



UvA-DARE (Digital Academic Repository)

Combining machine learning and econometrics: Application to commercial real estate prices

Francke, M.; van de Minne, Alex

DOI

[10.1111/1540-6229.12483](https://doi.org/10.1111/1540-6229.12483)

Publication date

2024

Document Version

Final published version

Published in

Real Estate Economics

License

Article 25fa Dutch Copyright Act (<https://www.openaccess.nl/en/policies/open-access-in-dutch-copyright-law-taverne-amendment>)

[Link to publication](#)

Citation for published version (APA):

Francke, M., & van de Minne, A. (2024). Combining machine learning and econometrics: Application to commercial real estate prices. *Real Estate Economics*, 52(5), 1308-1339. <https://doi.org/10.1111/1540-6229.12483>

General rights

It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations

If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: <https://uba.uva.nl/en/contact>, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (<https://dare.uva.nl>)

Combining machine learning and econometrics: Application to commercial real estate prices

Marc Francke¹  | Alex van de Minne^{2,3} 

¹Finance Group, Amsterdam Business School, University of Amsterdam, Amsterdam, Netherlands

²School of Business, University of Connecticut, Storrs, Connecticut, USA

³R&D Labs, Ortec Finance, Rotterdam, Netherlands

Correspondence

Alex van de Minne, School of Business, University of Connecticut, Storrs, CT 06269.

Email: avdminne@uconn.edu

Abstract

In this article, we combine a random effects model with different machine learning algorithms via an iterative process when predicting commercial real estate asset values. Using both random effects and machine learning allows us to combine the strengths of both approaches. The random effects will be used to estimate a common trend, property type trends, location value, and property random effects for properties that sold more than once. The machine learning algorithm will fit the observed characteristics (features) in a complex nonlinear fashion. The model is applied to a small sample of 2652 transactions in Phoenix (AZ) between 2001 and 2021. We only observe a limited number of property characteristics. The average out-of-sample MAPE is below 11%, which is as good or even better compared to the average appraisal error found in literature. The out-of-sample MAPE is even 9% for properties that sold more than once in the training set. In addition, our model provides indexes and locational heatmaps. These have their own uses and cannot be obtained with standard machine learning algorithms.

KEYWORDS

MCMC, neural network, prediction, structural time series

1 | INTRODUCTION

This article focuses on the estimation (out-of-sample prediction) of property prices in the commercial real estate (CRE) asset market. We do so by combining two cultures of statistical modeling (Breiman, 2001): Statistical models using stochastic data models (econometric models, like linear regression) and algorithmic models (machine learning). Machine Learning (ML) algorithms have become easier to use due to recent developments in both soft- and hardware¹ and can describe complex nonlinear relations including interaction effects between explanatory variables. Unlike standard ML algorithms, econometric models can explicitly control for spatial and temporal correlations and unobserved heterogeneity.

The main contribution of the article is providing ways to combine the two approaches, either in one go or in an iterative way, depending on the ML algorithm used. This allows us to combine the strengths of both ML and econometrics. In our application of CRE price modeling, this means that we can use a flexible and nonlinear function for property characteristics (ML part) while obtaining estimates of location values (heatmaps) and price indexes (econometric part).

ML algorithms have become popular and helpful tools for classification, clustering, pattern recognition, and prediction in many different disciplines (Abiodun et al., 2018), including real estate (Deppner & Cajias, 2022; Deppner et al., 2023; Kok et al., 2017; Lorenz et al., 2023; Peterson & Flanagan, 2009). One of the key differences between supervised ML algorithms (which predict one or more target variables, for example, a transaction price) with econometric models, lies in the model structure. Econometric models require a priori assumptions on the Data Generating Process (DGP), formulated by the econometrician: The specification of the appropriate functional form, including transformations of variables and potential interaction effects, and distributional assumptions on the error term. Conditional on the DGP one can obtain, among others, estimates and credible intervals of the unknown model parameters. Moreover, it is possible to test the validity of the assumptions made afterward. Unlike econometric models, most ML algorithms like neural networks, support vector machines, and regression trees determine the model's structure and parameter values simultaneously (Athey, 2018), avoiding the specification of the DGP. They are less rigid compared to standard econometric models like linear regression, providing nonlinear relations and interaction effects between variables. The goal of most ML algorithms is the “best” out-of-sample prediction accuracy, whereas the focus of econometric values is mostly on (causal) inference on the unknown model parameters (Mullainathan & Spiess, 2017). One of the main advantages is that ML algorithms can easily outperform simple econometric models in terms of out-of-sample prediction precision. Not surprisingly, ML algorithms have been proposed for the appraisal of residential asset and rental values (see among others, Chiarazzo et al., 2014; Deppner & Cajias, 2022; Nghiep & Al, 2001; Worzala et al., 1995).

However, ML algorithms have some serious drawbacks. It is well known that they are hungry for data, both in terms of number of observations and features, and in many applications, these are not available. Moreover, most algorithms implicitly assume identically and independently distributed error terms (Varian, 2014). Spatial and temporal data, for example, violate this assumption (Pace & Hayunga, 2020). Finally, ML algorithms lack explainability of the predictions

¹ As a point of reference, the “caret” package in R comes with 238 different preinstalled ML tools, which one can run within the same environment; see: <https://topepo.github.io/caret/available-models.html>. (Consulted on Nov. 15, 2023.) The package is optimized to run the code on parallel (CPU) cores. New developments in Python (Pandas) even allow ML software to run on parallel GPU cores, decreasing computation time even further, without specialized knowledge in CUDA.

(Mullainathan & Spiess, 2017) due to their complex structure that varies for each repeated calibration of the algorithm. Despite the substantial body of literature on interpretable ML (Arrieta et al., 2020; Lorenz et al., 2023; Molnar, 2019), there is still a large gap with econometrics in terms of complexity and standardization of methods.

CRE and other illiquid markets, like smaller housing markets, are characterized by a small number of sales. To compensate for these low counts, long sample periods are often used. Moreover, in many practical applications, only a limited number of observed characteristics is available, so there is considerable unobserved heterogeneity for the econometrician. Finally, real estate is characterized by the importance of spatial location (Francke & Van de Minne, 2021). Because of the small number of observations and characteristics, the unobserved heterogeneity, and the temporal and spatial dependence of the data, there are few applications of ML in CRE property price (index) models. Exceptions include Calainho et al. (2022), Kok et al. (2017), and Deppner et al. (2023). Kok et al. (2017) develop an automated valuation model focusing only on the largest metro area in their data (Los Angeles) and employing a small sample period, from 2011 to 2016. Moreover, they manually collect a large set of additional variables about local amenities to deal with the spatial dependence. Consequently, it is difficult to generalize such a methodology to other cities or illiquid markets. Calainho et al. (2022) use ML algorithms for the construction of CRE property price indexes. They calibrate ML algorithms period-by-period and calculate price indexes in a Paasche setup. However, this approach requires a large amount of observations per period. Finally, Deppner et al. (2023) use ML algorithms to calibrate appraisal errors, the log difference between sale price and appraisal value per square foot, for different CRE property types, using feature importance techniques to analyze which factors contribute to appraisal errors. They use properties included in the NCREIF Property Index between 1997 and 2021.

In this article, we combine machine learning and econometrics to model CRE property prices, and construct price indexes and heat maps. The joint model for sale prices consists of five components: (1) a common trend; (2) a property type trend; (3) a spatial component; (4) a property random effect for unobserved heterogeneity; and (5) a property characteristics component. We assume a DGP for the first four components. When the fifth component is modeled by, say, a linear function, one has a fully specified, econometric model. This model is closely related to the one in Francke and Van de Minne (2021). The common and property type trends are specified as random walks; however, more complex trend structures can be easily incorporated. From the common and property type trends, one easily obtains constant quality price indexes (conditional on the observed property characteristics). The spatial component is modeled by a Besag model (Besag & Kooperberg, 1995). From these spatial random effects, one can derive “constant quality” heat maps. Note that the Besag random effects can be replaced by an alternative spatial model. The fourth component represents unobserved heterogeneity specified by property random effects. These are in specific useful, in terms of out-of-sample prediction accuracy, for properties that are sold more than once (Francke & Van de Minne, 2021).

In our mixed econometric machine learning approach, we replace the linear function for the property characteristics component (the fifth component) by a neural network, allowing for non-linear relations and interaction effects. One advantage of using a parametric ML algorithm such as a neural network is that the joint model can still be estimated using for example Bayesian methods, so parameter estimates, and credible intervals are readily available. This is not possible for nonparametric ML algorithms, especially tree-based ones.² Therefore, we also apply an iterative approach, inspired by Ceyhan (2017). First, we estimate the joint model with a linear function

²Regression trees and random forests are arguably the most famous ones that fall within this category.

for the property characteristics component. Second, we take the ML “residual” as the observed value minus the estimates of components 1 to 4 and calibrate a ML algorithm. Third, we take the DGP “residual” as the observed value minus the prediction of component 5, and estimate the econometric model, consisting of components 1 to 4. We repeat these steps until convergence occurs.

To illustrate our newly proposed methodology, we utilize data provided to us by MSCI/Real Capital Analytics. The data contains transaction prices of 2600 CRE properties between 2001 and 2021 in Phoenix, AZ. Other than transaction prices, we also observe the net operating income (in the year leading up to the sale), the property size, quarter of sale, the age of the building, its walk score, the exact location (latitude/longitude), and a unique property identifier. We run (1) a basic neural network and (2) a random effects hedonic price model, inspired by Francke and Van de Minne (2021). Both these models provide us with a baseline to compare the performance of our “mixed” models with. Subsequently we run (3) a fully specified Bayesian model which combines the neural network with the random effects hedonic price model in one, and finally (4) the proposed iterative approach. We perceive this final model as our main contribution to existing literature. The full Bayesian estimation was approximately 400 times slower during our test and cannot be generalized to (most) other machine learning algorithms. To stress this latter point we therefore also replace the neural network in our algorithmic approach with a Support Vector Machine, Gradient Boosting Machine, and a Regression Tree, as a robustness test.

We employ 10 *K*-folding to measure the out-of-sample fit. We find that the neural network has an out-of-sample mean absolute prediction error (MAPE) of 14% compared to a MAPE of 15% for the random effects hedonic price model. The combined model has a MAPE of 11%, an improvement on the neural network and the random effects hedonic price model by 22% and 26%, respectively. For properties that were sold more than once in the training sample, the MAPE reduces further to just 9%, which is below the average appraisal error in the United States found in previous literature (Cannon & Cole, 2011; Fisher et al., 1999). Moreover, if we omit key variables (net operating income and walk score), the MAPE does not deteriorate much for properties that were sold more than once. The differences in results between the full Bayesian estimation and our proposed iterative approach are negligible. Both have an out-of-sample MAPE of 11%, and the correlation between the fitted values is 1.00. Adding spatiotemporal features to a basic neural network helps improve its fit, but our mixed model approach is still superior by far. Using other ML algorithms does not change the fit erratically and can therefore be used as well. These results are very encouraging, as we only lay out the basic principle of mixing machine learning with econometric models. We are not optimizing the fit per se, and future research should be able to improve on this quite considerably.

As noted earlier, other than improving the out-of-sample fit of ML algorithms, our proposed methodology can be used to compile price indexes and heatmaps of location values. Indexes have their own use—separate from appraising property values—as they can be used as input in forecasting models of real estate returns (a crucial input for pro formas, Van de Minne et al., 2022), giving a sense of what the current mood is in the real estate market, and as a future tool for derivative trading (Deng & Quigley, 2008). Even though it is hard to measure the “fit” of an index (Guo et al., 2014; Van de Minne et al., 2020), the estimated indexes look very promising and do not deviate much from the indexes estimated from the random effects hedonic price model.

This article provides a straightforward framework to combine machine learning algorithms with more standard econometric models, while keeping the strengths of both. As such, this article fits in the literature of automated valuation models in CRE (Kok et al., 2017), although it can easily

be extended to other types of illiquid assets. Given that our method produces indexes, our research also relates to the small field that studies real estate indexing using ML (Calainho et al., 2022). The article also fits in the growing field of Bayesian(-like) model estimation for real estate, where fixed effects are replaced by random effects (Bollerslev et al., 2016; Francke, 2010). These models have the benefit of being able to fit quite well on smaller and/or higher frequency samples, a weakness of ML algorithms in general. Finally, this article adds to the more general discussion on which type of statistical modeling is preferred, and when: Stochastic data models versus algorithmic modeling (Breiman, 2001).

This article is structured as follows. We start Section 2 with describing the standard neural network, and the random effects hedonic price model, which will both be used a baseline for the other models. After this, the full Bayesian estimation and the iterative approach is discussed. In Section 3, we give a description of the data, followed by the results in Section 4. Finally, Section 6 concludes.

2 | METHODOLOGY

2.1 | Baseline neural network (NN)

The term “neural network” has its origins in attempts to find mathematical representations of information processing biological systems (McCulloch & Pitts, 1943; Rumelhart et al., 1986; Widrow & Hoff, 1960). Even though the biological realism is debatable (Bishop & Nasrabadi, 2006), neural networks have been applied successfully to a variety of different disciplines. Abiodun et al. (2018) survey many applications of neural network techniques in various disciplines which include computing, science, engineering, medicine, environmental, agriculture, mining, technology, climate, business, arts, and nanotechnology. Throughout this article, we employ a single-layer feed-forward network. Such models are represented by the following equations:

$$y_i = f^{NN}(x_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2), \quad (1a)$$

$$f^{NN}(x_i) = \lambda_0 + \sum_{m=1}^M f(a_{m,i}) \lambda_m, \quad (1b)$$

$$a_{m,i} = \sum_{k=1}^K \omega_{k,m} x_{i,k} + \omega_{0,m}, \quad (1c)$$

$$f(a_{m,i}) = (1 + \exp(-a_{m,i}))^{-1}, \quad (1d)$$

where y_i is the dependent variable (log sale price) for individual transaction $i = 1, \dots, N$. The error term ϵ is assumed to be independently normally distributed with mean zero and standard deviation σ_ϵ . The functional form of the neural network is given by Equation (1b). We have M hidden nodes $f(a_{m,i})$ in our single layer, with corresponding coefficients λ_m , and λ_0 is a constant. The quantities $a_{m,i}$ in Equation (1c), known as activations, are a linear function of the explanatory variables $x_{i,k}$ with weights $\omega_{k,m}$, and a bias $\omega_{0,m}$. The activations are transformed by a differentiable, nonlinear activation function, in our case a sigmoid; see Equation (1d), with values between 0 and 1. Such models can be calibrated by a variety of methods. For our baseline neural network,



we use an iterative backpropagation algorithm, as detailed in Bishop and Nasrabadi (2006), which is readily available in R (Günther Fritsch, 2010) and other software packages.

A few observations are in order here. First, because of the sigmoid transformation, it is recommended to normalize or scale the variables. If a variable has a large (positive or negative) value, it decreases the variability in f . In our article, we log transform our variables and subtract the mean of the log of said variable. This makes it easier to interpret the results (compared to normalizing), especially when estimating indexes. Second, how many nodes M should one use? In principle, one can try multiple nodes, and compare the in-sample MAPE. In our case, due to the large amount of models we already have to estimate, this is not feasible. Instead, we use the rule-of-thumb, that the number of nodes should be approximately $M = K \times 2 + 1$. We also limit ourselves to one hidden layer only. Finally, the issue of multimodality persists, even though the system is identified (Yao et al., 2018). In practice, this means that you can achieve different parameter estimates, depending on the starting values. We solve this by rerunning the model multiple times (seven in our case)³ with different starting values and picking the starting values which resulted in the lowest (in-sample) MAPE.

2.2 | Random effects model (REM)

Our random effects model is based on Francke and Van de Minne (2021) (FM henceforward). The FM is a hedonic price model with several components, including (1) a trend component, (2) a spatial component, and (3) property specific random effects. We also include sub trends for property types, as was done earlier for residential real estate (Francke & Vos, 2004) in a hedonic price model and for CRE in a repeat sales model by Francke and van de Minne (2017).

The model is given by

$$y_{itp} = x'_{itp} \beta + \mu_t + \delta_{tp} + \theta_i + \phi_i + \epsilon_i, \quad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2), \quad (2a)$$

$$\mu_t \sim \mathcal{N}(\mu_{t-1}, \sigma_\mu^2), \quad u_{t=1} = 0, \quad (2b)$$

$$\delta_{t,p} \sim \mathcal{N}(\delta_{t-1,p}, \sigma_\delta^2), \quad \delta_{t=1,p} = 0, \quad (2c)$$

$$\theta_i | \theta_{-i}, \sigma_\theta^2 \sim \mathcal{N}\left(\frac{1}{m_i} \sum_{q \in \Omega_i} w_{iq} \theta_q, \frac{\sigma_\theta^2}{m_i}\right), \quad \sum_{i=1}^I \theta_i = 0, \quad (2d)$$

$$\phi_i \sim \mathcal{N}(0, \sigma_{\epsilon_\phi}^2), \quad (2e)$$

where y are the log sale prices of individual transactions i of property type p in quarter t (in deviation from the mean to keep it comparable with the baseline neural network). The covariates are given by x with corresponding parameter vector β and include among others a constant and dummy variables per property type. The error term ϵ is assumed to be independently normally distributed with mean zero and standard deviation σ_ϵ .

³We ran this model on a laptop with 8 cores. Hence, we run seven neural networks with different starting values on 8 cores in parallel. Having more starting values can increase the fit even further. We fixed the seed (12345) to ensure reproducibility.

The common trend is provided by μ_t , specified as a random walk; see Equation (2b). The property type subtrends, in deviation from the common trend, are also specified as random walks; see Equation (2c). We assume that the standard deviation parameter σ_δ is shared by all property type trends. Note that $\mu_{t=1} = 0$ and $\delta_{t=1,k} = 0$ for identification purposes, as a constant and property fixed effects have already been included in the covariates x .

The spatial random effects component θ is specified as an improper intrinsic conditional autoregressive (ICAR) model (Besag, 1974; Besag & Kooperberg, 1995; Besag et al., 1991): A set of conditional distributions of one spatial unit θ_i given all other units θ_{-i} ; see Equation (2d). Let w_{iq} denote a symmetric proximity measure for properties i and q . It is nonnegative when $i \neq q$ and 0 otherwise. In our application, we use $w_{iq} = 1$ if the distance between the properties is smaller than a predefined threshold, and 0 otherwise. Let Ω_i denote all m_i neighbors of property i , all properties q for which it holds that $w_{iq} \neq 0$. As a result, the expected value of the spatial random effect θ_i will be the average of the spatial random effect of property i 's neighboring properties.⁴ The precision is higher if there are more neighboring properties. We impose that the sum of the spatial random effects has to equal zero for identification purposes: $\sum_{i=1}^I \theta_i = 0$.

Finally, individual property random effects ϕ are given by Equation (2e). Conditional on the standard deviations ($\sigma_\epsilon, \sigma_\phi$), it is possible to estimate (ϵ, ϕ) , also for properties that have only been sold once. In order to estimate $(\sigma_\epsilon, \sigma_\phi)$ in absence of prior information, it is necessary to have some repeat sales for some properties (multiple sales of the same property). Francke and Van de Minne (2021) show that the out-of-sample prediction accuracy is much higher for properties that have at least one sale in the training set. We estimate Equations (2a)–(2e) with Integrated Nested Laplace Approximation (INLA) due to its computation speed (Rue et al., 2009).

2.3 | Full Bayesian representation of the neural network and random effects model (fNNREM)

In this section, we combine the random effects model with a neural network. To do so, in Equation (2a), we replace the linear term $x'_{itp} \beta$ by the neural network by the neural network $f^{NN}(x_{itp})$, giving:

$$y_{itp} = f^{NN}(x_{itp}) + \mu_t + \delta_{tp} + \theta_i + \phi_i + \epsilon_{itp}, \quad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2). \quad (3)$$

The components $(\mu, \delta, \theta, \phi)$ are given by Equations (2b)–(2e). The specification of the neural net component $f^{NN}(x_{itp})$ is given by Equations (1b)–(1d). Because model (3) is a mix of an algorithmic (neural network) and a statistical (random effects model) model, it cannot be estimated by back-propagation (for neural networks) or INLA (for models that can be expressed as Gaussian Markov Random Fields). Instead, we will use the No-U-Turn-Sampler (Hoffman & Gelman, 2014), with mostly uninformative priors (Gelman, 2006).

For identification purposes, we impose the following two constraints on our hidden nodes. First, we have an ordering constraint on our biases in Equation (1c), where $\omega_{0,1} < \omega_{0,2} \dots < \omega_{0,K}$. Our second constraint is that the sum of the coefficients *within* a node (excluding the bias) has to equal zero. As with the baseline neural network, the issue of multimodality still persists even with

⁴ We chose a distance of 800 m as a threshold. In other words, if a property is within 800 m of another property (about half a mile) they are considered “neighbors.” With this distance, at least 90% of the observations have at least 1 “neighbor.” This is inline with Francke and Van de Minne (2021).

such constraints; see Yao et al. (2018). We therefore use the same principle as with our baseline neural network and estimate the model seven times with different starting values and pick the one with the lowest in-sample MAPE.⁵

To ensure that the code is efficient and as fast as possible, we employed multiple “tricks.” Most of these have to do with variable transformations and vectorization of the code. However, it is out of the scope of this article to detail how we exactly achieved this. For full transparency, we therefore provide the Stan code in the Appendix A.⁶

2.4 | Iterative approach combining ML and random effects model (iNNREM)

Combining a fully specified stochastic data model, as in Equation (2), with a ML algorithm is generally not possible. Most ML algorithms cannot be formulated in an explicit functional form, as is the case for a neural network, and therefore cannot be estimated by likelihood-based methods. And if one can, estimating such a combined model requires a lot of computing power.⁷ We therefore propose the following iterative estimation approach as specified in algorithm 1.

Algorithm 1 Iterative approach for ML algorithm with random effects models (iNNREM)

- 0. Initialize:** Estimate REM (Equation 2) including linear term $x'_{itp}\beta$ using INLA:
 - Compute model prediction and corresponding MAPE⁽⁰⁾;
 - Set MAPE⁽⁻¹⁾ to a large number;
while | MAPE^(j) – MAPE^(j-1) | < ε **do**
- 1. Train:** ML algorithm on $\{y_{itp} - \widehat{\mu}_t - \widehat{\delta}_{itp} - \widehat{\theta}_i - \widehat{\phi}_i, x_{itp}\}$, where the hats indicate estimates from REM. This gives the predictions from the ML algorithm: $\widehat{f}(x_{itp})$;
 - Compute $y_{itp}^* = y_{itp} - \widehat{f}(x_{itp})$;
- 2. Estimate:** REM (Equation 2) without linear term $x'_{itp}\beta$ with dependent variable y_{itp}^* by INLA;
 - Compute model predictions and corresponding MAPE^(j);
end while
-

We first initialize the model using our baseline random effect model as provided by Equation (2), which is computationally efficient estimated by INLA. Subsequently, we compute its MAPE (Step 0). Next, we adjust the prices by subtracting the estimates from the random effects model, apart from the linear term $x'_{itp}\beta$. This is equal to the residual plus $x'_{itp}\hat{\beta}$. We can now use any

⁵ It should be noted here that Yao et al. (2018) propose a different technique. They propose to estimate the Bayesian neural network with different starting values (similar as before), but subsequently weigh the estimated parameters by the log density of the posteriors. However, we use the more straightforward method of picking one chain based on the in-sample fit as it is more comparable with the approach used for the baseline Neural Network.

⁶ Stan is a compiler that can run such Bayesian code in R or Python; see Hoffman and Gelman (2014).

⁷ It is for example possible to estimate decision trees using a Gibbs sampler using Dirichlet distributions. This will require even *more* computing power (Hoffman & Gelman, 2014). Even in our application, it took a week to estimate the full Bayesian model described in Section 2.3, whereas the proposed iterative procedure in this section takes about 30 minutes. To be more precise, it takes approximately 20 hours to estimate onefold within Stan, and we estimate 10-folds; see Section 2.5.

ML algorithm to increase the fit of the model using the adjusted prices and the covariates in x (Step 1). For the main results, we will focus on neural networks, as provided by Equation (1), but in the robustness section, we will also experiment with other ML algorithms. Subsequently, we subtract the predicted values by the ML algorithm from the transaction prices, giving y^* . Now we can re-estimate (using INLA) the random effects model, this time without the linear term $x'_{itp}\beta$, on the dependent variable y^* (Step 2). We compute its MAPE, and until the change in the MAPE is smaller than some threshold ε , we repeat Steps 1 and 2. In this article, we set $\varepsilon = 0.001$.

Note that one can use another fit measure as convergence statistic, like Root Mean Square Error (RMSE). Further note that it is not required that the MAPE improves every step, $\text{MAPE}^{(j)} < \text{MAPE}^{(j-1)}$. It is only required that the absolute difference is smaller than a threshold. Theoretically, there is no guarantee for this highly nonlinear model that this iterative method leads to a global optimum (but that also applies to the standard calibration of ML algorithms). So, it may happen that a local rather than a global optimum is found. In other words, there could in theory be even better performing models than the ones found. We do not consider this a problem. The iterative approach already leads to a substantial improvement over both the baseline neural network and the random effects model. Moreover, in order to increase the likelihood of finding a global optimum, we run the algorithms with different starting values. Finally, we do compare our results from the iterative approach to the ones from the full Bayesian estimation of the neural network and random effects model, and see that results are very close to each other.

2.5 | K-folding and index range over the folds

In order to measure out-of-sample fit we employ 10 K -folding. More specifically, we randomly select 10% of the data and omit these observations when training the model. The omitted data is also referred to as the test data. The 90% of the data we keep for training the models is typically referred to as the training data. The estimated parameters from the training set are subsequently used to predict the (log) prices of the test data. The predicted values from the test set are used to calculate the out-of-sample MAPE and is reported throughout this article. We redo this analysis 10 times, where the test set is unique at each draw. As such, we get an out-of-sample MAPE for every transaction in the data exactly once. The out-of-sample MAPE will give us a good sense of how well the model can predict values of properties not (yet) sold.

Second, we want to measure the quality of the estimated indexes. Extant literature has established that testing for index “fit” is hard, if not impossible (Clapham et al., 2006; Clapp & Giaccotto, 1999; Guo et al., 2014; Van de Minne et al., 2020). The core of the issue being that we do not know the “true” index, and as such cannot calculate residuals and subsequent fit.

To still get a sense of the quality of the index, we look at the range of estimated index values ($\widehat{i}_{tp} = \widehat{\mu}_t + \widehat{\delta}_{tp}$) over the 10-folds. More specifically, for every period t we first take the range in (log) index per property type—or: $\max(\widehat{i}_{tp}) - \min(\widehat{i}_{tp})$ —and subsequently take an average over all t . This exercise gives us an idea of how stable the estimates are. Indexes are at their most useful if they do not revise whenever new data is added (Van de Minne et al., 2020). Economic models of mortgage prepayment and default, measures of (national) wealth, the total value of collateral behind a portfolio of mortgage loans, real estate derivatives, among many other applications, are all informed by real estate indexes subject to unwanted, but substantial revisions.

TABLE 1 Descriptive statistics of our main variables.

Statistic	N	Mean	St. Dev.	Min	Max
Sales price	2652	\$17,702,190	\$21,448,988	\$787,000	\$280,000,000
NOI (p. sqft)	2652	\$11.309	\$8.837	\$1.239	\$115.914
Building size (sqft)	2652	129,127	132,461	2000	1,366,600
Age of building	2652	19.627	13.572	1	78
Walk score	2652	47.006	16.470	0	96
Property type count	All	Apartment	Industrial	Office	Retail
	2652	1174	263	484	731

Note: NOI: net operating income. Walk score is a number from 0 to 100, where 0 means not walkable, and 100 denotes very walkable.

3 | DATA

Our data is provided to us by MSCI/Real Capital Analytics and captures approximately 90% of all commercial property transactions in the United States over \$2.5 million. The data provider has at least two independent sources confirming the transaction prices, resulting in a high-quality database. In total, our database contains 2652 prefiltered transactions for Phoenix (AZ) for which we have all necessary explanatory variables, for the period 2001–2021. As a result, we observe 130 transactions per year on average. Other than transaction prices and the quarter of sale, our data contains the net operating income (NOI) per square foot of building in the year prior to the sale,⁸ property type (apartment/industrial/office/retail), the age and size of the structure (in square feet), latitude and longitude, a unique property identifier, and finally the walk score (with zero being not walkable, and 100 being very walkable). The average transaction price is about \$17.7 million; the average net operating income per square foot is approximately \$11. Combined with an average structure size of about 130,000 square feet, this results in a total yearly net operating income of almost \$1.5 M at the time of sale. The average walk score is 47, and the average age is 20 years. Most properties are designated apartment (44% of transactions), followed by retail (27%), office (18%), and finally industrial (10%) properties (Table 1).

Figure 1 gives a map of the city of Phoenix and the location of our transactions (red dots). We just focus on the city on the map for readability, but our transaction data contains the entirety of the metro area. As such, Figure 1 only represents approximately 70% of our transactions. Phoenix (AZ) is a unique city, as there is no clear center. Indeed, the black square (representing highways) in the middle of Figure 1 is typically seen as the central business district, however, relatively little transactions took place there. Phoenix has been characterized by rapid population growth, partly fueled by little space and regulatory constraints for new development (Saiz, 2010). The little supply constraints means that new development at the boundary of the city is often cheaper than increasing the city's density whenever there is a demand shock. As a result, Phoenix's commercial real estate is more "spread out," akin to cities like Houston and Dallas.⁹

⁸ Net operating income is defined as the total rental income and other type of income of the property, minus operating expenses. Operating expenses include property tax, utilities, regular maintenance, and management fees.

⁹ Interestingly, in June 2023, Arizona determined that there is not enough groundwater for all of the development that has already been approved in the Phoenix area and will stop developers from building some new subdivisions. This is likely a start of more stringent supply constraints going forward in Phoenix. However, our data ends before this pivotal decision

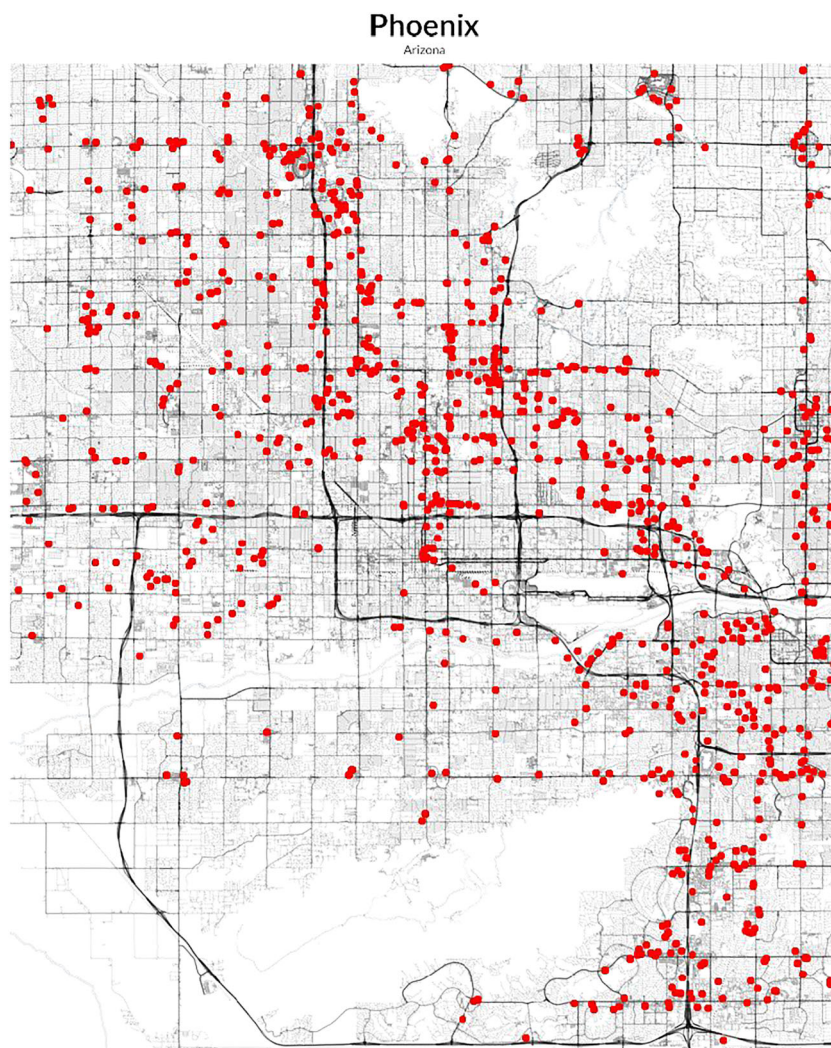


FIGURE 1 Every red dot represents a property sold in our data. There are also some transactions outside of the city, but within the MSA, which we omitted in this Figure to conserve space and to make it more readable. [Color figure can be viewed at [wileyonlinelibrary.com](https://onlinelibrary.wiley.com/doi/10.1111/1540-6229.12483)]

4 | RESULTS

4.1 | Cross-sectional fit

Table 2 gives the main results for Phoenix. We provide the out-of-sample and in-sample statistics (MAPE) for all our different specifications. The results for the index revisions can be found in Figure 4.

was made by the state. See <https://www.nytimes.com/2023/06/01/climate/arizona-phoenix-permits-housing-water.html> for more details.

TABLE 2 Main results for Phoenix.

	(1)	(2)	(3)	(4)
	NN	REM	fNNREM	iNNREM
MAPE, out of sample, all	0.139	0.147	0.111	0.109
MAPE, out of sample, $k = 0$	0.142	0.157	0.111	0.114
MAPE, out of sample, $k = 1$	0.136	0.138	0.113	0.106
MAPE, out of sample, $k > 1$	0.133	0.117	0.102	0.090
MAPE, in sample, all	0.127	0.095	0.104	0.079
Obs, all	2652	2652	2652	2652
Obs, $k = 0$	1685	1694	1694	1694
Obs, $k = 1$	659	647	647	647
Obs, $k > 1$	308	311	311	311

Note: NN = baseline neural network, as provided by Equations (1a)–(1d). REM = random effects model, as given by Equations (2a)–(2e). fNNREM = full Bayesian representation of a neural network with random effects; see Equation (3). iNNREM is an iterative approach of combining random effects with neural networks, as shown in algorithm 1. MAPE = mean absolute prediction error. Obs = number of observations. k gives the amount of times the (unique) property was sold in the training sample. We K -folded 10 times to get our out-of-sample statistics. The explanatory variables are net operating income (per square foot), walk score, size of property, age, industrial (Dummy), retail (Dummy), and office (Dummy). The dependent variable is the log sale price minus the average log sale price.

The standard neural network (NN) gives an average out-sample MAPE of 0.14, which is lower compared to the baseline random effects model (REM) with a MAPE of 0.15. However, note that the fit is equal between these two models for properties that were sold at least once in the training data ($k = 1$), and even better (albeit modestly) for the REM when the property was sold more than once ($k > 1$). As explained in Francke and Van de Minne (2021), this is mostly due to the property level random effects that effectively control for most unobserved heterogeneity.

Similarly, note that the in-sample MAPE is considerably lower compared to the out-of-sample MAPE for the REM model. (This is also the case for the iNNREM model described later.) More specifically, the average out-of-sample MAPE is 0.147. In contrast, the average in-sample MAPE is 0.095, or 50% lower. This “overfitting” is mostly caused by the property random effects. Albeit it is not completely fair to compare the out-of-sample and in-sample fit for such models. The reason being that the training set is 9 times larger compared to the test set and, therefore, has a larger probability of containing multiple sales of the same property. As explained earlier, the fit improves drastically when a property sold multiple times. Note that the averages for the in-sample statistics are provided over 90% of the data \times 10-folds. The out-of-sample statistics are given over 10% of the data \times 10 (nonoverlapping) folds.

Next, we turn our attention to the fit of the mixed models (fNNREM and iNNREM). Both the full Bayesian specification (fNNREM, column 3) and the iterative approach of mixing random effects with the neural network (iNNREM, column 4) give an out-of-sample MAPE of 0.11 on average, an approximate 20% decrease compared to the baseline models. Even with no similar property in the training set ($k = 0$) do both mixed models outperform the REM model with more than 1 similar property ($k > 1$) in the training set. A few things can be noted. First, the differences in fit between the full Bayesian specification and our iterative approach are negligible. This is further buttressed by plotting the residuals and fit in a scatterplot; see Figure 2a and b, respectively. The correlation between the residuals is 0.89 and between the fitted values 1.00. This gives us confidence that our iterative approach gives similar results compared to the full Bayesian

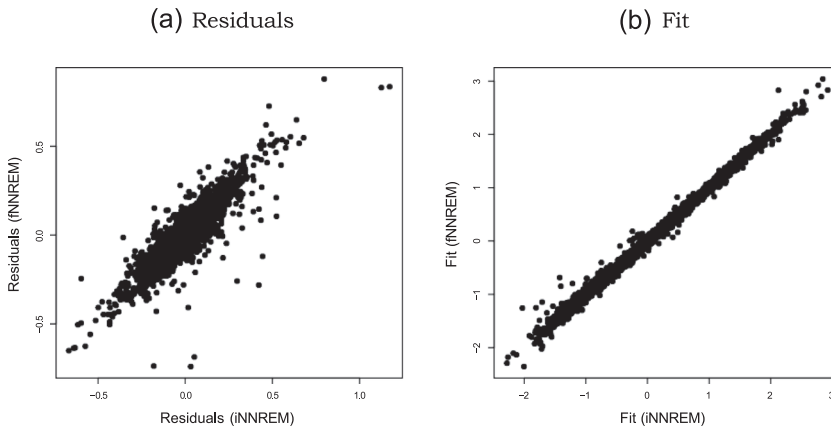


FIGURE 2 Scatterplot of residuals and fit between the full Bayesian model (fNNREM) and our iterative approach of mixing random effects models with neural networks (iNNREM).

specification. This is important as the iterative approach is faster to estimate and is easier to generalize (i.e., use other ML algorithms). As a result, we will focus on this iterative approach throughout this article, instead of providing similar statistics twice. Second, we find that the fit is close to or even below (for properties that sold more than once) 10%. Previous literature found that the average error of professional appraisers has been somewhere between 10% and 14%; see Fisher et al. (1999) and Cannon and Cole (2011). Given that databases mature over time—meaning more repeat sales—the overall model fit will improve even more in the future.

In Figure 3, we provide a “heatmap” of θ_i (Equation 2d), which represents the property specific locational premium/discount. The estimates are taken from the iNNREM model and are calculated by taking the average over all 10 K -folds. Our estimates show a clear pattern in clusterings of locational values. For example, we find large discounts in the Southwest part of the provided map, and the larger premia can be found in the Eastern part of downtown. The range is between approximately -10% and $+10\%$. This relatively low range on location premia has two causes. First, we control for net operating income. Thus, the discount/premia reflect cap rates, which is only one component of property values. Second, Phoenix is characterized by high supply elasticity (Saiz, 2010). Demand therefore more easily translates into new development, instead of higher prices.

4.2 | Indexes

In this section, we will discuss the resulting (log) price indexes per property type for the REM and the iNNREM. Note that the baseline neural network (NN) does not provide us with indexes, and we leave out the results of the full Bayesian model (fNNREM), given how similar it behaves to the iNNREM. These indexes are available upon request. We can plot an index and take the returns by taking the average values of the relevant index over the 10 K -folds. Subsequently, we take the range (max–min) per time period to get a sense of the stability of the estimated index. The indexes (black line) and subsequent ranges (blue area) are visualized in Figure 4. The average returns and range per property type and model is provided in Table 3.

The left column of Figure 4 gives the results for our baseline random effects model (REM), whereas the right column gives the results for our iterative approach of combining neural

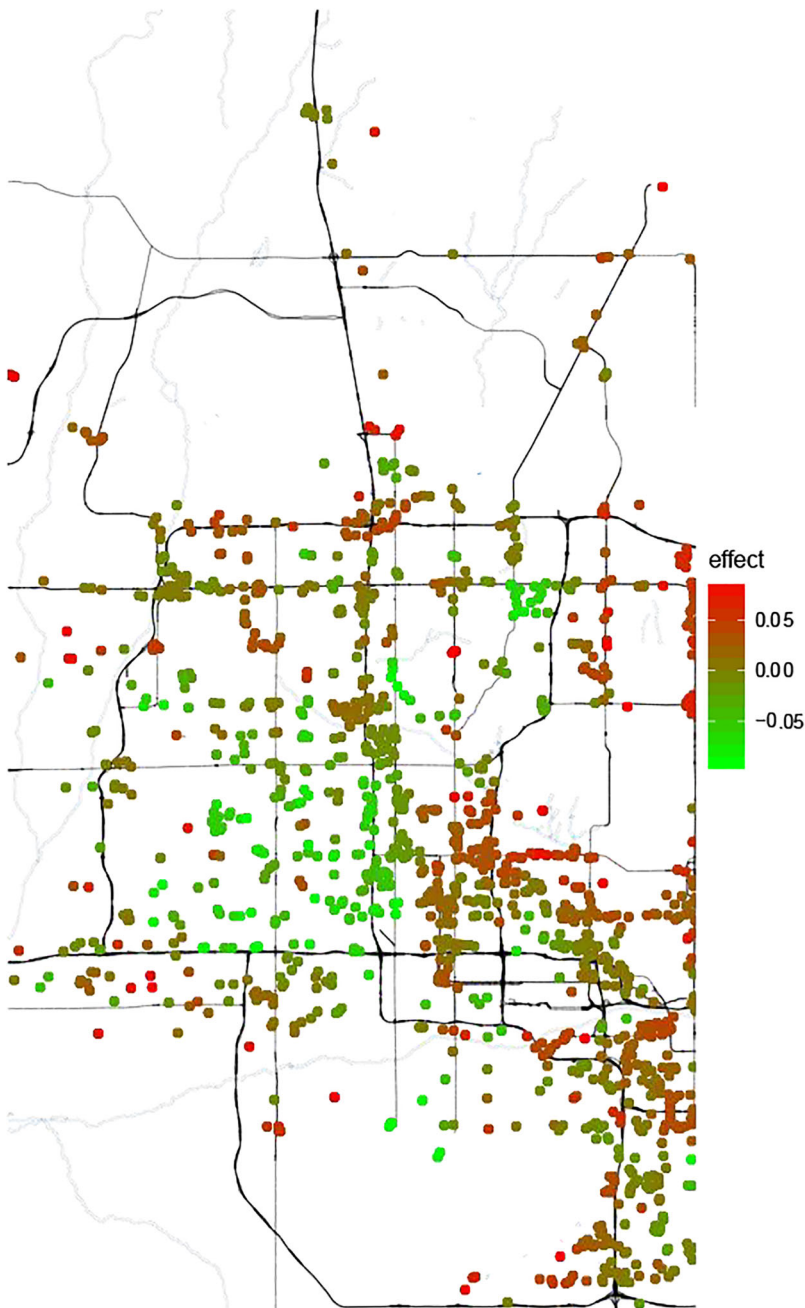


FIGURE 3 Heatmap. Every red dot represents a property sold in our data. Darker red (green) color represents higher (lower) value for our Besag spatial random effect (θ_i in Equation 2d). The estimates are taken from the iNNREM model and is the average over the 10 K -folds. [Color figure can be viewed at wileyonlinelibrary.com]

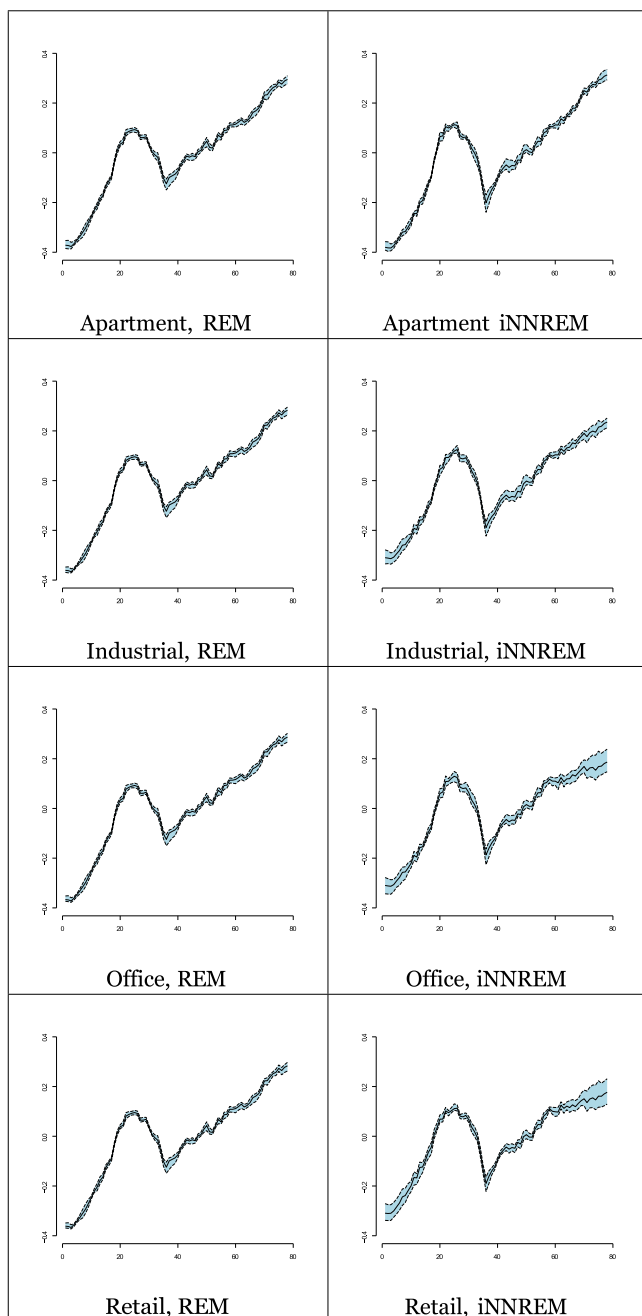


FIGURE 4 Estimated subtrends. REM: random effects model, iNNREM is our iterative approach of combining random effects models with neural networks. [Color figure can be viewed at wileyonlinelibrary.com]

networks with random effects models (iNNREM). The rows provide the results for the different property types: apartment, industrial, office, and retail. Some correlation between the indexes (in the returns) is expected. Especially because in our model we hold constant for the net operating income. Thus, the indexes represent something close to the changes in (the inverse of the) cap rates (Francke & Van de Minne, 2022). Cap rates are driven by investors that compete on a single capital market. For example, lower interest rates are expected to lower cap rates for all property types throughout the country.

**TABLE 3** Average index returns and range (in log levels) for Phoenix.

Property type	Returns		Range	
	(1) REM	(2) iNNREM	(3) REM	(4) iNNREM
Apartment	0.0346	0.0360	0.0235	0.0272
Industrial	0.0334	0.0283	0.0225	0.0337
Office	0.0339	0.0258	0.0236	0.0424
Retail	0.0335	0.0253	0.0230	0.0427

Note: REM = random effects model, as given by Equations (2a)–(2e). iNNREM is an iterative approach of combining random effects with neural networks, as shown in algorithm 1. The average returns are calculated by taking the average values over all the K -folds. We multiply the returns by 4 to get annualized returns. The range is computed by taking the max value for every period per property type and subtracting the minimum value. The reported numbers are the averages over these ranges per property type. The explanatory variables are: net operating income (per square foot), walk score, size of property, age, industrial (Dummy), retail (Dummy), and office (Dummy). The dependent variable is the log sale price minus the average log sale price.

However, the comovement between the property types for the REM specification is larger than expected. In fact, all property types grew between 3.3% (industrial) and 3.5% (apartment) annually between 2001 and 2021 (Table 3). This could be caused by the relative lack of observations, resulting in the variance parameter on the subtrends going toward zero. In contrast, the differences between the property type indexes are larger for the iNNREM model. More specifically, retail prices grew the least with an average of 2.5% per annum, whereas apartment prices increased the most with 3.6% per annum. On the downside, it is also obvious from the get-go that the differences per K -fold are larger for the iNNREM model compared to the baseline REM, meaning less stable estimates. As shown in Table 3, the range is almost twice the size for office and retail for the iNNREM model as compared to the REM model. Thus, on the one hand, our iterative model results in more variation in the indexes (which is more desirable), but also in less stable estimates (which is less desirable).

5 | ROBUSTNESS

5.1 | Results without net operating income and walk score

In Table 4, we present the same models, on the same dataset as in Section 4, but we purposefully omit key variables: (1) net operating income and (2) the walk score. Thus, we only have the following features: size of the building (in square footage), log of age of the building, and dummies for property type (industrial, retail, and office, where apartment is left out to avoid the dummy trap).

The idea behind this exercise is twofold. First, it shows that random effects models are good at explaining away unobserved heterogeneity and are less affected by dropping variables that have a spatial and/or time component. Second, these variables are difficult to get by in many other countries, asset types, or lower-level geographies. For example, MSCI/Real Capital Analytics (our data provider, Section 3) observes net operating income during the time of sale for only about a third of all transactions. Our data came prefiltered, and net operating income is an important variable. This is fine in a larger market like Phoenix but may result in too little data in secondary and tertiary cities. Other asset classes, like single family housing, will not have a net operating income variable as well.

TABLE 4 Results for Phoenix with reduced set of features.

	(1)	(2)	(3)	(4)
	NN	REM	fNNREM	iNNREM
MAPE, out of sample, all	0.313	0.234	0.163	0.148
MAPE, out of sample, $k = 0$	0.317	0.268	0.196	0.163
MAPE, out of sample, $k = 1$	0.306	0.193	0.143	0.132
MAPE, out of sample, $k > 1$	0.302	0.132	0.101	0.100
MAPE, in sample, all	0.292	0.083	0.066	0.060
Obs, all	2652	2652	2652	2652
Obs, $k = 0$	1685	1694	1694	1694
Obs, $k = 1$	659	647	647	647
Obs, $k > 1$	308	311	311	311

Note: NN = baseline neural network, as provided by Equations (1a)–(1d). REM = random effects model, as given by Equations (2a)–(2e). fNNREM = full Bayesian representation of a neural network with random effects; see Equation (3). iNNREM is an iterative approach of combining random effects with neural networks, as shown in algorithm 1. MAPE = mean absolute prediction error. Obs = number of observations. k gives the amount of times the (unique) property was sold in the training sample. We K -folded 10 times to get our out-of-sample statistics. The explanatory variables are size of property, age, industrial (Dummy), retail (Dummy), and office (Dummy).

In the first column (NN), we find that the fit decreases dramatically, from 0.14 in Table 2 to 0.31 in Table 4. In other words, the MAPE increased by approximately 130%. Our baseline random effects model (REM) on the other hand deteriorates from 0.15 in Table 2 to 0.23 in Table 4, or an increase of 60%. Even though this decrease is not as steep compared to the standard neural network in column 1, it is still substantial on its own. Looking at our iterative approach of mixing random effects with neural networks (iNNREM), the fit deteriorates on average with 36%, as it goes from 0.11 in Table 2 to 0.15 in Table 4. However, what is more interesting, is that the fit hardly deteriorates for properties that were sold more than once ($k > 1$). As noted earlier, databases mature, and more repeat sales enter the data. This means that we can achieve the 10% MAPE even when only using a small set of features. Also note that our baseline random effect model (REM) gives a MAPE for properties that sold more than once ($k > 1$) of 0.132, or 32% higher compared to the iNNREM model. This indicates that even though we have very little features, the neural network still improves fit compared to the linear representation in the REM. As before, there is hardly any difference between the fully specified Bayesian model (fNNREM) in column 3, and our iterative approach in the fourth column (iNNREM).

In Figure 5, we give a graphical representation of the Besag component of the iNNREM model with a reduced set of features. Compared to the same map on the full set of features in Figure 3, the regions with premia and discounts largely coincide. This result indicates that in general there is some sort of (negative) correlation between the net operating income and cap rates in Phoenix. Properties with high net operating income also tend to have low cap rates. This in itself might be driven by a risk channel, where a high net operating income property is perceived to be a less risky investment. However, there is one large difference between the heatmaps (Figures 3–5), namely, the range of possible values. Whereas on the full set of features, the range only went from approximately -10% to $+10\%$, the range of the Besag estimates on the reduced dataset goes approximately from -40% to $+20\%$. The spatial random effect effectively controls for the unobserved heterogeneity created by leaving out the key variables. Both net operating income and walk

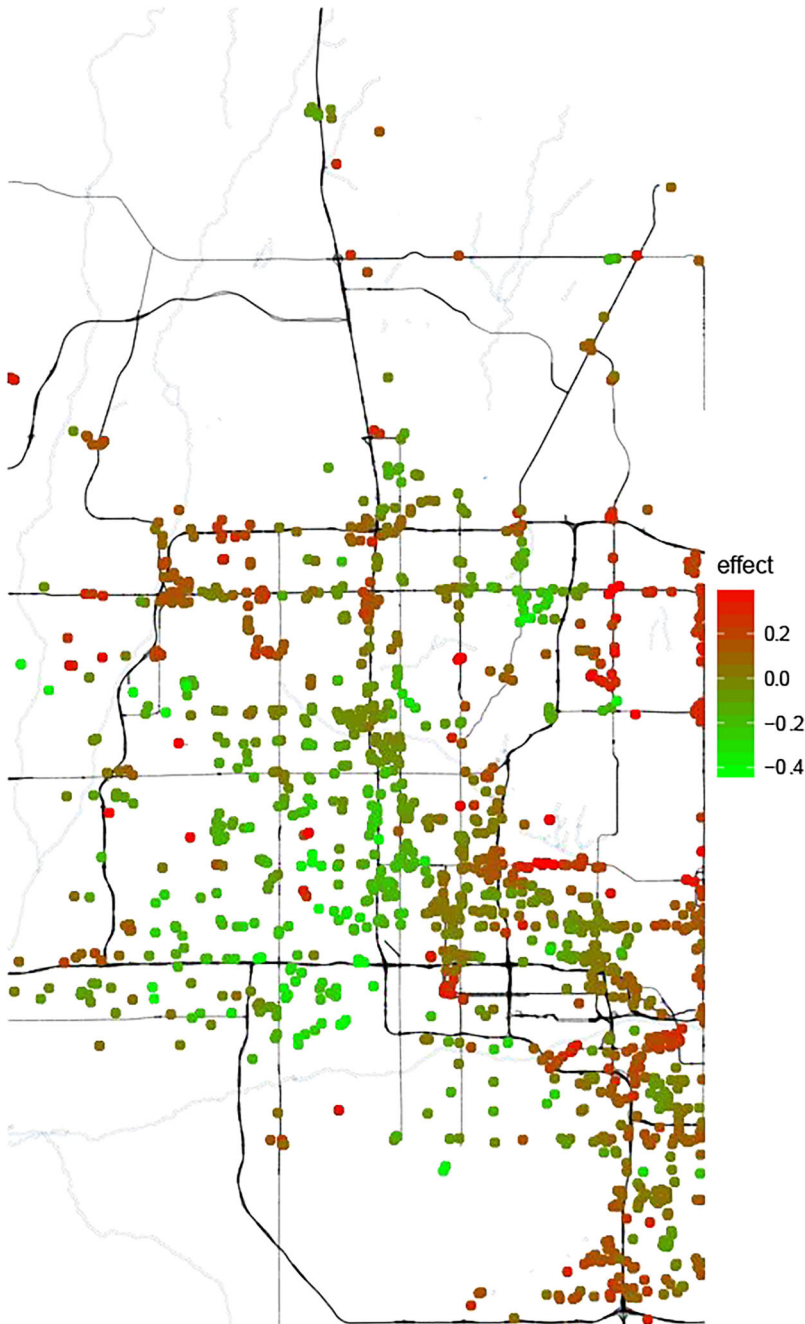


FIGURE 5 Heatmap on reduced dataset. Every red dot represents a property sold in our data. Darker red (green) color represents higher (lower) value for our Besag spatial random effect (θ_i in Equation 2d). The estimates are taken from the iNNREM model, and is the average over the 10 K -folds. [Color figure can be viewed at wileyonlinelibrary.com]

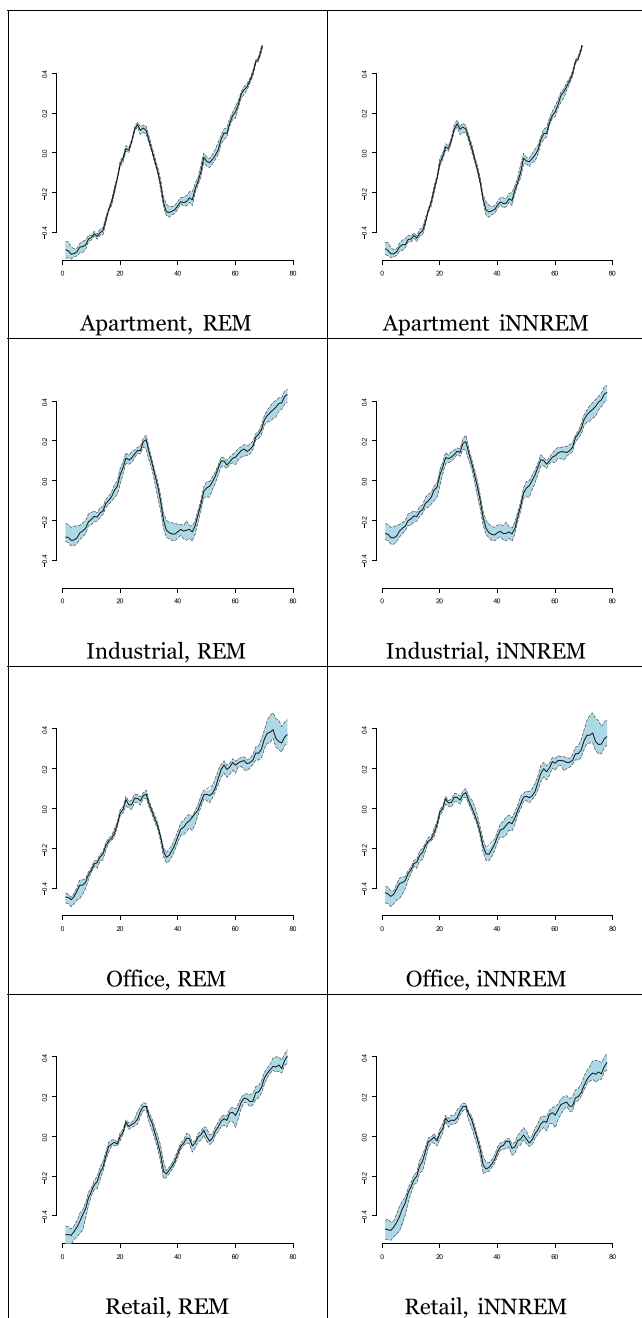


FIGURE 6 Estimated subtrends, using subset of Features only. REM: random effects model, iNNREM is our iterative approach of combining random effects models with neural networks. These models do not include net operating income, nor walk score. [Color figure can be viewed at wileyonlinelibrary.com]

score are spatially clustered. Hence, the performance does not deteriorate as much when such key variables are omitted from the model.

Next, we will focus on the estimated property type subtrends. Note that in this case, we can directly interpret the trends as price indexes now, as we are not controlling for net operating income in this section. As such, you cannot directly compare the estimated log trends between the reduced and the full set of data. We provide the indexes themselves in Figure 6 and a summary of the average returns and ranges over the 10 *K*-folds in Table 5.


TABLE 5 Average index returns and range (in log levels) for Phoenix, reduced set of data.

Property type	Returns		Range	
	(1) REM	(2) iNNREM	(3) REM	(4) iNNREM
Apartment	0.0674	0.0672	0.0385	0.0385
Industrial	0.0372	0.0368	0.0620	0.0617
Office	0.0424	0.0407	0.0592	0.0654
Retail	0.0466	0.0435	0.0542	0.0634

Note: REM = random effects model, as given by Equations (2a)–(2e). iNNREM is an iterative approach of combining random effects with neural networks, as shown in algorithm 1. The average returns are calculated by taking the average values over all the K -folds. We multiply the returns by 4 to get annualized returns. The range is computed by taking the max value for every period per property type and subtracting the minimum value. The reported numbers are the averages over these ranges per property type. For this analysis, we dropped the net operating income (per square foot) and the walk score, as compared to the results in Table 3. The remaining explanatory variables are size of property, age, industrial (Dummy), retail (Dummy), and office (Dummy). The dependent variable is the log sale price minus the average log sale price.

As compared to the earlier findings in Table 3 (where we used the full dataset), the estimated log returns are more in line between the REM and iNNREM models. For example, the REM gives an average annual appreciation of 4.2% for office, whereas it is 4.1% according to the iNNREM. On the full dataset, the estimated returns were 3.4% and 2.6%, respectively. (The fact that the returns are lower on the full dataset indicates that—on a market level—net operating income increased over time.) In general, the REM estimated returns are not as correlated as they were before. To be more specific, the average correlation between the index returns using the full set of characteristics was 0.99, whereas it is “only” 0.80 for the log index returns on the reduced set. On the reduced set, we find that industrial properties appreciated the least in value during our analyzed period (with 3.7% annually), and apartment the most (with 6.7% annually). Unsurprisingly, we find that without net operating income and walk score, the range of the estimated (log) indexes increase over the 10 K -folds, making these estimates less stable. However, the ranges are more in line as well between the two models.

To give more insight into the differences between the performance of the iNNREM when applying it to the full and reduced dataset, we provide a variance decomposition of the fitted values for the iNNREM. More specifically, let g be all the individual components that constitute the fit (\hat{y}) of the iNNREM model, or $g = \{\hat{\mu}, \hat{\delta}, \hat{\theta}, \hat{\phi}, f(x_{itp})\}$, where $\sum g = \hat{y}$. In words, these elements are: the common trend, the property type subtrends, the spatial random effects (Figures 3 and 5), the property specific random effects, and the impact of the property characteristics on property values respectively. Then the individual contribution of each element is given by: $|g| / \sum |g|$. We only look at the out-of-sample fit for this exercise. The resulting fractions are given in Table 6. Column 1 contains the results using the full data, and column 2 gives the results for the reduced dataset.

A few things are of interest. First, note that less variation is explained by the property type subtrends in the full dataset, as compared to the reduced dataset. This was expected, as the net operating income (which is omitted in the reduced dataset) explains changes in property values over time as well as cap rates. The common trend is not impacted as much, as this captures the change in cap rate, which we do not control for.¹⁰ The combined explanatory power of the temporal components is between 18% and 22%. Second, is that the impact of the (Besag) spatial random

¹⁰ Cap rates are driven by the asset markets. Most famously, interest rates and subsequent discount rates. This impact real estate values relatively similarly throughout the US.

TABLE 6 Contribution of the individual elements to the total fit of the iNNREM, for both full and reduced datasets.

Description	Parameter	(1)	(2)
		Full	Reduced
Common trend	$\hat{\mu}$	15.36%	14.94%
Property type subtrends	$\hat{\delta}$	2.92%	8.46%
Spatial random effect	$\hat{\theta}$	3.96%	11.55%
Property random effect	$\hat{\phi}$	0.07%	2.80%
Property characteristics	$f(x_{itp})$	77.69%	62.26%

Note: The full set of explanatory variables are NOI (p. sqf), walk score, size of property, age, industrial (Dummy), retail (Dummy), and office (Dummy). For the *reduced* dataset, we omit net operating income and walk score. The contributions are calculated by dividing the absolute values of individual elements/parameter, by the sum of the absolute values of these same elements/parameters.

effect triples between the full and reduced dataset. More specifically, the contribution of the spatial random effect on the fit of the iNNREM is 3.96% on the full dataset, and 11.55% on the reduced dataset. These further buttress how the random effects control for the unobserved heterogeneity. The same can be observed when looking at the property specific random effects ($\hat{\phi}$). On the full dataset, the property specific random effect has close to zero impact. For the reduced dataset, the contribution is close to 3%. This number is still modest, because it represents the average of the entire out-of-sample MAPE. For properties with repeat sales in the training set, this component increases in importance (not shown, but available upon request). In total, these two random effects contribute 4%–15% of the total fit. Finally, the fit that is provided by the neural network (and thus the property characteristics) is given in the last row of Table 6. For the full dataset, we find that the contribution to the total fit is 78%. This number drops considerably to 62% for the reduced dataset. This drop was obviously expected, given we omitted key variables from X .

5.2 | Including spatial and time series data in neural network

So far, we have not included time and location dummies in the neural network. The reason is simply that this can result in too many parameters to estimate, and as a consequence will dominate the neural network. To elaborate, we only have 4 property characteristics, and we would have to include (20 years times 4 quarters) 80 quarterly dummies. This does not even include the interaction with the property types and any location dummies we might want to include. However, there are more parsimonious ways of including time and location parameters in a neural network. This could make the comparison between the performance of the standard neural network and the random effects and mixed models more fair.

First, we add a linear time trend. Second, we want to include the latitude and longitude. Using latitude and longitude coordinates have been used in previous literature (see Yadav & Chandel, 2014, for a review). However, using said coordinates creates an issue as they are 2 features that represent a three-dimensional space. For example, one issue might arise when looking at two extreme coordinates. In this case, the actual observations will be close by. One way to circumvent this and other issues is by transforming latitude and longitude in the following way:

$$s^x = \cos(\text{Lat}) \times \cos(\text{Lon}), s^y = \cos(\text{Lat}) \times \sin(\text{Lon}), s^z = \sin(\text{Lat}).$$

**TABLE 7** Results for neural network with spatial and time series data included.

	(1) Full	(2) Reduced
MAPE, out of sample, all	0.121	0.250
MAPE, out of sample, $k = 0$	0.125	0.265
MAPE, out of sample, $k = 1$	0.116	0.239
MAPE, out of sample, $k > 1$	0.114	0.190
MAPE, in sample, all	0.095	0.214
Obs, all	2652	2652
Obs, $k = 0$	1685	1685
Obs, $k = 1$	659	659
Obs, $k > 1$	308	308

Note: All estimates are based on the baseline neural network, as provided by Equations (1a)–(1d). MAPE = mean absolute prediction error. *Obs* = number of observations. k gives the amount of times the (unique) property was sold in the training sample. We K -folded 10 times to get our out-of-sample statistics. The *full* set of explanatory variables are NOI (p. sqf), walk score, size of property, age, industrial (Dummy), retail (Dummy), and office (Dummy), a time trend, $s^x (= \cos(\text{Lat}) \times \cos(\text{Lon}))$, $s^y (= \cos(\text{Lat}) \times \sin(\text{Lon}))$, and $s^z (= \sin(\text{Lat}))$. For the *reduced* dataset, we omit net operating income and walk score.

By applying these transformations, the spatial dependence between nearby properties is captured. Together with the time trend, it gives in total four new features within the neural network. We provide the results of the neural network with a time trend and locational feature for both the full (column 1) and the reduced data set (column 2) in Table 7.

Somewhat comforting is the fact that the fit—indeed—improves when adding spatiotemporal features. For example, the overall MAPE of the neural network went from 0.139 (Table 2, column 1) to 0.121 (Table 7, column 1), or an improvement in MAPE of 13%. On the reduced set, the improvement is even larger with 20%, indicating that the spatiotemporal features partly control for the left-out variables. A second observation is that including these features helps the fit when a property got sold more than once, especially on the reduced set. More specifically, a property that did not sell in the training set ($k = 0$) has an average MAPE of 0.265 (column 2, Table 7), whereas if the property was sold more than once in the training set ($k > 1$), the MAPE is 0.190 on average. This is an improvement of almost 30%. The final observation is that the neural network with spatiotemporal features still performs worse compared to our iterative approach. Our iNNREM got an average MAPE of 0.109 and 0.148 for the full and reduced sample respectively (column 4, Tables 2–4). In addition, it is still not possible to produce an index or a heatmap using the spatiotemporal augmented neural network.

5.3 | Using other machine learning algorithms

In this section, we will replace the neural network in our algorithm, as described in Section 2.4, with other readily available machine learning algorithms. We use all the features available to us: net operating income, size of property (in square footage), log age of the building, walk score, and property type dummies. Even though we cannot provide the full specification of all the different machine learning algorithms used in this section, we will provide a short description of the parameters we will be tuning for each model.

TABLE 8 Results of other ML model for Phoenix.

	(1) iSVM	(2) iRT	(3) iGBM
MAPE, out of sample, all	0.110	0.116	0.116
MAPE, out of sample, $k = 0$	0.114	0.122	0.120
MAPE, out of sample, $k = 1$	0.107	0.114	0.112
MAPE, out of sample, $k > 1$	0.092	0.092	0.099
MAPE, in sample, all	0.085	0.085	0.082
Obs, all	2652	2652	2652
Obs, $k = 0$	1694	1694	1694
Obs, $k = 1$	647	647	647
Obs, $k > 1$	311	311	311

Note: iSVM = iterative approach as given in Section 2.4, but we replace the neural network with a Support Vector Machine. iRT = iterative approach as given in Section 2.4, but we replace the neural network with a Regression Tree (the M5 algorithm to be exact). iGBM = iterative approach as given in Section 2.4, but we replace the neural network with a Gradient Boosting Machine. MAPE = mean absolute prediction error. Obs = number of observations. k gives the amount of times the (unique) property was sold in the training sample. We K -folded 10 times to get our out-of-sample statistics. The explanatory variables are net operating income (per square foot), walk score, size of property, age, industrial (Dummy), retail (Dummy), and office (Dummy). The dependent variable is the log sale price minus the average log sale price.

Our first alternative is the Support Vector Machine (iSVM). We use a polynomial kernel—which we found gave best fit in earlier experiments—and tune the following hyperparameters: learning distance and the margin of the hyperplane, also known as the “ σ ” and “ C ” in the SVM literature, respectively. The second alternative is a Regression Tree. More specifically, we use the M5-algorithm (Wang & Witten, 1996). This model is denoted *iRT*. The trees are pruned and smoothed. Finally, our third alternative example is that of a Gradient Boosting Machine (iGBM). With this specification, we tune the number of trees, the interaction debt, and the shrinkage parameter (also known as the “learning rate”). All hyperparameter tuning is done via a separate 10 K -folding within each fold. The resulting MAPE statistics can be found in Table 8. Some summary statistics on the estimated indexes can be found in Table 9.

The model fit is similar to the earlier findings when using a neural network (Table 2). The average out-of-sample MAPE is between 0.110 and 0.116, whereas the neural network gave a fit of 0.109 (column 4, Table 2). As before, the fit improves whenever the property is sold (at least once) in the training sample. We find that especially the estimated indexes between the Support Vector Machine and our earlier estimated neural network are very similar. Both in average return (column 1) and average range over the K -folds (column 4). In fact, the range of possible indexes is lower for the aSVM compared to the iNN, except for apartment. The Regression Tree and Gradient Boosting Machine result in the least stable indexes; see columns 5–6, respectively, in Table 9. Thus, even though the cross-sectional fit is comparable to the other machine learning algorithms used in this article, their indexes leaves room for improvement.

5.4 | Using root mean squared errors as metric of accuracy

Throughout this article, we have been using the Mean Average Prediction Error (MAPE) to measure the accuracy of our models. Another popular way to express accuracy is by looking at the



TABLE 9 average index returns and range (in log levels) for Phoenix, using other machine learning algorithms.

Property type	Returns			Range		
	(1) iSVM	(2) iRT	(3) iGBM	(4) iSVM	(5) iRT	(6) iGBM
Apartment	0.0360	0.0157	0.0219	0.0301	0.0438	0.0405
Industrial	0.0269	0.0200	0.0220	0.0297	0.0516	0.0406
Office	0.0229	0.0218	0.0215	0.0385	0.0367	0.0406
Retail	0.0223	0.0255	0.0224	0.0397	0.0793	0.0412

Note: iSVM = is an iterative approach of combining random effects with a Support Vector Machine. iRT is an iterative approach of combining random effects with a Regression Tree (M5 algorithm). iGBM is an iterative approach of combining random effects with a Gradient Boosting Machine. The principle of mixing both approaches is given in algorithm 1. The average returns are calculated by taking the average values over all the K -folds. We multiply the returns by 4 to get annualized returns. The range is computed by taking the max value for every period per property type and subtracting the minimum value. The reported numbers are the averages over these ranges per property type. The explanatory variables are net operating income, walk score, size of property, age, industrial (Dummy), retail (Dummy), and office (Dummy). The dependent variable is the log sale price minus the average log sale price.

TABLE 10 Main results for Phoenix using the RMSE as the measure of accuracy.

	(1) NN	(2) REM	(3) fNNREM	(4) iNNREM
MAPE, out of sample, all	0.189	0.210	0.151	0.151
MAPE, out of sample, $k = 0$	0.190	0.225	0.152	0.156
MAPE, out of sample, $k = 1$	0.189	0.190	0.156	0.147
MAPE, out of sample, $k > 1$	0.182	0.163	0.139	0.127
MAPE, in sample, all	0.168	0.135	0.141	0.108
Obs, all	2652	2652	2652	2652
Obs, $k = 0$	1685	1694	1694	1694
Obs, $k = 1$	659	647	647	647
Obs, $k > 1$	308	311	311	311

Note: RMSE = root mean squared error. In this table, we replicate the results from Table 3, but we express the measure of fit as the RMSE instead of the MAPE. NN = baseline neural network, as provided by Equations (1a)–(1d). REM = random effects model, as given by Equations (2a)–(2e). fNNREM = full Bayesian representation of a neural network with random effects; see Equation (3). iNNREM is an iterative approach of combining random effects with neural networks, as shown in algorithm 1. MAPE = mean absolute prediction error. Obs = number of observations. k gives the amount of times the (unique) property was sold in the training sample. We K -folded 10 times to get our out-of-sample statistics. The explanatory variables are net operating income (per square foot), walk score, size of property, age, industrial (Dummy), retail (Dummy), and office (Dummy). The dependent variable is the log sale price minus the average log sale price.

Root Mean Square Error (RMSE). *Average* errors are less sensitive to large outliers as compared to *squared* errors, meaning the MAPE and RMSE can give different results. Therefore, as a robustness, we replicate our main findings of Table 2 but express the measure of fit using the RMSE. The results can be found in Table 10.

Compared to the MAPE statistics in Table 2, the RMSE results are very similar. More specifically, the RMSE is still superior for the mixed models (fNNREM and iNNREM), especially for properties sold more than once. The difference in fit *between* the fully Bayesian and its iterative counterpart is negligible as was previously the case as well. Taken together,

we do not believe our specific measure of accuracy (MAPE) is a first order concern in this article.

5.5 | Index stability (alternative)

In Section 4.2, we inspected the stability of the indexes by looking at the range of possible indexes over the 10K-folds. Here, we introduce an alternative approach of testing index stability by following the procedure detailed in Francke and van de Minne (2017) and Calainho et al. (2022).

More specifically, we pick a market with plenty of observations and estimate the property type subtrends from this data using our algorithmic approach (iNNREM). Afterward, we keep sampling—without replacement—smaller subsets of the data and re-estimate the subtrends from that subset, making the data “scarce.” If the subtrends based on a small subset of data still resembles the “true” estimates based on the full sample, we can conclude that the methodology provides robust trends even in small data environments.

In addition, we will also compare our estimated indexes with the indexes estimated from a more conventional hedonic model. An index based on structural time series models are expected to be more smooth in the short-run compared to indexes based on fixed effects (Francke & van de Minne, 2017). However, in the long-run the indexes should more or less coalesce. This helps us validate our methodology. The hedonic model is only estimated on the *full* sample.

The conventional hedonic model is given by

$$y_{itp} = x'_{itp} \beta + \delta_{tp} + \varepsilon_i, \quad \varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2), \quad (4)$$

where x includes (log) square footage, walk score, (log) age, and location fixed effects for six submarkets, of which one is omitted to circumvent the dummy trap.¹¹ The parameter estimates are given by β . These will not be reported here but are available upon request. More importantly, δ_{tp} are quarterly times property type fixed effects. The model is estimated by OLS.

We picked the largest market (New York metro area) in our dataset for this exercise and dropped net operating income (we miss NOI for about two-thirds of our data). The total amount of observations for the full dataset is over 24,000. (A summary of the data can be found in Table 11.) We sample without replacement 25% (6250 *obs*), 10% (2400 *obs*), 5% (1200 *obs*), 2.5% (600 *obs*) and 1% (240 *obs*) from this data and re-estimate the subtrends. The results are presented in the Figure 7. Note that that there is no K -folding in this exercise. We simply iterate until the in-sample MAPE does not improve anymore and report the in-sample subtrends.

The blue line in Figure 7 represents the subtrends based on only 1% of the data (or approximately 240 observations). These trends are the only visible dissonant among its counterparts for all property types. The 1% index stays relatively flat, which is likely caused by the lack of observations itself (Francke et al., 2023). However, already the 2.5% index (orange line with cross) do we see subtrends that resembles the subtrends based on the full sample (black line with blocks) to a large extent for all property types. The correlation between the returns of the 5%-index and the full sample index is already 0.85 on average (not reported here, but available upon request).

¹¹ The markets are Long Island, Manhattan, North New Jersey, the Boroughs, Stamford, and Westchester.

**TABLE 11** Descriptive statistics of our main variables in New York.

Statistic	N	Mean	St. Dev.	Min	Max
Sales price	24,336	\$23,672,558	\$109,354,087	\$80,000	\$5,400,000,000
Building size (sqft)	24,336	71,084	177,636	1016	10,247,000
Age of building	24,336	68.973	32.870	1	273
Walk score	24,336	82.986	24.440	0	100
Property type count	All	Apartment	Industrial	Office	Retail
	24,336	10,542	3593	4670	5531

Note: Walk score is a number from 0 to 100, where 0 means not walkable, and 100 denotes very walkable.

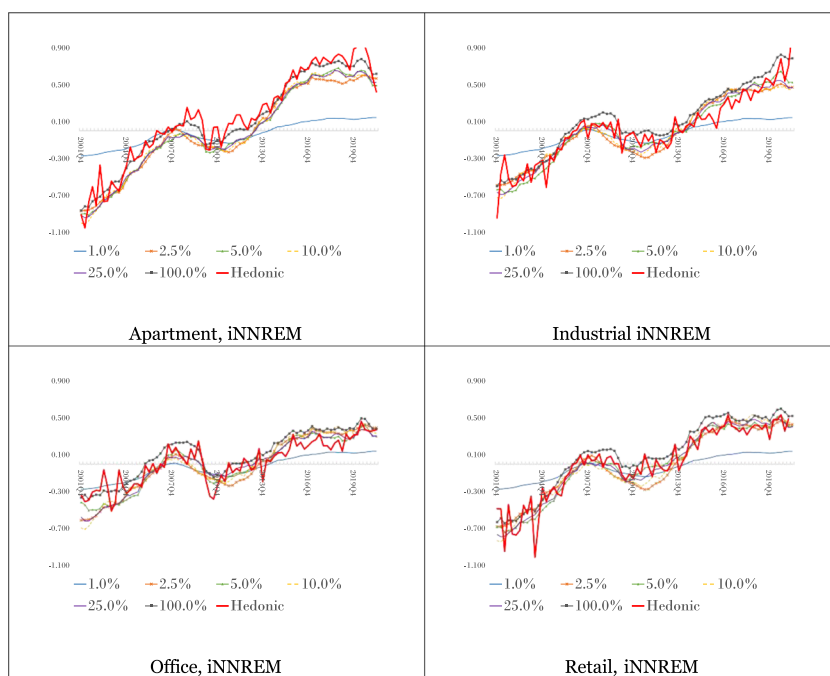


FIGURE 7 Estimated subtrends for New York after Sampling the Data. iNNREM is our iterative approach of combining random effects models with neural networks. The model is estimated on New York ($N = 25,000$) without the inclusion of net operating income due to missing observations for said variable. We sample the data without replacement and re-estimate the data. This gives a sense of index stability. We also provide the index from a standard hedonic model with time dummies (times property type), this model is only estimated on the full data using Equation (4). [Color figure can be viewed at wileyonlinelibrary.com]

As such, we can say that the evidence so far indicates that our proposed methodology produces reliable and stable indexes even in small sample environments.

If we compare our findings with the standard hedonic model (thick red line), we get results that were expected a priori. The estimated indexes from the more conventional hedonic model are impacted by (cross-sectional) noise, resulting in volatile indexes. However, in the long-run, both the hedonic model and our iterative approach result in similar indexes. Given that the indexes are based on a very different model and estimation technique, some differences are expected. The differences in Figure 7 do not raise any concerns.

6 | CONCLUSION

This article focuses on the prediction of property prices in the commercial real estate (CRE) asset market by combining two cultures of statistical modeling: statistical models using stochastic data models (econometric models, like linear regression) and statistical models using algorithms (machine learning). This allows us to combine the strengths of both approaches. Machine learning (ML) algorithms are easy-to-use and can describe complex nonlinear relations including interaction effects between explanatory variables. Unlike ML algorithms, econometric models can explicitly control for spatial and temporal correlations and unobserved heterogeneity. Our combined approach leads to accurate out-of-sample predictions and to estimates of location values (heatmaps) and property price indexes.

For our baseline specification, we break down sales prices into five components: (1) a common trend; (2) a property type trend; (3) a spatial component; (4) a property random effect to capture unobserved heterogeneity; and (5) a property characteristics component. The first four components are modeled by a random effects model, estimated using Integrated Laplace Approximation. The fifth component is calibrated using a neural network. To achieve this, we first estimate the joint model with a linear function for the property characteristics component. Second, we take the ML “residual” as the observed value minus the estimates of components 1 to 4, and calibrate a ML algorithm. Third, we take the DGP “residual” as the observed value minus the prediction of component 5, and estimate the econometric model, consisting of components 1 to 4. We repeat these steps until convergence occurs.

We apply our proposed iterative methodology to 2652 commercial real estate transactions in Phoenix, AZ. The data was provided to us by MSCI/Real Capital Analytics for the period 2001–2021. We find an average out-sample MAPE of 11%, which is as good or even lower compared to the average appraisal error found in the United States (Cannon & Cole, 2011; Fisher et al., 1999). The fit is especially good for properties that sold at least once in the training set, which gives a MAPE of 9%. Moreover, it is possible to estimate a full Bayesian model that encompasses all five components into one model (including the neural network). This model takes approximately 400 times longer to estimate and is not possible for all machine learning algorithms. However, it does provide an important benchmark to which we can compare our iterative approach. Fortunately, the estimates are close to identical to our iterative approach, with a correlation of 1.00 between the fitted values of the two models. When leaving out key variables, net operating income and the walk score, we find that the spatial and property random effects play a greater role. This stresses how important such random effects are in controlling for unobserved heterogeneity. Augmenting the baseline neural network with spatiotemporal variables, does improve its fit, but it is still nowhere near the fit produced by our proposed methodology. We also replace the neural network with other machine learning tools: Support Vector Machine, Regression Tree, and Gradient Boosting Machine. The fit is comparable, although the estimated indexes of the tree-based models are less stable.

The fit reported above is impressive. However, there is room for improvement in future research. The goal of this article is to present the methodology, not to get the best fit per se. For example, there are 100s of other off-the-shelf machine learning algorithms that could be tested. And even within the machine learning algorithms used in this article, there are many smaller and larger optimizations possible that we did not pursue for the sake of brevity (e.g., changing the number of hidden layers and/or nodes in the neural network). For the spatial random effects, a different definition of “neighbor” could be used, or a different spatial structure could be proposed altogether. The same goes for the temporal components, which we modeled as



simple random walks. However, extant literature found that real estate prices evolve more around autoregressive terms (Nagaraja et al., 2011; Van de Minne et al., 2020). Finally, we do not consider an “ensemble” of models in this article. An easy example of an ensemble would be obtained by taking the average (weighted) prediction of multiple different models. If the residuals of the individual models have low correlation (or preferably negative), the fit of the ensemble model will be superior to the fit of the individual models (Mullainathan & Spiess, 2017). In short, an even more impressive fit should easily be achievable in the future. Future research could also apply the proposed methodology to other illiquid markets with little features, like low granular housing markets.

ACKNOWLEDGMENTS

We would like to thank MSCI/RCA for providing us with the data. We are also grateful to the participants of the CRE research seminar at MIT in 2022, and for the participants of the Real Estate Finance and Investment Symposium in Hong Kong in 2023, providing us with useful feedback. Finally, we would like to express our gratitude to our referees.

ORCID

Marc Francke  <https://orcid.org/0000-0002-7239-4868>

Alex van de Minne  <https://orcid.org/0000-0003-1675-5419>

REFERENCES

- Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4, e00938.
- Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, S., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2020). Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58, 82–115.
- Athey, S. (2018). The impact of machine learning on economics. In *Economics of artificial intelligence*. University of Chicago Press.
- Besag, J. (1974). Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 36, 192–236.
- Besag, J., & Kooperberg, C. (1995). On conditional and intrinsic autoregressions. *Biometrika*, 82, 733–746.
- Besag, J., York, J., & Mollie, A. (1991). Bayesian image restoration, with two applications in spatial statistics. *Annals of the institute of statistical mathematics*, 43, 1–20.
- Bishop, C. M., & Nasrabadi, N. M. (2006). *Pattern recognition and machine learning*. Springer.
- Bollerslev, T., Patton, A. J., & Wang, W. (2016). Daily house price indices: Construction, modeling, and longer-run predictions. *Journal of Applied Econometrics*, 31, 1005–1025.
- Breiman, L. (2001). Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical Science*, 16, 199–231.
- Calainho, F. D., van de Minne, A., & Francke, M. K. (2022). A machine learning approach to price indices: Applications in commercial real estate. *The Journal of Real Estate Finance and Economics*, 1–30.
- Cannon, S. E., & Cole, R. A. (2011). How accurate are commercial real estate appraisals? Evidence from 25 years of NCREIF sales data. *The Journal of Portfolio Management*, 37, 68–88.
- Ceyhan, C. (2017). Evaluation of machine learning techniques for house valuations. Technical report. Erasmus University.
- Chiarazzo, V., Caggiani, L., Marinelli, M., & Ottomanelli, M. (2014). A neural network based model for real estate price estimation considering environmental quality of property location. *Transportation Research Procedia*, 3, 810–817.
- Clapham, E., Englund, P., Quigley, J. M., & Redfearn, C. L. (2006). Revisiting the past and settling the score: Index revision for house price derivatives. *Real Estate Economics*, 34, 275–302.

- Clapp, J. M., & Giaccotto, C. (1999). Revisions in repeat-sales price indexes: Here today, gone tomorrow. *Real Estate Economics*, 27, 79–104.
- 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, 191–209.
- Deppner, J., & Cajias, M. (2022). Accounting for spatial autocorrelation in algorithm-driven hedonic models: A spatial cross-validation approach. *The Journal of Real Estate Finance and Economics*, 68, 235–273.
- Deppner, J., von Ahlefeldt-Dehn, B., Beracha, E., & Schaefers, W. (2023). Boosting the accuracy of commercial real estate appraisals: An interpretable machine learning approach. *The Journal of Real Estate Finance and Economics*, 1–38.
- Fisher, J. D., Miles, M. E., & Webb, R. B. (1999). How reliable are commercial appraisals? Another look. *Real Estate Finance*, 16, 9–15.
- Francke, M., Rolheiser, L., & Van de Minne, A. M. (2023). Estimating census tract house price indexes: A new spatial dynamic factor approach. *The Journal of Real Estate Finance and Economics*, 1–32.
- Francke, M. K. (2010). Repeat sales index for thin markets: A structural time series approach. *Journal of Real Estate Finance and Economics*, 41, 24–52.
- Francke, M. K., & van de Minne, A. M. (2017). The hierarchical repeat sales model for granular commercial real estate and residential price indices. *The Journal of Real Estate Finance and Economics*, 55, 511–532.
- Francke, M. K., & Van de Minne, A. M. (2021). Modeling unobserved heterogeneity in hedonic price models. *Real Estate Economics*, 49, 1315–1339.
- Francke, M. K., & Van de Minne, A. M. (2022). Daily appraisal of commercial real estate: A new mixed frequency approach. *Real Estate Economics*, 50, 1257–1281.
- Francke, M. K., & Vos, G. A. (2004). The hierarchical trend model for property valuation and local price indices. *Journal of Real Estate Finance and Economics*, 28, 179–208.
- Gelman, A. (2006). Prior distributions for variance parameters in hierarchical models. *Bayesian Analysis*, 1, 515–534.
- Günther, F., & Fritsch, S. (2010). Neuralnet: Training of neural networks. *R J.*, 2, 30.
- Guo, Xi., Zheng, S., Geltner, D. M., & 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.
- Hoffman, M. D., & Gelman, A. (2014). The No-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15, 1593–1623.
- 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, 202–211.
- Lorenz, F., Willwersch, J., Cajias, M., & Fuerst, F. (2023). Interpretable machine learning for real estate market analysis. *Real Estate Economics*, 51, 1178–1208.
- McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5, 115–133.
- Molnar, C. (2019). Interpretable Machine Learning. A Guide for Making Black Box Models Explainable. (Leanpub, <https://christophm.github.io/interpretable-ml-book/>)
- Mullainathan, S., & Spiess, J. (2017). Machine learning: An applied econometric approach. *Journal of Economic Perspectives*, 31, 87–106.
- Nagaraja, C. H., Brown, L. D., & Zhao, L. H. (2011). An autoregressive approach to house price modeling. *The Annals of Applied Statistics*, 5, 124–149.
- Nghiep, N., & Al, C. (2001). Predicting housing value: A comparison of multiple regression analysis and artificial neural networks. *Journal of Real Estate Research*, 22, 313–336.
- Pace, R. K., & Hayunga, D. (2020). Examining the information content of residuals from hedonic and spatial models using trees and forests. *The Journal of Real Estate Finance and Economics*, 60, 170–180.
- Peterson, S., & Flanagan, A. (2009). Neural network hedonic pricing models in mass real estate appraisal. *Journal of Real Estate Research*, 31, 147–164.
- Rue, H., Martino, S., & Chopin, N. (2009). Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71, 319–392.
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533–536.
- Saiz, A. (2010). The geographic determinants of housing supply. *The Quarterly Journal of Economics*, 125, 1253–1296.



- Van de Minne, A. M., Francke, M. K., & Geltner, D. M. (2022). Forecasting US commercial property price indexes using dynamic factor models. *Journal of Real Estate Research*, *44*, 29–55.
- Van de Minne, A. M., Francke, M. K., Geltner, D. M., & White, R. (2020). Using revisions as a measure of price index quality in repeat-sales models. *The Journal of Real Estate Finance and Economics*, *60*, 514–553.
- Varian, H. R. (2014). Big data: New tricks for econometrics. *Journal of Economic Perspectives*, *28*, 3–28.
- Wang, Y., & Witten, I. H. (1996). Induction of model trees for predicting continuous classes. Technical report. University of Waikato, Department of Computer Science.
- Widrow, B., & Hoff, M. E. (1960). Adaptive switching circuits. Technical report. Stanford Univ Ca Stanford Electronics Labs.
- 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*, 185–201.
- Yadav, A. K., & Chandel, S. S. (2014). Solar radiation prediction using artificial neural network techniques: A review. *Renewable and Sustainable Energy Reviews*, *33*, 772–781.
- Yao, Y., Vehtari, A., Simpson, D., & Gelman, A. (2018). Using stacking to average Bayesian predictive distributions (with discussion). *Bayesian Analysis*, *13*, 917–1007.

How to cite this article: Francke, M., & van de Minne, A. (2024). Combining machine learning and econometrics: Application to commercial real estate prices. *Real Estate Economics*, *52*, 1308–1339. <https://doi.org/10.1111/1540-6229.12483>

STAN code

```

model_string = "
data {
  int<lower=0> N; // number of observations
  int<lower=0> Nt; // number of time periods
  int<lower=0> Np; // number of explanatory variables
  int<lower=0> Nd; // number of property types
  int<lower=0> Nh; // number of knots (only 1 layer)
  int<lower=0> Nn; // scale for ICAR parameter
  int<lower=0> N_edges; // size of vectorized distances
  int<lower=0> Ntest; // size of testing dataset

  int<lower=0,upper=Nn> id[N]; // identification
  int<lower=0,upper=Nd> pt[N]; // property type identification
  int<lower=0,upper=Nn> node1[N_edges]; // vector neighbor 1
  int<lower=0,upper=Nn> node2[N_edges]; // vector neighbor 2
  int<lower=0,upper=Nt> sell[N]; // period sold, sell = 1, ..., Nt

  vector[N] yVar; // explained variable
  matrix[N,Np] x; // explanatory variables

  vector[Ntest] yTest; // test y variable
  int<lower=0,upper=Nt> sTest[Ntest]; // test period sold
  matrix[Ntest,Np] xTest; // testing dataset
  int<lower=0,upper=Nn> idTest[Ntest]; // identification
  int<lower=0,upper=Nd> ptTest[Ntest]; // property type identification
}

```

```

transformed data {
  matrix [Np,Np] A = diag_matrix(rep_vector(1,Np));
  matrix [Np,Np-1] A_qr;
  for(i in 1:Np-1)
    A[Np,i] = -1;
  A[Np,Np] = 0;
  A_qr = qr_Q(A)[1:(Np-1)];
}
parameters {
  vector[Nt-1] innovMu; // time series parameter
  matrix[Nt-1,Nd] innovDelta; // subtrends time series
  vector[Nn] theta; // icar random effects
  vector[Nn] phi; // normal random effects
  real fMu; // first Mu, i.e. the constant.
  ordered[Nh] beta; // bias

  matrix<lower=-10,upper=10>[Np-1,Nh] omega_clean; // weights in layer
  vector[Nh] lambda; // estimates measurement Eq.
  real<lower=0,upper=2> sigMu; // variance of innovations (mu)
  real<lower=0,upper=2> sigDelta; // variance of sub innovations (delta)
  real<lower=0,upper=2> sigEps; // RMSE of measurement Eq.
  real<lower=0> sigTheta; // variance of icar
  real<lower=0> sigPhi; // variance of random effects
}
transformed parameters {
  vector[N] yHat;
  vector[Nt] mu;
  matrix[Nt,Nd] delta;
  matrix[N,Nh] h;
  matrix[Np,Nh] omega;

  vector[N] resids_in;

  // hidden layers
  for(k in 1:Nh)
    omega[k,k] = A_qr * omega_clean[k,k];

  // common trend, including constant
  mu[1] = fMu;
  for(t in 2:Nt)
    mu[t] = mu[t-1] + innovMu[t-1]*sigMu;

  // property type subtrends
  for(p in 1:Nd){
    delta[1,p] = 0;
    for(t in 2:Nt){
      delta[t,p] = delta[t-1,p] + innovDelta[t-1,p]*sigDelta;
    }
  }
  for(k in 1:Nh)
    h[k,k] = x*omega[k,k] + beta[k];
  for(i in 1:N)
    yHat[i] = mu[sell[i]] + delta[sell[i],pt[i]] + theta[id[i]]*sigTheta +
      phi[id[i]]*sigPhi + (inv_logit(h[i])*100)*lambda;
  resids_in = yVar - yHat;
}
model {
  lambda ~ normal(0,1);
  to_vector(omega_clean) ~ normal(0,1);
  to_vector(innovDelta) ~ normal(0,1);
  beta ~ normal(0,1);
  innovMu ~ normal(0,0.1);
  theta ~ normal(0,1);
  phi ~ normal(0,1);
  fMu ~ normal(0,10);
}

```



```

sigEps      ~ normal(0,0.5);
sigMu       ~ normal(0,0.5);
sigTheta    ~ normal(0,1);
sigPhi      ~ normal(0,1);

for(k in 1:Nh)
  sum(omega_clean[,k]) ~ normal(0,1/sqrt(1-inv(Np))); // zero sum constraint within node
yVar        ~ normal(yHat, sigEps);
target      += -0.5*dot_self(theta[node1] - theta[node2]); // vectorized ICAR
sum(theta)   ~ normal(0, 0.001 * Nn); // zero sum constraint on ICAR
}
generated quantities {
  vector[Ntest] yPred;
  matrix[Ntest,Nh] hTest;
  vector[Ntest] resids_out;

  for(k in 1:Nh)
    hTest[,k] = xTest*omega[,k] + beta[k];
  for(i in 1:Ntest)
    yPred[i] = mu[sTest[i]] + delta[sTest[i],ptTest[i]] + theta[idTest[i]]*sigTheta +
              phi[idTest[i]]*sigPhi + (inv_logit(hTest[i])*100)*lambda;
  resids_out = yTest - yPred;
}

```