Journal of Property Research

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/rjpr20

House price prediction with gradient boosted trees under different loss functions

Anders Hjort, Johan Pensar, Ida Scheel & Dag Einar Sommervoll

To cite this article: Anders Hjort, Johan Pensar, Ida Scheel & Dag Einar Sommervoll (2022): House price prediction with gradient boosted trees under different loss functions, Journal of Property Research, DOI: 10.1080/09599916.2022.2070525

To link to this article: https://doi.org/10.1080/09599916.2022.2070525

JOURNAL OF PROPERTY RESEARCH
https://doi.org/10.1080/09599916.2022.2070525

House price prediction with gradient boosted trees under different loss functions

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

[a]Department of Mathematics, University of Oslo, Oslo, Norway; [b]School of Economics and Business, Norwegian University of Life Sciences and NTNU Trondheim Business School, Norway; [c]Eiendomsverdi AS, Oslo, Norway

ABSTRACT
Many banks and credit institutions are required to assess the value of dwellings in their mortgage portfolio. This valuation often relies on an Automated Valuation Model (AVM). Moreover, these institutions often report the models accuracy by two numbers: The fraction of predictions within $\pm 20\%$ and $\pm 10\%$ range from the true values. Until recently, AVMs tended to be hedonic regression models, but lately machine learning approaches like random forest and gradient boosted trees have been increasingly applied. Both the traditional approaches and the machine learning approaches rely on minimising mean squared prediction error, and not the number of predictions in the $\pm 20\%$ and $\pm 10\%$ range. We investigate whether introducing a loss function closer to the AVMs actual loss measure improves performance in machine learning approaches, specifically for a gradient boosted tree approach. This loss function yields an improvement from 89.4% to 90.0% of predictions within $\pm 20\%$ of the true value on a data set of $N = 126719$ transactions from the Norwegian housing market between 2013 and 2015, with the biggest improvements in performance coming from the lower price segments. We also find that a weighted average of models with different loss functions improves performance further, yielding 90.4% of the observations within $\pm 20\%$ of the true value.

ARTICLE HISTORY
Received 29 October 2021
Accepted 20 April 2022

KEYWORDS
Appraisal; valuation; quantitative analysis; real estate; appraisal accuracy

1 Introduction

The housing market is keenly watched by policymakers and the general populace. The financial crisis of 2007 told us that a housing market bust come with dire consequences for the larger economy. The bust affected consumer spending and solvency of banks as houses served as mortgage collaterals. As a protective measure, banks in many countries are required to assess their risk exposure regularly by monitoring the value of the houses in their mortgage portfolio. One way of efficiently doing this is by using an Automated Valuation Model (AVM), that is, a statistical model that estimates the price of a dwelling or a portfolio of dwellings.

Historically, AVMs have fallen into two groups. The first and foremost is made up of hedonic regression models (Goodman and Thibodeau (2003); Rosen (1974)), where house prices are sought to be predicted by regressing transaction prices (or the log of transaction

CONTACT Anders Hjort anderdh@math.uio.no

prices) on a set of dwelling characteristics like size, number of bedrooms, etc. The hedonic models face three main challenges. The first is that the linear assumption can be unrealistic. Such concerns are usually addressed by adding squares and cubes of variables as well as interaction terms. The second challenge is possible omitted variable bias as characteristics that have a bearing on transaction prices may be unavailable due to data limitations. Finally, the third challenge pertains to data quality. Even if a large array of explanatory variables are available in the data, oftentimes quite a few tend to be of poor quality. They may be missing all together or be poorly measured. This curbs the upside of more refined models.

The second group of models tend to circumvent the challenge of omitted and poorly measured determinants of transaction prices by considering repeated sales only (Bailey et al. (1963); Case et al. (1993)). This allows us to observe price changes of identical objects, provided that the dwelling is unchanged between sales. This approach has apparent challenges of its own. It may be hard to know or check that a dwelling is indeed unchanged between sales. Moreover, the notion of unchanged is not straightforward, especially if the holding time between sales is long. A house that was new 10 years ago may appear dated, even in the case where the owners have put great care to counteract natural tear and wear.

This latter point has led to hybrid models (Peter et al. (1998); Quigley (1995)) that seek to utilise the strengths of both approaches; the repeated sales models independence of hedonic characteristics and the hedonic model's use of all observations. A variety of spatial methods has also been applied in order to capture the heterogeneity that is often present in house price data (Clapp et al. (2004); Páez et al. (2010)).

Recent years have seen machine learning models replace or supplement hedonic models. Machine learning models have gained enormous popularity due to its high predictive accuracy and flexible nature that often manages to capture non-linear relationships in ways that some traditional statistical models struggle with. A popular class of methods are tree-based models that combine multiple decision trees, with random forest (Breiman (2001)) and gradient boosted trees (Freund and Schapire (1996); Friedman et al. (2000)) as the most notable examples. Artificial neural networks is another class of methods that is widely used, including specialised networks like long short-term memory networks (Hochreiter and Schmidhuber (1997)) and convolution neural networks (LeCun et al. (1999)).

Tree-based models such as random forest and gradient-boosted trees have proven to be good at predicting property prices (Baldominos et al. (2018); Ho et al. (2020); Kim et al. (2021); Park and Bae (2015) and Sing et al. (2021)). Approaches based on neural networks have also lead to promising results in the realm of house price prediction (Xiaojie and Zhang (2021), Terregrossa and Ibadi (2021)). In particular, some methods use neural networks to process images of a dwelling and use this as input in other machine learning models in order to capture visual characteristics of the dwelling (Lim et al. (2016); Poursaeed et al. (2018) and Zillow (2019)) or the surrounding areas (Yencha (2019)). Other machine learning techniques that have been used for house price prediction include genetic algorithms (Sommervoll and Sommervoll (2019)), support vector machines (Wang et al. (2014)) and fuzzy logic (Giudice et al. (2017)). A comparison of modern approaches to house price prediction is presented in McCluskey et al. (2013), while a broader overview of machine learning in the real estate industry is presented by Viriato (2019).

Machine learning models rely on minimisation of a user-defined loss function. A common default choice is the squared error (SE) loss function, which minimises the sum of the squared distance between a prediction and its corresponding observation. Although using the squared error loss function is sensible in many applications in statistics and machine learning, some problems would benefit from more tailor-made solutions. For instance, Huber (1964) introduced the Huber loss, a novel loss function that serves as a compromise between squared error and absolute error loss. Furthermore, Varian (1975) and Cain and Janssen (1995) both applied asymmetric loss functions to the problem of house price prediction for cases where one want to penalise underestimation harder than overestimation. The idea of tailoring a loss function for a specific problem can also be seen in other machine learning applications: Barron (2019) introduced a novel loss function for the purpose of image recognition, while Gabriel et al. (2017) proposed a tailored loss function that regularises neural networks and demonstrated its performance on both image recognition tasks and natural language processing tasks.

A widely used performance measure to quantify the performance of an AVM is the fraction of predictions that falls within a $\pm 20\%$ or $\pm 10\%$ range of the transaction price.[1] Going forwards we will refer to this as the 20% measure and 10% measure, or generally as the percentage measure. The European AVM Alliance (Nitschke et al. (2019)) refers to the percentage measure as the most commonly used measure for dispersion. Furthermore, they recommend a range of at least 10%, and note that 20% is the most commonly quoted range. Zillow, an American property technology company, includes both the 10% measure and 20% measure when reporting the accuracy of their AVM, while the American credit rating agency Fitch Ratings Inc. acknowledges that the percentage measure is 'the most standard measurement utilized in the industry' (FitchRatings (2019)).

This is our point of departure: Will a different loss function than the standard SE loss lead to better model performance when using the percentage measure as our target measure? We study this empirically by comparing two different loss functions for the training of machine learning-based models for house price prediction. We note that we are not the first one to suggest this; Miriam et al. (2021) comment that the choice of performance measure should be made together with a decision on the loss function, but for practical reasons their focus lie solely on exploring different measures. In this paper, we will do the opposite: With a predetermined measurement of model performance (the percentage measure), we ask whether we can improve the model's performance by carefully choosing an appropriate loss function.

Our contribution relates to two strands of literature. In the aftermath of the 2007–2008 financial crisis the relationship between appraisals, AVMs and bank's mortgage risk exposure has received more academic attentions (Demiroglu and James (2018); Kruger and Maturana (2020)). The improvement of AVM performance is therefore of great interest to banks. We also contribute to the general machine learning literature: For the lion share of applications, the interplay between the performance measure and the loss function is a best implicit. Tree-based machine learning techniques are on the way to becoming a 'swiss army knife', where implementations are easily available through off-the-shelf software packages that require little to no tuning. This is likely to result in an overuse of the standard loss functions where there is potential for significant improvement by introducing more tailored loss functions.

More specifically, for our data set we find that introducing a tailored loss function improves performance from 89.4% to 90.0% when using the 20% measure to evaluate model performance. Furthermore, a weighted average of two models – one with the standard squared error loss and one with a squared percentage error (SPE) loss – yields a further improvement, giving a performance of 90.4% on the same measure. We find that the improvement in performance is highest for the lower price segments, with a slight decline in performance for the highest price segments.

The remainder of the paper is organised as follows: In Section 2 we present the housing market data set used in this paper. It consists of all housing market transactions ($N = 126\,719$) in the counties Oslo, Viken, Trøndelag and Innlandet in the time period 2013–2015. In Section 3 we present four different models – linear regression, nearest neighbour regression, random forest and gradient boosted trees – and study their performance in predicting house prices with the standard loss function. In addition, we explore the role of the loss function in gradient boosted trees and argue that a relative loss function is more appropriate for the given performance measure. In Section 4 we compare the gradient boosted trees trained with SE loss and SPE loss, and find that the latter improves the model when using the 20% measure as performance measure. Finally, in Section 5 we investigate whether a combination of the two models can lead to an even bigger improvement. In Section 6 we conclude and outline some possible ideas for further research.

2 Data

2.1 About the Norwegian housing market

Norway has a strong tradition for owning rather than renting. The home ownership rate in Norway is 72%, slightly above the OECD average. However, most people cannot afford to own outright: 69% of owners in Norway have used a mortgage to finance the acquisition, compared with an OECD average of 36%.[2] This high mortgage percentage creates a tight connection between the housing market and the banking sector in Norway. Governmental regulations determine the size of the mortgage provided by the banks based on the estimated value of the dwelling and the buyers' income and equity. Most importantly, the potential borrower can at most borrow five times their annual income and the mortgage can not exceed 85% of the estimated dwelling value. In most cases banks take the dwelling as mortgage collateral, and it is therefore crucial for banks to estimate the value of the dwelling regularly for risk management purposes. Estimates of the dwelling value is usually updated quarterly by an AVM.

When a potential buyer has been granted a mortgage from the bank, they are ready to submit a bid for a dwelling on the market. The overwhelming majority of transactions in the Norwegian housing market take place as open auctions where anyone can participate. An asking price is determined by the seller, typically guided by a valuation conducted by a professional appraiser or an assessment by a real estate agent. The auction normally starts – and finishes – the day after the open house. This swift decision making process is another characteristic of the Norwegian housing market, which is in stark contrast to other markets where the transaction requires lengthy negotiations before an agreement is reached.

Another characteristic of the Norwegian housing market is the spatial heterogeneity. Some parts of the country has limited housing market activity due to mountain ranges, forestry and farming. This creates big differences in both price levels and market activity between urban and rural areas. 52% of transactions in 2021 occurred in the four biggest cities (Oslo, Bergen, Trondheim and Stavanger), despite making up only an estimated 25% of the housing stock.[3]

2.2 *About the data set*

We use a data set containing all arms' length transactions from the Norwegian housing market between 2013 and 2015 from the four counties Oslo, Viken, Trøndelag and Innlandet ($N = 126719$).[4] This includes Oslo and Trondheim, the largest and third-largest cities in Norway. Figure 1 shows a map of these four counties with mean price per square metre at municipality level. It also shows the distribution of price per square metre per estate type, measured in Norwegian kroners (NOK).[5]

Table 1 shows summary statistics for the variables in the data set. The data set includes several variables that are of high importance to homeowners, such as size, floor number, age of the building and the number of bedrooms. It also includes information about the area around the dwelling in question, such as distance to the nearest lake and the number of homes and other buildings in the surrounding area. Further details on the data preparation procedure is found in Appendix A.

We split the full data set into two subsets: A training set (75% of the data) used to train a model and a test set (25% of the data) used to test or evaluate the model. The partition is done randomly, and not based on the values of the variables. Since the performance of the model is vulnerable to how exactly the training and test split is done, we will

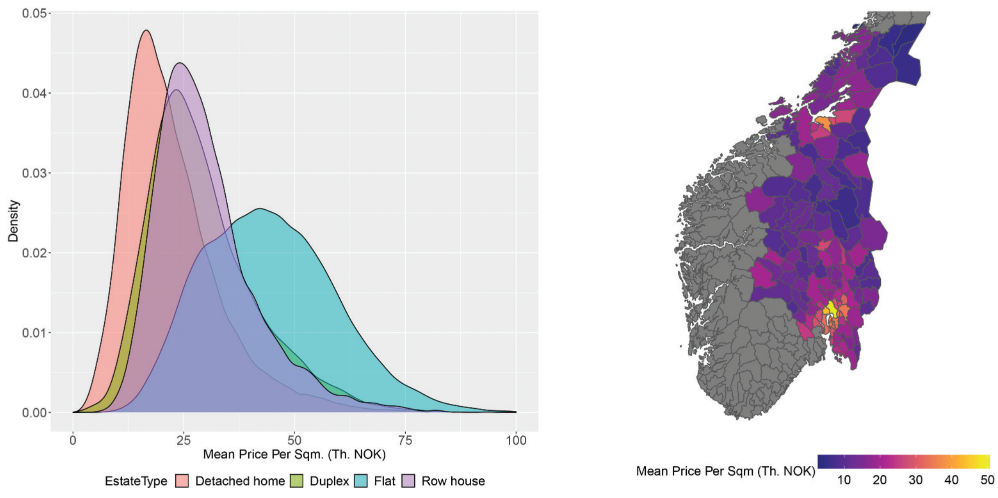

Figure 1. Left plot: The distribution of price per square metre (in thousands NOK) per estate type in the data set. Right plot: Mean price per square metre (in thousands NOK) per municipality in the data set.

Table 1. 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.)	3.16	1.78	0.14	38.00	Numerical
Estate Type[1]	–	–	–	–	–	Categorical
Municipality[2]	–	–	–	–	–	Categorical
City District[3]	–	–	–	–	–	Categorical
Sale Date	quarters	6.53	3.42	1.00	12.00	Categorical
Longitude	degrees	266,181	22,068	109,164	451,605	Numerical
Latitude	degrees	6,713,095	149,240	6,535,384	7,222,338	Numerical
Altitude	m	109.51	86.18	0	989.00	Numerical
Size [4]	m^2	97.25	52.92	15.00	642.00	Numerical
Floor[5]	–	1.99	1.56	−4	14	Categorical
Bedrooms	–	2.41	1.17	0	58	Categorical
Footprint Area	m^2	33.82	64.32	0.00	693.00	Numerical
Dwelling Age	years	46.41	33.16	0	415.00	Numerical
Lot Area	m^2	397	2,458	0	109,786	Numerical
Balcony[6]	–	0.65	0.48	0	1	Binary
Elevator[6]	–	0.19	0.39	0	1	Binary
Lot Slope	degrees	5.42	4.69	0.00	41.00	Numerical
Lot Slope Direction	degrees	181.08	84.28	0.00	360.00	Numerical
Units On Address[7]	–	10.55	20.11	0.00	274.00	Numerical
Coast Distance	m	4,824	3,968	0	16,000	Numerical
Lake Distance	m	1,107	757	0	5,885	Numerical
Sunset Hour[8]	hours	20.83	0.67	15.57	24.00	Numerical
Close Neighbourhood Homes[9]	–	1,305	1,484	1	6,746	Numerical
Extended Neighbourhood Homes[10]	–	3,124	3,697	1	14,936	Numerical
Close Neighbourhood Buildings[9]	–	105	108	0	1,323	Numerical
Extended Neighbourhood Buildings[10]	–	268	256	0	1,796	Numerical

[1]There are four distinct estate types: flat, duplex, row house, detached home.
[2]There are 135 distinct municipalities in the data set.
[3]There are 21 distinct city districts in the data set. Dwellings outside of the city do not have a value.
[4]The living area in m^2.
[5]If the dwelling has multiple floor, this variable will be the lowest floor. For detached homes this is set to 1.
[6]In cases where the information is missing, this is set to 0.
[7]In some cases, e.g. in apartment buildings, multiple dwellings have the same address.
[8]The time of sunset as of July 1st.
[9]Norway is divided into squares of 250 m × 250 m. This variable counts the number of homes or other buildings (stores, schools, churches etc.) in all the adjacent squares to the square that the target dwelling is in, that is, the 8 neighbouring squares.
[10]This counts the number of homes or other buildings (stores, schools, churches etc.) in the adjacent eight 250 m × 250 m squares, as well as the neighbours of these eight squares. In total, this includes 25 squares.

throughout this paper consequently conduct 100 random splits into training and test data and report the mean and standard deviation of the performance measure across for the 100 models trained and tested on these sets.

In cases where we need to find optimal hyper parameters for models we also separate a fraction (25%) of the training set into a validation set. When the optimal hyper parameters are found we use these on the full training set.

3 Benchmark models

We aim to make a statistical model that estimates the sale price $y_i \in \mathbb{R}^+$ of a dwelling given its features $\mathbf{x}_i \in \mathbb{R}^d$. We denote this estimate $\hat{y}_i = f(\mathbf{x}_i)$. In this section, we will consider four models for the function $f(\mathbf{x}_i)$: Linear regression, nearest neighbour (NN) regression, random forest and gradient boosted trees. This section gives a brief introduction to each of the methods.

3.1 Linear regression

For the linear regression, we have

$$f(\mathbf{x}_i) = \sum_{j=1}^{d} x_{ij} \cdot \beta_j$$

where $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{id}]$ is the vector of features describing dwelling i and $\beta = [\beta_1, \beta_2, \ldots, \beta_d]$ is found via maximum likelihood estimation. The features included in $\mathbf{x}_i$ are the ones listed in Table 1.

3.2 Nearest neighbour regression

Nearest Neighbour (NN) regression algorithms refer to algorithms that make predictions based on similar objects. One such algorithm is k Nearest Neighbour (k-NN) regression, which finds the k closest objects (in feature space) in the data set and use those k objects to estimate the unknown price of the target object, typically through a simple mean of the objects. The estimate is thus

$$f(\mathbf{x}_i) = \frac{1}{k} \sum_{\mathbf{x}_i \in N_k(\mathbf{x}_i)} y_i$$

where $N_k(\mathbf{x}_i)$ is a neighbourhood of the k nearest neighbours around $\mathbf{x}_i$. There are multiple ways of determining the neighbourhood in $\mathbb{R}^d$, but the L_2 norm is often used. This highlights an important practical challenge with k-NN regression: If the data is not preprocessed properly (scaling and centring) there is a risk that one or a few of the d features will dominate the neighbourhood calculation due to difference in magnitude between the features.

Another practical challenge is to calculate an $N \times N$ distance matrix when N becomes large. There exists various approximation methods, often denoted Approximate Nearest Neighbour (ANN) techniques. In this paper, we will use the R package RANN (Sunil et al. (1998); Sunil et al. (2019)) and L_2 norm for distance metrics. We centre all the variables by subtracting the mean value, and scale the variables by dividing them by their standard deviation. We find $k = 2$ to be optimal via a validation set.

3.3 Random forest

A random forest builds an ensemble of decision trees on bootstrapped samples of the training data and uses this ensemble to make predictions about new instances. A prediction from a single decision tree is denoted $h(x; q)$, where q represents the tree structure of the decision tree. This tree structure defines a series of splits that partition the feature space into J disjoint subsets $\mathcal{I}_1, \ldots, \mathcal{I}_J$. The subsets are found by subsequently making binary splits in the data set based on an information gain criterion. This process continues until it reaches a user specified max tree depth or another stopping criterion.

The subsets are represented as leaves on the decision tree, and each leaf j is equipped with a constant value $w(j)$ that serves as a prediction for new instances x for which we want to make a prediction. For a new instance, x_i the prediction becomes

$$h(x_i;q) = \sum_{j=1}^{J} w(j) \cdot \mathbb{I}\left(x_i \in \mathcal{I}_j\right).$$

A random forest is a collection of M such trees and the prediction is the average of all trees,

$$f(x_i) = \frac{1}{M}\sum_{m=1}^{M} h_m(x_i;q_m).$$

Breiman (2001) showed that aggregating multiple trees yields precise results if each tree is trained on a bootstrapped sample of the original data. Another important trait of a random forest model is that the model picks a random subset of features as candidates for each split, ensuring that the individual trees are as uncorrelated as possible. A major benefit of random forest is its ability to handle both numerical, categorical, and binary covariates with very little data preparation, and still produce precise estimates. Construction of a random forest involves some choices of hyper parameters, most notably the number of trees M to be constructed and the number of splits J in each tree. For detailed discussions about these choices, see Breiman (2001); Hastie et al. (2009) and Louppe (2014). We used $M = 500$ trees and a minimum node size of 10 observations, implicitly defining the J (the algorithm stops splitting when the number of observations in that node is 10).

3.4 Gradient boosted trees

Gradient boosted trees also combine multiple decision trees, but unlike random forest, where trees are trained in parallel on bootstrap samples of the original data set, the trees are trained in an sequential manner. The prediction of y_i after training M trees is expressed as

$$f_M(x_i) = \sum_{m=1}^{M} \eta \cdot h_m(x_i;q_m),$$

where $h_m(x_i;q_m)$ is a decision tree and $\eta \in (0,1]$ a shrinkage parameter that determines how quickly the method should converge. At every iteration, we find the tree structure q_m through minimisation of some loss function $L(y_i, \hat{y}_i^{(m-1)})$, where $\hat{y}_i^{(m-1)}$ is the current estimate of y_i after the previous iteration, that is,

$$\begin{aligned} q_m &= \arg \underset{q}{min}\{(L(y_i, f_m(x_i))\} \\ &= \arg \underset{q}{min}(L(y_i, f_{m-1}(x_i) + \eta \cdot h_m(x_i;q))). \end{aligned}$$

The term 'boosting' stems from the machine learning community, where it has been used about ensembles of weak learners, that is, predictive models with low accuracy. Surprisingly, combining many weak learners in an additive fashion produced impressive results, as demonstrated via the Adaboost algorithm presented by Freund and Schapire (1996). The idea has later been analysed from a statistical point of view by among others Friedman (2001) and Friedman et al. (2000), linking it to gradient descent methods. In recent years, gradient boosted trees have been become faster and more accessible through computationally efficient implementations, such as XGBoost (Chen and Guestrin (2016)), CatBoost (Prokhorenkova et al. (2018)) and LightGBM (Guolin et al. (2017)). In this paper, we will specifically consider the XGBoost implementation. We use $J = 3000$ trees and a learning rate $\eta = 0.01$. Each tree has a maximum depth $M = 6$.

3.5 Benchmark model performance

We applied the considered method on the data set presented in Section 2. Table 2 reports the Root Mean Squared Error (RMSE), Median Error (MdE), the 10% measure, the 20% measure and R^2 for each of the methods. We used the standard squared error loss function for XGBoost and random forest.

XGBoost achieves the best results on four of the five performance measures, achieving an RMSE of 16.2%, 65.7% on the 10% measure and 89.4% on the 20% measure . Random forest is the second best performer in terms of the 10% measure, the 20% measure and R^2, achieving close to XGBoost with 63.1%, 87.1% and 90.4%, respectively. The NN regression achieves the best MdE, but a significantly worse RMSE than the tree-based models. Finally, linear regression achieves the worst performance of the four models on all measures.

Figure 2 displays the relative errors (in percent of the sale price) from the four models. The figure shows the superiority of the tree-based models: While the mean of all the densities are all close to zero, the densities of random forest and XGBoost have a much narrower distribution, and a larger part of the area is between the stapled lines.

Table 3 displays model performance (measured by the 20% measure) per county. All of the models give the best performance in Oslo and the poorest performance in Innlandet. Even for XGBoost, the best model, the model performance ranges from 95.1% in Oslo to 75.1% in Innlandet. The performance correlates with the number of observations per county: Oslo has 48 143 transactions, Viken has 48 476 transactions, Trøndelag has 19 612 transactions and Innlandet just 10 488 transactions. It is worth noting that all the models perform consistently better in Oslo than Viken, despite having approximately the same number of observations. One likely explanation is the fact that dwellings in Oslo are more homogeneous than those in Viken, as there is a larger concentration of apartment buildings in Oslo. A map showing mean absolute prediction

Table 2. Root Mean Squared Error (RMSE), Median Error (MdE), the 10% measure, the 20% measure and R^2 for linear regression, nearest neighbour regression, random forest and XGBoost. The values indicate the mean and standard deviation across the 100 simulations with different training and test set. The best values for each performance measure is highlighted in boldface.

	RMSE (%)	MdE (%)	10% measure (%)	20% measure (%)	R^2(%)
Linear regression	32.7 (0.5)	−1.4 (0.2)	34.1 (0.2)	60.2 (0.3)	73.0 (0.43)
NN regression[1]	28.4 (0.9)	**0.0 (0.1)**	44.7 (0.2)	70.9 (0.2)	73.4 (0.47)
Random forest[2]	18.8 (0.4)	−1.2 (0.1)	63.1 (0.3)	87.1 (0.2)	90.4 (0.31)
XGBoost[3]	**16.2 (0.5)**	0.4 (0.1)	**65.7 (0.2)**	**89.4 (0.1)**	**91.9 (0.23)**

[1]We use $k = 2$, found to be optimal on a validation set consisting of 25% of the training data.
[2]We build 500 trees with minimum node size of 10 observations.
[3]We build 3000 trees, each with tree depth of 6 splits and learning rate of 0.01.

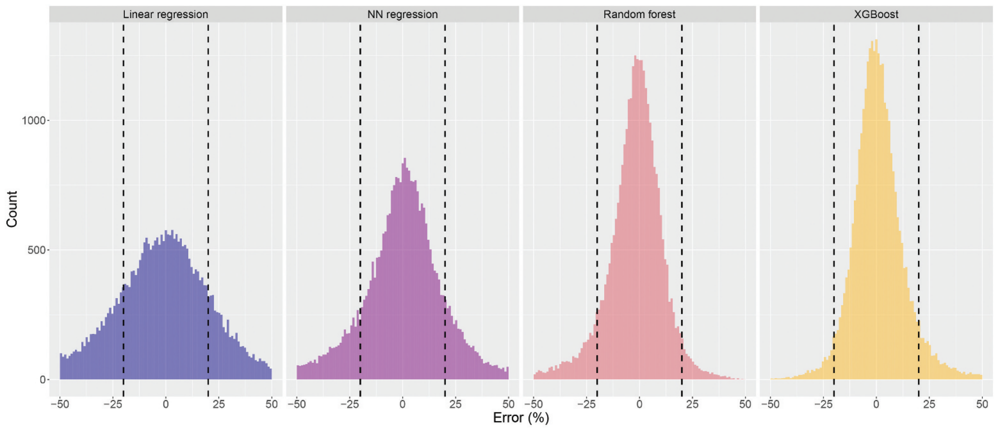

Figure 2. Histograms of relative errors (in percent) from the linear regression, NN regression, random forest and XGBoost. The vertical dotted lines indicate $\pm 20\%$. The values on the *y* axis are normalised such that the integral of each density equates to one.

error per municipality can be seen in Figure 3. It further strengthens our belief that random forest and XGBoost are the superior performers, but also highlights that these methods struggle to perform in some municipalities in the central part of the map. These are areas with very few observations, in many cases less than 10 observations per municipality.

3.6 The role of the loss function in XGBoost

Subsection 3.4 showed how XGBoost builds trees sequentially via minimisation of a loss function $L(y_i, \hat{y}_i)$, which is , typically, the SE loss in the regression setting. Most machine learning models apply the Squared Error (SE) loss, $L_{SE} = \frac{1}{2}(y_i - \hat{y}_i)^2$. XGBoost seeks to build the next tree $h_m(x_i; q)$ in a way that minimises a regularised loss function,

$$\begin{aligned}\mathcal{L}(y_i, \hat{y}_i^{(m)}) &= \sum_{i=1}^{N} L(y_i, \hat{y}_i^{(m)}) + \Omega(h_t; \lambda, \gamma) \\ &= \sum_{i=1}^{N} L(y_i, \hat{y}_i^{(m-1)} + \eta \cdot h_m(x_i; q)) + \Omega(h_t; \lambda, \gamma)\end{aligned} \tag{1}$$

Table 3. Model performance per county measured by the 20% measure for linear regression, nearest neighbour regression, random forest and XGBoost. The values indicate the mean and standard deviation across the 100 simulations with different training and test set.

	Innlandet	Oslo	Trøndelag	Viken
Linear regression	40.4 (0.9)	68.8 (0.4)	62.8 (0.6)	55.0 (0.4)
NN regression[1]	56.6 (0.9)	76.7 (0.4)	71.7 (0.6)	68.0 (0.4)
Random forest[2]	71.0 (0.8)	93.7 (0.2)	85.9 (0.6)	84.6 (0.4)
XGBoost[3]	75.1 (0.8)	95.1 (0.2)	87.9 (0.4)	87.6 (0.2)

[1]We use $k = 2$, found to be optimal on a validation set consisting of 25% of the training data.
[2]We build 500 trees with minimum node size of 10 observations.
[3]We build 3000 trees, each with tree depth of 6 splits and learning rate of 0.01.

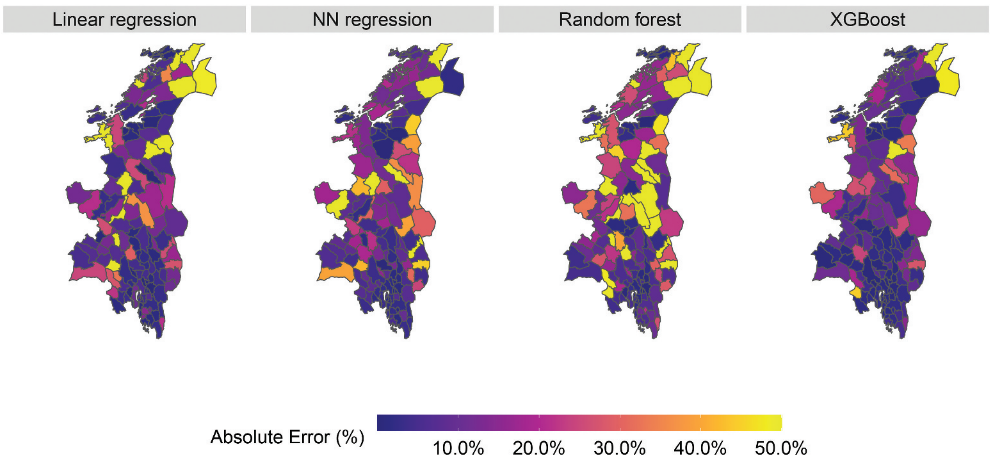

Figure 3. Absolute Error per municipality for one run of linear regression, NN regression, random forest and XGBoost. The errors are capped at 50%.

where $\Omega(h_t; \lambda, \gamma)$ is a function that penalise complicated trees, and λ, γ are regularisation parameters. This term is added to avoid over fitting and thus increase the robustness of the model.

To make optimisation easier, a second-order approximation of (1) is often used. This approximation relies on using the gradient (first derivative) $G_i = \partial\mathcal{L}(y_i, \hat{y}_i)/\partial\hat{y}_i^{(m-1)}$ and the hessian (second derivative) $H_i = \partial^2\mathcal{L}(y_i, \hat{y}_i)/\partial(\hat{y}_i^{(m-1)})^2$. Chen and Guestrin (2016) demonstrate that the optimal value on leaf j of a decision tree then can be expressed analytically as

$$w^*(j) = -\frac{\sum_{i\in\mathcal{I}_j} G_i}{\lambda + \sum_{i\in\mathcal{I}_j} H_i}, \tag{2}$$

where $\mathcal{I}_j$ is the subset of all observations in leaf j. Since gradient boosted trees are trained sequentially, the value (2) correspond to the increment that will be added to all the predictions in leaf j at the next iteration. For detailed derivation of this, see, Section 2 of Chen and Guestrin (2016). Equation (2) reveals that we can easily obtain the next increment in the prediction, as long as we know which loss function we want to use and calculate its first and second derivative. For the default choice of loss function, L_{SE}, (2) becomes

$$w^*_{SE}(j) = \frac{\sum_{i\in\mathcal{I}_j} (y_i - \hat{y}_i)}{\lambda + N_j}, \tag{3}$$

where N_j is the number of observations in leaf j. Note that if $\lambda = 0$ this becomes the average of the residuals in the given leaf. Higher values of λ will make w^*_{SE} go towards 0, which will require more iterations for the algorithm to converge. A simple, but practical example of XGBoost with SE loss and two iterations is presented in Appendix C.

Given that we are measuring the performance of the machine learning model in relative terms, we should also train the model on a relative loss function. The relative analogue to the SE loss is the Squared Percentage Error (SPE) loss,

$$L_{SPE} = \frac{1}{2}\left(\frac{y_i - \hat{y}_i}{y_i}\right)^2. \tag{4}$$

This will return the relative error (in percent) rather than the absolute error (in NOK). The first and second derivative is $G_i = -\frac{1}{y_i^2}(y_i - \hat{y}_i)$ and consequently $H_i = \frac{1}{y_i^2}$, respectively. Inserting this in (2), we obtain a leaf value of

$$w^*_{SPE}(j) = \frac{\sum_{i\in\mathcal{I}_j} \frac{1}{y_i^2}(y_i - \hat{y}_i)}{\lambda + \sum_{i\in\mathcal{I}_j} \frac{1}{y_i^2}}. \tag{5}$$

Although the expressions (3) and (5) differ slightly, they can both be written on the common form

$$w^*_{common}(j) = \frac{\sum_{i\in\mathcal{I}_j} H_i \cdot (y_i - \hat{y}_i)}{\lambda + \sum_{i\in\mathcal{I}_j} H_i}, \tag{6}$$

with $H_i = 1$ for L_{SE} and $H_i = 1/y_i^2$ for L_{SPE}. This common form highlights the differences between the leaf values of the two loss functions: While w_{SE} is a regularised mean of the residuals in each node, w_{SPE} is a regularised weighted mean of the residuals, where the weighting of each residual corresponds to $1/y_i^2$. Thus, high y_i values (i.e. expensive dwellings) will receive less weight in the training process when using the SPE loss compared with the SE loss.

4 Results and analysis

In this section, we investigate whether changing from SE loss to SPE loss will improve the performance of XGBoost. Going forward we will denote the models XGBoost-SE and XGBoost-SPE.

Table 4 shows the performance of XGBoost-SPE in comparison with XGBoost-SE and the other benchmark models. XGBoost-SPE achieves a slightly better relative RMSE, which is to be expected as the model rely on minimising the relative error. It also yields a significant improvement in the 20% measure, with a 0.6% points increase over XGBoost-SE. Somewhat surprisingly, this same improvement is not present in the 10% measure, where the two models both achieve 65.7%.

In order to understand the model performance better, we divide the dwellings into 10 price segments and evaluate the performance per segment. The segments are created by calculating the deciles of the full data set before we split it into training and test data. Table 5 shows the performance of the two models per segment. It is clear that XGBoost-SPE significantly outperforms XGBoost-SE for the lower price segments, while XGBoost-SE yields the best performance for higher price segments. In total, XGBoost-SPE yields the best performance for eight of ten price segments, with the difference between the models gradually decreasing the higher the price segment. The observed improvement among the lower price segments is in line with the theoretical results from Section 3: The

Table 4. Root Mean Squared Error (RMSE), Median Error (MdE), the 10% measure, the 20% measure and *R* for linear regression, NN regression, random forest, XGBoost-SE and XGBoost-SPE. The values indicate the mean and standard deviation across the 100 simulations with different training and test set. The best values for each performance measure is highlighted in boldface.

	RMSE (%)	MdE (%)	10% measure (%)	20% measure (%)	R^2(%)
Linear regression	32.7 (0.5)	−1.4 (0.2)	34.1 (0.2)	60.2 (0.3)	73.0 (0.43)
NN regression[1]	28.4 (0.9)	**0.0 (0.1)**	44.7 (0.2)	70.9 (0.2)	73.4 (0.47)
Random forest[2]	18.8 (0.4)	−1.2 (0.1)	63.1 (0.3)	87.1 (0.2)	90.4 (0.31)
XGBoost-SE[3]	16.2 (0.5)	0.4 (0.1)	65.7 (0.2)	89.4 (0.1)	**91.9 (0.23)**
XGBoost-SPE[3]	**15.4 (0.4)**	−1.1 (0.1)	**65.7 (0.3)**	**90.0 (0.2)**	91.4 (0.30)

[1]We use $k = 2$, found to be optimal on a validation set consisting of 25% of the training data.
[2]We build 500 trees with minimum node size of 10 observations.
[3]We build 3000 trees, each with tree depth of 6 splits and learning rate of 0.01.

data points are effectively weighted by a factor of $1/y_i^2$ in the training process, assigning more weight to dwellings in the lower price segments, thus learning better how to predict these dwellings.

Figure 4 shows a density plot of the relative errors from the two models. This indicates that the distribution of relative errors from XGBoost-SPE is slightly skewed to the left compared with XGBoost-SE. This is also highlighted by the fact that the MdE for XGBoost-SPE is -1.1% compared with 0.4% for XGBoost-SE, indicating that the new model yields more conservative estimates.

We divide the errors into three groups: Critically low errors (below -20%), acceptable errors (between -20% and 20%) and critically high errors (above 20%). Table 6 shows the number of observations that on average move between the three classes when we apply XGBoost-SPE instead of XGBoost-SE. The diagonal entries in the table indicate how many that stay in the same category, and the off-diagonal elements reflect observations that move between categories. Interestingly, there are a total of six observations that swing from the critically low category to the critically high category, or vice versa. Furthermore, 27 380 observations (86.4%) are within the acceptable error threshold for both models. Perhaps the most interesting part is to count the number of observations that have moved in or out of the acceptable threshold when changing loss function. There are a total of 1 114 observations that were either critically low or critically high with

Table 5. The 20% measure for XGBoost-SE and XGBoost-SPE. The best in each price segment is in boldface. The rightmost column shows the increase/decrease in performance in terms of percentage points.

Decile	Price segment[1]	N	XGBoost-SE (%)	XGBoost-SPE (%)	Δ(%)
1	(0,1.64]	3 160	71.6 (0.7)	**74.8 (0.7)**	+3.2
2	(1.64,1.96]	3 228	87.7 (0.5)	**89.5 (0.5)**	+1.8
3	(1.96,2.22]	3 102	91.8 (0.5)	**93.1 (0.5)**	+1.3
4	(2.22,2.45]	3 169	93.2 (0.5)	**94.1 (0.4)**	+0.9
5	(2.45,2.7]	3 332	93.8 (0.4)	**94.4 (0.4)**	+0.6
6	(2.7,2.99]	3 021	93.7 (0.4)	**94.2 (0.4)**	+0.5
7	(2.99,3.4]	3 242	92.9 (0.4)	**93.1 (0.4)**	+0.2
8	(3.4,4]	3 087	91.7 (0.4)	**91.8 (0.4)**	+0.1
9	(4,5.25]	3 207	**90.9 (0.5)**	90.4 (0.6)	−0.5
10	(5.25,100]	3 130	**87.0 (0.4)**	84.1 (0.6)	−2.9
		31,678	89.4 (0.1)	90.0 (0.2)	+0.6

[1]Measured in million NOK.

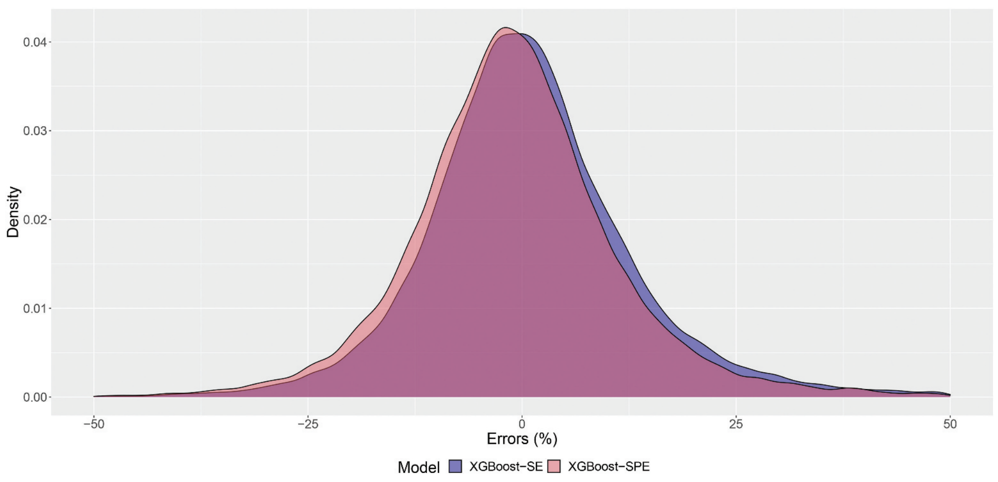

Figure 4. Density plots of the relative errors (in %) from XGBoost-SE and XGBoost-SPE, showing that the XGBoost-SPE seem to yield more conservative estimates than XGBoost-SE.

XGBoost-SE that are now within the acceptable range. There are also a total of 949 observations that have moved the other way, that is, errors that were acceptable with XGBoost-SE but critical for XGBoost-SPE. This means that in total, XGBoost-SPE yields a reduction in critical errors of 165 dwellings when we go from XGBoost-SE to XGBoost-SPE. In other words: 5.0% of the 3 349 critical errors from XGBoost-SE have been removed, which is a sizeable reduction of errors. Note also that the two models agree on a total of 29 609 observations (the trace of the table), equivalent to 93.5% of the test set.

The total value of the test set, that is, the sum of the actual observed transaction prices, is 100.20 $\pm$ 0.15 billion NOK. XGBoost-SE predicts a total value of 100.26 $\pm$ 0.16 billion NOK, whereas XGBoost-SPE predicts a total value of 97.96 $\pm$ 0.15 billion NOK, indicating a much stronger performance from XGBoost-SE. Although not surprising, as the SE loss function specifically aims to minimise the absolute error in NOK, it is nevertheless an interesting observation since banks tend to use AVMs to value a portfolio of dwellings, typically related to a portfolio of mortgages. This emphasises the importance of choosing a loss function that suits your performance measure. If the end goal is exclusively to value a portfolio of mortgages, then our experiments indicate that XGBoost-SE achieves a lower error. However, if the end goal is to minimise the number of errors that are

Table 6. A matrix showing the mean number of observations that have moved between the categories for XGBoost-SE (horizontal) and XGBoost-SPE (vertical). The column sums give the total number in each category for XGBoost-SE, while the row sums give the total number in each category for XGBoost-SPE.

	Error (SE) $\in$ $(-100\%, -20\%]$	Error (SE) $\in$ $(-20\%, 20\%)$	Error (SE) $\in$ $[20\%, \infty)$	**Sum**
Error (SPE) $\in (-100\%, -20\%]$	815	638	3	1 456
Error (SPE) $\in (-20\%, 20\%)$	249	27,380	865	28,494
Error (SPE) $\in [20\%, \infty)$	3	311	1 414	1 728
Sum	1 067	28,329	2 282	31,678

outside some kind of threshold (as done via the percentage measure), the XGBoost-SPE model does a better job. Furthermore, these goals are often not mutually exclusive and financial institutions often need to rely on both.

In terms of performance per county, we see an improvement in all counties from the XGBoost model shown in Table 3. The performance has increased from 95.1% to 95.4% in Oslo, from 75.1% to 75.9% in Innlandet, from 87.9% to 88.6% in Trøndelag and from 87.6% to 88.2% in Viken. The biggest improvement comes from Trøndelag and Innlandet, the two counties with the poorest performance, but also the counties with the lowest average prices. This gives support to the arguments presented in Subsection 3.6, namely that the SPE loss function will yield the largest improvement in prediction accuracy for dwellings in the lower price segments.

4.1 What are the differences between the two models?

What is the fundamental difference between XGBoost-SE and XGBoost-SPE? Figure 5 shows the feature importance for the 10 most important features for both models. The feature importance plots show how much performance is improved, measured by the reduction of total loss, on average each time a decision trees performs a split on the given feature in the training process. A big improvement in performance indicates that the feature is important. The feature importance values depend on the choice of loss function, and the feature importance values therefore cannot be directly compared between models with different loss functions. Nevertheless, the ranking is still useful because it highlights the importance of the different variables for the individual models.

The size of the dwelling is the most important feature for both models, followed by geographical features, such as whether or not the municipality is Oslo, as well as the Longitude and Latitude. This is not surprising, as location and size are two of the most important features for any buyer. A more surprising result, however, is that the number of homes in the extended neighbourhood is the second most important feature to XGBoost-SPE. A high value indicates a densely populated areas (typically in the cities), whereas a low value indicates fewer homes (typically rural areas).

Further down the list, we find that whether or not the dwelling is in Bærum is more important to XGBoost-SE than XGBoost-SPE. This is the municipality in the data set with the largest mean sale price (4 895 observations with an average sale price of 4.90 million NOK). The fact that this is such an important variable for XGBoost-SE underpins the results from Table 5, namely that this model is the best performer in the higher price segments. This argument is also supported by the fact that municipality being Oslo is the second most important feature for XGBoost-SE, while only being the fifth most important for XGBoost-SPE. It is also worth noting that estate type is not among the 10 most important features for either model, although its importance is probably implicitly accounted for by other variables. For instance, a small size will imply that the estate type is a flat in many cases, and variables, such as the number of homes in extended neighbourhood might also indicate whether or not a given dwelling lies in an area consisting predominantly of apartment blocks with flats or areas with primarily detached homes and so on.

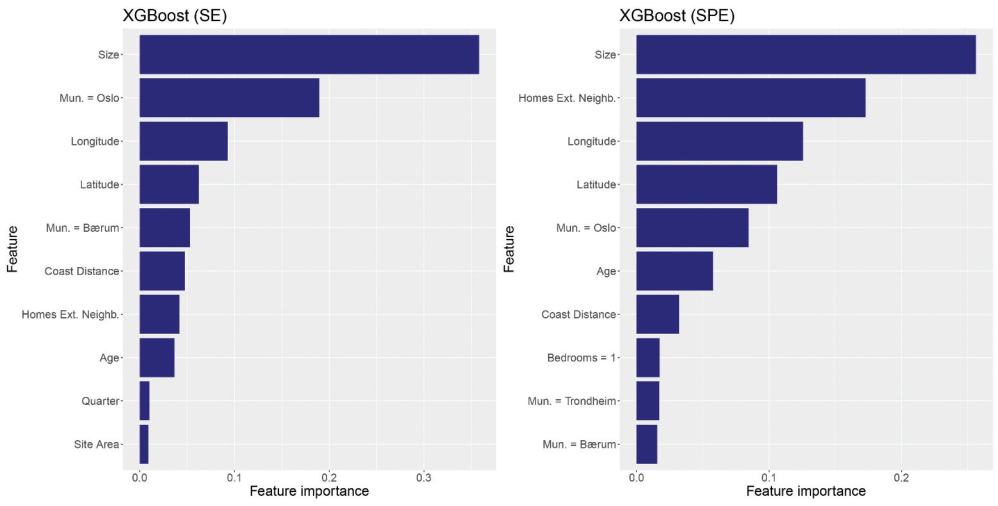

Figure 5. The ten most important features for XGBoost-SE and XGBoost-SPE. Feature importance measure the average impact of making a split on a given feature, measured by the change in accuracy after performing the split. A high value indicates that the feature is important for the model.

5 Combining XGBoost-SE and XGBoost-SPE

We noted in Section 4 that the two models perform differently in different price segments: XGBoost-SPE yielded the best results for dwellings in the lower price segments, and XGBoost-SE gave the best results for dwellings in the higher price segments. A likely explanation for this is the fact that the SPE loss function weighs each data point by a factor $1/y_i^2$ in the training process, effectively giving more weight to dwellings from lower price segments. The consequence is that XGBoost-SPE assigns more weight to dwellings from less liquid markets, that is, markets with lower transaction volume, as prices are often lower in these markets.[6] A combination of the two models might improve performance by utilising the strengths of both models. We therefore introduce the weighted average of the two predictions,

$$\text{XGBoost-Combined} = a \cdot \text{XGBoost-SE} + (1 - a) \cdot \text{XGBoost-SPE} \tag{7}$$

for some $a \in [0, 1]$. For $a = 0.5$ this would equate to a simple average of the two models, whereas $a = 0.0$ or $a = 1.0$ would give XGBoost-SPE or XGBoost-SE, respectively.

By revisiting the table of movements between error categories, Table 6, we actually find a theoretical lower bound for the combined model. Since 86.4% of the observations are within the acceptable error threshold for both models, any convex combination will also be within. Similarly, 7.0% of the observations are critical errors of the same category for both models. Hence, any weighted average of the two will also be in the same category. The maximum performance of a weighted average of XGBoost-SE and XGBoost-SPE on this data set will thus be 93.0% on the 20% measure .

To select a value for a we optimise with respect to the 20% measure on a validation set, resulting in $a = 0.39$ as the empirical optimum.[7] However, the choice of the a value will depend on how the user weigh different performance measurements against each other, and will likely also vary with the specific application and data set. Although $a = 0.39$ is the optimal value for the 20% measure, other values will likely optimise e.g. the 10% measure. Figure 6 shows the performance of the weighted average model for different values of a on the full test set. The parabola shape indicates that there indeed exists some a other than zero and one that achieves a better result on the test set, that is, there exists a convex combination of the model that outperforms the standalone models. It also confirms how the optimal a value for the 10% measure is somewhat higher than for the 20% measure . Going forward we will report the performance of XGBoost-Combined for $a = 0.39$.

Table 7 shows the performance of XGBoost-Combined. We see a further improvement in 20% measure of 0.4% over XGBoost-SPE, giving a total improvement of 1.0% from the model trained on SE loss. Notably, there is also an improvement of 1.1% points in performance when using 10% measure. This is interesting given that the two models have the same performance of 65.7% as standalone models. The combined model also yields a marginally better RMSE than XGBoost-SPE and a clear improvement in MdE.

We previously noted that XGBoost-SPE had a net reduction of 165 dwellings that were outside of the $\pm 20\%$ error band. The combined model yields a reduction of 311 dwellings, corresponding to a 9.3% reduction in critical errors from the 3 349 dwellings that XGBoost-SE model got wrong. A detailed table showing the number of observations in each error category for the combined model can be found in Appendix B.

Table 8 shows the results per price segment for XGBoost-Combined compared with the two submodels individually. For seven of the price segments XGBoost-Combined performs as good as or better than XGBoost-SPE, while XGBoost-SPE still is the best performer for the first and second price segments. Similarly, XGBoost-SE is still the leading performer for the highest price segment. It worth noticing, however, that even for

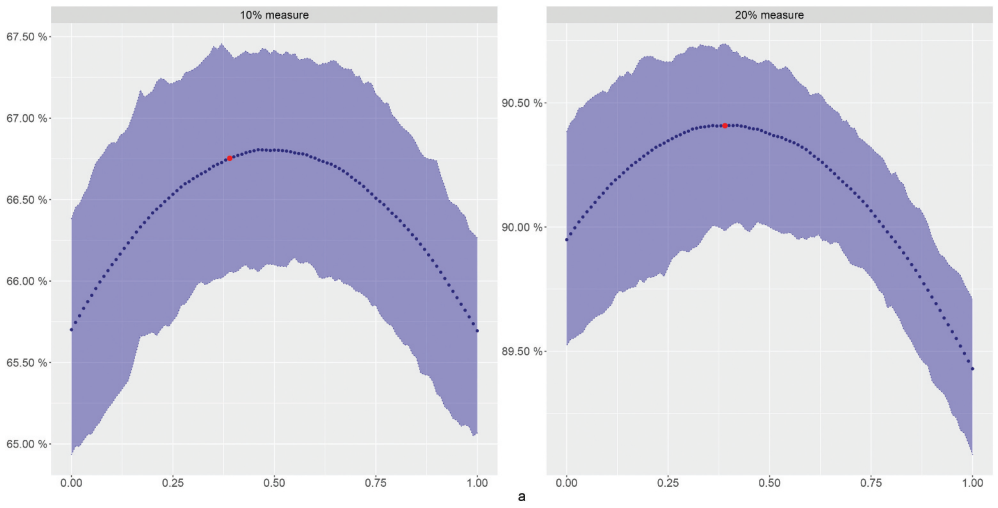

Figure 6. Empirical results for XGBoost-combined on the full test set with different values of a. For each value of a we report the mean (dots) in addition to the minimum and maximum value (the shaded area) for the 100 simulations. $a = 0$ and $a = 1$ equates the pure XGBoost-SPE and XGBoost-SE model, respectively. The red dot shows $a = 0.39$, found to be the empirical optimum of the 20% measure on a validation set.

Table 7. Root Mean Squared Error (RMSE), Median Error (MdE), the 10% measure, the 20% measure and R^2 for XGBoost-SE, XGBoost-SPE and XGBoost-Combined. The values indicate the mean and standard deviation across the 100 simulations with different training and test set.

	RMSE (%)	MdE (%)	10% measure (%)	20% measure (%)	R^2(%)
XGBoost[1]	16.2 (0.5)	**0.4 (0.1)**	65.7 (0.2)	89.4 (0.1)	91.9 (0.23)
XGBoost-SPE[1]	15.4 (0.4)	−1.1 (0.1)	65.7 (0.3)	90.0 (0.2)	91.4 (0.30)
XGBoost-Combined[1]	**15.3 (0.4)**	−0.5 (0.1)	**66.8 (0.2)**	**90.4 (0.1)**	**92.1 (0.26)**

[1]We build 3000 trees, each with tree depth of 6 splits and learning rate of 0.01.

Table 8. The 20% criterion for XGBoost-SE and XGBoost-SPE. The best in each price segment is in boldface. The rightmost column shows the increase/decrease in performance in terms of percentage points.

Decile	Price segment[1]	N	XGBoost-SE (%)	XGBoost-SPE (%)	XGBoost-Combined (%)
1	(0,1.64]	3 160	71.6 (0.7)	**74.8 (0.7)**	74.5 (0.7)
2	(1.64,1.96]	3 228	87.7 (0.5)	**89.5 (0.5)**	**89.5 (0.5)**
3	(1.96,2.22]	3 102	91.8 (0.5)	**93.1 (0.5)**	93.0 (0.5)
4	(2.22,2.45]	3 169	93.2 (0.5)	94.1 (0.4)	**94.2 (0.4)**
5	(2.45,2.7]	3 332	93.8 (0.4)	94.4 (0.4)	**94.6 (0.3)**
6	(2.7,2.99]	3 021	93.7 (0.4)	94.2 (0.4)	**94.5 (0.3)**
7	(2.99,3.4]	3 242	92.9 (0.4)	93.1 (0.4)	**93.5 (0.4)**
8	(3.4,4]	3 087	91.7 (0.4)	91.8 (0.4)	**92.4 (0.4)**
9	(4,5.25]	3 207	90.9 (0.5)	90.4 (0.6)	**91.3 (0.5)**
10	(5.25,100]	3 130	**87.0 (0.4)**	84.1 (0.6)	86.5 (0.5)
		31,678	89.4 (0.1)	90.0 (0.2)	**90.4 (0.1)**

[1]Measured in million NOK.

the segments where the combined model is not the best, it is still very close to the best model. The performance of the combined model is, for every price segment, better than a linear combination of the performances achieved by the two submodels.

Finally, note that both the SE and SPE loss functions are special cases of a more general family of loss functions, $L_\alpha(y_i, \hat{y}_i) = y^{-\alpha} \cdot (y_i - \hat{y}_i)^2$, with $\alpha = 0$ and $\alpha = 2$, respectively. Other values of α might yield promising results depending on the performance measure of interest. To investigate this, we performed a brief simulation study where we evaluated the the effect of α for a range of values with respect to the 20% measure. The results of this study can be found in Appendix D. While the results indicate that we are able to find an α that yields a slight improvement compared with the considered loss functions, the accuracy was still lower than that of the combined model.

6 Conclusion

Machine learning approaches for house price prediction tend to outperform more classical approaches, such as hedonic regression and repeated sales models. Moreover, as Automated Valuation Models (AVMs) are a requirement for credit institutions, machine learning-based AVMs are likely to dominate in near future.

In this paper, we demonstrate an improvement in XGBoost when it is trained on a tailored loss function that more closely mirrors the performance measure used, the percentage criterion, rather than the standard loss function that relies on squared errors.

The introduction of the SPE loss yields an improvement in performance from $89.4 \pm 0.1\%$ to $90.0 \pm 0.2\%$ when using the 20% measure . While XGBoost-SE gives 3 349 predictions that do not satisfy the 20% measure, this number is reduced by 4.9% with XGBoost-SPE. The biggest improvement comes from the lower price segments, while the two highest price segments experience a decline in performance when changing loss function from SE loss to SPE loss.

A somewhat surprising finding is that the two models may be combined to make an even better hybrid model that utilises both the strength of XGBoost-SPE on the lower price segments and the strength of XGBoost-SE on the higher price segments. This weighted average model gives $90.4 \pm 0.1\%$ by the 20% measure, while also reducing the number of observations outside the 20% measure to 3 038, corresponding to a reduction of 9.3% from XGBoost-SE.

The first implication of these findings relates to the awareness of the choice of loss function. Picking a suitable loss function for the application and performance measure at hand can have the potential to improve model performance for house price prediction models, as demonstrated in this paper. Practitioners and academics ought to explore different loss functions as part of their modelling work as opposed to relying only on the default choices offered by the pre-implemented software packages. Specifically, one might consider training multiple models with different loss functions or applying different loss functions to different subsets of the data set. Policy makers and industry stakeholders should also be also be aware of the interplay between the choice of performance measure and choice of loss function: If a specific and potentially unusual performance measure is being applied, one should be aware of how the choice of loss function might potentially affect performance.

The second implication relates to the broader picture. The housing market is an essential part of most modern economies, especially in countries with high ownership rate and a majority of the homeowners relying on mortgages to finance the acquisition. Banks need AVMs in order to provide mortgages, as dwellings are used as mortgage collateral. It is therefore important to get reliable estimates of the value of a dwelling, also in the absence of a recent transaction. Furthermore, house price prediction models also help market efficiency by providing both buyer and seller with information about a dwellings common value. Improving the models used for these purposes will lead to more informed decisions from both financial institutions and potential buyers and sellers.

We conclude that the introduction of a more application specific and tailored loss function can have a significant impact on the performance of AVMs. In terms of future research, introducing more sophisticated and data driven weightings in the combined model is an interesting next step. Specifically, assigning individual weights for each dwelling rather than a constant weight, is an enticing option that would likely result in a further improvement in accuracy. Also, although we found no conclusive answers in the simulation study with different values of α for the family of loss functions, this is an interesting topic for further research. By considering more flexible families of loss functions, one might be able to discover even more appropriate loss functions, which have been fine-tuned for the application and data set at hand.

Notes

1. We choose to use the term 'performance measure' rather than 'performance metric' to avoid confusion with the strict mathematical definition of a 'metric', although we acknowledge that the latter term is often used in the literature.
2. OECD Affordable Housing Database. Numbers from 2019, accessed 12th of January 2021.
3. Research by Eiendomsverdi AS, a Norwegian property technology company.
4. There is a total of eleven counties in Norway.
5. 1 Norwegian krone $\approx$ 0.12 US Dollars as of 22th of October 2021.
6. This can also be found in our data set, where the mean price in the cities is 3.55 million NOK, compared with 2.80 million NOK outside the cities.
7. This is done by conducting a further split: We use 75% of the training set for model training and the other 25% of the training set for validation of different values of a. The test set (25% of the full data) is never used for this purpose.

Acknowledgement

We would like to extend our gratitude to Eiendomsverdi AS for providing the data set and for their valuable insights on the Norwegian housing market in general. We are grateful towards the four anonymous reviewers for providing helpful remarks that improved the article. Finally, we also express our gratitude towards Dr. Bryan S. Weber for his valuable comments on an early version of this work that was presented at the WEAI 96th Annual Conference in July 2021.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

AH was partly funded by the Norwegian Research Council through the industrial PhD scheme under grant number 322779.

Notes on contributors

Anders Hjort is a PhD research fellow in the Department of Mathematics at the University of Oslo. His research is focused on the use of machine learning models for house price prediction. He is writing his PhD in cooperation with Eiendomsverdi AS.

Johan Pensar is an Associate Professor in Statistics at the Department of Mathematics at the University of Oslo. His research includes probabilistic graphical models and the development of statistical models and machine learning methods for real world applications. He completed his PhD in statistics at Åbo Akademi University (Finland) in 2016.

Ida Scheel is an Associate Professor in Statistics at the Department of Mathematics at the University of Oslo. Her research includes data science in a statistical perspective, Bayesian statistics and recommender systems. She completed her PhD in statistics at the University of Oslo in 2008.

Dag Einar Sommervoll is a Professor at the School of Economics and Business at the Norwegian University of Life Sciences (NMBU) and a Professor at the NTNU Business School. His research includes housing economics and house price prediction. He completed his PhD in Mathematics at the University of Oslo in 1997.

References

Arya, S., David, M., Kemp, S. E., & Jefferies, G. (2019). *RANN: Fast Nearest Neighbour Search (Wraps ANN Library) Using L2 Metric*. The Comprehensive R Archive Network (CRAN). R package version 2.6.1. url. https://CRAN.R-project.org/package=RANN

Arya, S., Mount, D. M., Netanyahu, N. S., Ruth, S., & Wu, A. Y. (1998). An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the A CM* , *45*(6), 891–923. https://doi.org/10.1145/293347.293348

Bailey, M. J., Muth, R. F., & Nourse, H. O. (1963). A regression method for real estate price index construction. *Journal of the American Statistical Association*, *58*(304), 933–942. https://doi.org/10.1080/01621459.1963.10480679

Baldominos, A., Blanco, I., Moreno, A., Iturrarte, R., Bernárdez, Ó., & Afonso, C. (2018). Identifying real estate opportunities using machine learning. *Applied Sciences*, *8*(11), 2321. https://doi.org/10.3390/app8112321

Barron, J. T. (2019). “A general and adaptive robust loss function”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* Long Beach, California, pp. 4326–4334.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–23. https://doi.org/10.1023/A:1010933404324

Cain, M., & Janssen, C. (1995). Real estate price prediction under asymmetric loss. *Annals of the Institute of Statistical Mathematics*, *47*(3), 401–414. https://doi.org/10.1007/BF00773391

Case, K. E., Shiller, R. J., & Weiss, A. N. (1993). Index- based futures and options markets in real estate. *The Journal of Portfolio Management*, *19*(2), 83–92. https://doi.org/10.3905/jpm.1993.409441

Chen, T., & Guestrin, C. (2016). “XGBoost: A scalable tree boosting system”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* San Francisco, California (New York (NY), United States: Association for Computing Machinery).

Clapp, J., Case, B., Dubin, R., & Rodríguez, M. (2004). Modeling spatial and temporal house price patterns: A comparison of four models. *The Journal of Real Estate Finance and Economics*, *29*(2), 167–191. https://doi.org/10.1023/B:REAL.0000035309.60607.53

Demiroglu, C., & James, C. (2018). Indicators of collateral misreporting. *Management Science*, *64*(4), 1747–1760. https://doi.org/10.1287/mnsc.2016.2597

Englund, P., Quigley, J. M., & Redfearn, C. L. (1998). Improved price indexes for real estate: Measuring the course of Swedish housing prices. *Journal of Urban Economics*, *44*(2), 171–196. https://doi.org/10.1006/juec.1997.2062

FitchRatings (2019). *Conservatism Still the Best Path for AVMs in U.S. RMBS*. Accessed 2022 January 12.

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

Friedman, J. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, *29*(5), 1189–1232. https://doi.org/10.1214/aos/1013203451

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Special invited paper. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, *28*(2), 337–374. https://doi.org/10.1214/aos/1016218223

Gabriel, P., George, T., Jan, C., Lukasz, K., & Hinton, G. E. (2017). “Regularizing neural networks by penalizing confident output distributions”. In: *5th International Conference on Learning Representations, ICLR 2017, Workshop Track Proceedings* April 24-26, 2017 Toulon, France.

Giudice, V., Paola, P., & Cantisani, G. (2017). Valuation of real estate investments through fuzzy logic. *Buildings*, *7*(1), 23 https://doi.org/10.3390/buildings7010026.

Goodman, A. C., & Thibodeau, T. G. (2003). Housing market segmentation and hedonic prediction accuracy. *Journal of Housing E C Onomics*, *12*(3), 181–201. https://doi.org/10.1016/S1051-1377(03)00031-7

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning. In *Springer series in statistics* 2nd. Springer New York Inc, 587-603. https://doi.org/10.1007/978-0-387-84858-7

Ho, W. K. O., Tang, B.-S., & Wong, S. W. (2020). Predicting property prices with machine learning algorithms. *Journal of Property Research* https://doi.org/10.1080/09599916.2020.1832558

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Huber, P. J. (1964). Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, *35*(1), 73–101. https://doi.org/10.1214/aoms/1177703732

Ke, Guolin, Meng, Qi, Finley, Thomas W., Wang, Taifeng, Chen, Weiwei Yu, Ma, Weidong, Ye, Qiwei, Liu, Tie-Yan, et al. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, *30*(1), 3146–3154. https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf} .

Kim, J., Won, J., Kim, H., & Heo, J. (2021). Machine-learning-based prediction of land prices in Seoul, South Korea. *Sustainability*, *13*(23), 202–211. https://doi.org/10.3390/su132313088

Kruger, S., & Maturana, G. (2020). Collateral misreporting in the residential mortgage-backed security market. *Management Science*, *67*(5), 2657–3320. https://doi.org/10.1287/mnsc.2019.3569.

LeCun, Y., Haffner, P., Bottou, L., & Bengio, Y. (1999). Object recognition with gradient- based learning. *Shape, Contour and Grouping in Computer Vision*, 319–345. https://doi.org/10.1007/3-540-46805-6_19

Lim, W., Wang, L., Wang, Y. L., & Chang, Q. (2016). Housing price prediction using neural networks *12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery* August 13-15, 2016 Changsa, China, 518–522 https://doi.org/10.1016/j.iswa.2021.200052 .

Louppe, G. (2014). "Understanding Random Forests: From Theory to Practice". PhD thesis. University of Liège.

McCluskey, W. J., McCord, M., Davis, P. T., Haran, M., & McIlhatton, D. (2013). Prediction accuracy in mass appraisal: A comparison of modern approaches. *Journal of Property Research*, *30*(4), 239–265. https://doi.org/10.1080/09599916.2013.781204

Miriam, S., Hill, R. J., & Pfeifer, N. (2021). Metrics for evaluating the performance of machine learning based automated valuation models. *Journal of Property Research*, *38*(2), 99–129. https://doi.org/10.1080/09599916.2020.1858937

Nitschke, S., Biguzzi, A., Bücker, A., Magnolfi, S., Selleri, F., & Vetrano, P. (2019). *European standards for statistical valuation methods for residential properties*. Tech. rep. European AVM Alliance.

Páez, A., Gallo, L., Julie, B., Ron, & Dall'Erba, S. (2010). Progress in Spatial Analysis: Introduction Antonio Páez and Julie Gallo and Ron N. Buliung and Sandy Dall'erba. Progress in Spatial Analysis Advances in Spatial Science (ADVSPATIAL) (Springer). 1–13. ISBN: 978-3-642-03324-7. https://doi.org/10.1007/978-3-642-03326-1.

Park, B., & Bae, J. K. (2015). Using machine learning algorithms for housing price prediction: The case of Fairfax County, Virginia housing data. *Expert Systems with Applications* 6, 42. https://doi.org/10.1016/j.eswa.2014.11.040 .

Poursaeed, O., Matera, T., & Belongie, S. (2018). Vision-based real estate price estimation. *Machine Vision and Applications*, *29*(4), 667–676. https://doi.org/10.1007/s00138-018-0922-2

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: Unbiased boosting with categorical features. *Advances in Neural Information Processing Systems*, *31*, 6639–6649. https://doi.org/10.5555/3327757.3327770

Quigley, J. M. (1995). A simple hybrid model for estimating real estate price indexes. *Journal of Housing Economics*, *4*(1), 1–12. https://doi.org/10.1006/jhec.1995.1001

Rosen, S. (1974). Hedonic prices and implicit markets: Product differentiation in pure competition. *Journal of Political Economy*, *82*(1), 34–55. https://doi.org/10.1086/260169

Sing, T. F., Yang, J. J., & Shi Ming, Y. (2021). Boosted tree ensembles for artificial intelligence based automated valuation models (AI-AVM). *The Journal of Real Estate Finance and Economics*, *63*(4), https://doi.org/10.1007/s11146-021-09861-1

Sommervoll, Å., & Sommervoll, D. (2019). Learning from man or machine: Spatial fixed effects in urban econometrics. *Regional Science and Urban Economics*, *77*, 239–252. https://doi.org/10.1016/j.regsciurbeco.2019.04.005

Terregrossa, S., & Ibadi, M. (2021). Combining housing price forecasts generated separately by hedonic and artificial neural network models. *Asian Journal of Economics, Business and Accounting* 21(1) , 130–148. https://doi.org/10.9734/ajeba/2021/v21i130345

Varian, H. R. (1975). A Bayesian approach to real estate assessment. In S. E. Feinberge and A. Zellner, Eds. *Studies in Bayesian Econometrics and Statistics*, 195–208.

Viriato, J. C. (2019). AI and machine learning in real estate investment. *The Journal of Portfolio Management*, *45*(7), 43–54. https://doi.org/10.3905/jpm.2019.45.7.043

Wang, X., Wen, J., Zhang, Y., & Wang, Y. (2014). Real estate price forecasting based on SVM optimized by PSO. *Optik*, *125*(3), 1439–1443. https://doi.org/10.1016/j.ijleo.2013.09.017

Xiaojie, X., & Zhang, Y. (2021). House price forecasting with neural networks. *Intelligent Systems with Applications*, *12*, 200052. https://doi.org/10.1016/j.iswa.2021.200052

Yencha, C. (2019). Valuing walkability: New evidence from computer vision methods. *Transportation Research Part A: Policy and Practice*, *130*(12), 689–709. https://doi.org/10.1016/j.tra.2019.09.053

Zillow (2019). *Introducing a new and improved Zestimate algorithm*. Retrieved January 17, 2022.

Appendices

A. Data preparation procedure

In Table 9 we give a step-by-step overview of the data preparation process.

Table 9. The steps in the data preparation procedure.

Data Operation	Rows
Raw data: Arm's Length Sales in Oslo, Viken, Innlandet, Trøndelag in 2013–2015[1]	136 487
Observations with living area	134 889
Observations with site area and footprint area[2]	134 664
Observations with coordinates and valid build year	134 018
Observations with number of bedrooms	128 619
Observations with floor number[3]	126 938
Observations with distance to closest coast line and lake	126 812
Observations with less than 15 floors	126 782
Observations with living area less than 800 square metres	126 781
Observations where sale price is in the range of 0.5–1.5x of asking price	126 719

[1]There are four estate types: flat, duplex, row house, detached home
[2]Applicable to detached homes only. Other estate types are assumed to have zero site area and footprint area.
[3]Applicable to flats only. Other estate types are assumed to start at ground floor.

B. Movements between error categories for XGBoost-Combined

Table 10 shows the number of observations that have moved between the three error categories for XGBoost-SE and XGBoost-SPE. Table 11 shows the mean number of observations that have moved between the error categories.

Table 10. A matrix showing the mean number of observations that have moved between the categories for XGBoost-SE (horizontal) and XGBoost-Combined (vertical). The column sums give the total number in each category for XGBoost-SE, while the row sums give the total number in each category for XGBoost-SPE.

	Error (SE) $\in (-100\%, -20\%]$	Error (SE) $\in (-20\%, 20\%)$	Error (SE) $\in [20\%, \infty)$	**Sum**
Error (Comb.) $\in (-100\%, -20\%]$	887	302	0	1 189
Error (Comb.) $\in (-20\%, 20\%)$	179	27,860	600	28,639
Error (Comb.) $\in [20\%, \infty)$	1	167	1 682	1 850
Sum	1 067	28,330	2 282	31,678

Table 11. A matrix showing the mean number of observations that have moved between the categories for XGBoost-SPE (horizontal) and XGBoost-Combined (vertical). The column sums give the total number in each category for XGBoost-SE, while the row sums give the total number in each category for XGBoost-SPE.

	Error (SE) $\in (-100\%, -20\%]$	Error (SE) $\in (-20\%, 20\%)$	Error (SE) $\in [20\%, \infty)$	**Sum**
Error (Comb.) $\in (-100\%, -20\%]$	1117	72	0	1 189
Error (Comb.) $\in (-20\%, 20\%)$	339	28,154	146	28,639
Error (Comb.) $\in [20\%, \infty)$	0	268	1 582	1 850
Sum	1 456	28,495	1 728	31,678

C. Worked example of gradient boosted trees

Here, we present a quick-worked example of the first two iterations of the XGBoost algorithm on a simple data set of a total of 100 observations from a specific city district in Oslo. For simplicity, we only include two covariates in the prediction model: Size, that is, the number of square metres, and Floor, that is, which floor the dwelling is on. We use a shrinkage of $\eta = 1$ and regularisation parameter $\lambda = 0$. We train the model on the standard Squared Error (SE) loss function. The first and second decision tree can be seen in Figure 7. Each tree makes two splits, splitting the data set into a total of four subsets each time. Every subset has a Value (which has previously been denoted $w(j)$) as well as a Cover number, which indicates how many of the 100 observations that are in this leaf. The splits are also equipped with a Gain parameter, which indicates how much the precision in the trees increase as a consequence of this split. The numbers on the arrows (such as "< 62") is the decision rule: Any observation with a Size feature less than 62 m^2 will follow that arrow.

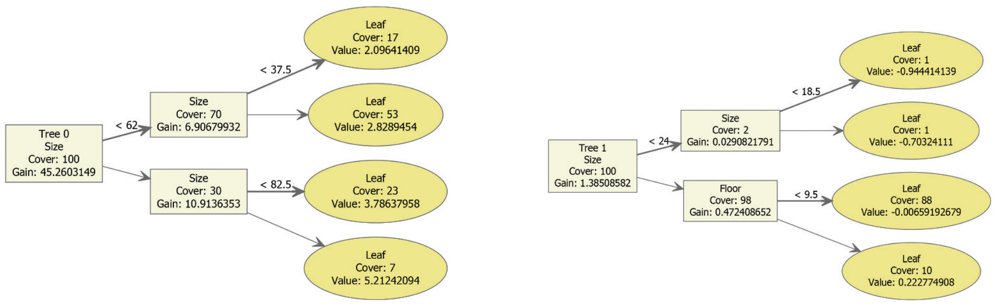

Figure 7. The first and second decision tree (left and right, respectively). Cover indicates how many observations that are in a given leaf, while Gain indicates the increase in precision as a consequence of the split. Value shows the actual prediction from the tree.

Consider now a new dwelling with unknown y_i but with known features: Size $= 70\ m^2$ and Floor $= 10$. In the first tree, this instance will get a prediction $h(x_i) \approx 3.79$ mill NOK, since it belongs in the third leaf from above. In the second tree, the same dwelling belongs to the fourth leaf node (with Value ≈ 0.22), since it is bigger than 24 m^2 and is higher than floor 9.5. After two iterations, the final prediction will thus be

$$f_2(x_i) = \sum_{m=1}^{2} \eta \cdot h_m(x_i; q_m)$$

$$\approx 3.79 + 0.22 = 4.01.$$

The algorithm will then calculate the error from $f_2(x_i)$ (i.e. the difference between 4.01 million NOK and the true price y_i) and use this to train the third tree. A normal procedure involves hundreds or thousands of sequential trees and typically a shrinkage parameter that is lower (e.g. $\eta = 0.01$), but this example is deliberately made as simple as possible.

D. A family of loss functions

Both the SE loss and the SPE loss belong to a broader class,

$$L_\alpha(y_i, \hat{y}_i) = y^{-\alpha} \cdot (y_i - \hat{y}_i)^2$$

where $\alpha = 0$ for SE and $\alpha = 2$ for SPE. Rather than a weighted average of two models trained on $L_0(y_i, \hat{y}_i)$ and $L_2(y_i, \hat{y}_i)$, we might imagine that there exist some α, possibly between 0 and 2 that achieves the same feat. In Figures 8, 9 we see the results of simulations with different values of α. There seems to be values of α in the region around $\alpha = 1.5$ in Figure 8 that yields better performance than $\alpha = 2.0$. Figure 9 also show the 20% measure per price segment. For many of the lower price segments the optimal α seems to lie in the region between 1.5 and 2.0, with performance declining for $\alpha > 2.0$.

While the results are promising, more research is needed to conclude about the efficiency of using the hybrid loss function.

There is no single value of α that achieves a better performance than the weighted average model introduced in the text. While this family of loss functions seems interesting due to its flexibility, more investigation, tuning and testing is required to find out if it is useful in this context.

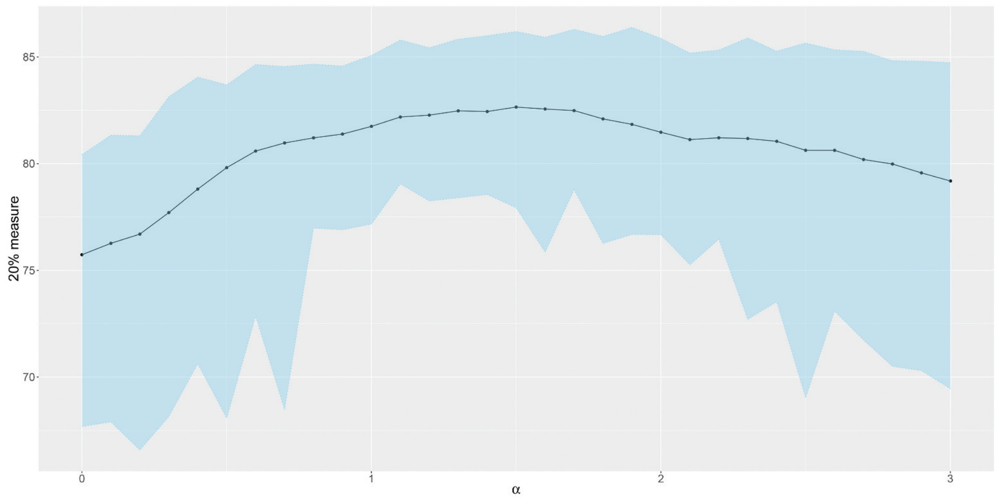

Figure 8. Training XGBoost models with loss function $L_\alpha(y_i, \hat{y}_i) = y^{-\alpha} \cdot (y_i - \hat{y}_i)^2$ for $\alpha = 0.0, 0.1, \ldots, 3.0$. For every value of α we train 100 models and report the mean performance (dotted line) and min/max performance. The performance measure is the 20% measure . The full data consists of 40 000 rows, and is a randomly sampled subset of the full data set used in the article.

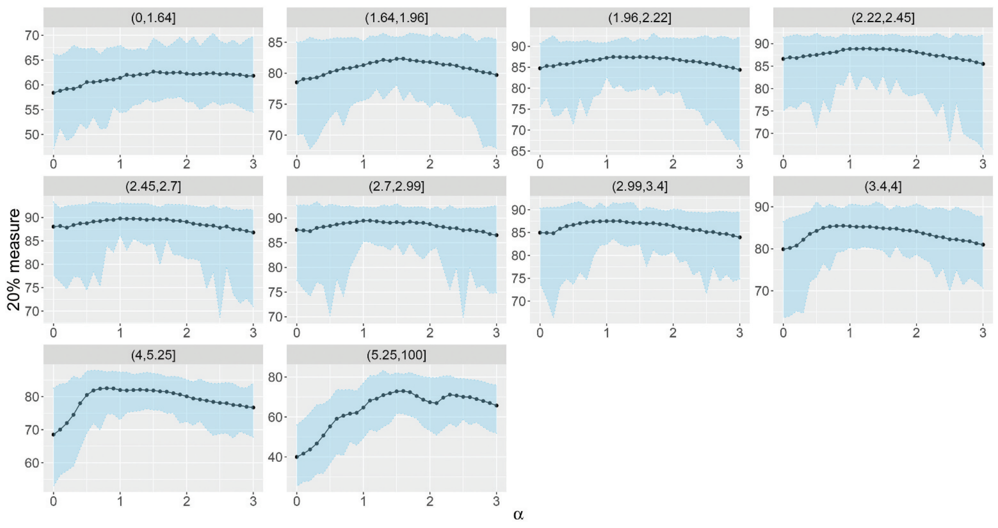

Figure 9. Training XGBoost models with loss function $L_a(y_i, \hat{y}_i) = y^{-a} \cdot (y_i - \hat{y}_i)^2$ for $a = 0.0, 0.1, \ldots, 3.0$. For every value of a we train 100 models and report the mean performance (dotted line) and min/max performance. The performance measure is the 20% measure, and each plot shows a different price segment. The full data consists of 40000 rows, and is a randomly sampled subset of the full data set used in the article.