These include setting a minimum number of observations required for each region, or setting a minimum reduction in the loss function required to make a further partition. Another approach consists of growing the tree until a minimum amount of leafs is reached, and then pruning it by sequentially collapsing the internal nodes which hold the observations that are the least similar to each other.

D.2 Gradient Boosted Decision Trees

Generally, predictions by CARTs present high variance due to the hierarchical nature of the tree-building process (Hastie et al., 2009). Instead, a significant increase in predictive performance can be achieved by using an ensemble of trees. A widely used ensemble method in the context of CARTs is the boosting algorithm, which combines the predictions several weak learners (the CARTs) to produce more accurate predictions. GBDT is an example of a boosting algorithm that combines a sequence of tree models by addition, based on the idea of gradient descent. Algorithm 1 shows the steps involved in GBDT.

Algorithm 1 Gradient Boosted Decision Trees

```
1: $F_0(x) = \arg\min_{\hat{y}} \sum_{i-1}^{N} l(y_i, \hat{y})$
2: for $m = 1$ to $M$ do
3:     Compute $n$ pseudo-residuals $r_{im} = -\left[\frac{\partial l(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$
4:     Fit a regression tree $f_m$ to the set $\{x_i, r_{im}\}_{i=1}^{n}$ to create $J_m$ terminal regions $R_{jm}$
5:     For regions $j = 1....J_m$ compute $\hat{y}_{jm} = \arg\min_{\hat{y}} \sum_{x_i \in R_{jm}} l(y_i, F_{m-1}(x_i) + \hat{y})$
6:     Update $F_m(x) = F_{m-1}(x) + \eta \sum_{j=1}^{J_m} \hat{y}_{jm} I(x \in R_{jm})$
7: end for
```

The GBDT algorithm starts by finding an initial value for $\hat{y}$, denoted by $F_0(x)$, which minimizes a loss function $l(y_i, \hat{y})$ across all observations in the training set. The loss function $\frac{1}{2}(y - \hat{y})^2$ is usually used for this purpose because it is particularly convenient, given that its derivative with respect to $\hat{y}$ is simply the sum of the negative residuals. The result is that the first prediction for y is its mean across all observations of the training set. In this way, the algorithm begins with the prediction of a single tree with only one leaf.

The algorithm proceeds by adding the predictions of a sequence of M regression trees to the previous predictions, down-scaled by a factor η called the learning rate[7]. For each step m in this sequence, the algorithm first computes the derivative of the loss function with respect to the prediction $F(x)$, evaluated at the previous prediction $F_{m-1}(x)$[8]. Considering the squared loss function used, this is equivalent to calculating the pseudo-residuals r_{im} of the previous prediction for all observations. Next, a regression tree $f_m(x_i)$ is fit using the features as inputs, and the pseudo-residuals as target variables. The fitted tree creates J_m terminal regions R_{jm} into which the training set is divided. Step 5 involves computing the optimal predicted value for the observations in each leaf of the tree (which, given the loss function chosen, is equal to the mean of the pseudo-residuals in each region). Finally, the previous prediction $F_{m-1}(x)$ for each observation is updated by adding the predicted value of the latest estimated regression tree, multiplied by the learning rate $0 < \eta \leqslant 1$. In sum, the algorithm takes small steps towards improving its accuracy by correcting the error of its last prediction using multiple regression trees.

D.3 Extreme Gradient Boosting

XGBoost adapts the GBDT algorithm to improve its out-of-sample performance. It does so mainly by adding a regularization term to the objective function of the gradient boosting algorithm (Chen & Guestrin, 2016). Formally, the modified objective function is:

$$L = \sum_{i=1}^{N} l(y_i, \hat{y}_i) + \sum_{m=1}^{M} \Omega(f_m) \tag{7}$$

$$\text{where} \quad \Omega(f_m) = \gamma J_m + \frac{1}{2}\lambda \|\hat{y}_m\|^2 \tag{8}$$

For each of the M regression trees f_m estimated, an additional term $\Omega(f_m)$ is added to the original loss function $l(y_i, \hat{y}_i)$ of GBDT in the objective function. The coefficient γ puts a penalty on the number of leafs J_m in each tree, while λ penalizes the size of the predictions of all leafs in the tree, denoted by the Euclidean the norm of $\hat{y}_m$.

[7]For simplicity, the learning rate η is assumed to be pre-defined and constant across iterations. This is actually how XGBoost operates, but other methodologies may admit a varying value for η.

[8]This derivative is the gradient that gives the name to the algorithm. Taking its negative provides guidance to perform gradient descent. By calculating the value of this negative gradient, the algorithm finds the "steepest" direction in which one could move along the domain of the loss function, starting from the position of the last prediction, to achieve the largest possible reduction in the loss. By doing this in any iterative fashion, the algorithm aims to take the shortest route towards a minimum of the loss function.

As with GBDT, after an initial guess, the tree ensemble is constructed by greedily adding subsequent regression trees, only that the optimization problem that has to be solved for each new tree is now the following:

$$L_m = \sum_{i=1}^{N}[l(y_i, \hat{y}_{m-1,i} + f_m(x_i))] + \Omega(f_m) \tag{9}$$

To calculate the gradient of the objective function for optimization, the algorithm uses a second-order Taylor approximation:

$$L_m \approx \sum_{i=1}^{N}[l(y_i, \hat{y}_{m-1,i}) + g_i f_m(x_i) + \frac{1}{2}h_i f_m^2(x_i)] + \Omega(f_m) \tag{10}$$

where g_i and h_i are the the first and second order gradients of the loss function $l(y_i, \hat{y}_{m-1,i})$ with respect to $\hat{y}_{m-1,i}$. Again, if the loss function $\frac{1}{2}(y-\hat{y})^2$ is used, g_i is simply the negative residual of the last prediction, and h_i equals 1. Using this approximation, by expanding $\Omega(f_m)$ and taking the derivative of L_m with respect to $\hat{y}_{jm}$, the optimal value for the prediction in each leaf j takes the form:

$$\hat{y}_{jm}^* = -\frac{\sum_{i \in R_{jm}} g_i}{\sum_{i \in R_{jm}} h_i + \lambda} \tag{11}$$

If the parameter λ is set to 0, then $\hat{y}_{jm}^*$ would be simply the mean value of the residuals for all observations in each region. When $\lambda > 0$, the magnitude of the prediction for each leaf is reduced, and this effect is proportionately greater for regions with fewer observations.

Next, by plugging the obtained expression for $\hat{y}_{jm}^*$ into equation (10), and ignoring the constant term $l(y_i, \hat{y}_{m-1,i})$, the optimal value for the approximated loss function is obtained:

$$L_m^* = -\frac{1}{2}\sum_{i \in R_{jm}} \frac{(\sum_{j=1}^{J_m} g_i)^2}{\sum_{i \in R_{jm}} h_i + \lambda} + \gamma J_m \tag{12}$$

The value of L_m^* is the basis for deciding whether or not to split a node, and on which split point. The gain or loss reduction obtained by splitting a node is equal to:

$$\Delta L_m = \frac{1}{2}\left[\frac{(\sum_{i \in R_{jm}^{left}} g_i)^2}{\sum_{i \in R_{jm}^{left}} h_i + \lambda} + \frac{(\sum_{i \in R_{jm}^{right}} g_i)^2}{\sum_{i \in R_{jm}^{right}} h_i + \lambda} - \frac{(\sum_{i \in R_{jm}} g_i)^2}{\sum_{i \in R_{jm}} h_i + \lambda}\right] - \gamma \tag{13}$$

where R_{jm}^{left} and R_{jm}^{right} are the left and right regions of the new split. Each of the three terms inside the brackets in equation (13) can be interpreted as a similarity score for each leaf. Assuming the squared loss function $\frac{1}{2}(y - \hat{y})^2$ is used, this score is equivalent to the sum of the residuals, squared, over the number of residuals plus λ.

When γ is set to 0, the gain in terms of loss reduction after any split is equal to the similarity scores for the left and right regions created by the split, minus the similarity score of the root or preceding node[9]. This gain is compared across candidates for feature-split point combinations[10]. A split is then made if the gain of the best candidate is greater than 0.

Given that the two new regions will always have less observations than their parent region, increasing λ decreases their score in greater proportion than the score of the parent node. In this way, λ acts as a pruning parameter, discouraging node splits. Additionally, when γ is different from 0, it also discourages tree-splitting. A split will be made only if the gain is greater than the chosen value for γ. Thus, a greater value for γ will lead to trees with fewer branches.

Once the new regression tree is constructed, the algorithm proceeds in the same manner as GBDT, updating its previous prediction by adding the prediction of the new tree, down-scaled by a learning rate η. It finishes after M trees are built.

XGBoost also allows for the use of additional strategies to prevent overfitting. These include: setting a maximum level of splits or maximum depth for each tree; using a random sub-sample of the observations without replacement to train each tree; using a random sub-sample of the features without replacement for each tree, split level, or node; and setting the requirement of a minimum sum of the Hessians h_i in each leaf (which are equivalent to the amount of observations in each leaf if the squared loss function mentioned above is used) to further partition a node. Lastly, the algorithm boasts several computational optimization techniques used to make its training faster and scalable to large data sets.

Appendix E Hyper-parameter tuning

Table E.1 contains the grids of values tested for each hyper-parameter, together with the individual values that yielded the best performance in terms of the mean squared error of the cross-validation sample.

[9]Given that the similarity score is only used as a relative measure, the $\frac{1}{2}$ in equation (13) can be ignored.

[10]Differently from GBDT, XGBoost allows the use of an *approximate* greedy algorithm for finding optimal split points, instead of the exact greedy algorithm. Instead of trying all feature values in the training data, this algorithm tries their percentiles to reduce computing costs.

Table E.1: Hyper-parameter grids for model calibration

	Hyper-parameter	Grid values	Value chosen
Stage 1	Learning rate	0.01, 0.1, 0.15, 0.3	0.15
	Number of estimators	50, 100, 200, 300	300
	Minimum child weight	1, 3, 5, 7	7
	Maximum tree depth	3, 5, 7, 9	7
Stage 2	Minimum split loss	0, 0.1, 0.2, 0.4	0
	Column sub-sample by tree	0.5, 0.75, 0.9, 1	1
	L1 regularization	0, 1, 50, 100	0

In the first stage, the values of the first group of four parameters were selected. Then, a second iteration of the Grid Search algorithm was run, fixing the values for the parameters found in the first stage, and searching over the grids for the three remaining parameters.