

IFC-Bank of Italy Workshop on "Data science in central banking: enhancing the access to and sharing of data"

17-19 October 2023

Error spotting with gradient boosting: a machine learning-based application for central bank data quality¹

Csaba Burger,
Magyar Nemzeti Bank,

Mihály Berndt,
Clarity Consulting

¹ This contribution was prepared for the workshop. The views expressed are those of the authors and do not necessarily reflect the views of the Bank of Italy, the BIS, the IFC or the other central banks and institutions represented at the event.

Error spotting with gradient boosting

A machine learning-based application for central bank data quality

Csaba Burger, PhD, CFA, data science advisor at the Central Bank of Hungary (MNB)
Mihály Berndt, data scientist at Clarity Consulting

Abstract

Supervised machine learning methods, in which no error labels are present, are increasingly popular methods for identifying potential data errors. Such algorithms rely on the tenet of a 'ground truth' in the data, which in other words assumes correctness in the majority of the cases. Points deviating from the ground truth, outliers, are flagged as potential errors.

This paper implements an outlier-based error-spotting algorithm using gradient boosting, and presents a blueprint for the modelling pipeline. It uses a cross sectional view on the loan-to-value and its related columns of the Credit Registry (Hitelregiszter) of the Central Bank of Hungary (MNB), and introduces a set of synthetic error types. The paper shows that gradient boosting is not materially impacted by the choice of the imputation method, hence, replacement with a constant, the computationally most efficient, is recommended. Second, the Huber-loss function is better in capturing data errors. Finally, errors in the target variable are captured best, while errors in the predictors are hardly found at all. These empirical results may generalize to other cases, depending on data specificities.

Keywords: data quality, machine learning, gradient boosting, central banking, loss functions, missing values

JEL classification: C5, C81, E58

Acknowledgements: the authors would like to express their gratitude to Lívia Réka Ónozó for her insightful comments on a previous version of this paper. However, only the authors should be held responsible for any errors, omissions or opinions expressed herein.

Contents

1. Introduction.....	3
Supervised learning for data error spotting	4
The sparsity-aware logic of xgboost.....	5
The machine learning pipeline.....	7
Data and Methodology.....	9
Data	9
Modelling pipeline.....	9
Bayesian optimization for hyperparameter-tuning	12
Synthetic errors and model evaluation	13
Results.....	15
The baseline model	15
Hypotheses 1 and 2	18
Testing Hypothesis 3	20
Conclusion.....	22
References.....	23
Appendix.....	25

1. Introduction

Granular central bank data, a relatively recent phenomenon of the last decade, has unique characteristics. First, it is usually delivered by a manageable number of data providers, which produce their data using their own scripts. This may introduce data errors on provider level. Second, such data sets may have several hundred columns, of which some are used only for a subset of its observations. To illustrate: collateral value is only relevant for loans with collateral, hence, it is missing for others. Third, relationships between data columns rely on economic processes, hence, the term 'outlier' or 'error' may win additional interpretations, should an observation not correspond to an expected relationship.

Once rule-based data errors are taken care of (format and constraint violations, values that violate syntactic and semantic constraints, duplicates (Rahm and Do, 2000, Kim et al., 2003), outlier detection algorithms may be applied. While the use of unsupervised outlier detection methods is more widespread, supervised learning algorithms *"empower learning methods with application-specific knowledge so as to obtain application-relevant anomalies"* (Aggarwal, 2017: 219). A specific subset of supervised methods used for data error identification is where there is no *ex ante* label for data errors. Particularly the 'attribute-wise learning for scoring outliers' (ALSO) (Paulheim and Meusel, 2015) has gained the attention of central bankers (Benatti, 2018), where a target variable of the granular data set is explained with the help of other variables, predictions are calculated, and the residuals of each observation serve as the starting point for outlier detection.

This paper is an empirical study of a gradient boosting error detection mechanism using synthetic errors we introduced to the data. It proposes a modelling pipeline to uncover potential data errors in a central bank granular data set. By doing so, it calculates how two evaluation metrics, calculated on the test dataset, change when certain modelling decisions are made. The metrics were: (1) the share of discovered errors out of all errors, and (2) the relationship between the share of intentional errors within outliers, compared to the share of intentional errors in the whole test data set. The modelling decisions were about the imputation of missing values, the choice of the xgboost loss function and the location of the error. The data used for illustration purposes is the Loan-to-value (LTV) relevant mortgage portfolio within the Credit Registry (*Hitelregiszter*) of the Central Bank of Hungary.

The findings underscore the fact that with xgboost, the using a constant value plus a flag, may be a decision as good as a prediction-based imputation to deal with missing values. Moreover, an xgboost with a Huber loss function seems capture more data errors than using the squared error loss function. Finally, the algorithm finds non-trivial errors in the target variable far better than errors in the explanatory values, hence, it may help in directing human attention towards the location of the actual error. We believe that our paper offers points to consider in future studies.

The paper is structured as follows. The next two sections contain the recommendations the literature holds on the xgboost-based outlier detection

pipeline. The fourth section discusses the data and methodology in detail, while the fifth presents the outcome of the error simulations. The final section concludes.

Supervised learning for data error spotting

Data error may refer to a set of issues, including rule-violations (e.g. a NOT NULL constraint), semantic pattern errors (such as a value outside 0 and 100, when it should be within), duplicates (best captured with record-linkage algorithms), or to outliers (Abedjan et al., 2016). The latter represents a subset, and have to fulfill two requirements. First, they are quantitative or qualitative deviations from an expected true value, and second, they should be confirmed to be errors. Outlier-based machine learning algorithms can do the former, whereas error-confirmation requires additional human intervention.

Hawkins (1980) defined outliers as observations which deviate from others "*as to arouse suspicions that it was generated by a different mechanism*". In that sense, it assumes the existence of a "ground truth", a relationship between variables which are expected to represent the baseline mechanism. For data quality purposes the 'ground truth' hypothesis assumes that most of the data on which the model is trained is right. Since it is hardly possible to limit the error to the test data only, the error will be present in the train data too. This means that any algorithm only flags outliers if they break the learned function, which is not necessarily the ground truth.

Supervised learning for outlier detection may not be the first choice. Unknown data errors are untagged, hence, the target variable cannot be a label stating if an observation contains an error or not. We often do not know a priori how errors look like, they may be heterogeneous and stem from a variety of sources (Heidari et al., 2019). Because of these, unsupervised algorithms (Aggarwal, 2017) are popular. Among these, distance-based methods identify 'densely' located records as normal observations, and rare instances as errors. Such an approach, however, faces problems with the increase in the number of dimensions, and may struggle with extreme values.

To overcome these problems, Paulheim and Meusel (2015) propose a novel way to use supervised learning for identifying outliers. They assume erroneous observations do not reflect the true relationship between the variables. They propose a method called 'attribute-wise learning for scoring outliers' (ALSO), in which a target variable is explained with several explanatory features. Subsequently, the residual, the difference between the prediction and the actual value, is calculated. The authors loop through all columns as target variable, and use the residual values for each instance to calculate a final outlier score¹ for each observation. Their approach translates the

¹ If i'_k is the predicted and i_k the actual value for an observation using the variable k as the target variable, its final, unweighted outlier score as proposed by Paulheim and Meusel (2015) is calculated as follows: $\sqrt{\sum_{k=1}^n (i_k - i'_k)^2}$. One may weigh the difference between i_k and i'_k to reflect the strength of each relationship using their relative squared error values. With the weights assigned, the weighted

data points into a new residual feature space. Within it, outliers can be considered as observations that are far away from the origin².

Benatti (2018) follows Paulheim and Meusel (2015) in that he also creates a model for each column using the remaining columns and calculates residuals for each instance. At the same time, he does not aggregate the obtained residuals into one single outlier score. Instead, he carries out a clustering of the residual values to identify groups of observations. According to his logic, observations far away from such clusters may be flagged as potential data errors.

The work of Benatti (2018) is relevant for further reasons. First, in his paper he looks for potential data errors in a granular central bank dataset, a similar challenge to ours. Second, his model was specified with a gradient boosting algorithm (xgboost), which provides sufficient flexibility to account for categorical and data provider-level specificities. It is capable of learning highly non-linear relationships and variable interactions, works well with high dimensional data, and it can deal with missing data within the modelling phase (Chen and Guestrin, 2016). In short, xgboost is an excellent tool to learn true, complex relationships within the data.

The sparsity-aware logic of xgboost

The xgboost algorithm entails a sequential development of several decision trees³. The first tree attempts to predict the target variable with a simple aggregation⁴, the second tree predicts the error of the first. Then the predictions of these two trees are added and a new prediction error is calculated. The following tree is grown on the error of the previous trees, and its prediction is added to the last one. This process is repeated several times. The final number of trees can be determined by cross-validation or Bayesian Optimization, discussed later.

This section presents a simplified overview of the xgboost, based on Chen and Guestrin (2016). A tree ensemble model uses K additive functions to predict the output.

residuals in the k-th model are: $o(i) := \sqrt{\frac{1}{\sum_{k=1}^n w_k} \sum_{k=1}^n w_k (i_k - i'_k)^2}$, and the weights are calculated by
 $w_k := 1 - \min(1, R_k)$, where $R_k := \sqrt{\frac{\sum_{j=1}^m ((i_j)_k - (i'_j)_k)^2}{\sum_{j=1}^m ((i_j)_k - \bar{k})^2}}$.

² These are not distance-based outliers, but points which do not fit in the relationships between the features, which makes its use particularly appropriate for error spotting.

³ In short, these are not the usual decision trees (sometimes called xgboost trees), and their purpose can be viewed as clustering the residuals by their directions. Residuals are the differences between the predictions and the actual values, and the direction is determined by whether the prediction is greater (overestimation) or smaller (underestimation) than the actual value. The algorithm clusters the observations along the values of the features according to the values of the residuals.

⁴ The aggregation can be specified to be the mean (L2 case of the loss function), or the median (L1 case).

$$\hat{y}_i = \sum_{k=1}^K f_k(x_i), \quad f_k \in \mathcal{F}, \quad (1)$$

where \mathcal{F} is a set of regression trees, but the main difference compared to decision trees is that each regression tree contains a continuous score on each leaf node. Each f_k is an independent tree structure, the final prediction of a tree is the sum of the scores of the corresponding leaves. Since the "model includes functions as parameters, the model is trained in an additive manner". At iteration t the objective is the following:

$$\mathcal{L}^{(t)} = \sum_{i=1}^n l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t). \quad (2)$$

The authors did not use the cost function of the weak learner to fit the residual, but the second order approximation, which can be viewed as a cost/performance trade-off, and because of the Taylor's expansion it is a good approximation at the point we evaluate. The second order extension:

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^n \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t), \quad (3)$$

whereby $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ and $h_i = \partial_{\hat{y}^{(t-1)}}^2 l(y_i, \hat{y}^{(t-1)})$ are the first and second order gradient statistics on the loss function. At iteration t the goal is to achieve the greatest possible reduction of the loss. The authors measure the quality of a fixed tree structure q , by the following equation:

$$\tilde{\mathcal{L}}^{(t)}(\mathbf{q}) = \frac{1}{2} \sum_{j=1}^T \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T. \quad (4)$$

In an L_2 loss regression case the nominator is the sum of the residuals, squared, and the denominator is the number of residuals. It can be viewed as the algorithm's attempt to cluster the residuals by their direction and magnitude. It is impossible to enumerate all the possible tree structures q . Therefore, a greedy algorithm that starts from the single leaf containing every node (or possibly a subset controlled by a hyperparameter subsample) iteratively adds branches to the tree. The loss reduction after a split is calculated by the following equation:

$$\mathcal{L}_{\text{split}} = \frac{1}{2} \left[\frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda} \right] - \gamma, \quad (5)$$

where I_L and I_R are the instances of the left and right nodes after a split, and $I = I_L \cup I_R$. The gain after a split must be positive, otherwise the branch is removed. Hence, λ directly controls overfitting and reduce the prediction's sensitivity to individual observations, while γ post-prunes the regression trees.

An advantage of xgboost's sparsity-aware split finding is that observations with missing data are classified to the default direction at a split that is determined only

by non-missing data. There are two possible directions at each split for the candidates. The algorithm tries both directions, and determines the default direction by evaluating the splits by Equations (4) and (5).

The machine learning pipeline

In this paper, we discuss (1) missing value treatment options, the (2) use of various loss functions in xgboost, and (3) the identification of the erroneous variable, which helps in locating the data errors not just row-wise but column-wise too.

To deal with missing values among predictors, one rely on the sparsity-aware method of xgboost, which puts missing instances in the direction with the greatest loss reduction. Using this may reduce the bias on the train dataset, but may result in high prediction errors during the prediction phase, since the underlying 'ground truth' is not learnt by the model.

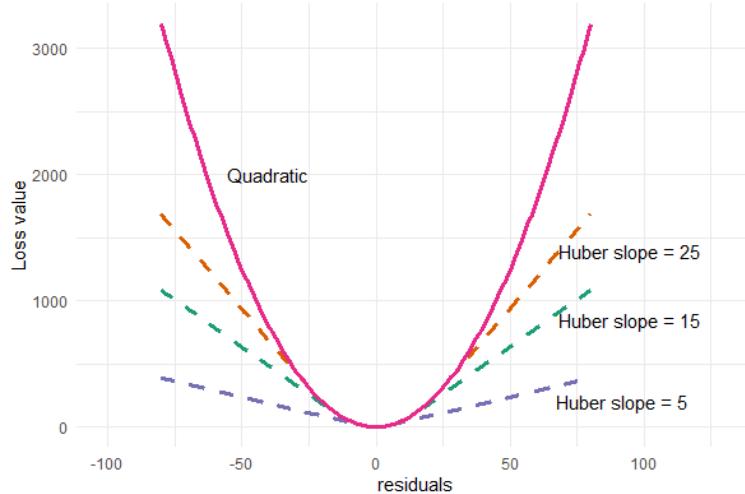
Alternatively, one may use explicit missing value replacement *before* modelling. Emmanuel et al. (2021) describe several estimator functions, stretching from the simple replacement with a constant, to a learner-function based prediction, using cases where values are filled. There is no single one best way to deal with missingness, particularly when it is an issue both during model training and during deployment (Khosravi et al. 2020). Despite that, Twala et al. (2008) do not find convincing difference in the performance of a classification tree when comparing a simple 'missing' flag with imputation. We also argue that a sufficiently flexible tree-based algorithm will not require a sophisticated missing value imputation. **We argue that replacement with a constant and the addition of an imputed value flag variable (1/0) is not inferior to an imputation algorithm. In addition, we recommend being cautious with the xgboost sparsity-aware model fitting, since it may not learn the underlying 'true' relationships within the data (hypothesis 1).**

The second step, referred to as 'dealing with high-leverage point outliers' is an expectation stemming from the data error identification approach. Our estimator function should not be materially impacted by the data errors we are about to identify, which may be achieved with a robust estimator function. Begashaw and Yohannes (2020) provide a solution in a linear regression setting; Sadouk et al. (2020) propose one for deep learning applications. With gradient boosting we propose changing the default objective function which is minimized during the model fitting: we argue for replacing the squared loss function with the Huber loss function. This latter makes sure that the residual (denoted as r) turns from quadratic (as in the case of squared loss) to linear above a certain threshold. This threshold is often referred to as the Huber-slope, denoted as δ , and its value is determined during the

hyperparameter-optimization process, discussed later. The simple⁵ Huber-loss function is reflected by **Equation 6** and by **Figure 1**.

$$L_\delta(r) = \begin{cases} \frac{1}{2}r^2 & \text{for } |r| \leq \delta \\ \delta(|r| - \frac{1}{2}\delta) & \text{otherwise} \end{cases} \quad (6)$$

Figure 1. Loss values of a quadratic and of several Huber-loss functions



The Huber-function ensures that small deviations are squared, but large residuals count in a linear manner only. As a result, parameter tuning will not chase outliers and can learn the underlying ("true") function. **We argue that an xgboost with a Huber-loss function is better suited to uncover data errors than the squared loss objective function (Hypothesis 2).**

Finally, the outliers of one single model are investigated for being potential data errors. **We argue that errors are better identified when they are in the target variable as opposed to the explanatory variables (Hypothesis 3).** This hypothesis relies on the assumption that there is a certain degree of (linear or non-linear) relationship between explanatory variables. If a data error distorts an explanatory variable 'A', the other explanatory column 'B', which is not independent from 'A', may take over its role in explaining Y. As a result, A loses its importance in the final xgboost function, while B's feature importance increases. Because of that, the algorithm may not yield outliers on these errors within 'A', only if they are in the target. In sum, it may be a good practice to look for the data error within the target variable and not in the explanatory variables.

⁵ The xgboost implementation uses the pseudo Huber-loss function, which is a smooth approximation of the Huber loss function, meaning continuous derivatives: $L_\delta(r) = \delta^2 \left(\sqrt{1 + (r/\delta)^2} - 1 \right)$

Data and Methodology

Data

The Credit Register (Hitelregiszter) of the Central Bank of Hungary (MNB) is a monthly updated collection of all loan contracts issued in Hungary, containing 22 reporting tables, 482 attributes and 31 identifiers. These together usually yield between 30-60 million new lines each month. The dataset goes through of around two thousand logical quality rules. Aggregate time series are checked with the help of ARIMA- and further filters, whereas static changes are probed on record / cell level. Finally, a cross-sectional algorithm, the subject of this paper, is applied on the data.

This paper uses the loan-to-value (LTV) column and its related fields of the dataset. This limits the analysis to mortgages and home equity loans of the residential sector, issued after 1st October 2021, the date of a recent legal change. The final table is a combination several Hitreg tables, comprising data on the debtor(s), collateral(s) and on the loan contract. Altogether we work with 73 thousand lines of one single reporting date of 30th September 2022, using 274 columns. Metrics and the distribution of the target variable, and of a subset of the explanatory variables, are presented in the Appendix.

The target variable was the LTV-value at credit issuance, which is subject to strict legal limitations, hence its accuracy is of utmost interest. As a consequence, we only kept the columns which were relevant at the time of issuance (to illustrate: current outstanding loan amount was removed, since it has little relevance for the LTV at issuance).

In theory, calculating LTV is straightforward: the loan amount divided by the allocated collateral value. In practice, however, it is rather complex, since many decisions, including the choice of the collateral evaluation method and the allocation process of a collateral to the loan contract, are at the data providers' discretion. As a result, the columns at hand are insufficient to calculate LTV directly. The relationship between LTV and the other columns is complex, which makes xgboost an appropriate choice.

In short, LTV has highly non-linear relationship with the explanatory variables, and it does not contain missing values, as it is an enforced condition of a successful data submission to the central bank. As a result, it may serve as a prime candidate for hypothesis testing.

Modelling pipeline

Although xgboost predictions are not affected by multicollinearity, it may distort feature importance. Since we expect certain degree of interpretability, we kept one out of the pairwise collinear variables only (over 0.90). We also eliminated quasi constant values, where the frequency ratios of the most frequent and the second most

frequent values were extremely high (the variance of the feature close to zero), to avoid overfitting or misleading conclusions.

In a similar manner, we recoded categorical and discrete variables with rare values. Any category below a threshold of 2.5 percent was recoded into an 'other' category, and upper, rare values of discrete numerical values were summarized (e.g. the number of residential real estate collaterals above the value of three was recoded into three). Categorical variables with missing values were given the value of 'missing'. In most cases these were '*missing not at random*' (MNAR) variables, and in such cases, simple flagging is preferred to imputation.

Next, we converted dates into the number of days elapsed between the start date and the reporting date.

During encoding (dealing with discrete non-numeric variables), we found that *label encoding*⁶ improves model performance most even for variables that enumerations cannot be sorted (also known as nominal variables). It may be explained by avoiding the "*curse of dimensionality*": as the number of predictors increases, the difference between the maximum and minimum distances⁷ approaches to zero. Boosting algorithms tend to suffer from the curse of dimensionality as they tend to overfit, hence label encoding can be recommended.

The next step contained the treatment of missing values for continuous variables (**Figure 2**), the focus of the first hypothesis (**Table 1**). We have not used columns where the share of missing values exceeded 20 percent at all. Among the rest, the feature with the most missing values had a missing value share of 13 percent, followed by several columns with 12 percent. There seems to be some degree of systematic rule for missingness, since certain feature values are missing for the same observations (**Figure 2**). This is to be expected: some variables are not interpreted for certain loan contracts.

Three approaches were assessed to fill missing fields. First, missing values were recoded into an arbitrary outlier value (e.g. collateral market value of zero HUF). The model may learn that this special value is not necessarily linked to an economic meaning.

⁶ Label encoding converts a categorical variable to a numerical value. While at first it may sound surprising, this kind of treatment works well with the tree-based xgboost-method, since it can learn any difference between 1 and 2, and this is not necessarily the same difference as between 2 and 3. In addition, label encoding does not increase dimensionality, as opposed to the one-hot-encoding technique.

⁷ Maximum and minimum distances are distances calculated between the furthest and nearest observations respectively.

Figure 2. Missing values among the continuous variables. Please see the Appendix for further details

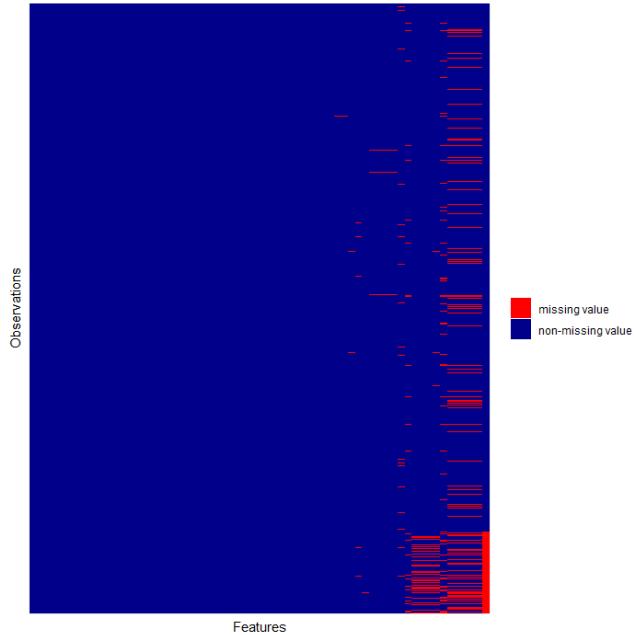


Table 1. Missing value treatment options for **Hypothesis 1**

Missing value treatment	Description	Assumption
Replacement with a constant	The constant (zero) is infrequent or nonexistent in the original data	Xgboost is capable of learning that it is a special value. If the ratio of missing values exceeds 5 percent, an indicator column is added to show the manually filled rows ⁸ .
Imputation with MissRanger	All complete predictors are used (but not the target)	There are existing relationships between predictors
Xgboost sparsity-aware algorithm	The dataset with missing values is fed directly to the xgboost optimization.	This approach puts instances with missing values in the class with the greatest loss reduction when growing a new branch on a tree. This approach delivers a good model on the train data but may not capture the 'ground truth'.

In the second approach, we imputed missing values using the MissRanger package of R (Stekhoven et al. 2012). This is a sophisticated and parallelized, random forest-based algorithm, which uses both categorical and continuous data, and optimizes the imputation process without the need of a test data set. We chose to use all complete predictors to impute the missing values for incomplete predictors. The target variable was not used for the imputation to prevent data leakage. The third method was not to replace missing values at all: we left blank cells blank and let the xgboost deal with them, as discussed earlier.

⁸ The presence or absence of the additional flag column is not material: it changes little on model performance or variable interpretation.

Bayesian optimization for hyperparameter-tuning

The last step was the design of the xgboost function. One choice is about the loss function, subject to our second hypothesis and discussed above. The Huber slope parameter, which makes the loss function quadratic up to it, and linear above, is determined with a Bayesian optimization procedure (Snoek et al., 2012), along with all other hyperparameters. This is an efficient method that fits a Gaussian process to the results, looks for hyperparameters which obtain the best evaluation metric (RMSE or the mean pseudo-Huber error) on the test data.

The objective function is the evaluation metric based on the hyperparameter setups, is unknown. The algorithm samples hyperparameter sets, observes the evaluation metric and updates the prior form of the function. There is a predefined similarity measure (Kernel) between observations, whereby similar hyperparameters correspond to similar evaluation metrics. The next samples are determined by the acquisition function, which is based on the previously queried observations, the Kernel function and other hyperparameters, that is related to the exploitation-exploration trade-off, which corresponds to the expected value of the evaluation metric and the variance, based on the previously explored areas and the updated posterior function.

The Bayesian Optimization framework is cited from Shahriari et al. (2016) and Garrido-Merchán et al. (2023). To retrieve the optimum hyperparameters \mathbf{x}^* of the unknown objective function, we solve the following:

$$\mathbf{x}^* = \arg \min_{x \in \mathcal{X}} \mathbf{f}(x). \quad (7)$$

The Bayesian Optimization framework in our case can be defined by the following tuple:

$$\mathcal{A} = (\mathcal{GP}, \alpha(\cdot), p(f(x)|\mathcal{D})), \quad (8)$$

where $f(x)$ is the optimized function, \mathcal{GP} is a Gaussian Process, $\alpha(\cdot)$ is an acquisition function, and $p(f(x)|\mathcal{D})$ is a predictive distribution (Garrido-Merchán et al, 2023). The task is to provide a predictive distribution for test observations $\{x_i^*\}_{i=1}^M$ with a given dataset $\mathcal{D}_n = \{(x_i, y_i)\}_{i=1}^N$. A GP is fully defined by the following:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N} \left(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \hat{\mathbf{K}}_{\mathbf{X}, \mathbf{X}} & \mathbf{K}_{\mathbf{X}, \mathbf{X}^*} \\ \mathbf{K}_{\mathbf{X}^*, \mathbf{X}} & \mathbf{K}_{\mathbf{X}^*, \mathbf{X}^*} \end{bmatrix} \right), \quad (9)$$

where $y_i = f(x_i) + \varepsilon_i$ are the noisy observations of the optimized function $f(x)$ with $\varepsilon_i \sim \mathcal{N}(0, \sigma_\varepsilon^2)$, the train and test observations are arranged in matrices \mathbf{X} and \mathbf{X}^* , \mathbf{f} and \mathbf{f}^* are the unobserved true function outputs for all the inputs, $\mathbf{K}_{\mathbf{X}, \mathbf{X}}$ is N-by-N matrix containing the similarities between the observations calculated by the Radial Basis Function $\kappa(\mathbf{x}_i, \mathbf{x}'_i | \boldsymbol{\tau}) = \sigma^2 \exp \left(-\frac{1}{2} \left(\frac{\mathbf{x}_i - \mathbf{x}'_i}{\ell} \right)^2 \right)$, where the hyperparameters of the Kernel denoted by $\boldsymbol{\tau}$, and $\hat{\mathbf{K}}_{\mathbf{X}, \mathbf{X}} = \mathbf{K}_{\mathbf{X}, \mathbf{X}} + \sigma_\varepsilon^2 \mathbf{I}$. Since the observations and the true

function values at the test points are jointly distributed as a multivariate Normal, the predictive posterior distribution $p(f(x^*)|\mathcal{D})$ are given by:

$$\mu_n(x^*) = \mathbf{K}_{X^*,X} \tilde{\mathbf{K}}_{X,X}^{-1} \mathbf{y}, \quad (10)$$

$$\sigma_n^2(x^*) = \mathbf{K}_{X^*,X^*} - \mathbf{K}_{X^*,X} \tilde{\mathbf{K}}_{X,X}^{-1} \mathbf{K}_{X,X^*} \quad (11)$$

The predictive distributions are the estimates of unknown functions areas, and the next samples are determined by the acquisition function. It deals with the exploration-exploitation trade-off, in the sense of the search space and the promising areas (Shahriari et al, 2016). In our analysis we used the upper confidence bound as an acquisition function:

$$\alpha_{UCB}(x^*; \mathcal{D}_n) := \mu_n(x^*) + \beta_n \sigma_n(x^*). \quad (12)$$

The approach, preferably parallelized, vastly reduces computation time compared to a brute force-based grid search. The permitted bounds for the hyperparameter-optimization, defined after a set of initial experiments, are reflected by **Table 2**.

Table 2. Permitted hyperparameter ranges

Hyperparameter	Descipriton	Bounds
Eta (learning rate)	Step size used in update	0.01 - 0.1
Max depth	Maximum depth of a tree.	4 - 8
Gamma	Minimum loss reduction required to make a further partition on a leaf node of the tree.	0-8
Alpha	L1 regularization term on weights.	10-20
Lambda	L2 regularization term on weights.	1-12
Huber-slope*	Defines the range for the loss function piecewise.	10-30

* with Huber loss function only. The maximum number of trees was set to 280

Synthetic errors and model evaluation

We introduced three types of synthetic errors to the data (**Table 3**). Synthetic errors are often used to test data quality-algorithms (Abedjan et al. 2016). We added the following errors to 5 percent of randomly selected cases (both in train and in test datasets):

- LTV divided by 100 (an LTV of 50 percent is to be submitted as '50', but is often submitted as 0,5)
- LTV set to 80 (as if a dummy value was submitted, which also happens in reality). This error type was applied to observations with a true LTV of 60 or less to have a sufficiently sizeable error.

- LTV multiplied by a sample drawn from uniformly distributed ranges of $U(0.3, 0.5)$ and $U(1.2, 1.4)$ ⁹ random variables. The two ranges are not symmetrical to limit the number of errors above 100 (percent) of LTV.

In a separate experiment, errors were introduced to one of the predictors. We chose to alter the second most important predictor in terms of gain in the feature importance matrix (described later), the allocated collateral value. Data providers have more discretion during the allocation process than it is the case with loan amount (the most important feature) - this latter is explicitly written in a loan contract. Therefore, variations, errors and outliers are more probable in the case of the allocated collateral value. The errors we introduced were:

- Values were multiplied by a random value sampled from the uniformly distributed ranges $U(0.3, 0.5)$ or $U(1.2, 1.4)$. This in essence means a multiplication by a value in any of the mentioned bounds.
- Values were set to 10 mln HUF (approx. 25 thousand EUR). This error type was applied to observations with a true allocated collateral value of 40 mln HUF or more to have a sufficiently sizeable error. Such an error would occur when dummy value is reported to the central bank.

Table 3. Data errors used for algorithm evaluation

Error location	Error description	Error rationale
Response variable	values divided by 100	An LTV of 50 percent is expected to be submitted as '50', it is often submitted as 0,5
Response variable	values set to 80	as if a dummy value was submitted
Response variable	Values were multiplied by a random value. This random value was drawn for each observation from a uniform distribution of between $U(0.4, 0.6)$ and $U(1.2, 1.4)$.	A random error
Predictor (allocated collateral value)	Values set to 10 mln HUF	The use of a dummy value or an internal cap on allocation
Predictor (allocated collateral value)	Values were multiplied by a random value. This random value was drawn for each observation from a uniform distribution of between $U(0.4, 0.6)$ and $U(1.2, 1.4)$.	A random error

The outlier detection approaches were evaluated with two metrics (**Table 4**). First, we look at the share of discovered errors out of all errors. An error was considered as discovered when its absolute residual was above a pre-defined threshold of 20 (since the target variable is expected to be between 0 and 80 (a legal limitation), an absolute threshold does not materially distort the results). Ceteris paribus, the more errors we identify, the better the model is. At the same time, we need an additional metric: the share of intentional errors within outliers, compared to the share of intentional errors

⁹ We used the usual abbreviation of the uniform distribution (denoted by U).

in the whole test data set (referred to as 'lift'). Error density should be considerably higher in the outlier set than in the data.

We tested two outlier definitions. First was based on a pre-defined (unstandardized¹⁰) residual value, above which all observations were treated as outliers. The second definition relied on the top pre-defined share of observations, in our case, 5 percent, which were classified as outliers, when ranked according to the absolute value of its residuals. One argument for this second definition could be the limited time available to follow up outlier-issues with the data providers.

Table 4. Error detection evaluation metrics

Metric	Metric description	Outlier definition
Share of known errors discovered	Synthetic errors in the outlier set as a proportion of all synthetic errors	Definition 1: Outlier set defined as the observations with an absolute residual of a pre-defined value or higher Definition 2: A pre-defined share of observations with the greatest residual values
Lift	Frequency of synthetic errors in the outlier set, divided by the frequency of synthetic errors in the entire data	

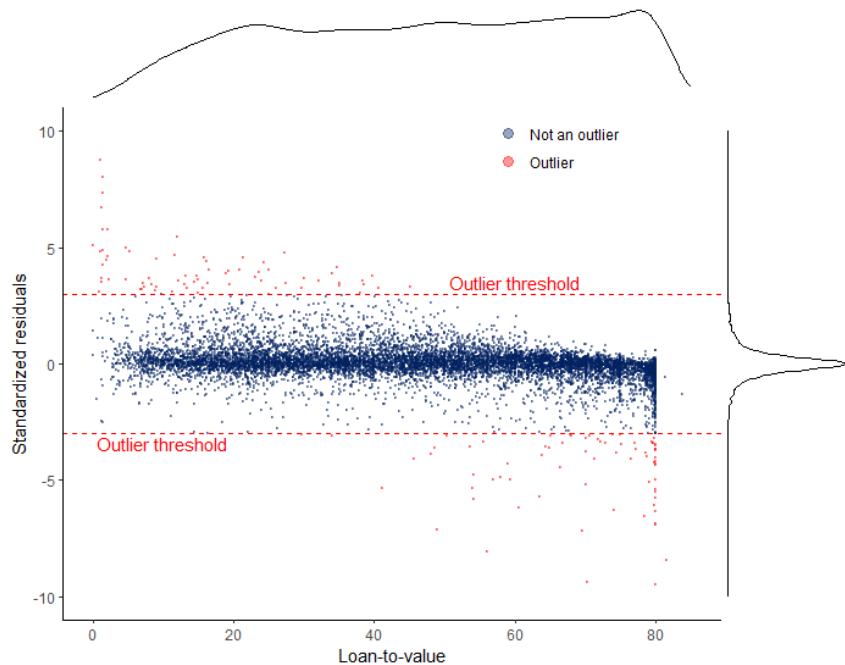
Results

The baseline model

We prepared a baseline xgboost model, imputed missing values using a constant, and a squared loss function (and with no synthetic errors). Although the model's residual plot still reflects a systematic pattern (**Figure 3**) - it overestimates low observations and underestimates high ones - it performs well. The RMSE is 5.6 percent, and the share of outliers is 1.4 percent, using the cutoff value on *standardized* residuals of 3. A sample of our flagged outliers were identified as data errors by domain experts. The algorithm also found intuitive errors (e.g. LTV between 0 and 1 instead of 0 and 100).

¹⁰ We decided on using the unstandardized value for better interpretability, given the fact that LTV is expected to move between 0 and 100. In more heterogeneous cases or if the ALSO-algorithm is to be carried out in its full extent, one may standardize the residuals.

Figure 3. Residual plot of the baseline model with marginal distributions (graph contains a random sample of data points only)



The two most important features in the model are loan amount and allocated collateral value at loan origination (**Table 5**). Since LTV at loan origination is related to the division of these two values, it is promising to see these features on the top of the list. The reason why we say LTV is only related to these two features is that banks are permitted to apply certain rules (caps, modifications) in using these values during LTV-calculation both to the nominator and to the denominator.

Table 5. Feature importance in the baseline model (top 10)

Feature	Gain	Cover	Frequency
Loan amount (in HUF)	0,32	0,13	0,07
Allocated collateral value (at loan origination)	0,25	0,07	0,09
Full collateral market value at loan origination	0,12	0,06	0,06
Collateral value	0,05	0,03	0,04
Monthly repayment amount	0,04	0,07	0,06
Effective interest rate	0,03	0,02	0,03
Financed real estate type ¹¹	0,02	0,03	0,02

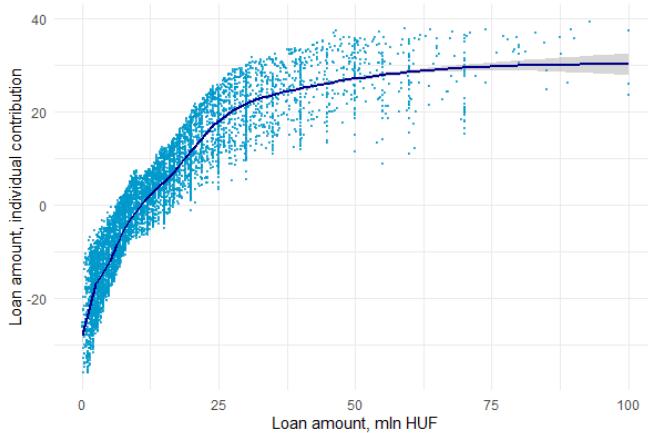
¹¹ The categorical variable converted to a numerical value using label encoding. While at first it may sound surprising, this kind of treatment works well with the tree-based xgboost-method, and does not increase dimensionality, as opposed to the one-hot-encoding technique.

Days past since collateral value determination ¹²	0,02	0,02	0,03
Rate driver ¹³	0,02	0,00	0,01
Expected loss	0,01	0,04	0,04

Figure 4 and **Figure 5** depict the individual feature contributions for a randomly selected 10 thousand observations for the two most importance features (by gain). The individual feature contribution is a weight the given value of the predictor has on the target. Since xgboost develops a complex set of trees, including predictor interactions (see section 4.2), the same predictor value may have varying impacts on the prediction. Each point on the plot gives the impact the actual feature set of the observation on the target.

These plots reflect the division between the 'Loan amount' (which is related to the nominator) and the 'Allocated collateral value' (which is related to the denominator). Note, however, the strange curve of the 'Allocated collateral value' around zero. This reflects the impact of the missing value treatment. The cases where this value was missing were given decisively heterogeneous and broad individual contribution values by the model. We assume that here the algorithm learned the underlying function along other variables. Beyond zero, the individual contribution plot for this predictor is reminiscent of a hyperbole.

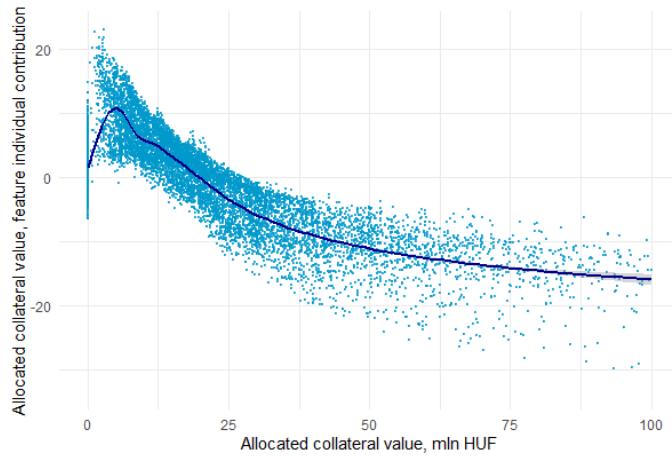
Figure 4. Individual feature contribution of 'Loan amount' (with an illustrative LOESS-function)



¹² The maximum amount, in case of multiple collaterals.

¹³ The categorical variable converted to a numerical value using label encoding. While at first it may sound surprising, this kind of treatment works well with the tree-based xgboost-method, and does not increase dimensionality, as opposed to the one-hot-encoding technique.

Figure 5. Individual feature contribution of 'Allocated collateral value' (with an illustrative LOESS-function)



Hypotheses 1 and 2

We classified all observations with an absolute residual of 20 or higher as outliers, and calculated the evaluation metrics. The 80-dummy and the div_100 error types were discovered with a lift value of 10-12, with the exception the 'none' (no explicit missing value replacement) and Huber-loss combinations (**Table 6**). In this latter case, the share of outliers was similar, but the models' RMSE and MAE values and the total share of outliers were higher, and therefore lift values were lower. This poor performance of the 'none' – 'Huber-loss' combinations were consistent across all three error-types.

Table 6. Evaluation metrics when using an absolute outlier threshold of 20; with a synthetic error in the target variable

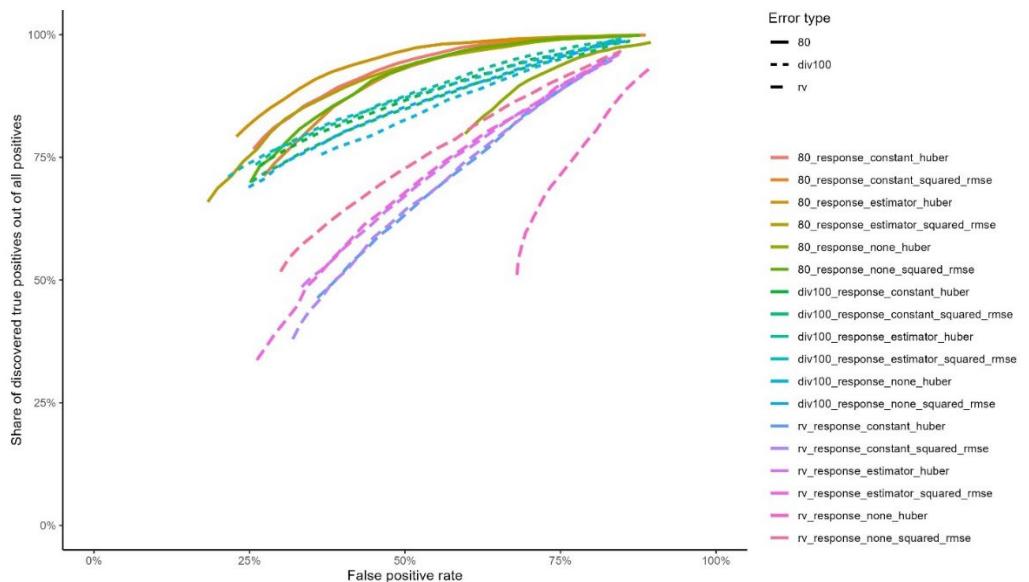
Error type	Missing value replacement	Loss function	Outliers as % of total	Share of discovered errors	Lift	RMSE	MAE
80	estimator	Huber	6,8%	88,9%	13,00	12,10	6,82
80	none	Huber	13,3%	88,0%	6,60	17,90	9,73
80	constant	Huber	6,8%	87,2%	12,80	11,60	6,54
80	constant	squared loss	6,9%	85,3%	12,40	11,30	7,00
80	none	squared loss	6,5%	83,1%	12,80	10,90	6,69
80	estimator	squared loss	5,6%	80,1%	14,30	10,20	6,14
div100	none	Huber	8,1%	82,3%	10,20	13,30	6,97
div100	estimator	Huber	6,4%	81,3%	12,60	12,40	6,32
div100	constant	Huber	6,6%	81,1%	12,40	12,30	6,34
div100	estimator	squared loss	5,9%	79,1%	13,50	11,50	6,26
div100	constant	squared loss	6,2%	78,4%	12,60	11,50	6,37
div100	none	squared loss	6,2%	78,1%	12,60	11,30	6,22

rv	none	squared loss	5,3%	64,2%	12,20	9,30	5,40
rv	none	Huber	11,2%	64,2%	5,76	22,10	9,59
rv	constant	Huber	5,4%	59,0%	10,90	9,24	5,36
rv	estimator	Huber	5,1%	58,8%	11,60	9,11	5,29
rv	constant	squared loss	4,4%	52,0%	12,00	8,25	4,86
rv	estimator	squared loss	3,6%	48,4%	13,30	7,62	4,49

There is no visible tradeoff between discovered error shares and lift values. Moreover, models with imputed missing values do not outperform models with a constant-replacement. Further, the Huber objective function performs better than the squared loss functions do in terms of the first metric, the share of discovered intentional errors. However, the share of outliers as of all observations is not necessarily larger with the Huber loss function, if a missing value imputation was used.

Figure 5 illustrates model performance¹⁴ over the various error types in the response variable when gradually decreasing the absolute outlier threshold value from 25 to 5. The x-axis shows the share of non-errors in the outlier set, in other words, the false positive rate, while the y-axis represents the share of known, outlier errors as a percentage of all synthetic errors. **Figure 6** reinforces model stability and the messages of our research.

Figure 6. False- and true-positive rates when moving the outlier threshold from 25 to 5 (synthetic errors in the target variable)



¹⁴ The difference **Figure 6** and a ROC-curve is that in our case, we do not have a direct probability value, only a cutoff outlier value.

Table 7 contains model metrics when a relative outlier threshold of 5 percent of all observations is used. In a similar vein, **Table 7** confirms that the estimation of missing values as imputation in our case delivers better outlier recognition performance than using a constant.

Regarding our initial Hypotheses: the Huber-loss often delivers better results than the squared loss, and an imputation is most often better than no imputations. These are indications in favor of Hypotheses 1 and 2, but every dataset should be tested for it.

Table 7. Error detection metrics when using a relative outlier threshold of 5 percent (synthetic error in the target)

Error type	Missing value replacement	Loss function	Outliers as % of total	Share of discovered errors	Lift	RMSE	MAE
80	estimator	Huber	5,0%	76,5%	15,30	12,10	6,82
80	constant	Huber	5,0%	75,7%	15,10	11,60	6,54
80	estimator	squared loss	5,0%	75,0%	15,00	10,20	6,14
80	none	squared loss	5,0%	73,1%	14,60	10,90	6,69
80	constant	squared loss	5,0%	72,3%	14,50	11,30	7,00
80	none	Huber	5,0%	54,7%	10,90	17,90	9,73
div100	estimator	squared loss	5,0%	74,5%	14,90	11,50	6,26
div100	estimator	Huber	5,0%	73,9%	14,80	12,40	6,32
div100	constant	Huber	5,0%	73,5%	14,70	12,30	6,34
div100	constant	squared loss	5,0%	72,3%	14,50	11,50	6,37
div100	none	squared loss	5,0%	71,8%	14,40	11,30	6,22
div100	none	Huber	5,0%	70,7%	14,10	13,30	6,97
rv	none	squared loss	5,0%	62,6%	12,50	9,30	5,40
rv	estimator	squared loss	5,0%	58,9%	11,80	7,62	4,49
rv	estimator	Huber	5,0%	58,4%	11,70	9,11	5,29
rv	constant	squared loss	5,0%	56,8%	11,40	8,25	4,86
rv	constant	Huber	5,0%	56,5%	11,30	9,24	5,36
rv	none	Huber	5,0%	31,6%	6,33	22,10	9,59

Testing Hypothesis 3

The outcome of the same experiments (recall that we use slightly different errors) is summarized by **Table 8**.

Table 8. Error detection metrics with an absolute outlier threshold of 20 ([synthetic error in the second most important predictor](#))

Error type	Missing value replacement	Loss function	Outliers as % of total	Share of discovered errors	Lift	RMSE	MAE
10	none	Huber	3,5%	3,0%	0,87	7,91	4,73
10	estimator	Huber	3,0%	2,1%	0,70	7,47	4,36
10	none	rmse	1,5%	1,6%	1,05	5,83	3,43
10	constant	Huber	2,3%	1,3%	0,57	6,98	4,21
10	constant	rmse	1,3%	1,1%	0,83	5,55	3,25
10	estimator	rmse	1,1%	0,9%	0,83	5,27	3,18
rv	none	rmse	2,0%	10,7%	5,41	6,30	3,69
rv	estimator	Huber	4,5%	7,9%	1,75	9,18	6,11
rv	none	Huber	2,5%	5,8%	2,31	7,10	4,29
rv	constant	Huber	3,0%	4,3%	1,43	7,72	4,75
rv	constant	rmse	1,5%	2,1%	1,41	5,84	3,46
rv	estimator	rmse	1,3%	1,9%	1,51	5,54	3,37

Table 8 shows dismal error recognition metrics with disappointingly low values: the models are not better than randomly selecting observations from the data. The metrics send a similar message with relative outlier threshold (not shown).

To understand the reason for that, we compare the feature importance matrix for the baseline model and for the model with synthetic errors in the predictor 'Allocated collateral value'. While the feature importance matrix can only signal changes and is not a mathematical proof, it indicates change directions. **Table 9** shows that the importance of this predictor decreases slightly for all three aspects (gain, cover and frequency). At the same time, the third predictor, exhibits higher importance values, among other changes. These moves point out the reason why the algorithm is less capable to identify errors in the predictors. Our intuition is that xgboost can approximate the underlying function despite the added noise to 'Allocated collateral value', because it finds a structure using other variables.

Table 9. Feature importance of the baseline model and the model with random variable synthetic errors in the second predictor

Baseline model				Peer model with synthetic errors in the predictor			
Feature	Gain	Cover	Frequency	Feature	Gain	Cover	Frequency
Loan amount (in HUF)	0,32	0,13	0,07	Loan amount (in HUF)	0,32	0,15	0,07
Allocated collateral value (at loan origination)	0,25	0,07	0,09	Allocated collateral value (at loan origination)	0,23	0,08	0,09
Full collateral market value at loan origination	0,12	0,06	0,06	Full collateral market value at loan origination	0,13	0,07	0,07
Collateral value	0,05	0,03	0,04	Collateral value	0,05	0,03	0,04
Monthly repayment amount	0,04	0,07	0,06	Monthly repayment amount	0,03	0,06	0,05
Effective interest rate	0,03	0,02	0,03	Effective interest rate	0,03	0,03	0,03
Financed real estate type*	0,02	0,03	0,02	Financed real estate type*	0,02	0,04	0,02
Days past since collateral value determination**	0,02	0,02	0,03	Expected loss	0,02	0,03	0,04
Rate driver*	0,02	0,00	0,01	Days past since collateral value determination**	0,02	0,03	0,04
Expected loss	0,01	0,04	0,04	Original maturity (in days)	0,01	0,04	0,04

* the categorical variable converted to a numerical value using label encoding. While at first it may sound surprising, this kind of treatment works well with the tree-based xgboost-method, and does not increase dimensionality, as opposed to the one-hot-encoding technique.

** The maximum amount, in case of multiple collaterals

Conclusion

In this paper, we analyzed technical aspects of a supervised, gradient boosting-based machine learning method, described by Paulheim and Meusel (2015) and Benatti (2018), to identify potential data errors in a central banking dataset. The algorithm relies on the assumption that most of the data points are correct and there are 'ground truth' relationships between the features. Data points deviating from such a relationship, outliers, are flagged as a potential errors, which are to be investigated further. While the results we found may be specific to the data used, the paper provides a detailed overview of the aspects one may consider during a similar modelling pipeline.

We used the Credit Registry (Hitelregiszter) dataset collected by the Central Bank of Hungary (MNB), which is a collection of datasets provided by supervised entities (banks) to the central bank. More specifically, we looked at the loan-to-value (LTV) part of the data for one business date, carrying out a cross-sectional analysis. In order to test three technical hypotheses relevant during model development, we introduced three types of synthetic errors to the data.

First, we tested the treatment of missing values in the data (when the data is missing both during model training and prediction). We found that an imputation method most often proposed by the literature is not necessarily superior to using a constant value as a placeholder for 'missing' label. This is explained by the flexible nature of the xgboost algorithm. Moreover, one should be careful when using the sparsity-

aware splitting method native to the xgboost algorithm, because it may worsen model performance, particularly when used in combination with the Huber loss function.

Second, we compared two loss functions for xgboost. We found that the Huber loss function captures more errors than the traditional, squared loss function. Squared loss-function can be more sensitive to outliers, and when using it, xgboost is more likely to 'learn' data errors as true relationships than it would be the case with the Huber-function.

Finally, we showed that the algorithm captures errors in the target variable, as opposed to errors in the predictors. The reason for that lies in the flexibility of the xgboost and high dimensionality of our dataset. During model training with errors in the predictors, the 'role' of the erroneous predictor is taken over by others, strongly diminishing error identification performance.

Potential future research questions could be directed towards expanding the cross-sectional nature of our investigation by looking at pattern development over time. Another potential research idea relates to providing better – and automated – answers why an outlier seems to be an error. While individual feature contributions seem to be interpretable options, they are still a far cry from being able to formulate an actionable question to the data provider.

References

Aggarwal C. C. (2017) **Outlier Analysis**. Springer, Second Edition. DOI 10.1007/978-3-319-47578-3

Benatti, N. (2018) A machine learning approach to outlier detection and imputation of missing data. Paper presented at the Ninth IFC Conference on "Are post-crisis statistical initiatives completed?" Basel, 30-31 August 2018. Available at: https://www.bis.org/ifc/publ/ifcb49_48.pdf

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 (KDD '16). Association for Computing Machinery, New York, NY, USA, 785–794. <https://doi.org/10.1145/2939672.2939785>

Daniel J. Stekhoven, Peter Bühlmann, MissForest—non-parametric missing value imputation for mixed-type data, *Bioinformatics*, Volume 28, Issue 1, 1 January 2012, Pages 112–118, <https://doi.org/10.1093/bioinformatics/btr597>

Garrido-Merchán, E. C., Piris, G. G., Vaca, M. C. (2023) Bayesian Optimization of ESG Financial Investments. <https://arxiv.org/pdf/2303.01485.pdf>

Khosravi, P., Vergari A., Choi, J., Liang, Y., Van den Broeck, G. (2020) Handling Missing Data in Decision Trees: A Probabilistic Approach. Paper presented at the conference 'The Art of Learning with Missing Values Workshop at ICML, Vienna, Austria, 2020', available at: <http://starai.cs.ucla.edu/papers/KhosraviArtemiss20.pdf>

Kim, W., Choi, BJ., Hong, EK. et al. A Taxonomy of Dirty Data. *Data Mining and Knowledge Discovery* 7, 81–99 (2003). <https://doi.org/10.1023/A:1021564703268>

Paulheim, H., Meusel, R. (2015) A decomposition of the outlier detection problem into a set of supervised learning problems. *Mach Learn* 100, 509–531 (2015). <https://doi.org/10.1007/s10994-015-5507-y>

Rahm, E., & Do, H. H. (2000). Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4), 3-13.

Sadouk, L., Gadi, T., Essoufi, E.H. (2020). Robust Loss Function for Deep Learning Regression with Outliers. In: Bhateja, V., Satapathy, S., Satori, H. (eds) *Embedded Systems and Artificial Intelligence. Advances in Intelligent Systems and Computing*, vol 1076. Springer, Singapore. https://doi.org/10.1007/978-981-15-0947-6_34

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., De Freites, N. (2016) Taking the Human Out of the Loop: A Review of Bayesian Optimization. In: *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148-175, Jan. 2016, doi: 10.1109/JPROC.2015.2494218.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25.

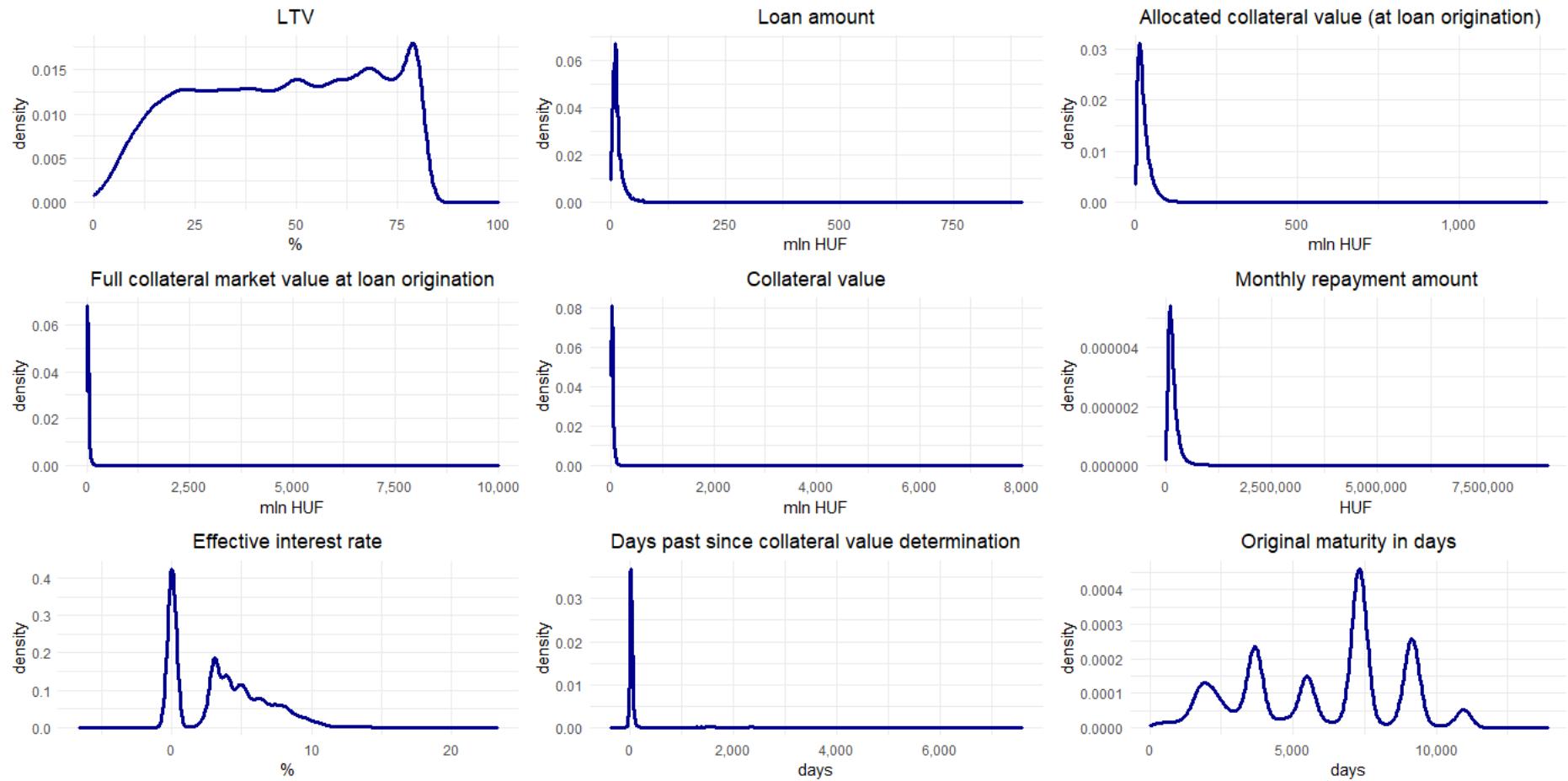
Twala, B. E. T. H.; Jones, M. C. and Hand, D. J. (2008). Good methods for coping with missing data in decision trees. *Pattern Recognition Letters*, 29(7) pp. 950–956.

Appendix

Table 10. Selected summary statistics of the target and the most important explanatory variables by feature importance gain

n = 72 996	Number of missing values	Mean	Standard deviation	Median	Median absolute deviation from the median	Skewness	Kurtosis
LTV	-	47	22	48	28	-0.13	-1.16
Loan amount	-	14 076 346	13 538 794	10 000 000	7 413 000	8	294
Allocated collateral value (at loan origination)	9 732	26 976 614	25 248 320	20 300 000	15 122 520	6	150
Expected loss	2 803	11 527 849	30 627 914	7 835 219	8 584 562	29	1 145
Debt-to-income percentage	549	30	18	30	13	100	18 497
Monthly repayment amount	544	175 902	139 058	143 876	83 582	9	324
Effective interest rate	5 929	4	3	4	5	0.46	-0.43
Full collateral market value at loan origination	8 933	34 025 618	60 486 728	27 000 000	18 532 500	121	19 042
Collateral value	8 933	25 340 908	48 613 488	20 000 000	15 715 560	120	18 711
Original maturity in days	-	5 967	2 681	7 297	2 731	-0.18	-1.05
Days past since collateral value determination**	8 930	204	644	28	22	5	36
Elapsed days since loan origination	0	204,15	150,86	192	127,5	7,82	170,33
APR	64	5,49	2,61	5,49	2,95	2,18	81,03
Birth year of the main debtor	0	1982,11	10,45	1983	10,38	-1,33	6,08
Loan cost percentage	64	6,47	2,85	6,1	2,33	2,26	56,92
Mortgage rate	3842	5,63	2,46	5,2	2,14	1,07	1,1

Figure 7. Density functions for the target variable (LTV) and for the most important explanatory features (by feature importance gain) **Note:** amounts are in HUF



Dr. Csaba Burger, CFA (MNB)
Mihály Berndt (Clarity Consulting)



3rd IFC and Bank of Italy Workshop on “Data Science in Central Banking: Enhancing the access to and sharing of data”

ERROR SPOTTING WITH GRADIENT BOOSTING

Rome, 18th October 2023



The views expressed are those of the authors and do not necessarily reflect the official view of the Central Bank of Hungary (Magyar Nemzeti Bank).



Background

- MNB's commitment to high data quality
- Machine learning is suitable for large data volumes
- The role of ML in data quality checks is not yet standardized

Results

- Un-labelled supervised learning can uncover relationships within the data
- State-of-the-art modelling techniques (XGBoost, Bayesian optimization)
- We present a few recommendations to flag potential data errors

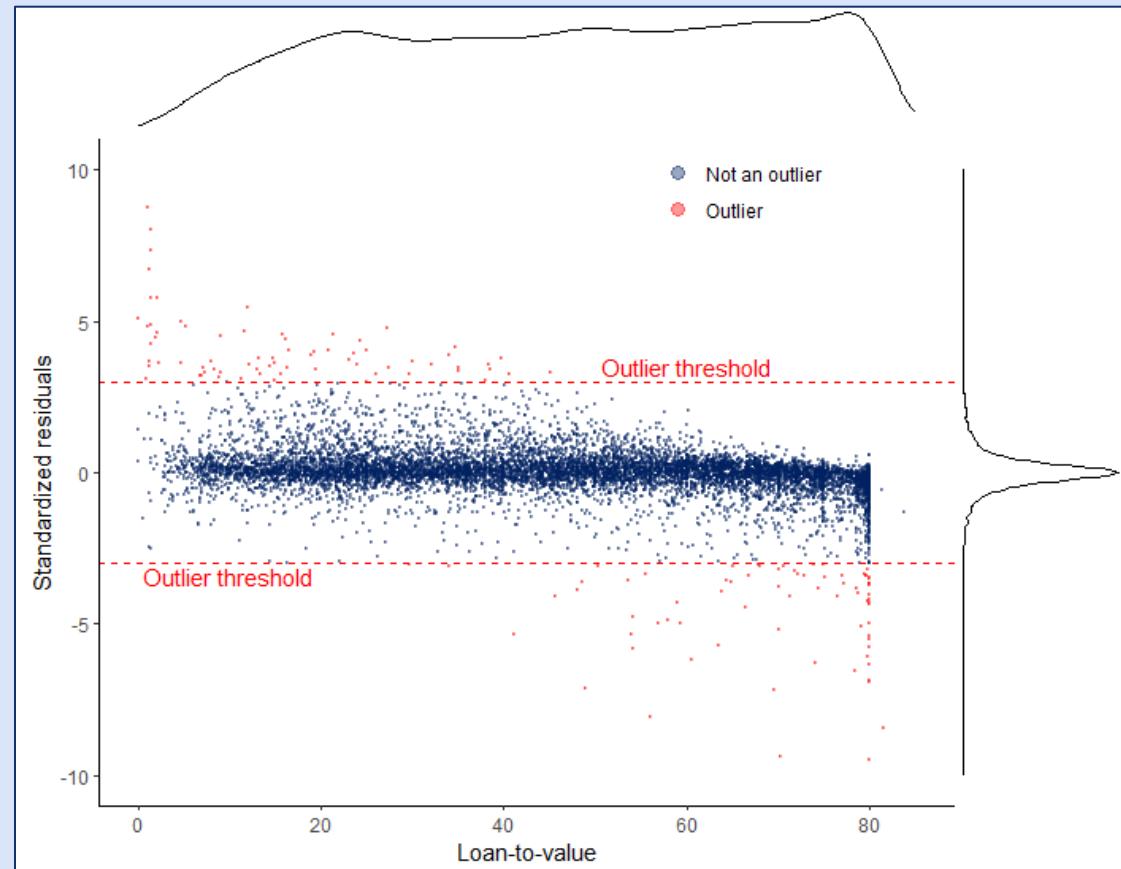
Unlabelled
supervised
methods we use

1 Aggregated time series

2 Cross-sectional - granular

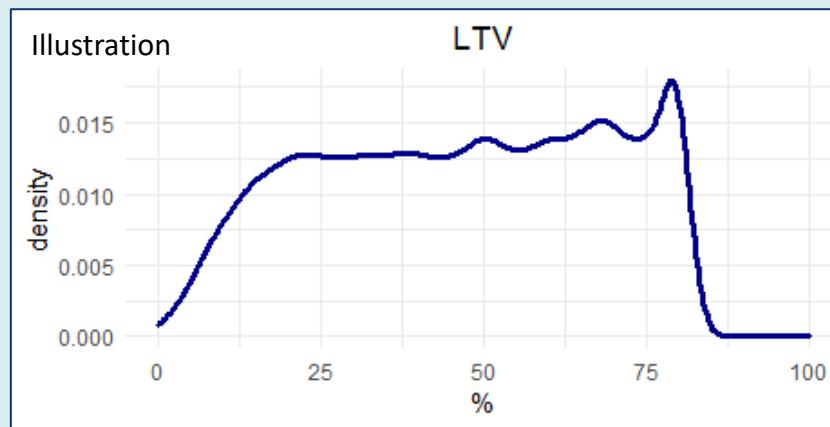
3 Granular time series

Residual plot in a model explaining a selected target variable



MNB LTV report

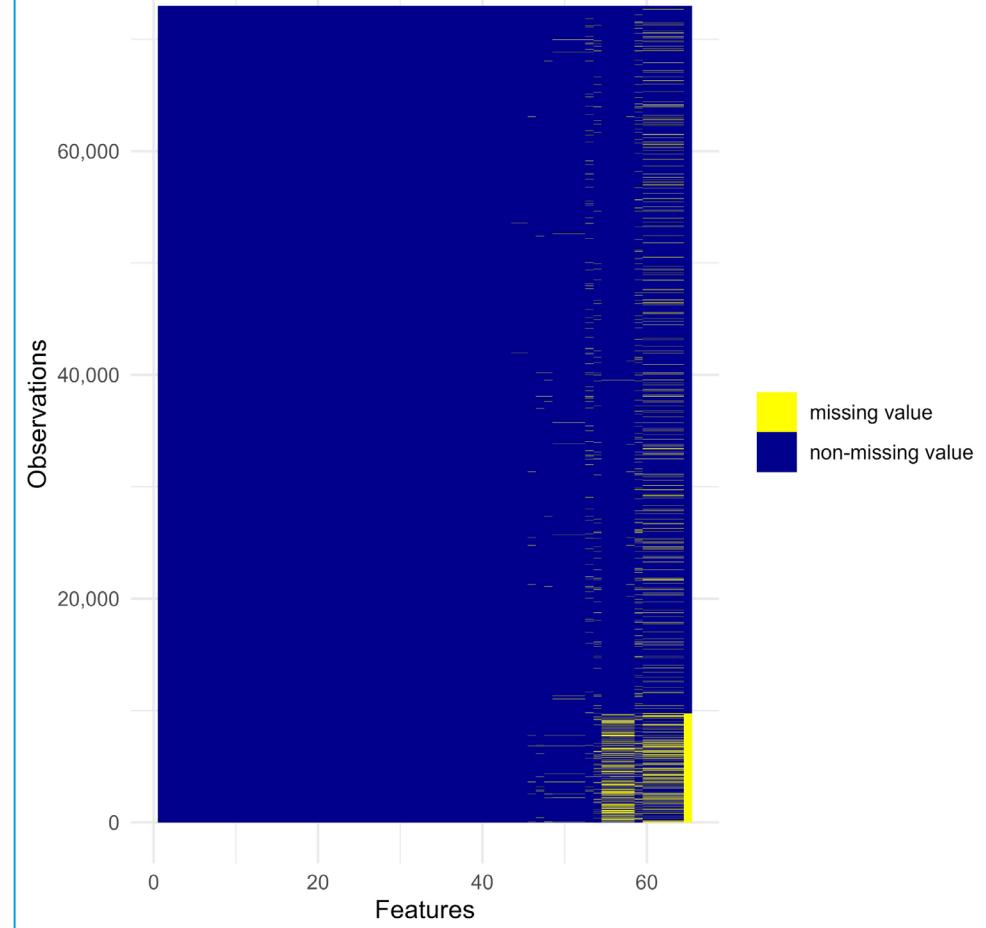
- First ranking mortgages with a start date after 1st Oct 2021
- Approx. 73 thousand lines
- 274 columns → 69 columns (high correlations, missing value share ≥ 20 percent)



Just a theory

$$LTV = \frac{\text{Loan amount}}{\text{Allocated collateral value}}$$

Missing values





Loss reduction
calculation

$$\mathcal{L}_{split} = \frac{1}{2} \left[\frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda} \right] - \gamma.$$

Similarity scores based on:

- residual **direction**
- residual **magnitude**

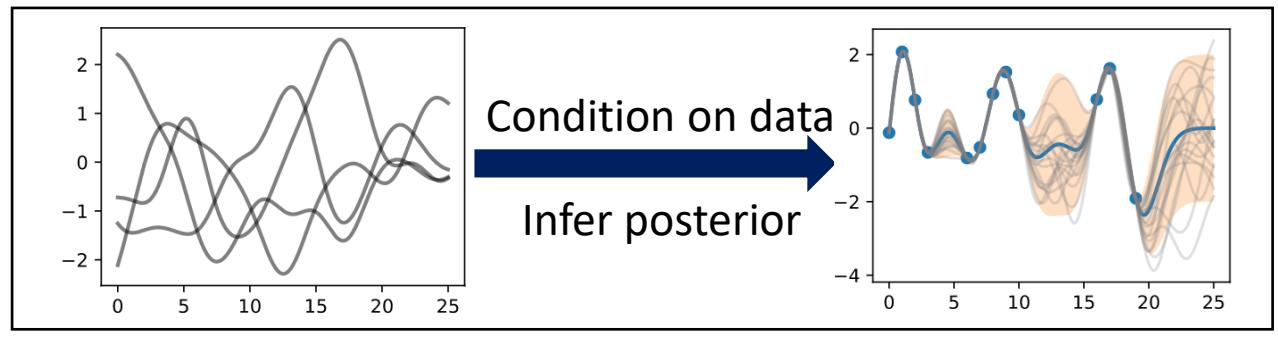
Many hyperparameters to optimize

Sparsity-aware
split finding

1. Visit only non-missing entries
2. Determine the best split and **default direction** for missing value based on the Similarity score above

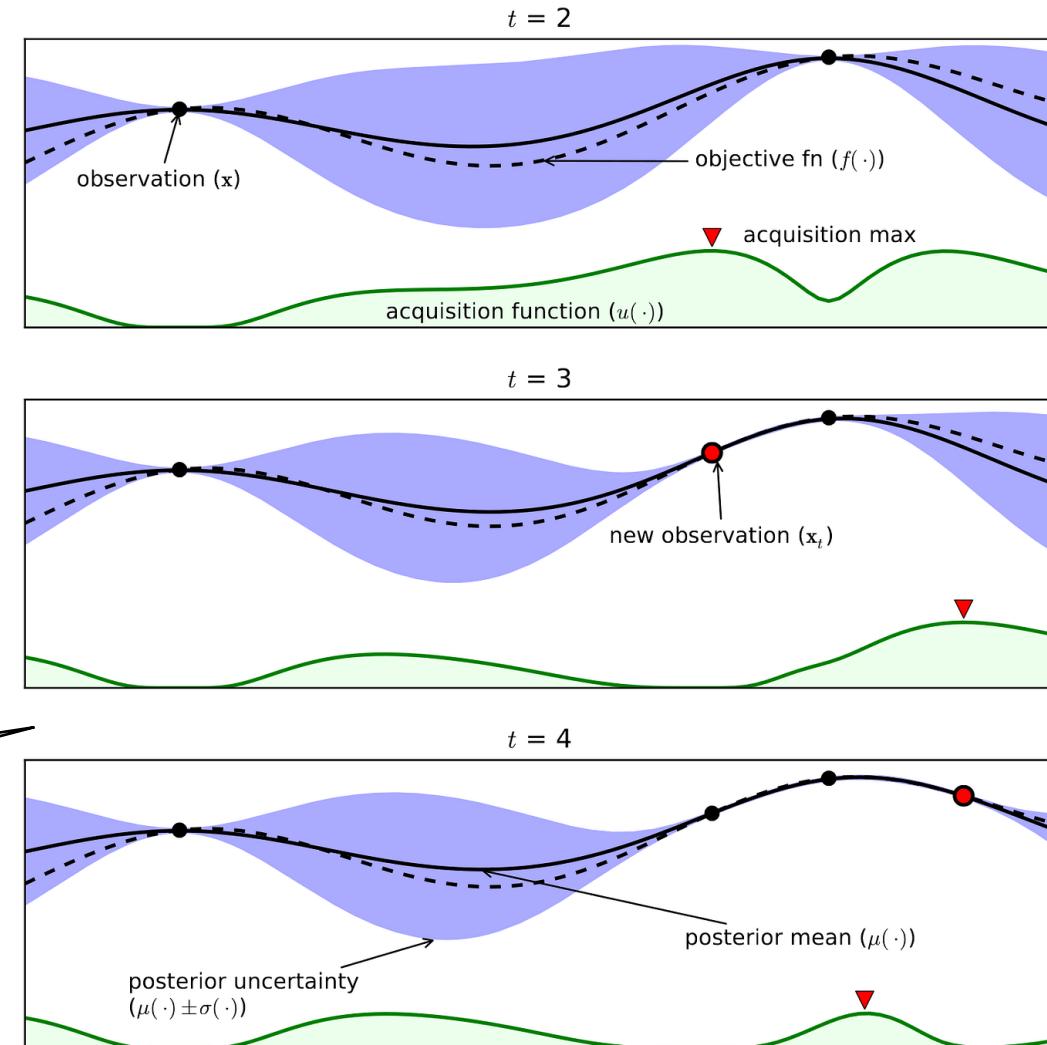
GP: model of the objective function behaviour

- Train and test points are jointly distributed as multivariate normal
- Kernel encodes similarities between data points (shape of the prior)



How to determine new samples?

- Acquisition Function
- Exploration-exploitation trade-off



Treatment of missing values

Treatment of rare values

Determining the loss function

Bayesian optimization

Interpreting the results

Question 1

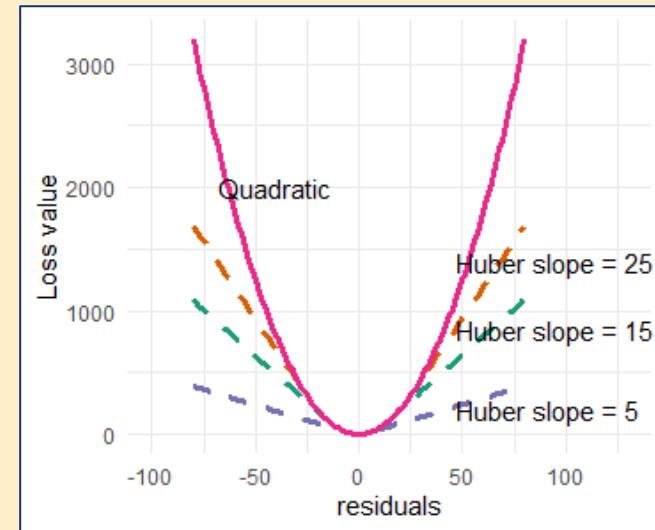
Ways to deal with missing values:

- Estimator (missRanger)
- Constant (unusual dummy)
- Xgboost's sparsity-aware split finding

Question 2

Loss function choice:

- Mitigate the impact of existing errors on finding the ground truth



Question 3

We assume:

- **If:** explanatory column 'B', is not independent from 'A'
- **and** data error distorts an explanatory variable 'A'
- Then B takes over from A

Location	Description
Response variable	Values divided by 100.
Response variable	Values set to 80.
Response variable	Values were multiplied by a random value, drawn for each observation from $U(0.4, 0.6)$ and $U(1.2, 1.4)$
Predictor (2nd most important)	Values set to 10 mln HUF
Predictor (2nd most important)	Values were multiplied by a random value, drawn for each observation from $U(0.4, 0.6)$ and $U(1.2, 1.4)$

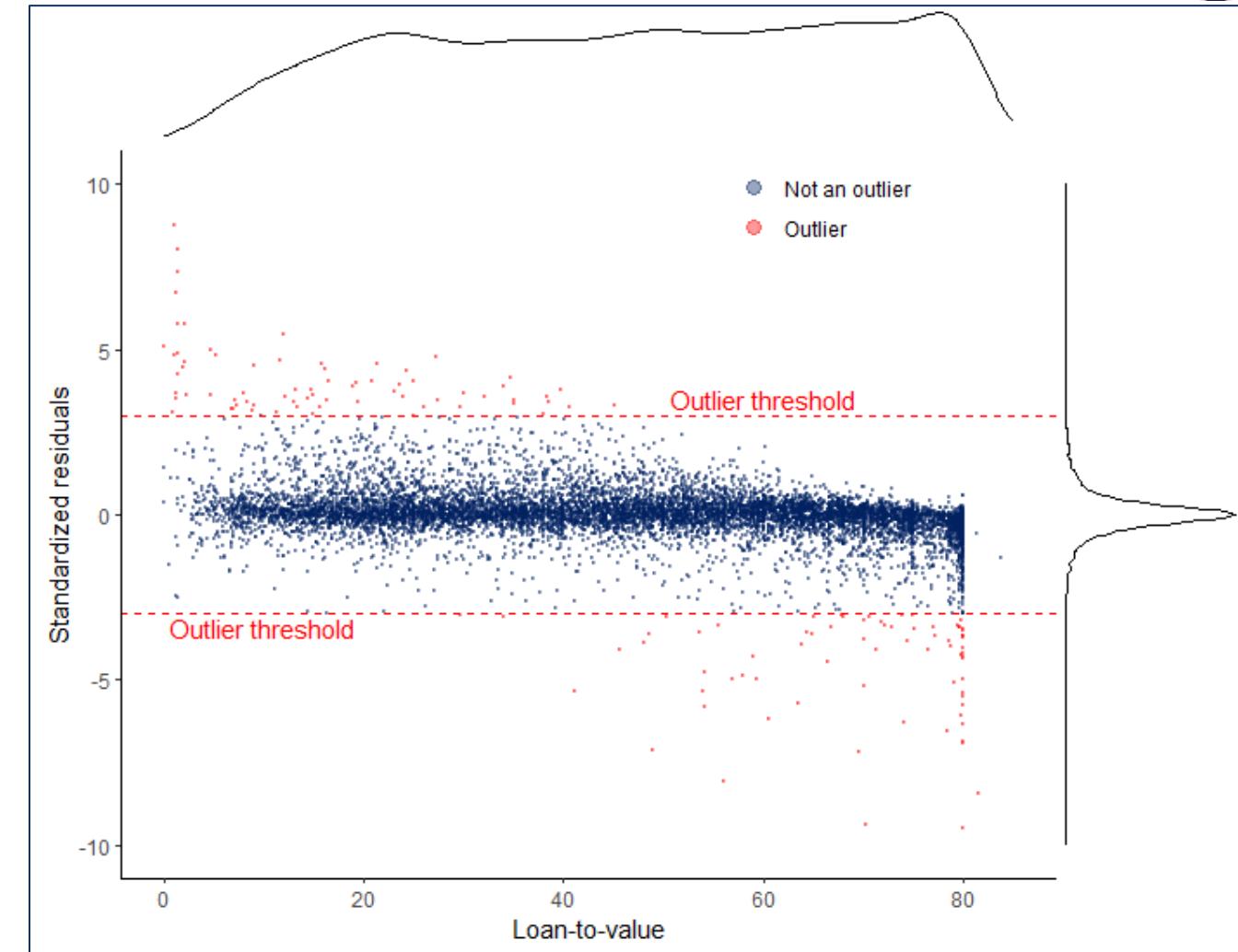
Errors in 5% of all observations, both in train and test sets

The baseline model

- missing values using a constant
- squared loss function
- **no synthetic errors**

Model performance

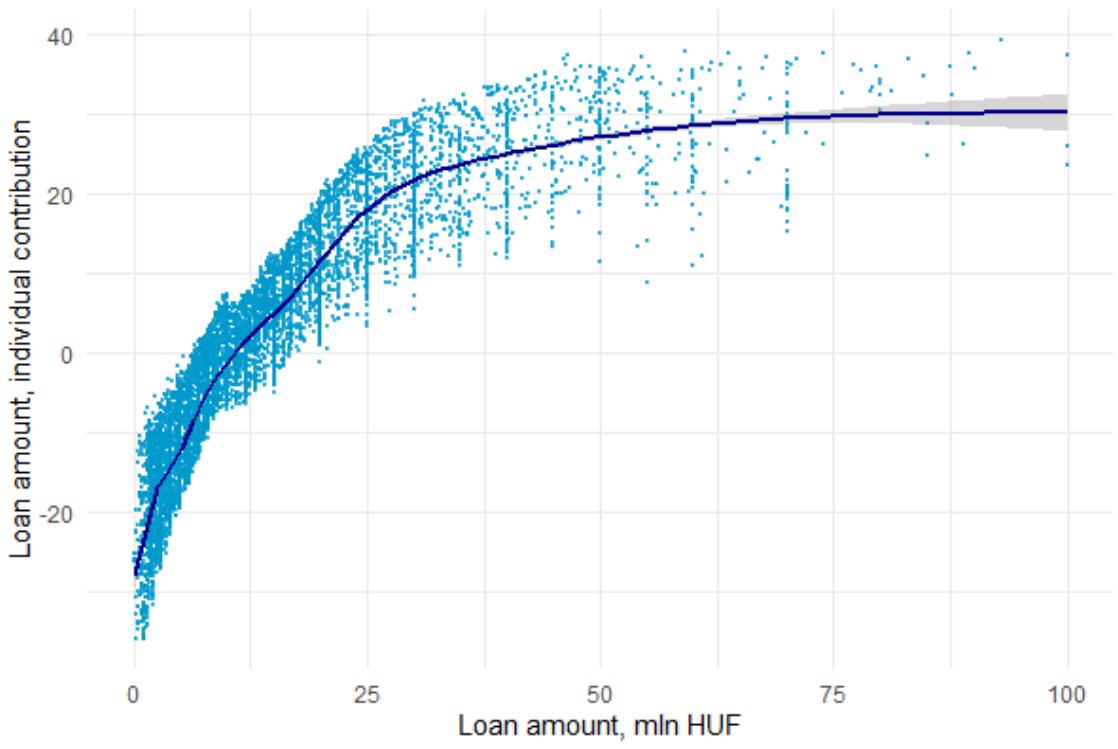
- RMSE = 5.6 percent, MAE = 3.2 percent
- the share of outliers is 1.4 percent only (cutoff of *standardized* residuals of 3)
- The algorithm found intuitive errors (LTV as a fraction between 0 and 1)



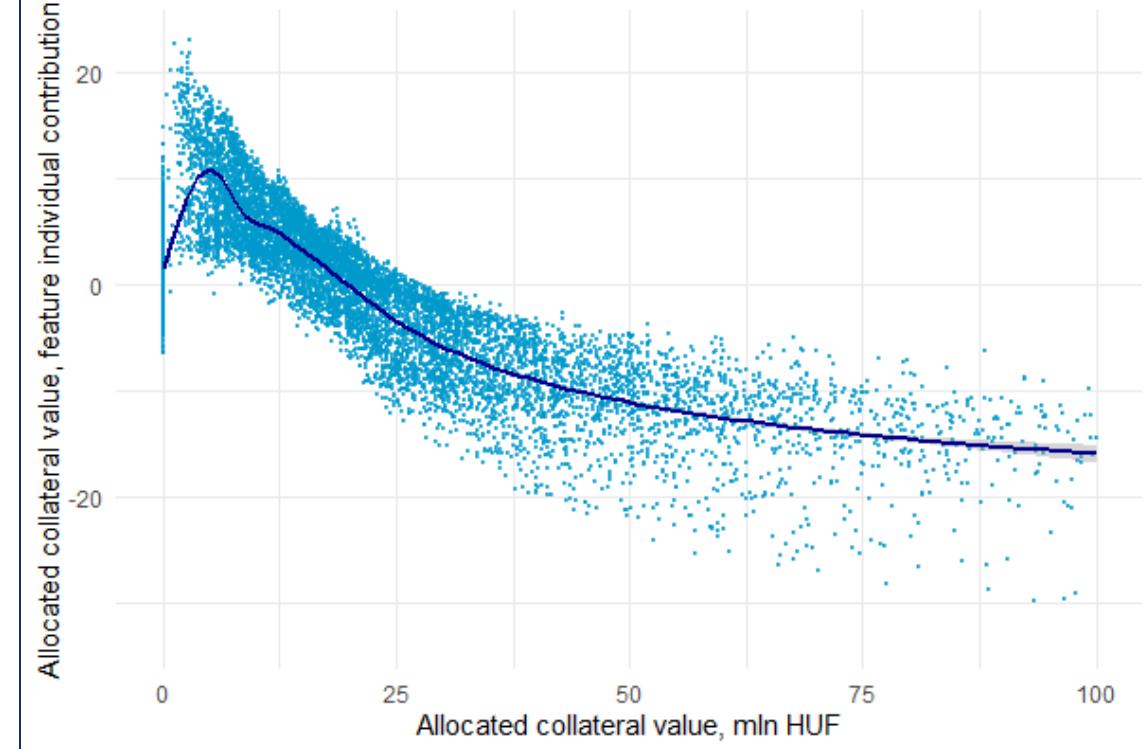
THE BASELINE MODEL – INDIVIDUAL FEATURE CONTRIBUTIONS (IFC)



IFC for Loan amount
+ a LOESS function



IFC for allocated collateral value
+ a LOESS function



$$LTV = \frac{\text{Loan amount}}{\text{Allocated collateral value}}$$

Formula

Share of discovered errors

$$Disc. \text{ error sh.} = \frac{\text{Errors among outliers}}{\text{All errors}}$$

Rationale

Did we find every synthetic error?

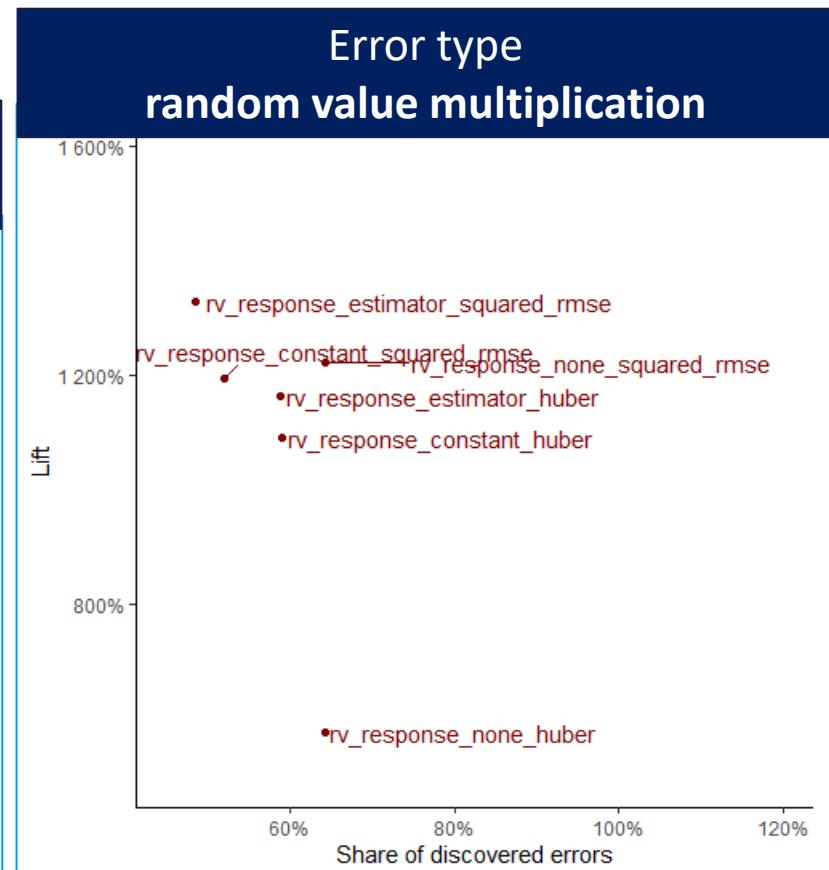
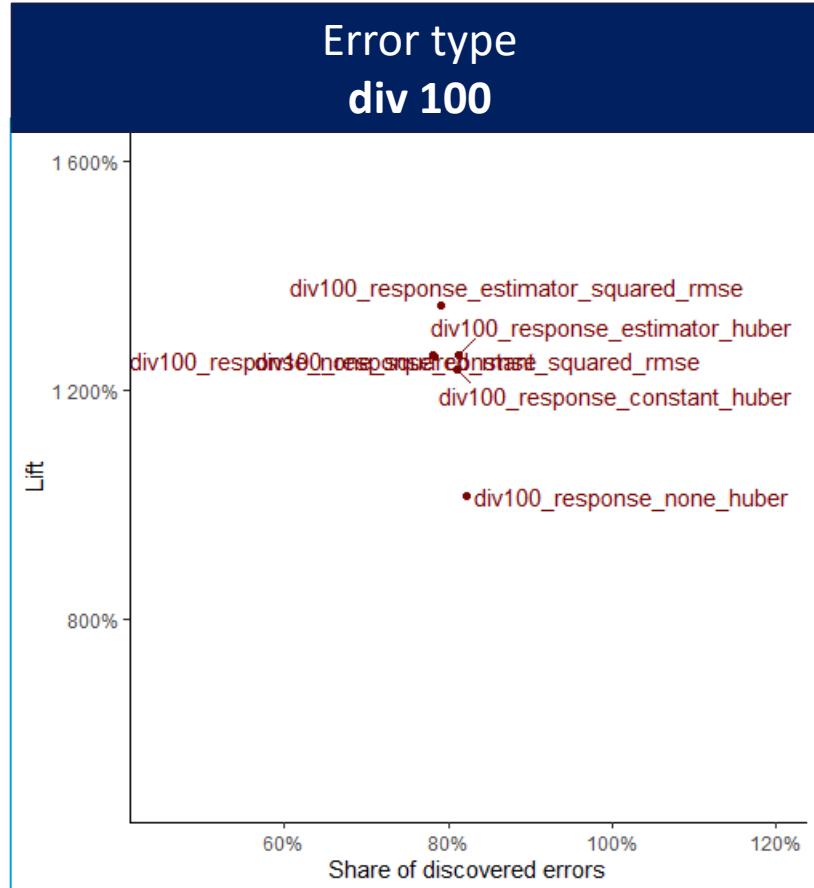
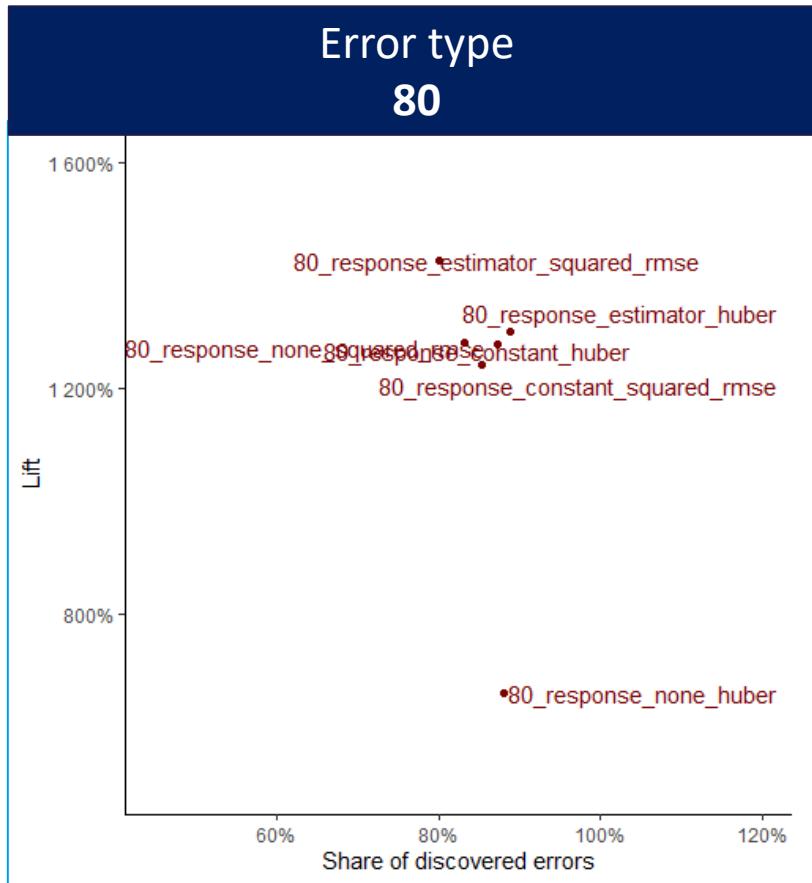
Lift value

$$Lift = \frac{\text{Error share among outliers}}{\text{Error share in all data}}$$

Am I any better off by looking at outliers
than going through the raw data?

One metric is insufficient

HYPOTHESIS 1 (MISSING VALUE REPLACEMENT) AND HYPOTHESIS 2 (LOSS FUNCTION)



HYPOTHESIS 3 – ERROR IN ALLOCATED COLLATERAL VALUE

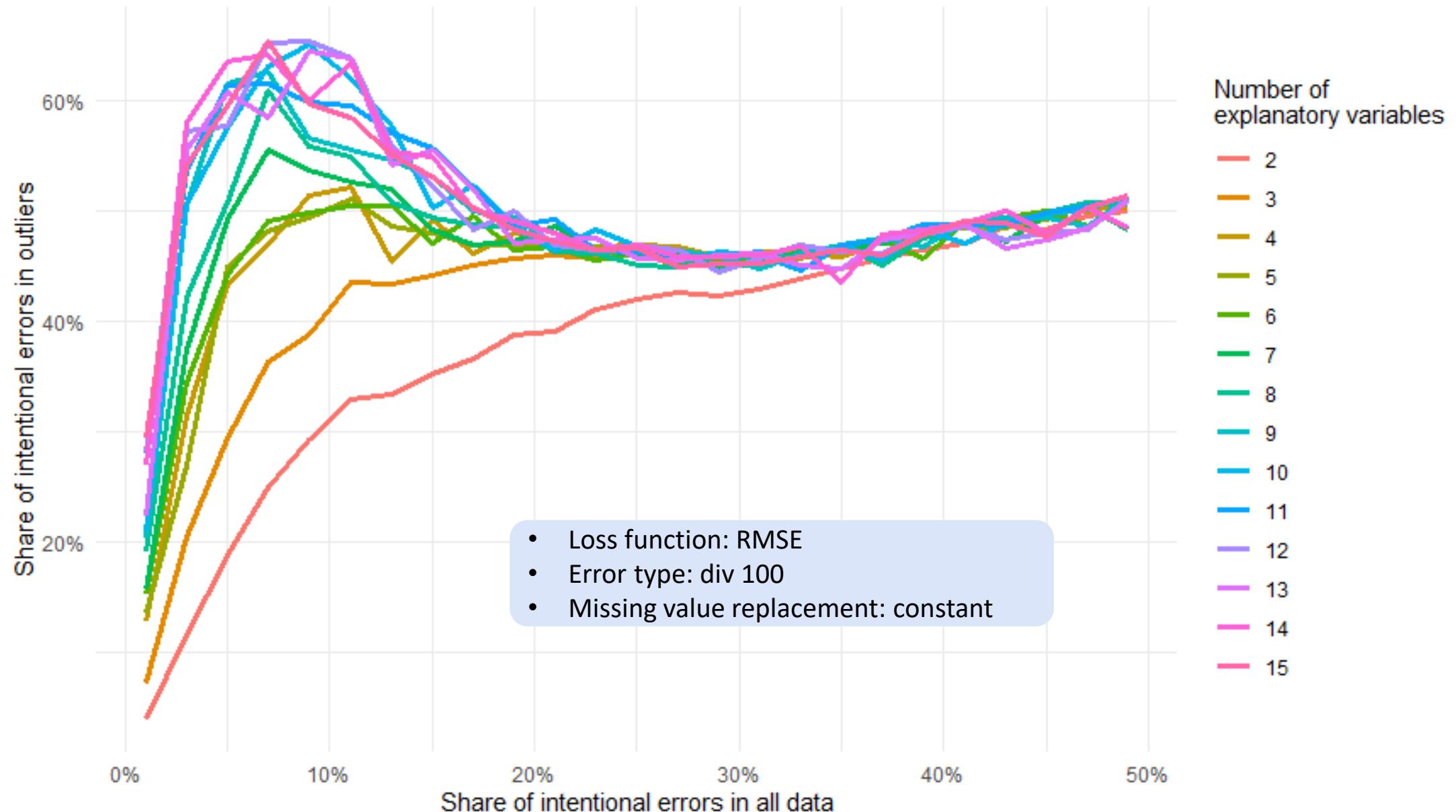


Error type	Missing value replacement	Loss function	Outliers as % of total	Share of discovered errors	Lift
10	none	Huber	3,5%	3,0%	0,87
10	estimator	Huber	3,0%	2,1%	0,70
10	none	rmse	1,5%	1,6%	1,05
10	constant	Huber	2,3%	1,3%	0,57
10	constant	rmse	1,3%	1,1%	0,83
10	estimator	rmse	1,1%	0,9%	0,83
rv	none	rmse	2,0%	10,7%	5,41
rv	estimator	Huber	4,5%	7,9%	1,75
rv	none	Huber	2,5%	5,8%	2,31
rv	constant	Huber	3,0%	4,3%	1,43
rv	constant	rmse	1,5%	2,1%	1,41
rv	estimator	rmse	1,3%	1,9%	1,51

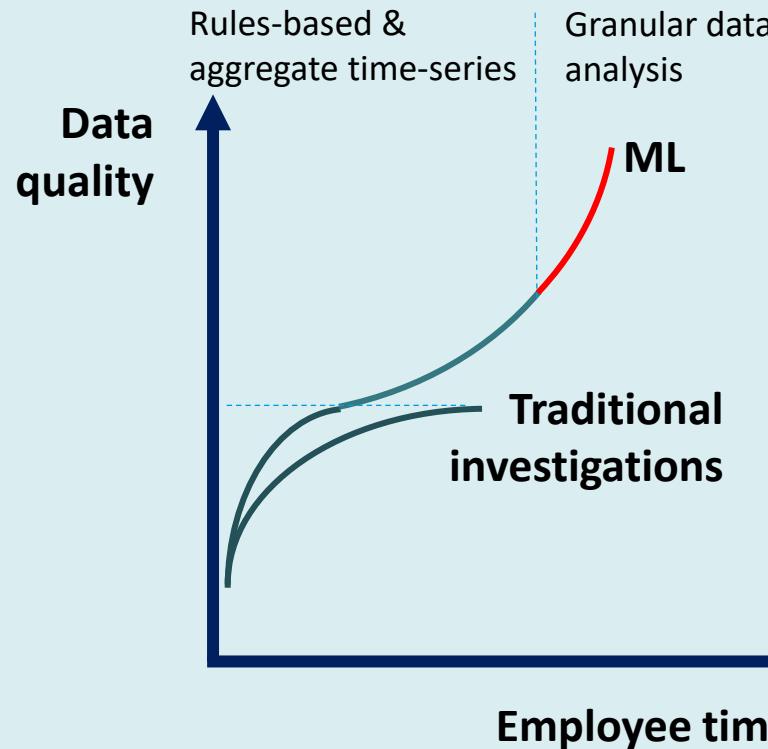
Vs. 70-80 % when
error in target

Vs. 10-12 when
error in target

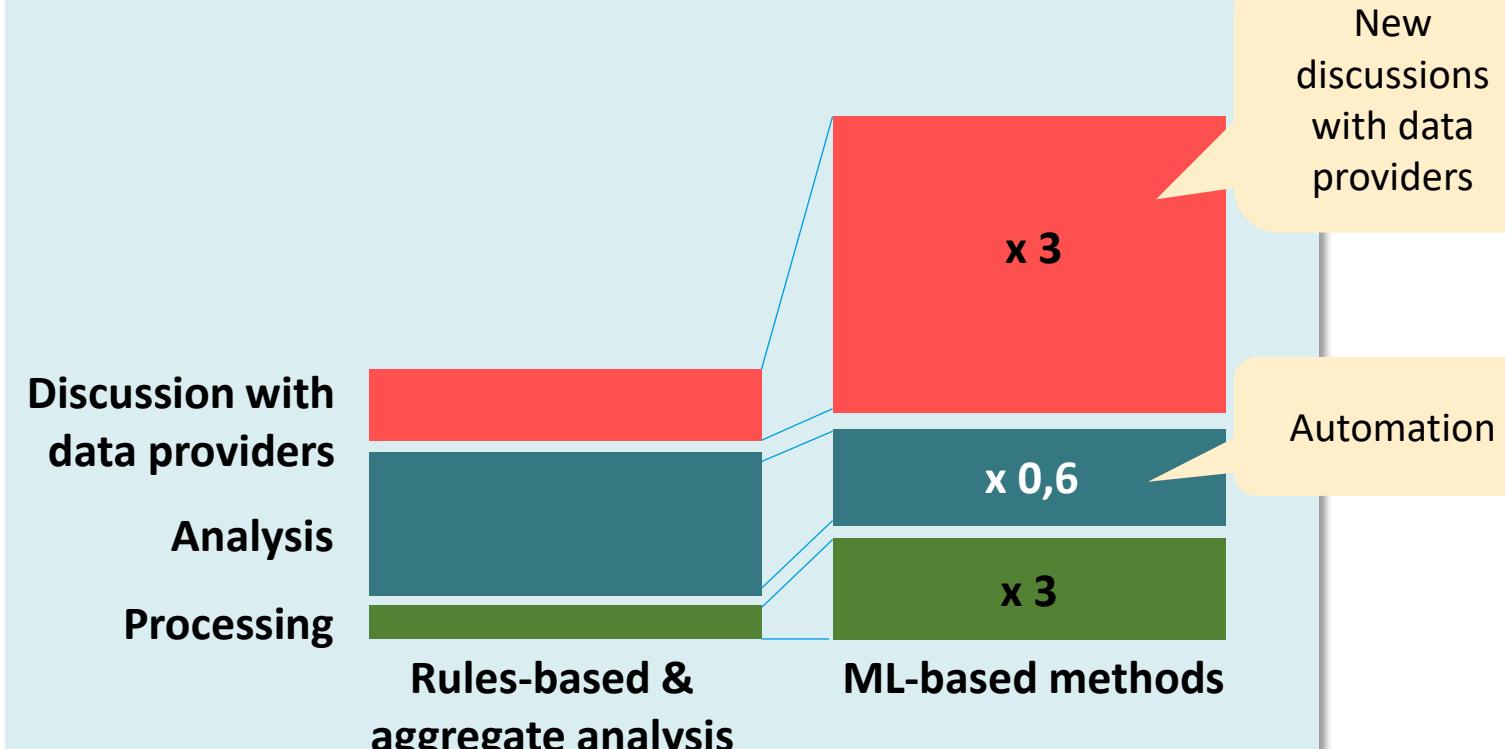
ABOVE NINE EXPLANATORY FEATURES AND AN ERROR SHARE OF AROUND 10 PERCENT EFFICIENCY STARTS TO DROP



Data quality explodes but only if you work with it



Required human labour



Findings recap

- A supervised learning algorithm to flag potential data errors
- The method successfully identifies synthetic errors
- It provides hints to their location
- We also analysed various steps during the preprocessing phase (missing values and loss function) which may improve performance

Implications

- Our results helps the data providers
- The 'last mile problem' is still there: error flags do not provide interpretation
- Our results help modellers: model predictions may be used instead of actual values

Thank you for your attention!