Article

Machine Learning Applications to Land and Structure Valuation

Michael Mayer [1], Steven C. Bourassa [2,*], Martin Hoesli [3,4] and Donato Scognamiglio [5,6]

[1] Actuarial Department, la Mobilière, 3001 Bern, Switzerland; michael.mayer@mobiliar.ch
[2] Department of Urban and Regional Planning, Florida Atlantic University, Boca Raton, FL 33431, USA
[3] Geneva School of Economics and Management, University of Geneva, 1211 Geneva, Switzerland; martin.hoesli@unige.ch
[4] Business School, University of Aberdeen Business School, Aberdeen AB24 3FX, UK
[5] IAZI AG, 8050 Zurich, Switzerland; donato.scognamiglio@iazi.ch
[6] Institute for Financial Management, University of Bern, 3012 Bern, Switzerland
[*] Correspondence: sbourassa@fau.edu

Abstract: In some applications of supervised machine learning, it is desirable to trade model complexity with greater interpretability for some covariates while letting other covariates remain a "black box". An important example is hedonic property valuation modeling, where machine learning techniques typically improve predictive accuracy, but are too opaque for some practical applications that require greater interpretability. This problem can be resolved by certain structured additive regression (STAR) models, which are a rich class of regression models that include the generalized linear model (GLM) and the generalized additive model (GAM). Typically, STAR models are fitted by penalized least-squares approaches. We explain how one can benefit from the excellent predictive capabilities of two advanced machine learning techniques: deep learning and gradient boosting. Furthermore, we show how STAR models can be used for supervised dimension reduction and explain under what circumstances their covariate effects can be described in a transparent way. We apply the methodology to residential land and structure valuation, with very encouraging results regarding both interpretability and predictive performance.

Keywords: land and structure valuation; machine learning; structured additive regression; gradient boosting; deep learning; interpretability; transparency; hedonic modeling

1. Introduction

Machine learning (ML) techniques—such as artificial neural networks, random forests, and gradient boosting—can be more accurate than standard approaches—such as ordinary least squares—because they automatically learn relevant nonlinearities, transformations, and high-order interactions among the independent variables. These modeling techniques are increasingly being applied to house price modeling. Examples include Worzala et al. (1995), Din et al. (2001), and Zurada et al. (2011) for neural networks, Yoo et al. (2012) for random forests, Kagie and Wezel (2007) and Sangani et al. (2017) for boosting, and many contributions to the machine learning competition platform kaggle.com. Machine learning techniques generally yield more accurate predictions than standard linear models.

While ML techniques can be powerful predictive tools, they are often considered black boxes due to their limited interpretability (Din et al. 2001; Mayer et al. 2019). In some applications, this is particularly problematic due to a need for transparency in describing the effects of certain covariates. For example, in the case of hedonic modeling of house prices, it is desirable to identify the effects of some features, while others do not require individual interpretation. Specifically, there is often a need to explicitly measure the effects of structural (or building) characteristics, while locational attributes can be considered as a

group. This is the case, for example, when appraising properties for taxation or mortgage underwriting purposes. Moreover, there is typically collinearity among locational attributes, making it difficult to identify their separate effects in any case. This paper demonstrates machine learning techniques with restricted complexity that allow for transparency with respect to a subset of the covariates while also achieving excellent predictive results. We apply the methods to house price valuation in South Florida and Switzerland. We believe that this is the first application of these techniques to the problem of real estate valuation.

The model structure considered in this article is described in Hothorn et al. (2010). There, a distributional property ξ of a response variable Y is represented as a function f of input features $X = (X^{(1)}, \ldots, X^{(q)})$ in the structured form

$$\xi(Y \mid X = x) = f(x) = \sum_{j=1}^{p} f_j(x), \tag{1}$$

where each additive component $f_j(x)$ uses a subset of the q covariates. Such models can be viewed as a generalization of *structured additive regression* (STAR) models; see, e.g., Chapter 9 in Fahrmeir et al. (2013). Throughout this article, for the sake of simplicity, we also call them STAR models.

Important special cases of (1) are the generalized linear model (GLM, Nelder and Wedderburn 1972):

$$g(\mathbb{E}(Y \mid X = x)) = \beta_0 + \beta_1 x^{(1)} + \cdots + \beta_p x^{(p)}$$

and the generalized additive model (GAM) without interactions (Hastie and Tibshirani 1986, 1990; Wood 2017)

$$g(\mathbb{E}(Y \mid X = x)) = \beta_0 + f_1(x^{(1)}) + \cdots + f_p(x^{(p)}). \tag{2}$$

The property ξ is implied by specifying a loss function ρ relevant to the problem at hand. Important examples are the squared error loss ($\xi = \mathbb{E}$), the absolute error loss ($\xi =$ Median), and the log loss associated with the negative log-likelihood of the Bernoulli distribution ($\xi = \text{logit} \circ \mathbb{E}$).

Having access to training data (y_i, x_i) with optional case weights w_i, the task to estimate f_j with functions $\hat{f}_j$ amounts to minimizing the average weighted loss over the training data, i.e., solving the task

$$(\hat{f}_1, \ldots, \hat{f}_p) = \underset{(f_1, \ldots, f_p)}{\arg \min} \sum_{i=1}^{n} w_i \rho \left(y_i, \sum_{j=1}^{p} f_j(x_i) \right) \tag{3}$$

subject to algorithmic constraints; see Hothorn et al. (2010) for details.

Different algorithms exist for solving the optimization task (3), each with its own characteristics and limitations. Traditional options include penalized iteratively reweighted least squares (Wood 2017), Bayesian optimization (Umlauf et al. 2012), and model-based gradient boosting or "mboost" (Bühlmann and Hothorn 2007). The latter uses component-wise gradient boosting, where each component represents a base learner chosen from a rich set of possibilities, including (penalized) least squares for linear effects and fixed or random categorical effects, (penalized) B-splines for smooth effects and two-dimensional interaction surfaces, and decision trees for representing the effects of any number of features. During each gradient update, the component with the best loss improvement is multiplied by the specified learning rate and added to the current prediction function. The algorithm "mboost" is implemented in the R package `mboost`; see, again, Hothorn et al. (2010).

In the next section, we show how STAR models in the form of (1) can also be fitted with deep learning, as well as with the high-performance gradient boosting implementations of XGBoost (Chen and Guestrin 2016) and LightGBM (Ke et al. 2017). Additionally, we discuss the possibility of using STAR models for supervised dimension reduction, e.g.,

to compress the information represented by a large set of locational features, and we explain under what circumstances feature effects can be transparently described, including the effects of certain multivariate components. The following two sections apply these ideas to the data from Florida and Switzerland, respectively. A final section discusses our conclusions.

2. Methodology

2.1. STAR Models and Deep Learning

Recently, owing to the flexibility of implementations such as TensorFlow (see Abadi et al. 2016), deep learning approaches can also be used for fitting STAR models. In Agarwal et al. (2021), *neural additive models* are discussed. Such models use sub-networks to represent each input variable $X^{(j)}$ with a smooth function f'_j and connect their output nodes directly to the response Y, which is possibly transformed by a non-linear activation function g^{-1}; see Figure 1a for an illustration. A neural additive model thus allows the fitting of additive models of the form

$$\xi(Y \mid X = x) = \beta_0 + \sum_{j=1}^{p} \beta_j \underbrace{f'_j(x^{(j)})}_{f_j(x^{(j)})},$$

in line with (2). To fit a STAR model in the general case of (1) through deep learning, we need to slightly generalize the structure of the neural additive model to allow the sub-networks to use as input all covariates used in component f_j; see Figure 1b for a schematic example with the equation

$$\xi(Y \mid X = x) = \beta_0 + \beta_1 x^{(1)} + \underbrace{\beta_2 f'_2(x^{(2)})}_{f_2(x^{(2)})} + \underbrace{\beta_3 f'_3(x^{(3)}, \dots, x^{(q)})}_{f_3(x^{(3)}, \dots, x^{(q)})}.$$

Related models are discussed in Rügamer et al. (2021) in an even more general context of multivariate ξ, allowing for modeling of the full conditional distribution of Y.
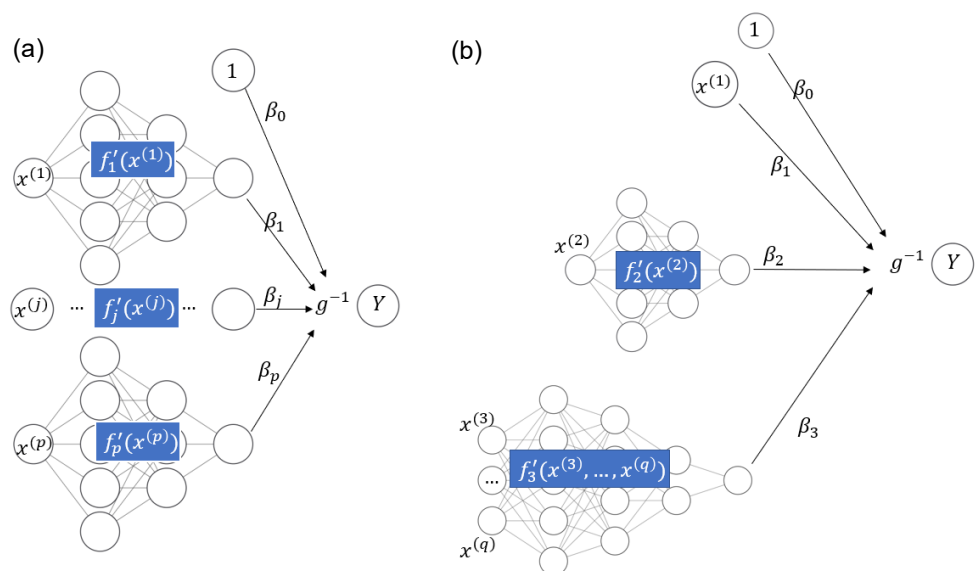
Figure 1. (**a**) Neural additive model. (**b**) STAR model with one linear component, one smooth component of a single covariate, and one smooth component of multiple covariates.

2.2. STAR Models and Gradient Boosting

Tree-based gradient boosting machines (Friedman 2001) belong to the best performing ML algorithms for tabular data, and major implementations, such as XGBoost (Chen

and Guestrin 2016), LightGBM (Ke et al. 2017), and CatBoost (Prokhorenkova et al. 2018), dominate corresponding ML competitions (see Arik and Pfister 2019). It is well known that such models can be used to learn additive models of the form (2) by using tree stumps (trees with a single split) as base learners; see Lou et al. (2012) or Nori et al. (2019). Allowing more splits in the trees would introduce interaction effects, leading to non-additive models.

However, it is less known that some modern implementations of gradient boosting are actually able to non-parametrically fit STAR models of the form (1). The solution is based on imposing so-called *feature interaction constraints*, an idea that goes back to Lee et al. (2015). Their approach was implemented in 2018 in XGBoost, and, on our request, (Mayer 2020) in LightGBM in 2020 as well. Interaction constraints are specified as a collection of feature subsets $\mathcal{X}_1, \ldots, \mathcal{X}_p$, where each $\mathcal{X}_j$ specifies a group of features that are allowed to interact. Algorithmically, the constraints are enforced by the following simple rule during tree growth: *At each split, the set of split candidates is the union of those $\mathcal{X}_j$ that contain all previously used split variables of the current branch.*

Consequently, each tree branch uses features from one feature set $\mathcal{X}_j$ only. Its associated prediction on the scale of ξ contributes to the component f_j of an implicitly defined STAR model of the form (1), where each model component f_j uses feature subset $\mathcal{X}_j$ as specified by the constraints.

An important type of constraint is feature *partitions*, i.e., disjoint feature sets $\mathcal{X}_j$. They include the special case of the collection of singletons $\{X^{(1)}\}, \ldots, \{X^{(p)}\}$ that would produce an additive model of the form (2). For partitions, by the above rule, the first split variable of a tree determines the feature set X_j to be used throughout that tree. Figure 2 illustrates a simple example of such a model with the following equation:

$$\mathbb{E}(\text{Price of house}) = f_1(\text{Living area})$$
$$+ f_2(\text{Distance to center, Distance to rail, Distance to ocean}).$$

The corresponding constraints form a feature partition specified by

$$\mathcal{X}_1 = \{\text{Living area}\} \quad \text{and}$$
$$\mathcal{X}_2 = \{\text{Distance to center, Distance to rail, Distance to ocean}\}.$$
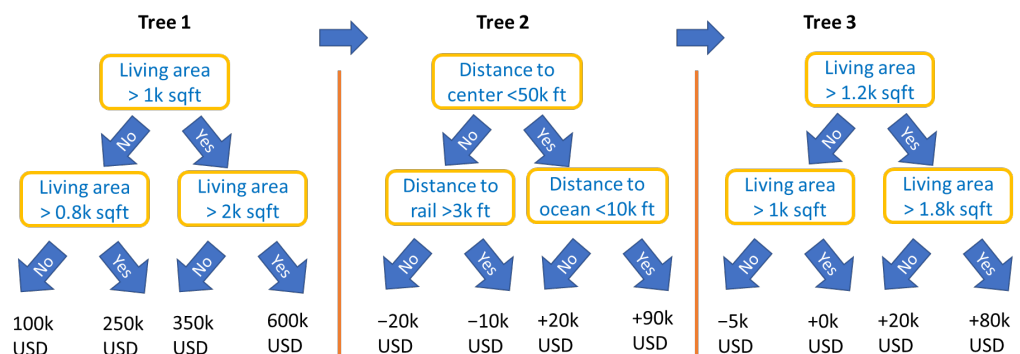
Figure 2. Schematic view of three boosted trees with interaction constraints, representing a STAR model being additive in a living area and in a group of locational variables.

In the two programming languages R and Python, interaction constraints for XGBoost and LightGBM are specified as part of the parameter list passed to the corresponding `train` method. Table 1 shows how to specify the `interaction_constraints` parameter for an example with covariates A, B, C, and D and interaction constraints $\{A\}$ and $\{B, C, D\}$ corresponding to a model as in Figure 2.

Table 1. Specifying the `interaction_constraints` parameter.

	XGBoost	LightGBM
R	`list(0, c(1, 2, 3))`	`list('A', c('B', 'C', 'D'))`
Python	`'[[0], [1, 2, 3]]'`	`[[0], [1, 2, 3]]`

XGBoost and LightGBM offer different loss/objective functions. These specify the functional ζ to be modeled; see Table 2 for some of the possibilities.

Table 2. Important losses and functionals ζ and how their corresponding objectives are specified in XGBoost and LightGBM.

Loss	ζ	XGBoost	LightGBM
Squared error	$\mathbb{E}$	`"reg:squarederror"`	`"mse"`
Log loss	$\mathrm{logit} \circ \mathbb{E}$	`"binary:logistic"`	`"binary"`
Absolute error	median	-	`"mae"`
Quantile loss	p-quantile	-	`"quantile"`
Poisson loss	$\log \circ \mathbb{E}$	`"count:poisson"`	`"poisson"`
Gamma loss	$\log \circ \mathbb{E}$	`"reg:gamma"`	`"gamma"`
Tweedie loss	$\log \circ \mathbb{E}$	`"reg:tweedie"`	`"tweedie"`
Huber-like loss	M-estimator	`"reg:huber"`	`"huber"`
Cox neg. log-lik	$\log \circ$ hazard	`"survival:cox"`	-

Remark 1.
- *At the time of writing and until at least XGBoost version 1.4.1.1, XGBoost respects only non-overlapping interaction constraint sets $\mathcal{X}_j$, i.e., partitions. LightGBM can also deal with overlapping constraint sets.*
- *Boosted trees with interaction constraints support only non-linear components, unlike, e.g., deep learning and component-wise boosting, both of which allow a mix of linear and non-linear components. See Remark 4, as well as our two case studies, for a two-step procedure that would linearize some effects fitted by XGBoost or LightGBM, thus overcoming this limitation.*
- *Feature pre-processing for boosted trees is simpler than for other modeling techniques. Missing values are allowed for most implementations, feature outliers are unproblematic, and some implementations (including LightGBM) can directly deal with unordered categorical variables. Furthermore, highly correlated features are only problematic for interpretation, not for model fitting.*
- *Model-based boosting "mboost" with trees as building blocks is an alternative to using XGBoost or LightGBM with interaction constraints.*

2.3. STAR Models and Supervised Dimension Reduction

Since STAR models (including GAMs with two-dimensional interaction surfaces) typically use only a small subset of (interacting) features per component, the keyword "dimension reduction" rarely appears in connection with this type of model. However, a strength of tree-based components (e.g., trees as base learners in model-based boosting "mboost" or the approach via interaction constraints) and deep learning is that some components can use even a large number of interacting features. Such components serve as one-dimensional representations of their features, conditional on the other features. The values of such components (or sometimes of *sums* of multiple components) might be used as derived features in another model or analysis. Thus, STAR models offer an effective way to perform (one-dimensional) supervised dimension reduction. Note that, by *supervised*, we mean that the dimension reduction procedure uses the response variable of the model. Some examples illustrate this.

Example 1.

1. *House price models with additive effects for structural characteristics and time (for maximal interpretability) and one multivariate component using all locational variables with complex interactions (for maximal predictive performance). The model equation could be as follows:*

$$\mathbb{E}(\log(Price) \mid X = x) = \underbrace{f_1(x^{(1)}) + \cdots + f_{p-2}(x^{(p-2)})}_{\text{additive building characteristic effects}} \\ + \underbrace{f_{p-1}(x^{(p-1)})}_{\text{time effect}} + \underbrace{f_p(x^{(p)}, \ldots, x^{(q)})}_{\text{complex locational effects}} . \tag{4}$$

    Component f_p provides a one-dimensional representation of all locational variables. We will see an example of such a model in the Florida case study.
2. *This is similar to the first example, but adding the date of sale to the component with all locational variables, leading to a model with time-dependent location effects. The component depending on locational variables and time represents those variables by a one-dimensional function. Such a model will be shown in the Swiss case study below.*

In the above examples, the feature subsets $\mathcal{X}_j$ used by components f_j are non-overlapping, i.e., are partitions. For a STAR model f of the general form (1), where features might appear in multiple components, we can extend the above idea of dimensionality reduction.

Definition 1 (Purely additive contributions and encoders)**.** *Let $\mathcal{X}'$ be a feature subset of interest and $\hat{f}_j$ the fitted additive components of a STAR model*

$$\xi(Y \mid X = x) = f(x) = \sum f_j(x)$$

as in Equation (1). The contribution of $\mathcal{X}'$ to the predictor $\hat{f}(x)$ is given by the partial predictor

$$\hat{f}_{J(\mathcal{X}')}(x) := \sum_{j \in J(\mathcal{X}')} \hat{f}_j(x),$$

where $J(\mathcal{X}') := \{j : \mathcal{X}' \cap X_j \neq \varnothing\}$ denotes the index set of components using features in $\mathcal{X}'$. Furthermore, we call $\hat{f}_{J(\mathcal{X}')}(x)$ a purely additive contribution *or* encoder *of $\mathcal{X}'$ if $\bigcup_{J(\mathcal{X}')} \mathcal{X}_j \subseteq \mathcal{X}'$, i.e., if $\hat{f}_{J(\mathcal{X}')}(x)$ depends solely on features in $\mathcal{X}'$. In this case, we say that $\mathcal{X}'$ has a purely additive contribution or $\mathcal{X}'$ is encodable and write the encoder as $\hat{f}_{J(\mathcal{X}')}(x')$, where x' are values of the feature vector X' corresponding to $\mathcal{X}'$.*

Remark 2.

*   *From the above definition, it directly follows that the purely additive contribution $\hat{f}_{J(\mathcal{X}')}(x')$ of a (possibly large) set of features $\mathcal{X}'$ provides a supervised one-dimensional representation of the features in $\mathcal{X}'$, optimized for predictions on the scale of ξ and conditional on the effects of other features.*
*   *For simplicity, we assume that each model component is irreducible, i.e., it uses only as many features as necessary. In particular, a component additive in its features would be represented by multiple components instead.*
*   *The encoder $\hat{f}_{J(\mathcal{X}')}(x')$ of $\mathcal{X}'$ is defined up to a shift, i.e., $\hat{f}_{J(\mathcal{X}')}(x') + c$ for any constant c is an encoder of $\mathcal{X}'$ as well. If c is chosen so that the average value of the encoder is 0 on some reference dataset, e.g., the training dataset, we speak of a centered encoder.*

Example 2. *Consider the fitted STAR model*

$$\hat{f}(x) = \hat{f}_1(x^{(1)}, x^{(2)}) + \hat{f}_2(x^{(1)}, x^{(3)}) + \hat{f}_3(x^{(2)}, x^{(3)}) + \hat{f}_4(x^{(4)}, x^{(5)}, x^{(6)}).$$

Following the above definition, we can say:

1. *Let $\mathcal{X}' = \{X^{(1)}, X^{(2)}, X^{(3)}\}$. Then,*

$$\hat{f}_{J(\mathcal{X}')}(x^{(1)}, x^{(2)}, x^{(3)}) = \hat{f}_1(x^{(1)}, x^{(2)}) + \hat{f}_2(x^{(1)}, x^{(3)}) + \hat{f}_3(x^{(2)}, x^{(3)})$$

   is a purely additive contribution of $\mathcal{X}'$.

2. *Due to interaction effects, the singleton $\{X^{(1)}\}$ and $\{X^{(1)}, X^{(2)}\}$ are not encodable.*

3. *Let $\mathcal{X}' = \{X^{(4)}, X^{(5)}, X^{(6)}\}$ and $x' = (x^{(4)}, x^{(5)}, x^{(6)})$. Then,*

$$\hat{f}_{J(\mathcal{X}')}(x') = \hat{f}_4(x')$$

   is an encoder of $\mathcal{X}'$.

4. *The fitted model $\hat{f}(x)$ is an encoder of the set of all features. This is true for STAR models in general.*

   Next, we consider the fitted STAR model

$$\hat{f}(x) = \hat{f}_1(x^{(1)}) + \hat{f}_2(x^{(2)}) + \hat{f}_3(x^{(3)}, x^{(4)}, x^{(5)}).$$

Here, the feature sets $\{X^{(1)}\}$, $\{X^{(2)}\}$, and $\{X^{(3)}, X^{(4)}, X^{(5)}\}$ form a partition. As a direct consequence of Definition 1, each feature set (and all its possible unions) of a partition is encodable. This property will implicitly be used in both of our case studies.

Depending on the implementation of the STAR algorithm, it might be possible to directly extract the encoder of a feature set $\mathcal{X}'$ from the fitted model. A procedure that works independently of the implementation is described in the following Algorithm 1, which requires only access to the prediction function $\hat{f}(x)$ on the scale of ξ.

Algorithm 1 Encoder extraction

---

1. Assert that $\mathcal{X}'$ is encodable.

2. Let h be the mapping that extends x' to x, padding the remaining vector components by arbitrary but fixed values. As a direct consequence of Definition 1, the predictor

$$\hat{f}(h(x')) \tag{5}$$

   agrees with the encoder $\hat{f}_{J(\mathcal{X}')}(x')$ up to a shift and, thus, serves as an (uncentered) encoder of feature set $\mathcal{X}'$.

3. Depending on the application, the *centered* encoder

$$\hat{f}^c_{J(\mathcal{X}')}(x') := \hat{f}(h(x')) - c \tag{6}$$

   is easier to interpret, where

$$c = \sum_{i=1}^{n} w_i \hat{f}(h(x'_i)) \Big/ \sum_{i=1}^{n} w_i$$

   is the (possibly weighted) average value of the predictions $\hat{f}(h(x'_i))$ and x'_i, $i = 1, \ldots n$, denote observed values of x' (with optional case weights w_i) in some reference dataset, e.g., the training set.

Remark 3 (Raw scores and boosting). *By default, predictions of XGBoost and LightGBM are on the scale of Y. If the functional ξ includes a link function (e.g., log or logit), one can obtain predictions on the scale of ξ via the argument* `outputmargin` *(XGBoost) or* `rawscore` *(LightGBM) of their* `predict` *method. This is relevant for the application of Algorithm 1 and also for interpretation of effects, as in Section 2.4.*

The values of a (possibly centered) encoder of a feature set $\mathcal{X}'$ can be used as a one-dimensional representation of $\mathcal{X}'$ for subsequent analyses, e.g., as a derived covariate in a simplified regression model. In the case studies below, we will see that this approach produces models with an excellent trade-off between interpretability and predictive strength.

Example 3. *For instance, after fitting the STAR model (4) of Example 1 with XGBoost, we could extract the (centered) purely additive contribution $\hat{f}^c_{J(\mathcal{X}')}$ of all location covariates $\mathcal{X}'$ using Algorithm 1 and calculate a subsequent linear regression of the form*

$$\mathbb{E}(\log(Price) \mid X = x) = \beta_0 + \underbrace{\beta_1 x^{(1)} + \cdots + \beta_{p-2} x^{(p-2)}}_{building\ characteristic\ effects}$$

$$+ \underbrace{g(x^{(p-1)})}_{time\ effects} \tag{7}$$

$$+ \underbrace{\beta_p \hat{f}^c_{J(\mathcal{X}')}(x')}_{locational\ effects},$$

where $x' = (x^{(p)}, \ldots, x^{(q)})$ represent the values of the locational variables. The main difference with the initial XGBoost model is that the effects of the building characteristics are now linear.

Remark 4 (Modeling strategy). *The workflow in the last example is in line with the following general modeling strategy. Groups of related features (for instance, a large set of locational variables) are sometimes difficult to represent in a linear regression model. How should the features be transformed? What interactions are important? How should one deal with strong multicollinearity? These burdens could be delegated to an initial STAR model of suitable structure. Then, the model components representing such feature groups would be extracted and plugged as high-level features into a subsequent linear regression model. This allows one, e.g., to linearize some additive effects of a fitted boosted trees model with interaction constraints.*

Encoders are also helpful for interpreting effects in general STAR models, as will be explained in the next section.

2.4. Interpreting STAR Models

In this section, we provide some information on how to interpret effects of features and feature sets in general STAR models (1), with a special focus on simple, fully transparent descriptions. These techniques will be used later in the case studies.

One of the most common techniques to describe the effect of a feature set $\mathcal{X}'$ in an ML model is the partial dependence plot (PDP) introduced in Friedman (2001). It visualizes the average partial effect of $\mathcal{X}'$ by taking the average of many *individual conditional expectation* profiles (ICE, see Goldstein et al. 2015). The ICE profile for the ith observation with observed covariate vector x_i and feature set $\mathcal{X}'$ is calculated by evaluating predictions $\hat{f}(x)$ over a sufficiently fine grid of values for vector components corresponding to $\mathcal{X}'$, keeping all other vector components of x_i fixed. The stronger the interaction effects from other features, the less parallel the ICE profiles across multiple observations are. Thus, a visual test for additivity can be performed by plotting many ICE profiles and checking if they are parallel (see Goldstein et al. 2015). Conversely, if all ICE profiles of $\mathcal{X}'$ are parallel, $\mathcal{X}'$ is represented by the model in an additive way. In that case, a single ICE profile (or the

PDP) serves as a fully transparent description of the effect of $\mathcal{X}'$ in the sense that it is clear how $\mathcal{X}'$ acts on $\hat{f}(x)$ globally for all observations ceteris paribus.

Studying ICE and PDP plots is not only interesting for interpreting feature effects in complex ML models. Along with other general-purpose tools from the field of explainable ML (see, e.g., Biecek and Burzykowski 2021; Molnar 2019), they can also be used to interpret models of restricted complexity, such as linear regression models, GAMs, or STAR models. There, they serve as (feature-centric) alternatives to partial residual plots (Wood 2017) that are frequently used to visualize effects of *single model components*. Note that, up to a vertical shift, partial residual plots coincide with ICE curves for features that appear in only one (single-feature) component.

In the context of STAR models, from Definition 1, it is easy to see that, for a feature set $\mathcal{X}'$ that has a purely additive contribution, ICE profiles of $\mathcal{X}'$ are parallel across all observations, and their values correspond to values of the encoder $\hat{f}_{J(\mathcal{X}')}(x')$ evaluated over the domain of x' (up to an additive constant). This includes centered or uncentered encoders as extracted by Algorithm 1. In fact, Algorithm 1 differs from the calculation of ICE profiles only in terms of technical details.

Thus, the effects of an encodable feature set $\mathcal{X}'$ can be described in a simple yet transparent way by, e.g., showing one ICE profile, the PDP, or by evaluating its encoder over the domain of x'. This is a major advantage of STAR models over unstructured ML models, where transparent descriptions of feature effects are unrealistic due to complex high-order interactions.

However, since it is difficult to describe ICE/PDP/encoders of more than two features, this concept is limited from a practical perspective to feature sets with only one or two features.

Remark 5.

- *To benefit from additivity, effects of features in STAR models are typically interpreted on the scale of ξ.*
- *ICE profiles and the PDP can be used to interpret effects of non-encodable feature sets as well. Due to interaction effects, however, a single ICE profile or a PDP cannot give a complete picture of such an effect.*
- *In this paper, we describe feature effects by ICE/PDP/encoders. For alternatives, see Molnar (2019) or Biecek and Burzykowski (2021).*

We have mentioned that describing multivariable ICE/PDP/encoders of more than two features is difficult in practice. However, depending on the modeling situation, it is not uncommon that a possibly large, encodable feature set $\mathcal{X}'$ represents a low-dimensional feature set $\mathcal{X}''$ with mapping $X' = \phi(X'')$. Here, X' and X'' denote feature vectors corresponding to feature sets $\mathcal{X}'$ and $\mathcal{X}''$. In this case, the ICE/PDP/encoder can be evaluated over values of X'' to provide, again, a fully transparent description of the effects of X'' and X'. Thus, instead of using the encoder $\hat{f}^c_{J(\mathcal{X}')}(x')$ with high-dimensional x', we would use the equivalent encoder $\hat{f}^c_{J(\mathcal{X}')}(\phi(x''))$ with low-dimensional x'' to describe the effects of feature set $\mathcal{X}'$ (or $\mathcal{X}''$), even if we cannot exactly describe the effects of *single features* in $\mathcal{X}'$.

We have seen three such situations in Example 1: The full set of location variables can represent low-dimensional features, such as:

- The administrative unit (a one-dimensional feature).
- The address (also a one-dimensional feature).
- Latitude/longitude (a two-dimensional feature).

Similarly, time-dependent location effects could represent the two features "administrative unit" and "time".

We use this projection trick in both of our case studies in Sections 3 and 4 to describe multivariate location effects.

3. Case Study 1: Miami Housing Data

To apply the above methods, we consider two case studies involving hedonic house price models. Hedonic models are widely used in the context of real estate valuation, especially for residential properties (Malpezzi 2003). Such models apply information about a sample of properties that transacted to estimate models that are then used to predict the values of out-of-sample properties that did not transact. The dependent variable is the price of the properties and the independent variables are a set of physical and locational attributes. Hedonic models are a valuable tool for property tax appraisers, mortgage underwriters, valuation firms, and regulatory authorities. Popular online resources, such as Zillow.com in the United States, rely on such models to provide regularly updated estimates of property values. Various studies have successfully applied machine learning techniques to hedonic models (e.g., Mayer et al. 2019; Wei et al. 2022), but interpretability remains an issue.

3.1. Data

This case study uses information on 13,932 single-family homes sold at arm's length in Miami in 2016. The data are collected by county property tax appraisers and distributed for research purposes by the Florida Department of Revenue. Aside from sale price and time, the dataset contains five building characteristics and eight locational variables; see Table 3 for definitions of the variables and Figures 3–6 for a descriptive analysis.

Table 3. Variables used in the Miami case study.

| Variable Name	Symbol	Meaning
log_price	Y	Logarithmic sale price in USD
month_sold	T	Sale month in 2016 (1: January, . . . , 12: December)
log_living	$S^{(1)}$	Living area in logarithmic square feet
log_land	$S^{(2)}$	Land area in logarithmic square feet
log_special	$S^{(3)}$	Logarithmic value of special features (e.g., swimming pool)
age	$S^{(4)}$	Age of the structure when sold
structure_quality	$S^{(5)}$	Measure of building quality (1: worst, 5: best)
s_dist_highway	$R^{(1)}$	Square root of distance to nearest highway in 1000 ft
s_dist_rail	$R^{(2)}$	Square root of distance to nearest rail line in 1000 ft
dist_ocean	$R^{(3)}$	Distance to ocean in 1000 ft
log_dist_water	$R^{(4)}$	Logarithmic distance in 1000 ft to nearest body of water
s_dist_center	$R^{(5)}$	Square root of distance to central business district in 1000 ft
s_dist_sub	$R^{(6)}$	Square root of distance to nearest subcenter in 1000 ft
latitude	$R^{(7)}$	Latitude
longitude	$R^{(8)}$	Longitude
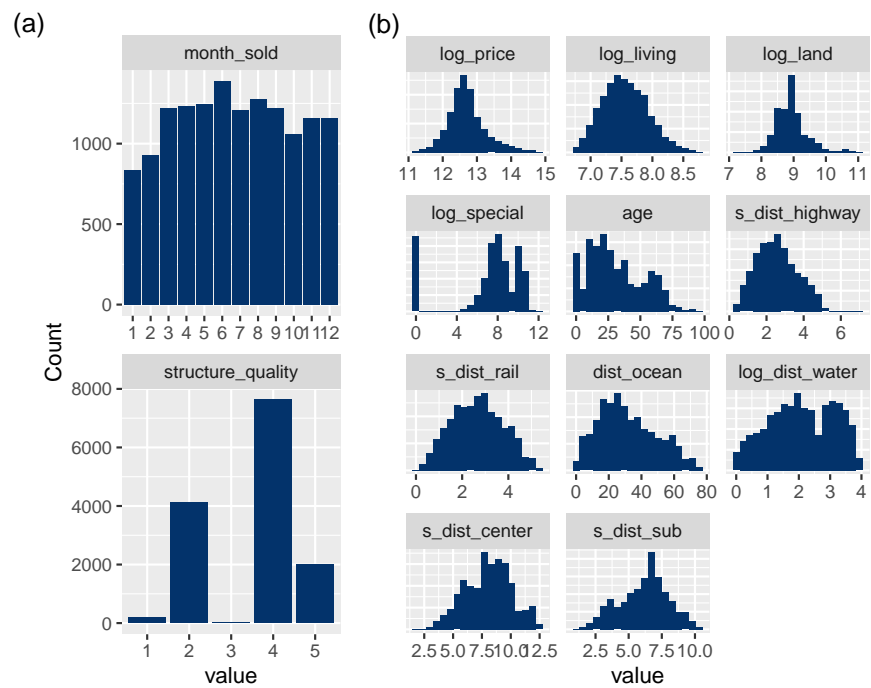
Figure 3. (**a**) Bar plots of discrete variables. (**b**) Histograms of transformed continuous variables, except latitude/longitude.

The set of locational variables corresponding to the feature vector $R = (R^{(1)}, \dots, R^{(8)})$ is denoted by $\mathcal{R}$. Furthermore, observed values of any feature or feature vector are denoted by lowercase letters, such as $s^{(1)}$ or r.
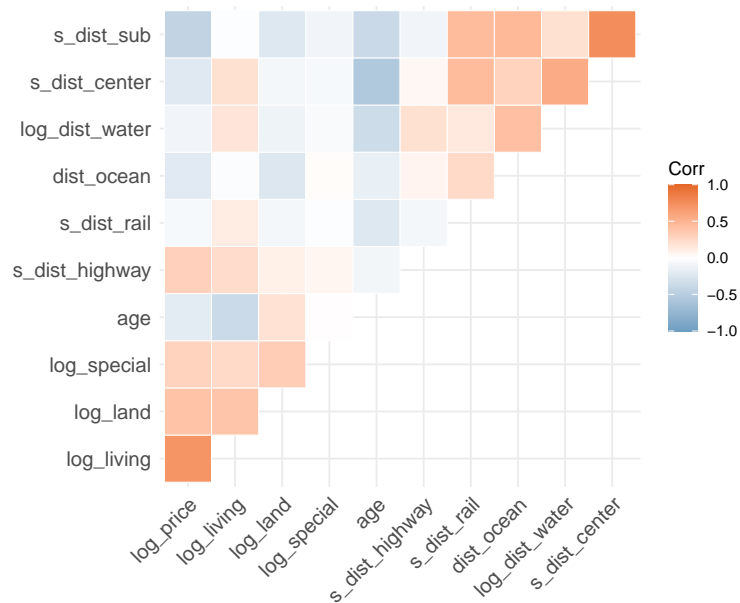
Figure 4. Correlogram of continuous variables (except latitude/longitude).
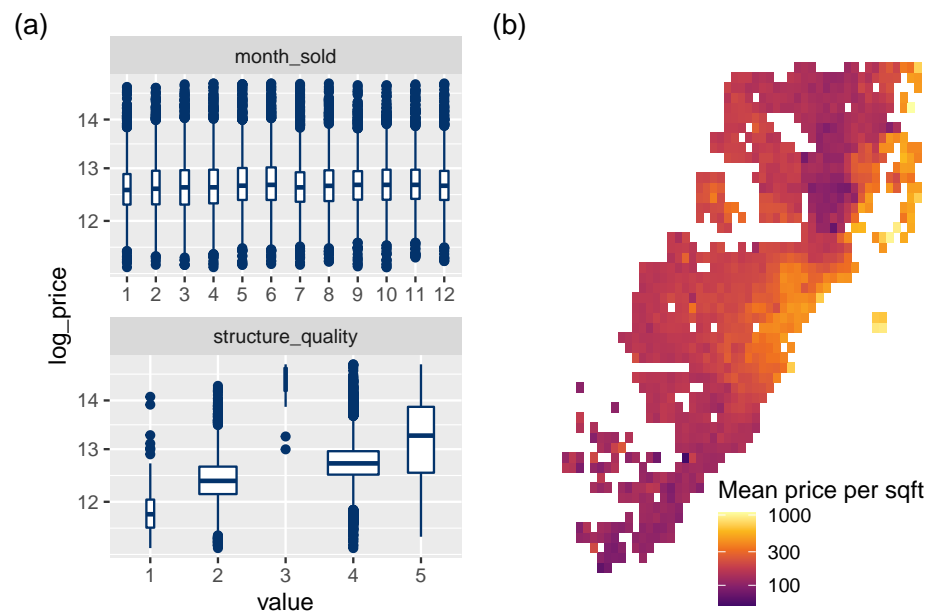
Figure 5. (**a**) Association between discrete variables and the response `log_price` (boxplots). (**b**) Logarithmic heatmap of price per living area.
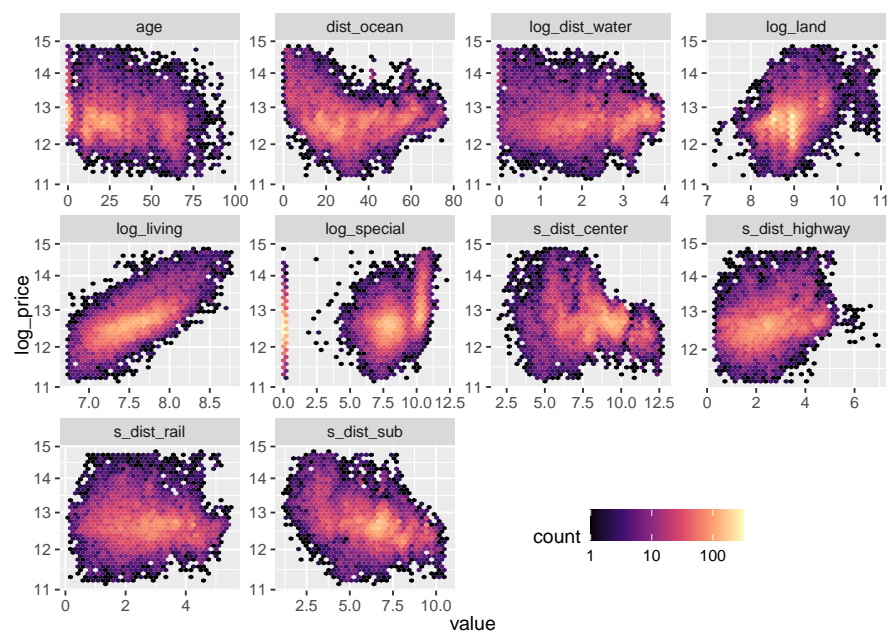
Figure 6. Logarithmic heatmaps between the response `log_price` and the continuous variables (except latitude and longitude).

3.2. Models

To illustrate the methods discussed in the previous sections, we consider six models:

1. OLS: Linear regression benchmark with maximal interpretability.
2. XGB (unconstrained): XGBoost benchmark for maximal performance.
3. XGB STAR: XGBoost STAR model for high interpretability and performance.
4. OLS with XGB loc: Linear model with location effects derived from XGB STAR.
5. NN STAR: Deep neural STAR model.
6. mboost STAR: mboost STAR model.

Details for each model are given below. Before fitting the models, the dataset was randomly split into 80% training and 20% test observations. The test dataset was used solely for evaluating the performance of the models, not for tuning them.

As a simple benchmark with maximal interpretability, we consider the OLS regression with time dummy effects g

$$\mathbb{E}(Y) = \beta_0 + \beta_1 s^{(1)} + \cdots + \beta_5 s^{(5)} + \beta_6 r^{(1)} + \cdots + \beta_{11} r^{(6)} + g(t)$$

fitted with R function `lm()`. This model does not use latitude/longitude, as there is no obvious way to represent them in this case.

To get a rough benchmark of the maximally achievable model performance, we consider the unconstrained XGBoost model, XGB (unconstrained), with learning rate 0.1. Its most relevant hyper-parameters (including tree depth, ridge and lasso regularization, and column and row subsampling) were tuned on the training data by a randomized search using five-fold cross-validation. A suitable number of trees (469) was chosen by early stopping on the cross-validation score. After tuning, the model was refitted on the full training data. The calculations were made with the R package `xgboost` (Chen et al. 2021) using squared error loss.

The next model considered (XGB STAR) is a STAR model fitted with XGBoost and interaction constraints to enforce time and building characteristic effects to be additive. Only the locational factors are allowed to interact with each other. The structured model equation follows the form of Equation (1):

$$\mathbb{E}(Y) = f_1(s^{(1)}) + \cdots + f_5(s^{(5)}) + f_6(r) + f_7(t).$$

In this way, the feature sets corresponding to the model components form a partition, which holds for all STAR models used in our case studies. Using the same learning rate and tuning strategy as the unconstrained XGBoost model, the cross-validated optimal number of trees (1583) was clearly higher to compensate for the additivity constraints.

From the XGB STAR model, we extracted the centered encoder $\hat{f}^c_{J(\mathcal{R})}(r)$ using Algorithm 1. The values of this function represent the combined effect of all location variables, which we plugged as a derived feature into the OLS with XGB loc regression:

$$\mathbb{E}(Y) = \beta_0 + \beta_1 s^{(1)} + \cdots + \beta_5 s^{(5)} + \beta_6 \hat{f}^c_{J(\mathcal{R})}(r) + g(t),$$

as outlined in Remark 4. This model can be viewed as a smoothed, linearized version of the XGB STAR model.

The remaining two models show alternative ways to fit a STAR model. Like the OLS with XGB loc model, they are constrained to have linear effects for the building characteristics. The first of these models is a neural network (NN STAR) with the equation

$$\mathbb{E}(Y) = \beta_0 + \beta_1 s^{(1)} + \cdots + \beta_5 s^{(5)} + \underbrace{\beta_6 f'(r)}_{f(r)} + \underbrace{\beta_7 g'(t)}_{g(t)},$$

which is depicted in Figure 7. The architecture of the location sub-network (three hidden layers having 12-6-3 non-intercept nodes and hyperbolic tangent activations), the learning rate (0.003), and the number of epochs (105) were chosen by simple validation on the training data. As a loss function, the squared error was used. The model has 231 parameters, most of which are used to represent the location sub-network. Time is represented by an embedding layer. We used the R package `keras` (Allaire and Chollet 2021) to fit the model via TensorFlow. This model serves as alternative to XGB STAR, but is not expected to be as accurate for a sample of this size. Following the recommendation in LeCun et al. (2012), we scaled numeric input features to a common scale in order to improve the convergence properties.
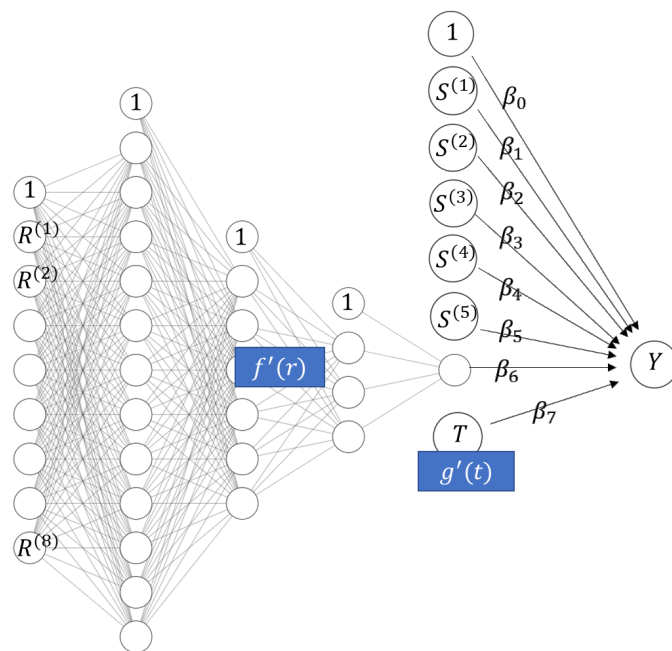
Figure 7. Architecture of the neural network STAR model of Miami house prices.

Remark 6. *Neural networks are often biased in the sense that their average prediction on the training data is different from the corresponding average response; see Wüthrich (2020) for ways to address this. We skipped this calibration step for simplicity.*

The last model considered (mboost STAR) is a model-based booster fitted with the R package `mboost` (Hothorn et al. 2021):

$$\mathbb{E}(Y) = \beta_0 + \beta_1 s^{(1)} + \cdots + \beta_5 s^{(5)} + f(r) + g(t).$$

Per boosting round, the locational variables *R* are represented by a decision tree, while the other variables are fitted by linear functions via unpenalized least squares (time *T* is internally dummy encoded). The main tuning parameter, the number of boosting rounds, was chosen by five-fold cross-validation and amounted to 1022 rounds at a learning rate of 0.1. The tree depth was set to five in order to allow for a flexible representation of the location variables comparable to the XGBoost models.

3.3. Results

The models were evaluated on the 20% test data and interpreted using the `flashlight` package (Mayer 2021).

3.3.1. Performance

Model performance is measured by out-of-sample root mean square error and R-squared (Table 4). The constrained, highly structured XGBoost STAR model performs almost as well as the unconstrained, fully flexible XGBoost model, which is remarkable: By sacrificing only 0.2 percentage points of out-of-sample R-squared, we were able to create a transparent model (see figures below), avoiding the typical trade-off between performance and interpretability. The OLS with XGB loc model provides a model that is even simpler to interpret, here at the cost of one percentage point of R-squared. The other STAR models perform worse, but, as usual, such comparisons must be undertaken with care due to the different tuning techniques and different structures of the model components. Thus, we purposely do not show the p values for pairwise comparisons of performance.

Table 4. RMSE and R-squared evaluated on the Miami hold-out dataset.

| Model	RMSE	R-Squared
OLS	0.252	0.799
XGB (unconstrained)	0.132	0.945
XGB STAR	0.134	0.943
OLS with XGB loc	0.145	0.933
NN STAR	0.182	0.895
mboost STAR	0.159	0.920

3.3.2. Effects of Single Variables

As outlined above, feature effects in general ML models can be described by ICE curves (Goldstein et al. 2015). Figure 8 shows such curves for the living area. One curve explains how the predicted logarithmic price of a selected house changes for different values of the living area, keeping all other feature values fixed. As expected, the curves are parallel for all models that are additive in the living area, which is the case for all models except the unconstrained XGBoost model. For these models, the effect of the living area can thus be described in a fully transparent way by studying a single ICE curve. The same holds for each other building characteristic and time.

For the mboost model, the deep neural net, and the two linear regressions, the curves are linear, leading to maximum interpretability. In these cases, the effect of the living area could be described by a single number, namely, the slope of any ICE curve, which equals the corresponding model coefficient.
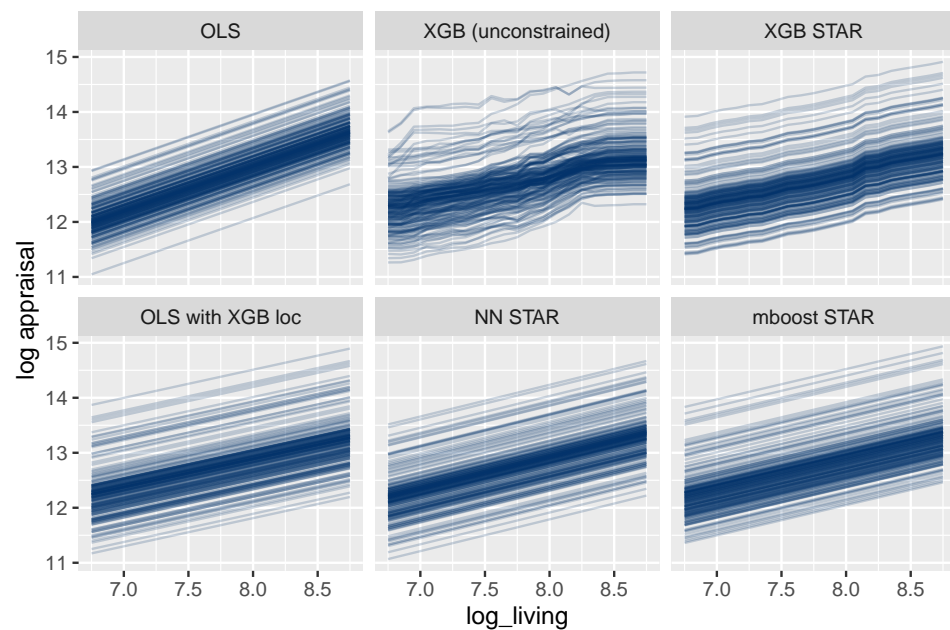
Figure 8. ICE curves for `log_living`. All curves are parallel and simple to interpret, except for the unconstrained XGBoost model.

A completely different picture emerges for `log_dist_water` (Figure 9). The ICE curves of this location variable are parallel only for the oversimplified OLS model. The other models allow this feature to interact with other features, leading to non-parallel curves. Using the terminology discussed above, the feature set {`log_dist_water`} is not encodable for these models. Consequently, a single ICE curve or a PDP cannot provide a complete description of its modeled effects. We will, however, show that the *combined effect* of all location features can be described transparently by a single visualization.
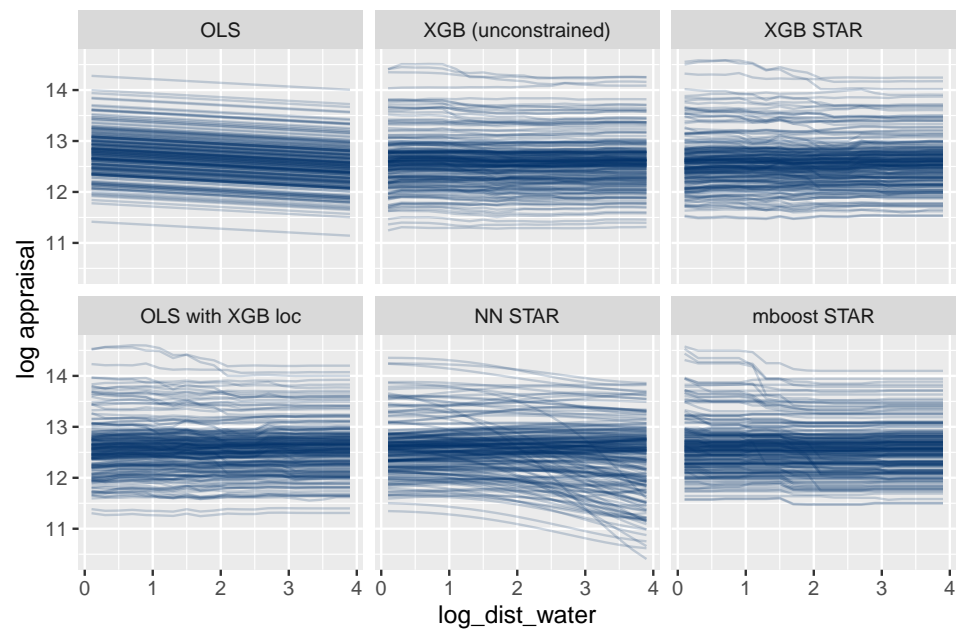
Figure 9. ICE curves for the locational variable `log_dist_water` show non-additive effects, except for the OLS regression.

3.3.3. Interaction Effects

The field of explainable ML offers different ways to study interaction effects; for instance, multidimensional ICE curves or Friedman's H measure of interaction strength (Friedman and Popescu 2008), which we consider here. Figure 10 shows H for selected variable pairs. A value of 0 means that there are no interactions, while higher values represent stronger interactions relative to the two corresponding main effects. The results in Figure 10 suggest that all STAR models have been correctly specified. For instance, the interaction strength between the living area and latitude is 0 for all models, except for the unconstrained XGBoost model.
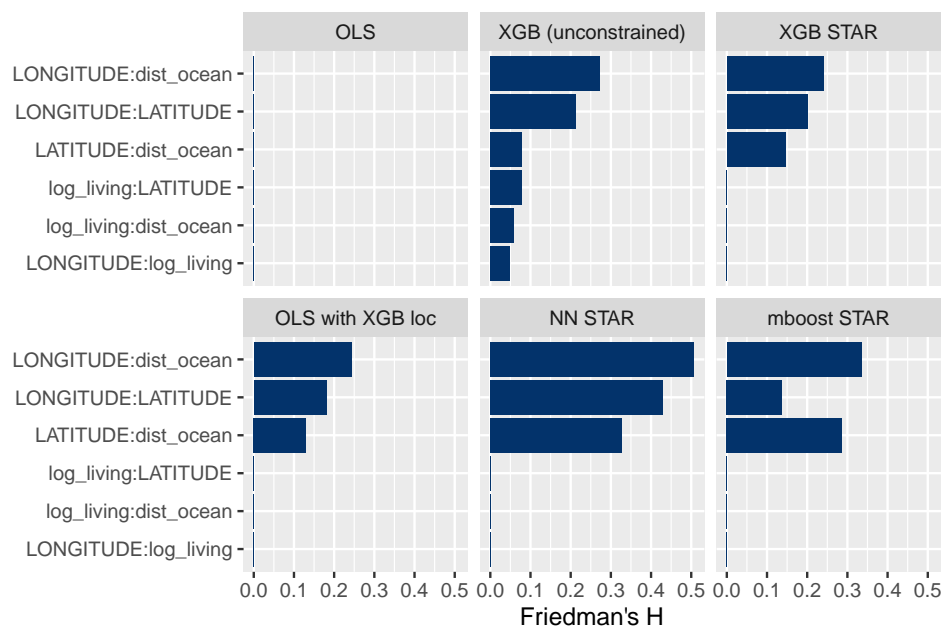

Figure 10. Friedman's H measure of interaction strength for selected variable pairs. The STAR models appear to be correctly specified.

3.3.4. Supervised Dimension Reduction and Location Effects

As noted in the discussion of the methodology, STAR models can sometimes be used for supervised dimensionality reduction. For the STAR models in this case study, this is done by representing all location variables R with a single model component. Once fitted, the values of that component serve as a one-dimensional representation of the set of locational variables. We used Algorithm 1 to extract centered encoders of R from each of the three explicit STAR models. To illustrate the workflow of Remark 4, the encoder of XGBoost STAR was subsequently plugged as a derived feature into the simplified OLS with XGB loc model, with excellent results (see above).

Furthermore, since there is a trivial mapping from the two features of latitude and longitude to the encodable set of location features $\mathcal{R}$, we were able to use the projection trick described above to provide a fully transparent description of the combined location effects. To do so, we visualized as heatmaps the values of the centered encoders of the three STAR models (XGB STAR, NN STAR, and mboost STAR) over all latitude/longitude pairs appearing in the full dataset (Figure 11).

Since the model response is logarithmic, the values of the centered encoders can be interpreted as relative location effects. For example, a location with value 0.2 would be about 20% above the average price level of Miami. The results in Figure 11 look highly plausible. Due to its superior performance, the relative price map produced by the XGBoost STAR model shows slightly more pronounced effects, e.g., very high prices for the exclusive area Key Biscayne.
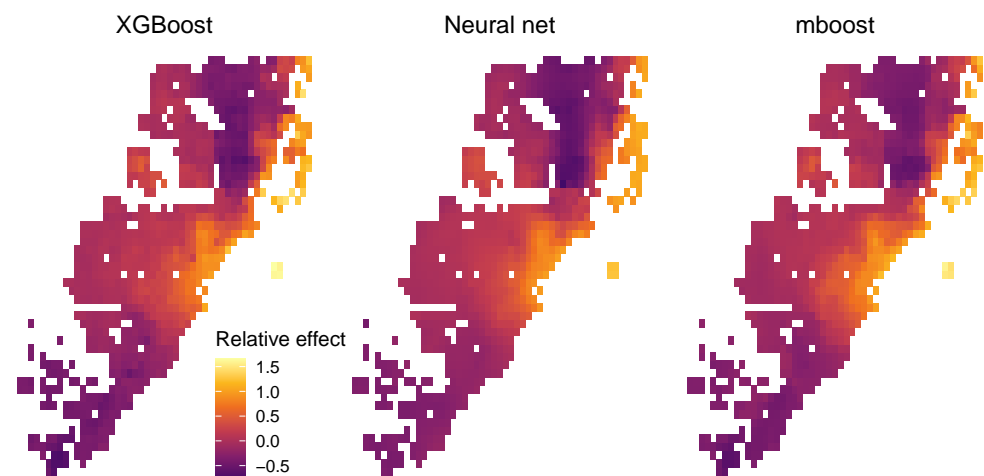
Figure 11. Relative location effects extracted from the three STAR models: XGB STAR, NN STAR, and mboost STAR.

4. Case Study 2: Swiss Housing Data

4.1. Data

Our second case study uses a rich geocoded dataset covering over 67,000 houses sold at arm's length between 2010 and 2021 in Switzerland. It was provided by the Informations- und Ausbildungszentrum für Immobilien AG (IAZI), a Swiss company specializing in property valuation. The median house has 5.5 rooms, 150 m^2 of living area, and a lot size of 576 m^2, and was 34 years old at transaction time. For confidentiality reasons, the sales prices were multiplied by a fixed, unknown constant.

The aim of this second case study is to derive models for the *median* logarithmic sales price, conditional on the following 90 features:

- Twelve pre-transformed property characteristics (logarithmic living area, number of rooms, age, etc.) represented by the feature vector $S = (S^{(1)}, \ldots, S^{(12)})$.
- Transaction time (year and quarter) T.

- A set of 25 untransformed locational variables mostly available at the municipal level, such as the average taxable income or the vacancy rate.
- Latitude and longitude.
- For 25 points of interest (POIs), such as schools, shops, or public transport, their individual counts within a 500 m buffer.
- The distance from the nearest POI of each type to the house (maximum 500 m).

The feature set containing time T and all 77 locational variables is denoted as $\mathcal{R}$ and the corresponding feature vector as R. As usual, we denote observed values of features by lowercase letters.

4.2. Models

Our house price models have the structure

$$\text{Median}(Y \mid S = s, R = r) = f(s, r),$$

where the response Y equals the logarithmic sales price. More specifically, we consider these three models:

1. LGB (unconstrained): LightGBM benchmark for maximum performance. Hyper-parameter tuning was done by stratified five-fold cross-validation, using a learning rate of 0.1 and systematically varying the number of leaves, row and column subsampling rates, and L1 and L2 penalization. As for the XGBoost models in the first case study, the number of trees (1918) was chosen by early stopping on the cross-validation score. The model was fitted with the R package `lightgbm` (Ke et al. 2021), using the absolute error loss in order to provide a model for the conditional *median* (Table 2).

2. LGB STAR: LightGBM STAR model for high interpretability and performance. This is built similarly to the first model (with 2225 trees), but with additive effects for building variables enforced by interaction constraints. Locational variables and transaction time were allowed to interact with each other. This leads to a model of the form

$$\text{Median}(Y) = \sum_{j=1}^{12} f_j(s^{(j)}) + f_{13}(r).$$

3. QR with LGB loc: Linear quantile regression (Koenker 2005) with time-dependent location effects derived from LGB STAR model, using Algorithm 1 to extract the centered encoder $\hat{f}^c_{J(\mathcal{R})}$. The model equation is thus

$$\text{Median}(Y) = \beta_0 + \sum_{j=1}^{12} \beta_j(s^{(j)}) + \beta_{13}\hat{f}^c_{J(\mathcal{R})}(r)$$

and can be viewed as a linearized LightGBM quantile regression model. The model was fitted with the R package `quantreg` (Koenker 2021).

For the sake of brevity, we do not consider deep learning and mboost in this case study. The models were tuned and trained on 80% of all observations and evaluated on the remaining 20%. The data split and the cross-validation folds (for tuning LightGBM) were created by stratification on ten quantile groups of the response variable to improve balance across subsamples.

4.3. Results

4.3.1. Performance

As in the first case study, the model performance is evaluated on the independent test dataset. In line with the aim of modeling a conditional *median*, the relevant performance measure will now be the mean absolute error (MAE), while the RMSE is added for information only. Table 5 shows that the constrained, highly interpretable LightGBM STAR model

performs only slightly worse than the unconstrained model. The loss in performance of the linear quantile regression with the LightGBM encoder is larger, but still surprisingly small. It could be improved by adding non-linear terms. An MAE of 0.168 implies, thanks to the logarithmic response, that the average relative appraisal error of the LightGBM STAR model is about 16.8%.

Table 5. MAE and RMSE evaluated on the Swiss hold-out dataset.

Model	MAE	RMSE
LGB (unconstrained)	0.165	0.255
LGB STAR	0.168	0.259
QR with LGB loc	0.174	0.266

### 4.3.2. Effects of Additive and Non-Additive Features

Figure 12 shows ICE curves of the logarithmic surface area for a small random sample of properties. As expected, the unconstrained model shows non-parallel curves, and thus, its feature effects are difficult to interpret. In contrast to this behavior, the curves for the LightGBM STAR model are intrinsically parallel, leading to a fully transparent description of the effect. The corresponding effect in the linear quantile regression can be described by a single number, namely, the slope of the lines.
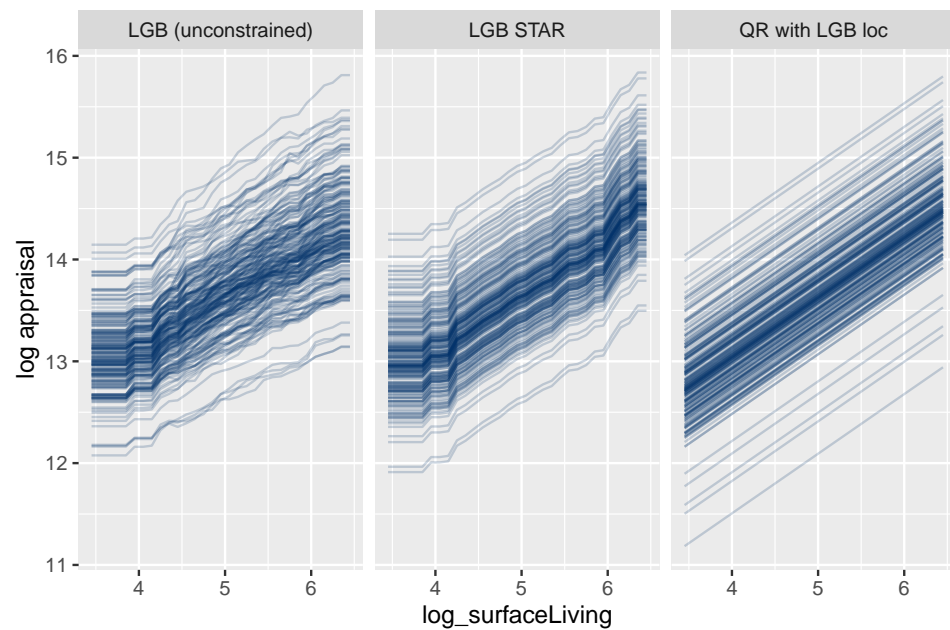
Figure 12. ICE curves for logarithmic living area.

On the other hand, the effect of the transaction time T is non-additive for all three models due to interaction effects with location (Figure 13).
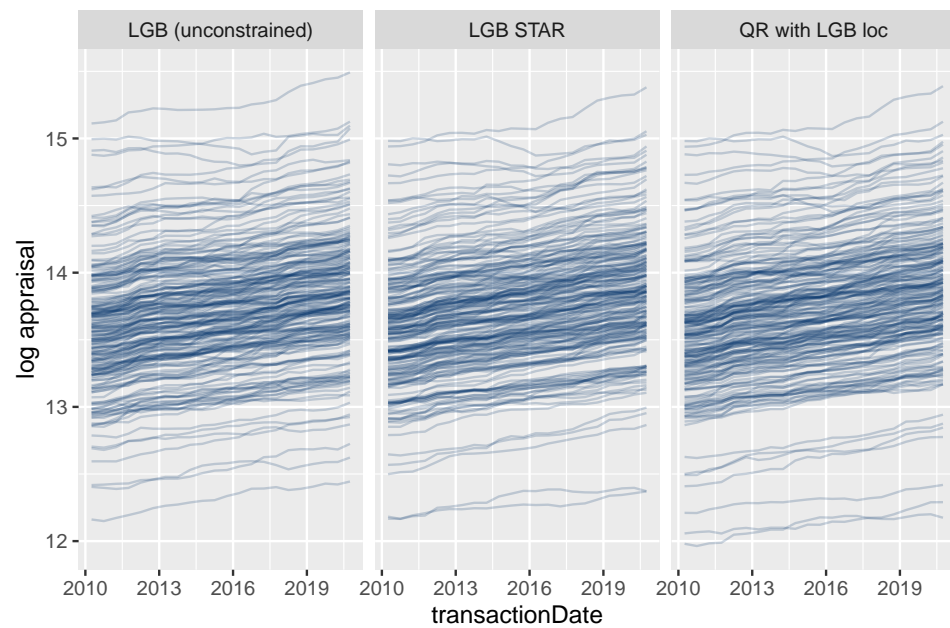
Figure 13. ICE curves for the date of transaction.

4.3.3. Extracting Time-Dependent Location Effects

In this case study, the centered encoder $\hat{f}^c_{J(\mathcal{R})}$ serves as a one-dimensional representation of both time and locational effects. In the QR with LGB loc model, we followed the workflow of Remark 4 and plugged the encoder as a derived feature into a simple-to-interpret linear model while retaining complex time-varying locational effects.

Furthermore, since the large feature set $\mathcal{R}$, consisting of 77 locational features plus time T, has the natural low-dimensional representation $(T, \text{Latitude}, \text{Longitude})$, we can use the projection trick discussed above to visualize the combined effect of R. To do so, we plot heatmaps of the centered encoder over all unique combinations of latitude and longitude and for different values of T. Figure 14 shows realistic relative price maps with the usual hotspots of Zurich, Geneva, and St. Moritz. An increasing price trend is clearly visible. Due to interaction effects between time and location, the time trend varies across locations.
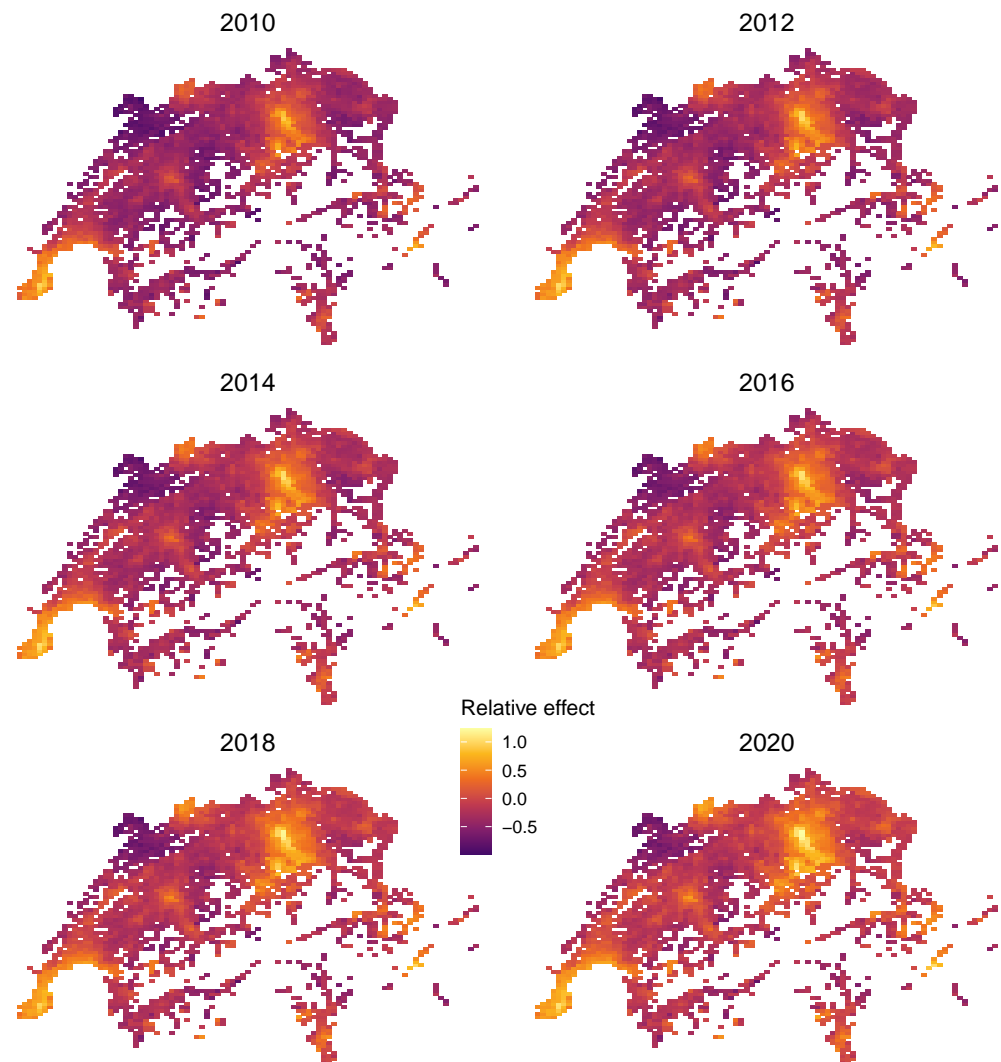
Figure 14. Time-dependent relative location effects in Switzerland extracted from the LightGBM STAR model.

Remark 7 (Regional hedonic indices). *In Figure 14, we visualized regional effects for selected time points. Similarly, it would be straightforward to visualize time effects for selected addresses. In fact, each ICE curve in Figure 13 shows an address-specific time effect. Regional indices at some administrative unit (e.g., the canton) could be created by a STAR model using additive components for the object characteristics, one component for all regional effects, and one component for canton and time.*

5. Discussion and Conclusions

We have shown that, thanks to interaction constraints, the two high-performance gradient boosting libraries XGBoost and LightGBM can be used to create highly intelligible and accurate GAM and STAR models. In contrast to traditional ways of fitting such models, tree-based additive components can use large groups of features. The combined effects of such groups can be extracted as derived one-dimensional features and used in subsequent analyses, such as in a simplified linear regression model.

This paper is the first to apply these techniques to house price valuation, where a large group of location features that contribute to land value can be represented with maximum

flexibility while restricting the effects of each building characteristic to be additive for full transparency.

Our case studies illustrate the usefulness of these techniques, as well as their flexibility. In the Miami case study, we found that the XGBoost methods performed very well with respect to predictive accuracy, with a constrained model that allowed for some transparency performing nearly as well as an unconstrained model that was completely opaque. Moreover, an OLS model with a single location variable derived from the constrained XGBoost model also performed quite well. In the Swiss case study, the highly interpretable Light-GBM STAR (constrained) model performed almost as well as the unconstrained LightGBM model. A linear regression model with a single location variable derived from the constrained model also performed well. In general, these approaches combine two desirable features: high predictive accuracy and interpretability for selected variables.

A further application in real estate modeling would be to add high-dimensional numeric information extracted from text and/or images—just like a group of locational features—to the model, without reducing the interpretability of the other covariates. The techniques are not only relevant in situations involving housing data. Consider, for example, automobile insurance pricing models, where one could represent driver characteristics in an additive and fully transparent way while letting characteristics of the car interact freely with each other. Finally, further research is needed to identify applications where structured deep learning would outperform gradient boosting with interaction constraints.

References

Abadi, Martin, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, and et al. 2016. Tensorflow: A system for large-scale machine learning. Paper presented at 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), Savannah, GA, USA, November 2–4; pp. 265–83. [CrossRef]

Agarwal, Rishabh, Levi Melnick, Nicholas Frosst, Xuezhou Zhang, Ben Lengerich, Rich Caruana, and Geoffrey E. Hinton. 2021. Neural additive models: Interpretable machine learning with neural nets. *Advances in Neural Information Processing Systems* 34.

Allaire, Joseph J., and François Chollet. 2021. *Keras: R Interface to 'Keras'*. R Package Version 2.4.0. Available online: https://CRAN.R-project.org/package=keras (accessed on 1 June 2021).

Arik, Sercan Ömer, and Tomas Pfister. 2019. Tabnet: Attentive interpretable tabular learning. *arXiv*, arXiv:2004.13912

Bühlmann, Peter, and Torsten Hothorn. 2007. Boosting Algorithms: Regularization, Prediction and Model Fitting. *Statistical Science* 22: 477–505. [CrossRef]

Biecek, Przemyslaw, and Tomasz Burzykowski. 2021. *Explanatory Model Analysis*. New York: Chapman and Hall/CRC. [CrossRef]

Chen, Tianqi, and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. Paper presented at 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, San Francisco, CA, USA, August 13–17. New York: Association for Computing Machinery, pp. 785–94. [CrossRef]

Chen, Tianqi, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, Rory Mitchell, Ignacio Cano, Tianyi Zhou, and et al. 2021. *Xgboost: Extreme Gradient Boosting*. R Package Version 1.4.1.1. Available online: https://CRAN.R-project.org/package=xgboost (accessed on 1 June 2021).

Din, Allan, Martin Hoesli, and André Bender. 2001. Environmental variables and real estate prices. *Urban Studies* 38: 1989–2000. [CrossRef]

Fahrmeir, Ludwig, Thomas Kneib, Stefan Lang, and Brian Marx. 2013. *Regression: Models, Methods and Applications*. Berlin: Springer. [CrossRef]

Friedman, Jerome H. 2001. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics* 29: 1189–232. [CrossRef]

Friedman, Jerome H., and Bogdan E. Popescu. 2008. Predictive learning via rule ensembles. *The Annals of Applied Statistics* 2: 916–54. [CrossRef]

Goldstein, Alex, Adam Kapelner, Justin Bleich, and Emil Pitkin. 2015. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics* 24: 44–65. [CrossRef]

Hastie, Trevor, and Robert Tibshirani. 1986. Generalized Additive Models. *Statistical Science* 1: 297–310. [CrossRef]

Hastie, Trevor, and Robert Tibshirani. 1990. *Generalized Additive Models*. Hoboken: Wiley Online Library.

Hothorn, Torsten, Peter Bühlmann, Thomas Kneib, Matthias Schmid, and Benjamin Hofner. 2021. *Mboost: Model-Based Boosting*. R Package Version 2.9-5. Available online: https://CRAN.R-project.org/package=mboost (accessed on 5 July 2021).

Hothorn, Torsten, Peter Bühlmann, Thomas Kneib, Matthias Schmid, and Benjamin Hofner. 2010. Model-based boosting 2.0. *The Journal of Machine Learning Research* 11: 2109–13.

Kagie, Martijn, and Michiel Van Wezel. 2007. Hedonic price models and indices based on boosting applied to the dutch housing market. *Intelligent Systems in Accounting, Finance & Management: International Journal* 15: 85–106. [CrossRef]

Ke, Guolin, Damien Soukhavong, James Lamb, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu, and et al. 2021. *Lightgbm: Light Gradient Boosting Machine*. R Package Version 3.2.1.99. Available online: https://github.com/microsoft/LightGBM (accessed on 13 August 2021).

Ke, Guolin, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*. New York: Curran Associates, Inc., Volume 30, pp. 3149–57.

Koenker, Roger. 2005. *Quantile Regression*. Econometric Society Monographs. Cambridge: Cambridge University Press. [CrossRef]

Koenker, Roger. 2021. *Quantreg: Quantile Regression*. R Package Version 5.86. Available online: https://CRAN.R-project.org/package=quantreg (accessed on 13 August 2021).

LeCun, Yann, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. 2012. Efficient backprop. In *Neural Networks: Tricks of the Trade*, 2nd ed. Edited by G. Montavon, G. B. Orr and K.-R. Müller. Volume 7700 of Lecture Notes in Computer Science. New York: Springer; pp. 9–48. [CrossRef]

Lee, Simon C. K., Sheldon Lin, and Katrien Antonio. 2015. *Delta Boosting Machine and Its Application in Actuarial Modeling*. Sydney: Institute of Actuaries of Australia.

Lou, Yin, Rich Caruana, and Johannes Gehrke. 2012. Intelligible models for classification and regression. Paper presented at 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12–16. New York: Association for Computing Machinery, pp. 150–58. [CrossRef]

Malpezzi, Stephen. 2003. *Hedonic Pricing Models: A Selective and Applied Review*. Hoboken: John Wiley & Sons, Ltd., Chapter 5, pp. 67–89. [CrossRef]

Mayer, Michael. 2020. Github Issue. Available online: https://github.com/microsoft/LightGBM/issues/2884 (accessed on 6 March 2009).

Mayer, Michael. 2021. *Flashlight: Shed Light on Black Box Machine Learning Models*. R package version 0.8.0. Available online: https://CRAN.R-project.org/package=flashlight (accessed on 1 June 2021).

Mayer, Michael, Steven C. Bourassa, Martin Hoesli, and Donato Scognamiglio. 2019. Estimation and updating methods for hedonic valuation. *Journal of European Real Estate Research* 12: 134–50. [CrossRef]

Molnar, Christoph. 2019. *Interpretable Machine Learning*. Available online: https://christophm.github.io/interpretable-ml-book (accessed on 1 July 2021).

Nelder, John Ashworth, and Robert W. M. Wedderburn. 1972. Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)* 135: 370–84. [CrossRef]

Nori, Harsha, Samuel Jenkins, Paul Koch, and Rich Caruana. 2019. Interpretml: A unified framework for machine learning interpretability. *arXiv*, arXiv:1909.00922.

Prokhorenkova, Liudmila, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. Catboost: Unbiased boosting with categorical features. Paper presented at the 32nd International Conference on Neural Information Processing Systems, NIPS'18, Montréal, QC, Canada, December 3–8. New York: Curran Associates Inc., pp. 6639–49.

Rügamer, David, Chris Kolb, and Nadja Klein. 2021. Semi-structured deep distributional regression: Combining structured additive models and deep learning. *arXiv*, arXiv:2002.05777.

Sangani, Darshan, Kelby Erickson, and Mohammad al Hasan. 2017. Predicting zillow estimation error using linear regression and gradient boosting. Paper presented at the 2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS), Orlando, FL, USA, October 22–25; pp. 530–34. [CrossRef]

Umlauf, Nikolaus, Daniel Adler, Thomas Kneib, Stefan Lang, and Achim Zeileis. 2012. *Structured Additive Regression Models: An R Interface to BayesX*. Working Papers 2012–10. Innsbruck: Faculty of Economics and Statistics, University of Innsbruck.

Wei, Cankun, Meichen Fu, Li Wang, Hanbing Yang, Feng Tang, and Yuqing Xiong. 2022. The research development of hedonic price model-based real estate appraisal in the era of big data. *Land* 11: 334. [CrossRef]

Wood, Simon N. 2017. *Generalized Additive Models: An Introduction with R*, 2nd ed. Boca Raton: CRC Press. [CrossRef]

Worzala, Elaine, Margarita Lenk, and Ana Silva. 1995. An exploration of neural networks and its application to real estate valuation. *Journal of Real Estate Research* 10: 185–201. [CrossRef]

Wüthrich, Mario V. 2020. Bias regularization in neural network models for general insurance pricing. *European Actuarial Journal* 10: 179–202. [CrossRef]

Yoo, Sanglim, Jungho Im, and John E. Wagner. 2012. Variable selection for hedonic model using machine learning approaches: A case study in Onondaga County, NY. *Landscape and Urban Planning* 107: 293–306. [CrossRef]

Zurada, Jozef, Alan S. Levitan, and Jian Guan. 2011. A comparison of regression and artificial intelligence methods in a mass appraisal context. *Journal of Real Estate Research* 33: 349–88. [CrossRef]