Locally Interpretable Tree Boosting: An Application to House Price Prediction

Anders Hjort[a,b,*], Ida Scheel[a], Dag Einar Sommervoll[b,c,d], Johan Pensar[a]

[a] *Department of Mathematics, University of Oslo*
[b] *Eiendomsverdi AS*
[c] *School of Economics and Business, Norwegian University of Life Sciences (NMBU)*
[d] *NTNU Trondheim Business School*

Abstract

We introduce Locally Interpretable Tree Boosting (LitBoost), a tree boosting model tailored to applications where the data comes from several heterogeneous yet known groups with a limited number of observations per group. LitBoost constraints the complexity of a Gradient Boosted Trees model in a way that allows us to express the final model as a set of local Generalized Additive Models, yielding significant interpretability benefits while still maintaining some of the predictive power of a Gradient Boosted Trees model. We use house price prediction as a motivating example and demonstrate the performance of LitBoost on a data set of $N = 14\,382$ observations from 15 different city districts in Oslo (Norway). We also test the robustness of LitBoost in an extensive simulation study on a synthetic data set.

Keywords: Gradient Boosted Trees, Generalized Additive Models, Explainable Boosting Machines, Interpretable Machine Learning, House Price Prediction

1. Introduction

The Gradient Boosted Trees (GBT) model has become a popular model choice among statisticians and data science practitioners due to its record of producing predictions with impressive accuracy in a wide variety of applications (Freund and Schapire (1996), Chen and Guestrin (2016)). Despite its predictive accuracy, it can often be challenging to apply the model in real-world applications where interpretability is essential due to the black-box nature of the model. Explaining or interpreting decisions from prediction models can be particularly important in high-stakes decisions within fields such as medicine, finance, or criminal justice. Using explainable models has become even more critical in recent years in the EU after the introduction of the General Data Protection Regulation (GDPR) laws, which according to Goodman and Flaxman (2017), induces a "right to explanation" to decisions made by algorithms of significant importance. Coussement

*Corresponding author.
Visiting Address: Moltke Moes vei 35, Niels Henrik Abels hus, 0851 Oslo.
Postal Address: Postboks 1053 Blindern, 0316 Oslo.
Email: anderdh@math.uio.no.
Telephone: +47 995 82 778.

and Benoit (2021) give a review of the challenges in applying data science in decision support systems, and conclude that the "widespread adoption of data science and analytics goes hand in hand with the increasing need for interpretability." For these reasons, among others, several explanation frameworks have been developed for black box models like GBT, with SHAP (SHapley Additive exPlanations; Lundberg and Lee (2017)) or LIME (Local Interpetable Model-agnostic Explanations; Ribeiro et al. (2016)) being among the most popular alternatives. These tools are examples of *post-hoc* explanation approaches that approximate the relationship between feature values and predictions from the underlying model to provide a human with a helpful explanation. While such methods can help explain complex black box models, some critics advocate using inherently interpretable models rather than post-hoc explanation tools, especially for high-stakes decisions (Rudin (2019)).

A specific example of machine learning models used to make high-stakes decisions is in the realm of Automated Valuation Models (AVMs) for house price prediction. Financial institutions often rely on AVMs to estimate the value of dwellings, subsequently using these estimates as the basis for decisions about mortgages, property tax, or insurance premiums (Glumac and Des Rosiers (2021)). Recent years have seen an increase in the use of black box machine learning models for this purpose (Steurer et al. (2021)), which also leads to increasing challenges related to model interpretability, given the implication of these decisions for homeowners and financial institutions.

A Generalized Additive Model (GAM; Hastie and Tibshirani (1986)) is a class of models that relates the univariate response variable y to some feature variables $x_1, ..., x_p$ according to

$$g(\mathbb{E}(y)) = f_0 + f_1(x_1) + ... + f_p(x_p), \tag{1}$$

where $g(\cdot)$ is a link function, f_0 is an intercept and $f_1(x_1), ..., f_p(x_p)$ are referred to as the *shape functions* or *partial effect* of $x_1, ..., x_p$, respectively. GAMs are widely considered a fully interpretable class of models (Wood (2006)) due to their additive nature, which allows us to visualize $f_j(x_j)$ to learn about the specific effect of feature x_j. Another appealing consequence is that the final prediction is a sum of the contributions from each shape function $f_j(x_j)$. Recent research efforts by Lou et al. (2012) and Lou et al. (2013) shed light on the connection between the full-complexity GBTs and the interpretable GAMs; using trees with a tree depth of one, referred to as *tree stumps*, removes any interactions terms and thus effectively makes the GBT into a GAM. It also demonstrates that GAMs with tree stumps tend to outperform GAMs created with splines, which is the traditional approach presented by Hastie and Tibshirani (1986).

The drawback of the additive structure of a GAM is the lack of interaction terms. This is problematic in settings where the data stems from an underlying process of heterogeneous groups for which the mapping between the features and the response might vary. The housing market is a textbook example of such heterogeneity, in the sense that the effect a specific feature has on the sale price can differ dramatically based on which geographical region or city district the dwelling comes from. The intuitive handling of a categorical variable x_c, representing, for example, a city district, in a GAM is to include it as a one-hot encoded variable, resulting in a fixed effect for each category. While this might capture the signal in the data well for certain groups, a more flexible model is generally preferable. Another option is to train separate models for each city district, that is, to train individual models only on the subset of the training data from the corresponding city district. This approach will result in tailored models for each category but also prevent information sharing between the groups during training.

This paper introduces LitBoost (Locally Interpretable Tree Boosting), a tree-based boosting model that retains the interpretability benefits offered by a GAM while still producing locally

adaptive and accurate predictions. We achieve this by modifying the standard GBT framework to allow for trees deeper than stumps but by carefully limiting interactions to only those involving the feature that defines a grouping structure. The resulting model is a collection of K local models that are *jointly trained*, meaning that each model has had access to data from the other groups during training. LitBoost allows each local model to borrow strength from similar groups in a manner that would not have been possible if the models were independently trained, that is if we train K entirely separate local models. To facilitate information sharing in the construction of the decision trees, we propose multi-hot encoding, a feature engineering technique for the grouping variable that allows for more flexible splits.

We evaluate our method on a data set of $N = 14\,382$ observations from the housing market in Oslo (Norway) from 2018. We also demonstrate the idea on a synthetic data set inspired by a well-known simulation setup proposed by Friedman (1991). This simulation study allows us to control the between-group similarities, thus investigating how the proposed model performs under different levels of heterogeneity between the groups.

The main contribution of this paper are: (i) We introduce LitBoost, a constrained version of GBT that makes the final model fully interpretable, (ii) We introduce a novel proximity measure to quantify the data sharing between the groups in the training process, and (iii) We contribute to the AVM literature by demonstrating the performance of LitBoost on a novel data set of transactions from the housing market in Oslo (Norway).

The rest of the paper is structured as follows. Section 2 gives an overview of the related literature before Section 3 outlines the details of GBT, GAM, and the link between them, and subsequently introduces our proposed method, LitBoost. Section 4 introduces the housing market application along with a data set, consisting of transactions from the Norwegian housing market, and demonstrates the performance of LitBoost on the data set. Section 5 presents a simulation study where we evaluate the performance of LitBoost in a controlled setting and for different configurations of the data generating process. Section 6 concludes and highlights some challenges and opportunities for further research.

2. Related Methods

A Gradient Boosted Trees model (GBT; Freund and Schapire (1996), Friedman (2001)) fits a sequence of decision trees, each new tree successively trained on the errors from the previous one. GBT has become among the most precise prediction tools in machine learning for tabular data thanks to its flexible nature and ability to capture intricate and non-linear relationships with multiple high-order interactions. Numerically efficient implementations, such as XGBoost (Chen and Guestrin (2016)) and LightGBM (Ke et al. (2017)), have further increased their popularity.

A Generalized Additive Model (GAM) (Hastie and Tibshirani (1986)) is any model that takes the additive form

$$g(\mathbb{E}(y)) = f_0 + f_1(x_1) + ... + f_p(x_p), \tag{2}$$

where the functions $f_j(x_j)$ are referred to as shape functions. Although splines historically have been the preferred functional form of the shape functions (Wood (2006)), Lou et al. (2012) proposed to train a GAM by using a GBT but limiting the tree depth in each tree to one. Furthermore, Lou et al. (2013) expanded upon this work by introducing the GA^{2}M model, a GAM that allowed

a carefully selected set of pairwise interactions, yielding the additive form of

$$g(\mathbb{E}(y)) = f_0 + f_1(x_1) + ... + f_p(x_p) + \sum_{k,l \in \mathcal{I}} f_{kl}(x_k, x_l), \tag{3}$$

where $\mathcal{I}$ denotes the set of interactions included in the model. To select $\mathcal{I}$, Lou et al. (2013) propose an algorithm that greedily includes an interaction pair $f_{kl}(x_k, x_l)$ based on an approximation of its predictive power. Nori et al. (2019) have released an efficient implementation of the GA^{2}M model under the name Explainable Boosting Machines (EBM). In addition to quick interaction detection, EBM applies cyclic boosting to speed up training. Cyclic boosting cycles through the features systematically in a round-robin fashion instead of the more standard boosting procedures, which greedily choose which feature to split on in each iteration.

Other approaches for training the shape functions $f_j(x_j)$ have also been proposed. Agarwal et al. (2021) introduce the Neural Additive Model (NAM), which learns the shape functions $f_j(x_j)$ using deep neural networks, such that the resulting model still is a fully interpretable GAM. Furthermore, GAMI-Net (Yang et al. (2021)) expands upon the NAM model to include carefully selected interaction pairs $f_{kl}(x_k, x_l)$ which are also trained by a neural network. The interactions in GAMI-Net are chosen similarly as in GA^{2}M, albeit with additional constraints relating to the heredity of the included features.

GAMs with interactions are part of a wider model class called Structured Additive Regression (STAR) models (Fahrmeir et al. (2004)), which take the form $g(\mathbb{E}(y)) = h_1(\tilde{\boldsymbol{x}}_1) + ... + h_q(\tilde{\boldsymbol{x}}_q)$ where each argument $\tilde{\boldsymbol{x}}_j$ contains a subset of $x_1, ..., x_p$. As an illustrative example with $p = 4$ features, the set of possible STAR model specifications includes, but is not limited to, model specifications such as $g(\mathbb{E}(y)) = h_1(x_1, x_2) + h_2(x_3, x_4)$ or $g(\mathbb{E}(y)) = h_1(x_1, x_2, x_3) + h_2(x_4)$. The STAR formulation makes q shape functions, each possibly multivariate, rather than the univariate shape functions $f_j(x_j)$ used in a GAM. Mayer et al. (2022) points out that a STAR model can be trained using the XGBoost machinery using the existing methodology for *feature interaction constraints*, which allows the user to specify the features allowed to interact within a single decision tree.

The idea of utilizing tree-based models to create local models for specific data groups has also been explored. Sigrist (2022) introduces Gaussian Process Boosting (GPBoost), which combines tree boosting with grouped random effects in a way that aims to model correlations between categorical groups in the data set. Outside of boosting, Zeileis et al. (2008) introduce an ensemble of segmented tree models where local interpretable models are fitted to different parts of the feature space, defined as the levels in a tree structure. Here, the feature space segmentation and the local models' training are done in a joint fashion.

In summary, the links between the additive class of GAM models and tree-based machine learning models like GBT have been explored in various ways in the literature, with the recently introduced EBM as a notable example. Most of these methodological advances are motivated by a desire to maintain the interpretability of a GAM while improving the predictive accuracy through more flexible shape functions. The motivation behind our research follows the same lines but focuses on the particular task of training a set of locally interpretable models given some available grouping structure in the data.

3. Methodology

In this section, we briefly present Gradient Boosted Trees (GBT) and Generalized Additive Models (GAM), two popular methods for predictive tasks, before introducing our proposed

model, Locally Interpretable Tree Boosting (LitBoost). In addition, we introduce multi-hot encoding, a feature preprocessing technique that allows LitBoost to utilize information across groups in a more flexible way. Finally, we present a novel groupwise proximity measure that quantifies the information sharing across groups during training.

3.1. A Decision Tree

Consider a supervised machine learning problem where the goal is to build a prediction model $f(\boldsymbol{x})$ that returns a prediction $\hat{y} \in \mathbb{R}$ based on features $\mathbf{x} \in \mathbb{R}^p$. A flexible way to build this prediction model is to train a decision tree based on some available training data, containing observations of the form $(\mathbf{x}, y)$, and a loss function that quantifies the difference between the model predictions and the actual values. Let $h(\boldsymbol{x}; q)$ denote the output of a decision tree with tree structure q. Encoded in the tree structure is the division of the feature space into J distinct and non-overlapping regions, each denoted by $\mathcal{R}_j$ and equipped with a value $v_j \in \mathbf{R}$ for $j = 1, ..., J$. Formally, we then have that

$$h(\boldsymbol{x}; q) = \sum_{j=1}^{J} v_j \cdot \mathbb{I}(\boldsymbol{x} \in \mathcal{R}_j).$$

The regions $\mathcal{R}_j$ are usually created to minimize the total variance of the y values in each of the leaf nodes, meaning that the decision tree finds splits that group observations with a similar response. This approach was first introduced by Breiman et al. (1984).

3.2. Gradient Boosted Trees

A Gradient Boosted Trees model trains a sequence of trees such that the next tree is trained on the residuals of the previous tree. After training a sequence of M such trees, the final prediction is then given by

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} \eta \cdot h_m(\boldsymbol{x}; q_m),$$

where h_m is the output function of tree number m in the sequence and $\eta \in (0, 1]$ is a regularization parameter known as the learning rate. The number of trees M, the learning rate η, and the tree depth are hyperparameters that affect the performance of the model and rate of convergence of the training process. The hyperparameters can be tuned based on the application at hand. Increasing the tree depth will allow the model to detect more interactions but also increase the risk of overfitting. XGBoost (Chen and Guestrin (2016)) uses a default tree depth of 6, whereas LigthGBM (Ke et al. (2017)) constrains the complexity of the trees by limiting the number of leaf nodes a default maximum number of 32.

3.3. Generalized Additive Models

A Generalized Additive Model is any model that takes the form

$$g(\mathbb{E}(y)) = f_0 + f_1(x_1) + f_2(x_2) + ... + f_p(x_p), \tag{4}$$

where x_1, ..., x_p are features, $\mathbb{E}(y)$ is the expected value of the response y, $g(\cdot)$ is a link function and $f_1(\cdot), ..., f_p(\cdot)$ are called *shape functions*. This general formulation encapsulates a variety of

statistical models. For instance, we obtain the traditional linear regression model by using the identity link function $g(\mathbb{E}(y)) = \mathbb{E}(y)$ and letting $f_j(x_j) = \beta_j \cdot x_j$ for $j = 1, ..., p$.

The simplest form of decision trees are trees with only a single split, referred to as *tree stumps*. Using tree stumps, each tree divides the feature space into two regions,

$$\mathcal{R}_1 = \{\boldsymbol{x} : x_i \leq a\}, \quad \mathcal{R}_2 = \{\boldsymbol{x} : x_i > a\}$$

for some value a. Note that each tree in the GBT will then be a simple step function that only utilizes one of the p covariates. Although one tree stump will provide a quite coarse binary split of the x_i feature space, an aggregation of multiple binary splits on x_i will generate a non-smooth and possibly rather complicated function $f_j(x_j)$. Thus, training a GBT with tree stumps makes it possible to express the final model as a GAM, as tree stumps do not introduce interactions. To visualize the shape function pair $(x_j, f_j(x_j))$ we can accumulate all the trees that split on x_j and discard the other trees. This procedure can be repeated for all the p covariates, resulting in the additive structure of a GAM.

To increase the expressiveness provided by standard GAMs, Lou et al. (2013) propose to modify the GAM framework to be of the form

$$g(\mathbb{E}(y)) = \sum_{j=1}^{p} f_j(x_j) + \sum_{k,l \in \mathcal{I}} f_{kl}(x_k, x_l), \tag{5}$$

where the latter term involves a small set of carefully chosen pairwise interactions $\mathcal{I}$. The identification of the relevant interactions is an important computational bottleneck. In principle, we must evaluate the predictive power of all combinations of possible pairwise interactions to determine the optimal $\mathcal{I}$. However, since this is usually computationally infeasible, Lou et al. (2013) use a greedy forward stagewise selection strategy called FAST to identify the interactions in $\mathcal{I}$. FAST approximates the true gain in predictive power adding $f_{kl}(x_k, x_l)$ by taking the average of the response in various subsets of the (x_k, x_l) feature space, which makes it possible to evaluate all possible pairs at reasonably low computational and memory costs. An efficient implementation of models in the form of (5) was released by Nori et al. (2019) under the name Explainable Boosting Machine (EBM).

3.4. Interpretability in Tree-Based Models

A significant drawback of GBT with deeper trees is the lack of interpretability. While a single decision tree may be easy to inspect, a sequence of M trees quickly becomes hard to understand as the number of trees is increased. A single tree of depth D divides the feature space into at most 2^D distinct regions while also introducing a potentially large number of interactions between the covariates, making it difficult to isolate the effect of each covariate.

Unlike with a GAM, the shape functions $f_j(x_j)$ are not readily available with GBT. Tools exist to visualize marginalized shape functions for deeper trees, most notably through a *partial dependence plot*. To generate a partial dependence plot for covariate x_j we must marginalize out the effect of all the other covariates $x_1, ..., x_{j-1}, x_{j+1}, ..., x_p$. Greenwell (2017) presents a practical way of estimating the partial dependence plots, essentially by keeping x_j fixed while changing the other features in a discrete grid and storing the prediction for each grid point. While the partial dependence plot gives a hint about the contribution of a covariate to the final prediction $\hat{y}$, it does not give a perfect explanation in the following sense. Due to the interaction terms that are introduced by having deeper trees, a partial dependence plot of $(x_j, \hat{y})$ will not reveal to us the

exact prediction corresponding to a set of feature values $(x_1, ..., x_p)$. Instead this plot will give us the prediction given x_j, averaged over all the other input variables. Thus, the partial dependence plot cannot decompose any prediction $\hat{y}$ in the same additive manner as a GAM.

In the case of EBM, where additional interaction pairs (x_k, x_l) are included in the model, one can still visualize the model through the shape functions $f_j(x_j)$. However, in addition, one needs to resort to, for example, heat maps $f_{kl}(x_k, x_l)$ in the two-dimensional (x_k, x_l) space. Although most humans find it easier to visually analyze a one-dimensional shape function $f_j(x_j)$ than a heat map $f_{kl}(x_k, x_l)$, Lou et al. (2013) argues that this is still intelligible.

3.5. LitBoost

Instead of using a single interpretable model (GAM or EBM), we propose using a collection of local interpretable models defined by some designated grouping variable x_c. To efficiently train the proposed model class using existing and highly optimized GBT implementations, we introduce LitBoost (Locally Interpretable Tree Boosting), a method for training a collection of local and interpretable tree boosting models through interaction constraints. For the sake of simplicity, consider a prediction problem with a numerical variable x_1 and a categorical variable x_c, where $x_c \in \{A, B, C, D\}$, that is, we have $K = 4$ distinct groups. Figure 1 shows a simple decision tree that divides the (x_1, x_c) space into four distinct regions. The tree can be expressed as a prediction function $f(x_1, x_c)$ that returns the prediction v_1, v_2, v_3, v_4 depending on which leaf the observations belong to. Due to the nature of the categorical x_c, we can express $f(x_1, x_c)$ in an additive manner as

$$f(x_1, x_c) = f_1^A(x_1) + f_1^B(x_1) + f_1^C(x_1) + f_1^D(x_1) = f_1^{x_c}(x_1),$$

where the subscript indicates which feature the shape function belongs to, and the superscript indicates which of the K local models the shape function belongs to.

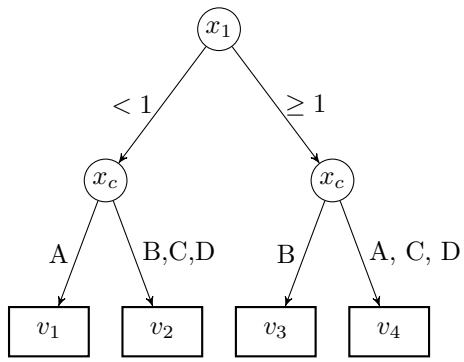

Figure 1: A simple decision tree dividing the feature space into four regions with predicted value v_1, v_2, v_3, v_4 in the four regions. The first split is made on x_1, with values smaller than 1 going to the left and values larger than (or equal to) 1 going to the right. The second split is on the categorical feature x_c, with different splits on each side of the tree.

Figure 2 displays visually how the tree in Figure 1 can be re-expressed as a decomposition of four local models. We can further generalize this idea to the more general case where we have p

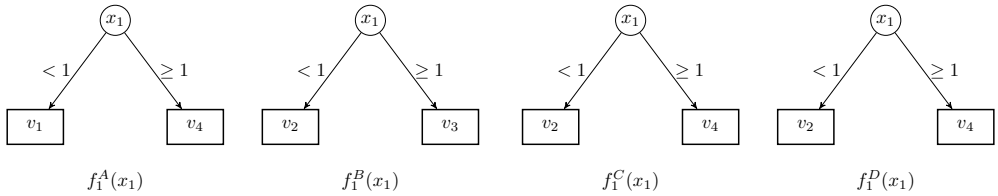

Figure 2: Four local models $f_1^A(x_1), f_1^B(x_1), f_1^C(x_1), f_1^D(x_1)$ that still conveys the same information as the tree $f(x_1, x_c)$. The leaf values v_1, v_2, v_3, v_4 are mapped to the same parts of feature space as in Figure 1.

model features $x_1, .., x_p$ in addition to the categorical feature x_c with $|x_c| = K$, as

$$f(x_1, .., x_p, x_c) = \sum_{x_c=1}^{K} \sum_{j=1}^{p} f_j^{x_c}(x_j), \tag{6}$$

where subscript j indicates the feature and superscript x_c indicates which of the K groups (or local models) the shape function belongs to. The Local GAM for a given group A can thus be expressed as

$$f(x_1, ..., x_p | x_c = A) = \sum_{j=1}^{p} f_j^A(x_j). \tag{7}$$

This type of decomposition is not possible in any decision tree. The nature of the tree-based models is to search greedily for the split that improves some objective function. A decision tree with a tree depth of more than one could thus consist of a split on, e.g., x_1 followed by x_2, creating an interaction between x_1 and x_2. To be able to enforce the model structure described in (7) from a jointly trained model, we must impose *feature interaction constraints*. More specifically, we only allow interactions between the categorical covariate x_c and one other covariate at the time. Explicitly, within one tree, we will allow either of the interactions (x_c, x_1), (x_c, x_2), ..., (x_c, x_p), but not any other. If a tree has made a split on x_j, it must either continue to split on x_j or split on x_c. The conditional GAM structure of (7) is maintained regardless of tree depth as long as the interaction constraints are imposed.

Training the K local models jointly with interaction constraints allows the model to borrow strength between the groups in a way that is impossible if we independently train K separate models on subsets of the data. Each decision tree has access to data from the other groups, thus allowing multiple splits on x_j that will be common between all groups before creating local effects by splitting on x_c.

We have now discussed a handful of standard model specifications based on boosted trees in addition to the introduction of LitBoost. As a summary to aid the reader, Table 1 shows an overview of the methods presented in this work.

Model	Training type	Interactions	Interpretability
GBT	Gradient boosting	Higher-order	Partial dep. plot
EBM	Cyclic gradient boosting	Pairwise interactions, automatically detected	Shape functions and interaction heat maps
GAM	Gradient boosting	None	Shape functions
LitBoost	Gradient boosting	Only involving user-specified categorical variable x_c	Local shape functions for each group

Table 1: A comparison of the methods used in this work.

3.6. Multi-Hot Encoding of Categorical Variable

The conventional way of handling categorical variables in decision trees is to transform them into binary indicators, often referred to as one-hot encoding (Chen and Guestrin (2016), Breiman (2001)). A categorical variable x_c with K levels $G_1, ..., G_K$ is transformed into K new features by means of the indicator function $x_{ci} = \mathbb{I}(x_c = G_i)$, such that x_{ci} takes the value 1 if the data point is part of the group G_i and the value 0 otherwise. Another option is integer encoding, which does not create K new features but maps the categories of x_c to a set of integer values. While this might be intuitive if the levels of x_c have some inherent ordering, for instance, if x_c represents university grades ranging from A to F, this is seldom the case for categorical variables.

A challenge with applying one-hot encoding is that the decision trees are forced to create splits that partition the feature space into two highly imbalanced nodes, where one of the K categories is present in one node while all the other $(K - 1)$ categories are present in the other node. This is exemplified in Figure 1, where the left branch of the tree splits on $\{A\}$ in one node and $\{B, C, D\}$ in the other. Although this might be unproblematic if K is reasonably small, it might pose a problem as K grows, in particular, since it limits the flexibility of a tree in terms of finding a good partitioning based on x_c.

To overcome this, we introduce *multi-hot encoding*, a method for enriching the data set with additional x_c related columns before training. We translate x_c to a set of binary indicator features in a similar fashion as one-hot encoding, but in addition, add a feature that take value 1 if x_c is part of a set of categories. Consider, for instance, the three-hot encoded variable

$$x_{cijk} = \mathbb{I}(x_c = G_i \vee x_c = G_j \vee x_c = G_k), \tag{8}$$

where $\vee$ is the "or" operator. With (8), x_{cijk} gets the value 1 if x_c is part of either group G_i, G_j or G_k. The benefit of introducing multi-hot encoding, in addition to one-hot encoding, is that it allows the model to identify similar clusters of categories and use these in the splitting process. It is likely to lead to more efficient training in the sense that the algorithm will achieve better predictions and quicker convergence in the training error, as the model shares data between similar groups in a way that is not possible with one-hot encoding.

An example that exploits two-hot encoding of $x_c \in \{A, B, C, D\}$ can be seen in Figure 3. Due to the added flexibility given by the two-hot encoding, the tree model can split the feature space into groups consisting of two and two groups in a manner that would not have been possible with one-hot encoding. However, when applying multi-hot encoding, one should be aware that the number of new columns might quickly become infeasibly large if K is high. Depending on the implementation and tree model being applied, this might lead to computational challenges.

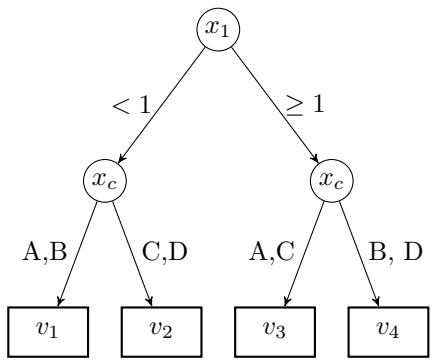

Figure 3: A decision tree with two-hot encoding, allowing splits that are not merely of type one-against-the-rest.

3.7. Groupwise Proximity Measure

Since the local models are trained in a joint fashion it will allow two groups to borrow strength from each other during training in a way that would not be possible if each local model were trained independently. The amount of data sharing will vary for different pairs of groups, and as a way of quantifying it, we define a measure based on the *proximity matrix* of each decision tree. Breiman (2001) defined the proximity between observation a and observation b in a decision tree with structure q as

$$\text{Prox}_q(a, b) = \begin{cases} 1 & \text{if } a \text{ and } b \text{ are in the same leaf,} \\ 0 & \text{otherwise.} \end{cases} .$$

The proximities between observations can be aggregated to a groupwise proximity measure that quantifies the similarity between group G_i and group G_j, two levels of x_c, as the mean proximity between every pair of observations across the two groups:

$$\tilde{H}_q(G_i, G_j) = \frac{1}{N_i N_j} \sum_{a \in G_i} \sum_{b \in G_j} \text{Prox}(a, b; q), \tag{9}$$

where N_i and N_j denotes the total number of observations in the respective groups. The value of $\tilde{H}_q(G_i, G_j)$ should be interpreted as the mean number of times an observation from G_i is in the same leaf as an observation in G_j. A high value of $\tilde{H}_q(G_i, G_j)$ indicates a significant degree of information sharing between the groups; since the observations often land in the same leaf of the decision tree, the predictions of new test instances from group G_i will be affected by the training instances from G_j and vice versa. This, in turn, might indicate a degree of similarity between the training samples from G_i and G_j.

Furthermore, we can aggregate the proximity measure over all the M trees in the sequence, yielding

$$\tilde{H}(G_i, G_j) = \frac{1}{M} \sum_{m=1}^{M} \tilde{H}_{q_m}(G_i, G_j),$$

where q_m encodes the tree structure of tree m. Finally, we normalize the proximity measure in a way that ensures $\tilde{H}(G_i, G_i) = 1$ for every group G_i, and our final measure is thus given by

$$H(G_i, G_j) = \frac{\tilde{H}_{q_m}(G_i, G_j)}{\sqrt{\tilde{H}_{q_m}(G_i, G_i)\tilde{H}_{q_m}(G_j, G_j)}}. \tag{10}$$

A few comments to guide the reader in their interpretation of $H(G_i, G_j)$ are in order. $H(G_i, G_j)$ will be a value between 0 and 1 that quantifies how often observations from group G_i and group G_j are placed in the same leaf during the training process, implying the degree of information sharing between the groups during training. For a given group, for instance, G_1, we can study the $(K-1)$ groupwise proximity measures $H(G_1, G_2), ..., H(G_1, G_K)$ and use this as an indication of which groups that are most similar to G_1 based on the tree structures. It is also natural to think in terms of a groupwise proximity matrix, which will be a symmetric $K \times K$ matrix with value 1 on the diagonal and $H(G_i, G_j)$ on position (i, j). As an extreme example, if one trains K local models without data sharing, the proximity matrix reduces trivially to the identity matrix.

With the use of deeper trees, one could expect to see a decrease in all $H(G_i, G_j)$ values since deeper trees partition the feature space into more subsets, effectively making the proximity matrix $\text{Prox}_q(a, b)$ sparser. One must therefore be careful when analyzing and comparing groupwise proximity measures across different data sets and models, but instead use it as an explanatory tool to gain insights about a specific data set and a specific model trained on this data set.

4. Oslo Housing Price Application

4.1. The Norwegian Housing Market

Norway has a high ownership rate compared with the Organisation for Economic Co-operation and Development (OECD) average and a strong tradition for owning rather than renting.[1] Most homeowners get a mortgage to finance the acquisition, often up to five times the annual salary. For this reason, banks desire to continuously monitor the fair value of the dwellings being used as collateral. While physical inspection often is the optimal way to assess the value of the dwelling, it is often not feasible to manually appraise hundreds of thousands of dwellings regularly. Therefore, most banks rely on Automated Valuation Models (AVMs) instead. An AVM is a statistical model used for estimating the fair value of a dwelling at the current time, given the characteristics of the dwelling. Using AVMs to estimate dwelling values is not a novel idea, and Bailey et al. (1963) and Rosen (1974) are early examples of applying regression models to appraise a portfolio of dwellings statistically. Although different statistical models might be favorable in different markets and geographical regions, the majority of modern AVM research suggests that tree-based models like random forest and gradient boosted trees (Ho et al. (2020), Kim et al. (2021), Hjort et al. (2022)) often outperform traditional statistical methods such as linear regression or nearest neighbor regression.

[1] OECD Affordable Housing Database, accessed 2nd of January 2023.
`https://www.oecd.org/housing/data/affordable-housing-database/`

4.2. The Oslo Data Set

We use a data set of all arms' length transactions of apartments in Oslo (Norway) from 2018. Oslo is the capital and largest city in Norway. The total number of observations in the data set is $N = 14\,382$, where each data point represents a single sale, i.e., a transaction of one dwelling. Each transaction includes a sale price (the response variable in the regression) and $p = 14$ covariates for each transaction. These features contain information that is typically of high importance to homeowners, for instance, the dwelling size (in m^2), the number of bedrooms, the floor the apartment is on, and the age of the building.

The data set also includes information about the area surrounding the dwelling. The data set is based on a division of Oslo into grids of size 250×250 meters. We utilize this to measure the number of nearby amenities, like shops, schools, and churches, in the closest eight grid cells to the cell of the targeted dwelling. This information is summarized in the covariates NearbyHomes and NearbyBuildings. This information is valuable to potential buyers of an apartment, as it gives insight into essential neighborhood characteristics.

Furthermore, Oslo is divided into $K = 15$ distinct and non-overlapping city districts. These city districts are depicted in Figure 4 with color coding showing the mean price per square meter. The prices are measured in thousands NOK.[2] We have concentrated the analysis on sales registered as apartments, thus excluding other estate types such as row houses, detached homes, and duplexes. This will naturally lead to a higher density of transactions from central areas, as detached homes often dominate the areas outside of the city center.

A summary of the data is provided in Table 2.

Table 2: The variables in the data set with summary statistics for the numerical variables.

Variable	Unit	Mean	St. Dev.	Min	Max	Type
Sale Price	NOK (mill.)	4.44	1.97	1.1	45.0	Numerical
City District[1]	-	-	-	-	-	Categorical
Sale Date	months	6.34	3.23	1	12	Numerical
Altitude	m	91.43	61.99	0	480	Numerical
Size	m^2	65.43	23.74	15	267	Numerical
Floor[2]	-	3.01	1.90	-3	14	Numerical
Bedrooms	-	1.78	0.76	0	9	Categorical
Dwelling Age	years	61.47	37.13	0	189	Numerical
Balcony[3]	-	0.75	0.43	0	1	Binary
Elevator[3]	-	0.37	0.48	0	1	Binary
Units On Address[4]	-	20.45	28.51	0	274	Numerical
Coast Distance	m	3,197	2,411	5	12,201	Numerical
Lake Distance	m	969.60	500.34	32	3,018	Numerical
Nearby Homes[5]	-	2,820	1,591	100	6,746	Numerical
Nearby Buildings[5]	-	166.3	145.32	6	1,323	Numerical

[1] There are 15 distinct city districts in the data set.
[2] If the dwelling has multiple floors, this variable will be the lowest floor.
[3] In cases where the information is missing, this is set to 0.
[4] In apartment buildings multiple dwellings might have the same address.
[5] Norway is divided into squares of 250m×250m. This variable counts the number of homes or other buildings (stores, schools, churches) in all the adjacent squares to the square where the targeted dwelling is, i.e., the 8 neighboring squares.

[2] 1 NOK ≈ 0.1 USD as of June 22nd, 2023.

Figure 4 shows a map of the 15 city districts present in the Oslo data set. The city districts are color-coded according to the mean sale price (million NOK) per city district. This reveals a higher average sale price pattern in the western parts of the city, with Frogner and Ullern as the areas with the highest average price. Similarly, the city districts with the lowest average sale price in the data set are the north-eastern districts of Stovner and Grorud, as well as the south-eastern district of Søndre Nordstrand.

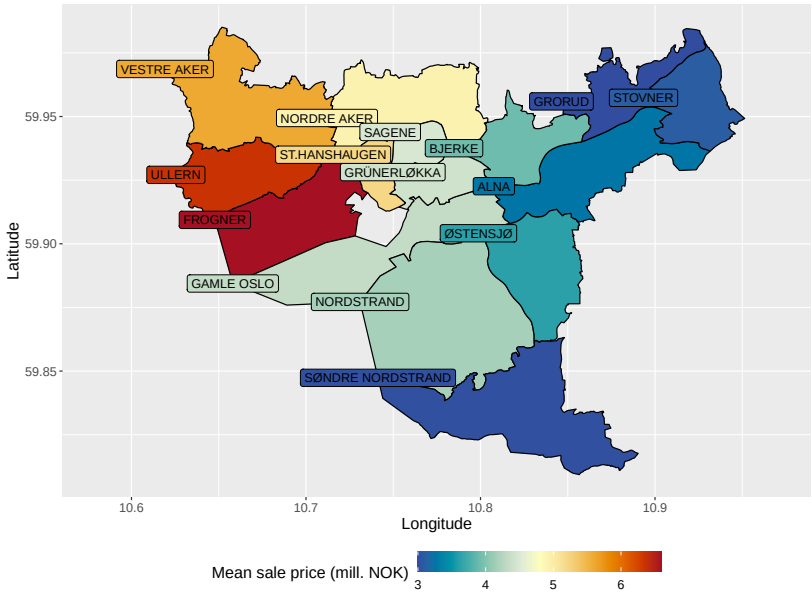

Figure 4: A map of the $K = 15$ city districts in the data color-coded according to the mean sale price (in million Norwegian kroner). The city districts with lower average sale prices have a blue color, whereas the city districts with higher average sale prices have a red color. The empty city district east of Frogner is a district (Sentrum) without any observations.

4.3. Simulation Setup and Implementation Details

We compare the performance of LitBoost with GBT, GAM, and Local GAM. At each experiment, we sample a fixed number of observations per city district for training and use the rest of the data for testing. We vary the number of observations per group in the range $(10, 20, 40, 80, 160, 320)$, but also report the results when 50% of the original data is sampled for training, and the other 50% are used for testing. This is referred to as N_{max}, since the number of observations differs between groups in the complete data set.

We repeat each experiment 20 times to account for randomness in the data set splitting and report the average Root Mean Squared Error (RMSE) across the 20 simulations. The performance of LitBoost is reported in two different variants: One with the presence of both one-hot and two-hot encoded variables for city districts and one with only one-hot encoding.

LitBoost can, in practice, be run by using the XGBoost library in `R` (Chen and Guestrin (2016)) and utilize the built-in `interaction_constraints` functionality. This functionality permits the user to send a list of pairwise column names into the training of the XGBoost model, constraining the XGBoost model to only include the interactions that are featured in the user-specified list.

Each model is trained using the `XGBoost` library in `R` (Chen and Guestrin (2016)), with a learning rate of $\eta = 0.01$, and a total number of $M = 2000$ iterations per model. We use a tree depth of 1 for the GAM and the Local GAM, and a tree depth 3 for LitBoost. For the GBT model, we include a version with a tree depth 3 (the same as LitBoost) and a version with a tree depth 6 (the default option). All models are trained on the squared error loss function, which is also the default option in XGBoost. We also report the results using EBM as the base model instead of GAM, including one pairwise interaction. In this case, we use the `interpret` package (Nori et al. (2019)) to train the EBM and the Local EBM. In the LitBoost variant where one additional interaction is allowed, we identify the best interaction (x_k, x_l) by training a simple model $f_{kl}(x_k, x_l)$ on the residuals of the model without interactions. While this is not computationally optimal, it is a brute-force application of the idea presented in Nori et al. (2019). We then allow `XGBoost` to utilize the identified interaction in addition to the interactions involving the city district variable, as described in Subsection 3.5.

4.4. Results

The results are summarized in Figure 5, showing that LitBoost performs significantly better than the global GAM but worse than both GBT models for the larger sample sizes. LitBoost yields significantly better performance than the Local GAM when N per city district is moderate, although the gap in performance between the two decreases as N increases. This is as expected in the sense that when the number of observations in each group is sufficiently large, entirely local models can capture all the specific effects of this city district quite well, but they struggle when N is small. This also highlights the advantage that LitBoost gains from sharing information between groups during training, yielding excellent accuracy in comparison to the other methods when N is small.

We also see a slight improvement in LitBoost when using two-hot encoded variables. This is because the tree models are better equipped to identify clusters of similar city districts when they are allowed to split using two groups simultaneously in the splitting process. Inspection of feature importance values for LitBoost with two-hot encoding shows that the model can identify city districts that make sense in terms of geographical distance and socioeconomic similarities. For instance, the most important two-hot encoded variable based on feature importance from XGBoost is the one that includes Sagene and Gamle Oslo, two city districts that are close both geographically and in terms of the mean price level.

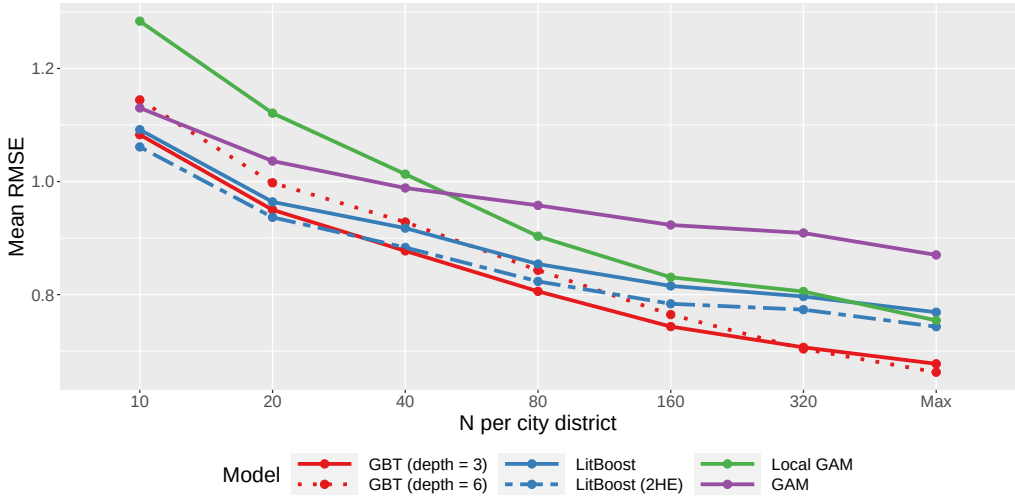

Figure 5: Mean RMSE of 20 simulations with varying number of observations per city district for GBT with two different tree depths, GAM, Local GAM, and LitBoost with and without two-hot encoded variables (2HE). The rightmost value on the x axis, N_{max}, refers to the setting where 50% of the full data set is used for training and the rest for testing, giving an unequal number of observations per group. The prediction errors are measured in million NOK.

We repeat the same experiment but include an additional interaction $f_{kl}(x_k, x_l)$ in LitBoost. We thus compare with EBM instead of GAM. The results, shown in Figure 6, are similar to those without interactions presented in Figure 5. Once again, LitBoost with two-hot encoding is performing at the level of the benchmark model GBT when N is small, with GBT doing progressively better as N grows. The EBM follows LitBoost somewhat closer than what the GAM did in Figure 5, which demonstrates the benefit of including an interaction. Although the Local EBM improve significantly as N grows, they cannot surpass the EBM even for the maximum number of observations. This contrasts the performance of Local GAM in Figure 5, which performed significantly better than the GAM as N grew. This highlights how much a global model can improve by including only one carefully selected interaction.

It might be surprising to see that the Local EBM in Figure 6 seemingly performs worse than the Local GAM in Figure 5, as the EBM is simply a GAM with a single additional interaction allowed. This is most likely due to differences in the implementation of the gradient boosting method between the GAM and the EBM, as described in Subsection 3.3, and one should keep this in mind when comparing the performance directly between the two.

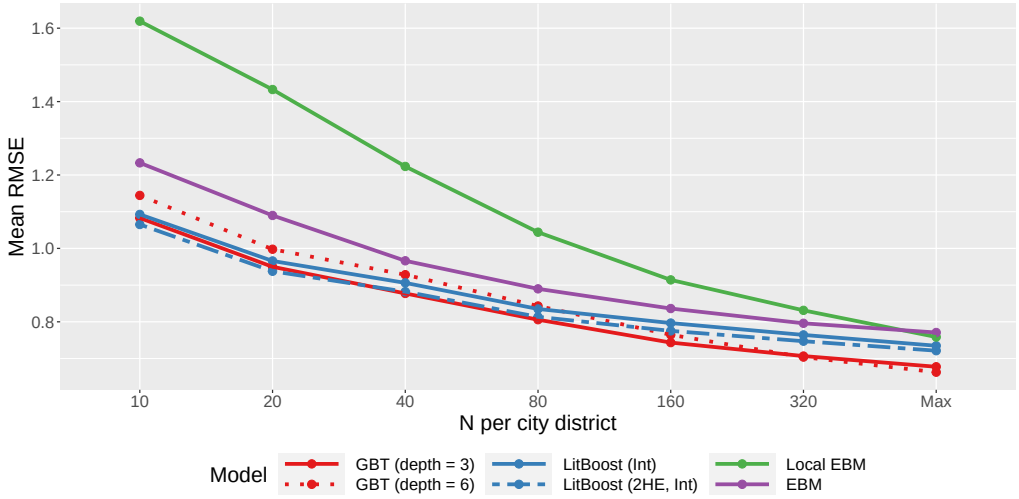

Figure 6: Mean RMSE of 20 simulations with a varying number of observations per city district for GBT with two different tree depths, EBM, Local EBM, and a LitBoost model that allows for one additional interaction. The results for LitBoost are reported both with and without two-hot encoded variables (2HE). The rightmost value on the x axis, N_{max}, refers to the setting where 50% of the full data set is used for training and the rest for testing, giving an unequal number of observations per group. The prediction errors are measured in million NOK.

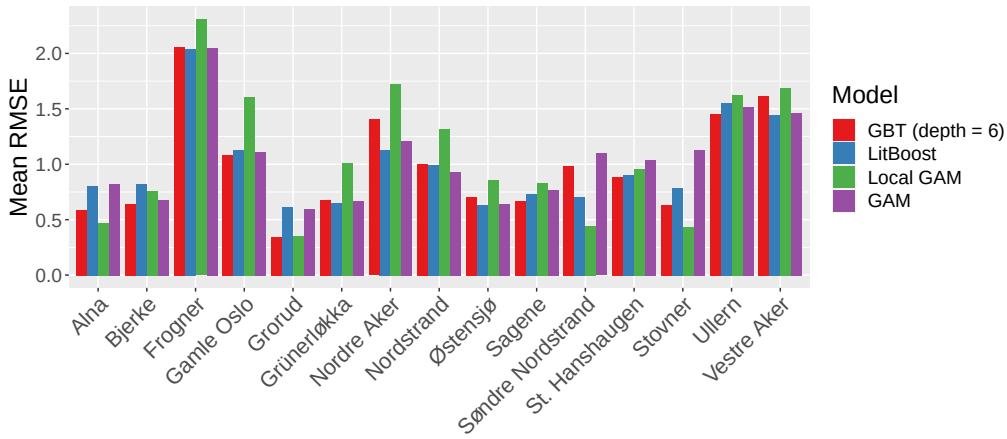

Figure 7: Mean RMSE per city district over 20 simulations with $N = 80$ observations per city district in the training set for GBT, GAM, Local GAM, and LitBoost. The prediction errors are measured in million NOK.

Figure 7 displays the mean RMSE per city district for the GBT with depth 6, LitBoost, the Local GAM, and the GAM. The errors are the average of 20 simulations, each with $N = 80$ per city district. The figure reveals that the errors are, unsurprisingly, higher in the city districts with the highest sale price, such as Frogner and Ullern. Interestingly, the Local GAM displays a quite volatile performance across the districts, clearly outperforming GBT in the areas with lower sale prices (Alna, Grorud, Stovner, Søndre Nordstrand) but performing worse in more expensive

regions such as Frogner, Gamle Oslo, and Nordre Aker.

LitBoost performs slightly worse than Local GAM and GBT in the areas where sale prices are on average lower. Even if we have a relatively modest number of observations per group, it seems to be enough for the Local GAM to learn a meaningful model in these areas, most likely due to the homogeneity of the dwellings that are often present in these city districts. The LitBoost, the GAM, and, to some degree, the GBT model struggle in these areas relative to the Local GAM.

However, the opposite is true for areas like Nordre Aker and Vestre Aker, where LitBoost significantly outperforms the Local GAM and GBT. We also see slightly lower errors for LitBoost compared to GBT in Grünerløkka and Østensjø. Overall, the results suggest that LitBoost performs at approximately the same level as GBT in most city districts and slightly better than GAM on average for the considered sample size.

4.5. Shape Functions

The shape functions f_{Size} for the GAM, Local GAM, and LitBoost can be seen in Figure 8. This shows the effect of the size of a dwelling, measured in m^2, on the final prediction. The size of a dwelling is widely regarded to be among the most important determinants for the final sale price, so it is interesting to study its shape function(s). While the GAM learns a single shape function for every city district, the Local GAM learns a separate shape function f_{Size} for each city district, completely independent of each other. The LitBoost shape functions display the same trends as the Local GAM shape functions but are pushed toward the general global trend. This type of "regularization" can be essential in parts of the feature space where one city district has very few observations, such as for small values of x_{Size}.

The shape functions produced by LitBoost inherit the interpretability benefits of the GAM. Practitioners can study the shape functions for a specific city district of interest, revealing how the prediction model behaves in different parts of the feature space. Similar plots of $(x_j, f_j(x_j))$ can be produced for every covariate x_j. In addition to the global interpretability offered by the shape functions, one could also create local explanations for specific predictions by studying how much each shape function $f_j(x_j)$ contributes towards the final prediction for that dwelling, which can be highly useful for practitioners. The shape functions for the other numerical features for LitBoost, GAM, and Local GAM are shown in Appendix B.

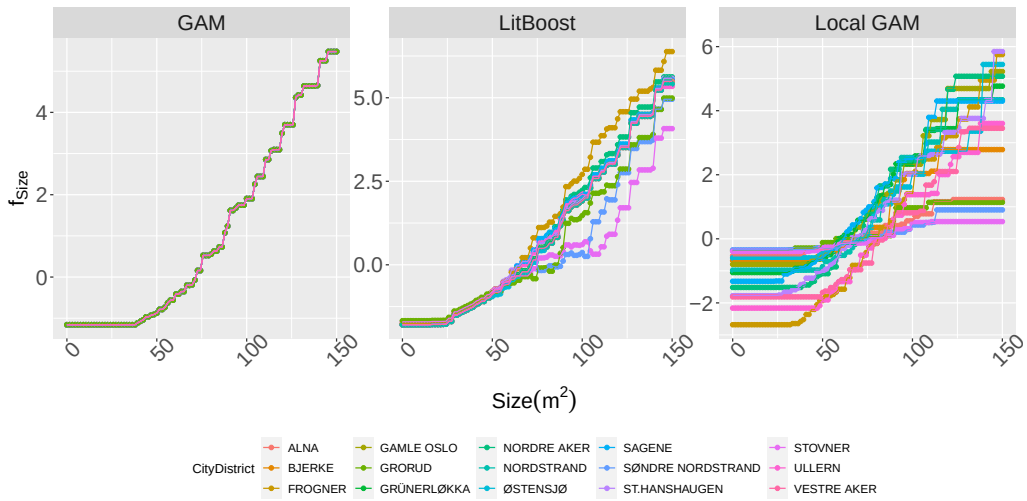

Figure 8: The shape function f_{Size} for the GAM, LitBoost, and Local GAM. The shape function shows how much the contribution f_{Size} changes when the covariate value of size is changed.

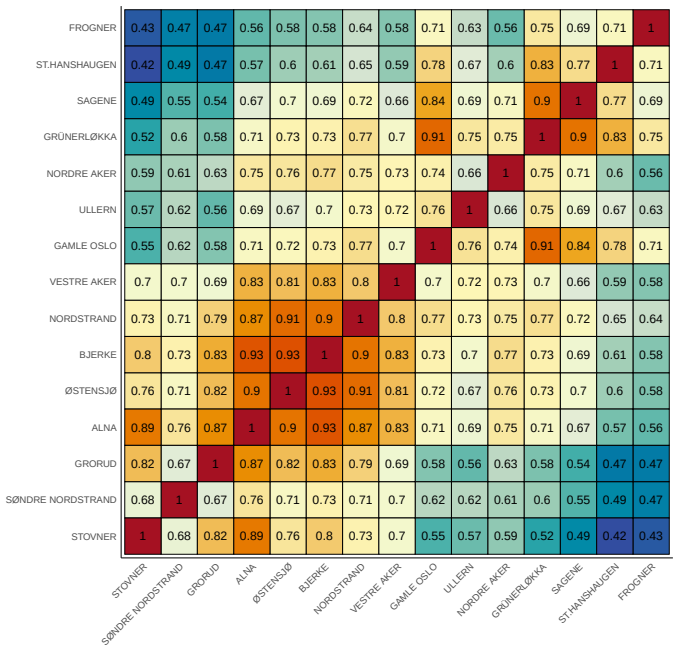

Figure 9: A matrix with the groupwise proximity measure for the city districts in Oslo. A red indicates a high level of proximity between the city districts, and a blue indicates low proximity. This shows the average of 20 simulations, each time with a subsample of $N = 2000$ observations used to train the model, to work around memory allocation challenges when N gets larger. The groups are sorted from the lowest mean price per square meter (Stovner) to the highest mean price per square meter (Frogner). For any given row (or corresponding column), the numbers represent the proximity between that city district and each of the others.

4.6. Groupwise Proximity Measure

Figure 9 shows a symmetric 15×15 matrix with the average groupwise proximity measure between the city districts in Oslo. The highest numbers are typically found close to the diagonal, which is expected as the groups are sorted based on price. This trend indicates that most groups primarily borrow strength from city districts with comparable prices. The low numbers in the upper left and lower right corners signal the same effect: the local models associated with the more expensive districts borrow less strength from data from less expensive districts and vice versa.

The lower left area of the matrix generally has higher values than the upper right corner, indicating that the borrowing of strength is greater between the six or seven city districts with the lowest mean price than the six or seven with the highest mean price. This suggests that city districts with lower average prices more often borrow strength from each other than those with higher average prices. One possible explanation is that the cheaper city districts are dominated by apartment blocks that share the same size, number of bedrooms, and other characteristics. In contrast, the luxurious areas often are more heterogeneous, with a greater variety of dwelling characteristics.

5. Simulation Study on Synthetic Data Set

This section performs a simulation study of LitBoost on synthetic data, which allows us to control the differences between the groups by adjusting the data generating process. We first introduce the data generating process and then show results with and without interactions under three different configurations of the data generating process. Finally, we study some of the shape functions produced by LitBoost and analyze the groupwise proximity measures.[3]

5.1. The Data Generating Process

We are following a similar simulation setup as the one suggested by Friedman (1991):

$$\begin{aligned} &x_1, ..., x_{10} \overset{\text{iid}}{\sim} Unif[0,1] \\ &y = 10 \cdot \sin(\pi x_1 x_2) + 20 \cdot (x_3 - 0.5)^2 + 10 \cdot x_4 + 5 \cdot x_5 + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2), \end{aligned} \tag{11}$$

which contains both linear terms, non-linear terms, and interaction terms, as well as five features that are not included in the outcome model. Thus, this data-generating process is well suited to studying additive models with certain interactions allowed. In order to test the model introduced in Section 3, we edit the data generating equations in (11) to facilitate different effects for different categories. In order to achieve this, we introduce a new variable x_C that assigns each observation uniformly to one of K groups. Furthermore, we modify the model parameters in the original scheme based on which group the observations come from. Formally,

$$\begin{aligned} &x_1, ..., x_{10} \overset{\text{iid}}{\sim} Unif[0,1] \\ &x_C \sim \{1, 2, ..., K\} \\ &y = a_C \cdot \sin(\pi x_1 x_2) + b_C \cdot (x_3 - 0.5)^2 + c_C \cdot x_4 + d_C \cdot x_5 + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2), \end{aligned} \tag{12}$$

[3]The R code used to train LitBoost, display shape functions, and generate the synthetic data used in this section is available at Github. The URL is omitted for double-anonymized reviewing.

where a_C, b_C, c_C, d_C are thus group-specific parameters for each category. This way, we can control the similarity between the groups: Choosing $a_1, ..., a_K$ to be very close will lead to a similar data generating process for each group, whereas choosing very different values $a_1, ..., a_K$ will lead to large between-group differences. The within-group variance is the same for all groups, controlled by σ_ε^2.

A plot of the true data generating shape functions from (12) can be found in Figure 10. Here we have chosen $K = 3$ and model parameters $\boldsymbol{a} = (5, 10, 15)$, $\boldsymbol{b} = (15, 20, 25)$, $\boldsymbol{c} = (5, 10, 15)$, $\boldsymbol{d} = (3, 5, 7)$. This illustrates how all three groups follow the same trend, but the trend differs in intensity based on the values of the model parameters. The first two figures show f_1 plotted against x_1 and x_2, respectively, which is a projection of a three-dimensional surface $(x_1, x_2, \sin(\pi x_1 x_2))$ onto x_1 and x_2, explaining why the data in these figures look noisier than the data in the final three figures. These figures also reflect what a GAM trained on this data will do, as interaction effects between x_1 and x_2 are projected onto the univariate shape functions corresponding to covariate x_1 and x_2.

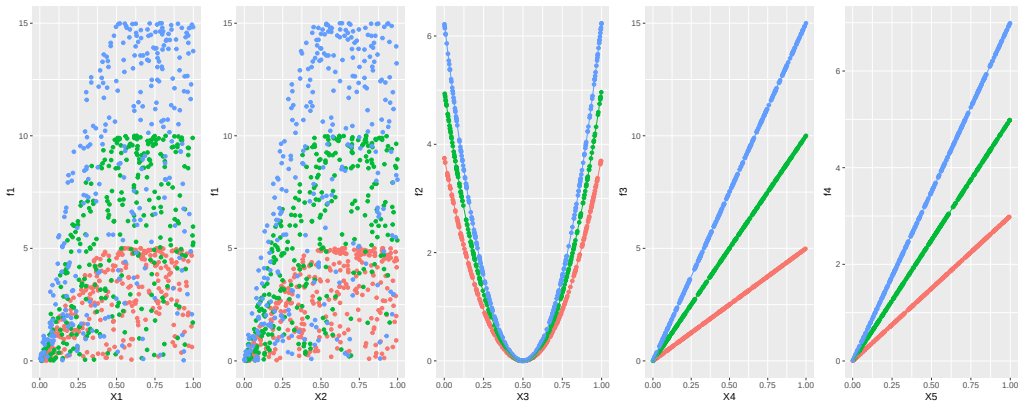

Figure 10: The data generating process with three different groups, $N = 300$ observations per group and parameters $\boldsymbol{a} = (5, 10, 15)$, $\boldsymbol{b} = (15, 20, 25)$, $\boldsymbol{c} = (5, 10, 15)$, $\boldsymbol{d} = (3, 5, 7)$.

5.2. Simulation Setup

We simulate from the data generating process presented in (12) with $K = 9$ distinct groups. To emulate a setup where there are clusters of groups that display similar trends, as seen in the Oslo housing data set, we generate the hyperparameters as follows:

$$
\begin{aligned}
a_1, a_2, a_3 &\sim \mathcal{N}(5, \sigma^2) & a_4, a_5, a_6 &\sim \mathcal{N}(10, \sigma^2) & a_7, a_8, a_9 &\sim \mathcal{N}(15, \sigma^2) \\
b_1, b_2, b_3 &\sim \mathcal{N}(15, \sigma^2) & b_4, b_5, b_6 &\sim \mathcal{N}(20, \sigma^2) & b_7, b_8, b_9 &\sim \mathcal{N}(25, \sigma^2) \\
c_1, c_2, c_3 &\sim \mathcal{N}(5, \sigma^2) & c_4, c_5, c_6 &\sim \mathcal{N}(10, \sigma^2) & c_7, c_8, c_9 &\sim \mathcal{N}(15, \sigma^2) \\
d_1, d_2, d_3 &\sim \mathcal{N}(3, \sigma^2) & d_4, d_5, d_6 &\sim \mathcal{N}(5, \sigma^2) & d_7, d_8, d_9 &\sim \mathcal{N}(7, \sigma^2).
\end{aligned}
$$

The fact that we draw the first three groups from the same distribution, the second three groups from another distribution, and the final three groups from the third distribution, creates three *clusters* of similar groups. The cluster information is not available to the methods during training, but this will test the models' ability to identify and borrow strength between similar

groups. The σ^2 parameter can be varied in order to control the differences between the three clusters in the sense that higher σ^2 values will make the three clusters less distinct and increase the difference within each cluster. In contrast, smaller σ^2 values will make the within-cluster differences smaller and the between-cluster differences relatively larger. By studying the resulting groupwise proximity measures, we can understand how LitBoost borrows strength between groups under different values of σ^2.

We once again run 20 simulations, sampling a new data set according to (12), with newly sampled parameters $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}$ each time. We vary σ^2 using the values $(0.01, 1, 5)$, and let the number of observations per group go from $N = 10$ to $N = 320$. We use $\sigma_\varepsilon^2 = 0.1$ for every group. We report the results for LitBoost in two variants, one that includes one-hot encoding only and one that includes both two- and three-hot encoding. We also report the results when one general pairwise interaction is allowed, in which case we compare with EBM instead of GAM.

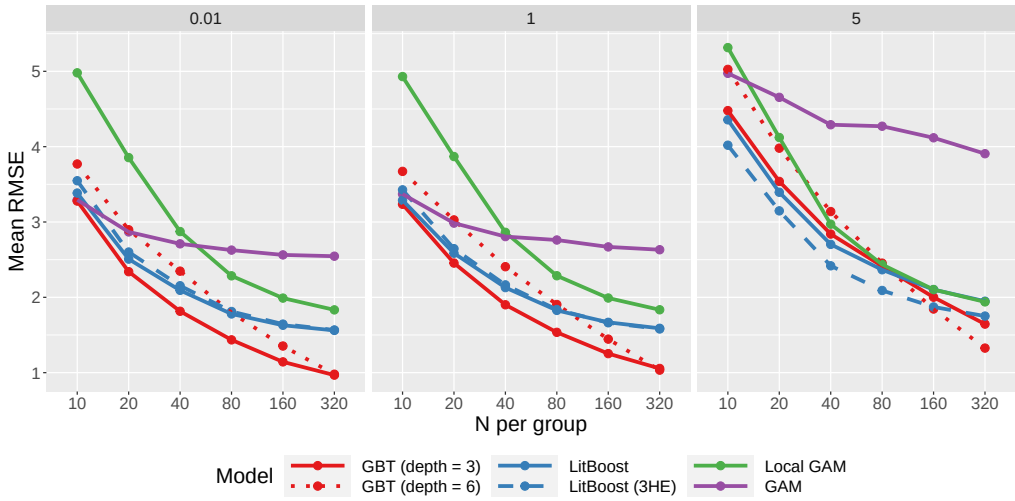

Figure 11: Mean RMSE of 20 simulations with a varying number of observations per group for GBT, GAM, Local GAM, and LitBoost with and without three-hot encoded variables (3HE). The three panels display simulations with $\sigma^2 = 0.01$ (left), $\sigma^2 = 1$ (middle) and $\sigma^2 = 5$ (right).

5.3. Results

The results are reported as a function of N per group in Figure 11. Similar to the results on the Oslo housing data set, the Local GAM displays poor performance when the number of observations per group is small, but it gets increasingly precise with growing N. LitBoost, both with and without the additional three-hot encoded variables, outperforms the Local GAM but struggles to beat GBT due to the interaction term in the true model. Interestingly, the GAM delivers results on par with the GBT and LitBoost when N is small, but the latter two models improve significantly as N is increased.

When we increase σ^2, thus making the clusters less distinct, the overall performance of all models is reduced compared to the case where $\sigma^2 = 0.01$. Interestingly, while the GAM performs considerably worse compared to the other models for higher σ^2, LitBoost improves compared to the other models. The regularizing effect of the interaction constraints may hinder the model from overfitting to noise in the data set, explaining why LitBoost outperforms the

full complexity model GBT. There is no improvement in performance when including three-hot encoded variables for $\sigma^2 = 0.01$ and $\sigma^2 = 1$, but a slight improvement is observed for $\sigma^2 = 5$.

Figure 12 displays the result when one additional pairwise interaction is allowed in LitBoost, and EBM is used for comparison instead of GAM. The interaction in LitBoost is again detected using the data-driven technique described in Section 4.

LitBoost performs even more impressive in this setup, generally outperforming the state-of-the-art GBT. Given that the true simulation setup only contains a single interaction between x_1 and x_2, and that the data-driven interaction detection method can detect this most of the time, the constraints imposed on LitBoost are, in this case, advantageous compared to GBT that is more likely to adapt to noise in the data sets. Again, both the EBM and the Local EBM perform significantly worse than LitBoost, highlighting the strengths of LitBoost also when additional interactions are allowed in the model.

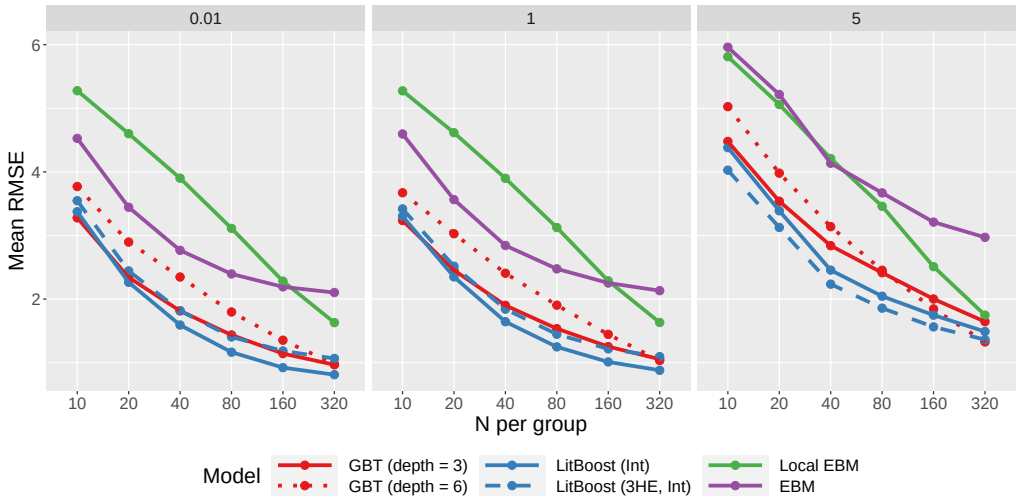

Figure 12: Mean RMSE of 20 simulations with a varying number of observations per group for GBT, EBM, Local EBM, and LitBoost that allow for one additional interaction. The three panels display simulations with $\sigma^2 = 0.01$ (left), $\sigma^2 = 1$ (middle) and $\sigma^2 = 5$ (right).

5.4. Shape Functions

Figure 13 shows a plot of $(x_1, f_1(x_1))$ for the GAM, Local GAM, and LitBoost when trained on the Friedman data set with $\sigma^2 = 1$ and $N = 1000$ observations per group. While the GAM enforces similar shape functions for all the nine groups, the Local GAM creates completely local effects, and the LitBoost shape functions behave as a regularized version of the Local GAM shape functions. The shape functions look reasonable compared to the true data generating mechanisms, as displayed in Figure 10. The presence of the three clusters, which we introduced through choices of hyperparameters in the data generating mechanism, is also clearly captured in the shape functions from LitBoost.

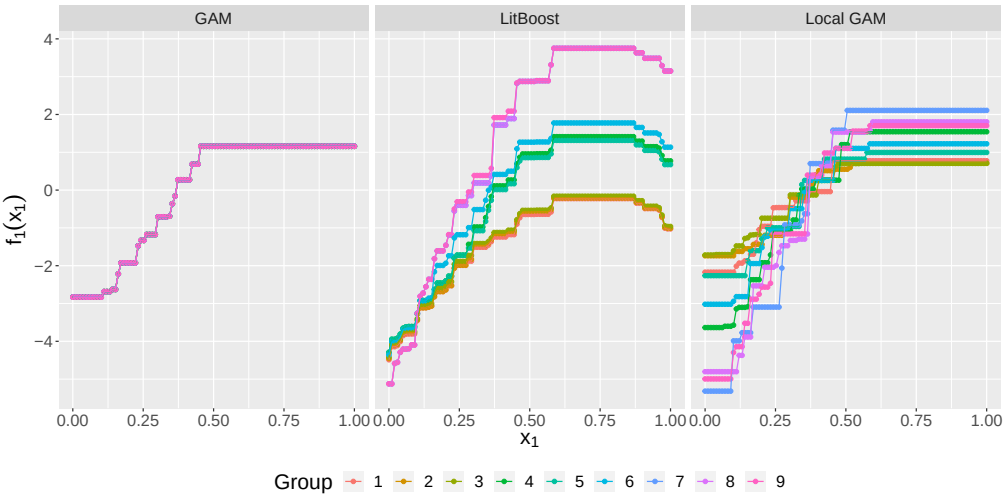

Figure 13: Shape functions for $f_1(x_1)$ for the simulated Friedman data set with $\sigma^2 = 1$ and $N = 1000$ observations in each group.

5.5. *Groupwise Proximity Measure*

Figure 14 shows the 9×9 groupwise proximity matrix for the 9 different groups in the simulated data set, averaged across 100 simulations within each σ^2 value. The clusters of groups introduced in the data generating process are, to some degree, captured by the groupwise proximity matrix. The cluster comprising groups $4, 5$ and 6 has higher internal proximity than the other clusters. For instance, $H(G_4, G_6) = 0.96$ in the simulation with $\sigma^2 = 0.01$, indicating that Group 4 shares data with group 6 almost as often as it shares with itself. This trend is less clear when $\sigma^2 = 5$, which can be attributed to the generally higher noise level in the data generating process and thus less distinct cluster structure.

Furthermore, the cluster of groups $1, 2$ and 3 rarely shares information with the cluster of groups $7, 8$ and 9. This is an intuitive result, given that the data from these two clusters are generated with very different parameters. The clusters are most distinct when σ^2 is low, as there is little noise in the data generating process.

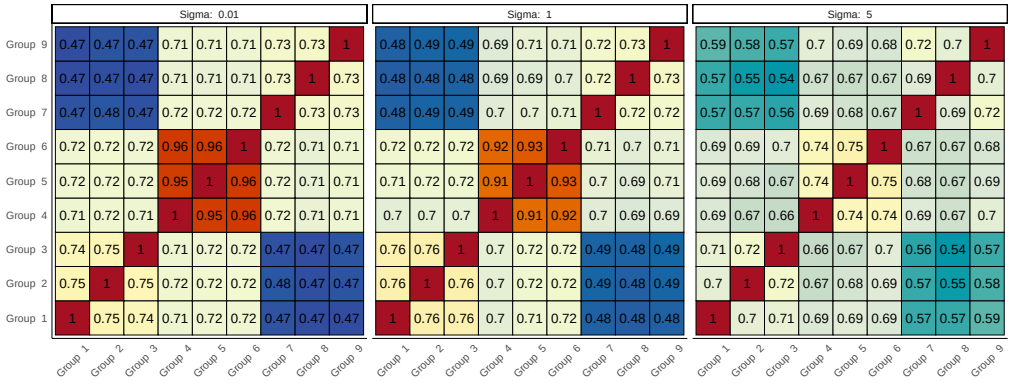

Figure 14: Groupwise proximity measures for the synthetic Friedman data simulation. A high value between group i and group j shows that observations from these groups often end up in the same leaf, indicating that they borrow strength from each other to a high degree. The three panels show simulation results with different σ^2 values. Each 9×9 matrix shows the average across 100 simulations with $N = 100$ observations from each group.

6. Conclusion

Due to their flexible nature and high predictive accuracy, tree-based supervised machine learning methods are popular among researchers and data scientists. When faced with data sets that include data from K different (and known) groups, we have studied three possible modeling approaches from the existing literature:

1. Train a Gradient Boosted Trees (GBT) model, which generally yields accurate predictions but is hard to interpret.
2. Train a Generalized Additive Model (GAM), which is easy to interpret, but does not necessarily adapt well to each group in the data set.
3. Train a Local GAM, i.e., a set of local models that will be specifically tailored to each city district at the cost of only using a fraction of the available data in each model.

To reap the interpretability benefits of a GAM while getting accuracy closer to GBT, we propose Locally Interpretable Tree Boosting (LitBoost). This hybrid option trains K local models – one per group in the data set – while still borrowing strength from the other groups in the data set during training. To achieve this, we modify the standard GBT framework by imposing interaction constraints that only allow interactions that involve the categorical group variable x_c. By doing so, we can achieve the favorable interpretability attributes of a GAM (or EBM), yet may get closer to the prediction accuracy of a full complexity GBT.

The results on $N = 14\,382$ observations from the Oslo (2018) data set indicate that LitBoost performs close to the full complexity GBT, often considered the state-of-the-art in house price prediction problems, and significantly better than a traditional GAM. The relative performance of LitBoost is best when the number of observations in each group is small, whereas the Local GAM improves its accuracy as N per city district increases. Similar results are achieved in an extensive simulation study on a synthetic data set, where we control the variance between the groups in the data-generating process. LitBoost outperforms GBT on the synthetic data set when we allow

for one interaction, which is expected since the underlying data-generating process only contains one interaction. An inspection of the shape functions from LitBoost on the synthetic data set indicates that the model correctly identifies the local effects for each group. The borrowing of strength across groups helps LitBoost to be more robust to overfitting to group-specific effects, which is also evident when studying the shape functions for each group. We also introduce the groupwise proximity measure, which quantifies the degree of data sharing between groups during training. This serves as another diagnostic tool that can be used to get some insight into the behavior of the LitBoost model.

In this work, we focused on GAM and EBM as base models, but recent research by Mayer et al. (2022) has indicated some interesting links between black box models like GBT and the class of Structured Additive Regression (STAR) models. An interesting way forward is to apply the idea of jointly training local models with STAR as base models, which could yield higher predictive accuracy while maintaining some interpretability benefits. Furthermore, investigating methods to scale up the multi-hot encoded variables efficiently is also something that could improve performance further, as the number of new columns quickly explodes if we go beyond two- or three-hot encoding. Finally, the groupwise proximity measure used to analyze the similarities between groups could be further improved by extending its applicability to be comparable across simulations and models.

While the application in this work is centered around house price prediction, this work also contributes to the larger and ongoing debate regarding the existence of a trade-off between interpretability and accuracy in machine learning methods. More specifically, this work builds on the recent and significant research by Lou et al. (2013), Chang et al. (2021), Nori et al. (2019), and Mayer et al. (2022) to bridge the gap between the modern class of methods based on gradient boosting and the traditional additive models presented by Hastie and Tibshirani (1986). The results presented here demonstrate that the performance of fully interpretable models such as GAMs can be even closer to that of full complexity models with careful engineering based on domain knowledge.

6.1. Acknowledgements

The authors are grateful to Eiendomsverdi AS for providing the data set and valuable domain knowledge. We are thankful to the reviewers for valuable feedback that strengthened the initial version of the paper. The research is funded by Norwegian Research Council through the Industrial PhD grant number 322779 to AH and through the Center of Excellence grant 332645 (*Integreat*) to IS and JP.

References

Y. Freund and R. E. Schapire (1996). "Experiments with a new boosting algorithm". In: *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*. Morgen Kaufmann Publishers Inc, pp. 148–156.

T. Chen and C. Guestrin (2016). "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

B. Goodman and S. Flaxman (Oct. 2017). "European Union Regulations on Algorithmic Decision-Making and a "Right to Explanation"". In: *AI Magazine* 38.3, pp. 50–57.

K. Coussement and D. F. Benoit (2021). "Interpretable data science for decision making". In: *Decision Support Systems* 150. Interpretable Data Science For Decision Making, p. 113664.

S. M. Lundberg and S.-I. Lee (2017). "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., pp. 4765–4774.

M. T. Ribeiro, S. Singh, and C. Guestrin (2016). ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, pp. 1135–1144.

C. Rudin (2019). "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead". In: *Nature Machine Intelligence* 1, pp. 206–215.

B. Glumac and F. Des Rosiers (2021). "Practice briefing – Automated valuation models (AVMs): their role, their advantages and their limitations". In: *Journal of Property Investment & Finance* 39.5, pp. 481–491.

M. Steurer, R. J. Hill, and N. Pfeifer (2021). "Metrics for evaluating the performance of machine learning based automated valuation models". In: *Journal of Property Research* 38.2, pp. 99–129. eprint: `https://doi.org/10.1080/09599916.2020.1858937`.

T. Hastie and R. Tibshirani (1986). "Generalized Additive Models". In: *Statistical Science* 1.3, pp. 297–310.

S. Wood (2006). *Generalized Additive Models: An Introduction With R*. Vol. 66, p. 391.

Y. Lou, R. Caruana, and J. Gehrke (2012). "Intelligible Models for Classification and Regression". In: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '12. Beijing, China: Association for Computing Machinery, pp. 150–158.

Y. Lou et al. (2013). "Accurate Intelligible Models with Pairwise Interactions". In: *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '13. Chicago, Illinois, USA: Association for Computing Machinery, pp. 623–631.

J. Friedman (1991). "Multivariate Adaptive Regression Splines". In: *The Annals of Statistics* 19.1, pp. 1–67.

— (2001). "Greedy Function Approximation: A Gradient Boosting Machine". In: *The Annals of Statistics* 29.5, pp. 1189–1232.

G. Ke et al. (2017). "Lightgbm: A highly efficient gradient boosting decision tree". In: *Advances in neural information processing systems* 30, pp. 3146–3154.

H. Nori et al. (Sept. 2019). *InterpretML: A Unified Framework for Machine Learning Interpretability*. ArXiv.

R. Agarwal et al. (2021). "Neural Additive Models: Interpretable Machine Learning with Neural Nets". In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer et al.

Z. Yang, A. Zhang, and A. Sudjianto (2021). "GAMI-Net: An Explainable Neural Network based on Generalized Additive Models with Structured Interactions". In: *Pattern Recognition* 120, p. 108192.

L. Fahrmeir, T. Kneib, and S. Lang (2004). "Penalized Structured Additive Regression for Space-Time Data: A Bayesian Perspective". In: *Statistica Sinica* 14.3, pp. 731–761.

M. Mayer et al. (2022). "Machine Learning Applications to Land and Structure Valuation". In: *Journal of Risk and Financial Management* 15.5.

F. Sigrist (2022). "Gaussian Process Boosting". In: *Journal of Machine Learning Research* 23.232, pp. 1–46.

A. Zeileis, T. Hothorn, and K. Hornik (2008). "Model-Based Recursive Partitioning". In: *Journal of Computational and Graphical Statistics* 17.2, pp. 492–514.

L. Breiman et al. (1984). *Classification and Regression Trees*. Wadsworth and Brooks.

B. M. Greenwell (2017). "pdp: An R Package for Constructing Partial Dependence Plots". In: *The R Journal* 9.1, pp. 421–436.

L. Breiman (2001). "Random forests". In: *Machine Learning* 45, pp. 5–23.

M. J. Bailey, R. F. Muth, and H. O. Nourse (1963). "A regression method for real estate price index construction". In: *Journal of the American Statistical Association* 58.304, pp. 933–942.

S. Rosen (1974). "Hedonic prices and implicit markets: product differentiation in pure competition". In: *Journal of Political Economy* 82.1, pp. 34–55.

W. K. O. Ho, B.-S. Tang, and S. W. Wong (2020). "Predicting property prices with machine learning algorithms". In: *Journal of Property Research*.

J. Kim et al. (2021). “Machine-Learning-Based Prediction of Land Prices in Seoul, South Korea”. In: *Sustainability* 13.23, pp. 202–211.

A. Hjort et al. (2022). “House price prediction with gradient boosted trees under different loss functions”. In: *Journal of Property Research* 39.4, pp. 1–27.

C.-H. Chang et al. (2021). “How Interpretable and Trustworthy Are GAMs?” In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. KDD ’21. Virtual Event, Singapore: Association for Computing Machinery, pp. 95–105.

Appendix A. Code

Code for for training and prediction with LitBoost in `R` can be found at `https://github.com/adhjort/LitBoost`. The data generating process from the simulation study in this work is also available here.

Appendix B. Shape Functions, The Oslo Data Set

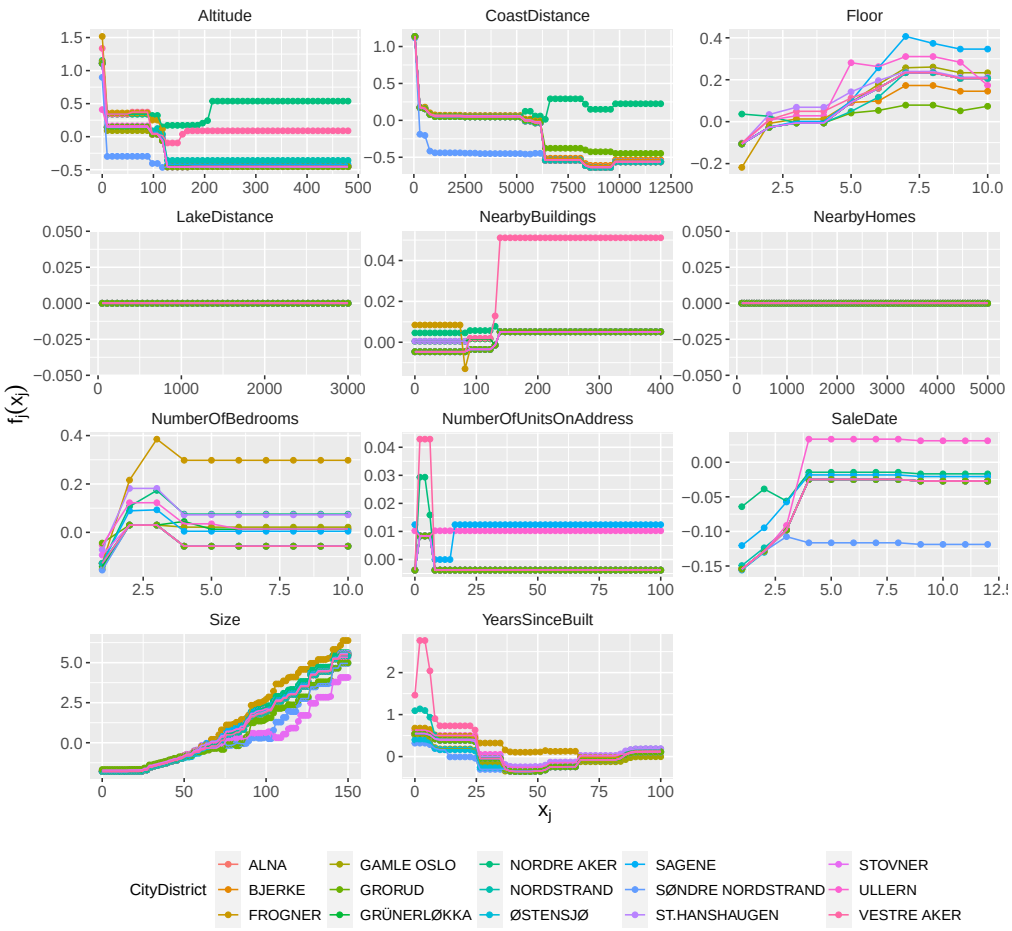

Figure B.15: All shape functions $(x_j, f_j(x_j))$ for LitBoost.

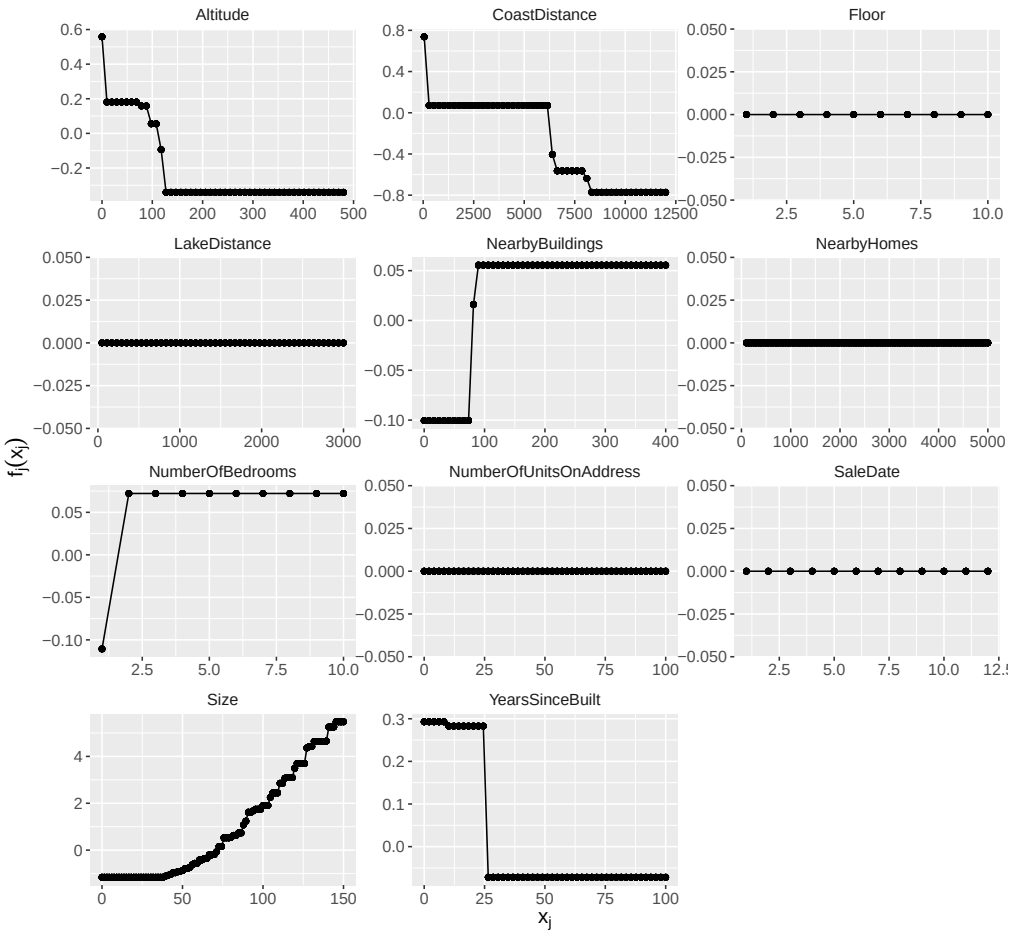

Figure B.16: All shape functions $(x_j, f_j(x_j))$ for GAM.

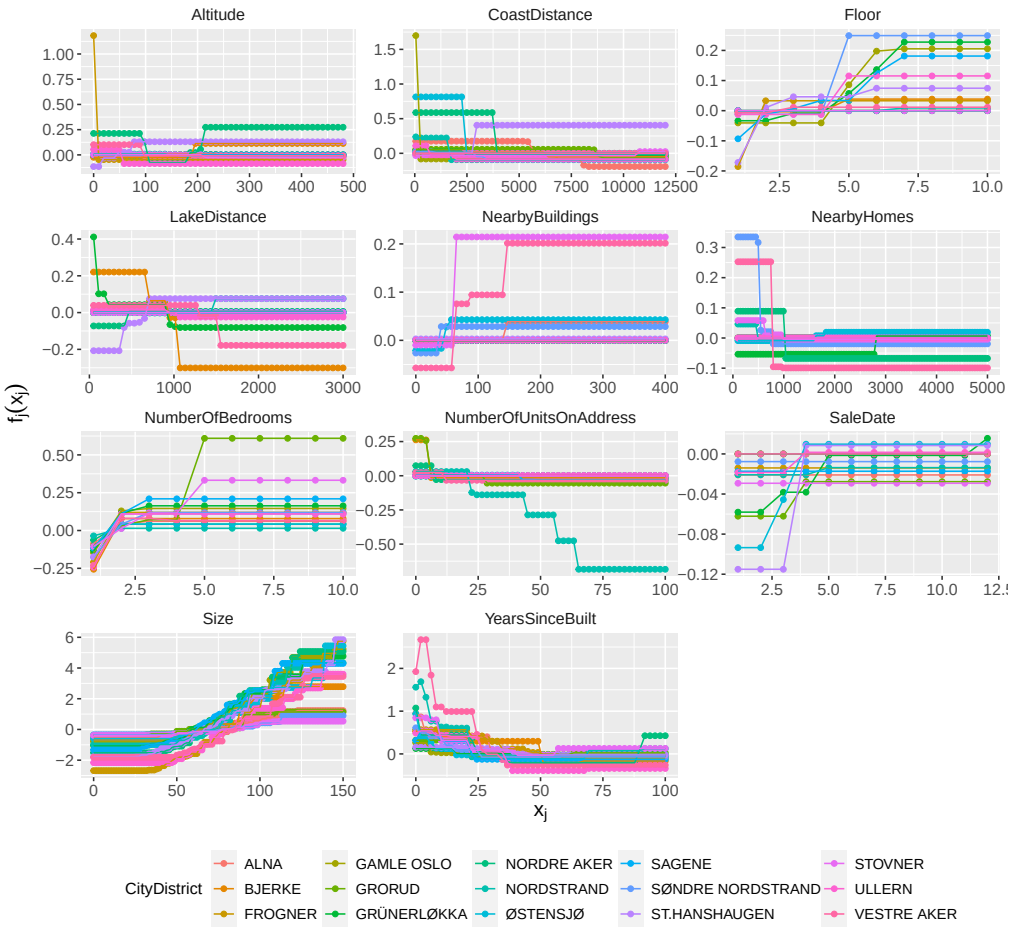

Figure B.17: All shape functions $(x_j, f_j(x_j))$ for the Local GAM.