UiO : Department of Mathematics
University of Oslo

House Price Prediction: Classical Methods vs. Machine Learning Methods

WEAI 96th Annual Conference

Anders Hjort

July 1, 2021

Introduction

Background:

- Automated Valuation Models (AVMs) are useful tools for banks, insurance companies and home owners.
- Hedonic regression models have been used in the past, but machine learning models are becoming increasingly popular.

Introduction

Background:

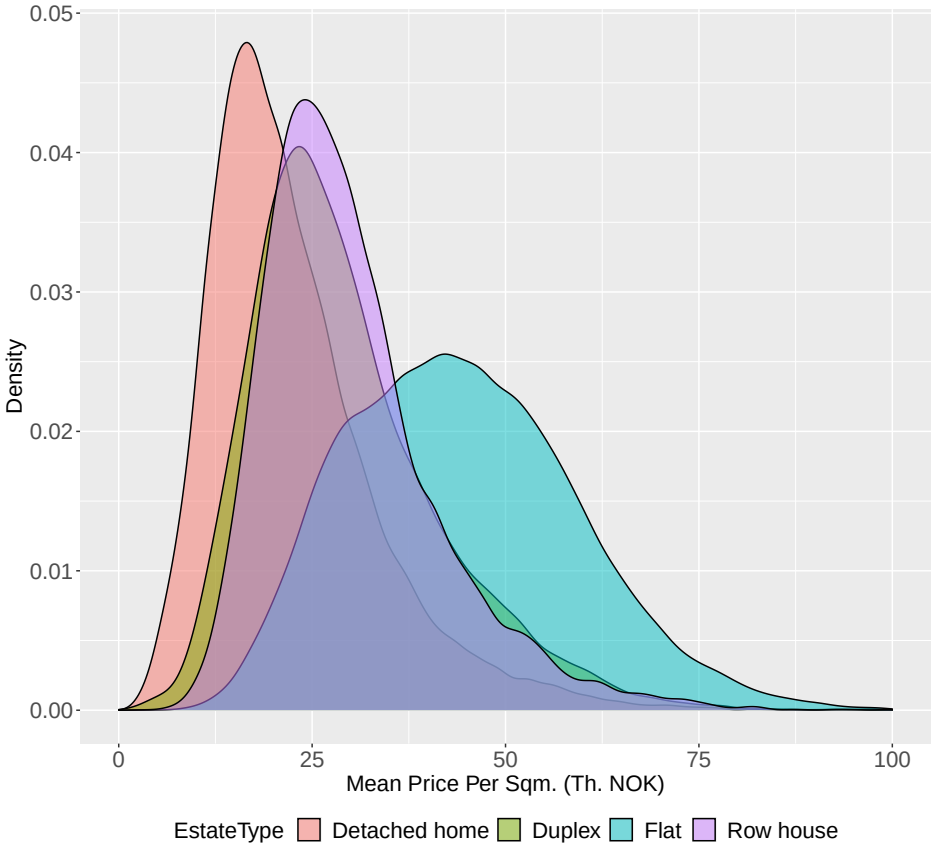
- Automated Valuation Models (AVMs) are useful tools for banks, insurance companies and home owners.
- Hedonic regression models have been used in the past, but machine learning models are becoming increasingly popular.

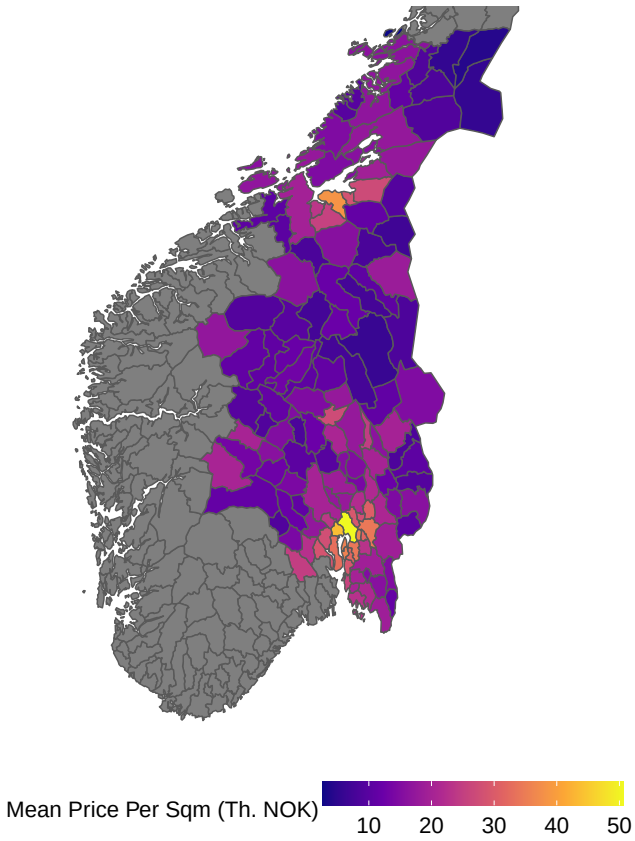
How to evaluate an AVM?

- The **20% criterion**: What fraction of predictions are within $\pm 20\%$ of the true value?
- Norwegian banks are obliged to use the 20% criterion for regulatory purposes.

The Norwegian housing market

We are studying $N = 126\,719$ transactions from 2013-2015. We limit ourselves to four counties (of 11) in the South-Eastern part of Norway. All prices in NOK [1]





[1] 1 NOK $\approx$ 0.12 USD as of July 1st 2021.

Prediction challenge

We want to predict the price of an dwelling (y_i) based on the available information about the dwelling ($\boldsymbol{x}_i$):

- Location: Coordinates, Municipality, County etc.
- Dwelling properties: Square meter, floors, balcony, garage etc.
- Neighborhood properties: Number of houses nearby, distance to ocean etc.

Prediction challenge

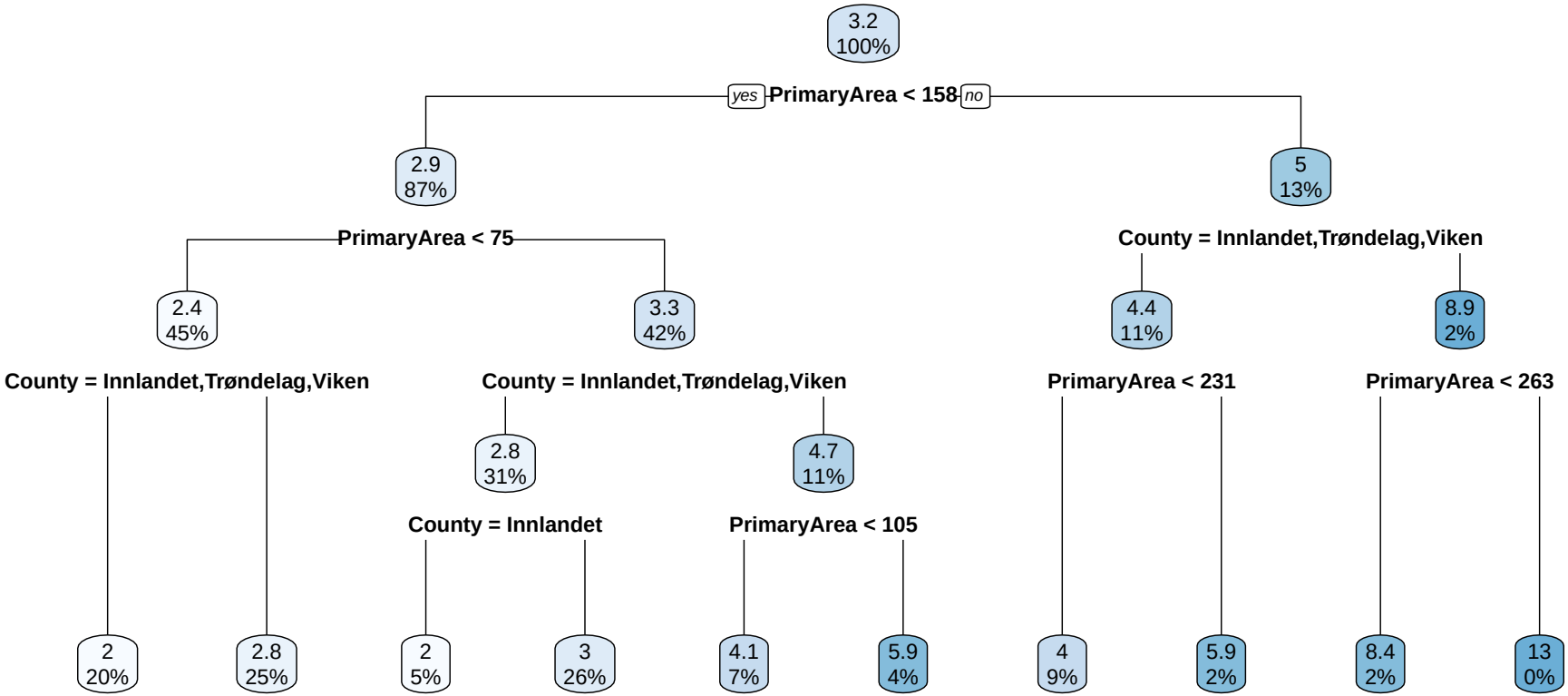
We want to predict the price of an dwelling (y_i) based on the available information about the dwelling ($\boldsymbol{x}_i$):

- Location: Coordinates, Municipality, County etc.
- Dwelling properties: Square meter, floors, balcony, garage etc.
- Neighborhood properties: Number of houses nearby, distance to ocean etc.

We consider the following models:

- Linear regression.
- k Nearest Neighbour (kNN) regression.
- Random forest.
- Boosted trees, specifically the XGBoost implementation by [1].

A Simple Decision Tree



A simple decision tree to predict price of a dwelling based on County and Primary Area. The three has four splits, resulting in $T = 10$ leaves. The score at leaf j, w_j, is the predicted price (in million NOK) for a dwelling in this bracket.

Tree based methods in a nutshell

- **Random forest:** Build multiple decision trees in *parallel*, with bootstrapped data.
  Final prediction: Average of all trees.
- **Boosted trees:** Build multiple decision trees *sequentially*, where each tree is trained on the residuals from the previous tree.
  Final prediction: Sum of all trees.

Tree based methods are the best performers

Experimental set up: Split the data into train set (75%) and test set (25%).[2]

	Mean Sq. Error	Median Error	10% criterion	20% criterion
Linear regression	10.7%	-1.7%	34.4%	60.3%
k Nearest Neighbour	14.1%	-3.5%	32.5%	57.7%
Random forest	3.7%	-1.3%	62.8%	87.0%
XGBoost	2.6%	0.4%	65.6%	89.5%

[2] $N_{train} = 95\,041$, $N_{test} = 31\,678$

Main research challenge

Challenge:

AVMs are trained to optimize absolute error, but evaluated using relative error.
Can we improve the performance of XGBoost by introducing a better suited loss function?

Main research challenge

Challenge:

AVMs are trained to optimize absolute error, but evaluated using relative error.
Can we improve the performance of XGBoost by introducing a better suited loss function?

Example:

- House A is sold for NOK 10 million, the AVM predicts NOK 11 million.
  $\Longrightarrow$ Absolute error of NOK 1 mill. Relative error of 10%.
- House B is sold for NOK 2 million, the AVM predicts NOK 2.2 million.
  $\Longrightarrow$ Absolute error of NOK 0.2 mill. Relative error of 10%.

How is the next decision tree constructed?

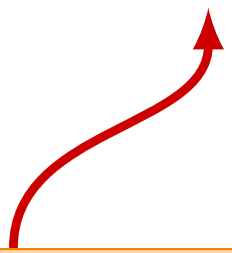
Find the predictions $\hat{y}_1, \hat{y}_2, ..., \hat{y}_N$ that minimizes the following expression:

$$\mathcal{L} = \sum_{i=1}^{N} L(y_i, \hat{y}_i) + \Omega(\text{tree structure})$$

How is the next decision tree constructed?

Find the predictions $\hat{y}_1, \hat{y}_2, ..., \hat{y}_N$ that minimizes the following expression:

$$\mathcal{L} = \sum_{i=1}^{N} L(y_i, \hat{y}_i) + \Omega(\text{tree structure})$$



Loss function: Penalize poor predictions $\hat{y}_i$

How is the next decision tree constructed?

Find the predictions $\hat{y}_1, \hat{y}_2, ..., \hat{y}_N$ that minimizes the following expression:

$$\mathcal{L} = \sum_{i=1}^{N} L(y_i, \hat{y}_i) + \Omega(\text{tree structure})$$

Loss function: Penalize poor predictions $\hat{y}_i$

Regularization: Penalize trees that are too complicated

What loss function should we use?

Most prediction tasks use the Squared Error (SE) loss,

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

or the Absolute Error (AE) loss,

$$L_{AE} = |y_i - \hat{y}_i|.$$

What loss function should we use?

Most prediction tasks use the Squared Error (SE) loss,

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

or the Absolute Error (AE) loss,

$$L_{AE} = |y_i - \hat{y}_i|.$$

We propose to use the Squared Percentage Error (SPE) loss,

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

XGBoost (SPE) outperforms XGBoost (SE) slightly

	Mean Sq. Error	Median Error	10% criterion	20% criterion
Linear regression	10.7%	-1.7%	34.4%	60.3%
k Nearest Neighbour	14.1%	-3.5%	32.5%	57.7%
Random forest	3.7%	-1.3%	62.8%	87.0%
XGBoost (SE)	2.6%	0.4%	65.6%	89.5%
XGBoost (SPE)	**2.3%**	**-1.0%**	**65.8%**	**90.0%**

Summary of models, but expanded with the new XGBoost (SPE).

XGBoost (SPE) notably better for cheap houses

Price Class	N	XGBoost (SE)	XGBoost (SPE)
(0,2]	6903	80.4%	**82.7%**
(2,4]	18431	93.0%	**93.6%**
(4,6]	4284	**90.7%**	90.0%
(6,8]	1327	**88.2%**	85.2%
(8,12]	618	**84.3%**	81.6%
(12,100]	118	**67.8%**	52.1%
-	31678	89.5%	**90.0%**

The 20% criterion for XGBoost (SE) and XGBoost (SPE). The best in each price class is in boldface.

Can we introduce some kind of weighting?

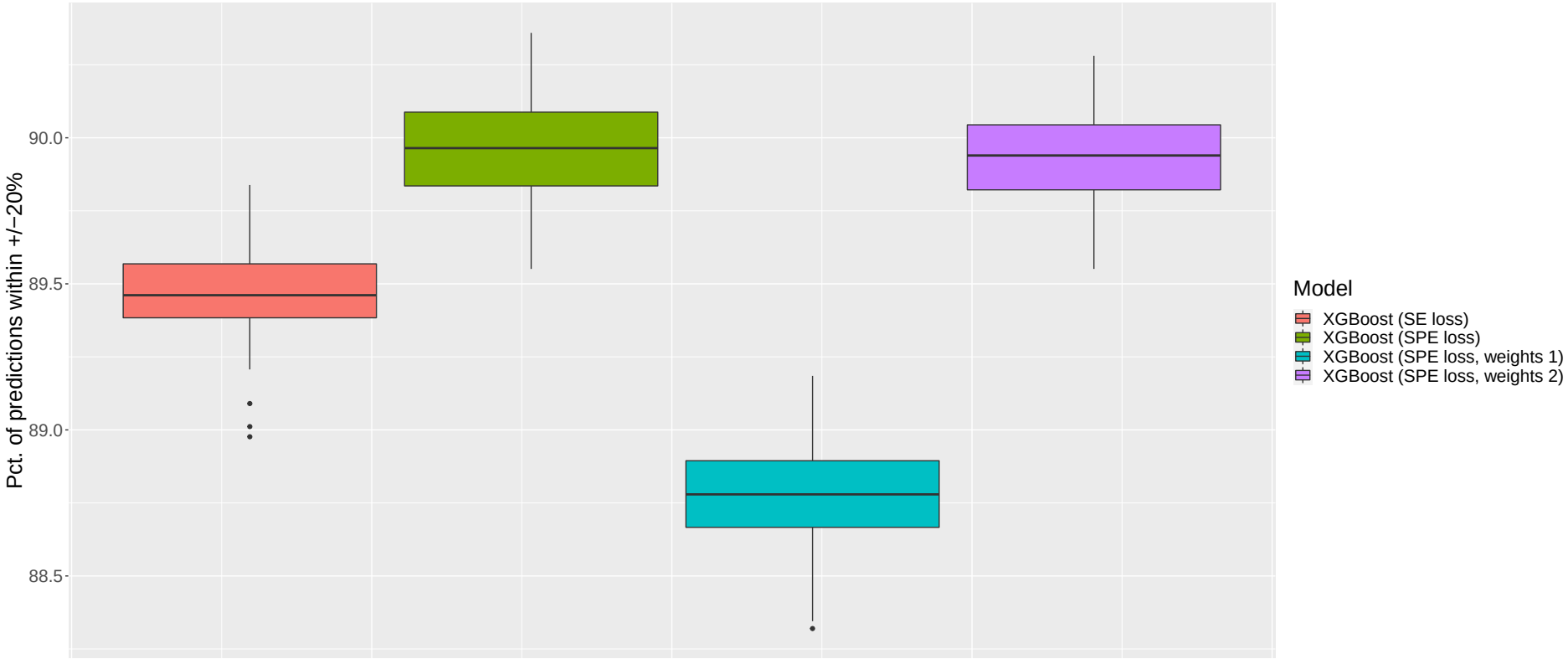
- **Weight Strategy 1:** $w_j = N_{max}/N_j$, i.e. give higher weight to price classes with few observations.
- **Weight Strategy 2:** $w_j = j$, i.e. weights that are linearly increasing for each price class.

Price Class	N	Weight Strategy 1	Weight Strategy 2
(0,2]	6903	$18431/6903 \approx 2.67$	1
(2,4]	18431	$18431/18431 = 1$	2
(4,6]	4284	$18431/4284 \approx 4.3$	3
(6,8]	1327	$18431/1327 \approx 13.9$	4
(8,12]	618	$18431/618 \approx 29.8$	5
(12,100]	118	$18431/118 \approx 156.2$	6

As expected, some improvement in higher price classes

Price Class	N	XGB (SE)	XGB (SPE)	XGB (SPE), WS1	XGB (SPE), WS2
(0,2]	6903	80.4%	82.7%	**83.9%**	81.2%
(2,4]	18431	93.0%	93.6%	90.86%	**93.6%**
(4,6]	4284	**90.7%**	90.0%	89.0%	90.6%
(6,8]	1327	88.2%	85.2%	**88.7%**	87.4%
(8,12]	618	**84.3%**	81.6%	84.3%	84.1%
(12,100]	118	67.8%	52.1%	**69.1%**	64.0%
-	31678	89.5%	**90.0%**	88.8%	89.90%

No overall improvement with higher weighting of some price classes



We run each model 100 times because of small variations due to train/test split. Figure shows box plot of these 100 simulations.

Summary and plan ahead

Summary:

- AVMs are trained to optimize absolute error, but evaluated using (a variation of) relative error.
- We propose a relative loss function and show that XGBoost can be improved slightly with this.
- We try introducing weighting of data to improve performance further, but without luck so far.

What's next?

- More systematic approaches to weighting of data. Perhaps more balanced categories are better.
- Comments and suggestions are more than welcome!

References I


Chen, T. and Guestrin, C.
'XGBoost: A Scalable Tree Boosting System'
'Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data ining'

UiO : Department of Mathematics
University of Oslo

Anders Hjort

House Price Prediction: Classical Methods vs. Machine Learning Methods

WEAI 96th Annual Conference