The Journal of Real Estate Finance and Economics (2024) 68:624–653
https://doi.org/10.1007/s11146-022-09893-1

A Machine Learning Approach to Price Indices: Applications in Commercial Real Estate

Felipe D. Calainho[1] · **Alex M. van de Minne**[2] · **Marc K. Francke**[1,3]

Accepted: 22 February 2022 / Published online: 9 April 2022

Abstract

This article presents a model agnostic methodology for producing property price indices. The motivation to develop this methodology is to include non-linear and non-parametric models, such as Machine Learning (ML), in the pool of algorithms to produce price indices. The key innovation is the use of individual out-of-time prediction errors to measure price changes. The data used in this study consist of 29,998 commercial real estate transactions in New York, in the period 2000–2019. The results indicate that the prediction accuracy is higher for the ML models compared to linear models. On the other hand, ML algorithms depend more on the data used for calibration; they produce less stable results when applied to small samples and may exhibit estimation bias. Hence, measures to reduce or eliminate bias need to be implemented, taking into consideration the bias and variance trade-off.

Keywords Commercial real estate · Price indices · Machine learning

JEL Classification R33 · C43 · C51

✉ Felipe D. Calainho
f.dutracalainho@uva.nl

Alex M. van de Minne
avdminne@uconn.edu

Marc K. Francke
m.k.francke@uva.nl

[1] Amsterdam Business School, Universiteit van Amsterdam, Plantage Muidergracht 12 1018 TV, Amsterdam, The Netherlands

[2] Center for Real Estate and Urban Economic Studies, University of Connecticut, 2100 Hillside Road, Unit1041 Storrs, Connecticut, CT 06269, USA

[3] Ortec Finance, Naritaweg 51, 1043, BP, Amsterdam, The Netherlands

Introduction

Having reliable real estate price indices is pivotal for several reasons. Firstly, index volatility is an important input in determining the cost of capital of real estate (Geltner et al., 2017). This is helpful for investors, underwriters and policymakers to determine interest rates. Secondly, it allows one to (re-)appraise real estate portfolios mark-to-market, by multiplying (historic) book values by the corresponding index returns (Francke & Van de Minne, 2021a). This is of obvious interest for investors, but also for owner-occupiers in determining their household wealth. This can also help appraise commercial and residential property values for property tax purposes. Finally, real estate indexing would allow for derivative trading (Deng & Quigley, 2008), especially in the absence of index revisions. However, producing real estate indices is a nontrivial task as real estate is a heterogeneous asset class that transacts infrequently (Francke & Van de Minne, 2021b).

In the real estate literature, several methods haven been proposed to compile transaction price indices. Hedonic pricing models and repeat sales are two of the most popular ones. The hedonic pricing model is the "original" method of price index construction (Malpezzi, 2002), and will be the focus of this paper. The hedonic pricing model has its origins in appraising farmland values (Haas, 1922a; 1922b; Wallace, 1926), and has been used for making constant quality indices in the automobile industry (Court, 1939). Later examples of literature using hedonic pricing models include Griliches (1961), Lancaster (1966), and Rosen (1974). The hedonic pricing model assumes that the price of a commodity is composed by aggregating the individual contributions of each of its characteristics.[1]

Classic hedonic approaches employ linear models, estimated by Ordinary Least Squares (OLS), or its generalizations for index construction. These linear algorithms have the convenience of the β parameter, that elicits a linear relationship, in units, between the dependent and independent variables. Nevertheless, the dependence on estimating a parameter vector β constrains the use of a wide family of other regression algorithms, such as Machine Learning (ML) approaches, that are non-linear or non-parametric. This paper proposes a (hedonic) framework which admits the usage of parametric as well as non-parametric/non-linear models to construct price indices.

The proposed method can be viewed as a model that predicts the price of a property ($\hat{Y}$) sold in the current period (t) as if it had been sold in the previous period ($t - 1$) The difference between the estimated price ($\hat{Y}_{t-1}$) and actual price (Y_t) at a property level can be decomposed into a market price change and some noise. More specifically, we first estimate some (linear/non-linear/non-parametric) model using only transactions in (or up to and including) period $t-1$. In a second step, we predict the values of all properties sold in period t, using the estimated model in the first step. In the final step, we take the out-of-sample prediction errors for transactions in period t,

[1] The repeat sales approach, initially proposed by Bailey et al. (1963), is a method that utilizes the relative price difference of a real estate property that was sold twice or more in a given time interval as the basis for index construction.

and compute the average change in order to compute the price index change. To find the returns in subsequent periods, we simply redo these three steps for the remaining time periods ($t, t+1, \ldots, T-2, T-1$). This methodology is called the chained "Paasche" price index (Geltner et al., 2017). Note that this three step framework is agnostic as to which model and estimation technique is used in the first step, and it can include, but is not limited to, ML techniques.

One of the many reasons to have an price index methodology that incorporates other (regression) algorithms such as ML is that these permit a more flexible relationship between variables when compared to linear models (Varian, 2014). This flexibility increases our expectations of better approximating the unknown complex data generating process that underlies real estate property prices. In other words, it might improve our ability to find the "true" price change between periods that is less affected by noise.

Following this line of thought, this article proposes a new methodology that generalizes the chained hedonic approach in a way that any ML regression algorithm can be used to compile property price indices. Hence, this paper contributes to the existing literature for building non-linear price indices, such as McMillen and Dombrow (2001), and it connects the fields of number theory, econometrics, and ML. In fact, to the best of our knowledge, this paper is the first to estimate price *indices* using ML approaches.[2] Our proposed framework therefore opens up the field of chained hedonic pricing models for numerous new possibilities.

The data used in this study have been provided by Real Capital Analytics (RCA) and consist of 29,998 individual transactions from commercial real estate properties in New York metropolitan area, in the period from 2000 to 2019. We construct an aggregate price index and indices per property type.

In general, the results show that using individual predictions of real estate transaction prices is a viable solution for building price indices. Another finding is that Out-of-Time prediction accuracy is higher for the ML algorithms when compared to OLS. However, ML algorithms are more dependent on the data used for estimating the models, and have less stability when applied to smaller data sets. Additionally, the bias and variance trade-off from the ML algorithms has an important role in this methodology, as bias affects the index being estimated. A solution for regression algorithms that exhibit estimation bias is the use of the double imputation method.

We perform a stress test to determine sensitivity to sample size and to examine how much leverage the sample size has on the proposed methodology. This test is performed by sampling 50%, 25%, 10%, and 5% of the available transactions 30 times for each percentage level, totalling 120 samples. The number of repetitions was chosen to ensure that the hyperparameters would be estimated 30 times.

Additionally, we allow the model's predicted property prices in period t to be estimated on *multiple* periods up to and including period $t-1$. Two approaches for

[2] However, it should be noted that research in the field of real estate price *levels*, or appraised values, is quite established by now. See for example: Tay and Ho (1992), Do and Grudnitski (1992), Evans et al. (1992), Worzala et al. (1995), McGreal et al. (1998), Nghiep and Al (2001), Wong et al. (2002), Peterson and Flanagan (2009), and Kok et al. (2017).

determining the optimal window size are tested: Rolling Window (RW) with windows of 2 through 8 years and Expanding Window (EW). These tests determine the relation between model complexity (additional observations at the cost of additional year variables) and model accuracy for each of the algorithms used.

Chained index results for the full sample show that all ML algorithms have lower Root Mean Squared Error (RMSE) and higher R^2 than OLS. Regarding volatility and first-order autocorrelation (ACF1), ML algorithms have similar results to OLS; none of the indices exhibit extreme volatility or a negative ACF1. The trade-off between having more observations by using expanding or rolling windows versus the cost of including time fixed effects, affects the algorithms differently. Another finding is that while performing the stress test, ML algorithms have higher errors compared to OLS when data is restricted. This is more evident at the 5% data restriction level, which suggests a higher data dependence for the ML algorithms.

Methodology

This paper presents a single imputation Chained Paasche approach towards building real estate price indices (Balk et al., 2013). Linear and non-linear models are used for imputing the value of a real estate property in a different time period. The linear model, estimated by ordinary least squares (OLS), is used as a benchmark. For each period, hyperparameter tuning, model estimation and price prediction are performed. Therefore, there is a model estimated or trained using each algorithm for each period.

Hyperparameter tuning is performed to minimize the risk of overfitting the data, consequently lowering the out-of-sample generalization power. To best capture the price dynamics and to avoid overfitting, an optimization of a vector of hyperparameters λ is required. Therefore, a five-fold cross-validation with random search is performed to estimate the best set of hyperparameters for each regression algorithm. This optimization is explained in depth further in Section labeled "Training and bias-variance trade-off".

Chained Index

The chained hedonic approach is a series of (numeric) indices for a long sequence of periods. It is obtained by connecting price changes covering shorter time intervals. This approach controls for changes in the pool of real estate properties available in period t by specifying a "representative property". This allows the indexing process to explicitly relate property price change either to changes in the implicit prices of its characteristics or to general market conditions and omitted longitudinal variables (Geltner et al., 2017).

According to Geltner et al. (2017), the representative property can be specified relatively freely within the bounds of the sample characteristics. As a result, the chained hedonic index is a more powerful approach for exploring asset price dynamics. In this study, the representative property is the average real estate property in period t.

The chained hedonic index effectively requires separate, purely cross-sectional regression algorithms to be run on each single index period (t), hence, constituting an out-of-sample estimation. The index number formula can be defined so as to establish a Laspeyres, Paasche or Fisher index. In this research, only the Paasche index is applied.

The hedonic price modelling approach is susceptible to omitted variable bias or model misspecification as it is not possible to control for and observe every real estate characteristic. Therefore, the resulting price indices are constant quality with respect to the observed characteristics.[3]

The Chained Paasche Index (CP) is calculated each period by taking log sale prices Y_t and characteristics X_t in period t for training and then predicting log prices of properties transacted in period $t+1$ as if they had been transacted in period t, $\hat{Y}^t_{t+1}|(X_{t+1}, \mathcal{F}_t)$, where the superscript in $\hat{Y}^t_{t+1}$ denotes the price level date t, and the subscript denotes the transacted properties in period $t+1$, and $\mathcal{F}_t$ is the training set, given by (Y_t,X_t). All transactions in period t are used for training the model, and all transactions in period $t+1$ for out-of-sample predictions with price level date t. Note that $\hat{Y}^t_{t+1}|(X_{t+1}, \mathcal{F}_t)$ does not use time fixed effects, as the data used for estimation (training) is for a single period, t. Thus, within the training data set, time is a constant.

The average difference between observed log sale prices in period $t+1$ and its corresponding predictions in period t is an estimate of the log price change between period t and $t+1$, denoted by δ_{t+1},

$$\hat{\delta}_{t+1} = \frac{1}{n_{t+1}} \sum_{i \in S_{t+1}} \left(Y_{i,t+1} - \hat{Y}^t_{i,t+1}|(X_{t+1}, \mathcal{F}_t) \right), \tag{1}$$

where S_{t+1} denotes the set of transactions in period $t+1$, and n_{t+1} its corresponding number.

The Paasche Single Imputation index can be calculated as

$$\mathrm{CP}_t = \mathrm{CP}_{t-1} \times \exp(\hat{\delta}_{t+1}). \tag{2}$$

As Balk et al. (2013) point out, an alternative to the single imputation approach is to use the predicted (fitted) values $\hat{Y}^{t+1}_{t+1}|(X_{t+1}, \mathcal{F}_t + 1)$ instead of observed prices Y_{t+1}. This approach is known as double imputation. The estimated log price change between period t and $t+1$ is then given by

$$\hat{\delta}_{t+1} = \frac{1}{n_{t+1}} \sum_{i \in S_{t+1}} \left(\hat{Y}^{t+1}_{i,t+1}|\mathcal{F}_{t+1} - \hat{Y}^t_i|(X_{t+1}, \mathcal{F}_t) \right) \tag{3}$$

[3] Omitted variable bias arising from the lack of quality controls can be partially mitigated using other data sources such as text. See for example, Nowak and Smith (2020).

It can be argued that double imputation is a better way to construct indices, as biases derived from omitted variables in the model would at least partially offset each other (Balk et al., 2013). For the present study, both single and double imputation have been investigated. For simplicity, this paper focuses on the single imputation approach. Brief results of double imputation are presented in the Section labeled "Double and single imputation".

Additionally, we allow the model to predict property prices in period t to be estimated on *multiple* periods up to and including period $t-1$. We test two variants of the CP model: (1) Rolling Window (RW) and (2) Expanding Window (EW) models. Both the rolling and expanding window variants have the benefit that they use more observations to train the model. For each additional period in the training window, a time dummy variable is included in the model, increasing the model complexity (dimensionality). The window approach has the objective of analyzing the trade-off between model complexity and accuracy.

RW models are estimated based on all observations in periods $t-1$ and t (a 2-period window), so $\mathcal{F}_t = (y_t, X_t), (y_{t-1}, X_{t-1})$, and then predict the prices of properties sold in period $t+1$ as if they had been sold in t. EW models are estimated based on all observations available from the first period in the time series (2000) until period t and then predict the prices of real estate sold in period $t+1$ as if they had been sold in period t. In this approach, the extension of the training window varies (expands) over time, so $\mathcal{F}_t = (y_t, X_t), (y_{t-1}, X_{t-1}), \ldots, (y_1, X_1)$.

Machine Learning Algorithms and comparison metrics

There are several studies in the real estate literature that implement machine learning algorithms. For example, Kok et al. (2017) use machine learning to construct an Automated Valuation Model (AVM). Most of these studies benchmark traditional hedonic pricing models to Artificial Neural Networks (Tay and Ho, 1992; Do & Grudnitski, 1992; Evans et al., 1992; Worzala et al., 1995; McGreal et al., 1998; Nghiep & Al, 2001; Wong et al., 2002; Peterson & Flanagan, 2009).

As there are countless ML regression algorithms that allow us to predict the value of next period's transaction prices $\hat{Y}^t_{t+1} | (X_{t+1}, \mathcal{F}_t)$, a preselection of algorithms is necessary for practicality. The preselection phase consisted in applying several ML algorithms e.g. Lasso (Tibshirani, 1996), Random Forest (Breiman, 2001), and k-Nearest Neighbours (Altman, 1992) to the same trial dataset and selecting the four with the lowest RMSE. The algorithms which had the best performance in the preselection phase were Support Vector Regression (SVR) (Drucker et al., 1997), Extreme Gradient Boosting Tree (XGBT) (Chen & Guestrin, 2016), Neural Networks Using Model Averaging (avNNet) (Ripley, 1996) and Cubist (Quinlan & et al., 1992). These algorithms are among the most popular in the ML domain and are applied in a variety of fields.

The linear model estimated by OLS is selected as our benchmark for two reasons. First, it is the most popular regression model. Second, it is the norm for chained indices (Balk et al., 2013). SVR, XGBT, avNNet and Cubist are non-linear ML regression algorithms. Specifically, SVR is a variation of Support Vector Machine (SVM) for regression, XGBT and avNNet are algorithms that can be used for both regression and classification, while Cubist is an algorithm used exclusively for regression.

The estimated indices are assessed on two criteria: Regression model accuracy and index quality. RMSE and (adjusted) R^2 are among the most popular metrics for measuring the performance of regression algorithms. According to Steurer et al. (2019), these measures are used by several researchers for evaluating regression algorithms in the AVM field. Hence, these are the selected metrics for assessing the regression algorithms that form the indices.

For assessing the quality of price indices, Guo et al. (2014) argue that volatility and first-order autocorrelation (ACF1) of the index returns are more appropriate measures. Volatility is defined as the standard deviation of the index returns. The authors argue that noise (cross-sectional random error) in the index adds to volatility, beyond and in addition to the true volatility. Thus, noise generates excess or erroneous volatility in the index. This excess volatility reduces the ACF1 to the point that a pure noise index would yield an ACF1 of -0.5. Hence, extreme volatility and a negative ACF1 are indicative of poor index quality. Finally, the four factors established to compare the estimated price indices are: out-of-sample RMSE, R^2, volatility and the ACF1 of the index returns.

Training and bias-variance trade-off

The training of a ML algorithm is marked by the trade-off between bias (central tendency) and variance (deviations from the central tendency) of the predicted variable (Geman et al., 1992; Webb, 2000). The objective is to have the least biased model with the highest generalization power. Hyperparameters are settings used to control how flexible the resulting model should be when fitting the training data. Refinements in ML implementation emphasize stable out-of-sample performance to explicitly guard against overfitting (Gu et al., 2018). Choosing the best set of hyperparameters minimizes the occurrence of overfitting and ensures out-of-sample generalization.

Cross-Validation (CV) is a sampling technique used during training for hyperparameter optimization, which results in a nearly unbiased estimator of the generalization error given a finite sample (Scholköpf and Smola, 2002). When employing k-fold CV, the lowest bias can be achieved by setting k to n, where n equals the number of observations in the training data set. This type of CV is also known as Leave One Out Cross-Validation (LOOCV), and can be performed at the cost of increasing the error variance (Hastie et al., 2009; Smola & Scholköpf, 2004). Conversely, setting k to 10 or 5 yields the best compromise between bias and variance (Breiman & Spector, 1992; Kohavi & et al., 1995).

Considering the trade-off between bias and variance, in this paper we opt for performing a 5-fold CV with hyperparameter values selected via random search. This choice favors the balance between bias and variance. As a result, it cannot be asserted that all performed forward predictions are mean unbiased. Note, however, that the proposed methodology allows the choice of hyperparameter tuning to warrant either less bias or less variance.

Consider $\mathcal{G}_{\mathcal{X}}$ as a natural distribution with parameters Θ and $\mathcal{X}^{(train)}$ as an i.i.d. sample set drawn from $\mathcal{G}_{\mathcal{X}}$. The learning algorithm $\mathcal{A}$ operates as a function that maps the data set $\mathcal{X}^{(train)}$ to a function *f* which minimizes some expected loss $\mathcal{L}(x;\ f)$. Bergstra and Bengio (2012) argue that the ultimate goal of a standard learning algorithm $\mathcal{A}$ is to find the function *f*. A learning algorithm usually produces *f* through the optimization of a training criterion in connection with the set of parameters Θ, e.g. mean and variance of $\mathcal{G}_{\mathcal{X}}$. For the present work, the selected training criterion is RMSE.

The learning algorithm has additional features called *hyperparameters* (λ) which control the shape of function *f*. The actual learning algorithm is the one obtained after selecting λ, which can be denoted $\mathcal{A}_\lambda$, where $f = \mathcal{A}_\lambda\left(X^{(train)}\right)$ for training set $X^{(train)}$. Identifying the best set of values for the hyperparameters λ is called *hyperparameter optimization*. The objective is to choose λ that minimizes generalization error, hence

$$\mathbb{E}_{x\sim\mathcal{G}_x}\left[\mathcal{L}\left(x;\mathcal{A}_\lambda\left(\mathcal{X}^{(train)}\right)\right)\right].$$

Essentially, there are no efficient algorithms to perform the optimization:

$$\left(\hat{Y}^t_{t+1} \mid X_{t+1}, \mathcal{F}_t, \hat{\lambda}_t\right) \tag{4}$$

Additionally, it is not possible to evaluate the expectation over the unknown natural distribution$\mathcal{G}_x$. Hence, cross-validation is employed to estimate the expectation of$\mathcal{G}_x$. Cross-validation replaces the expectation with the mean over a *validation set*$\mathcal{X}^{(train)}$, whose elements are drawn i.i.d$x \sim \mathcal{G}_x$.

$$\lambda^{(*)} = \underset{\lambda\in\Lambda}{\arg\min}\,\mathbb{E}_{x\sim\mathcal{G}_x}\left[\mathcal{L}\left(x;\mathcal{A}_\lambda(\mathcal{X}^{(train)})\right)\right] \tag{5}$$

$$\approx \underset{\lambda\in\Lambda}{\arg\min}\ \underset{x\in\mathcal{X}^{(valid)}}{mean}\ \mathcal{L}\left(x;\mathcal{A}_\lambda(\mathcal{X}^{(train)})\right) \equiv \underset{\lambda\in\Lambda}{\arg\min}\,\Psi(\lambda) \tag{6}$$

$$\approx \underset{\lambda\in\{\lambda^{(1)}\ldots\lambda^{(S)}\}}{\arg\min}\ \Psi(\lambda) \equiv \hat{\lambda} \tag{7}$$

Equation 6 expresses the hyperparameter optimization problem in terms of a *hyperparameter response function*, Ψ. Hyperparameter optimization is the minimization of $\Psi(\lambda)$ over $\lambda \in \Lambda$. Different data sets, tasks, and learning algorithm families yield different sets Λ and functions Ψ. As the information about Ψ and Λ are scarce, the current dominant strategy for finding a good λ is to choose some number (*S*) of trial points $\left\{\lambda^{(1)},\ldots,\lambda^{(S)}\right\}$, to evaluate $\Psi(\lambda)$ for each one, and return the $\lambda^{(i)}$ that works the best as $\hat{\lambda}$. This strategy is represented by Eq. 7.

The trial points $\left\{\lambda^{(1)},\ldots,\lambda^{(S)}\right\}$ are randomly selected, hence the name random search. Bergstra and Bengio (2012) suggest that a random search is better than other methods, such as a grid search, because it has a higher chance of finding the global

Table 1 New York sample summary statistics

Year	Avg. Price	SD. Price	Avg. Area	SD. Area	Obs.
2000	37,478,974	96,366,158	178,061	323,727	313
2001	27,056,215	81,077,075	132,114	236,834	387
2002	30,258,411	83,920,959	141,908	258,308	450
2003	27,545,576	102,639,934	130,694	260,487	640
2004	23,705,006	60,436,108	113,049	211,754	1,077
2005	21,373,296	74,473,445	84,128	184,572	1,895
2006	26,404,120	159,581,210	79,109	274,189	2,164
2007	26,954,778	109,581,309	72,508	161,258	2,227
2008	22,357,677	112,259,746	63,023	134,594	1,354
2009	11,472,711	42,536,163	57,141	135,668	719
2010	25,275,415	99,605,492	87,926	200,933	892
2011	37,932,610	140,323,829	96,011	239,206	1,149
2012	23,290,705	81,586,946	62,811	140,172	1,907
2013	25,782,133	126,978,651	66,192	157,284	2,311
2014	29,147,148	127,380,169	68,031	166,633	2,429
2015	33,435,428	173,464,626	68,909	254,577	2,681
2016	35,058,311	158,851,647	74,161	200,809	2,205
2017	26,996,847	121,011,114	72,363	175,786	1,927
2018	29,107,151	130,042,150	63,662	144,370	2,082
2019	27,169,438	99,302,021	66,360	131,411	1,189

Avg. Price = Average Price in US Dollars; *SD. Price* = Price Standard Deviation in US Dollars; *Avg. Area* = Average Area in Square Feet; *SD. Area* = Area Standard Deviation in Square Feet; *Obs.* = Number of Observations

optima. The ML regression algorithms explored in this research have different numbers of hyperparameters.

Data

The data used in this study consist of individual commercial real estate transactions in New York over the period 2000–2019. It is divided in five regions, four property types and four building periods. For each property, the data contain the price in USD and the area in squared feet. The data are provided by Real Capital Analytics (RCA), a well-established commercial real estate transaction dataset in the United States. Founded in 2000, they collect over 90% of all transactions of "investable" real estate by now.

Observations in the bottom and top 1% of the distribution log(price)/ log(area) have been removed. The main reason for doing so is that data entry errors and outliers tend to be concentrated at the extremes of the distribution (Steurer et al., 2019). After performing the previous step and removing any observation with missing

Table 2 Categorical variables summary

Regions	Obs.	Property Type	Obs.	Building Period	Obs.
Long Island	2,014	Apartment	12,265	1 - 1732 to 1920	9,313
Manhattan	10,465	Industrial	4,848	2 - 1921 to 1931	7,057
New Jersey	4,418	Office	6,196	3 - 1932 to 1970	6,174
NYC Boroughs	11,035	Retail	6,689	4 - 1971 to 2019	7,454
Stamford	969				
Westchester	1,097				
Total			29,998		

Obs. = Number of Observations

values from the original 31,727 observations, the resulting data set is composed of 29,998 observations.

The yearly summary by transaction date is available in Table 1. A few observations can be made from this table. First of all, the average square footage of properties sold decreased substantially in the first four years. This is mostly explained by the fact that the data collecting process improved in that period.[4] This is also reflected in the relatively low number of observations in the same period. Secondly, note that the Great Financial Crisis (GFC, 2007 – 2009) is clearly visible as well. The average transaction price was \$11M during the trough of the GFC, considerably lower compared to the \$27M long run average. The number of observations also fell almost by half between 2008 and 2009, falling from 1,354 to 719.

A categorical variable for building period has been created by splitting the data into quartiles of the building date. Information about the categorical variables is available in Table 2. Note that over 70% of the transactions happen in just Manhattan and the NYC Boroughs (Brooklyn, Queens, Bronx and Staten Island). Likewise, over 40% of the transactions are income producing apartment complexes.

Results

This section first presents chained index estimates, followed by stress tests that analyze the sensitivity of the results to the sample size. The Section labeled "Optimal Window Size" contrasts estimation results using rolling and expanding windows. Finally, the bias-variance trade-off confirms that double imputation is better for regression algorithms that suffer from estimation biases.

[4] Before 2005, RCA used a \$5M lower boundary threshold for collecting data. They lowered this number to \$2.5M in 2005. Even though they then retroactively collected transactions of properties sold for less than \$5M for the period 2000–2004, it is probably still incomplete.

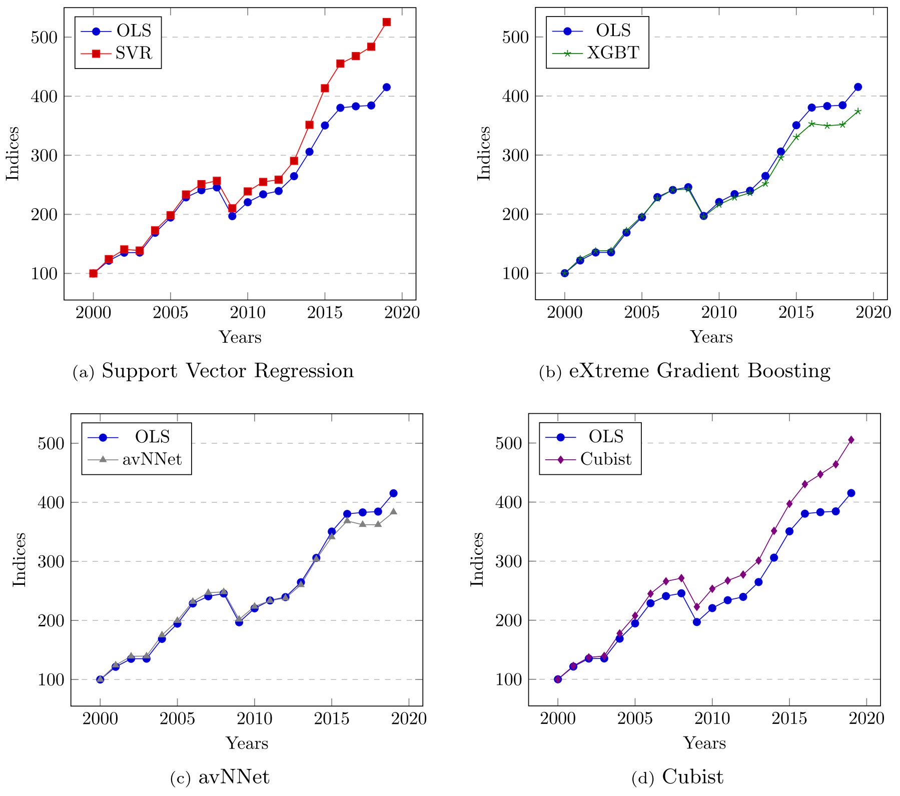


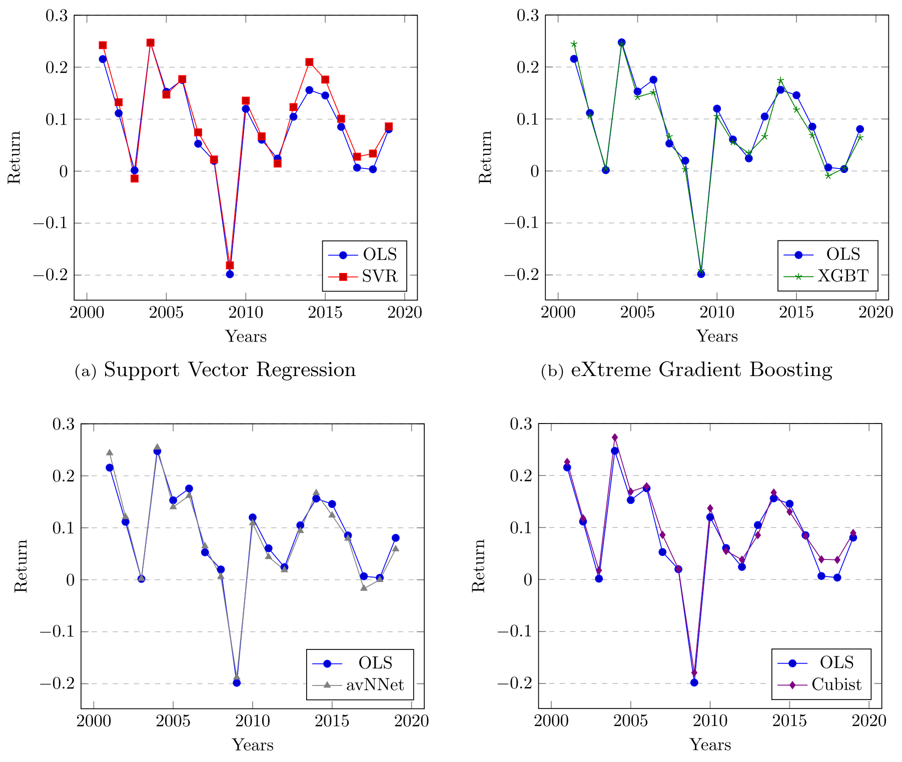
Fig. 1 New York Chained Paasche Price Indices. *OLS* = Ordinary Least Squares; *XGBT* = Extreme Gradient Boosting Tree; *SVR* = Support Vector Regression; *avNNet* = Neural Networks Using Model Averaging

Chained index results

Figure 1 shows the yearly CP indices. It shows that SVR (Fig. 1a) and Cubist (Fig. 1d) indices have higher cumulative price changes compared to OLS. Also, XGBT (Fig. 1b) and avNNet (Fig. 1c) have almost identical results as OLS for most of the time, but are slightly lower in the last five years.

Figure 2 shows the yearly CP index returns. Despite the fact that the indices are different in levels (see Fig. 1), these differences is not that evident when looking at the index returns (first differences). Among all the regression algorithms, the most notable differences in the returns are displayed by SVR (Fig. 2a) and Cubist (Fig. 2d), both of which yield overall higher returns than OLS.

Figure 3 and Table 3 show that overall, the non-linear models have a lower RMSE and a higher R^2 when compared to OLS. This is expected, as non-linear models are more flexible and usually yield a better fit to the data, especially when the data is complex and sparse as is the case for real estate. Table 3 shows that for all the



(a) Support Vector Regression (b) eXtreme Gradient Boosting

(c) avNNet (d) Cubist

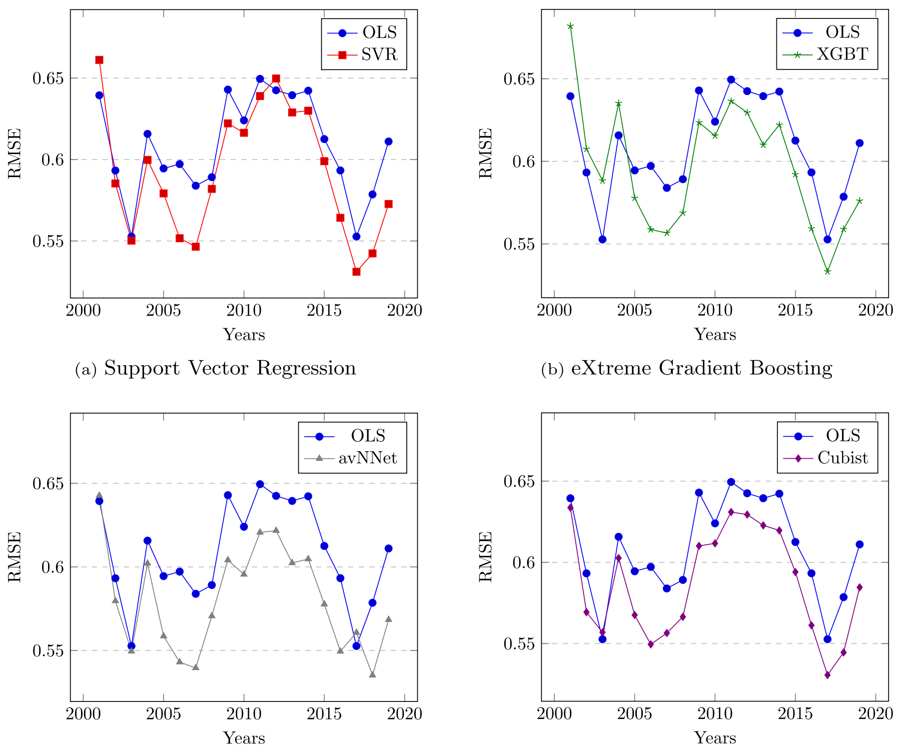
Fig. 2 New York Chained Paasche Price Returns. *OLS* = Ordinary Least Squares; *XGBT* = Extreme Gradient Boosting Tree; *SVR* = Support Vector Regression; *avNNet* = Neural Networks Using Model Averaging

regression algorithms, the index returns have a non-negative first-order autocorrelation and do not exhibit extreme volatility.

Since the commercial real estate market in New York is distinct for each property type, an index per property type has been calculated. Figure 9a exhibits the index for apartments, Fig. 9b for retail, Fig. 9c for offices and Fig. 9d for industrial. Figure 10a, b, c and d present the respective RMSE values for the underlying algorithms.

As the regression algorithms used to estimate the property type indices were trained on a restricted data set, the RMSE for the ML algorithms are higher when compared to the indices with the full sample. Apartment is the most prevalent property type. Therefore, the ML algorithms show a lower RMSE than the benchmark, with the only exception being SVR. XGBT and avNNet produce an index almost identical to the one generated by the benchmark, while SVR and Cubist have much higher values.

Industrial properties are the type with the fewest transactions. The RMSE for this category is around the same level for all the algorithms used, with ML being below



(a) Support Vector Regression (b) eXtreme Gradient Boosting (c) avNNet (d) Cubist

Fig. 3 New York Chained Paasche Indices RMSEs. *OLS* = Ordinary Least Squares; *XGBT* = Extreme Gradient Boosting Tree; *SVR* = Support Vector Regression; *avNNet* = Neural Networks Using Model Averaging; *RMSE* = Root Mean Squared Error

Table 3 Chained Paasche performance summary

	RMSE	Residual Mean	Rsquared	Volatility	ACF(1)
OLS	0.6081	0.0749	0.7473	0.0991	0.1360
XGBT	0.5964	0.0694	0.7588	0.0992	0.1693
SVR	0.5922	0.0873	0.7610	0.1020	0.1169
avNNet	0.5803	0.0707	0.7709	0.1012	0.1719
Cubist	0.5864	0.0853	0.7657	0.0968	0.1317

OLS = Ordinary Least Squares; *XGBT* = Extreme Gradient Boosting Tree; *SVR* = Support Vector Regression; *avNNet* = Neural Networks Using Model Averaging; *RMSE* = Root Mean Squared Error; *Volatility* = Index Return Standard Deviation; *ACF(1)* = Index Return First-Order Autocorrelation

the benchmark in some years and above in others. The variation of the RMSE values is higher for the ML algorithms when compared to the benchmark. For this property type, XGBT has a very similar index to the benchmark, whereas SVR and Cubist have much higher index values. avNNet is the only regression algorithm that exhibits an index with a lower value than the benchmark.

Office and Retail have similar numbers of transactions and the RMSE of the ML algorithms is usually higher than the benchmark. SVR, XGBT and avNNet generated indices with values lower than the benchmark. Cubist produces an index that has lower values than the benchmark up until 2015, but after that the index values are higher.

The restriction per property type imposed in the training sample limits the performance of the ML algorithms. This is more evident after examining the RMSE plots, which show no conclusive improvement when compared to the benchmark.

Stress Test

In order to check model and index stability, a stress test has been performed by sampling 50%, 25%, 10% and 5% of the available data, 30 times for each percentage level, so in total 120 samples. An index has been generated for each sample.

Figure 4 shows the average index return for each percentage level. Interestingly, all indices exhibit a higher volatility at the 5% level, detaching from the other percentage levels in years 2001, 2004 and 2011. Table 4 presents summary statistics of the volatility (standard deviation of the index returns) per percentage level. All ML algorithms have a similar behaviour, displaying almost no noticeable change in the index until the 5% mark. Only at the 5% level is there a significant index return change, with the index return exhibiting a higher volatility when compared to the full sample. It is possible to corroborate the results in Fig. 4 by examining the mean volatility of the 30 samples in Table 4, where, only at the 5% level, a significant increase in volatility of the returns is noticeable for all regression algorithms.

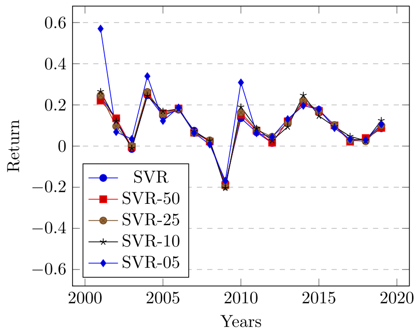
Figure 5 presents the average RMSE over all stress tests performed. The RMSE for the ML algorithms is noticeably more sensitive to data availability. Nevertheless, for several percentage levels, the RMSE for the ML algorithms is lower than the OLS benchmark. Starting from the 10% level, the RMSE of the ML algorithms increases, decoupling from the previous levels. At the 5% level all ML algorithms display a higher RMSE than the OLS benchmark.

Table 4 provides insight in how “stable” the index returns remain when using less observations (see Francke and Van de Minne (2017), who employ a similar stress test). On average, the mean of the volatility of the returns for OLS is lower than the ML algorithms for all suppressed percentage levels, except for the 5% level where XGBT and Cubist have a lower average return volatility. A common trend among all regression algorithms is that the volatility of the returns increases as more data are suppressed. The standard deviation of the return volatility also increases when more data are removed. Only at the 50% and 10% levels is the standard deviation of the return volatility from the benchmark lower than the ML algorithms.

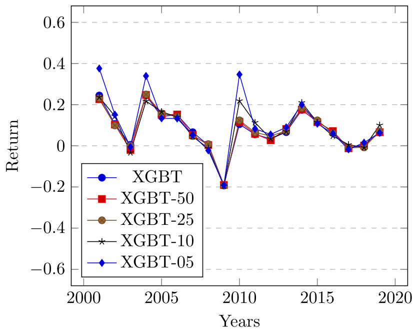
Table 4 Summary statistics of the volatility from the 30 samples with 50%, 25%, 10% and 5% of the available observations

		50%	25%	10%	5%
OLS	Min.	0.0885	0.0915	0.0970	0.1422
	1st Qu.	0.1009	0.1025	0.1312	0.1947
	Median	0.1054	0.1106	0.1495	0.2251
	Mean	0.1046	0.1138	0.1466	0.2471
	3rd Qu.	0.1079	0.1179	0.1536	0.2623
	Max.	0.1239	0.1679	0.1836	0.5313
	SD.	0.0071	0.0171	0.0220	0.0975
XGBT	Min.	0.0900	0.0887	0.1171	0.1403
	1st Qu.	0.0975	0.1060	0.1387	0.1871
	Median	0.1066	0.1120	0.1486	0.2187
	Mean	0.1053	0.1144	0.1581	0.2258
	3rd Qu.	0.1102	0.1190	0.1696	0.2417
	Max.	0.1224	0.1574	0.2369	0.4328
	SD.	0.0089	0.0139	0.0316	0.0629
SVR	Min.	0.0925	0.0930	0.0973	0.1324
	1st Qu.	0.1023	0.1079	0.1396	0.2038
	Median	0.1077	0.1198	0.1610	0.2211
	Mean	0.1081	0.1194	0.1583	0.2621
	3rd Qu.	0.1128	0.1287	0.1729	0.2681
	Max.	0.127	0.1610	0.2112	1.0164
	SD.	0.0082	0.0147	0.0275	0.1546
avNNet	Min.	0.0856	0.0898	0.1010	0.1466
	1st Qu.	0.1015	0.1114	0.1331	0.2086
	Median	0.1064	0.1179	0.1539	0.2468
	Mean	0.1079	0.1218	0.1544	0.253
	3rd Qu.	0.1109	0.1295	0.1672	0.2896
	Max.	0.1283	0.1894	0.2401	0.3852
	SD.	0.0103	0.0198	0.0308	0.0573
Cubist	Min.	0.0846	0.0903	0.1180	0.1430
	1st Qu.	0.1014	0.1080	0.1342	0.1819
	Median	0.1057	0.1167	0.1534	0.2261
	Mean	0.1055	0.1174	0.1573	0.2281
	3rd Qu.	0.1120	0.1255	0.1846	0.2642
	Max.	0.1206	0.1445	0.2024	0.3789
	SD.	0.0089	0.0132	0.0259	0.0576

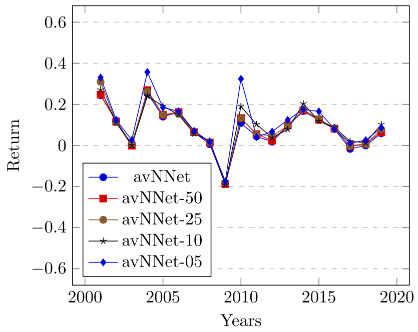
Index return volatility is, on average, lower for OLS when compared to ML algorithms. Also, only at the 5% level a significant increase in index return volatility is noticeable for all regression algorithms; *Min.* = Minimum volatility; *1st Qu.* = First Quartile volatility; *3rd Qu.* = Third Quartile volatility; *Max.* = Maximum volatility; *SD.* = Standard Deviation of index volatility



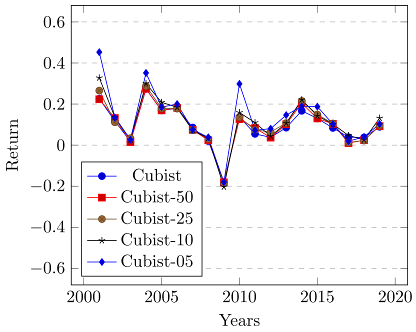
(a) Support Vector Regression



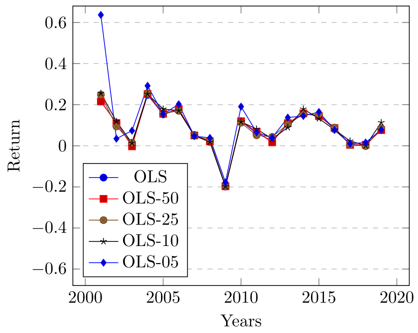
(b) eXtreme Gradient Boosting



(c) avNNet



(d) Cubist

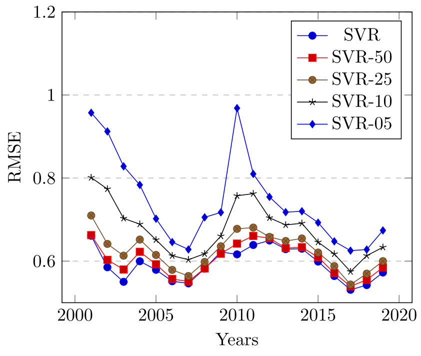


(e) OLS

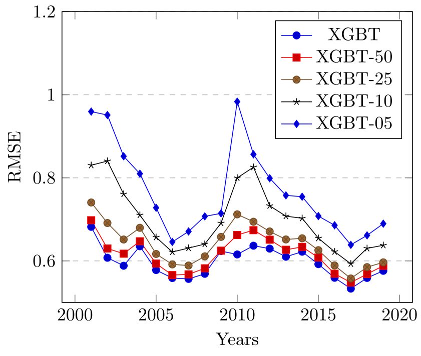
Fig. 4 Returns for Stress Tests. All the indices returns are similar until the 10% level. At the 5% level, indices returns exhibit a higher volatility, detaching from the other percentage levels in years 2001, 2004 and 2011. Numbers refer to the percentage of the sample size being used: 50, 25, 10 and 5 percent. When omitted the full sample has been used

Optimal Window Size

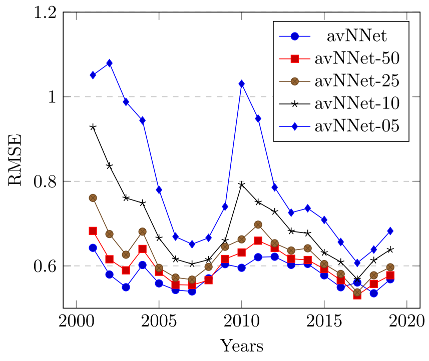
This subsection presents model results from both rolling (RW) and expanding windows (EW) samples. The main motivation for investigating alternative samples is to account for the trade-off between model complexity and out-of-sample accuracy.



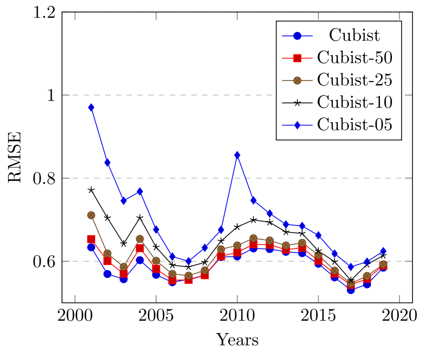
(a) Support Vector Regression



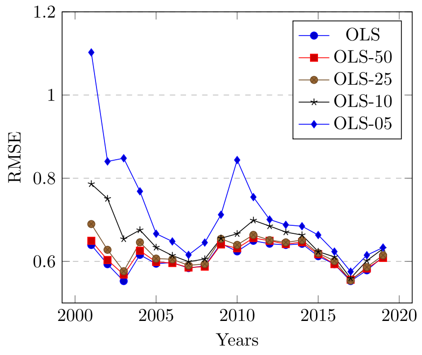
(b) eXtreme Gradient Boosting



(c) avNNet



(d) Cubist



(e) OLS

Fig. 5 RMSEs for Stress Tests. The RMSE for the ML algorithms is noticeably more sensitive to data availability as it is higher then OLS. Starting from the 10% level, the RMSE of the ML algorithms increases, decoupling from the previous levels. At the 5% level all ML algorithms display a higher RMSE than the OLS benchmark. Numbers refer to the percentage of the sample size being used: 50, 25, 10 and 5 percent. When omitted the full sample has been used

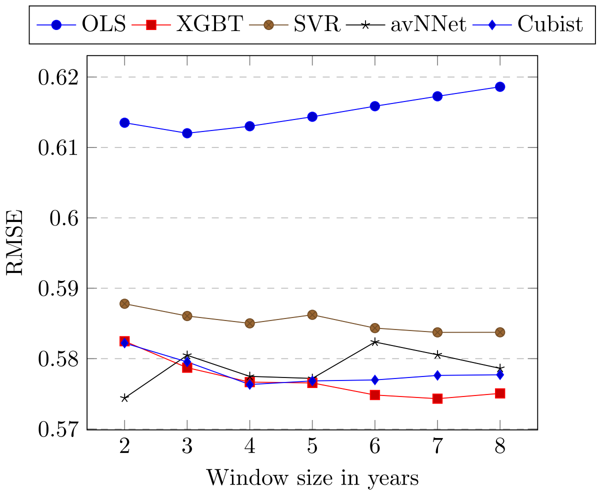


Fig. 6 RMSEs per Window Size. Each regression algorithm behaves differently, depending on the training window size. The best performing algorithms are avNNet with a window size of 2 years and XGBT with a window size of 7 years; *RMSE* = Root Mean Squared Error

Table 5 Optimal window performance summary

	RMSE	Residual Mean	R^2	Volatility	ACF(1)	OWS
OLS	0.6120	0.0432	0.7283	0.0913	0.2203	3
XGBT	0.5751	0.0454	0.7600	0.0901	0.1678	7
SVR	0.5837	0.0567	0.7526	0.1024	0.2062	7
avNNet	0.5744	0.0413	0.7605	0.0892	0.1584	2
Cubist	0.5763	0.0651	0.7590	0.0951	0.3031	4

ML algorithms display a lower RMSE then the OLS benchmark. The lowest RMSE pertains to avNNet with a window size of 2 years; *RMSE* = Root Mean Squared Error; *ACF(1)* = Index Return First-Order Autocorrelation; *Volatility* = Index Return Standard Deviation; *OWS* = Optimal Window Size in Years

A larger time window size is associated with more observations and more year control variables, thus adding more dimensions and complexity to the models. Up to an Optimal Window Size (OWS), the addition of an extra year is expected to have an overall benefit of increasing the out-of-sample accuracy, as the model fits the data better and becomes more generalizable. The balance between model complexity and out-of-sample accuracy can be explored by testing different window sizes and measuring the impact in terms of out-of-sample RMSE.

In our search for the optimal window size, we test windows of 2 to 8 years. . Window sizes larger than 8 years would hamper this exercise by being too limiting on the data availability for training. The test data used for the different window sizes must be identical to allow for a fair comparison of results. Therefore, for each window size m, the regression algorithms have been trained on data in the years $t - m + 1$ to t and predicted for year $t + 1$, where $t + 1 =$ 2008,…,2019.

Table 6 Expanding window performance summary

	RMSE	Residual Mean	R^2	Volatility	ACF(1)
OLS	0.6186	0.0414	0.7226	0.0914	0.2092
XGBT	0.5761	0.0376	0.7588	0.0898	0.2405
SVR	0.5852	0.0593	0.7513	0.1010	0.2110
avNNet	0.5791	0.0416	0.7560	0.0914	0.1633
Cubist	0.5773	0.0594	0.7584	0.0926	0.3182

Expanding window performed worst than the optimal rolling window or using only one year as in the chained index methodology. XGBT regression algorithm has the best performance for the expanding window methodology; *RMSE* = Root Mean Squared Error; *ACF(1)* = Index Return First-Order Autocorrelation; *Volatility* = Index Return Standard Deviation

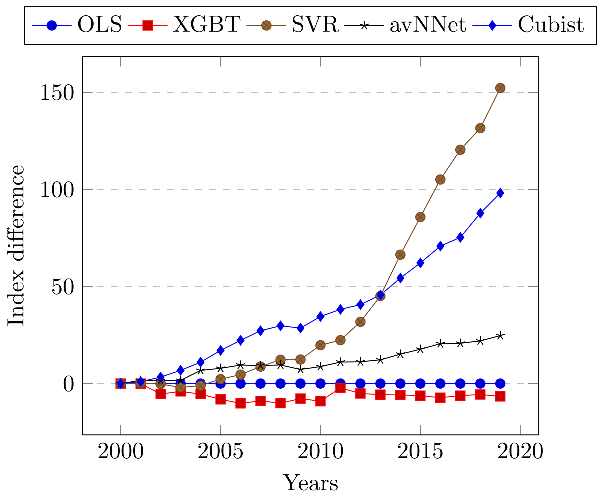


Fig. 7 Cumulative difference between single and double imputation. All ML algorithms exhibit a difference between single and double imputation, SVR and Cubist have the highest cumulative difference; *Index difference* = Single minus double imputation difference

Figure 6 presents out-of-sample RMSEs for the different algorithms and window sizes. The figure shows that each regression algorithm behaves differently, depending on the training window size. This shows that the out-of-sample model performance and corresponding indices are quite sensitive to the window length. Table 5 provides model and index performance statistics for the optimal window size for each algorithm. Table 6 presents the results for the EW approach, which has a lower performance when compared to RW. The results for RW and EW suggest that the optimal window size is bigger than two years and smaller than using all years as in EW. The best performing algorithms are avNNet with a window size of 2 years and XGBT with a window size of 7 years.

Fig. 8 New York Chained Paasche Price Indices with Double Imputation. Starting from year 2013 all the indices built using the ML regression algorithms are below the benchmark, different from the single imputation approach, where SVR and Cubist had all their values above the benchmark

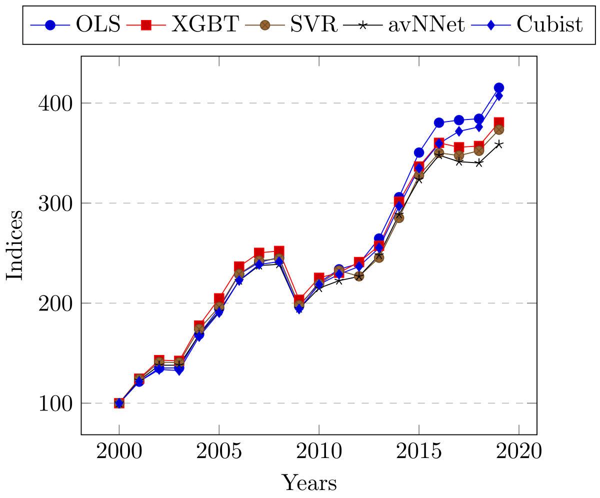


Table 7 Difference between true and fitted log price means

Year	OLS	XGBT	SVR	avNNet	Cubist
2001	0.0000	−0.0056	0.0014	0.0134	0.0081
2002	0.0000	−0.0410	−0.0032	−0.0029	0.0165
2003	0.0000	0.0017	−0.0123	0.0013	0.0254
2004	0.0000	0.0026	0.0078	0.0276	0.0141
2005	0.0000	0.0023	0.0178	0.0007	0.0217
2006	0.0000	−0.0013	0.0083	0.0017	0.0095
2007	0.0000	−0.0011	0.0155	−0.0033	0.0127
2008	0.0000	−0.0017	0.0136	0.0007	0.0082
2009	0.0000	−0.0006	0.0117	−0.0026	0.0211
2010	0.0000	0.0037	0.0257	0.0032	0.0094
2011	0.0000	−0.0021	0.0055	0.0087	0.0077
2012	0.0000	−0.0002	0.0393	−0.0003	0.0042
2013	0.0000	0.0006	0.0376	−0.0001	0.0058
2014	0.0000	−0.0001	0.0405	0.0030	0.0037
2015	0.0000	0.0000	0.0232	0.0020	0.0021
2016	0.0000	0.0006	0.0300	0.0041	0.0095
2017	0.0000	−0.0015	0.0350	0.0017	0.0047
2018	0.0000	0.0002	0.0198	0.0036	0.0253
2019	0.0000	−0.0054	0.0246	0.0040	0.0062

The increase in variance of the fitted values for the ML regression algorithms comes at the cost of having some estimation bias. The OLS benchmark has no estimation bias

Table 8 Difference between true and fitted log price variance

Year	OLS	XGBT	SVR	avNNet	Cubist
2001	0.3302	0.3768	0.2315	0.3271	0.3162
2002	0.3127	0.2053	0.3234	0.2919	0.2467
2003	0.2954	0.3239	0.2296	0.2711	0.2838
2004	0.3136	0.2865	0.2683	0.5195	0.2838
2005	0.3242	0.2838	0.2382	0.2833	0.2740
2006	0.3206	0.2801	0.1976	0.2747	0.2151
2007	0.3333	0.2633	0.2489	0.2936	0.2638
2008	0.3252	0.2648	0.2773	0.6765	0.2581
2009	0.3323	0.3042	0.3230	0.2880	0.3039
2010	0.3566	0.3263	0.3533	0.3736	0.2768
2011	0.3885	0.3550	0.3367	0.3377	0.3387
2012	0.3922	0.4243	0.3393	0.3630	0.3120
2013	0.3903	0.3517	0.3345	0.3558	0.3095
2014	0.3812	0.3640	0.3025	0.3151	0.2741
2015	0.3466	0.3034	0.2993	0.3414	0.2864
2016	0.3349	0.3574	0.2761	0.3023	0.2441
2017	0.2988	0.3472	0.2528	0.2737	0.2147
2018	0.3218	0.3190	0.2458	0.2693	0.2740
2019	0.3570	0.2638	0.2645	0.3040	0.2175

All the ML regression algorithms present a fitted variance closer to the true variance when compared to the OLS benchmark

Double and single imputation

When comparing the results obtained using single and double imputation (Fig. 7) one can notice a significant cumulative difference of single minus double imputation for the ML algorithms. SVR and Cubist have the highest cumulative difference. Conversely, avNNet shows a moderate cumulative difference, while XGBT is the only ML algorithm that exhibits a negative difference between indices using the true or fitted values. OLS shows no cumulative difference because the fitted mean is equal to the true mean by definition (when a constant is included).

As presented in the Section labeled "Training and bias-variance trade-off", there is a trade-off between bias and variance that should be taken into consideration when selecting and evaluating ML algorithms. Algorithms with higher fitted variance have a tendency to overfit, which implies that the mean of the fitted values will be equal or virtually equal to the mean of the true (observed) values as the bias is close to zero.

Table 7 shows the difference between true mean log price and fitted mean log price. All ML algorithms display values different from zero, which indicates bias.

On the other hand, Table 8 shows that all ML algorithms have a lower difference between true and fitted variance. All ML algorithms have a higher fitted variance when compared to the benchmark. Hence, Tables 7 and 8 display and quantify the bias-variance trade-off for this application.

The double imputation approach is preferred for the ML algorithms as the bias in the fitted value $\left(\hat{Y}_{t+1}^{t+1}|\mathcal{F}_{t+1}\right)$ and the Out-of-Time prediction $\left(\hat{Y}^{t}|(X_{t+1},\mathcal{F}_{t})\right)$ will partially offset each other. Hill and Melser (2008) also suggest double imputation for similar reasons. For the purpose of price index construction, it is necessary to be aware of the estimation bias, as this can be transferred into the index. A biased model can potentially produce an index that over or under estimates long-term price trends, especially over long time periods, as the index is built using prior index values and the bias will accumulate.

Figure 8 shows the price indices for the New York commercial real estate market using double imputation. Notice that after the year 2013 all the indices built using the ML regression algorithms are below the benchmark, different from the single imputation approach, where SVR and Cubist had all their values above the benchmark.

Additionally, one of the central assumptions of this paper is that the difference of the means is the periodic price change. With a biased estimator this difference will be composed of the periodic price change plus bias. Hence, when using the proposed methodology, the bias-variance trade-off should be taken into consideration as well as the measures to attenuate or eliminate the estimation bias. As mentioned in the Section labeled "Training and bias-variance trade-off", in the context of this paper, increasing the number of folds during training might be a solution to mitigate estimation bias during training.

As a further check, Welch's *t*-tests were performed in all actual/fitted price pairs to test the hypothesis that the two populations have equal mean. The test results, such as *t*-values, *p*-values and confidence intervals can be seen in the Appendix labeled "Welch's *t*-Test Results". Note that the algorithms which exhibit a higher difference between single and double imputation indices are the ones with lower *p*-values on average. However, with a 95% confidence interval, none of the tests rejected the null hypothesis. This result indicates the likelihood that the two populations have equal means (Tables 9, 10, 11, 12 and 13).

Conclusion

This paper presents a model-agnostic approach based on Out-of-Time individual transaction predictions to build price indices for commercial real estate using a variety of non-linear machine learning (ML) algorithms. The key innovation is the use of prediction error to measure time trends. The results obtained support the viability of using ML for constructing price indices. Overall, the non-linear ML algorithms yielded higher accuracy and lower volatility with non-negative first-order autocorrelation of index returns.

The comparison between single and double imputation shows that some of the ml algorithms have display estimation bias. Using the proposed methodology for index construction requires attention to the bias and variance tradeoff. The findings also highlight the importance of the hyperparameter selection phase in minimizing the introduction of bias while keeping the out-of-sample generalization power. Regression algorithms that exhibit estimation bias could use the double imputation method as a straightforward way to reduce the bias problem.

The stress tests show that linear models (OLS) generate overall more stable indices when few training data are available than the non-linear ML regression algorithms used in this paper, as linear models are less dependent on the number of observations. Also, looking at the index volatility in the stress test, OLS has lower values, on average, than the ML algorithms. Additionally, the variations of the loss function across the tests with 50, 25, 10 an 5 percent of the data are higher for the ML algorithms, especially at the 5% level. The RMSE from the property type indices corroborate the idea that ML regression algorithms are more dependent on sample size. These indices were generated using regression algorithms estimated using only the property type sample. Data sets composed of property types like apartment that contain more observations produce lower RMSE when compared to the other types with fewer observations, such as industrial real estate.

The analysis of the optimal window size for the Rolling Window (RW) approach demonstrates that the magnitude of the optimal window varies greatly across the different algorithms; the window sizes range from 2 to 8 years. The single year window corresponds to the Chained Paasche index (CP) and is ruled out, as the RMSE is higher than both the RW and the Expanding Window (EW).

Considering all the tests performed in this study, it is possible to conclude that in cases where more observations are available, even at the cost of adding more dimensions (controls or features), ML algorithms tend to produce better results than OLS. Cases where few observations or characteristics are accessible favor OLS, as it performs better in restricted data sets when compared to ML.

Appendix A: New York Indices and Errors per Property Type

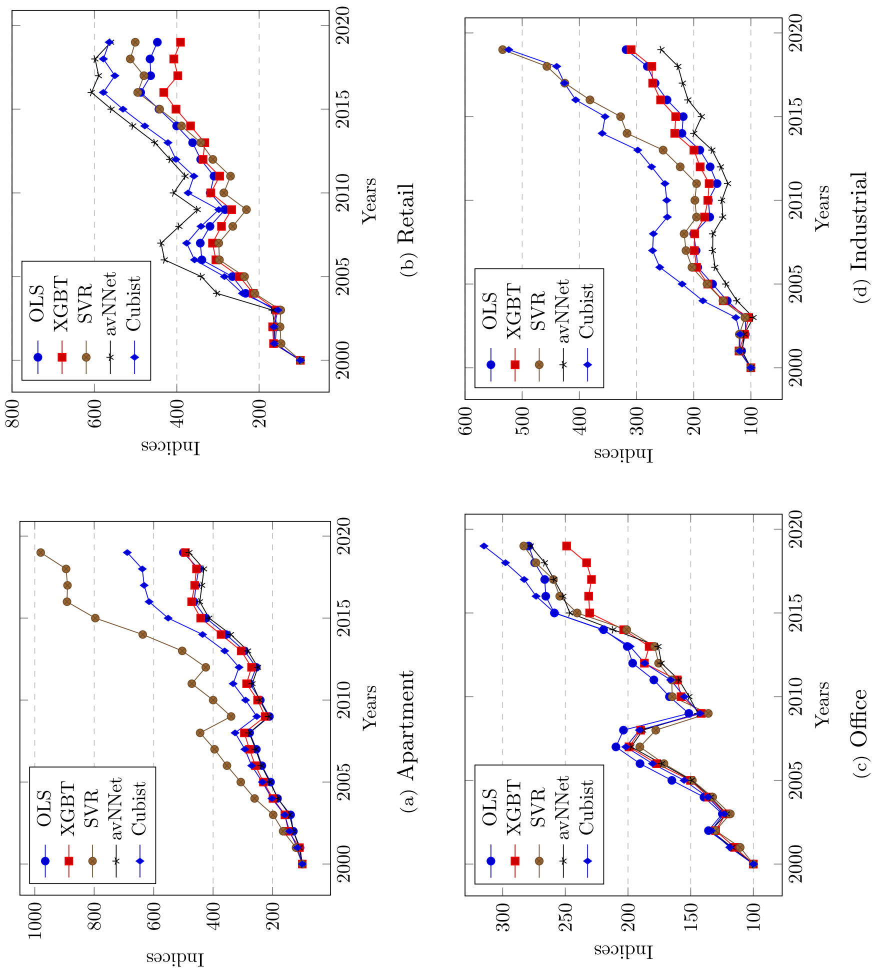
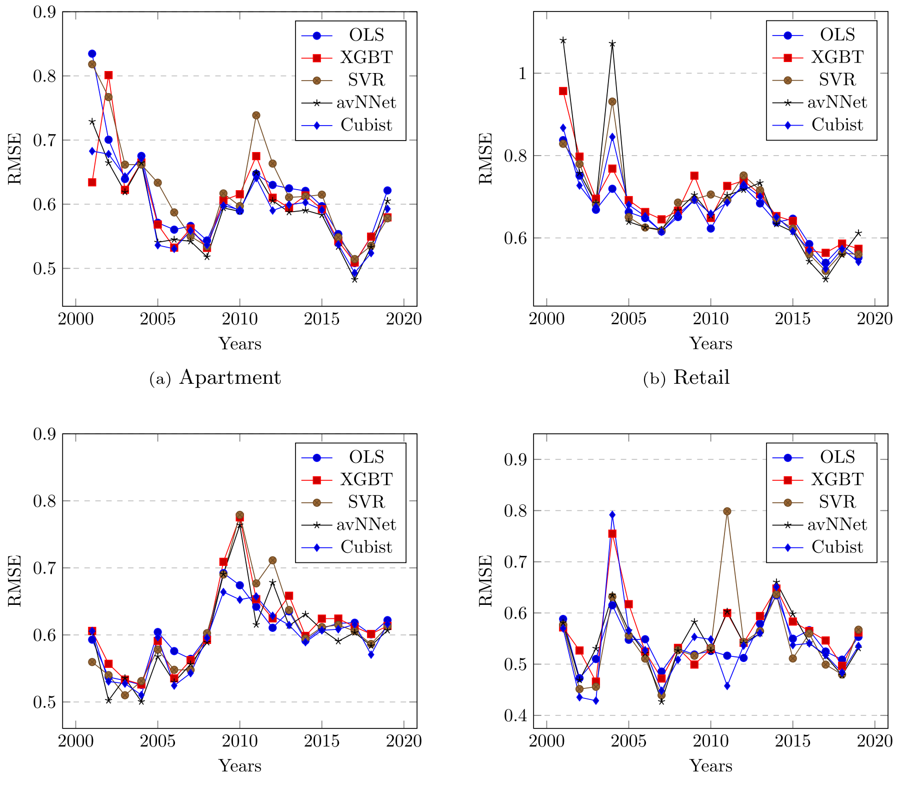


Fig. 9 New York Chained Paasche Price Indices per Property Type. *OLS* = Ordinary Least Squares; *XGBT* = Extreme Gradient Boosting Tree; *SVR* = Support Vector Regression; *avNNet* = Neural Networks Using Model Averaging



(a) Apartment (b) Retail

(c) Office (d) Industrial

Fig. 10 New York Apartment Chained Paasche Indices per Property Type RMSEs. *OLS* = Ordinary Least Squares; *XGBT* = Extreme Gradient Boosting Tree; *SVR* = Support Vector Regression; *avNNet* = Neural Networks Using Model Averaging; *RMSE* = Root Mean Squared Error

Appendix B: Welch's *t*-Test Results

Table 9 OLS *t*-test Results

Sample Year	*t*-value	*p*-value	LBCI	UBCI
2000	−0.0358	0.9715	−0.2644	0.2549
2001	0.0149	0.9881	−0.1908	0.1937
2002	0.0156	0.9876	−0.1839	0.1868
2003	0.0168	0.9866	−0.1454	0.1480
2004	−0.0148	0.9882	−0.1059	0.1043
2005	−0.0177	0.9859	−0.0708	0.0695
2006	0.0002	0.9999	−0.0650	0.0650
2007	0.0070	0.9944	−0.0660	0.0665
2008	−0.0245	0.9805	−0.0851	0.0830
2009	0.0072	0.9943	−0.0918	0.0924
2010	−0.0088	0.9930	−0.1054	0.1045
2011	0.0023	0.9982	−0.1067	0.1070
2012	0.0115	0.9908	−0.0698	0.0706
2013	0.0072	0.9943	−0.0630	0.0634
2014	−0.0176	0.9859	−0.0614	0.0603
2015	−0.0122	0.9902	−0.0610	0.0602
2016	−0.0233	0.9814	−0.0676	0.0660
2017	−0.0080	0.9936	−0.0690	0.0685
2018	−0.0100	0.9920	−0.0678	0.0671

LBCI = Lower Bound of the Confidence Interval; *UBCI* = Upper Bound of the Confidence Interval

Table 10 XGBT *t*-test Results

Sample Year	*t*-value	*p*-value	LBCI	UBCI
2000	−0.0245	0.9804	−0.2626	0.2561
2001	0.0393	0.9687	−0.1852	0.1928
2002	0.0635	0.9494	−0.1801	0.1921
2003	−0.0741	0.9409	−0.1501	0.1392
2004	0.0197	0.9843	−0.1044	0.1065
2005	−0.1352	0.8925	−0.0754	0.0656
2006	−0.0252	0.9799	−0.0664	0.0647
2007	0.0414	0.9670	−0.0654	0.0682
2008	−0.0408	0.9675	−0.0843	0.0809
2009	−0.0691	0.9449	−0.0948	0.0884
2010	−0.0137	0.9891	−0.1062	0.1047
2011	0.0460	0.9633	−0.1045	0.1095
2012	0.0875	0.9303	−0.0664	0.0726
2013	0.1258	0.8999	−0.0594	0.0675
2014	0.0106	0.9916	−0.0601	0.0607
2015	−0.0582	0.9536	−0.0616	0.0580
2016	−0.0702	0.9440	−0.0687	0.0640
2017	0.0184	0.9853	−0.0681	0.0694
2018	−0.0635	0.9494	−0.0699	0.0655

LBCI = Lower Bound of the Confidence Interval; *UBCI* = Upper Bound of the Confidence Interval

Table 11 SVR *t*-test Results

Sample Year	*t*-value	*p*-value	LBCI	UBCI
2000	−0.1645	0.8694	−0.2836	0.2397
2001	−0.1158	0.9078	−0.2037	0.1810
2002	0.1279	0.8982	−0.1724	0.1964
2003	0.0403	0.9679	−0.1430	0.1490
2004	−0.2014	0.8404	−0.1141	0.0928
2005	−0.6683	0.5040	−0.0951	0.0468
2006	−0.2633	0.7924	−0.0754	0.0575
2007	−0.4853	0.6275	−0.0840	0.0507
2008	−0.1754	0.8608	−0.0924	0.0772
2009	−0.2705	0.7868	−0.1045	0.0791
2010	−0.6156	0.5382	−0.1386	0.0724
2011	0.1647	0.8692	−0.0995	0.1177
2012	−0.8191	0.4128	−0.1010	0.0415
2013	−0.9747	0.3297	−0.0957	0.0321
2014	−1.2794	0.2008	−0.1019	0.0214
2015	−0.6992	0.4845	−0.0832	0.0394
2016	−0.9726	0.3308	−0.1009	0.0340
2017	−0.8629	0.3882	−0.0999	0.0388
2018	−0.6286	0.5296	−0.0903	0.0465

LBCI = Lower Bound of the Confidence Interval; *UBCI* = Upper Bound of the Confidence Interval

Table 12 avNNet *t*-test Results

Sample Year	*t*-value	*p*-value	LBCI	UBCI
2000	−0.0792	0.9369	−0.2716	0.2505
2001	0.0012	0.9990	−0.1925	0.1928
2002	0.0436	0.9652	−0.1817	0.1900
2003	−0.0210	0.9832	−0.1483	0.1452
2004	−0.0475	0.9621	−0.1079	0.1028
2005	−0.0156	0.9876	−0.0712	0.0701
2006	−0.1705	0.8646	−0.0714	0.0600
2007	−0.0448	0.9643	−0.0682	0.0652
2008	−0.1927	0.8472	−0.0930	0.0764
2009	−0.2213	0.8249	−0.1040	0.0829
2010	−0.0046	0.9963	−0.1054	0.1049
2011	−0.0099	0.9921	−0.1081	0.1070
2012	−0.0583	0.9535	−0.0729	0.0687
2013	0.0572	0.9544	−0.0619	0.0657
2014	0.0511	0.9592	−0.0598	0.0630
2015	0.1346	0.8929	−0.0569	0.0652
2016	−0.1186	0.9056	−0.0715	0.0633
2017	0.0599	0.9522	−0.0670	0.0712
2018	−0.1053	0.9162	−0.0716	0.0643

LBCI = Lower Bound of the Confidence Interval; *UBCI* = Upper Bound of the Confidence Interval

Table 13 Cubist *t*-test Results

Sample Year	*t*-value	*p*-value	LBCI	UBCI
2000	−0.0069	0.9945	−0.2632	0.2613
2001	−0.1327	0.8944	−0.2057	0.1796
2002	−0.1619	0.8715	−0.2015	0.1708
2003	−0.3393	0.7344	−0.1723	0.1215
2004	−0.3779	0.7055	−0.1256	0.0850
2005	−0.6837	0.4942	−0.0958	0.0463
2006	−0.2806	0.7790	−0.0757	0.0567
2007	−0.5355	0.5924	−0.0856	0.0488
2008	−0.1580	0.8745	−0.0916	0.0779
2009	−0.4876	0.6259	−0.1158	0.0697
2010	−0.2805	0.7791	−0.1215	0.0911
2011	−0.2065	0.8364	−0.1187	0.0961
2012	−0.0990	0.9212	−0.0748	0.0676
2013	−0.0371	0.9704	−0.0654	0.0630
2014	−0.0699	0.9443	−0.0643	0.0599
2015	−0.1221	0.9028	−0.0652	0.0575
2016	−1.1266	0.2600	−0.1057	0.0286
2017	−0.9633	0.3354	−0.1031	0.0352
2018	−0.6509	0.5152	−0.0906	0.0454

LBCI = Lower Bound of the Confidence Interval; *UBCI* = Upper Bound of the Confidence Interval

Acknowledgements The authors would like to express special thanks to: Real Capital Analytics for providing the data; John Clapp, Andy Krause, David Geltner, Massachusetts Institute of Technology, Maastricht University Research Seminar, the two anonymous referees and the guest editor for their helpful comments and suggestions.

References

Altman, N.S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, *46*(3), 175–185.

Bailey, M.J., Muth, R.F., & Nourse, H.O. (1963). A regression method for real estate price index construction. *Journal of the American Statistical Association*, *58*(304), 933–942.

Balk, B., Diewert, W.E., Fenwick, D., Prud'homme, M., & de Haan, J. (2013). Handbook on residential property prices indices (RPPIs). Technical report, Eurostat.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*, 281–305.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Breiman, L., & Spector, P. (1992). Submodel selection and evaluation in regression. The x-random case. *International Statistical Review/Revue Internationale de Statistique*, pp. 291–319.

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

Court, A. (1939). Hedonic price indexes with automotive examples.

Deng, Y., & Quigley, J.M. (2008). Index revision, house price risk, and the market for house price derivatives. *The Journal of Real Estate Finance and Economics*, *37*(3), 191–209.

Do, A.Q., & Grudnitski, G. (1992). A neural network approach to residential property appraisal. *The Real Estate Appraiser*, *58*(3), 38–45.

Drucker, H., Burges, C.J., Kaufman, L., Smola, A.J., & Vapnik, V. (1997). Support vector regression machines. In *Advances in neural information processing systems* (pp. 155–161).

Evans, A., James, H., & Collins, A. (1992). *Artificial Neural Networks: An Application to Residential Valuation in the UK*. University of Portsmouth, Department of Economics.

Francke, M., & Van de Minne, A. (2021a). Daily appraisal of commercial real estate a new mixed frequency approach. *Real Estate Economics*.

Francke, M., & Van de Minne, A. (2021b). Modeling unobserved heterogeneity in hedonic price models. *Real Estate Economics*, *49*(4), 1315–1339.

Francke, M.K., & Van de Minne, A. (2017). The hierarchical repeat sales model for granular commercial real estate and residential price indices. *The Journal of Real Estate Finance and Economics*, *55*(4), 511–532.

Geltner, D., Francke, M., Shimizu, C., Fenwick, D., & Baran, D. (2017). Commercial property price indicators: sources methods and issues. Technical report, Eurostat.

Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, *4*(1), 1–58.

Griliches, Z. (1961). Hedonic price indexes for automobiles: An econometric of quality change. In *The price statistics of the federal goverment*. NBER (pp. 173–196).

Gu, S., Kelly, B., & Xiu, D. (2018). Empirical asset pricing via machine learning. Technical report, National Bureau of Economic Research.

Guo, X., Zheng, S., Geltner, D., & Liu, H. (2014). A new approach for constructing home price indices: The pseudo repeat sales model and its application in china. *Journal of Housing Economics*, *25*, 20–38.

Haas, G.C. (1922a). *Sale prices as a basis for farmland appraisal*, vol. 9. University Farm.

Haas, G.C. (1922b). A statistical analysis of farm sales in blue earth county, minnesota, as a basis for farm land appraisal. Technical report.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction*. New York: Springer Science & Business Media.

Hill, R.J., & Melser, D. (2008). Hedonic imputation and the price index problem: an application to housing. *Economic Inquiry*, *46*(4), 593–609.

Kohavi, R., et al. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, (Vol. 14 pp. 1137–1145). Montreal, Canada.

Kok, N., Koponen, E.-L., & Martínez-Barbosa, C.A. (2017). Big data in real estate? From manual appraisal to automated valuation. *The Journal of Portfolio Management*, *43*(6), 202–211.

Lancaster, K.J. (1966). A new approach to consumer theory. *Journal of Political Economy*, *74*(2), 132–157.

Malpezzi, S. (2002). Hedonic pricing models: a selective and applied review. *Housing Economics and Public Policy*, pp. 67–89.

McGreal, S., Adair, A., McBurney, D., & Patterson, D. (1998). Neural networks: the prediction of residential values. *Journal of Property Valuation and Investment*, *16*(1), 57–70.

McMillen, D.P., & Dombrow, J. (2001). A flexible fourier approach to repeat sales price indexes. *Real Estate Economics*, *29*(2), 207–225.

Nghiep, N., & Al, C. (2001). Predicting housing value: A comparison of multiple regression analysis and artificial neural networks. *Journal of Real Estate Research*, *22*(3), 313–336.

Nowak, A.D., & Smith, P.S. (2020). Quality-adjusted house price indexes. *American Economic Review: Insights*, *2*(3), 339–56.

Peterson, S., & Flanagan, A. (2009). Neural network hedonic pricing models in mass real estate appraisal. *Journal of Real Estate Research*, *31*(2), 147–164.

Quinlan, J.R., et al. (1992). Learning with continuous classes. In *5th Australian joint conference on artificial intelligence*, (Vol. 92 pp. 343–348). World Scientific.

Ripley, B.D. (1996). *Pattern recognition and neural networks*. Cambridge: Cambridge University Press.

Rosen, S. (1974). Hedonic prices and implicit markets: product differentiation in pure competition. *Journal of Political Economy*, *82*(1), 34–55.

Scholköpf, B., & Smola, A.J. (2002). *Learning with Kernels*. Cambridge: MIT Press.

Smola, A.J., & Scholköpf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, *14*(3), 199–222.

Steurer, M., Hill, R., & et al. (2019). Metrics for measuring the performance of machine learning prediction models: An application to the housing market Technical report, University of Graz, Department of Economics.

Tay, D.P., & Ho, D.K. (1992). Artificial intelligence and the mass appraisal of residential apartments. *Journal of Property Valuation and Investment*, *10*(2), 525–540.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58* (1), 267–288.

Varian, H.R. (2014). Big data: New tricks for econometrics. *Journal of Economic Perspectives*, *28*(2), 3–28.

Wallace, H.A. (1926). Comparative farmland values in iowa. *Journal of Land and Public Utility Economics*, 385–392.

Webb, G.I. (2000). Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, *40*(2), 159–196.

Wong, K., So, A.T., & Hung, Y. (2002). Neural network vs. hedonic price model: Appraisal of high-density condominiums. In *Real estate valuation theory* (pp. 181–198). Springer.

Worzala, E., Lenk, M., & Silva, A. (1995). An exploration of neural networks and its application to real estate valuation. *Journal of Real Estate Research*, *10*(2), 185–201.